Praktikum 8 - Matakuliah Pilihan 1 (Web)
Program Studi: Teknik Informatika

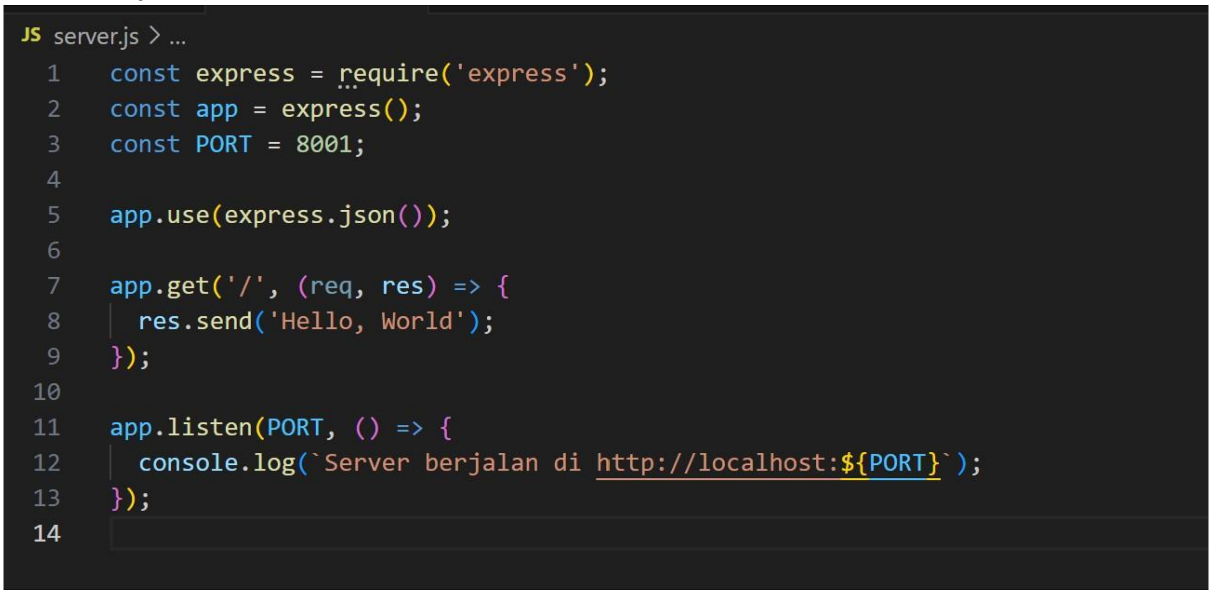
Lakukan praktikum dibawah ini, dan buat screenshot untuk pembuktian mengerjakan setiap poin dengan mengisi tabel dibawah, kemudian tunjukan hasil akhir dari men-share repository github yang telah dibuat.

A. Membuat Server API dengen Expess.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



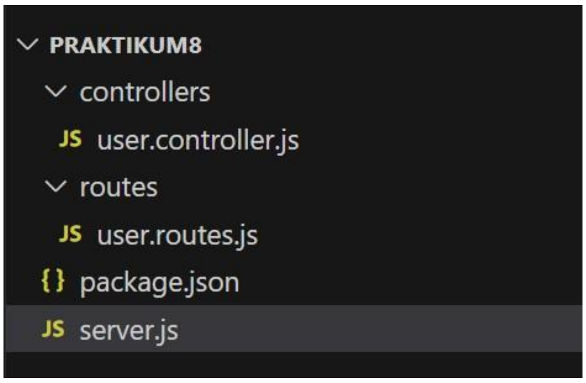
```
server.js > ...
const express = require('express');
const app = express();
const PORT = 8001;

app.use(express.json());

app.get('/', (req, res) => {
  res.send('Hello, World');
});

app.listen(PORT, () => {
  console.log(`Server berjalan di http://localhost:${PORT}`);
});
```

4. Jalankan server.js dengan mengetik Ketik:

   node server.js

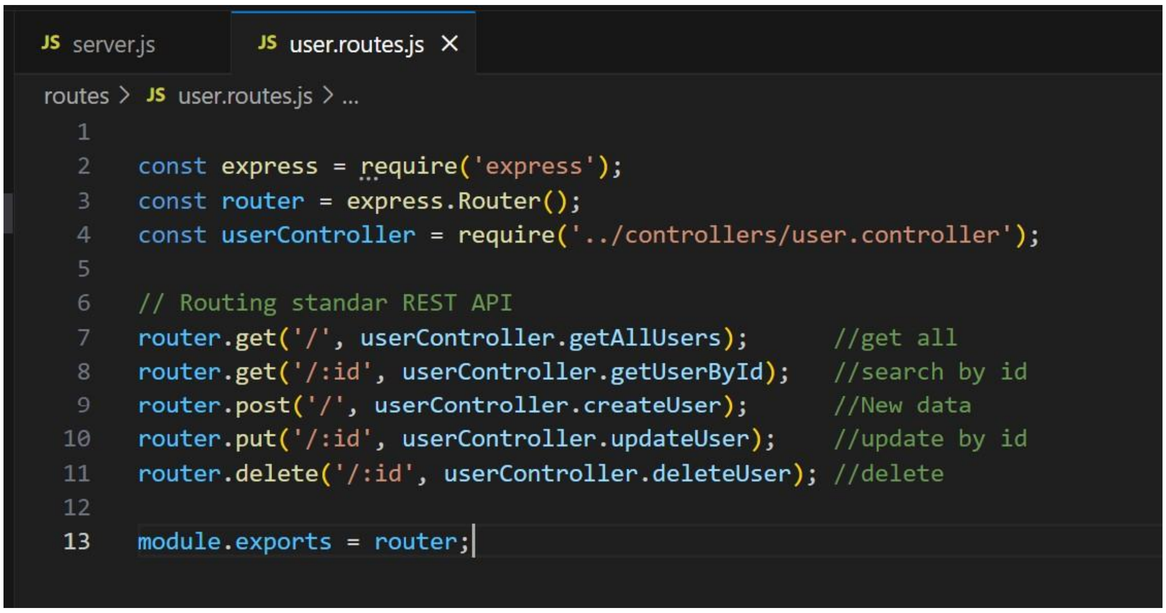
B. Membuat Struktur MVC (Routes-Controller) 1. Buat folder **routes, controllers** dan **models**

2. Kemudian didalam folder routes buat sebuah file dengan nama user.routes.js



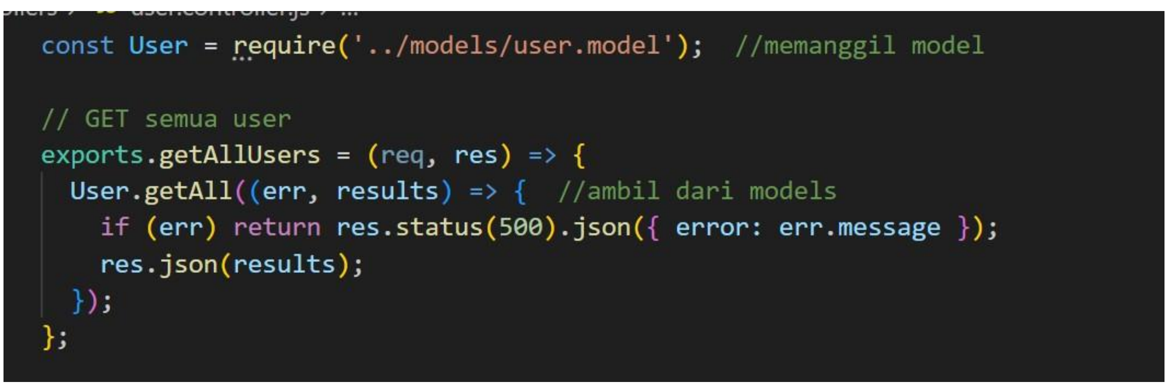
```
PRAKTIKUM8
  controllers
    user.controller.js
  routes
    user.routes.js
  package.json
  server.js
```

3. Tulis kode program di file user.routes.js seperti pada gambar dibawah ini



```
const express = require('express');
const router = express.Router();
const userController = require('../controllers/user.controller');

// Routing standar REST API
router.get('/', userController.getAllUsers);      //get all
router.get('/:id', userController.getUserById);   //search by id
router.post('/', userController.createUser);      //New data
router.put('/:id', userController.updateUser);    //update by id
router.delete('/:id', userController.deleteUser); //delete

module.exports = router;
```

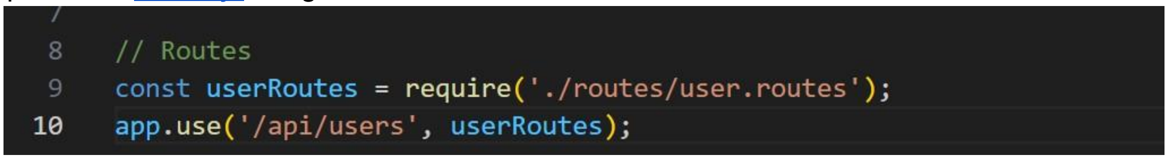
4. Buat file di dalam folder controllers dengan nama user.controller.js
5. Tulis kode program di dalam file user.controller.js seperti pada gambar dibawah ini



```
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



```
// Routes
const userRoutes = require('./routes/user.routes');
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);

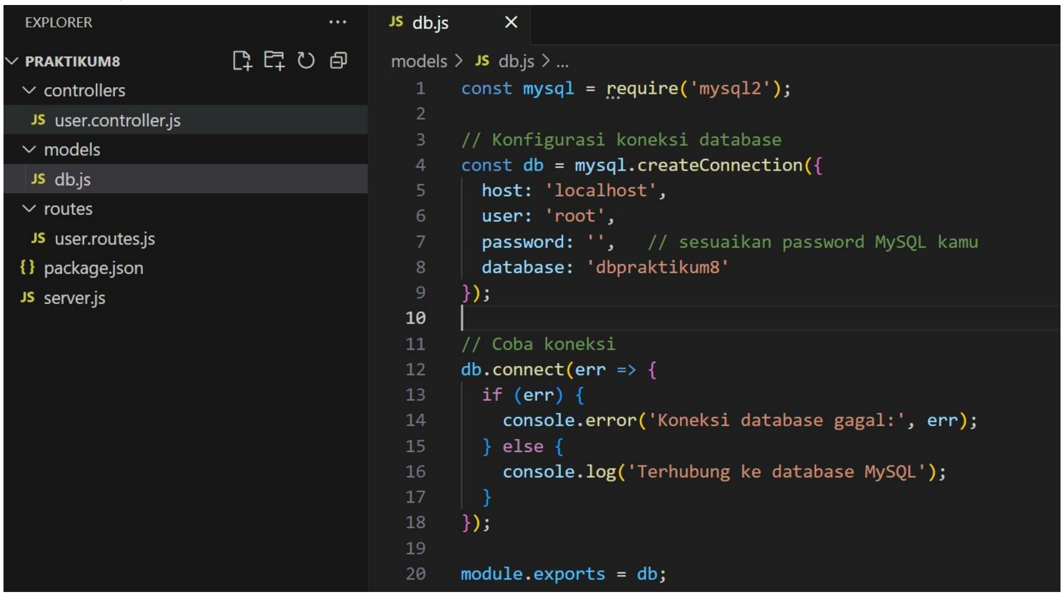
2. Lalu masukan data dummy ke dalamnya

   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

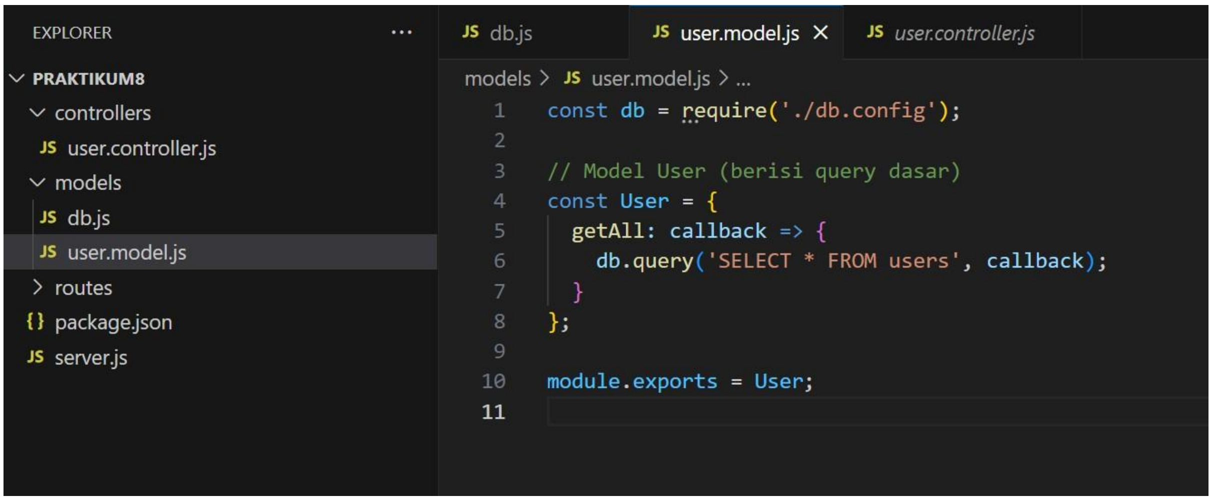
4. Install Module mysql2 dengan menggunakan node. Masih di folder project ketik perintah berikut: npm install express mysql2

5. Kemudian buat sebuah file di dalam folder models, dengan nama db.config.js dan ketikan seperti berikut



```
const mysql = require('mysql2');

// Konfigurasi koneksi database
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',   // sesuaikan password MySQL kamu
  database: 'dbpraktikum8'
});

// Coba koneksi
db.connect(err => {
  if (err) {
    console.error('Koneksi database gagal:', err);
  } else {
    console.log('Terhubung ke database MySQL');
  }
});

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



```javascript
const db = require('./db.config');

// Model User (berisi query dasar)
const User = {
  getAll: callback => {
    db.query('SELECT * FROM users', callback);
  }
};

module.exports = User;
```

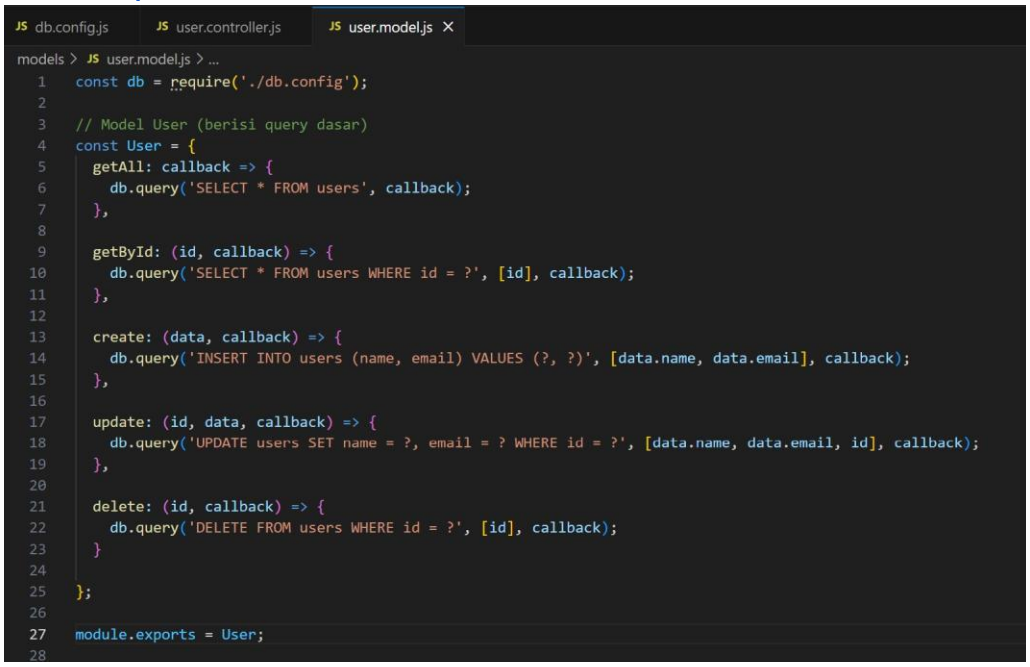
8. Jalankan atau restart ulang node server.js
   (Pastikan mysql sudah running, user password mysql sudah benar)

C. Melakukan Test API

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

D. Lengkapi Controllers dan Model

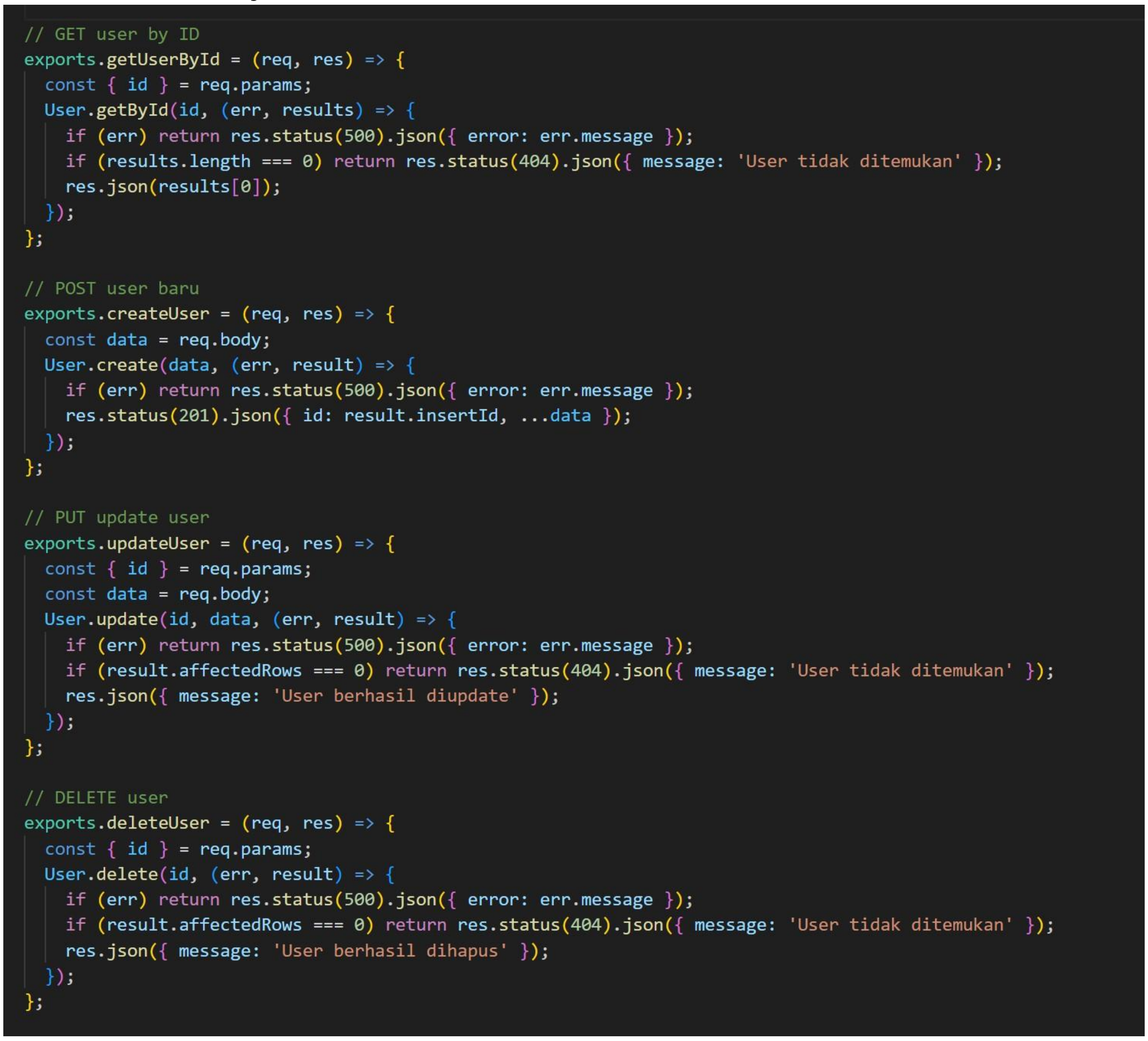
1. Tambahkan class untuk model baru, agar terhubung dengan controller. Ubah pada file user.model.js



```javascript
const db = require('./db.config');

// Model User (berisi query dasar)
const User = {
  getAll: callback => {
    db.query('SELECT * FROM users', callback);
  },

  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);
  }

};

module.exports = User;
```

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

File: user.controller.js



```javascript
// 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"
}
```

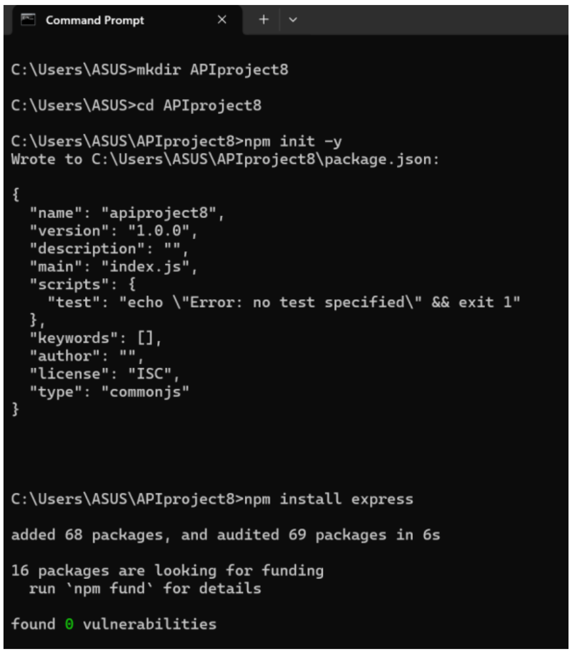
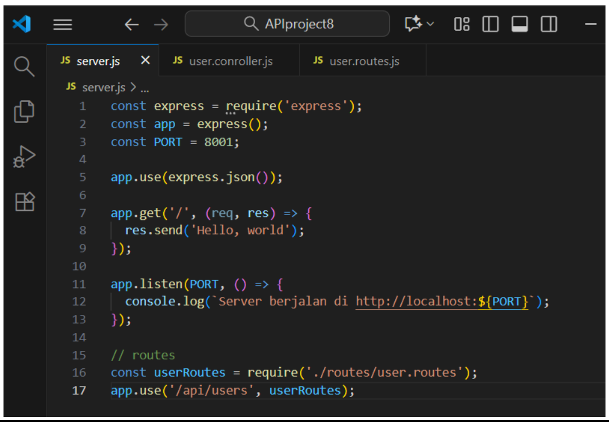
6. Menguji /api/users/3 (Method: DELETE)

F. Github + Visual Code

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

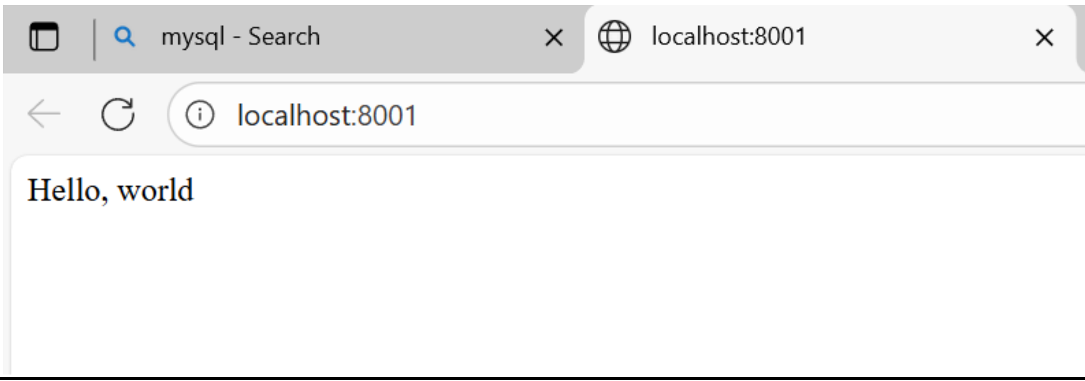
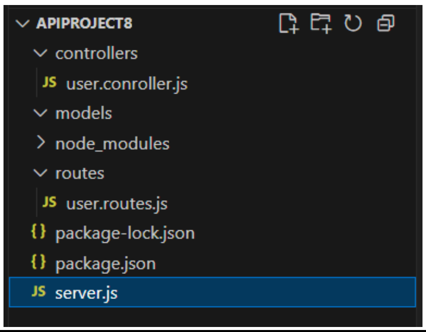
```
 git init
git add
.
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/agunghakase/Latihan8.git
git push -u origin main
```

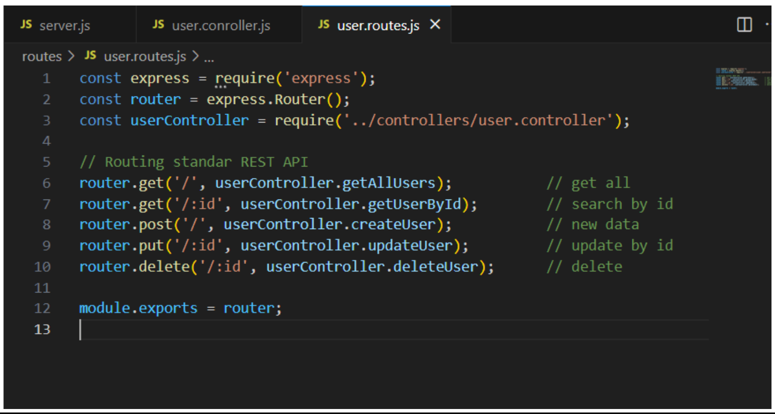
Hasil Pengerjaan

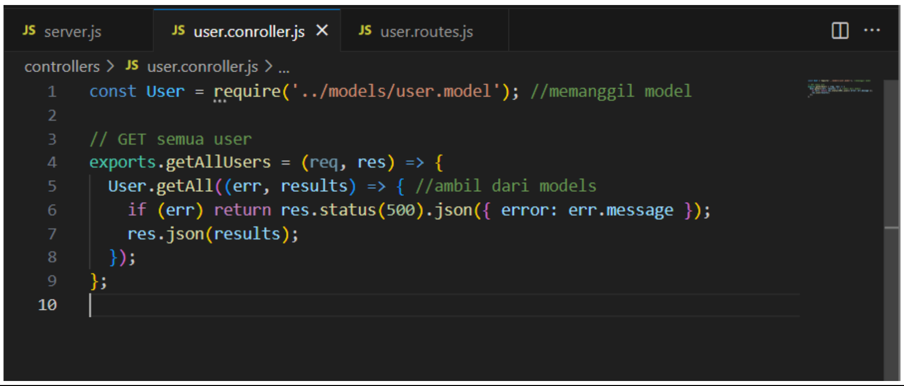
No.	Instruksi	Screenshot
A.	Installasi dan Konfigurasi	
1.	Membuat folder APIproject8 dan menginstal npm init -y , dan npm install express	
2.	Membuat server api	

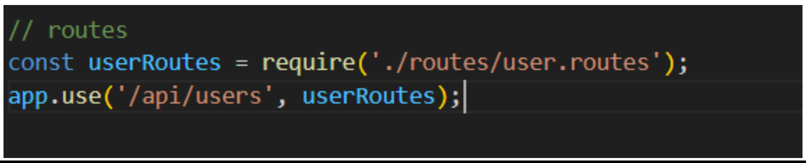
```
C:\Users\ASUS>mkdir APIproject8

C:\Users\ASUS>cd APIproject8

C:\Users\ASUS\APIproject8>npm init -y
Wrote to C:\Users\ASUS\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"
}



C:\Users\ASUS\APIproject8>npm install express

added 68 packages, and audited 69 packages in 6s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

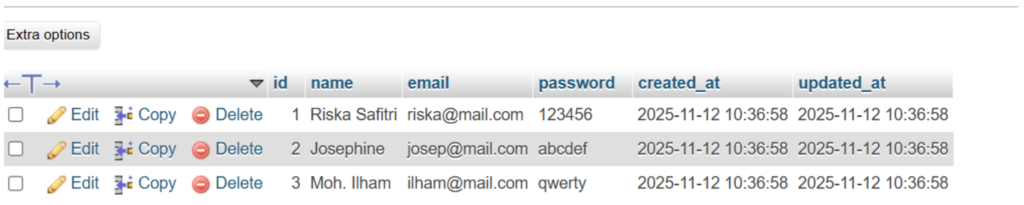
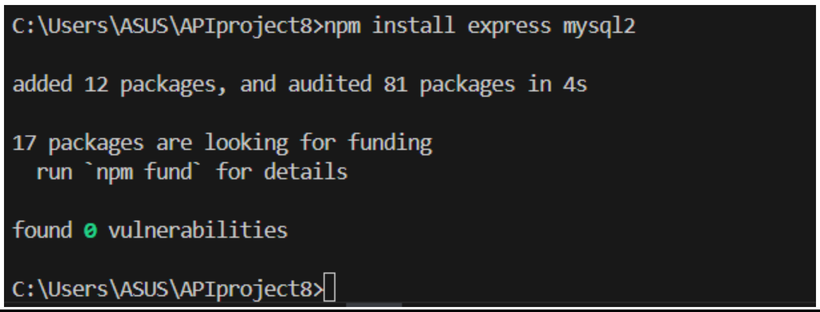
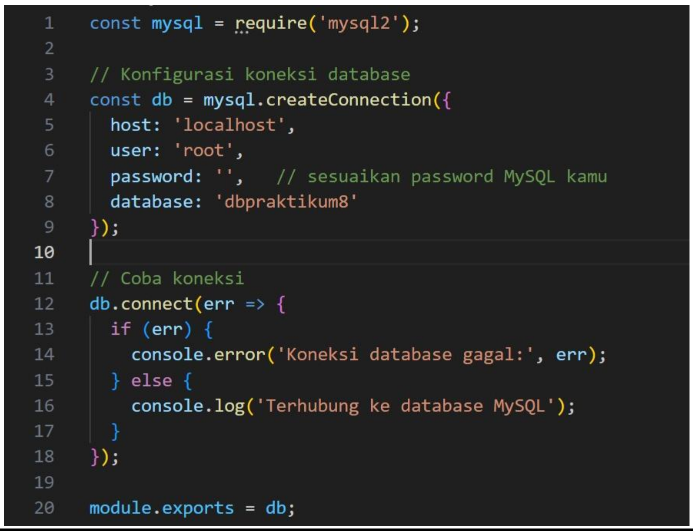
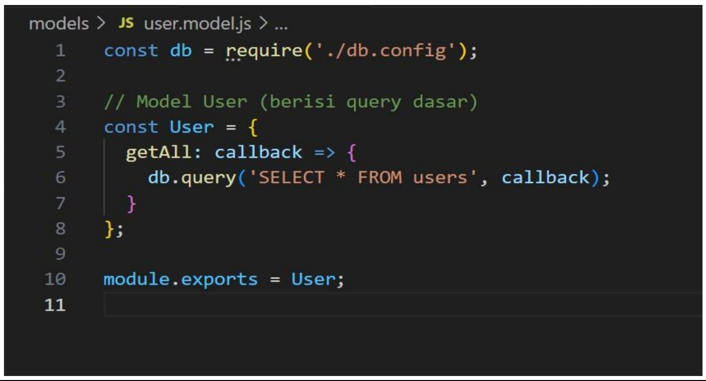
```javascript
const express = require('express');
const app = express();
const PORT = 8001;

app.use(express.json());

app.get('/', (req, res) => {
  res.send('Hello, world');
});

app.listen(PORT, () => {
  console.log(`Server berjalan di http://localhost:${PORT}`);
});

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

		 mysql - Search localhost:8001 localhost:8001 Hello, world
B.	Github dan Viscode	
1.	Membuat folder routes, controllers, dan models	 APIPROJECT8 controllers user.conroller.js models node_modules routes user.routes.js package-lock.json package.json server.js

2.	Menulis kode program di routes	 server.js · user.conroller.js · user.routes.js
		routes > user.routes.js > ...
		`const express = require('express');`
		`const router = express.Router();`
		`const userController = require('../controllers/user.controller');`
		
		`// Routing standar REST API`
		`router.get('/', userController.getAllUsers);          // get all`
		`router.get('/:id', userController.getUserById);       // search by id`
		`router.post('/', userController.createUser);          // new data`
		`router.put('/:id', userController.updateUser);        // update by id`
		`router.delete('/:id', userController.deleteUser);     // delete`
		
		`module.exports = router;`

3.	Menulis kode program di controller	 server.js · user.conroller.js · user.routes.js
		controllers > user.conroller.js > ...
		`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);`
		`  });`
		`};`

4.	Update server.js	 `// routes`
		`const userRoutes = require('./routes/user.routes');`
		`app.use('/api/users', userRoutes);`

c.		
1.	Mengisi database	
2.	Menginstall express mysql2	
3.	Membuat file db.config	
4.	Membuat file user.model.js	
D		

Extra options

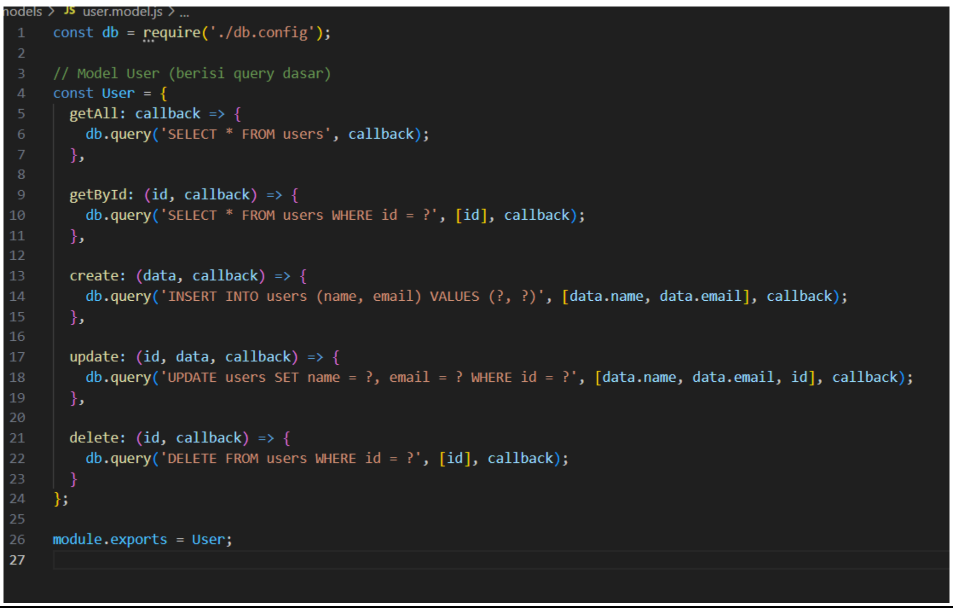
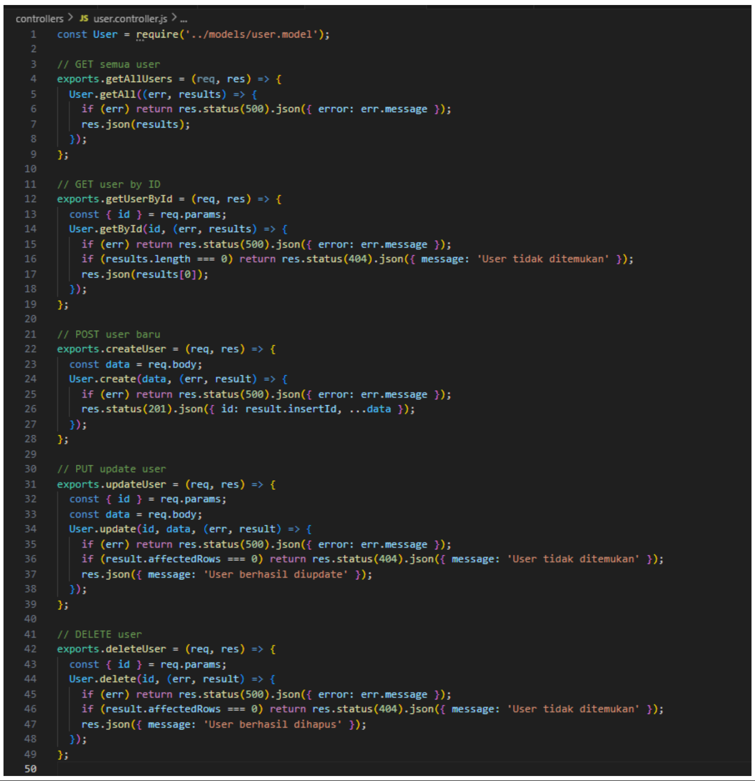
				id	name	email	password	created_at	updated_at
☐	Edit	Copy	Delete	1	Riska Safitri	riska@mail.com	123456	2025-11-12 10:36:58	2025-11-12 10:36:58
☐	Edit	Copy	Delete	2	Josephine	josep@mail.com	abcdef	2025-11-12 10:36:58	2025-11-12 10:36:58
☐	Edit	Copy	Delete	3	Moh. Ilham	ilham@mail.com	qwerty	2025-11-12 10:36:58	2025-11-12 10:36:58

```
C:\Users\ASUS\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

C:\Users\ASUS\APIproject8>
```

```javascript
const mysql = require('mysql2');

// Konfigurasi koneksi database
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',   // sesuaikan password MySQL kamu
  database: 'dbpraktikum8'
});

// Coba koneksi
db.connect(err => {
  if (err) {
    console.error('Koneksi database gagal:', err);
  } else {
    console.log('Terhubung ke database MySQL');
  }
});

module.exports = db;
```

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

```javascript
const db = require('./db.config');

// Model User (berisi query dasar)
const User = {
  getAll: callback => {
    db.query('SELECT * FROM users', callback);
  }
};

module.exports = User;
```

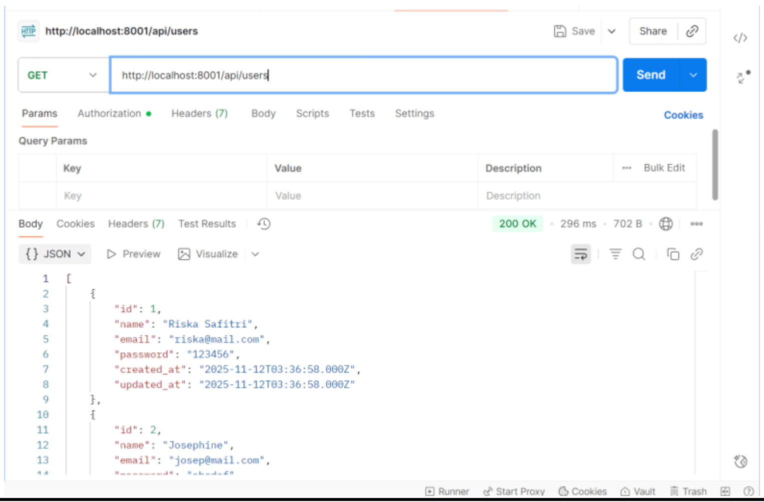
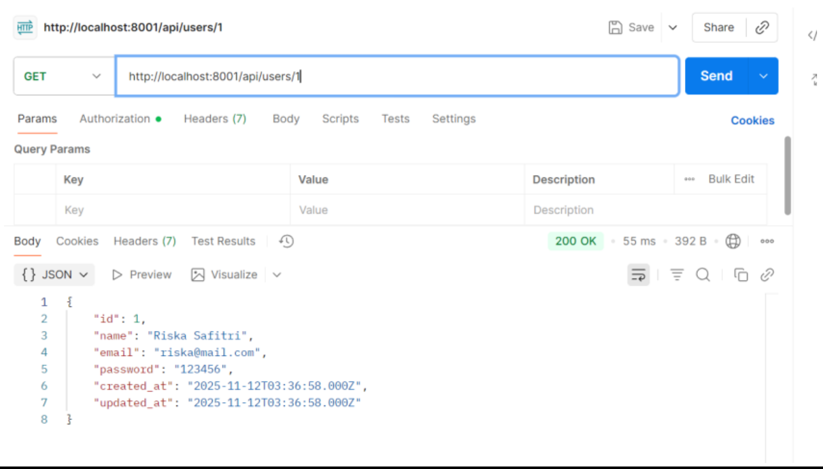
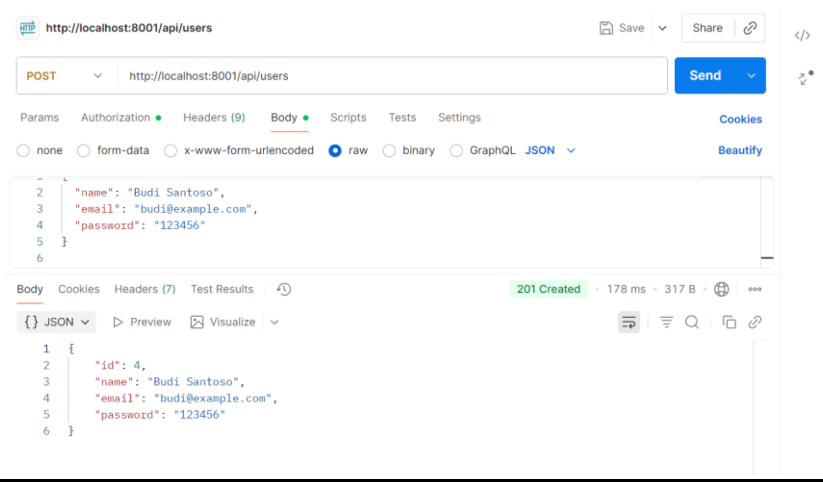
1.	Menambahkan class baru di model	
2.	Menambah class baru untuk routes	

nodels > JS user.model.js > ...

```javascript
const db = require('./db.config');

// Model User (berisi query dasar)
const User = {
  getAll: callback => {
    db.query('SELECT * FROM users', callback);
  },

  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);
  }
};

module.exports = User;
```

controllers > JS user.controller.js > ...

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

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

// 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' });
  });
};
```

3.	Menampilkan semua user	
4.	Menampilkan user dengan id	
5.	Menambah user baru	

http://localhost:8001/api/users

GET http://localhost:8001/api/users

Query Params: Key | Value | Description

Body — 200 OK · 296 ms · 702 B

```
[
    {
        "id": 1,
        "name": "Riska Safitri",
        "email": "riska@mail.com",
        "password": "123456",
        "created_at": "2025-11-12T03:36:58.000Z",
        "updated_at": "2025-11-12T03:36:58.000Z"
    },
    {
        "id": 2,
        "name": "Josephine",
        "email": "josep@mail.com",
```

http://localhost:8001/api/users/1

GET http://localhost:8001/api/users/1

Query Params: Key | Value | Description

Body — 200 OK · 55 ms · 392 B

```
{
    "id": 1,
    "name": "Riska Safitri",
    "email": "riska@mail.com",
    "password": "123456",
    "created_at": "2025-11-12T03:36:58.000Z",
    "updated_at": "2025-11-12T03:36:58.000Z"
}
```

http://localhost:8001/api/users

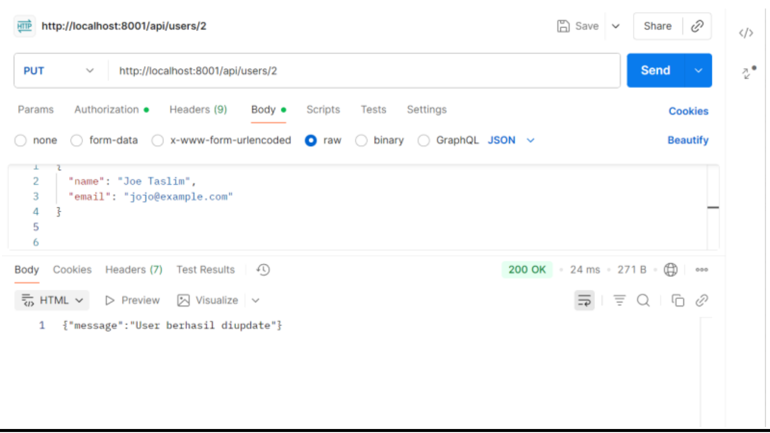
POST http://localhost:8001/api/users

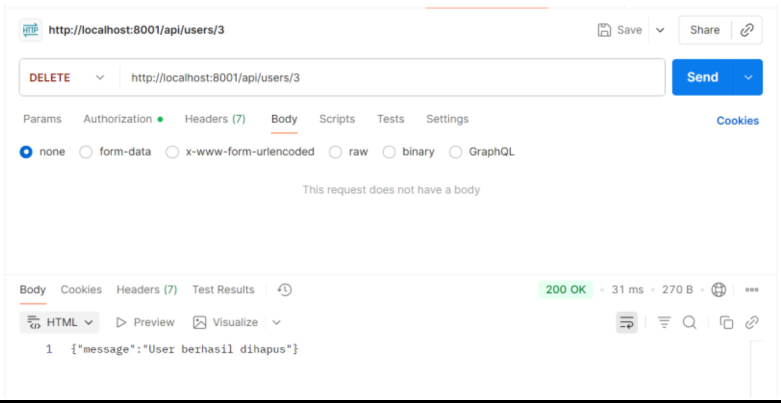
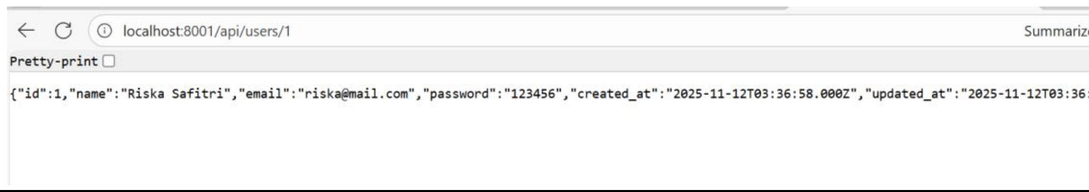
Body: raw, JSON

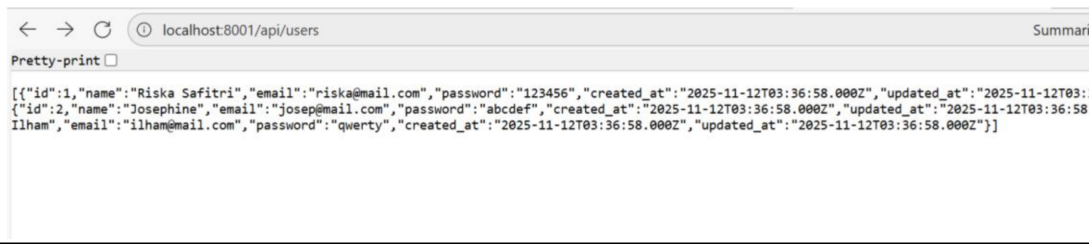
```
    "name": "Budi Santoso",
    "email": "budi@example.com",
    "password": "123456"
}
```

Body — 201 Created · 178 ms · 317 B

```
{
    "id": 4,
    "name": "Budi Santoso",
    "email": "budi@example.com",
    "password": "123456"
}
```

6.	Mengaupdate data user yang sudah ada	 http://localhost:8001/api/users/2
		PUT http://localhost:8001/api/users/2
		Body: raw JSON
		`{`
		`"name": "Joe Taslim",`
		`"email": "jojo@example.com"`
		`}`
		200 OK · 24 ms · 271 B
		`{"message":"User berhasil diupdate"}`

7.	Menghapus data user berdasarkan id	 http://localhost:8001/api/users/3 DELETE http://localhost:8001/api/users/3 Body: none This request does not have a body 200 OK · 31 ms · 270 B `{"message":"User berhasil dihapus"}`
		 localhost:8001/api/users/1 Pretty-print ☐ `{"id":1,"name":"Riska Safitri","email":"riska@mail.com","password":"123456","created_at":"2025-11-12T03:36:58.000Z","updated_at":"2025-11-12T03:36:`

		 localhost:8001/api/users
		Pretty-print ☐
		`[{"id":1,"name":"Riska Safitri","email":"riska@mail.com","password":"123456","created_at":"2025-11-12T03:36:58.000Z","updated_at":"2025-11-12T03:`
		`{"id":2,"name":"Josephine","email":"josep@mail.com","password":"abcdef","created_at":"2025-11-12T03:36:58.000Z","updated_at":"2025-11-12T03:36:58`
		`Ilham","email":"ilham@mail.com","password":"qwerty","created_at":"2025-11-12T03:36:58.000Z","updated_at":"2025-11-12T03:36:58.000Z"}]`

E		
1.	Link github	https://github.com/ftrnaa/latihan-8.git