Online állatkölcsönző

Rendszerleírás

Szoftverarchitektúrák (VIAUMA06) házi feladat

Készítette:

Pollák Tamás (BABQ9E) és Kincses Gergő (A7SOEA)

2015. november. 24

**Tartalom**

[A rendszer specifikációja 3](#_Toc436237146)

[Feladatkiírás 3](#_Toc436237147)

[Az alkalmazás által biztosított funkciók 3](#_Toc436237148)

[Állat attribútumai 3](#_Toc436237149)

[Az alkalmazás felülete 4](#_Toc436237150)

[Ügyfél funkciók 4](#_Toc436237151)

[Adminisztrátor funkciók 5](#_Toc436237152)

[Az alkalmazás környezete 6](#_Toc436237153)

[Megvalósítás és architektúra 7](#_Toc436237154)

[Adatbázis 7](#_Toc436237155)

[Adatbázisterv 7](#_Toc436237156)

[Backend 9](#_Toc436237157)

[Adatbáziskezelő metódusok 10](#_Toc436237158)

[REST service metódusok 12](#_Toc436237159)

[A metódusok: 12](#_Toc436237160)

[Frontend 12](#_Toc436237161)

[A GUI főbb elemei: 12](#_Toc436237162)

[Az oldal működése 13](#_Toc436237163)

[Regisztráció és bejelentkezés 13](#_Toc436237164)

[Menedzsment 14](#_Toc436237165)

[Böngészés és kölcsönzés 15](#_Toc436237166)

[Telepítési leírás 16](#_Toc436237167)

[Továbbfejlesztési lehetőségek 17](#_Toc436237168)

[Hivatkozások 18](#_Toc436237169)

# A rendszer specifikációja

## Feladatkiírás

Ki ne akart volna már néhány napra szórakozásból hazavinni egy édes kis nyuszikát, hörcsögöt, vagy éppen tyúkot, tehenet? Most valóra válthatják az álmukat! A feladat egy webes rendszer elkészítése, mely lehetőséget kínál állatok regisztrálására és követi a foglalásaikat, kölcsönzésüket.

## Az alkalmazás által biztosított funkciók

A feladat megoldása során a cél egy olyan webes alkalmazás létrehozása, melyen keresztül lehetőség nyílik különböző állatok kikölcsönzésére. Az alkalmazás biztosítja az állatok közti böngészés lehetőségét, ill. a kikölcsönzés utáni visszaadásukat. Az állatok adatait egy adminisztrátor tölti fel az adatbázisba, és ő is tartja karban őket. Egy állatot tetszőleges időtartamra ki lehet kölcsönözni.

## Állat attribútumai

Egy állatot a következő tulajdonságai jellemeznek:

* kép – egy kis kép az állatról
* név – az állat neve
* faj – az állat faja
* fajta – az állat fajtája
* nem – az állat neme
* kor – az állat életkora
* szín – az állat színe
* térfogat – az állat térbeli kiterjedése
* súly – az állat súlya
* kedvenc étel – az állat kedvenc étele
* hobbi – ezt szereti az állat általában csinálni
* elérhető-e – az állat jelenleg ki van-e kölcsönözve
* törött-e – az állat egészségi vagy mentális állapota megengedi-e, hogy kikölcsönözzék

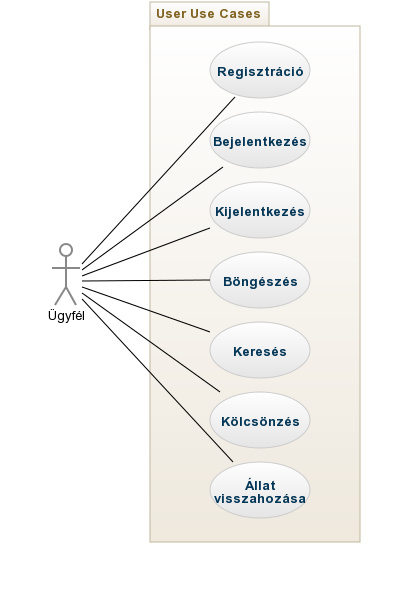
## Az alkalmazás felülete

Az alkalmazásnak alapvetően három féle felülete van:

* Bejelentkező felület: Itt tudnak az ügyfelek és az adminisztrátorok bejelentkezni az alkalmazásba, valamint itt tudnak az ügyfelek regisztrálni. A rendszerből történő kijelentkezés után ide kerülnek a felhasználók.
* Lista felület: Itt tudnak az ügyfelek böngészni és keresni az állatok között. A felületen egymás alatt jelennek meg az egyes állatok minden attribútumukkal.
* Menedzsment felület: Itt tud az adminisztrátor új állatot bevinni a rendszerbe, állatot eltávolítani a rendszerből, ill. módosítani a rendszerben lévő állatok egyes attribútumain.

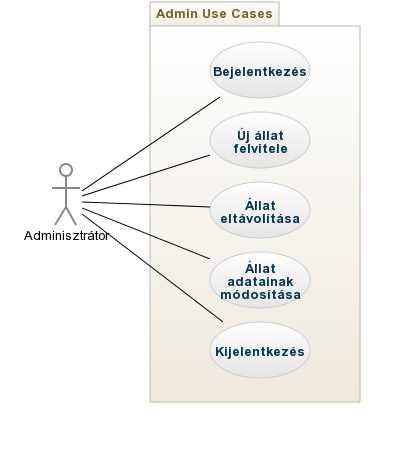
## Ügyfél funkciók

* Az ügyfél a bejelentkező felületen be tud regisztrálni az alkalmazásba.
* Az ügyfél a bejelentkező felületen be tud jelentkezni az alkalmazásba.
* A bejelentkezett ügyfél a lista felületen tud böngészni az állatok között.
* A bejelentkezett ügyfél a lista felületen tud keresni közöttük a fajuk, fajtájuk, elérhetőségük és épségük alapján.
* A bejelentkezett ügyfél a lista felületen ki tud kölcsönözni tetszőleges számú állatot.
* A bejelentkezett ügyfél a lista felületen vissza tud hozni tetszőleges számú kikölcsönzött állatot, amennyiben azok a felhasználónál vannak.
* A bejelentkezett ügyfél a lista felületen ki tud jelentkezni az alkalmazásból.



***1. ábra*** *Ügyfél use case-ek*

## Adminisztrátor funkciók

* Az adminisztrátor a bejelentkező felületen be tud jelentkezni az alkalmazásba, egy előre kialakított profilon keresztül.
* Az adminisztrátor a menedzsment felületen tud új állatot vinni az alkalmazásba.
* Az adminisztrátor a menedzsment felületen tud az alkalmazásból eltávolítani állatokat.
* Az adminisztrátor a menedzsment felületen tud módosítani az alkalmazásban lévő állatok attribútumainak értékein.

***2. ábra*** *Adminisztrátor use case-ek*

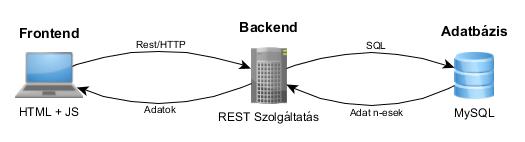
* Az adminisztrátor a menedzsment felületen tud kijelentkezni az alkalmazásból.

## Az alkalmazás környezete

Az alkalmazás egy kliensből és egy szerverből áll. A szervert Java nyelven készítjük el, a kliens HTML felhasználásával áll elő. A kliens REST interfészen éri el a szervert, és a szerver ugyanezen keresztül küld adatot a kliensnek. A szerver egy MySQL adatbázist használ az adatok tárolására. A program további magától nem értetődő követelményeket nem támaszt a futtató számítógép felé.

# Megvalósítás és architektúra

Az alkalmazást három komponens (réteg) segítségével valósítottuk meg. Elkülönítettünk egy klienst, egy szervert és egy adatbázist. A komponensek és azok kommunikációja a következő ábrán látható.



***3. ábra*** *Architektúra*

## **Adatbázis**

**Feladata:** az adatbázis réteg felel az adatok perzisztálásáért.

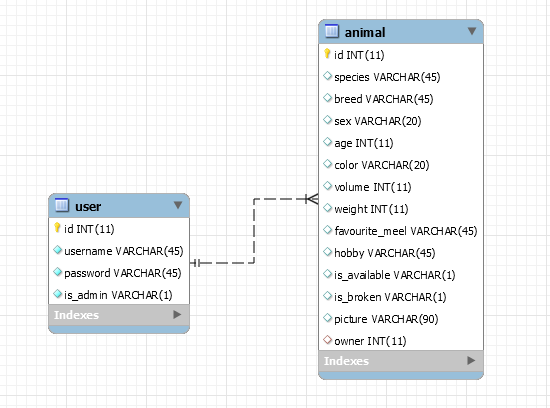
Az adatbázisban tároljuk a weboldalon kikölcsönözhető állatok adatait, a regisztrált felhasználók adatait és az adminisztrátorok adatait.

Az adatbázist a MySQL adatbáziskezelő rendszer segítségével valósítottuk meg a következők miatt:

* az egyik legelterjedtebb adatbáziskezelő rendszer napjainkban
* számos hasznos dokumentáció és fórum foglalkozik a MySQL adatbázisok létrehozásával, használatával, menedzselésével, ami nagyban megkönnyíti a fejlesztést és a munka során felmerülő problémák megoldását
* ingyenesen elérhető a rendszer
* könnyű a MySQL adatbázisok kezelése Java platformon

### **Adatbázisterv**

Az adatbázisban két fő tábla van, melyek tartalmazzák a felhasználók (ill. adminisztrátorok) és az állatok adatait.



*4. ábra Adatbázis tábláinak kapcsolata*

#### **User tábla**

A *User* táblában tároljuk a felhasználók és az adminisztrátorok adatait. Egy *User* elem négy attribútumot tartalmaz:

* azonosító (**id**)
* felhasználónév (**username**)
* jelszó (**password**)
* **is\_admin**: megmondja, hogy az adott felhasználó adminisztrátor-e vagy sem

A *User* entitás 1:N kapcsolatban áll az *Animal* entitással, ami azt jelenti, hogy egy *User*-hez tartozhat több állat, de egy állatnak csak egy kölcsönzője lehet.

#### **Animal tábla**

Az *Animal* táblában tároljuk az állatokhoz tartozó összes adatot. Egy állat tizennégy attribútummal rendelkezik:

* azonosító (**id**)
* faj (**species**)
* fajta (**breed**)
* nem (**sex**)
* életkor (**age**)
* szín (**color**)
* térfogat (**volume**)
* súly (**weight**)
* kedvenc étel (**favourite\_meal**)
* hobbi (**hobby**)
* elérhető-e (**is\_available**)
* törött-e (**is\_broken**)
* kép (**image**)
* tulajdonos (**owner**): idegen kulcs a *User* tábla *id* mezőjére, ezzel az attribútummal azonosítjuk a kölcsönzőt

Tervezői döntés: az állatokról a hozzájuk tartozó kép nevét tároljuk kiterjesztéssel együtt, ennek a segítségével azonosítja a megjelentő réteg a betöltendő képet.

## **Backend**

**Feladata:**

* kommunikáció az adatbázissal
  + adatbázisban lévő adatok módosítása
  + adatok lekérdezése az adatbázisból
* a Frontend réteg felől érkező kérések fogadása és kiszolgálása

A Backend-et funkcionalitását így két részre tudjuk elkülöníteni: adatbáziskezelő metódusokra, és Rest service metódusokra. A Rest metódusok használják az adatbáziskezelő metódusokat.

A programban külön package-ekben helyezkednek az említett metódusok.

A backendet egy REST szolgáltatás valósítja meg, amely egy WildFly szerveren [1] fut.

### **Adatbáziskezelő metódusok**

Az adatbáziskezelő metódusok a következők:

* insertUser – beilleszt egy felhasználót az adatbázisba
* login – visszaad egy felhasználót az adatbázisból a neve és a jelszava alapján
* checkUserName – ellenőrzi, hogy az adott felhasználónév létezik-e az adatbázisban
* listAnimals – visszaadja az adatbázisban lévő összes állatot
* searchAnimals – a listázott állatokat szűri bizonyos feltételek alapján
* rentAnimal – az adatbázisban egy állat is\_available mezőjét „N”-ra állítja és a paraméterben kapott információk alapján tulajdonost rendel az állathoz
* bringBackAnimal - az adatbázisban egy állat is\_available mezőjét „Y”-re állítja
* insertAnimal – felvesz egy új állatot az adatbázisba
* updateAnimal – egy az adatbázisban már létező állat adatait módosítja
* deleteAnimal – eltávolít az adatbázisból egy állatot

Az adatbázis kezeléséhez a Java Persistence API-t (JPA) használtunk. A JPA object relational mapping (ORM) [2] segítségével dolgozik az adatbázison. A módszer segítségével a kódban lévő adatstruktúrákat (praktikusan osztályokat) egy relációs adatbázis-tábla elemeinek (soroknak) lehet megfeleltetni. Ezeknek az osztályoknak a példányain végzett műveletek pedig perzisztálódnak az adatbázisban. Ennek a technológiának a segítségével akár nincs is szükség SQL lekérdezések írására, közvetlenül objektumokon dolgozhatunk.

Az általunk használt JPA provider az EclipseLink.

#### ***User entitás***

**Célja:** egy felhasználó adatainak reprezentálása

**Leképezése a kódban:** a *User* osztály

**Leképezése az adatbázisban:** a *user* tábla 1-1 sorára

**Tulajdonságai**

|  |  |  |
| --- | --- | --- |
| **Java mezőnév** | **Java adattípus** | **MySQL adattípus** |
| id | int | INT – PRIMARY KEY |
| username | String | VARCHAR |
| password | String | VARCHAR |
| is\_admin | String | VARCHAR |

#### ***Animal entitás***

**Célja:** egy állat adatainak reprezentálása

**Leképezése a kódban:** az *Animal* osztály

**Leképezése az adatbázisban:** az *animal* tábla 1-1 sorára

**Tulajdonságai**

|  |  |  |
| --- | --- | --- |
| **Java mezőnév** | **Java adattípus** | **MySQL adattípus** |
| id | int | INT – PRIMARY KEY |
| species | String | VARCHAR |
| breed | String | VARCHAR |
| sex | String | VARCHAR |
| age | Integer | INT |
| color | String | VARCHAR |
| volume | Integer | INT |
| weight | Integer | INT |
| favourite\_meal | String | VARCHAR |
| hobby | String | VARCHAR |
| is\_available | String | VARCHAR |
| is\_broken | String | VARCHAR |
| picture | String | VARCHAR |
| owner | Integer | INT – FOREIGN KEY |

### **REST service metódusok**

A Frontend-el való kommunikációhoz a REST architektúrát használjuk. A REST egy webes szabványokon alapuló architektúra, ami a HTTP protokollt használja az adatkommunikációra.

A REST használta során a szerver nyújt hozzáférést az erőforrásokhoz a kliensnek. Az erőforrásokat URI-k segítségével azonosítjuk.

A működése során az egyes *service* metódusai JSON objektumokat küldenek és fogadnak, ill. használják az adatbázis komponensben definiált eljárásokat.

### A metódusok:

* getUser – ellenőrzi, hogy egy adott felhasználó benne van-e az adatbázisban
* regUser – regisztrál egy adott felhasználót, amennyiben még a név még szabad
* getAnimals – kilistázza az adatbázisban lévő állatokat
* rent – egy adott felhasználóhoz rendel egy adott állatot
* bringBackAnimal – egy állatot „elválaszt” a bérlőtől és „visszahelyez” a készletbe
* searchAnimal – szűkíti az állatok listáját bizonyos tulajdonságok alapján
* insertAnimal – beilleszt egy új állatot az adatbázisba
* deleteAnimal – töröl egy állatot az adatbázisból
* updateAnimal – módosítja egy már felvett állat adatait

## **Frontend**

Az alkalmazás böngészőben történő megjelenítéséhez HTML oldalakat, Javascriptet és CSS-t használunk. Az oldal designja a Bootstrap keretrendszeren alapul. Az oldalon struktúrájának dinamikus változtatását az Angular JS keretrendszerrel oldottuk meg.

### **A GUI főbb elemei:**

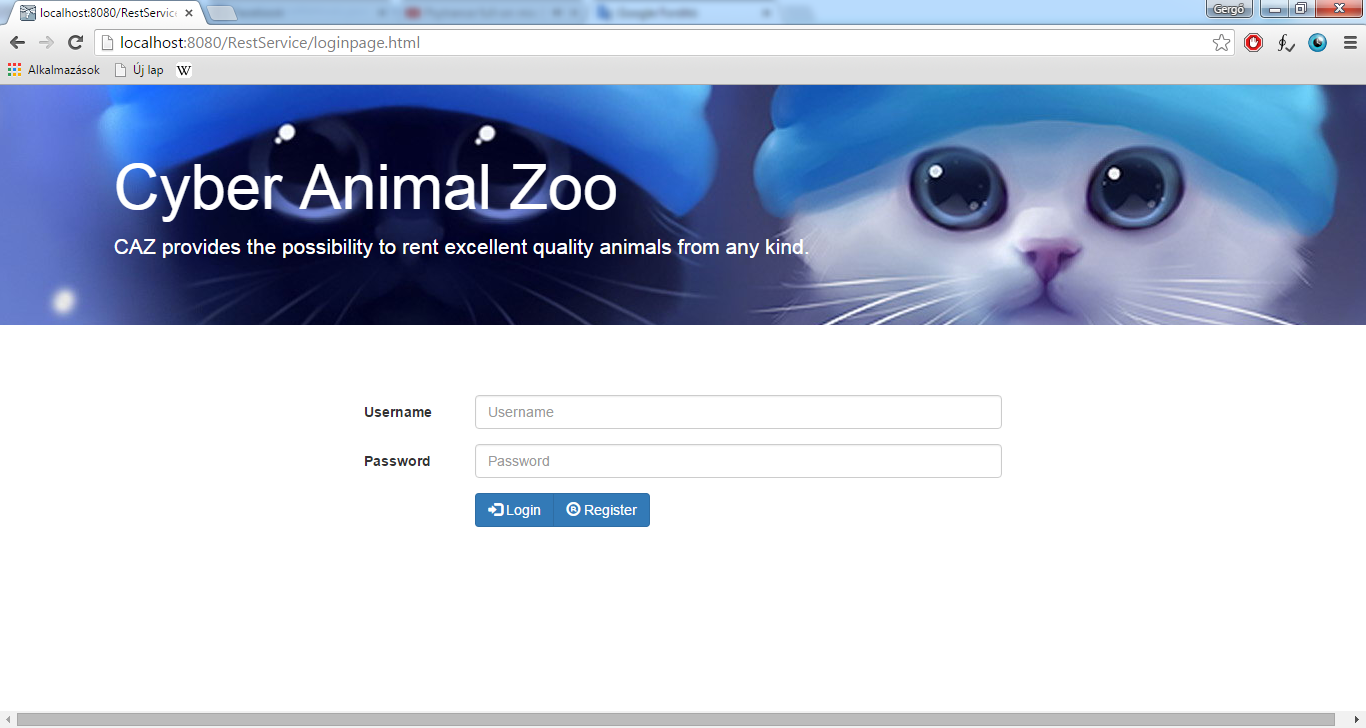
* Bejelentkező oldal – ezen keresztül lehet bejelentkezni, ill. regisztrálni
* Management oldal – az adminisztrátorok ezen keresztül menedzselhetik az alkalmazást
* Lista oldal – ez a fő felület, a felhasználók itt válogathatnak és kölcsönözhetnek az állatok közül

A fenti három GUI elemet (a designtól eltekintve) a következő egységek valósítják meg:

* Bejelentkező oldal – loginpage.html, loginpage.js
* Management oldal – managementpage.html, managementpage.js
* Lista oldal – listpage.html, listpage.js

# Az oldal működése

## Regisztráció és bejelentkezés



*5. ábra Bejelentkező oldal*

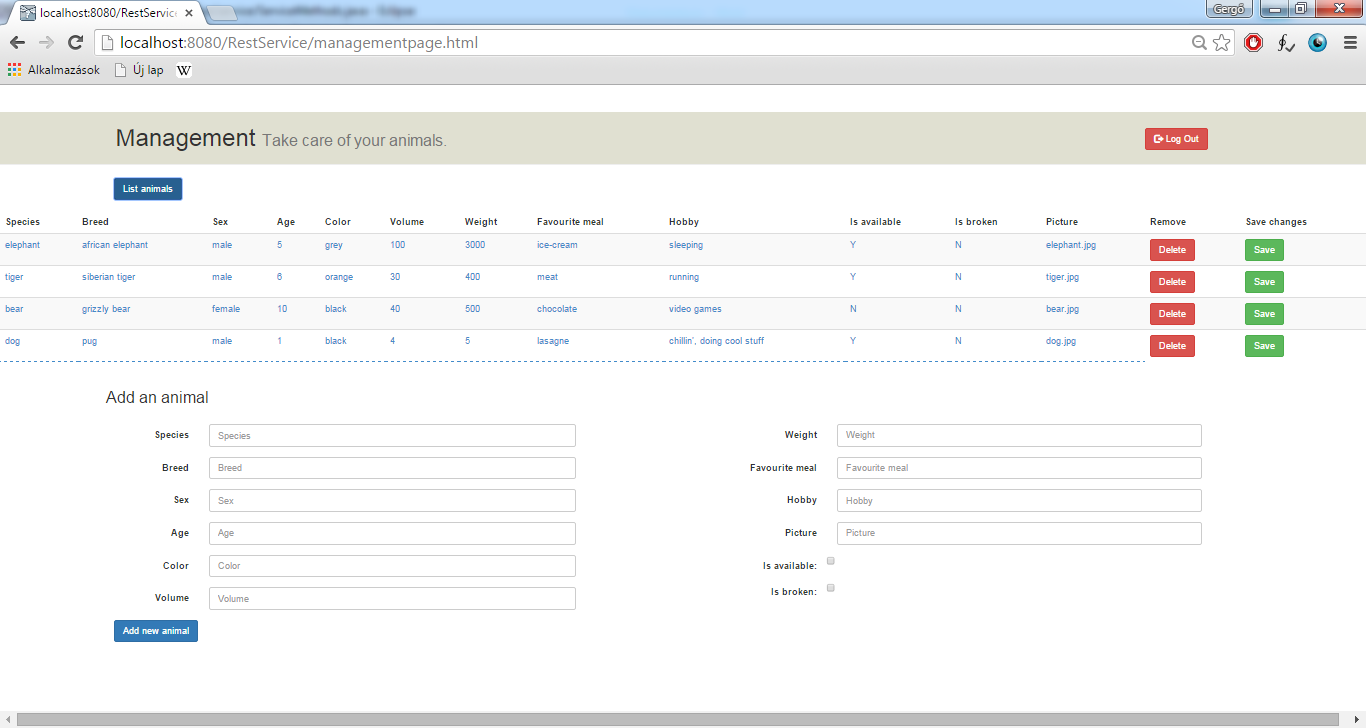
A felhasználók az ábrán látható felületen regisztrálhatnak a kölcsönző online felületére. Amennyiben már van regisztrációjuk, itt tudnak bejelentkezni. A kölcsönző adminisztrátorai is itt tudnak belépni a rendszerbe.

Amennyiben hibás adatokkal próbál valaki belépni, a rendszer hibaüzenetet küld. A rendszerben a felhasználónevek egyediek, így két user nem tud ugyanazon felhasználónévvel regisztrálni.

Az oldal hibaüzenettel figyelmeztet, ha nem töltjük ki valamelyik beviteli mezőt, és így próbálunk belépni, vagy regisztrálni.

Sikeres bejelentkezés esetén a lista oldalra irányítja a rendszer a felhasználót, amennyiben nem adminisztrátor. Az adminisztrátorokat a menedzsment felületre irányítja.

## **Menedzsment**



*6. ábra Menedzsment felület*

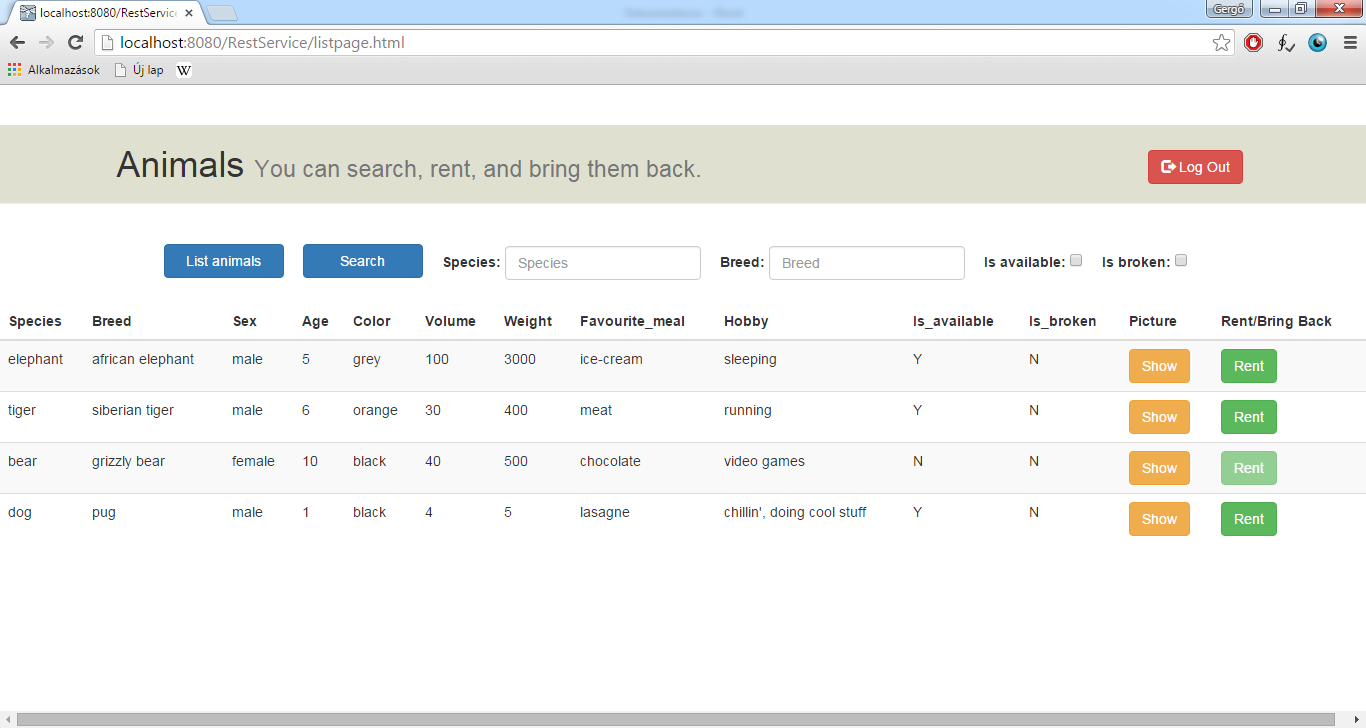
A kölcsönző adminisztrátorai ezen a felületen tudják a rendszerben lévő állatok adatait menedzselni, ill. új állatot felvinni a rendszerbe. A felső táblázat lehetőséget nyújt az egyes állatok adatainak mezőnkénti módosítására. A sor szélén található *„Save”* gombbal tudjuk érvényre juttatni a sorban véghezvitt változtatásokat.

A sorban található „*Delete”* gombok szolgálnak egy állat törlésére az adatbázisból.

Az alul látható beviteli mezők pedig egy új kisállat bevitelére szolgálnak a kölcsönző adatbázisába. Az összes mezőt ki kell tölteni, hogy fel tudjunk vinni egy új állatot. A mezők típusai ellenőrzöttek, amelyik mezőbe numerikus adatokat várunk, oda nem adhatunk meg szöveget.

A sikeres módosításokról képernyőn megjelenő üzenetek formájában értesíti a rendszer a felhasználót.

## **Böngészés és kölcsönzés**



*7. ábra Állatok listája*

A felhasználó bejelentkezés után tud böngészni az állatok között egy lista nézetben. Ebben a listában tud szűrni specifikációban rögzített négy attribútum alapján.

Az állatokról elérhetők a képeik, melyek a *„Show”* gomb megnyomása után jelennek meg az adatokat tartalmazó táblázat alatt.

Ha megtalálta az igényeinek megfelelő állatot (vagy állatokat), akkor a *”Rent”* gomb megnyomásával kikölcsönözheti. Ha a felhasználó vissza kívánja hozni az állatot, akkor ezt a „Rent” gomb helyén megjelenő *„Bring back”* gomb használatával teheti meg.

Azon állatok esetén, akiket valamilyen okból nem lehet kikölcsönözni a Rent gomb inaktív, azaz nem lehet rákattintani.

Amennyiben egy felhasználó olyan állatot próbál kikölcsönözni, amelyet egy másik felhasználó az után kölcsönzött ki, miután az első már kilistázta az állatokat, akkor az első felhasználót informálja az oldal, hogy már nem elérhető az adott állat, és frissíti az állatok listáját.

# Telepítési leírás

TODO

# Továbbfejlesztési lehetőségek

Az alkalmazással kapcsolatban többféle továbbfejlesztési lehetőséget látunk, ezek közül néhány:

* a weboldal design-jának teljesen reszponzívvá tétele
* asztali menedzsment kliens készítése a webes mellé / helyett
* lehetőség az oldal bizonyos tartalmainak megosztására közösségi portálokon
* pl.: állatok kedvelése, állatok megosztása
* „feliratkozni” nem elérhető állatokra e-mail címmel: a felhasználó értesítést kapni a címére, amikor az általa kiválasztott állat szabaddá válik

A rendszer jelenlegi architektúrája úgy gondoljuk eléggé flexibilis, hogy bármelyik továbbfejlesztésbe bele lehessen kezdeni nagyobb méretű architektúra módosítás nélkül.

# Hivatkozások

[1] - <http://wildfly.org/>

[2] - <https://en.wikipedia.org/wiki/Object-relational_mapping>