UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Carrera de Ingeniero Informático

SISTEMA PARA LA GESTIÓN WEB DEL CATÁLOGO DE PRODUCTOS DE UNA EMPRESA

PABLO LÓPEZ LÓPEZ

Dirigido por: JESÚS HERRERA DE LA CRUZ

Supervisado por: Dra. BEATRIZ BARROS BLANCO

Curso 2005-2006 (convocatoria de Septiembre)

SISTEMA PARA LA GESTIÓN WEB DEL CATÁLOGO DE PRODUCTOS DE UNA EMPRESA

Proyecto de Fin de Carrera de modalidad *oferta general*

PABLO LÓPEZ LÓPEZ

Dirigido por: JESÚS HERRERA DE LA CRUZ (firma)

Supervisado por: Dra. BEATRIZ BARROS BLANCO (firma)

Tribunal calificador:

Presidente: D./D^{a}.....
(firma)

Secretario: D./D^{a}. .....
(firma)

Vocal: D./D^{a}. .....
(firma)

Fecha de lectura y defensa:.....
Calificación:.....

Índice general

Índice de figuras

Índice de cuadros

Parte I

Resumen e Introducción del Proyecto

Capítulo 1

Resumen

Resumen

El presente trabajo consiste en el diseño, desarrollo y prueba de una aplicación informática siguiendo las fases habituales de la Ingeniería del Software. La aplicación a desarrollar consiste en crear un sistema para la gestión del catalogo *Web* de una tienda de discos al que puedan acceder clientes, administradores, vendedores y gestores de almacén. Se evalúan diferentes metodologías de desarrollo, y se elige un modelo secuencial, el de ciclo de vida en cascada. Las fases de este modelo que se consideran adecuadas al presente trabajo son: Análisis de requisitos, Diseño, Implementación y Pruebas. Para cada una de estas fases se hace un estudio teórico previo de las distintas alternativas existentes, justificando la elección de las metodologías, modelos y tecnologías seleccionados.

Palabras clave

Ingeniería del Software, catálogo *Web*, modelo de ciclo de vida en cascada, análisis de requisitos, diseño, implementación, pruebas.

Abstract

This work involves the design, development and trial of a computer application following the normal phases of software engineering. It consists of creating a system for managing the *Web* catalogue of a record shop which can be accessed by clients, managers, sales and warehouse staff. Various development technologies are assessed and a sequential model is chosen: the Waterfall life cycle model. The model phases considered relevant to the current work are: analysis of requirements, design, implementation and trials.

A previous theoretical study of the various existing possibilities is carried out for each phase to justify the methodologies, models and technologies selected.

Key words

Software Engineering, *Web* catalogue, waterfall life cycle model, requirements analysis, design, implementation, trials.

Capítulo 2

Introducción

La presente memoria presenta el trabajo realizado por el alumno como Proyecto Fin de Carrera (PFC) de los estudios de Ingeniería Informática en la Universidad Nacional de Educación a Distancia (UNED).

En este capítulo se trata de situar el tema del Proyecto en un contexto adecuado para su realización.

El trabajo a realizar en el presente PFC consiste en desarrollar un sistema para la gestión *Web* del catálogo de productos de una empresa.

Como paso previo a la realización del trabajo se realizó un anteproyecto en el que se especificaban los objetivos que se pretendían conseguir, se definía claramente el enunciado del problema y se hacía un estudio preliminar de la metodología de desarrollo a utilizar así como de las distintas tecnologías existentes y la elección de las más adecuadas para el presente proyecto.

En el anteproyecto también se incluye un diagrama temporal de las actividades y costes de las mismas.

Los resultados de este estudio preliminar se incluyen en la memoria puesto que algunos análisis, como los relativos a las tecnologías utilizadas en la implementación, son suficientemente exhaustivos y no será necesario repetirlos en el desarrollo del proyecto.

Se hace un estudio sobre el estado actual de las técnicas y tecnologías relativas al modelado del desarrollo del software, de los fundamentos del análisis de requisitos y del diseño del software, y de los distintos métodos de prueba.

Se eligen las distintas metodologías, modelos, técnicas y notaciones que se utilizaran en las diferentes fases del proyecto.

Se desarrolla el proyecto siguiendo las fases habituales en Ingeniería del Software, para ello en primer lugar, como el tipo de empresa y el producto que oferta es de libre elección según la normativa del PFC, se determinará el problema a resolver especificando en lenguaje claro las necesidades y re-

querimientos de la empresa tal como una hipotética empresa encargaría el trabajo a un ingeniero del software.

Una vez obtenido el enunciado del problema a resolver el siguiente paso será elegir el modelo de desarrollo del software que mejor se adapte a dicho problema, para ello se analizan varios modelos de desarrollo eligiendo de entre ellos el que se considere que tiene las características adecuadas.

Los diferentes modelos de desarrollo incluyen una serie de fases, que suelen ser las mismas, diferenciándose un modelo de otro en el énfasis e importancia que da a las diferentes fases, así como al orden en que se ejecutan. Las fases mas importantes y que suelen estar comprendidas en todos los modelos de desarrollo son: estudio de viabilidad y análisis de requisitos, diseño, implementación, pruebas y validación, integración y mantenimiento.

En este proyecto no será necesaria la fase de mantenimiento puesto que con la entrega del trabajo termina la relación con la supuesta empresa.

Para las fases de análisis y diseño se discutirán diferentes metodologías, eligiendo la mas adecuada al problema en cada caso.

En el análisis se especificarán claramente y sin ambigüedades todos los requisitos y restricciones que debe cumplir el sistema y se definirán los modelos estáticos y dinámicos del mismo así como las estructuras de datos necesarias. Como resultado final se obtendrá un documento de especificación de requisitos. Es importante resaltar que el análisis deberá decirnos qué hace y qué no hace el sistema pero no cómo lo hace.

En la fase de diseño, partiendo de los modelos obtenidos en el análisis, se describe detalladamente cómo se llevan a cabo las funciones que debe realizar el sistema. Estas funciones se describen utilizando una notación que es independiente de cualquier lenguaje de programación.

Para la fase de implementación se hará un estudio de las diferentes tecnologías existentes para el desarrollo de una aplicación *Web*. Se analizarán sistemas de bases de datos, servidores *Web* y lenguajes de programación eligiendo la combinación de ellos que se crea más adecuada al problema a resolver.

Finalmente en la fase de pruebas se diseñarán las pruebas que se consideren necesarias para validar el sistema, es decir, para justificar que el sistema cumple los requisitos que se especificaron en la fase de análisis.

En los capítulos finales se repasa la historia del desarrollo del proyecto haciendo una estimación de las horas empleadas, se comenta brevemente cómo podría ampliarse el trabajo en un futuro, y se exponen las conclusiones finales.

Se incluyen una serie de anexos con el manual de usuario de la aplicación, el manual de instalación y la ubicación de los archivos del sistema en un sistema operativo Microsoft Windows®.

Capítulo 3

Estructura de la memoria

En el capítulo 4 (página 21) se muestran los objetivos que se pretenden alcanzar en el presente trabajo.

En el capítulo 5 (página 23) se encuentra una exposición detallada del problema a resolver.

En el capítulo 6 (página 25) se muestran los resultados obtenidos en el estudio preliminar del anteproyecto analizando los métodos de desarrollo del software y las distintas tecnologías implicadas.

En el capítulo 7 (página 53) se hace un estudio de los diferentes modelos de desarrollo del software. Se repasan los fundamentos del análisis de requisitos. Se analizan igualmente los fundamentos del diseño del software, y finalmente se hace una introducción de los diferentes tipos de pruebas que se pueden realizar a un producto software.

La sección 7.1 (página 53) se dedica a exponer los principales modelos de desarrollo del software y a justificar la elección de uno de ellos para el desarrollo del proyecto.

La sección 7.2 (página 62) estudia los fundamentos del análisis de requisitos. Se comentan los principios y tareas del análisis y los distintos modelos que se usarán para especificar el software de la aplicación.

En la sección 7.3 (página 70) se analizan los fundamentos del diseño del software. Se estudian los conceptos en que se basa el diseño. Se analizan los métodos de diseño orientado a flujo de datos y orientado a objetos. Se estudian el diseño de datos, arquitectónico, de interfaz y procedimental.

En el capítulo 8 (página 89) se seleccionan, justificadamente, los modelos, técnicas y notaciones que se utilizarán para el desarrollo del proyecto.

El capítulo 9 (página 95) está dedicado a la especificación completa del software de la aplicación. Se desarrollan los modelos de datos, funcional y de comportamiento, y se analizan los casos de uso del sistema.

El capítulo 10 (página 123) se dedica a la especificación completa del diseño de la aplicación. Se desarrollan los diseños de datos, arquitectónico, de interfaz y procedimental.

El capítulo 11 (página 155) se refiere a la implementación. Se indica qué software se utiliza, cómo hay que configurarlo para el correcto funcionamiento y dónde se ubican los archivos que son utilizados por el sistema.

El capítulo 12 (página 163) está dedicado a las pruebas del sistema. Se analizan las pruebas a realizar y el resultado de las mismas.

El capítulo 13 (página 169) recoge el ritmo de realización del proyecto. Se acompaña de una estimación del coste del proyecto en horas empleadas.

El capítulo 14 (página 171) se dedica a analizar como podría mejorarse la aplicación en el futuro. Se incluyen ideas que han surgido durante la realización del proyecto y que no estaban en la especificación inicial y otras que no son aplicables al trabajo actual debido a que requieren más medios y/o tiempo de trabajo.

En el capítulo 15 (página 173) se repasan básicamente los objetivos propuestos y se comenta su grado de satisfacción. Se comentan tanto aspectos positivos (puntos originales o más creativos) como negativos (objetivos incumplidos y dificultades encontradas).

El anexo A (página 177) incluye el manual de usuario de la aplicación, mostrando cómo se pueden utilizar las distintas funciones del sistema.

El anexo B (página 185) muestra el manual de instalación. Se indican las instrucciones para instalar, configurar y usar los distintos programas y paquetes software utilizados en la aplicación.

El anexo C (página 189) muestra los archivos utilizados en la aplicación.

Parte II

Anteproyecto

Capítulo 4

Objetivos

Los objetivos del presente proyecto son, ordenados cronológicamente:

- Definir el enunciado del problema tal como lo plantearía la empresa que hace el encargo del catálogo *Web*.
- Analizar diferentes modelos de desarrollo del software y elegir el que mejor se adapte al problema en cuestión.
- Elegir una metodología para la fase de análisis de requisitos.
- Analizar el sistema para obtener el Documento de Especificación de Requisitos.
- Elegir una metodología para la fase de diseño.
- Diseñar el sistema especificando cómo se deben realizar las distintas funciones necesarias para satisfacer los requisitos.
- Hacer un estudio de las distintas tecnologías existentes y elegir las más adecuadas para implementar la solución del problema.
- Desarrollar un prototipo que realice las funciones requeridas por el sistema.
- Diseñar y realizar las pruebas necesarias para comprobar la funcionalidad del sistema y que sirvan para validarlo frente a los requisitos.

Capítulo 5

Definición del problema

La empresa E se dedica a la venta de discos y tiene una decidida voluntad de tener presencia activa en la *Web*, así como de aprovechar este proceso de informatización para actualizar tecnológicamente su modo de funcionamiento.

En este proceso de informatización la empresa pretende definir e implantar un dominio corporativo que dé servicio tanto a clientes como a gestores y empleados de la empresa. En este contexto se pretende desarrollar una solución que permita realizar la creación, consulta, modificación y mantenimiento del catálogo de productos de la empresa.

La empresa quiere tener la información de los discos disponibles en una base de datos a la que puedan acceder diferentes tipos de usuario con diferentes cometidos y permisos de acceso.

Los tipos de usuario y sus funciones o permisos son:

Cliente: Los clientes podrán realizar consultas sobre un disco o un intérprete. También podrán realizar reservas de discos.

Vendedor: Un vendedor puede consultar la BD[1] para ver de qué existencias se dispone y puede efectuar una venta, que en este caso consistirá en dar de baja en la BD el o los discos vendidos (no se considera ningún tipo de transacción económica).

Almacenista: La función del almacenista es dar de alta discos en la BD, cuando llegan a la tienda, y dar de baja discos cuando se retiran de la tienda.

Administrador: La función principal del administrador es asignar, y quitar, identificadores y contraseñas a los diferentes tipos de usuarios del siste-

[1] BD, Base de Datos

ma. Además los administradores tendrán acceso total, es decir pueden realizar cualquier acción permitida a cualquier tipo de usuario, es decir funciones de modificación y mantenimiento del catálogo de productos de la empresa.

La empresa quiere disponer de una única interfaz principal de acceso al sistema desde la cual los diferentes tipos de usuario se dirijan a sus respectivas interfaces.

Capítulo 6

Estudio preliminar

6.1. Objetivos

Los objetivos perseguidos en el estudio preliminar realizado en el anteproyecto son:

- Elegir la metodología de desarrollo que se seguirá en el proyecto. Los fundamentos teóricos de dicha metodología se analizarán posteriormente en el proyecto.
- Hacer un estudio de diferentes tecnologías *Web* y justificar la elección que se hace para el proyecto.

6.2. Método de desarrollo y elaboración

El ciclo de vida del desarrollo del software determina el conjunto de actividades que se han de realizar durante el proceso. El ciclo de vida determina el orden en que se realizan dichas actividades.

El ciclo de vida software es "el periodo que comienza cuando un producto software es concebido y termina cuando deja de estar disponible" (CC95). Un ciclo de vida se divide en varias fases.

Entre los distintos modelos de ciclo de vida, (modelo en cascada, modelo en V, uso de prototipos, modelo en espiral, incremental, evolutivo, etc...) se ha elegido el Ciclo de Vida en Cascada (ver figura 6.1) por ser el que se cree que mejor se adapta al problema a resolver, ya que se recomienda su uso cuando los requisitos son estables y estén bien comprendidos, el diseño y la tecnología esté probada y madura, la duración del proyecto sea relativamente corta y el cliente no necesite versiones intermedias, y todos estos supuestos se dan en este proyecto.

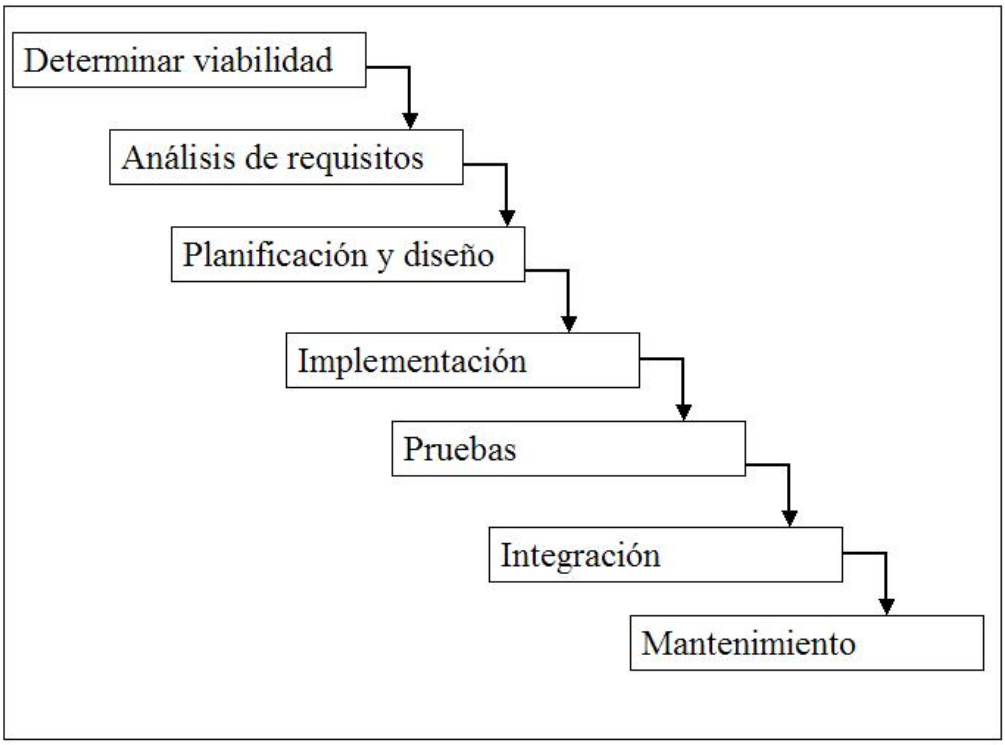

Figura 6.1: Ciclo de vida en cascada (CC95)

Se comentan a continuación las fases principales del modelo de ciclo de vida en cascada.

6.2.1. Viabilidad y análisis de requisitos

En esta fase se analiza si el proyecto es viable y se establecen los requisitos que debe cumplir el sistema, así como un modelo del mismo.

Con el análisis de requisitos se trata de caracterizar el problema a resolver (Pre97). El objetivo global del análisis es obtener las especificaciones que debe cumplir el sistema a desarrollar. El medio para lograr dicho objetivo es obtener un modelo válido y suficiente para recoger todas las necesidades y exigencias que el cliente precisa del sistema y, además, todas aquellas restricciones que debe verificar el sistema. Las especificaciones se obtendrán basándose en el modelo obtenido.

Para lograr una especificación correcta, el modelo deberá tener las siguientes propiedades:

- Completo conciso y sin ambigüedades.
- Sin detalles de diseño o implementación.
- Entendible por el cliente.

Además, deberá ser capaz de:

- Separar los requisitos funcionales y no funcionales.
- Dividir y jerarquizar el sistema.
- Fijar los criterios de validación.

Las tareas principales del análisis serán:

- Desarrollar un modelo del sistema.
- Elaborar un documento de especificación de requisitos.

6.2.2. Diseño

En esta fase se realiza el diseño de la aplicación determinando los diferentes módulos que incluirá y especificando a alto nivel las funciones que realizará el sistema.

Las actividades normales en el diseño de un sistema son (Pre97):

Diseño Arquitectónico Aborda aspectos estructurales y de organización del sistema y su posible subdivisión en subsistemas y módulos.

Diseño Detallado Aborda la organización de los módulos.

Diseño Procedimental Aborda la organización de las operaciones o servicios que ofrecerá cada uno de los módulos.

Diseño de Datos Aborda la organización de la base de datos del sistema partiendo de los diagramas entidad-relación de la especificación.

Diseño de la Interfaz de Usuario Aborda la organización de la interfaz de usuario.

Para el diseño se utilizará la metodología de Diseño Estructurado que utiliza diagramas de estructura. La tarea de diseño consiste en pasar de los Diagramas de Flujo de Datos obtenidos en el análisis a los diagramas de estructura, estableciendo una jerarquía o estructura de control que no está implícita en el modelo funcional.

El diseño de datos consistirá en diseñar una base de datos a partir del modelo entidad-relación.

6.2.3. Implementación e integración

En esta fase se traducen los módulos, funciones y procedimientos obtenidos durante el diseño a los lenguajes de programación elegidos para implementar la aplicación.

6.2.4. Pruebas

En esta última fase se realizan pruebas para comprobar el correcto funcionamiento de la aplicación.

6.3. Medios y tecnologías

En esta sección se introducen algunas de las tecnologías *Web* disponibles y se seleccionan la que se usarán en el Proyecto.

6.3.1. Sistemas operativos

Un sistema operativo (SO) es un conjunto de programas destinados a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera eficiente. Comienza a trabajar cuando se enciende el ordenador, y gestiona el hardware de la máquina desde los niveles más básicos.

El sistema operativo es una capa compleja entre el hardware y el usuario que facilita al usuario o al programador las herramientas e interfaces adecuadas para realizar sus tareas informáticas, abstrayéndole de los complicados procesos necesarios para llevarlas a cabo.

Los sistemas operativos desempeñan una serie de funciones básicas esenciales para la gestión de la máquina. Entre las más importantes podemos reseñar las siguientes:

- Gestión de los recursos de la máquina.
- Ejecución de servicios para los programas.

- Ejecución de instrucciones (comandos) de los usuarios.
- Gestión de procesos.
- Gestión de memoria.
- Gestión de la E/S (Entrada/Salida).
- Gestión de archivos y directorios.
- Comunicación y sincronización entre procesos.
- Seguridad y protección.

Las familias mas utilizadas de sistemas operativos son:

Microsoft Windows Familia de sistemas operativos desarrollados por la empresa de software Microsoft Corporation®. Todos ellos tienen en común el estar basados en una interfaz gráfica de usuario que utiliza el paradigma de ventanas.

Apple Macintosh Abreviado Mac es el nombre de una serie de ordenadores fabricados por Apple Computer® desde 1984. Los Apple Macintosh son comercializados con el sistema operativo Mac OS X. También es posible instalar en ellos Linux, y ahora existe la posibilidad de instalar Windows de forma nativa.

UNIX Sistema operativo portable, multitarea y multiusuario; desarrollado en los laboratorios Bell de ATT®, por Ken Thompson, Dennis Ritchie y Douglas McIlroy.

Desde el punto de vista técnico, UNIX se refiere a una familia de sistemas operativos que comparten unos criterios de diseño e interoperabilidad en común. Esta familia incluye más de 100 sistemas operativos desarrollados a lo largo de 20 años. No obstante, es importante señalar que esta definición no implica necesariamente que dichos sistemas operativos compartan código o cualquier propiedad intelectual.

Desde el punto de vista legal, Unix es una marca de mercado. Dicha marca es propiedad de "The Open Group", una organización de estandarización que permite el uso de dicha marca a cualquier sistema operativo que cumpla con sus estándares publicados. Resumiendo, la marca Unix no es propiedad de ninguna compañía.

Solaris Sistema operativo de la empresa Sun Microsystems® basado en el sistema UNIX. Quizá sea uno de los UNIX comerciales más usados, principalmente en el entorno Internet[1]. Proporcionó desde sus primeros momentos un excelente soporte para aplicaciones de red en protocolos IP[2], y fue el primer entorno donde se desarrolló el sistema Java. Proporciona prácticamente todas las funcionalidades típicas de los sistemas UNIX en entorno servidor. En los últimos tiempos la compañía ha puesto en marcha una clara estrategia de acercamiento entre Solaris y Linux desarrollando productos que permiten ejecutar programas de Linux en Solaris.

Linux Desde 1984, Richard Stallman y voluntarios están intentando crear un sistema operativo libre con un funcionamiento similar a UNIX, recreando todos los componentes necesarios para tener un sistema operativo funcional que se convierta en el sistema operativo GNU[3].

Una distribución Linux es un conjunto de aplicaciones reunidas que permiten brindar mejoras para instalar fácilmente un sistema Linux. Son variantes de Linux que, en general, se destacan por las herramientas para su configuración y los paquetes de software a instalar.

Existen numerosas distribuciones Linux. Cada una de ellas puede incluir, aparte del núcleo básico común a todas, diferentes módulos de software adicionales (libre o no), como los que facilitan la instalación del sistema, además de una enorme variedad de aplicaciones, entre ellas: entornos gráficos, suites ofimáticas, servidores *Web*, servidores de correo, servidores FTP, etcétera.

[1] Internet, es una red mundial de computadores interconectados usando un conjunto común de protocolos. Aparece por primera vez en 1960. Cuando se la denomina *red de redes* se hace referencia a que es una red formada por la interconexión de otras redes menores

[2] IP, Internet Protocol, protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

[3] El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completo libre: el sistema GNU.

GNU es un acrónimo recursivo que significa "GNU No es Unix".

El sistema GNU fue diseñado para ser totalmente compatible con UNIX. Esto implica que GNU está compuesto de pequeñas piezas individuales de software, muchas de las cuales ya estaban disponibles para UNIX.

Para asegurar que el software GNU permaneciera libre para que todos los usuarios pudieran "ejecutarlo, copiarlo, modificarlo y distribuirlo", el proyecto se hizo bajo una licencia diseñada para garantizar esos derechos, la Licencia General Pública de GNU (GPL).

6.3.2. Arquitectura

Se utilizará la clásica arquitectura CLIENTE - SERVIDOR.

La arquitectura cliente-servidor es una forma de dividir y especializar programas y equipos de cómputo a fin de que la tarea que cada uno de ellos realiza se efectúe con la mayor eficiencia, y permita simplificarlas.

En esta arquitectura la capacidad de proceso está repartida entre el servidor y los clientes.

Un servidor en informática o computación es:

- Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Este es el significado original del término. Es posible que un ordenador cumpla simultáneamente las funciones de cliente y de servidor.
- El ordenador en el que se ejecuta un programa que realiza alguna tarea en beneficio de otras aplicación llamada clientes.

El cliente recibe los servicios que ofrece un servidor. El término se usó inicialmente para dispositivos que no eran capaces de ejecutar programas por sí mismos, pero podían interactuar con ordenadores remotos por red.

Las ventajas de esta arquitectura son:

- El servidor no necesita tanta potencia de procesamiento, parte del proceso se reparte con los clientes. Se reduce el tráfico de red considerablemente.
- Idealmente, el cliente se conecta al servidor cuando es estrictamente necesario, obtiene los datos que necesita y cierra la conexión dejando la red libre para otra conexión.

6.3.3. Protocolos implicados

- HTTP[4] sobre TCP/IP[5] (puerto 80)
- HTTPS[6] sobre TCP/IP con SSL[7] o TLS (puerto 443)

Se utilizará el protocolo HTTP sobre TCP/IP que es el de uso mas extendido.

[4]HTTP, HyperText Transfer Protocol (protocolo de transferencia de hipertexto)

[5]TCP/IP, Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP)

[6]El protocolo HTTPS es la versión segura del protocolo HTTP

[7]Secure Sockets Layer (SSL) y Transport Layer Security (TLS) son protocolos criptográficos que proporcionan comunicaciones seguras en Internet.

6.3.4. Navegadores *Web* y tecnologías de programación en el cliente

Navegadores *Web*

- Internet Explorer de la compañía Microsoft®.
- Netscape Navigator de la compañía Netscape Communications®.
- Firefox es un navegador *Web* del proyecto Mozilla. Es software de libre distribución.

Se intentará que la aplicación funcione correctamente en los navegadores más extendidos en el mercado, que para el S.O. Windows son los anteriormente citados.

SGML

SGML[8] es un lenguaje de marcado genérico, sirve para especificar la estructura de cualquier documento sin tener en cuenta los aspectos relativos a la presentación. Un mismo documento se puede presentar de distintas maneras, de acuerdo a las normas de estilo que se apliquen.

SGML es un sistema para especificar lenguajes de marcado, es decir, un metalenguaje. Esto hace posible que, mediante la utilización de una definición de tipo de documento, DTD (Document Type Definition), se pueda especificar la estructura lógica de una clase de escrito.

Una DTD es una definición formal que indica qué elementos se incluyen como contenidos y en qué orden. Cada elemento en el documento se marca mediante una etiqueta de comienzo y otra de final. Estas etiquetas vienen especificadas mediante un identificador genérico, que define el tipo de elemento (párrafo, cabecera, etc.) y unas características, o atributos, que califican al identificador.

Una DTD considera un documento como un árbol, cuya raíz es el propio documento. Puede haber muchos documentos que cumplan la misma DTD, aunque no debería haber más de una DTD para un tipo de documento.

Cada lenguaje de formato de documentos definido con SGML se llama *aplicación SGML*, por ejemplo HTML es una aplicación SGML.

[8] SGML, Standard Generalized Markup Language (Lenguaje de Marcación Generalizado), sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares (ISO) normalizó este lenguaje en 1986.

HTML

HTML como lenguaje

HTML[9] es un lenguaje para la estructuración de textos, pero también existe la posibilidad de colocar imágenes o contenidos multimedia en forma de referencias e integrarlos en el texto (Mun01).

Con HTML se pueden crear cabeceras, párrafos de texto, listas y tablas. Se pueden crear vínculos a otras páginas *Web*. Se pueden integrar formularios en el texto. HTML ofrece interfaces para lenguajes suplementarios como Hojas de Estilo en Cascada (CSS) o JavaScript, con cuya ayuda se pueden formatear los elementos HTML o realizar interacciones con los usuarios.

HTML no es adecuado para trabajar con gráficos ni para acceder a bases de datos.

HTML como formato de texto claro independiente de software

HTML tiene un formato de texto claro. Se pueden crear archivos HTML con cualquier editor de texto. Por tanto no es necesario ningún software específico para la creación de archivos HTML. Aunque existen programas que están especializados en la edición de archivos HTML.

HTML como lenguaje de marcación

HTML es un lenguaje de marcación (Markup Language), está basado en el lenguaje SGML, es decir es una *aplicación SGML*.

Tiene la función de describir los componentes lógicos de un documento. Como lenguaje de marcación HTML ofrece la posibilidad de marcar elementos típicos de un documento, por ejemplo cabeceras, párrafos, listas, tablas o referencias de imágenes.

Todos los documentos de tipo HTML contienen los mismos elementos y los mismos atributos, es decir todos los documentos de este tipo tienen la misma estructura pero no los mismos contenidos. Todos los documentos que cumplen la norma HTML siguen la especificación de una DTD concreta que es interpretable por los navegadores. En el proceso de interpretación el navegador se encarga de transformar cada una de las marcas que definen la estructura del documento en una representación física que el usuario pueda comprender.

[9] HTML, HyperText Markup Language (lenguaje de marcado de hipertexto), actualmente es responsabilidad del World Wide Web Consortium, W3C (http://www.w3.org), que es una organización que produce estándares para la World Wide Web.

HTML para hipertexto

Una de las principales cualidades de HTML es la posibilidad de definir enlaces. Los enlaces ("hipervínculos") pueden conducir a otros lugares del mismo documento, pero también a otros documentos o a cualquier dirección en la World Wide Web.

Hojas de estilo en cascada

Las Hojas de Estilo en Cascada, Cascading Style Sheets (CSS)[10] nos permiten cambiar el formato de todos los textos de un sitio *Web* entero en pocos segundos, en vez de cambiar una hoja tras otra (Mun01).

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. Los lenguajes de presentación, CSS es uno de ellos, son aquellos que sirven para definir las características de presentación que finalmente tendrá la información contenida en los documentos.

Con HTML se define la estructura básica de las páginas *Web*, los elementos, los enlaces, los elementos referenciados. Sin embargo HTML no ha sido diseñado para especificar exactamente el aspecto que tiene un elemento, ni el tamaño, ni el tipo de fuente. Ese trabajo lo hace el navegador. Para ello utiliza una combinación de ajustes básicos que el usuario puede escoger, y la representación fija y programada de algunos elementos de HTML.

En este punto entran en juego las hojas de estilo en cascada (CSS). Las hojas de estilo son un lenguaje suplementario que fue especialmente desarrollado para HTML. CSS se adapta a HTML y permite dar formato a algunos elementos de este lenguaje.

CSS permite la definición de formatos centrales, por ejemplo para todas las cabeceras de primer nivel, o para todos los párrafos de texto con un determinado nombre de clase. Los formatos centrales se pueden referir a un archivo HTML, pero también pueden ser puestos en un archivo de estilo externo y después ser referido en los archivos HTML deseados.

CSS es, igual que HTML, un lenguaje de texto claro.

[10] CSS es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo.

XML

XML como lenguaje de definición para lenguajes de marcación

Con ayuda de XML[11] se pueden definir lenguajes de marcación, como HTML. XML es una aplicación de SGML, lo que significa que en su especificación se indica como se deben describir los elementos que participan en el documento pero no los elementos en sí. Cuando se quiere describir un documento mediante XML hay que describir en primer lugar el tipo de documento en que se basa, es decir la DTD, y a continuación los contenidos concretos asociados a cada elemento.

Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

Con XML se pueden definir elementos y sus propiedades fundamentales, pero no se puede especificar cómo un navegador debe formatear un elemento en el momento de la representación. Sólo por medio del formato de elementos se logra una representación útil en los lenguajes XML. Para ello existe, como en HTML, un lenguaje de estilo complementario. Este lenguaje es denominado XSL (eXtensible Stylesheet Language - lenguaje de estilo extensible).

Objetivos de XML

XML se creó para que cumpliera varios objetivos:

- Que fuera idéntico a HTML a la hora de servir, recibir, y procesar la información para aprovechar toda la tecnología implantada de este.
- Que fuera conciso desde el punto de vista de los datos y la manera de guardarlos.
- Que fuera extensible, para que lo puedan utilizar en todos los campos del conocimiento.

[11] XML, eXtensible Markup Language, (lenguaje de marcado extensible), es un lenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

- Que fuese fácil de leer y editar.
- Que fuese fácil de implantar, programar y aplicar a los distintos sistemas.

Ventajas de XML

Comunicación de datos. Si la información se transfiere en XML cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella.

Migración de datos. Si trabajamos en formato XML sería muy sencillo mover datos de una base de datos a otra.

Aplicaciones *Web*. Con XML hay una sola aplicación que maneja los datos y para cada navegador podemos tener una hoja de estilo o similar para aplicar el estilo adecuado.

Comparativa SGML, XML, HTML

XML es funcionalmente equivalente a SGML:

- es más sencillo
- no es tan potente:
  - uso obligatorio de comillas en los atributos
  - distingue mayúsculas de minúsculas en los nombres de los elementos
  - no permite restricciones en el anidado de los elementos
  - no permite la minimización de elementos
- tiene nuevas normas asociadas, algunas de lass cuales rompen la compatibilidad con SGML.

Tanto el XML como el HTML tienen su base en el SGML, el cual, es un metalenguaje que nos permite definir lenguajes para definir la estructura y el contenido de nuestros documentos. La definición de la estructura y el contenido de un tipo de documento se realiza en una DTD. En ella definimos los elementos que conformarán ese tipo de documentos y como tienen que estar organizados para que sea correcto.

HTML no es más que un tipo de documento SGML que se utiliza en la *Web*, y esto es importante, ya que aquí radica su principal diferencia con el XML.

XML no es ningún tipo de documento SGML, sino que es una versión abreviada de SGML optimizada para su utilización en Internet. Esto significa que con él vamos a poder definir nuestros propios tipos de documentos y, por tanto, ya no dependeremos de un único e inflexible tipo de documento HTML.

Richard Ligth escribió en su libro Presenting XML, "XML ofrece el 80 % de las ventajas del SGML con un 20 % de su complejidad". Y es que los diseñadores de XML intentaron dejar fuera sólo aquellas partes que raramente se utilizan. Esta reducción resultó ser muy importante: la especificación XML ocupa aproximadamente 30 páginas, frente a las 500 del SGML.

HTML es simplemente un lenguaje, mientras que XML es un metalenguaje, esto es, un lenguaje para definir lenguajes. Y esa es la diferencia fundamental.

En resumen, tanto SGML como XML son metalenguajes utilizados para definir lenguajes de marcación. XML es una aplicación SGML, es mas sencillo y menos potente, pero tiene algunas normas asociadas que les hacen incompatibles, es decir XML no puede ser definido usando SGML y viceversa.

Por su parte HTML es una aplicación SGML y se define usando este lenguaje. Asimismo HTML se puede definir usando XML como metalenguaje, de hecho la redefinición de la DTD de HTML usando XML ha dado origen a XHTML.

XHTML

XHTML[12], es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas *Web*. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML.

Su objetivo es avanzar en el proyecto del World Wide Web Consortium de lograr una *Web* semántica, donde la información, y la forma de presentarla estén claramente separadas. En este sentido, XHTML serviría únicamente para transmitir la información que contiene un documento, dejando para hojas de estilo y JavaScript su aspecto y diseño en distintos medios (ordenadores, PDAs, teléfonos móviles, impresoras...).

XHTML es una reformulación de HTML 4 como aplicación XML. Además XHTML permite la compatibilidad con los navegadores que ya admitían HTML 4.

[12]XHTML, eXtensible HyperText Markup Language (lenguaje extensible de marcado de hipertexto)

XSL

XSL[13], es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio específico.

El proceso de construcción consiste en dos pasos:

- En el primero se transforma el documento XML expandiéndolo y calculando los apartados dependientes del estilo (tipo de ordenación, inclusión de índices, etc.). Para realizar esta tarea se utiliza XSLT (siglas de eXtensible Stylesheet Language Transformations, lenguaje de hojas extensibles de transformación), que permite convertir documentos XML de una sintaxis a otra (por ejemplo, de un XML a otro o a un documento HTML).

- En el segundo paso, se prepara el documento para su presentación en el dispositivo de visualización (navegador, impresora, teléfono móvil, etc.) adaptándolo a las características propias del dispositivo.

JavaScript / Jscript

JavaScript[14] es un lenguaje de programación que se interpreta y se ejecuta en el cliente. Es muy útil para realizar tareas en el lado del cliente (navegador), como mover imágenes por la pantalla, crear menús de navegación interactivos, etc. JavaScript es aceptado por la mayoría de los navegadores (Mun01).

Páginas *Web* como aplicaciones

En HTML se puede, entre otras cosas, definir formularios. Tales formularios pueden contener campos de entrada, listas de selección, botones etc. El usuario puede llenar un formulario y enviarlo por la *Web* a un servidor. Sin embargo HTML no le permite al proveedor verificar los datos después de que el usuario haya llenado el formulario y antes de que éste envíe los datos. Otro ejemplo: aunque se puede en HTML enlazar un archivo multimedia, una vez

[13] XSL, eXtensible Stylesheet Language, expresión inglesa traducible como "lenguaje extensible de hojas de estilo"

[14] JavaScript fue inventado por Brendan Eich en la empresa Netscape Communications. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0. En junio de 1997 fue adoptado como un estándar ECMA, European Computer Manufacturers' Association, con el nombre de ECMAScript. Poco después también lo fue como un estándar ISO.

que el archivo esté en la *Web*, no se puede saber si el usuario posee un navegador que pueda visualizar el correspondiente formato multimedia. Lo práctico aquí sería hacer depender el enlace del archivo, con la posibilidad del navegador de visualizar el formato. Esto es posible hacerlo con JavaScript (Keo05).

Información sobre JavaScript

JavaScript al contrario que HTML, CSS o XML no es una tecnología independiente, sino un lenguaje de programación patentado por Netscape. Aunque JavaScript en el MS Internet Explorer funciona de la misma manera que en los navegadores de Netscape, en él se esconde en realidad otro lenguaje llamado JScript.

JScript es la implementación de ECMAScript de Microsoft, muy similar al JavaScript de Netscape, pero con ciertas diferencias en el modelo de objetos del navegador que hacen a ambas versiones con frecuencia incompatibles.

Para evitar estas incompatibilidades, el World Wide Web Consortium (W3C) diseñó el estándar Document Object Model (DOM, ó Modelo de Objetos del Documento en castellano). El DOM, es una forma de representar los elementos de un documento estructurado (tal como una página *Web* HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. En efecto, el DOM es una API[15] para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript.

VBScript

VBScript (Visual Basic Scripting) - La respuesta de Microsoft a JavaScript. VBScript es una buena herramienta para cualquier sitio destinado a ser mostrado exclusivamente en el navegador Microsoft Internet Explorer. Por este motivo discriminante y limitante, es preferible usar JavaScript por su amplia aceptación en los diversos navegadores de internet.

Applets Java

Java[16] y HTML

[15] API (Application Programming Interface - Interfaz de Programación de Aplicaciones)

[16] Java es un lenguaje de programación orientado a objetos desarrollado por James Gosling y sus compañeros de Sun Microsystems al inicio de la década de 1990. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es ejecutado por una máquina virtual Java.

Java es un lenguaje de programación, independiente de la plataforma, sobresale sobre todo por las siguientes características:

- Orientado a objetos: Java es estrictamente orientado en objetos y pone a la disposición del programador una amplia gama de objetos elementales y complejos.

- Archivos de programa independientes de plataforma: Los programas en Java son compilados en código de objeto como programas normales, pero no entrelazado con un determinado entorno de procesador o sistema operativo, es decir, se genera un código intermedio interpretable por una máquina virtual adecuada. Por eso, los programas en Java funcionan en todas las plataformas cuando está instalado un intérprete de código de objeto de Java. Los navegadores *Web* que ejecutan Java, arrancan su propia máquina virtual de Java para este fin.

Los Applets son programas escritos en Java diseñados para el uso en Internet. Los Applets se pueden referenciar en archivos HTML permitiendo que la aplicación se muestre en la página *Web*. Las interacciones entre el usuario y el programa tienen lugar en la ventana del navegador *Web*.

Los Applets son una forma especial de programas Java, están limitados en sus prestaciones para evitar accesos no permitidos al ordenador del usuario, se ejecutan en un área de seguridad.

Dentro del modelo cliente-servidor los programas Java pueden aparecer en ambos lados. Los Applets, que son ejecutados en el navegador del usuario, están en el lado del cliente. Pero los Applets se comunican a menudo, a través de sus propios protocolos, con programas en un servidor. Generalmente detrás de Applets Java se esconden aplicaciones distribuidas que consisten de un applet y los programas correspondientes del servidor.

Componentes ActiveX en Visual C++, Visual Basic o .NET

ActiveX es una tecnología introducida por Microsoft para ejecutar códigos de programas en páginas *Web* y la pretensión de ser una alternativa o competencia para Java. ActiveX es una definición general para diferentes componentes de software. Todos esos componentes se basan en el llamado Component Object Model (COM), que es una plataforma para componentes de software introducida por Microsoft en 1993. Esta plataforma es utilizada para permitir la comunicación entre procesos y la creación dinámica de objetos, en cualquier lenguaje de programación que soporte dicha tecnología.

Importantes componentes de ActiveX son los llamados controles ActiveX. Tales controles son programas o módulos de programas que se pueden poner en archivos HTML, de forma semejante a los Applets Java. El código del programa es ejecutable en la memoria principal del ordenador del usuario (o sea del usuario que llama la página *Web*). Con ActiveX es posible realizar toda clase de aplicaciones. No existen normas fijas sobre en qué lenguaje de programación debe ser escrito el código ActiveX.

ActiveX sólo puede ser directamente ejecutado por el MS Internet Explorer. Actualmente está en discusión el concepto de seguridad de ActiveX. Si un usuario permite que un control ActiveX sea cargado en su ordenador, entonces ese programa puede hacer lo que quiera. No existe ninguna restricción para los comandos de ActiveX, sino tan sólo una "barrera de confianza" (el mensaje del navegador preguntando si el usuario está de acuerdo con la ejecución del programa en su ordenador). Desde el punto de vista de la programación se pueden realizar muchas más cosas que con Applets Java, sin embargo desde el punto de vista del usuario los módulos de ActiveX son mucho más inseguros que los Applets Java.

6.3.5. Servidores *Web* y tecnologías de programación en el servidor

Servidor *Web*

Un servidor *Web* es un programa que implementa el protocolo HyperText Transfer Protocol (HTTP). Este protocolo está diseñado para transferir lo que llamamos hipertexto: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonido.

Cabe destacar el hecho de que la palabra servidor identifica tanto al programa como a la máquina en la que dicho programa se ejecuta. Existe, por tanto, cierta ambigüedad en el término.

Un servidor *Web* se mantiene a la espera de peticiones HTTP llevadas a cabo por clientes HTTP que solemos conocer como navegadores. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita. A modo de ejemplo, al teclear *www.unapagina.htm* en nuestro navegador, éste realiza una petición HTTP al servidor de dicha dirección. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo muestra en pantalla. Como vemos con este ejemplo, el cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Sobre el servicio *Web* clásico podemos disponer de aplicaciones *Web*. Éstas son fragmentos de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

- Aplicaciones en el lado del cliente: el cliente *Web* es el encargado de ejecutarlas en la máquina del usuario. Son las aplicaciones tipo Java o JavaScript: el servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta. Es necesario, por tanto, que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones (también llamadas scripts). Normalmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje JavaScript y Java, aunque pueden añadirse más lenguajes mediante el uso de plugins[17].

- Aplicaciones en el lado del servidor: el servidor *Web* ejecuta la aplicación; ésta, una vez ejecutada, genera cierto código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP.

  Al ejecutarse la aplicación en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad adicional, como sí ocurre en el caso de querer ejecutar aplicaciones JavaScript o java. Así pues, cualquier cliente dotado de un navegador *Web* básico puede utilizar este tipo de aplicaciones.

  Por otra parte no conviene sobrecargar al servidor, por ello se delega al cliente todo aquello que razonablemente puede hacer. En mi opinión, actualmente se prefiere un sistema mixto cliente-servidor, aunque el servidor realiza la mayor parte del cómputo.

Los principales servidores *Web* son:

- **Internet Information Server (IIS)**, también conocido como Internet Information Services, es una serie de servicios para los ordenadores que funcionan con Windows. Los servicios que ofrece son: FTP[18],

[17] Un plugin (o plug-in) es un programa de ordenador que interactúa con otro programa para aportarle una función o utilidad específica, generalmente muy específica. Este programa adicional es ejecutado por la aplicación principal. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, codificar/decodificar emails, filtrar imágenes de programas gráficos...

[18] FTP es un protocolo de la red Internet, significa File Transfer Protocol (Protocolo de Transferencia de Ficheros) y es el ideal para transferir grandes bloques de datos por la red.

SMTP[19], NNTP[20] y HTTP/HTTPS.

Este servicio convierte a un computador en un servidor de Internet o Intranet es decir que en la computadora que tiene este servicio instalado se pueden publicar páginas *Web* tanto local como remotamente (servidor *Web*). El servidor *Web* se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas, por ejemplo Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl.

- **Apache** El servidor HTTP Apache es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor. Era, en inglés, *a patchy server* (un servidor parcheado).

  El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

  Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

  En la actualidad (2005), Apache es el servidor HTTP más usado, siendo el servidor HTTP del 68 % de los sitios *Web* en el mundo y creciendo aún su cuota de mercado. La arquitectura del servidor Apache es muy modular. El servidor consta de un sección "núcleo" y mucha de la funcionalidad que podría considerarse básica para un servidor *Web* es provista por módulos.

- **Apache - Tomcat** Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP)[21] de Sun Microsystems. Se le considera un servidor de aplicaciones.

[19] SMTP, Simple Mail Transfer Protocol, o protocolo simple de transferencia de correo electrónico. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos.

[20] NNTP, Network News Transport Protocol, o protocolo de transferencia de noticias.

[21] JSP, JavaServer Pages), es la tecnología para generar páginas *Web* de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java.

El entorno Tomcat funciona con cualquier servidor *Web* con soporte para servlets y JSPs. Tomcat incluye el compilador Jasper, que compila páginas JSPs convirtiéndolas en servlets.

Las páginas JSP contienen una mezcla de código Java y HTML. Los servlets son programas Java que se ejecutan en el servidor. Cuando una página JSP es referenciada por primera vez, el compilador Jasper la convierte en un servlet, que es ejecutado a continuación. Cada llamada sucesiva a la página provoca la ejecución del servlet directamente sin que sea necesaria una nueva compilación, salvo que la página cambie.

Tomcat puede, asimismo, funcionar como servidor *Web* por sí mismo.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual.

PHP

PHP[22] es un lenguaje usado en el lado del servidor. Es muy eficiente, permitiendo el acceso a bases de datos usando productos como MySQL, y puede ser usado para crear páginas dinámicas complejas.

PHP y HTML

Sólo con HTML no se pueden crear contenidos generados dinámicamente. Para eso, se necesitará alguna forma de lenguaje script. JavaScript es muy práctico para muchos casos pequeños. La ventaja de JavaScript es que se puede integrar sin problemas en los archivos HTML. Con CGI/Perl (ver 6.3.5) hay muchas más posibilidades que con JavaScript - especialmente todo lo que requiere un procesamiento de datos en el servidor. La razón es que hay que guardar datos en un lugar central para que estén al alcance de todos los visitantes de la página. La desventaja de CGI y Perl es que los scripts Perl son archivos separados que por las características de la interfaz CGI normalmente tienen que guardarse en otros archivos distintos a los archivos HTML del proyecto *Web*. Otro problema es que Perl está diseñado como lenguaje universal y no fue desarrollado especialmente para páginas *Web* dinámicas.

PHP, es la abreviatura de Hypertext Preprocessor (preprocesador de hipertexto). El código PHP, similar al de JavaScript, puede ser anotado directamente dentro de archivos HTML en un lugar útil para esto. Una vez que el archivo HTML está guardado en la *Web* y llamado por un navegador *Web*,

[22] PHP es un lenguaje de programación, el nombre es el acrónimo recursivo de "PHP: Hypertext Preprocessor" (inicialmente Personal Home Page Tools).

el servidor reconoce por ciertas convenciones que el archivo que transmite al navegador no se trata de un archivo HTML normal sino de un archivo HTML con código PHP integrado. Por eso, primero se procesa por un intérprete PHP instalado en el lado del servidor. Este lee los pasajes de código PHP dentro de archivo HTML, ejecuta el código y genera el código HTML final que es transmitido al navegador. PHP puede hacer todo lo que se puede hacer con CGI y Perl.

Servlets

Los servlets son objetos que corren dentro del contexto de un servidor de aplicaciones (ej: Tomcat) y extienden su funcionalidad generando contenido dinámico (Mun01).

La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas escritos en Java que se ejecutan en el contexto de un navegador *Web*. Por contraposición, un servlet es un programa que se ejecuta en un servidor *Web*.

El uso más común de los servlets es generar páginas *Web* de forma dinámica a partir de los parámetros de la petición que envíe el navegador *Web*.

Un servlet es un objeto que se ejecuta en un servidor o contenedor J2EE, especialmente diseñado para generar contenido dinámico desde un servidor *Web*, para ser enviado al cliente (navegador) que lo solicitó. Otras opciones que permiten generar contenido dinámico son los lenguajes ASP, PHP y Python.

Los servlets forman parte de Java 2 Enterprise Edition (J2EE), que es una ampliación de Java 2 Standard Edition (J2SE).

Un servlet es un objeto Java que implementa la interfaz javax.servlet.Servlet ó hereda alguna de las clases más convenientes para un protocolo específico. Esta interfaz muestra los servicios que ofrece el servlet y que pueden ser conseguidos por el usuario a través del contenedor.

Entre el servidor de aplicaciones (ó contenedor *Web*) y el servlet existe un contrato que determina cómo han de interactuar.

ASP

ASP (Active Server Pages) - Las "Páginas Activas" se utilizan para ejecutar acciones del lado del servidor. De forma opuesta a JavaScript, que realiza procedimientos en la máquina de cada usuario, ASP genera en el servidor los resultados que luego se mostrarán en las pantallas de cada navegador. Aunque hay versiones de ASP para Unix y Linux, fue desarrollado principalmente para ser usado en servidores *Web* basados en sistemas Microsoft. Las páginas

activas, o dinámicas, son especialmente útiles para mantener bases de datos, crear buscadores dinámicos, hacer carritos de compras, y todo aquello que necesite una interacción del navegador y el servidor para elaborar un resultado.

ASP y HTML

ASP representa en forma similar a PHP una alternativa para CGI/Perl. Con ASP se trata de crear páginas *Web* dinámicas. Similarmente al código PHP, se puede insertar el código directamente en HTML - al contrario que Perl o CGI no son necesarios scripts separados que deben encontrarse en directorios especiales. A diferencia de PHP, ASP se encuentra acoplada sobre todo al mundo de Microsoft y Windows. Aunque también existe ASP para Linux y otros sistemas basados en Unix, y además de los servidores *Web* de Microsoft, también es soportado por el servidor Apache y muchos otros. Pero la integración de ASP esta optimizada para servidores *Web* de Microsoft y por tal razón se utiliza sobre todo bajo Windows. ASP es, a diferencia de PHP, solamente un "entorno", y no un lenguaje script. Como lenguajes scripts se consideran los lenguajes propios de Microsoft como JScript y VBScript. Pero también es posible el uso de Perl para ASP. VBScript es sin embargo el lenguaje estándar para ASP. El principio de ASP es el mismo que el de PHP. Por razón de algunas convenciones, el servidor *Web* reconoce que un archivo HTML es un archivo ASP. El servidor con el entorno ASP integrado ejecuta entonces el código del script contenido en el archivo y envía al navegador el código HTML que ha sido generado.

Common Gateway Interface (CGIs) Perl

CGI (Common Gateway Interface) - La "Interfaz Común de Entrada" es uno de las más antiguos estándares en internet para trasladar la información desde una página *Web* a un servidor *Web*. CGI no es en absoluto un lenguaje de script, de hecho las rutinas de CGI son habitualmente escritas en lenguajes interpretados como Perl o por lenguajes compilados como C.

Perl - Un lenguaje de programación que nos permite que el CGI tradicional se ejecute en el servidor. Perl es muy fácil de aprender y llano en su funcionamiento. Es principalmente usado para libros de visita, formularios de consulta y otras tareas sencillas. Para tareas complejas, un lenguaje del lado del servidor (server-side) como PHP o ASP es mucho más conveniente.

Interfaz CGI e intérprete Perl

La interfaz CGI es una posibilidad de poner en la *Web* programas o scripts que pueden ser llamados desde archivos HTML y que ellos mismos pueden generar código HTML y enviarlo a un navegador *Web*. En contraste con JavaScript, estos scripts o programas no son ejecutados en el navegador del usuario después del envío, sino antes de que el navegador reciba los datos enviados por el servidor. Por eso, los scripts CGI o programas CGI sólo pueden ser ejecutados cuando hay una comunicación basada en HTTP entre el navegador *Web* y el servidor *Web*. En esta comunicación, CGI esta plantado en el lado del servidor. El procesamiento de datos también es ejecutado en el servidor. Los programas CGI pueden guardar datos en el servidor. Un script CGI puede también consultar bases de datos instaladas en el servidor. Con una llamada correspondiente un programa CGI puede leer datos guardados y generar código HTML de ellos. Este código HTML generado "dinámicamente" es transmitido al navegador *Web* de un usuario, y allí se pueden visualizar los datos en forma de HTML. El software del servidor *Web* debe soportar la interfaz CGI.

No hay reglas para el lenguaje de programación en que un programa CGI debe estar escrito. Para que el programa sea ejecutable en el servidor, o debe ser compilado para el entorno del sistema operativo del servidor como programa ejecutable o en el servidor debe haber un intérprete en tiempo de ejecución que ejecute el programa. La mayoría de los programas CGI hoy en día no son programas compilados, sino solamente scripts los cuales son ejecutados por un intérprete en el momento de llamada. El intérprete más conocido y popular en este sentido es el intérprete de Perl. Perl es un lenguaje de programación que puede ser descrito como una mezcla entre los lenguajes de programación clásicos como C y lenguajes de script como el script de shell de Unix. CGI sólo es una norma para una interfaz de programación que debería ser soportada por el software de servidores *Web* y Perl es un lenguaje de script usable de modo universal que, sin duda, es muy apropiado para la programación CGI.

6.3.6. Elección de las tecnologías para el Proyecto

Teniendo en cuenta lo expuesto anteriormente se considera que las tecnologías adecuadas para realizar el Proyecto son las siguientes:

Servidor *Web* Como servidor *Web* se elige Apache HTML que es de libre distribución y es también muy ampliamente utilizado.

Lenguaje de Programación Para programar la aplicación en el servidor se elige PHP, que es un lenguaje sencillo, potente, desarrollado precisamente para páginas *Web* dinámicas, con interfaz sencilla para acceder

a Base de Datos (muy integrado con MySQL) y que además es de libre distribución. También se podía haber elegido Perl, pero el código no se integra en el fichero html, sino que está en ficheros separados.

Base de Datos Como servidor de Bases de Datos se usará MySQL, ya que está ampliamente extendido, es de código abierto y fácil de administrar. Además está muy bien integrado con PHP que es el lenguaje de programación elegido para el lado del servidor.

Procesador de texto Para generar la documentación del Proyecto se usará LaTeX, utilizando los entornos para Windows MiKTeX y WinEdt.

6.4. Cronograma

Un posible cronograma para la realización del Proyecto se muestra en la tabla 6.1.

Noviembre	
4ª semana	Envío del anteproyecto al director y comienzo del análisis de requisitos.
Diciembre	
1ª semana	Análisis de requisitos.
2ª y 3ª semanas	Diseño de la aplicación
4ª semana	Vacaciones.
Enero	
1ª y 2ª semanas	Implementación de la base de datos y de la página del usuario cliente.
3ª y 4ª semanas	Vacaciones
Febrero	
1ª y 2ª semanas	Implementación de la página del usuario vendedor
3ª y 4ª semanas	Implementación de la página del usuario almacenista
Marzo	
1ª y 2ª semanas	Implementación de la página del usuario administrador
3ª y 4ª semanas	Integración de las páginas y ajustes finales
Abril	
1ª y 2ª semanas	Pruebas
3ª y 4ª semanas	Escritura de la memoria
Mayo	
1ª y 2ª semanas	Envío de la memoria al director y correcciones pertinentes
3ª y 4ª semanas	Vacaciones
Junio	
1ª y 2ª semanas	Redacción definitiva de la memoria
3ª y 4ª semanas	Resolución de problemas residuales y entrega de la memoria
Julio	
Todo el mes	Mes de reserva para solucionar cualquier posible contingencia

Cuadro 6.1: Cronograma

Parte III

Proyecto

Capítulo 7

Estado actual de las técnicas y tecnologías

En este capítulo se hace un estudio de los diferentes modelos de desarrollo del software comentando sus ventajas e inconvenientes. Se repasan los fundamentos del análisis de requisitos, comentando los principios, las tareas y las diferentes técnicas de modelado. Se analizan igualmente los fundamentos del diseño del software, estudiando sus conceptos principales y los diferentes métodos. Finalmente se hace una introducción de los diferentes tipos de pruebas que se pueden realizar a un producto software.

7.1. Modelo de desarrollo del software

El proceso de desarrollo de software se puede caracterizar estableciendo un marco común del mismo, definiendo un pequeño número de actividades que son aplicables a todos los proyectos de software, con independencia de su tamaño y complejidad (Pre97).

Para definir el proceso de desarrollo software estas actividades se pueden agrupar en tres fases genéricas:

Fase de definición Se centra en el *qué*. Incluye tareas como la planificación del proyecto y el análisis de requisitos.

Fase de desarrollo Se centra en el *cómo*. Incluye tareas como el diseño, la generación de código y la prueba del software.

Fase de mantenimiento Se centra en el *cambio*. Incluye tareas como corrección, mejora, adaptación.

Para un caso real se debe incorporar una estrategia que acompañe al proceso, seleccionando las tareas adecuadas al problema en cuestión y el orden en que se realizarán. Esta estrategia se denomina ***modelo de desarrollo del proceso***. Se selecciona un modelo de proceso según la naturaleza del proyecto, de los métodos y herramientas a utilizar y de los controles y entregas que se requieren.

El ciclo de vida software determina el orden en que se realizan las actividades. Es el periodo que comienza cuando un producto software es concebido y termina cuando deja de estar disponible.

Los principales modelos de ciclo de vida software son: los secuenciales, incrementales, evolutivos y los de desarrollo de prototipos.

7.1.1. Modelos secuenciales

Los modelos secuenciales de ciclo de vida plantean el desarrollo y la explotación de una aplicación software como una secuencia de actividades sucesivas y diferentes que se van realizando una tras otra.

Las actividades típicas o fases que se realizan, precisamente en el orden indicado, son: análisis de viabilidad, análisis de requisitos, diseño, implementación, pruebas, integración y mantenimiento.

Estos modelos tratan de aislar cada fase de la siguiente, de manera que cada fase pueda ser realizada por grupos de personas diferentes. Para conseguir esta independencia es fundamental que en cada fase se genere una información de salida precisa y suficiente para acometer la siguiente fase (Pre97).

Estos modelos presentan las siguientes ventajas:

- Son fáciles de comprender.
- Son fáciles de planificar y seguir.
- Son los más antiguos, están suficientemente probados y existen múltiples herramientas que los soportan.

Por el contrario tienen como desventajas:

- Exigen la definición completa de los requisitos al principio.
- No se ajustan bien a los cambios de requisitos.
- El producto no está disponible hasta que está terminado, el cliente debe tener paciencia.

- Los proyectos reales pocas veces son secuenciales.

Se recomienda el uso de estos modelos cuando:

- El proyecto es similar a alguno ya realizado con éxito.
- Los requisitos son estables y están bien comprendidos.
- El diseño y la tecnología esta probada y madura.
- La duración del proyecto es relativamente corta.
- El cliente no necesita versiones intermedias, se entrega todo el producto al final.

Entre los principales modelos secuenciales se encuentran el modelo en cascada y el modelo en V. El modelo en cascada, (ver figura 7.1) es el paradigma más antiguo y más extensamente utilizado (CC95). El esquema del modelo en V puede verse en la figura 7.2.

7.1.2. Modelos incrementales

Los modelos incrementales suponen, al igual que los secuenciales, que los requisitos están bien comprendidos desde el principio, se realiza el diseño y a partir de este momento se producen una serie de entregables en la que cada uno de ellos incorpora una parte de las capacidades planificadas del sistema. El sistema funciona desde el primer entregable aunque con funcionalidad limitada, con el último entregable el sistema se completa.

Estos modelos presentan las siguientes ventajas:

- Reducen los riesgos de retrasos en la entrega.
- Los entregables intermedios facilitan la realimentación con los subsiguientes entregables.
- El usuario puede validar el sistema a medida que se construye.

Por el contrario tienen como desventajas:

- Es sensible a la planificación de entregables.
- Requiere una gestión del proyecto muy estricta.

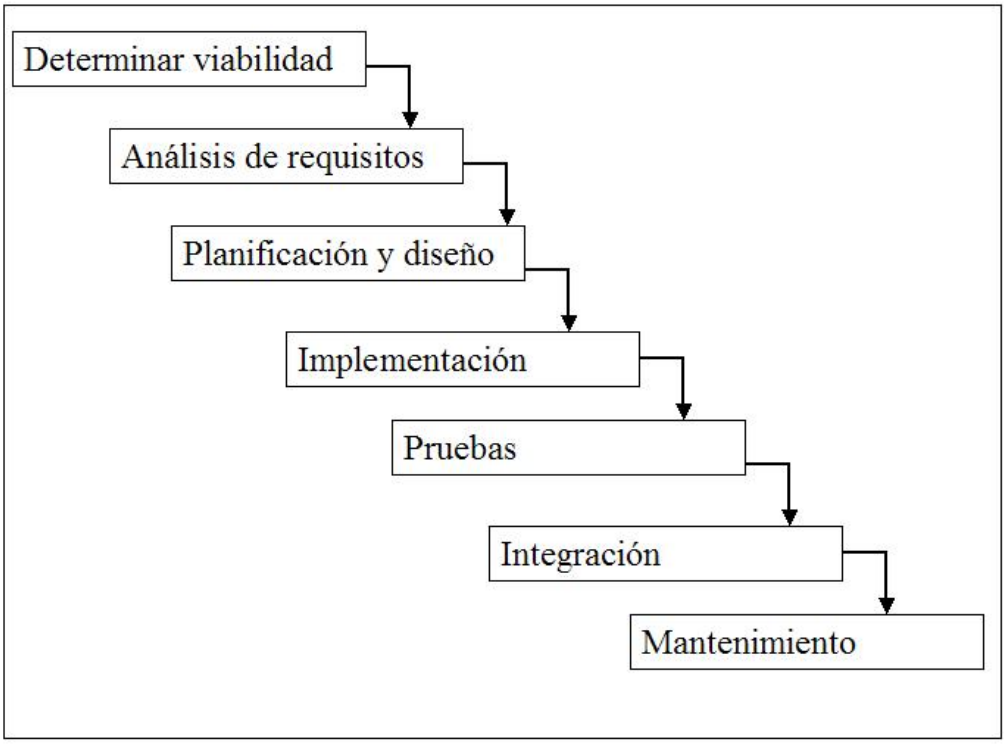

Figura 7.1: Ciclo de vida en cascada (CC95)

Se recomienda el uso de estos modelos cuando:

- El proyecto es similar a alguno ya realizado anteriormente.
- Los requisitos son estables y están bien comprendidos.
- El diseño y la tecnología están probados y maduros.
- La duración del proyecto es relativamente larga.
- El cliente necesita versiones intermedias rápidamente.

El modelo incremental se puede ver en la figura 7.3

7.1.3. Modelos evolutivos

El ciclo de vida evolutivo desarrolla también el sistema en fases, generando un entregable al final de cada iteración. El modelo evolutivo se diferencia

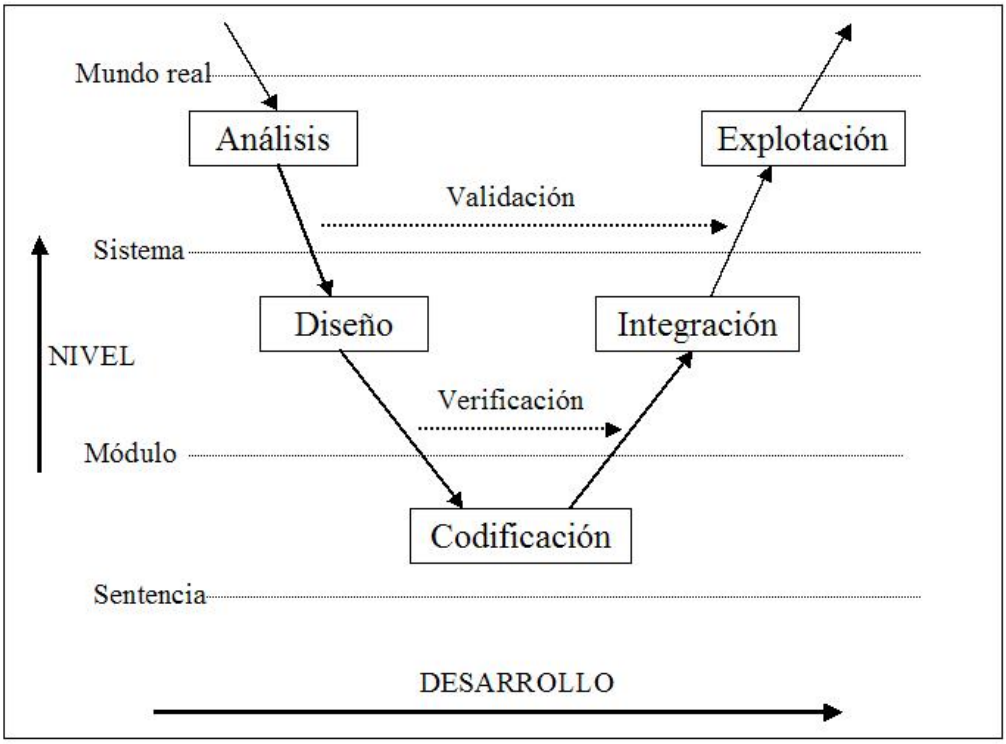

Figura 7.2: Ciclo de vida en V (CC95)

del incremental en que se reconoce que las necesidades del usuario no son completamente comprendidas ni están completas. Ello requiere una vuelta a analizar, planificar y diseñar después de la implementación de cada entregable (Pre97).

Estos modelos presentan las siguientes ventajas:

- No se necesita conocer los requisitos al principio.
- Los entregables intermedios facilitan la realimentación con los subsiguientes entregables.
- Las técnicas de prototipado permiten demostrar la funcionalidad del sistema con poco esfuerzo desde el principio.

Por el contrario tienen como desventajas:

- Es difícil estimar el esfuerzo necesario.

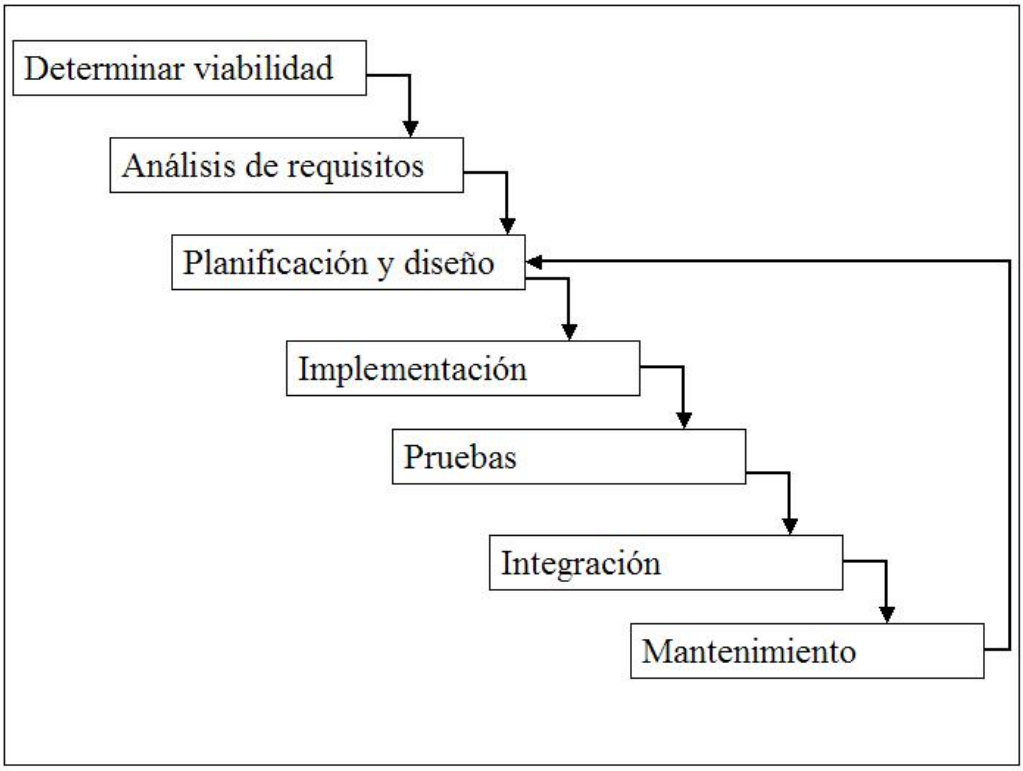

Figura 7.3: Ciclo de vida incremental (CC95)

- Se tiene el riesgo de no terminar nunca, se entra en una espiral de entrega de versiones.

Se recomienda el uso de estos modelos cuando:

- Los requisitos no están completamente definidos y son posibles los cambios.
- Se están probando nuevas tecnologías.
- La capacidad del sistema necesita la evaluación de los usuarios.
- Existen grupos diversos de usuarios con intereses diferentes.

El modelo evolutivo se puede ver en la figura 7.4

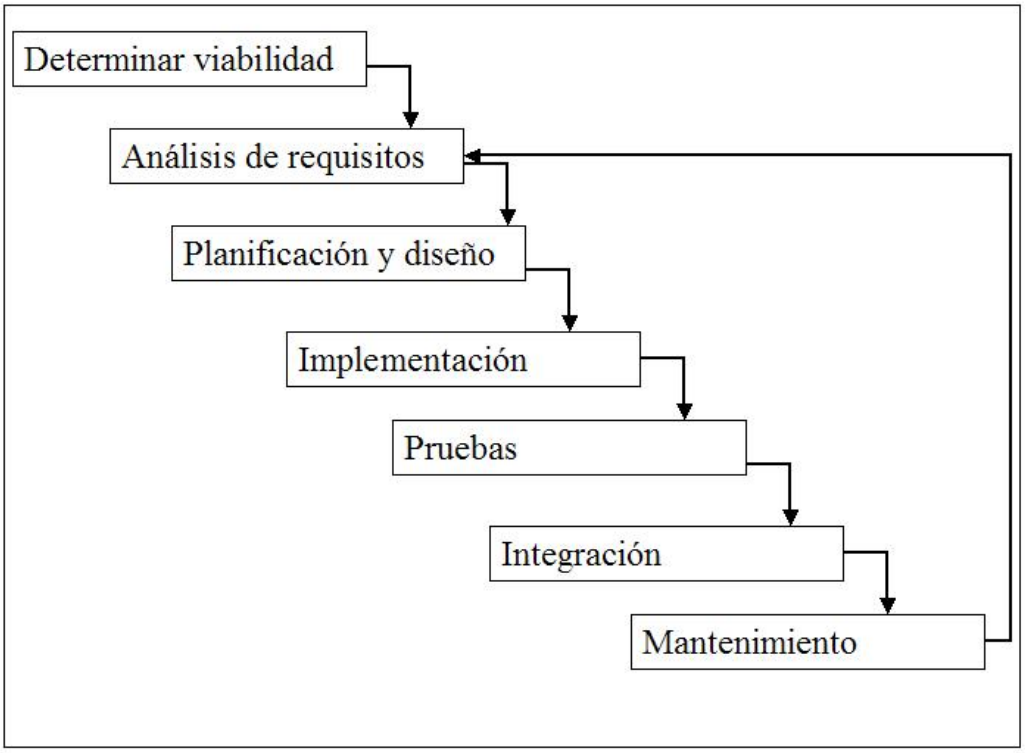

Figura 7.4: Ciclo de vida evolutivo (CC95)

Uno de los principales modelos evolutivos es el modelo en espiral. Este modelo tiene como elemento distintivo que introduce la actividad de análisis de riesgo como elemento fundamental para guiar la evolución del proceso de desarrollo. El ciclo de iteración del modelo evolutivo se convierte en una espiral al añadir como dimensión radial una indicación del esfuerzo total realizado hasta ese momento, que será un valor siempre creciente.

El modelo en espiral se puede ver en la figura 7.5

7.1.4. Modelos de construcción de prototipos

Se dan muchos casos en que un cliente define un conjunto de objetivos generales, pero no identifica los requisitos detallados de entrada y salida. En otros casos el responsable del desarrollo puede no estar seguro de la eficiencia de un algoritmo, o de cómo debe ser la interacción hombre-máquina. En estos y en muchos otros casos, un paradigma de construcción de prototipos puede ofrecer el mejor enfoque.

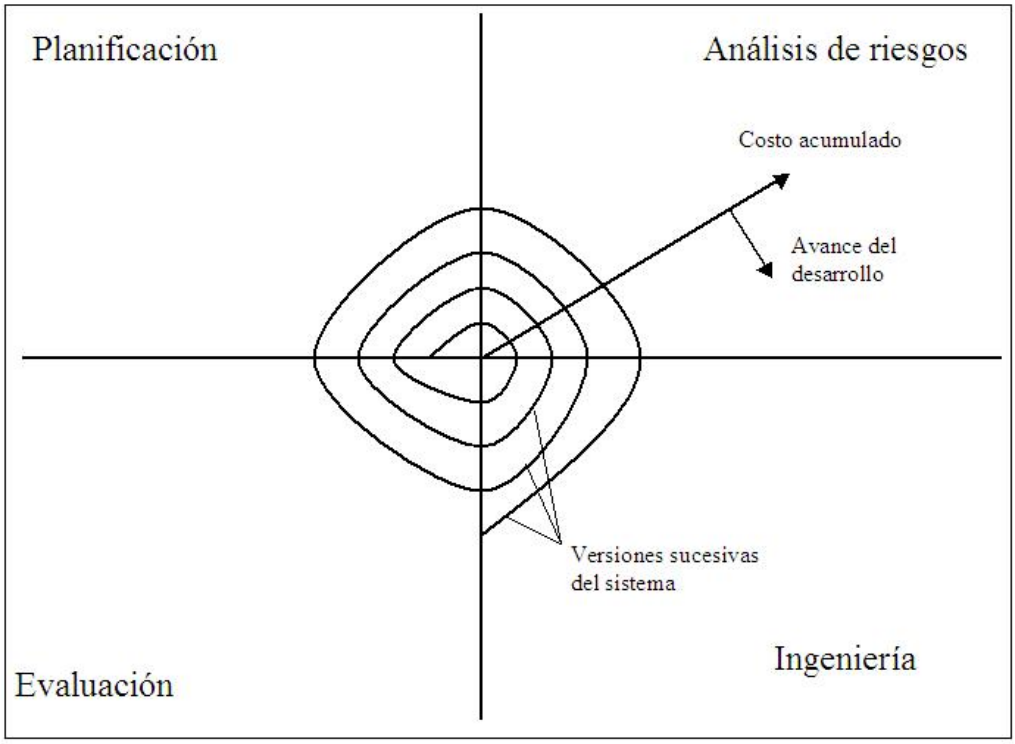

Figura 7.5: Ciclo de vida en espiral (CC95)

El paradigma de construcción de prototipos (ver figura 7.6) comienza con la recolección de requisitos. Se encuentran y definen los objetivos globales, se identifican los requisitos conocidos. Entonces se hace un diseño rápido centrado en los aspectos del software que serán visibles por el usuario. El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente y se utiliza para refinar los requisitos a desarrollar.

Se pueden distinguir dos clases de prototipos, según se pretenda aprovechar el código del mismo, o sólo la experiencia obtenida del mismo.

Prototipos rápidos Su finalidad es sólo adquirir experiencia. Se suelen aprovechar en las fases iniciales de análisis y diseño. Una vez completadas estas fases son desechados y el sistema se codifica partiendo de cero.

Prototipos evolutivos Se trata de aprovechar el código al máximo. El pro-

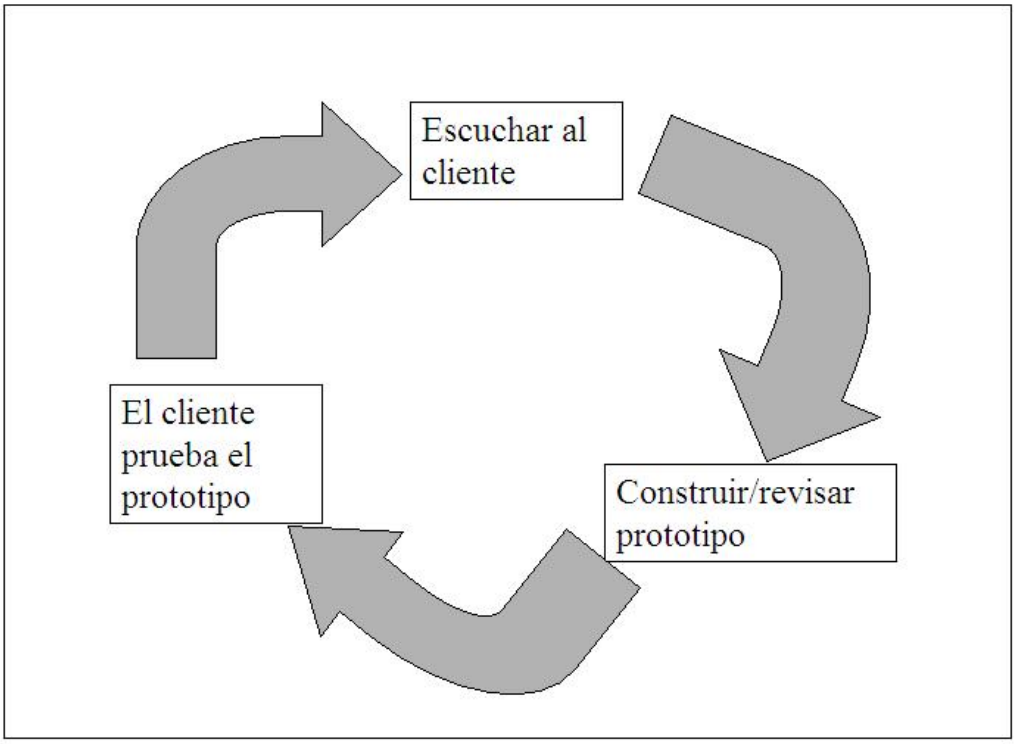

Figura 7.6: Paradigma de la construcción de prototipos (Pre97)

totipo inicial sólo tendrá algunas funcionalidades del sistema, y se le van añadiendo otras en sucesivas iteraciones.

7.2. Fundamentos del análisis de requisitos

El análisis de requisitos trata de caracterizar el problema a resolver. Permite especificar la función y el rendimiento del software, indica la interfaz con otros elementos del sistema y establece las restricciones que deben cumplirse.

El análisis de requisitos permite construir modelos de los dominios de datos, funcional y de comportamiento del sistema, modelos que en la fase siguiente se plasmarán en el diseño del sistema.

Finalmente la especificación de requisitos proporciona los medios para valorar la calidad una vez construido el software.

7.2.1. Principios del análisis

Hay diferentes métodos de análisis, cada uno con diferentes puntos de vista. Sin embargo todos los métodos siguen un conjunto de principios operativos:

1. Debe representarse y entenderse el dominio de la información del problema.

2. Deben definirse las funciones que ha de realizar el software (modelo funcional).

3. Debe representarse el comportamiento del software ante acontecimientos externos (modelo de comportamiento).

4. Deben dividirse jerárquicamente los modelos (partición del sistema).

5. El proceso de análisis debe ir desde la información esencial hasta el detalle de la implementación.

El dominio de la información

El software se construye para procesar datos, es decir, para aceptar la entrada de datos, procesarlos y producir una salida de información.

En este sentido será necesario analizar el contenido de la información, el flujo de la información y la estructura de dicha información.

Modelado

Los modelos se crean para entender mejor la entidad que se va a construir. Los modelos se centran en el qué debe hacer el sistema, no en cómo lo hace.

En los modelos funcionales se parte de un modelo a nivel contextual y tras sucesivas iteraciones se obtiene una minuciosa definición de la funcionalidad del sistema.

Los modelos de comportamiento crean una representación de los estados del sistema y de los acontecimientos o eventos que causan que cambie de estado.

Partición

Cuando un sistema es demasiado grande es necesario dividirlo en partes fácilmente entendibles y establecer la interacción entre las partes.

En esencia la partición descompone un sistema en sus partes constituyentes, estableciendo una relación jerárquica entre ellas.

7.2.2. Tareas del análisis

Las tareas principales del análisis serán:

- Desarrollar un modelo del sistema.
- Elaborar un documento de especificación de requisitos.

Para lograr una especificación correcta, el modelo deberá tener las siguientes propiedades:

- Completo conciso y sin ambigüedades.
- Sin detalles de diseño o implementación.
- Entendible por el cliente.
- Separación de los requisitos funcionales y no funcionales.
- División y jerarquización el sistema.
- Establecimiento de los criterios de validación.

7.2.3. Modelado del análisis

Aunque hay diferentes métodos para el modelado del análisis de requisitos, dos tendencias son las dominantes (Pre97): el *análisis estructurado* y el *análisis orientado a objetos*. En este proyecto se usará básicamente el análisis estructurado complementado con algún elemento tomado del análisis orientado a objetos, como son los casos de uso.

El análisis estructurado es una actividad de construcción de modelos. Se crean modelos que representan el contenido y el flujo de la información, de la función y del comportamiento. Los elementos de este modelo de análisis son:

Diccionario de datos Que es un depósito con las definiciones de los datos utilizados por el software.

DER El Diagrama Entidad-Relación representa la relaciones entre los objetos de datos, el DER es la notación utilizada para modelar los datos.

Descripción de objetos de datos Describe los atributos de los objetos señalados en el DER.

DFD El Diagrama de Flujo de Datos indica cómo se transforma la información a medida que avanza por el sistema y representa las funciones que transforman el flujo de datos.

EP La Especificación de Proceso describe cada función representada en el DFD.

DTE El Diagrama de Transición de Estados indica cómo se comporta el sistema ante sucesos externos. Para lograr esto se representan los diferentes modos de comportamiento (estados) y la manera en que se hacen las transiciones entre estados.

EC La Especificación de Control describe aspectos sobre el control del software.

Modelado de datos

El modelado de datos identifica los objetos de datos que procesa el sistema, cuál es su composición y atributos, dónde residen y cuál es la relación entre los objetos y los procesos que los transforman.

Para modelar los datos se utiliza el diagrama entidad-relación (DER). Los datos se estudian independientemente del procesamiento que los transforma.

El modelo se compone de tres elementos interrelacionados:

Objetos de datos Son representaciones de cualquier información que deba utilizar el software.

Atributos Describen las propiedades de los objetos de datos.

Relaciones Representan la forma en que se conectan los objetos de datos, definiendo un conjunto de parejas objeto-relación.

Otros elementos que dan información adicional a los elementos básicos son:

Cardinalidad Representa el número de ocurrencias en una relación.

Modalidad Indica si una relación ha de existir necesariamente o es opcional.

El diagrama entidad-relación, DER, representa gráficamente las entidades de datos del sistema y sus relaciones, así como su cardinalidad y modalidad. Los atributos suelen representarse en tablas anejas. La notación de un DER se puede ver en la figura 7.7.

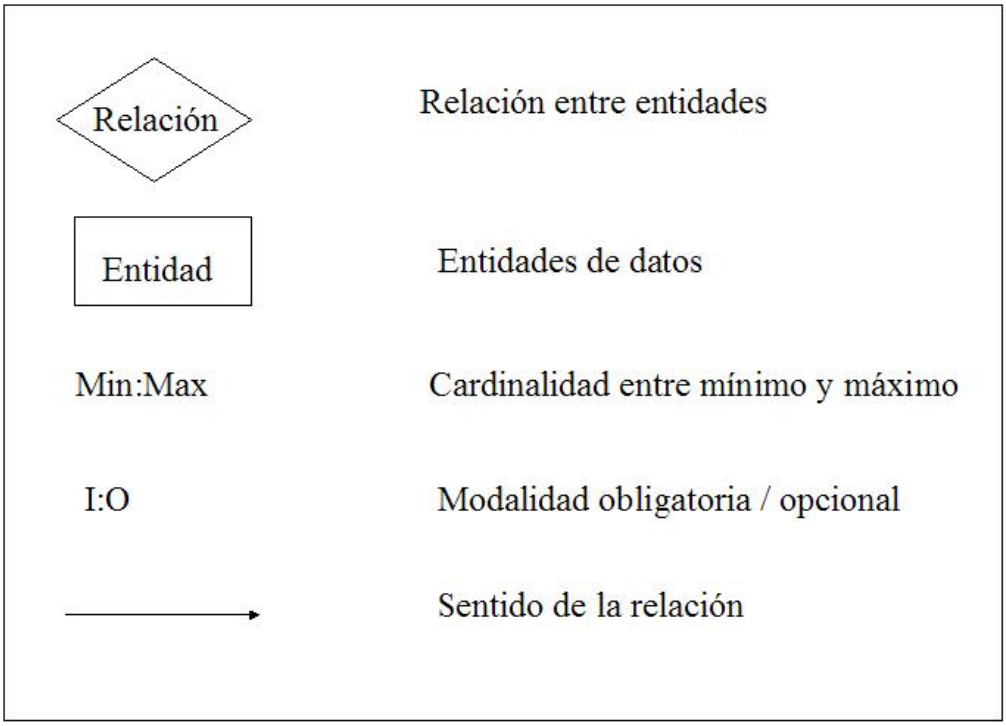

Figura 7.7: Notación de Diagramas Entidad-Relación (CC95)

Modelado funcional y flujo de información

La información se transforma a medida que fluye por el sistema. El diagrama de flujo de datos (DFD) es una técnica que representa el flujo y la transformación de la información desde la entrada a la salida.

Se pueden representar los niveles jerárquicos del sistema con DFD´s que muestran más detalles. Así un DFD de nivel 0 representa al sistema completo como una única burbuja con datos de entrada y salida. Al explotar (particionar) una burbuja aparecen otros procesos (burbujas) y caminos de datos adicionales. Las entradas y salidas de datos de una burbuja que se explota deberán estar representadas en el DFD de nivel inferior obtenido. Este proceso de refinamiento continuará hasta que las funciones representadas en DFD sean suficientemente simples para poder ser descritas en lenguaje natural o en seudocódigo. La notación básica de un DFD se puede ver en la figura 7.8

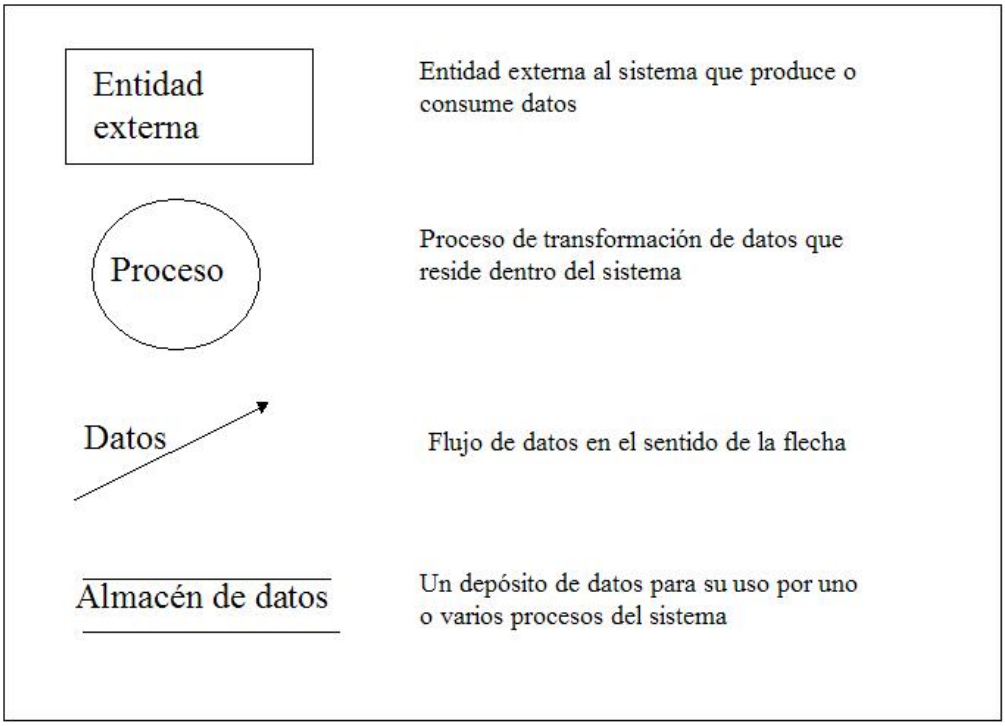

Figura 7.8: Notación de Diagramas de Flujo de Datos (CC95)

Un diagrama de flujo de datos representa el flujo de la información pero no la lógica del procesamiento, esto se representará en los modelos de comportamiento.

Los datos que etiquetan las flechas del diagrama estarán definidos en el diccionario de datos, otro componente del análisis estructurado que se explicará posteriormente.

Modelado del comportamiento

Para representar el comportamiento del sistema se utilizan los diagramas de transición de estados (DTE), que muestran los estados del sistema y los sucesos que hacen que cambie de estado. Un estado es un modo observable del sistema. Un DTE muestra cómo se mueve el sistema de us estado a otro. La notación básica de un DTE se puede ver en la figura 7.9

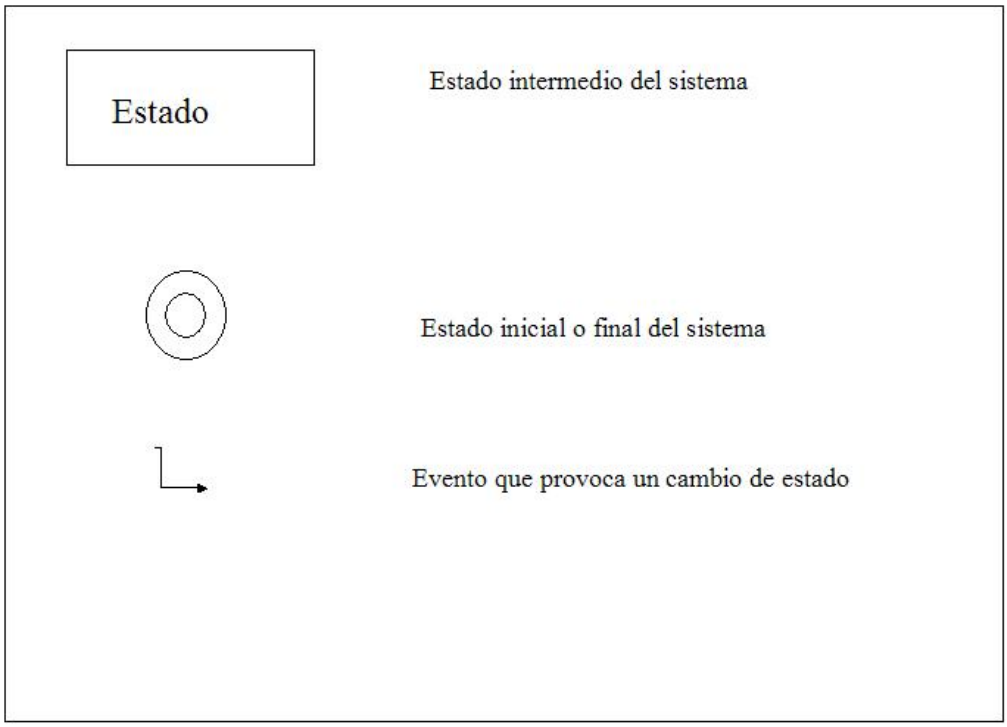

Figura 7.9: Notación de Diagramas de Transición de Estados (CC95)

Diccionario de datos

Para que los modelos de análisis sean consistentes es necesario representar de forma organizada las características de los objetos de datos. Esto se realiza en el diccionario de datos.

Un diccionario de datos debería tener una gramática casi formal para describir el contenido de los objetos definidos durante el análisis estructurado. La información mínima que debería contener sería:

Nombre El nombre del dato, del almacén de datos o de una entidad externa.

Dónde/cómo se usa Un listado de los procesos que los utilizan.

Descripción del contenido Contenido del dato representado en alguna notación.

La notación utilizada para describir el contenido de un diccionario de datos puede verse en la figura 7.10

A = B	A está compuesto de B
A + B	Concatenación de A y B
[A|B]	Selección entre A o B
$\{A\}^n$	Repetición n veces de A
(A)	Inclusión opcional de A
/ comentario/	Separador de comentarios

Figura 7.10: Notación para Diccionario de Datos (CC95)

Casos de uso

El análisis orientado a objetos (AOO) se basa, al igual que el análisis estructurado, en cinco principios básicos: modelado de datos, modelado funcional, modelado del comportamiento, partición del sistema e ir de lo esencial a los detalles durante el proceso de análisis.

Sin embargo, ambos paradigmas tienen una visión diferente del sistema. El enfoque estructurado considera a los datos separadamente de los procesos que los transforman, el comportamiento tiende a jugar un papel secundario y hace un fuerte uso de la descomposición funcional. El enfoque orientado a objetos centra su esfuerzo en definir las clases que son relevantes para el problema. Una clase es un concepto orientado a objetos que encapsula los datos y los procedimientos necesarios para describir su contenido y su comportamiento, ahora los datos están muy relacionados con los procesos que los transforman.

Hay múltiples métodos de análisis orientado a objetos. En uno de ellos, el método de Jacobson, se da mucha importancia a los casos de uso.

Un caso de uso es una descripción o escenario que especifica cómo el usuario interactúa con el sistema; aporta una descripción de cómo debe ser usado éste.

Para crear un caso de uso se deben identificar los actores que usan el sistema. Un actor es cualquier cosa externa al sistema que se comunique con él. Un actor y un usuario no son la misma cosa. Un usuario puede desempeñar un cierto número de roles cuando usa el sistema, mientras que un actor representa una entidad externa que representa un único rol.

Un caso de uso es una narración escrita que describe, de forma no ambigua, el papel de un actor al interactuar con el sistema. Los casos de uso también pueden utilizarse para especificar requisitos y restricciones del sistema.

En resumen, los casos de uso tienen las siguientes características:

- Están expresados desde el punto de vista del actor.
- Se documentan con texto informal.
- Describen tanto lo que hace el actor como lo que hace el sistema, aunque el énfasis está puesto en la interacción.
- Son iniciados por un único actor.
- Están acotados al uso de una determinada funcionalidad, claramente diferenciada, del sistema.

Los casos de uso son muy útiles para modelar los requisitos funcionales del sistema, haciendo énfasis en el comportamiento. En casos donde puede haber diferentes usuarios del sistema con diferentes roles, como es el caso de este proyecto, se pueden utilizar los casos de uso para completar el modelo de comportamiento del sistema, puesto que en el enfoque estructurado el comportamiento juega un papel secundario, pues está mas enfocado a procesos secuenciales.

7.3. Fundamentos del diseño del software

Con el diseño se trata de definir el sistema con el suficiente detalle para permitir su realización física. Para ello se aplicarán diferentes técnicas y principios.

La fase de diseño toma como punto de partida los requisitos que se han obtenido y especificado en la fase previa del ciclo de vida. El análisis de requisitos indica el "qué" hace el sistema mientras que el diseño nos dice el "cómo" lo hace.

7.3.1. Conceptos de diseño

Existen una serie de conceptos fundamentales de diseño del software, que son la base para obtener un software de calidad y ayudan a seleccionar un conjunto de criterios para particionar el software y a extraer la función y la estructura de datos de los modelos conceptuales. Los principales conceptos de diseño son (CC95):

Abstracción Se parte de un nivel alto de abstracción, con una solución en términos amplios, bajando hasta los niveles más bajos de la jerarquía donde la solución tiene un nivel de detalle mucho mayor. Este concepto se aplica tanto a los datos como a las funciones del sistema.

Refinamiento Es una estrategia de diseño descendente. En cada paso se descomponen elementos en otros más detallados, terminando el proceso cuando el elemento se puede expresar en un lenguaje de programación. La abstracción y el refinamiento son conceptos complementarios. La abstracción permite suprimir detalles de bajo nivel, mientras que el refinamiento muestra esos detalles a medida que se progresa en el diseño.

Modularidad Se trata de dividir el software en componentes identificables y tratables por separado, llamados módulos. Estos módulos integrados constituyen el sistema. La modularidad permite que un programa sea conceptualmente manejable.

Arquitectura del software Alude a la estructura jerárquica de los módulos, a la manera de interactuar de éstos y a la estructura de los datos utilizados.

Partición Alude a cómo se divide la estructura del programa. La partición horizontal define ramas separadas en la jerarquía para cada función. La

partición vertical sugiere que los módulos superiores de la jerarquía se dedican a la toma de decisiones y funciones de control, mientras que los módulos inferiores son los que realizan el trabajo de procesamiento.

Estructura de datos Representa la relación lógica entre los elementos de datos. Se parte de unas pocas estructuras básicas para construir otras mas sofisticadas.

Ocultación de información Se trata de diseñar módulos cuyos procedimientos y datos sean inaccesibles a otros módulos que no necesitan esa información. Se trata de crear interfaces que digan lo que hace el módulo ocultando a los demás cómo lo hace.

7.3.2. Descomposición modular

La modularidad se ha convertido en un elemento básico del desarrollo del software. Un diseño modular reduce la complejidad, facilita los cambios y favorece una implementación paralela de diferentes partes del sistema (CC95).

La descomposición modular para ser efectiva debe tener una serie de cualidades mínimas. Estas cualidades son:

Independencia funcional La independencia funcional se consigue desarrollando módulos que realizan una función única y que tienen una interacción mínima con otros módulos, con ella el software es más fácil de desarrollar y mantener y las interfaces se simplifican. La independencia funcional se mide usando dos criterios cualitativos: cohesión y acoplamiento.

Cohesión La cohesión es una medida de la fuerza interna funcional de un módulo. Un módulo con cohesión alta realiza una única tarea, requiriendo poca interacción con otras partes del programa. Hay diferentes tipos de cohesión que se pueden agrupar en una escala. Así, en la parte baja estaría la cohesión casual donde la relación entre las partes del módulo es pura coincidencia, en la parte alta estaría la cohesión funcional en la que todos los elementos del módulo cooperan para la realización de un único objetivo común. Entre estos dos extremos de la escala hay un amplio espectro de tipos intermedios. El objetivo del diseño debe ser el de conseguir una cohesión alta o media.

Acoplamiento El acoplamiento es una medida de la interdependencia relativa entre los módulos. Depende de la complejidad de la interfaz de conexión entre módulos y de los datos que se pasan a través de ella.

En un acoplamiento débil solo se pasaran datos a través de la interfaz, en un acoplamiento moderado se pasará información de control entre módulos y finalmente un acoplamiento fuerte ocurre cuando un módulo usa los datos o la información de control mantenidos en otro módulo. El objetivo del diseño será obtener un acoplamiento entre módulos lo más bajo posible.

En resumen, la descomposición modular con una mayor independencia funcional se consigue con un acoplamiento débil entre sus módulos y una cohesión alta dentro de cada uno de ellos.

7.3.3. Métodos de diseño

Aunque existen diferentes métodos de diseño del software, los mas importantes, desarrollados y utilizados son el diseño orientado a flujo de datos y el diseño orientado a objetos.

Diseño orientado a flujo de datos

La metodología de diseño orientada a flujo de datos se centra en el movimiento y transformación de los datos a medida que se mueven por el sistema. Considera los datos de entrada al sistema, estudia todos los caminos que recorren y su transformación hasta que alcanzan la salida. Estas metodologías son consideradas clásicas o convencionales frente a las más novedosas orientadas a objeto.

Durante la fase de diseño se realizan una serie de actividades cuyo objetivo es pasar de las ideas más o menos informales recogidas en la fase de análisis a definiciones detalladas y precisas. Estas actividades son:

Diseño de datos Transforma el modelo de datos obtenido en el análisis en las estructuras de datos necesarias para el sistema. La base para esta actividad son el Diagrama Entidad-Relación (DER) y el diccionario de datos.

Diseño arquitectónico Define la estructura modular del sistema mostrando la relación entre los principales elementos estructurales del sistema. Se parte de los diagramas de flujo de datos obtenidos en el análisis.

Diseño de la interfaz Describe cómo se comunica el software con otros sistemas y con los usuarios del mismo.

Diseño procedimental En esta actividad se transforman los elementos estructurales en una descripción procedimental de los componentes. La base para esta actividad son los requisitos funcionales obtenidos en la fase de análisis.

Todas estas actividades se desarrollan en un proceso iterativo mediante el cual se traducen los requisitos en una representación del software. Se parte de un alto nivel de abstracción refinándolo en cada iteración hasta obtener un nivel lo suficientemente bajo para que se pueda pasar a la fase de implementación.

El modelo de diseño se puede representar como una pirámide, que es un objeto muy estable. En la base se sitúa una amplia capa muy estable con el diseño de los datos. En las capas medias se sitúan el diseño arquitectónico y de interfaz, dejando el pico de la pirámide para el diseño procedimental. La estabilidad de este modelo se basa en situar en las capas bajas los elementos que menos cambian durante el desarrollo del software como es la estructura de los datos, mientras que en la punta de la pirámide se sitúan los elementos que más cambiarán como son las funciones y procedimientos del sistema. El modelo de diseño se muestra en la figura 7.11.

Diseño Orientado a Objetos

El diseño orientado a objetos transforma el modelo de análisis obtenido mediante el análisis orientado a objetos en un modelo de diseño para la construcción del software. Las componentes principales del sistema se organizan en módulos denominados subsistemas. Los datos y las operaciones que los manipulan están encapsulados en objetos, una forma modular que es el bloque de construcción básico de un sistema orientado a objetos (OO).

El diseño OO descansa en cuatro importantes conceptos del diseño: abstracción, ocultación de la información, independencia funcional y modularidad.

En el diseño OO se pueden definir cuatro capas básicas necesarias para obtener un modelo completo del sistema, estas capas son:

La capa de subsistemas Contiene una representación de los subsistemas que permiten al software cumplir con los requisitos definidos en la etapa de análisis.

La capa de clases y objetos Contiene la jerarquía de clases y objetos del sistema.

La capa de mensajes Contiene los detalles que permiten que los objetos se comuniquen entre si. También se establecen las interfaces del sistema.

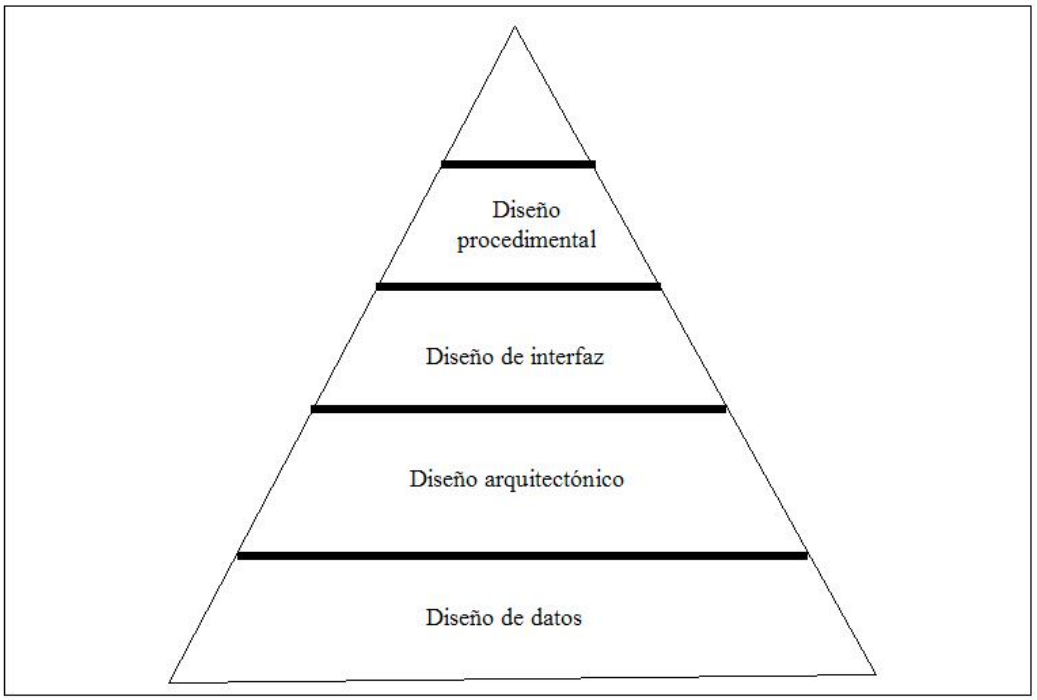

Figura 7.11: Modelo de diseño orientado a flujo de datos (Pre97)

La capa de responsabilidades Contiene las estructuras de datos y el diseño algorítmico para los atributos y operaciones de los objetos.

Al igual que en la metodología orientada a flujo de datos, también se pueden representar estas capas en una pirámide cuya base sería la capa de diseño de subsistemas, situándose sobre ella la de clases y objetos, la de mensajes y la de responsabilidades. El modelo de diseño orientado a objetos puede verse en la figura 7.12.

Hay que resaltar que aunque los métodos convencionales y orientados a objetos utilizan notaciones y heurísticas diferentes para derivar el modelo de diseño del modelo de análisis, se pueden establecer correspondencias entre cada elemento del análisis convencional y una o mas capas del análisis OO.

Así el diseño OO aplica diseño de datos cuando se representan los atributos, diseño de interfaz cuando se desarrolla el modelo de paso de mensajes y diseño procedimental al diseñar las operaciones. Sin embargo el diseño arquitectónico es diferente ya que el diseño OO no exhibe una estructura jerárquica, de hecho la estructura del diseño OO tiene más que ver con las colaboraciones entre objetos que con el flujo de datos y control.

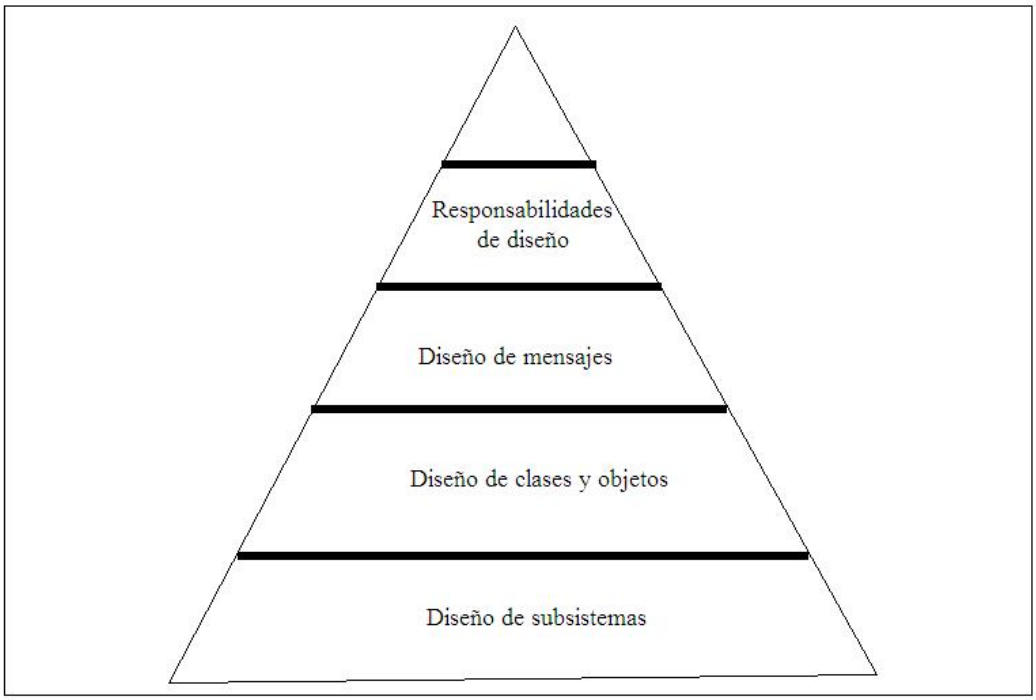

Figura 7.12: Modelo de diseño OO (Pre97)

Elección de la metodología de diseño

La metodología basada en el flujo de datos es la apropiada cuando en la fase de análisis se elige como técnica de desarrollo el análisis estructurado, que también es orientado a flujo de datos y existen técnicas desarrolladas para pasar fácilmente de los modelos de análisis a los modelos de diseño.

Otro factor que puede influir en la elección de la metodología de diseño es la tecnología de implementación que se utilizará, ya que, por ejemplo, resulta más directo pasar de un diseño orientado a objetos a una implementación en un lenguaje OO como Java e, igualmente, es mas sencillo pasar de un diseño orientado a flujo de datos a una implementación en lenguajes estructurados como C.

En las siguientes secciones se analizan las actividades de la metodología orientada a flujo de datos: diseño de datos, diseño arquitectónico, diseño de interfaz y diseño procedimental.

7.3.4. Diseño de datos

La actividad principal del diseño de datos es seleccionar representaciones lógicas de los objetos y estructuras de datos identificados durante la fase de análisis de requisitos.

Una actividad relacionada es identificar los módulos del programa que deben operar sobre las estructuras de datos lógicos.

La mayoría de las aplicaciones requieren almacenar información de forma permanente, lo que se hace generalmente usando una base de datos subyacente.

El proceso de diseño consistirá en convertir el modelo de datos del análisis, típicamente un diagrama entidad-relación, en una base de datos relacional. Se trata de convertir las entidades y las relaciones en tablas de la base de datos del sistema.

La eficiencia de la BD en el modelo relacional se contempla desde dos puntos de vista. Por una parte se establecen las llamadas *formas normales*, que tienden a evitar redundancias en los datos almacenados. Por otra parte se estudia el empleo de *índices* para mejorar la velocidad de acceso a los datos.

Las formas normales llegan hasta la quinta, pero para una aplicación normal basta con que la BD esté en tercera forma normal. Una BD está en tercera forma normal si:

- La información asociada a cada columna de cada tabla es un valor único.
- Para cada tabla hay una clave primaria que distingue a cada fila y cada casilla que no sea clave primaria depende de toda la clave primaria.
- En todas las tablas el valor de cada columna que no es clave primaria depende directamente de la clave primaria.

Los índices permiten acceder rápidamente a un dato concreto, reduciendo el tiempo de acceso. Pero esto se hace a costa de aumentar el espacio de almacenamiento, y de aumentar el tiempo necesario para almacenar un nuevo dato, y más aún para modificar un atributo indexado. En general, si hay que acceder a datos a través de sus relaciones con otros, será conveniente mantener índices sobre las claves primarias y columnas de referencia de las entidades relacionadas.

El proceso para convertir entidades o relaciones en tablas de la BD será el siguiente:

Entidades En el modelo relacional cada entidad del modelo entidad-relación se convierte en una tabla, con una fila por cada elemento de esa clase y una columna por cada atributo de la entidad. Un número o código de referencia sirve perfectamente como clave primaria de la tabla.

Relaciones La manera de almacenar la información de las relaciones en tablas depende de la cardinalidad de la relación. La técnica general es traducir la relación a una tabla conteniendo referencias a las entidades relacionadas, así como los atributos propios de la relación. Esto es válido para relaciones con cualquier cardinalidad, incluyendo N-N. La referencia a las entidades relacionadas se hará mediante la clave primaria de cada una de ellas.

Si la cardinalidad de la relación es 1-N, es decir la cardinalidad de un lado de la relación está limitada a uno, se pueden incluir los datos de la relación en la entidad de cardinalidad uno.

Si la cardinalidad es 1-1 se pueden fundir las tablas de las dos entidades en una sola.

7.3.5. Diseño arquitectónico

El objetivo principal del diseño arquitectónico es desarrollar una estructura modular del programa y representar las relaciones de control entre los módulos. Además en el diseño arquitectónico se combina la estructura del programa y las estructuras de datos, definiendo las interfaces que permiten el flujo de los datos a través del programa.

Las principales técnicas de diseño arquitectónico siguen un modelo de diseño descendente. La descomposición del sistema se hace desde el punto de vista funcional, se atiende a la función o funciones que tiene que realizar el sistema y se van refinando en otras cada vez mas sencillas. Se obtiene así una descomposición modular con el detalle suficiente para ser traducido fácilmente a un lenguaje de programación.

Como técnicas de diseño funcional descendente pueden citarse: el desarrollo por refinamiento progresivo, la programación estructurada de Jackson (Jackson Structured Programming, JSP) y el diseño estructurado.

Desarrollo por refinamiento progresivo

Esta técnica se basa en la metodología de programación estructurada que conduce a la construcción de programas mediante refinamientos sucesivos.

Se emplean estructuras de control claras y sencillas como la secuencia, la selección entre alternativas y la iteración. Cada paso de descomposición

consistirá en refinar la operación considerada en ese momento según una estructura de las mencionadas anteriormente.

La aplicación de esta técnica de programación al diseño consistirá en realizar sólo los primeros niveles de refinamiento sin entrar en detalles propios de los lenguajes de programación.

Programación estructurada de Jackson

La programación estructurada de Jackson (JSP) también sigue las ideas de la programación estructurada en cuanto a las estructuras recomendadas (secuencia, selección, iteración) y el método de refinamientos sucesivos para construir la estructura del programa de forma descendente.

La aportación principal de esta metodología está en la recomendación para construir la estructura del programa, que debe hacerse similar a las estructuras de datos de entrada y salida.

La técnica JSP se basa en los siguientes pasos:

1. Analizar el entorno del problema y describir las estructuras de datos a procesar.

2. Construir la estructura del programa basada en las estructuras de datos.

3. Definir las tareas a realizar en términos de operaciones elementales y situarlas en los módulos apropiados.

Diseño estructurado

Esta técnica de diseño es el complemento de la técnica de análisis estructurado, y por eso es la elegida para realizar el diseño en este proyecto.

Para el diseño se utilizan los diagramas de estructura (ver figura 7.13), que muestran la descomposición jerárquica de los módulos y operaciones del sistema. Los rectángulos representan módulos o funciones. Las líneas indican que un módulo superior llama o utiliza al módulo inferior. El diagrama de estructura no establece ninguna secuencia concreta de utilización de los módulos y tan sólo refleja la organización estática de los mismos.

La tarea del diseño consiste en pasar de los diagramas de flujo de datos (DFD) obtenidos en la fase de análisis a los diagramas de estructura. La principal dificultad de este proceso consiste en establecer una jerarquía o estructura de control entre los módulos, que no está implícita en el modelo funcional descrito mediante los diagramas de flujo de datos.

Para establecer una jerarquía de control razonable entre los diferentes módulos y operaciones descritas en los DFD, la técnica de diseño estructurado

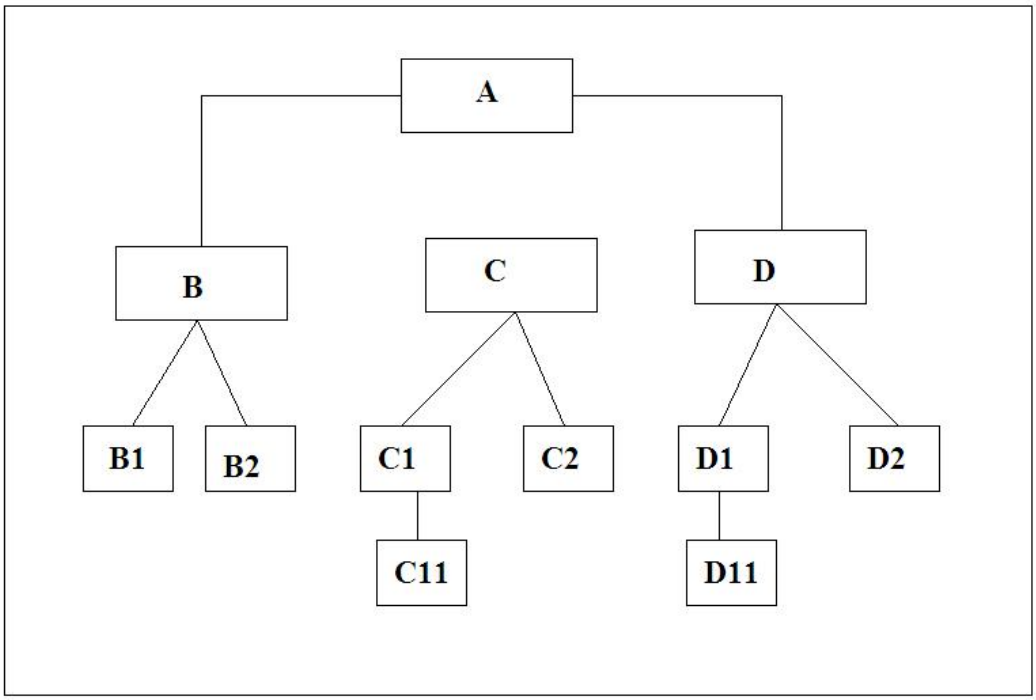

Figura 7.13: Diagrama de Estructura (CC95)

recomienda hacer ciertos análisis del flujo de datos global. En concreto se recomienda realizar los análisis denominados: *flujo de transformación* y *flujo de transacción.*

El análisis del flujo de transformación consiste en identificar en un diagrama de flujo de datos un flujo global de información desde los elementos de entrada al sistema hasta los de salida. Los procesos se deslindan en tres regiones denominadas de flujo de entrada, de transformación y de flujo de salida. Para obtener la estructura modular del programa se asignan módulos para las operaciones del DFD y se añaden módulos de coordinación que realizan el control de acuerdo con la distribución del flujo de transformación. Un ejemplo puede verse en la figura 7.14.

El análisis del flujo de transacción es aplicable cuando el flujo de datos puede descomponerse en varias líneas separadas, cada una de las cuales corresponde a una función o transacción distinta, de manera que sólo una de estas líneas se activa para cada entrada de datos de tipo diferente. El análisis consiste en identificar el llamado centro de transacción, a partir del cual se

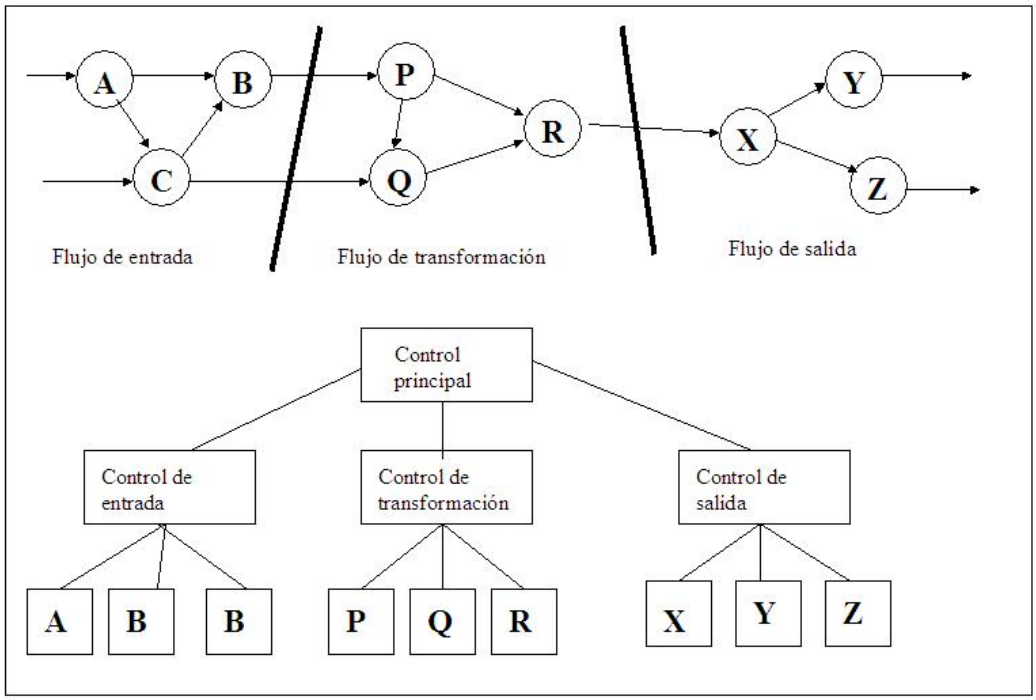

Figura 7.14: Diseño basado en el flujo de transformación (CC95)

ramifican las líneas de flujo. Un ejemplo de análisis de transacción puede verse en la figura 7.15.

El diseño orientado al flujo de datos permite una cómoda transición desde el modelo de análisis a una descripción del diseño de la estructura del programa. La transición desde el flujo de información, representado por un diagrama de flujo de datos, a una estructura se realiza siguiendo los siguientes pasos:

1. Se establece el tipo de flujo de información (transformación o transacción).
2. Se indican los límites del flujo.
3. Se convierte el DFD en la estructura del programa.
4. Se define la jerarquía de control.
5. Se refina la estructura resultante usando heurísticas de diseño. Esto se consigue aumentando o reduciendo el número de módulos para producir

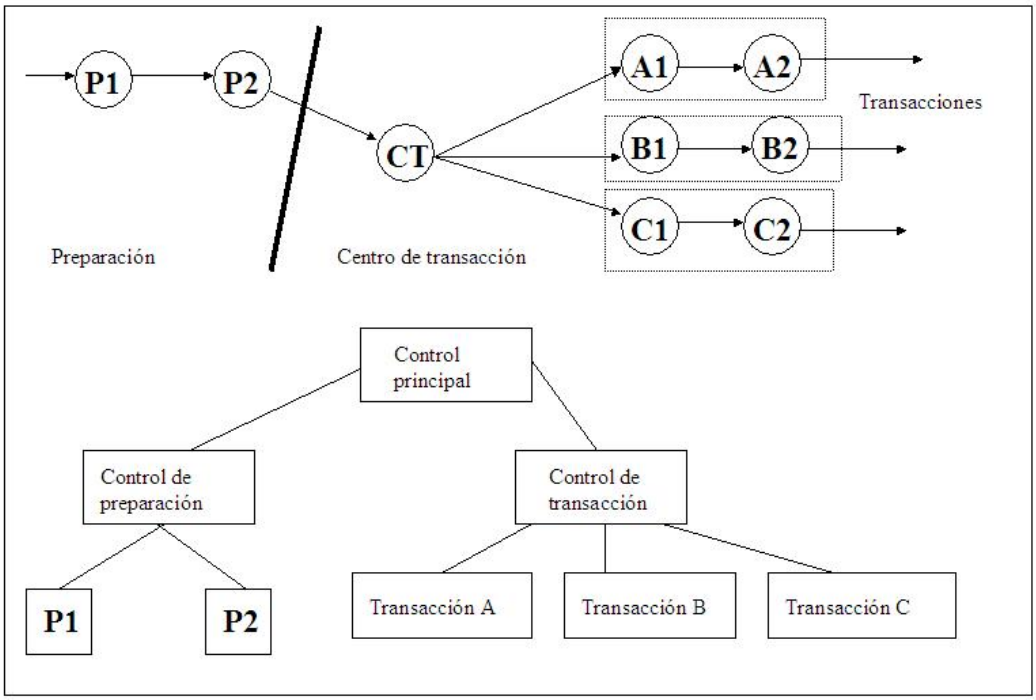

Figura 7.15: Diseño basado en el flujo de transacción (CC95)

una descomposición razonable con una alta cohesión y un acoplamiento mínimo, obteniendo una estructura fácil de implementar, probar y mantener.

7.3.6. Diseño de la interfaz

El diseño arquitectónico proporciona una imagen de la estructura del programa. El diseño de la interfaz muestra las conexiones del sistema tanto internas como externas (Pre97).

El diseño de la interfaz se centra en tres áreas importantes:

1. **La interfaz interna.** El diseño de la interfaz entre los módulos del software depende de los datos que deben fluir entre ellos. Las flechas (objetos de datos) que fluyen hacia y desde cada transformación en un diagrama de flujo de datos del modelo de análisis deben convertirse en un diseño para la interfaz del módulo que corresponda a esa transformación.

2. **La interfaz externa.** El diseño de interfaces entre el software y otros productores o consumidores de información no humanos. El diseño de la interfaz externa comienza con una evaluación de cada entidad externa representada en cada DFD del modelo de análisis. Se determinan los requisitos de datos de la entidad externa y se diseñan las interfaces apropiadas.

3. **La interfaz de usuario.** El diseño de la interfaz entre el hombre y la computadora tiene tanto que ver con el estudio de las personas como con los aspectos de la tecnología.

Diseño de la interfaz hombre-máquina

A la hora de diseñar una interfaz hombre-máquina (IHM) entran en juego cuatro modelos, cuatro visiones distintas del mismo problema.

Modelo de diseño Lo crea el ingeniero e incorpora representaciones de datos, arquitectónicas, de interfaz y procedimentales.

Modelo de usuario muestra el perfil de los usuarios finales del sistema. Los usuarios pueden ser novatos sin conocimientos del sistema, usuarios esporádicos con conocimientos y usuarios frecuentes con conocimientos.

Percepción del sistema (modelo del usuario) Es la imagen que tiene el usuario final del sistema.

Imagen del sistema Combina la manifestación exterior del sistema con toda la información de soporte que describe la sintaxis y la semántica del sistema.

Cuando coinciden la imagen del sistema y la percepción del sistema el usuario se siente a gusto con el software y lo utiliza eficazmente. Para conseguir esta fusión de modelos, el diseño debe desarrollarse para incluir la información del modelo de usuario y la imagen del sistema debe reflejar fielmente la información sintáctica y semántica de la interfaz.

Aspectos del diseño

Hay una serie de aspectos comunes que suelen aparecer en el diseño de la interfaz de usuario, estos son:

El tiempo de respuesta del sistema El tiempo que pasa desde que el usuario solicita una acción hasta que ésta es completada. Dos características importantes son la duración y la variabilidad o desviación del tiempo medio de respuesta.

La facilidad de ayuda al usuario La ayuda puede ser integrada (se diseña con el software desde el principio) o agregada (se añade después de construido el sistema).

La información de errores Los mensajes de error deberían describir el problema, proporcionar información para recuperarse del error, indicar cualquier consecuencia negativa del error e ir acompañado de alguna señal audible o visible.

El etiquetado de órdenes Una de las formas más comunes de interacción entre el usuario y el software del sistema es el uso de órdenes escritas con el teclado. Las funciones software se pueden invocar a través de una secuencia de órdenes del teclado. El etiquetado de órdenes se refiere al formato que deben tener esas secuencias de pulsaciones de teclado. Otra manera de ejecutar estas órdenes es seleccionarlas de un menú con pulsaciones de ratón.

Directrices para el diseño de interfaces

Aunque el diseño de interfaces de usuario se basa profundamente en la experiencia, existen una serie de directrices de diseño de IHM que favorecen el desarrollo de una interfaz eficiente y amigable.

Interacción general :

- Ser consistente.
- Ofrecer respuestas significativas.
- Pedir verificación de cualquier acción destructiva importante.
- Permitir deshacer la mayoría de las acciones.
- Reducir la cantidad de información a memorizar entre acciones.
- Minimizar el número de pulsaciones.
- Autoprotegerse de los posibles errores.
- Proporcionar ayudas sensibles al contexto.
- Usar verbos de acción sencillos para nombrar las órdenes.

Visualización de la información :

- Mostrar sólo información relevante.
- No abrumar al usuario con datos, permitir una rápida asimilación.
- Usar etiquetas consistentes y estándar.

- Permitir al usuario mantener el contexto visual.
- Producir mensajes de error significativos.
- Usar mayúsculas, minúsculas y tabulaciones para facilitar la comprensión.
- Usar ventanas para compartimentar diferentes tipos de información.
- Estudiar la "geografía" disponible en la pantalla y usarla eficientemente.

Entrada de datos :

- Minimizar el número de acciones de entradas de datos que debe realizar el usuario
- Mantener la consistencia entre la visualización y la introducción de datos.
- Permitir al usuario personalizar la entrada.
- La interacción debería ser flexible adaptándose al modo preferido por el usuario.
- Desactivar las órdenes inapropiadas en el contexto actual.
- Dejar que el usuario controle el flujo interactivo.
- Proporcionar ayuda de todas las acciones de entrada.
- Eliminar las entradas innecesarias.

7.3.7. Diseño procedimental

El diseño procedimental es la última actividad a realizar en el diseño tras el diseño de datos, arquitectónico y de interfaz. Se deben definir los detalles de los algoritmos de todas las funciones y procedimientos de la aplicación.

Se trata ahora de elegir una notación para representar este diseño, hay que tener en cuenta que el diseño procedimental debe especificar los detalles procedimentales sin ambigüedades.

Programación estructurada

La programación estructurada usa un conjunto de construcciones lógicas básicas con las que es posible formar cualquier programa. Estas construcciones son: *la secuencia*, *la condición* y *la repetición*. Estas tienen una estructura lógica predecible lo que permite seguir fácilmente el flujo procedimental.

El uso de construcciones estructuradas reduce la complejidad del programa y mejora su comprensión, prueba y mantenimiento.

Notación gráfica para el diseño

Las notaciones gráficas proporcionan unas formas que describen muy bien el detalle procedimental. Sin embargo, si se emplean mal pueden producir una imagen incorrecta que conduzca a un software erróneo.

Algunas notaciones gráficas son:

Diagramas de flujo Se utilizan rectángulos para indicar un paso del proceso, un rombo para indicar una condición lógica y flechas para indicar el flujo de control. Con estos elementos se pueden representar las tres construcciones estructuradas comentadas anteriormente (secuencia, condición y repetición). Las construcciones estructuradas pueden anidarse unas en otras.

Diagramas de cajas El elemento fundamental del diagrama es la caja. Con ellas se construyen las diferentes estructuras de secuencia, condición y repetición. También son fácilmente anidables. Un diagrama de cajas está estratificado en múltiples páginas a medida que se refinan los elementos de procesamiento del módulo.

Notación tabular de diseño

En algunas aplicaciones puede ser necesario un módulo para evaluar una combinación compleja de condiciones y seleccionar las acciones adecuadas basándose en esas condiciones. Las tablas de decisión proporcionan una notación que traduce las acciones y condiciones a una forma tabular.

Para desarrollar una tabla de decisión se siguen los siguientes pasos (Pre97):

1. Se hace una lista con las acciones.
2. Se hace una lista de las condiciones.
3. Se asocian conjuntos específicos de condiciones con acciones específicas.
4. Se definen reglas indicando que acción o acciones ocurren para un conjunto de condiciones.

Lenguaje de diseño de programas

El lenguaje de diseño de programas o seudocódigo se parece a cualquier lenguaje de programación. La diferencia estriba en que en seudocódigo se puede empotrar texto descriptivo directamente en las instrucciones del lenguaje. Además el seudocódigo no puede compilarse.

El seudocódigo debería tener las siguientes características (Pre97):

- Una sintaxis fija de palabras clave.
- Sintaxis libre en lenguaje natural.
- Facilidades para declarar tipos de datos.
- Definición de subprogramas.
- Facilidades para representar la condición y la repetición de la programación estructurada
- Construcciones de entrada/salida.

7.4. Métodos de prueba del software

Durante las fases anteriores de definición y desarrollo se intenta construir el software partiendo de un concepto abstracto y llegando a una implementación tangible. A continuación, en la fase de pruebas, se crean una serie de casos de prueba que intentan hacer fallar el software construido.

La prueba es un proceso de ejecución de un programa con el objetivo de descubrir un error. Una prueba tiene éxito si descubre un error no detectado hasta entonces, al contrario de la idea normal que considera que una prueba tiene éxito si no descubre errores (Pre97).

La prueba demuestra hasta que punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos.

Las técnicas de prueba del software responden a dos estrategias fundamentales que son:

Pruebas de caja negra Se basan en el conocimiento de la función específica para la que fue diseñado el producto, se llevan a cabo pruebas que demuestren que cada función es completamente operativa y buscando errores en la función. Los casos de prueba pretenden demostrar que las funciones software son operativas, que la entrada se acepta de forma adecuada, que se produce un resultado correcto y se asegura la integridad de la información externa.

Pruebas de caja blanca o transparente Se basan en el conocimiento del funcionamiento interno del producto, se desarrollan pruebas que aseguren que la operación interna se ajusta a las especificaciones. Se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software. Se examina el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

7.4.1. Prueba de interfaces gráficas de usuario (IGU)

A medida que el software se hace más complejo, crece la necesidad de enfoques de pruebas especializadas. Los métodos de prueba de caja negra y caja blanca son aplicables a todos los entornos y aplicaciones, pero a veces se necesitan enfoques y directrices únicos para las pruebas.

La complejidad de las interfaces gráficas de usuario es cada vez mayor, lo que origina más dificultad en el diseño y ejecución de casos de prueba.

Dado que las IGU modernas tienen la misma apariencia y filosofía, se pueden dar unas directrices para la creación de una serie de pruebas genéricas.

Ventanas :

- Se abren por teclado o menú
- Se mueven y cambian de tamaño
- El contenido es accesible
- Se regenera al sobreescribir
- Todas las funciones están operativas
- Están disponibles todos los menús, barras, botones, iconos, ...
- Se resalta la ventana activa

Menús emergentes y ratón :

- Se muestra la barra de menú apropiada
- Funcionan lass operaciones de despliegue
- Funcionan los menús y barras de herramientas
- Se muestran las funciones emergentes
- Todas las funciones son accesibles por ratón y por teclado
- El formato de texto es correcto
- Es posible invocar funciones con órdenes alternativas
- Se resaltan o difuminan las funciones según la ventana actual
- Se ejecutan las funciones de menú anunciadas
- Los nombres del menú son claros
- Existe ayuda disponible
- Se reconocen apropiadamente las operaciones del ratón
- Se reconocen apropiadamente los botones del ratón
- Cambia adecuadamente el cursor o puntero al invocar diferentes operaciones

Entrada de datos :

- Se introducen adecuadamente los datos alfanuméricos
- Los modos gráficos de entrada de datos funcionan correctamente
- Se reconocen los datos no válidos
- Los mensajes de entrada de datos son inteligibles

Capítulo 8

Elección de modelos, metodologías, notaciones y tecnologías

8.1. Elección del modelo de desarrollo

Las características del sistema a desarrollar son:

- Los requisitos del sistema están completamente definidos.
- No se espera que el cliente modifique los requisitos del sistema durante el desarrollo.
- El proyecto es similar a muchos otros ya realizados.
- El cliente no necesita una versión rápida del sistema.
- El producto se entregará completo al final del desarrollo.
- La duración del proyecto no es muy larga.

Se elige el **modelo secuencial para el desarrollo del software** ya que estas características se ajustan bastante bien a las que se recomiendan para seguir dicho modelo.

Se elige el **modelo de ciclo de vida en cascada**, que es el más representativo de los modelos secuenciales.

Se adapta el modelo al problema particular suprimiendo las fases del modelo que no son aplicables al mismo.

Estudio de viabilidad Las necesidades de la empresa no son complejas, consisten básicamente en desarrollar una base de datos sencilla a la que se accede con peticiones también sencillas. Esto lleva a estimar un tiempo corto de desarrollo. Por otra parte el equipo de análisis, diseño y desarrollo consta de una sola persona que tendrá que realizar todas esas funciones. Comparando las necesidades de la empresa con los recursos de ingeniería disponibles se determina que el proyecto es viable en un plazo de tiempo no superior a 6 meses.

Fase de integración Se puede eliminar, pues no existen diferentes partes o módulos que haya que desarrollar por separado.

Fase de mantenimiento Se puede eliminar, ya que con la entrega del trabajo finaliza la relación del desarrollador con el cliente.

En resumen el modelo de ciclo de vida en cascada constará de las siguientes fases:

- **Análisis de requisitos.**
- **Diseño.**
- **Implementación.**
- **Pruebas.**

8.2. Elección de la metodología de análisis de requisitos

El análisis de requisitos permite construir modelos de los dominios de datos, funcional y de comportamiento del sistema.

En este proyecto se usará básicamente el **análisis estructurado** complementado con algún elemento tomado del análisis orientado a objetos, como son los **casos de uso**.

Los elementos utilizados son:

Diccionario de datos Un depósito con las definiciones de los datos. La notación utilizada para el diccionario de datos puede verse en la figura 7.10 (página 68).

DER Diagrama Entidad-Relación, utilizado para modelar los datos. La notación de un DER se puede ver en la figura 7.7 (página 65).

Descripción de objetos de datos Describe los atributos de los objetos del DER.

DFD Diagrama de Flujo de Datos, es modelo funcional. La notación básica de un DFD se puede ver en la figura 7.8 (página 66).

EP Especificación de Proceso, describe la función representada en el DFD.

DTE Diagrama de Transición de Estados, modela el comportamiento. La notación básica de un DTE se puede ver en la figura 7.9 (página 67).

EC Especificación de Control, describe los aspectos de control.

Casos de uso Para modelar los requisitos funcionales del sistema, haciendo énfasis en el comportamiento. Especifica cómo el usuario interactúa con el sistema.

8.3. Elección de la metodología de diseño

Con el diseño se trata de definir el sistema con el suficiente detalle para permitir su realización física.

La metodología que se usará en este trabajo será la que se basa en el **flujo de datos**. En la fase de análisis se eligió como técnica de desarrollo el análisis estructurado que también es orientado a flujo de datos y hay técnicas desarrolladas para pasar fácilmente de los modelos de análisis a los modelos de diseño.

8.3.1. Diseño de datos

El proceso de diseño consistirá en convertir el modelo de datos del análisis, un diagrama entidad-relación, en una base de datos relacional. Se trata de convertir las entidades y las relaciones en tablas de la base de datos del sistema.

8.3.2. Diseño arquitectónico

El objetivo principal del diseño arquitectónico es desarrollar una estructura modular del programa y representar las relaciones de control entre los módulos.

Se elige la técnica de **diseño estructurado**, que es el complemento de la técnica de análisis estructurado utilizada en el análisis de requisitos.

Para el diseño se utilizan los diagramas de estructura (ver figura 7.13, página 79). La tarea del diseño consiste en pasar de los diagramas de flujo de datos (DFD) obtenidos en la fase de análisis a los diagramas de estructura.

8.3.3. Notación para el diseño procedimental

En el diseño procedimental se transforman los elementos estructurales en una descripción procedimental de los componentes. Se deben definir los detalles de los algoritmos de todas las funciones y procedimientos de la aplicación.

El **seudocódigo** será la notación elegida para representar el diseño procedimental en este proyecto.

Para ello se utilizará la siguiente sintaxis:

- Las acciones se representan en lenguaje natural.
- La secuencia se representa colocando una acción a continuación dde otra.
- La condición se construye:

  SI condición ENTONCES
  lista de acciones
  SINO
  lista de acciones
  FIN-SI

- La repetición se construye:

  PARA-CADA lista de elementos HACER
  lista de acciones
  FIN-PARA

- El bucle con condición inicial se construye:

  MIENTRAS condición HACER
  lista de acciones
  REPETIR

- El bucle con condición final se construye:

  REPETIR
  lista de acciones
  HASTA condición

- La selección múltiple se construye:

  SELECCIÓN tipoopción
  CASO opción
  lista de acciones
  FIN-CASO
  .....
  DEFECTO
  lista de acciones
  FIN-SELECCIÓN

- Todas estas construcciones son anidables unas en otras.

- Para invocar una función o procedimiento se utiliza LLAMA

8.4. Elección de tecnologías para la implementación

Se considera que las tecnologías adecuadas para realizar el Proyecto son las siguientes:

Base de Datos Como servidor de Bases de Datos se usará MySQL, ya que está ampliamente extendido, es de código abierto y fácil de administrar. Además está muy bien integrado con PHP que es el lenguaje de programación elegido para el lado del servidor.

El software se puede descargar a través de Internet en, http://www.mysql.com.

Servidor *Web* Como servidor *Web* se elige Apache HTML que es de libre distribución y es también muy ampliamente utilizado.

El software se puede descargar a través de Internet en, http://www.apache.org.

Lenguaje de Programación Para programar la aplicación en el servidor se elige PHP, que es un lenguaje sencillo, potente, desarrollado precisamente para páginas *Web* dinámicas, con interfaz sencilla para acceder a Base de Datos (muy integrado con MySQL) y que además es de libre distribución. Como alternativa interesante al uso de PHP se tiene el lenguaje Perl, pero no se considera su utilización debido a que el código no se integra en el fichero HTML sino que está en ficheros separados, lo que dificultaría el desarrollo y el mantenimiento.

El software se puede descargar a través de Internet en, http://www.php.net.

Procesador de texto Para generar la documentación del Proyecto se usará LaTeX, utilizando los entornos para Windows MiKTeX y WinEdt. Es un programa muy potente y robusto que facilita mucho la edición del texto y la gestión de capítulos y secciones, así como las citas y la bibliografía.

El software se puede descargar a través de Internet en, http://www.miktex.org, y en, http://www.winedt.com.

8.5. Elección de métodos de prueba

Las pruebas de caja negra de la aplicación consistirán en analizar el comportamiento de las diferentes funciones del sistema con datos de entrada correctos e incorrectos y comprobar que responde de manera adecuada, es decir, con los resultados de la operación cuando los datos son correctos y con mensajes de error adecuados cuando son incorrectos. Como la aplicación funciona básicamente con la pulsación de botones, será a estas pulsaciones a las que se les apliquen las pruebas.

Las pruebas de caja blanca de la aplicación consistirán en analizar el funcionamiento interno de las distintas funciones comprobando que los valores de retorno son adecuados y correctos y que no hay caminos internos que no producen ninguna acción o retorno.

Capítulo 9

Documento de especificación de requisitos

En este capítulo se desarrolla el análisis de requisitos del problema objeto del proyecto. Se desarrollan los modelos completos del análisis, se especifican todos los requisitos que debe satisfacer el sistema y se detallan los casos de uso de los diferentes actores que interactúan con el sistema. Este capítulo es el Software Requirement Document (SRD), Documento de Especificación de Requisitos) del proyecto.

9.1. Objetivo

El objetivo del sistema es facilitar la gestión, vía *Web*, del catálogo de una pequeña tienda de discos. Tendrán acceso al sistema: administradores, almacenistas, vendedores y clientes. Estos usuarios realizarán funciones de alta y baja de usuarios, listado de autores y discos, reservas de discos, ventas, altas y bajas de discos.

9.2. Ámbito

El sistema se denominará Catálogo *Web*, y consistirá en una serie de páginas *Web* que incluirán todas las funciones necesarias.

La navegación entre páginas tendrá en cuenta el tipo de usuario para permitir el acceso y poder realizar las funciones permitidas a ese usuario.

Aunque en el sistema se podrán realizar ventas y reservas, estas afectarán únicamente al número de existencias de los discos en la Base de Datos. No se realizará ningún tipo de gestión económica.

9.3. Panorámica del documento de especificación de requisitos

El resto de este documento contiene la descripción del modelo del sistema, y la lista de requisitos específicos.

Para desarrollar el modelo se ha utilizado la metodología de Análisis Estructurado complementada con un estudio de los Casos de Uso tomados del Análisis Orientado a Objetos.

En la sección 7 se incluyen los diagramas entidad-relación, en la sección 8 los de flujo de datos, en la sección 9 el de transición de estados, en la sección 10 el diccionario de datos y en la sección 11 los casos de uso del sistema. Las especificaciones de todas las funciones se incluyen en el apartado de requisitos funcionales de la sección 12.

9.4. Objetivo y funciones

La tienda dispone de una base de datos con los discos disponibles y con la relación de usuarios y contraseñas de acceso al sistema.

Los tipos de usuario son: cliente, vendedor, almacenista y administrador.

El cliente puede:

- Consultar discos.
- Consultar autores.
- Reservar discos.

El vendedor puede:

- Realizar consultas de discos.
- Realizar consultas de autores.
- Realizar ventas (disminuir existencias en la base de datos).

El almacenista puede:

- Dar de alta discos en la base de datos.
- Dar de baja discos en la base de datos.

- Realizar consultas de discos.
- Realizar consultas de autores.

El administrador puede:

- Dar de alta usuarios.
- Dar de baja usuarios.
- Modificar datos de usuarios.
- Realizar las funciones propias del vendedor.
- Realizar las funciones propias del almacenista.

9.5. Relaciones con otros sistemas

La base de datos del sistema estará ubicada en un servidor de bases de datos MySQL. Será necesario disponer de una cuenta y una contraseña conocidos para acceder a dicho sistema.

Las páginas *Web* estarán ubicadas en un servidor *Web* Apache HTML.

Los servidores *Web* y de base de datos deberán estar configurados para el perfecto funcionamiento de la aplicación.

9.6. Restricciones generales

- El sistema se desarrollará siguiendo la metodología de análisis y diseño estructurado, complementado con el análisis de casos de uso.
- Se usará una base de datos relacional.
- Para la implementación de las páginas *Web* se usará el lenguaje PHP. Este lenguaje es de libre distribución y puede descargarse de su página *Web* (http://www.php.net).
- La aplicación deberá ejecutarse en cualquier navegador, es decir, producirá HTML estándar (W3C05). HTML es el lenguaje utilizado para generar páginas *Web*.

9.7. Creación del diagrama entidad-relación

El estudio del enunciado del problema lleva a considerar que los objetos de datos que entran y salen del sistema son dos: *discos* y *usuarios*. Podría pensarse que *autor* es una entidad independiente de discos pero en nuestro caso la única información que necesitamos de autor es su nombre por lo que se puede incluir como atributo del objeto discos sin ningún problema, otra cosa sería que se incluyeran más datos del autor como dirección, nacionalidad, etc.

En cuanto a las relaciones que conectan estas entidades, se puede definir la relación *Reserva*, que se produce cuando un cliente reserva un disco. Esta relación tendrá una cardinalidad muchos a muchos puesto que un cliente puede reservar varios discos y un disco puede ser reservado por varios clientes. La modalidad será opcional puesto que un cliente puede, o no, reservar un disco.

Los atributos de la entidad **Discos** serán:

IDdisco Identificador único del disco. Obligatorio
Nombre Nombre del disco. Obligatorio
Autor Nombre del autor. Obligatorio
Existencias Número de discos en la base de datos. Obligatorio
Reservados Número de reservas, que será menor o igual que Existencias. Obligatorio
Año Año de publicación. Opcional
Duración Duración del disco. Opcional
Discográfica Compañía discográfica que edita el disco. Opcional

Los atributos de la entidad **Usuarios** serán:

IDusuario Identificador único del usuario. Obligatorio
Usuario Nombre del usuario. Obligatorio
Contraseña Contraseña del usuario. Obligatorio
Tipousuario El tipo del usuario. Obligatorio

Los atributos de la relación **Reserva** serán:

IDdisco Identificador único del disco. Obligatorio
IDusuario identificador único del usuario. Obligatorio
Fecha fecha de la reserva. Opcional

El diagrama entidad-relación que se obtiene del análisis puede verse en la figura 9.1

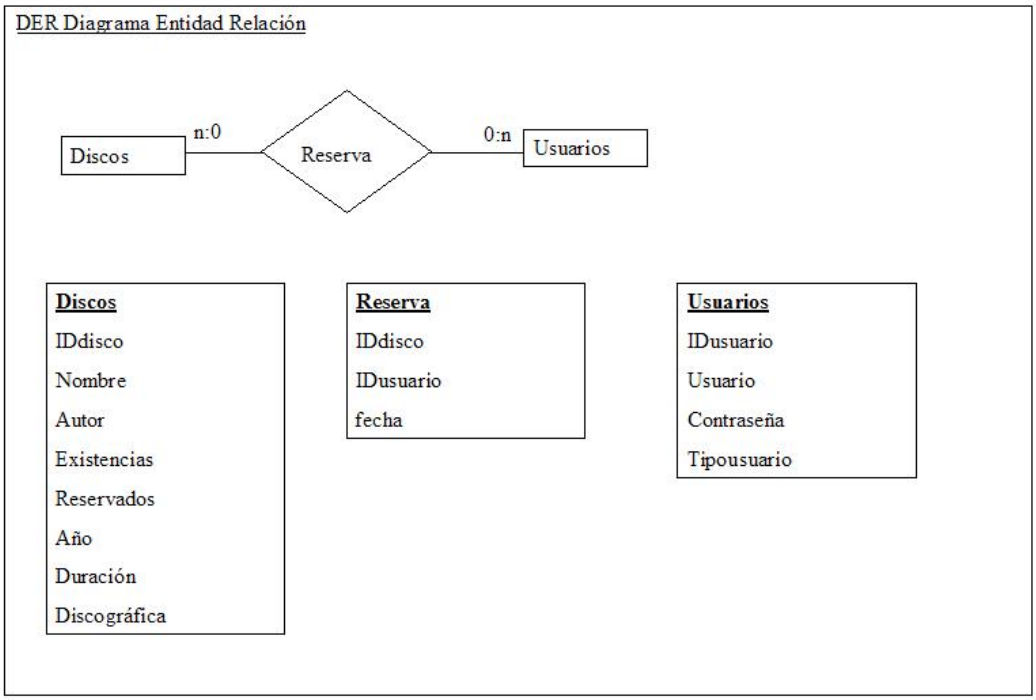

Figura 9.1: Diagrama Entidad-Relación del Catalogo *Web*

9.8. Creación del modelo de flujo de datos

Para operar el sistema se dispondrá de un terminal con pantalla, teclado y ratón. El sistema operará mediante una serie de páginas *Web* que contendrán la funcionalidad adecuada a cada tipo de usuario. La navegación entre páginas se realizará por una serie de botones. Las funciones se activarán también mediante botones. Las pantallas tendrán las casillas necesarias para introducir y presentar los datos, así como listas para mostrar los listados requeridos.

Para representar el modelo funcional del sistema se parte de un diagrama de flujo de datos, DFD, de nivel superior o de contexto que representa a todo el sistema. Posteriormente se explotan las burbujas de cada DFD descendiendo en la jerarquía hasta que obtenemos funciones lo suficientemente simples para poder ser descritas el lenguaje claro, terminando así el proceso de refinamiento.

El DFD0 o de contexto consta de una sola burbuja que representa al sistema. Las entidades externas son los usuarios del sistema y la pantalla de

la terminal. Las entradas al sistema son las órdenes dadas por los usuarios y las salidas son los mensajes y los listados mostrados en pantalla. El diagrama de contexto puede verse en la figura 9.2

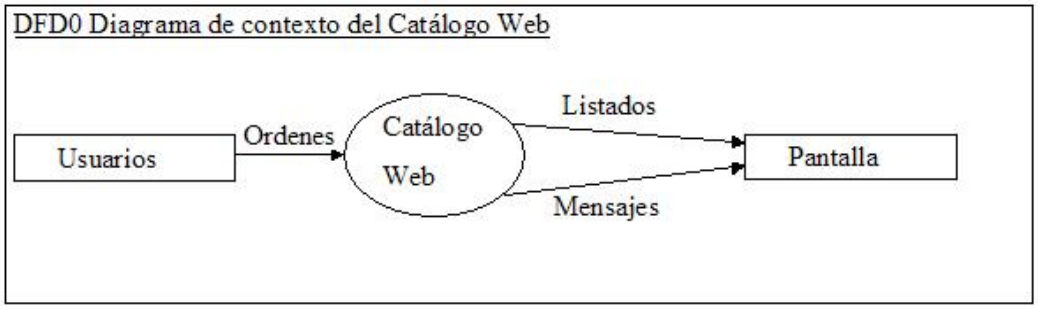

Figura 9.2: Diagrama de contexto del Catálogo *Web*

El siguiente paso es explotar la burbuja Catálogo *Web* para obtener los principales grupos funcionales de la aplicación. Hay que tener en cuenta que para mantener la consistencia del modelo todos los datos que entran o salen de una burbuja deben entrar o salir del DFD que se obtiene al explotar dicha burbuja. En nuestro caso los grupos funcionales serán los que recojan las funciones que pueden realizar los cuatro tipos de usuario que tiene el sistema; clientes, vendedores, almacenistas y administradores. El DFD.1 obtenido puede verse en la figura 9.3

Ahora explotamos las cuatro burbujas del DFD.1 obteniendo los DFD de nivel 2, que en nuestro caso tendrán ya suficiente detalle para poder describir sus funciones directamente en lenguaje claro.

Estos grupos funcionales se describen a continuación. La descripción precisa de cada función se describe en la sección "requisitos funcionales" 9.12.1 (página 114).

9.8.1. Gestión Clientes

El DFD.2.1 de la burbuja Gestión Clientes se muestra en la figura 9.4.

Las funciones de Gestión Clientes son las siguientes:

Función 1.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de clientes.

Función 1.2 Validar contraseña: Valida el usuario y contraseña introducidos.

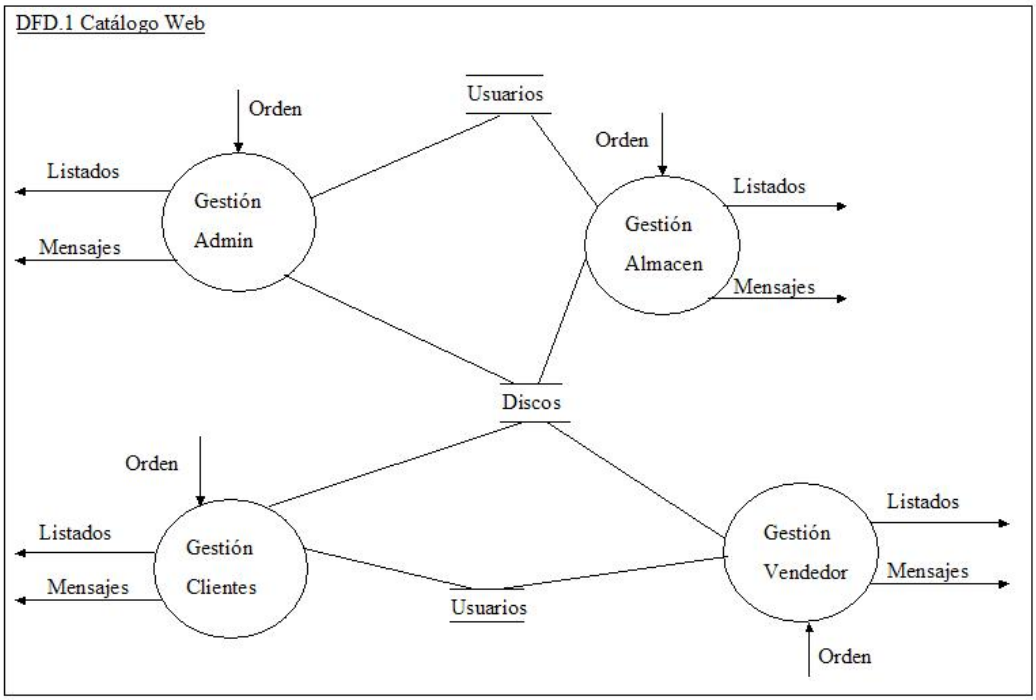

Figura 9.3: DFD.1 Catálogo *Web*

Función 1.3 Ver pantalla cliente: Muestra la pantalla de clientes.

Función 1.4 Ver lista autores: Muestra un listado de autores disponibles.

Función 1.5 Ver autor: Muestra un listado de los discos de un autor.

Función 1.6 Ver disco: Muestra los datos de un disco.

Función 1.7 Reservar disco: Realiza la reserva de un disco.

Una descripción completa de todas estas funciones se encuentra en la sección "requisitos funcionales" 9.12.1 (página 114).

9.8.2. Gestión Administradores

El DFD.2.2 de la burbuja Gestión Admin se muestra en la figura 9.5.

Las funciones de Gestión Admin son las siguientes:

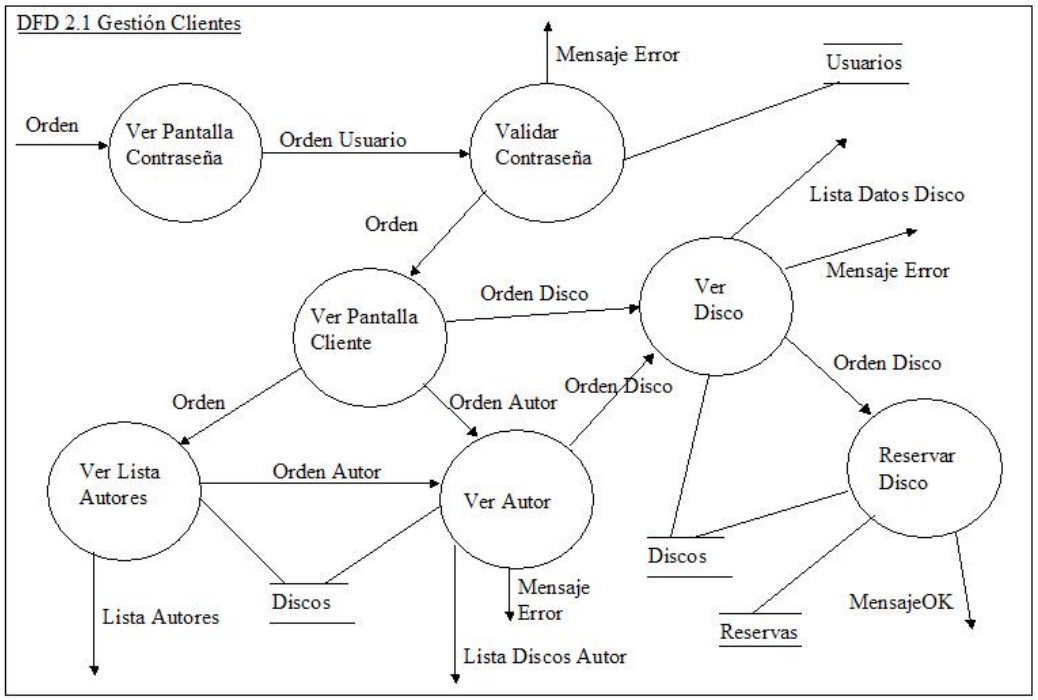

Figura 9.4: DFD.2.1 Gestión Clientes

Función 2.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de administradores.

Función 2.2 Validar contraseña: Valida el usuario y contraseña introducidos.

Función 2.3 Ver pantalla admin: Muestra la pantalla de administradores.

Función 2.4 Ver lista usuarios: Muestra un listado de los usuarios del sistema.

Función 2.5 Dar alta usuario: Da de alta un usuario en la base de datos del sistema.

Función 2.6 Dar baja usuario: Da de baja un usuario en la base de datos del sistema.

Función 2.7 Modificar usuario: Modifica un usuario en la base de datos del sistema.

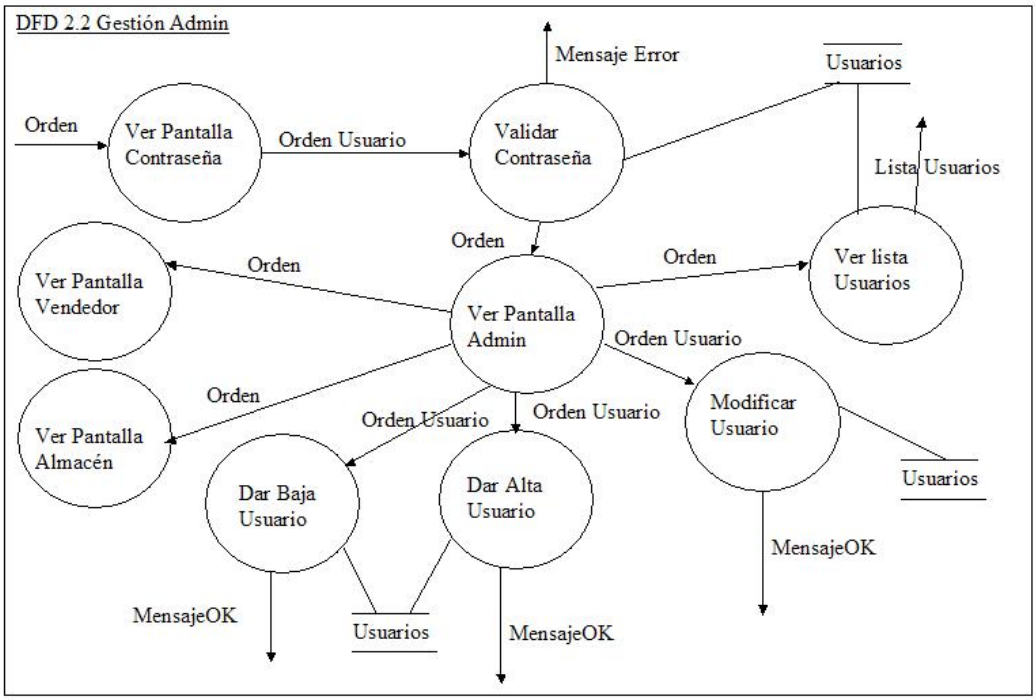

Figura 9.5: DFD.2.2 Gestión Admin

Función 2.8 Ver pantalla vendedor: Muestra la pantalla de vendedores.

Función 2.9 Ver pantalla almacén: Muestra la pantalla del almacén.

Hay que tener en cuenta que los administradores tienen privilegios para hacer también las funciones que hacen los vendedores y los almacenistas, por eso hay un par de funciones que redirigen a las pantallas correspondientes, donde podrán realizar todo lo que ellos puedan realizar.

9.8.3. Gestión Vendedor

El DFD.2.3 de la burbuja Gestión Vendedor se muestra en la figura 9.6.

Las funciones de Gestión Vendedor son las siguientes:

Función 3.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de vendedores.

Función 3.2 Validar contraseña: Valida el usuario y contraseña introducidos.

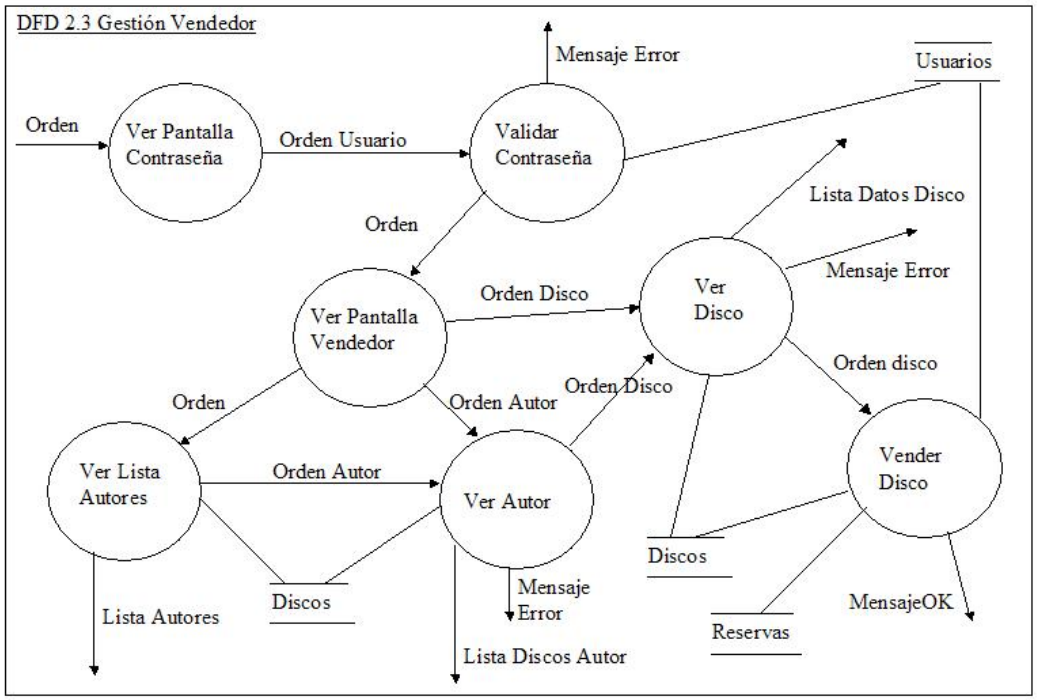

Figura 9.6: DFD.2.3 Gestión Vendedor

Función 3.3 Ver pantalla vendedor: Muestra la pantalla de vendedores.

Función 3.4 Ver lista autores: Muestra un listado de autores disponibles.

Función 3.5 Ver autor: Muestra un listado de los discos de un autor.

Función 3.6 Ver disco: Muestra los datos de un disco.

Función 3.7 Vender disco: Realiza la venta de un disco.

9.8.4. Gestión Almacén

El DFD.2.4 de la burbuja Gestión Almacén se muestra en la figura 9.7.

Las funciones de Gestión Almacén son las siguientes:

Función 4.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de almacenistas.

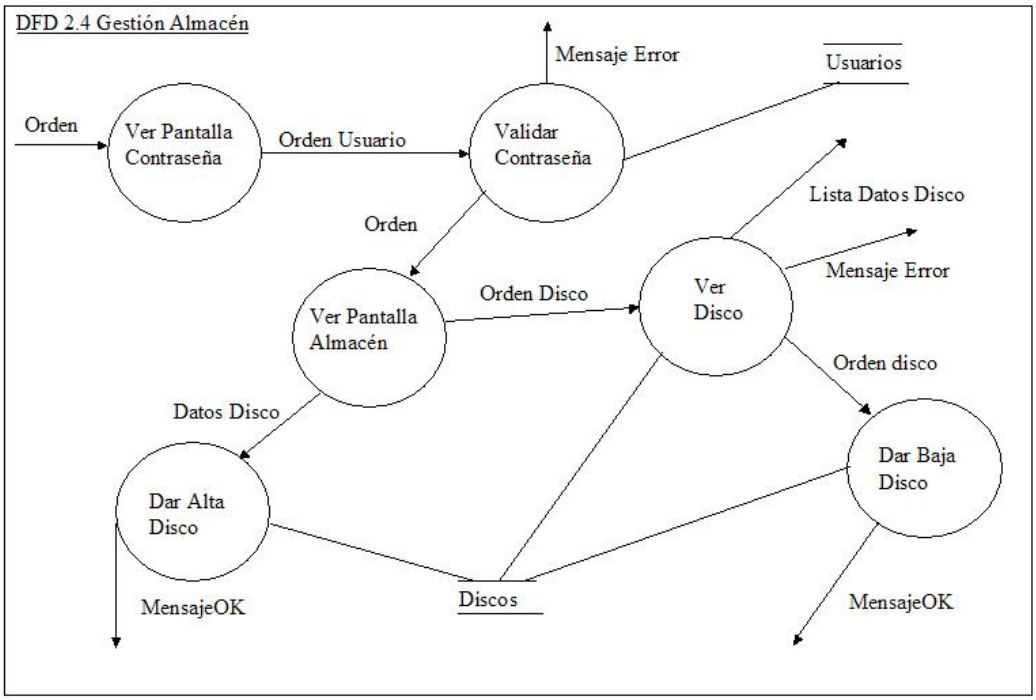

Figura 9.7: DFD.2.4 Gestión Almacén

Función 4.2 Validar contraseña: Valida el usuario y contraseña introducidos.

Función 4.3 Ver pantalla almacén: Muestra la pantalla de almacén.

Función 4.4 Dar alta disco: Da de alta un disco en la base de datos.

Función 4.5 Dar baja disco: Da de baja un disco en la base de datos.

Función 4.6 Ver disco: Muestra los datos de un disco.

9.9. Diagrama de transición de estados

El diagrama de transición de estados (DTE) se utiliza para modelar el comportamiento del sistema, el DTE es una especificación secuencial de este comportamiento. El DTE del sistema puede verse en la figura 9.8

Las flechas de transición etiquetadas muestran cómo responde el sistema a los sucesos a medida que pasa por los diferentes estados.

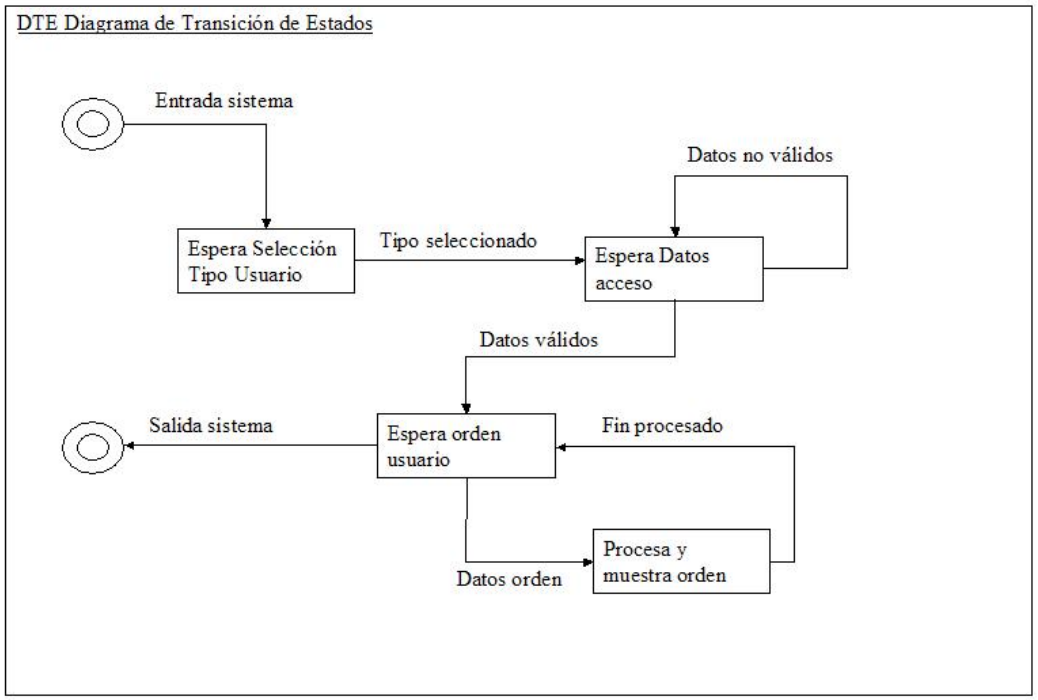

Figura 9.8: Diagrama de Transición de Estados

9.10. Diccionario de datos

El diccionario de datos proporciona un enfoque organizado para representar las características de los objetos de datos. Consiste en un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas, que permiten que tanto los usuarios del sistema como los analistas y diseñadores tengan una visión uniforme de las entradas, salidas, almacenes de datos y cálculos intermedios. El diccionario de datos del sistema puede verse en la tabla 9.1

9.11. Casos de uso

Una vez obtenidos los requisitos funcionales del sistema en las secciones anteriores se pueden crear los casos de uso que aportan una descripción acerca de cómo el sistema será usado.

Los actores se corresponden con los posibles usuarios del sistema y serán por tanto el cliente, el vendedor, el almacenista y el administrador. Como hay

DATO	***DESCRIPCIÓN***
DISCOS	{FICHA-DISCO}
RESERVA	{FICHA-RESERVA}
USUARIOS	{FICHA-USUARIO}
FICHA-DISCO	ID-DISCO + NOMBRE-DISCO + DATOS-DISCO
FICHA-RESERVA	ID-DISCO + ID-USUARIO + FECHA
FICHA-USUARIO	ID-USUARIO + DATOS-USUARIO
ID-DISCO	NUMERO
ID-USUARIO	NUMERO
NOMBRE-DISCO	/nombre del disco/
DATOS-DISCO	/autor, existencias, reservados, año, duración, discográfica/
DATOS-USUARIO	/usuario, contraseña, tipousuario/
FECHA	/fecha/
ORDENES	[ORDEN | ORDEN-AUTOR | ORDEN-DISCO | ORDEN-USUARIO]
LISTADOS	[LISTA-DISCOS-AUTOR |LISTA-AUTORES | LISTA-DATOS-DISCO | LISTA-USUARIOS]
MENSAJES	[MENSAJEOK | MENSAJE-ERROR]
ORDEN	/orden sin parámetros, selección de alternativas o menús/
ORDEN-AUTOR	/nombre autor/
ORDEN-DISCO	NOMBRE-DISCO
ORDEN-USUARIO	DATOS-USUARIO
LISTA-DISCOS-AUTOR	{NOMBRE-DISCO}
LISTA-AUTORES	/lista de nombres de autores/
LISTA-DATOS-DISCO	FICHA-DISCO
LISTA-USUARIOS	USUARIOS
MENSAJEOK	/indica si la operación se realizó o no/
MENSAJE-ERROR	/mensaje de error de la operación/
NUMERO	{DÍGITO}
DÍGITO	[0|1|2|3|4|5|6|7|8|9]

Cuadro 9.1: Diccionario de datos del sistema

una serie de funciones que puede realizar cualquier tipo de usuario, como ver un disco por ejemplo, se define un quinto actor que representa a todos los tipos de usuario y que se denomina usuario genérico.

Se identifican los siguientes casos de uso asociados a sus respectivos actores:

usuario genérico Acceder al sistema, ver lista autores, ver autor, ver disco

cliente reservar disco

vendedor vender disco

almacenista dar alta disco, dar baja disco

administrador dar baja usuario, dar alta usuario, modificar usuario, ver lista usuarios, vender disco, dar alta disco, dar baja disco

Estos casos de uso están representados gráficamente en la figura 9.9

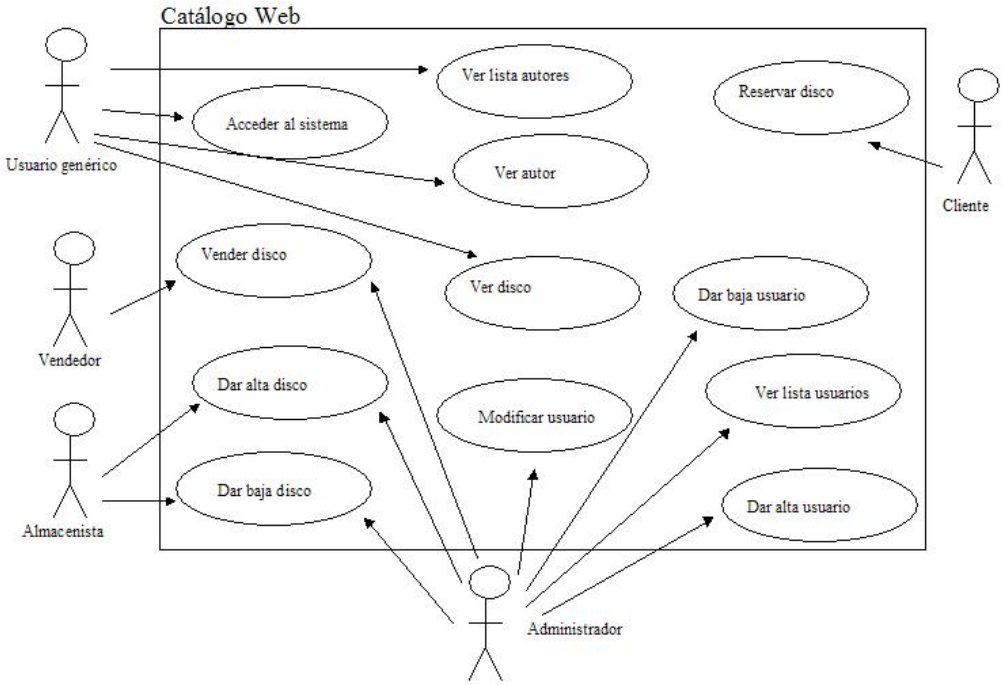

Figura 9.9: Casos de uso del sistema Catálogo *Web*

A continuación se describen detalladamente los casos de uso. Para cada uno se especifica su nombre y el actor que interviene en el caso de uso. Se especifican los pasos de la ejecución normal del caso de uso y cuando sea

necesario las posibles alternativas que pueden aparecer como posibles errores o datos mal introducidos.

El caso de uso "Acceder al sistema" lo utiliza cualquier tipo de usuario pues todos deben autentificarse para acceder, por eso el actor es un usuario genérico.

Caso de Uso: Acceder al Sistema	
Actor: Usuario Genérico	
Curso normal	**Alternativas**
1) El usuario selecciona el tipo de usuario con que quiere acceder al sistema	
2) El sistema muestra la pantalla de introducción de contraseña para ese tipo de usuario	
3) El usuario introduce su nombre de usuario y su contraseña	
4) El sistema accede a la base de datos y comprueba los datos introducidos	4.1) Si falla el acceso a la base de datos se muestra mensaje de error
5) El sistema muestra la pantalla correspondiente al tipo de usuario seleccionado	5.1) Si los datos son incorrectos el sistema muestra un mensaje de error

Los casos de uso "Ver Lista Autores", "Ver Autor" y "Ver Disco" también pueden ser utilizados por cualquier tipo de usuario y se usa un usuario genérico.

Caso de Uso: Ver Lista Autores	
Actor: Usuario Genérico	
Curso normal	**Alternativas**
1) El usuario pulsa el botón "Ver lista de autores"	
2) El sistema accede a la base de datos, genera una lista de autores y la muestra en pantalla	2.1) Si falla el acceso a la base de datos se muestra mensaje de error

Caso de Uso: Ver Autor	
Actor: Usuario Genérico	
Curso normal	**Alternativas**
1) El usuario introduce el nombre del autor en la casilla correspondiente	1.1) El usuario selecciona un autor de una lista generada previamente
2) El usuario pulsa el botón "Ver autor"	
3) El sistema accede a la base de datos y muestra una lista con los discos de ese autor	3.1) Si no hay discos de ese autor se muestra un mensaje indicándolo 3.2) Si falla el acceso a la base de datos se muestra mensaje de error

Caso de Uso: Ver Disco	
Actor: Usuario Genérico	
Curso normal	**Alternativas**
1) El usuario introduce el nombre de un disco en la casilla correspondiente	1.1) El usuario selecciona un disco de una lista de discos de un autor generada previamente
2) El usuario pulsa el botón "Ver disco"	
3) El sistema accede a la base de datos y muestra los datos del disco seleccionado	3.1) Si el disco no está en la base de datos se muestra un menaje indicándolo 3.2) Si falla el acceso a la base de datos se muestra mensaje de error

Caso de Uso: Reservar Disco	
Actor: Cliente	
Curso normal	**Alternativas**
1) El Cliente mediante el caso de uso "Ver Disco" selecciona un disco	
2) El usuario pulsa el botón "Reservar disco"	
3) El sistema accede a la base de datos y comprueba que hay existencias sin reservar de ese disco	3.1) Si no hay existencias sin reservar se muestra un mensaje indicándolo y se termina el caso de uso 3.2) Si falla el acceso a la base de datos se muestra mensaje de error
4) El sistema incrementa el contador de reservados de ese disco	
5) El sistema crea una nueva entrada en la base de datos anotando la reserva de ese disco por ese cliente y muestra un mensaje indicándolo	5.1) Si falla el acceso a la base de datos se muestra mensaje de error

Caso de Uso: Vender Disco	
Actor: Vendedor, Administrador operando como vendedor	
Curso normal	**Alternativas**
1) El Vendedor mediante el caso de uso "Ver Disco" selecciona un disco elegido por un cliente	1.1) Con el nombre de usuario y nombre del disco recupera una reserva
2) El Vendedor comprueba que hay existencias o reservas	2.1) Si no hay existencias o reservas termina el caso de uso
3) El Vendedor pulsa el botón "Vender disco"	
4) El sistema accede a la base de datos y decrementa en una unidad las existencias y/o elimina la reserva de ese cliente	4.1) Si falla el acceso a la base de datos se muestra mensaje de error
5) El sistema muestra un mensaje de operación realizada	

Caso de Uso: Dar Alta Disco	
Actor: Almacenista, Administrador operando como almacenista	
Curso normal	**Alternativas**
1) El Almacenista introduce los datos del disco	
2) El Almacenista pulsa el botón "Dar Alta Disco"	
3) El sistema comprueba que no faltan campos obligatorios	3.1) Si faltan campos obligatorios el sistema avisa
4) El sistema accede a la base de datos y genera una nueva entrada con los datos aportados	4.1) Si falla el acceso a la base de datos se muestra mensaje de error
5) El sistema muestra un mensaje de operación realizada	

Caso de Uso: Dar Baja Disco	
Actor: Almacenista, Administrador operando como almacenista	
Curso normal	**Alternativas**
1) El Almacenista mediante el caso de uso "Ver Disco" selecciona un disco	
2) El Almacenista pulsa el botón "Dar Baja Disco"	
3) El sistema accede a la base de datos y elimina la entrada con los datos aportados	3.1) Si falla el acceso a la base de datos se muestra mensaje de error
4) El sistema muestra un mensaje de operación realizada	

Caso de Uso: Dar Alta Usuario	
Actor: Administrador	
Curso normal	**Alternativas**
1) El Administrador introduce los datos del usuario	
2) El Administrador pulsa el botón "Dar Alta Usuario"	
3) El sistema comprueba que no faltan campos obligatorios	3.1) Si faltan campos obligatorios el sistema avisa
4) El sistema accede a la base de datos y genera una nueva entrada con los datos aportados	4.1) Si falla el acceso a la base de datos se muestra mensaje de error
5) El sistema muestra un mensaje de operación realizada	

Caso de Uso: Dar Baja Usuario	
Actor: Administrador	
Curso normal	**Alternativas**
1) El Administrador introduce el nombre del usuario	
2) El Administrador pulsa el botón "Dar Baja Usuario"	
3) El sistema accede a la base de datos y elimina la entrada con los datos aportados	3.1) Si falla el acceso a la base de datos se muestra mensaje de error
4) El sistema muestra un mensaje de operación realizada	

Caso de Uso: Modificar Usuario	
Actor: Administrador	
Curso normal	**Alternativas**
1) El Administrador introduce los datos del usuario	
2) El Administrador pulsa el botón "Modificar Usuario"	
3) El sistema comprueba que no faltan campos obligatorios	3.1) Si faltan campos obligatorios el sistema avisa
4) El sistema accede a la base de datos y sobrescribe la entrada con los datos aportados	4.1) Si falla el acceso a la base de datos se muestra mensaje de error
5) El sistema muestra un mensaje de operación realizada	

Caso de Uso: Ver Lista Usuarios	
Actor: Administrador	
Curso normal	**Alternativas**
1) El Administrador pulsa el botón "Ver lista de usuarios"	
2) El sistema accede a la base de datos, genera una lista de usuarios y la muestra en pantalla	2.1) Si falla el acceso a la base de datos se muestra mensaje de error

9.12. Requisitos específicos

Todos los requisitos se consideran obligatorios, salvo que se indique lo contrario.

9.12.1. Requisitos funcionales

Almacenamiento de datos

El sistema mantendrá permanentemente almacenados los datos indicados en DISCOS, USUARIOS y RESERVAS.

Funciones principales

El sistema realizará las funciones que se describen a continuación.

Gestión de Clientes

Función 1.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de clientes.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla para que el cliente introduzca su nombre de usuario y su contraseña.
Excepciones:

Función 1.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Entrada: ORDEN-USUARIO
Salida:
Usa: USUARIOS
Actualiza:
Efecto: Valida los datos introducidos por el usuario consultando la base de datos.
Excepciones: muestra un mensaje de error si los datos son incorrectos.

Función 1.3 Ver pantalla cliente: Muestra la pantalla de clientes.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla en la que los clientes pueden efectuar todas sus operaciones.
Excepciones:

Función 1.4 Ver lista autores: Muestra un listado de autores disponibles.
Entrada:
Salida: LISTA-AUTORES
Usa: DISCOS
Actualiza:
Efecto: Muestra un listado completo de todos los autores registrados en la base de datos del sistema.
Excepciones:

Función 1.5 Ver autor: Muestra un listado de los discos de un autor.
Entrada: ORDEN-AUTOR
Salida: LISTA-DISCOS-AUTOR
Usa: DISCOS
Actualiza:
Efecto: Muestra un listado completo de los discos existentes de un autor en la base de datos del sistema.
Excepciones: Muestra un mensaje de error si no hay datos.

Función 1.6 Ver disco: Muestra los datos de un disco.
Entrada: ORDEN-DISCO
Salida: LISTA-DATOS-DISCO
Usa: DISCOS
Actualiza:
Efecto: Muestra la ficha completa del disco seleccionado.
Excepciones: Muestra un mensaje de error si no hay datos.

Función 1.7 Reservar disco: Realiza la reserva de un disco.
Entrada: ORDEN-DISCO
Salida:
Usa: DISCOS
Actualiza: RESERVAS
Efecto: Realiza la reserva de un disco por el cliente anotándolo en la base de datos.
Excepciones:

Gestión de Administradores

Función 2.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña del administrador.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla para que el administrador introduzca su nombre de usuario y su contraseña.
Excepciones:

Función 2.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Identica a la función 1.2

Función 2.3 Ver pantalla admin: Muestra la pantalla de administradores.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla en la que los administradores pueden efectuar todas sus operaciones.
Excepciones:

Función 2.4 Ver lista usuarios: Muestra un listado de usuarios del sistema.
Entrada:
Salida: LISTA-USUARIOS
Usa: USUARIOS
Actualiza:
Efecto: Muestra un listado de los usuarios del sistema.
Excepciones:

Función 2.5 Dar alta usuario: Da de alta un usuario en la base e datos del sistema.
Entrada: ORDEN-USUARIO
Salida:
Usa:
Actualiza: USUARIOS
Efecto: Da de alta un usuario en la base e datos del sistema y muestra un mensa de función realizada.
Excepciones:

Función 2.6 Dar baja usuario: Da de baja un usuario en la base e datos del sistema.
Entrada: ORDEN-USUARIO
Salida:
Usa:
Actualiza: USUARIOS
Efecto: Da de baja un usuario en la base e datos del sistema y muestra un mensaje de función realizada.
Excepciones:

Función 2.7 Modificar usuario: Modifica un usuario en la base e datos del sistema.
Entrada: ORDEN-USUARIO
Salida:
Usa:
Actualiza: USUARIOS
Efecto: Modifica un usuario en la base e datos del sistema y muestra un mensaje de función realizada.
Excepciones:

Función 2.8 Ver pantalla vendedor: Muestra la pantalla de vendedores.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla de vendedores con todas sus funciones dentro de la pantalla de administradores.
Excepciones:

Función 2.9 Ver pantalla almacén: Muestra la pantalla del almacén.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla del almacén con todas sus funciones dentro de la pantalla de administradores.
Excepciones:

Gestión de Vendedores

Función 3.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña del vendedor.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla para que el vendedor introduzca su nombre de usuario y su contraseña.
Excepciones:

Función 3.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Identica a la función 1.2

Función 3.3 Ver pantalla vendedor: Muestra la pantalla de vendedores.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla en la que los vendedores pueden efectuar todas sus operaciones.
Excepciones:

Función 3.4 Ver lista autores: Muestra un listado de autores disponibles.
Identica a la función 1.4

Función 3.5 Ver autor: Muestra un listado de los discos de un autor.
Identica a la función 1.5

Función 3.6 Ver disco: Muestra los datos de un disco.
Identica a la función 1.6

Función 3.7 Vender disco: Realiza la venta de un disco.
Entrada: ORDEN-DISCO
Salida:
Usa: DISCOS, RESERVAS
Actualiza: DISCOS, RESERVAS
Efecto: Realiza la venta de un disco disminuyendo las existencias del mismo y anulando la reserva del mismo en caso de que la hubiera.
Excepciones:

Gestión de Almacén

Función 4.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña del almacén.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla para que el almacenista introduzca su nombre de usuario y su contraseña.
Excepciones:

Función 4.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Identica a la función 1.2

Función 4.3 Ver pantalla almacén: Muestra la pantalla del almacén.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla en la que los almacenistas pueden efectuar todas sus operaciones.
Excepciones:

Función 4.4 Dar alta disco: Da de alta un disco en la base de datos.
Entrada: DATOS-DISCO
Salida:
Usa:
Actualiza: DISCOS
Efecto: Da de alta un disco en la base de datos mostrando un mensaje de que se ha hecho.
Excepciones:

Función 4.5 Dar baja disco: Da de baja un disco en la base de datos.
Entrada: DATOS-DISCO
Salida:
Usa: DISCOS
Actualiza: DISCOS
Efecto: Da de baja un disco en la base de datos mostrando un mensaje de que se ha hecho.
Excepciones:

Función 4.6 Ver disco: Muestra los datos de un disco.
Identica a la función 1.6

9.12.2. Requisitos de capacidad

El servidor de bases de datos deberá tener la capacidad suficiente para contener todos los datos de la aplicación, y para responder sin demasiada demora a las peticiones que se le hagan.

9.12.3. Requisitos de operación

La selección de las operaciones se hará mediante un sistema de botones.

Los botones que realizan las funciones propias de cada usuario estarán siempre visibles en la página correspondiente a dicho usuario.

9.12.4. Requisitos de pruebas de aceptación

Se deben probar todas las funciones, tanto con entradas de datos normales como con datos que provoquen errores. Esto implica que deben mostrarse mensajes de error para los casos de datos erróneos o de funcionamiento anormal del sistema.

9.12.5. Requisitos de documentación

Se creará un sencillo manual de usuario que describa el uso del sistema.

Se creará un manual de instalación que describa cómo instalar y configurar los diferentes programas y soportes necesarios para el funcionamiento de la aplicación.

Capítulo 10

Desarrollo del diseño de la aplicación

En este capítulo se desarrolla el diseño de la aplicación objeto del proyecto. Se realizan los diseños de datos, arquitectónico, de interfaz y funcional. Este capítulo es el Documento para la Especificación del Diseño del proyecto.

10.1. Objetivo

El objetivo del sistema es facilitar la gestión, via *Web*, del catálogo de una pequeña tienda de discos. Tendrán acceso al sistema administradores, almacenistas, vendedores y clientes. Estos usuarios realizarán funciones de alta y baja de usuarios, listado de autores y discos, reservas de discos, ventas, altas y bajas de discos.

10.2. Ámbito

El sistema se denominará Catálogo *Web*, y consistirá en una serie de páginas *Web* que incluirán todas las funciones necesarias.

La navegación entre páginas tendrá en cuenta el tipo de usuario para permitir el acceso y poder realizar las funciones permitidas a ese usuario.

Aunque en el sistema se podrán realizar ventas y reservas, estas afectarán únicamente al número de existencias de los discos en la Base de Datos, no se realizará ningún tipo de gestión económica.

10.3. Descripción funcional

Para operar el sistema se dispondrá de un terminal con pantalla, ratón y teclado. El sistema operará mediante una serie de páginas *Web* que contendrán la funcionalidad adecuada a cada tipo de usuario. La navegación entre páginas se realizará por una serie de botones. Las funciones se activarán también mediante botones. Las pantallas tendrán las casillas necesarias para introducir y presentar los datos, así como listas para mostrar los listados requeridos.

Las funciones a realizar por el sistema se pueden organizar en varios grupos funcionales, que se describen a continuación.

Gestión Clientes:
Función 1.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de clientes.
Función 1.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Función 1.3 Ver pantalla cliente: Muestra la pantalla de clientes.
Función 1.4 Ver lista autores: Muestra un listado de autores disponibles.
Función 1.5 Ver autor: Muestra un listado de los discos de un autor.
Función 1.6 Ver disco: Muestra los datos de un disco.
Función 1.7 Reservar disco: Realiza la reserva de un disco.

Gestión Administradores:
Función 2.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de administradores.
Función 2.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Función 2.3 Ver pantalla admin: Muestra la pantalla de administradores.
Función 2.4 Ver lista usuarios: Muestra un listado de los usuarios del sistema.
Función 2.5 Dar alta usuario: Da de alta un usuario en la base e datos del sistema.
Función 2.6 Dar baja usuario: Da de baja un usuario en la base e datos del sistema.
Función 2.7 Modificar usuario: Modifica un usuario en la base e datos del sistema.
Función 2.8 Ver pantalla vendedor: Muestra la pantalla de vendedores.
Función 2.9 Ver pantalla almacén: Muestra la pantalla del almacén.

Gestión Vendedor:
Función 3.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de vendedores.
Función 3.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Función 3.3 Ver pantalla vendedor: Muestra la pantalla de vendedores.
Función 3.4 Ver lista autores: Muestra un listado de autores disponibles.
Función 3.5 Ver autor: Muestra un listado de los discos de un autor.
Función 3.6 Ver disco: Muestra los datos de un disco.
Función 3.7 Vender disco: Realiza la venta de un disco.

Gestión Almacén:
Función 4.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de almacenistas.
Función 4.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Función 4.3 Ver pantalla almacén: Muestra la pantalla de almacén.
Función 4.4 Dar alta disco: Da de alta un disco en la base de datos.
Función 4.5 Dar baja disco: Da de baja un disco en la base de datos.
Función 4.6 Ver disco: Muestra los datos de un disco.

10.4. Diseño de datos

Para diseñar los datos del sistema se parte del diagrama entidad-relación y se derivan de él las tablas de la base de datos que se usarán en la aplicación.

Como podemos ver en el DER (figura 10.1) Hay dos entidades, Discos y Usuarios y una relación, Reserva. Las dos entidades se convierten automáticamente en tablas. La relación tiene cardinalidad N:N, es decir varios a varios, un usuario puede realizar varias reservas de un disco y un disco puede ser reservado por varios usuarios, en estas condiciones la relación Reserva tiene que traducirse en una tabla independiente en la base de datos.

La tabla Discos tendrá como clave primaria el campo IDdisco, además como las búsquedas se harán por nombre de autor o por nombre de disco, haremos que esos dos campos sean indexados para un acceso mas rápido.

La tabla Usuarios tendrá como clave primaria el campo IDusuario y como campo indexado "usuario".

La tabla Reserva al provenir de una relación que conecta las entidades Discos y Usuarios tendrá como clave primaria a las dos claves primarias de esas entidades, es decir su clave primaria será IDdisco + IDusuario.

A continuación se describen estas tablas incluyendo los campos, su tipo, longitud, descripción, obligatoriedad e indexación.

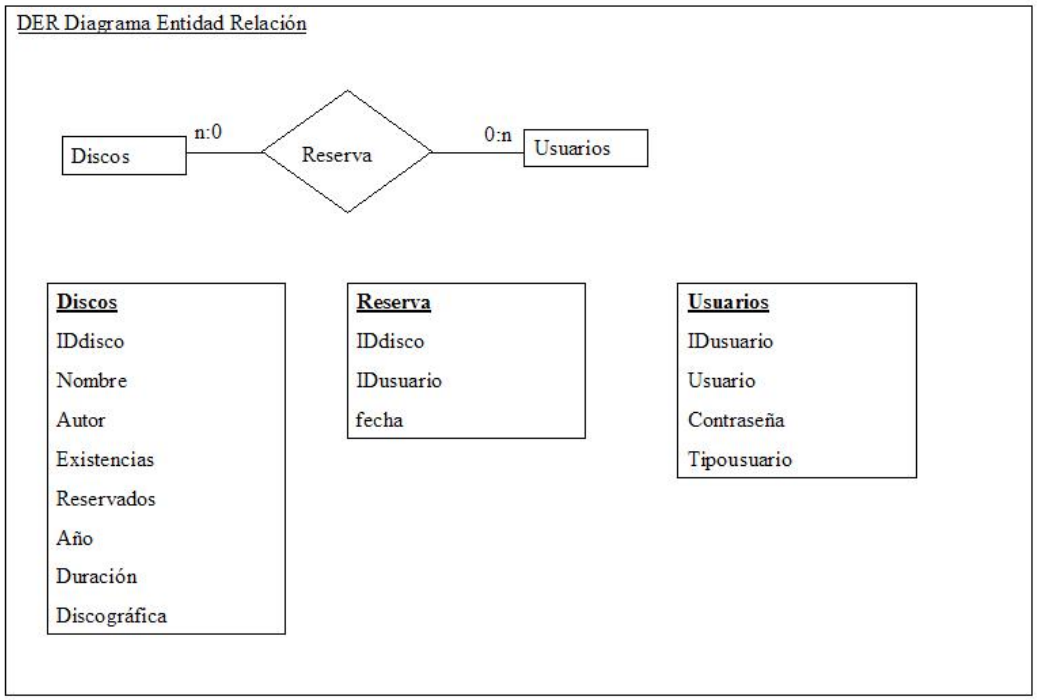

Figura 10.1: Diagrama Entidad-Relación del Catalogo *Web*

Tabla: DISCOS					
Campo	Tipo	Long.	Oblig.	Índice	Descripción
IDdisco	Núm.	3	SI	SI	Número de referencia del disco
Nombre	Texto	60	SI	SI	Nombre del disco
Autor	Texto	20	SI	SI	Nombre del autor
Existencias	Num.	2	SI	NO	Cantidad de discos en la BD
Reservados	Num	2	SI	NO	Cantidad de discos reservados
Año	fecha	8	NO	NO	Fecha de publicación
Duración	Num	2	NO	NO	Minutos de duración del disco
Discográfica	Texto	20	NO	NO	Casa discográfica editora

Tabla: USUARIOS					
Campo	Tipo	Long.	Oblig.	Índice	Descripción
IDusuario	Núm.	2	SI	SI	Número de referencia del usuario
Usuario	Texto	20	SI	SI	Nombre del usuario
Contraseña	Texto	10	SI	NO	Contraseña del usuario
Tipousuario	Num.	1	SI	NO	Número que identifica al tipo de usuario

Tabla: RESERVAS					
Campo	Tipo	Long.	Oblig.	Índice	Descripción
IDdisco	Núm.	3	SI	SI	Número de referencia del disco
IDusuario	Núm.	3	SI	SI	Número de referencia del usuario
Fecha	fecha	8	NO	NO	Fecha de la reserva

10.5. Diseño arquitectónico

Se realiza el diseño arquitectónico empleando la técnica del diseño estructurado para convertir los diagramas de flujo de datos obtenidos en el análisis en diagramas de estructura.

El primer paso será analizar los tipos de flujos de datos que circulan por el sistema. Para ello se puede reformular el DFD.1 (Catálogo *Web*) prescindiendo de los almacenes de información. En el diagrama se aprecia claramente que el flujo de datos corresponde al tipo transacción. En la estructura el centro de transacción no se corresponde con un proceso o función concretos, sino simplemente con la ramificación del flujo de órdenes de entrada. De aquí se puede derivar un primer nivel de la estructura del sistema tal como se muestra en la figura 10.2.

El proceso se repite ahora para cada subsistema (gestión admin, gestión cliente, gestión almacén y gestión vendedor) para refinar aún más la estructura del sistema. Para ello se reformulan los DFD correspondientes eliminando los almacenes de información.

En el DFD correspondiente a la gestión del cliente (ver figura 10.3) se aprecia que el tipo de flujo de datos es de transacción. En una primera fase de preparación el flujo de datos fluye hasta que se muestra la pantalla del cliente, en este punto se estaría ante el centro de transacción que como en el caso anterior no se corresponde con una función concreta sino con una agrupación de funciones o secuencia de funciones que se han asociado al

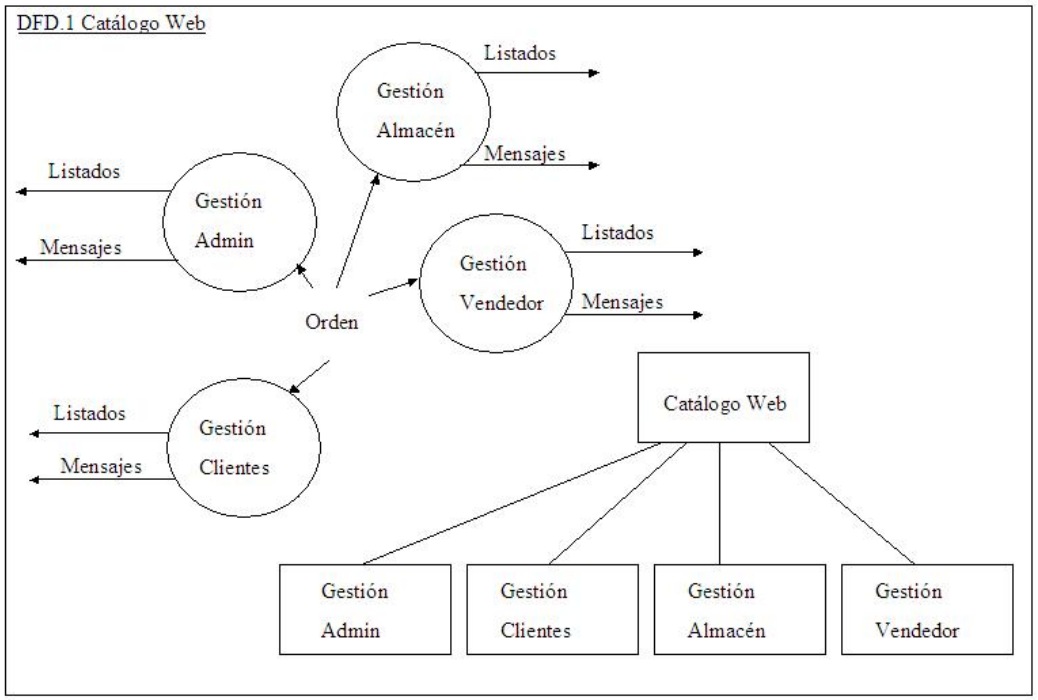

Figura 10.2: Catálogo *Web*. Diseño inicial

diagrama por la afinidad de los datos sobre los que operan. Es decir tenemos una serie de funciones a las que se puede acceder mediante un grupo de órdenes.

El diagrama de estructura que se deriva tiene un módulo principal que es "gestión cliente" del que depende un módulo que se llama "mostrar pantalla" con las funciones previas al centro de transacción ("ver pantalla contraseña", "validar contraseña" y "ver pantalla cliente"), la función "ver lista autores", la función "ver autor", la función "ver disco" y la función "reservar disco". El diagrama correspondiente puede verse en la figura 10.3.

Procediendo de la misma forma obtenemos el diseño del subsistema "gestión Admin" (ver figura 10.4).

Procediendo de la misma forma obtenemos el diseño del subsistema "gestión vendedor" (ver figura 10.5).

Procediendo de la misma forma obtenemos el diseño del subsistema "gestión almacén" (ver figura 10.6).

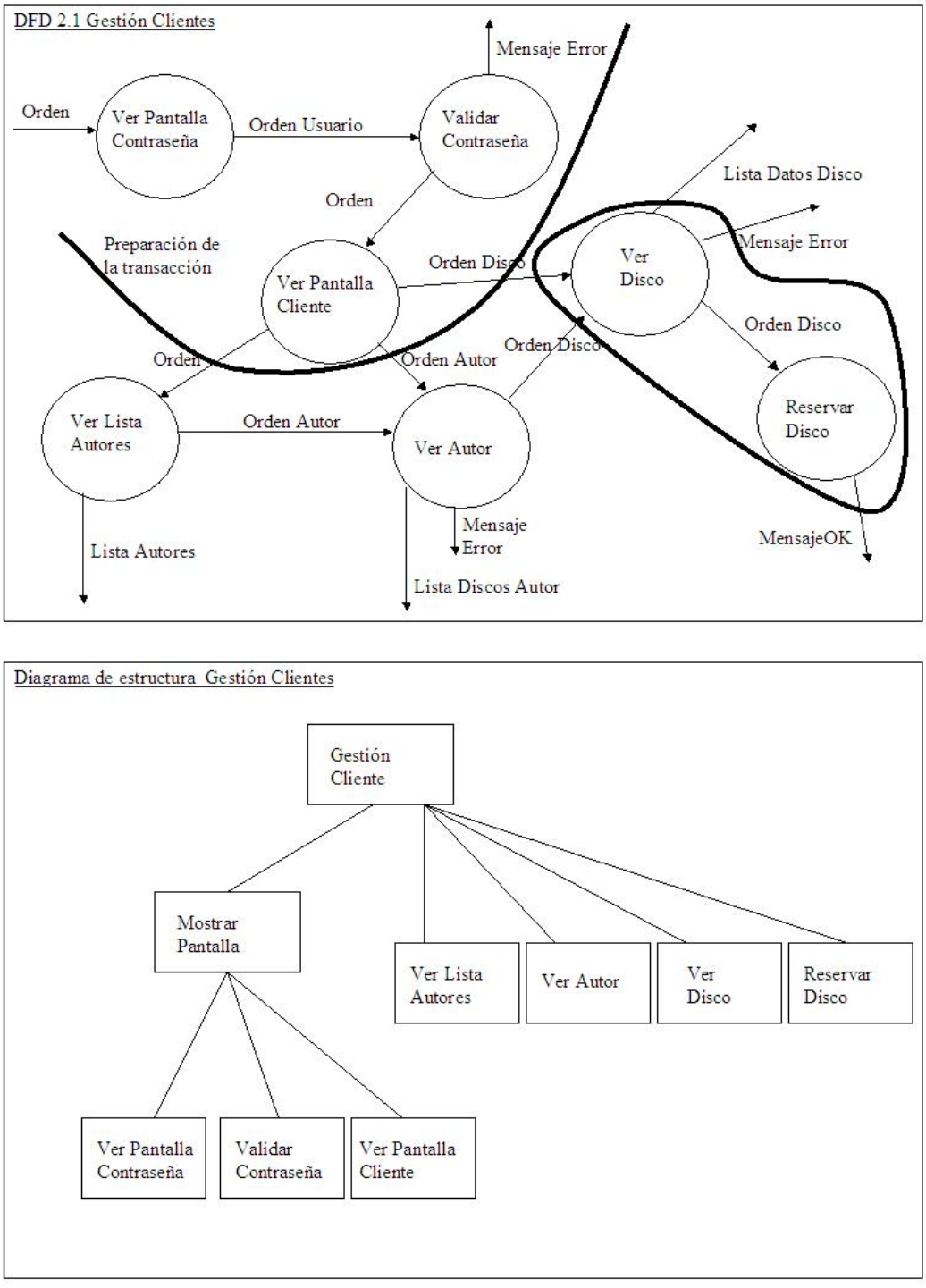

Figura 10.3: Diseño de gestión cliente

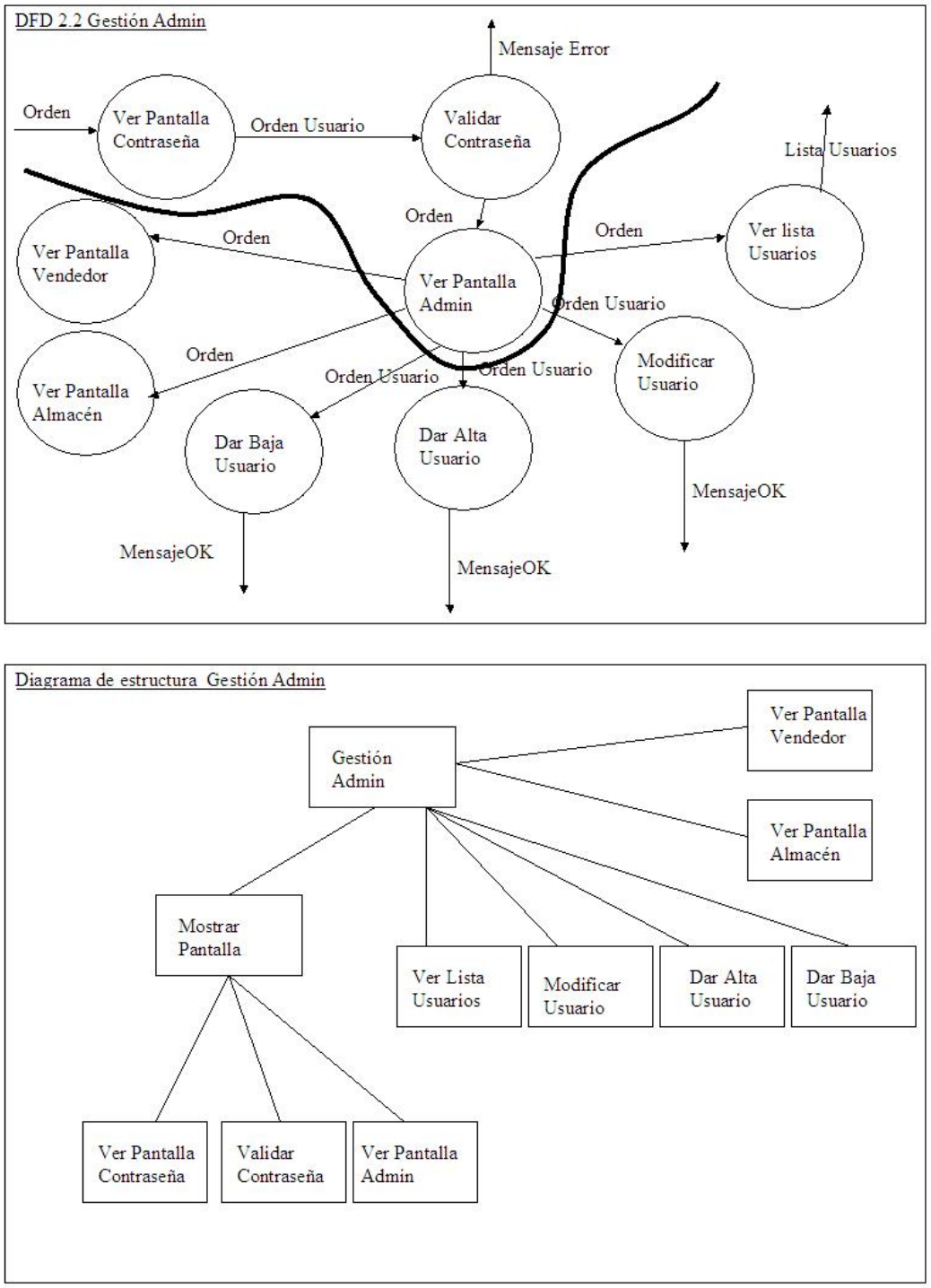

Figura 10.4: Diseño de gestión administrador

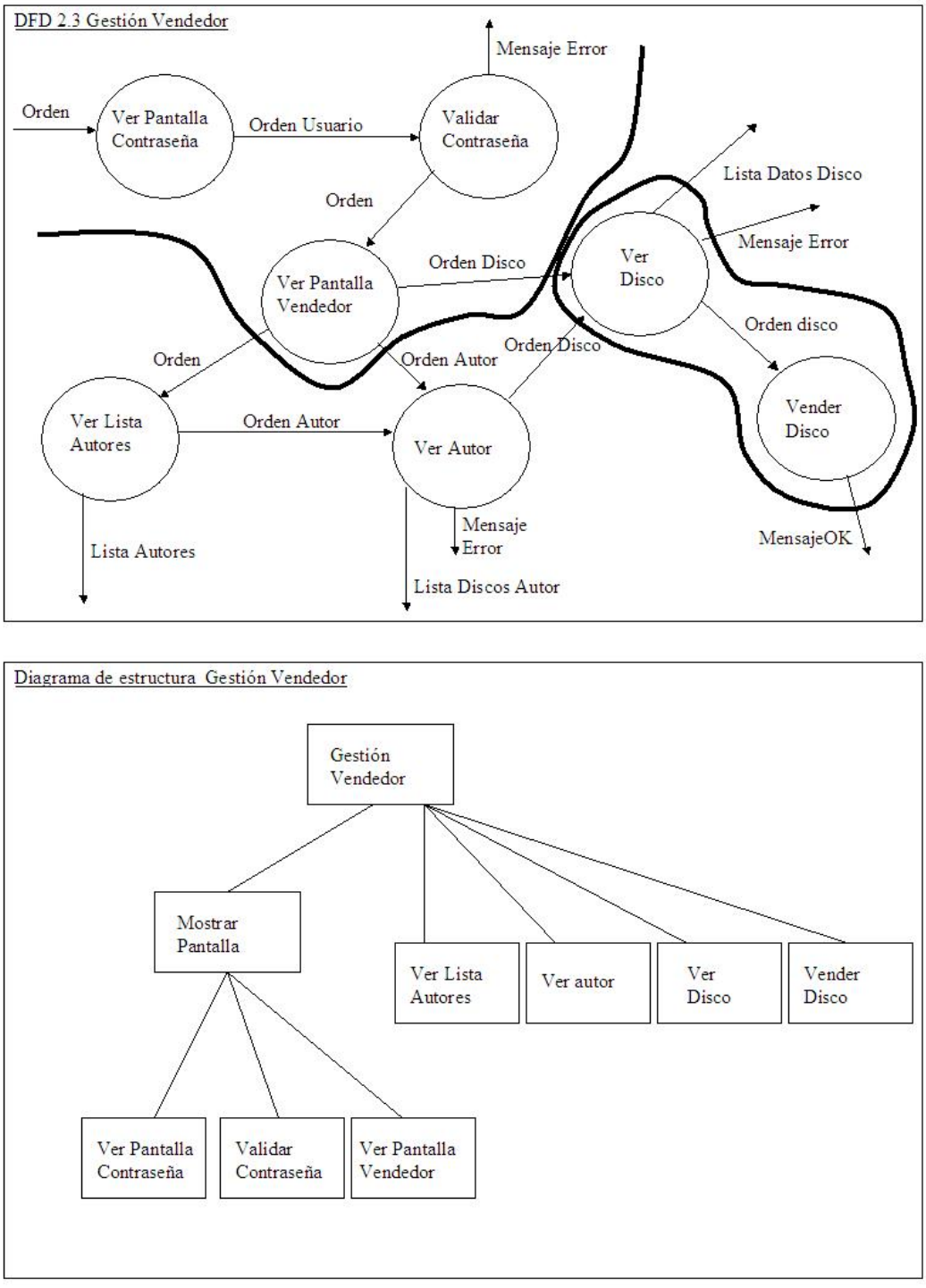

Figura 10.5: Diseño de gestión vendedor

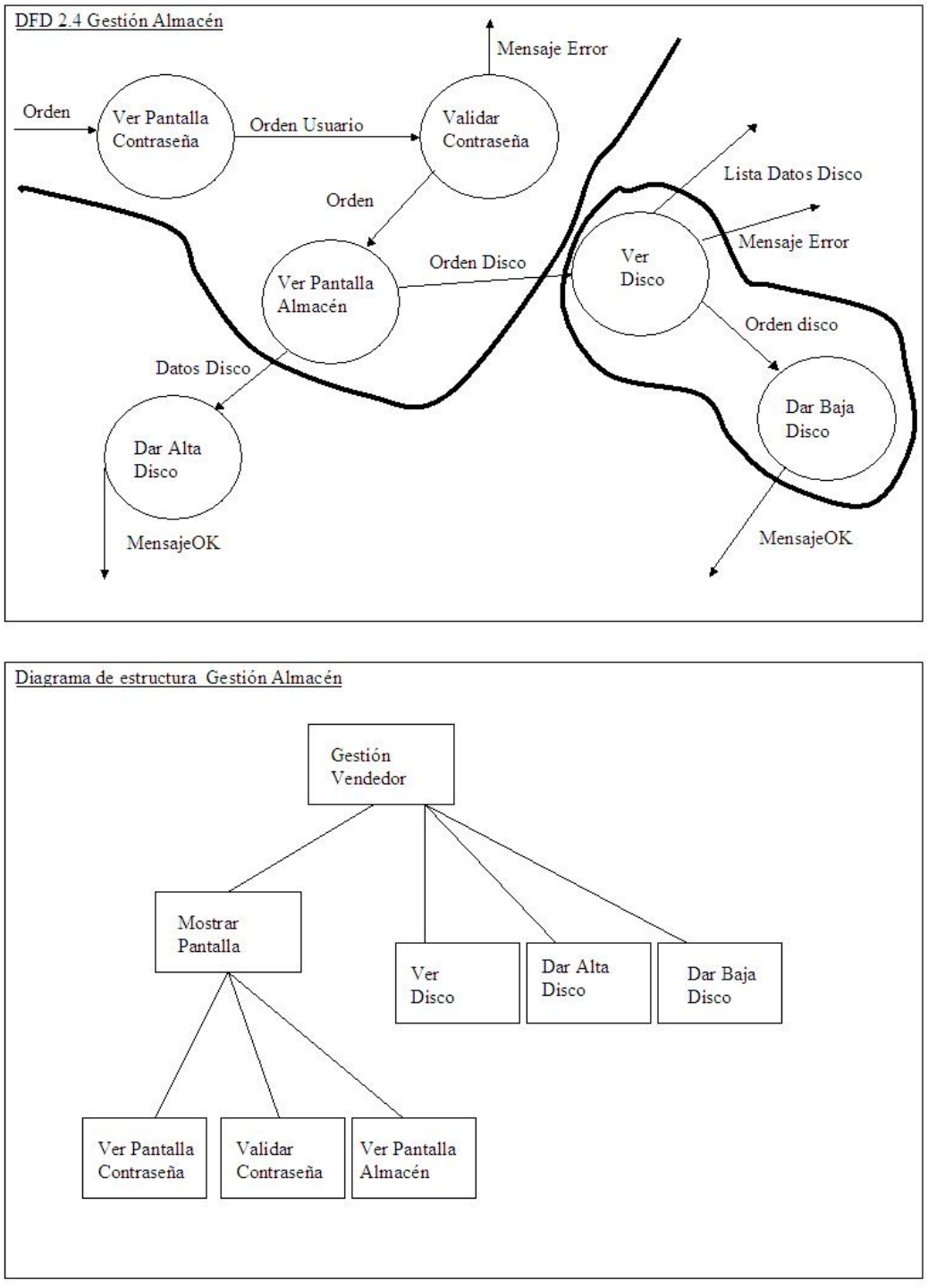

Figura 10.6: Diseño de gestión almacén

Una vez se ha obtenido el diagrama de estructura de cada subsistema se puede obtener el diagrama del sistema completo uniendo todos ellos como puede verse en la figura 10.7.

10.6. Diseño de la interfaz

Para realizar el diseño de la interfaz del sistema se tienen en cuenta las posibles conexiones que tendrá ésta, tanto las internas entre sus módulos o funciones, como las externas con productores o consumidores de información tanto humanos como no humanos.

10.6.1. Interfaz interna

La interfaz interna hace referencia al flujo de información que fluye entre los módulos del sistema. En los diagramas de flujo de datos obtenidos en la fase de análisis estaba representado este flujo como objetos de datos que entraban o salían de los procesos de dichos diagramas.

En nuestro caso la interfaz interna consistirá en especificar para cada función del sistema los datos que acepta como entrada, los datos que produce como salida, las tablas de la base de datos que utiliza o actualiza y las posibles excepciones o mensajes de cualquier tipo que pueden producirse durante la ejecución de la función.

Todos estos datos que constituyen la interfaz interna del sistema se recogen en la especificación detallada de las funciones que se realiza en la sección 10.7, página 139.

10.6.2. Interfaz externa

La interfaz externa hace referencia a las conexiones del sistema con productores o consumidores de información externos al programa y que no son humanos.

En nuestro caso, al fijarse en el diagrama de contexto obtenido en el análisis (figura 9.2), se verá que la única entidad que requiere una interfaz externa es la pantalla del ordenador.

Todos los datos que entran o salen del sistema lo harán a través de la interfaz mostrada en la pantalla. Para poder introducir y visualizar datos sera necesario habilitar una casilla de texto para cada campo de las tablas de la base de datos, cada casilla irá acompañada de una etiqueta identificando dicho campo. Además será necesario habilitar una lista para mostrar los resultados de consultas consistentes en listados.

Las órdenes se le darán al sistema mediante pulsación de botones.

10.6.3. Interfaz de usuario

A los usuarios del sistema se les supone el conocimiento informático mínimo para saber rellenar campos con los datos adecuados y para moverse entre páginas *Web*.

10.6.4. Prototipos de las páginas *Web*

Con los requisitos de las interfaces interna, externa y de usuario comentados previamente se puede bosquejar el aspecto que tendrían las páginas *Web* del sistema.

Se muestran a continuación unos prototipos en los que sólo se pretende mostrar los elementos que deben contener, no el aspecto final que tendrán.

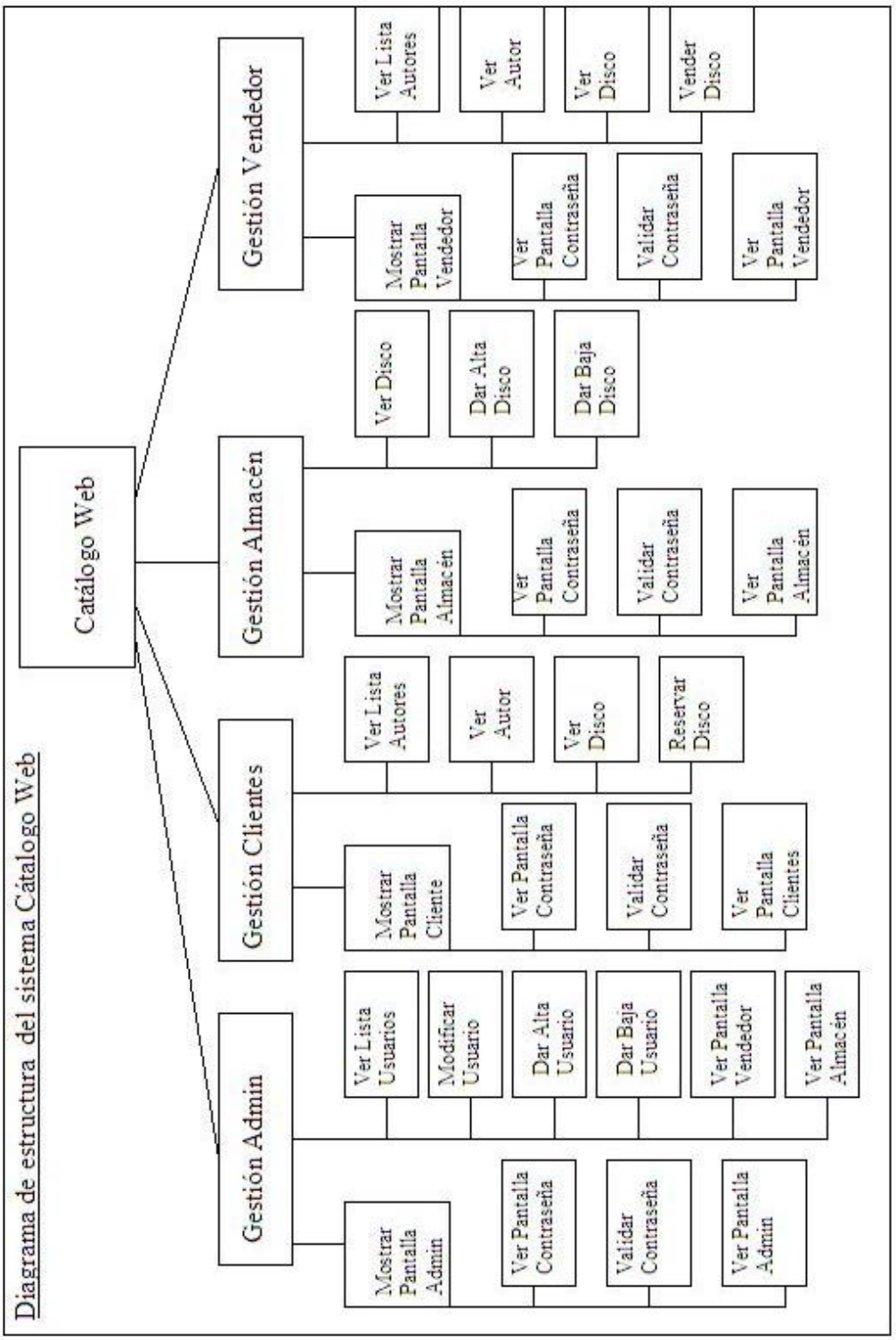

Figura 10.7: Diagrama de estructura del sistema Catálogo *Web*

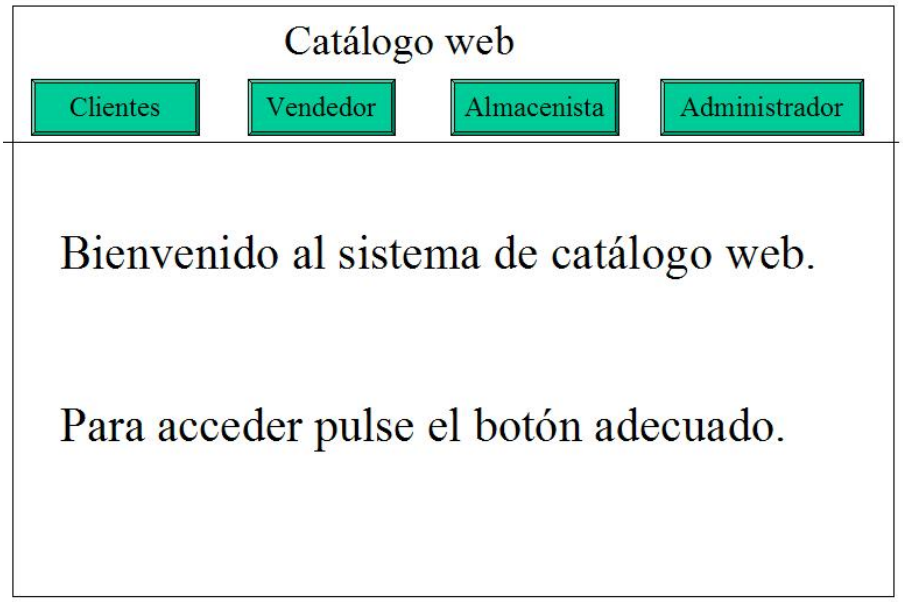

Figura 10.8: Página *Web* inicial

Figura 10.9: Página *Web* de acceso al sistema

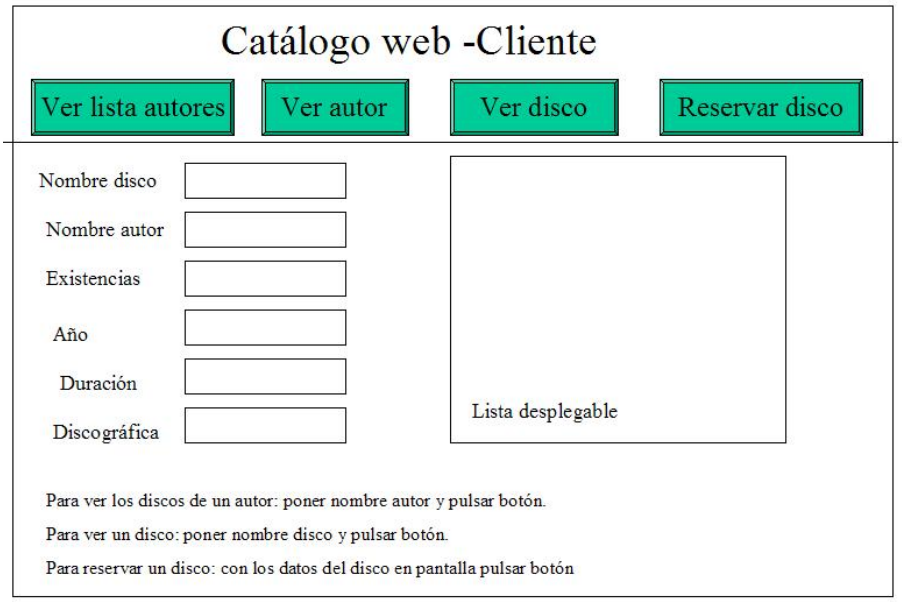

Figura 10.10: Página *Web* del cliente

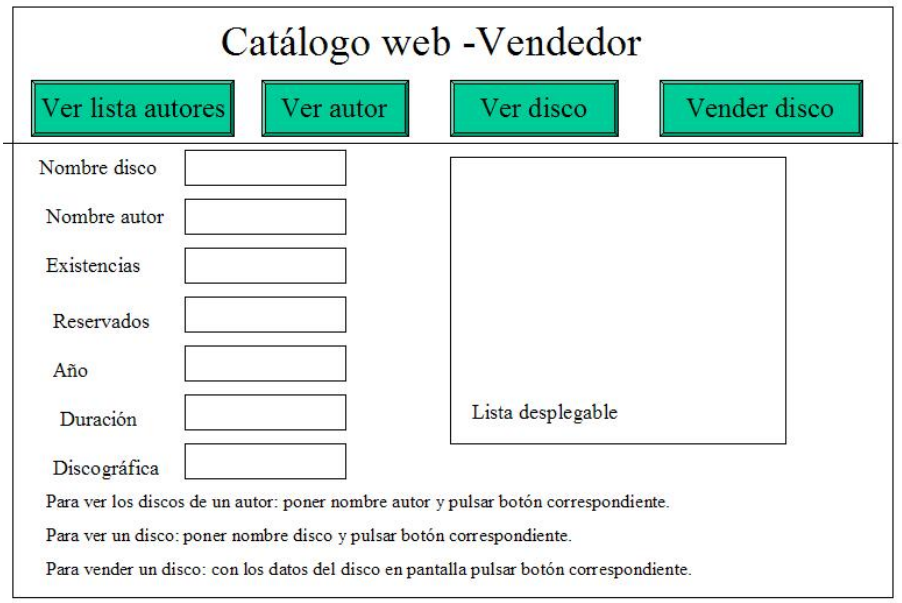

Figura 10.11: Página *Web* del vendedor

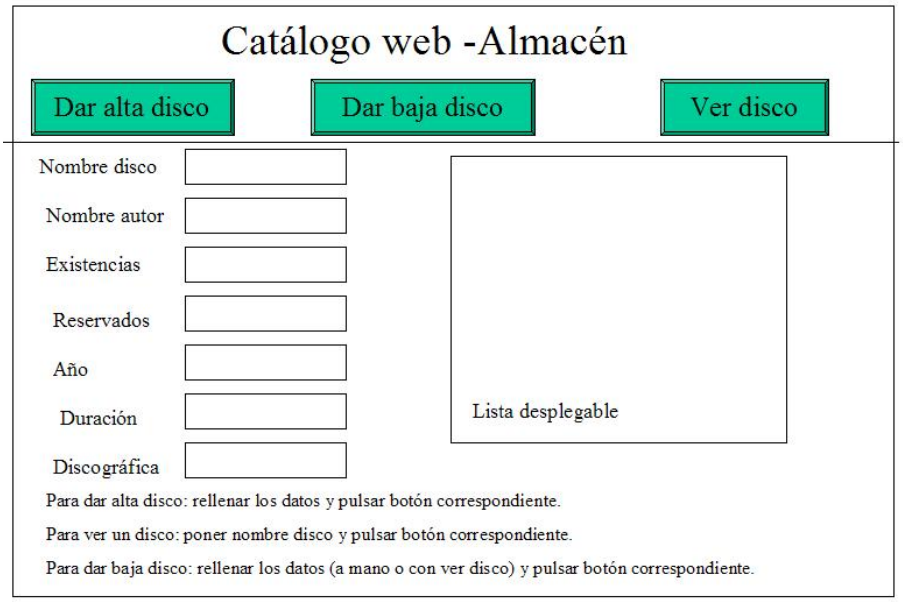

Figura 10.12: Página *Web* del almacén

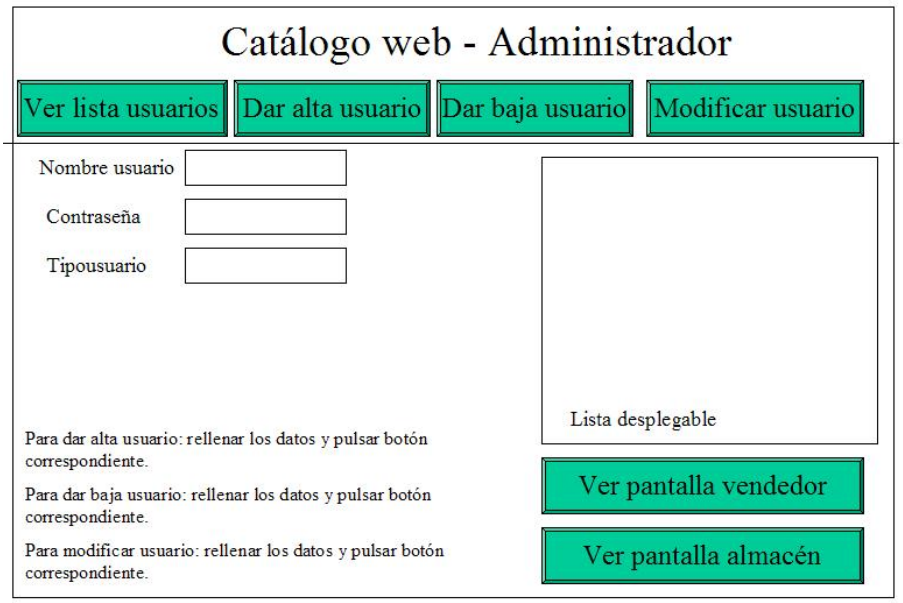

Figura 10.13: Página *Web* del administrador

10.7. Diseño procedimental

En esta sección se especifican en seudocódigo los detalles algorítmicos de los módulos y funciones de la aplicación. Se utiliza la notación de seudocódigo definida en la sección 8.3.3 (página 92).

10.7.1. Módulo: Catálogo *Web*

En este módulo se encuentra la página inicial de la aplicación. Se presentan al usuario unos botones para que elija como qué tipo de usuario quiere acceder al sistema.

Proceso del módulo Catálogo *Web*:

-iniciar la sesión
-seleccionar tipo de usuario pulsando el botón correspondiente
SELECCIÓN tipousuario
  CASO Cliente
  -iniciar el módulo Gestión Cliente
  FIN-CASO
  CASO Administrador
  -iniciar el módulo Gestión Admin
  FIN-CASO
  CASO Vendedor
  -iniciar el módulo Gestión Vendedor
  FIN-CASO
  CASO Almacenista
  -iniciar el módulo Gestión Almacén
  FIN-CASO
FIN-SELECCIÓN

10.7.2. Módulo: Gestión Cliente

Este módulo se encarga de mostrar la pantalla del cliente, si los datos de acceso son válidos, y realiza las funciones propias del cliente.

Proceso del módulo Gestión Cliente:

-LLAMA ver pantalla contraseña
-datos <- datos introducidos por el usuario
-LLAMA validar contraseña (datos)
-LLAMA ver pantalla cliente
REPETIR
  SI consulta requiere datos
    -introduce datos consulta
  FIN-SI
  SELECCIÓN operación
    CASO ver lista autores
    -LLAMA ver lista autores
    FIN-CASO
    CASO ver autor
    -LLAMA ver autor
    FIN-CASO
    CASO ver disco
    -LLAMA ver disco
    FIN-CASO
    CASO reservar disco
    -LLAMA reservar disco
    FIN-CASO
  FIN-SELECCIÓN
HASTA salir de la página

Las funciones realizadas por este módulo son las siguientes:

Función 1.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña de clientes.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla para que el cliente introduzca su nombre de usuario y su contraseña.
Excepciones:
Proceso:
  -muestra la pantalla de contraseña.

Función 1.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Entrada: ORDEN-USUARIO
Salida:
Usa: USUARIOS
Actualiza:
Efecto: Valida los datos introducidos por el usuario consultando la base de datos.
Excepciones: muestra un mensaje de error si los datos son incorrectos.
proceso:
-asigna tipousuario según botón pulsado
-lee el nombre de usuario
-lee la contraseña
MIENTRAS no existe ese usuario, en tipousuario, en la BD
-mostrar mensaje de error (usuario desconocido)
-leer nombre usuario
FIN-MIENTRAS
-obtiene la contraseña del usuario
MIENTRAS contraseñas no coinciden
-mostrar mensaje de error (contraseña incorrecta)
-leer contraseña
FIN-MIENTRAS
-mensaje (acceso autorizado)

Función 1.3 Ver pantalla cliente: Muestra la pantalla de clientes.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla en la que los clientes pueden efectuar todas sus operaciones.
Excepciones:
Proceso:
-muestra la pantalla de cliente.

Función 1.4 Ver lista autores: Muestra un listado de autores disponibles.
Entrada:
Salida: LISTA-AUTORES
Usa: DISCOS
Actualiza:
Efecto: Muestra un listado completo de todos los autores registrados en la base de datos del sistema.
Excepciones:
Proceso:
-accede a la BD y obtiene un listado de autores
-presenta en pantalla el listado de autores

Función 1.5 Ver autor: Muestra un listado de los discos de un autor.
Entrada: ORDEN-AUTOR
Salida: LISTA-DISCOS-AUTOR
Usa: DISCOS
Actualiza:
Efecto: Muestra un listado completo de los discos existentes de un autor en la base de datos del sistema.
Excepciones: Muestra un mensaje de error si no hay datos.
Proceso:
-leer nombre autor de la casilla correspondiente
-SI nombre está en BD
-obtener la lista de discos de ese autor
-presentar la lista en pantalla
SINO
-mensaje (No hay discos del autor disponibles)
FIN-SI

Función 1.6 Ver disco: Muestra los datos de un disco.
Entrada: ORDEN-DISCO
Salida: LISTA-DATOS-DISCO
Usa: DISCOS
Actualiza:
Efecto: Muestra la ficha completa del disco seleccionado.
Excepciones: Muestra un mensaje de error si no hay datos.
Proceso:
-leer nombre disco de la casilla correspondiente
-SI nombre está en BD
-obtener todos los datos del disco
-presentar los datos rellenando lass casillas correspondientes
SINO
-mensaje (Disco no existe en el catálogo)
FIN-SI

Función 1.7 Reservar disco: Realiza la reserva de un disco.
Entrada: ORDEN-DISCO
Salida:
Usa: DISCOS
Actualiza: RESERVAS
Efecto: Realiza la reserva de un disco por el cliente anotándolo en la base de datos.
Excepciones:
proceso:
-leer nombre disco de la casilla correspondiente
-SI (nombre está en BD) Y (existencias >reservados)
-incrementar reservados en la BD
-mensaje (El disco ha sido reservado)
SINO
-mensaje (Imposible reservar este disco)
FIN-SI

10.7.3. Módulo: Gestión Admin

Este módulo se encarga de mostrar la pantalla del administrador, si los datos de acceso son válidos, y realiza las funciones propias del administrador.

Proceso del módulo Gestión Admin:

-LLAMA ver pantalla contraseña
-datos <- datos introducidos por el usuario
-LLAMA validar contraseña (datos)
-LLAMA ver pantalla administrador
REPETIR
  SI consulta requiere datos
    -introduce datos consulta
  FIN-SI
  SELECCIÓN operación
    CASO ver lista usuarios
    -LLAMA ver lista usuarios
    FIN-CASO
    CASO modificar usuario
    -LLAMA modificar usuario
    FIN-CASO
    CASO dar alta usuario
    -LLAMA dar alta usuario
    FIN-CASO
    CASO dar baja usuario
    -LLAMA dar baja usuario
    FIN-CASO
    CASO ver pantalla almacén
    -LLAMA ver pantalla almacén
    -oculta botón ver pantalla almacén
    -muestra botón ver pantalla de la pantalla de origen de la orden
    FIN-CASO
    CASO ver pantalla vendedor
    -LLAMA ver pantalla vendedor
    -oculta botón ver pantalla vendedor
    -muestra botón ver pantalla de la pantalla de origen de la orden
    FIN-CASO
    CASO ver pantalla admin
    -LLAMA ver pantalla admin
    -oculta botón ver pantalla admin
    -muestra botón ver pantalla de la pantalla de origen de la orden
    FIN-CASO
  FIN-SELECCIÓN
HASTA salir de la página

Las funciones realizadas por este módulo son las siguientes:

Función 2.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña del administrador.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla para que el administrador introduzca su nombre de usuario y su contraseña.
Excepciones:
Proceso:
-muestra pantalla contraseña

Función 2.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Identica a la función 1.2

Función 2.3 Ver pantalla admin: Muestra la pantalla de administradores.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla en la que los administradores pueden efectuar todas sus operaciones.
Excepciones:
Proceso:
-muestra la pantalla del administrador.

Función 2.4 Ver lista usuarios: Muestra un listado de usuarios del sistema.
Entrada:
Salida: LISTA-USUARIOS
Usa: USUARIOS
Actualiza:
Efecto: Muestra un listado de los usuarios del sistema.
Excepciones:
Proceso:
-accede a la BD y obtiene un listado de usuarios
-presenta en pantalla el listado de usuarios

Función 2.5 Dar alta usuario: Da de alta un usuario en la base e datos del sistema.
Entrada: ORDEN-USUARIO
Salida:
Usa:
Actualiza: USUARIOS
Efecto: Da de alta un usuario en la base e datos del sistema y muestra un mensaje de función realizada.
Excepciones:
Proceso:
-leer nombre usuario y tipo usuario de las casillas correspondientes
-SI nombre y tipo usuario están en BD
-mensaje (usuario ya existe)
SINO
-leer nombre, tipousuario y contraseña de las casillas correspondientes
-generar nueva entrada en la BD con esos datos
-mensaje (Usuario dado de alta correctamente)
FIN-SI

Función 2.6 Dar baja usuario: Da de baja un usuario en la base e datos del sistema.
Entrada: ORDEN-USUARIO
Salida:
Usa:
Actualiza: USUARIOS
Efecto: Da de baja un usuario en la base e datos del sistema y muestra un mensaje de función realizada.
Excepciones:
Proceso:
-leer nombre usuario y tipo usuario de las casillas correspondientes
-SI nombre y tipo usuario NO están en BD
-mensaje (usuario no existe)
SINO
-leer nombre, tipousuario de las casillas correspondientes
-eliminar la entrada en la BD con esos datos
-mensaje (Usuario dado de baja correctamente)
FIN-SI

Función 2.7 Modificar usuario: Modifica un usuario en la base de datos del sistema.
Entrada: ORDEN-USUARIO
Salida:
Usa:
Actualiza: USUARIOS
Efecto: Modifica un usuario en la base e datos del sistema y muestra un mensaje de función realizada.
Excepciones:
Proceso:
-leer nombre usuario y tipo usuario de las casillas correspondientes
-SI nombre y tipo usuario NO están en BD
-mensaje (usuario no existe)
SINO
-leer nombre, tipousuario y contraseña de las casillas correspondientes
-actualizar la entrada en la BD con esos datos
-mensaje (Contraseña modificada correctamente)
FIN-SI

Función 2.8 Ver pantalla vendedor: Muestra la pantalla de vendedores.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla de vendedores con todas sus funciones dentro de la pantalla de administradores.
Excepciones:
Proceso:
-muestra la pantalla del vendedor

Función 2.9 Ver pantalla almacén: Muestra la pantalla del almacén.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla del almacén con todas sus funciones dentro de la pantalla de administradores.
Excepciones:
Proceso:
-muestra la pantalla del almacén

10.7.4. Módulo: Gestión Vendedores

Este módulo se encarga de mostrar la pantalla del vendedor, si los datos de acceso son válidos, y realiza las funciones propias del vendedor.

Proceso del módulo Gestión Vendedor:

```
-LLAMA ver pantalla contraseña
-datos <- datos introducidos por el usuario
-LLAMA validar contraseña (datos)
-LLAMA ver pantalla vendedor
REPETIR
  SI consulta requiere datos
    -introduce datos consulta
  FIN-SI
  SELECCIÓN operación
    CASO ver lista autores
    -LLAMA ver lista autores
    FIN-CASO
    CASO ver autor
    -LLAMA ver autor
    FIN-CASO
    CASO ver disco
    -LLAMA ver disco
    FIN-CASO
    CASO vender disco
    -LLAMA vender disco
    FIN-CASO
  FIN-SELECCIÓN
HASTA salir de la página
```

Las funciones realizadas por este módulo son las siguientes:

Función 3.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña del vendedor.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla para que el vendedor introduzca su nombre de usuario y su contraseña.
Excepciones:
Proceso:
-muestra pantalla contraseña

Función 3.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Identica a la función 1.2

Función 3.3 Ver pantalla vendedor: Muestra la pantalla de vendedores.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla en la que los vendedores pueden efectuar todas sus operaciones.
Excepciones:
Proceso:
-muestra pantalla contraseña

Función 3.4 Ver lista autores: Muestra un listado de autores disponibles.
Idéntica a la función 1.4

Función 3.5 Ver autor: Muestra un listado de los discos de un autor.
Idéntica a la función 1.5

Función 3.6 Ver disco: Muestra los datos de un disco.
Idéntica a la función 1.6

Función 3.7 Vender disco: Realiza la venta de un disco.
Entrada: ORDEN-DISCO
Salida:
Usa: DISCOS, RESERVAS
Actualiza: DISCOS, RESERVAS
Efecto: Realiza la venta de un disco disminuyendo las existencias del mismo y anulando la reserva del mismo en caso de que la hubiera.
Excepciones:
Proceso:

```
-leer nombre disco de la casilla correspondiente
-leer nombre usuario de la casilla correspondiente
-SI (reserva disco por usuario está en BD) O (existencias >reservados)
  -decrementar existencias
  SI reserva disco por usuario está en BD
    -eliminar la reserva de la BD
  FIN-SI
  -incrementar reservados en la BD
  -mensaje (El disco ha sido vendido)
SINO
  -mensaje (Imposible vender este disco)
FIN-SI
```

10.7.5. Módulo: Gestión Almacén

Este módulo se encarga de mostrar la pantalla del almacén, si los datos de acceso son válidos, y realiza las funciones propias del almacén.

Proceso del módulo Gestión Almacén:

-LLAMA ver pantalla contraseña
-datos <- datos introducidos por el usuario
-LLAMA validar contraseña (datos)
-LLAMA ver pantalla almacén
REPETIR
  SI consulta requiere datos
    -introduce datos consulta
  FIN-SI
  SELECCIÓN operación
    CASO ver disco
    -LLAMA ver disco
    FIN-CASO
    CASO dar alta disco
    -LLAMA dar alta disco
    FIN-CASO
    CASO dar baja disco
    -LLAMA dar baja disco
    FIN-CASO
  FIN-SELECCIÓN
HASTA salir de la página

Las funciones realizadas por este módulo son las siguientes:

Función 4.1 Ver pantalla contraseña: Muestra la pantalla de la contraseña del almacén.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla para que el almacenista introduzca su nombre de usuario y su contraseña.
Excepciones:
Proceso:
  -ver pantalla contraseña

Función 4.2 Validar contraseña: Valida el usuario y contraseña introducidos.
Identica a la función 1.2

Función 4.3 Ver pantalla almacén: Muestra la pantalla del almacén.
Entrada:
Salida:
Usa:
Actualiza:
Efecto: Muestra la pantalla en la que los almacenistas pueden efectuar todas sus operaciones.
Excepciones:
Proceso:
-ver pantalla almacén

Función 4.4 Dar alta disco: Da de alta un disco en la base de datos.
Entrada: DATOS-DISCO
Salida:
Usa:
Actualiza: DISCOS
Efecto: Da de alta un disco en la base de datos mostrando un mensaje de que se ha hecho.
Excepciones:
Proceso:
-leer todos los datos del disco de las casillas correspondientes
MIENTRAS falta algún campo obligatorio (nombre, autor, existencias,reservados)
-mensaje (falta el dato correspondiente)
FIN-MIENTRAS
-generar nueva entrada en la BD con esos datos
-mensaje (Disco dado de alta correctamente)

Función 4.5 Dar baja disco: Da de baja un disco en la base de datos.
Entrada: DATOS-DISCO
Salida:
Usa: DISCOS
Actualiza: DISCOS
Efecto: Da de baja un disco en la base de datos mostrando un mensaje de que se ha hecho.
Excepciones:
Proceso:
-leer nombre disco de la casilla correspondiente
-SI nombre disco NO está en BD
-mensaje (disco no existe)
SINO
-leer nombre de la casilla correspondiente
-eliminar la entrada en la BD con esos datos
-mensaje (Disco dado de baja correctamente)
FIN-SI

Función 4.6 Ver disco: Muestra los datos de un disco.
Identica a la función 1.6

Capítulo 11

Implementación

En este capítulo se desarrolla la implementación del sistema. Para cada aplicación utilizada se comentan los aspectos relevantes a tener en cuenta. Los temas relacionados con la configuración se especifican en el "Manual de Instalación" (anexo B). Los distintos archivos con el código de la aplicación o utilizados como soporte se detallan en "Archivos del sistema" (anexo C), donde también se muestra su ubicación.

Para la implementación y funcionamiento de la aplicación se utilizará el siguiente software:

MySQL Gestor del sistema de bases de datos.

Apache Servidor de páginas *Web*.

HTML Lenguaje para presentar páginas *Web*.

PHP Lenguaje de programación de páginas *Web* del lado del servidor.

JavaScript Lenguaje de programación de páginas *Web* del lado del cliente.

11.1. Implementación de la base de datos

Para gestionar los datos que utiliza la aplicación es necesario crear una base de datos basándose en el modelo de datos obtenido en la fase de diseño del sistema. Esto implica definir un usuario de la base de datos, asignarle una contraseña y crear la base de datos y sus tablas, los nombres elegidos para estos elementos son:

catweb Nombre del usuario de la base de datos.

catweb2005 Contraseña de acceso a la base de datos.

catwebDB Nombre de la base de datos.

catweb.sql Archivo con las instrucciones SQL necesarias para crear la base de datos y sus tablas, así como para cargar una serie de datos iniciales en ella.

11.2. Configuración del servidor Apache

La configuración de Apache consiste en indicarle al servidor que módulos debe cargar para que reconozca los archivos PHP. Los detalles se encuentran en el Manual de Instalación, en el anexo B.2

11.3. Configuración de PHP

La descripción detallada de la instalación y configuración de PHP se encuentra en el Manual de Instalación, en el anexo B.3

11.4. Implementación de la interfaz *Web*

La implementación de la interfaz *Web* s hace en páginas que incluyen información estática representada en lenguaje HTML, información dinámica gestionada en el cliente (navegador) representada en lenguaje JavaScript, e información dinámica gestionada en el servidor representada en lenguaje PHP.

La información se adquiere a través de los campos de un formulario que el usuario debe rellenar con los datos adecuados. El sistema muestra la información solicitada en esos mismos campos y en listas desplegables, el sistema también muestra mensajes indicando situaciones de error, de datos mal introducidos o de operación realizada correctamente.

A continuación se muestran algunas páginas a modo de ejemplo. En la figura 11.1 se muestra la página inicial del sistema en la que se debe elegir como qué tipo de usuario se quiere acceder al sistema.

Una vez seleccionado el tipo de usuario, el sistema muestra una página para autentificarse introduciendo el nombre de usuario y la contraseña. Esta página para los clientes se muestra en la figura 11.2

Si los datos de acceso son correctos el sistema mostrará la página correspondiente al tipo de usuario seleccionado.

La página muestra el tipo de usuario, la fecha y hora de acceso y el nombre del usuario actual, ver figura 11.3. Se muestran los campos para introducir

Figura 11.1: Página inicial del sistema

y mostrar información, así como una colección de botones para ejecutar las funciones que puede realizar el usuario seleccionado. Finalmente un botón permite acceder a una página de ayuda que indica cómo utilizar el sistema.

Se muestran a continuación varias páginas del sistema.

En la figura 11.4, un administrador obtiene los datos de un usuario.

En la figura 11.5, un vendedor muestra los datos de un disco.

En la figura 11.6 se muestra una página en la que un administrador del sistema está actuando como vendedor. Se recuerda que los administradores, aparte de sus funciones, también podían realizar las tareas propias de vendedores y almacenistas. Una serie de botones permiten a los administradores seleccionar como qué tipo de usuario quieren operar en cada momento.

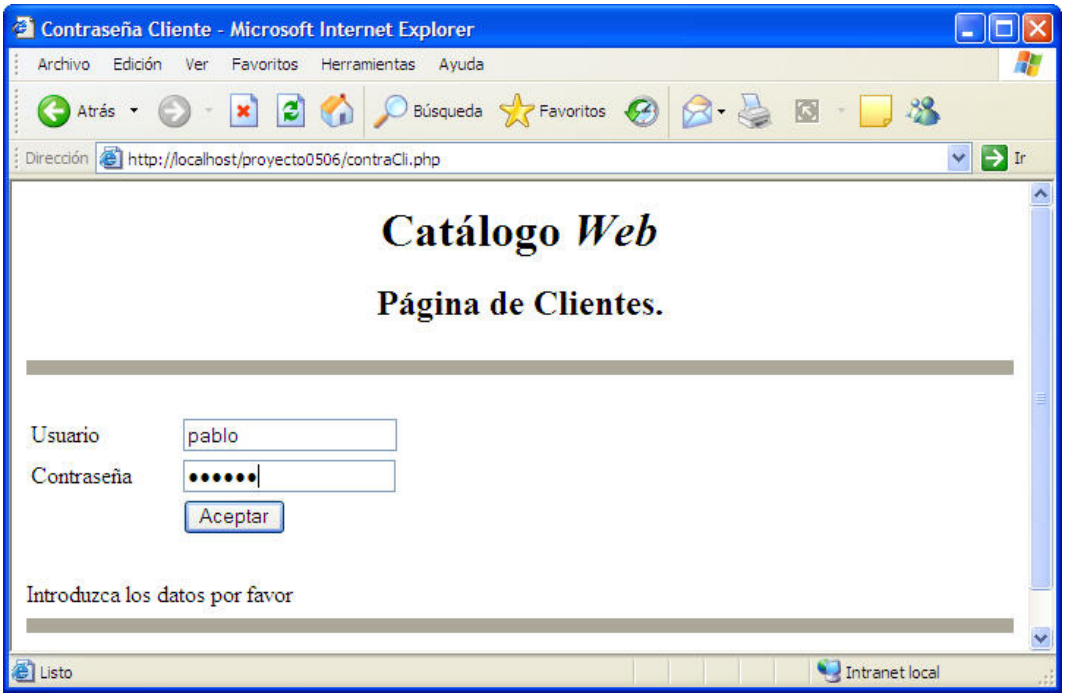

Figura 11.2: Página de acceso al sistema

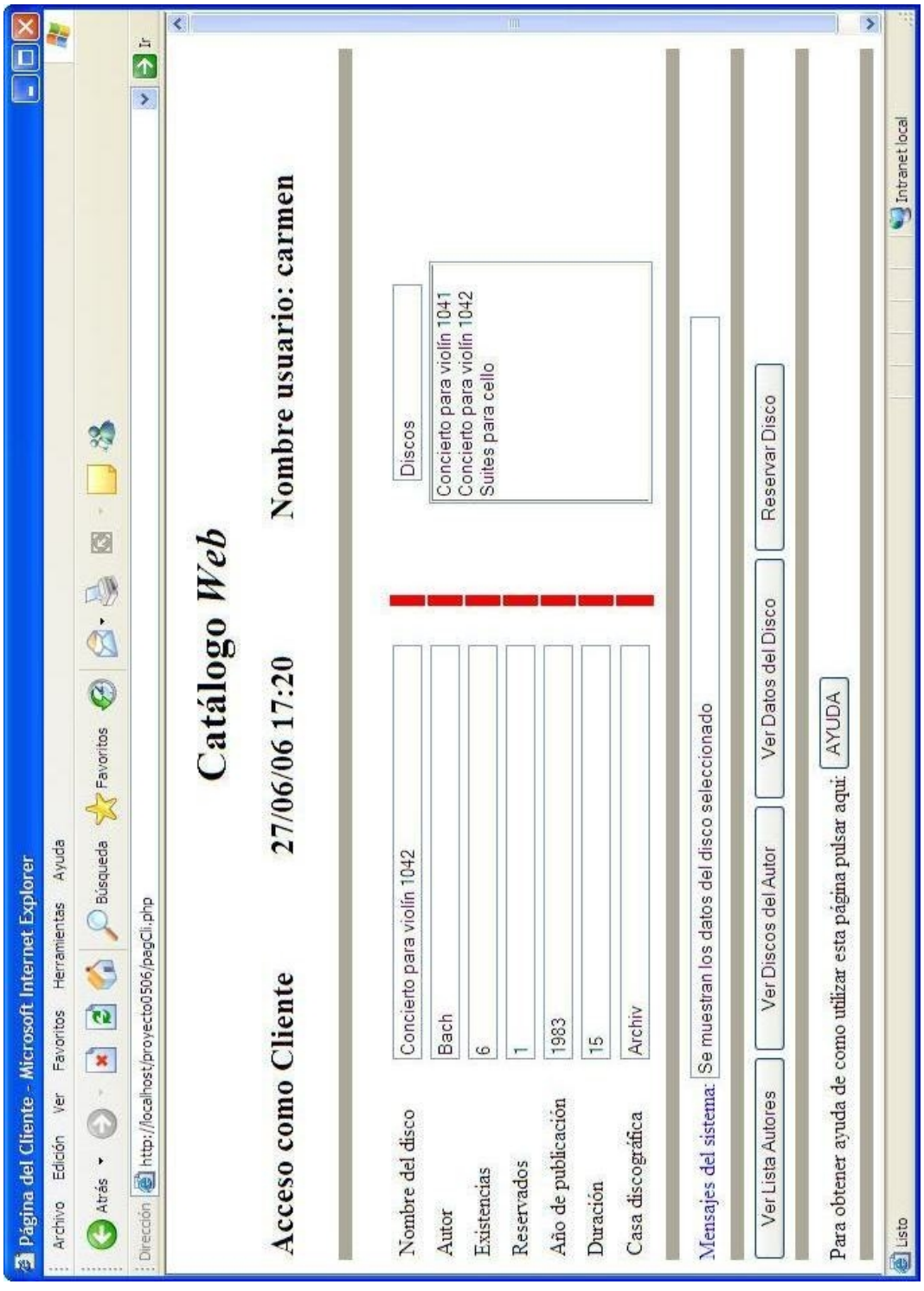

Figura 11.3: Página el la que un cliente accede a los datos de un disco.

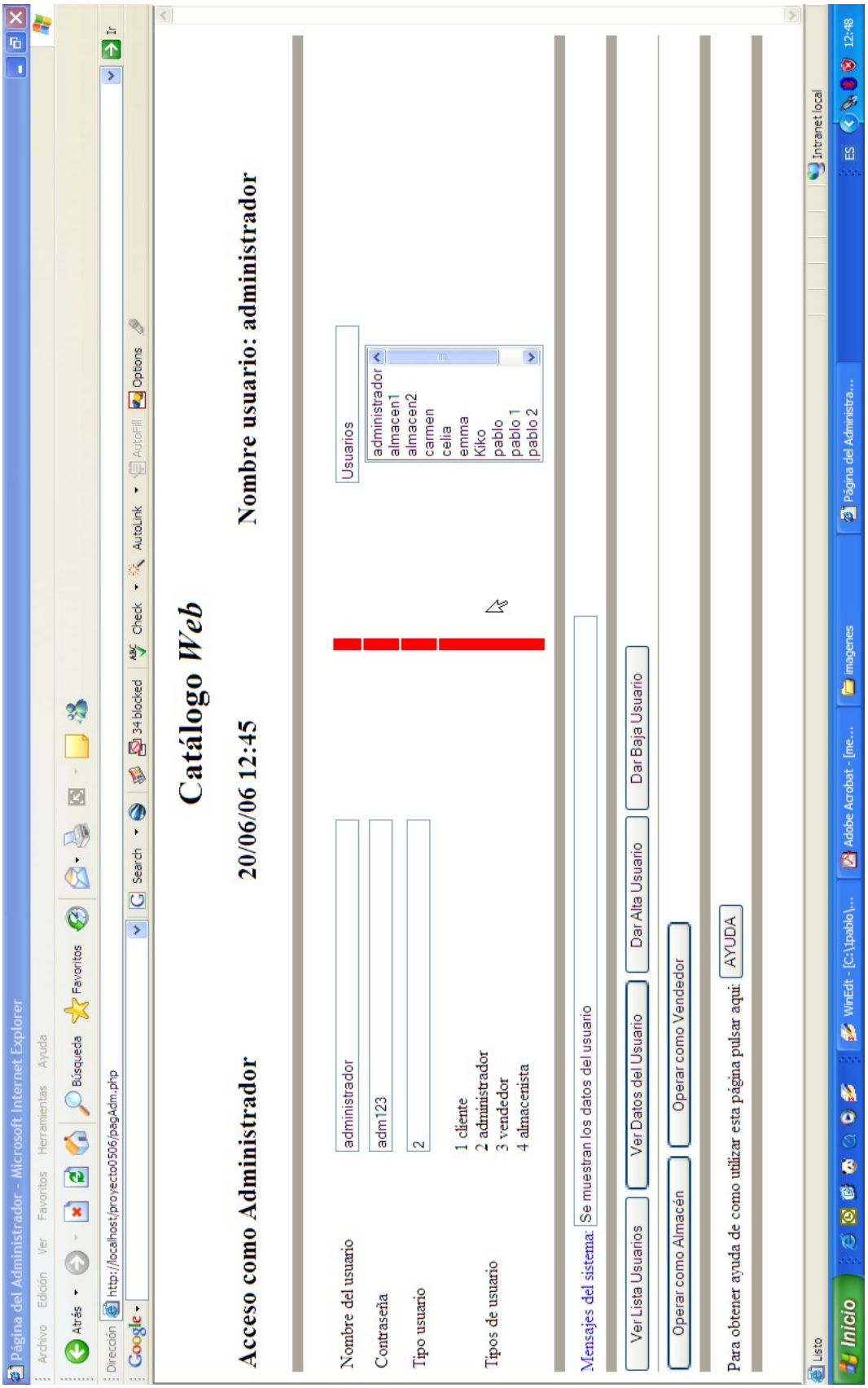

Figura 11.4: Página en la que el administrador obtiene los datos de un usuario

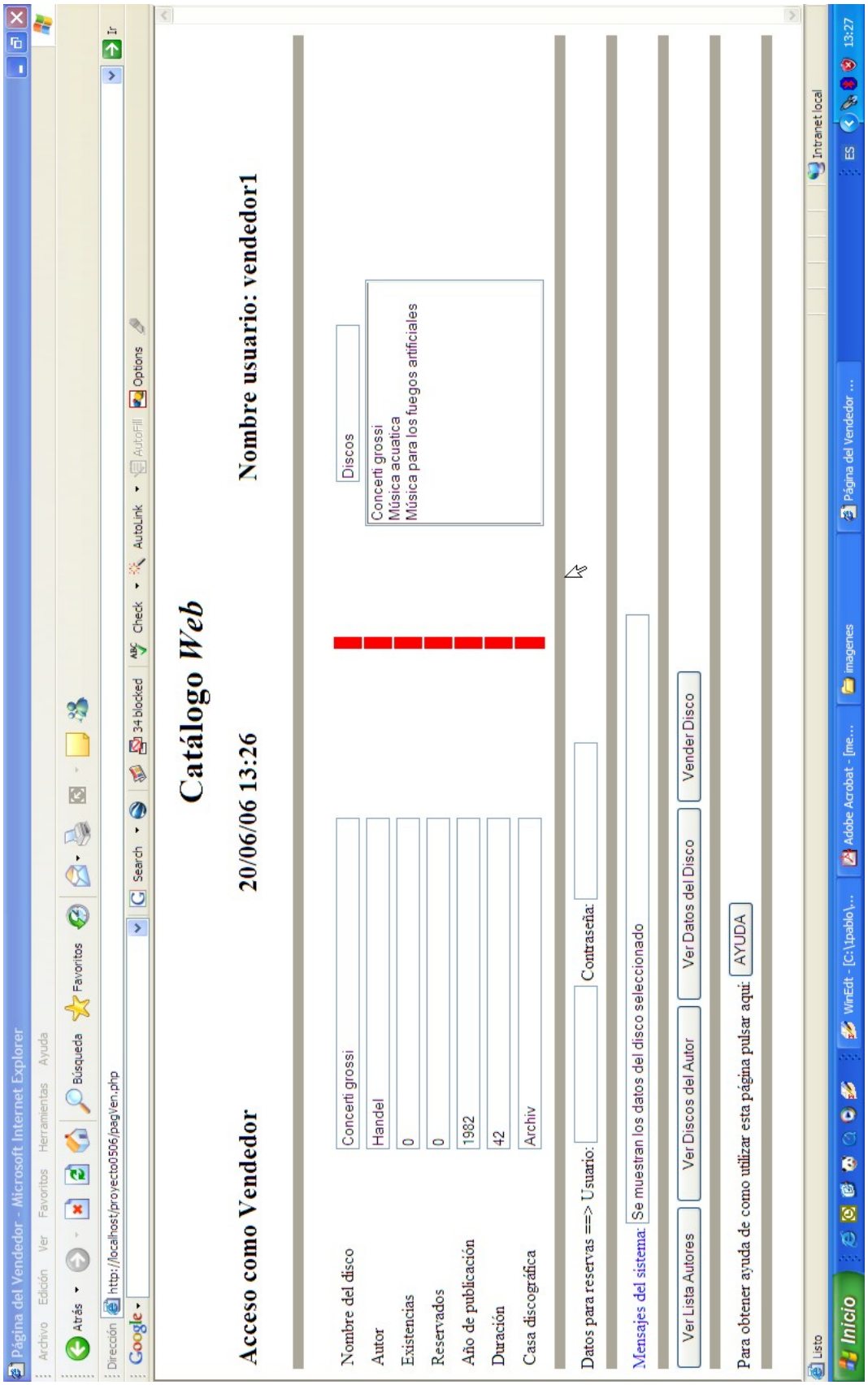

Figura 11.5: Página del vendedor mostrando datos de un disco

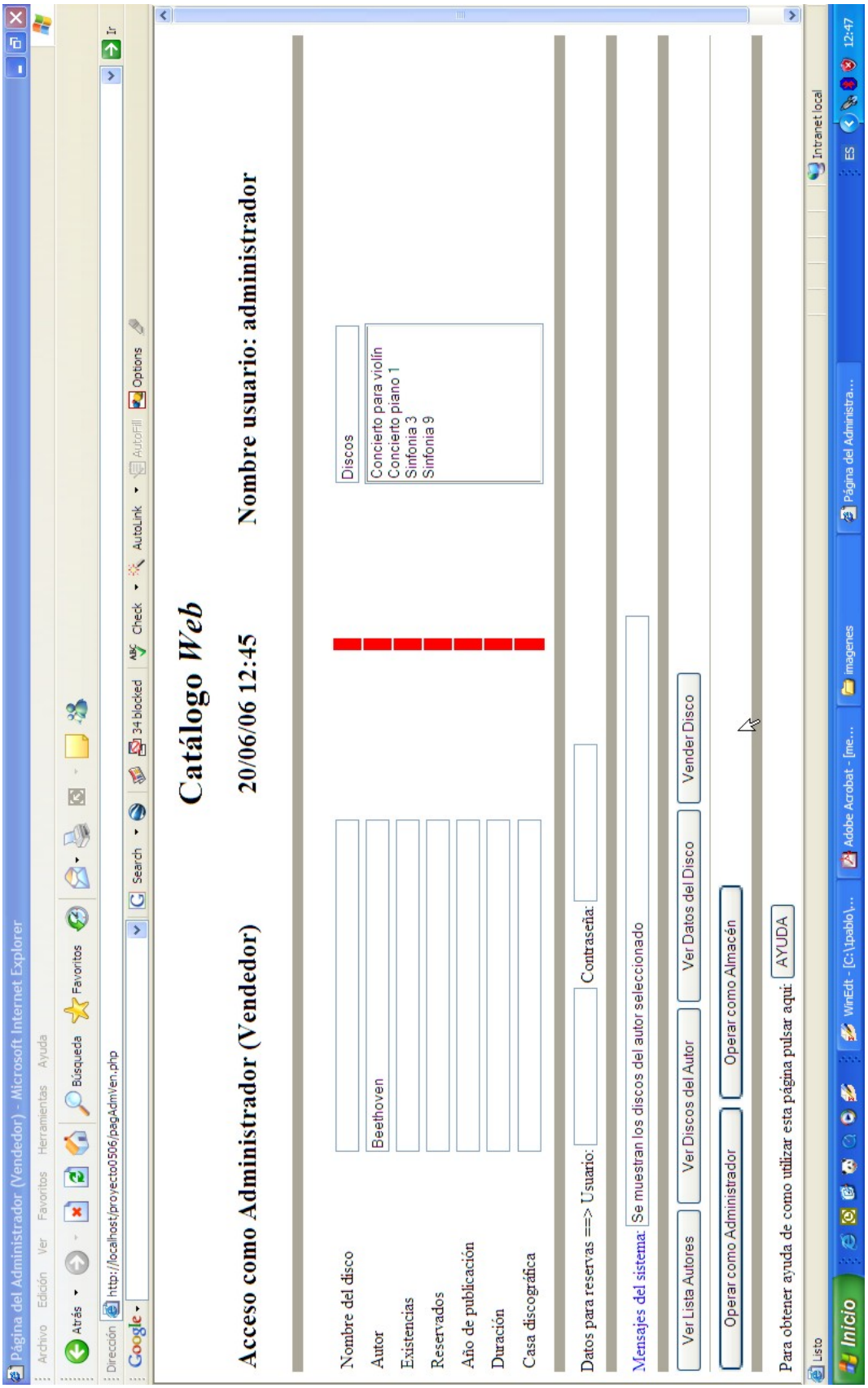

Figura 11.6: Página del administrador-vendedor

Capítulo 12

Pruebas

12.1. Pruebas de caja negra

Las pruebas de caja negra de la aplicación consistirán en analizar el comportamiento de las diferentes funciones del sistema con datos de entrada correctos e incorrectos y comprobar que responde de manera adecuada, es decir, con los resultados de la operación cuando los datos son correctos y con mensajes de error adecuados cuando son incorrectos. Como la aplicación funciona básicamente con la pulsación de botones, será a estas pulsaciones a las que se les apliquen las pruebas.

A continuación se muestran las pruebas de caja negra realizadas junto con los resultados obtenidos.

Pruebas de acceso al sistema	
Prueba realizada	**Resultado obtenido**
Pulsar botón "Cliente" en pantalla inicial	Se muestra la pantalla de contraseña del cliente
Pulsar botón "Vendedor" en pantalla inicial	Se muestra la pantalla de contraseña del vendedor
Pulsar botón "Almacén" en pantalla inicial	Se muestra la pantalla de contraseña del Almacén
Pulsar botón "Administrador" en pantalla inicial	Se muestra la pantalla de contraseña del administrador
Pantalla contraseña Cliente, Vendedor, Almacén o Administrador: usuario mal	Mensaje de usuario erróneo
Pantalla contraseña Cliente, Vendedor, Almacén o Administrador: usuario bien y contraseña mal	Mensaje de contraseña errónea
Pantalla contraseña Cliente, Vendedor, Almacén o Administrador: usuario bien y contraseña bien	Acceso a la página del Cliente, Vendedor, Almacén o Administrador

Pruebas de la página del Cliente	
Prueba realizada	**Resultado obtenido**
Pulsar botón "Ver lista autores"	Se muestra la lista de autores
Pulsar botón "Ver discos del autor" sin autor seleccionado	Mensaje de autor no seleccionado
Pulsar botón "Ver discos del autor" con autor seleccionado	Se muestran los discos del autor seleccionado
Pulsar botón "Ver datos del disco" sin disco seleccionado	Mensaje de disco no seleccionado
Pulsar botón "Ver datos del disco" con disco seleccionado	Se muestran los datos del disco seleccionado
Pulsar botón "Reservar disco" sin autor seleccionado	Mensaje de autor no seleccionado
Pulsar botón "Reservar disco" sin disco seleccionado	Mensaje de disco no seleccionado
Pulsar botón "Reservar disco" con autor y disco seleccionados	Mensaje de disco reservado o motivo por el que no se puede reservar
Pulsar botón "Ayuda"	Se muestra la página de ayuda del cliente

Pruebas de la página del Vendedor	
Prueba realizada	**Resultado obtenido**
Pulsar botón "Ver lista autores"	Se muestra la lista de autores
Pulsar botón "Ver discos del autor" sin autor seleccionado	Mensaje de autor no seleccionado
Pulsar botón "Ver discos del autor" con autor seleccionado	Se muestran los discos del autor seleccionado
Pulsar botón "Ver datos del disco" sin disco seleccionado	Mensaje de disco no seleccionado
Pulsar botón "Ver datos del disco" con disco seleccionado	Se muestran los datos del disco seleccionado
Pulsar botón "Vender disco" sin autor seleccionado	Mensaje de autor no seleccionado
Pulsar botón "Vender disco" sin disco seleccionado	Mensaje de disco no seleccionado
Pulsar botón "Vender disco" con autor y disco seleccionados	Mensaje de disco vendido o motivo por el que no se puede vender
Pulsar botón "Ayuda"	Se muestra la página de ayuda del vendedor

Pruebas de la página del Almacén	
Prueba realizada	**Resultado obtenido**
Pulsar botón "Ver lista autores"	Se muestra la lista de autores
Pulsar botón "Ver discos del autor" sin autor seleccionado	Mensaje de autor no seleccionado
Pulsar botón "Ver discos del autor" con autor seleccionado	Se muestran los discos del autor seleccionado
Pulsar botón "Ver datos del disco" sin disco seleccionado	Mensaje de disco no seleccionado
Pulsar botón "Ver datos del disco" con disco seleccionado	Se muestran los datos del disco seleccionado
Pulsar botón "Dar alta disco" sin algún campo obligatorio	Mensaje de campo no seleccionado
Pulsar botón "Dar alta disco" con todos los campos obligatorios	Si el disco ya existe lo indica sino mensaje de disco dado de alta
Pulsar botón "Modificar disco" sin autor o disco seleccionados	Mensaje del campo que falta
Pulsar botón "Modificar disco" con autor y disco seleccionados	Si el disco no existe, mensaje de error Si nuevas existencias menor que reservados, mensaje de error En otro caso mensaje de disco modificado
Pulsar botón "Dar baja disco" sin autor o disco seleccionados	Mensaje del campo que falta
Pulsar botón "Dar baja disco" con autor y disco seleccionados	Si el disco no existe, mensaje de error Si hay reservas pendientes se informa y no se produce la baja En otro caso mensaje de disco eliminado
Pulsar botón "Ayuda"	Se muestra la página de ayuda del almacén

Pruebas de la página del Administrador	
Prueba realizada	**Resultado obtenido**
Pulsar botón "Ver lista usuarios"	Se muestra la lista de autores
Pulsar botón "Ver datos del usuario" sin usuario seleccionado	Mensaje de usuario no seleccionado
Pulsar botón "Ver datos del usuario" con usuario seleccionado	Se muestran los datos del usuario seleccionado
Pulsar botón "Dar alta usuario" sin nombre, contraseña o tipo	Mensaje de campo no seleccionado
Pulsar botón "Dar alta usuario" con todos los campos obligatorios	Si el usuario ya existe lo indica, sino mensaje de usuario dado de alta
Pulsar botón "Dar baja usuario" sin nombre o contraseña seleccionados	Mensaje del campo que falta
Pulsar botón "Dar baja usuario" con nombre y contraseña seleccionados	Si el usuario no existe, mensaje de error En otro caso mensaje de usuario eliminado Si hay reservas pendientes se eliminan
Pulsar botón "Ayuda"	Se muestra la página de ayuda del administrador

12.2. Pruebas de caja blanca

Las pruebas de caja blanca de la aplicación consistirán en analizar el funcionamiento interno de las distintas funciones comprobando que los valores de retorno son adecuados y correctos y que no hay caminos internos que no producen ninguna acción o retorno.

A continuación se muestran las pruebas de caja blanca realizadas junto con los resultados obtenidos.

Función probada	Resultado obtenido
Validar()	La función siempre retorna un valor y/o un mensaje de texto
VerListaAutores()	La función siempre retorna un valor y/o un mensaje de texto
Redirige()	La función genera código HTML y JavaScript correctamente
Redirige2()	La función genera código HTML y JavaScript correctamente
VerAutor()	La función siempre retorna un valor y/o un mensaje de texto
VerDisco()	La función siempre retorna un valor y/o un mensaje de texto
ReservarDisco()	La función siempre retorna un valor y/o un mensaje de texto
VenderDisco()	La función siempre retorna un valor y/o un mensaje de texto
DarAltaDisco()	La función siempre retorna un valor y/o un mensaje de texto
ModificarDisco()	La función siempre retorna un valor y/o un mensaje de texto
DarBajaDisco()	La función siempre retorna un valor y/o un mensaje de texto
VerListaUsuarios()	La función siempre retorna un valor y/o un mensaje de texto
VerDatosUsuario()	La función siempre retorna un valor y/o un mensaje de texto
ObtenerIdUsuario()	La función siempre retorna un valor y/o un mensaje de texto
DarAltaUsuario()	La función siempre retorna un valor y/o un mensaje de texto
DarBajaUsuario()	La función siempre retorna un valor y/o un mensaje de texto

Capítulo 13

Historia del Proyecto

Durante el mes de Octubre se contacta con el Departamento de Lenguajes y Sistemas Informáticos para solicitar la realización del Proyecto Fin de Carrera ofertado.

Durante la primera quincena del mes de Noviembre se realiza un anteproyecto del trabajo para ser remitido al profesor director. El anteproyecto se centra en discutir los pasos a realizar en el trabajo y en analizar las distintas tecnologías existentes para la realización de una aplicación *Web* y elegir las que se consideran mas apropiadas. Para su realización se utilizan unas 20 horas de trabajo.

La segunda quincena de Noviembre se dedica al Análisis de Requisitos. Se hace primero un estudio teórico del tema y posteriormente el análisis práctico del proyecto (25 horas).

La primera quincena de Diciembre se dedica al Diseño. Se hace primero un estudio teórico del tema y posteriormente el diseño práctico del proyecto (25 horas).

La tercera semana de Diciembre se dedica a la implementación de la base de datos del sistema, se crea además un fichero script con datos para la carga inicial de la base de datos (12 horas).

La cuarta semana de Diciembre y todo el mes de Enero se dedica a la implementación de las páginas *Web* de la aplicación y del código PHP necesario, así como a la integración de todas ellas con la estructura de navegación adecuada (50 horas).

La primera semana de Febrero se dedica a las pruebas y depuración del código de la aplicación. Se hace un estudio de diferentes tipos de pruebas del software y se aplican algunas de ellas buscando errores y depurando los que se encuentran (6 horas).

La segunda semana de Febrero se dedica a redactar los anexos de la memoria del trabajo. El manual de usuario también se incluye como ayuda

en diferentes páginas *Web* (8 horas).

En Marzo se redacta la memoria y se envía al director para su seguimiento y evaluación (10 horas).

Durante el mes de Junio se prepara la presentación de la defensa pública del proyecto (10 horas).

El tiempo total estimado para la realización del proyecto asciende a **166 horas**.

Capítulo 14

Indicaciones para posibles desarrollos futuros

El trabajo realizado para el Proyecto Fin de Carrera es acorde con las limitaciones de tiempo y medios empleados. Disponiendo de más tiempo se podrían añadir más funcionalidades al sistema para mejorar tanto la presentación como las capacidades del mismo.

El estudio teórico, tanto de análisis de requisitos como de diseño es bastante completo y su mejora sólo sería necesaria en caso de cambios sustanciales en las necesidades de la empresa en lo relativo a la aplicación. Habría que incluir las nuevas tecnologías que aparecieran en el futuro.

Un apartado fácilmente mejorable, si se pretendiesen exceder los límites razonables de este PFC, es el contenido de la base de datos. En la tabla usuarios se podría incluir la dirección y teléfono del usuario, una fotografía, el e-mail, edad, preferencias musicales, etc. Esto permitiría catalogar a los clientes y comunicarse con ellos via e-mail. La tabla discos podría mejorarse añadiendo fotografías de la portada del disco y de los autores, fragmentos de sonido de las canciones para una audición previa, etc.

En cuanto a la aplicación se podría introducir un sistema de comunicación vía e-mail tanto con los clientes como con los usuarios internos (administradores, almacenistas y vendedores). Se podría incluir un sistema de transacciones que permitiera la venta on line, no sólo la reserva, aunque en principio esto parece un salto cualitativo importante en la complejidad de la aplicación. También podría implantarse un sistema de puntos para clientes asiduos y descuentos en función de los puntos obtenidos.

En cuanto a la presentación se mejoraría incorporando las fotografías y los sonidos comentados anteriormente.

Para un sistema con gran número de discos en catálogo y con bastantes usuarios accediendo concurrentemente sería necesario mejorar los algoritmos

de acceso a la base de datos para optimizar su rendimiento. Teniendo en cuenta el alcance y los objetivos del presente trabajo esto no se ha tenido en cuenta pues se trabaja con una base de datos pequeña y con pocos usuarios.

Para una aplicación más grande y compleja sería necesario desdoblar la estructura actual de un solo nivel en el que se mezclan las capas de presentación, de políticas de la empresa y de acceso a bases de datos en tres, o al menos, en dos capas diferenciadas. La capa de presentación contendría todo lo relativo a la interfaz gráfica, la capa de negocio se encargaría de implementar la política de la empresa: como formas de acceso, tipos de usuario, sistema de reservas y ventas, renovación del catálogo, etc, finalmente la capa de base de datos se encargaría de todo lo relativo al acceso a la base de datos. Si la aplicación no es muy compleja estas dos últimas capas pueden fundirse en una sola, pero la capa de presentación siempre debería ser independiente del resto de la aplicación.

Capítulo 15

Conclusiones

A partir de los objetivos inicialmente planteados y tras el desarrollo del trabajo se concluye la consecución de los mismos.

En el capítulo 5 (página 23) se hace una exposición detallada del problema a resolver.

En el capítulo 7.1 (página 53) se exponen los principales modelos de desarrollo del software y se justifica la elección del modelo de ciclo de vida en cascada para el desarrollo del proyecto.

En el capítulo 7.2 (página 62) se revisan los fundamentos del análisis de requisitos. Se comentan los principios y tareas del análisis y se comentan los distintos modelos que se usan para especificar el software de la aplicación.

El capítulo 7.3 (página 70) se dedica a los fundamentos del diseño del software. Se analizan los conceptos en que se basa el diseño. Se analizan los métodos de diseño orientado a flujo de datos y orientado a objetos. Se estudian el diseño de datos, arquitectónico, de interfaz y procedimental.

El capítulo 9 (página 95) se dedica a la especificación completa del software de la aplicación. Se desarrollan los modelos de datos, funcional y de comportamiento, y se analizan los casos de uso del sistema.

El capítulo 10 (página 123) se dedica a la especificación completa del diseño de la aplicación. Se desarrollan los diseños de datos, arquitectónico, de interfaz y procedimental.

El capítulo 11 (página 155) se dedica a la implementación. Se indica que software se utiliza, como hay que configurarlo para el correcto funcionamiento y donde se ubican los archivos que son utilizados por el sistema.

El capítulo 12 (página 163) se dedica a las pruebas del sistema. Se analizan las pruebas a realizar y el resultado de las mismas.

Al definir el enunciado del problema se ha intentado hacer un planteamiento tal y como lo haría una empresa real que realizara el encargo del

trabajo.

Se analizan los principales modelos de desarrollo del software: secuenciales, incrementales, evolutivos y de construcción de prototipos. Se concluye que el modelo más apropiado al problema es el modelo secuencial en el que se suceden las fases de análisis de requisitos, diseño, codificación y pruebas. Dentro de los modelos secuenciales se elige el más representativo de ellos, el "modelo de ciclo de vida en cascada".

Se revisan los principios y tareas del análisis de requisitos. Se comentan las dos tendencias principales del análisis: el análisis estructurado y el análisis orientado a objetos. Se elige el paradigma estructurado y se describen los modelos que incluye: modelado de datos, modelado funcional y flujo de información, modelado del comportamiento y diccionario de datos, completado con el estudio de casos de uso que es un concepto tomado del análisis orientado a objetos.

Se realiza el análisis de requisitos del sistema utilizando los modelos y técnicas seleccionados en las secciones anteriores.

Se analizan los conceptos de diseño, la descomposición modular con los conceptos de independencia funcional, cohesión y acoplamiento. Se analizan los métodos de diseño orientados a flujo de datos y orientados a objetos, eligiendo para el trabajo la primera metodología. Se analizan el diseño de datos, el diseño arquitectónico, diseño de la interfaz y el diseño procedimental.

Se realiza el diseño utilizando las técnicas y modelos seleccionados para obtener las funciones necesarias para satisfacer los requisitos del sistema.

Se estudian las distintas tecnologías existentes para el desarrollo *Web* y se eligen las mas convenientes para el presente trabajo que son: Apache como servidor *Web*, MySQL como gestor de bases de datos y PHP como lenguaje de implementación.

Se desarrolla un prototipo que realiza las funciones requeridas por el sistema y se efectúan pruebas para comprobar el correcto funcionamiento. Una vez depurado el software y eliminados todos los defectos detectados se considera que el prototipo es adecuado para la entrega al cliente, en este caso la empresa que solicitó el desarrollo del catálogo *Web*.

Parte IV

Anexos

Apéndice A

Manual de Usuario

Se describen a continuación los pasos necesarios para el uso correcto del sistema, estas instrucciones se encuentran también en las páginas de ayuda de la aplicación.

A.1. Entrada al sistema

Se accede al sistema abriendo la página *Web* "index.php" que se encuentra en el directorio "proyecto0506" del servidor *Web* utilizado.

En esta página se pulsará uno de los botones disponibles según queramos acceder como clientes, almacenistas, vendedores o administradores.

En la página que aparece a continuación será necesario introducir un nombre de usuario autorizado para esa página y una contraseña válida. Si los datos introducidos son erróneos se mostrará un mensaje de error, si son correctos se accederá a la página correspondiente al tipo de usuario en cuestión.

A.2. Página de Clientes

Lista de autores

Para ver la lista de autores disponibles en la tienda pulse el botón "Ver Lista Autores". Los autores se mostrarán en la lista desplegable de la pantalla.

Discos de un autor

Para ver los discos de un autor concreto siga estos pasos:

1. Muestre la lista de autores

2. Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
3. Pulse el botón "Ver Discos del Autor".
4. La lista de los discos de ese autor se mostrarán en la lista desplegable.

Datos de un disco

Para ver los datos de un disco concreto siga estos pasos:

1. Muestre la lista de autores
2. Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
3. Pulse el botón "Ver Discos del Autor".
4. La lista de los discos de ese autor se mostrarán en la lista desplegable.
5. Seleccione un disco pulsando su nombre en la lista desplegable. El nombre del disco se mostrará en la casilla "Nombre disco".
6. Pulse el botón "Ver Datos del Disco". Todos los datos del disco se mostrarán en su casilla correspondiente.

Reservar un disco

Para reservar un disco siga estos pasos:

1. Muestre la lista de autores
2. Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
3. Pulse el botón "Ver Discos del Autor".
4. La lista de los discos de ese autor se mostrarán en la lista desplegable.
5. Seleccione un disco pulsando su nombre en la lista desplegable. El nombre del disco se mostrará en la casilla "Nombre disco".
6. Pulse el botón "Reservar Disco". En la casilla "Mensajes el sistema" se informará de si la operación ha tenido éxito o no se ha podido realizar.

A.3. Página de Vendedores

Lista de autores

Para ver la lista de autores disponibles en la tienda pulse el botón "Ver Lista Autores". Los autores se mostrarán en la lista desplegable de la pantalla.

Discos de un autor

Para ver los discos de un autor concreto siga estos pasos:

1. Muestre la lista de autores
2. Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
3. Pulse el botón "Ver Discos del Autor".
4. La lista de los discos de ese autor se mostrarán en la lista desplegable.

Datos de un disco

Para ver los datos de un disco concreto siga estos pasos:

1. Muestre la lista de autores
2. Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
3. Pulse el botón "Ver Discos del Autor".
4. La lista de los discos de ese autor se mostrarán en la lista desplegable.
5. Seleccione un disco pulsando su nombre en la lista desplegable. El nombre del disco se mostrará en la casilla "Nombre disco".
6. Pulse el botón "Ver Datos del Disco". Todos los datos del disco se mostrarán en su casilla correspondiente.

Vender un disco

Para vender un disco siga estos pasos:

1. Muestre la lista de autores

2. Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
3. Pulse el botón "Ver Discos del Autor".
4. La lista de los discos de ese autor se mostrarán en la lista desplegable.
5. Seleccione un disco pulsando su nombre en la lista desplegable. El nombre del disco se mostrará en la casilla "Nombre disco".
6. Sí el disco estaba reservado pedir al cliente su nombre y contraseña e introducirlos en las casillas correspondientes.
7. Pulse el botón "Vender Disco". En la casilla "Mensajes el sistema" se informará de si la operación ha tenido éxito o no se ha podido realizar.

A.4. Página de Almacenistas

Lista de autores

Para ver la lista de autores disponibles en la tienda pulse el botón "Ver Lista Autores". Los autores se mostrarán en la lista desplegable de la pantalla.

Discos de un autor

Para ver los discos de un autor concreto siga estos pasos:

1. Muestre la lista de autores
2. Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
3. Pulse el botón "Ver Discos del Autor".
4. La lista de los discos de ese autor se mostrarán en la lista desplegable.

Datos de un disco

Para ver los datos de un disco concreto siga estos pasos:

1. Muestre la lista de autores
2. Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".

3. Pulse el botón "Ver Discos del Autor".
4. La lista de los discos de ese autor se mostrarán en la lista desplegable.
5. Seleccione un disco pulsando su nombre en la lista desplegable. El nombre del disco se mostrará en la casilla "Nombre disco".
6. Pulse el botón "Ver Datos del Disco". Todos los datos del disco se mostrarán en su casilla correspondiente.

Dar de alta un disco en la base de datos

Para dar de alta un disco siga estos pasos:

1. Seleccione un autor (hay dos formas de hacerlo)
   - **A1** Muestre la lista de autores
   - **A2** Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
   - **B1** Escriba el nombre del autor en la casilla "Autor"
2. Escriba el nombre del disco en la casilla "Nombre disco".
3. Escriba el número de existencias en el campo "Existencias".
4. Rellene los demás campos. El campo "Reservados" no tiene efecto, siempre se inicia a cero.
5. Pulse el botón "Dar Alta Disco". En la casilla "Mensajes el sistema" se informará de si la operación ha tenido éxito o no se ha podido realizar.

Modificar un disco en la base de datos

Para modificar un disco siga estos pasos:

1. Seleccione un autor (hay dos formas de hacerlo)
   - **A1** Muestre la lista de autores
   - **A2** Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".
   - **B1** Escriba el nombre del autor en la casilla "Autor"
2. Seleccione un disco (hay dos formas de hacerlo)

A1 Muestre los discos del autor seleccionado.

A2 Seleccione un disco pulsando su nombre en la lista desplegable. El nombre del disco se mostrará en la casilla "Nombre disco".

B1 Escriba el nombre del disco en la casilla "Nombre disco".

3. Rellene los demás campos. El campo "Existencias es obligatorio. El campo "Reservados" no tiene efecto, no se puede modificar. Si las nuevas existencias son menores que reservados no se realiza la modificación.

4. Pulse el botón "Modificar Disco". En la casilla "Mensajes el sistema" se informará de si la operación ha tenido éxito o no se ha podido realizar.

Dar de baja un disco en la base de datos

Para dar de baja un disco siga estos pasos:

1. Seleccione un autor (hay dos formas de hacerlo)

   A1 Muestre la lista de autores

   A2 Seleccione un autor pulsando su nombre en la lista desplegable. El nombre del autor se mostrará en la casilla "Autor".

   B1 Escriba el nombre del autor en la casilla "Autor"

2. Seleccione un disco (hay dos formas de hacerlo)

   A1 Muestre los discos del autor seleccionado.

   A2 Seleccione un disco pulsando su nombre en la lista desplegable. El nombre del disco se mostrará en la casilla "Nombre disco".

   B1 Escriba el nombre del disco en la casilla "Nombre disco".

3. Pulse el botón "Dar Baja Disco". Si nombre de disco y autor no existen da un mensaje de error Si hay reservados no se puede eliminar el disco de la base de datos y se avisa de ello. En la casilla "Mensajes el sistema" se informará de si la operación ha tenido éxito o no se ha podido realizar.

A.5. Página de Administradores

Lista de usuarios

Para ver la lista de usuarios de la aplicación pulse el botón "Ver Lista Autores". Los autores se mostrarán en la lista desplegable de la pantalla.

Datos de un usuario

Para ver los datos de un usuario concreto siga estos pasos:

1. Muestre la lista de usuarios
2. Seleccione un autor pulsando su nombre en la lista desplegable. Cuando hay varios usuarios con el mismo nombre aparece un número detrás del nombre.
3. Pulse el botón "Ver Datos del Usuario". Todos los datos del usuario se mostrarán en su casilla correspondiente.

Dar de alta un usuario en la base de datos

Para dar de alta un disco siga estos pasos:

1. Seleccione un autor (hay dos formas de hacerlo)

   A1 Muestre la lista de usuarios

   A2 Seleccione un usuario pulsando su nombre en la lista desplegable.

   A3 Pulse el botón "Ver Datos del Usuario"

   B1 Escriba el nombre del autor en la casilla "Autor"

2. Escriba la contraseña en su casilla.
3. Escriba el número de tipo de usuario en su casilla.
4. Pulse el botón "Dar Alta Usuario". En la casilla "Mensajes el sistema" se informará de si la operación ha tenido éxito o no se ha podido realizar.

Dar de baja un usuario en la base de datos

Para dar de baja un disco siga estos pasos:

1. Seleccione un usuario (hay dos formas de hacerlo)

   A1 Muestre la lista de usuarios

   A2 Seleccione un usuario pulsando su nombre en la lista desplegable.

A3 Pulse el botón "Ver Datos Usuario".

B1 Escriba el nombre del autor en la casilla "Autor"

2. Escriba la contraseña del usuario. Si en el apartado anterior eligió "Ver Datos Usuario", la contraseña ya estará seleccionada.

3. Pulse el botón "Dar Baja Usuario". Si nombre de usuario y contraseña no existen da un mensaje de error. Si hay reservados para ese usuario se eliminan las reservas. En la casilla "Mensajes el sistema" se informará de si la operación ha tenido éxito o no se ha podido realizar.

Apéndice B

Manual de Instalación

En este manual se incluyen las instrucciones para instalar, configurar y usar los distintos programas y paquetes software utilizados en la aplicación. Estos son:

MySQL Utilizado como gestor de bases de datos.

Apache Utilizado como servidor de páginas *Web*.

PHP Utilizado como lenguaje de programación de páginas *Web*.

El sistema operativo (S.O.) utilizado sera Microsoft Windows.

B.1. Configuración de MySQL

Como servidor de bases de datos se utiliza MySQL en su versión 4.1. En un sistema Windows la ubicación del servidor es: **ruta\MySQL\MySQL server 4.1**, donde "ruta" es el directorio elegido para situar el gestor de base de datos.

En la estructura de directorios, los más importantes son:

ruta\MySQL\MySQL server 4.1\bin Contiene los programas cliente y el servidor mysqld.

ruta\MySQL\MySQL server 4.1\data Contiene los archivos de bases de datos.

Para poder trabajar desde la consola del ordenador en cualquier posición es necesario modificar el PATH para que apunte a ruta\MySQL\MySQL server 4.1\bin

Para la gestión de usuarios y bases de datos se utiliza la herramienta **"MySQL administrator"** instalada en el directorio "ruta\MySQL\MySQL administrator 1.1\"

Los pasos necesarios para crear y configurar el usuario y la base de datos que se usará en el proyecto son:

1. Acceder a MySQL administrator como root
2. Crear el usuario **"catweb"**
3. Asignarle la contraseña **"catweb2005"**
4. Crear la base de datos **"catwebDB"**
5. Asignar al usuario **"catweb"** todos los privilegios sobre la base de datos **"catwebDB"**
6. Desde la consola del sistema acceder a MySQL como usuario **"catweb"**
7. Seleccionar la base de datos de trabajo con la sentencia **"USE catwebDB"**
8. Cargar un archivo que contiene lass sentencias de creación de tablas y algunos valores iniciales con la sentencia **"SOURCE catweb.sql"**. La consola se debe haber abierto desde el directorio donde se encuentra el archivo catweb.sql

B.2. Configuración del servidor Apache

Como servidor de páginas *Web* se utiliza Apache en su versión "apache_2.0.55" para Windows. La ubicación del servidor es: **"c:\archivos de programa\Apache Group\Apache2"**.

Los subdirectorios mas importantes son:

bin Contiene los archivos ejecutables del servidor.

conf Contiene los archivos de configuración del servidor.

htdocs Es el directorio donde se colocan las páginas *Web* que gestionará el servidor.

La configuración de Apache se realiza mediante la edición del archivo httpd.conf que se encuentra en la carpeta "conf". Se edita el archivo con cualquier editor de textos y se realizan los siguientes pasos:

1. Buscar la sección donde se encuentran las directivas "LoadModule" y añadir las siguientes para indicarle a Apache que cargue el módulo PHP y para asociar los archivos que tengan extensión php:
   - LoadModule php5_module "C:/windows/system32/php/php5apache2.dll"
   - AddType application/x-httpd-php .php
2. La ruta de acceso que corresponde a la biblioteca dinámica php5apache2.dll se debe adaptar adecuadamente al sitio donde se descomprimió el archivo que contenía los binarios PHP 5, en este caso la carpeta de sistema de Windows.
3. También se puede añadir la siguiente línea para que los archivos con extensión .phps y que contengan código PHP se puedan visualizar con la ayuda del color en la sintaxis:
   - AddType application/x-httpd-php-source .phps

La carpeta htdocs de la instalación Apache es la carpeta raíz del sitio *Web*. Se puede acceder a esta carpeta desde el equipo local navegando a la URL:

- **http://localhost**

Y desde otro equipo navegando a la URL:

- **http://nombreservidor**

Donde nombre servidor es el nombre del ordenador que alberga el servidor Apache.

B.3. Configuración de PHP

Como lenguaje de programación se utiliza PHP 5. Las páginas PHP se procesan en el servidor y se genera código HTML que se envía al cliente. PHP se integra perfectamente con MySQL.

Los pasos para instalarlo y configurarlo son:

- Copiar el archivo php-5.0.0-Win32.zip en la carpeta c:\PHP5 y desempaquetarlo.
- Copiar todo el contenido de la subcarpeta PHP en la subcarpeta C:\windows\system32\php

- Copiar el archivo php5ts.dll en C:\windows\system32
- Copiar el archivo php.ini-dist a C:\windows y cambiarle el nombre a php.ini
- Abrir el archivo php.ini y añadir o asignar las líneas
  doc_root = "C:\Archivos de programa\Apache Group\Apache2\htdocs"
  extension_dir = "C:\windows\system32\inetsrv\php\extensions"
  extension_dir = "C:\PHP\ext\"
  extension=php_mysql.dll
  extension=php_mysqli.dll
  session.auto_start = 1

Apéndice C

Archivos del sistema

C.1. MySQL

catweb.sql Archivo que contiene las instrucciones de creación de la base de datos y de las tablas y unos datos mínimos para cada tabla. Este archivo se cargará una sola vez al instalar la aplicación.

C.2. PHP

Todos los archivos de la aplicación tienen extensión .php y se sitúan en el directorio "proyecto0506" que a su vez se sitúa en el directorio raíz del servidor apache del ordenador donde reside la aplicación.

Los archivos en cuestión son:

- ayudaAdm.php
- ayudaAlm.php
- ayudaCli.php
- ayudaVen.php
- contraAdm.php
- contraAlm.php
- contraCli.php
- contraVen.php
- funciones.php

- index.php
- pagAdm.php
- pagAdmAlm.php
- pagAdmVev.php
- pagAlm.php
- pagCli.php
- pagVen.php

La página inicial de la aplicación es index.php, luego para acceder habrá que teclear:

http://localhost/proyecto0506/index.php

Donde localhost se puede sustituir por el nombre del computador donde reside la aplicación.

Apéndice D

Listado de siglas, abreviaturas y acrónimos

AOO Análisis Orientado a Objetos

BD Base de Datos

CSS Cascade Sheets Style, Hojas de Estilo en Cascada.

DER Diagrama Entidad-Relación, utilizado en el modelo de análisis estructurado.

DFD Diagrama de Flujo de Datos, utilizado en el modelo de análisis estructurado.

DTD Document Type Definition, es una definición en un documento SGML ó XML que específica restricciones en la estructura del mismo.

DTE Diagrama de Transición de Estados, utilizado en el modelo de análisis estructurado.

EC Especificación de Control, utilizado en el modelo de análisis estructurado.

EP Especificación de Proceso, utilizado en el modelo de análisis estructurado.

HTML HyperText Markup Language, Lenguaje de marcación de hypertexto.

HTTP HyperText Transfer Protocol, Protocolo de Transferencia de Hypertexto.

JSP Jackson Structured Programming.

IGU Interfaz Gráfica de Usuario.

IHM Interfaz Hombre-Máquina.

Página *Web* Documento digital que contiene información específica de un tema en particular y que es almacenado en algún sistema de cómputo que se encuentre conectado a la red mundial de información denominada Internet, de tal forma que este documento pueda ser consultado por cualquier persona que se conecte a esta red mundial de comunicaciones y que cuente con los permisos apropiados para hacerlo. Una página *Web* es la unidad básica del *World Wide Web*.

PFC Proyecto Fin de Carrera.

SGML Standard Generalized Markup Language (Lenguaje de Marcación Generalizado), es un sistema para la organización y etiquetado de documentos.

Sitio *Web* Conjunto de archivos electrónicos y páginas *Web* referentes a un tema en particular, que incluye una página inicial de bienvenida, generalmente denominada *home page*, con un nombre de dominio y dirección en Internet específicos.

SQL Strudtured Querry Language, lenguaje estructurado de consultas para bases de datos.

SRD Software Requeriments Document (Documento de especificación de requisitos)

UNED Universidad Nacional de Educación a Distancia.

Web World Wide Web, o simplemente *Web*, es el universo de información accesible a través de Internet.

XHTML eXtensible HyperText Markup Language (lenguaje extensible de marcado de hipertexto).

XML eXtensible Markup Language, Lenguaje de Marcado Extensible.

XSL eXtensible Stylesheet Language, (lenguaje extensible de hojas de estilo), lenguaje para transformar, formatear y presentar la información de un documento XML en un medio específico.

Bibliografía

[ACDPSU+04] I. Aedo Cuevas, P. Díaz Pérez, M.A. Sicilia Urbán, A. Vara de Llamo, A. Colmenar Santos, P. Losada de Dios, F. Mur Pérez, M.A. Castro Gil, and J. Peire Arroba. *Sistemas Multimedia: Análisis, Diseño y Evaluación*. UNED, 2004.

[CA02] G. Cuevas Agustín. *Gestión del proceso software*. Centro de estudios Ramón Areces, 2002.

[Cas00] E. Castro. *HTML 4, cuarta edición*. Prentice Hall, 2000.

[CC95] J. A. Cerrada and M. Collado. *Introducción a la Ingeniería del Software*. editorial UNED, 1995.

[Cer] S. Ceria. Casos de uso, un método practico para explorar requerimientos. Universidad de Buenos Aires.

[CSLSMR+03] B. Cascales Salinas, P. Lucas Saorín, J. M. Mira Ros, A. J. Pallarés Ruiz, and S. Sánchez-Pedreño Guillén. *El libro de LaTeX*. Pearson Educación, 2003.

[D´Aa] E. D´Andrea. *MySQL 5 Curso profesional de programación de bases de datos*. INFORBOOK´S.

[D´Ab] E. D´Andrea. *PHP 5 Curso profesional de programación*. INFORBOOK´S.

[Dat86] C. J. Date. *Introducción a los sistemas de Bases de Datos*. Addison-Wesley Iberoamericana, 1986.

[GFLR98] L. Grau Fernández and I. López Rodríguez. *Problemas de Bases de Datos*. Sanz y Torres, 1998.

[Hum01] W. S. Humphrey. *Introducción al Proceso Software Personal PSP*. Addison Wesley, 2001.

[Keo05] Jim Keogh. *JavaScript.* Anaya Multimedia, 2005.

[KSS98] H. F. Korth, A. Silberschatz, and S. Sudarshan. *Fundamentos de Bases de Datos, tercera edición.* McGraw Hill, 1998.

[Mañ05] J. A. Mañas. Ideas para realizar el proyecto fin de carrera. [en línea, 2001], [Noviembre, 2005]. http://jungla.dit.upm.es/ pepe/pfc.html.

[Mun01] S. Munz. Tecnologías *Web.* [en línea], 2001. http://es.selfhtml.org/introduccion/tecnologias/.

[Oet05] T. Oetiker. The not so short introduction to LaTeX 2. [en línea], 2005. http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf.

[Pre97] R. S. Pressman. *Ingeniería del Software, un enfoque práctico.* McGraw Hill, cuarta edición, 1997.

[W3C05] W3C. Consorcio world wide web. [en línea], [Noviembre, 2005]. http://www.w3c.es/.

[Wik06] Wikipedia. Enciclopedia libre multilingüe. [en línea], [2005,2006]. http://www.wikipedia.org/.