

# 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
```

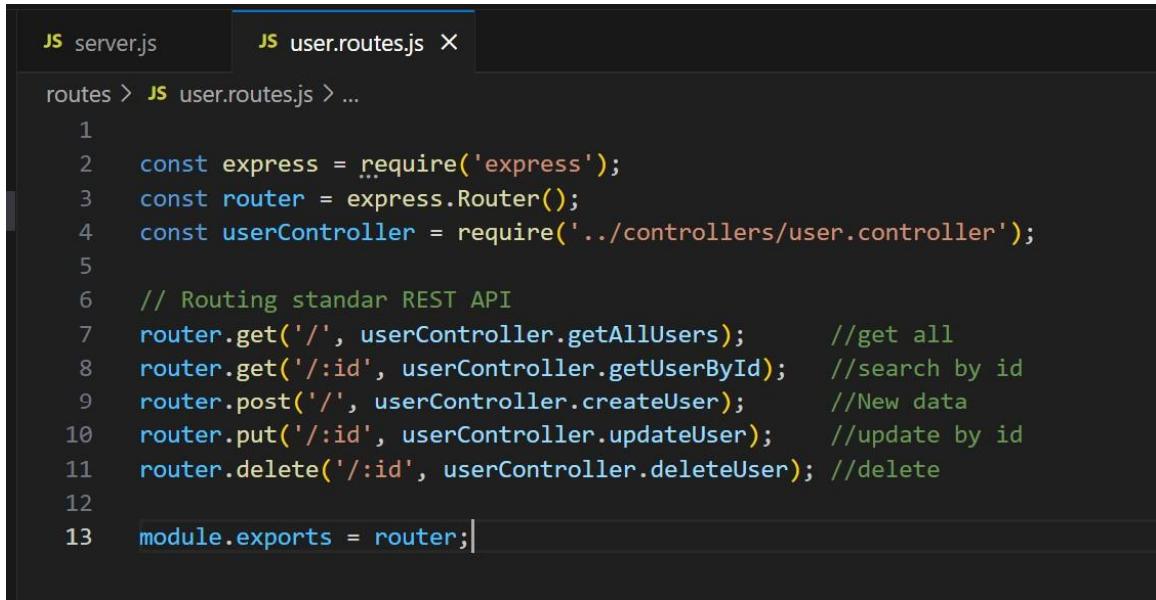
4. Jalankan server.js dengan mengetik  
Ketik: node server.js

### 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



```
users > JS user.controller.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);
    });
};
```

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 two tabs: 'EXPLORER' on the left and 'JS db.js' on the right. The EXPLORER tab shows a project structure with folders 'PRAKTIKUM8', 'controllers', 'models', 'routes', and files 'user.controller.js', 'db.js', 'user.routes.js', 'package.json', and 'server.js'. The 'models' folder is expanded, and 'db.js' is selected. The JS tab displays 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:
  - PRAKTIKUM8
    - controllers
    - JS user.controller.js**
    - models
      - JS db.js**
      - JS user.model.js** (highlighted)
    - routes
    - {} package.json
    - JS server.js
- ...**  button
- JS db.js** tab
- JS user.model.js X** tab
- JS user.controller.js** tab

The `user.model.js` tab 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

```

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` tab with the following updated code:

```

JS db.config.js JS user.controller.js JS user.model.js X
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

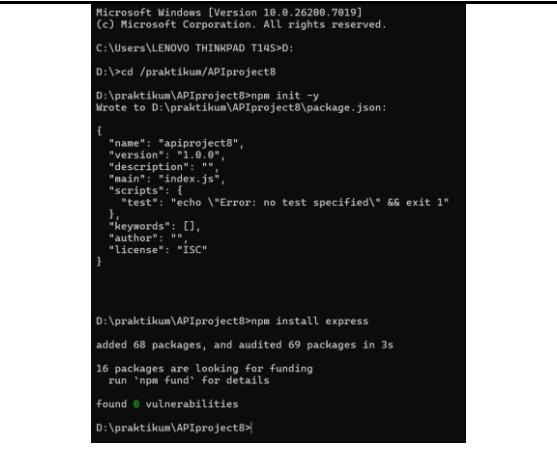
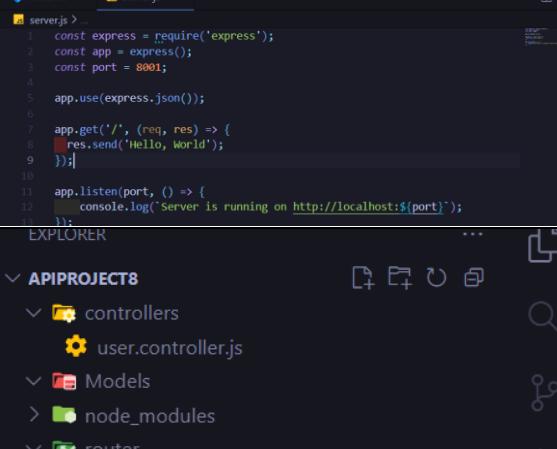
```
{  
  "name": "Joe Taslim",  
  "email": "jojo@example.com"  
}
```
6. Menguji /api/users/3 (Method: DELETE)

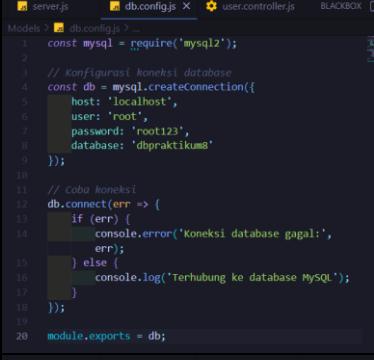
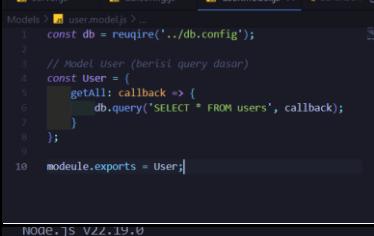
## F. Github + Visual Code

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.	Installasi dan Konfigurasi		
1.	Membuat Server API dengan Express.js	<pre>Microsoft Windows [Version 10.0.26200.7819] (c) Microsoft Corporation. All rights reserved.  C:\Users\LENOVO THINKPAD T14S&gt;D: D:\praktikum\APIproject8&gt;pm init -y Write to D:\praktikum\APIproject8\package.json:  {   "name": "apiproject8",   "version": "1.0.0",   "description": "",   "main": "index.js",   "scripts": {     "test": "echo \\"Warning: no test specified\\&amp;&amp; exit 1"   },   "keywords": [],   "author": "",   "license": "ISC" }  D:\praktikum\APIproject8&gt;npm install express added 68 packages, and audited 69 packages in 3s 16 packages are looking for funding   run `npm fund` for details   found 0 vulnerabilities  D:\praktikum\APIproject8&gt;</pre> 	
2.	Membuat struktur MVC		
3.	Membuat database dengan models	<pre>Enter password: ***** Welcome to the MariaDB monitor. Commands end with ; or \g. Your MariaDB connection id is 12 Server version: 10.6.23-MariaDB mariadb.org binary distribution  Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  Type 'help;' or '\h' for help. Type 'c' to clear the current input statement.  MariaDB [(none)]&gt; CREATE DATABASE IF NOT EXISTS dbpraktikum8; Query OK, 1 row affected (0.002 sec)  MariaDB [(none)]&gt; 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 ); ERROR 1064 (42000): No database selected MariaDB [(none)]&gt; use dbpraktikum8; Database changed MariaDB [dbpraktikum8]&gt; 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 ); ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'CREATE TABLE IF NOT EXISTS users ( id INT AUTO_INCREMENT PRIMARY KEY, na' at line 1 MariaDB [dbpraktikum8]&gt; 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 ); Query OK, 0 rows affected (0.024 sec)  MariaDB [dbpraktikum8]&gt; INSERT INTO users (name, email, password) VALUES ('Riska Safitri', 'riska@mail.com', '123456'), ('Josephine', 'josep@mail.com', 'abcdef'), ('Moh. Ilham', 'ilham@mail.com', 'qwerty'); Query OK, 3 rows affected (0.011 sec) Records: 3  Duplicates: 0  Warnings: 0</pre>	

4.	Install module mysql2	<pre>... powershell + ~ ⌂ ⌂ ...   [] × PS D:\praktikum\APIproject8&gt; npm install express mysql2  added 12 packages, and audited 81 packages in 2s  17 packages are looking for funding   run `npm fund` for details  found 0 vulnerabilities</pre>	
5.	Membuat db.config.js	 <pre>server.js db.config.js user.controller.js BLACKBOX ... Models &gt; db.config.js ... 1 const mysql = require('mysql2'); 2 3 // Konfigurasi koneksi database 4 const db = mysql.createConnection({ 5   host: 'localhost', 6   user: 'root', 7   password: 'root123', 8   database: 'dbpraktikum' 9 }); 10 11 // Coba koneksi 12 db.connect(err =&gt; { 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;</pre>	
6.	Membuat user.model.js	 <pre>server.js db.config.js user.model.js BLACKBOX ... Models &gt; user.model.js ... 1 const db = require('../db.config'); 2 3 // Model User (berisi query dasar) 4 const User = { 5   getAll: callback =&gt; { 6     db.query('SELECT * FROM users', callback); 7   } 8 }; 9 10 module.exports = User;</pre>	
7.	Jalankan server.js	<pre>Node.js v22.19.0 PS D:\praktikum\APIproject8&gt; node server.js D:\praktikum\APIproject8\node_modules\router\lib\route.js:228     throw new TypeError('Argument handler must be a function')     ^ TypeError: Argument handler must be a function     at Route.&lt;computed&gt; [as get] (D:\praktikum\APIproject8\node_modules\router\lib\route.js:228:15)     at Router.&lt;computed&gt; [as get] (D:\praktikum\APIproject8\node_modules\express\index.js:418:10)</pre>	
8.	Melengkapi Controllers dan Model	 <pre>// GET semua user exports.getAllUsers = (req, res) =&gt; {   User.getAll((err, results) =&gt; { //ambil dari models     if (err) return res.status(500).json({ error: err.message });     res.json(results);   }); };  // GET user by ID exports.getUserById = (req, res) =&gt; {   const data = req.body;   User.create(data, (err, results) =&gt; {     if (err) return res.status(500).json({ error: err.message });     res.status(201).json({ id: results.insertId, ...data });   }); };  // POST user baru exports.createUser = (req, res) =&gt; {   const data = req.body;   User.create(data, (err, results) =&gt; {     if (err) return res.status(500).json({ error: err.message });   }); };</pre>	

	<pre> const db = require('./db.config');  const User = {   getAll: callback =&gt; {     db.query('SELECT * FROM users', callback);   }    getById: (id, callback) =&gt; {     db.query('SELECT * FROM users WHERE id = ?', [id], callback);   }    create: (data, callback) =&gt; {     db.query('INSERT INTO users (name, email) VALUES (?, ?)', [data.name, data.email], callback);   }    update: (id, data, callback) =&gt; {     db.query('UPDATE users SET name = ?, email = ? WHERE id = ?', [data.name, data.email, id], callback);   }    delete: (id, callback) =&gt; {     db.query('DELETE FROM users WHERE id = ?', [id], callback);   } } </pre>
9.	Menguji endpoint /
10.	Menguji endpoint /api/users (Method: GET)

11.	Menguji endpoint /api/users/1 (Method: GET)	<pre> 1 { 2   "id": 1, 3   "name": "Riska Safitri", 4   "email": "riska@gmail.com", 5   "password": "123456", 6   "created_at": "2025-11-12T03:13:20.000Z", 7   "updated_at": "2025-11-12T03:13:20.000Z" 8 } </pre>	
12.	Menguji endpoint /api/users (Method: POST)	<pre> 1 { 2   "id": 4, 3   "name": "Budi Santoso", 4   "email": "budi@example.com" 5 } </pre>	
13.	Menguji /api/users/2 (Method: PUT)	<pre> 1 { 2   "name": "Joe Taslim", 3   "email": "jojo@example.com" 4 } </pre> <p>Body</p> <pre> 1 { 2   "message": "User berhasil diupdate" 3 } </pre>	
14.	Menguji /api/users/3 (Method: DELETE)	<pre> 1 Ctrl+Alt+P to Ask AI </pre> <p>Body</p> <pre> 1 { 2   "message": "User berhasil dihapus" 3 } </pre>	
B.	Github dan Viscode		

1.

Upload di github

