



Eötvös Loránd Tudományegyetem Informatikai Kar
Programozási Nyelvek és Fordítóprogramok Tanszék

Aukciós ház webes alkalmazás

Témavezető:

Kitlei Róbert
Tanársegéd

Szerző:

Bakos János
programtervező informatikus
Bsc

Budapest 2017

TARTALOM

1	Bevezetés.....	3
1.1	Felhasználói rendszerkövetelmények.....	3
2	A program ismertetése.....	4
2.1	Kezdőképernyő.....	4
2.2	Regisztráció / Bejelentkezés.....	4
2.3	Saját profil (Fülek)	5
2.3.1	My profile.....	6
2.3.2	Edit profile.....	6
2.3.3	My Auctions	6
2.3.4	My Items	6
2.3.5	My Bids.....	7
2.3.6	Notifications.....	7
2.4	Tárgy hozzáadása	8
2.5	Aukciót létrehozása.....	9
2.6	Aukciók böngészése	10
2.7	Licitálás (Aukció részleteinek megtekintése)	11
2.8	Felhasználók értékelése	12
2.9	Felhasználó értékeléseinek megtekintése	13
2.10	Felhasználók listázása	13
3	Fejlesztői dokumentáció.....	14
3.1	Üzembe helyezés, telepítés	14
3.1.1	Projekt elérhetősége.....	14
3.1.2	Üzemeltetői, fejlesztői rendszerkövetelmények (ajánlot).....	14
3.1.3	Adatbázis beállítása (Derby)	14
3.1.4	Alkalmazáserver beállítása (Glassfish).....	15
	Telepítés (deploy)	17
3.2	Használt technológiák	18
3.3	Architektúra – EJB réteg.....	19
3.3.1	JPA.....	19
3.3.2	Entitások	19
3.3.3	Kapcsolati ábrák.....	23
3.3.4	Managerek	

3.3.5	Üzleti logika.....	28
3.3.6	Ütemezők.....	29
3.3.7	Naplózás.....	29
3.4	Architektúra – WEB réteg	32
3.4.1	Megjelenés (kliens)	32
3.4.2	Vezérlők	33
3.4.3	Hibakezelés	33
3.5	Tesztelés.....	34
3.5.1	Automatikus tesztek	34
3.5.2	Manuális tesztesetek	34
3.6	Továbbfejlesztési lehetőségek	36
3.6.1	Konfiguráció	36
3.6.2	Értesítések.....	36
3.6.3	Back End validációk.....	37
3.6.4	Extrém inputok	37
3.6.5	Dao réteg bevezetése	37
3.6.6	ID-k kiváltása	37
4	Irodalom	37
5	Mellékletek.....	39
5.1	Adatbázis tábla létrehozó scriptek.....	39

1 BEVEZETÉS

A megoldandó feladat egy olyan webes alkalmazás készítése ahol a felhasználóknak lehetőségük van különféle tárgyakat aukcióra bocsátani. Regisztráció nélkül csak böngészhetünk a meghirdetett tárgyak között. Regisztrálás után lehetőségünk nyílik ezekre az elemekre licitálni, illetve a profil oldal alatt saját tárgyakat feltölteni, azokat aukciókba szervezni, illetve személyes adatainkat módosítani / megtekinteni. A kívánt termék megtalálásához keresési feltételekkel szűkíthetjük a találatok számát. A felhasználóknak kettő fajtáját különböztetjük meg, sima illetve adminisztrátori jogosultságokkal rendelkező.

1.1 FELHASZNÁLÓI RENDSZERKÖVETELMÉNYEK.

Az alkalmazást webes felületen keresztül érhetjük el így használatához nincs másra szükségünk, mint egy böngészőre. Az esetleges kompatibilitási (megjelenési) problémák elkerülése végett törekedjünk minél frissebb verziót használni ezekből. A felületen megjelenő elemek bizonyos mértékig skálázódnak, de alapvetően normál monitor méretre van optimalizálva.

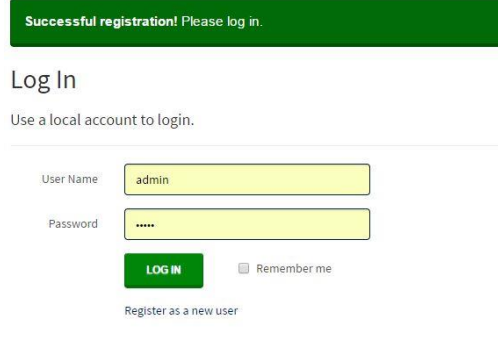
2 A PROGRAM ISMERTETÉSE

2.1 KEZDŐKÉPERNYŐ

Az elérési cím beírása után vendégként érkezünk az oldalra. Itt lehetőségünk van a többi felhasználó által meghirdetett aukciók között böngészni. A funkciók megjelenése korlátozott, elérésükhöz bejelentkezésre van szükség.

2.2 REGISZTRÁCIÓ / BEJELENTKEZÉS

A regisztrációs illetve a bejelentkezéshez szükséges felületet a menü sorból a megfelelő link-re kattintva érjük el. Regisztrációhoz szükséges egy felhasználónév illetve jelszó megadása. A felhasználónevek egyediek a rendszerben, így ha az általunk megadott név már foglalt új nevet kell választanunk. A jelszónak és az megerősítő jelszónak értelemszerűen egyeznie kell. A mezők kitöltésének szükségességéről is értesülünk.



Successful registration! Please log in.

Log In

Use a local account to login.

User Name: admin

Password:

LOG IN

☐ Remember me

[Register as a new user](#)

2.1 Bejelentkező felület

Create new account



User Name: [Empty field]

Password: [Empty field]

Confirm Password: [Empty field]

REGISTER

Already has account?

Username is required!

Password is required!

Password confirmation is required!

2.2 Regisztrációs felület, üresen hagyott mezőkkel való próbálkozás

2.3 SAJÁT PROFIL (FÜLEK)

Bejelentkezést követően a menüsorban megjelenik a profil oldalunkra vezető link. A bal oldali megjelenő fülek között váltva tudjuk szerkeszteni a profilunkat, nyomon követni a licitálásainkat, aukcióinkat. Továbbá itt van lehetőségünk tárgyak feltöltésére, értesítéseink elolvasására.

Welcome Bakos János"

My Profile
Edit profile
My Auctions
My items
My Bids
Notifications

	
Login Name:	admin
Real Name:	Bakos János
Date of birth	Mon Nov 02 01:00:00 CET 1992
Roles:	admin, user,
Email:	noreply@email.hu
Phone number:	06301234567

2.3 Profil oldal, "My Profile" fül

2.3.1 My profile

Az adatainkat jeleníti meg szerkesztési lehetőség nélkül. Továbbá itt található a felhasználó értékeléseit megjelenítő felületre vezető link.

2.3.2 Edit profile

A szerkeszthető adatainkat, profilképünket tudjuk itt módosítani. A szerkeszthető mezők között nincs kötelező mező.

2.3.3 My Auctions

Az aktív és már lezárt aukciókat soroljuk fel táblázatos formában.

2.3.4 My Items

A feltöltött termékeink jelennek meg táblázatos formában. Azon elemek melyek még nincsenek aukcióhoz kötve „Create Auction” gomb jelenik meg melyre kattintva átnavigálunk az aukció létrehozásához szükséges felületre. Amennyiben már aukcióhoz van rendelve a tárgy akkor a „View Auction” gomb jelenik meg mellyel az aukció megtekintésére navigálunk.

Id	Item name	Item name
352	flower	CREATE AUCTION
401	asdas	VIEW AUCTION
451	asdasdas	VIEW AUCTION
		<div>New</div> <div>Delete</div>

2.4 Termékeket megjelenítő táblázat

Továbbá ezen a fülön van lehetőségünk tárgyakat feltölteni, törölni. Törölni csak olyan elemeket lehet, ami még nincs aukcióhoz kötve. Az add gombra kattintva jutunk el a tárgy hozzáadása felületre.

2.3.5 My Bids

Az eddigi aktív illetve már lezárt aukciókra tett licitjeink jelennek meg táblázatos formában. Egy aukcióra tett több licit esetén mindig csak az utolsó jelenik meg a táblázatban. (tehát aukciónként csoportosítva vannak a legutolsó licitjeink). Az aktív táblázatban jelöljük, hogy épp vezet-e a licitünk az adott aukción, míg a lezárt aukciónál azt jelöljük, hogy megnyertük-e a licitálást. Amennyiben megnyertük úgy megjelenik egy felhasználó értékelése gomb.

Expired Auction bids			
Closed bids			
	Bid Id	Auction Id	Winner
C	203	202 VIEW AUCTION	true RATE USER

2.5 A lezárt aukciókra tett licitjeink táblázata

2.3.6 Notifications

A rendszer által küldött értesítéseinket tekinthetjük meg. Értesítést kaphatunk arról, hogy megnyertünk egy aukciót, illetve arról hogy lezárult egy meghirdetett aukciónk. A már elolvasott üzeneteket törölhetjük.

New			
Id	Title	Sender	Read
101	This is a title with timepam Fri May 12 14:26:16 CEST 2017	Demo sender	READ
102	This is a title with timepam Fri May 12 14:26:16 CEST 2017	Demo sender	READ
204	Auction Won	ESPACE noreply	READ
251	Auction Won	ESPACE noreply	READ
Readed			

2.6 Új értesítések listája

Sender:	ESPACE noreply
Message:	You have won this auction: 202, please contact with the seller!
Date:	Fri May 12 17:44:00 CEST 2017

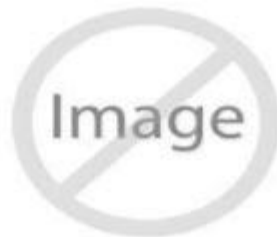
2.7 Egy megnyitott üzenet

2.4 TÁRGY HOZZÁADÁSA

A profil oldalon a „My Items” fülről kerülhetünk erre az oldalra. A tárgynak kötelező nevet adni illetve kategóriát választani.

Create new item

Illustration:



Product name:

Category:

2.8 Tárgy létrehozása felület

2.5 AUKCIÓT LÉTREHOZÁSA

A profil oldalon a „My Items” fülről kerülhetünk erre az oldalra. A még aukció nélküli tárgyakkal megjelenő „Create Auction” gombra kattintva.

„Auction header” – ez az aukció címe, kötelező.

„Starting price” – minimálisan ajánlható licit.

„Start time” – ekkor jelenik meg a kereshető aukciók között

„Expire time” – ekkor záródik le az aukció, kötelező

„Description” – leírása az aukciónak.

Create new auction

Auction header	<input type="text"/>
Starting price	<input type="text"/>
Start time:	<input type="text"/>
Expire Time:	<input type="text"/>
Description	<input type="text"/>

CREATE AUCTION

2.6 AUKCIÓK BÖNGÉSZÉSE

A menüsáv bal oldalán található list auctions fülre (illetve az Espace logora) kattintva jutunk azon oldalra ahol az aukciók között válogathatunk.

A keresés megkönnyítéséhez az alábbi szűrési feltételeket adhatunk meg:

- Item name: a meghírdetett tárgy nevében való keresés
- Title: az aukció címében való keresés
- Minimum price: a legkisebb ajánlható összegre való szűrés. Amennyiben még nem licitáltak a termékre akkor ez az érték a „start bid” lesz. Ha már van licitáló akkor a legtöbbet licitáló értékre szűr.
- Maximum price: felső értékhatár, értéke a „minimum price”-hoz meg egyező logikán alapul.
- Description: Az aukció leírásában való keresés
- User: Az aukciót feladó felhasználó nevében való keresés
- Kategóriák: jelöletlen mező esetén nem szűrünk. Jelölt mező esetén csak az adott kategóriába eső tárgyak aukcióját jelenítjük meg.

Filters



Item name:	<input type="text"/>	Description:	<input type="text"/>
Title:	<input type="text"/>	User:	<input type="text"/>
Minimum price:	<input type="text"/>	Maximum price:	<input type="text"/>
<input type="button" value="Reset"/>	<input type="button" value="Submit"/>	Order:	Newest first <input type="button" value="v"/>

☐ item_category_1
☐ item_category_2
☐ item_category_3
☐ item_category_4
☐ item_category_5
☐ autók
☐ lakás, kertészet

2.10 Szűrő mezők a listázott aukciókhoz

A találatokat táblázatos formában jelenítjük meg.

Auctions

(1 of 23) 1 2 3 4 5 6 7 8 9 10 >> > 3	
	Title: Virágocska Desc: asd Min Bid: 2000.0 Top Bid: 2200.0 Close: 05/10/2017 03:07:00 View
	Title: Virag Desc: virag vagyok virág de egy hosszú virágvirag vagyok virág de egy hosszú virágvirag vagyok virág de egy hosszú virágvirag vagyok virág de egy hosszú virág Min Bid: 1000.0 Top Bid: 1700.0 Close: 11/01/2018 10:10:00 View


2.11 A találatok listája

2.7 LICITÁLÁS (AUKCIÓ RÉSZLETEINEK MEGTEKINTÉSE)

Az aukciókat megjelenítő táblázatban a kiválasztott elem „view” gombjára kattintva átnavigálunk annak részletes adatait megjelenítő oldalra. Itt tudunk ajánlatot is tenni. Az ajánlat nem lehet kisebb, mint a „Starting bid” illetve a legmagasabb ajánlatnál, ha ez létezik.

Hibaüzenetet kapunk, ha saját aukcióra akarunk licitálni, illetve ha az aukció már lezárult.

test




Header:	test
Name:	desert
Description:	description
Start date:	Sun May 07 23:57:10 CEST 2017
Expiration date:	Thu Nov 01 11:10:00 CET 2018
Starting bid:	20.0
Current bid:	810.0
Category:	item_category_1
	MAKE BID
Bid:	<input type="text"/>

2.12 Aukció részletei

2.8 FELHASZNÁLÓK ÉRTÉKELÉSE

Azután van lehetőségünk értékelni egy felhasználót, hogy annak egy aukcióját sikeresen megnyertük. Az értékeléshez a „Profile -> My Bids” fül alatt az „Expired bids” táblázatban találjuk a lezárt licitjeinket. Itt jelenik meg a „Rate user” gomb is amennyiben nyertesek lettünk az licittel. A gombra kattintva egy felugró ablakban választhatunk 1-5 „csillagot” illetve szöveges értékelést is hozzá kell fűznünk.



The image shows a 'Rate User' dialog box. It has a title bar with the text 'Rate User' and a close button (X). The main area contains a form with two columns. The left column has a 'rate:' label and a 'Text:' label. The right column has a star rating system with five stars (the first three are yellow, the last two are grey) and a text input field containing the text 'Kiváló eladó!'. Below the form is a 'Send' button.

2.13 Felugró ablak a felhasználó értékeléséhez

2.9 FELHASZNÁLÓ ÉRTÉKELÉSEINEK MEGTEKINTÉSE

Az értékeléseket egy felhasználó megtekintése oldalon nézhetjük meg. Ezt a felületet elérhetjük az „aukciók listázása” felületről a hirdető nevére kattintva. Illetve a saját oldalunk „My Profile” fül alatt a „View My Ratings” hivatkozásra kattintva.

Amennyiben a felhasználó akit épp nézünk nem mi vagyunk akkor megjelennek bal oldalt az adatai is. Ha saját magunk értékeléseit olvassuk, akkor csak a táblázat jelenik meg.

		Ratings			
User Name:	user_1	Date	Rate	Writer	Text
Real Name:		Fri May 12 18:15:49 CEST 2017	3	admin	Kiváló eladó!
Email:		Fri May 12 18:16:36 CEST 2017	3	admin	Kiváló eladó!
Phone number:		Fri May 12 18:29:05 CEST 2017	3	admin	nice guy
Number of rates:	3				
AvgRate:	3.0				

2.14 Felhasználó értékeléseinek, adatainak megtekintése

2.10 FELHASZNÁLÓK LISTÁZÁSA

Ez a felület csak adminisztrátori jogosultsággal érhető el. Jogosultságok kiosztására van itt lehetőségünk.

List of Users

Search name:

Id	Login Name	Name	Date of Birth	Status
1	admin	Unknown	Wed May 03 00:00:00 CEST 2017	MAKE AS USER
9	user_1	Unknown	Unknown	PROMOTE ADMIN
11	user_2	Unknown	Unknown	PROMOTE ADMIN
13	user_3	Unknown	Unknown	PROMOTE ADMIN

2.15 Felhasználókat listázó táblázat

3 FEJLESZTŐI DOKUMENTÁCIÓ

3.1 ÜZEMBE HELYEZÉS, TELEPÍTÉS

A projektnek kód szinten nincs erős alkalmazás illetve adatbázis szerver specifikus függősége így kis módosítással tetszőleges struktúrára telepíthető. A jelenlegi dokumentáció Windows 7 alatt Derby adatbázissal és Glassfish alkalmazásszerverrel való üzemeltetést részletezik. Nagy terheltségű felhasználás esetén javasolt Weblogic Oracle Server és adatbázis használata.

3.1.1 Projekt elérhetősége

A projekt elérhető a „<https://github.com/deveroxxx/Espace/>” címen illetve a mellékelt cd-n.

3.1.2 Üzemeltetői, fejlesztői rendszerkövetelmények (ajánlot)

- 64 bites Windows 7 (vagy újabb) operációs rendszer.
- Java 8
- Glassfish 4.1 [<https://glassfish.java.net/>]
- Apache Derby [<http://db.apache.org/derby/>] db-derby-10.13.1.
- Maven [<https://maven.apache.org/>]

3.1.3 Adatbázis beállítása (Derby)

Másoljuk vagy töltsük le a derby-t egy tetszőleges mappába. (Ez legyen most a „C:\Apache\derby” ahol a „derby” a főkönyvtár)

Nyissuk meg a windows konzolt. (Futtatás -> cmd.exe)

Állítsuk be a „DERBY_INSTALL” környezeti változót az alább parancs futtatásával.

```
„set DERBY_INSTALL=C:\Apache\derby”
```

Lépünk bele a telepítési helyre

```
„cd %DERBY_INSTALL%\bin”
```

A „CLASSPATH” egyszerű beállításához futtassuk az alábbi parancsot

```
C:\Apache\derby\bin> „setNetworkServerCP.bat”
```

Az adatbázis létrehozásához több lehetőségünk is van. A létrehozó sémák megtalálhatók a mellékelt cd-n a „createSQL.sql” fájlban.

IJ-vel való létrehozás leírása:

[http://db.apache.org/derby/papers/DerbyTut/ij_intro.html]

3.1.4 Alkalmazáserver beállítása (Glassfish)

Az alkalmazás szervert a „startserv.bat” konzolból való futtatásával indíthatjuk el. Érdemes windows service-ként felvenni így a szervergép újraindulása esetén automatikusan elindul az alkalmazás szerver is. Az alkalmazás szervert úgynevezett „domain”-ek konfigurálják. Frissen telepített GlassFish esetén alapértelmezetten létrejön egy „domain1” nevű domain. Hozhatunk létre újat vagy konfigurálhatjuk ezt is az alkalmazásunkhoz.

3.1.4.1 Adatbázis elérés

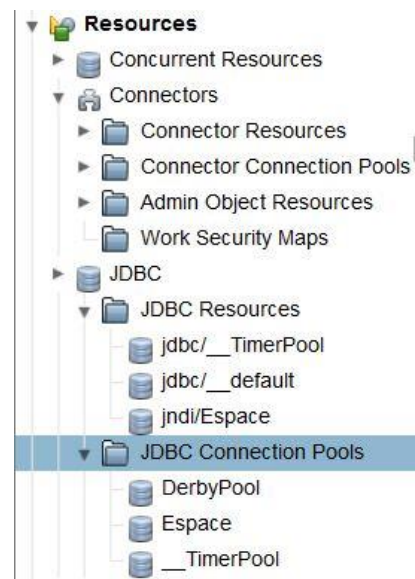
Ahhoz hogy elérjük az adatbázisunkat fel kell vennünk egy új JDBC connection pool-t. Ehhez navigáljuk az ábrán látható menühöz majd adjuk hozzá egyet „Espace” névvel. Itt két két fontos beállítás van.

A Datasource Classname:

org.apache.derby.jdbc.ClientDataSource

Resource Type:

javax.sql.DataSource



3.1 Glassfish, erőforrások menü

Ezután adjuk hozzá a JDBC Resources-hez. JNDI név: „jdbc/Espace”, Connection pool az imént hozzáadott „Espace”.

3.1.4.2 Felhasználó kezelés

Ahhoz hogy felhasználói be és kijelentkezéseket, illetve jogosultságokat tudjunk kezelni fel kell venni egy „Security realm”-ot s ezt beállítani alapértelmezettként. Az ábrán (3.3) látó menüpontból kattintsunk a „New...” ikonra. Az elnevezés tetszőleges lehet. A ClassName-nél válasszuk a:

`com.sun.enterprise.security.auth.realm.jdbc.JDBCRealm`

A többi beállítást az ábra szerint végezzük el. Fontos, ha ezektől eltérünk akkor azt kövessük le a User-t leíró táblákban is.



3.2 Glassfish, beállítások menü

Properties specific to this Class

JAAS Context: *	<input type="text" value="jdbcRealm"/> Identifier for the login module to use for this realm
JNDI: *	<input type="text" value="jndi/Espace"/> JNDI name of the JDBC resource used by this realm
User Table: *	<input type="text" value="UserTable"/> Name of the database table that contains the list of authorized use
User Name Column: *	<input type="text" value="userName"/> Name of the column in the user table that contains the list of user i
Password Column: *	<input type="text" value="password"/> Name of the column in the user table that contains the user passw
Group Table: *	<input type="text" value="RoleTable"/> Name of the database table that contains the list of groups for this
Group Table User Name Column:	<input type="text" value="userName"/> Name of the column in the user group table that contains the list o
Group Name Column: *	<input type="text" value="groupRole"/> Name of the column in the group table that contains the list of gro
Password Encryption Algorithm: *	<input type="text" value="SHA-256"/> This denotes the algorithm for encrypting the passwords in the dai

3.3 Security Realm beállításai

3.1.4.3 Naplózás beállítás

Az alkalmazás log4j-t használ. Ennek megfelelő működéséhez szükség van annak bekonfigurálására.

A „Configuration -> server-config -> JVM Settings -> JVM Options” fülön az „Add JVM Option”-al adjuk hozzá az alábbi sort:

```
„-Dlog4j.configuration=file:///${com.sun.aas.instanceRoot}/config/log4j.properties”
```

Ezzel mondjuk meg, hogy hol keresse a konfigurációs fájlt.

Telepítés (deploy)

Az alkalmazásunk futtatásához 2 fájl telepítése szükséges. A cd mellékleten megtalálható a „Espace-ejb-1.0-SNAPSHOT.jar” és a „Espace-web-1.0-SNAPSHOT.war” csomagok. A war-nak függősége az ejb, ezért telepítéskor az ejb-t rakjuk fel előbb a szerverre.

Fejlesztőkörnyezet nélküli telepítés:

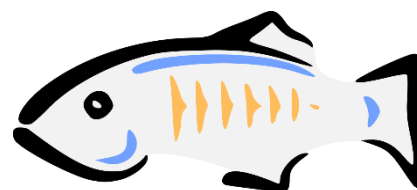
A Glassfish „Applications” menüpont alatt kattintsunk a „deploy”-ra majd tarlózzuk a megfelelő fájlt. A szükséges mezők automatikusan kitöltődnek. Majd „Ok” gomb.

Fejlesztőkörnyezetben (IntelliJ):

Konfiguráljunk fel egy alkalmazásszerveret, majd a deployment-nél az ejb és a war artifactot adjuk hozzá.

3.2 HASZNÁLT TECHNOLOGIÁK

Az alkalmazás JAVA EE 7 alapú. Az adatbázis egy Derby server mely ideális „demo” projektekhez könnyű konfigurálása miatt. A projekt Maven struktúrára épül így a függőségek kezelésével nem kell külön foglalkoznunk azon felül, hogy a pom.xml-ben megfelelően feltüntetjük őket. Két fő részre bontható, Back End illetve Front End. A Back End az adatbázisra EclipseLink-en keresztül kapcsolódik, az adatbázis béli struktúrát java oldalon Enitítások reprezentálják. Az entitásokat EJB beanok (nevezzük őket managgereknek) kezelik. Az üzleti logikát Serveice-k valósítják meg melyek a managgereket CDI-on keresztül érik el. A Front End kezeléséért a JavaServer Faces (JSF) keretrendszer felel Primefacessal kiegészítve. Az egyes nézeteket ManagedBean-eken keresztül vezéreljük. A megjelenést Bootstrapel egészítjük ki. Az alkalmazás szerver Glassfish, választás rá az adatbázishoz hasonlóan az egyszerű konfigurálhatósága miatt esett.



3.3 ARCHITEKTÚRA – EJB RÉTEG

3.3.1 JPA

Ahhoz hogy a java persistence api (jpa)-t használni tudjuk, szükségünk van egy persistence.xml file-ra EJB esetén a META-INF alá. A JPA előnye hogy elfedi előlünk a különböző adatbázisokkal történő kapcsolattartás specifikusságait.

3.3.2 Entitások

Az adatbázis táblákat Java oldalon Entitásokkal tudjuk reprezentálni. Az entitások és az adatbázis között a JPA tartja a kapcsolatot. Ahhoz hogy létrehozzunk egy entitást szükségünk van egy POJO—ra (egyszerű java osztály) melyet megannotálunk a @Entity-vel illetve neki egy mezőjét @Id-val ezzel jelezve hogy ő rá entitásként tekintsen a fordító. Az entitásaink a memóriában élnek.

3.3.2.1 *Base Entity*

Az összes entitásnak az ősoosztálya.

Mezők:

- id – generált érték, adatbázis kulcs
- deleted – flag, nem fizikai törlés jelölése
- createdOn – Az entitás létrehozásának dátuma

3.3.2.2 *User*

A regisztrált felhasználót reprezentáló entitás.

Mezők:

- phoneNumber - telefonszám
- notifications – értesítések listája
- emailAdress – email cím
- items – tárgyak listája melyeket feltöltött a felhasználó
- realName – valódi név (Pl.: Kovács Béla)
- myBids –eddig licitek
- myAuctions – létrehozott aukciók

- dateOfBirth – születési dátum
- userName – felhasználó név (bejelentkezési egyben, egyedi)
- picture – felhasználó profilkép elérési útvonal
- password – jelszó SHA256-ban tárolva
- witedRaitings – A felhasználó által írt értékelések listája
- myRaitings – A felhasználó értékelései

3.3.2.3 *UserRating*

Az adott felhasználó értékeléseit reprezentáló entitás.

- rating – értékelés pontszáma (0-5)
- text – szövege az értékelésnek
- auction – mely aukcióhoz tartozik
- recipient – akit értékeltünk felhasználó
- sender – az értékelő felhasználó

3.3.2.4 *GroupRole*

Felhasználót és jogosultságot összekötő entitás.

Mezők:

- userName – felhasználó bejelentkezési neve
- groupRole - ENUM, jogosultságot reprezentál (admin, user...)

3.3.2.5 *Item*

A felhasználó által létrehozott tárgyakat leíró entitás.

Mezők:

- name – tárgy neve
- picture – képhez tartozó elérési útvonal
- auction – melyik aukcióhoz van kötve
- category - kategória
- user – kié a tárgy

3.3.2.6 *ItemCategory*

Tárgyakak kategóriáját leíró entitás.

Mezők:

- name – kategória neve (Egyedi)
- items – ebbe a kategóriába eső tárgyak listája

3.3.2.7 *Auction*

A felhasználó által létrehozott aukciót leíró entitás.

Mezők:

- minimumBid – ez a minimum ajánlható ár
- startDate – mikortól kezdődjön az aukció
- expirationDate – mikor érjen véget
- description – leírása az aukciónak
- topBider – a licitálók közül az utolsó. (számolható lenne, de sok lekérdezés ki optimalizál)
- item – az aukcióhoz tartozó tárgy
- owner – hirdető felhasználó
- closed – lezárult-e az aukció
- header – fejléc szöveg
- bids – eddigi licitek
- userRating – Az aukcióhoz kötődő felhasználói értékelés

3.3.2.8 *Bid*

Egy konkrét licitet leíró entitás.

Mezők:

- auction – mely aukcióra vonatkozik
- user – ki volt a licitáló
- bid – a licit mértéke

3.3.2.9 *Notification*

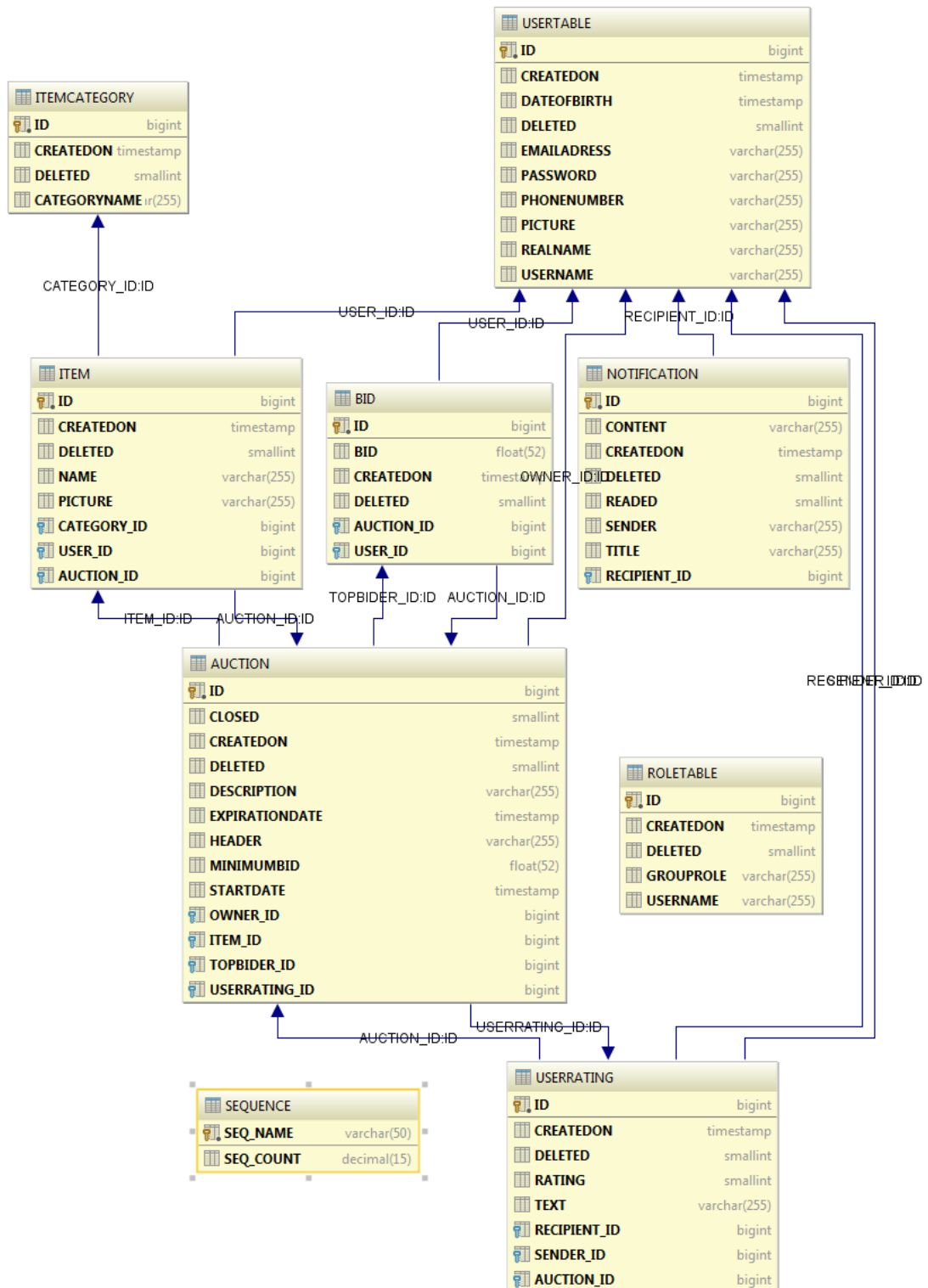
Felhasználónak címzett rendszer üzenetet reprezentál.

Mezők:

- sender – Küldő
- content – értesítés szöveges tartalma
- readed – Elolvasta-e a felhasználó (megnyitotta)
- title – értesítés fejléc
- recipient – felhasználó, akinek szól

3.3.3 Kapcsolati ábrák

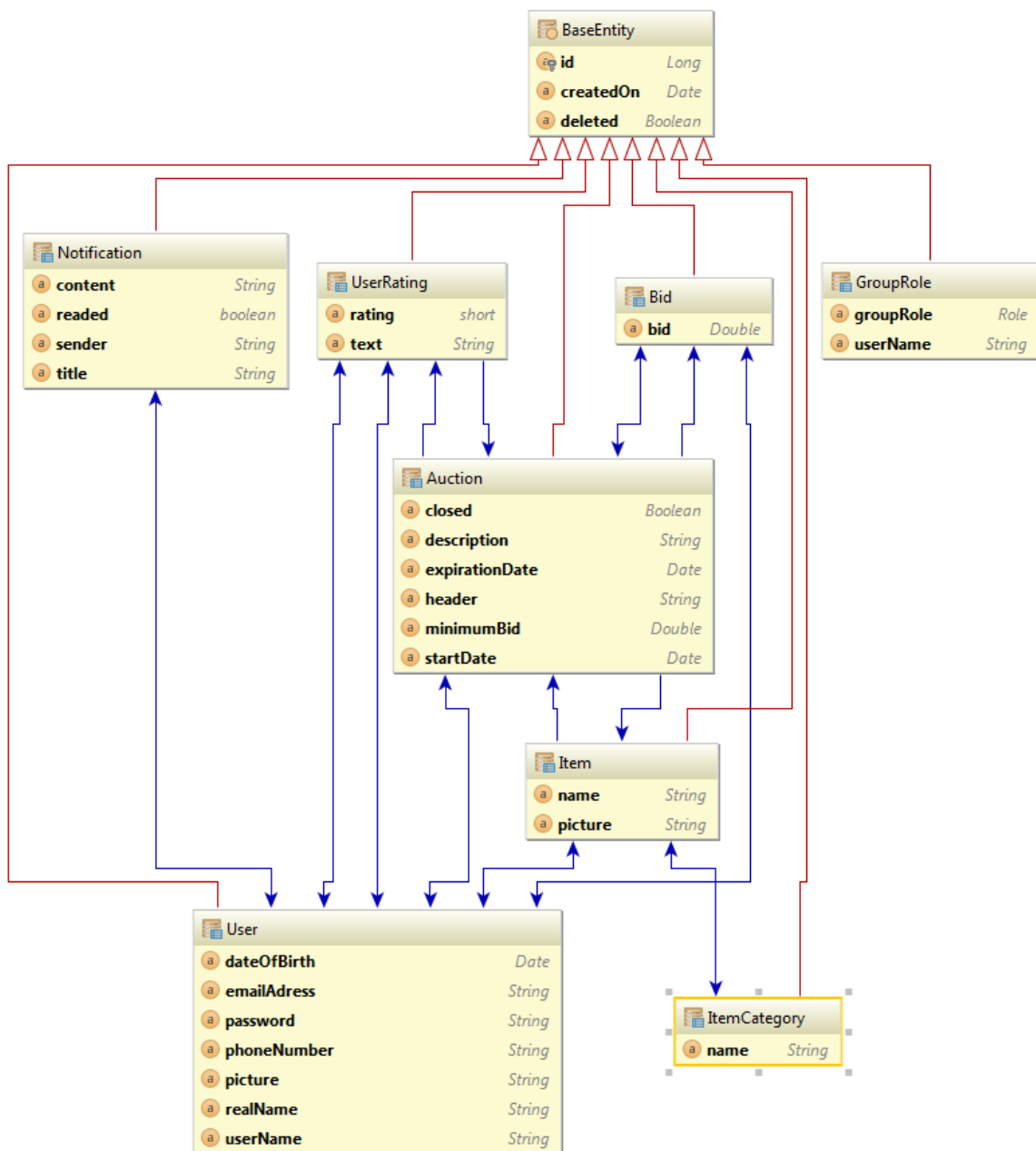
3.3.3.1 Adatbázis



Powered by yFiles

3.4 Adatbázis táblák és kapcsolataik

3.3.3.2 Entitás



Powered by yFiles

3.5 Entitások és kapcsolataik

3.3.4 Managerek

Az entitásokat managerekon keresztül kezeljük. Céljuk elsősorban az adatbázis táblák írása, olvasása üzleti logika nélkül. A közös műveletek ki lettek szervezve egy abstract „TemplateManagerBase<T>” osztályba. Ezt „extend”-eli a TemplateManager<T> mely egyben implementálja az entity manager-t.

```
@PersistenceContext(unitName="EspacePersistence")
protected EntityManager em;

@Override
protected EntityManager getEntityManager() {
    return em;
}
```

Ez a felosztás kódbővítés szempontjából hasznos, ha más persistens contex-ben dolgozó managereket is implementálunk.

3.3.4.1 TemplateManager főbb függvények

(T az egy generikus paraméter)

Név	add
Paraméter	T t
Visszatérés	t
Működés	persistálja a t entitást

Név	get
Paraméter	Long id
Visszatérés	T
Működés	referenciát ad vissza a T entitásra (első ráhivatkozáskor történik a fetch, ha nem létezik Exception-t kapunk)

Név	select
Paraméter	Long id
Visszatérés	T
Működés	id alapú lekérdezés, találat esetén T egyébként null

Név	selectForUpdate
Paraméter	Long id
Visszatérés	T
Működés	id alapú lekérdezés, találat esetén T egyébként null annyi bővítéssel hogy pesszimista írás jogot rak az objektumra. (Tehát a tranzakció végéig más nem tudja írni / olvasni)

Név	listAll
Paraméter	
Visszatérés	List<T>
Működés	az összes elemet visszaadja a táblából vagy üres listát ha nincs találat

Név	listAll
Paraméter	String query
Visszatérés	List<T>
Működés	az összes elemet visszaadja a táblából vagy üres listát ha nincs találat a query string alapján pl.: „select t from T where t.id > 20”

Név	listByFilter
Paraméter	String query, Map<String, Object> params
Visszatérés	List<T>
Működés	paraméterezett query lekérdezés. Fontos hogy ha a query stringben adtunk meg paramétert azt a params-ba is töltsük.

Név	deleteById
Paraméter	Long id
Visszatérés	void
Működés	Fizikai törlés az adatbázisból id alapján

Név	update
Paraméter	T t
Visszatérés	t
Működés	mergeli az entitás állapotát a persistens contexbe

3.3.4.2 Implementáló manager

Minden entitáshoz tartozik egy manager mely implementálja a „TemplateManager<T>” az Entitás class-al. Az őszosztályban lévő alap függvények a „getMyClass()” abstarct metóduson keresztül tudják hogy melyik entitással kell dolgozniuk. Egy entitást implementáló manager-nek így értelemszerűen az entitás class-t adjuk vissza ebben a függvényben. Ez a felépítés kedvez a bővítésnek ugyanis gyorsan tudunk új entitásokat létrehozni már alap műveletekkel.

Egy példa a kódból:

```
@Stateless
@LocalBean
@Log
public class UserManager extends TemplateManager<User> {
```

```

        protected Class getMyClass() {
            return User.class;
        }
    }

```

A `@Log` egy saját annotáció. A logolás fejezetben lesz róla bővebb szó. A `@LocalBean`-al Bean-t csinálunk az osztályból. Ez azért előnyös mert így annak életciklusáról, inicializálásáról stb. nem nekünk kell gondoskodni, hanem a Java EE konténer kezelője elvégzi helyettünk. Bárhol ahol példányosítani akarjuk ezt az osztályt ott elég csak egy `@Inject` (vagy `@EJB`) annotációval helyi változóként felvenni. Pl.:

```

@Inject
private UserManager userManager;

```

És már használhatjuk is ..

```

user = userManager.select(id);

```

A `@Stateless` a bean élettartamát mondja meg. Jelen esetben csak a kérés erejéig szól az élettartama. (ez nem azt jelenti, hogy egyből törlődik, ezt majd a konténer kezelő üzemezi).

3.3.4.3 Jpql lekérdezések

A template manager által nyújtott alap lekérdezések többsége query stringet és egy paraméter map-et vár inputként. Az őt implementáló managgerek túlnyomó része ezeket hívja meg. Azaz a lekérdezéseknél jpql-t preferálok a Java Criteria Api-val szemben. Ennek oka az hogy a jpql-el könnyebb megfogalmazni bonyolultabb lekérdezéseket és annak olvashatósága is jobb. Performanciában nincs szignifikáns különbség a két módszer között. A Criteria Api előnye a jpql-el szemben a fordítás idejű típusbiztosság. Mindkét változat védelmet nyújt az sql injection ellen azáltal hogy paraméterezett a query. A kódban előfordul a „where '1' = '1'” feltétel hozzáfűzése a query stringhez. Ennek oka, hogy ha dinamikusan rakunk össze egy query-t akkor nem tudjuk előre, hogy szerepel-e már a „where” kulcsszó. A fenti módszer segítségével az erre vonatkozó ellenőrzéseket spóroljuk meg ami a kódot is olvashatóbbá teszi.

3.3.5 Üzleti logika

A komolyabb üzleti logikát a „Service” rétegek valósítják meg.

3.3.6 Ütemezők

Az alkalmazásban előfordulhatnak olyan műveletek melyeket a felhasználó, üzemeltető beavatkozása nélkül rendszerességgel szeretnénk elvégezni. Ezeket hívjuk ütemezett feladatoknak. Jelen alkalmazás logikában az ütemezett feladatokat a „CoreSchedulers” osztály tartalmazza.

```
@Singleton
@Startup
@Log
public class CoreSchedulers { ... }
```

@Singleton-al jelezzük, hogy ebből a bean-ból csak egy példányosodhat. A singletonok veszélye hogy, a műveleti végrehajtás szinkron, így ha egy művelet „megfogja” a bean-t (végrehajtási szálat) akkor a többi behívó félnek meg kell várnia amíg ők jutnak vezérléshez.

@Startup ezzel azt mondjuk meg, hogy a alkalmazás indulásakor (deploy)-kor ez a bean hivatkozás nélkül is inicializálódjon. Csak Singleton beanokat jelölhetünk @Startup-al.

Az az aukciók lezárása ha letelt a záró dátum egy ütemezett feladat melyet az alábbi függvény végez:

```
@Schedule(hour = "*", minute = "*", second = "0", persistent = false)
public void doAuctionEndCheck() { ... }
```

A @Schedule-val adjuk meg, hogy ez egy ütemezett feladat, melyet az annotáció paraméterében megadott értékek gyakoriságával hajtunk végre. Ez az ütemezett task nem persistent mivel „szolgáltatás kimaradása” esetén nem szükséges minden egyes kimaradt eseményre végrehajtnia, hogy ellássa a feladatát. A @Schedule bizonyos alkalmazáserver verziók alatt érzékeny lehet a „Runtime Exception”-ra (például hálózat leszakadásból eredő adatbázis elérési hibák) és hajlamos az ütemezést felfüggeszteni. Ha ilyennel találkozunk, akkor célszerű, hogy az annotációval ellátott függvény egyből áthívjon egy @Asynchronous annotációval ellátott függvénybe (ezáltal a dobódó exception nem gyűrűzik vissza az ütemezőig.)

3.3.7 Naplózás

Ahhoz hogy az alkalmazásunk működési folyamatai nyomon követhetők legyenek, illetve az esetlegesen felmerülő rendellenes működésnek az okát feltárhassuk megfelelő naplózásra van szükségünk.

3.3.7.1 Interceptoros naplózás

A függvény be és kilépésének naplózására kiválóan alkalmas az interceptoros naplózás.

Ezt egy saját annotációval valósítjuk meg:

```
@Inherited
@InterceptorBinding
@Target({TYPE, METHOD})
@Retention(RUNTIME)
public @interface Log {
    @Nonbinding
    LogLevel level() default LogLevel.INFO;
}
```

melyet hozzákötünk a lenti logolást végrehajtó „Interceptor”-hoz.

```
@Interceptor
@Log
@Priority(100)
public class LoggingInterceptor { ... }
```

Ezen elemek definiálása után elegendő a logolni kívánt bean-t vagy beanon belüli metódust (mivel csak bean-okra működik ez a fajta „Interceptor”) megannotálni egy @Log-al vagy annak paraméterezett változatával. Pl.: „@Log(level = LogLevel.DEBUG).

A „LoggingInterceptor” osztályon belül az egyetlen metódusunk az @AroundInvoke-val van megjelölve, ezzel jelezve, hogy ez a függvény fusson az annotált classokra. A metódus belsejében a @Log-ban megadott szint szerint (default INFO) szinten logojuk a be és ki lépést egy függvényből paraméterekkel együtt. Használat:

Az annotáció osztályra rakva annak minden publikus függvény hívásának ki és be lépését fogja logolni. Rakható csak method-ra is így csak az az egy függvény lesz naplózva. Az osztályra rakott log szintet felül lehet definiálni a pluszban metódusra rakott annotációval is. Pl.:

```
@Log
public class ItemManager extends TemplateManager<Item> { ... }
```

Azon belüli method pedig

```
@Log(level = LogLevel.TRACE)
public Item getItemById(Long itemId) {...}
```

Ezzel azt érjük el hogy a ItemManager-en belül minden public method INFO szinten lesz logolva kivéve a „getItemById” amelyik csak TRACE szinten jelenik meg.

3.3.7.2 Általános logolás

Ahol nem elég a ki és belépési pontot logolni azon osztályokban felveszünk egy loggert:

```
private static final Logger logger = Logger.getLogger(AuctionService.class);
```

A getLogger paramétere mindig az aktuális osztály.

Ezután a logolni kívánsz résznél elég így hivatkoznunk:

```
logger.info("Auction: " + auction.getId() + ... );
```


3.4 ARCHITEKTÚRA – WEB RÉTEG

A webes megjelenéshez JSF technológiát használunk. Az egyes oldalak „.xhtml” formátumúak így elkerülhetjük, hogy akaratlanul nyers jsf kód jelenjen meg a felületen. Illetve JSF 2.x óta is ez jobban támogatott. A felületi elemek könnyebb megjelenítéséhez Primefaces keretrendszert használunk melynek számtalan hasznos komponense van. (Timepicker, táblázatok, stb.)

3.4.1 Megjelenés (kliens)

A kliensnél megjelenő tartalmat a .xhtml végződésű fájlok tartalmazzák. A „commonLayout.xhtml”-t hozzáfűzzük minden más oldalhoz. Ez tartalmazza a menü elemeket és a lábléceket. Kódszervezésben törekedtünk az átláthatóságra így ez egyes mappák alá az adott témakörbe eső felületek kerülte.

[Shared]

commonLayout.xhtml - Közös elemek, menü és lábléc.

globalError.xhtml – Nem várt hiba esetén navigálunk ide

[Account]

login.xhtml – Bejelentkeztető felület

profile.xhtml – Profil oldal és annak fülei (későbbiekben megfontolandó modulokra bontani az átláthatóság érdekében.)

register.xhtml – Regisztrációs felület

viewNotification.xhtml – Értesítés megtekintése felület

[Auctions]

createAuction.xhtml – Aukció létrehozása felület

listAuctions.xhtml – Aukciók listázása, keresése felület

viewAuction.xhtml – Kiválasztott aukciót megjelenítő felület

[Admin]

statistics.xhtml – Rendszer statisztikák megtekintése.

usersList.xhtml – Felhasználók listázása, jogosultság kiosztás.

[Utils]

utilLinks.xhtml – Teszteléshez használatos elemek (nem a felhasználónak van szánva)

3.4.2 Vezérlők

Az előző részben említett.xhtml oldalak számos generálódó kódot tartalmaznak a.html-en felül. Ezeknek a generálásáért a vezérlő elemek felelősek. Jelen esetben „ManagedBean”-ek. Ezek tartják a kapcsolatot a BackEnd bean-jeivel is. Mivel a BackEnd nem tartalmaz remote beanokat így a frontend és a backend komponensei egy lokális gépen kell, hogy elhelyezkedjenek. A backend bean-eket itt is @Inject-el inicializálhatjuk és használhatjuk. A ManagedBean-ek többsége @RequestScoped / @ViewScoped azaz csak egy kérés idejéig őrzik meg változóiknak a tartalmát.

3.4.3 Hibakezelés

A ManagedBeanok függvény hívásaiban elkapjuk a vár kivételeket, de óhatatlanul is előfordul nagy rendszerekben, hogy nem vár kivétel dobódik. Annak elkerülésére hogy a felhasználó egy „Stacktrace”-t, response error-t vagy egy félig – hibásan megjelenített oldalt lásson konfiguráltunk egy globális kivétel kezelőt (faces-config.xml). „CustomExceptionHandlerFactory”. Ez elkapja az el nem kapott kivételeket a felületen majd naplózza őket s a felhasználót átirányítja a „globalError.xhtml” oldalra ahol értesül róla hogy valami hiba történt, és ha ez rendszeres akkor tájékoztassa az üzemeltetőket.

Ezen kezelőt lehet igény szerint speciálisabbra bővíteni, például hogy ha lejárt a „session” akkor azt a kivételt másképp kezelje.

3.5 TESZTELÉS

3.5.1 Automatikus tesztek

Automatikus tesztesetekre olyan JUnit teszteket „írtunk” melyek egy átlag felhasználható által nem feltétlenül reprodukálható, vagy az eredmény nem vagy nehezen észrevehető. (Elemek aszinkron módosítása, kulcsütközések tesztelése stb.)

3.5.2 Manuális tesztesetek

A manuális tesztesetek alatt főként olyan eseteket tárgyalunk, amik a felületet használva átlag felhasználóként is reprodukálható.

3.5.2.1 Regisztráció – Bejelentkezés

- Regisztrálni próbálunk, de nem töltünk ki minden mezőt
 - A ki nem töltött mező mellett pirosan üzenettel jelöljük, hogy a mező kötelező.
- Bejelentkezés nem létező felhasználóval, vagy hibás jelszóval - felhasználónévvel.
 - a „Login Failed, Invalid name or password” hibaüzenetet kapjuk, nem történik bejelentkeztetés, maradunk a bejelentkező felületen.
- A regisztrációkor a két jelszó nem egyezik.
 - „Passwords does not match” hibaüzenettel jelezzük.
- A felhasználónévvel már korábban mi vagy egy másik személy regisztrált.
 - „Username is already used” üzenettel közöljük, hogy a felhasználónév már foglalt.
- Egy még nem létező felhasználót regisztrálunk úgy, hogy a két jelszó megegyezik.
 - Átnavigálunk a bejelentkező felületre, és üzenetben értesülünk a regisztráció sikerességéről
- Az imént regisztrált felhasználóval, helyes jelszóval a „Login”-re kattintunk.
 - A rendszer bejelentkeztet minket, és a profil oldalra navigálunk.

3.5.2.2 Tárgy feltöltése - Törlése

- A tárgy hozzáadása felületen kép feltöltése gombra kattintunk azt megelőzően, hogy hozzáadtunk volna egy képet.
 - Értesítést kapunk, hogy válasszunk ki egy fájlt a gombra kattintás előtt
- Üresen hagytuk a tárgy neve mezőt, és a hozzáadásra kattintunk.
 - Piros hibaüzenettel jelezzük a mező mellett, hogy ez kitöltendő
- Kitöltöttük a név mezőt, képet tetszőlegesen adtunk meg vagy nem.
 - Visszanavigálunk a profil oldalra és jelezzük, hogy a tárgyat sikeresen hozzáadtuk.
- Kiválasztunk egy olyan tárgyat mely már aukcióhoz van kötve és törölni próbáljuk.
 - Értesítést kapunk, hogy csak olyan elemek törölhetők melyek nincsenek aukcióhoz rendelve.

3.5.2.3 Aukció létrehozása

- Valamelyik mezőt üresen hagyjuk az alábbi 3 közül („Auction header”, „starting price”, „Expire Time”).
 - Az üresen hagyott mező mellett megjelenik, hogy kötelező kitölteni.
- A beviteli mezőknek nem megfelelő értéket adunk meg. A kezdő árra szám helyett szöveget, vagy negatív értéket.
 - A hibásan kitöltött mező mellett megjelenik az elvárt formátum.
- Egy már aukcióhoz rendelt tárgyat próbálunk ismételt aukcióhoz rendelni.
 - Értesítést kapunk, hogy a tárgynak már van aukciója.

3.5.2.4 Licitálási folyamatok

- Saját aukcióra próbálunk licitálni.
 - Értesítést kapunk, hogy saját aukcióra nem licitálhatunk.
- A minimum árnál vagy ha van jelenlegi licit és annál kisebbet akarunk licitálni.

- Értesítést kapunk, hogy a minimum licitet nem haladtuk meg.
- Üresen hagyott vagy nem számmal kitöltött licitálási mező.
 - Értesülünk, hogy a beviteli mező nem helyesen van kitöltve.
- Lejárt, lezárt aukcióra próbálunk licitálni.
 - Értesülünk, hogy az aukciót már lezárták.

3.5.2.5 Felhasználók értékelése

- Az adott felhasználót már értékeltük korábban (ezen az aukción keresztül).
 - Értesítést kapunk, hogy ezt az „aukciót” már értékeltük
- Nem adunk meg értékelést és, vagy szöveges üzenetet.
 - Megrázkódik az ablak és értesítést kapunk, hogy a mezők kitöltése kötelező.

3.6 TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

3.6.1 Konfiguráció

Fontos hogy egy alkalmazás általános is legyen, de egyben, ha az ügyfél úgy kívánja személyre szabható is legyen. Erre jó módszer a fájlból felolvasott konfigurációk alapján való működés. Jelenleg ez egyáltalán nincs megvalósítva. Ilyen lehetne például a lokalizáció, alap adatok betöltése (pl.: nem létezik admin user akkor indításkor készítsünk-e). Egy-egy ügyfélnél eltérő funkciók konfigurációból történő ki-be kapcsolása.

3.6.2 Értesítések

Az eredeti koncepcióban az értesítéseket küldő szolgáltatás egy külön alkalmazás. Jelenleg ez a mostani alkalmazásunkba van integrálva és az értesítés küldése csak a felhasználói fiókra küld egy üzenetet. Ezt lehetne bővíteni egy tényleges külső alkalmazással mely, ha értesítést kap a rendszertől akkor a megadott email címre is elküldené ugyanazt az üzenetet amit a felhasználó kapott a saját fiókjába a rendszeren belül

3.6.3 Back End validációk

A kód jelenleg támaszkodik a felhasználók jóindulatára és elég sok ellenőrzést elhagy, mely átlag felhasználás mellett nem szükséges de „package” , url, manipuláció esetén problémát okozhat. Ilyen jellegűek ellenőrzések hogy az adott elem tényleg a felhasználóhoz tartozik-e, van-e jogosultsága azt módosítani, stb.

3.6.4 Extrém inputok

Alap ellenőrzések vannak arra, hogy egy szöveges mező tartalma csak szöveg lehet vagy szám, de azt a legtöbb rendszer arra nem figyel külön, hogy ezek az értékek ne haladják meg az esetleges adatbázis mezők tárolási értékét. Például megadok egy 2500 karakter hosszú felhasználónevet vélelmezhetően a hiba csak az adatbázis rétegnél bukik ki.

3.6.5 Dao réteg bevezetése

Nem szerencsés ha a fornted is az Entitásokkal dolgozik ugyanis számtalan felesleges adatot tartalmazhat egy entitás aminek a felületi megjelenítéshez semmi köze. Erre szokták használni a DAO objektumokat amely csak a szükséges mezőket tartalmazza.

3.6.6 ID-k kiváltása

Jelenleg az fontend-en megadott azonosítók tényleges adatbázis béli id-k mindenféle módosítás nélkül. Ez biztonsági hibákat eredményezhet. Helyettük maskolt azonosítókat lehetne használni.

3.6.7 Felület

A felület jelenleg nem túl felhasználó barát, sokkal inkább azt a célt szolgálja hogy megmutassa mit tud a „core” funkcionalitás. Az egyes oldalak közötti navigáláson sokat lehetne optimalizálni.

4 IRODALOM

- [1] A projekt elérhetősége: <https://github.com/deverxxx/Espace> [2017. május 14.]
- [2] Derby adatbázis: <https://db.apache.org/derby/> [2017. május 14.]
- [3] Glassfish alkalmazáserver: <https://glassfish.java.net/> [2017. május 14.]
- [4] Maven: <https://maven.apache.org/> [2017. május 14.]
- [5] Java 8: <http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html> [2017. május 14.]
- [6] Java EE7: <https://docs.oracle.com/javaee/7/> [2017. május 14.]
- [7] Primefaces: <https://www.primefaces.org/> [2017. május 14.]
- [8] Bootstrap: <http://getbootstrap.com/> [2017. május 14.]
- [9] Log4j: <https://logging.apache.org/log4j/1.2/> [2017. május 14.]
- [10] Stack Overflow: <http://stackoverflow.com/> [2017. május 14.]
- [11] Fejlesztőkörnyezet, IntelliJ: <https://www.jetbrains.com/idea/> [2017. május 14.]
- [12] EclipseLink <http://www.eclipse.org/eclipselink/> [2017. május 14.]

5 MELLÉKLETEK

5.1 ADATBÁZIS TÁBLA LÉTREHOZÓ SCRIPTEK

```
CREATE TABLE USERTABLE
```

```
(
    ID BIGINT PRIMARY KEY NOT NULL,
    CREATEDON TIMESTAMP,
    DATEOFBIRTH TIMESTAMP,
    DELETED SMALLINT DEFAULT 0,
    EMAILADDRESS VARCHAR(255),
    PASSWORD VARCHAR(255),
    PHONENUMBER VARCHAR(255),
    PICTURE VARCHAR(255),
    REALNAME VARCHAR(255),
    USERNAME VARCHAR(255)
)
```

```
CREATE TABLE ROLETABLE
```

```
(
    ID BIGINT PRIMARY KEY NOT NULL,
    CREATEDON TIMESTAMP,
    DELETED SMALLINT DEFAULT 0,
    GROUPROLE VARCHAR(255),
    USERNAME VARCHAR(255)
)
```

```
CREATE TABLE ITEMCATEGORY
```

```
(
    ID BIGINT PRIMARY KEY NOT NULL,
    CREATEDON TIMESTAMP,
    DELETED SMALLINT DEFAULT 0,
    CATEGORYNAME VARCHAR(255)
)
```

```
CREATE TABLE ITEM
```

```
(
    ID BIGINT PRIMARY KEY NOT NULL,
    CREATEDON TIMESTAMP,
    DELETED SMALLINT DEFAULT 0,
    NAME VARCHAR(255),
    PICTURE VARCHAR(255),
    CATEGORY_ID BIGINT,
    USER_ID BIGINT,
    AUCTION_ID BIGINT,
    CONSTRAINT FK_ITEM_AUCTION_ID FOREIGN KEY (AUCTION_ID) REFERENCES
AUCTION (ID),
    CONSTRAINT ITEM_CATEGORY_ID FOREIGN KEY (CATEGORY_ID) REFERENCES
ITEMCATEGORY (ID),
    CONSTRAINT FK_ITEM_USER_ID FOREIGN KEY (USER_ID) REFERENCES
USERTABLE (ID)
)
```



```

CREATE TABLE AUCTION
(
    ID BIGINT PRIMARY KEY NOT NULL,
    CLOSED SMALLINT DEFAULT 0,
    CREATEDON TIMESTAMP,
    DELETED SMALLINT DEFAULT 0,
    DESCRIPTION VARCHAR(255),
    EXPIRATIONDATE TIMESTAMP,
    HEADER VARCHAR(255),
    MINIMUMBID FLOAT(52),
    STARTDATE TIMESTAMP,
    OWNER_ID BIGINT,
    ITEM_ID BIGINT,
    TOPBIDER_ID BIGINT,
    USERRATING_ID BIGINT,
    CONSTRAINT AUCTIONTOPBIDER_ID FOREIGN KEY (TOPBIDER_ID) REFERENCES
BID (ID),
    CONSTRAINT FK_AUCTION_ITEM_ID FOREIGN KEY (ITEM_ID) REFERENCES
ITEM (ID),
    CONSTRAINT UCTIONUSERRATINGID FOREIGN KEY (USERRATING_ID)
REFERENCES USERRATING (ID),
    CONSTRAINT AUCTION_OWNER_ID FOREIGN KEY (OWNER_ID) REFERENCES
USERTABLE (ID)
)

```

```

CREATE TABLE BID
(
    ID BIGINT PRIMARY KEY NOT NULL,
    BID FLOAT(52),
    CREATEDON TIMESTAMP,
    DELETED SMALLINT DEFAULT 0,
    AUCTION_ID BIGINT,
    USER_ID BIGINT,
    CONSTRAINT FK_BID_AUCTION_ID FOREIGN KEY (AUCTION_ID) REFERENCES
AUCTION (ID),
    CONSTRAINT FK_BID_USER_ID FOREIGN KEY (USER_ID) REFERENCES
USERTABLE (ID)
)

```

```

CREATE TABLE NOTIFICATION
(
    ID BIGINT PRIMARY KEY NOT NULL,
    CONTENT VARCHAR(255),
    CREATEDON TIMESTAMP,
    DELETED SMALLINT DEFAULT 0,
    READED SMALLINT DEFAULT 0,
    SENDER VARCHAR(255),
    TITLE VARCHAR(255),
    RECIPIENT_ID BIGINT,
    CONSTRAINT NTFCATIONRCPIENTID FOREIGN KEY (RECIPIENT_ID)
REFERENCES USERTABLE (ID)
)

```

```

CREATE TABLE USERRATING
(
    ID BIGINT PRIMARY KEY NOT NULL,
    CREATEDON TIMESTAMP,
    DELETED SMALLINT DEFAULT 0,
    RATING SMALLINT,
    TEXT VARCHAR(255),
    RECIPIENT_ID BIGINT,
    SENDER_ID BIGINT,
    AUCTION_ID BIGINT,
    CONSTRAINT SERRATINGAUCTIONID FOREIGN KEY (AUCTION_ID) REFERENCES
AUCTION (ID),
    CONSTRAINT SRRATINGRCIPIENTID FOREIGN KEY (RECIPIENT_ID)
REFERENCES USERTABLE (ID),
    CONSTRAINT USERRATINGSENDERID FOREIGN KEY (SENDER_ID) REFERENCES
USERTABLE (ID)
)

```