

Praktikum 9 - 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. Menyediakan API Enpoints

1. Lanjutkan Project Praktikum 8, dengan menggunakan file yang sama (copy)
2. Tambahkan pada database, sebuah tabel produk Isi tabel produk seperti pada tabel berikut:
(Buat 10 item)

id	nama	deskripsi	harga	foto
1.	Indomie Goreng		2500	images/miegor eng.jpg
2.
10.

3. Seperti pada perintah praktikum 8 buatkan beberapa endpoints
GET : localhost:8001/api/products/
GET : localhost:8001/api/products/:id
POST : localhost:8001/api/products/
PUT : localhost:8001/api/products/:id
DELETE: localhost:8001/api/products/:id
4. Pastikan memiliki file dengan struktur sebagai berikut.
controllers/[products.controller.js](#)
routes/[products.routes.js](#)
models/[products.model.js](#)
5. Test API dengan menggunakan **POSTMAN**

B. Menambahkan Proteksi API (Simple - Bearer Method)

1. Buat sebuah folder bernama middlewares, kemudian buat didalamnya sebuah file dengan nama [auth.middleware.js](#) dengan kode program seperti dibawah ini

```
export const authBearer = (req, res, next) => {
    const authHeader = req.headers.authorization;

    // Tidak ada Authorization
    if (!authHeader) {
        return res.status(401).json({ message: "No authorization header" });
    }

    // Harus Bearer
    if (!authHeader.startsWith("Bearer ")) {
        return res.status(401).json({ message: "Bearer token required" });
    }

    // Ambil token
    const token = authHeader.split(" ")[1];

    // Token yang benar (misal hardcode)
    const VALID_TOKEN = "12345TOKENRAHASIA";

    if (token !== VALID_TOKEN) {
        return res.status(403).json({ message: "Invalid token" });
    }

    next();
};
```

2. Pada file [product.routes.js](#) panggil [auth.middleware.js](#) *import { authBearer } from "../middleware/auth.middleware.js"*
3. Lalu tambahkan authBearer ke endpoints yang perlu di proteksi
router.post("/", authBearer, createProduct); router.put("/:id", authBearer, updateProduct); router.delete("/:id", authBearer, deleteProduct);
4. Gunakan POSTMAN untuk akses 3 endpoints ini dengan menambahkan bearer
12345TOKENRAHASIA

F. Github + Visual Code

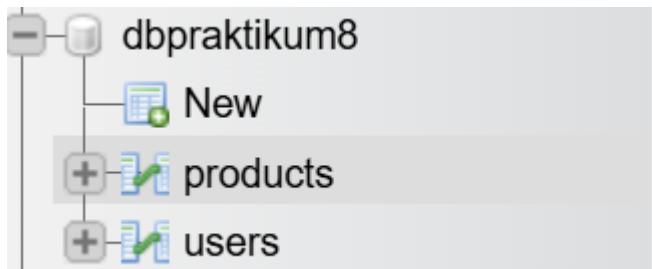
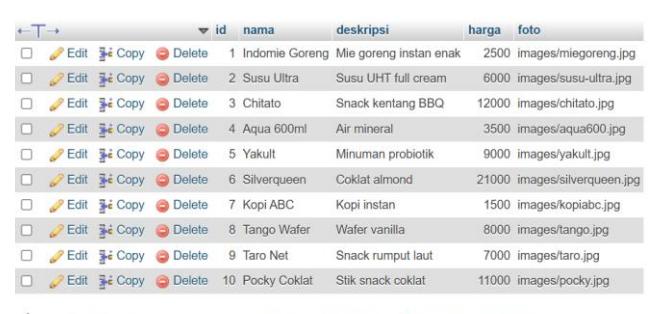
- Buat proyek di Github dengan nama **Latihan9**

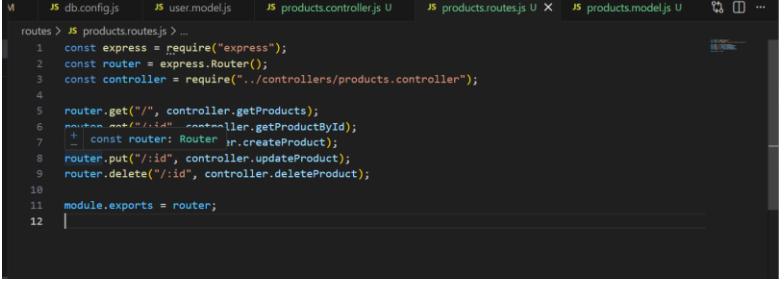
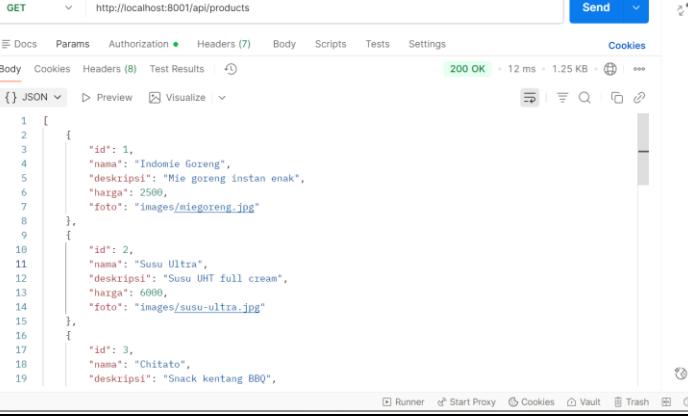
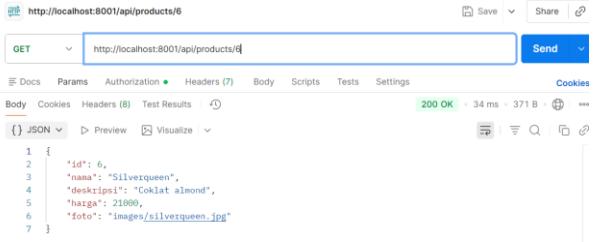
```
git init
```

```
git add
```

```
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/agunghakase/Latihan9.git
git push -u origin main
```

Hasil Pengerjaan

No.	Instruksi	Screenshot	Keterangan
A.	Installasi dan Konfigurasi		
1.	Membuat table baru products		
2.	Mengisi table products		

3.	<p>Membuat endpoint</p>	 <pre> routes > JS products.routes.js > ... 1 const express = require("express"); 2 const router = express.Router(); 3 const controller = require("../controllers/products.controller"); 4 5 router.get("/", controller.getProducts); 6 router.get("/:id", controller.getProductById); 7 router.post("/", controller.createProduct); 8 router.put("/:id", controller.updateProduct); 9 router.delete("/:id", controller.deleteProduct); 10 11 module.exports = router; 12 </pre>
4.	<p>Struktur file</p>	 <pre> APIPROJECT8 controllers JS products.controller.js JS user.controller.js middlewares models JS db.config.js JS products.model.js JS user.model.js routes JS products.routes.js JS user.routes.js {} package-lock.json {} package.json JS server.js </pre>
5.	<p>Test API localhost:8001/api/products/</p>	 <pre> GET http://localhost:8001/api/products 200 OK - 12 ms - 1.25 KB [{"id": 1, "nama": "Indomie Goreng", "deskripsi": "Mie goreng instan enak", "harga": 2500, "foto": "images/miegoreng.jpg"}, {"id": 2, "nama": "Susu Ultra", "deskripsi": "Susu UHT full cream", "harga": 6000, "foto": "images/susu-ultra.jpg"}, {"id": 3, "nama": "Chitato", "deskripsi": "Snack kentang BBQ", "foto": "images/chitato.jpg"}] </pre>
6.	<p>Test API localhost:8001/api/products/</p>	 <pre> GET http://localhost:8001/api/products/6 200 OK - 34 ms - 371 B [{"id": 6, "nama": "Silverqueen", "deskripsi": "Coklat almond", "harga": 21000, "foto": "images/silverqueen.jpg"}] </pre>

7.

POST :
localhost:8001/api/products/

The screenshot shows a Postman interface with a POST request to `http://localhost:8001/api/products`. The request body contains a JSON object with fields: `nama: "yupi"`, `deskripsi: "yupi permen paling enak"`, `harga: 3000`, and `foto: "images/yupi.jpg"`. The response status is `200 OK` with a response time of 184 ms and a size of 414 B. The response body is a JSON object indicating the product was created successfully with an ID of 11.

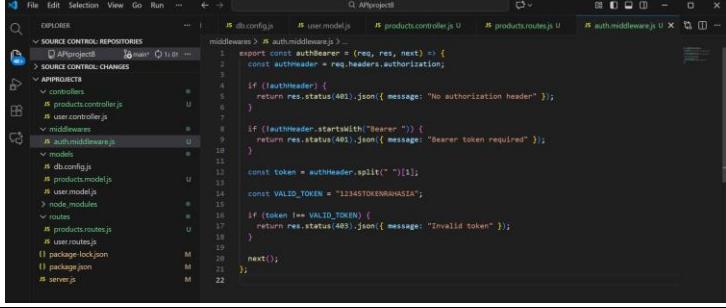
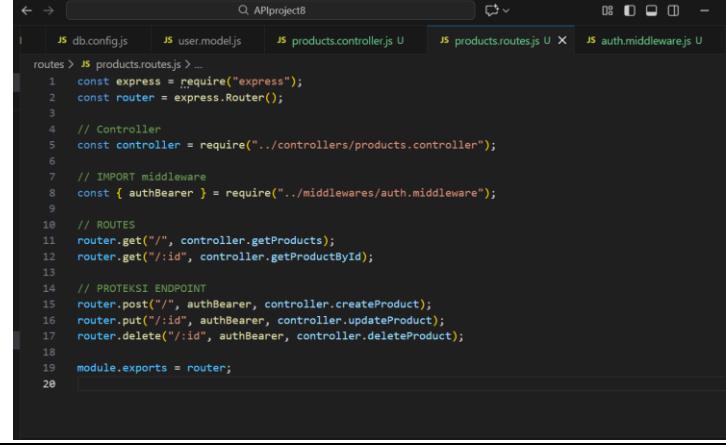
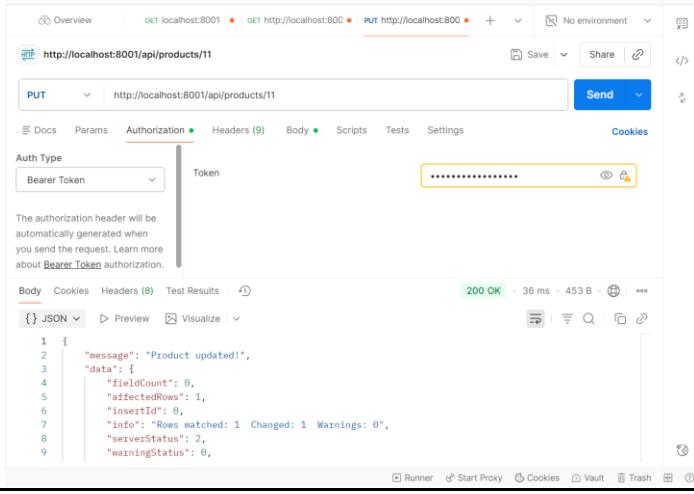
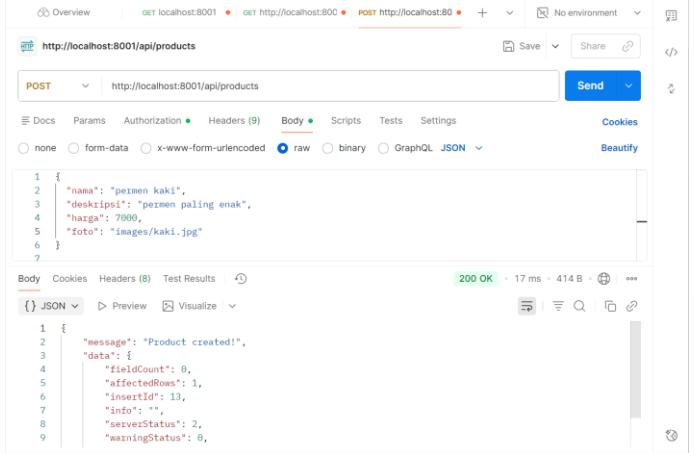
PUT :
localhost:8001/api/products/:id

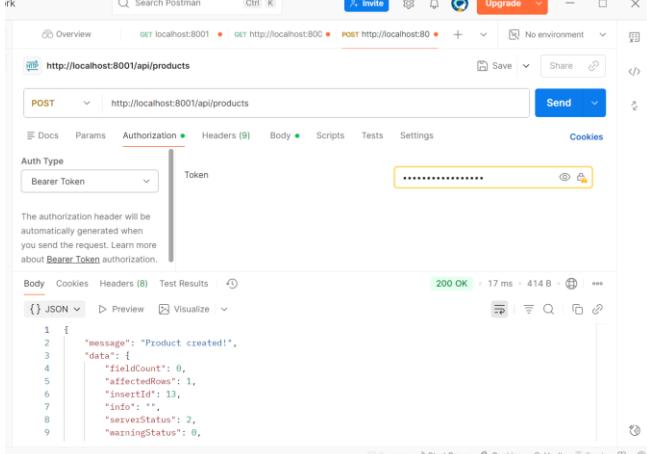
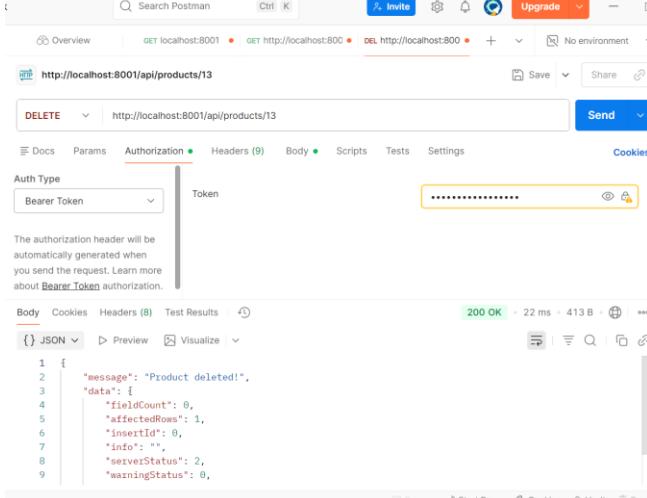
The screenshot shows a Postman interface with a PUT request to `http://localhost:8001/api/products/1`. The request body is identical to the POST request. The response status is `200 OK` with a response time of 112 ms and a size of 453 B. The response body indicates the product was updated successfully with an ID of 1.

DELETE:
localhost:8001/api/products/:id

The screenshot shows a Postman interface with a DELETE request to `http://localhost:8001/api/products/5`. The response status is `200 OK` with a response time of 47 ms and a size of 413 B. The response body indicates the product was deleted successfully with an ID of 5.

B. Github dan Viscode

1.	<p>Membuat folder middleware dan file auth.middleware.js</p>	 <pre> const authHeader = req.headers.authorization; if (!authHeader) { return res.status(401).json({ message: "No authorization header" }); } if (!authHeader.startsWith("Bearer ")) { return res.status(401).json({ message: "Bearer token required" }); } const token = authHeader.split(" ")[1]; const VALID_TOKEN = "12345TOKENKRAHOSIA"; if (token === VALID_TOKEN) { next(); } else { return res.status(403).json({ message: "Invalid token" }); } </pre>
2.	<p>Memambahkan autbearer di products.routes.js</p>	 <pre> const express = require("express"); const router = express.Router(); // Controller const controller = require("../controllers/products.controller"); // IMPORT middleware const { authBearer } = require("../middlewares/auth.middleware"); // ROUTES router.get("/", controller.getProducts); router.get("/:id", controller.getProductById); router.post("/", authBearer, controller.createProduct); router.put("/:id", authBearer, controller.updateProduct); router.delete("/:id", authBearer, controller.deleteProduct); module.exports = router; </pre>
3.	<p>Mengakses put setelah di bearer</p>	 <p>PUT http://localhost:8001/api/products/11</p> <p>Auth Type: Bearer Token</p> <p>Token: <input type="text"/></p> <p>Body (JSON)</p> <pre> { "message": "Product updated!", "data": { "fieldCount": 0, "affectedRows": 1, "insertId": 0, "info": "Rows matched: 1 Changed: 1 Warnings: 0", "serverStatus": 2, "warningStatus": 0 } } </pre>
4.	<p>Mengakses post setelah di bearer</p>	 <p>POST http://localhost:8001/api/products</p> <p>Body (raw)</p> <pre> { "nama": "permen kaki", "deskripsi": "permen paling enak", "harga": 7000, "foto": "images/kaki.jpg" } </pre> <p>Body (JSON)</p> <pre> { "message": "Product created!", "data": { "fieldCount": 0, "affectedRows": 1, "insertId": 13, "info": "", "serverStatus": 2, "warningStatus": 0 } } </pre>

5.	<p>Mengakses post setelah di bearer</p>	 <pre> 1 { 2 "message": "Product created!", 3 "data": { 4 "fieldCount": 0, 5 "affectedRows": 1, 6 "insertId": 13, 7 "info": "", 8 "serverStatus": 2, 9 "warningStatus": 0 } </pre>
6.	<p>Mengakses delete setelah di bearer</p>	 <pre> 1 { 2 "message": "Product deleted!", 3 "data": { 4 "fieldCount": 0, 5 "affectedRows": 1, 6 "insertId": 0, 7 "info": "", 8 "serverStatus": 2, 9 "warningStatus": 0 } </pre>
E.	GITHUB	
1.	Link github	https://github.com/ftrnaa/latihan-8.git