Fejlesztői leírás az Elfoglaltság programhoz

(a program működése és továbbfejlesztési lehetőségek)

Készítette:
Nyíri Gábor, **hdd@nc-studio.com**
GDF Abakusz regisztrációs kód: GDFAba26

Tartalomjegyzék

1. Feladatleírás

Egy tanári kar számítógép segítségével szeretné nyilvántartani elfoglaltságait – hogy könnyebb legyen szabad kapacitást találni egy-egy tanári kötelezettség *(előadás, gyakorlat, vizsga, konzultáció stb.)* elvégzésére vagy időpontot találni valamilyen közös rendezvénynek *(értekezletnek, kirándulásnak, munkamegbeszélésnek stb.)*.

1.1. Nem funkcionális követelmények

Módszertan:

- Egységesített eljárás, UML

Fejlesztési környezet:

- Visual Studio 2005 Professional
- Microsoft Windows XP Professional
- CASE eszköz: Microsoft Visio 2003
- Microsoft SQL Server 2000 Enterprise Edition
- Microsoft SQL Server 2005 Management Studio Express CTP

Futtatási környezet:

- Microsoft Windows operációs rendszer – *Ajánlott: Windows XP Professional 32bit*
- Microsoft .NET Framework 2.0

1.2. Funkcionális követelmények

A tanárokat és a kötelezettségeket a diszpécser tartja számon. Minden tanárról nyilvántartja például a nevét és azt, hogy milyen tantárgyakat tanít. Egy kötelezettség egy vagy több alkalomra szólhat, és a következő adatokat lehet róluk jegyezni:

- **Azonosító**
- **Tantárgy**
- **Típus** – *például előadás, gyakorlat, vizsga, konzultáció*.
- **Évfolyam** – például Műszaki informatika/I.
- **Csoportszám**
- **Napok** – rendszerint ezekre esnek a nyilvántartandó kötelezettségek. *Például: kedd, csütörtök.*
- **Tanár neve** – *ez a tanár teljesíti a kötelezettséget*
- **Létszám** – *a jelentkezettek, illetve részt vevők száma*
- Minden alkalomhoz a **dátum**, s az, hogy valamettől valameddig *(8 és 20 óra között egész órától egész óráig)*, ezenfelül a helyszín
- **Megjegyzés**

A kötelezettségek listája szűrhető **tanár, kezdési dátum és a befejezés dátuma** szerint**.**
Egy-egy tanár elfoglaltságait táblázatnak kell mutatnia. A táblázat egyik dimenziója a nap óráit *(8–9, 9–10, 10–11, ..., 18–19, 19–20)*, a másik dimenziója a napokat mutatja. Elvárás, hogy egyszerre egy hét legyen látható a képernyőn a könnyű áttekinthetőség végett. A táblázat celláiban levő különböző típusú elfoglaltságokat különböző színekkel jelezzük, melyeket diszpécser adhasson meg!

1.3. Rendszerkövetelmények

A rendszer adjon lehetőséget közös rendezvények – *például értekezlet, kirándulás vagy munkamegbeszélés* – tervezésére, és adjon tájékoztatást, a következőkről:

- **Az összes tanár elfoglaltsága valamely napon vagy napokon:** Egy táblázatban az egyik dimenziója a tanárok, a másik dimenziója pedig a nap óráit jelölje *(8–9, 9–10 stb.)*
- **Mely tanárok vállalhatják el valamely tanári kötelezettséget?** Jelezni kell, ha egy tanárnak a kötelezettség vállalásával 6-nál több órája lenne!
- **Van-e lehetőség valamely adott időszakban egy bizonyos rendezvény megtartására?** A rendezvény több alkalomra is kiterjedhet. Figyelembe kell venni, hogy a rendezvénynek nélkülözhetetlen tagjai is lehetnek! Meg kell adni a szóba jövő lehetőségeket, és fel kell tüntetni a lehetőségek mellett azt is, hány tanár vehetne részt a rendezvényen! A lista a tanárok száma szerint kell, rendezve legyen – *az álljon legelöl, ha mindenki el tudna jönni!*

1.4. Az elfoglaltságtábla szerkesztési lehetőségei

- **Új elfoglaltság felvétele** – a tanári kötelezettségek listájából kiválasztunk valamit *(ha még nem választotta ki más tanár)*. Ekkor a kötelezettségek listájában kitöltődik a tanár neve. Ezt a kötelezettséget más már nem választhatja. *Egyéni elfoglaltságokat is lehessen felvinni – például fogadóórát, saját szervezésű konzultációt, sőt magánelfoglaltságot is*
- **Elfoglaltság törlése** – Megszabadulni is lehessen egy kötelezettségtől: vissza lehessen tenni a választható kötelezettségek listájába. Ezt csak a legelső esedékesség időpontja előtt lehet megtenni.

1.5. Belépés

A rendszer az előírások alapján **többfelhasználós** kell, legyen, a megfelelő jogosultságokkal. A rendszerbe felhasználói névvel és jelszóval lehet belépni. Egy tanár elfoglaltságait maga a tanár vagy a diszpécser tartja karban. A diszpécser esetleges változtatásairól a rendszer naplót vezet.

2. Szótár

Elfoglaltság ..... Egy vagy több tanár számára elfoglaltságot jelentő esemény

Kötelezettség ..... Lásd: elfoglaltság

Tanár ..... Egy ember, aki az adott intézményben dolgozik és az adatok egy része felett rendelkezhet – új eseményt vehet fel, törölhet, stb.

Tantárgy ..... A tanárok által tanított ismeretkek egy szűkebb része

Csoportszám ..... Az adott csoport azonosító száma

Évfolyam ..... Az adott tanulói csoport hányadik éve folytatja tanulmányait

Típus ..... Elfoglaltság lehetséges fajtája, típusa – pl.: előadás

Létszám ..... Egy eseményen résztvevő tanárok száma

Diszpécser ..... Az a személy, aki a tanárokat és a kötelezettségeket tartja számon.

Felhasználónév ..... Az a karakterlánc, amellyel egy felhasználó azonosítja magát a rendszerben

Felhasználó ..... A programot használó személy – pl. tanár, diszpécser.

Jelszó ..... Hitelesítő adat/karakterlánc, melynek egy példánya minden felhasználónévhez hozzá van rendelve. Célja: biztosítani az adatok védelmét

Közös rendezvény .... Olyan kötelezettség, amelyre minden tanár hivatalos – *például értekezlet, kirándulás vagy munkamegbeszélés*

Beépített fiók ..... Az a felhasználónév – és a hozzátartozó jelszó – amellyel lehetőség van egy diszpécser jogosultságokkal rendelkező felhasználó létrehozására és az adatbázis-kapcsolat beállításait módosítani – első indításkor lehet szükség rá. *Az adatbázis-kapcsolat sikertelen felépítése esetén van lehetőség a használatára.*

3. SQL háttér

Minden információt szinte kivétel nélkül adatbázisban érdemes tárolni. Olyan kivételekre kell gondolni, mint az adatbázishoz való kapcsolódás paraméterei, amit a Windows regisztrációs adatbázisban vagy egy konfigurációs állományban célszerű tartani.

Azért nagy előny, ha minden adat adatbázisban tárolódik, mert ilyen formában az archiválás jelentősen leegyszerűsödik, és jóval kézben tarthatóbbá válik.

Az adatbázismotor vagy SQL 2000 illetve annak frissebb verziója, vagy SQL 2003 *Express – az SQL 2003 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 ezek elkészítése jelentősen megnövelné a fejlesztési időt, ezért várhatóan az adatbázis-manipuláció nem lesz teljes egészében kizárólag tárolt eljárásokon keresztül megvalósítva. Az egyes táblák közti konzisztenciát még a program vagy a tárolt eljárások fogják fenntartani és nem az adatbázis szerver. Erre feltételezhetően adatbázis oldalon is van megoldás.

Az adatbázis-kezelések minden esetben *sqlDataAdapter* és *sqlCommand* objektumok segítségével történnek. Jelen dokumentációban nem kerül feltüntetésre minden lekérdezés csak azok, amelyek nagyobb bonyolultságúak lehetnek, ezzel segítve a dokumentáció áttekinthetőségét és megértését.

Az adatok kezelése során optimista konkurenciakezelést fogok alkalmazni, mivel feltételezhető, hogy az adatokhoz de legalábbis az egyes felhasználókhoz tartozó adatokat egyszerre csak egy helyről fogják használni – *hibaforrást jelent, ha a diszpécserrel párhuzamosan próbálja egy felhasználó módosítani az adatait.*

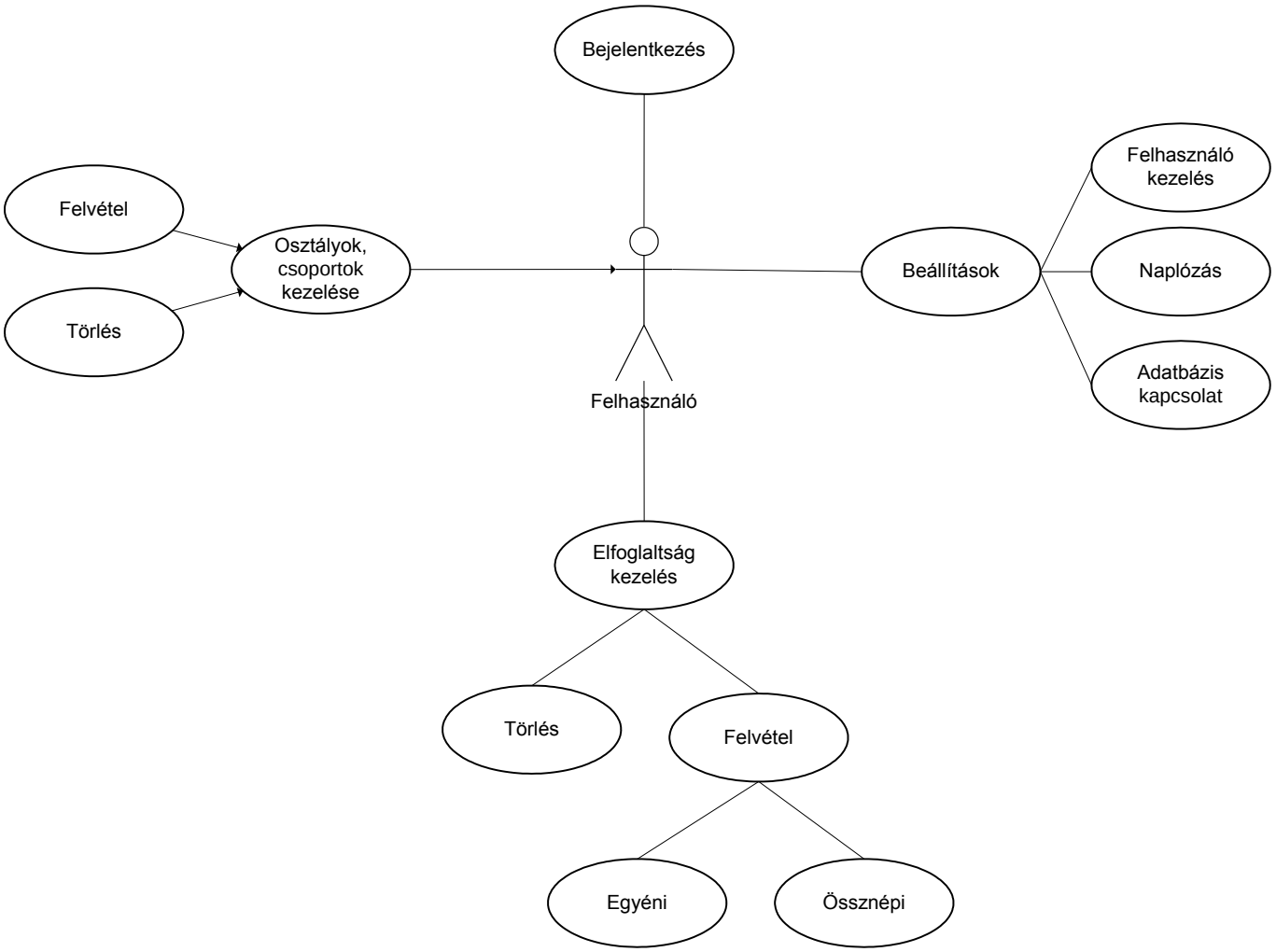
Fontos, hogy minden felhasználónak rendelkeznie kell a megfelelő jogosultságokkal az SQL kiszolgálón – *tárolt eljáráshívás, lekérdezés, nézet, beszúrás, törlés*

Adatbázis létrehozását a program nem támogatja, viszont ennek elősegítése érdekében készül egy backup, egy mintaadatokkal feltöltött és egy üres adatbázisról **elfoglaltsagok.bck** és **samples.bck** néven illetve egy script fájl, ami egy üres adatbázis létrehozását célozza meg. Mindkét állomány a **Database backup** alkönyvtárban található majd meg.

[1] Csak bizonyos feltételek teljesülése esetén!

4. Használati esetek

A feladat leírása alapján tárjuk fel a programtól elvárt funkcionális követelményeket.



1. ábra – Use Case nézet

A használati esetek részletezése:

1. **Bejelentkezés**

   Minden felhasználónak azonosítania kell magát egy felhasználónévvel és jelszóval.

2. **Súgó**

   A program használatát megkönnyítő leírás – *felhasználói dokumentáció* – valamint a készítő névjegye.

3. **Beállítások**

   A program működéséhez szükséges beállítások.

   3.1. Felhasználó kezelés

   A tanárok és a diszpécser adatai (tanárnév, jelszó. jogosultságok)

3.2. Naplózás

A naplózás felügyelete. A keletkezett naplóadatok törlése, exportálása szöveges fájlba.

3.3. Adatbázis kapcsolat

Az adatbázis szerver eléréséhez szükséges paraméterek. Ezek elérhetőek egy XML fájlban is a kezdeti értékek megadásához.

4. Osztályok, csoportok kezelése

A felhasználó itt módosíthatja a meglévő osztályokat, illetve csoportok adatait.

4.1. Törlés

Itt van lehetősége a felhasználónak a meglévő osztályok vagy csoportok valamelyikét törölni.

4.2. Felvétel

A felhasználó itt vehet fel új osztályokat és csoportokat.

5. Elfoglaltság kezelés

A bejelentkezett felhasználó elfoglaltságainak összesített megjelenítése. A felhasználó különböző kritériumok alapján listázhatja az elfoglaltságait.

5.1. Törlés

Valamely elfoglaltság törése.

5.2. Felvétel

Új elfoglaltság felvétele.

5.2.1. Egyéni

Csak az adott személyt érintő elfoglaltság felvétele.

5.2.2. Össznépi

Mindenkit érintő elfoglaltság felvétele.

6. Tantárgyak kezelése

Tantárgyak listázása.

6.1. Törlés

Tantárgy törlése

6.2. Felvétel

Új tantárgy felvétele

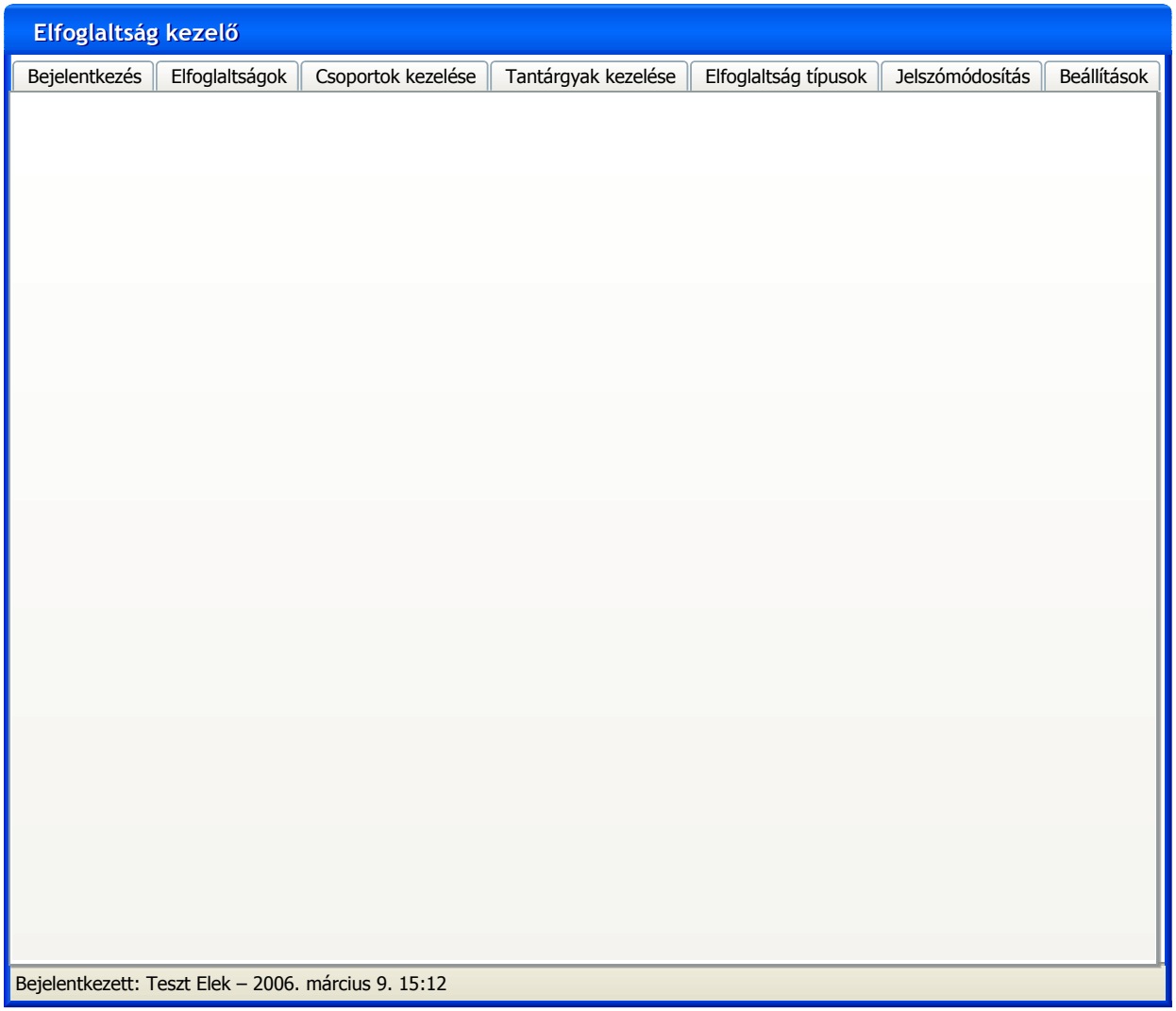
4.1. Hibalehetőségek

- Hibás hitelesítő adatok – *rossz felhasználónév és/vagy jelszó*
- Adatbázis-kapcsolat hiba
- Hibás adat megadása felvételkor, módosításkor
- Adatbázis optimista konkurenciakezelésből eredő hibák – *diszpécser és felhasználó*

5. Felhasználói felület

A program *WinForms* felületű, szabványos Windows elemeket fog használni a felhasználóval való kommunikációra. A különböző típusú műveletek miatt szükség van olyan objektumokra a M*ainForm*-on belül, amelyek egymástól jól elkülöníthetőek és külön-külön jeleníthetőek meg. Egyes funkciókat a M*ainForm* részeként is meg lehet valósítani, de várhatóan lesznek olyan feladatok is, amelyek már túlzsúfolttá tennék ezt a felületet, ezért külön *form*-ra kell kerülniük.

A kontrollok méretezésénél a 800 x 600 minimális képernyőfelbontást fogom feltételezni, mivel a napjainkban, forgalomban lévő leggyengébb képességű megjelenítők is, képesek ennél sokkal nagyobb felbontás megjelenítésére.



2. ábra – Layout scheme

5.1. A felület részei

Az ablak alsó sávjában lévő állapotsoron látható az éppen bejelentkezett felhasználó neve és a bejelentkezés pontos ideje. Közvetlen az ablak fejléce alatt található az egyes feladatokhoz tartozó objektumok elkülönítését szolgáló vezérlő. Bizonyos esetekben szükség lehet újabb ablakok megjelenítése is könnyebb áttekinthetőség végett. Új ablak megjelenésekor a felhasználó a háttérben lévő „inaktív” ablak kontroljait nem használhatja, azok mindaddig letiltva maradnak, amíg a fölötte lévő *form* meg nem szűnik, vagy legalábbis be nem záródik. Ezzel biztosítható, hogy a lehető leginkább „megvezetésre kerüljön” a programot használó személy illetve nehezebben idézhessen elő hibás működést.

6. Analízis, modell kidolgozása

6.1. A használati esetekből feltárt osztályok általános leírása

A program egy *WinForms* alkalmazás, melye **.***NET C#* nyelven fog készülni. Az egyes feladatokat külön modulok – osztályok – fognak megvalósítani a fentebbi *use case* diagramm alapján.

A használati esetből feltárt osztályok:

- **Bejelentkezés – Login**
  Ez a modul gondoskodik a felhasználó megfelelő hitelesítéséről annak érdekében, hogy csak azokat az adatokat módosíthassa, vagy tekinthesse meg melyekhez rendelkezik a megfelelő jogosultságokkal. A hitelesítés az első verzióban felhasználónév, jelszó párral történik. Hibás jelszó esetén, vagy ha az adatbázis-kapcsolat felépítése sikertelen lehetőséget ad, a beépített fiók használatára.
- **Súgó – Help**
  Gyakorlatilag a felhasználói dokumentáció. Rövid útmutató a program használatához.
- **Beállítások – Settings**
  A *diszpécser* itt módosíthatja a program működéséhez szükséges paramétereket – pl. adatbázis kiszolgáló címe, jelszava, adatbázis neve illetve a többi felhasználó/tanárok nevei, jogosultságai.
- **Elfoglaltság kezelés – MainForm**
  A bejelentkezett felhasználó itt tekintheti meg, vagy módosíthatja az elfoglaltságainak a listáját. Ez a modul a *main form*-on kap helyet.
- **Tantárgyak kezelése – SubjectOperator**
  A diszpécser itt hozhat létre új tantárgyakat, törölheti vagy módosíthatja a meglévők neveit.
- **Tanulói csoportok kezelése - GrpupOperator**
  A diszpécser itt módosíthatja a tanulói csoportok adatait, illetve vehet fel újakat, vagy törölhet meglévőket.

6.2. Adatszerkezet

Microsoft SQL Serverhez kapcsolódik.

6.2.1. Táblák

Az adatbázisba 10 táblára van szükség:

- ***Naplo***

  A program működése során keletkezett események egy részét tartalmazza.

  Mezői:

  1. *Azonosito* – Egy eseményrekordot azonosít.
  2. *Esemenykod* – A bekövetkezett esemény *Esemenyek* táblabeli kódja.
  3. *Tanar* – Annak a felhasználónak (tanárnak) az azonosítója akihez az esemény köthető
  4. *Datum* – Az esemény bekövetkezésének pontos időpontja

- ***Esemenyek***

  Eseménykódok szöveges megfelelői.

  Mezői:

  1. Azonosito – Az esemény szöveges megjelenítéséhez szükséges szám
  2. *Text* – Az esemény szöveges leírása.

- ***TanCsop***

  Az intézményen belüli tanulói csoportok adatait írja le.

  Mezői:

  1. *Azonosito* – Rekordazonosító
  2. *Evfolyam* – A csoport évfolyama
  3. *Csoportszam* – Az adott csoport sorszáma.

- ***Tanarok***

  Tanárok adatait tartalmazza.

  Mezői:

  1. Azonosito – A rekordnak az egyedi azonosítója. Erre vannak hivatkozások a többi táblából.
  2. Nev – A tanár neve.

3. Jelszo – Az a jelszó – *karakterlánc* – amivel az adott tanár nevében belehet jelentkezni a programba.
4. Jogok – Leírja, hogy az adott felhasználó – *tanár* – milyen műveletekre jogosult. Az egyes ablakokat – *használati eseteket* – engedélyezni vagy tiltja.

- ***Resztvevok***

  Az egyes elfoglaltságok résztvevői – *az elfoglaltság elszenvedője*.

  Mezői:

  1. *Azonosito* – Rekordazonosító
  2. *Elfoglaltsag* – Annak az elfoglaltságnak a rekord azonosítja amihez a résztvevők tartoznak.
  3. *Tanar* – Annak a tanárnak az azonosítója aki részt vesz az eseményen.

- ***Elfoglaltsagok***

  A tanárok elfoglaltságait tartalmazza.

  Mezői:

  1. *Azonosito* – Elfoglaltság azonosító.
  2. *Tantargy* – Tantárgy azonosító.
  3. *Tipus* – Az elfoglaltság típusának azonosítója.
  4. *TanCsop* – Megadja, hogy az adott elfoglaltság melyik tanulói csoporttal van – *ha olyan jellegű az elfoglaltság, vagyis ha megadta a felhasználó*
  5. *megjegyzes – Megjegyzés*.

- ***ElfoglaltsagTipusok***

  Az elfoglaltságok lehetséges típusait tartalmazza – *például előadás, gyakorlat, vizsga, konzultáció*.

  Mezői:

  1. *Azonosito* – Rekordazonosító
  2. *Nev* – Az elfoglaltság típusának megnevezése.
  3. *Szin* – Ezzel a színnel fog megjelenni az adott elfoglaltság típus.

- ***Tantargyak***

  A tantárgyak adatait tartalmazza.

  Mezői:

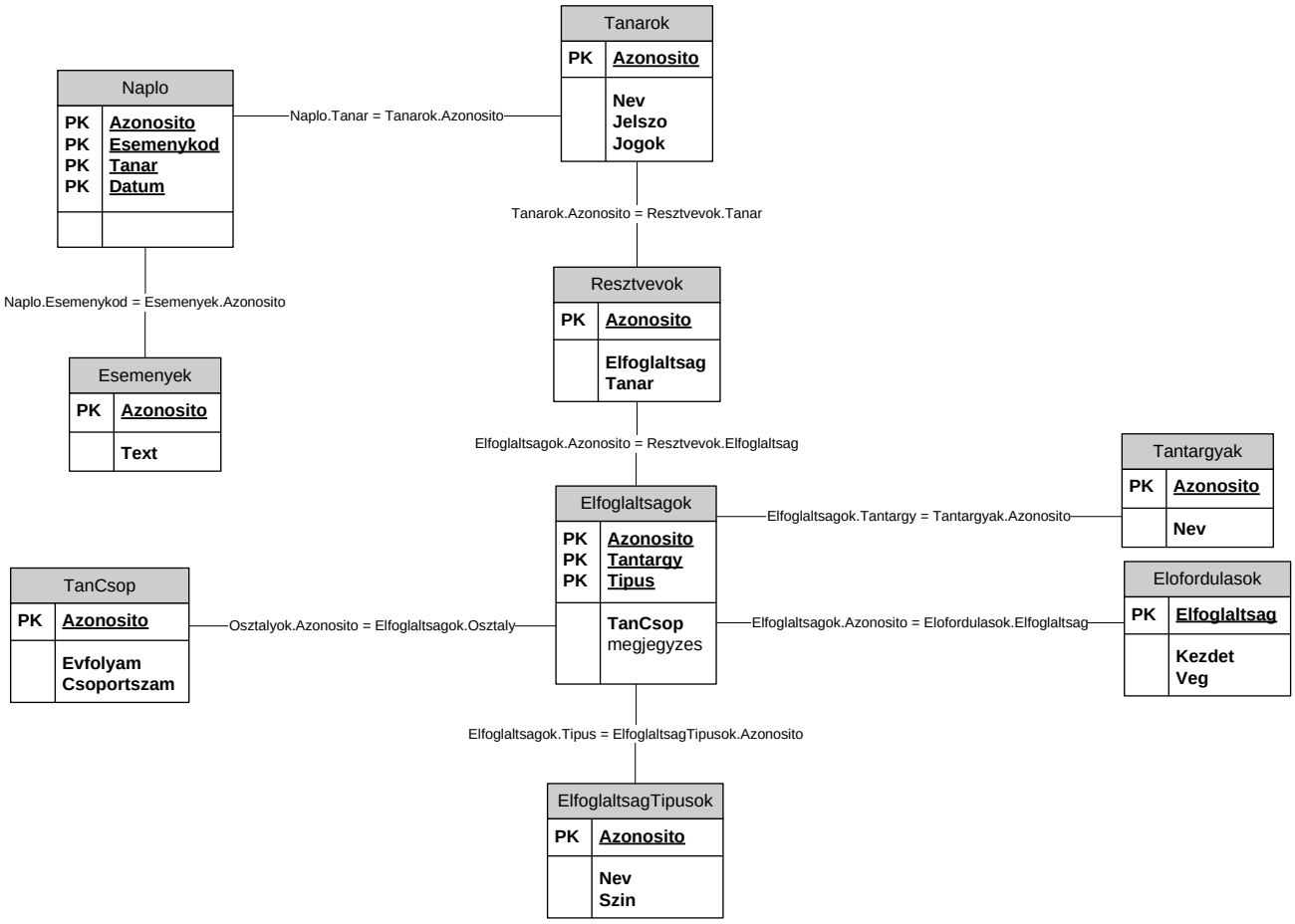
  1. *Azonosito* – Rekordazonosító.
  2. *Nev* – Tantárgy neve.

- ***Elofordulasok***

  Az elfoglaltságok egyes előfordulásai.

  Mezői:

  1. *Elfoglaltsag* – Annak az elfoglaltságnak a rekord azonosítja, amihez a dátum tartozik.
  2. *Kezdet* – Az elfoglaltság előfordulásának kezdő dátuma.
  3. *Veg – Az elfoglaltság előfordulásának vég dátuma.*



3. ábra – Adat modell

6.2.2. Nézetek

Bár biztonsági szempontból kérdéseket vethet fel, de figyelembe véve, hogy a feladat nem kíván meg különösebb biztonsági óvintézkedéseket, a felhasználó hitelesítése nézetek segítségével, fog történni, melynek során a kliens alkalmazás *LogIn* osztálya lekérdezi egy nézet segítségével az összes felhasználó adatait, majd azok alapján végzi az authentikációt. Ez a nézet egy elég egyszerű lekérdezésben megfogalmazható, így azt nem szükséges tovább részletezni.

Az eseménynaplóhoz szükséges adatokat egy ***vNaplo*** nevű nézet szolgáltatja, amely a következő képen valósítható meg, az adat-modell alapján:

```
SELECT Datum,
        (SELECT Text FROM dbo.Esemenyek
         WHERE (Azonosito = dbo.Naplo.Esemenykod)) AS Text,

        (SELECT Nev
        FROM dbo.Tanarok
        WHERE (Azonosito = dbo.Naplo.Tanar)) AS Nev
FROM dbo.Naplo
```

A tanárok számára megjelenítendő elfoglaltságokat a ***vElfoglaltsagok*** nézet valósítja meg. Paraméterként a tanár azonosítóját – *int* – és egy időintervallumot – *datetime* – kell megadni.

```
SELECT dbo.vElfok.Szin, dbo.vElfok.ElfoglaltsagTipus, dbo.vElfok.TantargyNev,
dbo.vElfok.EvfolyamCsoportszam, dbo.vElfok.Kezdet, dbo.vElfok.Veg, dbo.vElfok.Azonosito AS
ElfoglaltsagAzonosito, dbo.vElfok.Megjegyzes, dbo.vElfok.Tipus, dbo.vElfok.TanCsop, dbo.vElfok.Tantargy,
dbo.vTanElf.TanAz AS Azonosito, dbo.Tanarok.Nev
FROM dbo.vElfok INNER JOIN dbo.vTanElf ON dbo.vElfok.Azonosito = dbo.vTanElf.ElfAz INNER JOIN
dbo.Tanarok ON dbo.vTanElf.TanAz = dbo.Tanarok.Azonosito
```

Azokat az elfoglaltságokat amelyeket egy-egy tanár még felvehet az éppen számára megjelenített időintervallumban azokat az eseményeket a ***FreeJobs*** lekérdezés adja vissza:

```
SELECT 'Típus: ' + ElfoglaltsagTipus + ' Megj.:' + Megjegyzes AS Osszegzes, Kezdet, Veg, Azonosito
FROM dbo.vElfok
WHERE (NOT (Azonosito IN
        (SELECT ElfAz
         FROM dbo.vTanElf
         WHERE (TanAz = @tanar)))) AND
                (Kezdet >= @kezdet) AND
                (Kezdet < @veg) AND
                (Veg > @kezdet) AND
                (Veg <= @veg)
```

Az előző nézetben hivatkozott ***vElfok*** nézet visszaadja az összes elfoglaltságok egy táblában az összes hozzátartozó adattal – *szín, név, évfolyam, csoportszám, kezdet, vég:*

```
SELECT
        (SELECT Szin
        FROM dbo.ElfoglaltsagTipusok
        WHERE (Azonosito = dbo.Elfoglaltsagok.Tipus)) AS Szin,

        (SELECT Nev
        FROM dbo.ElfoglaltsagTipusok AS ElfoglaltsagTipusok_1
        WHERE (Azonosito = dbo.Elfoglaltsagok.Tipus)) AS ElfoglaltsagTipus,

        (SELECT Nev
        FROM dbo.Tantargyak
        WHERE (Azonosito = dbo.Elfoglaltsagok.Tantargy)) AS TantargyNev,

        (SELECT Evfolyam + '/' + Csoportszam AS Expr1
        FROM dbo.TanCsop
        WHERE (Azonosito = dbo.Elfoglaltsagok.TanCsop)) AS
EvfolyamCsoportszam,

        (SELECT Kezdet
        FROM dbo.Elofordulasok
        WHERE (Elfoglaltsag = dbo.Elfoglaltsagok.Azonosito)) AS Kezdet,

        (SELECT Veg
        FROM dbo.Elofordulasok AS Elofordulasok_1
        WHERE (Elfoglaltsag = dbo.Elfoglaltsagok.Azonosito)) AS Veg, Megjegyzes, Azonosito, Tantargy,
Tipus, TanCsop FROM dbo.Elfoglaltsagok
```

Szintén az előző lekérdezésben volt hivatkozás egy ***vTanElf*** nézetre mely az egyes elfoglaltságokat rendeli össze az azokon résztvevő tanárokkal:

```
SELECT dbo.Tanarok.Azonosito AS TanAz, dbo.Elfoglaltsagok.Azonosito AS ElfAz
FROM dbo.Elfoglaltsagok INNER JOIN
dbo.Resztvevok ON dbo.Elfoglaltsagok.Azonosito = dbo.Resztvevok.Elfoglaltsag INNER JOIN
dbo.Tanarok ON dbo.Resztvevok.Tanar = dbo.Tanarok.Azonosito
```

6.2.3. Tárolt eljárások

A Microsoft SQL 2000 Server-ben a tárolt eljárások futtatását külön engedélyezni kell a elhasználóknak az *Enterprise Managerben*!

Tárolt eljárásoknak kell megvalósítani az egyes elfoglaltságok adatbázisbeli létrehozását, törlését, az elfoglaltságokon történő részvétel bejegyzését.

Az ***AddCopy*** létrehoz egy elfoglaltsághoz egy előfordulást – még egy dátumot rendel az elfoglaltsághoz:

```
PROCEDURE dbo.AddCopy

        (
        @ElfoglaltsagID int = 5,
        @StartDate DateTime,
        @EndDate DateTime
        )

AS

        INSERT INTO Elofordulasok
        VALUES(@ElfoglaltsagID, @StartDate, @EndDate);
        RETURN
```

Az ***AddLob*** létrehoz egy új elfoglaltságot:

```
PROCEDURE dbo.AddJob

        (
        @subject int  = 0,
        @type int  = 0,
        @group int  = 0,
        @comment text  = '',
        @StartDate DateTime,
        @EndDate DateTime
        )

AS

        INSERT INTO Elfoglaltsagok
        VALUES(@subject, @type, @group, @comment);

        DECLARE @ID int;

        SET @ID = @@IDENTITY;

        INSERT INTO Elofordulasok
        VALUES(@ID, @StartDate, @EndDate);

        RETURN @ID
```

Az ***AddType*** Létrehoz egy új elfoglaltság típust:

```
PROCEDURE dbo.AddType

	(
	@szin text = '',
	@nev text = ''
	)

AS

	INSERT INTO ElfoglaltsagTipusok
	VALUES(@nev, @szin);

	RETURN (SELECT Azonosito FROM ElfoglaltsagTipusok WHERE ElfoglaltsagTipusok.Nev like
@nev and ElfoglaltsagTipusok.Szin like @szin);
```

A ***DelCopy*** töröl egy elfoglaltsághoz tartozó előfordulást – dátumot:

```
PROCEDURE dbo.DelCopy

	(
	@ElfoglaltsagID int = 5,
	@StartDate DateTime,
	@EndDate DateTime
	)

AS

	DELETE FROM Elofordulasok
	WHERE Elfoglaltsag = @ElfoglaltsagID and Kezdet = @StartDate and Veg = @EndDate;
	RETURN
```

A ***DelFromJob*** egy elfoglaltság résztvevői közül töröl egy résztvevőt - tanárt

```
PROCEDURE dbo.DelFromJob

	(
	@elfoglaltsag int = 0,
	@tanar int = 0
	)

AS
	DELETE FROM Resztvevok
	WHERE Elfoglaltsag = @elfoglaltsag and Tanar = @tanar;
	RETURN
```

A ***DelJob*** töröl egy elfoglaltságot az összes hozzátartozó adattal együtt:

```
POCEDURE dbo.DelJob

	(
	@JobID int = 0
	)

AS

	DELETE FROM Elfoglaltsagok
	WHERE Azonosito = @JobID;

	DELETE FROM Elofordulasok
	WHERE Elfoglaltsag = @JobID;

	DELETE FROM Resztvevok
	WHERE Elfoglaltsag = @JobID;

	RETURN
```

A ***JobCount*** Visszaadja, hogy egy adott időintervallumban, egy adott tanárnak hány elfoglaltsága van:

```
PROCEDURE dbo.JobCount

	(
	@start DateTime,
	@stop DateTime,
	@teacher int
	)

AS

	RETURN dbo.FJobCount(@start, @stop, @teacher);
```

A ***JoinJob*** hozzáad egy résztvevőt egy elfoglaltsághoz:

```
ALTER PROCEDURE dbo.JoinJob

	(
	@elfoglaltsag int = 0,
	@tanar int = 0
	)

AS
	INSERT INTO Resztvevok VALUES(@elfoglaltsag, @tanar);
	RETURN
```

*A **jobTypeIsExists*** megnézi, hogy létezik e már egy elfoglaltság típus. Ha nem létezik, akkor 0 értéket ad vissza.

```
ALTER PROCEDURE dbo.jobTypeIsExists
	(
	@typeName varchar(50)
	)
AS

	DECLARE @count int;

	RETURN (SELECT count(*)
	FROM ElfoglaltsagTipusok
	WHERE upper(Nev) = upper(@typeName))
```

Az **existsTeacher** visszaadja, hogy egy adott felhasználónév már létezik e. Visszatérési értéke *nulla,* ha még nem létezik.

```
PROCEDURE dbo.existTeacher

	(
	@name varchar(50)
	)

AS
	RETURN (SELECT count(*) FROM Tanarok
	WHERE Nev = @name)
```

Léterhoz egy új felhasználót/tanárt.

```
ALTER PROCEDURE dbo.newTeacher

	(
	@nev varchar(50),
	@jelszo varchar(50),
	@jogok int
	)

AS

	INSERT INTO Tanarok VALUES(@nev, @jelszo, @jogok);
	RETURN
```

6.2.4. Függvények

Az **FJobCount** függvény visszaadja, hogy egy adott időintervallumban, egy adott tanárnak hány elfoglaltsága van. Mivel a függvényeket kívülről nem lehet elérni ezért szükséges egy tárolt eljárás, – **JobCount** – mely segítségével a függvény kívülről paraméterezhetően elérhetővé válik. Erre azért van szükség, mert *SELECT*-ből csak függvényt lehet hívni, tárol eljárást nem.

```
FUNCTION dbo.FJobCount
      (
      @start DateTime,
      @stop DateTime,
      @teacher int
      )
RETURNS int
AS
      BEGIN
      RETURN (
                SELECT count(*)
                FROM Elofordulasok
                WHERE (Elfoglaltsag IN
                                  (
                                    SELECT Elfoglaltsag
                                    FROM Resztvevok
                                    WHERE (Tanar = @teacher))
                                  )
                                  AND
                                  (
                                    (Kezdet between @start and @stop) or
                                    (Veg between @start and @stop) or
                                    (@start between Kezdet and Veg) or
                                    (@stop between Kezdet and Veg))
                                  )
              );
      END
```

6.2.5. Hibakódok – események tábla

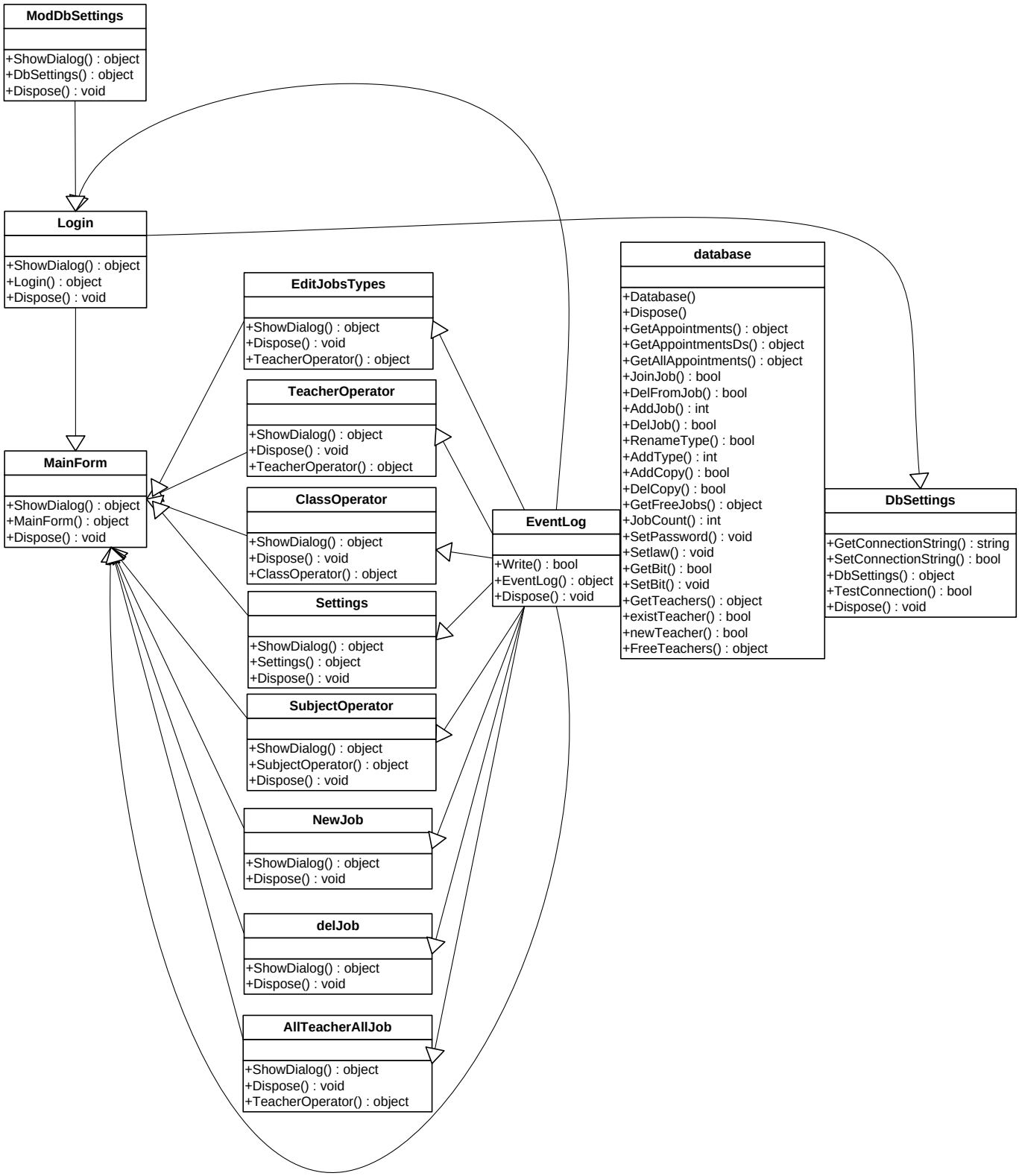
Az események naplózása során az adatbázisba, csak az események szövegéhez rendelt kódok kerülnek tárolásra, így az alkalmazás lokalizációja könnyebb és a hibaüzenetek is egységessé válnak.

A hibakódok szöveges megfelelői a következők:

0	Hibás felhasználónév vagy jelszó! Sikertelen bejelentkezés!
1	Sikeres bejelentkezés!
2	Sikeres kijelentkezés!
3	A jelszó sikeresen módosítva!
4	Az eseménynapló törölve!
5	Csoport adatok módosítva!
6	Tantárgy lista módosítva!
7	Egy felhasználó jogosultságai sikeresen módosítva lettek!
8	Egy új felhasználó létrehozva!

7. Az osztályok kapcsolatai

Az osztályok eljárásait, változóit, kapcsolatait a következő ábra mutatja:



4. ábra - Osztály diagramm

7.1. Osztályok leírása példányosodásuk alapján

Login – Ennek az osztálynak a feladata, hogy hitelesítse a felhasználót annak nevével és jelszavával. A sikeres bejelentkezésről illetve sikeres kijelentkezésről eseményt generál, melyekben visszaadja a főprogram számára a be vagy kilépett felhasználó nevét, azonosítóját és jogosultságait leíró adatot. A program futása során csak egyszer példányosul az alkalmazás betöltése során és végig fenn is marad, azonban majdhogynem semmi szerepe nincsen. A felhasználó számára, mint vizuális elem jelenik meg a *MainForm*-on. A *UserControl* osztály egy leszármazottja.

ModDbSettings – A futás alatt többször is példányosíthatja a *Login* osztály, attól függően, hogy a felhasználó hányszor a bejelentkezést megelőzően akarja e szerkeszteni az adatbázis-kiszolgálóhoz való kapcsolódás paramétereit. Bezárásával egyidejűleg az osztály megszűnik. Ennek megfelelően a feladata az adatbázis-kapcsolathoz szükséges paraméterek grafikus megjelenítése, azok szerkeszthetővé tétele. A felhasználó számára, mint önálló ablak jelenik meg. A *form* osztály egy leszármazottja.

MainForm – Ez az osztály testesíti meg gyakorlatilag a főprogram szerepét. Minden további osztályt ez fog meghívni és köztük a kapcsolatot valamelyest fenntartani, a különböző metódushívások és események lekezelése révén. Az alkalmazás futása során csak egyszer példányosulhat és végig aktívan fenn kell maradnia feladataiból eredően. A felhasználó számára, mint az alkalmazás „főablaka” jelenik meg. A *form* osztály egy leszármazottja.

TeacherOperator – Ennek az osztálynak a feladata, hogy grafikusan szerkeszthetővé tegye az összes felhasználó – *tanár* – adatait, jogosultágait, stb. A futás során csak egyszer példányosítja a *MainForm* osztály, és a kilépésig fenn is marad. A felhasználó számára mint vizuális elem jelenik meg a *MainForm*-on. A *UserContorl* osztály leszármazottja.

ClassOperator – Feladata az évfolyamok illetve a csoportok nyilvántartása. A futás során csak egyszer példányosítja a *MainForm* osztály. A felhasználó számára egy meglehetősen egyszerű vizuális elemként jelenik meg. A *UserControl* osztály leszármazottja.

Settings – Feladata a program az előbbi kategóriákba be nem sorolható beállításainak grafikus megjelenítése és szerkeszthetővé tétele.

SubjectOperator – Feladata a tantárgyak listájának kezelése. A futtatás során csak egyszer kerül példányosításra a *MainForm* osztály részéről. A felhasználó számára egy meglehetősen egyszerű vizuális vezérlőként jelenik meg. A *UserControl* osztály leszármazottja.

NewJob – Feladata a létrehozandó elfoglaltság adatainak a grafikus szerkeszthetővé tétele. A program futása során többször is példányosulhat igény szerint. A felhasználó számára önálló ablakként jelenik meg. Bezárásával egyidejűleg megszűnik. A *form* osztály leszármazottja.

delJob – Feladata a törlendő elfoglaltság adatainak a grafikus megjelenítése és megerősítés kérése a felhasználótól a törlést illetően. A program futása során többször is példányosulhat igény szerint. A felhasználó számára önálló ablakként jelenik meg. Bezárásával egyidejűleg megszűnik. A *form* osztály leszármazottja.

AllTeacherAllJob – Feladata az összes tanár elfoglaltságainak egyidejű, grafikus megjelenítése. A felhasználó számára mint vizuális elem jelenik meg. A program futása során csak egyszer példányosul és végig fennmarad. A *UserControl* osztály leszármazottja.

EventLog – Feladata a program futtatása során keletkező események naplózása. A felhasználó számára soha nem jelenik meg. A program futása során többször is példányosul valahányszor naplózandó esemény keletkezik.

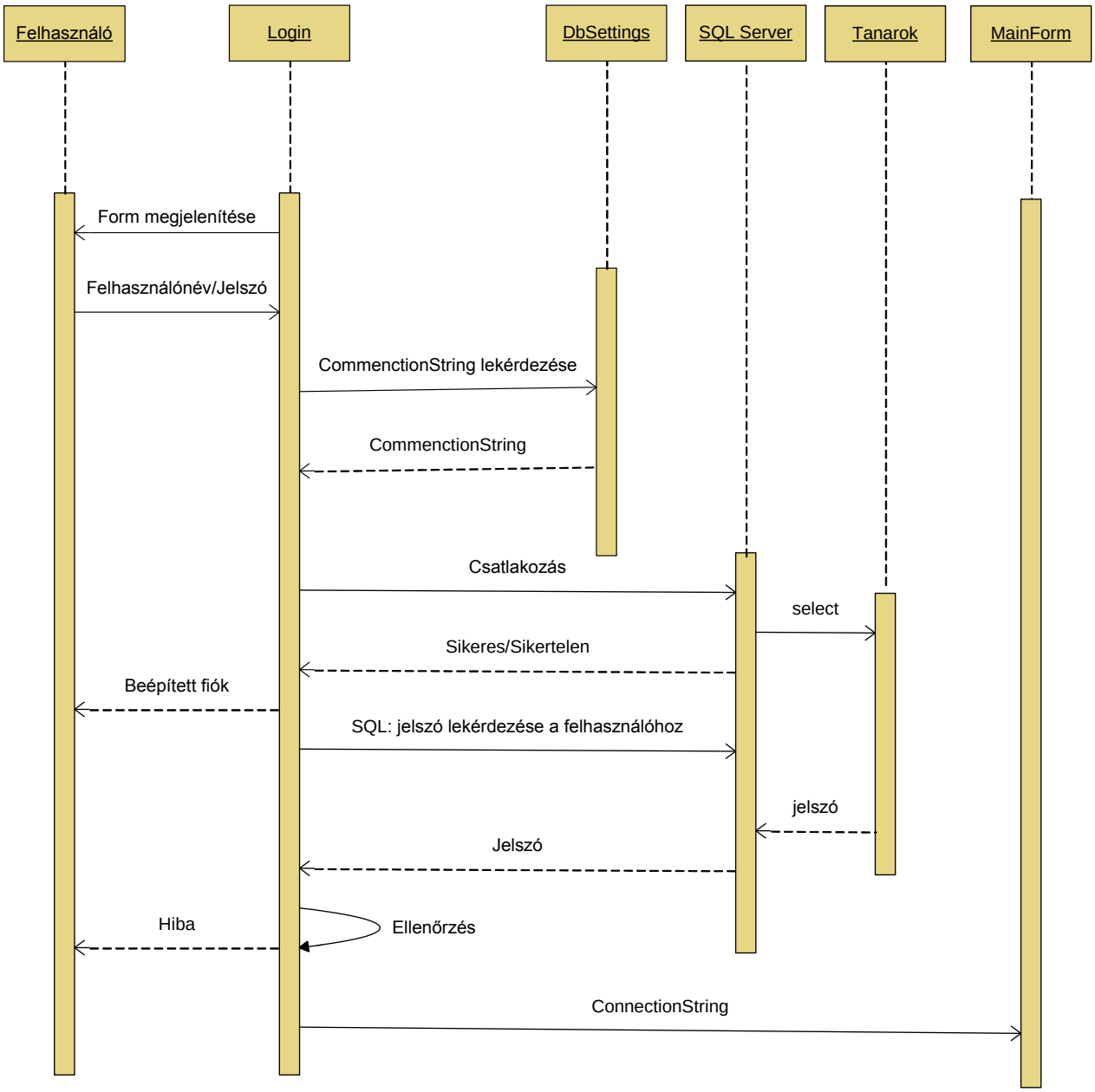
Database – Feladata az adatbázissal való közvetlen kapcsolattartás, annak kezelése. A futás során többször is példányosul. Minden példány egy tranzakciót kell végrehajtson. A felhasználó számára ez az osztály sosem jelenik meg.

DbSettings – Feladata az adatbázis-kapcsolat paramétereinek tárolása illetve szükség esetén annak visszaadása vagy módosítása. A futtatás során többször is példányosulhat valahányszor szükség van ilyen jellegű műveletre. Általában egy osztály csak egyszer példányosítja a konstruktorában, majd segítségével kiolvassa a *connectionString*-et.

EditJobsTypes – Feladata az elfoglaltság típusok grafikus szerkeszthetővé illetve új elfoglaltságfajták felvételének lehetővé tétele. A felhasználó számára mint vizuális elem jelenik meg. A program futtatása során csak egyszer példányosul és végig fennmarad. A *UserControl* osztály leszármazottja.

8. Folyamatok részletezése

Fontosabb folyamatok bemutatása szekvencia-diagrammokkal:



5. ábra – Bejelentkezés

9. Tesztelés

Minden osztály a fejlesztésükkel párhuzamosan tesztelendő, annak érdekében, hogy az eseményvezéreltségből adódó hibák egyszerűbben felderíthetőek legyenek.

A tesztelés a valóságos használatnak a lehető leginkább megfelelő módon kell történnie valósághű adatokkal. Ez azt jelenti, hogy felváltva több felhasználó nevében szükséges különböző elfoglaltságok létrehozása illetve azoknak a böngészése – a feladatdefinícióban leírt műveletek tematikus tesztelése.

A tesztadatokkal feltöltött adatbázis megtalálható lesz a **Database backup** alkönyvtárban **samples.bck** néven.

A tesztelés folyamán egy esetben, eddig nem tisztázott okokból kifolyólag az alkalmazást nem lehetett elindítani, mert az egy Windows hibaüzenettel – *nem a framework általános kivételekről értesítő ablakával* – leállt, melyet a rendszer automatikusan *továbbítani akart a Microsoft felé* is. A hibaüzenet elemzése során felmerült a gyanúja, hogy az adatbázis-kapcsolatot nem sikerült valamilyen ismeretlen okból kifolyólag felépíteni.

10. Fejlesztési lehetőségek

- Az idő rövidsége miatt nem volt lehetőségem, a felhasználói dokumentáción kívül más, helyérzéken súgó készítésére.

- Az adatbázis-kezelés során keletkező kivételek kezelését célszerű lenne felülvizsgálni, mert azok korántsem biztos, hogy a megfelelő helyen kerülnek megjelenítésre és ezáltal elképzelhető, hogy a tranzakciók hiba esetén nem kerülnek törlésre, ami nyilvánvalóan veszélyezteti az adatbázis konzisztenciát.

- A programot ellehetne készíteni több idegen nyelven is. Ezt hivatott többek között elősegíteni az adatbázis *események* táblája is.

- A felhasználó hitelesítése adatbiztonsági szempontból biztonságosabbá válhatna tárolt eljáráshívás használatával.

- Az adatbázisból való törlés jelenleg teljes egészében nem megoldott az idő rövidsége miatt. Bár különösebben nem lenne összetett feladat, ám a minden felhasználó számára korrekt, egyértelmű és egyszerű kezelés kigondolása miatt nem jutott rá idő.

11. Felhasznált irodalom

- Hatékony C# / Bill Wagner. – Kiskapu Kft : Budapest, 2005
- Bemutatkozik a Microsoft NET / David S. Platt. – Szak Kiadó Kft : Bicske, 2001
- Programtervezés / Douglas Bell, Ian Morrey, John Pugh. – Kiskapu Kft : Budapest, 2003
- Az UML / Molnár Ágnes. – NetAcademia Kft : Budapest, 2002
- Zárolások az SQL 2000-ben / Soczó Zsolt. - NetAcademia Kft : Budapest, 2001
- SQL kézikönyv / Stolnicki Gyula. – ComputerBooks Kiadói Kft : Budapest, 1999