ДОДАТКИ

ДОДАТОК А

Вміст проекту pc-builder

Controllers/authController:

```
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const { findUserByEmail, createUser } = require('../models/userModel');
const register = async (req, res) => {
  const { username, email, password } = req.body;
  try {
    const existingUser = await findUserByEmail(email);
    if (existingUser) return res.status(400).json({ error: 'Email already in use' });

    const hashedPassword = await bcrypt.hash(password, 10);
    const newUser = await createUser(username, email, hashedPassword);

    res.status(201).json({ message: 'User registered successfully', user: newUser });
  } catch (err) {
    res.status(500).json({ error: 'Server error during registration' });
  }
};
const login = async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await findUserByEmail(email);
    if (!user) return res.status(400).json({ error: 'Invalid credentials' });
    const isMatch = await bcrypt.compare(password, user.password_hash);
    if (!isMatch) return res.status(400).json({ error: 'Invalid credentials' });

    const token = jwt.sign({ id: user.id, role: user.role }, process.env.JWT_SECRET,
{ expiresIn: '1d' });
    res.json({ token, user: { id: user.id, email: user.email, role: user.role } });
  } catch (err) {
```

```
        res.status(500).json({ error: 'Server error during login' });
          }
        };
        module.exports = { register, login };
```

Controllers/componentController:

```
          const model = require('../models/componentModel');
const getAllOrders = async (req, res) => {
  try {
    const orders = await model.getAll(); // або інша назва методу з models
    res.json(orders);
  } catch (err) {
    res.status(500).json({ error: 'Failed to fetch orders' });
  }
};
async function getAll(req, res) {
  const data = await model.getAllComponents();
  res.json(data);
}
async function getById(req, res) {
  const component = await model.getComponentById(req.params.id);
  if (!component) return res.status(404).json({ error: 'Not found' });
  res.json(component);
}
async function create(req, res) {
  const created = await model.createComponent(req.body);
  res.status(201).json(created);
}
async function update(req, res) {
  const updated = await model.updateComponent(req.params.id, req.body);
  res.json(updated);
}
async function remove(req, res) {
  await model.deleteComponent(req.params.id);
```

```
 res.json({ message: 'Deleted' });
}

module.exports = { getAll, getById, create, update, remove, getAllOrders };
```

Controllers/computerController:

```
const model = require('../models/computerModel');
// Отримати всі комп’ютери
async function getAll(req, res) {
  try {
    const data = await model.getAllComputers();
    res.json(data);
  } catch (err) {
    res.status(500).json({ error: 'Failed to get computers' });
  }
}
// Отримати комп’ютер за ID
async function getById(req, res) {
  try {
    const computer = await model.getComputerById(req.params.id);
    if (!computer) {
      return res.status(404).json({ error: 'Computer not found' });
    }
    res.json(computer);
  } catch (err) {
    res.status(500).json({ error: "Failed to get computer" });
  }
}
// Створити нову конфігурацію ПК
async function create(req, res) {
  try {
    const { name, description, price, componentIds } = req.body;
    const created = await model.createComputer({ name, description, price, componentIds });
    res.status(201).json(created);
```

```
} catch (err) {
    res.status(500).json({ error: "Failed to create computer" });
  }
}
async function update(req, res) {
  const { name, price, componentIds } = req.body;
  try {
    await model.updateComputer(req.params.id, name, price, componentIds);
    res.json({ message: 'Updated successfully' });
  } catch (err) {
    console.error("Update error:", err);
    res.status(500).json({ error: "Failed to update computer" });
  }
}
// Видалити конфігурацію ПК
async function remove(req, res) {
  try {
    await model.deleteComputer(req.params.id);
    res.json({ message: 'Deleted' });
  } catch (err) {
    res.status(500).json({ error: "Failed to delete computer" });
  }
}
module.exports = { getAll, getById, create, remove, update };
```

Controllers/orderController:

```
const db = require("../db");
const model = require("../models/orderModel");

// Отримання замовлень поточного користувача
const getMyOrders = async (req, res) => {
  try {
    const orders = await model.getOrdersByUser(req.user.id);
    res.status(200).json(orders);
```

```
} catch (err) {
    res.status(500).json({ error: "Failed to fetch user orders" });
  }
};
// Створення нового замовлення
const create = async (req, res) => {
  try {
    const userId = req.user.id;
    const { computerId } = req.body;
    // Перевірити чи передано ID
    if (!computerId) {
      return res.status(400).json({ error: "Computer ID is required" });
    }
    // Витягуємо ціну комп'ютера
    const result = await db.query('SELECT price FROM computers WHERE id = $1', [computerId]);
    if (result.rows.length === 0) {
      return res.status(404).json({ error: "Computer not found" });
    }
    const total_price = result.rows[0].price;
    // Створити замовлення
    const order = await model.createOrder(userId, total_price, computerId);
    res.status(201).json(order);
  } catch (err) {
    console.error("ORDER CREATE ERROR:", err);
    res.status(500).json({ error: "Failed to create order" });
  }
};
const remove = async (req, res) => {
  try {
    const result = await model.deleteOrder(req.params.id);
    res.json({ message: "Order deleted" });
  } catch (err) {
    res.status(500).json({ error: "Failed to delete order" });
  }
```

```
};
// Отримання всіх замовлень (адміну)
const getAllOrders = async (req, res) => {
  try {
    const orders = await model.getAllOrders();
    res.json(orders);
  } catch (err) {
    res.status(500).json({ error: 'Не вдалося отримати всі замовлення' });
  }
};
// Отримання замовлень конкретного користувача (адмін)
const getUserOrders = async (req, res) => {
  try {
    const orders = await model.getOrdersByUser(req.user.id);
    res.json(orders);
  } catch (err) {
    res.status(500).json({ error: "Failed to get user orders" });
  }
};
// Отримання замовлення по ID
const getById = async (req, res) => {
  try {
    const orderId = req.params.id;
    const order = await model.getOrderById(orderId);
    if (!order) {
      return res.status(404).json({ error: 'Order not found' });
    }
    const items = await model.getOrderItems(orderId);
    res.json({ ...order, items });
  } catch (err) {
    res.status(500).json({ error: 'Failed to get order' });
  }
};
// Оновлення статусу
```

```
const updateStatus = async (req, res) => {
  try {
    const updated = await model.updateStatus(req.params.id, req.body.status);
    res.json(updated);
  } catch (err) {
    res.status(500).json({ error: "Failed to update status" });
  }
};

// Експорт
module.exports = {
  getMyOrders,
  create,
  remove,
  getUserOrders,
  getById,
  updateStatus,
  getAllOrders
};
```

Controllers/userController:

```
const db = require('../db');
const getAllUsers = async (req, res) => {
  try {
    const result = await db.query('SELECT id, email, role FROM users ORDER BY id');
    res.json(result.rows);
  } catch (err) {
    res.status(500).json({ error: 'Не вдалося отримати користувачів' });
  }
};
const updateUserRole = async (req, res) => {
  const { id } = req.params;
  const { role } = req.body;
  if (!['user', 'admin'].includes(role)) {
```

```
return res.status(400).json({ error: 'Невірна роль' });
  }
  try {
    await db.query('UPDATE users SET role = $1 WHERE id = $2', [role, id]);
    res.json({ message: 'Роль оновлена' });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Не вдалося оновити роль' });
  }
};
module.exports = {
  getAllUsers,
  updateUserRole,
};
```

Middlware/authMiddleware:

```
onst jwt = require("jsonwebtoken");
function verifyToken(req, res, next) {
  const authHeader = req.headers.authorization;
  if (!authHeader || typeof authHeader !== "string") {
    return res.status(403).json({ error: "No token provided or token format is incorrect" });
  }
  const token = authHeader.startsWith("Bearer ") ? authHeader.slice(7) : authHeader;
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (err) {
    return res.status(401).json({ error: "Invalid token" });
  }
}
module.exports = { verifyToken };
```

Middlware/roleMiddleware:

```
function isAdmin(req, res, next) {
```

```
if (!req.user || req.user.role !== 'admin') {
   return res.status(403).json({ error: 'Access denied. Admins only.' });
  }
  next();
}
function roleMiddleware(requiredRole) {
  return (req, res, next) => {
    if (!req.user || req.user.role !== requiredRole) {
      return res.status(403).json({ error: 'Access denied' });
    }
    next();
  };
}
module.exports = { roleMiddleware, isAdmin };
```

Models/ComponentModel:

```
const db = require('../db');
async function getAllComponents() {
  const result = await db.query('SELECT * FROM components ORDER BY id');
  return result.rows;
}
async function getComponentById(id) {
  const result = await db.query('SELECT * FROM components WHERE id = $1', [id]);
  return result.rows[0];
}
async function createComponent(data) {
  const { name, type, description, price, stock, image_url } = data;
  const result = await db.query(
    `INSERT INTO components (name, type, description, price, stock, image_url)
     VALUES ($1, $2, $3, $4, $5, $6) RETURNING *`,
    [name, type, description, price, stock, image_url]
  );
  return result.rows[0];
}
async function updateComponent(id, data) {
```

```
const { name, type, description, price, stock, image_url } = data;
  const result = await db.query(
    `UPDATE components SET name=$1, type=$2, description=$3, price=$4, stock=$5,
image_url=$6
     WHERE id=$7 RETURNING *`,
    [name, type, description, price, stock, image_url, id]
  );
  return result.rows[0];
}
async function deleteComponent(id) {
  await db.query('DELETE FROM components WHERE id=$1', [id]);
}
module.exports = {
  getAllComponents,
  getComponentById,
  createComponent,
  updateComponent,
  deleteComponent
};
```

Models/ComputerModel:

```
const db = require('../db');
const getAllComputers = async () => {
  const computersRes = await db.query('SELECT * FROM computers');
  const computers = computersRes.rows;
  for (const comp of computers) {
    const result = await db.query(
      `SELECT c.* FROM components c
       JOIN computer_components cc ON c.id = cc.component_id
       WHERE cc.computer_id = $1`,
      [comp.id]
    );
    comp.components = result.rows;
  }
```

```
return computers;
};
const getComputerById = async (id) => {
  const res = await db.query("SELECT * FROM computers WHERE id = $1", [id]);
  const computer = res.rows[0];
  if (!computer) return null;
  const comps = await db.query(
    `SELECT c.* FROM components c
     JOIN computer_components cc ON c.id = cc.component_id
     WHERE cc.computer_id = $1`,
    [id]
  );
  computer.components = comps.rows;
  return computer;
};
async function createComputer({ name, description, price, componentIds }) {
  const result = await db.query(
    'INSERT INTO computers (name, description, price) VALUES ($1, $2, $3) RETURNING *',
    [name, description, price]
  );
  const computer = result.rows[0];
  for (const compId of componentIds) {
    await db.query(
      'INSERT INTO computer_components (computer_id, component_id) VALUES ($1, $2)',
      [computer.id, compId]
    );
  }
  return computer;
}
async function updateComputer(id, name, price, componentIds) {
  await db.query('UPDATE computers SET name = $1, price = $2 WHERE id = $3', [name, price,
id]);
  await db.query('DELETE FROM computer_components WHERE computer_id = $1', [id]);
  for (const compId of componentIds) {
```

```
await db.query(
     'INSERT INTO computer_components (computer_id, component_id) VALUES ($1, $2)',
     [id, compId]
    );
  }
}
async function deleteComputer(id) {
  await db.query('DELETE FROM computers WHERE id = $1', [id]);
}
module.exports = {
  getAllComputers,
  getComputerById,
  createComputer,
  deleteComputer,
  updateComputer
};
```

Models/OrderModel:

```
const db = require('../db');
async function createOrder(userId, total_price, computerId) {
  const result = await db.query(
    'INSERT INTO orders (user_id, total_price, computer_id) VALUES ($1, $2, $3) RETURNING
*',
    [userId, total_price, computerId]
  );
  return result.rows[0];
}
async function createOrderFromComputer(userId, computerId) {
  const result = await db.query(
    `INSERT INTO orders (user_id, computer_id, status)
     VALUES ($1, $2, 'очікується') RETURNING *`,
    [userId, computerId]
  );
  return result.rows[0];
```

```
}
async function getOrdersByUser(userId) {
  const result = await db.query(
    'SELECT * FROM orders WHERE user_id = $1 ORDER BY created_at DESC',
    [userId]
  );
  return result.rows;
}
async function getOrderById(orderId) {
  const result = await db.query(
    'SELECT * FROM orders WHERE id = $1',
    [orderId]
  );
  return result.rows[0];
}
async function getOrderItems(orderId) {
  const result = await db.query(
    `SELECT oi.quantity, c.name
     FROM order_items oi
     JOIN computers c ON oi.computer_id = c.id
     WHERE oi.order_id = $1`,
    [orderId]
  );
  return result.rows;
}
async function updateStatus(id, status) {
  const result = await db.query(
    'UPDATE orders SET status = $1 WHERE id = $2 RETURNING *',
    [status, id]
  );
  return result.rows[0];
}
async function getAllOrders() {
  const result = await db.query(
```

```
`SELECT o.*, u.email AS user_email, c.name AS computer_name
    FROM orders o
    LEFT JOIN users u ON o.user_id = u.id
    LEFT JOIN computers c ON o.computer_id = c.id
    ORDER BY o.created_at DESC`
  );
  return result.rows;
}
module.exports = {
  createOrder,
  createOrderFromComputer,
  getOrdersByUser,
  getOrderById,
  getOrderItems,
  updateStatus,
  getAllOrders
};
```

Models/UserModel:

```
const db = require('../db');
async function findUserByEmail(email) {
  const result = await db.query('SELECT * FROM users WHERE email = $1', [email]);
  return result.rows[0];
}
async function createUser(username, email, passwordHash) {
  const result = await db.query(
    'INSERT INTO users (username, email, password_hash) VALUES ($1, $2, $3) RETURNING *',
    [username, email, passwordHash]
  );
  return result.rows[0];
}
module.exports = { findUserByEmail, createUser };
```

Routes/authRoutes:

```
const express = require('express');
```

```
const router = require('express').Router();
const { register, login } = require('../controllers/authController');
router.post('/register', register);
router.post('/login', login);
module.exports = router;
```

Routes/componentRoutes:

```
const express = require('express');
const router = require('express').Router();
const controller = require('../controllers/componentController');
const { verifyToken } = require('../middleware/authMiddleware');
const { roleMiddleware, isAdmin } = require('../middleware/roleMiddleware'); // ← важливо
router.get('/', controller.getAll);
router.get('/:id', controller.getById);
// Адмінські маршрути
router.post('/', verifyToken, isAdmin, controller.create);
router.put('/:id', verifyToken, isAdmin, controller.update);
router.delete('/:id', verifyToken, isAdmin, controller.remove);
router.get('/', verifyToken, roleMiddleware('admin'), controller.getAllOrders);
module.exports = router;
```

Routes/computerRoutes:

```
const express = require('express');
const router = express.Router();
const controller = require('../controllers/computerController');
const { verifyToken } = require('../middleware/authMiddleware');
const { isAdmin } = require('../middleware/roleMiddleware');
const computerController = require('../controllers/computerController');
// Публічні
router.get('/', controller.getAll);
router.get('/:id', controller.getById);
router.put('/:id', computerController.update);
// Адмінські
router.post('/', verifyToken, isAdmin, controller.create);
router.delete('/:id', verifyToken, isAdmin, controller.remove);
```

```
module.exports = router;
```

Routes/orderRoutes:

```
const express = require('express');
const router = express.Router();
const { getMyOrders } = require('../controllers/orderController');
const controller = require('../controllers/orderController');
const { verifyToken } = require('../middleware/authMiddleware');
const { roleMiddleware, isAdmin } = require('../middleware/roleMiddleware');
// Створення нового замовлення
router.post('/', verifyToken, controller.create);
// Отримати всі замовлення поточного користувача
router.get('/my', verifyToken, getMyOrders);
// Отримати замовлення за ID (для користувача або адміністратора)
router.get('/:id', verifyToken, controller.getById);
// Оновити статус (для адміністратора)
router.put('/:id/status', verifyToken, isAdmin, controller.updateStatus);
// Видалити замовлення (для адміністратора)
router.delete('/:id', verifyToken, isAdmin, controller.remove);
// Отримати всі замовлення (тільки для адміністратора)
router.get('/', verifyToken, roleMiddleware('admin'), controller.getAllOrders);
module.exports = router;
```

Routes/userRoutes:

```
const express = require("express");
const router = express.Router();
const { updateUserRole, getAllUsers } = require("../controllers/userController");
const { verifyToken } = require("../middleware/authMiddleware");
const { isAdmin } = require("../middleware/roleMiddleware");
router.get("/", verifyToken, isAdmin, getAllUsers);
router.put('/:id/role', verifyToken, isAdmin, updateUserRole);
module.exports = router;
```

.env:

```
PORT=3001
DB_HOST=localhost
```

```
DB_PORT=5432
DB_USER=postgres
DB_PASSWORD=artemka
DB_DATABASE=mydb
JWT_SECRET=your_very_secret_token
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: Розробка web-сервісу для організації процесів конфігурування, складання та продажу комп’ютерів засобами PostgreeSQL,Vite, RestApi

Обсяг пояснювальної записки: 51 сторінка

6 таблиць

26 рисунки

1 додаток

Дата завершення роботи: 12 червня 2025р.

Підпис студента- _____ Артем МАКОВІЙЧУК