Proyecto Fin de Carrera

Gestión de la Productividad en la Delegación del Gobierno en la Junta de Andalucía

Plan de Trabajo

Eduardo Gutiérrez Martos

Consultor: Javier Ferro García

Dedicatoria y Agradecimientos

Este documento es la culminación de una etapa muy importante para mi. Por este motivo, quiero agradecer sobre todo a mi mujer y a mis hijos por todos los ánimos que me han dado y disculparme por todas las horas que no les he podido dedicar mientras estudiaba.

A mi madre y mi suegra por cuidar de los niños para que yo pudiese terminar este proyecto.

A mis compañeros de trabajo, por aguantarme y quitarme todo el trabajo posible para que yo pudiese aprovechar mi tiempo, ya no sólo con el proyecto si no durante todos los semestres que he necesitado para terminar la carrera.

Y no puedo olvidar a los distintos consultores y a los tutores, que han hecho esta travesía un poco más sencilla.

A todos ellos, muchas gracias.

INDICE

1.- Introducción

1.1.- Situación Actual

Actualmente, en las Delegaciones Provinciales de la Junta de Andalucía, existe un programa que se utiliza para realizar las nóminas mensuales de todos los funcionarios. Sin embargo, existe un concepto, Productividad, que no está recogido al 100% en esta aplicación.

La productividad, es una retribución que se paga a los funcionarios cada cuatrimestre según una serie de valoraciones realizadas por su superior jerárquico.

Cada cuatrimestre se libra un dinero concreto para repartir entre los funcionarios, pero como este reparto no es igual para todos, ya que depende de una serie de factores que son valorados por el superior jerárquico (normalmente el jefe de servicio).

Actualmente, existen cinco factores valorables, que además tienen pesos distintos. Son los siguientes:

- Calidad: 30%
- Iniciativa: 20%
- Asistencia: 10%
- Disponibilidad: 25%
- Formación: 15%

Para poder cargar la Productividad, en la aplicación de nóminas, hay que crear un fichero de texto, con un formato determinado.

Actualmente, la Delegación del Gobierno en Almería, dispone de una hoja de cálculo en Excel, que realiza los cálculos de las cantidades que cada funcionario debe percibir, y crea el fichero de texto para la carga en la aplicación de nóminas. El histórico que se lleva es la copia de cada uno de los libros de Excel por cuatrimestres.

1.2.- Motivación

Hay varios motivos por el que el desarrollo de este proyecto me resulta muy interesante.

En primer lugar un motivo profesional, ya que la Junta de Andalucía, dispone de pocas licencias de Microsoft Excel, por lo que no debemos extender su uso. Además, la propia Junta de Andalucía, esta haciendo una gran apuesta por el software libre, como demuestra el impulso que está dando a la versión de Linux, Guadalinex.

Debido a esto, y haciendo un estudio de las principales plataformas existentes en la actualidad para el desarrollo de aplicaciones distribuidas, como son CORBA, .NET y

JEE, creo que la mejor solución para empresas que quieran basar su arquitectura en productos basados en software libre es JEE, que nos ofrece entre otras las siguientes ventajas:

- Soporte de múltiples sistemas operativos: Al ser una plataforma basada en el lenguaje Java, es posible desarrollar arquitecturas basadas en JEE utilizando cualquier sistema operativo donde se pueda ejecutar una máquina virtual Java.
- Organismo de control: La plataforma JEE está controlada por el JCP, un organismo formado por más de 500 empresas. Entre las empresas que lo forman están todas las más importantes del mundo informático (SUN, IBM, Oracle, SAP, HP, AOL, etc.) lo que garantiza la evolución de la misma.
- Competitividad: Muchas empresas crean soluciones basadas en J2EE y ofrecen características como rendimiento, precio, etc., muy diferentes. De este modo el cliente tiene una gran cantidad de opciones a elegir.
- Madurez: Creada en el año 1997 como respuesta a la tecnología MTS de Microsoft, J2EE tiene ya nueve años de vida y una gran cantidad de proyectos importantes a sus espaldas.
- Soluciones libres: En la plataforma JEE es posible crear arquitecturas completas basadas única y exclusivamente en productos de software libre. No sólo eso, sino que los arquitectos normalmente disponen de varias soluciones libres para cada una de las partes de su arquitectura.

Desde el punto de vista académico, aunque ya he realizado algunos pinitos con la tecnología JEE, en la asignatura Ingeniería del Software y Componentes Distribuidos, me gustaría seguir practicando con ella, y sobre todo conocer otras alternativas y Frameworks disponibles en la actualidad. Este proyecto me puede servir, entre otras cosas, para conocer en profundidad las tecnologías actuales, y poder compararlas entre sí para poder elegir entre ellas cuál se adapta mejor a mi solución.

1.3.- Definición General del Proyecto

En este trabajo se pretende que el alumno lleve a cabo un proyecto desde su inicio hasta el final, pasando por todas las fases de desarrollo del mismo.
Más concretamente, trabajaré en el análisis, diseño e implementación de un sistema de gestión de la productividad en centros oficiales de la Junta de Andalucía.

Este sistema, dará la posibilidad a los funcionarios de una determinada Delegación, de conocer su productividad vía web, a los jefes de Servicio introducir las evaluaciones de los empleados a su cargo sin necesidad de reunión previa, y al jefe de personal la de validar la productividad y exportar el fichero resultado a la aplicación de Nóminas.
Para poder realizar el proyecto, debemos hacer un análisis de la seguridad, mantenibilidad, facilidad de programación, y, en definitiva, estudiar que ventajas e inconvenientes aporta este nuevo sistema de gestión de la productividad que estará basado en la tecnología JEE.

1.4.- Objetivos

Este Proyecto Fin de Carrera, tiene como objetivo principal, la realización del análisis, diseño e implementación de un sistema de gestión y creación de la denominada "Productividad" en los Organismos Oficiales de la Junta de Andalucía.
No debemos olvidar que para conseguir este objetivo, deberemos realizar previamente un análisis en detalle de ventajas no funcionales de las tecnologías seleccionadas estudiando entre otros aspectos, la eficiencia, seguridad, facilidad de mantenimiento y facilidad de programación.

1.5.- Producto final y resultados esperados

Al final del proyecto se obtendrá un estudio de la tecnología existente, con sus ventajas e inconvenientes, y una aplicación basada en la arquitectura JEE, que permitirá gestionar la productividad en la Delegaciones Provincial de Almería, de la Consejería de Gobernación de la Junta de Andalucía.

2.- Planificación.

2.1.- Recursos

Los recursos humanos se limitan a una sóla persona, con una dedicación aproximada de 110 horas.
En cuanto a los recursos físicos, dispondré de:

- un portátil Toshiba Tecra con las características siguientes:
    - HD: 80 GB.
    - Velocidad: 1,8 GHz.
    - Memoria RAM: 1 GB.

- y un PC de sobremesa cuyas características principales son:
    - HD: 80 GB.
    - Velocidad: 3 GHz.
    - Memoria RAM: 1 GB.

2.2.- Requerimientos de Software

El software que utilizaré para el desarrollo del proyecto será:

- JDK 1.6.0
- Jboss 4.0.4. GA
- MySQL Server 5.0
- NetBeans 5.5 como interfaz de desarrollo

2.3.- Riesgos del proyecto

El riesgo más importante, es mi poca experiencia en la implementación de aplicaciones JEE, y el poco conocimiento de herramientas como Struts, Hibernate, EJB3, ... por lo que antes de la elección de una u otra tecnología, habrá que realizar un análisis exhaustivo de cada una de ellas, viendo que ventajas e inconvenientes nos aportan.

Esta inexperiencia y falta de conocimiento, puede dar lugar a una ralentización en el desarrollo de la implementación.

Una posible forma de mitigar este riesgo, es dedicar, desde el principio del proyecto, tiempo para la formación y práctica con estas herramientas.

2.4.- Fases del Proyecto

El comienzo del proyecto tiene lugar el día 19 de Septiembre de 2007, y debe estar concluido para el día 14 de Enero de 2008, coincidiendo con la entrega final, donde se presentará la Memoria, el aplicativo, y una Presentación Virtual del proyecto.

Se prevé realizar tres entregas parciales:

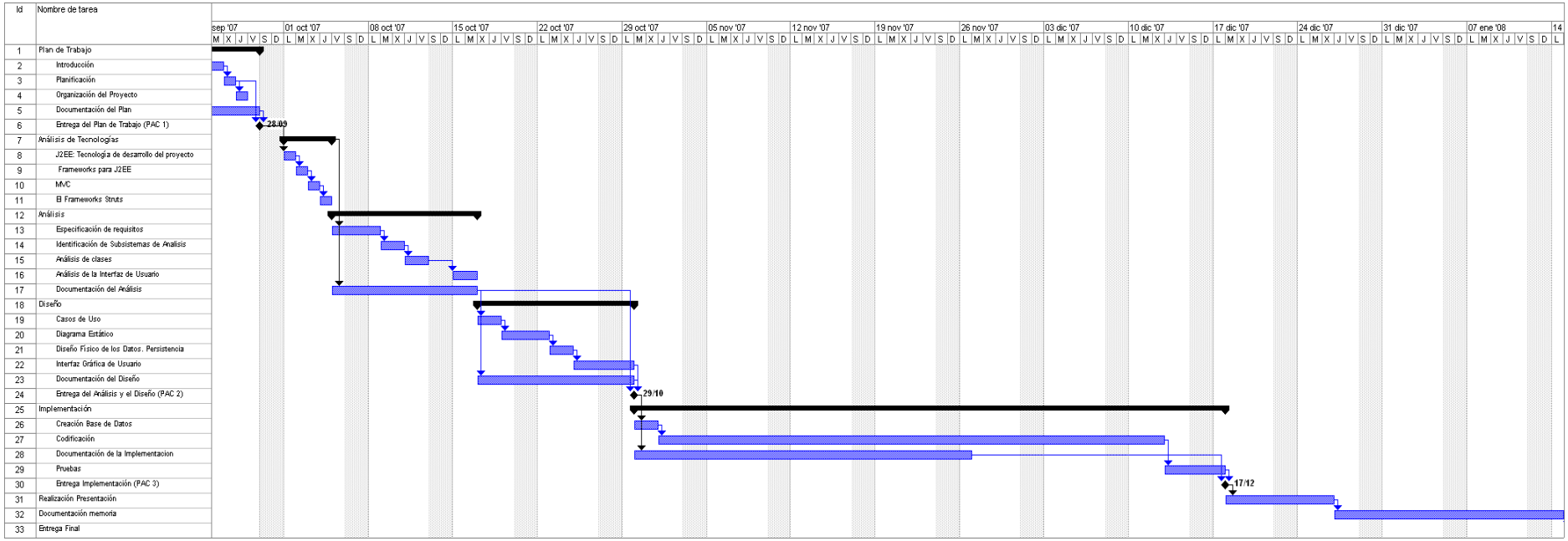
· Plan de Trabajo (PAC 1): 28/09/2007

· Análisis y Diseño (PAC 2): 29/10/2007

· Implementación (PAC 3): 17/12/2007

2.5.- Calendarización

A continuación se muestra un diagrama de Gantt con la temporalización de las distintas actividades:



3.- Organización del Proyecto

3.1.- Descomposición en Actividades

Como podemos observar en el Diagrama de Gantt, hemos dividido el proyecto en una serie de actividades que se muestran a continuación:

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Plan de Trabajo**	**4 días**	**mar 25/09/07**	**vie 28/09/07**	
2	Introducción	1 día	mar 25/09/07	mar 25/09/07	
3	Planificación	1 día	mié 26/09/07	mié 26/09/07	2
4	Organización del Proyecto	1 día	jue 27/09/07	jue 27/09/07	3
5	Documentación del Plan	4 días	mar 25/09/07	vie 28/09/07	
6	Entrega del Plan de Trabajo (PAC 1)	0 días	vie 28/09/07	vie 28/09/07	3;5
7	**Análisis de Tecnologías**	**4 días**	**lun 01/10/07**	**jue 04/10/07**	
8	J2EE: Tecnología de desarrollo del proyecto	1 día	lun 01/10/07	lun 01/10/07	6
9	Frameworks para J2EE	1 día	mar 02/10/07	mar 02/10/07	8
10	MVC	1 día	mié 03/10/07	mié 03/10/07	9
11	El Frameworks Struts	1 día	jue 04/10/07	jue 04/10/07	10
12	**Análisis**	**8 días**	**vie 05/10/07**	**mar 16/10/07**	
13	Especificación de requisitos	2 días	vie 05/10/07	lun 08/10/07	7
14	Identificación de Subsistemas de Analisis	2 días	mar 09/10/07	mié 10/10/07	13
15	Análisis de clases	2 días	jue 11/10/07	vie 12/10/07	14
16	Análisis de la Interfaz de Usuario	2 días	lun 15/10/07	mar 16/10/07	15
17	Documentación del Análisis	8 días	vie 05/10/07	mar 16/10/07	7
18	**Diseño**	**9 días**	**mié 17/10/07**	**lun 29/10/07**	
19	Casos de Uso	2 días	mié 17/10/07	jue 18/10/07	17
20	Diagrama Estático	2 días	vie 19/10/07	lun 22/10/07	19
21	Diseño Físico de los Datos. Persistencia	2 días	mar 23/10/07	mié 24/10/07	20
22	Interfaz Gráfica de Usuario	3 días	jue 25/10/07	lun 29/10/07	21
23	Documentación del Diseño	9 días	mié 17/10/07	lun 29/10/07	17
24	Entrega del Análisis y el Diseño (PAC 2)	0 días	lun 29/10/07	lun 29/10/07	22;23;17
25	**Implementación**	**35 días**	**mar 30/10/07**	**lun 17/12/07**	
26	Creación Base de Datos	2 días	mar 30/10/07	mié 31/10/07	24
27	Codificación	30 días	jue 01/11/07	mié 12/12/07	26
28	Documentación de la Implementacion	20 días	mar 30/10/07	lun 26/11/07	24
29	Pruebas	3 días	jue 13/12/07	lun 17/12/07	27
30	Entrega Implementación (PAC 3)	0 días	lun 17/12/07	lun 17/12/07	29;28
31	Realización Presentación	7 días	mar 18/12/07	mié 26/12/07	30
32	Documentación memoria	13 días	jue 27/12/07	lun 14/01/08	31
33	Entrega Final	0 días	lun 14/01/08	lun 14/01/08	32

3.2.- Hitos principales

Los principales hitos de este proyecto, coinciden con las entregas a realizar, y son los siguientes:

- Plan de Trabajo (PAC 1): 28/09/2007
- Análisis y Diseño (PAC 2): 29/10/2007
- Implementación (PAC 3): 17/12/2007
- Entrega final (PAC 3): 14/01/2008

4.- Análisis de Tecnologías.

4.1.- Comparación de tecnologías existentes

4.1.1.- CORBA

En un sentido general CORBA "envuelve" el código escrito en otro lenguaje en un paquete que contiene información adicional sobre las capacidades del código que contiene, y sobre cómo llamar a sus métodos. Los objetos que resultan pueden entonces ser invocados desde otro programa (u objeto CORBA) desde la red. En este sentido CORBA se puede considerar como un formato de documentación legible por la máquina, similar a un archivo de cabeceras pero con más información.

CORBA utiliza un lenguaje de definición de interfaces (IDL) para especificar los interfaces con los servicios que los objetos ofrecerán. Implementaciones estándar existen para Ada, C, C++, Smalltalk, Java y Python. Hay también implementaciones para Perl y TCL.

Al compilar una interfaz en IDL se genera código para el cliente y el servidor (el implementador del objeto). El código del cliente sirve para poder realizar las llamadas a métodos remotos. Es el conocido como stub, el cual incluye un proxy (representante) del objeto remoto en el lado del cliente. El código generado para el servidor consiste en unos skeletons (esqueletos) que el desarrollador tiene que rellenar para implementar los métodos del objeto.

Las ventajas que ofrece CORBA son muy importantes:

- Soporte de múltiples sistemas operativos
- Soporte de múltiples lenguajes
- Gran cantidad de servicios : mensajería, eventos, transacciones, persistencia, etc.
- Está controlada por un organismo serio, OMG

A pesar de esto, CORBA arrastra unos problemas que suponen un verdadero lastre:

- Complejidad: CORBA es una plataforma de desarrollo muy compleja, aunque existen capas de abstracción que facilitan el desarrollo de aplicaciones.
- Burocracia: La evolución de las especificaciones de CORBA está sujeta a demasiados pasos de burocracia, lo que origina en que para ver novedades en la plataforma sea necesario esperar grandes cantidades de tiempo
- Pocas soluciones libres: Como consecuencia de su complejidad y de la lentitud de su evolución se deriva que existen pocas soluciones libres. OpenORB es un ejemplo y existen algunos otros pero la cantidad no es demasiado elevada.

4.1.2.- .NET

La plataforma de desarrollo empresarial de Microsoft ofrece a los desarrolladores algunas ventajas interesantes.

- Soporte de múltiples sistemas lenguajes: con .NET es posible desarrollar aplicaciones utilizando simultáneamente varios lenguajes de programación.
- Ideal para entornos Microsoft: Si en nuestra empresa disponemos de gran cantidad de software y hardware dependiente de Microsoft, probablemente la mejor opción.
- Visual Studio .NET: La plataforma .NET dispone de esta gran herramienta que además de su potencia ofrece un entorno homogéneo de desarrollo.
- Requiere desarrolladores poco experimentados: Bajo la plataforma de desarrollo de Microsoft es posible utilizar lenguajes como VB .NET que hacen muy sencilla la creación de aplicaciones empresariales. De este modo es posible tener un equipo de desarrolladores poco experimentados y sin embargo que éstos puedan crear fácilmente aplicaciones.

Aún así, si la lista de ventajas es bastante grande, la lista de desventajas no le tiene nada que envidiar:

- No soporta múltiples sistemas operativos: El mundo de .NET gira en torno al sistema operativo Windows y aunque se están intentando trasladar partes importantes de la plataforma , como la CLR o C#, a otros sistemas operativos, lo cierto es que estas partes forman una parte ínfima de la totalidad de la plataforma de Microsoft.
- Un único dueño: La plataforma .NET está dominada única y exclusivamente por Microsoft. Esto supone un grave problema ya que es una única empresa la que puede añadir y quitar características según crea necesario. Además esto hace que la competencia sea nula y no se estimula la evolución de la plataforma.
- Pocas soluciones libres: No existe una correspondencia exacta entre las partes de la plataforma .NET y soluciones libres. Existen proyectos como Mono o dotGNU que están portando algunas de sus partes, aún así, no se puede crear una arquitectura completa utilizando solamente productos basados en software libre.

4.1.3.- JEE

JEE, la plataforma creada por SUN en el año 1997 es la que ofrece mejores perspectivas de desarrollo para empresas que quieran basar su arquitectura en productos basados en software libre. JEE, nos ofrece entre otras las siguientes ventajas:

- Soporte de múltiples sistemas operativos: Al ser una plataforma basada en el lenguaje Java, es posible desarrollar arquitecturas basadas en JEE utilizando cualquier sistema operativo donde se pueda ejecutar una máquina virtual Java.
- Organismo de control: La plataforma JEE está controlada por el JCP, un organismo formado por más de 500 empresas. Entre las empresas que lo forman están todas las más importantes del mundo informático (SUN, IBM, Oracle, SAP, HP, AOL, etc.) lo que garantiza la evolución de la misma.
- Competitividad: Muchas empresas crean soluciones basadas en JEE y que ofrecen características como rendimiento, precio, etc., muy diferentes. De este modo el cliente tiene una gran cantidad de opciones a elegir.
- Madurez: Creada en el año 1997 como respuesta a la tecnología MTS de Microsoft, JEE tiene una gran cantidad de proyectos importantes a sus espaldas.
- Soluciones libres: En la plataforma JEE es posible crear arquitecturas completas basadas única y exclusivamente en productos de software libre. No sólo eso, sino que los arquitectos normalmente disponen de varias soluciones libres para cada una de las partes de su arquitectura.

Aún así, la plataforma de J2EE también tiene desventajas, algunas importantes:

- Depende de un único lenguaje: La plataforma J2EE depende exclusivamente del lenguaje Java. Sólo se puede utilizar este lenguaje para desarrollar aplicaciones lo que puede suponer un gran problema si nuestro equipo no dispone de los conocimientos suficientes o tiene otras preferencias.
- Complejidad: Aunque no es una plataforma tan compleja como CORBA, no existe un VB .NET en Java. La creación de aplicaciones bajo J2EE requiere normalmente desarrolladores más experimentados que los necesarios para desarrollar bajo .NET
- Heterogeneidad: Existe una gran heterogeneidad en las soluciones de desarrollo. No existe en J2EE un símil a Visual Studio .NET. La gran cantidad de herramientas disponibles causa confusión dentro de los desarrolladores y puede crear dependencias dentro de las empresas.

4.1.4.- SERVICIOS WEB

Los servicios web son servicios disponibles a través de Internet que utilizan mensajería basada en XML, y que no están vinculados a ningún sistema operativo ni lenguaje de programación. Cuando se produce una llamada a un servicio web, se envía un documento XML con la petición a través de la red y, opcionalmente, se recibe una respuesta en otro documento XML. Los estándares de los servicios web definen el formato de los mensajes, así como los mecanismos para su publicación y búsqueda, entre otras muchas cosas.

Recientemente, los servicios web están siendo bastante utilizados para integrar desarrollos con la parte cliente en Windows Forms de .NET y la lógica de negocio en EJBs. En este caso, las operaciones de los EJBs se publican como servicios web, y desde la aplicación cliente de .NET se llama a estos servicios. Esto permite aprovechar las ventajas de cada tecnología, ya que los Windows Forms permiten un desarrollo más sencillo y rápido de aplicaciones cliente pesadas, que no Swing. Respecto a los EJBs, son una buena opción para hacer la parte de la lógica de negocio, pero como no es posible la comunicación directa entre EJBs y .NET, ésta se lleva a cabo por medio de servicios web, que de esta forma demuestran que son una excelente herramienta de integración entre tecnologías heterogéneas.

Los servicios web están basados en diversas especificaciones y estándares. Algunos de los principales son los siguientes:

- HTTP : Es el protocolo de comunicaciones más popular de Internet, utilizado para enviar los mensajes XML a través de la red.
- SOAP : Lenguaje XML utilizado para codificar la información de las peticiones y respuestas en mensajes XML.
- WSDL : Lenguaje XML utilizado para describir un servicio web, y que un cliente pueda así conocer todo lo que necesita para interactuar con un servicio, como el nombre de los métodos, tipos de datos de los parámetros y valor de retorno, métodos de comunicación soportados, etc.
- UDDI : Repositorio de enlaces a servicios web, pensado para que un cliente pueda encontrar cualquier servicio web, en base al tipo de servicio, nombre de la empresa, etc.

4.1.5.- Elección de Tecnología

En un primer momento, creo que el mejor estilo que se adapta a nuestro sistema es un estilo centrado en cliente-servidor, basado en componentes y organizado en capas o niveles, y utilizando al máximo posible soluciones de software libre, por lo que una vez vistas algunas de las tecnologías existentes creo que la mejor elección es JEE.
Además, JEE es más sencilla que CORBA y más portable que .NET. También debemos tener en cuenta que podemos ejecutar las aplicaciones JEE desde cualquier S.O.

4.2.- JEE: Tecnología de desarrollo del proyecto

4.2.1.- ¿Qué es JEE?

JEE es una plataforma de desarrollo empresarial (propuesta por Sun Microsystems en el año 1997) que define un estándar para el desarrollo de aplicaciones empresariales multicapa.
JEE simplifica el desarrollo de estas aplicaciones basándolas en componentes modulares estandarizados, proporcionando un conjunto muy completo de servicios a estos componentes y gestionando automáticamente muchas de las funcionalidades o características complejas que requiere cualquier aplicación empresarial (seguridad, transacciones, etc.), sin necesidad de una programación compleja.
Los conceptos básicos y los puntos clave que hay detrás de la plataforma JEE:
• JEE es una plataforma abierta y estándar. No es un producto, sino que define un conjunto de estándares que todos los contenedores deben cumplir para comunicarse con los componentes. En algunos ámbitos, se dice que JEE es una especificación de especificaciones.
• JEE define un modelo de aplicaciones distribuido y multicapa con n-niveles. Con este modelo, podemos dividir las aplicaciones en partes y cada una de estas partes se puede ejecutar en distintos servidores. La arquitectura JEE define un mínimo de tres capas: la capa cliente, la capa intermedia y la capa de sistemas de información de la empresa.

4.2.2.- Modelo de programación de aplicaciones (Blueprints)

La plataforma JEE utiliza un modelo de aplicación distribuida, basada en componentes y multicapa con n-niveles.
Los Blueprints para la plataforma JEE describen recomendaciones de diseño para dividir una aplicación JEE en capas y decidir qué componentes hay que poner en cada capa. Definen el modelo de programación que deben seguir de las aplicaciones que se

desarrollan bajo la plataforma y todo un conjunto de buenas prácticas a la hora de utilizar la plataforma.

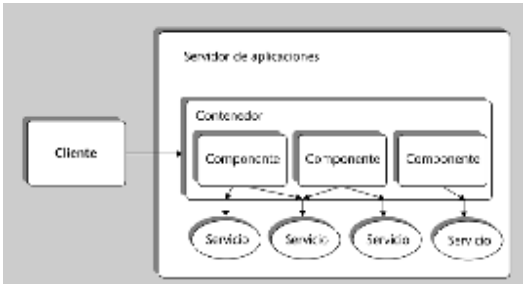
Para entender cuál es el modelo de programación de aplicaciones de JEE, tenemos que profundizar en los conceptos de componente, contenedor y servicio y establecer la relación que hay entre éstos y el modelo de programación de aplicaciones JEE.

4.2.2.1.- Componentes, contenedores, servicios y servidores de aplicaciones

Una aplicación JEE estará formada por un conjunto de componentes que se ensamblan y despliegan en contenedores JEE para ofrecer a los usuarios la funcionalidad deseada.

Los contenedores son piezas de software que ofrecen acceso a los servicios a los componentes que están desplegados dentro del contenedor.

Los servidores de aplicaciones son piezas de software que implementan los servicios que ofrecen los contenedores a los componentes. Los contenedores forman parte de los servidores de aplicaciones.



La mayoría de las aplicaciones empresariales que queremos desarrollar tienen que utilizar un conjunto de servicios de bajo nivel idénticos: gestión del ciclo de vida de los componentes, seguridad, transacciones, balanceo de carga, gestión de recursos, etc.

Una primera opción es que cada aplicación implemente estos servicios. Sin embargo, puesto que estos servicios se repiten en todas las aplicaciones que desarrollamos, es conveniente definirlos e implementarlos una sola vez y dejar que todas las aplicaciones puedan hacer uso de los mismos.

Las aplicaciones JEE se construyen ensamblando componentes y desplegándolos en un contenedor. Las aplicaciones están formadas por componentes que se ejecutan dentro de contenedores. Los contenedores proporcionan un entorno de ejecución y el acceso a un conjunto de servicios de bajo nivel a los componentes que forman la aplicación.

El proceso de ensamblaje de un componente consiste en empaquetar en un formato estándar todo lo que forma el componente (clases, interfaces, descriptores, etc.) para poder hacer su despliegue.

El despliegue de un componente es la fase de instalación del componente en el contenedor correspondiente. En esta fase, se configuran los servicios del contenedor que necesita el componente. La configuración de los servicios se suele hacer de manera declarativa, con ficheros XML que se empaquetan con los componentes.
Por ejemplo, un componente puede configurar la seguridad de manera declarativa al ser desplegado en el contenedor y decir qué usuarios o grupos de usuarios están autorizados a ejecutar qué métodos del componente.
En un modelo de componentes y contenedores, los componentes tienen la lógica de negocio y los contenedores les proporcionan un entorno de ejecución y todo un conjunto de servicios de bajo nivel.
Es importante tener en cuenta que en un modelo de componentes y contenedores los clientes interactúan con los contenedores en lugar de hacerlo con los componentes directamente. Los contenedores interceptan las llamadas a los componentes, ejecutan las tareas que se han definido de manera declarativa para el componente y le pasan el control para que ejecute la lógica programada por esta llamada.
Para que los contenedores puedan gestionar el ciclo de vida, proporcionar un entorno de ejecución y ofrecer un conjunto de servicios a los componentes, se tienen que definir una serie de normas (un contrato estándar) que los componentes y los contenedores deben cumplir.
JEE define un modelo de componentes y contenedores abierto y estándar. Esto quiere decir que los contratos entre los componentes, los contenedores y los servicios que tienen que proporcionar los define una especificación estándar.
De esta manera, se consigue que cualquier fabricante pueda desarrollar contenedores capaces de ejecutar componentes JEE; sólo tiene que cumplir los contratos estándar e implementar los servicios requeridos por la especificación.
Las plataformas de software que implementan estos contenedores y servicios se llaman servidores de aplicaciones y hay muchos fabricantes diferentes de los mismos, algunos comerciales, otros de código libre.

4.2.2.2.- Arquitectura multicapa con n-niveles

El modelo de aplicaciones que define JEE se enmarca dentro de un estilo arquitectónico heterogéneo que combina características de diferentes estilos, y se centra en un estilo cliente-servidor, basado en componentes y organizado en capas o niveles.

JEE extiende la arquitectura típica en tres capas o niveles para definir una arquitectura en n-capas.

4.2.2.3.- Ciclo de vida

Una vez desarrollados los componentes que formarán la aplicación JEE, se tienen que empaquetar y desplegar en el servidor de aplicaciones para que los clientes los puedan utilizar. Para desplegarlos, primero hay que crear un módulo empaquetando los componentes desarrollados con los ficheros relacionados y un descriptor de despliegue del componente.

• Un descriptor de despliegue es un fichero XML asociado a un componente, un módulo o una aplicación JEE que proporciona la información necesaria para hacer el despliegue en el servidor de aplicaciones. Indica en el contenedor declarativamente qué comportamiento tendrán los componentes que forman el módulo con respecto a seguridad, transacciones, persistencia, etc. El hecho de tener una configuración declarativa y no programática facilita la portabilidad de los componentes y módulos.

• Un módulo es la unidad mínima que se puede desplegar en un servidor de aplicaciones. Aunque los módulos se pueden desplegar sin que formen parte de una aplicación, lo habitual es juntar estos módulos para crear una aplicación JEE.

4.2.3.- Plataforma JEE

Al analizar el modelo de componentes y contenedores, vimos que los contenedores daban a los componentes un conjunto de servicios. Estos servicios simplifican el desarrollo de aplicaciones y permiten su personalización en tiempo de despliegue.

El conjunto de API, servicios y tecnologías que tiene que soportar una determinada versión de la plataforma JEE se recoge en la especificación formal de la plataforma que define Sun Microsystems.

En el nivel de servicios, JEE nos proporciona los siguientes:

a) Servicio de nombres: Permite el acceso a recursos y componentes mediante una API unificada de acceso (JNDI). Cuando un componente o un recurso se despliega en el servidor de aplicaciones, el servicio de despliegue lo ubica en el árbol JNDI del servidor para que los clientes puedan acceder al mismo. Los clientes utilizan el servicio de nombres y directorios para acceder a los recursos y los componentes desplegados.

b) Servicio de transacciones: Este servicio hace que los desarrolladores no tengan que escribir código para gestionar el comportamiento transaccional de los componentes. La

especificación del comportamiento transaccional se efectúa en el descriptor de despliegue y se configura cuando se ensambla el componente.

c) Servicio de seguridad: Asegura que sólo accedan a los componentes y recursos aquellos que tengan acceso. Este servicio también proporciona mecanismos de autentificación y de confidencialidad.

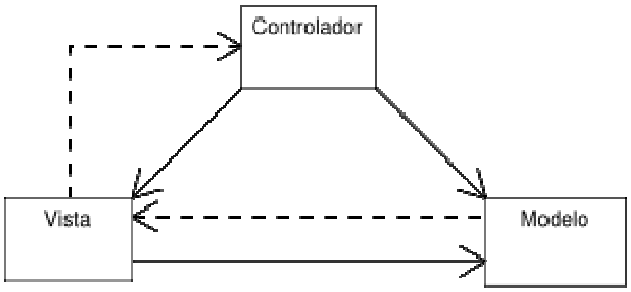
d) Servicios de despliegue: Antes de que un componente se pueda ejecutar, se debe montar y desplegar en su contenedor. El servicio de despliegue configura e "instala" los componentes en los contenedores y habilita el acceso a las aplicaciones JEE.

4.4.- El patrón MVC (Modelo Vista Controlador)

4.4.1.- Introducción

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio.

El patrón fue descrito por primera vez en 1979 por Trygve Reenskaug, entonces trabajando en Smalltalk en laboratorios de investigación de Xerox.



4.4.2.- Descripción del patrón

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario, etc.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1.- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace)

2.- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

3.- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4.- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

5.- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

4.5.- Frameworks

4.5.1.- Introducción

Según la wikipedia:

En el desarrollo de software, un Framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, librerías y un lenguaje de scripting entre

otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

FreeDictionary extrae una definición de framework de Jonhson y Foote 1988:
Un "Software Framework" es un diseño reusable de un sistema (o subsistema). Está expresado por un conjunto de clases abstractas y el modo en que sus instancias colaboran para un tipo específico de software. Todos los frameworks de software son diseños orientados a objetos"

4.5.2.- Arquitecturas Modelo 1 y Modelo 2

Los términos Modelo 1 y Modelo II surgen de borradores de las primeras especificaciones de Java Server Pages (JSP) en donde se describían dos patrones básicos para construir aplicaciones basadas en esa tecnología.
El modelo 1 carece del concepto de controlador central que sí tiene el modelo 2.
Una arquitectura de Modelo 1 consiste básicamente en que desde la página JSP accedida desde el navegador se tuviera acceso a las clases del modelo, se les aplicara la lógica del negocio y se seleccionara la siguiente vista a ser enviada al navegador. Cada página procesaba su propio input.
En una arquitectura Modelo 2 existe un servlet que actúa de controlador central que recibe todos los requests y a partir de la dirección de origen, de los datos de entrada, el estado actual de la aplicación selecciona la siguiente vista a mostrar. En este servlet se pueden concentrar todos los temas relativos al conjunto total de las llamadas como la seguridad y la auditoría (logging). Las aplicaciones son más extensibles porque las vistas no se relacionan con el modelo ni entre sí. Por otro lado, a la las clases que implementan la lógica de la aplicación les llega la información digerida para su tratamiento. Este servlet central recibe el nombre de Front Controller.

4.5.3.- Comparativa de Frameworks Webs

Si aplicamos la definición anterior de Framework al desarrollo web, podemos llegar a la conclusión que un framework web es una estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones web. En cierto sentido podemos afirmar que nos proveen una capa de abstracción sobre la arquitectura original ocultándola o adaptándola para no tener que utilizar el protocolo http de manera nativa y así acelerar los tiempos de desarrollo y mantenimiento.

Sin tener que buscar mucho por Internet, comprobamos que existen multitud de Frameworks web, y sin lugar a dudas, no podemos decir con certeza, cuál de ellos es el mejor. Cada uno tiene sus pros y sus contras, y tan sólo después de haber utilizado algunos de ellos, cada uno podrá elegir el que a su gusto es el mejor.
Por tanto, y una vez dicho esto, voy a analizar unos cuantos Frameworks webs, que no tienen por qué ser los mejores y mucho menos los únicos.

4.5.3.1.- Struts

Struts es por el momento el más difundido de los frameworks opensource en el ámbito java. Está basado en el modelo 2 y consta, por lo tanto, de un servlet que actúa de controlador central que recibe todas las peticiones de los clientes. Las facilidades de desarrollo que ofrece son:

- Lógica de navegación entre páginas: Struts tiene un vocabulario específico para definir cono funciona. Los términos más importantes de definir son los siguientes: Actions, ActionMaping, ActionServlet y ActionForm.
- Binding entre java y el html: Utiliza etiquetas (tags) que permiten generar vistas conteniendo sólo tags y sin código java. Estos tags se dividen en 4 grupos: struts-bean, struts-html, struts-logic y struts-nested
- Validación de entradas: Struts provee mecanismos de validación de las entradas ingresadas. Existen dos maneras principales: Redefiniendo el método validate() de los ActionForms o a través de lo que primero fue un plugin y luego se incorporó a la versión principal y que se denomina struts-validator.
- Internacionalización: Struts brinda soporte para internacionalización extendiendo la funcionalidad que ya provee java. Permite internacionalizar una aplicación utilizando archivos de texto conteniendo conjuntos de datos con el formato clave=valor y referenciando estas claves en los archivos de la vista.
- Independencia del motor de visualización: Struts en principio es independiente del motor de visualización aunque generalmente se elija jsp para mostrar las vistas. Existen formas de que struts envíe las vistas para que sean procesadas por motores de plantillas como velocity, transformadores de estilos de documentos XSLT u otros frameworks de presentación como JSF. Por lo tanto struts no está ligado a un motor de visualización particular.

- Maquetación: La maquetación de la aplicación WEB es facilitada a través de Struts Tiles, un plugin que permite componer a partir de porciones de página, la página definitiva que será enviada al cliente. La composición de las partes se puede definir de tres maneras:
    - A través de un xml
    - Dentro de las páginas jsp
    - Programáticamente desde las Actions

4.5.3.2.- Tapestry

Tapestry es otro framework open-source modelo 2 mantenido por la comunidad Apache, y una de sus principales características es que está basado en un modelo de componentes. Esto provee una estructura consistente, permitiendo al framework asumir responsabilidades sobre conceptos como la construcción de URL, el despacho, el almacenamiento del estado en el cliente o en el servidor, la validación del usuario, la localización/internacionalización, el manejo de reportes, etc. Un componente es un objeto que tiene sus responsabilidades definidas por el diseño y la estructura del framework en el cual se encuentra. Es decir, sigue una serie de convenciones (nomenclatura, implementación de ciertas interfaces, etc) que le exige el framework. Tapestry, al igual que todos los frameworks de modelo 2, tiene un servlet que centraliza toda la lógica de comunicación.

Existen ciertos conceptos clave que se definen para entender el funcionamiento:

Página: las aplicaciones consisten de una colección de páginas identificadas unívocamente a través de su nombre. Cada página contiene un template y otros componentes.

Template: Un template puede ser para una página o un componente. Contiene html plano con algunos tags marcados con un atributo especial para indicar que en ese lugar deben situarse los componentes.

Componente: Un objeto reusable que puede ser usado como parte de una página. Los componentes generan html cuando se renderiza la página y pueden participar cuando se selecciona un enlace o se envía un formulario.

Parámetro: Los componentes tienen parámetros que sirven para enlazar las propiedades de los componentes con las propiedades de la página. Los componentes generalmente leen los parámetros pero a veces pueden modificarlos haciendo que se actualicen las propiedades de la página que estaban relacionadas con ese parámetro.

Las facilidades de desarrollo que ofrece Tapestry son:

- Transparencia en la construcción de las vistas: Las vistas de tapestry no son ni más ni menos que archivos en html estándar. No existen las bibliotecas de tags ni código java. La única diferencia es la existencia de algunos atributos extras en los elementos que aparecen.
- Binding entre java y html: Para vincular el código java con el html, Tapestry utiliza un lenguaje especial en los templates html llamado OGNL (Object Graph Navigation Lenguaje) y permite expresar el acceso a un valor de algún objeto en forma de cadena de texto. A partir de un objeto que sirve como punto de partida se puede navegar a través de sus propiedades hasta llegar al elemento deseado. El OGNL se utiliza para vincular valores y métodos con propiedades y manejadores de eventos respectivamente. Por otro lado, cada página, además de un html tiene otras dos partes: una especificación y una clase asociada. El archivo de especificación posee la extensión .Page y es un xml que indica cual es la clase que se hará cargo del template y cuales son los tipos de las propiedades a utilizar
- Manejo de eventos: El manejo de eventos se realiza a través de la suscripción de listeners a los componentes que lanzan dichos eventos
- Construcción de componentes: En Tapestry todo son componentes, incluyendo las páginas. Por lo tanto, al crear páginas estamos creando nuevos componentes. La creación de componentes se realiza de manera similar a la creación de páginas. Por lo general se requiere un template.html, un archivo de definición (xml) y un archivo con la implementación (java).
- Validación de entradas: Los componentes que reciben la entrada del usuario, permiten la validación a través de dos parámetros: displayname y validators.
- Internacionalización: Para la internacionalización Tapestry utiliza lo que denomina catálogos de mensajes que vendrían a ser algo similar a los ResourceBoundles de java donde se guardan pares de cadenas con el formato clave=valor. Cada componente puede tener un set de catálogos de mensajes. Estos catálogos se nombran con el mismo nombre que el componente pero su extensión es .properties. Si una clave no se encuentra en ninguno de los catálogos Tapestry no informa ningún error sino que genera un valor propio. Para indicar una referencia a una clave dentro del html se utiliza la palabra message.

4.5.3.3.- Java Server Faces

Java Server Faces no es una implementación sino una especificación (JSR 127) aprobada por el Java Community Process (JCP) para construir interfaces de usuario para las aplicaciones que corren en un servidor. Es decir, es un estándar y pueden existir varias implementaciones mientras que cumplan con lo que exija la especificación.

JSF es otro framework modelo 2 que posee un controlador central (FrontControler) que se encarga de manejar todas las peticiones del cliente y gestionar su ciclo de vida. Está basado en un modelo de componentes para la interfaz de usuario. Un componente JSF es un elemento reusable y configurable que se puede utilizar en la interfaz de usuario. Los componentes se pueden anidar.

Un aspecto fundamental de JSF es el ciclo de vida de una petición web. El ciclo de vida está separado en 6 fases principales

1. Restore View: Crea el árbol de componentes de la página solicitada y carga el estado si esta ya había sido solicitada previamente.
2. Apply Request Value: Itera sobre el árbol de componentes recuperando el estado de cada uno asignándole los valores que viene desde el cliente.
3. Process Validations: Se realizan las validaciones de cada componente
4. Update Model Values: Se actualizan los valores de los backing beans del modelo cuyas propiedades estaban vinculadas a propiedades de los componentes de la vista.
5. Invoke application: Se ejecuta la lógica del negocio y se selecciona la próxima vista lógica.
6. Render Response: Se arma la vista con el estado actualizado de los componentes y se la envía al cliente.

Otra concepto importante es el de los beans administrados. JSF provee un contenedor de inversión de control para administrar los beans que realizan el binding entre la vista y el modelo. En estos beans se recuperan los valores ingresados una vez que estos fueron validados, es decir, en la fase Update Model Values.

Las facilidades de desarrollo que ofrece JSF son:

- Lógica de navegación entre páginas: El punto de entrada a la aplicación es el FacesServlet, que se encarga de controlar el ciclo de vida de cada petición. JSF permite definir la lógica de navegación a través de uno o más archivos de configuración (faces-config.xml). Dicha lógica se construye a través de reglas de

navegación. Cada regla se activa según se cumpla el patrón indicado en el elemento from-view-id. Dentro de las reglas de navegación pueden haber de cero a muchos casos de navegación. El caso de navegación se selecciona en función del valor utilizado para indicar la acción a realizar y del valor de retorno de las llamadas al método invoke() de dicha acción que es expresado en el tag from-outcome. Por último, se selecciona la vista a mostrar indicada en el elemento to-view-id del caso.

- Binding entre la vista y los beans de negocio: El modelo de componentes permite el enlace de valores (value binding), de métodos (method binding) y de componentes (component binding).
- Manejo de eventos: JSF implementa un modelo que permite la notificación mediante eventos y la subscripción a dichos eventos mediante listeners. Todos los eventos generados desde los componentes de la interfaz son subclases de FacesEvent. Las dos subclases estándares que derivan de FacesEvent son ActionEvent y ValueChangeEvent. Un evento debe indicar en que etapa quiere ser entregado informándolo a través del método getPhaseId().
- Internacionalización: La internacionalización de JSF está construida sobre la base del soporte que ya brindaba java, la especificación de servlets y la de JSPs. JSF maneja el concepto de Locale (configuración local) activo. Este se utiliza cuando se accede a los recursos, cuando se utilizan los conversores, etc. Los posibles Locales que tendrá la aplicación se definen en el archivo de configuración
- Validación de entradas: La inteligencia de la validación de entrada reside en los Validadores (Validators). Un validador se encarga de realizar comprobaciones sobre el valor de un componente durante la fase Process Validations. También es posible llamar a los validadores en cualquier momento a través del validate() del componente.
- Independencia del dispositivo de presentación: La codificación de los valores de los componentes para mostrarlos en la vista y la decodificación necesaria de los valores que llegan de las peticiones varía dependiendo del dispositivo. En JSF, esta codificación/decodificación se puede realizar de dos maneras, utilizando un modelo de implementación directa o utilizando un modelo de implementación delegada.
- Construcción de componentes: JSF permite la creación de componentes propios con o sin render asociado. Sin embargo, la creación no es sencilla y se necesita crear varios archivos dependiendo el tipo de componente que se desea crear.

4.5.3.4.- Cocoon

Cocoon es un framewok creado en 1999 por Stefano Mazzocchi. Está desarrollado en java y es completamente diferente a los que se detallaron anteriormente. Se basa en un modelo de componentes y en el concepto de tuberías (pipelines) de componentes. Este concepto se asemeja a una línea de producción donde cada componente se especializa en una operación en particular (existen componentes generadores, transformadores, serializadores, etc). El contenido ingresa en la tubería y es modificado por las sucesivas etapas hasta generar la respuesta que es enviada al cliente. De esta manera se busca separar el contenido, el estilo, la lógica y la administración de un sitio web basado en contenido XML.

El elemento global que utiliza para definir una aplicación se denomina sitemap. Este sitemap incluye todo lo que tendrá el sitio web y es configurado a través del archivo xml sitemap.xmap. En este archivo se especifican los componentes, las vistas, los recursos, los conjuntos de acciones y las tuberías que tendrá la aplicación.

Los componentes pueden ser:

- Generadores (generators): generan XML como eventos SAX y dan inicio al procesamiento del pipeline
- Transformadores (transformers): transforman los eventos SAX recibidos en otros eventos SAX
- Serializadores (serializers): transforman los eventos SAX en streams binarios o de texto que pueden ser entendibles por el cliente.
- Selectores (selectors): permiten implementar lógica condicional básica (if-then o switch)
- Mapeadores (matchers): mapea un patrón de entrada con un recurso
- Acciones (actions): son invocadas desde el pipeline y ejecutan algun tipo de operación antes de continuar con el resto del proceso. Las acciones tienen un método act() en donde ejecutan su lógica y devuelven un conjunto de resultados en forma de diccionario (map). Generalmente son las encargadas de invocar a la lógica del negocio.

Todos los componentes poseen un nombre y una clase donde tienen su implementación.

Las facilidades de desarrollo que ofrece Cocoon son:

- Control de flujo de navegación: Cocoon provee un control de flujo avanzado permitiendo describir el orden en que las páginas deben ser enviadas al cliente. Las

aplicaciones tradicionales son orientadas a eventos donde, al recibir un evento se modifica el estado interno y se emite una respuesta. Esto genera complicaciones a la hora de requerir entradas de datos en varios pasos y de definir en donde se ejecuta la lógica que genera la respuesta. Cocoon busca una aproximación más secuencial en el cual se pueda escribir una conversación entre el cliente y el servidor de manera natural. Para esto, Cocoon implementa lo que denomina "continuations". Una "continuation" es un objeto que, llegado un punto de ejecución, guarda una copia del estado actual del hilo incluyendo la pila, los valores de las variables y la posición de la próxima instrucción a ejecutar. De esta manera, cuando se requiere la entrada de datos de un cliente todo el estado del hilo que estaba en el servidor se almacena en una "continuation". Cuando el cliente devuelve los datos requeridos informa a Cocoon cual es la continuation que debe restaurar.

- Separación de incumbencias: En un desarrollo en el que están involucradas varias personas difícilmente puedan llevar a cabo sus tareas sin interferirse. Cocoon identifica 4 áreas principales que se pueden aislar y en las que los participantes generalmente tienen habilidades bien definidas. Estas cuatro áreas son Lógica, Contenido, Estilo y Administración. Este framework fue diseñado para poder aislar estas cuatro áreas.
- Internacionalización: La internacionalización y localización se logra a través de un transformador llamado I18nTransformer que utiliza diccionarios XML para los datos que deben estar en varios idiomas. Es posible traducir texto, atributos, texto con parámetros y localizar formatos de fecha/hora, numeración y moneda.
- Independencia del dispositivo de presentación: Cada tubería finaliza con un serializador que convierte el contenido xml en un stream binario o de texto. Cocoon provee serializadores para html, xml, pdf, xhtml, wml, svg, vrml, zip, ps, etc. A la vez es capaz de reconocer el tipo de dispositivo del cual proviene la petición. Esto permite construir la aplicación sin depender del dispositivo en el que se visualizará.
- Validación de entradas: Cocoon tiene su propio modelo de formularios para obtener datos de los clientes llamado Cforms. Un form está compuesto por un conjunto de widgets. Un widget es un objeto que sabe como leer un valor que llega desde un Request de un cliente, sabe como validarlo, y posee una representación en XML de si mismo. A la vez, un widget es capaz de guardar su estado interno por lo que no es necesario utilizar objetos externos para almacenar su valor.

4.5.4.- Otros Frameworks

Hasta ahora hemos hablado de Frameworks Webs, pero no podemos olvidar que existen otros Frameworks, como Swing, Hibernate, ... que no son Frameworks Web.
Veamos alguno de ellos:

4.5.4.1.- Hibernate

Hibernate es una capa de persistencia objeto/relacional y un generador de sentencias sql. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada podremos generar BBDD en cualquiera de los entornos soportados: Oracle, DB2, MySql, etc.. Y lo más importante de todo, es Open Source.
Uno de los posibles procesos de desarrollo consiste en, una vez tengamos el diseño de datos realizado, mapear este a ficheros XML siguiendo la DTD de mapeo de Hibernate. Desde estos podremos generar el código de nuestros objetos persistentes en clases Java y también crear BBDD independientemente del entorno escogido.
Hibernate se integra en cualquier tipo de aplicación justo por encima del contenedor de datos.

4.5.4.2.- Spring

Spring es un framework de aplicaciones Java/J2EE. Soporta la persistencia mediante el mapeo objeto-relacional a través de anotaciones en las clases Java o mediante ficheros XML.

Persistencia:

Spring soporta a la vez varios proveedores de persistencia, como Hibernate, JDO, Toplink, iBatis o cualquier otro proveedor compatible con JPA. Además soporta la persistencia JDBC.
Para implementar una caché de objetos compartidos por distintas unidades de trabajo (transacciones), lo que se hace en Spring es fijar la sesión (y la transacción) al hilo de ejecución, usando variables locales del hilo de ejecución. Spring proporciona clases para facilitar la implementación.
Para implementar un mecanismo de “lazy loading”, mediante el cual la ejecución de una consulta a base de datos no se realice hasta que realmente sea necesaria, en Spring se usa un interceptor que debe ser configurado (OpenSessionInViewFilter).

Gestión de transacciones:

En Spring, la lógica de negocio no tiene que ocuparse explícitamente de la gestión de transacciones, ya que esto se realiza de forma declarativa.

La demarcación, la propagación y el aislamiento de transacciones se declaran mediante programación orientada aspectos, definiendo proxies para las transacciones en los ficheros de configuración XML. Spring se integra con diferentes APIs de transacciones, incluyendo JDBC, Hibernate y JTA.

Manejo del estado:

Hay que decir que las arquitecturas con estado no tienen el mismo rendimiento ni se escalan igual que las que no tienen estado. Pero hay que tener en cuenta que el estado es importante para muchas aplicaciones, debido a que la interacción del usuario con el sistema implica a veces una serie de pasos con efecto acumulativo.

Para que un bean de Spring tenga estado no deben ser compartidos entre llamadas. Para ello, el alcance de los beans debe configurarse como "prototype", de manera que cada vez que se recupera un bean de la factoría se cree una nueva instancia.

5.- ANALISIS

Se pretende conseguir una aplicación fácil de programar, fácilmente mantenible, y que ofrezca un interfaz amigable para la creación y consulta de la Productividad en las Delegaciones de la Junta de Andalucía, principalmente en la Delegación del Gobierno.

El proyecto se va a realizar utilizando el lenguaje de programación Java en su especificación JEE. Otra característica importante de la implementación es la utilización del patrón MVC, lo que nos permitirá agilizar el desarrollo y el posterior mantenimiento de la aplicación.

5.1.- Análisis de requisitos

He dividido los requisitos en funcionales y no funcionales, como se ve en los puntos siguientes:

5.1.1.- Requisitos funcionales

Los requisitos funcionales que debe cumplir la aplicación son los siguientes:

- Gestión de funcionarios:
    - Insertar nuevo funcionario

    - Alta funcionario existente
    - Modificar funcionario
    - Baja funcionario
- Gestión de la Productividad
    - Crear productividad
    - Modificar productividad
    - Publicar productividad
    - Eliminar productividad
    - Modificar valores funcionarios productividad
- Gestión histórico de productividades
    - Ver productividad
    - Eliminar productividad
    - Exportar productividad
- Control de acceso
    - Protocolo de seguridad de acceso a los usuarios mediante login y password que le restringirá de ciertas acciones.

5.1.2.- Requisitos no funcionales

El sistema debe cumplir además con los siguientes requisitos no funcionales:

- Entorno amigable: el sistema tiene que tener un entorno fácil de usar para todos los usuarios.
- Rápido: debe ser lo más eficiente posible en cuanto a velocidad.
- Coste: el proyecto debe realizarse con uso prioritario de software libre
- Ampliable: la aplicación tiene que ser fácilmente ampliable, con tal de poder ampliarla más adelante con otros servicios como puede ser el absentismo.
- Multithread: el sistema puede ser utilizado por más de una persona a la vez.
- Documentación: el sistema debe estar documentado lo mejor posible para futuras mejoras o ampliaciones.

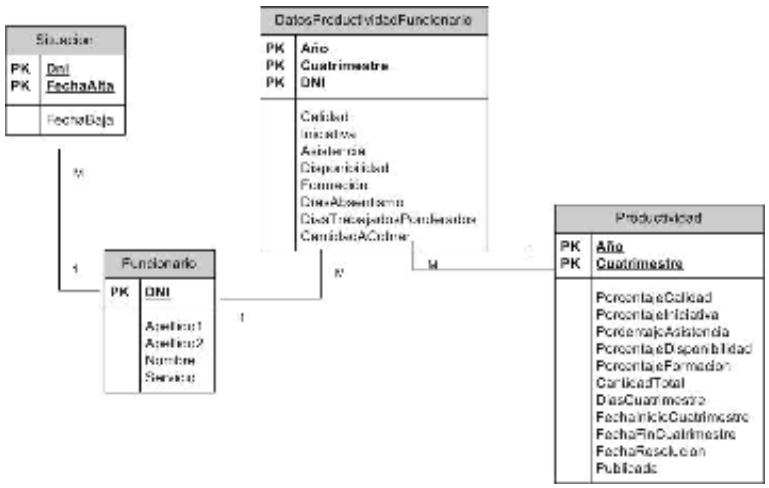
5.1.3.- Roles

Se han definido tres roles o usuarios tipo diferentes que pueden utilizar la aplicación. Se distinguen principalmente por los permisos que disponen y por las acciones que pueden llegar a realizar. Son los siguientes:

- Usuario: Cualquier usuario que acceda a la aplicación, tendrá la opción de ver los listados de la productividad de cualquier cuatrimestre que ya esté publicada. No necesitan validarse.
- Jefe de Servicio: Usuario registrado, podrá modificar la productividad de los funcionarios pertenecientes a su servicio.
- Jefe de Personal: Será la persona con permisos totales sobre la aplicación, el administrador de la misma. Podrá agregar usuarios, crear nuevas productividades, publicar la productividad, ...

5.2.- Diagrama Conceptual de Datos

El modelo conceptual de datos sería el siguiente:



6.- DISEÑO

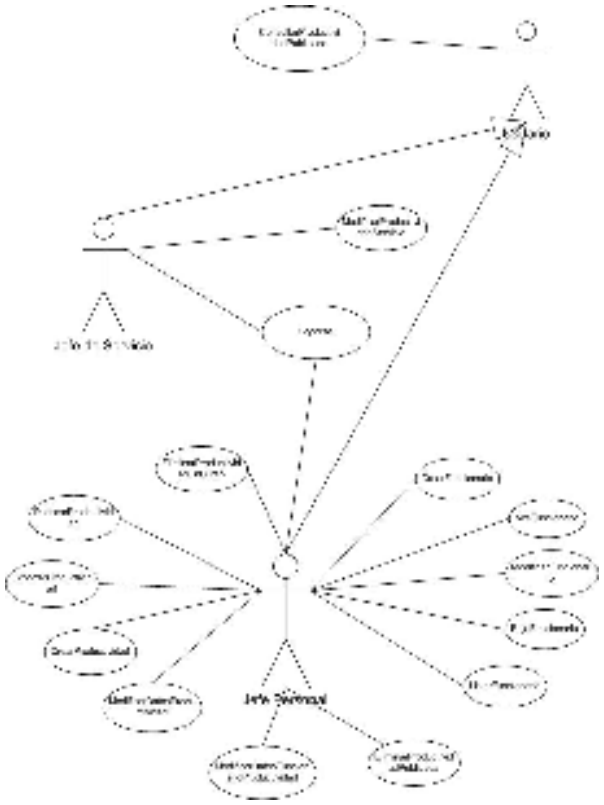
La metodología utilizada para el diseño y modelado de la aplicación se ha basado en la tecnología UML. Gracias a esta desarrollaré los diagramas más comunes y útiles para la posterior implementación del sistema.

6.1.- Diagrama de casos de uso

En este diagrama relacionaremos los actores con las acciones.

Los actores ya los tenemos definidos en el punto 5.1.3. (Roles), así pues pasaremos a describir las acciones que aparecen en nuestro sistema junto con su relación de actores. En el esquema de caso de uso de la figura siguiente se observan las diferentes acciones

que pueden realizar los diferentes actores. Hay que destacar, que tanto el jefe de servicio como el Administrador pueden realizar todas las acciones del usuario.



A continuación explicaré en detalle cada uno de los casos de uso:

Caso de uso: 1.- Consultar Productividad Publicada

Resumen de la funcionalidad: Ver un listado de una productividad.

Actores: Usuario, Jefe de Servicio y Administrador

Precondición: La productividad ha sido publicada.

Proceso:

1. El sistema muestra un listado de con todas las productividades publicadas.
2. El usuario elige la productividad que desea consultar.
3. El sistema muestra un listado con los datos de los funcionarios existentes para esa productividad.

Caso de uso: 2.- Logarse

Resumen de la funcionalidad: Introducir el usuario y la contraseña.

Actores: Jefe de Servicio y Administrador

Postcondición: Se ha accedido al sistema con el perfil de Jefe de Servicio o de Jefe de Personal.

Proceso:

1. Se introduce el nombre de usuario y contraseña y se pulsa el botón aceptar
2. Si el usuario o la contraseña son incorrectos, el sistema da un mensaje de error y vuelve a mostrar la misma página.
3. Si es correcto accede al menú de Administrador o al menú de Jefe de Servicio.

Caso de uso: 3.- Modificar productividad del Servicio

Resumen de la funcionalidad: Se modifican los datos relativos a los funcionarios de un servicio en la productividad en curso.

Actores: Jefe de Servicio

Precondición: La productividad ya ha sido creada, pero no publicada.

Postcondición: Se ha modificado los datos de los funcionarios en la productividad en curso.

Proceso:

1. El Jefe de servicio selecciona la opción de modificar los datos.
2. El sistema muestra un listado con los funcionarios de su servicio con los valores actuales.
3. El Jefe de Servicio modifica los datos que considera oportunos y pulsa el botón modificar.
4. El sistema recalcula los datos de la productividad y vuelve a mostrar los datos.

Caso de uso: 4.- Crear funcionario

Resumen de la funcionalidad: Se crea un nuevo funcionario.

Actores: Jefe de Personal

Precondición: No hay ninguna productividad en curso.

Postcondición: Se ha creado un nuevo funcionario, y una nueva situación para este funcionario.

Proceso:

1. El Jefe de Personal selecciona la opción de Alta de funcionarios.

2. El sistema muestra un listado con los funcionarios que están de baja, es decir, funcionarios que pertenecieron a la Delegación, pero que ya no están.
3. El Jefe de Personal pulsa el botón Crear nuevo..
4. El Jefe de Personal introduce los datos del funcionario: dni, nombre, apellidos y la fecha de alta.
5. El sistema almacena los datos del nuevo funcionario, y la nueva alta del funcionario.

Caso de uso: 5.- Alta funcionario

Resumen de la funcionalidad: Se crea una nueva alta para un funcionario.

Actores: Jefe de Personal

Precondición: No hay ninguna productividad en curso y el funcionario ya está creado.

Postcondición: Se ha creado una nueva situación de alta para este funcionario.

Proceso:

1. El Jefe de Personal selecciona la opción de Alta de funcionarios.
2. El sistema muestra un listado con los funcionarios que están de baja.
3. El Jefe de Personal pulsa el enlace "Dar de Alta", en uno de los funcionarios listados.
4. El Jefe de Personal introduce la fecha de alta.
5. El sistema almacena la nueva alta del funcionario.

Caso de uso: 6.- Baja funcionario

Resumen de la funcionalidad: Se crea una nueva baja para un funcionario.

Actores: Jefe de Personal

Precondición: No hay ninguna productividad en curso y el funcionario ya está creado. Además no puede haber ninguna productividad en curso.

Postcondición: Se ha creado una nueva situación de baja para este funcionario.

Proceso:

1. El Jefe de Personal selecciona la opción de Baja de funcionarios.
2. El sistema muestra un listado con los funcionarios que están en activo actualmente.
3. El Jefe de Personal pulsa el enlace "Dar de Baja", en uno de los funcionarios listados.
4. El Jefe de Personal introduce la fecha de baja.

5. El sistema almacena la fecha de baja en la situación del funcionario.
6. El sistema muestra el nuevo listado de los funcionarios activos actualmente.

Caso de uso: 7.- Listar funcionario

Resumen de la funcionalidad: Se muestra un listado de los funcionarios activos.

Actores: Jefe de Personal

Postcondición: Se ha consultado el listado de funcionarios.

Proceso:

1. El Jefe de Personal selecciona la opción de Listar/Modificar funcionarios.
2. El sistema muestra un listado con los funcionarios que están en activo actualmente.

Caso de uso: 8.- Modificar funcionario

Resumen de la funcionalidad: Se modifican los datos personales de un funcionario.

Actores: Jefe de Personal

Precondición: El funcionario tiene que estar creado y en situación de alta. Además no puede haber ninguna productividad en curso.

Postcondición: Se han modificado los datos personales de un de funcionarios.

Proceso:

1. El Jefe de Personal selecciona la opción de Listar/Modificar funcionarios.
2. El sistema muestra un listado con los funcionarios que están en activo actualmente.
3. El Jefe de Personal pincha el enlace "Modificar", del funcionario elegido.
4. El sistema muestra los datos personales del funcionario en cuestión.
5. El Jefe de Personal modifica los datos y pulsa el botón modificar.
6. El sistema almacena los nuevos datos y muestra nuevamente el listado de los funcionarios en activo.

Caso de uso: 9.- Crear Productividad

Resumen de la funcionalidad: Se crea una nueva productividad.

Actores: Jefe de Personal

Precondición: Debe haber algún funcionario creado y no debe haber ninguna productividad sin publicar.

Postcondición: Se han almacenado los datos generales de la productividad y los datos de los funcionario actualmente en activo en esta productividad.

Proceso:

1. El Jefe de Personal selecciona la opción de Crear Productividad.
2. El Jefe de Personal rellena los datos generales de la productividad: año, cuatrimestre, % asignado a la calidad, % asignado a la Iniciativa, % asignado a la Asistencia, % asignado a la Disponibilidad, % asignado a la Formación, Cantidad Total asignada al cuatrimestre, Días del cuatrimestre, y pulsa el botón “Crear”.
3. El sistema almacena los datos generales de la productividad, y crea la nueva productividad para todos los funcionarios que están en activo en ese cuatrimestre.
4. El sistema muestra los datos de los funcionarios para esta productividad.

Caso de uso: 10.- Modificar Productividad

Resumen de la funcionalidad: Se modifican los datos generales de la productividad en curso.

Actores: Jefe de Personal

Precondición: Debe haber una productividad creada y sin publicar.

Postcondición: Se han modificado los datos generales de la productividad y se han recalculado los datos de los funcionarios actualmente en activo en esta productividad.

Proceso:

1. El Jefe de Personal selecciona la opción de Modificar Productividad.
2. El Jefe de Personal modifica los datos generales de la productividad y pulsa el botón “Modificar”.
3. El sistema almacena los datos generales de la productividad, y recalcula la nueva productividad para todos los funcionarios que están en activo en ese cuatrimestre.
4. El sistema muestra los datos de los funcionarios para esta productividad.

Caso de uso: 11.- Modificar Datos Productividad Funcionarios

Resumen de la funcionalidad: Se modifican los datos de los funcionarios para la productividad en curso.

Actores: Jefe de Personal

Precondición: Debe haber una productividad creada y sin publicar.

Postcondición: Se han modificado los datos generales de los funcionarios para la productividad en curso.

Proceso:

1. El Jefe de Personal selecciona la opción de Modificar Datos Funcionarios Productividad.
2. El sistema presenta todos los funcionarios con sus valores asociados para la productividad en curso y recalcula la productividad.
3. El Jefe de Personal modifica los datos de los funcionarios y pulsa el botón "Modificar".
4. El sistema almacena los datos, y recalcula la nueva productividad.
5. El sistema muestra los datos de los funcionarios para esta productividad.

Caso de uso: 12.- Eliminar Productividad en Curso

Resumen de la funcionalidad: Se elimina la productividad en curso, y todos los datos de los funcionarios relativos a esa productividad en curso.

Actores: Jefe de Personal

Precondición: Debe haber una productividad creada y sin publicar.

Postcondición: Se elimina la productividad en curso, y los datos relativos a los funcionarios de esta productividad.

Proceso:

1. El Jefe de Personal selecciona la opción de Eliminar Productividad en Curso.
2. El sistema presenta un mensaje de confirmación.
3. El Jefe de Personal Acepta.
4. El sistema elimina la productividad en curso y los datos relativos a los funcionarios de esta productividad.
5. El sistema muestra el menú principal.

Caso de uso: 13.- Publicar Productividad

Resumen de la funcionalidad: Se almacena la productividad como publicada y se crea un fichero de texto para su exportación a la aplicación de nóminas.

Actores: Jefe de Personal

Precondición: Debe haber una productividad creada y sin publicar.

Postcondición: Se modifica la productividad en curso, pasando a ser publicada.

Proceso:

1. El Jefe de Personal selecciona la opción de Publicar Productividad.
2. El sistema presenta todos los funcionarios con sus valores asociados para la productividad en curso.
3. El Jefe de Personal pulsa la opción Publicar.
4. El sistema almacena la productividad como publicada.
5. El sistema muestra el listado de productividades publicadas.

Caso de uso: 14.- Exportar Productividad

Resumen de la funcionalidad: Se crea un fichero txt con la productividad seleccionada.

Actores: Jefe de Personal

Precondición: Debe haber una productividad publicada.

Postcondición: Se crea un fichero txt con los datos de la productividad. El fichero guarda un registro por linea con los datos siguientes:

DNI + 000 + Importe + Fecha de Resolución + Fecha Inicio Cuatrimestre + Fecha Fin Cuatrimestre.

El importe está formateado con 6 dígitos la coma de decimal y dos dígitos para decimales.

Las fechas en formato dd/mm/aaaa

Proceso:

1. El Jefe de Personal selecciona la opción de Exportar Productividad.
2. El sistema presenta un listado con todas las productividades publicadas.
3. El Jefe de Personal pulsa la opción Exportar de la productividad elegida.
4. El sistema crea un fichero con los datos de esa productividad y lo muestra en una nueva pantalla.

Caso de uso: 15.- Eliminar Productividad Publicada

Resumen de la funcionalidad: Se elimina una productividad del histórico.

Actores: Jefe de Personal

Precondición: La productividad debe estar publicada.

Postcondición: Se elimina la productividad seleccionada.

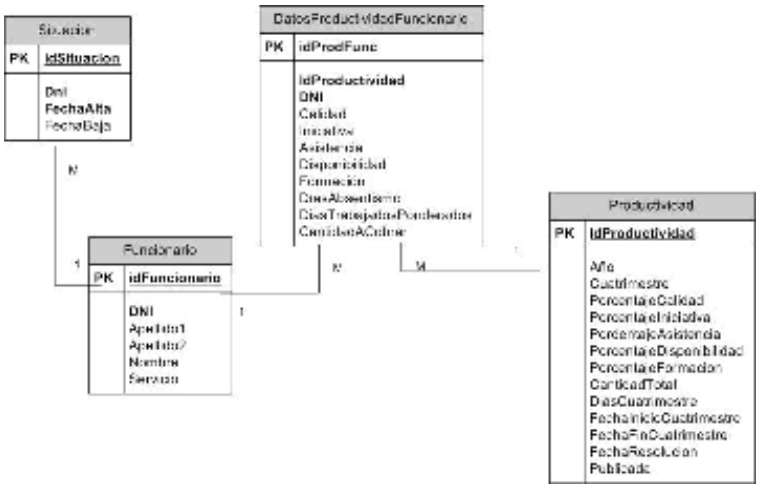
Proceso:

1. El Jefe de Personal selecciona la opción de Eliminar Productividad.

2. El sistema presenta un listado con todas las productividades publicadas.
3. El Jefe de Personal pulsa la opción Eliminar de la productividad elegida.
4. El sistema presenta por pantalla un mensaje de confirmación.
5. El Jefe de Personal Acepta el mensaje
6. El sistema elimina la productividad elegida.
7. El sistema vuelve a presentar un listado con todas las productividades publicadas.

6.2.- Diagrama Estático

A partir del modelo conceptual, visto en el apartado 5.3, vamos a obtener el diagrama lógico relacional.



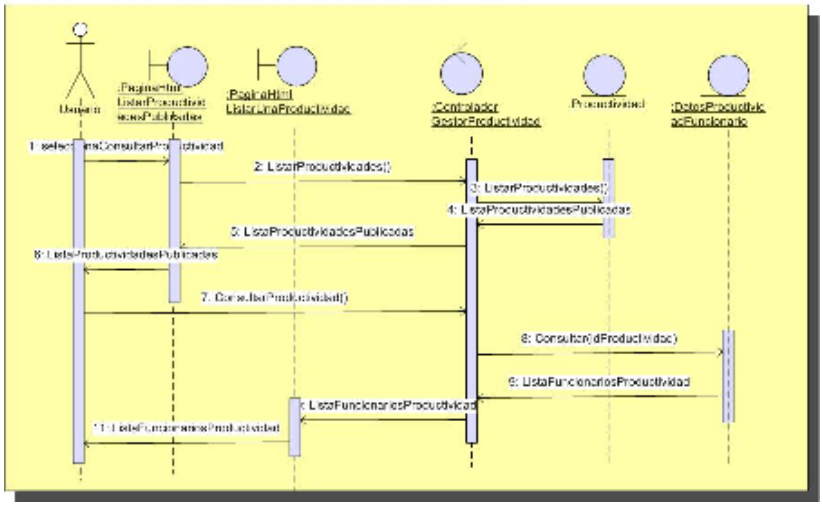
Como podemos comprobar, hemos añadido claves artificiales a todas las tablas. Las claves anteriores, tendrán la restricción UNIQUE, ya que seguirán siendo claves candidatas.

En este caso, no hemos necesitado eliminar la herencia, puesto que no existe.

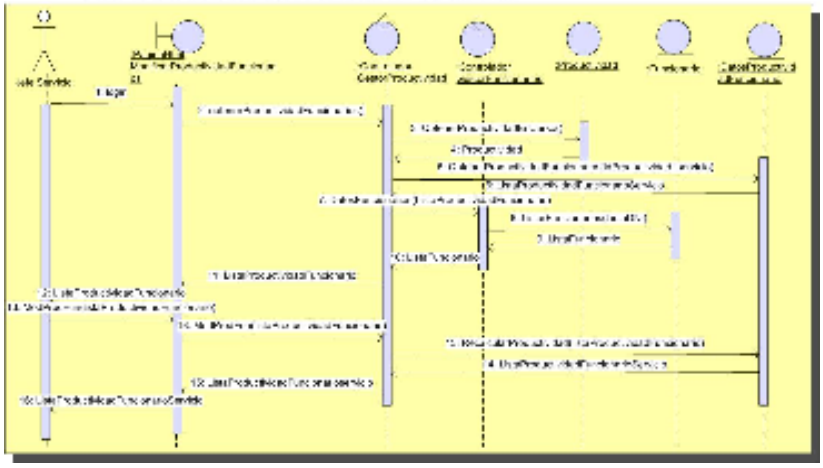
6.3.- Diagramas de colaboración

A continuación se expondrán los diagramas de colaboración para los distintos casos de uso:

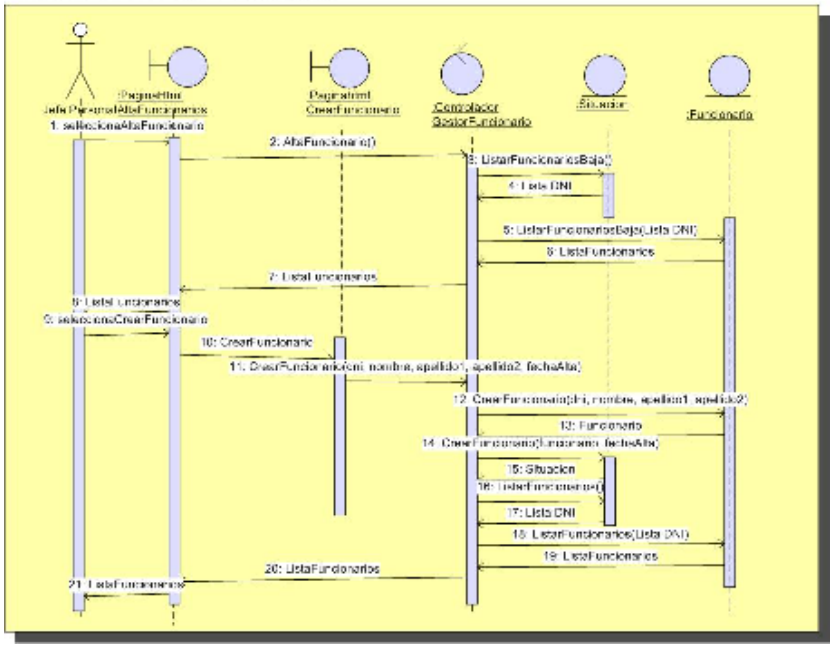
Caso de Uso 1. Consultar Productividad Publicada.



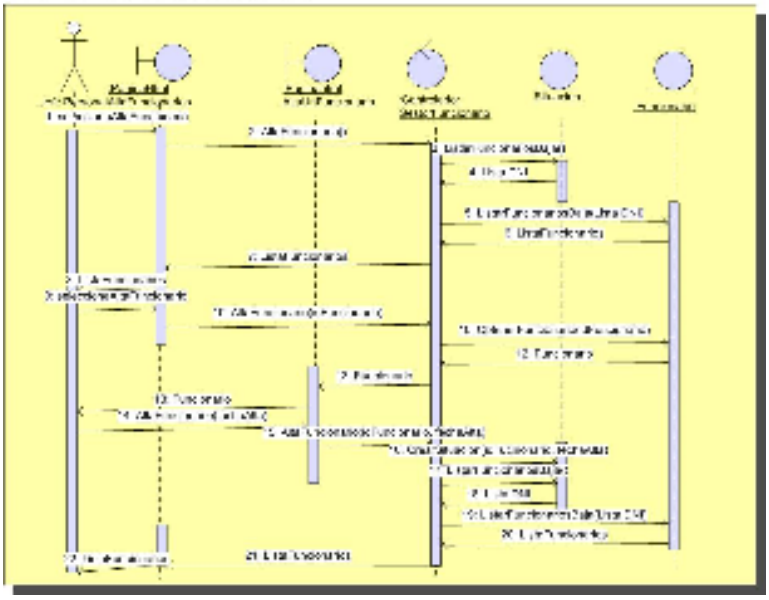
Caso de Uso 3. Modificar Productividad del Servicio



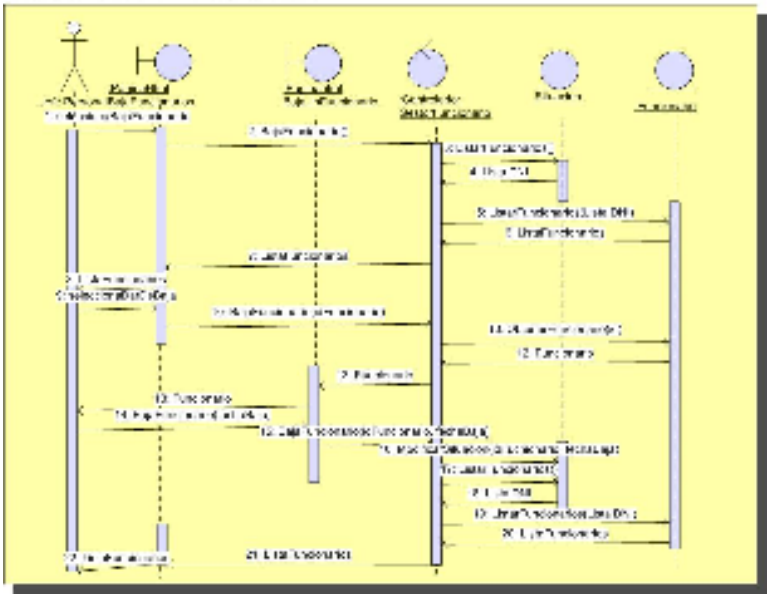
Caso de uso 4. Crear funcionario



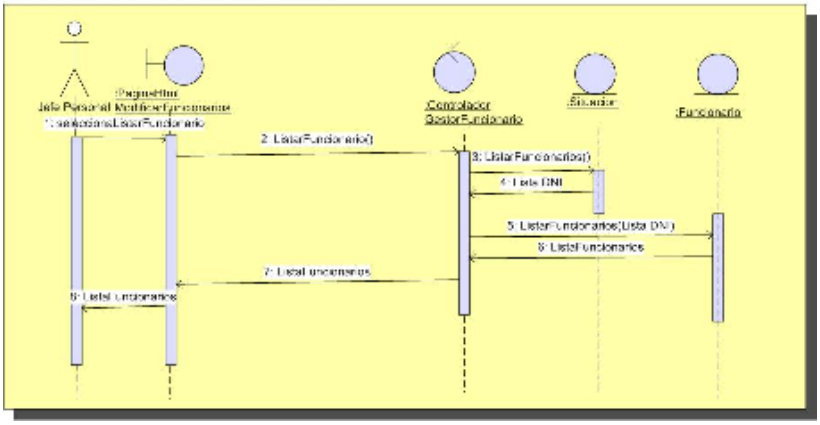
Caso de Uso 5. Alta funcionario



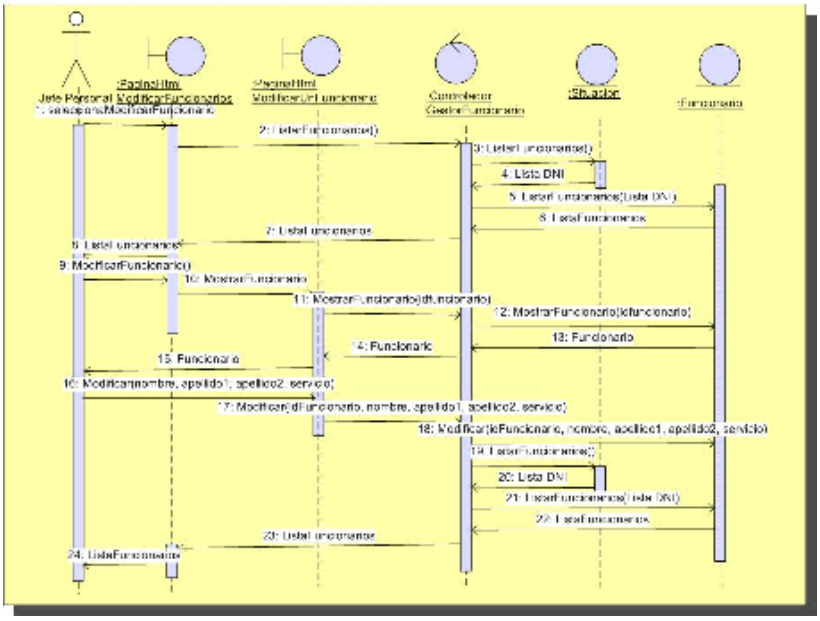
Caso de Uso 6. Baja funcionario



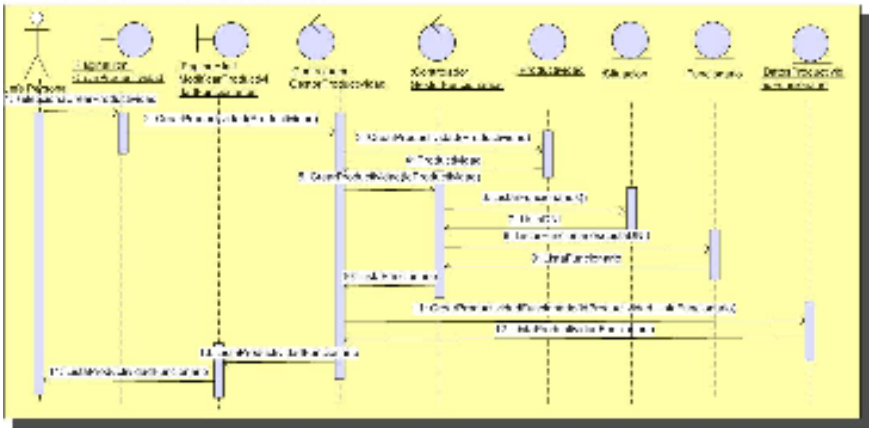
Caso de Uso 7. Listar funcionario



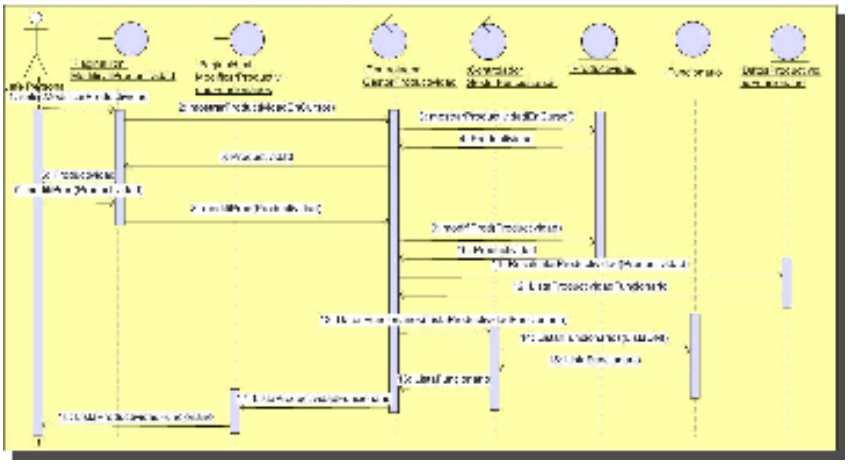
Caso de Uso 8. Modificar funcionario



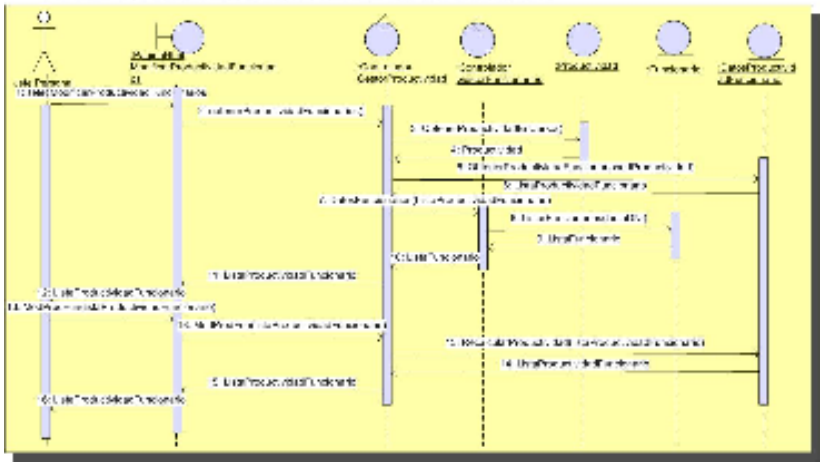
Caso de Uso 9. Crear Productividad



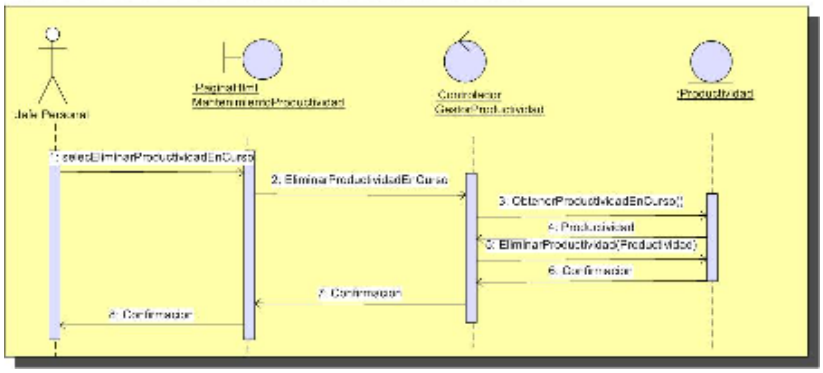
Caso de Uso 10. Modificar Productividad



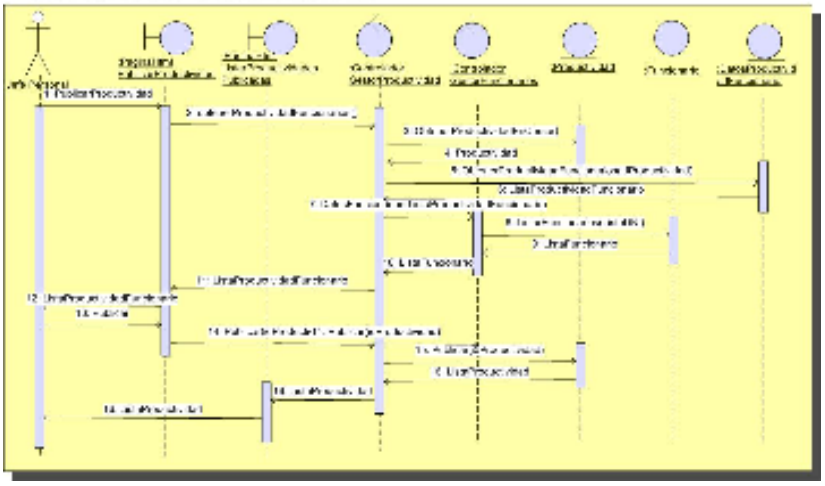
Caso de Uso 11. Modificar Datos Productividad Funcionarios



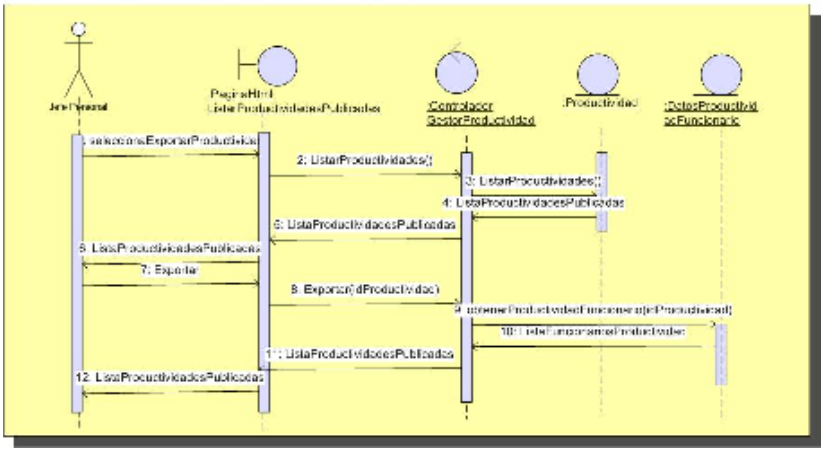
Caso de Uso 12. Eliminar Productividad En Curso



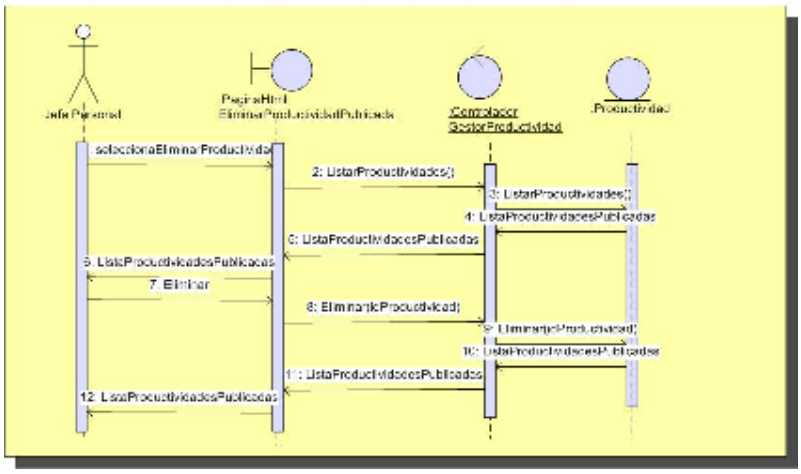
Caso de Uso 13. Publicar Productividad



Caso de Uso 14. Exportar Productividad.

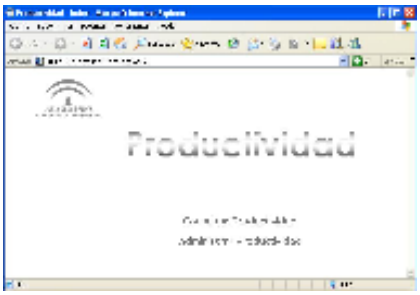


Caso de Uso 15. Eliminar Productividad Publicada.



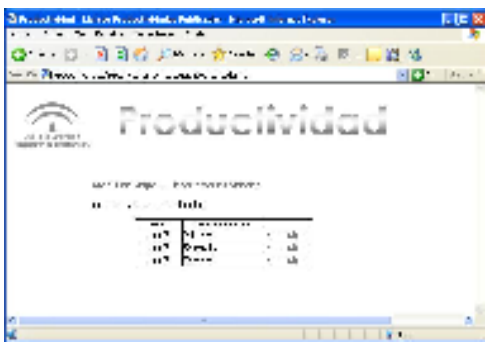
6.4.- Interfaz de usuario

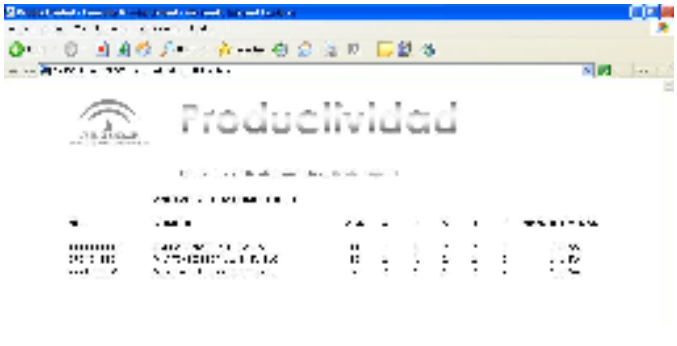
La pantalla de inicio de la aplicación, se muestra a continuación:



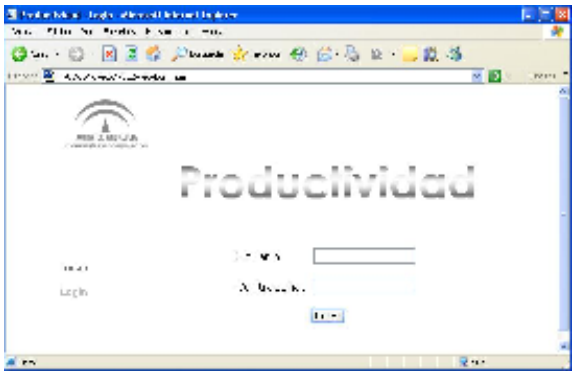
Desde aquí tenemos dos opciones, consultar las productividades que ya han sido publicadas, o entrar en la administración de la aplicación donde se nos pedirá un usuario y una contraseña.

Si elegimos la primera opción, nos aparecen las pantallas que presento a continuación:

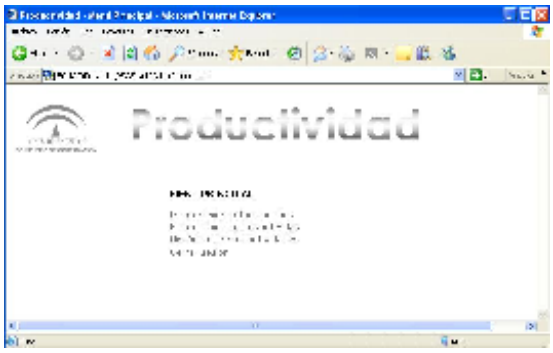




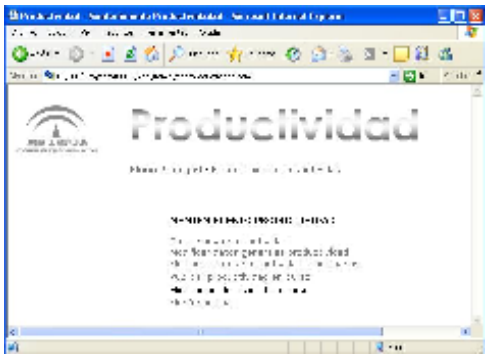
Es lo único que podemos hacer si no estamos registrados en la aplicación. Ahora bien, si intentamos administrar la aplicación, nos aparecerá la pantalla siguiente:

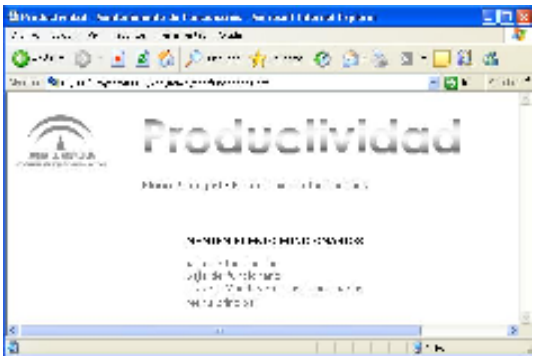


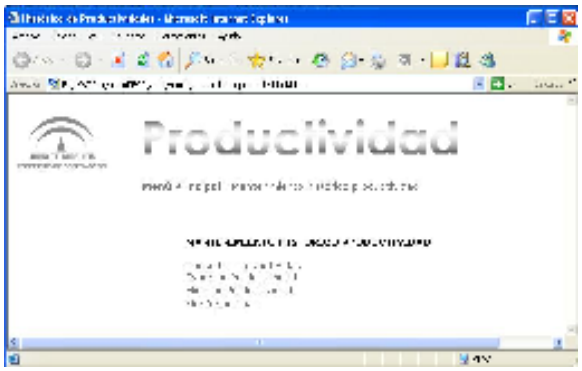
Al logarnos, existen dos posibles perfiles, el de Jefe de Personal (Administrador) y el de Jefe de Servicio. Veamos primero las opciones del Jefe de Personal, ya que es el más importante. Una vez logado, aparecerá un menú, como muestra la figura siguiente:



Como podemos observar, de aquí tenemos acceso a otros tres submenús, uno para el mantenimiento de funcionarios, otro para el mantenimiento de la productividad, y otro para el mantenimiento de las productividades, cada uno de ellos con una serie de opciones, que aparecen en las imágenes siguientes:

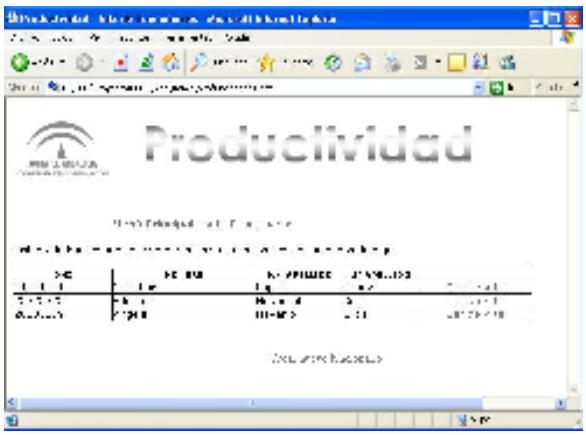


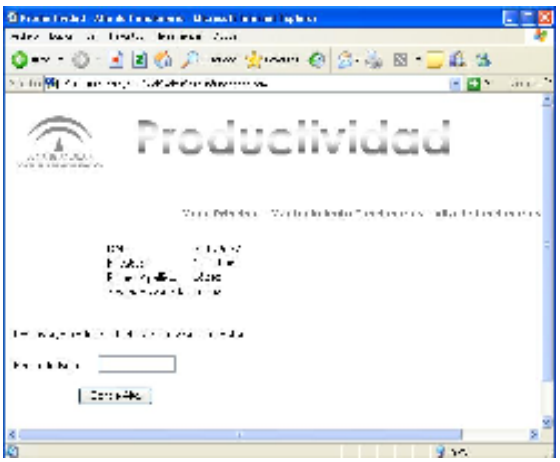


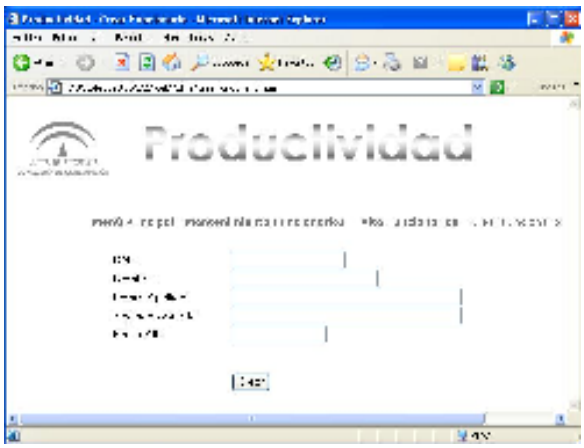


Las opciones del mantenimiento de funcionarios, que coinciden con los casos de uso números 4, 5, 6, 7 y 8, se ven en las pantallas que expongo a continuación:

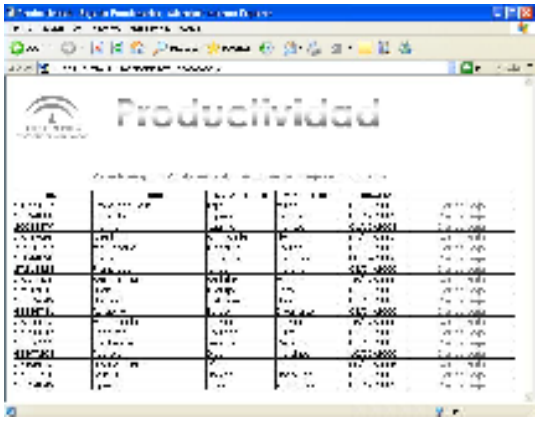
Para dar de Alta un funcionario, o crearlo, tendremos que pasar por las pantallas siguientes:

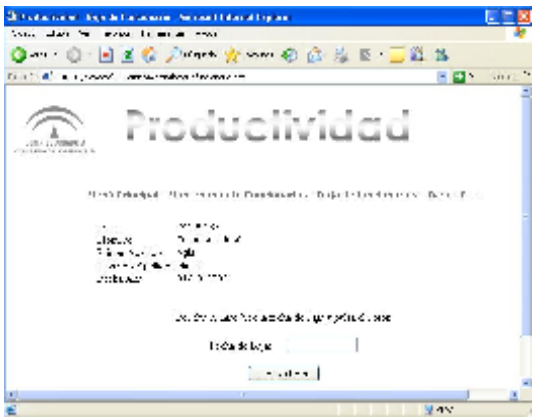




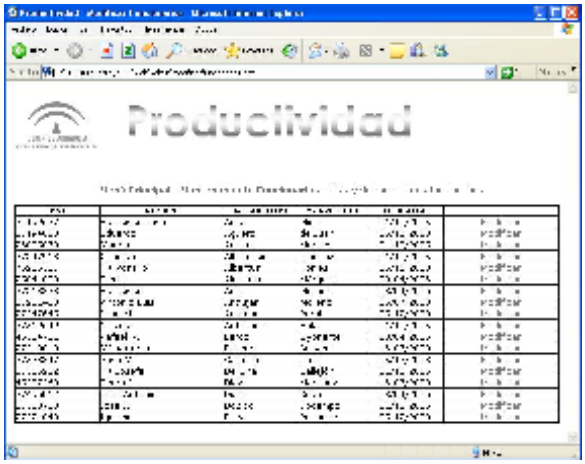


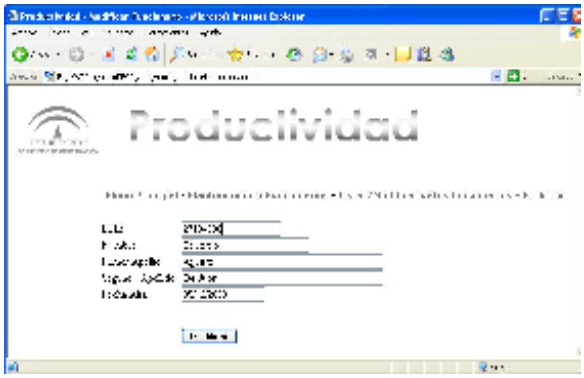
Si lo que queremos es dar de baja al funcionario, las pantallas serán:





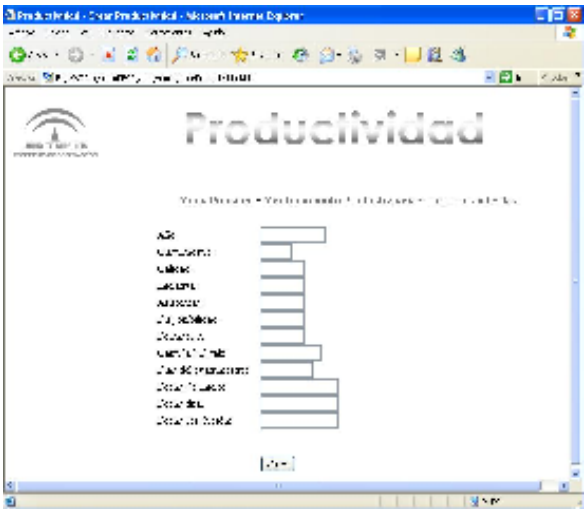
Por último, si lo que nos interesa es listar o modificar los datos de algún funcionario, las pantallas que nos presentará el sistema serán:



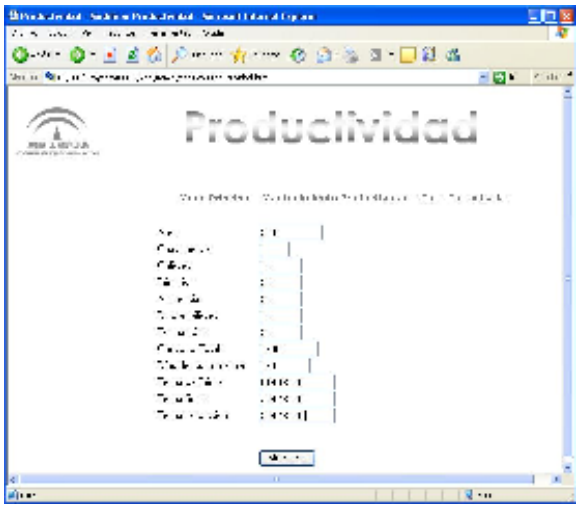


Continuamos ahora mostrando las opciones del menú del mantenimiento de productividad. En este caso, coincide con los casos de uso números 9, 10, 11, 12 y 13.

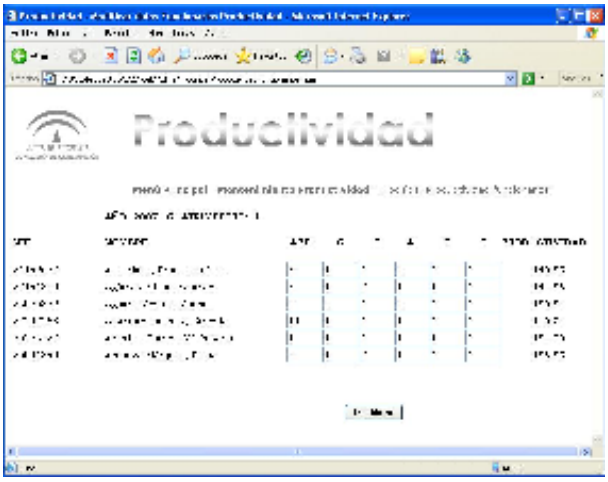
En el caso de Crear Productividad, la pantalla mostrada por el sistema será:



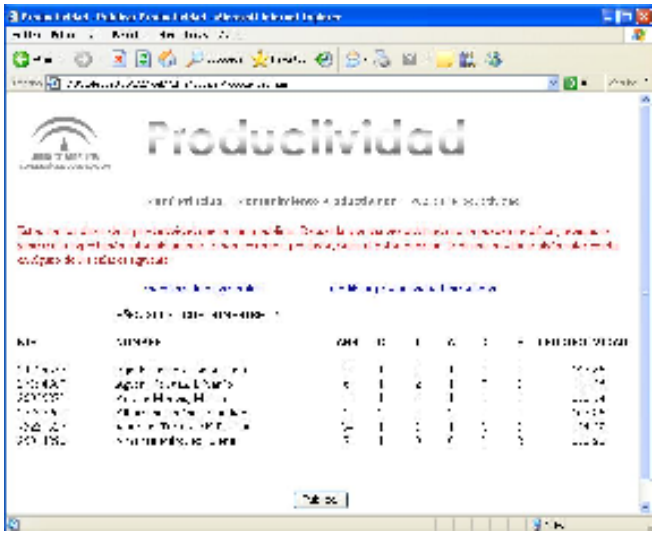
Modificar productividad, nos presenta una pantalla igual a la anterior, simplemente con los datos ya rellenos, para que podamos modificar el que nos interese.



Si queremos modificar los datos de los funcionarios para la productividad en curso, la pantalla que el sistema nos muestra es la siguiente:



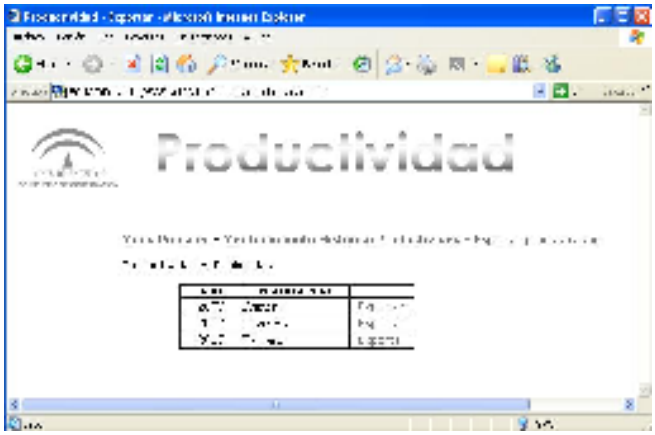
Si vamos a publicar la productividad, la pantalla es:



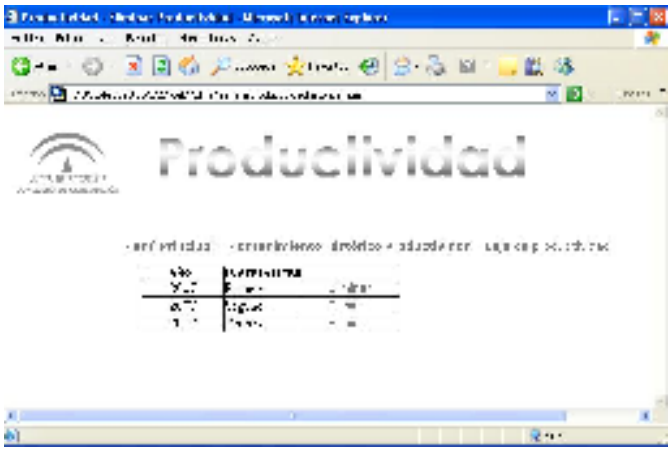
Si queremos eliminar la productividad en curso, no pasaremos por ninguna pantalla, simplemente nos saldrá un aviso para que confirmemos que efectivamente queremos eliminar la productividad en curso.

Por último, en el mantenimiento del histórico de productividades, tenemos los casos de uso números 14 y 15.

En el caso de que queramos exportar la productividad, tendremos la pantalla siguiente:



Y si lo que queremos es borrar alguna de las productividades ya publicadas:



Tan sólo nos queda ver el perfil del Jefe de Servicio. Recordamos que sólo puede modificar los datos de la productividad, para los funcionarios de su propio servicio, por tanto, al entrar, le aparece un mensaje de que no existe productividad en curso, o si hay productividad en curso, le aparecen los datos de los funcionarios de su servicio en productividad en curso para que pueda modificarla. La pantalla es la misma que le aparece al Jefe de Personal, con la única diferencia que ahora sólo muestra los funcionarios de un servicio.

6.5.- Diseño en tres capas. Patrones utilizados

Como ya hemos podido ir viendo a lo largo del desarrollo de este punto, el diseño de la aplicación, va a seguir un diseño de desarrollo en tres capas. Vamos a definir para cada una de ellas que elementos y que patrones vamos a utilizar:

6.5.1.- Capa de datos (Persistencia)

En cuanto a la capa de datos, usaremos MySQL, para el almacenamiento de datos, ya que en el momento actual, podemos decir que es uno de los mejores Gestores de Bases de Datos, si no el mejor, en cuanto al software libre se refiere.

En un primer momento, había pensado usar EJB 2.0, pero una vez estudiadas las tecnologías actuales existentes, veo que EJB 2.0 tiene deficiencias bastante grandes en cuanto a diseño, funcionalidad y rendimiento. Por tanto, para el acceso a los datos, usare cuatro beans de entidad, que seguirán la especificación EJB 3.0 y que por debajo, tendrán persistencia mediante Hibernate.

Estos cuatro beans, mapearan las cuatro tablas que hemos obtenido anteriormente. Habrá un bean de entidad para la Situación, otro para los Funcionarios, otro para la Productividad, y otro para los Datos.

6.5.2.- Capa de negocio

En nuestra aplicación, tendremos un EJB de entidad por cada clase persistente, y tendremos un bean de sesión sin estado, que implementarán el patrón "**Session Facade**".

Mediante el uso de este patrón, se encapsula la complejidad de las interacciones entre los objetos de negocio participantes en un flujo de trabajo. El Session Facade maneja los objetos de negocio y proporciona un servicio de acceso uniforme a los clientes.

El uso de este patrón, presenta las siguientes ventajas:

- Introduce una Capa Controladora para la Capa de Negocio.
- Expone un Interface Uniforme.
- Reduce el Acoplamiento e Incrementa la Manejabilidad.
- Mejora el Rendimiento y Reduce los Métodos Específicos.
- Proporciona Acceso Genérico.
- Centraliza el Control de Seguridad.

Además, para la comunicación entre la capa de negocio y la capa de presentación, usaré el patrón "**Value Object**".

Los value objects son objetos Java planos. Son clases que encapsulan una serie de datos útiles. Esto evita el realizar múltiples llamadas remotas en un EJB para guardar o recibir información. No implementan ningún método, sólo getters y setters para acceder a los métodos encapsulados.

6.5.3.- Capa de Presentación

Para la capa de presentación, vamos a usar el modelo 2, Model-View-Controller (MVC), que ya ha sido explicado detalladamente con anterioridad (ver apartado 4.4), con el objetivo de separar al máximo la presentación de la lógica de negocio.
En la capa de presentación he optado por utilizar una arquitectura basada en el modelo 2

7.- IMPLEMENTACION

7.1.- Elección de la tecnología

La implementación de la aplicación se ha desarrollado sobre JEE, y dentro de esta tecnología se ha decidido el uso de EJB 3.0, tratando la persistencia bajo Hibernate.
En cuanto a las decisiones de implementación, destacaré las siguientes:
Para manipular los datos de las tablas en la BD, diseñare los objetos que nos permitirán manipular estos datos. Lo realizare con EJB de entidad.
Para el diseño de los EJB de entidad, he tomado las siguientes decisiones:

- Utilizaré entity beans
- Los EJB siguen la especificación EJB 3.0.

Además voy a crear un EJB de sesión. Para la implementación del EJB de sesión he decidido lo siguiente:

- Como no es necesario que se mantenga ningún estado entre las diferentes llamadas de los clientes, he utilizado EJB de sesión sin estado.
- Utilizare value objects que nos servirán para intercambiar información entre las diferentes capas de la aplicación. En este caso, utilizaré los value objects para interactuar con los EJB de entidad y para pasar la información a la capa de presentación.

Para la capa de presentación he elegido utilizar una arquitectura basada en modelo 2. En esta arquitectura las JSP no invocan directamente a la capa de negocio, sino que todas las peticiones pasan por una servlet que se encarga de recibir las peticiones del usuario y para cada petición:

- Escoger qué método de la capa de negocio se debe ejecutar.
- Llamar al método de negocio y obtener los datos.
- Guardar los datos en un lugar accesible para la siguiente vista.
- Escoger qué vista hay que mostrar al usuario, en función de la invocación de negocio hecha y el resultado de ésta.

En cuanto a la seguridad, tenemos tres partes diferenciadas, una donde pueden acceder todos los usuarios, con lo cual no tiene ninguna protección, ni tan siquiera necesitan validarse. Y otras dos partes donde sólo tiene acceso el Administrador o los Jefes de Servicio. Para el acceso a estas otras dos partes de la aplicación necesitamos introducir un usuario y una contraseña. Todas las comunicaciones entre las JSP donde aparece la contraseña, está se muestra cifrada, además la contraseña también se guarda cifrada en la Base de Datos.
En todas las JSP, he introducido una cabecera que comprueba que la sesión y el usuario son correctos, además esta comprobación también se hace en el Servlet. De esta manera se evita, por ejemplo, que se copie la dirección de una página intermedia y se pueda acceder a ella sin habernos logado.

7.2.- Problemas surgidos debido a la tecnología y soluciones propuestas. Impacto sobre el diseño

El uso de EJB 3.0, ha resultado muy eficaz, y no he tenido que modificar el diseño en ninguno de sus aspectos.
El mayor problema encontrado a la hora de implementar la aplicación con EJB 3.0, es que al desplegar la aplicación, siempre me aparecían varios errores. Después de muchas horas de lectura, he encontrado un documento en el que textualmente se decía:
¡¡¡ Ojo, JEE 5 soporta inyección de dependencias en los Servlets, Filtros, Tags, ... es decir, no sería necesario hacer el loockup en el Servlet; pero JBoss 4.2.1.GA todavía no soporta esta funcionalidad (no es totalmente JEE 5 "compliant"). JBoss 4.2.1.GA sólo soporta inyección de dependencias entre EJBs. Luego veremos un ejemplo de ello !!!

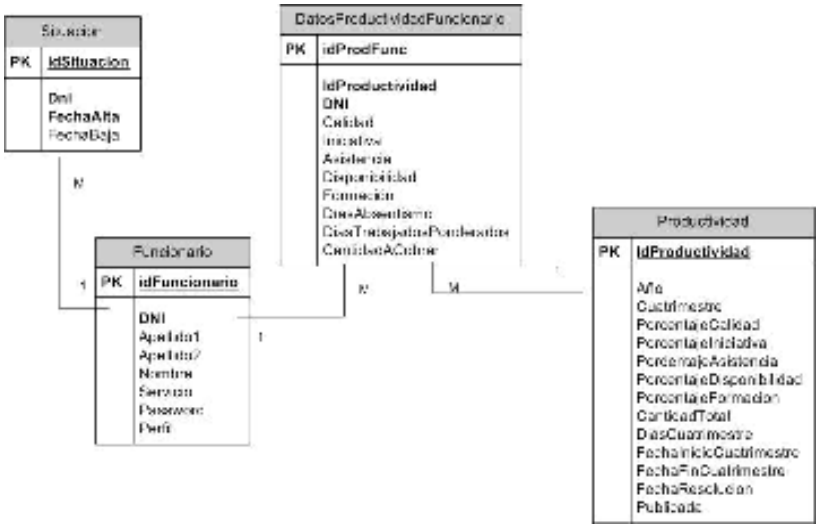
Como para la implementación del proyecto he usado NetBeans 5.5, que lleva integrado jboss-4.0.4.GA, he probado a hacer el loockup, y efectivamente todo ha empezado a funcionar correctamente.

El único impacto sobre el análisis y el diseño, es que en estos, tuvimos en cuenta que debíamos tratar la seguridad de la aplicación, pero no especificamos como.

Para tratar la seguridad de la aplicación, el nombre del usuario para acceder a la misma, va a ser el DNI, pero necesito otro campo, para almacenar la contraseña.

Además, como tenemos tres posibles roles, voy a utilizar otro campo, denominado "perfil", para almacenar los roles.

Una vez solucionado esto, el nuevo diagrama estático es el siguiente:



7.3.- Instalación de la aplicación

Para poner en funcionamiento la aplicación, necesitamos tener instalado:

- MySQL, que será nuestro gestor de BD.
- JBOSS, en mi caso en la versión 4.0.4. con soporte para EJB 3.0. (El equipo debe tener instalado el Runtime de Java)

El proceso de instalación es el siguiente:

- Desde MySQL[1] ejecutar el script denominado "creaciontablas.sql".
- Copiar los ficheros "jboss-ds.xml" y "productividad.ear" dentro de JBOSS, concretamente en la carpeta \server\default\deploy.
- Copiar en el directorios JBOSS_HOME\server\default\lib el conector de la base de datos, en mi caso, el fichero "mysql-connector-java-5.0.3-bin.jar"[2].

[1] Desde "MySQL Command Line Client" ejecutar: mysql> source ruta/creaciondetablas.sql
Por ejemplo, si el archivo está guardado en c:\Instala, escribiremos:
mysql> source c:/Instala/creaciondetablas.sql

[2] http://dev.mysql.com/downloads/connector/j/5.0.html

- copiar el fichero productividad.ear (la aplicación propiamente dicha), en la ruta JBOSS_HOME \server\default\deploy.

Para acceder a la aplicación, debemos abrir el navegador y poner la dirección, http://nombreServidor:8080/productividad.

7.3.1.- Fichero de instalación

Para facilitar la instalación, he creado un fichero denominado instala.bat

Para ejecutarlo, debemos abrir una ventana MS-DOS, situarnos en la carpeta donde se encuentra este fichero y escribir su nombre. Nos pedirá la contraseña del root de MySQL, y automáticamente se copiarán los ficheros necesarios y se ejecutará el script de creación de tablas.

Sólo hay que tener en cuenta que el fichero presupone que la instalación de MySQL se encuentra en C:\Archivos de programa\MySQL\MySQL Server 5.0\bin\. Si no es así, debemos editar el fichero bat y escribir la ruta donde se encuentra el MySQL.

7.3.2.- Consideraciones sobre la instalación

En la instalación, se crean una serie de registros en las tablas "funcionario" y "situación", para que podamos comprobar el funcionamiento de la aplicación.

En concreto, se crea:

- un usuario, con dni "1" y contraseña "123456" con perfil "Jefe de Servicio",
- y otro usuario con dni "27532673" y contraseña "digobal" con perfil "Jefe de Personal".

Además, se crea también un usuario de MySQL, con permisos para trabajar sobre las tablas creadas.

7.4.- Manual de usuario

Destacar también que he creado un manual para el Administrador, y otro para el Jefe de Servicio. Podemos descargar estos archivos desde el menú principal de cada una de las aplicaciones

8.- VALORACIONES FINALES

8.1.- Conclusiones

Creo que los objetivos del proyecto se han alcanzado con solvencia, y la realización de este proyecto me ha servido para darme cuenta que soy capaz de poner en práctica los

conocimientos adquiridos a lo largo de estos años de estudio, y además ver que soy capaz de afrontar un proyecto de cierta complejidad.

Uno de los objetivos que nos hemos marcado, es el análisis de las tecnologías existentes. He hecho un estudio detallado de algunas de estas tecnologías como son Corba, .NET, JEE y Servicios Web.

Una vez estudiadas estas tecnologías, he decidido desarrollar el proyecto con JEE, haciendo un estudio, más detallado aún, si cabe, sobre JEE.

Además, he profundizado en los distintos Frameworks que se pueden usar con JEE, implementando finalmente el proyecto con EJB 3.0 e Hibernate en la lógica de negocio y la persistencia.

Otro gran objetivo de este proyecto, era el análisis, diseño e implementación de un sistema de gestión y creación de la denominada "Productividad" en los Organismos Oficiales de la Junta de Andalucía.

Para la realización del análisis, he tenido que realizar entrevistas con el Jefe de Personal, con la encargada de la nómina, y sobre todo con el Secretario General, que ha sido quien finalmente ha dado su visto bueno para la implantación del sistema.

Tanto para el análisis como para el diseño, he tenido que repasar conceptos obtenidos en otras asignaturas, por lo que este proyecto me ha servido, no sólo para adquirir nuevos conocimientos si no también para afianzar conceptos que ya tenía adquiridos previamente.

Por último, para la implementación en sí, he usado una herramienta que no había utilizado nunca, como es NetBeans en su versión 5.5 (para la implementación con EJB 3.0), comprobando su facilidad de uso, y Eclipse con el IDE para JBOSS (para la implementación con EJB 2.0)

Además, me ha servido para realizar una comparativa totalmente práctica de las tecnologías EJB 2.0 y EJB 3.0. Y aunque ha supuesto un gran esfuerzo realizar la implementación en ambas tecnologías, creo que ha valido la pena, ya que los conocimientos adquiridos han sido muy grandes.

Para la presentación, aunque he estudiado y analizado los Frameworks Struts y JSF entre otros, y he realizado algunos ejemplos sencillos, he decidido finalmente utilizar un Servlet (Front End) como controlador, que decide que página JSP servir para cada operación.

También quiero resaltar el hecho, que ya está implantada la aplicación, en la Delegación del Gobierno en Almería, y ya se ha utilizado para comprobar que la productividad del segundo cuatrimestre del año 2007 es correcta.

8.2.- Mejoras para próximas versiones

Como en la mayoría de aplicaciones, siempre hay cosas que se pueden mejorar. Una vez probada la aplicación, por parte de los usuarios, se me han sugerido algunas mejoras:

- Crear otro perfil para el encargado de nóminas, de tal forma, que el encargado de las nóminas pueda exportar directamente los archivos necesarios.
- Poder dar de baja una productividad publicada, pero que no sea borrada. Es decir, la productividad publicada, no se podrá consultar por los usuarios, pero si por el Jefe de Personal.
- Los administradores me solicitan también, una opción en el menú principal para poder acceder a su cambio de contraseña directamente.

BiBliografía

http://javahispano.org/contenidos/es/comparativa_de_frameworks_web/;jsessionid=C507275DEEDCAC594D79DB1FCE1ADD04

http://www.springhispano.info/portal/?q=node/21

http://www.laliluna.de/tutorial-struts-eclipse-espanol.html

http://www.programacion.com/articulo/inukisoft/

http://www.programacion.net/java/tutorial/patrones2/

http://javaejb.osmosislatina.com/curso/patrones.htm

http://www.javahispano.org/contenidos/es/manual_hibernate/

http://www.elholgazan.com/2007/10/comparacin-entre-spring-y-ejb-30.html

http://www.jtech.ua.es/jornadas/06/charlas/Persistencia.pdf

http://emanuelpeg.hostinggratisargentina.com/#_Toc146392571

Apuntes de Ingeniería del Software y Componentes en Sistemas Distribuidos.

Apuntes de Ingeniería del Software orientado al objeto.

ANEXO 1: IMPLEMENTACIÓN CON EJB 2.0

A1.1.- Introducción

Debido a la planificación temporal, y como consecuencia de la aparición de varios problemas a la hora de implementar la aplicación con EJB 3.0, he decidido implementar, para poder cumplir el plazo de entrega de la PEC 3, con EJB 2.0.

A1.2.- Decisiones de la Implementación

En cuanto a las decisiones de implementación, destacaré las siguientes:

Para manipular los datos de las tablas en la BD, usaré EJB de entidad.

Para el diseño de los EJB de entidad, he tomado las siguientes decisiones:

- Utilizaré entity beans
- Los EJB siguen la especificación EJB 2.0.
- Estos EJB tendrán persistencia gestionada por el contenedor
- Para las relaciones entre los EJB de entidad utilizaré CMR.
- Los EJB de entidad tendrán interfaces de acceso local.
- Se utilizan las opciones por defecto de JBoss a la hora de hacer mapeos entre la BD y los EJB, lo que implica no tener que crear el fichero jboss.xml para especificar estos mapeos.

Para la implementación de los EJB he desarrollado un EJB de entidad para cada tabla existente y dos EJB de sesión para encapsular el acceso a los EJB de entiddad. He utilizado interfaces locales para definir todos los EJB de entidad ya que la especificación de EJB dice que se debe utilizar acceso local para los EJB origen de una relación CMR.

Para definir las relaciones existes entre las distintas tablas, he tenido que definir las relaciones en el descriptor.

Para la implementación de los EJB de sesión he decidido lo siguiente:

- Como no es necesario que se mantenga ningún estado entre las diferentes llamadas de los clientes, utilizaremos EJB de sesión sin estado.

- Utilizare value objects para intercambiar información entre las diferentes capas de la aplicación. En este caso, utilizaré los value objects para interactuar con los EJB de entidad y para pasar la información a la capa de presentación.

Los EJB de entidad se tienen que embalar para el despliegue con un descriptor de despliegue XML que describe el componente y su esquema abstracto de persistencia. El descriptor se denomina ejb-jar.xml

En cuanto a la capa de presentación he elegido utilizar una arquitectura basada en modelo 2, exactamente igual, que en la implementación realizada con EJB 2.0, ya que la capa de presentación no la he modificado.

A1.3.- Problemas surgidos debido a la tecnología y soluciones propuestas. Impacto sobre el diseño

Problema 1

Uno de los problemas surge en las consultas que podemos realizar, ya que estas consultas, no admiten, por ejemplo consultas anidadas.

EJB 2.0, nos obliga a utilizar un descriptor, ejb-jar.xml, donde debemos definir las entidades, las consultas que haremos en esas entidades, y las relaciones existentes entre ellas

Al no poder usar consultas anidadas, si puedo obtener los funcionarios dados de alta en un momento dado, ya que la fecha de baja está a "null", pero, sin embargo, no puedo obtener los funcionarios que no está dados de alta en un momento dado.

Solución:

La elección de utilizar. Además también debemos definir en este fichero cuál es la ruta de los beans de sesión, y sobre que entidades van a trabajar.

Para solucionar esto, he añadido un campo a la tabla "Funcionario", denominado "Activo", que tendrá el valor "S", si el funcionario está dado de alta, y "N" si el funcionario no está dado de alta. Este campo es totalmente transparente al usuario

Problema 2

En el análisis y el diseño, tuvimos en cuenta que debíamos tratar la seguridad de la aplicación, pero no especificamos como.

Solución:

Accederemos a la aplicación, con el DNI, pero necesito otro campo, para almacenar la contraseña, este campo se denomina "password".

Además, necesito saber que rol tiene el usuario que se valida, por lo que he añadido otro campo denominado "perfil". Ambos campos pertenecen a la tabla "Funcionario", como se puede ver en el diagrama estático de la página siguiente.

Problema 3

Quizás el mayor problema con el que me he encontrado es el uso de claves autonuméricas, ya que con EJB 2, no podemos gestionar este tipo de claves. En nuestro caso, tenemos dos claves autonuméricas que se corresponden con la clave de la tabla "Funcionario", y con la clave de la tabla "Situación".

Solución:

Para solucionar este problema, he tenido que crear otra tabla, que he denominado "Generador Clave" y que se compone tan sólo de dos campos, el primero con el nombre de la tabla a la que corresponde el índice, y el segundo campo con el índice por el que vamos.

Problema 4

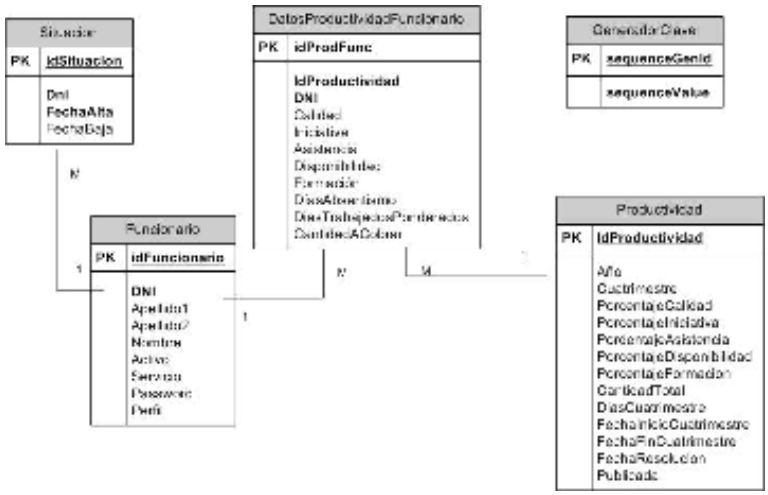
Otro de los problemas encontrados, ha sido las restricciones UNIQUE, ya que no he encontrado como poder describir en el descriptor que un campo es unique. Este problema lo he observado en el campo "dni" de la tabla funcionario, por lo que al intentar crear un funcionario cuyo "dni" existe, se produce un error del tipo:

com.mysql.jdbc.exceptions.MySQLIntegrityConstraintViolationException: Duplicate entry '1' for key 2.

Solución:

Este error, lo tengo controlado mediante un bloque "Try – Catch". No se inserta el registro y se presenta una pantalla de error, continuando el funcionamiento normal de la aplicación, pero sin embargo, aparece esa excepción en la consola de JBOSS.

Una vez estudiados y resueltos los problemas, nos queda el siguiente diagrama estático:



A1.4.- Instalación de la aplicación

Para poner en funcionamiento la aplicación, necesitamos instalar:

- MySQL, que será nuestro gestor de BD.
- JBOSS, en mi caso en la versión 4.0.4. (El equipo debe tener instalado el Runtime de Java)

El proceso de instalación es el siguiente:

- Desde MySQL[3] ejecutar el script denominado "creaciontablas.sql".
- Copiar los ficheros "mysql-ds.xml" y "productividad.ear" dentro de JBOSS, concretamente en la carpeta \server\default\deploy.
- Copiar en los directorios JBOSS_HOME\server\default\lib y JBOSS_HOME \lib el conector de la base de datos, en mi caso, el fichero "mysql-connector-java-5.0.3-bin.jar"[4], y el fichero iText-2.0.7.jar, con la librería iText[5].
- Copiar el fichero login-config.xml en el directorio "server/default/conf"

[3] Desde "MySQL Command Line Client" ejecutar: mysql> source ruta/creaciondetablas.sql
Por ejemplo, si el archivo está guardado en c:\Instala, escribiremos:
mysql> source c:/Instala/creaciondetablas.sql

[4] http://dev.mysql.com/downloads/connector/j/5.0.html

[5] http://www.lowagie.com/iText/download.html

- Copiar el ficherio “standardjbosscmp-jdbc.xml” en “server/default/conf”
- Copiar el fichero productividad.ear (la aplicación propiamente dicha), en la ruta JBOSS_HOME \server\default\deploy.

Editar, modificando la etiqueta CallByValue a true (por defecto, se encuentra en false) los XML siguientes:

• C:\jboss-4.0.4.GA\server\default\deploy\ear-deployer.xml
• C:\jboss-4.0.4.GA\server\default\conf\jboss-service.xml

Para acceder a la aplicación, debemos abrir el navegador y poner la dirección, http://nombreServidor:8080/productividad.

A1.5.- Mejoras de EJB 3.0 respecto a EJB 2.0

El objetivo principal de EJB 3.0, fue simplificar el desarrollo de aplicaciones java y estandarizar el API de persistencia. Para ello, presenta una serie de novedades como son:

- No precisa de contenedor. Funciona tanto en J2EE como J2SE
- Puede usarse independientemente del resto de los servicios (transacciones, seguridad, ..)
- Metadata con Anotaciones: Se eliminan los deployment descriptors, se reduce drásticamente el número de clases a crear
- Configuraciones por defecto: reducen la cantidad de configuración a especificar
- No Intrusión: los objetos a persistir (Entity Beans) no necesitan implementar interfaces EJB
- Herencia y poliformismo
- Lenguaje de consulta (EJBQL) mejorado: inner and outer join, operaciones bulk, sql nativo.

Gracias a estas novedades, EJB 3.0, aporta las siguientes ventajas:

- Testing
- Simplicidad: una única clase para declarar la persistencia (con la ayuda de anotaciones)
- Facilidad de aprendizaje
- Transparencia: las clases a persistir son simples POJOs
- No hay restricciones con respecto a relaciones entre objetos (herencia, poliformismo)

Si nos basamos en las dos implementaciones que hemos llevado a cabo, podemos comprobar que hay una gran diferencia entre las dos especificaciones. Es mucho más sencilla la implementación con EJB 3.0, además no se produce ningún impacto con respecto al diseño que habíamos propuesto, ya que podemos realizar consultas SQL con mayor facilidad, y sobre todo, nos hemos ahorrado todos los descriptotes que necesitábamos con EJB 2.0.