DIPLOMAMUNKA

Balmaz Tamás

Debrecen

2010

Debreceni Egyetem

Informatika Kar

WEBES ALKALMAZÁSFEJLESZTÉS ORACLE ADATBÁZISON

Témavezető:

Bérczes Tamás

egyetemi tanársegéd

Készítette:

Balmaz Tamás

programtervező matematikus

Debrecen

2010

Tartalomjegyzék

Bevezetés

Körülbelül 10 évvel ezelőttre tehető az az időszak, amikor az internet igen széles elterjedése következtében egyre több igény merült fel az online szolgáltatásokra. Az ember természetéből adódóan szeretné, például a vásárlással töltött idejét, minimálisra csökkenteni, abból a célból, hogy a többi tevékenységére több ideje maradjon. A szolgáltatók tehát igyekeztek kielégíteni a társadalom igényeit, ezáltal egyre több olyan weboldal készült, amely lehetővé tette a webshopping-ot, azaz a weben történő vásárlást, valamint ügyintézést. Ezen irányba való törekvés eredményeképp napjainkban szinte minden szolgáltatás elérhető a világ bármely pontjáról, bármikor, bárkinek. Érthető tehát, hogy a jelen embere egyre szívesebben használja az internet lehetőségeit akár a mindennapos bevásárlásra is.

A webfejlesztő cégek a bekövetkező események hatására belátták, hogy a profiljukat bővíteni kell az online vásárlás területére is, azaz elindultak a webshop alapú webes alkalmazások fejlesztési irányába. A dolgozatom írásának idejében már teljesen természetesnek vehető, hogy néhányuk erre a területre specializálódott és szinte csak ilyen jellegű munkákból tartja fent magát. Tehát véleményem szerint a webes alkalmazásfejlesztés egyik legfontosabb irányzatává nőtte ki magát a webbolt alapú rendszerek fejlesztése.

Feltételezem, hogy vannak már olyan emberek, akik nem tudnák az életüket elképzelni pl. e-Bay[1] vagy Vatera[2] nélkül. A webshopok gyorsak, egyszerű őket használni, egyre biztonságosabbak, több esetben olcsóbbak és nem utolsó sorban kényelmesek. Azonban a sok pozitívum mellett hátránya is van az ilyen típusú vásárlásnak: a ”nem kézzelfoghatóság”. Sok ember igényli azt, hogy fizikailag a kezében tartsa a megvásárolni kívánt terméket, vagy ruhadarab esetén szeretné felpróbálni azt, mielőtt súlyos összegeket fizetne érte. Akár műszaki áru esetén is szeretjük azt élőben látni, gondolok jelen esetben például arra, ha

[1] http://www.ebay.com/
[2] http://www.vatera.hu/

televíziót kívánunk vásárolni, ildomos megnéznünk, hogy milyen minőségű képet tud produkálni. Ezen kevésbé jó tulajdonságok közül kivételt jelentenek az online éttermek, ételrendelők, hiszen egy étteremben sem tudhatjuk előre, hogy az adott elnevezésű fogás pontosan mit takar mind ízvilágában, mind minőségében.

A diplomamunkámmal kapcsolatos választásom az előbb említett okból esett egy online étterem elkészítésére valamint bemutatására e dolgozatban.

Az alkalmazás elkészítéséhez a Microsoft .NET Framework 3.5-ös[1] verziójú szoftverfejlesztői keretrendszert és Microsoft Visual Studio 2008[2] fejlesztőkörnyezetet használtam ASP.NET[3] webfejlesztési technológiával és C#[4] programozási nyelvvel. Az étterem Oracle 10g Express Edition[5] adatbázist applikál, amely könnyen összekapcsolható a Visual Studioval az Oracle Development Tools[6] segítségével. Az adatbázis adatok könnyebb, használhatóbb elérését az Oracle Data Acces Components biztosítja.

„A .NET keretrendszert, ami a szintén Microsoftos COM+ futtatási környezetből fejlődött ki, 2000-ben jelentették be. A .NET igazi népszerűsége 2005-ben alapozódott meg a 2.0-ás verzió publikálásával. A dolgozat írásának idejében a verziószám 4.0-nál tart. A .NET definíciója: „A .NET különböző méretű és jellegű szoftverek fejlesztéséhez alkalmazható előregyártott infrastruktúra. Új szemléletet vezet be a fejlesztéshez. Több puszta specifikációnál, része egy termékhalmaz is, amely arra szolgál, hogy a mobil eszközöktől az asztali számítógépeken át a legnagyobb rendszerekig egységesen, konzisztens módon és gyorsan lehessen szoftvert fejleszteni, beleértve a webes és a nem webes megoldásokat egyaránt. A .NET-hez három csomag tartozik: a .NET Framework, a Visual.NET és a háttérkiszolgálók.”[7]

Választásom azért esett e keretrendszer használatára, mert fejlesztése folyamatosan folyik, a Microsoft nagyfokú támogatást biztosít hozzá, beleértve ebbe a Microsoft Developer

[1] http://www.microsoft.com/downloads/en/details.aspx?FamilyId=333325fd-ae52-4e35-b531-508d977d32a6&displaylang=en
[2] http://msdn.microsoft.com/en-us/vstudio/default.aspx
[3] http://www.asp.net/
[4] http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx
[5] http://www.oracle.com/technetwork/database/express-edition/overview/index.html
[6] http://www.oracle.com/technetwork/developer-tools/visual-studio/overview/index.html
[7] forrás: A .NET Framework és programozása (SZAK kiadó Kft. 2004, A .NET definíciója)

Network[1] (MSDN) működését, valamint a bőséges, példákkal teli online dokumentációkhoz való hozzáférhetőség biztosítását, továbbá rendkívüli módon használható gyors alkalmazásfejlesztésre. Sajnálatos módon, mint minden keretrendszerben, a .NET-ben is időről-időre felfedeznek kisebb nagyobb bugokat, amelyek néha furcsa működést eredményezhetnek. Például 2010 szeptemberében egy kritikus biztonsági hibát véltek felfedezni benne, amit azóta tudomásom szerint sikerült kijavítaniuk.

Online éttermem elkészítéséhez egy igen erős integrált fejlesztői környezetet, a Visual Studio 2008-at választottam, amely számtalan funkciót tartalmaz annak érdekében, hogy a fejlesztő minél kényelmesebben, gyorsabban tudjon haladni a munkájával. Az első Visual Studio-t 1997-ben adták ki és azóta több állomáson keresztül jutott el napjainkig, a 2010-es verzióig. A diplomamunkámban általam is használt C# nyelv csak a 2002-es verziótól volt elérhető. A C# egy Java és C++ alapokra épülő objektum orientált programozási nyelv, melynek kifejlesztését Anders Hejlsberg vezette. A nyelv 2001-ben ECMA[2], majd 2 évre rá ISO[3] szabvány lett. A szintaktikája egyértelműen a Java-ra hasonlít. Az online étterem megvalósításához ASP.NET webalkalmazás-keretrendszert használtam, amely dinamikus weboldalak, webalkalmazások építését teszi lehetővé.

A fent leírt eszközökkel, szoftverekkel egyik jóbarátom hatására kezdtem el ismerkedni és meglátva, hogy mennyire sokoldalú és szerteágazó dolgokat lehet megvalósítani velük, az érdeklődésem a webes alkalmazásfejlesztés irányába tolódott el. Az elején kisebb feladatokon keresztül kezdtem megismerni a .NET-et, majd online dokumentációk olvasása révén próbáltam egyre jobban elmélyedni a témában. A .NET teljes megismerésére, véleményem szerint, több évtizedre lenne szüksége egy embernek, azért is, mert szüntelen fejlődés jellemzi az informatikát, valamint számtalan technológiát, technikát támogat.

A diplomamunka célja, hogy egy konkrét, általam tervezett és implementált webshop alapú webalkalmazás segítségével bemutassam egy ilyen típusú alkalmazás rejtelmeit, hogyan helyezkedik el az alkalmazás mögött egy erős, széles körben preferált adatbázis, valamint azt, hogy a Visual Studio 2008 hogyan nyújt mindehhez megfelelő segítséget, megvalósítási lehetőségeket a programozónak. A dolgozatom teljes megértéséhez szükséges alapszintű

[1] http://msdn.microsoft.com/hu-hu/default%28en-us%29.aspx
[2] http://www.ecma-international.org/publications/standards/Ecma-334.htm
[3] http://www.iso.org/iso/catalogue_detail.htm?csnumber=36768

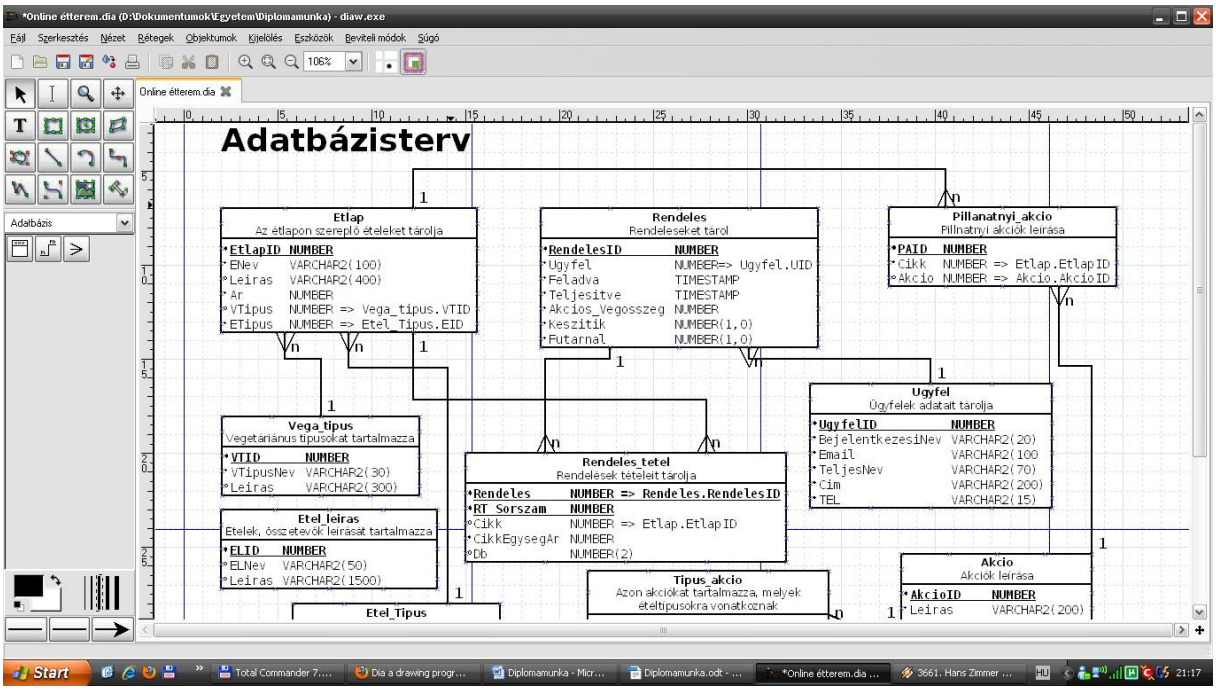
ismeret a következőkből: C#, ASP.NET, SQL, PL/SQL. Feltételezem, hogy az olvasó rendelkezik alapszintű ismerettel az objektum orientált paradigmáról.

A fejlesztés során használt eszközök rövid bemutatása

Ebben a fejezetben az alkalmazás során igénybevett eszközöket szeretném röviden bemutatni. Egy program vagy projekt elkészítését nagymértékben megkönnyíti a különböző erre specializált szoftverek és egyéb termékek használata. Legyen szó tervező-, implementáló-, vagy adatbáziskezelő-eszközről napjainkban egyre több áll rendelkezésre és biztosan találunk az igényeinknek megfelelőt.

Dia diagramkészítő szoftver

A Dia[1] egy olyan ingyenes diagramszerkesztő szoftver, amely a leggyakrabban használt diagramok (pl.: UML, Cisco), tervek elkészítéséhez nyújt segítséget. A program Drag&Drop alapú, tehát az oldalsávról az egérrel megragadva a munkaterületre lehet pakolni a komponenseket és összekötni azokat a megfelelő kapcsolatokat szimbolizáló elemekkel. Többfajta exportálási formátumot támogat, amivel például jpeg-be is menthetjük az ábráinkat. Az étterem adatbázisának megtervezéséhez a Dia 0.97.1-es verzióját használtam.

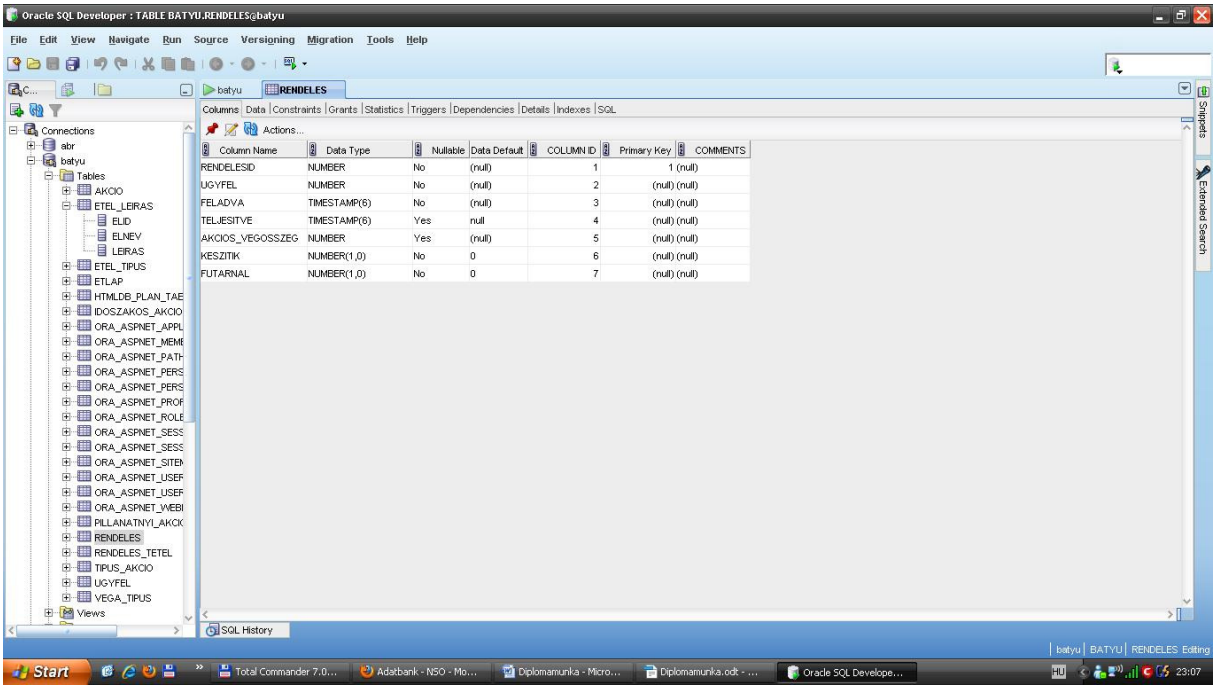


1. ábra: a Dia 0.97.1 képernyőképe

[1] http://projects.gnome.org/dia/

Oracle SQL Developer

Az Oracle SQL Developer[1] egy ingyenes adatbázis-fejlesztő eszköz, mellyel böngészhetünk az adatbázis-objektumaink között, SQL-parancsokat, scripteket, PL/SQL utasításokat futtathatunk, riportokat készíthetünk, kapcsolódhatunk egyéb adatbázisokhoz és akár migrálhatjuk is őket. Az online étterem készítésekor ezt az eszközt használtam a programba ágyazott SQL-parancsok kipróbálására, illetve az adatbázis exportálására biztonsági mentésekkor.

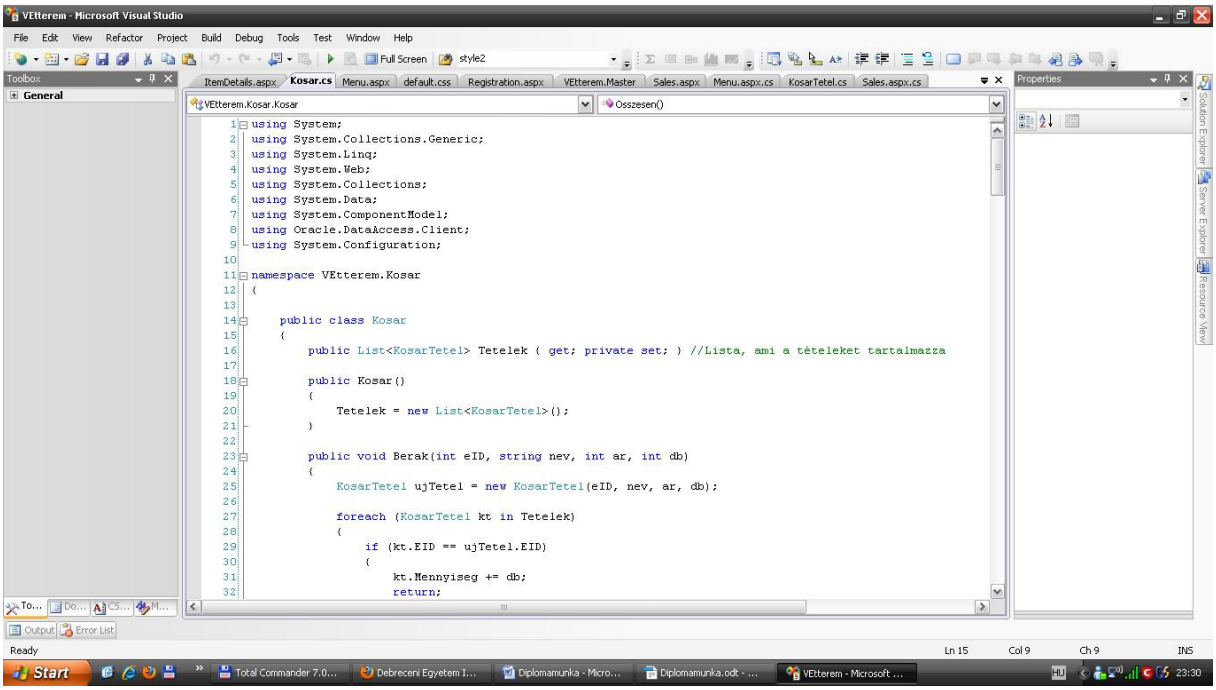


2. ábra: Oracle SQL Developer

Microsoft Visual Studio 2008

A Microsoft Visual Studio 2008 egy integrált fejlesztői környezet, amely széleskörű támogatást biztosít a programozónak pl. egy webalkalmazás írásakor. A fejlesztői tevékenységem egészét ebben az IDE[2]-ben folytattam.

[1] http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html
[2] Integrated Development Environment



3. ábra: Visual Studio 2008

Egyéb eszközök

Diplomamunkám készítése során használtam még az Oracle Express Edition webes kezelőfelületét. Ahogy azt a bevezetőben már említettem az általam írt alkalmazás Oracle Express Edition adatbázist használ, melyhez egy igen jól használható webes adminisztrációs felületet biztosítanak a gyártók. Segítségével böngészhetünk az adatbázis-objektumaink között, SQL-parancsokat, scripteket futtathatunk, vagy akár saját alkalmazásokat is építhetünk.

Ezen túl a tervezési folyamatkor készült USE-CASE diagramot a Software Ideas Modeller[1] szoftverrel készítettem el.

Itt szeretném még megemlíteni a Notepad++[2] és a Gimp[3] eszközöket is, az előbbi egy szövegszerkesztő, az utóbbi egy képszerkesztő, képmanipuláló szoftver, melyek szintén segítették munkámat.

[1] http://www.softwareideas.net/
[2] http://notepad-plus-plus.org/
[3] http://www.gimp.hu/

A Vétterem, mint webes alkalmazás

A webalkalmazás célja

A diplomamunkám mellékleteként leadott webalkalmazás arra hivatott, hogy kényelmesebbé tegye az ételrendelést, mind a rendelést feladó szempontjából, mind a pultos, vagy rendelésfelvevő szempontjából. A Vétterem, ugyanis ezzel a névvel láttam el az online éttermet vegetáriánus ételkínálata végett, lehetővé teszi, hogy egy böngésző segítségével kényelmesen, otthonról tudjunk ételt, italt rendelni egy webshop-szerű felületet használva. A program segítségével végig tudja követni a pultos, hogy ki, mikor, milyen rendeléseket adott fel és milyen státuszban van a megrendelés. Ezen túlmenően különböző típusú akciók felvitelére is lehetőséget biztosítok számára. E feladatok megoldására egy egyszerű, könnyen kezelhető felület áll rendelkezésére. Elsődlegesen egy online étterem célja az, hogy ne kelljen az étkezni kívánó embernek kimozdulnia otthonról egy-egy jó fogás reményében, sőt még a telefont se kelljen felvennie, hogy rendeljen valamit, hanem egy böngészővel és internetkapcsolattal tudjon ételhez, italhoz jutni. Kényelmesebb abból az aspektusból, hogy nem érzi magát sürgetve az ügyfél, hogy az étlapról válasszon, mint egy étteremben, hanem ráérősen átnézheti mit szeretne rendelni, belerakhat a virtuális kosárba fogásokat, kivehet belőle és végül véglegesítheti a rendelési igényét. Másrészt folyamatosan figyelemmel követheti a fizetendő összeg alakulását a kosár megtekintésével.
Itt szeretném kijelenteni, hogy az általam írt alkalmazás egy fiktív étterem fiktív rendelési folyamatát hivatott modellezni, bár megpróbálja életszerűen megvalósítani azt.

Igények az alkalmazással szemben

Hogy egy webshop-típusú webalkalmazás életszerű, működőképes legyen, néhány igénynek meg kell felelnie. A következőkben konkrétan a diplomamunkámban tárgyalt alkalmazásra levetítve fogom az elvárásokat bemutatni.

Első és egyik legfontosabb tulajdonsága a Vétteremnek, hogy lehetősége van benne regisztrálnia magát az ügyfélnek, sőt, ha rendelni akar, akkor ezt kötelezően meg kell tennie. Regisztrációkor egy online regisztrációs űrlapot kell kitöltenie a felhasználónak, ahol a következő kötelező adatokat kell megadnia: bejelentkezési név (egyedi), jelszó, e-mail cím (egyedi), teljes név, cím és telefonszám. A sikeres regisztráció után a felhasználó képes bejelentkezni, illetve kijelentkezni a weboldalra, illetve a weboldalról. Az étterem alkalmazásnak tudnia kell kezelni azt is, hogy bizonyos opciók csak a bejelentkezett, azaz már regisztrált ügyfelek számára lehessen csak elérhető. Tehát biztosítanunk kell az authentikációt, valamint a jogosultságkezelést is. Ezen túlmenőleg kétféle felhasználót szeretnénk kezelni. Az egyik fajta felhasználónk a user, azaz az ügyfél, aki rendeléseket tud feladni és böngészni tud a kínálatban vagy az oldal egyéb, számára megengedett információt tartalmazó oldalakon. A másik típusú felhasználó a pultos elnevezést kapta, aki bejelentkezés után a rendelésekkel kapcsolatos információkat látja és különböző állapotokba viheti át a rendeléseket, illetve akciókat tud definiálni és hozzárendelni - pl. fogásokhoz - akciótípustól függően. Tehát a webalkalmazásnak szerepköröket is tudnia kell kezelni.

A szerepköröket és az authentikációt, valamint a weboldal tagságszolgáltatását az OracleMembershipProvider és az OracleRoleProvider szolgáltatók valósítják meg.

Naplózás

A rendszer, ahogy az a gyakorlatban igen jellemző, egy naplófájlt vezet az alkalmazásban történt fontosabb eseményekről a későbbi visszatekintések, esetleges statisztikák miatt. A Vétterem naplójában a következő események jelennek meg:

- Regisztráció
- Bejelentkezés
- Kijelentkezés
- Rendelés feladások/teljesítések/törlések
- Akciók felvétele/törlése
- Hibák

A 'hibák' bejegyzésen kívül minden naplóbejegyzés tartalmazza az esemény dátumát, idejét, az esemény kiváltójának bejelentkezési nevét, és IP-címét.

Egy rendelés folyamatának leírása

A rendelési folyamat alapvetően az étlap megtekintésével kezdődik. De kezdjük most egy kicsit másképp az áttekintését. Első lépés a rendeléshez, hogy az ügyfél bejelentkezik a felhasználónevével és jelszavával. Ezután az étlap böngészése közben, kiválasztja a neki megfelelő fogásokat és hozzáadja a kosárhoz a kívánt darabszám beállításával. Miután leellenőrizte kosarának tartalmát, véglegesíti a rendelését. Ebben a pillanatban az adatbázisban letárolódnak a megfelelő adatok, melyeket felhasználva a program a pultos felé jelzi, hogy bejött egy új megrendelés. A pultos szóban jelzi a rendelést a szakácsnak, és a webes felület segítségével a rendszer felé a 'készítik' állapot beállításával. Mihelyt a szakács elkészül a fogással, odaadja azt a pultosnak. A pultos ezután továbbítja a futárnak, aki elindul vele a megadott címre. Ekkor jelzi a rendszer felé a pultos, hogy 'futárnál' van a rendelés. Ha a futár visszaérkezett, akkor már teljesítettnek tekinthető a rendelés, ezután tehát jelezheti a rendszer felé a teljesítés tényét. Tehát egy rendelés 4-féle állapotban lehet: feladva (ilyen állapotúnak akkor tekintjük, ha a további három állapot egyikében sincs), készítik, futárnál, teljesítve. Lehetőség van még továbbá a rendelések törlésére is.

Az alkalmazás tervezése

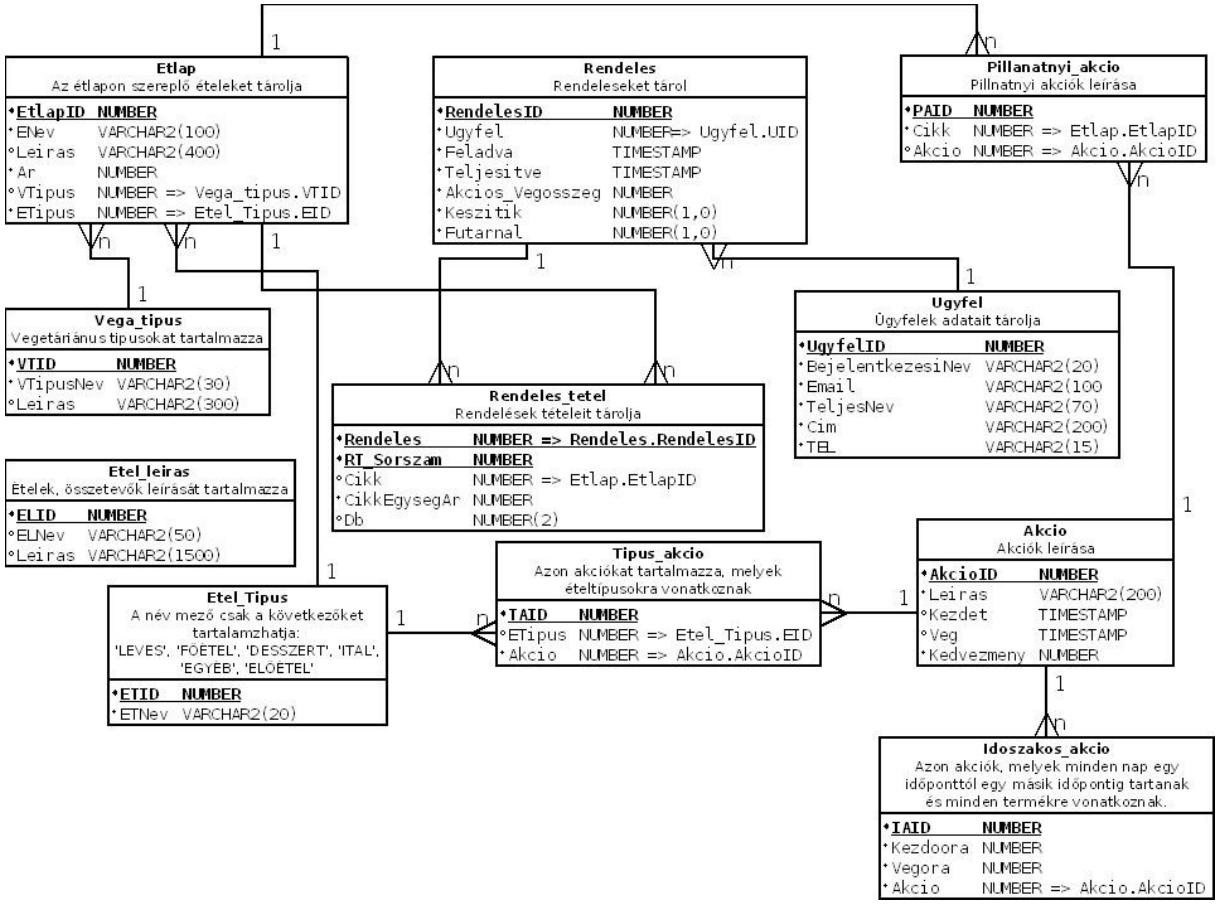
A szoftverfejlesztési, vagy rendszerfejlesztési folyamat egyik nagyon fontos szakasza a tervezés, amikor át kell gondolnunk, mit kell tudnia, milyen követelményeket kell teljesítenie az implementálni kívánt rendszernek. A tervezési folyamatok közül is véleményem szerint az egyik leglényegesebb az alkalmazás adatbázisának átgondolása, megtervezése, azaz ha egy gyorsan futó programot szeretnénk előállítani, akkor alaposan át kell gondolnunk a felépítését, a táblákat, azok mezőit és a köztük lévő kapcsolatokat. Egy olyan adatbázisban, ahol már több száz tábla van és ezekhez több tízezer rekord tartozik, már nagyon fontos, hogy ésszerűen alakítsuk ki az adatbázis, mert egyes rekordleválogatások ideje megnőhet, és ezáltal zavaróan lelassulhat az alkalmazás futása. Ez különösen igaz egy webes program esetén.

Egyetemi tanulmányaim alatt az Oracle adatbázis-kezelő rendszerrel ismerkedtem leginkább. Ebből az okból kifolyólag, valamint egyéb tapasztalataim alapján döntöttem úgy, hogy a diplomamunkám elkészítésekor is ezt a terméket fogom használni.

Az ASP.NET technológia jellege miatt kevés osztállyal dolgozom, hiszen Web Forms[1] filozófia szerint fejlesztek, azaz a kódok nagy részét az oldalak ”codebehind” részébe helyezem el.

Az adatbázis terve

Az adatbázis felépítését a következő ábra mutatja be:



4. ábra: adatbázisterv

Ahogyan az ábrán is látható létezik egy *Etlap* tábla, amely az étterem kínálatát foglalja magában. Minden egyes étlap rekord, azaz fogás besorolható egy vegetáriánus típusba. E típusok egy külön táblában tárolódnak; a következő vegetáriánus-típusok rekordok találhatóak benne: lakto, ovo, ovo-lakto, vegán és egyéb. Az egyéb érték az italok és a besorolhatatlan ételek miatt került bele. Az *Etlap* tábla hivatkozik egy másik táblára, ami az étel típusát

[1] http://www.asp.net/web-forms

mutatja meg. E tábla neve az *Etel_tipus* nevet kapta és a következő ételtípus-rekordok jelenhetnek meg benne: leves, főétel, desszert, ital, egyéb, előétel. Található továbbá egy független tábla az adatbázisban, az *Etel_leiras*, ami az ételek és összetevők leírását tartalmazza. Létezik még egy *Ugyfel* nevű tábla is, ami regisztrált ügyfelek fontosabb, a kiszállításhoz is szükséges adatait tartalmazza. Megtalálható még egy Rendelés tábla is amely értelemszerűen a rendeléseket tárolja. Hivatkozik az *Ugyfel* táblára, azaz megmutatja, melyik ügyfélhez tartozik az adott rendelés. Ezentúl olyan információkat is hordoz, hogy a rendelés mikor lett feladva, mikor lett teljesítve és e két esemény között épp milyen státuszban van. A státuszok a következők lehetnek:

- feladva: ebben az állapotban van az adott rendelés, ha a *Teljesitve* mező nullértékű és a *Keszitik* és *Futarnal* mezők értéke 0.
- készítik: ezen állapotban van egy rendelés, ha a *Teljesitve* mező nullértékű és a *Keszitik* 1, a *Futarnal* mező értéke pedig 0.
- futárnál: ilyen az állapota a rendelésnek, ha a *Teljesitve* mező nullértékű és a *Keszitik* 0, a *Futarnal* mező értéke pedig 1.
- teljesítve: a rendelés állapota ez, ha a *Teljesítve* mező nem null.

Minden rendelés egy vagy több rendelési tételből áll. Ezen tételeket tartalmazza a *Rendeles_tetel* tábla, ami hivatkozik a *Rendeles* táblára, így látható melyik rendeléshez tartozik az adott tétel.

Az akciók tárolására hivatott a többi tábla. Az *Akcio* tábla egy általános akció tábla ezért olyan mezőket tartalmaz, amik az étterem által támogatott három fajta akciótípus, mindegyikéhez szükséges. Ezek az adatok a következők: az akció leírása, kezdete, vége és a kedvezmény százaléka. A *Tipus_akcio* tábla megmutatja, hogy milyen típusú ételre vonatkozik az akció, azaz hivatkozik az *Etel_Tipus* táblára. Ide kerül az adat, ha pl. az összes levesre szeretnénk kedvezményt adni. Az *Idoszakos_akcio* tábla azt az órában értendő időintervallumot mutatja meg, amiben érvényes a hivatkozott akció. Itt lesz található az adat, ha például azt mondjuk, hogy este 8 és 10 óra között, minden termékre szeretnénk kedvezményt adni. A *Pillanatnyi_akcio* táblában egy konkrét fogásra vonatkozó akciót láthatjuk.

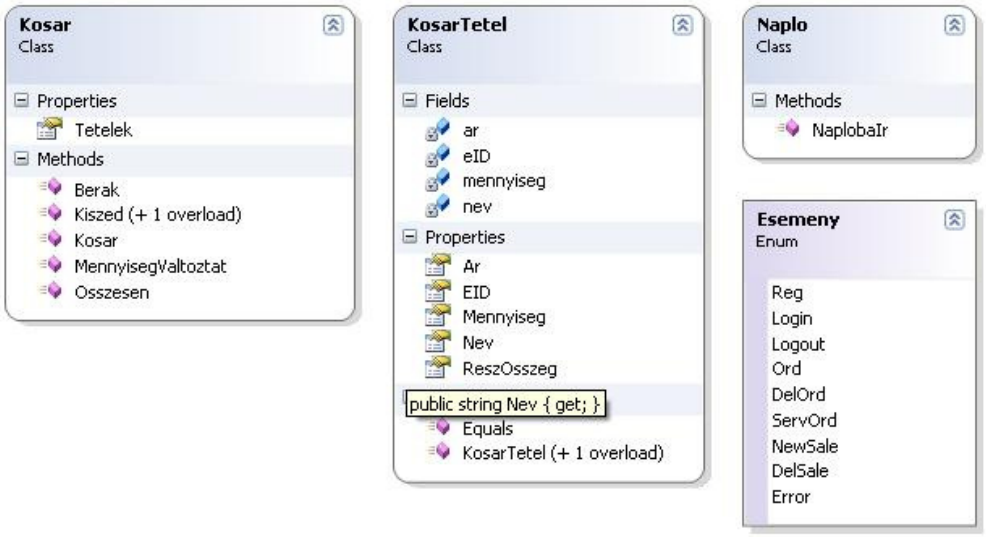
A fent bemutatott adatbázis-táblákon kívül az alkalmazásom egyéb táblákat is használ. A Visual Studio 2008 és az MSSQL Server 2008[1] az ASP.NET alkalmazásokhoz biztosít egy SqlMembershipProvider-t, mint tagságszolgáltatót és egy SqlRoleProvider-t, mint szerepkör-szolgáltatót. Azonban én Oracle adatbázist használok, ezért az előbb említett szolgáltatók Oracle-ös megfelelőit alkalmazom a Vétteremben, azaz az OracleRoleProvider-t és az OracleMembershipProvider-t. Ehhez első lépésben le kellett töltenem az ODAC[2]-ot (Oracle Data Access Components) az Oracle weblapjáról és telepítenem kellett az Oracle Development Tools-t a Visual Studio-ba. Ezután az Oracle által biztosított SQL-script lefuttatása után létrejött adatbázis-objektumok lehetővé teszik a szolgáltatók működését. Ahhoz hogy a webes alkalmazás tudja, hogy e szolgáltatókat szeretnénk használni a web.config nevű fájlba jeleznünk kell a megfelelő szintaktikával. Az OracleMembershipProvider és OracleRoleProvider táblái közül a következőket szeretném kiemelni:

- ORA_ASPNET_MEMBERSHIP: ebben a táblában a tagsággal kapcsolatos információk találhatóak, úgy mint pl. a felhasználók azonosítója, jelszava, e-mail címe, és a bejelentkezéssel kapcsolatos biztonsági információk (pl.:utolsó bejelentkezés időpontja, sikertelen bejelentkezések száma, stb.)
- ORA_ASPNET_ROLES: Itt láthatjuk a szerepkörökkel kapcsolatos információkat.
- ORA_ASPNET_USERSINROLES: Egy kapcsolótábla, amely megmutatja, hogy az egyes felhasználók, milyen szerepkörökkel rendelkeznek.
- ORA_ASPNET_USERS

Osztálydiagram

Egy objektumorientált webes alkalmazás tervezésekor érdemes osztálydiagramot készíteni. Az én alkalmazásomban a jellegéből és a technológia sajátosságai miatt mindössze három osztály jelenik meg. Az osztálydiagram a következőképpen néz ki:

[1] http://www.microsoft.com/sqlserver/en/us/default.aspx
[2] http://www.oracle.com/technetwork/developer-tools/visual-studio/downloads/index.html



5. ábra: Osztálydiagram

Az osztályok metódusainak és adattagjainak leírása

A *Kosar* osztály a vásárlásnál használatos kosár szerepét hivatott modellezni. Egyetlen property-vel rendelkezik, amely a *Tetelek* nevet kapta és ez tulajdonképpen egy *KosarTetel* objektumpéldányokból álló lista, azaz a kosárba több tételt lehet ”belepakolni”. A következő metódusok használhatóak a *Kosar* osztály objektumpéldányain:

- Berak(int eID, string nev, int ar, int db): Berakja az *eID EtlapID*-val rendelkező fogást a kosárba az aktuális árral és berakott mennyiséggel.
- MennyisegValtoztat(int eID, int db): Egy konkrét fogás kosárban lévő mennyiségét változtatja meg.
- Kiszed() és Kiszed(int eID): A paraméter nélküli változat kiüríti a kosarat, a paraméteres változat pedig kiszedi az *eID EtlapID*-jű fogást a kosárból
- Osszesen(): Kiszámolja a végösszeget az akciók figyelembe vétele nélkül.
- OsszesenAkc(): Kiszámolja a végösszeget a különböző akciók közül pontosan azt az egyet kiválasztva, amely a legnagyobb kedvezményt adja.

A *KosarTetel* osztály a Vétterem fogásait modellezi, úgy hogy kosárelemként tekint rá. Az osztály mezői a következők: *ar* (a tétel ára), *eID* (a tétel ID-ja), *mennyiseg*, (egy tétel konkrét kosárban lévő mennyisége), *nev* (a tétel neve). Az osztály property-i az előbb felsorolt

mezőkből adódnak, továbbá létezik még egy, melynek ReszOsszeg a neve, és egy kosártétel részösszegét számolja ki egyszerűen úgy, hogy megszorozza az árát a mennyiségével. Az Equals metódust felüldefiniálja az osztály a következőképpen: összehasonlítja két fogás ID-ját és ez alapján veszi azonosnak vagy különbözőnek azokat.

A *Naplo* osztály a webalkalmazásban használt naplózást segíti. Tartalmaz egy enumerációt, ami felsorolja a naplóba írást kiváltó eseményeket. Az enumeráció elemei a következőben felsoroltak: *Reg* (regisztráció), *Login* (bejelentkezés), *Logout* (kilépés), *Ord* (rendelés feladása), *DelOrd* (rendelés törlése), *ServOrd* (rendelés teljesítése), *NewSale* (új akció felvétele), *DelSale* (akció törlése), *Error* (hiba).

Az osztály egyetlen, naplóba író metódusa:

- NaplobaIr(string nev, string ip, Esemeny e): A naplóállományba írja a megfelelő bejegyzést a paraméterekként megkapott adatok felhasználásával.

Az alkalmazás látványterve

A diplomamunka mellékleteként leadott Vétterem alkalmazás látványterve a technológia sajátosságai miatt közvetlenül a Visual Studio 2008-cal készült úgymond fejlesztés előtt és közben. A fejlesztőeszköz igen kényelmes és praktikus lehetőségeket biztosít a vizuális elemek előállítására és módosítására. A Manage Styles panelen lehet stíluslapokat menedzselni. Itt különböző "beszédes" vizuális elemek mutatják, hogy az aktív aspx oldalon milyen stílusok vannak használva, definiálva. Más szín jelöli a különböző típusú komponenseket, így könnyebben kiigazodhatunk. A másik oldalpanel, ami segíthet nekünk, a CSS Properties elnevezést kapta, ahol a kihelyezett vizuális controlok CSS tulajdonságait tudjuk manipulálni és figyelemmel követni. A Vétterem design-jáért egy design.css nevű stíluslap felel, amelyben definiálva van bizonyos elemek kinézete. A Visual Studio-ba integrált stíluskezelő eszközök hatékonyak, viszont megfelelő designer ismeretek nélkül nem tudunk kimagasló látványt elérni.

Az ASP.NET webalkalmazás felépítése Visual Studio 2008 fejlesztőeszközzel

Ahogy azt már említettem, a Vétterem Visual Studio 2008 (továbbiakban VS) fejlesztői eszköz segítségével jött létre. Egy webalkalmazás létrehozásához a File->New Project... menün belűl oldalt a Visual C#-ból lenyitva a Web template-ekben böngészve választható ki az ASP.NET Web Application template. Itt megadhatjuk, hogy hova szeretnénk menteni a projektünket. A VS létrehoz az új projektünknek egy könyvtárszerkezetet a következő almappákkal: Properties, References, App_Data. A References mappában az alkalmazás által hivatkozott referenciákat, azaz a névterek hivatkozásait helyezi el a program. Az App_Data mappába az adatbázisaink fájljai kerülnek, ha pl. MSSQL adatbázis-szervert használunk. Ezeken kívül létrejön még automatikusan egy WebForm1.aspx ASP oldal és a hozzá tartozó "codebehind" állomány, melynek neve WebForm1.aspx.cs. Azért jó, hogy két állomány tartozik egy oldalhoz, mert így könnyedén külön tudjuk választani a designt a működési logikától. Valamint láthatunk a projektben egy web.config nevű fájlt, amelyben az alkalmazás webes beállításait tudjuk manipulálni.

Mi magunk is hozhatunk létre mappákat, almappákat ezzel rendszerezve oldalainkat. A mappák létrehozása ezen túl abban az esetben is igen hasznos, ha különböző jogosultságú szinteket, eléréseket szeretnénk létrehozni az alkalmazásunkra nézve. Ekkor könyvtáranként, azaz csoportosan is tudjuk korlátozni az oldalakhoz való hozzáférhetőséget.

Továbbá hozzáadhatjuk még pl. az App_Themes mappát a projektünkhöz, amibe értelemszerűen az oldalakhoz tartozó designnel kapcsolatos témák kerülhetnek majd. Az ilyen típusú mappákat ASP.NET Folder-eknek hívják és a következőképpen tudjuk a projektünkhöz hozzáadni: Jobb egérgomb a Solution Explorerben Add..., Add ASP.NET Folder, és itt kiválaszthatjuk a megfelelő mappát.

Az App_Code mappába a saját osztályaink, névtereink forrásfájljait helyezhetjük el.

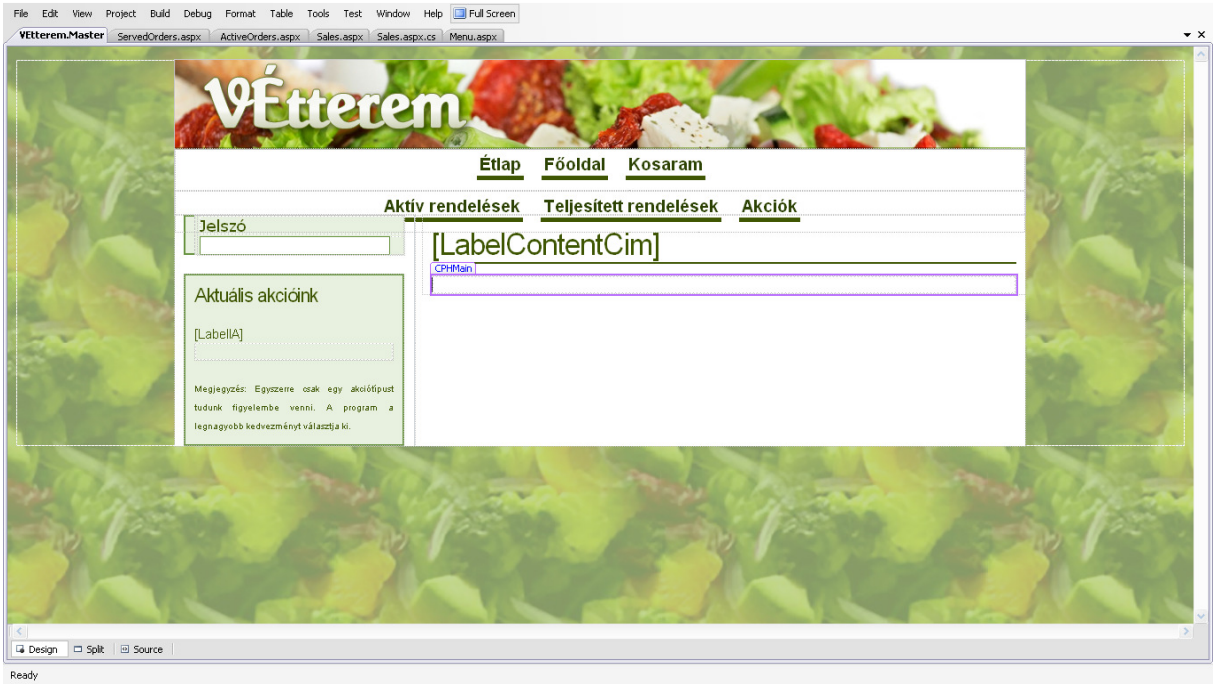
A Global.asax

A Global.asax állományban alkalmazás és session szintű események bekövetkezte esetén végrehajtandó kódokat helyezhetünk el. Segítségével történhet például a látogatók számolása egy weboldalon. Néhány metódusa pl. Session_Start, Session_End, előbbi egy session indulásakor, utóbbi akkor fut le, ha egy session véget ér.

Mesteroldal az ASP.NET-ben

A mesteroldal nagyon hatékony eszköz többek között arra, hogy egységes kinézetet tudjunk kölcsönözni az alkalmazásunk összes oldalának. Ezentúl alkalmas még arra is, hogy olyan kódot implementáljunk benne, ami több oldalnál azonos. Kiterjesztése .master, bár ugyanúgy asp oldal. A mesteroldalon ki tudunk alakítani egy fix kinézetet ASP controlok segítségével, majd a ContentPlaceHolder (helyőrző) controlt használva megjelölhetjük azokat a területeket, ahova később a dinamikus részeket szeretnénk elhelyezni a leszármazott oldalakban. Lehetséges mesteroldalból hierarchikusan további mesteroldal(aka)t származtatni. Tegyük fel, hogy létrehozunk egy masterpage-t és kirakunk rá egy helyőrzőt. Ekkor, ha egy olyan formot szeretnénk kreálni, ami a mesteroldalból származik, akkor ki kell választatnunk a nekünk szükséges mesteroldalt is. Ezután a VS Design nézetére rápillantva azt tapasztaljuk, hogy csak a masterpage-n létrehozott helyőrzőkben tudunk tevékenykedni, az "ős" többi részét nem tudjuk módosítani erről a formról. Ezt a stop jellé alakult kurzor jelzi számunkra. Kód szinten úgy kapcsolódik össze a helyőrző és tartalom, hogy, minden helyőrzőnek meg kell adnunk egy ID-t, mely azonosítóra tudunk hivatkozni a tartalmazó oldal Content tagjének a ContentPlaceHolderID tulajdonságát ráállítva.

A Vétterem mesteroldalának neve Vetterem.Master, és a következőképpen néz ki:



6. ábra: A Vétterem mesteroldala

Az egész Vetterem.Master egy div html tag-ben helyezkedik el és további megfelelően elhelyezett div tagek, panelek tagolják a tartalmakat. Középen felül egy menü található az oldalak közötti navigáció érdekében. Bal oldalt láthatunk egy bejelentkezési felületet, vagy egy kijelentkezést biztosító linkgombot attól függően, hogy be vagyunk-e jelentkezve, vagy sem. A bejelentkezés doboz alatt találhatjuk még az aktuális akciókkal kapcsolatos tudnivalókat. Mivel mind a menüt, mind a bejelentkezési/kijelentkezési tartalmat, mind az akciókat érdemes úgy megjeleníteni, hogy a felhasználó mindig láthassa, ezért ezeket a komponenseket az alkalmazás mesteroldalára helyeztem el és ezután minden oldalam ezen masterpage-t használja. A mesteroldal közepén található az egyetlen helyőrző, amibe a dinamikus tartalom kerül majd.

A web.config fájl az ASP.NET webalkalmazásokban

Egy ASP.NET webalkalmazás fizika elhelyezkedésének gyökerében található egy web.config fájl, amelyben széleskörű beállításokat végezhetünk az alkalmazásra vonatkozóan. Ez a fájl tulajdonképpen egy xml dokumentum, melyben különböző szekciók találhatóak, amelyek csoportosítják a különböző beállításokat. Található benne egy olyan szekció ahol az

ún. connectiontringek-et lehet megadni. A connectionstringek arra valók, hogy pontosan leírjanak egy adatbáziskapcsolatot. Miután ezt a szekciót beírtuk a konfigurációs állományba, a program többi részéből egy név segítségével tudunk rá hivatkozni, ahogyan azt a következő, Vétteremben előforduló sorok is mutatják:

```
oc.ConnectionString = ConfigurationManager.ConnectionStrings["VCS"].ConnectionString;
vagy
<asp:SqlDataSource ... ConnectionString="<%$ ConnectionStrings:VCS %>"… </asp:SqlDataSource>
```

Az első sor egy OracleConnection objektumpéldány ConnectionString property-jét állítja be a ”VCS” nevű ConnectionString-re a ConfigurationManager oszály segítségével. A második sorban azt láthatjuk, hogy egy SqlDataSource szintén a ConnectionStrings tulajdonságát állítjuk be.

A web.config-ban állíthatjuk be továbbá a szerepkör-menedzsert, a tagságszolgáltatót, az authentikációs módot és a fordító tulajdonságait. A webalkalmazások nagy százalékában mappákba, almappákba vannak sorolva a logikailag összefüggő formok és egyéb fájlok. Elhelyezhetünk egy almappába is web.config állományt, de azt tudnunk kell, hogy örökli az alkalmazás gyökerében lévő azonos nevű konfigurációs fájl tartalmát, ezáltal biztosítva azt, hogy az alkönyvtárban írt fájlba a lehető legkevesebbet kelljen írnunk. A Vétteremben például a Pultos könyvtár rendszerezi be azon oldalakat, amelyeket csak a pultos láthat. Az almappában lévő oldalakat csak a megfelelő szerepkörben lévő felhasználók láthatják, így küszöböljük ki, hogy egy user hozzáférhessen a pultosok felületéhez. Az almappában lévő web.config tartalma a következő:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <system.web>
        <authorization>
            <allow roles="pultos" />
            <deny roles="ugyfel" />
            <deny users="?" />
        </authorization>
    </system.web>
</configuration>
```

Látható, hogy a pultosok számára engedélyezve, míg a userek számára tiltva vannak az aktuális mappában elhelyezett elemek. A userek számára tiltott oldalak a következők: ActiveOrders.aspx, ActiveOrdersDetails.aspx, Sales.aspx, ServedOrders.aspx, ServedOredersDetails.aspx. Természetesen az Anonymus felhasználók sem láthatják a tiltott oldalakat.

A Visual Studio 2008 felülete és eszközei

A Visual Studio 2008 egy kényelmes, jól kezelhető felületet biztosít az ASP.NET fejlesztéshez, amit magunknak is konfigurálhatunk.

Általában a felület bal oldalán található a Toolbox, ami az aktuálisan megnyitott lapra helyezhető controlokat, interfészeket, komponenseket, eszközöket tartalmazza. Innen drag&drop (fogd és vidd) módszerrel, vagy dupla kattintással tudjuk az oldalunkra helyezni a megfelelő elemeket. Az elemek kategóriákba vannak sorolva. Az Ajax Control Toolkit[1]-et (ACT) használtam még az alkalmazás felépítéséhez. Az ACT egy .NET Ajax keretrendszerre épülő nyílt forráskódú projekten belül készült. Például, amikor új időszakos akciót viszünk fel pultosként az alkalmazásba, akkor a validálásnál megjelenik szövegbuborékként a validáló üzenet. Ezt a ValidatorCalloutExtender segítségével valósítottam meg. Ezt a Toolkit-elemet össze kell kapcsolnunk egy validátorral és a validátor által generált üzenetet sima szöveg helyett egy bezárható szövegbuborékban láthatjuk.

Használható egy ún. Document Outline oldalpanel, ami a forráskódokban való könnyebb tájékozódást segíti oly módon, hogy fastruktúrába rendezi a forráskódot és megjeleníti azt. Egy-egy elemre kattintva az adott elem forrását kijelöli.

Található még egy Manage Styles és CSS Properties oldalsáv, amik a design létrehozásában, stíluslapok kezelésében lehetnek segítségünkre.

A Visual Studio ablak tetején a szokásos menüsor és ikonsor helyezkedik el alapértelmezésben. Különböző oldalsávokat húzhatunk még be, mint például a Solution Explorer, ahol a Solution-ünk Felépítését, fájlszerkezetét, referenciáit láthatjuk. Az egyik legfontosabb oldalsáv a Properties oldasáv. Ezen az ablakrészen jelennek meg a

[1] http://www.asp.net/ajaxlibrary/act.ashx

munkaterületen kiválasztott objektum tulajdonságai és bizonyos esetekben a hozzá tartozó konfigurálható események listája is. Kényelmes kezelhetőséget biztosít, hiszen egy táblázatos felületet használva szerkeszthetjük azokat és innen generálódnak a bejegyzések a kód megfelelő helyére. Ezentúl még érdemes megemlíteni a Server Explorer-t is, ahol a regisztrált szervereinket, adatbázisszerver kapcsolatainkat láthatjuk. Egy adatbázis-kapcsolatunkra kattintva egy faszerkezet jelenik meg.

A Visual Studio 2008-ba egy egyszerűbb adatbázis-kezelő rendszer modul van integrálva, mely alkalmas arra, hogy az adatbázis-objektumainkat manipuláljuk egy grafikus felületről. Táblák, nézetek, (Oracle adatbázis esetén) szekvenciák vagy akár triggerek létrehozásában is nagy támogatást nyújt vagy akár egy meglévő adatbázis-objektum kreáló SQL scipt-jét is legenerálhatjuk. Tehát ha nincs esetleg egyéb adatbázis-manipuláló szoftverünk, akkor az alapvető, vagy akár haladó műveleteket is képesek vagyunk megvalósítani ezzel az integrált felülettel.

Középen, a munkaterületen jelenik meg a munkánk. Három fajta nézetet támogat a VS2008: design, split és source. A design nézet "**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et ", azaz "amit látsz, azt kapod", elv alapú szerkesztő. Itt vizuális felületen láthatjuk, manipulálhatjuk az objektumokat. A source nézetben a forráskódot láthatjuk, szerkeszthetjük; a forrásszöveg könnyebb átláthatóságát színes kiemelésével segíti a fejlesztőeszköz. A split mód a 2005-ös verzióban még nem volt, viszont a 2008-ba már benne van. Ez a mód az ablakot szétosztva, egyszerre mutatja a design és source ablakot. Én személy szerint nagyon hasznosnak tartom ezt a módot, mert párhuzamosan látom a design-t és a forrást és például, ha a design részablakon rákattintok egy control-ra akkor a forrás részablakon kiemeli a hozzátartozó kódrészletet. Sajnos néha át kell kattintani az egyik részablakról a másik részablakra, hogy szinkronban legyen a két mód, ami sokszor igen kényelmetlen.

Egy másik igen kényelmes, napjainkban a legtöbb fejlesztői környezetbe integrált forráskód írást segítő eszköz az ún. Intellisense, amely automatikus kódkiegészítésre képes. Ezt úgy kell elképzelnünk, hogy elkezdünk gépelni a forráskód írásakor például egy hosszabb osztálynevet és az Intellisense felajánlja az eddig begépelt szöveg alapján a lehetséges szövegbefejezéseket, így nem kell beírnunk minden egyes betűt és gyorsabbá válhat a fejlesztés. Ezen kívül a kiegészítendő programozási eszköznévhez tartozó dokumentáció is

olvasható egy felugró panelben, vagy például ha egy metódust szeretnénk meghívni, akkor a láthatjuk a várt paramétereket és azok típusát. Gondoljuk el milyen hatékony eszköz az Intellisense például ha a következő osztálynevet sokszor szeretnénk begépelni a programunk írásakor: AttachedPropertyBrowsableWhenAttributePresentAttribute.

Mindent összevetve a Visual Studio 2008 egy igen erős fejlesztői támogatást nyújtó fejlesztőeszköz, amibe számtalan hasznos modul van integrálva. Véleményem szerint az egyik legjobb, leghatékonyabb eszköz a gyors alkalmazásfejlesztéshez. Viszonylag könnyen használható és gyorsan tanulható. Természetesen debug-olási lehetőséget is felkínál nekünk a VS2008, amivel futási időben, akár sorról sorra figyelemmel követhetjük az alakuló referenciákat, változóértékeket. Sajnos, mint minden fejlesztői környezetben, vannak benne hibák, melyek néha bosszantóak a fejlesztő számára. Gondolok itt arra például, hogy egy ASP formon egyik controlról egy másikra kattintva a properties oldalsáv nem frissül rendesen. Ezt a "bugot" úgy lehet kikerülni, hogy átváltjuk a nézetet source nézetre, majd vissza és ekkor már frissülni fog a properties oldalsáv.

Az Oracle adatbázis a webalkalmazás tükrében

Az Oracle cég termékei igen népszerűek, ennél fogva nem csoda, hogy a piaci részesedésük rendkívül nagy százalékértékű. Napjainkra a cégnek szinte minden fajta informatikai területet sikerült lefednie és széleskörű támogatást nyújt a felhasználóinak, ügyfeleinek és a fejlesztőknek.

"Az Oracle objektumrelációs adatbázis-kezelő rendszer, amely alapvetően relációs eszközöket tartalmaz és ezeket egészíti ki objektumorientált eszközökkel. Mint a relációs adatbázis-kezelő rendszerek általában, az Oracle is a szabványos SQL nyelvet használja az adatok kezelésére. Az Oracle az SQL:1999 szabványt támogatja. Az SQL tipikus negyedik generációs nyelv, amely deklaratív jellemzőkkel rendelkezik. Ez azt jelenti, hogy a nyelv parancsai segítségével leírjuk, hogy mit kell csinálni, de azt nem adjuk meg hogy hogyan. A parancs végrehajtásának részletei rejtettek maradnak."[1]

Az Oracle adatbázis fontosabb parancsai, jellemzői

Az Oracle adatbázisban a klasszikus objektumrelációs elemek, eszközök megtalálhatóak. Az adatbázist több fajta paranccsal manipulálhatjuk. Az adatbázis elsődlegesen táblákból épül fel, amelyek tárolják a szükséges adatokat. Egy adatbázistáblában egy egyed-előforduláshoz tartozó adatok rekordokba vannak rendezve. Egy rekord szemléletesen a tábla egy sorának tekinthető. Az Oracle az SQL szabványt támogatja, így ez alapján három fő utasításcsoportot tudunk elkülöníteni:

- DDL (Data Definition Language): magyarul adatdefiníciós nyelv, mellyel adatbázis objektumokat tudunk létrehozni, karbantartani, kezelni.

[1] forrás: Juhász István – Gábor András: PL/SQL-programozás (Panem kiadó 2002)

- DML (Data Manipulation Language): azaz adatmanipulációs nyelv, ezzel a csoporttal tudunk adatokat felvinni, törölni, frissíteni az adatbázisból, valamint ide tartozik még az adatok kinyerése is, azaz a lekérdezés is.
- DCL (Data Control Language): azaz adatvezérlő nyelv, amely segítségével többek között kezelhetjük a tranzakciókat, vagy a hozzáférési szabályokat, jogosultságokat szabályozhatjuk.

Az általam írt alkalmazás adatbázisa viszonylag kis méretű, csupán tizenegy tábla fordul elő benne. A táblákat egy-egy DDL utasítással hoztam létre, lássuk most az *Etlap* tábla létrehozásának SQL-parancsát:

```
CREATE table "ETLAP" (
"ETLAPID" NUMBER NOT NULL,
"ENEV" VARCHAR2(100) NOT NULL,
"LEIRAS" VARCHAR2(400),
"AR" NUMBER NOT NULL,
"VTIPUS" NUMBER NOT NULL,
"ETIPUS" NUMBER NOT NULL,
constraint "ETLAP_PK" primary key ("ETLAPID")
)
```

Tehát a fenti paranccsal létrehoztam a táblát, oly módon, hogy felsoroltam a mezőket, az egyes mezőkhöz tartozó típusokat és kiegészítő információkat. Az utolsó sorban láthatjuk egy megszorítás megadását. Ebben az esetben egy elsődleges kulcsot definiálunk a táblán, ami természetesen egyedi azonosítóként használatos, azaz minden egyes rekordnál különböző az értéke. Ezt nem "kézzel" generáljuk, hanem az Oracle egy eszközével, melynek neve szekvencia. Az *Etlap* táblához tartozó szekvencia létrehozása:

```
CREATE SEQUENCE "ETLAP_SEQ" START WITH 100 INCREMENT BY 10
MAXVALUE 10000
MINVALUE 100
```

A szekvenciák kényelmesen használhatók egyedi azonosítók generálásához. A fentre írt kód egy olyan szekvenciát hoz létre az adatbázis sémában, amely 100-tól indulva 10-esével növelhető és 10000-ig generál értékeket. A szekvencia objektumok egyedi, egész értékeket generálnak a létrehozásukkor leírt tulajdonságuk alapján. Ha az aktuálisan generált értékére vagyunk kíváncsiak, akkor a .CURRVAL, ha a soron következő értékre, akkor a .NEXTVAL oszlop lekérdezésével tehetjük azt meg.

Az Etlap tábla nem különül el a többi táblától, két táblához is kapcsolódik közvetlenül. Egyrészt az egyik kapcsolatból láthatjuk, hogy az adott rekordban tárolt fogás milyen vegetáriánus-típushoz tartozik, másrészt pedig azt, hogy milyen típusú étel. Az Oracle-ben ezt külső kulcsokkal (Foreign Key, idegen kulcs) tudjuk jelölni. Az *Etlap* táblához tartozó idegen kulcsok beállításának parancsa:

```
ALTER TABLE "ETLAP" ADD CONSTRAINT "ETLAP_FK_V"
FOREIGN KEY ("VTIPUS")
REFERENCES "VEGA_TIPUS" ("VTID")

ALTER TABLE "ETLAP" ADD CONSTRAINT "ETLAP_FK_E"
FOREIGN KEY ("ETIPUS")
REFERENCES "ETEL_TIPUS" ("ETID")
```

Az első parancs módosítja az *Etlap* táblát úgy, hogy hozzáad egy olyan külső kulcsot, amely a *Vtipus* mezőhöz tartozik és a *Vega_Tipus* nevű tábla VTID nevű mezőjére hivatkozik. Ez felfogható egy szülő-gyermek kapcsolatnak is; ebben az esetben a gyermektábla az *Etlap* tábla, a szülő tábla pedig *Vega_Tipus* nevű tábla. A megszorítás azt mutatja meg, hogy a gyermek tábla egy adott sora a szülő tábla mely sorához tartozik. A külső kulcs mindig a másik tábla elsődleges kulcsára (primary key) vagy egyedi kulcsára (unique key) hivatkozik. A webalkalmazás által használt többi tábla és kapcsolat is a fent említett módon jött létre.

Előfordul olyan eset, amikor két táblát szeretnénk olyan módon összekapcsolni, hogy az egyik tábla egyik sorához a másik tábla nem csak egy sora tartozhat. Ekkor ún. Kapcsolótáblákat használhatunk, amiknek felépítése például olyan lehet, hogy szerepel benne egy ID, és ezen kívül még két idegen kulcs. A két kulcs az összekapcsolandó táblákra mutat.

Az ODAC és az ODT

Az Oracle weboldalán rendelkezésre áll az ún. Oracle Data Access Components, amely az Oracle Developer Tools for Visual Studio eszközökkel együtt letölthető. Ez az eszköz együttes többek között tartalmaz Data Providert a .NET-hez, ODBC drivert, SQL*Plus-t, Oracle klienst, stb. Tehát olyan komponenseket tartalmaz, amik támogatják a .NET-et és az Oracle adatbázisokat, valamint A VS2008 környezetébe beemeli az Oracle adatbázis használatát segítő fejlesztői eszközöket, így biztosítva a kapcsolatot a két (vagy több) technológia között. Az ODT tartalmaz egy Oracle névteret, amelyben olyan objektumok

vannak implementálva, melyek támogatják az Oracle adatbázissal történő kapcsolódást a .NET-es programokból, objektumokból.

Az Oracle névtér

Az Oracle névtér segítségével Oracle adatbázishoz írt osztályokat használhatunk a .NET-es projektünkben. Ezen osztályok közül szeretnék néhányat megemlíteni, leginkább azokat, amelyeket alkalmaztam a Vétterem implementálása során. A következő névterek használtam a diplomamunkám írásakor:

- Oracle.Web.Security
- Oracle.DataAccess.Client
- Oracle.DataAccess.Types

Ahogy a neve is mutatja az Oracel.Web.Secutity névtér a biztonságért felelős. Két osztályt használtam ebből a névtérből, az OracleMembershipProvider-t és az OracleRoleProvider-t.

Az OracleMembershipProvider osztály egy leszármazott osztály, ami a MembershipProvider absztrakt osztályból származik. A MembershipProvider definiálja azokat a metódusokat, propertyket, eseményeket amelyek segítségével egyszerűen menedzselhetjük - az én esetemben - a webalkalmazás felhasználóit. Az OracleMembershipProviderben konkrétan az Oracle adatbázisra vannak implementálva e metódusok, propertyk és események. Mivel az őse egy absztrakt osztály, ezért bizonyos metódusait kötelezően implementálnia kell a megvalósító osztálynak. Ilyen metódusok például a CreateUser, ami egy felhasználót "készít", vagy a GetUserNameByEmail, ami visszaadja az adott email címhez tartozó embert. Az Oracle által implementált tagságszolgáltatót használom a Vétteremben az authentikációhoz és a regisztrációhoz. A tagsághoz kapcsolódó adatbázistáblákba való beszúrást, módosítást és törlést a szolgáltató által implementált metódusok valósítják meg, így a fejlesztőnek ezekkel már nem kell foglalkoznia, ha a beépítettek megfelelnek számára. Természetesen lehetőség van egy saját MembersipProvider osztály írására is, ekkor a saját igényeink és adatbázisunk szerint implementálhatjuk le az absztrakt programozási eszközöket.

Az OracleRoleProvider, a OracleMembershipProvider osztályhoz hasonlóan, egy ősosztálytól származik, amely az RoleProvider nevet kapta. Ez az osztály a szerepkörök kezelését támogatja. Az ősosztályban is absztrakt metódusok szerepelnek, mint pl. a CreateRole vagy a FindUsersInRole metódusok. Az előbbi egy szerepkört hoz létre, az utóbbi pedig visszaadja azon felhasználókat, akik az adott szerepkörbe beletartoznak.

Az Oracle.DataAccess.Client névtérben található osztályok közül a Vétterem webalkalmazás a következőket használja: OracleConnection, OracleCommand, OracleDataReader, OracleDbType. Ezen osztályok segítségével építettem ki kapcsolatot a webform és az adatbázis között, valamint nyertem ki adatokat az adatbázisból, de a dolgozatban erre később részletesebben ki fogok térni.

Az Oracle.DataAccess.Types névtérből az OracleTimeStamp osztályt használom.

A webalkalmazás kapcsolódása az Oracle adatbázishoz

Egy webalkalmazás fejlesztése során igen sarkalatos pont, hogy milyen adatbázismotort használunk és hogyan oldjuk meg az adatbázishoz való kapcsolódást, az adatok kinyerését, törlését és karbantartását. A különböző programozási környezetekben közel hasonló módon lehet felépíteni a kapcsolatot az alkalmazások és az adatbázisok között. A Vétteremben a feljebb már leírt Oracle Data Access Components, Orace Data Tools for Visual Studio és az Oracle névtér objektumai segítségével kapcsolódok a használni kívánt adatbázishoz.

Ahhoz, hogy kényelmesen tudjuk használni a VS2008-ból az Oracle adatbázisunkat a megfelelő komponensek feltelepítése után néhány beállításra lesz szükségünk. Először is adjunk hozzá egy új kapcsolatot az adatkapcsolatokhoz. Ezt a VS2008-ban a Server Explorer oldalsávon tudjuk megtenni a Data Connections-re kattintva jobb egérgombbal és az Add Connection helyi menü kiválasztásával. A felpattanó ablakban beállíthatjuk, hogy milyen adatbázishoz szeretnénk kapcsolódni és milyen komponens segítségével. Az adatbázisszerver neve, valamint a felhasználónév és jelszó megadása után létrehoztunk egy kapcsolatot. Ekkor már a fejlesztő környezet segítségével egyszerűbb menedzselési feladatokat el tudunk látni az adatbázisunkon, legyen szó adatbázis táblák, tárolt eljárások, függvények kezeléséről.

Lekérdezések és adatmanipuláció

Tegyük fel, hogy C# kódból szeretnénk valamilyen adatmanipulációt végezni az adatbázisunkon. Ekkor az első lépés, hogy importáljuk a számunkra szükséges névteret, a következőképpen: using Oracle.DataAccess.Client;. Ezután már közvetlenül elérjük a névtér engedélyezett programozási eszközeit (pl.: metódusait, osztályait). Egy konkrét, az általam írt webalkalmazásból vett példán szeretném bemutatni. A következő forráskód a pultos felületen elhelyezkedő akció felviteli form mögötti kódrészletet mutatja be, ami akkor fut le, ha a "Rögzítés" gombra kattintunk:

```
...
int counter = 0;
using (OracleConnection sqlc = new OracleConnection())
{
        sqlc.ConnectionString = ConfigurationManager.ConnectionStrings["VCS"].ConnectionString;
        OracleCommand cmd = new OracleCommand();
        cmd.Connection = sqlc;
        sqlc.Open();
        cmd.CommandType = CommandType.StoredProcedure;

        switch (MilyenRadioButtonList.SelectedValue)
        {
                case "Tipus_akcio":
                {
                    OracleTimeStamp ots = new OracleTimeStamp(CalendarEddig.SelectedDate.Year,
                            CalendarEddig.SelectedDate.Month,
                    CalendarEddig.SelectedDate.Day, 23, 59, 59);
                    cmd.CommandText = "INSERTTIPUSAKCIO";
                    cmd.Parameters.Clear();
                    cmd.Parameters.Add("p_leiras", OracleDbType.Varchar2).Value = TextBoxLeiras.Text;
                    cmd.Parameters.Add("p_kezdet", OracleDbType.TimeStamp).Value =
                            CalendarEttol.SelectedDate;
                    cmd.Parameters.Add("p_veg", OracleDbType.TimeStamp).Value = ots;
                    cmd.Parameters.Add("p_kedvezmeny", OracleDbType.Int32).Value =
                            Int32.Parse(TextBoxSzazalek.Text);
                    cmd.Parameters.Add("p_etipus", OracleDbType.Int32).Value =
                            ETDropDownList.SelectedValue;
                    break;
        }

...
```

```
        counter = cmd.ExecuteNonQuery();
}
```

Az OracleConnection egy olyan osztály, ami egy Oracle adatbázis-kapcsolatot reprezentál. Azzal, hogy using blokkba tettük azt értük el, hogy bármilyen programhiba vagy kivétel esetén az objektum Dispose() metódusa lefut, azaz az objektum által lefoglalt összes erőforrást felszabadítja. Tehát a using blokkban létrehozunk egy OracleConnection objektumpéldányt, majd beállítjuk a ConnectionString tulajdonságát, a web.configban definiált "VCS" nevű kapcsolati string-re. Ezt követően egy OracleCommand objektumpéldányt hozunk létre, ami egy SQL-parancsot, tárolt eljárást, vagy táblanevet reprezentál. Mindezek után az OracleCommand objektumpéldány Connection tulajdonságát az előbb létrehozott OracleConnection-re állítjuk. Egy kapcsolatot az Open() metódussal tudunk megnyitni. A parancsot szimbolizáló objektum CommandType tulajdonsága azt jelöli, hogy egy SQL szöveges parancsot (Text), egy táblanevet (TableDirect) vagy egy tárolt eljárást (StoredProcedure) szeretnénk használni a paranccsal kapcsolatban. A fenti kódrészletben jelezzük, hogy egy tárolt eljárást szeretnénk futtatni majd. A kódot tovább olvasva láthatjuk, hogy a form-on lévő rádiógomblistában bejelölt elemtől függően a switch-case szerkezet eldönti mely utasítások hajtódjanak végre. Majd azt láthatjuk, hogy a parancs CommandText tulajdonságának, ebben az esetben, az adatbázisunkban létező tárolt eljárás nevét adjuk értékül. Egy Oracle-parancs (OracleCommand) objektumpéldánynak létezik egy Parameters nevű tulajdonsága, amelybe a szükséges paramétereket tudjuk beilleszteni. A forráskódban először kiürítjük a Clear() paranccsal, ha esetlegesen maradt volna benne feleslegesen, majd az Add() tízszeresen túlterhelt metódus segítségével új paramétereket pakolunk bele. A túlterhelés azt jelenti, hogy a metódusok neve ugyanaz (Add), viszont a paraméterlistájuk különböző. Az általam használt metódus első paramétere a tárolt eljárás (INSERTTIPUSAKCIO) egy formális paraméterének neve, ezt pontosan meg kell adnunk, mert egyrészt a név szerint illesztődik rá, másrészt az Add metódusok sorrendje is lényeges. Tehát a parancs objektum Parameters tulajdonságához elsőként hozzáadott paraméter a tárolt eljárás első formális paramétere kell legyen. Meg kell még adnunk a paraméter típusát, amit az OracleDBType nevű enumerációt használva tehetünk meg. Ezt követően, ahogy a forrásból is látszik, egyből tudunk neki értéket is adni. A Vétterem esetében általában ASP.NET-es controlok tulajdonságértékeit használva töltjük fel a paramétereket. Ahhoz, hogy a parancs

lefusson, a parancsobjektum példányának Execute***() metódusait futtathatjuk le. Ezek a következők lehetnek:

- ExecuteNonQuery: Egy SQL-parancsot futtat le az adatbázison, akkor használjuk, ha nem lekérdezést szeretnénk futtatni. A módosított, törölt vagy beszúrt sorok számát adja vissza.
- ExecuteReader: Végrehajtja a parancs objektumpéldány CommandText tulajdonságában megadott szöveges parancsot és egy OracleDataReader objektumpéldánnyal tér vissza, melyből lekérhetjük pl. a visszaadott rekordok mezőértékeit. Tipikusan lekérdezésekkor használjuk.
- ExecuteScalar: A visszaadott eredményhalmaz első sorának és első oszlopának értékét adja vissza.
- ExecuteStream
- ExecuteToStream
- ExecuteXmlReader: Végrehajtja a parancsot és egy XML dokumentum formájában kapjuk meg az eredményt.

A fenti forráskódban az ExecuteNonQuery metódust használom, melynek eredménye az lesz, hogy lefut a megadott tárolt eljárás a megadott paraméterekkel. Ebben a konkrét esetben egyrészt az *Akcio* nevű táblába, másrészt a *Tipus_akcio* táblába történik adatbeszúrás, azaz rögzítettünk az adatbázisba egy ételtípusra vonatkozó akciót.

A következő forráskód részlettel egy C#-ból végrehajtott SQL-lekérdezést futtatását és az eredmény feldolgozását szeretném bemutatni:

```
oc.CommandText = "select ak.kezdet, ak.veg, ak.kedvezmeny, ta.etipus, et.etnev " +
  " from akcio ak join tipus_akcio ta on ak.akcioid = ta.akcio " +
  " join etel_tipus et on et.etid = ta.etipus";
using (OracleDataReader odr = oc.ExecuteReader())
{
  int counter = 1;
  while (odr.Read())
  {
        DateTime kezdDT = odr.GetDateTime(0);
        DateTime vegDT = odr.GetDateTime(1);
        int kedv = (int)odr.GetDecimal(2);
        int etipus = (int)odr.GetDecimal(3);
        string etnev = odr.GetString(4);
```

```
        if (kezdDT < most && most < vegDT)
        {
            string megnevezes = "";
            switch (etnev)
            {
                case "LEVES": { megnevezes = "levesre"; break; }
                case "FŐÉTEL": { megnevezes = "főételre"; break; }
                case "DESSZERT": { megnevezes = "desszertre"; break; }
                case "ITAL": { megnevezes = "italra"; break; }
                case "ELŐÉTEL": { megnevezes = "előételre"; break; }
                case "EGYÉB": { megnevezes = "egyéb kategóriájú fogásra"; break; }
            }
            Label label = new Label();
            label.ID = "LA" + (counter++).ToString();
            label.Text = "Ha most rendel, minden " + megnevezes + " " + kedv + "% kedvezményt
                kap!<br/>";
            PanelETA.Controls.Add(label);
    }
}
```

A kódrészlet előtt a kapcsolat létrehozásához minden be van állítva. Mivel az előző kódrészletnél leírtam ezen elemek működését, most csak a konkrét sql-kérdéstől érdemes vizsgálni a forrást. A CommandTextben megadhatjuk az SQL-lekérdezésünk szövegét.

Ezután egy using blokkban deklarálunk egy OracleDataReader típusú változót, amelynek egyből értéket is adunk az ExecuteReader metódus meghívásából származó visszatérési értékkel. Egy OracleDataReader (ODR) objektum egy csak olvasható eredményhalmazt reprezentál. Az ODR Read() metódusa a tárolt halmaz következő sorát olvassa ki és egy logikai értékkel tér vissza, azaz igaz értékkel, ha létezik következő sor, valamint hamissal, ha nem. Tehát egy while-ciklus feltételében használhatjuk, így az összes sort visszakaphatjuk feldolgozásra. A beolvasás után az ODR objektum Get***(i) metódusaival tudjuk az adatokat kiolvasni. A csillagok helyére valamilyen típust tudunk beírni, ezáltal előírva, milyen típusú adatot várunk. Az i helyére az oszlopindexet kell írnunk, azaz ha az adott sor i-edik oszlopához tartozó adatot szeretnénk megkapni, akkor az i-1-et kell írnunk, mert nullától kezdődik a számozás. A forráskódból láthatjuk, hogy változókba olvasom be a visszakapott adatokat a további felhasználás céljából, ami egy kimeneti szöveg összeállítását jelenti, amivel tájékoztatom az ügyfélt az éppen aktuális ételtípusra vonatkozó akciókról. Használhatjuk még a GetValue() metódust is az érték kiolvasásához. Ez akkor lehet

hasznos, ha nem tudjuk milyen típusú értéket fogunk kiolvasni. A metódus egy .NET típusként olvassa ki az értéket és egy Object típusú objektumot ad vissza.

Paraméterek és SQL injection

Mikor parancsokat írunk a forráskódba, akkor lehetőségünk van paraméterezni azokat. Ha Oracle adatbázist és C# programozási nyelvet használunk, akkor például a következőképpen nézhet ki egy paraméterezett parancs és a paraméter értékének beállítása:

```
OracleCommand ocmd = new OracleCommand("SELECT ugyfelid from UGYFEL WHERE
      bejelentkezesinev = :pnev", oc);
ocmd.Parameters.Add("pnev", OracleDbType.Varchar2).Value = User.Identity.Name;
```

Tehát a parancsot leíró string-be úgy kell jelölnünk a paramétereket, hogy elé egy kettőspontot írunk, vagy MSSQL adatbázist használva @-ot. Ez a módszer nem csak olvashatóbbá teszi a kódot, de biztonságosabbá is, mintha string konkatenációt használnánk. Az SQL injection támadások kivédésében is igen hatékony, ha a fenti módszer szerint írjuk meg parancsainkat. Az ilyen típusú támadások azt használják ki, hogy a webalkalmazásokban sok esetben dinamikusan vannak előállítva az SQL parancsok és ezen utasítások közé vagy után káros kódot próbál beilleszteni a támadó. Egy erősebb támadáskor a támadó információkat szerezhet az alkalmazás mögött működő adatbázis táblák, objektumok neveiről és módosíthatja is azokat, akár adatokat is törölhet belőle. Mivel webes felületen egy-egy SQL-parancs paramétereit általában a felületen lévő formról nyerjük, ezért pl. egy textboxba beírt megfelelő karaktersorozat megváltoztathatja az általunk felépített parancs jelentését. Tehát amikor tehetjük, mindig használjunk paramétereket az efféle támadások kivédése érdekén.

ASP.NET vizuális controlok és kapcsolódásuk az adatbázishoz

ASP.NET vizuális control alatt a webes felületen megjelenő táblázatokat, listákat, szövegeket értem. A vizuális controlok esetében is fontos szerepet játszik a connectionString,

ami leírja, hogy melyik adatbázishoz kapcsolódjon az adott elem. Dinamikus weboldalaknál nagyon fontos, hogy tudjuk szabályozni bármilyen esemény bekövetkezésekor az adatbázisból érkező adatok megjelenítését, kezelését.

Minden, adatkötésre alkalmas, objektumnak létezik egy DataBind() metódusa, melynek segítségével adatokat nyerhetünk az adatbázisunkból és kezelhetjük azokat. Az adatkötés a felhasználói interfész és az üzleti logika közötti kapcsolat kiépítését jelenti. Az alkalmazásomban én is előszeretettel használom ezt a fajta adathozzáférési módot. Például az étlap oldalon található asp-kódrészletből is látszik ez:

```
<asp:HyperLink ID="ReszletekHyperLink" runat="server" NavigateUrl='<%# Eval("EtlapID",
        "ItemDetails.aspx?EtlapID={0}") %>' Text='<%# Eval("ENEV") %>'>
</asp:HyperLink>
```

Itt egyrészt egy hiperhivatkozásnak a NavigateUrl tulajdonságát állítjuk be egy futási időben végrehajtott adatkötéssel, oly módon, hogy az EtlapID nevű mezőobjektum értékét kérem le és illesztem be a hivatkozás URL-jébe; másrészt az ENEV nevű mezőobjektum értéket iratom ki a hiperhivatkozás szövegeként. Az Eval metódus a DataBinder osztály egyik módszere, amely segítségével futás idejű adatkötés-kifejezéseket tudunk készíteni.

A Vétterem alkalmazás több helyen használ .NET-es GridView, DetailsView, ListView controlokat. A GridView egy táblázatos nézetet tesz lehetővé, ahol az oszlopok az adatforrás egyes mezőit, a sorok pedig a rekordjait reprezentálják. A DetailsView az adatforrás egy darab rekordját reprezentálja oly módon, hogy a DetailsView egy sora a rekord egy mezőjét jelöli. A ListView nézet hasonló a GridView nézethez. Az előbb említett controlok mindegyike használ egy adatforrást (datasource), amiből az adatokat kapja. Az adatforrások a következők lehetnek a .NET-ben:

- AccessDataSource
- SqlDataSource
- ObjectDataSource
- XmlDataSource
- SiteMapDataSource
- LinqDataSource

- EntityDataSource

Az alkalmazásomban én ezek közül csak az SqlDataSource-t használom. Ez a fajta adatforrás egy SQL lekérdezés eredményeként megkapott rekordhalmazt reprezentál. Például ha egy adatforrást valamilyen adatmegjelenítési nézetet megvalósító controlhoz szeretnénk kapcsolni, akkor a control DataSourceID tulajdonságát kell beállítanunk a felhasználni kívánt DataSource azonosítójára. Az SqlDataSource SelectCommand metódusában megadhatjuk az SQL-lekérdezést, amit használni szeretnénk. Ugyanakkor a DeleteCommand, UpdateCommand és InsertCommand tulajdonságok segítségével beleírhatjuk a rendre törlő, frissítő és beszúró SQL-parancsokat is. Ezeket akkor kell megadnunk, ha a pl. GridView-ban engedélyezzük a törlést és a szerkesztést is. Megadhatunk paraméteres Sql-mondatokat is az SqlDataSource-okban, ekkor felsorolhatjuk a paramétereket és típusukat is. Szeretném a következő példát bemutatni az általam készített webalkalmazás forrásából:

```
<asp:SqlDataSource ID="ItemSqlDataSource" runat="server"
ConnectionString="<%$ ConnectionStrings:VCS %>"
ProviderName="<%$ ConnectionStrings:VCS.ProviderName %>"
SelectCommand="SELECT ... WHERE (ETLAP.ETLAPID = :EtlapID)">
<SelectParameters>
<asp:QueryStringParameter Name="ETLAPID" QueryStringField="EtlapID"/>
</SelectParameters>
</asp:SqlDataSource>
```

Itt láthatjuk, hogy az egyéb megadott tulajdonságokon kívül megadtam a SelectCommand tulajdonság értékét, amelyben használok egy paramétert is. Ezt :paraméternév alakban kell megadnunk ha Oracle adatbázist használunk. Ezután vehetjük észre, hogy egy ún. QueryStringParameter tag-et használva adom meg a paramétert a lekérdezésnek. Ez a fajta paraméter az URL-ben megadott paramétert figyeli és használja. Ez a lekérdezés akkor fut le, amikor egy fogás részleteit tekintjük meg. Ekkor az URL-ben például a következő jelenik meg: http://localhost:3216/ItemDetails.aspx?EtlapID=150. Tehát a QueryStringParameter értéke ekkor 150 lesz.

Nagyrészt az adatmegjelenítést a fent bemutatott módon oldottam meg. Van azonban egy olyan oldal, ahol az adatkötés egy speciálisabb módját választottam. A kosarat megjelenítő oldalnál nem beépített controlok segítségével oldottam meg a dolgot, hanem oldal mögötti kód és osztályok segítségével. Tehát van egy *Kosar* és *KosarTetel* nevű osztály

implementálva a projektben. Egy *Kosar* objektumpéldány hivatott reprezentálni a felhasználó kosarát. Egy *Kosar* objektumpéldányban egy *KosarTetel* lista található, ami a kosárban lévő fogásokat tartalmazza. A kosár objektumot egy Session objektumban tárolom és onnan olvasom ki. A felhasználó kijelentkezésekor felszabadítom a kosarat, azaz gyakorlati szempontból kiürítem azt. A Cart.aspx oldalon, ahol a kosár tartalmát láthatjuk egy Gridben, egy codebehind-ból kezelt adatkötést valósítok meg. Írtam egy BindData nevű metódust, ami adatkötést végez a Grid és a Kosar objektumpéldány között. Első lépésben beolvassa a Session objektumot, majd a Grid DataSource tulajdonságának beállítja a kosár tételeinek listáját. És ezután meghívja a Grid DataBind metódusát. További metódusok segítségével oldom meg a fizetendő végösszeg kiszámolását. A kiszámításnál a program figyelembe veszi a különféle akciókat és a legnagyobb kedvezményt adó akciót választja ki egy adott megrendeléshez, tehát halmozottan nem érvényesíti azokat. Erről a felhasználó tájékoztatást is kap a baloldali oldalsávban. A kosár megtekintésekor még változtathatjuk a kosár tartalmát, tehát törölhetünk belőle, adhatunk hozzá, vagy módosíthatjuk a rendelni kívánt darabszámot. Ezeket is külön erre a célra írt metódusokkal oldom meg. A *Kosar* osztályban a következő metódusok vannak implementálva:

- Berak: adott azonosítójú, nevű és árú terméket berak a kosárba. Meg kell adnunk a berakni kívánt darabszámot is.
- MennyisegValtoztat: a megadott azonosítójú étel kosárban lévő darabszámát változtatja a megadott darabszámra.
- Kiszed: túlterhelt metódus, vagy az összes tételt kiszedi a kosárból, vagy csak a megadott azonosítójú tételből az összest.
- Osszesen: kiszámítja a kosárban lévő termékek végösszegét akciók nélkül.
- OsszesenAkc: kiszámítja a kosárban lévő termékek végösszegét az akciók közül a legkedvezőbbet kiválasztva.

A dolgozatban már említettem, hogy az alkalmazás használ Oracle tagság-, és szerepkör-szolgáltatót. Ezen szolgáltatókhoz kapcsolódik a mesteroldalon található Login control is, amely a bejelentkezést hivatott megoldani a webalkalmazások esetében. A control MembershipProvider tulajdonságát kitöltve megadhatjuk, hogy milyen tagságszolgáltató

szeretnénk használni, tehát ez a control is kapcsolódik az adatbázishoz egy tagságszolgáltató segítségével. A Vétterem alkalmazást használva viszonylag sok adatbázis-manipuláció történik leginkább nyomógombok megnyomása esetén, például az egyik ilyen eset, amikor rendelünk. Ekkor a háttérben több sql-parancs fut le, amelyek a megfelelő táblákba szúrnak be adatokat.

Összegezve elmondhatjuk, hogy szinte minden oldalon történik adatkötés vagy adatáramlás a vizuális elemek és az adatbázis között, ahogy azt egy dinamikus weboldal esetében elvárjuk. Véleményem és tapasztalatom szerint a .NET igen széleskörű támogatást biztosít a szükséges tevékenységek végrehajtásához.

A Vétterem felületének és működésének áttekintése

Ebben a fejezetben áttekintést nyújtok az alkalmazás felületéről és működéséről.

Áttekintés felhasználói szempontból

A felhasználó indít egy böngészőt és beírja a webcímét a fiktív étteremnek. Ekkor egy üdvözlőablakot láthat, ami így néz ki:

7. ábra: Főoldal

A felül elhelyezkedő menüt, a baloldalon látható bejelentkező/kijelentkező dobozt valamint az akciókat jelző dobozt minden egyes oldalra navigálva látni fogja a felhasználó, ugyanis ezen komponensek a mesteroldalon találhatóak. Tehát a felhasználó be tud jelentkezni a bejelentkező dobozban a felhasználónevének és jelszavának megadásával, ha már regisztrált az oldalra. Ha még nem, akkor a regisztráció linkre kattintva tudja azt megtenni, miután a következő felület lesz látható:

8. ábra: Regisztráció

Itt a megfelelő adatokat kitöltve tudja magát regisztrálni a rendszerbe a felhasználó. A regisztrációs űrlapon sokfajta validáció található. Ha közülük egy is nem sikerül figyelmeztető üzenetet kapunk az oldalról. Többek között az is vizsgálva van, hogy a jelszó minimum 5 karakter legyen és tartalmazzon egy nem alfanumerikus karaktert is, vagy hogy az email cím érvényes legyen. Ezeken túl ellenőrizve van még a telefonszám formátuma és a teljes név formátuma oly módon, hogy a névnek két részből kell állnia. Ha az összes validációs feltételnek megfelelően sikerült kitölteni az űrlapot, akkor sikeres regisztráció történhet. Ezután már be tud jelentkezni a felhasználó az oldalra.

Az étlap megtekintéséhez nem kötelező bejelentkezni. Az étlapot a következő felület segítségével érhető el:

9. ábra: Étlap

Étel típus szerint vannak csoportokra bontva a fogások. Láthatjuk a nevüket és az árukat. Ha a nevükre kattintunk, akkor többet is megtudhatunk egy-egy fogásról:

10. ábra: A fogás részletei

Olvashatunk egy leírást az ételről, látjuk az árát valamint megnézhetjük, hogy milyen vegetáriánus-típus tartozik hozzá. Az alul megjelenő csuszka segítségével be tudjuk állítani, hogy hány darabot szeretnénk a kosárba helyezni. A kiválasztott fogást a zöld színű "kosárba!" gombra kattintva tehetjük a kosárba. Miután a kosárba helyeztünk néhány fogást a felső menüből kiválasztva a Kosaram menüpontot tekinthetjük meg a kosarunkat:

11. ábra: Kosár

Itt láthatjuk, hogy milyen fogások, milyen árral és hány darab van a kosarunkban, továbbá a fizetendő összeget is. Lehetőségünk van átírni a rendelni kívánt mennyiséget soronként és a "Frissít" gombra kattintva az adatokat frissíteni a kosárban., valamint teljes sorokat törölni a kosárból a bal oldalt elhelyezkedő piros x-szel áthúzott kosarat ábrázoló gombra nyomva. Ha a "tovább a megrendeléshez" gombra kattintunk, akkor még egyszer áttekinthetjük a kosarunk tartalmát, majd a "Rendelek!" gombra kattintva feladhatjuk a rendelésünket. Kapunk egy üzenetet a rendelés feladásának sikerességéről vagy sikertelenségéről.

12. ábra: Rendelés

Áttekintés a pultos szempontjából

Az alkalmazásba kétféle szerepkörben lehet bejelentkezni: pultos és user. Ha a pultos bejelentkezik az oldalra a következő képernyőképet látja:

13. ábra: Aktív rendelések

Itt láthatja az aktív rendeléseket. A legújabb rendelés van mindig a legfelső sorban, tehát a rendelések beérkezésének csökkenő sorrendbe van rendezve a megjelenítő tábla (alapértelmezett módon). Ha egy adott sor elején lévő checkbox-ot bepipáljuk, akkor az alul elhelyezkedő gombok közül a megfelelőre kattintva Készítik, Futárnál vagy Teljesítve állapotba teheti a rendelést a pultos. Ezen opciókon túl lehetősége van törölni is a kiválasztott rendeléseket. Minden sor végén található egy Részletek link. Erre kattintva a pultos megnézheti az adott rendeléshez tartozó összes információt:

14. ábra: Rendelés részletei

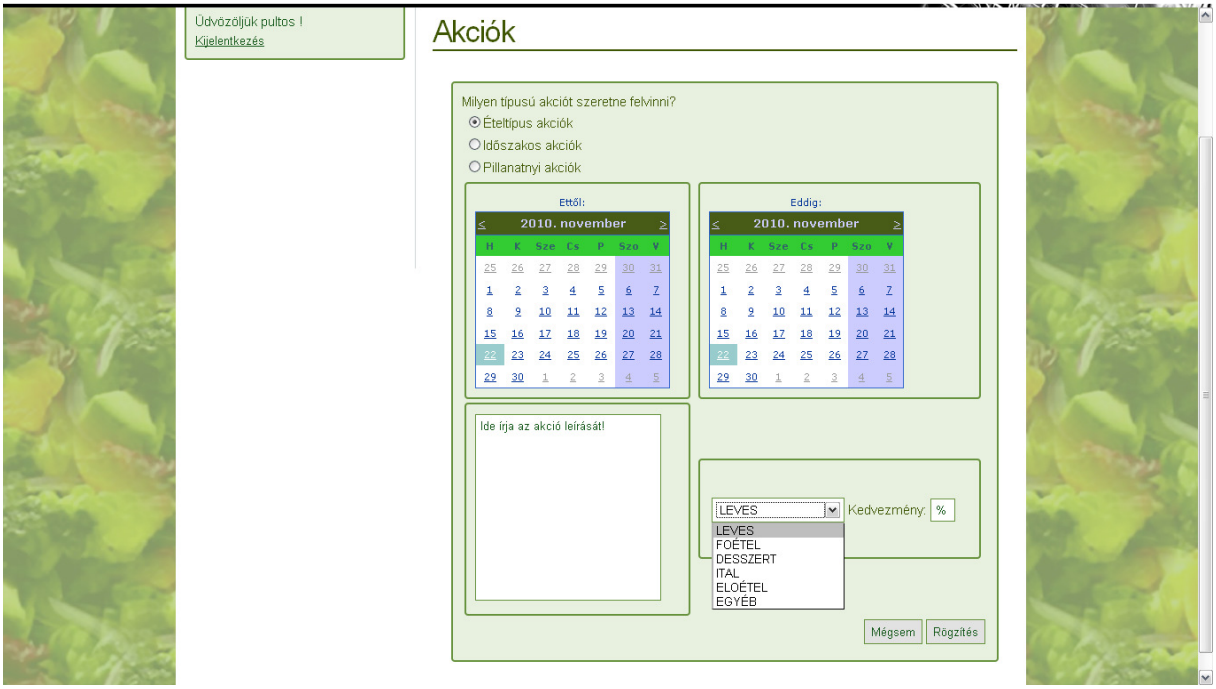
Ha egy rendelést teljesítetté tesz a pultos, akkor az aktív rendelések táblázatból eltűnik és a Teljesített rendelések menüpontban található táblázatban jelenik meg ezentúl. Itt is használható a Részletek link az aktív rendelések menüponthoz hasonló eredménnyel.

15. ábra: Teljesített rendelések

A pultosnak lehetősége van új akciókat felvinni, vagy törölni azokat, valamint láthatja az épp érvényben lévő akciókat. E funkciókat az Akciók menüpont alatt érheti el:

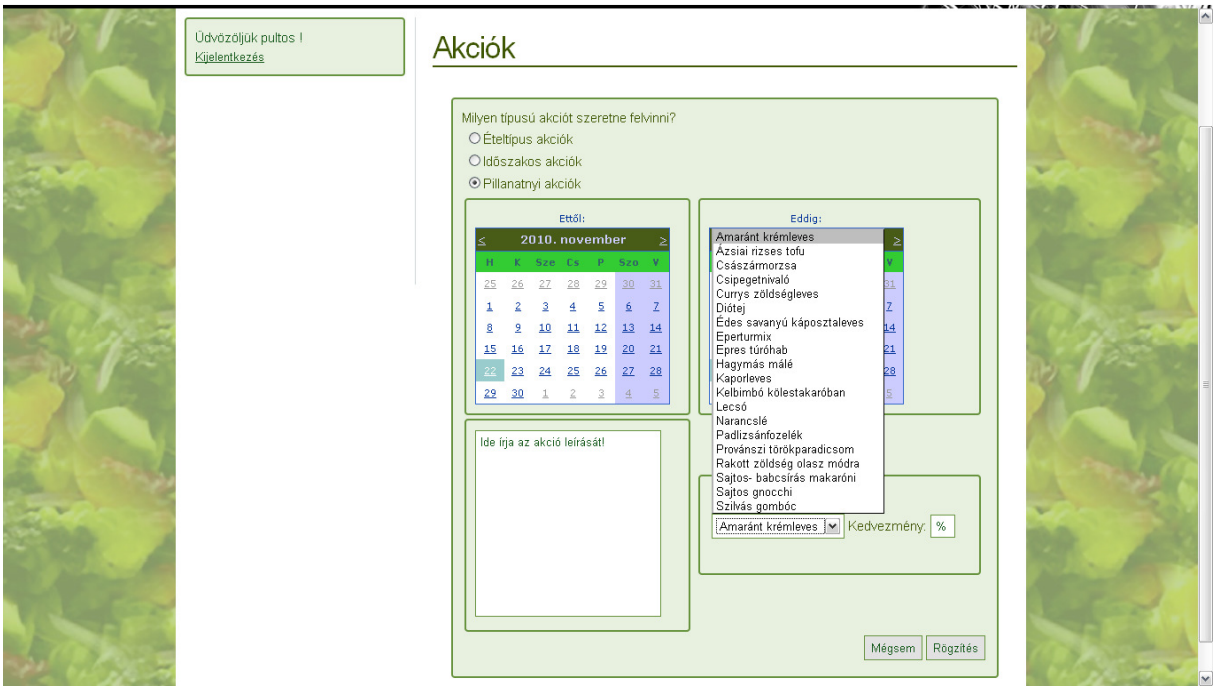
16. ábra: Akciók

Itt táblázatos formában nézheti meg, hogy milyen akciók vannak definiálva jelenleg az oldalon, és itt törölhet is akciókat. Alul az “Új akció felvitele” gombra kattintva tud a pultos új akciókat felvinni a rendszerbe.



17. ábra: Akciók felvétele 1

Háromféle akciótípus közül választhat a megfelelő rádiógombra kattintva, ami után az akció felvitelére készített form dinamikusan változik, mindig a kiválasztott akciótípus sajátosságaihoz igazodva.



18. ábra: Akciók felvétele 2

Egy akció felvitelénél a pultosnak mindig ki kell választania a kezdő-, és végdátumot és kitöltheti az akció leírására alkalmas szövegdobozt, valamint az akciótípustól függően további speciális adatokat kell megadnia és természetesen a kedvezményt százalékban. Mindezek után a mégsem gombbal elvetheti az új akció felvitelét, vagy a Rögzítés gombot használva rögzítheti azt. Természetesen itt is történik validáció a beállított adatokra nézve. A bejelentkezett illető a Kijelentkezés gombra kattintva tud kijelentkezni az oldalról.

Összefoglalás

A dolgozatom igyekezett bemutatni egy webshop alapú rendszer fejlesztését az ASP.NET technológiára és az Oracle adatbázismotor használatára épülve. Az Oracle adatbázis használata közel hasonló módon történik, mint ha MSSQL adatbázismotort használnánk, ugyanakkor az előbbi, véleményem szerint, szélesebb körben elterjedt és professzionálisabb, stabilabb működést biztosít számunkra. A dolgozatban próbáltam kiemelni a fontosabb lépeseket, területeket egy ilyen típusú webalkalmazás fejlesztése estében. Részletesebben fejtettem ki a fejlesztői munkámat segítő eszközökről, bemutattam a Vétterem víziójától a megvalósulásáig vezető utat végighaladva az egyes lépéseken. Ezeken túl betekintést engedtem a Visual Studio 2008 integrált fejlesztői környezet néhány tulajdonságába, sajátosságába és programozás filozófiájába. Bemutattam, hogyan lehetséges az Oracle adatbázisból adatokat elérni, feldolgozni vagy éppen adatokat elhelyezni az adatbázisba különböző programozástechnikai eszközökkel.

Szeretném azonban megjegyezni, hogy az általam írt online étterem csak egy tesztalkalmazás, azaz ahhoz, hogy élesben is működhessen egy ilyen típusú rendszer, több ötletet is meg kellene valósítani benne. Gondolok itt például az email küldő funkciók használatára, azaz regisztrációkor esetleg egy visszaigazoló email küldése a felhasználónak a regisztrációs adatai leírásával és egy regisztrációt megerősítő linkkel, vagy szintén egy-egy üzenet küldése a rendelés különböző állapotba kerüléséről, így szinte percre pontosan tudná követni a felhasználó, hogy milyen stádiumban van a rendelése. Továbbfejlesztési lehetőségként szeretném megemlíteni még a felhasználói profil oldal és profilmódosító oldal integrálását az alkalmazásba, ugyanis jelenleg csak a regisztrációs adatait és jelszavát tudja a felhasználó használni, tehát ha elköltözik egy új címre, vagy nem otthonra szeretne ételt rendelni, akkor nem tudja módosítani a profilja adatait. Vagy beszélhetnénk fejlesztési lehetőségként még az oldal adatelérést átalakító törekvésre is, ami sql parancsok írása helyett Microsoft LINQ [1](Language Integrated Query, azaz nyelvbe integrált lekérdezés) komponenst használna ezzel kényelmesebbé, esetlegesen gyorsabbá téve a kódolást. Lehetne továbbá több képet használni az oldalon, gondolok itt az étlapra leginkább. A valóságban a felhasználó

[1] http://msdn.microsoft.com/en-us/netframework/aa904594.aspx

könnyben tud választani fogást, ha azt is látja, hogy hogyan fog az kinézni. Ezeken túl a pultos felületet is érdemes lenne bővíteni egy olyan formmal, ami segítségével képes lenne új étlapelemeket felvinni az adatbázisba. Ezen továbbfejlesztési javaslatok megvalósítása után már egy jobban használható online éttermet kaphatnánk.

A Visual Studio fejlesztőeszközzel folyamatosan foglalkozom és már a megismerése után nagyon tetszett robosztussága és az, hogy számtalan fejlesztést segítő komponenst foglal magában, valamint lehetővé teszi a vizuális fejlesztést is és nagyon átlátható módon szét van választva benne a design és a forráskód vagy üzleti logika. Az Oracle adatbázis motorról pedig mindig is jó véleménnyel voltam egyrészt az egyetemi oktatóim hozzáállása, másrészt a személyes tapasztalatok alapján.

Mindent összegezve ajánlom minden webfejlesztés után érdeklődőnek a dolgozatban használt eszközök, technológiák megismerését, mert azok segítségével igen kényelmes módon tudunk látványos és jól működő eredményeket elérni, annak ellenére, hogy a .NET keretrendszerben kisebb hibák előfordulhatnak.

Köszönetnyilvánítás

Szeretném megköszönni témavezetőmnek, Bérczes Tamásnak, a diplomamunkám elkészítése közben nyújtott segítségét, valamint szeretném megköszönni ismerőseimnek, szeretteimnek a folyamatos támogatást. Ezentúl köszönetet szeretnék mondani még munkatársaimnak, akik szakmai tudásukkal és tapasztalataik megosztásával segítették elő a dolgozat és az alkalmazás megszületését.

Irodalomjegyzék

Albert István, Balássy György, Charaf Hassan, Erdélyi Tibor, Horváth Ádám, Levendovszky Tihamér, Péteri Szilárd, Rajacsics Tamás: A .NET Framework és programozása
(SZAK Kiadó 2004 - Budapest)

Juhász István, Gábor András: PL/SQL-programozás Alkalmazásfejlesztés Oracle9*i*-ben
(Panem Könyvkiadó 2002 - Budapest)

Dr. L. Nagy Éva: SQL röviden
(Egyetemi jegyzet 2005 - Debrecen)

Internetes források:

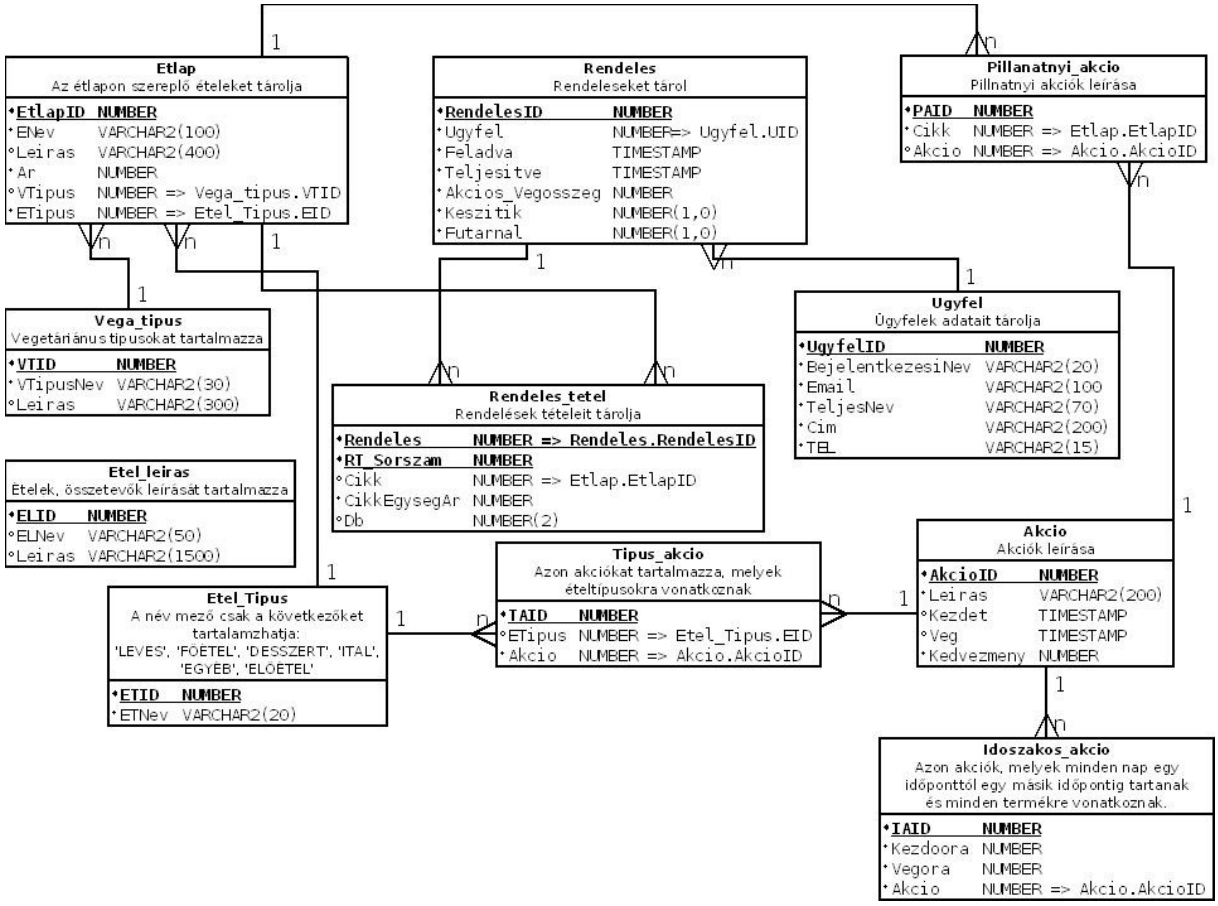
Oracle Data Provider for .NET Developer's Guide:
http://download.oracle.com/docs/html/A96160_01/intro.htm

ASP.NET and Visual Web Developer (Documentations):
http://msdn.microsoft.com/en-us/library/dd566231.aspx

MSDN Library (Documentations):
http://msdn.microsoft.com/en-us/library/ms123401.aspx

http://www.asp.net/
http://www.asp.net/ajax/ajaxcontroltoolkit/samples/
http://hu.wikipedia.org/wiki/C_Sharp
http://en.wikipedia.org/wiki/ASP.NET
http://hu.wikipedia.org/wiki/.NET_keretrendszer
http://hu.wikipedia.org/wiki/Oracle
http://en.wikipedia.org/wiki/SQL_injection

Függelék – Adatbázisterv



Függelék – Alkalmazás specifikáció

Online étterem specifikáció

Az étterem működése:

Az online étterem lehetőséget biztosít az ügyfeleknek, hogy online rendeljenek ételt és akár italt is. Az ügyfél regisztráció és bejelentkezés után feladhatja a rendelését a virtuális pultosnak. A rendelést a véglegesítés előtt módosíthatja az ügyfél, azaz törölhet és hozzáadhat a rendelésének listájához. Ezután leadhatja a rendelést, amit még áttekinthet, majd véglegesítenie kell. Ezen folyamat után érkezik az étterembe a rendelés, melyet a pultos fogad és továbbítja az igényt a szakácsnak. A szakács elkészíti az ételt és jelez a pultosnak, ha készen van. Ekkor a pultos kiadja a rendelt ételeket, italokat a futárnak, aki a rendelésben megadott címre szállítja azt. A futár étterembe való visszatérése után a rendelés teljesítettnek tekinthető, melyet a futár jelez a pultos felé (szóban), majd a pultos jelez a rendszer felé. A pultos továbbá képes akciókat definiálni és törölni. Három féle akciót lehet felvenni a rendszerbe: ételtípusra vonatkozó (pl. levesekre), naponta ismétlődő időszakra vonatkozó (pl. minden nap este 8-tól 11-ig tartó időszak) és konkrét cikkekre vonatkozó (pl. köménymaglevesre). Feltételezzük, hogy csak helyben van házhozszállítás.

Az interfész:

Az ügyfelek bejelentkezési nevük, jelszavuk, email címük, teljes nevük, címük és telefonszámuk megadásával tudnak regisztrálni, ahol a bejelentkezési névnek és az email címnek egyedinek kell lennie. Ezután már be tud jelentkezni a bejelentkezési neve és jelszava megadásával. Az étlap megtekintéséhez nem szükséges bejelentkezni.

A bejelentkezett *ügyfél* láthatja az étlapot és rendelhet a megfelelő termék kiválasztásával, valamint látja az aktuális rendelésének tételeit, árait és az összesített árat.

A bejelentkezett *pultos* látja a beérkező rendeléseket, a folyamatban lévő rendeléseket (azaz ami épp elkészítés/kiszállítás alatt van), a teljesített rendeléseket visszamenőleg valamint az aktuális akciókat. Ezenkívül lehetősége van akciókat felvenni/törölni.

A Rendszer:

A rendszer egy naplót vezet a következő eseményekről:

- felhasználó regisztráció
- felhasználó bejelentkezés/kijelentkezés
- pultos bejelentkezés/kijelentkezés
- rendelés feladások
- rendelés teljesítések
- rendelés törlések
- akciók felvétele
- akciók törlése
- hibák

A napló minden bejegyzésének tartalmaznia kell az esemény dátumát, idejét, a felhasználó vagy pultos nevét (eseménytől függően) és az IP-címét.

Függelék – Implementációs terv

Vétterem Solution

- App_Code könyvtár
  - Kosar.cs
  - KosarTetel.cs
  - Naplo.cs
- App_Themes könyvtár
  - Theme könyvtár
    - css könyvtár
      - img könyvtár
      - design.css
- kep könyvtár
- Pultos könyvtár
  - ActiveOrders.aspx
  - ActiveOrdersDetails.aspx
  - Sales.aspx
  - ServedOrders.aspx
  - ServedOredersDetails.aspx
  - web.config
- Cart.aspx
- Default.aspx
- Global.asax
- ItemDetails.aspx
- Menu.aspx
- Order.aspx
- Registration.aspx
- Vetterem.Master
- web.config

Függelék – USE-CASE diagram