SZAKDOLGOZAT

Szabó Attila

Debrecen
2010

Debreceni Egyetem
Informatikai Kar
Információtechnológiai tanszék

Vállalatok pénzügyi dokumentumainak kezelőrendszer fejlesztése Java 6-ban

Témavezetők

Espák Miklós
Egyetemi tanársegéd

Bóka Antal
Prog. Mat.

Készítette

Szabó Attila
Prog. tervező inf. BSC

Debrecen
2010

Plágium - Nyilatkozat

Szakdolgozat készítésére vonatkozó szabályok betartásáról nyilatkozat

Alulírott (Neptunkód: GI5SRR……) jelen nyilatkozat aláírásával kijelentem, hogy a *Vállalatok pénzügyi dokumentumainak kezelőrendszer fejlesztése Java 6-ban*……..

című szakdolgozat/diplomamunka

(a továbbiakban: dolgozat) önálló munkám, a dolgozat készítése során betartottam a szerzői jogról szóló 1999. évi LXXVI. tv. szabályait, valamint az egyetem által előírt, a dolgozat készítésére vonatkozó szabályokat, különösen a hivatkozások és idézések tekintetében.

Kijelentem továbbá, hogy a dolgozat készítése során az önálló munka kitétel tekintetében a konzulenst, illetve a feladatot kiadó oktatót nem tévesztettem meg.

Jelen nyilatkozat aláírásával tudomásul veszem, hogy amennyiben bizonyítható, hogy a dolgozatot nem magam készítettem vagy a dolgozattal kapcsolatban szerzői jogsértés ténye merül fel, a Debreceni Egyetem megtagadja a dolgozat befogadását és ellenem fegyelmi eljárást indíthat.

A dolgozat befogadásának megtagadása és a fegyelmi eljárás indítása nem érinti a szerzői jogsértés miatti egyéb (polgári jogi, szabálysértési jogi, büntetőjogi) jogkövetkezményeket.

.....

hallgató

Debrecen, 2010. április 19.

Tartalomjegyzék

1. Bevezetés

Napjainkban számtalan kis- és középvállalkozás létezik, melyek száma napról napra nő. Ezek pénzügyi dokumentumaikat különféle módokon kezelik: vannak melyek már elektronikus úton végzik ezt a tevékenységet, azonban sokuk még a papír alapú megoldás híve.

Szakdolgozatom célkitűzése egy szoftver létrehozása volt, mely talán még több vállalatot ösztönöz az e-ügyintézésre váltásra. Ennek megvalósításakor a következő szempontok lebegtek szemem előtt:

- kis méretű szoftver létrehozása
- a program használata egyszerű legyen (minimális informatikai ismeretekkel is lehessen megfelelően kezelni)
- bárhonnan elérhető adatbázis használata
- platform függetlenség.

Tudtam, hogy számlázó szoftverből már nagyon sok létezik, azonban a legtöbb csak egy bizonyos operációs rendszeren képes futni (s ez az esetek döntő többségében valamilyen Windows). Így esett választásom a Java™[1] nyelvre, melynek fő előnye, hogy a forráskód fordítása után előálló bájtkód bármilyen rendszeren elfut, melyen van Java futtató környezet (Java Runtime Enviroment – JRE).

A bárhonnan elérhető adatbázis, illetve a kis méretű szoftver együtt jár. Maga a program bár összetett, nem áll olyan sok komponensből, hogy néhány megabyte-nál többet foglalna. Az ilyen típusú programok esetében mindig az adatok tárolásához szükséges a nagy tárterület. Bár az adatok tárolása kivitelezhető lett volna a gépen tárolt állományokban, a megnövekedett tárigény miatt ezt a lehetőséget inkább elvetettem, s az adatbázis használata mellett döntöttem. Igaz, hogy az adatbázis telepíthető az adott felhasználó gépére, de a lényeg itt épp az, hogy bármilyen, akár ingyenes tárhelyszolgáltatóhoz telepíthetőek, ezáltal mentesül a felhasználó számítógépe a terhelés alól (ebben az esetben, hálózati kapcsolat szükséges).

Az adatbázisok használatának több előnye van:

1 A továbbiakban csupán Java

- A tárigény mellett, a szerializált objektumok visszaolvasása jelentősen lassítaná a program futását
- Az adatbázis kapcsolaton keresztül az adatok lekérése amellett hogy egyszerű, a lekérdezések eredményét könnyedén lehet szűrni, azokban keresni
- Adatbázisok használata mellett lehetőség van felhasználói autentikációra, így a szoftver használata, illetve az adatok tárolás biztonságosabbá válik

A program jelenleg két adatbáziskezelő-rendszer használatát támogatja, melyek száma azonban később – a Java nyelvnek köszönhetően – egyszerűen bővíthető. Ezek:

1. Az Apach Derby, mely könnyen telepíthető és használható, ezen felül Java alapú, s biztosít egy beépített drivert, melynek segítségével bármilyen Java programból könnyedén elérhető.
2. A MySQL, mely a világ legnépszerűbb ingyenes adatbázis kezelő rendszere[2], s a Derby-hez hasonlóan képes könnyedén együttműködni a Java JDBC szolgáltatásával.

Utolsó kitűzött szempontom, a könnyű használat elérésében, a Java nyelv modularitásának a kihasználására volt szükség. Szoftverem felépítése egyszerű lett, a főablakban jól elkülönülnek az egyes komponensek. Hogy még inkább segítsem a papírról való áttérést, az új tételek felvitelének rendszere, valamint a kinyomtatott számlák felépítése teljesen megfelel a hagyományos kinézetű, az APEH által is jóváhagyott, boltban kapható számlatömbök lapjainak. Mindezek mellett, mivel magyar felhasználóknak készült a programom, szövegezése is magyar lett.

Szakdolgozatom hátralévő részében részletesen bemutatom a felhasznált technológiákat, leírást adok a szoftver fejlesztésének folyamatáról, s zárásképp a szoftverhez tartozó egyszerűbb felhasználói kézikönyvet készítem el.

2 A MySQL hivatalos oldaláról: http://www.mysql.com/about/

2. Alkalmazott technológiák

2.1. Java

A Java egy imperatív, objektum orientált programozási nyelv, melyet James Gosling kezdett el fejleszteni a Sun Microsystemsnél. Első kiadása 1995-ben érkezett.

Szintaktikáját nagyrészt a C és a C++ nyelvből örökölte, sőt alapvetően a C++ tökéletesítéseként jött létre[3], azonban megszünteti annak mutatóorientáltságát, ezáltal biztonságosabbá téve a fejlesztést, s egyszerűbb objektum modellt valósít meg.

A Java nyelv különlegessége, hogy fordítóprogramos és interpreteres nyelv. Fő erőssége a megírt alkalmazások platformfüggetlenségében rejlik, nevezetesen: a Java fordítója egy bájtkódot hoz létre (*.class fájlban), melyet a Java Virtuális Gép (JVM – Java Virtual Machine), futtat le.

A JVM ugyan már platformfüggő, de elérhető minden nagyobb rendszerre. A JVM tekinthető egy absztrakt számítógépnek, amelyet egy általában szoftveresen szimulálunk egy konkrét platformon (ugyanakkor lehet a JVM egy tényleges hardver is). Ennek az előnye, hogy mivel minden platformon ugyanaz a JVM érhető el, ezáltal minden rendszeren ugyanúgy futnak le a programok.

2.1.1. A Java nyelv története

Megjelenés előtt – A Java nyelv fejlesztésének projektje 1991. júniusában kezdődött, az egyik set-top box projekthez (ez egy jelátalakító amit a televízióhoz lehet csatlakoztatni). Eredeti neve Oak volt, mert egy tölgyfa állt Gosling ablaka előtt (oak jelentése tölgy), s csak később nevezték el Javanak, melyet szavak véletlenszerűen összeállított listájából választottak ki[4]. Gosling célja egy virtuális gép implementálása volt, és hogy egy C/C++ alapokon nyugvó nyelvet hozzon létre.

1.0 – A Java első kiadása 1995-ben jelent meg. Szlogenje a WORA volt, vagyis Write Once, Run Anywhere (Írd meg egyszer és futtasd bárhol), kötetlen futtatást biztosítva a

3 Juhász István – Programozás 2 (17. oldal)
4 Frank Yellin nyilatkozata - http://www.javaworld.com/javaworld/jw-10-1996/jw-10-javaname.html

népszerű platformokon. Állítható biztonsági szinteket nyújtott, valamint engedélyezte a hálózati- és fájlhozzáférési megszorításokat. A népszerű böngészők hamarosan képesek lettek futtatni a Java appleteket a weboldalakon belül, s így a Java hamar népszerűvé vált.

1.2 – 1998. decemberében jelent meg. Az új kiadásnak a különböző platformoknak megfelelően különböző változatai jelentek meg, úgymint:

- J2SE – Java Standard Edition
- J2EE – Java Enterprise Edition, mely a vállalati környezetet célozza meg
- J2ME – Java Micro Edition, mely a mobiltelefonokra való fejlesztés eszköze

2006-ban marketing okok miatt átnevezték a kiadásokat Java SE-re, Java EE-re illetve Java ME-re.

Mára a Java a 6-os verziónál tart s hamarosan érkezik a 7-es változat.

2.1.2. Implementációk

A Java programok futtatásához a rendszerre telepíteni kell a Java futtatói környezetet (JRE – Java Runtime Environment), melynek a legújabb változata mindig elérhető ezen a honlapon: http://java.com/en/download/index.jsp

Programfejlesztésre a JDK, vagyis a Java Development Kit szolgál, mely minden alapvető, a fejlesztéshez szükséges csomagot, osztályt tartalmaz.

A Sun által fejlesztett hivatalos Java SE platform elérhető a különböző Linux disztribúciókhoz, Mac OS X-hez és a Sun saját operációs rendszeréhez a Solarishoz, valamint a Microsoft Windows rendszerekhez is, azonban a partneri hálózat miatt más, nem hivatalos Java környezetek is elérhetők ezen rendszerekre.

A Sun védjegy egyik feltétele, hogy a JRE kompatibilis legyen minden változattal, azonban a Microsoft implementációja nem támogatta az RMI (Remote Method Invocation – Távoli eljáráshívást megvalósító komponens) és JNI (Java Native Interface – Más nyelven írt forráskódok felhasználását lehetővé tévő komponens) technológiákat, helyette a saját platformspecifikus komponenseiket tették bele. Ez perhez vezetett, melyet a Sun 2001-ben megnyert, s a partneri szerződés felbontásra került. Ez az oka, hogy a Microsoft többé nem

építi bele operációs rendszerébe a JRE-t, illetve az Internet Explorer csak külső fejlesztőtől származó összetevővel képes futtatni a Java Appleteket.

2.1.3. Teljesítmény és memóriakezelés

Elterjedt hit, hogy a Java nyelven írt programok lassabbak és több memóriát használnak a többi nyelven írtnál, azonban az évek során jelentős fejlődés következett be ezen a téren. Ez egyfelől betudható a Java 1.1-ben bemutatkozó röpfordítónak (JIT – Just-in-time compiler), az új, jobban megírt osztályoknak, illetve a JVM-ben véghez vitt jelentős optimalizálásnak

A C/C++-os világban a lefoglalt memóriaterületeket manuálisan kellett felszabadítani. Ha ezt nem tettük meg, a programok futtatása közben egy idő után jelentős memóriaaprózódás következett be, futási problémák jelentkeztek, ráadásul veszélyes is volt, mivel a programozók közvetlenül hozzáfértek a memóriához, s így a rendszer működése szempontjából kritikus területeket is törölhettek, felülírhattak, ezáltal hibát előidézve.

A Java ezt az örökséget nem vette át, elzárta a memóriát a programozók elől, helyette inkább az automatikus szemétgyűjtőt (garbage collector) vezette be. Ennek lényege, hogy ha egy objektum megszűnik, a hozzá tartozó memóriaterület automatikusan kerül felszabadításra.

A szemétgyűjtő bármikor lefuthat, a programozónak nem kell meghívnia, azonban lefutása kikényszeríthető, pl:

```
System.gc();
```

A Java nem teljesen objektumorientált nyelv, mivel használja a C++-tól átvett primitív típusokat.

2.1.4. Összegzés

A Java nyelv számos előnnyel rendelkezik a legtöbb programozási nyelvvel szemben, melyek közül kétségtelenül a platformfüggetlensége a legjelentősebb.

Szigorú szintaktikája révén rendszerezésre tanít, emellett egyszerűbb programok könnyedén fejleszthetők rajta, így különösen alkalmas első nyelvnek azok számára akik szeretnének programozást tanulni.

A statisztikák szerint körülbelül 4,5 milliárd eszközön fut valamilyen Java program[5] (pc-k, mobiltelefonok, multimédiás eszközök stb), s ezek száma rohamosan gyarapszik, melynek újabb lendületet adhatnak az újonnan megjelenő technológiák, mint például a dvd-t leváltani hivatott blu-ray.[6]

Nem véletlenül, eddig körülbelül 6,5 millió aktív, hivatásos programozó választotta a Javat a fő fejlesztési nyelvének, és feltehetőleg a többszöröse ezeknek a száma akik már valaha használták, illetve csak alkalmanként fejlesztenek ezen a nyelven. Ezek miatt választottam én is szakdolgozatom megírásához és szeretnék én is hivatásos Java fejlesztő lenni.

5 http://java.com/en/about/
6 http://java.sun.com/developer/media/deepdivebluray.jsp

2.2. JDBC

A JDBC, vagyis Java DataBase Connectivity, mint ahogy a neve is mutatja nem más, mint egy, a Java nyelvhez tartozó API, mely definiálja miként érhetik el a kliensek az adatbázisokat.

A soron következő fejezetekben ejtek néhány szót magáról a JDBC-ről, rövid leírásban ismertetem az egyes verziók közti különbségeket[7], s végül a JDBC alapvető szolgáltatásainak használatát fogom bemutatni egy Java programban egyszerűbb példák felhasználásával, melyben részletesen kitérek majd a lekérdezések készítésének módjaira, illetve az általuk szolgáltatott eredmények feldolgozására.

2.2.1. JDBC mint technológia

A JDBC API a Java API része, melynek segítségével bármilyen tagolt adatokhoz való hozzáférésre alkalmas, különösen a relációs adatbázisokban tároltakéhoz.

A JDBC segíti olyan Java programok megírását, melyeknek a követező három tevékenységet kell kezelnie[8]:

1. Adatforrásokhoz (pl.: adatbázisokhoz) való csatlakozás
2. Kiválasztó és frissítő lekérdezések készítése, küldése
3. A lekérdezések eredményeinek fogadása és feldolgozása

A JDBC mint termék, négy részből áll:

- *JDBC API*: a relációs adatbázisokhoz és az egyéb adatforrásokhoz biztosít hozzáférést, illetve az itt található adatok manipulálásához nyújt eszközöket
- *JDBC Driver Manager*: a DriverManager osztály objektumain keresztül valósul meg a kapcsolat az adatbázissal, ez adja lényegében az architektúra gerincét
- *JDBC Test Suite*: ez a csomag segít eldönteni, hogy melyik JDBC Driver-re van szükség a program futtatásához. A tesztek nem átfogóak és alaposak, de sok fontos

7 Hivatalos JDBC API specifikációk felhasználásával
8 The Java Tutorial

JDBC szolgáltatás tesztelhető velük

- *JDBC-ODBC Bridge*: ODBC drivereken keresztül valósul meg a JDBC adatelérése. Ezen drivereket tartalmazó bináris állományokat minden kliens gépen be kell tölteni

A JDBC API egyaránt támogatja a kétrétegű és a háromrétegű architektúrák használatát is.

2 rétegű architektúra esetén a kliensen futó Java programok (vagy appletek) közvetlenül kommunikálnak az adatbázis szerverrel, míg 3 rétegű architektúra esetén megjelenik egy köztes réteg. Ekkor a kliens ennek a köztes rétegnek küldi a parancsokat – valamilyen protokollon keresztül –, majd a parancs innen továbbítódik az adatbázis szerver felé. A 3 rétegű architektúra előnyei közé tartozik, hogy mivel nem közvetlen elérést biztosít, jelentősen megkönnyíti mind az adatbázis, mind a kliens oldalon a szoftverfejlesztéseket, valamint számos esetben teljesítmény területén is jobbnak bizonyul a kétrétegűhöz képest.

2.2.2. Verziók

JDBC 1.0 – 1997. februárjában debütált a JDK 1.1 részeként, s már tartalmazta az alapvető eszközöket az adatbázisok eléréséhez, mint például a meghajtó programok kezelése, kapcsolatok létrehozása, lekérdezések készítése, típusok támogatása, metaadatok lekérése, használata.

JDBC 2.0 – A JDK 1.2-ben jelent meg, s a JDBC API-ját két részre bontották:

- az alap API, mely lényegében csak néhány újabb osztállyal bővült, azonban jelentős optimalizáláson esett át a teljesítmény, a funkcionalitás és az osztályok közötti összetartó erők terén, valamint most már támogatta az új SQL3 (vagy másképp SQL-99) adattípusait.

  Új funkcióként megjelent az eredményhalmazok görgethetősége, a kötegelt frissítések végrehajtása, programozott beszúrások, törlések és frissítések futtatása, teljesítmény tanácsok, teljes pontosság a `java.math.BigDecimal` értékekhez, karaktercsatornák a nemzetközi unicode karakterekhez, valamint támogatta az időzónákat a Dátum, az Idő és az Időbélyeg adattípusoknál.

- egy opcionális csomag (mely a `javax.sql` csomagban érhető el), ahol lehetőség lett

adatforrások megadására a JNDI-n (Java Naming and Directory Interface – Java név és katalógus szolgáltatás) keresztül, elosztott adatbázis-tranzakciók végrehajtására, vagyis egy lekérdezésen belül több adatbázisszerver használatára, valamint megvalósult a kapcsolatok újrafelhasználása, illetve a saját eredménytáblák is megjelentek

JDBC 3.0 – A JDK 1.4-es verziójában jelent meg, az alábbi újításokkal az előző változathoz képest:

- Mentési pontok támogatása: a `Savepoint` interfész segítségével lehetőség van mentési pontok létrehozására, törlésére, illetve a tranzakciók mentési pontig történő visszagörgetésére
- Az előfordított utasításokhoz (`PreparedStatment`) tartozó objektumok is újrafelhasználhatóvá váltak újrafelhasználható kapcsolatok esetén
- Az újrafelhasználható kapcsolatok konfigurálhatók lettek, a tulajdonságok beállításával megadható, hogy milyen kapcsolatok használhatók az adatforrásokból
- A `PreparedStatement`-ek paramétereinek száma, típusa lekérdezhetővé vált az új `ParameterMetaData` interfész segítségével
- Automatikusan generált oszlopértékek lekérdezhetőek lettek
- Több megnyitott eredményhalmaz (`ResultSet`) használata
- Kurzorok tárolásának lehetősége
- `BOOLEAN` adattípus támogatása (logikailag ekvivalensen a `BIT`tel)
- `BLOB, CLOB, ARRAY, REF` adattípusok frissítése (`REF` esetén a lekérdezhetőség is új)
- `DATALINK/URL` adattípus támogatása

JDBC 4.0 – 2006-ban jelent meg a JDK 1.5 részeként, az alábbi újításokkal az előző változathoz képest:

- A `java.sql.Driver` automatikus betöltése: a

`DriverManager.getConnection` módosításával elérték, hogy ne kelljen meghívni a `Class.forName` metódust

- `ROWID` adattípus támogatása
- Nemzeti karakterekre konvertálás lehetősége
- Javított `BLOB` és `CLOB` támogatás
- `SQL/XML` és `XML` támogatás
- Lehetővé vált a JDBC osztályok kibontása a külső fejlesztőktől származó, nem sztenderd metódusok használatához
- Továbbfejlesztett kivételkezelés az `SQLException` osztályban
- Javított kapcsolat kezelés
- Különböző változtatások a JDBC API-ban, melyek még könnyebbé teszik az adatbázisok kezelését

2.2.3. JDBC és a Java

Az első, és legfontosabb tudnivaló a JDBC használatához, hogy minden szükséges osztály két Java csomagban található a JDBC 2.0 óta: a `java.sql`-ben, mely az alap APIt tartalmazza, a `javax.sql` csomagban, mely az opcionális csomagot testesíti meg.

Az adatbázissal való kommunikáció folyamatának lépései:

1. lépés Betölteni az adatbázishoz tartozó meghajtóprogramot, vagyis drivert. Ez a művelet a `Class.forName(String)` utasítással történhet, ahol a `String` a driver elérési útját tartalmazza. A népszerűbb adatbáziskezelő-rendszerekhez elérhetőek meghajtóprogramok különböző fejlesztőktől, melyeket az alábbi honlapról lehet bármikor letölteni: http://developers.sun.com/product/jdbc/drivers

2. lépés A meghajtóprogram betöltése után következik a kapcsolódás az adatbázishoz. Erre 2 különböző mód van és mindkettőnek meg van a maga előnye:

    - A `DriverManager` osztály használata: ez a mód lényegesen egyszerűbb,

mint a másik. A `DriverManager` osztály segítségével URL-lel megadott drivert tölthetünk be, sőt inicializálásakor automatikusan megpróbálja végrehajtani a `jdbc.driver` rendszerpropertyben szereplő drivereket.

A kapcsolat kialakításához szükséges URL felépítése a következő: `jdbc:<DBMS>:<adatbázisnév>[property_lista]`

A kapcsolódáshoz egy `Connection` objektum létrehozása szükséges:

```
Connection con = DriverManager.getConnection(String)
```

A fenti utasításban a `String` az URL. Amennyiben az adatbázishoz való csatlakozáshoz szükséges felhasználói név és jelszó, ezeket is át kell adni a metódusnak újabb `String` objektumokként.

- A `DataSource` interfész használata: ennek alkalmazása sokkal bonyolultabb, azonban jelentősen megnöveli a program hordozhatóságát. A `DataSource` objektum konfigurálása történhet automatikusan egy eszköz segítségével, vagy akár manuálisan is, ekkor a programkódba beépített utasításokkal állíthatjuk be az egyes tulajdonságokat.

3. lépés A kapcsolódás megtörténte után van lehetőség a különböző DDL és DML utasítások készítésére, futtatására.

Tábla létrehozás

Táblák létrehozásához (már amennyiben jogosultságunk van rá), nem kell mást tennünk, mint a `Connection` objektumon először meghívni a `createSatetement()` metódust, s az így létrejövő `Statement` objektum segítségével már ténylegesen futtathatjuk az SQL utasításunkat. Ez a művelet a továbbiakban is szükséges az utasítások futtatásához.

Íme egy egyszerű példa, ahol a `conn` objektum `Connection` típusú:

```
Statement stmt = conn.createStatement();
stmt.executeUpdate(„CREATE TABLE...”);
```

A példában az SQL utasítás nem teljes, azt ki kell kiegészíteni, hogy szintaktikailag

helyes legyen. Meg kell adni a tábla nevét, majd zárójelek között fel kell sorolni az egyes mezők nevét, illetve azok típusát.

Frissítő lekérdezések készítése

Egy frissítő lekérdezés készítése és futtatása, legyen az INSERT, DELETE, vagy esetleg UPDATE utasítás, elvégezhető ugyanazzal a módszerrel, mint a táblák létrehozása, azonban előfordulhat, hogy nagyon hasonló utasításokat szeretnénk sokszor végrehajtani. Ekkor újabb és újabb SQL utasítások fordítása rendkívül processzorigényes művelet. Ilyen helyzetben ajánlott a `Statment` helyett a `PreparedStatment` osztály használata, mely – mint a neve is mutatja –, előkészített, előre lefordított utasítást takar.

`PreparedStatment` objektum létrehozásakor nem a teljes SQL parancsot adjuk meg, hanem annak csak a vázát. A specifikus részeket, mint például egy WHERE feltételben egy egyenlőség vizsgálatnál a kérdéses értéket, csak egy kérdőjellel szimbolizáljuk, s ezeket a paramétereket később adjuk meg, minden változtatás esetén tetszőlegesen cserélhetjük. A `PreparedStatement` egy úgynevezett előfordításon esik át, a váza lefordul, s így az egyes futtatásakor az SQL fordítómotornak csak a paramétereket kell hozzátennie.

Íme egy egyszerűbb példa:

```
PreparedStatement updateSalary = con.prepareStatement(
"UPDATE employee SET salary = ?*salary WHERE employee_name
like ? ");
updateSalary.setFloat(1, 1.1);
updateSalary.setString(2, "Kovács Dezső");
updateSalary.executeUpdate():
```

Amint a példán is látható, az egyes paramétereket a `PreparedStatment` `setXXX()` metódusaival állíthatjuk be, ahol az XXX helyére a megfelelő típust kell írnunk. Ezek a metódusok két paramétert várnak: egy egész számot, mely azt jelzi, hogy hányadik paramétert szeretnénk beállítani, illetve egy, a jelzett típussal egyező típusú változót.

Megjegyzés: Amint az imént kiderült, `PreparedStatement`-ekkel dolgozni

egyszerűbb lekérdezések esetén lényegesen bonyolultabb, de amikor több fajta típusú paraméter előfordulhat, vagy sok hasonló lekérdezés futtatása történik, kifizetődő. Mivel programom elsősorban kis- és középvállalkozások számára készült, ahol az újonnan érkező számlák és megrendelések száma nem túl magas, nem tartottam célszerűnek használatát.

Választó lekérdezések készítése

Egy választó, vagy másképp SELECT lekérdezés készítéséhez a táblakészítésnél ismertetett módon először egy `Statement` objektumot kell létrehoznunk, azonban a lekérdezés futtatásához már az `executeQuery()` metódusát kell meghívnunk, mely az `executeUpdate()`-hez hasonlóan egy `String` paramétert kap, melynek szintaktikailag helyes SELECT utasításnak kell lennie. A metódus a lekérdezés eredményét egy `ResultSet` objektumban adja vissza.

A `ResultSet` bejárása alapesetben a `ResultSet next()` metódusával történhet, amely az eredményhalmazban mindig a következő sorra ugrik és amíg ez sikeres, igaz értéket ad vissza. Amennyiben elértük az eredményhalmaz végét, az újabb `next()` metódus hívásra az utolsó sor utáni helyre ugrik és hamis értékkel tér vissza, ily módon iterátorként használható a halmaz bejárására.

A `createStatement()` metódusnak létezik két paraméteres változata is, melyek segítségével szabályozható hogy a későbbi eredmények feldolgozása hogyan történhet, valamint hogy a `ResultSet` eredményei módosíthatóak-e. Ha igen, amint módosításra kerülnek, az adatbázisban is átíródnak az értékek.

Az első paraméter három különböző értéket vehet fel:

- `TYPE_FORWARD_ONLY`: A feldolgozás csak előrefelé történhet (alapértelmezett mód)
- `TYPE_SCROLL_INSENSITIVE`: Visszafelé is történhet a feldolgozás, lehet ugrani a sorok között relatív és abszolút pontra is
- `TYPE_SCROLL_SENSITIVE`: Az `INSENSITIVE`-vel egyező

tulajdonságokkal rendelkező mód, azonban ennél a `ResultSet` automatikusan követi az adatbázisban időközben bekövetkező változásokat

A második paraméter két értéket vehet fel:

- `CONCUR_READ_ONLY`: Az eredmények csak olvashatók (alapértelmezett mód)
- `CONCUR_UPDATABLE`: Az eredmények módosíthatóak is, s ezek a módosítások rögzítésre kerülnek az adatbázisban is

Az egyes sorokból a megfelelő adatokat a `getXXX()` metódusok segítségével nyerhetjük ki, ahol az XXX a megfelelő adattípust jelöli. Minden ilyen metódusnak két fajtája létezik. Ezek az átadott paraméter típusában különböznek: egyik fajtája egy `String`et vár, mely a megfelelő mező neve, míg a másik egy egész számot, mely a kívánt oszlop indexét jelenti. A második használata ajánlatosabb, ugyanis ezáltal nagyobb rugalmasságot biztosítunk a programunknak, hiszen a táblákban a mezők nevei változhatnak, illetve az egyes adatbázis-kezelő rendszerekben más-más elnevezési konvenciók érvényesek.

Mind a `Statement`, mind a `ResultSet` objektumokat használatuk után le kell zárnunk az objektumok `close()` metódusával, hogy a szemétgyűjtő felszabadíthassa a nekik lefoglalt memóriát, és lezárjuk az adatbázissal a kapcsolatot. Ajánlatos a `close()` metódus hívását a kivételkezelő `finally` ágában elhelyezni, ily módon biztosan le fog futni.[9]

Tárolt eljárások hívása

Említést kell tennem még a `Statement` és a `PreparedStatement` objektumok után egy harmadik módról, mellyel a tárolt PL/SQL eljárásokat hívhatjuk meg, ez pedig a `CallableStatement`. Ez tartalmazhat mind kimenő, mind bejövő paramétereket, melyeket megadásukkor a `PreparedStatement`hez hasonlóan kérdőjelekkel szimbolizálhatunk. Maga az eljárást az adatbázis tartalmazza, itt csak meghívjuk.

9 Joshua Bloch - Effective Java

Futtatása az `execute()` metódussal történhet, mely az `executeQuery()` általánosításának tekinthető. Használata két esetben javallott: ha többfajta eredményt várunk, vagy nem tudjunk, hogy az eredmény milyen típusú. Ha eredményhalmaz, akkor a `getResultSet()`-tel kérhetjük le az eredménykomponenst, egyéb esetben pedig a `getMoreResult()` metódust kell használnunk.

A változtatások véglegesítése, visszagörgetése

Alapértelmezés szerint minden DML utasítás (INSERT, DELETE, UPDATE) véglegesítődik lefutása után. Ezt hívjuk autocommitnak. Ez a funkció letiltható a `Connection` objektumon meghívott `setAutoCommit(false)` metódussal.

Amennyiben az autocommitot letiltottuk, explicit módon kell véglegesítenünk a változtatásokat. Emellett lehetőségünk van mentési pontok (SAVEPOINT) létrehozására, melyekhez a `rollback()` metódussal görgethetünk vissza – ezek hiányában a legelejéig, vagy az utolsó véglegesítésig görgetődnek vissza a tranzakciók. A kapcsolat lezárásakor egy implicit COMMIT is végrehajtódik, véglegesítve a még nem véglegesített módosításokat.

Végül a már nem használt kapcsolatot mindig zárjuk le, mert ekkor minden, a kapcsolathoz lefoglalt erőforrás felszabadításra kerül.

2.3. Adatbázis-kezelő rendszerek

2.3.1. Apache Derby

Az Apache Derby az Apache DB egyik alprojektje, egy teljes mértékben Java nyelven írt, nyílt forráskódú, így ingyenes Relációs Adatbázis-kezelő Rendszer (RDBMS), melyet az Apache License 2.0-s verziója véd, s a teljes program letölthető innen: http://db.apache.org/derby/derby_downloads.html

Installálás előtt 2 fajta futási beállítás közül választhatunk:

- *Beépített (embedded)*, mely során a Derby ugyanazon a JVM-en fut, mint a szoftver amely használja, ily módon majdhogynem láthatatlan a végfelhasználók számára, hiszen egyszerre indul és áll le a programmal, valamint általában nem igényel autentikációt. Csak egyedi csatlakozást engedélyez.
- *Szerver (Server-based)*, mely többfelhasználós módot engedélyez hálózaton keresztül. A Derby JVM-je a szerveren fut, s az alkalmazások különböző JVM-ekből csatlakozhatnak hozzá, hogy elérjék az adatbázist. A Derby hálózati szerver a szoftvercsomag része, s kiválóan együtt tud működni egyéb fejlesztésű szerveroldali alkalmazásokkal is.

A Derby DB futásához a gépen telepítve lennie kell minimum 1.4.2-es, vagy magasabb verziójú JDK-nak, majd a telepítés következő lépéseként a letöltött .zip fájlt ki kell csomagolnunk. Legyen mondjuk a kicsomagolás célkönyvtára a C:\Apache Windows alatt (a továbbiakban is csupán a Windows-os telepítési lépéseket ismertetem, azonban a szoftver honlapján megtalálható a Unix alapú operációs rendszerek alatti teendők részletes leírása is).

A kitömörítés után környezeti változók beállítása szükséges. A parancssorban a következő parancsok kiadása szükséges (a példa során a kicsomagolt könyvtár elérési útja a C:\Apache\Derby, amennyiben máshová történt a kitömörítés, ez a része a parancsnak, értelemszerűen változik)[10]:

```
set DERBY_HOME=C:\Apache\Derby
```

10 A Derby DB felhasználói kézikönyvéből - http://db.apache.org/derby/docs/dev/getstart/

```
set PATH=%DERBY_HOME%\bin;%PATH%.
set CLASSPATH=%DERBY_HOME%\lib\derby.jar;%DERBY_HOME
  %\lib\derbytools.jar;.
```

Ha minden lépést megtettünk, ezek után lehetőségünk van a következő különböző eszközök használatára (a teljesség igénye nélkül):

- *sysinfo* – egy eszköz, mely adatokat szolgáltat a Java környezetről, valamint a használt Derby változatról
- *ij* – egy JDBC eszköz, mellyel szkripteket, lekérdezéseket futtathatunk az adatbázison
- *dblook* – olyan eszköz, mellyel a meglévő adatbázishoz, vagy annak bizonyos részeinek újra létrehozásához generálhatunk DDL utasításokat, melyeket akár a képernyőre, akár rögvest egy fájlba írathatunk ki

Ezek után már lehetőségünk van újabb adatbázisok létrehozására, illetve azok konfigurálására, azokon lekérdezések futtatására, vagyis a telepítéssel készen vagyunk, használatba vehetjük a DBMS-t.

Adatbázis létrehozása:

```
ij> connect 'jdbc:derby:Adatbazis_nev;create=true';
```

Adatbázishoz kapcsolódás: `ij> connect 'jdbc:derby:Adatbazis_nev';`

Amennyiben probléma merül fel bármilyen téren, a szoftver honlapján minden kérdésünkre választ kaphatunk: részletes leírások, oktatóanyagok, példák találhatóak.

2.3.2. MySQL

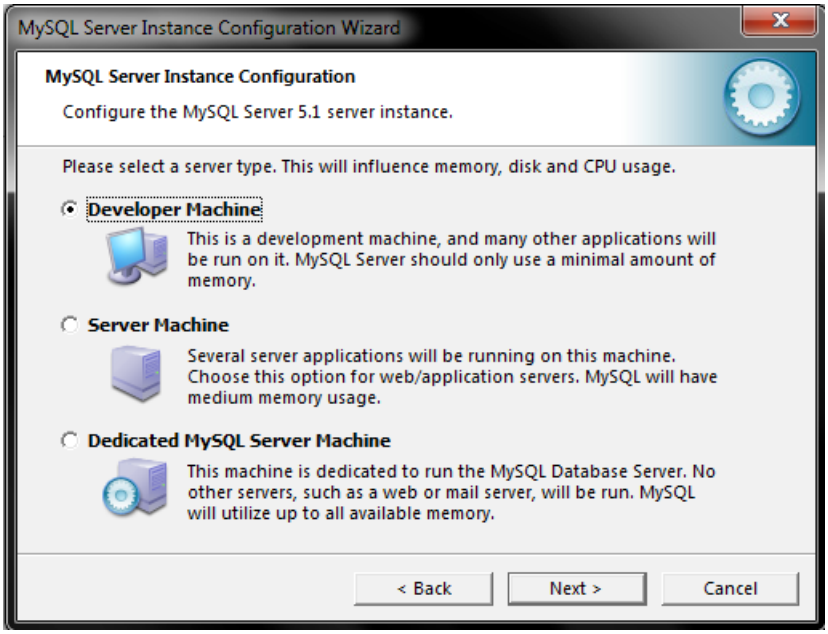
A MySQL a legnépszerűbb nyílt forráskódú adatbázis-kezelő rendszer, melyet eredetileg a MySQL AB nevű cég fejlesztett, akit aztán felvásárolt a Sun Microsystems. Mára a Sun-t felvásárolta az Oracle Corporation, így a céggel a MySql is az Oracle tulajdonába került.

A szoftver első változata 1995. májusában jelent meg, mára az 5.5-ös verziónál tart. A fejlesztéseknek hála a legfejlettebb DBMS-ek közé tartozik: gyors, megbízható és funkciókban gazdag.

Innen tölthetjük le a megfelelő operációs rendszer kiválasztása után (a leírásban, a Derby DB-hez hasonlóan csak a Windowsos változat telepítésének lépéseit írom le[11], de a kézikönyvben részletesen megtalálható az egyéb rendszerekre történő installálás is): http://www.mysql.com/downloads/mysql/. Ajánlatos az MSI kiterjesztésű fájlt letölteni, hiszen a telepítés lényegesen egyszerűbbé válik: a Microsoft Installer szolgáltatásait kihasználva, a telepítő végigvezet minden lépésen: kiválaszthatjuk a telepítés helyét és módját.

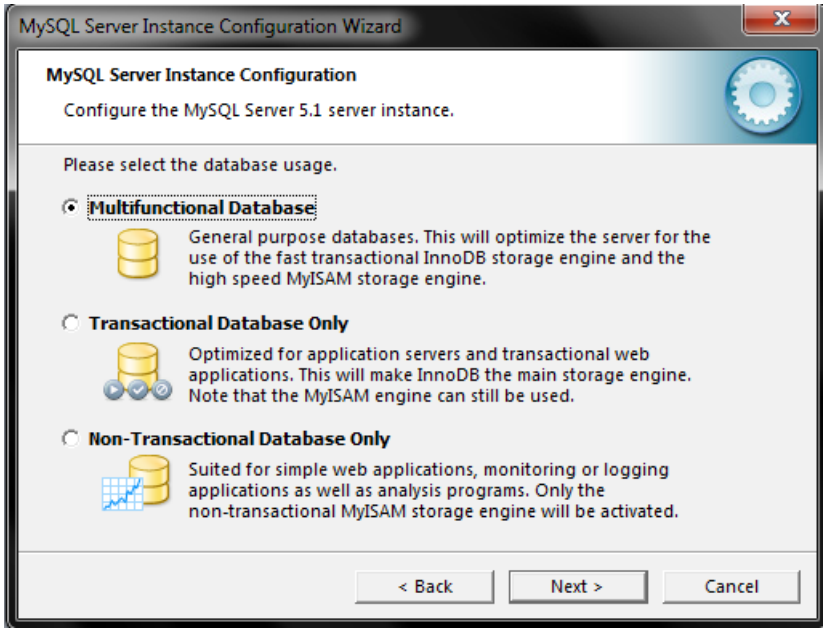
Amennyiben az egyéni telepítést választjuk, be tudjuk állítani, hogy a MySQL szerveren kívül milyen egyéb segédprogramok települjenek gépünkre, valamint a teljes dokumentációra is szert tehetünk. Amennyiben ezeket feleslegesnek érezzük, úgy telepítésüket mellőzzük, mindazonáltal hasznosnak bizonyulhatnak, különösen a segédprogramok:

Miután a telepítés befejeződött, következhet a szerver konfigurálása. Bár ezt később is megtehetjük, javasolt most rögtön megtenni.

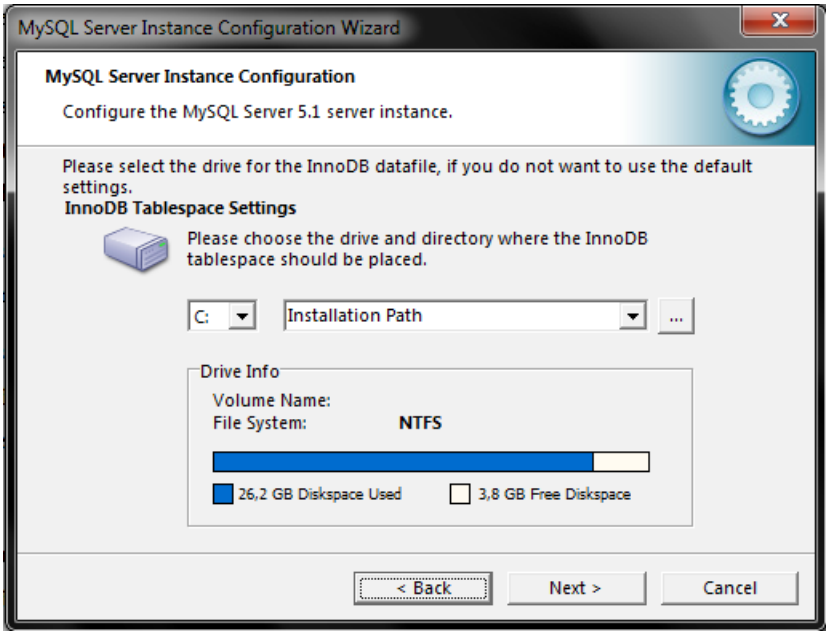


Első lépés kiválasztani, hogy milyen gépen történik a szerver futtatása, vagyis hogy csak szerverként funkcionál, vagy egyéb célokra is használva van-e, ettől függ ugyanis a memóriafogyasztása (és természetesen a sebessége is).

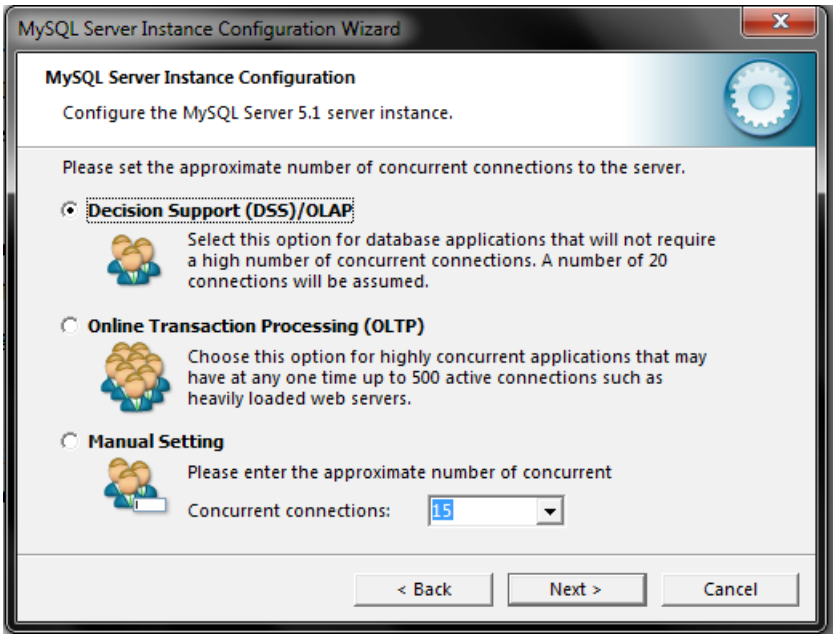
11 A hivatalos MySQL kézikönyv alapján



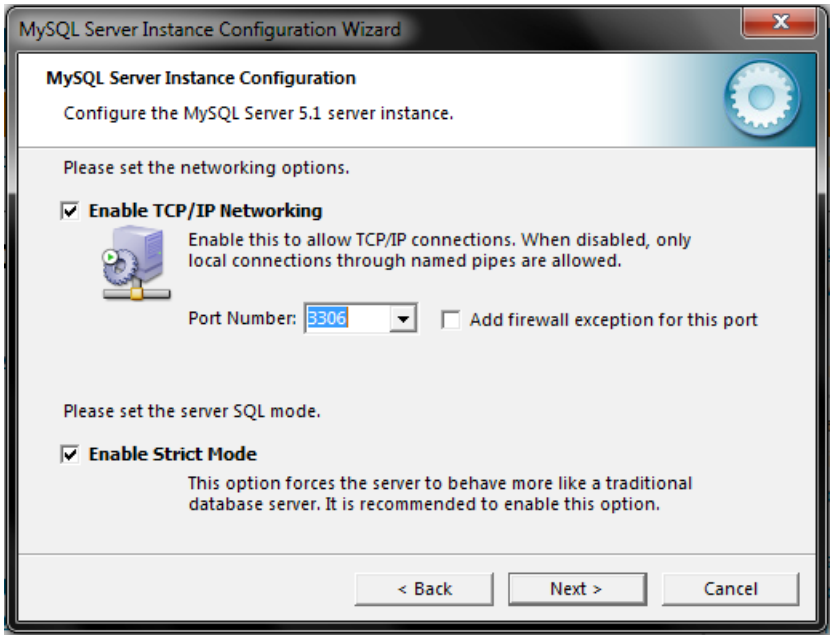
A következő képernyőn azt választhatjuk ki, hogy milyen célokra kívánjuk használni az adatbázist: fontos adatbázis műveletekhez, csak monitorozásra, naplózásra, esetleg mindkettőre.



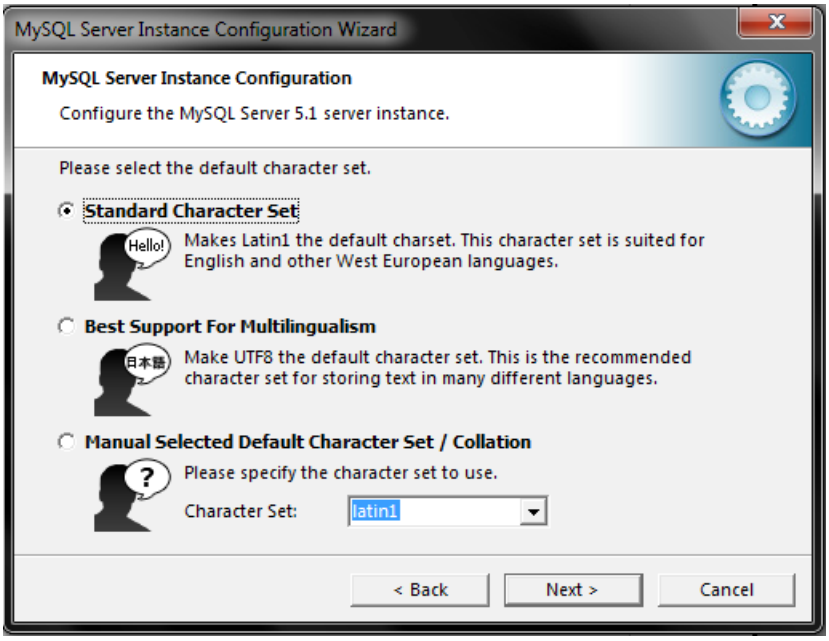
Ezután választhatjuk ki, hogy az adatbázisokat melyik meghajtón tárolja a program, s megadhatjuk pontos elérési útjukat: ez határozza meg, az adatbázis maximális méretét.



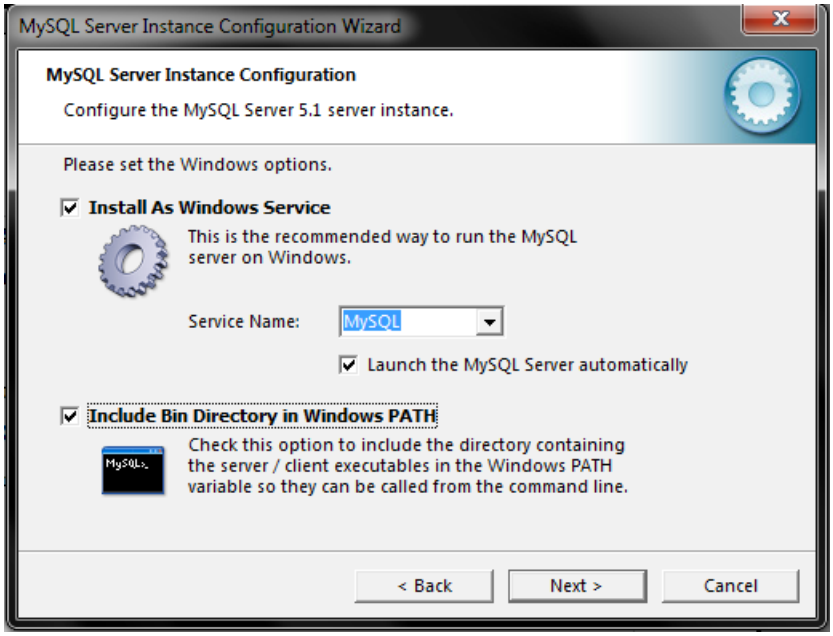
A következő panelen a szerverhez egyszerre maximálisan kapcsolódó felhasználók száma állítható be,(az előre definiáltakon kívül egyéni értéket is megadhatunk).



Ezek után beállíthatjuk, hogy TCP/IP protokollon keresztül elérhető legyen-e a szerver, s ha igen megadhatjuk a hozzá kapcsolódó port számot, valamint beállíthatjuk, hogy külső szemlélő felé a hagyományos adatbázis-szerver látszatát keltse-e a program.



Majd megadhatjuk, hogy milyen karakterkészletet kezeljen az adatbázis. Javasolt az UTF8 választása esetünkben, így biztosak lehetünk benne, hogy semmilyen magyar karakter kezelése nem okoz problémát az adatbázisnak.



Ezek után a MySQL futtatását szabályozhatjuk, s végül a felhasználói adminisztrációt szabhatjuk testre.

Ezek után az adatbázis műveletekhez nem kell mást tennünk, mint futtatni a MySQL

Command Line Client nevű segédprogramot, majd a megfelelő parancsok begépelésével egyszerűen elvégezhetünk bármit:

- Adatbázis létrehozása: `CREATE DATABASE adatbazis_nev;`
- Adatbázis használata: `USE adatbazis_nev`
- Ezek után bármilyen lekérdezést (megfelelő SQL szintaktikával) lefuttathatunk

Természetesen létezik számos, grafikus felületű kliens MySQL adatbázisok karbantartásához, de ez a kis segédprogram is képes bármire amire azok. Az ingyenesek közül népszerű SQLyog (melynek létezik fizetős változata is több funkcióval) és a HeidiSQL, fizetősek közül pedig kedvelt az SQLWave. Az adatbázisok tervezéséhez a MySQL gyártója is kínál egy ingyenes, grafikus felülettel is rendelkező segédprogramot, ez a MySQL Workbench.

3. Referencia-kézikönyv, a fejlesztés folyamata

3.1. Követelmények összegyűjtése

Minden szoftverfejlesztés folyamatának első lépcsőfokán a követelmények összegyűjtése áll, vagyis amikor a megrendelő és a fejlesztő tisztázza, hogy a szoftvernek milyen elvárásoknak kell megfelelnie, milyen funkciókat kell tudnia, s úgy egyáltalán, hogy a szoftver milyen probléma megoldására használható.

1. Az első és legfontosabb követelmény, hogy a szoftver használata teljes mértékben helyettesítse a papír alapú számlázást, vagyis minden – mind az eladóra, mind a vevőre, mind pedig az érintett termékre/szolgáltatásra vonatkozó – adat tárolásra kerüljön! Az alapvető felhasználói műveleteket az 1. számú mellékletben található használati eset diagram szemlélteti.
2. Elengedhetetlen, hogy a rögzített számlákat ki lehessen nyomtatni, azok formailag és tartalmilag megfeleljenek a magyar jogszabályi előírásoknak: fel legyen tüntetve rajtuk a számlát kibocsájtó cég neve, telephelye, adó- és számlaszáma, a vásárló neve, telephelye, valamint az egyes tételekről soronként szerepeljen annak megnevezése, a terméket/szolgáltatást sújtó ÁFA mértéke, a termék/szolgáltatás mennyiségi egysége, mennyisége, nettó egységára és a nettó összege, végül a számla legalján szerepeljen a számla nettó és bruttó végösszege, valamint a teljes ÁFA tartalom!
3. A szoftver az adatokat egy bárhonnan elérhető adatbázisszerveren tárolja!
4. A szoftver legyen platformfüggetlen!
5. A rögzített adatoknál, legyen mód azok semmissé tételére (rontottként való megjelölésére), de törlésére ne!
6. A szoftver használata legyen egyszerű, könnyen elsajátítható, felülete egyszerű, átlátható, egyértelmű!
7. A szoftver legyen képes diagramon ábrázolni a bevételek, kiadások, valamint a szabad készpénz alakulását, a diagramot ki lehessen nyomtatni!

3.2. Adatbázis tervezés

Mivel a követelmények között az egyik kitételként az szerepel, hogy a program a rögzített adatokat egy adatbázisban tárolja, meg kell tervezni a majdan használt adatbázis sémáját, a felhasznált táblákat, azok mezőit, illetve a mezők típusait is.

Mivel egy adatbázisban akár több cég adatai is tárolásra kerülhetnek, nem lehet egyértelműen meghatározott neve az egyes tábláknak. Emiatt a táblák neveit a rendszer automatikusan generálja az adott vállalat neve alapján, így az alábbi táblák születtek:

- CompanyName – A cég nevét viselő tábla, ebbe kerülnek a vállalathoz beérkező, illetve a vállalat által kiadott számlák. Elsődleges kulcsaként a minden számla esetén egyedi sorszám szerepel
- CompanyName_Rows – Mivel egy-egy számlához több sor is tartozhat, az egyes számlákban lévő tételeket célszerű új táblában tárolni. Erre hivatott ez a tábla, mely összetett elsődleges kulccsal rendelkezik: minden tétel esetén egyedi lesz a számla sorszáma, illetve számlán elfoglalt helye
- CompanyName_Partners – A vállalattal kapcsolatban álló más cégek adatait tartalmazó tábla. Minden partner egyedi azonosítóval rendelkezik, így ez a mező alkalmas lesz az elsődleges kulcs szerepének betöltésére

A 2. számú mellékletben lévő adatbázissémán megtalálhatóak az egyes táblák a mezők típusaival, részletesebben csak a rendhagyónak mondható típusválasztásokra térek ki.

Az említett, rendhagyó választások oka abból fakad, hogy a szélesebb körű használhatóság érdekében több adatbáziskezelő-rendszert is támogat a szoftver, azonban ezek nem mindenben kompatibilisek, sőt még az adattípusokban sem. A fő problémát azon mezők okozzák, melyek típusának értelemszerűen a `BOOLEAN`, vagyis a logikai adattípust adnánk. Míg a MySQL gond nélkül kezeli az ilyen mezőket, sajnos az Apache Derby DB-ben már nem került megvalósításra, így emiatt arra kényszerültem fejlesztés során, hogy `BOOLEAN` típus helyett inkább a C programozási nyelvhez hasonló, ám attól egy kicsit eltérő megoldást válasszak, nevezetesen ezen mezők `INTEGER` adattípusúak lettek, ahol a 0 érték jelenti a hamis, az 1-es pedig az igaz értéket. Az említett mezők a CompanyName táblában találhatóak:

- *Type (Típus)* – eredetileg a neve az Income (Bevétel) lett volna, azonban a típusváltozás miatt, a mező is átnevezésre került, hogy jobban szemléltesse funkcióját
- *Wasted (Rontott)* – egy egyszerű jelölőmező, mely a rontottként megjelölt számlák esetében rendelkezik igaz értékkel

Amint az általában lenni szokott és a sémán is megfigyelhető, a táblák között különféle kapcsolatok vannak. Ezek a kapcsolatok külső kulcsok, vagyis szabályozzák a táblákba felvihető adatokat:

- *CompanyName_Partners – CompanyName*: a partnereket tartalmazó tábla ID oszlopának értékei a CompanyName tábla Partner_ID oszlopának külső kulcsai. Ez a kapcsolat biztosítja, hogy csak olyan partnerazonosítójú számlák kerülhessenek fel, mely azonosítóhoz már tartozik partner
- *CompanyName – CompanyName_Rows*: a CompanyName tábla Serial oszlopának értékei jelennek meg a CompanyName_Rows tábla Serial oszlopának külső kulcsaiként, ily módon biztosítva, hogy nem létező számlához ne kerülhessenek fel tételsorok.

3.3. A grafikus felhasználói felület és tervezése

A felhasználó szempontjából a szoftver egyik legfontosabb része a felhasználói felület.

A legtöbb esetben a program teljes működése igényli a folyamatos felhasználói közbeavatkozást, interakciót, ebben az esetben ajánlott valamilyen felhasználói felület kialakítása.

A felhasználói felületen (User Interface) keresztül irányíthatja a felhasználó a program működését, adatokat vihet fel, illetve a program által közölt információkat, műveleti eredményeket olvashatja le. Ezek megvalósíthatóak karakteres felületen is, azonban a modernebb programok grafikus felhasználói felülettel rendelkeznek (GUI – Graphical User Interface).

A grafikus felhasználói felület egyik legnagyobb előnye a karakteressel szemben, hogy az emberi agy sokkal könnyebben jegyez meg grafikákat, jól megkülönböztethető ábrákat,

szimbólumokat, ezáltal a szoftver használatának megtanulása is jelentősen leegyszerűsödik. A GUI kényelmesebb használatot biztosít és tetszetősebb is, igaz megtervezésénél figyelni kell arra, hogy ne legyen túl színes (aranyszabály, hogy maximum hét különböző színt használjunk és mindnek legyen funkciója is), valamint szem előtt kell tartani az országok közötti kulturális különbségeket (pl.: míg a legtöbb európai országban a gyász színe a fekete, addig Egyiptomban és Burmában a sárga, Szíriában és Iránban pedig a kék)[12].

Az objektumorientált programozási nyelveknél a grafikus komponenseket is objektumok reprezentálják. A komponenseket három különböző csoportra oszthatjuk, az alábbiak szerint:

- beviteli komponensek – pl.: textfield, textarea stb...
- vezérlő komponensek – pl.: button stb...
- megjelenítő komponensek – pl.: label, panel stb...

A Java grafikus komponensei két csomagban találhatóak:

- `java.awt` – a csomag által tartalmazott komponensek az adott ablakkezelő-rendszernek megfelelően rajzolódnak ki
- `javax.swing` – a csomag által tartalmazott komponensek minden rendszeren ugyanúgy jelennek meg

3.3.1. A felületek tervezése

A felületek összeállításánál szem előtt kell tartani a formai és tartalmi követelményeket. Mivel egy mindennapos használatra tervezett programról van szó, felületének összeállításakor a praktikumnak kell dominálnia a díszesség felett, ám nem hátrány, ha a külseje nem csúnya.

A Java által biztosított Swing csomag komponenseinek külseje elfogadható kinézetű, jól megkülönböztethetőek, így a csupán ezekből összeállított felhasználói felület megfelelő lesz egy számlázó szoftverhez.

A grafikus felületek összeállítása csupán programkód használatával rendkívül nehézkes feladat, de nem lehetetlen. A Java különböző, úgynevezett elrendezési eszközöket (Layout managereket) ad, mely alapján felépülnek a felületek. Azonban, főleg a bonyolultabb felületek

12 http://www.geographic.hu/index.php?act=napi&id=5971

összeállításánál jelentősen meggyorsítja a folyamatot valamilyen automatikus eszköz, melynek segítségével egyszerűen, egérmozdulatokkal összeállíthatjuk a felületet. A NetBeans IDE rendelkezik ilyen funkcióval, így én is ennek segítségével állítottam össze őket.

A program főablaka valóban ablakként (Frame-ként) jelenik meg, míg minden más ablak párbeszédpanelként, vagyis a `JDialog` osztály leszármazottjaként. A program felhasználói felületén az alábbi komponensek jelennek meg (ezek mellett szerepelnek még nem látható, úgynevezett konténer komponensek, melyek az alábbiak tárolására, csoportosítására szolgálnak):

- *Címke (JLabel)* – Előre rögzített adatok, illetve a program közléseinek megjelenítésére szolgáló eszköz. Tartalmazhat egyaránt szöveges, vagy képi adatot is.
- *Szövegmező (TextField)* – A leggyakoribb adatbevitelt lehetővé tévő komponens, mely 1 soros szöveg bevitelére (vagy megjelenítésére) alkalmas.
- *Legördülő lista (ComboBox)* – Ezen komponens szintén adatbevitelt szolgál, s mint a neve is mutatja, a legördülő elemek közül választhatunk ki egyet.
- *Rádiógombok (RadioButton)* – Adatbevitelt szolgáló eszköz, melynek lényege, hogy a rádiógombok csoportjából csupán egyet választhatunk ki.
- *Jelölőnégyzet (CheckBox)* – A rádiógombhoz hasonló komponens, azonban a csoportból itt többet is kiválaszthatunk.
- *Gomb (JButton)* – Az egyik legfontosabb vezérlőeszköz, mely megkülönböztetésére használható szöveg és kép, ennek lenyomásával elvethetünk vagy véglegesíthetünk adatokat.

A program által felhasznált grafikus elemek után következzék egy rövid áttekintés az egyes ablakokról, azok funkcióiról. Az ablakok kinézete a felhasználói kézikönyvbe helyezett képernyőképeken láthatóak, a szoftver két legfontosabb tevékenységét – a szoftver használatának megkezdését, valamint a tételek rögzítését – leíró aktivitás diagramok pedig megtalálhatóak a mellékletek között.

A főablak – A program főablaka két nagy részből áll össze, ez egy `JPanel`, mely a vezérlőgombokat tartalmazza, valamint az ablak felső részén egy menürendszert, ahonnan a

program indításának fő mozzanatait vezérelhetjük.

Adatbázis beállítások – Egy párbeszédpanelként jelenik meg, melynek segítségével az adatbázis-kapcsolathoz szükséges adatokat vihetjük fel a rendszerbe.

Új tétel felvitele – A párbeszédpanelen a számlák adatait adhatjuk meg: az számla fő adatait, valamint az egyes tételsorok tartalmát is.

Új partner felvitele – A párbeszédpanelen az új partnerek rögzítésére nyílik lehetőségünk, ezek adatait vihetjük fel.

Tételek keresése – A párbeszédpanelen a különböző megadott paraméterek alapján kereshetünk a korábban rögzített számlák között.

Eredménylista – A keresés eredményének megjelenítésére szolgáló ablak. Innen nyílik lehetőség a nyomtatásra, illetve a számlák rontottként való megjelölésére.

Számla – A képernyőn nem megjelenő ablak, melynek tartalma csupán nyomtatásban látható. Az ablak egy számla adatait tartalmazza.

Kimutatások készítése – A párbeszédpanelen a különböző megadott paraméterek alapján megszűrt számlák felhasználásával készíthetünk oszlopdiagramokat, s ezeket nyomtathatjuk is.

3.4. Referencia-kézikönyv

Referencia-kézikönyv egy program dokumentációjának az a része, mely a programozó szempontjából közelíti meg a rendszert (szemben a felhasználói kézikönyvvel, mely egy, a szoftvert használó átlagos user szempontjából nyújt leírást a rendszerről).

Egy jól használható referencia-kézikönyv megírása rendkívül időigényes és sok munkát, odafigyelést igénylő munka. Részletes áttekintést kell adjon a programkódban szereplő csomagok, osztályok funkcióiról, az osztályokon belül található metódusok működéséről.

Sok más mellett, a Java előnyei közé tartozik, hogy szolgáltat egyfajta keretrendszert, mellyel a referencia-kézikönyvhöz hasonló dokumentáció megírása jelentősen leegyszerűsödik: ez a Javadoc.

A Javadoc használata rendkívül egyszerű. A keretrendszer a programkódban található dokumentációs megjegyzésekből (ezek a /** és a */ között találhatóak), HTML oldalakat generál. Ezek felépítése tükrözi a csomagszerkezetet és az osztályhierarchiát. Automatikusan generál egy faszerkezetet az osztályhierarchia alapján, melyen elérhetőek az egyes osztályokhoz tartozó HTML aloldalak.

Minden osztály aloldala az alábbi módon épül fel:

- Fastruktúra, mely az osztály hierarchiában való helyét mutatja
- Az implementált interfészek, majd az osztályból származó osztályok listája (mindegyik oldalához link, ha elérhető)
- Rövid vagy hosszabb leírás az osztályról
- Beágyazott osztályok/interfészek összefoglaló listája aszerint bontva, hogy honnan örökölte
- Konstansok összefoglaló listája aszerint bontva, hogy honnan örökölte
- Az elérhető konstruktorok összefoglaló listája, rövid leírással
- A látható (tehát nem privát) metódusok összefoglaló listája visszatérési típussal, teljes paraméterlistával és rövid leírással, majd az örökölt látható metódusok listája, aszerint

bontva, hogy honnan örökölte

Ezek után a konstruktorok és a metódusok részletes leírása következik, ahol újfent szerepel a rövid leírás, majd a paraméterek listája magyarázatokkal, s végül, visszatérési érték esetén, annak leírása.

Érdekesség, hogy a Java hivatalos APIját is Javadoc-kal készítették, így az általunk generált HTML oldalak felépítése is ezzel azonos lesz.

Most következik egy rövid áttekintő leírás a csomagok és osztályok funkcióiról, tulajdonságairól. A teljes dokumentáció megtalálható a szoftver mellett a CD lemezen.

3.4.1. Classes csomag

A csomagban a szoftver adatainak tárolására alkalmas osztályok találhatóak. A osztályok nevei tükrözik az osztály funkcióját.

Company osztály – A `Company` osztály egy vállalatnak, a program működése szempontjából alapvető adatainak tárolására való. A osztály implementálja a `Serializable` interfészt, mely révén majd szerializálásra kerülhet, így a felhasználó gépén kerülnek tárolásra végül az adatok. A létrehozott állomány neve megegyezik a vállalat nevével, kiterjesztése .ceg lesz.

Entry osztály – Az `Entry` osztály az egyes bejegyzések, számlák, adatainak tárolására szolgál

EntryRow osztály – Az `EntryRow` osztályban a számlák egy-egy soráról tárolhatunk adatokat.

Partner osztály – A `Partner` osztály segítségével a vállalat üzletfeleiről tárolhatunk információkat

Search osztály – A `Search` osztály a számlák közötti kereséshez használt beállításokat tárolja

Statistics osztály – A `Statistics` osztály segítségével azon információkat tárolhatjuk, melyek alapján a kimutatásokat készítjük.

3.4.2. Classes.Utilities csomag

A `Utilities` csomag, amint a neve is mutatja, hasznos osztályok, metódusok gyűjteménye, melyeket bármelyik osztály felhasználhat, de nem feltétlenül, emiatt nem emelhetők ki ősosztálynak. Ebben a csomagban kaptak helyet az olyan osztályok, mint pl. az `OpenDialog`-oknál használatos, saját definiálású szűrők, vagy az adatbázis-kapcsolatoknál használt SQL parancsokat tartalmazó osztály.

FileFilters osztály – A program működése közben használt fájlok kitallózásához szükséges szűrők szülőosztálya.

ImageFilter osztály – Az `ImageFilter` osztály segítségével egy szűrőt definiálunk, mely a megnyitás ablakban csak a program által képként elismert gif, jpeg, jpg, png és tiff kiterjesztésű fájlokat engedi látni.

SheetFilter osztály – A `SheetFilter` osztály segítségével egy szűrőt definiálunk, mely a megnyitás ablakban csak a program által, a vállalatok adatainak tárolására használt .ceg kiterjesztésű fájlokat engedi látni.

SqlStatements osztály – Az `SQLStatements` osztály, mint a neve is mutatja, nem más, mint a program működése során használt SQL utasítások gyűjteménye.

Validators osztály – A `Validators` osztály olyan metódusok halmaza, melyek az adatlapokról bekért különböző értékekről döntik el, hogy megfelelnek-e bizonyos formai vagy tartalmi követelményeknek.

3.4.3. Frames csomag

A `Frames` csomagban található osztályok a képernyőn jelennek meg panelként vagy ablakként.

MainFrame osztály – A program indító osztálya. Létrehozza a főablakot, majd innen vezérelhetjük a program működését. Lehetőségünk van az adatbázis-kapcsolat beállítására, majd létrehozhatunk új céges adatlapot, vagy folyathatjuk egy korábban megkezdett munkamenetünket. A céges adatlap betöltése után innen érhetjük el a szoftver fő szolgáltatásait.

OneRecord osztály – A `OneRecord` osztály tulajdonképpen egy `JPanel`, melyben a keresés során megkapott eredmények egy sorát tároljuk. A létrejövő panel tartalmaz két gombot is:

- Rontottá tesz - Ennek segítségével egy adott számlát megjelölhetünk rontottként
- Nyomtatás - Ezen gomb megnyomásával a számlát kinyomtathatjuk

OneRow osztály – A `OneRow` osztály a `JPanel` osztály leszármazottja. A `OneRow` osztály egy számlán szerepelő tételsor adatait jeleníti meg, mely a számlák nyomtatásánál kerül felhasználásra. A osztály konstruktora az objektum létrejöttekor automatikusan kitölti a címkéket a megfelelő értékekkel. Ezek sorrendben:

- Megnevezés - A termék/szolgáltatás megnevezése
- VTSZ - A termék/szolgáltatás VTSZ száma, illetve a hozzá fűzött egyéb megjegyzés
- ÁFA - A terméket/szolgáltatást sújtó áfakulcs
- Mennyiségi egység - A termék/szolgáltatás mennyiségi egysége
- Mennyiség - A termék/szolgáltatás mennyisége
- Egységár - A termék/szolgáltatás nettó egységára
- Összeg - A termék/szolgáltatás nettó egységárának és a mennyiségének szorzata

RecordList osztály – A `RecordList` osztály a `JDialog` osztály leszármazottja, így ez is párbeszédpanelként jelenik meg. A létrejövő ablak dinamikusan épül fel, alapértelmezett mérete egy számla megjelenítésére alkalmas, de 10 számláig növekszik az ablak mérete, afelett pedig megjelenik egy görgetősáv az ablak bal szélén. Az egyes számlák `OneRecord` objektumokban jelennek meg, s ezek az objektumok kerülnek fel az ablak belsejében található `JPanel`re.

StatisticsDiagram osztály – A `StatisticsDiagram` a `JComponent` osztály leszármazottja. A `StatisticsDiagram` a kimutatásokhoz készít egy diagramot: a beállított értékek alapján legenerálja az oszlopokat, megjeleníti őket a beállított színekkel, az egyes oszlopok alatt megjeleníti a hozzájuk tartozó dátumot, azok felett pedig a szimbolizált

összeget.

3.4.4. Frames.Forms csomag

A `Forms` csomag olyan, a `JDialog` osztályból származó osztályokat tartalmaz, melyek (mint a csomag neve is mutatja) valamilyen adatlapot jelenítenek meg. Ezek többsége kérdőívként működik, vagyis az adatok felvitelére használatos, azonban akad köztük olyan is, mely csupán a korábban eltárolt adatok alapján kitöltődik.

BillForm osztály – A `BillForm` osztály a `JDialog` osztály leszármazottja, így ez is egy párbeszédpanel. Az osztály a számlák nyomtatásánál épül fel dinamikusan: kitölti a cég és a partner adatai, majd a már korábban létrehozott `OneRow` objektumokat elhelyezi az adatlapon, s végül kiszámítja a számla nettó és bruttó végösszegét, valamint a teljes számlán az áfa értékét. A létrehozott ablak a képernyőn nem jelenik meg, csupán a nyomtatás után a papíron látható.

CompanyForm osztály – A `CompanyForm` osztály a `JDialog` osztály leszármazottja, így ez is egy párbeszédpanel, melyen a cég adatait adhatjuk meg a megfelelő szövegbeviteli mezőkben. Az adatlapon mindent megadhatunk ami csak szükséges a későbbiekben nyomtatandó számlákhoz: alapadatokat (név, számlaszám, adószám), elérhetőségeket. Nem kötelező jelleggel megadhatjuk a cég logóját, valamint felvihetjük a cég telefonszámát, email címét. Az adatokat menteni a Mentés, elvetni a Mégse gombbal tudjuk. A Mentés megtörténte előtt a szoftver ellenőrzi, hogy minden kötelező adat meg lett-e adva, valamint, hogy a megadott adatok formailag megfelelnek-e a követelményeknek. A mentés részeként létrejön egy szerializált, .ceg kiterjesztésű fájl a felhasználó számítógépén, mely megnyitásával később bármikor folytathatja a megkezdett munkát.

DataBaseForm osztály – A `DataBaseForm` osztály a `JDialog` osztály leszármazottja, így maga is egy párbeszédpanel, mely segítségével beállíthatjuk, hogy a program hol és hogyan tud csatlakozni az adatbázishoz. Az osztály a korábban tárolt adatokat a `database.properties` fájlból próbálja meg beolvasni. Ha a beolvasás során hiba lép fel, esetleg a fájl nem létezik, akkor újra létrehozza üres értékekkel. Az adatlapon megadott adatokat a Mentés gombra kattintva rögzíthetjük. Elvetni a Mégse gombbal lehet.

EntryForm osztály – Az `EntryForm` osztály a `JDialog` osztály leszármazottja, így ez is egy párbeszédpanel, melyen a számlák felviteléhez szükséges adatokat adhatjuk meg a megfelelő szövegbeviteli mezőkben. Az adatlapon mindent megadhatunk ami csak szükséges, az adott számlához több sort is csatolhatunk. A tételt rögzíteni a Mentés, elvetni a Mégse gombbal lehet. A Mentés megtörténte előtt a szoftver ellenőrzi, hogy minden kötelező adat meg lett-e adva, valamint, hogy a megadott adatok formailag megfelelnek-e a követelményeknek. Ebből az ablakból nyílik lehetőségünk új partnerek rögzítésére is.

PartnerForm osztály – Az `PartnerForm` osztály a `JDialog` osztály leszármazottja, így ez is egy párbeszédpanel, melyen új üzletfeleket vehetünk fel a rendszerbe. Meg kell adnunk az üzleti partner nevét, és telephelyét, valamint megadhatjuk a számlaszámát és az adószámát. A kötelező mezők kitöltöttségét a rendszer ellenőrzi, valamint azt is, hogy az adott partner nem szerepel-e még a rendszerben.

ParentForm osztály – A adatlapok ősosztálya, mely definiálja a közösen használt statikus változókat.

SearchForm osztály – Az `SearchForm` osztály a `JDialog` osztály leszármazottja, így ez is egy párbeszédpanel, mely segítségével az eltárolt számlák között végezhetünk keresést. A keresés első lépéseként ki kell választanunk, hogy az eredményben milyen típusú számlák jelenjenek meg: bevétel, kiadás, vagy esetleg mindkettő. Lehetőség van a keresés további szűkítésére. Megadhatjuk:

- a számla nettó végösszegének alsó, illetve felső határát
- a számla dátumának alsó, illetve felső határát
- a számlán szereplő tételek megnevezését és VTSZ számát

Ha a keresésnek van eredménye létrejön egy `RecordList` objektum, ellenkező esetben felugró ablakban közli a szoftver, hogy nincs a keresésnek eleget tevő számla a rendszerben.

StatisticsForm osztály – A `StatisticsForm` osztály a `JDialog` osztály leszármazottja, így ez is egy párbeszédpanel, melyen az eltárolt számlák alapján generáltathatunk oszlopdiagramokat. A diagramon szerepelhetnek a kiadások (piros), a bevételek (zöld) és a felhalmozott szabad készpénz (kék) alakulása. Minden oszlop alatt

megtalálható a dátum, melyhez kapcsolódik, valamint az oszlopok felett az összeg, melyet szimbolizál. Az oszlopok magassága automatikusan határozódik meg a legnagyobb és az aktuális érték függvényében. Az értékek kiszámításához kapcsolódó számlák szűrhetőek dátum alapján. Az elkészült diagramot kinyomtathatjuk, valamint valós időben válthatunk, hogy napi vagy havi lebontásban jelenítse meg az eredményeket.

4. Felhasználói kézikönyv

4.1. Használat előtti teendők

Szinte minden szoftverre igaz, hogy mielőtt használatba vehetnénk, különböző dolgokat kell tennünk, illetve egyéb, a szoftver működéséhez újabb programokat kell beszereznünk, a rendszerre telepítenünk, valamint megbizonyosodnunk, hogy a program akadálymentes futásához a számítógép hardverkiépítése megfelelő.

4.1.1. Szükséges szoftverek

A program futásához minimálisan szükséges a Java 5-os futtatókörnyezet (JRE). Amennyiben ez nincs, vagy ennél régebbi változat található a számítógépén, telepítse a legújabb JRE változatot, melyet bármikor elérhet a következő webcímen: http://java.com/en/download/index.jsp

A JRE mellett rendelkeznie kell a két támogatott adatbázis-kezelő rendszer (DBMS) egyikével a saját gépén, vagy bármely másik, hálózaton keresztül elérhető számítógépen. Mivel mindkettő ingyenes és nyílt forráskódú, szabadon letölthető a fejlesztő webhelyéről Windows és különféle Linux disztribúciók alá. Ezeket az ott megtalálható leírás alapján konfigurálni kell:

ApacheDerby: http://db.apache.org/derby/derby_downloads.html

MySQL: http://dev.mysql.com/downloads/mysql/

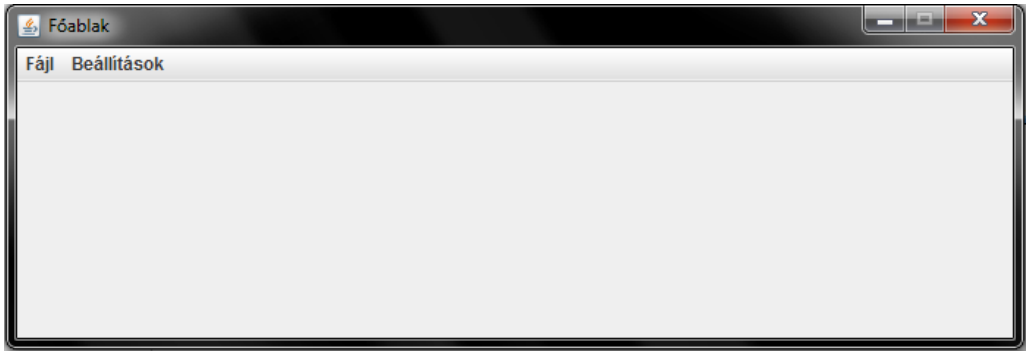
4.1.2. Minimális hardverigény

A szoftvernek nincs különösebb hardverigénye, így ha a kiegészítő szoftverek (a JRE, valamint helyi számítógépen futó adatbázis esetén a DBMS) elfutnak, a program is gond nélkül el fog. Ezen felül csupán egyetlen igénye van a szoftvernek, ez pedig valamilyen nyomtató megléte és csatlakoztatása, enélkül ugyanis nem lesz képes kinyomtatni az elkészült számlákat, illetve a kimutatások diagramjait.

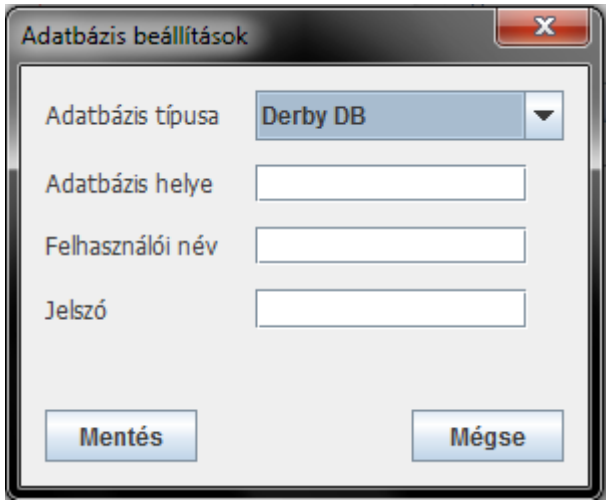
4.2. A program üzemeltetése

4.2.1. A program első indítása

A program indításakor a főablak fogadja:



Amennyiben első ízben indítja el a programot, mindenekelőtt konfigurálnia kell az adatbázis használatot. Ezt a *Beállítások* → *Adatbázis beállítások* menüpontra kattintva teheti meg. Ekkor a következő ablak ugrik fel:



Itt adhatja meg az adatbázis eléréséhez szükséges információkat.

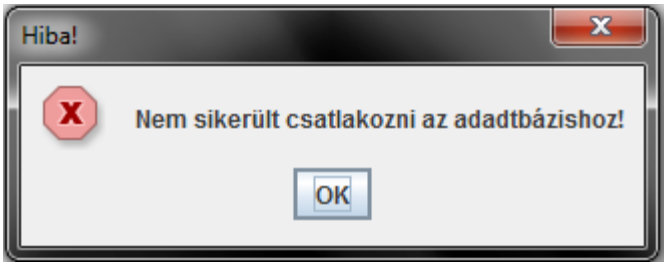
Először is ki kell választania az adatbázis-kezelő rendszer típusát. Amennyiben ezt helytelenül teszi meg, kapcsolódási hiba léphet fel.

Ezután meg kell adnia az adatbázis elérhetőségét a következő formában: `jdbc:`*`DBMS`*`://`*`gép_ip_címe`*`:`*`port_szám`*`/`*`adatbázis_neve`*, ahol:

- *DBMS* – az adatbázis-kezelőrendszer típusa, jelen esetben: Derby DB esetén a `derby`, MySQL esetén a `mysql` szónak kell itt szerepelnie
- *Gép ip címe* – annak a számítógépnek az IP címe, melyen a DBMS fut, helyi számítógép esetén elegendő a `localhost` szót megadni
- *Port szám* – annak a portnak a száma, melyen keresztül a megadott számítógéphez hozzáférhet a TCP/IP protokoll segítségével
- *Adatbázis neve* – az adatbázis neve, melyben majd a céghez tartozó adatok tárolásra kerülnek

Végül, amennyiben van, adja meg az adatbázishoz való csatlakozáshoz szükséges felhasználói nevet és jelszót.

Ezek mindegyikét az adatbázis adminisztrátorától kérheti. Kisebb vállalkozások esetén ez tipikusan ön lesz, aki feltelepítette a DBMS-t, s ezen paramétereket a telepítés során kellett megadnia (kivéve IP címét, mely adott). Amennyiben valahol helytelen adatokat adott volna meg, vagy az adatbázis jelenleg nem fut, a program jelezni fogja a kapcsolódás sikertelenségét:



A megadott információkat a *Mentés* gombra kattintva rögzítheti. Ekkor a program létrehoz a mappájában egy `database.properties` nevű fájlt, melyben ezeket tárolja. Amennyiben ez a fájl véletlenül törlésre kerülne, illetve tartalma módosulna, semmi baj nem történik, adatai az adatbázisszerveren biztonságban vannak, Önnek csupán újfent végre kell hajtani az adatlap kitöltését.

Ezen paraméterek módosítására a jövőben is bármikor lehetősége van (amennyiben nincs aktív adatbázis-kapcsolat), csupán a menüpontra kattintva az előugró adatlapon át kell írnia a megfelelő információkat, ily módon bármikor lehetséges az adatbázis áthelyezése más gépre,

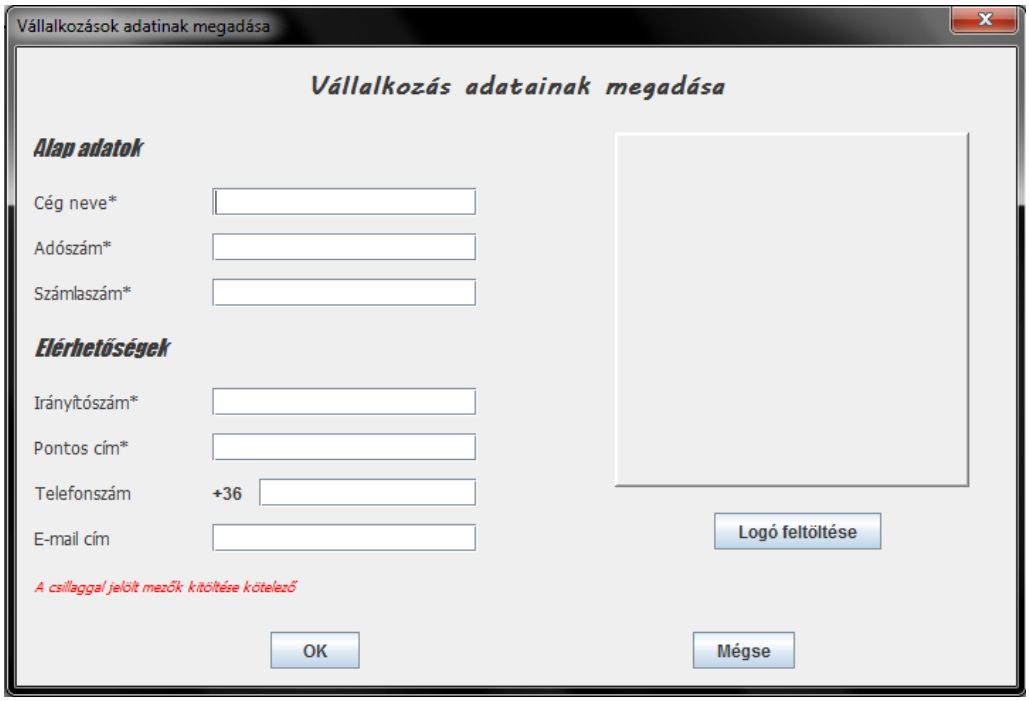
esetleg adatbáziskezelő-rendszer váltása, de az adatbázis-adminisztrátornak kell gondoskodnia arról, hogy az adatok is átkerüljenek az új helyre.

4.2.2. Céges adatok felvitele

A program indítása után mindig két lehetősége van:

- Újabb cég dokumentumainak kezelése
- Céges adatlap betöltése a munkamenet folytatásához

Amennyiben új céges adatlapot szeretne kitölteni, kattintson a *Fájl → Új cég* megadása menüpontra, s ekkor az alábbi képernyő fogadja:



Az adatlapot értelemszerűen kell kitöltenie, ahol a csillaggal jelölt mezők mindegyikének kitöltése kötelező:

- *Cég neve* – A vállalkozásának hivatalos, és teljes neve. Nem 0 hosszúságú szöveg
- *Adószám* – A vállalkozás adószáma. Kötelező formátuma minden esetben: *12345678-1-12*
- *Számlaszám* – A vállalkozás által hivatalosan használt bankszámla száma. Kötelező

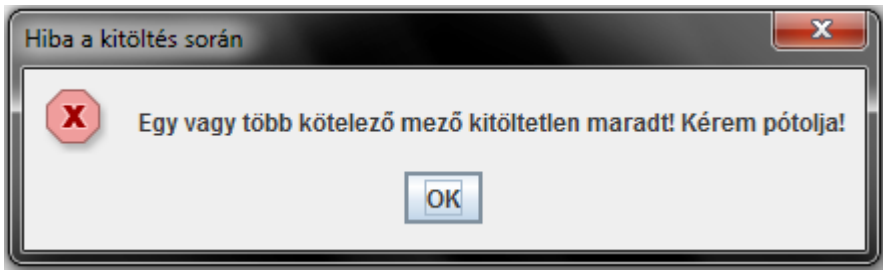
formátuma minden bank esetében: *12345678-12345678-12345678*

- *Irányítószám* – A vállalkozás telephelyének irányítószáma. Kötelezően egy 1000 és 9999 közé eső 4 jegyű szám
- *Cím* – A vállalkozás telephelyének pontos címe. Nem 0 hosszúságú szöveg, ahol fel kell tüntetni a várost, utcát, házszámot (esetleg emelet, ajtót is)
- *Telefonszám* – Opcionális mező. Lehet mobil vagy vezetékes telefonszám, elválasztójelek nélkül. Ország- és előhívó számot nem kell megadni, körzetszámot viszont igen
- *E-mail cím* – Opcionális mező. Csak olyan, érvényes email cím adható meg, mely megfelel az RFC 5322-es szabványban leírtaknak

Az adatlap kitöltése mellett lehetősége van (amennyiben az létezik) céges logó feltöltésére. Ehhez nem kell mást tennie, mint a *Logó feltöltése* gombra kattintania, majd a megjelenő panelen kitallóznia a cég logóját.

A rendszer a Java által natívan is támogatott kiterjesztéseket fogadja el képnek, ezek: *gif, jpeg, jpg, png, tiff*. Amennyiben a cég logója nem ezen formátumok valamelyikében tárolódik, kérjük először konvertálja át egy arra alkalmas alkalmazással. Fontos hogy a feltöltött fájl valóban kép legyen, ha nem az, a program hibaüzenetben jelzi.

Ha befejezte az adatlap kitöltését, kattintson a *Mentés* gombra, elvetéshez pedig Mégse opciót válassza. Ha valamilyen kötelező mező kitöltetlen maradt, vagy hibásan került kitöltésre, a program hibaüzenetben figyelmeztet és instrukciókat ad a kitöltés helyes módjára, s egész addig, míg minden mező nem helyes, nem engedi a mentést. Ha úgy dönt, hogy az opcionális mezőket is kitölti, a rendszer azok helyességét is szigorúan megköveteli és amíg az nem teljesül, addig ugyanúgy figyelmeztet a hibára, mint a kötelező mezők esetében.



Ha minden mező helyesnek bizonyul, a *Mentés* gombra kattintva a program létrehoz egy `ceg` kiterjesztésű fájlt, melynek neve a cég neve, s ebben tárolja az összes megadott adatot, majd a már korábban beállított adatbázishoz próbál kapcsolódni.

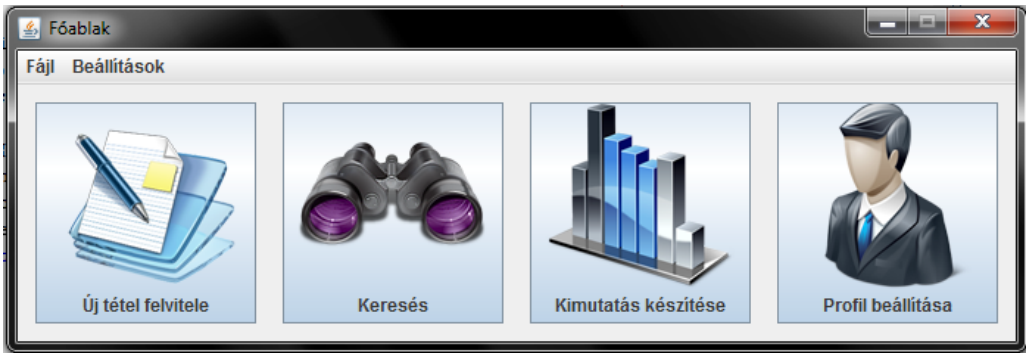
Amennyiben ez a fájl elvész, vagy sérül, az adatbázisba felvitt adatokért aggódnia nem kell, hiszen ez is csak egy konfigurációs fájl, s csupán a cég adatait tárolja. Nyugodtan újra létrehozhatja, s az adatbázishoz való csatlakozás után újra elérhető minden adata.

Ha korábban már létrehozta a céges adatlapot és munkáját szeretné folytatni, a *Fájl → Cég adatbázisának megnyitása* menüpontra kattintás után kitallózhatja a fájlt és betöltheti. Ebben az esetben a rendszer, szintén automatikusan megkísérli a kapcsolódást az adatbázishoz, s amennyiben ez sikertelen, a korábbiakhoz hasonlóan hibaüzenetben jelzi.

Profilja módosítására a későbbiekben bármikor lehetősége van. A profil betöltése után előbukkanó főmenüben a *Profil beállítása* gombra kattintva a korábbi ablak jelenik meg újra, immáron a felvitt adatokkal. Amennyiben menti, a konfigurációs fájl automatikusan felülírásra kerül.

4.2.3. Új tételek felvitele

Miután betöltötte a cég adatlapját, a főablakban megjelennek a vezérlőgombok, melyekkel a program fő szolgáltatásait érheti el:



Ezek közül az első az *Új tétel felvitele*, ahol is új kiadást, illetve bevételt rögzíthet az adatbázisban. A gombra kattintva előjön az adatlapja (kép a 47. oldalon).

Első teendő, hogy kiválassza a bejegyzés típusát, mely lehet bevétel vagy kiadás, itt a döntés értelemszerű.

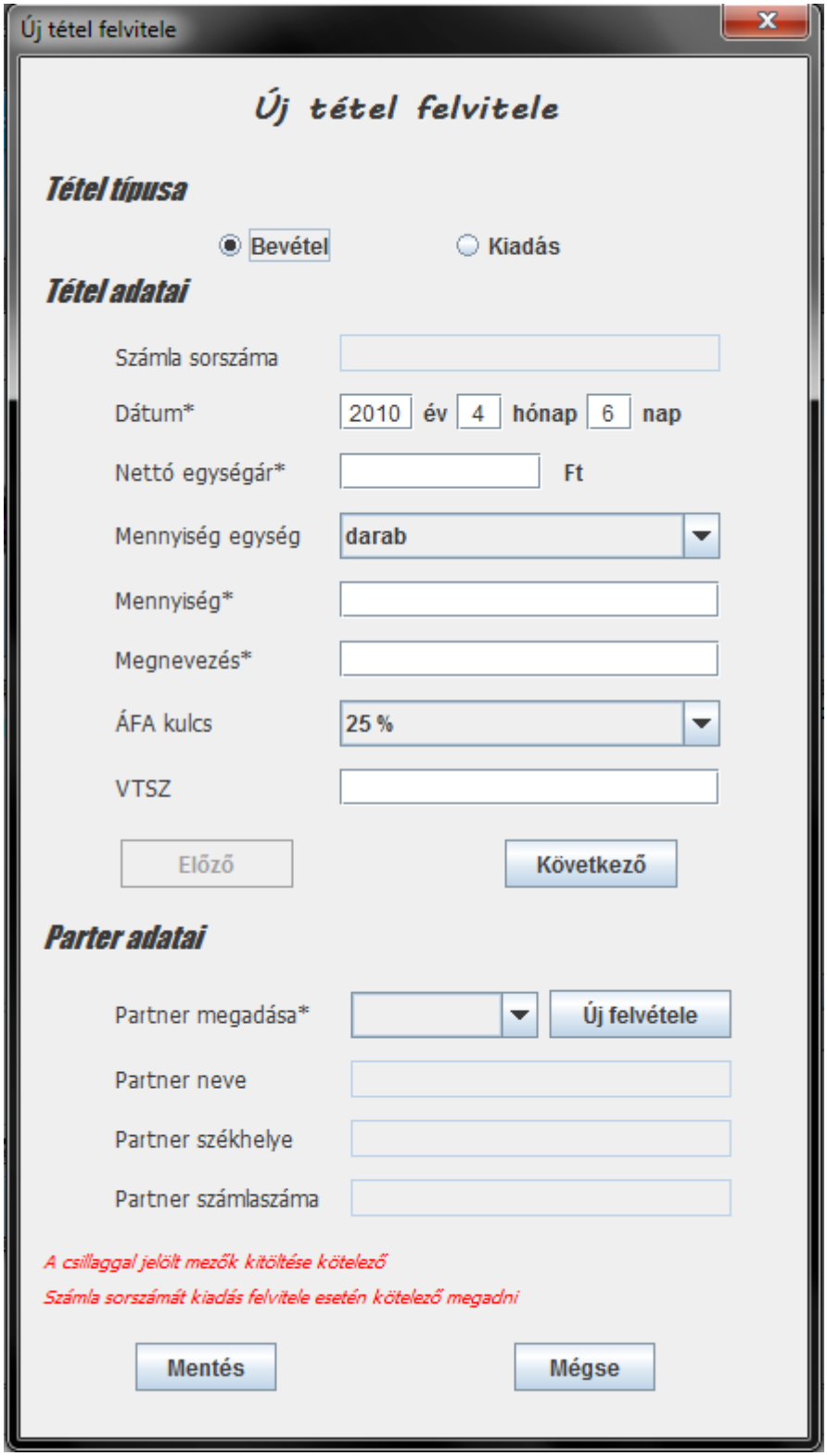
Ezt követően a számla adatait kell megadnia. Amennyiben kiadást szeretne rögzíteni, a számla sorszámát Önnek kell megadnia a megkapott számla alapján, azonban ha bevételt rögzít, tehát a számlát ön adja, annak a sorszámát a rendszer automatikusan generálja, tehát, Önnek ezen a ponton nincs teendője.

Ez után kötelező jelleggel következik a számla dátuma. Az ablak megjelenésekor automatikusan kitöltődik az aktuális dátummal, mivel általában a számla beérkeztekor/kiállításakor kerül felvitelre az adott tétel.

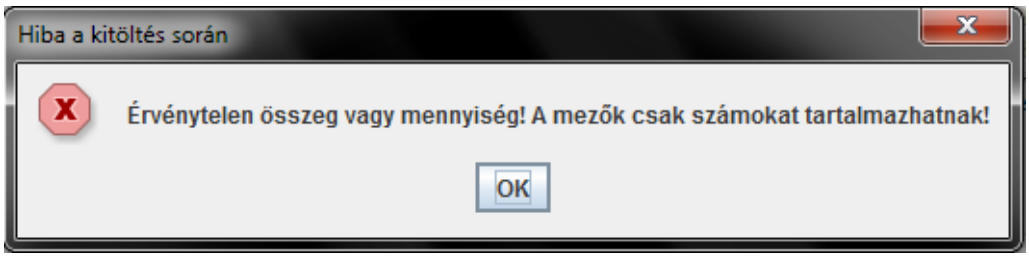
A dátum sora után a számlán szereplő egyes sorok felvitelére van lehetősége. Minimálisan egynek szerepelnie kell, maximalizálva nincs, de ahogy a papír alapú számlázásnál, úgy az elektronikusnál sincs értelme egyszerre túl sok rekordot szerepeltetni. Egy ésszerű maximum választása tanácsos, melynek értékét ajánlott 10 és 20 között megválasztani. Amennyiben egy tétel elkészült, a *Következő* gombra kattintva nyílik lehetősége újabb hozzáadására. Amennyiben már korábban rögzítetthez szeretne visszatérni, az *Előző* gombbal teheti meg. Az egyes tételsoroknál a csillaggal megjelölt mezőket kötelezően ki kell tölteni. Minden tételhez kapcsolódóan az alábbi adatokat kell/lehet megadnia:

- *Nettó egységár* – A tételben szereplő szolgáltatás/termék egységenkénti ára. Csak számot tartalmazhat.
- *Mennyiségi egység* – A termék/szolgáltatás mennyiségi egysége, mely lehet: darab, kilogramm, méter, liter, vagy alkalom (valamilyen munkavégzés esetén ajánlott választás).
- *Mennyiség* – A termék/szolgáltatás mennyisége, a számla végösszegének kiszámításához szükséges. Csak számot tartalmazhat.
- *Megnevezés* – A termék/szolgáltatás megnevezése, mely a számla egyértelműségéhez szükséges. Nem 0 hosszúságú szöveg.
- *ÁFA kulcs* – A terméket/szolgáltatást terhelő általános forgalmi adó kulcsa, mely lehet 25%, 18%, 5% és 0%. Itt általánosan a 25%-ot kell kiválasztania, a kedvezményes, 18%-os sávba, csak bizonyos termékek/szolgáltatások esnek bele, mint például a tej, liszt stb., illetve az 5%-ba a gyógyszerek, folyóiratok stb. A pontos, és teljes listát megtalálhatja az APEH honlapján, a www.apeh.hu webcímen.

- VTSZ szám – Az elvégzett szolgáltatás statisztikai besorolása vagy a termék vámtarifaszáma, melyet 2008. május 1-je óta nem kötelező szerepeltetni a számlákon.



Minden tételsor rögzítésénél, a rendszer ellenőrzi a megadott adatok helyességét, illetve a kötelező adatok szerepeltetését. Amennyiben hibát észlel, nem engedi sem a mentést, sem újabb sor felvitelét, valamint hibaüzenetben figyelmeztet a hibára:



Ahhoz, hogy egyáltalán elkezdhesse rögzíteni az tételsorokat, ki kell választania üzleti partnerét, akivel az üzletet megkötötte. Amennyiben a másik féllel már korábban kapcsolatba került, nem kell mást tennie, mint egyszerűen a legördülő listából kiválasztania, így a program az adatbázisból lekéri annak adatait.

Ha még nem volt üzleti kapcsolata korábban az aktuális üzletféllel, az *Új felvitele* gombra kattintva rögzítheti az adatbázisban a ügyfelet, melyhez az alábbi adatlapot kell kitöltenie:

Minimálisan meg kell adnia a partner nevét és a telephelyének címét. Amennyiben tudomása van a partner számlaszámáról és/vagy adószámáról, javasolt azokat is feltüntetnie, hiszen ezekkel pontosíthatja, részletesebbé teheti a későbbiekben nyomtatásra kerülő számlákat. Az adószámra és a bankszámla számra, ugyanazok a követelmények teljesülnek

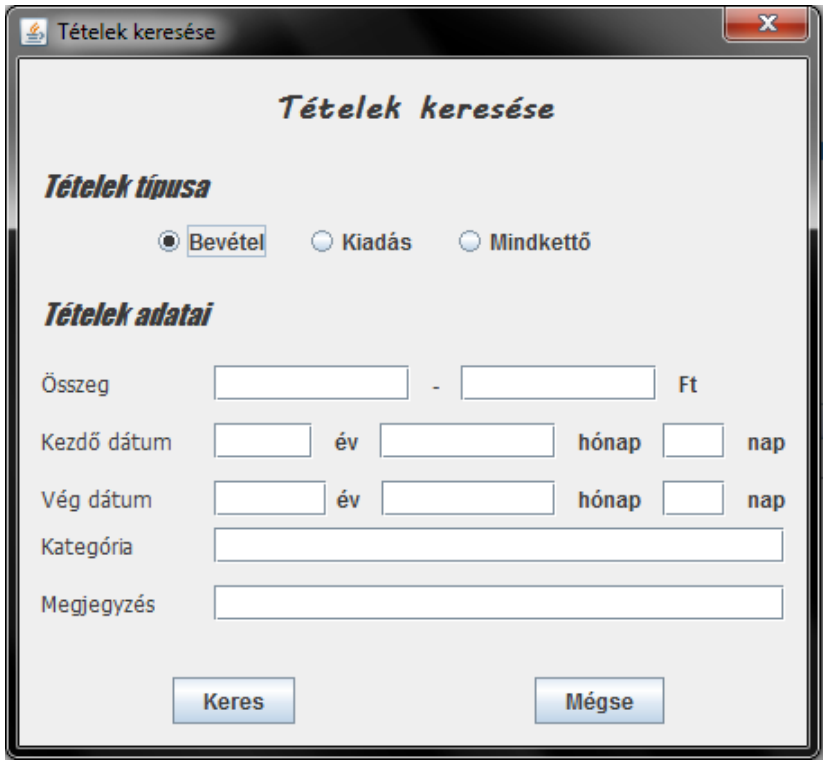
mint amelyek korábban, a céges adatlap kitöltésekor szerepeltek. Hiba esetén a rendszer, a korábbiakhoz hasonló módon, hibaüzenetben figyelmeztet.

Miután megfelelően kitöltötte az adatot, a *Mentés* gombbal rögzítheti az adatbázisban partnere adatait, ha esetleg meggondolta volna magát, a *Mégse* gombbal vetheti el az adatlapot.

Ha minden adat helyesen szerepel, megadta a partnert, rögzítésre kerülhet a tranzakció az adatbázisban, melyhez a *Mentés* gombra kell kattintania. Ha esetleg meggondolta magát és elvetné az adatlapot, a *Mégse* gombbal teheti meg.

4.2.4. Tételek visszakeresése

A főmenüből megnyitott ablakban a korábban rögzített tételek között kereshetünk különböző szempontok szerint.



Mindenekelőtt be kell állítania milyen tételeket keres: bevételt, kiadást vagy mindkettőt. A sikeres kereséshez minimálisan ezt meg kell határoznia.

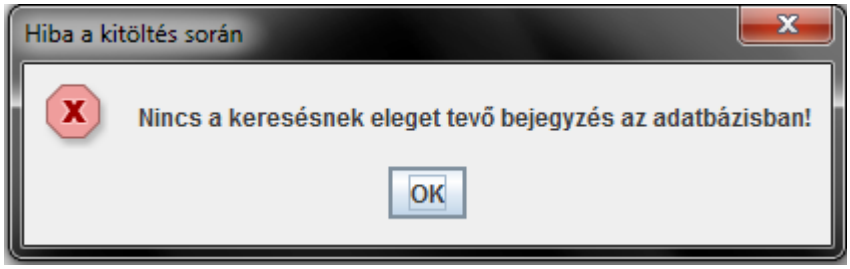
A típus meghatározás után egyéb részletek beállításával pontosíthatjuk a keresést, ily

módon szűkíthetjük a találatok listáját. Nem kötelező egyetlen más mezőt sem kitöltenie, azonban amennyiben valamelyiket kitölti, annak minden tekintetben helyesnek kell lennie. Ha nem az, a korábbiakhoz hasonlóan hibaüzenetben jelzi a program.

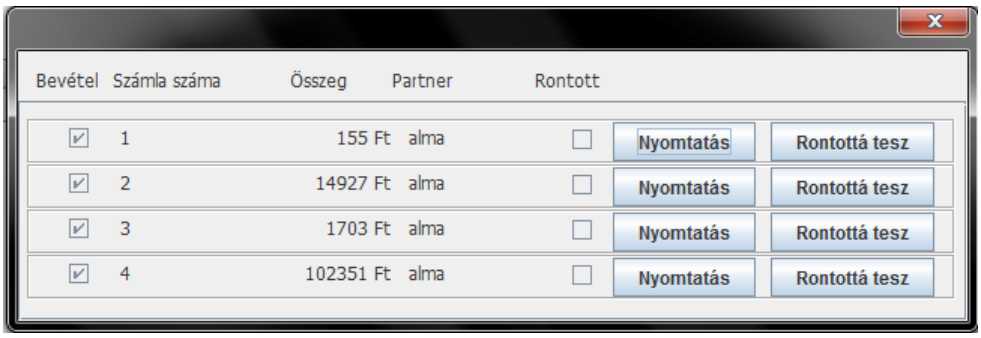
A alábbi információkat adhatjuk meg:

- *Összeg* – Az egyes bejegyzések végösszegére vonatkozik. Megadható csak az alsó és csak a felső határ is, de akár mindkettő is. Csak számok szerepelhetnek benne.
- *Kezdő dátum* – szabályos dátum kell legyen, azon tételek felelnek meg a feltételnek, melyek dátumánál későbbi dátum szerepel mint ez.
- *Vég dátum* – szabályos dátum kell legyen, azon tételek felelnek meg a feltételnek, melyek dátumánál korábbi dátum szerepel mint ez.
- *Kategória* – bármilyen szöveg beírható, azon tételek felelnek meg a feltételnek, melyek kategóriája megegyezik a beírt szöveggel.
- *Megjegyzés* – bármilyen szöveg beírható, azon tételek felelnek meg a feltételnek, melyek VTSZ száma megegyezik a beírt szöveggel.

Ha a keresésnek nincs eredménye, azt üzenetben jelzi a rendszer:



Amennyiben a keresés sikerrel jár, a program az eredményekből dinamikusan felépít egy táblát, mely az alábbihoz hasonlóan néz ki:



Minden sor egy-egy bejegyzést jelent, kitöltve az eredmények alapján.

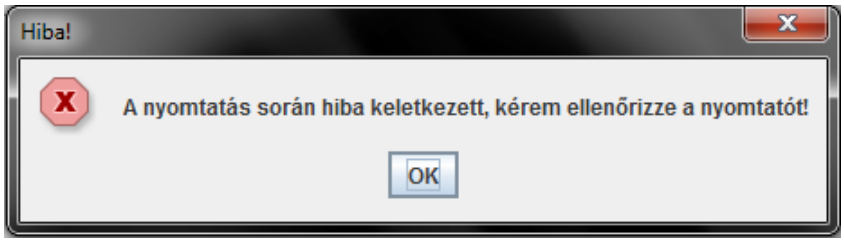
Az első oszlop egy jelölőnégyzetet tartalmaz, amely be van jelölve, ha bevétel az adott tétel, illetve nincs, ha az kiadás.

A második oszlop a számla sorszámát tartalmazza, kiadás esetén a teljes sorszámot, bevétel esetén csupán azt, hogy hányadikként került rögzítésre.

A harmadik oszlopban a számla ÁFA nélküli végösszege szerepel, majd ezt követően a partner neve látható, végül egy oszlop, mely azt jelöli, hogy az adott tételt rontottá tettük-e.

Az adatbázisból törvényi előírás miatt nem lehet törölni, ugyanis bármilyen APEH ellenőrzéskor el kell tudni számolni minden egyes korábban kiadott számlával. Előfordulhat azonban, hogy elront egy tételt, s ezt javítani szeretné. Mi ekkor a teendő? Ezt a célt szolgálja a *Rontottá tesz* gomb. Ezzel átállítódik az adott tétel állapota rontottá, s ezentúl úgy viselkedik, mintha nem is lett volna, nyugodtan újra létrehozhatja. Rontottá csak nem rontott, illetve bevételről szóló bejegyzést tehet, vagyis kiadásról szólót nem.

Ebben az ablakban van lehetősége a számlák nyomtatására is. Számla nyomtatása csak nem rontott bevételről lehetséges. A számla formanyomtatványát az adatlapon, a partnerek felvitelénél, illetve az egyes tételek felvitelénél megadott adatok alapján automatikusan tölti ki a rendszer, s egy, a következő oldalon található képhez hasonló eredményt hoz létre, mely azonban nem jelenik meg a monitoron, csak nyomtatásban. Amennyiben nincs csatlakoztatva nyomtató, vagy bármilyen egyéb probléma merül fel a nyomtató elérésnél, az alábbi hibaüzenetet generálja:



A formanyomtatvány alapesetben az oldal közepére, fektetve jelenik meg. Ez tetszőlegesen átállítható egy szokásosnak nevezhető, előbukkanó nyomtatási beállítások párbeszédpanelen. Ha tudjuk, hogy sok sorból áll, ajánlott az olvashatóság érdekében a tájolást átállítani állóra. A nyomtatvány bal felső részén lévő szürke négyzetben a cég logója jelenik meg, ha van.

A jobb felső sarokban találhatóak a partner adatai, valamint a kapcsolódó dátumok. Feltételezve, hogy a számla a teljesítéskor kerül kinyomtatásra, a Kibocsátás és a teljesítés dátuma megegyezik, a fizetési határidő, pedig minden esetben egy hónapban határozott meg. A számla teljes mértékben helyettesíti a papír alapú számlázást.

A papíron megjelenő kép annyiban tér el az ábrán láthatótól, hogy a fejlécben szerepel, hogy az adott számla hányadik példánya, emellett pedig első példány esetén az *Eredeti*, minden más példány mellett pedig a *Másolat* felirat.

4.2.5. Kimutatások készítése

Felmerülhet az igény a vállalatnál, hogy grafikon lássák a bevételek, kiadások, illetve a felhasználható szabad készpénz alakulását. Ennek több oka lehet, csak hogy egy példát említsek, egy-egy új termék piacra kerülésének hatásait vizsgálják a kereslet függvényében. Ebben segíthet ez a komponens, mely oszlopdiagramot generál a beállított paramétereknek megfelelő felvezetett tételek alapján.

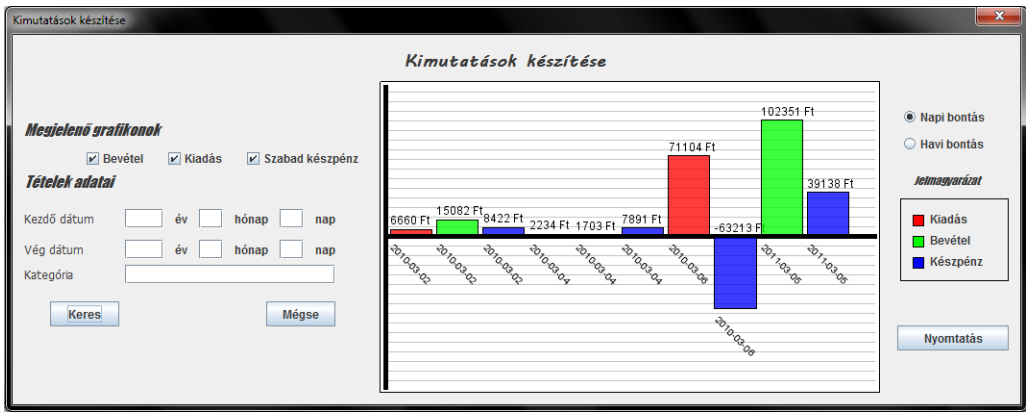
Elsőként kötelezően ki kell választania, hogy mely oszlopok jelenjenek meg majd a diagramon (bevétel, kiadás, szabad készpénz). Amennyiben ezt elmulasztja, a keresés nem hajtódik végre, valamint a hibára üzenet hívja fel a figyelmet.

A további szűkítés opcionális lehetőség, azonban ha úgy dönt használja őket, a program

megköveteli a helyes adatok megadását, s csak úgy mint korábban hiba esetén figyelmeztet. Ezen szűkítési lehetőségek a korábbiakhoz hasonlóak:

- *Kezdő dátum* – szabályos dátum kell legyen, azon tételek felelnek meg a feltételnek, melyek dátumánál későbbi dátum szerepel mint ez.
- *Vég dátum* – szabályos dátum kell legyen, azon tételek felelnek meg a feltételnek, melyek dátumánál korábbi dátum szerepel mint ez.
- *Kategória* – bármilyen szöveg beírható, azon tételek felelnek meg a feltételnek, melyek kategóriája (megnevezése) megegyezik a beírt szöveggel.

Amennyiben a keresés lefut és sikeres, az alábbihoz hasonló eredményt hoz:



A grafikontól jobbra található két választógomb, mellyel eldöntheti, hogy az eredményeket napi, vagy havi bontásban kívánja látni. Ez főleg akkor hasznos, ha nagyobb időintervallumot felölelő kimutatást készít.

A választógombok alatt egy jelmagyarázat helyezkedik el, mely egyértelműsíti, hogy melyik oszlop mit jelent, nevezetesen:

- Piros – Kiadás
- Zöld – Bevétel
- Kék – Szabad készpénz

Az egyes oszlopok fölött megtalálható az érték, melyet megjelenít, illetve alatta a hozzá kapcsolódó pontos dátum (havi bontás esetén az adott hónap).

Végül ez alatt egy gomb kapott helyet, mellyel kinyomtathatja a kimutatást. A nyomtatás nem korlátozódik csak a grafikonra, hanem a teljes ablak kinyomtatódik, így a későbbi áttekintés, illetve valamilyen beszámoló során rögtön látható, hogy milyen feltételek mellett alakult úgy a grafikon, ahogy. Az alapértelmezett nyomtatóbeállítás itt is az oldal közepére fektetett kicsinyítés, de ez a felugró *Nyomtatás beállítása* panelen természetesen megváltoztatható. Amennyiben a nyomtatás esetén hiba lép fel, ahogy azt korábban, most is hibaüzenetben jelzi a rendszer.

5. Összefoglalás

A szoftver fejlesztésekor számos problémába ütköztem, melyek leküzdése során jelentős mennyiségű hasznos tapasztalatot gyűjtöttem, a Java nyelvről eddig megszerzett ismereteim mélyültek, problémamegoldó képességem fejlődött.

Akadt azonban egy olyan probléma, melyet egyelőre nem sikerült leküzdenem, ez a weben végzett banki ügyintézés. A funkció lényege, hogy kihasználva a bankok online szolgáltatásait, interneten keresztül végezhet pénzügyi tranzakciókat a szoftver tulajdonosa, s az így végrehajtott műveleteket automatikusan rögzíti a program az adatbázisban, valamint aktív internetkapcsolat esetén figyeli a pénzmozgásokat.

A jövőbeni fejlesztések között mindenképp szerepel a fent említett webes funkció megvalósítása, valamint újabb adatbáziskezelő-rendszerek támogatása (például: Oracle).

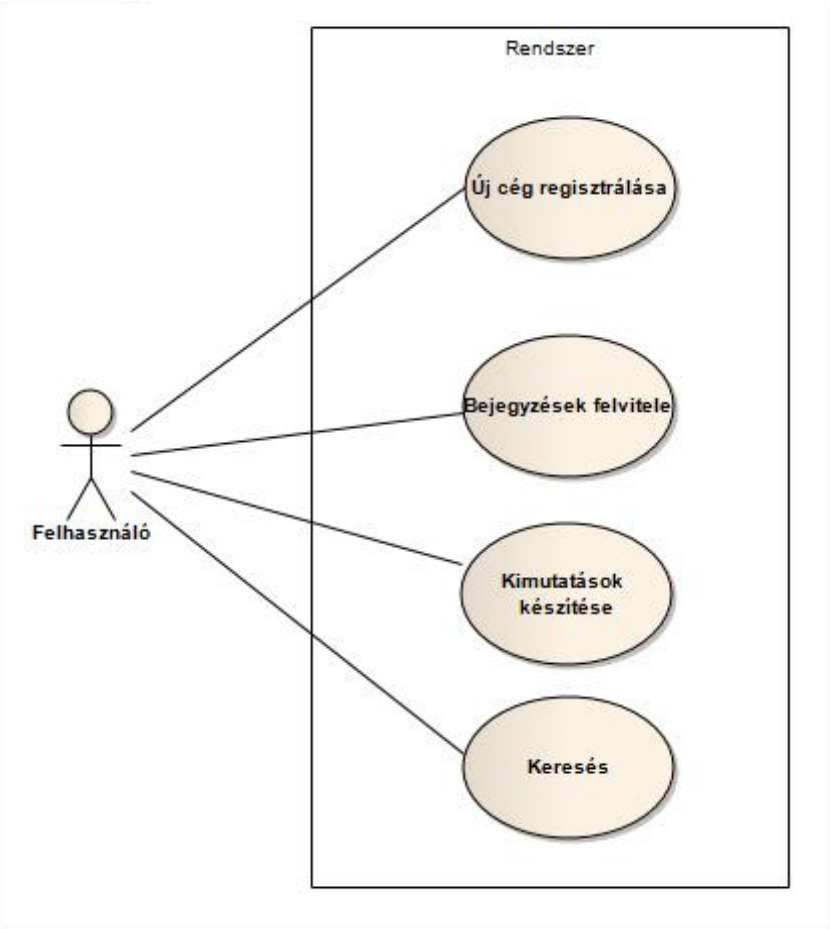
Valamint ahhoz hogy egy szoftvert teljesnek és befejezettnek nyilváníthassunk, szükség van a szoftver alapos teszteléséhez. Minden program esetében a tesztelés hosszú és nehézkes folyamat. A fejlesztési idő jelentős részét, kb. 60-80%-át ez képezi.

Léteznek automatikus tesztelést lehetővé tevő eszközök számos programozási nyelvhez, Java esetén a legelterjedtebb JUnit. Ezen eszközök (főleg nagyobb rendszerek esetén) jelentősen meggyorsítják a szoftver komponenseinek tesztelését. Ezen szoftvernél használatukra sajnos nem került sor, így a jövőben ez is mindenképp pótlásra fog kerülni. Mindemellett az igazsághoz hozzátartozik, hogy a szoftver huzamosabb használata során sem tapasztaltam a szoftver jelen verziója mellett semmiféle hibát egyik adatbázis-kezelő rendszer használata mellett sem.

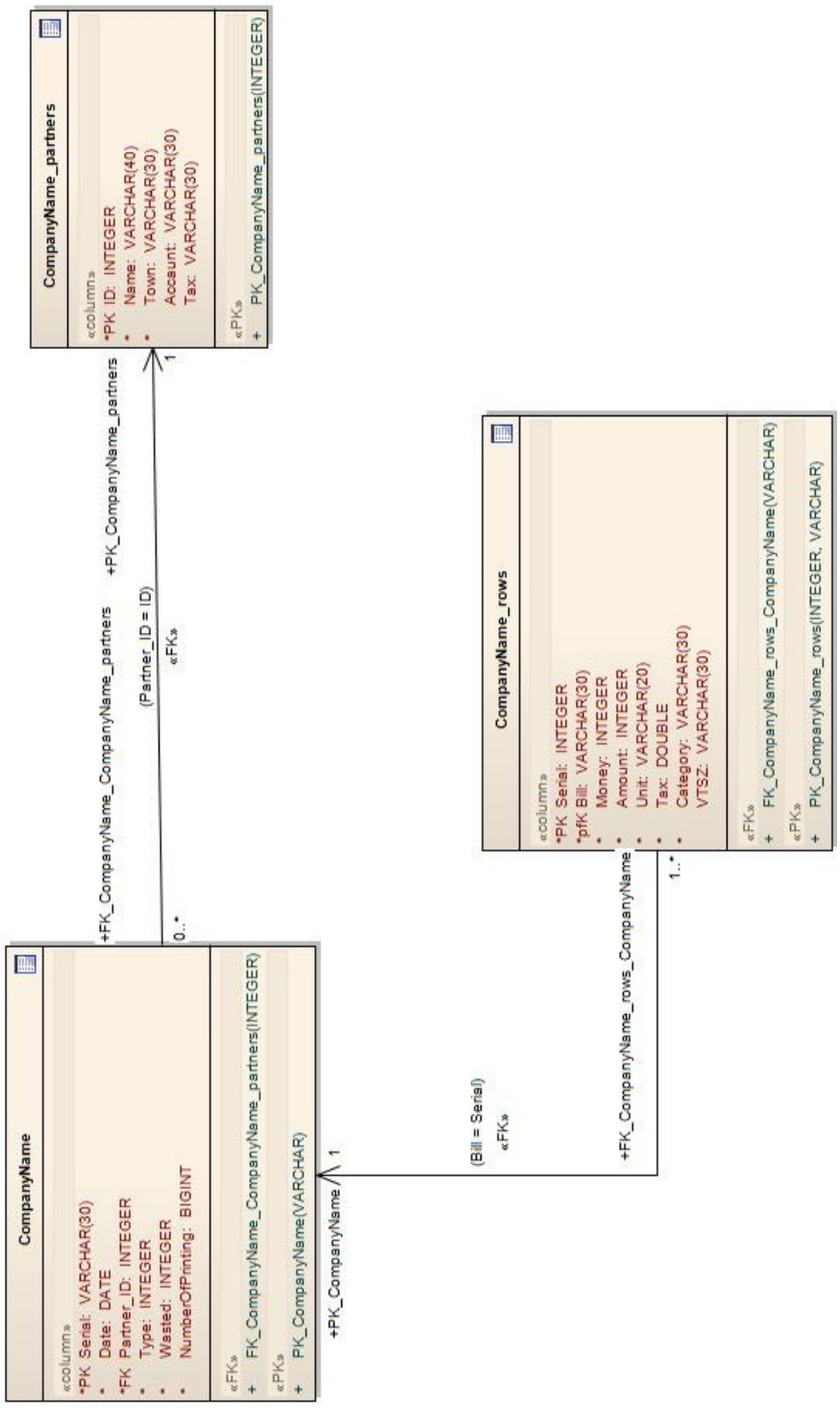
Remélem, hogy egy napon a szoftver az APEH által is elfogadásra kerül, s így hivatalosan is használható lesz és sok vállalkozót ösztönöz az elektronikus ügyintézésre való áttérésre.

6. Mellékletek

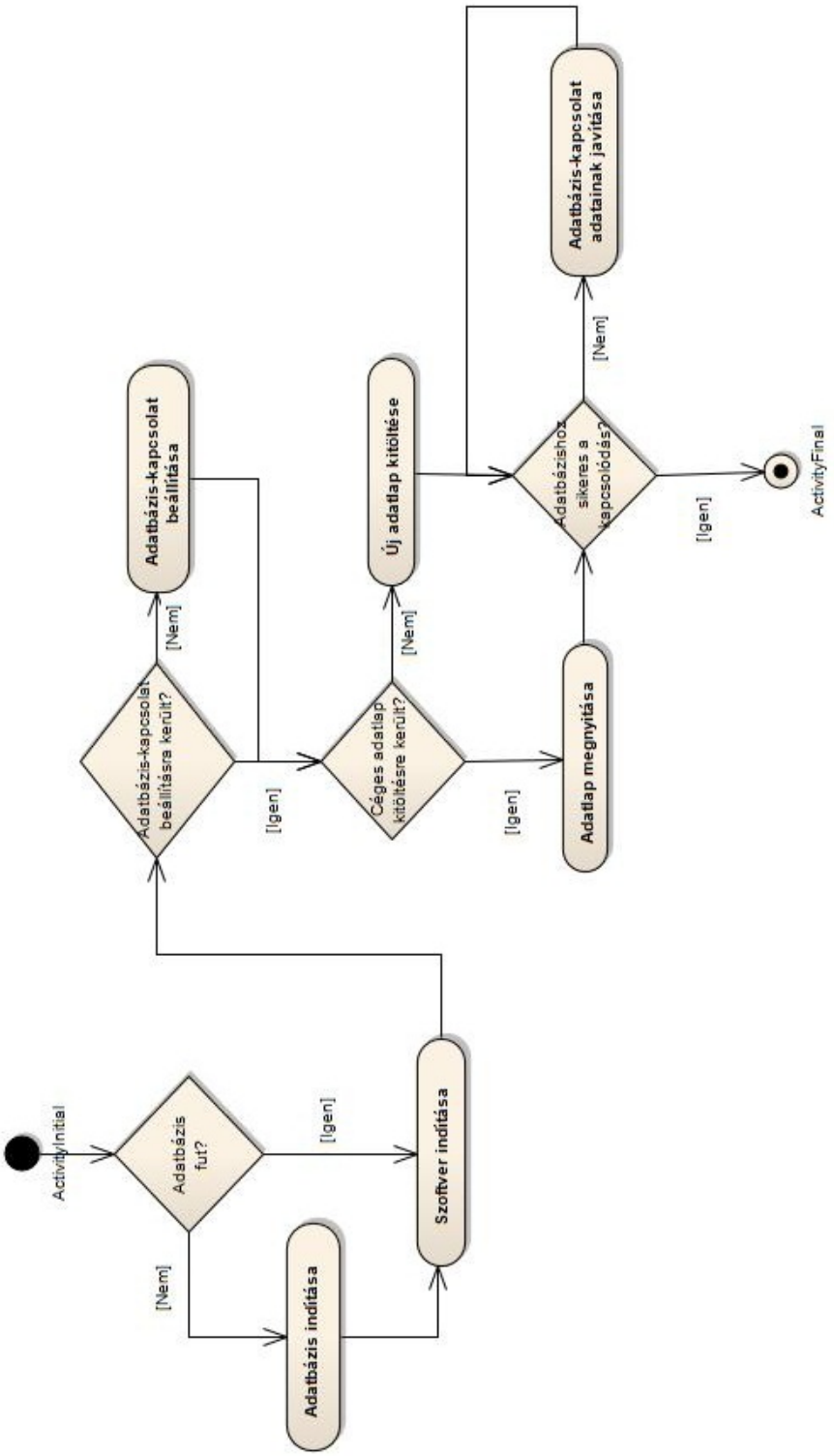
6.1. 1. számú melléklet – Használati eset diagram

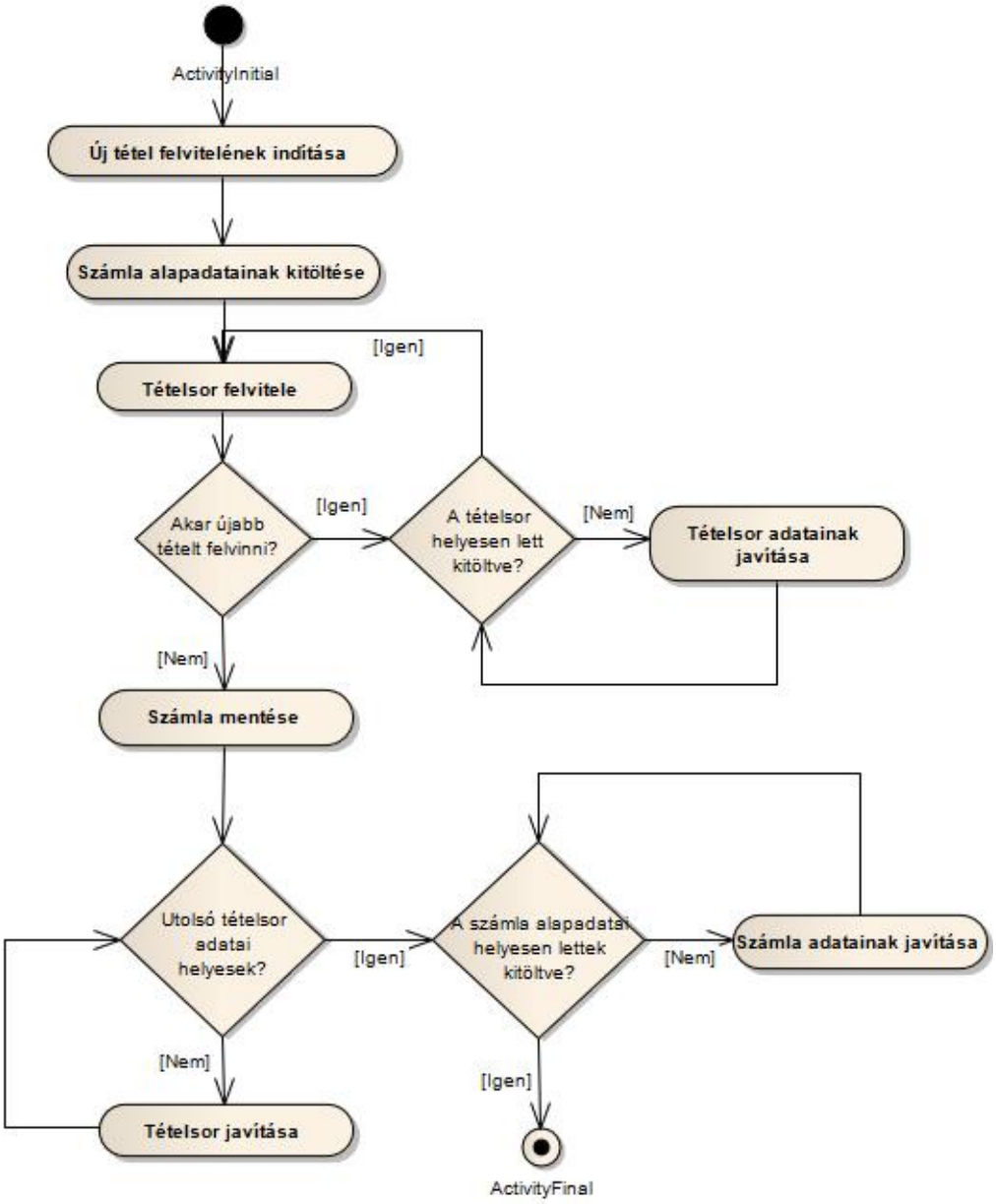


6.2. 2. számú melléklet – Adatbázis séma



6.3. 3. számú melléklet – Aktivitás diagramok




ActivityInitial
Új tétel felvitelének indítása
Számla alapadatainak kitöltése
[Igen]
Tételsor felvitele
Akar újabb tételt felvinni?
[Igen]
A tételsor helyesen lett kitöltve?
[Nem]
Tételsor adatainak javítása
[Nem]
Számla mentése
Utolsó tételsor adatai helyesek?
[Igen]
A számla alapadatai helyesen lettek kitöltve?
[Nem]
Számla adatainak javítása
[Nem]
Tételsor javítása
[Igen]
ActivityFinal

7. Irodalomjegyzék

- A főablakban ikonjai: http://www.iconspedia.com (minimálisan a „nem kereskedelmi célokra ingyenesen felhasználható” licenc megjelöléssel). A csomagok:
    - *Crystal Clear Actions*
    - *Application Interface*
    - *Next Series*
- A MySQL hivatalos honlapja [2]
- Juhász István – Programozás 2 [3]
- http://www.javaworld.com [4]
- Java hivatalos weboldala [5] [6]
- http://en.wikipedia.org/wiki/Java_%28programming_language%29
- Hivatalos JDBC API specifikációk [7]
- The Java Tutorial [8]
- Joshua Bloch - Effective Java [9]
- Apache Derby kézikönyv [10]
- Hivatalos MySQL kézikönyv [11]
- National Geographic honlapja [12]
- Nagy László - Egy szoftver fejlesztése Java Swing és HSQLDB környezetben

8. Köszönetnyilvánítás

Köszönöm Édesapámnak, aki vállalkozóként a vállalatok pénzügyeinek intézése során felhalmozott tapasztalatát, ismereteit használva szívesen adott tanácsot, segített, hogy szoftverem a hatályos jogszabályoknak megfelelően működjön.

Köszönöm témavezetőmnek, Espák Miklósnak, hogy végigkísért a szakdolgozat megírásánál, valamint köszönettel tartozom barátomnak és egykori tanáromnak, Bóka Antalnak, aki elvállalta a külső témavezető szerepét, s akihez bátran fordulhattam, ha a fejlesztés folyamán megoldhatatlannak látszó problémába ütköztem.