**MongoDB**

**Зад 1. Зареждане на данните от MongoDB**

В тази задача, ще заменим json server-a от предишното приложение с express сървър и MongoDB. За целта, изпълнете следните задачи:

1. Изтеглете **ZIP** версията на MongoDB Community Server (от менюто Products на <https://www.mongodb.com/>) и **ZIP** версията на Compass (от менюто Products на <https://www.mongodb.com/>)
2. Разархивирайте двата архива. В папката на Community Server създайте папката databases. В Community Server, отворете папката bin и от нея изпълнете следната конзолна команда, за да стартирате mongodb сървъра:

mongod.exe --dbpath "../databases/"

1. Стартирайте Compass и се свържете към базата от данни (MongoDB сървъра)
2. Създайте нова база от данни с името excursions.
3. Добавете колекции за отделните групи от данни – напр. екскурзии, потребители и др.
4. След като настроихме базата от данни, нека да модифицираме приложението. За целта, копирайте папката на проекта, за да запазим старата работеща версия (ако искате, работете върху старата версия на проекта).
5. Отворете новото копие и към него добавете библиотеките на express и mongodb, като изпълните следните команди:

npm install express

npm install nodemon

npm install mongodb

npm install cors

1. Във файлът package.json, в секцията скриптове, добавете следния крипт:

"start-express": "nodemon start-express-with-mongodb.js",

1. В папката на проекта, създайте файл с името start-express-with-mongodb.js
2. Във файла start-express-with-mongodb.js добавете следния код:

// Необходими библиотеки:

// npm install express

// npm install nodemon

// npm install cors     // Иначе трябва да се ползва res.setHeader(' Access-Control-Allow-Origin','\*')

//

// Стартиране на express:

// npm run start-express

// Скрипта изглежда така: "start-express": "nodemon start-express.js",

import express from 'express';

import cors from 'cors';

import { getSingleUser, test } from './mongoDB.js';

//import { createClient } from 'redis';

var app = express();        // създаване нанова инстанция

app.use(express.json());    // активиране на поддръжката на json

app.use(cors());            // Предотвратява CORS грешки

app.listen(3001);       // задаване на порт, на който да работи сървърът

// обработване на заявки

app.get('/', async function(req, res){

    // Изпращане на json обект към клиента

    let obj = {name:'Alex', age:20};

    res.setHeader('Content-Type','application/json')

    res.send(obj);

});

// обработване на заявки

app.get('/page2/', async function(req, res){

    res.setHeader('Content-Type','text/html')

    res.send(`

                <h1>Page 2</h1>

                <hr>

                    <a href='/'>Към начало</a>

                <hr>

            `);

});

// прочитане на параметри от url адреса

app.get('/:id?', function(req, res){

    res.send('Полъченото id е: ' + req.params.id);

});

// получаване на json съдържание през тялото на post заяка с обекта:

//   {

//      "user":"Alex",

//      "age":10

//  }

app.post('/', function(req, res){

    console.log(req.header("Content-type"));

    console.log(req.body);

    console.log(req.body.user);

    res.send("OK");

    //res.send(req.body);    // echo the result back

 });

app.put('/', function(req, res){

    res.send('PUT заявка');

 });

 app.delete('/', function(req, res){

    res.send('DELETE заявка');

 });

app.get('/mongo/mongodb', async function(req, res){

    let user = await getSingleUser();

    res.setHeader('Content-Type','application/json')

    res.send(user);

});

app.get('/mongo/test', async function(req, res){

    let user = await test();

    console.log(user)

    res.setHeader('Content-Type','application/json')

    res.send(user);

});

// Прихващане на всички невалидни адреси:

app.get('\*', function(req, res){

    res.send('Грешка 404. Страницата не е намерена.');

 });

1. Във Visual Studio Code отворете нов терминал и стартирайте скрипта start-express като в терминала изпълните следната команда:

npm run start-express

1. Променете адреса на fetch() заявката, за да зареди express. Aреса е:

localhost:3001

1. Променете съдържанието на файла start-express-with-mongodb.js, за да съдържа необходимите уеб услуги. За целта, прегледайте съдържанието на следния файл с името „mongoDB.js“:

Забалежка: може да прегледате и документацията:

https://www.mongodb.com/docs/drivers/node/current/quick-start/download-and-install/

//const { MongoClient } = require("mongodb");

import {MongoClient} from 'mongodb';

// Фунцкията намира и връща един потребител

export async function getSingleUser() {

    // Replace the uri string with your connection string.

    let uri = "mongodb://localhost:27017";

    let client = new MongoClient(uri);

    let user=null;

    try

    {

        let database = client.db('mydatabase');

        let users = database.collection('users');

        // Query for a movie that has the title 'Back to the Future'

        const query = { points: 100 };

        user = await users.find(query).toArray();

    }

    finally

    {

        // Ensures that the client will close when you finish/error

        await client.close();

    }

    return user;

}

export async function test() {

    let uri = "mongodb://localhost:27017";

    let client = new MongoClient(uri);

    let user=null;

    try

    {

        let database = client.db('mydatabase');

        let users = database.collection('users');

        // Намира всъчки потребители с точки на 5 и под 110

        // user = await users.find({points:{

        //                                  $gte: 5,        //https://www.mongodb.com/docs/manual/reference/operator/query/

        //                                  $lt: 110,

        //                              }}).toArray();

        // Намира всъчки потребители с точки на 5 и под 110, и показва само age и name

        // user = await users.find({points:{

        //                                  $gte: 5,

        //                                  $lt: 110,

        //                              }},).project({\_id:false, age:true, name:true}).toArray();

        // Намира всъчки потребители с точки на 5 и под 110 и сортира

        // по точки - нарастващо и по age - намаляващо

        // user = await users.find({points:{

        //                                  $gte: 5,

        //                                  $lt: 110,

        //                              }},{sort:{"points":1, "age":-1}}).toArray();

        // .skip(1)

        // .limit(1)

        // Намира всъчки потребители с точки равни на 5, 15 или 100

        // user = await users.find({points:{

        //                                  $in: [5, 15, 100]

        //                              }},).project({age:false}).toArray();

        // Намира всъчки потребители и ги групира по име

        // user = await users.aggregate([   {

        //                                  $match:{

        //                                              points:{

        //                                                      $gte: 5,

        //                                                      $lt: 110,

        //                                                  }

        //                                          }

        //                              },

        //                              {

        //                                  $group: {

        //                                              \_id: "$name",

        //                                              count: {

        //                                                          $sum: 1,

        //                                                      },

        //                                  },

        //                              }

        //                          ]).toArray();

        // Търси в текстовото поле name за текст "ki"

        //user = await users.find( { "name": {$regex : "ki"}} ).toArray(); // част

        // добавя нов потребител

        // const doc = { name: "Вики", age: 21, points:70 };

        // user = await users.insertOne(doc);

        // добавя няколко потребители

        // const docs = [{ name: "Вики", age: 23, points:71 },

        //            { name: "Вики", age: 24, points:75 }];

        // user = await users.insertMany(docs);

        // Вкъмва нов потребител с id

        // const doc = {\_id:2, name: "Bobi", age: 23, points:51 } ;

        // user = await users.insertOne(doc);

        // изтрива потеребител

        // let doc = {\_id:2} ;

        // let doc = {name:"Вики"} ;

        // user = await users.deleteMany(doc);

        // при обновяване може да се ползват:

        // $set: replaces the value of a field with a specified one

        // $inc: increments or decrements field values

        // $rename: renames fields

        // $unset: removes fields

        // $mul: multiplies a field value by a specified number

        // обновява точките на документ с id "1" - текстово id

        let filter = {\_id:"1"} ;

        let updateDocument = {

            $set: {

               points: 17

            }

        };

        user = await users.updateOne(filter, updateDocument);

        // // updateMany() -> обновява много документи

        // // replaceOne() -> заменя 1-н документ

    }

    finally

    {

        // Ensures that the client will close when you finish/error

        await client.close();

    }

    return user;

}