DIPLOMAMUNKA

Lőrinczi József

Debrecen

2008

Debreceni Egyetem

Informatikai Kar

JAVA ALKALMAZÁSFEJLESZTÉS

Témavezető:

Iszály György Barna

NYF-Főiskolai adjunktus

Készítette:

Lőrinczi József

Programtervező matematikus

Debrecen

2008

Tartalomjegyzék

1. Bevezetés

Manapság a javában való programozás elég divatos lett. Ezt egyrészt a platformfüggetlenségének, másrészt a könnyű elsajátíthatóságának és nem utolsó sorban az ingyenességének köszönheti.

Azért ezt a témát választottam a diplomamunkámnak, mert nekem is tetszik a java nyelvben a platformfüggetlenség. Nem kell az alkalmazásunkat külön lefordítani a kívánt platformra, mivel ennek a feladatnak az elvégzésére ott a Java virtuális gép. A másik dolog, ami a java nyelv mellett szól, az a szemétgyűjtő módszer. Nem kell azzal foglalkoznia a fejlesztőnek, hogy hogyan szabadítsa fel a lefoglalt memóriát, hisz a JVM ezt is megoldja helyette.

Az alkalmazásfejlesztést egy könyvtári nyilvántartó program fejlesztésén keresztül mutatom be. A programot használhatja mindenki. Ha nem könyvtáros és nem olvasó az illető, akkor is tud keresni az adatbázisban a könyvek között. Az olvasók ezeken túl meg tudják nézni milyen könyveket kölcsönöztek és bizonyos személyes adatokat is meg tudnak változtatni. Természetesen a könyvtárosoknak van a legtöbb joguk: könyvet tudnak felvinni az adatbázisba, képesek módosítani a tárolt adatokat, adott olvasónak könyvet tudnak kölcsönözni, s ha visszahoznak egy könyvet, akkor ezt is ők tudják feljegyezni.

A javában azon programozási módokat, technikákat mutatom be részletesen, amelyeket alkalmazásom fejlesztése során érintek. Az alkalmazás JDBC-n keresztül kommunikál majd az adatbázissal, ezért ezt a részt külön kiemelném a dolgozatomban. Azért is fordítok erre a részre nagyobb hangsúlyt, mert minden a felületen bekövetkezett eseményt, majdnem mindig adatbázis művelet követ.

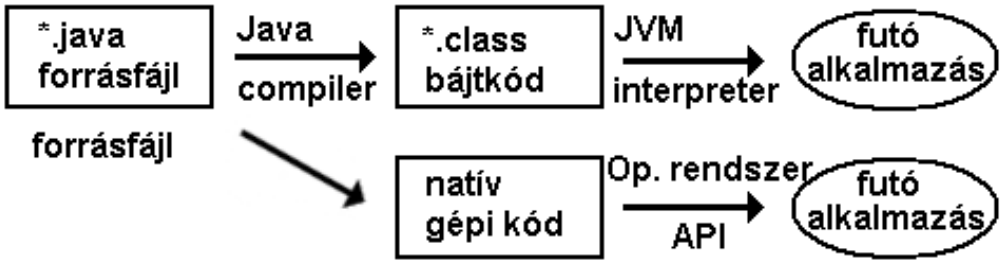
2. JAVA lehetőségei

2.1. A nyelv története

A Sun Microsystems a 90-es évek elején kezdte el fejleszteni a Java-t. A Green Project csapata az interaktív televíziózásban látta a jövőt, ezért elkezdtek megalkotni egy olyan nyelvet, amely platformfüggetlen. Ezáltal különböző platformok szorosabban együtt tudnak működni. Kezdetben a nyelv az Oak nevet kapta, ami tölgyfát jelent, de ez a név már foglalt volt, így 1995. május 23-án jelentették be a Java nyelvet.
A nyelv szintaxisa főleg a C és a C++ nyelvekéhez hasonlít. Teljesen objektumorientált programozási nyelv, de egyszerűbb objektummodellel rendelkezik, mint a C++.

2.2. Java futtató rendszere

Ahhoz, hogy egy java programot lefuttassunk, első lépésként a forrásállományokat bájtkódra kell lefordítani. Lehetőség van natív gépi kódra való fordításra is, ekkor egy adott processzor architektúrára történik a fordítás, de így a lefordított program elveszíti hordozhatóságát. A fordítás után előálló bájtkódot a java virtuális gép hajtja végre, interpretálja.



1. ábra Java forrás futtatása

2.3. A nyelv legfontosabb tulajdonságai

- **Egyszerű:** Mivel a nyelv szemantikája és szintaktikája nagyban hasonlít a C és a C++ nyelvekéhez, ezért ezen nyelvek ismerete esetén könnyű elsajátítani. Nagyban egyszerűsíti a programozó munkáját, hogy nem kell a lefoglalt memóriával foglalkoznia. Ezt a feladatot elvégzi a futtató rendszer, s automatikusan felszabadítja a már nem használt tárterületeket.

- **Objektumorientált:** Az egyszerű adattípusú változók kivételével minden objektum. A java program nem tartalmazhat se globális változókat, se globális eljárásokat. Minden adat, illetve eljárás valamely objektumhoz vagy osztályhoz kötődik.

- **Architektúrafüggetlen és hordozható:** Ahhoz, hogy egy programot bármelyik hardver architektúrán le tudjuk futtatni, kell, hogy legyen egy olyan virtuális ”felület”, amelyet mindegyik hardverre ”ráhúzhatunk”. Ezt a célt szolgálja a java virtuális gép. A java programunkat sem egy hardver architektúrára fordítjuk le, hanem az imént említett virtuális gép nyelvére. A java virtuális gépet kell feltelepíteni az adott architektúrára.

2. ábra Java architektúra

- **Interpretált és dinamikus:** A futtató rendszer a bájtkódot futtatás során interpretálja, ezáltal a Java programok futás közben megváltozhatnak. Futás során a használni kívánt bájtkódra lefordított osztályokat az osztálybetöltő tölti be dinamikusan.

- **Robusztus és biztonságos:** Erősen típusos nyelv. Minden adatnak fordításkor jól definiált típusa van. A nyelvből eltűntek a mutatók, helyettük a referencia típus használható, mely szintén megkönnyíti a programozó munkáját. Szintén segíti a programozót a java szemétgyűjtő módszere, melynek hatására nem kell a lefoglalt memóriákkal foglalkoznia.
  A fordító csak helyesen működő programot készít. Előfordulhat olyan eset is, hogy a betölteni kívánt bájtkód hibás, sérült. Ekkor sem fog a programunk helytelenül működni, mert mielőtt a java futtató rendszere lefuttatja a kódrészletet, leellenőrzi azt.

- **Többszálú:** A Java programok egyszerre több szálon futhatnak. A többszálú programozáshoz nyelvi szinten van biztosítva a kölcsönös kizárás szinkronizált módszerek vagy utasítások révén. Thread osztályt tartalmaz a szálak létrehozásához és szinkronizációjuk megvalósításához.

2.4. JDBC

Manapság egy komolyabb program nagyobb mennyiségű adattal dolgozik és ezt tárolnia is kell valahol. Az adatok tárolása nagyrészt adatbázisok által valósul meg, ezért térnék ki a java adatbáziskezelésére.

A JDBC API szolgáltatásai:

- Összekapcsolódás a relációs adatbázissal
- SQL utasítások végrehajtása
- SQL lekérdezések eredményeinek feldolgozása

A JDBC két része:

1. JDBC alap (Core) API:
    - java.sql csomag
    - az adatbázisok eléréséhez szükséges alapvető osztályok
    - java 1.1 verziótól
2. JDBC standard kiterjesztés (Extension) API:
    - javax.sql csomag
    - további haladó szintű osztályok
    - java 1.4 verziótól

Két oldalról közelíthetjük meg a JDBC-t:

1. Adatbázis oldalról:

   El kell készíteni a meghajtóprogramot, a JDBC specifikációnak megfelelően, egy adott adatbázis számára. A java.sql és/vagy a javax.sql csomagban lévő interfészeket kell leimplementálni. Ezzel az adatbázis-kezelő alkalmazásfejlesztőknek nem kell foglalkozniuk.
2. Alkalmazás oldalról:

   A programozónak az API-t kell megfelelően használnia. A java.sql és/vagy a javax.sql csomagot használja és a használt adatbázis-kezelő meghajtóprogramját a programhoz csatolja.

2.4.1. Adatbázisfüggetlenség

A JDBC egy olyan programozói interfész, amely segítségével olyan alkalmazást tudunk készíteni, ami képes lesz hozzáférni adatbázisokhoz. Egy szabványos eszközrendszert kapunk az adatbázisok felé. Ezáltal nem csak platformfüggetlen, hanem adatbázisfüggetlen alkalmazást is készíthetünk. Vagyis lehetőségünk van adatbázisfüggetlen alkalmazás fejlesztésére. Ez azért csak lehetőség, mivel adatbázist manipuláló SQL utasítások adatbázisonként különbözhetnek. A java ezeket az utasításokat nem figyeli sem fordításkor, sem futtatáskor. Ezek az adott adatbázisnak szóló utasítások.
Ha olyan programot szeretnénk, amely minél több adatbázissal képes kommunikálni, akkor szabványos strukturált adatbázis-lekérdező nyelvet (SQL-92) kell használni, amelyet

mindegyik adatbázis tud értelmezni. A program megírása során én is arra fogok törekedni, hogy minél több adatbázissal működjön majd a program.

2.4.2. Adatbázishoz kapcsolódás folyamata

Elsőnek a JVM betölti a meghajtóprogramot, majd beregisztrálja magát a DriverManager-nél. Ezek után a DriverManager-től kérhetünk majd Connection-t, ami egy munkamenetet reprezentál. A Connection-tól kérhetünk Statement-et, amely segítségével SQL utasításokat hajthatunk végre. Ha SELECT utasítást hajtunk végre, akkor az eredménytáblát a ResultSet segítségével dolgozhatjuk fel.

2.4.2.1. Meghajtóprogram betöltése

Adatbázishoz való kapcsolódás első lépése a meghajtóprogram betöltése. A Driver-t leimplementáló osztályban lefut a statikus inicializáló rész. Ezáltal beregisztrálja magát a DriverManager-hez.
A meghajtóprogramot négyféleképpen tudjuk betölteni:

- Példány létrehozásával

  new < Meghajtóprogram osztályneve>();
- forName metódussal

  Class.forName("<Meghajtóprogram osztályneve >");
- .class attribútummal

  Class c = < Meghajtóprogram osztályneve>.class;
- jdbc.drivers jellemző beállításával

  java -classpath <Meghajtóprogram>:. -Djdbc.drivers=< Meghajtóprogram osztályneve> <Alkalmazás osztályneve>

2.4.2.2. Kapcsolódás az adatbázishoz

Megadjuk az ”adatbázis URL”-t, ami meghatározza, hogy melyik adatbázishoz akarunk kapcsolódni.
URL="jdbc:<alprotokol>://<állomás címe>/<adatbázis név>?
user=<felhasználónév>&password=<jelszó>
[&<egyéb paraméter(ek)>]

Ezután meghívhatjuk a DriverManager getConnection statikus metódusát az URL-el, hogy kérjünk egy Connection-t. Ha nem történt hiba a csatlakozás során, akkor a kapcsolódott alkalmazás kommunikálhat az adatbázissal.

2.4.2.1. Információ a kapcsolatról

Az adatbázisról a DatabaseMetaData interfész segítségével kérhetünk le információkat, melyre a Connection interfész getDatabaseMetaData metódusa szolgál. Így a lekért objektum a DatabaseMetaData interfész egy implementációja. Az információk lekérdezésének formáját 2 részre bonthatjuk. Az egyik rész informál minket, hogy az adatbázis támogat-e valamit, Ekkor a metódusok a support szóval kezdődnek. A másik részhez az úgynevezett ”lekérdező utasítások” tartoznak, melyek neve get szóval kezdődik. Ezek között vannak olyanok is, amelyek az SQL lekérdező utasítások eredményeit is reprezentáló ResultSet interfész egy példányát fogják tartalmazni.

2.4.3. Tranzakciókezelés

Egy tranzakció SQL utasítások végrehajtásából áll, melyek eredményét vagy véglegesítjük az adatbázisban, vagy visszavonjuk minden változtatását, visszaállítva ezzel az adatbázis eredeti állapotát. A véglegesítést a Conection interfész commit metódusával, a visszavonást a Connection interfész rollback metódusával tehetjük meg, mely az adatbázist visszagörgeti az utolsó véglegesített állapotba. Alapértelmezettként, kapcsolat létesítése után minden SQL utasítás véglegesítődik, tehát egy SQL utasítás egy tranzakció. Ha az automatikus nyugtázási módot ki akarjuk kapcsolni, amit a Connection interfész setAutoCommit(false) metódushívással megtehetünk, akkor a programnak magának kell vezérelnie a tranzakciókezelést.

2.4.3.1. Tranzakció izolációs szintek

Manapság az adatbázis-kezelők többfelhasználós rendszerek, ezért előfordulhat, hogy az egyidejűleg futó tranzakciók hatással vannak egymásra. Például az egyik tranzakció olyan értékeket olvas, amit a másik tranzakció módosít, de még nem véglegesített. Ekkor felmerül a kérdés: Milyen adatokat adjon vissza a tranzakció? Ekkor beszélünk tranzakció izolációs szintekről, melyek azt szabályozzák, hogy az adatbázis hogy viselkedjen hasonló problémák hatására.
A Connection interfész öt ilyen tranzakció izolációs szintet definiál:

- TRANZACTION-NONE – nincs tranzakciókezelés
- TRANZACTION-READ-UNCOMMITED – olvasáskor mindig az aktuális értéket kapjuk
- TRANZACTION-READ-COMMITED – olvasáskor csak a véglegesített adatokat kapjuk
- TRANZACTION-REPEATABLE-READ – a tranzakció ideje alatt az általa olvasott értékek más véglegesített tranzakciók esetleges módosító hatása ellenére is mindig megegyeznek a tranzakció kezdetekor érvényben lévő értékekkel
- TRANZACTION-SERIALIZABLE – a tranzakció ideje alatt az általa olvasott értékeket más tranzakciók nem írhatják felül

A fent említett szinteknél az első a legalacsonyabb tranzakció izolációs szint, majd egyre nagyobb szintre térünk. Minél nagyobb szinten vagyunk, annál több ellenőrzést kell elvégezni az adatbázis-kezelő rendszernek a tranzakciók futtatása közben, ezáltal a tranzakciók feldolgozási ideje növekszik.

A tranzakció-izolációs szintet a Connection interfész setTranzactionIzolation metódusával lehet megváltoztatni. Tranzakció közben nem ajánlatos a változtatás, mivel automatikusan véglegesíti az aktuális tranzakciót és csak ezután módosul a tranzakció izoláció szint.

2.4.3.2. SQL utasítások végrehajtása

Az SQL utasításokat három interfész segítségével lehet végrehajtani:

- Statement – egyszerű SQL utasítások végrehajtására használható
- PreparedStatement – bemenő paraméterekkel is rendelkező SQL utasítások végrehajtására használható
- CallableStatement – ki/bemenő paraméterekkel is rendelkező tárolt SQL eljárások végrehajtására használható

2.4.3.2.1. Statement

Connection objektum createStatement metódusával hozható létre Statement objektum. Az SQL utasítást az egyik végrehajtó metódusának kell megadni.

A Statement interfész legfontosabb metódusai:

- executeQuerry – Lekérdező SQL utasítások végrehajtásakor használható. A paraméterben megadott SQL utasítást hajtja végre és annak eredményét eredménytáblában adja vissza.

- executeUpdate – A paraméterként megadott SQL utasítást végrehajtja és az adatbázistábla megváltoztatott sorainak számát adja vissza.
- Execute – Ez a metódus is a paraméterében megadott SQL utasítást hajtja végre, de a két előző metódus általánosításának is tekinthető. Akkor lehet használni, ha az SQL utasítás egyszerre többfajta eredményt is visszaadhat, vagy nem ismert, hogy milyen típusú a visszaadott eredmény. A visszatérési értéke megadja, hogy a végrehajtott SQL utasításnak van-e eredménye. Ha van eredmény, akkor az az executeQuerry metódus meghívásával lekérhető, ha nincs, akkor az executeUpdate metódussal kérhető le hány sorra volt hatással a művelet.

2.4.3.2.2. PreparedStatement

Adatbázis alkalmazásokban gyakran előfordul, hogy egy utasítást többször kell végrehajtani. Adatokat viszünk fel az adatbázisba, módosítjuk egy tábla tulajdonságait az azonosítója ismeretében, úgy hogy ez a művelet az alkalmazásban gyakran előfordul, vagy lekérdezzük az adatbázis adott tábláit egy lista minden elemére. Ezekre és ezekhez hasonló programrészek számára használható.

A legtöbb adatbázis-kezelő rendszer szerencsénkre támogatja a preparált utasításokat, melyeket egyszer kell létrehozni, elemezni és optimalizálni az adatbázisban, majd használni tudjuk újra meg újra az alkalmazásunkban. Ezáltal gyorsabban futnak le az SQL utasítások, mintha csak a Statementet használtuk volna.

A JDBC támogatja a preparált utasításokat a Statement osztály PreparedStatement alosztályával. Tehát a PreparedStatement interfész a Statement kiterjesztettje. Abban különbözik tőle, hogy ennek egy példánya már tartalmaz egy SQL utasítást. A másik különbség pedig az, hogy tartalmazhat bemenő paramétereket is.

A bemenő paramétereket úgy adjuk meg, hogy az SQL utasítás szövegében kérdőjeleket helyezünk el. Minden ilyen kérdőjel egy paraméter. Nem muszáj megadni a paramétert, de ha megadtuk, akkor kötelező legalább egyszer értéket adni neki, mert különben SQLException kivételt kapunk. Értéket a set<Típusnév> metódussal adhatunk, ahol a Típusnév a paraméter típusával kell, hogy megegyezzen. Null érték megadására a setNull metódus szolgál.

Létrehozása a fennálló kapcsolatot képviselő Connection objektum prepareStatement metódusával történik. Pl: prepareStatement(”UPDATE <táblanév> SET <tulajdonság1>=? WHERE <tulajdonság2>=?”). A PreparedStatement interfész a Statement-ből van származtatva.

2.4.3.2.3. CallableStatement

Ez is a Connection interfészből hozható létre a prepareCall metódus segítségével. Itt is a metódus hívásával egy időben adjuk át kérdőjelekkel felparaméterezve.

A CallableStatement interfész a PreparedStatement kiterjesztettje. Tárolt SQL eljárásokat hívhatunk meg vele. Egy tárolt eljárás hívása a bemenő paramétereken kívül kimenő paramétereket is használhat. Egy paraméter egyszerre lehet kimenő és bemenő paraméter is.

Az adatbázisban található eljárás meghívása a következő szintaxis segítségével történik:

- {call <eljárásnév> [(?,?,…)]} – visszatérési érték nélküli tárolt eljárás hívása
- { ?= call <eljárásnév>[(?, ?, …)]} – visszatérési értékkel rendelkező tárolt eljárás hívása

Végrehajtás előtt a bemenő paraméterek aktuális értékét be kell állítani és minden kimenő paraméternek meg kell adni a típusát a registerOutParameter metódus segítségével. Egy kimenő paraméter értékét a get<Típusnév> alakú metódusokkal lehet lekérdezni.

2.4.3.3. Eredménytáblák feldolgozása

A lekérdező utasítások eredményeinek feldolgozása eredménytábla segítségével történik. Ez a ResultSet interfész egy implementációjának példánya.

Az eredményt soronként tudjuk lekérdezni. Kezdetben a kurzor az első sor elé mutat. A régi JDBC verzióban csak sorról sorra történhetett a bejárás, de a második verziótól kezdve már lehetőség van a sorok tetszőleges sorrendben történő feldolgozására is.

A következő metódusokkal mozgathatjuk a kurzort:

- next, previous: Előre, illetve hátra mozgatja eggyel a kurzort, de ha nem sorra mutat, akkor hamis értékkel tér vissza.
- last, first: Az első, illetve az utolsó sorra mozgatja a kurzort és hamis értékkel tér vissza, ha nincs sor az eredménytáblában.
- beforeFirst, afterLast: Az első sor elé, illetve az utolsó sor után mozgatja a kurzort.
- absolute, relative: Az absolute metódusnál a paraméterben megadott sorra fog mutatni a kurzor. A relative-nál pedig annyi sorral mozog előre, illetve visszafelé a sorokon, amennyit a paraméterben megadtunk. Nem létező sorra való hivatkozás esetén ez is hamis értékkel tér vissza.

Eredménytábla tulajdonságainak lehetséges beállításai:

- kurzor:

    - TYPE-FORWARD-ONLY: Az eredménytábla kurzora csak előrefelé mozoghat.
    - TYPE-SCROLL-INSENSITIVE: Az eredménytábla kurzora mindkét irányba mozoghat, de nem érzékeli a mások által végzett esetleges módosításokat.
    - TYPE-SCROLL-SENSITIVE: Az eredménytábla kurzora mindkét irányba mozoghat, és a mások által elvégzett módosításokat is érzékeli.
- párhuzamosság:
    - CONCUR-READ-ONLY: Az eredménytáblát csak olvashatjuk, tartalmát nem változtathatjuk meg.
    - CONCUR-UPDATEBLE: Az eredménytábla tartalma megváltoztatható, de ekkor a lekérdezésben nem szerepelhet join és group by kifejezés, viszont szükséges az elsődleges kulcs megléte.
- tarthatóság:
    - HOLD_CURSORS_OVER_COMMIT: Az eredménytábla nem záródik be az adatbázis esetleges véglegesítése esetén.
    - CLOSE_CURSORS_AT_COMMIT: Az eredménytábla bezáródik az adatbázis esetleges véglegesítése esetén.

A fentebb említett beállításokat a Statementek létrehozásakor lehet megadni paraméterben.

Pl.: Statement stmt = conn.createStatement(<kurzor>, <párhuzamosság>, <tarthatóság>);

Majd a Statementeken keresztül tudunk lekérni adatokat az adatbázisból.

Pl.:ResultSet rs=stmt.executeQuery(”<lekérdezés szövege>”);

Ezután előállt az eredménytáblánk. Ezt egy egyszerű ciklus segítségével fel tudjuk dolgozni.

```
while (rs.next()) {
    //a kurzor az eredménytábla egy adott sorára mutat
    rs.get<Típusnév>(<oszlopindex vagy oszlopnév>);//az adott oszlopban található érték lekérdezése.
    rs.update<Típusnév>(<oszlopindex vagy oszlopnév>);//az adott oszlopban található érték módosítása.
}
rs.close();
```

Az eredménytáblát a close metódussal lehet lezárni, de akkor is lezárul, ha csak a Statementet zárjuk le.

2.4.4. Hibakezelés

Elhagyhatatlan dolog a hibakezelés. Ha az adatbázis kapcsolat során bármilyen hiba történne, akkor egy SQLException kivétel fog kiváltódni. Ez a kivétel a fellépett hibákról a következő információkat tartalmazza:

- a hiba szövegét, melyet a getMessage metódussal lehet lekérdezni,
- SQLstate szöveget, az X/OPEN SQLstate szabvány szerint,
- a hiba kódját, amely meghajtóprogram függő és rendszerint az adatbázis által visszaadott hibakódnak felel meg,
- hivatkozást a következő SQLException-ra. Ezzel lehet az aktuális hibaüzenetekhez hozzárendelt esetleges kiegészítő információkat kezelni.

Az előforduló esetleges figyelmeztetések lekezelését az SQLWarning kivételosztály végzi. Ilyen kivételek nem szakítják meg a program futását, mert a megfelelő metódusok elkapják és az aktuális objektumhoz láncolják azokat. A fellépő figyelmeztetéseket a getWarnings metódussal lehet lekérdezni, míg a clearWarnings metódussal lehet törölni. Lekérdezéskor mindig az első figyelmeztető üzenetet kapjuk, ehhez láncba van felfűzve a többi üzenet.

3. Követelmények meghatározása

Elsőnek a programmal szemben támasztott követelmények kerülnek meghatározásra. A programozó és a megrendelő elbeszélgetnek, s a megrendelő leírja, hogy milyen programot szeretne.
Fontosabb szempontok a programozó számára:

- kiknek készül a program, mennyien használják majd,
- hol fog működni, milyen hardver és szoftver környezetben,
- mire szeretné használni, mennyire fontos a gyorsaság.

3.1 Célok meghatározása

Olyan könyvtári nyilvántartó alkalmazás fejlesztése, amely egy kisebb könyvtár könyvkölcsönzéssel kapcsolatos feladatait látja el. A program képes az olvasók, könyvtárosok és könyvek kezelésére, felvitelére és módosítására.

3.2 Funkciók meghatározása

- Bárki számára elérhető funkciók:
    - könyvkeresés: A könyvtárban lévő könyvek között keresünk az adatbázisban. Arra az információra is szükség van, hogy a talált könyv kölcsönözhető vagy ki van kölcsönözve valaki által.
- Olvasók számára elérhető funkciók:
    - könyvkeresés: Az előbb említett módon.
    - kölcsönzött könyvek mutatása: Belépést követően az olvasó legyen informálva arról, hogy milyen könyveket kölcsönzött ki és mikor kell visszavinni azt a könyvtárba.
    - adatok módosítása: Az olvasó képes legyen bizonyos adatainak módosítására.
- Könyvtárosok számára elérhető funkciók:
    - könyvkeresés: Szintén a fent taglalt módon.

- könyvkölcsönzés: Adott olvasó olvasószám alapján képes legyen könyvet kölcsönözni, ha a kölcsönözni kívánt könyvnek van kölcsönözhető példánya és az olvasónak van érvényes olvasójegye.
- könyv visszavétele: Az olvasó visszahoz egy, vagy több könyvet. Ha ez határidőn túl történik, akkor befizeti a késedelmi díjat, amit úgy kell meghatározni, hogy kiszámolja az eltelt időt a lejárati nap és a visszahozás dátuma között, és megszorozza az aktuális napi késedelmi díjjal. Ezek után a program bejegyzi az adatbázisba a visszahozás tényét.
- könyv felvitele: Könyv felvitele az adatbázisba.
- könyv módosítása: Adott könyvet, adott ISBN szám alapján lehessen módosítani.
- olvasó keresése: Az adatbázisban szereplő minden olvasó között lehessen keresni.
- olvasó felvitele: Olvasó felvitele az adatbázisba.
- olvasó módosítása: Olvasó adatainak módosítása olvasószám alapján.
- könyvtáros felvitele: Könyvtáros felvitele az adatbázisba.
- adatok módosítása: A könyvtáros is tudjon az adatain módosítani.

4. Tervezés

4.1 Adatbázis megtervezése

Elsőnek összeírom azokat a tulajdonságokat, amelyek a program számára fontosak. Ha ez megvan, elkezdem ezeket csoportosítani. A feladat elemzése után 15 táblát kapok. Ezeket a következő jelrendszer használatával mutatnám be. A tábla tulajdonságainak neveit kiemelem félkövér betűvel, ha elsődleges kulcs, akkor alá is húzom. A típusát zárójelben melléírom. Ha értéket mindenképp kell kapnia (not null), akkor aláhúzom a típusát.

4.1.1. A Book tábla

Egy sora, egy könyvet reprezentál. Ekkor a könyv még csak logikailag létezik a könyvtárban. Nincsenek úgynevezett példányai.

A tábla tulajdonságai:

- **BookID**(integer): A Book tábla elsődleges kulcsa.
- **ISBN**(varchar(20)): A könyv ISBN száma, amely egyedi egy kiadására és változatára nézve.
- **Title**(varchar(255)): A könyv címe.
- **CategoryID**(integer): A könyv kategóriába való sorolásához kell megadni.
- **PublisherID**(integer): Kiadó azonosítója.
- **PublishDate**(integer): A kiadás dátuma.
- **CallNumber**(varchar(10)): A könyv raktári jelzete. A könyv könnyebb megkeresését segíti.
- **Barcode**(varchar(20)): A könyv vonalkódja.
- **Price**(integer):A könyv vételi ára.
- **PageNumber**(integer): A könyv oldalainak a száma.
- **SpineSize**(integer): A könyv gerincmérete centiméterben.

4.1.2. Az Author tábla

Ebben találhatóak a könyvek szerzői.

A tábla tulajdonságai:

- **AuthorID**(integer): Author tábla elsődleges kulcsa.
- **Name**(varchar(255)): A szerző neve.
- **Born**(integer): A szerző születési éve.

4.1.3. Az AuthorRelationship tábla

A szerzők és a könyvek közötti több-több kapcsolatot reprezentálja. Hatására egy szerzőnek lehet több könyve és egy könyvnek több szerzője.

A tábla tulajdonságai:

- **AuthorRelationshipID**(integer): AuthorRelationship tábla elsődleges kulcsa.
- **AuthorID**(integer): A szerző azonosítására szolgál.
- **BookID**(integer): A könyv azonosítására szolgál.

4.1.4. Category tábla

A könyvek kategóriáját reprezentálja, fa hierarchiában. Minden kategóriának lehet alkategóriája.

A tábla tulajdonságai:

- **CategoryID**(integer): Category tábla elsődleges kulcsa.
- **Name**(varchar(255)): Kategória neve.
- **CategoryLevel**(varchar(255)): Kategória szint. Arra szolgál, hogy a kategóriát elhelyezzük a fa hierarchiában.

4.1.5. A Publisher tábla

A könyvek kiadóit tárolja.

A tábla tulajdonságai:

- **PublisherID**(integer): Publisher tábla elsődleges kulcsa.
- **Name**(varchar(255)): Kiadó neve.
- **LocalityID**(integer): Kiadó székhelyének azonosítása.

4.1.6. Az Address tábla

Címek tárolására.

A tábla tulajdonságai:

- **AddressID**(integer): Address tábla elsődleges kulcsa.
- **PostalCode**(integer): A címhez tartozó irányítószám.
- **Street**(varchar(255)): Utca és házszám.

4.1.7. Az Availability tábla

Olvasók és könyvtárosok elérhetőségeinek leírására használható.

A tábla tulajdonságai:

- **AvailabilityID**(integer): Availability tábla elsődleges kulcsa.
- **AddressID**(integer): Cím azonosítója.
- **PhoneNumber**(varchar(255)): Telefonszám.
- **E_mail**(varchar(255)): E-mail cím.

4.1.8. A Person tábla

Személyek leírására szolgál. Ebből lesz származtatva a könyvtáros és az olvasó is.

A tábla tulajdonságai:

- **UserID**(integer): Person tábla elsődleges kulcsa. Ha olvasó származtatja, akkor ez az olvasószám.
- **Name**(varchar(255)): Személy neve.
- **IdentityCardNumber**(varchar(255)): Személyi igazolvány száma.
- **BirthPlace**(varchar(255)): Születési hely.
- **BirthDate**(date): Születési idő.
- **Password**(varchar(255)): Jelszó.

4.1.9. Az Users tábla

Olvasó leírására szolgál.

A tábla tulajdonságai:

- **UserID**(integer): Users tábla elsődleges kulcsa és egyben az olvasószám.

- **AvailabilityID**(integer): Elérhetőség azonosítója.
- **MembershipBeginDate**(date): Olvasó beiratkozási dátuma.
- **MembershipEndDate**(date): Olvasó tagságának a lejárati dátuma.
- **Job**(varchar(255)): Olvasó foglalkozása.
- **Workplace**(varchar(255)): Olvasó munkahelye.

4.1.10. A Librarians tábla

Könyvtáros leírására szolgál.

A tábla tulajdonságai:

- **UserID**(integer): Librarians elsődleges kulcsa.
- **AvailabilityID**(integer): Elérhetőség azonosítására.
- **BeginEmployment**(date): Könyvtáros munkaviszonyának kezdete.

4.1.11. A Lending tábla

Ez a kölcsönzés tábla. Az olvasó, ha kikölcsönöz egy könyvet, akkor itt hozzáadunk a táblához egy sort, ha visszahozza a könyvet, akkor törlésre kerül az adott sor. Mindig az aktuálisan kölcsönzött könyveket tartalmazza.

A tábla tulajdonságai:

- **LendingID**(integer): Lending tábla elsődleges kulcsa.
- **InventoryID**(integer): Könyv példányának azonosítására.
- **UserID**(integer): Olvasó azonosítása, olvasószám alapján.
- **LendingDate**(date): Kölcsönzés dátuma.
- **LendingDeadline**(date): Kölcsönzési határidő.

4.1.12. A LendingLog tábla

Kölcsönzési log. Itt tárolódnak a kikölcsönzött, de már visszavitt könyvek.

A tábla tulajdonságai:

- **LendingLogID**(integer): LendingLog tábla elsődleges kulcsa.
- **UserID**(integer): A kölcsönző olvasó olvasószáma.
- **InventoryID**(integer): Könyv példány azonosítására.
- **LendingDate**(date): Kölcsönzés dátuma.

- **BringBackDate**(date): Könyv visszavitelének dátuma.
- **Fine**(integer): Büntetés nagysága abban az esetben, ha az olvasó nem vitte vissza a könyvet időben.

4.1.13. Az Inventory tábla

Raktárkészlet. Egy könyv példányainak azonosítására használható.

A tábla tulajdonságai:

- **InventoryID**(integer): Inventory tábla elsődleges kulcsa.
- **BookID**(integer): Könyv azonosítására.
- **AllowLending**(integer): Kikölcsönzés engedélyezett-e vagy nem. Értéke 1, ha engedélyezett a könyv adott példányának a kölcsönzése. Egyébként 0, ha csak a könyvtárban olvasható.

4.1.14. A Locality tábla

A táblában helységnevek találhatóak. Közvetlen kapcsolatban állnak a Publisher táblával és közvetett kapcsolatban az Users és a Librarians táblákkal is.

A tábla tulajdonságai:

- **LocalityID**(integer): Locality tábla elsődleges kulcsa.
- **Name**(varchar(255)): Helység neve.

4.1.15. A PostalCode tábla

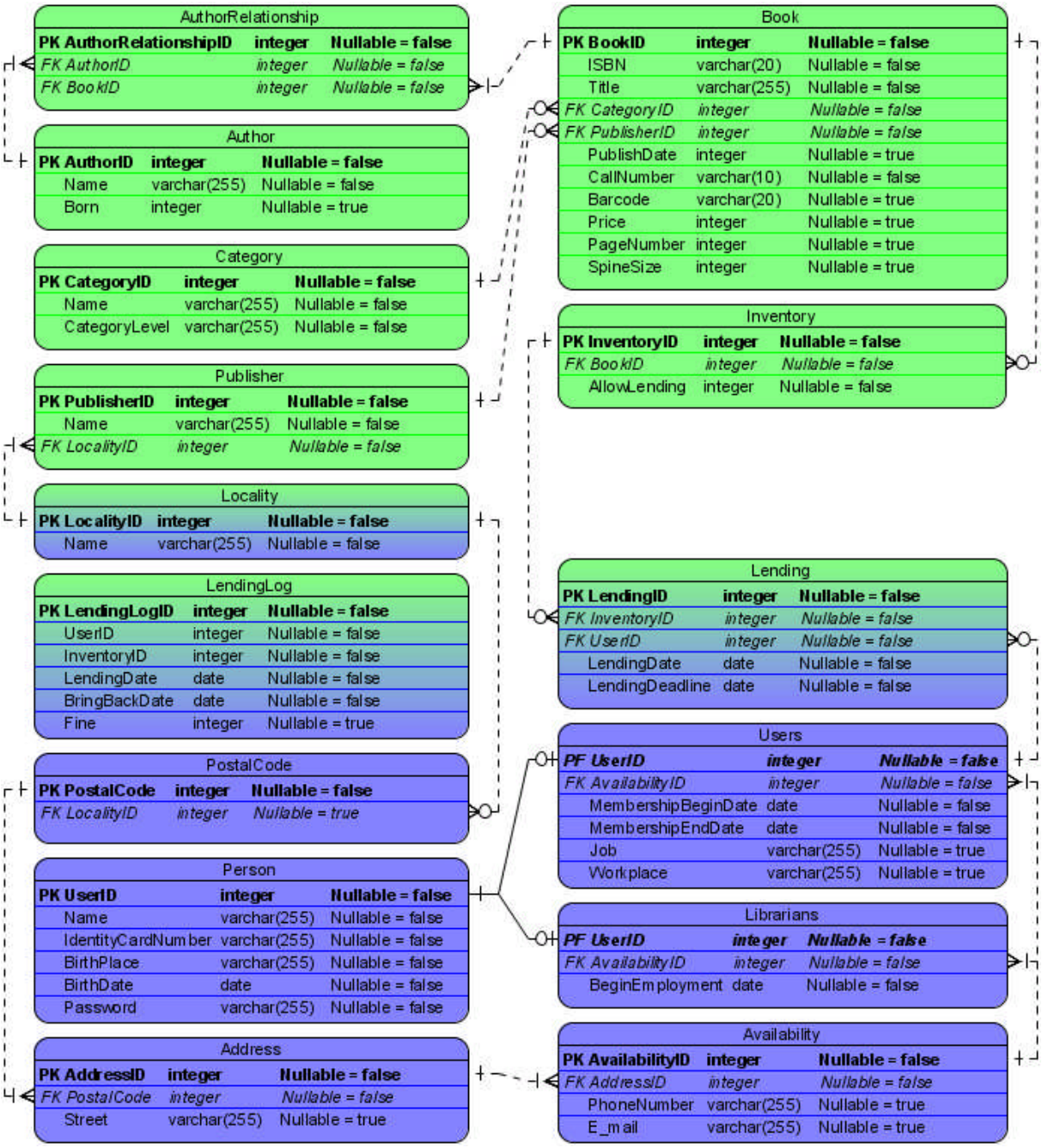
Irányítószámok azonosítására.

A tábla tulajdonságai:

- **PostalCode**(integer): PostalCode tábla elsődleges kulcsa, ami megegyezik az irányítószámmal.
- **LocalityID**(integer): Helység azonosítója.

4.1.16. Segédtáblák

Minden táblának van egy saját táblája, melyek nevei ”seq_<táblanév>” formában írhatóak le és egy next_hi oszloppal rendelkeznek.
A táblákra azért volt szükségem, mert nem minden adatbázis-kezelő támogatja az elsődleges kulcsgenerálást. A next_hi tulajdonsága tartalmazza a hozzá tartozó tábla elsődleges kulcsának következő értékét. Users és a Librarians tábláknak nincsen segédtáblája, mivel a táblák elsődleges kulcsai megegyeznek a Person tábla elsődleges kulcsával.



3. ábra Adatbázis séma

A Book, Author, Category, Publisher táblák egy könyv tulajdonságait írják le. Az Address, PostalCode, Locality, Availability, Person, Users, Librarians táblák olvasóhoz vagy könyvtároshoz tartozó táblák. A Lending táblában az éppen kikölcsönzött könyvek találhatóak. A LendingLog táblában pedig a kölcsönzésből visszahozott könyvek lesznek.

4.2. Interfész tervezés

Ebben a részben egy fontos interfészt emelnék ki. Ez az interfész az adatbázis és a felület között helyezkedik el. Adatbázis oldalon az interfész metódusait kell leimplementálni, míg a felület felől csak a megfelelő metódusokat kell meghívni a megfelelő paraméterekkel. Ezáltal két részre bomlik a feladat. Egyrészt az adatbázis és az interfész közötti rész leprogramozása, melyre a JDBC-t használtam. Másrészt a felület kialakítása, melyet a NetBeans segítségével valósítottam meg.

4. ábra Könyvtári nyilvántartói program architektúrája

```
public interface Adatbazis {

   //Adatbázis kapcsolat létrehozása.
   public boolean connecting();

   //Adatbázis kapcsolat lezárása.
   public boolean closeConnecting();

   //A bejelentkezett könyvtártag, olvasó-e?
   public boolean isUser();

   //A bejelentkezett könyvtártag, könyvtáros-e?
   public boolean isLibrarian();

   //Bejelentkezés.
   public int login(String name, String password)throws DatabaseException;
```

//A bejelentkezett neve.

public String **getName**();

//A bejelentkezett azonosítójának lekérdezése.

public int **getID**();

//Könyvtártag kijelentkeztetése.

public void **logout**();

//Feltételek alapján könyvet keresünk.

public List **getBookFind**(String title, String author, String category, String callNumber, String inventoryID, String ISBN, String spineSize, String pageNumber, String barcode, String price, String publisherName, String publisherPlace, String editionDate);

//Könyv részleteit kapjuk meg.

public List **getBookDetails**(String ISBN);

//Kölcsönözhető az ISBN számmal azonosított könyv?

public boolean **getAllowLending**(String ISBN);

//Könyv felvitele az adatbázisba.

public void **setBookInsert**(String title, String author, int category, String callNumber, String copy, String ISBN, String spineSize, String pageNumber, String barcode, String price, String publisherName, String publisherPlace, String editionDate)throws DatabaseException;

//Könyv adatainak módosítása.

public void **setBookUpdate**(String title, String author, String callNumber, String copy, String ISBN, String spineSize, String pageNumber, String barcode, String price, String publisherName, String publisherPlace, String editionDate)throws DatabaseException;

//Könyvtáros keres olvasókat.

public List **getLibrarianUserFind**(boolean user,String name,int userID, String address, String locality, String street, String birthPlace, String birthDate, String identityCardNumber, String membershipBeginDate, String phoneNumber, String email, String job, String workplace) throws DatabaseException;

//Könyvtáros felvisz egy olvasót vagy könyvtárost az adatbázisba.

public void **setLibrarianUserInsert**(boolean user,String name, String address, String locality, String street, String birthPlace, String birthDate, String identityCardNumber, String phoneNumber, String email, String job, String workplace) throws DatabaseException ;

//Könyvtáros az olvasó adatait módosítja.

public void **setLibrarianUserUpdate**(String name, int userID, String address, String locality, String street, String birthPlace, String birthDate, String identityCardNumber, String membershipEndDate, String phoneNumber, String email, String job, String workplace) throws DatabaseException ;

//Megkapjuk az olvasó részleteit.

public List **getLibrarianUserDetails**(boolean read,int userID)throws DatabaseException;

//Kik kölcsönöztek az adott könyvből.

public List **getWhoBorrow**(String ISBN);

//Olvasó adatainak módosítása.

public void **setUserUpdateUser**(int userID, String phoneNumber, String email, String job, String workplace, String password)throws DatabaseException;

//Olvasó kikölcsönzött könyveit adja vissza.

public List **getUserBorrowedBooks**(int userID)throws DatabaseException;

//Könyv részleteit adja vissza.

public List **getBookDeatils**(int inventoryID);

//Olvasó kiegészítése.
public List **getUserComplete**(String name,String userID);

//Könyv kiegészítése.
public List **getBookComplete**(String title,String ISBN);

//Könyv kölcsönzése.
public void **setBorrow**(int userID,String ISBN)throws DatabaseException;

//Az adott könyv késedelmi díja.
int **getFine**(int inventoryID);

//Könyv visszahozása.
public void **setBringBack**(int UserID, int inventoryID)throws DatabaseException;

//Meghosszabbítja az adott könyv kölcsönzési idejét.
void **setBookLengthen**(String inventoryID)throws DatabaseException;;

//Visszaadja a könyv kölcsönözhető vagy nem kölcsönözhető példányait.
public List **getCopy**(String ISBN,boolean allowLending);

//Szerzőket kiegészít.
List **getAuthorComplete**(String name);

//Kiadókat kiegészít.
List **getPublisherComplete**(String name);

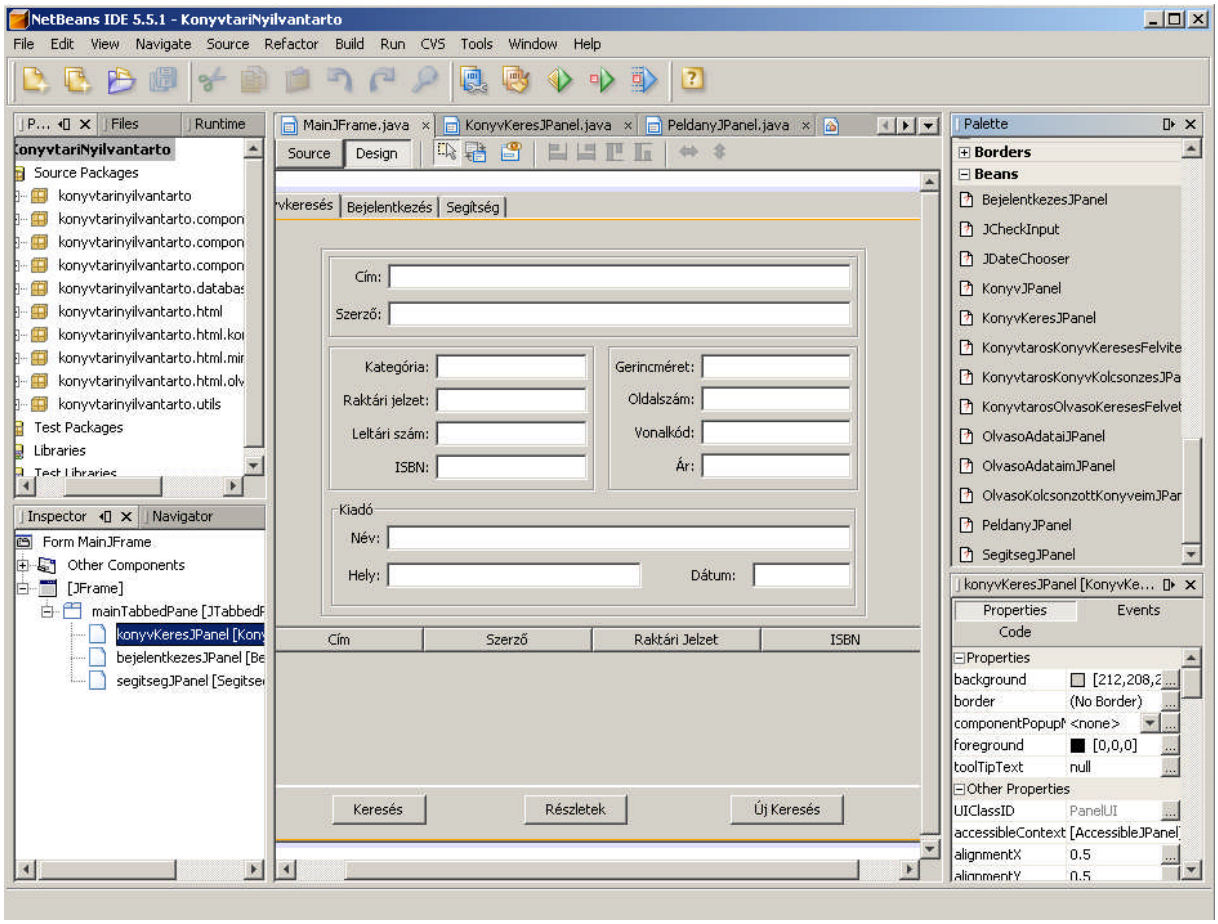
//Kategóriákat adja vissza.
public List **getCategory**();

//Kategória hozzáadása.
int **addCategory**(String level, String name)throws DatabaseException;

//Kategória törlése.
void **removeCategory**(int ID)throws DatabaseException;

4.3. Felületkialakítás

A felület kialakításához a NetBeans IDE 5.5.1 integrált fejlesztői környezetet használtam. Felülettervezőjével egyszerűbben és gyorsabban lehet dolgozni, mivel a tervezés során azt látom, amit futás során szeretnék. A felületet a Free Design layout kezeli, mely a Netbeans saját felületkezelője.



5. ábra NetBeans felületszerkesztője

Az 5. ábrán látható paneleket használtam a legtöbbet az alkalmazás felülettervezése során. Középen látható a Design terület, mely segít a grafikus felület megtervezésében. Mellette található jobb oldalon a Palette panel. Itt azon komponensek találhatóak, melyeket a ”fogd és vidd” (”drag and drop”) technikával tudunk ráhúzni a Design területre. Ezután méretezhetjük, pozícionálhatjuk tetszésünk szerint. Alatta található a Properties panel, mellyel a különböző panelekben található elemek tulajdonságait tekinthetjük meg. Bizonyos tulajdonságokat módosíthatunk is. Bal oldalon fent láthatjuk a Project panelt, itt található a projekttel

kapcsolatos összes fájl. Alatta az Inspector panel, mely a Design területen található komponenseket mutatja fa struktúrában. Segítségével szintén ki lehet jelölni komponenseket.
A felületet komponensek alkalmazásával ”illesztem” össze alap komponensekből, amelyek a java.swing csomag objektumai, egyre nagyobb komponenseket hozva ezzel létre.
Próbáltam arra törekedni, hogy egyértelmű és egyszerű legyen a kezelése, valamint a logikailag összetartozó komponensek közel legyenek egymáshoz, s így a funkciók megvalósítása minél egyszerűbb és könnyen elsajátítható legyen.
Majdnem minden eseményt – nyomógombra való kattintás, billentyű leütés – a főosztályban található metódusok segítségével kezelek le. Itt hívódnak meg az Adatbazis interfész metódusai a megfelelő paraméterekkel.

5. Alkalmazásfejlesztés

A szoftver elkészítésének azon folyamatát, mely a felmerülő problémától az elkészített megoldáshoz vezet, szoftverfejlesztésnek nevezzük. Adott egy ”probléma”, melyet a számítógép nyelvén próbálunk megoldani.

A fejlesztés folyamata az evolúciós modellhez hasonlít a legnagyobb mértékben, melynek lényege, hogy a specifikációt nem kell teljes mértékben megadni, mivel az a fejlesztés folyamán változhat, változtatható. Ezzel én is így voltam. Először kigondoltam, hogy mit is szeretnék csinálni, és hogy ezt hogyan tudnám megvalósítani. Később a fejlesztés folyamán módosítgattam, csiszolgattam az alkalmazás különböző részeit.

A felület és az adatbázis tervezése párhuzamosan történt. Elsőként az adatbázist terveztem meg, majd a felületet, de eközben módosítgattam az adatbázis tábláit, tulajdonságait is, bár ezek már csak csekélyebb módosítások voltak.

Ezek után következett az interfész, ami összekapcsolja a felületet az adatbázissal. Majd miután úgy gondoltam, hogy megfelelő lesz, elkezdtem megvalósítani az osztályokat, metódusokat. Itt többször volt, hogy akadályba ütköztem. Olyan probléma tárult elém, amelyet a felület kialakítása közben nem vettem figyelembe.

Ilyen volt például, hogy amikor a könyvtáros módosítani szerette volna a könyvtár tagok adatait, akkor a jelszó értékét nem tudta volna megváltoztatni, mivel ezt a beviteli mezőt a felület tervezése közben nem adtam meg. Mivel már nem volt hely, hogy elhelyezzek egy újabb beviteli mezőt, ezért ezt egy dialógusablak segítségével oldottam meg a felhasználói dokumentációban leírt módon.

A másik akadály, melyet a felület tervezése során nem vettem figyelembe, az volt, hogy a könyv felvitelénél, illetve módosításánál nem gondoltam a könyv példányaira. Ennek a megoldását is leírtam a felhasználói dokumentáció megfelelő részénél.

Az implementálás folyamán a tesztelés is jelen volt. Kisebb-nagyobb programegységek után leteszteltem, hogy az adott programrész azt a gondolatmenetet követi-e, amit én szeretnék.

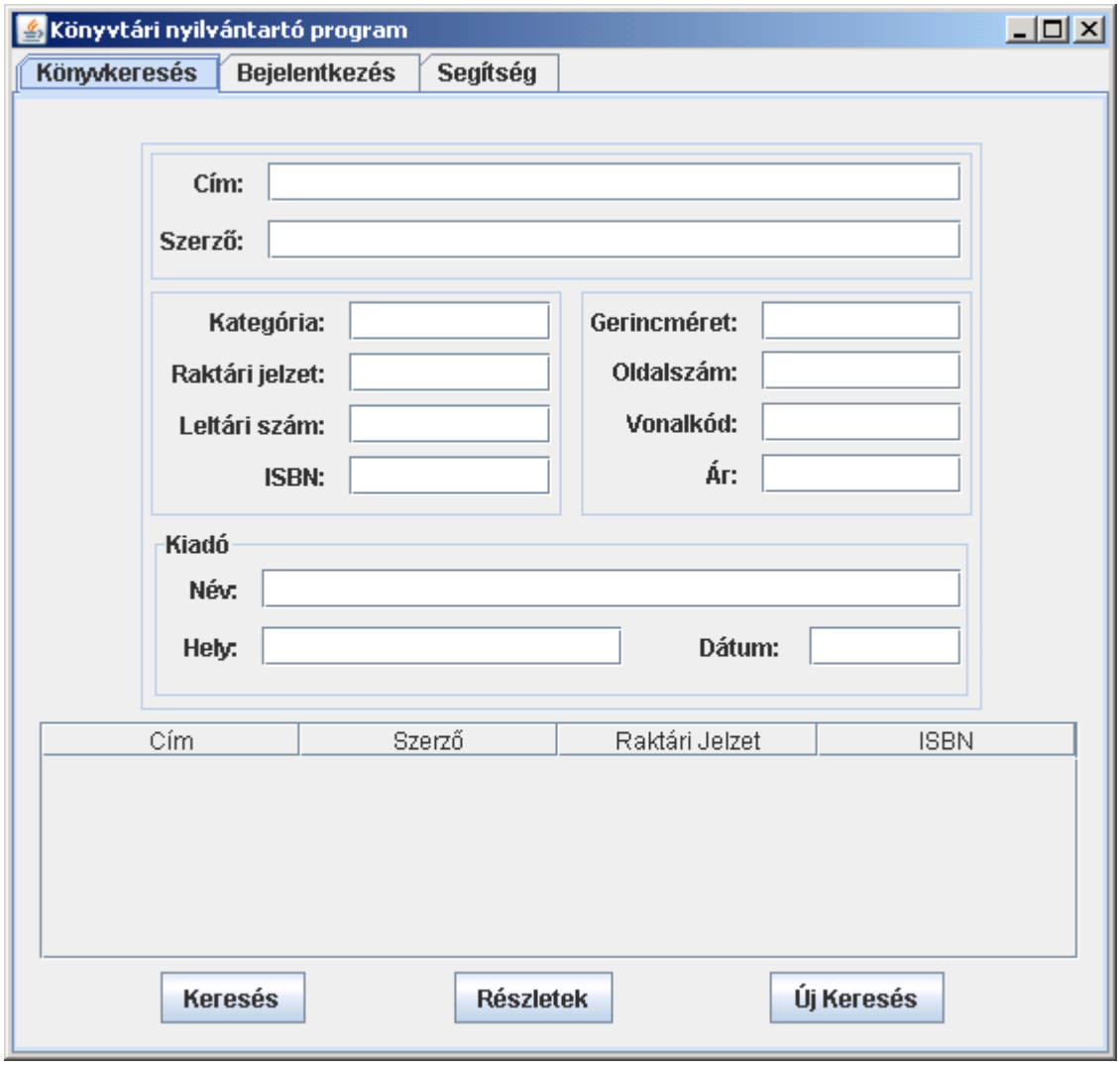
Az utolsó nagy tesztnél több adatbázison is teszteltem az alkalmazást, úgy mint Derby, MSSQL, MySQL, Oracle.

6. Felhasználói dokumentáció

6.1. Mindenki számára

6.1.1. Könyv keresése

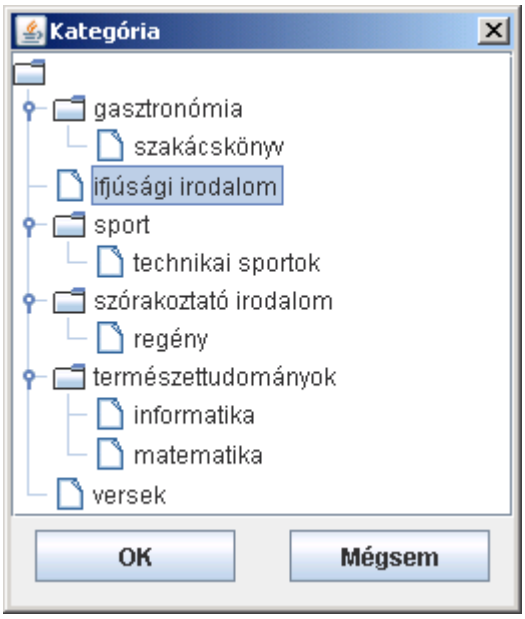
Az alkalmazás elindítása után a keresési felület köszönt minket.



6. ábra Keresés a könyvek között

A könyv minden tulajdonságára rá tudunk keresni, ami az adatbázisban is megtalálható. A cím, szerző, raktári jelzet, ISBN, vonalkód, kiadó név, kiadás helye mezőkben megengedett a

DOS-ban megszokott jelölések használata. A csillag akármennyi, a kérdőjel egy karaktert helyettesít. Használhatunk SQL konverziót is, itt a százalék jel akárhány karaktert helyettesít, az aláhúzás jel pedig csak egy karaktert. A gerincméretre, oldalszámra, árra és a kiadás dátumára is rá lehet keresni. Lehetőség van arra is, hogy egy adott értéknél kisebb vagy nagyobb értékekre keressünk rá. Ekkor az érték elé kisebb, vagy nagyobb jelet kell raknunk, ha az egyezést is megengedjük, akkor az egyenlőségjelet is kitehetjük.
Kategória megadásánál adott kategóriára, és az azon belüli alkategóriákra tudunk rákeresni. Kategóriát megadni úgy tudunk, hogy a mezőben lenyomunk egy billentyűt. Ekkor előugrik egy ablak, ahol ki tudjuk választani a kívánt kategóriát, ami majd megjelenik a beviteli mezőn.

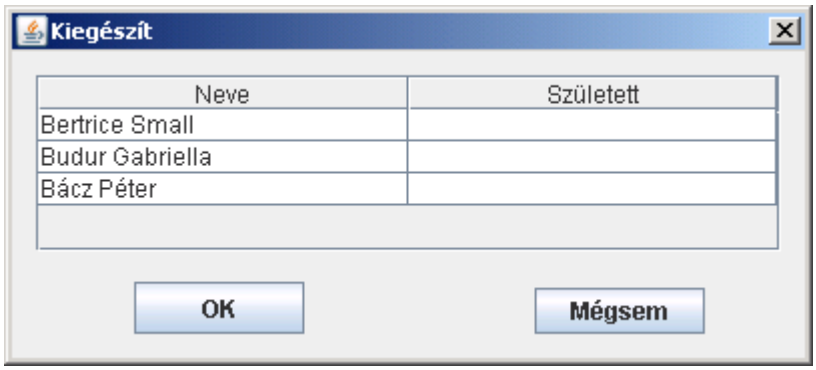


7. ábra Kategória kiválasztására szolgáló ablak

A kiadó és a szerzők megadása is egyszerű. Pár karakter begépelése után csak le kell nyomni az enter billentyűt.
A szerző mezőben enter lenyomása után a következő 3 lehetőség valamelyike történik:

- Ha az adott karakterekkel nem kezdődik az adatbázisban található szerzők nevének egyike sem, akkor a beviteli mezőben az utolsó karakter el fog tűnni. Vagyis a szó hossza csökken eggyel.
- Ha csak egy egyezés volt, akkor annak a szerzőnek a neve íródik bele a beviteli mezőbe, amelyikkel az egyezés történt.

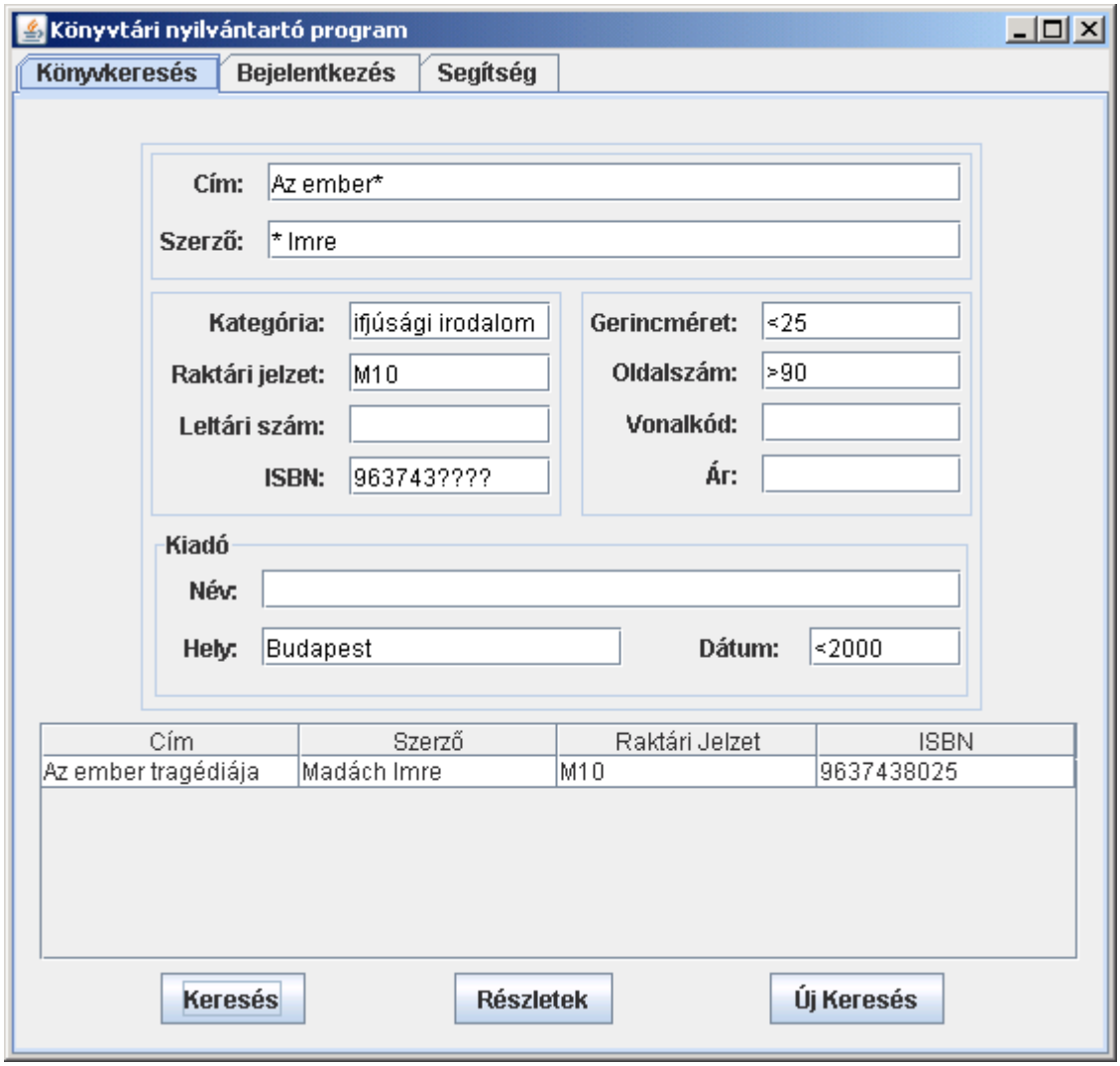
- Ha több egyezés is van, akkor előbukkan egy ablak. Itt lehet választani a szerzők közül.



8. ábra Szerző nevét kiegészítő ablak

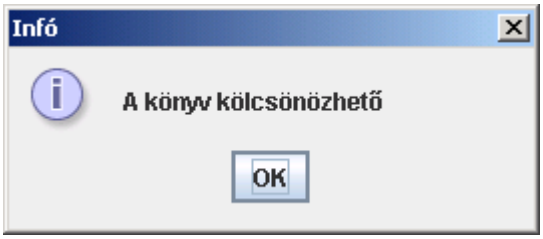
A kiadó nevénél is így működik a kiegészítés.

9. ábra Kiadó nevét kiegészítő ablak

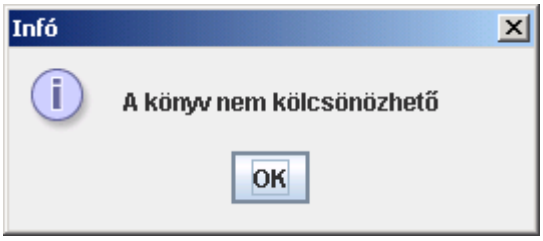


10. ábra A keresés eredménye, keresési feltételek alapján

Ha eredményes volt a keresés, akkor megnézhetjük, hogy kölcsönözhető-e a könyv, van-e kölcsönözhető példány a könyvből. Kattintsunk jobb egérgombbal az adott sorra, majd a ”Kölcsönözhető?” felbukkanó menüre.



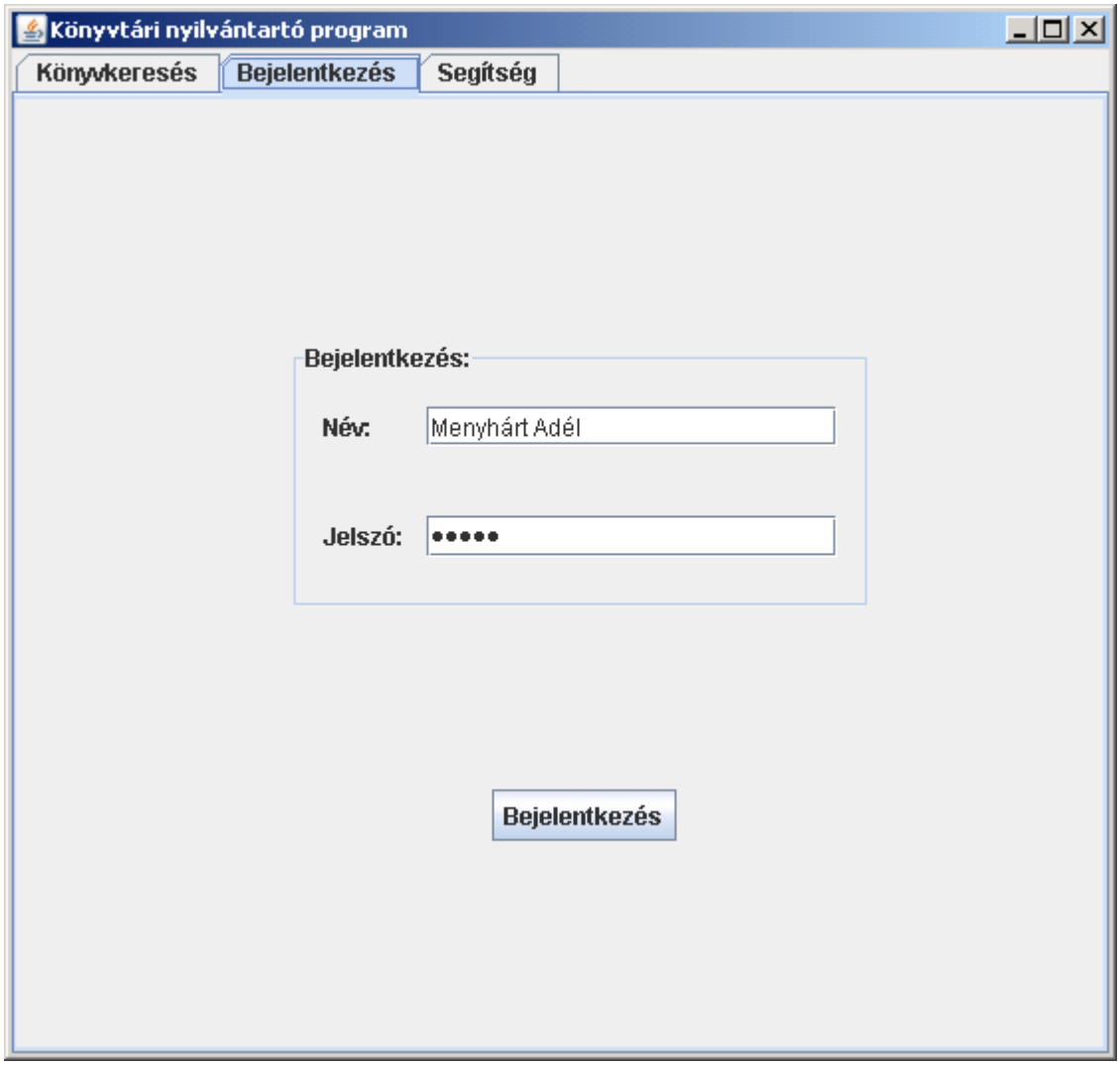
11. ábra Ha a keresett könyv kölcsönözhető



12. ábra Ha a keresett könyv nem kölcsönözhető

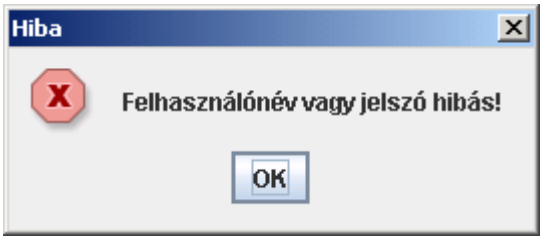
6.1.2. Bejelentkezés

Bejelentkezés előtt az alábbi panellel találkozunk.



13. ábra Bejelentkezési felület

Olvasóknak és könyvtárosoknak is itt kell bejelentkezni. Hibás bejelentkezés esetén hibaüzenetet kapunk.



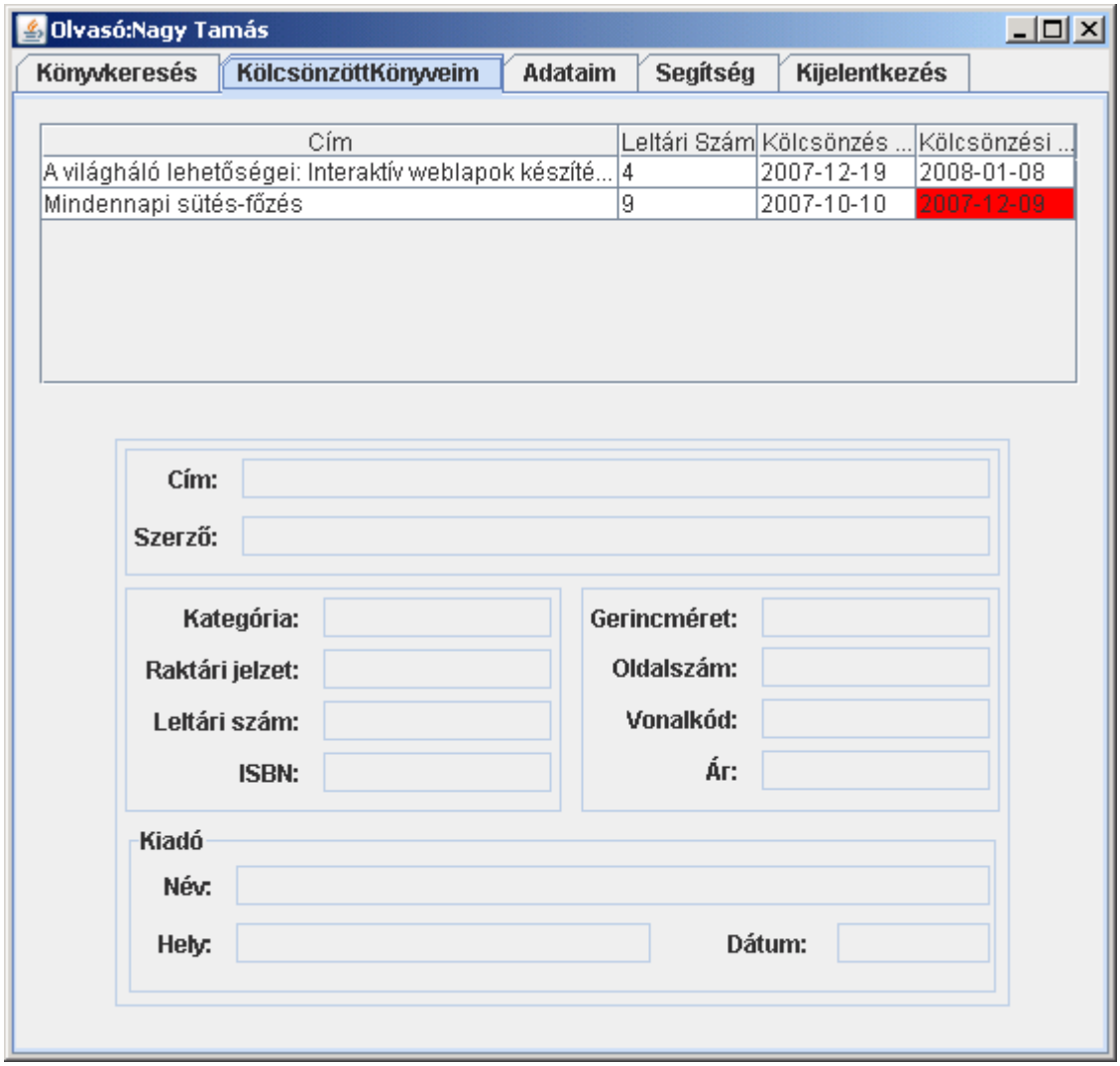
14. ábra Hibaüzenet hibás bejelentkezés esetén

6.2. Olvasók számára

6.2.1. Könyv keresése

Olvasó bejelentkezése után is szintén a keresési felület köszönt, ami megegyezik a program elindítása után elénk táruló keresési felülettel.

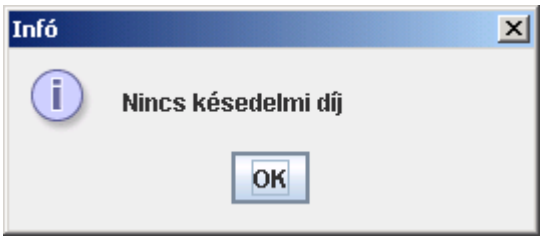
6.2.2. Kölcsönzött könyvek megtekintése



15. ábra Olvasó saját könyveinek megtekintése

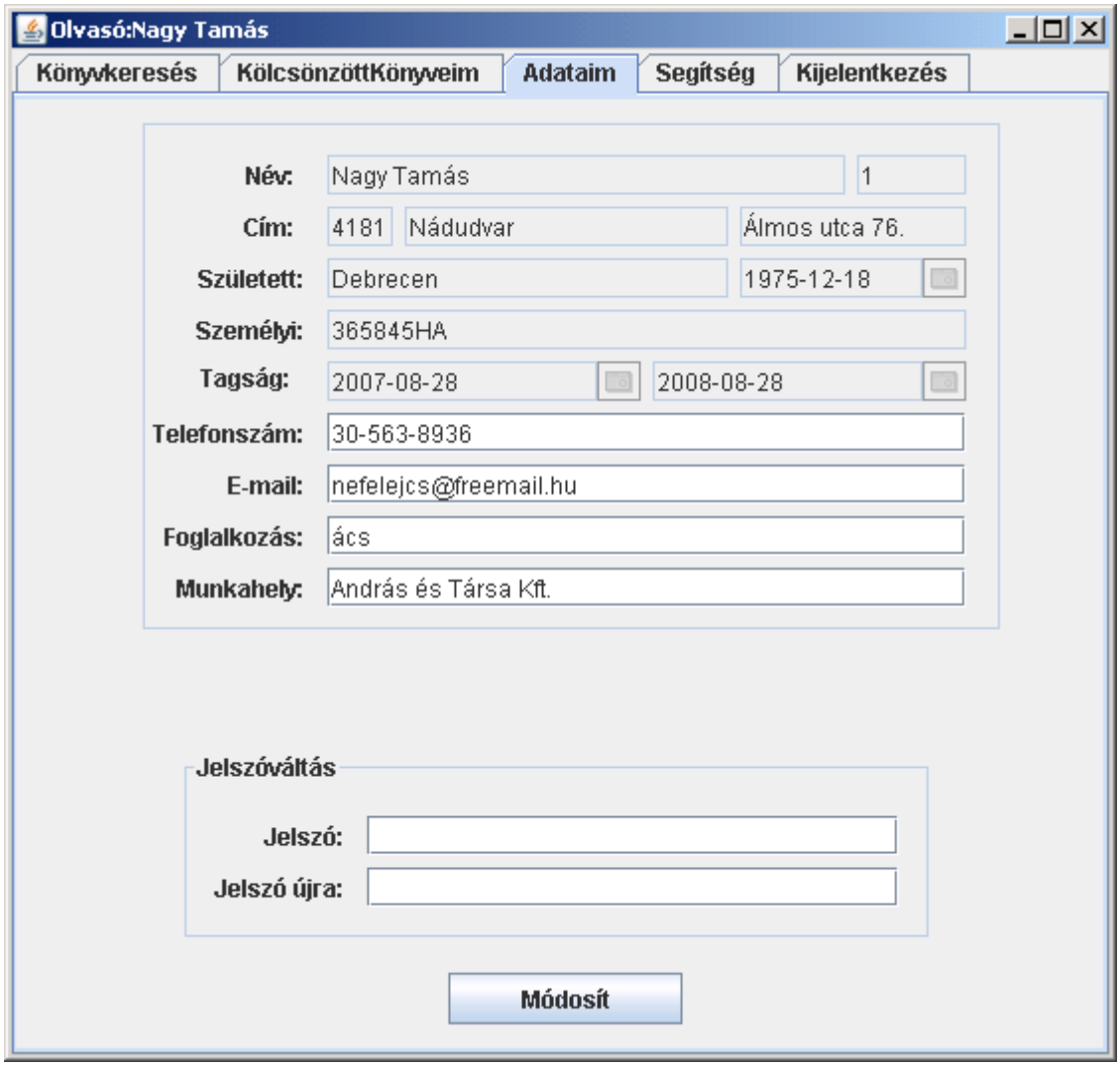
Ha kettőt kattintunk valamelyik kölcsönzött könyvre, akkor a könyv részletes adatai jelennek meg. Az utolsó oszlopban a könyv lejárati dátuma szerepel. Ha piros a háttér, akkor lejárt a kölcsönzési határidő és késedelmi díjat kell fizetnie az olvasónak. A késedelmi díj nagyságát úgy határozzuk meg, hogy jobb egérgombbal rákattintunk az adott kölcsönzött könyvre. Ekkor megjelenik a menü, melynek szövege ”Késedelmi díj”, amire ha ismét rákattintunk, megtudjuk a késedelmi díj nagyságát.

16. ábra Késedelmi díj nagysága



17. ábra Az adott könyvön nincs késedelmi díj

6.2.3. Adataim



18. ábra Olvasó személyes adatai

Az olvasó megnézheti saját adatait. A telefonszám, e-mail, foglalkozás, munkahely és jelszó adatait módosíthatja is. Más adatok módosítását csak a könyvtáros végezheti.

6.2.4. Kijelentkezés

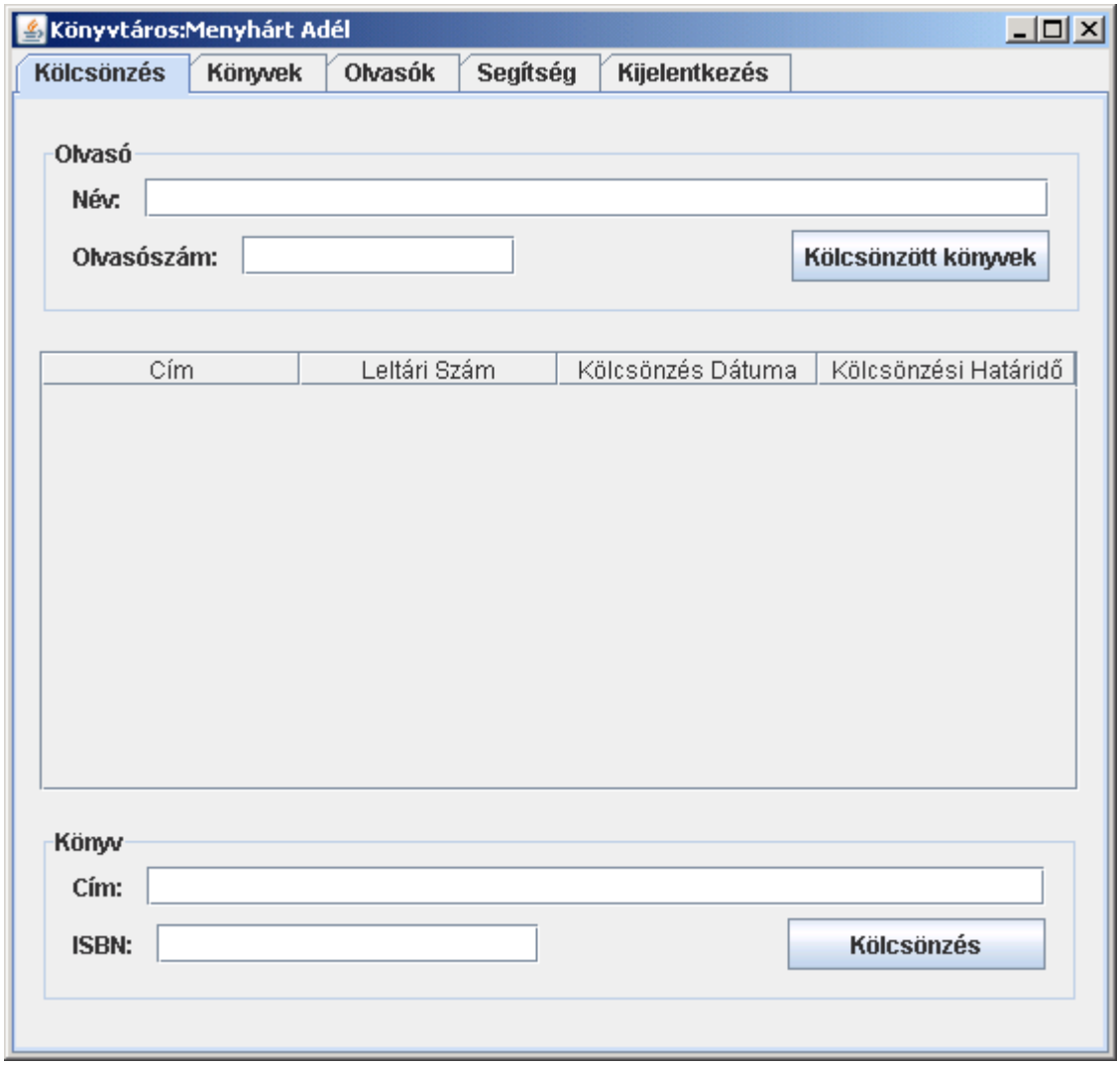
A kijelentkezés fülre kattintva megjelenik egy dialógus ablak, ami megkérdezi az olvasót, hogy valóban ki szeretne-e lépni. Az Igen gomb lenyomásával kijelentkezik és a bejelentkezési felületre érkezik.

19. ábra Kijelentkezés előtt megjelenő ablak

6.3. Könyvtáros számára

6.3.1. Kölcsönzés

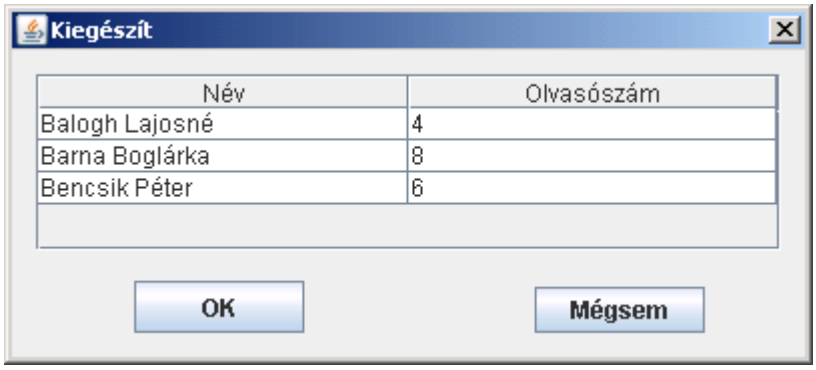
Könyvtárosként bejelentkezve az alábbi kép fogad minket.



20. ábra Könyvtáros könyv kölcsönzése és visszavitele

Itt a könyvtáros adott olvasónak tud kölcsönözni könyvet. Olvasószám begépelése után, ha entert ütünk, és létezik azzal az olvasószámmal olvasó, akkor kiírja az olvasó nevét. Ha a nevét kezdjük gépelni, és entert ütünk, a következő 3 lehetőség valamelyike történik:

- Kiíródik a teljes neve és az olvasószáma a megfelelő helyre, ha a begépelt karakterek megegyeznek egy olvasó nevének elejével.
- Begépelt karakterek utolsó karaktere eltűnik, ha nincs olyan olvasó, melynek neve a begépelt karakterrel kezdődik.
- Megjelenik egy dialógusablak és felkínálja, hogy válasszunk a lehetőségek közül, ha több olyan olvasó is van, akik neve a begépelt karakterrel kezdődik.



21. ábra Olvasó nevének kiegészítése

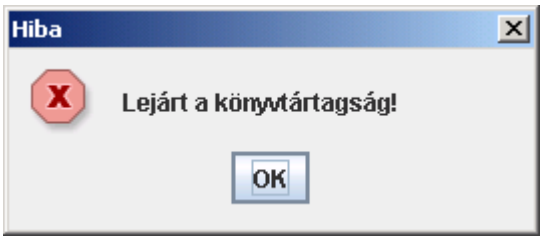
A kölcsönözni kívánt könyvnél is hasonló könnyítések vannak. Nem csak a könyv nevét nem kell teljesen beírni, hanem az ISBN számot sem muszáj teljesen leírni, csak entert kell ütni, és a program segít nekünk.

22. ábra Olvasó kölcsönözni kívánt könyvének kiválasztása

A kölcsönzött könyvek gombra kattintva informálódhat a könyvtáros az olvasó kölcsönzött könyveiről. Piros háttér jelzi, ha lejárt a kölcsönzési határidő, ekkor késedelmi díjat kell fizetnie az olvasónak. Ha jobb egérgombbal kattintunk a kikölcsönzött könyvek valamelyikére, akkor nem csak a késedelmi díjat tudjuk megnézni, hanem ha az olvasó

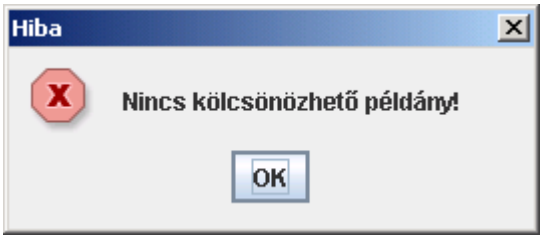
visszahozta a könyvet, akkor azt is elvégezhetjük vele. Ha nem járt le vagy 2 napja járt le a könyv, akkor meg is tudjuk hosszabbítani. Alapesetben kétszer tudunk hosszabbítani és egy hosszabbítás 20 napra szól.

A kölcsönzés gombra kattintva a könyvtáros kölcsönözhet könyvet az olvasónak az olvasószámot és az ISBN számot használva. Ha az olvasónak lejárt a tagsága, akkor megjelenik egy hibaüzenet, és nem kölcsönözhet könyvet, ameddig a tagságot meg nem hosszabbíttatja



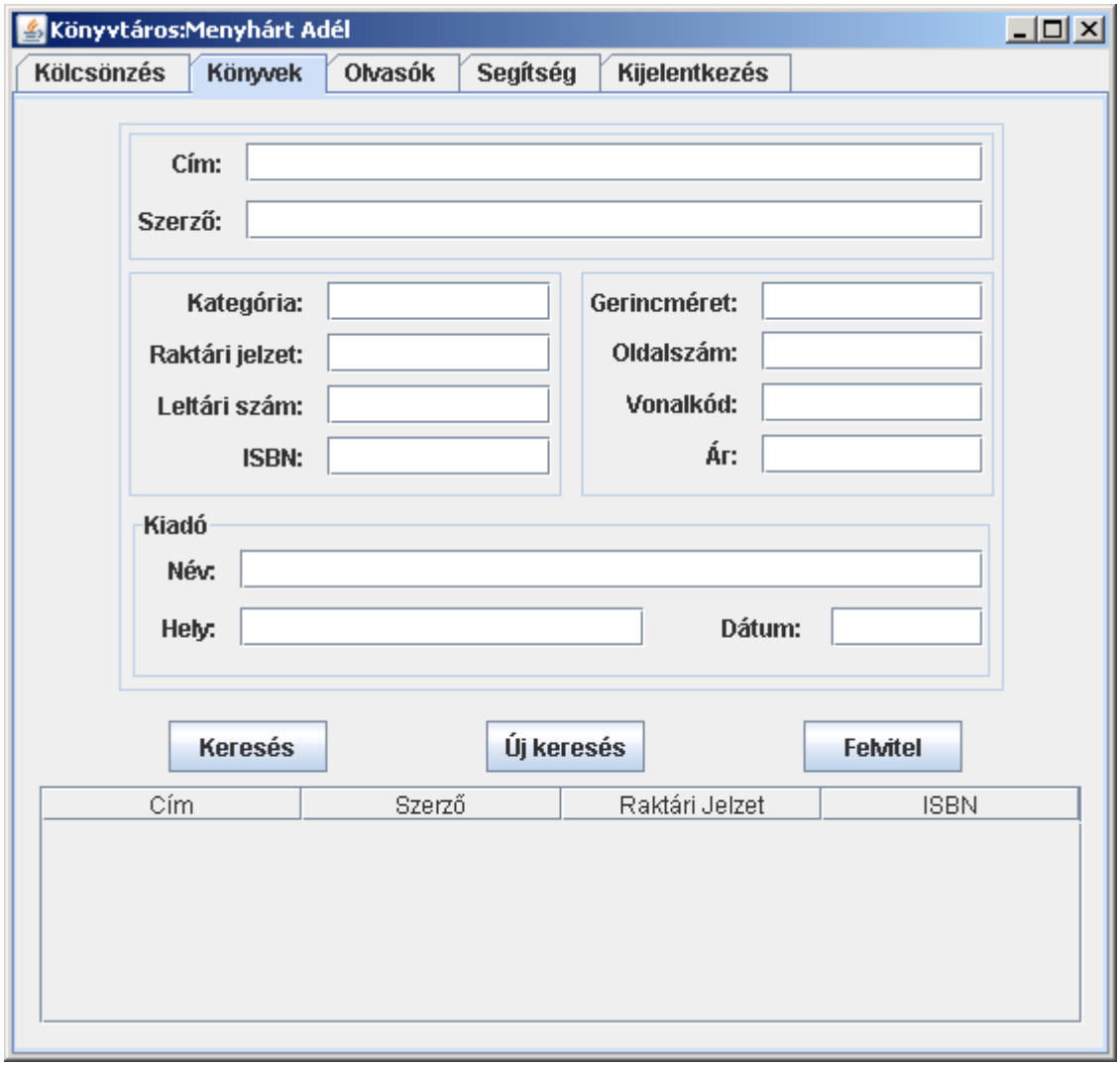
23. ábra Olvasónak lejárt a könyvtártagsága

Ha nincs olyan könyv a könyvtárban, vagy van, de ki van kölcsönözve, akkor az alábbi hibaüzenetet kapjuk:



24. ábra Nincs kölcsönözhető példány a könyvtárban

6.3.2. Könyv keresése, felvitele és módosítása

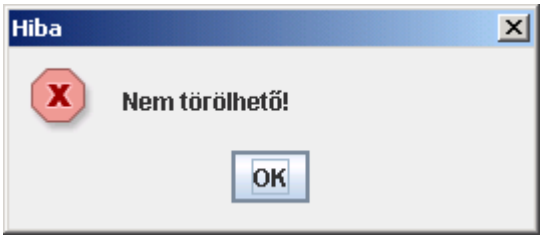


25. ábra Könyv keresése, felvitele és módosítása

Hasonlít a program elindítása utáni keresési felületre, de itt annál több funkció áll rendelkezésre. Amit ott is lehet használni, azt itt nem írom le, csak az azon felüli funkciókat. Kategóriánál nem csak kiválasztani lehet egy kategóriát, hanem újat is hozzá tudunk adni, illetve törölni. Jobb egérgombbal kattintva egy kategóriára a felugró menüből kiválaszthatjuk, hogy törölni szeretnénk-e azt, vagy a kiválasztott kategórián belül szeretnénk létrehozni egy új alkategóriát.

26. ábra Kategória felvitele

Ha olyan kategóriát próbálunk törölni, amely tulajdonsága egy könyvnek, hibaüzenetet kapunk.



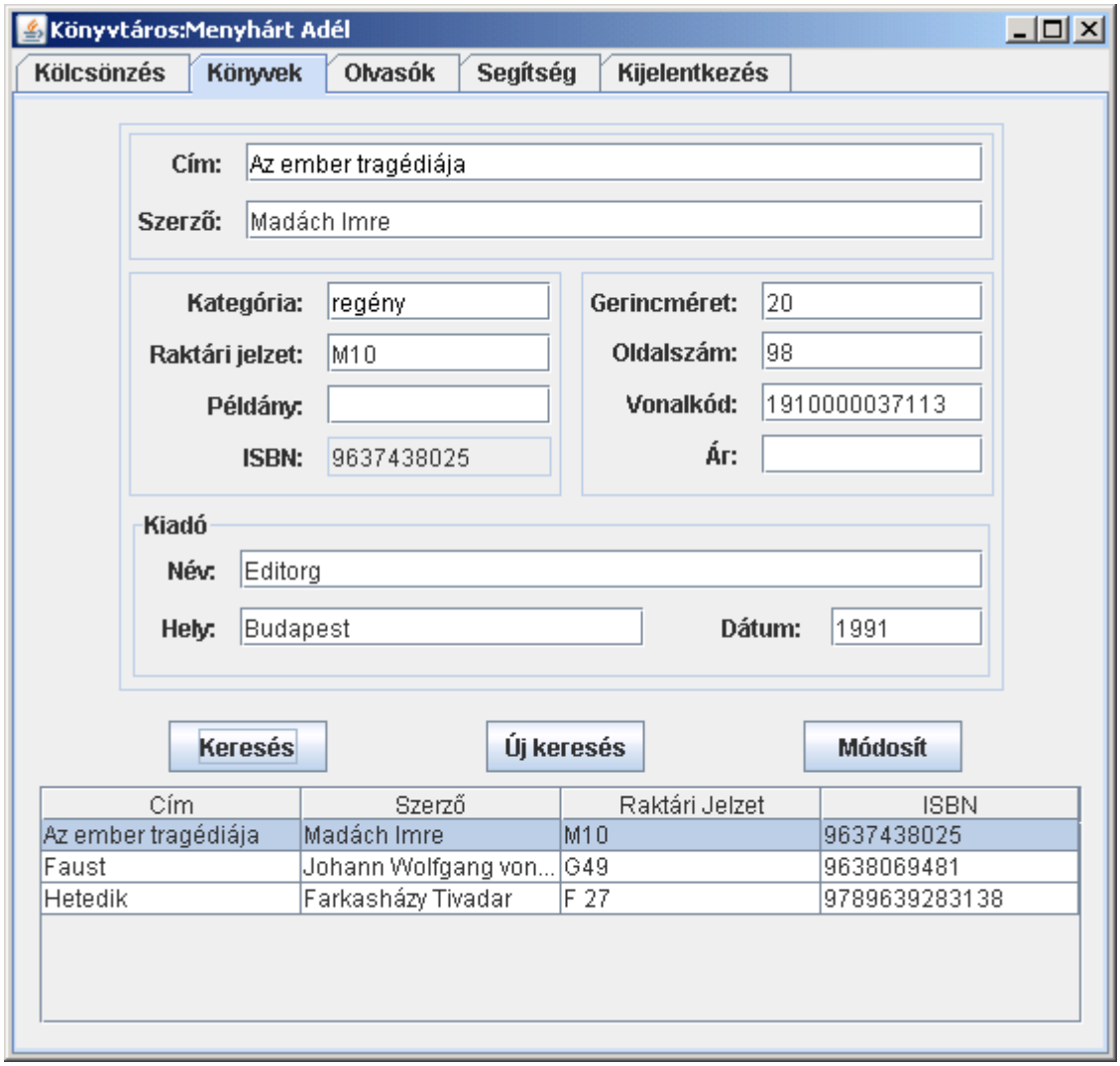
27. ábra Hiba kategória törlése közben

Keresés után, az eredményre elég csak kétszer rákattintani és már meg is jelennek a könyv részletei.

Jobb egérgombbal rákattintva négy lehetőség közül választhatunk:

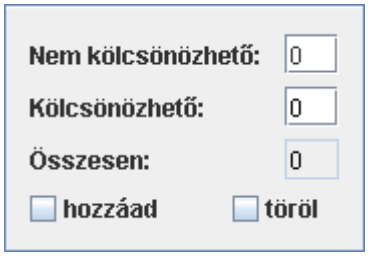
- Módosít
- Példány
- Kölcsönözték
- Kölcsönöz

Módosításra kattintva a könyv tulajdonságait tudjuk módosítani az ISBN számon kívül.



28. ábra Könyv attribútumainak módosítása

Könyv példányait a példány mezőben, valamely leütött billentyű hatásával módosíthatjuk. Ekkor felugrik egy ablak.



29. ábra Könyv példányainak módosítása

A nem kölcsönözhető mezőbe, természetesen a nem kölcsönözhető könyvek számát írjuk be, például a csak a könyvár olvasótermében használható példányok számát. Ha a hozzáad checkbox van bepipálva, akkor hozzáadja az adatbázishoz, ha a töröl, akkor törli az

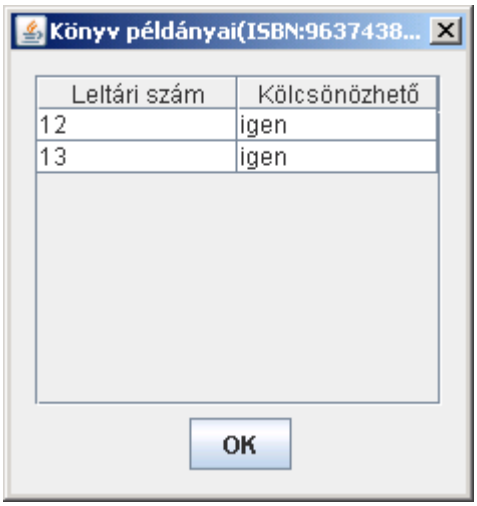
adatbázisból a megadott számú könyvet. Ha egyik sincs bepipálva, akkor a beírt értékekre állítja a könyv példányait.

Ha a kívánt értékeket megváltoztattuk, akkor azt a módosít gombra kattintva véglegesíthetjük. Ha nem történt hiba, akkor az alábbi üzenetet kapjuk:

30. ábra Könyv módosítása sikerült

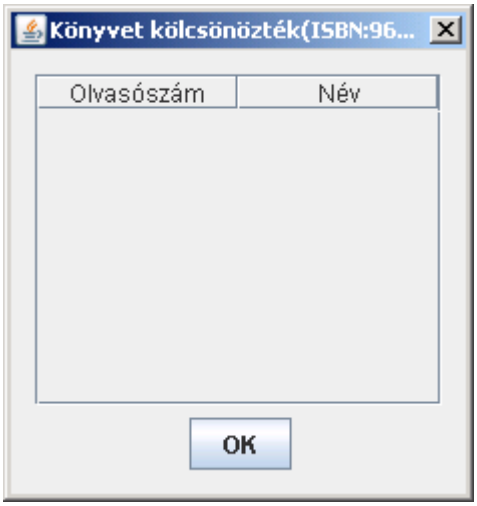
Ha nem ez jelenik meg, akkor a hibával kapcsolatos hibaüzenetet kapunk.

Példányra kattintva a könyvek példányairól informálódhatunk, hogy melyik kölcsönözhető és melyik nem.



31. ábra Könyv példányai kölcsönözhetőek-e?

A ”Kölcsönözték”-re kattintva megtudhatjuk, hogy kik kölcsönözték.



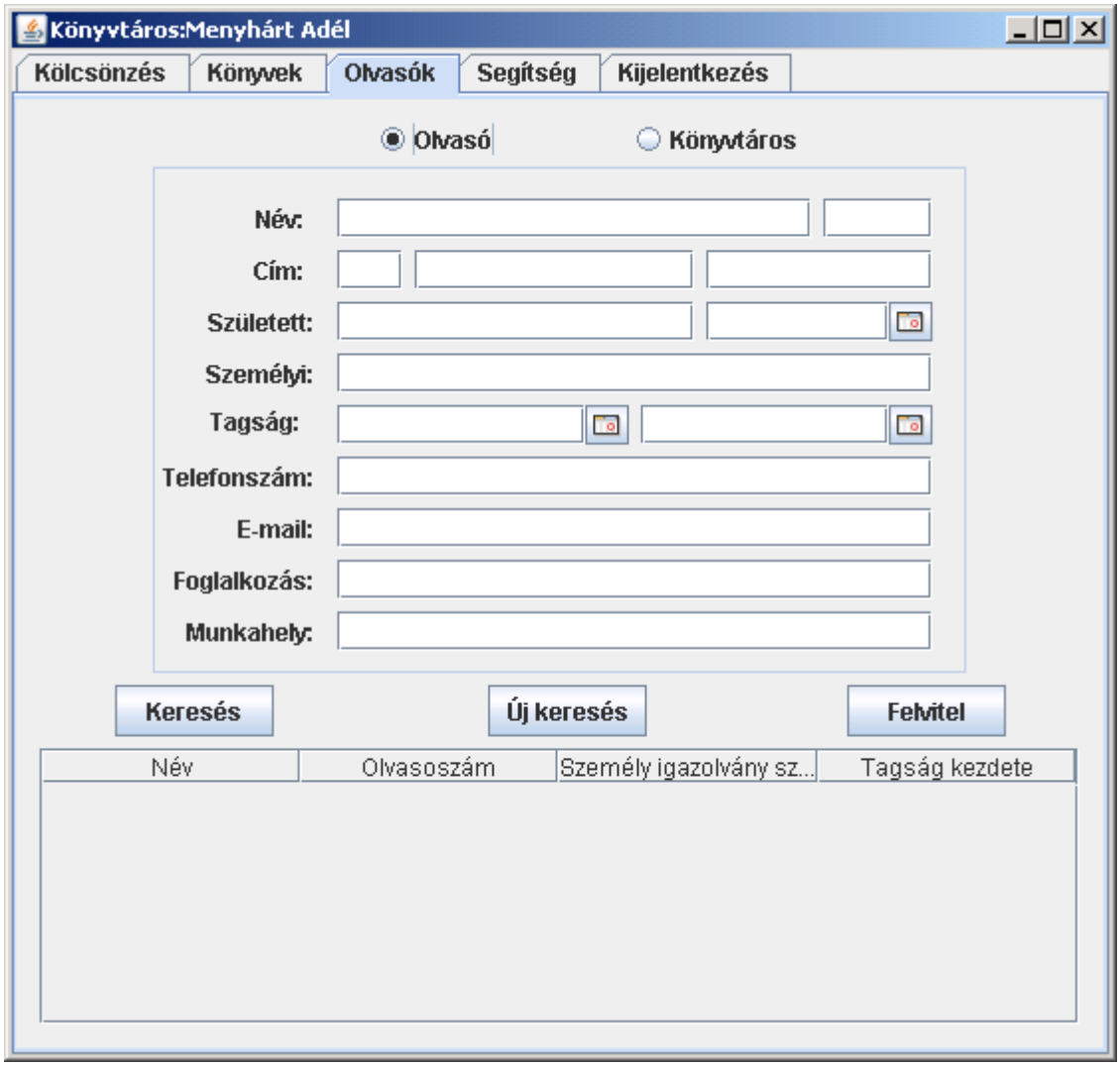
32. ábra Kik kölcsönözték az adott könyvet?

Az utolsó menüpont a ”Kölcsönöz” menüpont. Erre kattintva a könyvet beteszi a program a kölcsönzési panelbe.

”Új keresés” gombra kattintva törlődnek a mezők és ha módosítás módban voltunk, akkor visszaáll felviteli módba és a harmadik gomb felirata ”Felvitel”-re változik.

A beviteli mezők kitöltése után a ”Felvitel” gombra kattintva felvihetünk egy könyvet. Ha kitöltjük a leltári szám mezőt, akkor ez itt nem a tényleges leltári számot jelenti, hanem a könyv felvitele mellett az adatbázisba bevitt kölcsönözhető példányok számát. Ha nem töltjük ki, akkor csak a könyvet adja hozzá az adatbázishoz. Példányai ekkor nem lesznek. Ha példányokat is akarunk, akkor módosítanunk kell a fent említett módon.

6.3.3. Könyvtártag keresése, felvitele és módosítása



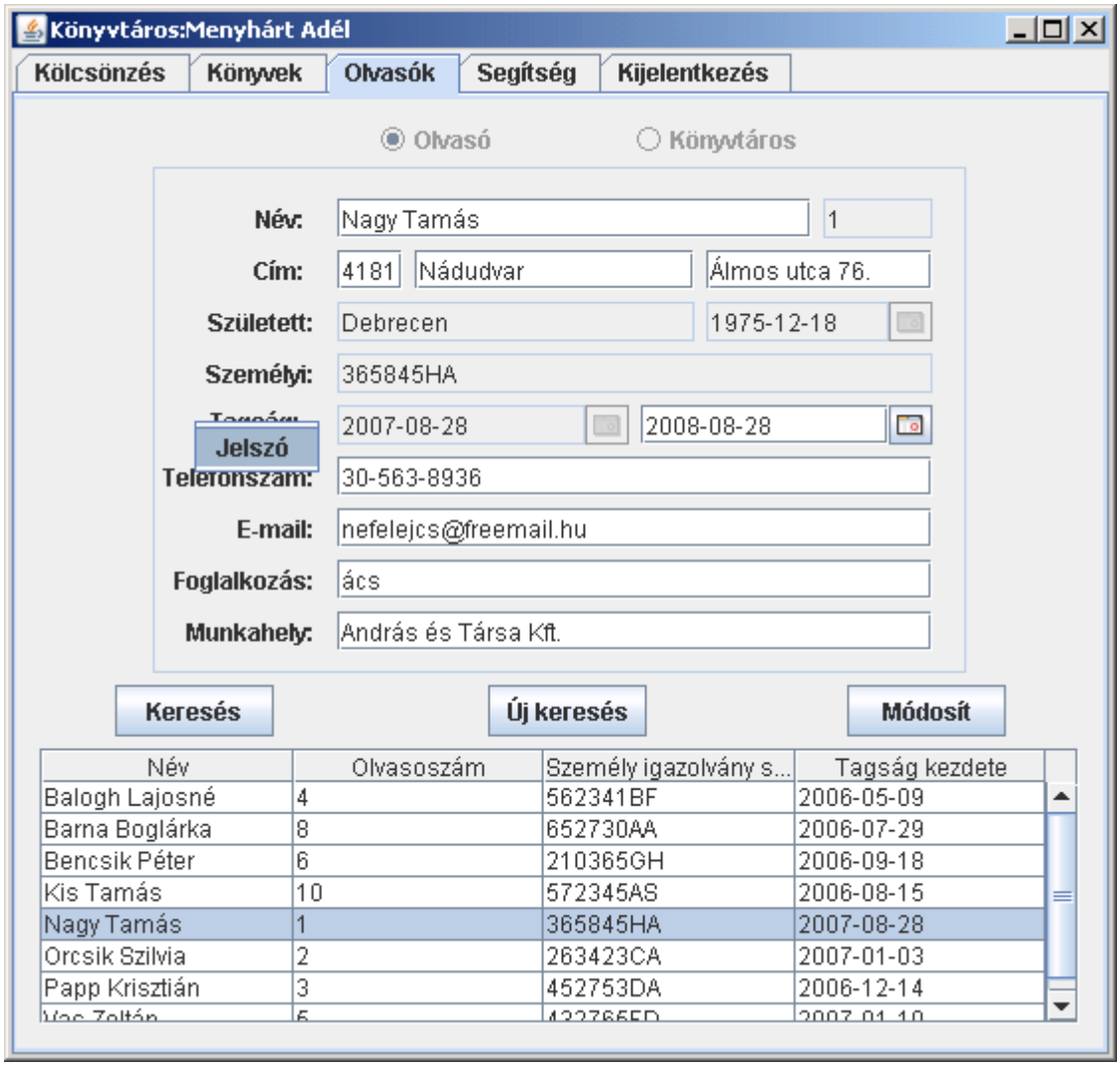
33. ábra Könyvtártag keresése, felvitele és módosítása

Felül kiválasztjuk, hogy olvasót vagy könyvtárost szeretnénk keresni, felvinni vagy módosítani. Ha keresni szeretnénk, itt is használhatjuk a könyvek keresésénél használt karaktereket. A dátum formátuma a következő ”yyyy-MM-dd”, ahol yyyy az évszámot, MM a hónapot és dd a napot jelöli. A dátum mező végén található ikonra kattintva egérrel is kiválaszthatjuk az adott dátumot.

34. ábra Dátum beállítását segítő dialógusablak

A talált olvasókra három művelet alkalmazható. Dupla kattintás hatására az olvasó részletes adatait tekinthetjük meg. Jobb egérgomb hatására a felugró menüből lehet kiválasztani, ha módosítani szeretnénk az adott olvasó adatait, ekkor a módosít fülre kattintunk. A másik lehetőség, amikor az olvasó könyvet szeretne kölcsönözni, ekkor a kölcsönöz szövegre kell kattintani. Hatására megjelenik a kölcsönöz panel, és ki lesz töltve az olvasó neve és olvasószáma a megfelelő helyen.

A módosításnál, ha jelszót szeretnénk módosítani, akkor a beviteli mezőknél kell egy jobb kattintást végezni, majd a felbukkanó menünél a jelszó feliratra kattintani.



35. ábra Jelszó megváltoztatása

36. ábra Jelszó megadása

6.3.4. Kijelentkezés

Működése megegyezik az olvasóknál leírt folyamattal.

6.4. Üzembe helyezés

A mellékleten található könyvtárszerkezet:

- conf könyvtár: Beállításokat tartalmazó fájlok.
- database könyvtár: Adatbázisokkal kapcsolatos fájlok, sql scriptek.
- lib könyvtár: A futáshoz szükséges osztályok gyűjteménye.
- log könyvtár: Naplózáshoz használt logfájlt tartalmazza.
- KonyvtariNyilvantarto.jar file: Maga az alkalmazás.

Üzembe helyezés lépései:

1. Melléklet tartalmát másoljuk egy általunk kiválasztott mappába.
2. Adatbázis-kezelő rendszer választása. A MySQL az alapértelmezett adatbázis-kezelő, de ha másikat szeretnénk, szerezzük be a JDBC meghajtóprogramját, másoljuk be a lib könyvtárba és nevezzük át JDBCDriver.jar-ra.
3. Hozzuk létre a KonyvtariNyilvantarto adatbázist. Más nevet is adhatunk az adatbázisunknak, de ekkor a conf/ beallitasok.properties fájlt a megfelelően módosítanunk kell. A hozzáférési név és jelszó megadásához is szintén ezt a fájlt kell módosítanunk.
4. Táblák létrehozása. A DB2, Derby, HSQL, MSSQL, MySQL, Oracle és a PostgreSQL adatbázisokhoz létre van hozva a táblákat létrehozó szkript, mely a database/<adatbázis> könyvtárakban található meg. A database könyvtárban található egy segédprogram, mely szkriptek lefuttatására használható. Használata: java –jar Database.jar –file <szkript fájl elérési útvonala>
5. Az alkalmazás használható. Futtatása a ”java –jar KonyvtariNyilvantarto.jar” paranccsal lehetséges. Belépéshez a fent említett beallitasok.properties fájlt kell módosítanunk. Belépés után, könyvtárosként lehet olvasókat, könyvtárosokat és könyveket felvinni és módosítani. Lehetőség van egy példa szkript lefuttatására is, hogy megnézzük az alkalmazás működését. Ennek módja:java –jar Database konyvtarinyilvantarto_example.sql

A beallitasok.properties fájlban szereplő értékek:

- DEBUG – Értéke true vagy false. Segítségével a program futása közben keletkezett hibát írattathatjuk ki.
- JDBC.DRIVER – JDBC driver osztály neve.

- JDBC.URL, USER, PASS – Segítségükkel adjuk meg, hogy melyik adatbázishoz kívánunk kapcsolódni, illetve hogy milyen felhasználónévvel és jelszóval kívánjuk ezt megtenni.
- KOLCSONOZHETO.SZER – Ennyiszer lesz meghosszabbítható a könyv.
- KOLCSONOZHETO.NAPOK – Azon napok számát jelöli, amely a kölcsönzés dátumától a visszavitel dátumáig terjed.
- NAPIDIJ – Késedelmi napidíj.
- ROOTUSER, ROOTPASS – Alkalmazás telepítése után az itt megadott felhasználónévvel, illetve jelszóval léphetünk be a programba, hogy feltöltsük és módosítsuk az adatbázist.

7. Összegzés

A fejlesztés folyamán számos új tapasztalattal lettem gazdagabb, melyeket reményeim szerint a későbbiekben megfelelően fogok tudni hasznosítani. Mélyebben megismerkedtem a java nyelvvel és nem utolsó sorban a JDBC nyújtotta lehetőségekkel is.

A konkrét alkalmazás fejlesztése során ismerkedtem meg, olyan a napjainkban használt eszközökkel, technológiákkal, melyeknek ismerete manapság már elengedhetetlen az informatikai munkaerőpiacon.

A diplomamunkám írása során jöttem rá arra, hogy a java alkalmazásfejlesztés legmegfelelőbb bemutatása az lett volna, hogy ha egy valós problémát és annak egy lehetséges megoldását mutattam volna be. Úgy olyan kérdésekre is összetettebb válaszokat kaptam volna, hogy milyen kérdések, problémák merülhetnek fel a fejlesztés folyamán, miközben a megrendelővel konzultálok.

Itt szeretnék köszönetet mondani témavezetőmnek, Iszály György Barnának. A dolgozat megírása közben ötletekkel, észrevételekkel segített abban, hogy a dolgozatom kerek egésszé váljon. Segített abban, hogy az eddig megszerzett ismereteimet rendszerezzem, s ennek megfelelően alakítsam dolgozatom.

8. Ábrajegyzék

9. Irodalomjegyzék

1. Angster Erzsébet: Objektumorientált Tervezés és programozás I-II, 4KÖR Bt., Budapest, 2003

2. Nyékyné Gaizler Judit: Java 2, Útikalauz programozóknak 1.3 I-III, ELTE, Budapest, 2001

3. Stolnicki Gyula: SQL kézikönyv : SQL92 és IBM DB2, DB2/2, SQL/DS, Informix, Ingres, MS SQL server, Novell XQL, Oracle, Sybase, ComputerBooks, Budapest, 1995

4. Vég Csaba, dr. Juhász István: Java - start!, Logos 2000, Debrecen, 1999

Internetes források

1. GUI Building in NetBeans IDE - http://www.netbeans.org/kb/55/quickstart-gui.html
2. Java documentation - http://java.sun.com/j2se/1.5.0/docs/api/
3. MySQL Documentation - http://dev.mysql.com/doc/
4. The Java Tutorial - http://java.sun.com/docs/books/tutorial/