

# 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
```

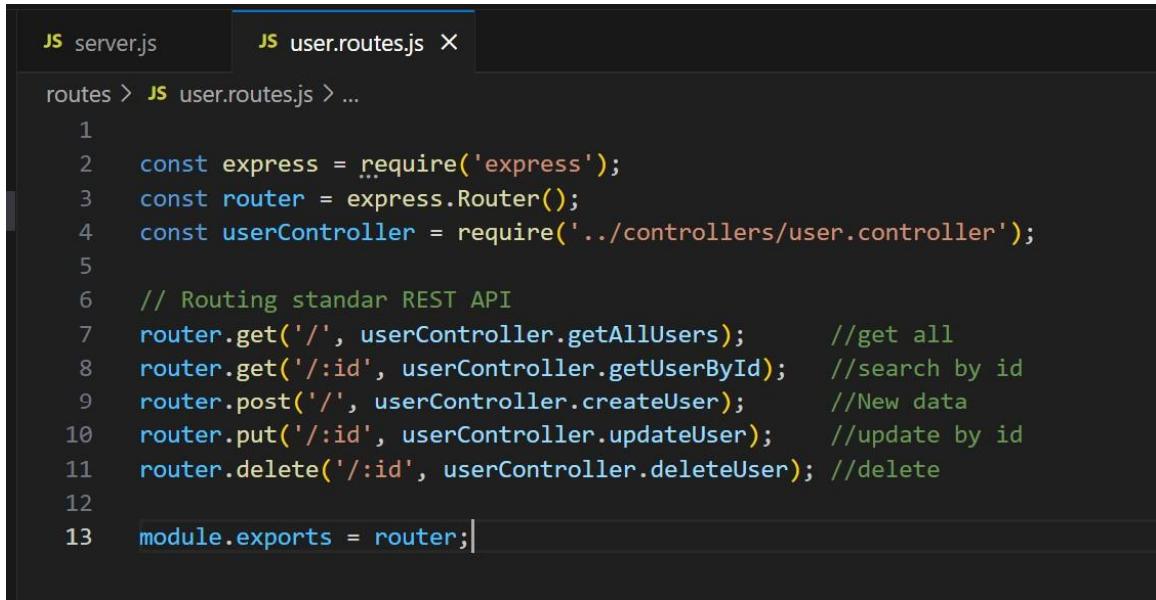
4. Jalankan server.js dengan mengetik  
Ketik: node server.js

### 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' and 'JS db.js'. 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 'JS db.js' tab is active, displaying 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

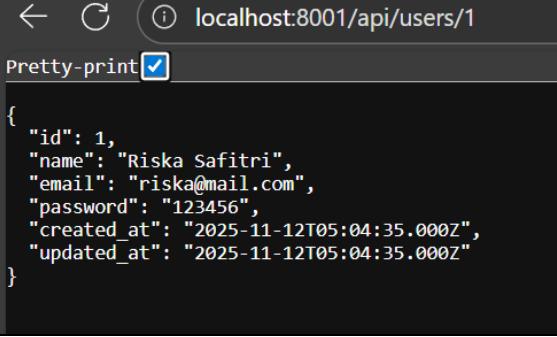
```
{  
  "name": "Joe Taslim",  
  "email": "jojo@example.com"  
}
```
6. Menguji /api/users/3 (Method: DELETE)

## 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 Penggeraan

No.	Instruksi	Screenshot	Kendala/Saran
A.	Installasi dan Konfigurasi		
1.	All users	 <pre> &lt; C ⓘ localhost:8001/api/users Pretty-print [✓]  [   {     "id": 1,     "name": "Riska Safitri",     "email": "riska@mail.com",     "password": "123456",     "created_at": "2025-11-12T05:04:35.000Z",     "updated_at": "2025-11-12T05:04:35.000Z"   },   {     "id": 2,     "name": "Josephine",     "email": "josep@mail.com",     "password": "abcdef",     "created_at": "2025-11-12T05:04:35.000Z",     "updated_at": "2025-11-12T05:04:35.000Z"   },   {     "id": 3,     "name": "Moh. Ilham",     "email": "ilham@mail.com",     "password": "qwerty",     "created_at": "2025-11-12T05:04:35.000Z",     "updated_at": "2025-11-12T05:04:35.000Z"   } ]</pre>	
2.	UserID	 <pre> &lt; C ⓘ localhost:8001/api/users/1 Pretty-print [✓]  {   "id": 1,   "name": "Riska Safitri",   "email": "riska@mail.com",   "password": "123456",   "created_at": "2025-11-12T05:04:35.000Z",   "updated_at": "2025-11-12T05:04:35.000Z" }</pre>	
3	Method POST		

4	PUT	
5	DELETE	

		<pre> HTTP My Collection / Post data DELETE http://localhost:8001/api/users/3 Params Authorization Headers (6) Body Scripts Settin none form-data x-www-form-urlencoded raw t 1 Ctrl+Alt+P to Ask AI  Body Cookies Headers (7) Test Results (1/1) [] JSON Preview Visualize 1 { 2   "message": "User berhasil dihapus" 3 } </pre>	
B.	Github dan Viscode		
1.	GITHUB	<pre> PROBLEMS OUTPUT TERMINAL ... powershell + v ⌂ ⌂ ... PS D:\expressjs\APIproject8&gt; git remote add origin https://github.com/Fahmi31/praktikum8.git PS D:\expressjs\APIproject8&gt; git push -u origin main Enumerating objects: 963, done. Counting objects: 100% (963/963), done. Delta compression using up to 4 threads Compressing objects: 100% (893/893), done. Writing objects: 100% (963/963), 1.04 MiB   965.00 KiB/s, done. Total 963 (delta 162), reused 0 (delta 0), pack-reused 0 remote: Resolving deltas: 100% (162/162), done. To https://github.com/Fahmi31/praktikum8.git  * [new branch]      main -&gt; main branch 'main' set up to track 'origin/main'. PS D:\expressjs\APIproject8&gt; </pre>	
2.			
...			