

Nama: Arifah Husaini

NIM: 3312411097

Kelas: IF3A Malam

Link Github: <https://github.com/arifah336/Latihan8.git>

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

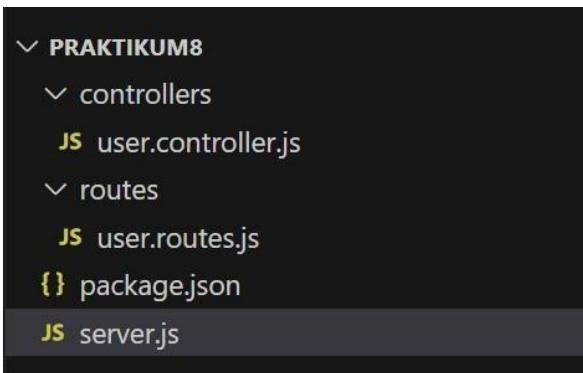
1. Buat sebuah folder proyek API dengan nama **APIproject8**
2. Lakukan seperti pada praktikum 3
Ketik: `npm init -y`, setelah itu `npm install express`
3. Buat file `server.js`

```
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`



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

```
JS server.js JS user.routesjs ×
routes > JS user.routesjs > ...
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

```

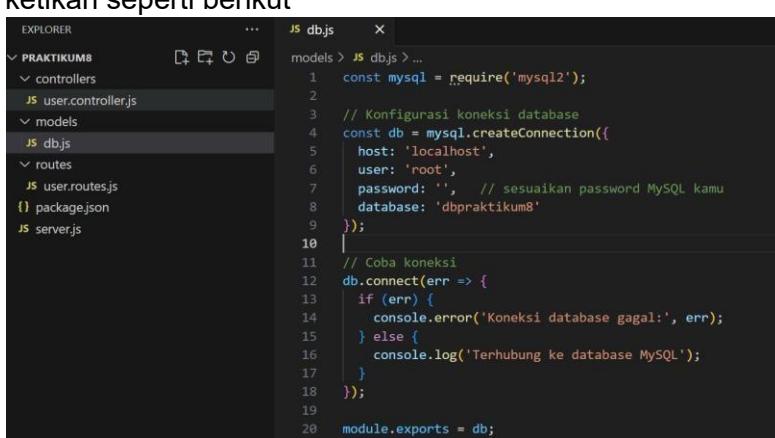
    /
8 // Routes
9 const userRoutes = require('./routes/user.routes');
10 app.use('/api/users', userRoutes);

```

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, updated_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 a code editor with the following file structure:

- EXPLORER**: Shows files in the project: PRAKTIKUM8 (controllers, models), routes, user.controller.js, user.routes.js, package.json, and server.js.
- JS db.js**: The code editor tab contains the following code:

```

models > JS 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 a code editor with the following structure:

- EXPLORER** pane on the left:
 - PRAKTIKUM
 - controllers
 - JS user.controller.js**
 - models
 - JS db.js**
 - JS user.model.js** (highlighted)
 - routes
 - { } package.json
 - JS server.js
- JS db.js** tab is active.
- JS user.model.js** tab is open and contains the following code:

```

models > JS user.model.js > ...
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

```

- JS user.controller.js** tab is present but inactive.
- JS user.controller.js** tab is present but inactive.

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

C. Melakukan Test API

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

D. Lengkapi Controllers dan Model

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

The screenshot shows the `user.model.js` file with the following updated code:

```

models > JS user.model.js > ...
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' });
};
```

E. 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)

F. Github + Visual Code

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

```
git init
```

```
git add
```

```
git commit -m "first commit"
```

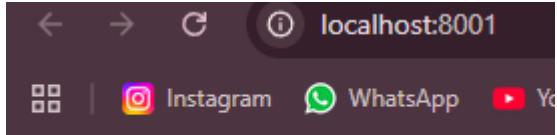
```
git branch -M main
```

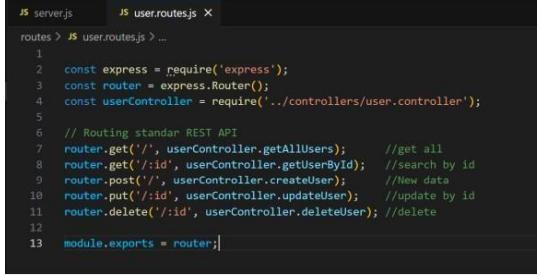
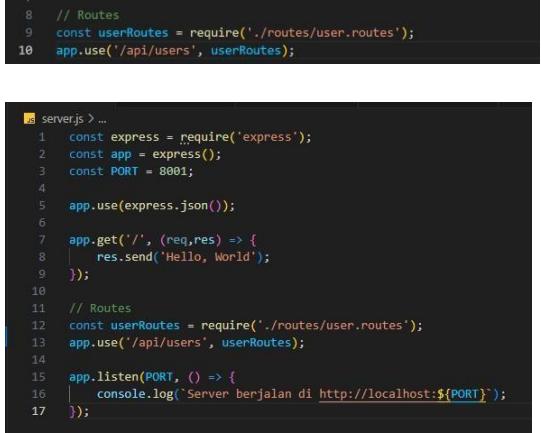
```
git remote add origin https://github.com/agunghakase/Latihan8.git
```

```
git push -u origin main
```

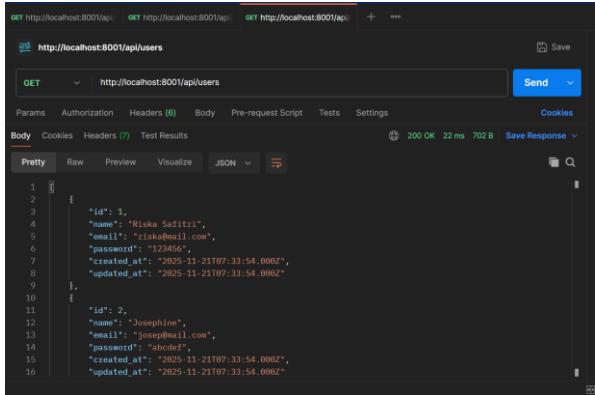
Hasil Pengerjaan

No.	Instruksi	Screenshot	Kendala/Saran
A.	Membuat Server API dengan Express.js		
1.	Membuat folder proyek API dengan nama APIproject8, kemudian jalankan perintah <i>npm init -y</i>	<pre>C:\Users\hp>cd APIproject8 C:\Users\hp\APIproject8>npm init -y Wrote to C:\Users\hp\APIproject8\package.json: { "name": "apiproject8", "version": "1.0.0", "description": "", "main": "index.js", "scripts": { "test": "echo \"Error: no test specified\" && exit 1" }, "keywords": [], "author": "", "license": "ISC", "type": "commonjs" }</pre>	
2.	Menginstall express.js di folder APIproject8 dengan perintah <i>npm install express</i>	<pre>C:\Users\hp\APIproject8>npm install express added 68 packages, and audited 69 packages in 4s 16 packages are looking for funding run `npm fund` for details found 0 vulnerabilities</pre>	
3.	Pada folder APIproject8 buat file bernama server.js lalu ketikkan kode program seperti gambar disamping	<pre>server.js ✘ APIproject8 > ↵ 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]);</pre>	

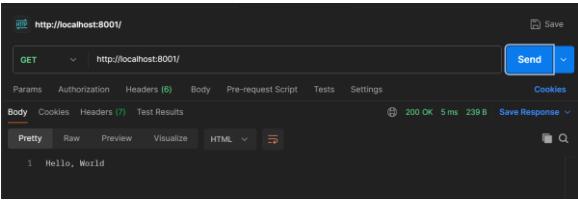
4.	Jalankan dengan mengetik node server.js pada terminal dan gunakan browser untuk mengecek hasilnya		
----	---	--	--

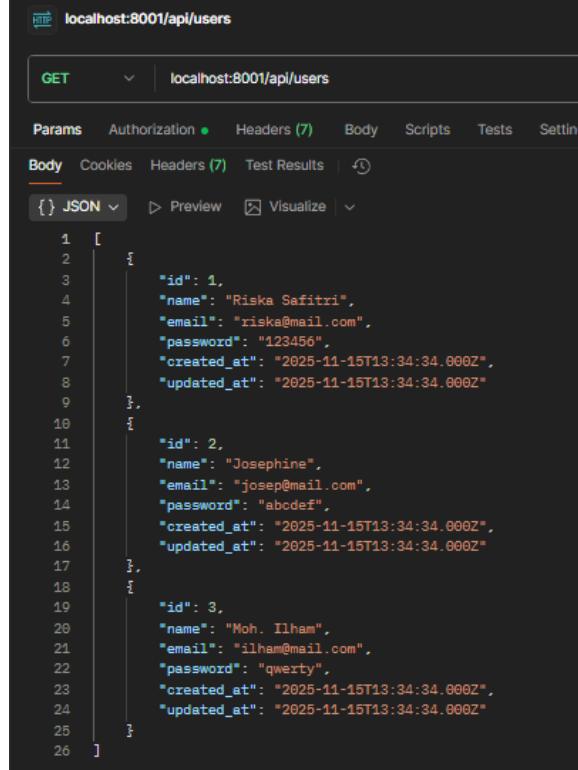
B.	Membuat Struktur MVC (Routers – Controller)		
1.	Buat folder routes, controllers, dan models. Di dalam folder routes buat file dengan nama user.routes.js dan ketik kode program seperti gambar disamping		
2.	Buat file dalam folder controllers dengan nama user.controller.js dan ketik kode program seperti gambar disamping		
3.	Update file server.js dengan menambahkan kode seperti gambar disamping		
C.	Membuat Koneksi Database dengan Models		

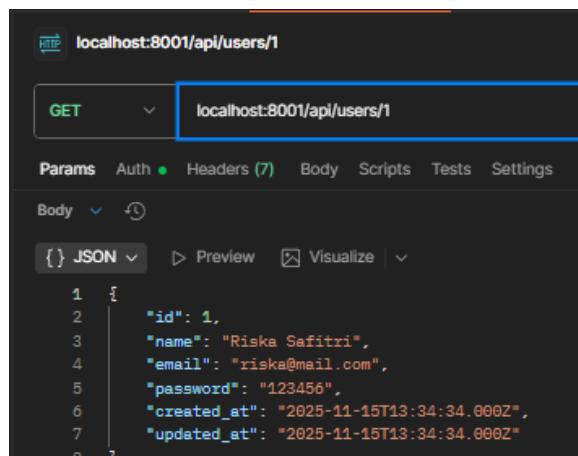
1.	Nyalakan mysql service dan buatlah sebuah database dengan nama dbpraktikum8	<pre>CREATE DATABASE IF NOT EXISTS dbpraktikum8; CREATE TABLE IF NOT EXISTS users (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(100) NOT NULL UNIQUE, password VARCHAR(255) DEFAULT NULL, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP);</pre>	
2.	Lalu masukan data dummy ke dalamnya	<pre>INSERT INTO users (name, email, password) VALUES ('Riska Safitri', 'riska@mail.com', '123456'), ('Josephine', 'josep@mail.com', 'abcdef'), ('Moh. Ilham', 'ilham@mail.com', 'qwerty');</pre>	
3.	Setelah itu, Install Module mysql2 dengan perintah <i>npm install express mysql2</i>	<pre>C:\Users\hp\APIproject8>npm install express mysql2 added 12 packages, and audited 81 packages in 4s 17 packages are looking for funding run `npm fund` for details found 0 vulnerabilities</pre>	
4.	Buat sebuah file dalam folder models dengan nama db.config.js dan ketik kode program seperti gambar disamping	<pre>Project8 > models > js db.config.js > ... 1 const mysql = require('mysql2'); 2 3 const db = mysql.createConnection({ 4 host: 'localhost', 5 user: 'root', 6 password: '', 7 database: 'dbpraktikum8', 8 port: 3307 9 }); 10 11 db.connect(err => { 12 if (err) { 13 console.error('Koneksi database gagal:', err); 14 } else { 15 console.log('Terhubung ke database MySQL'); 16 } 17 }); 18 19 module.exports = db;</pre>	
5.	Buat file untuk model user di dalam folder models dengan nama user.model.js	<pre>Project8 > models > js user.model.js > ... 1 const db = require('../db.config'); 2 3 const User = [4 getAll: callback => { 5 db.query('SELECT * FROM users', callback); 6 } 7]; 8 9 module.exports = User;</pre>	
D.	Melakukan Test API		

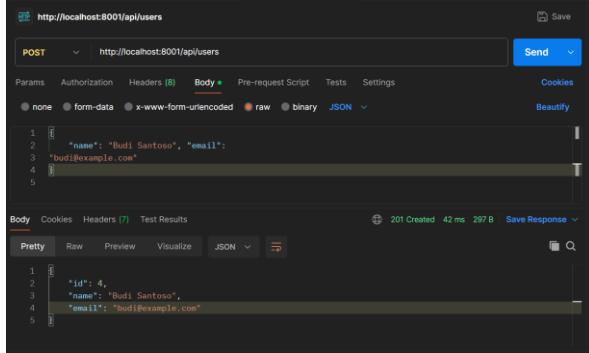
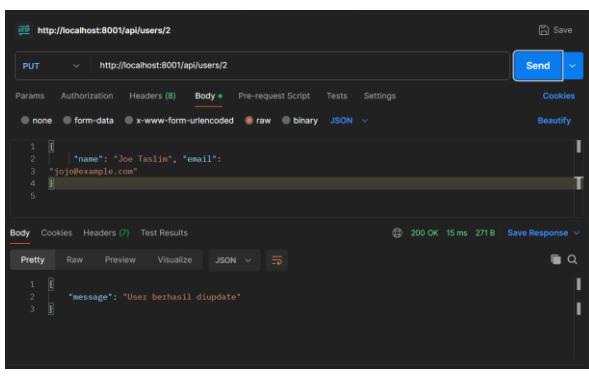
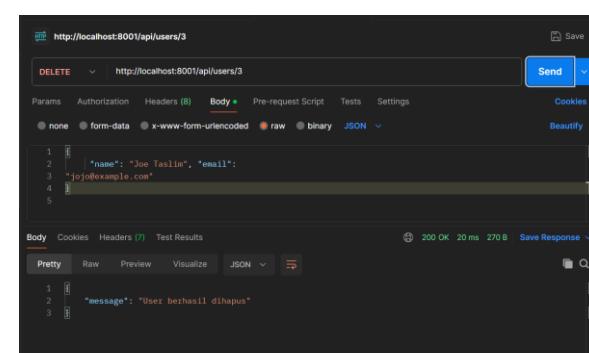
	Jalankan perintah node server.js dan gunakan postman untuk mendapatkan data getAll users dengan mengunjungi endpoints /api/users		
1.			
E.	Lengkapi Controllers dan Model		

1.	Tambahkan class untuk model baru, agar terhubung dengan controller. Tambahkan kode program seperti gambar disamping pada file user.model.js	<pre>getById: (id, callback) => { db.query("SELECT * FROM users WHERE id = ?", [id], callback); }, create: (data, callback) => { db.query("INSERT INTO users (name, email) VALUES (?, ?)", [data.name, data.email], callback); }, update: (id, data, callback) => { db.query("UPDATE users SET name = ?, email = ? WHERE id = ?", [data.name, data.email, id], callback); }, delete: (id, callback) => { db.query("DELETE FROM users WHERE id = ?", [id], callback); }</pre>	
2.	Tambahkan juga kode program seperti gambar disamping pada file user.controller.js	<pre>// GET user by ID exports.getById = (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.createUser(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.updateUser(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.deleteUser(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' }); }); };</pre>	
F.	Melakukan Test API secara Lengkap		

			
1.	Menguji endpoint /		

			
2.	Menguji endpoint /api/users (Method: GET)		

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

4.	<p>Menguji endpoint /api/users (Method: POST)</p> <p>Tambah body -> raw -> JSON { "name": "Budi Santoso", "email": "budi@example.com" }</p>	 <pre> POST http://localhost:8001/api/users { "name": "Budi Santoso", "email": "budi@example.com" } 201 Created { "id": 4, "name": "Budi Santoso", "email": "budi@example.com" } </pre>	
5.	<p>Menguji /api/users/2 (Method: PUT) Masukan Body -> raw -> JSON { "name": "Joe Taslim", "email": "jojo@example.com" }</p>	 <pre> PUT http://localhost:8001/api/users/2 { "name": "Joe Taslim", "email": "jojo@example.com" } 200 OK { "message": "User berhasil diupdate" } </pre>	
6.	<p>Menguji /api/users/3 (Method: DELETE)</p>	 <pre> DELETE http://localhost:8001/api/users/3 { "name": "Joe Taslim", "email": "jojo@example.com" } 200 OK { "message": "User berhasil dihapus" } </pre>	