

Praktikum 8 - Matakuliah Pilihan 1 (Web)

Program Studi: Teknik Informatika

Lakukan praktikum dibawah ini, dan buat screenshot untuk pembuktian mengerjakan setiap poin dengan mengisi tabel dibawah, kemudian tunjukan hasil akhir dari men-share repository github yang telah dibuat.

A. Membuat Server API dengan Express.js

1. Buat sebuah folder proyek API dengan nama **APIproject8**
2. Lakukan seperti pada praktikum 3
Ketik: `npm init -y`, setelah itu `npm install express`
3. Buat file `server.js`

```
JS server.js > ...
1  const express = require('express');
2  const app = express();
3  const PORT = 8001;
4
5  app.use(express.json());
6
7  app.get('/', (req, res) => {
8    res.send('Hello, World');
9  });
10
11 app.listen(PORT, () => {
12   console.log(`Server berjalan di http://localhost:${PORT}`);
13 });
14
```

4. Jalankan [server.js](#) dengan mengetik
Ketik: `node server.js`

B. Membuat Struktur MVC (Routes-Controller)

1. Buat folder **routes**, **controllers** dan **models**
2. Kemudian didalam folder routes buat sebuah file dengan nama [user.routes.js](#)

```
▼ PRAKTIKUM8
  ▼ controllers
    JS user.controller.js
  ▼ routes
    JS user.routes.js
    {} package.json
    JS server.js
```

3. Tulis kode program di file [user.routes.js](#) seperti pada gambar dibawah ini

```
JS server.js JS user.routes.js X
routes > JS user.routes.js > ...
1
2 const express = require('express');
3 const router = express.Router();
4 const userController = require('../controllers/user.controller');
5
6 // Routing standar REST API
7 router.get('/', userController.getAllUsers); //get all
8 router.get('/:id', userController.getUserById); //search by id
9 router.post('/', userController.createUser); //New data
10 router.put('/:id', userController.updateUser); //update by id
11 router.delete('/:id', userController.deleteUser); //delete
12
13 module.exports = router;
```

4. Buat file di dalam folder controllers dengan nama [user.controller.js](#)
5. Tulis kode program di dalam file [user.controller.js](#) seperti pada gambar dibawah ini

```
const User = require('../models/user.model'); //memanggil model

// GET semua user
exports.getAllUsers = (req, res) => {
  User.getAll((err, results) => { //ambil dari models
    if (err) return res.status(500).json({ error: err.message });
    res.json(results);
  });
};
```

Karena pada controller user tersebut require model bernama User, maka kita siapkan Model user, yang berkaitan dengan database.

6. Update file [server.js](#) dengan menambahkan kode berikut

```
8 // Routes
9 const userRoutes = require('./routes/user.routes');
10 app.use('/api/users', userRoutes);
```

Kode diatas pada file [server.js](#) untuk memberitahu ada routes bernama userRoutes dengan lokasi file di routes/user.routes (tidak perlu ditulis .js)

C. Membuat koneksi Database dengan Models

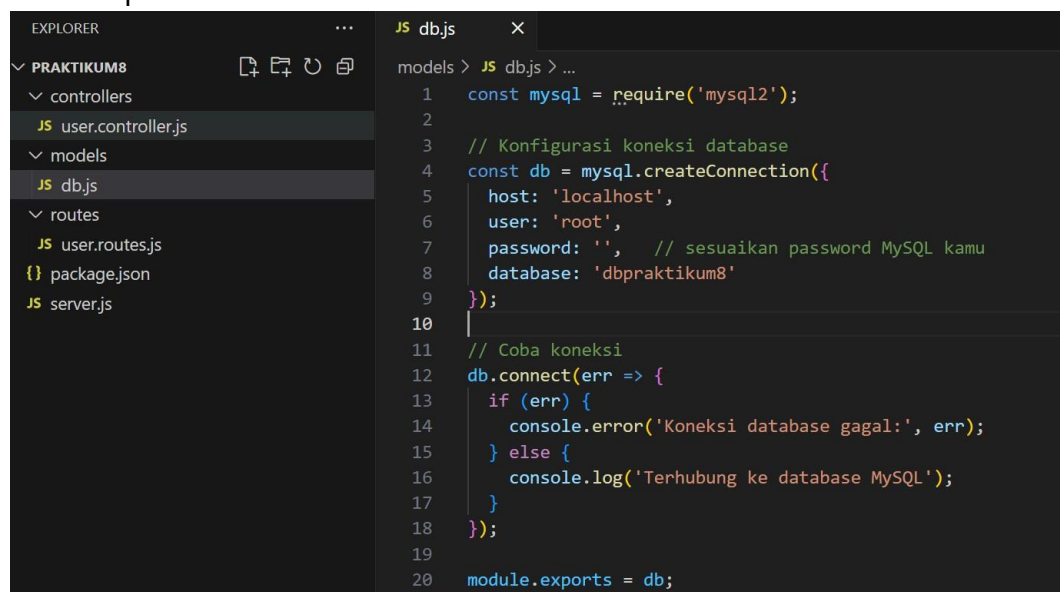
1. Nyalakan mysql service dan buatlah sebuah database dengan nama dbpraktikum8

```
CREATE DATABASE IF NOT EXISTS dbpraktikum8; CREATE TABLE IF NOT EXISTS users (  
id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR (100) NOT NULL, email  
VARCHAR (100) NOT NULL UNIQUE, password VARCHAR (255) DEFAULT NULL,  
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP);
```

2. Lalu masukan data dummy ke dalamnya
`INSERT INTO users (name, email, password) VALUES`

```
('Riska Safitri', 'riska@mail.com', '123456'),  
( 'Josephine', 'josep@mail.com', 'abcdef'),  
( 'Moh. Ilham', 'ilham@mail.com', 'qwerty');
```

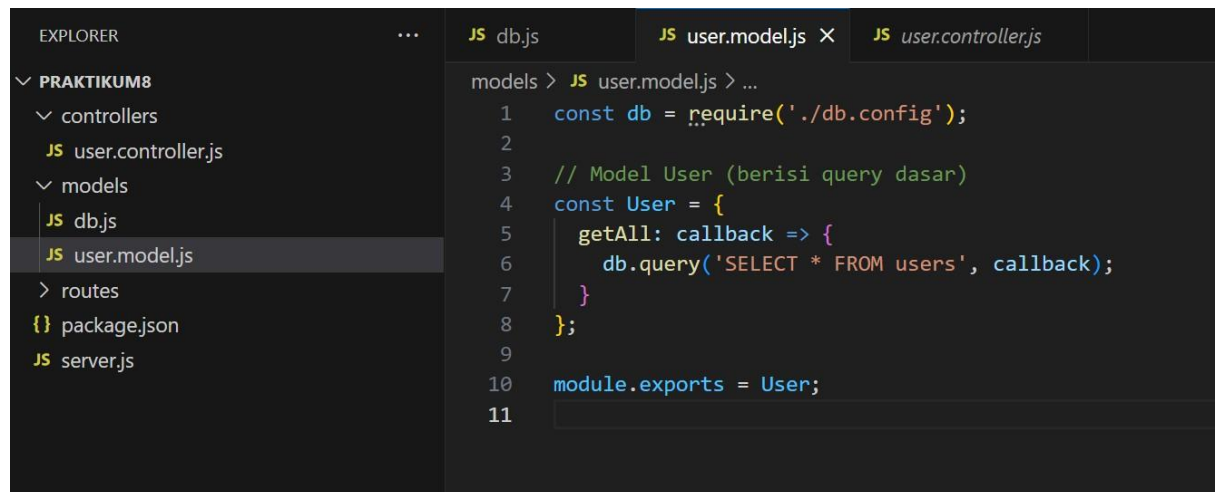
3. Jika database sudah terisi data di tabel users, lalu kita persiapkan kembali di [express.js](#)
4. Install Module mysql2 dengan menggunakan node. Masih di folder project ketik perintah berikut: `npm install express mysql2`
5. Kemudian buat sebuah file di dalam folder models, dengan nama [db.config.js](#) dan ketikan seperti berikut



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows a project structure with folders 'PRAKTIKUM8', 'controllers', 'models', and 'routes'. The 'models' folder is expanded, showing 'db.js'. The main editor area displays the code in 'db.js'.

```
1  const mysql = require('mysql2');  
2  
3  // Konfigurasi koneksi database  
4  const db = mysql.createConnection({  
5    host: 'localhost',  
6    user: 'root',  
7    password: '', // sesuaikan password MySQL kamu  
8    database: 'dbpraktikum8'  
9  });  
10  
11 // Coba koneksi  
12 db.connect(err => {  
13   if (err) {  
14     console.error('Koneksi database gagal:', err);  
15   } else {  
16     console.log('Terhubung ke database MySQL');  
17   }  
18 });  
19  
20 module.exports = db;
```

6. File [db.config.js](#) adalah sebagai class connector antara express dan database
7. Buat file lagi untuk model user, di dalam folder models. Dengan nama user.model.js



The screenshot shows the VS Code interface. On the left, the 'EXPLORER' sidebar displays a project structure under 'PRAKTIKUM8' with folders 'controllers' and 'models'. The 'models' folder is expanded, showing 'db.js', 'user.model.js' (selected), and 'server.js'. The main editor area shows the code in 'user.model.js' with line numbers 1 through 11. The code defines a 'User' model with a 'getAll' method that queries the database.

```
1 const db = require('./db.config');
2
3 // Model User (berisi query dasar)
4 const User = {
5   getAll: callback => {
6     db.query('SELECT * FROM users', callback);
7   }
8 };
9
10 module.exports = User;
11
```

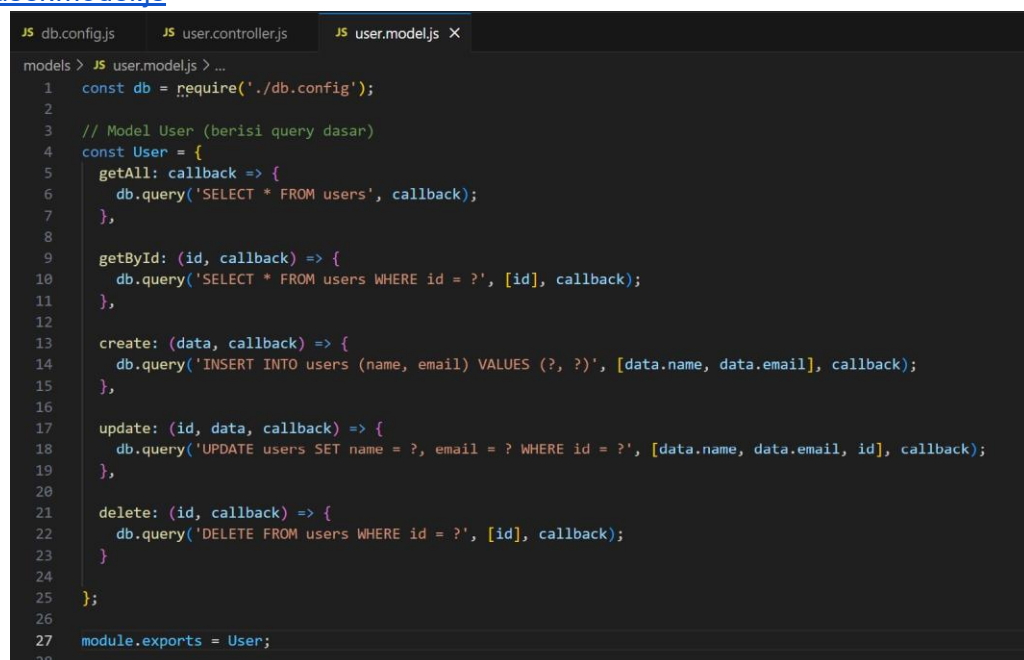
8. Jalankan atau restart ulang node [server.js](#)
(Pastikan mysql sudah running, user password mysql sudah benar)

D. Melakukan Test API

Gunakan browser/postman untuk mendapatkan data getAll users dengan mengunjungi endpoints `/api/users/`

E. Lengkapi Controllers dan Model

1. Tambahkan class untuk model baru, agar terhubung dengan controller. Ubah pada file [user.model.js](#)



The screenshot shows the VS Code interface with the 'user.model.js' file open. The code now includes methods for 'getAll', 'getById', 'create', 'update', and 'delete' users. The 'getAll' method is at line 5, 'getById' at line 9, 'create' at line 13, 'update' at line 17, and 'delete' at line 21. The file exports the 'User' model at line 27.

```
1 const db = require('./db.config');
2
3 // Model User (berisi query dasar)
4 const User = {
5   getAll: callback => {
6     db.query('SELECT * FROM users', callback);
7   },
8
9   getById: (id, callback) => {
10     db.query('SELECT * FROM users WHERE id = ?', [id], callback);
11   },
12
13   create: (data, callback) => {
14     db.query('INSERT INTO users (name, email) VALUES (?, ?)', [data.name, data.email], callback);
15   },
16
17   update: (id, data, callback) => {
18     db.query('UPDATE users SET name = ?, email = ? WHERE id = ?', [data.name, data.email, id], callback);
19   },
20
21   delete: (id, callback) => {
22     db.query('DELETE FROM users WHERE id = ?', [id], callback);
23   }
24 };
25
26
27 module.exports = User;
28
```

2. Tambahkan class baru untuk routes yang sudah dipersiapkan lainnya, bisa dilihat pada kode program dibawah ini

File: user.controller.js

```
// GET user by ID
exports.getUserById = (req, res) => {
  const { id } = req.params;
  User.getById(id, (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    if (results.length === 0) return res.status(404).json({ message: 'User tidak ditemukan' });
    res.json(results[0]);
  });
};

// POST user baru
exports.createUser = (req, res) => {
  const data = req.body;
  User.create(data, (err, result) => {
    if (err) return res.status(500).json({ error: err.message });
    res.status(201).json({ id: result.insertId, ...data });
  });
};

// PUT update user
exports.updateUser = (req, res) => {
  const { id } = req.params;
  const data = req.body;
  User.update(id, data, (err, result) => {
    if (err) return res.status(500).json({ error: err.message });
    if (result.affectedRows === 0) return res.status(404).json({ message: 'User tidak ditemukan' });
    res.json({ message: 'User berhasil diupdate' });
  });
};

// DELETE user
exports.deleteUser = (req, res) => {
  const { id } = req.params;
  User.delete(id, (err, result) => {
    if (err) return res.status(500).json({ error: err.message });
    if (result.affectedRows === 0) return res.status(404).json({ message: 'User tidak ditemukan' });
    res.json({ message: 'User berhasil dihapus' });
  });
};
```

F. Melakukan Test API secara Lengkap

Dengan menggunakan POSTMAN, lakukan pengujian berikut:

1. Menguji endpoint /
2. Menguji endpoint /api/users (Method: GET)
3. Menguji endpoint /api/users/1 (Method: GET)
4. Menguji endpoint /api/users (Method: POST)

Tambah body -> raw -> JSON

```
{
  "name": "Budi Santoso",
  "email": "budi@example.com"
}
```

5. Menguji /api/users/2 (Method: PUT)

Masukan Body -> raw -> JSON

```
{  
  "name": "Joe Taslim",  
  "email": "jojo@example.com"  
}
```

6. Menguji /api/users/3 (Method: DELETE)

G. Github + Visual Code

1. Buat proyek di Github dengan nama **Latihan8**

git init

git add

.

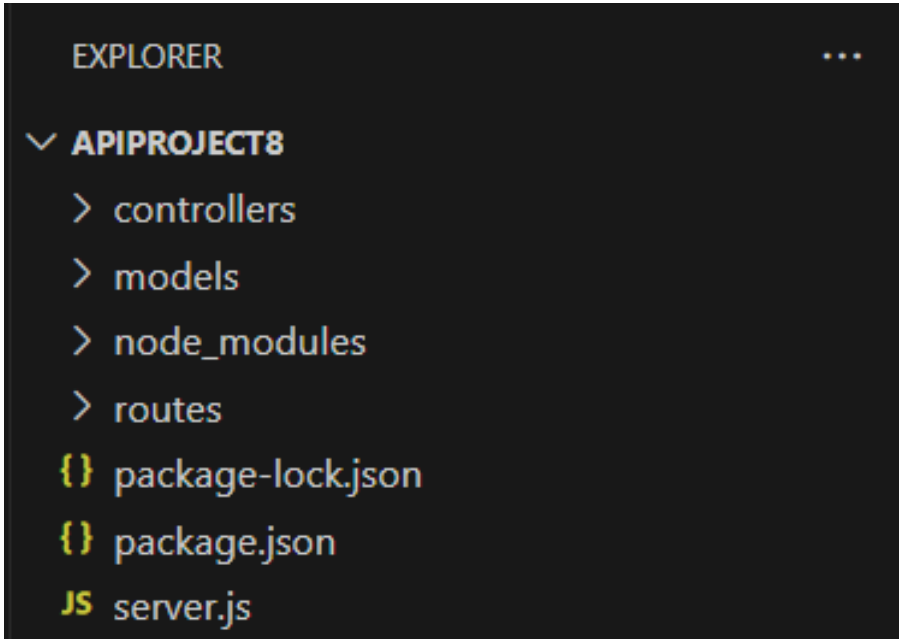
git commit -m "first commit"

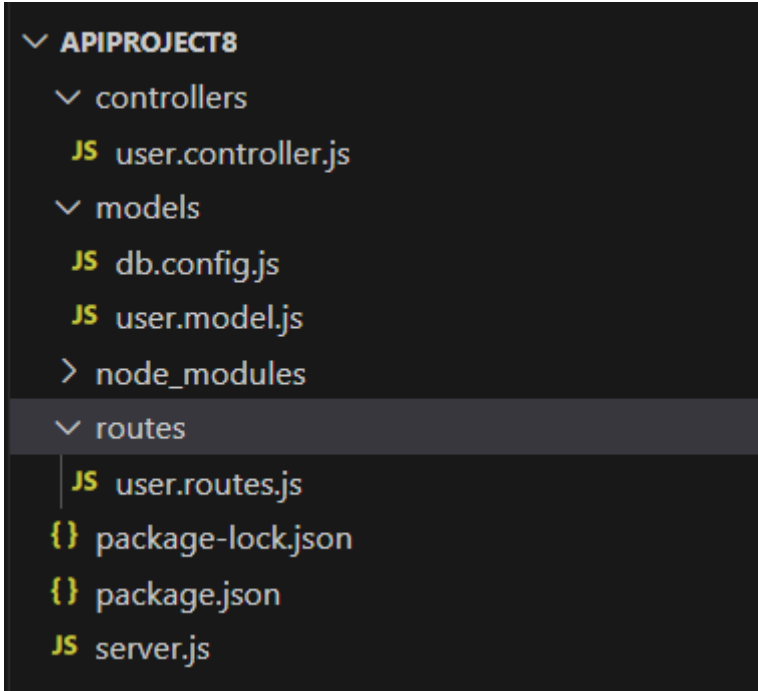
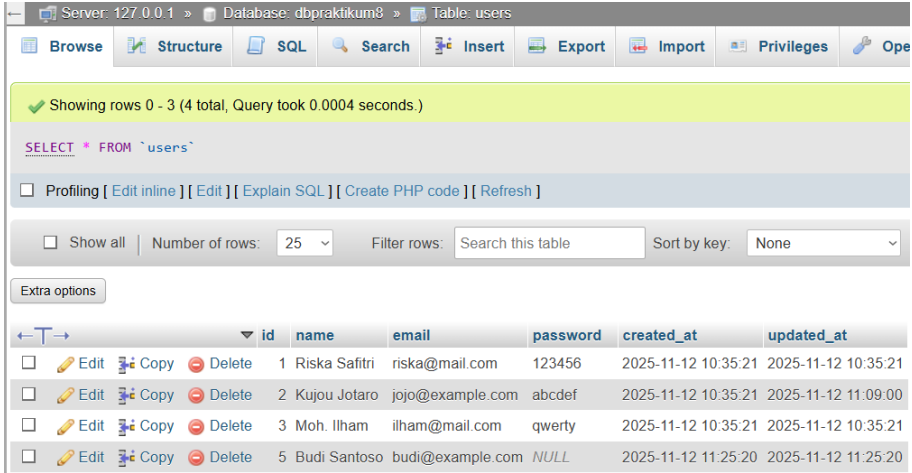

git branch -M main

git remote add origin <https://github.com/agunghakase/Latihan8.git>

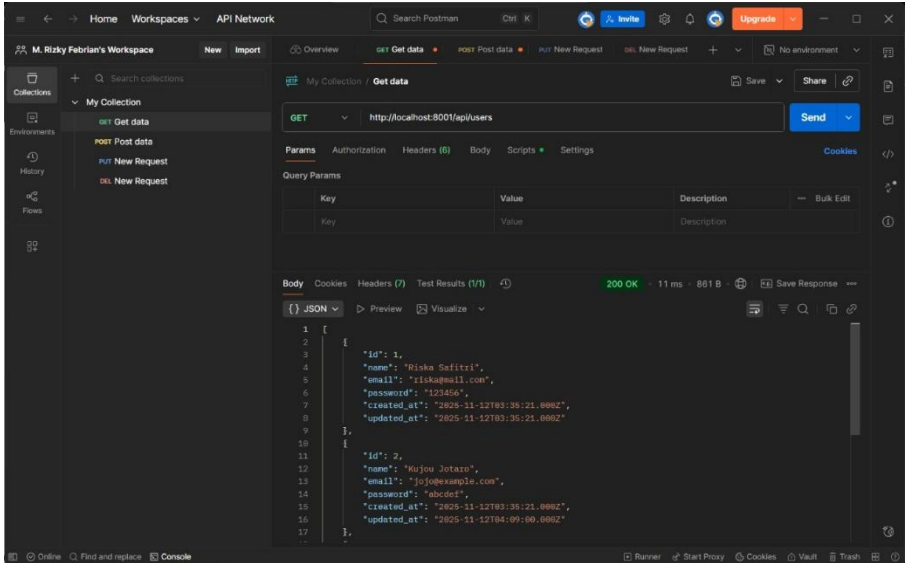
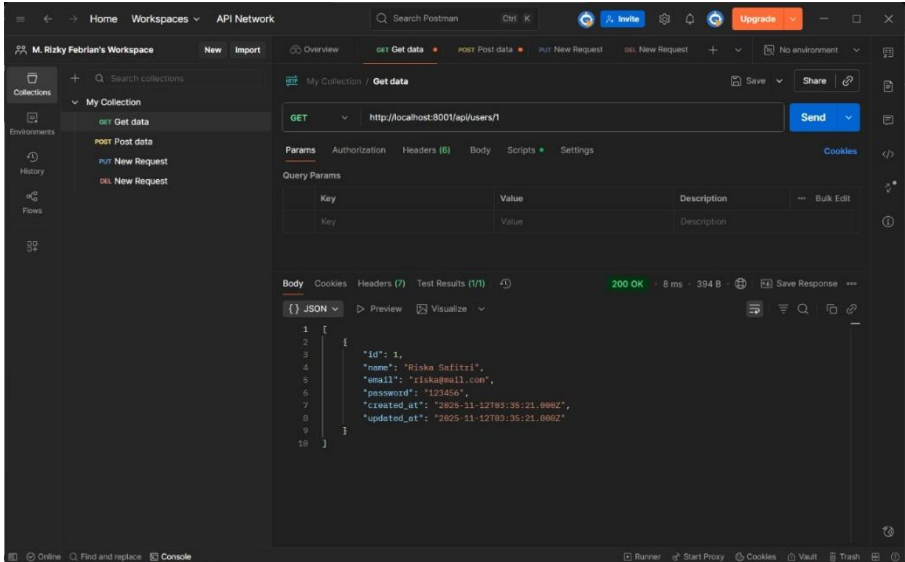
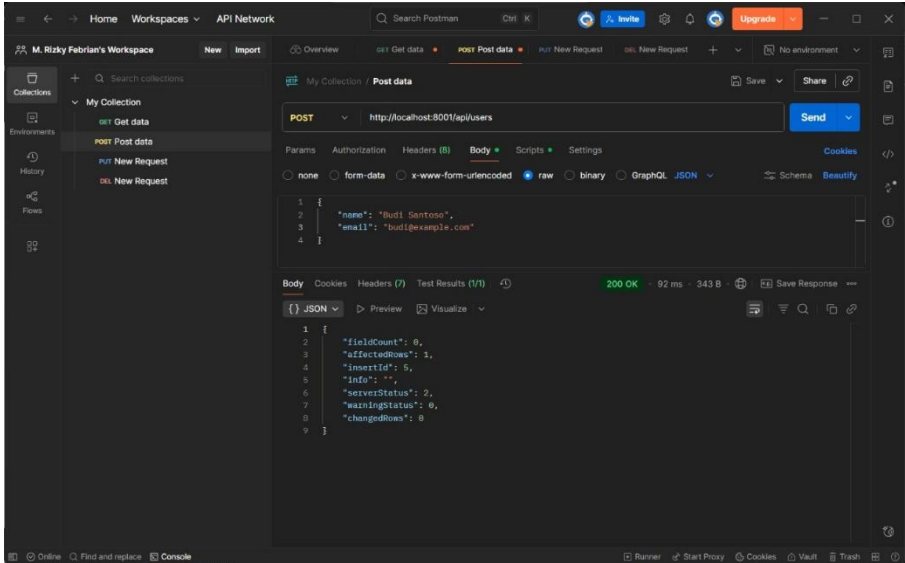
git push -u origin main

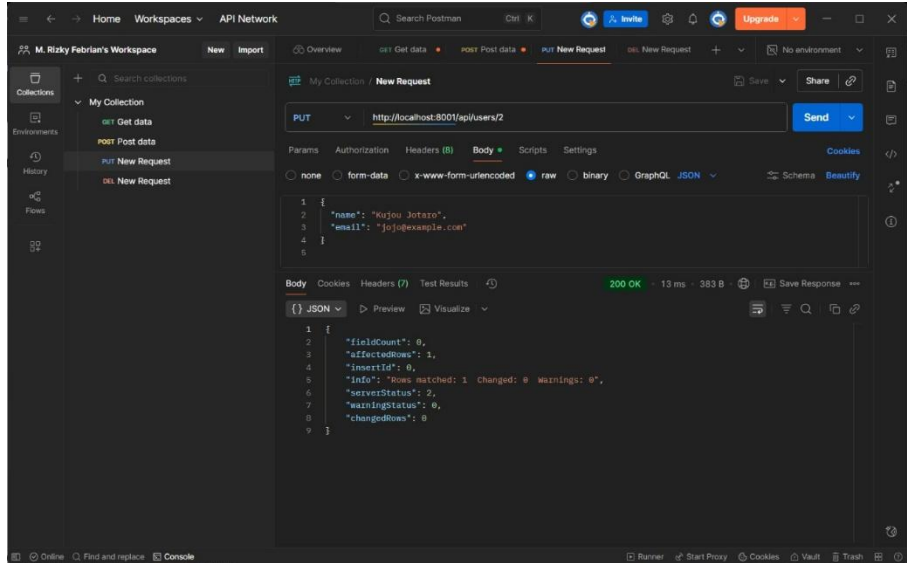
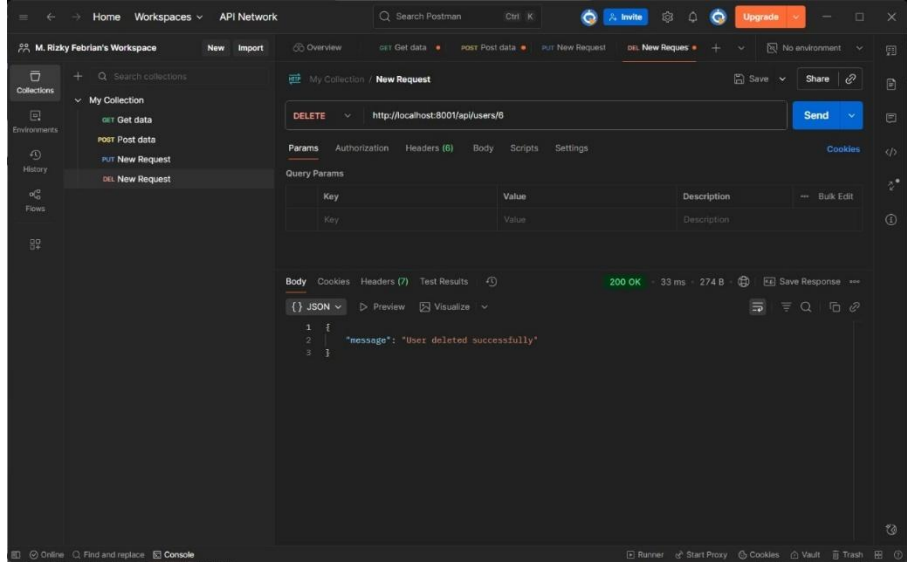
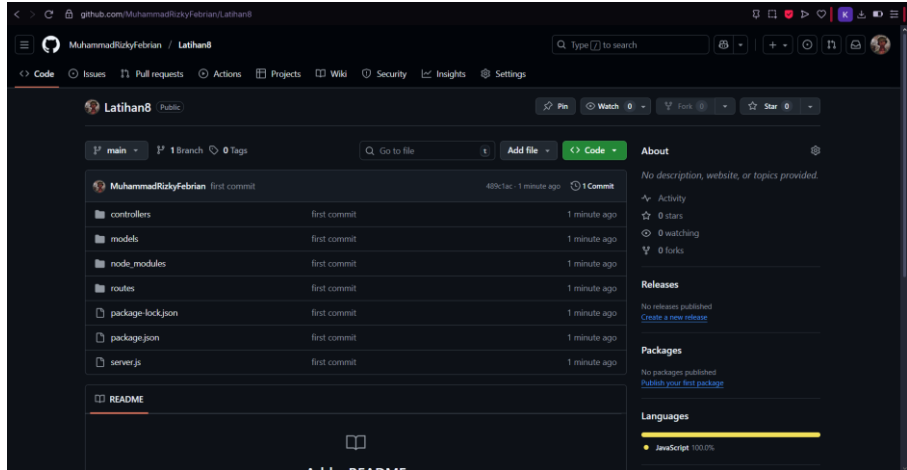
Hasil Pengerjaan

No.	Instruksi	Screenshot	Kendala /Saran
A.	Membuat Server API dengan Express.js		
1.	Step 1-4	 <pre> EXPLORER └─ APIPROJECT8 ├── controllers ├── models ├── node_modules ├── routes ├── package-lock.json ├── package.json └── server.js </pre>	
B.	Membuat Struktur MVC (Routes-Controller)		

1.	Step 1-6		
C.	Membuat koneksi Database dengan Models		
1.	Step 1-2		
2.	Step 3-8		
D.	Melakukan Test API		

1.	Tes Postman	 <p>The screenshot shows the Postman interface with a workspace named 'M. Rizky Febrina's Workspace'. A collection named 'My Collection' contains a 'Get data' request. The request is a GET method to the URL 'http://localhost:8001/api/users'. The response is a 200 OK status with a response time of 11 ms and a body size of 861 B. The response body is a JSON array containing two user objects. The first user has an ID of 1, name 'Riska Sefitri', email 'riskas@gmail.com', password '123456', created_at '2025-11-12T03:35:21.000Z', and updated_at '2025-11-12T03:35:21.000Z'. The second user has an ID of 2, name 'Kujou Jotaro', email 'jojooexample.com', password 'abcdet', created_at '2025-11-12T03:35:21.000Z', and updated_at '2025-11-12T03:35:21.000Z'.</p>	
F.	Melakukan Test API secara Lengkap		
1.	Test 1	 <p>The screenshot shows the Postman interface with the same workspace and collection. The 'Get data' request is now a GET method to the URL 'http://localhost:8001/'. The response is a 200 OK status with a response time of 16 ms and a body size of 240 B. The response body is an HTML document with the text 'Hello, world!'.</p>	

2.	Test 2	
3.	Test 3	
4.	Test 4	

5.	Test 5	 <p>The screenshot shows the Postman interface with a PUT request to <code>http://localhost:8001/api/users/2</code>. The request body is a JSON object: <code>{ "name": "Kusju Jotano", "email": "jotano@example.com" }</code>. The response is a 200 OK status with a detailed JSON body: <code>{ "fieldCount": 0, "affectedRows": 1, "insertId": 0, "info": "Rows matched: 1 Changed: 0 Warnings: 0", "serverStatus": 2, "warningStatus": 0, "changedRows": 0 }</code>.</p>	
6.	Test 6	 <p>The screenshot shows the Postman interface with a DELETE request to <code>http://localhost:8001/api/users/6</code>. The response is a 200 OK status with a JSON body: <code>{ "message": "User deleted successfully" }</code>.</p>	
G.	Github dan Viscode		
1.	Github	 <p>The screenshot shows a GitHub repository page for 'Latihan8' by user 'MuhammadRizkyFebrian'. The repository has 1 branch and 0 tags. The file structure includes folders <code>controllers</code>, <code>models</code>, <code>node_modules</code>, and <code>routes</code>, and files <code>package-lock.json</code>, <code>package.json</code>, and <code>server.js</code>. The README section is visible at the bottom.</p>	