



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

	Oldal
<b>I. Bevezetés</b>	<b>3</b>
1. A témaválasztás indoklása	3
2. Feladat leírás	3
<b>II. Felhasználói dokumentáció</b>	<b>4</b>
3. Telepítés	4
3.1. Java SE Runtime Environment 8 . . . . .	4
3.2. MySQL Community Server 5.6 . . . . .	4
3.3. Az adatbázis használatba vétele . . . . .	5
4. A póker játékról	5
4.1. Játékmenet . . . . .	6
4.2. Játékstílusok . . . . .	6
4.2.1. Classic . . . . .	7
4.2.2. Texas Hold’Em . . . . .	7
5. Futtatás	7
5.1. A póker szerver elindítása . . . . .	7
5.2. A póker kliens elindítása . . . . .	8
6. A póker játék használata	8
<b>III. Fejlesztői dokumentáció</b>	<b>11</b>
7. Felhasznált technológiák	11
7.1. Eclipse . . . . .	11
7.2. Maven . . . . .	11
7.2.1. Adatbázis „húzás” . . . . .	11
7.2.2. Javadoc . . . . .	12
7.3. MySQL . . . . .	12
7.4. Git . . . . .	12
7.5. Enterprise Architect . . . . .	12
8. Adatbázis séma	13

<b>9. Megoldási terv</b>	<b>14</b>
9.1. Probléma . . . . .	14
9.2. Tervezés . . . . .	14
9.3. Implementálás . . . . .	14
9.4. Elemzés . . . . .	14
<b>10. Modulok</b>	<b>14</b>
10.1. Model . . . . .	16
10.2. Shared . . . . .	16
10.2.1. Kivételek . . . . .	16
10.2.2. Kommunikációs rendszer . . . . .	17
10.2.3. Observer pattern [8] . . . . .	17
10.2.4. Session . . . . .	17
10.3. Persist . . . . .	18
10.3.1. Generikus DAO . . . . .	18
10.3.2. Adatbázis menedzser . . . . .	18
10.3.3. Kivételek átfordítása . . . . .	18
10.4. Kliens . . . . .	18
10.4.1. Model-View-Controller [9] . . . . .	19
10.5. Szerver . . . . .	19
10.6. javapokertexasholdem . . . . .	20
<b>11. Funkciók</b>	<b>20</b>
<b>12. Tovább fejlesztési lehetőségek</b>	<b>20</b>
<b>13. Tesztelés</b>	<b>21</b>
13.1. Funkcionális tesztelés . . . . .	21
<b>IV. Irodalomjegyzék</b>	<b>22</b>
<b>14. Hivatkozások</b>	<b>22</b>

## 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

## 3. Telepítés

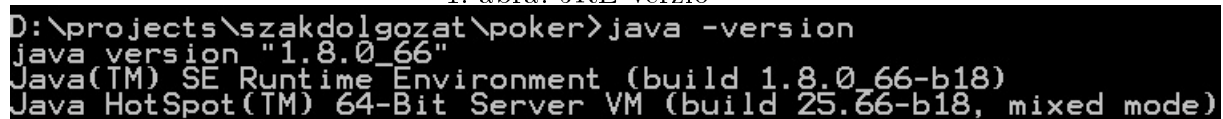
### 3.1. Java SE Runtime Environment 8

A programcsomag futtatásához legalább Windows XP operációs rendszer szükséges, amelyen Java SE Runtime Environment 8 futtató környezet [3] (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: C:\ProgramFiles\Java\jdk1.8.0\_60\bin. Ha nem, akkor pontosvesszővel (;) elválasztva egészítsük ki a változó értékét, majd indítsuk el a promptot (ha nyitva van, akkor indítsuk újra). Ha a

```
java -version
```

utasítás hatására az 1. ábrán látható szöveg jelenik meg a konzolon, akkor sikeres

1. 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)
```

volt a JRE telepítése és beállítása. További instrukciókért ld. melléklet.

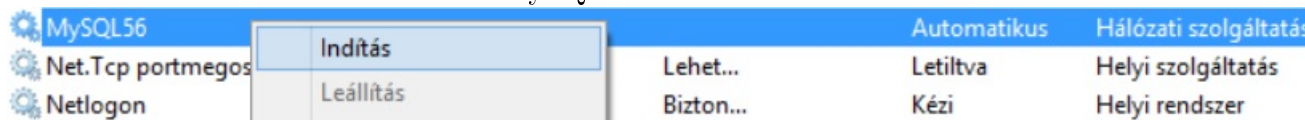
### 3.2. MySQL Community Server 5.6

A programcsomag megköveteli a MySQL Community Server 5.6 adatbázis-kezelő rendszer [4] (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 nyissuk meg a promptot (ha nyitva van, akkor indítsuk újra), majd navigáljunk a MySQL Server bin könyvtárba, ott pedig adjuk ki a

```
mysqld --install
```

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

2. ábra. MySQL Service



`services.msc`

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 felte-

3. ábra. MySQL Service

```
mysql> SELECT VERSION();
+-----+
| VERSION() |
+-----+
| 5.6.27 log |
+-----+
1 row in set (0.00 sec)
```

lepítettük az adatbázis-kezelő rendszert.

### 3.3. Az adatbázis használatba vétele

Ha sikeresen elindítottuk a MySQL Servert, akkor szükségünk lesz egy új adatbázis sémára (és demo adatokra), amelyet a `X:\poker\release\poker-db.sql` állományban találunk. Ezt a fílet 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 „felhúzása” megtörtént.

## 4. 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 játéktílusig. A játékot 52 lapos francia kártyapaklival játszik, amelyben 4 szín és 13 különböző értékű kártyalap található. A játékcélja, hogy minél több zsetont gyűjtsünk össze a partik során.

## 4.1. Játékmenet

Minden játékszervert 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 játéktílus esetén van egy BLIND kör, amikor a szerver bekéri a vakokat a játékosoktól. Az osztótól egyvel balra ülő játékos köteles betenni a kis vakot, a kis vaktól egyvel 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 (CALL), emelhetik (RAISE) 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 (FOLD) és a játék elhagyására (QUIT) is lehetőség van. A felhasználók (korlátozottan) visszanezézhetik a korábbi leosztásokat, és a partiban történt eseményeket (LOG). A játékosok asztalonkénti maximum száma 5 fő. A kliensek a játék elhagyását követően újra-csatlakozhatnak az adott játékszerverre a fentieket figyelembe véve. Lehetőség van asztalt váltani, és a játékszerverek (korlátozottan) képesek kezelni, ha a játékkal megszakad a kapcsolat. A kliens alkalmazások is (korlátozott mértékben) fel vannak készítve az esetleg kommunikációs hibákra.

## 4.2. Játéktílusok

A program kettő beépített játéktípust definiál

- Classic [1]

- Texas Hold'Em [2]

#### 4.2.1. Classic

A klasszikus játéktípus 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 pre-round van, amikor a szervert minden játékosnak öt-öt lapot osztott 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 szervert 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 (vagy majd megcsinálom, hogy automatizált quit legyen ez is.... csak most még bugos.... winnercardsnál úgy is van nullpointerexceptionöm....). Ha ebben a körben is mindenki nyilatkozott, akkor a játékasztal új partit indít.

#### 4.2.2. Texas Hold'Em

A klasszikus játéktípussal 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.

## 5. Futtatás

### 5.1. 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 4. ábrának megfelelő konzol loglistát látunk, akkor a szerveret sikeresen elindítottuk.



4. ábra. Szerver

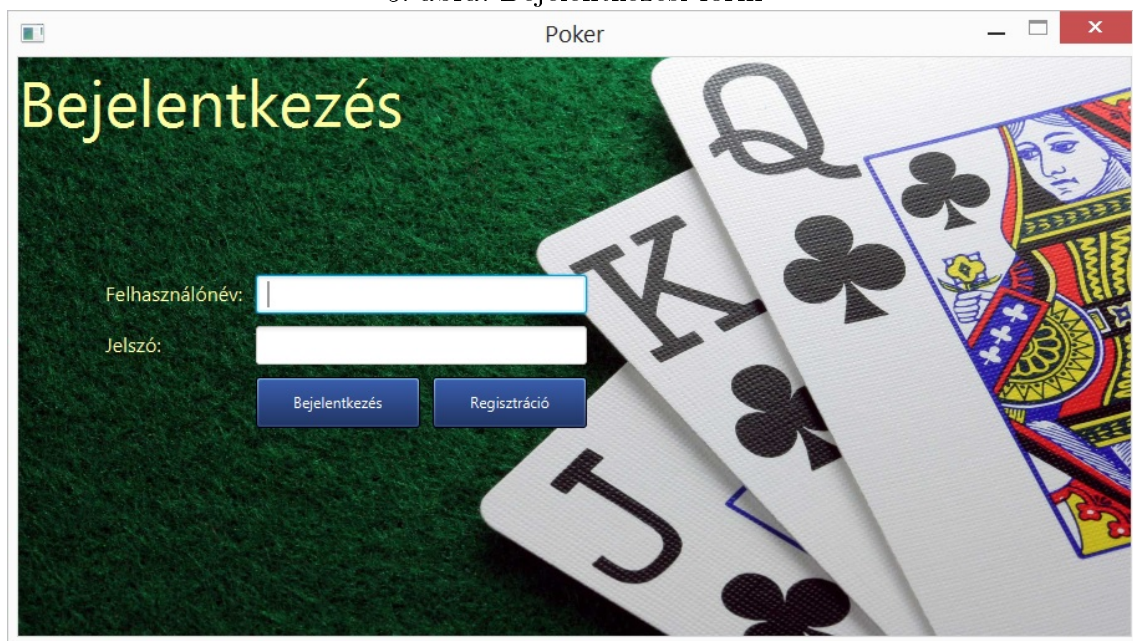
```
D:\projects\szakdolgozat\poker\release>java -jar poker-server-1.0.0.jar
***POKER SZERVER***
Port:1099
Szerver név: pokersv
A szerver elindult
```

## 5.2. 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 megfelelő parancsot. Jöhet az ábra, meg a kódot kicsit átírni, hogy logoljon konzolra, mint a szerver...

## 6. A póker játék használata

5. ábra. Bejelentkezési form



A játék indítása után az 5. ábrán látható bejelentkező felületet kapjuk. Ha még nem regisztráltuk magunkat a játékba, akkor kattinstunk a Regisztráció nevű gombra, amely felület a X. ábrán szerepel. Adjuk meg a regisztrálni kívánt felhasználó nevünket és jelszavunkat, majd regisztráljunk. (Képek kellene 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 formához, amelyet értelemszerűen kitöltve be tudunk jelentkezni a programba. A sikeres

### 6. ábra. Játékasztalok

**Poker**

Játékszerverek

Játékszerver neve	Játék stílusa	Maximum gond...	Játékosok maxi...	Nagy vak
szerver1	HOLDEM	5	5	100.0000000000...
#PL_125Q	CLASSIC	39	5	200.0000000000...
Holdem fun	HOLDEM	15	5	100.0000000000...
Classic fun	CLASSIC	5	5	20.0000000000...
?^xW!	CLASSIC	23	3	1.000000000000...
űűűŔŔűűű÷Ŕ~=	HOLDEM	33	2	10.0000000000...
próba szerver 1	HOLDEM	24	5	12.0000000000...

Csatlakozás  
Profil megtekintése  
Kijelentkezés

bejelentkezést követően a 6. ábrán látható felület fogad minket, ahol például asztalhoz csatlakozhatunk. Tetszőlegesen válasszunk ki egy asztalt, majd kattintsunk a Csatlakozás feliratú gombra. A program átirányított minket a 7. ábrán jelölt felü-

7. ábra. Üres játékasztal



letré. 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 játéktípusú, akkor 8. ábrán látható

8. ábra. Klasszikus játéktípusú asztal



felület fogad minket.

### III. rész

## Fejlesztői dokumentáció

### 7. Felhasznált technológiák

#### 7.1. Eclipse

A szakdolgozatomban eclipse Mars [10] fejlesztőkörnyezetben írtam, amelyhez a quick search [11] 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.).

#### 7.2. 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.

##### 7.2.1. Adatbázis „húzás”

Az adatbázist a fejlesztés elején kézzel húztam újra, majd rátaláltam a sql-maven-plugin nevezetű pluginra. A maven oldalán 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 eliminálni, vagyis letöltöttem az exec-maven-plugin nevű plugint, amelynek segítségével sikerült a

```
mvn exec:exec
```

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

```
mvn clean install -Pbootstrap
```

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

### 7.2.2. Javadoc

A mavennek 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.

## 7.3. MySQL

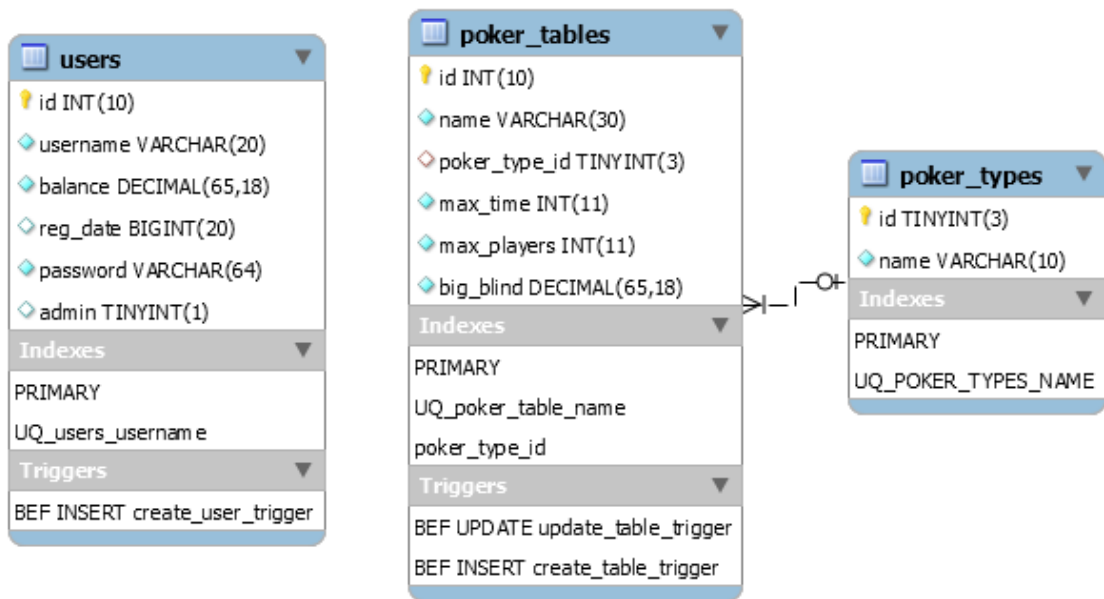
## 7.4. Git

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.

## 7.5. 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 hierchiák és UML diagramok ennek a szoftvernek a segítségével készültek el.

9. ábra. Adatbázis séma



## 8. Adatbázis séma

Az adatbázis (ld. X. ábra) 3 táblából épül fel

- users
- poker\_tables
- poker\_types

Minden tábla rendelkezik elsődleges kulccsal, amelynek típusa UNSIGNED INTEGER, kivétel a poker\_types táblát, ugyanis ennek a táblának az elsődleges kulcsa UNSIGNED TINYINT típusú, amelyet a MySQL Server 4 byteon tárol. Az adatbázis mértékének a szűkítése ebben az esetben indokolt, ugyanis a 255 különböző értékű UNSIGNED TINYINT típus kielégíti a játéktípusok által támasztott követelményeket. A játéktípusok neveit is el kell tárolni, amelynek a maximális hossza 10 karakterben lett meghatározva, amely triggerrel van védve. A users tábla tartalmazza a regisztrált felhasználókat. A regisztrált felhasználók felhasználónévvel és jelszó párossal tudnak regisztrálni, és ennek megfelelően ezek az adatok tárolásra is kerülnek. A felhasználónév maximális hossza 20 karakter, amelyet triggerrel ellenőrzök. A jelszót bcrypt függvénnyel nyírom. Só eltárolása nem szükséges a bcrypt implementációjából adódóan. A felhasználóról el kell még tárolni a regisztráció dátumát, amely a szerver ideje alapján számolódik és UNIX timestampként kerül letárolásra, továbbá a jogosultsági (admin) szintet, amely ugyancsak 4 byteon (TINYINT) kerül ábrázolásra. A 0/1 értékek megfeleltethetők a TRUE/FALSE logikai típusú konstans értékeknek, így tehát, ha az érték 0, vagyis FALSE, akkor az adott felhasználó nem rendelkezik admin jogkörrel, különben igen. Ugyancsak tárolandó érték a

felhasználó játékbeli egyenlege, amely BIGDECIMAL típusként van ábrázolva. A játéktáblákat a `poker_tables` adatbázis tábla tárolja. Szükségünk van eltárolni a játéktábla nevét, melynek felső korlátja 30 karakter és egyedinek kell lennie. Ezeket a megszorításokat szintén triggerekkel ellenőrzöm. A játék asztal játéktípusát is eltárolom, amely egyben idegenkulcs is a `poker_types` táblára nézve. Továbbá minden asztal tulajdonsága, hogy maximum hányan játszhatnak rajta, és hány másodpercig gondolkodhatnak a játékosok. Ezen két érték típusaként ugyancsak UNSIGNED TINYINT van meghatározva, ugyanis az egyes asztaloknál a játékosok száma legfeljebb 5 lehet, míg az egyes játékosok gondolkodási ideje maximum 40, de legalább 5 másodperc.

## 9. Megoldási terv

### 9.1. Probléma

### 9.2. Tervezés

### 9.3. Implementálás

### 9.4. Elemzés

## 10. 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.

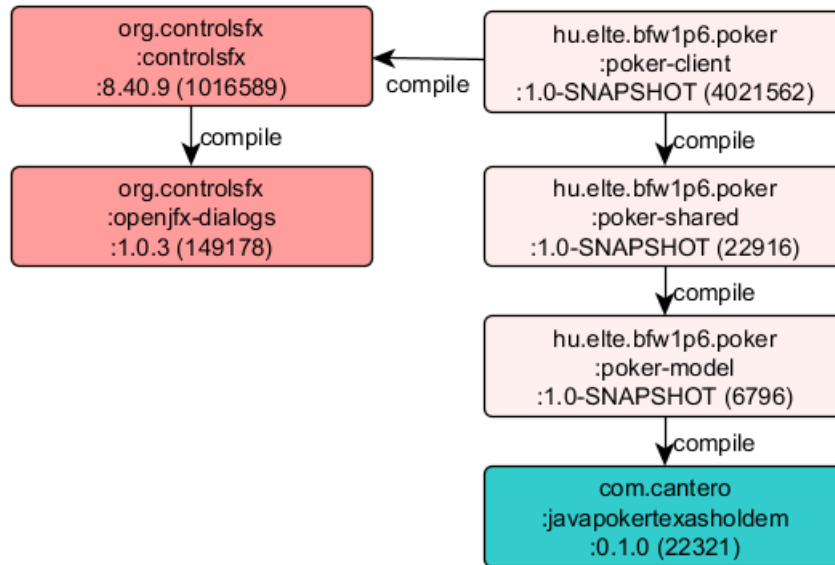
### **poker-model**

A póker játék modellezéséért felelős csomag.

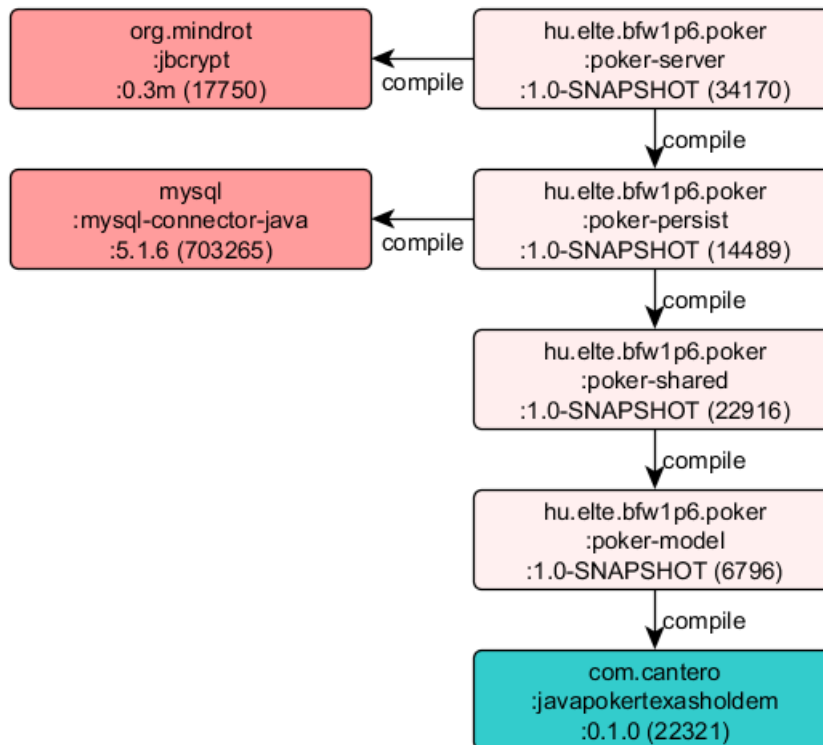
### **javapokertexasholdem**

Külső könyvtár, amely a nyertes játékos kiértékelési feladatot végzi.

10. ábra. Kliens modulra bontása



11. ábra. Szerver modulra bontása



A modulok közötti függőséget a 10. és a 11. ábra szemlélteti.

A programcsomag két főmodulra bontható

- poker-server
- poker-client



## 10.1. Model

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 feleltethetők meg a poker\_tables és a users adattáblák egyes rekordjainak. A PokerType enumeráció a játéktílusokat határozza meg. Ha egy új játéktípust 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 játéktípus nevére vonatkozólag. Természetesen ez a pár sor újonnan hozzáadott kód nem azt jelenti, hogy már használhatjuk is az új játéktípust. 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

## 10.2. Shared

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.

### 10.2.1. Kivételek

A játékcsoagnak 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 felléphetnek bejelentkezéskor, asztalhoz való csatlakozáskor, hibás jelszó megadás esetén, stb (ld. melléklet, javadoc...).

### 10.2.2. 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 játéksomagban:

- 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. 4.2). 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.

### 10.2.3. Observer pattern [8]

Az observer 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 üzenetet a szerver minden kliensnek szétszórja (broadcast, üzenetszórás... valahogy jól kéne megfogalmazni).

### 10.2.4. 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.

### 10.3. Persist

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

#### 10.3.1. Generikus DAO

A generikusság elengedhetett a kódismétlés elkerülése végett. Általánosan kell megfogalmazni az entitások viselkedését 10.1 (vagy fordítva...), és ezáltal a persist modul letisztult arculatot kap. A GenericDAO osztály tartalmaz minden olyan elemet, amelyekre a specializálódott DAO-knak szükségük lehet.

#### 10.3.2. Adatbázis menedzser

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

#### 10.3.3. 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.

### 10.4. Kliens

A kliens modult és annak minden függőségét egy jar fileba csomagolva kell szétterjeszteni a felhasználók között, akik majd ténylegesen használni fogják a programot. A 10. ábra alapján a jar file tartalmazni fog minden olyan osztályt és interfészt, amelyre ténylegesen szüksége lesz a kliens programnak. Az összecsomagoláshoz igénybe vehetjük a maven-t és annak az assembly pluginját. A poker-client könyvtárban a

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

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

#### 10.4.1. Model-View-Controller [9]

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

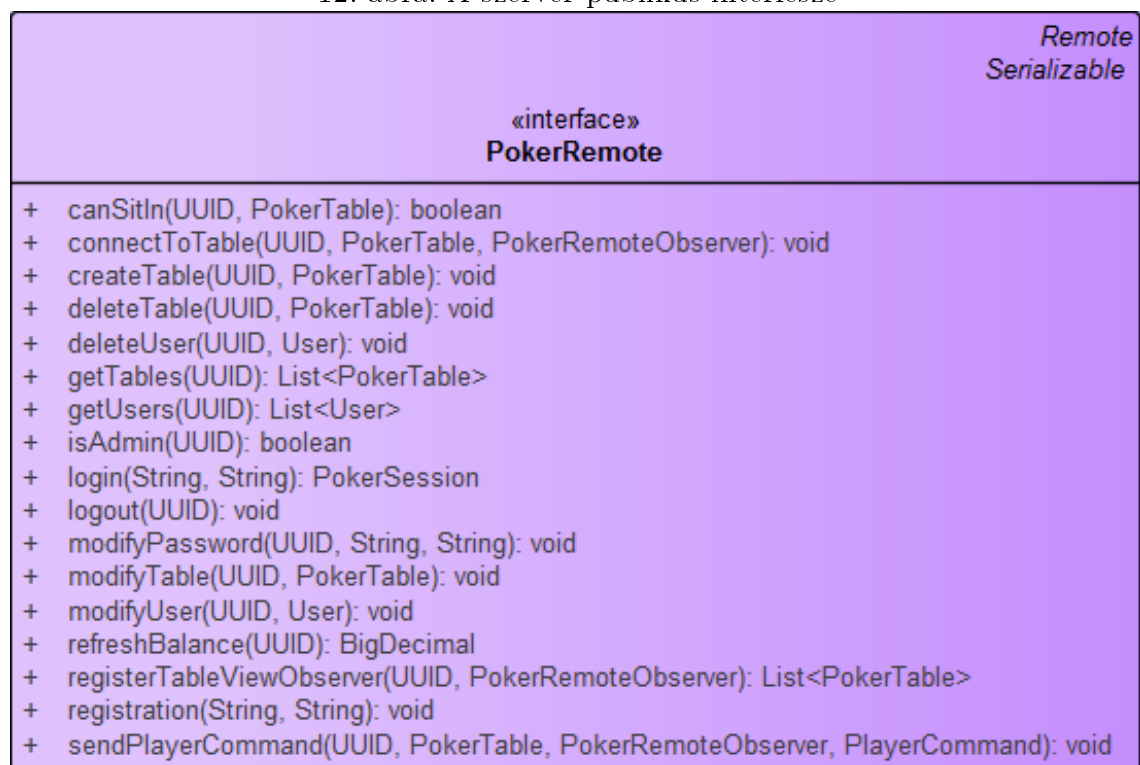
**Model**

**View**

**Controller**

#### 10.5. Szerver

12. ábra. A szerver publikus interfésze



A szerver feladata a játék biztosítása a kliensek számára. A 12. ábrán látható a szerver jóldefiniált publikus interfésze. A kliensek ezt az interfészt tudják elkérni a registryből [7]

A szerver a jelszavak titkosítására bcrypt eljárást alkalmaz, amelynek a biztonságát szózással növeli. Továbbá a szerver felhasznál még egy külső csomagot - mysql-connector-java -, amely az adatbázis kapcsolatért felel. A poker-shared modul felel

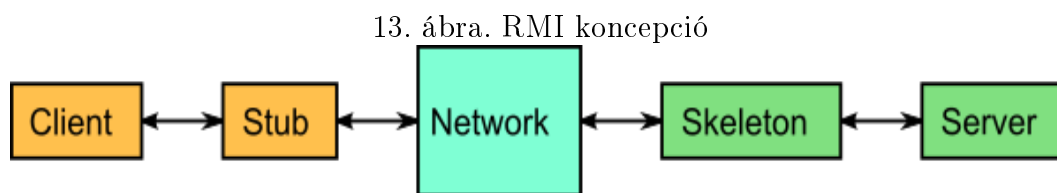
a szerver és a kliens jól definiált kommunikációjáért. A shared modul többek között tartalmazza a közös interfészeket, kivételeket és a póker utasítások megvalósítását. A kliens közvetlenül függ ettől a modultól, azonban a szerver és a shared modul közé beékelődött a poker-persist modul, amely az adatok adatbázisba való írásáért és abból olvasásáért felel.

## 10.6. javapokertexasholdem

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

## 11. Funkciók

Ahogy a témabejelentőben is szerepel..... RMI kép wikiről, majd azt megmagyarázni a shared modullal, kliens hívja, jól definiált interfész etc.... A szerver vázáért a



PokerRemote interface 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 registryből, mint kliens-oldali szervert, amelyeken a műveletek meg tudja hívni. Az összes megvalósított funkciót le kell írni? Felsorolás szintjén, vagy hogy? Rövid magyarázattal? És amelyek egyértelmű? Pl. felhasználó módosítása... login...

## 12. Tovább fejlesztési lehetőségek

- Az adatbázis viszonylag alacsony absztrakciós szinten került implementálására, 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, responzív 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.

## 13. Tesztelés

### 13.1. Funkcionális tesztelés

## IV. rész

# Irodalomjegyzék

## 14. Hivatkozások

- [1] Ötlapos póker  
[https://hu.wikipedia.org/wiki/Ötlapos\\_póker](https://hu.wikipedia.org/wiki/Ötlapos_póker)
- [2] Texas Hold'Em  
[https://hu.wikipedia.org/wiki/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] Java Poker Texas Holdem Hand Evaluator  
<https://github.com/phstc/javapokertexasholdem>
- [6] bcrypt  
<https://en.wikipedia.org/wiki/Bcrypt>
- [7] Java RMI  
[https://hu.wikipedia.org/wiki/Java\\_remote\\_method\\_invocation](https://hu.wikipedia.org/wiki/Java_remote_method_invocation)
- [8] Observer pattern  
[https://en.wikipedia.org/wiki/Observer\\_pattern](https://en.wikipedia.org/wiki/Observer_pattern)
- [9] Model-View-Controller  
<https://en.wikipedia.org/wiki/Model-View-controller>
- [10] Eclipse Mars IDE  
<https://projects.eclipse.org/releases/mars>
- [11] Eclipse Quick Search plugin  
<http://spring.io/blog/2013/07/11/eclipse-quick-search>