Szakdolgozat

Gáncsos Attila

Debrecen

2009

Debreceni Egyetem

Informatika Kar

Ügyfél regisztrációs és nyilvántartó rendszer

Témavezető:

Bérczes Tamás

Egyetemi tanársegéd

Készítette:

Gáncsos Attila

Mérnök informatikus

Debrecen
2009

Tartalomjegyzék

Köszönetnyilvánítás

Köszönettel tartozom Bérczes Tamás egyetemi tanársegédnek a munkám során nyújtott állandó támogatásért és folyamatos segítőkészségéért. Szakmai irányítása és útmutatásai elengedhetetlenek voltak a dolgozatom készítése, valamint regisztrációs rendszer fejlesztése közben.

Hálával tartozom Rutkovszky Edéné egyetemi tanársegédnek. Segítsége és ösztönzése nélkül nem készült volna el az Informatikai Szakmai napok regisztrációs és nyilvántartó rendszere.

Végül köszönetet szeretnék nyilvánítani a tesztelésben részvevő diákoknak, akik javaslataikkal és észrevételeikkel tovább javították a rendszert.

1. Bevezetés

1.1. Szakdolgozat témája

A szakdolgozatom témája Ügyfél regisztrációs és nyilvántartó rendszer, melynek keretén belül egy a Debreceni Egyetem Informatika Karán megrendezett Informatikai szakmai napok lebonyolítását megkönnyítő programot fejlesztek. A rendszer magába foglalja a regisztrációt, a nyilvántartást, a későbbi adat visszakeresést, statisztikák készítését, jelenléti ívek készítését.

1.2. Téma választás indoklása

Az Informatika karon elsőként megrendezett Szakmai napokon is már, mint szervező vettem részt. 2009. október 20-tól már a 3. alkalommal kerül megrendezésre a rendezvény. Hatalmas, több százas létszám jelent meg az eddigi Szakmai napokon, aminek a regisztrálása és nyilvántartása papíron folyt. A gondolat, hogy ezt másképp is lehetne a 2008-as (2. Informatikai Szakmai Napok) rendezvény alatt született meg. Azzal, ha lenne egy program, amivel nyilván tudnánk tartani a diákokat, megkönnyítenénk a szervezést, és a gyorsaság érdekében használhatnánk a diákigazolvány vonalkódját is. Ez elég ösztönzést adott arra, hogy egy olyan programot készítsek, amely megkönnyíti és leegyszerűsíti a szakmai napok, (minimális átalakítással akár más rendezvények) regisztrációját és nyilvántartását.

2. Programozási eszközök

2.1. Programozási nyelvek

2.1.1. C#

Mielőtt bemutatnám a nyelvet, röviden leírnám mi is az objektum orientált programozás. Maga a gondolkozásmód a valós világ lemodellezésén alapul, ahol osztályokat, és azon belül objektumokat hozunk létre. Az objektum orientált programozási módszereknél nem a műveletek megalkotása áll a középpontban, hanem az egymással kapcsolatban álló programegységek, és ezek hierarchiája létrehozása. Egy objektum orientált program nem más, mint egymással kommunikáló objektumok összessége. Minden objektumnak megvan a maga jól definiált feladata.

A C# a Microsoft által kifejlesztett objektumorientált programozási nyelv, azonban vannak nem objektum orientált jellemzői is. A nyelv alapjául a C++ és a Java szolgált. A C# Windows operációs rendszereken kívüli használata kevéssé elterjedt a .NET keretrendszer miatt, ugyanis az osztályhiearchiát szolgáltató framework importálása más rendszerek alá még nem valósult meg.

Rövid példa, Hello World c# nyelven:

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

2.1.2. ASP .NET

ASP (Active Server Pages). Dinamikus weboldalak generálására alkalmas szerveroldali keretrendszer, melyet a Microsoft fejlesztett ki. Maga az ASP 1996-ban jelent meg, viszont 2002-ben a .NET keretrendszer megjelenésével az ASP .NET váltotta fel.
Egyre inkább kezdenek elterjedni az ASP .NET-ben programozott honlapok, de sokak véleménye szerint soha nem lesz olyan népszerű, mint a nyílt PHP-ben programozott honlapok. Ennek egyik oka lehet, hogy kevés manapság a keretrendszert támogató host szerver, ugyanis nagyobb az erőforrás igénye mint az PHP-s honlapoknak.

Rövid példa, Hello World ASP .NET nyelven:

```
<html>
<body>
<%
  Response.Write "Hello, World!"
%>
</body>
</html>
```

2.1.3. SQL

A relációs adatbázis-kezelők általában az SQL nyelven programozhatók. Számos SQL nyelvjárásról beszélhetünk. Jellegét tekintve ez a programozási nyelv részben procedurális, részben deklaratív.

A nyelvi elemeket szokásos adatdefiníciós (Data Definition Language, DDL) és adatkezelési (Data Manipulation Language, DML) részekre bontani. A nyelvben az utasításokat a pontosvessző választja el egymástól.

Az SQL nyelv utasításai két csoportra bonthatóak (a felsorolás nem teljes körű):
Az első csoportba tartoznak az *Adatdefiníciós utasítások*:
CREATE - Adatbázis objektum létrehozása.
ALTER - Adatbázis-objektum módosítása.
DROP - Egy adatbázisbeli objektum megszüntetése
COMMENT - Megjegyzést fűz egy adatbázis-objektumhoz.
A második csoportba tartoznak az *Adatkezelő utasítások*:
SELECT - A SELECT utasítás az adatok egy halmazát válogatja ki egy táblázatba a relációs

adatbázisból, és teszi elérhetővé valamilyen technikával a felhasználó számára.
WHERE - Szűrési feltételeket fogalmaz meg, amelyek szűkítik az eredményhalmazt.
GROUP BY - Egyes sorok összevonását, csoportosítását írja elő az eredménytáblában.
HAVING - A WHERE-hez hasonlóan itt is szűrést fogalmazhatunk meg, azonban itt a csoportosítás utáni eredményhalmazra.
ORDER BY - Az eredményhalmaz rendezését adja meg.
CASE - CASE WHEN logikai vizsgálat THEN kifejezés ha igaz .. ELSE kifejezés ha az előzőekre nem illeszkedik END A logikai vizsgálat eredményétől függően vezérelhetjük, hogy mit szeretnénk az adott oszlopban látni.

2.2. Fejlesztői környezetek

2.2.1. Microsoft SQL Server 2005 Express edition

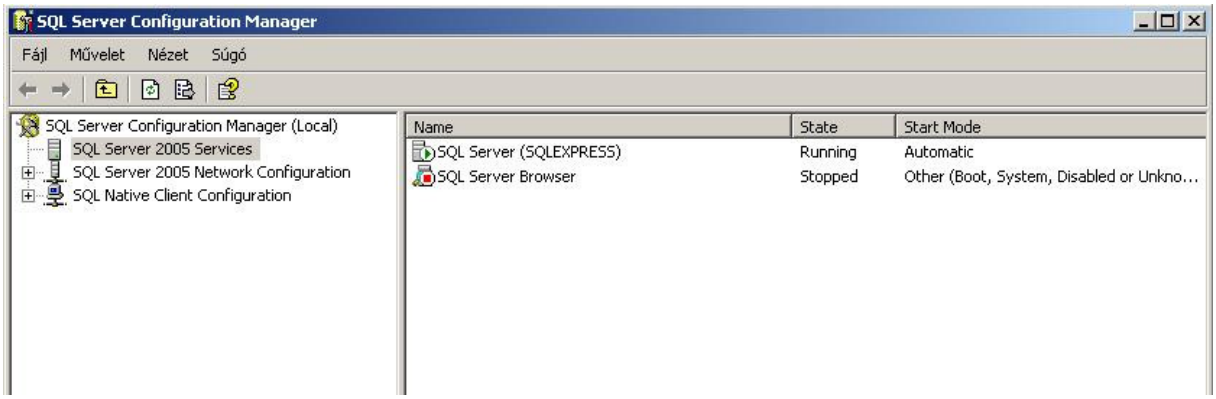
Miután megterveztem az adatbázisomat, választanom kellett egy adatbázis szervert, ahol ezt megvalósíthatom. Választásom a Microsoft adatbázis szerverére jutott.

Az SQL Server Express™ ingyenes, könnyen használható és robusztus adatbázis-kezelő, ezáltal minden fejlesztőnek segítséget nyújt nagy teljesítményű és megbízható alkalmazások készítéséhez. Az adatbázis-kezelő rendszerek a legtöbbször túlzottan bonyolultak az egyszerű alkalmazások elkészítéséhez.

Egyszerű telepíteni és beállítani, gyors letöltés után, egyszerű telepítési felületen keresztül installálható. Használata könnyű; atomatikus hangolás, a gyakori műveleteket segítő varázslók. Biztonság terén robosztusnak mondható, hisz az alapértelmezett beállítások biztonságosak, kifinomult jogosultság, Active Directory-támogatás, Windows-hitelesítés támogatása.

Sokoldalú adatbázis-funkciók: tárolt eljárások, nézetek, eseményindítók (triggerek), kurzorok, kiterjesztett indexek, pillanatkép-izolációs szint, korszerű lekérdezés optimalizáló (Query Optimizer), T-SQL-támogatás, XML-támogatás. Mélyreható integráció a Visual Studio fejlesztőkörnyezettel.

A telepítés után az adatbázis szerverünk már azonnal működőképes, ahogy azt a lenti képen is láthatjuk.

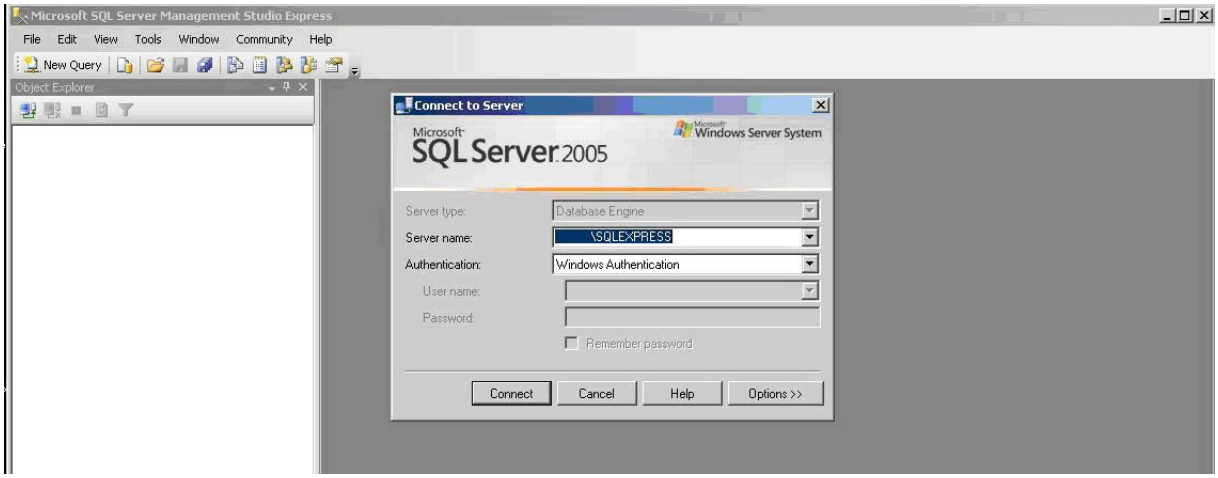


(A futó SQL szerver)

A fent leírtak alapján láthatjuk az adatbázis szerverünk egyszerűségét. Csupán egy, előre jól felkonfigurált szervert kapunk telepítés után, amihez azonnal csatlakozhatunk. Az alapértelmezett beállítások miatt csak Windows autentikáció segítségével léphetünk be, TCP/IP hozzáférés le van tiltva, amit engedélyeznünk kell. A beállításokat a fenti képen is látható SQL Server Network Configuration menüpont alatt tehetjük meg. (A pontos beállításokat, képekkel illusztrálva a 3.1.2. fejezet alatt bővebben kifejtem.) Az SQL szerveren az eléréshez szükséges beállításokon kívül más beállításokra nem volt szükség.

2.2.2. SQL Server Management Express edition (SSME)

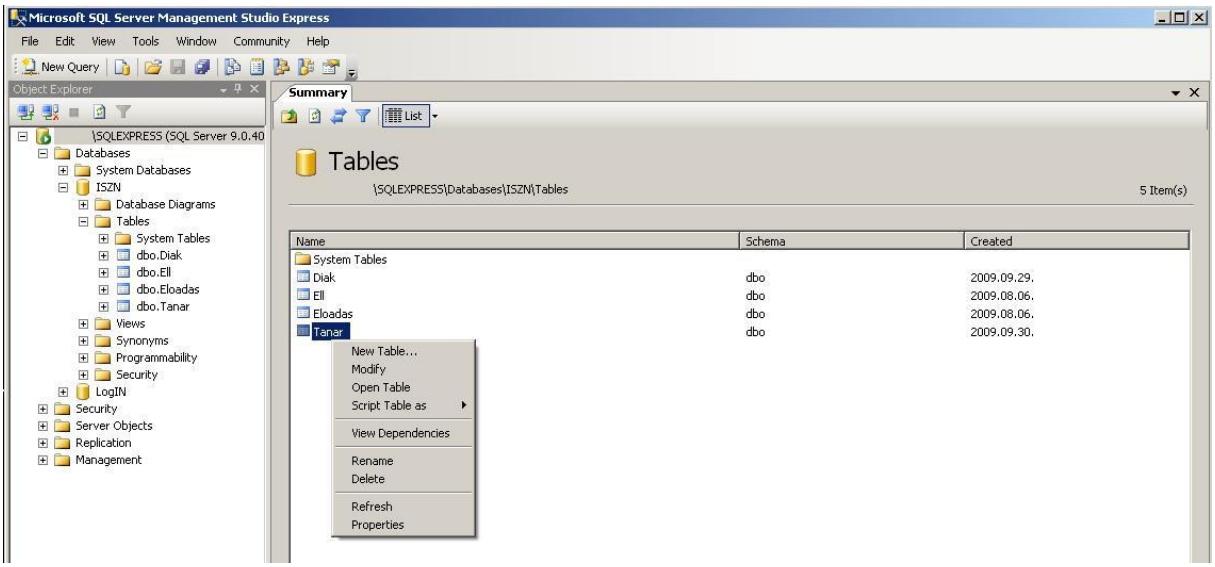
Az SQL Server Manegement segítségével csatlakozni tudunk adatbázis szerverhez, amit akár távolról is el tudunk érni a megfelelő beállításokkal. Az egyszerűen kezelhető grafikus felületnek köszönhetően mondhatni gyerekjáték adatbázist létrehozni, és azt menedzselni.



(Belépés az adatbázis szerverre)

Ez a kép a legelső, ami elénk ugrik rögtön az SSME indítása után. Az adatbázis megfelelő

beállítása után csupán csak az elérési utat kell megadnunk, és az autentikációt. Két féle autentikációt lehetséges: az első a Windows autentication, amit akkor használunk, ha tartományon belül, akár saját gépünkön van az SQL szerver, amihez csatlakozni kívánunk; második a Server autentication, ami a távoli szerverek elérésénél szükséges. A Server autentikációnál a szerver eléréséhez szükséges jelszóra, és felhasználói névre van szükség. (Külső szerver eléréséhez az SQL szerver, és a Windows Tűzfal megfelelő beállítása szükséges. 3.1.2-es fejezetben szó lesz a megfelelő beállításokról.) A sikeres belépés esetén a következő kép tárul elénk, ahol láthatjuk adatbázisunkat, táblákat hozhatunk létre, módosíthatunk, megnyithatunk, stb. (Az ISZN adatbázis a Szakmai napokon használt adatbázis mintájára jött létre. Felépítését a 3.1-es fejezetben magyarázom.)



(Adatbázisunk menedzselése)

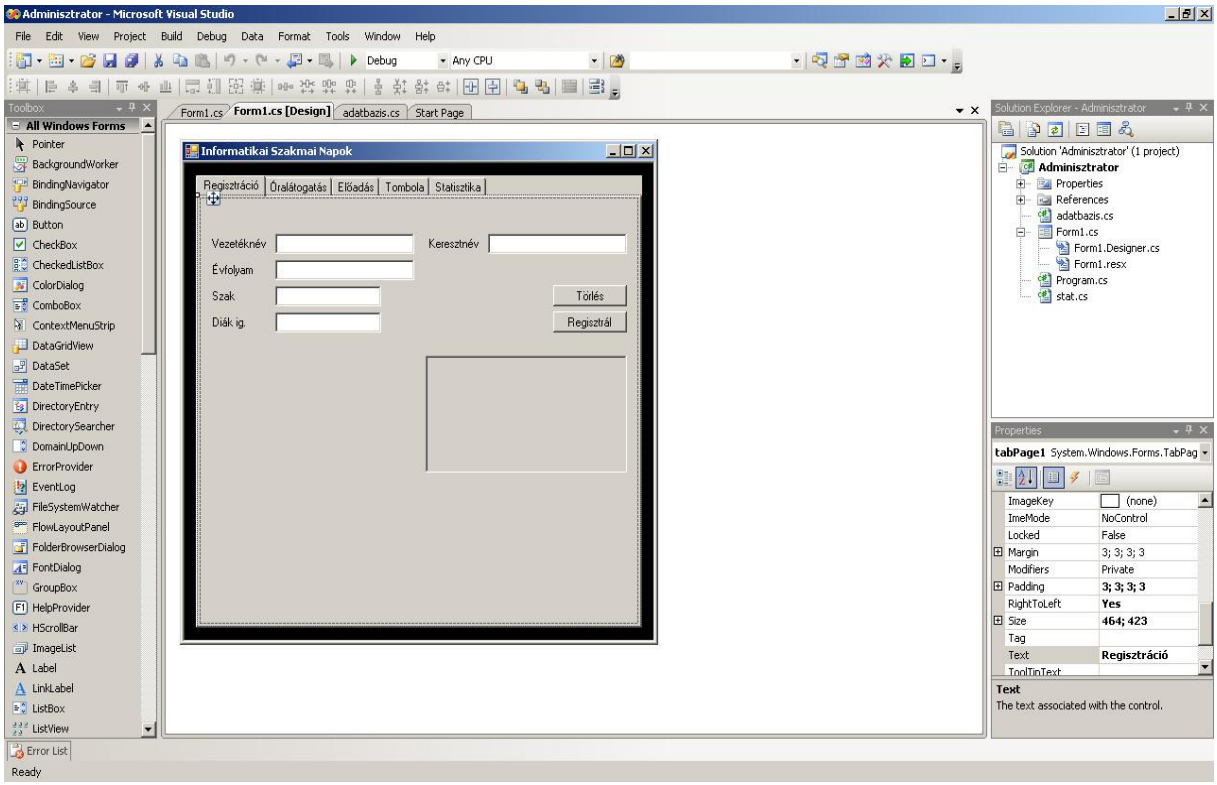
2.2.3. Microsoft Visual Studio 2008

Munkám során c# nyelven programoztam, amihez a Microsoft Visual Studio fejlesztői környezetet használtam.

A Visual Studio 2008 (későbbiekben VS) három területen, a gyors alkalmazásfejlesztés, a csoportmunka támogatása és a gazdag felhasználói élményt adó kezelőfelületek készítése terén jelentős újításokkal szolgál. A modellezéstől kezdve a kódoláson át a hibakeresésig továbbfejlesztett nyelvi, tervező, szerkesztő és adatkapcsolati funkciókat kínál, amelyek jelentősen növelik a fejlesztés hatékonyságát. A közös eszközök és a folyamatok integrációja révén az architektek, a projektmenedzserek, a fejlesztők, a grafikai tervezők és a tesztelők együttműködése hatékonyabbá válik, így számottevően csökkenthető a fejlesztéshez

szükséges idő.

A képernyő átlátható felépítése miatt egyszerűen és gyorsan készíthetünk saját projekteket, legyen szó akár konzolos, Windows Form-os felületről, vagy webes alkalmazásról, nem beszélve a VS által nyújtott már kész komponensekről. Program készítése során két projektet hoztam létre, ebből egyik Windows Form-os alkalmazás, a másik pedig egy ASP .NET WEB alkalmazás. Ezeket később még egy, az adatbázisokat szinkronizáló Windows Form-os felülettel egészítettem ki.



(Windows Form-os alkalmazás fejlesztése)

A Visual Studio nagy előnye hogy a Windows Form-os alkalmazások fejlesztése mellett ASP .NET-es Webes alkalmazás fejlesztést is támogatja. A web szerkesztőjében a felhasználók párhuzamosan, egymás mellett láthatják a forráskódot és a kész oldalt, így folyamatosan nyomon követhetik a változásokat. Emellett egy külön ablakban kaptak helyet a CSS-stíluslapok, amelyeket akár a forráskód, akár a kész design oldaláról is módosítani lehet, ha szükséges. Webes alkalmazások fejlesztésénél, mint a Windows Form-os projekteknél, itt is segítségünkre van a baloldalon található Toolbox, amin számtalan előre elkészített objektum található, amit egyszerűen csak a munkalapra kell helyezni. Az elhelyezett objektumokat, legyen szó gombokról (button), szöveges dobozokról (richtextboc, textbox) a jobb oldalon található tulajdonságok (properties) ablakban a legfinomabb beállításokat is

elvégezhetjük. A beállítások azonnal megjelennek a ASP kódok között, így nem kell a beállításokat begépelnünk, csupán néhány kattintással elérhető a kívánt beállítás.

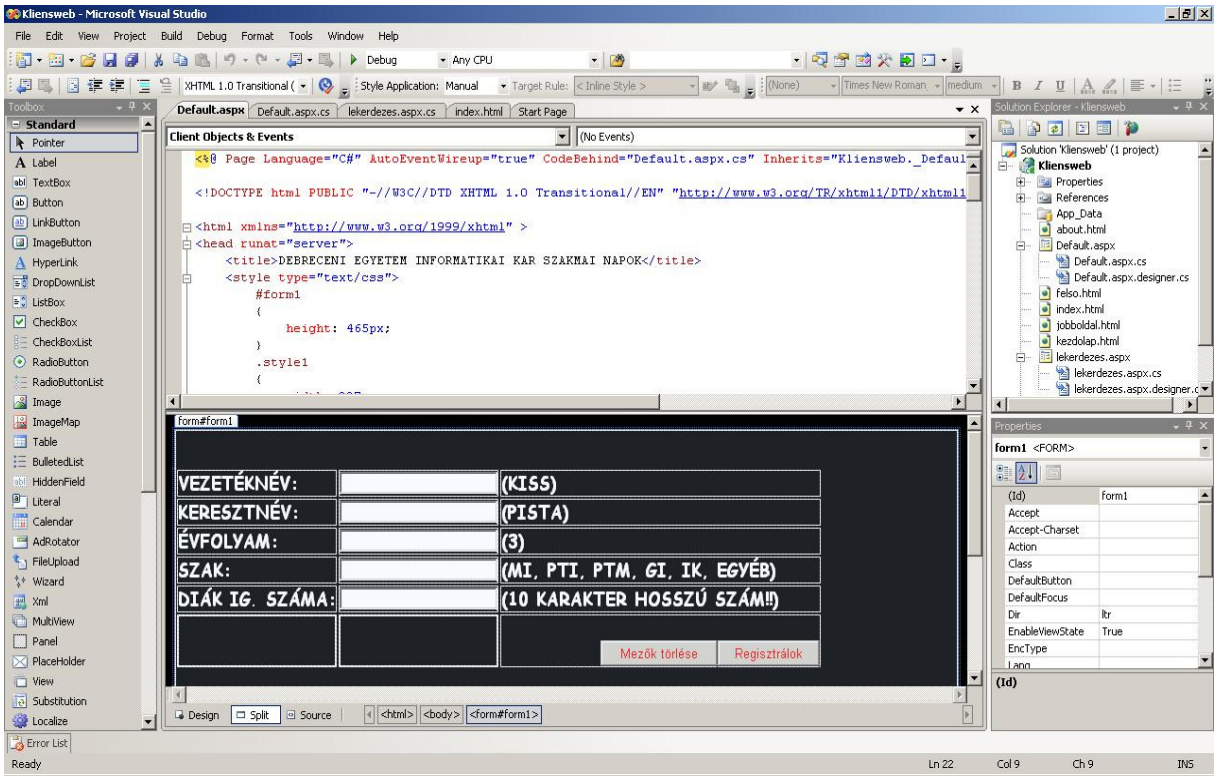
Egy példa a Visual Studio fejlesztést megkönnyítő eszközeinek sokaságából:

Egy szöveges dobozt helyeztünk el egy weblapon. Elhelyezéskor a következő kód jelenik meg a „Source” felületünkön.

```
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
```

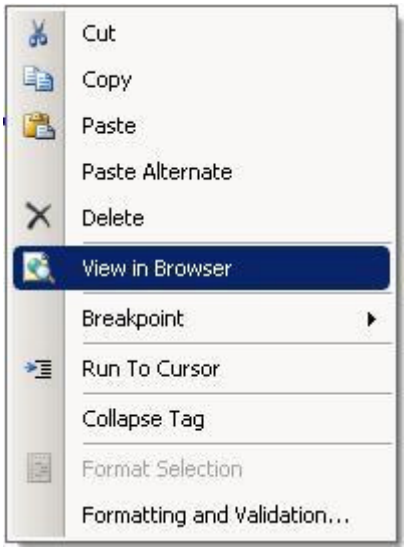
A jobb oldali Tulajdonságok (properties) ablakban beállíthatjuk, hogy a szöveges dobozba írt karakterek csupán pontokként jelenjenek meg, mintha jelszót gépelnénk be. A beállítás után a kódunkat nézve észrevehetjük a változást:

```
<asp:TextBox ID="TextBox2" TextMode="Password"
runat="server"></asp:TextBox>
```



(ASP .NET Web alkalmazás fejlesztése)

Webes alkalmazások fejlesztésénél nagy előny ha a készülő projektet akár már a böngészőnkben is láthatjuk és tesztelhetjük. A VS alatt erre lehetőségünk van. A desing vagy a code felületen jobb klikkre felhozott lehetőségek között a „View in browser”-re kattintva az alapértelmezett böngészőnkben megjelenik a fejlesztett honlapunk, legyen akár az kész, vagy félkész stádiumban.



(Böngészőben nézhetjük meg a fejlesztett honlapunkat)

Ez és számtalan könnyítés teszi azt lehetővé, hogy gyorsan, kevés energiával jó működő weblapokat hozhassunk létre, úgy is, hogy még csak most találkoztunk először ezzel a fejlesztői környezettel.

3. Program működésének leírása

3.1. Adatbázis

3.1.1. Az adatbázis megtervezése, létrehozása

A munkám során a legelső teendő volt az igények felmérése, majd a megfelelő adatbázis megtervezése, ugyanis ha később változtatásra van szükség az adatbázisban, a program nagy részére kiterjedhet, és a módosítás sok időbe telhet. Az adatbázist ISZN névvel láttam el, mely az Informatikai Szakmai Napok rövidítése. A táblák és azokban található mezőnevek beszédesek, ezzel is megkönnyítve az adatbázis kezelést.

Az adatbázis négy darab táblát tartalmaz, név szerint: Diak, Ell, Eloadas, Tanar.

A Diak tábla tartalmazza a diákok legfontosabb adatait: Név, évfolyam, szak, diákigazolvány szám. A tábla az elő regisztráció miatt akár a rendezvény előtt is fel lehet töltve adatokkal, és a rendezvény során is kaphat adatokat. A Diak tábla az alábbi mezőket tartalmazza, a következő adattípusokkal és megszorításokkal:
vezeteknev: 30 karakter hosszú karakter sorozat, kitöltése kötelező.
keresztnev: 30 karakter hosszú karakter sorozat, kitöltése kötelező.
evfolyam: szám típus, kitöltése kötelező.
szak: 6 karakter hosszú karakter sorozat, kitöltése kötelező.
diakig: 10 karakter hosszú karakter sorozat, kitöltése kötelező, elsődleges kulcs a táblában.

Az Ell tábla segítségével tartjuk nyílván az óralátogatásokat. A tábla adatokkal a rendezvény alatt töltődik fel. A tábla az alábbi mezőket tartalmazza, a következő adattípusokkal és megszorításokkal:
diakig: 10 karakter hosszú karakter sorozat, kitöltése kötelező, elsődleges kulcs a táblában.
eloadaskod: 11 karakter hosszú karakter sorozat, kitöltése kötelező, elsődleges kulcs a táblában.
A *diakig* és az *eloadaskod* mezők együtt véve elsődleges kulcsok, így elérhető az, hogy egy diakigazolvány szám és egy előadáskód páros csak egyszer fordulhasson elő, viszont a külön külön a diákigazolvány és az előadás kód többször is előfordulhassanak.

Az Eloadas tábla tartalmazza a rendezvény összes előadását. A tábla tartalma előre feltöltött. ha a tábla üres a Ell táblát nem tudjuk feltölteni. A tábla az alábbi mezőket tartalmazza, a

következő adattípusokkal és megszorításokkal:

eloadaskod: 11 karakter hosszú karakter sorozat, kitöltése kötelező, elsődleges kulcs a táblában.

cim: 30 karakter hosszú karakter sorozat, kitöltése kötelező.

eloadokod: 10 karakter hosszú karakter sorozat, kitöltése kötelező.

idopont: Dátum tipusú, év. hónap. nap. óra:perc formátumú, kitöltése kötelező.

terem: 10 karakter hosszú karakter sorozat, kitöltése kötelező.

Az Eloado tábla a rendezvényen részvevő előadók nevét, kódját, és cégének nevét tartalmazza. A tábla feltöltése a legelső, ezután lehet csak tölteni az adatbázis Eloadas tábláját. A tábla feltöltése nélkül az adatbázis feltöltése nem lehetséges. A tábla a következő mezőket, adattípusokat, és megszorításokat tartalmazza:

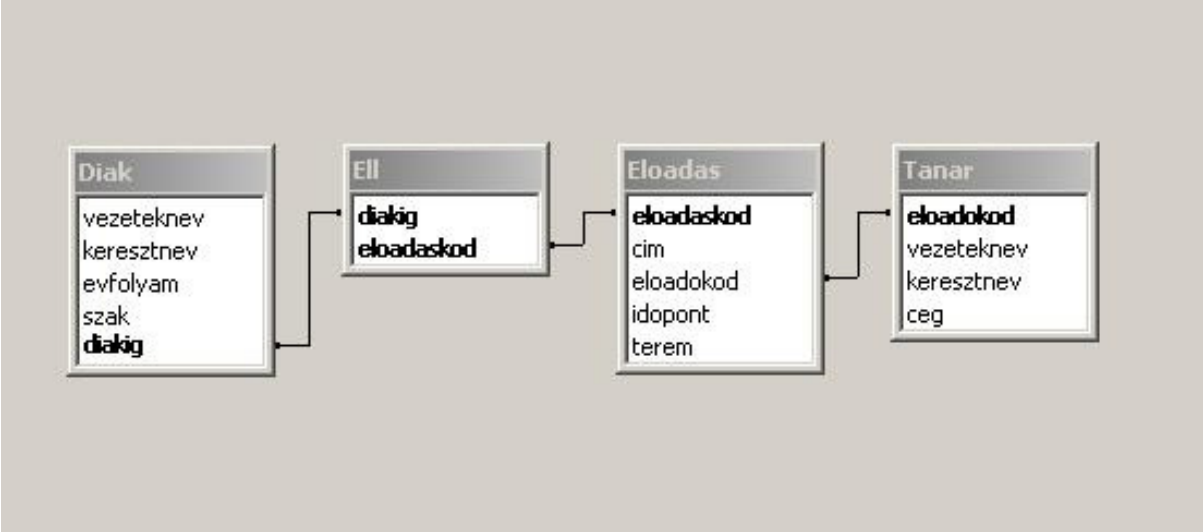
eloadokod: 10 karakter hosszú karakter sorozat, kitöltése kötelező, elsődleges kulcs a táblában.

keresztnev: 30 karakter hosszú karakter sorozat, kitöltése kötelező.

vezeteknev: 30 karakter hosszú karakter sorozat, kitöltése kötelező.

ceg: 50 karakter hosszú karakter sorozat, kitöltése kötelező.

Az ISZN adatbázis tábláinak kapcsolatát az alábbi ábra segítségével mutatnám be:



(Az ISZN adatbázis tábláinak kapcsolata)

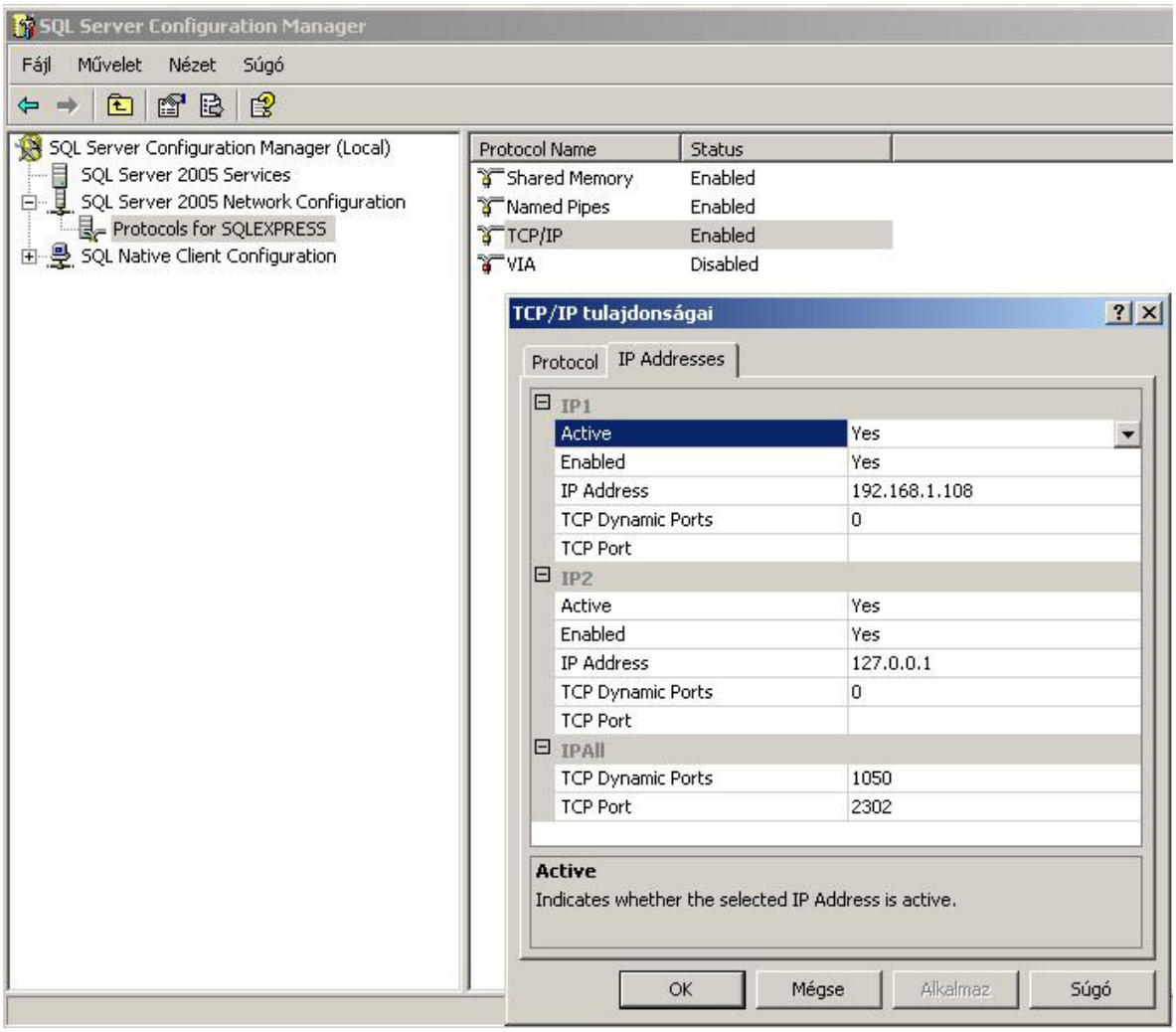
A Diak tábla az Ell táblához kapcsolódik. A kapcsolat a Diak diakig az Ell tábla diakig mezők között jött létre. Kapcsolat típusa egy a többhöz kapcsolat.(1:N). A Tanár tábla az eloadokod mezőn keresztül kapcsolódik az Eloadas tábla eloadokod mezőjéhez. Kapcsolat

típusa egy a többhöz kapcsolat.(1:N). Az Eloadas tábla az eloadaskod mezővel kapcsolódik az Ell tábla eloadaskod mezőjéhez. Kapcsolat típusa egy a többhöz kapcsolat.(1:N).

A ISZN adatbázis táblái között a kaszkádolt törlés opció be van kapcsolva, ami azt jelenti ha az egyik elsődleges kulcsú táblából törlődik egy adat, akkor az a külső kulcsot tartalmazó táblából is törlődni fog.

3.1.2. Az adatbázis szerver beállítása

Miután az adatbázist létrehoztam be kellett állítanom a szervert hogy azt akár kívülről, más hálózatból is elérjem, ne csak a lokális hálózaton. Először is az SSME (SQL Server Management Studio) segítségével beléptem a szervert futtató gépen Windows autentikációval a szerverre, ahol egy új Login-t hoztam létre. (adatbázis\security\logins; New Login) Egyik legfontosabb beállítás az új Logint beállítani SQL Server autentikációra, és egy megfelelő jelszóval ellátni. Miután ez megvolt a Loginhoz hozzárendelem a megfelelő jogosultságokat, azután hogy milyen adatbázisokhoz férhet hozzá, és státuszt állíthatunk be, majd mindezt lementjük. A változások életbe lépéséhez újra kell indítani az adatbázis szervert. Ezzel azonban még nincs vége.



(TCP/IP beállítások)

A szerver eléréséhez a szerver TCP/IP beállításokat engedélyezni kell, a megfelelő IP címmel és TCP porttal együtt. Ezeket megtehetjük Configuration Management elindításával, jobboldalt a Protocols for SQLEXPRESS, menü alatt található TCP/IP beállítások alatt, ahogy ezt a fenti ábra is mutatja. Mindezek után ellenőrizni kell a Tűzfalat, hogy ne blokkolja a szervert, és kívülről is el tudjuk érni azt. Ezt könnyen ellenőrizni is tudja az SSME segítségével. Server name helyére az elérési utat adjuk meg, ami lehet IP cím (publikus) vagy egy DNS is. Az autentikációt állítsuk át Windows autentikáció helyett SQL Server autentikációra, és adjuk meg az általunk létrehozott felhasználói adatokat; felhasználói név, jelszó.

Ha így elérjük szerverünket, a szerver külső címről elérhető. (Ha az SQL szerverünk router mögött van a routeren még néhány beállítás szükséges, mint például Port forwarding,

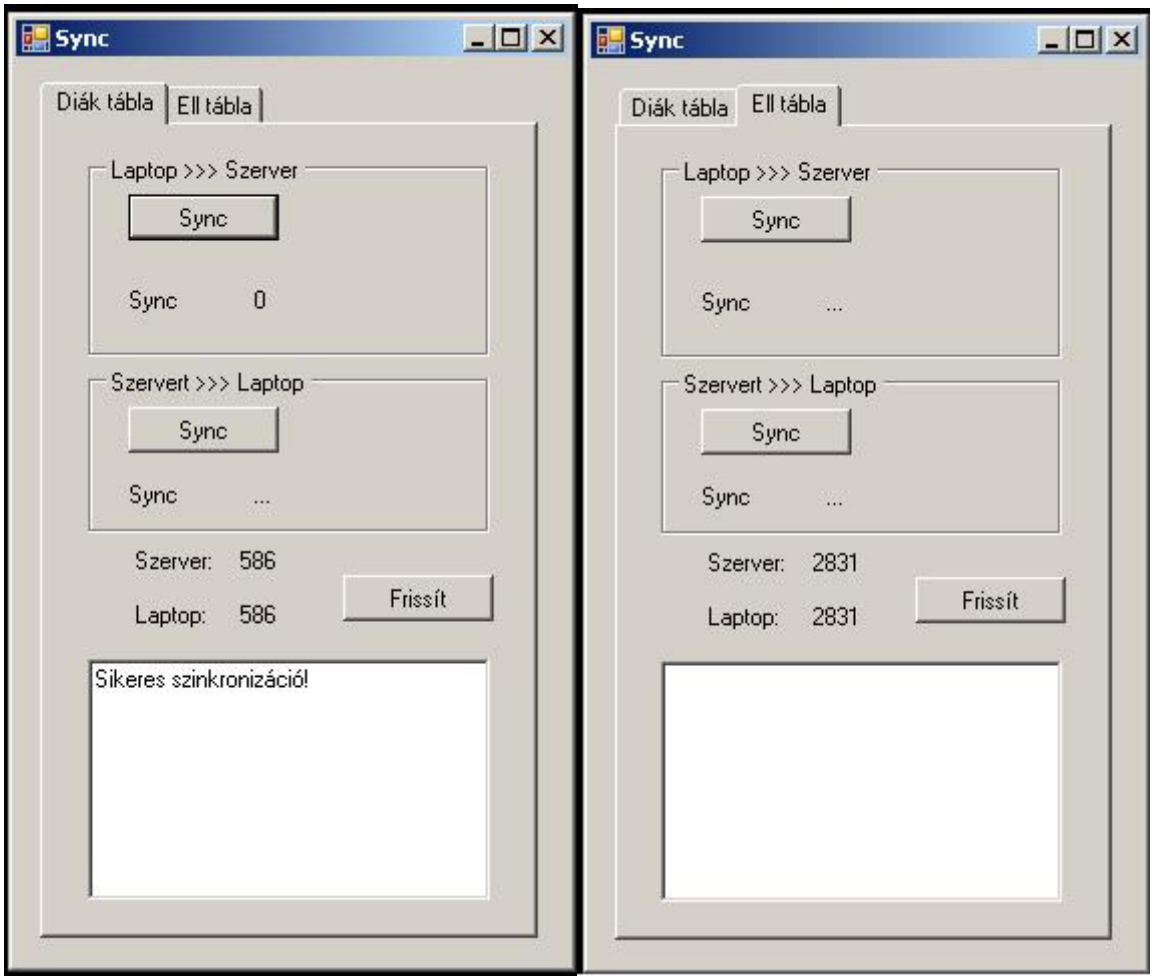
ami segítségével belső hálózation, nem publikus IP címmel ellátott szervert is elérhetünk.)

3.1.3. Szinkronizáció adatbázisok között

Az előkészületek során felmerültek olyan kérdések és gondok, hogy a Szakmai Napok alatt olyan helyen is lesznek előadások, ahol nincs hálózati kapcsolat, így a szervert közvetlenül nem lehet elérni. (Mint később kiderült sajnos egyáltalán nem volt kapcsolat, így a következő részben leírt megoldás tökéletes, és problémamentes volt.)

Mivel az adatbázisunk már a Szakmai Napok előtti héten, a webes elő regisztráció miatt folyamatosan töltődött, az a Web host-ot ellátó szerveren volt tárolva, míg magán a Szakmai Napokon is szükség volt egy adatbázisra, mivel ezt a szervert nem tudtuk elérni. A megoldás egyszerű: a szerveren és a rendezvény alatt szolgáló laptopon ugyanazt az adatbázist hoztuk létre, ugyanazokkal a beállításokkal, és paraméterekkel. Két tartalmilag dinamikusan változó táblánk van az adatbázisban, ezek a Diak, és az Ell táblák, így csak ezeknek a tábláknál volt szükségünk szinkronizációra. Az Előadás és az Előadó táblák már előre feltöltött táblák voltak, amik tartalma nem változott, így itt nem volt szükség a táblák szinkronizációjára. Az SSME (SQL Servver Management Studio) segítségével ezeket a táblákat egyik adatbázisból a másikba egyszerű másolási művelettel áttudtuk másolni. Amit viszont nem tudott az SSME az az adatbázisok közötti szinkronizáció. Így Visual Studióban írnom kellett egy programot, ami ha van hálózati kapcsolat, és elérem az adatbázis szervert a Diák és az Ell táblákat szinkronizálja.

A következő két kép a szinkronizációs program két táblájának szinkronizációs felületét mutatja be.



(Az adatbázis szerverek közötti szinkronizáló program)

Ahogy a fenti képeken is láthatjuk az egyszerűség és nem az esztétika a fő mérve a szinkronizációs programnak. Az ablak négy elkülönülő részre osztható, laptopról szerverre, szerverről laptopra történő szinkronizáció, laptopon és szerveren található adatok száma, és egy szövegdoboz, ahova az esetleges hiba üzenetek kerülhetnek ki, mint például, hogy nem tud kapcsolatot létesíteni az adatbázis szerverrel.

Ahogy elindítom a programot látom, hogy melyik adatbázisban mennyi adat található. Amint láthatjuk az Diák tábla szinkronizációs fülnél a szerveren és az laptopon található adatbázisok ugyanazokat, és ugyanannyi adatot tartalmaznak. Ezt az is igazolja hogy a laptopról megkísérelt a szerver felé történő adategyeztetés során nem történt semmilyen adatcsere, „Sync 0”, és hibaüzenetet nem kaptunk vissza, csak annyit, hogy „Sikeres szinkronizáció”.

A háttérben a következő dolgok zajlanak le a Diák tálba szinkronizációja során:

veszi a tábla legelső sorát, kivesszük belőle a diákigazolvány számot, mivel az egyedi és csak is egy lehet belőle. Miután ez megtörtént az adott diákigazolvány számot keressük a másik adatbázis Diák táblájában. Ha ott nem található ez, akkor a diák igazolványszámot a hozzá tartózó adatokkal együtt (vezetéknév, keresztnév, évfolyam, szak) beilleszti az adatbázisba. Ha már az adott adatbázisban létezik ilyen diákigazolvány tovább lépünk a ciklusban és megyünk, míg van diákigazolvány szám.

Az kódrészlet a következő képen néz ki.

```
szerver.Open();
laptop.Open();

        cmdstrlaptop = " SELECT * FROM Diak";
     cmdlaptop = dblaptop.sqlcommand(cmdstrlaptop, laptop);
     SqlDataReader myReader = null;
     myReader = cmdlaptop.ExecuteReader();
    while (myReader.Read())
    {
            vnev = myReader["vezeteknev"].ToString();
        knev = myReader["keresztnev"].ToString();
            evf = myReader["evfolyam"].ToString();
        szak = myReader["szak"].ToString();
            diakig = myReader["diakig"].ToString();
            if (keres(diakig) == false)
            {
             cmdstrszerver = "INSERT INTO diak
             (vezeteknev,"+ keresztnev, evfolyam, szak,
             diakig)" + "Values ( '" + vnev + "', '" + knev
             + "', '" + evf + "', '" +szak + "', '" + diakig
             + "')";
            cmdszerver = dbszerver.sqlcommand(cmdstrszerver,
            szerver);
            cmdszerver.ExecuteNonQuery();
            label2.Text = Convert.ToString(++synced);
```

```
                }
                else { }
            }
```

A kereső kódrészlet:

```
szerver.Open();
cmdstrszerver = " SELECT COUNT(diakig) FROM Diak WHERE diakig
= '" + diakig + "'";
cmdszerver = dbszerver.sqlcommand(cmdstrszerver, szerver);
int db = (int)cmdszerver.ExecuteScalar();
if (db == 0)
   {
        szerver.Close();
        return false;
   }
   else
   {
        szerver.Close();
        return true;
   }
```

Az Ell tábla szinkronizációja is ezen az elven működik azt leszámítva, hogy ott két elsődleges kulcsunk van, amit vizsgálnunk kell, mégpedig a diákigazolvány szám és az előadás kód.

Az SQL parancsok:

```
INSERT INTO ell (diakig,eloadaskod) Values ('diakig'
,'eloadaskod');
SELECT COUNT(diakig) FROM ell WHERE diakig = 'diakig' and
eloadaskod='eloadaskod';
```

3.2. Windows Form-os (adminisztrátor) felület

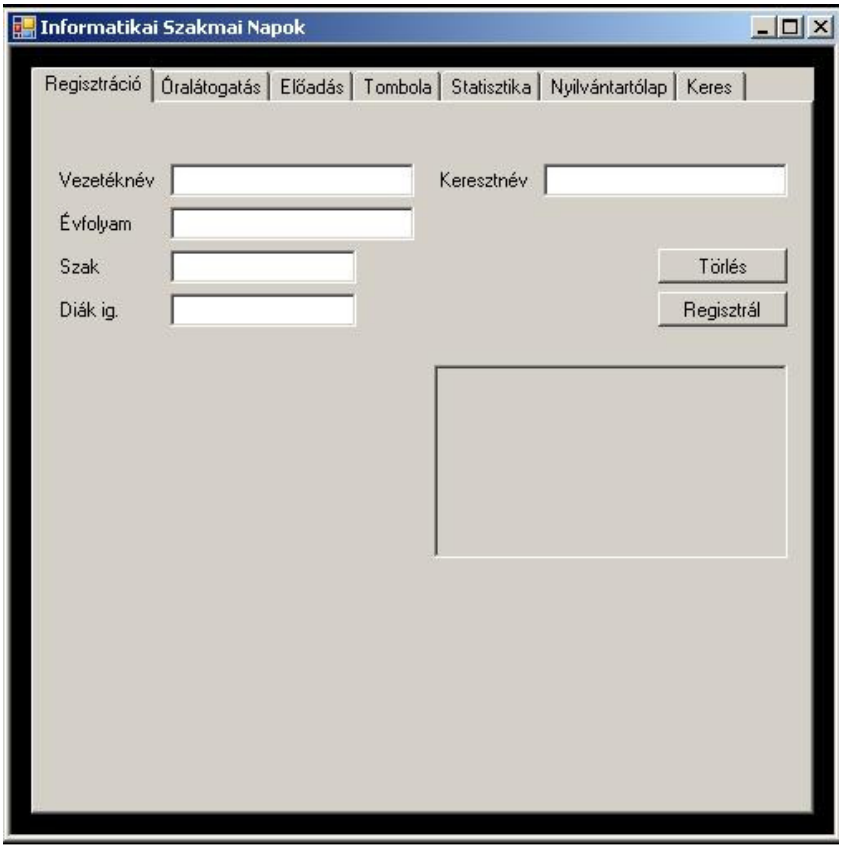
3.2.1. Felületek bemutatása

Az általam Adminisztrátor felületnek elnevezett Windows Form-os felület egy több célú program. Kezdve az előadók, előadások bevitele, a résztvevők helyszíni regisztrációja, előadás látogatások ellenőrzésén, adminisztrálásán keresztül a nyilvántartó lapok

legenerálásáig, sokféle feladatot képesek vagyunk vele elvégezni. Hogy milyen opciók kerüljenek bele az elmúlt két rendezvény tapasztalataiból szedtem össze és igyekeztem a saját, mint rendezvényszervező és a többi rendezvényszervező feladatát megkönnyíteni, nem is beszélve a résztvevőkről.

Az adminisztrációs felület c# nyelven íródott, a Visual Studio fejlesztői környezetben. Maga a fejlesztést addig nem kezdtem el, míg fel nem mértem a pontos igényeket, az adatbázist meg nem terveztem, és létre nem hoztam.

Első és egyik legfontosabb dolog, amit tudni kellett a felületnek, az a résztvevők beregisztrálása volt. Ami a következő felületen keresztül történik:



(Regisztrációs felület)

Akik Webes felületen nem regisztráltak be előre, azok a helyszínen is megtehetik ezt. Régebben, csak a helyszínen lehetett regisztrálni, ami papír alapon működött, és a Neptun kód alapján történt az azonosítás. A regisztrálónak csak a nevét, évfolyamát, szak nevét, és diákigazolvány számát kellett megadnia. Szak neve, és évfolyama a későbbi statisztikák elkészítése miatt volt rá szükség, és persze a hallgatók pontosabb azonosítása érdekében. A

regisztráció gyorsítása érdekében a diákigazolványokat egy vonalkód segítségével olvasom le. Magáról a vonalkód technikáról, és vonalkód olvasó beállításairól a fejezet végén a 3.2.1 fejezetben beszélnék bővebben.

A regisztrációs felület kialakítása, mint ahogy az egész adminisztrátor felület kialakítása egyszerű, lényegre törő. Nem a kinézet volt az előtérben a tervezés és a fejlesztés folyamán.

A regisztrációs felületen történő regisztráció során kitöltjük az egyes szövegdobozokat, majd ha megvagyunk a „Regisztrál" gombra kattintunk. Ha sikeres volt a regisztráció az alsó nagyobb szöveges dobozban „Sikeres regisztráció" láthatjuk kiírva fekete betűkkel. Ez azért is fontos hogy milyen a kiírt üzenet színe, mert ha sikertelen a regisztráció, vagy bármi más hibaüzenetet kapunk pirossal írja ki, ezzel megkönnyítve a regisztrálást, és jóval szembetűnőbb. „Sikertelen regisztráció" üzenetet akkor kapjuk, ha nem töltöttünk ki egy mezőt, esetleg nem helyesen, vagy ha az adatbázis szerverhez nem sikerült csatlakozni, esetleg a résztvevő már egyszer regisztrált. Nem helyesen kitöltött mező esetén a következő hibaüzeneteket kaphatjuk: „Az évfolyam nem szám!", „A diákigazolvány nem szám!", „A diákigazolvány nem 10karakter!". Ha egy mező üresen marad: „A vezetéknév hiányzik!", „A keresztnév hiányzik!", vagy éppen az a mező ami kitöltetlen. Ha a diákigazolvány már regisztrált a következő hibaüzenettel találkozunk: „Már regisztrált!". Ha az adatbázis szerverhez való csatlakozásnál lép fel hiba azt a program írásánál kivételként lekezeltem, majd a hibaüzenetet kiírattam.

Ha a „Regisztrál" gombra kattintunk a következő dolgok történnek a háttérben. Sorban vesszük a mezőket, és mindet egyesével megvizsgáljuk, megfelelnek-e a kritériumoknak. Ezek a kritériumok lehetnek például, kitöltött mező, évfolyam szám típus, diákigazolvány szám típus, és 10 karakter hosszú. Ha a kritériumoknak eleget tettek a mezők csatlakozunk az adatbázis szerverhez, ahol megvizsgáljuk hogy az adott diákigazolvány szám megtalálható-e az adatbázisban, mert ha igen az azt jelenti, hogy már regisztrált, ha nem akkor minden rendben van és mehetünk tovább, majd a megfelelő adatokkal feltöltjük a Diák táblánkat. Ha ez is megtörtént, és nem dobott kivételt a program, a kapcsolatot lezárjuk. Ha kivételt kapunk, azt hibaüzenet formájában kiírja, majd zárja a kapcsolatot az adatbázis szerverrel. Az adatbázis szerverrel történő csatlakozásnál mindig nyitni és zárni kell a kapcsolatot. Ha a kapcsolatot nem nyitjuk meg az SQL műveletek nem hajtódnak végre, ha nem zárjuk le a slot-ok nyitva maradnak ezzel terhelve az adatbázis elérhetőségét.

Egy rövid kódrészlet a kapcsolat megnyitásától az adatbázisba való bevitelen keresztül, a kapcsolat lezárásáig, és kivételek lekezelése:

```
try
{
      myConnection.Open();
      SqlCommand myCommand = new SqlCommand("SELECT
      COUNT(diakig) FROM Diak where diakig='" +
      textBox5.Text+"'", myConnection);
      int db = (int)myCommand.ExecuteScalar();
      if (db == 0)
      {
            myCommand = new SqlCommand("INSERT INTO diak
            (vezeteknev, keresztnev, evfolyam, szak, diakig)" +
            "Values ( '" + textBox1.Text+ "', '" + textBox2.Text
            + "', '" + textBox3.Text + "', '" +textBox4.Text +
            "', '" + textBox5.Text + "')", myConnection);
            myCommand.ExecuteNonQuery();
      }
      else
      {
            richTextBox1.Text = "Már regisztrált!\n";
            myConnection.Close();
            return false;
      }
      myConnection.Close();
      return true;
}
catch (Exception ex)
{
      richTextBox1.ForeColor = Color.Red;
      richTextBox1.Text = "Hiba történt:\n" + ex.Message + "\n";
      return false;
```

```
}
```

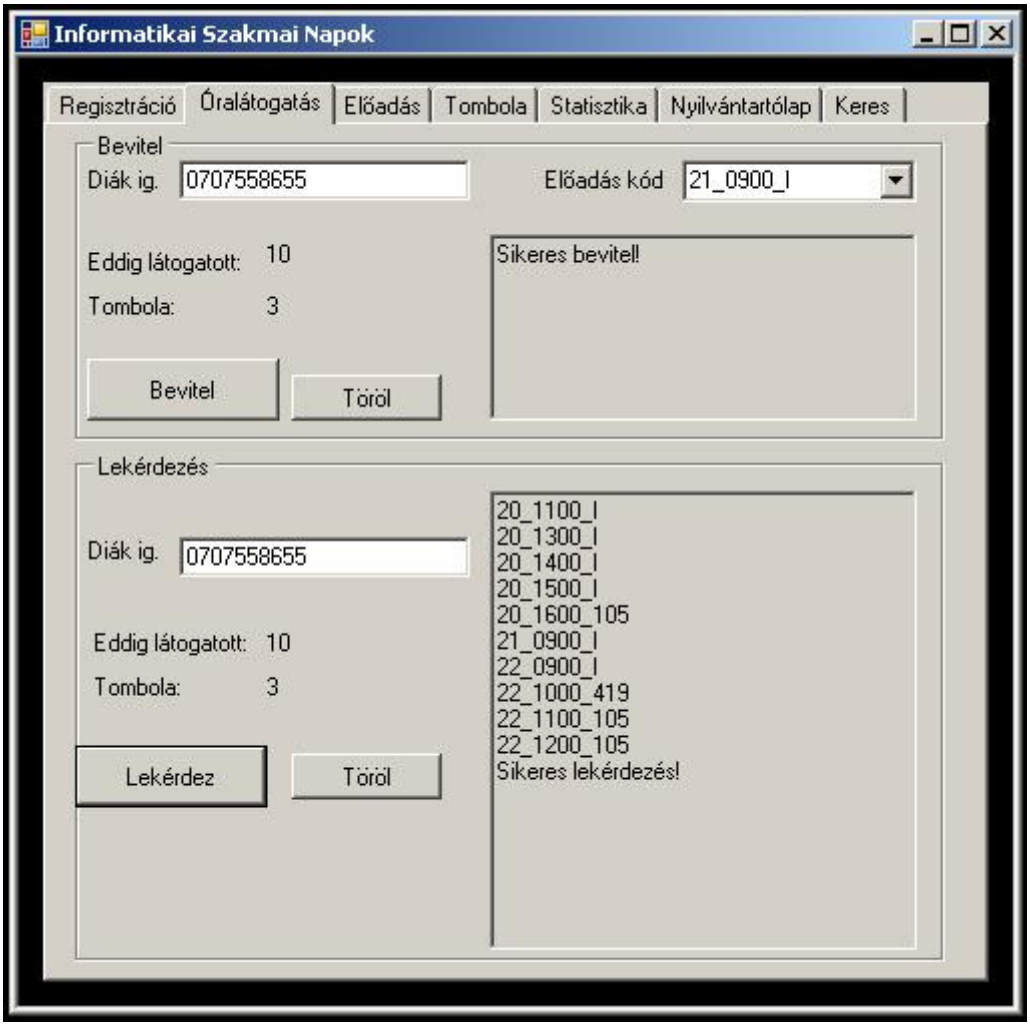
Az adatbázishoz való csatlakoznál megemlíteném a *Connection string*-et, ami tartalmazza az adatbázis szerver elérhetőségét, és a kapcsolathoz szükséges adatokat. Nézzük meg a következő ilyen stringet.

```
„Server=myServerAddress; Database=myDataBase; User ID=
myUsername; Password=myPassword; Trusted_Connection=False;
connection timeout=sec;”
```

Megadjuk a szerver elérhetőségét, ez lehet IP cím, vagy egy DNS, vagy akár a saját gépünkön lévő szerver amit a gépnél/szervernévvel érünk el. „Database=” után adjuk meg, hogy az adatbázis szerveren melyik adatbázishoz csatlakozzunk. Csak azokhoz az adatbázisokhoz csatlakozhatunk, amikhez van jogosultságunk. „User ID=” után, ha nem Windows autentikációról beszélünk, kerül a felhasználói nevünk, „Password=” után pedig a jelszavunk.

A regisztráció gomb alatt találhatunk egy „Töröl” gombot, amit lenyomva a törlődik az összes szöveges dobozunk tartalma. Ezután regisztrálhatjuk a következő résztvevőt.

A következő fontos opció, amit az adminisztrátor felületen található az előadás látogatások nyilvántartása.



(Előadás látogatás nyilvántartása)

A regisztráció után az egyik legfontosabb a rendezvényre regisztráltak előadás látogatásának nyilvántartása. Erre szolgál az adminisztrátor felület Óralátogatás fül. A felület két jól elkülönülő részre osztható. Egyik rész az óra látogatás bevitele, a másik alul elhelyezkedő rész pedig a gyors lekérdezésre szolgál, ami segítségével letudjuk kérdezni, hogy a regisztrált résztvevő mennyi előadást látogatott eddig, mennyi tombolát szerzett eddig, és hogy milyen előadásokon vett részt és mik ezen előadások kódjai

Vegyük előbb az előadások látogatásának bevitelét. Egy groupbox-ban vannak elhelyezve a bevitelhez szükséges eszközök. Egy beviteli mező, amibe a diákigazolványt visszük be, vagy a gyors bevitel érdekében vonalkód olvasóval visszük be a diákigazolvány számokat. Jobbra a beviteli mező mellett találunk egy ListBox-ot, azaz egy legördülő menüt, ahol az előadás kódok találhatóak. Az előadás kódok legenerált kódok, viszont beszédesek. Nézzük példaként a 22_1000_m419 kódot. A kód fontosabb részeit egy alulvonás jel „_”

választja el. A kód NAP_ÓRAPERC_TEREM formára épül fel. A legördülő menü elemeit vehetjük közvetlenül az adatbázisból, vagy akár mi is feltölthetjük adatokkal. Miután megadtuk a diákigazolvány számot, és beállítottuk az előadást a „Bevitel” gombra kattintva az alábbi dolgok történnek. Először is a megvizsgáljuk, hogy a mezők nem-e üresek, ha üres „Nincs Diákigazolvány szám!” vagy „Nincs előadás kód kiválasztva!” és „Sikertelen bevitel!” üzenetet kapunk a szövegdobozban. A hibaüzenetek itt is piros, jól feltűnő színnel jelennek meg, ugyanis itt még inkább fontos az hogy jól látható legyen, mert egy előadáson több hallgató is lehet, és ezeket minél hamarabb kell az adatbázisba bevinni. Ha nem történik hiba a szöveges dobozban mindig csak fekete színnel „Sikeres bevitel!” jelenik, meg. Viszont ha bármi hiba van a piros szín miatt, azonnal feltűnik, még ha nem is vesszük észre magunktól. Ha a mezőkben van adat azt vizsgáljuk, hogy a diákigazolvány szám mezőjében lévő adat megfelel-e a diákigazolvány kritériumainak, mégpedig 10karakter hosszú szám. Ha nem felel meg szintén hibaüzenetben figyelmeztet minket: „A diákigazolvány szám nem szám! Sikertelen regisztráció!”. Ezek után a csatlakozunk az adatbázishoz, ahol első lépésben megvizsgáljuk hogy a résztvevő egyáltalán regisztrált-e, azaz a diákigazolvány száma megtalálható a Diák táblában, ha nem akkor „Még nem regisztrált!” hibaüzenetet kapunk, ha már regisztrált tovább megyünk és vizsgáljuk, hogy volt-e már ezen az órán, azaz hogy a diákigazolvány, előadáskód páros előfordul-e az Ell táblában. Ha előfordul „Már volt ezen az órán!” hibaüzenetet kapunk. Ez azt jelenti hogy a diákigazolvány számát már egyszer leolvastuk, mehetünk a következő diákigazolvány számot leolvasni. Ha még nem szerepel a diákigazolvány szám, előadáskód páros az Ell táblában akkor a következő lépésként beillesztjük azokat. A diákigazolvány beviteli mező alatt a sikeres bevitel után megkapjuk hány előadáson volt összesen, és mennyi tombolát szerzett eddig a hallgató. Három előadásonként kapnak egy tombolát. Ezek után az adatbázis szerverrel való kapcsolatot lezárjuk.

Sikeres bevitel után a „Töröl” gombra kattintva a diákigazolvány beviteli mező tartalma törlődik, de az előadás kód tartalma változatlan marad, ezzel is megkönnyítve a gyors bevitelt, így egymás után gyorsan lehet a diákigazolvány számokat beírni, vagy vonalkód olvasóval a vonalkódot leolvasni.

Az óralátogatás fül alsó részén található a lekérdezés GroupBox, aminek elemei egy diákigazolvány bevitelére alkalmas mező, „Lekérdez”, és „Töröl” gombok, egy szövegdoboz, és két Label, amik a látogatott előadások és tombolák számát írja ki. Itt gyorsan

lekérdezhetjük ki mennyi előadáson volt, és a kódok alapján azt is tudjuk mik ezek az előadások.

A lekérdezés a következő képen folyik. Beírjuk a diákigazolvány számot, aminek az eddigiekben tárgyalt kritériumoknak meg kell felelni. Ezek a kritériumok az előadás kód kritériumaitól elvonatkoztatva ugyanazok, mint az előadás látogatás nyilvántartásának diákigazolvány beviteli mezőjére vonatkozó kritériumok. Ezután csatlakozunk az adatbázis szerverhez, ellenőrizzük van-e ilyen diákigazolvány szám, ha van a lekérdezéseket végrehajtjuk, ha nincs hibaüzenetet kapunk. Mindkét esetben az adatbázis szerverrel való kapcsolatot lezárjuk.

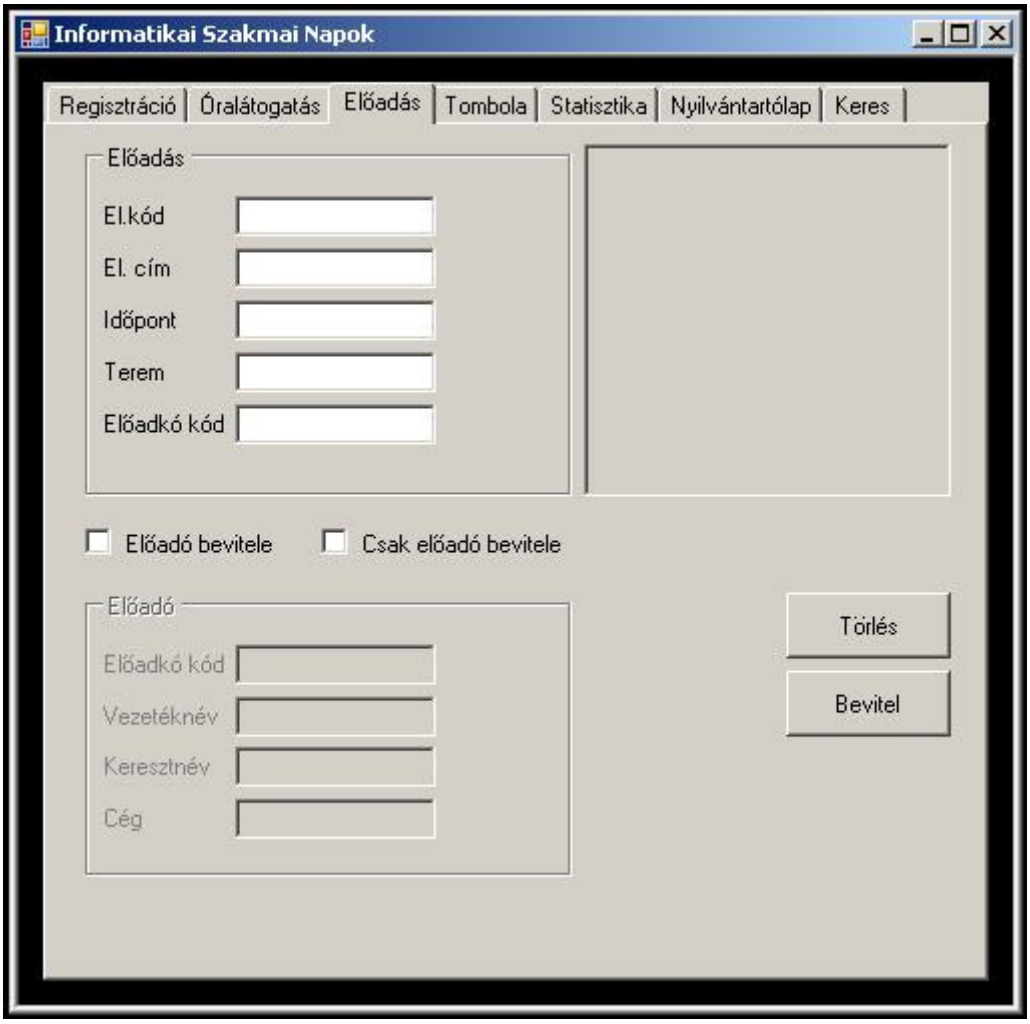
A következő választható fül az „Előadás” fül, az Adminisztrátor programban. A felület szerepe az előadások és előadók bevitele, akár együtt, akár külön-külön.

Ahogy a lenti képen látható az ablak felső része az előadások bevitelére, az alsó része az előadók bevitelére szolgál. Közte két CheckBox van elhelyezve. Alapértelmezettként az előadásokat tudjuk bevinni az adatbázisba, ha az „Előadó bevitele” négyzetet úgymond bepipáljuk, az ablak alján az előadó bevitelére alkalmas mezők is aktívvá válnak, az Előadás részből pedig az „Előadó kód” mező inaktív lesz. A közképen elhelyezkedő másik CheckBox segítségével csak előadót viszünk be az adatbázisunkba, azaz a felső Előadás GroupBox elemi inaktívak lesznek, még az Előadó GroupBox elemei aktívak lesznek. A két CheckBox össze van hangolva, így akár össze-vissza kattintva őket sem találjuk a programot értelmetlen helyzetben.

Ha egy GroupBox aktív és adatot akarunk bevinni értelemszerűen ki kell tölteni a mezőket. Hiányos, vagy hibás mező esetén hibaüzenetet kapunk. Az előadás kódok a már leírt módon nap_óraperc_terem formátum alapján állnak össze. Az előadó kód el(sorszám), például el01 az első beregisztrált oktató kódja. Az időpont mező *év.hónap.nap óra:per (másodperc)* formátumú. Helytelen formátumú beírása esetén hibaüzenetet kapunk, ugyanis az adatbázis kezelőben a beállított formátumhoz nem tudjuk beilleszteni a kívánt adatot, üresen mező pedig nincs engedélyezve.

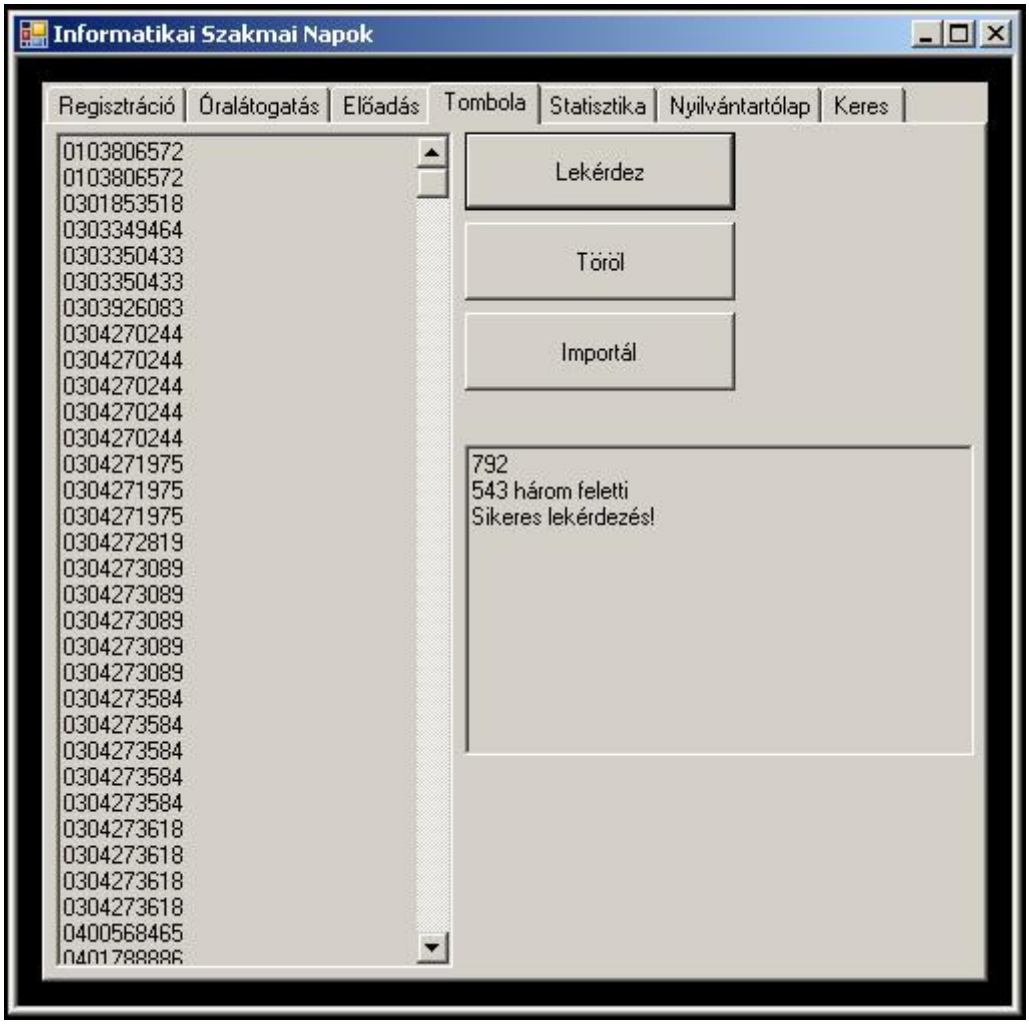
A felület működési elve a „Bevitel” gombra kattintva ugyanaz, mint az eddigi felületeké. Beviteli mezők ellenőrzése, csatlakozás az adatbázishoz, elsődleges kulcsok egyediségének vizsgálata, azaz hogy regisztráltuk-e már az előadást, vagy az előadót, vagy nem. Aztán adatok bevitele az Előadás és az Előadó táblába, majd kapcsolatot lezárjuk.

„Töröl” gomb segítségével a beviteli mezőket töröljük.



(Előadás és előadó bevitelére alkalmas felület)

Fontossági sorrend alapján a harmadik legfontosabb fül talán a „Tombola” fül lehetne. A felület lényege a hagyományos tombola húzás megtartása, viszont minden elektronikusan történik. A tombolák generálása, gyűjtése, eddig papíron történt, úgy hogy folyamatosan számolgattuk ki mennyi előadáson volt, ahogy megvolt a három látogatott előadása a nyilvántartó lap alapján egy kis cetlire ráírtuk a neptun kódját, és bedobtuk egy dobozba.



(Tombola legenerálására alkalmas felület)

Ez most teljesen másképp zajlott, kivéve a tombola húzást, csak csupán itt a diákok nem a neptun kódot hallották, hanem a diák igazolvány számukat, ha szerencsések voltak.

A tombolákat nem tároltuk külön adatbázisba, csupán egy lekérdezés segítségével összeszámoltuk diákigazolványonként hány előadáson volt és az eredményt osztjuk hárommal, majd a kapott eredményszer kiírtuk a diákigazolvány számot.

A kódrészlet a következő képen néz ki:

```
SqlCommand myCommand = new SqlCommand("SELECT diakig, Count(eloadaskod) AS \"COkod\" FROM Ell GROUP BY diakig", myConnection);

SqlDataReader myReader = myCommand.ExecuteReader();

int db=0;
```

```
while (myReader.Read())
 {
for(int i=0;
i<(Convert.ToInt32(myReader["COkod"].ToString())/3);i++)
        {
           richTextBox6.Text +=
           myReader["diakig"].ToString()+"\n";
           db++;
        }
  richTextBox7.Text =Convert.ToString(db)+"\n";
  }
```

Természetesen a legenerálás közben probléma lép fel hibaüzenetet azonnal megkapjuk.

A fenti képen láthatjuk a legenerált diákigazolvány számokat. Itt már egy diákigazolvány annyiszor szerepel, amennyi tombolája van. Jobb oldalt láthatjuk hány tombolánk is van, összesen 792 darab tombola szelvényt sikerült összegyűjteni a résztvevőknek a három nap alatt. Ezek kézzel való legyártásához és közben a szakmai napok rendezvényeinek lebonyolításához minimum egy tíz fős stábra volt szükség régebben, most két szervező elegendő volt, mivel a legenerált tombolákat, azaz a szöveges doboz tartalmát kiimportáltuk egy dokumentumba, kinyomtattuk, majd csak fel kell vágni.(Ez a folyamat szakmai napokon mindössze 3-4 percet vett igénybe az utolsó előadás után.) Ezzel a módszerrel kiiktattuk az emberi hibázás lehetőségét is, vagy akár a túlkapást. A kiimportálást az „Importálás” gombbal tehetjük meg, ami a c:\ meghajtóra helyezni a .doc kiterjesztésű dokumentumot. A következő kódrészlet ezt szemlélteti:

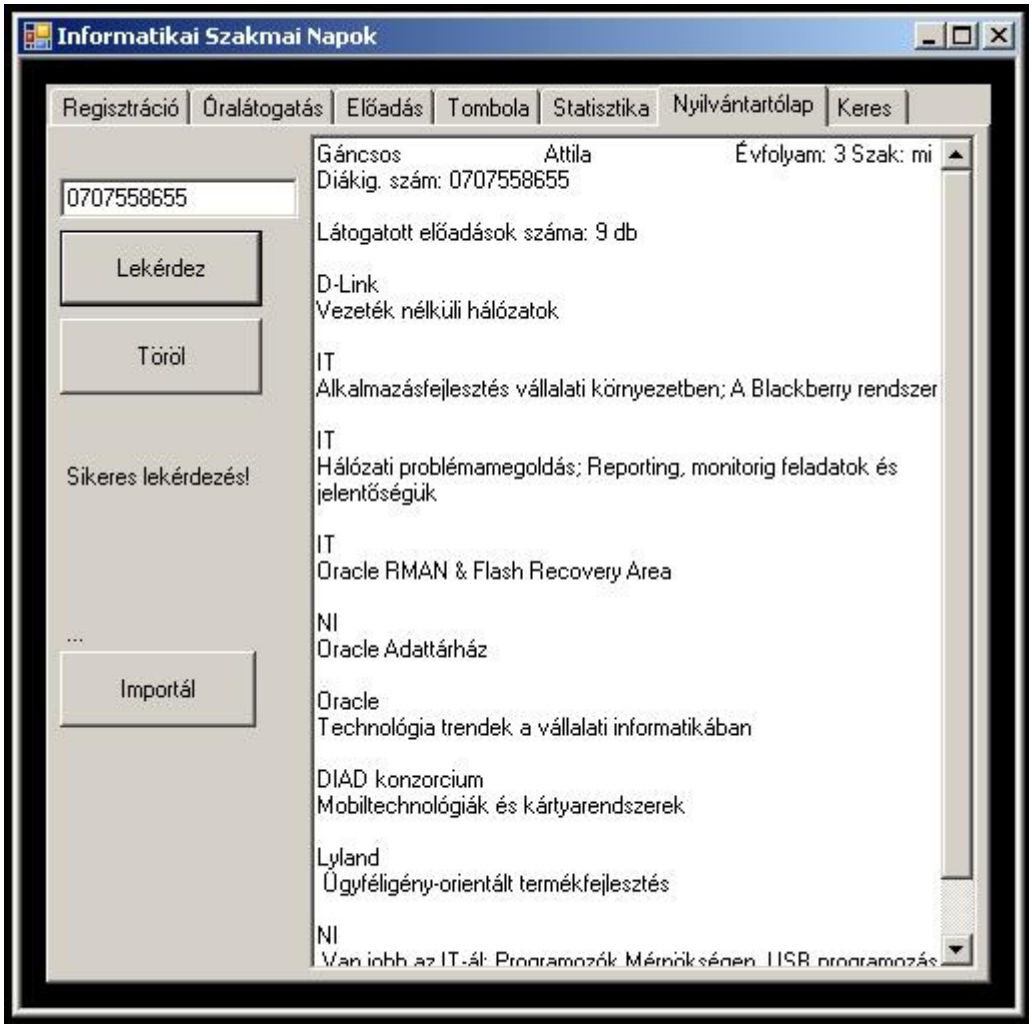
```
FileStream fs = new FileStream("c:\\tombola.doc",
FileMode.Create, FileAccess.Write);
StreamWriter st = new StreamWriter(fs);
st.WriteLine(richTextBox6.Text);
st.Close();
```

```
fs.Close();
```

Sikeres lekérdezés és importálás után a következő két sort kell hogy kapjuk jobb oldali szöveges dobozunkba: „Sikeres lekérdezés! Sikeres importálás!”

A „Töröl” gomb segítségével a mezőket törölhetjük.

Az informatikai szakmai napok többek között azért is ennyire sikeres, mert a résztvevő cégek bőkezűek tombolára felajánlott tárgyi nyeremények terén, és persze azért is mert a diákok a látogatott előadások száma után plusz pontokat és százalékokat kaphatnak tárgyaikból. A Tanároknak persze ezt ellenőrizni kell valahogy. Erre való a következő fül, a Nyilvántartó lap felület. A felületen egy diákigazolvány szám bevitelére alkalmas mező található. Egy, az ablak jelentős részét elfoglaló szöveges doboz, „Lekérdez”, „Töröl”, és „Importál” gombok. A felület a következő képen működik. A mezőbe bevisszük a diákigazolvány számot, majd a „Lekérdez” gombra kattintunk. Természetesen itt is ellenőrizzük, hogy a diákigazolvány szám valóban szám-e, vagy hogy megvan-e a kellő hosszúságú, vagy hogy a mező egyáltalán nem-e üres, csak ezek után csatlakozunk a szerverhez, ahol megvizsgáljuk hogy a résztvevő regisztrált-e. Bármilyen hiba esetén itt is hibaüzenetet kapunk. A hibaüzenetek lehetnek: „Nem regisztrált!”, „Hibás diákigazolvány szám!”, „A diákigazolvány szám nem tíz karakter!”, „Üresen hagyott mező!”.



(Nyilvántartó lap felület)

Miután minden előírt feltételeknek megfelelt a mező tartalma, először is a Diák táblában megkeressük a megfelelő diákigazolvány számhoz tartozó adatokat, majd ezeket ki is írjuk. Azután a Ell tábla segítségével megszámoljuk hány előadáson is volt a résztvevő, majd kiírjuk, majd a következő lépésben az Ell, Előadás és az Előadó tábla segítségével kiírjuk mik is voltak ezek az előadások, és hogy melyik cégtől is hallottuk. Ehhez a lekérdezéshez mind a három táblát egy SQL lekérdezésben kell összefoglalnunk. Maga az SQL lekérdezés a következő képen néz ki:

```
SELECT ell.diakig, Eloadas.cim, tanar.ceg FROM tanar INNER
JOIN (Eloadas Inner JOIN ell ON
Eloadas.eloadaskod=Ell.eloadaskod) ON tanar.eloadokod
=eloadas.eloadokod where (ell.diakig=' a beviteli mező
tartalma ');
```

Mindezeket az adatokat úgy iratom ki a szöveges dobozba, hogy az már különösebb formázás nélkül nyomtatható állapotban kerüljön. A formátum a következő:

Vezetéknév: *Keresztnév:* *Évfolyam:* *szak:*

Diákigazolvány száma:

Látogatott előadások száma:

Cég

Előadás címe

Cég2

Előadás címe

…

Igazolta: (szervező aláírása)

Ebben a formátumban a szöveges doboz tartalmát kiimportálhatjuk és nyomtathatjuk is azonnal. Az importálás a fájl nevétől eltekintve ugyanúgy működik mint a tombola szelvények importálása, azt a fontos dolgot beleszámítva hogy itt egy dokumentumba akár több hallgató adatait is kiimportálhatjuk és ezeket egyszerre nyomtathatjuk. Ezen kívül az is egy szem előtt tartott dolog volt hogy egy dokumentumba kiimportált hallgatók jelenléti íveit úgy is nyomtathatjuk hogy akár egy A4-es oldalra két oldalt nyomtatva, ezzel papír takarékosak is lehetünk. Fontos még többek között, hogy a szöveges doboz tartalma itt változtatható, azaz akár bele is írhatunk, ez az esetleg elgépelt nevek kijavítására szolgál.

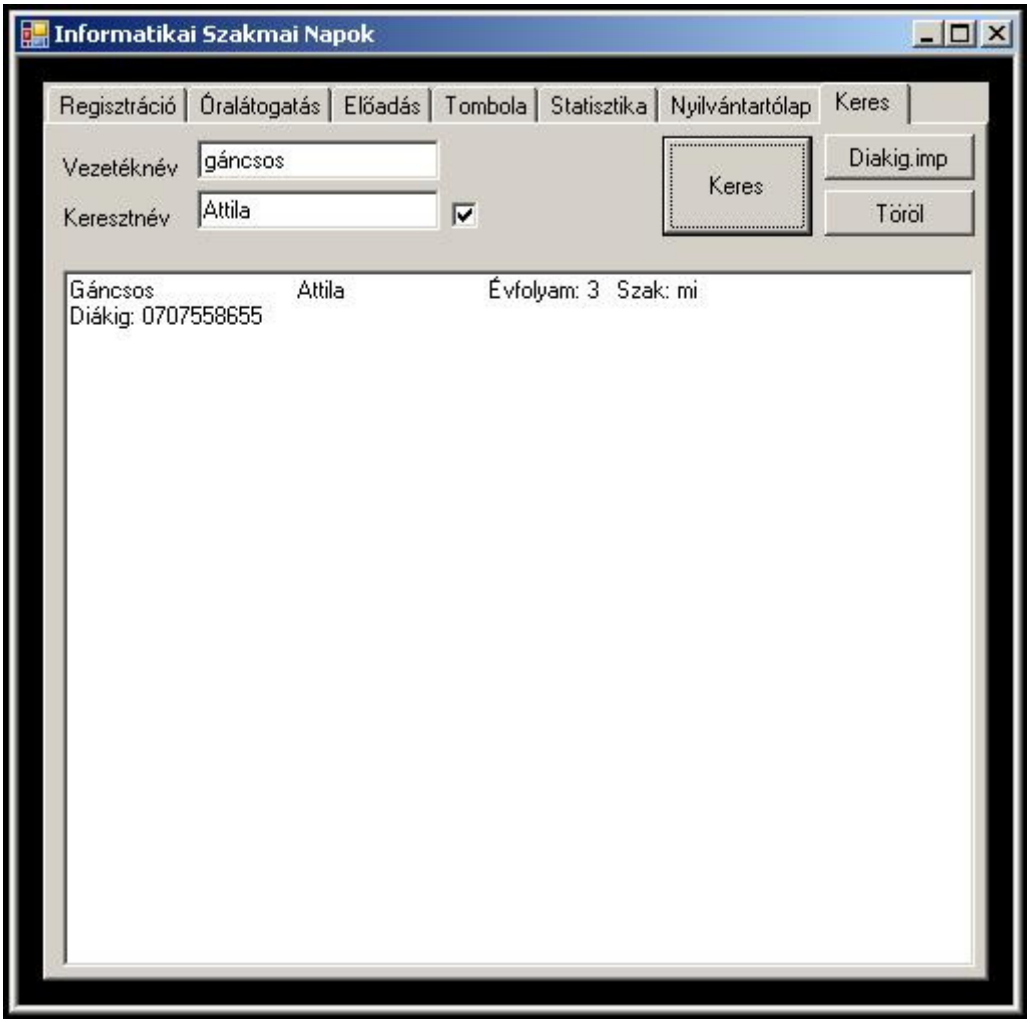
A „Töröl” gomb segítségével, mint az eddigiekben is az ablak összes mezőjének, szöveges dobozának tartalmát töröljük ki.

Az Adminisztrátor felület „Keres” felülete leginkább arra szolgál, hogy név alapján vissza tudjuk keresni az adatbázisból a diákigazolvány számot, és egyéb adatokat, mint például a szak és évfolyam. A felület azért is fontos, mert előfordult, hogy a diákigazolványt a

diákok elfelejtették elhozni, amikor a nyilvántartó lapokat osztottuk. Ilyen esetekben név alapján vissza tudtuk keresni, ha egyezés volt a szak, és az évfolyam ismerete mellett eltudtuk dönteni kit is keresünk.

A felületen a vezetéknév, és a keresztnév bevitelére alkalmas mezők, egy CheckBox „Keres”, „Diakig.imp”, „Töröl” gombok és egy az ablak jelentős felületét elfoglaló szöveges doboz, ahol a keresés eredménye, vagy több találat esetén eredményei jelennek meg.

A keresés a következő képen történik. Alapértelmezettként mind a két; vezetéknév, keresztnév, beviteli mező aktív. Keresésnél kereshetünk egyszerre a vezetéknévre és keresztnévre, ezzel szűkítve a találatok számát, vagy akár kereshetünk csak a vezetéknév alapján, viszont ilyenkor előfordulhat hogy egy vezetéknév többször is megtalálható az adatbázisban.

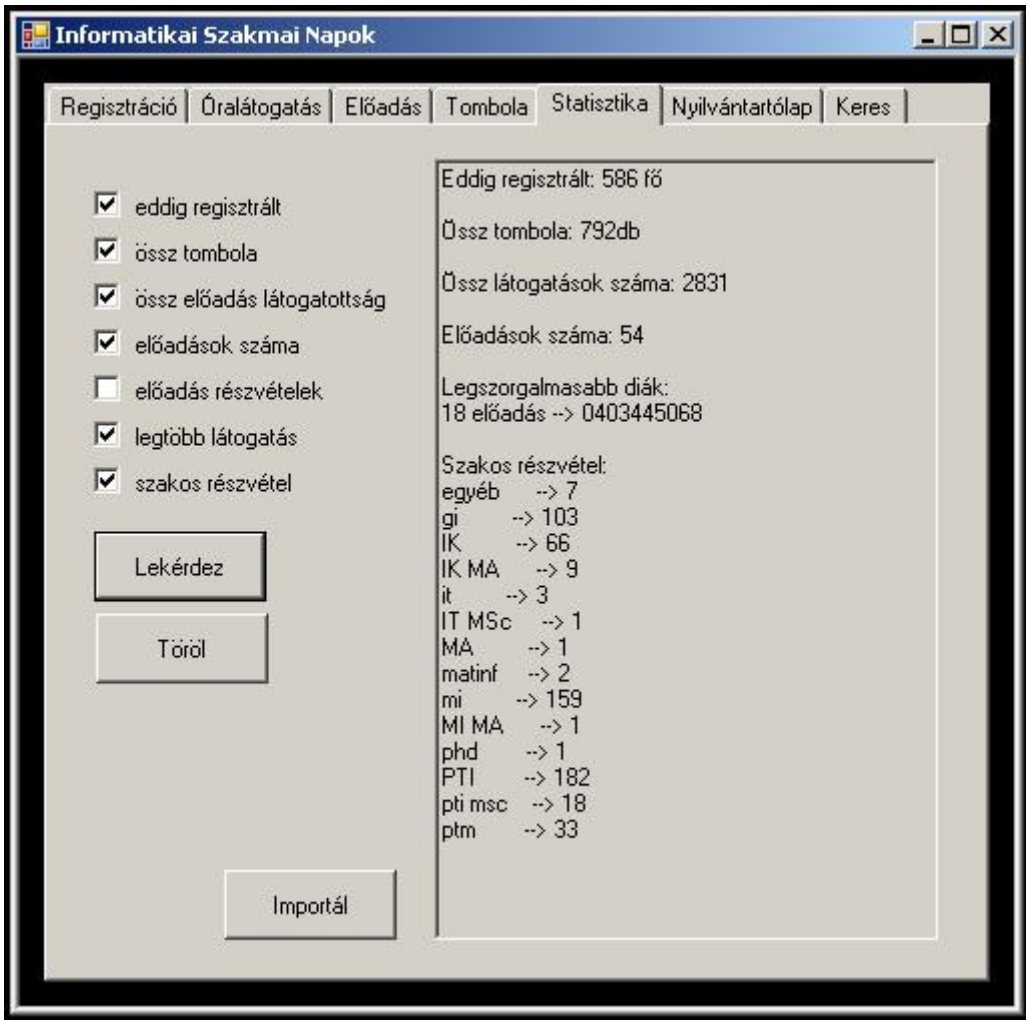


(Keres felület)

Ha beírjuk a keresendő nevet és a keres gombra kattintunk a program csatlakozik az adatbázis szerverhez, ahol a diák táblában megkeressük a nevet, majd kiírjuk a nevet, neveket, és az adatokat. Ha nincs ilyen név az adatbázisban azt kiírja a program a következő hibaüzenetben: „Nincs ilyen név regisztrálva!”. A mezőket a már megszokott „Töröl” gomb segítségével tudjuk törölni.

A „Diakig.imp” gomb szerepe a sikeres találat esetén az, hogy a diákigazolvány számot a „Nyílvántartólap” fül diákigazolvány bevitelére alkalmas mezőbe importálja, így megkönnyítve a felesleges másolást, és beillesztést.

A sorban nem a legutolsó, de legutoljára maradt a „Statisztika fül”, ami segítségével a szakmai napok alatt, de főleg utána tarthatjuk figyelemmel a különböző eredményeket.



(Statisztika felület)

Lekérdezhetjük mennyien regisztráltak eddig, mennyi tombola szelvény gyűlt eddig össze, az össze előadás látogatottságot, az előadások számát, az előadások részvételét, megtudhatjuk ki volt a legszorgalmasabb diák és, hogy szakonként mennyi diák regisztrált be.

Egyszerre akár az összes lekérdezést végrehajthatjuk, de akár egyenként is lekérdezhetjük az adatokat, erre szolgálnak a CheckBox-ok. A lekérdezett adatokat a képernyő jobb oldalán található szöveges dobozba kapjuk meg. A szöveges doboz inaktív, azaz nem lehet a benne lévő adatokat módosítani. Viszont az „Importál" gomb segítségével a szöveges mező tartalmát kiimportálhatjuk egy Stat.doc fájlba.

A lekérdezések menete a következő. Bepipáljuk a lekérdezni kívánt adatok melletti négyzeteket, majd a „Lekérdez" gombra kattintva csatlakozik az adatbázis szerverhez.. A csatlakozás után a megfelelő SQL utasítások hajtódnak végre.

Íme néhány közülük:

Az előadás látogatottságot lekérdező SQL utasítás, ami előadásonként kiírja mennyien vettek részt rajta.

```
SELECT (Count(ELL.diakig)) AS \"COkod\", Eloadas.cim FROM Ell
INNER JOIN Eloadas ON Ell.eloadaskod = Eloadas.eloadaskod
GROUP BY cím;
```

A legszorgalmasabb diák lekérdező SQL utasítás:

```
SELECT TOP 1 Diak.diakig, Count(Ell.diakig) AS \"E\" FROM Diak
INNER JOIN Ell ON Diak.diakig = Ell.diakig GROUP BY
Diak.diakig ORDER BY Count(Ell.diakig) DESC;
```

Az utasítások eredményei kiíródnak, majd a kapcsolat lezárul.

A lekérdezett adatokat a „Töröl" gomb segítségével törölhetjük a mezőből.

A fenti képen látott statisztikák a Szakmai Napok után születtek. Összesen 586 beregisztrált fő látogatott el az előadásokra. Több mint 2800 előadás látogatás történt a 3 nap alatt. Maximálisan 18 előadáson lehetett részt venni és van olyan diák, aki ezeket sorra meg is látogatta. A szakmai napokon a programtervező informatikus diákok voltak a legtöbben, szám szerint 182-en.

Ez és ezek azok a feladatok, amiket az adminisztrátor felülettel eltudunk végezni.

3.2.2. Vonalkód technika, vonalkód olvasó

A helyszínen történő beregisztrálásnál, az előadás látogatások nyilvántartásánál, és a jelenléti lapok kikérésénél a diákigazolvány szükséges, amin egy tíz számjegyű azonosító található, alatta egy vonalkóddal. A vonalkód típusa interleaved 2 of 5, ami a felette lévő tíz számjegyű diákigazolvány számot takarja.

Hogy elkerüljük minden egyes diákigazolványon található vonalkód leolvasását, egy vonalkód olvasóval megtehetjük, hogy beolvassuk azt.

A Szakmai Napok alatt több mint 3500 alkalommal használtuk a vonalkód olvasót, ami billentyűzetről történő bevitel esetében jóval megnehezítette volna a feladatot.

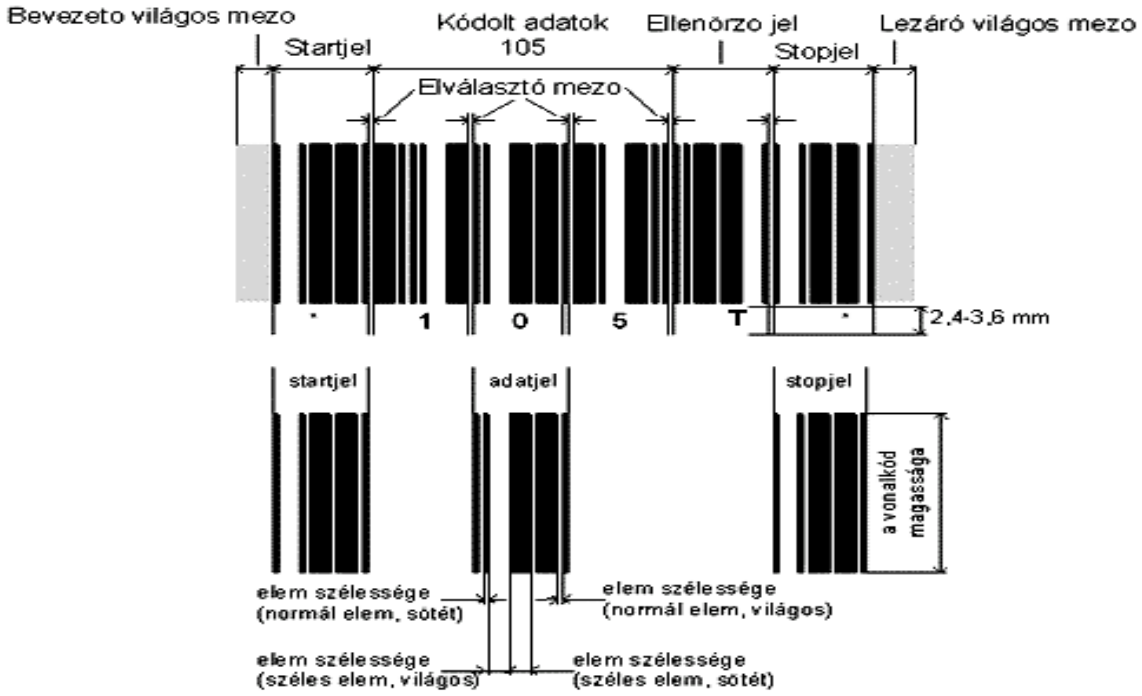
A vonalkód technika nem a napjainkban jelent meg, már 1940-ben voltak próbálkozások egy egységes kódrendszer kialakítására, ahol a vastag és vékony vonalak, illetve ezek távolsága hordozza az információt.

Manapság a legelterjedtebb kódok :

- Code 39, Code 128, Code 93
- Interleaved 2 of 5
- Codabar
- EAN/UPC

AZ EAN/UPC, a Codebar és az Interleaved 2 of 5 csak számokat kódol. A többi kód betűk kódolására is alkalmas.

A vonalkód meghatározott szabályok szerint felépülő, világos és sötét mezők váltakozásán alapuló optikailag érzékelhető kód. Típus szerint lehet: egydimenziós és kétdimenziós.



(Vonalkód felépítése)

Minden vonalkód rendelkezik egy különleges Start és egy Stop jellel. Így tudja az olvasó felismerni, ha előre vagy visszafelé olvasta a vonalsorozatot. Továbbá, egyes vonalkódoknak checkszum jele is van közvetlen a Stop jel előtt.

A vonalkód olvasók soros, PS2-es vagy USB csatlakozóval rendelkeznek. Működési elvük szerint lehetnek CCD vagy laser alapú vonalkód olvasók. A laser alapú készülék ára jóval magasabb, mint a CCD alapúaké. Cserébe strapabíróbbak és nagy távolságból, akár 30-40 cm is nagyon jól és pontosan olvasnak. A CCD alapúak fő előnye az olcsóság, bár kevésbé strapabíróak, mégsem kell többévi használat után sem aggódni.

Választásom egy CCD vonalkód olvasóra esetett, aminek a márkája és típusa Tysso CCD-800-USB-M. Az eszköz a következő, gyár által megadott paraméterekkel rendelkezik:

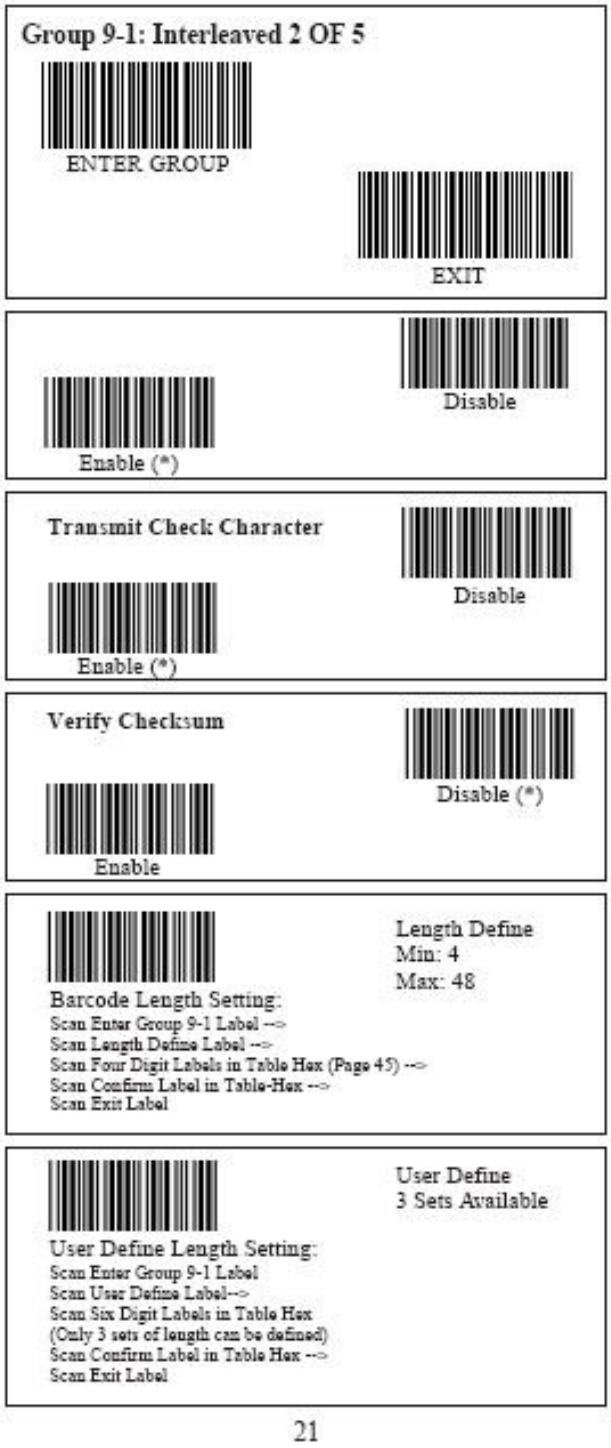
- *USB csatlakozó*
- *82mm maximális beolvasási szélesség*

- *Beolvasási távolság: 0-15mm*
- *ütésállóság: 1m magasságból szilárd talajra*
- *Vonalkód típusok: Code39, FULL ASCII Code39, UPC/EAN/JAN, CODABAR,*
- *Interleaved 2/5, Code128, Code11, Code93, MSI/PLESSEY, Code4*
- *130g tömeg, 2m hosszú kábel*
- *Olvasó fej: 2048 pixel CCD, kontraszt érzékenység: min 45%*

Vásárlás közben a fő szempontok a megadott vonalkód probléma nélküli olvasása, és az olcsóság. Az olvasó mindössze 10400 forintba került, és a kellő beállítások után tökéletesen olvasta a diákigazolványon lévő vonalkódot. Alapértelmezettként az olvasó nem olvassa a diákigazolványon lévő kódot, ez azért is van mert a nincs beállítva a Interleaved 2 of 5 vonalkód. A beállítása roppant egyszerű. Az olvasó dobozában található egy kézikönyv (Programing manual) amiben az olvasóra vonatkozó összes beállítási szempont megtalálható. Ezek a beállítások vezérlő vonalkódok segítségével történik, azaz a vonalkódokat olvassuk le a füzetből és így végrehajtva a szükséges beállításokat. Interface beállítások, a nyelvi beállítások, az olvasás utáni karakter hozzáadásának beállítása, szkennelési mód beállítása, ezek csak néhány beállítási lehetőségek, amiket a vonalkód típus beállítások előtt vagy után végrehajthatunk. A beállítások a következő képen történnek. Vesszük a kézikönyvet, ahol a megfelelő oldalt kinyitva megkeressük az általunk kívánt beállításokat. Beolvassuk az „ENTER GROUP” vonalkódot, amit a vonalkód olvasó egy rövid csipogással jelez, majd ezután a megfelelő paramétereket leolvassuk. Ha ezek megtörténtek az „EXIT” vonalkódot olvassuk le, amikor is a vonalkód olvasó egy hosszabb csipogással jelzi, hogy kilépett a beállításokból, és mentette a beállításokat. Ezt a műveletet más beállítási csoportokban is végrehajthatjuk. A megvétel után csupán csak néhány beállítást változtattam meg az olvasón, mint például a szkennelési beállításoknál, folyamatosra állítottam a scannelést, így az olvasón található gombot kiiktatva folyamatosan lehetőség van a vonalkódok olvasására. Ez természetesen azt is jelenti, hogy az olvasó folyamatosan aktív, és folyamatosan égnek a ledek. A másik ilyen beállítás a beolvasott vonalkód utáni karakter használata. Ez a karakter az Enter billentyű, de lehetett volna akár a Tabulátor, szóköz, vagy egyéb más billentyűzet. Minden beolvasott vonalkód után egy enter billentyűt is kapunk a vonalkód olvasó bufferba,

így ha a programot hozzáhangoljuk a vonalkód olvasóhoz akár gyorsíthatunk is a bevitelen azzal hogy nem kell minden egyes olvasás után a megfelelő gombra kattintani.

A legfontosabb beállítás a diákigazolványon található vonalkód típusának beállítása volt, ami a következő módon történik. A Programing Manual 21 oldalán található „ENTER GROUP” vonalkódot leolvassuk, amivel beléptünk az Interleaved 2 of 5 kód beállításának csoportjába. Ahogy a képen látszik itt először is engedélyezzük a kódot, majd engedélyezzük a check karaktert, aztán „EXIT” vonalkód leolvasásával kilépünk a beállításból amit egy időben ment is a vonalkód olvasó. Ezek után képesek vagyunk leolvasni a diákigazolványon található vonalkódot. A beállítások az eszköz eltávolítása után is megmaradnak. A „SET ALL DEFAULT” opció leolvasásával visszaállíthatjuk a vonalkód olvasót az eredet állapotába, a gyári beállításokkal.



3.3. Webes felület

A webes felület egy ASP .NET nyelven írt felület, melynek szerepe az információ megosztása, és az elő regisztrálás. Szerepe főleg az Informatikai Szakmai napok előtt és után van, előtte ugyanis regisztrálhatnak a diákok, ezzel elkerülve a helyszínen való regisztrációval

járó várakozást. A Szakmai Napok után a regisztrált diákok a diákigazolvány számukkal lekérdezhetik, és ellenőrizhetik mennyi előadáson is voltak, és mennyi tombolát szereztek, vagy éppen a rendezvény részvételi és egyéb statisztikáit nézhetik meg.

A felületet úgymond megosztása, host-olása, egy Windows Xp SP3 rendszerről egy Internet Information Service (IIS) 6.0 FTP és web host program segítségével történik. Az IIS XP egy kiegészítőjeként telepíthető. Kezelése egyszerű, a kívánt honlap elemeit a telepítés után a megadott könyvárba kell csupán másolni, majd ez el is érhető, ha az IP címünk kívülről is elérhető (publikus).

A webes felület szerkezetileg 4 elkülönülő részre osztható. Az első rész a felső vízszintes rész. A második rész a bal oldali menüsort tartalmazza, gombokkal. A jobb oldali rész tartalmazza a rendezvényen résztvevő cégek logóit tartalmazza. A középső rész tartalmazza a lényegi részeket, ez a főablak. A menügombok lenyomásával a középső rész tartalma változik.

A képernyő felosztását először framek segítségével osztottam fel.

```
<FRAMESET rows="150,*" BORDER=1>
  <FRAME src="felso.html" NORESIZE SCROLLING=NO >
  <FRAMESET cols="150,600,150" BORDER=1 >
        <FRAME src="menu.html" NORESIZE >
        <FRAME src="kezdolap.html" NORESIZE name=kozepe>
        <FRAME src="jobboldal.html" NORESIZE >
  </FRAMESET>
</FRAMESET>
```

A frame-k használatának vannak hátránya is. A probléma, hogy a felosztott képernyő, a felosztás száma, mint annyi letöltött honlapnak felel meg, és a hostolásra telepített IIS korlátozott számú klienst tud kezelni. Kliens alatt a honlapon böngésző személyt értem. Ahogy a honlapot felkeressük egyből 4 oldal töltődik le. A felső, két szélső és a középső rész mind-mind külön álló egységként.

Webes felület készítése során felmondtam a framek készítésével és a jóval hatékonyabb css-beli képernyő felosztást választottam, ami segítségével a betöltött képernyőnk nem négy hanem csupán egyetlen egy oldalként tölt be, ezzel az IIS is jelentősen megkímélve.

A CSS egy stílusleíró nyelv. Már a 90-es évek elején megjelentek a stílus alapok. Magát a CSS 1994-ben fejlesztették ki, és a mai napik eddig ez tűnik a leghatékonyabb stílusleíró nyelvnek. A CSS használata nagy körben elterjed, ezzel tudjuk egységesen átállítani a weblapok színét, betűk méretét, elrendezését, és más a megjelenéssel kapcsolatos beállításokat. Egyszerű szintaxissal rendelkezik, csupán néhány angol kulcsszót használunk fel a kívánt formázás eléréséhez.

A Visual studio-ban is könnyedén adhatunk ASP .NET webes alkalmazásunkhoz CSS dokumentumot. Jobb klikk, Add/New Item/ a felugró ablakban kiválasztjuk a Style Sheet ikont, az ablak alján megadjuk a nevét majd Add gomb segítségével a projektünkhöz is adtuk.

Rövid példa a CSS használatáról:

A CSS tartalma:

```
.style3
{
      font-size:medium;
      font-family:Comic Sans MS;
      color:Red;
}


body
{
      font-size: large;
      background-attachment: fixed;
      background-repeat: no-repeat;
      background-position:center;
```

```
    background-color:Black;
    background-image:none;
}
```

A html környezetbe a css-t a következő sor segítségével szúrjuk be:

```
<LINK REL=StyleSheet HREF="style.css" TYPE="text/css" MEDIA=screen>
```

A beszúrt tartalomnak a <HEAD> </HEAD> részben kell szerepelnie.

Használata pedig nagyon egyszerű, csupán hivatkzoni kell a CSS-ben található egységre:

```
<p class="style3> Ez lesz itt formázva </p>
```

A Webes felület csupán 5 lapból áll, csak a legfontosabb információkat tartalmazza, és csakis a funkcionalitást helyeztem előtérbe. A dizájn egyszerű, fekete és fehér a domináló színek. A fontosabb dolgokat, linkeket pirossal emeltem ki. Minden lapot a CSS segítségével egységesre terveztem, mint színben, mint betűtípusban és méretben is.

A Visual Studioban egy projekt létrehozásánál egy Default.aspx kiterjesztésű fájlt kapunk, ami teljesen tökéletes lenne a kezdőoldalnak. Kezdőlapok különböző fájlok lehetnek, a legelterjedtebbek az Index.html, php, fájlok. A kezdetben a framek miatt egy Index.html fájlt is létrehoztam a projektben, ami az előző oldalon található képernyő felosztására alkalmas kódokat tartalmazta csak, és a megfelelő hivatkozásokat a megfelelő oldalakra.

Miután beírjuk a böngészőbe az adott linket a következő dolog történik. Az IIS-t úgy állítottam be, hogy az Index.html fájl legyen a kezdőoldal, így először az töltődik be, ami a betöltéskor felosztja a képernyőt, és a megfelelő helyekre a hivatkozások alapján a megfelelő oldalakat tölti be. Ahogy az előző oldalon a kódrészletben is látható felső vízszintes részbe került a felso.html. A felső rész is egy külön megtervezett HTML oldal, amit akár magában teljes méretben is megtekinthetnénk. Az oldal két képet, a főépületet, és az Informatika kar logóját, és a „Debreceni Egyetem Informatika kar III. Szakmai Napok” szöveget tartalmazza. Ha tovább megyünk a bal oldali részbe a menü.html töltődik be. Az oldal csupán csak gombokból áll, amiket egy WebButtonShop program segítségével hoztam létre. A gombok egyenként egy-egy hivatkozást tartalmaznak. Vegyük példaként a statisztika gombot.

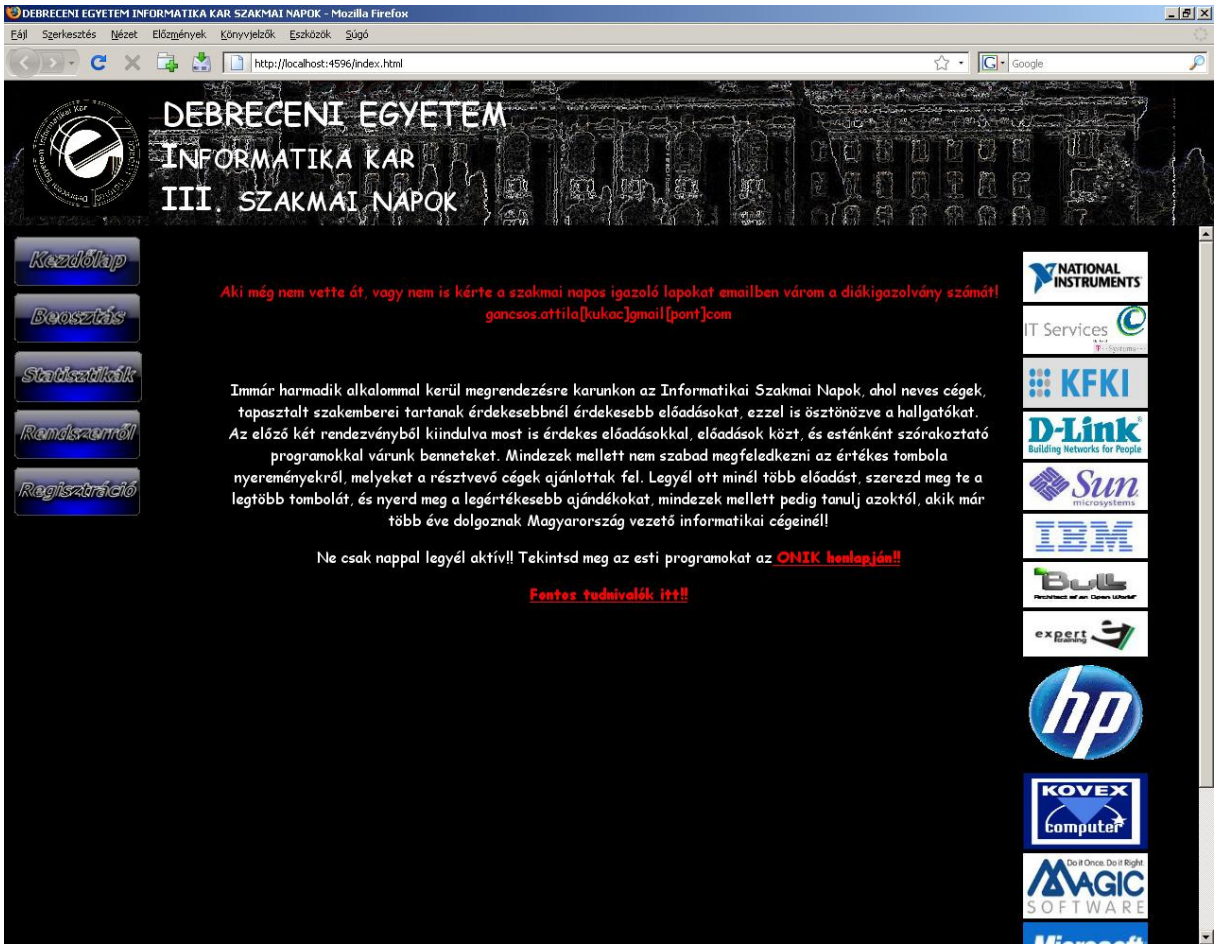
```
<a href=lekerdezes.aspx target=kozepe><img src=
Statisztikak.jpg style="border-style: none"/> </a>
```

A target = közepe adja meg, hogy a hivatkozott oldal hol jelenítődjön meg. A közepe elnevezést a képernyő közepén található nagyobb rész kapta.

```
<FRAME src="kezdolap.html" NORESIZE name=kozepe>
```

A ball oldali menüsör után a középső részbe betöltődik a kezdolap.html fájl. A középső részt nemsokára részletesebben is tárgyaljuk.

Legutoljára a jobb oldali rész, jobboldal.html töltődik be, ahol a cégek logói találhatóak. Ez, és a többi frame úgy van beállítva hogy scrollozható legyen, azaz ha a logók nem férnek egy képernyőbe bele oldalt megjelenik a csúszka.



A lényegi részek a „közepe” névvel ellátott frame-be jelennek meg. Itt a következő hivatkozások töltődhetnek be: kezdolap.html, menetrend.html, lekerdezes.aspx, about.html, Default.aspx. A fenti képen a kezdolap.html látható. Ez az alapért elemezett, minden

betöltéskor ez jelenik meg. Szerepe figyelem felkeltés, köszöntés, fontosabb dolgok megosztása. A következő szöveg olvasható a kezdő oldalon:

Immár harmadik alkalommal kerül megrendezésre karunkon az Informatikai Szakmai Napok, ahol neves cégek, tapasztalt szakemberei tartanak érdekesebbnél érdekesebb előadásokat, ezzel is ösztönözve a hallgatókat.
Az előző két rendezvényből kiindulva most is érdekes előadásokkal, előadások közt, és esténként szórakoztató programokkal várunk benneteket. Mindezek mellett nem szabad megfeledkezni az értékes tombola nyereményekről, melyeket a résztvevő cégek ajánlottak fel.
Legyél ott minél több előadást, szerezd meg te a legtöbb tombolát, és nyerd meg a legértékesebb ajándékokat, mindezek mellett pedig tanulj azoktól, akik már több éve dolgoznak Magyarország vezető informatikai cégeinél!

Ne csak nappal legyél aktív!! Tekintsd meg az esti programokat az ***ONIK honlapján!!***

Fontos tudnivalók itt!!

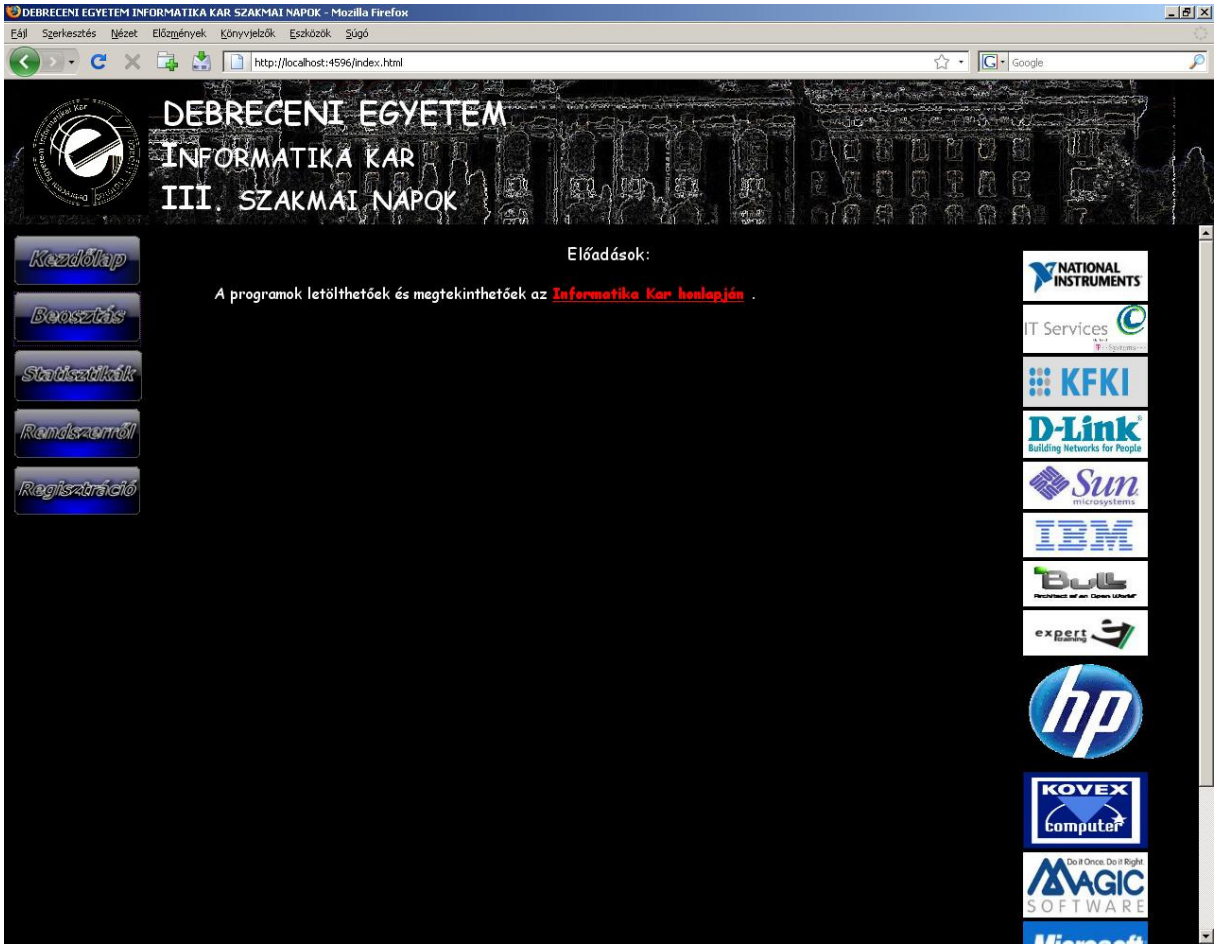
A pirossal, és aláhúzással jelölt szavak hivatkozások. Az *ONIK honlapján!!* A *http://onik.inf.unideb.hu/* címre hivatkozik. A honlap itt nem a középső részben jelenik meg, hanem egy új lapként a böngészőnkben. Ezt a target=_blank paracs segítségével érhetjük el. A *Fontos tudnivalók itt!!* az about.html oldalra mutat, és a közkéső részben jelenik meg a hivatkozás.

A kezdő oldal után a bal oldalon található gombok segítségével tovább léphetünk az oldalon.

A „Beosztások” gomb segítségével a lenti képen látható oldalt tölt be számunkra. Az oldal csupán egy sort tartalmaz, benne egy linkkel.

A programok letölthetőek és megtekinthetőek az ***Informatika Kar honlapján*** **.**

A link az Informatika Kar honlapjára mutat (www.inf.unideb.hu), ahol a Szakmai napok programja letölthető egy .xls kiterjesztésű Excel táblázatként.



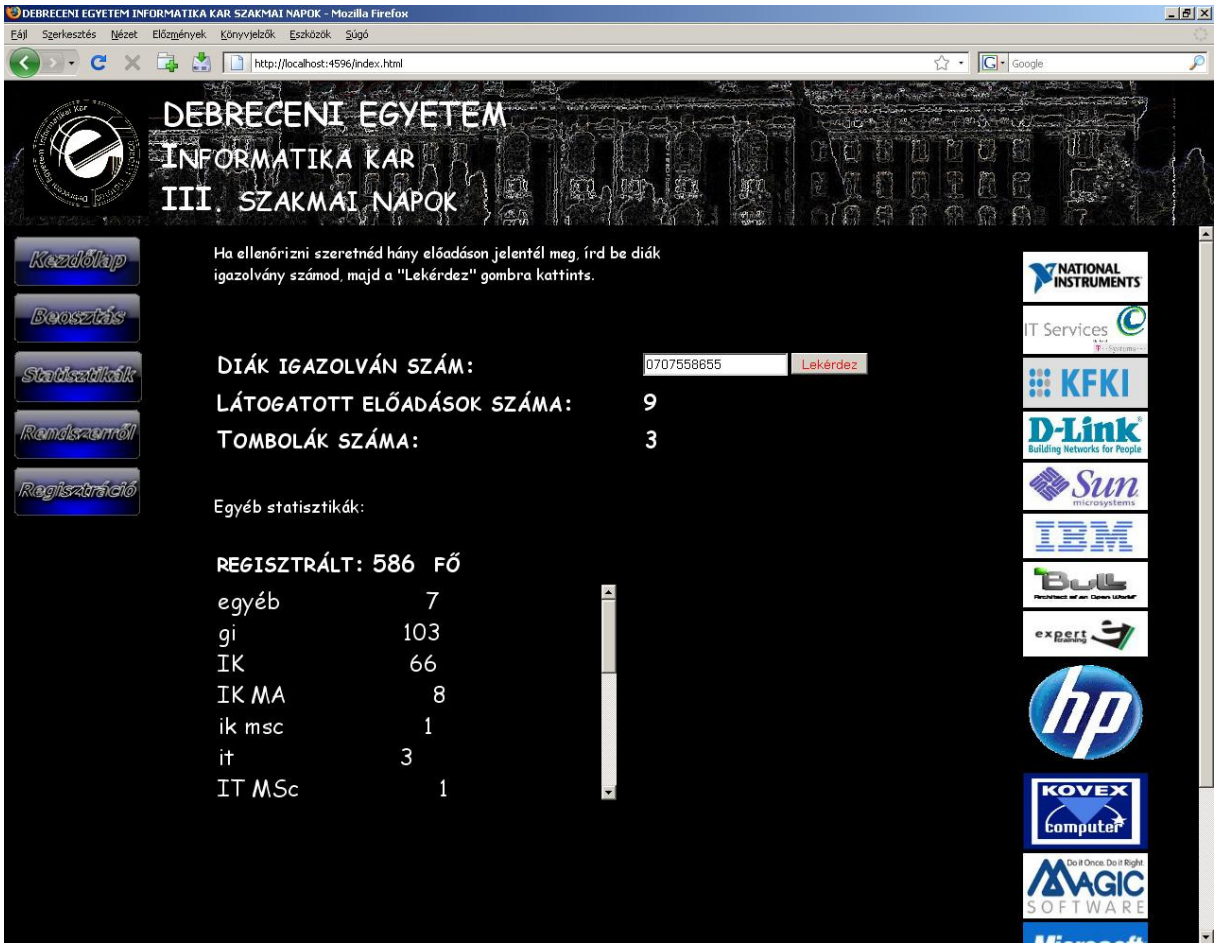
(Szakmai napok programjának linkje)

A következő gomb a „Statisztika” gomb, ami lenyomásával a következő, lentebb látható oldal töltődik be. A honlap talán egyik legfontosabb oldala ez, hiszen itt diákok ellenőrizhetik mennyi előadáson is voltak, és hány tombolát szereztek. Mindemellett látható mennyien regisztráltak eddig, és a szakos részvételt is megtekinthetjük. Mivel több adatbázissal dolgoztunk ezért fontos volt az adatbázisok közötti szinkronizáció, hogy a honlapon lévő adatok, és a lekérdezett adatok is naprakészek legyenek. Ez a 3.1.3 fejezett alatt olvasható adatbázis szinkronizációs program segítségével történt.

A statisztika.aspx kiterjesztésében is eltér az eddigi lapoktól, ugyanis eddig csupán html kiterjesztésű lapokra hivatkoztunk. Az aspx kiterjesztést az ASP .Net alatt megírt honlapok kapják kiterjesztésül. Itt az alap HTML kódokon kívül ASP .Net kódok is előfordulhatnak <ASP></ASP> határoló kódok között. A kódok jellegzetessége a `runat="server"`, azaz hogy forráskód nem töltődik le a kliens gépére, hanem a szerveren fut. Csupán csak a html kódok töltődnek le. Minden egyes ASP .NET-ben megírt honlapnak vagy egy mögöttes kódja,

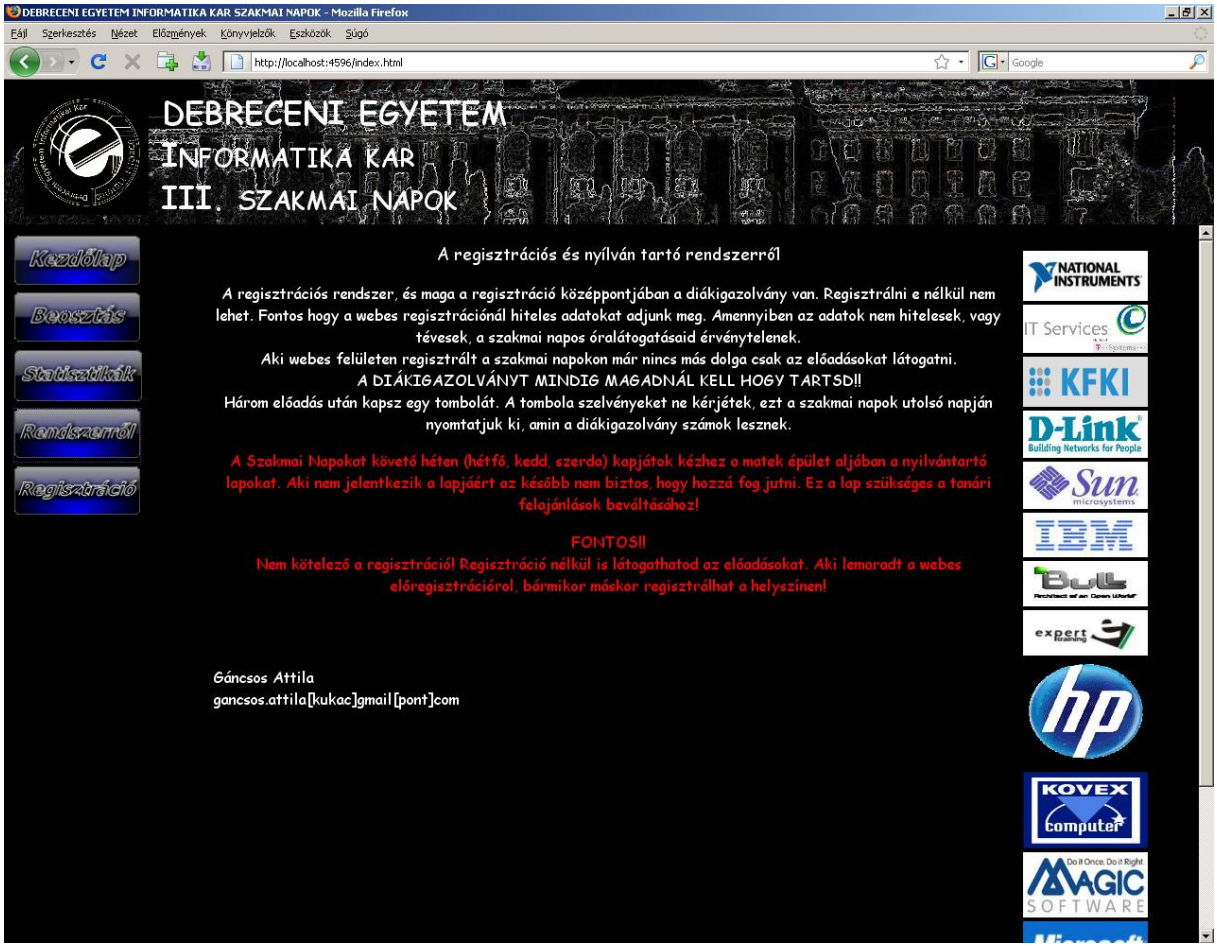
Code Behind. Ez az a kód ami nem töltődik le, ami mindig a szerveren fut. Nyelve alapján C#, kiterjesztése is lekerdezes.aspx.cs, mint az alap C# nyelven írt programoknak. A háttérben ugyan olyan objektum orientált programozás folyik, mintha egy konzolos programot hoznánk létre. Ez és ezek azok a fájlok amiket egy php, html hosting-ra felkonfigurált szerver nem képes futtatni. A php-ban megírt honlapoknál ilyen és ehez hasonló Code Behind fájlok nem találhatóak. Ott a php kódok a html kódok között találhatóak, a fájl kiterjesztése is php lesz.

Ahogy már említettem a code behind c# nyelven íródott, így lényegi eltérés nincs az adminisztrátor felület ugyanilyen statisztikát lekérdező kódjai között, csupán néhány kisebb szintaktikai eltérés van csak. Például TextBox, textBox.



(Statisztikák, jelenlét lekérdezése)

A „Statisztikák” gomb alatt található a „Rendszerről” gomb. A gombra kattintva az about.html oldal töltődik be, ahogy az a lenti képen is látszik. Az oldalon a regisztrációs rendszerről és a folyamatokról olvashatunk bővebben.



(A rendszerről)

A regisztrációs és nyílván tartó rendszerről

A regisztrációs rendszer, és maga a regisztráció középpontjában a diákigazolvány van. Regisztrálni e nélkül nem lehet. Fontos hogy a webes regisztrációnál hiteles adatokat adjunk meg. Amennyiben az adatok nem hitelesek, vagy tévesek, a szakmai napos óralátogatásaid érvénytelenek.

Aki webes felületen regisztrált a szakmai napokon már nincs más dolga csak az előadásokat látogatni.

A DIÁKIGAZOLVÁNYT MINDIG MAGADNÁL KELL HOGY TARTSD!!

Három előadás után kapsz egy tombolát. A tombola szelvényeket ne kérjétek, ezt a szakmai napok utolsó napján nyomtatjuk ki, amin a diákigazolvány számok lesznek.

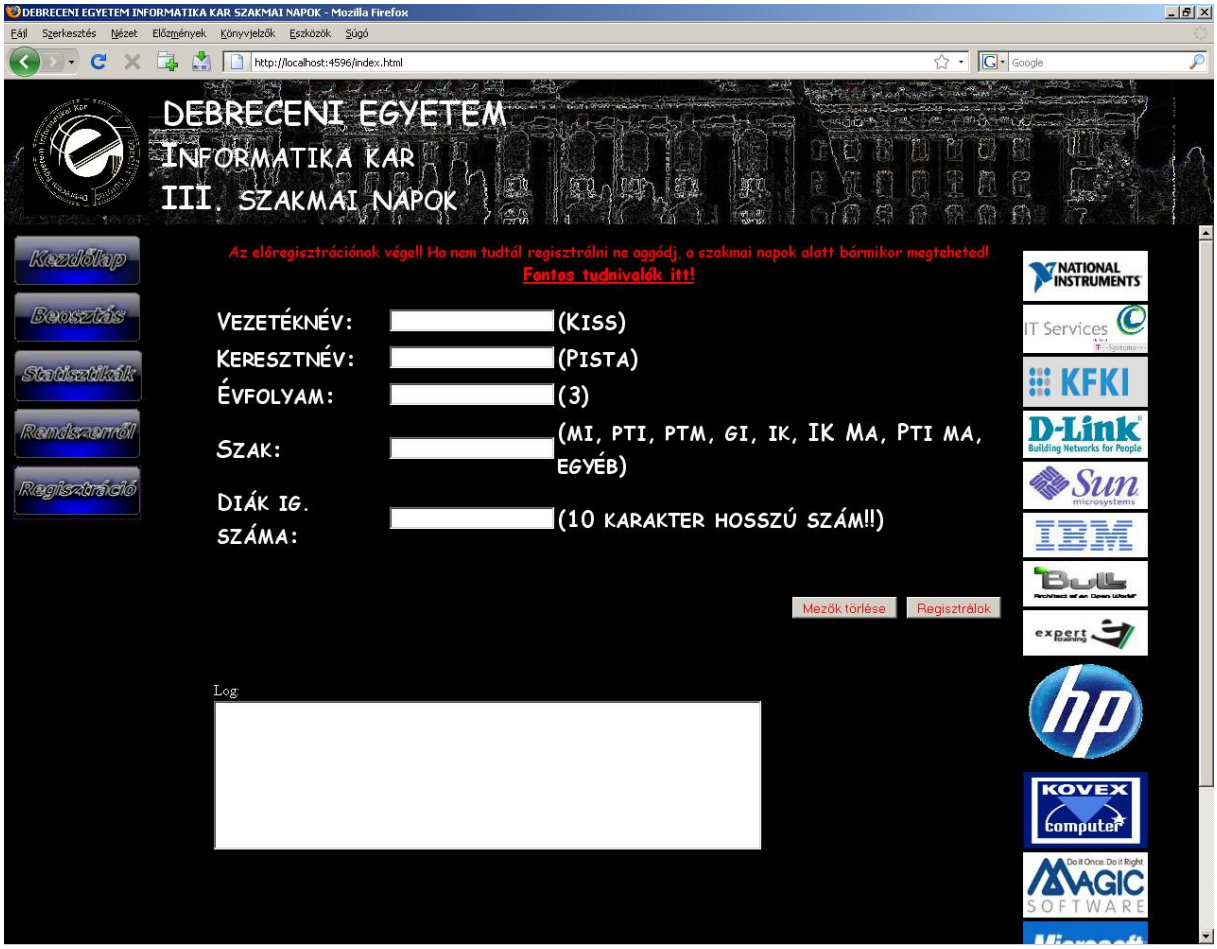
A Szakmai Napokat követő héten (hétfő, kedd, szerda) kapjátok kézhez a matek épület aljában a nyilvántartó lapokat. Aki nem jelentkezik a lapjáért az később nem biztos, hogy hozzá fog jutni. Ez a lap szükséges a tanári felajánlások beváltásához!

FONTOS!!

Nem kötelező a regisztráció! Regisztráció nélkül is látogathatod az előadásokat. Aki lemaradt a webes előregisztrációról, bármikor máskor regisztrálhat a helyszínen!

A következő „Regisztráció” gomb a honalap legfontosabb oldala. Lényegében ez az amiért elkészült a webes felület, hogy a szakmai napok előtt regisztrálhatni lehessen előre, ezzel elkerülve a sorbaállás kellemetlenségeit. Az alapötlet az onnan származott, hogy már az előző szakmai napok alatt is felvetődött miért ne regisztrálhatnának előre a diákok, ezzel megkönnyítve a diákok és a szervezők munkáját. A webes regisztráció olyannyira sikeres volt, hogy a látogatók majd fele, 251 fő itt regisztrált előre.

A regisztráció nagyon egyszerű, értelem szerűen kell kitölteni a mezőket, de segítségként példával illusztráltam, hova milyen adat kerül. A regisztrálás ugyanúgy történik, mint az adminisztrátor felületen, csupán azt leszámítva, hogy itt a diákok maguk regisztrálnak, míg ott egy szervező regisztrálta be őket.



(Regisztráció)

A mezőknek ugyanolyan kritériumoknak kell megfelelnie, mint az Adminisztrátor felületnél. Ezek a kritériumok a következők: egyik mező sem maradhat üresen, az évfolyam mezőben szám típusúnak kell lennie az adatnak, a diákigazolvány mezőben szintén szám típusúnak kell lennie, 10 karakter hosszúnak, és egyedinek. Ha ezen kritériumok csak egyikének nem felel meg máris sikertelen a regisztráció, és hibaüzenetet kapunk.

A „Regisztrálok" gombra kattintva a következő műveletek zajlanak le a host szerveren. A code behind megkapja a kliens gépen beírt adatokat, majd ellenőrzi azokat, hogy megfelelnek-e a kritériumoknak, majd csatlakozik a szintén szerveren található adatbázis szerverhez. Ez a szerver nem ugyanaz a szerver, amit az Adminisztrátor program használ, viszont a benne lévő adatbázis minden egyes paraméterével megegyezik a másik szerveren található adatbázissal. Ha csatlakoztunk az adatbázishoz ellenőrizzük, hogy a diákigazolvány megtalálható-e már a Diák táblában. Ha megtalálható „Már regisztrált!" hibaüzenetet kapunk, majd az adatbázissal való kapcsolat lezárul. Ha még nem szerepel az adatbázisban a diákigazolvány szám, akkor a következős lépésben bevisszük azt az adatbázisba a többi mezővel együtt, majd zárjuk a kapcsolatot. Sikeres regisztráció esetén „Sikeres regisztráció!" üzenetet kapunk. A töröl gomb segítségével az oldal mezőiben található adatokat törölhetjük.

Az elő regisztráció a rendezvény előtti nap 22:00-ig tartott. Ezután szinkronizáltam a két adatbázist.

4. Észrevételek és javítási lehetőségek, továbbfejlesztések a rendszerrel kapcsolatban

Természetesen akadtak dolgok, amiken még lehet javítani, vagy esetleg módosítani a rendszeren. Ilyenek például a teljes papírozás megszüntetése, azaz a diákok ne papíron nyomtatva kapják meg a felajánlásokat. A webes felület kibővíthető egy olyan oldallal, ahova csak a tanárok léphetnek be megfelelő felhasználói név és jelszó ismerete alapján. A Diák tábla még egy mezővel bővülne, aminek neve „latog" lenne. A mező értéke alapértelmezettként 0 lenne, és szám típusú. Itt tárolnánk hogy egy diák hány előadáson volt. A mező értéke csak a rendezvény után változna meg. Az Adminisztrátor programot kibővítenénk egy újabb felülettel, ami minden egyes diákigazolványt végignézve a Ell tábla alapján kiszámolná hogy a diák hány előadáson is volt jelen, majd a 0 értéket tartalmazó rekordokat a „latog" mezőben felülírjuk a megfelelő értékkel.

Miután az oktató belépett a védett oldalra az adott mezőbe beírhatja a diák diákigazolvány számát, ami alapján pontosan tud rákeresni az adott diákra. Ahogy rákeresett megjelenik mennyi előadáson is volt és mik voltak azok az előadások. Ezen a felületen lehetőség van az oktató által felhasznált előadások számának megadására. Amint ezt megadta az oktató a Diák táblában a „latog" mező értéke a megadott szám értékével csökken az adott diáknál. Ezzel a módszerrel teljesen papírmentessé tehetjük a rendezvényt. Hátránya csupán hogy plusz adminisztrációval jár az oktatók részéről.

Másik továbbfejlesztési lehetőség a gyorsaság javítása érdekében több leolvasó használata. Ez azt jelentené hogy több laptopot használunk. Ez két félek épp is megtörténhet. Egyik ilyen lehetőség lenne, ha nem lenne folyamatos hálózati elérés, hogy minden egyes laptopon ugyanarra a mintára, és ugyanazokkal a beállításokkal rendelkező adatbázisok lennének. Ezeket az adatbázisokat folyamatosan szinkronizálnánk óránként. Viszont ha van folyamatos hálózati elérés egy központi adatbázist használhatunk, amihez csatlakoznak az adatbázisok, és a webes elő regisztráció is. Így nem kell szinkronizáló programot használni. Előfordulhat, hogy egyes előadások más épületben történnek, ilyekor keverhetjük a módszereket. Azaz használhatunk egy központi adatbázist, de ha megszakad a kapcsolat, akkor a saját gépen lévő adatbázisba mentünk, majd ahogy a kapcsolat újra él a központi adatbázis szerverrel a szinkronizáció megtörténik.

További javítási lehetőség a webes felület dizájn része. Sajnos időhiány miatt a felület ezen irányú fejlesztése háttérbe szorult.

Talán az egyik legfontosabb dolog, amit a következő szakmai napon fontos lehet, az a folyamatos, biztonságos hálózati lefedettség, egy Microsoft-os adatbázis és web szerver hozzáférés. Ez lényegesen megkönnyítené az adatbázis, adatbázisok körüli bonyodalmakat.

5. Összegzés

A 2009 évi III. Informatikai Szakmai napok után elmondhatom, hogy egy minden szempontból sikeres rendezvény tudhatunk magunk mögött. A statisztikák szerint idén volt a legnagyobb a részvételi arány, pedig egyáltalán nem volt kötelező az órák látogatása. A tavalyinál kevesebb tanári felajánlások mellett is többen jöttek el és ülték végig az előadásokat, mint az eddigieknél. A cégek nem szerénykedtek a tombola ajándékok felajánlásánál. Több értékesebb nyereményt is kisorsoltunk mint például routereket, pendriveket, kabátokat, felsőkategóriás egereket.

Mindezek mellett teljesen új rendszer szerint zajlott a regisztráció és az óra látogatás nyilvántartása. Az eddigi tapasztalatok alapján megírt rendszer teljes mértékben helytállt, és hiba nélkül végezte feladatát. Az előző pontban leírt javításokkal még tökéletesebbé és gyorsabbá tehető a rendszer működése.

6. Irodalomjegyzék

[1] Dr. Kovács Emőd, Hernyák Zoltán, Radványi Tibor, Király Roland: A C# programozási nyelv a felsőoktatásban programozás tankönyv (*http://aries.ektf.hu/csharptk/*)

[2] Jason Price: C# adatbázis programozás mesteri szinten

[3] Shepherd George: Microsoft ASP.NET 2.0 lépésről lépésre