DEBRECENI EGYETEM
INFORMATIKAI KAR

DISZTRIBÚTORI WEBSHOP LÉTREHOZÁSA AJAX TÁMOGATÁSSAL, .NET FEJLESZTŐI KÖRNYEZETBEN.

Témavezetők:

Dr. Rutkovszky Edéné
egyetemi tanársegéd

Készítette:

Sós Viktor
programozó matematikus

DEBRECEN
2009

Bevezető

Szakdolgozatom téma választásánál fontos szerepet játszott a jelenlegi munkahelyem. Mivel üzletkötőként dolgozom az egyik hazai, vezető informatikai vállalatnál, ezért munkám elengedhetetlen részét képezi a disztribútorok webshopjainak folyamatos használata ügyfeleim ajánlatkéréseinek mielőbbi kiszolgálása érdekében. Ezen webáruházak kapcsolatot teremtenek a disztribútor saját vállalatirányítási rendszerével, ezáltal hasznos információkat nyújtva a viszonteladók kereskedői részére, mint például a termékek adott partnerre vonatkozó árai az aznap érvényes devizaárfolyamok figyelembevételével, aktuális raktárkészletek, hiányzó, raktáron nem lévő termékek esetén a várható szállítási határidő.
Szakdolgozatom célja egy olyan disztribútori webshop alkalmazás, mely interaktív kapcsolatot teremt a két vállalat között. A webes alkalmazások hátrányának megszüntetésére, csökkentésére, valamint a felhasználói élmény fokozására egy a napjaikban elterjedt web fejlesztési technikát az AJAX –ot használom. Az AJAX támogatásával elérhető, hogy az internetes alkalmazások felvegyék a versenyt a személyi számítógépeken futó programok interaktivitásával. A dolgozatom egy ilyen alkalmazás felépítését és működését mutatja be a Microsoft .NET fejlesztői környezetben.

AZ ASP.NET bemutatása

Az ASP.NET

Active Server Pages (ASP) azaz a klasszikus ASP, a Microsoft első szerveroldali szkript technológiája volt a dinamikusan generált web oldalakhoz. Az ASP az IIS (Internet Information Service) webkiszolgáló része már kezdetektől fogva tartalmazza ezt a programot, mely ingyenes komponense a Windows szervereknek a Windows 2000 szervertől kezdődően. Az ASP nem más, mint a HTML kódba ágyazott speciális programozási módszer, amely lehetővé teszi a beágyazott szkriptek futtatását. Ahhoz, hogy az IIS egy fájlt tényleg ASP oldalként kezeljen a kiterjesztésének .aspx –nek kell lennie, amennyiben ez nem teljesül úgy a kód végrehajtás nélkül eljut az „ügyfélhez” , mintha egy HTML oldal lenne.
ASP.NET az ASP technológia jogutódja ként jött létre a .NET framework 1.0 verziójának 2002 januári megjelenésével, melyet a Microsoft fejlesztett ki.

ASP.NET működése

Az ASP.NET egy szerver oldali programozási nyelv. Ebből következik, hogy valamilyen módon meg kell oldani a kliens és a szerver oldal közötti kommunikációt. Erre szolgál a HTTP protokoll. A kliens számítógépén futó böngésző, ezen keresztül kéréseket tud küldeni a szervernek. ASP.NET-nél általában háromféle HTTP parancsot használunk. Az alap parancsunk a GET, amely egy URI cím (az URL egy kibővített szabványa) segítségével határozza meg azt az oldalt, amelyet el szeretnénk érni. Az alábbi példa ezt jól szemlélteti.

GET /default.aspx http/1.1
Host: www.kozbeszerzes.gov.hu

A Host segítségével adjuk meg azt, hogy ne a gépünkön keresse a default.aspx oldalt, hanem a kozbeszerzes.gov.hu domain alatt. A másik fontos parancsunk a HEAD, amellyel a kért oldal fejrész adatait érhetjük el.
Minden HTTP kérésnél a böngésző az oldalkérésen kívül az őt működtető rendszerről számos információt küld a kiszolgálónak, mint például támogatja-e a böngésző a javaszkriptet. Erre az információra azért lehet szükség, mert az ASP.NET vezérlők egy része kliens oldali szkripteket használ ilyenek például a különböző validation kontrol -ok.

A harmadik és egyben a leggyakrabban használt parancs az a POST. Ezzel a paranccsal tud információt cserélni a kliens és a szerver. A GET paranccsal felvesszük a kapcsolatot a weboldalakkal, majd onnantól kezdve a POST paranccsal bonyolítjuk le a további kommunikációt. A szerver oldalon a kérésértelmező és kiszolgáló alkalmazás szerepét az IIS tölti be. Az IIS feladata az, hogy a különböző kérésekre dinamikusan generáltasson valamilyen HTML alapú választ, melyet a kliens böngészője értelmezni tud. Amikor a kliens elkéri a default.aspx lapot, akkor az IIS egy segédalkalmazást hív, amely tudja értelmezni a fájlban lévő kódokat. Ez általában az aspnet_isapi.dll.
A kliens és szerver közötti kommunikáció nem folyamatos, azaz szétkapcsolt adatarchitektúrát alkalmaznak.

A Microsoft .NET keretrendszer

A .NET keretrendszer egy a Microsoft által kifejlesztett objektum központú fejlesztői környezet, melynek segítségével webes alkalmazásokat fejleszthetünk. A .NET keretrendszert két fő összetevő alkotja az egyik a közös nyelvi futtató környezet (CLR), a másik pedig a .NET osztálykönyvtára.
A közös nyelvi futtatókörnyezet (CLR) kezeli a memóriát, a szálak, a kódok végrehajtását, biztonsági ellenőrzését, szerkesztését és más rendszereszközöket. Kezeli az objektum-elrendezést és az objektumhivatkozásokat, felszabadítva az objektumokat, ha már nincsenek használatban.

A .NET keretrendszer osztálykönyvtára újrahasználható típusok gyűjteménye, amely szorosan együttműködik a közös nyelvi futtatókörnyezettel. Az osztálykönyvtár objektumközpontúsága miatt, olyan típusokat szolgáltat, amelyekből az alkalmazások további funkciókat tudnak származtatni így típusai könnyen használhatóak.

.NET tervezésénél kitőzött célok:

- Objektum orientált környezet
- Lokálisan vagy távolban végrehajtható kód
- Nem megbízható kódok biztonságos végrehajtása
- Ne legyen olyan lassú, mint a szkripteltelt környezetek
- Közös szolgáltatások mindenféle típusú alkalmazás részére
- A kommunikáció szabványos protokollokra épüljön
- A legtöbb nyelv támogatása

.NET által támogatott nyelvek:

Beépített nyelvek:

- C#, a C++-hoz és a Sun Java nyelvéhez hasonló objektum-orientált nyelv
- JSzkript .NET, Microsoft JSzkript nyelvének fordított verziója
- J#, a Java és a J++ programozók átmeneti nyelve a .NET keretrendszer felé
- Managed C++, a C++ egy változata a .NET platformra
- Visual Basic .NET, a klasszikus Visual Basic egy továbbfejlesztett, objektum-orientált, többszálas verziója

Külső forrásból elérhető nyelvek a (teljesség igénye nélkül)

- APL
- COBOL
- Component Pascal
- Delphi 8 és Delphi 2005
- Eiffel
- F#
- FORTRAN
- IKVM, Java
- Lisp
- Perl
- RPG
- Smalltalk

Felhasználói autentikáció

Az ASP.NET beépített vezérlőkkel támogatja a felhasználói autentikációt. A szakdolgozatom készítése közben az alábbi Login kontrolokat használtam fel.

- Login kontrol: A kontrol lehetővé teszi számunkra a felhasználói autentikációt. Elég csak a weboldalunkra húzni a Visual Studio toolbox-ából és ugyan a default beállításokkal, de innentől kezdve kezeli a felhasználókat. A vezérlő több, már az előzőkben bemutatott eszközökből épül fel. Tartalmaz két darab *TextBox*-ot a felhasználónév és jelszó bevitelére. Egy *CheckBox*-ot arra az esetre, ha a felhasználó automatikusan autentikálni akarja magát, hogyha legközelebb is az oldalra látogat. Valamint tartalmaz még egy gombot is a belépéshez.

- LoginView kontrol: A *LoginView* kontrol lehetővé teszi, hogy különböző információkat tudjunk megjeleníteni a névtelen és a bejelentkezett felhasználók számára. A beállításokra kétfajta sablont használhatunk az *AnonymusTemplate* és *LoggedInTemplate-t.*

- CreatedUserWizards kontrol: A kontrol használatával egy regisztrációs felület alakítható ki, a default beállításokat saját ízlésünknek megfelelően testre szabhatjuk, valamint segítségével növelhetjük az oldal biztonságát.

ASP.NET szerver oldali kontrolok

A kiszolgáló oldali vezérlők olyan .NET objektumok, amelyek a felületi elemek megjelenítéséért és viselkedéséért felelnek. Feldolgozzák a kliens oldalról érkezett adatokat, eseményeket generálnak a szerver oldalon. Számtalan tulajdonságuk segítségével deklaratív módon testreszabhatóak, mind a viselkedésük, mind pedig a megjelenésük. Szerver oldalon állapotmentesek, állapotuk („ViewState") a lappal együtt kerül továbbításra. ID tulajdonsága segítségével lehet az oldalhoz tartozó kódból elérni.

A szerver kontrolok:

- **HTML Szerver Kontrolok** –HTML címkék
- **Web szerver kontrolok** – ASP.NET címkék
- **Érvényesítő (validator) szerver kontrolok** – a beviteli mezőkhöz kapcsolható érvényesítő kontrolok

HTML szerver kontrolok

Egy HTML TAG-et tesz elérhetővé a kódból, ami System.Web.UI.HtmlControls névtérben található. Ezek a vezérlők gyorsabbak, mint a bonyolultabb funkcionalitással bíró webkontrolok. Programozhatóvá tehetők a " runat="server" " tulajdonság megadásával így tudatjuk a szerverrel, hogy a HTML vezérlő szerver kontrolként funkcionál. Minden kontrol saját egyedi azonosítóval " ID" van ellátva melyeket automatikusan, illetve manuálisan beállíthatunk ezzel lehetővé téve, hogy futási időben hivatkozzunk rá. Tulajdonságait a kontrol nyitó *TAG*-jében lehet megadni, mint például az alábbi text bevitel mezőnél is látható.

```
<p>
        <input id="Text1" runat=server type="text" style="width:100px;"
/></p>
```

A runat=server tulajdonságból következik hogy szerveroldali vezérlőről beszélünk szélessége 100 pixel azonosítója Text1.

Web szerver kontrolok

Bonyolultabb funkcionalitással bíró kontrolok, melyek a System.Web.UI.WebControls névtérben találhatóak. Futási időben is létre tudjuk hozni dinamikusan

Az alkalmazás készítésekor gyakrabban használt webszerver kontrolok

- Label : szöveg megjelenítésére szolgál
- Textbox vezérlő: szövegbevitelére szolgál
- Button, ImageButton: események generálásra szolgál
- Checkbox, ChecboxList: kiválasztásra szolgál
- Placeholder: dinamikus tartalom megjelenítésére szolgál
- Panel: dinamikus és statikus tartalom megjelenítésére szolgál

Validátor (érvényesítő) szerver kontrolok

A felhasználói beviteli mezők érvényesítésére szolgáló szerver kontrolok. A beviteli mezőkhöz kötve lehet ellenőrizni az adott tartalmat, hogy az a programozó által definiált értéknek megfelelően lett-e kitöltve vagy egyáltalán ki lett-e töltve. A validátor ControlToValidate tulajdonságában meg kell adni a beviteli mező azonosítóját (ID).

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="DropDownList1" ErrorMessage= "Kötelező választani!">
</asp:RequiredFieldValidator>
```

A fenti validátor kötelezi a felhasználót, hogy válasszon a `"DropDownList1"` ID-val ellátott legördülő listából. Amíg ezt a felhasználó nem teszi meg, addig az oldalon nem váltható ki postback esemény és visszatér egy hiba üzenettel, amit a programozó definiálhat.

Beépített validátorok:

- RequiedFieldValidator: Használatakor a hozzá csatolt beviteli mezőt vizsgálja, hogy tartalmaz e bevitt értéket, vagy üres. Amennyiben üres, a programozó által módosítható hibaüzenettel tér vissza.

- CompareValidator: segítségével két beviteli mező értékét vagy egy beviteli mező és egy konstanst lehet összehasonlítani. A ControlToValidate-nél azt kell beállítani melyik beviteli mezőt, a ControlToCompare tulajdonságnál, pedig azt hogy melyik beviteli mezővel. Amennyiben konstans értékkel akarjuk összehasonlítani, akkor a ValueToCompare tulajdonságot használjuk.

- RangeValidator: Tartományérvényesítő. Ezzel a beviteli mező alsó és felső határértékét lehet szabályozni. Ezekhez kötelezően meg kell adni a MinimumValue illetve a MaximumValue tulajdonságokat. A Type tulajdonsággal lehet szabályozni a beírt adat adattípusát.

- RegularExpressionValidator: Reguláris kifejezés érvényesítő. Ez azt ellenőrzi, hogy az adat megfelel-e egy reguláris kifejezés által definiált mintának. Vagyis azt nézi, hogy a felhasználó az általunk megadott séma, minta alapján töltötte-e ki a beviteli mezőt.

ADO.NET

Adatbázis kezelése ADO.NET

Az ADO.NET is a .NET keretrendszer része, mely elsősorban a relációs adatbázisokhoz való hozzáférést támogatja. Két féle adatelérési réteget nyújt számunkra. Az egyik a kapcsolt (managed provider) a másik pedig a kapcsolat nélküli réteg a (dataset). A kapcsolat réteg segítségével kapcsolódunk az adatbázishoz és tudjuk lekérni az adatokat . A lekért adatokból egy másolatot készítünk (a másolatot adatkészletnek vagy dataset –nek nevezzük), majd a későbbiekben ezzel dolgozunk tovább, így egy szétkapcsolt adatarchitektúrát kapunk. A szétkapcsolt adatarchitektúra előnye, hogy kevesebb az adatforgalom a kliens és a szerver között. A hátránya azonban az, hogy nem mindig a legfrissebb adatokat látjuk.

ADO.NET adatelérési modell

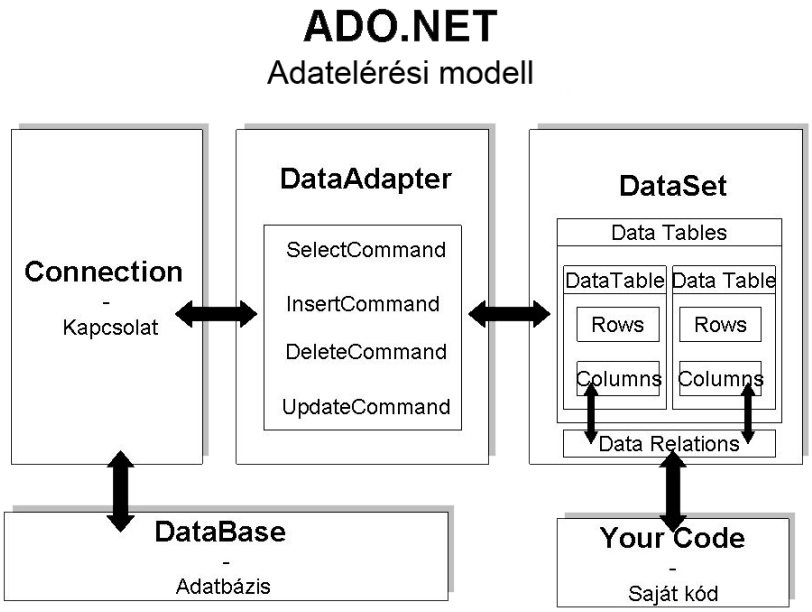
Az ADO.NET mint, ahogy az előzőekben is említettem egy szétkapcsolt adatarchitektúrát valósít meg. Ez a következőképpen működik. Az adatbázisból létrehozunk egy másolatot majd változtatásokat visszavezetjük az adatbázis szerverbe. Az adatbázis másolatot adatkészletnek, eredeti nevén DataSet-nek , az adatszinkronizációért felelős objektumot pedig adatillesztőnek, eredeti nevén DataAdapter-nek nevezzük.

DataAdapter

A DataAdapter adja ki a megfelelő parancsot az adatbázisnak majd a válaszképpen megkapott eredményt tárolja el egy adatkészletben. Az adatillesztő kizárólag akkor kapcsolódik az adatbázishoz, ha valamilyen műveletet akarunk végrehajtani amint ez megtörtént a kapcsolat inaktívvá válik. A DataAdapter négy Command objektumot tartalmaz melyek közül kötelezően a SelectCommand-ot kell megadni az adatkészlet feltöltéséhez.

DataSet

A lényegét tekintve a DataSet is egy adatbázis így tulajdonságai megegyeznek az eredetivel. Egyszerre több táblát tárolhatunk benne, melyek között relációkat adhatunk meg. A DataSet-et adatokkal, a DataAdapter Fill metódusának segítségével tölthetjük fel. A változások visszavezetéséhez pedig az Update metódust használjuk!



1. ábra ADO.NET adatelérési modell

AJAX

Az AJAX webfejlesztési technika bemutatása

Az Ajax (**A**synchronous **J**avaScript **A**nd **X**ML) egy olyan webfejlesztési technika, amely lehetővé teszi interaktív webes alkalmazások fejlesztését. Segítségével, mikor a felhasználó a linkre, gombra kattint, vagy valamit módosít az oldalon nem kell az oldal teljes tartalmát újra lekérni a szervertől hanem kizárólag csak a szükséges részeket kell frissítenie. Mivel így csökken a kliens szerver közötti adatforgalom ezáltal az oldal számottevően gyorsabban reagál a felhasználó interaktivitásaira, növelve ezzel a felhasználói élményt.
A szervertől kapott adatokból a HTML kód a kliens oldalon a használt böngészőben jön létre javaszkript segítségével. Napjainkban már a felhasználói számítógépek viszonylag nagy teljesítményűek és gyorsak, terheltségük pedig alacsony, ezért ez a megjelenítésnél nem okoz problémát. Egy-két kliens gép esetleges lassúsága miatt, a párhuzamosan az oldalt vagy szervert használó klienseket sem akadályozza a gyors kiszolgálásban. Nagyobb látogatottságú oldalak esetén a kliens oldali HTML generálás, ezáltal nagyobb előnyt jelent.
Az AJAX az itt leírt módszerek együttes használatával jelentős oldal letöltési sebesség növekedést és szerverterhelés csökkentést ér el.

Az AJAX a következő technikák kombinációja:

- XHTML (vagy HTML) és CSS a tartalom leírására és formázására.
- DOM kliens oldali szkript nyelvekkel kezelve a dinamikus megjelenítés és a már megjelenített információ együttműködésének kialakítására.
- XMLHttpRequest objektum az adatok aszinkron kezelésére a kliens és a webszerver között. Néhány Ajax keretrendszer esetén és bizonyos helyzetekben IFrame-et használnak XMLHttpRequest objektum helyett.
- XML formátumot használnak legtöbbször az adattovábbításra a kliens és a szerver között, bár más formátumok is megfelelnek a célnak, mint a formázott HTML vagy a sima szöveg.

AJAX Control Toolkit

Az Ajax Control Toolkit egy nyílt forráskódú **community project**, azaz a Microsoft-on kívüli fejlesztők együttműködésének eredménye. A Control Toolkit olyan komponensek-gyűjteménye, melyek lehetővé teszik, hogy RIA –t (Rich Internet Application) fejleszthessünk, gazdag kliens oldali felülettel. A Control Toolkit egyik legnagyobb előnye, hogy nem csak Internet Explorer 6 –ra optimalizált kódról van szó, hanem Internet Explorer7, FireFox és Safari alatt is garantálja a megfelelő működést és a helyes megjelenítést.
A nevéből is jól látszik, hogy főleg AJAX Extension –re illetve AJAX Library Framework –re épül. Mivel a Control Toolkit is egy keret rendszer ezért akár saját komponenseket is fejleszthetünk gyorsan, hatékonyan, melyhez egy eléggé nagy Software Development Kit nyújt segítséget.
Az AJAX Control Toolkit projekt régebben az „AJAX projekt“ része volt, amely ATLAS fedőnév alatt futott. A megjelenés előtt átnevezték, szétszedték, így lett az ATLAS –ból:

- AJAX Control Toolkit
- AJAX Extension
- AJAX Library

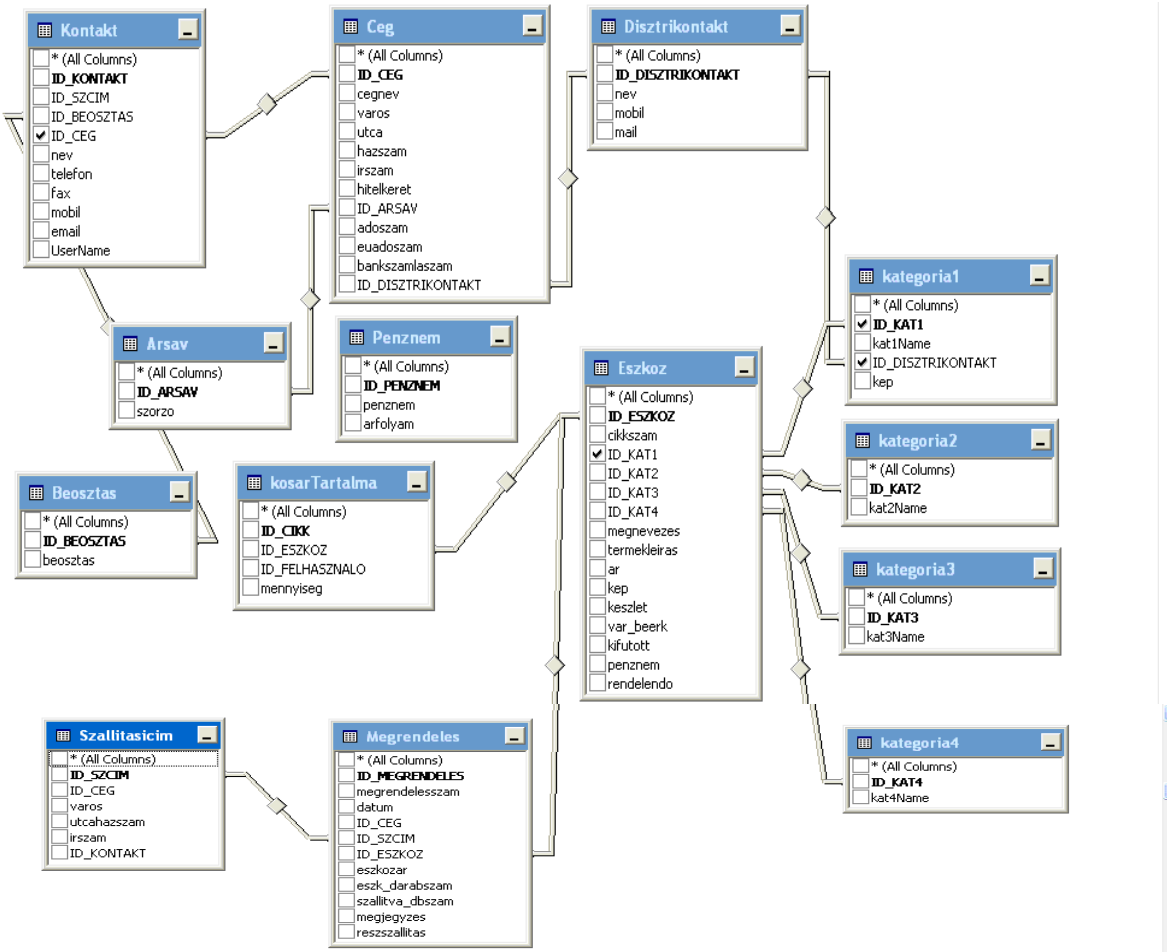
Az AJAX Control Toolkit három fő részből áll:

- Control
- Behavior
- Extender

A kontrol –ok olyan vezérlők, melyek AJAX –os funkcionalitással rendelkeznek.
A behavior, mint ahogy nevéből következik, egy már meglévő ASP.NET vezérlő viselkedését, funkcionalitását változtatja meg (általában kibővíti). Ez felelős a kliens oldali részért, ezért Javaszkriptben lett megírva. A behavior –öket általában együtt használjuk az Extender –ekkel. Ennek oka az, hogy az Extender a szerver oldali párja a dolognak. Ez a sok dolog egyetlen egy dll –ben kapott helyet, melyet nem kell a GAC –ban (global assembly cache) tárolnunk, hogy használhassuk.

Disztribútori webshop működésének és felépítésének bemutatása

Ebben a fejezetben a fentiekben bemutatásra került technológiák segítségével az általam elkészített disztribútori webshop működésével, felépítésével ismerkedhetünk meg. Az alkalmazás megvalósításához nélkülözhetetlen egy adatbázis, melyben a szükséges adatokat tudjuk tárolni. Ehhez az alábbiakban definiált SQL relációs adatbázist használom.



2. ábra adatbázis séma

Adatbázis felépítése

A Cég táblában tároljuk a viszonteladó partnerek adatait. Amint a kapcsolatokból is jól látszik, minden céghez hozzá van rendelve egy kapcsolattartó a webshop/disztribútor üzemeltetőjétől, aki az adott cég kiszolgálásáért felelős. Ez abban az esetben szükséges, ha a viszonteladó partner részéről egyedi igény, probléma merülne fel és ezt a honlapon keresztül nem tudja elintézni. Ez például lehet egy testre szabott ár kérése esetleges nagyobb tétel beszerzése esetén. Az esetek többségében a disztribútornál a viszonteladó partnerek ársávokba vannak rendezve, amit maga a disztribútor határoz meg a viszonteladó forgalma alapján. A lényegét tekintve nem más, mint egy szorzó, amit az adott cégre alkalmaznak. A *Ceg* táblához kapcsolódik továbbá egy Kontakt tábla. Ebbe a táblába a viszonteladó partner részéről kerülnek bele a kapcsolattartók például az ügyvezető, a számlázásért felelős személy, a kereskedő, a logisztikus stb. Mivel a Kontakt tábla több cég esetében igen mértesre nőhet és felesleges lenne az adatokat duplikáltan tárolni, ezért a kapcsolattartók beosztását egy külön táblában tárolom le, ezáltal elegendő egy int azonosítót tárolnom minden egyes személynél, ami a beosztás elsődleges kulcsa. Az Eszköz táblában kerülnek tárolásra a termékek adatai, cikkszám, megnevezés, egy rövid termék leírás a raktáron lévő mennyiség és hogy milyen kategóriába tartoznak. A kategória táblák a normalizálás után új táblákba kerültek a szűrések könnyebb kezelhetőségét hivatottak szolgálni, amiről a későbbikben részletesebben beszélek. Ha egy felhasználó bejelentkezés után rendel a webshopról akkor termék elsőnek a Kosár táblában tárolódik, majd ha befejezte a szükséges eszközök kiválasztását és ténylegesen megrendeli, akkor a kosárból törlődik és átkerül a megrendelések táblába amit a felhasználó már módosítani nem tud.

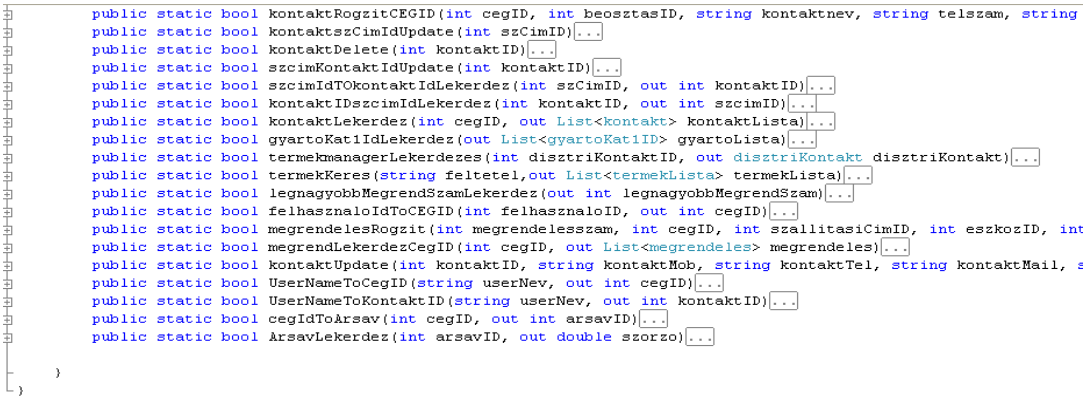
A táblák elsődleges kulcs segítségével vannak összekapcsolva és 3. normálformában definiáltak.

A webalkalmazás felépítésénél külön kell választani a regisztrált illetve a nem regisztrált felhasználókat, ezért megkülönböztetünk regisztrált felhasználói illetve látogatói nézetet.

A programban az adatbázis műveletekre vonatkozó metódusok teljes mértékben elkülönítésre kerültek az adatbázis lekérdezéseit, frissítéseit...stb megvalósító metódusok egy külön 'db' nevezetű névtéren belül, külön osztályokban lettek létrehozva, melyet a következő ábra szemléltet:

```
namespace db
{
    public struct kategoriak...
    public struct kategoriak2...
    public struct penznem...
    public struct beosztas...
    public struct szallitasicim...
    public struct arsav...
    public struct kontakt...
    public struct cegLista...
    public struct gyartoKat1ID...
    public struct disztriKontakt...
    public struct termekLista2...
    public struct termekLista...
    public struct kategoriaElemek...
    public struct kosarTermekLista...
    public struct megrendeles...

    public class Class1...
    public class abEditLekerdez
    {
        public abEditLekerdez()...

        public static bool katNameToKatID(int listBoxSzama, string katName, out int katID)...
        public static bool catIdLekerdez(int katSzam, string katName, out int katID)...
        public static bool katIDToKatName(int listBoxSzama, int katID, out string katName)...
        public static bool kategoriaLekerdezesek(string kat1Name, string kat2Name, string kat3Name, string kat4Name
        public static bool kategoriaLekerdezesek2(string kat1Name, string kat2Name, string kat3Name, string kat4Nam
        public static bool kategoriaLekerdezes(int kategoriaSzam, out List<kategoriaElemek> katElemek)...
        public static bool termekListaLekerdezes(string kat1Name, string kat2Name, string kat3Name, string kat4Name
        public static bool termekLekerdezes(int termekID, out termekLista termekJellemzok)...
        public static bool termekListaLekerdezes2(string kat1Name, string kat2Name, string kat3Name, string kat4Nam

        public static bool cikkszamTotermekLekerdezes(string cikkszam, out termekLista2 termekJellemzok)...
        public static bool ujCikkFelvitel(string cikkszam, string megnevezes, string termekleiras, string imgPic,in
        public static bool termekUpdate(int termekID,string cikkszam, string megnevezes, string termekleiras, strin
        public static bool katFelvitel(int katSzam,string katNev)...
        public static bool cikkszamLekerdezes(string cikkszam, out int
        public static bool cikkszamToID(string cikkszam, out int ID)...
        public static bool arfolyamLekerdezes(int penznemID, out penznem penznemLista)...
        public static bool penznemToArfolyam(string penznemNev, out int arfolyam)...
        public static bool bevetelezesUpdate(int termekID, int mennyiseg, int ar)...
        public static bool kosarTartalomLekerdezese(int felhasznaloID, out List<kosarTermekLista> kosarTermekLista)
        public static bool kosarbolEgyTermekLekerdezese(int felhasznaloID, int termekID, out List<kosarTermekLista>
        public static bool kosarbaTesz(int termekID, int mennyiseg, int szerepelE, int felhasznaloID)...
        public static bool kosarUpdateNovel(int termekID, int mennyiseg, int szerepelE, int felhasznaloID)...
        public static bool kosarUpdateCsokkent(int termekID, int kulonbseg, int szerepelE, int felhasznaloID)...
        public static bool kosarbanVanEMarIlyenTermek(int termekID, int felhasznalo, out int szerepelE)...
        public static bool kosarbolEszkozTorlese(int termekID, int felhasznalo)...
        public static bool kosarbolEszkozTorleseMegrendelesEseten(int termekID, int felhasznalo)...
        public static bool kosarbolEszkozFoglaltsaganaklekerdezese(int eszkozID, out int lefoglaltMennyiseg)...
        public static bool arsavLekerdezes(out List<arsav> arsavLista)...
        public static bool arsavIdTOarsavLekerdezes(int arsavID,out List<arsav> arsavLista)...
        public static bool cegRegisztracio(string cegNev, string varos, string utca, string hazsz, string irsz, int
        public static bool cegListaLekerdezes(out List<cegLista> cegLista)...
        public static bool cegAdatokLekerdezes(int cegID, out cegLista cegLista)...
        public static bool szCimLekerdezCegIDalapjan(int cegID, out List<szallitasicim> szCimLista)...
        public static bool adoszamToCEGID(string adoszam, out int cegID)...
        public static bool nevTocegListaLekerdezes(string  cegnev, out cegLista cegLista)...
        public static bool cegListaUpdate(int cegID,string cegNev, string varos, string utca, string hazszam, strin
        public static bool szallitasicimRogzites(int cegID, string varos, string utcaHazszam, string irszam,int kon
        public static bool beosztasLekerdez(out List<beosztas> beosztasLista)...
        public static bool beosztNevToIdLekerdez(int beosztasID, out string beosztMegnev)...
        public static bool szCimIdToKontakt(int szCimId, out kontakt Kontakt)...
```

class System.String
Represents text as a series of Unicode characters.



Az adatbázis műveleteket végrehajtó metódusok BOOLEAN típusúak, mivel egy kivételkezelő TRY-CATCH ágában vizsgálom, hogy a lekérdezés, a frissítés vagy a törlés megfelelően végrehajtódott-e. Amennyiben az SQL parancs végrehajtása során probléma lép fel úgy a try blokkban kivétel keletkezik és a program futása a TRY blokkhoz tartozó CATCH ágban folytatódik így a visszatérési érték False értékű lesz. Ha kivétel nem következik be a végrehajtás során, akkor a függvény True értékkel tér vissza.

Látogatói nézet

A rendszer fejlesztésénél az egységes megjelenítés érdekében egy masterpage-et használtam, ami minden egyes oldal betöltődésekor egységes keretet biztosít mind a regisztrált felhasználó, mind pedig a látogató részére. Használatával az egységes stílust egyetlen helyen történő változtatással, egyszerűen tudjuk módosítani.

A MasterPage felépítése:

<table>
<tr><td colspan="3">Fejléc</td><td colspan="2">Felhasználói autentikáció</td></tr>
<tr><td colspan="3">Menü</td><td colspan="2">Kereső</td></tr>
<tr><td colspan="5">Banner</td></tr>
<tr><td colspan="5">A masterpage oldalakat használó webformok betöltésének helye</td></tr>
<tr><td>LinkButton
ModalPopupExtender</td><td>LinkButton
ModalPopupExtender</td><td>LinkButton
ModalPopupExtender</td><td>LinkButton
ModalPopupExtender</td><td>LinkButton
ModalPopupExtender</td></tr>
</table>

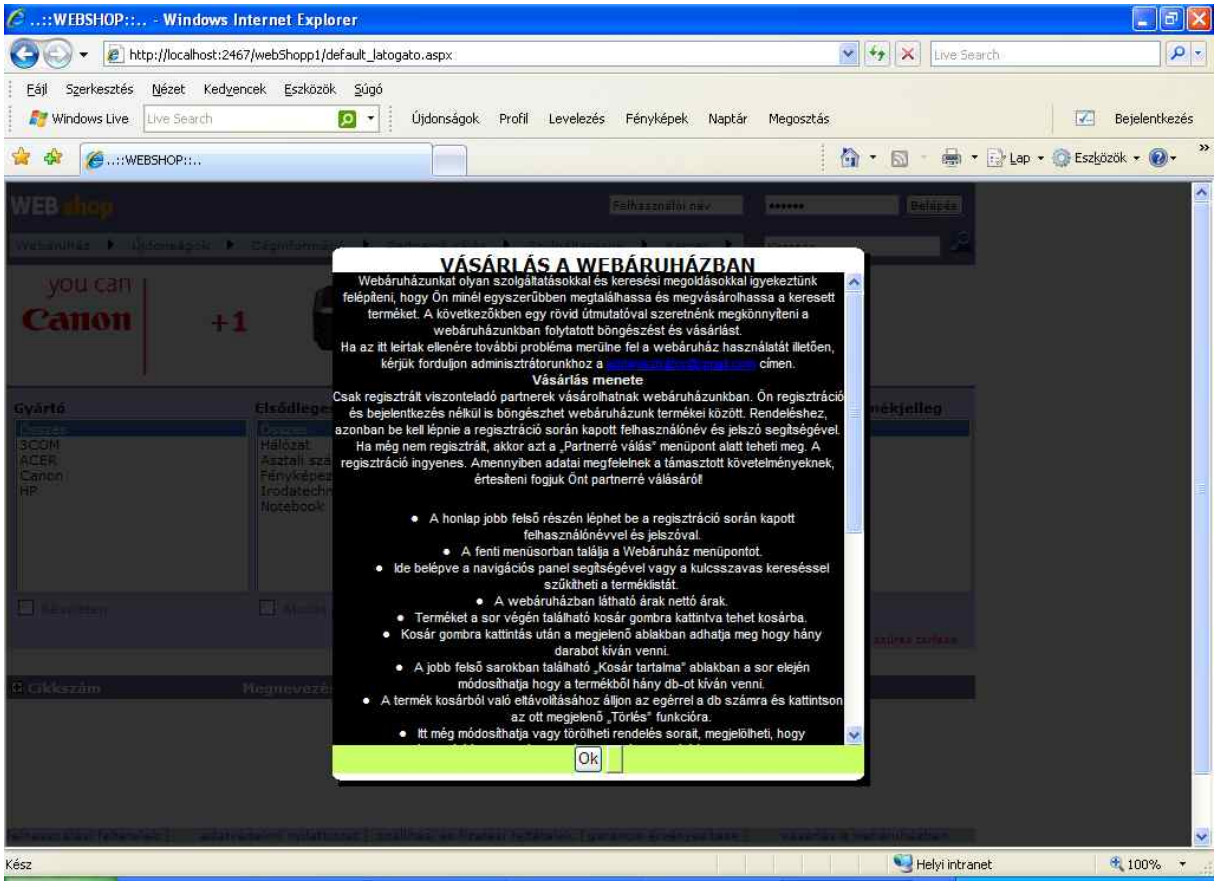
A masterpage funkcionális felépítése

A *Masterpage* alsó részén szereplő táblázatban *LinkButton* -okat helyeztem el, amire ha a felhasználó rákattint, akkor az oldal közepén megjelenik egy panel, amely információkat tartalmaz természetesen attól függően, hogy melyik gombra kattintunk. Ezt egy az Ajax Control Toolkit -ben szereplő előre definiált programozói eszköz a *ModalPopupExtender* segítségével valósítottam meg.

Megjeleníteni a masterpage.master.cs fileban szereplő *LinkButton* klikk eseményére a *ModalPopupExtender show()* metódusával tudjuk.

```
protected void LinkButton5_Click(object sender, EventArgs e)
{
    ModalPopupExtender1.Show();
}
```

A *ModalPopupExtender* használatakor rétegek képződnek a megjelenő ablak rárakódik az alatta elhelyezkedő oldalra és az alsó réteg, a webshop mindaddig inaktívvá válik, míg a felsőt be nem zárjuk. Az alsó réteg közben látható, de semmilyen felhasználói interaktivitásra nem reagál.



7. ábra ModalpopupExtender használata

Az oldal jobb felső sarkában egy *Login* kontrolt helyeztem el a felhasználók hitelesítésére így

a webshop használata során, bármelyik formról lehetőség nyílik a bejelentkezésre.

A táblázat második sorában baloldalon jelenik meg a navigációs *menü*. A felhasználói és a látogatói nézet függetlenül az egységes masterpage használatától ebben az egy dologban eltér egymástól. A masterpage két különböző menüt tartalmaz egyszerre, amiket az alábbi kód részlet segítségével tudok megjeleníteni attól függően, hogy bejelentkezett, vagy nem bejelentkezett felhasználó használja az oldalt.

A felhasználói vagy a látogatói menüt megjelenítő kód részlet a MasterPage.master File-ból:

```
protected void Page_Load(object sender, EventArgs e)
    {

        if (Page.User.Identity.IsAuthenticated) //ha be van jelentkezve
        {
            pnlLogin.Visible = true;
            Login1.Visible = false;
            Menu1.Visible = true;
            Menu3.Visible = false;
        }
        else //ha nincs bejelentkezve
        {
            pnlLogin.Visible = false;
            Login1.Visible = true;
            Menu1.Visible = false;
            Menu3.Visible = true;
        }

    }
```

Minden a masterpage-et használó oldal betöltődésekor a Page_Load metódusban megvizsgáljuk, hogy bejelentkezett felhasználó vagy látogató használja az oldalt. Ha az IF feltételben megadott *IsAuthenticated* boolean visszatérésű metódus *True* értékkel tér vissza abban az esetben *pnlLogin* azonosítóval rendelkező panel és a *Menu1* látható a *Login1* és a *Menu3* vezérlők láthatóságát pedig False értékre állítjuk. Ha a visszatérési érték false akkor az előzőekben említett folyamatok ellenkezője történik.
A Page_Load metódus egy olyan esemény, melyet az ASP.NET megért és az oldal betöltésének pillanatában hajt végre.

Kereső funkció bemutatása

A táblázat navigációs menüt tartalmazó sorában jobb oldalra egy *TextBox*-ot egy *Label*-t és egy *ImageButton*-t helyeztem, amiknek a segítségével a kereső funkciót valósítottam meg. A keresendő szöveg bevitelére a TextBox-ot használom.

A keresendő termék adatait a `' ','-','/',';'` elválasztó jelekkel lehet megadni.

A keresést a *db* névtérben definiált termekListaLekerdezes2() metódus alkalmazásával valósítom meg. Az elválasztó jelek mentén a *split()* metódus használatával feldarabolom a *TextBox*-ban szereplő stringet majd egymásba ágyazott *SQL select* utasítással a leszűröm az adatbázis *Eszköz* táblájából.

Részlet a termekListaLekerdezes2() metódusból

```
. . .

public static bool termekListaLekerdezes2(string kat1Name, string kat2Name, string
kat3Name, string kat4Name, string tbSearchText, out List<termekLista> termekLista)
{

termekLista = new List<termekLista>();

string feltetel1, feltetel2, feltetel3, feltetel4, sqlParancsKiegeszito="";

string[] keresesiFeltetel=tbSearchText.Split(new Char[]{' ','-','/',';' });

. . .

for (int i = 0; i < keresesiFeltetel.Count(); i++)

if (i == 0)
  {

        sqlParancsKiegeszito += "WHERE ID_ESZKOZ IN (SELECT ID_ESZKOZ FROM
        Eszkoz AS Eszkoz_" + i.ToString() + " WHERE (" + feltetel1 + " AND " +
        feltetel2 + " AND " + feltetel3 + " AND " + feltetel4 + " AND
        (cikkszam LIKE '%" + keresesiFeltetel[i] + "%' OR megnevezes LIKE '%"
        + keresesiFeltetel[i] + "%' OR termekleiras LIKE '%" +
        keresesiFeltetel[i] + "%')))";

}

else
{

        sqlParancsKiegeszito += "AND ID_ESZKOZ IN (SELECT ID_ESZKOZ FROM
        Eszkoz AS Eszkoz_" + i.ToString() + " WHERE (" + feltetel1 + " AND " +
        feltetel2 + " AND " + feltetel3 + " AND " + feltetel4 + " AND
        (cikkszam LIKE '%" + keresesiFeltetel[i] + "%' OR megnevezes LIKE '%"
        + keresesiFeltetel[i] + "%' OR termekleiras LIKE '%" +
        keresesiFeltetel[i] + "%')))";

}
```

```
try
{
        SqlConnection kapcsolat = new SqlConnection();

        kapcsolat.ConnectionString=ConfigurationManager.ConnectionStrings["webShopDB"].ToStrin
        g();

        SqlCommand parancs = new SqlCommand();
        parancs.Connection = kapcsolat;

        if (tbSearchText.Length == 0)
        {

                parancs.CommandText = "SELECT * FROM Eszkoz WHERE (" + feltetel1 + " AND " +
                feltetel2 + " AND " + feltetel3 + " AND " + feltetel4 + ")";

         }

       else
       {
                parancs.CommandText = "SELECT * FROM Eszkoz " + sqlParancsKiegeszito;

       }

          . . .
```

Fontos megemlíteni, hogy ha a kereső segítségével több olyan terméket is találunk ami megfelel a keresési feltételeinknek akkor ezen leszűrt termékeknél is működik a ListBox –ok használatával megvalósított szelektálás.

Akár a kereső akár a *ListBox*-ok használatával megvalósított szelektálást a ”default_latogato.aspx” és a ”Default.aspx” oldalakon a szűrés törlése gombra történő kattintás segítségével tudjuk feloldani.

A látogatói nézet menüsorának felépítése

- Webáruház
- Újdonságok
    - Kezdőoldal
- Partnerré válás
    - Offline regisztráció
- Szolgáltatások
    - Garanciális ügyintézés
    - Visszáru (RMA)
- Karrier
    - Álláshirdetések
    - Jelentkezési lap

Az újdonságok menüpont alatt található látogatói nézetben, a kezdőlap fül ez a már előbb említett kezdőlapra navigál.

A ”Céginformáció” menüpont alatt az alábbi fülek találhatók.

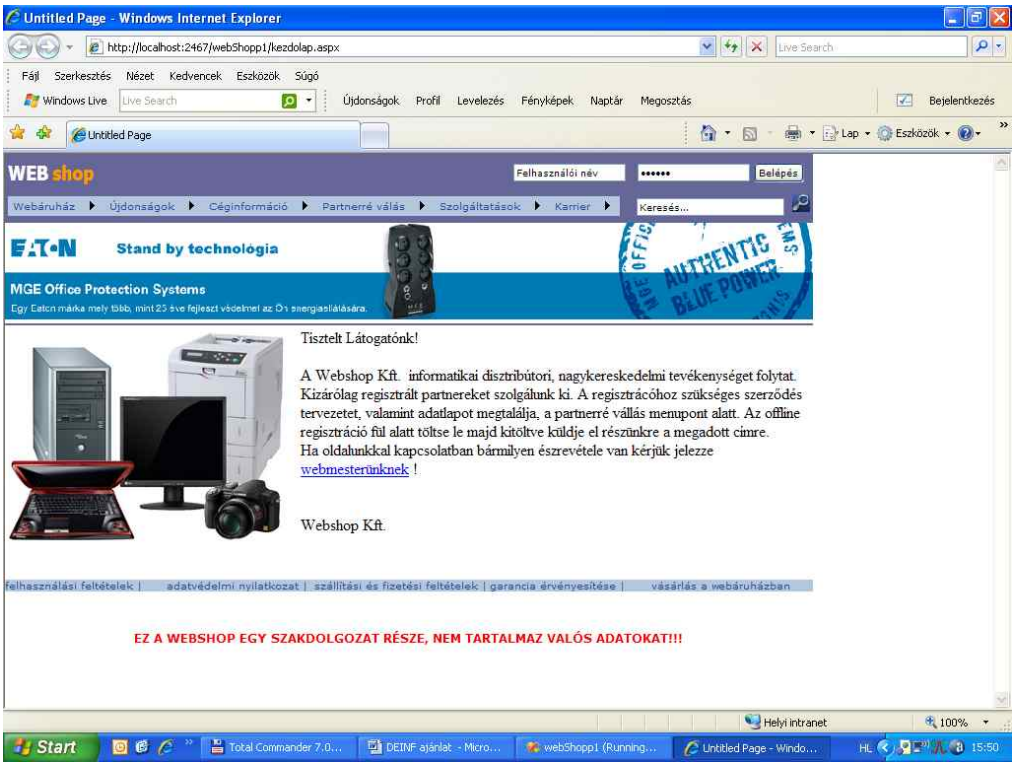
- Cégünk fül: elnavigál a ”cegunk.aspx” oldalra, ahol a lap információt ad a disztribútor történetéről, tevékenységéről, eredményeiről, működéséről, menedzsmentről stb.

- Elérhetőségünk fül: elnavigál a ”elerhetosegunk.aspx” oldalra ahol a lapon az elérhetőségünk, a nyitvatartásunk, a telefon, a fax és az e-mail cím látható.

- Gyártók fül: elnavigál a ”gyartok.aspx” oldalra, ahol azon gyártók ikonjai láthatóak, melyek termékeit a disztribútor forgalomba hozz.

Kezdőlap bemutatása

A honlapra történő navigáláskor a kezdőlapként beállított ”kezdolap.aspx” oldal töltődik be. Itt felhívjuk a felhasználó figyelmét, hogy a használathoz vagy be kell jelentkeznie vagy pedig regisztrálnia kell. Fontos megemlíteni, hogy a disztribútorok esetében nincs rá lehetőség, hogy online regisztráljanak, mint a hagyományos webshopoknál, kizárólag offline lehetséges. Az oldal felső részén elhelyezkedő menün a ”Partnerré válás” menüpont alatt elhelyezkedő offline regisztráció fülre kattintva töltődik be az ”offlineRegisztracio.aspx” oldal ahol a szükséges szerződés tervezetet, valamint a regisztrációs adatlapot le tudja tölteni. Ezt követően kitöltve vissza kell küldeni a webshop üzemeltetőjének, aki a szükséges bevizsgálás után (ha mindenben megfelelt) veszi fel partnernek, majd a bejelentkezéshez szükséges felhasználó nevet és jelszót levélben megküldi.



3.ábra kezdőlap

Regisztráció és bejelentkezés nélkül is böngészhet a felhasználó a webáruházban, de ebben az esetben csak a termékek és leírásaik jelennek meg. Megrendelést nem lehet generálni és az árakról sem kap információt.

A ”Default_Latogato” oldal felépítése

A ”Webáruház” menüpontra kattintva a böngészőben a ”Default_Latogato” oldal töltődik be, ahol a felhasználó betekintést nyerhet a disztribútor által forgalmazott termékekről. A négy megadott szűrő feltétellel illetve a kulcsszavas kereső segítségével szűkítheti a találati listát. Az árakról információt csak bejelentkezés esetén szolgáltat az oldal.

Természetesen, mint ahogy a többi oldalnál is az egységes megjelenítés érdekében a keretet a MasterPage.master oldal biztosítja. A *Panel*-re elhelyezett négy darab *ListBox* –ban jelennek meg az adatbázisunkban tárolt szűrőfeltételek így a gyártó, az elsődleges, a másodlagos és a harmadlagos termékjelleg.

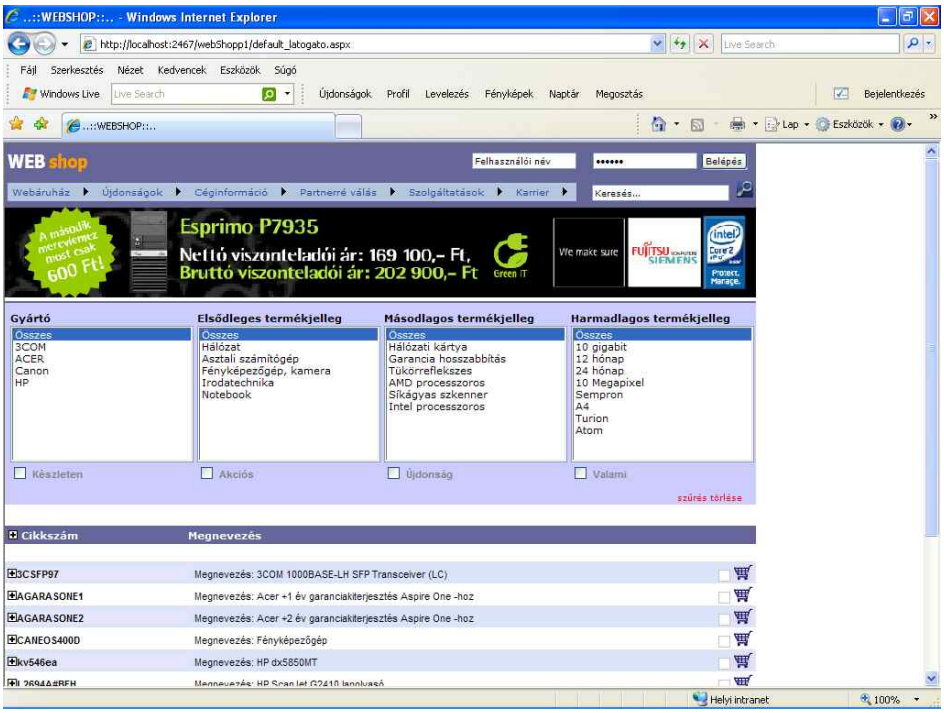
A *ListBox* -okat adatokkal futás közben töltjük fel dinamikusan, amit az alul látható metódus valósít meg. A metódus a *db* névtérben tárolt adatbázis lekérdező függvények segítségével lekérdezi a Kategória1, Kategória2, Kategória3, Kategória4, táblából a kategória neveket, majd egy FOR ciklussal a ListBoxok -hoz adja. A függvény, betöltődéskor az oldal Page_load metódusában kerül meghívásra.

```
protected void listBoxokFrissitese(string a, string b, string c, string d, string e)
{


aktualisA = a;
aktualisB = b;
aktualisC = c;
aktualisD = d;
aktualisE = e;

string katName;
lbGyarto.Items.Clear();
lbKat2.Items.Clear();
lbKat3.Items.Clear();
lbKat4.Items.Clear();

if (!lbGyarto.Items.Contains(new ListItem("Összes"))) { lbGyarto.Items.Add("Összes"); }
if (!lbKat2.Items.Contains(new ListItem("Összes"))) { lbKat2.Items.Add("Összes"); }
if (!lbKat3.Items.Contains(new ListItem("Összes"))) { lbKat3.Items.Add("Összes"); }
if (!lbKat4.Items.Contains(new ListItem("Összes")))
{
lbKat4.Items.Add("Összes");

System.Collections.Generic.List<db.kategoriak2> kategoriaTomb = new
System.Collections.Generic.List<db.kategoriak2>();

db.adabazisMuveletek.kategoriaLekerdezesek2(a, b, c, d, e, out kategoriaTomb);

for (int i = 0; i < kategoriaTomb.Count; i++)
{
        db.adabazisMuveletek.katIDToKatName(1, Convert.ToInt32(kategoriaTomb[i].kategoria1),
out katName);
       if (!lbGyarto.Items.Contains(new ListItem(katName))) { lbGyarto.Items.Add(katName); }

       db.adabazisMuveletek.katIDToKatName(2, Convert.ToInt32(kategoriaTomb[i].kategoria2),
out katName);
       if (!lbKat2.Items.Contains(new ListItem(katName))) { lbKat2.Items.Add(katName); }

       db.adabazisMuveletek.katIDToKatName(3, Convert.ToInt32(kategoriaTomb[i].kategoria3),
out katName);
       if (!lbKat3.Items.Contains(new ListItem(katName))) { lbKat3.Items.Add(katName); }
```

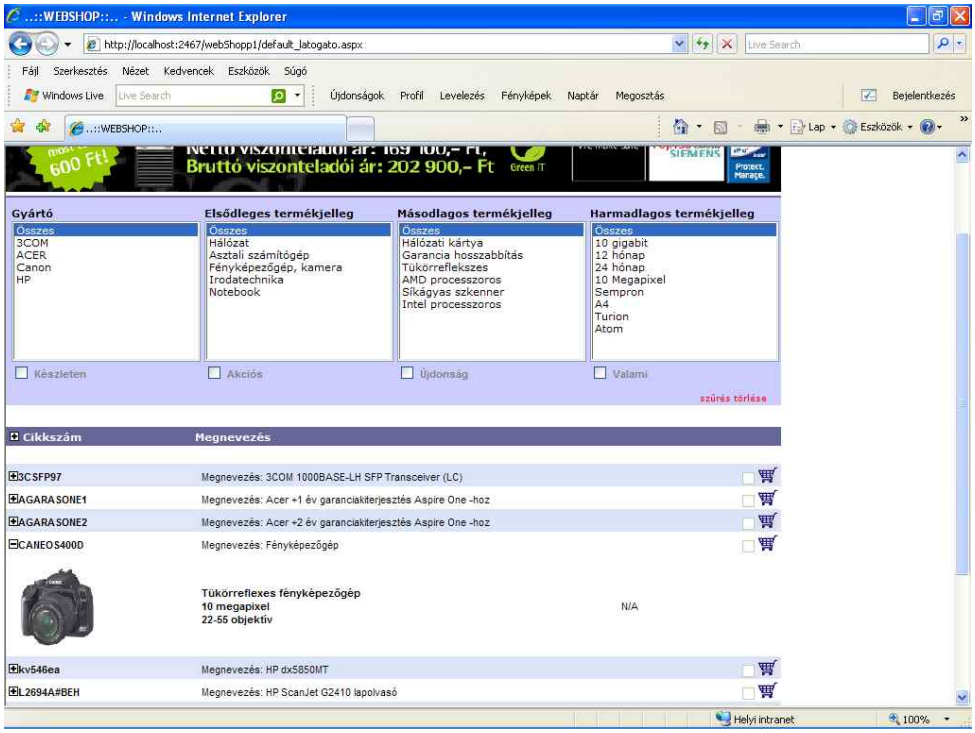
```
        db.adabazisMuveletek.katIDToKatName(4, Convert.ToInt32(kategoriaTomb[i].kategoria4),
out katName);
        if (!lbKat4.Items.Contains(new ListItem(katName))) { lbKat4.Items.Add(katName); }

}


lbGyarto.SelectedIndex=lbGyarto.Items.IndexOf(lbGyarto.Items.FindByValue(aktualisA));
lbKat2.SelectedIndex =lbKat2.Items.IndexOf(lbKat2.Items.FindByValue(aktualisB));
lbKat3.SelectedIndex =lbKat3.Items.IndexOf(lbKat3.Items.FindByValue(aktualisC));
lbKat4.SelectedIndex =lbKat4.Items.IndexOf(lbKat4.Items.FindByValue(aktualisD));

}
```

A *ListBox*-okat tartalmazó panel alatt egy másik panel került elhelyezésre, melyre a szűrési feltételeknek megfelelően kerülnek megjelenítésre az eszközök. A termékek adatait megjelenítő vezérlőket a termekListaFrissites() metódusban hozzuk létre FOR ciklus alkalmazásával. A metódus elsőnek lekérdezi az adatbázis *Eszkoz* táblájában tárolt adatokat és egy *termekLista* struktúrájú listába helyezi. A ciklus feltételének a lista elemszáma van megadva. Amíg a feltétel igaz addig a listából folyamatosan kinyerve a termékek adatait jelenítjük meg különböző webkontrolok használatával *(Label, ImageButton, CheckBox)* ezek egy részét táblázatba és egy *CollapsiblePanelExtender* –be tesszük.

A *CollapsiblePanelExtender-t* az AjaxControl Toolkit tartalmazza. A vezérlő két panelt kezel egyet a fejlécnek, a másikat pedig az adatok megjelenítésére. A fejlécre történő kattintás esetén az adatokat tartalmazó panel megjelenik. A megjelenítés kliens oldalon történik ezáltal nem terheljük sem a szervert sem pedig a hálózatot.



4. ábra default_latogato.aspx oldal



5. ábra CollapsiblePanelExtender használat közben

A ”gyártok.aspx” oldal működésének bemutatása

Az oldalra egy panel került, ami tartalmaz egy statikus táblázatot, valamint az alábbi HTML vezérlőt. A kontrol runat=”server” tulajdonsága lehetővé teszi számunkra, hogy az oldalt dinamikusan töltsük fel tartalommal, mely adatokat a már előzőkben prezentált SQL adatbázisból SELECT utasítások segítségével kapunk meg.

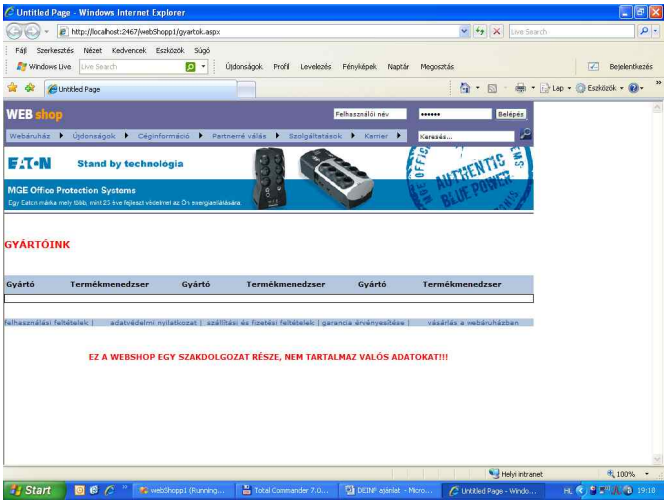
```
<div runat="server" id="div1"
        style="border: 1px solid #000000; font-family: verdana; font-size:
10px; width: 818px;">

</div>
```

Ehhez a *div* vezérlőhöz dinamikusan adjuk hozzá az adatokat a form kód részében.
A megvalósítás az alábbi kód segítségével történik:

```
protected void gyartoListaz()
{
div1.Controls.Clear(); // Töröljük a div1 ID-val ellátott div vezérlőt controljait
// Léterhuzunk egy a Class1.cs fileban a db namespace-ben definiált gyartoKat1ID
structurájú adatokat tartalmazó listát
System.Collections.Generic.List<db.gyartoKat1ID> gyartoLista = new
System.Collections.Generic.List<db.gyartoKat1ID>();

// Lekérdezzük az adatbázis Kategória1 táblájából az adatokat

db.adabazisMuveletek.gyartoKat1IdLekerdez(out gyartoLista);

// Léterhuzunk egy a Class1.cs fileban a db namespace-ben definiált disztriKontakt
strukturájú adatokat tartalmazó listát

db.disztriKontakt kontakt = new db.disztriKontakt();

// egy for ciklussal végig megyünk a gyartoListán

for (int i = 0, j=0; i < gyartoLista.Count; j++, i++)
{
// lekérdezzük azokat a termékmanagereket akik a megfelelő gyártóhoz vannak
rendelve

db.adabazisMuveletek.termekmanagerLekerdezes(gyartoLista[i].termekmanagerID, out
kontakt);

//dinamikusan létrehozzuk a Labeleket

Label nev = new Label();
nev.Text = kontakt.nev;
nev.Font.Bold = true;
nev.ID = "lblNev" + i.ToString();

...


// dinamikusan létrehozzuk a gyártói ikont tartalmazó képet
```

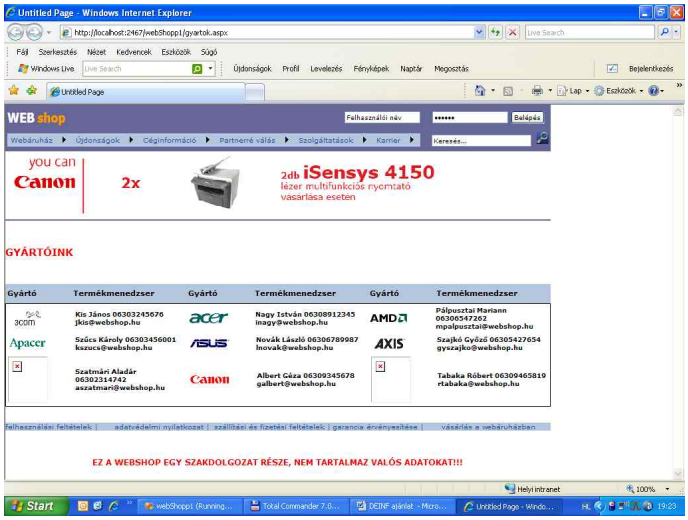
```
Image gyartoKep = new Image();
gyartoKep.ImageUrl = gyartoLista[i].kep;
gyartoKep.ID = "imgGyarto" + i.ToString();
gyartoKep.Width = System.Web.UI.WebControls.Unit.Pixel(60);

// létrehozzuk a táblázat celláit és hozzá adjuk a controlokat

if (j == 0)
{
tCell1 = new TableCell();
tCell2 = new TableCell();
tCell1.Width = System.Web.UI.WebControls.Unit.Pixel(100);
tCell2.Width = System.Web.UI.WebControls.Unit.Pixel(173);
tCell1.Controls.Add(gyartoKep);
tCell2.Controls.Add(nev);
tCell2.Controls.Add(tel);
tCell2.Controls.Add(email);
}

...


// létrehozzuk a táblázat sorát majd a sorokhoz hozzáadjuk a cellákat

TableRow tRow = new TableRow();

tRow.Controls.Add(tCell1);
// létrehozunk egy Táblázatot

Table table = new Table();
table.Width = System.Web.UI.WebControls.Unit.Pixel(820);
table.BorderStyle = BorderStyle.None;

// A táblázathoz hozzá adjuk a sort a panelhez a táblázatot a panelt pedig a div1
kontrolhoz
table.Controls.Add(tRow);
pnl.Controls.Add(table);
div1.Controls.Add(pnl);

j = -1;
}
}
```

Ha a függvényt ki kommentezzük akkor az alábbi oldal jelenik meg. Itt jól látható, hogy csak a statikus elemek kerültek megjelenítésre.



4. ábra gyartok.aspx lap dinamikus elemek nélkül

Ha a függvényt engedjük lefutni az oldal betöltődésekor, akkor már láthatóak a dinamikusan feltöltött elemek is.

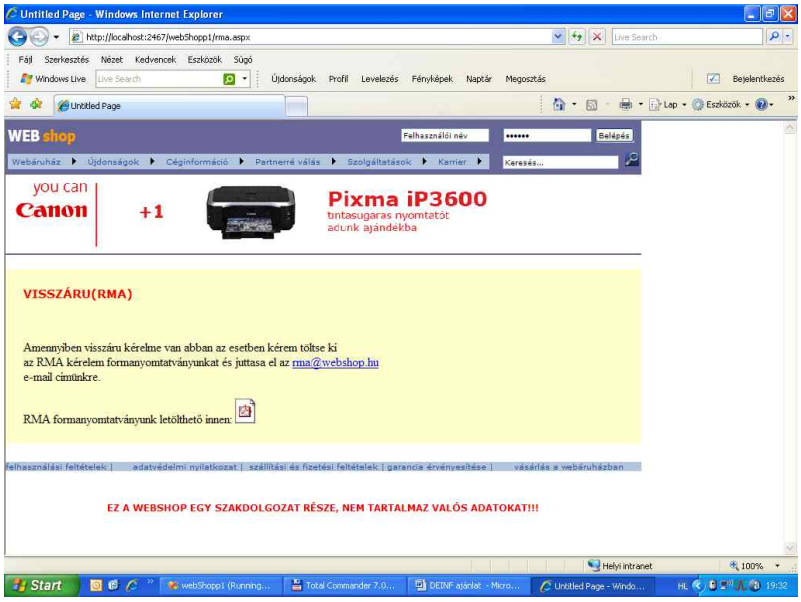


5. ábra gyartok.aspx lap dinamikusan létrejövő elemekkel

A ”Szolgáltatások” menü bemutatása:

- Garanciális ügyintézés fül: elnavigál a ”szerviz.aspx” lapra ahol egy leírás található a garanciális ügymenetről, valamint letölthető az árukísérő nyomtatvány a visszaküldéshez.

- Visszáru (RMA) fül: elnavigál a ”rma.aspx” lapra ahol egy leírás található a garanciális ügymenetről, valamint letölthető az RMA kérelmi nyomtatvány a visszaküldéshez.



6. ábra rma.aspx lap

Mindkét form tartalmaz egy *ImageButton* gombot. A gombra történő kattintás esetén, egy újablakban megnyílik a gomb click eseményéhez rendelt ”rma.pdf” file.

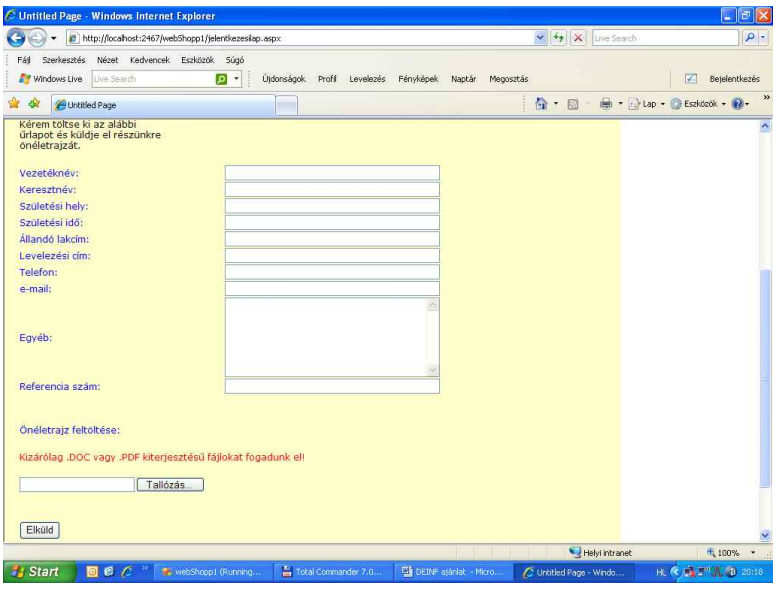
```
public partial class rma : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //Az ImageButton2-höz szerver oldalon hozzárendelt Onclick eseményét
        //felülírja, hogy ne történjen semmi azaz semmilyen postback
        //esemény ne hajtójon végre.

        ImageButton2.Attributes.Add("onclick", "return false");

        //A kliens oldalon egy klikk eseményt hoz létre és navigál az rma.pdf
        //-re ezt követően pedig az új ablakban megnyitja

        ImageButton2.OnClientClick = "window.open('szerz.pdf')";
      }
}
```

A ”Karrier” menüpont bemutatása

Általában véve a disztribútorok egy nagyobb létszámot foglalkoztató vállalatok (2-300 fő között) közé tartoznak, így ezen tulajdonságuknál fogva folyamatosan változnak a alkalmazottaik, felszabadulva betöltetlen pozíciók, ezért is tartottam szükségesnek az alábbi menü létrehozását.

- Álláshirdetések fül: elnavigál az ”allasAjanlatok.aspx” oldalra ahol az aktuális betöltetlen pozíciókat jelenítjük meg.

- Jelentkezési lap fül: elnavigál a ”jelentkezesilap.aspx” oldalra ahol egy webes űrlap található. Ezt az űrlapot kitöltve lehet jelentkezni az aktuálisan meghirdetett állásra. A *TextBox* beviteli mezők segítségével kérjük el a jelentkező adatait (Név, Cím, Születési idő, Telefon, Fax, Egyéb). A jelentkezőnek kötelező kitöltenie a mezőket. A kitöltés ellenőrzésére RequiredField validátort használok. Amennyiben az előbb említett mezőket kitölti és feltöltötte az önéletrajzát, akkor az elküld gombra történő kattintás hatására az adatok bekerülnek az adatbázis ”*Pályázók”* táblába, hogy a későbbiekben is a HR –es munkatárs segítségére legyenek.



7. ábra jelentkezesilap.aspx lap

Regisztrált felhasználói nézet

A fejezetben azokat a menüpontokat és oldalakat mutatom be, amik a látogatói nézetben nem kerülnek megjelenítésre.

Az előzőekben már említett regisztrációs folyamaton sikeresen átesett, regisztrált partnerek a masterpage jobb felső sarkában elhelyezett *Login kontrol* segítségével tudnak bejelentkezni az adminisztrátor által megküldött felhasználónév és jelszó együttes megadásával.
Ha a bejelentkezés sikeres volt abban az esetben a masterpage-en elhelyezett ID="Menu1" azonosítóval ellátott menüsor kerül megjelenítésre, amely eltér a látogatói nézet menüsorától. Ezt a MasterPage.Master.cs file-ban az előzőkben már említetésre került kód segítségével valósítottam meg.

Attól, hogy a menü láthatóságát letiltottuk a masterpage-en, még a böngészőbe beírva a form címét egy nem regisztrált felhasználó is el tudja érni a regisztrált felhasználókra vonatkozó oldalakat. A probléma elkerülése érdekében a látogatók számára tiltott oldalaknál az alábbi kód segítségével egy ilyen próbálkozás esetén, átirányítjuk a *"kezdolap.aspx"* oldalra. Ezt a kódrészletet az oldalak *Page_Load* metódusában helyeztem el így miden egyes oldal megnyitásakor elsőnek kiértékelődik a feltétel.

```
if (!Page.User.Identity.IsAuthenticated)
{

    Response.Redirect("kezdolap.aspx");

}
```

A regisztrált felhasználói nézet menüsorának felépítése:

- Webáruház
  - Részletes kosár
- Újdonságok
  - Kezdőoldal
- Céginformáció
  - Cégünk
  - Elérhetőség
  - Gyártók
- Partnerinfo
  - Áttekintés
  - Személyilap
  - Megrendelések
- Szolgáltatások
  - Garanciális ügyintézés
  - Visszáru (RMA)
- Karrier
  - Álláshirdetések
  - Jelentkezési lap

A ”Partnerré válás” menüpont helyett, a bejelentkezett felhasználók részére megjelennek a partner információk a ”Partnerinfo” menüpont alatt.

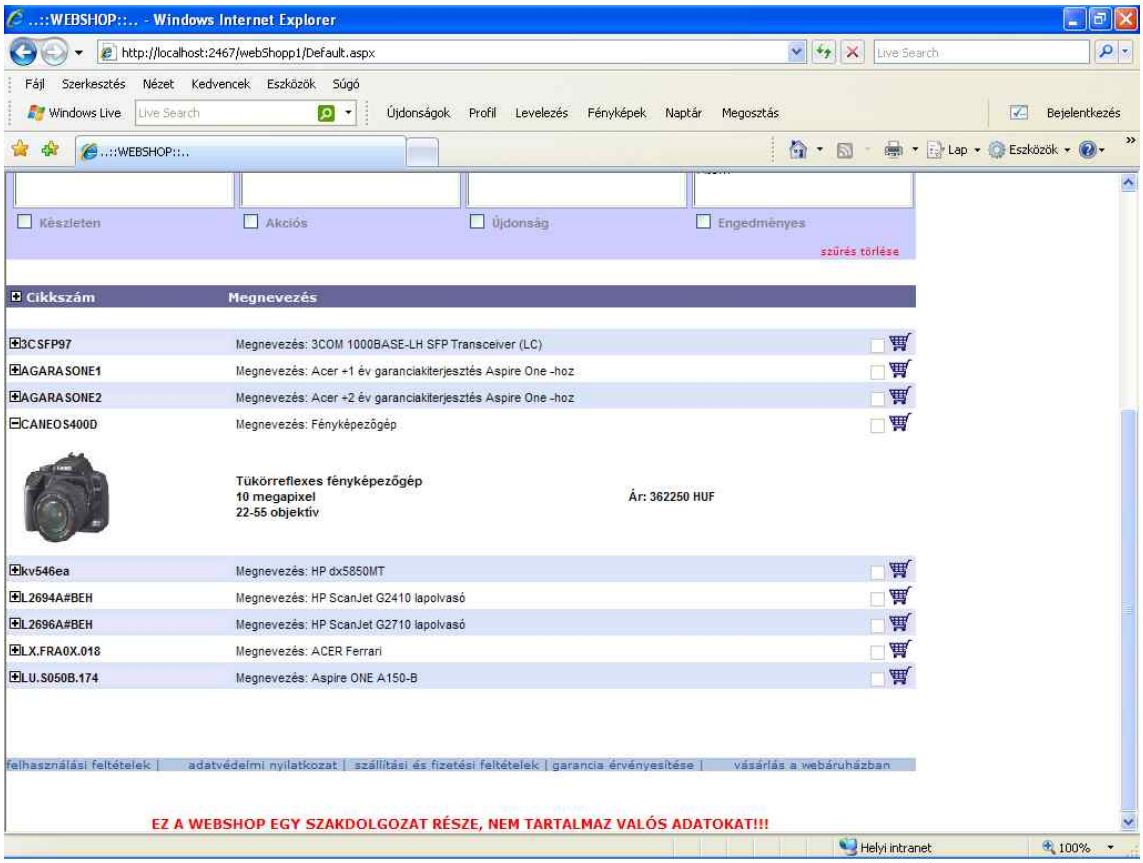
A megrendelés menete a ”Default.aspx” oldal bemutatása

A ”Webáruház” menüpontra kattintva, amennyiben bejelentkeztünk, a böngészőben a „Default.aspx” oldal töltődik be. Minden elemében megegyezik a ”default_latogato.aspx” oldallal, működését tekintve, viszont eltérnek egymástól.
A látogatói nézetben az oldalon feltüntetett árak nem jelennek, hanem egy *N/A* megjelöléssel hívtuk fel a felhasználó figyelmét, hogy az árak megtekintéséhez regisztráltatnia kell magát. Ha a regisztrációt elvégezte, akkor a masterpage jobb felső sorában elhelyezett *Login* vezérlő segítségével tud bejelentkezni .
Bejelentkezés után már a viszonteladó partnerre vonatkozó árakat tekintheti meg a felhasználó. Az árakat a következőképpen képezzük. A bevételezés során az eszközök bekerülési értékét tároljuk az adatbázis *Eszköz* táblájában az *ar* oszlopban. A viszonteladó partnereket a rendszerbe történő regisztráláskor egy árkategóriába soroljuk, ami az *Arsav* táblában került tárolásra, kezdéskor természetesen még a legalacsonyabba. A viszonteladó partner éves forgalmát figyelve ezen a besoroláson természetesen javítunk.
A termékek árai dinamikusan az oldal betöltésekor kerülnek meghatározásra. Az *Eszkoz* táblában szereplő árat beszorozzuk a partner *Arsav* azonosítójához tartozó *szorzó* –val.
A ”Default.aspx” oldal *Page_Load* metódusában elhelyezett adatbázis lekérdező függvények segítségével meghatározzuk a bejelentkezett felhasználó nevét (*UserName*), majd *UserName* alapján az adott cég azonosítóját ezt követően a *cegID* alapján az *arsavID* –t majd az *arsavID* alapján a szorzót.

```
//felhasználónév meghatárzása
string UserName = Page.User.Identity.Name;

//felhasználónév alapján a ID_CEG meghatározása
db.adabazisMuveletek.UserNameToCegID(UserName, out cegID);

//ID_CEG alapján a ID_ARSAV meghatárzása
db.adabazisMuveletek.cegIdToArsav(cegID, out arsavID);

//ID_ARSAV alapján a szorzo meghatározása
db.adabazisMuveletek.ArsavLekerdez(arsavID, out szorzo);
```

Majd a FOR ciklusban, ahol az eszközök megjelenítését végezzük a már korábban bemutatott ”default_latogato.aspx” file részletezésénél, a következő módon képezzük.

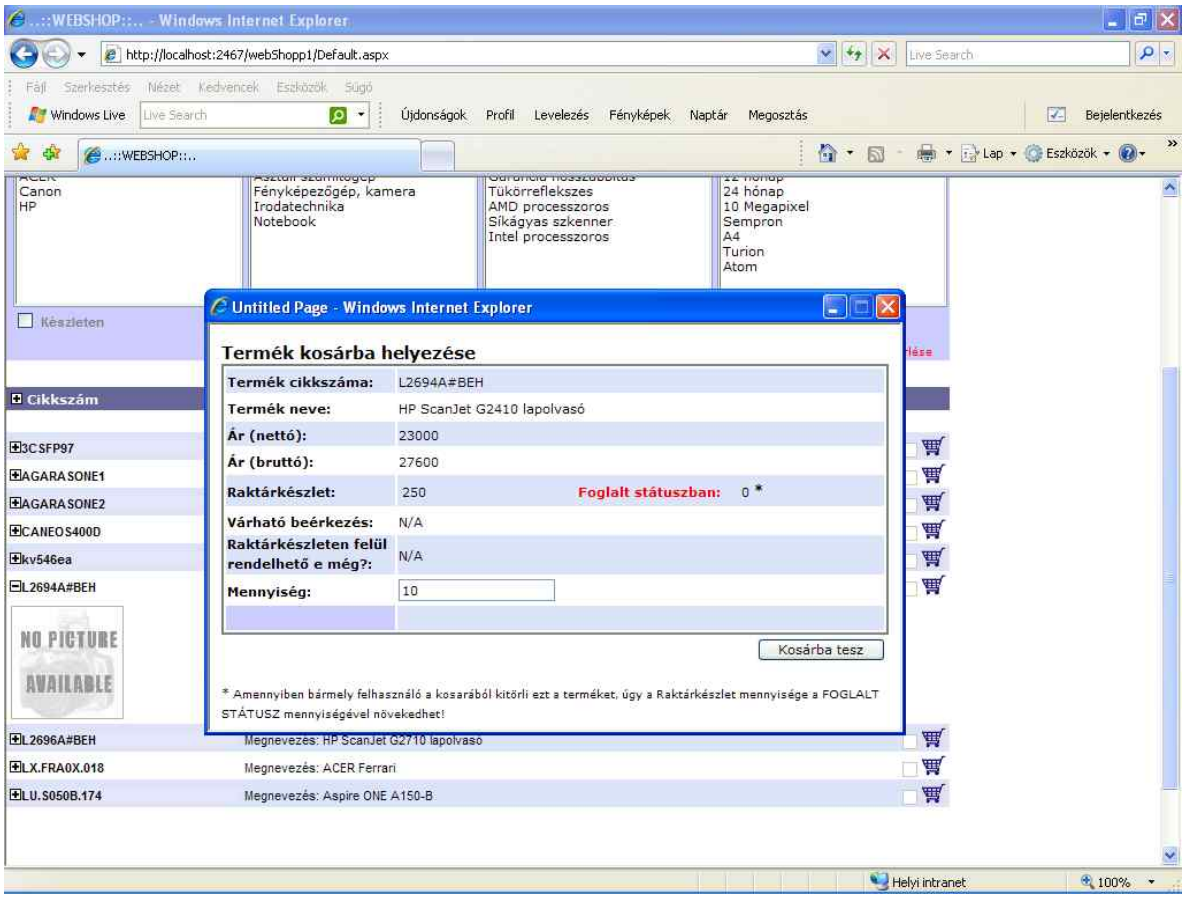
```
double kepzettAR = termekLista[j].ar * szorzo;
```



8. ábra Default.aspx lap

Ha termékből szeretnénk rendelni, akkor nincsen más dolgunk, mint rákattintani a termék mellett elhelyezett kosárra. A ”kosár” *ImageButton* kliens oldalon történő *onclick* eseményére felugró ablakként megjelenik, a *kosarbaTesz.aspx* oldal ahol a felhasználó megadhatja a megrendelni kívánt darabszámot, valamint információt kap az eszköz raktárkészletéről.

Ahhoz, hogy a kosárba kerüljön a termék a felhasználónak rá kell kattintania a kosárba tesz gombra ezt követően a rendszer egy megerősítést vár. Ha a partner megerősíti a kosárba tételt úgy az eszköz az adatbázisban definiált *kosarTartalma* táblába kerül.



9. ábra kosarbaTesz.aspx lap

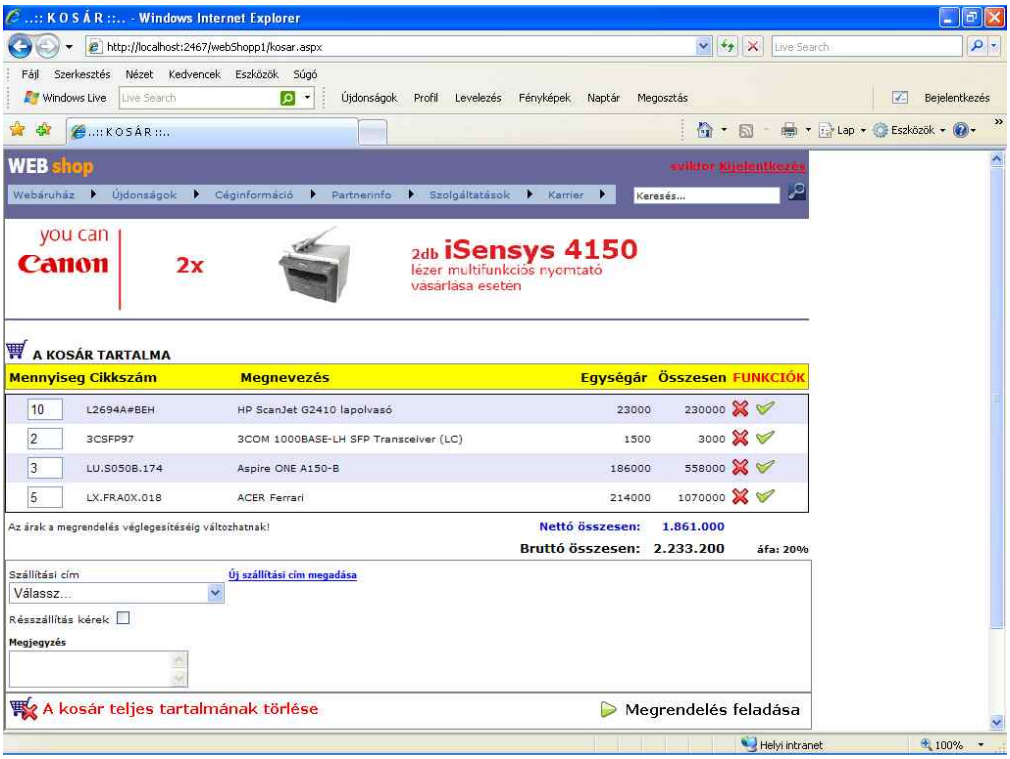
A kosárba tesz műveletet a felhasználónak annyiszor kell végre hajtania ahány különböző terméket szeretne vásárolni.

Ha az eszközök kiválogatásával végzett és minden olyan terméket a kosárba helyezett, amire szüksége van, akkor nincs más dolga, mint megrendelni. A megrendeléshez a *Webáruház* menüpont alatt megjelenő *Részleteskosár* fülre kattint. Ez esetben a böngésző a Kosar.aspx oldalra navigál. Itt még lehetőség van a termékek darabszámának módosítására illetve eltávolítására. Ha mindennel egyet értünk, akkor a legördülő listából kötelezően kiválasztva határozhatunk meg egy szállítási címet. Amennyiben a szállítási cím még nem szerepel az adott listában úgy a mellette megjelenő ”új szállítási cím megadása” *LinkButton* –ra történő kattintással hozhatunk létre új címet. Az esemény hatására a ”szallitasiadatok.aspx” oldal töltődik be ahol a form kitöltése és mentése után a legördülő listában már megjelenik az új szállítási cím, ezáltal kiválaszthatjuk.

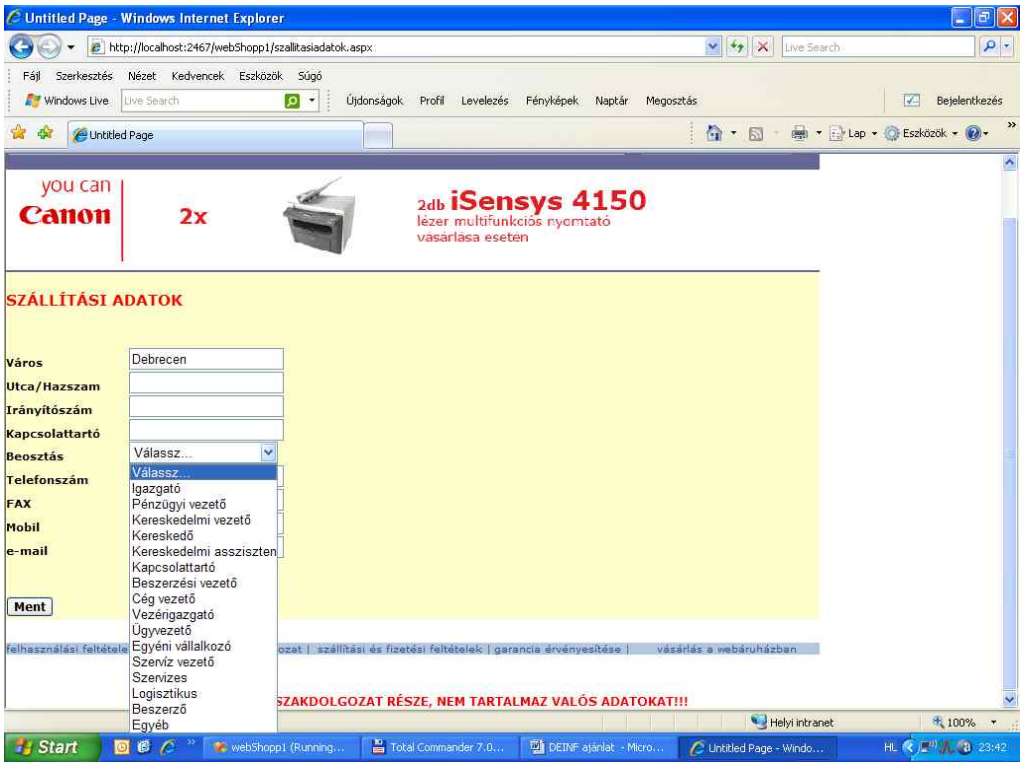
A megjegyzés rovatba, olyan információkat tüntethetünk fel, amiket mi szeretnénk, hogy megjelenjen a számlán. Ilyen például egy felhasználói megrendelésszám.

A felhasználói megrendelésszám központi logisztika esetén megkönnyíti az áru beazonosítását. Az átvevő személynek elég csak a feltüntetett hivatkozási számot a saját vállalatirányítási rendszerébe megkeresni.

Ha végeztünk az adatok megadásával, akkor a megrendelés gombra kattintva a *Kosar* táblából törlődnek az adatok és átkerülnek az adatbázis *Megrendelés* táblájába. Ebből a táblából már a vállalatirányítási rendszerünk ki tudja nyerni azokat az információkat, amik szükségesek az eszköz számlázásához és leszállításához.



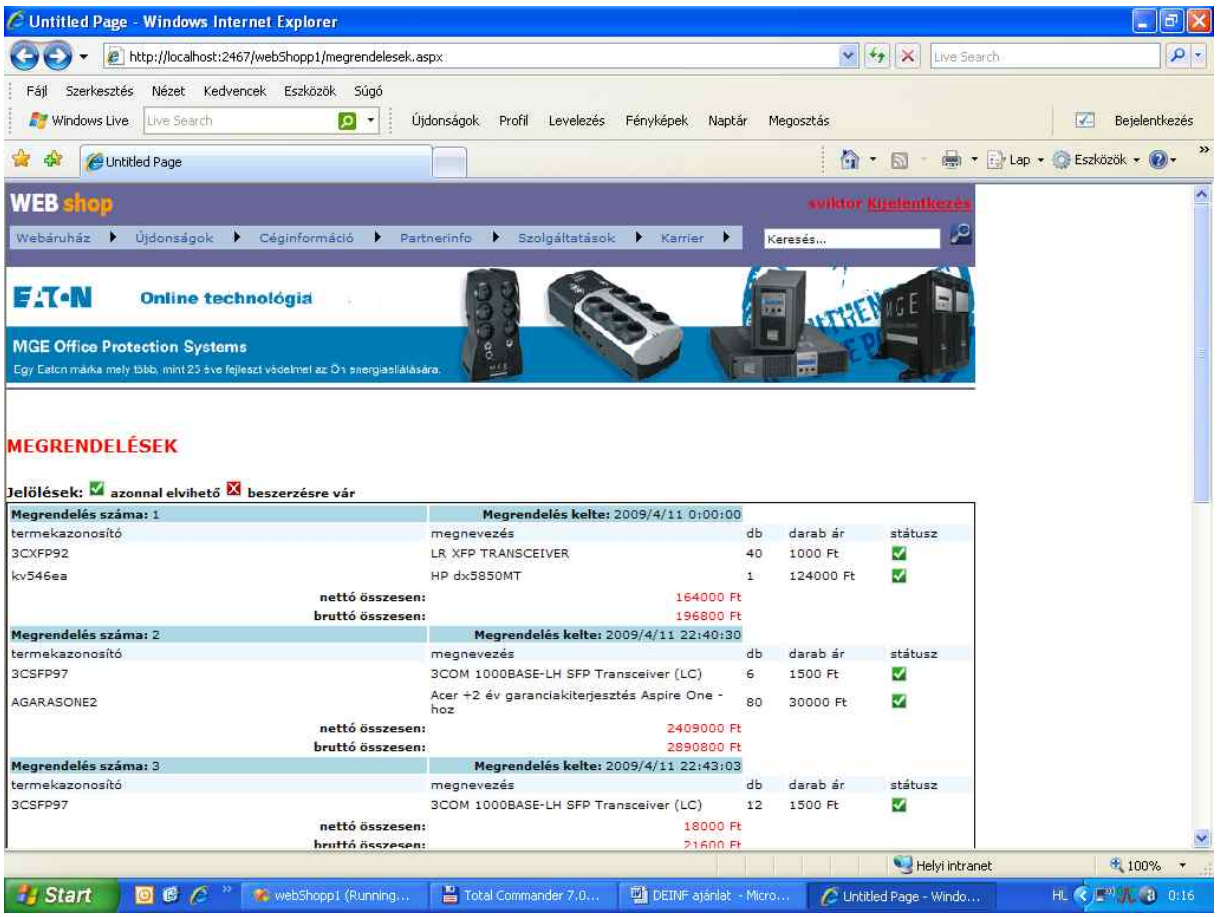
10. ábra kosar.aspx lap



11. ábra szallitasiadatok.aspx lap

Megrendelések nyomonkövetése a ”megrendelesek.aspx” oldal bemutatása

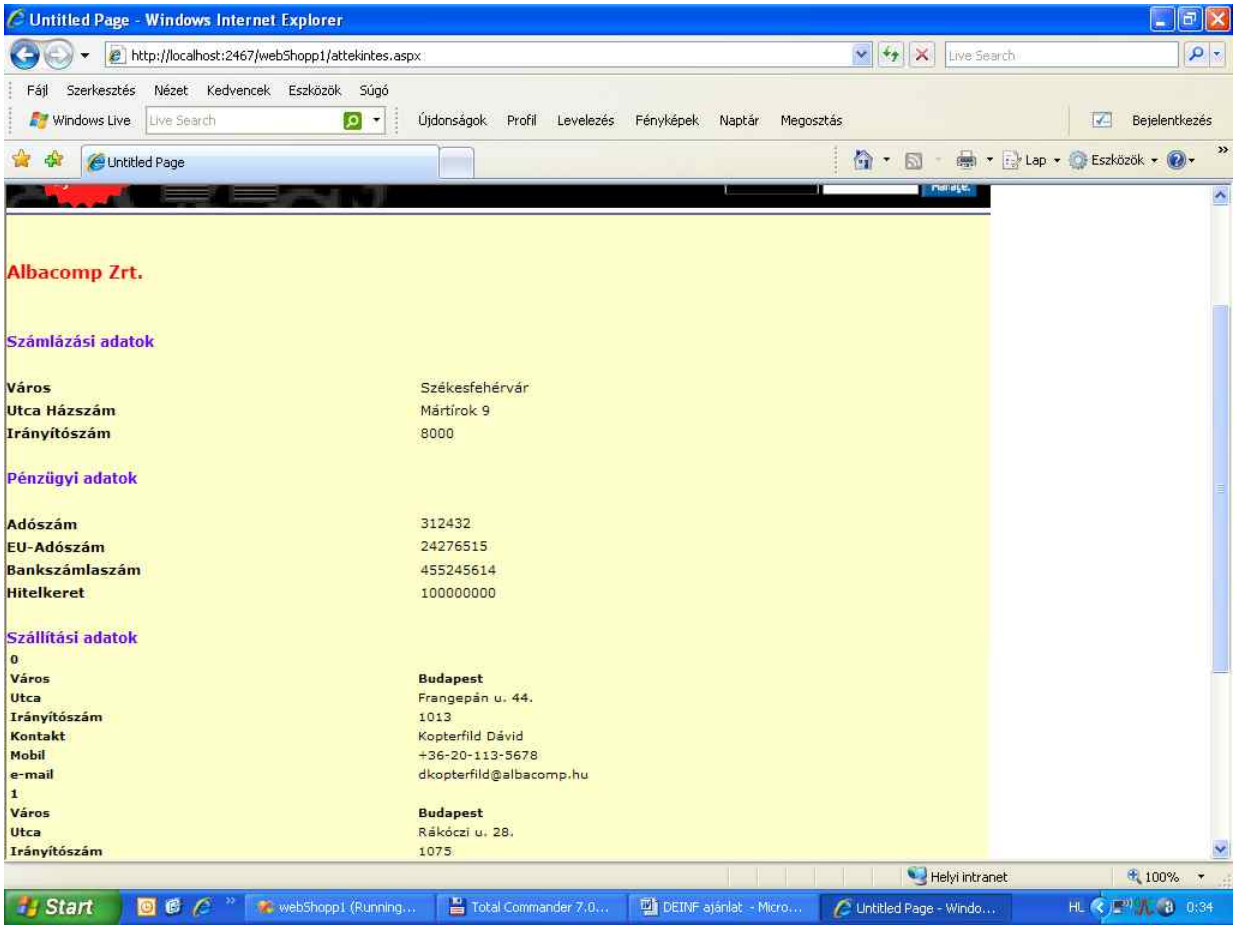
A Megrendeléseinket a ”Partnerinfo” menüpont alatt megjelenő almenü fülre kattintva tudjuk nyomonkövetni. A formon dinamikusan létrehozunk egy táblázatot az előzőekben már bemutatott módon és az adatbázis *Megrendelesek* táblájából kinyert adatokkal feltöltjük. Ha a megrendeléskor a megrendelt darabszám készleten volt úgy az azonnal elvihető státusszal jelöljük. Ez azt jelenti, hogy akár személyesen is egyből elhozhatjuk a disztribútortól, vagy szállítható a megrendeléskor megadott címre. Ha státusza beszerzésre vár abban az esetben a disztribútor vállalatirányítási rendszerében meg fog jelenni, mint beszerzendő eszköz. Ezt követően, ha az árut beszerezték és készletükre került el tudják indítani a szállítási folyamatokat (szállítólevél, számlakészítés, áru címzettnek történő feladása). A számlázást követően a megrendelések már nem jelennek meg.



12. ábra megrendelesek.aspx lap

Partner információk megjelenítése az ”attekintes.aspx” oldal bemutatása

Általános adatainkat a ”Partnerinfo” menüpont alatt megjelenő „Áttekintés” almenü fülre kattintva tudjuk leellenőrizni. Az adatokat az adatbázis *Ceg, Kontakt* és *Szallitasicim* táblákból kérjük le, majd dinamikusan feltöltjük vele a formunkat.
Megjelenítésre kerülnek a számlázási, pénzügyi és szállítási adatok a bejelentkezett partnerről.



13. ábra attekintes.aspx lap

Kapcsolattartók megjelenítése ”szemelyilap.aspx” oldal bemutatása

A ”Partnerinfo” menüpont alatt megjelenő ”Személyilap” fülre kattintva megjelenik a viszonteladó összes regisztrált partnere. A megjelenített adatokat az adatbázisunk *Kontakt* táblájából szelektáljuk az oldalra bejelentkezett felhasználóhoz köthető ID_CEG azonosító alapján. Az oldal használatával a felhasználók adataiban bekövetkezett változásokat módosíthatjuk, törölhetünk, illetve új kapcsolattartót is rögzíthetünk. A módosít *ImageButton* – ra kattintva az oldal átirányít a ”kontaktModosit.aspx” lapra az alábbi függvény segítségével.

```
protected void imgBtnModosit_Click(object sender, EventArgs e)
{

        ImageButton imgBtnModosit = (ImageButton)(sender);

        string[] kontakID = imgBtnModosit.ID.ToString().Split(new Char[] { '_' });
        Response.Redirect("kontaktModosit.aspx?kontaktmodosit="+kontakID[1]);

}
```

Mivel az *ImageButton* -okat is dinamikusan hoztuk létre, ezért saját magunk generálhatunk mindegyiknek azonosítót. A legcélszerűbb megoldás az volt, hogy egy kezdő sztringet megadva hozzáadjuk egy általunk meghatározott elválasztójel segítségével (jelen esetben ’_’ jellel) a gombhoz tartozó kontaktszemély azonosítóját (*ID_KONTAKT).* Ez a megoldás azt a célt szolgálja, hogy meg tudjuk határozni, éppen melyik felhasználót kell módosítanunk. A kezdő sztringet és az azonosítót az aláhúzás jel mentén feldaraboljuk így két sztring keletkezik, amiket egy tömbben tárolunk. A tömb első eleme lesz a kezdő sztring a második pedig a felhasználó azonosítója, amit már át tudunk adni a ”kontaktModosit.aspx” oldalnak.
Az átadott ID segítségével lekérdezzük adatbázisunkból a módosítani kívánt kapcsolattartót majd adatait TextBox -okban megjelenítjük. Ha módosítást elvégeztük, akkor a ”Módosít” gombra kattintva egy SQL UPDATE paranccsal a módosításokat visszavezetjük az adatbázisunkba.

Felhasználók regisztrálása

Új felhasználót adott céghez egy külön formon a „felhasznalo_rogzites.aspx” -en elhelyezett *CreateUserWizard* kontrol segítségével tudunk hozzáadni alkalmazásunkhoz. A vezérlő egy külön, nem általunk definiált adatbázisban rögzíti a felhasználói adatokat.

A felhasználókat tartalmazó adatbázist a *Login* kontrol elhelyezése után, egy hasznos segédeszköz a *Web Site Administration Tool (WSAT)* használatával hoztuk létre. WSAT-ot a Visual Studio Website menüjében megtalálható ASP.NET Configuration menüpontra kattintva indítjuk el. Miután rákattintottunk egy weboldal nyílik meg a böngészőnkben, amelyen különböző beállításokat tudunk eszközölni. Ilyen például, hogy az alkalmazásunk ne az alapértelmezett windows hitelesítést használja, hanem interneten keresztül az arra szolgáló oldalt. Esetünkben ez összes oldalra érvényes, mivel a login kontrolunkat azon helyeztük el.

Az ASP.NET-es alkalmazások esetében mappa alapú biztonság is beállítható. Ez azt jelenti, hogy a védett oldalainkat összegyűjtve egy mappába a hozzáférést korlátozhatjuk.

A *CreateUserWizard* vezérlőnél alapból csak a következő mezők találhatóak meg (UserName, Password, Confirm password, E-mail, Security Question, Security Answer) ez a 14.-es ábrán is jól látható. Ennyi információ részünkre, viszont nem elég egy felhasználó regisztrálásához. A lehetőség viszont adott, hogy testre szabjuk, bővítsük, átnevezzük vagy akár saját ízlésünknek megfelelően szinesebbé tegyük. A 15.-ös ábrán látható, amint a részünkre fontos adatok beviteléhez szükséges TextBox –okkal illetve DropDownList –el kibővítettük.

A *DropDownList*-hez dinamikusan adjuk hozzá a felhasználó cégének cégnevét, értéknek pedig a cég azonosítóját. A cég azonosítót a későbbiekben felhasználjuk, így egyszerűbb azt meghatározni a *DropDownList* egy elemének kiválasztásakor. Beosztás kiválasztásánál szintén ezt az eljárást alkalmazzuk.

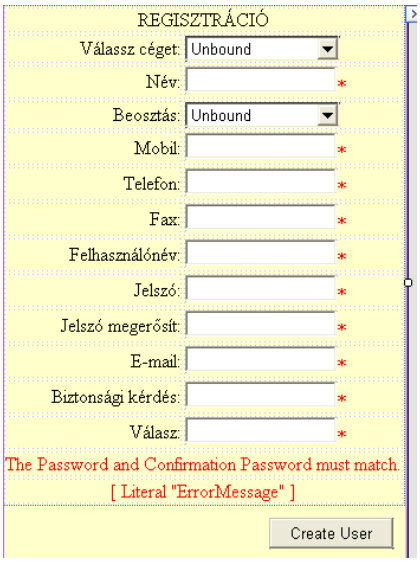
Ha kitöltöttük a mezőket akkor Create User gombra kattintva eltároljuk a felhasználókat, mind az általunk, mind pedig a Visual Studio által létrehozott adatbázisban. Ezzel rögzítettük is a felhasználót.

A regisztrált felhasználói nézet az alábbi két felhasználóinév és jelszó párossal tekinthető meg.

Felhasználóinév: sviktor **Jelszó:** Sviktor=

Felhasználóinév: Szoltan **Jelszó:** Szoltan=

14. ábra CUW alap mezők



15. ábra CUW kibővített mezőkkel

Összefoglalás

Az általam fejlesztett webalkalmazás segítségével egy rövid áttekintést kívántam adni az informatika és irodatechnikai eszközök disztribúciójával foglalkozó vállalatoknál használt webshopok működéséről.
A fejlesztéshez használt Microsoft Visual Studio 2008 fejlesztői környezet, eszközeinek és technológiáinak köszönhetően sikerült egy, olyan alkalmazást készíteni, aminek használatával gyorsabbá lehet tenni a kereskedelmi, a számlázási és a logisztikai folyamatokat.

Igazán jó és hasznos csak akkor lehet, hogyha a disztribútornál használt vállalatirányítási rendszerrel összehangolva, nagy részben automatizált folyamatok mellett képes működni. Sajnos ennek kipróbálására nem adódot lehetőségem.

További terveim között szerepel, hogy az alkalmazás felhasználói felületét egyéni ízlések esetén könnyen testreszabhatóvá tegyem, valamint kibővítem, olyan funkciókkal és megoldásokkal, melyek mind a felhasználók mind pedig az oldalt üzemeltető disztribútorok számára nagyobb hatékonyságot eredményez mindennapi munkájuk során.

Tapasztalva a webes technológiák folyamatos, nagyléptékű fejlődését valószínűsítem, hogy az informatika jövője is webes alkalmazások használatának irányába mutat.
Használata során nyílvánvalóvá vált számomra, hogy ezen alkalmazások fejlesztéséhez az egyik legalkalmasabb fejlesztői eszköz az ASP.NET technológiát használó Visual Studio.

Irodalomjegyzék

Papír alapú források:

Jesse Liberty, Dan Hurwitz and Brian MacDonald, ASP.NET 2.0 with AJAX, O'ReillyMedia Inc.

Internetes források:

http://devportal.hu/
http://msportal.hu/Default.aspx
http://www.asp.net/ajax/
http://hu.wikipedia.org/
http://www.asp.net/

Köszönetnyilvánítás
Ezúton mondok köszönetet témavezetőmnek Dr. Rutkovszky Edéné egyetemi tanársegédnek
a dolgozat elkészítésében nyújtott segítségéért