Львівський національний університет імені Івана Франка Факультет:
електроніки і комп’ютерних технологій

Лабораторна робота No 5
з курсу “Веб програмування на стороні сервера”

« Розробка веб-серверу для системи інвентаризації»

Виконав: ст. ФеІ-21 Зборівський Юрій

Перевірив: Чмихало Олександр Сергійович

Львів 2023

Мета роботи: Реалізація веб сервера створення користувачів для інвентаризації девайсів.

Хід роботи

Створення базового сервера Express та реалізацію основних запитів

1. Імпортую усі необхідні бібліотеки та оголошую змінні

```
const express = require('express');
const mysql = require('mysql');
const bodyParser = require('body-parser');
const multer = require('multer');
const path = require('path');
const bcrypt = require('bcrypt');
const saltRounds = 10;
const swaggerUi = require('swagger-ui-express');
const YAML = require('yamljs');
const swaggerDocument = YAML.load('./swagger.yaml');
const app = express();
const port = 8000;
```

2. Підключення до бази даних

```
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'jorge',
  password: '06072004',
  database: 'devices'
});

connection.connect();
```

3. Створення таблиці бази даних використовуючи Mysql

```
CREATE TABLE devices (
   id INT AUTO_INCREMENT PRIMARY KEY,
   device_name VARCHAR(255) NOT NULL,
   description TEXT,
   serial_number VARCHAR(255),
   manufacturer VARCHAR(255),
   image_path VARCHAR(255);
);
```

4. Реалізація основних запитів

```
app.get('/devices', (req, res) => {
  connection.query('SELECT * FROM devices', (error, results, fields) => {
    if (error) {
      res.status(500).send('Error on the server.');
```

```
      return;
    }
    res.json(results);
  });
});

app.get('/devices/:id', (req, res) => {
  const { id } = req.params;
  connection.query('SELECT * FROM devices WHERE id = ?', [id], (error, results) =>
{
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    if (results.length > 0) {
      res.json(results[0]);
    } else {
      res.status(404).send('Device not found.');
    }
  });
});

app.post('/devices', (req, res) => {
  const { device_name, description, serial_number, manufacturer } = req.body;
  connection.query('INSERT INTO devices (device_name, description, serial_number,
manufacturer) VALUES (?, ?, ?, ?)',
  [device_name, description, serial_number, manufacturer],
  (error, results) => {
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    res.status(201).send(`Device added with ID: ${results.insertId}`);
  });
});

app.put('/devices/:id', (req, res) => {
  const { id } = req.params;
  const { device_name, description, serial_number, manufacturer } = req.body;
  connection.query('UPDATE devices SET device_name = ?, description = ?,
serial_number = ?, manufacturer = ? WHERE id = ?',
  [device_name, description, serial_number, manufacturer, id],
  (error, results) => {
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    if (results.affectedRows > 0) {
      res.send('Device updated successfully.');
    } else {
      res.status(404).send('Device not found.');
```

```
    }
  });
});

app.delete('/devices/:id', (req, res) => {
  const { id } = req.params;
  connection.query('DELETE FROM devices WHERE id = ?', [id], (error, results) => {
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    if (results.affectedRows > 0) {
      res.send('Device deleted successfully.');
    } else {
      res.status(404).send('Device not found.');
    }
  });
});


const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'uploads/')
  },
  filename: function (req, file, cb) {
    cb(null, file.fieldname + '-' + Date.now() + path.extname(file.originalname))
  }
});

const upload = multer({ storage: storage });

app.post('/upload/:deviceId', upload.single('deviceImage'), (req, res) => {
  const { deviceId } = req.params;
  const filePath = req.file.path;
  connection.query('UPDATE devices SET image_path = ? WHERE id = ?', [filePath,
deviceId], (error, results) => {
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    if (results.affectedRows > 0) {
      res.send(`File uploaded as ${filePath}`);
    } else {
      res.status(404).send('Device not found.');
    }
  });
});


app.use('/uploads', express.static(path.join(__dirname, 'uploads')));
```

```
app.get('/device-image/:deviceId', (req, res) => {
  const { deviceId } = req.params;

  connection.query('SELECT image_path FROM devices WHERE id = ?', [deviceId],
(error, results) => {
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    if (results.length > 0 && results[0].image_path) {
      const imagePath = results[0].image_path;

      const fullPath = path.join('uploads', imagePath);

      res.send(`
        <!DOCTYPE html>
        <html lang="en">
        <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-scale=1.0">
          <title>Device Image</title>
        </head>
        <body>
          <h1>Device Image</h1>
          <img src="/${imagePath}" alt="Device Image" style="max-width: 500px;">
        </body>
        </html>
      `);
    } else {
      res.status(404).send('Image not found.');
    }
  });
});



app.post('/register', async (req, res) => {
  const { username, password, email } = req.body;
  if (!username || !password || !email) {
    res.status(400).send('Username, password, and email are required.');
    return;
  }

  try {
    const hashedPassword = await bcrypt.hash(password, saltRounds);
    connection.query('INSERT INTO users (username, password, email) VALUES (?, ?,
?)',
    [username, hashedPassword, email],
    (error, results) => {
      if (error) {
        res.status(500).send('Error on the server.');
```

```
        return;
      }
      res.status(201).send(`User created with ID: ${results.insertId}`);
    });
  } catch (error) {
    res.status(500).send('Server error while hashing the password.');
  }
});

app.get('/users', (req, res) => {
  connection.query('SELECT * FROM users', (error, results) => {
    if (error) {
      res.status(500).send('Error on the server: ' + error.message);
      return;
    }
    res.json(results);
  });
});



app.post('/devices/:id/checkout', (req, res) => {
  const { id } = req.params;
  const userId = req.userId;
  connection.query('UPDATE devices SET user_id = ? WHERE id = ? AND user_id IS
NULL',
  [userId, id],
  (error, results) => {
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    if (results.affectedRows > 0) {
      res.send('Device checked out successfully.');
    } else {
      res.status(400).send('Device is not available or does not exist.');
    }
  });
});

app.post('/devices/:id/checkin', (req, res) => {
  const { id } = req.params;
  connection.query('UPDATE devices SET user_id = NULL WHERE id = ?', [id], (error,
results) => {
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    if (results.affectedRows > 0) {
      res.send('Device returned successfully.');
    } else {
```

```
      res.status(404).send('Device not found.');
    }
  });
});

app.get('/user/:userId/devices', (req, res) => {
  const { userId } = req.params;

  connection.query('SELECT * FROM devices WHERE user_id = ?', [userId], (error,
results) => {
    if (error) {
      res.status(500).send('Error on the server.');
      return;
    }
    res.json(results);
  });
});

app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument));


app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

Описи функцій запитів, які реалізовані

GET /devices: Виводить список усіх пристроїв, що зберігаються в базі даних. Повертає дані у форматі JSON з інформацією про кожен пристрій.

GET /devices/:id:Отримує детальну інформацію про конкретний пристрій за його ID. Повертає дані про пристрій у форматі JSON.

POST /devices: Додає новий пристрій до бази даних з даними, які були надіслані у тілі запиту.

PUT /devices/:id: Оновлює інформацію про існуючий пристрій за вказаним ID з новими даними, наданими у тілі запиту.

DELETE /devices/:id: Видаляє пристрій з бази даних за вказаним ID.

POST /upload/:deviceId: Дозволяє завантажити зображення для певного пристрою за його ID. Оновлює шлях до зображення пристрою в базі даних

GET /device-image/:deviceId: Повертає HTML-сторінку з вбудованим зображенням конкретного пристрою за його ID.

POST /register: Реєструє нового користувача з даними, які були надіслані.
Пароль хешується перед збереженням в базі даних.

GET /users: Виводить список усіх користувачів, що зберігаються в базі даних. Повертає дані користувачів у форматі JSON.

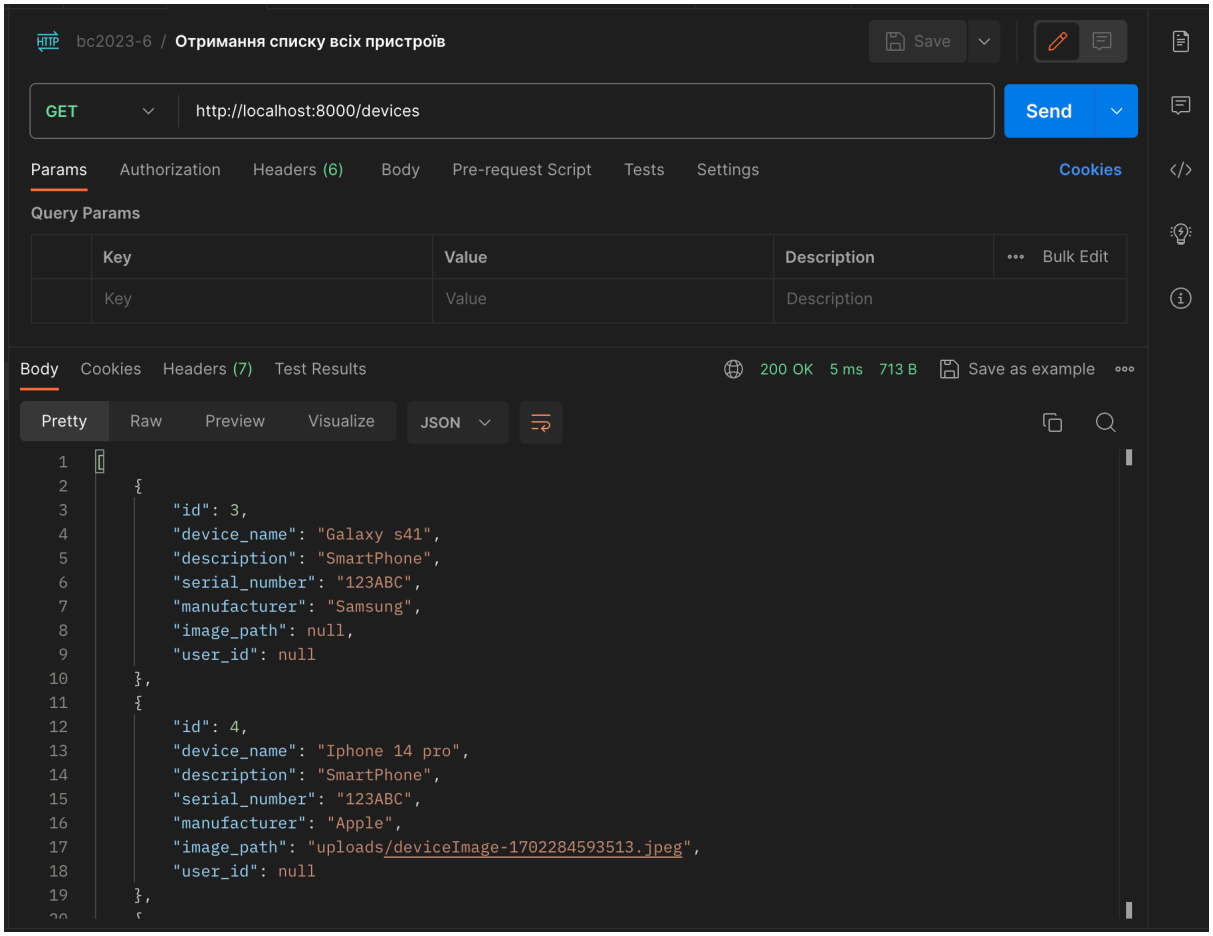
POST /devices/:id/checkout: Встановлює користувача як того, хто взяв у використання певний пристрій за ID.

POST /devices/:id/checkin: Повертає пристрій, який був взятий у користування, оновлюючи його статус у базі даних.

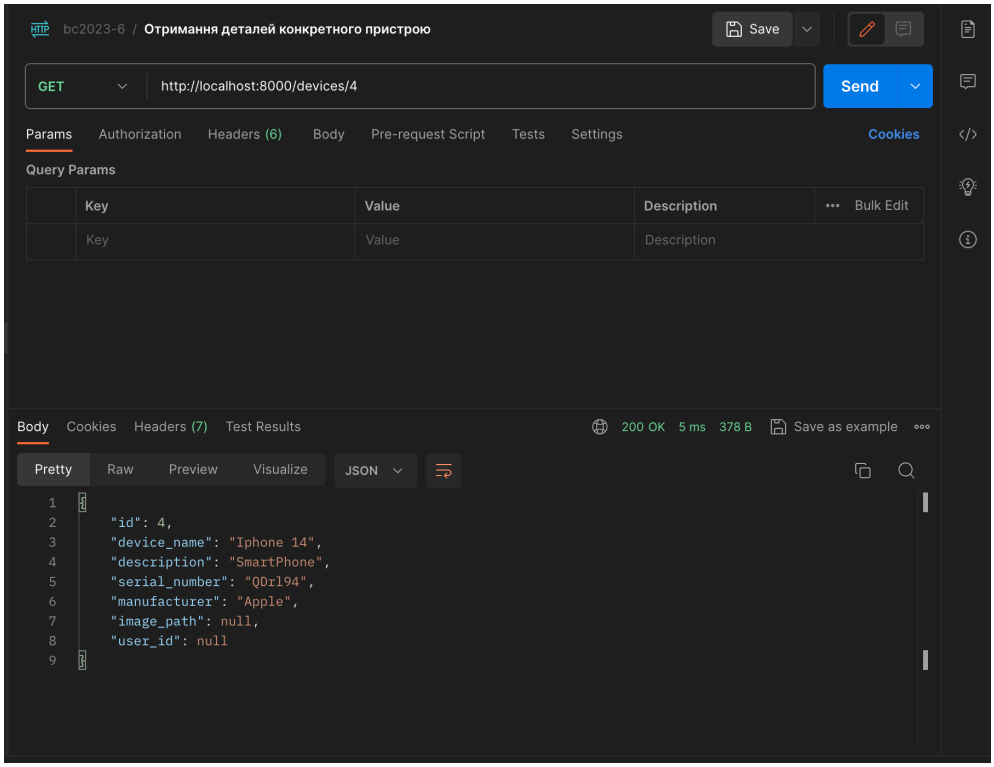
GET /user/:userId/devices: Виводить список пристроїв, які наразі знаходяться в користуванні у певного користувача за його ID. Повертає дані про пристрої у форматі JSON.

Тестування

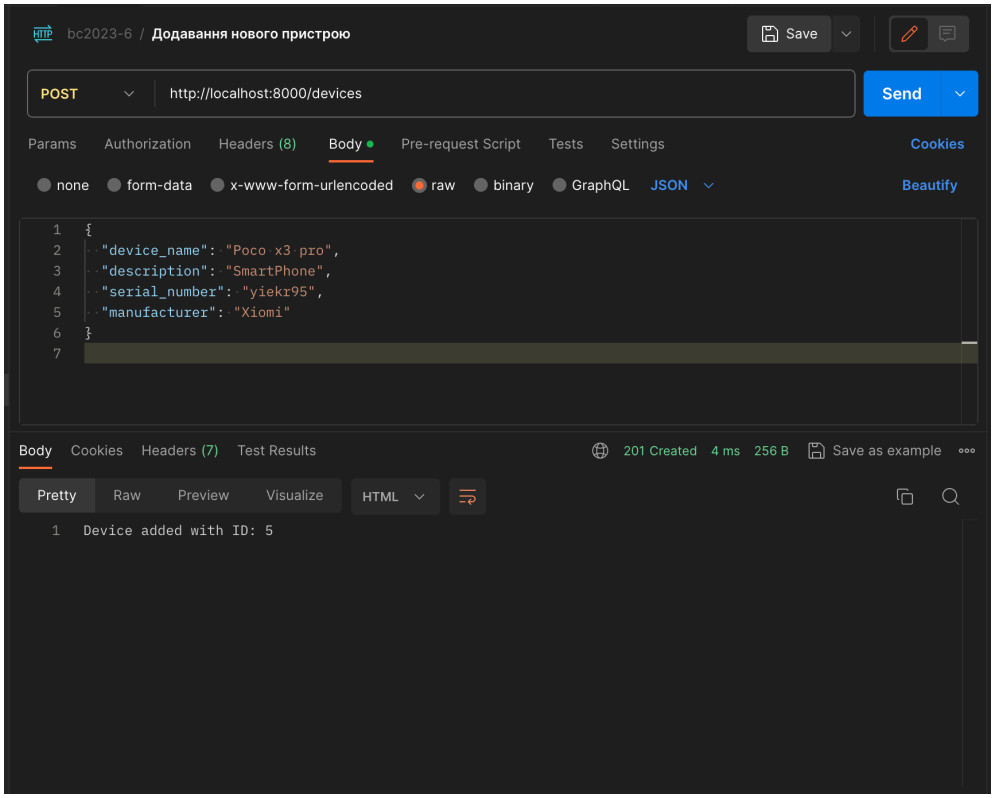
1. GET /devices: Виводить список усіх пристроїв



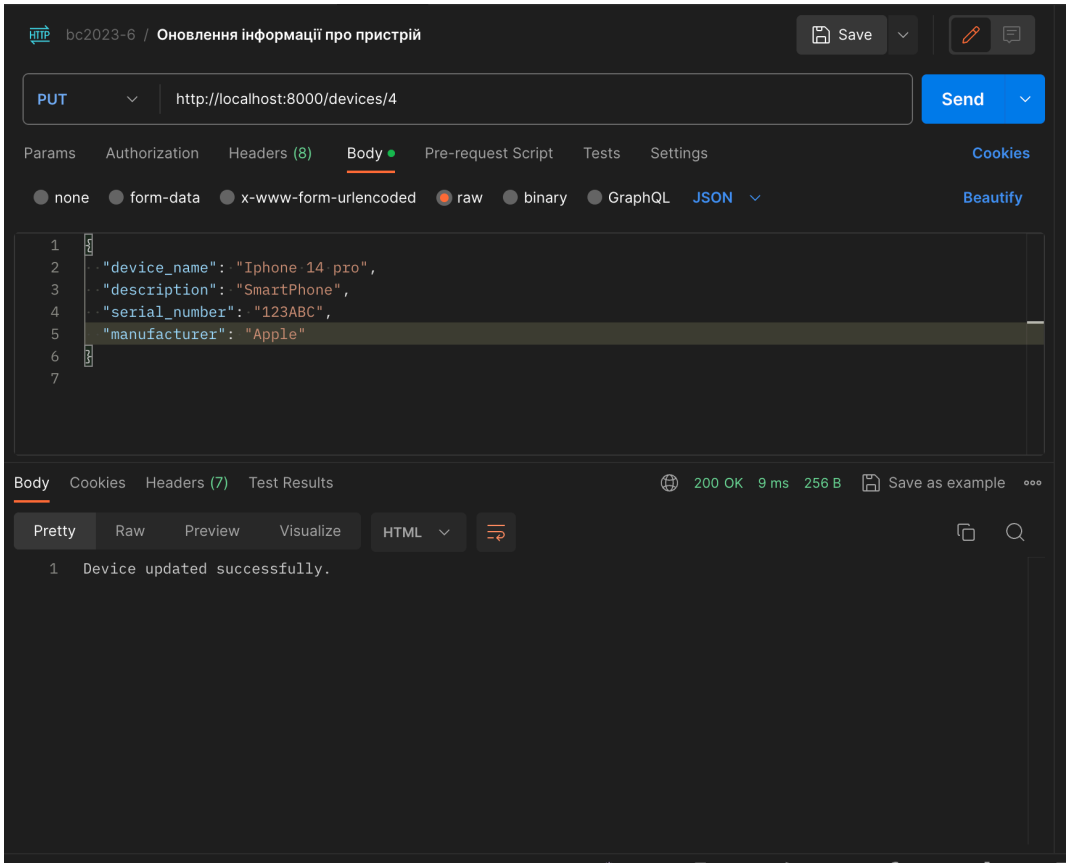
2. GET /devices/:id:Отримує детальну інформацію про конкретний пристрій за його ID.



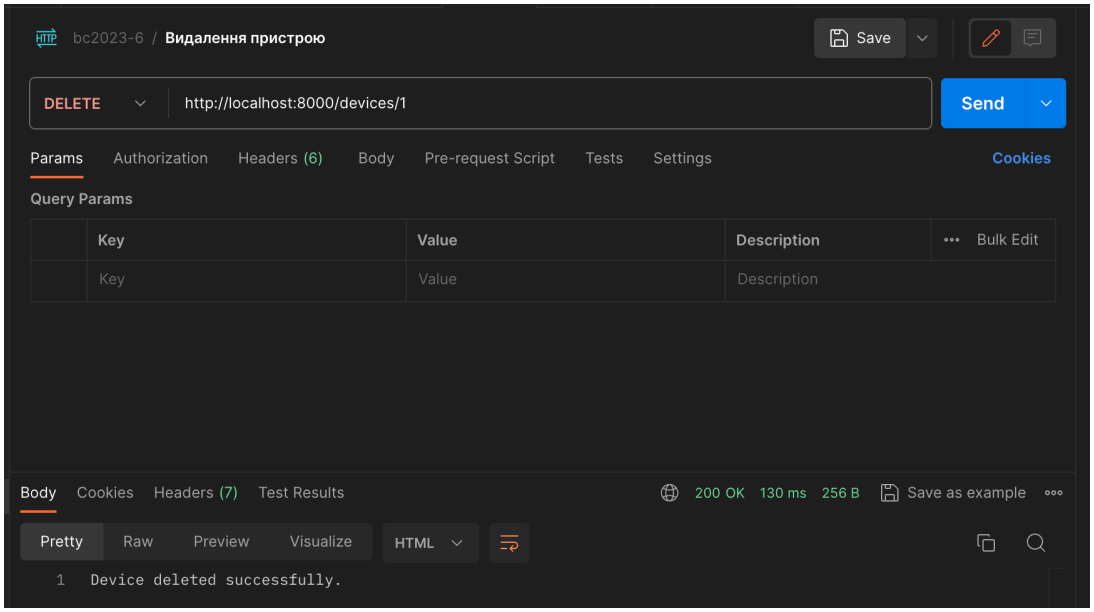
3. POST /devices: Додає новий пристрій до бази даних з даними, які були надіслані у тілі запиту.



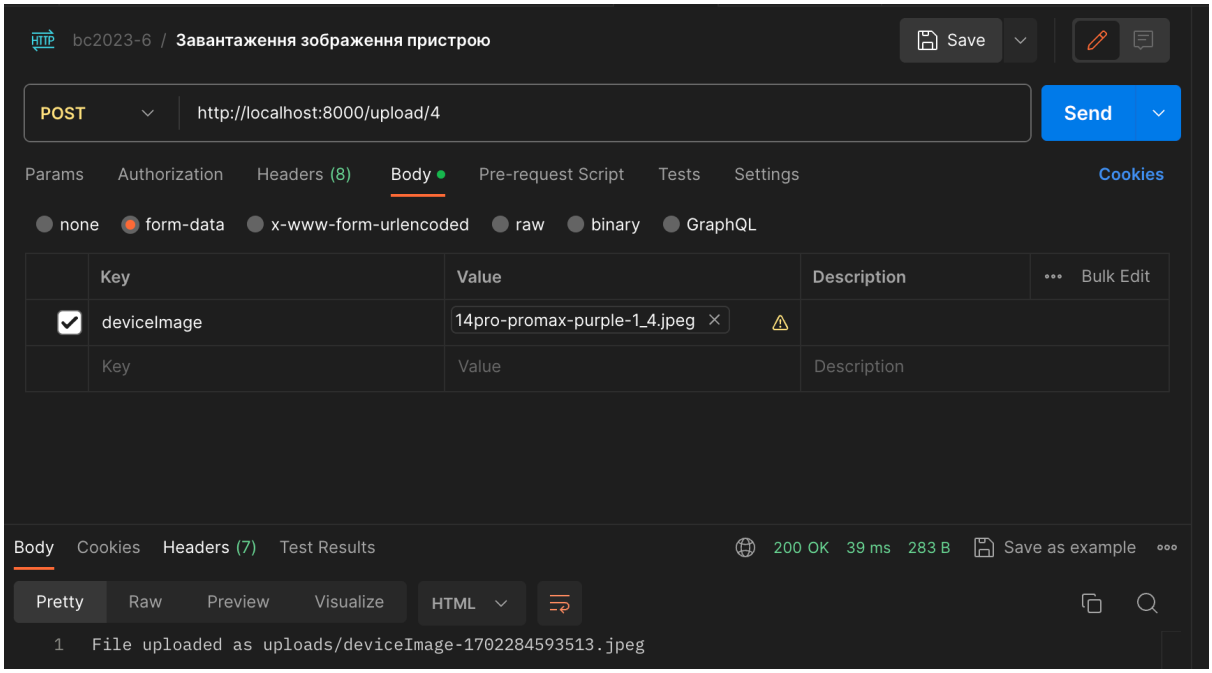
4. PUT /devices/:id: Оновлює інформацію про існуючий пристрій за вказаним ID з новими даними, наданими у тілі запиту.



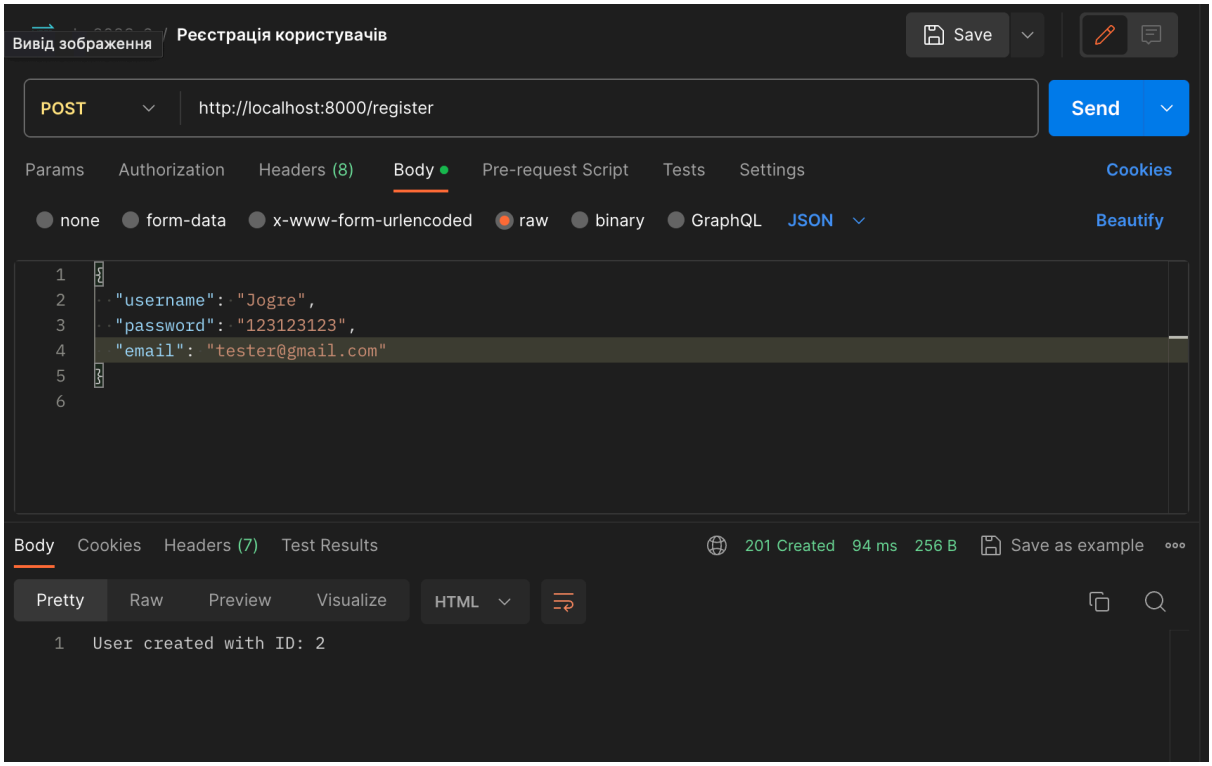
5. DELETE /devices/:id: Видаляє пристрій з бази даних за вказаним ID.



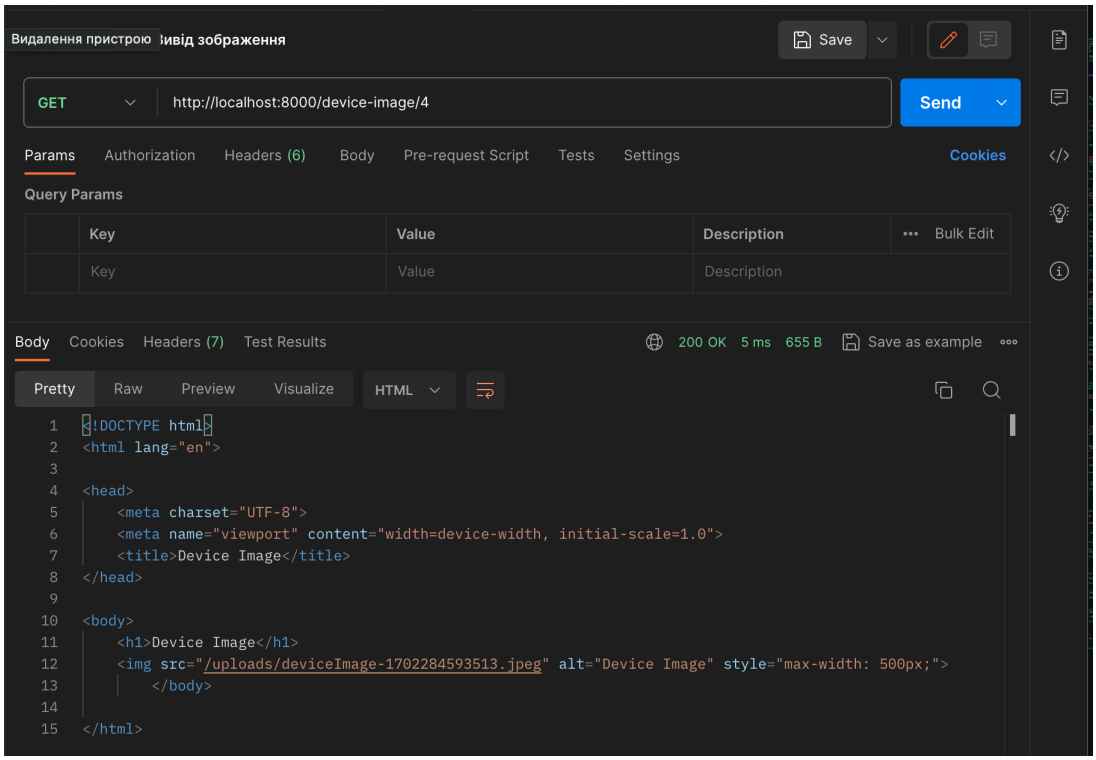
6. POST /upload/:deviceId: Дозволяє завантажити зображення для певного пристрою за його ID. Оновлює шлях до зображення пристрою в базі даних



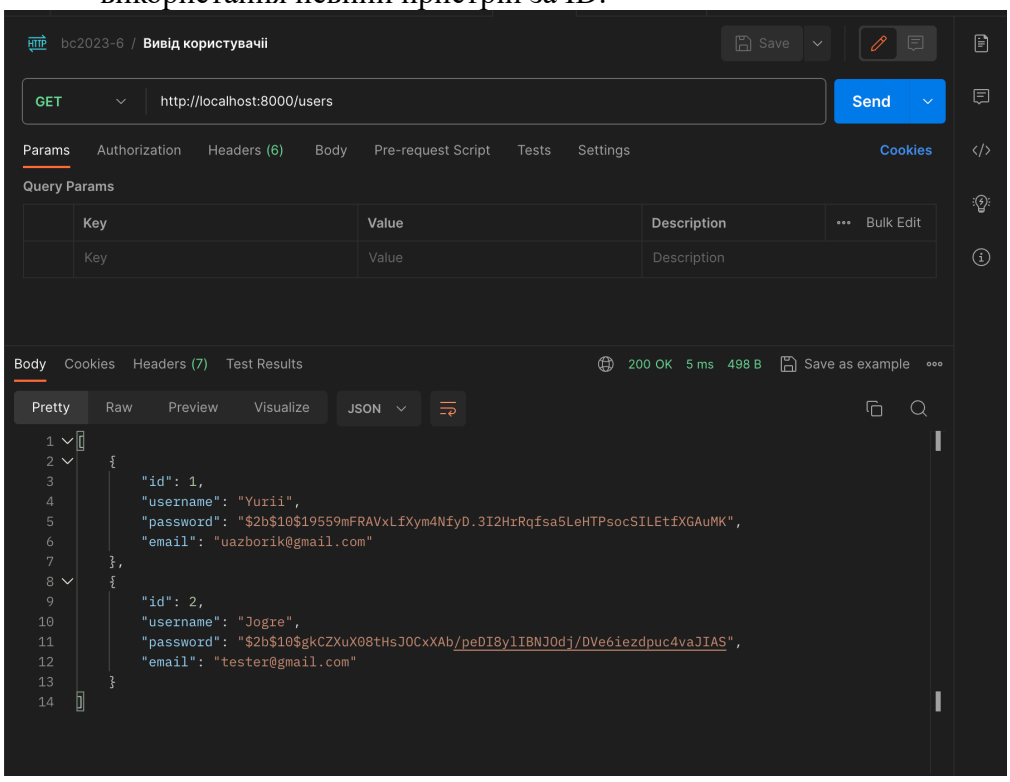
7. POST /register: Реєструє нового користувача з даними, які були надіслані.



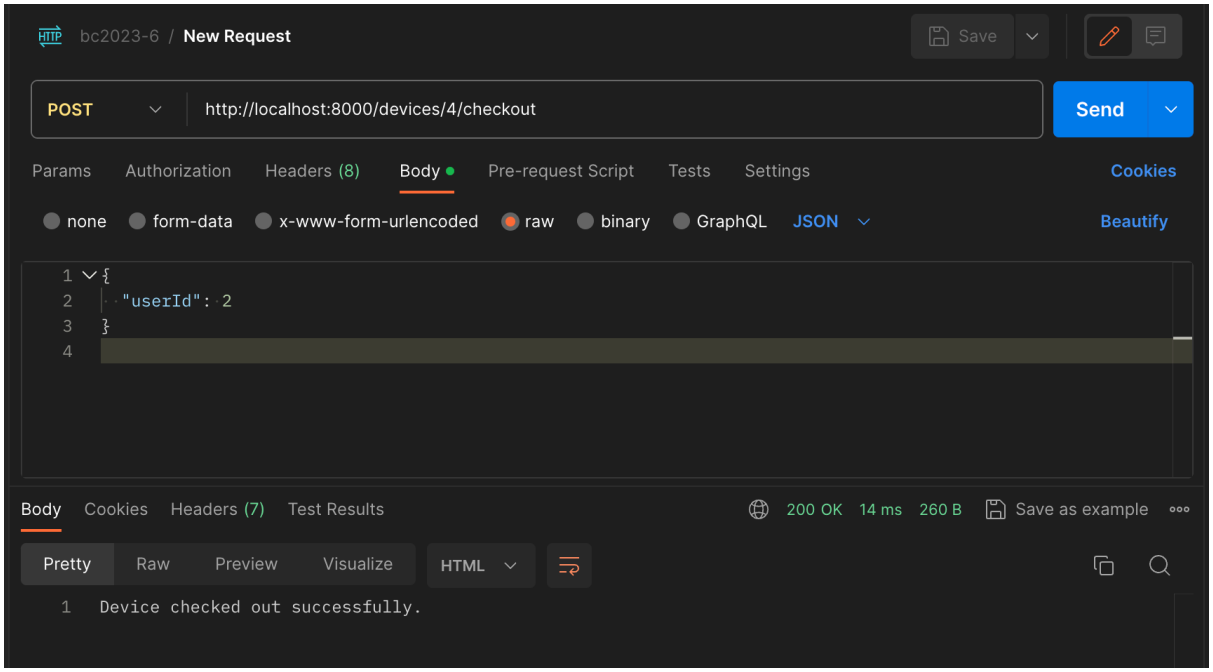
8. GET /users: Виводить список усіх користувачів, що зберігаються в базі даних. Повертає дані користувачів у форматі JSON.



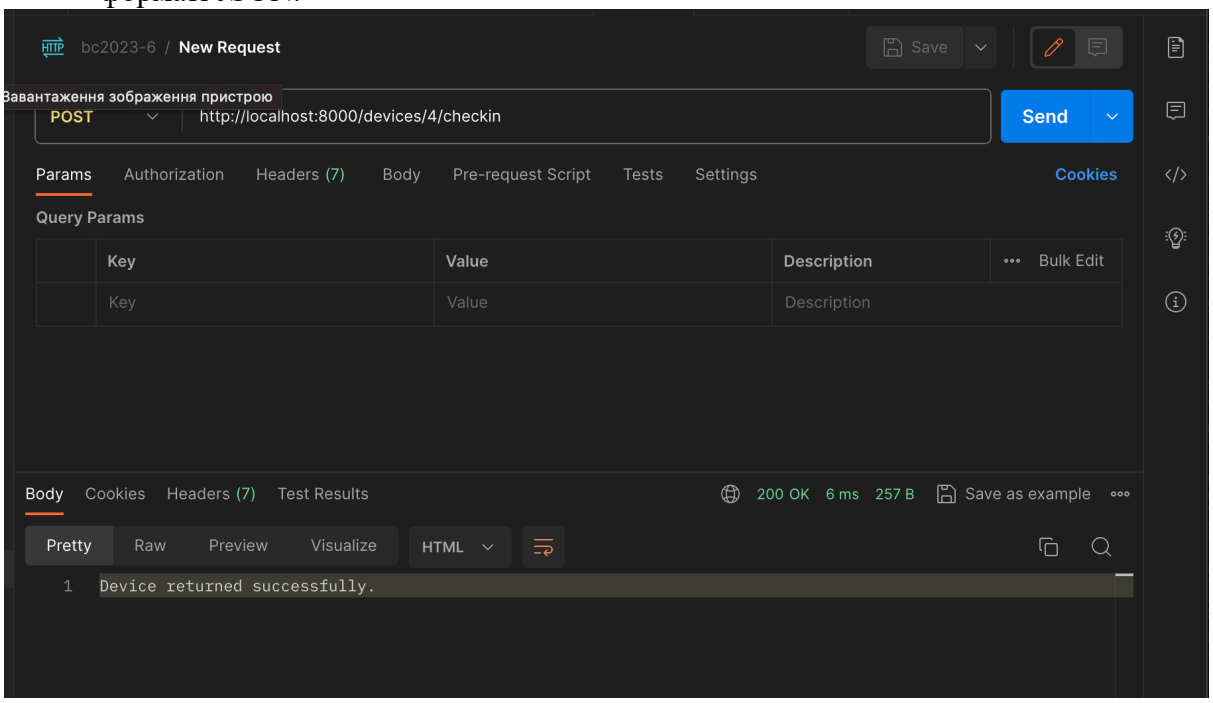
9. POST /devices/:id/checkout: Встановлює користувача як того, хто взяв у використання певний пристрій за ID.



10. POST /devices/:id/checkin: Повертає пристрій, який був взятий у користування, оновлюючи його статус у базі даних.



11. GET /user/:userId/devices: Виводить список пристроїв, які наразі знаходяться в користуванні у певного користувача за його ID. Повертає дані про пристрої у форматі JSON.



Висновок: При виконанні даної лабораторної роботи я створив сервер для завантаження девайсів з зображенням та реєстрацію користувачів, також написав документацію відповідних запитів використовуючи swagger.