rr

**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 №  8**

**з дисципліни “ Основи web-програмування”**

**тема “Створення REST API сервера”**

# 

|  |  |  |
| --- | --- | --- |
| **Виконав(ла)**  **студент(ка) II курсу**  **групи КП-83**  **Коваль Андрій Олекснадрович**  **(прізвище, ім’я, по батькові)**  **варіант № 10** |  | **Перевірив**  **“\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.**  **викладач**  **Гадиняк Руслан Анатолійович**  **(прізвище, ім’я, по батькові)** |

**Київ 2018**

**Мета роботи**

Реалізувати JSON REST API для доступу до ресурсів веб-сервера.

**Постановка завдання**

1. У серверній частині додати JSON REST API для виконання CRUD операцій над ресурсами сервера (Users, puzzles, carts, orders).
2. Отримувати об'єкти можна одиночно по ідентифікатору, або декілька списком. Отримання об'єктів списком дозволяє використовувати URL аргументи для фільтрації результату і обов'язково має мати пагінацію (не повертати всі результати одразу, а розбивати їх на підсписки фіксованої довжини - "сторінки").
3. Всі URL для доступу до API мають починатись з /api/v1 (наприклад, /api/v1/users, /api/v1/users/13 і т.д). При моделюванні шляхів використовувати загальні рекомендації.
4. Запит GET /developer/v1 має повертати HTML сторінку, у якій коротко задокументувати реалізований REST API версії v1:
   * Доступні ресурси, їх URL, методи доступу до них (які параметри мають, який результат повертається)
   * Спосіб авторизації для доступу до ресурсів
   * Формат і види помилок

**Тексти коду програм**

|  |
| --- |
| **api/api.js** |
| **const express = require('express');**  **const router = express.Router();**  **const cors = require('cors');**  **const bodyParser = require('body-parser');**  **const { checkAuth } = require('../../config/passport');**  **const userRoutes = require('./user');**  **const puzzleRoutes = require('./puzzles');**  **const cartsRoutes = require('./carts');**  **const authRoutes = require('./auth');**  **const orderRoutes = require('./order');**  **router.use(bodyParser.urlencoded({ extended: true }));**  **router.use(bodyParser.json())**  **const passport = require('passport');**  **router.use(passport.initialize());**  **router.use(cors());**  **router.use('/users', userRoutes);**  **router.use('/puzzles', puzzleRoutes);**  **router.use('/carts', cartsRoutes);**  **router.use('/auth', authRoutes);**  **router.use('/orders', orderRoutes);**  **router.get('', async (req, res) => {**  **res.json({});**  **});**  **router.post('/me', checkAuth, async (req, res) => {**  **if (!req.user) {**  **next({**  **status: 401,**  **message: 'Non authorized'**  **});**  **}**  **res.json(req.user);**  **});**    **router.use((err, req, res, next) => {**  **res.status(err.status || 500);**  **res.json({**  **error: {**  **message: err.message**  **}**  **});**  **});**  **module.exports = router;** |

|  |
| --- |
| **app.js** |
| const express = require('express');  const app = express();  const apiRoutes = require('./routes/api/api');  const { storage } = require('./config/multerStorage');  const PORT = parseInt(process.argv[2]) || process.env.PORT || 12010;  const mongoose = require('mongoose');  const dbUrl = require('./config/db\_connection');  mongoose.connect(dbUrl, { useNewUrlParser: true, useUnifiedTopology: true })  .then(() => console.log('connected to mongoDB'))  .catch(err => console.log(`Mongo Error: ${err}`))  .then(() => app.listen(PORT, () => console.log(`Server is ready on port ${PORT}`)))  .catch((err) => console.error(`Starting Error ${err}`));  app.use(storage.single("file"));  app.use(require('morgan')('dev'));  app.use('/api/v1', apiRoutes); |

|  |
| --- |
| **api/puzzles.js** |
| **const express = require('express');**  **const router = express.Router();**  **const Puzzle = require('../../models/puzzle');**  **const {checkAdmin, checkAuth, checkManager} = require('../../config/passport');**  **router.post('', async (req, res, next) => {**  **try {**  **const limit = req.body.limit;**  **const offset = req.body.offset;**  **if (limit === undefined) {**  **limit = 10;**  **} if (offset === undefined) {**  **offset = 0;**  **}**  **const response = await Puzzle.getFilteredSearch(req.body);**  **res.json(response);**  **} catch (err) {**  **console.log(err);**  **next(err);**  **}**  **});**  **router.get('/all', async (req, res, next) => {**  **try {**  **const puzzles = await Puzzle.getAll();**  **res.json(puzzles);**  **} catch (err) {**  **next(err);**  **}**  **});**  **router.delete('/:id([\\da-z]+)', checkAuth, checkManager, checkPuzzle, async (req, res, next) => {**  **try {**  **const puzzleId = req.params.id;**  **const result = await Puzzle.deleteById(puzzleId);**  **res.json(result);**  **} catch (err) {**  **next(err);**  **}**  **});**  **router.post('/new/mp', checkAuth, checkManager ,async (req, res, next) => {**  **const puzzle = await Puzzle.getPuzzleFromFormRequest(req);**  **if (puzzle == null) {**  **next({**  **status: 400,**  **message: 'Missing puzzle name, manydacturerId or typeId'**  **})**  **return;**  **}**  **try {**  **const insertedPuzzle = await Puzzle.insert(puzzle);**  **res.status(201).send({**  **puzzle: insertedPuzzle**  **});**  **} catch (err) {**  **next(err);**  **}**  **});**  **router.patch('/:id([\\da-z]{1,24})', checkAuth, checkManager, checkPuzzle, async (req, res, next) => {**  **const puzzle = await Puzzle.getPuzzleFromFormRequest(req);**  **const puzzleId = req.params.id;**  **puzzle.\_id = puzzleId;**  **try {**  **await Puzzle.update(puzzle);**  **} catch (err) {**  **next(err);**  **}**  **});**  **router.get('/filters', async (req, res, next) => {**  **try {**  **const filters = await Puzzle.getFilters();**  **res.json(filters);**  **} catch (err) {**  **next(err);**  **}**  **});**  **router.get('/:id([\\da-z]{1,24})', checkPuzzle, async (req, res, next) => {**  **console.log(req.puzzle);**  **res.json(req.puzzle);**  **});**  **async function checkPuzzle(req, res, next) {**  **const puzzleId = req.params.id;**  **const err = {**  **status: 404,**  **message: `Puzzle with id ${puzzleId} not found`**  **};**  **if (puzzleId.length != 24) {**  **next(err);**  **return;**  **}**  **const foundPuzzle = await Puzzle.getById(puzzleId);**  **if (!foundPuzzle) {**  **next(err);**  **return;**  **}**  **req.puzzle = foundPuzzle**  **next();**  **}**  **module.exports = router;** |

|  |
| --- |
| **api/users.js** |
| **const express = require('express');**  **const router = express.Router();**  **const User = require('../../models/user/user');**  **const { checkAdmin, checkAuth, checkManager } = require('../../config/passport');**  **router.post('/', checkAuth, async (req, res, next) => {**  **const limit = req.body.limit;**  **const offset = req.body.offset;**  **if (limit === undefined) {**  **limit = 10;**  **} if (offset === undefined) {**  **offset = 0;**  **}**  **try {**  **const response = await User.getPaginated(limit, offset);**  **res.json(response);**  **} catch (err) {**  **console.log(err);**  **res.status(500).json({**  **err**  **})**  **}**  **})**  **router.get('/all', checkAuth, async (req, res, next) => {**  **try {**  **const users = await User.getAll();**  **res.json(users);**  **} catch (err) {**  **res.sendStatus(500);**  **}**  **});**  **router.patch('/:id([\\da-z]{1,24})', checkUserById, checkAuth, async (req, res, next) => {**  **const userToPatchId = req.params.id;**  **const canUpdate = (req.user.role.toUpperCase() === 'ADMIN' || req.user.\_id === userToPatchId);**  **if (!req.body) {**  **next({**  **status: 400,**  **message: 'Requset body is empty'**  **})**  **return;**  **}**  **if (canUpdate) {**  **try {**  **const response = await User.update(req.body);**  **res.json(response);**  **} catch (err) {**  **console.log(err);**  **next(err);**  **return;**  **}**  **} else {**  **next({**  **status: 403,**  **message: 'No rights to update user'**  **});**  **return;**  **}**  **})**  **router.get('/:id([\\da-z]{1,24})', checkUserById, checkAuth, async (req, res, next) => {**  **res.json(req.foundUser);**  **});**  **async function checkUserById(req, res, next) {**  **const userId = req.params.id;**  **if (userId.length != 24) {**  **next({**  **status: 404,**  **message: `User with id ${userId} not found`**  **});**  **return;**  **}**  **const user = await User.getById(userId);**  **if (!user) {**  **next({**  **status: 404,**  **message: `User with id ${userId} not found`**  **});**  **return;**  **}**  **req.foundUser = user;**  **next();**  **}**  **module.exports = router;** |

|  |
| --- |
| **api/order.js** |
| **const express = require('express');**  **const router = express.Router();**  **const Order = require('../../models/order');**  **const Cart = require('../../models/cart');**  **const { checkAdmin, checkAuth, checkManager } = require('../../config/passport');**  **const orderStates = [**  **1, 2, 3**  **]**  **router.post('', checkAuth, checkManager, async (req, res) => {**  **//@todo поиск по подстроке какой либо**  **const limit = req.body.limit;**  **const offset = req.body.offset;**  **if (limit === undefined || offset === undefined) {**  **next({**  **status: 400,**  **message: 'Limit and offset are required fields'**  **});**  **return;**  **}**  **try {**  **const response = await Order.getFilteredSearch('', parseInt(limit), parseInt(offset));**  **res.json(response);**  **} catch (err) {**  **console.log(err);**  **next(err);**  **return;**  **}**  **});**  **router.post('/setstate/:id/:state', checkOrder, async (req, res) => {**  **const state = req.params.state;**  **if (orderStates.find((s) => s === state)) {**  **const response = await Order.setState(req.order.\_id, state);**  **res.json(response);**  **} else {**  **next({**  **status: 400,**  **message: 'Invalid order state'**  **});**  **return;**  **}**  **});**  **router.post('/new', checkAuth, checkCart, async (req, res) => {**  **const cartId = req.body.cartId;**  **const userId = req.user.\_id;**  **const price = Cart.getFullPrice(req.cart);**  **const newOrder = new Order(cartId, userId, price);**  **try {**  **const response = await Order.insert(newOrder);**  **res.status(201).json(response);**  **} catch (err) {**  **console.log(err);**  **next(err);**  **}**  **});**  **router.get('/:id', checkAuth, checkManager, checkOrder, async (req, res, next) => {**  **res.json(req.order);**  **});**  **async function checkCart(req, res, next) {**  **const cartId = req.body.cartId;**  **const err = {**  **status: 404,**  **message: `Cart with id ${cartId} not found`**  **};**  **if (cartId.length != 24) {**  **next(err);**  **return;**  **}**  **const cart = await Cart.getById(cartId);**  **if (!cart) {**  **next(err);**  **return;**  **}**  **req.cart = cart;**  **next();**  **}**  **async function checkOrder(req, res, next) {**  **const orderId = req.params.id;**  **const err = {**  **status: 404,**  **message: `Order with id ${orderId} not found`**  **};**  **if (orderId.length != 24) {**  **next(err)**  **return;**  **}**  **const order = await Order.getById(orderId);**  **if (!order) {**  **next(err);**  **return;**  **}**  **req.order = order;**  **next();**  **}**  **module.exports = router;** |

|  |
| --- |
| **api/carts.js** |
| **const express = require('express');**  **const router = express.Router();**  **const Cart = require('../../models/cart');**  **const User = require('../../models/user/user');**  **const Puzzle = require('../../models/puzzle');**  **const { checkAdmin, checkAuth, checkManager } = require('../../config/passport');**  **router.get('/:id([\\da-z]{24})', checkAuth, checkCartBelongingToUser, async (req, res, next) => {**  **const cart\_id = req.params.id**  **const cart = await Cart.getById(cart\_id);**  **if (!cart) {**  **next({**  **status: 404,**  **message: `No cart with id ${cart\_id} found`**  **})**  **return;**  **}**  **else**  **res.json(cart);**  **});**  **router.post('/remove/:puzzleId([\\da-z]{24})', checkAuth, checkUserCart, checkPuzzle, async (req, res, next) => {**  **const puzzleId = req.params.puzzleId;**  **try {**  **const responseCart = await Cart.removePuzzle(req.cart, puzzleId);**  **res.json({**  **cart: responseCart**  **})**  **} catch (err) {**  **next(err);**  **}**  **});**  **router.post('/insert/:puzzleId([\\da-z]{24})', checkAuth, checkUserCart, checkPuzzle, async (req, res, next) => {**  **const puzzleId = req.puzzleId;**  **const cartId = req.cartId;**  **try {**  **const responseCart = await Cart.insertPuzzle(cartId, puzzleId);**  **res.json({**  **cart: responseCart**  **});**  **return;**  **} catch (err) {**  **console.log(err);**  **next(err);**  **}**  **})**  **async function checkPuzzle(req, res, next) {**  **const puzzleId = req.params.puzzleId || '';**  **const err = {**  **status: 404,**  **message: 'Puzzle with id ' + puzzleId + ' not found'**  **};**  **if (puzzleId.length != 24) {**  **next(err);**  **return;**  **}**  **const puzzleFound = !!(await Puzzle.getById(puzzleId));**  **if (!puzzleFound) {**  **next(err);**  **return;**  **}**  **next();**  **}**  **async function checkUserCart(req, res, next) {**  **const puzzleId = req.params.puzzleId;**  **let cartId = req.user.cart || '';**  **if (!req.user.cart) {**  **try {**  **cartId = await User.setNewCart(req.user.\_id);**  **} catch (err) {**  **console.log(err);**  **next(err);**  **return;**  **}**  **}**  **req.puzzleId = puzzleId;**  **req.cart = await Cart.getById(cartId);**  **next();**  **}**  **async function checkCartBelongingToUser(req, res, next) {**  **const cart\_id = req.params.id**  **if (cart\_id !== req.user.cart) {**  **next({**  **status: 403,**  **message: 'No permission to get user cart'**  **});**  **return;**  **}**  **next();**  **}**  **module.exports = router;** |

**Приклади результатів**

Приклади результатів описані на сторінці /developer/v1. Ось деякі з них

|  |
| --- |
|  |
| Отримання поточного користувача |

|  |
| --- |
|  |
| **Шаблон помилки** |

|  |
| --- |
|  |
| Додавання/видалення пазлу з корзини покупок |

|  |
| --- |
|  |
| Пагінування пазлів та інших сутностей |

**Висновки**

Я навчився реалізувати JSON REST API для доступу до ресурсів веб-сервера.