# *DESENVOLUPAMENT AMB TECNOLOGIA MICROSOFT .NET FRAMEWORK*

## *ARQUITECTURA ASP .NET*

### *APLICACIÓ DE COMERÇ ELECTRÒNIC*

*U.O.C. Treball Fi de Carrera. 10 de gener de 2005*

**Manel MORON ARNAU**
Enginyeria Tècnica en Informàtica de Sistemes
**CONSULTOR: JORDI CEBALLOS VILLACH**

***A l'Anna, que ha compartit amb mi els bons moments i els no tan bons.***

***Al Lluís, que de ben segur després d'aquest treball podré ajudar.***

**RESUM**

Tot i que el cos d'aquest treball es refereix a la creació d'una botiga virtual per a la venda de bicicletes i altres components relacionats, l'objectiu terminal és conèixer l'arquitectura Asp.Net –integrada dins de la tecnologia Microsoft .NET Framework– i aprofundir en les seves característiques.

Així doncs, prenent com a punt de partida l'anàlisi, disseny i implementació d'aquesta aplicació de comerç electrònic es fa una explicació detallada de tot el procés que inclou una breu descripció de la plataforma.

S'explica també l'infrastructura pel desenvolupament, el mètode seguit i el patró que s'ha utilitzat, tanmateix es parla de la seguretat emprada per a la part privada de l'aplicació i del model de dades que s'ha fet servir.

El producte obtingut és una aplicació web de comerç electrònic que està dividida en dues parts, per una banda la botiga virtual i per una altra l'administració a la que només hi poden accedir els usuaris autenticats.

Encara que no està dins dels objectius principals del projecte també es fa una breu descripció del procés del disseny gràfic ja que es considera que està força lligat a altres aspectes del projecte, sobretot considerant que pel desenvolupament s'ha utilitzat l'entorn integrat Microsoft Visual Studio .NET.

Per acabar aquest resum, val a dir que el patró arquitectònic estructural que s'ha seguit per a crear l'aplicació, tot i que sempre es fa referència al *Model-View-Controller*, en aquest projecte n'és només una aproximació i que després d'una llarga recerca d'informació s'han trobat diverses divergències entre els diferents autors.

# Í N D E X

**Figures i gràfics**

# 1. MEMÒRIA

## 1.1. Introducció

### 1.1.1. Justificació.

Avui dia Internet està consolidat com un canal de comunicació entre les persones molt vàlid i tanmateix permet a l'usuari obtenir informació immediata amb agilitat i comoditat.
Aprofitant aquest canal, les empreses poden oferir la venda dels seus productes a través d'Internet i així estendre el seu mercat potencial.

Tot i que actualment la implantació d'aquest canal de venda encara no es pot comparar amb el tradicional –hi ha una certa reticència per part d'ambdues parts– és molt probable que en un futur no massa llunyà el comerç electrònic esdevingui una part força important del volum de vendes de les empreses.

En aquest context i dins del marc de Microsoft Framework.NET –l'objectiu principal del projecte és la introducció i aprofundiment en aquesta tecnologia– s'ha optat per la creació d'una botiga virtual emprant l'arquitectura ASP.NET pel desenvolupament i ADO.NET per a l'accés a la base de dades.

### 1.1.2. Punt de partida.

L'inici d'aquest projecte és l'estudi de la viabilitat d'un problema plantejat per una petita empresa amb clara tendència a l'expansió dedicada a la venda de bicicletes i els seus components.

L'empresa objecte de l'estudi està actualment en funcionament, disposa d'una pàgina web de presència i ha proposat la creació d'una botiga virtual on pugui oferir els seus articles.

Se li proposarà la creació d'una plana web activa dividida en dues parts:

- Botiga virtual on un visitant podrà convertir-se en un comprador que pot triar articles dividits per categories, incloure'ls en una cistella virtual, introduir les dades de lliurament i realitzar la compra.
- Una eina d'administració del web on un usuari amb privilegis –habitualment un empleat autoritzat– pugui fer el manteniment de la base de dades.

### 1.1.3. Objectius.

Per a aconseguir l'objectiu principal del projecte –l'aprofundiment en la tecnologia .NET– s'ha vinculat a un objectiu secundari, la solució del problema plantejat per l'empresa mitjançant un acurat anàlisi, i així mateix el disseny i implementació d'una aplicació de comerç electrònic que satisfaci els seus interessos.

Val a dir que un dels objectius –personal en aquest cas– també ha estat aconseguir la simplicitat i funcionalitat en el disseny de la plana web. Sovint es troben productes massa carregats amb publicitat i altra informació irrellevant que distreuen l'usuari i no en faciliten pas l'ús.

Així doncs amb aquest projecte es pretén fer una aportació –tot donant una visió personal– a la creació d'una botiga virtual emprant la tecnologia esmentada i descriure els problemes que vagin sorgint i la solució triada.

### 1.1.4. Enfocament i mètode seguit.

Com ja s'ha comentat, per a aconseguir l'objectiu principal del projecte l'enfocament està articulat sobre la solució d'un problema plantejat per una empresa.

En primer lloc el treball s'ha dividit en les etapes adequades per el desenvolupament de programari –anàlisi, disseny i implementació– per a estructurar tot el projecte.
En l'etapa d'anàlisi s'ha dut a terme la recollida de requeriments mitjançant reunions amb els responsables de vendes de l'empresa i a partir d'aquest estudi s'han detectat els objectes del domini del món real i les seves interaccions en l'àmbit de la venda a través d'internet. A la figura 1 se'n fa una representació parcial com a exemple.

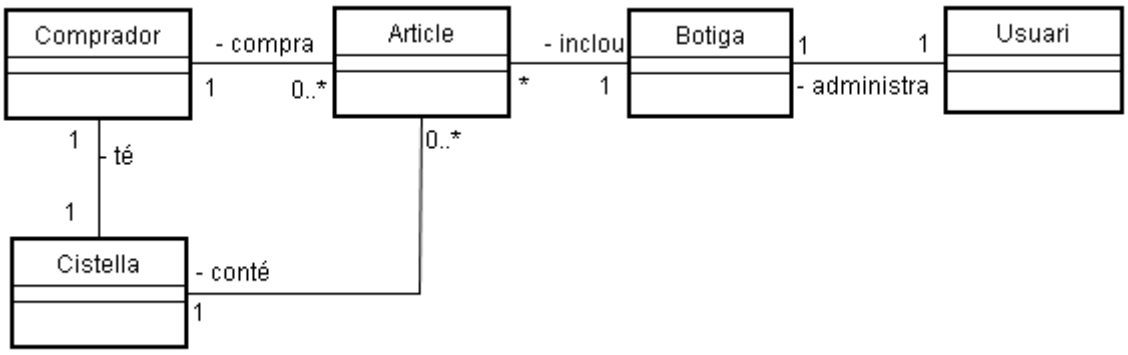


Figura 1. Representació dels objectes del domini

Com es pot veure (figura 1) hi haurà dos tipus d'usuari, per una banda el que visita el web i és susceptible de fer una compra i per una altra un usuari amb privilegis per a poder administrar la botiga.

Després d'haver detectat els objectes del domini, per a identificar les interaccions entre ells s'ha de tenir en compte abans de passar al disseny que poden no ser simples, ans al contrari, poden implicar altres accions que s'han representat mitjançant els diagrames de casos d'ús, en la figura 2 es mostra un exemple parcial d'un aquests diagrames en el procés inicial.

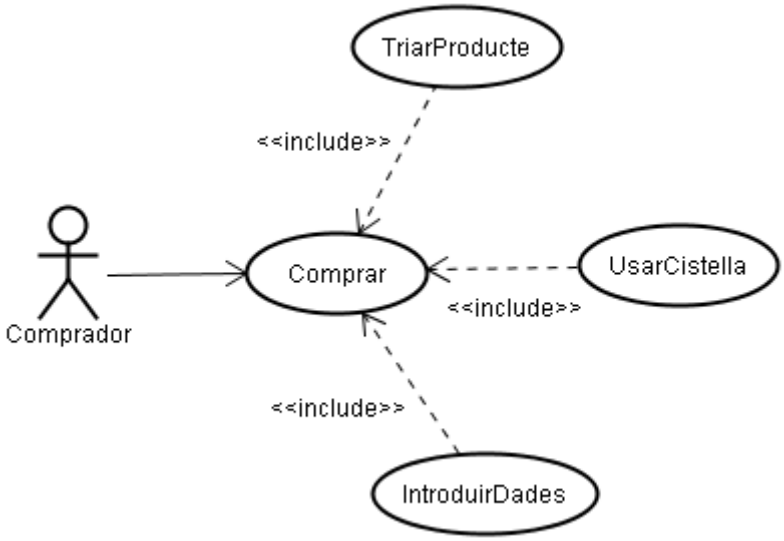


Figura 2. Vista parcial del diagrama del cas d'ús que representa l'acció principal i les altres implicades

A partir d'aquest anàlisi funcional s'ha passat al disseny tot cercant una solució conceptual que satisfaci els requeriments. Així doncs, en aquesta fase es descriuen els objectes de programari i l'esquema de la base de dades relacional necessària per aquest projecte.

A l'hora de fer el disseny la pretensió constant ha estat de separar la lògica de l'aplicació de les pàgines web –ASP.NET– per la qual cosa s'ha seguit una aproximació al patró 'Model-view-controller' adaptat a les característiques de l'eina utilitzada en el desenvolupament.

Fins aquest punt el mètode seguit és el comú d'enginyeria del programari per a qualsevol tipus de desenvolupament seriós. És en la següent etapa –la implementació– on entra en escena l'objecte principal d'aquest treball, és a dir, la tecnologia Microsoft Framework .NET.

Per a implementar el disseny obtingut, en primer lloc s'ha instal·lat al servidor el *Framework .NET* i per desenvolupar el programari hi havia diferents camins a seguir:

- Usar un editor de text pla i utilitzar el compilador proporcionat en la instal·lació del Framework en la línia de comandes.
- Utilitzar l'eina gratuïta *Web Matrix,* proporcionada per Microsoft que inclou un senzill editor visual.
- Fer servir l'entorn integrat de desenvolupament *Microsoft Visual Studio .NET.*

És evident que utilitzar la darrera opció facilita molt el treball i que des del punt de vista professional millora substancialment el rendiment, però per a aprofundir en el coneixement d'aquesta tecnologia s'ha considerat oportú seguir un mètode que podem anomenar combinat.

En primer lloc s'ha utilitzat la primera opció per fer les primeres proves, d'aquesta manera s'ha tingut una visió molt clara del funcionament intern de la tecnologia que ha estat d'una gran ajuda en el desenvolupament posterior.

En el cas del *Web Matrix,* s'ha volgut comprovar que sense fer cap despesa es pot desenvolupar amb aquesta tecnologia en un entorn visual.

Finalment, per la implementació, tot aprofitant els coneixements adquirits amb l'ús de les dues primeres opcions s'ha optat per a fer servir l'entorn de desenvolupament *Visual Studio .NET* .
Val a dir que per a resoldre problemes en la implementació han estat de gran ajuda les proves fetes amb les dues primeres opcions i que sense el seu ús no hagués estat possible conèixer en profunditat el funcionament d'aquesta arquitectura.

### 1.1.5. Planificació del projecte.

Per a dur a terme aquest projecte, com s'ha esmentat en l'apartat d'enfocament i mètode seguit, s'ha dividit en les següents fases pràctiques:

- **Definició.** Elaboració d'un document amb la descripció i planificació del projecte amb una durada d'una setmana. En aquesta fase es va considerar oportú també provar i triar les eines que es farien servir pel desenvolupament com són els entorns de programació i els gestors de bases de dades i tanmateix posar a punt altres eines necessàries com l'IIS o programari auxiliar per a la creació dels gràfics.

- **Anàlisi.** En aquesta fase d'una durada de dues setmanes es van fer les entrevistes amb els responsables de l'empresa a fi de

recollir els requeriments funcionals i detectar els objectes del domini. Totes aquestes tasques van donar lloc al document d'anàlisi funcional.

- **Disseny.** Un cop obtingut el document d'anàlisi es procedeix a l'elaboració del disseny amb una durada aproximada de dues setmanes. Com a tasques d'aquesta fase es poden citar la identificació dels objectes del negoci, la concreció del model de dades i el disseny de les classes conceptuals de programari a partir dels objectes del negoci detectats.
  En aquesta fase també s'ha creat un prototip de l'aplicació exclusivament amb elements *HTML* per a poder mostrar el disseny al client.

- **Implementació.** Després d'haver elaborat el disseny i amb l'aprovació del client s'ha procedit a la programació del web, codificació amb el llenguatge triat, creació de procediments emmagatzemats, taules, relacions etc.

- **Memòria.** Elaboració d'aquest document que descriu el progrés i resultat del projecte.

- **Presentació.** Elaboració d'una presentació de síntesi per a donar una visió general i clara del projecte.

A continuació es mostra el diagrama de Gantt de la planificació del projecte:

Figura 3. Diagrama de Gantt de la planificació del projecte (primera part)

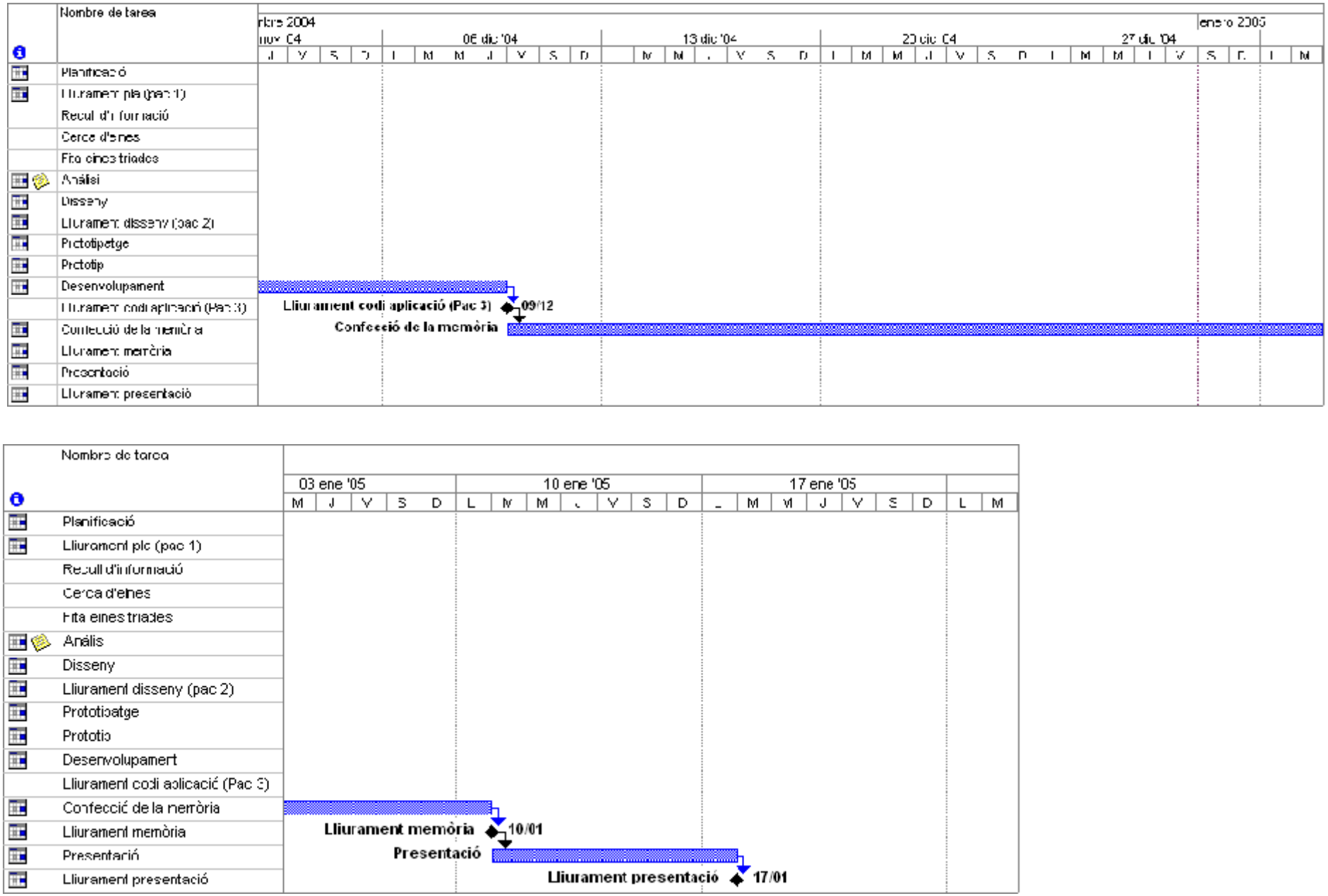


Figura 4. Diagrama de Gantt de la planificació del projecte (segona part )

### 1.1.6. Productes obtinguts.

Un cop finalitzat el projecte s'han obtingut els següents productes:

- Una aplicació web de comerç electrònic per a representar la tenda virtual dividida en dues parts:
    - La botiga virtual on un visitant pot comprar via Internet els articles exposats.
    - Una eina d'administració on els usuaris autoritzats per l'administrador poden modificar les dades.
- Una base de dades relacional que conté taules amb els articles per categories, dades dels clients, nom i paraula de pas dels usuaris i altres dades necessàries per l'aplicació.

### 1.1.7. Descripció dels capítols.

En els següents capítols es fa una descripció detallada de tot el procés d'implementació emfasitzant els següents punts:

- Infrastructura utilitzada.
- Solucions als problemes.
- Detall de les eines que s'han fet servir.
- Descripció i detall dels productes obtinguts.
- Elements de disseny
- El model de dades utilitzat
- La seguretat de la part privada
- Etc.

## 1.2. Infrastructura pel desenvolupament

### 1.2.1. Arquitectura ASP.NET - descripció

Degut a l'èxit comercial obtingut per l'aplicació *J2EE,* Microsoft es va veure relegat a un segon pla, això va motivar la companyia a crear una arquitectura integral per a permetre als desenvolupadors d'aplicacions oblidar-se del sistema operatiu, la gestió de la memòria etc.

Així doncs mitjançant diferents interfícies de programació suportades per biblioteques i plataformes d'execució comunes, es poden crear aplicacions i serveis web o locals.

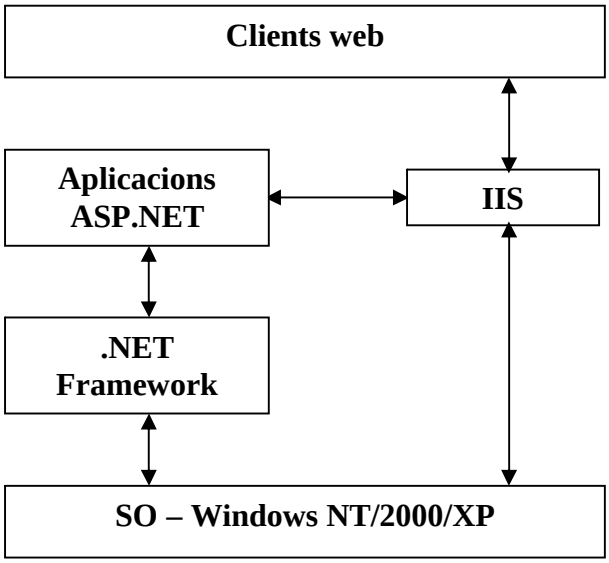


Figura 5. Funcionament de l'arquitectura ASP.NET

L'entorn necessari per a desenvolupar aplicacions ASP .NET és el Microsoft .NET Framework que permet tractar-lo –a diferència de les anteriors versions de ASP– com un llenguatge orientat a objectes.

Aquesta nova estructura fonamentalment està basada en els següents punts:

- Tot i que els llenguatges per a programar en ASP .NET són VB.NET, JScript i el nou Visual C#, es poden utilitzar diferents llenguatges de programació –n'hi ha més de 20– que permet a més desenvolupadors treballar en aquesta nova estructura.

- L'arquitectura ASP .NET està integrada en l'estructura .NET, no és una versió de ASP.

- Es poden crear aplicacions web ràpides, ampliables i flexibles i alhora fàcils d'entendre i de codificar.

- El codi es compila *'just in time'* mitjançant el motor CLR que optimitza i emmagatzema la compilació en la memòria cau.

- Tots els paràmetres de configuració es guarden en arxius en format XML, de lectura universal que es poden generar amb qualsevol editor de text.

- La seguretat basada en un conjunt d'esquemes d'autorització és de fàcil configuració i es pot adaptar a les necessitats de cada situació.

- .NET framework conté un gran nombre de biblioteques de classes que es poden utilitzar per a configurar transmissions TCP/IP a través de XML i amb els servidors web.

### 1.2.2. Arquitectura ASP.NET - funcionament.

Tot el procés que es duu a terme quan hi ha una petició de pàgina es produeix automàticament en segon pla, bàsicament l'únic que s'ha de fer és crear un arxiu de text amb el codi font i .NET Framework s'encarrega de transformar-lo en codi compilat. A continuació es fa una breu descripció d'aquest procés:

1. Es fa una petició de pàgina per a primer cop.

2. La pàgina es compila en una classe .NET

3. L'arxiu resultant es guarda en un directori especial en el servidor.

4. Si es sol·licita la mateixa pàgina i aquesta no ha canviat, s'executa l'arxiu corresponent guardat.

La pàgina ASP .NET no es compila en codi màquina directament, en realitat es compila en un llenguatge de nivell intermedi anomenat MSIL – *Microsoft intermediate language*– així mateix tots els llenguatges que es poden utilitzar sempre es compilen d'aquesta manera.

És quan el navegador sol·licita la pàgina quan es compila (figura 6) amb el *JIT* (justa a temps) de .NET Framework l'arxiu de classe guardat al directori especial.

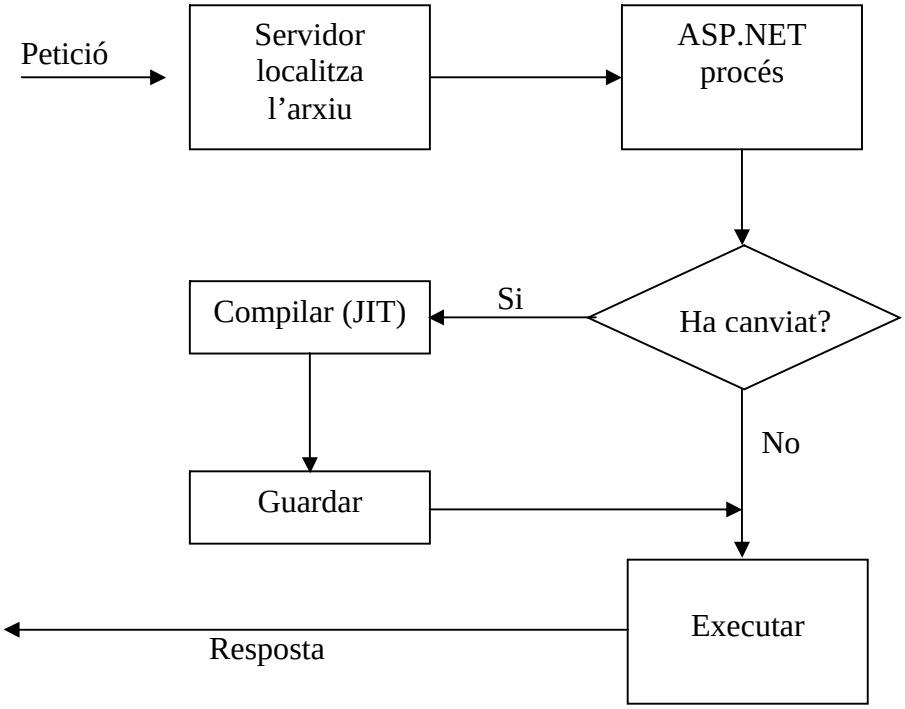


Figura 6. Petició d'una pàgina ASP .NET i resposta

Per a facilitar el desenvolupament Microsoft ha introduït el que s'anomena *Namespaces* –espai de noms– que no és més que un conjunt d'objectes del mateix tipus agrupats i relacionats entre si.

La utilitat principal d'aquest agrupament és la facilitat de cerca d'un determinat objecte. Tots els objectes deriven d'un conjunt principal anomenat *System*, així doncs, per exemple l'espai de noms *System.IO* conté tots els objectes necessaris per a llegir i escriure en fitxers.

Els contenidors d'objectes *Namespaces* s'emmagatzemen en biblioteques d'enllaç dinàmic (DLL) i el desenvolupador en pot crear de nous personalitzats segons les necessitats. Per a poder fer servir qualsevol espai de noms cal importar-lo amb una instrucció explícita:

<% Import Namespace="System.IO"%>

Aquesta instrucció indica al CLR que ha de fer referència a aquest espai de noms a l'hora de compilar l'aplicació ASP .NET per a poder localitzar els objectes del tipus IO.

### 1.2.3. Detall del mètode seguit en el projecte.

Com ja s'ha comentat en la introducció, per a dur a terme la part d'implementació d'aquest projecte i amb la finalitat de conèixer millor

l'arquitectura en primer lloc s'han programat les parts principals parts utilitzant un simple editor de text recolzat en alguns casos amb l'aplicació *Web Matrix* de lliure distribució proporcionada per Microsoft, tot aquest procés sense conèixer en absolut l'eina que posteriorment s'utilitzarà, l'entorn Microsoft Visual Studio .NET.

Després d'aquesta experiència es recomana a qualsevol desenvolupador que vulgui introduir-se en ASP .NET que abans de fer servir un entorn tant potent com VS utilitzi un editor i els compiladors de .NET framework, l'experiència és prou enriquidora.

Des de l'inici del projecte la idea ha estat sempre la de separar la presentació del contingut de les pàgines de la lògica de l'aplicació, d'aquesta manera en l'àmbit professional el procés de creació del lloc web es pot dividir en dues parts, el disseny i el desenvolupament, ja que realment són dos 'oficis' diferents.

En el marc de .NET framework es pot separar fàcilment el codi de la presentació mitjançant els següents mètodes:

- Empaquetant el codi de l'aplicació en components de negoci personalitzats tot posant la lògica de programació en un arxiu de classe independent. Aquesta seria la representació del *Model* en el patró.

- Aprofitar una de les característiques de les pàgines ASP .NET que permet separar el codi que representarà el *Controlador* del patró anomenada *codi de recolzament.*

D'aquesta manera aconseguirem tenir una aproximació al patró *Model-View-Controller* .
Per una banda tindrem la vista representada per les pàgines *aspx* i/o els controls d'usuari *ascx,* per una altra banda el *Controlador* representat pel codi de recolzament i finalment el *Model* on hi haurà tots els components del negoci.

Segons alguns autors aquest sistema no és una representació ortodoxa del patró *MVC*, però s'han consultat les pàgines oficials del MSDN i Microsoft en descriu una manera similar de la implementació en .NET *(http://msdn.microsoft.com/architecture/patterns).*

En el cas concret de la tenda virtual del projecte i per a fer una descripció que sigui prou entenedora s'ha articulat tot el procés d'implementació en el patró esmentat.

#### 1.2.3.1. Editor de text i compilació.

Com ja s'ha esmentat anteriorment, en primer lloc s'han fet les primeres proves d'implementació utilitzant un editor de text pla per a

crear els arxius necessaris: *pagina.aspx, objectesNegoci.vb, codiRecolzament.vb*. S'ha seguit el següent procés:

- Creació de la pàgina *aspx* que serà la que es mostrarà al client i representa la *Vista.( Figures 7 i 8 ).*

Figura 7. Exemple gràfic de la vista

- La part que representarà el *controlador* és el codi que donarà funcionalitat al botó anomenat 'añadir' que el que fa no és més que comprovar que el quadre de text no sigui buit i cridar un procediment del *Model* que afegirà la categoria d'articles a la base de dades. A la figura 9 es pot apreciar la primera línia de codi corresponent a *pagina.aspx* i que fa referència a aquest arxiu.

```
<%@ Page   Inherits="codiRecolzament" src="codiRecolzament.vb"%>

<HTML>
<HEAD>
<title>pagina.aspx</title>
</HEAD>
<body>
<form  Runat="Server">
<P>Categoria:<BR>
<asp:textbox id="txtNomCategoria" Runat="Server" /><BR>
Descripción:<br>
<asp:textbox id="txtDescripcióCategoria" Runat="Server" /></P>
<p><asp:button id="Button1" Text="añadir" Runat="Server"/></P>
</form>
</body>
</HTML>
```

Figura 8. Separació del codi, pàgina.aspx

```
<%@ Import Namespace="Components" %>
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
 if txtNomCategoria.Text <> "" Then
objecteNegoci.AfegirCategoria(txtNomCategoria.Text, txtDescripcioCategoria.Text)
 End If
End Sub
```

Figura 9. Separació del codi, codiRecolzament.vb

- El *Model* és representat per l'arxiu de la figura 10 on és pot veure un espai de noms amb una classe que conté el procediment *afegirCategoria* i que és cridat pel controlador per a accedir a la base de dades. Perquè el compilador sàpiga on es troba cal importar el *Namespace* amb la directiva que hi ha a la primera línia (figura 9).

```
Imports System
Namespace Components
Public Class objecteNegoci
Public Shared Sub AfegirCategoria(ByVal nom As String,
ByVal descripcio As String) As String
Dim connDades As SqlConnection
Dim cmdInsert As SqlCommand
Dim strInsert As String
connDades = Connecta()
strInsert = "INSERT Categories (NomCategoria, Descripcio)
VALUES (" + "'" + nom + "'" + ", " + "'" + descripcio + "'" + ")"
cmdInsert = New SqlCommand(strInsert, connDades)
cmdInsert.ExecuteNonQuery()
Desconnecta(connDades)
End Sub
End Class
End Namespace
```

Figura 10. Separació del codi, objecteNegoci.vb

Un cop s'han creat els tres arxius perquè l'aplicació funcioni s'han de seguir els passos següents:

- En el cas del component de negoci:
    - Compilar l'arxiu de classe amb el compilador proporcionat per la plataforma.
    - Copiar l'arxiu de classe compilat al directori /BIN de l'aplicació.
- En el cas del codi de recolzament no és necessari compilar-lo ja que es compila automàticament quan hi ha una petició de pàgina, de totes maneres si es vol es pot compilar.

Per a dur a terme la compilació només cal obrir una sessió DOS i des de la línia de comandes executar el compilador.

*Compilador de Visual Basic .NET:*

*vbc /t:library objecteNegoci.vb*

*vbc /t:library /r:system.dll, system.web.dll codiRecolzament.vb*

Figura 11. Sentències per a compilar

### 1.2.3.2. Ús de l'entorn Visual Studio .NET.

L'objectiu d'haver utilitzat un mètode rudimentari per les proves d'implementació ha estat el de conèixer a fons el funcionament de l'arquitectura però òbviament per a dur a terme projectes d'una certa envergadura, per optimitzar el projecte cal emprar alguna eina visual que faciliti la feina.

L'eina triada ha estat l'entorn Microsoft Visual Studio .NET que permet fer el disseny d'una manera visual arrossegant els controls necessaris i separa el codi i el compila d'una manera transparent per l'usuari.

Visual Studio .NET és un entorn integrat que serveix tant pel disseny com per la programació que permet utilitzar diferents llenguatges de programació que a més a més poden conviure en una mateixa aplicació.

En referència a l'apartat anterior el disseny i la implementació es facilita molt i es redueix a:

- Disseny visual de les pàgines ASPX mitjançant controls que ja estan implementats i que tenen una sèrie de propietats adaptables a gaire bé qualsevol projecte.
- Per a escriure el codi de recolzament només cal fer 'doble click' al control i s'accedeix al codi que està automàticament separat en un altre arxiu.
- El *Model* s'escriu en un altre arxiu però no cal compilar-lo manualment, l'entorn ja s'encarrega de tota la feina.

### 1.2.4. El patró per a la implementació.

A partir d'aquest punt se seguirà el patró MVC per a descriure amb detall tot el procés d'implementació.

En primer lloc es descriurà la *Vista* que correspon al treball de disseny de l'interfície que veurà l'usuari.

En segon lloc la programació del *Controlador,* és a dir, tot el codi que controla els esdeveniments quan l'usuari fa alguna acció sobre la pàgina.

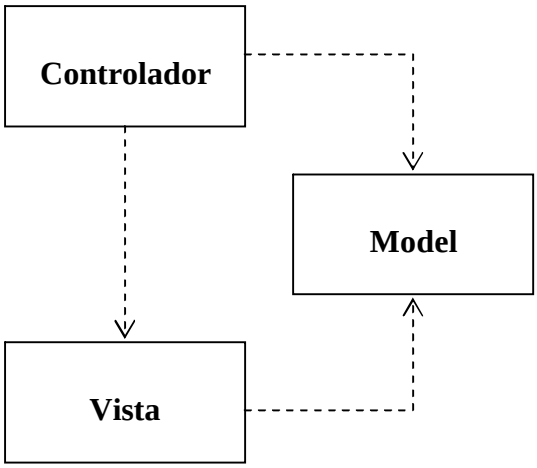

Figura 12. Esquema del patró MVC

Finalment es descriurà el que es pot anomenar el cor de l'aplicació, el *Model,* on s'emmagatzema tota la lògica de negoci, mètodes per accedir a les dades i altres procediments pel bon funcionament de l'aplicació.

#### 1.2.4.1. La Vista, disseny de l'interfície.

La vista la integren un grup de pàgines aspx i/o controls d'usuari ascx, ja que alguns objectes com la cistella i el menú es van repetint.

La intenció ha estat que tant la cistella actualitzada com el menú siguin sempre visibles excepte a la pàgina d'entrada, per tant s'ha optat per la utilització de controls d'usuari per aquest fi.

Perquè la presentació de les pàgines sigui més estàndard i funcioni amb més navegadors també s'ha desestimat la utilització de marcs, el que s'ha fet és utilitzar dos controls d'usuari normals pel menú i per la cistella i un altre control que es carregarà dinàmicament per a mostrar la informació dels articles.

En aquest punt cal comentar una mica què és i com funciona un control d'usuari:

El que Microsoft anomena control d'usuari no és més que una pàgina ASP .NET convertida en control i per tant tornar a utilitzar, la qual cosa facilita la programació quan s'ha de repetir el contingut o la lògica de l'aplicació.

Per a crear un control d'usuari només cal procedir com per a crear una pàgina web però sense posar els *TAG* html, per incloure'l primer s'ha de registrar posant al principi de la pàgina la directiva:

*<%@ Register TagPrefix="uc1" TagName="Menu"*
*Src="ControlsUsuari/Menu.ascx" %>*

Després com qualsevol control estàndard:

*<uc1:menu id="Menu1" runat="server"></uc1:menu>*

Així doncs es pot veure que és molt senzill, i encara ho és més si es fa servir l'entorn integrat Visual Studio, només cal arrossegar el control com si es tractés de qualsevol altre i el mateix entorn s'encarrega de registrar i escriure el codi necessari.

En la figura 13 es pot veure com queden ubicats els controls d'usuari per la cistella i el menú.


Figura 13. Ubicació dels controls d'usuari

Un cop s'ha explicat el concepte de control d'usuari es detalla l'estructura de l'aplicació i les solucions per a crear-la.

En primer lloc es necessita un sistema de navegació per a accedir a les diferents categories, per a aquest fi s'utilitza el control d'usuari *Menu.* S'ha creat el control amb un *DataList* que mitjançant les propietats de disseny imita un sistema de fitxes que representa les categories d'articles i que es pot modificar a través de l'eina d'administració. Figures 14 i 15.
Tot aprofitant les característiques dels controls d'usuari, aquest control s'aprofita en cascuna de les pàgines on faci falta.

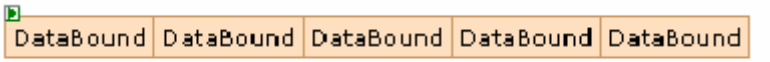

Figura 14. Control Menu en temps de disseny

Figura 15. Control Menu en temps d'execució

Cada cop que l'usuari selecciona una categoria en el *Menu* l'aplicació crida la pàgina Categoria.aspx i li envia una variable amb el nombre de la categoria triada:

*http://ruta_servidor/Categoria.aspx?cat=4*

Es detallarà el procés en els apartats corresponents al *Model* i *Controlador.*

Per a guardar els articles que l'usuari potencial comprador triï s'ha creat el concepte de cistella i s'ha dividit en dues parts, per una banda la funcionalitat –detallada en l'apartat del *Model*– i per una altra la part de presentació.

Per a la presentació s'utilitza el control d'usuari *Carret.ascx* ja que s'ha cregut convenient que sempre sigui visible, tot i que aquest tema és una opinió molt personal. Figures 16 i 17.

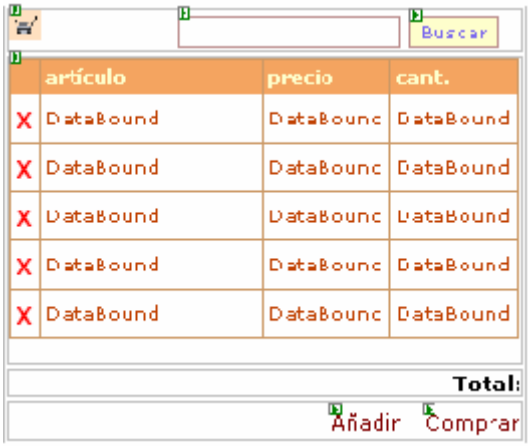

Figura 16. Control Carret en temps de disseny

S'ha aprofitat aquest control per a incloure un sistema de cerca d'articles ja que també cal que es vegi a las mateixes pàgines que la cistella.

Per a construir aquest control s'ha fet servir un control estàndard del Visual Studio anomenat *DataGrid* tot aprofitant també les diferents possibilitats de disseny que ofereix.

Figura 17. Control Carret en temps d'execució

Quant a la presentació dels articles el procediment ha estat diferent, s'utilitza també un control d'usuari que en realitat es comportarà com una plantilla que hereta d'una classe anomenada *PlantillaProducte* del codi del *Model*.

El motiu de crear una plantilla per a representar els articles és perquè és molt probable que els articles tot i pertànyer al mateix sector hi pot haver varietat i caldrà representar-los de diferent manera, amb una plantilla es pot fàcilment aprofitar la classe del *Model.*

Per raons de temps s'ha creat únicament la plantilla per defecte –*default.ascx*–, però per a ampliar el nombre de presentacions només caldrà crear diferents plantilles que seran cridades segons el tipus d'article, per a fer això possible es guarda a la base de dades el nom de la plantilla en un camp de la taula d'articles i amb una simple sentència SQL associar-la al producte desitjat:

*Update Productes Set Plantilla='Accessoris' Where CategoriaID=4*

Per a carregar dinàmicament la plantilla s'utilitza el mètode *LoadControl* i es col·loca en un control *PlaceHolder* que ha estat dissenyat específicament com a contenidor d'altres controls. Figures 18 i 19.

Figura 18. Control PlaceHolder en temps de disseny


Figura 19. Plantilla del producte carregada al control PlaceHolder

Fins ara s'ha parlat de la part presentació de la botiga, a continuació es detalla la presentació de l'administració de la web.

La presentació de l'administració ha de ser privada, només es permet l'accés a usuaris autoritzats, en l'apartat de seguretat es detallarà el sistema emprat per aquest fi.

Per l'administració del web s'ha optat per a presentar una pantalla que tingui incloses totes les funcions a la que s'hi pot accedir –prèvia autenticació– des de qualsevol lloc del web, s'han de poder dur a terme les següents accions:

- Afegir, eliminar o modificar les diferents categories d'articles.
- Afegir, eliminar o modificar Els diferents articles.
- Opció per a adjuntar una imatge descriptiva de l'article.
- Opció per a mostrar llistes tant de les comandes com dels clients.

A la figura següent es pot veure la pantalla que dóna accés a totes les funcions esmentades:


Figura 20. Pantalla d'opcions d'administració

Per a fer les modificacions tant dels articles com de les categories l'usuari pot seleccionar la opció adient i se li mostra una pantalla amb una graella editable:

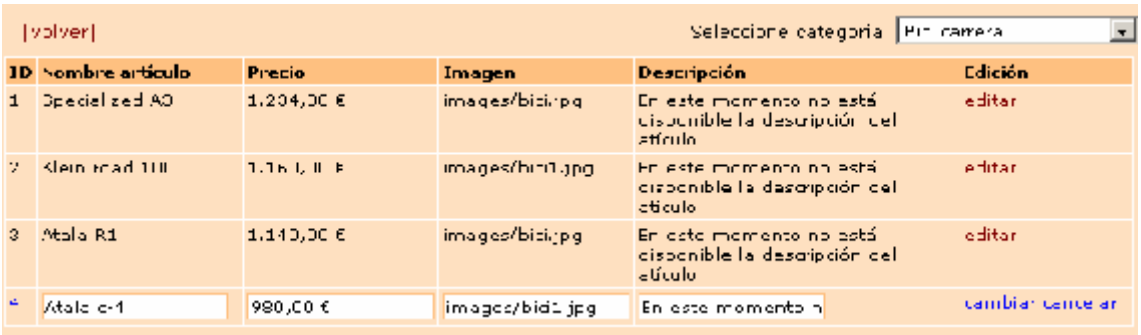


Figura 21. Graella editable amb els articles per categoria

La opció de llista de comandes mostra una pantalla amb les comandes on es pot veure si s'han servit o no i un enllaç a la factura corresponent. Figures 22 i 23.

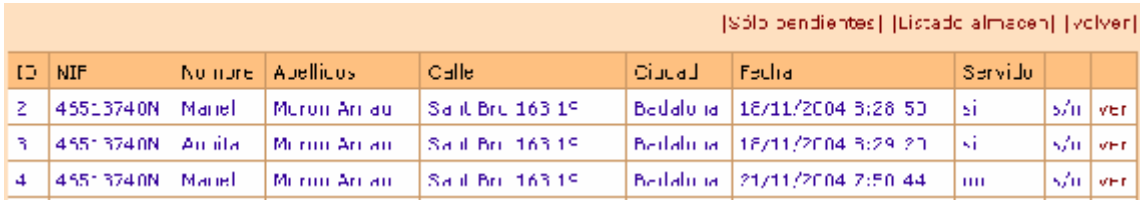

Figura 22. Llistat de comandes

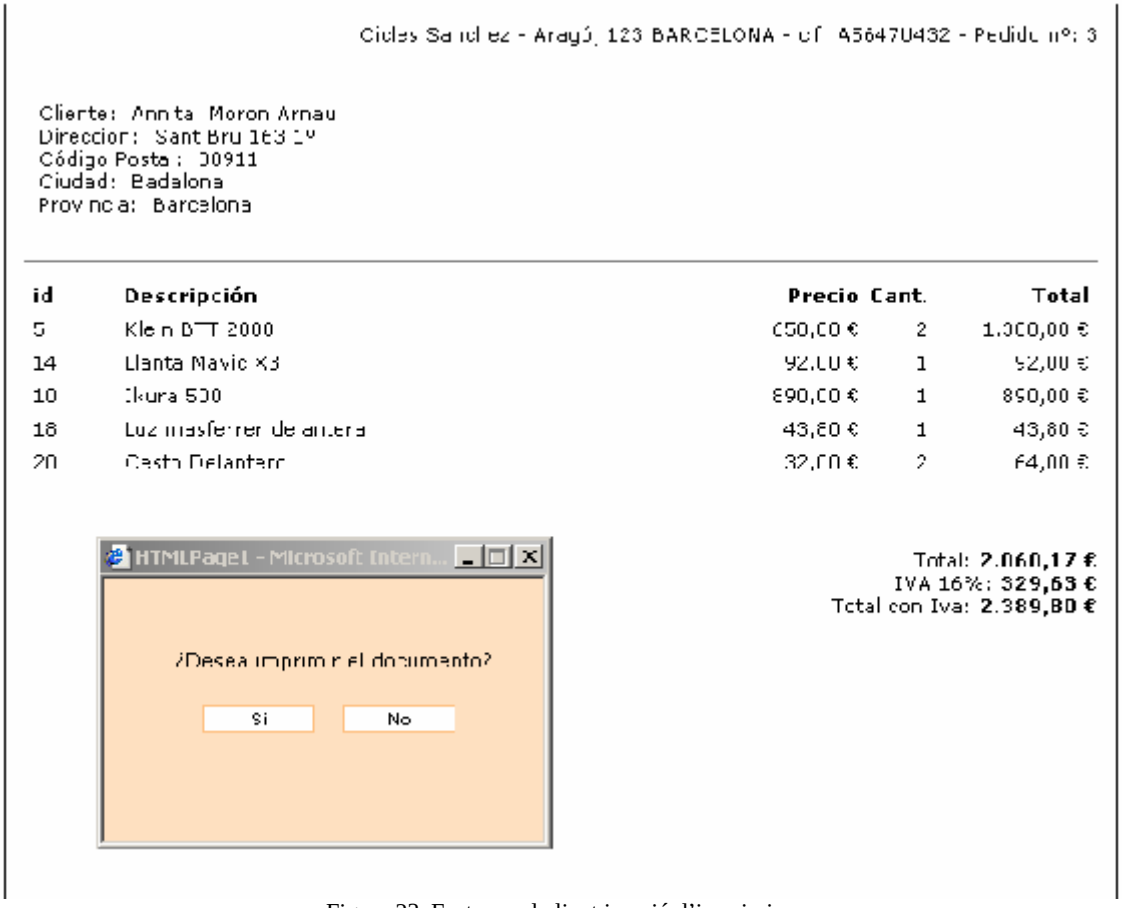

Figura 23. Factura pel client i opció d'imprimir

L'altra opció de llista mostra una llista dels clients que s'emmagatzemen quan compren a la botiga, s'ha inclòs una opció per a enviar correu personal o a tots els clients. Figures 24 i 25.

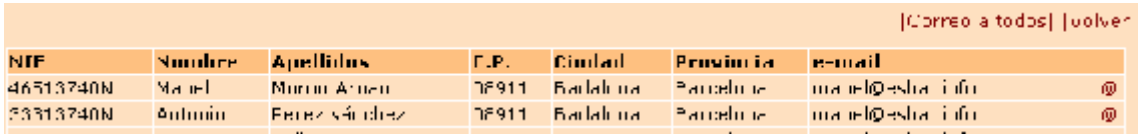

Figura 24. Llistat de clients


Figura 25. Pantalla per enviar correu

Per a afegir usuaris autoritzats l'administrador pot modificar la taula d'usuaris des del sistema gestor donant d'alta, eliminant o bé modificant els camps usuari o password, no s'ha cregut convenient que s'hi pugui accedir a través de l'aplicació.

#### 1.2.4.2. Model.

El model està representat en l'aplicació per dos espais de noms diferenciats per les dues parts de que es composa, la tenda i l'administració, *ComponentsTenda* i *Admin*. Figura 26.

El codi font d'aquests dos espais de noms resideix en un únic fitxer anomenat *ComponentsTenda.vb* que Visual Studio compila d'una manera transparent per l'usuari en una biblioteca d'enllaç dinàmic anomenada per defecte *tenda.dll*. Aquesta DLL es copia automàticament al directori BIN perquè pugui ser executada.

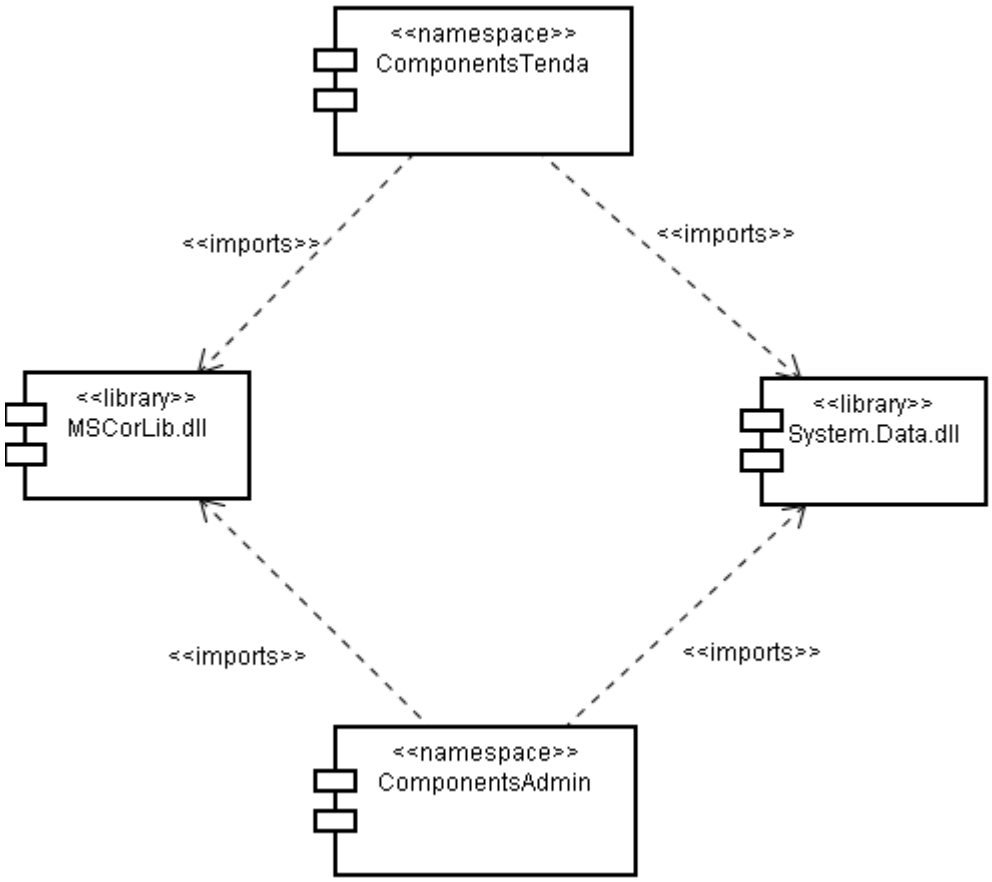


Figura 26. Esquema del Model.

**ComponentsTenda.**

L'espai de noms *ComponentsTenda* conté una sèrie de classes components del negoci que donen funcionalitat a l'aplicació, a continuació es descriuen els seus mètodes i la funcionalitat.
Aquest *Namespace* està integrat per quatre classes que es detallen a continuació:

**ComponentsTenda.DadesCache**

Aquesta classe encapsula les funcionalitats per a recuperar dades del context si hi són i si no les extreu de la base de dades, per aquest fi utilitza diferents mètodes públics compartits recolzats per altres de privats.

A més a més d'utilitzar las classes adients per a l'accés a la base de dades també fa servir la classe *httpContext* per a recuperar de la memòria temporal la informació.

Es mostra una vista parcial del diagrama de classes on es pot veure els mètodes que conté la classe. Figura 27.

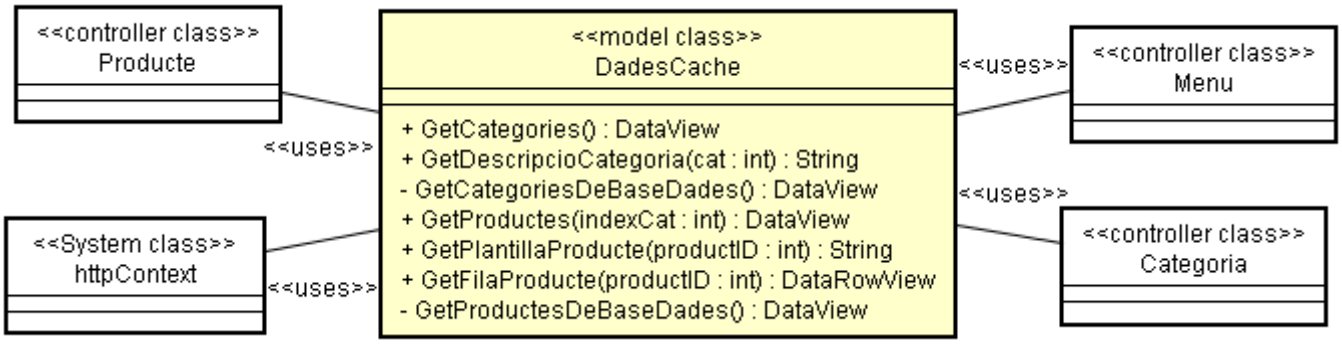


Figura 27. Diagrama de classes DadesCache

S'ha utilitzat aquest sistema aprofitant les característiques de l'arquitectura per raons de rendiment, així doncs quan se selecciona del menú una categoria o un article s'utilitza una còpia emmagatzemada a la memòria temporal.

Per exemple si el mètode *GetCategories* no pot recuperar les categories de la memòria temporal crida el mètode privat *GetCategoriesDeBasedades* que retorna la informació amb format en un *DataView.*

**ComponentsTenda.Cistella.**

Aquesta classe correspon a la segona part del carro de compra i té tota la funcionalitat de base de dades.

La cistella fa un seguiment dels usuaris a través d'una *cookie* que crea automàticament i llegeix en el moment d'instanciar-se.

Per a crear la *cookie s*'utilitza un GUID en el format per defecte de cadena predeterminada.

La cistella s'implementa com un *DataSet* i s'utilitza un *DataAdapter* per a crear-lo i actualitzar la taula de la base de dades quan es realitzen canvis. Figura 28

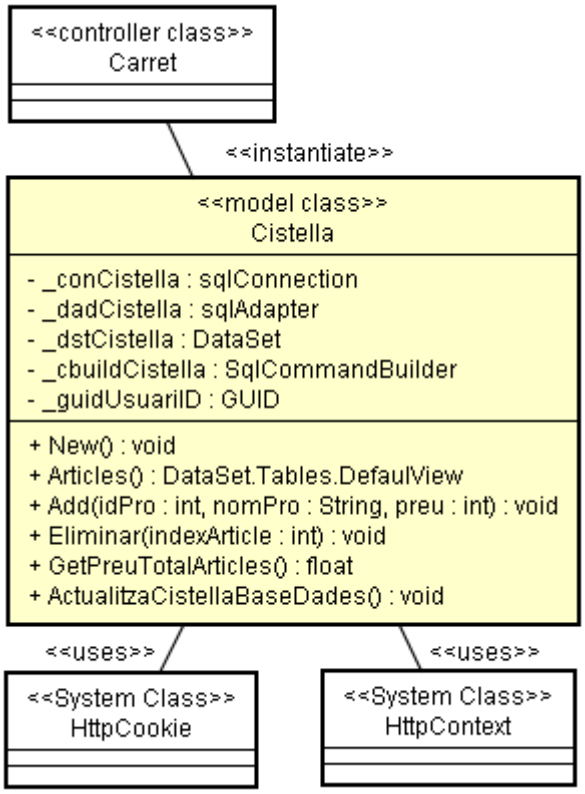

Figura 28. Diagrama de classes Cistella

**ComponentsTenda.PlantillaProducte.**

Per a aconseguir que els controls d'usuari que representen les plantilles dels articles –en principi només s'ha implementat *default.ascx*– quan es carreguin dinàmicament funcionin, han de tenir accés a la taula de productes.

Per aquest motiu s'ha fet l'herència. La classe producte representada per *producte.aspx* carrega dinàmicament el control *default.ascx* que hereta de la classe *PlantillaProducte* per a

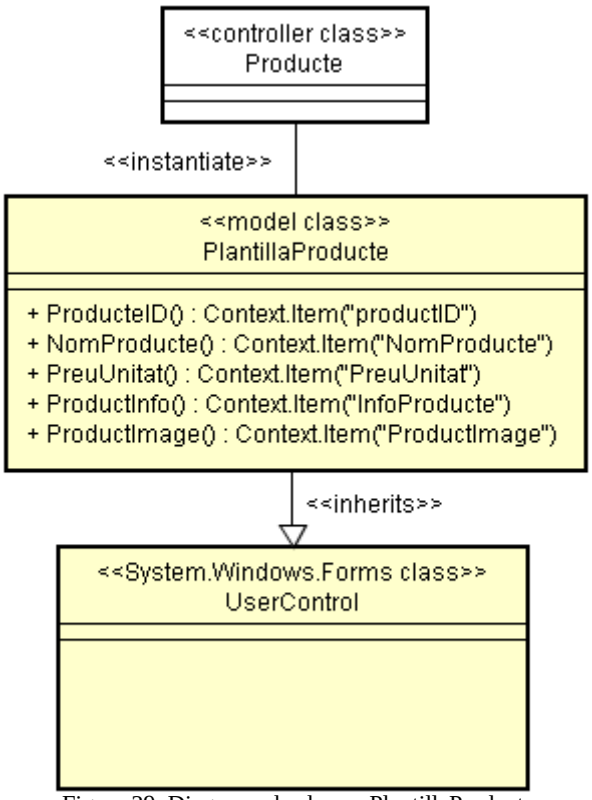

Figura 29. Diagrama de classes PlantillaProducte

poder tenir accés als seus mètodes i aquesta classe hereta de la classe del sistema *UserControl.* Figura 29.

**ComponentsTenda.Cerca.**

Aquesta classe s'ha creat exclusivament per a donar funcionalitat a l'opció de cerca que per motius de disseny s'ha inclòs dins del control d'usuari *carret.ascx.* (Veure pag. 23, figura 16).

Bàsicament conté un mètode que rep com a paràmetre un criteri de cerca i retorna un *dataSet* amb els articles trobats segons el criteri.

**Admin.**

L'espai de noms *Admin* conté dues classes, *AdminCategories* i *Autenticacio,* a continuació es detallen les classes:

**Admin.AdminCategories.**

En aquesta classe s'encapsulen totes les funcionalitats de *Privat.aspx,* per modificar la base de dades. Figura 30.

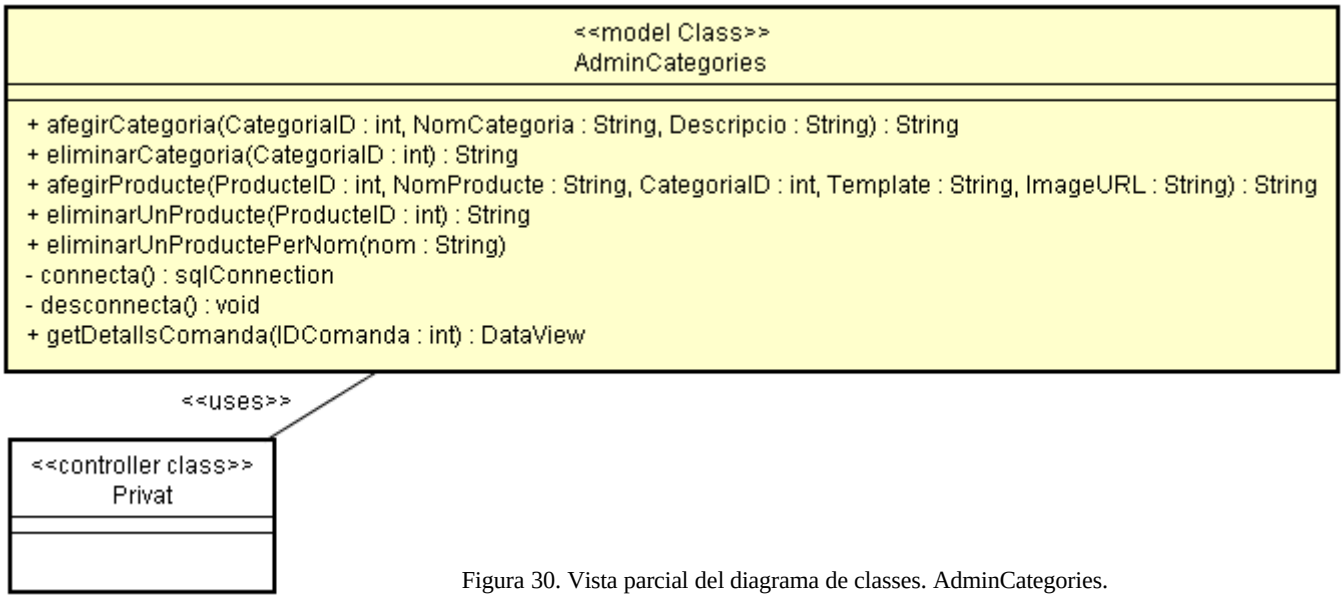


Figura 30. Vista parcial del diagrama de classes. AdminCategories.

**Admin. Autenticació.**

Per raons de claredat s'ha creat aquesta classe amb un únic mètode anomenat *LoginBD* que òbviament ofereix la funcionalitat per a controlar l'accés la part privada del web.

Com s'ha utilitzat com a gestor de base de dades –veure l'apartat del model de dades– SQLServer, per a aprofundir en el coneixement de les característiques d'aquest programari el mètode *LoginDB* utilitza *Stored Procedures,* és a dir mètodes compilats en l'SQLServer i emmagatzemats, la qual cosa optimitza el funcionament dels accessos a la base de dades.

Així doncs, l'únic que fa aquest mètode és cridar un *StoredProcedure* que té el mateix nom. Es mostra una fracció del codi:

```
 cmdSelect = New SqlCommand("LoginBD", conDades)
 cmdSelect.CommandType = CommandType.StoredProcedure
parmReturnValue = cmdSelect.Parameters.Add("RETURN_VALUE",
SqlDbType.Int)
 parmReturnValue.Direction = ParameterDirection.ReturnValue
cmdSelect.Parameters.Add("@username", strUsuari)
cmdSelect.Parameters.Add("@password", strPassword)
cmdSelect.ExecuteNonQuery()
intResult = cmdSelect.Parameters("RETURN_VALUE").Value
```

Figura 31. Detall del codi per accedir a procediments emmagatzemats a la BD

En l'apartat del model de dades es detallaran els procediments emmagatzemats d'SQLServer.

### 1.2.4.3. Controlador.

El controlador està format per totes les classes de les quals hereten tots els *webForms,* que alhora contenen els esdeveniments necessaris per a capturar les entrades que fa l'usuari i d'altres mètodes, d'aquesta manera tot el codi executable queda separat de la vista on resideixen las pàgines 'aspx' i els controls d'usuari 'ascx' amb els *webControls.* Totes aquestes classes són compilades d'una manera transparent per l'usuari per l'entorn Visual Studio .NET en la DLL anomenada *tenda.dll*.

Utilitzant pel desenvolupament Visual Studio .NET tot el codi de control s'escriu automàticament en arxius diferents de les pàgines *aspx* o *ascx,* així doncs no cal preocupar-se per la separació del codi, l'entorn ho fa per nosaltres.

El codi font de les classes de control es troba en arxius amb el mateix nom que les de la vista afegint-hi l'extensió VB ex. *Categoria.aspx.vb*

Com exemple de funcionament a la figura 32 es mostra en un diagrama de seqüències el funcionament de l'acció d'afegir una categoria.

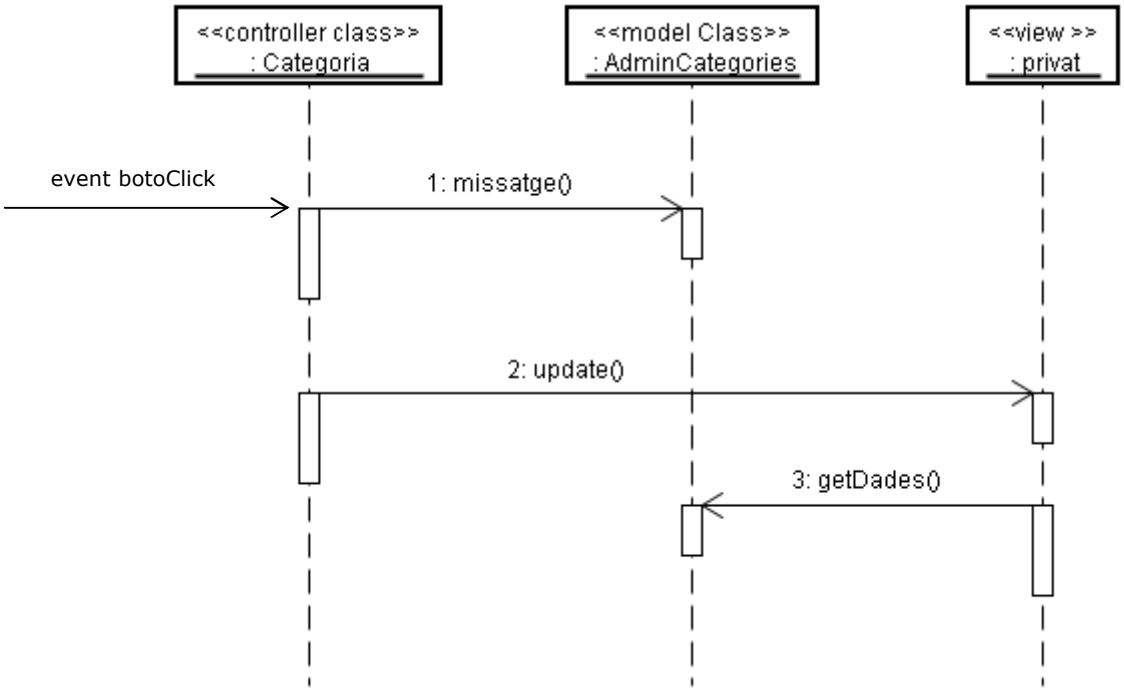


Figura 32. Diagrama de sequències. AfegirCategoria.

El *controlador* modifica el *Model* i informa a la *Vista* dels canvis i és el *Model* exclusivament el que interacciona amb la base de dades.

### 1.2.5. La seguretat i la part privada.

La botiga virtual naturalment és d'accés públic però a la part d'administració no hi ha de poder accedir tothom, per tant ha de ser un lloc restringit i cal tractar-lo de manera diferent.

L’arquitectura ASP.NET permet diferents mètodes d’autenticació però per aquest projecte s’ha utilitzat l’autenticació basada en formularis que utilitza cookies, a continuació es detallen els passos per a dur a terme la configuració de l’autenticació.

Per entendre bé el procés cal parlar primer de l’arxiu de configuració anomenat *web.config*. A l’arrel de l’aplicació hi ha un arxiu d’aquest tipus que serveix per a la configuració general, però d’una manera jeràrquica n’hi pot haver més en altres sots-directoris per a personalitzar la configuració

En primer lloc s’ha activat l’autenticació amb els següents passos:

- S’ha configurat el mode d’autenticació modificant la secció adient de l’arxiu arrel *web.config* de l’aplicació.

```
<authentication mode="Forms">
<forms
        name=".tendaCookie"
        loginUrl="Login.aspx"
        protection="All"
        timeout="10"
        path="/"/>
</authentication>
```

- S’ha denegat l’accés als usuaris anònims al directori *private* creat per a aquest fi creant dins d’aquest directori un nou arxiu *web.config.*

```
<configuration>
        <system.web>
                <authorization>
                        <deny users="?" />
                </authorization>
        </system.web>
</configuration>
```

- S’ha creat una pàgina de registre amb un formulari que permeti als usuaris introduir el nom i la contrasenya.

D’aquesta manera un usuari que vulgui accedir a qualsevol pàgina que es trobi dins del directori *privat* rebrà com a resposta la presentació de la pàgina *login.aspx*. *Figura 33.*

Amb aquest sistema fins i tot si s'intenta accedir directament a la URL d'una pàgina del directori privat sempre s'obté per resposta la pàgina de *login*.


Figura 33. Pantalla de registre d'usuaris.

### 1.2.6. El model de dades.

Per a crear la base de dades s'ha utilitzat el servidor de BD SQLServer, s'ha optat per a crear una base de dades senzilla amb les taules necessàries pel funcionament de l'aplicació.

Aprofitant que aquest gestor de bases de dades permet emmagatzemar procediments s'han creat alguns mètodes per a accedir a les dades que empra el model de l'aplicació.

Es mostra el diagrama entitat-relació del model de dades obtingut en l'etapa de disseny. Figura 34.

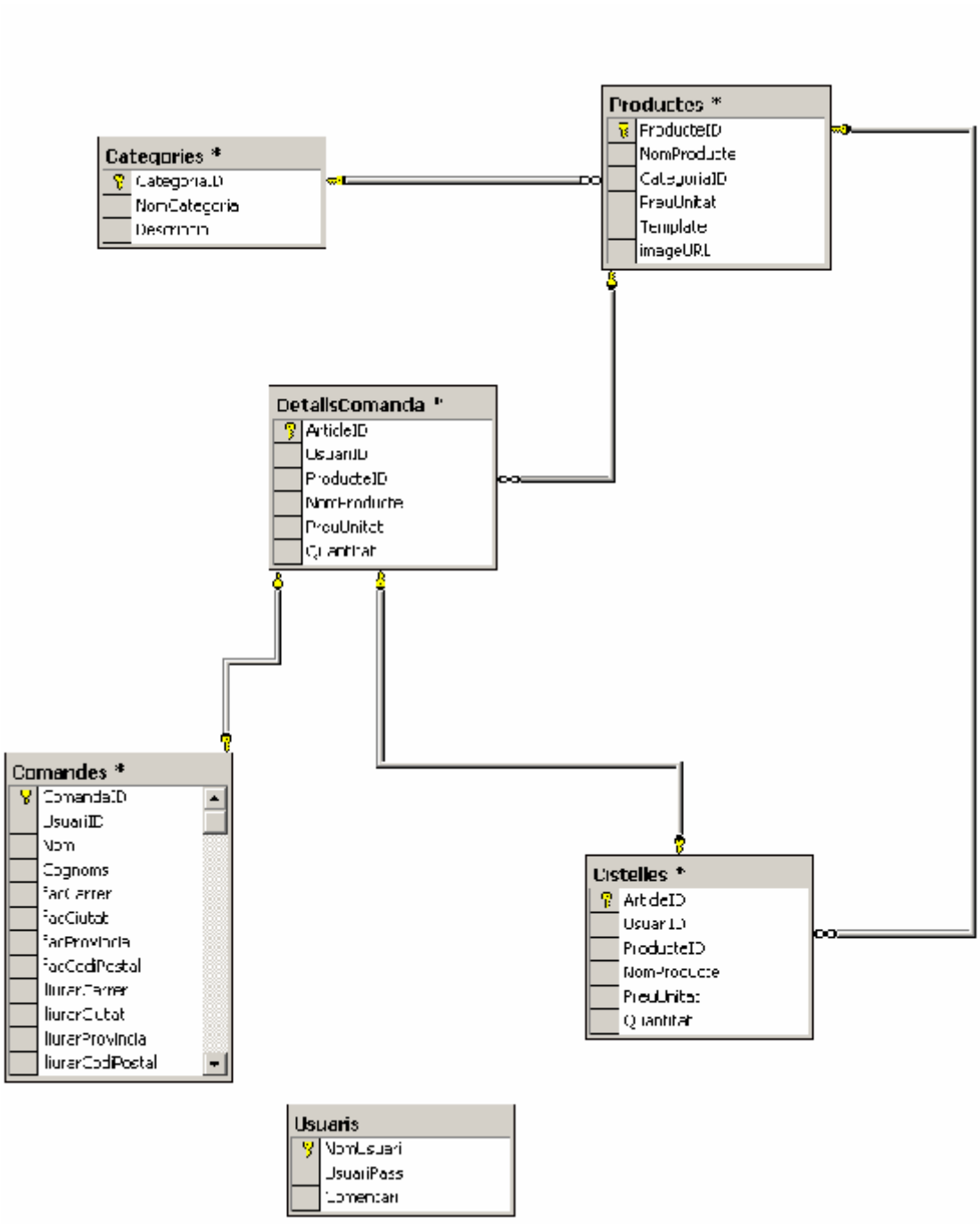


Figura 34. Diagrama ER

En l'etapa d'implementació s'ha cregut convenient guardar els usuaris que compren com a clients i s'ha creat una nova taula de clients.

Les taules Categories i Productes guarden les dades dels articles, la taula *DetallsComanda* conté les dades que mitjançant el procediment emmagatzemat *FerComanda* s'han passat de la taula *Cistelles.*

La taula *Comandes* conté les dades del comprador i està relacionada amb la taula *DetallsComanda.*
La taula *Usuaris* conté els noms i les paraules de pas dels usuaris autoritzats a modificar la base de dades.

**Procediments emmagatzemats.**

Per a augmentar el rendiment de l'aplicació s'ha cregut oportú crear els següents procediments emmagatzemats que crida el *Model*:

- *afegirProducte.* Rep com a paràmetres les dades de l'article i l'afegeix a la taula productes.

- *ClientNIF.* Rep com a paràmetre el NIF d'un client i retorna totes les dades del client.

- *e*xisteix*Client*. Rep com a paràmetre el NIF d'un client i retorna 0 o bé 1 segons si existeix o no el client.

- *f*er*Comanda.* Rep les dades de la comanda, les afegeix a la taula comandes, agafa les dades de la taula cistella i les afegeix a la taula *DetallsComanda,* finalment elimina la cistella.

- *GuardaClient.* Pren les dades del formulari de compra i las guarda a la taula *Clients.*

- *LoginBD.* Rep com a paràmetres el nom i la paraula de pas de l'usuari i retorna els valors per a comprovar que l'usuari existeix.

## 1.3. L'aplicació i l'usuari. Disseny gràfic.

Encara que l'objectiu del projecte sigui el coneixement de la plataforma *Framework.NET* s'ha considerat oportú fer una breu descripció de l'entorn que es trobarà l'usuari i del disseny quan accedeixi al web.
Amb l'entorn integrat Microsoft Visual Studio .NET està clar que s'ha volgut integrar la part del disseny i que sense sortir d'aquest programari és possible abastar les dues parts d'una aplicació d'aquest tipus.

Per una banda l'entorn facilita molt la implementació de la part activa de la pàgina, però per una altra banda gaire bé sense fer servir cap programari extern es pot aconseguir un disseny que sigui atractiu i alhora funcional.

En el disseny d'aquesta aplicació l'objectiu ha estat entre d'altres que no hi hagi elements que distreguin l'atenció de l'usuari i que l'interfície sigui prou intuïtiva.

S'ha tingut en compte per exemple que les resolucions de les pantalles no són sempre les mateixes, per això la mida horitzontal sempre es manté per

sota dels 700 px. i la mida vertical per sota dels 600 px. excepte quan la informació surt del camp de visió que s'han d'usar els *scrollbars.*

Així mateix s'ha fet ús dels fulls d'estil per a donar uniformitat al disseny, val dir que l'entorn facilita també aquesta feina de disseny, per exemple amb un 'arrossegar i deixar anar' es pot crear una referència a l'arxiu .cs.

Es mostren a continuació vàries captures de les principals pàgines amb explicacions:


Figura 35. Imatge de la pantalla principal

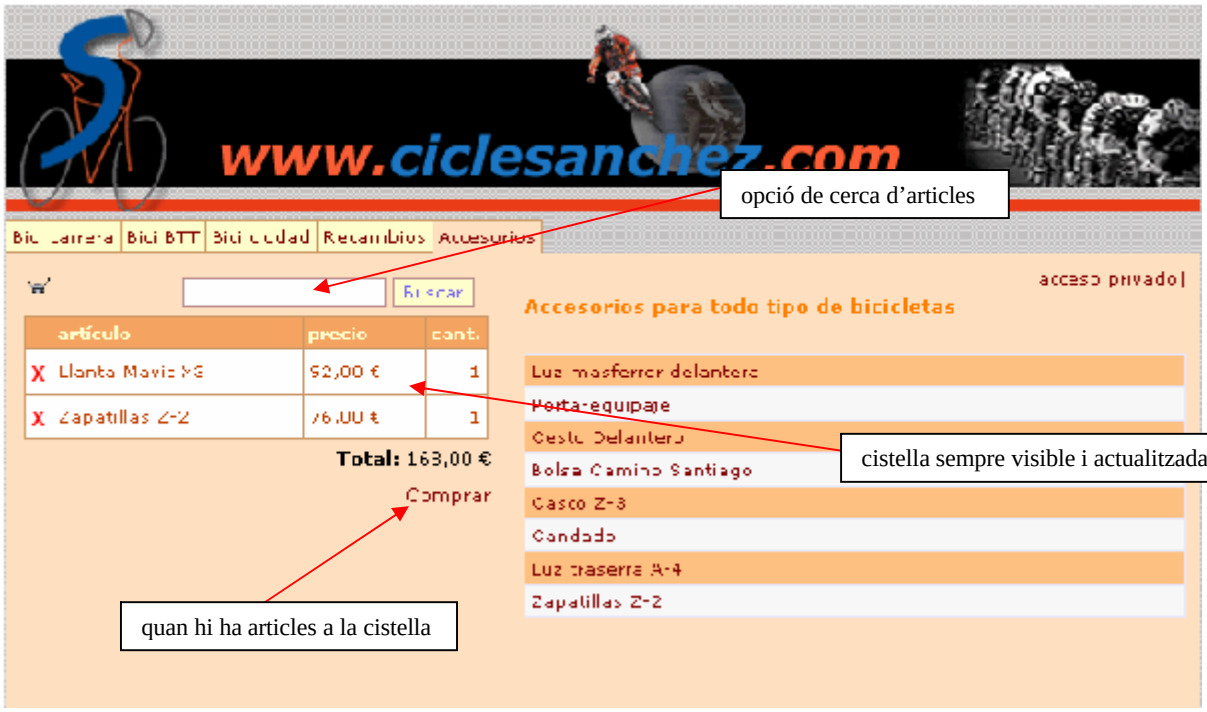


Figura 36. Imatge de la pantalla amb una categoria seleccionada


Figura 37. Pantalla de descripció de l'article

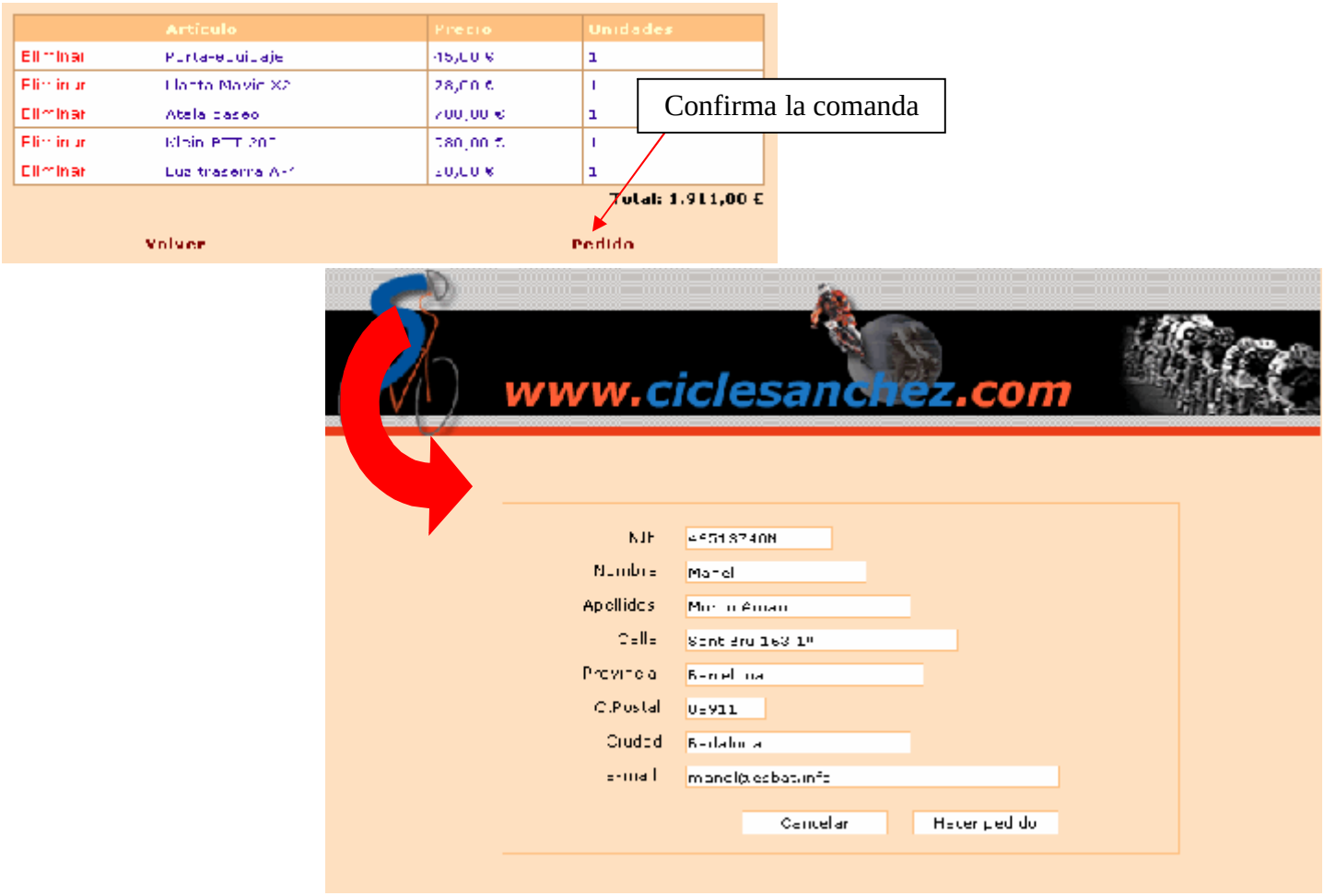


Figura 38. Pantalles per a fer les comandes

La part d'administració segueix el mateix model emprat per la part de la botiga virtual, en el capítol de la *Vista* ja s'ha tractat aquest tema, veure figura 20. Pantalla d'opcions d'administració.

### 1.4. Relació de les eines utilitzades.

A continuació es fa una relació del programari que s'ha utilitzat per aquest projecte:

- **UltraEdit 32.** Editor de text pla i hexadecimal amb el que l'autor està familiaritzat.

- **WebMatrix.** Senzill entorn integrat de MIcrosoft de lliure distribució.

- **Framework.NET.** Plataforma o marc necessari perquè funcionin les aplicacions, és de lliure distribució.

- **IIS (Internet Information Server).** Servidor de pàgines web que ve amb el paquet d'instal·lació dels sistemes operatius de Microsoft, s'ha utilitzat Windows XP amb Service pack 2.

- **Photoshop CS.** Programa d'edició gràfica amb el que l'autor està familiaritzat i que s'ha fet servir per a la creació dels gràfics de l'aplicació.

- **Microsoft Visual Studio.NET.** Entorn integrat de desenvolupament web, ha estat l'eina principal per a la implementació.

- **Microsoft SQL Server.** Sistema gestor de bases de dades.

## 1.5. Conclusions.

Si l'objectiu terminal d'aquest treball era aprofundir en el coneixement d'aquesta plataforma, crec francament que s'ha aconseguit.

Però tot i que s'hagi aconseguit l'objectiu principal, alguns aspectes de l'aplicació han quedat fora de l'abast del projecte, sobretot per la manca de temps.
Es podria fer una relació prou llarga de temes que resten pendents i que segurament el lector està pensant, malgrat tot se'n fa una resumida relació:

- Per una millor claredat les categories podrien dividir-se en sots-categories.

- Es podria arribar a algun acord amb una entitat bancària per a fer els pagaments via targeta de crèdit.

- Potser es podria simplificar el treball dels usuaris autoritzats per a realitzar canvis a la base de dades.

- Es podria distingir entre el simple visitant i el client que ja ha comprat.

Bé, i segurament moltes coses més, però no s'ha d'oblidar que aquest ha estat un projecte didàctic i com a tal ha estat una experiència prou enriquidora que pot servir de punt de partida pel desenvolupament en l'àmbit professional.

## 2. Glossari.

**Patró.** Estructura arquitectònica utilitzada en el disseny d'aplicacions.

**Model-View-Controller.** Model-Vista-Controlador, patró concret amb la finalitat de separar la codificació de la presentació de l'aplicació. En aquest projecte se'n fa servir una aproximació.

**Framework .NET.** De l'anglès, Bastiment. Marc on corren les aplicacions independentment del sistema operatiu. Llançat per Microsoft com a resposta a la plataforma Java.

**ASP.NET.** Arquitectura integrada en el Framework .NET per a desenvolupar aplicacions web.

**Objectes del domini.** Objectes del món real detectats en la fase d'anàlisi.

**Objectes del negoci.** Objectes que representen programari específic.

**Cas d'ús.** Acció que duu a terme un actor sobre un objecte, un actor no ha de ser necessàriament una persona.

**Compilador.** Programari que tradueix el codi font d'alt nivell que escriu el programador en codi màquina que entén el computador.

**Diagrama de Gantt.** Diagrama habitualment utilitzat per a representar les etapes d'un projecte.

**HTML.** Llenguatge d'etiquetes per a crear pàgines web estàtiques.

**Model de dades.** Estructura utilitzada per a guardar les dades amb un programari gestor de bases de dades.

**Servidor web.** Programari que controla i mostra planes web quan són sol·licitades.

**XML.** Format estàndard d'intercanvi de dades que permet la comunicació entre diferents sistemes.

**Control d'usuari.** Porció de codi que integra la vista i el controlador i que es pot reutilitzar fàcilment.

**DLL.** Dinamic Link library (Biblioteca d'enllaç dinàmic), codi executable que dóna suport a l'aplicació.

**Diagrama ER.** Entitat-relació, diagrama que mostra les taules d'una base de dades i les relacions que hi ha entre elles.

## 3. Bibliografia

### Llibres:

[Larman 2001] Craig Larman “UML y Patrones”, Prentice Hall.

[Balena 2002] Francesco Balena “Programación avanzada con Microsoft Visual Basic .NET”, McGraw Hill.

[Charte 2001] Francisco Charte Ojeda “Microsoft Visual Studio .NET – Guía práctica para usuarios”, Anaya Multimedia.

[Pratdepadua 2004] Joan Josep Pratdepadua Bufill “Domine ASP.NET”, Ra-Ma.

[Walther 2002] Stephen Walther “ASP al descubierto” Prentice Hall.

### Referències d’Internet:

[http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/netstart/html/sdk_netstart.asp, Novembre i Desembre 2004]

[http://www.webestilo.com/aspnet/aspnet00.phtml, 3 de Novembre de 2004]

[http://msdn.microsoft.com/architecture/patterns, Novembre i Desembre 2004]

[http://es.gotdotnet.com/quickstart/default.aspx, Novembre i Desembre 2004]

## 4. Annexos.

### 4.1. Annex A. Instruccions per a la instal·lació de l’aplicació.

Aquesta aplicació pot funcionar amb els sistemes operatius de Microsoft, Windows Xp Professional, Windows .NET Server i també amb Windows 2000 Professional o server.

S’ha d’instal·lar la plataforma Framework .NET de lliure distribució i tenir instal·lat i configurat l’IIS i l’SQLServer.

Un cop s’acompleixin aquests requeriments és molt senzill instal·lar l’aplicació:

- Copiar el directori amb l’aplicació que ja té l’estructura adient.

- Crear un directori virtual a l'IIS que apunti al directori de l'aplicació.

- Executar l'*script objDades.sql* des de SQLServer per a crear l'estructura de taules i els procediments emmagatzemats.

- Modificar els valors *CadenaConnexio, RutaImatge i ServidorCorreu* en l'arxiu *web.config* de l'arrel de l'aplicació amb els valors adients.

Després de seguir aquests passos si es posa en marxa el servidor web l'aplicació funcionarà correctament.

*  *  *