SZAKDOLGOZAT

Kerék Tamás és Tóth Zoltán

Programtervező informatikus

Debrecen

2011.

Debreceni Egyetem

Informatika Kar

WEBES ALKALMAZÁSFEJLESZTÉS SZABADON VÁLASZTOTT TÉMÁBAN

Témavezető:

Dr. Kuki Attila

egyetemi adjunktus

Készítette:

Kerék Tamás és Tóth Zoltán

programtervező informatikus

Debrecen

2011.

Tartalomjegyzék

1. BEVEZETÉS

A globális számítógépes hálózatok hálózata, azaz az Internet, mely az IP protokoll révén felhasználók milliárdjait kapcsolja össze, lehetővé teszi a WWW (World Wide Web) használatát. Az egyre bővülő technológiákat, szolgáltatásokat magánszemélyek és vállalatok is használják személyes és üzleti célokra egyaránt.

Szakdolgozatunk célja, hogy bemutassuk egy üzleti célra használható webes alkalmazás, építőipari web áruház, fejlesztési lépéseit, illetve ismertessük az alkalmazott technológiákat és eszközöket. Mivel egy ilyen alkalmazással létrehozása komplex feladat, a fejlesztésben is számos csoport van jelen. Ilyen csoportokat alkotnak maguk a szoftverfejlesztők, a grafikusok, a megrendelő valamint a felhasználók. A felhasználói és megrendelő csoport sokszínűségét tekintve, lehetetlen teljesen specifikálni az alkalmazás működését, struktúrája és funkciói az idő elteltével változnak, bővülnek. Bemutatjuk a fejlesztésünk során futtató környezetként szolgáló technológiákat, az Internet Information Services-t, a Microsoft SQL Server 2008-at, valamint a webáruház létrehozása során alkalmazott ASP.Net, CSS, LINQ To SQL és JavaScript technológiákat.

A mai igényeket kielégítő webes alkalmazások túlnyomó többsége mögött egy azonos minőségű, strukturált adatokból álló adatbázis található. Egy magánszemély vagy vállalat, a szöveges vagy multimédiás formában megjelenő információkat, webes alkalmazások segítségével, ilyen adatbázisokból nyeri. Az adatbázis jó megtervezése és karbantartása a webes alkalmazások üzemeltetése mellett az egyik legfontosabb feladat A fejlesztés kezdetén felmerült, hogy a teljes adatbázist mi hozzuk létre, de az alkalmazott eszközök által nyújtott lehetőségeket kihasználva, az adatbázisnak csak egy részét hoztuk létre. Kezdeti lépésként, amikor létrehoztunk egy ASP.Net Web Application projektet, készen kaptunk egy, a regisztrációhoz használt adatok letárolására alkalmas adatbázist különböző táblákkal. Bemutatjuk ezen táblákat, valamit ezen táblák bővítését, továbbá az általunk létrehozott táblák sajátosságait.

Több modell is támogatott a .NET által. Ezek közül a legkifinomultabb a Microsoft MVC modell, mely jelenleg a 3. verziónál tart. Mi mégsem ezt választottuk, mivel egy kezdő fejlesztő, vagy egy asztali alkalmazásokkal foglalkozó fejlesztő számára, ennek megértése jelentősebb erőbefektetést igényel. Választásunk az ASP.NET web forms megoldására esett. Ennek egyik előnye az MVC modellhez képest, hogy toolbox segítségével építhetjük fel a megjelenítendő elemeket. Ez tapasztaltabb fejlesztők számára idővel úgyis elfelejtődik, mivel gyakran egyszerűbb kód állítható elő saját magunk által begépelt kóddal. Másik nagy előnye, hogy a kódunk minden esetben típusos lesz. Tehát egy objektumnak mindig meg tudjuk határozni a típusát, metódusait. Harmadik nagy előnye, hogy állapotmegőrző, ebben rejlik viszont egy jelentős hibája is. Lassabbak a létrehozott alkalmazások egy MVC modellben írt alkalmazáshoz képest. Természetesen ez nem jelent komoly különbséget kis és közepes méretű alkalmazásoknál. Ezt ellensúlyozza a kód könnyű átláthatósága, fejlesztés sebessége és a könnyű átállás asztali alkalmazások fejlesztéséről. Az állapotmegőrzés nyilvánvaló előnyei mellett oda kell figyelnünk az oldal generálásának menetére.

Mivel az állapotunk megvan, könnyedén azt hihetnénk, hogy ha valaminek értéket adunk az oldalbetöltésekor, akkor azzal dolgozhatunk. Ez egy jelentős különbség az asztali alkalmazáshoz képest, mivel itt minden postback esemény esetén az oldalunk újratöltődik és a Page_Load esemény lefut. Az itt történt értékadás újra lefut, és ha ez egy már a későbbiekben megváltoztatott érték az újra erre az értékre áll vissza. Ennek egyszerű megoldása, hogy vizsgálni kell ezeknél az értékadásoknál, hogy postback esemény miatt töltődött e be az oldal. Ezt a IsPostBack érték tárolja számunkra.

Előnyöket és hátrányokat számításba véve ez tűnt a legjobb módszernek a webes alkalmazásfejlesztés elsajátításának az első lépéséhez. Régebbi szerverek is képesek a futtatására, és könnyen elsajátítható. Következő lépésként mindenképpen ajánljuk az MVC modellt, a lelkes .NET fejlesztők számára!

2. ALKALMAZOTT TECHNOLÓGIÁK ÉS ESZKÖZÖK

2.1 Az Internet Information Services (Kerék Tamás):

Az Internet Information Services, korábbi nevén az Internet Information Server, a Microsoft Windows platformon futó internet-alapú szolgáltatásokat összefogó termék. Használható statikus valamint dinamikus tartalom webes megjelenítésére egyaránt. 1996-ban kezdte meg máig is tartó pályafutását. Ma ez a második legnépszerűbb webkiszolgáló a világon, az Apache Web Server után. Ennek jelenlegi verziószáma az IIS 7.5, mely a Windows Server 2008 R2-ben valamint a Windows 7-ben található meg, de korábbi verziója már az NT4 alatti is megfigyelhető volt szervizcsomagok formájában.

Moduláris felépítésének és grafikus felügyeleti eszközeinek kiváló futtatási környezetet biztosít a web alkalmazások futtatásához, legyen az általunk használt Microsoft alapú .NET technológia, vagy PHP. Modularitása az IIS6 architektúrális jellegű újítása után alakult ki. A szervert darabokra, modulokra szedték. A 6-os verzió elődeihez képest azért volt nagy jelentőségű, mert alapban csak nagyon kevés szolgáltatás volt rajta engedélyezve. Miért legyen olyan szolgáltatás bekapcsolva, amit nem is használunk? Mi van, ha a nem használt szolgáltatásban van egy biztonsági rés? Ezen kérdések megválaszolása miatt volt az IIS6 jelentős verzió. A jelenlegi verzióban a nem használt szolgáltatások nem kikapcsolva vannak, hanem nincsenek feltelepítve, valamint DLL formában sem találhatóak meg. Ezzel kizárható, hogy egy hekker engedélyezze őket és visszaéljen velük. Ennek köszönhetően telepítéskor függőségek alakulnak ki, mivel vannak olyan komponensek, melyek működéséhez szükség egy másik komponens jelenléte is. Az IIS7 közel 40 darabra esett szét, ezeket a darabokat pedig telepítés során tetszőlegesen lehet ki-be kapcsolni.

A IIS telepítése alapértelmezés szerint nem történik meg a Windows telepítésekor. Ennek telepítése a Windows szolgáltatások be- és kikapcsolásánál tehető meg. Itt ki kell választanunk az **Internet Information Services** szolgáltatást, valamint az **Alkalmazásfejlesztés** szolgáltatáson belül a kíván dinamikus tartalomszolgáltatást is be kell jelölnünk. Ekkor a Webkezelési eszközökön belül az IIS-kezelő konzol is telepítésre kerül.

Segítségével az IIS adminisztrációjánál nem szükséges XML-alapú elosztott állományok kézzel történő szerkesztése, hanem használhatjuk a beépített GUI-t is.

2.2 A Microsoft SQL Server 2008 (Tóth Zoltán):

A Microsoft SQL Server 2008 R2 egy relációs adatbázis kezelő rendszer. 2010-ben jelent meg. A Microsoft SQL Server 2005 alapjaira épült. Megbízhatósága és skálázhatósága kiemelkedik az SQL szerverek közül. Rendelkezik saját report készítő eszközökkel, mely jelentősen megkönnyíti bármilyen report készítését. A legfőbb újdonságai között szerepel, hogy szabályokat (policy) hozhatunk létre, melyekkel egységesen juttathatunk érvényre beállításokat több SQL szerveren egyszerre. A Data Collector technológiával könnyedén eltárolja a teljesítmény és szerver aktivitás információt, hogy később ezeket elemezhessük. A tömörítésnek köszönhetően az adatok kevesebb helyet foglalnak, és a régebbi alkalmazásainkat sem kell ehhez átírni. Kiválasztható, hogy az adatbázis legyen-e tömörítve, és ettől függetlenül kiválasztható, hogy a backup legyen-e tömörítve. A Resource Governor technológiával biztosítja a fontos alkalmazásaink számára a mindig rendelkezésre álló erőforrásokat, így ezen alkalmazások soha nem lassulnak be. Az Enterprise változatban elérhető adatbázis szintű titkosítás, mely az adatok védelmét biztosítja, e mellett mégsem kell újraírni az SQL-re támaszkodó alkalmazásokat. Az új auditálás biztosítja, hogy mindig visszakereshető legyen, hogy ki változtatott jelszót, ki milyen adatot olvasott vagy törölt. A 64 bites Enterprise változat kiemelkedő funkciója, hogy leállítás nélkül, menet közben adhat az adatbázis adminisztrátor az SQL szervereimnek extra processzort és memóriát. Létezik ingyenesen elérhető változata. Ez a Microsoft SQL Server 2008 R2 Express. Egyetlen lényeges megkötése ezen verziónak, az egyes extra szolgáltatásokat leszámítva, hogy adatbázisonként maximum 10GB SQL tárhellyel rendelkezhetünk . Ezt a méretet a gyakorlatban átlagos felhasználással ritkán lépik át. Egyes nagyobb cégek, orvosi adatbázisok viszont akár nagyságrendekkel is nagyobb adatbázisokat használnak.

Rendelkezik saját fejlesztői környezettel, akárcsak az Oracle. Ez az SQL Server Management Studio, mellyel minden adatbázis műveletet elvégezhetünk. Egyszerűen optimalizálhatjuk, módosíthatjuk az adatbázisunkat, készíthetünk reportokat, mentéseket.

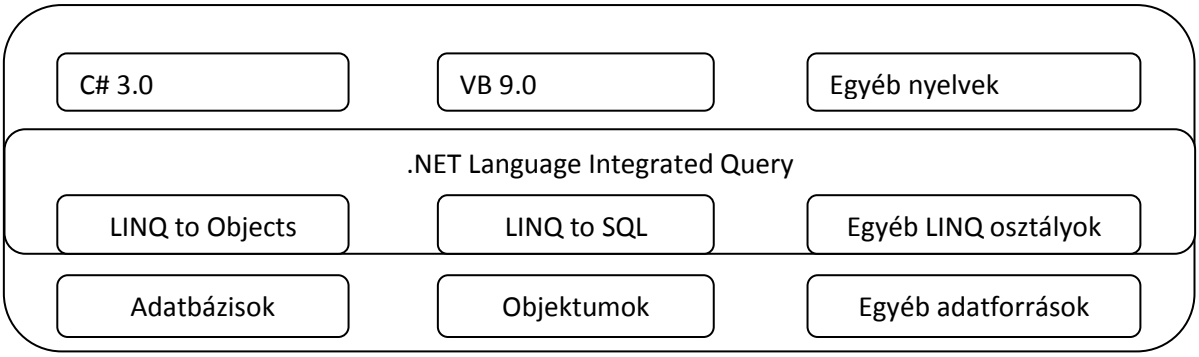
2.3 Language Integrated Query – LINQ (Tóth Zoltán):

A LINQ megjelenése előtt, bármilyen adatbázist használtunk, előtte meg kellett ismerni az adott adatbázis nyelvét, szintaktikáját. Ha MS SQL-t használtunk akkor meg kellett érteni a T-SQL nyelvet, Oracle estén annak saját SQL nyelvét. Ez előtt nem volt egységes nyelv és környezet arra, hogy ezeket egységesen kezeljük. Több megközelítés is létezik arra, hogy mi az a LINQ. A három legfőbb megközelítés: „A LINQ egy új adatabsztrakciós réteg." „A LINQ egy Tool amelynek segítségével SQL lekérdezéseket ágyazhatunk be a kódunkba." „A Linq egy egységes programozási modell, bármilyen adatforrásra nézve. A Linq lehetőséget biztosít, hogy egységes módon kérdezzünk le és módosítsunk adatok, függetlenül az adatforrástól." Ezen 3 meghatározás közül mind helyes, és együtt megfelelő képet kapunk a technológiáról.

Az első változata 2005 szeptemberben jelent meg. 2007-es megjelenése után biztossá vált a létjogosultsága. Az első időkben kevés adatforrást támogatott. Mivel bárki saját providert fejleszthetett, ezért ma már szinte bármilyen adatforrást kezelhetünk vele. Érdekességként megemlíteném, hogy létezik Linq To Facebook provider is.

Ahogy a neve is mutatja ez egy nyelvbe ágyazott lekérdezés. Segítségével nem kell körülményes SQL lekérdezéseket használnunk egy külön eszközzel ezt felparamétereznünk, elküldenünk a szerver felé és a kapott eredményt serializálnunk. A LINQ csak olyan adatforrásokon használható, melyek megvalósítják az Enumerable<T> vagy az IQueryable<T> interface-t. Ezeket könnyedén átalakítja a lekérdezés végén listákká, tömbökké.

2.4 LINQ TO SQL (Tóth Zoltán):



A LINQ egyik legelterjedtebb provider-e. Úgy biztosít számunkra futási idejű infrastruktúrát, a relációs adatobjektumok kezelésére, hogy nem veszítjük el a lekérdezés lehetőségét. Az alkalmazásunk szabadon módosíthatja az objektumokat, miközben a háttérben a Linq To Sql elvégzi a módosításokat. A .NET 3.5-től kezdve a keretrendszer része.

Ilyen adatbázis objektumokat könnyedén létrehozhatunk egy már meglévő adatbázisból. Ennek menete, először is hozzáadok a project-hez egy új LINQ to SQL Classes nevű objektumot. Ez után a tábláinkat a Visual Studio Server Explorer-éből áthúzom drag and drop módszerrel a Linq to SQL osztályra. Amennyiben a tábláink nem rendelkeznek kapcsolatokkal ez a toolbox-ból kiválasztott elemek segítségével pótoljuk. Az újonnan létrehozott objektumokat ezentúl akár a kapcsolatokon keresztül elérhetjük, join-ok megírása nélkül, úgy mint egyéb osztályok property-jeit. Ezt a következő módon tehetem meg:

```
ddlProdCat.SelectedValue = selectedProduct.aspnet_ProductSubCategory.aspnet_Produc
                           tCategory.ID.ToString();
```

Ebben a lekérdezésben 3 tábla összekapcsolása történt meg látszólag join nélkül, melynek végén megkapjuk a termék fő kategóriáját.

2.5 LINQ lekérdezések (Tóth Zoltán):

LINQ lekérdezések megírásakor nem kell foglalkoznunk a mögöttes adatbázisunk jellemzőivel. Ezzel együtt megjelent a var változó a .Net-ben. Ezt az implicit típusú lokális változó. Mivel nem minden lekérdezés estén tudjuk könnyedén meghatározni a visszatérési típust, ezért érdemes ilyenkor ezt használnunk. Ne vigyük túlzásba ennek használatát, mivel jelentősen csökkenti a kód olvashatóságát. Fontos megjegyezni, hogy mivel adatbázisból olvasunk, írunk, ezért lehet a visszatérési érték null. Ennek kezelésére, amennyiben változót töltünk fel Linq lekérdezéssel, a változó típusa után megadhatunk egy ? et mellyel jelöljük, hogy lehet null is az értéke. (Pl.: int? valtozo = null; egy valid deklaráció)

A lekérdezések leírása előtt példányosítanunk kell a Linq To SQL osztályunkat:

```
WebStoreDataClassesDataContext db = new WebStoreDataClassesDataContext();
```

Ez után elérhetővé tettük az adatbázisunkat a db változón keresztül.

`WebStoreDataClasses` A saját LinqToSQL osztályunkat jelöli. A `DataContext` rész pedig, hogy ez egy LINQ to SQL osztály.

2.5.1 LINQ – SELECT utasítás (Tóth Zoltán):

```
from [Aktuális rekord] in [db].[Táblanév] where [Feltétel] orderby [Aktuális sor].[Mező]
        descending select [Aktuális rekord].[Mező][, [Aktuális rekord][.Mező] …])
```

Amenyiben a teljes rekordra szükségünk van [Aktuális rekord]-ot kell megadni, a select után. Pl.:

```
var SelectedNews = (from nc in db.aspnet_News where nc.CategoryID == SelectedCategoryID
                    orderby nc.CreatedDate descending select nc);
```

Ugyanez az utasítás Lambda kifejezéssel:

```
var SelectedNews = (db.aspnet_News.Where(nc => nc.CategoryID ==
 SelectedCategoryID).OrderByDescending(nc => nc.CreatedDate));
```

2.5.2 LINQ - INSERT utasítás (Tóth Zoltán):

Adatbázis insert-hez létre kell hoznunk egy új record-ot. Ehhez használjuk a már meglévő Linq osztályunk megfelelő objektumát. Ezt a következőképen csinálhatjuk:

```
aspnet_New insered = new aspnet_New{ NewsName = txtNewsName.Text,
                                     ShortTextOfNews = EditorShort.Content,
                                     TextOfNews = EditorLong.Content,
                                     CategoryID = SelectedCategoryID,
                                     CreatedDate = DateTime.Now,
                                     LastModify = DateTime.Now};
```

Miután elkészítettük a megfelelő tábla record-ot, hozzá kell adnunk a táblánkhoz. Erre a következő utasítást használhatjuk:

```
db.aspnet_News.InsertOnSubmit(insered);
```

Ezzel elértük, hogy létrejöjjön az új rekord a LinqToSQL osztályunkban, de még nem jelent meg az adatbázisunkban. Az adatbázisba való mentést kérhetjük az Linqtól az alábbi módon:

```
db.SubmitChanges();
```

2.5.3 LINQ Update utasítás:

A update utasításhoz először is meg kell keresnünk a módosítandó rekordot. Ezt egy Linq select-el megtehetjük:

```
var modif = (from VAR in db.aspnet_News where VAR.ID == SelectedCategoryID)
                                    select VAR).Single();
```

Ez után egyszerűen módosítsuk az étékeket:

```
modif.LastModify = DateTime.Now;
modif.ShortTextOfNews = EditorShort.Content;
modif.TextOfNews = EditorLong.Content;
modif.CategoryID = Convert.ToInt64(ddlNewsCats.SelectedValue);
modif.NewsName = txtNewsName.Text;
```

Amint az összes módosítandó rekordot bejegyeztünk módosításra, akkor adjuk ki a parancsot a mentésre:

```
db.SubmitChanges();
```

2.5.4 LINQ Delete utasítás:

A delete utasításhoz először is meg kell keresnünk a törlendő rekordot. Ezt egy Linq select-el megtehetjük:

```
var modif = (from VAR in db.aspnet_News where VAR.ID == SelectedCategoryID select
                                    VAR).Single();
```

Ezután adjuk hozzá a tábla törlésre kijelönt rekordjai közé.

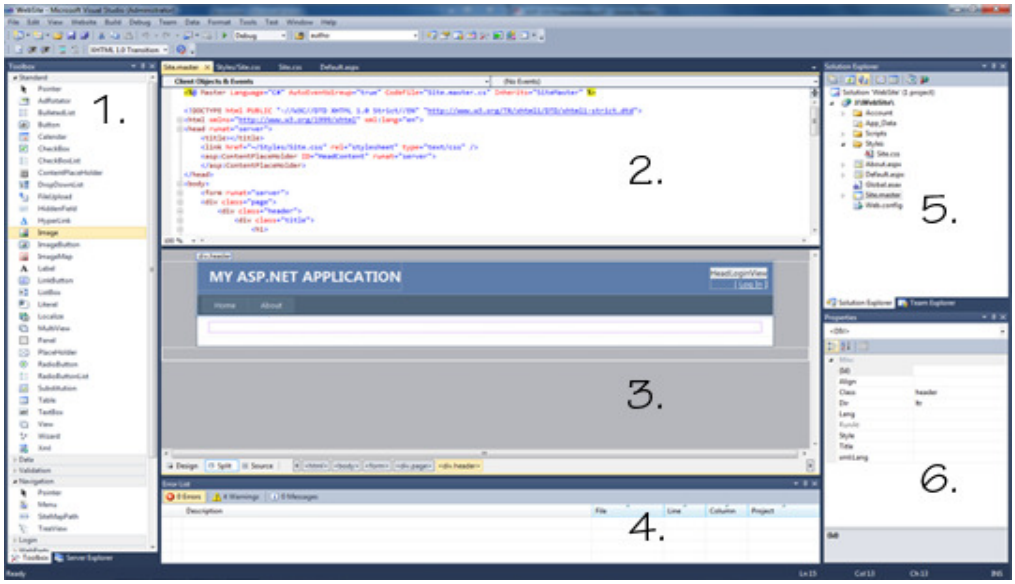
```
db.aspnet_News.DeleteOnSubmit(modif);
```

Amint az összes törlendő rekordot bejegyeztünk törlésre, akkor adjuk ki a parancsot a mentésre: `db.SubmitChanges();`

2.6 Az .NET 4.0 és a Visual Studio 2010 (Kerék Tamás):

A Visual Studio 2010 és a .NET 4.0 a .NET szoftverfejlesztés egy új, a mai trendeknek és technológiának megfelelő lehetőségét biztosítja. A Visual Studio 2010 és a .NET 4.0 sok újdonságot tartalmaz az előző verziókhoz képest. Ilyen újdonság a hatékonyság és kiterjeszthetőség területén bekövetkezett változás valamint a már meglévő technológiák használatának tökéletesebbé válása. A hatékonyság területén a Visual Studio 2010 Windows Presentation Foundation alapú IDE-jét vehetjük például, továbbá megfigyelhetjük, hogy a nyelvi fejlesztéseknek köszönhetően letisztultabb forráskódot kapunk. Az IDE már tartalmaz sok olyan hasznos szolgáltatást, amelyek az előző verziókban csak beépülő modulok segítségével voltak elérhetők. Az ilyen szolgáltatások közé tartozik a gyorskeresés, kódrészletek és osztálycsonkok létrehozása.

A meglévő technológiák, mint például az Asp.NET, már régóta léteznek és a 4.0-ás verzió megjelenése sem változtatott rajtuk nagymértékben. A Microsoft 2008-as bejelentését követően, a 4.0-ás verzióban megjelent a jQuery JavaScript könyvtár, amelyet webes alkalmazásfejlesztéskor szinte biztosan alkalmazunk. Ebben a kiadásban szorosabbá vált a Windows Workflow Foundation (WF) és a Windows Communication Foundation kapcsolata és a Windows Presentation Foundation is rengeteg újdonsággal egészült ki. Új, multitouchot is támogató API-t kapott és belekerült a Windows 7 tálca támogatása is.

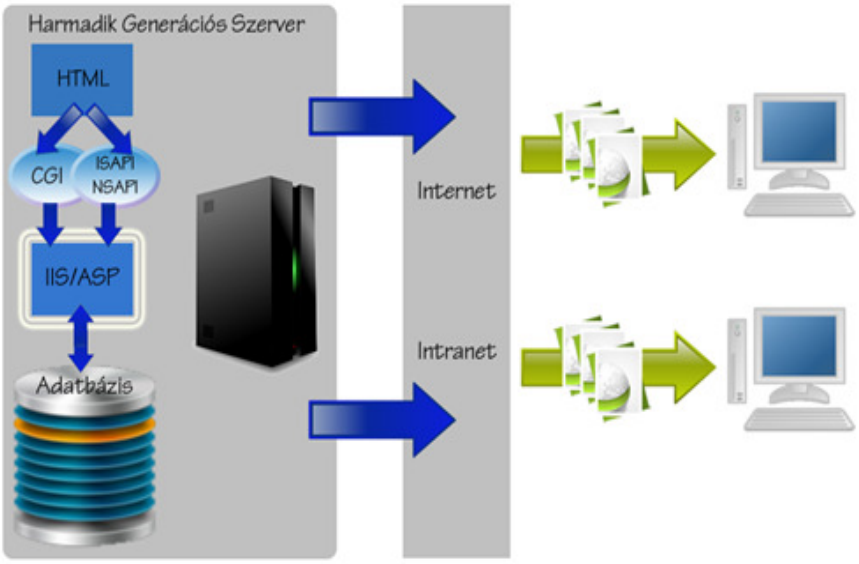


(A Visual Studio 2010 felülete)

A Visual Studio 2010 felületén az 1. ponttal jelöltük a Toolbox-ot. Itt találhatóak azok az eszközök, amelyeket használhatunk egy webes alkalmazás létrehozásakor. A 2. pont az aktuális lap forrását tartalmazza, ez a Source, míg a 3. pont a Design-t. Lehetőségünk van csak Source vagy csak Design nézetet alkalmazni. Ebben az esetben a 2. vagy a 3. ponttal jelölt terület hiányzik, a kettő közül csak az egyik van jelen. A jelenlegi nézet a Split, mely mindkét nézetet tartalmazza. A 4. ponttal jelölt területen értesítéseket, figyelmeztetéseket és üzeneteket kapunk a létrehozott kódról. A mi esetünkben a Warning fülnél gyakran több tíz figyelmeztetés is megjelent, habár ez csak a CSS3 alkalmazása miatt volt, mivel a Visual Studio csak a CSS 2.1-et ismerte, így nem validált kódként kezelte a CSS3 stílusdeklarációkat. Az 5. ponttal jelölt területen a Solution Explorer ablak található, melyben a készülő alkalmazáshoz tartozó állományokat, könyvtárakat, adatbázis objektumokat és egyéb az alkalmazás által használt állományokat találhatjuk. A 6. ponttal jelölt terület a Properties ablak, melyben az aktuálisan kijelölt control vagy HTML elem tulajdonságait állíthatjuk be, valamint itt tudjuk beállítani az események kezelését is. Természetesen nem kötelező minden tulajdonságot a Properties ablakban beállítani, ezt megtehetjük magában a kódban is, hiszen az Intellisense is ezt teszi, használva az automatikus kódkiegészítést.

2.7 Az Asp.NET (Kerék Tamás):

A web 1994-es feltalálása óta, a kezdeti statikus HTML oldalaktól egy hosszú fejlődés során eljutottunk a dinamikus és adatvezérelt webes alkalmazások világába. Habár a webes alkalmazások nem rendelkeznek a lokális alkalmazások minden tulajdonságával, a fejlesztéseknek köszönhetően a két alkalmazás-típus közötti különbség az idő előrehaladtával egyre kisebbé válik. Lényeges megemlíteni azonban hogy egy webalkalmazás esetén a felhasználók száma egy időben több, akár több száz is lehet. A webes alkalmazásfejlesztésre rendelkelő technológiák közül az egyik az ASP azaz a Microsoft Active Server Pages. Az ASP technológia lehetővé teszi a dinamikus és interaktív weblapokból álló webalkalmazások előállítását. A HTML nyelv értelmezése mellett lehetőséget biztosít úgynevezett ASP objektumok létrehozására, melyek szerveroldalon futnak le. Maga az ASP nem egy programozási nyelv, de felhasznál szkriptnyelveket, mint például a VBScript és a JScript. Az ASP alkalmazásakor a szerver oldali szkriptek nem függenek a kliens platformjától, mivel a szerver egy HTML kódot generál és azt bármelyik böngésző értelmezheti.



(Harmadik generációs szerver – Hatvany Béla Csaba: ASP 3.0 programozás)

Tehát egy ASP-ben készült webalkalmazás több weblapból (amik lehetnek HTML vagy ASP lapok), külső komponensekből (képek, hangállományok, videó állományok, stb.) és a hozzá tartozó speciális állományokból (az ADO és az adatbázisok) épül fel.

Egy ASP lap forrásának megtekintésekor azonnal szembetűnik, hogy az állomány két kód „együttese”. Ez, a már előzőekben ismertetett, HTML interpretáció és a felhasznált szkriptnyelv következménye. Tehát HTML kódot tartalmaz, valamint VBScript vagy JScript kódrészleteket. A VBScript vagy a JScript kódrészletek az állományon belül a <% és a %> elválasztók között szerepelnek, viszont az elválasztók között csak az egyik vagy a másik szkriptnyelv használható. Az ASP alapértelmezett szkriptnyelve a VBScript, tehát ha nem rendelkezünk másképpen, akkor az elválasztók között szereplő kódrészletet VBScript kódnak tekinti. Jelen esetben, a HTML kódban használt objektumok az <asp:OBJEKTUM> és a </asp:OBJEKTUM> határolók között jelennek meg. Tehát ha például egy ContentPlaceHolder-t használunk akkor az a következőképpen néz ki a kódban.

```
<asp:ContentPlaceHolder ID="HeadContent" runat="server"></asp:ContentPlaceHolder>
```

Jelen esetben az alkalmazott control a ContentPlaceHolder ami a kódban HeadContent ID-vel jelenik meg, a runat attribútum pedig a szerveren történő futtatást állítja be. Az ASP lap első sora tartalmazhat egy végrehajtási direktívát a <%@ és a %> elválasztók között. Ez több sajátos direktívából állhat, melyek a lap programozási nyelvére, a számok és a dátumok formázására valamint az egyéb elemekre vonatkozó végrehajtási módját határozzák meg. Itt határozhatjuk meg a lapon használt karakterkészletet valamint a lap programozási nyelvét is, ami esetünkben C#, de megadhatjuk az ASP lap MasterPage állományát is.

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="Site.master.cs"
                         Inherits="SiteMaster" %>
```

A fent látható kódban látszik, hogy az ASP lap programozási nyelve a C#, a CodeFile attribútumban pedig a használt Master Page állomány mögöttes kódját adtuk meg.

2.7.1 ASP.Net - Felhasználó azonosítás (Tóth Zoltán):

Programunkban két részre osztottuk a felhasználók kezelését. Az alkalmazásunk legtöbb oldala mindenki számára elérhető bejelentkezés nélkül. Más részek, mint például a megrendelés és az adminisztráció, bejelentkezéshez és felhasználói csoporthoz van kötve.

Az oldal felhasználói sikeres bejelentkezést követően érhetik el személyes információikat, esetlegesen csoportjukhoz tartozó funkciókat. Kezelésüket a legtöbb webes nyelven saját kezűleg kell implementálnunk, esetleg harmadik fél eszközeit kell használnunk. Ezzel ellentétben az ASP.Net beépített eszközöket nyújt számunkra. Ezek közül mi a webes környezetben leggyakrabban használt forms authentication elnevezésű eszközt használtuk. A felhasználók hitelesítését nem az alkalmazásunk hanem az IIS végzi. Mivel az oldal egyes részeit bárki, másokat bizonyos felhasználók láthatnak ezért meg kell tudnunk mondani az IIS számára, hogy melyik laphoz ki férhet hozzá. Erre a megfelelő eszköz a web.config elnevezésű file. Ebben tudjuk definiálni a hozzáférés szabályait. Az alkalmazás gyökér mappájában található a fő web.config. Ennek tulajdonságai öröklődnek az almappákra is, de tetszőlegesen felülbírálhatóak a megfelelő mappában.

```
<authorization>
   <allow users="*" />
   <deny users="?" />
 </authorization>
```

Ezzel a bejegyzéssel adhatjuk meg a mappa elérhetőségét. A * jelzi az IIS számára, hogy ez a szabály mindenkire érvényes, a ? pedig az azonosítatlan felhasználókat jelöli. Ezzel a bejegyzéssel elértük, hogy az oldalt mindenki láthatja kivétel az anonim felhasználók. A bejegyzések sorrendje nagyon fontos. Ha először a kivételt képző felhasználókat tiltjuk ki utána mindenkinek megadjuk a megtekintés jogát, azzal felülíródik az első feltétel, és bárki számára elérhetővé válik. Megadhatjuk ezeket a feltételeket felhasználói csoporthoz és oldalhoz kötve is. Ennek a módja a következő:

```
<configuration>
  <location path="Admin.aspx">
    <system.web>
      <authorization>
        <allow roles="Administrators"/>
        <allow roles="Sysadmin"/>
        <deny users="*"/>
      </authorization>
```

```
    </system.web>
  </location>
</configuration>
```

Itt a path kulccsal adjuk meg az oldalt, amire a szabály vonatkozik, a roles kulccsal pedig a felhasználói csoportot. Érdekességként itt az összes felhasználóra vonatkozó tiltás a végére került, de ez nem bírálja felül a role-hoz kötött engedélyeket.

2.7.2 Forms Authentication (Tóth Zoltán):

Az előző pontban tárgyalt beállításokkal megadtuk az oldalak elérhetőségeit, de semmit sem mondtunk az azonosítás módjáról. Ezt is a web.config állományban kell beállítani, a következő módon:

```
<authentication mode="Forms">
        <forms loginUrl="~/Account/Login.aspx" timeout="2880"/>
</authentication>
```

Itt a mode kulccsal adtuk meg, hogy az IIS forms authentikációt használja. Amennyiben egy oldal nem elérhető az adott felhasználó számára, a loginUrl kapcsolóban megadott oldalra irányítja a felhasználót. Ennek logikusan egy bejelentkező form-ot tartalmazó oldalnak kell lennie. A timeout kapcsolóban megadott érték az azonosítás élettartamát határozza meg. Az azonosításhoz szükséges információkat a felhasználó böngészőjében tárolt session-ben kerülnek tárolásra. A bejelentkezést a felhasználó azonosítását az IIS kezelni fogja, de meg kell mondanunk, hogy mely felhasználók bejelentkezését fogadjuk el. Ehhez szükségünk lesz két textboxra és egy button-ra. Az első textbox legyen a felhasználói név a második pedig a jelszó mező. A button eseményére pedig leellenőrizzük a megadott információkat.

```
private void Button1_Click(object sender, System.EventArgs e)
{
    if (txtUserName.Text == "user" && txtUserPass.Text == "pass")
    {
        FormsAuthentication.RedirectFromLoginPage(txtUserName.Text, false);
    }
    else
    {
        lblError.Visible = true;
    }
}
```

Itt leellenőriztük a jelszót és felhasználói nevet egy if elágazásban. Ha ez megfelel akkor „RedirectFromLoginPage”-ben engedélyezzük a belépést. Az első paraméter a Felhasználói név a második pedig, hogy tároljuk e le a cookie-ban. Ellenkező esetben megjelenítünk egy hibaüzenetet a felhasználó számára.

Egy másik, ennél még egyszerűbb megoldás ha a bejelentkezést a Visual Studio által generált bejelentkezést használjuk. Ennek nagy hátránya, hogy nehezen módosítható, mi mégis ezt a módszert választottuk egyszerű használata és kidolgozottsága miatt.

Egy webes alkalmazásnak a legtöbb esetben több felhasználója is lehet, és ezek száma folyamatosan nő. Ezért érdemes a felhasználókat és jogaikat is adatbázisban tárolni. Ezt automatikusan legeneráltathatjuk az oldalunkhoz. Ehhez csupán „Web Site Administration Tool” eszközt kell igénybe venni, a Solution Explorer menüjéből.

Ezután a megjelenő oldalon beállíthatjuk az oldalunkhoz kapcsolódó jogokat, alap felhasználókat, csoportokat. Amint ezt megcsináltuk megjelenik az alkalmazásunk App_Data mappájában egy új mdf kiterjesztésű adatbázis állomány. Ez tartalmazza az adatbázisunkat. Később, amikor adatbázis szervert szeretnénk használni, könnyedén exportálhatjuk az ebben létrehozott adatbázist. Létrejön a felhasználók kezeléséhez szükséges összes tábla. A bejelentkezés ezen lépés után máris használható.

2.8 A CSS (Kerék Tamás):

A CSS, azaz a Cascading Style Sheet a stíluslapok kifejezés rövidítése, amely egy szabványos, az oldalak formázásához használt leíró nyelv. A stíluslapok segítségével oldalainknak tetszőleges megjelenést adhatunk, úgy hogy a meglévő HTML oldalainkhoz egy .CSS állományt rendelünk stílusként. Az első webböngészők még maguk döntöttek az oldalak megjelenítésének miként jéről. A HTML fejlesztők a CSS megjelenését nagy örömmel fogadták, mivel a HTML-ből hiányzó lehetőségek pótlására alkalmasnak bizonyult. A World Wide Web Consortium (W3C) 1995-ös létrejöttével a CSS fejlesztése nagyobb ütemben haladt és még ebben az évben a Microsoft bejelentette hogy az Internet Explorer 3.0-ás verziója támogatni fogja. Ennek eredményeképpen, hogy ne maradjon le, a Netscape is alkalmazza a technológiát a Navigator 4.0-ás verziójában. A CSS első verziója 1996-ban jelent meg, ekkor vette fel a W3C az ajánlásai közé. A második verzió, azaz a CSS2 1998-ban jelent meg. Ez már támogatta a médiatípusokat, újabb méretezési és pozícionálási lehetőségeket vezetett be. A jelenlegi, harmadik verzió fejlesztése napjainkban is tart, várhatóan 2012-re ér el ajánlásra jelölt állapotát. Ennek a felépítése moduláris lett. Jelenleg az egyes modulok véglegeshez közeli vagy még kezdetleges állapotban vannak.

2.8.1 Formázás a CSS-sel (Kerék Tamás):

A CSS egyik legfontosabb tulajdonsága a folyamatos stíluslap – HTML lap kapcsolata. Egyik előnye, hogy a stílusbeli módosításokhoz nem kell minden oldalon az adott részt átalakítani, átformázni, hanem elegendő a stíluslapban végrehajtani a kívánt módosításokat. Természetesen a stíluslapokat nem csak összefüggően egy HTML lapra vonatkozóan használhatjuk, megadhatunk stílusokat egy-egy elemhez, bekezdéshez vagy táblázathoz is. A stílusmegadások két részből állnak, egy CSS kiválasztóból (ezt css szelektornak nevezzük) valami egy stílusdeklarációs részből. A css szelektor a formázandó HTML elem megnevezését tartalmazza (tehát minden HTML elem betöltheti a css szelektor szerepét), míg a stílusdeklaráció végzi el a szelektorban megadott elem formázását. A deklaráció is két részre bontható melynek egyik része a tulajdonság, a másik része pedig a tulajdonsághoz rendelt érték, például `background-color: #F0C84F;`. Ekkor egy HTML elem

háttérszínét (melyet megadhatunk hexadecimális értékként vagy a szín angol nevével is) adtuk meg.

Stíluslap megadása teljes oldalra vonatkozóan:

```
<link href="~/Styles/Site.css" rel="stylesheet" type="text/css" />
```

Az ASP lapon egy linket adtunk meg, amely egy stíluslapra hivatkozik.

Stíluslap megadása egy-egy elemre vonatkozóan:

```
<table style="width: 100%; border-radius: 5px 5px 5px 5px;">
```

Ebben a példában egy táblázatra vonatkozóan adtunk meg egy általunk használt stílust. A táblázatunk egy és egy tag között szerepelt. A `width: 100%;` stílusdeklaráció a táblázatunk méretét az őt tartalmazó elemhez igazította.

Ahhoz hogy csökkenthető legyen egy stíluslap mérete, megengedett a szelektorok vesszővel elválasztott csoportosítása, továbbá a deklarációk csoportosítása is.

```
h1, h2, h3, h4, h5, h6 { font-size: 1.5em; color: #F0C84F; margin-bottom: 0px; }
```

Ekkor a felsorolt HTML elemek azonos stílusbeli tulajdonságokkal fognak rendelkezni. A deklarációk csoportosítása esetén a

```
h1 { font-family: Arial; font-size: 14pt; font-style: normal; }
```

kódból a következő kódot kapjuk:

```
h1 { font: normal 14pt arial }.
```

A HTML elemek rugalmasabb stílusbeli formázása érdekében bevezetésre került az úgynevezett Class (osztály) attribútum. Az osztályba sorolt elemek az öröklődés szabályai vonatkoznak, öröklik a stílusértékeket a szülő elemeiktől. Az azonos osztályba tartozó elemek egyszerre is címezhetők. Például, a

```
.page { width: 960px; background-color: #F0C84F; }
```

kifejezés eredményeképpen a .page osztályba tartozó elemek azonos szélességi és háttérszín tulajdonságokkal fognak rendelkezni. A Class attribútum bevezetésével megjelent az ID attribútum is amely az egyedi azonosítók felvételét teszi lehetővé. Az ID attribútumot használhatjuk szelektorként a HTML elemek stílustulajdonságainak beállításához.

Fontos kiemelnünk, hogy a böngészők nem kompatibilisek egymással a stíluslapok használatát illetően. Egy böngésző akkor felel meg a CSS specifikáció követelményeinek, ha minden hivatkozott stíluslapot elér és értelmezi azokat, ha a deklarációkat rangsor szerint rendezi valamint a CSS funkcionalitását a megjelenítő eszköz korlátai közé tudja helyezni. Tehát egy böngésző akkor felel meg a stíluslap követelményeinek, ha a kimenete érvényes CSS stíluslap. Egyetlen böngésző sem tudja teljesíteni a CSS minden lehetséges funkcióját, azok csak megközelítik a stíluslap előírásait. A böngészők közötti különbség megfigyelhető az általunk létrehozott webes alkalmazás stíluslapjának használatakor. Az alkalmazásban elhelyezett táblázatok lekerekítésére a

```
border-radius: 15px 15px 15px 15px;
```

stílusdeklarációt alkalmaztuk. Ennek használatát a Mozilla Firefox böngészőjének 4.0-ás verziója támogatta, míg az Internet Explorer 9.0-ás verziója előtt is használható volt. Ugyan a Mozilla Firefox 3.6-os verziójában a kívánt eredmény szintén elérhető volt, de ehhez a stílusdeklarációt, egy a böngésző által értelmezhető `-moz-border-radius: 15px 15px 15px 15px;` formában kellett megadni.

Böngésző	**Firefox**		**Safari**		**Chrome**		**Internet Explorer**	
Renderelő motor	**Gecko**		**Webkit**		**Webkit**		**Trident**	
Verzió	**3.6**	**4**	**4**	**5**	**6-9**	**10**	**6-8**	**9**
Animációk (Animations)	nem	nem	igen	igen	igen	igen	nem	nem
Háttér Gradiensek (Background Gradients)	igen	igen	igen	igen	igen	igen	nem	nem
Háttér Méret (Background Size)	igen	igen	részben	igen	igen	igen	nem	igen
Határoló Kép (Border Image)	igen	igen	igen	igen	igen	igen	nem	nem
Határoló Sugár (Border Radius)	részben	részben	részben	igen	igen	igen	nem	igen
Doboz Árnyékolás (Boksz Shadow)	igen	igen	részben	igen	részben	igen	nem	igen
Hasábok (Columns)	igen	igen	igen	igen	igen	igen	nem	nem
Betűtipus Arculat (Font Face)	igen	igen	igen	igen	igen	igen	részben	igen
Többszörös Hátterek (Multiple Backgrounds)	igen	igen	igen	igen	igen	igen	nem	igen
Átlátszatlanság (Opacity)	igen	igen	igen	igen	igen	igen	nem	igen
Tükröződések (Reflections)	nem	nem	igen	igen	igen	igen	nem	nem
Szöveg Árnyék (Text Shadow)	igen	igen	igen	igen	igen	igen	nem	nem
CSS3 támogatás %-ban*	**73.5%**	**79.4%**	**88.2%**	**97.1%**	**94.1%**	**97.1%**	**2.9%**	**52.9%**

* A táblázat nem a CSS3 részletes leírását tükrözi, hanem a modern webdizájn által egyre fontosabbá vált vizuális jellemzők böngészőtámogatását.

(CSS3 támogatás az egyes böngészőkben)

2.9 A JavaScript (Kerék Tamás):

A JavaScript egy objektum alapú nyelv, melyet a Netscape fejlesztett ki eredetileg LiveScript néven. A JavaScript a Java valamint a LiveScript nyelvek ötvözéseképpen jelent meg, először a Netscape Navigator 2.0-ás verziójában. Maga a nyelv nem teljes mértékben objektum orientált, mivel alapvetően, a paradigmához köthető tulajdonságok hiányoznak belőle. Ilyen tulajdonság például az öröklődés. A JavaScript futási időben kerül értelmezésre, eltérően a Java appletektől. Folyamatában, az oldal letöltése után megjelenik az oldal tartalma és majd csak ez után kerülnek értelmezésre a script sorai. Tehát a scriptünk a HTML kódba ágyazva letöltésre kerül, ezért a script sorait bárki megtekintheti, aki letölti az oldalt. Mivel magát a scriptet a böngésző futtatja, a megírt scriptek platform függetlenek. Ez mindenképpen előnynek tekinthető. A hátrány viszont szintén a böngészőn történő futtatás jelenti, mivel az egymástól eltérő böngészők a szabványokat egymástól eltérő módon kezelik, így a script hibátlan futtatása is a böngészőtől függ.

2.9.1 A JavaScript használata a HTML dokumentumban (Kerék Tamás):

Egy HTML, vagy esetünkben egy ASP formátumú oldal, tartalmazhat JavaScript nyelven megírt kódrészleteket is a <script type="text/javascript"> és a </script> tag-ek közé zárva. Az ezen nyelven megírt függvényeket a dokumentum fejlécében definiáljuk a <HEAD> és </HEAD> tag-ek között. Az ilyen formában definiált függvények és változók, a dokumentum HTML elemeiben hivatkozhatók.
Fejlesztésünk korai szakaszában, scriptünket mi is a dokumentum fejlécében tároltuk, később pedig külön fájlba szerveztük az általunk használt függvényeket a Scripts könyvtárba. Ekkor a fejlécben csak a script forrását jelöltük meg, később pedig egyszerűen meghívhattuk az általunk megírt függvényeket. A módosítás a következőképpen nézett ki:

```
<script language="javascript" src="Scripts/opacityfnc.js"></script>.
```

A módosítás mindössze annyi volt, hogy az src attribútumban megadtuk a függvényeket tartalmazó .js kiterjesztésű fájl nevét.

Mivel több féle scriptnyelv is létezik, egy ASP dokumentumon belül többet is használhatunk. Erre szolgál a <script> tag language attribútuma, melyben a használt scriptnyelvet kell feltűntetni. Ha figyelembe vesszük, hogy egy ASP dokumentum életében is gyakoriak az események, akkor ezek bekövetkeztére meghívhatunk valamilyen scriptnyelven megírt eljárásokat vagy függvényeket, de ugyanakkor a mögöttes kódban C# nyelven megírt függvényeket vagy eljárásokat is meghívhatjuk. Kódunkban az opacityfnc(); függvényt JavaScript nyelven írtuk, amelyet minden oldal load eseményére meghívtunk a következőképpen:

```
<body onload="opacityfnc();">
```

Tehát a Body HTML elem betöltésekor lefut az általunk megírt script. A bekövetkezett esemény a load esemény, mely az oldal betöltésekor következik be, míg az eseménykezelő az onload. Az opacityfnc(); függvény egy -ben elhelyezett, két egymásra helyezett hátterének átlátszóságát állítja át folyamatosan, azt a hatást keltve, mintha egy Flash tartalom vagy animáció lenne.

2.9.2 Példa JavaScript függvényre (Kerék Tamás):

A következő bemutatásra kerülő példát, oldalunk fejlesztése során készítettük. Arra törekedtünk, hogy Flash tartalom és animált Gif elhelyezése nélkül animáció hatását keltsük. Ehhez az előzőekben leírt megoldásmódot választottuk.

```
function opacityfnc() {
    var uj;
    var uj2;

    if (document.all) {
        if (document.getElementById('header_1').filters.alpha.opacity > 0) {
            uj = parseFloat(document.getElementById('header_1').filters.alpha.opacity) - 1;
        }
        if (document.getElementById('header_2').filters.alpha.opacity < 100) {
            uj2 = parseFloat(document.getElementById('header_2').filters.alpha.opacity) + 1;
        }

        document.getElementById('header_1').filters.alpha.opacity = uj;
        document.getElementById('header_2').filters.alpha.opacity = uj2;

        if (uj == 0 && uj2 == 100) {
            opacityfnc2();
        }
        else {
            setTimeout("opacityfnc();", 40);
        }
    }

    else {

        if (document.getElementById('header_1').style.opacity > 0) {
            uj = parseFloat(document.getElementById('header_1').style.opacity) - 0.01;
        }
        if (document.getElementById('header_2').style.opacity < 1) {
            uj2 = parseFloat(document.getElementById('header_2').style.opacity) + 0.01;
        }

        document.getElementById('header_1').style.opacity = uj;
        document.getElementById('header_2').style.opacity = uj2;

        if (uj < 0.01 && uj2 > 0.09) {
            opacityfnc2();
        }
        else {
            setTimeout("opacityfnc();", 40);
        }
    }
}
```

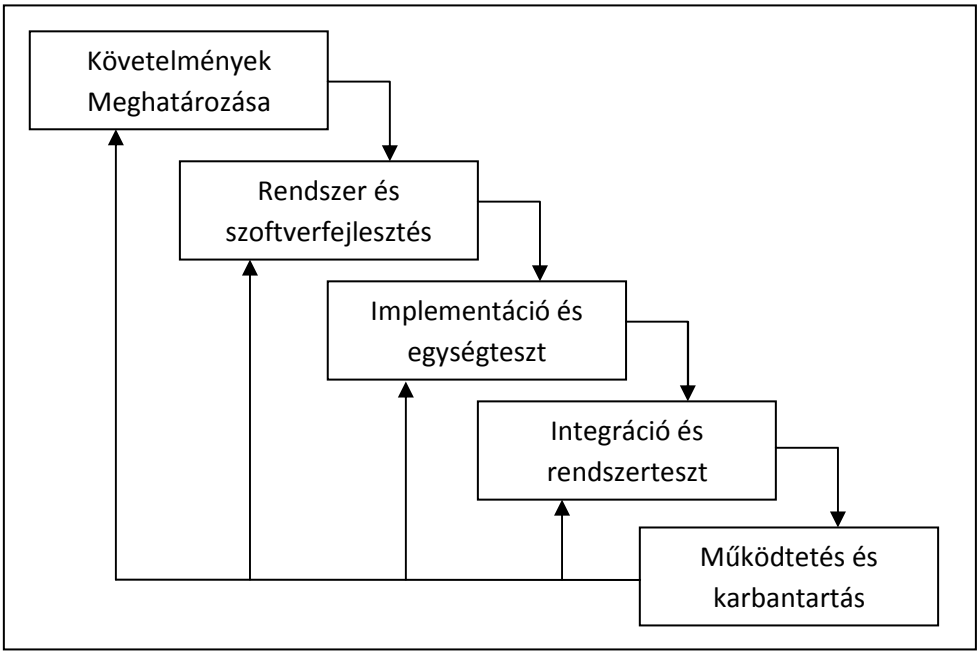
A példában megadott függvény a header_1 valamint a header_2 ID-vel rendelkező dokumentumbeli elem (ami jelen esetben két .png kiterjesztésű kép fájl) opacity tulajdonságát állítja át. Ha a header_1 opacity tulajdonsága nagyobb, mint 0 akkor az uj nevű változóhoz az 1-gyel csökkentett értéket rendeljük, míg a header_2-nél ha az opacity érték nem 100, azaz ha nem teljesen átlátszó a kép, akkor az uj2 nevű változóhoz az 1-gyel növelt értéket rendeljük. Nincsen szükség kezdeti lépésként az egyik képfájl teljesen átlátszóra állítására, mivel a képek egymáson helyezkednek el és az egyik teljesen takarja a másikat. Az elágazó utasítások kiértékelése után az uj és uj2 változókhoz a kívánt értékek rendelődnek, melyeket a header_1

és header_2 ID-vel rendelkező elemek új opacity értékeként állítunk be. Következő lépésként, ha az uj nevű változó a 0, míg az uj2 nevű változó a 100-as értékeket tartalmazza, akkor az opacity értékek átállításra kerültek és meghívásra kerül az opacityfnc2(); függvény mely az előző folyamatot visszafelé végzi el. Értelemszerűen az opacityfnc2(); függvényben, ha az opacity tulajdonságokat beállítottuk, az opacityfnc(); függvény kerül meghívásra, így elérve egy folyamatos animációnak tűnő hatást.

3. FEJLESZTÉSI FOLYAMATMODELLEK (TÓTH ZOLTÁN):

3.1 Vízesésmodell (Tóth Zoltán):

70-es években alakul ki, egy szekvenciális modell. Fázisokat szigorúan egymás után kell végrehajtani, és minden modell egy dokumentummal zárul, ami dönt arról, hogy érdemes e folytatni.



Használható: Ha a követelmények teljesen világosak, ez csak kisméretű szoftvereknél fordul elő.

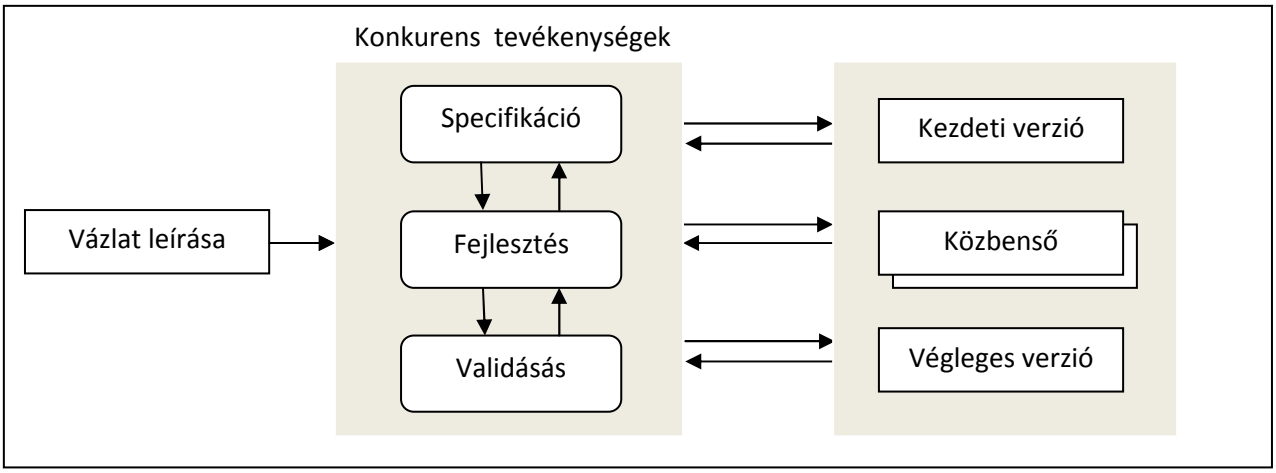
Hátránya: Egy követelményváltozás esetén elölről kell kezdeni a folyamatot.

3.2 Evolúciós modell (Tóth Zoltán):

A szekvencialitás tagadásaként jön létre. A központjában a párhuzamosság áll.
Prototípusokat állít elő, mely lehet:

- feltáró prototípuskészítés
- eldobható prototípuskészítés

Feltáró prototípuskészítés:



Első lépésként meg kell értenünk a követelmények egy részét. Ezt kell specifikálnunk, ezután implementálnunk, majd validálnunk. Ezek után áll elő a szoftver. Ezután az egészet elölről kezdjük, és addig ismételjük az összes követelménnyel, amíg előáll a végleges szoftver.
Így a program verzióinak sorozata jön létre. Minden újabb verzió újabb funkciókat tartalmaz a követelmények alapján.

Eldobható: Előáll a kezdeti követelmények alapján egy prototípus, és a rendelővel ez alapján pontosítunk, majd ezt eldobjuk, és újat csinálunk egészen addig, amíg a megrendelő igényeinek megfelelő termék jön létre.

Előnyei:

- Hamar működő szoftver áll a rendelkezésre.
- Először a lényeges dolgokat implementáljuk, vagy a megértett követelményeket.
- Van mindig használható eredmény.

Hátrányai:

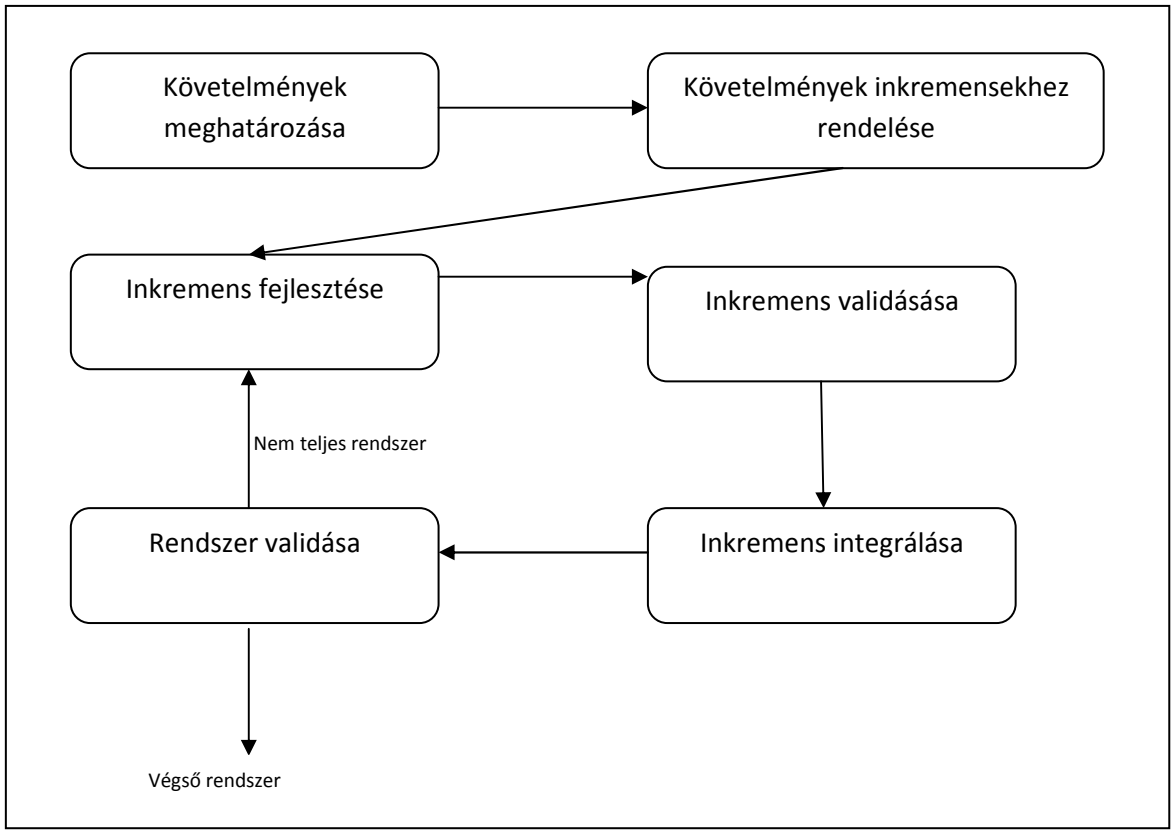
- Nem lehet tudni, hogy mikor van meg az összes követelmény, nincs teljes követelményrendszer.
- A teljes folyamat nem látszik, nehezen tervezhető, illetve nehezen időzíthető és költségelhető.
- A prototípusok általában szegényes architektúrájúak.

3.3 Inkrementális modell (Tóth Zoltán):

Az evolúciós modell továbbfejlesztéseként jött létre. Induláskor felépítünk egy vázlatos követelményt, és a feltáró prototípuskészítés alapján létrehozunk egy minimális rendszert. Ezt mindig egy inkremenssel bővítjük mindaddig, amíg a teljes követelményrendszert meghatározzuk és létrehozzuk a végleges szoftvert. Az inkremensek kis méretűek és behatárolhatóak.

Előnye: az evolúciós modellhez képest itt már a fejlesztés kezdetén is van rendszer architektúra.

Hátránya: Nehéz jól megtervezni az inkremenseket, és ezáltal a projektet.



4. MÓDSZERTANOK

Természetesen egy szakdolgozat 2 fővel nem elégítheti ki az agilitás összes feltételét, tulajdonságait, mégis mi megpróbáltuk a lehető legtöbb jellemzőjét felhasználni a project-ben. Módszertanok közül a legkönyebben alkalmazhatónak a Scrum ot tartjuk. A csapat létszáma (2 fő) miatt egyes feladatokat közösen végeztünk, más feladatokból többet is kiosztottunk 1 főre.

4.1 Agilis módszertanok (Tóth Zoltán):

A szó jelentése fürgeség. Egyesíti a vízesés modell és a monumentális szoftverfejlesztés előnyeit. „Az agilitás olyan adottság, amely egyaránt képes létrehozni és reagálni a változásokra és ezzel előnyt szerezni egy turbulens üzleti környezetben. Az agilis szervezetek befogadják, sőt generálják a változásokat, és ezzel versenyelőnyhöz jutnak. Az agilis szervezetek egyrészt fürgék és rugalmasak, másrészt képesek megtalálni a káosz és rend közötti egyensúlyt.” Az agilistás nem egy új módszertan. Több módszertan összefoglalása, melyek kielégítik az agilitás feltételeit. Előtérbe helyezi az egyént és a személyes kommunikációt a módszertanokkal szemben, és a működő szoftvert a dokumentációnál. Nem ragaszkodik mereven a szerződéshez, hanem a megrendelővel törekszik együttműködésre. Gyorsan reagál a bekövetkező változásokra.

4.2 Scrum (Tóth Zoltán):

A scrum egy folyamat váz, amely magában foglal bizonyos tevékenységeket és meghatározott szerepeket. A scrum főbb szerepkörei a "Scrum Master", aki a folyamatot felügyeli és a "Csapat" (Team) ami 5 főből áll és lefedi az összes munkafolyamatot.

Minden "futam" (sprint) során - amely 1 és 2 hét közötti időtartamot jelent (a csapat döntésétől függően) - a csapat egy működő szoftver egységet hoz létre. A futam során megvalósítandó funkciók a "Project Backlog"-ból (termék teendő lista) kerülnek ki, ami az elvégzendő munka magas szintű követelményeiből álló, fontossági sorrendbe állított lista.

Hogy a futam során a lista melyik elemei kerülnek megvalósításra azt a futam elején tartott "futam tervező" megbeszélés során választják ki. A futam folyamán a "futam teendő listát" nem lehet megváltoztatni, a futam során elvégzett tevékenységek rögzítettek. Amint a futam a végéhez ért, a csapat bemutatja az elkészült funkciókat.

A scrum egyik legfontosabb alapelve az, hogy felismeri és elfogadja, hogy a megrendelő a fejlesztés során meggondolhatja magát a követelményekkel kapcsolatban, és a váratlan változások nem kezelhetők könnyen a hagyományos, előzetes tervezési fázison alapuló módszerekkel. Ezért a scrum gyakorlati megközelítést választ, és elfogadja hogy nincs lehetőség a probléma teljes megértésére és definiálására. Inkább azt próbálja maximálisan elősegíteni, hogy a csapat napi "scrum" megbeszélés.

Szerepkörök

„Disznók" : Akik „mindenüket" beleadják

Terméktulajdonos: A vásárlót reprezentálja

Scrum Master: Maga a scrum működtetés, akadálymentesítés

Scrum Team: 5-9fős csapat, különböző területekről48

„Csirkék": Közvetetten részei a folyamatnak

Felhasználók: Vélemény, részeredmény (visszacsatolás)

Stakeholder-ek: Pl.: rendszergazda, igazgató

Tanácsadó szakértők: Akik nem szükségesek folyamatosan, csak 1-1 szakaszban válnak „disznóvá"

Dokumentumok

Story: a megrendelőtől érkező lényegi leírás

Product backlog: a story feldolgozása és priorizálása

Sprint backlog: a konkrét feladatok feltüntetése az adott sprintre (ki, mit, milyen határidővel vállalt be)

A futam során naponta tartott megbeszélés.

- A kommunikáció ebben az esetben e-mailben történik, mivel a csapat tagjai nem egy helyen dolgoznak.
- A csapat minden tagjának naponta kell helyzetjelentést adni.
- Az e-mailnek címszavakban kell tartalmazni a következőket:
  - Mi az, amit a tegnapi megbeszélés óta csináltam
  - Mi az, amit a mai nap tervezek csinálni
  - Vannak-e akadályok, amik gátolnak a cél elérésében.("Scrum Master" összegyűjti ezeket, és megpróbálja megoldani őket.)
- Minden nap ugyanabban az időpontban tartják

CI (Continous Integration - Folyamatos Integráció): (Hudson) nightly build-ek készítése, tesztek, metrikák automatikus futtatása a kódon, a kódminőség biztosítása érdekében.

- Checkstyle: Statikus kódminőség ellenőrzése, dokumentáltság, konvenciók betartása.
- Coverage: Kód tesztekkel való lefedettségének ellenőrzésére. a BDD segítségével 85-90% az elfogadható lefedettség a business rétegben.

A projektben résztvevők feladatai

Fejlesztő:

Előre megadott specifikáció és követelmények alapján elkészíti a kódot.A követelmények alapján fejlesztői tesztet végez.(Ellenőrzi saját munkáját.)

Területek:

- Business
- Presentation
- Integration

Technológiai felelős:

A hozzá tartozó terület technológiai kérdéseit hivatott megválaszolni, javaslatokat tesz a fejlesztés adott layerben történő egyszerűsítésére, hatékonyabbá tételére. Ő is fejleszt a saját területén.

Területek:

- Business
- Presentation

Project vezető:

Kapcsolatot tart az ügyféllel, a sprint indító megbeszélésen kiosztja a feladatokat, menedzseli azok állapotait, biztosítja a szükséges erőforrásokat.

Architecht:

Rálátása van a rendszer teljes technológiai felépítésére, nagy vonalakban ismeri a követelményeket az esetleges technológiai veszélyeket. Telepítési útmutatót és üzemeltetési dokumentumokat készít.

Tesztelő:

Az előre elkészített teszteseteket teszteli végig minden sprint végén, általa bejegyzett hibákat kezeli(ellenőrzi, a hiba státuszát állítja). Az elkészült funkciókat felhasználói dokumentációval látja el, esetleg elkészíti a súgót.

Quality manager:

Elkészíti a teszteseteket, ellenőrzi azok rendszeres futtatását. Koordinálja a tesztelés folyamatát. A tesztelők által elkészített dokumentumokat rendszerezi, ezekből összeállítja a végleges felhasználói kézikönyvet.

Scrum master:

Személye sprint elején dől el. Feladata az adott sprintben a scrum folyamatok vezetése, nyitó és záró dokumentumok elkészítése, illetőleg a staging rendszer felállítása teszteléshez, és a funkciók készültségi fokának ellenőrzése.

Szerepek:

Tóth Zoltán	**Kerék Tamás**	**Dr. Kuki Attila**
Fejlesztő: Business	Fejlesztő: Presentation	
Tesztelő	Tesztelő	
Quality manager Businnes	Quality manager Presentation	Architecht
		Project vezető

Egymással történő kommunikáció formái

- Napi szinten a scrum keretein belüli csoportos e-mail váltás(group mail).
- Sprint nyitó és záró értekezletek alkalmával személyesen megbeszélés.
- Bugot, taskot érintő kommunikáció e-mailen keresztül.
- Minden egyéb típusú kérdéssel kapcsolatban instant messenger alkalmazása.

Fejlesztési módszertan

FDD (Feature Driven Development):

Funkció vezérelt fejlesztés. A program tekinthető funkciók halmazának, a követelmények feltárása után ezen funkciók meghatározása a feladat. Ezen funkciók kerülnek kiadásra a fejlesztőknek. Akik tovább bontják azt. Ha egy funkció komplexitása miatt nem fér bele egy munkanapba, akkor tovább bontandó. A fejlesztő egy fejlesztési ciklusban egy funkciót fejleszt le.

BDD (Behaviour Driven Development)

A unit tesztelést elősegítő és megkövetelő fejlesztési metodológia. Segítségével 85-90%-os lefedettség érhető el, így a kódnak szinte minden sora már a fejlesztés során tesztelve van.

Fejlesztés menete:

A fejlesztő első lépésként létrehozza az implementálandó metódus headerjét, majd létrehozza a hozzá tartozó első tesztet. Majd mielőtt implementálni kezdené az üzleti logikát minden egyes követelményre elkészíti a megfelelő tesztet.

Teszt metódus felépítése:

Név: [MetódusNeve]should[Throw|Give|doSomeThing]

Részei:

- Given: Mockok beállítása
- When: metódus paraméterezése, meghívása
- Then: Eredmények ellenőrzése

Együtt működése:

A BDD megadja hogy mikor, és mit. Az FDD meg hogy hogyan.

4.3 Alkalmazásszerver (Tóth Zoltán):

Webáruház esetén egyik legfontosabb tényező az adatok biztonsága. Ebbe bele értjük az adatok integritásának megőrzését, és a személyes adatok védelmét.

Az adatintegritás megőrzése érdekében felállíthatunk egy többrétegű, lehetőleg 3 rétegű rendszer architektúrát. Ez egy kisebb vállalat számára nem megfizethető költségeket jelentene. Erre tökéletes alternatívát jelent a Microsoft Azure platformja, mely egy elosztott számítási felhőn alapszik. 3 földrészen 6 fő adatközpont és háromszoros adattárolás jellemzi. Mai rendszerek közül ez valósította meg legjobban a 99,999 rendelkezésre állást. A költség erőforrás használaton alapszik, így kisebb cégeknek is van lehetőségük optimalizálni a költségeiket. Amennyiben az alkalmazás felhasználói tábora jelentős növekedésnek indul, ellentétben a saját szerverpark lehetőségeivel, nem kell új szervereket vennünk, beállítanunk és integrálnunk a rendszerbe. Az Azure rendszerben grafikus felületen tudunk nagyobb erőforrásokat beállítani az alkalmazásunknak.

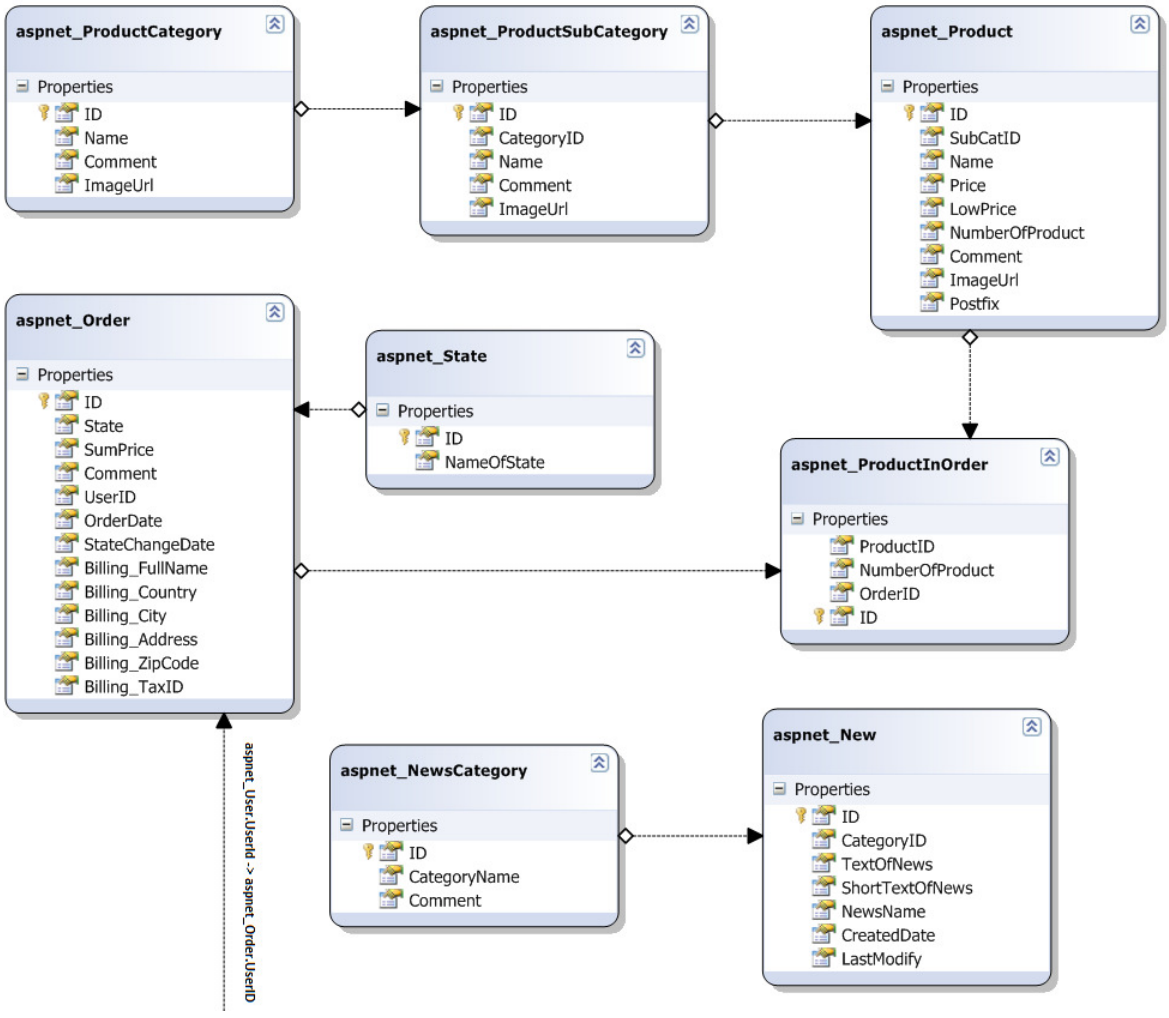
5. AZ ELKÉSZÜLT ALKALMAZÁS ISMERTETÉSE

A rendszer tervezésének első lépése, egy igényfelméréssel indult. Dokumentáltuk egy lehetséges megrendelő által támasztott igényeket, továbbá figyelembe vettük annak a rétegnek az elvárásait, akik ténylegesen használnák az alkalmazást. Ezen szempontok alapján alakítottuk ki a rendszer különböző funkcióit, amelyek tovább bővíthetők a rendszer életciklusa során.

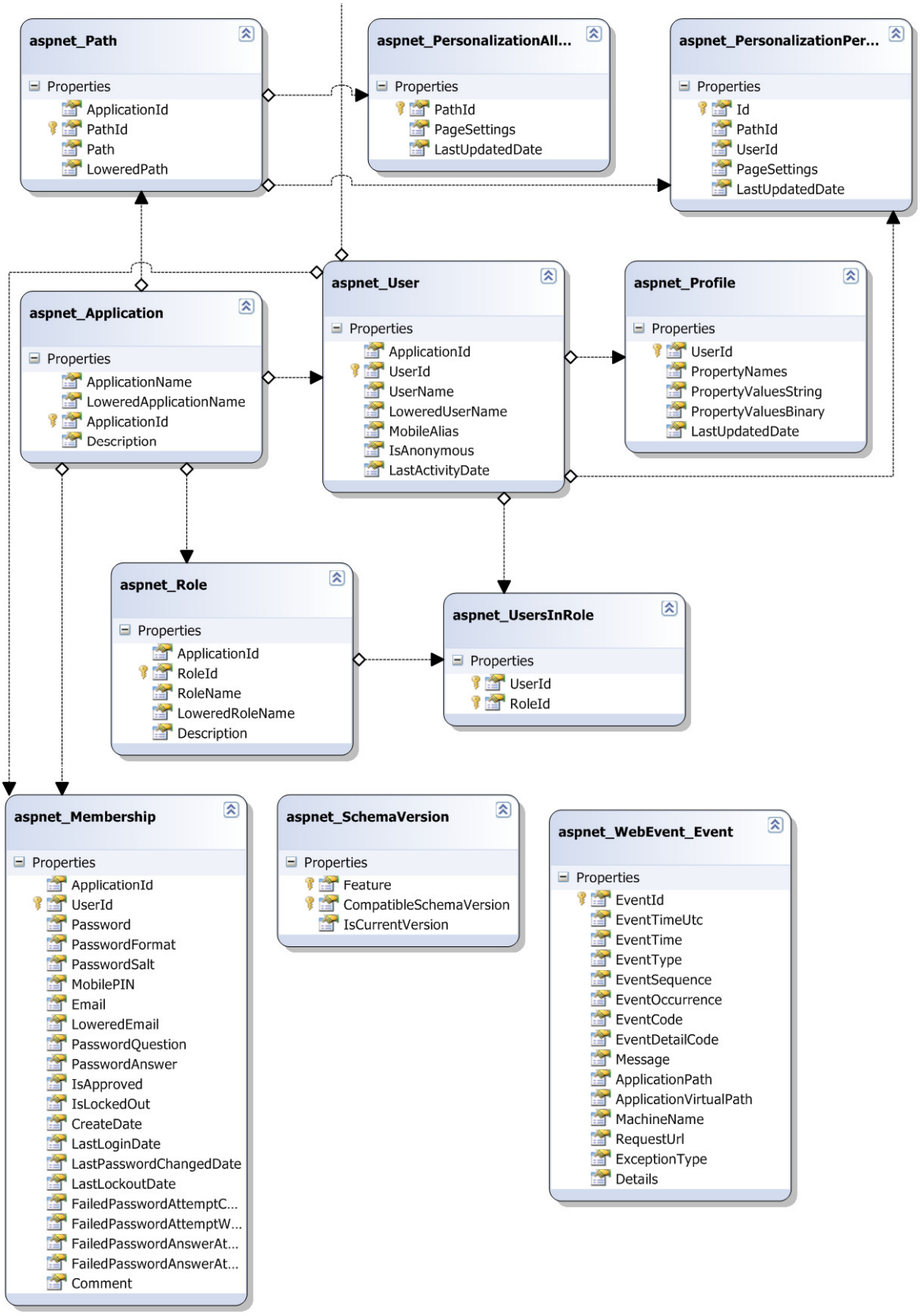
Az adatbázis létrehozása során, a bevezetőben említett ”készen kapott” adatbázis megismerése volt az első lépés. Törekedtünk ennek felhasználására és bővítésére úgy, hogy az adatok tárolása optimális legyen. Kerültük az adatbázisban a redundáns adatbáziselemeket, mivel a rendszer működésére a későbbi adatbázis frissítés során kihatással lehet, adatbázis inkonzisztenciát okozva. A következő lépés, a saját tábláink és az azokra vonatkozó megszorítások megtervezése volt. Ekkor már figyelembe kellett vennünk azokat a szempontokat, amelyek a kialakítandó rendszerrel kapcsolatosak. A tárolni kívánt adatokat több táblára bontottuk, kialakítottuk az ezen táblák közötti kapcsolatokat.

5.1 Az adatbázis kialakítása:

Az elkészült webáruház egyetlen adatbázist használ a felhasználókkal, a termékekkel valamint a megrendelésekkel kapcsolatos adatok tárolására. A táblák nevei és a mezők nevei angol főnevek lettek. Mivel a generált táblák nevei ’aspnet_’ előtagot kaptak, így az általunk létrehozott táblaneveket is ilyen előtaggal szerepeltettük. Azokban a táblákban, melyeket mi adtunk az adatbázishoz, minden első mező neve ’ID’ lett, melyek a tábla elsődleges kulcsaként funkcionálnak. Azokban a táblákban, melyekben külső kulcsot használtunk, a kulcsot tartalmazó mező neve a főtábla nevéből és az ’_ID’ utótagból tevődik össze.



(Az alkalmazás adatbázis-kapcsolatait realizáló diagram)



(Az ASP.NET felhasználó kezelésére használt adatbázis-kapcsolatokat realizáló diagram)

5.1.1 Az adatbázisban szereplő általunk létrehozott táblák ismertetése (Kerék Tamás):

Mivel fejlesztésünk során alkalmaztuk a fejlesztőeszköz által biztosított adatbázist, valamint ezt bővítve újabb táblákat hoztunk létre, ismertetjük a generált adatbázistáblákat valamint saját tábláinkat is.

Az 'aspnet_News' (Hírek) tábla:

- ID: Az 'aspnet_ News' tábla elsődleges kulcsa
- CategoryID: Külső kulcs az 'aspnet_NewsCategories' táblához való kapcsolódásra.
- TextOfNews: A hír szövegét tartalmazza
- ShortTextOfNews: A hír rövid leírását tartalmazza
- NewsName: A hír címe
- CreatedDate: Létrehozás dátuma
- LastModify: Utolsó módosítás dátuma

Az 'aspnet_NewsCategories' (Hír kategóriák) tábla:

- ID: Elsődleges kulcs
- CategoryName: Kategória neve
- Comment: Megjegyzés a kategóriához

Az 'aspnet_ProductCategories' (Termék kategóriák) tábla:

- ID: Elsődleges kulcs
- Name: A termékkategória neve
- Comment: Termékkategóriához tartozó megjegyzések
- ImageUrl: A termékkategóriához tartozó kép linkje

Az ’aspnet_Orders’ (Hírek) tábla:

- ID: Elsődleges kulcs
- State: A megrendelés állapota
- SumPrice: A megrendelés végleges összege
- Comment: A megrendelés folyamán megadott megjegyzés
- UserID: Külső kulcs az ’aspnet_Membership’ táblához való kapcsolódásra
- OrderDate: Megrendelés dátuma
- StateChangeDate: Az állapot módosításának dátuma
- Billing_FullName: Megrendelő neve
- Billing_Country: A címben megadott ország
- Billing_City: A címben megadott város
- Billing_Address: A megrendelésben megadott cím
- Billing_ZipCode: A megrendelésben megadott irányítószám
- Billing_TaxID: A megrendelő adószáma

Az ’aspnet_Products’ (Termékek) tábla:

- ID: Elsődleges kulcs
- SubCatID: Külső kulcs az ’aspnet_ProductCategories’ táblához való kapcsolódásra
- Name: A termék neve
- Price: A termék ára
- LowPrice: A termék akciós ára
- NumberOfProduct: A termék elérhető darabszáma
- Comment: Megjegyzés a termékhez
- ImageUrl: A termékhez tartozó kép linkje
- Postfix: Mennyiséget jelző utótag (pl.: db, kg, liter, tonna)

Az ’aspnet_ProductInOrders’ (Termékek a rendelésben) tábla:

- ID: Elsődleges kulcs
- ProductID: Külső kulcs az ’aspnet_Products’ táblához való kapcsolódásra
- NumberOfProduct: A rendelésben szereplő termék darabszáma
- OrderID: Külső kulcs az ’aspnet_Orders’ táblához való kapcsolódásra

Az ’aspnet_ProductSubCategory’ (Termék alkategória) tábla:

- ID: Elsődleges kulcs
- CategoryID: Külső kulcs az ’aspnet_ProductCategories’ táblához való kapcsolódásra
- Name: Az alkategória neve
- Comment: Megjegyzés az alkategóriához
- ImageUrl: Az alkategóriához tartozó kép linkje

5.1.2 Automatikusan generált adatbázis (Tóth Zoltán):

Az 'aspnet_User' (Felhasználók)

- ApplicationId: Alkalmazás azonosítója
- UserId: Elsődleges kulcs. Felhasználó azonosító
- UserName: Felhasználó neve
- LoweredUserName: Felhasználó neve kis betűkkel
- MobileAlias: Felhasználó mobil álneve. Jelenleg nem használt.
- IsAnonymous: Azonosítatlan felhasználó e
- LastActivityDate: Utolsó aktivitás

Az 'aspnet_Profile' (A felhasználók profil mezői)

- UserId: Elsődleges kulcs. Felhasználó azonosító
- PropertyNames: A profilmezők nevei és helyük sorrendben.
- PropertyValuesString: A stringé konvertálható mezők értékei
- PropertyValuesBinary: A bináris adatok
- LastUpdatedDate: Utolsó frissítés ideje

Az 'aspnet_Membership' (A felhasználók személyes információi)

- ApplicationId: Alkalmazás azonosítója
- UserId: Felhasználó azonosítója, Elsődleges kulcs
- Password: Felhasználó jelszava
- PasswordFormat: Jelszó formátuma
- PasswordSalt: Véletlen generált 128 bites érték a jelszó kódolásához
- MobilePIN: Felhasználó mobil telefonszáma
- Email: Felhasználó e-mail címe
- LoweredEmail: Felhasználó e-mail címe kis betűkkel

- PasswordQuestion: Elfelejtett jelszó visszaállítási kérdés
- PasswordAnswer: Elfelejtett jelszó visszaállítási kérdés válasza
- IsApproved: A felhasználó jóváhagyott e.
- IsLockedOut: A felhasználó zárolt e.
- CreateDate: Felhasználó létrehozásának ideje
- LastLoginDate: Felhasználó utolsó bejelentkezése
- LastPasswordChangedDate: Felhasználó utolsó jelszóváltása
- LastLockoutDate: Utolsó zárolás ideje.
- FailedPasswordAttemptCount: Rontott jelszó számláló
- FailedPasswordAttemptWindowStart: Az első hibás jelszó megadás dátuma.
- FailedPasswordAnswerAttemptCount: Jelszócsere biztonsági kérdésére adott hibás válasz számlálója.
- FailedPasswordAnswerAttemptWindowStart: Az első biztonsági kérdésre adott hibás válasz ideje.
- Comment: Megjegyzés

Az 'aspnet_PersonalizationPerUser' (Felhasználó szintű személyre szabás) tábla:

- Id: Elsődleges kulcs
- PathId: Elérési útvonal azonosítója
- UserId: Felhasználó azonosító
- PageSettings: Oldal beállítások
- LastUpdatedDate: Utolsó frissítés

Az 'aspnet_Path' (Elérési utvonalak) tábla:

- ApplicationId: Alkalmazás azonosító
- PathId: Elérési útvonal azonosítója. Elsődleges kulcs
- Path: Elérési út
- LoweredPath: Elérési út kis betűkkel

Az ’ aspnet_PersonalizationAllUser’ (Személyre szabás) tábla:

- PathId: Elérési útvonal azonosítója. Elsődleges kulcs
- PageSettings: Oldal beállítások
- LastUpdatedDate: Utolsó frissítés

Az ’ aspnet_Application’ (Alkalmazás) tábla:

- ApplicationName: Alkalmazás neve
- LoweredApplicationName: Alkalmazás neve kis betűkkel
- ApplicationId: Alkalmazás azonosítója. Elsődleges kulcs
- Description: Alkalmazás leírása

Az ’ aspnet_Role’ (Szerepkörök) tábla:

- ApplicationId: Alkalmazás azonosítója
- RoleId: Szerepkör azonosítója. Elsődleges kulcs
- RoleName: Szerepkör neve
- LoweredRoleName: Szerepkör neve kis betűkkel
- Description: Szerepkör leírása

Az ’ aspnet_UsersInRole’ (Kapcsoló a felhasználó és a szerepköre között) tábla:

- UserId: Felhasználó azonosítója. Elsődleges kulcs (összetett)
- RoleId: Szerepkör azonosítója. Elsődleges kulcs (összetett)

Az ’aspnet_WebEvent_Event’ (Kiváltódott események) tábla:

- EventId: Esemény azonosítója. Elsődleges kulcs

- EventTimeUtc: Esemény bekövetkezési ideje UTC fomátumban
- EventTime: Esemény bekövetkezési ideje
- EventType: Esemény típusa
- EventSequence: Esemény sorszáma (WebBaseEvent.EventSequence)
- EventOccurrence: Hányszor fordult elő az esemény
- EventCode: Esemény kódja (WebBaseEvent.EventCode)
- EventDetailCode: Esemény részletei
- Message: Esemény üzenete
- ApplicationPath: Alkalmazás elérési útja a szerveren
- ApplicationVirtualPath: Alkalmazás elérési útja root mappától.
- MachineName: Az eszköz neve amin az esemény kiváltódott.
- RequestUrl: Az URL amin az esemény kiváltódott
- ExceptionType: Kivétel típusa
- Details: Részelek

Az ’aspnet_SchemaVersion’ () tábla:

- Feature: Funkció neve. Elsődleges kulcs (összetett)
- CompatibleSchemaVersion: Kompatibilis séma verzió. Elsődleges kulcs (összetett)
- IsCurrentVersion: Ha aktuális verzió akkor 1, különben 0

5.2 A rendszer használatának ismertetése

5.2.1 A bejelentkezés

A böngésző címsorába, az oldal címét beírva, a felhasználó a következő oldalt látja (1. ábra). Ha egy felhasználó a korábbiakban már regisztrált az oldalon, akkor a bejelentkezés feliratra kattintva megadhatja felhasználónevét és jelszavát, amit egy azonosítás követ. Regisztrációra a bejelentkezés lapon megjelenő linken van lehetőség.

(A bejelentkezés)

Az azonosítás során, ha a felhasználónév és jelszó mezők kitöltésében nem történt hiba, megtörténik az azonosítás, lekérdezésre kerül a felhasználó jogköre, különben üzenetet kap a felhasználó a hibás adatok megadásáról. Lehetőség van a bejelentkezés folyamán, a bejelentkezési adatok megtartására, ekkor egy cookie jön létre. Az oldal felhasználói három jogkörbe sorolhatóak. Normál felhasználó, adminisztrátor és rendszer adminisztrátor. Ezen jogkörök figyelembe vételével, belépés után dinamikusan generálódik egy a jogkörhöz tartozó

menüsor. Bejelentkezés nélkül is lehetőség nyílik a következő menüpontok megtekintésére: Főoldal, Termékek, Hírek, Cégünkről és üzletszabályzat. Bejelentkezés után a felhasználó a főoldal menüpont alatt található oldalra lesz átirányítva, és visszajelzést kap a sikeres bejelentkezésről egy üdvözlet formájában, az átalakított bejelentkezés gomb mellett.. A bejelentkezés gomb (amely egy link gomb) szövege kijelentkezésre vált és a továbbiakban ezt a funkciót tölti be.

5.2.2 A regisztráció:

Azok a látogatók, akik még nem regisztráltak az oldalon, nem rendelkeznek felhasználói jogkörrel, az ismertetett módon a regisztrációs űrlapot kitöltve regisztrálhatnak. A regisztráció lebonyolításáért, a projekt kezdetén automatikusan generált, Register.aspx oldal a felelős. Ez a következőképpen néz ki (2. ábra):

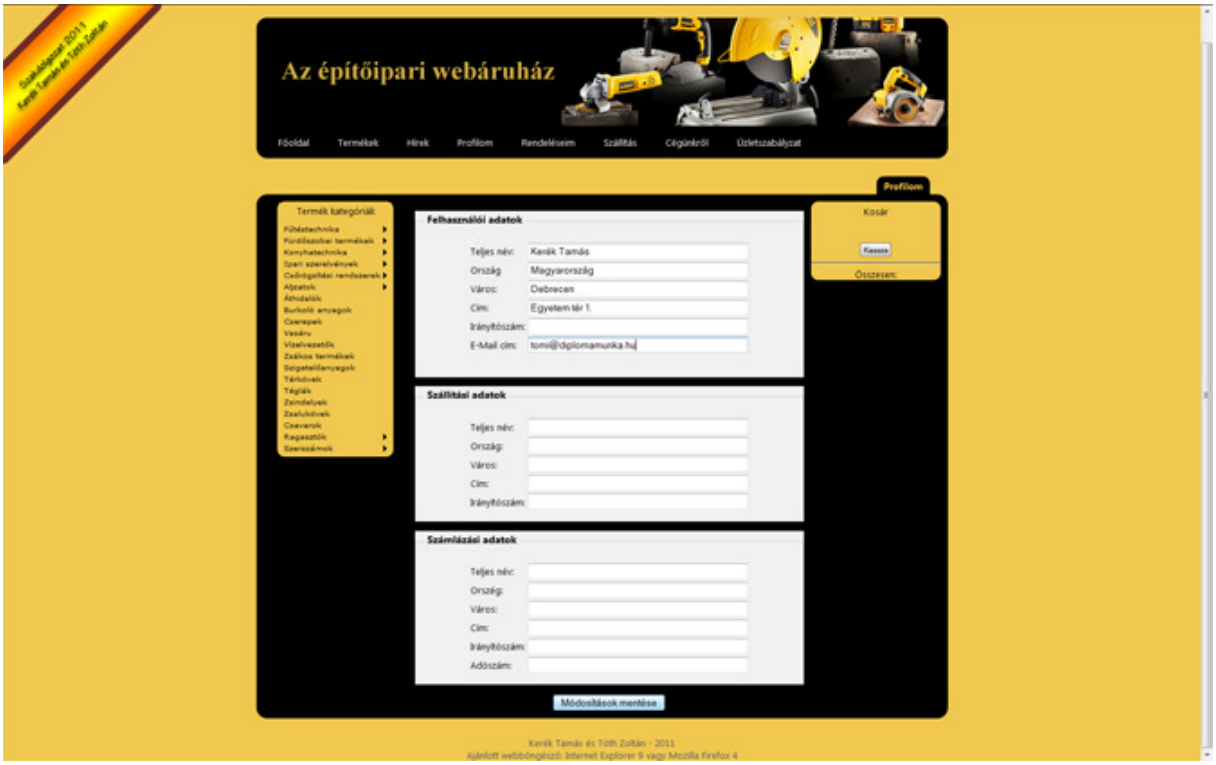
(Regisztráció)

Sikeres regisztráció után, a már felhasználói jogkörrel rendelkező felhasználót a rendszer bejelentkezteti, különben üzenetekben értesítést küld az esetleges hibákról. Ilyen hibák

lehetne a következőek: a regisztráció során nem lett kitöltve a felhasználónév, e-mail cím, jelszó és a jelszó megerősítése mező. A regisztrációs űrlapon szereplő mezők mindegyikének kitöltése kötelező.

5.3.3 A felhasználói profil:

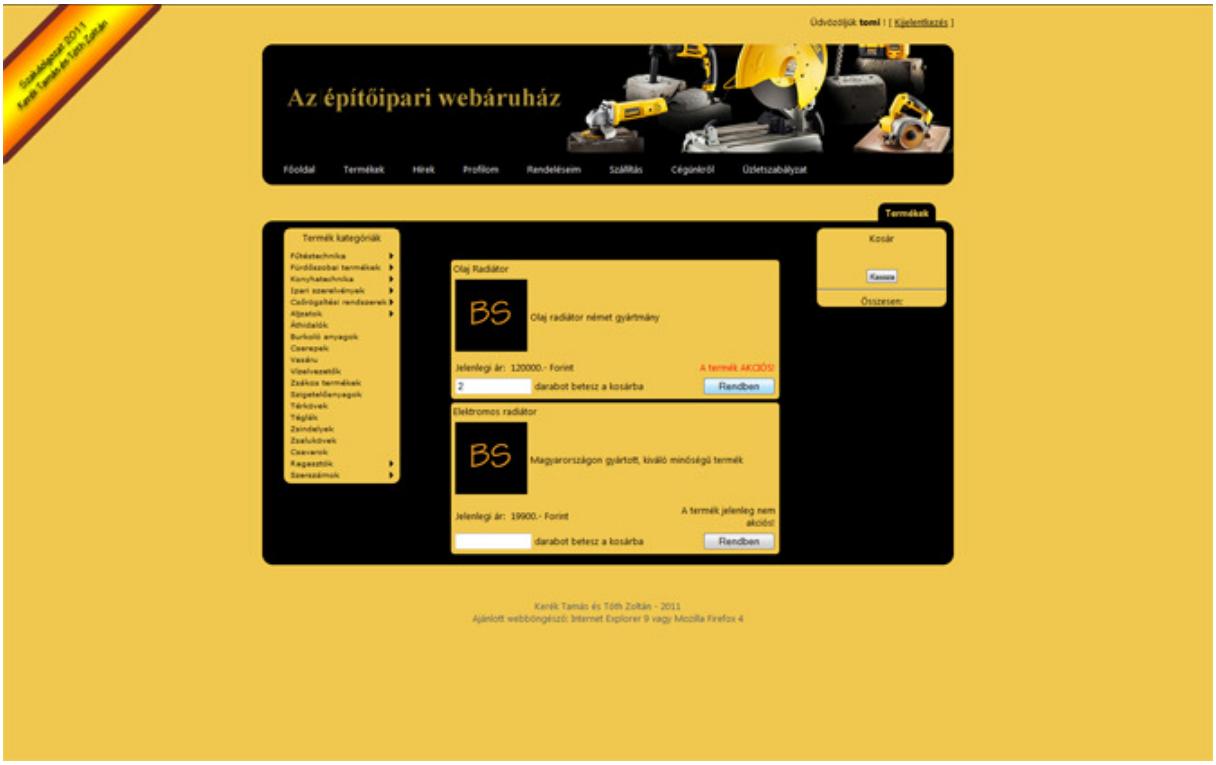
Miután a normál jogkörrel rendelkező felhasználó bejelentkezett az oldalra, az előzőekben már legenerálódott menüpontok mellett megjelennek Profilom és a Rendeléseim menüpontok. A Profilom menüpont alatt (3. Ábra) a normál jogkörrel rendelkező felhasználó is módosíthatja saját adatait, valamint kitöltheti azon mezőket, amelyeket a regisztráció során nem kértünk, hogy töltsön ki. Ezeket az adatokat az oldalon három csoportba osztottuk: Felhasználói adatok, szállítási adatok és számlázási adatok. Ha ezek bármelyikét módosítani kívánja a felhasználó, akkor a megfelelő mezők kitöltése vagy módosítása után a Módosítások mentése gombra kattintva elmentheti.



(Profil szerkesztése)

5.3.4 A rendelés menete:

Azok a látogatók, akik még nem regisztráltak be, ugyanúgy megtekinthetik a termékeket, mint azok, akik már beregisztráltak. Ellentétben a regisztrált felhasználókkal szemben, a látogatók a kosár funkciót nem használhatják. A termékek menüt és a kosarat úgy alakítottuk ki, hogy az közvetlenül elérhető legyen, bármelyik lapon tartózkodunk is jelenleg. A termékek menüben megjelennek az adatbázisban szereplő főkategóriák és ezen menüelemek almenüjeként pedig az alkategóriák. Ezekre bármikor kattintva megjelenik a terméklista az adott alkategóriában (4. Ábra).



(Termékek listázása az aktuális alkategóriában)

A megjelenő terméklistában, az aktuális alkategóriában, vagy ha főkategóriára kattintottunk, akkor a főkategóriában szereplő termékek jelennek meg. Ha az adatbázisban, a termékhez megadtunk képet hivatkozásként, akkor annak képe látható, különben egy alapértelmezett 'BS' feliratú kép jelenik meg. Minden termék alján látható egy mező, melynek értékül adhatja a felhasználó a rendelni kívánt mennyiséget. Miután a mennyiség beállítása megtörtént, a 'Rendben' feliratú gombra kattintva a termék a kosárba kerül. A kosár alján a termékektől és a termékek mennyiségétől függően egy összegzést láthat a felhasználó a fizetendő összegről.

Miután minden rendelni kívánt termék a kosárban van, a kosár ’Kassza’ feliratú gombján folytatódik a rendelés következő lépése. Az ekkor megjelenő oldalon (5. Ábra) részletes jelentést kapunk a kosárban lévő termékekről, azok áráról és a rendelésben megadott mennyiségről, továbbá módosíthatjuk a rendelni kívánt darabszámot, vagy törölhetünk is termékeket. A cím mező automatikusan kitöltődik a számlázási adatokban megadott címmel, de ez módosítható más címre is.

(A kassza)

Az ezen az oldalon található megjegyzés mezőben adhatja meg a felhasználó a rendeléssel kapcsolatos megjegyzését például a szállítás dátumával kapcsolatban. A ’Megrendelés’ gombra kattintva a felhasználó profil oldalon megadott e-mail címére a rendszer üzenetet küld és alul zöld felirat jelenik meg az email elküldéséről. A rendelések állapota a ’Rendelésiem’ menüpont alatt továbbkövethetők. Amint a rendelés feldolgozásra került, az itt látható ’Rendelés állapota’ oszlopban megjelenő ’Feldolgozás alatt’ állapot ’Feldolgozva’ állapotra vált át. Ha egy rendelés részleteire kíváncsi egy felhasználó, akkor az itt megjelenő rendelések mellett található részletek linkre kattintva, az aktuális rendelésről kap kimutatást.

5.3.5 Az adminisztráció:

A webáruház felhasználóinak, mint azt már ismertettük, lehet adminisztrátori és rendszeradminisztrátori jogköre is. Ezen jogköröket regisztráció alapján nem lehet megkapni, hanem egy rendszer adminisztrátori jogkörrel rendelkező felhasználó adhatja. Ennek menetét a későbbiekben ismertetjük. A felhasználó bejelentkezése után megtörténik az azonosítás. Ha itt adminisztrátorként azonosítja a rendszer a felhasználót, akkor a normál jogkörrel rendelkező felhasználó menüpontjain kívül elérhetővé válik az ’Adminisztráció’ menüpont melynek a következők az almenüjei: Felhasználók kezelése, Hírek kezelése, Adminisztrátorok kezelése, Termékek kezelése valamint a Megrendelések kezelése. Habár az adminisztrátori feladatok nagy része elvégezhető az adatbázisban lévő elemek módosításával vagy új elemek hozzáadásával, az ismertetett menüpontokon keresztül az adatbázis ismerete nélkül is véghezvihetőek a menüpontok által megnevezett funkciók.

5.3.6 A felhasználók kezelése:

Az adminisztrátori jogkört betöltő felhasználók, jogosultságot kapnak az ’Adminisztráció’ menüpont almenüpontjai által címzett, az ’Admin’ mappában található ASP lapok használatára. Az összes jogkörhöz tartozó menüpontok generálásáért egy Generate_Menu() nevű eljárás a felelős. Az eljárás megvizsgálja, hogy az aktuális felhasználó azonosított-e, valamint ha igen, akkor ellenőrzi annak jogkörét. Az eljárás egy részlete a következőképpen néz ki:

```
protected void Generate_Menu()
{
    if (Page.User.Identity.IsAuthenticated)
    {

        if (Page.User.IsInRole("Administrator"))
        {
            // Adminisztráció menü
            MenuItem Administration = new MenuItem();
            Administration.NavigateUrl = "~/Admin/Admin.aspx";
            Administration.Text = "Adminisztráció";
            NavigationMenu.Items.Add(Administration);
        }
    }

}
```

(A Generate_Menu() eljárás kódrészlete)

Miután az eljárás lefutott az azonosított és adminisztrátori jogkörrel rendelkező felhasználó ’Adminisztráció’ menüjében az első almenüpont a Felhasználók kezelése. A menüpont (6. Ábra) lehetőséget biztosít a felhasználók kezelésére, a velük kapcsolatok műveletek elvégzéséhez.

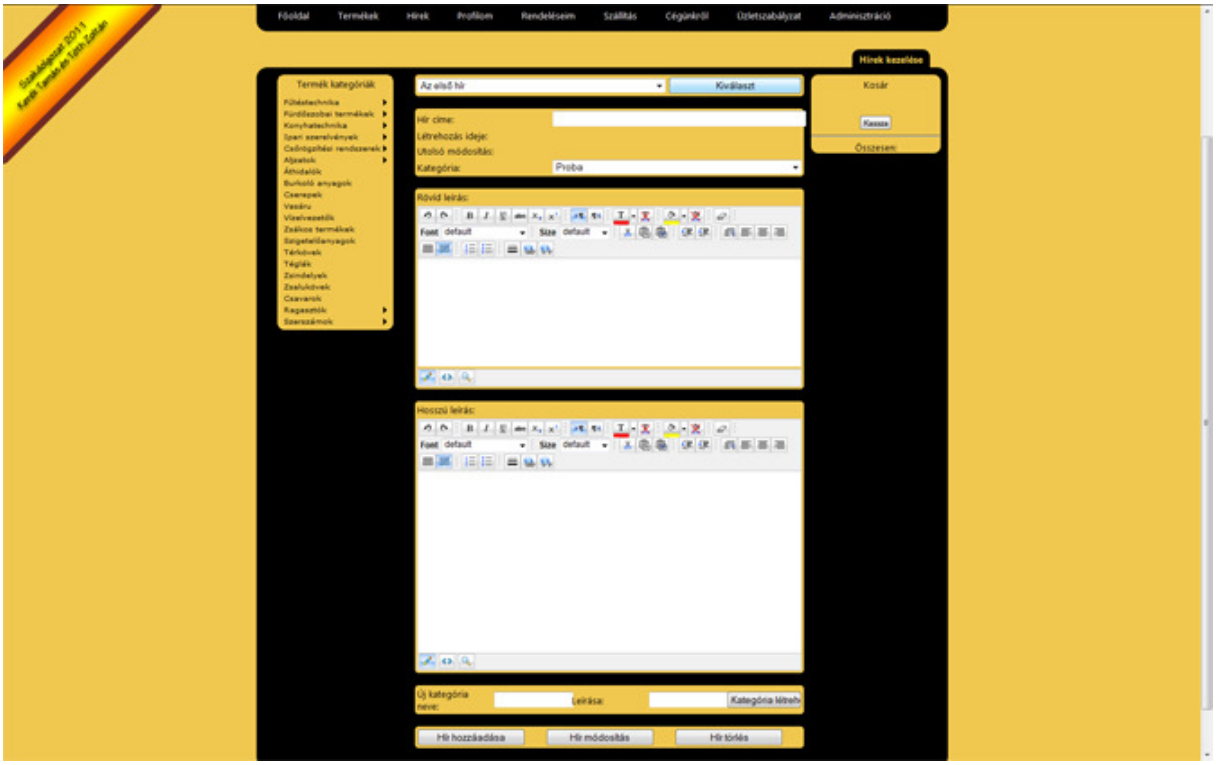
(A felhasználók kezelése)

Az oldal tetején egy görgethető listában találjuk a regisztrált felhasználókat. A lista melletti mező funkciója a szűrés, amely a listában található felhasználók szűrését végzi. A szűrés gombra kattintva, a mezőben található szöveget minden felhasználó nevében keresi a rendszer. Azok a felhasználók, akiknek a felhasználónevében megtalálható a szűrési feltétel, azok az oldal újragenerálódása után a lista elemeiként jelennek meg. Itt egy felhasználóra kattintva kitöltődnek a rá vonatkozó adatok a 6. ábrán látható mezőkben. Az oldal következő sorában a felhasználó jogköre, szerepköre kerül kitöltésre, a felhasználó kiválasztása esetén. Itt egy legördülő listában kiválaszthatunk egy jogkört, amit az aktuális felhasználónak szeretnénk adni, vagy éppen elvenni, ezt a ’Jog hozzáadása’ vagy a ’Jog elvétele’ gombokkal dönthetjük el. Az oldal harmadik részében a felhasználó adatai, szállítási adatai és számlázási

adatai töltődnek ki. Ezeket, a felhasználó kérésére az adminisztrátor módosíthatja. Az oldal jelenlegi állapotában a felhasználó törlését nem valósítottuk meg.

5.3.7 A hírek kezelése:

A hírek kezelése almenüpontban lehetőség nyílik új hír hozzáadására vagy egy már meglévő hír módosítására. Ebben a menüpontban végezhető el minden, a hírekkel kapcsolatos módosítás. A oldal első blokkjában egy legördülő lista található, melyben a hírek címei vannak felsorolva. Az e mellett található 'Kiválaszt' gomb megnyomására kitöltődnek a az oldalon található mezők (7. Ábra). A második blokk a hírekkel kapcsolatos információkat tartalmazza a hír kiválasztása után. Itt kapott helyet a hír címe, létrehozásának ideje, utolsó módosításának ideje valamint a hír kategóriája egy legördülő menüben. A legördülő menü elemei közül az van kiválasztva, amelyikbe a hírt soroltuk. A legördülő menü értékének megváltoztatásával a hír más kategóriába tehető át.



(A hírek kezelése)

A harmadik és negyedik blokkban egy AjaxControl, egy szerkesztő található, melyben a hír rövid szövegét és teljes tartalmát adhatjuk meg. Lehetőség nyílik a szövegek teljes formázására. Az ötödik blokkban találjuk az új kategória létrehozásához szükséges elemeket. Itt adhatjuk meg a kategória nevét és leírását, valamint a kategória létrehozása gombot, melyre kattintva létrejön az új kategória. A hatodik blokk gombokat tartalmaz, melynek segítségével hozzáadhatjuk a hírt, módosíthatjuk és törölhetjük azt.

5.3.8 A termékek kezelése:

A termékek kezelésére szolgáló oldal öt blokkra oszlik (8. Ábra). Az első blokkban egy GridView vezérlő található, melyben az áruház termékeiről kapunk információkat, például a raktáron lévő darabszámról. A vezérlő első oszlopa tartalmaz egy ’Kiválaszt’ linket minden egyes termékhez. Erre kattintva kitöltődnek a második blokkban található termékadatok, amelyek módosíthatóak. Ugyanezen blokkon belül, ha nem kattintottunk egyetlen termék ’Kiválaszt’ linkjére, az adatok kitöltésével új termék vihető fel. Kiválasztott termék törlésére ugyanezen blokkon belül a ’Törlés’ gomb funkcionál.

(A termékek kezelése)

A harmadik és negyedik blokk a termékek főkategóriáinak valamint alkategóriáinak létrehozására szolgál. Az utolsó, ötödik blokkban, egy a hírek kezelésénél is alkalmazott szerkesztőt helyeztünk el, a termékhez tartozó szöveg megadására.

5.3.9 A megrendelések kezelése:

Az adminisztrátori menü utolsó almenüjében, az áruház alkalmazottai a megrendeléseket követhetik nyomon egy GridView vezérlőn keresztül (9. Ábra). Az itt elhelyezett vezérlő, lehetőséget biztosít egy rendelés részleteinek a teljes megtekintésére.

(A megrendelések kezelése)

Az oldal legfelső részén elhelyezett panel segítségével szűrhetünk a rendelés állapota, valamint dátuma alapján. A dátum megadására egy AjaxControl segítségével megjelenő naptáron keresztül biztosítottunk lehetőséget.

A ’Részletek’ linkre kattintva, amely minden megrendelés mellett megtalálható, alul egy ’Rendelés részletei’ panel nyílik meg. Ebben a rendelésben megadott termékekről, rendelt darabszámáról és egységáráról kapunk információt. Az információk mellett található lista tartalmazza a rendelés lehetséges állapotait, mint például a Feldolgozás alatt vagy az Átadva a futárnak. Ha ebben a listában kiválasztunk egy állapotot, akkor az aktuális megrendelés állapota a kiválasztott állapotra módosul.

6. ÖSSZEGZÉS

Szakdolgozatunk céljaként törekedtünk egy webes alkalmazáson keresztül bemutatni az alkalmazott technológiákat. Az ok ami miatt ezt a témát és ezen eszközök használatát választottuk, hogy oktatás keretein belül lehetőségünk volt betekinteni a C# programozási nyelvbe valamint a .NET technológiába és szerettünk volna a .NET webes részével is megismerkedni. Sajnos komolyabb oktatás keretein belül erre nem volt lehetőségünk, ezért gondoltunk arra, hogy a szakdolgozat megírásának feladataként ezt a témát választjuk, így is rákényszerülve ennek megismerésére. Mivel az eddigiek során webes alkalmazás fejlesztésével nem foglalkoztunk, így csak a felsorolt szakirodalom folyamatos tanulmányozásával tudtuk elérni a kívánt célt.

A fejlesztés folyamán sikerült létrehoznunk egy olyan alkalmazást, egy olyan rendszert amely megfelelt elvárásainknak. Természetesen annak funkcióit tovább lehetne bővíteni, de szerintünk azt teljesen kialakítani két ember csak hosszú idő elteltével tudja. A kialakult felhasználói felületet saját elképzeléseink alapján terveztük meg, nyilvánvalóan egy grafikus segítségével sokkal jobb stílust lehetne a rendszer számára kialakítani. Az alkalmazást, ha lehetőségünk lesz rá a továbbiakban, bővíteni fogjuk egyre több, az igények szerint felmerült funkciókkal. Egyik fontos szempontunk is az volt, hogy az alkalmazás fejlesztése, bővítése könnyen megoldható legyen. Éles környezetben való tesztelése sajnos nem volt lehetséges, mivel egyenlőre csak egy ingyenesen használható IIS szervert tudunk Magyarországon, és a forráskódot sem szerettük volna mások által hozzáférhető helyen tartani a szakdolgozat leadásának időpontjáig.

A fejlesztés folyamán kiderült számunkra, hogy az ASP.NET ideális környezetet biztosít bárki számára web alkalmazás fejlesztésére. Összetettebb alkalmazások és egyszerű magán oldalak fejlesztésére is kiválóan alkalmas. Nem igényel a kezdetekben komolyabb erőbefektetést a megtanulása, azok számára sem akik eddig asztali alkalmazásokat készítettek .NET - ben. Természetesen nem a mi általink választott módszer az egyetlen a .NET eszközei közül. Az egyik leg kifinomultabb módszer a Microsoft MVC modell, ami jelenleg a 3. verziónál tart. Mi mégis a Web Forms technológiát választottuk, mivel egy asztali alkalmazásokat fejlesztésével foglalkozó fejlesztő, számára nem jelent komolyabb különbséget.

7. IRODALOMJEGYZÉK

- Alex Mackey: A .NET 4.0 és a Visual Studio 2010. Szak Kiadó, 2010
- Dr. Juhász István – Rendszerfejlesztés Technológiái előadása
- Hatvany Béla Csaba: ASP 3.0 programozás. ComputerBooks Budapest, 2004
- Matthew MacDonald, Adam Freeman, Mario Szpuszta: ASP.NET 4.0 in C# 2010 fouth edition. Apress, 2010
- Rob Cameron, Dale Michalk: Pro ASP.NET 3.5 Server Controls And AJAX Components. Apress, 2008
- http://web.axelero.hu/fodorsi/html/css1.html
- http://www.bibl.u-szeged.hu/inf/szakdoli/2004/rimar/css_szakdolgozat.pdf
- http://download.microsoft.com/download/7/F/8/7F8A26DD-B99A-426C-872C-584B1A7EBDF9/38-41.pdf
- http://users.iit.uni-miskolc.hu/ficsor/inftervseg/agilis.pdf
- http://avalon.aut.bme.hu/education/meresek/ailabor2/www.pdf
- http://en.wikipedia.org/wiki/Scrum_(development)
- http://www.agilealliance.hu/materials/books/ZsZs-ASD.pdf
- http://msdn.microsoft.com/en-us/library/7t6b43z4.aspx
- http://msdn.microsoft.com/en-us/vcsharp/aa336746
- http://www.adivo.com/samples/database/aspnetdb-lite/ASPNETDB.pdf