Universitat Oberta de Catalunya

Proyecto Final de Carrera – Memoria

Desarrollo de una aplicación para la gestión y reserva online de acciones formativas con tecnología ASP.NET MVC 3

Miguel Fernández Cano

Junio de 2012

INDICE DE CONTENIDOS

I - INTRODUCCIÓN ____ 4

II – DESCRIPCIÓN DEL PROYECTO ____ 5

JUSTIFICACIÓN DEL PROYECTO ____ 5

DESCRIPCIÓN GENERAL ____ 6

ESTUDIO DE MERCADO ____ 8

OBJETIVOS DEL PROYECTO ____ 9

Objetivos Específicos ____ 9

REQUISITOS DE LA SOLUCION ____ 11

Requisitos Funcionales ____ 11

Requisitos No Funcionales ____ 12

FUNCIONALIDADES A IMPLEMENTAR ____ 13

RESULTADOS ESPERADOS ____ 15

PRODUCTO OBTENIDO ____ 15

PLANIFICACION INICIAL Vs PLANIFICACION FINAL ____ 16

III - ANALISIS Y DISEÑO ____ 18

ANALISIS DE REQUISITOS ____ 18

Requisitos Funcionales ____ 18

Requisitos No Funcionales ____ 22

ANALISIS DE CASOS DE USO ____ 24

Definición de casos de uso y actores ____ 24

Diagramas de casos de uso ____ 25

MODELO CONCEPTUAL ____ 29

DIAGRAMAS DE ARQUITECTURA ____ 31

Arquitectura física. Hardware ____ 31

Arquitectura lógica. Software ____ 32

DISEÑO DE LA BASE DE DATOS ____ 34

MODELO DE CLASES ____ 35

DISEÑO DE LA INTERFAZ ____ 36

COMENTARIOS Y CONCLUSIONES SOBRE EL DISEÑO ____ 41

IV - IMPLEMENTACIÓN _____ 46
SOFTWARE UTILIZADO _____ 46
CAPAS DE LA APLICACION _____ 47
V - EVALUACION DE COSTES _____ 48
Desarrollo de Software _____ 48
Hardware _____ 49
Otros costes _____ 49
VI - TRABAJO FUTURO _____ 50
VII - CONCLUSIONES _____ 52
VIII - BIBLIOGRAFIA _____ 54

I - INTRODUCCIÓN

Mediante el presente documento se expone la memoria final del proyecto "*Desarrollo de una aplicación para la gestión y reserva online de acciones formativas con tecnología ASP.NET MVC 3*", llevado a cabo por el alumno Miguel Fernández Cano como proyecto final de carrera, durante el segundo semestre del curso 2011-2012.

Después de evaluar las distintas propuestas para los trabajos de fin de carrera, promovidas por los consultores responsables del área de .NET, mostramos interés, inicialmente, en la propuesta basada en el desarrollo de una aplicación utilizando la tecnología LightSwitch.

Sin embargo, después de realizar una primera fase de búsqueda y documentación inicial sobre las posibilidades que aporta la plataforma .NET, y dentro de las múltiples opciones que ofrece, nos resultó interesante la tecnología ASP.NET y muy especialmente con la integración del nuevo patrón MVC.

Apostamos, de esta forma, por una propuesta personal para el desarrollo del proyecto final de carrera, que consiste, según los términos definidos en el presente documento, en el desarrollo de una aplicación para la gestión y reserva online de acciones formativas con tecnología ASP.NET MVC 3.

II – DESCRIPCIÓN DEL PROYECTO

A continuación, pasamos a detallar una descripción del proyecto realizado, exponiendo de forma clara y concisa la justificación del mismo, de tal forma que intentamos responder a la pregunta que cuestionaría por qué se ha elegido desarrollar este proyecto y no otro.

JUSTIFICACIÓN DEL PROYECTO

La plataforma .NET es un entorno de trabajo o FrameWork desarrollado por Microsoft desde principios del año 2000 y que ha ido evolucionando en las distintas versiones hasta ampliar significativamente su funcionalidad a los diversos ámbitos del desarrollo de software. Actualmente, se ha convertido en una de las principales plataformas de desarrollo tanto para escritorio como para entornos web y dispositivos móviles.

Por otra parte, el paradigma MVC (Modelo Vista Controlador) es un patrón de diseño de software que separa los elementos principales de una aplicación en tres componentes distintos, los datos de la aplicación, la interfaz de usuario, y la lógica de negocio, definiendo el modelo, la vista y el controlador respectivamente. MVC es un patrón de diseño bastante conocido, que data de principios de los años 80 y está ampliamente extendido en otros entornos de desarrollo (como en Java o Ruby on Raills).

Sin embargo, hasta hace muy poco no se había incluido este patrón en la plataforma .NET. Es precisamente en la última versión, con el Visual Studio 2010, cuando se introduce el paradigma MVC para el desarrollo de aplicaciones web.

Consideramos muy interesante este nuevo enfoque en la implementación de aplicaciones ASP.NET y creemos que será de gran aplicación en el mundo empresarial, ya que puede contribuir a una mejora sustancial de la productividad y fiabilidad en el desarrollo de software. Por este motivo hemos realizado la referida propuesta de proyecto final de carrera, que pasamos a desarrollar en los siguientes apartados.

En cuanto al dominio de nuestra aplicación, se ha planteado el negocio de la formación porque hemos considerado que es uno de los pocos sectores que, dentro del actual entorno socio económico, puede esquivar las adversidades y verse favorecido por un aumento en la demanda. En efecto, diversos estudios demuestran que la formación es uno de los activos más importantes, tanto a nivel personal como profesional, para afrontar situaciones económicamente negativas y en el contexto actual se configura como un sector en alza.

DESCRIPCIÓN GENERAL

El planteamiento que se ha propuesto para el proyecto es el desarrollo de una aplicación para la gestión y reserva online de acciones formativas (cursos, seminarios, masters, etc…). Se ha pensando en la problemática con la que se enfrenta una empresa o institución de tamaño medio, dedicada a impartir formación y que pretende impulsar el desarrollo del negocio mediante la implantación de una web corporativa que pretende fomentar la participación de los potenciales clientes, posibilitando la reserva online de los distintos cursos que se imparten.

Desde un punto de vista académico, el proyecto se dividirá en dos fases. En una primera fase, más teórica, se trata de realizar un estudio, análisis y evaluación de la tecnología ASP.NET MVC que quedará reflejado de forma adecuada a través de esta memoria, presentado los aspectos más significativos de la arquitectura en este paradigma de desarrollo y las conclusiones derivadas de su aplicación.

Por otro lado, y en una segunda fase, se concreta el estudio realizado, mediante la implementación de una aplicación específica utilizando esta tecnología.

Se contempla la posibilidad de añadir elementos de complejidad al proyecto mediante la inclusión, además del patrón MVC, de otras tecnologías ofrecidas en la plataforma .NET como Entity Framework y Linq para el acceso a los datos, así como la utilización de plantillas Razor, jQuery y Ajax para las interfaces. La persistencia necesaria se implementara mediante base de datos Sql Server 2008.

En cuanto al contenido, el aplicativo deberá mostrar distintas pantallas y funcionalidades dependiendo del perfil del usuario, por lo que se requiere implementar un sistema de autenticación que, en principio, distinguirá dos roles para cubrir los dos escenarios posibles.

Escenario A

Usuario principal: Cliente potencial.

Objetivo: Presentar las distintas acciones formativas definidas por la empresa, posibilitando y promoviendo la reserva de plazas para las mismas.

Descripción: El cliente objetivo de nuestra empresa, se acerca a la web para consultar y buscar información acerca de los distintos cursos que se imparten, y potencialmente reservar una o varias plazas para los mismos.

En este sentido el aplicativo se comporta básicamente como una tienda virtual que implementa las funciones habituales, como la búsqueda y selección de cursos, presentación de información de los mismos, cálculo del importe total, consulta de la selección realizada, confirmación de la reserva y registro de datos personales, en los términos que se detallarán posteriormente.

En este escenario, la parte de administración de acciones formativas queda oculta. El aplicativo ofrece la posibilidad de buscar las distintos cursos que serán clasificados según su contenido en áreas temáticas.

Se ofrece además, la posibilidad de hacer reservas online del curso o cursos en los que se desea participar, indicando en todo momento el precio de los mismos.

Escenario B

Usuario principal: Personal administrativo de la empresa de formación.

Objetivo: Gestionar las distintas acciones formativas ofrecidas por la empresa, realizando su correcta alta y clasificación.

Descripción: Nos dirigimos en este caso al empleado de la empresa de formación, que gestiona y administra las distintas acciones formativas.

La aplicación deberá permitir una gestión completa de los cursos incluyendo listados, altas, bajas y modificaciones de los mismos. Se deberá incluir, además, y de forma similar, una gestión completa de las áreas formativas para poder clasificar los diversos cursos y permitir asignar posteriormente, a cada curso, un área de conocimiento definida.

Las acciones formativas pueden ser de tres tipos, cursos, seminarios o Masters y además se clasificarán, como se ha indicado, por áreas temáticas (Informática, Administración, Idiomas, etc....). Los Masters están formados

por dos o más cursos, que se pueden superar de forma independiente o dentro del programa del propio Master.

ESTUDIO DE MERCADO

La actual coyuntura económica no sólo ha repercutido en el mercado laboral sino también en el ámbito de la formación y cualificación profesional. Uno de los primeros efectos observados, ha sido que la mayor demanda de formación ha girado sobre todo hacia unos contenidos más prácticos y especializados que fomenten, en un futuro, la empleabilidad de los alumnos.

Son muchas las empresas de formación que se dedican a impartir cursos, por tanto la especialización es la clave para incrementar la competitividad y el desarrollo del negocio. Además, por supuesto, de ofrecer un alto nivel de calidad en los contenidos.

Sin embargo, si nos acercamos a las webs comerciales de las principales firmas del sector en nuestro país, podemos comprobar que no son mayoría las que ofrecen la posibilidad de reservar online sus productos, por lo que se obliga al usuario a realizar un primer contacto, habitualmente telefónico, pudiendo limitar nuestro objetivo potencial de clientes.

En base a esta percepción, hemos considerado que la opción de realizar las reservas de los cursos online, puede promover el volumen de negocio y favorecer el acercamiento de clientes y por tanto, la competitividad de nuestra firma.

OBJETIVOS DEL PROYECTO

El objetivo principal del proyecto final de carrera, es utilizar los conocimientos adquiridos a lo largo de los estudios cursados, para que, a través de las distintas tecnologías que proporciona la plataforma .NET, poder desarrollar una aplicación de dificultad media, o bien hacer un estudio relacionado con alguna de estas tecnologías.

En este sentido la propuesta realizada supone una combinación de ambas opciones. En una primera fase, hemos tratado de realizar un estudio, análisis y evaluación de las posibilidades que ofrece la tecnología MVC dentro del entorno .NET.

Posteriormente, este conocimiento adquirido, se ha concretado mediante la implementación de una aplicación específica para desarrollar una solución a la problemática planteada, en nuestro caso, una web para la gestión y reserva online de acciones formativas.

Por tanto, el objetivo general de este trabajo final de carrera es adquirir un conocimiento apropiado de la tecnología ASP.NET MVC3, de forma que estemos en disposición de aplicar esta metodología para promover soluciones a nivel profesional, independientemente del contexto y del dominio de la aplicación que estemos desarrollando.

Objetivos Específicos

OBJ_01.- Tecnologías a utilizar

Uno de los objetivos clave del proyecto es acercarnos a las diversas tecnologías que ofrece la plataforma .NET para solventar las tareas de desarrollo habituales.

Por tanto, se ha intentado usar tantas tecnologías como ha sido posible, dentro de la plataforma .NET, y en el contexto del dominio de nuestro proyecto.

Aunque desde el punto de vista profesional la especialización se configura como un factor clave y decisivo a largo plazo, en el proyecto que nos ocupa

y siguiendo un enfoque más académico y educativo, hemos considerado más importante el conocimiento y uso práctico de diversas tecnologías, cuantas más mejor.

Así ante la disyuntiva entre utilizar varias opciones tecnológicas a un nivel medio aceptable o profundizar en alguna de ellas en concreto a costa de sacrificar otras, hemos elegido siempre la primera opción.

Dicho esto, nos proponemos la aplicación con mayor o menor profundidad de las siguientes tecnologías y herramientas:

- Entorno de desarrollo: Visual Studio 2010
- Arquitectura de la aplicación: patrón de diseño MVC
- Gestor de Base de Datos: Sql Server 2008 R2
- Acceso a datos: Entity Framework
- Lenguaje de consultas: LinQ to Entities
- Diseño de interfaces: plantillas con Razor y hojas de estilos CSS
- Interacción entre interfaces: AJAX y JQuery

OBJ_02.- Contexto planteado y dominio de la aplicación

A nivel funcional, el objetivo del proyecto ha sido proponer una solución para el desarrollo de una web que implemente una plataforma de reservas online similar a una tienda virtual.

Se ha de facilitar, además, la gestión de las distintas entidades que intervienen en el proceso, en función del tipo de usuario que esté interactuando con el sistema.

OBJ_03.- Documentación, memoria y presentación

Tenemos que destacar que se ha planteado obtener, como resultado de todo el proceso de estudio, una adecuada documentación del proyecto, tanto a nivel funcional como de la arquitectura del mismo. Prestaremos especial atención a la redacción del texto, su coherencia y estructura.

Para finalizar, nos proponemos como último objetivo, la elaboración de una correcta presentación que permita transmitir la información y el aprendizaje adquirido en el desarrollo del proyecto, de una forma amena y eficiente.

REQUISITOS DE LA SOLUCION

Pasamos a definir, a continuación, los requisitos de la solución desarrollada, tanto a nivel funcional como no funcional. Este conjunto de requisitos se configura como la base o input que se utilizará para la fase de diseño del producto a implementar.

Requisitos Funcionales

Req_F01.- Se implementará una gestión completa de los cursos que incluirá las funciones habituales de listado, altas, bajas y modificación de los mismos.

Req_F02.- Los cursos deberán poder ser clasificados en distintas áreas temáticas, en función de su contenido.

Req_F03.- Es necesario desarrollar un modulo para la gestión de las áreas temáticas mencionadas, que deberá incluir un listado y las funciones para realizar altas, bajas y modificaciones.

Req_F04.- Los cursos podrán clasificarse, además, en función de su duración y complejidad en tres tipos definidos: Seminarios, Masters y Cursos estándar.

Req_F05.- Un Master podrá estar formado por varios cursos, a los que se puede acceder de forma independiente, o bien dentro del programa del propio Master.

Req_F06.- Es necesario implementar un modulo para la gestión básica de usuarios, que controle el acceso al sistema.

Req_F07.- El acceso se gestionará mediante autorización basada en perfiles o roles de usuario. Se definen dos roles: Usuario Administrador y Usuario estándar.

Req_F08.- El acceso a las pantallas de gestión de los cursos y las áreas temáticas estará restringido, según el rol del usuario, y será accesible solo con el perfil Administrador.

Req_F09.- Para el acceso con un perfil estándar (no administrador) se desarrollará, inicialmente, una interfaz que permita:

- Buscar y navegar entre los distintos cursos definidos previamente en el sistema.
- Seleccionar y mostrar el detalle con los datos de cada curso.
- Realizar la reserva de plazas para diferentes cursos.

Req_F10.- El aplicativo ha de contemplar la opción de reservar, dentro de la misma sesión, plazas para diferentes cursos e incluso varias plazas para un mismo curso.

Req_F11.- Tanto la búsqueda como la selección de los cursos se podrán realizar sin necesidad de que el usuario esté logado en el sistema.

Req_F12.- En el momento de confirmar la reserva, si se pedirá que el usuario esté registrado en la aplicación. Se solicitará, además, los datos personales y el número de cuenta bancaria donde se cargará el importe correspondiente.

Req_F13.- Una vez confirmada la reserva se asociará un numero identificativo de la misma, quedando ésta registrada de forma permanente.

Req_F14.- La propia selección de cursos también deberá tener persistencia, aunque limitada temporalmente, puesto que el usuario espera que se mantengan los cursos seleccionados en la interfaz, entre una visita y otra. En todo caso, una vez confirmada la reserva, ya no tendrá sentido mantener esta selección.

Req_F15.- Desde la interfaz, el usuario podrá tener acceso en todo momento a la selección de cursos realizada, mostrando además el número de cursos añadidos.

Req_F16.- A modo de sugerencia se mostrará, en la pantalla inicial de la aplicación, un listado con los cinco cursos más demandados, permitiendo acceder a sus datos directamente.

Requisitos No Funcionales

Req_NF01.- La aplicación se implementará mediante una arquitectura distribuida con un servidor web central y una interfaz que permita el acceso remoto de clientes a través del navegador por http.

Req_NF02.- Para la persistencia de los datos se utilizará una base de datos centralizada e implementada mediante el gestor MS SQL Server 2008.

Req_NF03.- El sistema se desarrollará utilizando la tecnología .NET de Microsoft en su última versión, con el Framework 4.0.

Req_NF04.- Se seguirá una arquitectura distribuida y un diseño definido en capas, aplicando el patrón MVC, Model-View-Controller.

Req_NF05.- La interfaz de usuario ha de ser suficientemente sencilla e intuitiva para permitir una navegación cómoda y atractiva incluso para usuarios poco experimentados.

Req_NF06.- Debido a la proyección internacional de la firma y en base a conseguir una estandarización global, se define el inglés como el idioma más apropiado para la interfaz del aplicativo.

FUNCIONALIDADES A IMPLEMENTAR

En base a los requerimientos de la solución definidos previamente, pasamos a detallar la funcionalidad que se ha de desarrollar para su correcta consecución.

F01.- Gestión de cursos

Según se ha definido en los requisitos funcionales, se implementará una gestión completa de los cursos que incluirá las siguientes funciones:

- Listado completo de cursos, en los que se mostrará toda la información correspondiente, así como el tipo y área formativa a la que pertenecen.
- Alta de un nuevo curso en el sistema.
- Edición de los cursos existentes en la aplicación.
- Borrado de cursos.

F02.- Gestión de áreas formativas

Se desarrollará un modulo para la gestión de las áreas formativas, que implemente las siguientes funciones:

- Listado de áreas formativas
- Alta de un área nueva.
- Edición de un área existente en el sistema.
- Borrado de áreas formativas.

F03.- Control de usuarios. Autorización y Registro.

El sistema deberá implementar una funcionalidad que permita el control del acceso a los usuarios mediante autorización basada en perfiles o roles.

Un usuario deberá poder registrarse en el sistema y la aplicación deberá guardar sus datos de registro, básicamente login y password, para posteriores sesiones.

Se definirá un perfil administrador y un usuario con ese perfil que será el único capaz de acceder a las funciones de gestión de cursos y áreas formativas.

Los usuarios sin perfil administrador, podrán utilizar el resto de funcionalidad del la web y solo se les pedirá el registro en el momento de confirmar la reserva de un curso previamente seleccionado.

F04.- Búsqueda y selección de cursos

La aplicación implementará la funcionalidad básica de una tienda virtual. Así un usuario que acceda a la web podrá realizar las siguientes funciones:

- Navegar y buscar entre los distintos cursos definidos previamente en el sistema.
- Acceder a los cursos clasificados por áreas formativas.
- Consultar la información referente a un curso en concreto.
- Seleccionar un curso para reservarlo posteriormente.
- Eliminar elementos seleccionados previamente.
- Mostrar el detalle de los cursos seleccionados con el importe total.
- Guardar la selección de cursos, para un mismo usuario, entre diferentes sesiones (previo registro del usuario).

F05.- Reserva de cursos

En un escenario de uso estándar para un usuario no administrador, la reserva de cursos, confirmando la selección realizada, culmina el proceso y la interacción con el sistema.

Así pues se implementan las siguientes funciones:

- Validación de listado de cursos seleccionados.
- Autenticación del usuario. Para poder continuar con el proceso, se pedirá que el usuario haga login en la aplicación. Si el usuario no está dado de alta previamente en el sistema, se le enviará al modulo correspondiente para que se registre. (Ver F03).
- Petición de los datos personales del usuario, así como de los datos bancarios para formalizar la reserva.
- Asignación de un numero identificativo que confirma la reserva.

RESULTADOS ESPERADOS

Antes de comenzar el desarrollo del proyecto se espera, lógicamente, que cumpla con los requisitos, tanto funcionales como no funcionales, definidos inicialmente y reflejados en el presente documento.

Se espera de forma especial, configurando el objetivo principal del trabajo realizado, adquirir el conocimiento y la base, tanto teórica como práctica, que nos permita implementar con éxito diversas soluciones de software ante problemáticas planteadas en diferentes ámbitos y dominios.

En cuanto al producto propiamente dicho, esperamos poder desarrollar una web atractiva, sencilla y funcional que contribuya a la promoción y el desarrollo del negocio de nuestra hipotética empresa de formación FormaSelect.

PRODUCTO OBTENIDO

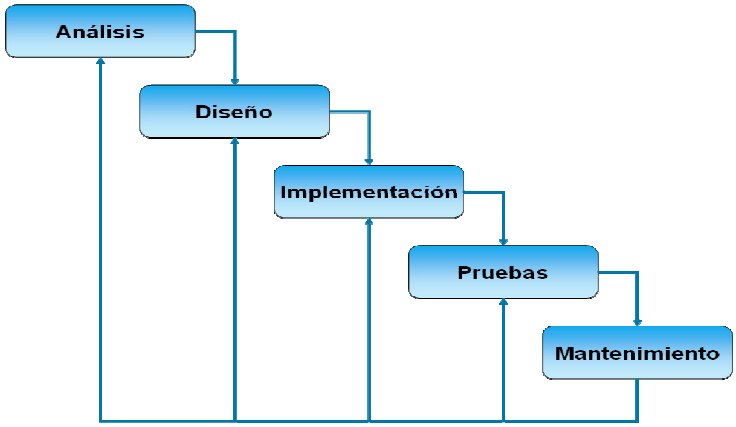
El producto obtenido constituye el software de aplicación necesario para el despliegue de una web comercial para la presentación y reserva online de cursos, en los términos que se han establecido en esta memoria.

Concretamente disponemos de una aplicación, preparada para ser puesta en producción, y desarrollada con tecnología .NET en la última versión del entorno, Framework 4.0, así como una base de datos diseñada y preparada para almacenar la información gestionada por la aplicación.

El producto de software desarrollado se complementa con la presente memoria, de forma que quede documentada, adecuadamente, la definición y el diseño del proyecto.

PLANIFICACION INICIAL Vs PLANIFICACION FINAL

Mostramos, seguidamente, una posible planificación inicial del proyecto, descomponiendo el desarrollo del mismo en las distintas fases habituales que constituyen el clásico modelo en cascada del ciclo de vida del software. Se realiza, además, una descripción y una estimación del tiempo necesario para cada una de las fases en el presente proyecto.



Fase1.- Análisis

En esta fase se analizan las necesidades de los potenciales usuarios del software para determinar qué objetivos debe cubrir, se describe el proyecto y se define la funcionalidad que se va a implementar. Se ha de elaborar el documento correspondiente a la especificación de requisitos, de forma parecida al modelo que se ha presentado en la primera parte de esta memoria. Esta tarea debe realizarse al inicio del proyecto y su duración estimada es de una semana y media.

Fase2.- Diseño

En esta fase se trata de definir las características de diseño, arquitectura y estructura de la aplicación. Se descompone y organiza el sistema en elementos que puedan elaborarse por separado, se define la estructura relacional global de los diferentes módulos del software, se define el mapa conceptual, el diagrama de entidades, modelo de datos, casos de uso y demás diagramas utilizados habitualmente aplicando las técnicas del UML.

La documentación obtenida se corresponde con la segunda parte de la presente memoria, y se estima una duración de 2 a 3 semanas.

Fase3.- Implementación

En esta fase se implementan los algoritmos necesarios y se programa el código fuente de la aplicación siguiendo los patrones, la arquitectura y el diseño definido en la fase anterior.

Esta fase se corresponde, propiamente, con la programación del software y el producto obtenido es la misma aplicación, para la que estimamos una duración de entre 6 y 8 semanas.

Fase4.- Pruebas

Antes de la puesta en producción del software, es importante realizar tantas pruebas como sea necesario para asegurar la calidad y garantizar la fiabilidad y coherencia de nuestro producto.

Se suele distinguir entre pruebas unitarias y de integración y como resultado se realiza un informe para chequear si el producto ha pasado o no, los distintos juegos de pruebas ejecutados, con la finalidad de corregir posteriormente los errores que se hayan podido detectar.

Fase5.- Mantenimiento

Estamos ante una fase post producción, es decir que se realiza después de la puesta en producción del sistema y normalmente durante toda la vida útil del mismo. Es una de las etapas más críticas, ya que diversos estudios han estimado que es la etapa en la que más recursos se consumen.

La utilización de técnicas que favorezcan la reutilización y estandarización del software, así como su modularidad durante la fase de implementación, jugarán aquí un papel capital en el coste de mantenimiento.

Podemos distinguir entre mantenimientos correctivos y evolutivos, y su duración y complejidad dependerán de la calidad inicial del software, de la vida útil esperada y de la evolución estimada para el mismo.

Tenemos que señalar que esta planificación se ha estimado de forma orientativa, intentando definir unos tiempos y plazos razonables para la consecución de las distintas tareas. Sin embargo, por diversas circunstancias, la planificación y la ejecución real del proyecto dista mucho de los tiempos y plazos que se ha expuesto anteriormente.

III - ANALISIS Y DISEÑO

En esta parte de la memoria vamos a presentar la documentación que se corresponde con las fases de análisis y diseño, según se ha indicado en la planificación del ciclo de vida del software.

Analizaremos los requerimientos funcionales y no funcionales partiendo de la definición presentada, y mostraremos los diversos diagramas utilizados para el diseño funcional y estructural, siguiendo el paradigma del UML.

Por último, presentaremos el modelo de datos, los diagramas relacionales del diseño de la base de datos y un primer diseño de las interfaces de nuestra aplicación web.

ANALISIS DE REQUISITOS

Analizaremos seguidamente los requerimientos funcionales y no funcionales partiendo de la definición presentada en el bloque anterior, correspondiente a la descripción del proyecto. En caso de ser necesario, se añadirán requisitos detectados en esta fase o se ampliarán a partir de los anteriores.

Requisitos Funcionales

Req_F01.- Gestión de cursos.

Se implementará una gestión completa de los cursos que incluirá las funciones habituales de listado, altas, bajas y modificación de los mismos, por tanto podemos definir los siguientes requisitos:

F01_1.- Implementación de un listado que muestre los cursos dados de alta en el sistema.

F01_2.- Pantalla para dar de alta un nuevo curso. En esta pantalla se informaran los datos correspondientes.

F01_3.- Pantalla que permita la modificación de un curso.

F01_4.- Permitir la selección del tipo de curso y área funcional desde las pantallas de creación y edición de cursos.

F01_5.- Se implementará una función para permitir el borrado de cursos.

F01_6.- Se pedirá confirmación en el borrado de los cursos.

Req_F02.- Los cursos deberán poder ser clasificados en distintas áreas temáticas, en función de su contenido.

Según se ha indicado en el requisito F01_4, la asociación entre el curso y el área funcional se permitirá desde las pantallas de creación y edición de cursos.

Hay que considerar, a efectos del diseño del modelo de datos, que un curso ha de estar asociado a una y solo una área temática, aunque lógicamente, un área temática puede estar formada por uno o varios cursos. Se define de este modo una relación 1-n entre área y curso.

Req_F03.- Gestión de áreas funcionales.

Es necesario desarrollar un modulo para la gestión de las áreas temáticas mencionadas, que deberá incluir un listado y las funciones para realizar altas, bajas y modificaciones. Podemos descomponer los siguientes requerimientos funcionales:

F03_1.- Se implementará un listado que muestre las distintas áreas formativas dadas de alta en el sistema.

F03_2.- Pantalla para dar de alta una nueva área. En esta pantalla se informaran los datos correspondientes, básicamente nombre y descripción.

F03_3.- Pantalla que permita la modificación de un área.

F03_4.- Se implementará una función para permitir el borrado de un área formativa seleccionada.

F03_5.- Se pedirá confirmación en el borrado de los áreas.

Req_F04.- Tipos de cursos.

Los cursos podrán clasificarse, además, en función de su duración y complejidad en tres tipos definidos: Seminarios, Masters y Cursos estándar.

Esta especificación de los tipos de cursos está definida y se cargara inicialmente en la base de datos. No se contempla la posibilidad de que

estos tipos puedan variar con el tiempo, por lo que es innecesario un módulo para su gestión.

Req_F05.- Un Master podrá estar formado por varios cursos, a los que se puede acceder de forma independiente, o bien dentro del programa del propio Master.

Esto es muy relevante, a efectos de diseño, ya que ha de tenerse en cuenta que un curso de tipo Master, puede estar compuesto por otros cursos.

En este caso, se nos plantean varias opciones para el diseño de la persistencia. Podemos optar por utilizar dos entidades diferenciadas una para cursos y otra para masters pero, atendiendo al fundamento original de que un Master es un tipo de curso, tomamos la decisión de implementarlo con una única entidad.

Por consiguiente, la entidad curso deberá implementar una relación reflexiva (ver modelo de datos de la BD) para permitir cursos formados por otros cursos.

Req_F06 y **Req_F07**.- Gestión de usuarios.

Es necesario implementar un modulo para la gestión básica de usuarios, que controle el acceso al sistema. Dicho acceso se gestionará mediante autorización basada en perfiles o roles de usuario, definiendo dos roles: Usuario Administrador y Usuario estándar.

Después de estudiar y analizar varias opciones para la implementación de un modulo de gestión de usuarios con estas características, se considera la utilización de la herramienta de configuración de aplicaciones para ASP.NET, porque cumple totalmente con nuestros requerimientos y se adapta a la perfección a una autenticación de usuarios basada en roles, como se detallara en los modelos posteriormente.

Req_F08.- El acceso a las pantallas de gestión de los cursos y las áreas temáticas estará restringido, según el rol del usuario, y será accesible solo con el perfil Administrador.

Una vez definidos los usuarios y los roles de cada usuario, aplicando los modelos de autenticación y registro de usuarios de .NET, configuramos un filtro para impedir el acceso a estas pantallas.

Cuando un usuario intenta acceder por primera vez, se pedirá que se registre en el sistema y se comprobará si su perfil es el adecuado y en caso contrario se restringe el acceso.

Req_F09.- Selección de cursos

Para el acceso con un perfil estándar (no administrador) se desarrollará, inicialmente, una interfaz que permita las siguientes funcionalidades:

F09_1.- Búsqueda y navegación entre los distintos cursos definidos previamente en el sistema y clasificados por área.

F09_2.- Selección de un curso concreto para mostrar todos sus datos con detalle, nombre, descripción, precio, plazas disponibles, etc…

F09_3.- Selección de un curso para guardarlo en nuestra cuenta virtual, como pendiente de reservar.

F09_4.- Eliminar un curso de una selección realizada previamente.

Req_F10.- El aplicativo ha de contemplar la opción de reservar, dentro de la misma sesión, plazas para diferentes cursos e incluso varias plazas para un mismo curso.

Se incluirá un atributo para la cantidad, dentro de la entidad carro, encargada de guardar la selección del usuario, previa a la reserva.

Req_F11.- Tanto la búsqueda como la selección de los cursos se podrán realizar sin necesidad de que el usuario esté logado en el sistema.

Req_F12 y **Req_F13**.- Gestión de reservas.

Una vez seleccionado el curso o los cursos adecuados, el usuario podrá pasar a confirmar la reserva de plaza.

En el momento de confirmar la reserva, se pedirá que el usuario esté registrado previamente en la aplicación. Si no está registrado, se le enviará a la pantalla correspondiente para dar de alta un nuevo usuario.

Se solicitará, además, los datos personales y el número de cuenta bancaria donde se cargará el importe correspondiente.

Una vez confirmada la reserva se asociará un numero identificativo de la misma, quedando ésta registrada de forma permanente. Este número será mostrado al usuario junto con el mensaje de confirmación.

Req_F14 y **Req_F15**.- Gestión del carro.

La propia selección de cursos también deberá tener persistencia, aunque limitada temporalmente, puesto que el usuario espera que se mantengan los cursos seleccionados en la interfaz, entre una visita y otra. En todo caso, una vez confirmada la reserva, ya no tendrá sentido mantener esta selección.

Para implementar este requisito funcional se ha de diseñar una entidad para guardar la selección de cursos realizada por el usuario previamente a la confirmación de la reserva. Esta entidad, que denominamos carro, se dotara de persistencia definiendo así la correspondiente tabla en la base de datos, como se mostrará en el modelo de entidades posteriormente.

Desde la interfaz, el usuario podrá tener acceso en todo momento a la selección de cursos realizada, mostrando además el número de cursos añadidos. Se podrá añadir elementos o borrar alguno de los añadidos previamente.

Req_F16.- Lista de cursos más populares

A modo de sugerencia se mostrará, en la pantalla inicial de la aplicación, un listado con los cinco cursos más demandados, permitiendo acceder a sus datos directamente.

Se entiende por cursos más demandados, los cinco primeros sobre los que se hayan efectuado más reservas confirmadas, no solo seleccionadas.

Requisitos No Funcionales

Req_NF01.- Arquitectura distribuida y acceso web.

La aplicación se implementará mediante una arquitectura distribuida con un servidor web central y una interfaz que permita el acceso remoto de clientes a través del navegador por http.

Req_NF02.- Requerimientos de base de datos.

Para la persistencia de los datos se utilizará una base de datos centralizada e implementada mediante el gestor MS SQL Server 2008.

Se considera la opción de usar dos bases de datos separadas, una para las entidades propias del dominio de la aplicación y otra para la gestión de usuarios que usará el modelo definido en la plataforma .NET.

Req_NF03.- Entorno tecnológico de desarrollo.

El sistema se desarrollará utilizando la tecnología .NET de Microsoft en su última versión, con el Framework 4.0. Principalmente, se usarán las siguientes herramientas:

- Visual Studio 2010
- SQL Server 2008 R2

- MS Visio 2007

Req_NF04.- Arquitectura de diseño.

Se seguirá una arquitectura distribuida y un diseño definido en capas, aplicando el patrón MVC, Model-View-Controller.

Model-View-Controller (MVC) es un patrón de arquitectura de software que separa de forma limpia y clara los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos.

Req_NF05.- Interfaz de usuario.

La interfaz de usuario ha de ser suficientemente sencilla e intuitiva para permitir una navegación cómoda y atractiva incluso para usuarios poco experimentados.

El concepto de usabilidad será clave en este apartado, para desarrollar una buena interfaz. Podemos definir la usabilidad como la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto.

En términos de ingeniería de software e interacción persona-ordenador, la usabilidad se refiere a la claridad y la elegancia con que se diseña la interacción con un programa de ordenador o un sitio web.

Por tanto, es importante tener siempre presente, pero especialmente en la fase de diseño, que el objetivo final de la interfaz es atraer y retener a potenciales clientes, aportando una experiencia de uso cómoda, atractiva y amigable para promover las reservas y favorecer la imagen de marca de la empresa, FormaSelect.

Req_NF06.- Lenguaje de la aplicación.

Debido a la proyección internacional de la firma y en base a conseguir una estandarización global, se define el inglés como el idioma más apropiado para la interfaz del aplicativo.

Por otro lado, debemos tener en cuenta que durante la fase de implementación y desarrollo se han utilizado algunas técnicas que emplean diversas plantillas de software, cuyo idioma nativo es, lógicamente el inglés.

Por todo esto, hemos considerado que un desarrollo completo en este idioma quedará más estandarizado, especialmente si entendemos que el inglés se ha convertido en *lingua franca* en el mundo del desarrollo de software.

ANALISIS DE CASOS DE USO

Presentamos, a continuación, un análisis de los casos de uso que se observan en la operativa y explotación del sistema de software diseñado.

Definición de casos de uso y actores

A partir de los requerimientos definidos y analizados anteriormente, y teniendo en cuenta especialmente los requisitos funcionales, vamos a estudiar los casos de uso para definir de forma adecuada los procesos y la interacción de los usuarios del aplicativo con el propio sistema.

Tenemos que destacar que, básicamente, nos enfrentamos a dos escenarios posibles. Por una parte tenemos al usuario estándar, que accede a nuestra web para buscar, seleccionar y opcionalmente reservar nuestros cursos. Por otro lado, tendremos que especificar la interacción del usuario con perfil de administrador, que será el encargado de la gestión de las distintas entidades definidas en el sistema, concretamente los cursos y las áreas funcionales.

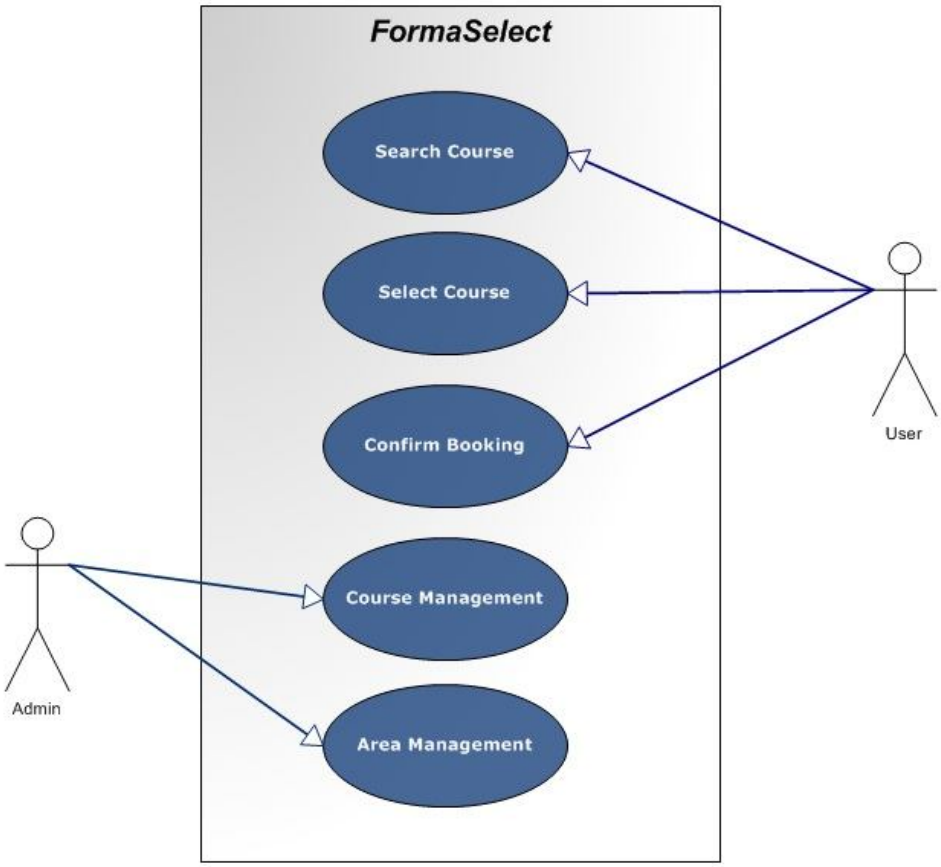
Así pues tendremos dos actores principales, que denominamos Usuario y Administrador (User y Admin).

A01.- User. Para el actor User se contemplan los siguientes casos de uso principales: Search Course, Select Course y Confirm Booking.

A02.- Admin. Para el actor Admin definimos dos casos de uso principales Course Management y Area Management.

A continuación mostramos los diagramas para los distintos casos de uso, realizando un análisis, descripción y si es necesario presentamos un detalle y descomposición de algunos de ellos.

Diagramas de casos de uso

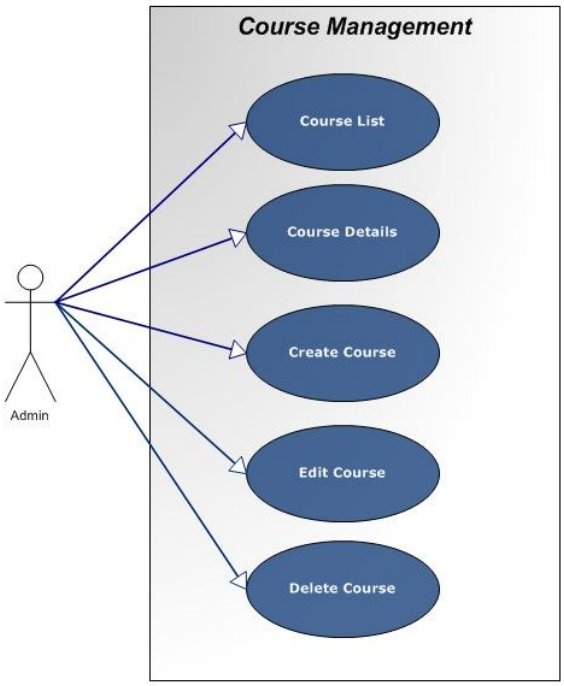


CU01.- Course Management

Actor: Admin

Descripción: Se trata de la gestión completa de los cursos en el sistema. Llegando a un mayor nivel de detalle, podemos descomponerlo en cinco casos de uso, como se indica en el siguiente diagrama, Course List, Course Details, Create Course, Edit Course, Delete Course.

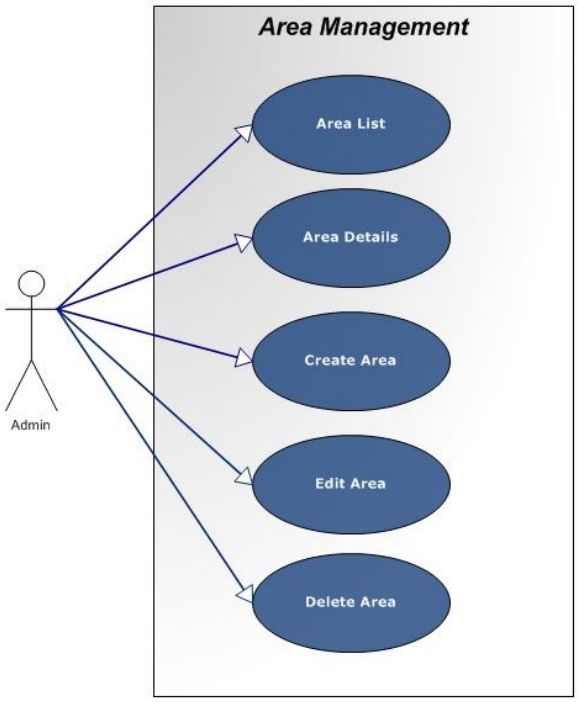
Entendemos que no es necesario explicar con más detalle la operativa de estas acciones, porque se trata de las operaciones CRUD habituales en un proceso de gestión de la entidad, curso en este caso.



CU02.- Area Management

Actor: Admin

Descripción: Se trata de la gestión completa de las áreas funcionales en el sistema. Si queremos llegar a un mayor nivel de detalle, podemos descomponerlo en cinco casos de uso, tal y como se indica:



CU03.- Search Course

Actor: User

Descripción: Este caso de uso se origina cuando el usuario accede a la aplicación para buscar o consultar los cursos dados de alta en el sistema. El flujo habitual de las acciones sería:

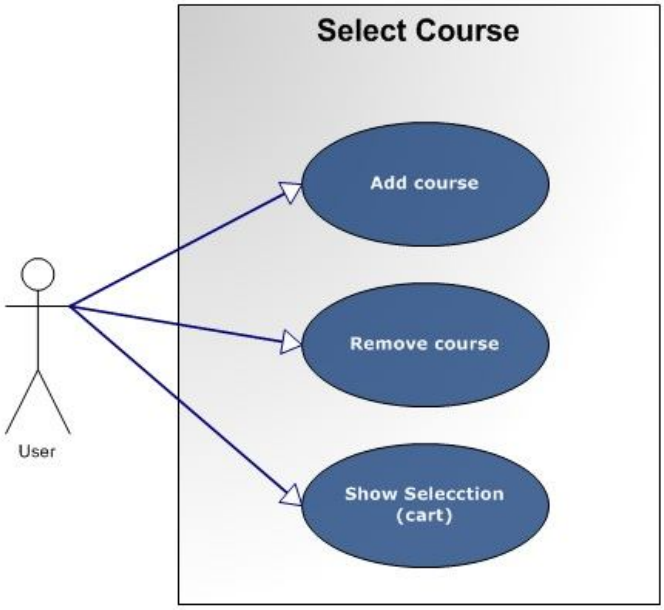
1- El usuario accede a la interfaz web.
2- El usuario busca y navega entre los distintos cursos definidos en el sistema.
3- Acceso a los cursos clasificados por áreas formativas (se mostrarán las áreas formativas mediante un menú con sus correspondientes enlaces).
4- El usuario consulta la información relevante del curso.

CU04.- Select Course

Actor: User

Descripción: Consiste en la selección de un curso concreto, una vez que el usuario lo ha buscado y ha accedido a sus datos. Debemos contemplar también la posibilidad de que el usuario elimine un curso de la selección realizada previamente, por lo que podemos descomponer este caso de uso atendiendo a estas posibles acciones.

Contemplamos además que en todo momento el usuario puede consultar los cursos que guarda en su selección, en una entidad que denominamos carro, definiendo con esta consulta, otra posible acción.

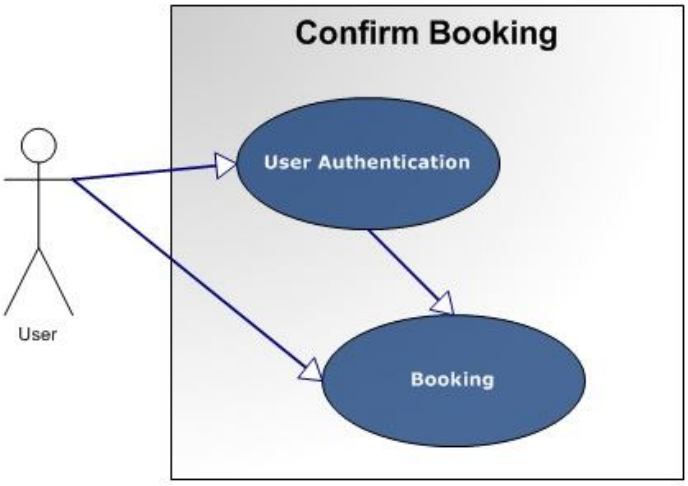


CU05.- Confirm Booking

Actor: User

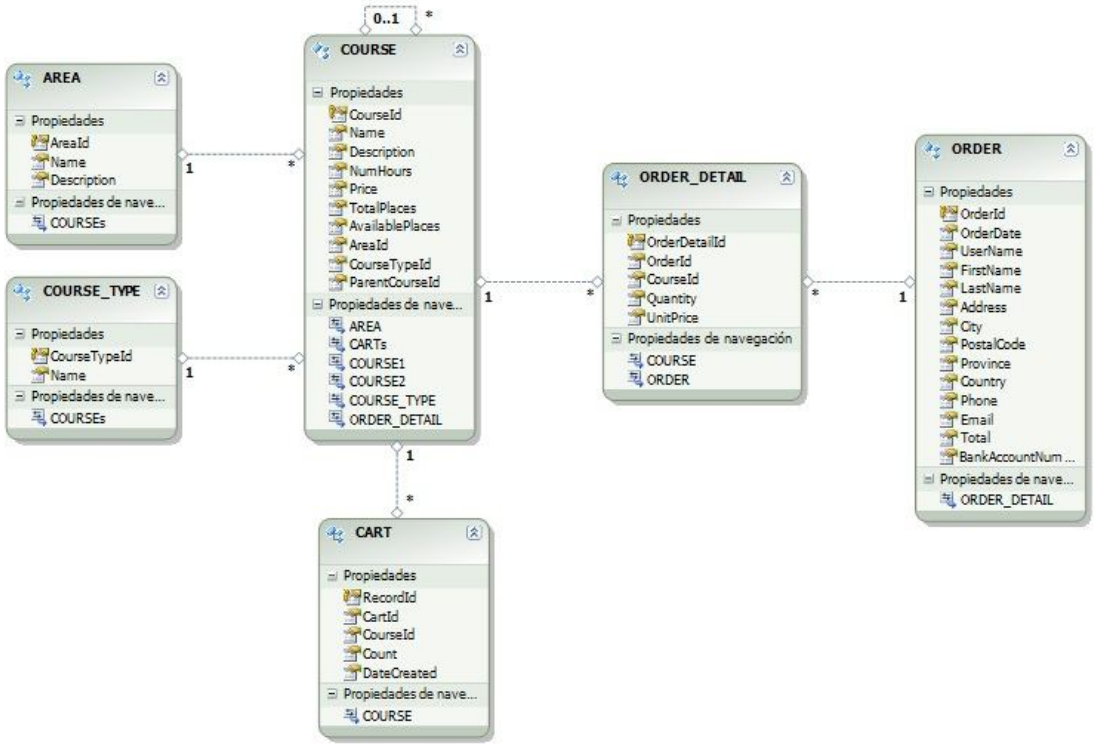
Descripción: Una vez que el usuario ha realizado la búsqueda y selección de los cursos que considera adecuados, puede pasar a confirmar la reserva. En función de si el usuario esta registrado previamente en el sistema o no, tendremos dos flujos de acciones alternativos.

1- En el primer caso, si el usuario está registrado previamente en el sistema, la confirmación de la reserva pedirá simplemente unos datos adicionales (personales y bancarios) y ejecutará el proceso.
2- En caso de que el usuario no esté registrado, se le enviará a la pantalla de registro para poder validar su acceso, previamente a la confirmación de la reserva.



MODELO CONCEPTUAL

Presentamos el modelo conceptual de la aplicación, en el reflejamos las principales entidades del dominio, utilizadas en el desarrollo, obteniendo así una visión general de la forma en que se interrelacionan. Posteriormente desarrollamos una descripción de cada una de ellas.



COURSE

Se trata de la entidad que se corresponde con los cursos que gestionará el sistema. Destacamos los siguientes atributos para el nombre, descripción, precio, número de horas, plazas totales, plazas disponibles, así como los atributos necesarios para definir las relaciones con otras entidades.

La entidad curso se relaciona con area, course_type, cart y order_detail, siguiendo las características presentadas en el diagrama.

AREA

Nos referimos a la entidad correspondiente a las áreas funcionales en las que podemos clasificar los diferentes cursos. Definimos su nombre y su descripción como atributos principales y destacamos su relación con cursos.

COURSE_TYPE

Esta entidad hace referencia a los tipos de cursos. Si bien, inicialmente, se han definido 3 tipos fijos (cursos, seminarios y masters), hemos optado por dar persistencia al concepto de tipo de curso, valorando la opción de que en un futuro la aplicación pudiera evolucionar y necesitar una gestión o ampliación de estos tipos.

CART

El concepto definido con esta entidad se corresponde al carro de la compra en un diseño similar al de una tienda virtual. Aquí es donde guardaremos y gestionaremos la selección de cursos que hará el usuario mientras navega por la aplicación y elige los que más le interesan. Una vez confirmada la reserva se volcará la información en la entidad order.

Destacamos los siguientes atributos: identificador del curso, cantidad seleccionada, fecha de alta y dos identificadores uno para la entidad y otro para el registro. La necesidad de este último identificador (RecordId) se deriva del requerimiento funcional de que un usuario pueda hacer selecciones sobre cursos al entrar en la web sin haberse logado previamente.

ORDER

Implementa el concepto de reserva. Se corresponde con la entidad que utilizaremos para dar persistencia a las reservas confirmadas por el usuario. Informa sobre los datos básicos del usuario, datos personales y bancarios. Se relaciona con order_detail, donde se encuentra un listado de los diferentes cursos reservados en esta petición concreta.

ORDER_DETAIL

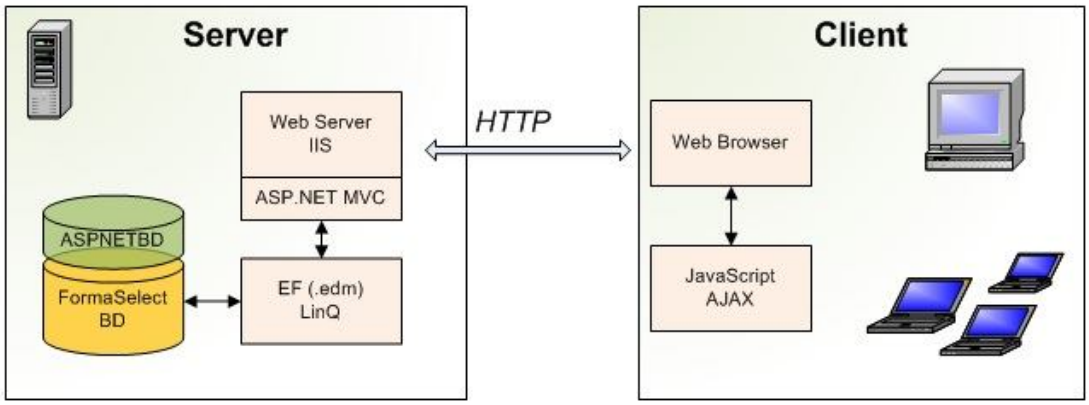
Define una relación entre una reserva y los cursos que la componen, atendiendo al requisito funcional de que un usuario pueda seleccionar y reservar más de un curso, o más de una plaza para un mismo curso, en una misma sesión de trabajo con la aplicación.

DIAGRAMAS DE ARQUITECTURA

Nos proponemos, a continuación, ofrecer una visión simplificada del sistema, de forma que podamos obtener un enfoque global de la estructura y arquitectura del mismo. Exponemos tres tipos de diagrama, en función del punto de vista utilizado para la representación de nuestro aplicativo.

Arquitectura física. Hardware

Se trata de exponer como se distribuyen los distintos elementos del software en cada una de las maquinas que forman el despliegue físico, de acuerdo a su funcionalidad especifica e interrelacionándose con los demás módulos para alcanzar la funcionalidad general ofrecida por el sistema.



Podemos observar que se trata de una arquitectura distribuida siguiendo el modelo cliente-servidor. Al tratarse de una aplicación web, la aplicación reside y se ejecuta en una maquina que actúa como servidor, pero la interfaz de usuario se expone, a través del navegador, en otra máquina que actúa como cliente, y desde la que el usuario se relaciona con el sistema.

La comunicación entre ambas maquinas, se realiza, como es habitual en este tipo de desarrollos utilizando el protocolo HTTP a nivel de aplicación y TCP/IP para la capa de transporte.

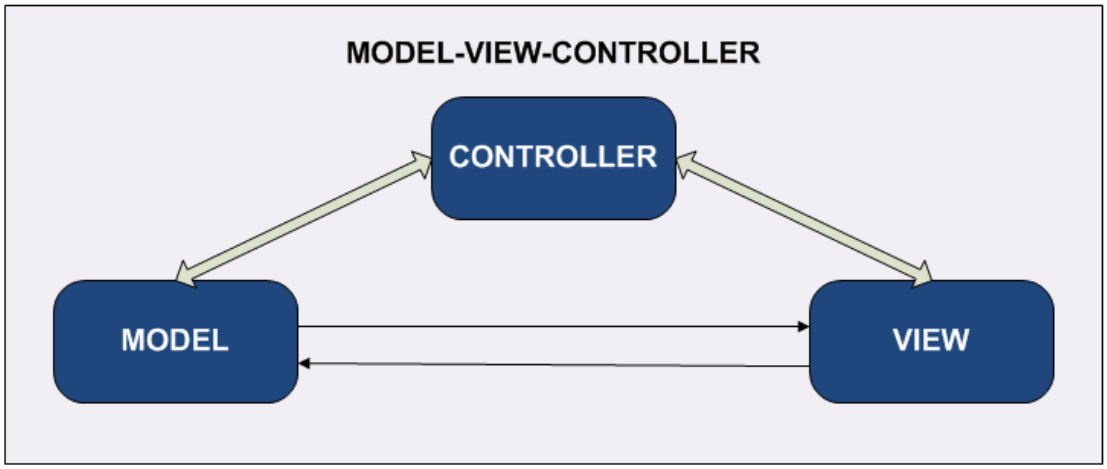
La máquina servidor, además de implementar el servidor web propiamente dicho, con IIS, también será la encargada de alojar y gestionar nuestro sistema gestor de base de datos (Servidor SQL 2008) y la propia base de datos del sistema.

Arquitectura lógica. Software

Mostramos a continuación el esquema de la arquitectura lógica que se ha seguido en el desarrollo del proyecto.

Se trata, como ya se ha comentado, del patrón MVC, una arquitectura orientada a desacoplar la interfaz de usuario (de entrada y presentación) aislándola de la lógica de negocio de la aplicación, y del modelo de datos implementado, permitiendo así que cada una sea desarrollada, probada y mantenida de forma independiente.

Siguiendo este patrón de diseño, la arquitectura del software queda separada en tres capas, según se puede ver en el diagrama.



Modelo

Se trata de la representación específica de la información o modelo de datos con el cual opera el sistema. En este caso el modelo es el encargado de mapear la información almacenada en las distintas clases que representan las entidades, pero es independiente del sistema de almacenamiento que se implementa en una base de datos centralizada con SQL Server 2008.

En el modelo también quedan definidas algunas reglas de negocio enfocadas a las validaciones haciendo uso de la técnica Data Annotations que proporciona ASP.NET MVC.

Vista

Se encarga de la presentación al usuario del modelo en un formato adecuado para que este pueda interactuar, lo que habitualmente se conoce como interfaz de usuario.

La vista recibe los datos del modelo y los presenta al usuario, recogiendo la información introducida por éste y reaccionando, según las funciones y métodos implementados en el controlador, a los distintos eventos lanzados por el usuario.

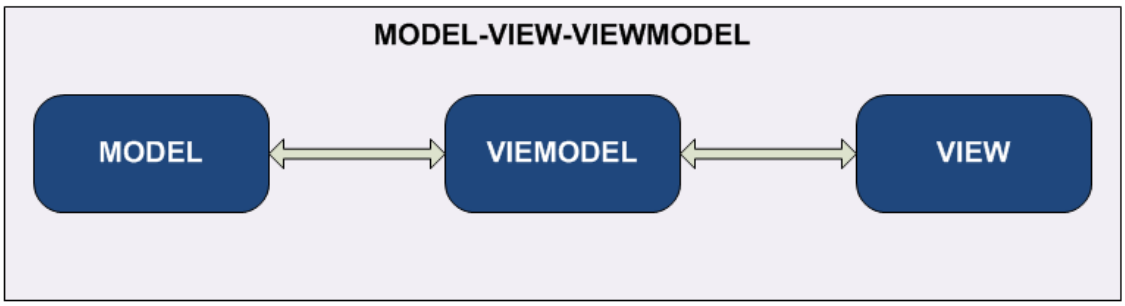
Controlador

Es el encargado de gestionar tanto la vista como el modelo, respondiendo a los eventos que recibe del usuario, invocando las diferentes peticiones al modelo y proporcionando la vista adecuada como respuesta.

ViewModel

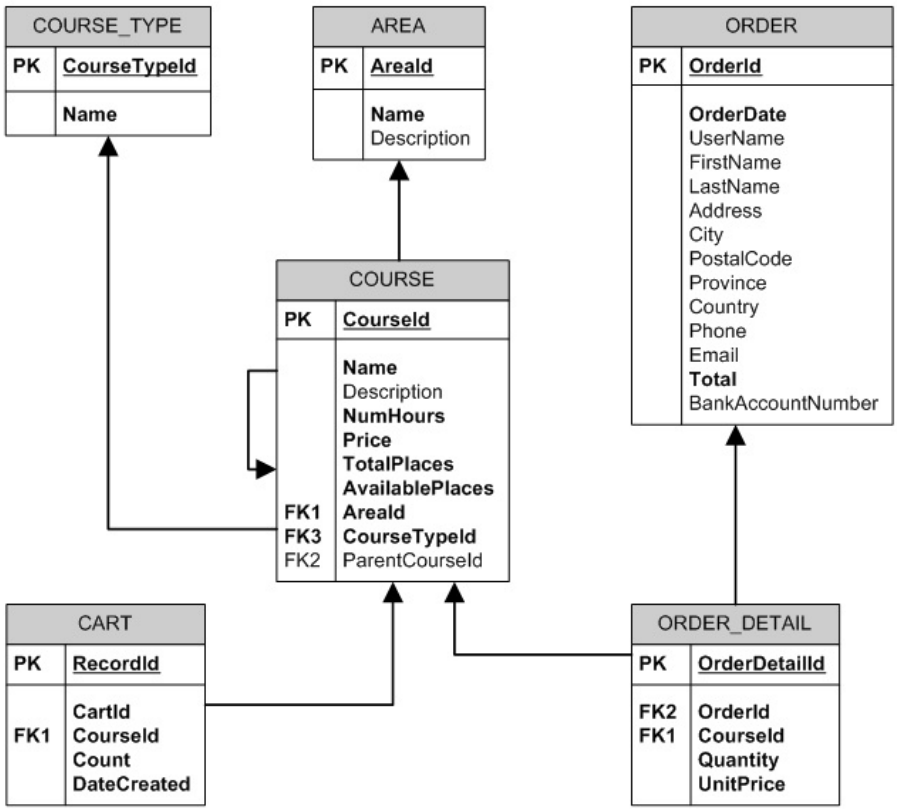
Además de esta arquitectura en tres capas, durante el desarrollo del proyecto se ha utilizado otro patrón de diseño denominado Model-View-ViewModel, para implementar algunas pantallas.

Este patrón es muy similar al MVC, de hecho es una variación de este, pero añade un elemento estructural más el ViewModel, que se sitúa como intermediario entre la vista y el modelo. El ViewModel es, en realidad, un modelo, pero diseñado a medida para la presentación de una vista concreta, cuando ésta no se adapta exactamente a alguno de los modelos definidos.



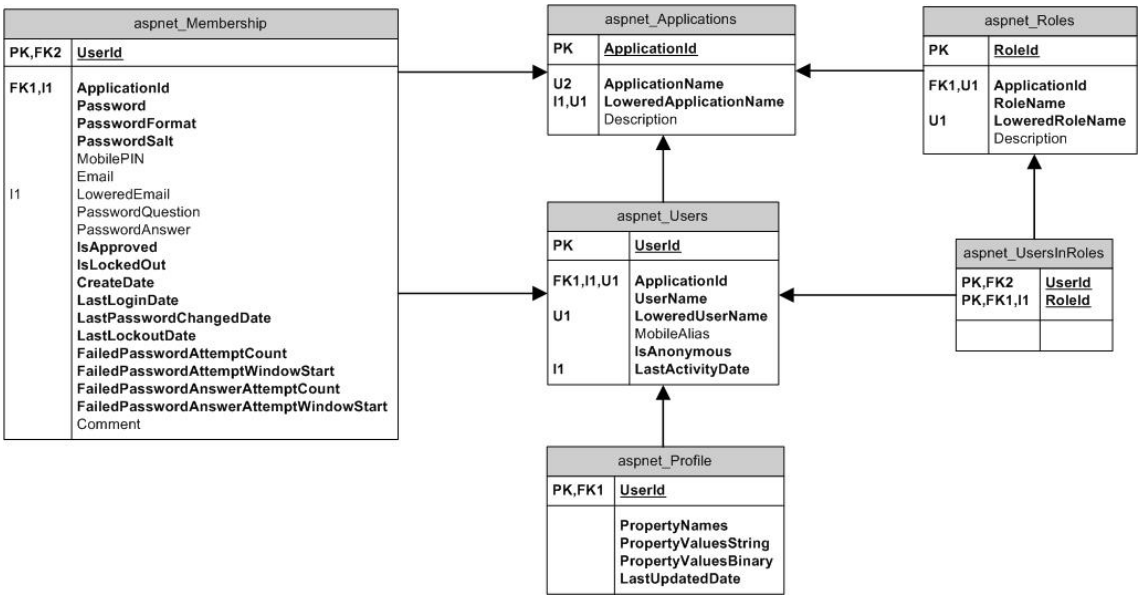
DISEÑO DE LA BASE DE DATOS

Presentamos, a continuación, el diseño de la persistencia en base de datos con el correspondiente diagrama entidad relación (diagrama E/R), para nuestro modelo de entidades definido previamente.



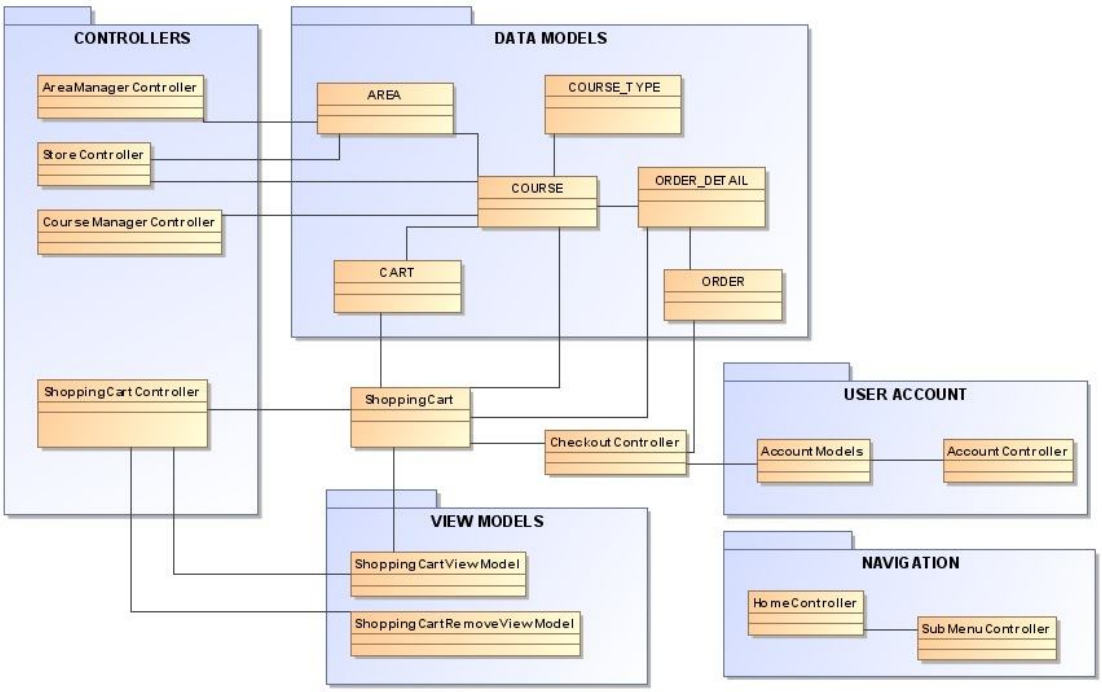
Cabe señalar que, además de esta base de datos, se ha utilizado otra independiente para la gestión de los usuarios, que hemos implementado según el modelo diseñado en .NET y que hemos preferido mantenerla en su ubicación original, definida por defecto como la base de datos aspnetbd.

Presentamos, a continuación, un diagrama de esta base de datos, pero mostrando solamente aquellas tablas que hemos utilizado para implementar el sistema de autenticación de usuarios.



MODELO DE CLASES

Se presenta, a continuación el modelo o diagrama de clases del proyecto. El diagrama de clases es un tipo de diagrama estático que define la estructura del sistema, mostrando las clases que lo componen y la forma en que se relacionan entre ellas.



Las clases presentadas, se han distribuido en subsistemas o paquetes, en función de su naturaleza o del módulo funcional al que pertenecen para tener, de esta forma, una visión más clara de la estructura de la aplicación.

Así podemos distinguir por ejemplo, las clases correspondientes a los controladores, al modelo de datos o a los ViewModels (ver arquitectura lógica).

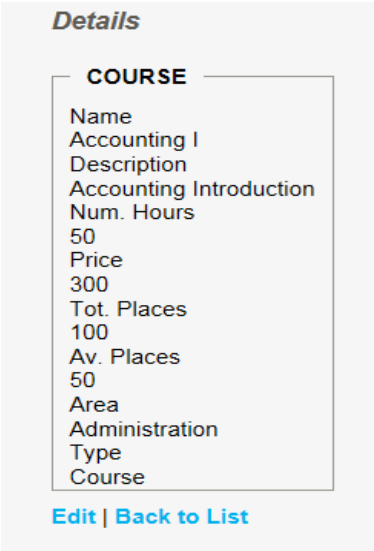
DISEÑO DE LA INTERFAZ

Seguidamente pasamos a detallar el diseño de las distintas pantallas que componen la interfaz de usuario.

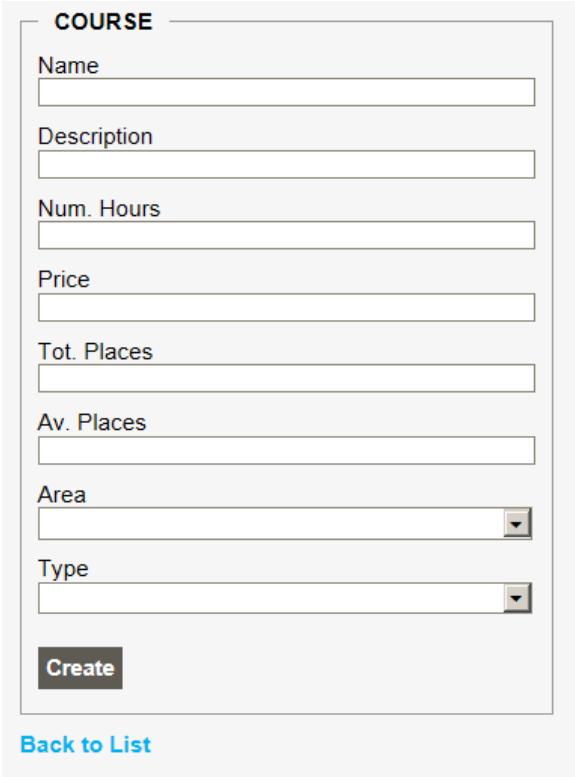
Interfaz que muestra el listado de cursos dados de alta en el sistema:

Area	Type	Name	Description	Hours	Price	Tot. Places	Av. Places	
Administration	Course	Accounting I	Accounting Introduction	50	300	100	50	Edit | Details | Delete
Administration	Course	Accounting II	Accounting Advanced	100	800	100	50	Edit | Details | Delete
Administration	Course	HR	HR Management	50	350	100	50	Edit | Details | Delete
Commercial	Course	Sales	Sales Management	50	400	100	50	Edit | Details | Delete
Commercial	Course	Marketing	Marketing Introduction	50	400	100	50	Edit | Details | Delete
Languages	Course	English I	English beginner level	50	250	100	50	Edit | Details | Delete
Languages	Course	English II	English intermediate level	100	600	100	50	Edit | Details | Delete
Languages	Course	English III	English advanced level	50	350	100	50	Edit | Details | Delete
Computers and IT	Course	Computers I	Computers Introduction	50	250	100	50	Edit | Details | Delete
Computers and IT	Course	Computers II	Computers advanced user skills	50	300	100	50	Edit | Details | Delete
Computers and IT	Course	Development I	Software development	50	350	100	50	Edit | Details | Delete

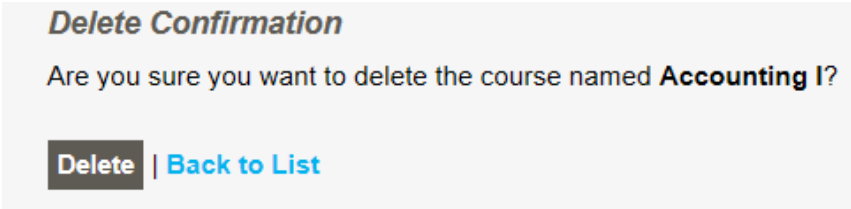
Detalle de un curso seleccionado:



Pantalla para realizar el alta de un nuevo curso, que se utilizará también para la edición, mostrando los datos del curso seleccionado.



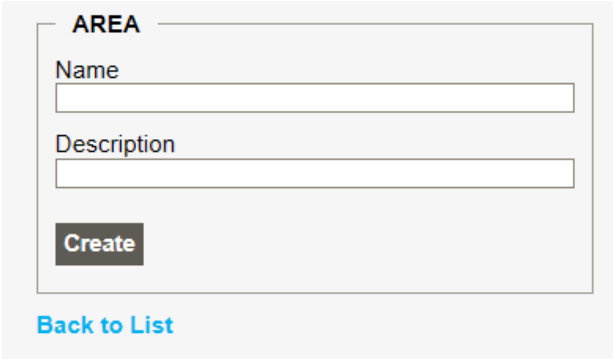
La interfaz pedirá confirmación del borrado de un curso:



Listado de Áreas:

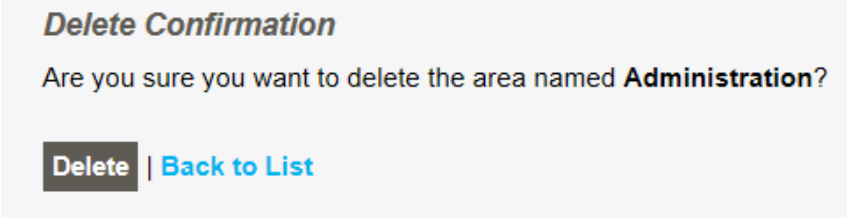
Name	Description	
Administration	Administration	Edit | Details | Delete
Commercial	Commercial	Edit | Details | Delete
Languages	Languages	Edit | Details | Delete
Computers and IT	Computers and IT	Edit | Details | Delete

Interfaz para dar de alta un área formativa nueva. Se utilizará esta misma pantalla para las ediciones, mostrando los datos del área seleccionada:



Detalle de un area seleccionada:

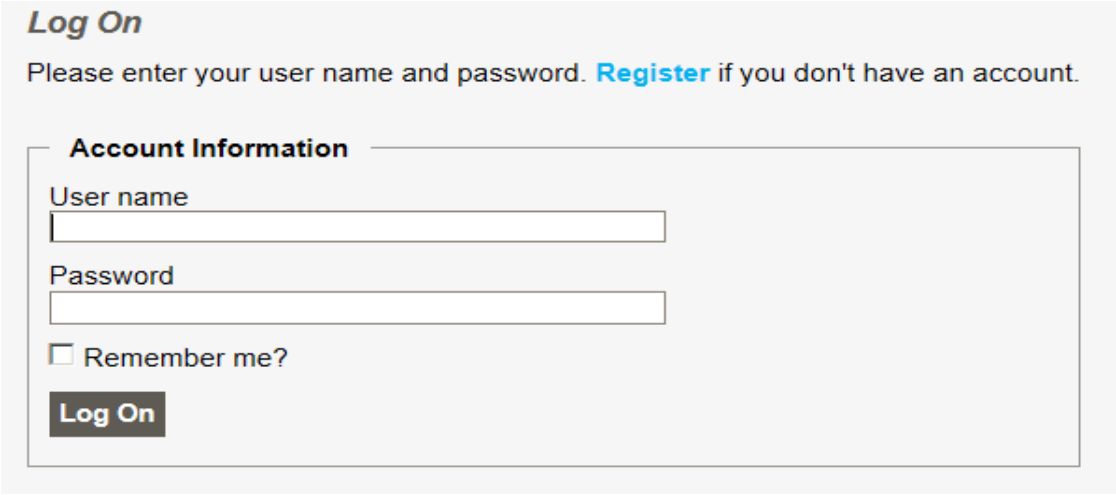
Interfaz para confirmar el borrado de un área formativa:



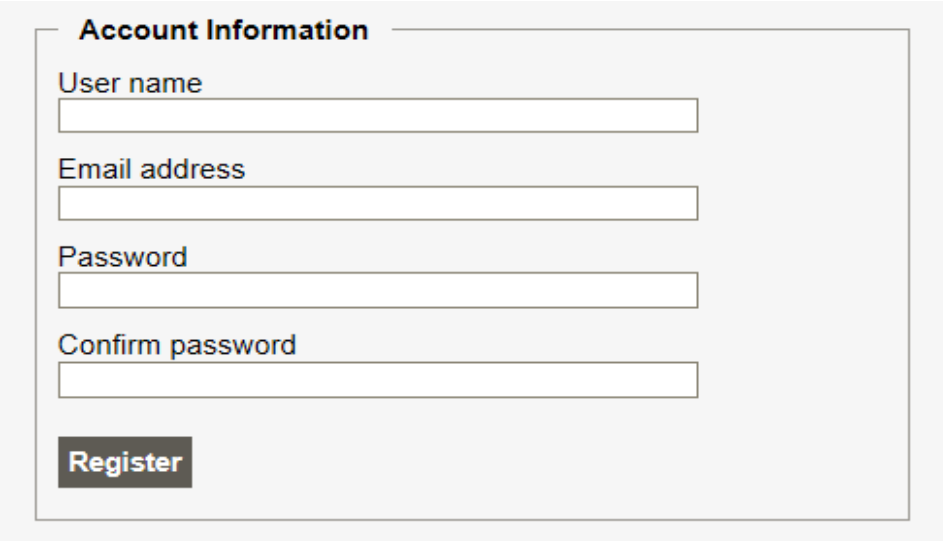
Interfaz que facilita la navegación y búsqueda de cursos clasificados en áreas formativas:

Selección de cursos para reservar:

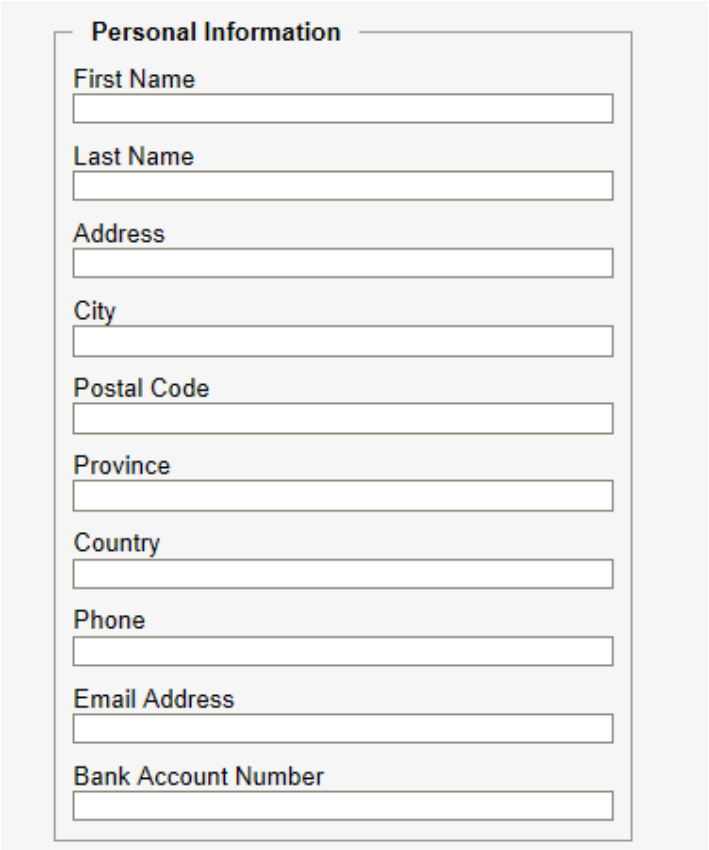
Petición de autenticación, previa a la confirmación de la reserva:



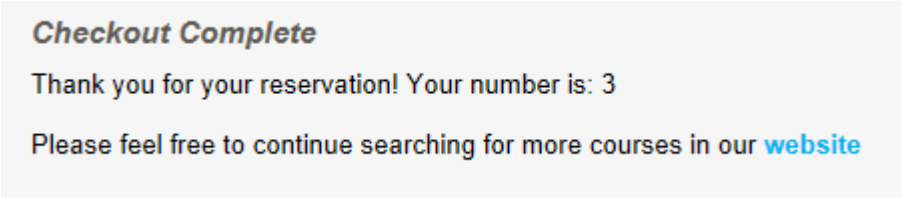
Pantalla de registro para un usuario nuevo:



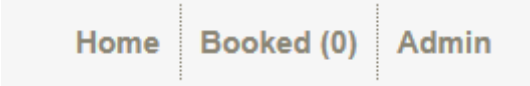
Interfaz para la petición de información personal en el momento de realizar la reserva de los cursos:

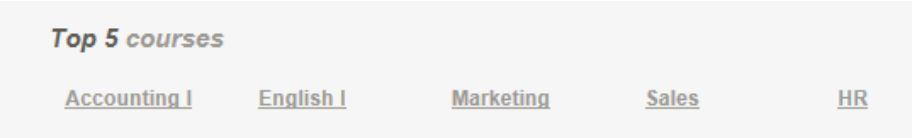


Pantalla de confirmación del registro de la reserva:



Menú que permite el acceso a los módulos de la aplicación, con el detalle de la propuesta de los 5 cursos más reservados:





COMENTARIOS Y CONCLUSIONES SOBRE EL DISEÑO

En este apartado queremos destacar algunos puntos clave y decisiones de diseño relevantes que consideramos de una importancia capital en el desarrollo de la aplicación.

Exponemos además, y de forma documentada, una primera aproximación a las conclusiones obtenidas después del estudio de las diversas tecnologías utilizadas en la implementación del software.

MVC - Convention over Configuration

Las aplicaciones desarrolladas siguiendo la arquitectura ASP.NET MVC, por defecto, dependen significativamente del uso de las convenciones, lo cual define en gran medida el diseño y la estructura de las mismas.

Esto permite a los desarrolladores liberarse de la parte del trabajo más repetitiva y automatizable, evitando tener que configurar y especificar ciertos aspectos que se puede inferir sobre la base de las convenciones.

Por ejemplo, MVC utiliza una convención para la estructura de directorios basada en la estructura de nombres para la resolución de las plantillas de las diferentes vistas. Esta convención nos permite omitir la ruta completa con la ubicación de la vista cuando se hace referencia a ella desde dentro de su clase controlador. De forma predeterminada, ASP.NET MVC buscará el fichero de la vista dentro del directorio \Views\[ControllerName]\...

ASP.NET MVC está diseñado completamente en torno a unos valores predeterminados, razonables y basados en convenios, pero que se pueden reemplazar cuando sea necesario.

Este concepto, que goza de bastante popularidad en otros entornos de desarrollo como Ruby on Raills, se conoce comúnmente como Convention over Configuration.

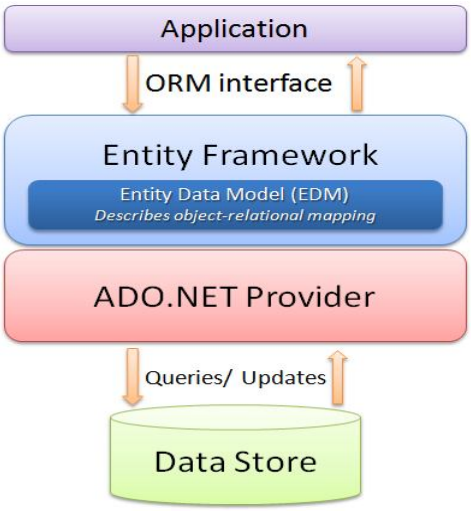
Durante el desarrollo del proyecto hemos aplicado extensivamente esta técnica, porque entendemos que aporta grandes ventajas de eficiencia.

No se trata simplemente de facilitar el diseño estructural y ahorrar trabajo en configuración, si no que la estandarización del código en base a estas convenciones, se convertirá en un factor clave y decisivo a tener en cuenta para el cálculo de los costes de mantenimiento, tanto correctivo como evolutivo, de nuestro software.

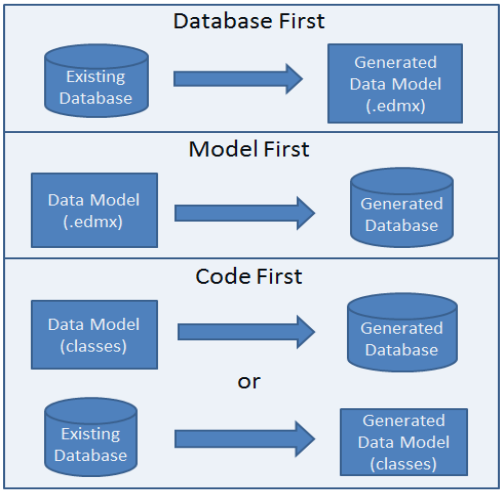
Acceso a datos con Entity Framework

Actualmente la plataforma .NET nos permite elegir entre varias opciones y tecnologías cuando nos enfrentamos a la tarea de diseñar la interacción entre nuestro software y el almacén de datos utilizado para implementar la persistencia de las entidades.

La tecnología Entity Framework (EF) simplifica el acceso a los datos, y nos permite no tener que trabajar directamente con la base de datos desde el propio código de la aplicación. En lugar de esto, podemos recuperar los datos ejecutando consultas directamente contra clases que mapean nuestro modelo de datos, lo que se conoce como Entity Data Model o EDM.



Utilizando la tecnología Entity Framework en su última versión tenemos, a su vez, tres enfoques diferentes para diseñar la persistencia en nuestra aplicación, Database First, Model First y Code First, en función del orden en que diseñemos las distintas capas del sistema, primero la base de datos, el modelo o el código, respectivamente, como podemos ver en la figura siguiente:



Para el desarrollo del presente proyecto, hemos optado por el enfoque Database First. Si bien las otras dos opciones hubieran sido igual de validas e incluso más rápidas en cuanto a la implementación se refiere, hemos considerado que diseñar la persistencia en la base de datos previamente, es el modelo más común y consolidado en entornos profesionales donde, incluso, es muy habitual desarrollar software contra bases de datos implementadas muy anteriormente y/o compartidas por varios aplicativos.

Por esto, decidimos diseñar la base de datos utilizando las herramientas que proporciona SQL Server 2008, previa e independientemente al desarrollo del código de la aplicación, para posteriormente implementar el acceso con Entity Framework y su diseño del EDM a partir de la BD.

Scaffolding

Siguiendo el concepto de convención sobre configuración anteriormente discutido, hemos hecho uso de la técnica conocida como Scaffolding que consiste en la generación casi automática, mediante diversas plantillas, de las vistas necesarias para implementar las operaciones CRUD habituales, es decir el listado, creación, edición y borrado de una entidad.

Para poder hacer uso de esta característica es necesario tener un modelo de datos perfectamente tipificado, debido a que el generador se acogerá a su definición en función de las convenciones configuradas en .NET.

Data Anotations para Validaciones

Para personalizar las clases del modelo diseñado en función de nuestras propias reglas de negocio y especialmente para las validaciones, hemos utilizado la técnica de Data Annotations, que consiste en la aplicación de un tipo específico de metadatos, tipificando las propias clases del modelo, para garantizar su coherencia e implementar ciertas reglas de validación.

La aplicación de esta técnica, potente y sencilla, complementa la utilización de Entity Framework y su modelo EDM (Entity Data Model), permitiendo una configuración personalizada de las clases para el mapeo de las entidades, en función de nuestros propios parámetros.

ViewModel

Este patrón consiste es una variación o ampliación del MVC que introduce una capa intermedia entre el modelo y la vista, el ViewModel (ver arquitectura software en esta memoria).

El ViewModel tiene sentido cuando los datos que se han de presentar en la interfaz de usuario (la vista), son demasiado específicos o no constituyen un modelo definido concreto, si no una combinación de varios. En estos casos,

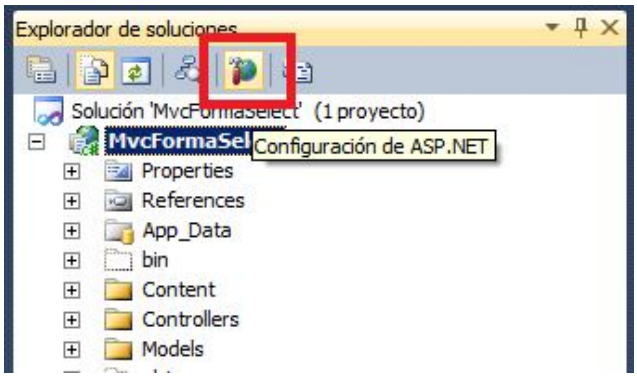
se puede diseñar un modelo específico para esa vista, que habitualmente utilizará datos de varios modelos definidos previamente.

El ViewModel se puede considerar, por tanto, un modelo diseñado a medida para la presentación de una vista concreta, cuando ésta no se adapta exactamente a alguno de los modelos definidos.

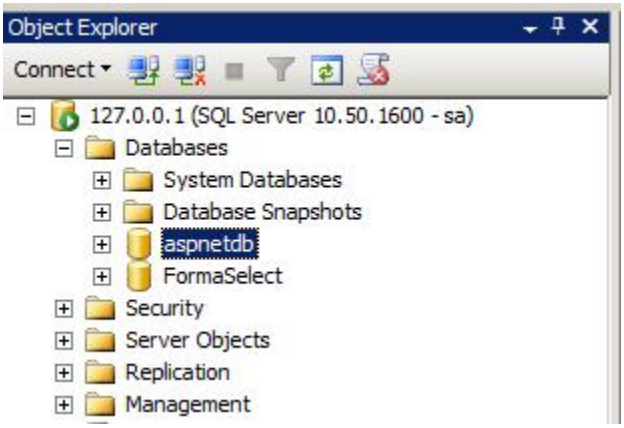
Hemos añadido una carpeta al proyecto con los ModelViews diseñados, donde se puede observar como se ha utilizado esta técnica.

Gestión de usuarios

Después de estudiar el modelo de gestión de usuarios basado en roles que propone la plataforma .NET mediante su website de configuración, hemos observado que se adaptaba perfectamente al modelo que queríamos implementar, por tanto hemos seguido este diseño.



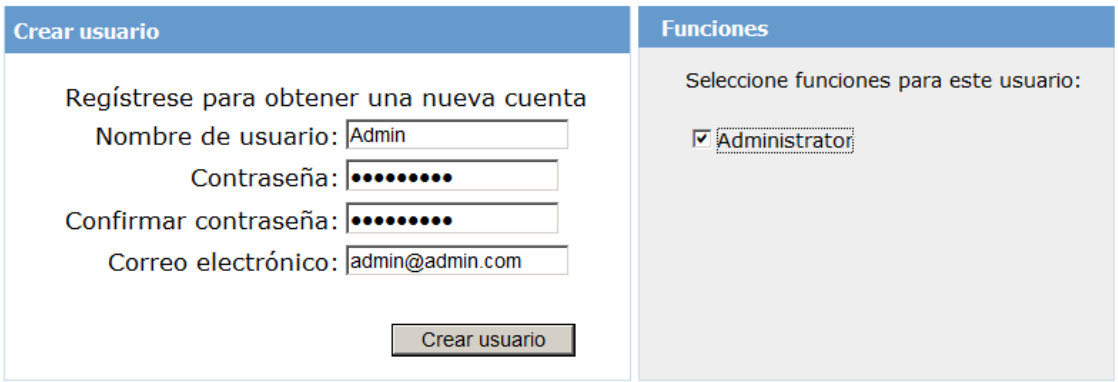
Primeramente debemos asegurarnos de tener instalada, registrada y accesible la base de datos de configuración del sistema aspnetdb. Para ello podemos ejecutar el programa aspnet_regsql.exe que usualmente se encuentra en la carpeta del Framework con el que estamos trabajando. [unidad:]\WINDOWS\Microsoft.NET\Framework\[numeroDeVersion]



Una vez registrada la base de datos, deberemos también asegurarnos de que tenemos configurado correctamente el proveedor AspNetSqlProvider, y ahora si ya podemos lanzar la herramienta de configuración de ASP.NET.

Simplemente tendremos que habilitar la función correspondiente para permitir los roles, crear un rol administrador y un usuario con ese rol.

En nuestro caso hemos añadido el usuario admin con password admin1234, y el rol Administrator como se puede observar en la pantalla.



IV - IMPLEMENTACIÓN

El producto implementado, que se entrega con la presente memoria, se corresponde con una aplicación software que proporciona una solución para las funcionalidades y requisitos definidos inicialmente.

SOFTWARE UTILIZADO

Para el desarrollo del proyecto se han utilizado las siguientes herramientas y tecnologías:

- Entorno de desarrollo: .NET Framework 4.0
- Herramienta de desarrollo: Visual Studio 2010
- Lenguaje de programación: C#
- Base de datos: SQL Server 2008 R2
- Acceso a datos: Tecnología Entity Framework (Entity Data Model) con el paradigma DataBase First.
- Consulta de datos: Lenguaje LinQ (LinQ to entities)
- Diseño de interfaces: HTML con el motor de vistas RAZOR y hojas de estilos CSS.
- Librería scripting: JQuery 1.5.1
- Interacción de las interfaces: AJAX (en algunas acciones).

Software utilizado para la fase de análisis y diseño:

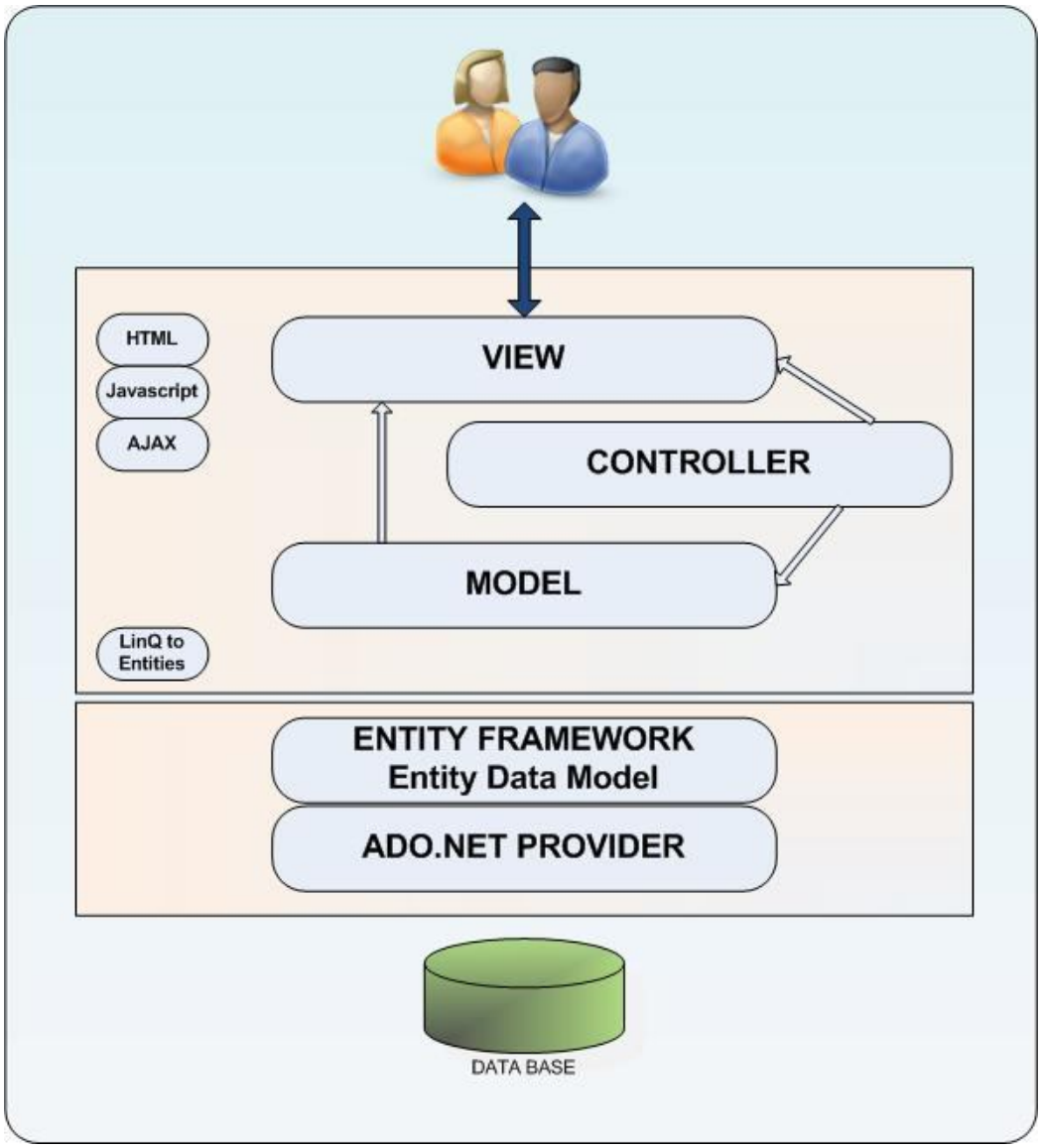
- Microsoft Office Visio 2007
- Magic Draw UML Personal Edition 16.0
- Microsoft Office Word 2007

CAPAS DE LA APLICACION

En el siguiente diagrama se muestra la estructura de la aplicación desde el punto de vista de las distintas capas de software que la componen.

Podemos Destacamos tres bloques bien diferenciadas:

1. Base de datos. Almacén o repositorio de datos (SqlServer 2008).
2. Capa de acceso a datos, implementada con Entity Framework.
3. Aplicación, implementada siguiendo el patrón MVC, modelo, vista, controlador.



V - EVALUACION DE COSTES

Podemos distinguir dos aspectos a tener en cuenta a la hora de realizar la evaluación de costes del proyecto. Por un lado tenemos los costes asociados al desarrollo del proyecto o costes del software, y por otro lado debemos contemplar los costes hardware asociados al despliegue para la puesta en producción del aplicativo.

Desarrollo de Software

La evaluación de los costes en el proceso de desarrollo de software para un producto informático no es una tarea sencilla, porque depende en gran medida de muchos factores externos sobre los cuales tenemos escaso control.

Por ejemplo, se ha demostrado que la fase de mantenimiento suele ser la fase que más recursos humanos y económicos consume. No por su intensidad, que solo pudiera sur puntual en caso de tener que solucionar algún error importante, si no por su extensión en el tiempo. Pero esto es algo que no podemos valorar, solo realizar una estimación aproximada, ya que no sabemos la evolución que el cliente espera para el software (posibles mantenimientos evolutivos).

Dicho esto, nos parece apropiado, dadas las características del proyecto que nos ocupa, y en el ámbito académico en el que nos encontramos, hacer una estimación del esfuerzo empleado en la construcción del software.

La medida habitual del costo es, evidentemente, la cantidad de dinero que cuesta producir un software determinado, mientras que la medida habitual del esfuerzo se puede simplificar al cálculo del trabajo que lleva a cabo un profesional en una determinada unidad de tiempo, día, mes o año, por lo que podemos y solemos hablar de persona-día, persona-mes o persona-año respectivamente. Aunque para este caso concreto, en el desarrollo del presente proyecto solamente contamos con una persona, por lo que la evaluación del esfuerzo ha de realizarse teniendo en cuenta esta característica peculiar.

Dicho esto, hemos estimado que el esfuerzo total de producción del sistema se ha quedado en torno a las 100 o 120 horas de trabajo, incluyendo una primera fase de documentación y aprendizaje de las tecnologías a utilizar, una segunda fase de análisis y diseño del producto, una tercera fase de implementación del mismo y por ultimo una fase de documentación y redacción de la presente memoria.

A partir de aquí, en un entorno profesional, deberíamos multiplicar esta cantidad de horas como mediada del esfuerzo realizado, por el precio unitario del profesional en cuestión, y con esto obtendríamos un primera aproximación al coste del proyecto que habría que complementar con otros aspectos como se indica a continuación.

Hardware

En cuanto al Hardware necesario para desplegar el sistema, y siguiendo la arquitectura diseñada cliente-servidor, tendríamos que utilizar un servidor central para almacenar la base de datos y que actuará a su vez como servidor web de la aplicación.

Para proyectos de gran tamaño sería recomendable tener la base de datos, o las distintas bases de datos en servidores distintos y dedicados. Pero en el caso que nos ocupa, el servidor podría ocuparse de ambas tareas, almacenar la BD y actuar como servidor web de la aplicación.

Tenemos que considerar por tanto el coste de la compra de este servidor, aunque también podríamos optar por contratar alojamiento para nuestro sitio web, ahorrando de esta forma en costes fijos del proyecto.

Otros costes

Tendríamos que tener en cuenta, además otros costes asociados al proyecto, como los costes de las licencias de software, la instalación propiamente dicha, la formación del personal de la empresa encargado de la administración del aplicativo, y sobre todo las tareas de mantenimiento, que como ya se ha indicado tienen una incidencia muy relevante en el coste total del proyecto.

VI - TRABAJO FUTURO

A nivel del producto de software obtenido, se han de contemplar algunas propuestas de mejora del aplicativo.

En cuanto al diseño, consideramos que se podría incrementar la usabilidad y mejorar la experiencia del usuario incluyendo un rediseño de la interfaz y utilizando de forma más extensiva la tecnología AJAX. En la versión actual del proyecto, la aplicación utiliza esta técnica para la selección y la eliminación de elementos de nuestro carro, pero su uso podría extenderse a otros ámbitos.

A nivel académico, y recordando los objetivos planteados en este mismo documento sobre el trabajo final de carrera, consideramos que se ha conseguido alcanzar una visión y un conocimiento adecuados de las diferentes tecnologías utilizadas, especialmente MVC. Pero para conocer todo el potencial que puede ofrecer la plataforma .NET, esto no debe acabar aquí. En este sentido, en un futuro, nos gustaría acercarnos al paradigma de desarrollo TDD.

El desarrollo guiado por pruebas, o Test Driven Development (TDD) es una práctica de programación que involucra otras dos prácticas conocidas como Test First Development y Refactoring, es decir, escribir las pruebas primero, y posteriormente refactorizar el código.

Para escribir las pruebas, generalmente, se utiliza como base las pruebas unitarias (Unit Test). En primer lugar, se escribe una prueba y se verifica si la prueba falla. A continuación, se implementa el código que hace que la prueba pase satisfactoriamente y, seguidamente, se refactoriza el código implementado.

El propósito del desarrollo guiado por pruebas es lograr un código limpio y que funcione bajo la idea subyacente de que los requisitos han de ser traducidos a pruebas, ya que de este modo, cuando las pruebas se ejecuten de forma correcta quedará garantizado que el software cumple con los requisitos que se habían establecido previamente.

Nos parece muy interesante esta aproximación al desarrollo de software y nos proponemos estudiar su aplicación para el presente proyecto, así como la utilización de las herramientas que proporciona NUnit.

NUnit es un Framework de código abierto para implementar pruebas unitarias para la plataforma Microsoft .NET, con sus dos variantes NUnit Forms y NUnit. ASP para su implementación en entornos de escritorio y web respectivamente.

Según nos hemos informado la aplicación del modelo TDD se complementa muy bien con el patrón de diseño MVC, puesto que este patrón favorece la separación de las capas del modelo de datos, la presentación o vista y el controlador. De ahí el interés de seguir avanzando por esta línea.

VII - CONCLUSIONES

El paradigma de desarrollo MVC es un patrón de diseño bastante conocido, y utilizado en otros entornos de desarrollo como Java o Ruby on Raills. Sin embargo su aplicación en la plataforma .NET es bastante reciente.

Después de haber estudiado este modelo y con la experiencia práctica que nos ha proporcionado el desarrollo de una aplicación especifica, podemos destacar algunos aspectos importantes y llegar a varias conclusiones.

Sin duda, estamos en disposición de afirmar que ASP.NET MVC proporciona una metodología de desarrollo muy potente, ágil, fiable y flexible.

Consideramos que es una plataforma potente, por cuanto proporciona multitud de posibilidades y herramientas que permiten un desarrollo más ágil y optimizado en comparación con asp.net webforms. Aquí entrarían en juego conceptos como Convenction over Configuration, el Scaffolding, o las DataAnnotations, que ya se han desarrollado en el apartado de conclusiones y comentarios sobre el diseño, en este mismo documento.

Estos conceptos son claves en la implementación del patrón MVC que nos proporciona la plataforma .NET, porque como ya comentábamos anteriormente, no se trata solo de ahorrar el trabajo mecánico al desarrollador, sino más bien de proporcionar un marco que permita la estandarización y que favorezca la optimización, y por tanto la fiabilidad del código a implementar, facilitando además su posterior mantenimiento.

Pero ASP.NET MVC va un paso más allá y proporciona un entorno totalmente flexible. Si bien, como se ha comentado, el uso de convenciones es la base de la plataforma, el desarrollador tiene en todo momento, una total libertad para personalizar y configurar cualquier aspecto y adaptarlo a sus propias necesidades.

Por otro lado, la separación de conceptos que proporciona MVC, distinguiendo entre las distintas capas lógicas del software, de una forma tan clara y a la vez tan sencilla e intuitiva, que permite que varios

desarrolladores puedan trabajar sobre el mismo proyecto e incluso sobre un mismo modulo de la aplicación, en función de su especialización en la parte de presentación o en la lógica de negocio o en el modelado de datos.

Aunque nuestras conclusiones ponen de manifiesto las bondades de MVC, tenemos que dejar claro que ASP.NET MVC no viene a reemplazar a ASP.NET WebForms, simplemente son dos opciones de desarrollo distintas.

Desde mi punto de vista, creo que se trata más bien un intento de Microsoft por implementar, de forma bastante acertada en mi opinión, un patrón muy popular en otras plataformas y del que hasta ahora .NET carecía.

Para terminar me voy a permitir traducir una cita de Jon Galloway, *Community Program Manager* de Microsoft y autor del libro "*Professional ASP.NET MVC3*" que tanto he utilizado para el aprendizaje de la plataforma.

"*La mejor analogía que podemos hacer, es que ASP.NET MVC3 es más como una motocicleta, mientras que ASP.NET WebForms se parece más a un monovolumen, lleno de airbags y con un reproductor de DVD por si tienes críos y no quieres que te molesten…*

A algunas personas les gustan las motocicletas y a otras personas les gustan los monovolúmenes. Ambas te llevan a donde necesitas ir, pero técnicamente ninguna es mejor que la otra."

- Jon Galloway, 2011 -

VIII - BIBLIOGRAFIA

Jon Galloway, Phil Haack, Brad Wilson, K. Scott Allen (2011) Professional ASP.NET MVC 3. (Wrox Programmer to Programmer). Indianapolis: John Wiley & Sons, Inc.

Microsoft asp.net (2009, enero). "ASP.NET MVC Overview" [artículo en línea]. [Fecha de consulta: 21 de marzo de 2012].
http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview

Wikipedia (2012, marzo). "ASP.NET MVC Framework" [artículo en línea]. [Fecha de consulta: 23 de marzo de 2012].
http://en.wikipedia.org/wiki/Asp.net_mvc

Wikipedia (2012, junio). "Model–view–controller" [artículo en línea]. [Fecha de consulta: 6 de junio de 2011].
http://en.wikipedia.org/wiki/Model-view-controller

Wikipedia (2012, Junio). "Test-driven development" [artículo en línea]. [Fecha de consulta: 10 de junio de 2012].
http://en.wikipedia.org/wiki/Test-driven_development