UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

DISEÑO DE UN PORTAL WEB DE GESTIÓN DE CARTERAS DE ACCIONES

DISCA-14

PROYECTO FINAL DE CARRERA

Autor: Antonio Poveda González

Director: Sergio Sáez Barona

Valencia, 2011

Índice:

1. Introducción ... 3
2. Especificación de requisitos ... 5
2.1. Introducción ... 5
2.1.1. Propósito ... 5
2.1.2. Ámbito ... 5
2.1.3. Definiciones, acrónimos y abreviaturas ... 5
2.1.4. Referencias ... 7
2.1.5. Visión global ... 8
2.2.1. Perspectiva del producto ... 8
2.2.2. Funciones del producto ... 8
2.2.3. Características del usuario ... 9
2.2.4. Restricciones ... 10
2.2.5. Supuestos y dependencias ... 10
2.3. Requisitos Específicos ... 11
2.3.1. Requisitos Funcionales ... 11
3. Análisis ... 17
3.1. Introducción ... 17
3.2. Diagrama de clases ... 17
3.3. Casos de uso ... 18
3.4. Diagramas de secuencia ... 20
4. Diseño ... 24
4.1. Introducción ... 24
4.2. Arquitectura de tres capas ... 24
4.2.1. Nivel de interfaz ... 25
4.2.2. Nivel de lógica de la aplicación ... 28
4.2.3. Nivel de datos o persistencia ... 29
4.3. Diagrama entidad relación ... 29
4.4. Diseño lógico ... 30
5. Implementación ... 32
5.1. Tecnologías ... 32
5.1.1. HTML ... 32
5.1.2. HTTP ... 33
5.1.3. Arquitectura cliente – servidor ... 33
5.1.4. PHP ... 34
5.1.5 CSS ... 34
5.1.6. jQuery ... 35
5.1.7. MySQL ... 35
5.2. Herramientas ... 36
5.3. Detalles de la implementación ... 36
5.3.1. Perfiles de usuario ... 36
5.3.2. Objetos: interfaz más manejable ... 37
5.2.3. Persistencia de sesión del usuario ... 37
5.2.4. Login ... 38
5.2.5. Zona Usuario ... 40

5.2.6. Alta Usuario (JavaScript) 40
5.2.7. Gráficos con amCharts: Estado actual 42
5.2.8. JQuery: Selector de fecha 43
6. Evaluación y pruebas 46
6.1. Evaluación 46
6.2 Pruebas 46
6.2.1. Pruebas de Validación HTML y CSS 46
6.2.2. Pruebas de compatibilidad en navegadores 47
6.2.3. Pruebas de compatibilidad de resolución 48
6.2.4. Pruebas unitarias 49
- **Proceso de introducción de operaciones** 50
- **Estado actual** 54
- **Listado de operaciones** 55
- **Información Fiscal** 55

7. Conclusión 58
7.1. Trabajo realizado 58
7. 2. Valoración personal 58
7.3. Posibles ampliaciones 59
8. Bibliografía 61

1. Introducción

Muchos bancos no ofrecen una plataforma sencilla donde poder analizar, detallar o consultar los movimientos que se realizan en bolsa. A veces se limitan a mostrar los resultados tal y como los trata Hacienda. Y nosotros, de vez en cuando, queremos darle otro enfoque.

El propósito de este proyecto es el desarrollo de una aplicación Web específica para la administración de carteras de valores. La aplicación permitirá al usuario simular carteras o fondos de inversión y crear operaciones bursátiles con títulos que cotizan en el Ibex 35. Estas operaciones podrán consultarse, así como mostrar estadísticas personalizadas, resúmenes, consultas, etc.

Cabe destacar que la aplicación no emite órdenes reales sobre el mercado. Simplemente es una herramienta de gestión de operaciones que pueden ser reales o ficticias.

En cuanto al contexto del proyecto, por un lado, puede ser una herramienta de ayuda para aquellas personas que operan en Bolsa en la realidad. El usuario dará de alta las operaciones que ha realizado, a priori, en el mercado, introduciendo los datos que le faciliten su broker o el banco.

Por otro lado, se quiere atraer al mayor número de usuarios posible, y no es necesario ser accionista real. Además de que cuenta con una estructura clara y sencilla, con esta aplicación podremos convertirnos en inversores ficticios, insertando las operaciones de la misma manera que si las hubiéramos hecho de verdad. Puede ser interesante, por ejemplo, para aquellas personas que quieren invertir en bolsa, pero aun no conocen la operativa, no se atreven, etc. Esta herramienta puede ser muy útil como primera aproximación.

El documento que se presenta se divide en diferentes capítulos, y se corresponden con las fases del desarrollo de software tradicional.

A continuación se describe la estructura de la memoria:

Tras la introducción, el segundo capítulo se centra en la especificación de requisitos, detectando los requisitos que debe tener el software. Por otro lado, se explican a fondo todos los objetivos del sistema, además de comentar todos los tipos de restricciones que ha tenido que respetar el proyecto.

El tercer capítulo, análisis, trata de explicar los aspectos generales del software, utilizando el lenguaje de modelado UML. Se

exponen aspectos que tienen que ver con la funcionalidad, la estructura y el comportamiento del software.

En el cuarto capítulo, referente al diseño, se indica cómo se ha puesto solución a los requisitos. También se explica la estructura armada para realizar el proyecto, la definición de capas y qué componentes se han usado.

El quinto capítulo se centra en la parte de implementación, donde se ilustra tanto de qué se ha implementado como de los problemas que se tuvieron y las soluciones que se aplicaron.

El sexto capítulo hace referencia a las posibles pruebas a las que debe someterse la aplicación, cómo documentarlas y sus resultados.

En el último capítulo se valoran las conclusiones y experiencias extraídas, se compara el resultado con los objetivos iniciales, se documenta el trabajo realizado, y además se dan ideas sobre posibles mejoras de la aplicación.

2. Especificación de requisitos

2.1. Introducción

En este apartado se describen el propósito, ámbito, definiciones, acrónimos y abreviaturas, las referencias de la especificación de requisitos y la visión global del producto.

2.1.1. Propósito

El propósito de la especificación de requisitos es servir de base para desarrollar la aplicación Web. También sirve para detallar los requisitos de diseño de interfaz, contenidos y funcionalidad de la misma. Así como la definir claramente los objetivos que se pretenden conseguir.

2.1.2. Ámbito

El producto que se describe a continuación desempeña el papel de una aplicación Web encargada de gestionar las acciones de una o varias carteras de valores. Se abordan todos los aspectos de la gestión de acciones: introducir las operaciones, gestión de carteras, y también mostrar las operaciones y sacar resultados, cálculo de rentabilidad, plusvalías, etc.

Las operaciones no son en tiempo real, son un reflejo de las operaciones que el usuario ha podido hacer en la realidad, o bien operaciones ficticias.

2.1.3. Definiciones, acrónimos y abreviaturas

Definiciones:

Canje: Es una operación de bolsa en la cual el inversor recibe las acciones de un valor a cambio de otro valor ya sea porque va a desaparecer, va a ser absorbido por otra empresa, u otro motivo.

Cliente: Usuario final que accede a un servicio remoto en otro ordenador, llamado servidor.

Dividendo: Operación bursátil en la cual la empresa reparte el beneficio anual obtenido entre sus inversores.

Ibex 35: Nombre como se conoce al índice español de referencia, compuesto por las 35 empresas españolas con más liquidez.

Login: Es el nombre con el que se identifica a un usuario, que con anterioridad ha realizado un proceso de registro. En nuestra aplicación no tendrá ninguna distinción entre roles, sólo hay un tipo de usuario.

Navegador: Aplicación para navegar por Internet y para visualizar documentos HTML.

Norma antiaplicación: Es una norma puesta por Hacienda Pública. Dice que si se venden acciones perdiendo dinero y dos meses antes o después de la venta, existe una compra, Hacienda entiende que son las mismas acciones y por tanto no permite que esa pérdida compense con otras ganancias.

***Password*:** Es el texto que sirve como código secreto de acceso, va relacionado con el Login.

Servidor: Ordenador que se encarga de ejecutar la aplicación y de servir los recursos y páginas a los usuarios o clientes.

Split: Es una operación bursátil. Significa que la empresa va a dividir el precio de la acción voluntariamente, a cambio, el inversor recibirá más acciones con el mismo ratio.

Usuario anónimo: Usuario que visita la Web y del cual no se tiene ningún tipo de información guardada. No puede realizar operaciones, sólo consultar el mercado.

Usuario registrado: Usuario que ha realizado el proceso de registro en la Web, de forma que se dispone de información personal para identificarlo y personalizar su visita a la Web.

Acrónimos y abreviaturas:

CSS: Hojas de estilo en cascada (*Cascading Style Sheets*), es un lenguaje empleado para definir la presentación de un documento escrito en HTML, XHTML o XML.

HTML: Lenguaje de marcado de hipertexto (*HyperText Markup Language)*. Es el principal lenguaje utilizado en las páginas web para describir su contenido y apariencia.

HTTP: Protocolo de transferencia de hipertexto (*HyperText Transfer Protocol)*, es el método de intercambio de información en la Web más utilizado.

JavaScript: es un lenguaje interpretado ligero y sin tipos de datos, utilizado para acceder a objetos en aplicaciones y que no requiere compilación.
MySQL: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario.

PHP: Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas Web dinámicas, usado en el lado del servidor.

SQL: Lenguaje de consulta estructurado (*structured query language)*, es un lenguaje declarativo de acceso a bases de datos relacionales que nos proporciona la posibilidad de precisar diversos tipos de operaciones en estas.

UML: Lenguaje Unificado de Modelado, es un lenguaje gráfico utilizado para construir, documentar, visualizar y especificar un sistema software.

WAMP: Usado para describir un sistema de infraestructuras de Internet que utiliza las herramientas siguientes: *Windows* como sistema operativo, *Apache* como servidor Web, *MySQL* como gestor de base de datos y *PHP* como lenguaje de programación.

Web: *World Wide Web* (WWW), es la red de redes, un medio de comunicación de texto, gráficos y multimedia a través de Internet.

2.1.4. Referencias

- BUENDIA GARCÍA, FÉLIX. *Guía para la realización y supervisión de proyectos de final de carrera (PFC) en el ámbito de la web*. Valencia, Editorial UPV.

- Apuntes ISG. Esquemas y temario.
http://poliformate.upv.es
Accedido Junio de 2011

- Diccionario de la Real Academia Española.
http://rae.es
Accedido Agosto de 2011.

- Diccionario.
http://wordreference.com
Accedido Agosto de 2011.

- Wikipedia. Definiciones.
http://es.wikipedia.org
Accedido en Agosto 2011.

2.1.5. Visión global

Vamos a centrar la especificación de requisitos en la descripción de la aplicación, sus características, sus restricciones generales, sus funciones, sus supuestos y dependencias, que son las que nos aportarán una mayor información del proyecto a desarrollar.

Cuando terminemos de desarrollar la descripción general, pasaremos a la descripción de los requisitos específicos de nuestra aplicación.

2.2. Descripción general

2.2.1. Perspectiva del producto

La aplicación pretende ser un portal Web para la administración de las carteras de valores de los usuarios de dicho portal. Así pues, realizará los cometidos básicos propios de un portal de gestión de carteras tales como alta, baja y modificación de éstas. La modificación de las carteras se realiza básicamente con la gestión de las operaciones asociadas a una acción determinada que está sujeta a negociación en el mercado de valores.

Aunque el producto se conecta a otra aplicación Web para recoger datos en tiempo real, podemos decir que la aplicación es dependiente, ya que esto no afecta a la gestión propia del producto y, en el supuesto caso de no disponer de la aplicación externa, la mayoría de la funcionalidad se ejecutaría correctamente.

2.2.2. Funciones del producto

-Función de consulta en tiempo real del Ibex 35: La aplicación se conecta a una aplicación Web externa, traerá y almacenará los datos de las cotizaciones del selectivo español. La aplicación mostrará siempre estos datos hasta que vuelvan a ser actualizados a petición del usuario, directa o indirectamente.

Esta función la puede realizar cualquier tipo de usuario.

-Funciones de gestión de operaciones: Esta función está disponible sólo para usuarios registrados y que previamente hayan dado de alta una cartera. Mediante la gestión de las operaciones, la cartera se constituye con determinadas acciones. Los tipos de operaciones disponibles serán: compra y venta, dividendos, canjes y splits.

-Funciones de gestión de cartera: Opción para usuarios registrados. Esta funcionalidad es la gestión propia de las carteras; es decir, alta de una nueva cartera, baja, con lo que eliminaremos todas sus operaciones, y selección de la cartera principal, con la cual se hacen las operaciones.

-Funciones de visualización de datos: La aplicación mostrará las operaciones que se han realizado en la cartera seleccionada. El usuario podrá indicar filtros previos, como fecha o tipo de operación.

También se podrá visualizar el estado actual de la cartera, su estructura, es decir, qué acciones tiene en el instante actual dicha cartera, su valor en euros y la revalorización porcentual.

Se dispondrá de un apartado de información fiscal, con la finalidad de ayudar al usuario con la declaración de la Renta. Se mostrarán los datos de la cartera de la misma manera que los trata Hacienda. Se podrá ver de forma fácil cuál fue el importe de compra de las ventas que obligadamente corresponde declarar, además de la plusvalía generada por cada valor.

2.2.3. Características del usuario

La aplicación Web puede ser usada por cualquier usuario, ya que está diseñada para ser intuitiva con usuarios sin experiencia tanto en el manejo de Internet como en la materia de la Web. Sin embargo, para una interacción fluida, es recomendable que el usuario tenga un mínimo conocimiento sobre lo que es una acción, cómo funciona el mercado de valores, la compra-venta y el resto de operaciones menos frecuentes.

Los usuarios que podrán interactuar con la aplicación son:

-Invitado: Dada la motivación del portal, tendrá una funcionalidad muy limitada. Sólo podrá registrarse y ver información sobre las acciones que están cotizando en el mercado en este momento.

-Usuario registrado: Es el usuario que se habrá dado de alta en el sistema. Podrán acceder a realizar todas las funciones de gestión de la aplicación Web.

2.2.4. Restricciones

-**Interfaces externas:**

-Interfaz hardware:

Será suficiente con que los usuarios dispongan de un equipo con conexión a Internet, con un nivel de prestaciones más bien bajo, ya que la aplicación apenas consume recursos.

Además, el monitor debe soportar resoluciones de, como mínimo, 1024x768 o superior.

-Interfaz software:

La parte cliente de la aplicación sólo requiere un navegador Web actualizado para poder conectarse.

La máquina que ofrecerá el servicio de la herramienta deberá tener instaladas las siguientes tecnologías:

-Apache 2.2, PHP versión 5 como mínimo y MySQL.
-Se recomienda la instalación de phpMyAdmin.

-Interfaz de comunicación:

Se utilizarán los protocolos HTTP y TCP/IP, que implementan los navegadores Web para la comunicación cliente-servidor.

2.2.5. Supuestos y dependencias

Se podrá utilizar cualquier sistema operativo en el ordenador donde se utilice el portal Web. El navegador óptimo y para el cual la aplicación ha sido diseñada es Mozilla Firefox, concretamente para versiones superiores a la 3.6.

Otros navegadores recomendados son: Opera, Chrome, Safari o Konkeror, actualizados en su última versión.

La aplicación no da soporte para Microsoft Internet Explorer 7 o anterior y no se garantiza que el funcionamiento sea el adecuado.

2.3. Requisitos Específicos

2.3.1. Requisitos Funcionales

Autentificación de usuario:

Propósito: Mediante un código de usuario único, reconocer al usuario que está ejecutando la aplicación, dándole acceso a todo el portal.
Entrada: Código de usuario y código de contraseña.
Proceso: Comprobar que el nombre de usuario y contraseña coinciden en algún registro de la base de datos.
Salida: Si no se encuentra coincidencia, la salida es la misma interfaz de autentificación con un mensaje de que el código de usuario no existente. Si la encuentra, mostrará el índice de la aplicación, y se desplegará en el menú contextual toda la funcionalidad.

Función de desconexión del usuario:

Propósito: El usuario deja de estar autenticado y pasa a ser un usuario anónimo.
Entrada: -
Proceso: Borra los atributos asociados a la actual sesión del navegador
Salida: La aplicación muestra el índice y la zona de usuario deja de ser visible.

Función de alta de usuario:

Propósito: Recoger datos del nuevo usuario mediante formulario y registrarlo en el sistema. Dar la posibilidad de que un usuario anónimo pueda entrar como registrado.
Entrada: Datos personales del usuario, nombre de usuario y contraseña.
Comprobación de que los campos no se dejan en blanco.
Comprobación de que no existe un nombre de usuario idéntico en el sistema.
Proceso: Crea el usuario
Salida: Muestra un mensaje indicando si el registro se ha realizado con existo. Un mensaje de error si hay campos en blanco, o un mensaje denegando el registro si ya existe otro con ese nombre de usuario.

Función de consulta de valores del Ibex 35:

Propósito: Realizar una consulta de datos de los valores del Ibex 35 almacenados en el sistema.
Entrada: -
Proceso: Lista de valores.
Salida: Una tabla con todos los campos de los valores.

Función de actualización de valores del Ibex 35:

Propósito: Actualizar los datos del sistema con información nueva.
Entrada: -
Proceso: Conexión a aplicación Web remota. Localiza los 35 valores del Ibex y guarda los datos en el sistema. Sobrescribe los datos anteriores. Si encuentra valores nuevos los inserta.
Salida: Muestra la tabla de valores.

Compra de acciones:

Propósito: Consiste en el alta de una operación de compra y asociar a la cartera. Aumentar el patrimonio de la cartera con un valor determinado, por ejemplo "BBVA".
Entrada: Valor, fecha, volumen, precio de la acción, comisión. Identificador de la cartera.
El sistema calcula el importe de la operación automáticamente antes de que comience el proceso.
Comprueba que el volumen es mayor que 0 y que los campos no estén en blanco.
Proceso: Registra el alta de la operación de compra y la asocia a la cartera
Salida: Salida al mismo formulario y un listado de las últimas operaciones de ese valor.

Venta de acciones:

Propósito: Realizar una asignación de operación de venta a la cartera. Disminuir el patrimonio de la cartera seleccionada vendiendo acciones de un valor determinado.
Entrada: Valor, fecha, volumen, precio de la acción, comisión. Identificador de la cartera. El sistema calcula el importe de la operación automáticamente antes de que comience el proceso.
Comprobación de que, a fecha de entrada, la cartera tiene suficiente número de acciones para ser vendidas o que el valor es mayor que 0.
Proceso: Registra el alta de la operación de venta y la asocia a la cartera

Salida: Salida al mismo formulario y un listado de las últimas operaciones de ese valor.

Alta de operación de dividendo:

Propósito: Realizar una asignación de operación de dividendo a la cartera.
Entrada: Valor, fecha, importe por acción, comisión.
Comprobar que los datos no están en blanco.
Proceso: Registra el alta de la operación y la asocia a la cartera
Salida: -

Alta de operación de canje:

Propósito: Realizar una asignación de operación de canje a la cartera.
Entrada: Valor, valor absorbido, fecha, y ecuación de canje.
Proceso: Realiza el alta del canje y lo asocia a la cartera.
Salida: -

Alta de operación de split:

Propósito: Realizar una asignación de operación de split a la cartera.
Entrada: Campos Valor, fecha y cambio.
Proceso: Registra el alta del split y lo asocia a la cartera
Salida: -

Alta de cartera de valores:

Propósito: Asignar al usuario registrado una nueva cartera de valores.
Entrada: Campos nombre, comentarios, principal.
Proceso: Registra la nueva cartera al usuario. Si fue seleccionada como principal, es tratada como tal.
Salida: -

Modificación de datos de una cartera:

Propósito: Opción para modificar los datos de la cartera. El usuario podría, por ejemplo, querer cambiar la descripción de la cartera, para indicar que se basa otro sector distinto al anterior que tuviere, por ejemplo el financiero.
Entrada: Campos nombre y descripción. Identificador de la cartera.
Proceso: Actualiza los datos de la cartera
Salida: -

Selección de cartera principal:

Propósito: Marcar como principal cartera de las operaciones a una determinada cartera
Entrada: Identificador de la cartera
Proceso: Listado de las carteras actuales del usuario. Selección de la cartera principal.
Salida: -

Dar de baja una cartera de valores:

Propósito: Realizará el borrado de una cartera, con sus operaciones asociadas.
Entrada: Identificador de la cartera a borrar
Proceso: Listado de las carteras asociadas al usuario.
Selección de la cartera a borrar.
Listar y borrar las operaciones de dicha cartera.
Borrado de la cartera.
Salida: Mensaje de confirmación de borrado con éxito

Visualización del estado actual:

Propósito: La aplicación listará la cantidad de acciones que tiene la cartera en el instante actual, y su valor en euros.
Entrada: Identificador de la cartera actual
Proceso: Listado de valores que tiene la cartera. Para cada valor:
Listado de las operaciones. Recorrerlas y hacer un sumatorio absoluto del volumen.
Salida: El resultado para cada valor es una cantidad de acciones y un precio medio. Esas acciones tendrán una cotización actual en el mercado, con lo que se calcula el valor total de la cartera en euros, y su revalorización actual desde su compra.

Visualización de las operaciones:

Propósito: Visualizar y/o filtrar información de las operaciones de la cartera asociada, con filtros opcionales.
Entrada: Rango de fechas, valor, tipo de operación.
Proceso: Listado las operaciones de la cartera en función de los parámetros del formulario. Ordenar dicha lista.
Salida: Tabla de las operaciones.

Visualización de la información Fiscal:

Propósito: Pretende ser una ayuda para el cálculo de las ganancias y pérdidas patrimoniales consecuentes de la operativa de las acciones de la cartera.
Se utiliza el método FIFO (primera entrada, primera salida), ya que así es como las trata Hacienda en la declaración.
También se tiene en cuenta la "norma antiaplicación" para las compras anteriores o posteriores a los 2 meses de la venta con pérdida.
Entrada: Listado de todos los valores de la cartera de los cuales están en una operación de venta
Proceso:
Para cada valor:
Listado de todas sus operaciones.
Para cada operación:
- Si es una venta, buscar las compras que le corresponden en base al volumen de la venta y almacenar una plusvalía en euros para esa asociación.

Si la venta tiene pérdidas, buscar compras 2 meses atrás y 2 meses adelante. Si hay, se considera recompra y restar a la minusvalía el importe de la pérdida que no se puede imputar, y guardarla en un vector.

Si la venta genera plusvalía o un volumen actual de 0 en la cartera, y no hay compra en los 2 meses posteriores, imputarle las pérdidas que hay almacenadas de operaciones anteriores.

- Si es un dividendo, calcular el importe total. (Siempre tendrá plusvalía positiva).

- Si es un split o canje, multiplicar el volumen actual por el valor de cambio, para saber el nuevo volumen del título. En el caso del canje, borrar en las compras el valor absorbido, ya que no habrá operación de venta de ese valor.
Salida: Una tabla por cada valor, con las asociaciones Compra-Venta encontradas, su plusvalía y ganancia/pérdida patrimonial correspondiente.
La suma total de cada título en cartera.

Visualización de beneficios de la cartera:

Propósito: Idéntico al anterior, excepto que se utilizará un algoritmo LIFO (ultimo en entrar, primero en salir) con lo que los resultados pueden llegar a variar. Tampoco se tiene en cuenta la "norma antiaplicación" ya explicada.

Entrada: Idéntico al anterior.
Proceso: Idéntico con las restricciones nombradas.
Salida: Idéntico al anterior.

Modificación de datos del usuario:

Propósito: Opción en la zona de usuario para modificar datos propios del usuario tales como teléfono, e-mail o contraseña.
Entrada: Nombre, apellidos, e-mail, teléfono, dirección, contraseña
Proceso: El sistema actualiza los datos pasados.
Salida: -

3. Análisis

3.1. Introducción

El análisis es la construcción de un modelo o especificación detallada del problema del mundo real.

Un buen desarrollo de software debe dedicar un alto porcentaje de los recursos exclusivamente a la etapa de análisis. El análisis constituye una de las actividades primordiales, donde se analizan todas las consideraciones técnicas del proyecto, así como la comprensión y solución del problema que se plantea.

Se debe desarrollar con profundidad el análisis para evitar errores de diseño difíciles de solucionar. No entender la funcionalidad o una mala comunicación con el cliente una vez está terminado el producto, provoca un error en todas las siguientes etapas del desarrollo, con lo cual es un problema grave.

La notación que se va a utilizar es la proporcionada por el estándar UML. En nuestro proyecto se usarán los casos de uso, diagramas de clases y algunos diagramas de secuencia.

3.2. Diagrama de clases

El primer paso va a ser la realización del diagrama de clases, que es el diagrama principal para el modelado orientado a objetos. Representa las clases del sistema junto con sus relaciones y atributos.

Con el diagrama de clases entenderemos qué funciones se pueden llevar a cabo y quien las realiza. Cómo se pueden desarrollar dichas funciones y qué tipo de objetos interactúan entre sí.

Hay dos tipos de usuarios, el usuario anónimo, es el más básico. Sólo podrá consultar y actualizar los valores del Ibex 35 aparte de registrarse. Está muy limitado.

El usuario registrado es una especialización del anónimo, que podrá llevar a cabo toda la funcionalidad del sistema si tiene creada alguna cartera.

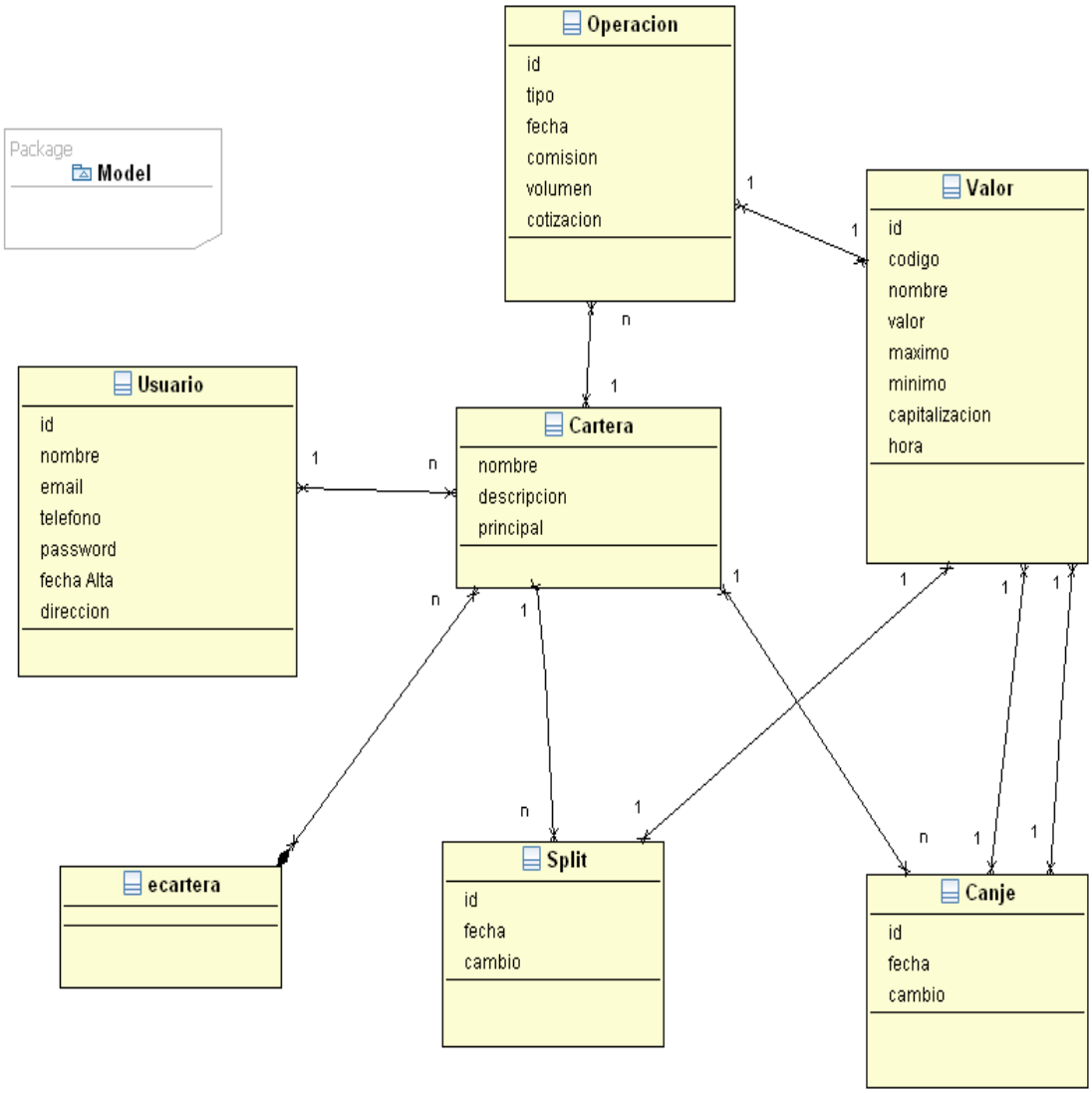


Figura 3-1. Diagrama de clases

3.3. Casos de uso

Un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software.

La utilidad de los casos de uso es capturar las funcionalidades del sistema. Cada caso se compone de uno o más escenarios que representan cómo interactúan los usuarios con el sistema.

La creación de los casos de uso se realiza junto al cliente, por tanto los casos no tienen que ser técnicos, ya que el cliente no dispondrá de conocimientos de Ingeniería Informática. Un Caso de Uso es una interacción cliente-software.

Los usuarios del sistema son llamados actores. Los actores siempre son entidades externas al sistema, los cuales suelen desempeñar un rol, como podría ser usuario, administrador… Estos

deberán realizar ciertas acciones para poder conseguir sus objetivos, cada uno de los objetivos quedan representado como un Caso de Uso.

En el siguiente diagrama mostramos el caso de uso realizado para nuestro proyecto:

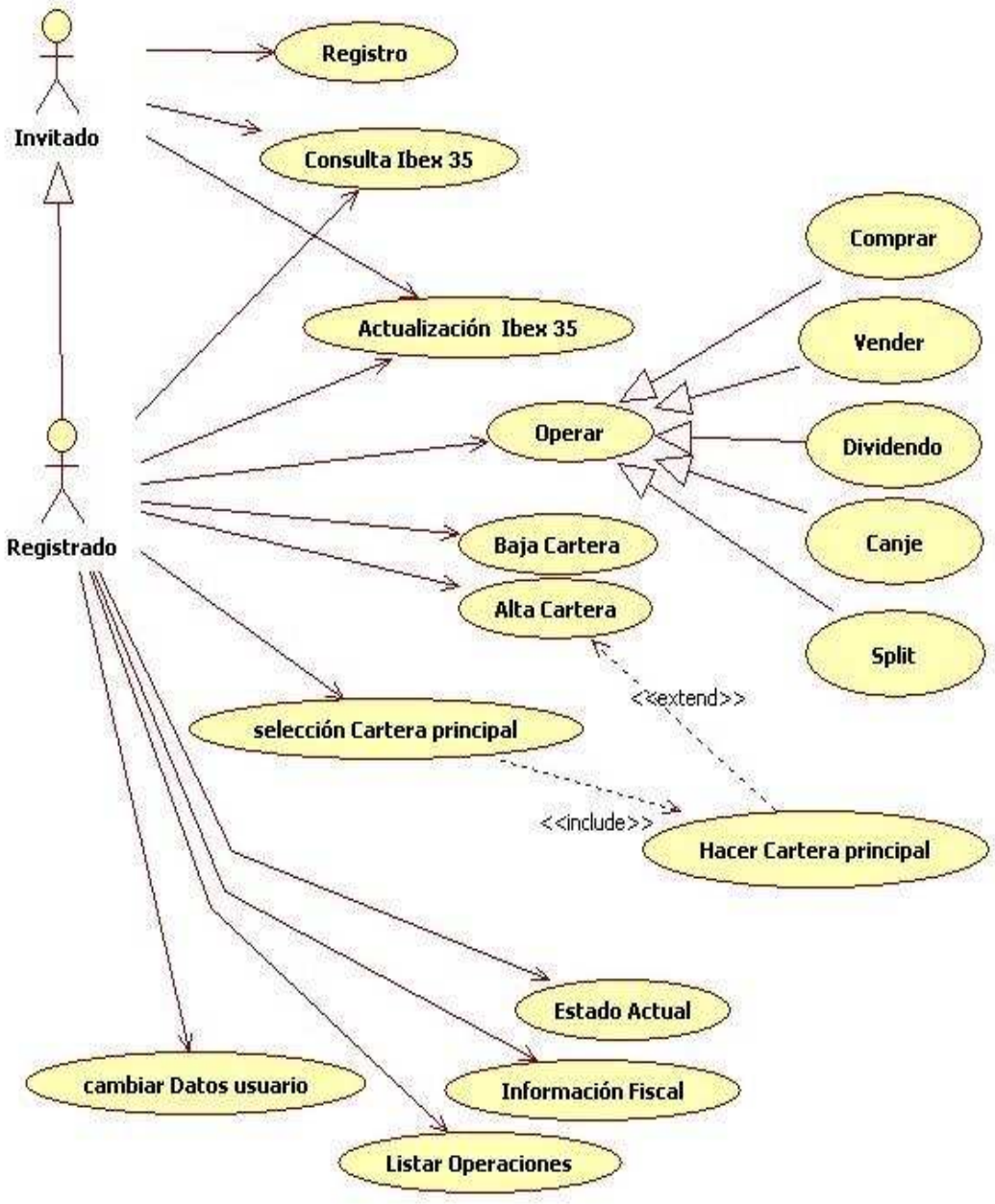


Figura 3-2. Casos de uso

La Figura 3-2 muestra los dos tipos de usuarios ya comentados en la sección anterior. Vemos como el usuario anónimo sólo puede hacer

tres cosas: registrarse y consulta o petición de actualización de datos del Ibex 35. El registrado puede hacer cualquier cosa: Operar con acciones y con cualquiera de los tipos: Compra, venta, etc. Gestionar sus carteras, dando de alta o baja. Al hacer un alta de una cartera es opcional marcarla como principal y si se indica hay que llamar a esa funcionalidad. También tiene acceso a todo el tema de la visualización: Operaciones, estado actual e información Fiscal. Por último, también puede modificar sus datos por si han cambiado.

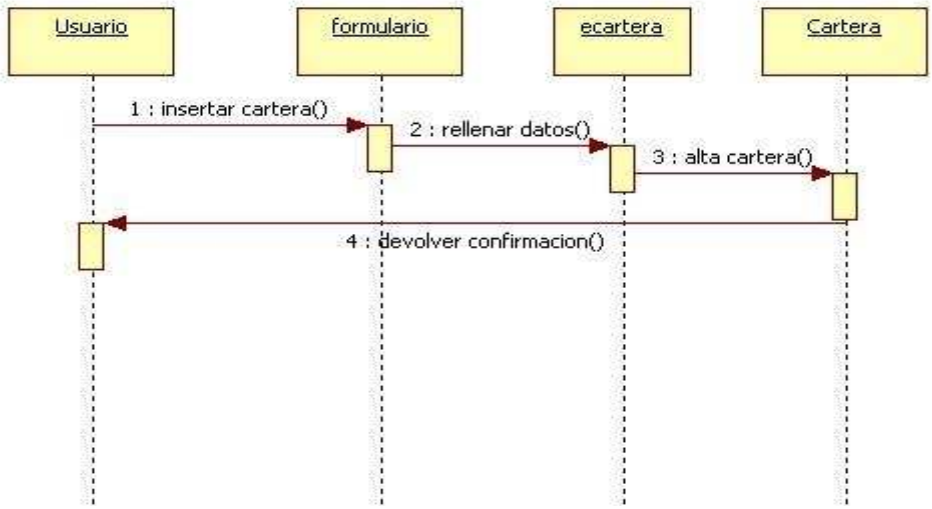
3.4. Diagramas de secuencia

Muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. El diagrama de secuencia tiene pequeños detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes o avisos que se producen en la interacción.

Se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario.

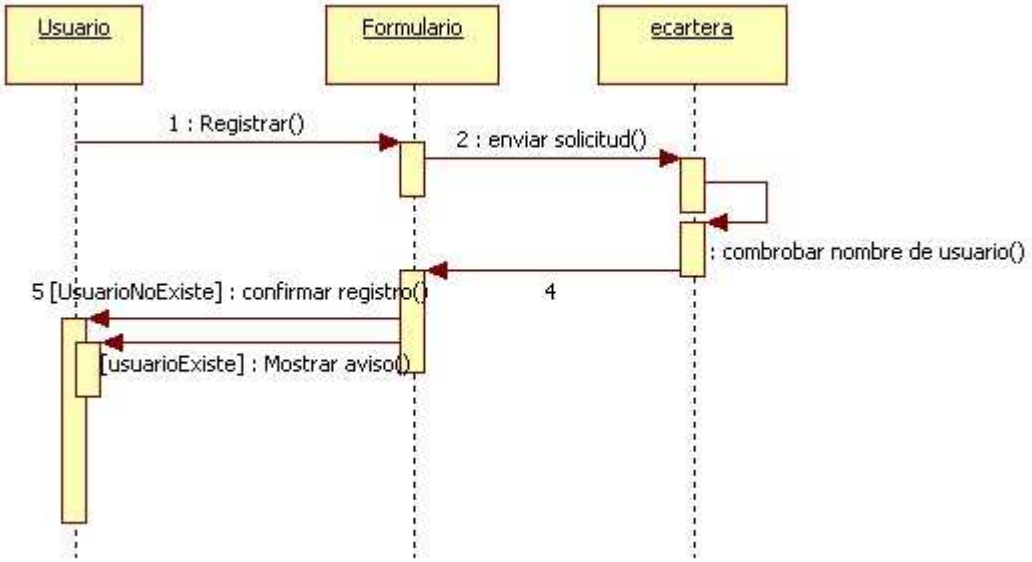
Alta Cartera

En este escenario se pide los datos en un formulario de alta de una cartera. Nombre, descripción, y si será principal. El sistema rellena los datos dando de alta una nueva cartera y devuelve una confirmación al usuario.



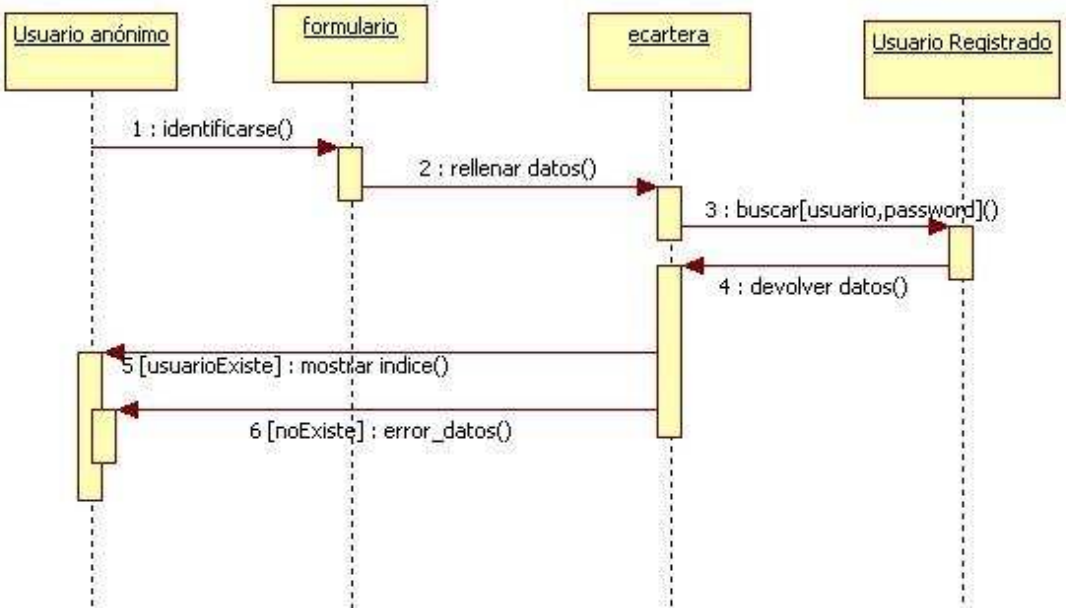
Registrarse

El formulario de registro sólo está accesible si el usuario no está autentificado. Comprueba que ningún dato esté en blanco y que el nombre de usuario no está ya registrado en la base de datos, puesto que es único. Si el nombre de usuario no está ocupado, devuelve una confirmación y crea el usuario. Si no, lanza un aviso.



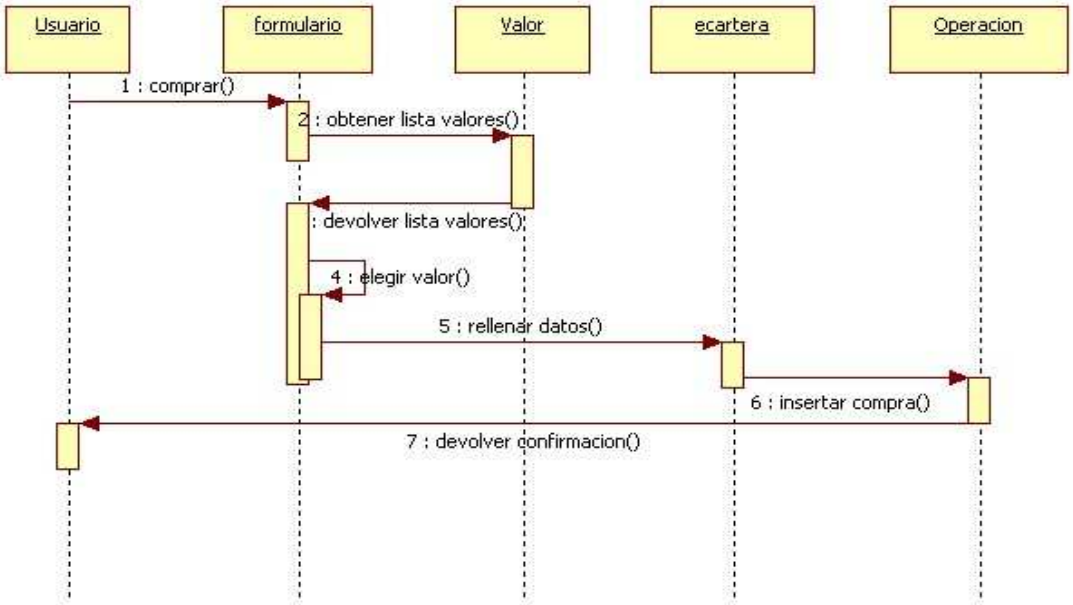
Login

La ventana de identificación es para los usuarios que previamente se han registrado. Con el nombre de usuario y el password podrá entrar a todo el menú de la aplicación web.



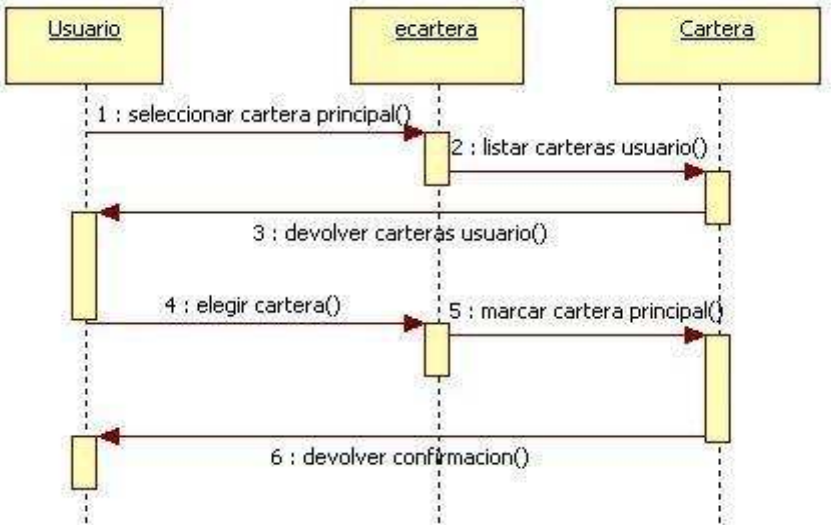
Comprar

Cuando el usuario abre el formulario para comprar, el sistema muestra todos los valores que existen. El usuario elige uno y rellena el resto de datos: Volumen, precio, fecha, etc. El sistema rellena los datos y genera una operación de compra. En el caso de una compra no hay comprobaciones, para otras operaciones sí según qué caso.



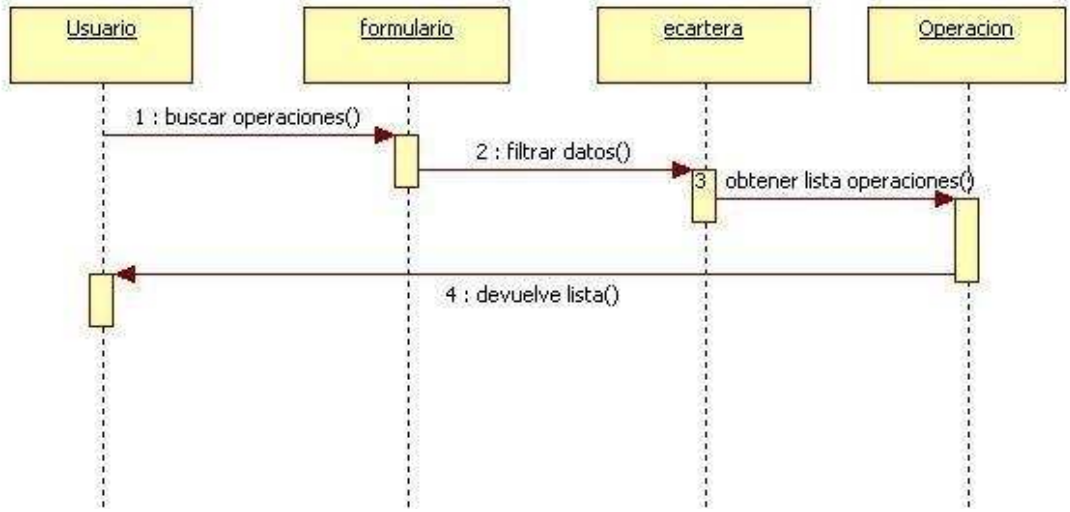
Seleccionar cartera principal

El formulario para seleccionar la cartera principal es muy sencillo. El sistema muestra primero al usuario todas sus carteras, indicando cual es la principal actualmente, si la hay. Entonces el usuario puede marcar otra cartera como la principal, y al darle al botón de "Aceptar", el sistema sustituirá la cartera principal. A partir de ahora el usuario trabaja con la nueva cartera seleccionada.



Filtrar Operaciones

El formulario de búsqueda de operaciones tiene cuatro filtros, fecha de inicio, fecha de fin, valor, y tipo de operación. Se pueden combinar para filtrar las operaciones deseadas por el usuario. Si no se marca nada, el sistema traerá todas las operaciones de la cartera principal.



4. Diseño

4.1. Introducción

Este capítulo se centrará en describir el proyecto con mayor nivel de especificación que en el capítulo anterior.

A partir de los modelos de los diagramas, casos de uso y demás elementos UML de la fase de análisis, se plantea como se lleva a cabo la implementación del producto pero sin entrar de lleno en las particularidades de una determinada tecnología. Ésta fase se abstrae aún de esos detalles.

Pasamos a describir con detalle la arquitectura que se ha construido para este proyecto, teniendo en cuenta el entorno Web y un esquema basado en cliente-servidor

4.2. Arquitectura de tres capas

La estructura de nuestra aplicación Web se basa en una arquitectura multicapa de tres capas.

Una arquitectura multicapa es un conjunto ordenado de subsistemas, cada uno de los cuales está constituido en términos de los que tiene por debajo y proporciona la base de la implementación de aquellos que están por encima de él.

Los objetos de cada capa suelen ser independientes, aunque pueden existir dependencias.

- Nivel de Interfaz, también llamada como capa de presentación, porque se encarga de la presentación de los resultados al usuario y la recogida de información que el usuario introduce.

- Capa de negocio o lógica de la aplicación, proporciona la funcionalidad de la aplicación. Se denomina capa de negocio o lógica de la aplicación a donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Aquí es donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de persistencia, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

- Nivel de almacenamiento o persistencia, este nivel es el encargado de que los datos persistan entre diferentes ejecuciones del

programa. Además de asegurar el acceso a la información de manera eficaz y segura. Este nivel lo componen el sistema de gestión de bases e datos y la base de datos en sí.

En nuestro proyecto sólo tendremos un servidor que hará a la vez de servidor de aplicaciones albergando la capa de negocio, y de servidor de base de datos.

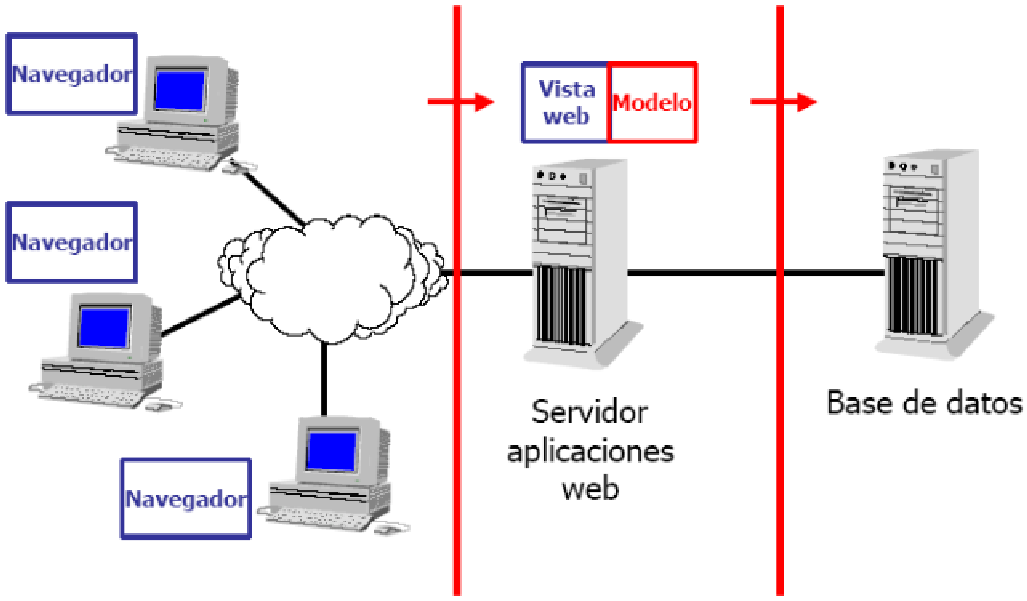


Figura 4-1. Arquitectura usada

4.2.1. Nivel de interfaz

La estructura de nuestro sitio web se compone de las siguientes partes:

Como se ha explicado en el capítulo anterior, tenemos dos usuarios y por tanto una vista distinta para cada uno.

En la vista del usuario registrado, llamada zona 1 en la Figura 4.2, aparece en la esquina superior izquierda, la zona del usuario. Esta zona tendrá un mensaje de bienvenida y el nombre de la cartera con la que estamos trabajando, es decir la llamada cartera principal del usuario. El usuario anónimo no dispone de esto como se puede ver en la Figura 4-3.

En la parte superior aparece el título de la aplicación.

Además, debajo del título, en la zona 4 del esquema, se encuentra el estado navegacional, es decir, una información de la ruta completa respecto al índice de dónde nos encontramos en el portal web. De gran ayuda para situar al usuario. Esto es igual en las dos vistas.

En la parte izquierda de la pantalla aparecerá el menú contextual de las opciones de la aplicación. Es la zona 3 de la Figura 4-2. El usuario anónimo sólo podrá ver dos: La página principal y el Ibex 35.

La zona 5 es la zona principal de la página. Es la zona más amplia, donde se mostrarán todos los resultados y la funcionalidad de las propias páginas web.

Por último, la zona 6, situada al pie de página, se han ubicado algunos enlaces a otra información como puede ser: publicidad, acerca de la empresa, contacta y datos legales.

Figura 4-2 Esquema de la vista del usuario registrado

Figura 4-3 Esquema de la vista del usuario anónimo. No aparece la zona de usuario y el menú sólo tiene las opciones disponibles.

Además, una última consideración: dentro del menú contextual del usuario registrado se distinguen dos casos. En caso de que el usuario no tenga una cartera principal asignada (caso que se puede dar o cuando el usuario es nuevo y no se ha creado ninguna cartera, o bien si borra su cartera principal sin asignar una nueva). En este caso, el usuario solo podrá ver la opción de gestión de carteras, es decir, alta, baja o modificación, y la zona de modificación de datos de usuario.

Si el usuario tiene una cartera principal asignada, podrá ver todo el resto de opciones: Todas las funciones de operar y todas las funciones de visualización de cartera.

A continuación se muestra una breve descripción de cada una de las opciones del menú:

Inicio: Ventana principal. Desde aquí el usuario anónimo puede identificarse, o bien registrarse si no lo ha hecho anteriormente.

Ibex 35: Desde esta ventana podremos ver los datos del selectivo español. Se podrán actualizar estos datos en cualquier momento para tener una información más actual. La aplicación se conectará a una página web externa y traerá los datos en tiempo real.

Cada valor del Ibex 35 dispondrá de un acceso directo a la interfaz de comprar o vender títulos, opción accesible sólo a usuarios registrados.

Operar:

Comprar. Desde la opción de comprar podremos comprar títulos eligiendo el valor que queremos, indicando el resto de parámetros: precio de compra, fecha, etc.

Vender: Lo mismo pero esta vez se indica que la operación es de compra.

Dividendo: Permite al usuario dar de alta una operación de dividendo.

Canje: Dar de alta una operación de canje.

Split: Dar de alta una operación de split.

Gestión de carteras:

Alta: Dar de alta una cartera. Tendremos la posibilidad de indicar que sea la principal, para trabajar con ella.

Baja: Dar de baja una cartera existente. Si se da de baja la principal, el usuario necesitará elegir otra para seguir operando.

Selección de cartera principal: Elegir la cartera con la que se desean asociar operaciones nuevas o bien para visualizar los datos de la cartera.

Visualización:

Estado actual: El usuario verá una relación de los títulos que tiene actualmente la cartera.

Operaciones: El usuario podrá ver el listado de operaciones de la cartera, y realizar búsquedas.

Información Fiscal: Informe de las ganancias y pérdidas patrimoniales.

4.2.2. Nivel de lógica de la aplicación

En el desarrollo de este proyecto se ha separado físicamente en ficheros la lógica de la aplicación con la de la interfaz. Se han definido los archivos .php en la carpeta "/logica" del proyecto, uno por cada vertiente. Por ejemplo "ValoresLogica.php", "operacionesLogica.php", etc. Cada uno corresponde a la lógica de los objetos correspondientes.

Las funciones de estos archivos sustituyen a los métodos que tendrían las clases correspondientes a los diagramas UML.

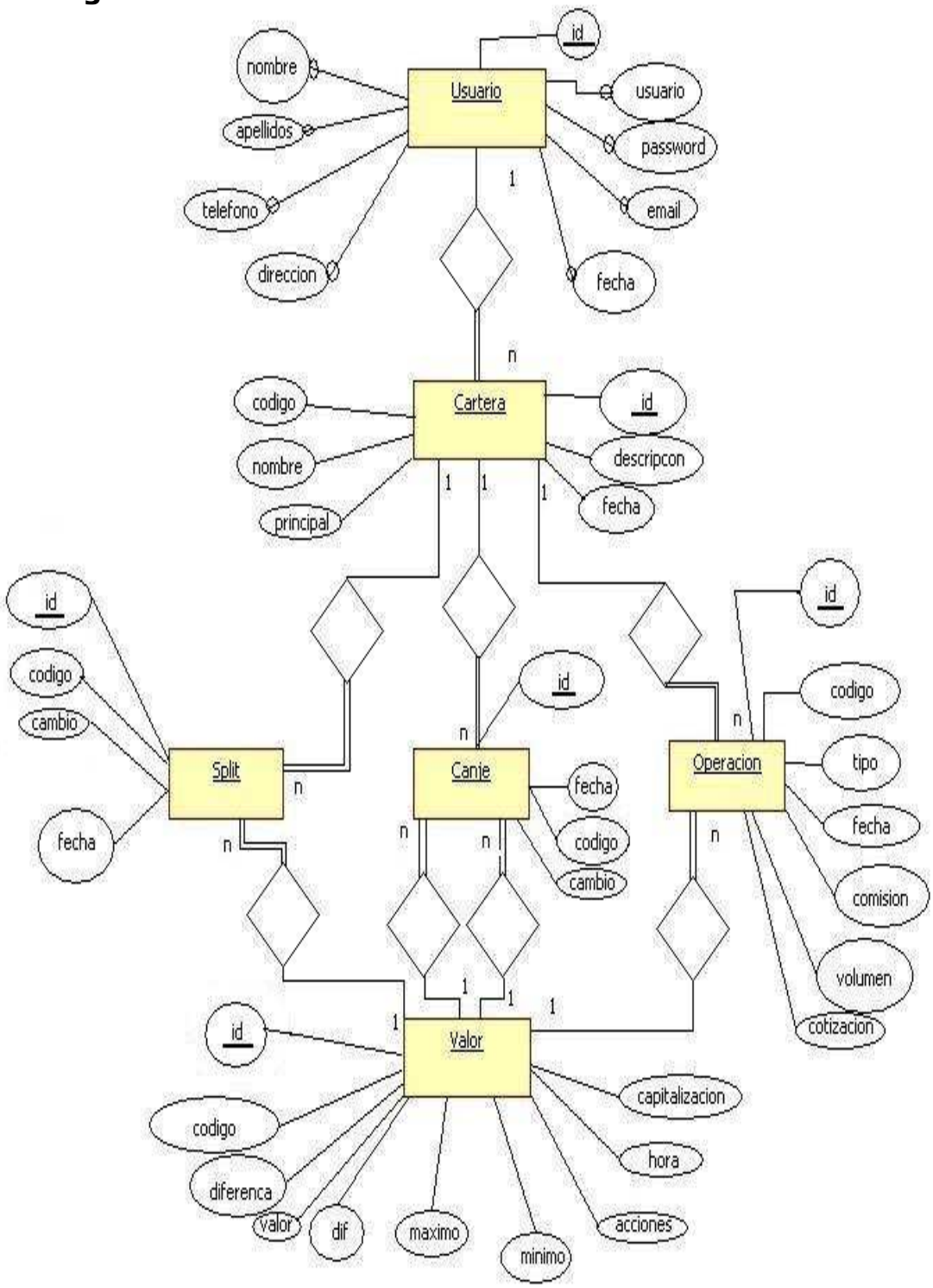
Su funcionalidad hará posible las operaciones descritas en apartados anteriores, es decir, conexión y manipulación de la base de datos, y presentar los resultados que la capa de interfaz recuperará.

Esta separación nos permite tener una independencia entre niveles, para que el mantenimiento y las posteriores modificaciones del proyecto puedan realizarse de forma muy cómoda, rápida y limpia incluso para otros desarrolladores.

4.2.3. Nivel de datos o persistencia

Este nivel está formado por un gestor de bases de datos con una base de datos relacional que realiza todo el almacenamiento de datos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de lógica.

4.3. Diagrama entidad relación



4.4. Diseño lógico

Usuario (id:int(10), nombre:varchar(80), apellidos:varchar(80), telefono:varchar(30), direccion:varchar(100), email:varchar(50), usuario:varchar(50), password:varchar(50), fecha: timestamp)

CP: {id}
VNN: {nombre, apellidos, telefono, direccion, email, usuario, password}

Valor (id:int(10), cod:varchar(20), valor: double, diferencia: double, dif: double, maximo:double, minimo: double, acciones: double, efectivo: double, capitalización: double, hora: timestamp)

CP: {id}
VNN: {cod, valor, hora}
UNI: {cod}

Cartera(id:int(10), codigo:varchar(50), nombre:varchar(80), comentarios: text, fechaAlta: timestamp, principal:tinyint(1), fkUsuario:int(10))

CP: {id}
VNN: {codigo, nombre, fechaAlta, fkUsuario}
UNI: {codigo}
CAj { fkUsuario } → Usuario

Operación(id:int(10), codigo:varchar(50), fkCartera: int(10), fkValor: int(10), tipo: enum(Compra, Venta, Dividendo), fecha: timestamp, comision:double, volumen: int(30), cotizacion:double)

CP: {id}
VNN: {codigo, fkCartera, fkValor, tipo, fecha}
UNI: {codigo}
CAJ {fkCartera} → Cartera
CAJ {fkUsuario} → Usuario

Canje(id: int(10), cambio: double, fecha:timestamp, fkValorF:int(10), fkValorD: int(10), fkCartera: int(10))

CP: {id}
VNN: {cambio, fecha, fkValorF, fkValorD, fkCartera}
CAJ {fkValorF} → Valor

CAJ {fkValorD} → Valor
CAJ {fkCartera} → Cartera

Split(id:int(10), cambio:double, fecha:timestamp, fkValor:int(10), fkCartera: int(10))

CP: {id}
VNN: {cambio, fecha, fkValor, fkCartera}
CAJ {fkValor} → Valor
CAJ {fkCartera} → Cartera

5. Implementación

Después de comprobar en capítulos anteriores cómo se permitía describir en los diversos niveles en que se puede descomponer un proyecto Web, este capítulo pretende mostrar la forma en que las tecnologías que han sido usadas intervienen en la implementación de estos niveles.

5.1. Tecnologías

Para poder realizar toda nuestra aplicación hemos utilizado distintas tecnologías y lenguajes de programación.

Para ello se ha optado por instalar WAMP Server 2.13.3, que se compone de las siguientes tecnologías:

- Apache versión 2.2.17 en versión Web

- PHP versión 5.3.5 lenguaje de programación interpretado.

- MySQL, 5.5.8 junto con phpmyadmin 3.3.9 que nos permite la creación y gestión de nuestras bases de datos.

- En el diseño de interfaces hemos utilizado el lenguaje HTML 4.0 y CSS.

- Javascript se ha utilizado para la comprobación de formularios, comprobar campos que se rellenen con espacios en blanco o vacíos, y además, cálculos sobre campos de texto del importe de las operaciones.

- Se ha utilizado jQuery 1.6.3, que es una biblioteca de JavaScript, para realizar la entrada de fechas por formulario de manera más fácil y vistosa. También para recargar datos sin tener que volver a recargar toda la página entera.

- amCharts. Librería gráfica de PHP que se ha utilizado para sacar el gráfico de desglose patrimonial en el menú de estado actual.

5.1.1. HTML

HTML, siglas de ***HyperText Markup Language*** («lenguaje de marcado de hipertexto»), es un lenguaje de marcación de elementos para la creación de documentos hipertexto, es el lenguaje predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares

(<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir scripts, el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

5.1.2. HTTP

Hypertext Transfer Protocol o **HTTP** (en español *protocolo de transferencia de hipertexto*) es el protocolo usado en cada transacción de la World Wide Web.

HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (usualmente un navegador Web) se lo conoce como agente de usuario A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones Web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

5.1.3. Arquitectura cliente – servidor

La **arquitectura cliente-servidor** es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que esperan peticiones y dan respuestas.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las

ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores Web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos a otros, la arquitectura básica es la misma. En este proyecto se usará un servidor Web.

La *arquitectura cliente-servidor* sustituye a la *arquitectura monolítica* en la que no hay distribución, tanto a nivel físico como a nivel lógico.

5.1.4. PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Aunque pede ser utilizado desde una línea de comandos, en este proyecto se usará para la interpretación del lado del servidor (*server-side scripting*).

Puede ser desplegado en la mayoría de los servidores Web y en casi todos los sistemas operativos y plataformas sin costo alguno. Es uno de los lenguajes más utilizados en la red, el cual nos permite la realización de páginas Web dinámicas de forma sencilla y potente.

Cuando el cliente hace una petición al servidor para que le envíe una web, el servidor ejecuta el intérprete de PHP. Éste procesa el script php solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente.

5.1.5 CSS

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web con un mínimo de complejidad.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML bien definidos y con significado completo (también

llamados *"documentos semánticos"*). Además, mejora la accesibilidad del documento, reduce la dificultad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para *marcar* los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

5.1.6. jQuery

jQuery es una biblioteca o framework de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Es un software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2.

Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript, que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

5.1.7. MySQL

MySQL, *Structured Query Languaje*, es un sistema de gestión de base de datos relacional multihilo y multiusuario. Las bases de datos relacionales almacenan los datos en tablas separadas y proporciona velocidad y flexibilidad.

Las tablas pueden estar enlazadas mediante relaciones, por tanto es posible combinar datos de varias tablas cuando realizamos una consulta.

Este *software* es de código abierto, *open source.*

5.2. Herramientas

Las herramientas utilizadas en el desarrollo de este proyecto han sido:

- WAMP versión 1.7.3, funcionando bajo Windows XP Service Pack 3.

-Con *NetBeans 6.9.1* se ha construido toda la parte de programación HTML, PHP, CSS y JavaScript.

-Para la creación del título, botones y retocar imágenes se ha utilizado el Gimp.

Respecto a la gestión de la base de datos se ha utilizado phpmyadmin. Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web.

Se pueden crear y eliminar Bases de Datos, crear, eliminar y alterar tablas o campos y ejecutar cualquier sentencia SQL.

-StarUML: Es una herramienta para el modelado de software basado en los estándares UML y MDA (Model Driven Arquitecture), que en un principio era un producto comercial y pasó a ser uno de licencia abierta GNU/GPL.

Con esta herramienta se han desarrollado los casos de uso, los diagramas de secuencia y el diagrama de clases.

5.3. Detalles de la implementación

5.3.1. Perfiles de usuario

Perfil de usuario anónimo

Este perfil es el que aparece siempre que la persona inicie la aplicación.

Sólo puede realizar las acciones de consulta de las cotizaciones del Ibex 35 y su actualización, además de contar con la posibilidad de registrarse o identificarse.

A pesar de que es deseable que este perfil de usuario pueda interactuar más con la aplicación, para familiarizarse con ella, su funcionalidad está limitada puesto que se necesita que el usuario esté registrado para poder operar carteras o visualizar datos, trabajando sobre una cartera creada.

Una vez esté registrado, la opción de registro desaparecerá.

Perfil de usuario registrado

Si el usuario se ha registrado, podrá realizar todas las funciones del proyecto descritas en este documento mediante una identificación.

5.3.2. Objetos: interfaz más manejable

El proyecto ha sido desarrollado con una pequeño enfoque orientado a objetos. Se ha definido un objeto para cada tabla de la base de datos. En vez de construir la conexión, la consulta y recorrer las filas con el puntero fetch de jdbc, lo que se hace es llamar a un método de una clase php, por ejemplo, "usuarioUtils.class.php". Dicha clase contiene todos los métodos que están relacionados con el objeto "Usuario" y que realizarán las operaciones en la base de datos. Estos métodos devolverán el objeto o una lista de objetos, con todos sus datos listos para ser "usados" con una simple llamada al método get() del atributo.

Las clases de utilidades se encuentran en la carpeta "logica" del proyecto, y se puede considerar como una subcapa dentro de la lógica de la aplicación, para comunicarse con la capa de persistencia. El resultado es que el código queda mucho más limpio, conciso, intuitivo y manejable. Con esto, me he centrado mejor a la hora de desarrollar el diseño de la Interfaz de usuario.

Aparte de que el código queda mucho más reutilizable y ligero, porque hay métodos que son llamados desde distintas interfaces, también vendrá bien en el supuesto caso de que haya mejoras en la aplicación.

5.2.3. Persistencia de sesión del usuario

Se manejan las sesiones de usuario del navegador para que la aplicación pueda mantener la coherencia de los datos del usuario desde que éste se identifica hasta que desconecta.

Con las sesiones también garantizamos restringir el acceso a la información de cualquier usuario de la base de datos a personas no autorizadas que intenten introducir a mano los parámetros mediante la URL o mediante otras artimañas.

Si la sesión con el identificador de usuario está rellenada, el usuario está conectado. Sino, se redirige a la página principal. También se podría poner un aviso de acceso no autorizado, etc.

```
<?php
session_start();
$idu = -1;
if (isset($_SESSION['idusuario']) != 0) {
   $idu = $_SESSION['idusuario'];
   $ncartera = $_SESSION['ncartera'];
   $nusuario = $_SESSION['nusuario'];
} else {
   header('Location: /ecartera/index.php');
}

?>
```

5.2.4. Login

En el fichero login.php se pasan los parámetros Usuario y contraseña por el formulario y se recogen. Primero, se comprueba si existe el usuario con la función "existeUsuario". Si no existe, mostramos un error. Si existe, se comprueba que ese usuario coincida con el password. Si coincide, el usuario pasará a estar identificado, dejará de ser anónimo, modificando las variables de sesión a través de otras dos funciones que traen, el nombre de usuario a partir de su identificador, y su cartera principal.

```
 <?php
if (isset($_POST['enviar'])) {

 include('conexion.inc.php');
 include('logica/usuarioUtils.class.php');
 include('logica/carteraUtils.class.php');

 $usuario = $_POST['usuario'];
 $passw = $_POST['password'];

 $conexion = conectarse();
 $existe = existeUsuario($conexion, $usuario);

 if($existe == true){
 $idu = login($conexion, $usuario, $passw);

 if ($idu == -1) {
  $error = "Error, contraseña incorrecta.";
 } else {
  $_SESSION['idusuario'] = $idu;
  $_SESSION['nusuario'] = getNombreUsuarioById($conexion, $idu);
```

```
   $cartera = getObjetoPrincipalCarteraDeUsuario($conexion, $idu);
   $_SESSION['ncartera'] = $cartera->getNombre();
   header('Location: /ecartera/index.php');
  }

  }else{
   $error = "Error, usuario inexistente.";
  }
  cerrarConexion($conexion);
}
echo $error ?>

          <form action="login.php" method="POST">

            <label>Usuario: </label><input type="text" name="usuario" value="" /><br>
            <label>Contraseña: </label><input type="password" name="password"
value="" /><br>

            <input class="buttonForm" type="submit" name="enviar" value="Login" />
          </form>
```

Y estas son las funciones “login” y “usuarioExiste” de la clase de utilidades.

```
<?php

function login($conexion, $nick, $pass) {
 $consulta = "SELECT * from Usuario where usuario='" . $nick . "' AND password='" .
$pass . "'";

 $res = mysql_query($consulta, $conexion) or die(mysql_error());
 $numres = mysql_num_rows($res);

 $id = -1;
 if ($numres > 0) {
  $row = mysql_fetch_assoc($res);
  $id = $row['id'];
 }
 return $id;
}
?>
```

5.2.5. Zona Usuario

Este código es una plantilla que será llamada desde todas las páginas de la aplicación. Es una pequeña región de la página en el marco superior izquierdo, que te indica en todo momento la cartera actual con la que estamos trabajando.

```
<!-- Zona Usuario-->
<div id ="usuario">
   <?php
   if ($idu != -1) {
      echo "Bienvenido, " . $nusuario . ". <br> ";
      $var;
      if ($ncartera != "") {
         $var = " La cartera actual es " . $ncartera;
      } else {
         $var = "No hay cartera seleccionada";
      }
      echo $var;
   ?>
      <br><a href="/ecartera/logout.php">Desconectarse</a>
<?php
   }
?>
</div>
<!-- FIN Zona Usuario-->
```

5.2.6. Alta Usuario (JavaScript)

Esta es la parte php de la función de alta de usuario. Se recogen los parámetros y se llama a la funcion "altaUsuario". Si tiene éxito el usuario se ha insertado y se redirige a la página principal. Si el usuario existe, muestra una advertencia.

Los campos son comprobados previamente mediante JavaScript. Se comprueba campo a campo porque queremos que ante un fallo en el registro, indique los campos no válidos en rojo, En este caso se ha asignado al atributo "class" del formulario el texto "textFieldFallo", para ser referenciado desde la hoja de estilo.

```
<?php
        if (isset($_POST['enviar'])) {

          include('conexion.inc.php');
          include('logica/usuarioUtils.class.php');
//recogida de parametros
```

```
            $nombre = $_POST['nombre'];
            $usuario = $_POST['nick'];
            $passw = $_POST['password'];
            $email = $_POST['correo'];
            $direccion = $_POST['direccion'];
            $telefono = $_POST['telefono'];
            $apellidos = $_POST['apellidos'];

            $conexion = conectarse();

            $exito = altaUsuario($conexion, $nombre, $apellidos, $telefono, $direccion,
$email, $usuario, $passw);

            if ($exito == true) {
              echo "Alta de usuario " . $usuario . " realizada con éxito. " .
              " Puedes autentificarte en la página de  "; ?><a
href="/ecartera/login.php">Login</a>
          <?php
              } else {
              echo "<div class='warning'>Alta de usuario " . $usuario . " rechazada. Éste
nombre de usuario ya está creado<div>";
            }

            cerrarConexion($conexion);
          }
          ?>
```

Código PHP

```
function comprobarCampos(){
      var size = 0;
      if(document.alta.nombre.value==""){
        size++;
        document.alta.nombre.className = "textFieldFallo";
      }
      if(document.alta.apellidos.value==""){
        size++;
        document.alta.apellidos.className = "textFieldFallo";
      }
      if(document.alta.nick.value==""){
        size++;
        document.alta.nick.className = "textFieldFallo"; }
      if(document.alta.password.value==""){
        size++;
        document.alta.password.className = "textFieldFallo";
      }
      if(document.alta.correo.value==""){
        size++;
        document.alta.correo.className = "textFieldFallo";
      }
```

```
if(document.alta.direccion.value==""){
  size++;
  document.alta.direccion.className = "textFieldFallo";
}
if(document.alta.telefono.value==""){
  size++;
  document.alta.telefono.className = "textFieldFallo";
}
if(size > 0){
  var res = "Todos los campos son obligatorios para el registro";
  alert(res);
  return false;
}else{
  return true;
} }
```

Código JavaScript

```
.textFieldFallo{
 background-color:#ffcc99;
}
```

Código CSS

5.2.7. Gráficos con amCharts: Estado actual

AmCharts es una librería para php gratuita con la que se pueden construir gráficos en la Web de manera sencilla. En realidad utiliza php y javascript.

Mediante un JavaScript, se instancia un objeto AmCharts. Se llamará a los distintos métodos y atributos del objeto para definir las propiedades del gráfico, como el color, título, el radio, el ángulo tridimensional, etc. En nuestro caso el objeto es "AmPieChart()", que representa un gráfico de tipo circular.

La línea importante es ' chart.dataProvider = chartData ' , este atributo contiene los datos del gráfico. Mediante php he generado el ChartData.

Para mostrarlo por pantalla, se necesita llamar al método de la clase "write", se pasa por parámetro el identificador del elemento DIV que contendrá el gráfico, con esto se relaciona el gráfico con el contenedor.

```
<?php
$data = " var chartData = [";

        for ($i = 0; $i < count($codigosParaGrafico); $i++) {
```

```
            $data .= "{accion:'" . $codigosParaGrafico[$i] . "', valor:'"
                    . $importesParaGrafico[$i] . "'},";
            }
            $data .= "];";
            //hay que quitar la ultima coma
            str_replace(",]", "]", $data);
            cerrarConexion($conexion);
            ?>

        <script language="SCRIPT" type="text/javascript">
<?php
            echo $data;
?>
  window.onload = function() {
    chart = new AmCharts.AmPieChart();
    chart.dataProvider = chartData;
    chart.titleField = "accion";
    chart.valueField = "valor";

    chart.depth3D = 10;
    chart.angle = 10;
    chart.innerRadius = "30%";
    chart.startDuration = 0;
    chart.labelRadius = 15;
    chart.colors =
["#FFFF00","#0404B4","#FF0000","#2EFE2E","#81F7F3","#190710"];
    chart.write("grafico");

  }
          </script>

          <!-- Este div contiene al gráfico, flotar a la derecha -->
          <div id="grafico" style="width:600px; height:400px;"></div>
          <!-- End main column -->
        </div>
```

5.2.8. JQuery: Selector de fecha

A continuación un ejemplo de jQuery ui, un selector de fecha y hora. Básicamente se utiliza un campo de texto que al seleccionar la fecha con el componente, la recoge en formato texto. La necesidad de importar un componente tan especializado es porque es necesario que en ciertas operaciones a muy corto plazo quede reflejada la hora de la operación.

Para transformar el campo de texto, simplemente es importar los ficheros javascript y la hoja de estilo que nos proporciona la librería. Después, en Javascript hay que indicar el atributo id del campo de texto, así:

“ $('#fecha').datetimepicker() “
En este caso, el campo de texto tiene como identificador “fecha”.

```
<script type="text/javascript" src="jquery-1.6.1.js" ></script>
<script type="text/javascript" src="jquery-ui-1.8.13.custom/js/jquery-ui-1.8.13.custom.min.js"></script>
<script type="text/javascript" src="jquery-ui-1.8.13.custom/development-bundle/ui/jquery-ui-timepicker-addon.js"></script>
<script type="text/javascript" src="jquery-ui-1.8.13.custom/development-bundle/ui/jquery.ui.datepicker-es.js"></script>
<link type="text/css" href="jquery-ui-1.8.13.custom/css/ui-lightness/jquery-ui-1.8.13.custom.css" rel="Stylesheet" />
<link rel="stylesheet" type="text/css" href="CSS/estiloLayout.css">
<link rel="stylesheet" type="text/css" href="CSS/base.css">
<link rel="stylesheet" type="text/css" href="CSS/estiloErrores.css">
```

Código html

```
<script type="text/javascript">
   $(document).ready(function () {
      $('#fecha').datetimepicker();
      $('#fecha').datetimepicker('option', {dateFormat: 'yy/mm/dd'});
   });
</script>
```

Código JavaScript

Figura 5-1. Selector de Fecha con hora y minutos

6. Evaluación y pruebas

En este capítulo, una vez finalizada la fase de implementación del proyecto, es necesario evaluar los resultados y comprobar los diversos aspectos que caracterizan la aplicación además de considerar su puesta en marcha y el mantenimiento que debe hacerse a posteriori.

6.1. Evaluación

Para evaluar correctamente la aplicación, debemos centrarnos tanto en el funcionamiento como en la navegabilidad, asegurándonos, además, de la compatibilidad de dicha aplicación en otros navegadores. Para ello se deben seguir una serie de pasos, tales como ponernos en el lugar del usuario y pensar en cómo desearíamos encontrar la información si fuésemos dicho usuario, así como tener en cuenta dificultades con las que puede encontrarse este usuario (el ancho de banda, escasez de conocimientos informáticos, etc.), y también es importante tener un buen diseño visual que logre atraer la atención de los posibles usuarios.

Por ello, tras seguir estos pasos, se ha intentado lograr una aplicación que resulte, a primera vista, sencilla y atractiva para que dichos usuarios encuentren los menores problemas posibles a la hora de utilizarla.

6.2 Pruebas

6.2.1. Pruebas de Validación HTML y CSS

Todos los archivos HTML tanto CSS de la aplicación se han comprobado con el validador Web del W3C y se han corregido los errores encontrados, excepto un archivo CSS. El validador además comprueba si hay enlaces rotos.

Con el archivo base.css encuentra dos errores y es porque tiene propiedades –webkit, una para redondear las esquinas del menú (radius) y otra para el gradiente.

Las páginas Web se han diseñado en HTML 4.01 estricto.

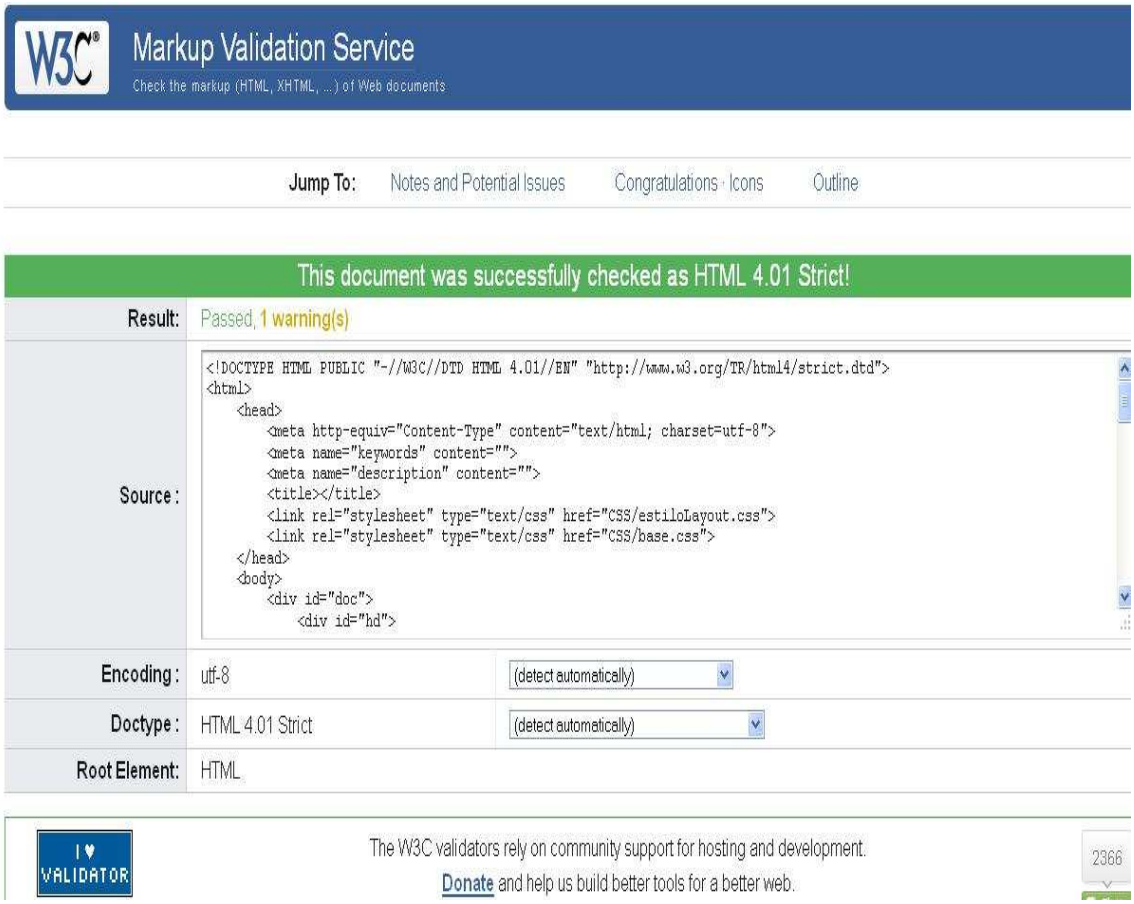


Figura 6-1. Validación de un archivo HTML

6.2.2. Pruebas de compatibilidad en navegadores

Se han pruebas en los siguientes navegadores:

-Mozilla Firefox 6.0.2:

Es el navegador predeterminado con el cual se ha diseñado la aplicación, aparece todo correctamente.

-Opera 11.51:

La anchura de las listas desplegables (comboBox) no es la correcta. Coge la anchura de la opción de la lista más larga, sin hacer caso a la hoja de estilo.

Los bordes de las filas de las tablas no aparecen. Además, las opciones del menú no aparecen con los cantos ligeramente redondeados como toca.

-Google Chrome 14.0.835.186:

Pasa lo mismo con las listas desplegables que en Opera.

-Safari 5.1:

Pasa lo mismo con las listas desplegables y los bordes de las tablas tampoco aparecen. El resto todo bien.

-Microsoft Internet Explorer 8:

Este navegador además de hacer lo mismo con las listas desplegables y no redondear los bordes, tampoco reconoce el gradiente del menú contextual. Por lo demás todo bien, se comporta mejor de lo que esperaba.

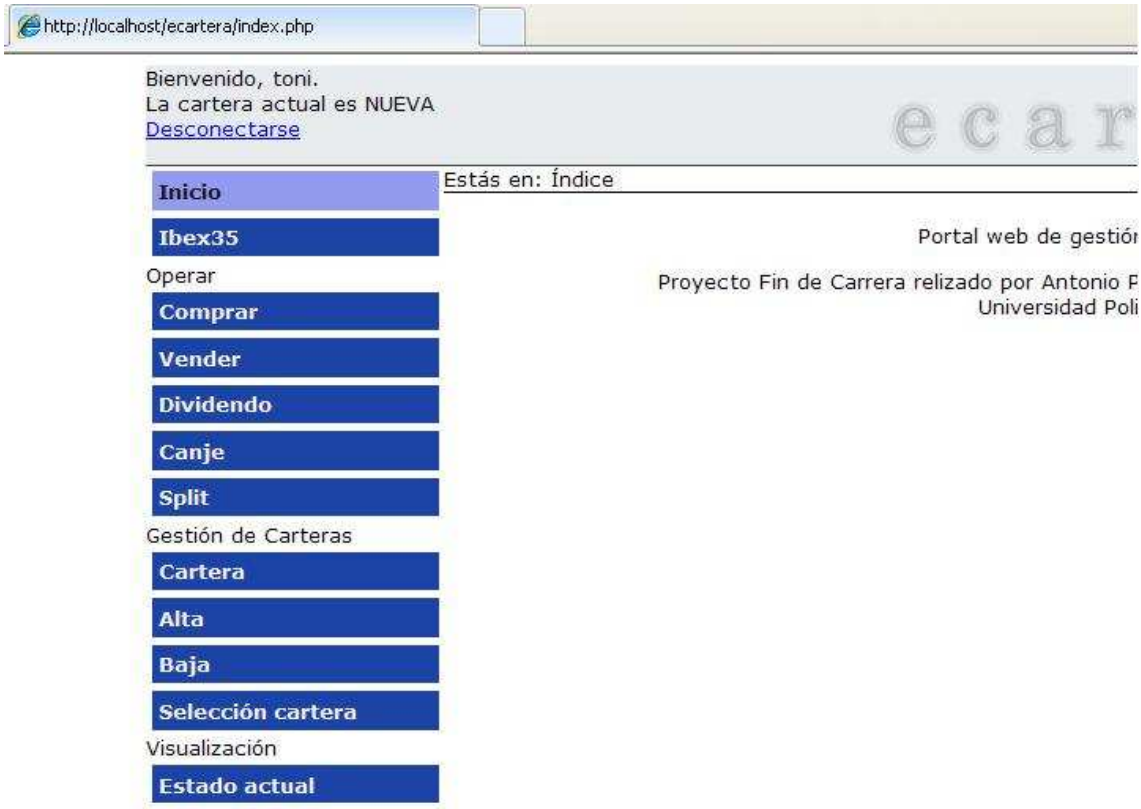


Figura 6-2. Imagen tomada desde Microsoft Internet Explorer 8

6.2.3. Pruebas de compatibilidad de resolución

Aunque la aplicación se ha probado en diferentes resoluciones y funciona correctamente, no se recomiendan resoluciones menores a 1024 x 768. Con resoluciones mayores no se presentan diferencias. A partir de esta resolución o menor, las tablas de información Fiscal o de operaciones que son las tablas más anchas, a veces descuadran

del ancho del contenedor general de la página, pero no es nada importante ya que son unos pocos píxeles.

Tampoco se recomiendan estas resoluciones tan bajas porque se ve todo muy grande y aunque podemos disminuir el tamaño de la visualización con la rueda del ratón, sigue siendo muy engorroso, como vemos en la Figura 6-3, ejemplo de resolución a 800 x 600.

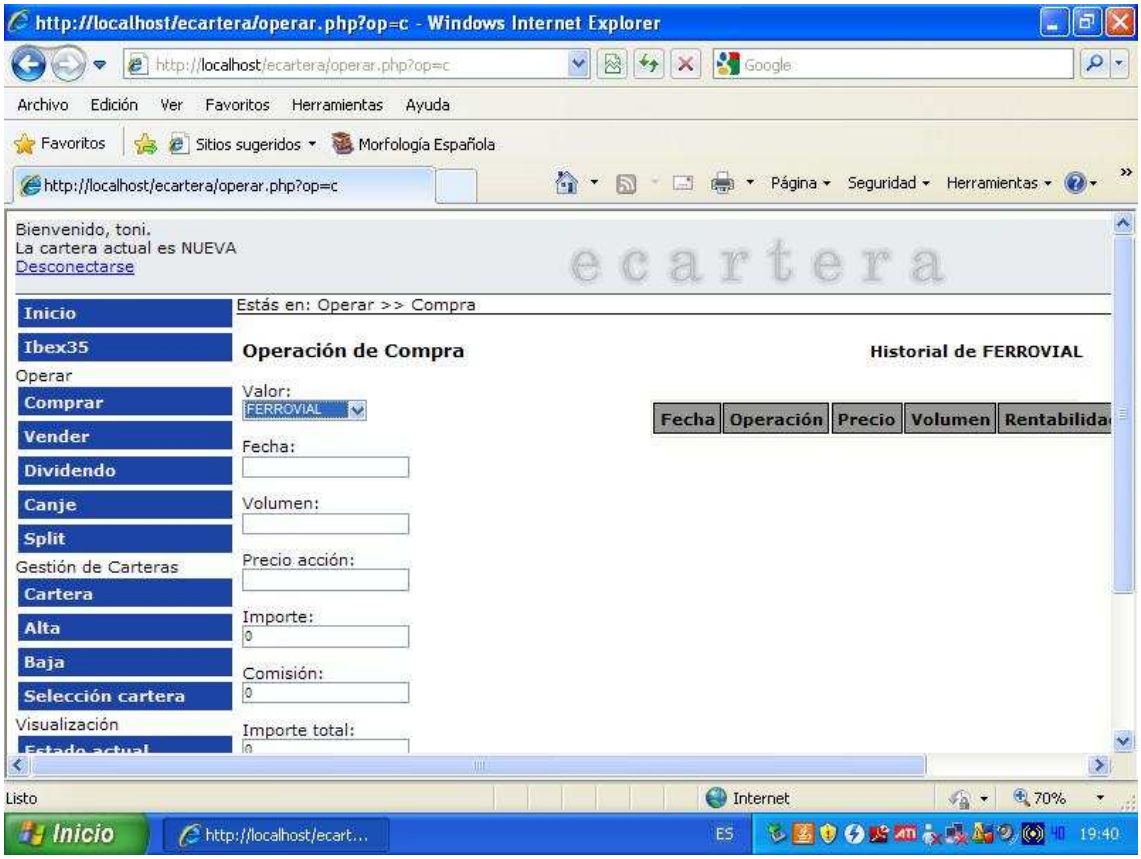


Figura 6-3. Visualización con resolución 800 x 600 en IE8

6.2.4. Pruebas unitarias

En este apartado probaremos individualmente y una por una todas las distintas funcionalidades de la aplicación para comprobar que funcionan correctamente. Probaremos cada funcionalidad en todas las diversas situaciones que se puedan dar, y además, simularemos posibles errores del usuario en la introducción y flujo de datos. La aplicación debe de estar correctamente preparada para cualquier eventualidad, mostrando un mensaje de aviso si no está preparada para ello o hay algún tipo de error, explicándolo en un lenguaje natural y sugiriendo soluciones.

Por razones de extensión, vamos a plasmar sólo cuatro ejemplos de los casos de uso, paso a paso, para que a la vez que los probamos nos sirva como demostración o como manual de la aplicación. Los casos de uso elegidos son: Operar (Compra y venta),

estado actual, listado de operaciones y información fiscal.Representan los casos de uso más frecuentes en la operativa, lo que significa que es el flujo de datos más común.

- **Proceso de introducción de operaciones**

Una vez logueados en la aplicación, tenemos que asegurarnos sobre qué cartera vamos a trabajar. Se ha definido una cartera llamada "Demo". Una vez tengamos la cartera deseada, nos dirigimos al apartado del menú contextual "Comprar". Allí aparece el formulario de alta de la operación. Vamos a coger por ejemplo el valor "BBVA", pero puede valor cualquiera que no tenga operaciones antes.

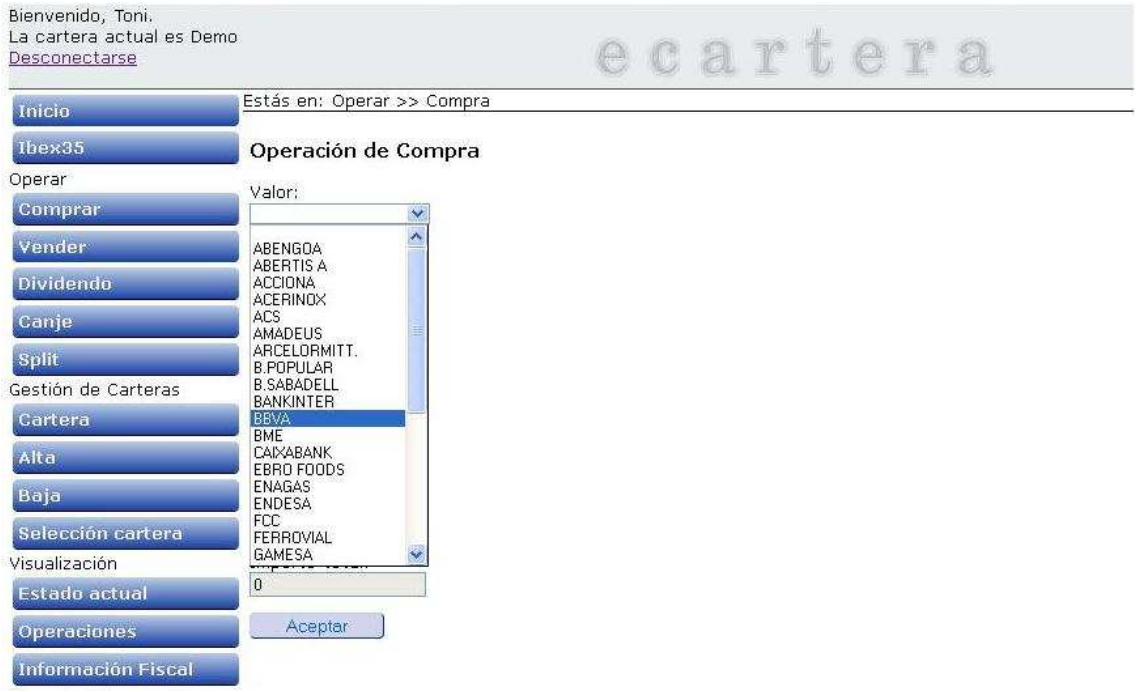


Figura 6-4. Introducción del valor

Indicamos Fecha 1 de Junio de 2011 a un precio de 9 euros la acción, un volumen de 200 acciones y aceptamos. Nótese que el sistema indica el importe, es decir lo que nos cuesta en total automáticamente. (9x200 = 1800 euros). Vamos a obviar las comisiones para simplificar, pero se pueden indicar para tener en cuenta, con lo que se restarían a los 1800 euros. Le damos a aceptar y compramos. Observar como se ha creado la compra en el historial reciente de BBVA, a la derecha.

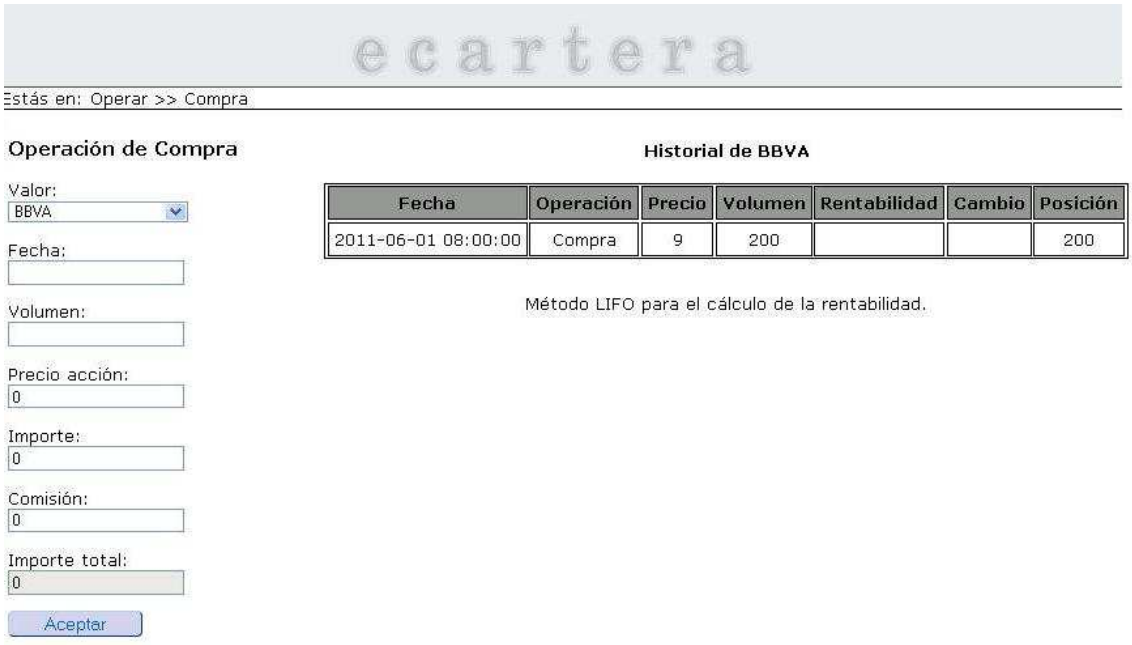


Figura 6-5. Historial BBVA tras la operación

Ahora nos vamos al apartado "Vender" del menú. Vamos a vender 100 acciones. Seleccionamos el valor "BBVA", a un precio de 7.5 euros la acción. Ponemos el 5 de Septiembre.

Tenemos 100 acciones de BBVA ahora mismo, vamos realizar 3 operaciones más:

Una compra de 50 acciones a 6 euros el 6 de Septiembre.
Una venta de 50 acciones a 8 euros el 9 de Septiembre.
Otra venta de 100 acciones a 9.5 euros el 18 de Septiembre, con lo que la cartera ya no tiene acciones de BBVA.

El historial de las operaciones se muestra en la siguiente imagen.

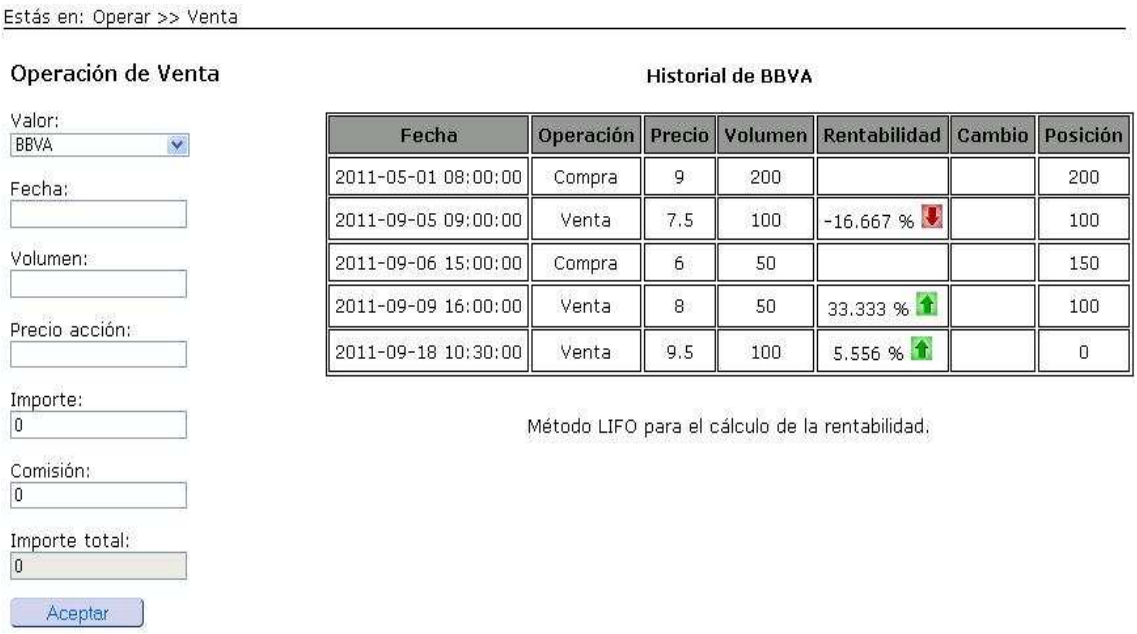


Figura 6-6. Historial BBVA

Una vez explicado el proceso de operar, vamos a realizar con otros valores más compras y más ventas:

Por ejemplo con el valor "ACS":

-Compra de 100 acciones a 30 euros el 5 de Septiembre.
-Venta de 100 acciones a 32 euros el 15 de Septiembre.
-Compra de 400 acciones a 28 euros el 19 de Septiembre.
-Venta de 200 acciones a 26 euros el 23 de Septiembre.

Figura 6-7. Historial ACS

Con el valor "ABENGOA":

-Compra de 100 acciones a 20 euros el 1 de Septiembre.
-Venta de 100 acciones a 18 euros el 5 de Septiembre.
-Compra de 100 acciones a 17 euros el 19 de Septiembre.

Figura 6-8. Historial ABENGOA

Para acabar, vamos a comprar acciones de Telefónica por un importe total de 6000 euros a un precio de 11 euros la acción. Simplemente se introduce el importe en el campo de texto, y el sistema genera las acciones que se pueden comprar con ese precio. En este caso 600/15 = 400. No las vamos a vender posteriormente, simulando una inversión a largo plazo.

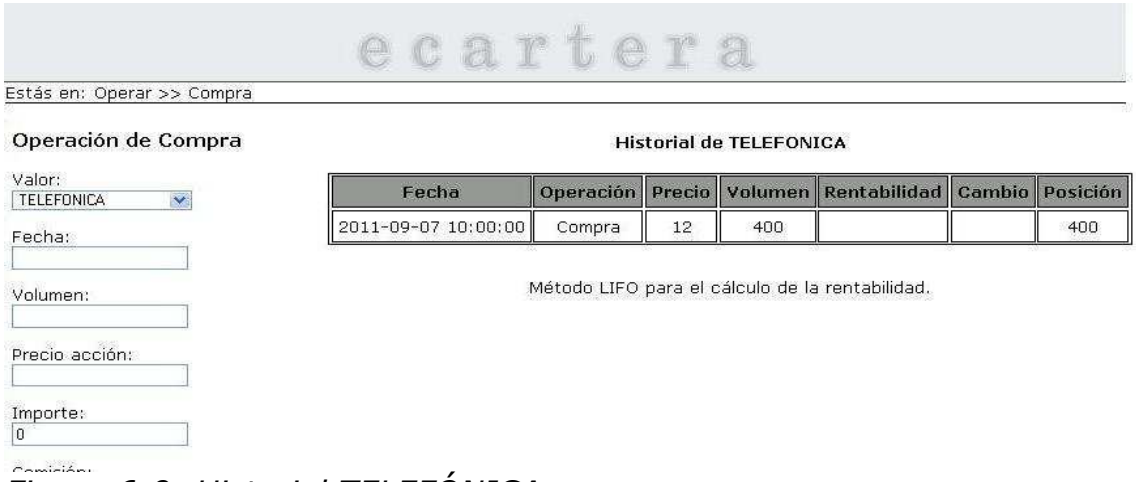


Figura 6-9. Historial TELEFÓNICA

- **Estado actual**

Vamos a ver cómo queda la cartera en conjunto. En el menú, seleccionamos "Estado actual".

Como se puede comprobar en la Figura 6-10, se muestran los títulos que la cartera tiene actualmente. De BBVA se vendieron todas, por tanto no aparece nada. Por otro lado, tenemos 400 acciones de Telefónica con un valor actual de 13.47 cada una lo que supone una rentabilidad casi el 11% respecto al precio de compra (12).

Tenemos 200 acciones de ACS. Recordar que se compraron 400 a 28 y luego se vendieron 200, así que las que quedan tienen un precio medio de 28. Ahora cotiza a 24.945 con lo cual estamos perdiendo dinero.

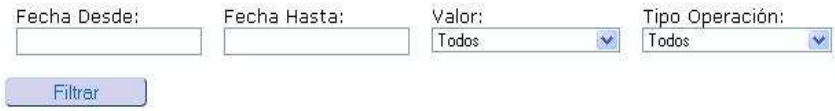
De ABENGOA quedan 100 acciones y también estamos perdiendo -173 € debido a que las acciones actualmente están más bajas.

Valor	Posición	Cotización actual	Valor Actual	Precio Medio	Ganancia	Rentabilidad
ACS	200	24.945	4989 €	28	-611 €	-12.25 %
TELEFONICA	400	13.47	5388 €	12	588 €	10.91 %
ABENGOA	100	15.27	1527 €	17	-173 €	-11.33 %
Total			11904 €		-196 €	-1.65 %

Figura 6-10. Estado actual de la cartera

- **Listado de operaciones**

Así es como queda el listado de las operaciones ordenadas por fecha. Se puede jugar con los filtros para encontrar la información más rápidamente.



Código	Tipo	Valor	Fecha	Cotizacion	Volumen	Comisión	Ec. canje	Valor absorbido
OP67	Compra	TELEFONICA	2011-04-01 10:00:00	12	400	0		
OP57	Compra	BBVA	2011-05-01 08:00:00	9	200	0		
OP69	Compra	ABENGOA	2011-09-01 09:00:00	20	100	0		
OP63	Compra	ACS	2011-09-05 09:00:00	30	100	0		
OP60	Venta	BBVA	2011-09-05 09:00:00	7.5	100	0		
OP70	Venta	ABENGOA	2011-09-05 10:00:00	18	100	0		
OP59	Compra	BBVA	2011-09-06 15:00:00	6	50	0		
OP61	Venta	BBVA	2011-09-09 16:00:00	8	50	0		
OP65	Venta	ACS	2011-09-15 09:00:00	32	100	0		
OP62	Venta	BBVA	2011-09-18 10:30:00	9.5	100	0		
OP64	Compra	ACS	2011-09-19 09:00:00	28	400	0		
OP71	Compra	ABENGOA	2011-09-19 10:00:00	17	100	0		
OP66	Venta	ACS	2011-09-23 09:00:00	26	200	0		

- **Información Fiscal**

Vamos a analizar cómo quedaría la información fiscal, las plusvalías reales de la cartera, y las pérdidas o ganancias patrimoniales que podemos imputar en la declaración de la renta.

Este informe agrupa las operaciones por valor, y está basado en las ventas. Para cada operación de venta, el sistema buscará con un método FIFO (Primero en entrar, primero en salir) las compras que le corresponde a esa venta. Es decir, que primero encontrará las compras más antiguas. Es importante destacar que se limitará al volumen de la venta. Es decir, si hay una compra de 200 y una venta de 100, se han vendido 100, así que los datos de la compra corresponderán sólo a 100 acciones, quedando otras 100 pendientes para futuras ventas. Si pasa esto en el campo "Tipo" aparecerá el texto: "C Par."

Hay que tener en cuenta la famosa norma "anti-aplicación". Si hay una venta con pérdidas y otra compra dos meses antes o después de la venta, se considerará como que se está recomprando y Hacienda no deja que esas pérdidas se puedan compensar con otras ganancias. Así la plusvalía será menos negativa o llegará ser 0 €. Ejemplos En la Figura 6-11, en ABENGOA y en la primera venta de BBVA.

Información Fiscal de la cartera

BBVA

Fecha	Tipo	Cod	Volumen	Precio Compra	Importe compra	Precio Venta	Importe Venta	Plusvalía	G/P Patrimonial
01-05-2011	C Par	OP57	100	9	900 €				
05-09-2011	V	OP60	100	9	900 €	7.5	750 €	-150 €	-75 €
01-05-2011	C Par	OP57	50	9	450 €				
08-09-2011	V	OP61	50	9	450 €	8	400 €	-50 €	-50 €
01-05-2011	C	OP57	50	9	450 €				
07-09-2011	C	OP59	50	6	300 €				
12-09-2011	V	OP62	100	7.5	750 €	9.5	950 €	200 €	200 €
	SUMA				750 €		2100 €	0 €	**75 €**

ACS

Fecha	Tipo	Cod	Volumen	Precio Compra	Importe compra	Precio Venta	Importe Venta	Plusvalía	G/P Patrimonial
05-09-2011	C	OP63	100	30	3000 €				
15-09-2011	V	OP65	100	30	3000 €	32	3200 €	200 €	200 €
19-09-2011	C Par	OP64	200	28	5600 €				
23-09-2011	V	OP66	200	28	5600 €	26	5200 €	-400 €	-400 €
	SUMA				5600 €		8400 €	-200 €	**-200 €**

ABENGOA

Fecha	Tipo	Cod	Volumen	Precio Compra	Importe compra	Precio Venta	Importe Venta	Plusvalía	G/P Patrimonial
01-09-2011	C	OP69	100	20	2000 €				
05-09-2011	V	OP70	100	20	2000 €	18	1800 €	-200 €	0 €
	SUMA				2000 €		1800 €	-200 €	**0 €**

Plusvalía Total	G/P Patrimonial Total
-400	-125

Figura 6-11. Información Fiscal de la cartera

En primer lugar, observar que no aparece Telefónica, pues no hemos hecho ninguna Venta.

Respecto a BBVA, recordemos que la primera compra es de 200 acciones y la primera venta es de 100 acciones, con lo que aún queda volumen de compra pendiente. La pérdida patrimonial es sólo de 75 € porque hay una compra de 50 acciones en los 2 meses siguientes, con lo que se considera una recompra a ojos de Hacienda. Como la recompra tiene la mitad del volumen de la venta, la mitad de las pérdidas no se imputan. (150 € / 2 = 75 €).

La segunda venta, de 50 acciones, se asocia aun a la primera compra (OP57) y finalmente la tercera venta de 100 "coge" los 50 restantes y otra compra de 50.

En cuanto a ACS, el ejemplo es claro. La última venta tiene una plusvalía negativa, por tanto una pérdida. En este caso sí se declara y es compensable, ya que no existen recompras anteriores ni posteriores.

No es así el caso de ABENGOA, que pasa lo contrario. Existe una compra posterior, que aunque esté pendiente de venta (aún las tenemos en cartera), hace que se considere recompra y la pérdida no sea compensable.

Finalmente muestra la plusvalía y la diferencia patrimonial total de la cartera.

7. Conclusión

7.1. Trabajo realizado

La aplicación desarrollada en este proyecto final de carrera, en general cumple con los requisitos establecidos.

Además, en el estado actual se ha implementado más funcionalidad que en los requisitos. Se muestra también el precio medio actual de las operaciones, que comparado con el precio de la cotización, muestra una revalorización de la cartera en el instante actual.

Por problemas de tiempo, la visualización de los beneficios de la cartera con el método LIFO no se ha desarrollado completamente como funcionalidad en un apartado independiente. Al final lo que se ha implementado es un historial de operaciones de cada valor, en los formularios de las operaciones. Éste historial recoge todas las operaciones y calcula, con el método LIFO, una rentabilidad, pero no muestra un informe completo con la suma de los importes totales para calcular pérdidas o ganancias, que es lo que ha faltado por desarrollar.

7. 2. Valoración personal

En primer lugar, la motivación que me ha llevado a escoger este proyecto ha sido su temática, que me resulta de gran interés. Por ello, durante todo el proceso he tenido un gran aliciente para desarrollarlo, ya que siempre me ha parecido interesante el mundo de la economía y, en particular, el mercado de valores, por lo que era una buena oportunidad para realizar por mi mismo un proyecto Web sobre el tema y, de esta forma, desarrollar una aplicación que me pueda servir a mí o a otra gente que esté interesada, y también adquirir más conocimientos sobre la operativa con acciones.

El desarrollo de este Proyecto Fin de Carrera ha sido para mí una experiencia muy positiva y totalmente enriquecedora, tanto a nivel personal y más todavía a nivel profesional y formativo.

He aprendido a utilizar varias tecnologías de desarrollo Web, por ejemplo con el lenguaje de programación PHP, nunca había desarrollado una aplicación completa y de este calibre en dicho lenguaje. En un principio pensé que me costaría poco adaptarme al lenguaje, pero la verdad es que, en el fondo me he llevado un poco de decepción. Hay que tener en cuenta que estoy acostumbrado a programar en Java y C#, lenguajes en definitiva muy tipados que son los que más se usan en la carrera. Pero en PHP por ejemplo cuando

vas a usar una variable y te equivocas escribiéndola en una letra, PHP define una nueva variable inicializada a cadena vacía, no muestra ningún error el entorno de desarrollo y es difícil de detectar. En definitiva, a este tipo de lenguajes cuesta adaptarse si no se está acostumbrado, pero en general me ha gustado la experiencia de aprendizaje y programación en los lenguajes interpretados, experiencia casi nula antes de este proyecto.

También ha sido de gran satisfacción ir viendo cómo se iban haciendo realidad una a una todas las funcionalidades que en un principio eran ideas y después fueron plasmadas en un documento de especificación de requisitos.

Algunas tareas difíciles fueron el script de recogida de datos de la página Web externa o la implementación de algunos tipos de operaciones. Lo más difícil y costoso fue el cálculo de la información fiscal de las carteras, puesto que parece un algoritmo sencillo pero a la hora de programarlo es realmente muy complicado.

También fue una tarea costosa la maquetación con las hojas de estilo, puesto que tengo muy poca experiencia y hay que saber utilizarlas, ya que al tener tantas propiedades, la mezcla de éstas puede conllevar un resultado inesperado. Lo he solucionado consultando numerosos tutoriales y libros en Internet sobre CSS y creo que el resultado final es de un diseño decente. Pienso que, en cierta manera, cuando maquetas una Web y quieres que quede bonita, se trata más de tener buen gusto antes que programar. En definitiva hay que tener *arte*.

En resumen, este proyecto me ha servido a conocer en primera persona, sobre los procesos de un desarrollo software, los recursos, tiempos, costes y problemas al igual que cualquier proyecto en el ámbito empresarial, que es en definitiva mi aspiración final y por la cual estudié esta carrera. He podido aplicar y profundizar en los conocimientos adquiridos en ella y también aprender nuevos. Ha sido en general un gran esfuerzo y una gran lección de aprendizaje del que estoy contento.

7.3. Posibles ampliaciones

En general el proyecto está bastante completo y es muy manejable respecto a la facilidad de uso.

Algunas mejoras que se podrían realizar son:

- Suscripción de acciones. Añadir un tipo de operación mediante la cual se introduzca un número de acciones y una fecha y se asocien a la cartera. Esto representa cuando una empresa

genera dividendos en forma de acciones, por ejemplo ampliaciones de capital. Aunque realmente es un dividendo, una buena opción sería que, internamente, la aplicación las tratara como splits, ya que el importe es en acciones, y no en efectivo, y así tributarían de la misma manera fiscal que las suscripciones, ya que éstas deben de imputarse fiscalmente como lo hacen los splits y canjes.

- Una ventana donde se pudieran borrar las operaciones o al menos modificar el volumen. Esto se haría para cubrir los casos donde el usuario se equivoca introduciendo datos. No es un caso muy común, pero ya que se trata de un portal de gestión, y no se podría hacer algo para modificar las operaciones. O quizá se podría hacer con una ventana de administración e incluir otro tipo de usuario: El administrador, que fuera él el que tuviera esta capacidad previa petición de los usuarios.

- En el formulario de Split, en vez de indicar la ecuación de canje con un número, poner 2 campos de texto, que se introduzca con 2 números por cuantas acciones recibes cada acción, por ejemplo: “2 acciones por cada 5”. Con esto simplificamos la entrada del dato al usuario.

8. Bibliografía

Enlaces:

- Desarrolloweb. jQuery.
http://www.desarrolloweb.com/articulos/componente-datepicker-jquery-ui.html
Accedido Julio 2011.

- Documentación amCharts
http://flex.amcharts.com/class_reference/com/amcharts/AmPieChart.html
Accedido Agosto 2011.

- Documentación CSS.
http://www.w3schools.com/css/
Accedido Septiembre 2011.

- Documentación PHP.
http://php.net/manual/es/
Accedido Septiembre 2011.

- Documentación XDebug.
http://wiki.netbeans.org/HowToConfigureXDebug#Overview
Accedido Septiembre 2011.

- Librosweb. CSS.
http://www.librosweb.es/css/
Accedido Agosto 2011.

- Librosweb. Javascript.
http://www.librosweb.es/javascript/
Accedido Agosto 2011.

- Librosweb. Tablas y CSS en general.
http://www.librosweb.es/css/capitulo10.html
Accedido Agosto 2011.

- Tutorial jQuery.
http://mundogeek.net/archivos/2010/04/21/tutorial-rapido-de-jquery/
Accedido Agosto 2011.

- WebEstilo. Uso de Sesiones PHP.
http://www.webestilo.com/php/php12b.phtml
Accedido Julio 2011.

Referencias físicas:

- BABIN, LEE, *Introducción a Ajax con PHP*. Anaya Multimedia, 2007.

- BUENDIA GARCÍA, FÉLIX, *Guía para la realización y supervisión de proyectos de final de carrera (PFC) en el ámbito de la web*. Valencia, Editorial UPV.

- GIL RUBIO, F. JAVIER, *Creación de sitios web con PHP5*. Madrid, McGraw-Hill
Interamericana de España, 2006.

- GLASS, MICHAEL, *Desarrollo web con PHP, Apache y MySQL*. Madrid, Anaya Multimedia, 2004.

- KABIR, MOHAMMED J., *La biblia de servidor Apache 2.* Anaya Multimedia

- MOTA HERRANZ, LAURA, *Bases de datos relacionales: teoría y diseño*. Universidad Politécnica de Valencia, 2003.

- NEGRINO, TOM, *JavaScript & Ajax para diseño web*. Pearson, 2007.

- ORÓS CABELLO, JUAN CARLOS, *Diseño de páginas Web con XHTML, JavaScript y CSS.* Ra-ma, 2008.

- RUMBAUGH, JAMES, *Lenguaje unificado de modelado, el manual de referencia*. Addison-Wesley, 2007.

- ULLMAN, LARRY; *PHP: guía de aprendizaje*. Pearson Educación, 2001.