Fejlesztői leírás a TávTagTár programhoz

(a program működése és továbbfejlesztési lehetőségek)

Készítette:

Nyíri Gábor, **hdd@nc-studio.com**

GDF Abakusz regisztrációs kód: GDFAba43

Tartalomjegyzék

Feladatdefiníció

Készítsen egy alkalmazást, mely segítségével egy ember a világ bármely pontjáról megtekintheti és karbantarthatja személyes és céges partnerei adatait. A partnereket különböző csoportokhoz lehet rendelni (pl. Boltok, Rokonok, Testvérek), egyszerre akár többhöz is. A csoportok egymásba ágyazhatók: egy csoportnak akárhány tagja lehet, ahol a tag lehet akár partner, akár újabb csoport, és egy csoport akárhány másik csoportba betehető (lásd Példa csoportszervezésre). A csoportok tagjait (partnereit és alcsoportjait) ki lehet listázni képernyőre, illetve nyomtatóra. A tagok és csoportok között keresni lehet; és levelet lehet küldeni a tetszőlegesen összeválogatott csoportoknak, partnereknek.

Szeretnénk tudni a tagok belépésének és törlésének (tagság megszűnésének) a dátumát, és szükség esetén ki szeretnénk listázni a régi tagokat is. A program adjon értesítést a fontosabb névnapokról és születésnapokról!

Az egyes partnerekről a következő adatokat szeretnénk tudni:

- Minden partnerről: név, postai cím, email címek (elsődleges cím megadásával), telefonok, Internetes lapok, számlaszám, esetleg fényképek.
- Személy esetén: névnap, születésnap.
- Cég esetén: kapcsolattartó személy.

Legyen meg a lehetőség az adatok vagy az adatok egy részének szöveges (esetleg XML) állományba való exportálására, illetve az onnan való importálására.

A program legyen jelszóval védhető.

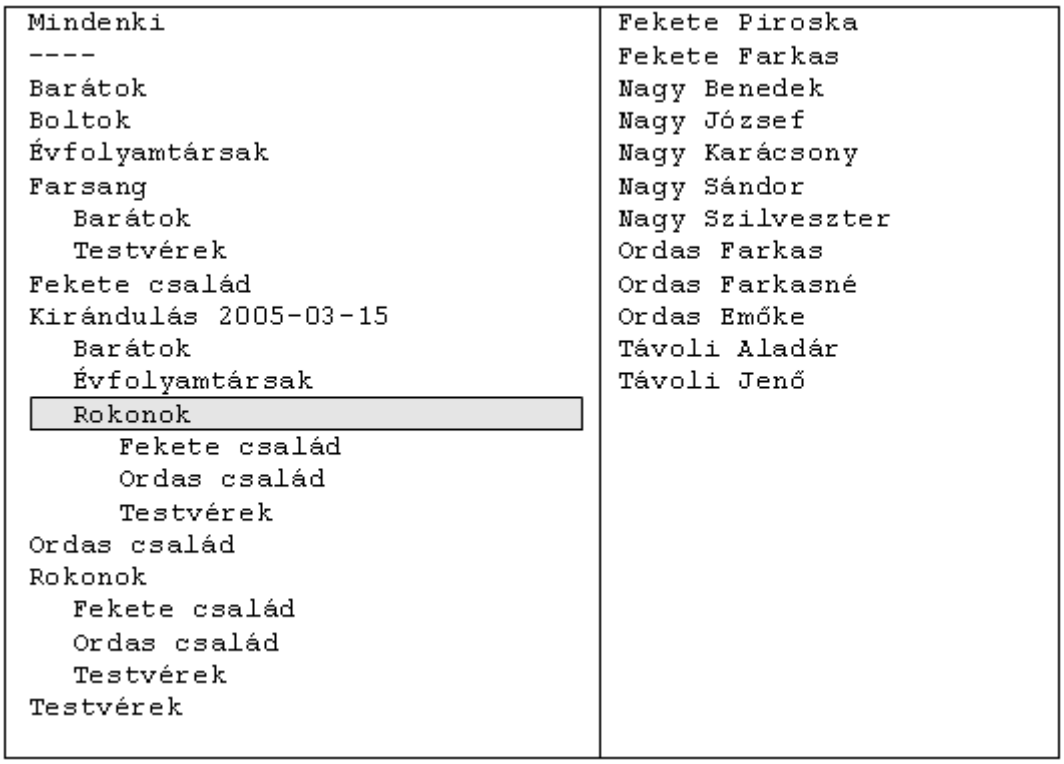
- Megjegyzések
- Az alkalmazás felületét el lehet készíteni akár ablakos formában (távoli hívásokkal), akár webes (böngészőből futtatható) formában.
- Ha szükséges, adjunk megszorításokat a csoportok egymásba tételére, hogy ne legyenek logikai ellentmondások, és a feladat megvalósítható legyen.
- Egy csoport tartalma mindig egyértelmű: ha egy csoport tartalmát bármilyen módon megváltoztatjuk, akkor az minden érintett csoportban realizálódik.
- Egy partner egy szinten csak egyszer lehet tagja a csoportnak. Az azonban elképzelhető, hogy ugyanaz a tag különböző alcsoportok tagjaként többszörösen is tagja egy csoportnak. Tegyük fel például, hogy Kun Feri már benne van a *Barátok* és az *Évfolyamtársak* csoportokban. Megtehetem, hogy a *Barátok* és az *Évfolyamtársak* csoportokat beteszem a *Kirándulás 2005-03-15* csoportba; és ezen kívül Kun Ferit

még külön is beteszem, mert őt biztosan el akarom hívni a kirándulásra, de nem emlékszem, mely csoportokban van benne. Többszörös tagság esetén csak egyszer jelenítsük meg a tagot a partnerlistán!

- Email küldésekor ügyeljünk arra, hogy többszörös tagság esetén is csak egy email-t küldjünk a partnernek!

Példa csoportszervezésre

A példában baloldalon egy fastruktúrában szerepelnek a csoportok. A legfelső szinten az összes csoport megjelenik ábécérendben, és minden csoport alatt kibonthatók a benne szereplő csoportok (ugyanaz a csoport természetesen több helyen is szerepelhet). Jobb oldalon látható a partnerlista, amely a kijelölt csoport összes partnerét tartalmazza, mindegyiket pontosan egyszer.



1. ábra

.NET[1] alapok

A .NET egy pár éve megjelent technológia, mely más eddig használt eljárásokat ötvöz, úgy, hogy az azok közti kompatibilitás továbbra is fennmaradjon, a migráció minél egyszerűbben véghezvihető legyen és a fejlesztőknek ne az infrastruktúra megteremtésével kelljen az idejüket tölteniük hanem azt készen kapják valakitől.

A .NET-nek az eddigi technológiákkal szembeni egyik nagy előnye – és részben talán hátránya is – hogy nem natív kódot fordít a compiler, hanem egy úgynevezett IL azaz köztes kódot, majd ezt a kódot fordítja tovább közvetlen a futás előtt a framework. Ennek következtében olyan alkalmazások készítésére nyílik lehetősége a fejlesztőknek, amik az eddigieknél sokkal rugalmasabbak és jelentősen kevesebb kompatibilitási problémával küszködnek majd várhatóan a jövőben.

A keretrendszer tervezői többek között figyelembe vették az egyes programbeli objektumok életciklusát is és a sok esetben előforduló „memória elfolyás” problémáját is ezért létrehozták az úgynevezett Garbage Collector-t – szemétgyűjtőt – ami egy bonyolult eljárás segítségével időnként végignézi a program által lefoglalt memóriát és törli mindazon objektumokat amikre már nincs hivatkozás, vagyis amiket a program a továbbiakban nem fog használni.

Egy talán a létező legrövidebb definíciója ezen technológiának a következő: előre gyártott infrastruktúra az internetes alkalmazásokban jelentkező problémák megoldására.

A választás így logikusan esett a .NET alapú fejlesztésre az előbb említett technikai előnyök miatt.

[1] Ejtsd: dotNET ; A témáról bővebben David S. Platt – Bemutatkozik a Microsoft .NET című könyvében lehet olvasni.

SQL háttér

Minden információt szinte kivétel nélkül adatbázisban tárol. Olyan kivételekre kell gondolni, mint például az adatbázishoz való kapcsolódás paraméterei és az egyes felhasználói beállítások – ez utóbbiak ebben a verzióban időhiány miatt nem kerültek megvalósításra.
Azért nagy előny, hogy minden adat adatbázisban tárolódik, mert ilyen formában az archiválás jelentősen leegyszerűsödik, és jóval kézben tarthatóbbá válik.
Az adatbázismotor vagy SQL 2000, vagy MSDE (az SQL 2000 rendszer felhasználói felület nélküli, ingyenes[1] változata, mely némi teljesítményoptimalizálástól és különleges funkcióktól eltekintve megegyezik komolyabb testvérével). Azért esett ezekre az eszközökre a választás, mert teljesítmény és funkcionalitás szempontjából feltehetőleg ezek voltak a legideálisabbak.
Az lenne az ideális, ha minden lekérdezés tárolt eljárásokon keresztül futna le, így egyfelől jelentősen gyorsabbak, másfelől biztonsági szempontból jóval védettebbek lennének, de ezeket egyrészt igazán a feladat sem kívánta meg – mert várhatón az adatbázis nem lesz jelentős méretű – másrészt a fejlesztést is jóval bonyolultabbá tette volna, ami jelentősen megnövelhette volna a fejlesztéshez szükséges időt.
Ha a TavTagTar-t nem csak installálni, hanem továbbfejleszteni is kívánja, mindenképpen érdemes jó (akár ingyenes) SQL kliens eszközöket beszerezni, mert a Visual Studio.NET nem minden esetben ad tökéletes megoldást az SQL lekérdezések szerkesztésére.
Ha bővítjük a rendszert, újabb lekérdezéseket/tárolásokat (insert, update) mindenképpen tárolt eljárások segítségével valósítsuk meg, a fent említett okok miatt.
Sajnos az egyes táblák közti konzisztenciát még a program tartja fenn és nem az adatbázis szerver. Erre feltételezhetően adatbázis oldalon is van megoldás, csak az WinForms oldalon nem került eddig megvalósításra.
Az adatbázis-kezelések minden esetben sqlDataAdapter-ek segítségével történnek. Jelen dokumentációban nem kerül feltüntetésre minden lekérdezés csak azok, amelyek nagyobb bonyolultságúak lehetnek, ezzel segítve a dokumentáció áttekinthetőségét és megértését.
Adatbázis létrehozását a program nem támogatja viszont ennek elősegítése érdekében készült egy BackUp egy már kész mintaadatokkal feltöltött adatbázisról **TavTagTar DB Backup.mssql2k** néven és egy SQL Script File **tavtagtar.sql** néven**.** Mindkét állomány a **Database backup** alkönyvtárban található meg.

[1] Csak bizonyos feltételek teljesülése esetén!

Az adatbázis szerkezete

Az adatbázis 5 táblából áll, melyek a következők:

- Szemtul
- Cegtul
- Csoporttagok
- Tagok
- Tipusok

Egy csoport, személy vagy cég felvételekor, törlésekor illetve adatainak módosításakor – csoport esetén csak átnevezésre van mód – a fő adatok a **tagok táblában kerülnek tárolásra**.
Ilyen adat:

- az **ID** (*key*, int, kötelező kitölteni, 4 bájt),
- a **Tipus** (int, kötelező kitölteni, 4 bájt),
- a **Nev** (varchar, kötelező kitölteni, 50 bájt),
- az **email** (varchar, nem kötelező kitölteni, 50 bájt),
- a **cim** (varchar, nem kötelező kitölteni, 100 bájt),
- a **telefon** (varchar, nem kötelező kitölteni, 50 bájt),
- a **letrehozva** (datetime, nem kötelező kitölteni, 8 bájt) és
- a **torolve** (datetime, nem kötelező kitölteni, 8 bájt) mezők.

Az **ID** mező segítségével történik a többi táblával az összekapcsolás. Minden tagok táblabeli rekord – sor – egy egyedi ID-t kap, ami egy automatikusan növekvő 4 bájtos szám.
A **tipus** mező mondja meg egy bejegyésről, hogy az cég, személy vagy csoport adatait tartalmazza. Csoport esetében ez a mező a nulla értéket veszi fel, személy esetében ez egy, míg cég esetében kettő. Ennek a mezőnek az ismeretében könnyen lekérdezhető a személyek vagy cégek adatai a megfelelő alárendelt táblából, ami a **szemtul** vagy a **cegtul** lehet.
Egy tag – legyen az személy, cég vagy csoport – felvételekor minden esetben az aktuális dátum kerül tárolásra a letrehozva és a torolve mezőkben. Így, ha ennek a két mezőnek a tartalma nem egyezik, akkor a tag törlésre került a torolve mezőben található időpontban.
Az egyéb kiegészítő adatokat a program a cegtul vagy a szemtul táblákban tárolja attól függően, hogy cégről avagy személyről van e szó.

A szemtul tábla öt mezőből áll:

Column Name	Data Type	Length	Allow Nulls
ID	int	4	
Otthoni_Telefon	varchar	50	✓
Mobil_Telefon	varchar	50	✓
szuletesidatum	datetime	8	✓
nevnap	datetime	8	✓

***2**. ábra*

Itt a rekordokat szintén az ID mező azonosítja, de ennek itt az elsődleges szerepe a sorok összerendelése a tagok tábla tartalmával.

Itt a névnap és a születésnap tárolására szolgáló két mező – nevnap, szuletesnap – még hagy némi kívánnivalót maga után a felhasználói felület terén.

A cegtul tábla négy mezőből áll:

Column Name	Data Type	Length	Allow Nulls
ID	int	4	
Kapcsolattarto_neve	varchar	50	✓
Fax	varchar	50	✓
Szamlaszam	varchar	50	✓

3. ábra

Itt a rekordokat szintén az ID mező azonosítja, de ennek itt az elsődleges szerepe a sorok összerendelése a tagok tábla tartalmával.

A csoporttagok tábla feladata a tagok és a csoportok alá illetve fölé rendelt viszonyát tárolni. Erre három mező áll a rendelkezésre melyből egy – *RID* – csupán csak technikai szerepet tölt be, azon kívül hasznos szerepe jelen verzióban még nincs.

Column Name	Data Type	Length	Allow Nulls
RID	int	4	
CSID	int	4	
ID	int	4	

4. ábra

A tényleges összerendelést a CSID és az ID mezők segítségével kerültek megvalósításra. A CSID mező tartalmazza a **Cs**oport, tagok táblabeli ID-jét, míg az ID mező szintén a tagok táblabeli azon tag ID-jét tartalmazza ami alárendelt viszonyba került a CSID által megnevezettel.

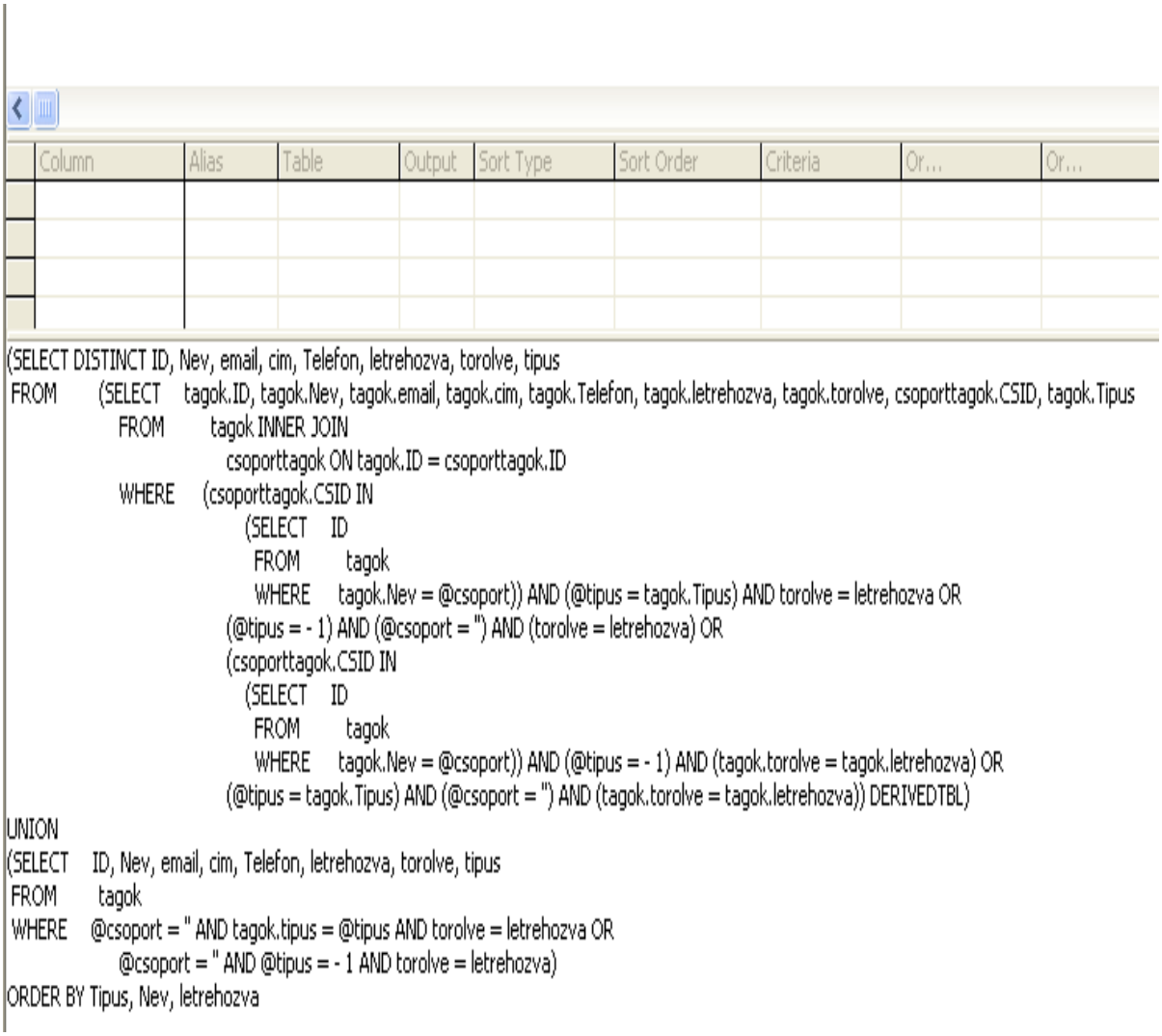
*Így például ha a **Barátok csoport** tagok táblabeli **ID-je 1**, **Marika ID-je: 2** és a **Csoporttársaké 3** akkor ha Marikát be szeretnénk tenni a Csoporttársak csoportba, majd a Csoporttársak csoportot pedig a Barátokba akkor ehhez a következőképpen kell feltöltenünk a táblát – az RID mező tartalma példánkban lényegtelen.*

RID	CSID	ID	Megjegyzés
1	3	2	Marika legyen a csoporttársak csoport tagja
2	1	3	A csoporttársak csoport legyen a Barátok tagja

Ezen összerendelés alapján Marika így mind a csoporttársak mind a barátok csoport tagja lett, habár ezt egy alcsoport segítségével értük el. Így ha - tegyük fel - e-mail-t kell küldeni a barátok csoport tagjainak azt Marika ugyanúgy megkapja, mintha a levelet közvetlen neki, vagy a csoporttársak csoport tagjainak küldenénk. Itt egy hibaforrásként jelentkezhet, hogy az alkalmazott rekurzív lekérdezés egyúttal szűrést is végezzen az ismétlések elkerülése végett.

Főablak

A főablakban megjelenő listát egy összetettebb SQ lekérdezés valósítja meg:



5. ábra

(Ezt a lekérdezést például a VisualStudio.NET 2003 Enterprise Architect már nem támogatja az unióképzés miatt, mint az a képen is látható.)

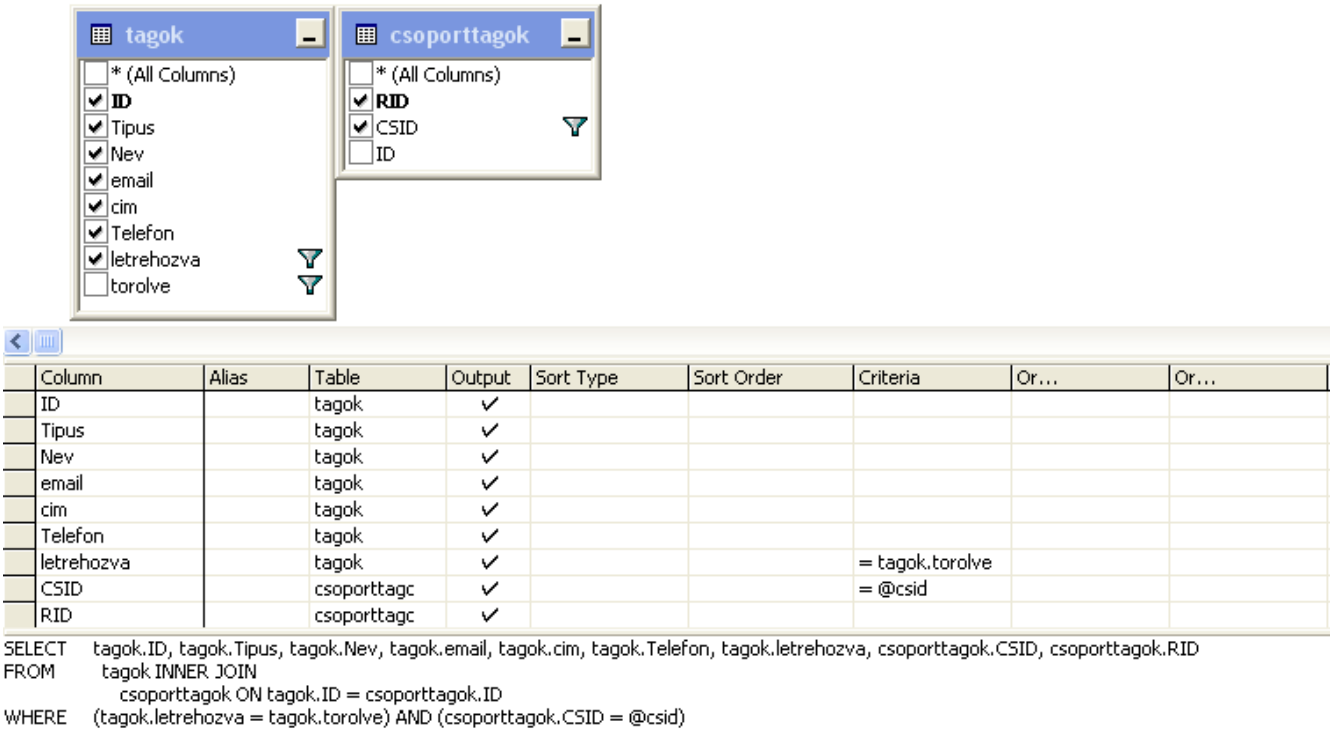
Nyomtatás

A nyomtatás a **Visual Studio.NET 2003**-ban is megtalálható **Crystal Reports** segítségével valósítottam meg.

Ennek következtében a program működéséhez szükség van a Crsystal Report Engine-re, ami a **print.rpt** sablonfájl alapján elkészíti a riportot.

A riport létrehozásához az adatokat egy típusos DataSet-ből kapja a riportkészítő. A DataSet tulajdonképpen nem más mint egy a memóriában elhelyezett adatbázis, ami jelen esetben egyetlen táblából áll, és tartalmazza egy SQL lekérdezés által visszaadott nyomtatandó – illetve az exportálandó – adatokat.

A lekérdezésben csak egyetlen paraméter található – a feladat definíció egyszerű értelmezése miatt – aminek a segítségével kiválasztható az a csoport aminek a tagjait ki szeretnénk nyomtatni.



6. ábra

E-mail küldés – körlevél

Az E-mail küldés – a körlevél funkció – az **RFC 822** szabvány segítségével lett megvalósítva. Az előbb megnevezett szabvány frissebb változata az **RFC 2822**.

A körlevélküldés megtervezésekor tervezve volt további lehetőségek megvalósítása is, mint például az elküldött levelek tárolása vagy a sablon alapján történő levélküldés, de ez eddig idő hiányában nem került megvalósításra.

Az E-mail küldő eljárást a **korlevel.cs** fájlban található **Send**(…) metódus valósítja meg:

public string Send(string SMTPServer, string From, string To, string Subject, string Message, System.Windows.Forms.ComboBox log)

Ennek lényege, hogy egy TcpClient objektumon keresztül felveszi a kapcsolatot az SMTP szerverrel, majd egy NetworkStream-en és egy StreamReader-en keresztül kommunikál a kiszolgálóval és elküldi neki először a fejlécet – feladó, címzett, tárgy – majd magát a levelet.

Ezt megelőzően természetesen össze kell állítani a címzettek listáját, ami részben szintén egy SQL lekérdezés segítségével lett megvalósítva.

Mivel a felhasználó a neki felkínált listából nem csak személyek vagy cégek címeit választhatja ki címzettként, hanem csoportokat is, ezért szükségessé vált a csoportok „kibontása”, majd az így létrejött lista ellenőrzése ismétlések elkerülése végett.

Az e-mail küldés tesztelése során problémaként jelentkezett, hogy jelenlegi verzióban a levél nem tartalmazhat ékezetes karaktereket az *ASCII* kódolás miatt.

Fejlesztési lehetőségek

A program felhasználói felületén több lehetőség is kínálkozik a fejlesztésekre. Sok esetben, jelentős mértékben lehetne egyszerűsíteni az alkalmazás kezelését, például több Hotkey létrehozásával, vagy az adatok szerkesztését lehetővé tevő formok egyszerűbb megjelenítésével, de az egyes beviteli mezők tartalmának tárolása is egyszerűsíthetné a kezelést.

Az adatbázis kezelést végző SQL lekérdezéseket a gyorsabb futtatás és a kisebb hálózati adatforgalom érdekében talán érdemesebb lenne tárolt eljárások hívásával végezni és esetenként a query stringeket is lehetne egyszerűbbé és gyorsabban végrehajthatóvá tenni. Természetesen ezekhez viszont az adatbázisszerver cseréjére lehet szükség amennyiben a felhasználó – vagy esetlegesen felhasználók – nagy, több ezer rekordból álló adatbázist kívánnak kezelni. Ez viszont ritkán fordul elő és a hazánkban egyre rohamosan terjedő szélessávú internet kapcsolatok miatt sem feltétlen lehet szükséges.

A program fejlesztése során tervezve volt a jelenlegihez képest nagyobb mértékben támogatni a körlevélküldést is, így ezen a téren is lenne még mit javítani.

Az e-mail küldés tesztelése során problémaként jelentkezett, hogy jelenlegi verzióban a levél nem tartalmazhat ékezetes karaktereket az *ASCII* kódolás miatt.