

## 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/miegoreng.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

1. Buat proyek di Github dengan nama **Latihan9**

git init

git add .

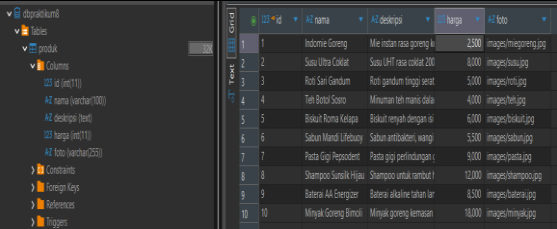
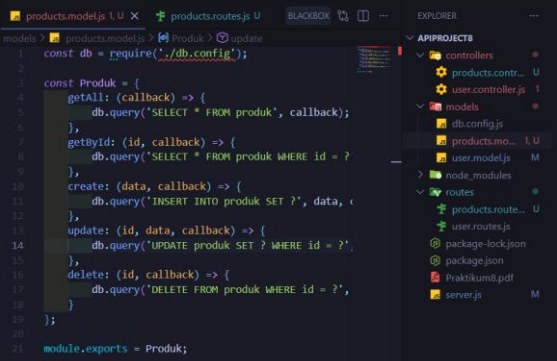
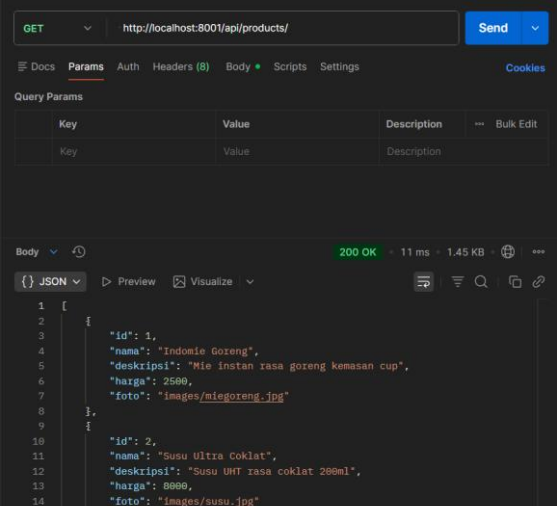
git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/agunghakase/Latihan9.git

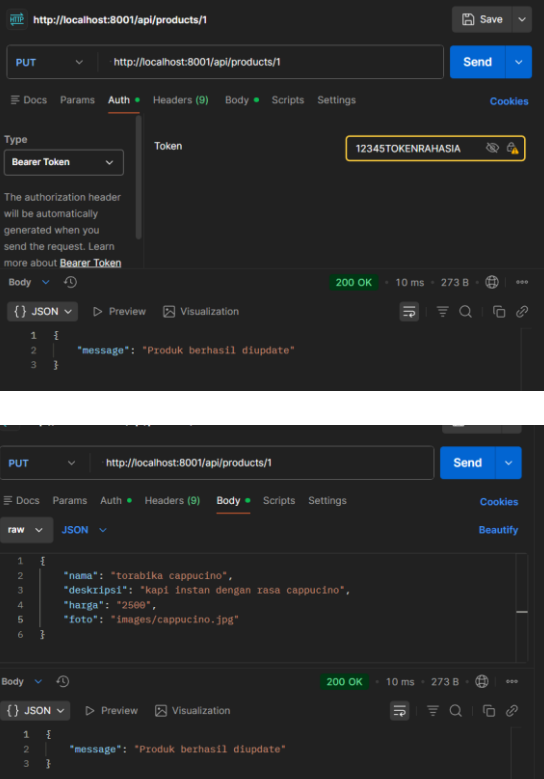
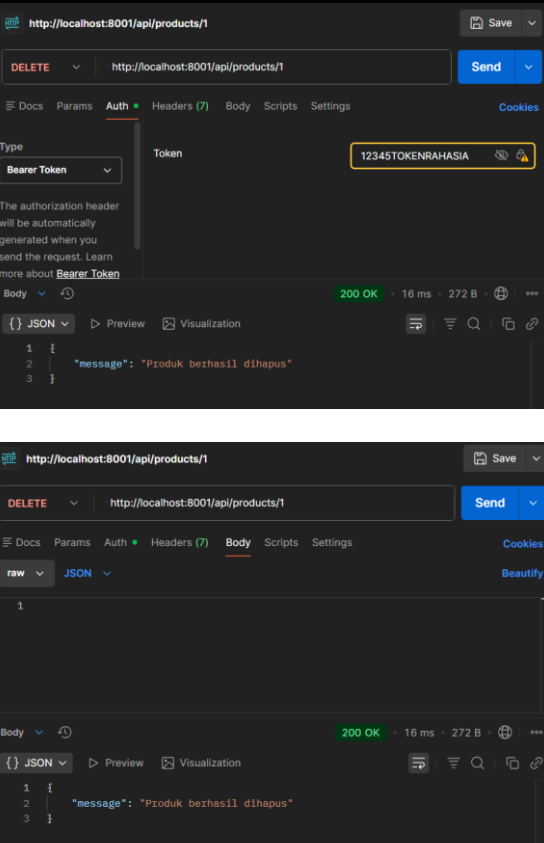
git push -u origin main

## Hasil Pengerjaan

No.	Instruksi	Screenshot	Kendala/Saran
A.	Instalasi dan Konfigurasi		
1.	Membuat tabel produk beserta isi data		-
2.	Buat struktur controller, routes, dan model produk		-
3.	Tes postman melihat semua produk		-

4.	Tes postman melihat produk berdasarkan id	 <p>GET <a href="http://localhost:8001/api/products/3">http://localhost:8001/api/products/3</a> Send</p> <p>Query Params</p> <table border="1"> <thead> <tr> <th>Key</th> <th>Value</th> <th>Description</th> <th>Bulk Edit</th> </tr> </thead> <tbody> <tr> <td>Key</td> <td>Value</td> <td>Description</td> <td></td> </tr> </tbody> </table> <p>Body</p> <pre> 1 { 2   "id": 3, 3   "nama": "Roti Sari Gandum", 4   "deskripsi": "Roti gandum tinggi serat, 2 potong", 5   "harga": 5000, 6   "foto": "images/roti.jpg" 7 }</pre> <p>200 OK · 17 ms · 357 B</p>	Key	Value	Description	Bulk Edit	Key	Value	Description		-
Key	Value	Description	Bulk Edit								
Key	Value	Description									
5.	Tes postman menambah produk baru	 <p>POST <a href="http://localhost:8001/api/products/">http://localhost:8001/api/products/</a> Send</p> <p>Body</p> <pre> 1 { 2   "nama": "Kopi Kapal Api", 3   "deskripsi": "Kopi instan rasa original", 4   "harga": "2000", 5   "foto": "images/kopi.jpg" 6 }</pre> <p>201 Created · 23 ms · 354 B</p>	-								
6.	Tes postman mengedit produk berdasarkan id	 <p>PUT <a href="http://localhost:8001/api/products/2">http://localhost:8001/api/products/2</a> Send</p> <p>Body</p> <pre> 1 { 2   "nama": "Roti Top Baker Bulat", 3   "deskripsi": "Roti enak rasa coklat", 4   "harga": "3500", 5   "foto": "images/roticoklat.jpg" 6 }</pre> <p>200 OK · 19 ms · 273 B</p> <pre> 1 { 2   "message": "Produk berhasil diupdate" 3 }</pre>	-								
7.	Tes postman menghapus produk berdasarkan id	 <p>DELETE <a href="http://localhost:8001/api/products/4">http://localhost:8001/api/products/4</a> Send</p> <p>Body</p> <pre> 1 Ctrl+Alt+P to Ask AI</pre> <p>200 OK · 20 ms · 272 B</p> <pre> 1 { 2   "message": "Produk berhasil dihapus" 3 }</pre>	-								

8.	Membuat auth.middleware.js	<pre> 1 export const authBearer = (req, res, next) =&gt; { 2   const authHeader = req.headers.authorization; 3 4   // Tidak ada Authorization 5   if (!authHeader) { 6     return res.status(401).json({ message: "No 7       authorization header" }); 8   } 9 10  // Harus Bearer 11  if (authHeader.startsWith("Bearer ")) { 12    return res.status(401).json({ message: 13      "Bearer token required" }); 14  } 15 16  // Ambil token 17  const token = authHeader.split(" ")[1]; 18 19  // Token yang benar (misal hardcoded) 20  const VALID_TOKEN = "12345TOKENRAHASIA"; 21 22  if (token !== VALID_TOKEN) { 23    return res.status(403).json({ message: 24      "Invalid token" }); 25  } 26 27  next(); 28 } </pre>	
9.	Memanggil auth.middleware.js dari product.routes.js dan menambahkan authBearer ke endpoint yang perlu di Proteksi	<pre> 1 const express = require('express'); 2 const router = express.Router(); 3 const productController = require('../controllers/ 4   product.controller'); 5 const { authBearer } = require('../middlewares/ 6   auth.middleware'); 7 8 router.get('/', productController.getAllProducts); 9 router.get('/:id', productController. 10  getProductById); 11 router.post('/', productController.createProduct); 12 router.put('/:id', productController. 13  updateProduct); 14 router.delete('/:id', productController. 15  deleteProduct); 16 17 // authBearer 18 router.post('/', authBearer, productController. 19  createProduct); 20 router.put('/:id', authBearer, productController. 21  updateProduct); 22 router.delete('/:id', authBearer, 23  productController.deleteProduct); 24 25 module.exports = router; </pre>	
10.	Tes postman menambah produk baru dengan menambahkan bearer	<p>Top Screenshot (Headers):</p> <ul style="list-style-type: none"> <li>URL: http://localhost:8001/api/products/</li> <li>Method: POST</li> <li>Auth: Bearer Token</li> <li>Token: 12345TOKENRAHASIA</li> <li>Status: 201 Created</li> <li>Body (JSON):       <pre> {   "id": 16,   "nama": "susu milku",   "deskripsi": "susu UHT rasa original",   "harga": "3500",   "foto": "images/susumilku.jpg" } </pre> </li> </ul> <p>Bottom Screenshot (Body):</p> <ul style="list-style-type: none"> <li>URL: http://localhost:8001/api/products/</li> <li>Method: POST</li> <li>Body (JSON):       <pre> {   "nama": "susu milku",   "deskripsi": "susu UHT rasa original",   "harga": "3500",   "foto": "images/susumilku.jpg" } </pre> </li> </ul>	

11.	Tes postman mengedit produk berdasarkan id dengan menambahkan bearer	 <p>The screenshot shows a Postman interface for a PUT request to <code>http://localhost:8001/api/products/1</code>. The 'Auth' tab is selected, showing a 'Bearer Token' type with the token <code>12345TOKENRAHASIA</code>. The 'Body' tab is also selected, showing a JSON body: <code>{ "nama": "tozabika cappucino", "deskripsi": "kopi instan dengan rasa cappucino", "harga": "2500", "foto": "images/cappucino.jpg" }</code>. The response is <code>200 OK</code> with a message: <code>"message": "Produk berhasil diupdate"</code>.</p>	
12.	Tes postman menghapus produk berdasarkan id dengan menambahkan bearer	 <p>The screenshot shows a Postman interface for a DELETE request to <code>http://localhost:8001/api/products/1</code>. The 'Auth' tab is selected, showing a 'Bearer Token' type with the token <code>12345TOKENRAHASIA</code>. The 'Body' tab is also selected, showing a JSON body: <code>{ "message": "Produk berhasil dihapus" }</code>. The response is <code>200 OK</code> with a message: <code>"message": "Produk berhasil dihapus"</code>.</p>	
B.	Github dan Viscode		

1. Upload di github

