SGOCF
Sistema de Gestión de Operaciones Comerciales de Financiación

Memoria

Trabajo de Fin de Carrera

Ingeniería Técnica de Informática de Sistemas

Autor: David Rodríguez Villar

Consultor: Oscar Escudero

14 de enero de 2008

2 Dedicatoria y agradecimientos

A David, María y Carmen.

Formáis una parte muy importante

de mi camino hacia la felicidad

y habéis estado soportándome durante

todas mis ausencias para

conseguir terminar este trabajo.

3 Resumen

Para la realización de este trabajo de fin de carrera, se ha escogido la tecnología J2EE (ahora nombrada JEE). Esta tecnología se ha convertido en el referente empresarial para el desarrollo de aplicaciones distribuidas.

El trabajo de fin de carrera se ha realizado como un proyecto de desarrollo de software, siguiendo un ciclo de vida en cascada. El proyecto al final tiene dos objetivos. El primero, el inicial y funcional, consistente en el desarrollo de un sistema de información (una aplicación java empresarial) que satisfaga las necesidades de una empresa financiera respecto a la gestión de las operaciones comerciales de financiación que realiza. El segundo, que surgió en la fase de análisis del proyecto, el desarrollo de un arquitectura general para aplicaciones empresariales basada las mejores prácticas de desarrollo en JEE y que permitiese un aumento en la productividad de desarrollo de dichas aplicaciones a la vez que una uniformidad en su estilo y estructura con el fin de utilizarla en nuevos proyectos de desarrollo posteriores (para esta empresa u otras).

En el ámbito funcional se ha completado el desarrollo dentro del cambio de alcance experimentado al introducirse el ámbito tecnológico de la arquitectura general.

En el ámbito tecnológico se ha desarrollado al completo una arquitectura general para aplicaciones empresariales, produciendo una gran satisfacción por el logro obtenido y las características que reúne. De hecho se está considerando continuar con el desarrollo de un generador de aplicaciones sobre esta arquitectura a partir de las definiciones de objetos sencillos (POJO) del modelo de negocio.

La arquitectura está construida sobre *Echo2* para el interfaz de usuario, *Hibernate* para la gestión de la persistencia y *Spring Framework* como pegamento para unirlo todo. Esta arquitectura proporciona un conjunto de servicios a las aplicaciones desarrolladas sobre ella como son la gestión de seguridad y permisos de grupos de usuarios, internacionalización y gestión centralizada de mensajes y literales de la aplicación, un Framework para el interfaz de usuario que simplifica su construcción y da soporte para las validaciones de campos y el control de la seguridad y el aspecto de cliente pesado dentro de un cliente ligero así como clases de ayuda para trabajar con la persistencia, las transacciones y los criterios de las búsquedas de información.

En los siguientes puntos de este documento, se tratarán con detalle todos los aspectos de las etapas por las que ha pasado el proyecto, para finalmente poder llevar a cabo una conclusión y un análisis de los resultados obtenidos.

4 Índice

5 Memoria

5.1 Introducción

5.1.1 Justificación

5.1.1.1 Justificación del TFC

El trabajo de fin de carrera (TFC) es una asignatura que está pensada para realizar un trabajo de síntesis de los conocimientos adquiridos en otras asignaturas de la carrera y que requiera ponerlos en práctica conjuntamente en un trabajo concreto. Normalmente el TFC es un trabajo eminentemente práctico y vinculado al ejercicio de la informática.

Hace ya unos años que la industria del software está adoptando cada vez más la orientación a objetos. Por otra parte, las aplicaciones de empresa, siguiendo las tendencias actuales, están adoptando cada vez más una estructura distribuida. Esto, añadido a la imposición amplia de Internet a nivel empresarial hace que cada vez más las aplicaciones de empresa usen la red como vehículo de comunicación básico.

Además, la posibilidad de usar la Web como interfaz está haciendo que muchas aplicaciones antiguas se estén trasladando hacia este entorno. El mundo del desarrollo del software reacciona cada vez con más rapidez a las necesidades empresariales, y en la actualidad ya han aparecido varias tecnologías para facilitar y estandarizar el desarrollo de aplicaciones en este entorno.

Concretamente, la tecnología Java se ha constituido desde ya hace unos años en un puntal fuerte en este aspecto. El Java como lenguaje de desarrollo, junto con la arquitectura JEE se han convertido en un estándar en el mundo de la industria para el desarrollo distribuido de aplicaciones empresariales a Internet.

Esta área de trabajo fin de carrera constituye una propuesta para que el estudiante se introduzca o profundice en este apasionante mundo del desarrollo para Internet usando tecnología Java. Esto permitirá al estudiante realizar un trabajo práctico con tecnologías concretas que son usadas ampliamente en la empresa: Servlets, JSP, EJB, arquitectura JEE, JAAS, JDBC, SGBD relacionales, JMS, JNI y otros..

5.1.1.2 Contexto en el que se desarrolla

Un amigo del estudiante tiene la necesidad de cubrir una parte de los procesos de trabajo de su empresa con una aplicación empresarial que de soporte a dichos procesos (el amigo es el director de dicha empresa financiera).

Aprovechando que el estudiante tiene que realizar el trabajo de fin de carrera y ha sido aceptado dentro del área JEE, área que como hemos mencionado en el anterior apartado, tiene como parte de sus objetivos el desarrollo de una aplicación empresarial, se pueden atender dicha necesidad con la solución que se desarrolle en el trabajo de fin de carrera JEE.

5.1.1.3 Aportación del TFC

La realización del TFC contribuirá con las siguientes aportaciones:

1. El estudiante podrá profundizar en los conocimientos de arquitectura JEE
2. La empresa financiera aumentará su productividad con el uso de la aplicación desarrollada en el TFC
3. La empresa financiera ahorrará el coste de contratar el desarrollo de la aplicación que necesitan

4. Se desarrollará un arquitectura de aplicaciones empresariales sobre la que se construirá la aplicación objetivo y que será reutilizada en posteriores proyectos de desarrollo.

5.1.2 Objetivos del TFC

5.1.2.1 Objetivos de la asignatura

Los objetivos de la asignatura son:

- El estudiante realice un trabajo de síntesis de los conocimientos adquiridos en otras asignaturas de la carrera y que requiera ponerlos en práctica conjuntamente en un trabajo concreto.
- El estudiante realice un trabajo eminentemente práctico y vinculado al ejercicio de la informática.
- El estudiante profundice en el conocimiento práctico de la plataforma JEE

5.1.2.2 Alcance del proyecto

El alcance del proyecto es el desarrollo de un sistema de información (una aplicación java empresarial) que satisfaga las necesidades de una empresa financiera respecto a la gestión de las operaciones comerciales de financiación que realiza.

Para delimitar el alcance (que será completamente especificado en la definición de los requisitos del sistema en la fase de análisis), en este apartado vamos especificar el contexto y los requisitos a grandes rasgos que deberá cumplir el sistema.

La empresa se dedica a financiar a personas físicas ofreciendo:

- Prestamos personales
- Financiación de operaciones de compra-venta de inmuebles
- Reunificación de deudas

La reunificación de deudas cubre deudas derivadas de préstamos hipotecarios, prestamos personales, tarjetas de crédito y otros conceptos que deben ser estudiados por la empresa para autorizarlos.

Cada operación de financiación que realiza la empresa está definida en lo que denominan un expediente. Un expediente recoge los siguientes datos:

- Solicitante o solicitantes (titulares)
- Tipo de solicitud
    - Préstamo personal
    - Compra-venta de inmuebles
    - Reunificación de deudas
    - Compra-venta de inmuebles con reunificación de deuda
- El colaborador que gestiona el expediente.
- Si el expediente es facturable (ya que también se actúa como intermediadores de otras empresas que realizan estos servicios)
- En los casos de reunificación que incluyan una hipoteca, el tipo de operación:
    - Constitución
    - Subrogación
    - Novación
    - Subrogación- Novación
- Las deudas que se financian

- La datos de la cuenta y el banco con el que tienen contraídas los titulares la deuda

La gestión del expediente implica el hacer una oferta de financiación para el cliente (titulares del expediente)

La oferta incluye el cálculo de todos los conceptos implicados aplicables al caso de financiación (importes, gastos y comisiones), así como otros factores que llevan al cálculo de la cuota resultante para la financiación, como por ejemplo:

- Comisiones de intermediación
- Tasaciones
- Gastos de compraventa
- Cancelaciones registrales
- Gastos de constitución
- Gastos de compraventa
- Gastos de subrogación
- Gastos de novación
- Comisiones de apertura
- Años de financiación
- Tipo de interés compuesto aplicado a la operación de financiación
- Cuotas anteriores que pagaban los titulares por sus deudas.

Estos conceptos no son constantes en todos los casos, siendo en ocasiones función del tipo de operación y los importes de las deudas y deberán ser tenidos en cuenta.

El sistema de información debe proporcionar:

- Gestión de expedientes
- Gestión de titulares asociadas a expedientes
- Gestión de bancos asociados a expedientes
- Gestión de prestamos a financiar asociada al expediente
- Gestión de colaboradores
- Gestión de los parámetros de las comisiones, importes y tarifas implicadas en el cálculo de las condiciones presentadas en las ofertas de las operaciones.
- Gestión de ofertas asociadas a expedientes
- Gestión de facturas asociadas a expedientes

Además se requiere que las ofertas y las facturas sean emitidas como libros de cálculo Microsoft Excel.

Nota: En el desarrollo del proyecto se ha producido un **cambio del alcance**. Este cambio surgió en la fase de análisis del proyecto: el desarrollo de un arquitectura general para aplicaciones empresariales basada las mejores prácticas de desarrollo en JEE y que permitiese un aumento en la productividad de desarrollo de dichas aplicaciones a la vez que una uniformidad en su estilo y estructura con el fin de utilizarla en nuevos proyectos de desarrollo posteriores (para esta empresa u otras).

Con el cliente de la aplicación se negoció un retraso de ciertas funcionalidades en aras del desarrollo de la arquitectura (de la que posteriormente se beneficiará en este y otros proyectos).

Del alcance inicial se ha retrasado a posteridad de la entrega de este TFC:

- Gestión de ofertas asociadas a expedientes
- Gestión de facturas asociadas a expedientes
- Dentro de la gestión de expedientes, los cálculos de análisis económico de expedientes excepto el de préstamo personal que sí entra en el alcance.

Al alcance inicial se han añadido:

- Desarrollo de una arquitectura general para aplicaciones empresariales: Gestión de seguridad y permisos de grupos de usuarios, internacionalización y gestión centralizada de mensajes y literales de la aplicación, un Framework para el interfaz de usuario que simplifica su construcción y da soporte para el control de la seguridad y el aspecto de cliente pesado dentro de un cliente ligero así como clases de ayuda para trabajar con la persistencia, las transacciones y los criterios de las búsquedas de información
- Gestión de usuarios
- Gestión de grupos de usuarios
- Gestión de permisos a grupos de usuarios
- Gestión de seguridad en el acceso a las pantallas

5.1.3 Enfoque y método seguido

Dado que es un proyecto de desarrollo de un sistema de información para un cliente y que el proyecto en todos sus requerimientos puede implicar más tiempo del que hay disponible en el TFC, se ha estimado conveniente utilizar el ciclo de vida clásico (o en cascada) con objeto de desarrollar el nuevo sistema. En la actividad de especificación de requisitos de la fase de análisis, se tendrá un conocimiento exacto de lo que el cliente desea para su sistema y para conseguir una estimación ajustada al volumen de trabajo que permite el TFC, se decidirán que requisitos se cumplirán dentro del proyecto desarrollado en el TFC.

Considero que la utilización de este ciclo de vida, con mi experiencia de análisis y estimación permitirá la selección del alcance correcto para poder cumplir con las exigencias de la asignatura.

5.1.4 Planificación del proyecto

Ya que el proyecto sigue el ciclo de vida en cascada, las actividades se agruparán en fases que utilizarán como entrada los productos generados en la fase anterior y proporcionarán como salida nuevos productos que serán utilizados por la fase posterior.

Se ha tenido en cuenta el enfoque de la asignatura de evaluación continua para fijar como hitos externos las entregas para dichas evaluación continua (PEC).

Dado que tengo otras obligaciones (trabajo, familia y otras asignaturas) se ha considerado para la estimación una dedicación de 15 horas semanales y distribuidas en días laborables de lunes a viernes (cada día que aparece equivale a 3 horas de trabajo)

Se ha utilizado Microsoft Project 2003 para realizar la planificación y con esta misma herramienta se realizará el control y seguimiento de la planificación.

5.1.4.1 Descomposición en fases, actividades e hitos

Id	Tarea	Dur,	Inicio	Fin	Dep.
1	**Fase de arranque de proyecto**	**6 días**	**20/09/2007**	**28/09/2007**	
2	Establecimiento del Alcance del Proyecto	3 días	20/09/2007	25/09/2007	
3	Definición del plan de proyecto	2 días	25/09/2007	27/09/2007	2
4	Revisión y aprobación de productos	1 día	27/09/2007	28/09/2007	3
5	***PEC1: Entrega***	0 días	28/09/2007	28/09/2007	1
6	**Fase de análisis de sistemas**	21 días	28/09/2007	29/10/2007	1
7	Especificación de Requisitos	7 días	28/09/2007	09/10/2007	
8	Análisis del sistema	9 días	09/10/2007	22/10/2007	7
9	Modelo de Negocio	1 día	09/10/2007	10/10/2007	
10	Casos de uso	5 días	10/10/2007	17/10/2007	9
11	Modelo de dominio	3 días	17/10/2007	22/10/2007	10
12	Definición de Interfaces de Usuario	3 días	22/10/2007	25/10/2007	11
13	Especificación de plan inicial de pruebas	1 día	25/10/2007	26/10/2007	12
14	Revisión y aprobación de productos	1 día	26/10/2007	29/10/2007	13
15	**PEC2: Entrega**	0 días	29/10/2007	29/10/2007	6

Id	Tarea	Dur,	Inicio	Fin	Dep.
16	**Fase de diseño de sistema**	22 días	29/10/2007	28/11/2007	6
17	Diseño	18 días	29/10/2007	22/11/2007	
18	Diseño de la Arquitectura Lógica del Sistema	18 días	29/10/2007	22/11/2007	
19	Diseño de la Arquitectura Física del Sistema	18 días	29/10/2007	22/11/2007	
20	Diseño Físico de Datos	18 días	29/10/2007	22/11/2007	
21	Diseño de la Interfaz de Usuario	18 días	29/10/2007	22/11/2007	
22	Diseño de la Lógica Interna de los Componentes	18 días	29/10/2007	22/11/2007	
23	Especificación Técnica del Plan de Pruebas	3 días	22/11/2007	27/11/2007	17
24	Revisión y aprobación de productos	1 día	27/11/2007	28/11/2007	23
25	***PEC3: Entrega***	0 días	17/12/2007	17/12/2007	16
26	**Fase de construcción de sistemas**	25 días	28/11/2007	02/01/2008	16
27	Preparación del entorno de construcción y preproducción	1 día	28/11/2007	29/11/2007	
28	Generación del código de componentes y procedimientos	20 días	29/11/2007	27/12/2007	27
29	Revisión y realización de pruebas	2 días	27/12/2007	31/12/2007	28
30	Generación de documentación de usuario y explotación	1 día	31/12/2007	01/01/2008	29
31	Revisión y aprobación de productos	1 día	01/01/2008	02/01/2008	30
32	**Fase de implantación y aceptación de sistema**	2,9 días	02/01/2008	07/01/2008	26
33	Preparación de plan de implantación	0,5 días	02/01/2008	02/01/2008	
34	Instalación en Preproducción	0,2 días	02/01/2008	02/01/2008	33
35	Pruebas en Preproducción	1 día	02/01/2008	03/01/2008	34
36	Instalación en producción	0,2 días	03/01/2008	04/01/2008	35
37	Revisión y aprobación de productos	1 día	04/01/2008	07/01/2008	36
38	**Fase de cierre de proyecto**	5 días	07/01/2008	14/01/2008	32
39	Finalización de memoria	3 días	07/01/2008	10/01/2008	
40	Elaboración de presentación	1 día	10/01/2008	11/01/2008	39
41	Revisión y aprobación de productos	1 día	11/01/2008	14/01/2008	40
42	***Entrega Final: Memoria, presentación y software***	0 días	14/01/2008	14/01/2008	38

5.1.4.2 Descripción de fases

Fase de Arranque de Proyecto

Esta fase persigue delimitar y definir el proyecto en cuanto a ámbito, alcance, objetivos, infraestructura básica, organización y planificación temporal.

Como resultado de esta fase se tendrá el Plan de proyecto (Plan de trabajo) dentro de la memoria.

Fase de análisis de sistemas de información

Esta fase persigue la obtención de una especificación detallada del sistema de información que satisfaga las necesidades de información de los usuarios y sirva de base para el posterior diseño del sistema.

Como resultado de esta fase se tendrá el capítulo de análisis del sistema que incluirá análisis de requisitos del sistema y el análisis propio del sistema, todo ello dentro de la memoria.

Fase de diseño de sistema de información

Esta fase persigue la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información.

Como resultado de esta fase se tendrá el capítulo de diseño del sistema dentro de la memoria.

Fase de construcción de sistemas de información

Esta fase persigue construir (codificar) el sistema de información conforme al diseño, ejecutar las pruebas para verificar que cumple con las especificaciones, y además preparar la documentación y formación para el usuario final.

Como resultado de esta fase se tendrá el capítulo de construcción del sistema dentro de la memoria y los anexos que se necesiten.

Fase de implantación de sistemas de información

Esta fase persigue realizar el paso al entorno de preproducción del nuevo software (en este caso se toma el entorno de preproducción y no el de producción para poder cumplir con los requisitos de la asignatura y teniendo las garantías de la calidad del producto software realizado de tal forma que la instalación en producción del cliente sea un mero trámite).

Como resultado de esta fase se tendrá el capítulo de implantación del sistema dentro de la memoria y los anexos que se necesiten.

Fase de cierre de proyecto

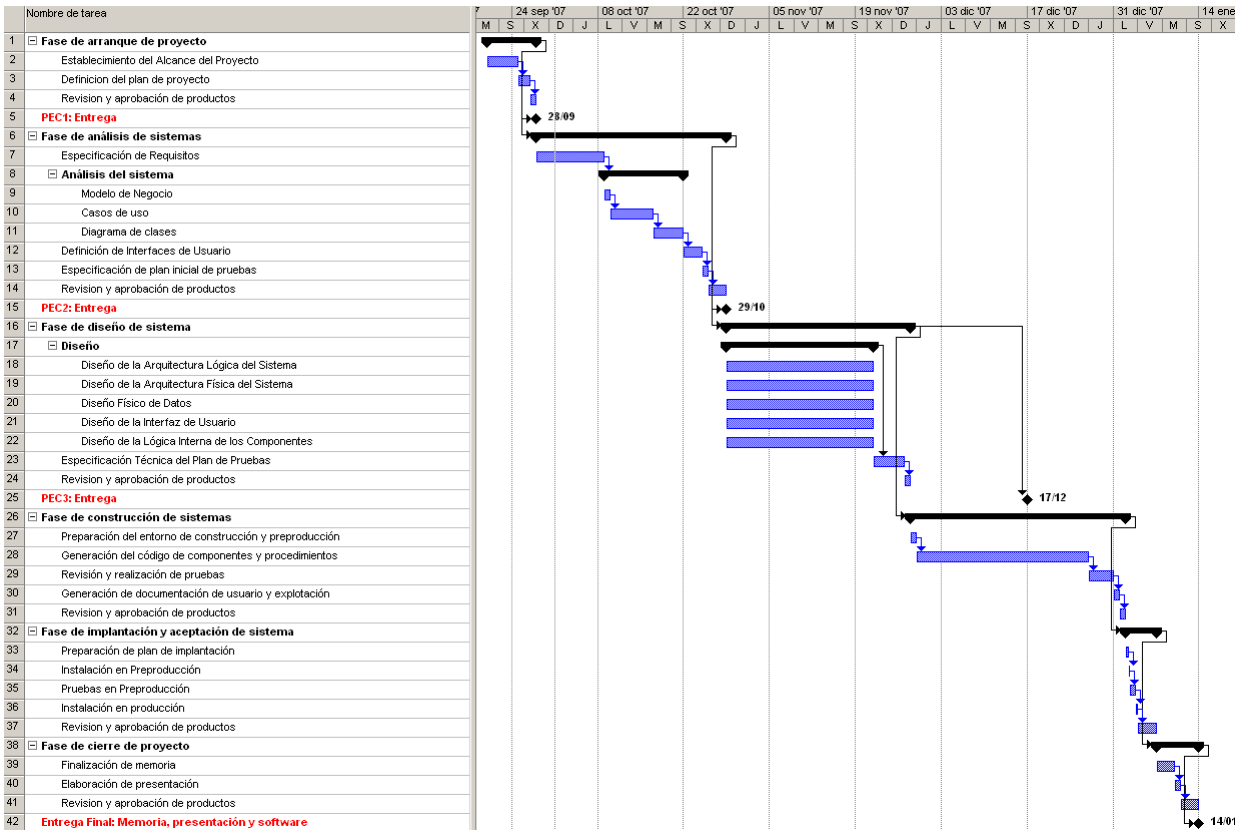
Esta fase persigue realizar el cierre del proyecto. En ella se obtendrán las conclusiones de la realización del proyecto, la terminación de la memoria del proyecto y el empaquetado y entrega de todos los productos del proyecto (incluyendo el sistema desarrollado).

Como resultado de esta fase se completará y revisará la memoria desarrollándose en concreto el capítulo de conclusiones. También se elaborará la presentación sobre el proyecto.

5.1.4.3 Hitos

Hito	Fecha
PEC1: Entrega	28/09/2007
PEC2: Entrega	29/10/2007
PEC3: Entrega	17/12/2007
Entrega Final: Memoria, presentación y software	14/01/2008

5.1.4.4 Cronograma



5.1.5 Productos obtenidos

Los productos obtenidos de este Trabajo Final de Carrera son los siguientes:

- El plan de proyecto, que recoge que recoge la planificación y estimación de las tareas necesarias para llevar a cabo los objetivos previstos
- El producto en sí, que es el software desarrollado (aplicación JEE) y su documentación técnica asociada.
- La presentación, que resume de forma clara y concisa el trabajo realizado y los resultados obtenidos.
- La presente memoria, que es el documento que sintetiza el todo el trabajo realizado a lo largo del proyecto incluyendo los productos de documentación resultado de las fases seguidas a lo largo del proyecto. (Plan de proyecto, análisis del sistema, diseño del sistema, construcción del sistema, implantación del sistema).

6 Análisis del sistema

6.1 Descripción general del sistema

El Sistema de Gestión de Operaciones Comerciales de Financiación (SGOCF) es un sistema que da soporte a la empresa en su actividad.

La empresa se dedica a la financiar a personas físicas ofreciendo:

- Prestamos personales
- Financiación de operaciones de compra-venta de inmuebles
- Reunificación de deudas

Para ello el sistema proporciona servicios para:

- Gestión de expedientes
- Gestión de titulares asociadas al expediente
- Gestión de bancos asociados al expediente
- Gestión de prestamos asociados al expediente
- Gestión de colaboradores
- Gestión de usuarios
- Gestión de grupos de usuarios
- Gestión de permisos a grupos de usuarios
- Gestión de seguridad en el acceso a las pantallas
- Gestión de los parámetros de las comisiones, importes y tarifas implicadas en el cálculo de las condiciones presentadas en las ofertas de las operaciones.
- Gestión de ofertas asociadas a expedientes
- Gestión de facturas asociadas a expedientes

6.2 Descripción general del entorno tecnológico

El sistema es una aplicación web empresarial con un backend de una base de datos.

El sistema sigue una arquitectura de 3 capas:

- Capa de presentación: los usuarios mediante un navegador web accederán al servidor donde se ejecuta la lógica de negocio del sistema.

- Capa de negocio: En ella se ejecutan las reglas de negocio del sistema.
- Capa de datos: en ella se almacena la información del sistema.

Estas capas lógicas se pueden separar hasta en tres equipos, aunque lo normal será que se ubiquen las dos últimas capas en un mismo equipo (servidor).

El navegador soportado será Internet Explorer 6 y 7 o Firefox 2, y los usuarios podrán usar cualquier sistema operativo en sus máquinas para acceder al sistema con el requisito de que la pantalla tenga una resolución mínima de 1024x768.

El sistema será desarrollado en la plataforma JEE, lo cual también permitirá que se puedan utilizar cualquier sistema operativo y utilizar servidores tanto de aplicaciones como de bases de datos con licencia gratuita con lo cual no será necesario comprar software de base.

6.3 Especificación de requisitos del sistema

6.3.1 Requisitos Funcionales

6.3.1.1 Gestión de expedientes

Se podrá consultar todos los expedientes que existen en el sistema.

En adelante aparece subrayado los datos que son de entrada por el colaborador que da de alta o modifica el expediente.

La información común que se recogerá para todos los expedientes será:

- Número de expediente
- Fecha del expediente
- Colaborador que gestiona el expediente
- Si el expediente es facturable
- Tipo de solicitud (préstamo personal, compraventa, reunificación, compraventa reunificación)
- Titulares del expediente
- Capital del expediente (este se calcula en base a las tasas, gastos y comisiones que más adelante se especifican según los tipos de préstamo)
- Años a financiar
- Interés anual
- Importe de cuota (estos campos se calculan en función de los demás datos del expediente)
- Diferencia con la cuota anterior (estos campos se calculan en función de los demás datos del expediente)
- Porcentaje de diferencia (estos campos se calculan en función de los demás datos del expediente)
- Ingresos de los titulares (estos campos se calculan en función de los demás datos del expediente)es
- Ratio de endeudamiento (estos campos se calculan en función de los demás datos del expediente)

Para los expedientes de tipo de **préstamo personal** se recogerá o calculará la siguiente información:

- Comisión
    - Importe. IVA. total (se calcula en función del importe total de los prestamos del expediente)

- Tasación
    - Importe. IVA. Total, Seguros. Importe. IVA. Total
- Gastos de constitución (se calculan en función de la suma de los importes anteriores)
    - Tipo aplicable. Importe
- Comisión Apertura (se calculan en función de la suma de los importes anteriores)
    - Tipo aplicable. Importe

Para los expedientes de tipo de **compra venta** se recogerá o calculará la siguiente información:

- Comisión
    - Importe. IVA. Total (se calcula en función del importe total de los prestamos del expediente)
- Tasación
    - Importe. IVA. Total
- Seguros
    - Importe. IVA. Total
- Gastos de compra venta (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Gastos de constitución (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Comisión Apertura (se calculan en función de la suma de los importes anteriores)
    - Tipo aplicable. Importe. Para los expedientes de tipo de **reunificación** se recogerá o calculará la siguiente información:

Se introduce un nuevo subtipo (constitución, subrogación, novación, subrogación + novación).

Si es del subtipo **constitución**:

- Comisión
    - Importe. IVA. Total (se calcula en función del importe total de los prestamos del expediente)
- Tasación
    - Importe. IVA. Total
- Seguros
    - Importe. IVA. Total.
- Gastos de cancelación registral (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Registro. Gestión. Total
- Gastos de constitución (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total

- Comisión Apertura (se calculan en función de la suma de los importes anteriores)
    - Tipo aplicable. Importe

Si es del subtipo **subrogación**:

- Comisión
    - Importe. IVA. Total (se calcula en función del importe total de los prestamos del expediente)
- Tasación
    - Importe. IVA. Total. Seguros. Importe. IVA. Total
- Gastos de novación (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Comisión Apertura (se calculan en función de la suma de los importes anteriores)
    - Tipo aplicable. Importe

Si es del subtipo **subrogación + novación**:

- Comisión
    - Importe. IVA. Total (se calcula en función del importe total de los prestamos del expediente)
- Tasación
    - Importe. IVA. Total
- Seguros
    - Importe. IVA. Total
- Gastos de subrogación (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Gastos de novación (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Comisión Apertura (se calculan en función de la suma de los importes anteriores)
    - Tipo aplicable. Importe

Para los expedientes de tipo de **compraventa + reunificación** se recogerá o calculará la siguiente información:

Se introduce un nuevo subtipo (constitución, subrogación, novación, subrogación + novación).

Si es del subtipo **constitución**:

- Comisión
    - Importe. IVA. Total (se calcula en función del importe total de los prestamos del expediente)
- Tasación
    - Importe. IVA. Total
- Seguros
    - Importe. IVA. Total

- Gastos de compra venta (se calculan en función de la suma de los importes anteriores)
    - Capital . Notaría. Impuestos. Registro. Gestión. Total
- Gastos de cancelación registral (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Registro. Gestión. Total
- Gastos de constitución (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Comisión Apertura (se calculan en función de la suma de los importes anteriores)
    - Tipo aplicable. Importe

Si es del subtipo **subrogación**:

- Comisión
    - Importe. IVA. Total (se calcula en función del importe total de los prestamos del expediente)
- Tasación
    - Importe. IVA. Total
- Seguros
    - Importe. IVA. Total
- Gastos de compra venta (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Gastos de subrogación (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Comisión Apertura (se calculan en función de la suma de los importes anteriores)
    - Tipo aplicable. Importe

Si es del subtipo **novación**:

- Comisión
    - Importe. IVA. Total (se calcula en función del importe total de los prestamos del expediente)
- Tasación
    - Importe. IVA. Total. Seguros. Importe. IVA. Total
- Gastos de compra venta (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Gastos de novación (se calculan en función de la suma de los importes anteriores)
    - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Comisión Apertura (se calculan en función de la suma de los importes anteriores)

- Tipo aplicable. Importe

Si es del subtipo **subrogación + novación**:

- Comisión
  - Importe. IVA. Total (se calcula en función del importe total de los prestamos del expediente)
- Tasación
  - Importe. IVA. Total. Seguros. Importe. IVA. Total
- Gastos de compra venta (se calculan en función de la suma de los importes anteriores)
  - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Gastos de subrogación (se calculan en función de la suma de los importes anteriores)
  - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Gastos de novación (se calculan en función de la suma de los importes anteriores)
  - Capital. Notaría. Impuestos. Registro. Gestión. Total
- Comisión Apertura (se calculan en función de la suma de los importes anteriores)
  - Tipo aplicable. Importe

Los expedientes se podrán dar de alta o modificar introduciendo todos los datos anteriormente señalados (los subrayados o que el sistema calcule los otros datos en base a las comisiones, tarifas y gastos).

Los expedientes se podrán eliminar.

La emisión de una oferta o una factura estará condicionada a que el expediente se haya dado de alta o modificado sin errores (ver requisitos correspondientes a ofertas y facturas)

6.3.1.2 Gestión de titulares asociados a un expediente

El sistema almacenará la información sobre titulares asociados a expedientes y se podrán modificar desde la gestión del expediente al que pertenecen.

A un expediente podrán ser asociados cero, uno o varios titulares.

La información sobre titulares que se recogerá será:

- Datos personales del titular:
  - DNI. Primer apellido. Segundo Apellido. Nombre. Dirección. Código postal. Población. Teléfono. Estado civil. Fecha de nacimiento
- Datos financieros del titular:
  - Tipo de titular (Avalista, Titular, Otros titulares). Ingresos netos mensuales. Cargos netos mensuales. Otros créditos mensuales
  - Comentarios que se quieran registrar sobre el titular.

Los titulares podrán ser dados de alta y modificados sus datos.

Los titulares podrán ser dados de baja siempre y cuando no exista un expediente en el que esté asociado.

6.3.1.3 Gestión de bancos asociados a un expediente

El sistema almacenará la información de bancos asociados a un expediente.

A un expediente de financiación podrá ser asociado cero, uno o varios bancos.

La información sobre el banco asociado al expediente será

- Nombre del banco

Los bancos asociados a un expediente podrán ser dadas de alta y modificados sus datos desde la gestión del expediente.

Un banco asociado a un expediente podrá ser dado de baja siempre y cuando no exista un expediente al que esté asociado.

6.3.1.4 Gestión de prestamos asociados al expediente

El sistema almacenará la información de prestamos asociados a un expediente.

A un expediente de financiación podrá ser asociado cero, uno o varios prestamos.

La información sobre el préstamo asociado al expediente será:

- Tipo de préstamo (Préstamo hipotecario, Préstamo personal, Tarjeta de crédito, Otros, Sobrante)
- Capital pendiente
- Comisión de cancelación
- Total del préstamo
- Cuotas mensuales

Los préstamos asociados a un expediente podrán ser dados de alta y modificados sus datos desde la gestión del expediente.

Un préstamo asociado a un expediente podrá ser dado de baja siempre y cuando no exista un expediente al que esté asociado.

6.3.1.5 Gestión de colaboradores

El sistema mantendrá la información de los colaboradores.

Se podrá consultar todos los colaboradores que existen en el sistema.

La información sobre colaboradores que recogerá el sistema será:

- Nombre
- Apellido1
- Apellido2
- Usuario asociado

Los colaboradores se podrán dar de alta o modificar introduciendo todos los datos anteriormente señalados.

Un colaborador podrá ser asociado a un expediente.

Los colaboradores se podrán eliminar siempre que no tengan un expediente asociado.

No podrán existir dos colaboradores con el mismo usuario.

6.3.1.6 Gestión de comisiones, tarifas y gastos

El mantenimiento de la información de comisiones, tarifas y gastos (alta, baja, modificación) por interfaz de usuario no está dentro del alcance del proyecto. Su mantenimiento se hará manual sobre la base de datos.

Esta información es utilizada en el cálculo de expedientes (en la información calculada recogida en los requisitos de expedientes)

6.3.2 Requisitos Rendimiento

En el caso de las consultas la respuesta del sistema debe ser inferior a 15 segundos.

En el resto de los casos el sistema debe dar respuestas a las peticiones en menos de 5 segundos.

6.3.3 Requisitos Seguridad

Para acceder al sistema es necesario identificarse mediante los credenciales de usuario (nombre de usuario y clave).

Para ello al iniciar la aplicación les solicitará que introduzcan su nombre de usuario y clave, el cual será validado contra la información que tiene el sistema y sólo les proporcionará acceso si coinciden.

El sistema permitirá cambiar la clave del usuario al usuario identificado en el sistema (su propia clave)

El sistema permitirá acceder a los servicios que proporciona solo a colaboradores mediante un usuario (a excepción del usuario administrador, es decir, si se introducen credenciales válidas de usuario, pero no existe un colaborador asociado al usuario, el proceso de acceso dará un error y no permitirá acceder al sistema)

Cada usuario pertenecerá a un grupo de usuarios.

Cada grupo de usuarios tendrá asignados unos permisos.

Cada permiso tendrá un nivel. Los valores del nivel serán ninguno, lectura o lectura / escritura.

Existirán permisos al menos por cada tipo de gestión que realiza el sistema

Para acceder a dicha gestión será requisito que el usuario mediante los permisos de su grupo tenga al menos permiso de nivel de lectura.

El sistema solo mostrará accesibles en el interfaz de usuario aquellas opciones de gestión que cumplan con los requisitos de permisos de acceso indicados.

El sistema permitirá asociar a cada colaborador un único usuario.

El sistema permitirá mantener la gestión (alta / baja / modificación y consulta) de usuarios, grupos y los permisos asociados a los grupos con sus niveles.

No podrá ser eliminado un usuario que tenga asociado un colaborador.

No podrá ser eliminado un grupo que tenga asociado un usuario.

Las comunicaciones entre el navegador y el sistema no serán cifradas.

6.3.4 Requisitos Implantación

El sistema deberá correr tanto en sistemas operativos Windows como Linux.

El navegador soportado será Internet Explorer 6 y 7 y Firefox 2.

El sistema será desarrollado en la plataforma JEE y será una aplicación de gestión empresarial con interfaz web.

El sistema deberá estar basado en tecnologías o productos gratuitos y preferiblemente de código abierto (open source).

El sistema desarrollará un arquitectura reusable para futuras ampliaciones de funcionalidades del sistema u otros sistemas empresariales con interfaz web.

6.3.5 Requisitos Disponibilidad del Sistema

El sistema se diseñará para que este operativo durante el horario de oficina de la empresa.

El sistema será actualizado fuera del horario de oficina (backups y restauraciones de la base de datos, actualizaciones de software)

6.4 Alcance del Sistema

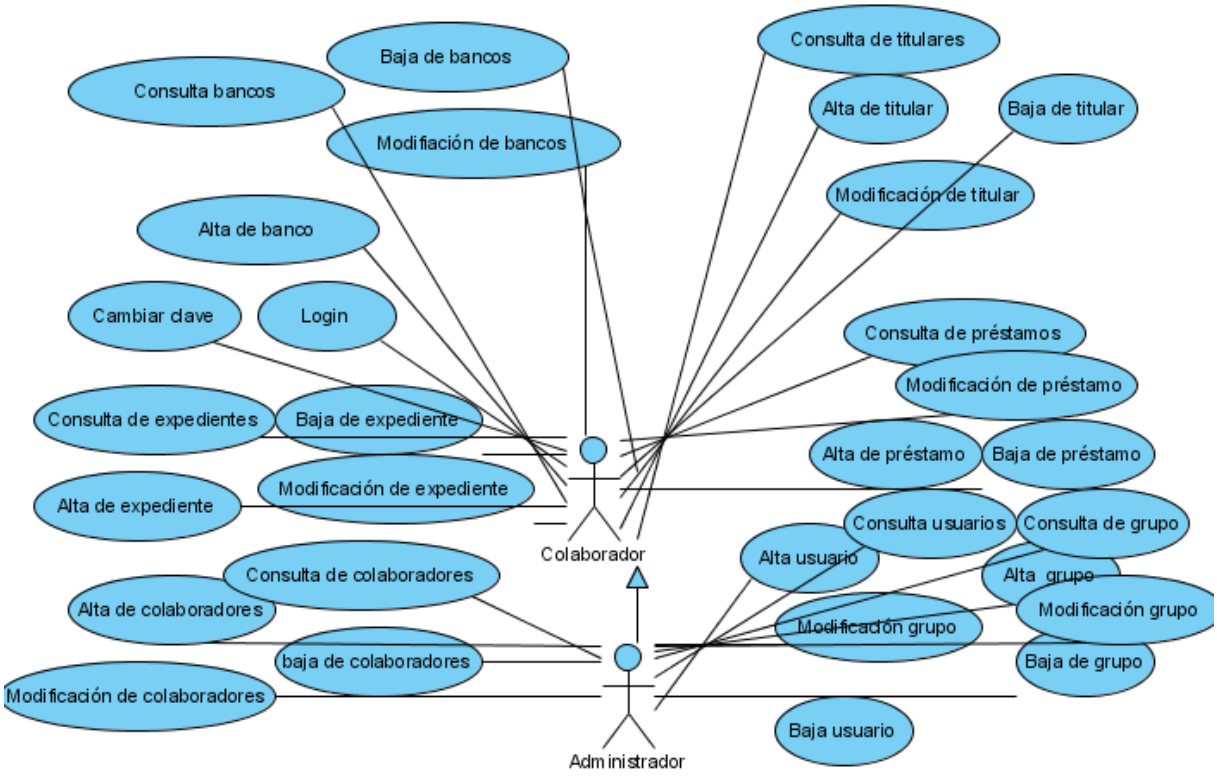
Para describir el alcance del sistema y utilizando la metodologías de UML se describirá el modelo de negocio el cual recoge los casos de uso que se incluyen en el alcance del sistema.

A continuación se especificará cada uno de esos casos de uso, en ellos el alcance del sistema queda definido por las interacciones del usuario u otros sistemas contra el sistema.

Para finalizar se especificará el modelo de las entidades del dominio del sistema, las cuales darán una visión de la información empresarial que maneja el sistema y las relaciones existentes entre las diferentes entidades de esta información empresarial.

6.4.1 Modelo de Negocio

Este modelo describe a grandes rasgos los procesos y entidades principales en el entorno de la aplicación, en términos de casos de uso y unidades de trabajo que intervienen. Se representa mediante el diagrama de casos de uso del modelo negocio



6.4.2 Casos de uso

6.4.2.1 Login

Actores involucrados:	Colaborador
Descripción breve:	El colaborador se autentica en el sistema y accede a la aplicación.
Descripción:	El sistema cuando el colaborador intenta acceder a la aplicación le solicita que introduzca sus credenciales (nombre de usuario y su clave) El sistema comprueba si son correctas las credenciales El sistema le muestra la pantalla de inicio de la aplicación con todas las opciones a las que tiene derecho a acceder habilitadas para su uso (por permisos del grupo del usuario asociado al colaborador).
Precondicionantes	El usuario no está autenticado y conectado “a la aplicación”

Poscondicionantes:	El colaborador está autenticado y “conectado” a la aplicación
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.2 Cambiar clave

Actores involucrados:	Colaborador
Descripción breve:	El colaborador cambia la clave del usuario asociado a él.
Descripción:	El colaborador selecciona la opción de menú de gestión de cambiar clave. El sistema le muestra un formulario para que introduzca su antigua clave y la nueva que desea El colaborador introduce los datos del formulario y solicita que se cambie la clave Si los datos son correctos de la antigua clave el sistema le muestra la pantalla principal Si los datos no son correctos de la antigua clave el sistema le muestra un mensaje indicando el error.
Precondicionantes	El usuario está autenticado y conectado “a la aplicación”
Poscondicionantes:	La clave del usuario asociado al colaborador ha cambiado.
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.3 Consulta de titulares

Actores involucrados:	Colaborador
Descripción breve:	El colaborador consulta los titulares asociados al expediente
Descripción:	El colaborador accede al apartado de titulares del expediente El sistema muestra un listado con todos los titulares asociados al expediente
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de titulares del expediente.
Poscondicionantes:	Ninguna
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.4 Alta de titular

Actores involucrados:	Colaborador
Descripción breve:	El colaborador da de alta un titular asociado a un expediente
Descripción:	El colaborador selecciona la opción de alta de titular en el resultado de la consulta de titulares El sistema le muestra un formulario para introducir los datos para el alta El colaborador rellena al menos los campos obligatorios del formulario de alta y solicita el alta

	El sistema vuelve a la consulta que incluirá el nuevo titular
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de titulares del expediente.
Poscondicionantes:	El sistema ha dado de alta el titular.
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.5 Modificación de titular

Actores involucrados:	Colaborador
Descripción breve:	El colaborador modifica un titular asociado a un expediente
Descripción:	El colaborador selecciona el titular que quiere modificar El colaborador solicita al sistema que lo quiere modificar. El sistema le muestra un formulario para modificar los datos. El colaborador introduce la modificación de datos en el formulario y solicita la modificación al sistema. El sistema vuelve a la consulta que incluirá los datos modificados del titular
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de titulares del expediente.
Poscondicionantes:	El sistema modifica el titular
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.6 Baja de titular

Actores involucrados:	Colaborador
Descripción breve:	El colaborador da de baja un titular asociado a un expediente
Descripción:	El colaborador selecciona el titular que quiere dar de baja. El colaborador solicita al sistema que lo dé de baja, El sistema le pide confirmación Si el colaborador contesta afirmativamente, el sistema lo da de baja. Si no, no lo da de baja. El sistema vuelve a la consulta que incluirá las consecuencias de la acción anterior
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de titulares del expediente.
Poscondicionantes:	El sistema da de baja al titular.
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.7 Consulta de prestamos

Actores involucrados:	Colaborador

Descripción breve:	El colaborador consulta los préstamos asociados al expediente
Descripción:	El colaborador accede al apartado de préstamos del expediente El sistema muestra un listado con todos los préstamos asociados al expediente
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de préstamos del expediente.
Poscondicionantes:	Ninguna
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.8 Alta de préstamo

Actores involucrados:	Colaborador
Descripción breve:	El colaborador da de alta un préstamo asociado a un expediente
Descripción:	El colaborador selecciona la opción de alta de préstamo en el resultado de la consulta de préstamos El sistema le muestra un formulario para introducir los datos para el alta El colaborador rellena al menos los campos obligatorios del formulario de alta y solicita el alta El sistema vuelve a la consulta que incluirá el nuevo préstamo
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login.. El colaborador accede al apartado de préstamos del expediente.
Poscondicionantes:	El sistema ha dado de alta el préstamo.
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.9 Modificación de préstamo

Actores involucrados:	Colaborador
Descripción breve:	El colaborador modifica un préstamo asociado a un expediente
Descripción:	El colaborador selecciona el préstamo que quiere modificar. El colaborador solicita al sistema que lo quiere modificar. El sistema le muestra un formulario para modificar los datos. El colaborador introduce la modificación de datos en el formulario y solicita la modificación al sistema. El sistema vuelve a la consulta que incluirá los datos modificados del préstamo
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de préstamo del expediente.
Poscondicionantes:	El sistema modifica el préstamo
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.10 Baja de préstamo

Actores involucrados:	Colaborador
Descripción breve:	El colaborador da de baja un préstamo asociado a un expediente
Descripción:	El colaborador selecciona el préstamo que quiere dar de baja. El colaborador solicita al sistema que lo dé de baja, El sistema le pide confirmación Si el colaborador contesta afirmativamente, el sistema lo da de baja. Si no, no lo da de baja. El sistema vuelve a la consulta que incluirá las consecuencias de la acción anterior
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de préstamos del expediente.
Poscondicionantes:	El sistema da de baja al préstamo.
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.11 Consulta de bancos

Actores involucrados:	Colaborador
Descripción breve:	El colaborador consulta los bancos asociados al expediente
Descripción:	El colaborador accede al apartado de bancos del expediente El sistema muestra un listado con todos los bancos asociados al expediente
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de bancos del expediente.
Poscondicionantes:	Ninguna
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.12 Alta de banco

Actores involucrados:	Colaborador
Descripción breve:	El colaborador da de alta un banco asociado a un expediente
Descripción:	El colaborador selecciona la opción de alta de banco en el resultado de la consulta de banco El sistema le muestra un formulario para introducir los datos para el alta El colaborador rellena al menos los campos obligatorios del formulario de alta y solicita el alta El sistema vuelve a la consulta que incluirá el nuevo banco
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de bancos del expediente.
Poscondicionantes:	El sistema ha dado de alta el banco.

Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.13 Modificación de banco

Actores involucrados:	Colaborador
Descripción breve:	El colaborador modifica un banco asociado a un expediente
Descripción:	El colaborador selecciona el banco que quiere modificar. El colaborador solicita al sistema que lo quiere modificar. El sistema le muestra un formulario para modificar los datos. El colaborador introduce la modificación de datos en el formulario y solicita la modificación al sistema. El sistema vuelve a la consulta que incluirá los datos modificados del banco
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de banco del expediente.
Poscondicionantes:	El sistema modifica el banco
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.14 Baja de banco

Actores involucrados:	Colaborador
Descripción breve:	El colaborador da de baja un banco asociado a un expediente
Descripción:	El colaborador selecciona el banco que quiere dar de baja. El colaborador solicita al sistema que lo dé de baja, El sistema le pide confirmación Si el colaborador contesta afirmativamente, el sistema lo da de baja. Si no, no lo da de baja. El sistema vuelve a la consulta que incluirá las consecuencias de la acción anterior
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador accede al apartado de bancos del expediente.
Poscondicionantes:	El sistema da de baja al banco.
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.15 Consulta de expedientes

Actores involucrados:	Colaborador
Descripción breve:	El colaborador consulta expedientes existentes en el sistema.
Descripción:	El colaborador selecciona la opción de menú de gestión de expedientes. El sistema muestra un listado con todos los expedientes existentes en el sistema.
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login

Poscondicionantes:	Ninguna
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.16 Alta de expediente

Actores involucrados:	Colaborador
Descripción breve:	El colaborador da de alta un expediente
Descripción:	El colaborador selecciona la opción de alta de expediente en el resultado de la consulta de expedientes. El sistema le muestra un formulario para introducir los datos para el alta El colaborador rellena al menos los campos obligatorios del formulario de alta y solicita el alta El sistema vuelve a la consulta que incluirá los datos dados de alta del expediente.
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login. El colaborador procede del caso de uso de consulta.
Poscondicionantes:	El sistema ha dado de alta el expediente
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.17 Modificación de expediente

Actores involucrados:	Colaborador
Descripción breve:	El colaborador modifica un expediente
Descripción:	El colaborador selecciona el expediente que quiere modificar El colaborador solicita al sistema que lo quiere modificar. El sistema le muestra un formulario para modificar los datos. El colaborador introduce la modificación de datos en el formulario y solicita la modificación al sistema. El sistema vuelve a la consulta que incluirá los datos dados modificados del expediente.
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login El colaborador procede del caso de uso de consulta.
Poscondicionantes:	El sistema ha modificado el expediente
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.18 Baja de expediente

Actores involucrados:	Colaborador
Descripción breve:	El colaborador da de baja un expediente
Descripción:	El colaborador selecciona el expediente que quiere dar de baja. El colaborador

	solicita al sistema que lo dé de baja, El sistema le pide confirmación Si el colaborador contesta afirmativamente, el sistema lo da de baja. Si no, no lo da de baja. El sistema vuelve a la consulta que incluirá las consecuencias de la acción anterior
Precondicionantes	El colaborador ha iniciado sesión en el sistema mediante el caso de uso login El colaborador procede del caso de uso de consulta.
Poscondicionantes:	El sistema ha dado de baja el expediente y todas las entidades asociadas (prestamos, titulares y bancos)
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.19 Consulta de colaboradores

Actores involucrados:	Administrador
Descripción breve:	El administrador consulta los colaboradores existentes en el sistema.
Descripción:	El administrador selecciona la opción de menú de gestión de colaboradores . El sistema muestra un listado con todos los colaboradores existentes en el sistema.
Precondicionantes	El administrador ha iniciado sesión en el sistema mediante el caso de uso login
Poscondicionantes:	Ninguna
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.20 Alta de colaborador

Actores involucrados:	Administrador
Descripción breve:	El administrador da de alta un colaborador
Descripción:	El administrador selecciona la opción de alta de colaborador en el resultado de la consulta de expedientes. El sistema le muestra un formulario para introducir los datos para el alta El administrador rellena al menos los campos obligatorios del formulario de alta y solicita el alta El sistema vuelve a la consulta que incluirá los datos dados de alta del colaborador.
Precondicionantes	El administrador ha iniciado sesión en el sistema mediante el caso de uso login. El administrador procede del caso de uso de consulta.
Poscondicionantes:	El sistema ha dado de alta el colaborador
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.21 Modificación de colaborador

Actores involucrados:	Administrador

Descripción breve:	El administrador modifica un colaborador
Descripción:	El administrador selecciona el colaborador que quiere modificar El administrador solicita al sistema que lo quiere modificar. El sistema le muestra un formulario para modificar los datos. El administrador introduce la modificación de datos en el formulario y solicita la modificación al sistema. El sistema vuelve a la consulta que incluirá los datos dados modificados del colaborador .
Precondicionantes	El administrador ha iniciado sesión en el sistema mediante el caso de uso login El administrador procede del caso de uso de consulta.
Poscondicionantes:	El sistema ha modificado el colaborador

6.4.2.22 Baja de colaborador

Actores involucrados:	Administrador
Descripción breve:	El administrador da de baja un colaborador
Descripción:	El administrador selecciona el colaborador que quiere dar de baja. El colaborador solicita al sistema que lo dé de baja, El sistema le pide confirmación Si el administrador contesta afirmativamente, el sistema lo da de baja. Si no, no lo da de baja. El sistema vuelve a la consulta que incluirá las consecuencias de la acción anterior
Precondicionantes	El administrador ha iniciado sesión en el sistema mediante el caso de uso login El administrador procede del caso de uso de consulta.
Poscondicionantes:	El sistema ha dado de baja el colaborador
Excepciones:	Si se produce un error el sistema mostrará un mensaje indicando que error se ha producido.

6.4.2.23 Consulta de usuarios

Por límites de espacio de la memoria no se pone todo el texto de este caso de uso. Sigue las mismas pautas que el caso de uso equivalente de colaboradores.

6.4.2.24 Alta de usuarios

Por límites de espacio de la memoria no se pone todo el texto de este caso de uso. Sigue las mismas pautas que el caso de uso equivalente de colaboradores.

6.4.2.25 Modificación de usuarios

Por límites de espacio de la memoria no se pone todo el texto de este caso de uso. Sigue las mismas pautas que el caso de uso equivalente de colaboradores.

6.4.2.26 Baja de usuarios

Por límites de espacio de la memoria no se pone todo el texto de este caso de uso. Sigue las mismas pautas que el caso de uso equivalente de colaboradores.

6.4.2.27 Consulta de grupos

Por límites de espacio de la memoria no se pone todo el texto de este caso de uso. Sigue las mismas pautas que el caso de uso equivalente de colaboradores.

6.4.2.28 Alta de grupos

Por límites de espacio de la memoria no se pone todo el texto de este caso de uso. Sigue las mismas pautas que el caso de uso equivalente de colaboradores.

6.4.2.29 Modificación de grupos

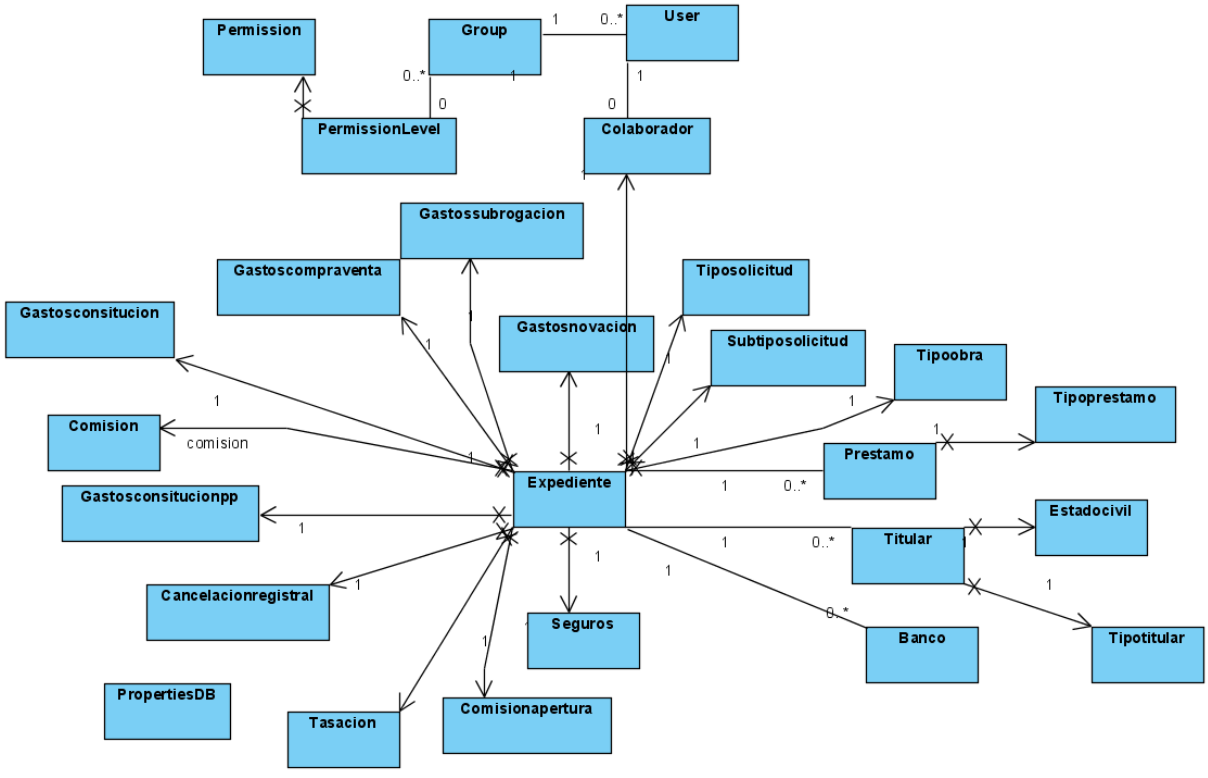
Por límites de espacio de la memoria no se pone todo el texto de este caso de uso. Sigue las mismas pautas que el caso de uso equivalente de colaboradores.

6.4.2.30 Baja de grupos

Por límites de espacio de la memoria no se pone todo el texto de este caso de uso. Sigue las mismas pautas que el caso de uso equivalente de colaboradores.

6.4.3 Modelo de dominio

El modelo de dominio recoge los tipos de clases, más importantes de la aplicación que representan las entidades de información manejadas por el modelo de negocio.



6.5 Plan de migración y carga inicial de datos

Para que el sistema funcione correctamente con los expedientes, será necesario que se cargue inicialmente la información de tarifas, comisiones, gastos y colaboradores.

Si el sistema no ha sido inicializado (lo comprobará) el sistema cargará en la base de datos la información del usuario administrador con usuario *admin* y clave *adminchangeme*, creará su grupo de *administration* y le asignará todos permisos para todas las operaciones posibles del sistema.

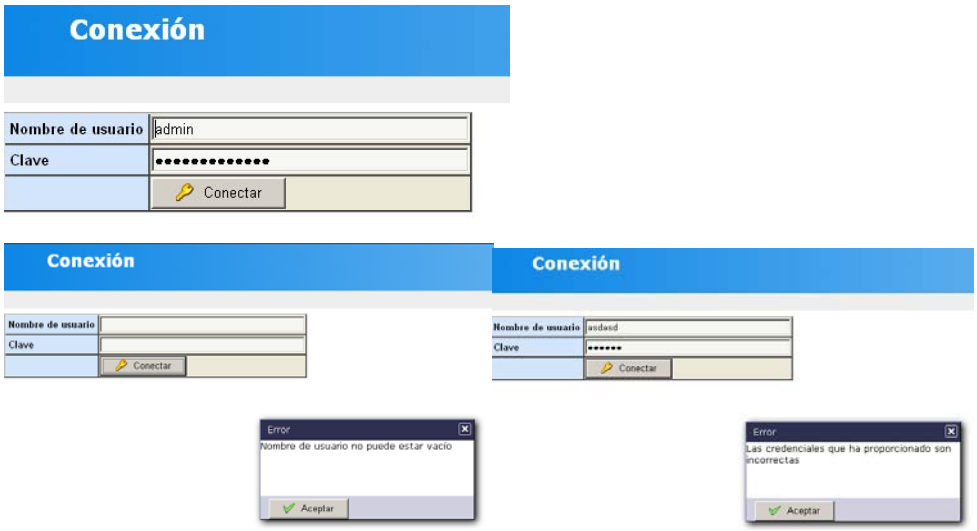
6.6 Descripción de interfaz con otros sistemas

No existen interfaces con otros sistemas

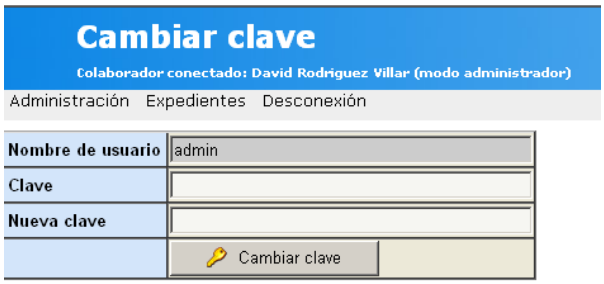
6.7 Especificación de interfaz de usuario

A continuación se presentan los bocetos del interfaz de las páginas que formarán el interfaz con el que el usuario (colaborador) interactuara con el sistema.

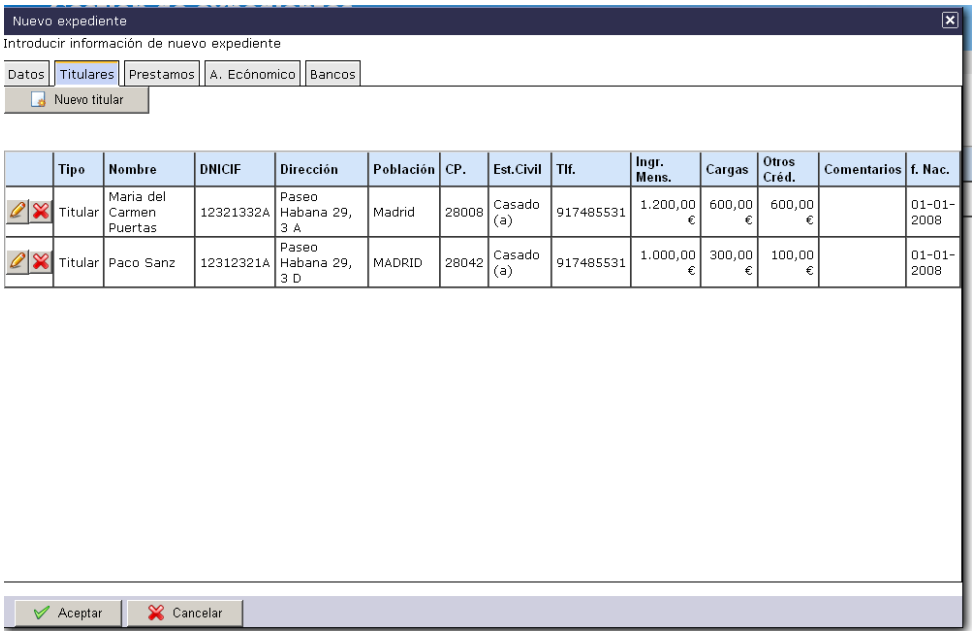
6.7.1 Login



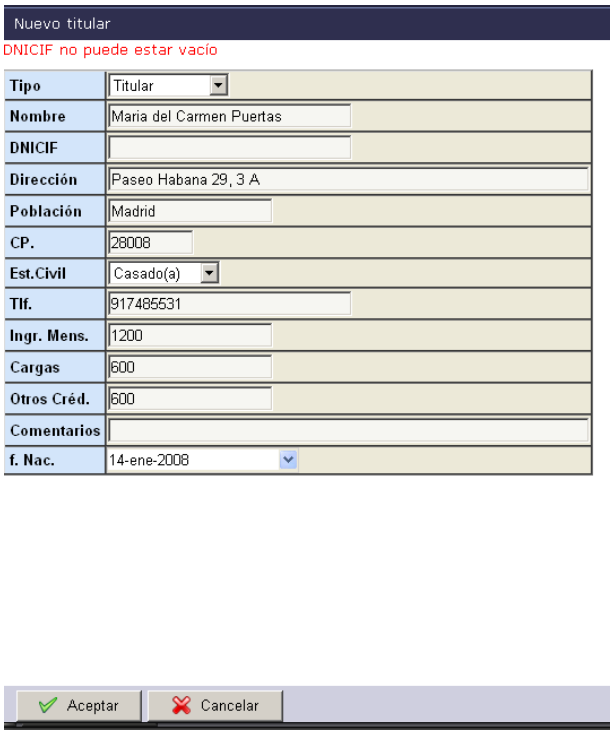
6.7.2 Cambiar clave



6.7.3 Consulta de titulares



6.7.4 Alta de titular



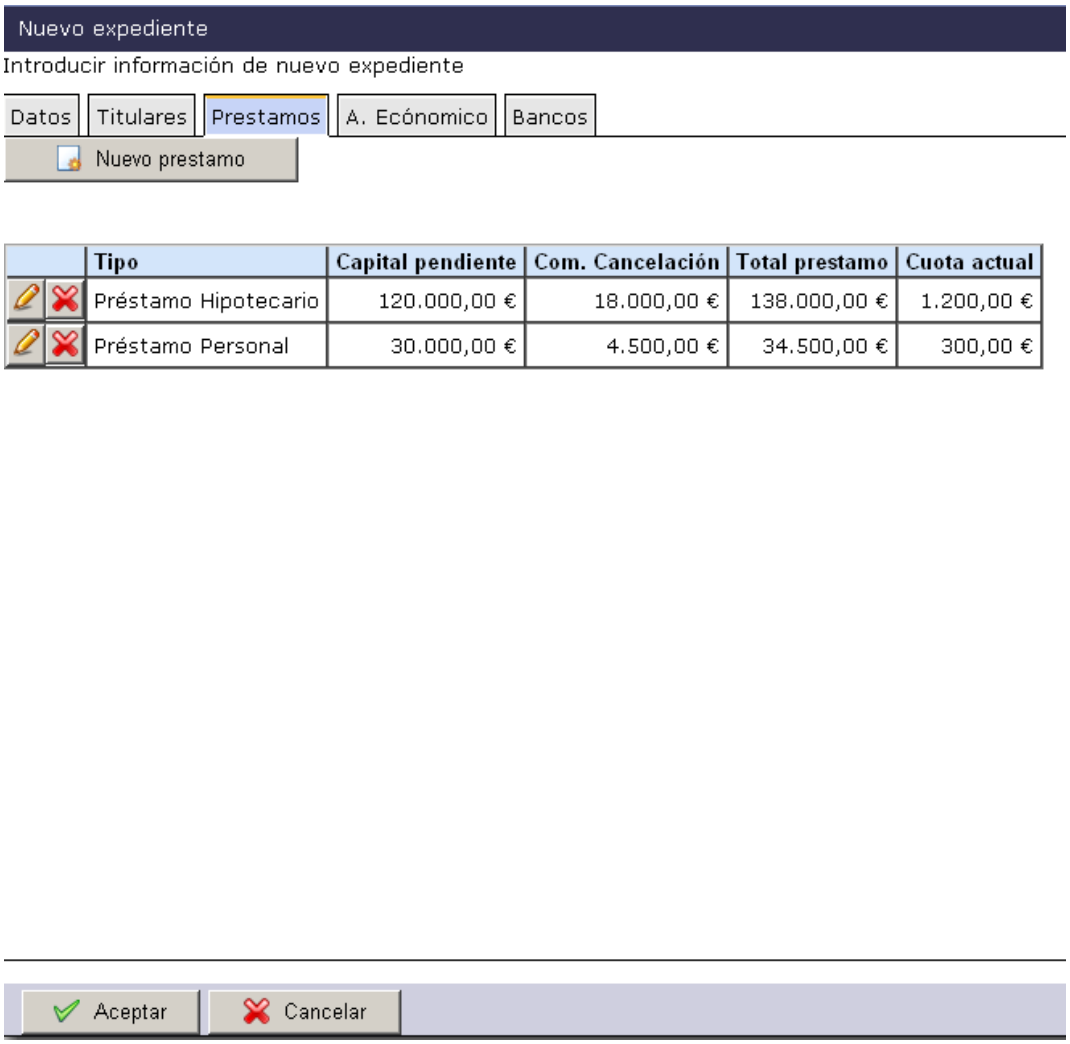
6.7.5 Modificación de titular

Igual que el alta pero modificando el título de la ventana a modificación.

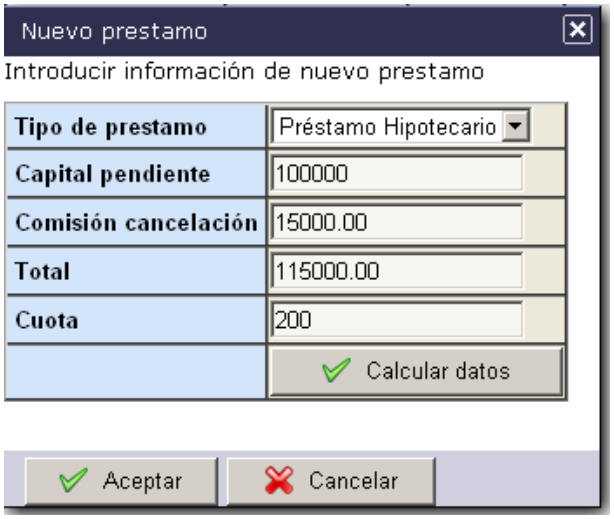
6.7.6 Baja de titular

Una ventana de mensaje sobre la de consulta solicitando la confirmación con sí o no.

6.7.7 Consulta de prestamos



6.7.8 Alta de préstamo



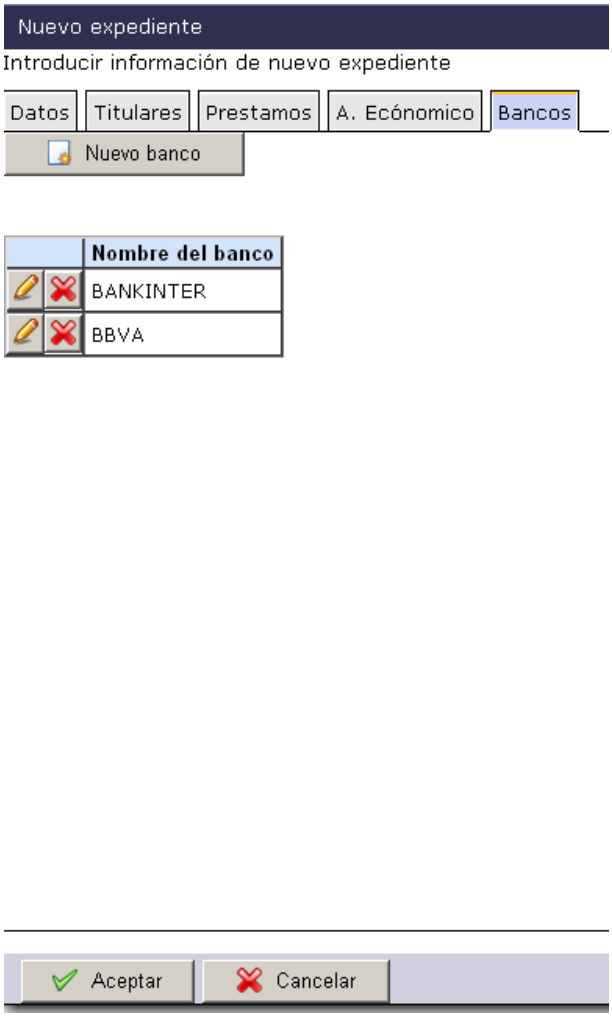
6.7.9 Modificación de préstamo

Igual que el alta pero modificando el título de la ventana a modificación.

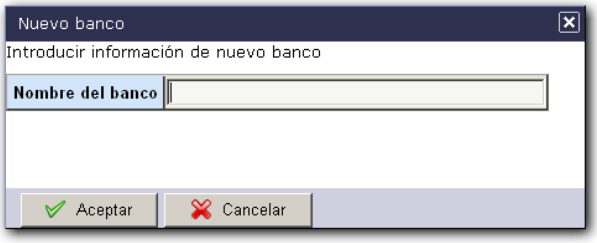
6.7.10 Baja de préstamo

Una ventana de mensaje sobre la de consulta solicitando la confirmación con sí o no.

6.7.11 Consulta de bancos



6.7.12 Alta de banco



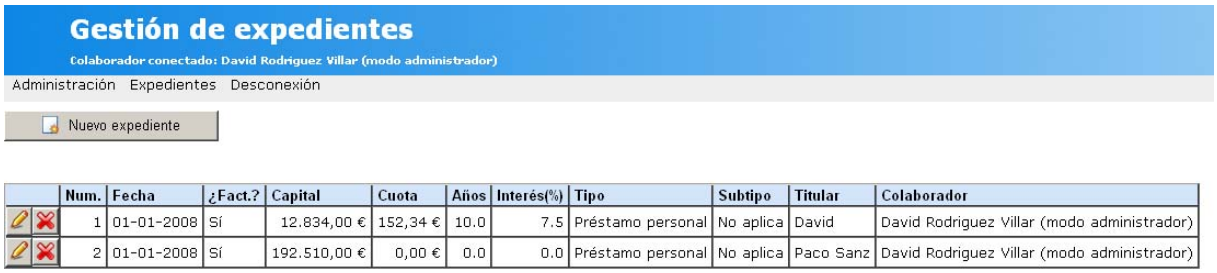
6.7.13 Modificación de banco

Igual que el alta pero modificando el título de la ventana a modificación.

6.7.14 Baja de banco

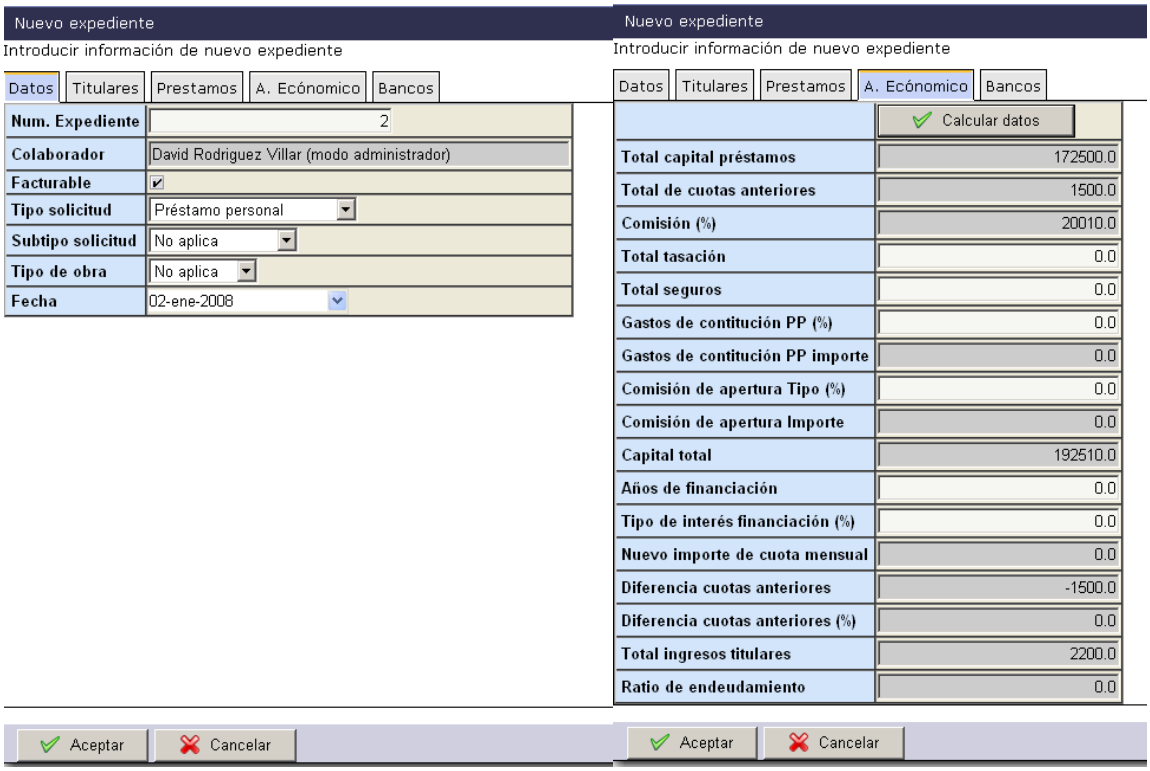
Una ventana de mensaje sobre la de consulta solicitando la confirmación con sí o no.

6.7.15 Consulta de expedientes



	Num.	Fecha	¿Fact.?	Capital	Cuota	Años	Interés(%)	Tipo	Subtipo	Titular	Colaborador
	1	01-01-2008	Sí	12.834,00 €	152,34 €	10.0	7.5	Préstamo personal	No aplica	David	David Rodriguez Villar (modo administrador)
	2	01-01-2008	Sí	192.510,00 €	0,00 €	0.0	0.0	Préstamo personal	No aplica	Paco Sanz	David Rodriguez Villar (modo administrador)

6.7.16 Alta de expediente



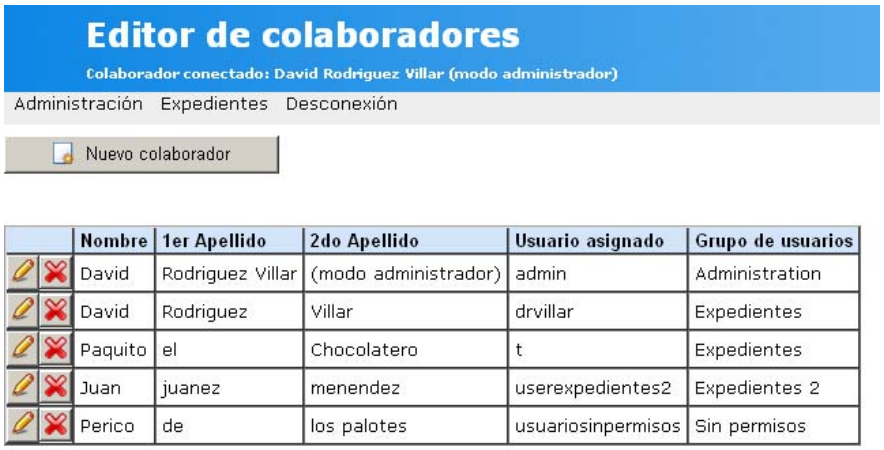
6.7.17 Modificación de expediente

Igual que el alta pero modificando el título de la ventana a modificación.

6.7.18 Baja de expediente

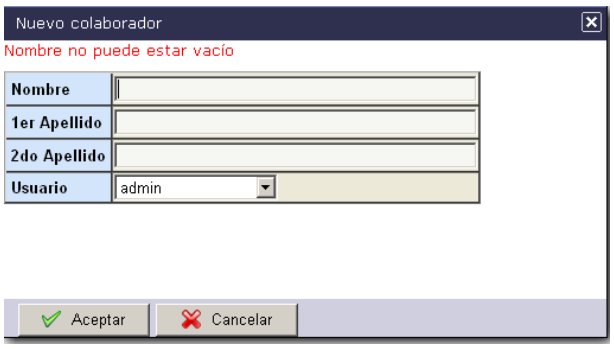
Una ventana de mensaje sobre la de consulta solicitando la confirmación con sí o no.

6.7.19 Consulta de colaboradores



	Nombre	1er Apellido	2do Apellido	Usuario asignado	Grupo de usuarios
	David	Rodriguez Villar	(modo administrador)	admin	Administration
	David	Rodriguez	Villar	drvillar	Expedientes
	Paquito	el	Chocolatero	t	Expedientes
	Juan	juanez	menendez	userexpedientes2	Expedientes 2
	Perico	de	los palotes	usuariosinpermisos	Sin permisos

6.7.20 Alta de colaborador



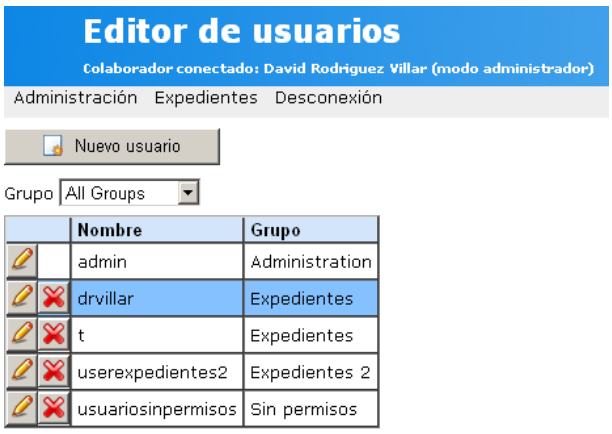
6.7.21 Modificación de colaborador

Igual que el alta pero modificando el título de la ventana a modificación.

6.7.22 Baja de colaborador

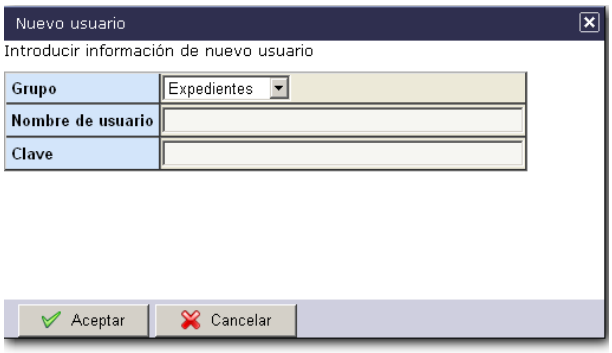
Una ventana de mensaje sobre la de consulta solicitando la confirmación con sí o no.

6.7.23 Consulta de usuarios



	Nombre	Grupo
	admin	Administration
	drvillar	Expedientes
	t	Expedientes
	userexpedientes2	Expedientes 2
	usuariosinpermisos	Sin permisos

6.7.24 Alta de usuarios



6.7.25 Modificación de usuarios

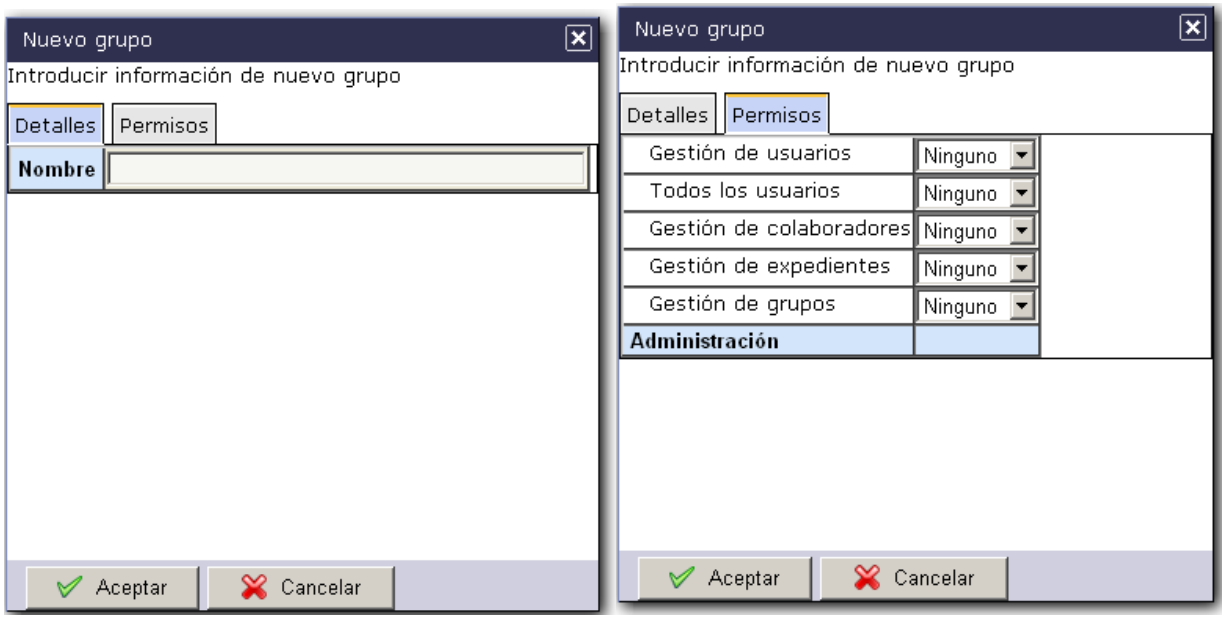
Igual que el alta pero modificando el título de la ventana a modificación.

6.7.26 Baja de usuarios

Una ventana de mensaje sobre la de consulta solicitando la confirmación con sí o no.

6.7.27 Consulta de grupos

6.7.28 Alta de grupos



6.7.29 Modificación de grupos

Igual que el alta pero modificando el título de la ventana a modificación.

6.7.30 Baja de grupos

Una ventana de mensaje sobre la de consulta solicitando la confirmación con sí o no.

7 Diseño del sistema

7.1 Arquitectura de la aplicación

7.1.1 Arquitectura JEE

Sun creó la plataforma de java empresarial (JEE) sobre 1999 (antes conocida por J2EE).

Las razones que empujaron a la creación de la plataforma JEE:

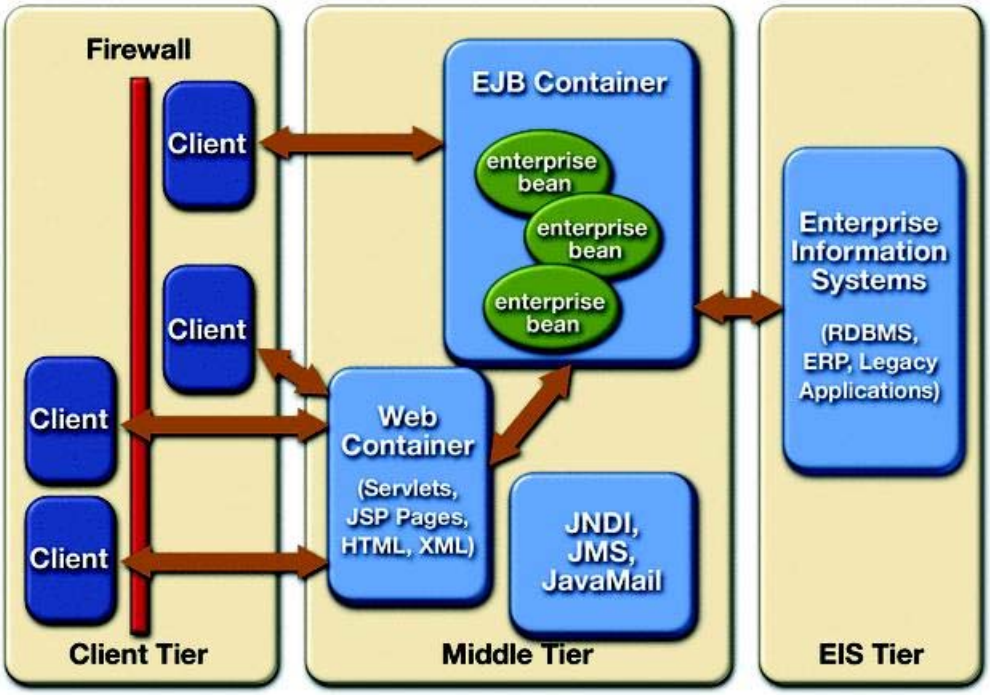
- Programación eficiente. Para conseguir productividad es importante que los equipos de desarrollo tengan una forma estándar de construir múltiples aplicaciones en diversas capas (cliente, servidor web, etc.). En cada capa necesitaremos diversas herramientas, por ejemplo en la capa cliente tenemos applets, aplicaciones Java, etc. En la capa web tenemos servlets, páginas JSP, etc. Con JEE tenemos una tecnología estándar, un único modelo de aplicaciones, que incluye diversas herramientas; en contraposición al desarrollo tradicional con HTML, Javascript en el lado del servidor, CGI, servidor web, etc. que implicaba numerosos modelos para la creación de contenidos dinámicos, con los lógicos inconvenientes para la integración.
- Extensibilidad frente a la demanda del negocio. En un contexto de crecimiento de número de usuarios es precisa la gestión de recursos, como conexiones a bases de datos, transacciones o balanceo de carga. Además los equipos de desarrollo deben aplicar un estándar que les permita abstraerse de la implementación del servidor, con aplicaciones que puedan ejecutarse en múltiples servidores, desde un simple servidor hasta una arquitectura de alta disponibilidad y balanceo de carga entre diversas máquinas.
- Integración. Los equipos de ingeniera precisan estándares que favorezcan la integración entre diversas capas de software.

El objetivo de crear esta plataforma fue que los equipos de desarrollo se centrasen en desarrollar la programación del modelo de negocio de la aplicación empresarial y que dejasen los detalles tecnológicos a una serie de servicios empresariales que utilizarían desde sus elementos software (mediante la definición de API estándar para dichos servicios y unos contendores que ofreciesen dicho API al software desarrollado y alojado en ellos)

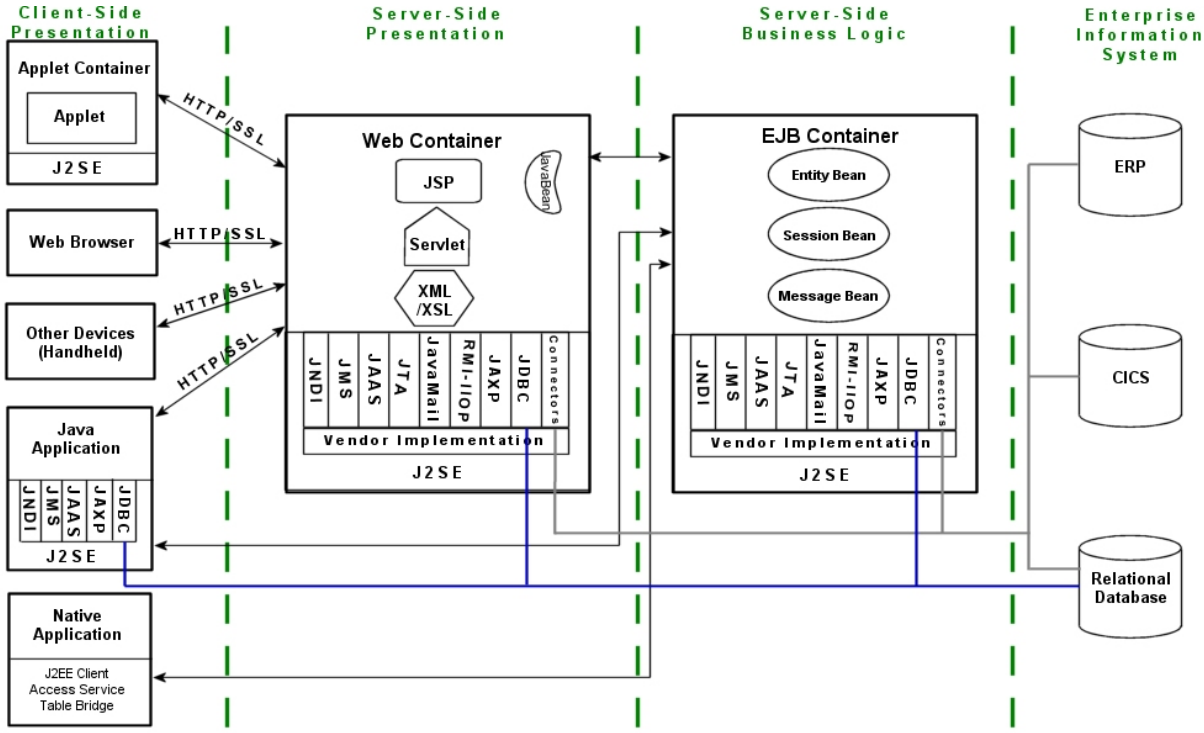
La plataforma JEE implica una forma de implementar y desplegar aplicaciones empresariales. La plataforma se ha abierto a numerosos fabricantes de software para conseguir satisfacer una amplia variedad de requisitos empresariales.

La arquitectura JEE implica un modelo de aplicaciones distribuidas en diversas capas o niveles (tier). La capa cliente admite diversas tipos de clientes (HTML, Applet, aplicaciones Java, etc.). la capa intermedia (middle tier) contiene subcapas (el contenedor web y el contenedor EJB). La tercera capa dentro de esta visión sintética es la de aplicaciones 'backend' como ERP, EIS, bases de datos, etc.

Como se puede ver un concepto clave de la arquitectura es el de contenedor, que dicho de forma genérica no es más que un entorno de ejecución estandarizado que ofrece unos servicios por medio de componentes. Los componentes externos al contenedor tienen una forma estándar de acceder a los servicios de dicho contenedor, con independencia del fabricante, como puede verse en el siguiente gráfico:



La plataforma está definida por el conjunto de API estándar para acceder a los servicios empresariales que proporciona la plataforma (Son estos estándares los que deben cumplir los fabricantes).



La plataforma JEE incluye los siguientes APIs para el acceso a sistemas empresariales:

- **Java Database Connectivity (JDBC)** es el API para acceso a Sistemas Gestores de Bases de Datos Relaciones (SGBDR) desde Java.
- **Java Transaction API (JTA)** es el API para manejo de transacciones a través de sistemas heterogéneos.

- **Java Naming and Directory Interface (JNDI)** es el API para acceso a servicios de nombres y directorios.
- **Java Message Service (JMS)** es el API para el envío y recepción de mensajes por medio de sistemas de mensajería empresarial como IBM MQ Series.
- **JavaMail** es el API para envío y recepción de email.
- **Java IDL** es el API para llamar a servicios CORBA.
- **Enterprise JavaBeans** es un conjunto de APIs que un contenedor distribuido soportará para proporcionar persistencia, llamadas a procedimientos remotos (usando RMI o RMI-IIOP), control de la concurrencia y control de acceso a objetos distribuidos.
- **Java API for XML Web Services (JAXP-WS)** API responsable de dar soporte a los servicios Web.

7.1.2 Patrones participantes en la arquitectura de la aplicación

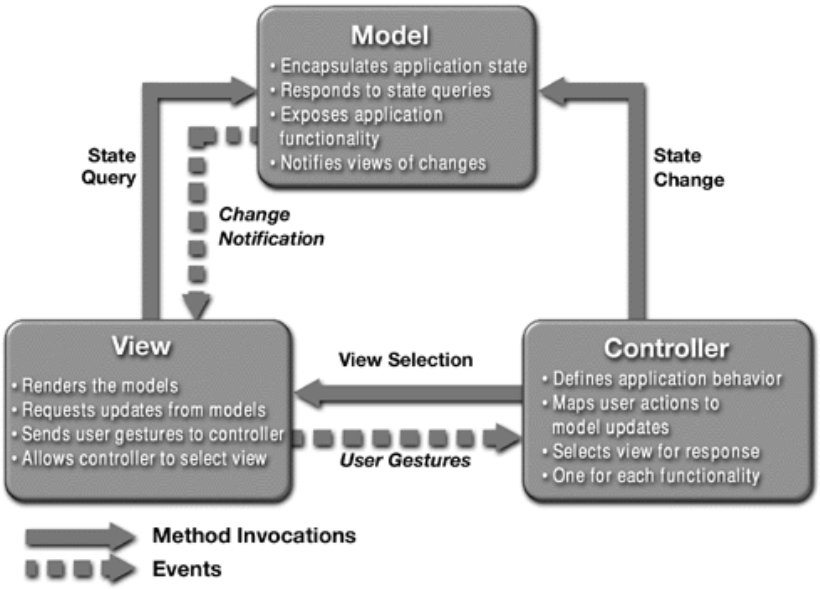
7.1.2.1 Modelo Vista Controlador (MVC)

El patrón arquitectónico de Modelo Vista Controlador es una aproximación arquitectónica para las aplicaciones interactivas. Divide la funcionalidad entre los objetos implicados en el mantenimiento y presentación de información para minimizar el grado de acoplamiento entre los objetos. Este patrón mapea las tradicionales actividades de aplicación (entrada, procesamiento y salida) a un modelo gráfico de interacción con el usuario.

The MVC architecture divides applications into three layers--model, view, and controller--and decouples their respective responsibilities. Each layer handles specific tasks and has specific responsibilities to the other areas.

El MVC divide una aplicación interactiva con GUI en 3 capas (modelo, vista y controlador) y desacopla sus respectivas responsabilidades. Cada capa maneja tareas específicas y tiene responsabilidades específicas que proporcionar a las otras capas:

- **Modelo** (Model): El modelo representa la información de negocio y la lógica de negocio o las operaciones que gobiernan el acceso y modificación de la información del negocio. A menudo el modelo sirve como una aproximación software a las funcionalidades del mundo real. El modelo notifica a las vistas cuando se han producido cambios y proporciona a la vista los medios necesarios para que pregunte al modelo sobre su estado. También proporciona al controlador los medios para acceder a las funcionalidades de la aplicación encapsuladas en el modelo.
- **Vista** (View): La vista dibuja los contenidos del modelo. Accede a la información del modelo y especifica como debe ser presentada. Actualiza la información presentada al usuario cuando el modelo cambia. La vista también redirige las entradas del usuario al controlador.
- **Controlador** (Controller): Un controlador define el comportamiento de la aplicación. Despacha las peticiones del usuario y selección las vistas para la presentación. Interpreta las entradas del usuario y las convierte en las acciones que deben ser desencadenadas sobre el modelo. En un GUI de cliente pesado, las entradas del usuario incluyen pulsaciones de botones y selecciones de menú. En aplicación web, las entradas se transforman en peticiones http get o post a la capa web. Un controlador selecciona la siguiente vista que se mostrará basado en las interaccione y la salida de las operaciones del modelo. Una aplicación normalmente tiene un controlador por cada tipo de funcionalidad relacionada. Algunas aplicaciones usan un controlador por cada tupo de cliente ya que la interacción con la vista varía en función de los tipos de clientes.



Las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación del modelo de los componentes vista y del controlador permite tener múltiples vistas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios.

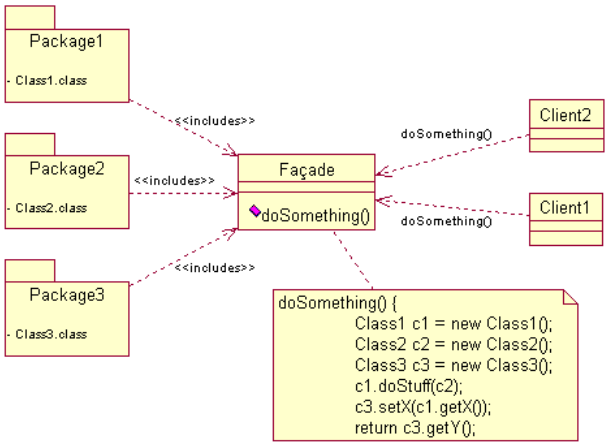
Por lo tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario.

Algunos de sus principales beneficios son:

- Menor acoplamiento
  - Desacopla las vistas de los modelos
  - Desacopla los modelos de la forma en que se muestran e ingresan los datos
- Mayor cohesión
  - Cada elemento del patrón esta altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio)
- Las vistas proveen mayor flexibilidad y agilidad
  - Se puede crear múltiples vistas de un modelo
  - Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente
  - Las vistas pueden anidarse
  - Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual
  - Se puede sincronizar las vistas
  - Las vistas pueden concentrarse en diferentes aspectos del modelo
- Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales
  - Una vista para cada dispositivo que puede varias según sus capacidades
  - Una vista para la Web y otra para aplicaciones de escritorio
- Más claridad de diseño
  - Facilita el mantenimiento
  - Mayor escalabilidad

7.1.2.2 Facade

Este patrón provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.



En este caso, las clases Java comunes (POJO) o EJB encapsulan lógica y datos de negocio, exponiendo sus interfaces y la complejidad de los servicios (puede que sean distribuidos) a la capa cliente.

El objetivo de este patrón es:

- Proporcionar a los clientes un interfaz sencillo que oculte todas las interacciones complejas entre los componentes de negocio.
- Ocultar al cliente las interacciones y las interdependencias entre los componentes de negocio, permitiendo de esta forma el aumento de flexibilidad y evitar que los cambios en los objetos de negocio repercutan en errores en la vista.
- Evitar la exposición directa de los objetos de negocios a los clientes, para mantener el acoplamiento entre las dos capas al mínimo.
- Centralizar los casos de uso en métodos centralizados sobre una clase.

Aplicándolo conseguiremos que una clase java común encapsule a través de su interfaz la complejidad de las interacciones entre los objetos de negocio participantes en la resolución de la regla de negocio.

El objeto façade maneja los objetos de negocio y proporciona un servicio de acceso uniforme a los clientes. Es decir, el cliente (controlador en nuestro caso) solo interactuara con el interfaz del facade y con colecciones o Value's Object que sean entradas y salidas del facade.

Resumiendo, con Façade se obtiene:

- Introducir una capa extra entre el Modelo y el Controlador: Agregando esta capa, se puede modelar los casos de uso de una forma centralizada, logrando también centralizar todo el acoplamiento a esta capa intermedia.
- Expone un interfaz uniforme para el acceso al Modelo.
- Cuando se utiliza con EJB, reduce el número de invocaciones remotas, aumentando la escalabilidad.
- Simplifica la gestión de transacciones y el mantenimiento de los casos de uso.

7.1.2.3 Data Access Objet (DAO)

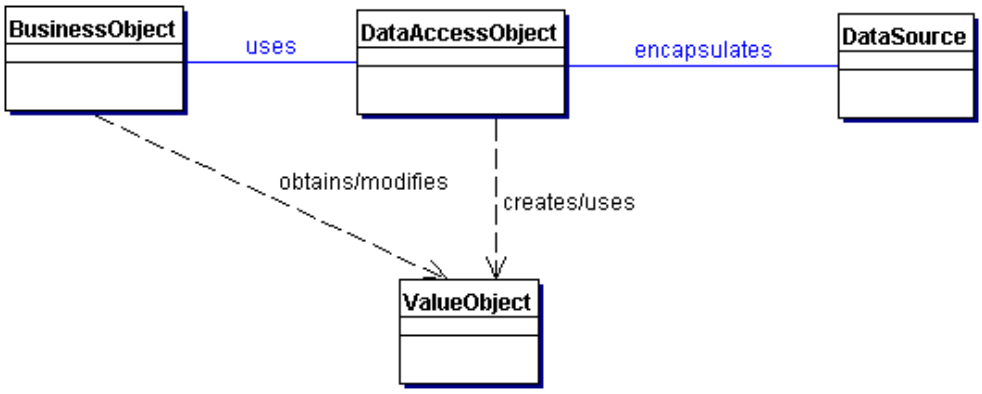
Cuando empezamos el desarrollo de una aplicación, puede llevarnos a tomar la decisión de acceder a datos a partir de XML's, Web Services, o cualquier otro sistema existente como por ejemplo una BD.

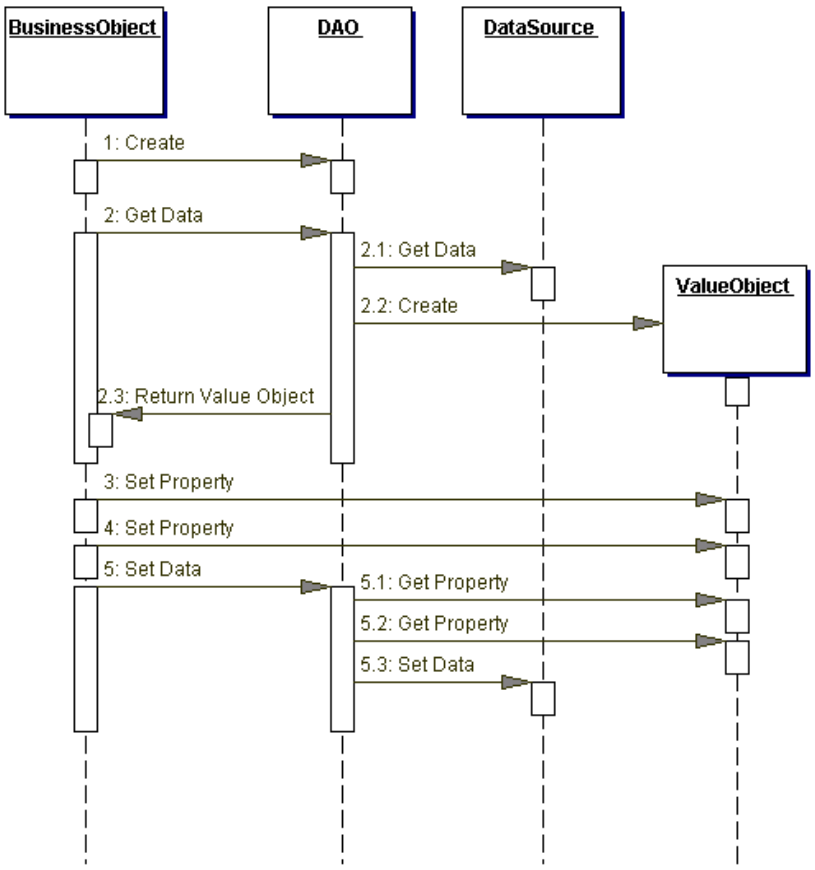
En una fase posterior del desarrollo puede darse el caso en que esto tenga que cambiar. Imaginemos el caso en que se decide utilizar SOAP como sistema de acceso a datos. Conforme la aplicación va creciendo y tiene que dimensionarse podemos experimentar problemas de rendimiento o una mala eficiencia del formato. Por ello deberíamos cambiar toda la capa de acceso a datos.

Dependiendo de cómo esté diseñada la arquitectura de nuestra aplicación este cambio podría tener un gran impacto y afectar no sólo al acceso a datos. Siguiendo con el ejemplo anterior si la lógica de nuestra aplicación utiliza objetos específicos arrastrados del modelo de datos, el cambio afectaría a dos capas.

Un ejemplo de esta problemática sería el cambio de motor de base de datos. Si inicialmente optamos por trabajar con una base de datos mysql pero después por motivos de rendimiento decidimos migrar a Oracle el cambio debería afectar sólo a la capa de acceso a datos y no a la lógica de la aplicación. Si en la aplicación he arrastrado JDBC a todas las capas, la migración será costosa ya que deberé buscarlo por todas las capas.

Utilizar una capa de objetos Data Access Object (DAO) nos será muy útil para abstraer y encapsular todos los accesos a la fuente de datos, logrando así desacoplar la lógica de negocio de la lógica de acceso a datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.





Este patrón nos será muy útil ya que:

• Permite la transparencia: Los objetos de negocio puede utilizar la fuente de datos sin conocer los detalles específicos de su implementación. El acceso es transparente porque los detalles de la implementación se ocultan dentro del DAO.

• Permite una migración más fácil: Una capa de DAOs hace más fácil que una aplicación pueda migrar a una implementación de base de datos diferente. Los objetos de negocio no conocen la implementación de datos subyacente, la migración implica cambios sólo en la capa DAO. Además, si se emplea la estrategia de factorías, es posible proporcionar una implementación de factorías concretas por cada implementación del almacenamiento subyacente. En este caso, la migración a un almacenamiento diferente significa proporcionar a la aplicación una nueva implementación de la factoría.

• Reduce la complejidad del código de los objetos de negocio: Como los DAOs manejan todas las complejidades del acceso a los datos, se simplifica el código de los objetos de negocio y de otros clientes que utilizan los DAOs. Todo el código relacionado con la implementación (como las sentencias SQL) están dentro del DAO y no en el objeto de negocio. Esto mejora la lectura del código y la productividad del desarrollo.

• Centraliza todos los accesos a datos en un capa independiente: Como todas las operaciones de acceso a los datos se ha delegado en los DAOs, esto se puede ver como una capa que aísla el resto de la aplicación de la implementación de acceso a los datos. Esta centralización hace que la aplicación sea más sencilla de mantener y de manejar.

7.1.2.4 Dependency Injection (DI)

Si intentamos buscar información acerca de *Inversion of Control*, nos encontraremos con varios artículos que nos llevarán al desconcierto. Algunos dicen que I*nversion of Control* es un patrón de diseño y que es un sinónimo de *Dependency Injection*. Otros lo muestran como un concepto, e indican que *Dependency Injection* es una forma de llevar a cabo dicho concepto. Tal y como lo concebimos, *Inversion of Control* e *Dependency Injection*, si bien están relacionados, no son equivalentes.

Desde nuestro punto de vista, *Inversion of Control* es un concepto mas amplio, como un criterio de buen diseño, que nos permite programar contra interfaces, para desacoplarnos de lógica especifica. Por otro lado *Dependency Injection* es un patrón de diseño desarrollado bajo el criterio de *Inversion of Control*.

La idea de *Dependency Injection* es proveer a los objetos de nuestro sistema otros objetos, denominados colaboradores, que les permiten llevar a cabo su finalidad. El patrón es parecido al *Builder*, pero con un objetivo diferente, proveer desacoplamiento entre los objetos de nuestro sistema.

Generalmente cuando utilizamos un objeto este no trabaja solo sino que lo hace comunicándose con otros objetos. Estos objetos (dependencias) son pasados como argumentos en constructores o asignados a propiedades (inyectados) al objeto por el contenedor.

Cabe destacar la importancia de la inyección de dependencias: El código se vuelve más claro y menos acoplado cuando los objetos no crean sus dependencias (no necesitan saber donde están localizados ni a que clase pertenecen).

Junto con la declaración del bean podemos incluir inyección de dependencias. Para lograr esto, se utilizará las capacidades de *Dependency Injection* de *Spring Framework*. De esta forma podemos seguir un paradigma de programación dirigida por interfaces (siendo muy sencillo cambiar la implementación de dichos interfaces mediante los archivos de configuración de *Spring*).

7.1.3 Tecnologías y Frameworks integrados en la arquitectura

7.1.3.1 Echo2

Echo2 es un Framework para construir aplicaciones Web que se aproximan a las capacidades de las aplicaciones de escritorio (clientes pesados). El interfaz de usuario se desarrolla usando un API orientado a componentes y notificaciones de interacciones basadas en eventos, eliminando la necesidad de tratar con “la naturaleza basada en páginas” de los navegadores y facilitando la implantación de un arquitectura de interfaz de usuario basada en el patrón MVC.

El API es muy similar al que se usa en la programación de interfaces de usuario basados en Swing (para clientes pesados en Java).

Las razones que han llevado a su inclusión en la arquitectura de la aplicación son:

- La arquitectura que se está desarrollando está pensada para interfaces de usuario de aplicaciones empresariales que tienen su interfaz en la capa web (no son aplicaciones web, sino aplicaciones empresariales con interfaz web).
- Es una tecnología open source.
- Es una tecnología puntera (por ejemplo, el GWT de Google en el que están basadas aplicaciones de Google como Gmail sigue conceptos similares a echo2).
- Sus principios arquitectónicos obligan a usar el patrón MVC
- La tecnología es bastante madura.
- Es una tecnología con un grado de mantenibilidad muy alto, tanto por la comunidad que mantiene echo2 como el mantenimiento de los interfaces de usuario desarrollados sobre echo2.
- El desarrollo de interfaces de usuario basados en echo2 es bastante productivo (tiempos de desarrollo más cortos que usando otras tecnologías en la capa web).
- Hace que el paradigma web de petición (request) respuesta (response) sea invisible: El Framework echo2 gestiona por debajo todo el asunto web, utilizando las tecnologías más punteras incluido AJAX.
- El API de estilos es fantástico con una precisión hasta nivel de pixel.
- El rendimiento para estar basado en AJAX es sorprendentemente alto, tanto en el rendimiento global como en el comparado con otras tecnologías punteras de conceptos similares.
- El diseño es completamente extensible, posibilitando el desarrollo de nuevos componentes de forma sencilla.

- Desde mi punto de vista, y teniendo más de 9 años de experiencia en tecnologías web y java (desde 1999), tecnologías como esta son el futuro del desarrollo de aplicaciones empresariales con interfaz web.

7.1.3.2 Hibernate

Trabajar con software orientado a objetos y bases de datos relacionales puede hacernos invertir mucho tiempo en los entornos actuales. *Hibernate* es una herramienta que realiza el mapeo entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos Java. El término utilizado es ORM (Object / Relational Mapping) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

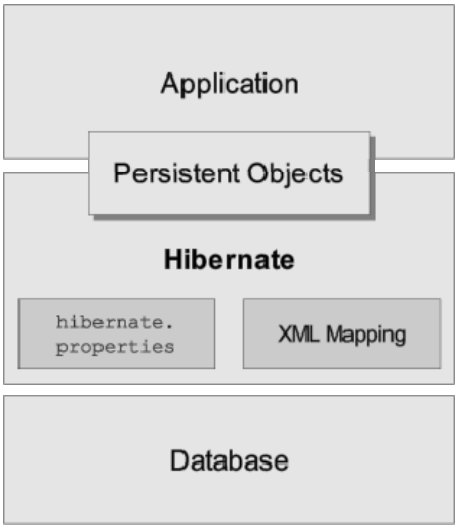
Hibernate no solo realiza esta transformación sino que nos proporciona capacidades para la obtención y almacenamiento de datos de la base de datos que nos reducen el tiempo de desarrollo.

Hibernate funciona asociando a cada tabla de la base de datos un Plain Old Java Object (POJO). Un POJO es similar a un Java Bean, con propiedades accesibles mediante métodos setter y getter. Para poder asociar el POJO a su tabla correspondiente en la base de datos, Hibernate usa los ficheros hbm.xml.

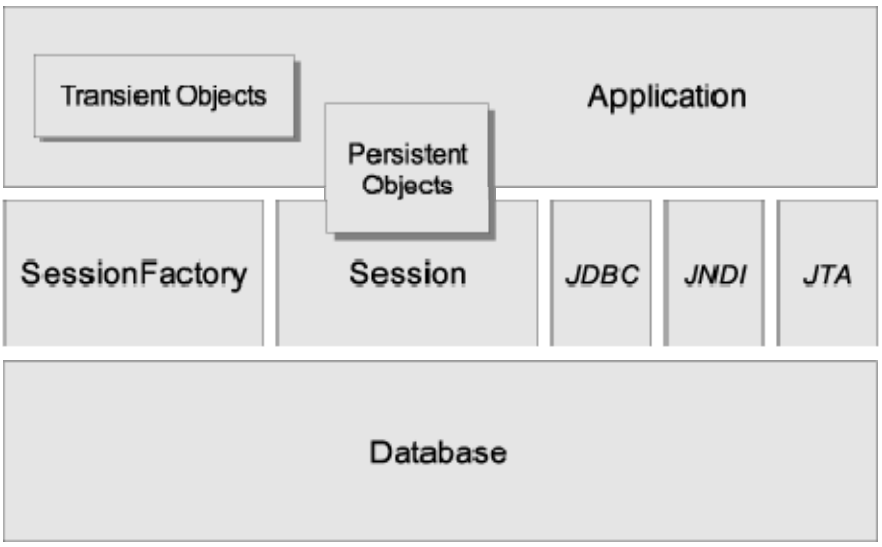
Los parámetros de conexión con la Base de Datos se configuran en el fichero hibernate.cfg.xml. Esto tiene la ventaja de hacer totalmente transparente la base de datos a la que se accede, pudiendo cambiar de base de datos sin necesidad de cambiar una línea de código de nuestra aplicación, simplemente cambiando los ficheros de configuración de Hibernate.

Hibernate tiene un reconocido prestigio en la comunidad de desarrolladores que ha influido en la nueva especificación EJB 3 para definir el API empresarial JEE JPA, el API de persistencia que sustituye a los antiguos entity beans -fracasados y abandonados- por los nuevos entity beans que copian el concepto de *Hibernate*. De hecho actualmente Hibernate es una de las implementaciones de JPA.

En la siguiente imagen aparece una vista de alto nivel de la arquitectura de *Hibernate*:

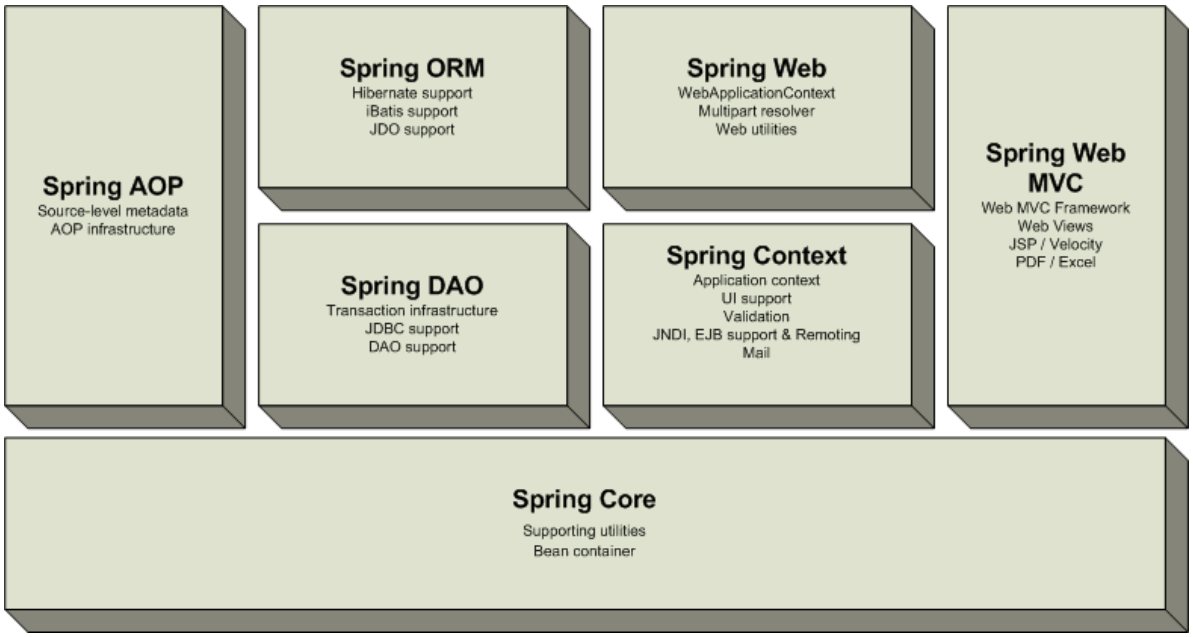


En la siguiente imagen aparece una vista más detallada de la arquitectura de *Hibernate*:



7.1.3.3 Spring Framework

Spring contiene muchas características que le dan una funcionalidad muy amplia; dichas características están organizadas en siete grandes módulos como se puede observar en el diagrama de abajo. Esta sección comenta someramente las características de cada módulo.



El **módulo Core** o "Núcleo" es la parte fundamental del framework ya que provee toda la funcionalidad de Inyección de Dependencias permitiéndote administrar la funcionalidad del contenedor de beans. El concepto básico de este módulo es el BeanFactory, que implementa el patrón de diseño Factory (fábrica) eliminando la necesidad de crear singletons programáticamente permitiéndote desligar la configuración y especificación de las dependencias de tu lógica de programación.

Encima del módulo core se encuentra el **módulo Context** (Contexto), el cual te provee de herramientas para acceder a los beans de una manera elegante, similar a un registro JNDI. El módulo de contexto hereda sus características del módulo de beans y añade soporte para mensajería de texto, como son resource bundles (para internacionalización), propagación de eventos, carga de recursos y creación transparente de contextos por contenedores (como el contenedor de servlets, por ejemplo).

El **módulo DAO** provee una capa de abstracción de JDBC que elimina la necesidad de teclear código JDBC tedioso y redundante así como el parseo de códigos de error específicos de cada proveedor de base de datos. También, el módulo JDBC provee de una manera de administrar transacciones tanto

declarativas como programáticas, no solo para clases que implementen interfaces especiales, pero para todos tus POJOs (por sus siglas en inglés, Viejos y simples objetos java).

El **módulo ORM** provee capas de integración para APIs de mapeo objeto - relacional, incluyendo, JDO, Hibernate e iBatis. Usando el módulo ORM tú puedes usar esos mapeadores en conjunto con otras características que *Spring* ofrece, como la administración de transacciones mencionada con anterioridad.

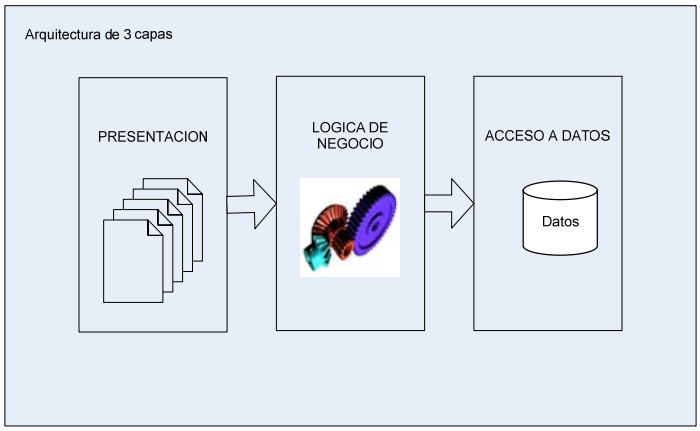
El **módulo AOP** provee una implementación de programación orientada a aspectos compatible con AOP Alliance, permitiéndote definir pointcuts e interceptores de métodos para desacoplar el código de una manera limpia implementando funcionalidad que por lógica y claridad debería estar separada. Usando metadatos a nivel de código fuente se pueden incorporar diversos tipos de información y comportamiento al código, un poco similar a los atributos de .NET

El **módulo Web** provee características básicas de integración orientado a la web, como funcionalidad multipartes (para realizar la carga de archivos), inicialización de contextos mediante servlet listeners y un contexto de aplicación orientado a web. Cuando se usa *Spring* junto con WebWork o Struts, este es el módulo que te permite una integración sencilla.

El **módulo Web MVC** provee de una implementación Modelo - Vista - Controlador para las aplicaciones web. La implementación de *Spring* MVC permite una separación entre código de modelo de dominio y las formas web y permite el uso de otras características de Spring Framework como lo es la validación.

7.1.4 Diseño de la arquitectura

La arquitectura como aplicación empresarial se va a basar en un arquitectura de 3 capas



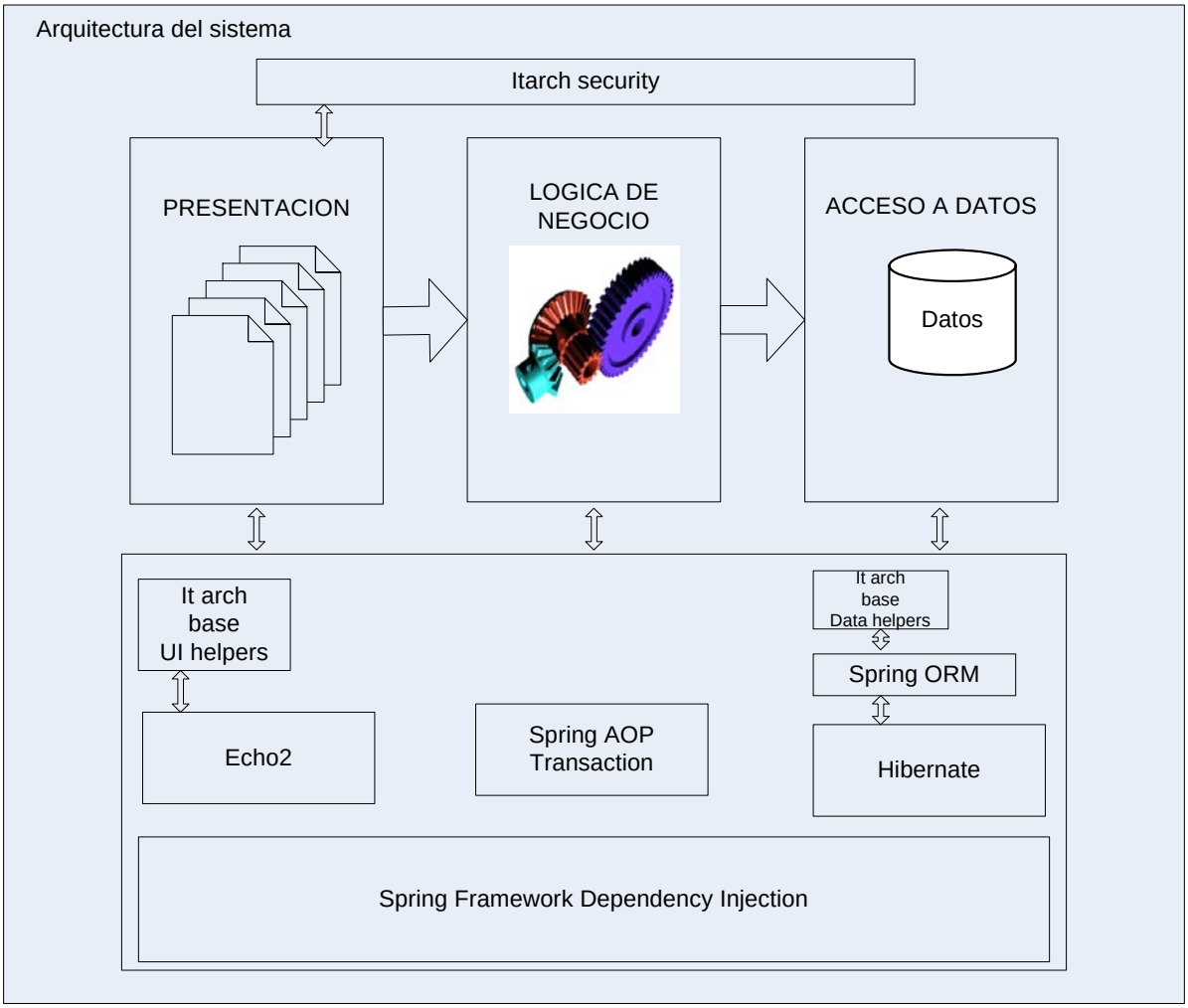
La **capa de presentación** contiene todos los elementos que constituyen la interfaz con el usuario. Esta capa incluye todo aquello con lo que el usuario puede interactuar, como por ejemplo las pantallas de las aplicaciones, el modelo de navegación del sistema y los adaptadores para cada modo de acceso.

En la capa de lógica de negocio-modelo se modela el comportamiento del sistema, basándose en los datos provistos por la capa de datos, y actualizándolos según sea necesario. Esta capa describe los distintos procesos de negocio que tienen lugar en las organizaciones, desde el ciclo de aprobación de un documento hasta la política de consultar un pedido.

La **capa de acceso a datos** representa el mecanismo por el cual se manipula y persiste la información. Consiste en un administrador de bases de datos relacional (RDBMS), y el esquema de datos propio de nuestra aplicación. Cuando hay varias aplicaciones presentes, los modelos de datos se complementan, evitando la duplicación de información y aumentando las facilidades de brinda el sistema como un todo.

Los diseños de 3 capas son ampliamente utilizados en el mercado, y a lo largo del tiempo han probado sus ventajas. Las aplicaciones en tres capas típicamente tienen mayor capacidad de crecimiento y son más sencillas de mantener, dada su naturaleza altamente modular.

Además de ser una arquitectura de 3 capas, para el diseño del sistema se desarrolla unos componentes arquitectónicos (todos los que cuelgan del paquete *es.itarch.arch*) de apoyo. Es a lo que nos referimos como la arquitectura adicional desarrollada para este proyecto.



En el anterior diagrama se ve la distribución de los elementos que la formarán y como se integran las tecnologías que se utilizan y se han comentado.

CAPA DE PRESENTACION

Los componentes propios del proyecto en esta capa cuelgan del paquete *es.itarch.sgocf.ui*

En *es.itarch.arch.base.ui* (*it arch base ui helpers* en el diagrama) da clases para apoyar:

- Gestión de pantallas
- Gestión de seguridad de pantallas según usuarios
- Internacionalización
- Gestión centralizada de mensajes y literales de la aplicación
- Gestión de las validaciones de campos
- Simplificación de uso de echo2 y Spring desde los objetos de la aplicación
- Integración de echo2 y Spring Dependency Injection
- Y con la contribución de *es.itarch.arch.security* (*itarch security* en el diagrama) soporte para el mantenimiento de usuarios, grupos y permisos

Echo2 proporciona el marco para dar aspecto de cliente pesado dentro de un cliente ligero, utilizar mediante su modelo de desarrollo de UI el uso de componentes y el patrón MVC.

CAPA DE LÓGICA DE NEGOCIO

Los componentes propios del proyecto en esta capa cuelgan del paquete *es.itarch.sgocf.service*

Para esta capa basta con la infraestructura de *Dependency Injection* de *Spring Framework* y se utiliza los mecanismos que dota para añadir configuración transaccional declarativa mediante los archivos xml de configuración de Spring. No hay que tocar el código para cambiar el comportamiento transaccional de un objeto de negocio.

Los objetos de esta capa se acaban en *Manager* y ocultan e los objetos de acceso a datos que se necesitan para realizar un método de negocio además de definir la transaccionalidad de negocio.

Como en esta gracias a la Dependency Injection de Spring se ven los objetos por interfaces para los que se ha dado una implementación actual pero que podría ser cambiada en el futuro por otra sin alterar el resto de las capas.

CAPA DE ACCESO A DATOS

Los componentes propios del proyecto en esta capa cuelgan del paquete *es.itarch.sgocf.dao*

Son los objetos que hablan con el motor de persistencia para recuperar y almacenar objetos.

Es esta capa también gracias a la Dependency Injection de Spring se ven los objetos por interfaces para los que se ha dado una implementación actual pero que podría ser cambiada en el futuro por otra sin alterar el resto de las capas.

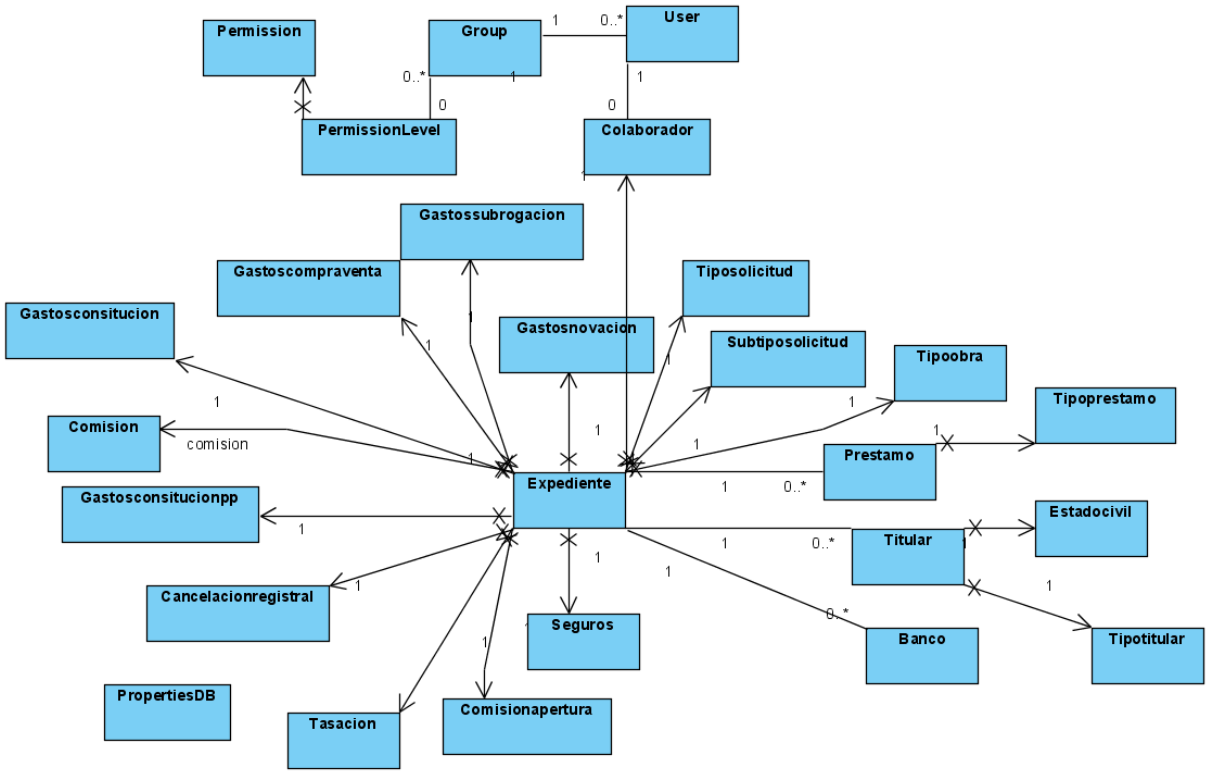
Por ejemplo podríamos cambiar el motor de persistencia de hibernate a ibatis tan solo tocando los objetos de implementación del DAO (¡no los interfaces!).

En esta capa se utiliza *es.itarch.arch.base.data* que proporciona algunas clases de ayuda para los criterios de búsqueda de hibernate.

También se usa las clases comentadas en el apartado de *Spring* del módulo de *Spring ORM* con los beneficios que aportan (ya comentados)

MODELO

Los componentes propios del modelo de datos del proyecto se arrastran por todas las capa como contenedores de la información de negocio y cuelgan del paquete *es.itarch.sgocf.model* y se utilizan por las clases que desarrollan el subsistema (por ejemplo, gestión de titulares) según los necesiten (en este ejemplo se usarán los objetos de titulares y sus entidades relaciones)



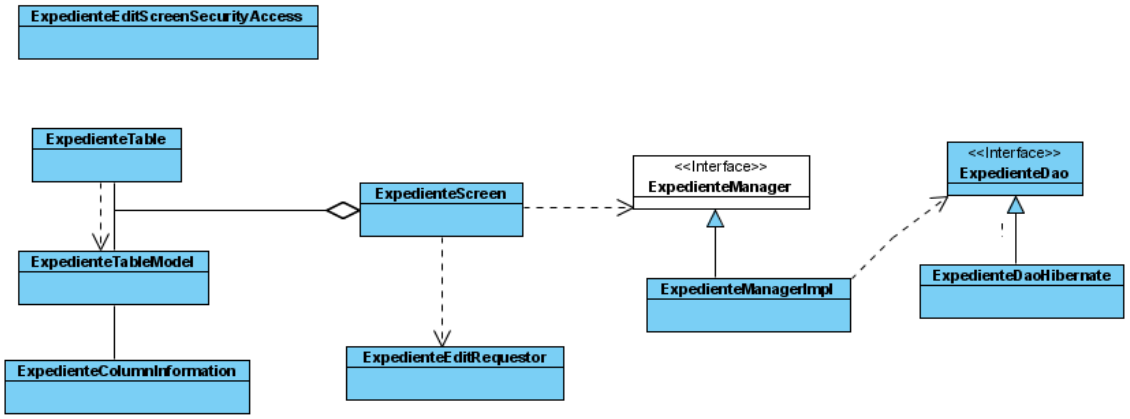
7.1.5 Diseño de subsistemas

La arquitectura es muy ortogonal. En los siguientes diagramas se muestran las clases que se generarán para cada subsistema (agrupación por funcionalidad de caso de uso) y como ser relacionan:

7.1.5.1 Gestión de expedientes

Recoge los casos de uso:

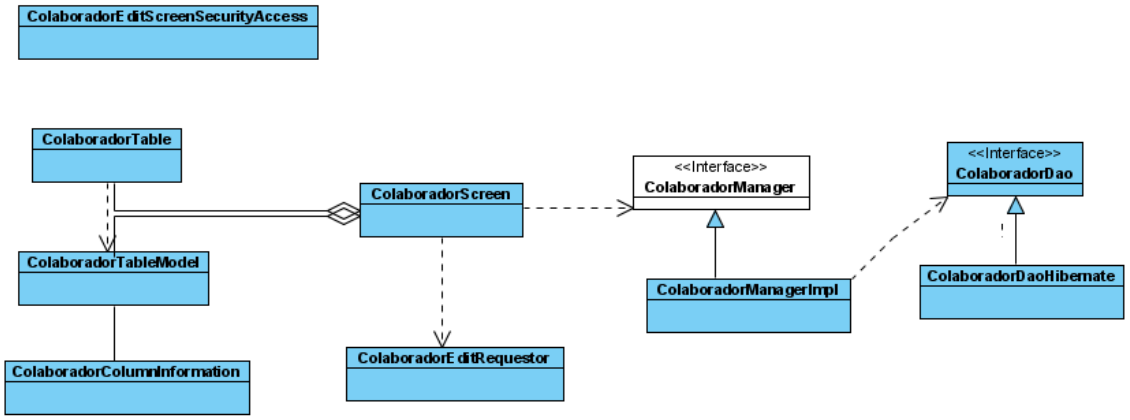
- Consulta de expedientes
- Alta de expediente
- Modificación de expediente
- Baja de expediente



7.1.5.2 Gestión de colaboradores

Recoge los casos de uso:

- Consulta de colaboradores
- Alta de colaborador
- Modificación de colaborador
- Baja de colaborador

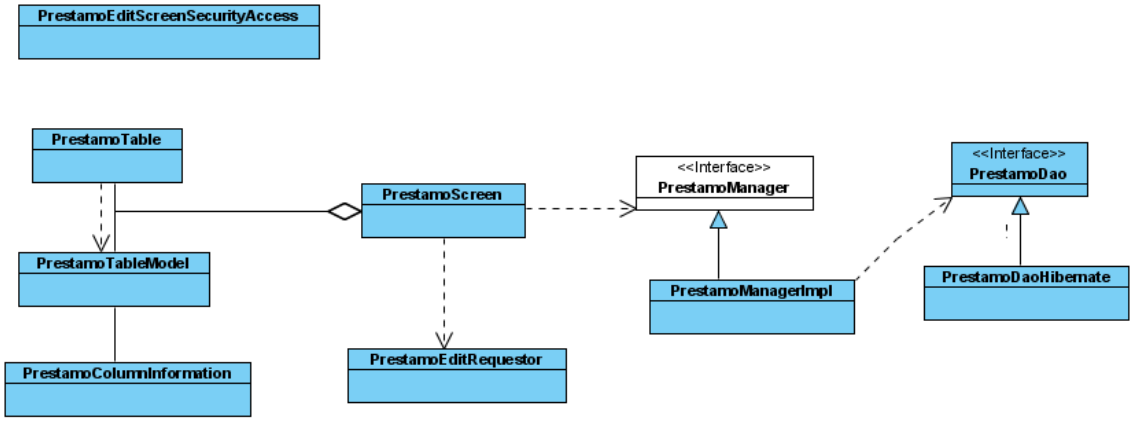


7.1.5.3 Gestión de prestamos

Recoge los casos de uso:

- Consulta de prestamos

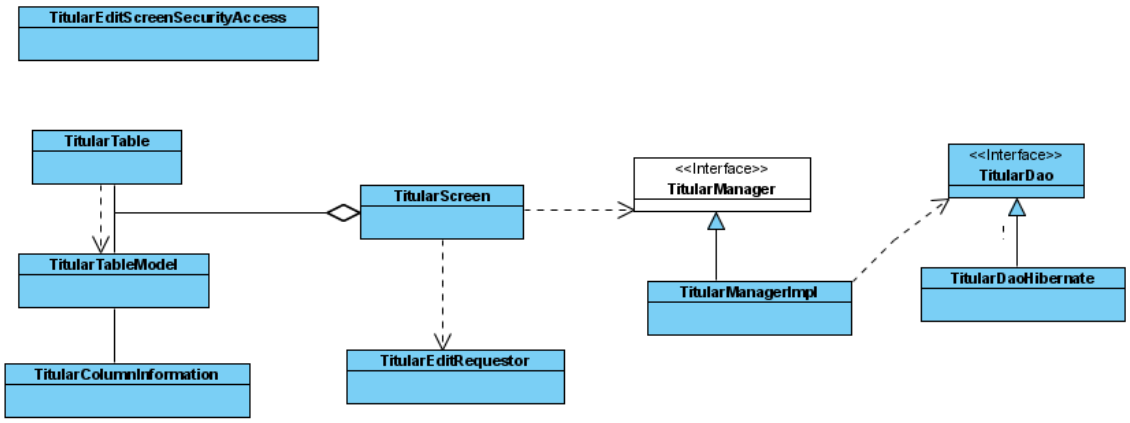
- Alta de préstamo
- Modificación de préstamo
- Baja de préstamo



7.1.5.4 Gestión de titulares

Recoge los casos de uso:

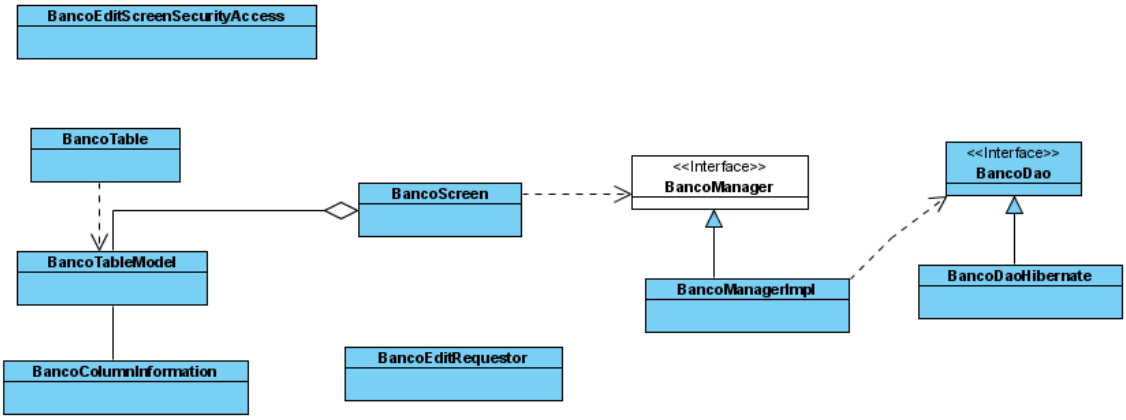
- Consulta de titulares
- Alta de titular
- Modificación de titular
- Baja de titular



7.1.5.5 Gestión de bancos

Recoge los casos de uso:

- Consulta de bancos
- Alta de banco
- Modificación de banco
- Baja de banco



7.1.5.6 Gestión de seguridad

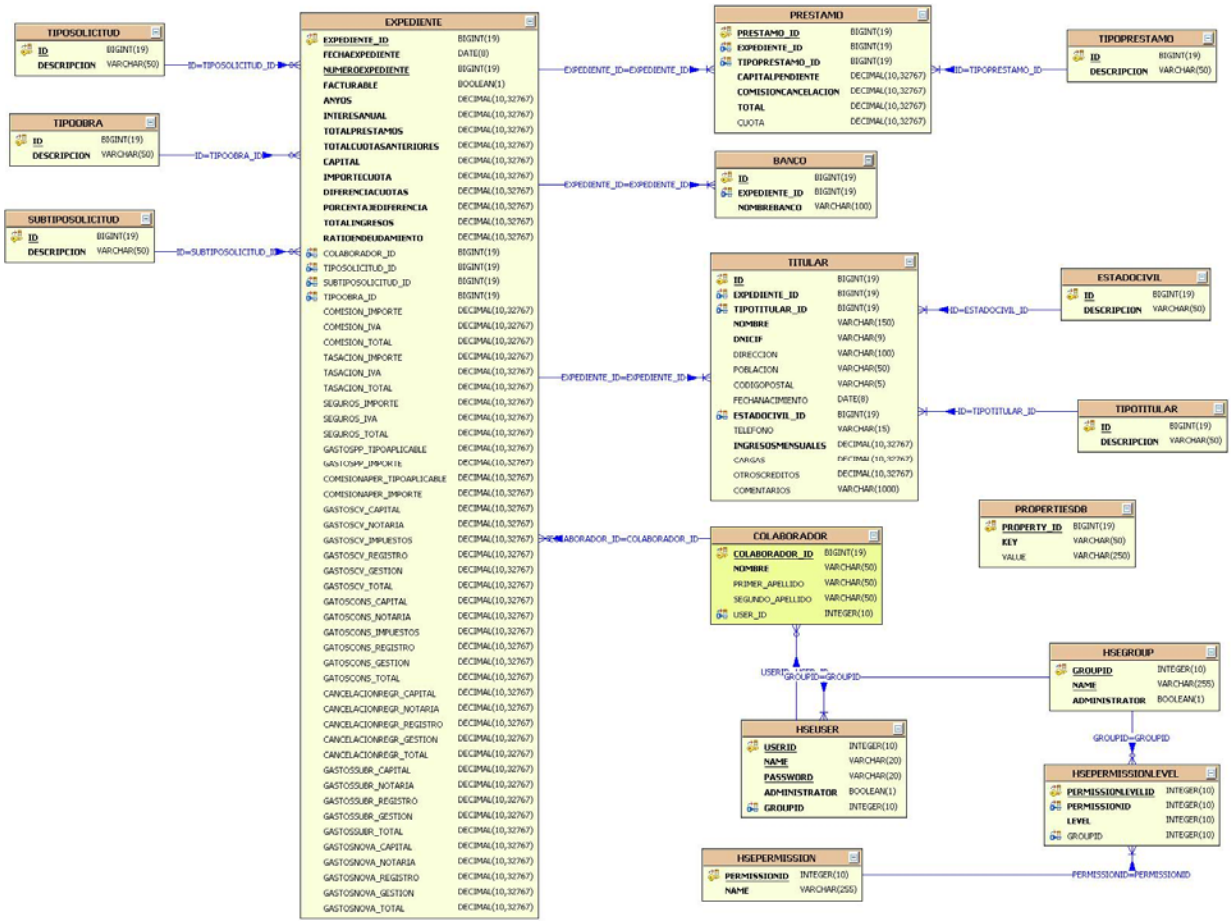
Recoge los casos de uso:

- Consulta de usuarios
- Alta de usuarios
- Modificación de usuarios
- Baja de usuarios
- Consulta de grupos
- Alta de grupos
- Modificación de grupos
- Baja de grupos

Sigue los patrones mostrados en el resto de subsistemas.

7.2 Modelo de datos

En el siguiente diagrama entidad relación se muestran las tablas que ha generado el sistema de persistencia del sistema.



Los campos de cada tabla se corresponden con atributos / propiedades de los objetos del modelo de la aplicación, considerando las siguientes traducciones de los tipos.

Modelo Entidad Relación	Modelo de clases java
BIGINT	Long
BOOLEAN	Boolean
INTEGER	Integer
DECIMAL	BigDecimal
VARCHAR	String
DATE	java.util.Date

Por ello no se especifican en los diagramas de clases que se centran más en las relaciones entre las clases que la definición de sus propiedades. Si se observa, coinciden los nombres con los expresados en la especificación de requisitos.

La única cosa peculiar es que las relaciones 1 a 1 de la clase *Expediente* con las clases *GastosX*, *ComisionX*, *CancelacionRegistral*, *Seguros* y *Tasación* por ser 1 a 1 e ir ligadas al ciclo de vida de *Expediente* se han mapeado de los objetos java por cada atributo de la clase como una columna de la tabla que mapea la entidad de *Expediente.*

8 Construcción del sistema

La construcción del sistema se ha realizado en el entorno integrado de desarrollo (IDE) *Eclipse*.

Se ha generado y se distribuye con el proyecto la documentación en javadoc. La documentación se ha desarrollado (salvo que exista algún gazapo) en inglés (así como la mayor parte de los nombres de clases y métodos, excepto aquellos que pertenecen al modelo del negocio como por ejemplo *Expediente*) Se ha realizado así por costumbre del alumno y en vistas de si libera parte del código como open source.

Se desarrollo también un manual de usuario.

Al usar *Dependency Injection*, los objetos relevantes se enlazan entre sí mediante lo especificado en los archivos xml de configuración de *Dependency Injection* de *Spring*:

```
/es/itarch/arch/base/baseContext.xml

/es/itarch/arch/standardsecurity/springContext.xml

/es/itarch/sgocf/springContext.xml

/es/itarch/sgocf/dao/hibernate/applicationContext-hibernate.xml

/es/itarch/sgocf/service/applicationContext-service.xml

/es/itarch/sgocf/ui/echo2/screen/applicationContext-screens.xml
```

Se han estructurado por capas y en los paquetes que corresponden por la arquitectura de paquetes de la aplicación.

Además estos archivos refieren a algunas configuraciones que están en archivos de *properties*:

```
\WEB-INF\application.properties

\WEB-INF\permissions.properties

\WEB-INF\screens.xml

Messages.properties
```

La configuración del acceso a base de datos está en *application.properties*

Todos los mensajes y literales que muestra la aplicación están en: *Messages.properties*

El subsistema de arquitectura de gestión de grupos lee *permissions.properties* para considerar que permisos utiliza en el interfaz de usuario.

El archivo *screens.xml* es utilizado por el gestor de pantallas de la arquitectura de interfaz de usuario para saber la organización de menús y que objetos son los que contienen las pantallas de cada opción, así como de que beans tiene que utilizar para verificar si un usuario puede acceder a dichas pantallas.

Todos los archivos de configuración han sido extensamente documentados dentro de sí mismos para que sea fácil el trabajar con ellos.

Se realizaron las pruebas unitarias -cuando se necesitaron- y de integración durante esta fase.

9 Implantación y aceptación del sistema

Se ha desarrollado el manual de instalación que se adjunta como anexo.

Se realizaron pruebas de implantación y pruebas de sistema y aceptación en varios entornos de despliegue.

Las pruebas de sistema y aceptación se han basado en realizar al menos un caso de prueba por cada requisito.

El cliente ha iniciado las pruebas de aceptación por su parte y se espera que antes de final de mes de por aprobado el producto.

10 Valoración económica

Se han dedicado 16 semanas de una persona a 15 horas semanales lo cual hace un total de 240 horas aproximadamente.

Dado que la persona ha despeñado todos los roles le vamos a asignar una tarifa media de 35 euros sin IVA (me baso en la experiencia laboral de ofertas que realiza mi empresa).

Por lo tanto, el valor de este software si se vendiese podría ser de 8.400 EUROS sin incluir el IVA.

No se han tenido en cuenta costes ni amortizaciones de software (todos los productos han sido gratuitos excepto el sistema operativo y el paquete office) ni de hardware (se ha desarrollado en el ordenador del alumno).

11 Conclusiones

La sensación final de pasar por esta experiencia es de gran satisfacción

En principio se han alcanzado los objetivos propuestos: se ha analizado, diseñado e implementado una aplicación real para una determinada empresa, consiguiendo para ello desarrollar una aplicación con un interfaz muy intuitivo que hará que la curva de aprendizaje del manejo del producto sea realmente corta.

Además se ha lidiado con el cambio de alcance que se produjo en la fase de análisis transformándose del desarrollo de un sistema que cumpliese con las funcionalidades esperadas a un doble proyecto de desarrollo: Por una parte del sistema que cumpliese con las funcionalidades y por otra una arquitectura sobre la que se desarrollaría este sistema que incorporase tecnologías punteras y las mejores prácticas de desarrollo.

En el ámbito tecnológico se ha desarrollado al completo esta arquitectura general para aplicaciones empresariales, produciendo una gran satisfacción por el logro obtenido y las características que reúne. De hecho se está considerando continuar con el desarrollo de un generador de aplicaciones sobre esta arquitectura a partir de las definiciones de objetos sencillos (POJO) del modelo de negocio.

La arquitectura está construida sobre *Echo2* para el interfaz de usuario, *Hibernate* para la gestión de la persistencia y *Spring Framework* como pegamento para unirlo todo. Esta arquitectura proporciona un conjunto de servicios a las aplicaciones desarrolladas sobre ella como son la gestión de pantallas, gestión de seguridad y permisos de grupos de usuarios, internacionalización y gestión centralizada de mensajes y literales de la aplicación, un Framework para el interfaz de usuario que simplifica su construcción y da soporte para la validación de campos y el control de la seguridad y el aspecto de cliente pesado dentro de un cliente ligero, así como clases de ayuda para trabajar con la persistencia, las transacciones y los criterios de las búsquedas de información.

Se ha conseguido desarrollar un software del mundo real con herramientas y tecnologías Open Source tanto para su desarrollo como para su implantación. Por tanto, al no tener el peso de una política de licencias a sus espaldas, con el ahorro tan enorme en costes que esto supone y dado que es la plataforma java, se puede implantar en cualquier infraestructura hardware disponible.

Por otra parte, se ha seguido un ciclo de vida del software para el proyecto que se ha desarrollado sin desviaciones excesivas respecto a lo planificado. La única desviación fue interna al proyecto debido a una dificultad de posibilidad de dedicación temporal en el inicio de la fase de construcción planificada, que luego fue compensada con más dedicación en las semanas posteriores y se ha conseguido llegar a la entrega final del producto software conforme a los requisitos pactados con el cliente.

La herramienta de desarrollo utilizada ya se dominaba (*Eclipse*) con lo cual no se ha aprendido mucho en este aspecto, pero sin embargo, a nivel de tecnologías JEE, se ha profundizado en el conocimiento inicial que ya se tenía de *Spring Framework* e *Hibernate* y se ha tenido la oportunidad, dado el objetivo de desarrollar una nueva arquitectura, de buscar, seleccionar y aprender la que el autor de esta memoria (y otros más) creen que es la mejor tecnología actual y puntera para el desarrollo potente, rápido y mantenible del interfaz de usuario en aplicaciones empresariales con interfaz Web.

En resumen, 313 archivos y 20922 bytes de código fuente, una gran satisfacción y deseos de continuar el asunto de la arquitectura (con la idea de un generador de código para ella). ☺

12 Glosario

BD	Base de datos.
Bean	Un Bean es un componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno. Un bean es un objeto que debe cumplir unos requisitos: implementar Serializable, tener todos sus atributos privados (private), tener métodos set() y get() públicos de los atributos privados que nos interesen y tener un constructor público por defecto
EJB	Enterprise JavaBeans. El API define un conjunto de APIs que un contenedor de objetos distribuidos soportará para suministrar persistencia, RPCs (usando RMI o RMI-IIOP), control de concurrencia, transacciones y control de acceso para objetos distribuidos.
Hibernate	ORM de código abierto de reconocido prestigio en la comunidad de desarrolladores que ha influido en la nueva especificación EJB 3 para definir el API empresarial JEE JPA. De hecho actualmente Hibernate es una de las implementaciones de JPA.
JEE	Java Enterprise Edition (antes conocida por J2EE). Especificaciones para el desarrollo de aplicaciones empresariales. Estándar liderado por Sun para aplicaciones empresariales.
JPA	Java Persistence API, es el API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB 3. Este API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de este API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí pasaba con EJB2, y permitir usar objetos regulares (conocidos como POJOs).
ORM	Object Relational Mapping. Puente entre el mundo de los objetos y las bases de datos relacionales
POJO	Un POJO (acrónimo de Plain Old Java Object) es una sigla creada por Martin Fowler, Rebecca Parsons y Josh MacKenzie en septiembre de 2000 y utilizada por programadores de Java para enfatizar el uso de clases simples y que no dependen de un framework en especial.
SGBDR	Sistema Gestor de Bases de Datos Relacionales.
Spring Framework	Es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. Entre todas las buenas prácticas que reune, destaca por dotar del patrón Dependency Injection y Aspect Object Programming (AOP) y otras muchas clases de soporte para las habituales necesidades de las arquitecturas de aplicaciones java empresariales.
TFC	Trabajo de Fin de Carrera
UML	Unified Modelling Language. Lenguaje para modelo utilizado como técnica del análisis y diseño de software.

13 Bibliografía

JEE

The Java EE 5 Tutorial.For Sun Java System Application Server 9.1

http://java.sun.com/javaee/5/docs/tutorial/doc/

Enterprise JavaBeans Specification Documentation 3.0 Final Release

http://java.sun.com/products/ejb/docs.html

Patrones de diseño y patrones de arquitectura

Patterns of Enterprise Application Architecture. (The Addison-Wesley Signature Series) by Martin Fowler

Design Patterns: Elements of Reusable Object-Oriented Software. (Addison-Wesley Professional Computing Series) by Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides

Core J2EE™ Patterns: Best Practices and Design Strategies. By Deepak Alur, John Crupi, Dan Malks. Publisher: Prentice Hall.

J2EE Design Patterns. By William Crawford, Jonathan Kaplan. Publisher: O'Reilly

Proceso de desarrollo de software y UML

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. (3rd Edition) (Hardcover) by Craig Larman

Refactoring: Improving the Design of Existing Code, (The Addison-Wesley Object Technology Series) (Hardcover) by Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts

The Unified Software Development Process. (Addison-Wesley Object Technology Series) (Hardcover) by Ivar Jacobson, Grady Booch, James Rumbaugh

The Unified Modeling Language Reference Manual. (2nd Edition) (The Addison-Wesley Object Technology Series) (Hardcover) by James Rumbaugh, Ivar Jacobson, Grady Booch

The Unified Modeling Language User Guide. (2nd Edition) (The Addison-Wesley Object Technology Series) by Grady Booch, James Rumbaugh, Ivar Jacobson

Use Case Modeling. (The Addison-Wesley Object Technology Series) by Kurt Bittner, Ian Spence.

Spring Framework

Spring in Action. By Craig Walls and Ryan Breidenbach. Publisher: Manning.

Spring Live. by Matt Raible. Publisher: SourceBeat

Spring: A Developer's Notebook. By Justin Gehtland, Bruce A. Tate Publisher: O'Reilly

POJOs in Action: Developing Enterprise Applications with Lightweight Frameworks. by Chris Richardson Pro Spring by Rob Harrop, Jan Machacek Publisher: Apress

Professional Java Development with the Spring Framework. By Rod Johnson, Juergen Hoeller , Alef Arendsen, Thomas Risberg, Colin Sampaleanu Publisher: Wiley Publishing, Inc.

Hibernate

Hibernate in Action. By Christian Bauer, Gavin King Publisher: Manning

Java Persistence with Hibernate. By Christian Bauer, Gavin King Publisher: Manning

Beginning Hibernate From Novice to Professional An introduction to all the new features of the Hibernate 3.2 persistence API. By Dave Minter and Jeff Linwood. Publisher: Apress

14 Anexos

14.1 Manual de instalación

A continuación se describirán las acciones para instalar la aplicación. Existen dos modos:

14.1.1 Instalación rápida

Este modo se ha pensado para hacer más fácil la tarea al evaluador del TFC (facilitando el proceso de validación) o para las pruebas de aceptación del cliente.

Además del producto entregado y su manual de instalación que viene en el siguiente apartado (más complejo), también está disponible un paquete de instalación mínima por si quien va evaluar el software quiere ahorrarse esos trámites de instalación por unos más sencillos.

Se ha construido un pack en un archivo zip que integra todo lo necesario para arrancar la aplicación (máquina virtual java, base de datos, y servidor de aplicaciones)

Este pack está pensado para trabajar bajo Windows.

Está disponible para su descargar en:

http://www.itarch.es/drodriguezvil_productoyainstalado.zip

Para instalarlo, se descarga y se descomprime a una carpeta la que se desee pero preferiblemente una raíz de una unidad o como mucho del siguiente nivel, porque se generan nombres de ruta de archivos largos.

> **Importante:** Para que funcione, el puerto 8080 tiene que estar libre. Si se tiene algún servidor web en dicho puerto se debería parar antes de seguir con la instalación y ejecución

Se entra en la carpeta descomprimida y se encuentran dos archivos .BAT:

arrancaraplicacion.bat Este arranca el servidor y lanza un navegador a la entrada de la aplicación

pararaplicacion.bat Este para el servidor y la aplicación.

Se pueden lanzar estos programas con dobleclick desde el explorador de Windows.

Si el navegador arranca y da un error de que no se puede conectar, se debe volver a intentar, ya que en ocasiones (dependiendo de la máquina) a esta aplicación le lleva un rato arrancar (por ejemplo cuando crea la BD). Para volver a intentarlo arrancar un nevegador web y acceder a la url de entrada a la aplicación:

http://localhost:8080/sgocf

Nota: El usuario y clave por defecto de administrador es *admin* y *adminchangeme.* Para mejorar el proceso de validación, se ha habilitado la opción de identificarse por defecto como administrador del sistema en la pantalla de login (los datos ya están introducidos) esto no estaría así en un entorno de despliegue de produccióng.

14.1.2 Instalación normal

Para la instalación de la aplicación es necesario lo siguiente:

1. Disponer de una *máquina virtual java 1.4 o superior*
2. Disponer de un servidor *Apache tomcat*. La versión debe ser 5.x o superior.

Respecto a la BD, el proyecto se ha desarrollado para correr en casi cualquier gestor de base de datos relacional para el que existan *drivers jdbc* (usa *Hibernate* para la persistencia que es independiente del gestor de BD).

En el pack de las librerías están incluidos los *drivers* y sistemas de gestión de bases de datos relacionales java H2 y apache Derby.

Por defecto, la configuración de la distribución que se entrega para instalarla de esta forma, trabajará en modo embebido con el SGBDR H2, creando la base de datos y los datos de carga inicial. Por lo tanto, no es requisito tener o instalar ningún servidor de base de datos. (ver detalles al final del apartado en *cambiar la BD que se utiliza*).

1.- Despliegue del producto en *Apache tomcat*

Se debe descomprimir el archivo *drodriguezvil_producto.zip* a un directorio temporal.

Una vez descromprimido quedará un árbol como el que aparece a continuación (aquí se ha quitado ramas que no son relevantes para la explicación actual)

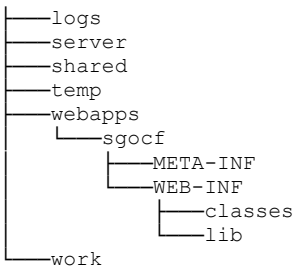
```
DRODRIGUEZVIL_PRODUCTO
├──build
│   └──sgocf
│       ├──META-INF
│       └──WEB-INF
│           ├──classes
│           └──lib
├──javadoc
└──src
    ├──java
    └──webroot
```

El directorio *buiild* contiene la aplicación compilada y lista para su despliegue en el servidor de aplicaciones tomcat en el cual se vaya a instalar.

Se debe copiar desde *sgocf* en *build* (incluido el *sgocf*) a la carpeta de aplicaciones web *webapps* del tomcat.

Después de realizar esta copia quedará una estructura como esta (aquí se ha quitado ramas que no son relevantes para la explicación actual):

```
\APACHE-TOMCAT-5.5.25
├──bin
├──common
├──conf
```



2.- Despliegue de las librería en *Apache tomcat*

Todas las liberias agrupadas se encuentran para descargar en:

http://www.itarch.es/drodriguezvil_librerias.zip

Se debe descomprimir el archivo *drodriguezvil_librerias.zip* al directorio

```
\apache-tomcat-5.5.25\webapps\sgocf\WEB-INF\lib
```

De la estructura del *apache tomcat* anteriormente mostrada como ejemplo (debe ser la del *tomcat* donde se instale)

3.- Arrancar servidor *Apache tomcat*

Se debe arrancar el servidor *apache tomcat*. Esto se realiza mediante la linea de comandos ejecutando el archivo .bat (en otros sistemas linux o unix, usar el comando adecuado del sistema):

```
\apache-tomcat-5.5.25\bin\startup.bat
```

4.- Acceder a la aplicación

Se debe iniciar un navegador web.

La url para entrar a la aplicación es:

http://localhost:8080/sgocf

Cambiar la BD que se utiliza:

Se debe alterar un fichero de configuración y añadir los drivers jdbc al directorio de librerías (si no los tiene). Las librerías de los drivers jdbc deben situarse en:

```
\apache-tomcat-5.5.25\webapps\sgocf\WEB-INF\lib
```

El fichero de configuración es:

```
\apache-tomcat-5.5.25\webapps\sgocf\WEB-INF\application.properties
```

Los comentarios en este archivo son líneas que empiezan por #

Para cambiar la BD hay que comentar las líneas de configuración de base de datos que estén sin comentar y descomentar las que apliquen y modificarlas con los datos necesarios de driver, url, dialecto hibernate,… Los nombres y ejemplos comentados en el archivo son bastante auto explicativos.

14.2 Manual de usuario

Por el límite del espacio de la memoria no se ha incluido el manual de usuario. No obstante, esta información en gran parte ya está recogida en el apartado de *Casos de uso* y el de *Especificación del interfaz de usuario*.