

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Csontos Valentin

Android alapú sorozat / film nyilvántartó rendszer

Konzulens

Braun Patrik János

BUDAPEST, 2015

Tartalomjegyzék

[1 Bevezetés 3](#_Toc436245294)

[2 Rendszerterv 4](#_Toc436245295)

[2.1 Architektúra 4](#_Toc436245296)

[2.2 Adatréteg 5](#_Toc436245297)

[2.2.1 Egyed-kapcsolat diagramon szereplő entitások és kapcsolataik 5](#_Toc436245298)

[2.2.2 ER diagramon szereplő entitások és kapcsolataik leképezése Java osztályokká 6](#_Toc436245299)

[2.2.3 Felhasználó jelszavának tárolása adatbázisban 6](#_Toc436245300)

[2.3 Üzleti logikai réteg 7](#_Toc436245301)

[2.3.1 Absztrakt generikus ősosztály 7](#_Toc436245302)

[2.3.2 Üzleti logikát megvalósító Session Bean-ek 8](#_Toc436245303)

[2.3.3 Mobil klienssel való kommunikáció tervezése 10](#_Toc436245304)

[3 Felhasznált technológiák, eszközök 15](#_Toc436245305)

[3.1 JavaEE 15](#_Toc436245306)

[3.2 JAX-RS 15](#_Toc436245307)

[3.3 NetBeans 15](#_Toc436245308)

[3.4 Glassfish 15](#_Toc436245309)

[3.5 Apache Maven 15](#_Toc436245310)

[3.6 Gson 16](#_Toc436245311)

[3.7 MySQL 16](#_Toc436245312)

[3.8 Postman 16](#_Toc436245313)

[3.9 Git 16](#_Toc436245314)

[3.10 Draw.io 16](#_Toc436245315)

[4 Telepítési dokumentáció 17](#_Toc436245316)

[Ábrajegyzék 18](#_Toc436245317)

# Bevezetés

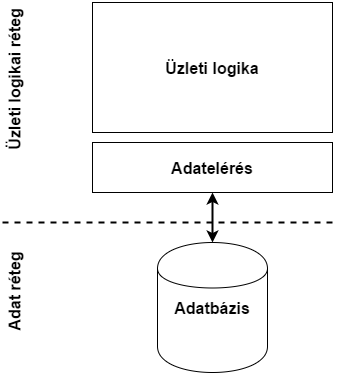
A dokumentáció célja, hogy az „Android alapú sorozat / film nyilvántartó rendszer” back-end oldali szolgáltatásainak tervét részletesen bemutassa. A leírásban található egy rövid ismertetés a felhasznált technológiákról és eszközökről. Továbbá egy telepítési útmutatót is tartalmaz, amelyben részletesen le van írva, hogyan kell telepíteni a szolgáltatásokat.

# Rendszerterv

Ebben a fejezetben a rendszer felépítését mutatom be a követelményspecifikációban leírtak szerint.

## Architektúra

Az alkalmazást a többrétegű architektúra elvének megfelelően építettem fel, melyben 2 réteg különül el egymástól: adat- és üzleti logikai réteg. Az alkalmazás rétegekre bontásának előnye, hogy könnyen karbantartható, fejleszthető és robosztusabb rendszereket hozhatunk létre, mivel az egyes rétegek egymástól függetlenül fejleszthetők, sőt akár cserélhetők is, ugyanis az egyes rétegek módosítása nincs hatással a többi réteg működésére.

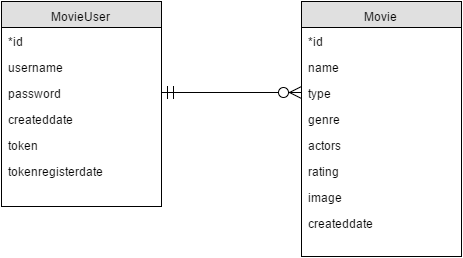


2.1 ábra: 2 rétegű architektúra

2 réteg fő tulajdonságai:

* Adat réteg: adattárolással és a lekérdezések kiszolgálásával foglalkozik
* Üzleti logikai réteg: üzleti folyamatok futtatásával, adatok manipulálásával, lekérdezésekkel, tranzakciókezeléssel, konkurenciakezeléssel és biztonsággal foglalkozik. Továbbá az adatok perzisztenciája és objektum-relációs adatmodellek közötti leképezés is a feladatainak részét képezi.

## Adatréteg



2.2 ábra: Egyed-kapcsolat diagram Crow's Foot jelölésrendszerrel

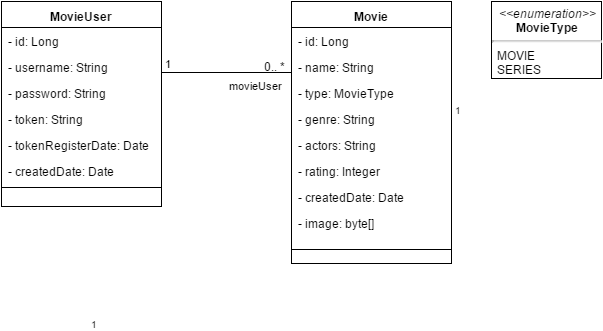
### Egyed-kapcsolat diagramon szereplő entitások és kapcsolataik

|  |  |
| --- | --- |
| **Entitás neve** | MovieUser |
| **Leírás** | Az alkalmazás a regisztrált felhasználók adatait tárolja. Névvel, jelszóval, regisztráció dátumával, tokennel és a token létrehozási dátumával rendelkezik. |
| **Kapcsolatai** | Movie [0..\*]: egy felhasználóhoz több film illetve sorozat tartozhat |

|  |  |
| --- | --- |
| **Entitás neve** | Movie |
| **Leírás** | A felhasználó által létrehozott filmek illetve sorozatok adatait tárolja. Film / sorozat nevét, típusát, műfaját, szereplőit, értékelését, borítóképét és létrehozási dátumát tartalmazza. |
| **Kapcsolatai** | MovieUser [1]: minden film / sorozatnak pontosan egy tulajdonosa van, mégpedig aki létrehozta. |

### ER diagramon szereplő entitások és kapcsolataik leképezése Java osztályokká

Az osztályokban szereplő valamennyi attribútumnak van getter illetve setter metódusa, kivéve az azonosítót (id).



2.3 ábra: ERD-ből képzett JPA entitások diagramja

### Felhasználó jelszavának tárolása adatbázisban

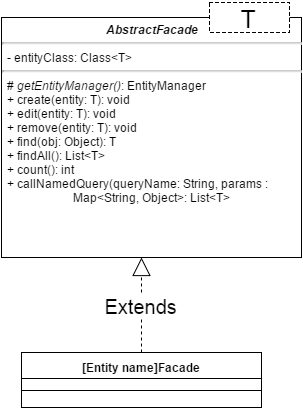
A jelszót biztonsági és jogi szempontok miatt a PBKDF2[[1]](#footnote-1) algoritmus felhasználásával tárolom el. Ez a következőképpen működik: a jelszóhoz hozzáfűz egy 24 byte méretű salt értéket, majd a kapott értékre lefuttatja az algoritmust, ezután ezt az értéket eltároljuk adatbázisban. Amikor a felhasználó bejelentkezést kísérel meg, akkor lefuttatjuk a felhasználó által beírt jelszóra az algoritmust, majd összehasonlítjuk az adatbázisban lévő értékkel, és ha a két hash-elt érték megegyezik, akkor a belépés sikeres volt. A salt hozzáfűzésével sokkal bonyolultabbá válik a jelszó feltörése.

## Üzleti logikai réteg

Az üzleti logikát állapotmentes session bean-ek használatával valósítottam meg. Ennek a rétegnek a feladata az adatbázisban lévő adatok hozzáférésének biztosítása.

Az adatok perzisztálását JPA használatával valósítottam meg. A CRUD[[2]](#footnote-2) műveleteket a session bean-ekbe injektált EntityManager-ek segítségével hajtottam végre. Minden egyes entitáshoz tartozik egy állapotmentes session bean. Továbbá van egy absztrakt generikus ősosztályom, amely tartalmazza a session bean-ek által használt közös metódusokat.

### Absztrakt generikus ősosztály



2.4 ábra: AbstractFacade

**Metódosuk funkciójának leírása**

* protected EntityManager getEntityManager(): egy absztrakt metódus, amely kikényszeríti a leszármazott osztályoknak, hogy megvalósítsák a getEntityManager-t
* void create(T entity): egy tetszőleges entitás példányt reprezentál
* void edit(T entity): egy tetszőleges entitás példányt módosít
* void remove(T entity): egy tetszőleges entitás példányt töröl
* T find(Object): egy adott entitás példányt keres elsődleges kulcs alapján. Sikeres találat esetén visszatér az objektummal, egyébként null-t ad vissza
* List<T> findAll(): visszadja az adott entitás összes példányát
* int count():visszadja az adott entitás összes példányának számát
* List<T> callNamedQuery(String queryName, Map<String, Object> params): Bemenő paraméterként kapott NamedQuery-t hívja meg a kapott paraméterekkel. Visszaadja a query-nek eleget tevő összes entitáspéldányt.

### Üzleti logikát megvalósító Session Bean-ek

Az egyes Session Bean-ek főbb funkciói kerülnek bemutatásra.

**MovieUserFacade**

|  |  |
| --- | --- |
| **Metódus** | + addUser(username: String, password: String): void |
| **Leírás** | Új felhasználó regisztrálása.  A regisztráció folyamata:   * input adatok validálása * ellenőrzés, hogy van-e már ilyen nevű felhasználó, ha igen, akkor a regisztráció sikertelen * jelszó hashelése * adatok perzisztálása |

|  |  |
| --- | --- |
| **Metódus** | + login(username: String, password: String): void |
| **Leírás** | Felhasználó belépési kísérlete.  A belépés folyamata:   * felhasználó létezésének ellenőrzése * megadott jelszó hash-elése és összehasonlítása az adatbázisban tárolt értékkel * token generálása, ami 3 részből tevődik össze: issuer + audience + uuid kulcs (mindegyik Base64 kódolással) |

|  |  |
| --- | --- |
| **Metódus** | + logout(username: String): void |
| **Leírás** | Felhasználó kilépési kísérlete.  A kilépés folyamata:   * felhasználóhoz tartozó token törlése |

**MovieFacade**

|  |  |
| --- | --- |
| **Metódus** | + createMovie(json: String, user: loggedUser): void |
| **Leírás** | Json formátumként kapott film / sorozat feldolgozása és perzisztálása a Movie táblába a megadott felhasználóhoz. |

|  |  |
| --- | --- |
| **Metódus** | + updateMovie(json: String, user: loggedUser): void |
| **Leírás** | A felhasználó egy már meglévő filmjének / sorozatának módosítása. |

|  |  |
| --- | --- |
| **Metódus** | + findAllMovieByUser(userId: Long): List<Movie> |
| **Leírás** | A felhasználóhoz tartozó összes film illetve sorozat lekérdezése. |

|  |  |
| --- | --- |
| **Metódus** | + getMovieFromJson(json: String, user: loggedUser): Movie |
| **Leírás** | Json formátumként kapott film illetve sorozat átalakítása Movie entitás objektummá. Ez a Gson Api felhasználásával történt. |

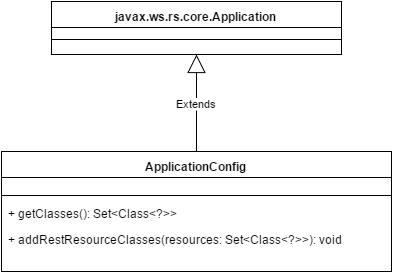
|  |  |
| --- | --- |
| **Metódus** | + findMovieByMovieIdAndUserId (movieId: Long): Movie |
| **Leírás** | A felhasználóhoz tartozó megadott movieId-val rendelkező filmet illetve sorozatot adja vissza. Ha nincs ilyen entitás, akkor null értékkel tér vissza. |

### Mobil klienssel való kommunikáció tervezése

A mobil klienssel való kommunikációt JAX-RS használatával kívánom megvalósítani.

RESTful webservice hívások esetén a szerver oldali erőforrás-állapotok URL által címezhetőek. Ezek általában HTTP kérések szoktak lenni.

**ApplicationConfig**



2.5 ábra: ApplicationConfig

ApplicationConfig osztályban kell beregisztrálni azokat az osztályokat, melyekben webservice metódusok vannak. Az ApplicationConfig Application osztályból való leszármazása biztosítja a JAX-RS beépített szolgáltatásait.

JAX-RS konfigurációkat a legegyszerűbben annotációk segítségével tudjuk változtatni. REST osztályok mindegyike injektálja a megfelelő állapotmentes Session Bean-t a @EJB annotációval. Ezzel biztosítom az EntityManager-hez való hozzáférést.

**MovieUserFacadeREST**

|  |  |  |
| --- | --- | --- |
| **Elérés** | filmmanager/rest/addUser | |
| **Metódus** | POST | |
| **Leírás** | Új felhasználó regisztrálása. | |
| **FORM paraméterek** | username=admin (String)  password=123456 (String) | |
| **Válasz típusa** | json | |
| **Válasz leírása** | {  username: ” username” (String)  message: ”Successful registration.” (String)  } | |
| **Válasz mezők jelentése** | username | Regisztrálást kezdeményező felhasználó neve. |
| message | Egy rövid leírást tartalmaz a regisztráció eredményéről. |

|  |  |  |
| --- | --- | --- |
| **Elérés** | filmmanager /rest/login | |
| **Metódus** | POST | |
| **Leírás** | Felhasználó belépési kísérlete. | |
| **FORM paraméterek** | username=admin (String)  password=123456 (String) | |
| **Válasz típusa** | json | |
| **Válasz leírása** | {  token: ”…”, (String)  } | |
| **Válasz mezők jelentése** | token | Sikeres belépés során a felhasználó egy kulcsot kap, amivel a rendszer szolgáltatásaihoz kap hozzáférést a beállított session idő lejártáig. Ha lejár a token, akkor újra be kell lépnie a felhasználónak. |

|  |  |  |
| --- | --- | --- |
| **Elérés** | filmmanager /rest/logout | |
| **Metódus** | POST | |
| **Leírás** | Felhasználó kilépési kísérlete. | |
| **HEADER paraméterek** | Authorization property az alábbi megadása: Bearer + token | |
| **Válasz típusa** | Response | |
| **Válasz leírása** | Sikeres kilépés esetén 200 OK | |
| **Válasz mezők jelentése** | token | Sikeres belépés során a felhasználó egy kulcsot kap, amivel a rendszer szolgáltatásaihoz kap hozzáférést a beállított session idő lejártáig. Ha lejár a token, akkor újra be kell lépnie a felhasználónak. |

**MovieFacadeREST**

Az összes rest hívás során az alábbi státuszok vannak:

|  |  |  |
| --- | --- | --- |
| **Válasz státusza** | 200 OK | Kérés sikeresen végrehajtva. |
| 401 Unauthorized | Token is invalid. – Érvénytelen token lett megadva.  Token is expired. – Újra be kell lépni, mert lejárt a session token érvényességi ideje. |

|  |  |  |
| --- | --- | --- |
| **Elérés** | filmmanager /rest/addMovie | |
| **Metódus** | POST | |
| **Leírás** | Új film felvétele. | |
| **HEADER paraméterek** | Authorization property-be az alábbi megadása: Bearer + token | |
| **Request json formában** | {  ”name”: ”Forrest Gump”, (String)  ”type”: ”movie”, (String)  ”genre”: ”drama, comedy”, (String)  ”actors”: ”Tom Hanks, Robin Wright”, (String)  ”rating”: ”5”, (String)  ”image”: ”…”, (String)  } | |
| **Json mezők jelentése** | name | Film/sorozat címe |
| type | MOVIE/SERIES |
| genre | Film műfaja |
| actors | Film szereplői |
| rating | Film osztályozása 1-5, ahol 5 a legjobb értékelés |
| image | Film borítóképe |

|  |  |
| --- | --- |
| **Elérés** | filmmanager /rest/getAllMovie |
| **Metódus** | POST |
| **Leírás** | Adott felhasználóhoz tartozó összes film és sorozat lekérése |
| **HEADER paraméterek** | Authorization property-be az alábbi megadása: Bearer + token |
| **Válasz típusa** | json |
| **Válasz leírása** | [  {  "id": 51,  "name": "Forrest Gump",  "type": "MOVIE",  "rating": 5  },  {  "id": 53,  "name": "Friends",  "type": "SERIES",  "rating": 5  }  ] |

|  |  |
| --- | --- |
| **Elérés** | filmmanager /rest/getMovieById/{id} |
| **Metódus** | POST |
| **Leírás** | Konkrét film/sorozat részletes adatainak lekérése |
| **HEADER paraméterek** | Authorization property-be az alábbi megadása: Bearer + token |
| **Válasz típusa** | json |
| **Válasz leírása** | {  "id": 51,  "name": "Forrest Gump",  "type": "MOVIE",  "genre": "drama, comedy",  "actors": "Tom Hanks, Robin Wright",  "rating": 5  } |

|  |  |
| --- | --- |
| **Elérés** | filmmanager /rest/deleteMovieById/{id} |
| **Metódus** | POST |
| **Leírás** | Konkrét film vagy sorozat törlése. |
| **HEADER paraméterek** | Authorization property-be az alábbi megadása: Bearer + token |

|  |  |
| --- | --- |
| **Elérés** | filmmanager /rest/updateMovie |
| **Metódus** | POST |
| **Leírás** | Film/sorozat értékeinek módosítása |
| **HEADER paraméterek** | Authorization property-be az alábbi megadása: Bearer + token |
| **Request json formában** | {  ”id”: 1, Kötelező (Long)  ”name”: ”Forrest Gump”, Opcionális (String)  ”type”: ”MOVIE”, Opcionális (String)  ”genre”: ”drama, comedy”, Opcionális (String)  ”actors”: ”Tom Hanks, Robin Wright”, Opcionális (String)  ”rating”: ”5”, Opcionális (String)  ”image”: ”…”, Opcionális (String)  } |

# Felhasznált technológiák, eszközök

## JavaEE

Java EE platform egy architektúra, amely a Java Standard Edition keretén alapszik, és számos olyan API-t szolgáltat, amely a többrétegű, elosztott és hibatűrő vállalati méretű alkalmazások fejlesztését támogatja a különböző internetes technológiák felhasználásával. Az alkalmazás elkészítése során a Java EE 7-et használtam.

## JAX-RS

A mobil klienssel való kommunikáció felépítésére JAX-RS API-ját használtam fel. A JAX-RS lehetővé teszi a REST architektúrájú webszolgáltatások létrehozását. Szerver oldali erőforrás-állapotok URL által címezhetőek, azaz a kliens egy létező URL címet meghívva kérheti le az adatokat.

## NetBeans

A Netbeans egy integrált fejlesztői környezet, melynek segítségével a fejlesztők egyszerűbben, gyorsabban és hatékonyabban tudnak fejleszteni. A Netbeans bárki számára ingyenes letölthető és használható korlátozások nélkül. Támogatást nyújt a Java EE 7-hez és a Glassfish legújabb verziójához is. A fejlesztés során a Netbeans 8.0.2-es verzióját vettem igénybe.

## Glassfish

A GlassFish egy nyílt forrású alkalmazás implementációja, amely támogatást nyújt a Java EE 7-hez. Könnyen integrálható a Netbeans környezettel. Az alkalmazásomat a GlassFish Server 4.1-es verziójára telepítettem.

## Apache Maven

A Maven egy ingyenes, nyílt forráskódú szoftver, amelyet szoftverprojektek menedzselésére és build folyamat automatizálására lehet használni. Leíró fájlját pom.xml-nek nevezzük, amely a projekt konfigurációját és függőségeit tartalmazza.

## Gson

A Gson egy nyílt forráskódú Java könyvtár, aminek segítségével Java objektumból JSON adatot (szerializálás) és JSON adatból Java objektumot (deszerializálás) tudunk konvertálni. A REST webservice hívások során használtam a kérés feldolgozására illetve a válasz előállítására.

## MySQL

A MySQL egy platformfüggetlen SQL-alapú relációs adatbázis-kezelő szerver. MySQL Community Edition-t használtam InnoDB tároló motorral a fejlesztés során.

## Postman

A megírt RESTFul webszolgáltatások működését a Postman alkalmazással teszteltem. A Postman egy REST kliens, aminek segítségével gyorsan lehet RESTFul kéréseket készíteni és a válasz Internet média típusát is meg lehet vele határozni.

## Git

A Git egy ingyenes verziókezelő rendszer, amely megkönnyíti a fejlesztők életét. A NetBeans támogatást nyújt a Git használatához, így egyszerűen és gyorsan lehetett a projekten commit, push és pull műveletet végezni.

## Draw.io

Egy webes rajzoló program, amellyel kényelmesen lehet különböző diagramokat készíteni. Ebben a dokumentációban található összes ER és UML diagram ezzel az ingyenes alkalmazással készült el.

# Telepítési dokumentáció

## Szerver oldal beüzemelése

Az alkalmazás lokális alkalmazásszerveren és adatbázison fut. Ebben a fejezetben részletesen leírom a telepítési lépéseket.

A telepítés lépései:

* MySQL 6.2-es verziójának telepítése, ahol létre kell hozni egy filmmanager nevű sémát
* Netbeans IDE 8.0.2-es verziójának telepítése és projekt importálása
* Netbeans-ben új adatbázis kapcsolat létrehozása: Services fül → Database → New connection → MySQL (Connector / J driver) kiválasztása → JDBC url + autentikáció beállítása
* Glassfish alkalmazásszerverben JDBC resource beállítása: Glassfish elindítása → Glassfish console megnyitása
  + Resources fül → JDBC → JDBC Connection Pools → New
  + Pool name: filmManagerPool
  + Resource Type: java.sql.Driver
  + Add mysql username/password property, add mysql url: jdbc:mysql://localhost:3306/filmmanager
  + Resources fül → JDBC → JDBC Resources → New

JNDI name: filmManager

Pool name: filmManagerPool

* JPA entitásokból Mysql táblák generálása: projektben lévő persistence.xml stratégiájának Create-re állítása → projekt build → projekt deploy → projekt futtatása → persistence.xml stratégiájának visszaállítása None-ra.

Ábrajegyzék

2.1 ábra: 2 rétegű architektúra 4

2.2 ábra: Egyed-kapcsolat diagram Crow's Foot jelölésrendszerrel 5

2.3 ábra: ERD-ből képzett JPA entitások diagramja 6

2.4 ábra: AbstractFacade 7

2.5 ábra: ApplicationConfig 10

1. PBKDF2: <http://en.wikipedia.org/wiki/PBKDF2> [↑](#footnote-ref-1)
2. CRUD = Create Read Update Delete [↑](#footnote-ref-2)