Eötvös Loránd Tudományegyetem
Informatikai Kar
Információs Rendszerek Tanszék

Többasztalos és -felhasználós póker játék adatbázis modellezése

Ács Zoltán
tanársegéd

Fehér Valentin
Programtervező Informatikus BSc

Budapest, 2016

Tartalomjegyzék

I. rész

Bevezetés

1. A témaválasztás indoklása

Mindenképpen egy online, többfelhasználós játékot szerettem volna megvalósítani. Később leszűkítettem a kört kártyajátékokra, és végül az ulti és a póker között vaciláltam. A döntésem a pókerre esett, ugyanis az ulti viszonylag bonyolultabb, mint a póker, több szabály, több megszorítás a partykra vonatkozólag, ráadásul ahány ház annyi szokás alapon könnyen nézet eltérések szoktak keletkezni az ultizás során. Pókerezni egyszerűbb - bár ezt sokan vitatják - és jó formán mindenki könnyen megérti a játék lényegét.

2. Feladat leírás

II. rész

Felhasználói dokumentáció

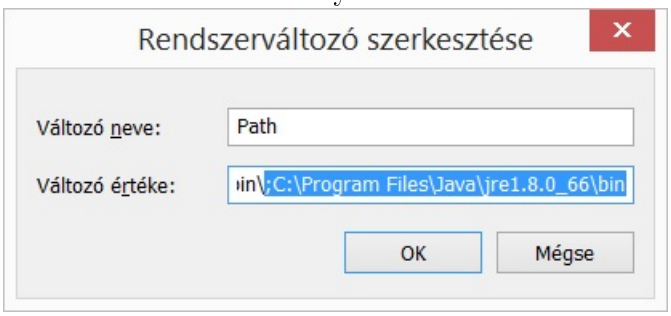
A programcsomagot JAVA programozási nyelven írtam meg, így a program számára biztosítani kell JAVA futtatókörnyezetet, továbbá a program működéséhez adatbázis szerverre is szükség lesz. A telepítés magába foglalja az adatbázis séma elkészítését és az adatbázis demó adatokkal való feltöltését is, így a telepítést követően extra dolgokra nem lesz szükség a szoftver használatához. A telepítés közben a számítógépet újraindítani nem szükséges.

3. Telepítés

3.1. Java SE Runtime Environment 8

A programcsomag futtatásához legalább Windows XP operációs rendszer szükséges, amelyen Java SE Runtime Environment 8 futtató környezet [?] (a továbbiakban: JRE) fut. A JRE feltelepítését követően manuálisan ellenőrizzük, hogy a rendszer felvette-e környezeti változóként az installációs könyvtárat. Navigáljunk az operációs rendszerben a környezeti változók módosítása panelhez, majd ellenőrizzük le, hogy a PATH nevű környezeti változóhoz hozzá lett-e adva az installációs könyvtár, mint például: `C:\ProgramFiles\Java\jre1.8.0_66\bin`. A telepítési könyvtár operációs rendszerenként eltérhet.

1. ábra. PATH környezeti változó értéke



Ha a JRE installációs könyvtár gyökerében található bin könyvtár nincs behivatkozva a PATH nevű környezeti változó értékéhez, akkor az 1. ábrán látható módon egészítsük ki az értéket, majd mentsük el a változtatásokat és indítsuk újra a promptot.
Ha sikeresen jártunk el, akkor a

```
java -version
```

2. ábra. JRE verzió

```
D:\projects\szakdolgozat\poker>java -version
java version "1.8.0_66"
Java(TM) SE Runtime Environment (build 1.8.0_66-b18)
Java HotSpot(TM) 64-Bit Server VM (build 25.66-b18, mixed mode)
```

utasítás hatására a 2. ábrán látható szöveg jelenik meg a konzolon, akkor sikeres volt a JRE telepítése és beállítása.

3.2. MySQL Community Server 5.6

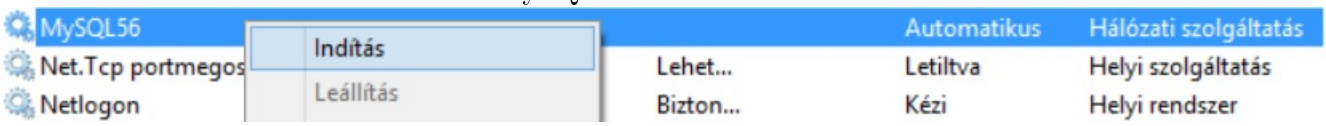
A programcsomag megköveteli a MySQL Community Server 5.6 adatbázis-kezelő rendszer [**?**] (a továbbiakban: MySQL Server) használatát is. Letöltés után csomagoljuk ki a zip állományt egy tetszőleges könyvtárba, majd a fentiekkel megegyező módon adjuk hozzá a PATH nevű környezeti változó értékéhez a MySQL Server bin könyvtár elérési útvonalát. Ha ezzel végeztünk, akkor nyissük meg a promptot (ha nyitva van, akkor indítsuk újra), majd navigáljunk a MySQL Server bin könyvtárába, ott pedig adjuk ki a

```
mysqld --install
```

parancsot. A parancs végrehajtása után navigáljunk a szolgáltatások panelhez, amelyet a legkönnyebben a promptban a

```
services.msc
```

3. ábra. MySQL Service



kiadott utasítással lehet elérni. Majd járjunk el a 3. ábrának megfelelően. Térjünk vissza a konzolra, ahol adjuk ki a

```
mysql -u root -p
```

parancsot, amely jelszót fog kérni. A beviteli sort hagyjuk üresen, nyomjunk entert. Ha sikeresen beléptünk az adatbázis-kezelő rendszerbe, akkor adjuk ki a

```
SELECT VERSION();
```

utasítást, és ha a 3. ábrának megfelelő képernyőképet kapunk, akkor sikeresen feltelepítettük az adatbázis-kezelő rendszert.

3.3. Az adatbázis használatba vétele

Ha sikeresen elindítottuk a MySQL Servert, akkor szükségünk lesz egy új adatbázis sémára és demó adatokra, amelyet a `X:\poker\release\poker-db.sql` állományban találunk. Ezt a filet kell lefuttatni az adatbázison, a hatása idempotens. A promptban adjuk ki a

```
mysql -u root -p < X:\poker\release\poker-db.sql
```

utasítást, amely jelszót fog kérni. A beviteli sort ugyancsak hagyjuk üresen. Ha sikeresen lefutott a parancs, akkor az adatbázis séma elkészült és a program demó adatokkal is feltöltötte.

4. Fogalomtár

Az alábbi parancsokat a felhasználók tudják kiadni. A szerver biztosítja, hogy csak a megfelelő parancsok legyenek hívhatóak a póker partik során.

4.1. Parancsok

CALL
A felhasználó ezt a parancsot adja ki, ha a tartozását ki szeretné egyenlíteni az asztal felé.

CHECK
Játékos által kiadható parancs, mely akkor adható ki, amikor a felhasználó nem tartozik az asztal felé, és emelni sem szeretne az adott játékos.

BLIND
Olyan parancs, amelyet a kliens oldali program automatikusan hív meg. Ha a szerver BLIND típusú utasítást küld, akkor a kliens automatikusan beteszi a kis vagy nagy vakot az asztalra attól függően, hogy a szerver mire kötelezte az adott felhasználót. A kis vak, vagy angol nevén small blind az asztalnál érvényben lévő alaptét felét, míg a nagy vak, vagy angolul big blind az alaptét egészét jelenti.

RAISE
Olyan speciális parancs, amely hatására a felhasználó tartozása kiegyenlítésre kerül az asztal felé és még emel a téten az asztal alaptétjének megfelelő értékkel.

QUIT
Asztal elhagyására szolgáló parancs. A kilépését követően a program visszairányítja a felhasználót a táblalistázó oldalra.

CHANGE
A parancs csak a classic játékstílusban érhető el. A parancs meghívásával a felhasználó utasítja a programot, hogy a cserére megjelölt kártyalapokat a szerver új kártyalapokkal helyettesítse.

LOG
A felhasználó játék közben vissza tudja nézni a már megtörtént főbb eseményeket. Például az előző játékos milyen parancsot hajtott végre.

4.2. Egyéb...?

Dealer
Magyarul osztó. A játék során az osztó gomb az óramutató járásával megegyező irányban halad körbe az asztalon, így minden parti esetén más felhasználó tölti be az a osztó szerepét.

5. A póker játékról

A póker, mint kártyajáték igen népszerű szerencsejáték. Akár élő tv adásokat is végig lehet követni, ahol hatalmas főnyereményeket osztanak ki a dobogós helyezetteknek. Viszonylag sok fajtája terjedt el szerte a világon, kezdve a klasszikus 5 lapos leosztásokkal egészen az OMAHA-án át a jól ismert Texas Hold'Em játékstílusig. A játékot 52 lapos francia kártyapaklival játszák, amelyben 4 szín és 13 különböző értékű kártyalap található. A játékstílusok igen különbözőek tudnak lenni. Megkölönböztetünk ante, illetve blind alaptétet. Ante alaptét esetén az asztalnál ülő összes játékos előre meghatározott tétet rak be. Blind alaptét esetén megkülönböztetünk kis és nagy vakot, amelyet körönként más és más játékosok raknak be. Minden esetben az osztótól közvetlenül balra ülő játékosnak kell beraknia a kis vakot. Az osztótól kettővel balra ülő játékosnak pedig a nagy vakot. Amíg az alaptétek nem kerültek be az asztalra, addig a ház nem kezdi meg a lapok kiosztását.
A 4. ábrán látható az OMAHA játékstílus. A Texas Hold'Em játékstílushoz igen hasonló póker játék változat. A játékosok a kezükbe négy darab kártyalapot kapnak a vakok betétele után, majd megkezdődik az első licitkör, amelynek a végén három közös lap kerül ki az asztal közepére. Újabb licitkör kezdődik, amelynek a végén egy, majd az újabb licitkör végén, még egy újabb kártyalap kerül az asztal közepére, mint közöslap. A felhasználóknak maximum öt lapot lehet felhasználiuk a legjobb kéz előállítására. Az öt lapból kötelezően kettő a saját kézből történik a maradék három pedig a közös lapokból. Értelemszerűen a legerősebb kéz nyer [**?**]. A játék célja, hogy minél több zsetont gyűjtsünk össze a partik során.

4. ábra. OMAHA póker játék

5.1. Játékmenet

Minden játékszerver úgy lett konfigurálva, hogy két játékos esetén a parti elkezdődjön. Ha valaki később csatlakozott az asztalhoz, az a megkezdett partiból semmit nem érzékel, mintha üres asztalnál ülne. Csak a következő partiba tud beszállni. Mindkét játékstílus esetén van egy BLIND kör, amikor a szerver bekéri a vakokat a játékosoktól. Az osztótól eggyel balra ülő játékos köteles betenni a kis vakot, a kis vaktól eggyel balra ülő játékos pedig köteles betenni a nagy vakot. Ebből a felhasználó nem lát semmit, köteles beadni a kis- vagy nagyvakot, amelyet egy automatizált eljárás hajt végre. Az osztó gomb a legelső körben kerül kiosztásra a legelsőként csatlakozott játékoshoz. Az osztó gomb az óramutató járásával megegyező irányban halad. Minden új megkezdett parti esetén az osztó gomb a következő játékoshoz kerül. Minden parti legelső körében a kezdő játékos az osztótól balra ülő harmadik játékos, minden további kört a kis vakra kötelezett játékos kezd meg.
A játékszerverek nem képesek kezelni, ha egy játékosnak elfogyott, illetve nincs elegendő zsetonja. Ebből kifolyólag esetenként előfordulhat negatív egyenleg, illetve nem definiált viselkedés. A játékosok szigorúan csak egymást követve küldhetnek utasításokat a szervernek, mindig az éppen soron levő játékos. A felhasználói grafikus felületen egyértelmű jelzéssel van ellátva az éppen soron levő játékos. A játékosok megadhatják, emelhetik a tétet, illetve, ha nem szeretnének az adott körben semmit

csinálni, akkor CHECK-elhetnek. Az emelés mértéke a mindenkori játékasztal alaptét felét jelenti. Ugyanakkor lap eldobásra és a játék elhagyására is lehetőség van. A felhasználók korlátozottan visszanézhetik a korábbi leosztásokat, és a partiban történt eseményeket. A játékosok asztalonkénti maximum száma 5 fő. A kliensek a játék elhagyását követően újracsatlakozhatnak az adott játékszerverre a fentieket figyelembe véve. Lehetőség van asztalt váltani, és a játékszerverek (korlátozottan) képesek kezelni, ha a játékossal megszakad a kapcsolat. A kliens alkalmazások is korlátozott mértékben fel vannak készítve az esetleg kommunikációs hibákra.

5.2. Játékstílusok

A program kettő beépített játékstílust definiál

- Classic [?]
- Texas Hold'Em [?]

Classic
A klasszikus játékstílus legfőbb ismertető jele, hogy mindenki öt lapot kap kézbe és nincsenek közös lapok. Először is a vakokra kötelezett játékosok rakják be a vakokat (automatizáltan), azután pedig úgynevezett előkör van, amikor a szerver minden játékosnak öt-öt lapot oszt kézbe, és ezek alapján lehet licitálni. Ha vége a körnek, akkor mindenki kicserélheti a lapjait, amiket saját maga választ ki a grafikus felületen a saját kártyalapjainak rákattintásával. Ha a kártyalap feljebb csúszott a grafikus megjelenítésen, akkor a kártyalapot cserére jelölte a játékos. A CHANGE feliratú gombbal cserélhetőek a kártyák. Mindenki akkor kapja meg az új kártyalapjait, ha már mindenki nyilatkozott a cseréről. Ezek után új kör indul, amikor is az új lapok birtokában tehetik meg a tétjeiket a játékosok. A kör után a szerver kihirdeti a nyertest, és a nyertes lapokat az asztal közepén jeleníti meg a grafikus felület. Kivétel, ha a nyertesek az éppen adott felhasználó, akkor a nyertes kártyalapok maga előtt jelen vannak, más kártyalapok nem kerülnek felfordításra. A felhasználók a nyertes lapok megtekintését követően kötelesek rákattintani a CHECK feliratú gombra, illetőleg ha kifutnak az időből, akkor a játék asztal kilépteti őket. Ha ebben a körben is mindenki nyilatkozott, akkor a játékasztal új partit indít.

Texas Hold'Em
A klasszikus játékstílussal erősen megegyező játékmenetű stílus. A különbség csupán annyi, hogy a ház 2-2 lapot oszt minden játékosnak, amelyekkel a játékosok a parti végéig rendelkeznek. Illetőleg a nyertes lapok semmilyen esetben sem középen, hanem a játékosoknál jelennek meg.

6. Futtatás

6.1. A póker szerver elindítása

A DVD lemezen a `\poker\release\` mappában található meg a poker-server-1.0.0.jar file. Nyissunk egy terminált a kijelölt könyvtárban, és adjuk ki a

```
java -jar poker-server-1.0.0.jar
```

parancsot. Ha a 5. ábrának megfelelő konzol loglistát látunk, akkor a szervert sike-

5. ábra. Szerver

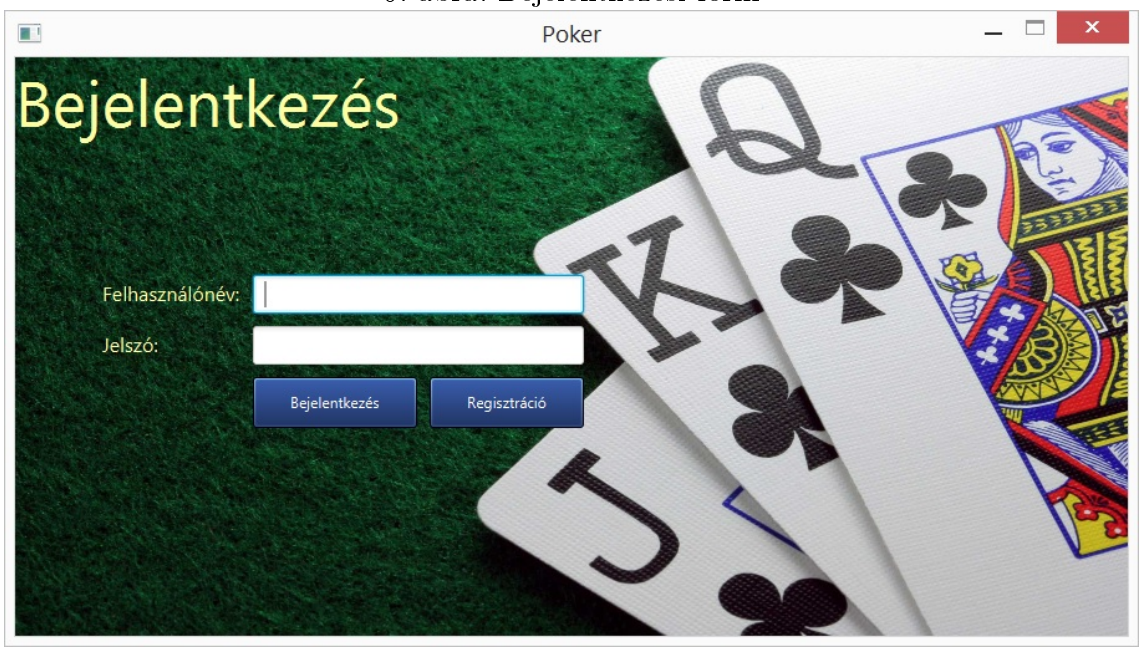
resen elindítottuk.

6.2. A póker kliens elindítása

A kliens futtatása hasonló módon történik, mint a szerveré. Navigáljunk a `\release\poker\kliens` mappába, és a konzolon adjuk ki a megfeleő parancsot. Jöhet az ábra, meg a kódot kicsit átírni, hogy logoljon konzolra, mint a szerver...

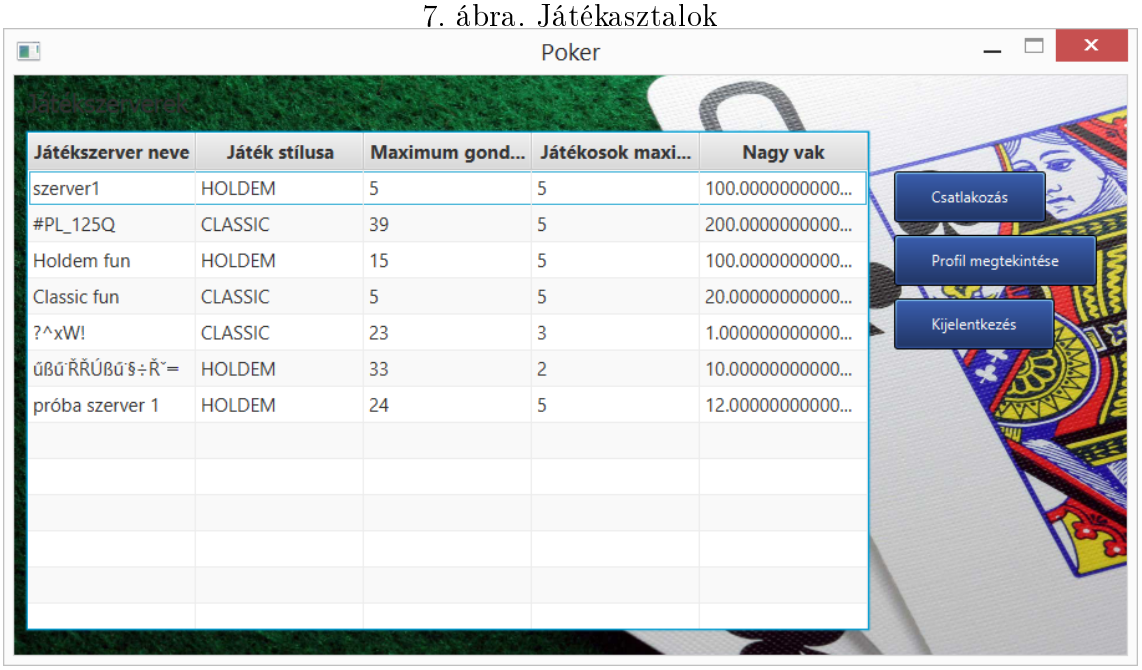
7. A póker játék használata

6. ábra. Bejelentkezési form



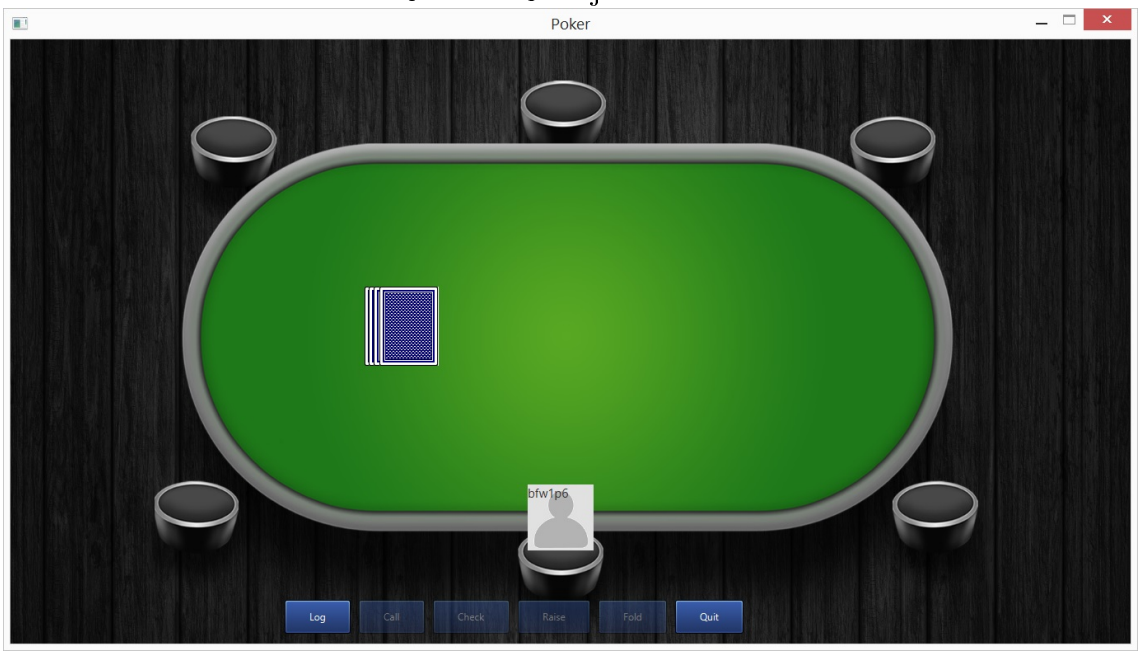
A játék indítása után az 6. ábrán látható bejelentkező felületet kapjuk. Ha még nem regisztráltuk magunkat a játékba, akkor kattintsunk a Regisztráció nevű gombra, amely felület a X. ábrán szerepel. Adjuk meg a regisztrálni kívánt felhasználó nevünket és jelszavunkat, majd regisztráljunk. (Képek kellenek ide, error, info?) A szerver értesít minket, hogy a művelet sikeres, vagy sikertelen volt. Ügyeljünk arra, hogy névütközéseket a szerver nem enged. Tehát, ha valaki XYZ névvel már regisztrálva van, akkor még egy ugyanolyan nevű felhasználót nem enged regisztrálni a szerver. Továbbá győződjünk meg arról, hogy a megadott jelszavak megegyeznek. Sikeres regisztrációt követően a program visszairányít minket a bejelentkezési formhoz, amelyet értelemszerűen kitöltve be tudunk jelentkezni a programba. A sikeres bejelentkezést követően a 7. ábrán látható felület fogad minket, ahol például asz-

7. ábra. Játékasztalok



talhoz csatlakozhatunk. Tetszőlegesen válasszunk ki egy asztalt, majd kattintsunk a Csatlakozás feliratú gombra. A program átirányított minket a 8. ábrán jelölt felületre. Az üres játék asztal (optimális esetben) két dolgot jelenthet, vagy azt, hogy az asztalnál éppen játszanak, vagy pedig azt, hogy az asztalnál mi vagyunk az egyedüli játékosok. Indítsunk el egy második klienst (hivatkozás?) is, majd csatlakozzunk ugyanahhoz az asztalhoz, amelyiknél az előző kliens helyet foglalt. A játék (optimális esetben) elindult, és ha az asztal klasszikus játékstílusú, akkor 9. ábrán látható felület fogad minket.

8. ábra. Üres játékasztal



9. ábra. Klasszikus játékstílusú asztal

III. rész

Fejlesztői dokumentáció

8. Felhasznált technológiák

ide kell egy kis bevezetés

8.1. Eclipse

A szakdolgozatom eclipse Mars [?] fejlesztőkörnyezetben írtam, amelyhez a quick search [?] nevű plugint is letöltöttem. A fejlesztést nagy mértékben megkönnyítette a fejlesztőkörnyezet használata, ugyanis többek között szintaktikus ellenőrzést, classpath ellenőrzést és ... végzett. Továbbá remekül testreszabható a „keymapping"-je, és rengeteg hasznos billentyű kombinációt használhatunk (pl. típushierarchia, típus keresés, resource keresés, stb.).

8.2. Maven

A népszerű buildrendszerből a 3.3.3-as verziót használtam, amelyhez igen sok különböző plugin szerezhető be. Eredetileg csupán a függőségek kezelésére alkalmaztam volna a programot, később egyre több és több pluginnal egészítettem ki, így egyre több feladatot volt képes ellátni a szoftver.

Adatbázis séma létrehozása és demó adatokkal való feltöltése

Az adatbázis sémát a fejlesztés elején mindig manuálisan készítettem el, majd rátaláltam a sql-maven-plugin nevezetű pluginra. A maven honlapján megtalálható teljeskörű dokumentáció segítségével bekonfiguráltam a plugint, így csupán a

```
mvn sql:execute@create-db
mvn sql:execute@create-triggers
mvn sql:execute@fill-db
```

utasításokat kellett kiadnom parancssorból, hogy friss adatbázissal dolgozhassam tovább. Igyekeztem a felesleges parancsismétléseket elkerülni, vagyis letöltöttem az exec-maven-plugin nevű plugint, amelynek segítségével sikerült a

```
mvn exec:exec
```

parancsra korlátozni az adatbázis újboli létrehozását. Próbáltam tovább általánosítani a feladatot, majd végül egy profilt készítettem el, amely az install fázissal együtt oldja meg az adatbázis kérdést. A

```
mvn clean install -Pbootstrap
```

paranncсsal nem csak a lokális repozitorinkba telepíthetjük fel a poker-persist modult, de még egy új adatbázist is kapunk.

Javadoc
A mavennel java dokumentációt is lehet generáltatni a

```
mvn javadoc:javadoc
```

parancs segítségével. Sőt, akár modul dokumentációt is lehet készíteni a

```
mvn site:site
mvn site:stage
```

utasításokkal. Az elkészült dokumentációk a modul `modul\target\staging` könyvtárba kerülnek. Kettő darab index.html állományt kell keresnünk, az egyik a gyökérkönyvtárban található a másik pedig az `apidocs` könyvtár gyökerében.

8.3. MySQL

8.4. Git

A fejlesztés során előfordult, hogy internet kapcsolat nélkül kellett verzióznom a kódomat. A git egyik legnagyobb előnye, hogy internet kapcsolat nélkül is lehet commit parancsot végrehajtani. A commit parancs hatására a lokális repozitorinkba egy új snapshot (pillanatfelvétel) készül a working directory tartalmáról, amelyet később push utasítással feltölthetünk egy távoli repozitoriba. A fejlesztéshez több branchet is használtam, így ha valami balul sült volna el, akkor egyszerűen visszaálltam volna a masterre.

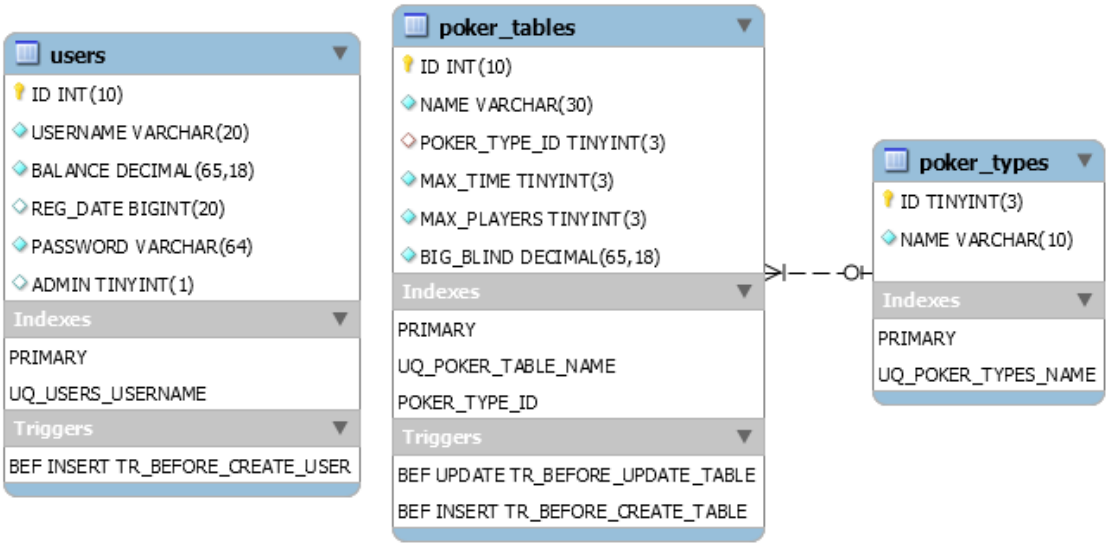
8.5. Enterprise Architect

A tervezés során igen nagy hasznát vettem ennek a szoftvernek. Rengeteg beállítás, funkció megtalálható a programban. Szinten minden testreszabható a segítségével. Az elkészült UML diagramokból pedig JAVA kódot tud generálni a program. A dokumentációban fellelhető osztály hiearchiák és UML diagramok ennek a szoftvernek a segítségével készültek el.

9. Adatbázis séma

Az adatbázis három darab táblából épül fel, amelyet a **??**. ábra szemléltet.

10. ábra. Adatbázis séma



9.1. Póker játék asztal tábla

Oszlopnév	Típus	Megszorítás
ID	UNSIGNED INT	Elsődleges kulcs a táblára nézve.
NAME	VARCHAR	Az oszlopban szereplő értékeknek egyedinek kell lenniük és legfeljebb 30 karakter hosszúak lehetnek.
POKER_TYPE_ID	UNSIGNED TINYINT	Az oszlop idegen kulcs a poker_types táblára nézve.
MAX_TIME	UNSIGNED TINYINT	Az értéknek bele kell esnie a [5-40] intervallumba.
MAX_PLAYERS	UNSIGNED TINYINT	Az értéknek bele kelle esnie a [2-5] intervallumba.
BIG_BLIND	DECIMAL	-

9.2. Felhasználók tábla

Oszlopnév	Típus	Megszorítás
ID	UNSIGNED INT	Elsődleges kulcs a táblára nézve.
USERNAME	VARCHAR	Az oszlopban szereplő értékeknek egyedinek kell lenniük és legfeljebb 20 karakter hosszúak lehetnek.
BALANCE	DECIMAL	-
REG_DATE	BIGINT	-
PASSWORD	VARCHAR	Az értéknek pontosan 64 hosszúnak kell lennie.
ADMIN	UNSIGNED TINYINT	Az érték csak 0 vagy 1 lehet.

9.3. Játékstílus tábla

Oszlopnév	Típus	Megszorítás
ID	UNSIGNED INT	Elsődleges kulcs a táblára nézve.
NAME	VARCHAR	Az oszlopban szereplő értékeknek egyedinek kell lenniük és legfeljebb 10 karakter hosszúak lehetnek.

10. Megoldási terv

10.1. Probléma

Egy hálózaton keresztül működő póker játékot szolgáltató kliens-szerver programcsomagot fogok megvalósítani JAVA programozási nyelven. A felhasználói és szolgáltatással kapcsolatos adatokat egy adatbázisban helyezem el, így is biztosítva a játék menet konzisztenciáját és konkurencia vezérlését. Az említett adatabázis többek között rögzíti az olyan profil adatokat, mint a felhasználói név, a jelszó vagy a regisztráció dátuma. Az alkalmazás szerver-oldali komponensei több olyan asztal szimultán futtatását biztosítja, amelyek különböző paraméterezésűek lehetnek. Ugyanis felhasználói oldalról különböző igények léphetnek fel (pl.: a tapasztaltabb játékosok rövidebb idő alatt tudnak reagálni egy leosztásra). Az asztalok leglénye-

gesebb paraméterei a játék stílus, a játék gyorsasága, illetve az elfoglalható helyek száma lesz, de ezeken felül további paraméterek is megadhatóak egy konkrét asztal instancia létrehozásakor. Ezen műveletek ellátására, azaz az asztalok konfigurálására, egy grafikus interfészt valósít meg a programcsomag.
A játékban a felhasználók jogok alapján is megkülönböztethetőek lesznek, így a magasabb jogkörrel rendelkező felhasználók speciális műveleteket hajthatnak végre a játékon és az adatbázison egyaránt. Ebből kifolyólag külön adminisztrációs felület is megvalósításra kerül majd, amelyen keresztül adott esetben akár rendezni (név, regisztráció dátuma, stb. alapján), ill. törölni is lehet majd felhasználókat, sőt a különböző felhasználóknak különböző jogokat is itt lehet majd kiosztani. Továbbá a felhasználók könnyen tudják majd menedzselni a saját felhasználói fiókjukat, ugyancsak egy grafikus interfész segítségével.

10.2. Tervezés

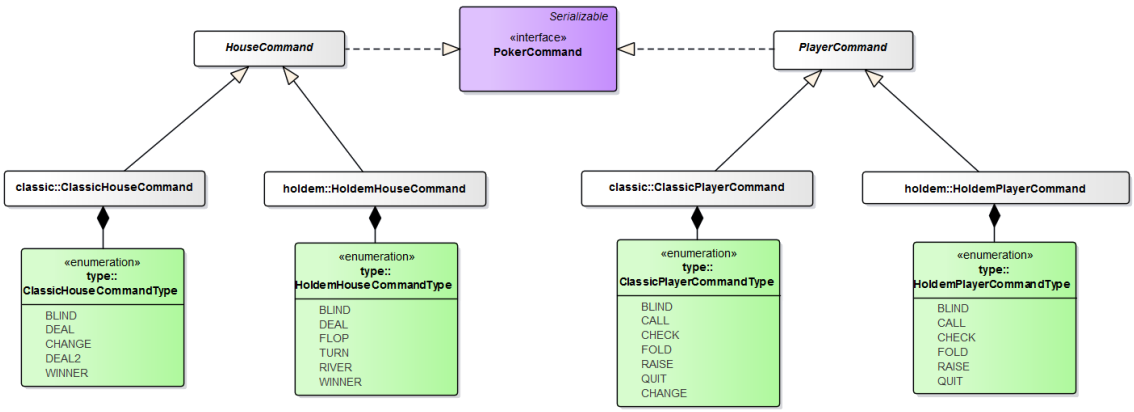
A probléma leírásából fakadóan egyértelműen kiderül, hogy legalább kettő darab modulra lesz szükségünk, egy kliens modulra és egy szerver modulra. A tervezés során végig kell gondolni, hogy mik azok az igen fontos nagyvonalú (nagy horderejű, ... valahogy meg kéne fogalmazni) események amelyek a program futása során bekövetkeznek, illetve bekövetkezhetnek. A szerver és a kliens hálózaton keresztül fognak kommunikálni, így kínálkozik a lehetőség az RMI architektúrára való építkezésre. Ha RMI, akkor egy jóldefiniált publikus interfészt kell elkészíteni, amely a szerver vázáért felel. A kliensek ezen az interfészen keresztül hívnak be a szerverhez. A probléma megfogalmazásából kiderül, hogy melyek azok az alapvető funkciók és szolgáltatások, amelyeket a szervernek biztosítania kell. Ezen megfogalmazás alapján már körvonalazódik a szerver modul feladata. A probléma leírásából a szerver és a kliens között kiépített kommunikációs rendszer csak implicit megfogalmazásban szerepel. Az, hogy hogyan és miként, illetőleg milyen szekvencia szerint fognak egymással kommunikálni a modulok az explicite nincs felvázolva. Az implementáció során ügyelni kell arra is, hogy párhuzamosan akár több kliens is csatlakozhatva lehet a szerverhez, illetve az egyes játék asztalokhoz. Az üzenetek egy előre meghatározott sorrend alapján küldhetőek és fogadhatóak. Fontos megjegyezni, hogy nem csak a kliensek küldhetnek üzeneteket a szervernek, hanem a szerver is értesítheti a klienseket a szerveren történő eseményekről, mint például, ha egy adott asztalnál egy játékos CALL utasítást küld a szervernek, akkor a szerver az asztalnál ülő játékosokat értesíti a bekövetkezett eseményről. Ennek megfelelően a kliensek mindig értesítést kapnak az őket érintő eseményekről. Ezáltal a kliens modulnak is meg kell fogalmazni egy publikus interfészt, amelyet a szerver fog felhasználni. Fontos megjegyezni, hogy a kliensek közvetlenül egymásnak nem küldhetnek üzeneteket. A

szervert úgy kell implementálni, hogy a béerkező üzenetek alapján az összes megfelelő klienst értesítse.
Az üzeneteknek kettő fajtáját különböztetjük meg

- Szerver által küldött üzenet
- Kliens által küldött üzenet

A megfelelő absztrakció elérése érdekében a **??**. ábrán látható PokerCommand interfésszel össze kell fogni az üzenet fajtákat. A **??**. ábráról leolvasható, hogy minden

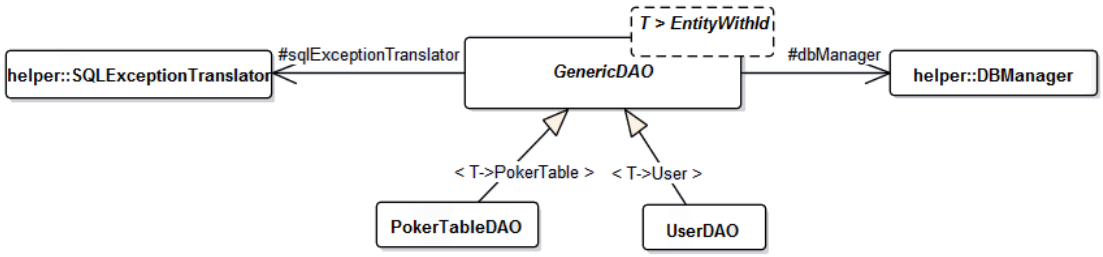
11. ábra. Póker üzenetek felépítése



üzenetnek van egy típusa. Az üzenet típusa függ az üzenet fajtájától és specializációjától is. Például ha egy HOLDEM játékstílusú asztal be szeretné kérni a vakokat a játékosoktól, akkor egy HoldemHouseCommand típusú HouseCommandType::BLIND utasítástípusú új objektum példányt kell szétküldenie az asztalnál ülő összes játékosnak. Ezt az üzenetküldést a szerver a kliensek publikus interfészén keresztül teheti meg.
A kliensnek az üzenetet fogadnia kell. Az üzenet fogadására egy új, ezt a funkciót ellátó objektumot kell definiálni.
A kliens oldal architektúrális felépítésének az alapját a Model-View-Controller tervezési minta adja. A beérkezett üzenetet az erre kijelölt objektum fogadja, majd továbbítja a kliens oldali vezérlő rétegnek. A vezérlő réteg értelmezi a bejövő üzenetet, majd ennek megfelelően a model rétegnek és a megjelenítési rétegnek is továbbítja a beérkezett üzenetet további feldolgozásra. A probléma különböző funkciók leírását is tartalmazza. Célszerű, hogy a különböző funkcióknak különböző grafikus interfészt biztosítsunk. Ugyanakkor új felület létrehozásakor új vezérlő objektumot is létre kell hoznunk, viszont a model rétegnek statikusnak kell lennie. Ugyanaz a model objektum lássa el például a bejelentkezési és regisztrációs kérelmeknek a továbbítását a szerver. Tehát különböző funkciók elérése esetén a vezérlési és megjelenítési réteget

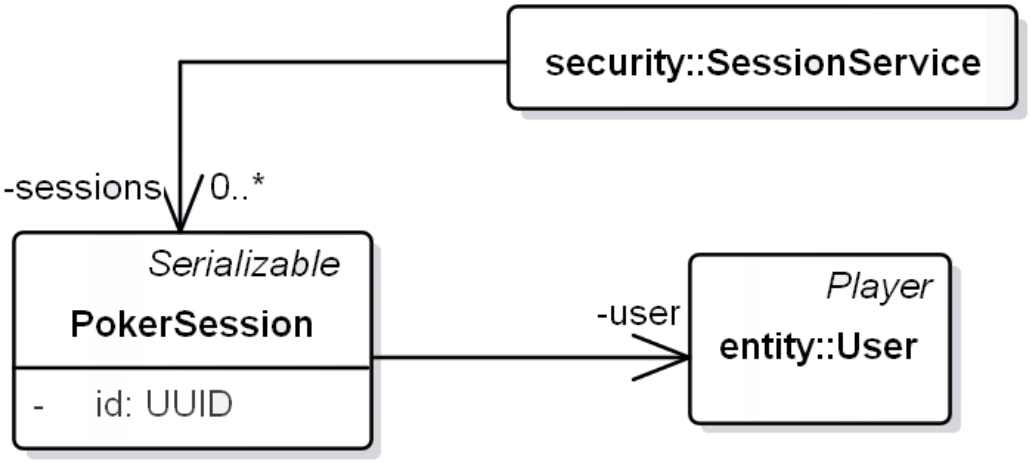
ki kell cserélni, viszont a modelt nem. Illetőleg kettő darab modelt kell megfogalmaznunk, az egyik látja el például a fentebb említett egyszerűbb funkciók meghívását a szerveren, a másik pedig a tényleges játék közbeni model, amelyet a vezérlői réteg értesíteni tud a beérkezett üzenetekkel kapcsolatban és kliens oldalon biztosítja a játék menetet.
A probléma leírásában szerepel adatbázissal való kommunikáció is. Erre ugyancsak egy új modult kell bevezetni, amely az adatbázissal történő kommunikációt bonyolítja le. Entitás osztályokat is kell írni és olvasni az adtbázisból, így ezeket az osztályokat általánosan kell megfogalmazni a kód duplikációk elkerülése végett. Ezen modul vázlatát a ??. ábra szemlélteti.
Az adatbázissal való kommunikációért felelős modul nem csak entitásokat tartalmaz,

12. ábra. Az adatbázissal kommunikáló modul



hanem további kettő darab objektumra is szükség lesz. Az SQLExceptionTranslator fontos része a modulnak. Az adatbázis triggerek és megszorítások által definiált hibaüzeneteket fordítja át JAVA kivételobjektumokra, amelyek a kódbeli feldolgozást könnyítik meg, továbbá ezek a saját kivételobjektumok már a felhasználó számára is értelmes hibaüzenetekkel bírnak. A DBManger osztály felelős az adatbázissal kiépített kapcsolat fenntartásáért és lezárásáért.
A szervernek szerves része egy objektum, amely a munkamenetek kezeléséért felelős. Az autentikált felhasználók egy-egy munkamenettel azonosíthatóak. Az autentikációért felelős részmodult a ??. ábra vázolja fel. A SessionService tartja nyilván a munkameneteket. Egy felhasználóhoz legfeljebb egy munkamenet rendelhető egy adott időben. Továbbá egy munkamenethez pontosan egy felhasználó tartozik. Ha a felhasználó kijelentkezik a póker játékból, akkor a felhasználó munkamenetét érvényteleníteni kell. Ugyanaz a munkamenet többször nem osztható ki, törölni kell. Ugyanakkor egy felhasználó különböző időpillanatokban rendelkezhet különböző munkamenetekkel. Tehát ha az imént kijelentkezett felhasználó újból be szeretne jelentkezni a játékba, akkor egy új munkamenettel kell azonosítani. A szervernek biztosítania kell, hogy a kliensek rendelkezzenek munkamenet objektumokkal, ugyanis bármilyen funkció hívása esetén a kliensek ezzel az objektummal azonosítják magukat. Pontosabban fogalmazva minden munkamenet egy-egy UUID értékkel van ellátva, és a klienseknek elég csak ezt az egyedi értéket elküldeni a szervernek. A felhasználóknak

13. ábra. A munkamenetek tárolása szerver szinten



nem engedhetjük meg, hogy egy felhasználónevet konkurensen használhassanak. Ez a felügyelet ugyancsak a SessionService objektum feladata.
A szerver és a kliens közötti bárminemű tevékenység, illetve kommunikáció egy közös modult kell, hogy felhasználjon. Például ha kivételobjektumot szeretnénk dobni szerver oldalon és azt kliens oldalon szeretnénk lekezelni, akkor értelemszerűen mind szerver mind a kliens classpathján szerepelnie kell az adott kivételobjektumnak. A közös modulban kell szerepelnie a szerver és a kliens publikus interfészeknek, a kommunikációs rendszert alkotó osztályok összessége és a saját kivételosztályok definíciói.
Külön modul fogalmazható meg a játékosok lapjainak a kiértékelésére is. Ez a modul nem kerül implementálásra, hanem külső könyvtárként kerül felhasználásra.

10.3. Implementálás

Az implementálást a kliens oldali megjelenítéssel kezdtem egy maven archetype generator segítségével. Így generálódott egy megfelelő kiindulási alap, amiből elkezdtem építkezni. Legelőször a bejelentkezési és a regisztrációs formot állítottam össze. Ezekután pedig a szerver publikus interfészének egy részét fogalmaztam meg kód szinten. Ezt az interfészt a kliens már el tudta kérni az RMI registry-ből, és sikeresen be tudott hívni a szerverhez. A szerver és a kliens kommunikációjának az alapjának a kiépítése után az adatbázis feltelepítése, elindítása majd elérése volt a legfőbb célom.

10.4. Elemzés

11. Modulok

A programcsomag 6 fő modult tartalmaz

poker-server

A póker játék szervere, amely magát a játékot szolgáltatja.

poker-client

A póker játék kliense, amely segítségével a szerverhez lehet csatlakozni.

poker-shared

A póker játék azon modulja, amelytől a szerver és a kliens egyaránt függ.

poker-persist

Az adatok letárolásáért felelős modul.

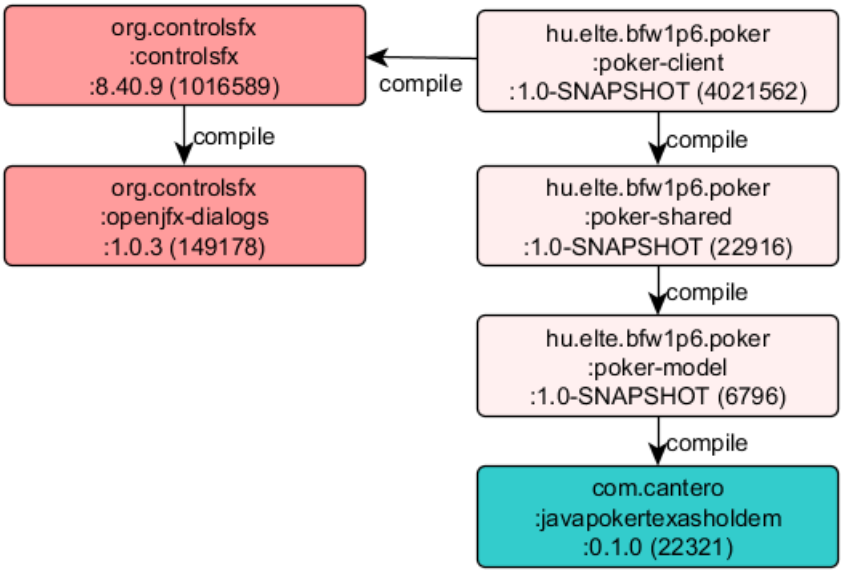
poker-model

A póker játék modellezéséért felelős csomag.

javapokertexasholdem

Külső könyvtár, amely a nyertes játékos kiértékelési feladatot végzi.

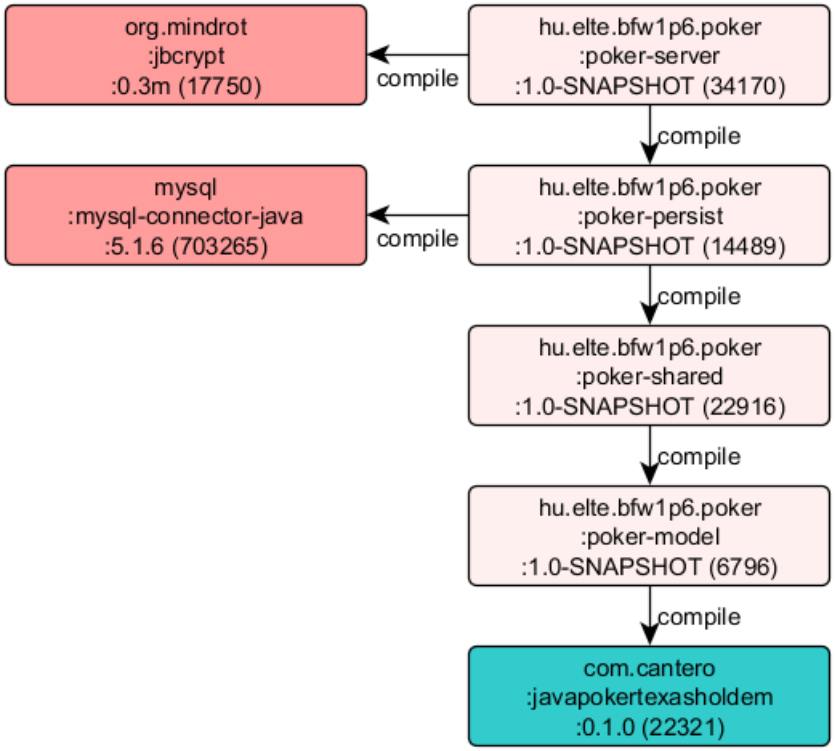
14. ábra. Kliens modulra bontása



A modulok közötti függőséget a **??**. és a **??**. ábra szemlélteti.
A programcsomag két főmodulra bontható

- poker-server
- poker-client

15. ábra. Szerver modulra bontása



11.1. Model modul

A poker-model modul felelős a játék modellezésért. A X. ábrán látható a modul felépítése. Az entitásokat össze kellett fogni az EntityWithId interfész segítségével, ezáltal a DAO (későbbi képre hivatkozás...?) objektumot generikusan lehetett megfogalmazni, amely segített a kód duplikációk elkerülésében. A PokerTable és a User osztályok példányai feleltethetőek meg a poker_tables és a users adattáblák egyes rekordjainak. A PokerType enumeráció a játékstílusokat határozza meg. Ha egy új játékstílust szeretnénk hozzáadni a játékhoz, akkor a kódban egy új PokerType enum objektumot kell felvennünk, és a poker_schema.sql állományban pedig fel kell venni egy új insert utasítást a poker_types táblára nézve az új játékstílus nevére vonatkozólag. Természetesen ez a pár sor újonnan hozzáadott kód nem azt jelenti, hogy már használhatjuk is az új játékstílust. A programcsomagot több ponton is ki kell bővíteni

- Szerver oldal
    1. AbstractPokerTableServer osztályt ki kell terjeszteni egy új osztállyal
- Kliens oldal
    1. Új leszármazott osztályokat kell létrehozni a kliens oldali MVC absztrakciós osztályaiból (ld. később) vagy hivatkozva...

2. Új fxml állományt kell létrehozni

11.2. Shared modul

A poker-shared modul olyan közös osztályokat tartalmaz, amelyeket a kliens és a szerver egyaránt használ. Itt helyezkednek el az egyéni kivételek, a kommunikációs rendszer definíciója és a kliens, mint megfigyelő interfésze.

Kivételek

A játékcsomagnak szüksége van egyéni kivételekre. Az olyan speciális eseteket, mint például adabázisba írás, illetve abból olvasás közbeni fellépő kivételt saját kivételobjektumokkal célszerű lekezelni. Ehhez hasonló egyéni kivételek fellépthetnek bejelentkezéskor, asztalhoz való csatlakozáskor, hibás jelszó megadás esetén, stb (ld. melléklet, javadoc...).

Kommunikációs rendszer

A szerver és a kliens üzeneteket küldhetnek egymásnak.
Ide jöhet a kép... Az ún. messaging systemet igyekeztem minél absztraktabb módon megfogalmazni, ezzel is elősegítve a későbbi bővíthetőségi nehézségeket. Azonban sajnos helyenként fellelhető kód duplikáció az enumerációkra való építkezés miatt.
Minden utasításnak kötelezően implementálnia kell a PokerCommand interfészt, ezzel is elősegítve az általános megfogalmazást a rendszerben. Két fajta utasítás létezik a játékcsomagban:

- Szerver utasítás
- Kliens utasítás

Továbbá a játékstílusnak megfelelően tovább szigorodik... (kéne egy jó szó) a küldendő üzenet fajtája. A programcsomag kettő játékstílust fogalmaz meg (ld. 5.2). A játékstílusok különböző utasítás fajtákat igényelnek. Értelemszerűen, ha egy Holdem játékstílusban résztvevő kliens üzenetet szeretne küldeni a játék asztal szervernek , akkor egy HoldemPlayerCommand típusú objektumot kell elküldenie a szerver csonkon keresztül.
Az üzeneteknek vannak fajtái, amelyeket a mellékletben meg lehet tekinteni.

Observer pattern [?]

Az oberserver tervezési minta segítségével megvalósítható RMI API-n keresztül az ún. event driven server. Így nem csak a kliens tud üzenet küldeni a szervernek, hanem a szerver is tudja értesíteni a klienseket. Például, ha egy játékos CHECK típusú utasítást küldött a szervernek (játékstílustól függetlenül), akkor azt az üzentet a

szerver minden kliensnek szétszórja (broadcast, üzenetszórás... valahogy jól kéne megfogalmazni).

Session

A szerver a klienseket session objektumokkal azonosítja. Minden kliens kap egy sessiont a bejelentkezéskor. A session addig él, amíg a felhasználó ki nem jelentkezik, akkor ugyanis a munkamenet érvénytelenítésre kerül. Illetőleg a munkamenetet akkor is érvényteleníteni kell, ha a kliens nem jelentkezett ki, de a kapcsolat valamilyen oknál fogva megszakadt.
Amikor egy kliens be szeretne jelentkezni a játékba, akkor a szerver az összes megszakadt kapcsolatú klienst felderíti, és a munkamenetüket érvénytelennek tekinti. A felderítés ún. pingeléssel történik. A szerver minden csatlakozott klienst megpróbál elérni, és amelyik kliensnél megszakadt a kapcsolat, azt eltávolítja a szerverről. Ezután a SessionService ellenőrzi, hogy az adott felhasználónévvel van-e aktív munkamenet, ha van, akkor kivételt dob az eljárás, különben az autentikáció sikeres, és a kliens oldali model eltárolja a loginkor kapott sessiont.

11.3. Persist modul

A poker-persist modul látja el az adatbázissal kapcsolatos teendőket. Ez a modul írja és olvassa az adatbázist a bejövő kérések és paraméterek alapján.

Generikus DAO

A generikusság elengedhetett a kódismétlés elkerülése végett. Általánosan kell megfogalmazni az entitások viselkedését **??** (vagy fordítva...), és ezáltal a persist modul letisztult arculatot kap. A GenericDAO osztály tartalmaz minden olyan elemet, amelykre a specializálódott DAO-knak szükségük lehet.

Adatbázis menedzser

Az AbstractDAO-nak szükséges van tényleges adatbázis kapcsolatra, amelyet a DBManager osztály szolgáltat.

Kivételek átfordítása

Az AbstractDAO rendelkezik egy SQLExceptionTranslator objektummal is, amely az adatbázisból érkező hibát fordítja át PokerDataBaseException típusú kivételre. A PokerDataBaseException kivétel osztály a felhasználó számára is értelmes hibaszöveget hordoz magában.

11.4. Kliens modul

A kliens modult és annak minden függőségét egy jar fileba csomagolva kell szétterjeszteni a felhasználók között, akik majd ténylegesen használni fogják a programot. A **??**. ábra alapján a jar file tartalmazni fog minden olyan osztályt és interfészt, amelyre ténylegesen szüksége lesz a kliens programnak. Az összecsomagoláshoz igénybe vehetjük a mavent és annak az assembly pluginját. A poker-client könyvtárban a

```
mvn clean compile assembly:single
```

paranccsot kiadva a poker-parent/release könyvtár alá csomagolódik be a futtatható jar állomány.

Model-View-Controller [?]

Az igen közkedvelt tervezési minta alapján valósítottam meg a kliens oldali programot.

Model

A kliens kettő darab model osztályt tartalmaz. Az Model nevű singleton osztály felelős a szerver felé irányuló kapcsolat kiépítésében és fenntartásában. Ez az egyszerű szerver hívásokat végzi el. A kliens indulásakor elkéri a registry-ből a szerver csonkot, és bejelentkezéskor eltárolja a szerver által generált munkamenet objektumot. A model segítségével hívhatóak meg a játék funkcióinak a zöme. Ide sorolható a játék minden olyan funkciója, amely nem játék asztalhoz köthető. Például ennek a modelnek a segítségével lehet bejelentkezni, asztalt és felhasználót módosítani, vagy akár a játékasztalokat elkérni a szervertől.
A kliens tartalmaz egy másik modelt - AbstractMainGameModel -, amely a tényleges játékért felel kliens oldalon. A szerver a parti megkezdésekor leküldi a klienseknek, hogy az adott partiban ki hol ül az asztalnál, hány játékos ül az asztalnál, a játékosok neveit és még néhány inicializációs értéket. Ezen értékek alapján a model el tudja dönteni, hogy az adott kliensnek kötelezően be kell rakni vakot. Minden parti elején minden kliensnek az adóssága a nagy vakkal egyezik meg. A további körökben az adósság leróható.

View

A megjelenítésért a JavaFX 2 [?] felel. Az alapkonstrukció, hogy a GUI-n megjelenő objektumok zömét fxml állományokba szervezzük és a megjelenítés színtérgráffal történik. Minden fxml állományhoz pontosan egy controller tartozik, amely az fxml állomány minden újratöltésekor inicializálódik. A controllert az adott fxml állomány gyökerébe kell bekötni.

A programcsomag által definiált játékstílusok igen hasonlóak, ezért egy AbstractMainView osztályban megfelelően kiszervezhetőek a közös részek, majd ennek az osztálynak a specializációiban (ClassicMainView, HoldemMainView) kerülnek megvalósításra a tényleges játékstílusbeli különbségek. Például eltérő különbség , amely a GUI-n is megjelenik, hogy a játékosok hány darab kártyalapot kapnak kézbe, illetve a Texas Hold'Em játékstílussal ellentétben a classic játékmódban nincsenek közös kártyalapok.

Controller

A controller általános szerepét ugyancsak egy abstract osztály, az AbstractMainGameController tölti be. A közös részek nem csak a megjelenítés és a model szintjén figyelhetőek meg, hanem az őket összekötő controller szintjén is. A beérkező utasítások alapján a controller hív tovább a megjelenítési és a model rétegbe.

Communication controller

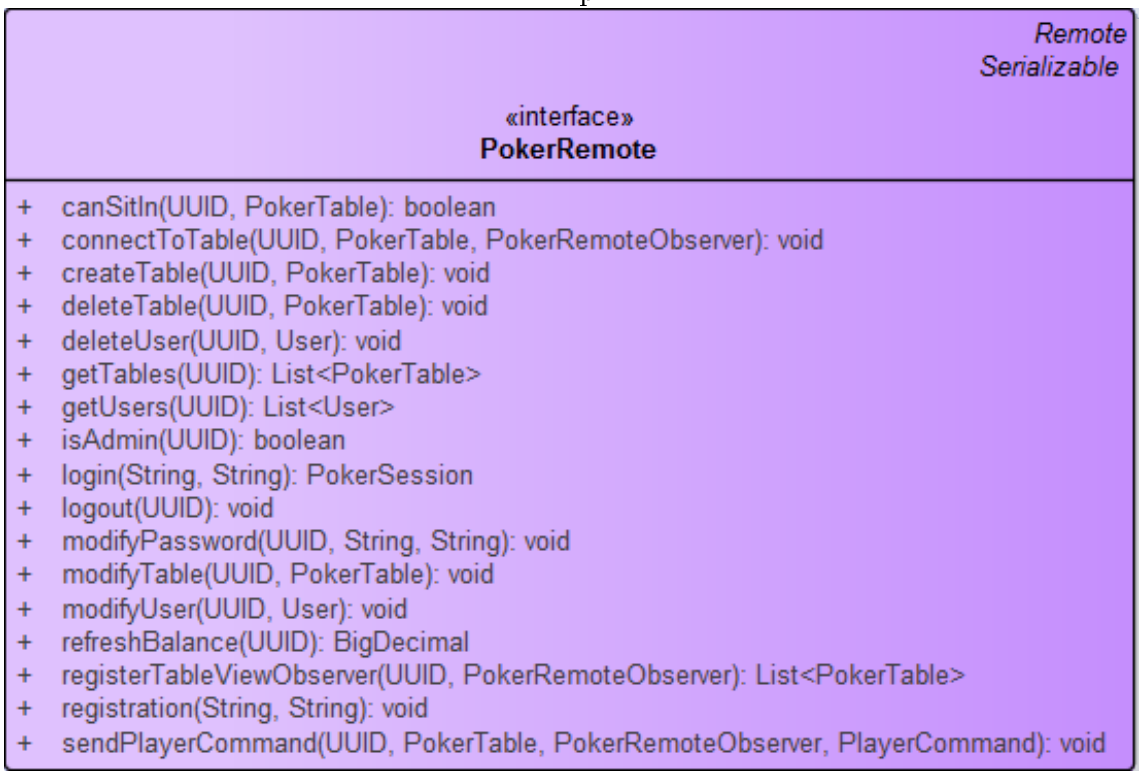
PokerRemoteObserver

FrameController

11.5. Szerver modul

A szerver feladata a játék biztosítása a kliensek számára. A **??**. ábrán látható a szerver jóldefiniált publikus interfésze. A kliensek ezt az interfészt tudják elkérni a registry-ből [**?**]. A PokerRemote interfészt a PokerRemoteImpl osztály implementálja, ennek egy instanciája a tényleges szerver objektum. A szerveren futnak a játékasztal-szerverek, amelyek egy-egy asztalnak feleltethetőek meg, amelyekhez a kliensek csatlakozni tudnak. Ha egy játékos csatlakozik egy játékasztalhoz, akkor az adott asztal feljegyzi a csatlakozott klienst egy lokális listába. A főszervernek továbbra is van referenciája a klienshez, ugyanis minden klienstől érkező utasítást a főszerver fogad, és irányít a megfelelő játékasztal-szerverhez. Erre az autentikáció miatt van szükség. Azonban ez a fajta megközelítés könnyen szűk keresztmetszetté válhat, hiszen (vagy ezt inkább tovább fejlesztési lehetőségek...)
A szerver a jelszavak titkosítására bcrypt eljárást alkalmaz, amelynek a biztonságát sózással növeli. Továbbá a szerver felhasznál még egy külső csomagot - mysql-connector-java -, amely az adatbázis kapcsolatért felel.

16. ábra. A szerver publikus interfésze



11.6. javapokertexasholdem modul

Külső könyvtár, amely eredetileg Texas Hold'Em játékstílusban tud kezeket kiértékelni. Az integrációt viszonylag könnyen meg lehetett oldani, ráadásul még általánosítani is sikerült a könyvtárat. Tehát classic és Texas Hold'Em játékstílusra is ugyanezt a könyvtárat használja a szerver a kezek kiértékelésére.

12. Funkciók

A szerver vázáért a PokerRemote interfész felel, amely a játékon végrehajtható műveletek összefogásáért felel. Itt található az összes funkció, amely megvalósításra került, mint például játék asztal létrehozása, új felhasználó létrehozás, admin jog kiosztása stb. A kliens ezt a vázat tudja elkérni az RMI registry-ből, mint kliensoldali szervercsonk, amelyeken a műveletek meg tudja hívni.

12.1. Általános funkciók

Regisztráció

A játékot csak regisztrált felhasználók tudják használni. A szerver biztosít regisztrációs lehetőséget a felhasználók számára. A regisztráció során felhasználónevet és jelszót köteles megadni a felhasználó, amely tárolásra kerül az adatbázisban. Minden

újonnan regisztrált felhasználó 1000 egységnyi egyenleggel rendelkezik. Az adatbázis séma nem engedi, hogy több azonos felhasználónévvel különböző felhasználók regisztrálva legyenek, erre a regisztráció során ügyelnie kell a felhasználónak.

Bejelentkezés

A játék használatához a játékosoknak be kell jelentkezniük. Egy felhasználónévvel egyidőben csak egy munkamenet létezhet. Ha a felhasználó már használatban lévő felhasználónévvel kíván beljelentkezni, akkor a szerver hibás bejelentkezési adatokra hivatkozva megtagadja az utasítást. Sikeres bejelentkezést követően a szerver példányosít egy PokerSerssion objektumot egy UUID [?] azonosítóval, amelyet lokálisan eltárol, majd a kliensnek elküld. A kliens a kijelentkezésig megőrzi ezt az objektumot, és a későbbi funkciók hívása során ezzel a munkamenettel azonosítja magát.

Kijelentkezés

A játékosok a kijelentkezés funkcióval hagyhatják el a játékot. A funkció hivása során a kijelentkezni kívánó felhaszáló munkamenetét a szerver érvényteleníti, és a felhasználó visszakerül a bejelentkezési formhoz. A szerver (korlátozott mértékben) fel lett készítve a nem szabályszerű kijelentkezésre, tehát ha például megszakad a kapcsolat a klienssel, akkor a legközelebbi bejelentkezési funkció hívása során a szerver érvényteleníti a megszakadt kliensek munkameneteit, ezzel elérve azt, hogy a foglalt felhasználónevekkel újra be lehessen jelentkezni.

Asztalok

Létrehozás

A funkció eléréséhez admin jogosultságra van szükség. Egy új asztalt lehet lérehozni az adatbázis séma (hivatkozás) irányelvei szerint megfelelően paraméterezve. Amikor a kliens asztalt szeretne létrehozni, akkor a szerver egyértelműen tájékoztatja, hogy az asztal létrehozás sikeres volt-e. Ha igen, akkor egy felugró üzenet jelenik meg a képernyőn, hogy az asztal létrehozása sikeres volt, majd a felhasználó vissza lesz irányítva a tábla listázó felületre. Ha az asztal létrehozása sikertelen volt, akkor a felület ugyancsak egyértelműen jelzi a felhasználónak, hogy a művelet miért hiúsult meg. Például előfordulhat hibás beviteli érték, szerverhiba és adatbázishiba is.

Módosítás

Egy már meglévő asztalt adminisztrációs joggal rendelkező felhasználó módosíthat. Fontos megjegyezni, hogy a módosítás csak üres asztalokra alkalmazható.

Törlés
A funkcióhoz ismételten adminisztrációs jog szükséges. Már létező asztal törölhető a funkció segítségével. A funkciót csak üres asztalokra lehet alkalamazni.

Lekérés
A bejelentkezést követően a felhasználó a táblalistázó felületre kerül, ahol megtekintheti a létrehozott és elindított játéktábla-szervereket (hivatkozás). A bejelentkezést követően a kliens ezeket az asztalokat elkéri a szervertől. Fontos megjegyezni, hogy ha bármelyik asztal módosításra került, akkor az asztalokat a szerver újra leküldi a klienseknek, így minden kliensnél az asztalok legfrissebb változata lesz elérhető.

Csatlakozás
A felhasználók létrehozott játékasztal-szerverekhez csatlakozhatnak. Fontos megjegyezni, hogy egy asztalnál maximum 5 játékos ülhet egyszerre, tehát ha egy asztal tele van (5 játékos ül az asztalnál, beleszámolva a következő partira váró játékosokat is), akkor kliens oldali kód nem engedi beülni a felhasználót az adott asztalhoz, amelyről a felhasználót a képernyőn értesíti. Ha egy tele asztaltól kilép az egyik játékos (vagy megszakad a kommunikáció), akkor az asztalnál felszabadul egy hely, amelyet egy új felhasználó elfoglalhat, ekkor az asztal ismét tele lesz.

Szabad helyek lekérdezése
Ha egy kliens be szeretne ülni az egyik játékasztal-szerverhez, akkor előbb le kell kérdeznie, hogy van-e még szabad hely az asztalnál.

Felhasználók

Létrehozás
A regisztráció során új felhasználó entitás jön létre, amelyet a szerver letárol az adatbázisban. A funkció külön, direktbe nem hívható, regisztrációhoz kötött.

Egyenleg visszaállítás
A felhasználók egyenlegét adminisztrációs joggal rendelkező felhasználó vissza tudja állítani alaphelyzetbe, amely 1000 egységnek felel meg.

Törlés
Ugyancsak adminisztrációs jogot követel meg egy felhasználó végleges eltávolítása a póker játékból.

Lekérés

Az adminisztrációs joggal rendelkező felhasználók lekérdezhetik az adatbázisból az összes regisztrált felhasználót. A felhasználók a képernyőn táblázatos formában jelennek meg, amelyből felhasználót ki lehet választani a további funkciók eléréséhez.

Jelszó csere

A felhasználóknak lehetőségük van jelszócserére.

Adminisztrációs jog

A szerverre beregisztrált felhasználók alapesetben nem rendelkeznek ilyen jogosultsággal. Az adminisztrációs jogot csak adminisztrációs joggal rendelkező felhasználó tudja kiosztani illetve megvonni. Friss adatbázis esetén létezik egy admin nevű felhasználó, aki adminisztrációs joggal rendelkezik. Kezdetben csak ez a felhasználó tud kiosztani adminisztrációs jogot.

12.2. Játékmenetbeli funkciók

Utasítás küldés

Egyenleg lekérése

13. Tesztelés

13.1. Funkcionális tesztelés

Funkció	Elvárt eredmény	Eredmény
Regisztráció	A felhasználó a regisztráció gomb megnyomása után tájékoztatást kapjon afelől, hogy a regisztráció sikeresen megtörtént.	A program az elvártnak megfelelően működött.
Bejelentkezés	A bejelentkezési formot helyesen kitöltve és a bejelentkezés gombra kattintva a program sikeresen autentikálta és átirányította a felhasználót a táblalistázó felületre.	A program az elvártnak megfelelően működött
Játék asztalhoz való beülés egyedüli játékosként	Az asztal betöltődik és a felhasználó profilján kívül a grafikus interfészen más profilkép nem jelenik meg. Tetszőleges idő eltelte után sem indulhat el a játék, kivétel ha legalább egy másik felhasználó csatlakozik az adott játék asztalhoz	A program az elvártnak megfelelően működött.
Játék asztalhoz való beülés nem egyedüli játékosként	Ha az asztalnál éppen parti zajlik, akkor ebből az újonnan csatlakozott játékos semmit nem érzékel. A játékos a következő partiba szállhat be. A következő parti megkezdődik, ha legalább kettő darab játékos ül az asztalnál.	A program az elvártnak megfelelően működött.
Tábla módosítás	-	-
Tábla törlés	-	-

14. Tovább fejlesztési lehetőségek

- Az adatbázis viszonylag alacsony absztrakciós szinten került implementására, azonban mivel néhány tábláról beszélhetünk csak, ezért igyekeztem elkerülni a keretrendszerek általi overheadet. Ugyanakkor ezen a ponton sokat fejlődhet a programcsomag, ha a későbbiek során esetlegesen bonyolultabban kellene modellezni a játékot adatbázis szempontjából. Például dialektusok - akár Liquibase (hivatkozás) - használata elfedheti a tényleges adatbázis-kezelő rendszer általánosságait, így eggyel magasabb szintre helyezhető a megvalósítás.

- A felhasználói élményen sokat javíthat az animációk használata. A megjelenítés sokkal lágyabb, folyékonyabb lehetne Transition/Animation (bibliográfiába hivatkozás...) objektumok használatával.

- Akár a komplett RMI architektúrát le lehetne váltani, és helyette REST szoftverarchitektúrát alkalmazni, amely modernebb megjelenést (AngularJS, reszponzív design) és modernebb fejlesztői eszközöket, API-kat vonna maga után.

- A játék nem képes kezelni olyan eseteket, amikor egynél több játékos nyer az adott körben.

- A játék nincs felkészítve arra a szélsőséges esetre, ha valakinek elfogy a zsetonja, akkor pontosan minek (és hogyan) kell történnie.

- A kódban viszonylag sok kód duplikáció van jelen, ugyanis az HouseCommandType és a PlayerCommandType enum típusú objektumok szűk keresztmetszetnek tudható be. Ha a PokerCommand interfacet implementáló osztályokat generikusan tudnánk megfogalmazni, akkor jelentősen letisztulna a kód.

- Az admin jogot el lehetne távolítani a játékból, és helyette MVC tervezési minta alapján a szerver oldalra is implementálni lehetne egy grafikus interfészt, amelyen keresztül a szerver teljeskörű karbantartása és adminisztrációja elvégezhető lenne.

- A programcsomag a váratlan adatbázis hibákat nem tudja megfelelően lekezelni. Ezek javításra szorulnak.

IV. rész

Irodalomjegyzék

15. Hivatkozások

[1] Ötlapos póker
https://hu.wikipedia.org/wiki/ÃŰtlapos_pÃşker

[2] Texas Hold'Em
https://hu.wikipedia.org/wiki/Texas_Hold'Em

[3] Java SE Runtime Environment 8 letöltése
http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

[4] MySQL Community Server 5.6
https://dev.mysql.com/downloads/mysql/5.6.html

[5] Kártyakombinációk
https://hu.wikipedia.org/wiki/KÃąrtyakombinÃąciÃşk_(pÃşker)

[6] Java Poker Texas Holdem Hand Evaluator
https://github.com/phstc/javapokertexasholdem

[7] bcrypt
https://en.wikipedia.org/wiki/Bcrypt

[8] Java RMI
https://hu.wikipedia.org/wiki/Java_remote_method_invocation

[9] Observer pattern
https://en.wikipedia.org/wiki/Observer_pattern

[10] Model-View-Controller
https://en.wikipedia.org/wiki/Model-View-controller

[11] Eclipse Mars IDE
https://projects.eclipse.org/releases/mars

[12] Eclipse Quick Search plugin
http://spring.io/blog/2013/07/11/eclipse-quick-search

[13] JavaFX 2
http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

[14] UUID
https://en.wikipedia.org/wiki/Universally_unique_identifier