Catalogo.NET
Sistema multicliente con arquitectura SOA basado en tecnología .NET

José María Aguilar Esteban
Ingeniería Técnica en Informática de Sistemas

Consultor: David Gañán Jiménez

Junio de 2007

A Inma, mi esposa, sin cuyo apoyo habría sido imposible llegar a este punto.
A Inma y María, nuestras hijas, por el tiempo que no he podido dedicarles.

Vuestro es el mérito, yo sólo he tenido que hacer los deberes.

Os quiero.

Resumen

La plataforma .NET ha supuesto, desde su aparición, una constante revolución en el mundo del desarrollo de software. Nuevos lenguajes con características cada vez más potentes, frameworks y librerías de componentes reutilizables de gran alcance, compatibilidad y versiones específicas para múltiples dispositivos y sistemas operativos, facilidades de comunicación a través de estándares, y entornos de desarrollo impensables hace unos años hacen de esta plataforma una tecnología ideal para el desarrollo de sistemas empresariales.

Por otra parte, desde el punto de vista arquitectural, el paradigma SOA conceptualiza el software estableciendo una separación entre las aplicaciones y los componentes de lógica empresarial, proponiendo que éstos sean diseñados como servicios independientes de la tecnología, alto nivel de abstracción, gran independencia, muy reutilizables, y un débil o nulo acoplamiento con las aplicaciones que los consumen o utilizan.

El presente Trabajo de Fin de Carrera une ambos conceptos mediante el análisis, diseño y desarrollo de un Catálogo de Productos con Arquitectura Orientada a Servicios y tecnología .NET como marco de trabajo. Asimismo, se han introducido como valor añadido una serie de tecnologías novedosas que contribuyen, sin duda, a la mejora del resultado final.

El sistema completo creado incluye los siguientes elementos:

- Un sistema central, compuesto a su vez por los componentes de persistencia (implementada utilizando el ORM NHibernate contra SQL Server), los elementos de lógica de negocio y un interfaz basado en Servicios Web XML que facilita el acceso remoto a estos últimos.

- Una aplicación de escritorio de gestión del catálogo de productos, utilizando la tecnología de reciente aparición Windows Presentation Foundation (WPF), y conectado a la central a través de Servicios Web.

- Un sistema web completo de consulta del catálogo, utilizando ASP.NET, XHTML, CSS, y AJAX, e incluso teniendo en cuenta criterios de accesibilidad WAI a nivel doble A. Al igual que los demás clientes, utilizará los servicios proporcionados por el sistema central comunicándose con él a través de Servicios Web XML.

- Una aplicación para Pocket PC 2003, utilizando Compact Framework, que también utiliza Servicios Web para comunicar con la central de servicios.

En todos los casos se ha utilizado C#, el lenguaje por excelencia de la plataforma .NET.

Índice de Contenidos

1. Introducción 7
1.1. Justificación y contexto del proyecto 7
1.2. Descripción del proyecto 7
1.3. Objetivos 9
1.4. Planificación 10
1.4.1. Ciclo de vida 10
1.4.2. Detalle de actividades 10
1.4.3. Temporalización del proyecto 13
1.5. Herramientas utilizadas 15
1.6. Productos obtenidos 15
1.7. Estructura de este documento 15
2. Requisitos iniciales 16
2.1. Escenario de partida 16
2.2. Usuarios a considerar 17
2.3. Requisitos funcionales 17
2.3.1. Funcionalidades de Seguridad 17
2.3.2. Funcionalidades del Catálogo 18
2.4. Requisitos no funcionales 19
2.4.1. Requisitos de interfaz 19
2.4.2. Requisitos de seguridad 20
2.4.3. Requisitos de información 20
3. Análisis del sistema 21
3.1. Diagramas de casos de uso 21
3.2. Descripción textual de casos de uso 23
3.2.1. CU01-Iniciar sesión 23
3.2.2. CU02-Finalizar sesión 23
3.2.3. CU03-Cambiar la clave 23
3.2.4. CU04-Buscar productos 23
3.2.5. CU05-Visualizar producto 23
3.2.6. GU01-Gestionar usuarios 23
3.2.7. GU01-Crear usuario 23
3.2.8. GU03-Modificar usuario 23
3.2.9. GU04-Eliminar usuario 23
3.2.10. GF01-Gestionar familias 23
3.2.11. GF02-Crear familia 23
3.2.12. GF03-Modificar familia 23
3.2.13. GF04-Eliminar familia 23
3.2.14. GS01-Gestionar subfamilias 23
3.2.15. GS02-Crear subfamilia 23
3.2.16. GS03-Modificar subfamilia 23
3.2.17. GS04-Eliminar subfamilia 23
3.2.18. GP01-Gestionar productos 23
3.2.19. GP02-Crear producto 23
3.2.20. GP03-Modificar producto 23
3.2.21. GP04-Eliminar producto 23
3.3. Requisitos no funcionales 23
3.3.1. Detalle de requisitos de interfaz 23
3.4. Entidades del dominio 23
4. Diseño 23
4.1. Arquitectura global 23
4.1.1. Vista física 23
4.1.2. Vista lógica 23
4.1.3. Vista de componentes 23
4.2. Decisiones tecnológicas 23

4.3. Diagrama estático de diseño 23
4.4. Diagrama de secuencia 23
4.5. Diseño de la persistencia 23
4.5.1. Modelo relacional de la base de datos 23
4.5.2. Diagrama de base de datos 23
4.6. Prototipos de la interfaz de usuario 23
4.6.1. Aplicación de escritorio 23
4.6.2. Aplicación móvil 23
4.6.3. Aplicación Web 23
5. Implementación 23
5.1. Componente "CatalogoCentral" 23
5.1.1. Entidades 23
5.1.2. Gestores 23
5.1.3. Gestor de sesiones interno 23
5.1.4. Excepciones 23
5.1.5. Persistencia de entidades 23
5.1.6. Persistencia de imágenes 23
5.2. Servicio "ServicioCatalogo" 23
5.2.1. Página web de comprobación de configuración 23
5.2.2. Control de acceso 23
5.2.3. Métodos de servicio 23
5.3. Aplicación "CatalogoPDA" 23
5.3.1. Conexión con el servidor central 23
5.3.2. Archivos de configuración 23
5.3.3. Generación de imágenes 23
5.3.4. Capturas de pantalla del producto 23
5.4. Aplicación "CatalogoWPF" 23
5.4.1. Creación del interfaz de usuario 23
5.4.2. El archivo "code-behind" 23
5.4.3. Databindings 23
5.4.4. Procesado de imágenes 23
5.4.5. Capturas de pantalla del producto 23
5.5. Aplicación "CatalogoWEB" 23
5.5.1. Uso de estándares 23
5.5.2. Implementación del sitio web completo 23
5.5.3. Páginas de error 404 personalizadas 23
5.5.4. Mapa web, migas de pan 23
5.5.5. Otros elementos de accesibilidad 23
5.5.6. Cambio de clave AJAX 23
5.5.7. Buscador de productos AJAX 23
5.5.8. Obtención de imágenes 23
5.5.9. Capturas de pantalla 23
6. Conclusiones 23
7. Glosario 23
8. Fuentes de información 23

Índice de ilustraciones

Ilustración 1. Vista general del sistema 9
Ilustración 2. Ciclo de vida en cascada 10
Ilustración 3. Gantt de la planificación del proyecto 14
Ilustración 4. Casos de uso 21
Ilustración 5. Casos de uso: Gestión del catálogo 22
Ilustración 6. Casos de uso: Gestión de usuarios 22
Ilustración 7. Modelo del dominio 23
Ilustración 8. Vista física en escenario reducido 23
Ilustración 9. Vista física en escenario extendido 23
Ilustración 10. Arquitectura lógica 23
Ilustración 11. Arquitectura lógica detallada 23
Ilustración 12. Vista de componentes del sistema 23
Ilustración 13. Diagrama estático de diseño 23
Ilustración 14. Diagrama de secuencia 23
Ilustración 15. Correspondencia de componentes con capas lógicas 23
Ilustración 16. Diagrama de Base de Datos 23
Ilustración 17. Prototipo WPF: Acceso al sistema 23
Ilustración 18. Prototipo WPF: Ficha del producto 23
Ilustración 19. Prototipo PDA: Acceso al sistema 23
Ilustración 20. Prototipo PDA: Búsqueda de productos 23
Ilustración 21. Prototipo Web: Acceso al sistema 23
Ilustración 22. Prototipo Web: Cambio de clave 23
Ilustración 23. Jerarquía de excepciones 23
Ilustración 24. Página de comprobación de parámetros del servicio web 23
Ilustración 25. Diagrama de secuencia del uso de un servicio 23
Ilustración 26. Pantallas finales de CatalogoPDA 23
Ilustración 27. Composición de interfaces con Microsoft Expression Blend 23
Ilustración 28. Diseñador de Visual Studio para WPF 23
Ilustración 29. Menú principal de CatalogoWPF 23
Ilustración 30. Pantalla de cambio de clave en CatalogoWPF 23
Ilustración 31. Gestión de productos en CatalogoWPF 23
Ilustración 32. Ficha de productos en CatalogoWPF 23
Ilustración 33. Ficha de usuarios en CatalogoWPF 23
Ilustración 34. Confirmación de eliminación de familias en CatalogoWPF 23
Ilustración 35. Uso de estándares en CatalogoWeb 23
Ilustración 36. Página de error personalizada en CatalogoWeb 23
Ilustración 37. Archivo Web.sitemap 23
Ilustración 38. Características de accesibilidad de CatalogoWEB 23
Ilustración 39. Propiedades del extensor PasswordStrength 23
Ilustración 40. Cambio de clave al estilo AJAX 23
Ilustración 41. Panel de criterios de búsqueda plegado 23
Ilustración 42. Desplegables enlazados en criterios de búsqueda 23
Ilustración 43. Autocompletador AJAX en acción 23
Ilustración 44. Imagen por defecto de un producto 23
Ilustración 45. Pantalla de fin de sesión en CatalogoWEB 23
Ilustración 46. Ficha de productos en CatalogoWEB 23

1. Introducción

1.1. Justificación y contexto del proyecto

El presente Trabajo de Fin de Carrera se realiza en el contexto del área temática .NET, cuyo objeto es la creación de una aplicación basándose en esta plataforma tecnológica, con indicación expresa de utilización de WPF, patrones MVC o cualquier tecnología adicional que aportase valor añadido al producto.

De los distintos escenarios de aplicación propuestos para la realización del TFC, este proyecto se centra en el desarrollo de un Catálogo de Productos. La elección se ha realizado simplemente por tratarse de un tema cercano y conocido, facilitando la obtención de requisitos funcionales y permitiendo una orientación del proyecto más hacia la utilización de tecnologías que hacia la implementación de multitud de funcionalidades. Además, un catálogo de productos resulta muy rico en cuanto a la información a manejar, los dispositivos desde los que tiene sentido acceder al mismo, y aspectos asociados a una herramienta de gestión como la transaccionalidad en las operaciones o seguridad en los accesos.

Así, para dar un mayor sentido y coherencia al proyecto, se ha ideado un escenario en el que una ficticia empresa comercializadora de productos informáticos expone una serie de necesidades funcionales y operacionales que deben ser cubiertas. Este escenario se recoge detalladamente en la sección correspondiente al análisis en este mismo documento.

Una vez decidido el sistema a desarrollar, se han establecido las bases tecnológicas en las que se sustentaría la solución. La mayoría de decisiones tomadas se basaron en el interés por el aprendizaje y utilización de nuevas técnicas más que en la necesidad real impuesta por el software a desarrollar que, como se ha comentado anteriormente, era relativamente simple desde el punto de vista funcional.

1.2. Descripción del proyecto

El presente proyecto tiene por objeto realizar el análisis, diseño e implementación de un sistema software empresarial, concretamente un Catálogo de Productos, utilizando el paradigma arquitectural denominado SOA (Service-Oriented Application) y la tecnología .NET de Microsoft.

El modelo SOA representa una evolución en los sistemas distribuidos proponiendo la separación y desacoplamiento entre los componentes de proceso de negocio del dominio del problema, que son encapsulados en forma de *servicios*, y el resto de elementos de la solución, que consumen los servicios proporcionados por los primeros para dar cobertura a las funcionalidades propuestas.

Los *servicios* son aplicaciones independientes que exponen sus funcionalidades a través de interfaces basadas en mensajes a las que es posible acceder a través de una red utilizando tecnologías estándar y protocolos abiertos como Web Services y su conjunto de protocolos asociados (XML, SOAP, WSDL...) siempre que se cumplan sus contratos o especificaciones exactas, que también son publicadas para referencia de los clientes (o consumidores) de los mismos.

A lo largo del Trabajo de Fin de Carrera se ha desarrollado un sistema central que implementa todos los procesos de negocio relacionados con el catálogo de productos y el almacenamiento de datos en forma de *servicios*, cumpliendo las premisas básicas dictadas por este paradigma arquitectural:

- *Independencia de la tecnología.* Dado el encapsulamiento de los servicios, el software que los consume no puede ni debe ser consciente de la tecnología subyacente, lo que facilitaría la evolución o sustitución de la misma de forma transparente.

- *Independencia entre servicios.* La implementación de los servicios debe realizarse sin asumir orden de llamadas entre ellos y sin establecer dependencias de cualquier tipo. Cada uno realizará su tarea, siendo ajeno al resto.

- *Alta granularidad.* Los servicios deben exponer un conjunto de funcionalidades reducido, de muy alto nivel y con tareas concretas a realizar, evitando así el mal uso de funciones de un nivel de abstracción menor.

- *Escaso acoplamiento*. Los servicios están muy débilmente acoplados con las aplicaciones que hacen uso de ellos. El acoplamiento existe únicamente en cuanto al contrato definido por los interfaces funcionales.

- *Atomicidad y transaccionalidad*. Un servicio debe realizar la tarea concreta a la que se destina de forma total, dejando siempre al sistema en un estado coherente incluso en caso de producirse algún problema.

- *Gran reusabilidad*. Los servicios deben ser reutilizables de forma sencilla por otros servicios, aplicaciones o, en general, cualquier tipo de agente software.

- *Orientación a mensajes*. Las llamadas a los servicios debe poder realizarse a través de mensajes, enviados de forma síncrona o asíncrona. Es habitual, asimismo, el uso de estándares abiertos como SOA o WSDL para facilitar la comunicación.

Gracias a estas características es sencillo construir clientes heterogéneos (aplicaciones de usuario de cualquier tipo, componentes u otros servicios) que accedan a estos servicios de forma muy similar. Por ello, a lo largo del presente proyecto han sido desarrollados:

- Una aplicación de escritorio que actúa como consumidor directo de los servicios expuestos desde el servidor central. Se comunicará con éste a través de una red TCP/IP, como podría ser una red local o a través de Internet.

- De la misma forma, una aplicación móvil que accede a funcionalidades expuestas desde la central a través de invocaciones remotas.

- Una aplicación Web que actúa como componente cliente del servicio. En este caso, se pretende que el usuario utilice su navegador para interactuar con esta aplicación que, a su vez, utilizaría los servicios centrales como un consumidor más.

Un escenario general de funcionamiento podría ser, por tanto, el reflejado de la siguiente ilustración:

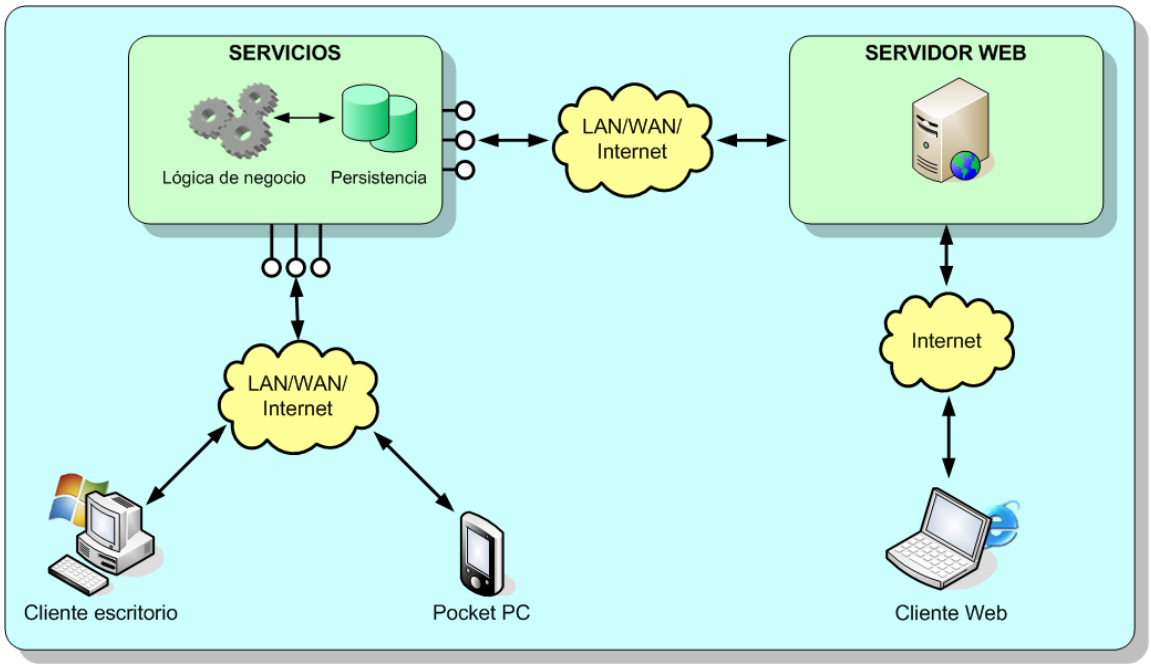


Ilustración 1. Vista general del sistema

La tecnología .NET ofrece herramientas específicas para el desarrollo de cada una de las aplicaciones descritas y la sencilla integración entre ellas utilizando protocolos estándar, por lo que se erige como una plataforma perfecta para la implementación del proyecto.

1.3. Objetivos

Los objetivos que se pretenden conseguir mediante la realización de este trabajo son los siguientes:

- Poner en práctica conocimientos adquiridos a lo largo de la carrera sobre ingeniería del software y técnicas de desarrollo.
- Profundizar en la tecnología .NET para el desarrollo de soluciones de gestión, tomando como referencia un catálogo estándar de productos.
- Aprender a trabajar con las diversas herramientas de desarrollo, tanto incluidas en Visual Studio como externas, para la composición de interfaces en dispositivos y entornos heterogéneos.
- Trabajar con tecnologías de actualidad como Servicios Web, WPF, ORMs, AJAX y otros que serán descritos más adelante.
- Demostrar la validez del paradigma de Orientación a Servicios como fundamento arquitectural de una aplicación de gestión.
- Demostrar la solvencia de la tecnología .NET para la implementación de este tipo de soluciones distribuidas.

1.4. Planificación

1.4.1. Ciclo de vida

El proyecto ha sido desarrollado tomando como base el ciclo de vida clásico de creación de software (también denominado ciclo en cascada) ligeramente adaptado a las necesidades del Trabajo de Fin de Carrera, y se ha compuesto de las siguientes etapas:

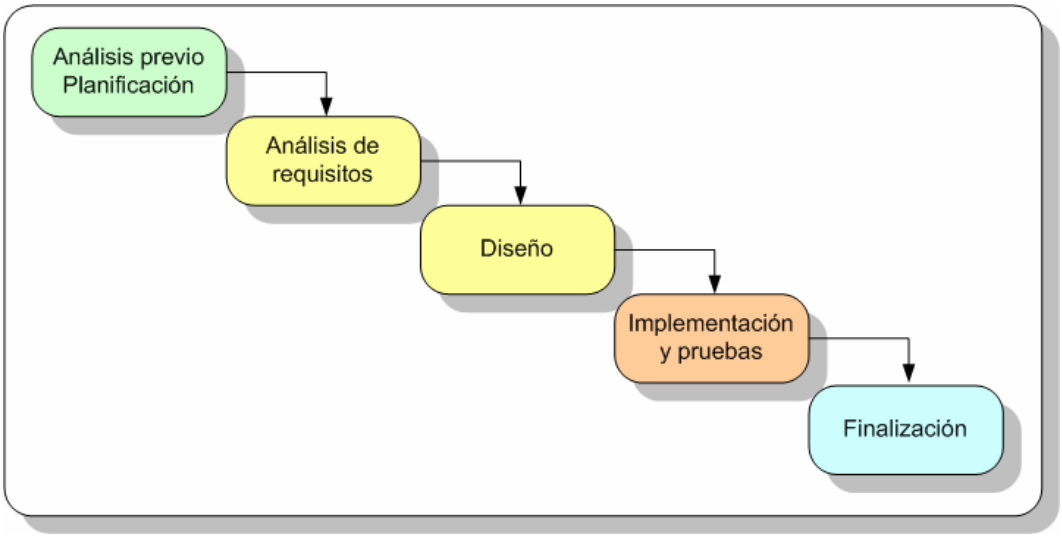


Ilustración 2. Ciclo de vida en cascada

A lo largo de las distintas fases planteadas se ha ido generando documentación que ha sido entregada en primer lugar en los hitos establecidos inicialmente en el plan de trabajo, y cuyos contenidos han sido condensados en esta memoria final del proyecto.

1.4.2. Detalle de actividades

Las siguientes subsecciones recogen, por cada etapa que compone el ciclo de vida, las principales actividades realizadas con objeto de obtener un producto que cumpliese los requisitos establecidos, con la calidad esperada y dentro de los plazos definidos.

1.4.2.1. Análisis previo y planificación

En esta etapa se ha pretendido obtener una visión global del sistema a crear, así como establecer una guía temporalizada de las actividades a llevar a cabo.

Actividad	Descripción
Selección del proyecto	Selección del proyecto entre las dos opciones planteadas.
Preparación del proyecto	Lectura de materiales, descarga de software
Definición del proyecto	Conceptualización del proyecto a realizar, a nivel de requisitos y tecnología a emplear.
Planificación del proyecto	Temporalización del proyecto conforme a la metodología a utilizar y los hitos de la asignatura.
Creación del documento	Generación del Plan de Trabajo
Envío de la PEC1	Envío del documento correspondiente a esta etapa

1.4.2.2. Análisis de requisitos

Durante esta fase se han detallado las necesidades a cubrir por el software, tanto desde el punto de vista funcional como operacional, sin entrar en detalles de implementación o tecnología a utilizar en el desarrollo de la solución.

Las actividades realizadas fueron las siguientes:

Actividad	Descripción
Especificación de requisitos funcionales	Creación y documentación de casos de uso
Especificación de requisitos no funcionales	Identificación y documentación de requisitos no funcionales

1.4.2.3. Diseño

Esta etapa ha dado forma al sistema informático definiendo cómo debería ser implementado para cubrir las necesidades y objetivos planteados en las fases anteriores, incluyendo ya aspectos concretos sobre las tecnologías a utilizar.

Las actividades realizadas fueron las siguientes:

Actividad	Descripción
Diseño arquitectónico	Diseño de la arquitectura de la solución
Diseño de clases	Creación del diagrama estático de diseño
Diseño de la persistencia	Definición de la estructura de la base de datos
Diseño de interfaces de usuario	Creación de prototipos no funcionales de interfaces de usuario para cada tipo de interfaz.
Estudio de alternativas tecnológicas	Estudio de distintas tecnologías disponibles para la implementación de determinados componentes (MVC, persistencia, etc.), siempre dentro del marco actual.
Preparación del documento	Recopilación de información y preparación del documento de análisis y diseño de la aplicación.
Entrega PEC2	Envío de la PEC2.

1.4.2.4. Implementación y pruebas

Durante esta fase se ha realizado la implementación del software teniendo en cuenta las especificaciones definidas hasta el momento, así como las pruebas necesarias para asegurar la corrección de la solución.

Actividades realizadas:

Actividad	Descripción
Desarrollo de Central de Servicios	Implementación de componentes de la central: • Base de datos • Componentes de acceso a datos • Entidades • Lógica de negocio • Servicios web • Pruebas unitarias
Desarrollo de aplicación de escritorio	Implementación de los siguientes componentes: • Interfaz específica • Componentes de conexión con central • Pruebas unitarias • Pruebas de funcionalidades
Desarrollo de aplicación dispositivo móvil	Implementación de los siguientes componentes: • Interfaz específica • Componentes de conexión con central • Pruebas unitarias • Pruebas de funcionalidades
Desarrollo de aplicación Web	Implementación de los siguientes componentes: • Interfaz específica • Componentes de conexión con central • Pruebas unitarias • Pruebas de funcionalidades • Validaciones de estándares
Preparación del paquete entregable	Confección del documento de instrucciones de instalación, creación del paquete software.
Envío de la PEC3	Entrega de la tercera PEC del trabajo.

1.4.2.5. Finalización

En un proyecto de software esta etapa no existe como tal, en su lugar se incluyen las fases de implantación, puesta en marcha, etc. Sin embargo, en este caso, la última etapa ha sido utilizada para generar la documentación final del trabajo de fin de carrera.

Se incluyen las siguientes actividades:

Actividad	Descripción
Redacción memoria final	Recopilación de documentos y confección de la memoria final
Creación presentación virtual	Creación de la presentación virtual
Entrega final	Entrega de documentación

1.4.3. Temporalización del proyecto

El siguiente calendario muestra la temporalización inicial de las tareas que componen el proyecto.

Análisis previo y planificación	**13 días**	**01/03/2007**	**13/03/2007**
Selección del proyecto	1 día	01/03/2007	01/03/2007
Preparación del proyecto	3 días	02/03/2007	04/03/2007
Definición del proyecto	4 días	05/03/2007	08/03/2007
Planificación del proyecto	1 día	09/03/2007	09/03/2007
Creación del documento	4 días	10/03/2007	13/03/2007
Entrega PEC1 (13/03/2007)	Hito	13/03/2007	
Análisis de Requisitos	**6 días**	**14/03/2007**	**19/03/2007**
Especificación requisitos funcionales	5 días	14/03/2007	18/03/2007
Especificación requisitos no funcionales	1 día	19/03/2007	19/03/2007
Diseño	**20 días**	**20/03/2007**	**09/04/2007**
Diseño arquitectónico	2 días	20/03/2007	21/03/2007
Diseño de clases	2 días	22/03/2007	23/03/2007
Diseño de la persistencia	2 días	24/03/2007	25/03/2007
Estudio de alternativas tecnológicas	13 días	22/03/2007	03/04/2007
Preparación del documento	5 días	04/04/2007	08/04/2007
Entrega PEC2 (9/04/2007)	Hito	09/04/2007	
Implementación y pruebas	**49 días**	**09/04/2007**	**28/05/2007**
Desarrollo de Central de Servicios	12 días	09/04/2007	20/04/2007
Desarrollo de aplicación de escritorio	12 días	21/04/2007	02/05/2007
Desarrollo de aplicación dispositivo móvil	10 días	03/05/2007	12/05/2007
Desarrollo de aplicación Web	12 días	13/05/2007	24/05/2007
Preparación del entregable e instrucciones	3 días	25/05/2007	27/05/2007
Entrega PEC3 (28/05/2007)	Hito	28/05/2007	
Finalización	**15 días**	**28/05/2007**	**11/06/2007**
Redacción memoria final	8 días	28/05/2007	04/06/2007
Creación presentación virtual	7 días	05/06/2007	11/06/2007
Entrega final (11/06/2007)	Hito	11/06/2007	11/06/2007

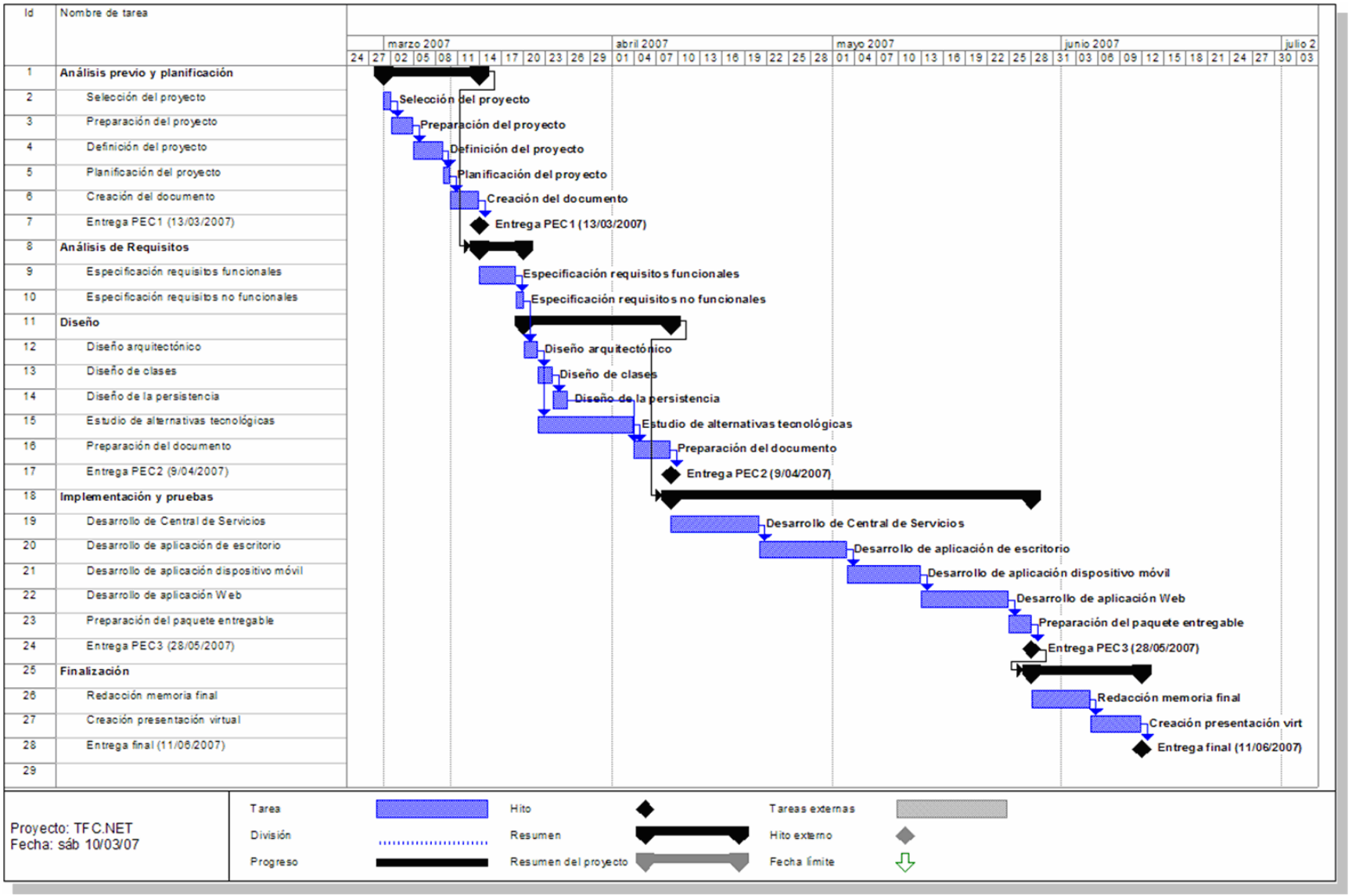


Ilustración 3. Gantt de la planificación del proyecto

1.5. Herramientas utilizadas

Para la realización del Trabajo de Fin de Carrera se han utilizado las siguientes herramientas:

- Visual Studio 2005, el entorno integrado de desarrollo (IDE) de Microsoft para .NET.
- Microsoft Expression Blend, herramienta de diseño de interfaces WPF.
- SQL Server Management Studio, para el diseño de las bases de datos.
- Adobe Photoshop CS, para la creación de prototipos y elementos gráficos del sistema Web y determinadas ilustraciones de la documentación.
- Internet Explorer y Firefox, con extensiones de desarrollador, para la composición y testeo del sistema Web.
- Microsoft Word, para la redacción de documentos.
- Microsoft Visio, para el diseño de diagramas de la documentación.
- Microsoft Project, para la planificación del proyecto.
- Microsoft PowerPoint, para la creación de la presentación del proyecto.

1.6. Productos obtenidos

Durante la realización del presente trabajo se han generado los siguientes entregables:

- Documento del Plan de Trabajo.
- Documento de Análisis y Diseño del sistema.
- Sistema Catalogo.NET, compuesto por los siguientes proyectos:
    - *CatalogoCentral*, donde se implementan de la lógica de negocio de los servicios y la persistencia de entidades.
    - *ServicioCatalogo*, que implementa la como fachada de las funcionalidades ofrecidas por CatalogoCentral utilizando Servicios Web XML.
    - *CatalogoWPF*, la herramienta de gestión del catálogo diseñada como aplicación de escritorio basada en Windows Presentation Foundation.
    - *CatalogoPDA*, el cliente para dispositivos móviles.
    - CatalogoWEB, el sistema web de acceso al catálogo.
- Manual de instalación del sistema.
- Memoria final del proyecto, este documento.
- Presentación virtual.

1.7. Estructura de este documento

El resto de esta memoria recoge los aspectos más destacables de las tareas realizadas, comenzando por los requisitos iniciales, que describen el escenario sobre el que se ha trabajado.

Acto seguido se introducirá el resultado de la fase de análisis, con un detalle de los requisitos funcionales expresados en forma de casos de uso, y la descripción de aspectos y restricciones no funcionales, así como un modelo de entidades del dominio. A continuación se detallará el diseño del sistema desde el punto de vista arquitectónico, las tecnologías elegidas, el diagrama estático de diseño, modelos dinámicos, y algunos prototipos de las distintas interfaces de usuario.

Después se entrará a describir la fase de implementación, donde se comentarán las principales particularidades y decisiones que han sido tomadas durante esta etapa del proyecto para cada uno de los componentes creados y se describirán los aspectos más destacados.

El último capítulo se centrará en las conclusiones finales a las que se han llegado a lo largo de la realización del proyecto, y a continuación se podrá encontrar información adicional como el glosario y las fuentes utilizadas.

2. Requisitos iniciales

Los siguientes epígrafes describen el escenario ficticio sobre el que se va a desarrollar el presente Trabajo de Fin de Carrera, así como las principales necesidades establecidas desde el inicio.

2.1. Escenario de partida

PCEXPRESS, S.L. es una compañía de ámbito regional dedicada a la comercialización de productos informáticos al por mayor.

En la actualidad la empresa edita catálogos comerciales en papel de forma trimestral. Estas publicaciones relacionan, ordenados según familias y subfamilias, todos los productos que comercializan, incluyendo su nombre, referencia, descripción y fotografía. El importe no aparece reflejado de forma directa; en su lugar, se incluye un código que más adelante será transformado por los comerciales en el precio del producto, atendiendo al cliente, stock u otros factores.

El catálogo es utilizado en primer lugar por el personal comercial, que lo utiliza para consultar referencias y características durante sus contactos con los clientes, y en segundo lugar por los clientes, que los usan como referencia para realizar pedidos o consultar productos disponibles.

La existencia de estos catálogos en soporte papel presenta varios inconvenientes:

- El catálogo está obsoleto en el mismo momento en que es impreso, pues casi semanalmente se producen entradas, salidas o modificaciones en la cartera de productos comercializados.
- La distribución física de los catálogos es compleja, dado el ámbito regional de la empresa.
- El acceso a la información no es flexible, la categorización a veces dificulta la localización de productos y, en cualquier caso, es poco eficiente.
- Existe un alto coste asociado a la producción, edición y distribución.

Por estos motivos, se estima conveniente la creación de un sistema de información que facilite la Gestión del Catálogo de Productos, con funcionalidades de alto valor añadido asociadas a utilización de las nuevas tecnologías de información y comunicaciones.

De esta forma se obtienen los siguientes beneficios inmediatos:

- La información del catálogo siempre está actualizada y es más fácil de consultar.
- No es necesario fabricar ni distribuir los catálogos físicos, con el consiguiente ahorro de costes y mejora en eficiencia que esto conlleva.
- Mejora de la imagen de la empresa hacia sus clientes por la utilización de la tecnología como medio para incrementar la calidad del servicio y los procedimientos internos.

Es importante para el proyecto destacar la movilidad y diversidad de escenarios de trabajo en los que estas tareas de gestión deben llevarse a cabo. La mayoría de los accesos se producirán desde la oficina, donde se trabaja mayoritariamente con Windows, así como desde el almacén, donde el personal dispone de PDA's. En ambos casos se cuenta con una red local Ethernet con puntos de acceso Wi-fi convenientemente distribuidos por las instalaciones.

En determinadas ocasiones, los clientes o el personal comercial desplazado necesita acceder al catálogo de productos de la compañía, por lo que se estima absolutamente necesario disponer de un acceso vía Web al catálogo para consultar características de productos.

2.2. Usuarios a considerar

Los principales tipos de usuario que deberán ser tenidos en cuenta en el sistema son los siguientes:

- Usuario Anónimo, que sólo tendrá acceso a las funcionalidades de entrada al sistema.
- Usuario Comercial, que podrá consultar el catálogo de productos y acceder a sus descripciones completas.
- Usuario Cliente, que, como en el caso anterior, deberá tener acceso al catálogo de productos, aunque no podrá ver toda la información. Estos usuarios usarán el sistema exclusivamente a través de Internet.
- Usuario Gestor, tendrá acceso a las herramientas de mantenimiento de productos e información relacionada, además de a las funcionalidades permitidas al Usuario de Consultas.
- Usuario Administrador, que además de las funciones descritas con anterioridad, podrá llevar a cabo tareas relacionadas con la gestión de usuarios del sistema.

2.3. Requisitos funcionales

Como se ha comentado anteriormente, el principal objetivo del sistema de información es permitir la gestión de un Catálogo de Productos por parte de un colectivo de usuarios, aportando a cada uno de ellos, en función de su tipología, las herramientas necesarias para llevar a cabo sus tareas.

Los requisitos se han agrupado en dos bloques principales:

- Funcionalidades de seguridad, donde se recogen las necesidades relativas al mantenimiento de usuarios y control de acceso al sistema.
- Funcionalidades del Catálogo, donde se reflejan los requisitos relativos al tratamiento de la información relativa a los productos.

A continuación se desarrollan cada uno de ellos.

2.3.1. Funcionalidades de Seguridad

Las siguientes funcionalidades se encuentran dirigidas a gestionar los aspectos relacionados con la seguridad que habrán de ser contemplados por el sistema.

Iniciar sesión
Permitirá a los usuarios anónimos identificarse ante el sistema y, si procede, acceder al resto de las funcionalidades ofrecidas por éste en función de su tipología. La identificación se realizará mediante la aportación de un nombre de usuario y una contraseña.

Finalizar sesión
Finaliza la sesión del usuario actual, posibilitando la entrada de un nuevo usuario.

Cambio de clave
Permite a un usuario, previamente identificado en el sistema, modificar su propia contraseña de acceso.

Alta de usuarios
Permitirá a los Administradores crear nuevos usuarios en el sistema, permitiendo, además, especificar su tipología (Consultas, Gestor ó Administrador).

Baja de usuarios
Deberá posibilitar la eliminación de un usuario del sistema por parte de los Administradores. Para evitar dejar bloqueado el sistema y prevenir situaciones anómalas, no deberá ser posible que un Administrador se elimine a sí mismo.

Modificación de usuarios
Esta funcionalidad permitirá a los Administradores modificar los datos de los usuarios registrados en el sistema.

Consulta de usuarios
Deberá permitir a los Administradores consultar los usuarios registrados en el sistema.

2.3.2. Funcionalidades del Catálogo

En este bloque encontraremos las funcionalidades a crear destinadas a dar cobertura a las necesidades de tratamiento de información de productos del catálogo. Sin embargo, según sus objetivos concretos, distinguiremos tres paquetes de funcionalidades distintos:

- *Productos*, con las funcionalidades de mantenimiento y acceso al catálogo de productos.
- *Familias*, que contendrá las herramientas necesarias para la gestión de familias en las que se podrán agrupar los productos.
- *Subfamilias*, que, como en el caso anterior, incluirá las funcionalidades requeridas para la gestión de subfamilias en las que agrupar los productos.

2.3.2.1. Productos

Alta de productos
Permitirá a los usuarios Gestores introducir un nuevo producto en el sistema, para lo que deberán rellenar una ficha con la información oportuna.

Baja de productos
A través de esta funcionalidad, se permitirá a los Gestores eliminar productos que no vayan a ser utilizados.

Modificación de productos
Esta funcionalidad permitirá a los usuarios Gestores del catálogo modificar información sobre los productos almacenados.

Búsqueda de productos
Permitirá a todos los usuarios del sistema realizar una búsqueda en el catálogo mediante la introducción de criterios. El resultado será un listado sobre el que el usuario podrá seleccionar un elemento para ampliar información.

Visualización de productos
Permitirá visualizar los datos de un producto, aunque el alcance de la información a mostrar vendrá dado por el tipo de usuario que esté ejecutando la función.

2.3.2.2. Familias

Alta de familias de productos
Permitirá a los usuarios Gestores introducir una nueva familia de productos en el sistema.

Baja de familias de productos
Esta funcionalidad deberá permitir eliminar una familia de productos del sistema. Sin embargo, si la familia a eliminar tiene subfamilias con productos asociados, debe ser posible indicar a qué familia deben ser movidos, o bien si se desea eliminar las subfamilias y productos vinculados a ella.

Modificación de familias de productos
Deberá permitir modificar los datos de una familia de productos.

Consulta de familias
Facilitará la consulta de familias registradas en el sistema.

2.3.2.3. Subfamilias

Alta de subfamilias de productos
Permitirá a los usuarios Gestores introducir una nueva familia de productos en el sistema.

Baja de subfamilias de productos
Esta funcionalidad deberá permitir eliminar una subfamilia de productos del sistema. Habrá que tener en cuenta que si la subfamilia a eliminar tiene productos asociados, debe ser posible indicar a qué otra subfamilia deben ser movidos, o bien si se desea eliminar los productos vinculados a ella.

Modificación de subfamilias de productos
Deberá permitir modificar los datos de una subfamilia de productos.

Consulta de subfamilias
Facilitará la consulta de las subfamilias registradas en el sistema.

2.4. Requisitos no funcionales

2.4.1. Requisitos de interfaz

El sistema deberá contar con los siguientes interfaces especializados:

- Un interfaz específicamente destinado para trabajar en entorno Windows, desde el que puedan realizarse todas las funciones del sistema. Se pretende con esto que los usuarios acostumbrados a trabajar con aplicaciones de escritorio sigan haciéndolo, minimizando el esfuerzo necesario para el aprendizaje.

- Un interfaz diseñado para funcionar en PDA's, con acceso a las funcionalidades de consulta del catálogo, para el personal de almacén.

- Un interfaz Web con acceso a las funcionalidades de consulta tanto para Clientes de la empresa como para personal Comercial desplazado.

2.4.2. Requisitos de seguridad

El sistema, previamente a su utilización, deberá identificar al usuario con objeto de, en primer lugar, comprobar si está autorizado a usar la aplicación, y en segundo, qué tipo de usuario se trata para poder ofrecerle las herramientas apropiadas para la realización de sus tareas.

La autenticación del usuario se realizará mediante la introducción y comprobación de un par usuario/clave que le habrán debido ser inicialmente suministradas, y que el usuario podrá modificar cuando estime oportuno.

2.4.3. Requisitos de información

La información de los productos que deberá considerarse es:

- Referencia del producto, un código único asignado por la empresa al añadirlo a la cartera de elementos comercializados, como “M-11202”.
- Nombre, la denominación del producto, por ejemplo, “Flatron L1730S”
- Descripción, una descripción textual del producto.
- Marca, el nombre del fabricante del componente, por ejemplo, “LG”.
- Familia, la categoría principal a la que se asigna el producto, por ejemplo, “Monitores”.
- Subfamilia, la subcategoría (dentro de la familia) a la que se asigna el producto, por ejemplo “TFT”. Todo producto estará obligatoriamente asignado a una subcategoría.
- Fotografía, imagen del producto en cuestión.
- Código de precio, una clave libre de tres dígitos que usan los comerciales para obtener el precio de los productos (de forma externa al sistema). Un ejemplo podría ser “142”.
- Stock, número de unidades existentes en almacén.

Es importante destacar que los dos últimos campos sólo podrán ser mostrados a personal de la empresa, los Clientes no dispondrán de acceso a esa información.

Del resto de elementos (entidades) no se establecen requisito previo alguno.

3. Análisis del sistema

Esta sección recoge el resultado de la fase de análisis del sistema. Para ello, partiendo de la vista general aportada por los diagramas de casos de uso, se detallará cada uno de ellos, y se formalizarán y concretarán los requisitos no funcionales expresados en el epígrafe anterior.

3.1. Diagramas de casos de uso

El siguiente diagrama recoge una vista global de los actores y casos de uso que describen los requisitos funcionales de la aplicación:

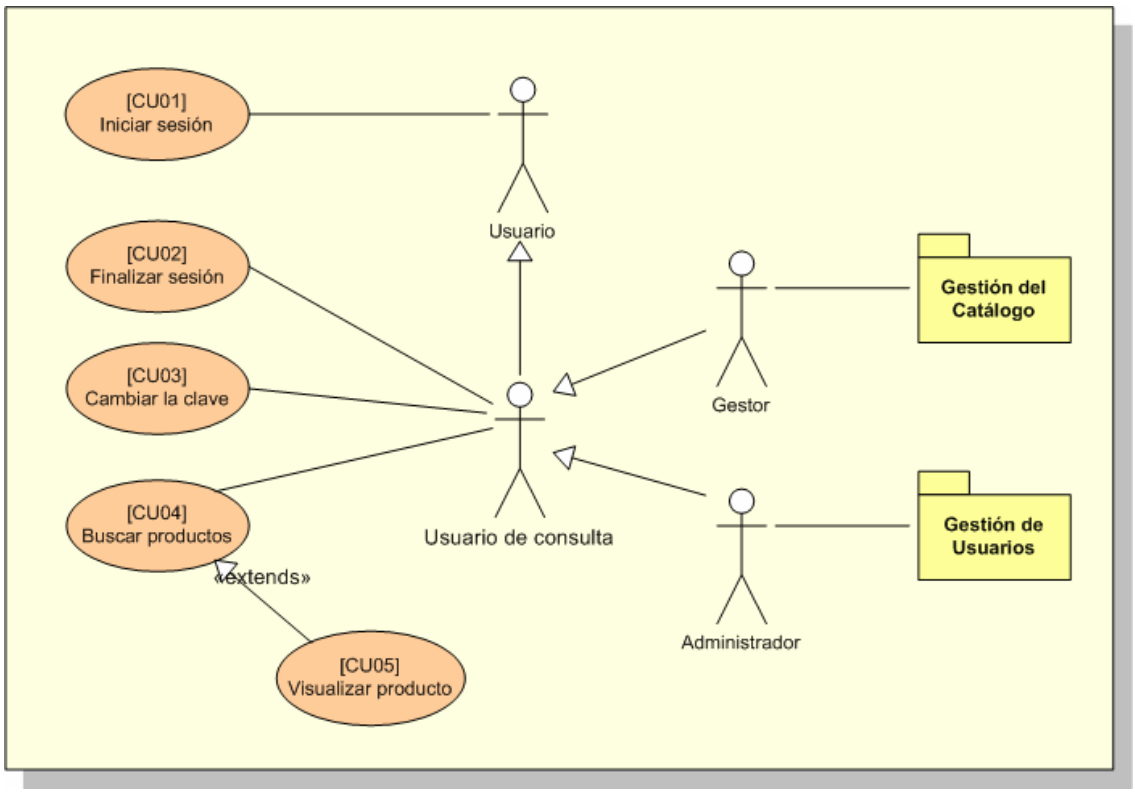


Ilustración 4. Casos de uso

Como se puede observar, el sistema distinguirá cuatro tipos de actores diferentes:

- *Usuario*, que representa a todos los usuarios que a través de cualquiera de los interfaces de la aplicación, pueden interactuar con la misma.
- *Usuario de consulta,* una especialización del primero, en el que se incluyen todos aquellos actores que hayan superado el procedimiento de autenticación exigido por el sistema.
- *Gestor*, un tipo de usuario autenticado cuya misión principal será mantener la información gestionada por el Catálogo de Productos.
- *Administrador*, un tipo de usuario autenticado al que se le permitirá gestionar los usuarios del sistema.

Para facilitar la lectura, en el diagrama anterior se han agrupado en paquetes un conjunto de funcionalidades que afectan al mismo actor y cuyo objetivo es común, por una parte la *Gestión*

del Catálogo, utilizada por el usuario *Gestor*, y por otra la *Gestión de Usuarios*, funcionalidades disponibles para el usuario *Administrador*.

Los siguientes diagramas de casos de uso recogen el contenido de estos paquetes funcionales.

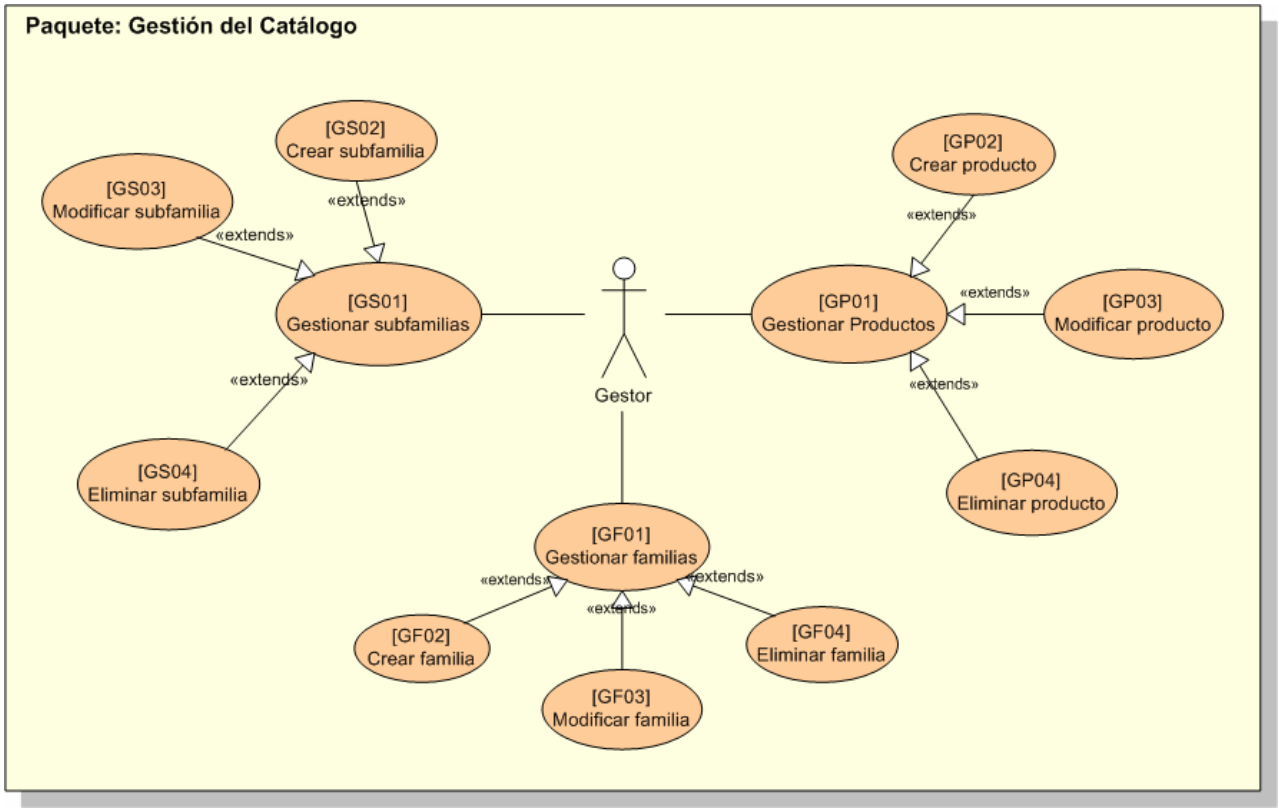


Ilustración 5. Casos de uso: Gestión del catálogo

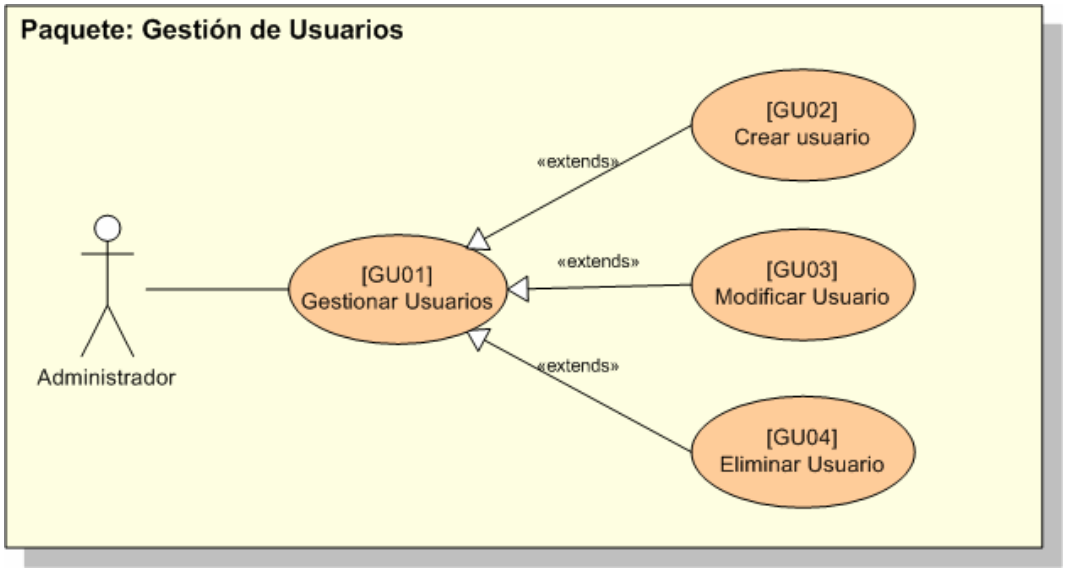


Ilustración 6. Casos de uso: Gestión de usuarios

La siguiente tabla resume los casos de uso contemplados, que serán desarrollados en la sección posterior.

Código	Descripción	Actor(es)
CU01	Iniciar sesión en el sistema	Usuario
CU02	Finalizar sesión actual	Usuario de consulta
CU03	Cambiar la clave del usuario activo	Usuario de consulta
CU04	Visualizar producto	Usuario de consulta
CU05	Buscar productos	Usuario de consulta
GU01	Gestionar usuarios	Administrador
GU02	Crear usuario	Administrador
GU03	Modificar usuario	Administrador
GU04	Eliminar usuario	Administrador
GF01	Gestionar familias	Gestor
GF02	Crear familia	Gestor
GF03	Modificar familia	Gestor
GF04	Eliminar familia	Gestor
GS01	Gestionar subfamilias	Gestor
GS02	Crear subfamilia	Gestor
GS03	Modificar subfamilia	Gestor
Gs04	Eliminar subfamilia	Gestor
GP01	Gestionar Productos	
GP02	Crear producto	Gestor
GP03	Modificar datos de un producto	Gestor
GP04	Eliminar producto del sistema	Gestor

3.2. Descripción textual de casos de uso

3.2.1. CU01-Iniciar sesión

Identificador	CU01
Nombre	Iniciar sesión
Autor	José María Aguilar
Resumen	Este caso de uso muestra cómo un usuario se identifica en el sistema
Actor(es)	Usuario
Precondiciones	No hay ninguna sesión activa
Postcondiciones	El usuario ha iniciado la sesión o bien el actor ha finalizado la aplicación
Flujo normal	1. El caso de uso se inicia cuando el actor accede por primera vez al sistema, o bien cuando finaliza una sesión anterior. 2. El sistema solicita al actor la introducción de su nombre de usuario y contraseña. 3. El usuario introduce esta información y valida el formulario 4. El sistema comprueba si los datos suministrados son correctos. 5. Si la información es correcta, la sesión queda iniciada y se muestra el menú principal de la aplicación, finalizando este

	caso de uso.
Flujos alternativos	• Si los datos de identificación suministrados en el paso 3 no son correctos, se muestra un error y el flujo vuelve al paso 2. • El actor puede decidir finalizar la aplicación pulsando un botón.
Inclusiones	Ninguna
Extensiones	Ninguna

3.2.2. CU02-Finalizar sesión

Identificador	CU02
Nombre	Finalizar sesión
Autor	José María Aguilar
Resumen	Este caso de uso muestra cómo un usuario finaliza su sesión en el sistema.
Actor(es)	Usuario de consulta
Precondiciones	El actor ha iniciado previamente la sesión
Postcondiciones	No hay ninguna sesión activa
Flujo normal	1. El caso de uso se inicia cuando el actor selecciona en el menú la opción correspondiente al cierre de sesión. 2. El sistema muestra un aviso y solicita confirmación del cierre de sesión al actor. 3. El actor confirma la finalización de la sesión. 4. El sistema da por finalizada la sesión actual y finaliza el caso de uso, pasando a ejecutarse CU01 (Iniciar sesión).
Flujos alternativos	• Si el actor, en el paso 3, decide no confirmar el fin de sesión, se finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Ninguna

3.2.3. CU03-Cambiar la clave

Identificador	CU03
Nombre	Cambiar la clave
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un usuario del sistema puede modificar su clave de acceso
Actor(es)	Usuario de consulta
Precondiciones	El actor ha iniciado previamente la sesión
Postcondiciones	El actor ha modificado su clave de acceso, o el proceso ha sido cancelado.
Flujo normal	1. El caso de uso se inicia cuando el actor selecciona en el menú la opción correspondiente al cambio de clave. 2. El sistema muestra un formulario solicitando la clave anterior y la nueva clave por duplicado. 3. El actor introduce la información requerida y valida el formulario. 4. El sistema modifica la clave de acceso del actor.
Flujos alternativos	• Si la clave anterior introducida no coincide con la del usuario activo, se muestra un error y se vuelve al paso 2. • Si al introducir por duplicado la nueva clave ambos valores no coinciden, están en blanco, o son iguales a la anterior, se muestra un error y se vuelve al paso 2. • El actor puede decidir no validar el formulario en el paso 3. Esto finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Ninguna

3.2.4. CU04-Buscar productos

Identificador	CU04
Nombre	Buscar productos
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un usuario del sistema realiza búsquedas de productos en el Catálogo.
Actor(es)	Usuario de consulta
Precondiciones	El actor ha iniciado previamente la sesión
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor acceder al Catálogo de Productos. 2. El sistema muestra un formulario con los posibles criterios de búsqueda (referencia, familia, subfamilia y descripción). 3. El actor introduce la información requerida. 4. El sistema muestra un listado con los productos que cumplen los criterios establecidos. 5. Finaliza el caso de uso.
Flujos alternativos	• El actor puede visualizar la ficha completa de un producto, seleccionándolo e iniciando el caso de uso CU05 (Visualizar producto).
Inclusiones	Ninguna
Extensiones	CU05 (Visualizar producto)

3.2.5. CU05-Visualizar producto

Identificador	CU05
Nombre	Visualizar producto
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un usuario del sistema puede visualizar los datos de un producto.
Actor(es)	Usuario de consulta
Precondiciones	Ninguna
Postcondiciones	Ninguno
Flujo normal	1. El caso de uso se inicia cuando un usuario selecciona un producto (CU04) y pulsa la opción de visualizar el detalle del mismo. 2. El sistema muestra la ficha completa del producto, aunque el stock y el código de precio sólo se mostrarán si el actor es un Gestor de la empresa. 3. El actor pulsa el botón volver y finaliza el caso de uso.
Excepciones	• Si se produce algún tipo de problema a la hora de obtener la información del producto (por ejemplo, que haya sido eliminado por otro usuario), se muestra un mensaje de error y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende al caso de uso CU04 (Buscar productos)

3.2.6. GU01-Gestionar usuarios

Identificador	GU01
Nombre	Gestionar usuarios
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Administrador puede gestionar los usuarios del sistema.
Actor(es)	Administrador
Precondiciones	Ninguna
Postcondiciones	Ninguno
Flujo normal	1. El caso de uso comienza cuando el administrador pulsa en el menú principal la opción que le permite acceder a la Gestión de Usuarios. 2. El sistema muestra la lista de usuarios almacenados en el sistema, y una serie de botones que permiten manipular la información. 3. Cuando el actor acaba de realizar las tareas de gestión de los usuarios, pulsa un botón para salir, finalizando el caso de uso.
Flujos alternativos	• El actor puede iniciar los casos de uso GU03 (Modificar usuario) y GU04 (Eliminar usuario) seleccionando un usuario concreto sobre la lista. Una vez finalizados, se vuelve al paso 2. • El actor también puede iniciar el caso de uso GU02 (Crear usuario) pulsando sobre un botón. Una vez finalizado, se vuelve al paso 2.
Inclusiones	Ninguna
Extensiones	GU02 (Crear usuario) GU03 (Modificar usuario) GU04 (Eliminar usuario)

3.2.7. GU01-Crear usuario

Identificador	GU01
Nombre	Crear usuario
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un administrador crea un nuevo usuario
Actor(es)	Administrador
Precondiciones	Ninguno
Postcondiciones	Se ha creado un nuevo usuario, o bien se ha cancelado el proceso
Flujo normal	1. El caso de uso comienza cuando el administrador pulsa en el menú la opción dispuesta para crear usuarios. 2. El sistema muestra la ficha vacía del usuario. 3. El actor cumplimenta la información solicitada y acepta el formulario. 4. El sistema almacena los datos del nuevo usuario y finaliza el caso de uso.
Flujos alternativos	• Si el nombre de usuario ya existe en el sistema, se muestra un mensaje de error y se vuelve al paso 3.
Excepciones	• Si ocurre algún error durante la grabación, el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GU01 (Gestionar usuarios)

3.2.8. GU03-Modificar usuario

Identificador	GU03
Nombre	Modificar usuario
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Administrador modifica los datos de un usuario almacenado en el sistema.
Actor(es)	Administrador
Precondiciones	Hay un usuario seleccionado
Postcondiciones	Los datos del usuario han sido modificados, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando GU01 (Gestionar usuarios), selecciona a uno de ellos y pulsa el botón correspondiente a la modificación de registros. 2. El sistema muestra un formulario con los datos del usuario seleccionado. El nombre de usuario (username) aparecerá deshabilitado para prevenir errores en la modificación. 3. El actor modifica la información y acepta el formulario. 4. El sistema almacena los cambios y finaliza el caso de uso.
Flujos alternativos	• El actor puede pulsar Cancelar en el paso 3, finalizando el caso de uso.
Excepciones	• Si ocurre algún error en la modificación (por ejemplo, que el usuario haya sido eliminado por otra persona), el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GU01 (Gestionar usuarios)

3.2.9. GU04-Eliminar usuario

Identificador	GU04
Nombre	Eliminar usuario
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Administrador elimina un usuario almacenado en el sistema.
Actor(es)	Administrador
Precondiciones	Hay un usuario seleccionado
Postcondiciones	El usuario ha sido eliminado, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando GU01 (Gestionar usuarios), selecciona a uno de ellos y pulsa el botón correspondiente a la eliminación de registros. 2. El sistema solicita confirmación para la eliminación del usuario. 3. El actor confirma el borrado. 4. El sistema elimina el usuario y finaliza el caso de uso.
Flujos alternativos	• El actor puede pulsar Cancelar en el paso 3, finalizando el caso de uso.
Excepciones	• Si ocurre algún error en la eliminación (por ejemplo, que el usuario haya sido eliminado previamente por otro actor), el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GU01 (Gestionar usuarios)

3.2.10. GF01-Gestionar familias

Identificador	GF01
Nombre	Gestionar familias
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Administrador puede gestionar las familias de productos del catálogo.
Actor(es)	Gestor
Precondiciones	Ninguna
Postcondiciones	Ninguno
Flujo normal	1. El caso de uso comienza cuando el administrador pulsa en el menú principal la opción que le permite acceder a la Gestión de Familias. 2. El sistema muestra la lista con las Familias almacenadas en el sistema, y una serie de botones que permiten manipular la información. 3. Cuando el actor acaba de realizar las tareas de gestión de familias, pulsa un botón para salir, finalizando el caso de uso.
Flujos alternativos	• El actor puede iniciar los casos de uso GF03 (Modificar familia) y GF04 (Eliminar familia) seleccionando una familia concreta sobre la lista. Una vez finalizados, se vuelve al paso 2. • El actor también puede iniciar el caso de uso GF01 (Crear familia) pulsando sobre un botón. Una vez finalizado, se vuelve al paso 2.
Inclusiones	Ninguna
Extensiones	GF02 (Crear familia) GF03 (Modificar familia) GF04 (Eliminar familia)

3.2.11. GF02-Crear familia

Identificador	CU21
Nombre	Crear familia
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Gestor crea una nueva familia de productos.
Actor(es)	Gestor
Precondiciones	Ninguno
Postcondiciones	La familia ha sido almacenada en el sistema, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando el caso de uso GF01 (Gestionar familias) pulsa el botón correspondiente a crear una nueva familia de productos. 2. El sistema muestra el formulario de datos de la nueva familia de productos en blanco, consistente en un código y una descripción. 3. El actor cumplimenta la información solicitada y acepta el formulario. 4. El sistema almacena los datos de la nueva familia y finaliza el caso de uso.
Flujos alternativos	• Si el código de la familia ya existe en el sistema, se muestra un mensaje de error y se vuelve al paso 3.
Excepciones	• Si ocurre algún error durante la grabación, el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GF01 (Gestionar familias)

3.2.12. GF03-Modificar familia

Identificador	GF03
Nombre	Modificar usuario
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Administrador modifica los datos de un usuario almacenado en el sistema.
Actor(es)	Administrador
Precondiciones	Hay una familia seleccionada
Postcondiciones	Los datos del usuario han sido modificados, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando GF01 (Gestionar familias), selecciona una de ellas y pulsa el botón correspondiente a la modificación de registros. 2. El sistema muestra un formulario con los datos de la familia seleccionada. El código de la familia aparecerá deshabilitado para prevenir errores en la modificación. 3. El actor modifica la información y acepta el formulario. 4. El sistema almacena los cambios y finaliza el caso de uso.
Flujos alternativos	• El actor puede pulsar Cancelar en el paso 3, finalizando el caso de uso.
Excepciones	• Si ocurre algún error en la modificación (por ejemplo, que la familia haya sido eliminada por otra persona), el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GF01 (Gestionar familias)

3.2.13. GF04-Eliminar familia

Identificador	GF04
Nombre	Eliminar familia
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Gestor elimina una familia de productos.
Actor(es)	Gestor
Precondiciones	Hay una familia seleccionada
Postcondiciones	La familia ha sido eliminada, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando GF01 (Gestionar familias), selecciona una de ellas y pulsa el botón correspondiente a la eliminación de registros. 2. El sistema muestra al actor un formulario de confirmación, donde además éste puede decidir a qué nueva familia pasarán las subfamilias asignadas a la familia que va a ser eliminada. 3. El actor introduce esta información y pulsa aceptar. 4. Si se seleccionó una nueva familia, el sistema mueve las subfamilias que tenía asignadas la antigua familia a la nueva, se elimina la familia solicitada, y finaliza el caso de uso.
Flujos alternativos	• El actor puede pulsar Cancelar en el paso 3, finalizando el caso de uso.
Excepciones	• Si ocurre algún error en la eliminación (por ejemplo, que la familia haya sido eliminada previamente por otro actor), el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GF01 (Gestionar familias)

3.2.14. GS01-Gestionar subfamilias

Identificador	GS01
Nombre	Gestionar subfamilias
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Administrador puede gestionar las subfamilias de productos del catálogo.
Actor(es)	Gestor
Precondiciones	Ninguna
Postcondiciones	Ninguno
Flujo normal	1. El caso de uso comienza cuando el administrador pulsa en el menú principal la opción que le permite acceder a la Gestión de Subfamilias. 2. El sistema muestra la lista con las Subfamilias almacenadas en el sistema, y una serie de botones que permiten manipular la información. 3. El actor puede, seleccionando de un desplegable, ceñir la lista de subfamilias mostradas a las que pertenezcan a la familia seleccionada. 4. Cuando el actor acaba de realizar las tareas de gestión de subfamilias, pulsa un botón para salir, finalizando el caso de uso.
Flujos alternativos	• El actor puede iniciar los casos de uso GS03 (Modificar subfamilia) y GS04 (Eliminar subfamilia) seleccionando previamente una subfamilia concreta sobre la lista. Una vez finalizados, se vuelve al paso 2. • El actor también puede iniciar el caso de uso GS01 (Crear subfamilia) pulsando sobre un botón. Una vez finalizado, se vuelve al paso 2.
Inclusiones	Ninguna
Extensiones	GS02 (Crear subfamilia) GS03 (Modificar subfamilia) GS04 (Eliminar subfamilia)

3.2.15. GS02-Crear subfamilia

Identificador	GS02
Nombre	Crear subfamilia
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Gestor crea una nueva subfamilia de productos.
Actor(es)	Gestor
Precondiciones	Ninguna
Postcondiciones	La subfamilia ha sido almacenada en el sistema, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando el caso de uso GS01 (Gestionar subfamilias) pulsa el botón correspondiente a crear una nueva subfamilia de productos. 2. El sistema muestra el formulario de datos de la nueva subfamilia de productos en blanco, consistente en el código, la familia a la que pertenece, y una descripción. 3. El actor cumplimenta la información solicitada y acepta el formulario. 4. El sistema almacena los datos de la nueva familia y finaliza el caso de uso.
Flujos alternativos	• Si el código de la subfamilia ya existe en el sistema, se muestra un mensaje de error y se vuelve al paso 3.
Excepciones	• Si ocurre algún error durante la grabación, el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GS01 (Gestionar subfamilias)

3.2.16. GS03-Modificar subfamilia

Identificador	GS03
Nombre	Modificar subfamilia
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Gestor modifica los datos de una subfamilia de productos.
Actor(es)	Gestor
Precondiciones	Hay una subfamilia seleccionada
Postcondiciones	Los datos de la subfamilia han sido modificados, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando GS01 (Gestionar subfamilias), selecciona una de ellas y pulsa el botón correspondiente a la modificación de registros. 2. El sistema muestra un formulario con los datos de la subfamilia seleccionada. El código de la subfamilia aparecerá deshabilitado para prevenir errores en la modificación. 3. El actor modifica la información y acepta el formulario. 4. El sistema almacena los cambios y finaliza el caso de uso.
Flujos alternativos	• El actor puede pulsar Cancelar en el paso 3, finalizando el caso de uso.
Excepciones	• Si ocurre algún error en la modificación (por ejemplo, que la subfamilia haya sido eliminada previamente por otro usuario), el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GS01 (Gestionar subfamilias)

3.2.17. GS04-Eliminar subfamilia

Identificador	GS04
Nombre	Eliminar subfamilia
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Gestor elimina una subfamilia de productos.
Actor(es)	Gestor
Precondiciones	Hay una subfamilia seleccionada
Postcondiciones	La subfamilia ha sido eliminada, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando GS01 (Gestionar subfamilias), selecciona una de ellas y pulsa el botón correspondiente a la eliminación de registros. 2. El sistema solicita confirmación de la eliminación, a la vez que permite indicar la subfamilia a la que pasarán los productos que ésta tenga asignados, aunque este dato es opcional. 3. El usuario introduce esta información y pulsa aceptar. 4. Si procede, el sistema mueve los productos que tenía asignadas la antigua subfamilia a la nueva. 5. El sistema elimina la subfamilia solicitada con todos los productos que tenga asociada, y finaliza el caso de uso.
Flujos alternativos	• El actor puede pulsar Cancelar en el paso 3, finalizando el caso de uso.
Excepciones	• Si ocurre algún error en la eliminación (por ejemplo, que la subfamilia haya sido eliminada previamente por otro actor), el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GS01 (Gestionar subfamilias)

3.2.18. GP01-Gestionar productos

Identificador	GP01
Nombre	Gestionar productos
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Administrador puede gestionar los productos del catálogo.
Actor(es)	Gestor
Precondiciones	Ninguna
Postcondiciones	Ninguno
Flujo normal	1. El caso de uso comienza cuando el administrador pulsa en el menú principal la opción que le permite acceder a la Gestión de Productos. 2. El sistema muestra la lista con los productos almacenados en el sistema, y una serie de botones que permiten manipular la información. 3. El actor puede, ceñir la lista de productos mostrados a los que cumplan determinados criterios de búsqueda. 4. Cuando el actor acaba de realizar las tareas de gestión de productos, pulsa un botón para salir, finalizando el caso de uso.
Flujos alternativos	• El actor puede iniciar los casos de uso GP03 (Modificar producto) y GP04 (Eliminar producto) seleccionando previamente un producto concreto sobre la lista. Una vez finalizados, se vuelve al paso 2. • El actor también puede iniciar el caso de uso GP01 (Crear producto) pulsando sobre un botón. Una vez finalizado, se vuelve al paso 2.
Inclusiones	Ninguna
Extensiones	GP02 (Crear producto) GP03 (Modificar producto) GP04 (Eliminar producto)

3.2.19. GP02-Crear producto

Identificador	GP02
Nombre	Crear producto
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Gestor añade un nuevo producto al Catálogo.
Actor(es)	Gestor
Precondiciones	Ninguna
Postcondiciones	El producto ha sido almacenado en el sistema, o el proceso ha sido cancelado.
Flujo normal	1. El caso de uso comienza cuando el actor, ejecutando el caso de uso GP01 (Gestionar productos) selecciona la opción de creación de un nuevo producto. 2. El sistema muestra el formulario de datos del producto en blanco. 3. El actor cumplimenta la información solicitada y acepta el formulario. 4. El sistema almacena los datos del nuevo producto y finaliza el caso de uso.
Flujos alternativos	• Si la referencia del producto ya existe en el sistema, se muestra un mensaje de error y se vuelve al paso 3.
Excepciones	• Si ocurre algún error durante la grabación, el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GP01 (Gestionar productos)

3.2.20. GP03-Modificar producto

Identificador	GP03
Nombre	Modificar producto
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Gestor modifica los datos de un producto del catálogo
Actor(es)	Gestor
Precondiciones	Hay un producto seleccionado
Postcondiciones	Los datos del producto han sido modificados, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando GP01 (Gestionar productos), selecciona uno de ellos y pulsa el botón correspondiente a la modificación de registros. 2. El sistema muestra un formulario con los datos del producto seleccionado. La referencia del producto aparecerá deshabilitada para prevenir errores en la modificación. 3. El actor modifica la información y acepta el formulario. 4. El sistema almacena los cambios y finaliza el caso de uso.
Flujos alternativos	• El actor puede pulsar Cancelar en el paso 3, finalizando el caso de uso.
Excepciones	• Si ocurre algún error en la modificación (por ejemplo, que el producto haya sido eliminado previamente por otro usuario), el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GP01 (Gestionar productos)

3.2.21. GP04-Eliminar producto

Identificador	GP04
Nombre	Eliminar producto
Autor	José María Aguilar
Resumen	El caso de uso muestra cómo un Gestor elimina un producto del Catálogo.
Actor(es)	Gestor
Precondiciones	Hay un producto seleccionado
Postcondiciones	El producto ha sido eliminado, o bien se ha cancelado el proceso.
Flujo normal	1. El caso de uso comienza cuando el administrador, ejecutando GP01 (Gestionar productos), selecciona uno de ellos y pulsa el botón correspondiente a la eliminación de registros. 2. El sistema muestra al actor un formulario de confirmación. 3. El actor confirma el borrado. 4. El sistema elimina el producto y finaliza el caso de uso.
Flujos alternativos	• El actor puede pulsar Cancelar en el paso 2, finalizando el caso de uso.
Excepciones	• Si ocurre algún error durante la eliminación (por ejemplo, que el producto haya sido eliminado previamente por otro usuario), el sistema muestra un mensaje y finaliza el caso de uso.
Inclusiones	Ninguna
Extensiones	Extiende a GP01 (Gestionar productos)

3.3. Requisitos no funcionales

Estos requisitos son los recogidos en el capítulo 2 (Requisitos iniciales), y más concretamente en su sección 2.4 (Requisitos no funcionales). Sin embargo, tras su análisis han sido identificados aspectos importantes relacionados con el interfaz de usuario, que se incluyen a continuación.

3.3.1. Detalle de requisitos de interfaz

El sistema deberá contar con los siguientes interfaces especializados:

- Un interfaz específicamente destinado para trabajar en entorno Windows, desde el que puedan realizarse todas las funciones del sistema. Se pretende con esto que los usuarios acostumbrados a trabajar con aplicaciones de escritorio sigan haciéndolo, minimizando el esfuerzo necesario para el aprendizaje.
- Un interfaz diseñado para funcionar en PDA's, con acceso a las funcionalidades de consulta del catálogo, para el personal de almacén.
- Un interfaz Web con acceso a las funcionalidades de consulta tanto para Clientes de la empresa como para personal Comercial desplazado.

El siguiente cuadro resume las funcionalidades, actores e interfaces cuyas relaciones deben permitirse por el sistema.

Funcionalidades disponibles		**Para quiénes**	**Desde dónde**		
Código	**Descripción**	**Actor(es)**	**Desktop**	**PDA**	**Web**
CU01	Iniciar sesión en el sistema	Usuario	◆	◆	◆
CU02	Finalizar sesión actual	Usuario de consulta	◆	◆	◆
CU03	Cambiar clave	Usuario de consulta	◆	◆	◆
CU04	Visualizar producto	Usuario de consulta		◆	◆
CU05	Buscar productos	Usuario de consulta		◆	◆
GU01	Gestionar usuarios	Administrador	◆		
GU02	Crear usuario	Administrador	◆		
GU03	Modificar usuario	Administrador	◆		
GU04	Eliminar usuario	Administrador	◆		
GF01	Gestionar familias	Gestor	◆		
GF02	Crear familia	Gestor	◆		
GF03	Modificar familia	Gestor	◆		
GF04	Eliminar familia	Gestor	◆		
GS01	Gestionar subfamilias	Gestor	◆		
GS02	Crear subfamilia	Gestor	◆		
GS03	Modificar subfamilia	Gestor	◆		
GS04	Eliminar subfamilia	Gestor	◆		
GP01	Gestionar productos	Gestor	◆		
GP02	Crear producto	Gestor	◆		
GP03	Modificar producto	Gestor	◆		
CU43	Eliminar producto	Gestor	◆		

3.4. Entidades del dominio

El modelo del dominio representa las principales clases identificadas en el dominio del problema a solucionar, en este caso, el Catálogo de Productos tal y como ha sido descrito.

El modelo del dominio del problema es realmente simple, como se puede apreciar en el siguiente diagrama:

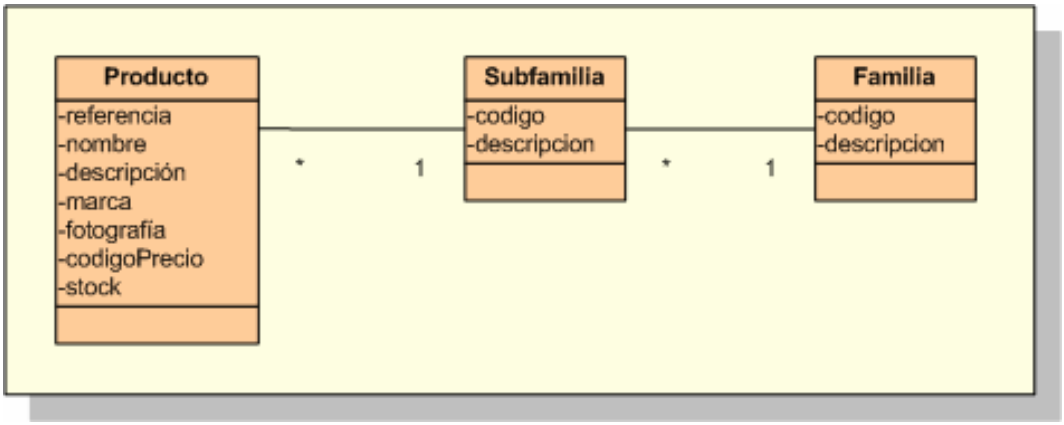


Ilustración 7. Modelo del dominio

Las principales entidades detectadas son:

- *Familia*, que representa al primer nivel en la categorización posible de los productos. Cada objeto de este tipo puede tener asociado un número indeterminado de subfamilias, que representarán un segundo nivel en la jerarquía de categorización.

- *Subfamilia*, que representará al segundo nivel en la categorización de productos; cada subfamilia pertenecerá obligatoriamente a una familia, y podrá contener un número indeterminado de productos.

- *Producto*, cuyos objetos se corresponderán con los productos existentes en el catálogo real, y que incluye atributos propios de éstos.

4. Diseño

4.1. Arquitectura global

La arquitectura global de la solución viene marcada en gran medida por el carácter distribuido del sistema, la adopción del modelo SOA y la multitud de tipos de cliente que se van a permitir.

Para lograr una mayor comprensión del modelo pretendido, se van a detallar distintas vistas del sistema:

- La vista física, donde se reflejarán cómo están distribuidos físicamente los distintos nodos que intervienen en el sistema.
- La vista lógica, que indica los componentes conceptuales desarrollados para dar cobertura a las necesidades propuestas.
- La vista de componentes, que indica los elementos creados, y su correspondencia con los anteriores.

4.1.1. Vista física

El modelo que va a utilizarse en el diseño del sistema es altamente flexible y escalable. De esta forma, será posible realizar un despliegue físico reducido, óptimo cuando la carga de trabajo (número de usuarios, consultas) no es muy elevado, o bien podrá escalarse para dar servicio en escenarios con requisitos de carga más elevados.

Escenario 1: Despliegue compacto
El siguiente diagrama recoge una posible distribución de nodos en un escenario de uso reducido del sistema. Se puede observar que existe un único servidor central que presta servicios de publicación de Web, actúa como servidor de Aplicaciones, y al mismo tiempo, ejerce de almacén de información.

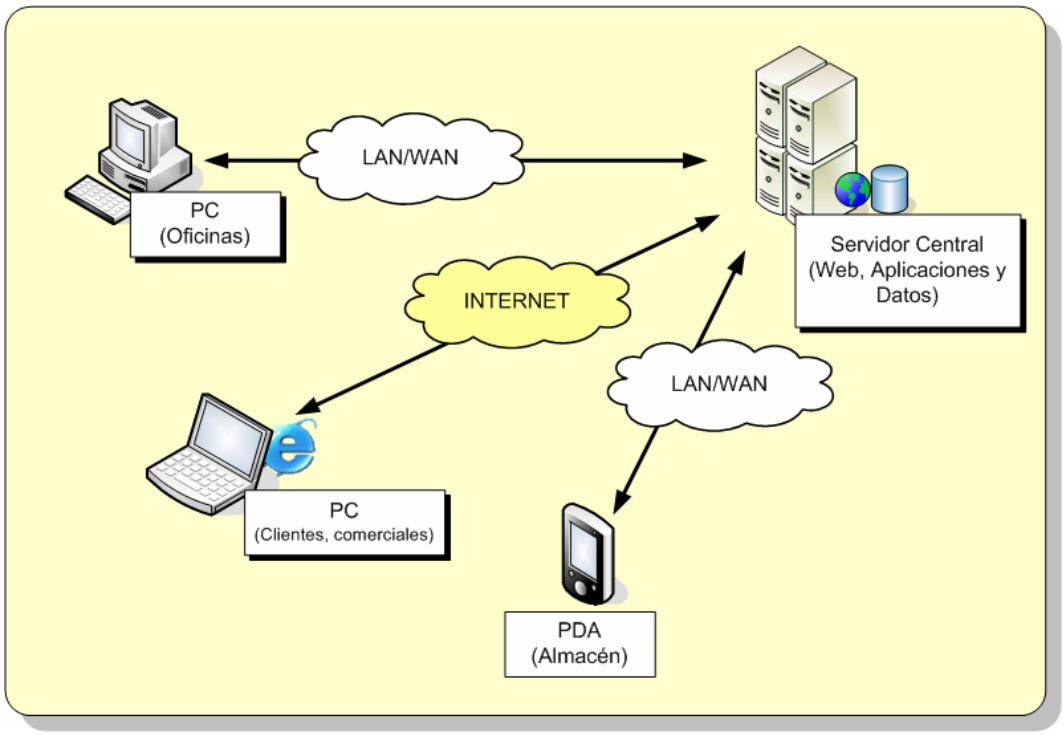


Ilustración 8. Vista física en escenario reducido

Escenario 2: Despliegue extendido

Este escenario muestra la estructura física de la solución en un escenario de extensión máxima, donde cada función principal ha sido asignada a servidores especializados, internos o externos.

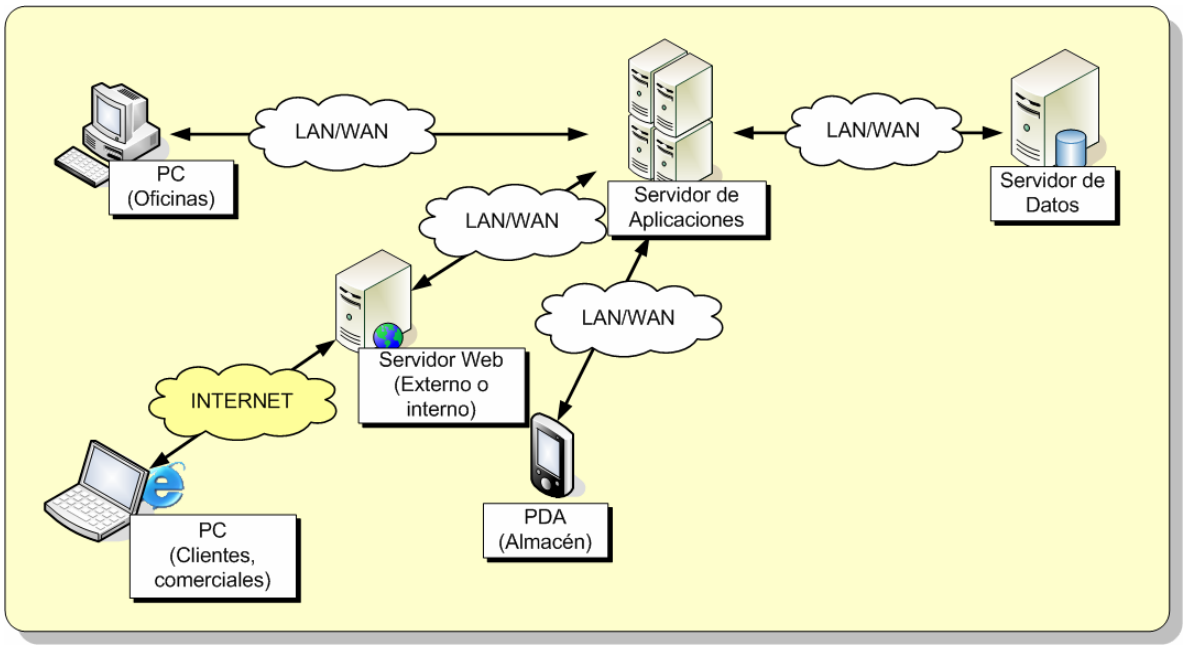


Ilustración 9. Vista física en escenario extendido

Como podemos observar, el servidor web ha sido independizado y será dedicado en exclusiva a esta tarea. Cada vez que sea necesario contactará con el servidor de aplicaciones, donde se encontrarán los servicios web, los componentes de lógica de negocio y los de acceso a datos. Éstos últimos, a su vez, se comunicarán con el servidor de datos dedicado cuando necesiten realizar tareas de almacenamiento u obtención de información.

4.1.2. Vista lógica

Las capas que estructuran la solución son recogidas en el siguiente diagrama, y se describen justo a continuación:

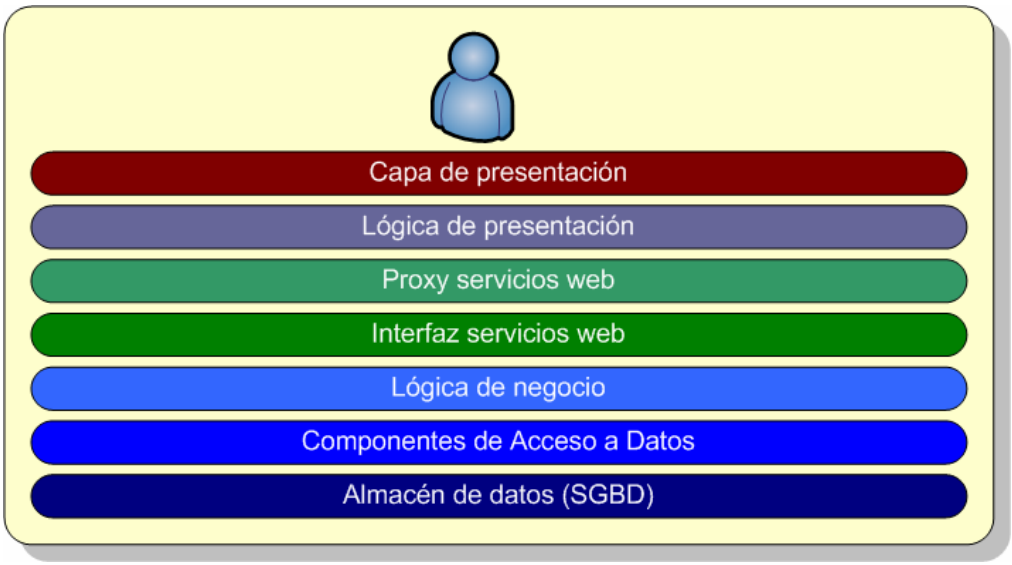


Ilustración 10. Arquitectura lógica

- *Almacén de datos*, en el que recae la responsabilidad de almacenar y ofrecer los mecanismos de recuperación de información utilizada por el sistema.

- *Componentes de Acceso a Datos*, que facilitan el acceso al almacén ocultando las particularidades de éste al resto de capas de la solución, lo que permitiría en última instancia la sustitución el SGBD de forma transparente.

- *Lógica de Negocio*, que utiliza los servicios prestados por los Componentes de Acceso a Datos para implementar las funcionalidades requeridas por la aplicación. Debido a la adopción del paradigma SOA, las operaciones incluidas cumplen las siguientes características:

    - Ser atómicas y transaccionales, es decir, deben efectuar una tarea concreta de principio a fin, y en cualquier caso, dejar al sistema siempre en un estado coherente.
    - Deben realizar tareas de alto nivel, con una gran granularidad.
    - Ser utilizables de forma independientes, es decir, no debe existir ningún requisito en el orden de llamadas entre ellas.
    - Debe existir un acoplamiento muy débil con el cliente.
    - Deben ser reutilizables en distintos contextos y sistemas.

- *Fachada servicios Web,* cuyos componentes actúan de interfaz hacia el exterior de la Lógica de Negocio utilizando Servicios Web XML. En otras palabras, se encargan de recibir los mensajes enviados por los clientes a los servicios, procesarla utilizando los componentes de lógica de negocio, y de devolver la respuesta.

- *Proxy servicios web*, son los conectores que permiten consumir los servicios prestados por la aplicación desde las aplicaciones cliente de forma transparente. Han sido creados utilizando las herramientas automáticas de Visual Studio y retocados para adaptarlos a las necesidades concretas del sistema.

- *Lógica de presentación*, cuyos elementos toman las acciones del usuario y las trasladarán al formato específico del dispositivo (y medio) de acceso utilizado. Será desarrollada a medida para cada tipo de dispositivo.

- *Capa de Presentación*, la más cercana al usuario, cuya misión será únicamente encargarse de la relación con éste, comunicando sus acciones a los elementos de Lógica de Presentación, y mostrando resultados adaptados al dispositivo de visualización.

La distribución de componentes, sin embargo, no es homogénea, pues según el tipo de cliente podrán ser ubicados en unos nodos u otros. Es importante destacar también las posibilidades de comunicación entre ellos, pues determinan la extensibilidad del modelo que fue descrita desde el punto de vista físico.

El siguiente diagrama recoge la misma información que el anterior, pero detallando la ubicación de los elementos en los distintos nodos según el tipo de aplicación cliente, así como señalando las comunicaciones existentes entre ellos. Las flechas bidireccionales que unen dos capas indican que ambas intercambian mensajes (o pueden hacerlo) utilizando infraestructura de comunicaciones, mientras que las que capas adyacentes se comunican de forma directa, a través de instancias dentro del mismo dominio de aplicación.

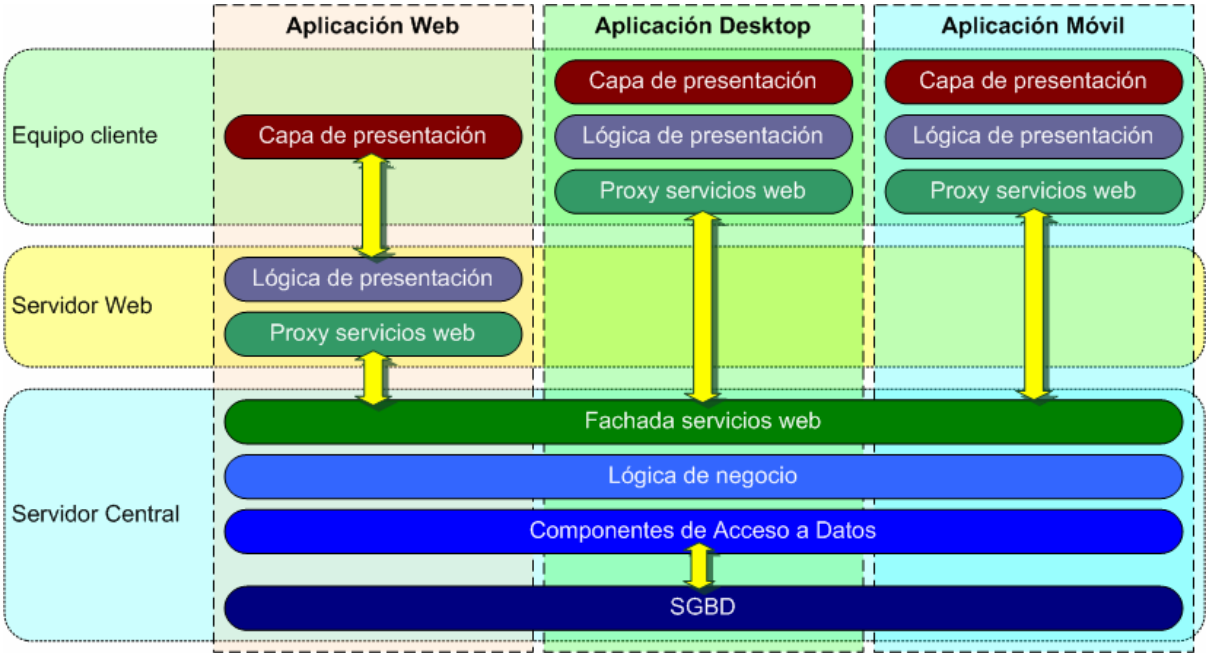


Ilustración 11. Arquitectura lógica detallada

4.1.3. Vista de componentes

El siguiente diagrama resume la estructura de la solución desde el punto de vista de los componentes desarrollados.

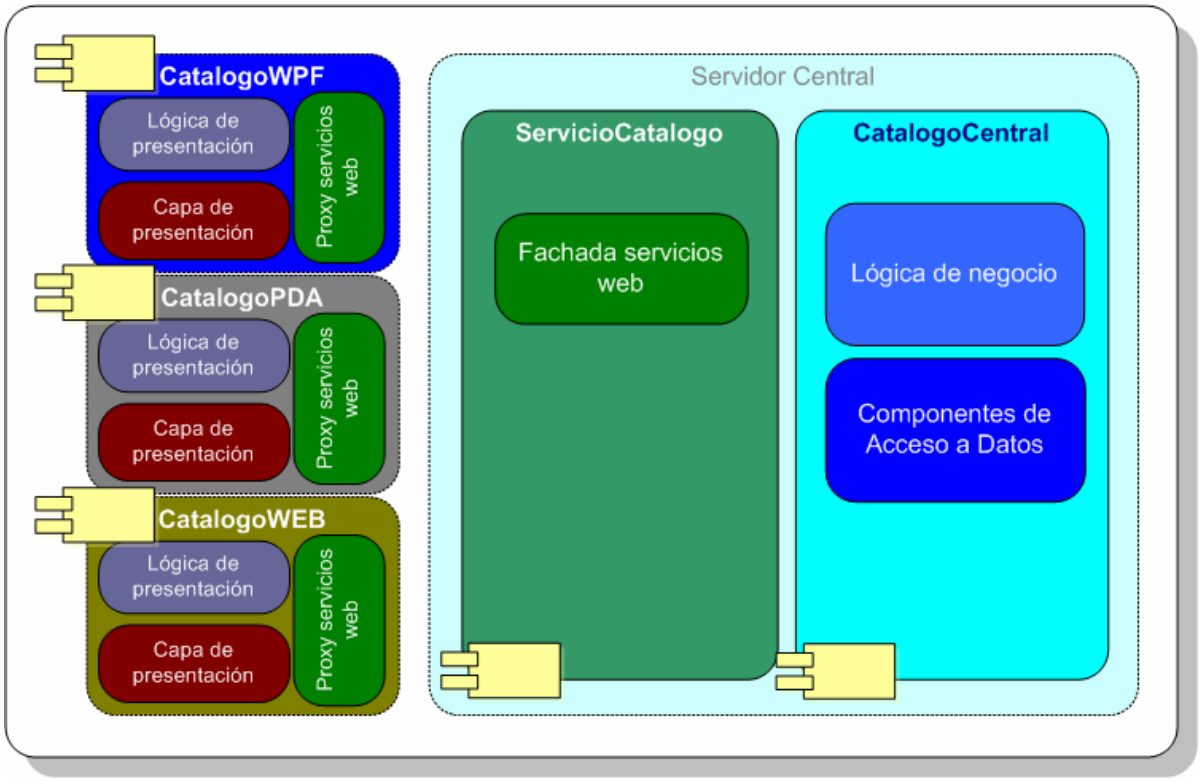


Ilustración 12. Vista de componentes del sistema

Como se puede observar, la solución se estructura en los siguientes componentes:

- *CatalogoCentral*, donde se implementan la lógica de negocio de los servicios y la persistencia de entidades. Se ha separado de la fachada de servicios para permitir su fácil adaptación a otro tipo de conectores distintos del implementado en el proyecto.

- *ServicioCatalogo*, que implementa la como fachada de las funcionalidades ofrecidas por CatalogoCentral utilizando Servicios Web XML.

- *CatalogoWPF*, la herramienta de gestión del catálogo diseñada como aplicación de escritorio basada en Windows Presentation Foundation.

- *CatalogoPDA*, el cliente para dispositivos móviles.

- CatalogoWEB, el sistema web de acceso al catálogo.

4.2. Decisiones tecnológicas

A continuación se justifican las decisiones tomadas para el desarrollo del sistema respecto a la tecnología a aplicar en cada caso.

Lenguaje de desarrollo

Los distintos componentes del sistema están desarrollados utilizando C# 2.0, el lenguaje por excelencia de la plataforma .NET Framework.

SGBD

El SGBD sobre el que se ha desarrollado la solución es **Microsoft SQL Server 2005**, aunque también funciona correctamente con la **edición Express**.

Los motivos de esta elección han sido la contrastada potencia de estos motores, su gran integración en el entorno de desarrollo y la simplicidad de las herramientas de diseño incorporadas.

Acceso a datos

Se ha utilizado **NHibernate** para la implementación del acceso a datos del sistema. Esto, además de aportar las ventajas de un ORM, consigue la independencia total del software respecto al motor de datos subyacente. Para ello, ha sido necesario trabajar con el modelo de mapeo propio, basado en **XML**, y con **HQL.**

Se ha elegido NHibernate por tratarse de un por de Hibernate, uno de los frameworks de persistencia de objetos más utilizados en el mundo Java.

Sistema de comunicaciones entre clientes y servidor

Para esta labor se utilizan **Servicios Web XML**, una de las formas más estandarizadas y potentes de acceso a objetos remotos. Se utilizan, por tanto, las tecnologías habituales relacionadas con ellos, como **SOAP**, **WSDL** ó **XML**.

Se ha elegido esta tecnología por su interoperatividad con distintas plataformas (como compact framework, incluido en el proyecto) y su integración en los entornos de desarrollo actuales.

Aunque podría haber resultado interesante la inclusión de WSE, se descartó por no disponer de soporte en el framework específico de dispositivos móviles.

Presentación

Los componentes pertenecientes a la capa de presentación y lógica de los clientes utilizarán distintas tecnologías, en función de la tipología del mismo:

- En el acceso web, se ha optado por los estándares **XHTML 1.1** y **CSS 2.1** para la composición de interfaces de usuario, **Javascript** para la programación de interacciones avanzadas en cliente y **ASP.NET** para la lógica de servidor.

  Con el fin de dotar al sistema de modelos de interacción más ricos, ha sido incluida también tecnología **AJAX** en determinadas funcionalidades.

- La aplicación móvil se ha realizado utilizando los **Windows Mobile Forms** de la plataforma **.NET Compact Framework**.
- El cliente de escritorio utiliza la nueva tecnología **Windows Presentation Foundation**, cuyos interfaces han sido compuestos utilizado **XAML**.

4.3. Diagrama estático de diseño

El siguiente diagrama estático recoge las entidades utilizadas por el sistema:

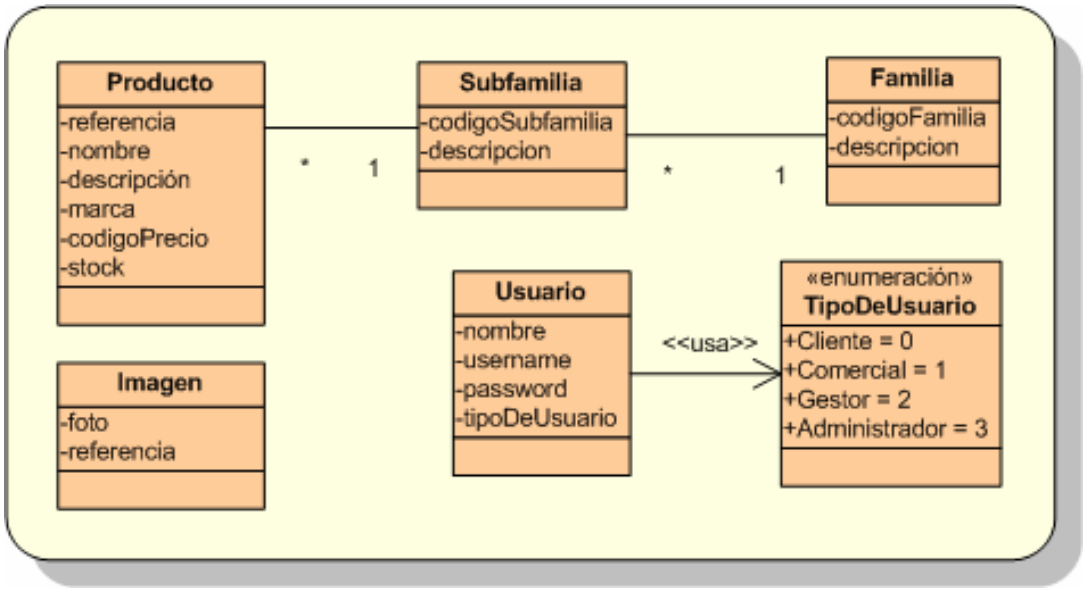


Ilustración 13. Diagrama estático de diseño

Como se puede observar, a las clases propias del modelo del dominio como son las correspondientes a productos, familias y subfamilias, han sido añadidas una clase para representar a las imágenes y otra para los usuarios del sistema y una enumeración con sus tipologías.

Las imágenes de los productos podían haberse implementado como un atributo de éstos, pero se ha optado por mantenerlos separados. De esta forma la obtención y almacenamiento de una imagen puede ser independiente del producto al que pertenece, lo cual facilita y simplifica el trasiego de información entre los clientes y el servidor.

4.4. Diagrama de secuencia

El principal objetivo del diagrama de secuencia es mostrar las interacciones entre los objetos de un sistema. Dado que la forma de interactuar es muy similar en todas las funcionalidades y el tipo de objetos utilizados es homogéneo, vamos a desarrollar exclusivamente el diagrama de secuencia correspondiente al caso de uso GF02 (Crear familia).

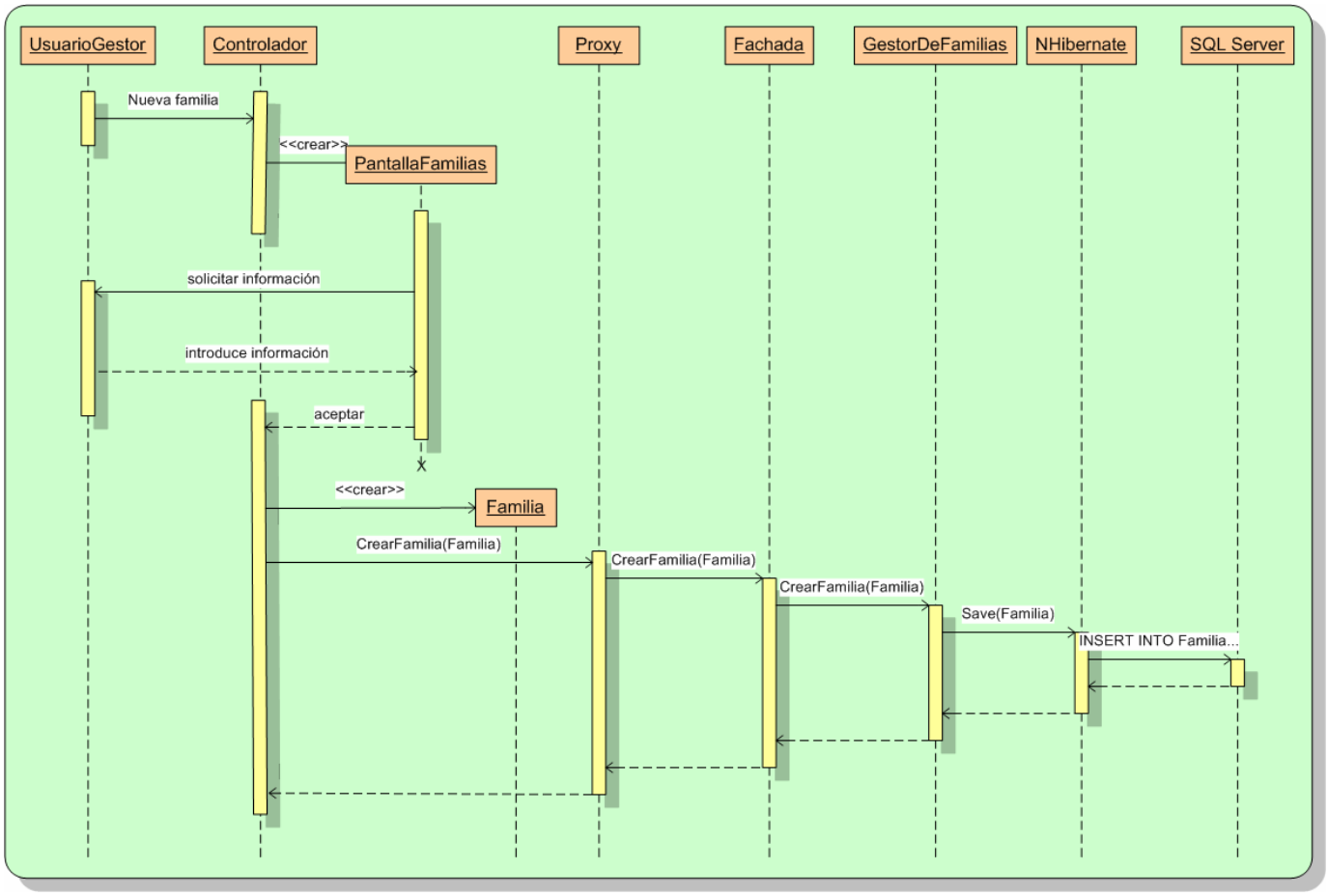


Ilustración 14. Diagrama de secuencia

La secuencia reflejada en el diagrama es la siguiente:

- El usuario (Gestor) envía una señal al Controlador de su interfaz su intención de crear una nueva familia de productos.
- El controlador crea una pantalla de entrada de información.
- La pantalla es mostrada al usuario y espera a que éste introduzca los datos solicitados.
- La pantalla notifica al controlador que el usuario ha aceptado los datos.
- El Controlador crea una instancia de la entidad Familia, poblando sus propiedades con los datos introducidos por el usuario.
- A continuación, el controlador envía al Proxy un mensaje CrearFamilia(), pasándole como parámetro el objeto Familia recién creado.
- El Proxy se pone en contacto con la Fachada, enviándole un mensaje CrearFamilia() con el mismo parámetro que le pasó el controlador.
- La Fachada envía un mensaje CrearFamilia al GestorDeFamilias, reenviándole de nuevo el objeto Familia.
- El GestorDeFamilias envía a un objeto Session (propio de NHibernate, el mecanismo de persistencia utilizado) el mensaje Save(), con el mismo objeto Familia como parámetro.
- NHibernate es quien, finalmente, se pone en contacto con el motor SGBD enviándole las instrucciones oportunas para su almacenamiento, ya en un lenguaje que éste entiende (SQL).

El siguiente diagrama muestra cómo encajan los distintos objetos y componentes citados en la arquitectura lógica descrita con anterioridad:

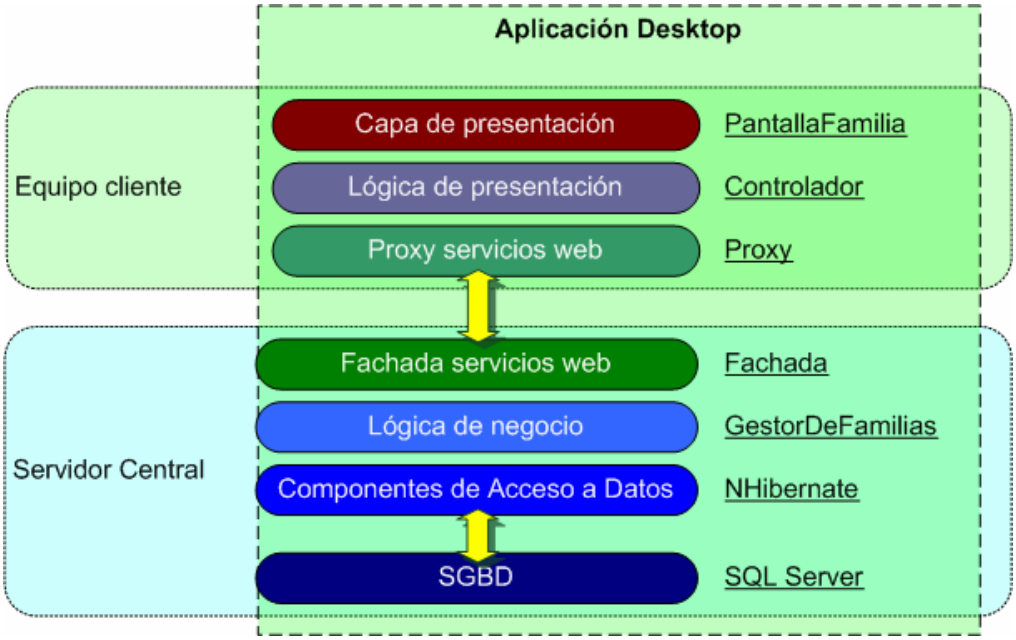


Ilustración 15. Correspondencia de componentes con capas lógicas

4.5. Diseño de la persistencia

4.5.1. Modelo relacional de la base de datos

PRODUCTOS (referencia, nombre, descripción, marca, codigoSubfamilia, stock, codigoprecio)
codigoSubfamilia es clave foránea hacia SUBFAMILIAS

SUBFAMILIAS (codigoSubfamilia, codigoFamilia, descripción)
codigoFamilia es clave foránea hacia FAMILIAS

FAMILIAS (codigoFamilia, descripción)

USUARIOS (username, password, nombre, tipoDeUsuario)
tipoDeUsuario debe admitir valores {1, 2, 3, 4, 5}

IMAGENES (referencia, foto)
referencia es clave foránea hacia PRODUCTOS

4.5.2. Diagrama de base de datos

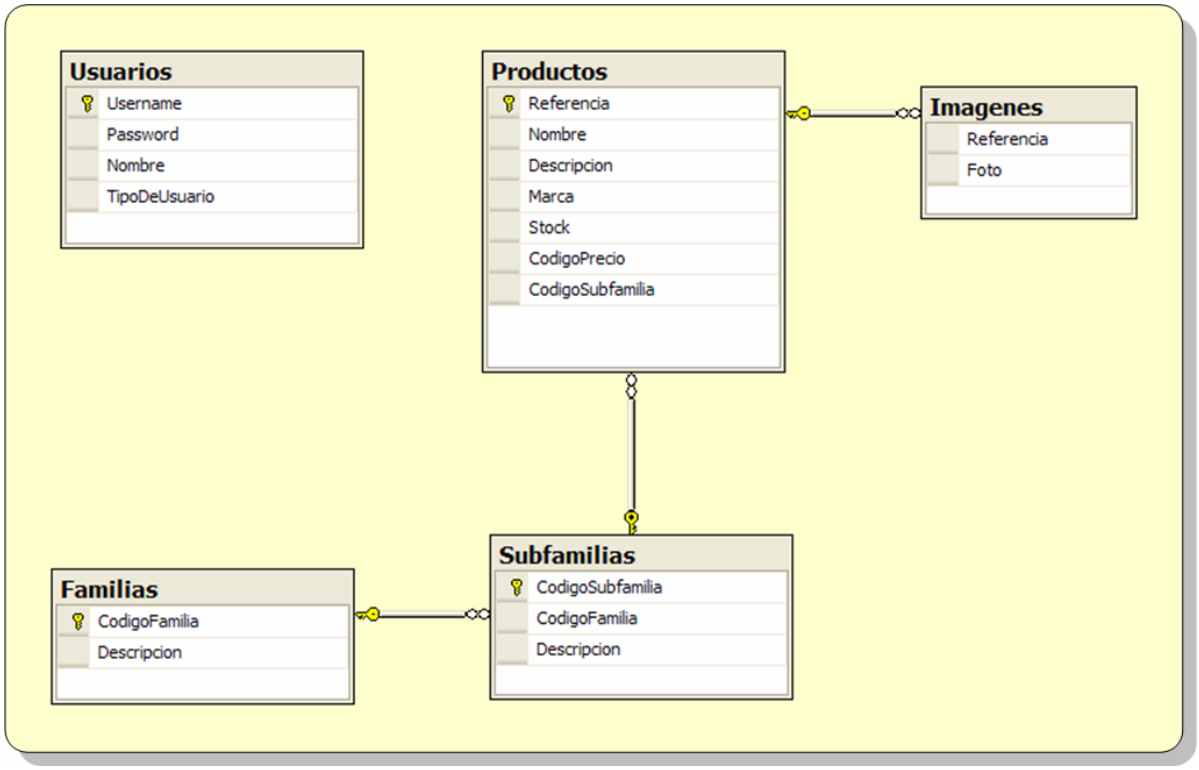


Ilustración 16. Diagrama de Base de Datos

4.6. Prototipos de la interfaz de usuario

Durante la fase de diseño del sistema se desarrollaron una serie de prototipos de las principales pantallas, ventanas y elementos mediante los que la aplicación interactuaría con el usuario, para cada uno de los tres tipos de interfaz (escritorio, web y móvil) incluido en el proyecto.

Dado que estos prototipos se encuentran recogidos en el documento entregado en dicha etapa y no tienen más validez que la presentación conceptual de los interfaces antes de acometer su desarrollo, no van a ser recogidos en su totalidad en este documento. Simplemente se incluirán dos capturas por cada tipo de aplicación.

4.6.1. Aplicación de escritorio

Ilustración 17. Prototipo WPF: Acceso al sistema

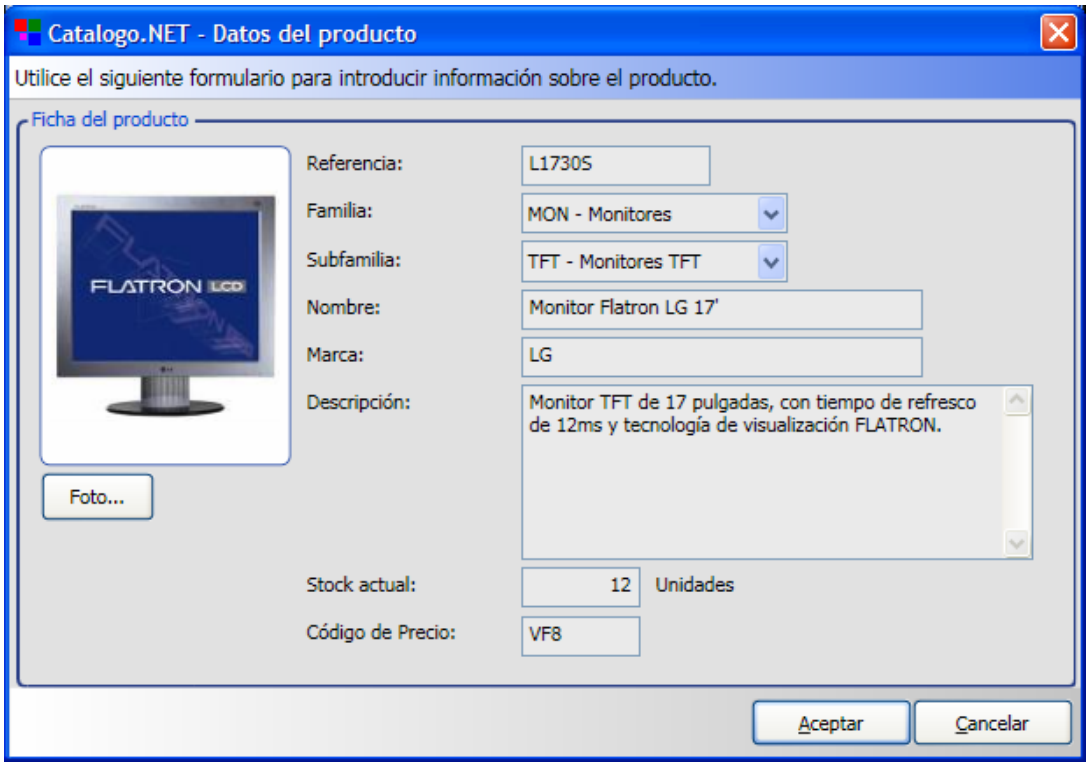


Ilustración 18. Prototipo WPF: Ficha del producto

4.6.2. Aplicación móvil

Ilustración 19. Prototipo PDA: Acceso al sistema

Ilustración 20. Prototipo PDA: Búsqueda de productos

4.6.3. Aplicación Web

Ilustración 21. Prototipo Web: Acceso al sistema

Ilustración 22. Prototipo Web: Cambio de clave

5. Implementación

Los siguientes epígrafes describen el resultado de la implementación de cada uno de los componentes, sus principales características, particularidades, así como problemas específicos que han debido ser solventados para la consecución exitosa del proyecto.

5.1. Componente "CatalogoCentral"

Como se ha descrito anteriormente en este mismo documento, este proyecto implementa la lógica de negocio del sistema, por lo que es el núcleo de los servicios ofrecidos al resto de aplicaciones.

Dado que es independiente de los sistemas que consumen estos servicios, no dispone de elementos de interfaz de usuario y se ha definido como un proyecto de biblioteca de clases.

Asimismo, el ensamblado resultante es absolutamente independiente del modelo de comunicaciones elegido para la transferencia de información entre los clientes y los servicios. Esta particularidad hace posible acceder a estas funcionalidades desde un proyecto de servicios web como el que se incluye en la solución, utilizar una fachada basada en remoting, o vincularlo directamente a una aplicación de usuario.

Los siguientes epígrafes describen los elementos que pueden encontrarse en el interior, así como las particularidades de su implementación.

5.1.1. Entidades

Se han desarrollado clases de datos para las entidades del dominio, en las cuales se han definido simplemente los campos necesarios para almacenar su información.

Únicamente incluyen código en las asignaciones de determinadas propiedades para validar que su contenido sea el correcto. Aunque esta tarea podría realizarse en los gestores (que serán descritos más adelante), se ha decidido incluirla en las entidades para que estas comprobaciones se realicen siempre, asegurando por tanto la integridad de la información que contienen. El intento de asignación de un valor incorrecto lanzará una excepción, que será propagada por las distintas capas del sistema hasta llegar a aquella que pueda tratarla.

5.1.2. Gestores

Los gestores implementan la lógica de negocio de los servicios, poniéndose en contacto con los componentes de persistencia para la lectura o escritura de entidades en el SGBD subyacente.

En general, el patrón seguido para la implementación de las operaciones es similar. En primer lugar se realizan las comprobaciones necesarias para asegurar que la operación puede realizarse para, a continuación, ejecutarla. Cualquier tipo de problema o error generará una excepción tipada con información detallada de su causa.

Es importante destacar que los métodos incluidos en los distintos gestores no realizan comprobaciones de seguridad algunas para no crear dependencias innecesarias entre la lógica de negocio propiamente dicha y el sistema de seguridad de los consumidores de los servicios prestados. Esta tarea se delega a capas superiores de la solución, como la que actúa de fachada de los servicios, que será la responsable de asegurar que el usuario que realiza la petición es correcto, dispone de permisos para acceder a la funcionalidad pretendida, y que su sesión está activa.

Se ha implementado un gestor para cada una de las principales entidades del dominio (productos, familias, subfamilias y usuarios), y un gestor de utilidad con funcionalidades que facilitan el control de sesiones de usuarios, que será descrito a continuación.

5.1.3. Gestor de sesiones interno

El hecho de incluir un sistema propio de control básico de sesiones se debe a la flexibilidad aportada al modelo, que en ningún caso asegura que el sistema de comunicaciones cliente-servicio elegido disponga de su propio gestor de sesiones.

De hecho, como se ha comentado con anterioridad, sería perfectamente posible utilizar una fachada de acceso a los servicios basada en remoting, canales TCP programados ad-hoc, o cualquier otro sistema.

Para asegurar su funcionamiento de este elemento, desde el componente de fachada de servicio, implementado con Servicios Web XML, no se hace uso del sistema de sesiones provisto por la plataforma .NET.

En cualquier caso, el sistema propuesto consiste en mantener en memoria una lista de usuarios activos identificados de forma única por un GUID asignado al mismo en el momento de realizar el login a la plataforma. Para facilitar las búsquedas en la lista, se ha implementado en forma de diccionario genérico.

```
private Dictionary<Guid, Usuario> sesionesActivas =
                                    new Dictionary<Guid, Usuario>();
```

5.1.4. Excepciones

El sistema define excepciones propias para informar de los errores que pueden darse durante la realización de las operaciones.

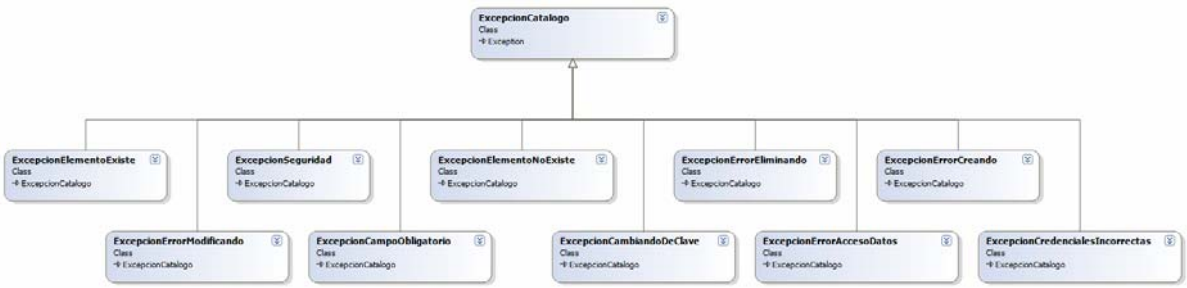


Ilustración 23. Jerarquía de excepciones

Todas ellas heredan de *ExcepcionCatalogo*, e incorporan un código único y un mensaje que facilita la identificación del problema. Esto se ha realizado así debido a la incapacidad de determinados medios (como los servicios web) de propagar las excepciones tipadas propias del servicio hacia el cliente, lo cual, por otra parte, es bastante lógico.

Por tanto, era necesario un mecanismo que desde cliente permitiera identificar claramente la excepción lanzada desde el servicio. En el caso de los servicios web, toda excepción producida en el servidor es enviada al cliente serializada en forma de *SoapException*, por lo que era necesario que la descripción incluyera identificadores que permitieran, manteniendo

el escaso acoplamiento entre ambos componentes, conocer qué problema se había producido en la aplicación y poder eliminar mensajes no apropiados para los usuarios.

5.1.5. Persistencia de entidades

La utilización de NHibernate como motor de persistencia ha hecho necesaria la creación de archivos de mapeo (mappings) entre el modelo de entidades y el relacional.

El siguiente código XML, tomado del archivo *Subfamilia.xhbm.xml*, describe la relación de la tabla *Subfamilias* del SGBD con la entidad *Subfamilia* del sistema, mapea los campos *CodigoSubfamilia*, *Descripcion* y *Familia* hacia las propiedades del mismo nombre, y muestra, además, la multiplicidad de tipo *many-to-one* hacia la entidad *Familia*.

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.0">
  <class name="CatalogoCentral.Entidades.Subfamilia, CatalogoCentral" table="Subfamilias">
    <id name="CodigoSubfamilia" column="CodigoSubfamilia">
      <generator class="assigned" />
    </id>
    <property name="Descripcion" column="Descripcion" type="String(50)" not-null="true" />
    <many-to-one name="Familia" class="CatalogoCentral.Entidades.Familia, CatalogoCentral"
                 column="CodigoFamilia"/>
  </class>
</hibernate-mapping>
```

Para facilitar la localización de estos archivos por el framework y evitar tener que desplegarlos acompañando a los ensamblados ejecutables, han sido incrustados como recursos en el propio resultado del proyecto.

Por otra parte, ha sido necesario indicar en el archivo de configuración de NHibernate qué motor SGBD debe utilizar para salvar los objetos, y cómo ponerse en contacto con él. Esto, además, debe hacerse en todos los componentes que tengan referencias hacia CatalogoCentral puesto que, en última instancia, la plataforma .NET utilizará el archivo de configuración de la aplicación host de este componente.

```
<nhibernate>
  <add key="hibernate.connection.provider"
       value="NHibernate.Connection.DriverConnectionProvider" />
  <add key="hibernate.dialect"
       value="NHibernate.Dialect.MsSql2000Dialect" />
  <add key="hibernate.connection.driver_class"
       value="NHibernate.Driver.SqlClientDriver"  />
  <add key="hibernate.connection.connection_string"
       value="Server=localhost;initial catalog=CatalogoNET;User ID=sa;Password=sa;" />
</nhibernate>
```

Por último, cabe indicar la utilización del lenguaje de consulta *HQL* para la extracción de determinados conjuntos de datos, fundamentalmente cuando se utilizaban criterios de búsqueda.

5.1.6. Persistencia de imágenes

Aunque el almacenamiento de imágenes podría haberse realizado sin contar con el SGBD, puesto que podrían ser almacenadas directamente sobre el sistema de archivos, se ha decidido introducirlas en la base de datos, principalmente por motivos de seguridad e integración con el resto de información manejada por el sistema.

Por ello, ha sido creada la entidad de datos *Imagen* y mapeada en NHibernate al igual que las demás.

5.2. Servicio "ServicioCatalogo"

Este componente implementa la fachada de los servicios basándose en tecnología XML Web Services, es decir, es el componente que actúa como pasarela entre las aplicaciones cliente y los servicios de la plataforma.

A continuación se describen los aspectos más interesantes de este componente.

5.2.1. Página web de comprobación de configuración

Dado que los servicios son publicados bajo un sitio web con una URL propia, se ha creado una página por defecto para el sitio que permite comprobar si la configuración del sistema es correcta a nivel de servicios de publicación web (como IIS) como de parámetros de configuración de la base de datos.

Así, una llamada directa con el navegador a la dirección de publicación del servicio (sin incluir el nombre del archivo concreto) resultaría:

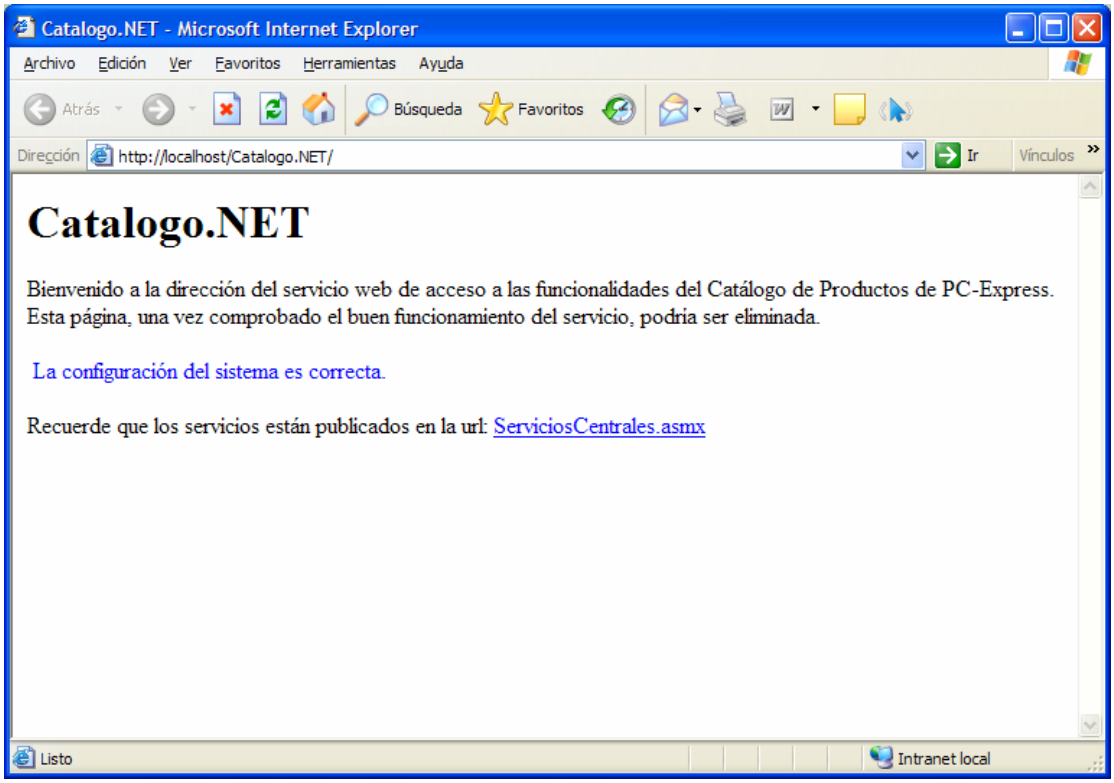


Ilustración 24. Página de comprobación de parámetros del servicio web

5.2.2. Control de acceso

Como se ha comentado anteriormente, es este componente el encargado de realizar las tareas de control de acceso del sistema antes de pasar el control a la lógica de negocio de la aplicación.

El siguiente diagrama de secuencia muestra la interacción entre la aplicación cliente y el servicio necesaria antes de la invocación de una operación. El cliente solicita el acceso al sistema enviándole unas credenciales, y si son correctas, el sistema le devuelve un Token identificativo de la sesión (GUID), el cual deberá acompañar a todas las llamadas subsiguientes a los métodos del servicio.

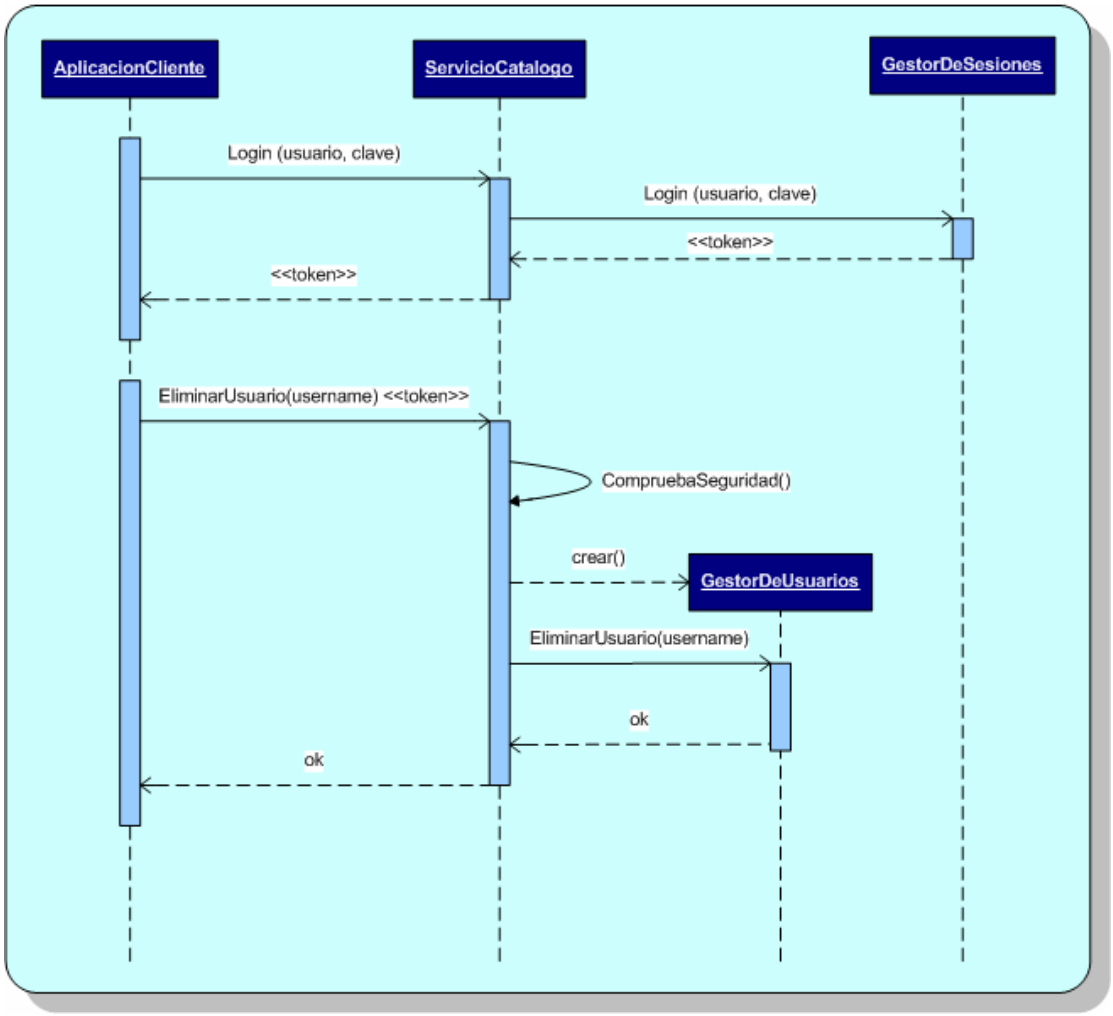


Ilustración 25. Diagrama de secuencia del uso de un servicio

El token se envía desde el cliente en el encabezado del mensaje SOAP, de forma que no interfiere en la firma (signatura) de los métodos.

5.2.3. Métodos de servicio

La práctica totalidad de los métodos expuestos al exterior se encuentran implementados en la clase *ServiciosCentrales* (disponible en ServiciosCentrales.asmx), la cual hereda de *WebService* e implementa todos los interfaces de los gestores declarados en el componente ServicioCentral, para asegurar su coherencia.

La práctica totalidad de métodos se adscriben al siguiente patrón:

```
[WebMethod(), SoapHeader("InfoSesion")]
public bool EliminarProducto(string referencia)
{
      CompruebaNivelSeguridadMinimo(TipoDeUsuario.Gestor);
      GestorDeProductos gp = new GestorDeProductos ();
      return gp.EliminarProducto(referencia);
}
```

Se pueden observar los siguientes aspectos:

- El atributo *WebMethod()* previo a la declaración del método indica que este método se publicará en el servicio web, siendo descrito a través del contrato WSDL.

- El atributo *SoapHeader* indica que la petición debe ir acompañada de un objeto de tipo InfoSesion en su encabezado; en caso contrario, no será procesada. Este atributo no aparecerá en aquellos métodos para los que no es necesario estar previamente identificado, como por ejemplo, *Login()*.

- La primera línea de código realiza una comprobación del nivel de seguridad de la función. Si el usuario que ha realizado la petición no pertenece como mínimo a ese nivel, se lanzará una excepción.

- Por último, se instancia el gestor apropiado y se invoca al método apropiado, retornando, si procede, el valor devuelto por éste.

5.3. Aplicación "CatalogoPDA"

Esta aplicación implementa el cliente del sistema creado específicamente para dispositivos móviles utilizando Pocket PC 2003 y .NET Compact Framework 2.0.

La utilización de Servicios Web XML para la comunicación con el servidor ha facilitado en gran medida la programación del sistema, puesto que se ha reducido prácticamente a la composición de los interfaces de usuario apropiados para el dispositivo y la implementación de ciertos controles y lógica asociada a la presentación.

Sin embargo, son interesantes algunos detalles en su implementación, que son descritos a continuación.

5.3.1. Conexión con el servidor central

La plataforma .NET 2.0 incorpora un interesante mecanismo de parametrización de las direcciones de acceso a servicios web que no está disponible en las ediciones compactas del framework por carecer de soporte para los archivos de configuración. Por este motivo, ha sido necesaria la preparación de un sistema a medida que solventara el problema y facilitara la configuración de la herramienta.

Por este motivo, el Proxy generado automáticamente desde Visual Studio partiendo de las descripciones WSDL de los servicios ha tenido que ser retocado de forma manual. De hecho, en el constructor del mismo, incluido en el archivo "reference.cs" se ha sustituido la asignación *hard-coded* de la URL del servicio web por una dirección obtenida de la configuración del sistema:

```
public ServiciosCentrales()
{
        this.Url = Utilidades.ObtenerURL();
}
```

El método estático ObtenerURL() devuelve la dirección del servicio web donde debe conectarse el sistema para acceder a las funcionalidades ofrecidas por la central, intentando, en primer lugar, obtenerla de un archivo de configuración que debe estar ubicado en el mismo directorio que el ejecutable; de no encontrarlo, se utiliza el valor de la constante definida al principio de la clase:

```
// URL al servicio web, desde la PDA
private const string SERVICE_URL =
    "http://192.168.1.2/Catalogo.NET/ServiciosCentrales.asmx";
```

5.3.2. Archivos de configuración

Como se ha comentado con anterioridad, el Compact Framework no dispone de soporte para archivos de configuración como el incluido en las ediciones superiores de la plataforma, donde se utilizan los archivos XML app.config o web.config para almacenar y recuperar parámetros de forma muy sencilla.

Por este motivo ha sido necesario crear un formato propio, basado gramatical y sintácticamente en los .config citados, que son leídos durante el arranque del sistema para obtener la dirección del servicio (código tomado del archivo Utils.cs):

```
...
XmlTextReader reader = new XmlTextReader(path);
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.Load(reader);
reader.Close();
XmlNodeList nodes = xmlDoc.GetElementsByTagName("add");
foreach (XmlNode node in nodes)
{
    if (node.Attributes["key"].Value.ToUpper().Equals("SERVICEURL"))
          return node.Attributes["value"].Value;
}
...
```

5.3.3. Generación de imágenes

Las imágenes son almacenadas en el servidor en la base de datos y transferidas a los clientes como un array de bytes en el interior de la entidad *Imagen*, por lo que se requiere cierta manipulación antes mostrarlas en un formulario.

En este caso, basta con crear un bitmap en memoria partiendo del stream (array de bytes) contenido en el objeto imagen, y establecerlo como imagen en el control visual:

```
Imagen im = proxy.ObtenerImagen(prod.Referencia);
pbImagen.Image = new Bitmap(new System.IO.MemoryStream(im.Foto));
```

5.3.4. Capturas de pantalla del producto

Aunque el prototipo de pantallas que se presentó durante la fase de diseño era conceptualmente correcto, la estética se ha variado considerablemente para hacerlo más acorde con el resto del producto. A continuación se muestran algunas de las pantallas principales de la aplicación en funcionamiento.

Ilustración 26. Pantallas finales de CatalogoPDA

5.4. Aplicación "CatalogoWPF"

Es la principal herramienta de gestión del catálogo de productos, y está conceptualizada como aplicación de escritorio para entornos Windows con .NET Framework 3.0. La principal novedad de este componente es la utilización de Windows Presentation Foundation.

5.4.1. Creación del interfaz de usuario

Los formularios y elementos visuales han sido creados utilizando XAML, bien directamente o bien utilizando algunos de los editores disponibles, dentro del propio Visual Studio o externos como Microsoft Expression Blend.

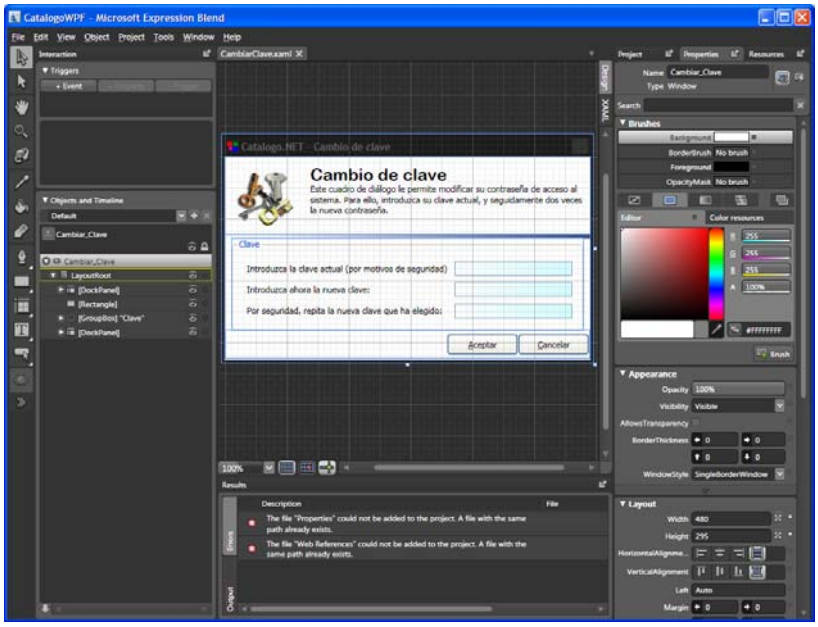


Ilustración 27. Composición de interfaces con Microsoft Expression Blend

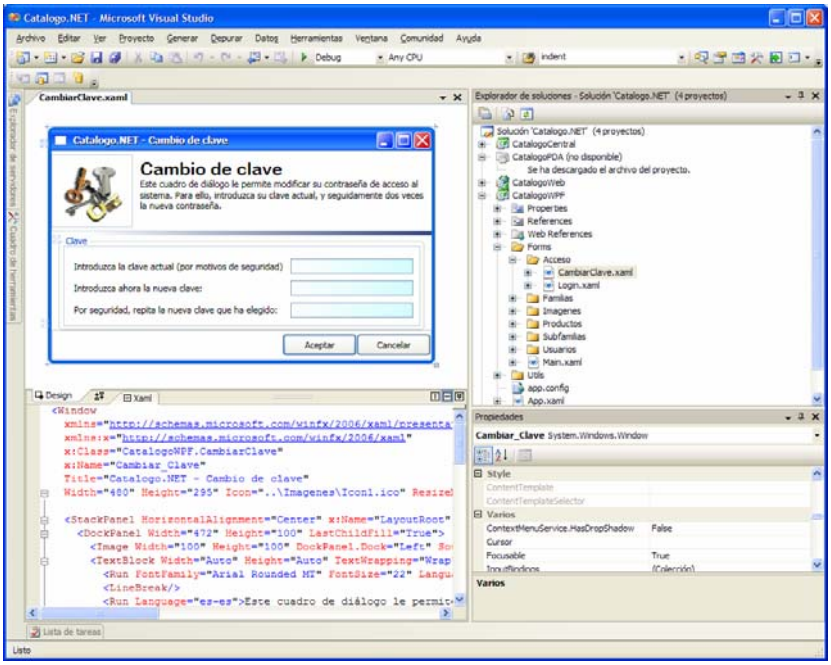


Ilustración 28. Diseñador de Visual Studio para WPF

La escasa madurez de estos editores visuales en el momento de creación del sistema han supuesto más de una vez un auténtico problema, así como la falta de compatibilidad entre ambos, debiendo resolver los conflictos trabajando directamente sobre el archivo .xaml con un editor de textos estándar.

5.4.2. El archivo "code-behind"

Los formularios y elementos WPF siguen un modelo de desarrollo bastante similar al que podemos usar en sistemas web programados con ASP.NET: por una parte se dispone de un archivo que contiene todos los elementos de presentación y nula o muy escasa lógica, y por otra parte un archivo de código adicional donde se implementa la lógica real.

El sistema, ya en tiempo de compilación, generará una clase parcial con los atributos que representan a los elementos visuales incluidos en el archivo XAML, y cuyo nombre y namespace se especifica en la etiqueta de apertura del elemento:

```
<Window
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   x:Class="CatalogoWPF.CambiarClave"
   x:Name="Cambiar_Clave"
   Title="Catalogo.NET - Cambio de clave"
   Width="480" Height="295" Icon="..\Imagenes\Icon1.ico"
   ...
```

La implementación de eventos, inicializaciones o cualquier otra funcionalidad se realizaría en el archivo "code-behind", en el interior de una clase parcial con el mismo nombre que la anterior, lo que le da acceso directo a los controles del formulario a través de sus atributos.

Así, una declaración de un botón en XAML al que se le pretende asociar un controlador de evento de pulsación sería:

```
<Button Width="92"    Height="27"      Content="_Aceptar"
        Cursor="Hand" IsDefault="True"
        Name="btnAceptar" Click="btnAceptarClick"
/>
```

Como consecuencia, la clase de código debería contener un método como el que sigue, que recibiría el evento de pulsación del botón:

```
private void btnAceptarClick(object sender, RoutedEventArgs e)
{
      ... hacer algo
}
```

5.4.3. Databindings

El enlace a datos de controles visuales es una interesantísima capacidad de WPF, aunque ya estaba disponible en tecnologías anteriores como Windows Forms.

Básicamente, consiste en la capacidad de asociar a un control visual una fuente de datos y configurar qué debe mostrar y cómo hacerlo.

CatalogoWPF, que es una aplicación cliente que se comunica con la central a través de los servicios web, no dispone de acceso directo a las fuentes de datos por lo que el enlace se realiza con las colecciones de objetos (entidades) devueltas por los servicios de la plataforma central.

El siguiente código muestra la porción de XAML donde se define una vista en rejilla de las familias de productos:

```
<ListView  Margin="149.842,71.953,8,16.381" Grid.Row="1"
      IsSynchronizedWithCurrentItem="True" SelectionMode="Single"
      Cursor="Hand" SelectionChanged="Lista_SelectionChanged" >
  <ListView.View >
    <GridView>
      <GridViewColumn x:Name="ColCodigo" Header="Código de Familia"
            DisplayMemberBinding ="{Binding CodigoFamilia}"  />
      <GridViewColumn x:Name="ColDesc" Header="Descripción"
            DisplayMemberBinding ="{Binding Descripcion}" />
    </GridView>
  </ListView.View>
</ListView>
```

Como se puede observar, en la rejilla se definen dos columnas, siendo la primera para visualizar el código de la familia y la segunda para la descripción. En cada una se ha especificado un atributo llamado “DisplayMemberBinding” donde se indica de qué propiedad se obtendrá el valor a mostrar en cada caso.

Ya desde el código se realiza el enlace con los datos, una vez obtenida la lista de familias desde el servidor:

```
Binding bind = new Binding();
Lista.DataContext = listaDeFamilias;
Lista.SetBinding(ListView.ItemsSourceProperty, bind);
```

De esta forma, a la hora de cargar los datos en el control visual (grid), el sistema recorrerá los elementos del array (o colección) *listaDeFamilias*, creando una fila de datos por cada uno de ellos, y mostrando en las columnas el valor de los atributos *CodigoFamila* y *Descripcion*.

Sin embargo, aunque este modelo es válido cuando la colección de datos a las que se enlaza el control no va a variar, presenta algunas contraindicaciones cuando el escenario de utilización incluye la actualización de los datos, como es el caso en las funcionalidades de mantenimiento de la aplicación. Esto es así debido a que el enlace a datos se realiza en la inicialización del formulario y si cambia algún elemento de la lista (imaginemos, por ejemplo, que el usuario edita la descripción de una familia), el control visual es ajeno a ello.

Por esta razón ha sido necesario crear *wrappers*, clases que envuelven a las entidades a mostrar, que implementan el interfaz *INotifyPropertyChanged* y son capaces de notificar al control visual que las contiene la modificación de algunas de sus propiedades para forzar su actualización.

5.4.4. Procesado de imágenes

Esta aplicación incorpora, al igual que CatalogoPDA, un mecanismo para tomar el array de bytes contenido en la entidad Imagen y transformarlo en una imagen mostrable en un control visual.

El siguiente código muestra como se obtiene una instancia de *BitmapImage* partiendo del objeto de tipo Imagen que contiene los bytes de la fotografía del producto. Esta instancia puede utilizarse directamente sobre la propiedad *source* de un control Image del formulario.

```
BitmapImage bmp = new BitmapImage();
MemoryStream ms = new MemoryStream(imagen.Foto);
bmp.BeginInit();
bmp.StreamSource = ms;
bmp.EndInit();
```

Además, CatalogoWPF incluye el proceso contrario, es decir, el envío de imágenes a la plataforma a través del servicio web, para lo que previamente hay que transformarla en un array de bytes.

El siguiente código muestra que, en primer lugar, se obtiene un array de bytes correspondiente al archivo cuya ruta (*filename)* conocemos, y a continuación se crea un *BitmapImage* con esa información.

```
FileStream fs = new FileStream(filename, FileMode.Open);
this.bufferImagenActual = new byte[fs.Length];
cargaArray(fs, bufferImagenActual);
BitmapImage bmp = new BitmapImage();
bmp.BeginInit();
bmp.StreamSource = new MemoryStream(bufferImagenActual);
bmp.DecodePixelWidth = 130; // Ancho por defecto
bmp.EndInit();
this.imgFoto.Source = bmp;  // Lo mostramos en pantalla
fs.Close();
```

5.4.5. Capturas de pantalla del producto

A continuación se muestran algunas pantallas capturadas de la aplicación CatalogoWPF en ejecución.

Ilustración 29. Menú principal de CatalogoWPF

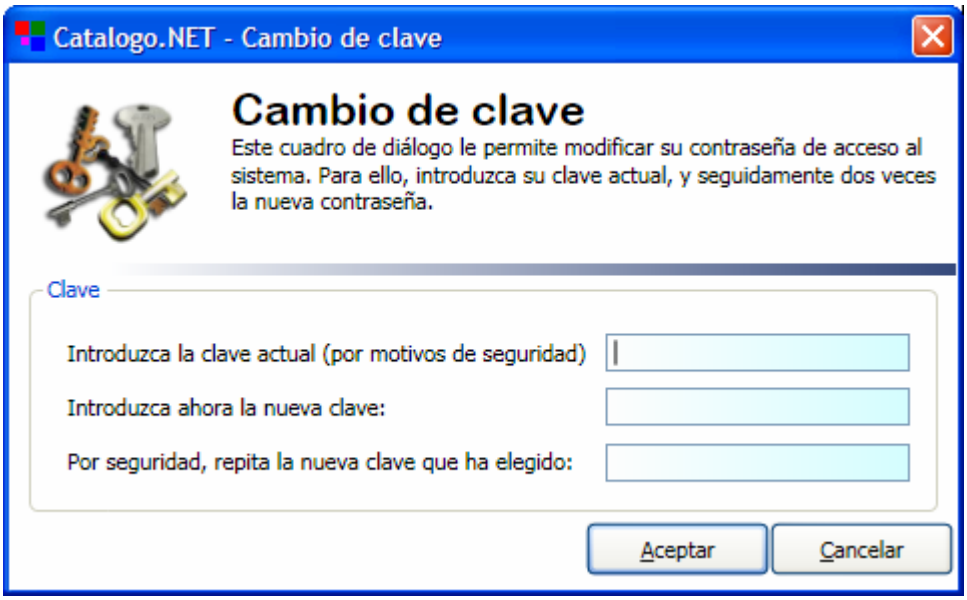


Ilustración 30. Pantalla de cambio de clave en CatalogoWPF

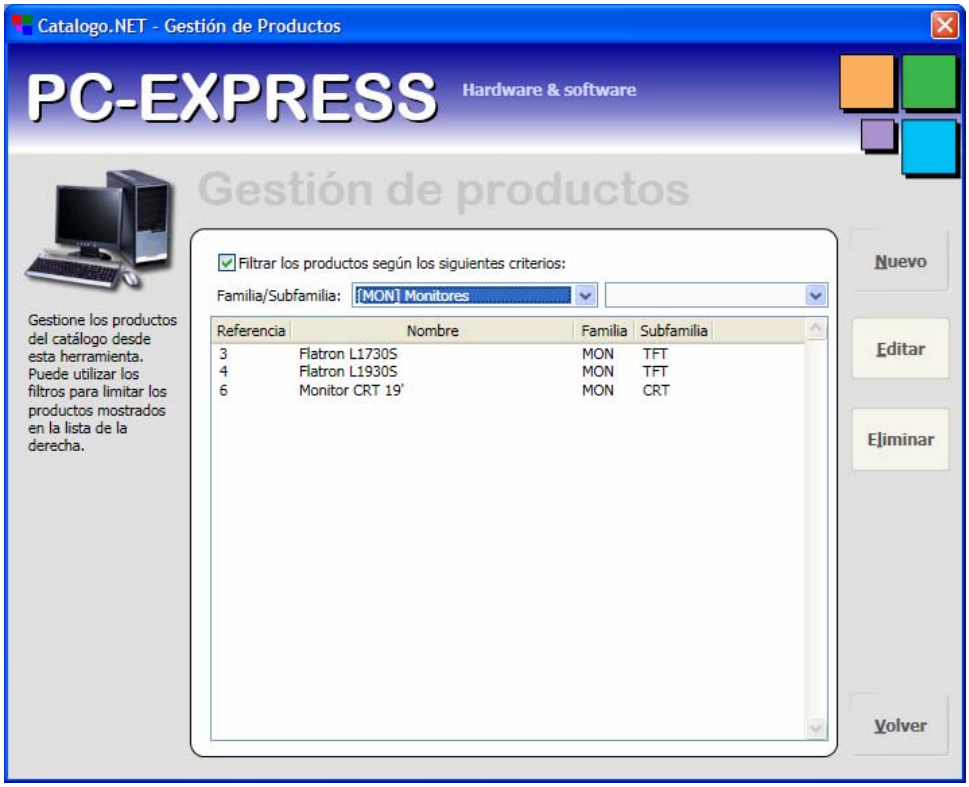


Ilustración 31. Gestión de productos en CatalogoWPF

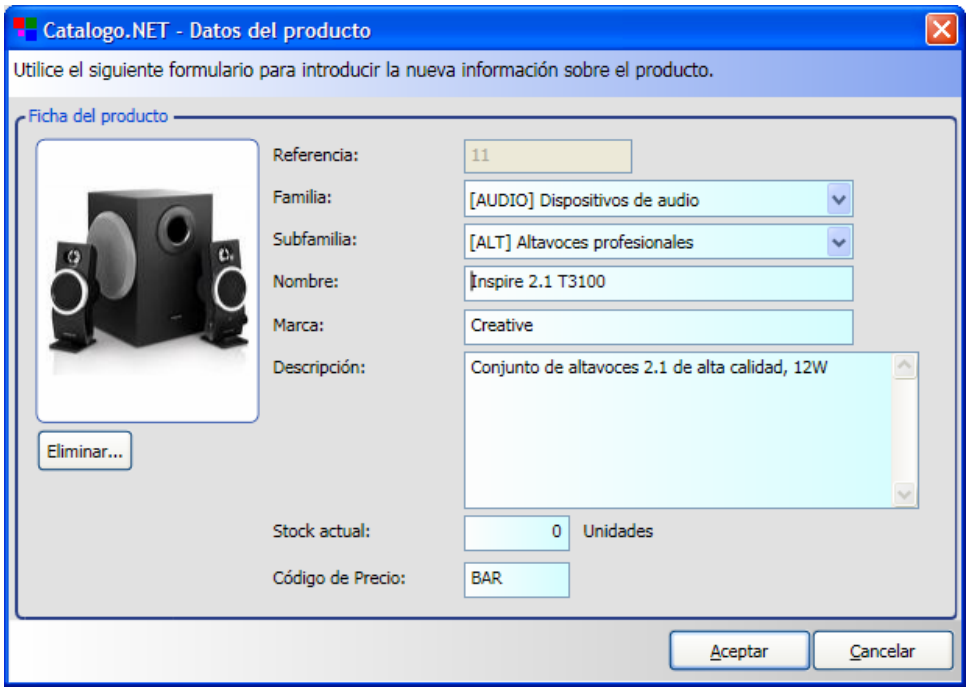


Ilustración 32. Ficha de productos en CatalogoWPF

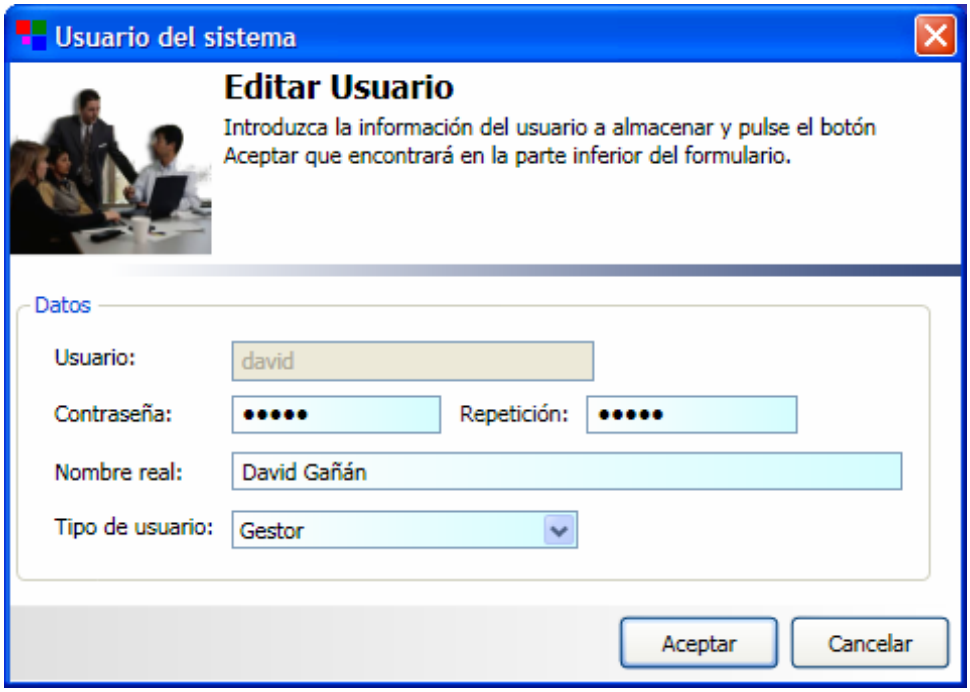


Ilustración 33. Ficha de usuarios en CatalogoWPF

Ilustración 34. Confirmación de eliminación de familias en CatalogoWPF

5.5. Aplicación "CatalogoWEB"

Esta aplicación constituye el acceso a la consulta del catálogo de productos vía Internet utilizando un navegador. Este sistema, a su vez, se comunica con la central mediante servicios web, por lo que su instalación podría encontrarse en equipos físicamente no contiguos.

Las siguientes secciones desarrollan aspectos interesantes de la implementación de este sistema.

5.5.1. Uso de estándares

Para la maquetación de contenidos se ha optado por seguir los estándares XHTML 1.1 y CSS 2.1. Las páginas han sido validadas utilizando herramientas on-line proporcionadas por la W3C y se ha comprobado la correcta presentación con los navegadores más utilizados del mercado, Internet Explorer 6 y Firefox 2.0.

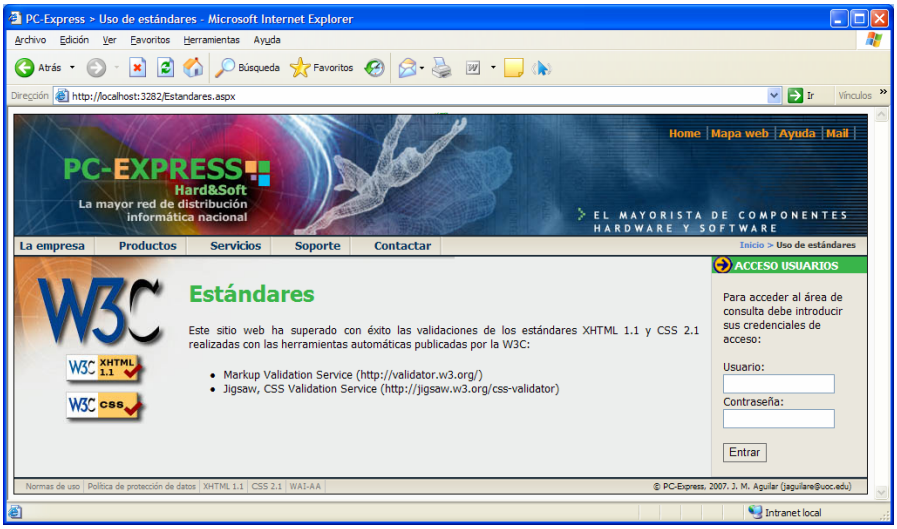


Ilustración 35. Uso de estándares en CatalogoWeb

Ha sido necesario activar el modo estricto en la sección <system.web> del archivo de configuración web.config para que los controles propios de la plataforma .NET generaran código XHTML válido, de la siguiente manera:

```
<xhtmlConformance mode="Strict" />
```

En cualquier caso, cuando se han detectado controles que no cumplen la normativa, como los mapas web, se han descartado y se han realizado implementaciones propias.

5.5.2. Implementación del sitio web completo

Aunque quedaba fuera del alcance inicial previsto para el presente Proyecto de Fin de Carrera, se ha desarrollado todo el entorno en el que se implementaría el acceso al Catálogo de Productos.

Esto se ha decidido así principalmente por tener la oportunidad de probar características interesantes de ASP.NET 2.0, como las páginas maestras, mapa web, el control breadscrumb (migas de pan), etc. Asimismo, se consideró interesante la inclusión del acceso privado al

catálogo de productos en el entorno donde realmente debería encontrarse, en el seno de una web corporativa.

Dado que, obviamente, se trata de una empresa ficticia, los contenidos de las distintas secciones son textos de relleno, salvo en:

- Mapa Web, donde se describe el contenido del sitio utilizando la información del archivo *Web.Sitemap*.
- XHTML y CSS, que incluye información de las validaciones realizadas al código.
- WAI, donde se describen características de accesibilidad reales de la página.

5.5.3. Páginas de error 404 personalizadas

ASP.NET permite la creación de páginas de error personalizadas. La forma de activarlos es indicando en el archivo de configuración que desea usarse esta capacidad e indicando el nombre de página que responde a cada código de error concreto.

```
<customErrors mode="On">
    <error statusCode="404" redirect="PaginaNoEncontrada.aspx" />
</customErrors>
```

Las páginas de error personalizadas dan un aspecto muy profesional a los sitios web, y ayudan a los usuarios a recuperarse de situaciones anómalas o problemas con el sitio. Para el sitio web de PC-Express, se ha controlado el error estándar 404 del protocolo HTTP (página no encontrada) y se han establecido enlaces a páginas inexistentes en algunas opciones (Ayuda, Normas de uso y Protección de datos) con objeto de provocar conscientemente el salto a la misma.

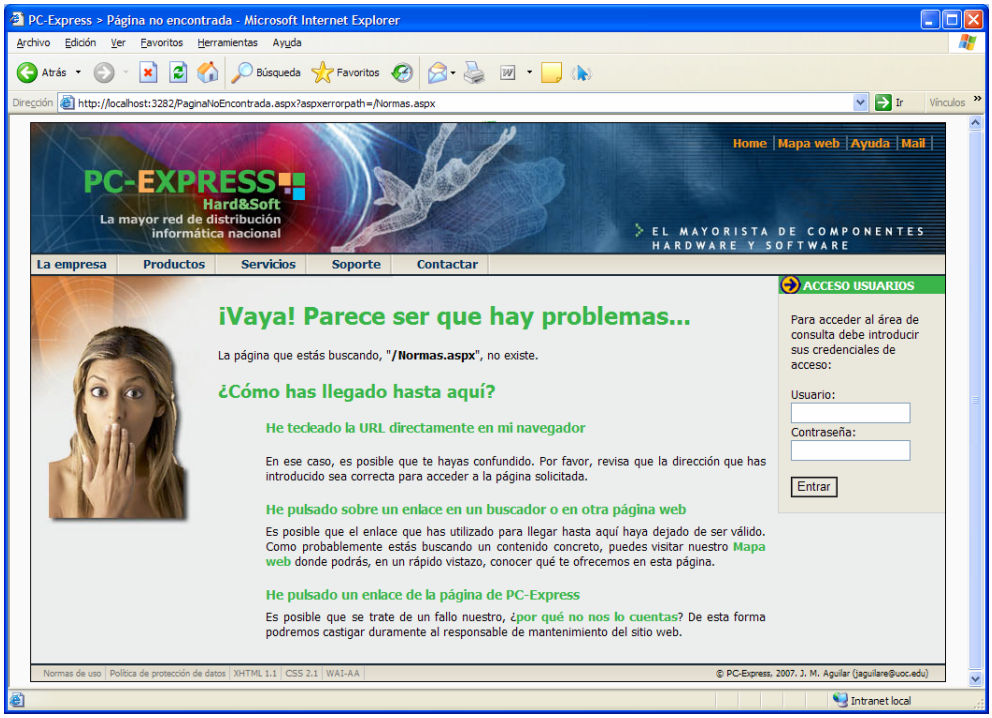


Ilustración 36. Página de error personalizada en CatalogoWeb

5.5.4. Mapa web, migas de pan

ASP.NET 2.0 provee de un mecanismo integrado de descripción de contenidos de un sitio web basado en la implementación de un archivo especial (.sitemap), que puede ser después utilizado para la composición de mapas web y el seguimiento de la ubicación actual del usuario.

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="default.aspx" title="Inicio"  description="Página de inicio">
    <siteMapNode url="Mapaweb.aspx" title="Mapa web"  description="Mapa de contenido de la web" />
    <siteMapNode url="Ayuda.aspx" title="Ayuda"  description="Ayuda a la navegación" />
    <siteMapNode url="Empresa.aspx" title="La empresa"  description="Información sobre PC-Express" />
    <siteMapNode url="Productos.aspx" title="Productos"  description="Productos comercializados por la empresa" />
    <siteMapNode url="Servicios.aspx" title="Servicios"  description="Servicios ofrecidos" />
    <siteMapNode url="Soporte.aspx" title="Soporte"  description="Área de soporte de productos" />
    <siteMapNode url="Contacto.aspx" title="Contactar"  description="Contactar con PC-Express" />
    <siteMapNode url="" title="Área de usuarios"  description="Área de usuarios" >
      <siteMapNode url="Catalogo.aspx" title="Catálogo de productos"  description="Catálogo de productos"  />
      <siteMapNode url="Detalles.aspx" title="Detalle de producto" description="Detalle de producto" />
      <siteMapNode url="Cambiarclave.aspx" title="Cambiar clave"  description="Cambiar clave" />
    </siteMapNode>
    <siteMapNode url="Accesibilidad.aspx" title="Accesibilidad"  description="Características de accesibilidad de la página" />
    <siteMapNode url="Estandares.aspx" title="Uso de estándares"  description="Utilización de estándares" />
  </siteMapNode>
</siteMap>
```

Ilustración 37. Archivo Web.sitemap

El mapa de CatalogoWEB se encuentra en la página Mapaweb.aspx, al que se puede acceder pulsando la opción correspondiente del menú superior, muestra el contenido de este mapa. Sin embargo, dado que el control original incluido en el framework no genera código válido XHTML, ha sido necesario implementar una versión propia del mismo, que muestra la estructura de contenidos realizando un recorrido recursivo en profundidad sobre el árbol definido en el archivo Web.sitemap.

Para las migas de pan (breadscrumb), sin embargo, se ha utilizado el control estándar incluido en la plataforma .NET.

Es importante destacar que ambos elementos son considerados fundamentales por la mayoría de criterios de accesibilidad y usabilidad en sitios web.

5.5.5. Otros elementos de accesibilidad

El sitio CatalogoWEB ha sido concebido teniendo en cuenta las pautas de accesibilidad descritas en las WCAG 1.0, así como otras recomendaciones en el mismo sentido:

- Utilización de un orden lógico de presentación de los elementos en pantalla y de los contenidos en el código.
- Uso de los estándares del W3C
- Gramática XHTML 1.1 y CSS 2.1
- Cumplimiento de las pautas de Accesibilidad para el Contenido Web (WCAG) en su nivel doble A (AA), además de ciertas reglas de nivel AAA.
- Separación de contenido y presentación mediante hojas de estilo.
- Uso de unidades relativas (permiten redimensión de texto).
- Juego de colores con contraste.
- Estructura homogénea y navegación coherente:
    - Menú general, donde se encuentran enlaces hacia la página de inicio, mapa web, ayuda y contacto por email.
    - Menú de contenidos, desde donde se puede navegar a seccíones de la web
    - Zona de usuario, con opciones donde los usuarios registrados pueden realizar ciertas operaciones
    - Contenidos adicionales, al pie de la página, con información diversa.

- Inclusión de atajos de teclado:
  - Es posible en cualquier momento navegar directamente hasta la sección de contenidos utilizando el atajo de teclado "C".
  - También es posible acudir al mapa web de forma directa utilizando el atajo de teclado "M".
  - Acceso al formulario de login de usuarios, atajo "U".
  - Acceso al formulario de criterios de búsqueda de productos, atajo "R".

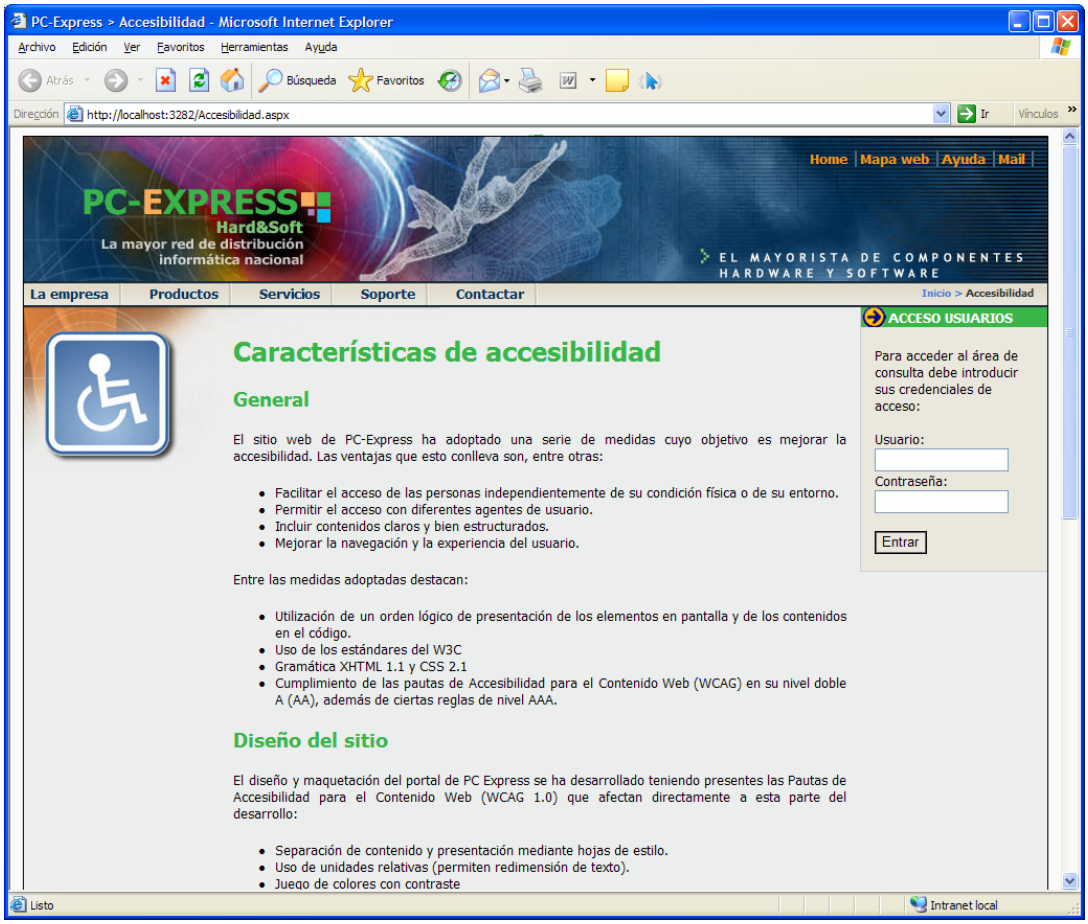


Ilustración 38. Características de accesibilidad de CatalogoWEB

A pesar de todo, es posible que se encuentren problemas en la navegación utilizando dispositivos especiales debido a la utilización de AJAX. Esta novedosa tecnología de interacción con el usuario requiere Javascript, por lo que no será posible utilizar determinadas funcionalidades si esta característica no está activada.

5.5.6. Cambio de clave AJAX

Para la funcionalidad de cambio de clave se ha utilizado un control, disponible en el AJAX Control Toolkit, que aportan un alto nivel de interactividad con el usuario mientras teclea la nueva contraseña: *PasswordStrength*.

Se trata de una extensión al *TextBox* (o *input* de tipo texto o *password*), que comprueba las características de la contraseña que el usuario va introduciendo conforme va tecleando, y muestra mensajes que le ayudan a introducir una clave con cierta resistencia a ataques de fuerza bruta.

Una vez arrastrado el control a la página, se le indica el control al que deberá atender mediante la propiedad *TargetControlID* y a partir de este momento, se podrán configurar en el control destino las características de la validación que se desea hacer utilizando el diseñador visual:

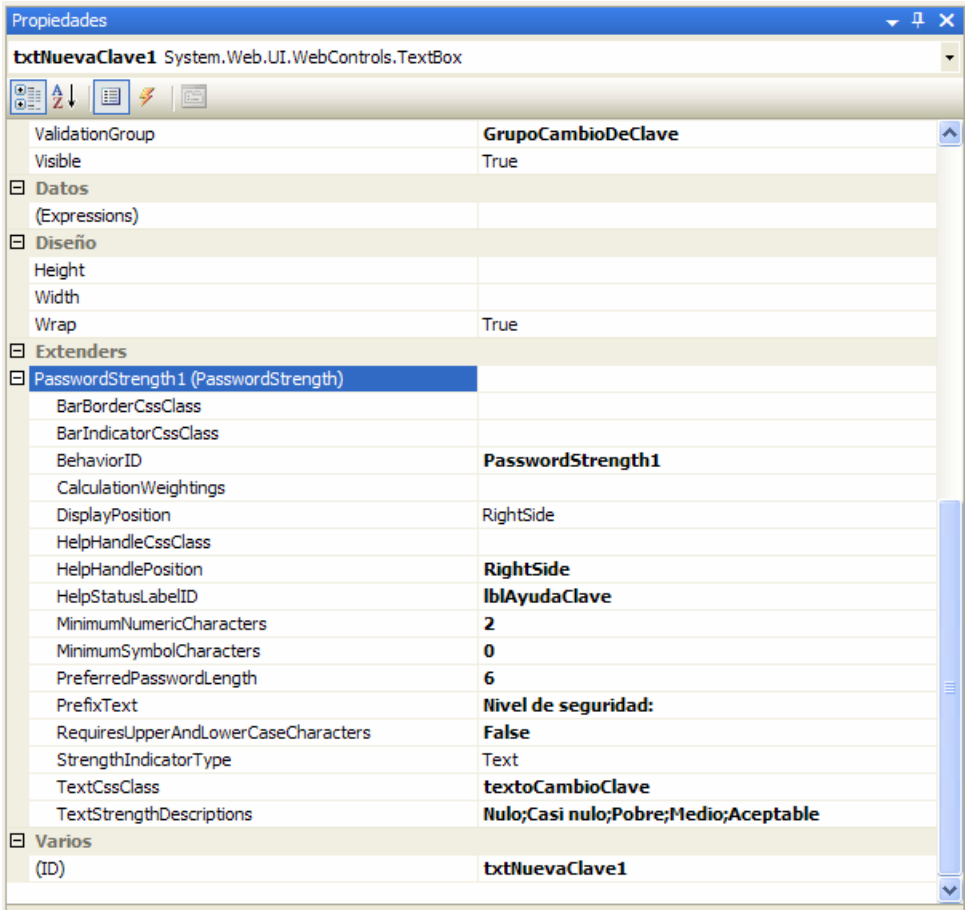


Ilustración 39. Propiedades del extensor PasswordStrength

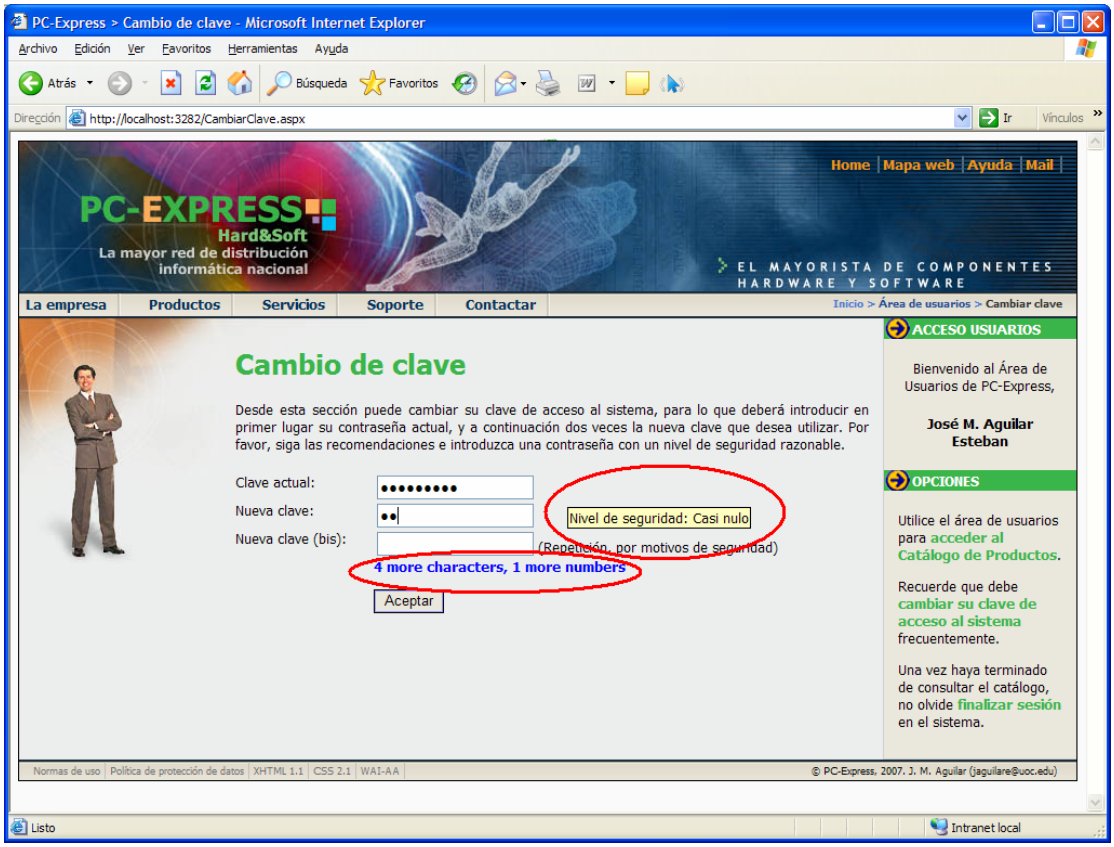


Ilustración 40. Cambio de clave al estilo AJAX

PasswordStrength comprueba en cada pulsación de tecla sobre el cuadro de edición qué nivel de seguridad tiene la contraseña en ese instante y como se puede observar en la captura anterior, indica qué elementos faltan para que ésta pueda ser considerada segura.

Es importante destacar que en el momento de creación de este proyecto el control no estaba aún localizado al español, por lo que los mensajes de las indicaciones aparecen en inglés.

5.5.7. Buscador de productos AJAX

Si bien la funcionalidad anterior, el cambio de clave, ya incluye funciones avanzadas de scripting, es en el buscador de productos donde realmente se ha implementado AJAX en su totalidad, demostrando hasta qué punto puede ser potente esta tecnología.

Paradójicamente, el resultado es bastante mejor utilizando Firefox que Internet Explorer. En este último las recargas parciales de página no son todo lo suaves que lo son en el primero, pudiendo producirse pantallazos algo molestos. Incluso uno de los controles (CollapsiblePanel) no funciona correctamente.

A continuación se describen las características AJAX incluidas en el buscador de productos.

5.5.7.1. Paneles plegables

El panel de criterios de búsqueda está incluido en un panel plegable (Collapsible Panel), que permite cerrarlo, ocultando así su contenido, o abrirlo, volviendo a mostrarlo.

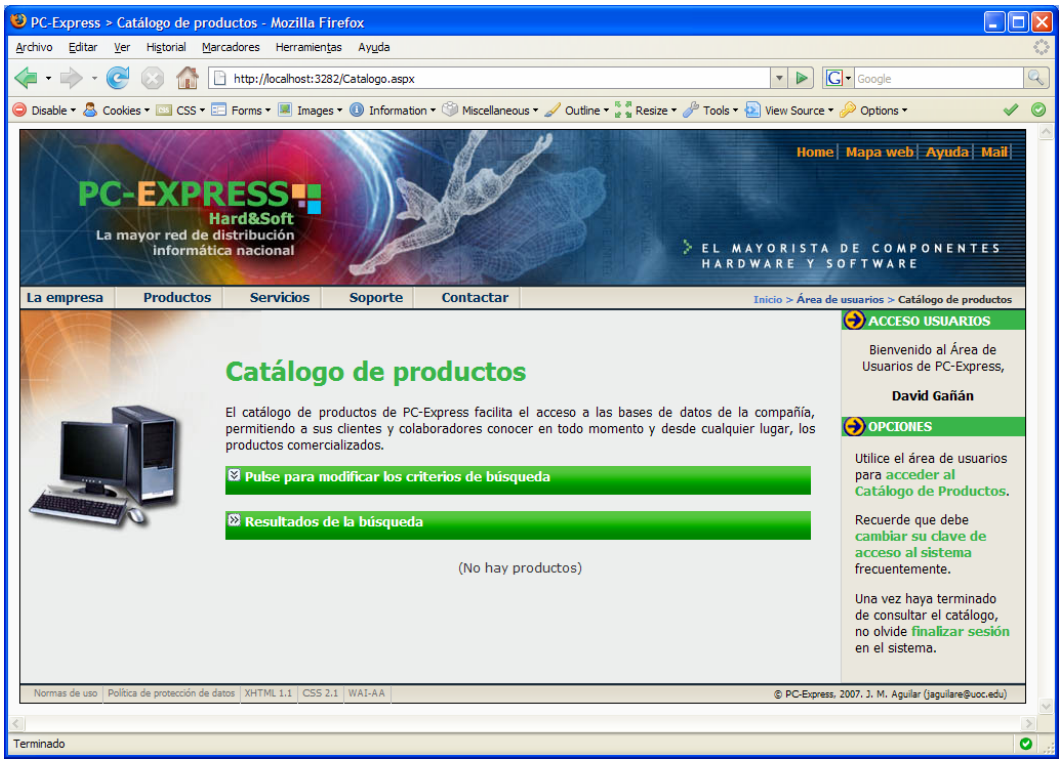


Ilustración 41. Panel de criterios de búsqueda plegado

Como se ha comentado anteriormente, el control no funciona correctamente en Internet Explorer, pero sí lo hace en Firefox.

5.5.7.2. Cascading Dropdowns

Dado que el buscador de productos debía incorporar como criterios la familia y subfamilia a la que pertenecen los elementos a localizar y que existe una relación de dependencia jerárquica entre ambos atributos, se decidió incluir desplegables enlazados con carga dinámica.

En otras palabras, el buscador incluye un desplegable de familias de productos; al seleccionar una de ellas, el sistema se pone en contacto con el servidor y solicita las subfamilias que tiene asociadas, cargando un segundo desplegable. El llenado de ambos desplegable es asíncrono, y utiliza javascript para comunicar con un servicio web en el servidor es que actuará como proveedor de datos para los controles de la página.

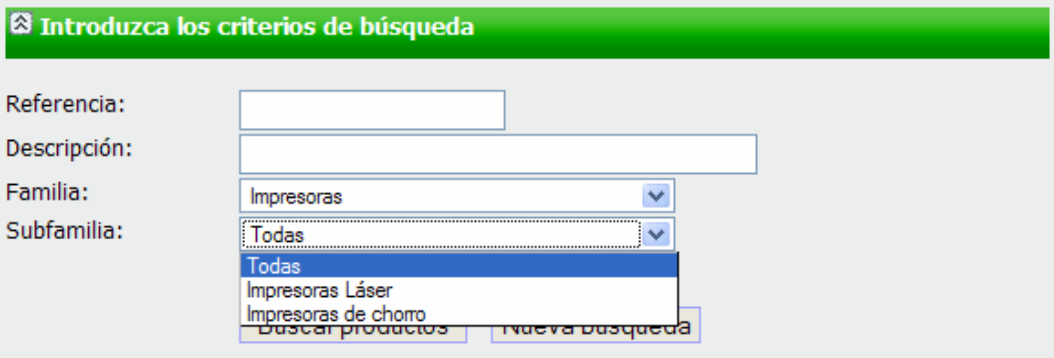


Ilustración 42. Desplegables enlazados en criterios de búsqueda

Para conseguir este comportamiento en el lado del cliente se han utilizado dos componentes de tipo *CascadingDropDown* asociados a los desplegables, y se han configurado con las direcciones del servicio web al que deben llamar para obtener sus datos, así como a los métodos a utilizar en cada caso.

En el servidor ha sido necesario crear un servicio web (ServicioWeb.asmx) donde se han implementado dos métodos, "ObtenerFamilias" y "ObtenerSubfamilias". El primer de ellos devuelve todas las familias de productos disponibles en el sistema, mientras que el segundo obtiene las subfamilias asociadas a la familia que le llega como parámetro.

La forma de implementar estos dos métodos es exactamente igual a un servicio web tradicional, salvo que ha sido necesario indicar con el atributo *[System.Web.Script.Services.ScriptService]* que el servicio será accedido desde scripts, y que cada método ha tener definido asimismo el atributo *[System.Web.Script.Services.ScriptMethod]*.

La comunicación entre cliente y servidor se realiza utilizando JSON, que es un formato más ligero y rápido de procesar que XML, basado en la notación de literal de objetos en lenguaje javascript. De esta forma, cuando el cliente selecciona una familia de productos, en lugar de utilizar una compleja y extensa petición SOAP, envía mediante POST al servidor lo siguiente:

```
{"knownCategoryValues":"Familia:MON;","category":"Subfamilia"}
```

Y la respuesta desde el servidor es la siguiente:

```
[{"__type":"AjaxControlToolkit.CascadingDropDownNameValue","name":"Monit
ores CRT", "value":"CRT", "isDefaultValue": false},  {"__type":
"AjaxControlToolkit.CascadingDropDownNameValue", "name": "Monitores de
Plasma", "value": "PLAS", "isDefaultValue": false}, {"__type":
"AjaxControlToolkit.CascadingDropDownNameValue","name":"Monitores TFT",
"value":"TFT","isDefaultValue":false}]
```

Podemos observar que es fácilmente legible, como XML, resulta menos pesado que su equivalente usando SOAP y, además, es evaluado de forma muy simple en cliente.

5.5.7.3. Autocompletador

El AJAX Control Toolkit para ASP.NET incluye un interesante control para la implementación de las llamadas "listas de autocompletar". Estas listas facilitan al usuario

la introducción de textos repetitivos, o lo asisten ofreciéndole la posibilidad de seleccionar una opción similar a la información que está introduciendo mediante el teclado.

La siguiente captura de pantalla muestra la lista de autocompletar: mientras el usuario teclea la palabra "moni", el sistema le muestra una serie de denominaciones de producto que son similares o contienen el texto introducido, lo que puede ahorrarle tiempo en su búsqueda.

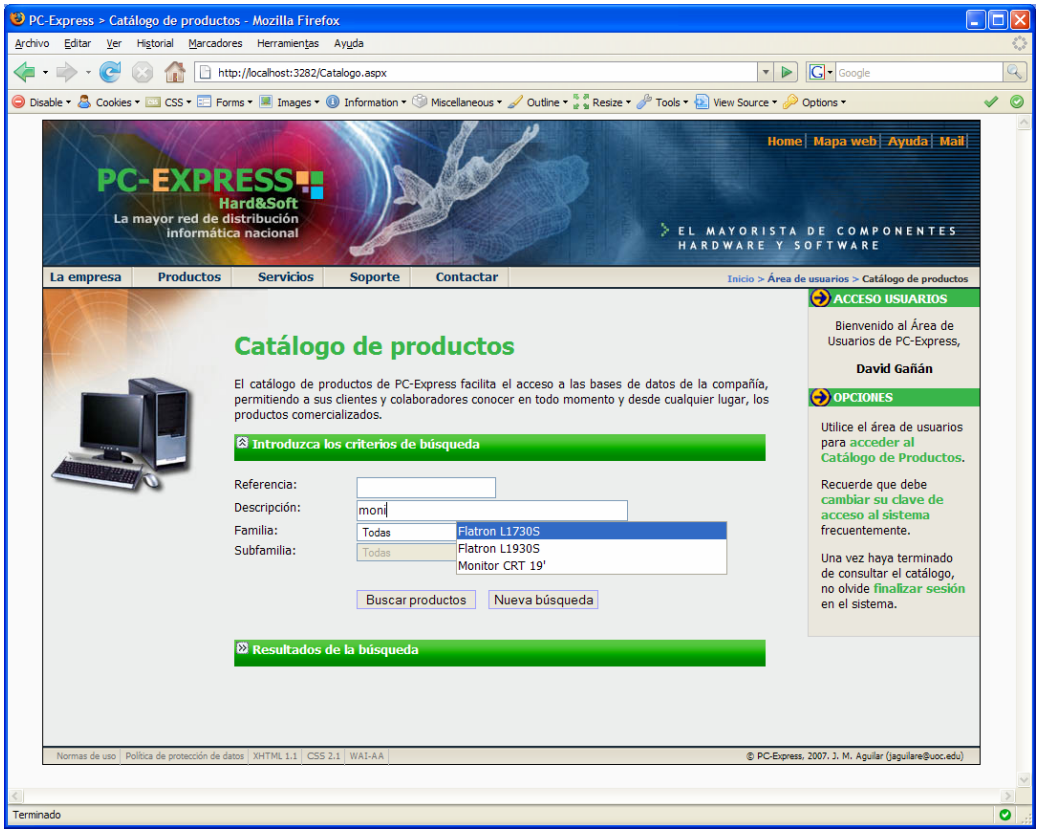


Ilustración 43. Autocompletador AJAX en acción

El control *AutoCompleteExtender*, una vez asociado al cuadro de texto en el que el usuario introduce la información y configurados ciertos parámetros, se encarga de llamar al servidor para obtener la lista de sugerencias en cada pulsación de tecla.

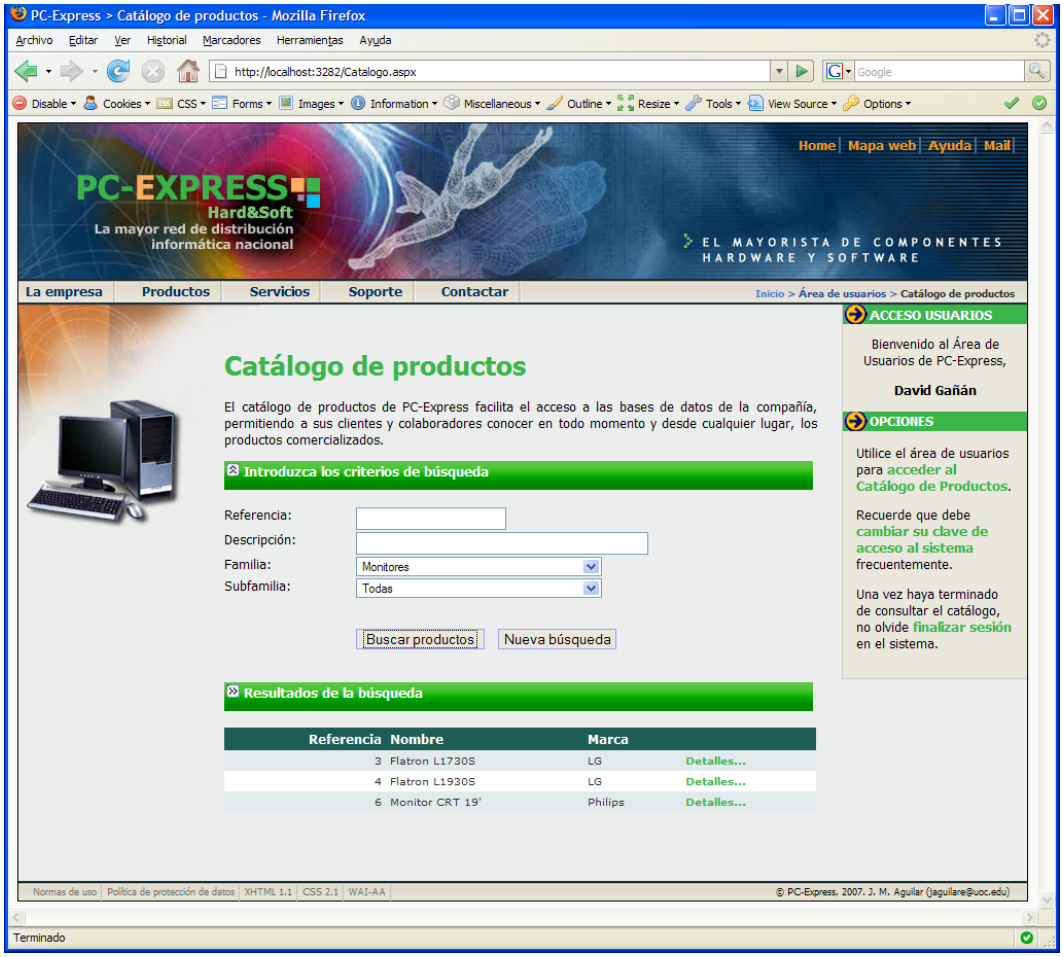
Para evitar búsquedas muy pesadas, es posible configurarlo para que realice peticiones sólo con textos a partir de una longitud determinada. CatalogoWEB utiliza un tamaño mínimo de cuatro caracteres para comenzar a solicitar datos de forma asíncrona.

En el servidor, un servicio web atenderá las solicitudes y devolverá las descripciones coincidentes, previa llamada a la central de servicios para obtenerlas.

5.5.7.4. Panel dinámico

Una de las características más llamativas del framework AJAX de Microsoft es la capacidad de aprovechar el renderizado parcial de páginas para actualizar contenidos de paneles sin necesidad de recargar páginas completas.

CatalogoWEB incluye esta característica para la presentación de los resultados de la búsqueda de productos en el catálogo de PC-Express. Una vez el usuario introduce los criterios y pulsa el botón "Buscar productos", el sistema llenará la rejilla de resultados sin recargar la página.



Es interesante destacar que este efecto de recarga parcial de una página se implementa con extrema sencillez gracias a la infraestructura provista por ASP.NET AJAX. Basta con incluir un control *UpdatePanel* y añadir en su interior todos aquellos controles y contenidos que puedan modificarse de forma conjunta. Será el framework el encargado de procesar los eventos y actualizar su contenido de forma independiente al resto de la página y sin producir postbacks de página completa.

Como aspecto negativo importante, es conveniente comentar en este punto que el uso de estas tecnologías de actualización parcial pueden resultar altamente perjudiciales desde el punto de vista de la usabilidad y accesibilidad de la página. Por ejemplo, el botón "Atrás" del navegador no se comportará de la forma prevista, ni será posible el uso de software lector de páginas debido a la actualización dinámica del contenido, entre otros aspectos.

Esto implica una dificultad añadida a la hora de establecer mecanismos de navegación sensibles al contexto como el botón de volver atrás disponible en la ficha de productos. La implementación habitual consistiría en crear un enlace de tipo javascript con una orden history.back(). Sin embargo, dado que la carga de la página anterior es dinámica, esta operación llevaría al usuario a una ficha de introducción de criterios de búsqueda vacía, sin los parámetros indicados y sin la lista de productos llena.

Para evitar este problema ha sido necesario propagar los criterios de búsqueda hasta la página de detalle del producto. Ésta toma los parámetros y los vuelve a enviar a la pantalla de búsqueda para que sus controles se inicialicen de forma correcta, incluido el gris con los resultados y la página del mismo donde se encontraba el usuario en el momento de ampliar los datos de un producto.

Además, el uso de AJAX en los desplegables en cascada (cascading dropdowns) hacen aún más complejo el tema, pues están preparados para ser recargados de forma autónoma atendiendo a las acciones del usuario, y no ha resultado sencillo establecerle valores por defecto.

5.5.8. Obtención de imágenes

Dado que las imágenes están almacenadas en la base de datos del servidor central, es absolutamente imposible acceder a ellas de forma directa desde el cliente (entendiendo como tal el equipo del usuario que acceder al sistema) sin romper la estructura de la solución, por lo que ha sido necesario establecer una pasarela entre ambos, de la misma forma que se hace con el resto de datos de los productos.

Además, la carga asíncrona de imágenes desde una página web hace necesaria la creación de un recurso específicamente diseñado para el envío de imágenes al cliente, cuya URL será especificada en la etiqueta imagen correspondiente para que el navegador del cliente pueda acceder a ella y mostrarla.

La imagen del producto está definida en la página *Detalles.aspx* de la siguiente forma:

```
<img  alt="Fotografía del producto"
      src="" id="imgImagen" runat="server" />
```

Podemos observar que se trata de un control de tipo imagen al que se tiene acceso como control de servidor. Esto se utiliza para, durante la carga de la página, establecer su atributo *src*, indicando la URL de la página que genera la imagen junto con el parámetro que indica la referencia del producto cuya fotografía se desea mostrar:

```
                // p es el producto consultado
imgImagen.Src = "ObtenerImagen.aspx?referencia=" + p.Referencia;
```

ObtenerImagen.aspx es una página ASP.NET especial: no devuelve un contenido textual, HTML ni XHTML, sino un contenido en binario de tipo image/jpeg. El contenido lo obtiene solicitando al servidor central la imagen del producto requerido a través del servicio web.

Se ha añadido además la capacidad de generar imágenes al vuelo para productos que no dispongan de una propia, para lo que se crea un Bitmap en memoria, se rellena el fondo con un degradado y se superpone el texto "Sin imagen":

```
bmp = new Bitmap(140, 140);
Graphics g = Graphics.FromImage(bmp);
LinearGradientBrush b =
      new System.Drawing.Drawing2D.LinearGradientBrush (
          new Rectangle(0, 0, 140, 140),
          Color.White,
          Color.FromArgb(210, 210, 210),
          (float)0.0);

g.FillRectangle(b, 0, 0, 140, 140);
g.DrawString("(Sin imagen)",
               new Font("Verdana", 8), Brushes.Black, 10, 40);
```

El resultado es el siguiente:

Ilustración 44. Imagen por defecto de un producto

5.5.9. Capturas de pantalla

Dado que a lo largo de la sección 5.5 se han ido incluyendo capturas de pantalla para mostrar distintas características del sistema, y para no dilatar en exceso el documento, simplemente se van a añadir algunas que no han aparecido hasta el momento.

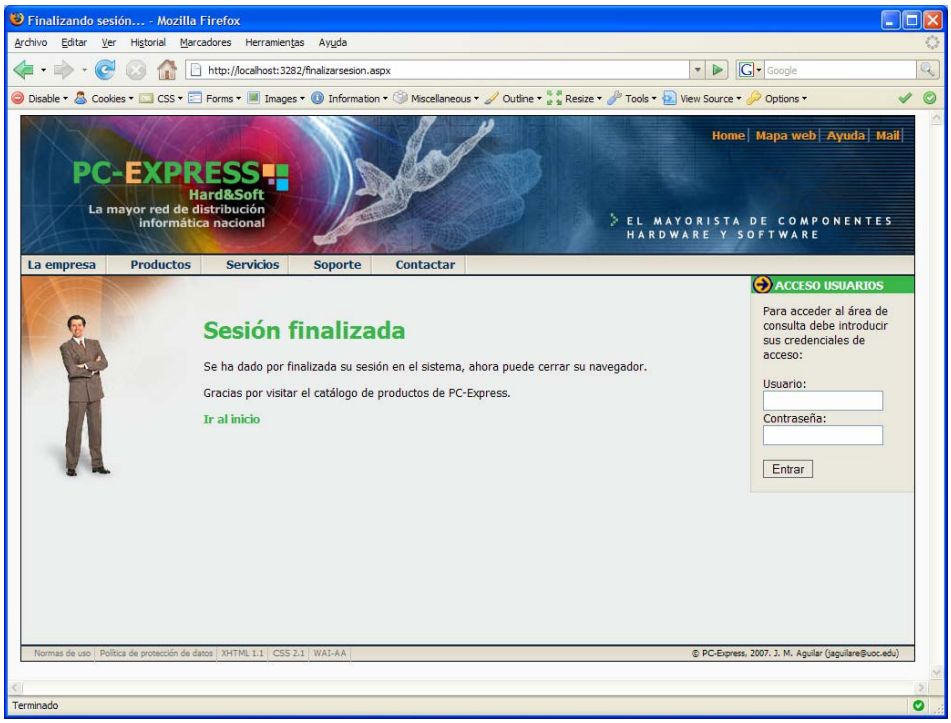


Ilustración 45. Pantalla de fin de sesión en CatalogoWEB

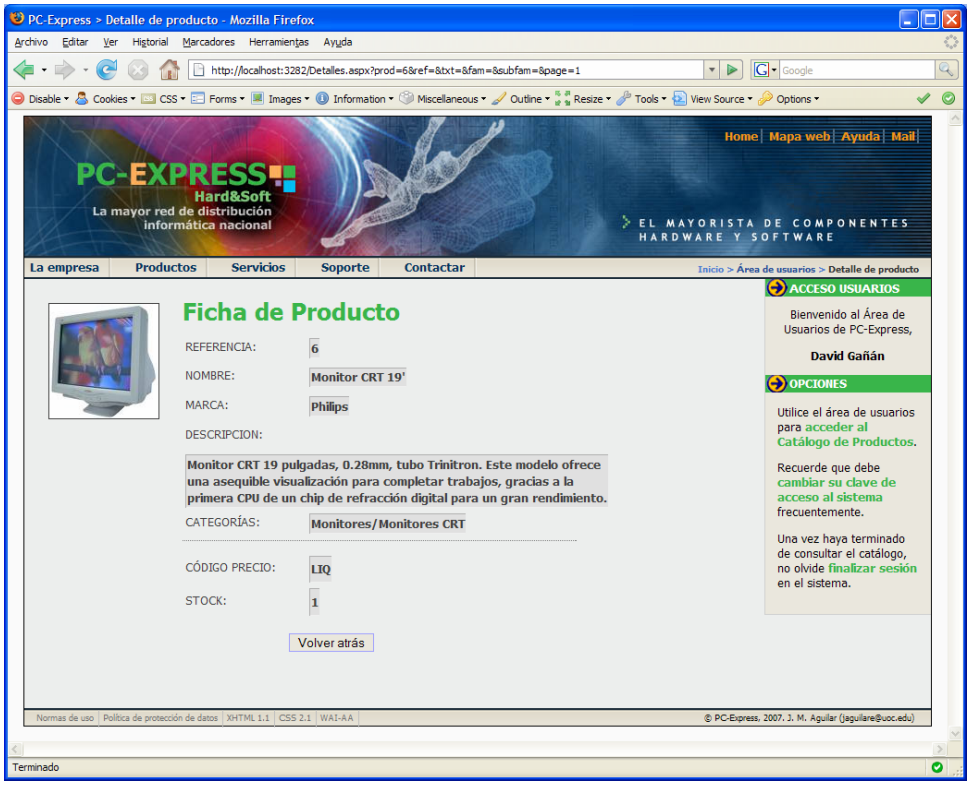


Ilustración 46. Ficha de productos en CatalogoWEB

6. Conclusiones

A lo largo del presente proyecto se ha implementado con éxito un sistema de gestión multicliente basado en arquitectura SOA y tecnologías basadas en la plataforma Microsoft.NET, lo que ha permitido conseguir los objetivos que fueron expuestos al comienzo de este documento.

El modelo arquitectural elegido (SOA) ha resultado enormemente válido y flexible para la implementación de servicios muy reutilizables, e invocables desde aplicaciones cliente de cualquier tipo. La orientación a mensajes y el uso transversal de las entidades de datos ha facilitado la integración y uso del ORM Hibernate como aproximación más natural a la persistencia de la información manejada por el sistema que el paradigma relacional.

Se han integrado con éxito tecnologías procedentes de distintas fuentes, tanto de Microsoft (ASP.NET AJAX, el propio framework, etc.) como procedentes del mundo del software libre (NHibernate, AJAX Control Toolkit), lo que demuestra el alto grado de interoperatividad existente entre ellas.

La metodología y planificación también han jugado un papel importante en el desarrollo del proyecto, permitiendo distribuir temporalmente distintas tareas a realizar desde el comienzo, y guiando en el cumplimiento de los hitos previstos.

Y, por último, desde el punto de vista personal, la experiencia ha sido enriquecedora, permitiendo la toma de contacto con un gran número de tecnologías de gran actualidad y amplia utilización en el mundo del desarrollo profesional de software.

7. Glosario

La siguiente tabla contiene términos, abreviaturas y acrónimos necesarios para la correcta comprensión del documento.

AJAX	Asynchronous Javascript And XML, conjunto de tecnologías utilizadas para mejorar la experiencia del usuario en la web basándose en el lenguaje javascript y el intercambio de mensajes XML con el servidor de forma asíncrona.
ASP.NET	Active Server Pages .NET, tecnología propia de la plataforma .NET para la realización de sitios web de contenido dinámico.
CSS	Cascading StyleSheets, hojas de estilo en cascada, lenguaje declarativo utilizado para la maquetación de contenidos (X)HTML de páginas web.
ECMA	European Computer Manufacturers Association (hoy conocida por ECMA International), asociación internacional dedicada a la elaboración de estándares de comunicación e información.
GUID	Globally Unique IDentifier, identificador único global, número de 16 bytes generado de forma aleatoria utilizado para identificar de forma única cualquier tipo de elemento.
HQL	Hibernate Query Language, lenguaje de consulta propio del motor ORM Hibernate.
HTTP	HyperText Transfer Protocolo, protocolo de transferencia hipertexto, utilizado principalmente para el acceso a contenido web desde navegadores.
IDE	Integrated Development Environment, entorno integrado de desarrollo de aplicaciones.
IIS	Internet Information Server, servidor web de Microsoft.
JSON	JavaScript Object Notation, es un formato ligero para el intercambio de datos.
MVC	Model-View-Controller, patrón de arquitectura software modelo-vista-controlador, que propone la separación de la presentación, lógica de control y de negocio en componentes independientes.
.NET	Plataforma de desarrollo y explotación de software, creada por Microsoft. Determinados componentes están, además, estandarizados por ECMA.
NHibernate	Implementación de Hibernate, el famoso ORM, para .NET.
ORM	Object-Relational Mapping, técnica que permite que objetos puedan persistir sobre bases de datos relacionales para posteriormente ser recuperados.
SGBD	Sistema Gestor de Bases de Datos, software especializado en las tareas de almacenamiento y recuperación de información, principalmente según el modelo relacional.
SOA	Service Oriented Architecture, Arquitectura Orientada a Servicios, paradigma de diseño de soluciones basadas en el desacoplamiento total entre los procesos de negocio, accesibles en forma de servicio, y las aplicaciones que consumen éstos.
SOAP	Simple Object Access Protocol, protocolo simple de acceso a objetos, estándar de la W3C ideado para el intercambio de información entre objetos remotos utilizando XML.
SQL	Structured Query Language, lenguaje estructurado de consulta utilizado para la manipulación de información en el seno de un SGBD.
SQL Server	SGBD de Microsoft orientado a soluciones profesionales.

UML	Unified Model Language, lenguaje unificado de modelado de sistemas de información y componentes.
URL	Uniform Resource Locator, localizalizador uniforme de recursos o dirección única de un elemento en Internet.
Visual Studio	Entorno de desarrollo específico para .NET. La versión actual es la 2005, aunque en este momento disponible en Beta la siguiente edición (codename "Orcas").
W3C	World Wide Web Consortium, organismo internacional que promueve el uso de estándares en los sistemas de información, y especialmente para la web.
WAI	Web Accesibility Initiative, grupo de trabajo de la W3C dedicado al impulso de estándares y buenas prácticas para facilitar la accesibilidad en la web.
WCAG	Web Contents Accesibility Guidelines, guía para el diseño de páginas web accesibles, propuesto por WAI.
WPF	Windows Presentation Foundation, infraestructura proporcionada por .NET Framework 3.0 para la creación e imlpementación de interfaces ricos de usuario.
WSDL	Web Service Description Language, lenguaje de descripción de servicios web basado en XML.
WSE	Web Service Enhancements, mejora sobre el estándar de servicios web que incluye aspectos como seguridad entre extremos y rutado de mensajes, entre otros.
XAML	Extensible Application Markup Language, Lenguaje extensible de marcado para aplicaciones utilizado para la composición de interfaces para WPF.
XHTML	Extensible HyperText Markup Language, lenguaje de marcas hipertexto extensible, especificación W3C de un lenguaje de descripción de contenidos para páginas web.
XML	Extensible Markup Language, lenguaje extensible de marcas multipropósito, muy utilizado para el intercambio de información entre sistemas heterogéneos.

8. Fuentes de información

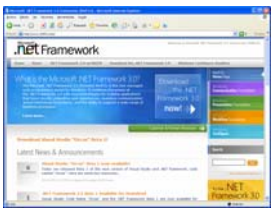
Para la realización del presente proyecto se han utilizado las fuentes de información que se relacionan a continuación.

A fondo C#
Editorial McGraw-Hill
Tom Archer

CSS Profesional
Hojas de estilo en cascada para el diseño web
Editorial Anaya Multimedia
Christopher Schmitt, Mark Trammell, Ethan Marcotte, Dunstan Orchad, Todd Dominey

Microsoft Developer Network
Web de desarrolladores de Microsoft
http://msdn2.microsoft.com/es-es/default.aspx

The Official Microsoft .NET AJAX Site
Sitio web oficial del framework AJAX de Microsoft
http://ajax.asp.net/



Microsoft .NET Framework 3.0 Community
Sitio web oficial de NetFx3
http://www.netfx3.com/

NHibernate - Hibernate for .NET
Sitio web oficial de NHibernate
http://www.nhibernate.org

World Wide Web Consortium
Oficina Española
http://www.w3c.es