Komponens alapú kétszemélyes játék

# Projekt haladási napló

## 2014-03-06: Megbeszélés

* Feladat általános átbeszélése
* A következő héten ötleteket fogunk gyűjteni a lehetséges megvalósítható játékokról
* Mindenki elgondolkodik rajta, hogyan bontaná komponensekre a programot
* Lehetőségek a megvalósítás platformjára

## 2014-03-13: Megbeszélés

* Beszélgetés a lehetséges játékokról
* Valószínűleg webes szoftvert készítünk, a GUI modulokat ennek megfelelően lehet meghatározni
* Népszerűségi szavazás a játékokról: sakk, malom, ostábla, amőba, dáma

## 2014-03-20: Megbeszélés

* A játék szavazásnak nem lett egyértelmű eredménye. A sakk és a dáma kevesebb pontot kaptak, de ezt leszámítva mindenki mindenre szavazott
* Döntés másfajta felosztás mellett: többfajta játék megvalósítása úgy, hogy az MI komponenst általánosítjuk ki
* A komponenstípusok:
  + Keretrendszer (felelős a felhasználó kezelésért, valamint a GUI-ért)
  + Játék (egy konkrét játék megvalósítása)
  + Mesterséges intelligencia (tetszőleges játékhoz, általános kétszemélyes játék)
  + Statisztika (információk gyűjt és szolgáltat a már lejátszott játékokról)
* Megegyezés, hogy ki milyen típusú komponenst készít

## 2014-03-25: Tervezés

* A dokumentáció első verziójának elkészítése
* A terv körbeküldése emailben, hogy mindenkinek megfelel e
* Az elvállalt komponensek listájának véglegesítése

## 2014-03-26: Tervezés

* Hibajavítások a dokumentációban

## 2014-03-27: Tervezés

* Dokumentáció véglegesítése

## 2014-04-04 TerveZés

* Interfészek véglegesítése

## 2014-05-24 Statusz report

* Ki hogyan halad, appróbb módosítások a tervben
* AI interfész hiba javítása

## 2014-05-08 Be mutató előtti megbeszélés

* Elkészült / haladó komponensek megtekintése, átnézése, ötletelés.

# Rendszer környezeti térképe

## Használati modell



A projekt célja egy olyan program készítése, amiben felhasználók regisztrálhatnak, majd bejelentkezhetnek, és kétszemélyes játékokat játszhatnak egymás, illetve mesterséges intelligenciák ellen. Kétszemélyes játék alatt olyan klasszikus társasjátékot értünk, amiben:

* Két ember játszik egymás ellen
* A játékosok felváltva, egymás után választanak véges sok lehetség közül
* A játékban nem kap szerepet a szerencse, minden kizárólag a játékosok döntésén múlik
* A játék garantáltan véget ér véges számú kör elteltével

A felhasználó szemszögéből a program használata a bejelentkezéssel kezdődik. Amennyiben nem rendelkezik még felhasználóval a rendszerben, ehhez regisztrálnia is kell. Bejelentkezés után lehetősége van meglévő játékhoz csatlakozni / újat indítani, valamint kijelentkezni. A játékokat bármikor félbeszakíthatja, vagy amennyiben annak sorsa eldől, ez automatikusan is megtörténik – ez után a játékos ugyanazokat a lehetőségeket kapja, mint bejelentkezés után.

Ezeknek az átmeneteknek a pontos lehetőségeit az alábbi állapot átmeneti diagram szemlélteti[[1]](#footnote-1):



### Regisztráció

A regisztráció során a felhasználónak meg kell adnia adatait, amik segítségével a jövőben azonosítani tudja magát. Sikeres regisztráció után a bejelentkezés következik.

#### Előfeltétel:

* ne legyen bejelentkezett felhasználó.

#### elvárt tevékenység:

* A felhasználó adjon meg egy felhasználónevet és egy jelszót
* A felhasználónév még ne legyen foglalt

#### utófeltétel

* A felhasználó létrejön az adatbázisban

### Bejelentkezés

A bejelentkezésnél a felhasználó a regisztrációnál megadott azonosító adatok segítségével azonosítja magát. Amennyiben helyes adatokat adott meg, tovább léphet a játék főmenüjébe. Sikeres bejelentkezés után a főmenübe jutunk.

#### előfeltétel

* ne legyen bejelentkezett felhasználó.

#### Elvárt tevékenység

* A felhasználó adjon meg egy felhasználónevet és egy jelszót
* A felhasználónév és a jelszó egyezzen meg egy már regisztrált felhasználó adataival

### utófeltétel

* A felhasználó bejelentkeztetésre kerül, és a főmenübe kerül.

### Főmenü

A menüben a felhasználónak lehetősége van:

* Új játékot indítani a lehetséges játéktípusok közül
* Meglévő, még csak egy játékost tartalmazó játékhoz csatlakozni
* Megtekinteni a várakozó állapotú játékok listáját
* Kijelentkezni

#### előfeltétel

* legyen bejelentkezett felhasználó.

#### elvárt tevékenység:

* a felhasználó válasszon a lehetséges akciók közül

#### utófeltétel

nincs

### Kijelentkezés

A kijelentkezéssel a felhasználó alapállapotba hozza a programot, lehetőséget adva ezzel másoknak is a belépésre.

#### előfeltétel

* legyen bejelentkezett felhasználó.

#### Utófeltétel:

* nincs

### Új játék indítása

Új játék indításakor a felhasználó választhat a lehetséges játéktípusok közül, valamint eldöntheti, hogy mesterséges intelligencia, vagy másik élő játékos ellen szeretne játszani.

Amennyiben mesterséges intelligenciát választ, meghatározhatja annak típusát, valamint a pontszámításhoz használt heurisztikát is a lehetőségek listájából.

Amennyiben másik élő játékost szeretne, a játék várakozó állapotba kerül, amíg valaki nem csatlakozik.

#### előfeltétel

* legyen bejelentkezett felhasználó.

#### elvárt tevékenység:

* a játékos adja meg az induláshoz szükséges adatokat
  + másik játékos elleni játékhoz a játék típusát
  + MI elleni játékhoz ezen túl az MI és a pontszámító algoritmus típusát

### utófeltétel

* A játékot reprezentáló objektum jöjjön létre az adatbázisban a tábla kezdeti állásával
* Felhasználó elleni játék esetén a játék kerüljön várakozó állapotba
* MI elleni játék esetén az első (emberi) játékos lépése következik

### Csatlakozás játékhoz

Amennyiben a játékos már meglévő játékhoz szeretne csatlakozni, megtekintheti azoknak a játékoknak a listáját, amiben csak egy játékos tartózkodik, és csatlakozhat ezek egyikéhez.

#### előfeltétel

* legyen bejelentkezett felhasználó
* a kiválasztott játék létezzen
* a kiválasztott játék legyen várakozó állapotban

#### Elvárt tevékenység

* egy játék kiválasztása a listából

#### Utófeltétel:

* a kiválasztott játékhoz hozzárendelődik a játékos
* a játék várakozóból átkerül „az első játékos jön” állapotba
* az első játékos értesítése a változásról

### Lépés

Miután mindkét játékos bekerült egy játékba, az egymást követő lépések sorozatából áll. A játék addig tart, ameddig annak szabályai szerint nem dőlt el egyértelműen a kimenetele, vagy az egyik játékos nem lépett idő előtt ki. A játékok lehetséges kimenetelei: az egyik játékos egyértelmű győzelme, vagy döntetlen. Amennyiben az egyik játékos idő előtt kilép, az feladásnak, és így a másik játékos győzelmének minősül.

#### előfeltétel

* a játék létezik
* a játék folyamatban van
* a megfelelő játékos jön

#### Elvárt tevékenység

A játéktábla egy kattintható elemére történő kattintás

#### Utófeltétel

* a játék állásának frissítése
* a következő játékos beállítása
* amennyiben emberi játékosról van szó, a másik játékos értesítése
* amennyiben MI ellen játszunk, az MI lépés lefolytatása, majd a játékos ismét értesítése

### Eredmények megtekintése

A játék vége után lehetőségünk van arról egy rövid összefoglalót megtekinteni, amiből megtudhatjuk például, hogy mi lett annak végeredménye.

#### Előfeltétel

* A kiválasztott játék létezzen
* A kiválasztott játék legyen véget ért állapotban

#### Elvárt tevékenység

A játékállás, és végeredmény megtekintése

#### Utófeltétel

* a játéktábla és eredmény megjelenik

## Strukturális modell

A szoftver implementációja négy jól elkülönülő komponensre bontható, amik egymáshoz néhány egyszerű interfész segítségével kapcsolódnak. Ezek a komponensek:

* A keretrendszer, ami a felhasználó kezelésért, a játékindításért, valamint a program során felmerülő egyéb háttérműveletekért felelős. A keretrendszer működéséből a felhasználó közvetlenül nem vesz észre semmit, ez csak a többi komponens számára biztosít segítséget.
* A felhasználói felület, ami a játék valamint a keretrendszer segítségével megjeleníti a felhasználók (játékosok) számára a játékokat, valamint azok listáját, illetve lehetőséget biztosít a regisztrációra – bejelentkezésre.
* A játék, ami egy kétszemélyes játék megvalósítását tartalmazza
* A mesterséges intelligencia, amely képes tetszőleges játékhoz döntéseket hozni

A játék ezen kívül tartalmaz egy alkomponenst, ami szorosan kapcsolódik az adott komponensért, de egy játékhoz több, egymástól független is tartozhat: a pontszámításért felelős heurisztikát, aminek a segítségével a mesterséges intelligenciák a játékgráf elemeinek pontszámait meg tudják határozni.



### Framework

A keretrendszer biztosítja a többi komponens számára szükséges interfészeket, osztályokat. Ezen túl a keretrendszer felelős azért, hogy tudjon a többi komponens létezéséről, valamint biztosítsa a perzisztens objektumok (felhasználók, játékállások) tárolását.

### GUI

A keretrendszer feladata, hogy biztosítsa a felhasználók kezelésével kapcsolatos minden funkcionalitást, azaz a regisztrációt, bejelentkezést, valamint kijelentkezést. Emellett feladata megjeleníteni a menüt, amiben a bejelentkezett felhasználó választhat a cselekvései között, lehetőséget ad játék létrehozásához illetve játékok listázásához, valamint a statisztika modul információinak megtekintéséhez.

### Game

A játék komponens egy kétszemélyes játékot valósít meg, amely megfelel a 2. a pontban ismertetett követelményeknek. A játéknak saját magának kell megvalósítania felhasználói felületét, a keretrendszer által adott interfész segítségével.

Az MI komponens felé ki kell ajánlania egy interfészt, amely segítségével az bejárhatja a lehetséges lépések gráfját, hogy döntést tudjon hozni.

A Két komponens viszonylag szoros kapcsolatban áll egymással, a vezérlő szerepet a játék tölti be, ami az MI komponenst az általa biztosított interfész segítségével kéri meg rá, hogy hozza meg következő döntését.

### AI

A mesterséges intelligenciát megvalósító komponens feladata, hogy egy kapott gráf interfész segítségével hozzon döntést, hogy a gráf első szintjén melyik élt válassza. A komponensnek felhasználói felületet nem kell megvalósítania, hanem egy interfészt kell biztosítania más komponensek számára.

A komponens a gráf mellett felhasználhatja a statisztika komponens által biztosított információkat is döntése meghozásához.

### ScoreCalculator

A pontszámítást végző heurisztika, egy játék aktuális állását jelző objektumból, annak szerkezetének ismeretében, egy pontszámot határoz meg, aminek a segítségével az MI-k értékelhetik a játék haladását.

## Tevékenység és interakciós modell

### Új játék indítása



Ahhoz, hogy játékot indíthasson, a felhasználnak először be kell jelentkezni. Ezt a bejelentkezést előtte regisztráció is megelőzheti, illetve lehetséges, hogy a játékos már egy korábbi játéka előtt bejelentkezett- ez esetben nem kell újra bejelentkeznie.

Miután bejelentkezett, lehetősége van új játékot indítani. Ehhez kiválasztja a játékot, valamint a(z opcionális) mesterséges intelligenciát, majd elkezdődik a játék.

A már létrehozott játékhoz történő csatlakozás hasonlóan működik, azonban az ennél egyszerűbb, mivel ott nincs szükség új játék létrehozására, mindössze le kell kérdezni azt a listából.

### Játék Folyamata



A játék lépések folyamatából áll, addig, amíg lehet lépni, vagy az egyik játékos kilépés mellett nem döntött. Az egyik játékos szemszögéből nézve, amíg a másik lép, addig neki várakoznia kell annak eredményére.



### Mesterséges intelligencia lépése



Amennyiben a mesterséges intelligencia az egyik játékos, úgy annak számításáért az AI komponens a felelős. Ehhez, amikor a játék komponensnek a második játékosnak kell megadnia a lépés lehetőséget, meghívja az AI komponens interfészét, paraméterként átadva saját, a lépések gráfját exportáló interfészét.

# Eredeti Feladatbeosztás

|  |  |
| --- | --- |
| Komponens | Fejlesztő |
| Projektterv | Parragi Zsolt |
| Framework | Szikszai Gergő |
| Petrás András |
| Game | Tóth Tamás |
| Kresz Marcell |
| AI | Angeli Dávid |
| Németh András |
| Tóth Gábor |
| Stat | Dananaj Pál |
| Ancsin Attila |

# Interfész leírások

## GameInterface / AbstractGame

A GameInterface tartalmazza a játékok által megvalósítandó eljárásokat. Ennek egy részleges megvalósítását, ami a legtöbbször ugyanúgy működő általános eljárásokból áll, a framework komponens által megvalósított AbstractGame tartalmazza.

### GetgameTypeName(): String

Visszaadja a játék nevét. Pl. „tictactoe”. A keretrendszer ezzel a névvel tárolja el a típusnévvel tárolja el az adatbázisban a játék aktuális állását.

### GetEntityInfo(): Game

Visszaadja a játékot az adatbázisban tárolt adatait reprezentáló osztályt.

**Az AbstractGame megvalósítja.**

### loadGameId(ID): VOID

Betölti az adott ID-jű játékot

**Az AbstractGame megvalósítja.**

### getCurrentBoard(PLAYER): Board

Visszaadja az aktuális játéktáblát, a paraméterként megkapott játékos szemszögéből reprezentálva.

### clickedOn(Player, clickPosition): VOID

A GUI hívja meg, amikor a játékos a játéktábla egy elemére kattintott. Az eljárásnak a kattintás alapján módosítania kell a játék állását, és elmenteni azt az adatbázisban.

### getScoreCalculators(): Set<String>

Visszaadja a játékhoz tartozó pontszámító algoritmusok listáját.

### doAiStep(): Void

A felhasználói felület hívja meg, amikor egy mesterséges intelligenciának kell lépnie.

**Az AbstractGame megvalósítja.**

### CreateGame(FirstPlayer)

Létrehoz egy új játékot, és várakozó állásba helyezi azt.

**Az AbstractGame megvalósítja.**

## AbstractGame

Az AbstractGame az előzőeken túl megkövetel néhány további eljárást a játékoktól, valamint kötelezi a játékot a GameGraphInterface megvalósítására. Az AbstractGame ezen túl egyszerű lehetőséget biztosít a játékoknak a GameService használatára.

### getScoreCalculator(Name):ScoreCalculator

Visszaadja az adott nevű pontszámító algoritmust.

### createBoard(): String

A kezdeti játéktábla létrehozásáért felelős eljárás.

## GameGraphInterface

A játékfa bejárását lehetővé tévő interfész.

### getCurrentStep(): Object

Visszaadja az aktuális játékállást

### GetPossibleSteps(FromStep): Iterable

Visszaadja az adott lépésből lehetséges további játékállások iterátorát.

### GetScoreForPlayer(Step, Player)

Megmondja, hogy az adott játékállás az adott játékos szempontjából hány pontot ér.

## Game

A játékállás adatbázis reprezentációjáért felelős osztály.

### ID: GEt/set

A játék azonosítója

### Typename: get

A játék típusának neve

### Boardinfo: get/set

Az aktuális játékállás reprezentációja

### Player1: GET

Az első játékos, aki létrehozta a játékot

### Player2: GET/SET

A második játékos.

### LastStepAt: GET/SET

Mikor történt az utolsó lépés

### GameState: GET/SET

A játék aktuális állapota:

* várakozik második játékosra
* folyamatban, az első játékos lép
* folyamatban, a második játékos lép
* az első játékos nyert
* a második játékos nyert
* döntetlen

### CURRENTPLAYER: Get

Az aktuális játékos

### isCurrentPlayerAi: boolean

Igaz, ha az aktuális játékos AI

## Board

A játéktábla egy állapotát reprezentáló osztály, elemek kétdimenziós tömbjét tartalmazza. Az indexek nullától kezdődnek, és növekednek.

### Board(W, H): BOARD

Létrehoz egy adott szélességű és magasságú táblát

### getPieces(): Piece[][]

Visszaadja az elemek mátrixát

### getAt(X, Y): Piece

Visszaadja az adott pozícióban lévő elemet.

### getWidth(): int

Visszaadja a tábla szélességét

### getHeight(): Int

Visszaadja a tábla magasságát

## Piece

A játéktábla egy elemét reprezentáló interfész. Egy egyszerű teljes megvalósítása a SimplePiece, ami további segédfüggvényeket ad a játékok számára.

### isClickable(): Boolean

Igaz, ha az elem kattintható

### getCssClass(): String

Visszaadja az elem CSS osztályát

### GetStyleOverride(): String

Visszaadja az elem inline style tagjának tartalmát

### GetContent(): String

Visszaadja az elemet reprezentáló tábla cella tartalmát

## SimplePiece

A simplepiece a Piece minden metódosához tartalmaz settereket, biztosít egy konstrukturt amiben mindet meg lehet adni, valamint a következő további eljárásokat adja:

### setTextColor(String): Void

Beállítja a szöveg színét

### setBackgroundColor(String): Void

Beállítja a háttér színét

## GameService

A szolgáltatás játékok kezelésével kapcsolatos egyszerű műveleteket tartalmaz.

### findGameById(ID): Game

Megkeresi az adott azonosítójú játékot. Ha nem találja meg, null-t ad vissza.

### FindGameTypeByName(Name): GAmeType

Megkeresi az adott nevű játéktípus információit

### FindWaitingGames(): List<Game>

Visszaadja a várakozó állapotban lévő játékok listáját

### FindGameInterfaceByName(NAme): GameInterface

Visszaadja az adott nevű játéktípus implementációját

### persistGame(Game): Void

Elmenti a játékot

### updateGame(GAME): VOID

Frissíti a már elmentett játékállást.

### listAllAi(): List<Aitype>

Az AI algoritmusok listáját adja vissza

### findAiByName(AIName): AiType

Visszaadja az adott nevű mesterséges intelligencia információit.

### GetScoreCalculatorNames(GameType): List<String>

Visszaadja a pontszámító algoritmusok neveit

## UserService

Felhasználók kezelését segítő szolgáltatás

### createUserAndGroup(username, password, groupname)

Létrehozza az adott paraméterekkel rendelkező felhasználót.

# Végfelhasználói tesztesetek

## Keretrendszer tesztelése

* Felhasználó regisztrációja
* Bejelentkezés
* Főmenü / játéklista megtekintése
* Játék létrehozása emberi játékos ellen
* Másik számítógépen / böngészőben:
  + felhasználó regisztrációja
  + Bejelentkezés
  + Főmenü / játéklista megtekintése
  + Az előzőleg létrejött játékhoz történő csatlakozás

A folyamat feltételezi, hogy a keretrendszer tartalmaz legalább egy játékot. Játék nélkül a keretrendszer működése nem tesztelhető. A keretrendszer teszteléséhez mesterséges intelligenciára nincs szükség.

Elvárt működés:

* az itt leírt lépések végrehajthatóak
* a lépések végrehajtása után az első böngészőben a játékos értesült róla, hogy a játékba még egy játékos csatlakozik, és neki kell lépnie
* a második böngészőben a játékos látja a játéktáblát, valamint annak a tényét, hogy az első játékosra kell várakoznia

Egy felhasználó nem csatlakozhat kétszer ugyanahhoz a játékhoz, azaz nem játszhat maga ellen.

Ennek a tiltását a következő folyamattal tudjuk tesztelni:

* Felhasználó regisztrációja / Bejelentkezés
* Főmenü / játéklista megtekintése
* Játék létrehozása emberi játékos ellen
* Másik számítógépen / böngészőben / másik böngészőablakban:
  + Bejelentkezés ugyanazzal a felhasználóval (ha ugyanaz a böngésző, nem szükséges)
  + Főmenü / játéklista megtekintése
  + Az előzőleg létrejött játékhoz történő csatlakozás nem lehetséges

Amennyiben tudunk csatlakozni a saját magunk által létrehozott játékhoz, az hibának tekintendő, egyébként a teszt sikerült.

## Játék tesztelése

Az MI játék teszteléséhez szükséges, hogy a keretrendszer működjön (az 5.A. teszt végrehajtható legyen, és ne akadjon el hibával), mivel a játék létrehozásának a folyamata ugyanaz, mint annak a tesztnek a folyamata.

* Felhasználó regisztrációja / Bejelentkezés
* Főmenü / játéklista megtekintése
* Játék létrehozása emberi játékos ellen
* Másik számítógépen / böngészőben:
  + felhasználó regisztrációja / Bejelentkezés
  + Főmenü / játéklista megtekintése
  + Az előzőleg létrejött játékhoz történő csatlakozás
* **Mindkét böngészőben: játéklépések végrehajtása**

A játék tesztelése mindössze a játék lépések egymás utáni végrehajtásából áll. A játékot akkor tekintjük működőnek, ha:

* a játék véges körön belül garantáltan véget ér
* a játék megfelel annak a szabályainak, azaz nem enged meg tiltott lépéseket
* a játék nem kínál fel olyan kattintható mezőt, amire kattintva nem történik semmi
* a játék megfelelően hirdeti ki a győztest / döntetlent
* a játékosoknak felváltva, egymás után kell lépnie

## Mi tesztelése

Az MI játék teszteléséhez szükséges, hogy a keretrendszer működjön (az 5.A. teszt végrehajtható legyen, és ne akadjon el hibával), valamint legyen egy játékunk, amiről tudjuk, hogy MI nélkül működik (5.B. teszt végrehajtható, és nem akad el hibával).

A játéktól továbbá elvárjuk, hogy rendelkezzen egy működő pontszámító algoritmussal, amit az 5.B. teszt nem ellenőriz – az, hogy a teszt elakadását az MI vagy a pontszámító algoritmus hibája okozta, végfelhasználói szempontból nem ellenőrizhető.

Az MI játék teszteléséhez szükséges, hogy legyen egy MI, amit tesztelni tudunk.

A teszt nem teszteli a mesterséges intelligencia minőségét és erejét, mindössze arra fókuszál, hogy az MI hajtson végre lépéseket.

* Felhasználó regisztrációja / bejelentkezés
* Főmenü
* **Új játék létrehozása: a kiválasztott MI ellen, egy olyan játékkal, ami rendelkezik működő pontszámító algoritmussal**
* **Lépések sorozata, amíg a játék véget nem ér**

A folyamat első két pontja csak a teszt előfeltétele, az MI-t a második kettő teszteli.

A lépések sorozatánál azt várjuk el, hogy az MI lépjen, azaz a játék ne szálljon el hibaüzenettel, illetve ne kerüljön végtelenciklusba. Praktikusan elvárható, hogy az MI „gyors legyen”, azaz 10 másodpercen belül válaszoljon.

A keretrendszer az MI hibáját úgy kezeli, hogy az MI feladta a játékot, így amennyiben tudjuk, hogy az MI nem tesz direkt ilyen lépést, a feladás hibának tekintendő.

A teszt sikeres, ha a játékot vagy a játékos, vagy az MI megnyerte, és nem kerültünk hibába.

# Telepítési Instrukciók

## Szükséges szoftverek

### MySql

A programnak szüksége van egy adatbáziszerverre, amiben a felhasználókat, játékállásokat, valamint egyéb konfigurációit tárolja. Erre a célra megfelel tetszőleges SQL szerver, a példa telepítés során MySQL-t használunk. Ehhez le kell töltenünk a MySQL Cummunity Edition-t a honlapjáról, és fel kell telepítenünk.

A MySQL windowsos telepítője: <http://dev.mysql.com/downloads/file.php?id=451576>

### Glassfish

1. A program egy Enterprise Java alkalmazás, futtatásához tetszőleges Java EE Server megfelelő. A telepítési instrukciók az ingyenes és opensource GlassFish Server-t veszik alapul, de egyszerűen adaptálhatóak más szerverekhez is.
2. Amennyiben az alkalmazást fejlesztői módban, Netbeans vagy Eclipse segítségével szeretnénk futtatni, nem kell külön alkalmazásszervert letöltenünk és telepítenünk, mivel azok ezt megteszik helyettünk.
3. A Glassfish telepítőjének letöltése: <https://glassfish.java.net/download.html>

### Glassfish – MySQL connector

Az alkalmazás szervernek szüksége van egy adapterre, aminek segítségével kapcsolódni tud az SQL szerverhez. MySQL esetén ez a MySQL JDBC Adapter, amit a következő oldalról tölthetünk le:

<http://dev.mysql.com/downloads/connector/j/5.0.html>

A zipből a mysql-connector-java-5.1.30\mysql-connector-java-5.VER.ZIO-bin.jar fájlt a Glassfish glassfish\lib könyvtárába kell másolnunk.

## Szoftverek konfigurálása

### MySql: felhasználó és adatbázis

A szoftver futtatásához szükség van egy adatbázisra, valamint egy felhasználónévre és jelszónak, akinek van joga annak az adatbázisnak a használatára. A felhasználónak célszerű az adatbázison belül minden privilégiumot megadni, mivel a program a táblákat saját magának manageli.

Egy példa felhasználó létrehozására:

**CREATE** USER 'kompgame'@'localhost' **IDENTIFIED** **BY** '\*\*\*';  
  
**GRANT** **USAGE** **ON** \* . \* **TO** 'kompgame'@'localhost' **IDENTIFIED** **BY** '\*\*\*' **WITH** **MAX\_QUERIES\_PER\_HOUR** 0 **MAX\_CONNECTIONS\_PER\_HOUR** 0 **MAX\_UPDATES\_PER\_HOUR** 0**MAX\_USER\_CONNECTIONS** 0 ;  
  
[**CREATE** **DATABASE**](http://localhost/phpmyadmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.5%2Fen%2Fcreate-database.html&token=52673e5bcbdb0b84415e974bb5f6763c) [**IF**](http://localhost/phpmyadmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.5%2Fen%2Fcontrol-flow-functions.html%23function_if&token=52673e5bcbdb0b84415e974bb5f6763c) [**NOT**](http://localhost/phpmyadmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.5%2Fen%2Flogical-operators.html%23operator_not&token=52673e5bcbdb0b84415e974bb5f6763c) **EXISTS** `kompgame` ;

### Glassfish: Jdbc Connection pool

Az alkalmazásszerverünkben létre kell hoznunk egy JDBC Conenction Poolt, aminek a segítségével csatlakozunk az adatbázishoz. Ezt a Glassfish admnisztrációs felületén tudjuk megtenni, ami alapértelmezésben a <http://localhost:4848/> címen található.

A JDBC Connection Pool hozzáadásához menjünk a Resources->JDBC->JDBC Connection pools fülre, majd válasszuk a „New” menüpontot.

Az első oldalon adjuk meg a pool nevét (jdbc-kompdb), típusát (DataSource), valamint az adatbázis kapcsolat driverét (MySQL)

A következő oldalon az Additional Properties alatt töröljünk ki mndent a User, ServerName, Port, DatabaseName, Password mezőkön kívül, azokat pedig töltsük ki adatbáziskapcsolatunk adataival.

A kapcsolat elmenteése után a „Ping” gombbal lehetőségünk van kipróbálni, hogy a beállításai helyesek e.

### Glassfish: Jdbc Resource

Miután létrehoztunk egy connection poolt, azt elérhetővé kell tennünk az alkalmazások számára, ehhez fel kell vennünk egy resource-ként.

Ezt a Resource->JDBC->JDBC Resources alatt tudjuk megtenni, a „New” gomb segítségével. Connection poolnak válasszuk ki az előbb létrehozott jdbc-kompdb-t, névnek pedig adjuk meg, hogy jdbc-r-kompdb. Ez a név fontos, mivel ilyen néven fogja keresni az alkalmazás.

### Glassfish: Security Realm

Az alkalmazás authentikálásához fel kell vennünk a komp-realm nevű security realmet. Ehhez menjünk a configurations -> server config -> security -> realms fülre, és válasszuk a „New” opciót. Típusnak a legördülő listából válasszuk a JDBCRealm-et, majd a következő opciókat adjuk meg:

A következő értékeket adjuk meg:

|  |  |
| --- | --- |
| Név | Érték |
| Realm name | komp-realm |
| JAAS context | jdbcRealm |
| JNDI | jdbc-r-kompdb |
| User Table | USERS |
| User Name Column | USERNAME |
| Password Column | PASSWORD |
| Group Table | GROUPS |
| Group Name Column | NAME |
| Password Encryption Algorithm | SHA-256 |
| Assign Groups | users |

## Az alkalmazás telepítése

Az alkalmazást lehetőségünk van kétféleképp is telepíteni: az EAR binárist deployolni a glassfish webes felületén, vagy egy Java fejlesztőkörnyezet, pl. Netbeans segítségével fejlesztői módban.

### A bináris telepítése

A bináros telepítéséhez menjük a Glassfish admin felületére, majd ott az Applications fül alatt válasszuk a „Depley”-t. A tallózó segítségével váálasszuk ki az alkalmazás binárisát, típusnak jelöljük meg az Enterprise Applicationt, majd nyomjuk meg az ok gombot.

### Netbeans segítségével, fejlesztési módban

A projekt eclipseben történő megnyitásához használjuk az Open Project menüpontot, majd válasszuk ki a projekt forrásának főkönyvtárát. Ezután a modulok között nyissuk meg a kompgame-ear alprojektet, majd kattintsunk rá jobb gombbal, és válasszuk, hogy „Build with dependencies”.

Várjuk meg, amíg a maven végez a függőségek letöltésével, majd a kontextusmenüben a „Run” paranccsal elindíthatjuk az alkalmazást. Miután sikeresen elindult, a netbeans meg is nyit egy böngészőt a program nyitólapjával.

## Komponensek felvétele

Az alkalmazáshoz lehetőség van AI, illetve játék komponensek regisztrálására. Regisztráció előtt egy játék sem fog megjelenni az alkalmazásban. A program EAR binárisa önmagában tartalmaz néhány komponenst, amelyet a lejjebb lévő listában láthatunk, valamint felvehetők hozzá más forrásból telepített komponensek is, akár olyanok is, amik másik alkalmazásszerveren futnak.

### Játék telepítése

Játék hozzáadásához a GAMETYPE táblába kell felvennünk a játéknak egy id-t, egy belső nevet, valamint egy URL-t, aminek a segítségével az alkalmazásszerver megtalálja a komponenst.

Az EAR-ben alapból megtalálható játékok:

|  |  |
| --- | --- |
| Rövid név | URL |
| kamisado | java:app/kompgame-game-kamisado-1.0-SNAPSHOT/hu.elte.komp.kamisado |
| tictactoe | java:app/kompgame-game-tictactoe-1.0-SNAPSHOT/hu.elte.komp.tictactoe |
| tic-tac-toe-test | java:app/kompgame-game-tictactoe-test-1.0-SNAPSHOT/hu.elte.komp.tictactoe-test |

### AI telepítése

MI hozzáadásához az AITYPE táblába kell felvennük a mesterséges intelligenciának egy id-t, egy belső nevet, valamint egy URL-t, aminek a segítségével az alkalmazásszerver megtalálja a komponenst.

Az EAR-ben alapból megtalálható MI-k:

|  |  |
| --- | --- |
| Rövid név | URL |
| random | java:app/komponens-ai-random-1.0-SNAPSHOT/hu.elte.komp.ai.random |
| alfabeta | java:app/komponens-ai-alfabeta-1.0-SNAPSHOT/hu.elte.komp.ai.alfabeta |
| minimax | java:app/komponens-ai-minimax-1.0-SNAPSHOT/hu.elte.komp.ai.minimax |
| negamax | java:app/kompgame-ai-negamax-1.0-SNAPSHOT/hu.elte.komp.ai.negamax |
| avg | java:app/kompgame-ai-avg-1.0-SNAPSHOT/hu.elte.komp.ai.avg |

# 7. A komponensek fejlesztői

|  |  |
| --- | --- |
| Komponens | Fejlesztő |
| Framework | Parragi Zsolt  Petrás András  Szikszai Gergely |
| Web | Szikszai Gergely |
| Web2 | Petrás András |
| Kamisado Game | Tóth Tamás |
| Tictactoe Game | Tóth Gábor |
| Tic-tac-toe-test Game | Németh András |
| Alfabeta AI | Angeli Dávid |
| Minimax AI | Angeli Dávid |
| Negamax AI | Németh András |
| Avg AI | Németh András |
| Kamisado.WinCalculator | Angeli Dávid |

1. Az átláthatóság kedvéért az ábrákról lemaradtak logikus és létező, de a program lényegi működését nem befolyásoló átmenetek, mint például a játék elkezdése után természetesen egyetlen lépés nélkül is ki lehet belőle lépni, vagy az új játék indítása esetből játék tényleges indítása nélkül ismét elérhető a főmenü. [↑](#footnote-ref-1)