

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 3

з дисципліни “Основи web-програмування”

тема “Публікація веб-додатку”

|  |  |  |
| --- | --- | --- |
| Виконала  студентка Ⅱ курсу  групи КП-92  Шестакова Олександра Вячеславівна  (*прізвище, ім’я, по батькові*) |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Вивчити основні принципи асинхронного програмування в JavaScript.

Навчитись асинхронно взаємодіяти з базою даних.

Налаштувати взаємодію з віддаленою базою даних та сховищем медіа.

Підготувати і опублікувати веб-сервіс в мережі Інтернет.

**Постановка завдання**

Змоделювати за допомогою GUI клієнта бази даних об'єкти сутностей із попередніх лабораторних робіт та додати нову композиційну сутність відповідно до варіанту.

Переписати модуль сховища даних для взаємодії з базою даних. Змінити інтерфейс сховища для використання асинхронних функцій.

Створити Heroku Application для веб-сайту, налаштувати його для доступу до віддаленого сховища медіа.

**Лістинг** нових та змінених **модулів**

|  |
| --- |
| app.js |
| const express = require('express');  const bodyParser = require('body-parser');  const busboyBodyParser = require('busboy-body-parser');  const mstRouter = require('./routes/mstRouter');  const morgan = require('morgan');  const consolidate = require('consolidate');  const path = require('path');  const mustache = require('mustache-express');  //  const UserRepository = require('./repositories/userRepository');  const userRepository = new UserRepository('data/users.json');  const MuseumRepository = require('./repositories/museumRepository');  const museumRepository = new MuseumRepository('data/museums.json');  const config = require('./config');  //  const app = express();  app.use(bodyParser.urlencoded({ extended: true }));  app.use(bodyParser.json());  app.use(busboyBodyParser());  app.use(morgan('dev'));  const expressSwagger = require('express-swagger-generator')(app);  app.use(express.static('./public'));  app.use(express.static('./data'));  app.engine('mst', mustache());  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'mst');  app.get('/', function (req, res) {  res.render('index', {});  });  app.get('/about', function (req, res) {  res.render('about', {});  });  app.use('', mstRouter);  app.use((req, res) => {  res.status(400).send({ message: "Error in route."});  });  app.get('/users/:id', function (req, res) {  const user = userRepository.getUserById(parseInt(req.params.\_id));  res.render('user', { user });  });  app.listen(config.app.port, function() {  console.log('Server is ready');  }); |

|  |
| --- |
| museumsController.js |
| const path = require('path');  const MuseumRepository = require('../repositories/museumRepository');  const museumRepository = new MuseumRepository();  const MediaRepository = require('../repositories/mediaRepository');  const mediaRepository = new MediaRepository(path.resolve(\_\_dirname, '../data/media'));  const Museum = require('../models/museum');  const ReviewRepository = require('../repositories/reviewRepository');  const reviewRepository = new ReviewRepository();  module.exports = {  async getArtMuseums(req, res) {  try {  let page = req.query.page;  let name = req.query.name;  const page\_size = 10;  if (!page) page = 1;  else page = Number(page);  let result = await museumRepository.getMuseumsPaginated(Number(page), page\_size, name);  let museums = result.museums\_res;  //console.log(museums);  let pagesNumber = Number(result.totalPages);  let pages = { currentPage: Number(result.currentPage) };  if (page !== 1) pages.prevPage = page - 1;  if (page !== pagesNumber) pages.nextPage = page + 1;  if (name) pages.namePage = name;  res.status(200).render('museums', { museums: museums, pagesNumber: pagesNumber, pages: pages});  } catch (err) {  console.log(err.message);  res.status(500).send({ museums: null, message: 'Server error.' });  }  },    async getArtMuseumById(req, res) {  //console.log("GET ArtMuseumById"+req.params.\_id);  const museum = await museumRepository.getArtMuseumById(req.params.\_id);  if (museum) {  const reviews = await reviewRepository.getRewiewsByMuseumId(req.params.\_id);  res.status(200).render('museum', { museum: museum, reviews: reviews});  }  else {  res.status(404).send({ museum: null, message: "Museum id is incorrect." });  }  },    async addArtMuseum(req, res) {  const image = await mediaRepository.addMedia(req.files['imageUrl'].data);  //console.log(req.body);  const new\_museum = new Museum(null, req.body.Mname, req.body.country, req.body.founded,  Number(req.body.artistNum), Number(req.body.exhibitNum), image.url);  const newId = await museumRepository.addArtMuseum(new\_museum);  //console.log(newId);  res.redirect('/museums/' + newId);  },    async deleteArtMuseum(req, res) {  //console.log("DELETE " + req.params.\_id);  const deletedId = await museumRepository.deleteArtMuseum(req.params.\_id);  res.redirect('/museums');  },    async updateArtMuseum(req, res) {  if (!req.body)  res.sendStatus(400);  else {  const museum = museumRepository.getArtMuseumById(req.body.\_id);  if (museum) {  museumRepository.updateArtMuseum(req.body);  res.send(museumRepository.getArtMuseumById(req.body.\_id));  res.end();  }  else  res.sendStatus(404);  }  }  }; |

|  |
| --- |
| mediaRepository.js |
| const config = require('../config');  const cloudinary = require('cloudinary');  cloudinary.config({  cloud\_name: config.cloudinary.cloud\_name,  api\_key: config.cloudinary.api\_key,  api\_secret: config.cloudinary.api\_secret  });  class MediaRepository {  async addMedia(media) {  return new Promise((resolve, reject) => {  cloudinary.v2.uploader  .upload\_stream(  { resource\_type: 'raw' },  (err, result) => {  if (err) {  reject(err);  } else {  resolve(result);  }  })  .end(media);  });    }  }  module.exports = MediaRepository; |

|  |
| --- |
| mongoStorage.js |
| const MongoClient = require('mongodb').MongoClient;  const ObjectId = require('mongodb').ObjectId;  const config = require('./config');  const { db: { host, login, password, name } } = config;  const dbName = `${name}`;  const uri = `mongodb+srv://${login}:${password}@${host}/?writeConcern=majority`;  //const uri = "mongodb+srv://User123:123qweasd@lab4.0pfe1.mongodb.net/?writeConcern=majority";  class MongoStorage {    async readItems(collectionName, findQry) {  //console.log("readItemsCall");  let result = null;  const mongoClient = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });  try {    await mongoClient.connect();  const collection = mongoClient.db(dbName).collection(collectionName);  result = await collection.find(findQry).toArray();  //console.log(result);  }  catch (err) {  console.log(err);  }  finally {  //console.log("closingClient")  mongoClient.close();  return result;  }  }  async getItem(collectionName, id) {  let item = null;  const mongoClient = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });  try {  await mongoClient.connect();  const collection = mongoClient.db(dbName).collection(collectionName);  const objectId = new ObjectId(id);  item = await collection.findOne({\_id: objectId});  //console.log(item);  }  catch (err) {  console.log(err);  }  finally {  mongoClient.close();  return item;  }  }  //insert multiple documents  async writeItem(collectionName, item) {  let newItemId = null;  const mongoClient = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });  try {  await mongoClient.connect();  const itemsCollection = mongoClient.db(dbName).collection(collectionName);  console.log("writeItem id: "+item.\_id);  if (!item.\_id){//new entry mode  item.\_id = new ObjectId();    newItemId = (await itemsCollection.insertOne(item)).insertedId;  }  else{ //update existing doc mode  const fltr = {\_id : item.\_id};//define filter to update existing item document  const optns = {upsert: true}; //instruct mongodb to create a doc if it's not mached by filter  newItemId = (await itemsCollection.replaceOne(fltr,item,optns)).upsertedId;  }  }  catch (err) {  console.log(err);  }  finally {  mongoClient.close();  return newItemId;  }  }  async deleteItem(collectionName, id) {  let item = null;  const mongoClient = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });  try {  await mongoClient.connect();  const collection = mongoClient.db(dbName).collection(collectionName);  const objectId = new ObjectId(id);  item = await collection.findOneAndDelete({\_id: objectId});  //console.log(item);  }  catch (err) {  console.log(err);  }  finally {  mongoClient.close();  return item;  }  }  };  module.exports = MongoStorage; |

|  |
| --- |
| config.js |
| require('dotenv').config();  const config = {  app: {  port: process.env.PORT || 3000  },  db: {  host: process.env.DB\_HOST || 'lab4.0pfe1.mongodb.net',  login: process.env.DB\_LOGIN || 'User123',  password: process.env.DB\_PW || '123qweasd',  name: process.env.DB\_NAME || 'Labs'  },  cloudinary: {  cloud\_name: 'blacktwig',  api\_key: '279191578123166',  api\_secret: 'M1SQLyCNbjcSSg-9sGfvPPaqrFY'  }  };  module.exports = config; |

**Приклади результатів**

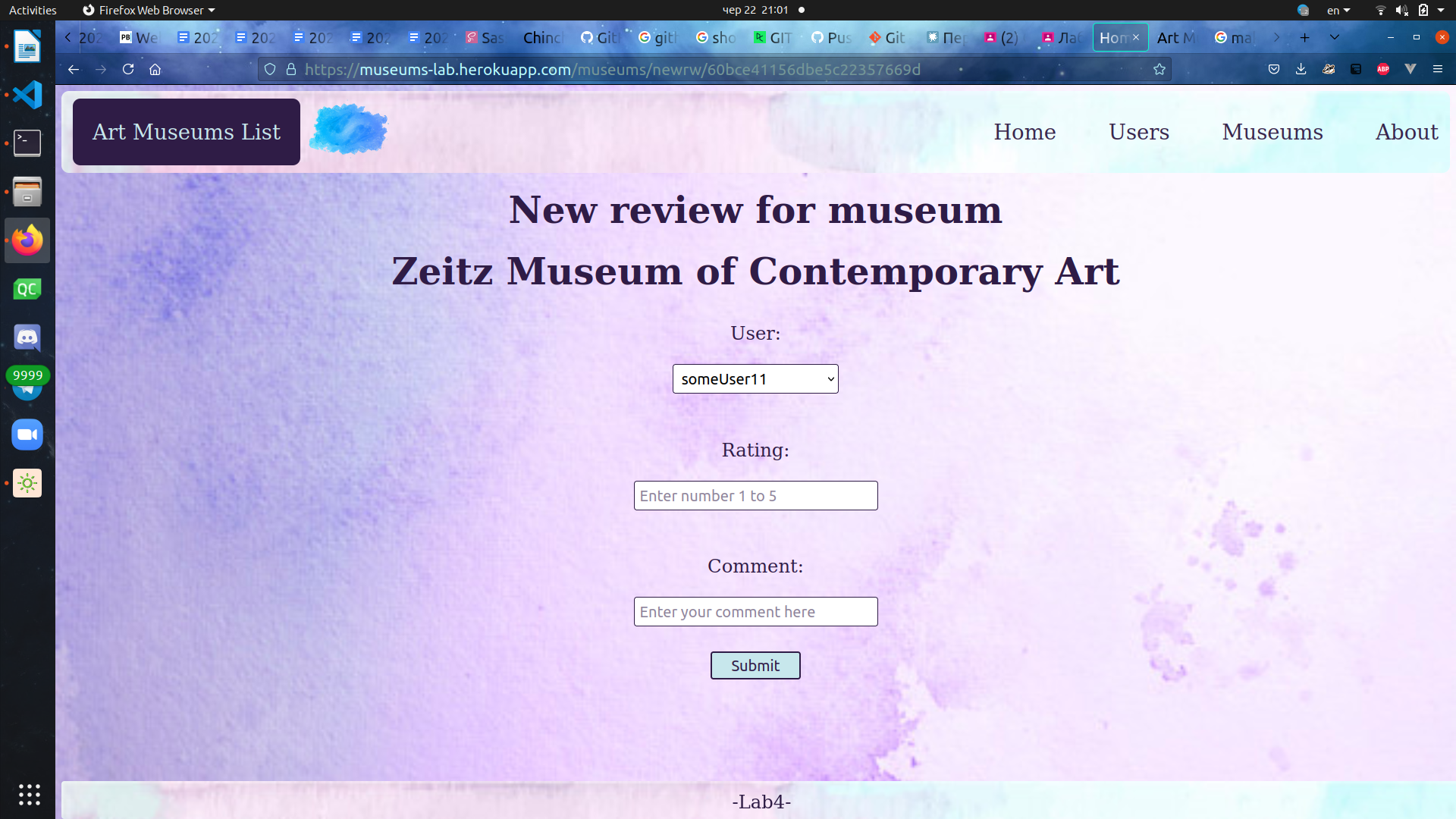
Зображення всіх нових та оновлених сторінок:

1. museum.mst

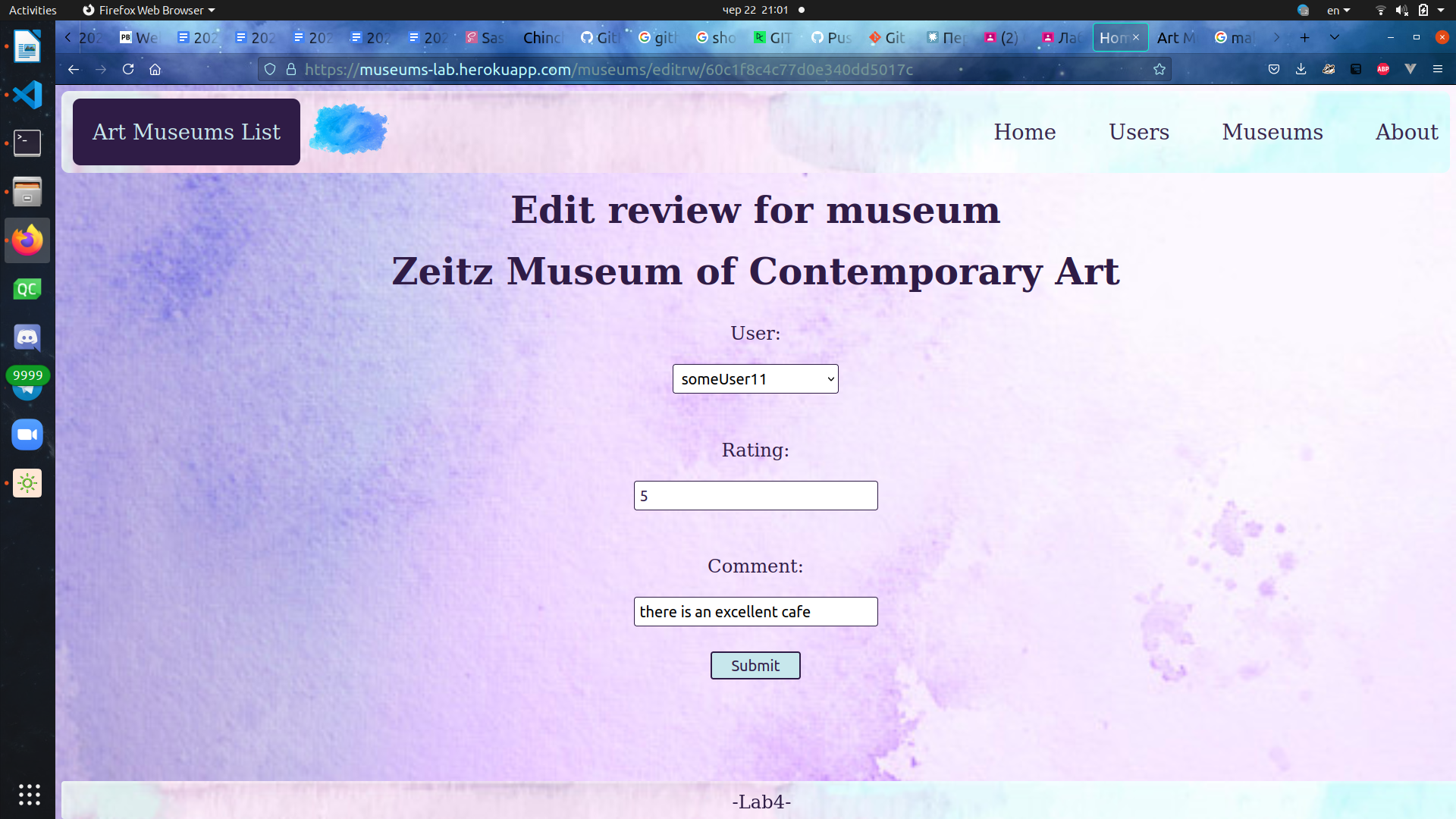


2. reviewform.mst

new review:



edit review:



**Висновки**

Виконуючи дану лабораторну роботу ми ознайомились з хостингом веб-додатків Heroku, хостингом файлів Cloudinary та сервісами для створення та управління віддаленою БД MongoDB Atlas. Також ми скористалися принципами асинхронного програмування, доступними у мові JavaScript, щоб взаємодіяти з базою даних. До того ж ми налаштували сховище медіа для взаємодії з віддаленою базою даних, та опубліковали веб-сервіс в мережі Інтернет.