Prototipo de sistema de comercio electrónico

Trabajo de Libre Elección de Comercio Electrónico

06/04/2012

EII Universidad de Oviedo
Armando Ramírez Vila

Contenido

Ilustraciones

Introducción

Como el titulo indica, se trata de un prototipo para un sistema de comercio electrónico, encaja con el trabajo propuesto como prototipo de una tienda web, aunque desde una perspectiva más general.

La finalidad del trabajo, es explorar como podría ser la implementación de un sistema de comercio electrónico con soporte para todos los implicados, clientes, proveedores y poseedores del sistema de comercio electrónico.

En este último párrafo se encuentra la razón de que haya cambiado el título del trabajo, ya que la mayor parte de mis esfuerzos se han concentrado en pensar una arquitectura física, lógica y tecnológica que soporte un sistema con múltiples caras y permitía un plug and play de aplicaciones sobre un núcleo también extensible y accesible mediante servicios, una arquitectura SOA.

El sistema se desarrollado íntegramente con tecnologías .NET, cuenta con un núcleo funcional estratificado internamente que ofrece las funcionalidades en forma de servicios a un conjunto de clientes con funciones y ubicaciones distintas, en principio se han desarrollado 3 aplicaciones, dos clientes de escritorio y una aplicación web (La Tienda).

En el apartado que he denominado **descripción** del sistema explicare desde un punto de vista funcional las partes del sistema, en los **aspectos teóricos** abordare muy por encima cada una de las tecnologías usadas en el desarrollo así como otras posibles a usar en ampliaciones, los siguientes 3 apartados **casos de uso**, **arquitectura** y **diseño** examinan el sistema desde el punto de vista de un ingeniero de software,(de forma muy breve ya que no he tenido tiempo de hacer ingeniería), y por último en el apartado de despliegue comentare los detalles de despliegue de este sistema.

Descripción

En términos de un usuario este sistema ofrece una serie de productos genéricos agrupados en categorías para facilitar su gestión y distribución, podemos tener una categoría ordenadores en la cual tendremos varios pc en venta con un determinado número de existencias.

Los proveedores pueden proveer los productos que un administrador haya registrado en el sistema, con lo que aumentarán el stock de productos y generarán un contrato.

Los clientes de la tienda podrán comprar online los productos de dicha tienda que tengan stock con lo que generarán un pedido.

Como he dicho antes el sistema consta de un núcleo funcional que expone una serie de funcionalidades mediante servicios, (aunque me refiera a él como un núcleo, internamente es un sistema en capas), las funcionalidades ofrecidas obviamente tienen relación con la gestión de la tienda, ej: obtener todos los productos.

Sobre estos servicios he construido 3 aplicaciones que he identificado como necesarias en un sistema de comercio electrónico, un cliente de administración, un cliente para proveedores y una aplicación web para la venta de los productos.

Cliente de Administración

El cliente de administración es una aplicación de escritorio que ofrece a un administrador del sistema de comercio la gestión de todo lo que existe en el sistema, esto implica obviamente consultarlo.

El administrador puede mediante este cliente ver todos los productos que hay hasta el momento en la tienda, modificarlos eliminarlos o añadir nuevos, además podrá modificar la agrupación de estos en categorías, por tanto el administrador podrá también crear, eliminar y modificar categorías.

Por otro lado con este cliente se podrá también gestionar los proveedores y clientes del sistema. Los clientes pueden ser añadidos por el administrador o se pueden registrar en la web, en cambio los proveedores solo los podrá añadir el administrador. Las labores de gestión de clientes y administradores implican la creación modificación y eliminación de estos, teniendo en cuenta además sus datos de acceso.

Por último el administrador del sistema desde este cliente se encargará de gestionar los pedidos generados por la acción de los clientes y los contratos generados por las acciones de los proveedores.

Cliente de Proveedores

Los proveedores tendrán un cliente que les permitirá modificar su información así como proveer productos, para proveer productos a los proveedores estos se les agrupan en categorías, si quieren proveer algún producto tendrá que añadir la categoría de dicho producto y luego proveer un determinado stock del producto.

Aplicación Web

Por último mediante una aplicación web en la cual los clientes pueden adquirir los productos, agrupados en categorías. Los clientes siempre podrán ver los productos pero para ver la sección del carrito de la compra tendrán que acceder con credenciales, previamente asignadas por el administrador u obtenidas mediante registro.

Los clientes pueden añadir al carrito los productos que deseen, el carrito les permite ver los productos añadidos, aumentar o disminuir la cantidad de estos así como limpiar el carrito, si se añade un producto repetido se aumentara en uno la cantidad.

Para realizar la compra el cliente deberá ir al carrito y seleccionar comprar, esto generará un pedido para el cliente dado con un listado de productos y una cantidad de cada producto.

Con esta funcionalidad se cierra el círculo entre proveedores, vendedores y clientes, en una forma mínima ya que se trata de un **prototipo**.

Otras Aplicaciones

Dada la arquitectura orientada a servicios del sistema en su conjunto, se hace posible el añadir nuevas aplicaciones sobre dichos servicios así como servicios que combinen los existentes con externos para generar más funcionalidad. Por ello me parece interesante analizar algunas aplicaciones que se podrían construir:

Cliente RIA para compradores:

Se ha estudiado construir una aplicación enriquecida con tecnologías cercanas a la usada en la aplicación de venta, en concreto se valora integrar una aplicación que permita a los clientes ver y modificar toda su información personal, eliminarse del sistema o ver su historial de pedidos.

Servicio de venta a terceros

Se plantea el ofrecer a otros sistemas de comercio electrónico la compra de productos en modo masivo, pasar de vendedor a proveedor, se habla de un servicio y no de una aplicación para que pueda ser automático le hecho de que otros proveedores integren nuestros productos en sus aplicaciones de venta, generando algún tipo de comisión, abriendo así la puerta para que nos ofrezcan otros a nosotros la posibilidad de ofrecer sus productos en nuestras aplicaciones.

Bus de Intercambio

En un extremo se plantea la administración de un bus de servicios, en el que varios sistemas similares sean capaces de intercambiar productos de manera directa.

Aspectos Teóricos

En este apartado comentare los aspectos más técnicos de este trabajo, aunque pueda parecer secundario, junto con la arquitectura ha sido uno de los núcleos de mi esfuerzo, todas y cada una de las tecnologías abordadas han presentado novedades para mí, algunas menos pero otras su totalidad.

Con el fin de que se pueda estimar mi esfuerzo en este trabajo expongo brevemente mis estudios en el marco tecnológico, con la mínima extensión posible.

ADO.NET

ADO.NET constituye la tecnología de acceso a datos más estándar en el mundo .NET, es el equivalente a usar JDBC en java, (o a una implementación de ella JDBC es un estándar no un framework), los mecanismos son bastante análogos a otras tecnologías, abrimos conexiones, preparamos nuestras acciones con la consulta y parámetros adecuados y la ejecutamos.

El cambio más radical es que para obtener las conexiones y ejecutar nuestras acciones necesitamos siempre trabajar con un proveedor de datos adecuado, la tecnología incorpora uno para cada uno de los SGBD más importantes, así como uno para orígenes de datos ODBC.

Desde Microsoft nos dicen explícitamente que usar ODBC es bastante menos eficiente que usar sus implementaciones concretas de acceso a los SGBD.

Como no nos suele gustar estar condicionados, la tecnología incorpora implementado una factoría abstracta de estos proveedores, con lo que podemos trabajar con interfaces sin enterarnos del proveedor usado, este se indica mediante un fichero de configuración y se puede cambiar sin recompilar.

Spring.NET

Para los desarrolladores del mundo java Spring Framework no necesita presentación, surgido como un framework para dar soporte a la inyección de dependencias y la orientación a aspectos y devenido a una suite de frameworks para todo lo que necesites, persiguiendo siempre el buen diseño y la facilidad, Spring llega al mundo .NET con la misma filosofía, se presenta como una alternativa fuerte para la inyección de dependencias y la única de peso a la hora de encapsular los problemas transversales en aspectos. Además al igual que en el mundo JEE, Spring ofrece plantillas de diseño simplificadas para usar las tecnologías de la plataforma, siempre ofreciendo la posibilidad de integrarlas con la inyección de dependencias y la AOP.

Entre otras Spring facilita el uso de ADO.NET ofreciendo una plantilla, que simplifica enormemente el trabajo con ADO.NET, además permite hacer un diseño de la capa de acceso a datos mediante Inyección de dependencias.

Spring da soporte también para el manejo de .NET Remoting, Enterprise Services, WCF y ADO.NET Web Forms, permitiendo paliar la gran desventaja de los Web Forms (Acoplamiento con negocio) así como inyección de dependencias en esta capa.

En este desarrollo se ha usado Spring para ensamblar los componentes de la capa de negocio, eliminando el operador de instanciación desde el comienzo de esta hasta la capa de servicios.

Además he usado Spring para exponer los servicios de WCF inyectándole a estos las dependencias de la capa de negocio.

El uso de WCF excluye la tecnología de .NET Remoting y en principio no se hace necesario interoperar con componentes COM+, por lo que Spring Remoting y Spring Services no se han usado.

No he desechado la ayuda de Spring para ADO.NET pero no se ha implantado ya que al estudiar Spring ya se había desarrollado la arquitectura de acceso a datos.

Por otro lado la tecnología elegida para la web ha sido ASP.NET MVC y no web forms, por lo que no se hace útil a prior usar el soporte de Spring para estos.

WCF

Esta tecnología es la última apuesta de Microsoft por los sistemas distribuidos y arquitecturas SOA, ofrece un modelo de objetos para las comunicaciones, que mediante una arquitectura interna en capas construidas con otras tecnologías como NET Remoting , abstrae al desarrollador de varios de los problemas de desarrollar sistemas distribuidos.

WCF permite ofrecer servicios acorde a los estándares de web services, pero es mucho más que web services, ofrece la posibilidad de exponer los servicios mediante distintos protocolos ofreciendo mayor rendimiento que un servicio web tradicional.

Un servicio WCF se encapsula mediante tres elementos llamados el ABC de WCF, Address, Binding y Contract, la dirección nos abstrae la ubicación del servicio, el binding nos abstrae el protocolo de transporte y los contratos son una abstracción que nos permite definir las operaciones de un servicio mediante interfaces y atributos (anotaciones para los que somos Java), todo esto es definido mediante configuración dejando en el código un modelo de objetos común para cualquier tipo de servicios.

Otro tema clave es el hosting del servicio, un servicio de WCF se tiene que alojar vida o muerte en un App Domain (concepto de .NET, un Proceso Windows tiene al menos un App Domain), este APP Domain puede ser desde una aplicación de escritorio hasta un servicio de Windows, pudiendo como alternativa más eficiente alojarlos en IIS, esto es también en un APP Domain, pero un APP Domain que inicia IIS cuando se hacen peticiones al servicio.

WPF

Otra tecnología relativamente reciente que ha lanzado Microsoft es Windows Presentation Foundation, WPF nos trae una tecnología de diseño de aplicaciones de escritorio mucho más moderna que su prima Windows Forms, nos facilita la separación de entre vista y lógica de aplicación imposible en Windows forms mediante un lenguaje de marcado denominado XAML (Xtensible Aplication Markup Language, este no es exclusivo de esta tecnología) y las propiedades de dependencia.

El patrón por excelencia en el desarrollo de este tipo de aplicaciones es el Modelo Vista Modelo de la Vista MVVM, adaptación del Model View Presenter que es una adaptación del legendario Model View Controller, si añadimos el patrón Command que viene expresamente soportado por la tecnología con una interfaz ICommand accesible desde el XAML, obtenemos unas aplicaciones extremadamente limpias.

La herramienta de Microsoft nos ofrece un editor gráfico que nos genera el lenguaje de marcado según vamos dejando caer los controles, y al contrario de lo que pueda pensar cualquiera es mucho más fácil de leer esto que el código C# o VB generado por el editor de Windows Forms.

Que nadie se asuste, no hay gran penalización en el rendimiento, este lenguaje de marcado se hace corresponder directamente con tipos de la plataforma, no se parsea nada, se compila todo.

ASP.NET y ASP.NET MVC

ASP.NET

ASP.NET es resumiendo mucho y dolo en principio la versión en .NET de las antiguas páginas ASP, como todo en la plataforma, ha sido reinventado para encajar en un sistema como .Net framework y con sus objetivos, sin entrar mucho en detalles las paginas ASP.NET no son interpretadas, se compilan.

ASP.NET WebForms

Sobre esta tecnología se construyeron los ASP.NET Webforms, que son un mecanismo para construir aplicaciones web de forma similar a las de escritorio, con eventos y code behind.

ASP.NET MVC

ASP.NET es un framework de la plataforma construido sobre ASP.NET pero que obliga a trabajar mucho más cerca del protocolo, define unos convenios de nombrado y espacios de nombre propios y lo más importante limpia el desastre del code behind en los WebForms habilitando un desarrollo web más tradicional sobre la plataforma, permitiendo pruebas unitarias sobre los controladores, (Para los que hayan usado a Struts2 se da un aire bastante fuerte) , en mi opinión sobrepasa a los web forms en flexibilidad, con este enfoque es muy fácil cambiar la aplicación y más claro (Aquello que nos pasó a todos de que al mes no se entiende el código), pero los Web Forms tienen una curva más suave y te libran hasta cierto punto de JQuery y compañía, cosa que aquí si quieres hacer algo con apariencia algo más fina es difícil.

La tecnología usada para este prototipo ha sido ASP.NET MVC, pensando en una aplicación MVC como la web central y en futuras mini aplicaciones hechas con web forms y Silverlight integradas en esta.

En el tintero

Quedarían el tintero para un sistema más grande Windows Workflow Foundation y Silverligth,

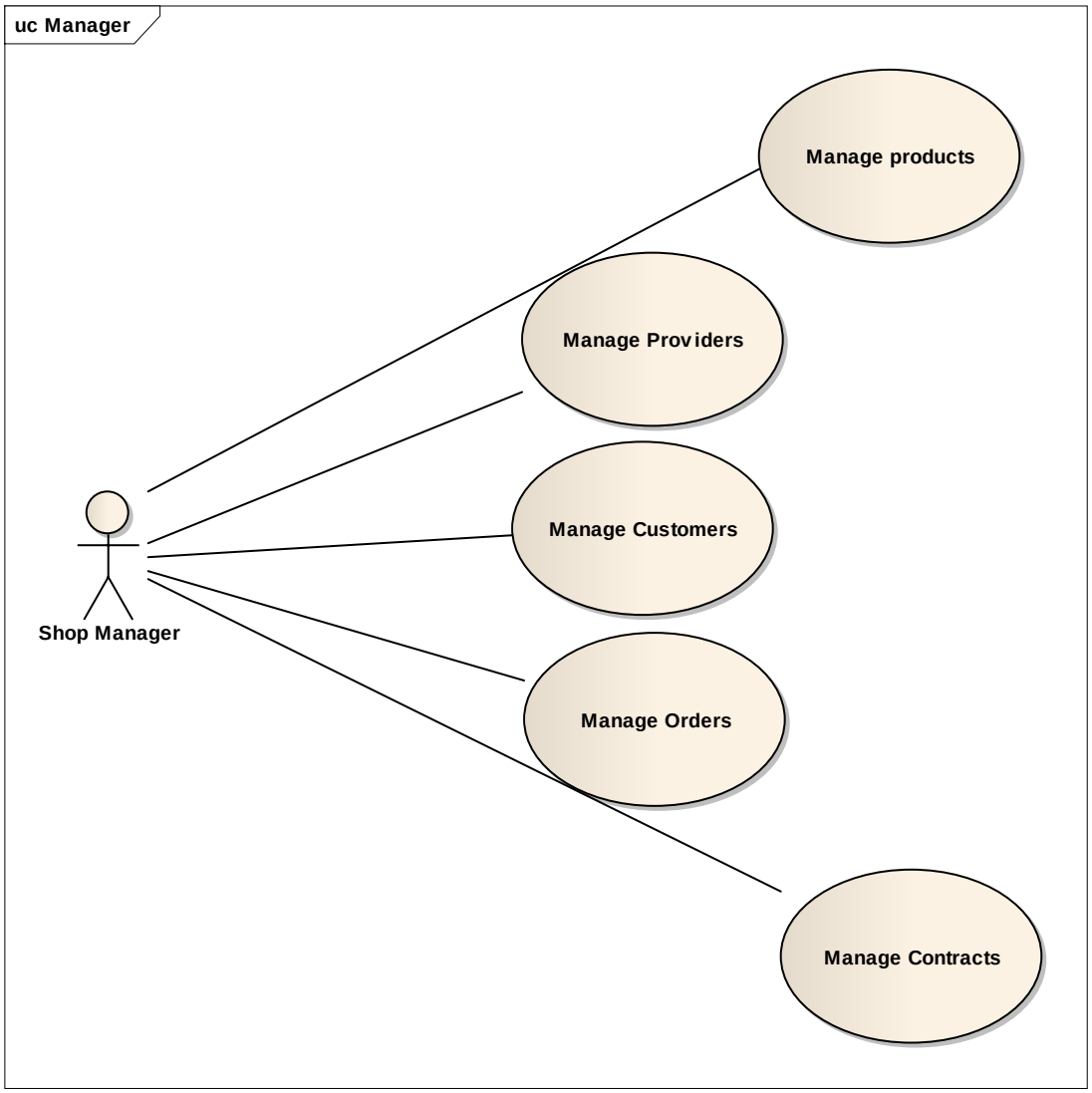
La primera nos permitiría definir flujos entre aplicaciones, lo que de cara a un sistema de intercambio entre muchas tiendas sería interesante.

LA segunda la vería para empotrar en la aplicación MVC central aplicaciones enriquecidas con aspecto similar, de esta forma dejaríamos la limpieza del MVC y obtendríamos los beneficios de las RIAs en puntos muy concretos y atomicos de la Web Global.

EJ: Podríamos poner un enlace en la página de inicio a la aplicación de consulta de clientes, para ver su historial de compras y modificar su estado, aplicación hecha con Web forms o Silverligth que tendría aspecto similar a la web central pero seria más flexible.

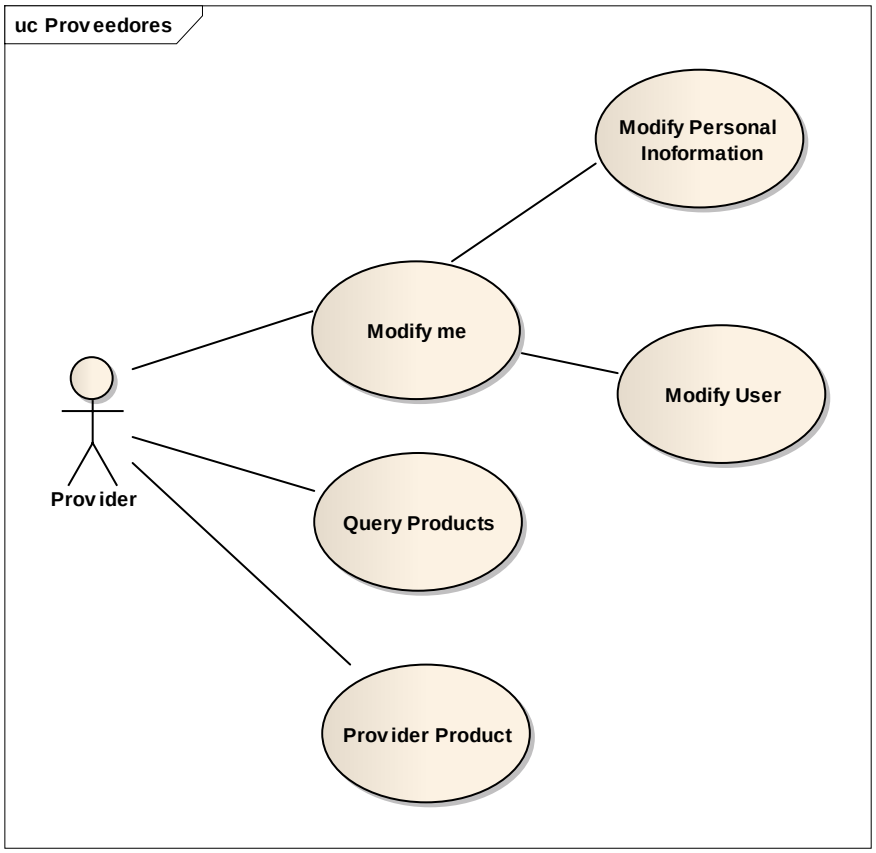
Casos de Uso

Gestión



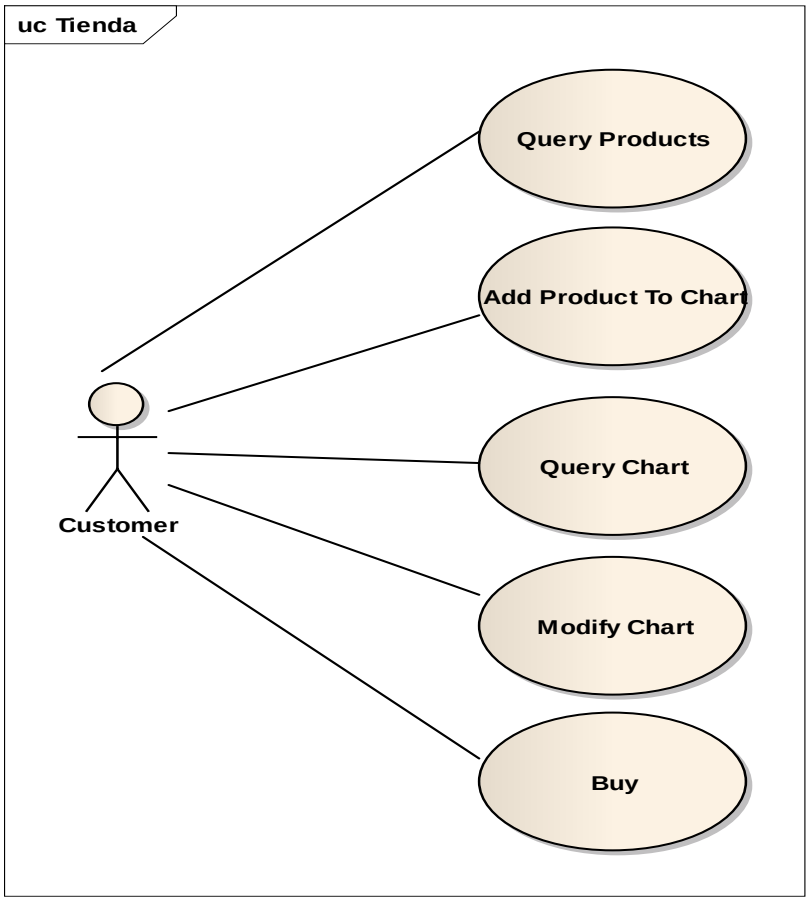
Casos de Uso 1

Proveedores



Casos de Uso 2

Tienda



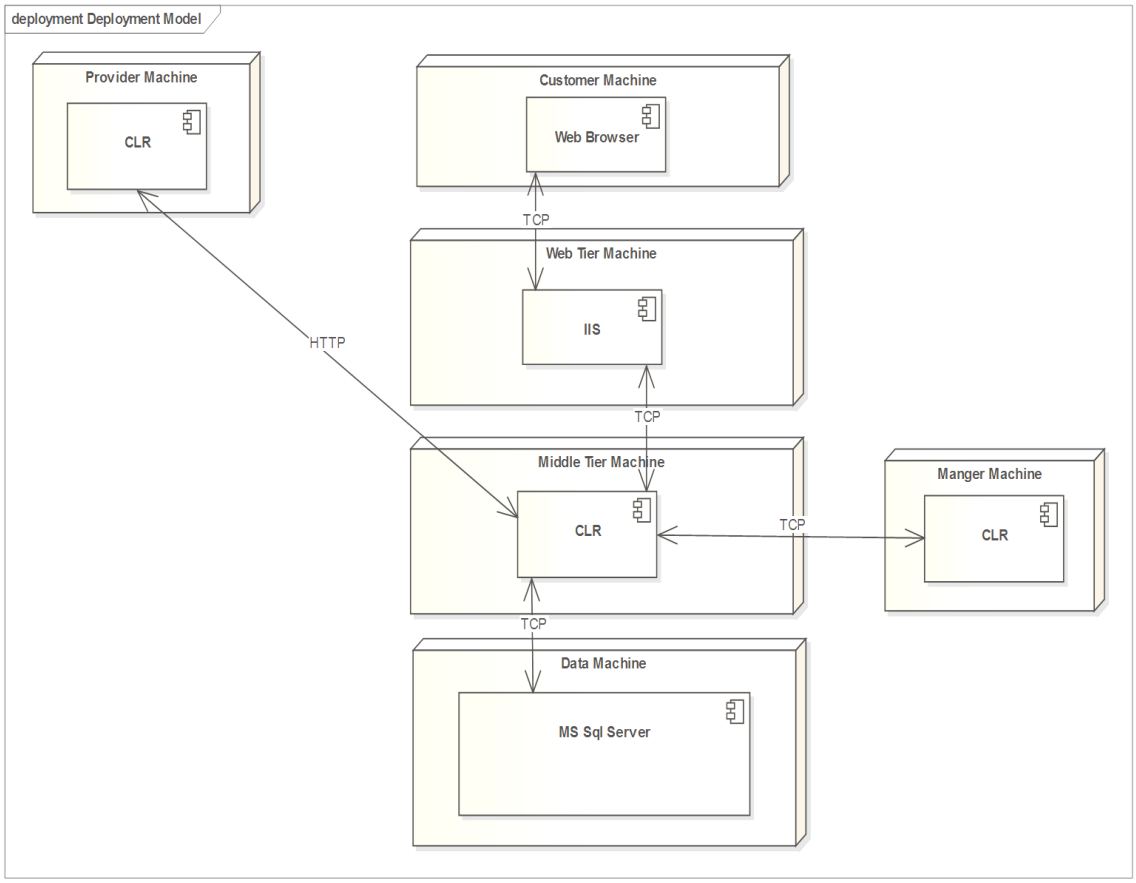
Casos de Uso Customer 3

Arquitectura

El sistema sigue arquitectura en N niveles o N- Tiers, esto es una arquitectura que aplica el patrón arquitectónico Layers a fondo para a partir de una arquitectura lógica estratificada en capas (Layers) separar niveles físicos conocidos como Tiers, es muy típico oír hablar de Web Tier, Middle Tier o Business Tier y de Integration Tier. Como he dicho para lograr esta separación aplicamos un patrón en capas que están condicionadas también por las tecnologías y funciones que debe realizar el sistema, en ese orden solemos tener capa de acceso a datos, capa de negocio, capa de servicios y capa de presentación.

Personalmente me gusta separar la arquitectura en ámbitos, físico o de despliegue, arquitectura de software o lógica y arquitectura tecnológica.

Arquitectura de Despliegue

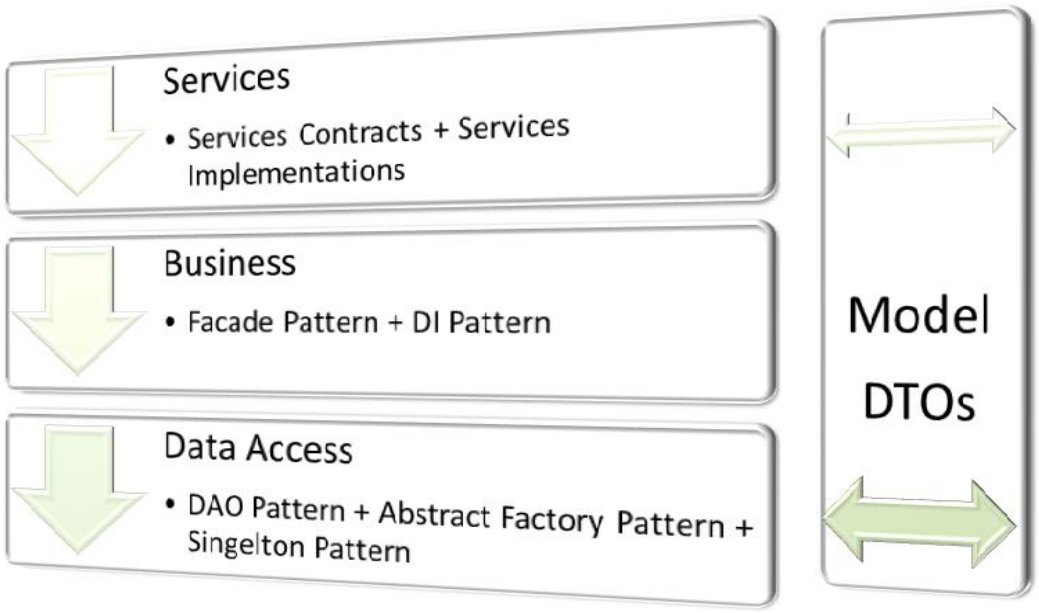


Arquitectura de Despliegue 4

Arquitectura de Software

Para estudiar la arquitectura de software es necesario separar el sistema estudiare la arquitectura del nivel intermedio, luego la arquitectura de los clientes de escritorios que es muy parecida y por último la arquitectura de la Aplicación Web.

Arquitectura del nivel Intermedio



Arquitectura de Software 5

Es importante no confundir este modelo con el del patrón MVC o sus derivados, este modelo se refiere al modelo del sistema Producto, Orden, se implementa siguiendo un patrón conocido como DTO, Data Transfer Objet que no es más que definir una serie de objetos simples y serializables que viajan por todas las capas con los datos del sistema.

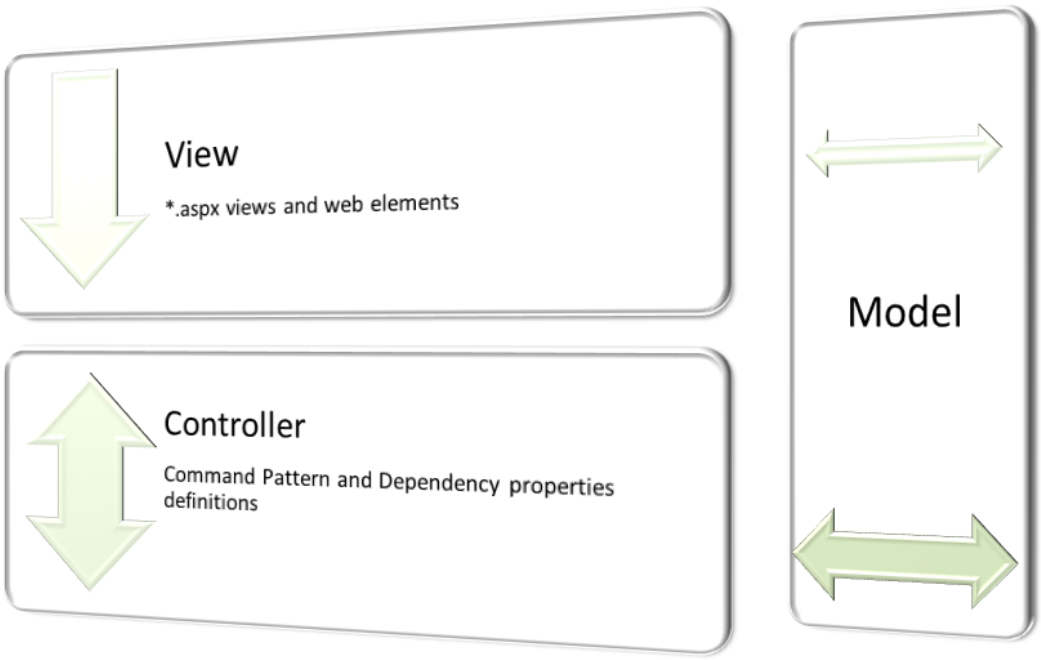
Arquitectura Clientes de Escritorio

Arquitectura de Software 6

Este modelo sí que es el del MVVM, que encapsula básicamente a todo lo que no es ni la vista ni el modelo de la vista, allí dentro esta las clases para acceder a los servicios que siguen un patrón conocido en el mundo JEE como Business Delegate, este encapsula la lógica de acceso a los servicios en una clase que ofrece la funcionalidad de dicho servicio delegandolo a este. Este modelo encapsula además al propio modelo de las capas inferiores y que la tecnología nos lo ofrece tal cual en esta capa (podría no ser así) y a demás lógica de aplicación.

Si aplicásemos el patrón puro la vista no tendría que ser consciente del modelo, pero en mi implementación hay algunos detalles que han llegado a ella.

Arquitetura Tienda Web

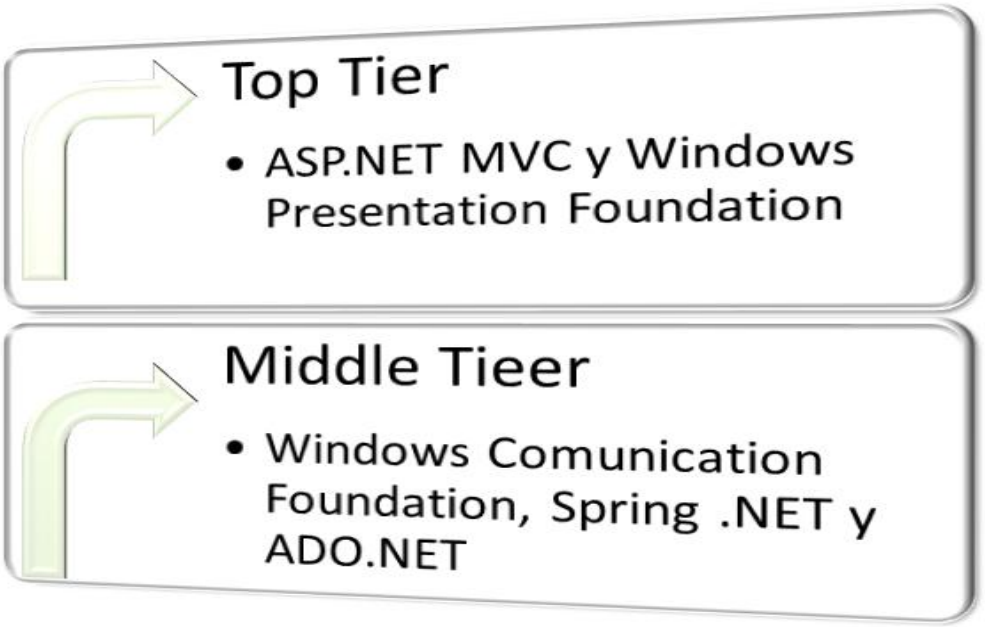


Arquitectura de Software 7

La tienda web sigue un patrón clásico MVC donde las vistas son ficheros .aspx, el controlador se implementa con clases especiales que solo actúan como tal, reciben peticiones, hablan con el modelo y modifican las vistas.

Nuevamente este modelo no son transfer objects, incluye todo lo que no son vistas ni controlador, por lo que además de clases de aplicación y Business Delegates incluye a los DTOs que nos llegan mediante servicios así como una serie de objetos de modelo que podrían ser considerados DTOs de presentación. Enfatizo esto de presentación, en un sistema mediano grande el patrón MVC no deja de ser un mecanismo de estructurar la **presentación** en relación con el resto del sistema.

Arquitectura Tecnológica



Arquitectura Tecnológica 8

Estas son las tecnologías usadas distribuidas sobre la arquitectura más general, empezando por debajo, ADO.NET se usa en la capa de acceso a datos y mediante una factoría de DAOs, se encapsula toda la lógica de acceso a los datos de forma que la capa superior no tiene ni idea de ADO.NET, aislamos la tecnología de una capa a otra.

En la capa de negocio usamos Spring.Net para inyectar las dependencias a todos los objetos, mientras que finalmente en con WCF exponemos una serie de servicios mediante TCP para las aplicaciones detrás del firewall y mediante HTTP para las aplicaciones totalmente externas,

Cliente providers.

Diseño

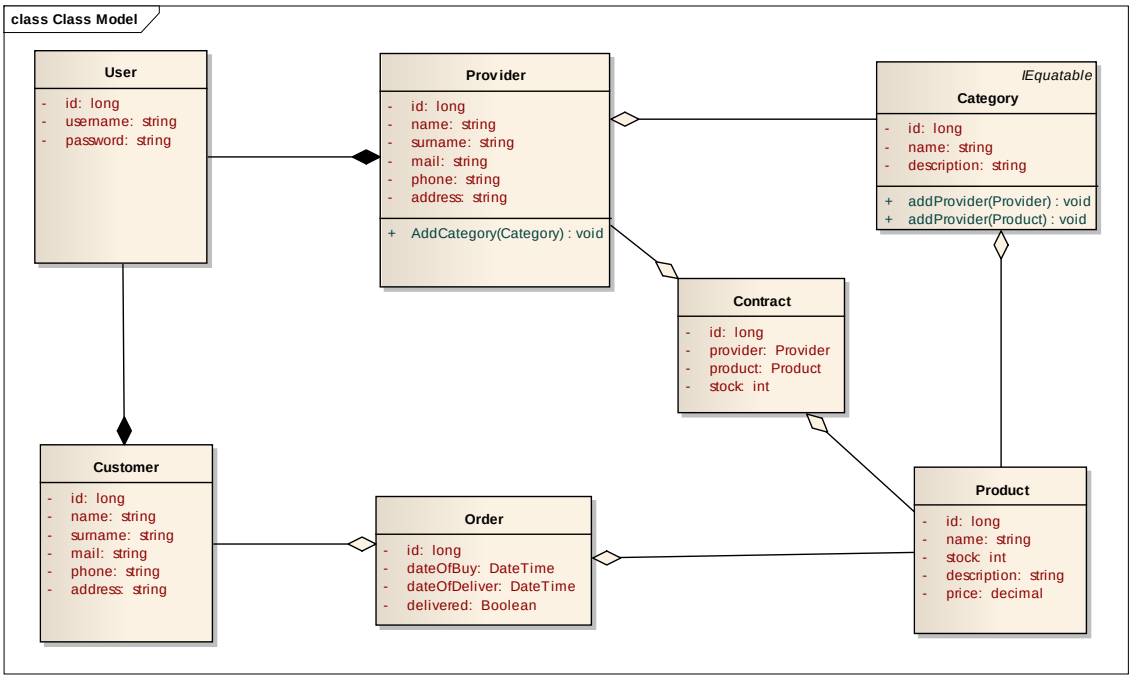
En este apartado comentare el diseño de detalle del sistema, en una fuerte crítica hacia mi tengo que decir que he descuidado mucho el diseño debido a las prisas, pido compresión y recuerdo que es un prototipo, en cualquier caso creo que no me he descuidado lo suficiente como para que no se pueda retocar.

Seguiré la división hecha en la arquitectura de software.

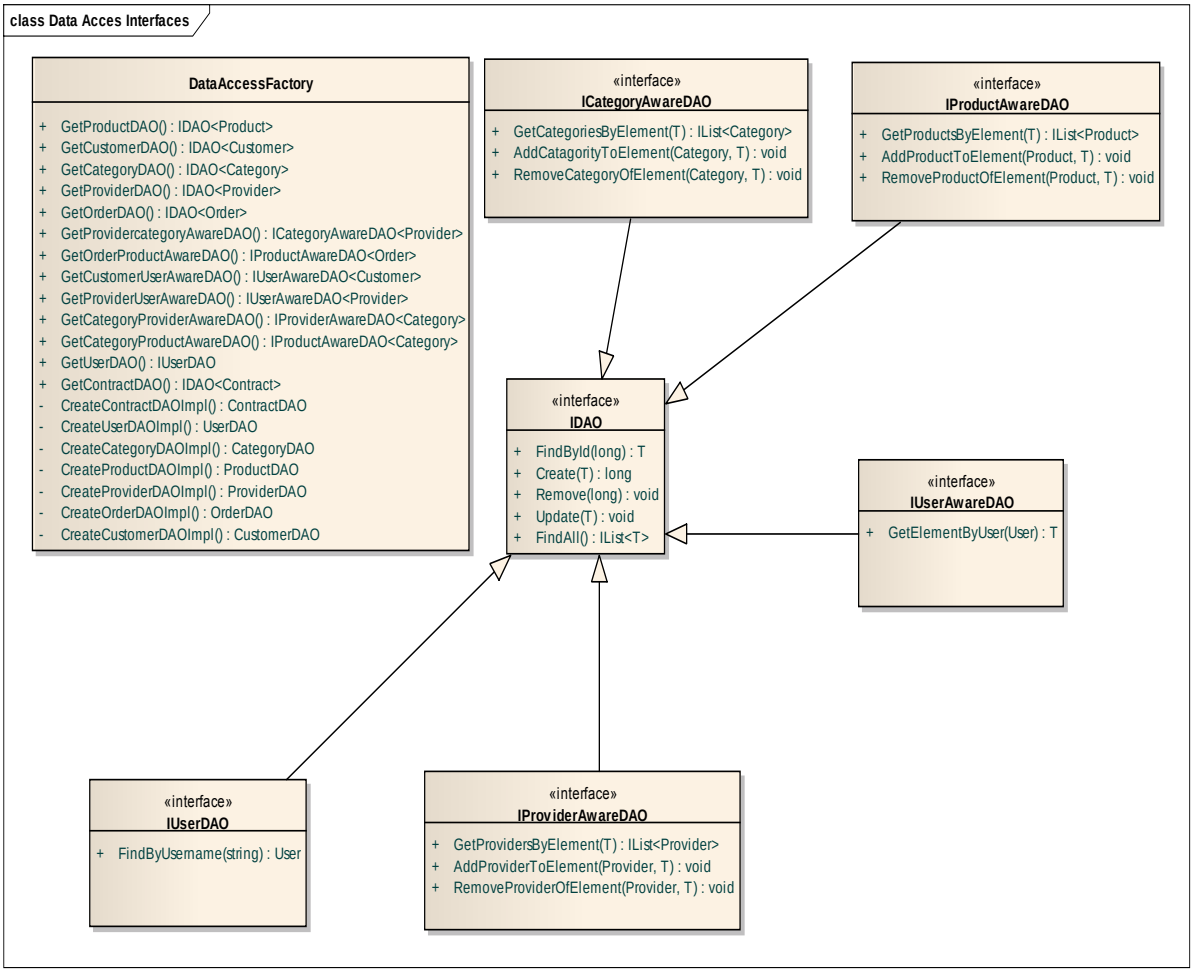
Diseño en el Nivel intermedio

Aquí se distinguen 4 módulos Modelo, Acceso a Datos, Negocio y Servicios

Modelo


class Class Model
User
- id: long
- username: string
- password: string
Provider
- id: long
- name: string
- surname: string
- mail: string
- phone: string
- address: string
+ AddCategory(Category) : void
IEquatable
Category
- id: long
- name: string
- description: string
+ addProvider(Provider) : void
+ addProvider(Product) : void
Contract
- id: long
- provider: Provider
- product: Product
- stock: int
Customer
- id: long
- name: string
- surname: string
- mail: string
- phone: string
- address: string
Order
- id: long
- dateOfBuy: DateTime
- dateOfDeliver: DateTime
- delivered: Boolean
Product
- id: long
- name: string
- stock: int
- description: string
- price: decimal

Acceso a Datos

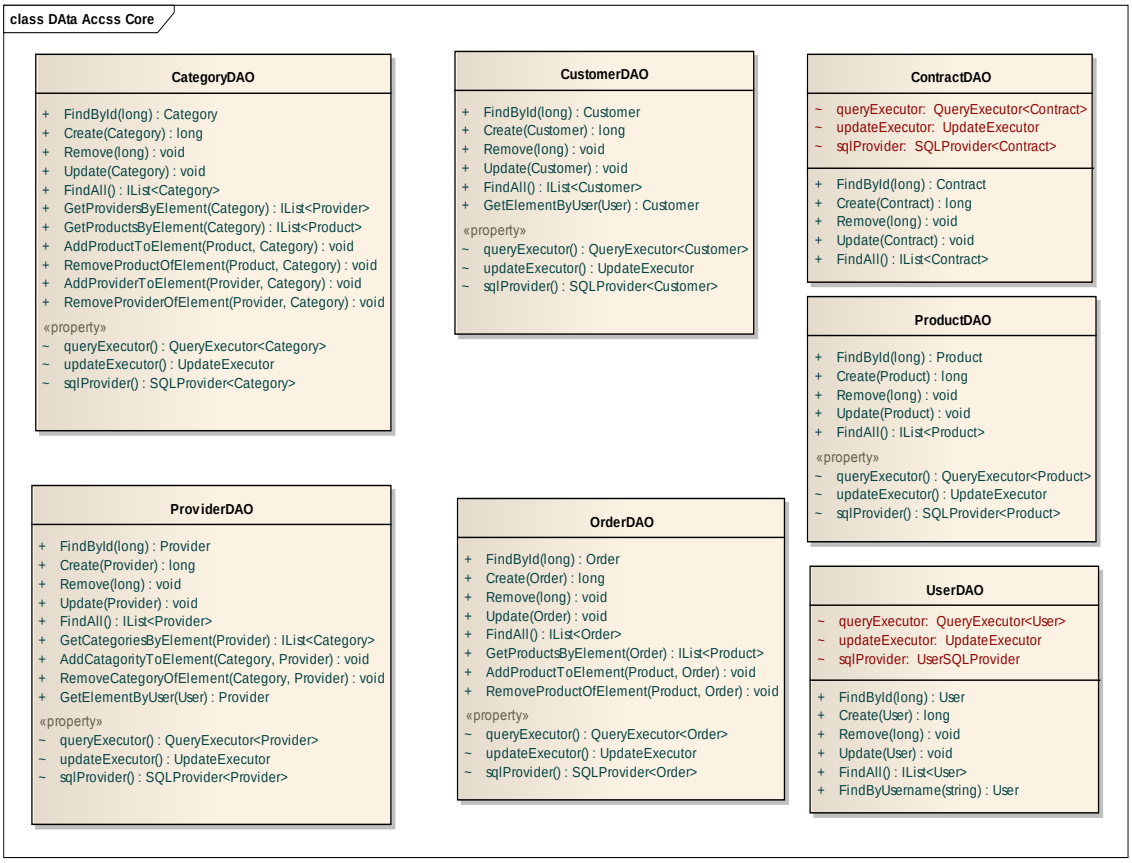


Diseño Nivel Intermedio 10

En el diagrama de arriba se puede ver una factoria y una serie de interfaces, la interfaz padre es IDAO<T>, esta define las operaciones básicas de un DAO para una entidad T, las implementaciones de IDAO<User> aplicaran estas operaciones a objetos del modelo User.

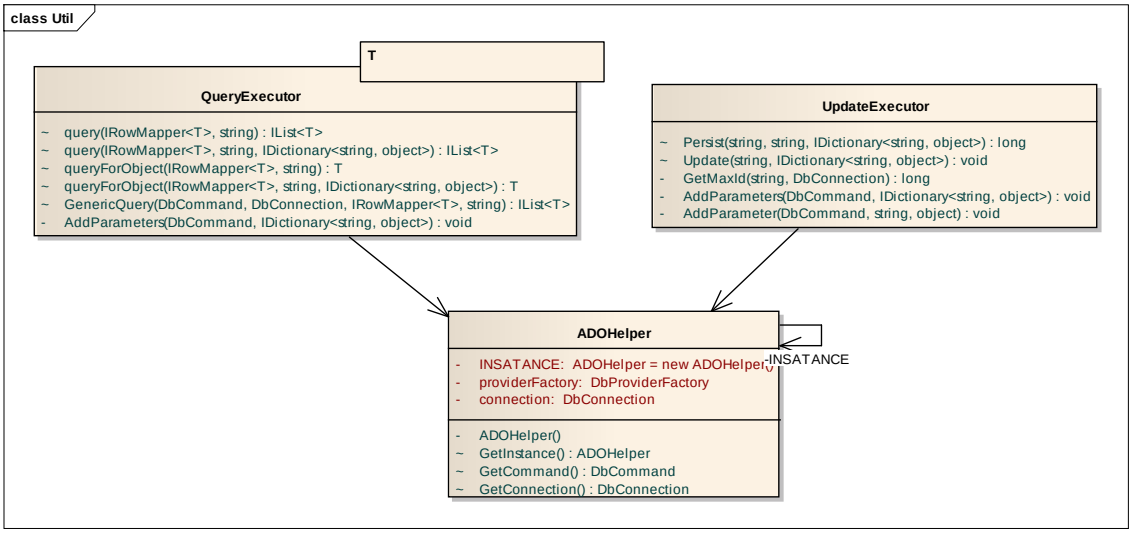
El resto de interfaces extienden la funcionalidad de IDAO y añaden algo más, por ejemplo IProviderAware hace a un DAO<T> caulquiera consciente de la entidad Provider, esto es que define operaciones del estilo GetProviderByT.

Las implementaciones de estas interfaces que son en cualquier caso las que se ven desde arriba son las siguientes:



Diseño Nivel Intermedio 11

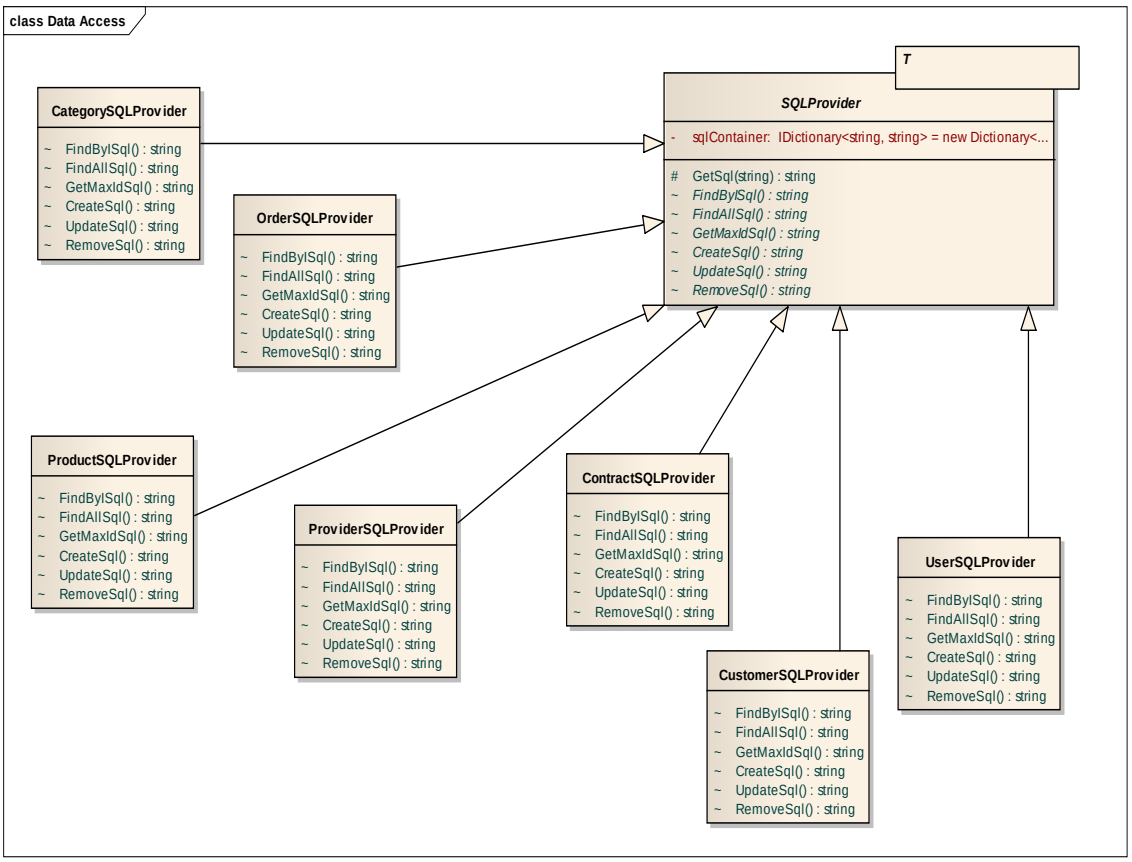
Para obtener acceso a la infraestructura de ADO.NET estas clases usan a las siguientes definidas en el nampespace Util.



Diseño Nivel Intermedio 12

Otro namespace de utilidad de esta capa es el Sql, este define una clase abstracta genérica con las operaciones para recuperar el Sql para cualquier entidad, desde el fichero de configuración,

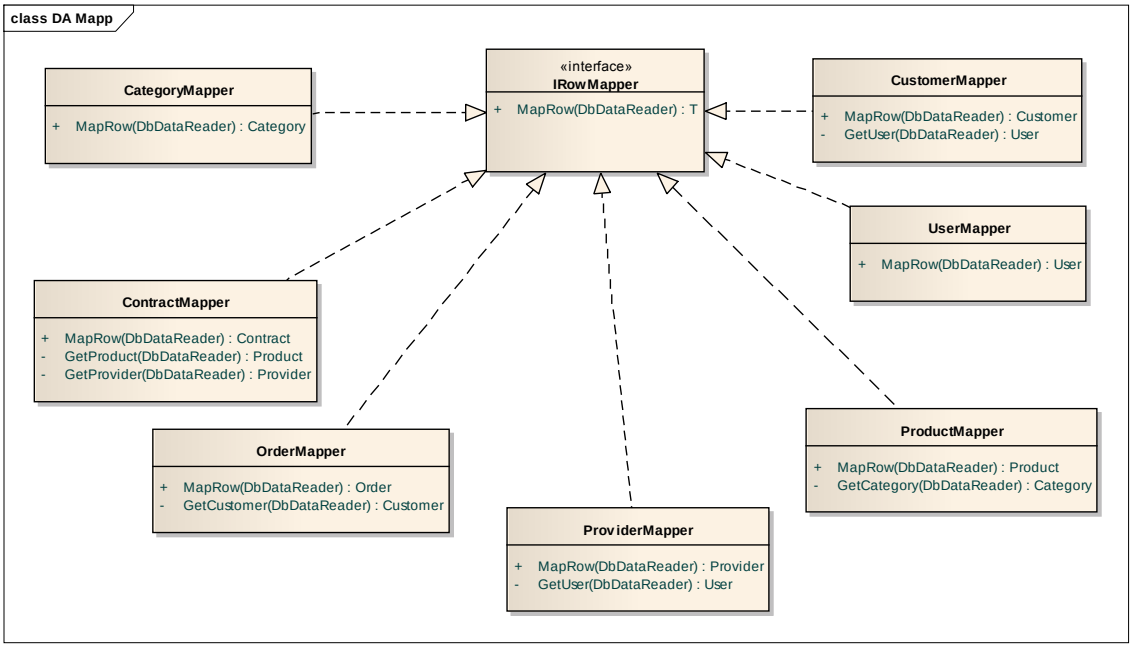
de esta forma no tenemos el Sql en código, además esta clase abstracta implementa cacheo de Sql para no perder rendimiento en el acceso a fichero.



Diseño Nivel Intermedio 13

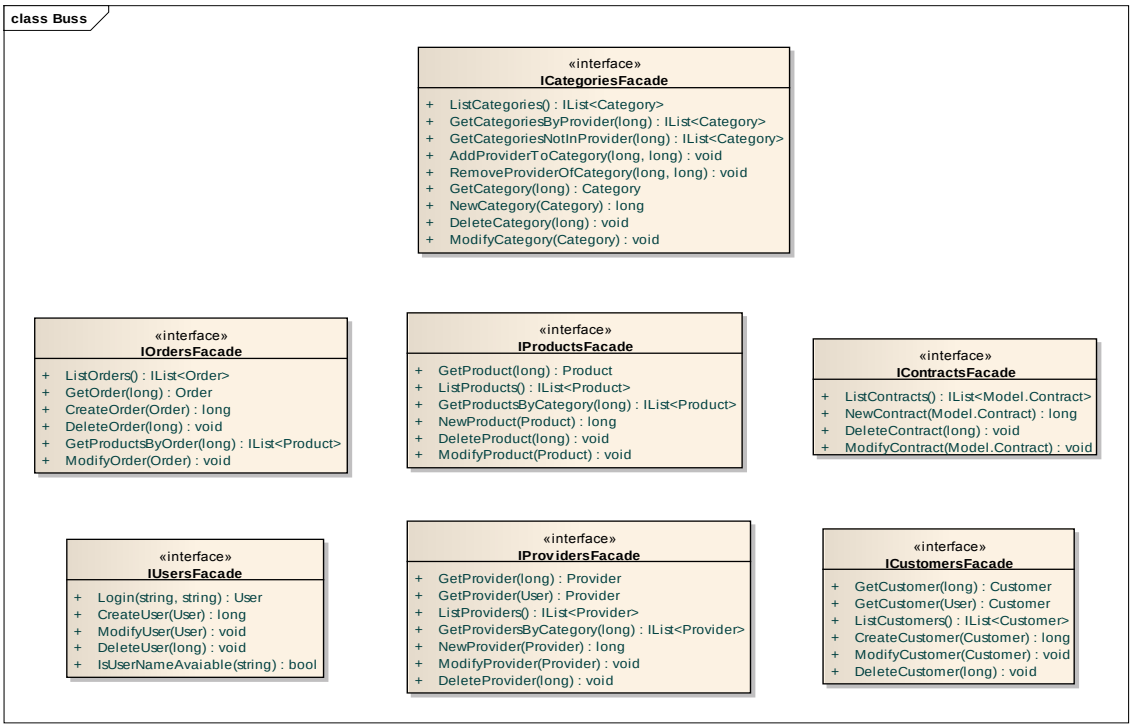
Cada una de las implementaciones sabe recuperar Sql para su entidad, el OrderSqlProvider sabe como recuperar Sql para encontrar por id pedidos. Para las operaciones más concretas existe un ConcreteSQLProvider.

Con la misma filosofía se definen las clases del namespace Mapping, que se encargan de mappear el objeto devuelto por ADO.NET de una consulta en objetos del modelo.



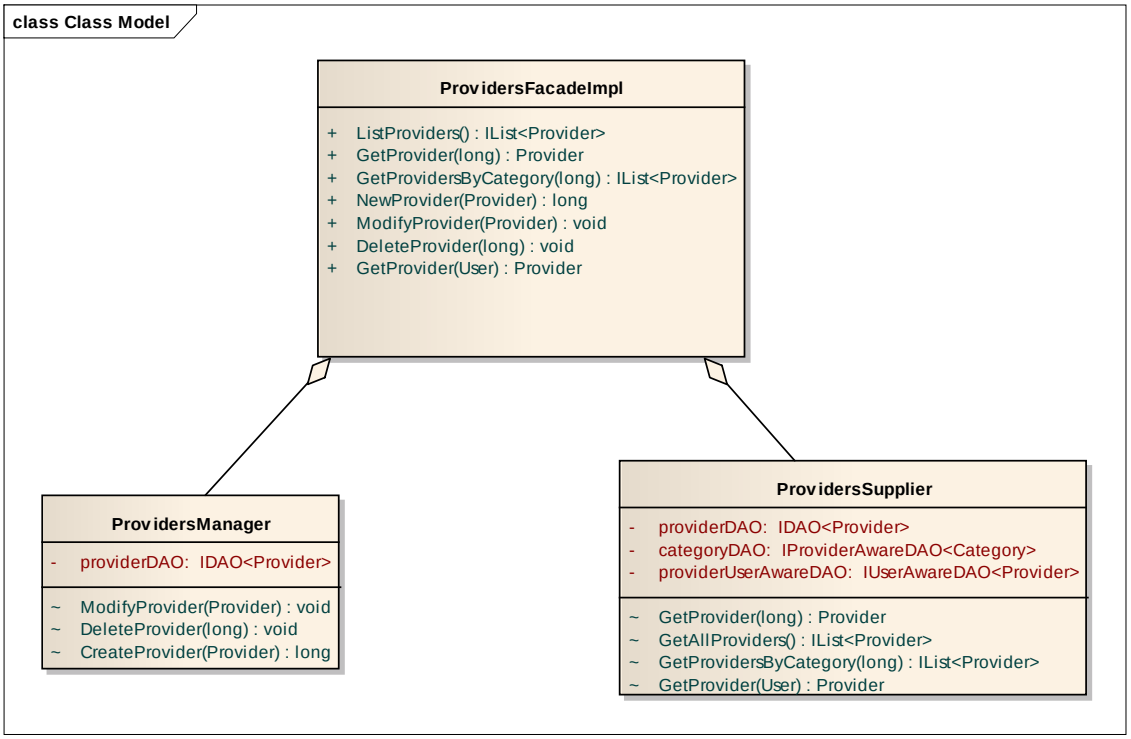
Negocio

En el diseño de la capa de negocio he utilizado un patrón Fachada para encapsular bajo varias fachadas las relaciones y la lógica de dicha capa, el principio del patrón fachada es unificar bajo cierta funcionalidad un conjunto más complejo de relaciones, pero en este caso dado la extensión de la fachada la he divido en varias fachadas por secciones, siempre procurando que estas desacoplen el resto de la capa de negocio de las capas superiores.



En general las implementaciones de estas fachadas delegan las operaciones a objetos de negocio, o en casos muy sencillos directamente a una objeto de acceso a datos.

Para ilustrar esto y no cargar con diagramas repetitivos pondré un ejemplo



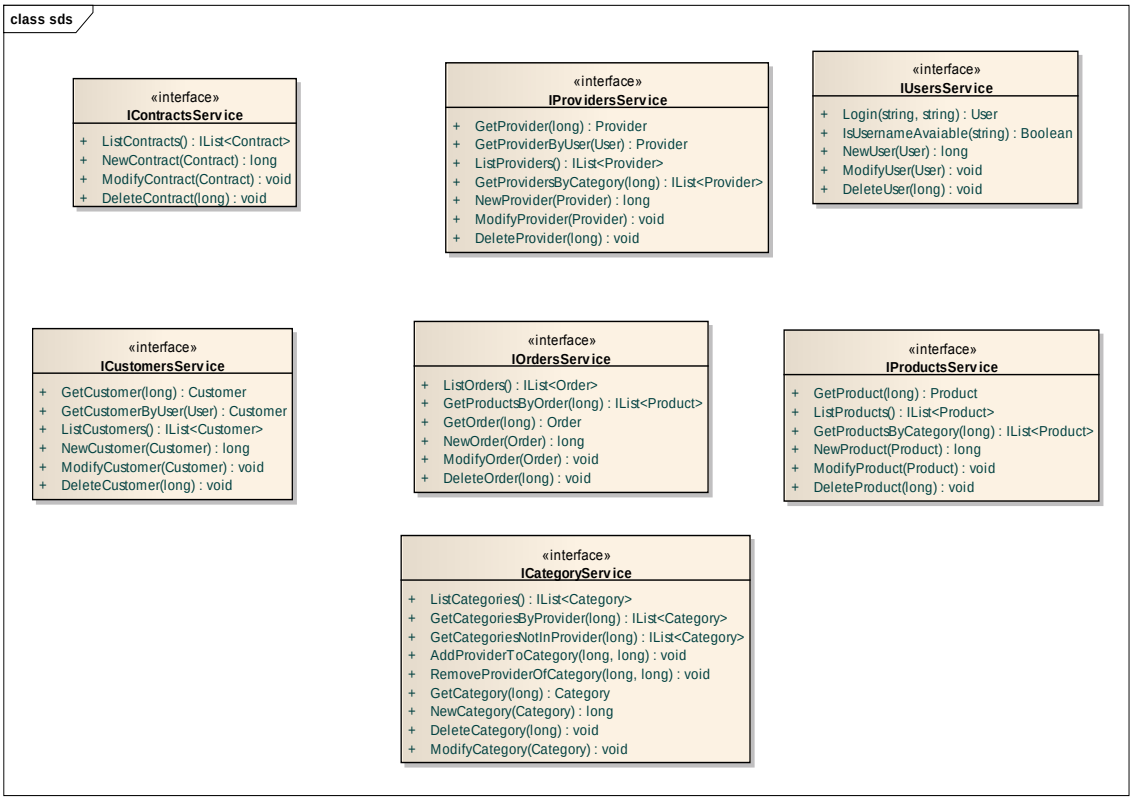
Diseño Nivel Intermedio 16

Tanto los objetos que agrega la implementación de la fachada como los DAOs agregados a su vez por estos son inyectados por Spring.

Tengase en cuenta que en este punto se podría estar accediendo a distintos SGBD, consumiendo servicios y ejecutando lógica mucho más compleja.

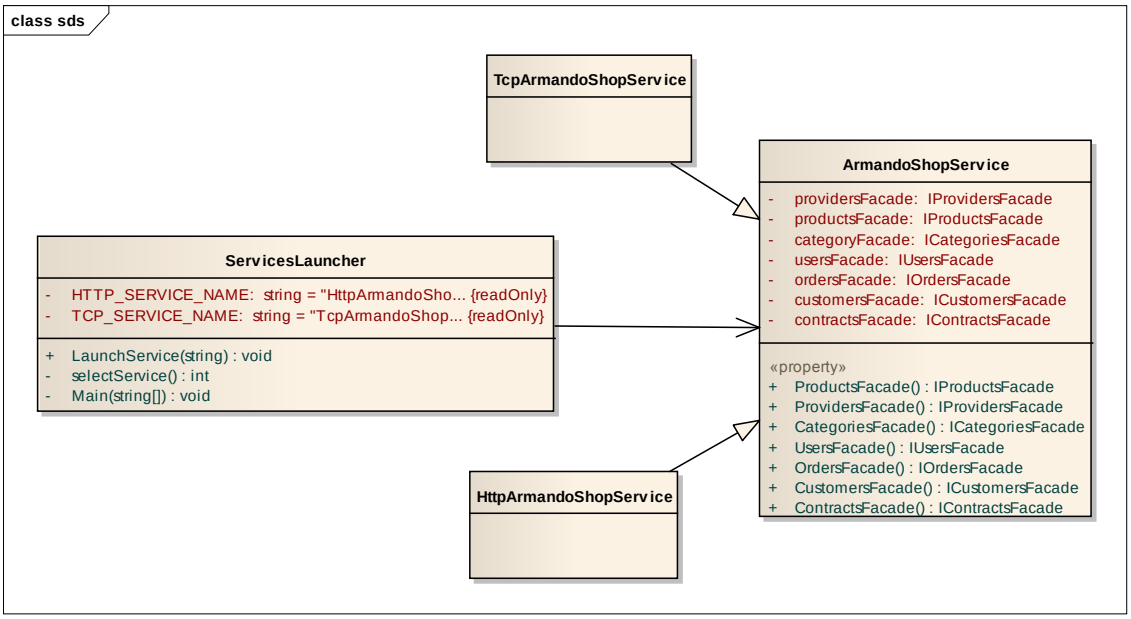
Servicios

La capa de servicios en los sistemas en N niveles suele tener la función de encapsular la funcionalidad ofrecida por las fachadas de negocio en servicios accesibles desde otros puntos, aquí es donde flaquea mi diseño, he expuesto toda la funcionalidad por HTTP y TCP cuando aquí tendríamos que diseñar servicios o contratos específicos para los clientes y exponer solo lo necesario.



El modelo de programación de WCF nos permite exponer servicios por solo definirlos en interfaces anotadas como contratos, el resultado es que los clientes tienen acceso a interfaces como las que se ven arriba.

Para exponer dichos contratos WCF nos exige que implementemos todas las interfaces en una clase que será la que expongamos como servicio, para limpiar el código separe dicha clase en partials clases, el siguiente diagrama muestra la central, y dos hijas que uso para distinguir un servicio como HTTP y otro como TCP.



Diseño Nivel Intermedio 18

Diseño del Cliente de Gestión

Como ya dije antes este cliente sigue un patrón MVVM, el siguiente diagrama muestra los modelos de las vistas, cada una de las siguientes clases ofrece propiedades y operaciones encapsuladas en Commands a sus respectivas vistas.

Diseño Nivel Superior 19

Las clases de las vistas no las añado ya que entiendo que solo aportan ruido, estas clases piden la funcionalidad a los Delegados de la capa de negocio, con lo cual constituyen toda la complejidad de la aplicación.

Diseño del Cliente de Proveedores

Al igual que en la aplicación anterior toda la dificultad recae en el modelo de la vista que hace llamadas a los servicios adecuados mediante los Delegate*Services.

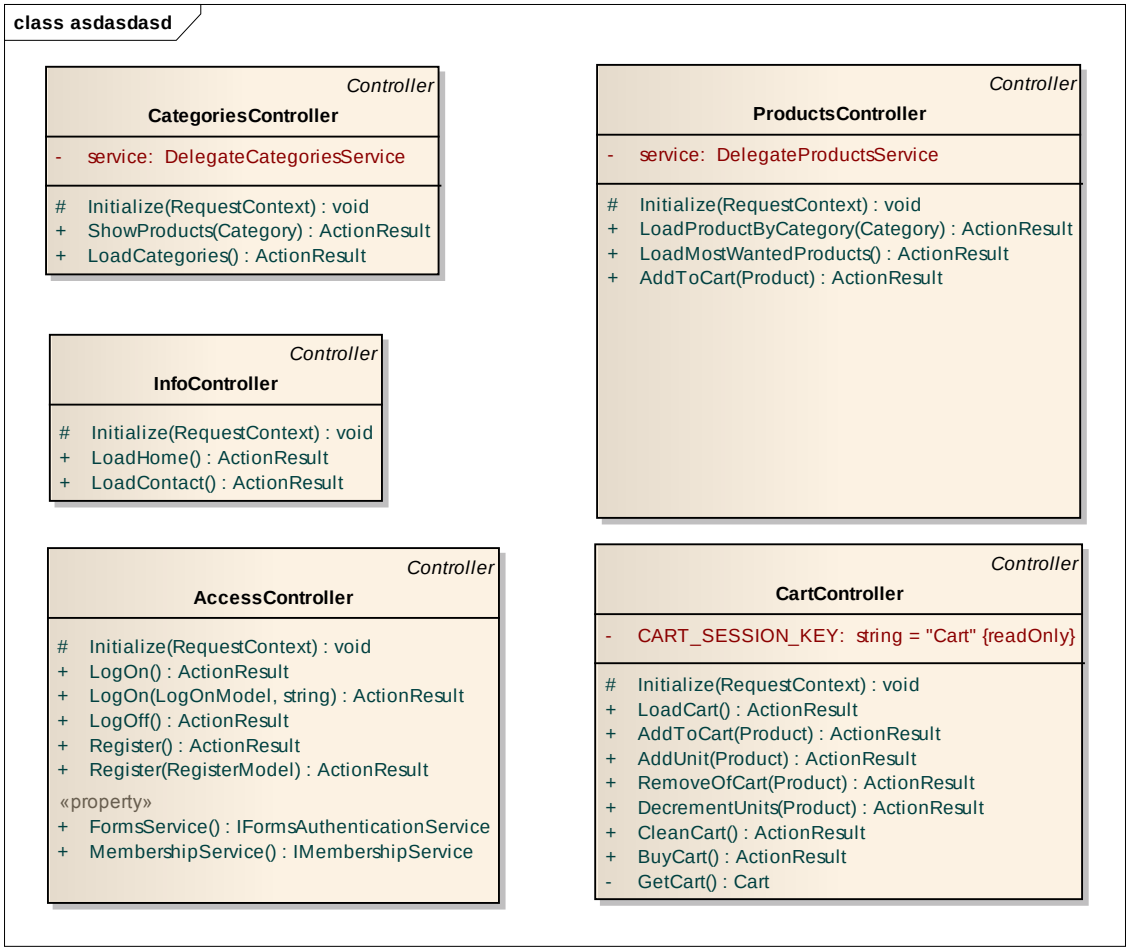
Diseño Nivel Superior 20

Debido a que la aplicación es mucho más sencilla, tiene menos vistas, luego menos modelos de la vista.

Diseño de la Aplicación Web

Esta aplicación sigue un patrón MVC, esto significa que tiene tres componentes las Vistas, los controladores y el modelo. En este punto nos encontramos con algo similar a las aplicaciones anteriores, las vistas no llevan lógica, luego no pondré sus clases.

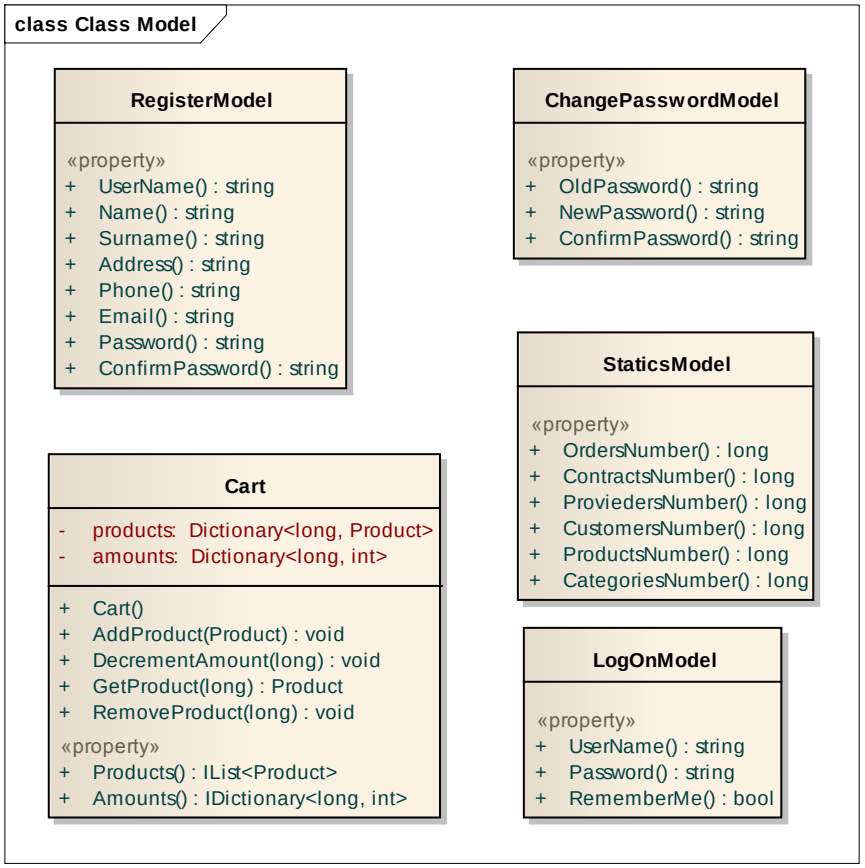
Controladores



Modelo

A Diferencia de las aplicaciones de escritorio en este caso el modelo no consta solo de los delegados de servicios, se han añadido además otras clases que constituirían un modelo de presentación (no el modelo de MVC – (V+C)) si no el modelo que solo modela, como el de negocio, pero en este caso de presentación.

Lo que nos viene a facilitar este modelo, es que en las vistas de ASP.NET MVC se puede obtener acceso a objetos del modelo de una forma muy limpia, con lo que en muchos casos resulta interesante encapsular ciertos campos en un pequeño objetos como LogOn.



Diseño Nivel Superior 22

Despliegue

En este apartado comentare los contenidos aportados asi como su uso.

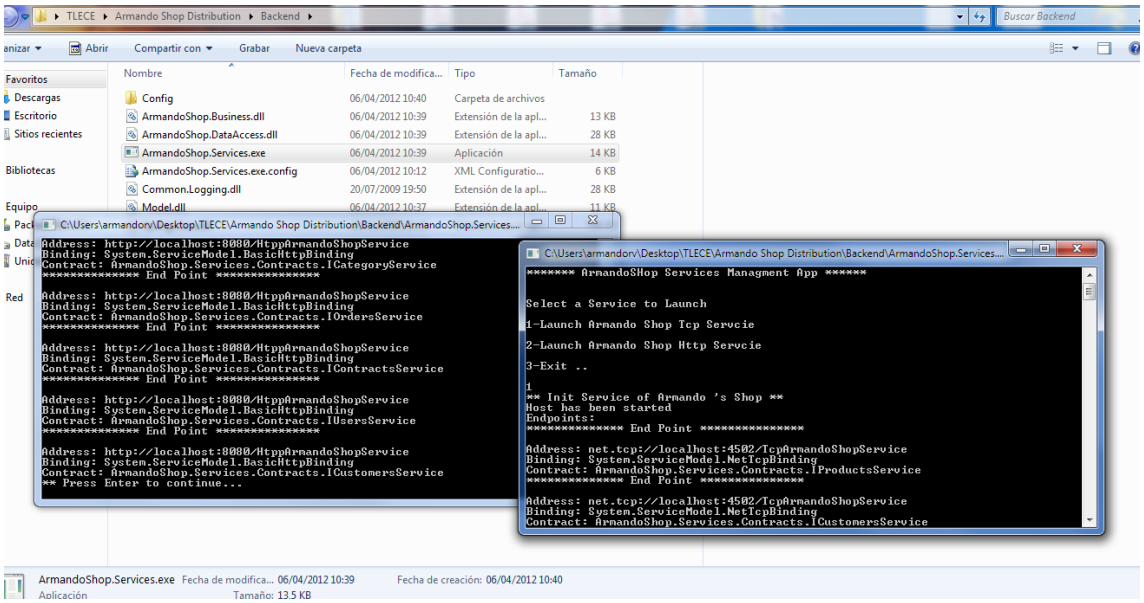
En el directorio raíz además de este documento y la presentación se encuentran dos directorios, uno con los contenidos de desarrollo que se exploran en el apéndice 2, y otro con la distribución del sistema, que es la que explicare aquí como desplegar.

Los requisitos para este sistema son .NET v4.0.

Desplegando Servicios

Para poner a funcionar nuestros servicios en **modo prototipo**, solo tenemos que ejecutar el ejecutable que se encuentra en Armando Shop Distribution\Backend, este nos ofrece dos opciones, exponer mediante TCP y mediante HTTP, debemos seleccionar TCP para el cliente de gestión y la aplicación web y HTTP para el cliente de proveedores, importante, la opción 2 solo funciona si ejecutamos como administradores, por tanto botón derecho ejecutar como administrador.

Una vez hecho esto tenemos 2 aplicaciones de consola ejecutándose con el pequeño matiz de que alojan sendos servicios de WCF, recordemos que son hosting agnostic, pueden ir en cualquier app domain.

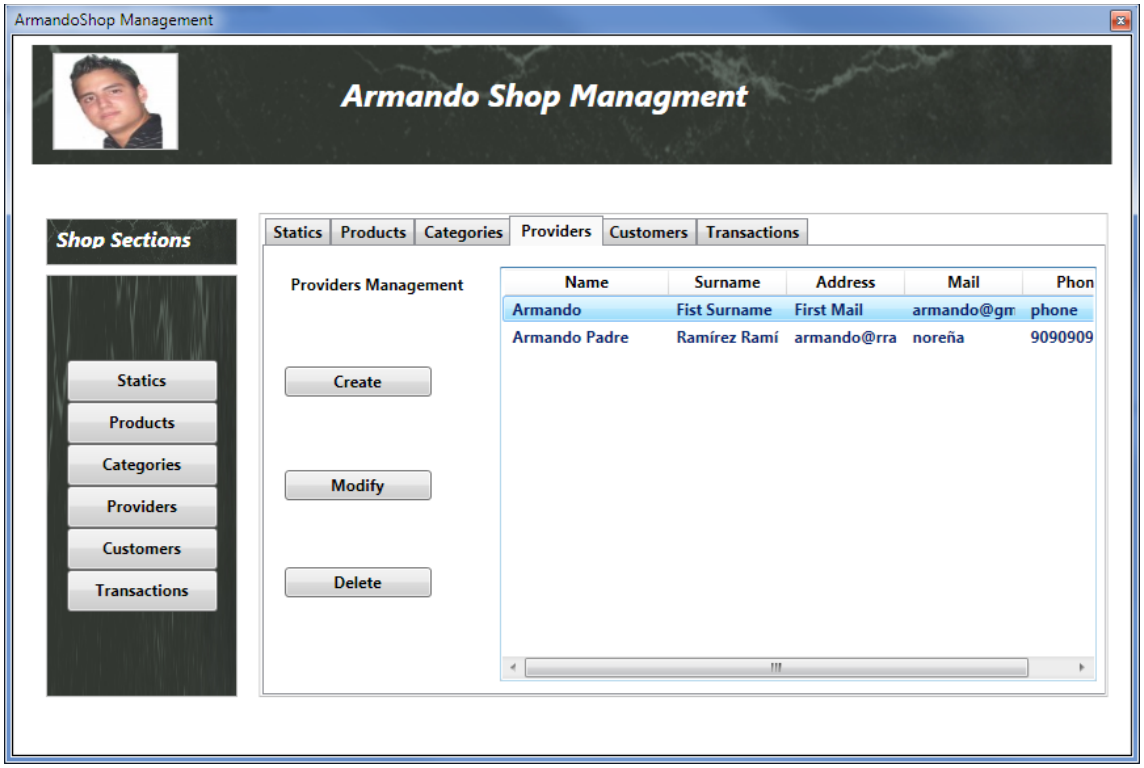


Despliegue 23

Ejecutando Clientes de Escritorio

Para ejecutar los clientes de escritorio basta con ir a la carpeta adecuada, Armando Shop Distribution\Managment Client para el cliente de gestión o C:\Users\armandorv\Desktop\TLECE\Armando Shop Distribution\Providers Client para el cliente de proveedores y hacer doble click, si asi de fácil.

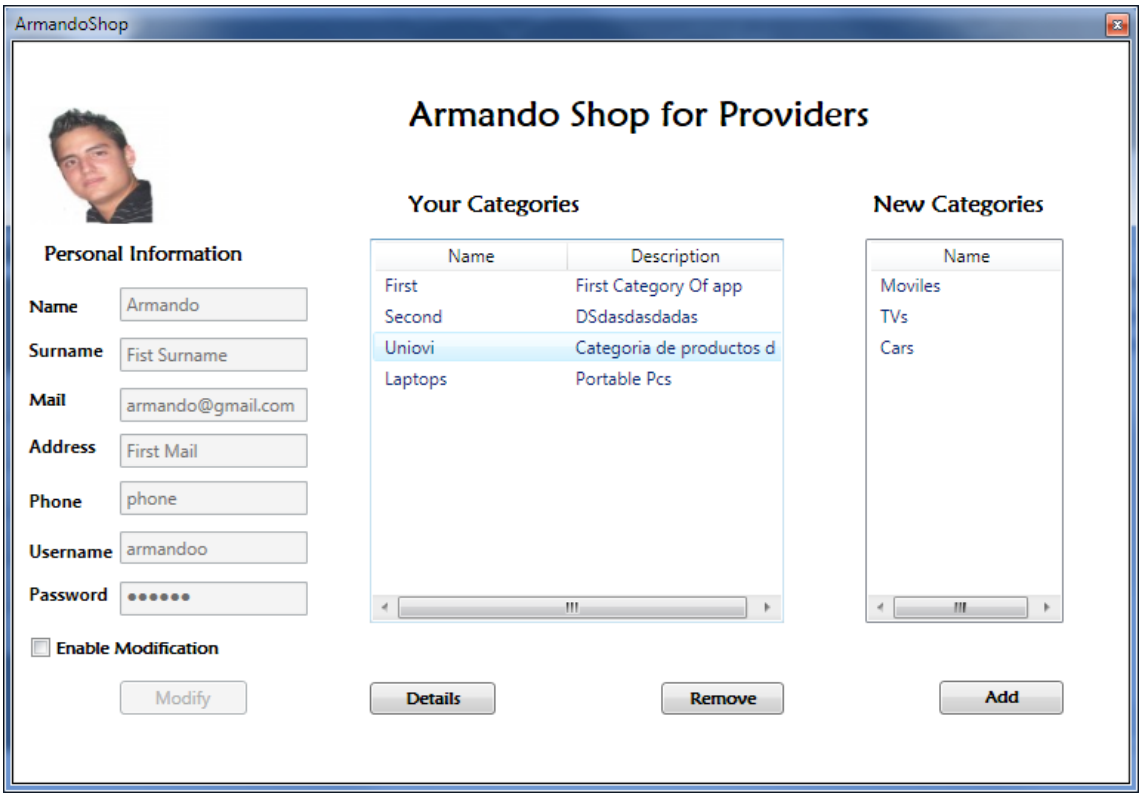
Para obtener acceso al cliente de proveedores abrir primero el de gestión y ver datos.



Despliegue 24

Si damos en modify veremos sus datos, no está bien pero estamos en modo prototipo.

Ahora podemos entrar al cliente de proveedores con los datos adecuados.



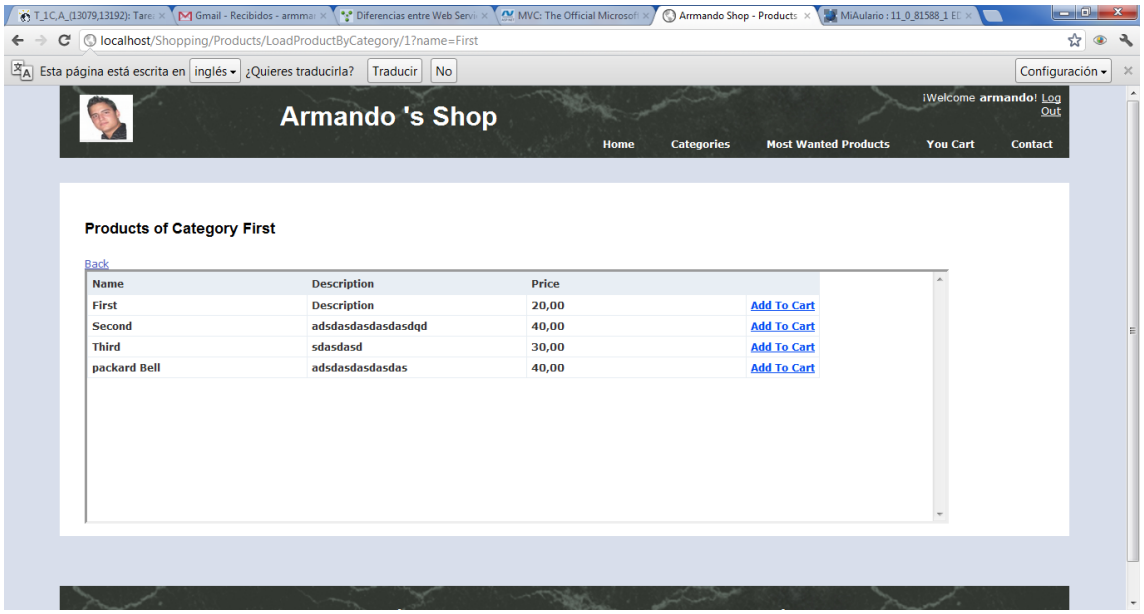
Despliegue 25

Desplegando Aplicación Web

Antes de dar este paso recomiendo leerse el Apéndice 1, ya que para desplegar la distribución debe hacerse en una aplicación de IIS como la descrita allí.

Una vez se tenga la aplicación creada debería bastar con copiar el contenido de Armando Shop Distribution\Web Shop en el directorio de la aplicación.

Si esto da problemas, véase apéndice dos para ejecutar en el servidor de pruebas.



Despliegue 26

Base de Datos

En principio para probar el prototipo se ha incluido una base de datos de SQL Server Compact Edition, que está en el directorio Armando Shop Distribution\DataBase, no se debe mover de este directorio ya que los servicios la referencian relativamente.

No obstante para cambiar el SGBD bastaría con modificar el Fichero de configuración adecuado.

Bibliografía

- Andrew Troelsen (2008).Pro C# 2008 and the .NET 3.5 Platform. Appress.
- Matthew MacDonald, Adam Freeman, Mario Szpuszta (2010) Pro ASP.NET 4 in C# 2010.Appress.
- Aroa Solana (2011). Windows Comunication Foundation 4.0.Luarna.
- Aroa Solana (2010) Desarrollo de Aplicaciones Windows con WPF 4.0.Luarna.
- Microsoft Application Architecture Guide 2nd Edition (2009).
- Spring .Net Reference Documentation (2009)
- MSDN

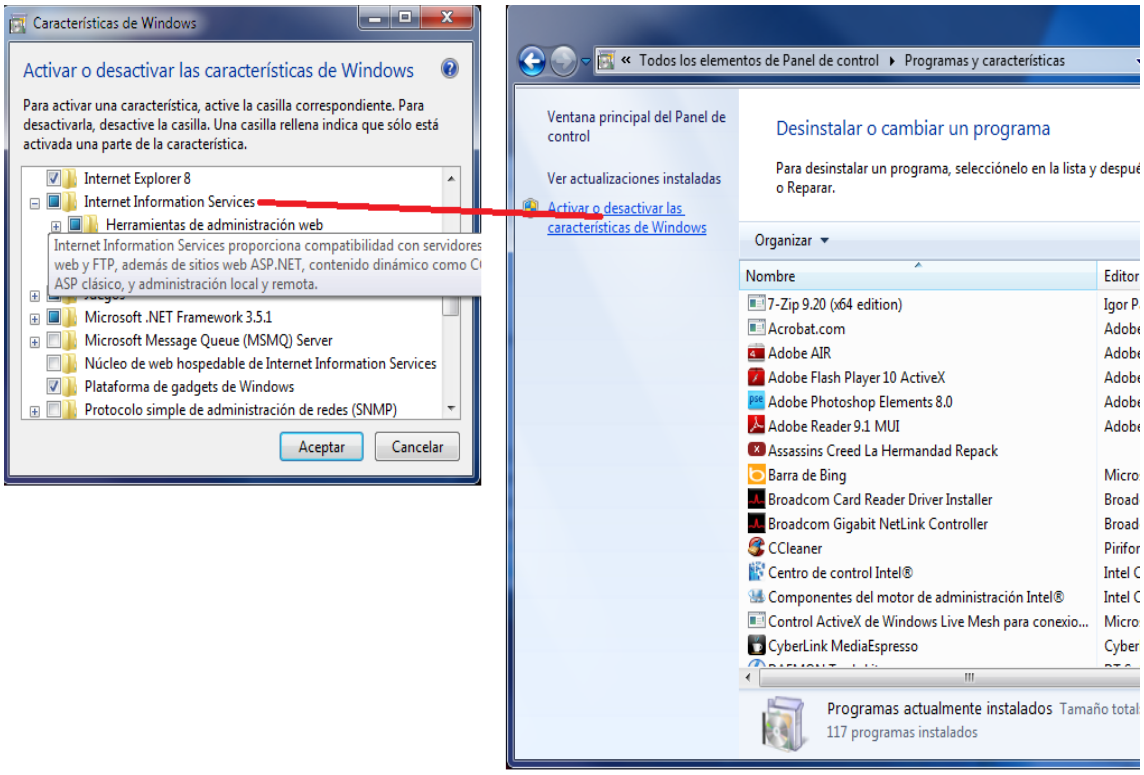
Apéndices

Apéndice 1: Preparando IIS

Parte de este sistema está pensado para ser expuesto mediante un Servidor de IIS, este servidor es un producto muy completo con muchas características que viene integrado con Windows, y se puede activar sin mayores problemas, yo he probado con IIS 7 y tengo entendido que del 6 cambia bastante.

Activando IIS

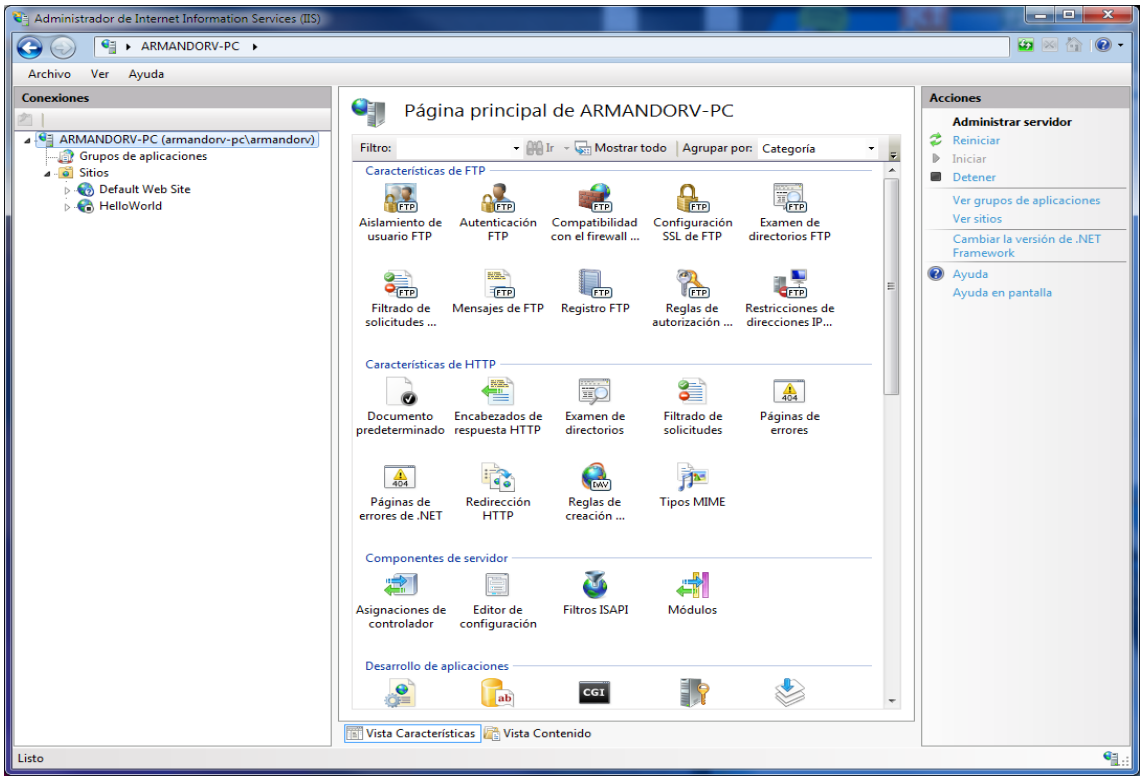
Para activar el producto debemos ir al panel de control de Windows y seleccionar Programas Y Características (La de desinstalar programas) y una vez allí activar o desactivar características de Windows.



Activando IIS 27

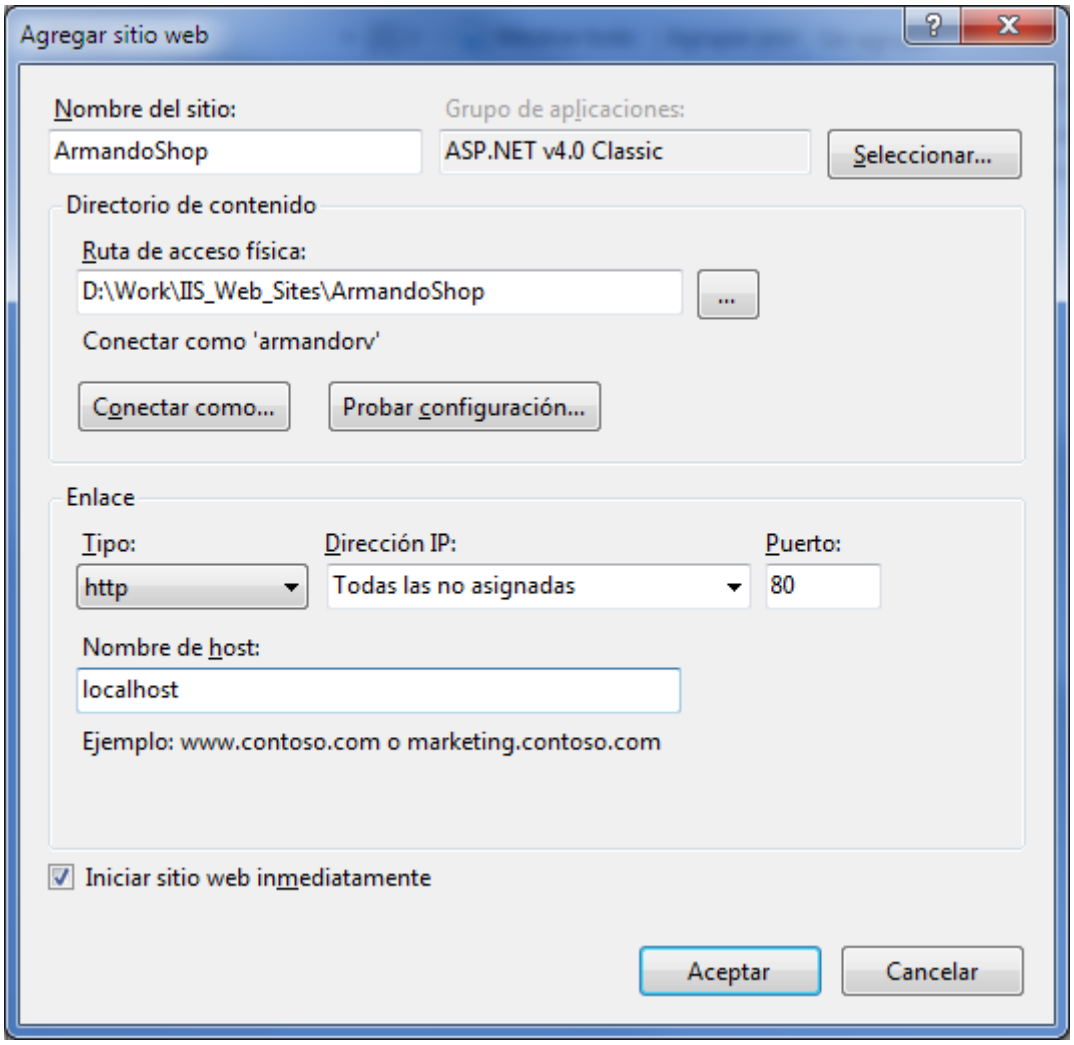
Con esto ya tenemos el IIS funcionando.

Ahora es interesante entender un par de conceptos, IIS te permite definir “Sitios”, un sitio se correspondería con un sitio web entero (Uniovi.es), el hecho de que podamos definir varios significa que para una maquina podemos cambiarlos.



Activando IIS 28

Por defecto tenemos un sitio, pero yo prefiero crear uno para este sistema:

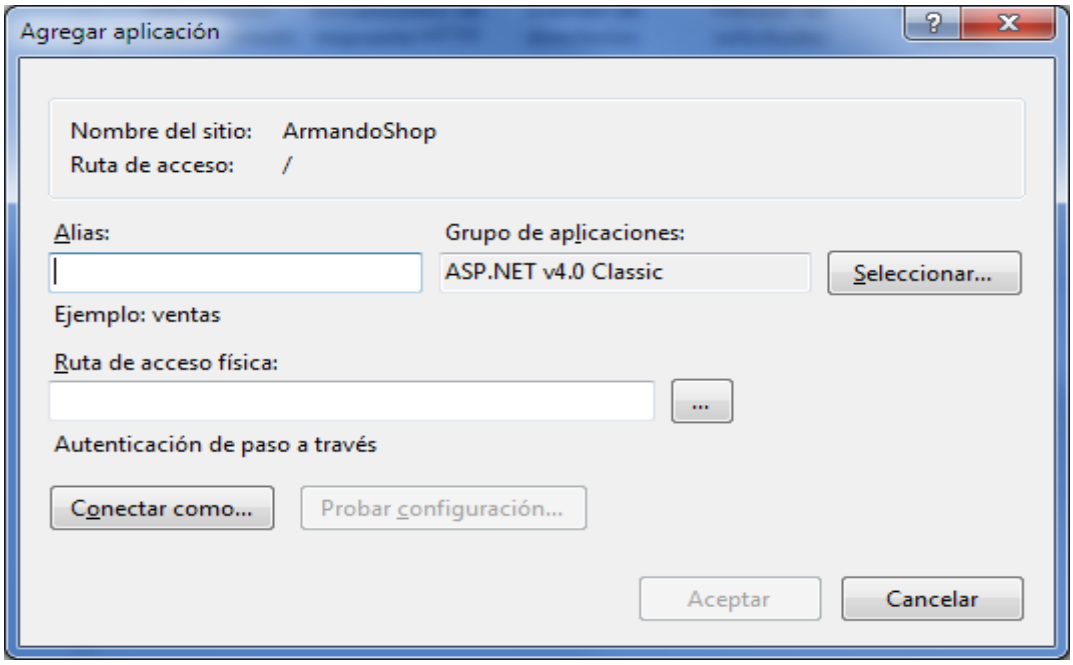


Activando IIS 29

Con botón derecho en sitios podemos agregar uno, tendremos entonces un dialogo como el anterior, donde es importante seleccionar en conectar como un usuario que este en el sistema operativo, el directorio del sitio es donde estarán los ficheros de las aplicaciones que tenga el sitio y el nombre es el nombre lógico por el que IIS lo identifica.

Otro punto que deberíamos hacer es crear una aplicación en el sitio, esto es Uniovi.es/Catalogo, pues catalogo seria nuestra aplicación.

Grupo de aplicaciones, para este sistema debe ser ASP.NET v4.0 Clasic, en conectar como se debe indicar un usuario de Windows valido.



Activando IIS 30

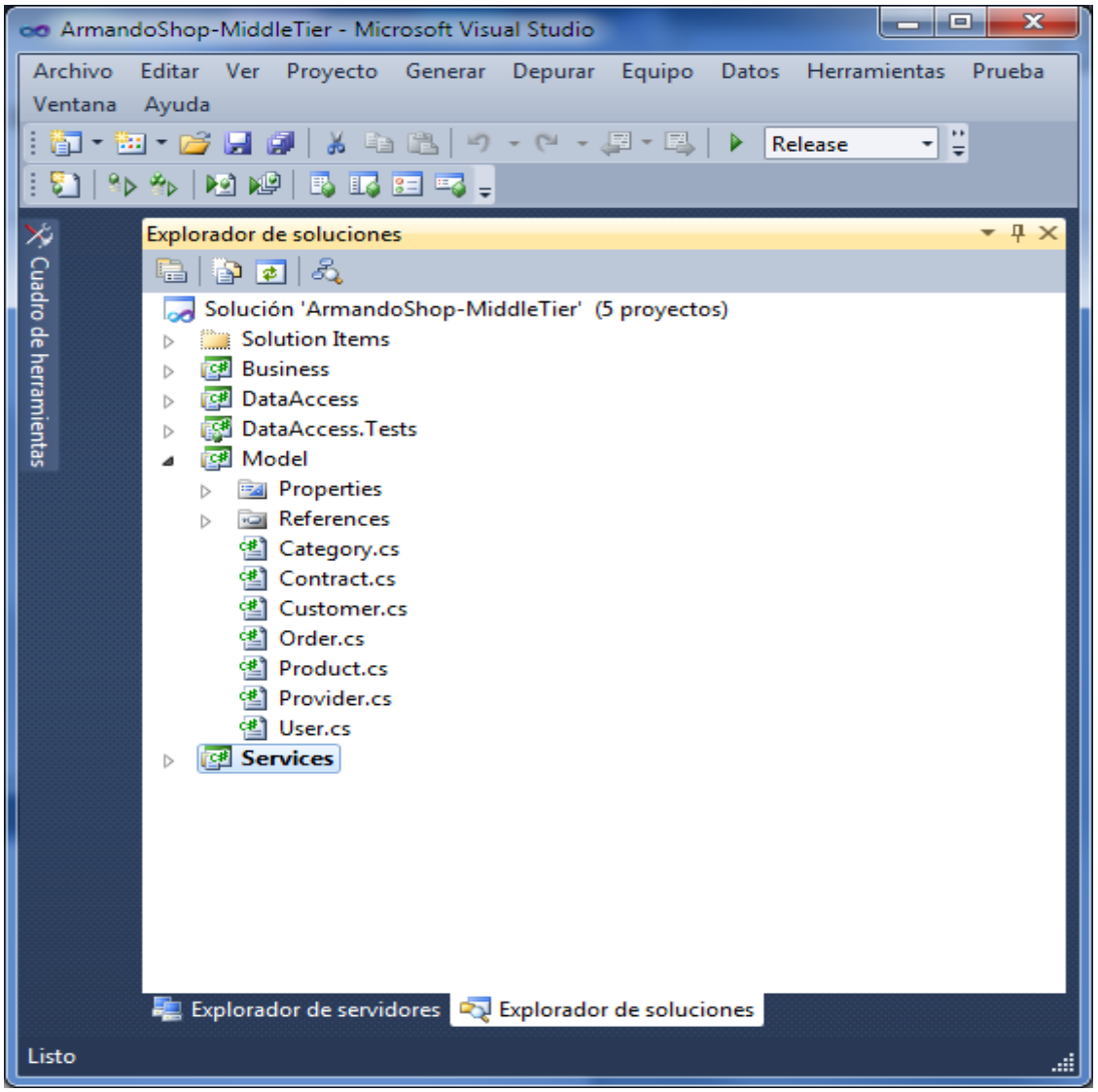
Una vez creada la aplicación la deberíamos poder ver en el sitio, y dentro del directorio de esa aplicación podremos dejar nuestra distribución web.

Como nota preventiva, asegurese de que en la sección IIS restricciones ISAPI y CGI están todas permitidas.

Apendice 2: Explorando el Proyecto

En este apéndice explicare como explorar los ficheros del directorio Armando Shop Development. En dicho directorio hay otros dos, cada uno corresponde a una solución de Visual Studio 2010, para explorarlos al mismo tiempo hacer doble click en el fichero .sln que hay en cada carpeta.

Explorando el Nivel Intermedio



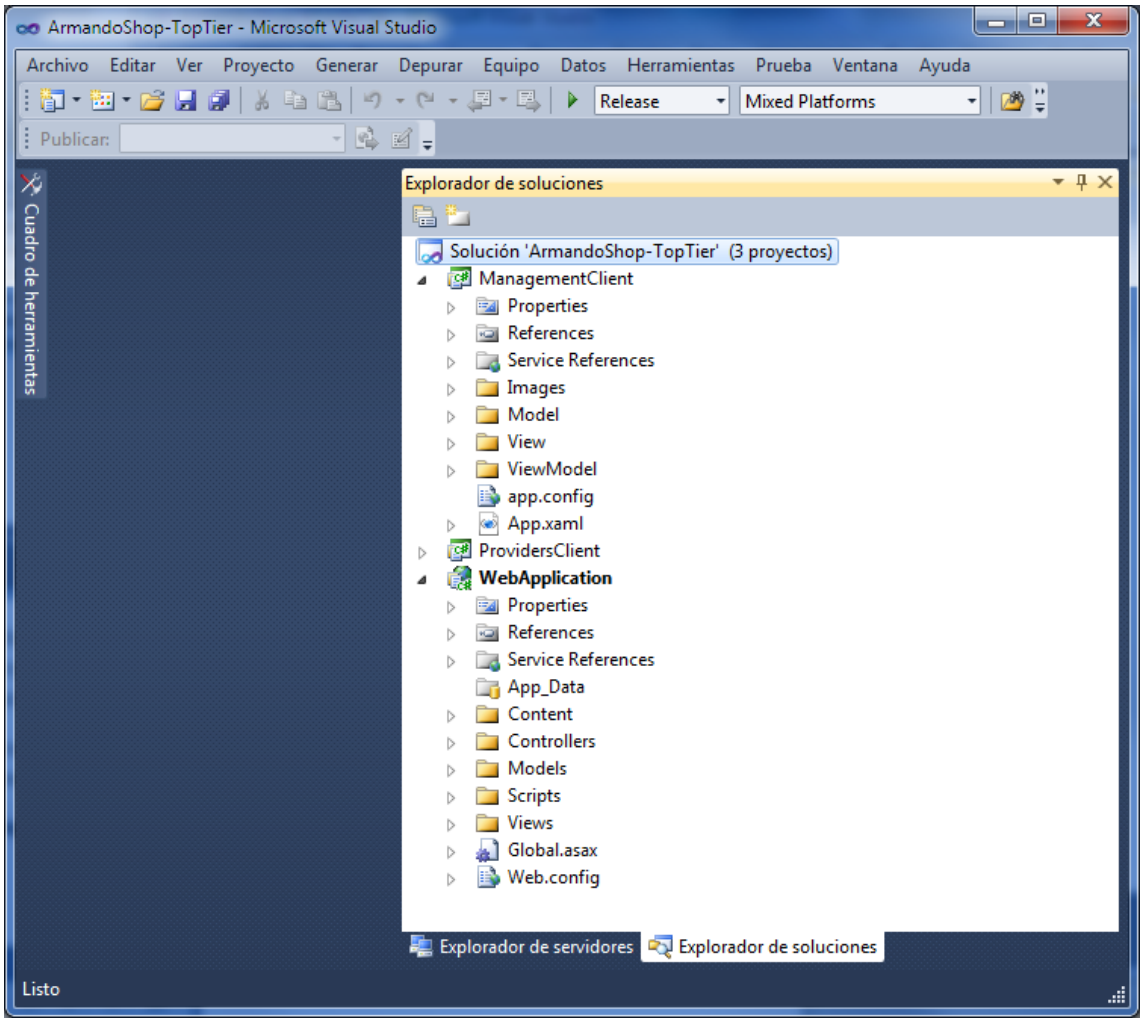
Explorando el Nivel Intermedio 31

En esta solución se incluye un proyecto por cada capa del nivel intermedio, y se ha seleccionado como proyecto de inicio el de services, si ejecutamos con la flecha verde debería arrancarnos la aplicación de consola para elegir el servicio.

En esta solución la única opción válida como proyecto de inicio es la seleccionada ya que el resto son librerías, el producto obtenido de ellos es una ddl.

Si vamos al explorador de servidores podremos explorar el fichero incluido como base de datos, esto es ver las tablas, modificar esquema , todo.

Explorando el nivel superior



Explorando el Nivel Superior 32

En este caso análogamente al anterior tenemos una solución con varios proyectos, uno por aplicación.

Mucho Ojo aquí, si no pudimos desplegar en IIS, y ejecutamos el proyecto WebApplication nos lo desplegara en el servidor de desarrollo y esto siempre suele funcionar, salvo errores de fuerza mayor.

Por lo demás es todo a jugar con el código.