Análisis, Diseño e implementación de un sistema B2E en una administración pública.

José Lorenzo Rodríguez Currais
Enginyeria en Informàtica

Nom Consultor: JAVIER FERRO GARCIA

14 de Enero de 2008

1. RESÚMEN

Se pretende implementar un sistema B2E (abreviatura de la expresión **business to employee** , de la Empresa a empleado) que permita a través de un cliente lWeb, de forma ágil, rápida, segura y sencilla, realizar los procesos de comunicación más habituales entre un organismo público y sus profesionales específicamente en el ámbito de tramitación administrativa.

Este portal, integrará una serie de servicios basados en una arquitectura SOA (orientada a servicios WEB) que independizará los sistemas operacionales (sistemas de gestión de personal y nóminas) de la consulta y la gestión de peticiones del propio sistema B2E. Para nos basaremos datos consolidados a través de Sistemas DataMart de la propia organización.

La implementación se ha realizado utilizando código Java utlizando una arquitectura J2EE y que ha permitido construir una aplicación robusta, escalable y reutilizable. Se han utilizado una serie de patrones de diseño y basados en MVC y Struts. Estas decisiones han permitido desarrollar un producto con una separación de funcionalidades entre los diferentes componentes todo con la finalidad de desacoplar la capa de acceso a datos de la capa de lógica de negocios y de la capa de presentación y así proveer de una arquitectura más mantenible.

2. INTRODUCCIÓN

Este proyecto se inicia con la necesidad de de desarrollar una aplicación WEB de gestión que presente una arquitectura escalable, robusta y reutilitzable.

Se planteará la construcción del sistema utilizando los conocimentos adquiridos a lo largo del estudio de las diferentes asignaturas y aplicado todos aquellos aspectos formales necesarios para la obtención de un producto técnicamente solvente.

La utilización de las características de J2EE como el desacoplamiento de componentes y el reaprovechamiento a partir de un patrón basado en MVC utilizando el framework Struts, nos permite llevar a la práctica de forma conjunta y sincronizada todos aquellas técnicas con las que que se han tenido contacto en las diversas etapas de la carrera.

Como objetivo particular y técnico el desarrollo de este proyecto me ha permitido reconocer y aplicar las ventajas y características del desarrollo de aplicaciones WEB basadas en J2EE, además de implantar el desarrollo a través las técnicas más habituales de esta arquitectura, se han incoporado patrones aprovenchado las ventajas inherentes a su utilización; tanto en el tiempo de análisis como en la de diseño e implementación.

3. OBJETIVOS

El objetivo principal será el proporcionar una **herramienta de consulta**, que les permita interactuar con la organización de forma sencilla, rápida y directa. De esta manera, los empleados podrán acceder a un portal en el que dispongan de herramientas para poder consultar información personal y desde el cual podrán realizar, además, procedimientos de solicitud de cambio y actualización de sus datos personales, familiares, laborales ...

De la misma forma, en las plataformas disponibles en el mercado de soluciones “Self-Service” de recursos humanos, aparece el concepto de portal para el “manager” en el que el responsable de un departamento podrá obtener cierta información de interés de los profesionales de su departamento (período de vacaciones, días libres...).

Dado que este sistema estará limitados a la consulta, modificación y actualización personalizada de información por parte de los empleados, un aspecto fundamental al que debe de dar respuesta el diseño y desarrollo del mismo es la **seguridad de acceso** a esta información, es decir, asegurar que una persona accede única y exclusivamente a su información y no a la de ningún otro.

Otros objetivos del sistema que podemos destacar son:

- Posibilitar la *comunicación directa* entre los empleados y la administración de forma ágil, rápida y sencilla
- *Cubrir las necesidades funcionales* demandadas por los empleados dividiendo estas necesidades en tareas de consulta (consulta de datos personales, históricos, simulación de nóminas...) y tareas de gestión administrativa (solicitud de modificación de datos personales, solicitud de vacaciones o días de libre disposición...)
- *Adaptar los procesos administrativos* de la Organización a las nuevas tecnologías
- *Evitar trabajos internos* realizados de forma manual
- *Proporcionar a los profesionales una herramienta por y para su servicio*: el sistema estará basado en el servicio al del empleado
- *Permitir el acceso a la información en cualquier momento* y desde cualquier punto de la organización: incrementar la autogestión de los profesionales

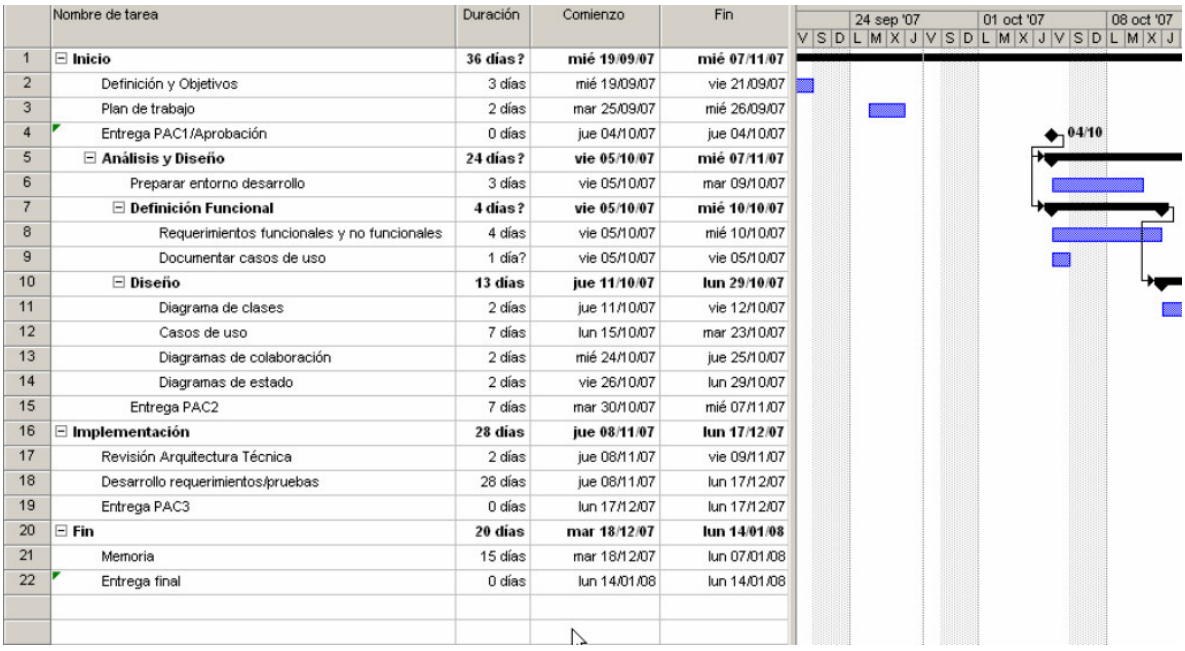
- El sistema debe de *integrarse totalmente con los sistemas de información de gestión* de personal de la propia organización (personal y nóminas, dietas...)
- Implementación del sistema en un *entorno seguro* que identifique unívocamente el acceso a la información: un empleado sólo podrá acceder a su información y nunca a la de ningún otro. En este punto se hace básico el cumplimiento de la normativa establecida por la LOPD que debe de cubrir el sistema
- Permitir la consulta en tiempo real de la *información personal y económica* de los empleados.
- *Unificar y simplificar la consulta de los datos económicos* de los empleados, independientemente de los centros dependientes de la organización en los que haya trabajado a lo largo de su vida laboral

Por todas las características anteriores, los sistemas de autoservicio permiten un significativo ahorro en tiempo y recursos, una descentralización de tareas, automatización de los procesos de aprobación, actualización de la información y mejora del clima laboral de la empresa.

Como objetivos específicos, podemos detallar dentro de la elaboración de este proyecto:
- Plan de trabajo.
- Definición funcional
- Definición de arquitectura Técnica
- Implementación del sistema

4. PLAN DE TRABAJO

La temporización del proyecto se regirá dependiendo de las siguientes fechas:



	Nombre de tarea	Duración	Comienzo	Fin
1	**Inicio**	**36 días?**	**mié 19/09/07**	**mié 07/11/07**
2	Definición y Objetivos	3 días	mié 19/09/07	vie 21/09/07
3	Plan de trabajo	2 días	mar 25/09/07	mié 26/09/07
4	Entrega PAC1/Aprobación	0 días	jue 04/10/07	jue 04/10/07
5	**Análisis y Diseño**	**24 días?**	**vie 05/10/07**	**mié 07/11/07**
6	Preparar entorno desarrollo	3 días	vie 05/10/07	mar 09/10/07
7	**Definición Funcional**	**4 días?**	**vie 05/10/07**	**mié 10/10/07**
8	Requerimientos funcionales y no funcionales	4 días	vie 05/10/07	mié 10/10/07
9	Documentar casos de uso	1 día?	vie 05/10/07	vie 05/10/07
10	**Diseño**	**13 días**	**jue 11/10/07**	**lun 29/10/07**
11	Diagrama de clases	2 días	jue 11/10/07	vie 12/10/07
12	Casos de uso	7 días	lun 15/10/07	mar 23/10/07
13	Diagramas de colaboración	2 días	mié 24/10/07	jue 25/10/07
14	Diagramas de estado	2 días	vie 26/10/07	lun 29/10/07
15	Entrega PAC2	7 días	mar 30/10/07	mié 07/11/07
16	**Implementación**	**28 días**	**jue 08/11/07**	**lun 17/12/07**
17	Revisión Arquitectura Técnica	2 días	jue 08/11/07	vie 09/11/07
18	Desarrollo requerimientos/pruebas	28 días	jue 08/11/07	lun 17/12/07
19	Entrega PAC3	0 días	lun 17/12/07	lun 17/12/07
20	**Fin**	**20 días**	**mar 18/12/07**	**lun 14/01/08**
21	Memoria	15 días	mar 18/12/07	lun 07/01/08
22	Entrega final	0 días	lun 14/01/08	lun 14/01/08

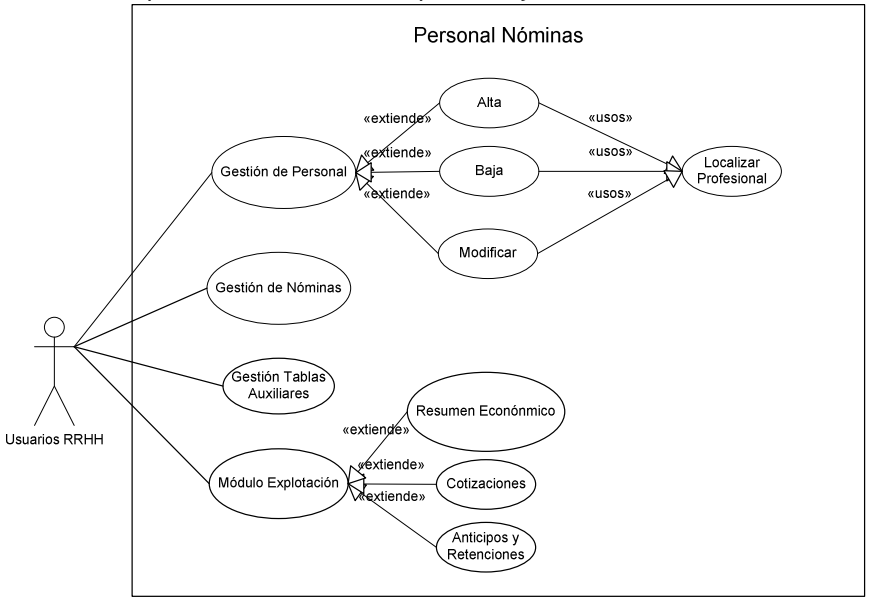
5. ESPECIFICACIONES Y ANÁLISIS DE REQUERIMIENTOS

5.1. Descripción lógica del sistema actual

La organización cuenta con un área de RRHH que mantiene un sistema operacional que permite la gestión de los recursos humanos vinculados a la empresa. Este sistema está accesible en la intranet corporativa y su arquitectura está basada en Cliente/servidor con un BD en Informix.

Además de este sistema, la organización mantiene un LDAP basado en el directorio activo de Microsoft, dando acceso a todos los recursos de la organización (correo, aplicaciones, impresoras, ficheros, etc). Este directorio ha sido extendido con el campo "CodPn" (código personal), que permite asociar a cada cuenta del directorio activo conl el código personal que gestiona la aplicación de RRHH. Así pues, dentro de nuestro B2E, podremos asociar la información a través del directorio activo y relacionarlo con la BD de RRHH

La aplicación de Gestión de Personal y Nóminas es utilizada por usuarios del departamento de RRHH y se trata de una aplicación centralizada que trabaja contra una base de datos local.



Entre las funcionalidades de la aplicación es posible realizar:

- *Gestión de Personal*: mantenimiento (Alta/Baja/Modificación) de los profesionales vinculados laboralmente .
- *Gestión de Nóminas*: mantenimiento de la información económica (nóminas) de los profesionales pertenecientes a la organización.
- *Gestión Tablas Auxiliares*: mantenimiento de las tablas auxiliares del sistema
- *Módulo Explotación*: módulo de consulta sobre la información del sistema de gestión de personal y nóminas

5.2. Especificaciones de los requisitos

En este apartado se consigna todos los requisitos funcionales y no funcionales (rendimiento, seguridad, implantación y disponibilidad del sistema) del proyecto.

5.2.1. Requisitos Funcionales

El acceso al sistema requerirá a los usuarios que se autentiquen, proporcionando los datos de su cuenta de usuario en el dominio (*Login* y *Clave*). Además, para el acceso de usuarios desde *Internet*, éstos deberán de disponer también de un certificado digital de usuario válido expedido por una entidad certificadora válida. Una vez proporcionados los datos de la cuenta de usuario se realizará la validación del usuario en el *Directorio Activo* del dominio, utilizado como repositorio de usuarios. El último paso para permitir el acceso al sistema será verificar que la cuenta de usuario dispone de información del campo CodPn de la cuenta de usuario.

Los requisitos funcionales del sistema serán:

A) RF-001. Multi-idioma

La aplicación implementará multi-idioma en todas sus funcionalidades. El usuario podrá establecer el idioma con el que desea utilizar la aplicación.

B) RF-002. Ayuda en línea

Todas las funcionalidades de la aplicación dispondrán de la funcionalidad de ayuda en línea que permitirá acceder desde cualquier sección de la aplicación, a la ayuda disponible para ese apartado en particular.

C) RF-003. Aviso Legal

Funcionalidad que permitirá visualizar las condiciones de uso del portal que han de seguir los usuarios del sistema.

D) RF-004. Ayuda

Funcionalidad que permitirá abrir en un navegador o guardar en el equipo el documento que contiene el *Manual de Usuario* completo de la aplicación. El archivo en el que se proporcionará el manual de usuario tendrá formato PDF y podrá ser enviado a la impresora.

E) RF-005. Presentación

Página de presentación en la que se muestra información acerca del último acceso realizado por el profesional al portal web.

F) RF-006. Consulta Ficha Personal

Funcionalidad de consulta que permitirá visualizar los datos personales del profesional.

G) **RF-007. Consulta resúmen de Datos Económicos**

Funcionalidad de consulta que permite visualizar la información económica del profesional. Esta consulta permitirá obtener, para un ejercicio dado, un listado resumen con la información económica del profesional. El listado resultado de la búsqueda podrá ser enviado a la impresora.

H) RF-008. Consulta de Datos Económicos – Nómina

Funcionalidad de consulta que permitirá realizar la búsqueda entre las nóminas del profesional, permitiendo aplicar filtros de búsqueda para un intervalo de fechas. Esta consulta permitirá acceder a la información de detalle de una nómina del profesional, en el que se detallarán los conceptos retribuidos. El listado de nóminas resultado de la búsqueda y el detalle de una nómina podrá ser enviado a impresora. La información de las nóminas del profesional será consultada directamente en el sistema de *Gestión de Personal y Nóminas* de la organización

I) RF-009. Consulta de Datos Económicos – Cotizaciones SS

Esta consulta permitirá acceder a la información de detalle de las cotizaciones a la Seguridad Social aportadas por la empresa en las nóminas del profesional. Esta funcionalidad desglosará el detalle de cotizaciones realizadas de forma mensual por ejercicio. El informe de Cotizaciones SS será similar al informe enviado por la Tesorería General de la Seguridad Social, en el que se detallan las cotizaciones de cualquier profesional. El informe con la información de Cotizaciones SS podrá ser impreso.

J) RF-010. Consulta de Permisos y Ausencias

Funcionalidad de consulta que permitirá obtener un listado con el histórico de los permisos y ausencias laborales del profesional, permitiendo aplicar diferentes filtros de búsqueda (motivo, intervalo de fechas y/o centro de gestión).

K) RF-011. Consulta de Datos Económicos – Anticipos y Retenciones

Esta consulta permitirá acceder a la información de detalle de los anticipos y retenciones. Esta funcionalidad permitirá aplicar diferentes filtros de búsqueda como puede ser un tipo, ejercicio, intervalo de fechas y/o importe, visualizar el listado de elementos que cumplen los filtros de búsqueda y acceder al detalle de un anticipo/retención concreto. El listado resultado de la búsqueda podrá ser impreso, así como el detalle de un anticipo/retención.

L) RF-012. Solicitud de Cambio de Datos Personales

Funcionalidad que permitirá proponer cambios a los datos personales del profesional.

5.2.2. **Requisitos no funcionales**

a) **RNF-001**: Será necesario verificar el tiempo de respuesta del sistema utilizando diferentes tipos de conexión (módem, ADSL...).

b) **RNF-002:** en las funcionalidades que accedan a información de nivel alto se implementarán las políticas y procedimientos de seguridad establecidos por la legislación actual.

c) **RNF-003:** el acceso desde la red interna requiere de usuario y contraseña de dominio y la comunicación entre el cliente y el servidor se realizarán a través del protrocolo “http”. El acceso desde internet requiere además el uso de un certificado digital y securización en la comunicaciones “https”.

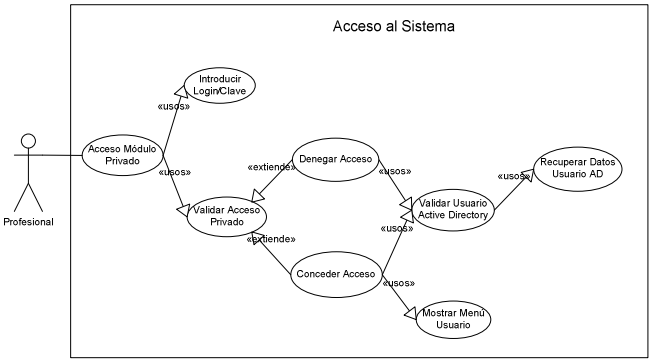
d) RNF-004: el sistema se desplegará en un entorno Web que permita el acceso a través de un navegador web. La arquitectura de desarrollo será J2EE de acuerdo a los estándares establecidos por Sun. En particular se utilizará el framework *Struts* para el desarrollo de la aplicación y para llevar a cabo la implementación del patrón de arquitectura MVC en el que se basará el diseño de la aplicación.

e) **RNF-005:** Se utilizará el servidor de aplicaciones WebSphere de IBM y como SGBD, Informix.

5.3. Especificaciones de Casos de Uso

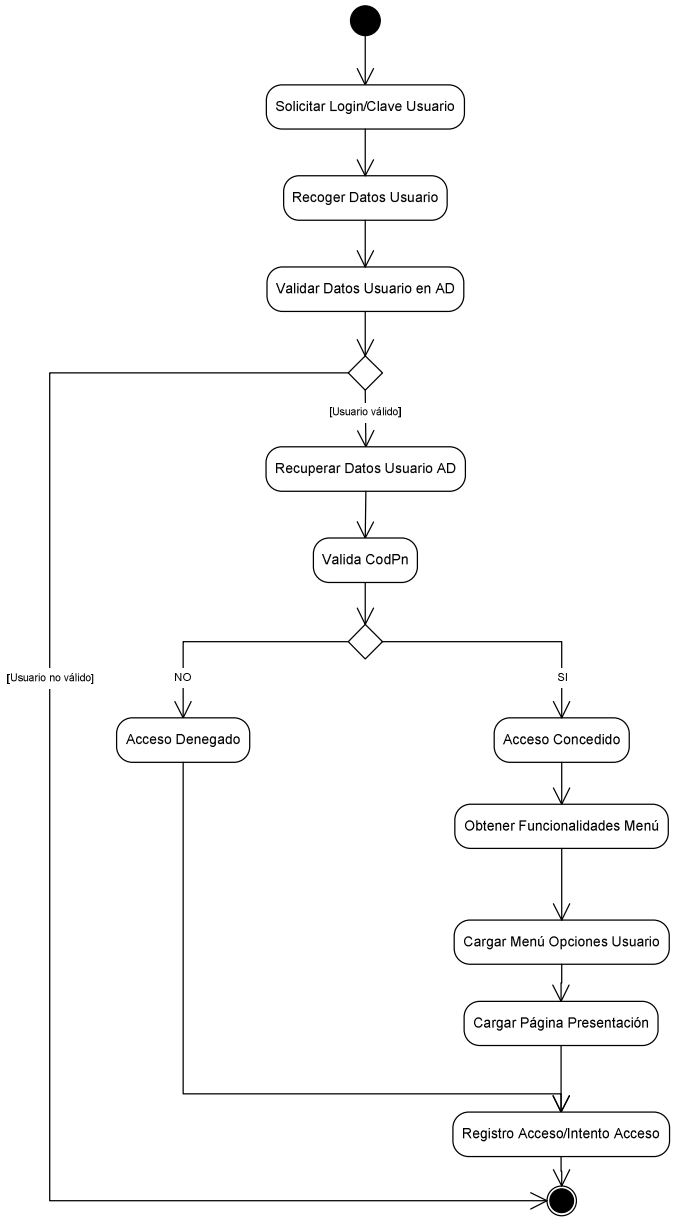
5.3.1. Acceso al Portal

El acceso al sistema requerirá al profesional que se autentique, proporcionando su *Login* y *Clave* de cuenta de usuario en el dominio. El sistema realizará el proceso de validación del usuario en el *Directorio Activo*. El resultado de la validación de los datos proporcionados por el profesional determinará si se concede o deniega el acceso del profesional al sistema.



En la siguiente imagen se representa el diagrama de actividad, para el caso de uso de acceso al sistema de los profesionales.

Acceso al Portal - Diagrama de Actividad



Actores:

- Profesional

Utiliza

- Introducir Login/Clave
- Validar Acceso al Portal

Pre-condiciones

- La aplicación está disponible
- El usuario solicita el acceso al portal

Post-condiciones

- Acceso al sistema concedido o denegado

Flujo de eventos
- El usuario solicita acceso al portal
- La aplicación solicita al usuario la introducción de su *Login* y *Clave* de cuenta de usuario
- La aplicación valida los datos del usuario (Login/Clave) en el Active Directory
- La aplicación recupera el CodPn de la cuenta del usuario e intenta localizar al usuario en el sistema.
- Acceso concedido:
    - Validación en el AD correcta
    - El profesional dispone de NIF en el objeto usuario del AD
    - El profesional es localizado en el sistema a partir del CodPn
- Acceso denegado:
    - Validación en el AD incorrecta: *Login* o *Clave* incorrectos
    - El objeto usuario (*Login*) no dispone de CodPn
    - No es posible localizar al profesional en el sistema a partir del CodPn
- Si el acceso es concedido se recuperan las opciones de menú del profesional y se muestra la página de presentación del sistema

5.3.2. Cambio de Idioma

El usuario cambia el idioma que está utilizando actualmente para trabajar con la aplicación.

Actores
- Profesional

Utiliza
- Cambiar idioma aplicación

Pre-condiciones
- La aplicación está disponible
- El usuario accede al portal
- Seleccionar un idioma diferente al actual

Post-condiciones
- Se cambia la información al idioma seleccionado

Flujo de eventos
- El usuario selecciona el idioma
- Se carga la página con el nuevo idioma

5.3.3. Ayuda en Línea

Acceso a la ayuda de la sección del portal en la que se encuentra actualmente el usuario.Todas las funcionalidades de la aplicación dispondrán de ayuda on-line, en la que se detallarán las características de la funcionalidad en la que se encuentra actualmente el usuario.

Actores
- Profesional

Utiliza
- Localizar ayuda en línea para la funcionalidad actual

Pre-condiciones
- La aplicación está disponible
- El usuario accede al portal
- El usuario selecciona la opción de ayuda en línea para la funcionalidad actual

Post-condiciones
- Se muestra la ayuda disponible para la funcionalidad actual

Flujo de eventos
- El usuario selecciona la opción de ayuda en línea
- Abrir en una nueva pantalla la ayuda disponible para la sección actual

5.3.4. Aviso Legal

El usuario accede a la opción de aviso legal del portal.

Actores
- Profesional

Utiliza
- Obtener idioma actual
- Cargar texto aviso legal del portal

Pre-condiciones
- La aplicación está disponible
- El usuario accede al portal
- El usuario selecciona la opción de aviso legal

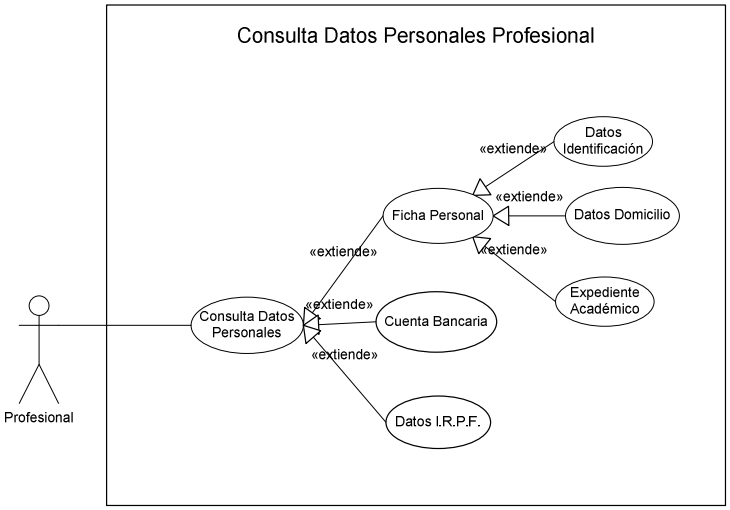
Post-condiciones
- Se carga el texto con el aviso legal

Flujo de eventos
- El usuario selecciona la opción de aviso legal
- La aplicación carga el texto del aviso legal en el idioma actual

5.3.5. Consulta de Datos Personales

Caso de uso para la consulta de los datos personales completos del profesional. Desde esta funcionalidad, los profesionales podrán acceder a toda su información personal gestionada desde el sistema de personal y nóminal. El usuario podrá consultar:

- Ficha Personal:
  - Datos de Identificación
  - Datos Domicilio Social
  - Datos Expediente Académico
- Datos Cuenta Bancaria
- Datos I.R.P.F.



Actores
- Profesional

Puntos de extensión
- Datos Identificación
- Datos Domicilio
- Datos Expediente Académico
- Datos Cuenta Bancaria
- Datos I.R.P.F.

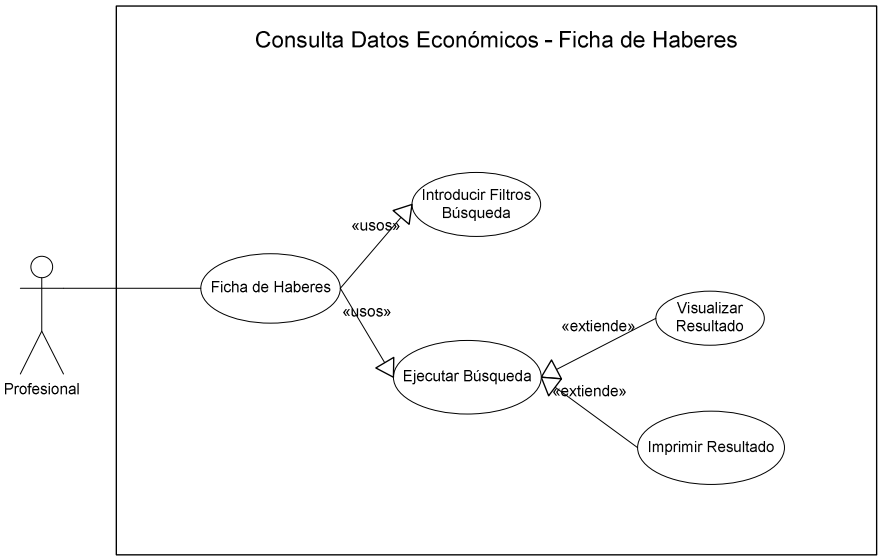
Pre-condiciones
- La aplicación está disponible
- El usuario ha accedido al portal
- El usuario accede a la funcionalidad de consulta de datos personales – datos económicos

Post-condiciones
- El usuario consulta la información personal del profesional

Flujo de eventos
- El usuario selecciona la opción de datos personales y navega entre las distintas secciones:
    - Datos Identificación
    - Datos Domicilio
    - Datos Expediente Académico
    - Datos Cuenta Bancaria
    - Datos I.R.P.F.
- La aplicación carga la información personal del usuario de la sección seleccionada

5.3.6. Consulta resumen datos económicos

El usuario accede a la opción de consulta de su información económica general. El profesional podrá establecer como filtro de búsqueda de la ficha de haberes un ejercicio.



Actores
- Profesional

Utiliza
- Establecer filtros de búsqueda
- Ejecutar búsqueda
    - Mostrar resumen económico
    - Imprimir resumen económico

Pre-condiciones
- La aplicación está disponible
- El usuario accede al portal
- El usuario accede a la funcionalidad de consulta de resumen económico

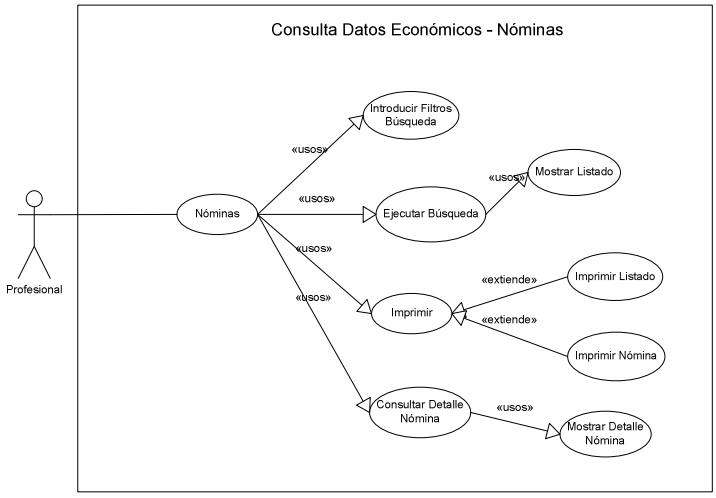
Post-condiciones
- El profesional consulta o imprime el resumen

Flujo de eventos
- El usuario selecciona la opción de resumen económico
- El usuario introduce los filtros de búsqueda y lanza la búsqueda
- La aplicación ejecuta la búsqueda y muestra el resultado en pantalla
- El usuario imprime el resumen económico

Condiciones de fallo
- No existe información económica para los filtros de búsqueda definidos por el usuario

5.3.7. Consulta de Datos Económicos – Nóminas

El usuario accede a la opción de consulta del detalle de sus nóminas y podrá imprimir una nómina en particular.El profesional podrá establecer como filtro de búsqueda de las nóminas un intervalo de fechas.



Actores
- Profesional

Utiliza
- Establecer filtros de búsqueda
- Ejecutar búsqueda
- Imprimir listado
- Imprimir nómina
- Consultar detalle nómina

Pre-condiciones
- La aplicación está disponible
- El usuario accede al portal
- El usuario accede a la funcionalidad de consulta de nóminas

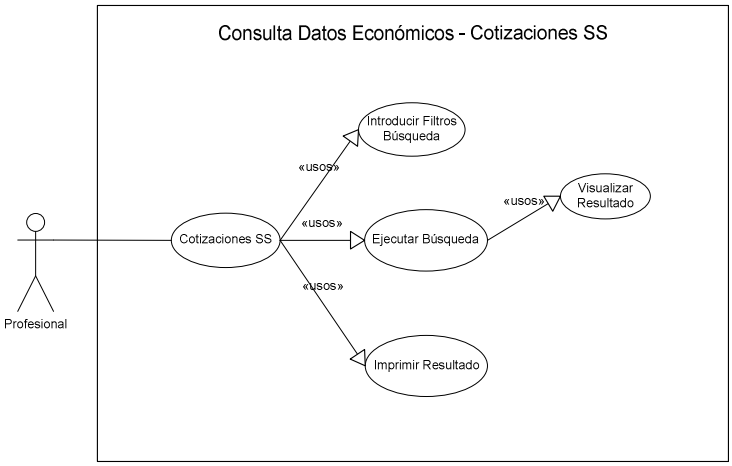
Post-condiciones
- El profesional consulta o imprime alguna de sus nóminas

Flujo de eventos
- El usuario selecciona la opción de consulta de nóminas
- El usuario introduce los filtros de búsqueda
- El usuario lanza la búsqueda de nóminas
- La aplicación ejecuta la búsqueda y muestra el listado de nóminas que cumplan los filtros de búsqueda en pantalla
- El usuario imprime el listado de nóminas
- El usuario selecciona una nómina para consultar el detalle
- La aplicación carga en pantalla la información de la nómina seleccionada
- El usuario imprime el detalle de la nómina

Condiciones de fallo
- No existen nóminas para los filtros de búsqueda definidos por el usuario

5.3.8. Consulta de Datos Económicos – Cotizaciones Seguridad Social

Descripción:
El usuario accede a la opción de consulta de su información económica, en este caso consulta de '*Cotizaciones*' aportadas por la empresa a la Seguridad Social.



Actores
- Profesional

Utiliza
- Establecer filtros de búsqueda
- Ejecutar búsqueda
- Imprimir detalle de cotizaciones

Pre-condiciones
- La aplicación está disponible
- El usuario accede al portal
- El usuario accede a la funcionalidad de consulta de cotizaciones

Post-condiciones
- El profesional consulta o imprime el detalle de cotizaciones

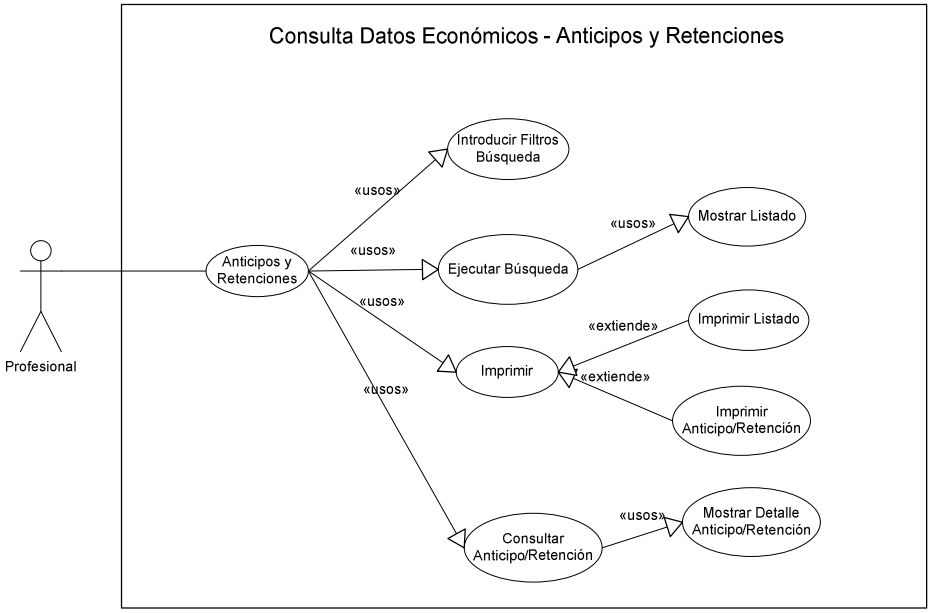
Flujo de eventos
- El usuario selecciona la opción de consulta de cotizaciones
- El usuario introduce los filtros de búsqueda
- El usuario lanza la búsqueda
- La aplicación ejecuta la búsqueda y muestra el resultado en pantalla
- El usuario imprime el detalle de cotizaciones

Condiciones de fallo
- No existe información económica para los filtros de búsqueda definidos por el usuario

5.3.9. Consulta de Datos Económicos – Anticipos y Retenciones

El profesional accede a la opción de consulta de su información económica, en este caso consulta de los '*Anticipos y Retenciones*'. Esta información se obtendrá de la base de datos de *Personal y*

Nóminas.El profesional podrá establecer filtros de búsqueda sobre los anticipos y retenciones por tipo, ejercicio, intervalo de fechas y/o intervalo de importes.



Actores
- Profesional

Utiliza
- Establecer filtros de búsqueda
- Ejecutar búsqueda
- Imprimir listado
- Consultar detalle anticipo/retención
- Imprimir detalle anticipo/retención

Pre-condiciones
- La aplicación está disponible
- El usuario ha accedido al portal
- El usuario accede a la funcionalidad de consulta de anticipos y retenciones

Post-condiciones
- El profesional consulta o imprime el listado de anticipos y retenciones o accede al detalle de un anticipo/retención y lo imprime

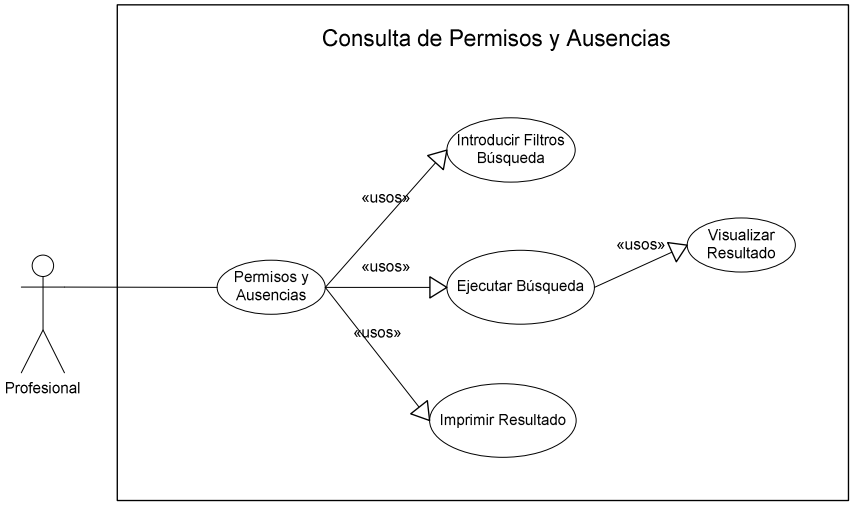
Flujo de eventos
- El usuario selecciona la opción de consulta de anticipos y retenciones
- La aplicación obtiene el histórico de anticipos y retenciones del profesional
- El usuario introduce los filtros de búsqueda
- El usuario lanza la búsqueda
- La aplicación ejecuta la búsqueda y muestra el listado de anticipos y retenciones en pantalla para los filtros de búsqueda establecidos
- El usuario imprime el listado de anticipos y retenciones resultado de la búsqueda

- El usuario selecciona un detalle del listado
- La aplicación carga en pantalla la información del anticipo/retención seleccionado
- El usuario imprime el detalle seleccionado

Condiciones de fallo
- No existen anticpos/retenciones para los filtros de búsqueda definidos por el usuario

5.3.10. Consulta de Permisos y Ausencias

El usuario accede a la opción de consulta del histórico de permisos y ausencias laborales del profesional. El profesional podrá establecer filtros de búsqueda de permisos y ausencias por motivo, intervalo de fechas y/o centro de gestión.



Actores
- Profesional

Utiliza
- Establecer filtros búsqueda
- Ejecutar la búsqueda
- Imprimir listado

Pre-condiciones
- La aplicación está disponible
- El usuario accede al portal
- El usuario accede a la funcionalidad de consulta de permisos y ausencias

Post-condiciones
- El profesional consulta o imprime el listado de permisos y ausencias

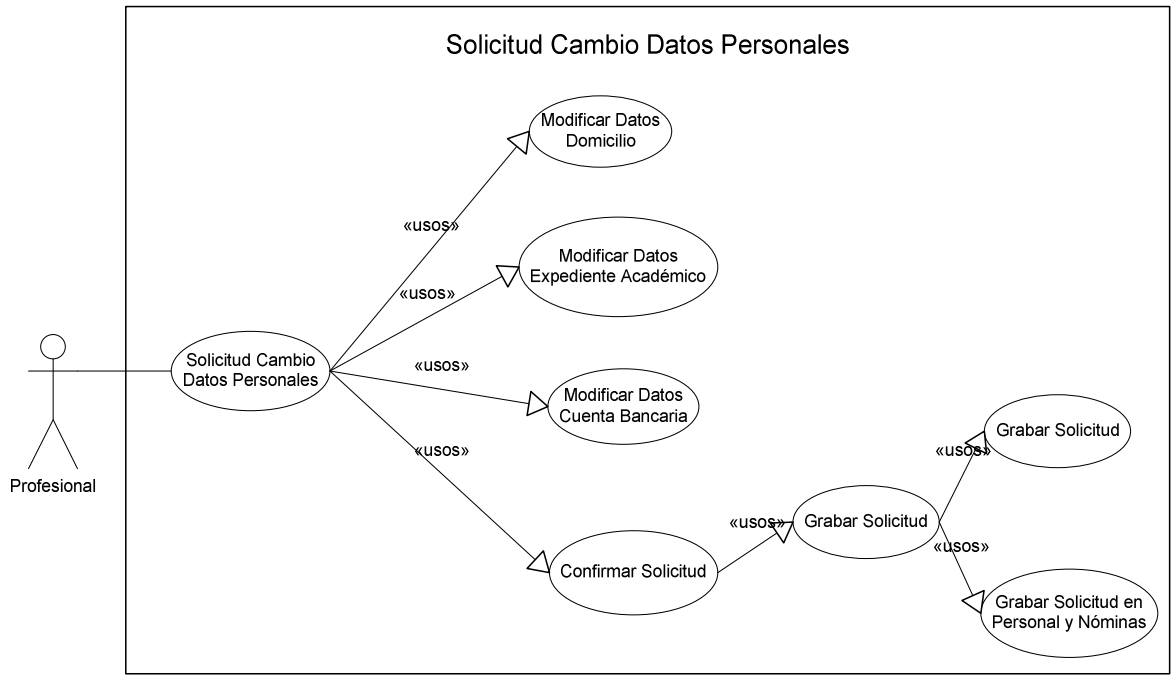
Flujo de eventos
- El usuario selecciona la opción de consulta de permisos y ausencias
- La aplicación muestra el histórico de permisos y ausencias laborales del profesional
- El usuario establece los filtros de búsqueda que desea aplicar
- El usuario lanza la búsqueda

- La aplicación ejecuta la búsqueda y muestra el resultado en pantalla
- El usuario imprime el listado de permisos y ausencias

Condiciones de fallo
- No existe información sobre permisos y ausencias del profesional

5.3.11. Solicitud Cambio Datos Personales

El profesional realiza una solicitud de modificación sobre sus datos personales. Desde esta funcionalidad los profesionales pueden modificar sus datos personales y enviar una solicitud de cambio al sistema. Las solicitudes de cambio serán revisadas y consolidadas por personal específico del área de recursos humanos. Las solicitudes serán almacenadas en la BD el sistema grabadas en la base de datos de *Personal y Nóminas.*



Actores
- Profesional

Puntos de extensión
- Modificar Datos Domicilio
- Modificar Datos Expediente Académico
- Modificar Datos Cuenta Bancaria
- Confirmar Solicitud de Cambio

Pre-condiciones
- La aplicación está disponible
- El usuario ha accedido al portal
- El usuario accede a la funcionalidad de cambio de datos personales

Post-condiciones

- El usuario realiza una solicitud de cambio en sus datos personales

Flujo de eventos

- El usuario modifica alguno de sus datos personales:
    - Datos Domicilio
    - Datos Expediente Académico
    - Datos Cuenta Bancaria
- El usuario confirmar los cambios realizados
- La aplicación graba la solicitud de cambio realizada

Condiciones de fallo

- No se realiza la confirmación de las modificaciones realizadas

5.3.12. Solicitud Cambio Datos Personales

El profesional realiza una solicitud de modificación sobre sus datos personales. Desde esta funcionalidad los profesionales pueden modificar sus datos personales y enviar una solicitud de cambio al sistema. Las solicitudes de cambio serán revisadas y consolidadas por personal específico del área de recursos humanos. Las solicitudes serán almacenadas en la BD el sistema grabadas en la base de datos de *Personal y Nóminas.*

5.4. Diagrama de contexto

En este apartado se representa el diagrama de contexto del sistema a través de los diagramas de casos de uso. Una vez reflejados los casos de uso y los actores, se detallarán los diferentes sub-sistemas.

5.4.1. Diagrama de Casos de Uso General del sistema propuesto.

En este apartado se detallan los casos de uso, funcionalidades, de carácter general para todos los profesionales, empezando por la funcionalidad de validación de acceso al portal.

Los actores que intervienen en el diagrama anterior y las funcionalidades que pueden realizar se describen a continuación.

Profesional Conjunto de profesionales usuarios de la aplicación que tienen vinculación contractual con la organización. El usuario con acceso al sistema podrán realizar las siguientes operaciones:

- *Validar Acceso*: los usuarios accederán al módulo privado del sistema proporcionando su Login/Clave de la cuenta de usuario en el dominio y la aplicación realizará la validación en el LDAP, recuperando de la cuenta de usuario los campos de CodPn; si la cuenta de usuario posee estos datos se intentará localizar al profesional en el sistema de Profesional y Personal y Nómina, si la búsqueda es fructífera el profesional podrá acceder al sistema, en caso contrario se le denegará el acceso indicándole el problema ocurrido
- *Consultar Datos Personales - Ficha Personal*: consulta de la ficha de información personal del profesional que mantiene en sus sistemas de información en la BD de Profesionales (datos de identificación, domicilio, expediente académico, datos bancarios y los datos de IRPF)
- *Consultar Información Económica (Resumen Económico)*: obtener la lista de datos económicos para un ejercicio y que contiene el desglose mensual de las cantidades percibidas por el profesional; el profesional puede imprimir el informe con la ficha de haberes
- *Consultar Información Económica (Cotizaciones SS)*: obtener el listado con las cotizaciones a la Seguridad Social de las nóminas del profesional; el profesional puede imprimir el informe de cotizaciones a la SS

- *Consultar Información Económica (Nóminas)*: obtener el listado con las nóminas del profesional; el profesional puede imprimir el listado de nóminas o acceder a una nómina en particular e imprimir el detalle
- *Consultar Información Económica (Anticipos y Retenciones)*: obtener el listado de anticipos y retenciones del profesional; el profesional puede imprimir el informe con el listado de anticipos y retenciones
- *Consultar Permisos y Ausencias*: obtener el listado con los permisos y ausencias del profesional; el profesional puede imprimir el listado de permisos/ausencias o acceder a un permiso/ausencia en particular e imprimir el detalle
- *Solicitar Cambio Datos Personales*: el profesional solicita el cambio en alguno de sus datos personales. La solicitud de cambio será almacenada en el sistema B2E, grabada en el sistema de *Personal y Nóminas* y enviada por correo electrónico a una cuenta de correo de RRHH para recibir las solicitudes de cambio realizadas por los profesionales
- *Consultar Ayuda*: guardar en disco o consultar en línea el manual de usuario de la aplicación
- *Cambiar Idioma*: cambiar el idioma utilizado actualmente en cualquiera de las funcionalidades del sistema

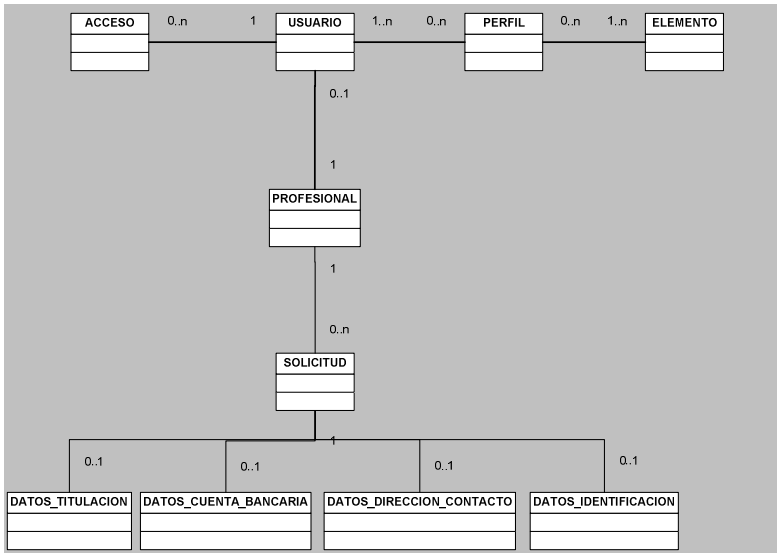
RRHH: Usuarios pertencientes a la unidad de Recursos Humanos de la organización, encarga:dos de mantener los datos de los diferentes sistemas de información, en concreto los de la aplicación del Gestión de personal y nóminas.

Sistemas: Personal encargado del mantenimiento del Directorio Activo y cuentas de correo de la oganización.

5.5. Diagrama de clases

A continuación se representa el Diagrama de Clases del sistema B2e.

Subsistema de Acceso
Diagrama de clases del subsistema de *Acceso*:



Subsistema de Consulta

Diagrama de clases del subsistema de *Consulta datos Personales*:

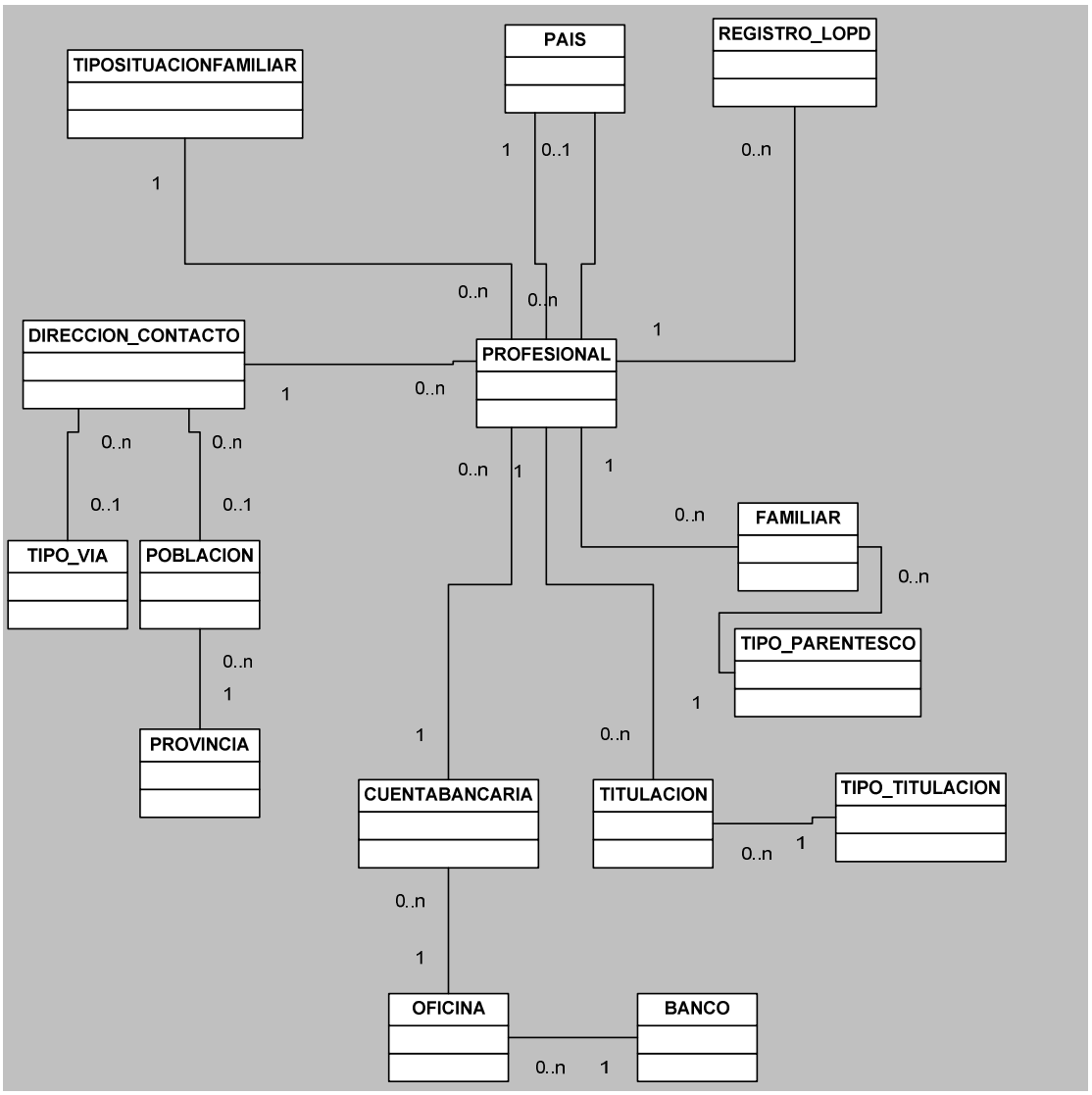
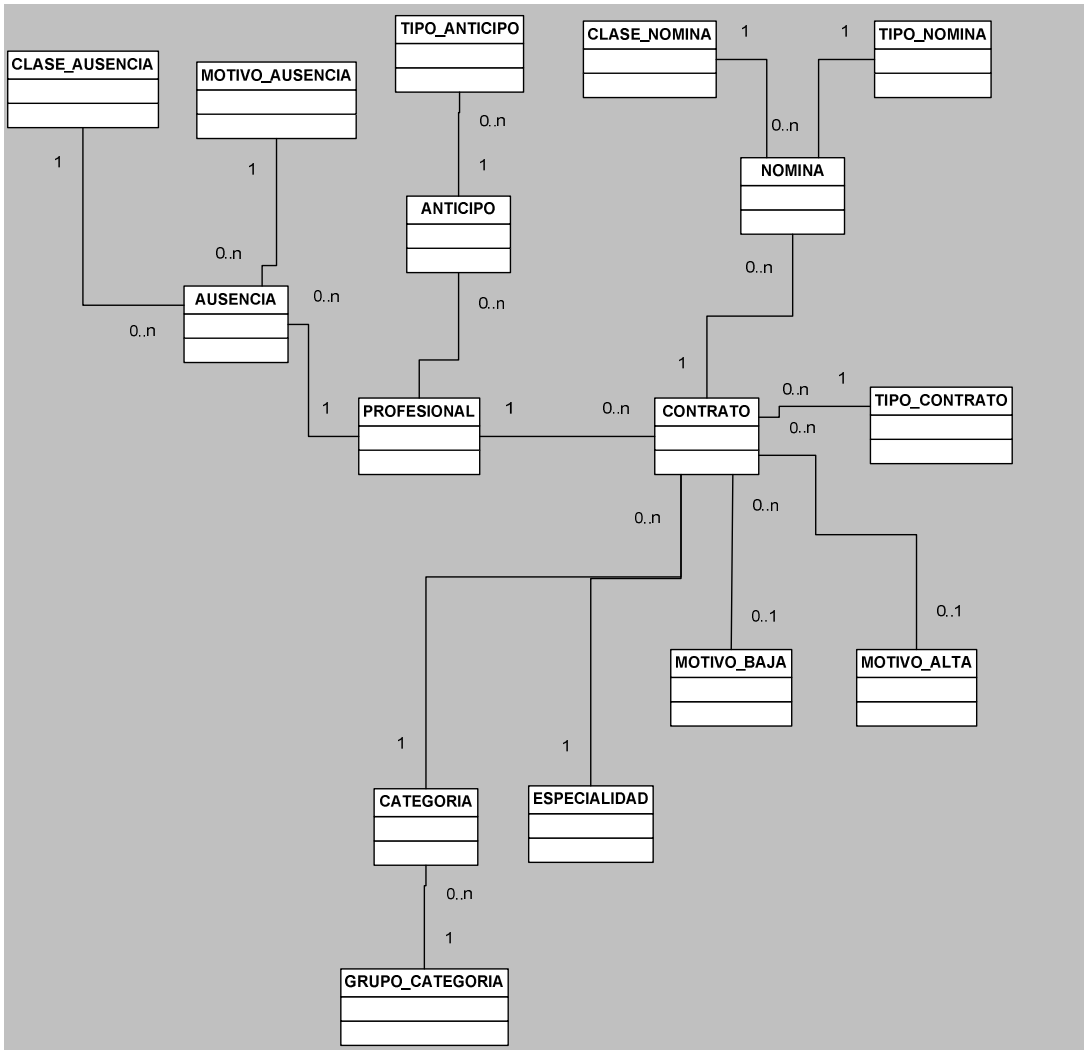
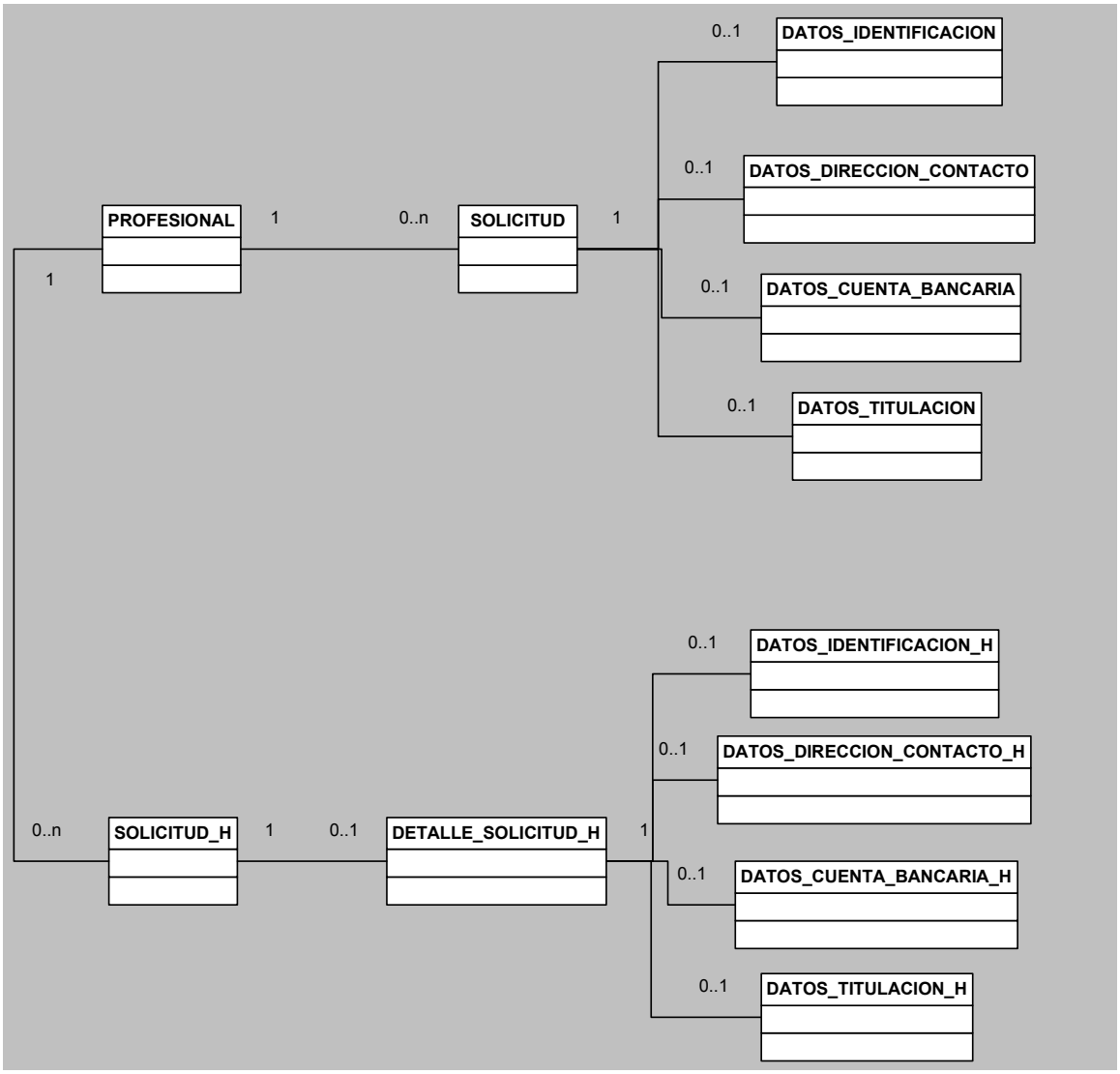


Diagrama de clases del subsistema de *Consulta datos Económicos*:


CLASE_AUSENCIA
MOTIVO_AUSENCIA
TIPO_ANTICIPO
CLASE_NOMINA
TIPO_NOMINA
NOMINA
ANTICIPO
AUSENCIA
PROFESIONAL
CONTRATO
TIPO_CONTRATO
MOTIVO_BAJA
MOTIVO_ALTA
CATEGORIA
ESPECIALIDAD
GRUPO_CATEGORIA
1
0..n
0..1

Subsistema de Solicitudes

Diagrama de clases del subsistema de *Solicitudes*:



En los diagramas anteriores se realiza la representación jerárquica del diagrama de clases a partir de las entidades de negocio identificadas en el análisis funcional del sistema B2e, a partir de los casos de uso y de los subsistemas de análisis. Se han representado 3 diagramas, 1 por cada subsistema identificado: acceso, solicitudes y consulta (datos *Personales* y datos *Económicos*).

6. DISEÑO

6.1. Introducción.

A continuación se justificarán las decisiones de arquitectura técnica, los patrones y se detallarán en fase de diseño los diferentes subsistemas definidos en la fase de análisis.

6.2. Justificaciones tecnológicas

La aplicación a desarrollar será una aplicación Web a la que accederán usuarios mediante un Navegador Web, bien desde la red interna o desde *Internet*. Para el soporte multidioma de la aplicación se utilizarán los archivos de propiedades generados de acuerdo al framework *Struts*.

Entorno de desarrollo planteado:

- Arquitectura J2EE
- *Eclipse NetBeans*
- *SDK 1.4 de Sun*
- Código *Java* para el desarrollo de los componentes
- Framework *Jakarta Struts 1.2.4* (implementación MVC)
- *Jakarta Standard Tag Libs (JSTL) 1.0.4*
- *JDBC – POJOs*
- *IBM Informix JDBC Driver 2.21.JC6* como driver de acceso al SGBD de Informix
- Macromedia DremweaverMX 2004
- Microsoft Internet Explorer 5.5, Mozilla Firefox 1.0, Netscape 7.0
- *Apache Tomcat 4.1* para probar el desarrollo

6.3. J2EE (Java 2 Enterprise Edition)

J2EE 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. J2EE 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.

Así pues, J2EE se enmarca dentro de un estilo arquitectónico heterogéneo, y aglutina distintas características correspondientes a estilos arquitectónicos en capas o niveles, cliente-servidor, orientada a objetos distribuidos e, incluso, orientada a servicios . De esta definición, podemos extraer los conceptos básicos y los puntos clave que hay detrás de la plataforma J2EE:

- J2EE 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 J2EE es una especificación de especificaciones.

- J2EE 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 J2EE define un mínimo de tres capas: la

capa cliente, la capa intermediay la capa de sistemas de información de la empresa (EIS - Enterprise Information Systems).

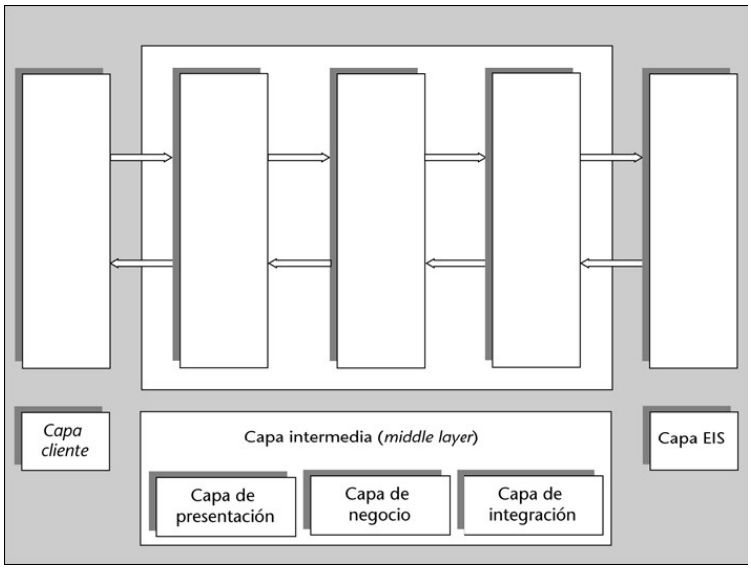
- J2EE basa las aplicaciones empresariales en el concepto de componentes modulares y estandarizados. Este concepto está muy vinculado al concepto contenedor. Los contenedores son entornos estándar de ejecución que proporcionan un conjunto de servicios a los componentes que ahorran mucho trabajo a la hora de desarrollar los componentes.

J2EE 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 J2EE; 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.

6.3.1. Arquitectura multicapa con n-niveles

El modelo de aplicaciones que define J2EE 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.

J2EE extiende la arquitectura típica en tres capas o niveles para definir una arquitectura en n-capas. Esta extensión se hace de manera natural, considerando que la capa intermedia se puede subdividir en el nivel lógico en diferentes capas, cada una con unas responsabilidades diferenciadas.

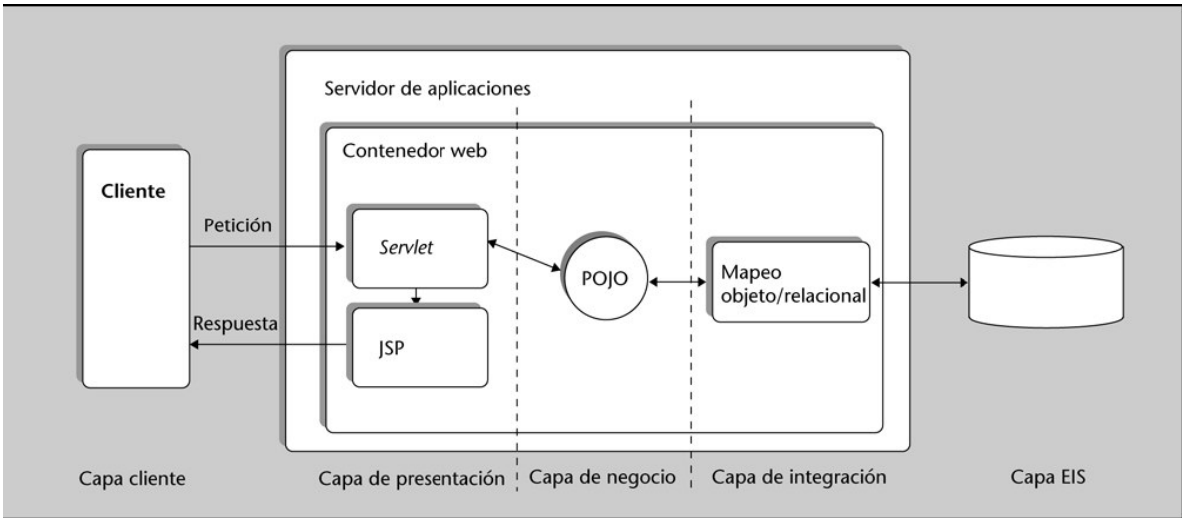


- Capa cliente. Contiene los componentes que se ejecutan en las máquinas cliente. Por ejemplo, páginas HTML que se ejecutan en los navegadores de los usuarios.
- Capa intermedia (middle layer). Normalmente, contiene el procesamiento de la lógica de negocio, situada entre la capa cliente y la capa EIS. Los componentes de esta capa se ejecutan en los diferentes contenedores que forman el servidor de aplicaciones. En la arquitectura J2EE, podemos dividir la capa intermedia en tres subcapas formadas por componentes distintos que se ejecutan en contenedores diferentes y que tienen funcionalidades distintas:

  - Capa de presentación. Controla la lógica de interacción entre el usuario y la aplicación, así como la forma con la que se presenta la información. Contiene los componentes de presentación que se ejecutan en el contenedor web del servidor de aplicaciones.
  - Capa de negocio. Contiene el código y las reglas de negocio para nuestra aplicación.
  - Capa de integración. Capa en la que se hacen las tareas de integración de la aplicación con los diferentes sistemas con los que tiene que interactuar (bases de datos, sistemas legacy, etc.).
- Capa EIS. En esta capa, encontramos el resto de los componentes del sistema de información que hay en la empresa. Por ejemplo, contiene los componentes de la capa de datos que residen en un servidor de datos (comopodrían ser las tablas de una base de datos relacional).

6.3.2. Arquitecturas más habituales – EJB's versus POJO.

La gran ventaja de los POJO en sistemas que no requieren componentes distribuidos es que son objetos muy ligeros que no añaden ningún tipo de carga adicional (gestión de transacciones, seguridad, ciclo de vida, etc.). Esto los hace muy fáciles de desarrollar, además de proporcionar muy buen rendimiento. Su principal inconveniente es que no tenemos ninguna de las características que tienen los EJB y si las necesitamos, las tendremos que implementar (o usar algún framework que las proporcione).



Ésta es la arquitectura más simple. Los componentes web y los componentes denegocio se ejecutan en la misma máquina virtual y los componentes de negocio no son EJB, sólo se necesita un contenedor web. Sin embargo, hay una clara separación entre los componentes de negocio y los componentes de presentación

En esta arquitectura tenemos:

- Capa de presentación con servlets y JSP siguiendo un patrón de diseño MVC.
- Acceso local de la capa de presentación a la capa de negocio.
- Capa de negocio implementada con POJO (clases java)
- Capa de integración normalmente desarrollada con algún framework de mapeo objeto/relacional o bien directamente con clases Java siguiendo el patrón DAO (Data Access Object).

6.4. MVC y J2EE

El principal objetivo de la arquitectura MVC es aislar tanto los datos de la aplicación como el estado (modelo) de la misma, del mecanismo utilizado para representar (vista) dicho estado, así como para modularizar esta vista y modelar la transición entre estados del modelo (controlador). Las aplicaciones MVC se dividen en tres grandes áreas funcionales:

- Vista: la presentación de los datos
- Controlador: el que atenderá las peticiones y componentes para toma de decisiones de la aplicación
- Modelo: la lógica del negocio o servicio y los datos asociados con la aplicación

El propósito del MVC es aislar los cambios. Es una arquitectura preparada para los cambios, que desacopla datos y lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los citados componentes.

En nuestro casl la aplicación será implementada con J2EE utilizando JSP para las vistas, servlets como controladores y JDBC para el modelo.

6.5. Framework Struts.

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE. Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Struts se basa en el patrón del Modelo-Vista-Controlador (MVC) el cual se utiliza ampliamente y es considerado de gran solidez. De acuerdo con este patrón, el procesamiento se separa en tres secciones diferenciadas, llamadas el modelo, las vistas y el controlador. Cuando se programan aplicaciones Web con el patrón MVC, siempre surge la duda de usar un solo controlador o usar varios controladores, pues si consideramos mejor usar un solo controlador para tener toda nuestra lógica en un mismo lugar, nos encontramos con un grave problema, ya que nuestro controlador se convierte en lo que se conoce como "fat controller", es decir un controlador de peticiones, Struts surge como la solución a este problema ya que implementa un solo controlador (ActionServlet) que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml).

Entre las características de Struts se pueden mencionar:

- Configuración del control centralizada.
- Interrelaciones entre acciones y página u otras acciones se especifican por tablas XML en lugar de codificarlas en los programas o páginas.
- Componentes de aplicación, que son el mecanismo para compartir información bidireccionalmente entre el usuario de la aplicación y las acciones del modelo.
- Librerías de entidades para facilitar la mayoría de las operaciones que generalmente realizan las páginas JSP.
- Struts contiene herramientas para validación de campos de plantillas bajo varios esquemas que van desde validaciones locales en la página (en javaScript) hasta las validaciones de fondo hechas a nivel de las acciones.

Struts permite que el desarrollador se concentre en el diseño de aplicaciones complejas como una serie simple de componentes del Modelo y de la vista intercomunicados por un control centralizado. Diseñando de esta manera puede obtenerse una aplicación más consistente y más fácil de mantener.

6.6. Definición de Patrones Utilizados

A continuación se describen los patrones identificados en el diseño de la solución para el desarrollo:

Front Controller

- *Problema:*

  El sistema requiere un punto de acceso centralizado para capturar las peticiones de la capa de presentación, que soporte la integración de servicios del sistema, manejo de vistas, navegación. Cuando un usuario accede a la vista directamente sin pasar por ese punto centralizado pueden dar dos problemas:
  - *Cada vista requiere sus propios servicios del sistema, dupicación de código*
  - *La navegación a través de las vistas se deja a las vistas. Esto da lugar a una mezcla entre el contenido de la vista y la navegación*

  Además el control distribuido es más difícil de mantener, dado que los cambios afectarán a múltiples lugares.
- *Solución*

  Usar un controlador como punto inicial de contacto para manejar la request. El controlador maneja las operaciones de la request, incluyendo la invocación de servicios de seguridad como autenticación y autorización, delegando la lógica de negocio, escogiendo la vista apropiada, controlando los errores, y manejando la selección de estrategias de creación de contenido.
- *Aplicabilidad*
  - *El tratamiento común de servicios de sistema completos por petición. Por ejemplo, el servicio de seguridad completa comprobaciones de autorización y la autenticación*
  - *La lógica es manejada mejor en un punto central en vez de replicarla dentro de numerosas vistas*
  - *Los puntos de decisión existen en función de la recuperación y la manipulación de datos*
  - *Múltiples vistas son usadas para responder a peticiones de negocio similares*
  - *Un punto centralizado de contacto para manejar una petición puede ser útil, por ejemplo, controlar y registrar el progreso de un usuario por la aplicación*
  - *Los servicios de sistema y la lógica de manejo de la vista son relativamente sofisticados*

View Helper

- *Problema*

  A menudo suceden cambios en la capa de presentación que son difíciles de mantener y desarrollar cuando la lógica de acceso a los datos y el formateo de la presentación están mezclados. Esto hace que el sistema sea menos flexible, menos reutilizable y generalmente menos resistente a cambios.
  La mezcla ente lógica de negocio y procesamiento de vistas reduce la modularidad y además proporciona una pobre separación entre los roles dedicados a la creación de Web y de desarrollo de software.
- *Solución*

Una vista contiene código formateado, delegamos las responsabilidades de ese proceso a las clases de apoyo, implementadas como java Beans o etiquetas personalizadas. Las clases de apoyo (Helpers) sirven como adaptadores de datos y almacenarán el modelo de datos intermedio de la vista.

- *Aplicabilidad*

  Los requerimientos de asimilación de datos no son triviales:
  - *Incrustar la lógica de negocio dentro de la vista promueve la reutilización del tipo copiar-pegar. Esto provoca problemas de mantenimiento porque un trozo de código es reutilizado en diferentes sitios o el mismo simplemente duplicándolo en su nueva localización*
  - *Es deseable promover una separación clara de cada tarea en función de los roles de desarrolladores del software*
  - *Una vista se usa normalmente para responder a una petición particular*

Dispatcher View

- *Problema*

  El problema aquí es una combinación de problemas que son resueltos por los patrones FrontController y ViewHelper, en la capa de presentación. No hay componente centralizado para manejar el control de acceso, la recuperación de contenido, o el manejo de las vistas, y hay código duplicado y repartido a través de varias vistas. Además la lógica de negocio y formateo de presentación se encuentran entremezclados en esas vistas, haciendo que el sistema sea menos flexible, reutilizable y generalmente menos resistente a cambios.
  La mezcla ente lógica de negocio y procesamiento de vistas reduce la modularidad y además proporciona una pobre separación entre los roles dedicados a la creación de Web y de desarrollo de software.
- *Solución*

  Combinar un controler y dispatcher con vistas y clases de apoyo, para manejar las peticiones de los clientes y preparar presentaciones dinámicas como repuesta. Los controladores no delegan la recuperación de contenido a las clases de apoyo. Un dispatcher es responsable del manejo y navegación de la vista y puede se encapsulado en un contoller, una vista o un componente separado
- *Aplicabilidad*
  - *Comprobaciones de autorización y autenticación son completadas por request*
  - *El código scriplet debe ser reducido*
  - *La lógica de negocio debe ser encapsulada en componentes diferentes de la vista*
  - *El flujo de control es relativamente simple y está típicamente basado en valores encapsulados con la petición*
  - *Manejo de lógica de las vistas es limitada en complejidad*

Session Facade

- *Problema*

  En una aplicación de la plata forma Java2, Enterprise Edition (J2EE), surge los siguientes problemas:
  - *Fuerte acoplamiento, que nos lleva a que haya una dependencia directa ente los objetos cliente y de negocio*
  - *Demasiadas invocaciones de métodos entre el servidor y clientes , ocasionando problemas de rendimiento de red*

- *Falta de una estrategia de acceso a los clientes, provocando un mal uso de los objetos de negocio*

La interacción entre clientes y objetos de negocio lleva a un alto acoplamiento entre ambos, Hace que el cliente dependa directamente de la implementación de los objetos de negocio. Esta dependencia directa significa que el cliente debe representar e implementar las complejas interacciones de los objetos de negocio y debe manejar las relaciones entre los objetos participantes, así como conocer la demarcación de la responsabilidad de las transacciones.
El problema aumenta cuando accedemos directamente a los objetos expuestos a los clientes, no hay una estrategia unificada para acceder a los objetos de negocio. Sin una estrategia de acceso uniforme, los objetos de la lógica de negocio son expuestos a los usuarios y se reducirá el uso consistente de éstos.

- *Solución*

Usar un Session Facade *como fachada para encapsular la complejidad de las interacciones entre los objetos de negocio que participan en un flujo de trabajo. El* Session Facade *maneja los objetos de negocio, y proporciona un servicio de acceso uniforme a la capa de acceso a los clientes*

- *Aplicabilidad*
  - *Proveer un interfaz más simple a los clientes ocultando todas las complejas interacciones entre los objetos de negocio*
  - *Reducir el número de objetos de negocio que son conocidos por el cliente a través de una capa de servicio en la red*
  - *Ocultar al cliente las interacciones y dependencias entre los componentes de negocio. Esto proporciona una mejor manejabilidad, centralización de interacciones, mayor flexibilidad, y mayor habilidad para adpatarse a los cambios*
  - *Proveer una capa de servicio para separar la implementación de los objetos de negocio de la abstracción de los servicios de negocio*
  - *Evitar la exposición de la lógica de negocio subyacente a los clientes para mantener el fuerte acoplamiento entre las dos capas al mínimo*

Abstract Factory

- *Problema*

El problema que intenta solucionar este patrón es el de crear diferentes familias de objetos. Uno de los usos más comunes es el de creación de interfaces gráficas de distinto tipo (gtk, qt, etc.), para lo cual se suele combinar con los patrones *Singleton* y *Adapter*

- *Solución*

La definición de interfaces para la familia de productos *genéricos* (ej: ventana, menú, botón...)
Implementación de las interfaces de los productos para cada una de las distintas familias concretas (ej: gtk_ventana, gtk_menú, gtk_botón... y qt_ventana, qt_menú, qt_botón...)
La definición de los métodos de creación de los productos genéricos en la interfaz de la fábrica (ej: construir_ventana, construir_menú, construir_botón...) cuyo tipo de retorno serán las interfaces genéricas.
Implementación de una fábrica para cada una de las familias concretas (ej: fabrica_gtk, fabrica_qt).

- *Aplicabilidad*

El patrón Abstract Factory está aconsejado cuando se prevé la inclusión de nuevas familias de productos, pero puede resultar contraproducente cuando se añaden nuevos productos o cambian los existentes

Business Delegate

- *Problema*

  Los objetos de la capa de presentación interactúan directamente con los servicios de negocio. Esta comunicación directa expone los detalles de implementación de los servicios de negocio API,a la capa de presentación. Como resultado los componentes de la capa de presentación son vulnerables a cambios producidos en la implementación de los servicios de negocio.
  Además, puede haber un impacto perjudicial sobre el funcionamiento de red porque los componentes de la capa de presentación que usan los API de los servicios de negocio hacen demasiadas invocaciones a través de la red. Esto ocurre cuando los componentes de la capa *de presentación usan directamente el API, sin usar mecanismos de caché*
- *Solución*

  Usar un Bussines Delegate para reducir el acoplamiento entre los servicios de negocio y la capa de presentación de los clientes. El Bussines Delegate oculta los detalles de implementación del servicio de negocio, así como detalles de búsqueda y acceso de la arquitectura de los EJB.
- *Aplicabilidad*
  - *La capa de presentación de los clientes necesita acceder a los servicios de negocio*
  - *Diferentes clientes, como los dispositivos, los clientes Web, y los clientes pesados necesitan acceder a los servicios de negocio*
  - *Apis de los servicios de negocio pueden cambiar sus requerimientos*
  - *Es deseable minimizar el acoplamiento entre la capa de presentación de los clientes y los servicios de negocio*
  - *Los clientes pueden necesitar implementar mecanismos de cache para servicios de negocio de información*
  - *Es deseable reducir el tráfico de red entre clientes y servicios de negocio*

Accessor

- *Problema*

  El software de empresa de desarrollo requiere una mezcla rica de programa y la experiencia de negocio. La lógica de aplicación debe reflejar con exactitud procesos de negocio dentro de su dominio así como utilizar el acceso de datos y recursos de sistema de manera eficiente
- *Solución*

  Encapsula detalles de acceso de datos físicos en un componente, exponiendo operaciones sólo lógicas. El código de aplicación mantiene el conocimiento sobre el modelo de datos subyacente, pero está liberado de responsabilidades de acceso de datos
- *Aplicabilidad*
  - Ocultar los detalles de la complejidad de acceso a datos y de la plataforma de la lógica de la aplicación
  - Utilizar funcionalidades adicionales a los proporcionados por el driver de acceso a la base de datos. Normalmente, los drivers de acceso a base de datos no proporcionan métodos para manejar la distribución de los datos o niveles de bloqueo, debido a que la implementación depende de la topología de las aplicaciones y a su arquitectura
  - Es necesario implementar diferentes métodos de acceso a los datos y poder elegir el adecuado en tiempo de ejecución. Diferentes implementaciones permiten

interactuar con diferentes plataformas de base de datos e incluso añadir nuevas tecnologías de base de datos, como consultas XML o bases de datos orientadas a objetos

Data Access Object (DAO)

- *Problema*

  Muchas aplicaciones del mundo real de la plataforma J2EE necesitan usar datos persistentes en algún punto. Para muchas aplicaciones, el almacenaje persistente es puesto en práctica con mecanismos diferentes, y hay diferencias marcadas entre las APIs usadas para acceder a esos diferentes mecanismos de almacenamiento persistente. Otras aplicaciones pueden tener que tener acceso a los datos que residen sobre sistemas separados. Por ejemplo, los datos pueden residir en sistemas de mainframe, LDAP, repositorios, etcétera. Otro ejemplo es donde los servicios proporcionan datos por sistemas externos como negocios entre empresas (B2B) sistemas de integración, el servicio de oficina de tarjeta de crédito, etcétera, etcétera
- *Solución*

  Usar un DAO para abstraer y encapsular todo el acceso a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar los datos
- *Aplicabilidad*

  Desacoplar la lógica de negocio de la lógica de acceso a datos, de manera que se pueda cambiar la fuente de datos fácilmente

Value Object (VO)

- *Problema*

  Las aplicaciones de la plataforma J2EE implementan componentes de negocio del lado del servidor como beans de sesión y de entidad. Algunos métodos expuestos por los componentes de negocio devuelven datos al cliente. Algunas veces, el cliente invoca a los métodos get de un objeto de negocio varias veces para obtener todos los valores de los atributos.

  Según se incrementa la utilización de estos métodos remotos, el rendimiento de la aplicación se puede degradar significativametne. Por lo tanto, utilizar varias llamadas a métodos get que devuelven simples valores de atributos es ineficiente para obtener valores de datos desde un bean enterprise.
- *Solución*

  Utilizar un Transfer Object para encapsular los datos de negocio. Se utiliza una única llamada a un método para envíar y recuperar el Transfer Object. Cuando el cliente solicita los datos de negocio al bean enterprise, éste puede construir el Transfer Object, rellenarlo con sus valores de atributos y pasarlo por valor al cliente
- *Aplicabilidad*

  Cuando es necesario representar un conjunto de atributos procedentes de uno o varios objectos del dominio

Page-by-Page Iterator (Value List Handler)

- *Problema*

  Las aplicaciones de la plataforma J2EE suelen tener un requerimiento de búsqueda y consulta para buscar y listar ciertos datos. En algunos casos las operaciones de búsqueda y consulta podrían devolver resultados que pueden ser bastante grandes. No es práctico devolver toda la hoja de resultados cuando los requerimientos del cliente son moverse por

los resultados, en vez de procesar el conjunto completo. Normalmente, un usuario utiliza los resultados de una consulta para propósictos de sólo lectura, como mostrar una lista de resultados.

- *Solución*

  Utilizar un Page-byPage Iterator para controlar la búsqueda, hacer un caché con los resultados, y proporcionar los resultados al cliente en una hoja de resultados cuyo tamaño y desplazamiento cumpla los requerimientos del cliente
- *Aplicabilidad*
  - Cuando es necesario visualizar la lista de Value Objects por trozos, pudiendo ir hacia delante o atrás
  - La lista completa no cabría en pantalla
  - La lista completa no cabría en memoria

Template method (Plantilla)

- *Problema*

  Las aplicaciones de la plataforma J2EE suelen necesitar de un framework que permita presentar múltiples documentos al usuario (abstracciones Aplicación y Documento). Para una abstracción específica, se especializan los conceptos de Aplicación y Documento. La abstracción aplicación conoce la operación genérica a realizar (por ejemplo, abrir un documento), pero parte de la operación depende del documento concreto.

- *Solución*

  Implementar un método que difiera en otros métodos abstractos las partes del algoritmo que no se conocen o que son específicas. Un método plantilla es un método que define un algorimo en base a una serie de operaciones abstractas que son redefinidas por las subclases para obtener un comportamiento concreto
- *Aplicabilidad*
  - Implementar las partes invariables de un algoritmo una única vez y permitir que las subclases definan el comportamiento que cambia
  - Factorización de comportamiento común de las subclases en la superclase (evitar replicación de código mediante generalización)
  - Para controlar las extensiones de las subclases. El método plantilla utiliza métodos especiales (*hooks*) que son los únicos puntos que pueden ser redefinidos

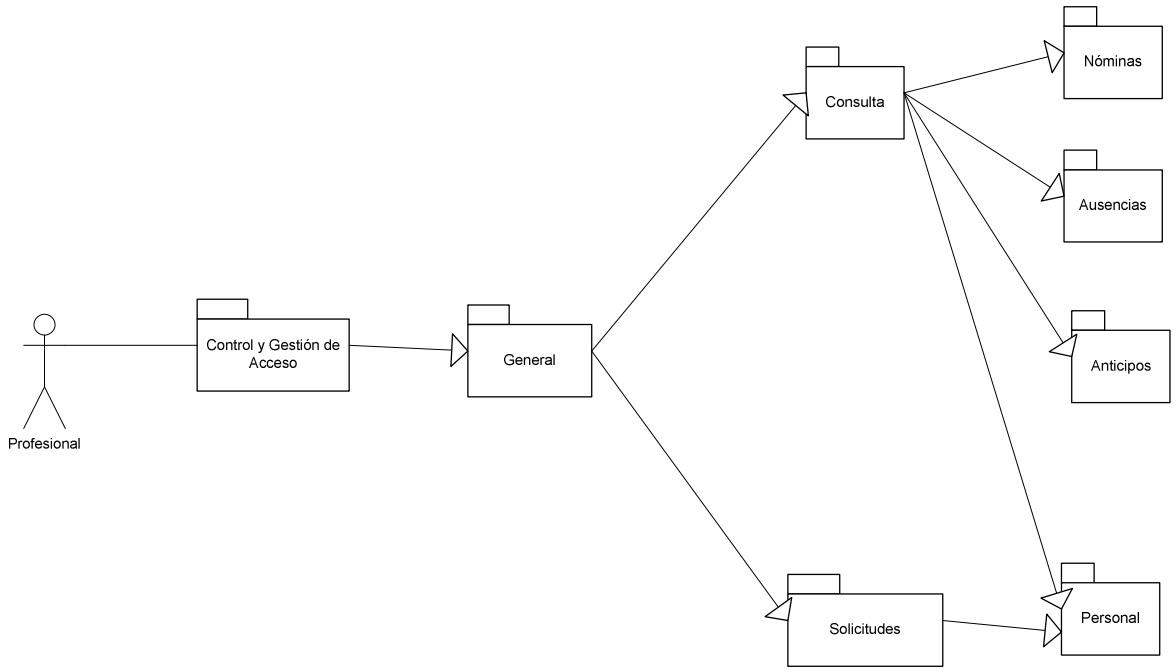
6.7. JDBC-POJOs vs EJBs de entidad

La gran ventaja de la alternativa de POJOs es que se trata de objetos muy ligeros. Éstos no añaden ningún tipo de carga adicional al proceso y por lo tanto, resultan muy fáciles de implementar, presentan un sencillo mantenimiento y tienen un gran rendimiento. La desventaja es que no ofrecen la funcionalidad proporcionada por los EJBs, por lo tanto será necesario utilizar librerías o implementar manualmente los servicios proporcionados por el contendor de EJBs.

El uso de JDBC-POJOs obligará a implementar la lógica de acceso a datos, operaciones CRUD (*Create, Read, Update, Delete*) que en los EJBs de entidad ya vienen implementados por defecto al utilizar la persistencia con CMP. De la misma forma, será necesario implementar por código las transacciones en las operaciones realizadas sobre los datos.

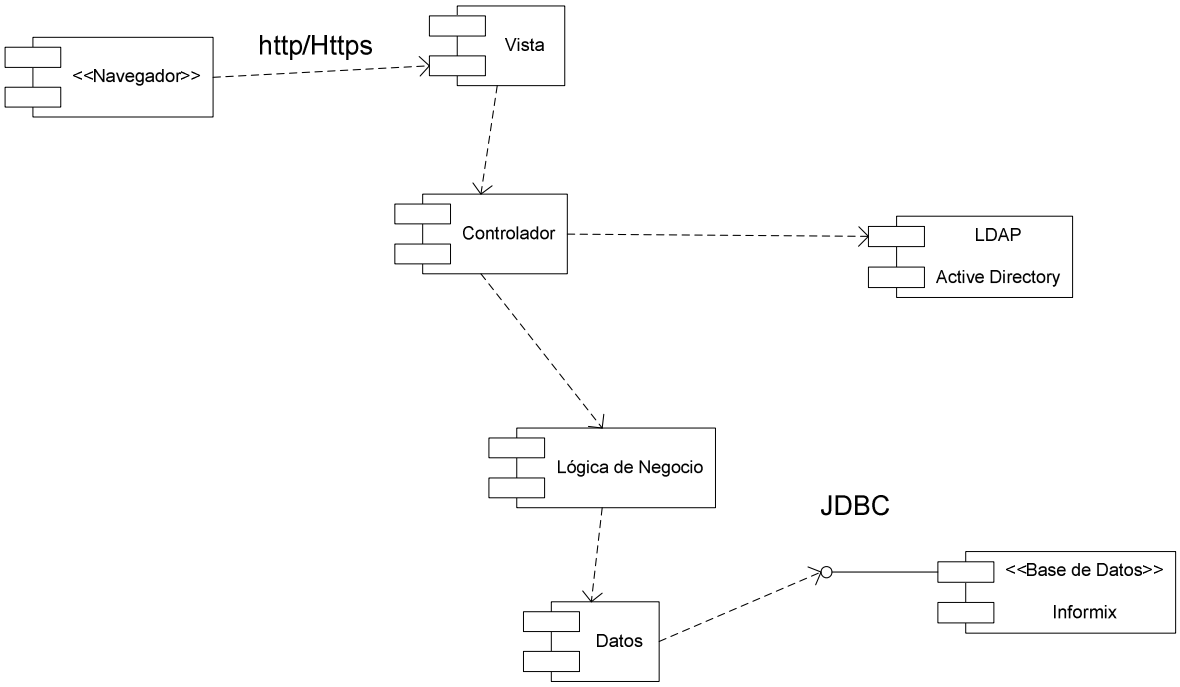
6.8. Descripción de los subsistemas de diseño.

- ***Subsistema de Control y Gestión de Acceso***. En este subsistema se englobarán todas las tareas de control y gestión de acceso de usuarios, seguridad, registro de logs…
- ***Subsistema General***. Contendrá las funcionalidades generales para todos los profesionales y cuya información se obtendrá accediendo al sistema de Personal y nóminas de RRHH
- ***Subsistema de Consulta***. Agrupará las funcionalidades de consulta que puedan realizar los usuarios de la aplicación (consulta de datos personales y/o económicos)
- ***Subsistema de Solicitudes***. Contendrá las funcionalidades de solicitud que puedan realizar los usuarios desde el sistema (solicitud de cambio de datos personales)



El subsistema de Control y Gestión de Acceso es un subsistema de ***soporte*** (resuelve servicios comunes a varios elementos del sistema: autenticación, registro de logs, interfaz gráfico) y los subsistemas General, Consultas y Solicitudes son subsistemas *específicos* pues resuelven funcionalidades propias del sistema B2e.

En el *Diagrama de Componentes* del sistema que se muestra a continuación podemos ver los distintos niveles lógicos en los que se divide la solución:



- **Capa Cliente**. Navegador Web instalado en el equipo cliente
- **Capa Presentación: Vista** y **Controlador**. La capa de presentación se diseñará e implementará utilizando el framework *Struts 1.2.4* que implementa el patrón MVC (Modelo-Vista-Controlador). El uso de *Struts* facilita el mantenimiento, creación y desarrollo de aplicaciones Web.

  El framework *Struts* proporciona herramientas en la **Vista** como son:
  - *JSP* y *JSTL* (Java Server Pages Standar Tag Libraries)
  - *Internacionalización* (LOCALE)
  - *Formularios* y su *Validación*
  - Plantillas (TILES)

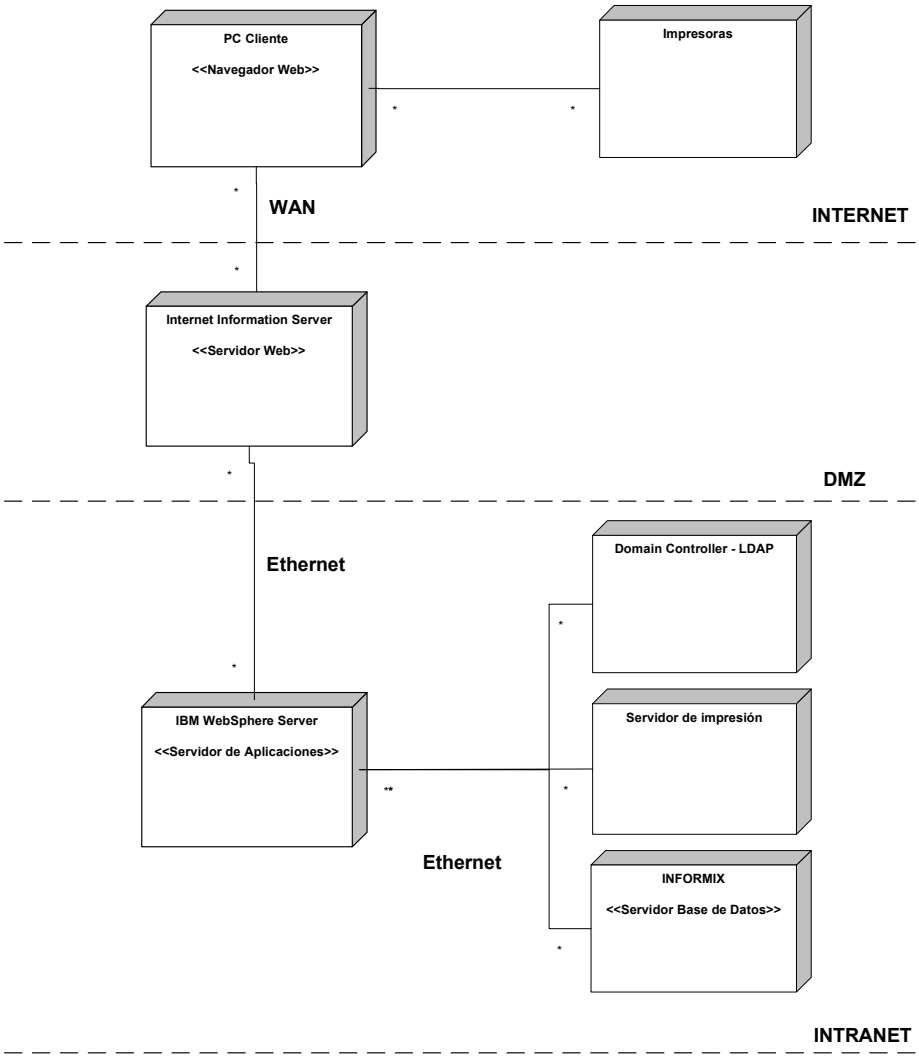
  Y en el **Controlador**:
  - *ActionServlet*: recibe las peticiones del Servidor Web y utiliza el fichero de configuración *Struts-config.xml*, en el que se define la navegación, para gestionar y redirigir todas las peticiones recibidas
  - *ActionForm*: proporciona los métodos necesarios (*Get* y *Set*) para el acceso a los datos de los formularios
  - *Action*: definen las acciones a realizar por cada formulario

- **Capa Lógica de Negocio (Modelo)**. Implementa las clases de la lógica de negocio del sistema. La implementación de las clases se realizará mediante lenguaje Java y se utilizarán patrones de diseño identificados para el desarrollo de sistemas bajo arquitectura J2EE

- **Capa Datos**. El sistema trabajará contra la base de datos definida por el sistema B2e y utilizará también las bases de datos del sistema de Gestión de Personal y nóminas de RRHH.

6.9. Particionamiento físico del Sistema.

A continuación se muestra la descripción de los niveles de arquitectura del sistema de información, representando las particiones físicas del sistema mediante nodos y comunicaciones entre nodos.



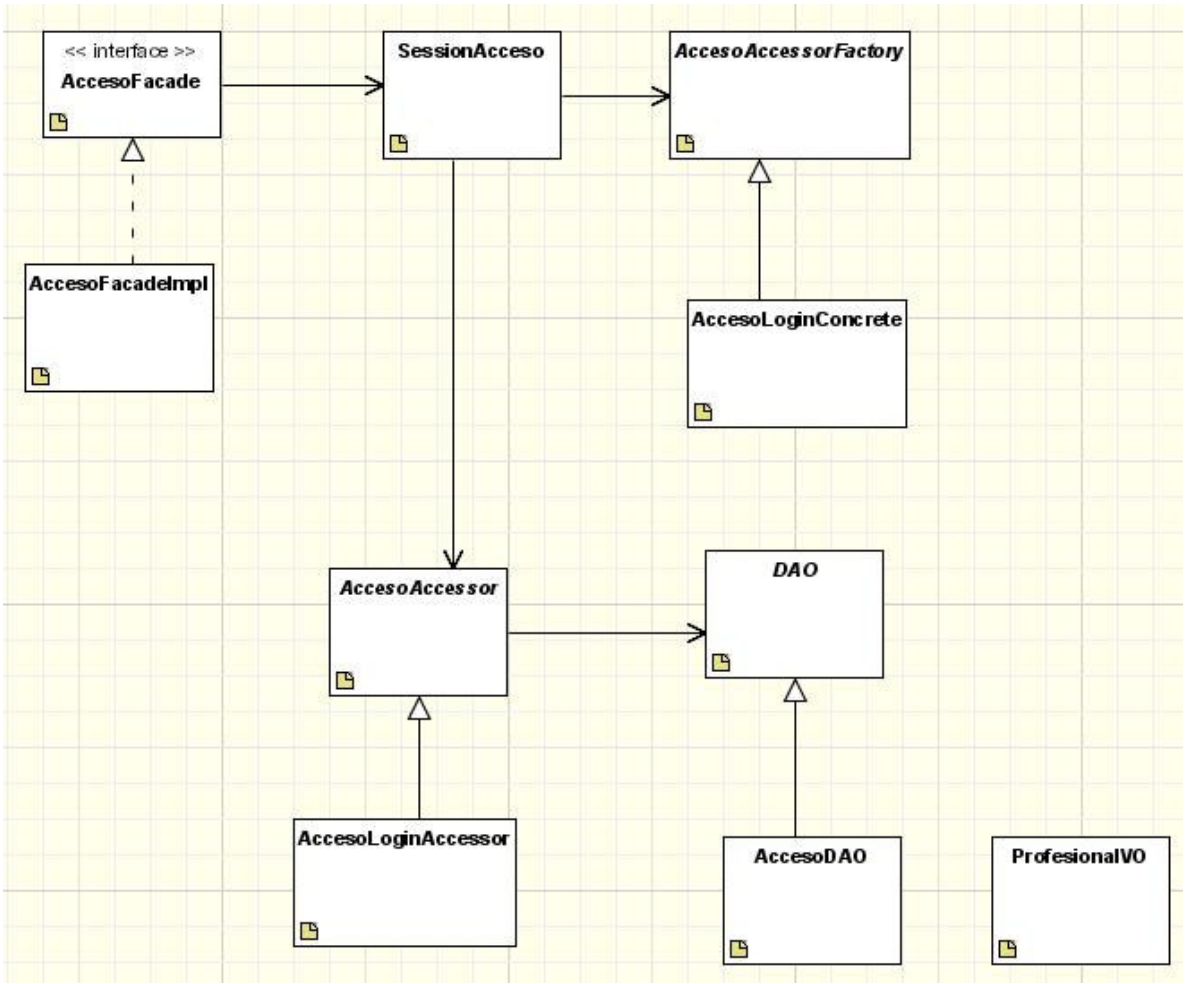
Identificación de los ***nodos*** incluidos en el *Diagrama de Despliegue* del sistema:

- **PC Cliente**. Equipos clientes desde los que se accederá al sistema y que pueden ser equipos conectados a la *Intranet* o equipos externos que accedan a través de *Internet*
- **Impresoras**. Dispositivos para la impresión de la documentación generada por la aplicación

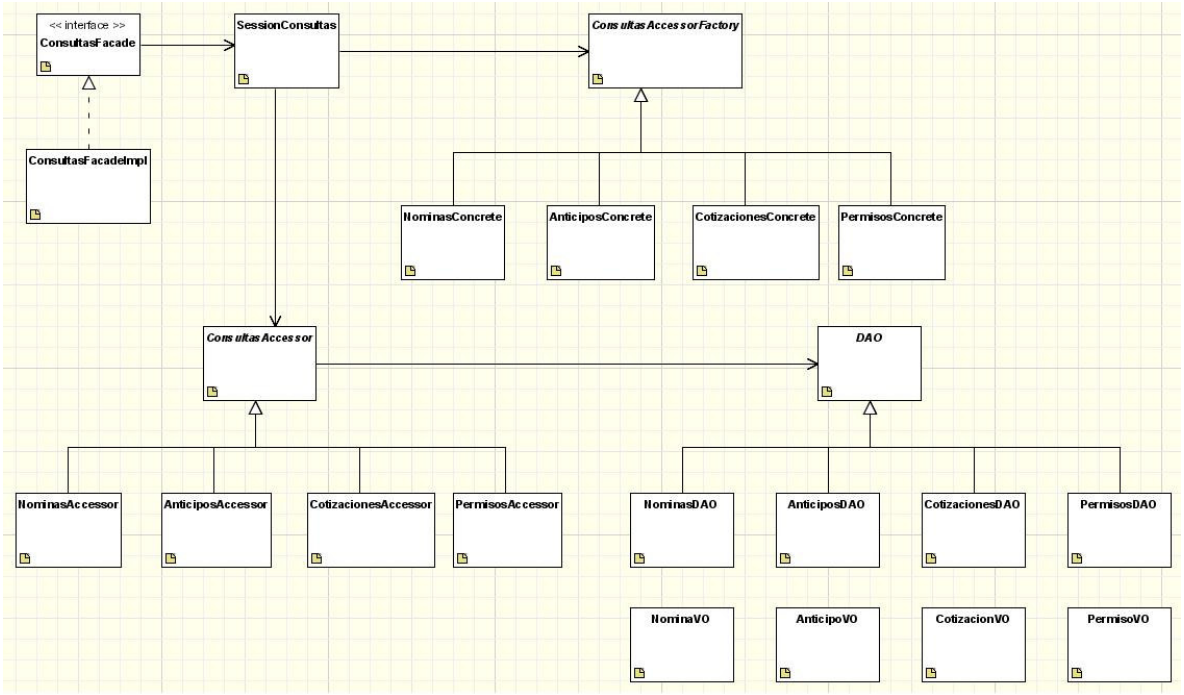
- **Internet Information Server (DMZ)**. Servidor Web situado en la *Demilitarized Zone* (DMZ) que recibe las peticiones realizadas por los clientes *Internet* y las redirige al Servidor de Aplicaciones,
- **IBM HTTP Server** . Servidor Web interno que recibirá todas las peticiones realizadas por los clientes de la *Intranet* y las redirige al Servidor de Aplicaciones en el que se ha desplegado la aplicación.
- **IBM WebSphere Application Server.** Servidor de aplicaciones en el que se han desplegado todos los componentes y elementos de la aplicación
- **Active Directory - LDAP**. El sistema utilizará el AD para validar los usuarios y los certificados que intentan acceder a la aplicación.
- **INFORMIX**. Sistema Gestor de Base de Datos en el que se ubicará la base de datos del sistema
- **Servidor de Impresión**. Servidor en la que se publican los informes llamados desde la aplicación. Los informes serán mostrados al usuario en formato PDF.

6.10. Diagramas de Clases.

6.10.1. Subsistema de Acceso



6.10.2. Subsistema de Consulta



6.10.3. Identificación de clases.

cat.uoc.http.view.actionforms

AnticiposAction: Clase de implementacion del Action correspondiente a la gestion de los anticipos
AvisoLegalAction: Clase de implementacion del Action correspondiente al aviso legal
CotizacionesAction: Clase de implementacion del Action correspondiente a la gestion de las cotizaciones
DatosAction: Clase de implementacion del Action correspondiente a la gestion de los datos
LocaleAction: Clase de implementacion del Action correspondiente a la gestion del idioma
LoginAction: Clase de implementacion del Action correspondiente a la gestion de la autenticacion en el portal
NominasAction: Clase de implementacion del Action correspondiente a la gestion de las nominas
PermisosAction: Clase de implementacion del Action correspondiente a la gestion de los permisos
PlantillaAction: Clase de implementacion del Action correspondiente al patron plantilla
SalirAction: Clase de implementacion del Action correspondiente a la salida

cat.uoc.http.view.actionforms

LocaleForm: Clase de Implementación del ActionForm correspondiente a la gestión de idiomas.
LoginForm: Clase de implementacion del ActionForm correspondiente a la gestion del login

cat.uoc.model.acceso.accesor

AccesoAccessor: Clase de implementacion del Accessor abstracto correspondiente a la gestion del acceso
AccesoLoginAccessor: Clase de implementacion del Accessor concreto correspondiente a la gestion del idioma

cat.uoc.model.acceso.dao

AccesoDAO: Clase de implementacion del DAO correspondiente a la gestion del acceso

cat.uoc.model.acceso.util.facade

AccesoFacade: Clase de implementacion de la Fachada correspondiente a la gestion del acceso
AccesoFacadeImpl: Clase de implementacion de la Fachada correspondiente a la gestion del acceso

cat.uoc.model.acceso.util.factory

AccesoAccesorFactory: Clase de implementacion del Factory Abstracto correspondiente a la gestion del acceso
AccesoLoginConcrete: Clase de implementacion del Factory correspondiente a la gestion del acceso

cat.uoc.model.acceso.util.session

SessionAcceso: Clase de implementacion del Bussines Delegate de Acceso

cat.uoc.model.acceso.vo

ProfesionalVO: Clase de implementacion del VO correspondiente al profesional

cat.uoc.model.consultas.accessor

AnticiposAcccessor: Clase de implementacion del Accessor de anticipos
ConsultasAccessor: Clase de implementacion del Accessor de consultas
CotizacionesAccessor: Clase de implementacion del Accessor de cotizaciones
NominasAccessor: Clase de implementacion del Accessor de consultas
PermisosAccessor: Clase de implementacion del Accessor de permisos

cat.uoc.model.consultas.dao

AnticiposDAO: Clase de implementacion del DAO de anticipos NominasAction:
CotizacionesDAO: Clase de implementacion del DAO de cotizaciones
NominasDAO: Clase de implementacion del DAO de consultas
PermisosDAO: Clase de implementacion del DAO de permisos

cat.uoc.model.consultas.util.facade

ConsultasFacade: Clase de implementacion de la Interface de la Fachada de consultas
ConsultasFacadeImpl: Clase de implementacion de la Fachada de consultas

cat.uoc.model.consultas.util.factory

AnticiposConcrete: Clase de implementacion de la Fabrica concreta de anticipos
ConsultasAccessorFactory: Clase de implementacion de la Fabrica abstracta concreta de consultas
CotizacionesConcrete: Clase de implementacion de la Fabrica concreta de cotizaciones
NominasConcrete: Clase de implementacion de la Fabrica concreta de nominas
PermisosConcrete: Clase de implementacion de la Fabrica concreta de permisos

cat.uoc.model.consultas.util.session

SessionConsultas: Clase de implementacion del Business Delegate de consultas

cat.uoc.model.consultas.vo

AnticipoVO: Clase de implementacion del VO de anticipos
CotizacionesVO: Clase de implementacion del VO de cotizaciones
NominaVO: Clase de implementacion del VO de nominas
PermisosVO: Clase de implementacion del VO de permisos

cat.uoc.model.util.accessor

AccessorConcrete: Clase de implementacion del Accessor concrete de Accesos

cat.uoc.model.util.dao

DAO: Clase de implementacion del DAO
TransactionalConnection: Clase de implementacion de obtencion de conexiones transaccionales

cat.uoc.model.util.datasource

DataSourceGenerator: Clase de obtencion del DataSource

cat.uoc.model.util.exceptions

AnticipoException: Clase exception de anticipos
CotizacionesException: Clase exception de cotizaciones
DatosException: Clase exception de datos
NominaExepction: Clase exception de nominas
PermisoException: Clase exception de permisos

cat.uoc.model.util.factory

AccessorFactory: Clase de implementacion del Accessor Factoru

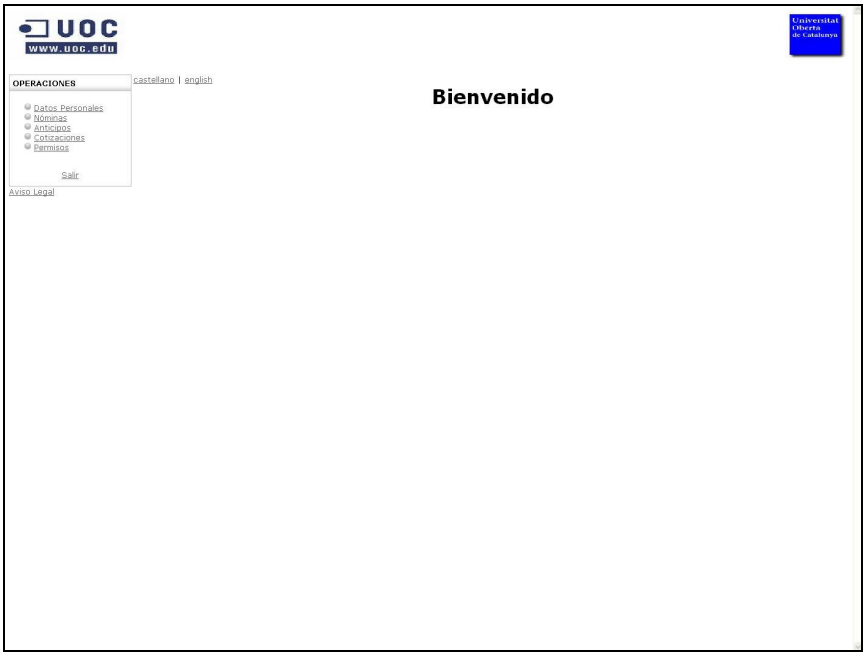
cat.uoc.servlets

MyPortalServlet: Clase de implementacion del Servlet inicial

6.11. Diseño de interfaz de usuario

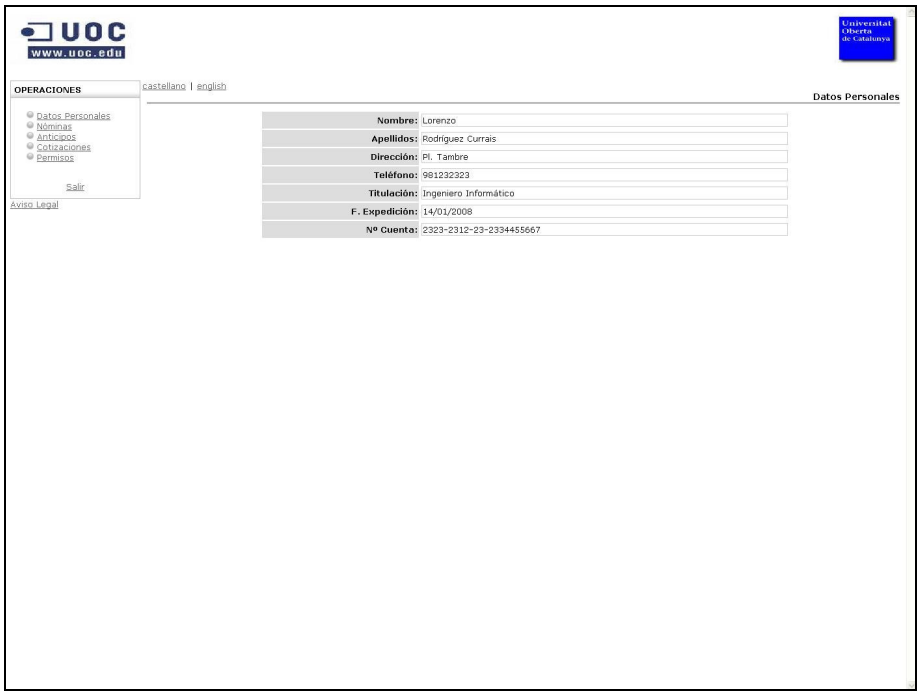
A continuación se presentan una serie de capturas del interfaz de usuario.

- Menú en Castellano:

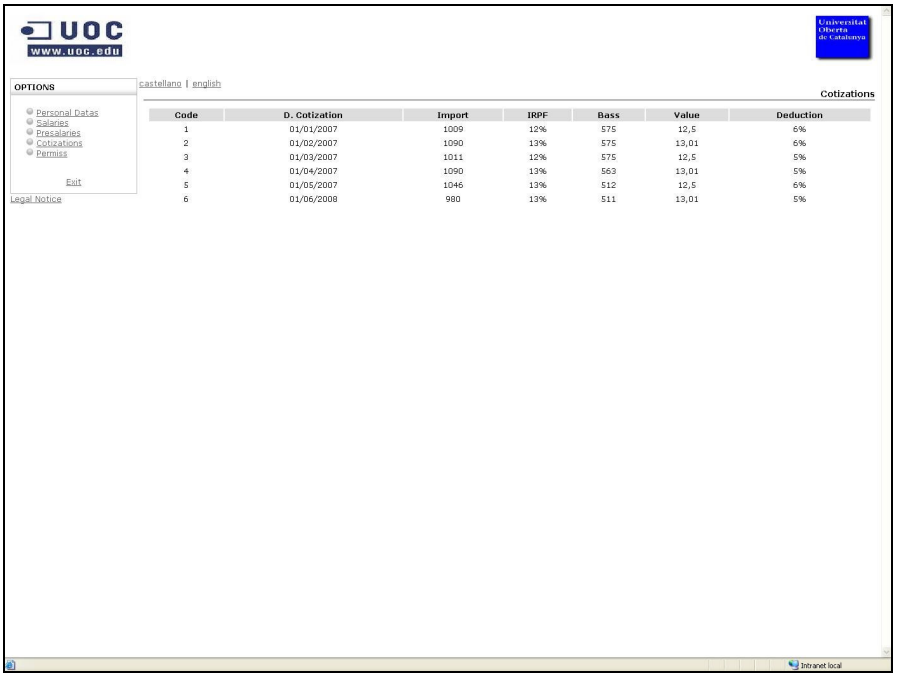


- Datos Personales:

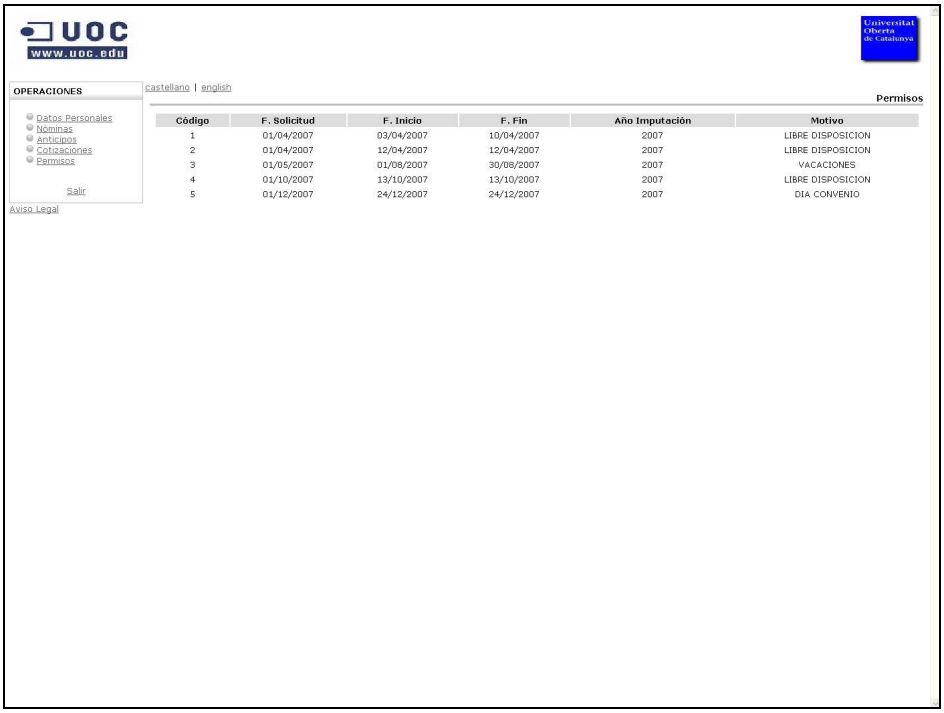
-



-
- Cotizaciones (inglés):


www.uoc.edu
Universitat Oberta de Catalunya
OPTIONS
castellano | english
Personal Datas
Salaries
Prescalaries
Cotizations
Permiss
Exit
Legal Notice
Cotizations
Code | D. Cotization | Import | IRPF | Bass | Value | Deduction
1 | 01/01/2007 | 1009 | 12% | 575 | 12,5 | 6%
2 | 01/02/2007 | 1090 | 13% | 575 | 13,01 | 6%
3 | 01/03/2007 | 1011 | 12% | 575 | 12,5 | 5%
4 | 01/04/2007 | 1090 | 13% | 563 | 13,01 | 5%
5 | 01/05/2007 | 1046 | 13% | 512 | 12,5 | 6%
6 | 01/06/2008 | 980 | 13% | 511 | 13,01 | 5%
Intranet local

- Permisos:


www.uoc.edu
Universitat Oberta de Catalunya
OPERACIONES
castellano | english
Datos Personales
Nóminas
Anticipos
Cotizaciones
Permisos
Salir
Aviso Legal
Permisos
Código | F. Solicitud | F. Inicio | F. Fin | Año Imputación | Motivo
1 | 01/04/2007 | 03/04/2007 | 10/04/2007 | 2007 | LIBRE DISPOSICION
2 | 01/04/2007 | 12/04/2007 | 12/04/2007 | 2007 | LIBRE DISPOSICION
3 | 01/05/2007 | 01/08/2007 | 30/08/2007 | 2007 | VACACIONES
4 | 01/10/2007 | 13/10/2007 | 13/10/2007 | 2007 | LIBRE DISPOSICION
5 | 01/12/2007 | 24/12/2007 | 24/12/2007 | 2007 | DIA CONVENIO

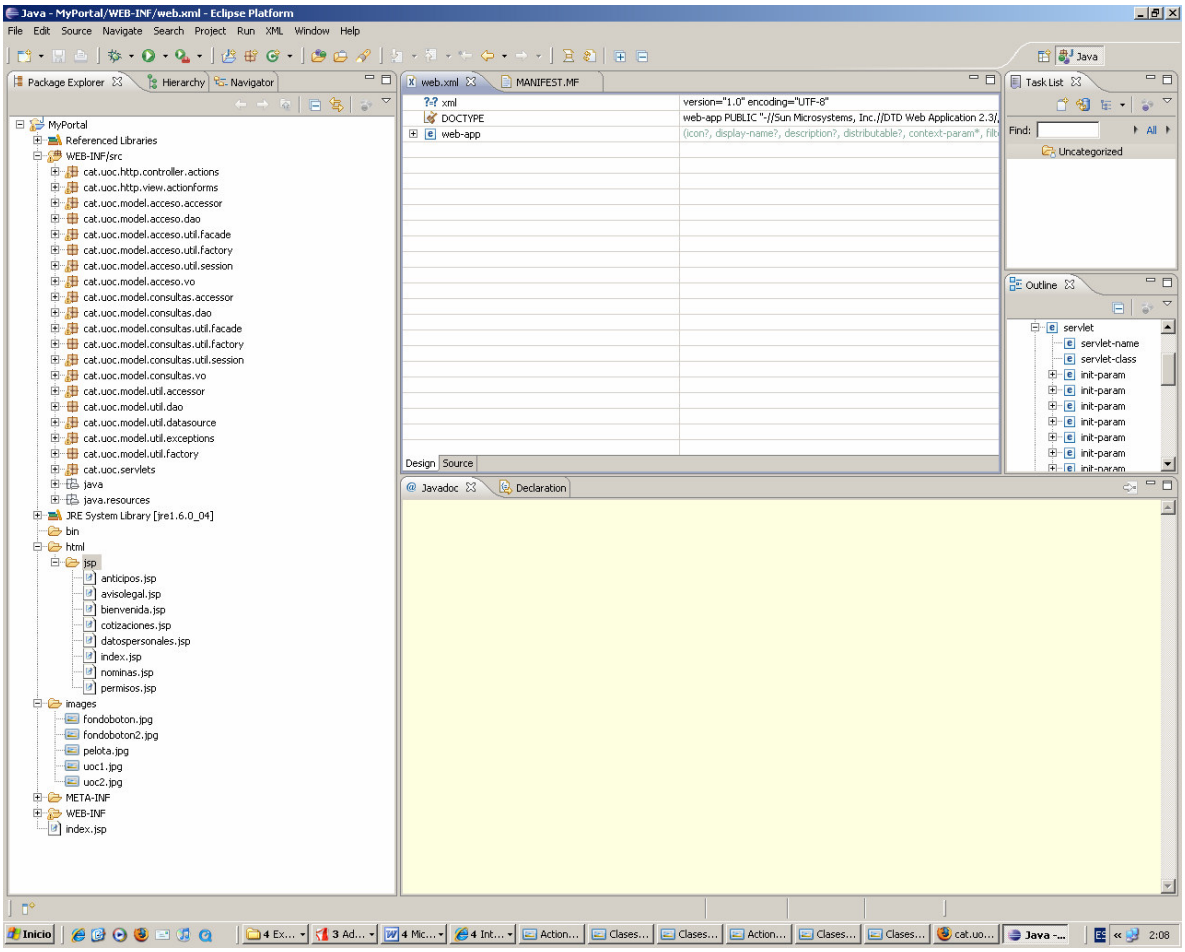
7. Implementación

Se explicará en este apartado como está estructurada la aplicación desarrollada utilizando el patrón MVC, los detalles técnicos y la finalidad de cada componente.

7.1. Estructura del proyecto

El proyecto tiene partes bien diferenciadas; la carpeta WEB-INF/SRC con las clases Java desarrolladas para la aplicación correspondientes a la lógica de negocio y las correspondientes al controlador, que serán las Action y Action Form de Struts.

La capa de presentación está en la carpeta html/jsp y que contienen los jsp's e images de las páginas web de la aplicación. Igualmente, en WEB-INF y META-INF se almacenan los ficheros de configuración y definición de Struts.



El despliegue de la aplicación se ha realizado en desarrollo sobre un servidor Tomcat 4.1, y es ahí donde se ha configurado la conexión a la Base de datos a través de JDBC.

7.2. Pruebas.

El plan de pruebas ha sido exhaustivo en el entorno de desarrollo, y se ha dirigido básicamente a comprobar el correcto funcionamiento de los métodos de las clases que componen la lógica de negocio de la aplicación. Se han probado todas las funcionalidades de consulta y acceso.

8. **Conclusiones.**

Se ha realizado una valoración de la cumplimentación del alcance funcional y se ha retrasado para una próxima fase el desarrollo de las siguientes funcionalidades recogidas en el análisis inicial:

- Ayuda
- Ayuda On-Line
- Solicitud de cambio de datos personales.

La valoración personal en cuanto a los tiempos y a las técnicas empleadas ha sido satisfactoria, consiguiendo los objetivos técnicos en el desarrollo de la aplicación trazados al principio de este informe.

Anexo I

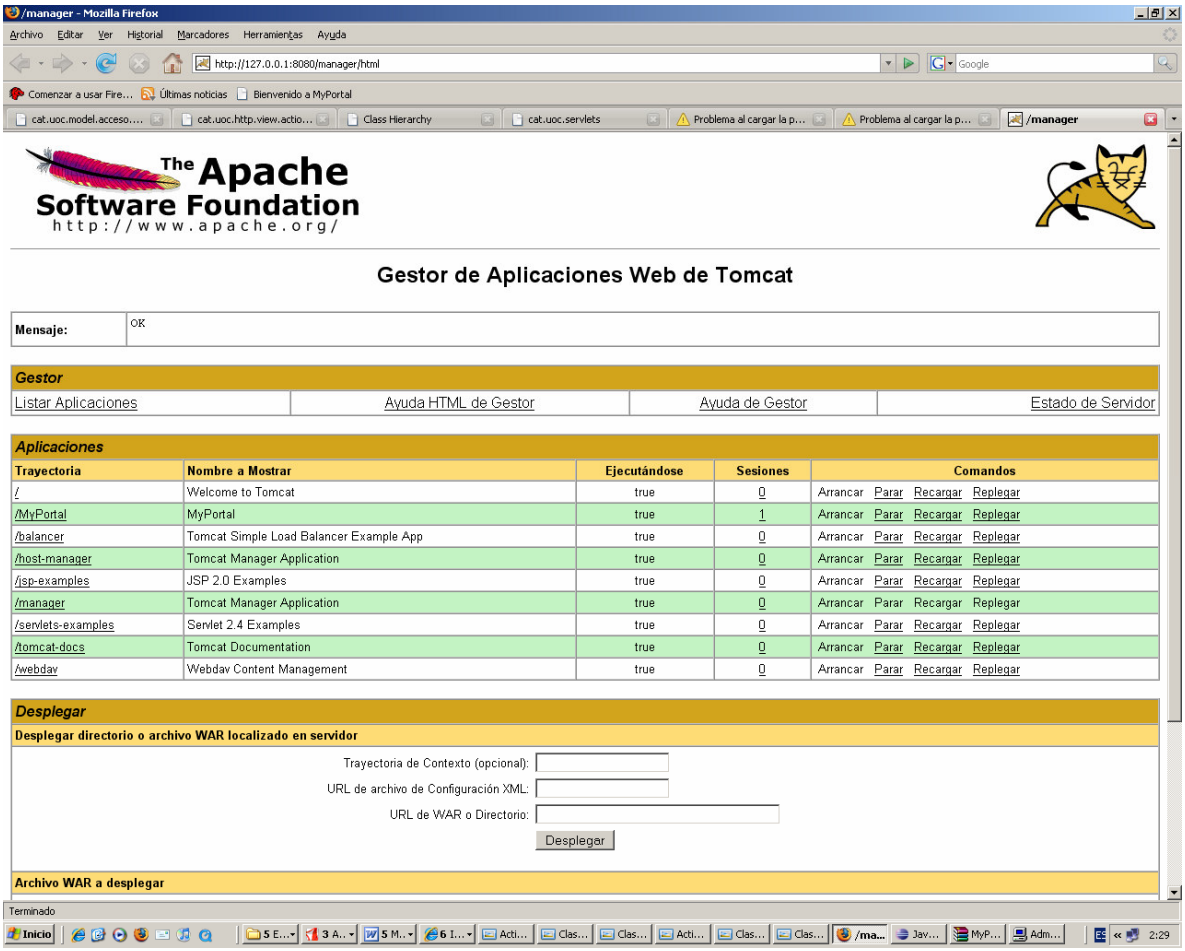
Se adjuntan al proyecto los ficheros:

- MyPortal.zip con toda la estructura del proyecto.
- JDoc.zip con la documentación generada en JavaDoc desde el IDE
- El fichero script.txt que contiene el código SQL para crear las tablas en desarrollo y poder probar la aplicación (se incluye el registro de usuario=”Lorenzo” y password =”Lorenzo” para el login en la aplicación)

Procedimiento de instalación en Tomcat 4.1

Instalar el Tomcat. Por ejemplo el 4.1

1. Desplegar la aplicación en el directorio \webapps de Tomcat o desplegarla desde el administrador del servidor de aplicaciones:



2. Incluir los drivers de Informix (ifxjdbc.jar) en el directorio ‘***common/lib***’ de la instalación
3. Configurar el Pool de Conexiones de la aplicación. Para ello hay que añadir las siguientes líneas en el fichero ‘*server.xml*’ situado en la carpeta ‘***conf***’ (en la sección de los contextos)

```
<!-- CONTEXTO MyPortal para datasource-->
    <Context path="/MyPortal" docBase="MyPortal"   debug="5" reloadable="true" crossContext="true">
      <Logger className="org.apache.catalina.logger.FileLogger" prefix="localhost_DBTest_log." suffix=".txt"
              timestamp="true"/>
      <Resource name="jdbc/MyPortal" auth="Container"  type="javax.sql.DataSource"/>
      <ResourceParams name="jdbc/MyPortal">
```

```
            <parameter>
              <name>factory</name>
              <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
            </parameter>
            <!-- Maximum number of dB connections in pool. Make sure you  configure your mysqld max_connections
large enough to handle  all of your db connections. Set to 0 for no limit.  -->
            <parameter>
              <name>maxActive</name>
              <value>100</value>
            </parameter>
            <!-- Maximum number of idle dB connections to retain in pool.Set to 0 for no limit.  -->
            <parameter>
              <name>maxIdle</name>
              <value>30</value>
            </parameter>
            <!-- Maximum time to wait for a dB connection to become available  in ms, in this example 10 seconds. An
Exception is thrown if  this timeout is exceeded.  Set to -1 to wait indefinitely.  -->
            <parameter>
              <name>maxWait</name>
              <value>10000</value>
            </parameter>
            <!-- MySQL dB username and password for dB connections  -->
             <parameter>
             <name>username</name>
             <value>informix</value>
            </parameter>
            <parameter>
             <name>password</name>
             <value>informix</value>
            </parameter>
            <!-- Class name for mm.mysql JDBC driver -->
            <parameter>
               <name>driverClassName</name>
               <value>com.informix.jdbc.IfxDriver</value>
            </parameter>
            <!-- The JDBC connection url for connecting to your MySQL dB. The autoReconnect=true argument to the url
makes sure that the  mm.mysql JDBC Driver will automatically reconnect if mysqld closed the connection.  mysqld by
default closes idle connections after 8 hours.  -->
            <parameter>
              <name>url</name>
                  <!-- <value>jdbc:informix-sqli://lorenzo-port:1526/myportal:informixserver=ol_iago</value>  -->
                  <value>jdbc:informix-sqli://HOST:PORT/myportal:informixserver=SERVER</value>
            </parameter>
          </ResourceParams>
        </Context>
    <!-- FIN DE CONTEXTO PARA MyPortal -->
```

4. Ejecutar la aplicación utilizando utilizando el root de apache (por ip, o a partir del localhost)