DOKUMENTÁCIÓ

Web technológiák 2

Készítette: **Nyíri Beáta**

Neptunkód: **I40FDC**

Dátum: **2023.06.20.**

Tartalom

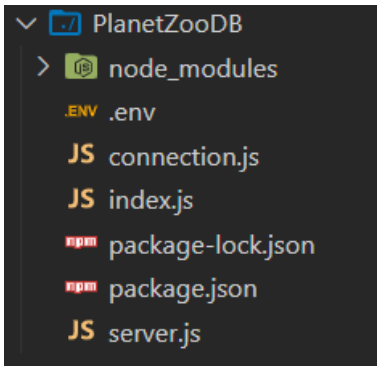
A feladat leírása

A beadandóm témájául ugyanazt választottam, amit a Web technológiák 1 beadandómhoz is, ez pedig az egyik kedvelt videójátékom, a Planet Zoo. A feladathoz a játékban szereplö állatokhoz készítettem egy nyílvántartó rendszert.

Ehhez a kiadott feladat leírásában feltüntetetett technológiákat alkalmaztam (Angular, NodeJS) az adatbáziskezelön kívül. Én ugyanis MongoDB helyett MySQL-t használtam. A programot Visual Studio Code-ban írtam, Postman és Mailinator segítségével teszteltem.

Kezdeti beállítások, adatbázis létrehozása

A programot a backend megírásával kezdtem. Feltelepítettem a szükséges csomagokat, létrehoztam egy adatbázist, majd létrehoztam a kapcsolatot az adatbázissal.



A .env és a connection.js fájl segítségével kapcsolódtam az adatbázishoz, amit MySQL 8.0 Command Line Client-tel hoztam létre.

PlanetZooDB > .ENV .env

```
1   //Server
2   PORT = 8080
3
4   //Connection
5   DB_PORT = 3306
6   DB_HOST =localhost
7   DB_USERNAME = root
8   DB_PASSWORD = 1234
9   DB_NAME = planetzoodb
```

PlanetZooDB > JS connection.js > ...

```
1   const mysql = require('mysql');
2   require('dotenv').config();
3
4   var connection = mysql.createConnection({
5       port: process.env.DB_PORT,
6       host: process.env.DB_HOST,
7       user: process.env.DB_USERNAME,
8       password: process.env.DB_PASSWORD,
9       database: process.env.DB_NAME
10  });
11
12  connection.connect((err) =>{
13      if(!err){
14          console.log("Connected");
15      }
16      else{
17          console.log(err);
18      }
19  });
20
21  module.exports = connection;
```

Azért pedig, hogy ne manuálisan kelljen frissítenem a servert minden változtatásnál, nodemon-t használtam. Ehhez a package.json fájlban hoztam létre egy új “start” scriptet.

PlanetZooDB > package.json > ...

```
1   {
2     "name": "planetzoodb",
3     "version": "1.0.0",
4     "description": "",
5     "main": "index.js",
      ▷ Debug
6     "scripts": {
7       "test": "echo \"Error: no test specified\" && exit 1",
8       "start": "nodemon server.js"
9     },
10    "author": "",
11    "license": "ISC",
12    "dependencies": {
13      "cors": "^2.8.5",
14      "dotenv": "^16.3.1",
15      "express": "^4.18.2",
16      "mysql": "^2.18.1"
17    },
18    "devDependencies": {
19      "nodemon": "^2.0.22"
20    }
21  }
```

Felhasználók kezelése

Az oldalt úgy építettem fel, hogy kétféle bejelentkezési lehetőség legyen. Be lehet lépni adminként és egyszerü felhasználóként. A különbség az, hogy az admin minden menüpontot lát és elér, a felhasználók pedig nem mindet, valamint ha egy felhasználó be szeretne lépni (Sign In), akkor azt az adminnak jóvá kell hagynia.

Létrehoztam egy table.sql fájlt, amiben a user tábla létrehozásához és kezeléséhez szükséges parancsok vannak tárolva.

PlanetZooDB > table.sql

```sql
create table user(
    id int primary key AUTO_INCREMENT,
    name varchar(250),
    email varchar(50),
    password varchar(250),
    status varchar(20),
    role varchar(20),
    UNIQUE (email)
);

insert into user(name,email,password,status,
role) values ('Admin','admin@gmail.com','admin',
'true','admin');
```

Regisztráció

Bevezettem a route-okat és elsőként a regisztrációhoz készítettem egy API-t. A user.js fájban az ehhez szükséges kód van. Az email alapján ellenörzöm a felhasználókat, ha egy email már szerepel az adatbázisban, akkor ezt jelzi a program egy üzenettel, ha pedig még nem szerepel és jó adatokat adott meg a regisztralni kívánó felhasználó, akkor az adatai bekerülnek az adatbázisba és egy üzenet jelenik meg, miszerint a regisztráció sikeres volt.

PlanetZooDB > routes > JS user.js > ...

```js
const express = require('express');
const connection = require('../connection');
const router = express.Router();

router.post('/signup',(req,res) =>{
    let user = req.body;
    query = "select email,password,role,status from user where email=?"
    connection.query(query,[user.email],(err,results) =>{
        if(!err){
            if(results.length <=0){
                query = "insert into user(name,email,password,status,role) values(?,?,?,'false','user')";
                connection.query(query,[user.name,user.email,user.password],(err,results) =>{
                    if(!err){
                        return res.status(200).json({message: "Successfully Registered"});
                    }
                    else{
                        return res.status(500).json(err);
                    }
                })
            }
            else{
                return res.status(400).json({message: "Email Already Exists."});
            }
        }
        else{
            return res.status(500).json(err);
        }
    })

})

module.exports = router;
```