

JEGYZŐKÖNYV

Adatbázisrendszerek I.

Féléves feladat: Állatkerthálózat

Készítette: Martinák Mátyás
Neptunkód: KLNSPG
Gyakorlat: Kedd 10-12
Gyakorlatvezető: Dr. Bednarik László

Miskolc, 2022

Tartalomjegyzék

1. A feladat leírása	2
2. Az adatbázis ER modellje	3
3. Az adatbázis konvertálása relációs modellre	4
4. Az adatbázis relációs modellje	5
5. Az adatbázis relációs sémája	5
6. Az adattáblák létrehozása	6
7. Az adattáblák feltöltése	7
8. Lekérdezések	12
9. SQL API, Backend service létrehozása	23
9.1. Felépítés	23
9.2. Modellek	25
9.3. Repository	25
9.4. Module	26
9.5. Controller	28
9.6. HTTP kérések küldése Postmannel	29

1. A feladat leírása

Adatbázisom egy vagy több állatkert hálózatát mutatja be, amiben helyet kapnak az egyes állatkertekben dolgozók, azok feladatai, az állatok és élőhelyeik, eledelük, az eledelt gyártó cégek, illetve az állatok örökbefogadói, ha vannak. Mind az adatbázis tervezésben és mind az SQL megvalósításban angol nyelvet használtam, ugyanis ez a legelterjedtebb nyelv a programozásban.

Összesen 6 egyedet hoztam létre, melyek a következők:

- Employee,
- Site,
- Habitat,
- Animal,
- Food,
- User

Legelőször is érdemes pár szót szólni a **Site** egyedről. Innen indul ki minden. Ez az egyed tárolja el az egyes állatkertek legfőbb tulajdonságait, mint pl. név, terület vagy éppen nyitva tartás. Elsődleges kulcsa a site_id, ami az állatpark azonosítója.

A Site és az **Employee** egyed között egy 1:N kapcsolat van, mivel egy állatkerthez több dolgozó is tartozhat, de egy dolgozó, csak egy állatkerthez tartozhat. Az 1:N kapcsolat neve: **Works**. Egy dolgozónak van azonosítója, vezetéke és keresztnéve (ami ER modellben egy többágú tulajdonság), neme, születési dátuma és ami a legfontosabb, a dolgozó feladatai, posztjai, amiből lehet egy vagy több, így ez egy többértékű tulajdonság lesz. Ez azért fontos, mivel a relációs modellnél ez a tulajdonság egy külön táblát kap majd, amiben lesz a posztnak egy id-ja, a poszt neve, illetve, hogy kihez tartozik.

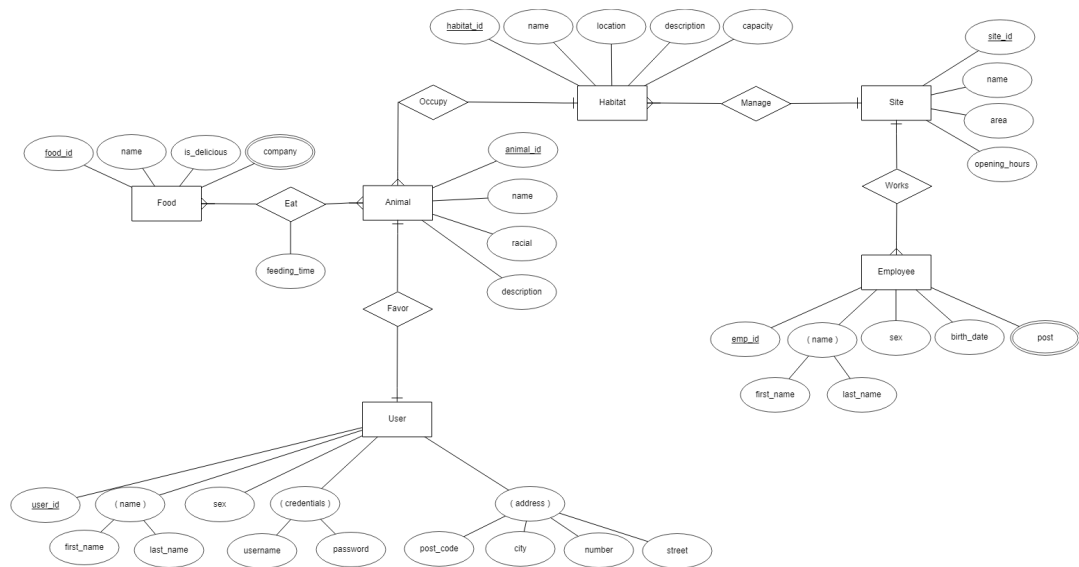
Egy állatkerthez több élőhely is tartozhat, de egy élőhely csak egy állatkerthez tartozik. Ezt ábrázolja a **Manage** kapcsolat, ami 1:N kapcsolattal köti össze a Site és a **Habitat** egyedeket. Az élőhelynek nincsenek "extra" tulajdonságai, van egy azonosítója, neve, térképen való elhelyezkedése, leírása és kapacitása, hogy mennyi állatot képes egyszerre befogadni.

Az **Occupy** kapcsolat szintén 1:N kapcsolattal köti össze a Habitat-ot az **Animal**-l. Az állatnak van azonosítója, neve, faja és leírása.

Itt jön a legelső N:M kapcsolat, az **Eat**, aminek lesz tulajdonsága, a feeding_time, az etetési idő. Fontos, hogy megjegyezzük, az N:M kapcsolat külön kapcsolótáblát fog kapni a relációs modellben. Az Eat köti össze az Animalt a **Food**-dal, ami az állat eledelét modellező egyed. Ennek van azonosítója, neve, egy boolean (logikai) értéke, ami azt dönti el, hogy finom-e az adott eledel, vagy sem. Ezen kívül van egy többértékű tulajdonsága is, az eledelket gyártó cégek, amik szintén külön táblát fognak majd kapni a relációs modellben.

Az állatokat örökbe is lehet fogani bizonyos **User**-eknek, ezt a **Favor** 1:1 kapcsolat modellezi. Talán a Usernek van a legtöbb tulajdonsága ebben az adatbázisban. Van természetesen azonosítója, két neve (vezeték és keresztnév), neme, bejelentkezési adatai (felhasználónév, jelszó), mivel online szeretnénk lebonyolítani az állatok örökbefogadását. Ezen kívül címe is van a felhasználónak, ami az irányítószám, város, utca, házszám tulajdonságokból tevődik össze.

2. Az adatbázis ER modellje



3. Az adatbázis konvertálása relációs modellre

Jobbról balra haladva, előbb létrehozzuk az Employee és az Employee_post táblákat. A többértékű tulajdonsághoz egy külön táblát kell rendelnünk, ahol a foreign key lesz a dolgozó azonosítója és primary key lesz a post illetve a post azonosító. Az utóbbi fog belekerülni az SQL-be, mint elsődleges kulcs. A post egy VARCHAR(30) és NOT NULL az integritási feltétel, ugyanis a munka megnevezését mindenképp meg kell adni. A post_id AUTO_INCREMENT PRIMARY KEY lesz, tehát automatikusan növekvő lesz az azonosító. Ez főként az SQL API, Backend felületén nyújt majd nekünk segítséget.

Az Employee táblában a site_id lesz az idegenkulcs, ami egy INT értékét. A többágú tulajdonságot kettébontjuk first_name és last_name tulajdonságokra. Mindkettő VARCHAR(30) típusú column. A birth_date DATE értéket vesz fel, a sex column pedig CHAR(1). Fontos megjegyezni, hogy itt kötelesek vagyunk csak egy darab karaktert megadni, a választási lehetőség pedig: 'M' = férfi(male), 'F' = nő(female). Természetesen minden column NOT NULL értéket vesz föl.

A Site tábla nem tartalmaz idegenkulcsot és azonosítója INT. A name egy VARCHAR(100), hogy a hosszabb nevű állatkert neve is beleférjen az adatbázisba. Az area egy FLOAT változó, hogy lebegőpontos érték megadására is képes legyen az adatbázis kezelője, illetve az opening_hours column egy VARCHAR(30). Itt is minden NOT NULL.

Jön a Habitat, aminek egyetlen idegenkulcsa van, ez pedig a site_id, ami az adott állatkertre mutat. Neve VARCHAR(30), ahogy a térképen való elhelyezkedés oszlopa is. A leírás, a maximális karakterméretet kapta, VARCHAR(255), ugyanis itt egy hosszabb leírást tehet az adatbázis kezelője az élőhelyről. A kapacitás INT és minden érték NOT NULL.

Az állat is rendelkezik két idegenkulccsal, ezek a: habitat_id, ami az adott élőhelyre mutatnak, illetve a user_id, ami pedig az örökbefogadóra. Ez lehet NULL, ugyanis nem biztos, hogy egy állatnak lesz örökbefogadója. A név és a faj VARCHAR(30), a leírás itt is VARCHAR(255).

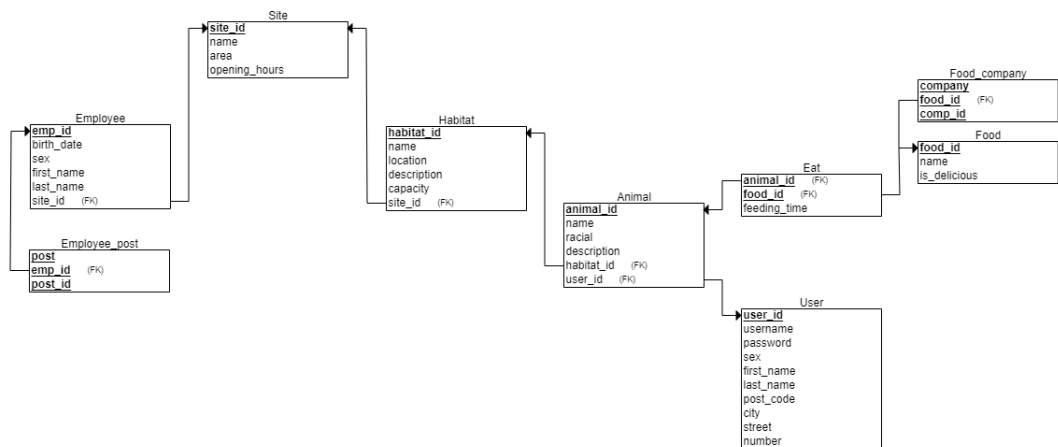
Folytassuk a Userrel, aminek nem lesz idegenkulcsa, csak elsődleges kulcsa, ami INT. A többágú tulajdonságok, mint a cím, a bejelentkezési adatok és a lakcím, itt is különválnak felhasználónév, jelszó, vezetéknév, keresztnév, irányítószám, város, utca és házszámra. Ezek mind VARCHAR(30) értéket vesznek fel a nemet kivéve, ami itt is CHAR(1) és a házszámot, ami INT. Minden érték NOT NULL.

Az N:M kapcsolat relációs táblájával folytatjuk, aminek két idegenkulcsa lesz, az animal_id és a food_id. Ezen kívül megkapta a feeding_time tulajdonságot, ami VARCHAR(30).

A kapcsolótáblából ki is lyukadunk a Food táblára, melynek PRIMARY KEY-e INT, neve VARCHAR(30) és is_delicious tulajdonsága BOOLEAN.

Nem utolsó sorban pedig a második többértékű tulajdonságunk táblája következik, ami a Food_company. Ennek is van saját azonosítója, ami INT, FOREIGN KEY-e, ami szintén egy INT és a Food táblára mutat, illetve egy company tulajdonsága, ami VARCHAR(30) és a cég nevét tartalmazza.

4. Az adatbázis relációs modellje



5. Az adatbázis relációs sémája

Employee [emp_id, birth_date, sex, first_name, last_name, site_id]

Employee_post [post_id, post, emp_id]

Site [site_id, name, area, opening_hours]

Habitat [habitat_id, name, location, description, capacity, site_id]

User [user_id, username, password, sex, first_name, last_name, post_code, city, street, number]

Animal [animal_id, name, racial, description, habitat_id, user_id]

Eat [animal_id, food_id, feeding_time]

Food [food_id, name, is_delicious]

Food_company [comp_id, company, food_id]

6. Az adattáblák létrehozása

Az adattáblák létrehozásánál ügyelni kell a helyes sorrendre. Én előbb azokat a táblákat hoztam létre, amire mutat idegenkulcs, utána pedig azokat, amik csak elsődleges kulcsot tartalmaznak. Az SQL műveleteket a MySQL Server és a Visual Studio Code segítségével írtam.

```
> Run on active connection | Select block
1 DROP DATABASE IF EXISTS Zoo;
2 CREATE DATABASE Zoo;
3
4 DROP TABLE IF EXISTS Zoo.Site;
5 CREATE TABLE Zoo.Site(
6     site_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
7     name VARCHAR(100) NOT NULL,
8     area FLOAT NOT NULL,
9     opening_hours VARCHAR(30) NOT NULL
10 );
11
12 DROP TABLE IF EXISTS Zoo.Employee;
13 CREATE TABLE Zoo.Employee(
14     emp_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
15     first_name VARCHAR(30) NOT NULL,
16     last_name VARCHAR(30) NOT NULL,
17     birth_date DATE NOT NULL,
18     sex CHAR(1) NOT NULL,
19     site_id INT NOT NULL,
20     FOREIGN KEY(site_id) REFERENCES Zoo.Site(site_id) ON DELETE CASCADE
21 );
22
23 DROP TABLE IF EXISTS Zoo.Employee_post;
24 CREATE TABLE Zoo.Employee_post(
25     post_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
26     post VARCHAR(30) NOT NULL,
27     emp_id INT NOT NULL,
28     FOREIGN KEY(emp_id) REFERENCES Zoo.Employee(emp_id) ON DELETE CASCADE
29 );
30
31 DROP TABLE IF EXISTS Zoo.Habitat;
32 CREATE TABLE Zoo.Habitat(
33     habitat_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
34     name VARCHAR(30) NOT NULL,
35     location VARCHAR(30) NOT NULL,
36     description VARCHAR(255) NOT NULL,
37     capacity INT NOT NULL,
38     site_id INT NOT NULL,
39     FOREIGN KEY(site_id) REFERENCES Zoo.Site(site_id) ON DELETE CASCADE
40 );
41
42 DROP TABLE IF EXISTS Zoo.User;
43 CREATE TABLE Zoo.User(
44     user_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
45     username VARCHAR(30) NOT NULL,
46     password VARCHAR(30) NOT NULL,
47     sex CHAR(1) NOT NULL,
48     first_name VARCHAR(30) NOT NULL,
49     last_name VARCHAR(30) NOT NULL,
50     post_code VARCHAR(30) NOT NULL,
51     city VARCHAR(30) NOT NULL,
52     street VARCHAR(30) NOT NULL,
53     number INT NOT NULL
54 );
```

```

55
56 DROP TABLE IF EXISTS Zoo.Animal;
57 CREATE TABLE Zoo.Animal(
58     animal_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
59     name VARCHAR(30) NOT NULL,
60     racial VARCHAR(30) NOT NULL,
61     description VARCHAR(255) NOT NULL,
62     habitat_id INT NOT NULL,
63     user_id INT,
64     FOREIGN KEY(habitat_id) REFERENCES Zoo.Habitat(habitat_id) ON DELETE CASCADE,
65     FOREIGN KEY(user_id) REFERENCES Zoo.User(user_id) ON DELETE CASCADE
66 );
67
68 DROP TABLE IF EXISTS Zoo.Food;
69 CREATE TABLE Zoo.Food(
70     food_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
71     name VARCHAR(30) NOT NULL,
72     is_delicious BOOLEAN NOT NULL
73 );
74
75 DROP TABLE IF EXISTS Zoo.Food_company;
76 CREATE TABLE Zoo.Food_company(
77     comp_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
78     company VARCHAR(30) NOT NULL,
79     food_id INT NOT NULL,
80     FOREIGN KEY(food_id) REFERENCES Zoo.Food(food_id) ON DELETE CASCADE
81 );
82
83 DROP TABLE IF EXISTS Zoo.Eat;
84 CREATE TABLE Zoo.Eat(
85     animal_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
86     food_id INT NOT NULL,
87     feeding_time VARCHAR(30) NOT NULL,
88     FOREIGN KEY(food_id) REFERENCES Zoo.Food(food_id) ON DELETE CASCADE
89 );

```

7. Az adattáblák feltöltése

A feltöltésnél ügyelni kell a helyes sorrendre és arra, hogy megfelelő változótípust használjunk.

```

-- Run on active connection | Select block
-- Table Zoo.Site
1 INSERT INTO Zoo.Site VALUES( 1, 'Miskolci Állatkert', 212000.35, '9:00 - 17:00' );
2 INSERT INTO Zoo.Site VALUES( 2, 'Nyíregyházi Állatpark', 300000.28, '9:00 - 17:00' );
3 INSERT INTO Zoo.Site VALUES( 3, 'Debreceni Állatkert és Vidámpark', 170000.00, '9:00 - 15:30' );
4 INSERT INTO Zoo.Site VALUES( 4, 'Kittenberg Kálmán Állatkert és Botanikus kert, Veszprém', 170500.65, '9:00 - 16:00' );
5 INSERT INTO Zoo.Site VALUES( 5, 'Fővárosi Állat- és Növénykert', 184000.53, '9:00 - 17:30' );
6
7

```



```

8  -- Table Zoo.Employee
9  INSERT INTO Zoo.Employee VALUES( 1, 'Kovács', 'János', '1979-11-02', 'M', 4 );
10 INSERT INTO Zoo.Employee VALUES( 2, 'Jakab', 'József', '1954-12-08', 'M', 1 );
11 INSERT INTO Zoo.Employee VALUES( 3, 'Menyhért', 'András', '2000-05-17', 'M', 2 );
12 INSERT INTO Zoo.Employee VALUES( 4, 'Kis', 'Renáta', '1999-10-10', 'F', 4 );
13 INSERT INTO Zoo.Employee VALUES( 5, 'Verőczei', 'Amália', '2001-12-03', 'F', 5 );
14 INSERT INTO Zoo.Employee VALUES( 6, 'Tóth', 'István', '1968-01-13', 'M', 1 );
15 INSERT INTO Zoo.Employee VALUES( 7, 'Kiss', 'Veronika', '1987-06-09', 'F', 3 );
16 INSERT INTO Zoo.Employee VALUES( 8, 'Magyar', 'Zsófia', '2001-02-28', 'F', 3 );
17 INSERT INTO Zoo.Employee VALUES( 9, 'Adorján', 'Zsolt', '1977-08-28', 'M', 2 );
18 INSERT INTO Zoo.Employee VALUES( 10, 'Mészáros', 'Attila', '1987-05-12', 'M', 3 );
19 INSERT INTO Zoo.Employee VALUES( 11, 'Vass', 'Zsombor', '1957-12-02', 'M', 1 );
20 INSERT INTO Zoo.Employee VALUES( 12, 'Hajdú', 'Patrícia', '1966-10-22', 'F', 5 );
21 INSERT INTO Zoo.Employee VALUES( 13, 'Balla', 'Zsombor', '1977-03-04', 'M', 3 );
22 INSERT INTO Zoo.Employee VALUES( 14, 'Sipos', 'István', '2002-09-10', 'M', 4 );
23 INSERT INTO Zoo.Employee VALUES( 15, 'Illés', 'Patrik', '1988-05-09', 'M', 5 );
24 INSERT INTO Zoo.Employee VALUES( 16, 'Horváth', 'Milla', '1969-03-14', 'F', 2 );
25 INSERT INTO Zoo.Employee VALUES( 17, 'Orbán', 'Kevin', '1999-07-13', 'M', 1 );
26 INSERT INTO Zoo.Employee VALUES( 18, 'László', 'Bence', '1987-12-30', 'M', 4 );
27 INSERT INTO Zoo.Employee VALUES( 19, 'Bogdán', 'Antal', '1965-05-03', 'M', 1 );
28 INSERT INTO Zoo.Employee VALUES( 20, 'Szűcs', 'Gábor', '1964-12-11', 'M', 5 );
29 INSERT INTO Zoo.Employee VALUES( 21, 'Balogh', 'Boglárka', '2000-11-04', 'F', 4 );
30 INSERT INTO Zoo.Employee VALUES( 22, 'Csonka', 'Klaudia', '1978-02-11', 'F', 3 );
31 INSERT INTO Zoo.Employee VALUES( 23, 'Balázs', 'Marianna', '2001-07-09', 'F', 4 );
32 INSERT INTO Zoo.Employee VALUES( 24, 'Faragó', 'Martina', '2003-11-13', 'F', 1 );
33 INSERT INTO Zoo.Employee VALUES( 25, 'Hegedűs', 'Flóra', '1996-01-16', 'F', 2 );
34 INSERT INTO Zoo.Employee VALUES( 26, 'Török', 'Izabella', '1978-05-22', 'F', 5 );
35 INSERT INTO Zoo.Employee VALUES( 27, 'Boros', 'Evelin', '2000-05-03', 'F', 3 );
36 INSERT INTO Zoo.Employee VALUES( 28, 'Bognár', 'Kornél', '1977-06-02', 'M', 2 );
37 INSERT INTO Zoo.Employee VALUES( 29, 'Kelemen', 'Klaudia', '2001-04-19', 'F', 2 );
38 INSERT INTO Zoo.Employee VALUES( 30, 'Dobos', 'Henriett', '1988-07-19', 'F', 5 );
39 INSERT INTO Zoo.Employee VALUES( 31, 'Váradi', 'Bence', '1998-11-07', 'M', 5 );
40 INSERT INTO Zoo.Employee VALUES( 32, 'Simon', 'Zoltán', '1969-12-08', 'M', 4 );
41 INSERT INTO Zoo.Employee VALUES( 33, 'Simon', 'Géza', '1972-03-17', 'M', 4 );
42 INSERT INTO Zoo.Employee VALUES( 34, 'Simon', 'Péter', '2000-10-10', 'M', 5 );
43 INSERT INTO Zoo.Employee VALUES( 35, 'Mészáros', 'Juliana', '1965-04-04', 'F', 3 );
44 INSERT INTO Zoo.Employee VALUES( 36, 'Török', 'Maja', '1978-04-15', 'F', 5 );
45 INSERT INTO Zoo.Employee VALUES( 37, 'Barna', 'Sándor', '1964-12-19', 'M', 3 );
46 INSERT INTO Zoo.Employee VALUES( 38, 'Péter', 'Patrik', '1985-01-28', 'M', 2 );
47 INSERT INTO Zoo.Employee VALUES( 39, 'Tamás', 'Dorottya', '1966-12-11', 'F', 3 );
48 INSERT INTO Zoo.Employee VALUES( 40, 'Pap', 'Aranka', '1953-04-11', 'F', 5 );
49 INSERT INTO Zoo.Employee VALUES( 41, 'László', 'Péter', '1968-12-17', 'M', 2 );
50 INSERT INTO Zoo.Employee VALUES( 42, 'Hegedűs', 'Boglárka', '1988-11-12', 'F', 3 );
51 INSERT INTO Zoo.Employee VALUES( 43, 'Szilágyi', 'Milán', '2003-02-15', 'M', 4 );
52 INSERT INTO Zoo.Employee VALUES( 44, 'Varga', 'Botond', '1977-06-13', 'M', 2 );
53 INSERT INTO Zoo.Employee VALUES( 45, 'Gáspár', 'Mária', '1959-09-17', 'F', 4 );
54 INSERT INTO Zoo.Employee VALUES( 46, 'Simon', 'Livia', '1997-06-14', 'F', 5 );
55 INSERT INTO Zoo.Employee VALUES( 47, 'Gáspár', 'Bence', '1999-07-29', 'M', 1 );
56 INSERT INTO Zoo.Employee VALUES( 48, 'Farkas', 'Áron', '2002-02-28', 'M', 1 );
57 INSERT INTO Zoo.Employee VALUES( 49, 'Fábián', 'Evelin', '1985-03-19', 'F', 1 );
58 INSERT INTO Zoo.Employee VALUES( 50, 'Fodor', 'Kata', '1999-09-23', 'F', 1 );

```

```

60  -- Table Zoo.Employee_post
61  INSERT INTO Zoo.Employee_post VALUES( 1, 'Szemétszedő', 1 );
62  INSERT INTO Zoo.Employee_post VALUES( 2, 'Etető', 2 );
63  INSERT INTO Zoo.Employee_post VALUES( 3, 'Kisállat gondozó', 3 );
64  INSERT INTO Zoo.Employee_post VALUES( 4, 'Terrárium takarító', 4 );
65  INSERT INTO Zoo.Employee_post VALUES( 5, 'Pénztáros', 5 );
66  INSERT INTO Zoo.Employee_post VALUES( 6, 'Jegyszedő', 6 );
67  INSERT INTO Zoo.Employee_post VALUES( 7, 'Karbantartó', 7 );
68  INSERT INTO Zoo.Employee_post VALUES( 8, 'Kisvasút vezető', 8 );
69  INSERT INTO Zoo.Employee_post VALUES( 9, 'Gondnok', 9 );
70  INSERT INTO Zoo.Employee_post VALUES( 10, 'Kalandpark igazgató', 10 );
71  INSERT INTO Zoo.Employee_post VALUES( 11, 'Etető', 11 );
72  INSERT INTO Zoo.Employee_post VALUES( 12, 'Jegyszedő', 12 );
73  INSERT INTO Zoo.Employee_post VALUES( 13, 'Terrárium takarító', 13 );
74  INSERT INTO Zoo.Employee_post VALUES( 14, 'Jegyszedő', 14 );
75  INSERT INTO Zoo.Employee_post VALUES( 15, 'Kisvasút vezető', 15 );
76  INSERT INTO Zoo.Employee_post VALUES( 16, 'Gondnok', 16 );
77  INSERT INTO Zoo.Employee_post VALUES( 17, 'Szemétszedő', 17 );
78  INSERT INTO Zoo.Employee_post VALUES( 18, 'Szemétszedő', 18 );
79  INSERT INTO Zoo.Employee_post VALUES( 19, 'Jegyszedő', 19 );
80  INSERT INTO Zoo.Employee_post VALUES( 20, 'Karbantartó', 20 );
81  INSERT INTO Zoo.Employee_post VALUES( 21, 'Kisállat gondozó', 21 );
82  INSERT INTO Zoo.Employee_post VALUES( 22, 'Etető', 22 );
83  INSERT INTO Zoo.Employee_post VALUES( 23, 'Etető', 23 );
84  INSERT INTO Zoo.Employee_post VALUES( 24, 'Állatorvos', 23 );
85  INSERT INTO Zoo.Employee_post VALUES( 25, 'Gondozó', 24 );
86  INSERT INTO Zoo.Employee_post VALUES( 26, 'Zoo pedagógus', 25 );
87  INSERT INTO Zoo.Employee_post VALUES( 27, 'Gondozó', 26 );
88  INSERT INTO Zoo.Employee_post VALUES( 28, 'Gondozó', 27 );
89  INSERT INTO Zoo.Employee_post VALUES( 29, 'Gondnok', 28 );
90  INSERT INTO Zoo.Employee_post VALUES( 30, 'Jegyszedő', 29 );
91  INSERT INTO Zoo.Employee_post VALUES( 31, 'Pénztáros', 30 );
92  INSERT INTO Zoo.Employee_post VALUES( 32, 'Terrárium takarító', 31 );
93  INSERT INTO Zoo.Employee_post VALUES( 33, 'Zoo pedagógus', 32 );
94  INSERT INTO Zoo.Employee_post VALUES( 34, 'Gondozó', 32 );
95  INSERT INTO Zoo.Employee_post VALUES( 35, 'Szemétszedő', 33 );
96  INSERT INTO Zoo.Employee_post VALUES( 36, 'Kisvasút vezető', 34 );
97  INSERT INTO Zoo.Employee_post VALUES( 37, 'Karbantartó', 35 );
98  INSERT INTO Zoo.Employee_post VALUES( 38, 'Zoo pedagógus', 36 );
99  INSERT INTO Zoo.Employee_post VALUES( 39, 'Gondozó', 37 );
100 INSERT INTO Zoo.Employee_post VALUES( 40, 'Szemétszedő', 38 );
101 INSERT INTO Zoo.Employee_post VALUES( 41, 'Etető', 39 );
102 INSERT INTO Zoo.Employee_post VALUES( 42, 'Jegyszedő', 40 );
103 INSERT INTO Zoo.Employee_post VALUES( 43, 'Gondnok', 41 );
104 INSERT INTO Zoo.Employee_post VALUES( 44, 'Jegyszedő', 42 );
105 INSERT INTO Zoo.Employee_post VALUES( 45, 'Pénztáros', 43 );
106 INSERT INTO Zoo.Employee_post VALUES( 46, 'Terrárium takarító', 44 );
107 INSERT INTO Zoo.Employee_post VALUES( 47, 'Zoo pedagógus', 45 );
108 INSERT INTO Zoo.Employee_post VALUES( 48, 'Kalandpark igazgató', 46 );
109 INSERT INTO Zoo.Employee_post VALUES( 49, 'Etető', 47 );
110 INSERT INTO Zoo.Employee_post VALUES( 50, 'Jegyszedő', 48 );
111 INSERT INTO Zoo.Employee_post VALUES( 51, 'Terrárium takarító', 49 );
112 INSERT INTO Zoo.Employee_post VALUES( 52, 'Gondozó', 50 );

```

```

114 -- Table Zoo.Habitat
115 INSERT INTO Zoo.Habitat VALUES( 1, 'Medve park', '#3', 'Az állatkerti medvék élőhelye. Jelenleg három medve található itt, Jázmin, Andor és Maty
116 INSERT INTO Zoo.Habitat VALUES( 2, 'Főka show', '#2', 'Az állatparki főka show ezen a helyen kerül megrendezésre a nyitást követő minden fél óráb
117 INSERT INTO Zoo.Habitat VALUES( 3, 'Elefánt lak', '#15', 'Az afrikai elefántjaink élőhelye. Trópus a mérsékelt éghajlaton. Csacsi és Béci a két k
118 INSERT INTO Zoo.Habitat VALUES( 4, 'Macik ketrcs', '#4', 'Borka és Dorka, állatkertünk két büszke macskája. Szeretik a látogatókat. Gabi maci, az
119 INSERT INTO Zoo.Habitat VALUES( 5, 'Muflonok dombja', '#22', 'Vadsparkunk muflonjai itt találhatók. Barátságosak, túrista kedvelők, szeretik a
120 INSERT INTO Zoo.Habitat VALUES( 6, 'Oroszlánok szavannája', '#15', 'Morci és Bamba ikertestvérek. A kan oroszlánok állandó civakodása, játéka min
121 INSERT INTO Zoo.Habitat VALUES( 7, 'Kisállatok taveránája', '#1', 'Misi, a mosómedve és barátja, Janka, a círmos vadica nagyon jól eléldegélnék eg
122 INSERT INTO Zoo.Habitat VALUES( 8, 'Majmócák ketrece', '#16', 'Viki és Sanyi állatkertünk büszke páviánjai. Szeretik a látogatókat, csak úgy, mint
123 INSERT INTO Zoo.Habitat VALUES( 9, 'Szurikáták szigete', '#18', 'Ki ne imádná a kis érdeklődő szurikátákat. Nálunk rögtön 4-et is örökre fogadhat
124 INSERT INTO Zoo.Habitat VALUES( 10, 'Baromfi udvar', '#30', 'Állatkertünk baromfi udvarában a hétköznapi házi baromfitól a páván át a gyöngytyúki
125 INSERT INTO Zoo.Habitat VALUES( 11, 'Zsiráfok ketrece', '#14', 'Lunkó és Törpe, a két zsiráf nagyon élvezzi az életet még a ketrecen belül is. Gon
126 INSERT INTO Zoo.Habitat VALUES( 12, 'Kecskek kuckója', '#34', 'Kecskeimogatás? Nálunk az is megvalósulhat. Ha gyermeke arra vágyik, hogy egy pár
127 INSERT INTO Zoo.Habitat VALUES( 13, 'Óposzumok kuckója', '#13', 'Kis szürke barátaink nagyon élvezik az állatkerti létet. Mici, az óposzumbébi,
128 INSERT INTO Zoo.Habitat VALUES( 14, 'Tigris lak', '#10', 'Marci a 10 éves bengáli tigris korához képest nagyon jól tartja magát. Szereti a látoga

130 -- Table Zoo.User
131 INSERT INTO Zoo.User VALUES( 1, 'Allatbarat', 'allat123', 'M', 'Kiss', 'Sándor', '8200', 'Veszprém', 'Petőfi Sándor utca', 3 );
132 INSERT INTO Zoo.User VALUES( 2, 'KisAllatokért', 'allatbaratvagyonok', 'F', 'Megyeri', 'Flóra', '1106', 'Budapest', 'Rákosszőlgyi utca', 27 );
133 INSERT INTO Zoo.User VALUES( 3, 'Vadoc', 'fegy082', 'M', 'Fegyver', 'Sándor', '4024', 'Debrecen', 'Kossuth utca', 26 );
134 INSERT INTO Zoo.User VALUES( 4, 'SzurikatakKedvence', 'szuri123', 'M', 'Hajnal', 'Sándor', '8200', 'Veszprém', 'Adám István utca', 15 );
135 INSERT INTO Zoo.User VALUES( 5, 'OroszlánMama', 'mamaoroszlán', 'F', 'Kis', 'Mária', '4002', 'Debrecen', 'Gombvirág utca', 12 );
136 INSERT INTO Zoo.User VALUES( 6, 'MaciNagyil', 'macinagyil', 'F', 'Sándorné Arany', 'Virág', '3525', 'Miskolc', 'Estike utca', 10 );
137 INSERT INTO Zoo.User VALUES( 7, 'VikiPapa', 'vikipapa', 'M', 'Andor', 'Ferenc', '3525', 'Miskolc', 'Mohostó utca', 15 );
138 INSERT INTO Zoo.User VALUES( 8, 'Possumlvr', 'possumlover', 'F', 'Kazai', 'Eszter', '3521', 'Miskolc', 'Új élet utca', 24 );
139 INSERT INTO Zoo.User VALUES( 9, 'MosóMisi', 'misimoso', 'M', 'Virág', 'György', '1181', 'Budapest', 'Klapka György utca', 4 );
140 INSERT INTO Zoo.User VALUES( 10, 'FokaMan', 'fokaman25', 'M', 'Károly', 'Mihály', '4405', 'Nyíregyháza', 'Nárcisz utca', 73 );
141 INSERT INTO Zoo.User VALUES( 11, 'Cicatover', 'cicatover123', 'F', 'Macskás', 'Márta', '1102', 'Budapest', 'Baross utca', 5 );

143 -- Table Zoo.Animal
144 INSERT INTO Zoo.Animal VALUES( 1, 'Mici', 'Óposzum', 'Az állatkert egyetlen óposzumbébije', 13, 8 );
145 INSERT INTO Zoo.Animal VALUES( 2, 'Borka', 'Medve', 'Az állatkert egyik nőstény medvéje', 4, 6 );
146 INSERT INTO Zoo.Animal VALUES( 3, 'Dorka', 'Medve', 'Az állatkert egyik nőstény medvéje', 4, NULL );
147 INSERT INTO Zoo.Animal VALUES( 4, 'Jázmin', 'Medve', 'Az állatkert nőstény medvéje', 1, NULL );
148 INSERT INTO Zoo.Animal VALUES( 5, 'Andor', 'Medve', 'Az állatkert egyik kan medvéje', 1, NULL );
149 INSERT INTO Zoo.Animal VALUES( 6, 'Matyko', 'Medve', 'Az állatkert egyik kan medvéje', 1, NULL );
150 INSERT INTO Zoo.Animal VALUES( 7, 'Csacsi', 'Elefánt', 'Az állatkert egyik elefántja', 3, NULL );
151 INSERT INTO Zoo.Animal VALUES( 8, 'Béci', 'Elefánt', 'Az állatkert egyik elefántja', 3, NULL );
152 INSERT INTO Zoo.Animal VALUES( 9, 'Morci', 'Oroszlán', 'Az állatkert egyik oroszlánja', 6, 5 );
153 INSERT INTO Zoo.Animal VALUES( 10, 'Bamba', 'Oroszlán', 'Az állatkert egyik oroszlánja', 6, NULL );
154 INSERT INTO Zoo.Animal VALUES( 11, 'Misi', 'Mosómedve', 'Az állatkert egyetlen mosómedvéje', 7, 9 );
155 INSERT INTO Zoo.Animal VALUES( 12, 'Janka', 'Vadmacska', 'Az állatkert egyetlen vadmacskája', 7, 11 );
156 INSERT INTO Zoo.Animal VALUES( 13, 'Viki', 'Pávián', 'Az állatkert nőstény páviánja', 8, 7 );
157 INSERT INTO Zoo.Animal VALUES( 14, 'Sanyi', 'Pávián', 'Az állatkert kan páviánja', 8, NULL );
158 INSERT INTO Zoo.Animal VALUES( 15, 'Lunkó', 'Zsiráf', 'Az állatkert egyik zsiráfja', 11, NULL );
159 INSERT INTO Zoo.Animal VALUES( 16, 'Törpe', 'Zsiráf', 'Az állatkert egyik zsiráfja', 11, NULL );
160 INSERT INTO Zoo.Animal VALUES( 17, 'Lóci', 'Főka', 'Az állatpark egyik főka kanja', 2, 10 );
161 INSERT INTO Zoo.Animal VALUES( 18, 'Móci', 'Főka', 'Az állatpark egyik főka kanja', 2, NULL );
162 INSERT INTO Zoo.Animal VALUES( 19, 'Hegyes', 'Muflon', 'Az állatkert kan muflonja', 5, NULL );
163 INSERT INTO Zoo.Animal VALUES( 20, 'Kis Hegyes', 'Muflon', 'Az állatkert nőstény muflonja', 5, NULL );
164 INSERT INTO Zoo.Animal VALUES( 21, 'Figyelő', 'Szurikáta', 'Az állatkert legidősebb szurikátája', 9, 4 );
165 INSERT INTO Zoo.Animal VALUES( 22, 'Mókás', 'Szurikáta', 'Az állatkert legfiatalabb szurikátája', 9, NULL );
166 INSERT INTO Zoo.Animal VALUES( 23, 'Tollas', 'Gyöngytyúk', 'Az állatkert egyetlen gyöngytyúkjá', 10, 1 );
167 INSERT INTO Zoo.Animal VALUES( 24, 'Szépséges', 'Páva', 'Az állatkert egyetlen pávája', 10, NULL );
168 INSERT INTO Zoo.Animal VALUES( 25, 'Szurkos', 'Szurikáta', 'Az állatkert egyik fiatal szurikátája', 9, NULL );
169 INSERT INTO Zoo.Animal VALUES( 26, 'Bébi', 'Szurikáta', 'Az állatkert egyik idősebb szurikátája', 9, 2 );
170 INSERT INTO Zoo.Animal VALUES( 27, 'Marci', 'Bengáli tigris', 'Az állatkert egyetlen bengáli tigrise', 14, 3 );
171 INSERT INTO Zoo.Animal VALUES( 28, 'Kis Bak', 'Kecske', 'Az állatkert egy fiatal kecskéje', 12, NULL );
172 INSERT INTO Zoo.Animal VALUES( 29, 'Nagy Bak', 'Kecske', 'Az állatkert egy idősebb kecskéje', 12, NULL );
173 INSERT INTO Zoo.Animal VALUES( 30, 'Jézsi', 'Házi baromfi', 'Az állatkert házi baromfija', 10, NULL );

```

```

175 -- Table Zoo.Food
176 INSERT INTO Zoo.Food VALUES( 1, 'Fagyasztott nyershús', false );
177 INSERT INTO Zoo.Food VALUES( 2, 'Sárgarépa', true );
178 INSERT INTO Zoo.Food VALUES( 3, 'Spenót', false );
179 INSERT INTO Zoo.Food VALUES( 4, 'Sertés borda', true );
180 INSERT INTO Zoo.Food VALUES( 5, 'Aszalt gyümölcsök', true );
181 INSERT INTO Zoo.Food VALUES( 6, 'Sült hús', true );
182 INSERT INTO Zoo.Food VALUES( 7, 'Fagyasztott hering', true );
183
184 -- Table Zoo.Food_company
185 INSERT INTO Zoo.Food_company VALUES( 1, 'Állati zöldség/gyümölcs', 2 );
186 INSERT INTO Zoo.Food_company VALUES( 2, 'Állati zöldség/gyümölcs', 3 );
187 INSERT INTO Zoo.Food_company VALUES( 3, 'Állati zöldség/gyümölcs', 5 );
188 INSERT INTO Zoo.Food_company VALUES( 4, 'Felix állati eledel', 4 );
189 INSERT INTO Zoo.Food_company VALUES( 5, 'Felix állati eledel', 6 );
190 INSERT INTO Zoo.Food_company VALUES( 6, 'Frosty food', 1 );
191 INSERT INTO Zoo.Food_company VALUES( 7, 'Frosty food', 7 );
192
193 -- Table Zoo.Eat
194 INSERT INTO Zoo.Eat VALUES( 1, 1, '07:00, 18:00' );
195 INSERT INTO Zoo.Eat VALUES( 2, 4, '05:00, 12:00, 19:00' );
196 INSERT INTO Zoo.Eat VALUES( 3, 4, '05:00, 12:00, 19:00' );
197 INSERT INTO Zoo.Eat VALUES( 4, 6, '09:00, 13:00, 20:00' );
198 INSERT INTO Zoo.Eat VALUES( 5, 6, '09:00, 13:00, 20:00' );
199 INSERT INTO Zoo.Eat VALUES( 6, 6, '09:00, 13:00, 20:00' );
200 INSERT INTO Zoo.Eat VALUES( 7, 2, '05:00, 12:00, 19:00' );
201 INSERT INTO Zoo.Eat VALUES( 8, 3, '05:00, 12:00, 19:00' );
202 INSERT INTO Zoo.Eat VALUES( 9, 4, '05:00, 13:00, 20:00' );
203 INSERT INTO Zoo.Eat VALUES( 10, 4, '05:00, 13:00, 20:00' );
204 INSERT INTO Zoo.Eat VALUES( 11, 5, '09:00, 18:00' );
205 INSERT INTO Zoo.Eat VALUES( 12, 1, '07:00, 20:00' );
206 INSERT INTO Zoo.Eat VALUES( 13, 5, '06:00, 14:00, 20:00' );
207 INSERT INTO Zoo.Eat VALUES( 14, 5, '06:00, 14:00, 20:00' );
208 INSERT INTO Zoo.Eat VALUES( 15, 2, '07:00, 12:00, 18:00' );
209 INSERT INTO Zoo.Eat VALUES( 16, 2, '07:00, 12:00, 18:00' );
210 INSERT INTO Zoo.Eat VALUES( 17, 7, '07:00, 12:00, 18:00' );
211 INSERT INTO Zoo.Eat VALUES( 18, 7, '07:00, 12:00, 18:00' );
212 INSERT INTO Zoo.Eat VALUES( 19, 3, '05:00, 12:00, 17:00' );
213 INSERT INTO Zoo.Eat VALUES( 20, 5, '05:00, 12:00, 17:00' );
214 INSERT INTO Zoo.Eat VALUES( 21, 1, '07:00, 19:00' );
215 INSERT INTO Zoo.Eat VALUES( 22, 6, '08:00, 13:00, 20:00' );
216 INSERT INTO Zoo.Eat VALUES( 23, 2, '05:00, 20:00' );
217 INSERT INTO Zoo.Eat VALUES( 24, 3, '05:00, 20:00' );
218 INSERT INTO Zoo.Eat VALUES( 25, 6, '08:00, 13:00, 20:00' );
219 INSERT INTO Zoo.Eat VALUES( 26, 1, '08:00, 13:00, 20:00' );
220 INSERT INTO Zoo.Eat VALUES( 27, 4, '05:00, 12:00, 18:00' );
221 INSERT INTO Zoo.Eat VALUES( 28, 2, '07:00, 12:00, 18:00' );
222 INSERT INTO Zoo.Eat VALUES( 29, 3, '07:00, 12:00, 18:00' );
223 INSERT INTO Zoo.Eat VALUES( 30, 2, '05:00, 20:00' );

```

8. Lekérdezések

1. Kérdezzük le, melyik állatkert van a legtovább nyitva!

π opening_hours (Zoo.Site \rightarrow zoo_site)

```
Run on active connection | Select block
1 SELECT zoo_site.name AS Zoo FROM Zoo.Site zoo_site
2 ORDER BY opening_hours DESC
3 LIMIT 1;
```

Zoo
Fővárosi Állat- és Növénykert

2. Számoljuk meg hány női dolgozó van!

π COUNT(sex) \rightarrow Number_of_females
 σ (sex = 'F')

```
Run on active connection | Select block
1 SELECT COUNT(sex) AS Number_of_females FROM Zoo.Employee
2 WHERE sex = 'F';
```

Number_of_females
24

3. Keressük ki, ki a legidősebb dolgozó!

π birth_date (Zoo.Employee)

```

Run on active connection | Select block
1 SELECT first_name AS Surname, last_name AS Forename FROM Zoo.Employee
2 ORDER BY birth_date ASC
3 LIMIT 1;

```

Results

Surname	Forename
Pap	Aranka

4. Listázzuk ki, hogy kik a 'Simon' vezetéknévűek és mikor születtek!

π emp.first_name \rightarrow Surname,
emp.last_name \rightarrow Forename,
emp.birth_date \rightarrow Birth_date
 σ (emp.first_name = 'Simon')

```

Run on active connection | Select block
1 SELECT emp.first_name AS Surname,
2     emp.last_name AS Forename,
3     emp.birth_date AS Birth_date
4 FROM Zoo.Employee emp
5 WHERE emp.first_name = 'Simon';

```

Results

Surname	Forename	Birth_date
Simon	Zoltán	Mon Dec 08 1969 00:00:00 GMT+0100 (közép-európai téli idő)
Simon	Géza	Fri Mar 17 1972 00:00:00 GMT+0100 (közép-európai téli idő)
Simon	Péter	Tue Oct 10 2000 00:00:00 GMT+0200 (közép-európai nyári idő)
Simon	Livia	Sat Jun 14 1997 00:00:00 GMT+0200 (közép-európai nyári idő)

5. Listázzuk ki a gondozókat betűrendben!

π^*
 O (post.post = 'Gondozó') (Zoo.Employee \rightarrow emp ∞ post.emp_id = emp.emp_id
 Zoo.Employee_post \rightarrow post)

Run on active connection | Select block

```

1 SELECT * FROM Zoo.Employee emp
2 JOIN Zoo.Employee_post post ON post.emp_id = emp.emp_id
3 WHERE post.post = 'Gondozó'
4 ORDER BY first_name ASC;

```

Results

emp_id	first_name	last_name	birth_date	sex	site_id	post_id	post
37	Barna	Sándor	Sat Dec 19 1964 00:00:00 GMT+0100 (közép-európai téli idő)	M	3	39	Gondozó
27	Boros	Evelin	Wed May 03 2000 00:00:00 GMT+0200 (közép-európai nyári idő)	F	3	28	Gondozó
24	Faragó	Martina	Thu Nov 13 2003 00:00:00 GMT+0100 (közép-európai téli idő)	F	1	25	Gondozó
50	Fodor	Kata	Thu Sep 23 1999 00:00:00 GMT+0200 (közép-európai nyári idő)	F	1	52	Gondozó
32	Simon	Zoltán	Mon Dec 08 1969 00:00:00 GMT+0100 (közép-	M	4	34	Gondozó

6. Keressük ki, hogy a 'Medve park' melyik állatkertben van!

π sit.name \rightarrow Zoo
 σ (hab.name = 'Medve park') (Zoo.Site \rightarrow sit ∞ sit.site_id = hab.site_id
Zoo.Habitat \rightarrow hab)

```
Run on active connection | Select block
1 SELECT sit.name AS Zoo FROM Zoo.Site sit
2 JOIN Zoo.Habitat hab
3 ON sit.site_id = hab.site_id
4 WHERE hab.name = 'Medve park';
```

Results X

Zoo
Fővárosi Állat- és Növénykert

7. Nézzük meg mennyi az átlag kapacitás!

π AVG(capacity) \rightarrow Avarage_capacity

```
Run on active connection | Select block
1 SELECT AVG(capacity) AS Avarage_capacity FROM Zoo.Habitat;
```

Results X

Avarage_capacity
4.2143

8. Keressük meg azt az állatot, melynek leírásában benne van, hogy 'etetni nem'!

π anim.name \rightarrow Name
 σ (hab.description LIKE '%etetni%nem%') (Zoo.Animal \rightarrow anim \propto anim.habitat_id = hab.habitat_id Zoo.Habitat \rightarrow hab)

```
Run on active connection | Select block
1 SELECT anim.name AS Name FROM Zoo.Animal anim
2 JOIN Zoo.Habitat hab ON anim.habitat_id = hab.habitat_id
3 WHERE hab.description LIKE '%etetni%nem%';
```

Results

Name
Mici

9. Írassuk ki azoknak az állatoknak a nevét, akiknek az örökbe-fogadjuk pesti!

π anim.name \rightarrow Name
 σ (user.city = 'Budapest') (Zoo.Animal \rightarrow anim \propto user.user_id = anim.user_id Zoo.User \rightarrow user)

```
Run on active connection | Select block
1 SELECT anim.name AS Name FROM Zoo.Animal anim
2 JOIN Zoo.User user ON user.user_id = anim.user_id
3 WHERE user.city = 'Budapest';
```

Results

Name
Bébi
Misi
Janka

10. Írassuk ki azoknak az állatoknak a faját, akik a harmadik térképhegyen helyezkednek el!

γ anim.racial \rightarrow Racial
 σ (hab.location = '#3') (Zoo.Animal \rightarrow anim ∞ anim.habitat_id = hab.habitat_id
 Zoo.Habitat \rightarrow hab)

```

  ▶ Run on active connection | ≡ Select block
1 SELECT DISTINCT anim.racial AS Racial FROM Zoo.Animal anim
2 JOIN Zoo.Habitat hab ON anim.habitat_id = hab.habitat_id
3 WHERE hab.location = '#3';

```

Results X ▶ □ ...

Racial
Medve

11. Írassuk ki annak a felhasználónak a keresztnévét, akinek az állata oposzum fajú!

π user.last_name \rightarrow Forename
 σ (racial = 'Oposzum') (Zoo.User \rightarrow user ∞ user.user_id = anim.user_id
 Zoo.Animal \rightarrow anim)

```

  ▶ Run on active connection | ≡ Select block
1 SELECT user.last_name AS Forename FROM Zoo.User user
2 JOIN Zoo.Animal anim
3 ON user.user_id = anim.user_id
4 WHERE racial = 'Oposzum';

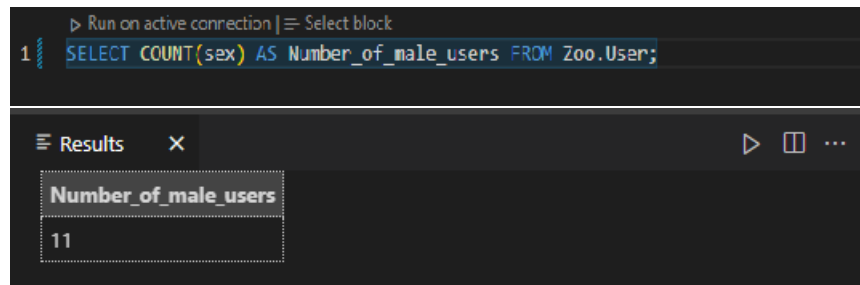
```

Results X ▶ □ ...

Forename
Eszter

12. Számoljuk meg, hány női felhasználó van!

π COUNT(sex) \rightarrow Number_of_male_users



```
1 SELECT COUNT(sex) AS Number_of_male_users FROM Zoo.User;
```

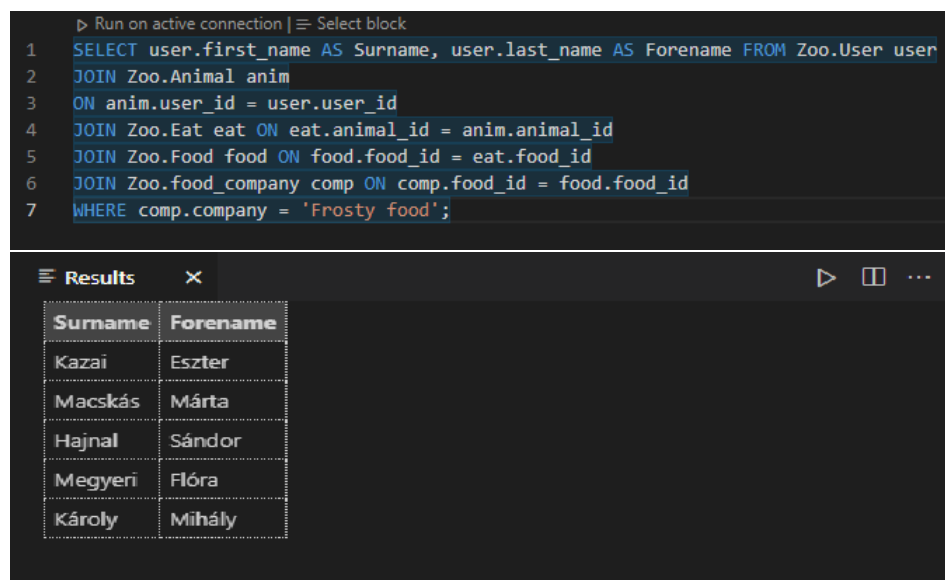
Results

Number_of_male_users
11

13. Írassuk ki azoknak a felhasználóknak a nevét, akiknek az állatai a 'Frosty food' eledelét eszik!

π user.first_name \rightarrow Surname,
user.last_name \rightarrow Forename

σ (comp.company = 'Frosty food') (Zoo.User \rightarrow user ∞ anim.user_id = user.user_id Zoo.Animal \rightarrow anim ∞ eat.animal_id = anim.animal_id Zoo.Eat \rightarrow eat ∞ food.food_id = eat.food_id Zoo.Food \rightarrow food ∞ comp.food_id = food.food_id Zoo.food_company \rightarrow comp)



```
1 SELECT user.first_name AS Surname, user.last_name AS Forename FROM Zoo.User user
2 JOIN Zoo.Animal anim
3 ON anim.user_id = user.user_id
4 JOIN Zoo.Eat eat ON eat.animal_id = anim.animal_id
5 JOIN Zoo.Food food ON food.food_id = eat.food_id
6 JOIN Zoo.food_company comp ON comp.food_id = food.food_id
7 WHERE comp.company = 'Frosty food';
```

Results

Surname	Forename
Kazai	Eszter
Macskás	Márta
Hajnal	Sándor
Megyeri	Flóra
Károly	Mihály

14. Listázzuk ki azoknak az állatoknak a nevét, akik a 'Felix' ele-
delét eszik és biztosan délben is esznek.

π anim.name \rightarrow Name
 σ (comp.company LIKE 'Felix%' AND eat.feeding_time LIKE '%12:00%')
 (Zoo.Animal \rightarrow anim ∞ anim.animal_id = eat.animal_id Zoo.Eat \rightarrow eat ∞
 eat.food_id = food.food_id Zoo.food \rightarrow food ∞ food.food_id = comp.food_id
 Zoo.food_company \rightarrow comp)

```

> Run on active connection | Select block
1 SELECT anim.name AS Name FROM Zoo.Animal anim
2 JOIN Zoo.Eat eat ON anim.animal_id = eat.animal_id
3 JOIN Zoo.food food ON eat.food_id = food.food_id
4 JOIN Zoo.Food_company comp ON food.food_id = comp.food_id
5 WHERE comp.company LIKE 'Felix%' AND eat.feeding_time LIKE '%12:00%';

```

Results

Name
Borka
Dorka
Marci

15. Írassuk ki annak az állatoknak a nevét és faját, akik sárgaré-
pát esznek este 6 órakor!

π anim.name \rightarrow Name,
 anim.racial \rightarrow Racial
 σ (food.name = 'Sárgarépa' AND eat.feeding_time LIKE '%18:00%') (Zoo.Animal
 \rightarrow anim ∞ anim.animal_id = eat.animal_id Zoo.Eat \rightarrow eat ∞ eat.food_id
 = food.food_id Zoo.Food \rightarrow food)

```

> Run on active connection | Select block
1 SELECT anim.name AS Name, anim.racial AS Racial FROM Zoo.Animal anim
2 JOIN Zoo.Eat eat ON anim.animal_id = eat.animal_id
3 JOIN Zoo.Food food ON eat.food_id = food.food_id
4 WHERE food.name = 'Sárgarépa' AND eat.feeding_time LIKE '%18:00%';

```

Results

Name	Racial
Lurkó	Zsiráf
Törpe	Zsiráf
Kis Bak	Kecske

16. Listázzuk ki azoknak az állatoknak a nevét és az eledelét, akik nem finom ételeket esznek. Ügyeljünk a betűrendre!

π anim.name \rightarrow Name
 σ (is_delicious = false) (Zoo.Animal \rightarrow anim \bowtie anim.animal_id = eat.animal_id
 Zoo.Eat \rightarrow eat \bowtie eat.food_id = food.food_id Zoo.Food \rightarrow food)

```

▶ Run on active connection | ≡ Select block
1 SELECT anim.name AS Name, food.name AS Food FROM Zoo.Animal anim
2 JOIN Zoo.Eat eat ON anim.animal_id = eat.animal_id
3 JOIN Zoo.Food food ON eat.food_id = food.food_id
4 WHERE is_delicious = false
5 ORDER BY anim.name ASC;

```

Results

Name	Food
Bébi	Fagyasztott nyershús
Béci	Spenót
Figyelő	Fagyasztott nyershús
Hegyes	Spenót
Janka	Fagyasztott nyershús
Mici	Fagyasztott nyershús
Nagy Bak	Spenót
Szépséges	Spenót

17. Írassuk ki azoknak a felhasználóknak a címét, akiknek az állata a Veszprémi állatkertben van!

π user.post_code \rightarrow Post_code,
 user.city \rightarrow City,
 user.street \rightarrow Street,
 user.number \rightarrow Number
 σ (zoo_site.name LIKE '%Veszprém%') γ user.street (Zoo.User \rightarrow user
 ∞ user.user_id = anim.user_id Zoo.Animal \rightarrow anim ∞ anim.habitat_id =
 habit.habitat_id Zoo.Habitat \rightarrow habit ∞ habit.site_id = zoo_site.site_id
 Zoo.Site \rightarrow zoo_site)

```

1 SELECT user.post_code AS Post_code, user.city AS City, user.street AS Street, user.number AS Number
2 FROM Zoo.User user
3 JOIN Zoo.Animal anim ON user.user_id = anim.user_id
4 JOIN Zoo.Habitat habit ON anim.habitat_id = habit.habitat_id
5 JOIN Zoo.Site zoo_site ON habit.site_id = zoo_site.site_id
6 WHERE zoo_site.name LIKE '%Veszprém%'
7 GROUP BY user.street;

```

Post_code	City	Street	Number
8200	Veszprém	Adám István utca	15
1106	Budapest	Rákospölgyei utca	27
8200	Veszprém	Petőfi Sándor utca	3

18. Listázzuk ki azokat a felhasználókat, akik Budapesten laknak és máshonnan fogadnak örökbe!

π user.first_name \rightarrow Surname,
 user.last_name \rightarrow Forename
 σ (user.city = 'Budapest' AND zoo_site.name NOT LIKE 'Fővárosi%')
 (Zoo.User \rightarrow user ∞ anim.user_id = user.user_id Zoo.Animal \rightarrow anim ∞
 habit.habitat_id = anim.habitat_id Zoo.Habitat \rightarrow habit ∞ zoo_site.site_id
 = habit.site_id Zoo.Site \rightarrow zoo_site)

```

1 SELECT user.first_name AS Surname, user.last_name AS Forename FROM Zoo.User user
2 JOIN Zoo.Animal anim ON anim.user_id = user.user_id
3 JOIN Zoo.Habitat habit ON anim.habitat_id = habit.habitat_id
4 JOIN Zoo.Site zoo_site ON habit.site_id = zoo_site.site_id
5 WHERE user.city = 'Budapest' AND zoo_site.name NOT LIKE 'Fővárosi%';

```

Surname	Forename
Megyeri	Flóra

19. Írassuk ki hány férfi dolgozója van a pesti állatkertnek, akik 1970 előtt születtek!

π COUNT(sex) \rightarrow Count,
first_name \rightarrow Surname,
last_name \rightarrow Forename

σ (zoo_site.name LIKE 'Fővárosi%' AND emp.sex = 'M' AND emp.birth_date < '1970-01-01') (Zoo.Employee \rightarrow emp ∞ zoo_site.site_id = emp.site_id Zoo.Site \rightarrow zoo_site)

```

> Run on active connection | Select block
1 SELECT COUNT(sex) AS Count, first_name AS Surname, last_name AS Forename FROM Zoo.Employee emp
2 JOIN Zoo.Site zoo_site ON zoo_site.site_id = emp.site_id
3 WHERE zoo_site.name LIKE 'Fővárosi%' AND emp.sex = 'M' AND emp.birth_date < '1970-01-01';

```

Count	Surname	Forename
1	Szűcs	Gábor

20. Listázzuk ki azokat a felhasználók felhasználónevét, akiknek az állata aszalt gyümölcsöt eszik és a felhasználó nem pesti!

π user.username \rightarrow Username

σ (food.name = 'Aszalt gyümölcsök' AND user.city != 'Budapest') (Zoo.User \rightarrow user ∞ anim.user_id = user.user_id Zoo.Animal \rightarrow anim ∞ eat.animal_id = anim.animal_id Zoo.Eat \rightarrow eat ∞ food.food_id = eat.food_id Zoo.Food \rightarrow food)

```

> Run on active connection | Select block
1 SELECT user.username AS Username FROM Zoo.User user
2 JOIN Zoo.Animal anim ON anim.user_id = user.user_id
3 JOIN Zoo.Eat eat ON eat.animal_id = anim.animal_id
4 JOIN Zoo.Food food ON food.food_id = eat.food_id
5 WHERE food.name = 'Aszalt gyümölcsök' AND user.city != 'Budapest';

```

Username
VikiPapa

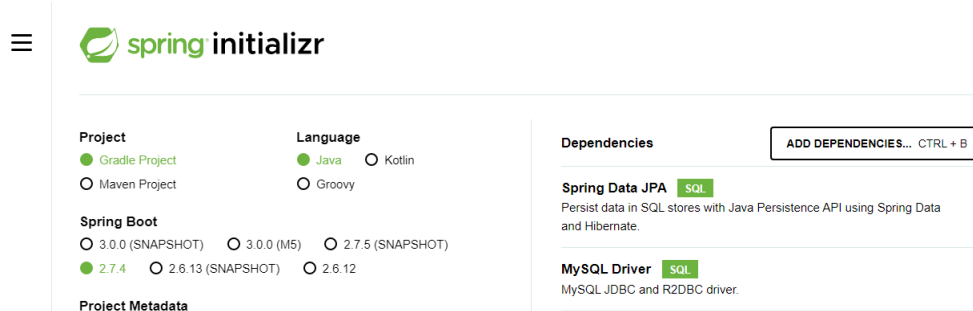
9. SQL API, Backend service létrehozása

9.1. Felépítés

A Backend servicem lényege, hogy a **CRUD** függvényeket megvalósítsam. Ez egy angol nyelvű rövidítés, ami a következőkből épül fel:

1. Create,
2. Read,
3. Update,
4. Remove

Azaz írni, olvasni, módosítani, törölni az adatbázisból, adatbázisba. A Backend felépítését Java nyelven készítettem el és a **Spring framework** keretrendszert használtam fel hozzá. A Spring egy nyílt forráskódú, inversion of controllt megvalósító Java alkalmazás keretrendszer. A Spring initializr-t használtam fel a saját API packagem létrehozására.



A dependences fülön egyértelműen kell a MySQL Driver, ami biztosítja a MySQL connectiont a java applikációban illetve szükségünk lesz még a SpringData JPA, azaz a Java Persistence Api-ra.

Mentsük le, csomagoljuk ki az API package-t, majd nézzünk bele a pom.xml-be. Itt találunk meg minden információt az applikációnkról, illetve a dependencies alatt az általunk behozott dependencyket. Fontos, hogy a megfelelő verziójú mysql-connectort töltsük be. Legelőször a main classal ismerkedünk meg.


```

1  package com.zoo.api;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  public class ApiApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(ApiApplication.class, args);
11     }
12 }

```

Ahhoz, hogy SQL API-t tudjunk írni szükségünk lesz egyéb classokra.

- Modellekre,
- Repository interface-kre,
- Module-okra,
- Controllerekre

A modellek fogják nekünk leírni az egyes adattáblákat Java-ban. Különböző annotációkkal fogjuk ellátni őket illetve a bennük található elemeket.

A repositorykban lesznek a megvalósítandó függvények definíciói. Itt kell azt is megadni majd, hogy mi a modell elsődleges kulcsa.

A moduleokban fogjuk implementálni a repositorykban definiált függvényeket.

A controllerekben pedig a HTTP Request-eket fogjuk tudni lekezelni.

Van egy fontos lépés még a classjaink létrehozása előtt. Nyissuk meg az **application.properties** fájlt és végezzünk el rajta pár módosítást.

```

application.properties
api > src > main > resources > application.properties
1  spring.datasource.url=jdbc:mysql://localhost:3306/zoo?useSSL=false&serverTimezone=UTC
2  spring.datasource.username=
3  spring.datasource.password=
4  spring.jpa.hibernate.ddl-auto=update

```

A spring.datasource.url tartalmazza azt a címet, ahol futni fog a Backendünk. Jelen esetben ez localhost lesz. Az URL tartalmazza az adatbázis nevét, ami jelen esetben zoo.

A username és a password az MySQL connection-nek a felhasználóneve és jelszava.

9.2. Modellek

```
1 package com.zoo.api.Models;
2
3 import javax.persistence.*;
4
5 @Entity
6 public class Site {
7
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    private int site_id;
11
12    @Column
13    private String name;
14
15    @Column
16    private float area;
17
18    @Column
19    private String opening_hours;
20
21    public int getSite_id() {
22        return site_id;
23    }
24    public void setSite_id(int site_id) {
25        this.site_id = site_id;
26    }
27    public String getName() {
28        return name;
29    }
30    public void setName(String name) {
31        this.name = name;
32    }
33 }
```

A `javax.persistence` importtal tudunk különböző adatbázis annotációkat rakni a classunkba. Jelen esetben a **@Entity** fogja jelölni, hogy ez egy külön entitás, melynek tulajdonságai vannak. Ezeket a tulajdonságokat a classon belül deklaráljuk. **@Id**-val és a **@GeneratedValue(strategy = GenerationType.IDENTITY)** annotációkkal definiáljuk, hogy ez lesz az a változó, ami az elsődleges kulcsa lesz az adattáblánknak. A **Column** a további oszlopokat jelöli. Hozzuk létre az alábbi modellhez a repositoryt.

9.3. Repository

```
1 package com.zoo.api.Repo;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import com.zoo.api.Models.Site;
5
6 public interface SiteRepo extends JpaRepository<Site, Integer> {
7
8 }
9
```

Mivel a `JpaRepository` minden tulajdonsága öröklődik, így nekünk nem kell újabb CRUD függvényeket írunk elég a meglévőket használni. Fontos, hogy a kacsacsőrök közé meg kell adnunk első paraméterként, hogy melyik modellre

akarunk CRUD utasításokat kiadni, illetve második paraméterként, hogy mi az elsődleges kulcs csomagoló osztálya. Jelen esetben ez egy Integer.

9.4. Module

```
1 package com.zoo.api.Modules;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7 import com.zoo.api.Models.Site;
8 import com.zoo.api.Repo.SiteRepo;
9
10 @Service
11 public class SiteModules {
12
13     @Autowired
14     private SiteRepo siteRepository;
15
16
17     public Site storeSite(Site site)
18     {
19         return siteRepository.save(site);
20     }
21
22     public List<Site> getAllSite()
23     {
24         return siteRepository.findAll();
25     }
26
27     public Site getSiteById(int id)
28     {
29         return siteRepository.findById(id).orElseThrow(() -> new RuntimeException("Site found for the id "+id));
30     }
31
32     public Site updateSite(Site site, int id) {
33         Site updatedSite = siteRepository.findById(id).get();
34         updatedSite.setName(site.getName());
35         updatedSite.setArea(site.getArea());
36         updatedSite.setOpening_hours(site.getOpening_hours());
37         return siteRepository.save(updatedSite);
38     }
39
40     public void deleteSite(int id) {
41         Site deleteSite = siteRepository.findById(id).get();
42         siteRepository.delete(deleteSite);
43     }
44 }
45
```

Első és legfontosabb, hogy deklaráljuk, hogy ez servicet építünk, ezt a **@Service** annotációval tesszük meg. **@Autowired**-del adunk értéket a repository változónknak. Ezután kezdődik a függvények megírása. Site visszatérése lesz a Create függvénynek, ugyanis egy konkrét Site-ot szeretnénk letárolni az adatbázisban. Paramétere is egy Site, amit be szeretnénk tölteni az adatbázisba. A repository változónk tartalmazza a **save** metódust, így ennek a megírásával már nem kell bajlódni. Fogjuk és elmentjük a paraméterként megadott Site-ot.

Jön a Read, ami jelen esetben a `getAllSite()` metódus lesz. Ez egy listával fog visszatérni, ugyanis az összes állatkertet szeretnénk kikérni. A repository a **findAll** metódussal kéri ki az adott adattábla tartalmát. Ha azt szeretnénk, hogy id alapján találja meg az adott állatkertet, akkor a **findById** metódust használjuk, ami kéri tőlünk az adott Site azonosítóját. Ha nem találja az állatkertet,

eldob egy kivételt, ami közli velünk, hogy nem találja az adott állatparkot. Ez a függvény természetesen egy Site-tal tér vissza.

Az Update-tel lesz talán a legtöbb gondunk. Ez egy Site-tal tér vissza és paraméterként is egy Site-ot vár, azt a Site-ot, amire szeretnénk módosítani a meglévőt, és a meglévő Site id-ját is bele kell írunk a függvényparaméterek közé. A megvalósításhoz szükségünk lesz egy id-val történő keresésre, amihez a findById-t használjuk fel ismét. Ha megtalálta a nekünk tökéletes Site-ot, akkor elkezdődhet annak módosítása. A getter illetve setter metódusokkal tudjuk manipulálni a változóinkat. Fontos, hogy mit settelünk be. Jelen esetben az updatedSite változónak hívjuk meg a setterjét, aminek a paraméterként megadott Site getterjét adjuk be. A sorrend is számít. Fontos az is, hogy amiket módosítottunk el is kell menteni az adatbázisba, máskülönben nem lesz nyoma a frissítésünknek. Erre szolgál a save metódus, ezzel térünk vissza.

A Delettel már könnyű dolgunk lesz. Ez egy void metódus lesz, aminek a paramétere egyedül az adott Site azonosítója. Megkeressük id alapján az állatkertet, majd ráhívjuk a **delete** metódust, aminek a paramétere a megtalált Site, és kitöröljük a sorból a rekordot.

9.5. Controller

```
1 package com.zoo.api.Controller;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.DeleteMapping;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.PathVariable;
9 import org.springframework.web.bind.annotation.PostMapping;
10 import org.springframework.web.bind.annotation.PutMapping;
11 import org.springframework.web.bind.annotation.RequestBody;
12 import org.springframework.web.bind.annotation.RequestMapping;
13 import org.springframework.web.bind.annotation.ResponseStatus;
14 import org.springframework.web.bind.annotation.RestController;
15 import org.springframework.http.HttpStatus;
16 import com.zoo.api.Models.Site;
17 import com.zoo.api.Modules.SiteModules;
18
19 @RestController
20 @RequestMapping("/sites")
21 public class SiteController {
22
23     @Autowired
24     private SiteModules siteModule;
25
26     @PostMapping
27     @ResponseStatus(HttpStatus.CREATED)
28     public Site create(@RequestBody Site site) {
29         return siteModule.storeSite(site);
30     }
31
32     @GetMapping
33     @ResponseStatus(HttpStatus.OK)
34     public List<Site> read() {
35         return siteModule.getAllSite();
36     }
37
38     @GetMapping("/{id}")
39     @ResponseStatus(HttpStatus.OK)
40     public Site read(@PathVariable int id) {
41         return siteModule.getSiteById(id);
42     }
43
44     @DeleteMapping("/{id}")
45     @ResponseStatus(HttpStatus.OK)
46     public void delete(@PathVariable int id) {
47         siteModule.deleteSite(id);
48     }
49
50     @ResponseStatus(HttpStatus.OK)
51     @PutMapping("/{id}")
52     public Site update(@PathVariable int id, @RequestBody Site site) {
53         return siteModule.updateSite(site, id);
54     }
55 }
56
```

Ahogy már írtam, a controller felel azért, hogy a szerver ki is küldje az adatokat az API-n keresztül. A **@RestController** annotációval jelöljük, hogy egy controller classt hozunk létre. A **@RequestMapping** jelöli, hogy mi lesz az URL címe ezeknek a függvényeknek. Szintén Autowired lesz a module változó, automatikusan kapja meg az értéket, nincs szükség konstruktor létrehozására.

@PostMapping-gel fogjuk jelölni a **POST HTTP Request**-et. **@ResponseStatus** az adatbázis kezelőnek adja vissza a szerver státuszát, amikor megtörténik a http kérés. Ez opcionális. Jelen esetben CREATED, azaz a szerver létrehozta az adattáblában az új rekordot. Maga a create függvény hasonlít

Az update függvény a PUT ígét használja, itt a **@PutMapping**-gel adjuk meg, hogy egy update metódust akarunk kivinni az API-ra. Ennek két paramétere lesz, egy PathVariable, ami id és egy RequestBody, ami pedig a Site, amire frissíteni akarunk. Felhívódik a module updateSite metódusa és ezzel is térünk vissza.

Indítsuk el az applikációnkat, futtassuk le a main metódust.

```

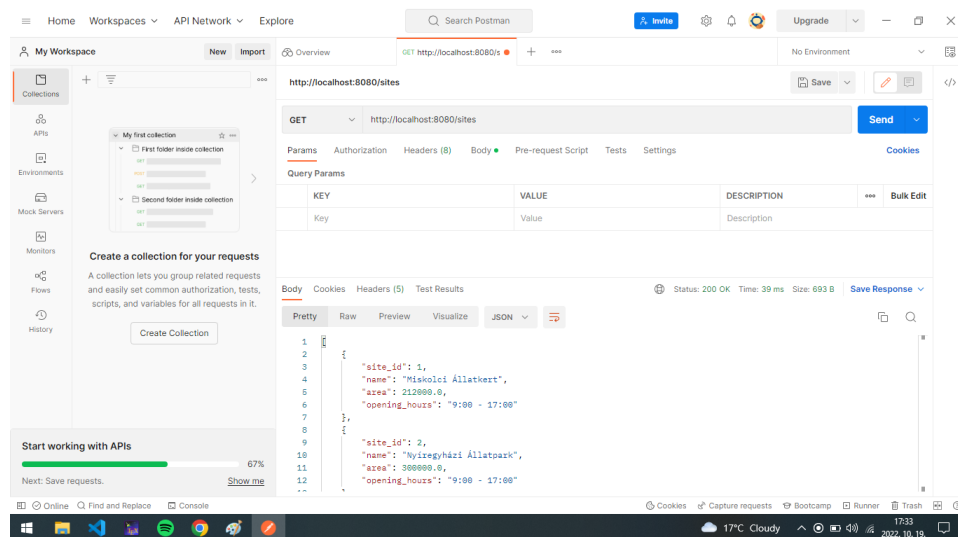
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER SQL CONSOLE
2022-10-19 16:54:30.750 INFO 5064 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform imple
mentation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-10-19 16:54:30.875 INFO 5064 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFacto
ry for persistence unit 'default'
2022-10-19 16:54:33.630 WARN 5064 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled
by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disab
le this warning
2022-10-19 16:54:35.485 INFO 5064 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (h
ttp) with context path ''
2022-10-19 16:54:35.719 INFO 5064 --- [main] com.zoo.api.ApiApplication : Started ApiApplication in 24.363 s
econds (JVM running for 26.238)

```

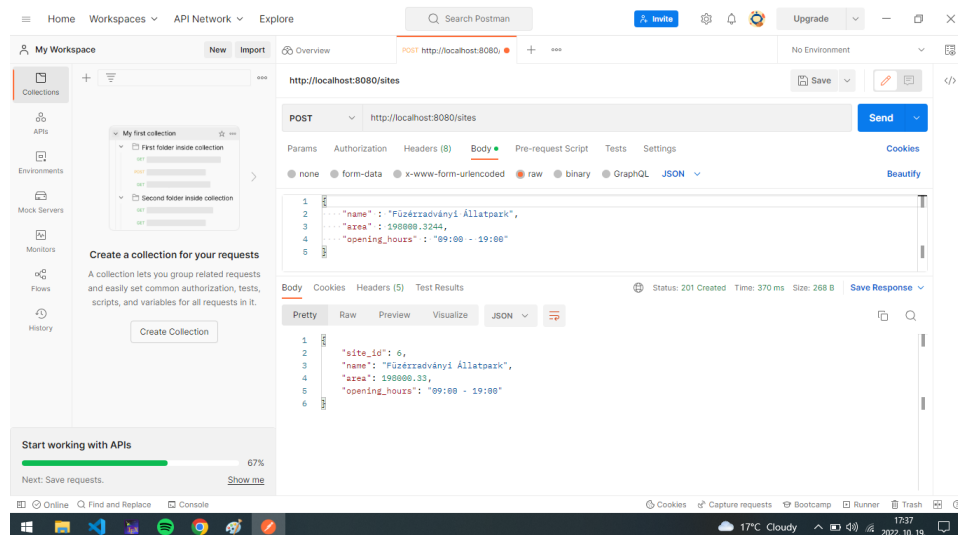
Egy kis idő elteltével el is indul a szerver, amit a terminálban jelez nekünk a program. Üssük be a keresőbe az URL cím helyre, hogy **localhost:8080/sites**. Ha mindent jól csináltunk, ki kell írnia az adattábla elemeit. Ha igényeljük, letölthetünk egy bővítményt a böngészőnkbe, ami a kiírt adatokat JSON formátummá alakítja hasonlóképpen:

```
[
  {
    "site_id": 1,
    "name": "Miskolci Állatkert",
    "area": 212000,
    "opening_hours": "9:00 - 17:00"
  },
  {
    "site_id": 2,
    "name": "Nyíregyházi Állatpark",
    "area": 300000,
    "opening_hours": "9:00 - 17:00"
  },
  {
    "site_id": 3,
    "name": "Debreceni Állatkert és Vidámpark",
    "area": 170000,
    "opening_hours": "9:00 - 15:30"
  },
  {
    "site_id": 4,
    "name": "Kittenberg Kálmán Állatkert és Botanikus kert, Veszprém",
    "area": 170501,
    "opening_hours": "9:00 - 16:00"
  },
  {
    "site_id": 5,
    "name": "Fővárosi Állat- és Növénykert",
    "area": 184001,
    "opening_hours": "9:00 - 17:30"
  }
]
```

Ahhoz, hogy írni, frissíteni és törölni tudjunk az adattáblából kell egy szoftver, amivel a HTTP kéréseket lehet tesztelni. Most a Postmant fogjuk használni a kérések lekezelésére.



Láthatjuk, hogy a Postman-ben is le tudjuk kérni az adattábla adatait. Kiválasztjuk az ígét, beírjuk az URL címet majd pedig elküldjük a HTTP kérést a szervernek, ami visszaadja nekünk az alábbi bodyt. Most vigyünk fel egy rekordot az adattáblába.

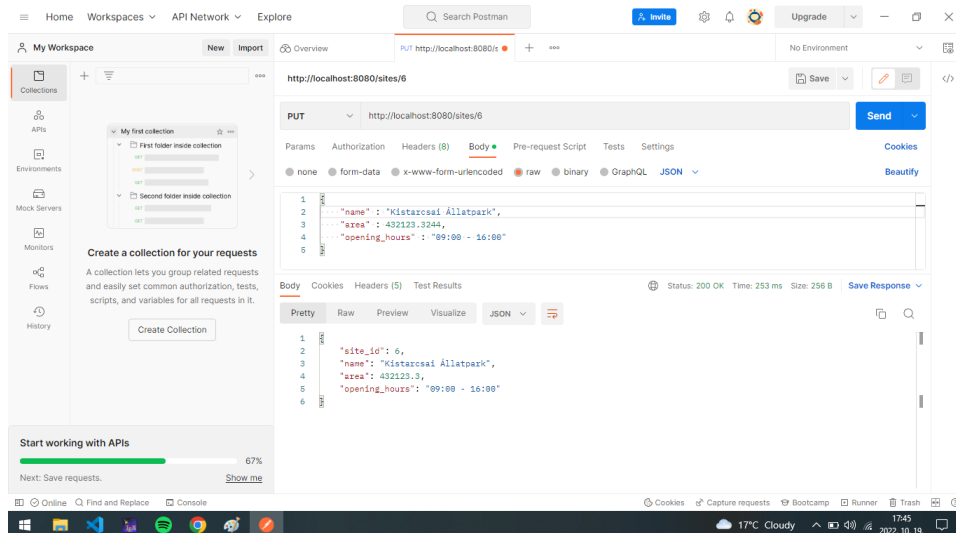


Most a POST verb-öt választjuk ki, az URL ugye ugyan az marad, itt nem változik semmi, csak a HTTP kérés ígéje. Lenyitjuk a paramétereket, kikeressük a bodyt, és raw JSON objektumként felvesszünk egy új rekordot a táblába. Ha mindent jól csináltunk (helyes sorrendben vettük fel a mezőket, helyes változótypusokkal dolgoztunk és a JSON objektumot is helyesen írtuk), akkor egy 201 es státusszal tér vissza a szerver, ami azt jelenti, hogy sikeres volt a HTTP kérés, felvittük a rekordot a táblába. Ellenőrizzük is le!

```
{
  "site_id": 6,
  "name": "Füzérradványi Állatpark",
  "area": 198000,
  "opening_hours": "09:00 - 19:00"
}
```

Ha ráfrissítünk az oldalra, akkor láthatjuk, hogy sikeres volt az adatfelvitel.

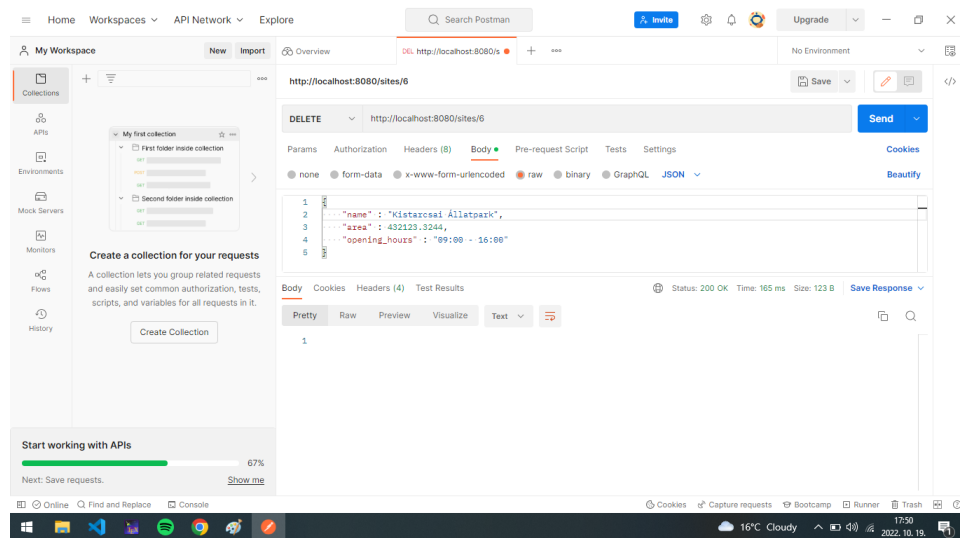
Nézzünk egy PUT metódust is. Ilyenkor a PUT igét kell kiválasztani felül, majd az URL marad az eddigi, viszont mögé kell tennünk azt az id-t, amit meg fogunk majd változtatni. Ezután megírjuk a bodyt.



Egy 200-as státusszal tér vissza a szerver, ellenőrizzük le a sikeres módosítást.

```
{  "site_id": 6,  "name": "Kistarcsai Állatpark",  "area": 432123,  "opening_hours": "09:00 - 16:00"}
```

Nem utolsó sorban a DELETE metódust is leteszteljük. Ennek hasonló lesz az URL-je az előzőhöz, itt is megadjuk az id-t a kérés paramétereiként.



A kéréssel eltűnt a 6-os sorszámú állatpark a sorból.

```
[
  {
    "site_id": 1,
    "name": "Miskolci Állatkert",
    "area": 212000,
    "opening_hours": "9:00 - 17:00"
  },
  {
    "site_id": 2,
    "name": "Nyíregyházi Állatpark",
    "area": 300000,
    "opening_hours": "9:00 - 17:00"
  },
  {
    "site_id": 3,
    "name": "Debreceni Állatkert és Vidámpark",
    "area": 170000,
    "opening_hours": "9:00 - 15:30"
  },
  {
    "site_id": 4,
    "name": "Kittenberg Kálmán Állatkert és Botanikus kert, Veszprém",
    "area": 170501,
    "opening_hours": "9:00 - 16:00"
  },
  {
    "site_id": 5,
    "name": "Fővárosi Állat- és Növénykert",
    "area": 184001,
    "opening_hours": "9:00 - 17:30"
  }
]
```

Természetesen ezt a folyamatot minden adattáblára meg kell csinálni, így áll össze a teljes Backend service, aminek a folytatása, hogy kirakjuk az internetre egy Kubernetes felhőre akár, és ez a felhő fog futni majd a felhasználó gépén is, aki a Frontenden keresztül jut majd el a Backendig.