**BACKEND**

1. **Munkamappa kiválasztása / létrehozása**
2. **Terminál megnyitása**
   1. „**npx express-generator --no-view backend**” telepítése (a –no-view-ot akkor használjuk, ha nem készülünk használni a frontendet)
   2. „backend” mappába belépés (term: cd backend / open in integrated terminal) > „**npm install**” > localhost:3000-es porton fut
   3. „**npm install nodemon mysql2 cors**” mellékletek telepítése
      1. Módosítások:
         1. **bin/www.js** állományban a port átírása: 8080
         2. **package.json** állományban nodemon elhelyezése
3. **Állományok létrehozása**
   1. **app.js** állományban a cors kódrész hozzáadása „var app = express();” alá:

**const cors = require('cors');**

**var app = express();**

**var corsOptions = {**

**origin: "http://localhost:3000"**

**}**

**app.use(cors(corsOptions))**

* 1. **app.js** állományban „**var adatokRouter = require('./routes/adatok');**” kód létrehozása a usersRouter alá és az „app.use('/users', usersRouter);” alá „**app.use('/adatok', adatokRouter);**” létrehozása
  2. **bin** mappa létrehozása > „dboperations.js” állományok létrehozása
  3. **routes/adatok.js** állomány létrehozása

**dboperations.js**

const mysql = require('mysql2/promise');

// Connection pool létrehozása

const pool = mysql.createPool({

  host: 'localhost',

  user: 'root',       // adatbázis felhasználónév

  password: 'root',       // adatbázis jelszó

  database: 'ora', // Az adatbázis neve

  waitForConnections: true,

  connectionLimit: 10, // Max. 10 egyidejű kapcsolat

  queueLimit: 0 //korlátlan várakozási sor

});

// Összes adat lekérése egy adott táblából

async function getAllData() {

  try {

    const [rows] = await pool.execute("SELECT \* FROM marka");

    return rows;

  } catch (error) {

    console.error("Hiba a getAllData() futtatása közben:", error);

    throw error;

  }

}

async function getFilterData(limit, offset) {

    try {

      const sql = `SELECT \* FROM marka LIMIT ? OFFSET ?`;

      const [rows] = await pool.query(sql, [limit, offset]);

      return rows;

    } catch (error) {

      console.error("Hiba a getAllDatalimit() futtatása közben:", error);

      throw error;

    }

  }

  async function getDataById(id) {

    try {

      const [rows] = await pool.execute("SELECT \* FROM marka where markaaz= ?", [id]);

      return rows[0];

    } catch (error) {

      console.error("Hiba a getDataByid() futtatása közben:", error);

      throw error;

    }

  }

  //szűrő

  async function getFilterrData(filters){

    let sql = "SELECT \* FROM oralekerdezes";

    const conditions = [];

    const values = [];

    if(filters.marka){

      conditions.push("marka = ?");

      values.push(filters.marka);

    }

    if(filters.tipus){

      conditions.push("tipus = ?");

      values.push(filters.tipus);

    }

    if(filters.szijszine){

      conditions.push("szijszine = ?");

      values.push(filters.szijszine);

    }

    if(conditions.length > 0){

      sql += " WHERE " + conditions.join(" AND ");

    }

    console.log("sql:", sql);

    try{

      const [rows] = await pool.query(sql, values);

      return rows;

    }

    catch(error){

      console.error("Hiba a getFilterrData() futtatása közben:", error);

      throw error;

    }

  }

module.exports = {

  getAllData,

  getFilterData,

  getDataById,

  getFilterrData

};

**adatok.js**

// Példa route (pl. routes/adatok.js)

const express = require('express');

const router = express.Router();

const db = require('../db/dboperations');

// Összes adat egy adott táblából

//példa url: http://localhost:8080/adatok

router.get('/', async (req, res) => {

  try {

    const adatok = await db.getAllData();

    res.json(adatok);

  } catch (error) {

    res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

  }

});

//http://localhost:8080/adatok/filtered?limit=2&offset=0

router.get('/filtered1', async (req, res) => {

  let limit= parseInt(req.query.limit, 10);

  let offset= parseInt(req.query.offset, 10);

    try {

        const results = await db.getFilterData(limit, offset);

            res.json(results);

    } catch (error) {

      res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

    }

  });

  //http://localhost:8080/adatok/2

  router.get('/azon:id', async (req, res) => {

    try {

      const id= req.params.id;

      const adatok = await db.getDataById(id);

      if(adatok){

        res.json(adatok);

      }

    } catch (error) {

      res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

    }

  });

  //http://localhost:8080/adatok/filtered?marka=ford&tipus=autó&szijszine=sötét

  router.get('/filtered2', async (req, res) => {

    try{

      const filters={

        marka: req.query.marka,

        tipus: req.query.tipus,

        szijszine: req.query.szijszine

      };

      const results = await db.getFilterrData(filters);

      res.json(results);

    }

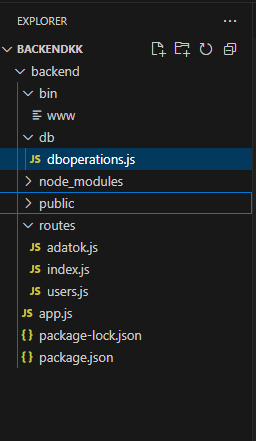
    catch (error) {

      res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

    }

  });

module.exports = router;



dboperations.js

const mysql = require('mysql2/promise');

// Connection pool létrehozása

const pool = mysql.createPool({

  host: 'localhost',

  user: 'root',       // adatbázis felhasználónév

  password: 'root',       // adatbázis jelszó

  database: 'ora', // Az adatbázis neve

  waitForConnections: true,

  connectionLimit: 10, // Max. 10 egyidejű kapcsolat

  queueLimit: 0 //korlátlan várakozási sor

});

// Összes adat lekérése egy adott táblából

async function getAllData() {

  try {

    const [rows] = await pool.execute("SELECT \* FROM marka");

    return rows;

  } catch (error) {

    console.error("Hiba a getAllData() futtatása közben:", error);

    throw error;

  }

}

async function getFilterData(limit, offset) {

    try {

      const sql = `SELECT \* FROM marka LIMIT ? OFFSET ?`;

      const [rows] = await pool.query(sql, [limit, offset]);

      return rows;

    } catch (error) {

      console.error("Hiba a getAllDatalimit() futtatása közben:", error);

      throw error;

    }

  }

  async function getDataById(id) {

    try {

      const [rows] = await pool.execute("SELECT \* FROM marka where markaaz= ?", [id]);

      return rows[0];

    } catch (error) {

      console.error("Hiba a getDataByid() futtatása közben:", error);

      throw error;

    }

  }

  //szűrő

  async function getFilterrData(filters){

    let sql = "SELECT \* FROM oralekerdezes";

    const conditions = [];

    const values = [];

    if(filters.marka){

      conditions.push("marka = ?");

      values.push(filters.marka);

    }

    if(filters.tipus){

      conditions.push("tipus = ?");

      values.push(filters.tipus);

    }

    if(filters.szijszine){

      conditions.push("szijszine = ?");

      values.push(filters.szijszine);

    }

    if(conditions.length > 0){

      sql += " WHERE " + conditions.join(" AND ");

    }

    console.log("sql:", sql);

    try{

      const [rows] = await pool.query(sql, values);

      return rows;

    }

    catch(error){

      console.error("Hiba a getFilterrData() futtatása közben:", error);

      throw error;

    }

  }

module.exports = {

  getAllData,

  getFilterData,

  getDataById,

  getFilterrData

};

adatok.js

// Példa route (pl. routes/adatok.js)

const express = require('express');

const router = express.Router();

const db = require('../db/dboperations');

// Összes adat egy adott táblából

//példa url: http://localhost:8080/adatok

router.get('/', async (req, res) => {

  try {

    const adatok = await db.getAllData();

    res.json(adatok);

  } catch (error) {

    res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

  }

});

//http://localhost:8080/adatok/filtered1?limit=2&offset=0

router.get('/filtered1', async (req, res) => {

  let limit= parseInt(req.query.limit, 10);

  let offset= parseInt(req.query.offset, 10);

    try {

        const results = await db.getFilterData(limit, offset);

            res.json(results);

    } catch (error) {

      res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

    }

  });

  //http://localhost:8080/adatok/2

  router.get('/azon/:id', async (req, res) => {

    try {

      const id= req.params.id;

      const adatok = await db.getDataById(id);

      if(adatok){

        res.json(adatok);

      }

    } catch (error) {

      res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

    }

  });

  //http://localhost:8080/adatok/filtered?marka=ford&tipus=autó&szijszine=sötét

  router.get('/filtered2', async (req, res) => {

    try{

      const filters={

        marka: req.query.marka,

        tipus: req.query.tipus,

        szijszine: req.query.szijszine

      };

      const results = await db.getFilterrData(filters);

      res.json(results);

    }

    catch (error) {

      res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

    }

  });

module.exports = router;

app.js

var express = require('express');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

var indexRouter = require('./routes/index');

var usersRouter = require('./routes/users');

const adatokRouter = require('./routes/adatok');

var app = express();

const cors = require('cors')

var corsOptions = {

    origin: "http://localhost:3000"

}

app.use(cors(corsOptions))

app.use(logger('dev'));

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', indexRouter);

app.use('/users', usersRouter);

app.use('/adatok', adatokRouter);

module.exports = app;

if (filters.marka){conditions.push("Marka LIKE ?"); values.push(`%${filters.marka}%`)} márka részletes keresés

if (filters.marka){conditions.push("Termekar >= ?"); values.push(number(filters.minar))} min ár

if (filters.marka){conditions.push("Termekar <= ?"); values.push(number(filters.minar))} max keresés

del, put stb adatok.js

const express = require('express');

const router = express.Router();

const db = require('../db/dboperations'); // Importáljuk az adatbázisműveleteket

// GET - Összes adat lekérése

router.get('/', async (req, res) => {

try {

const adatok = await db.getAllData();

res.json(adatok);

} catch (error) {

res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

}

});

// GET - Szűrt adatok lekérése (limit és offset alapján)

router.get('/filtered1', async (req, res) => {

let limit = parseInt(req.query.limit, 10);

let offset = parseInt(req.query.offset, 10);

try {

const results = await db.getFilterData(limit, offset);

res.json(results);

} catch (error) {

res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

}

});

// GET - Adatok ID alapján

router.get('/azon:id', async (req, res) => {

try {

const id = req.params.id;

const adatok = await db.getDataById(id);

if (adatok) {

res.json(adatok);

} else {

res.status(404).json({ error: 'Nem található adat ezzel az ID-val.' });

}

} catch (error) {

res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

}

});

// GET - Szűrés több paraméter alapján

router.get('/filtered2', async (req, res) => {

try {

const filters = {

marka: req.query.marka,

tipus: req.query.tipus,

szijszine: req.query.szijszine,

};

const results = await db.getFilterrData(filters);

res.json(results);

} catch (error) {

res.status(500).json({ error: 'Hiba történt az adatok lekérésekor.' });

}

});

// POST - Új adat hozzáadása

router.post('/', async (req, res) => {

try {

const { marka, tipus, szijszine } = req.body;

const id = await db.insertData(marka, tipus, szijszine);

res.status(201).json({ message: 'Új adat hozzáadva.', id });

} catch (error) {

res.status(500).json({ error: 'Hiba történt az adat hozzáadásakor.' });

}

});

// PUT - Adat módosítása ID alapján

router.put('/:id', async (req, res) => {

try {

const id = req.params.id;

const { marka, tipus, szijszine } = req.body;

const updatedRows = await db.updateData(id, marka, tipus, szijszine);

if (updatedRows > 0) {

res.json({ message: 'Adat frissítve.', id });

} else {

res.status(404).json({ error: 'Nem található adat ezzel az ID-val.' });

}

} catch (error) {

res.status(500).json({ error: 'Hiba történt az adat frissítésekor.' });

}

});

// DELETE - Adat törlése ID alapján

router.delete('/:id', async (req, res) => {

try {

const id = req.params.id;

const deletedRows = await db.deleteData(id);

if (deletedRows > 0) {

res.json({ message: 'Adat törölve.', id });

} else {

res.status(404).json({ error: 'Nem található adat ezzel az ID-val.' });

}

} catch (error) {

res.status(500).json({ error: 'Hiba történt az adat törlésekor.' });

}

});

module.exports = router;

dboperations.js

const mysql = require('mysql2/promise');

// Connection pool létrehozása

const pool = mysql.createPool({

host: 'localhost',

user: 'root', // adatbázis felhasználónév

password: 'root', // adatbázis jelszó

database: 'ora', // Az adatbázis neve

waitForConnections: true,

connectionLimit: 10, // Max. 10 egyidejű kapcsolat

queueLimit: 0, // Korlátlan várakozási sor

});

// Összes adat lekérése

async function getAllData() {

const [rows] = await pool.execute("SELECT \* FROM marka");

return rows;

}

// Adatok szűrése limit és offset alapján

async function getFilterData(limit, offset) {

const sql = "SELECT \* FROM marka LIMIT ? OFFSET ?";

const [rows] = await pool.query(sql, [limit, offset]);

return rows;

}

// Adatok lekérése ID alapján

async function getDataById(id) {

const [rows] = await pool.execute("SELECT \* FROM marka WHERE markaaz = ?", [id]);

return rows[0];

}

// Adatok szűrése több paraméter alapján

async function getFilterrData(filters) {

let sql = "SELECT \* FROM oralekerdezes";

const conditions = [];

const values = [];

if (filters.marka) {

conditions.push("marka = ?");

values.push(filters.marka);

}

if (filters.tipus) {

conditions.push("tipus = ?");

values.push(filters.tipus);

}

if (filters.szijszine) {

conditions.push("szijszine = ?");

values.push(filters.szijszine);

}

if (conditions.length > 0) {

sql += " WHERE " + conditions.join(" AND ");

}

const [rows] = await pool.query(sql, values);

return rows;

}

// Új adat beszúrása

async function insertData(marka, tipus, szijszine) {

const sql = "INSERT INTO marka (marka, tipus, szijszine) VALUES (?, ?, ?)";

const [result] = await pool.execute(sql, [marka, tipus, szijszine]);

return result.insertId;

}

// Adatok módosítása ID alapján

async function updateData(id, marka, tipus, szijszine) {

const sql = "UPDATE marka SET marka = ?, tipus = ?, szijszine = ? WHERE markaaz = ?";

const [result] = await pool.execute(sql, [marka, tipus, szijszine, id]);

return result.affectedRows;

}

// Adatok törlése ID alapján

async function deleteData(id) {

const sql = "DELETE FROM marka WHERE markaaz = ?";

const [result] = await pool.execute(sql, [id]);

return result.affectedRows;

}

module.exports = {

pool, // A pool exportálása is, ha közvetlenül hivatkozni kell

getAllData,

getFilterData,

getDataById,

getFilterrData,

insertData,

updateData,

deleteData,

};