

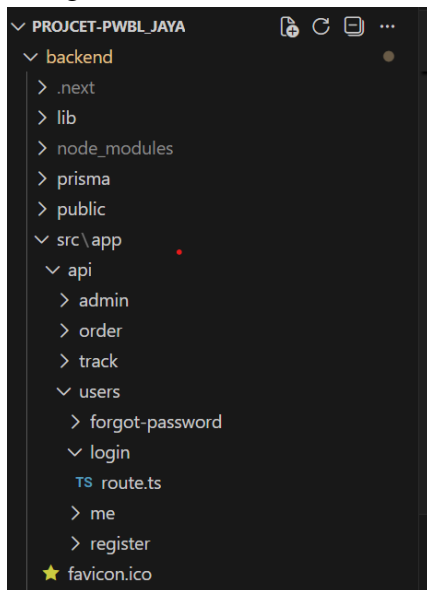
## Laporan Dokumentasi project pwbl\_team\_jaya

Nama : Ferdian Alfari

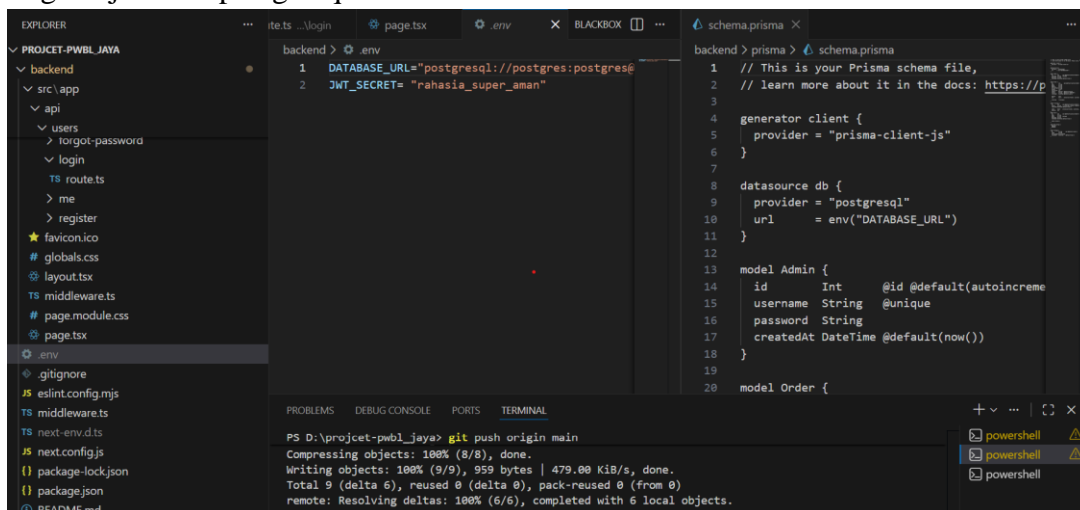
Npm : 23313007

Kelas : TI23 A

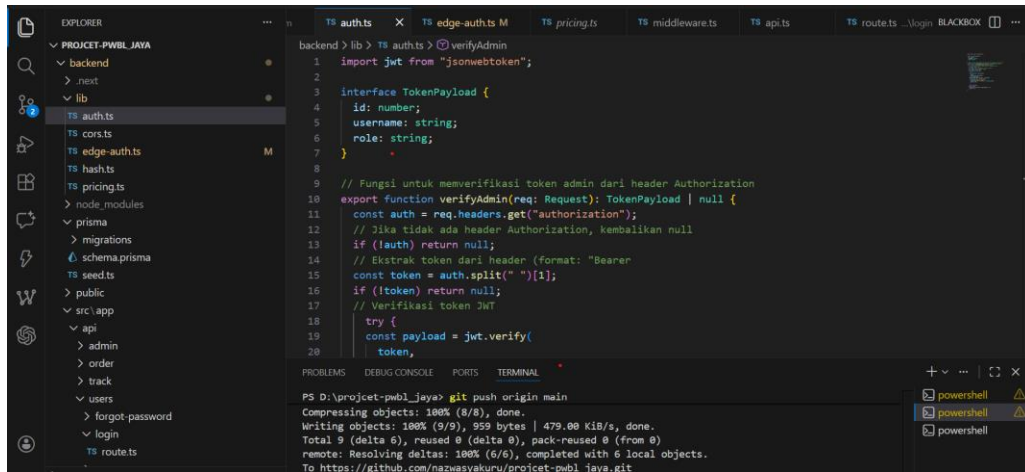
1. Menginstall framework next dengan nama backend, menginstall prisma client



2. Setelah itu buat schema prisma untuk membuat schema database dan ubah file .env agar migrasi jalan ke postgresql



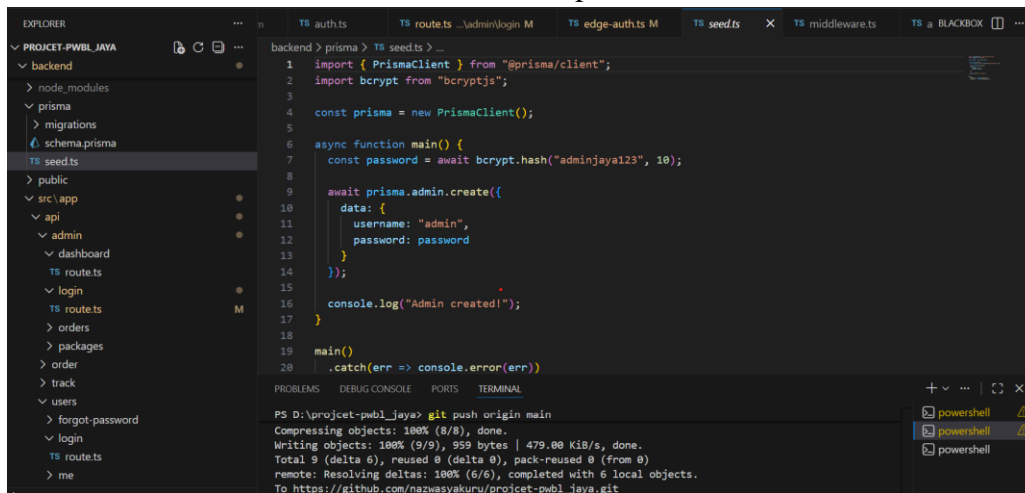
3. Buat folder lib dan buat file auth untuk authorization untuk admin, cors untuk menghubungkan fe, edge-auth autentikasi pada environment edge, hash untuk hashing dan verifikasi data sensitif, seperti password, pricing untuk mengelola logika harga, paket layanan, atau perhitungan biaya.



The screenshot shows a VS Code editor with a project named 'PROJECT-PWBL\_JAVA'. The file explorer on the left shows a folder structure with 'lib' containing 'auth.ts'. The main editor displays the code for 'verifyAdmin.ts' in the 'lib/auth.ts' file. The code defines a 'TokenPayload' interface with fields 'id', 'username', and 'role'. It then implements a 'verifyAdmin' function that checks the 'authorization' header, extracts the token, and verifies it using 'jsonwebtoken'. The terminal at the bottom shows the command 'git push origin main' being executed successfully.

```
1 import jwt from "jsonwebtoken";
2
3 interface TokenPayload {
4   id: number;
5   username: string;
6   role: string;
7 }
8
9 // Fungsi untuk memverifikasi token admin dari header Authorization
10 export function verifyAdmin(req: Request): TokenPayload | null {
11   const auth = req.headers.get("authorization");
12   // Jika tidak ada header Authorization, kembalikan null
13   if (!auth) return null;
14   // Ekstrak token dari header (format: "Bearer ")
15   const token = auth.split(" ")[1];
16   if (!token) return null;
17   // Verifikasi token JWT
18   try {
19     const payload = jwt.verify(
20       token,
```

4. Membuat file seed untuk membuat akun super admin



The screenshot shows a VS Code editor with a project named 'PROJECT-PWBL\_JAVA'. The file explorer on the left shows a folder structure with 'src/app' containing 'seed.ts'. The main editor displays the code for 'seed.ts' in the 'src/app/seed.ts' file. The code imports 'PrismaClient' and 'bcrypt' from 'prisma/client' and 'bcryptjs' respectively. It then implements a 'main' function that creates a new 'PrismaClient', hashes a password, and creates an admin user with the username 'admin' and the hashed password. The terminal at the bottom shows the command 'git push origin main' being executed successfully.

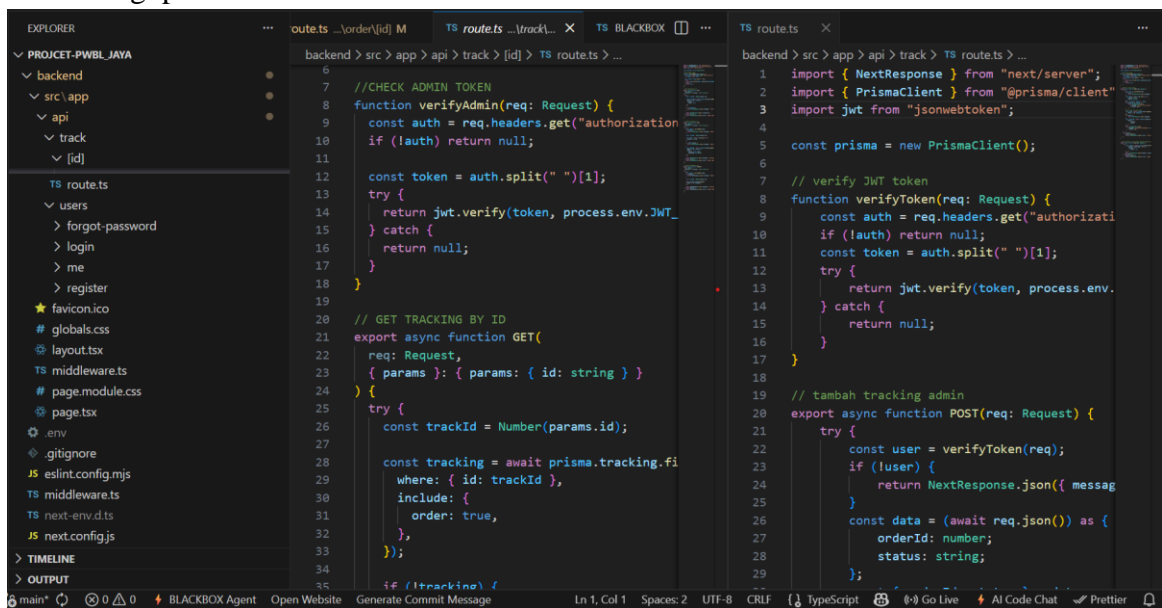
```
1 import { PrismaClient } from "prisma/client";
2 import bcrypt from "bcryptjs";
3
4 const prisma = new PrismaClient();
5
6 async function main() {
7   const password = await bcrypt.hash("adminjaya123", 10);
8
9   await prisma.admin.create({
10     data: {
11       username: "admin",
12       password: password
13     }
14   });
15   console.log("Admin created!");
16 }
17
18 main()
19   .catch(err => console.error(err))
20   .finally(() => prisma.$disconnect());
```

- The image shows a VS Code interface with a file explorer on the left and two code editors on the right. The file explorer shows the project structure for 'PROJECT-PWBL\_JAVA', including directories like 'backend', 'prisma', 'public', 'src/app', 'api', 'admin', 'dashboard', and 'TS route.ts'. The left code editor shows the 'TS route.ts' file in the 'admin' directory, containing code for handling login requests. The right code editor shows the 'TS route.ts' file in the 'dashboard' directory, containing code for handling dashboard requests.

- The screenshot displays the VS Code interface with the following details:

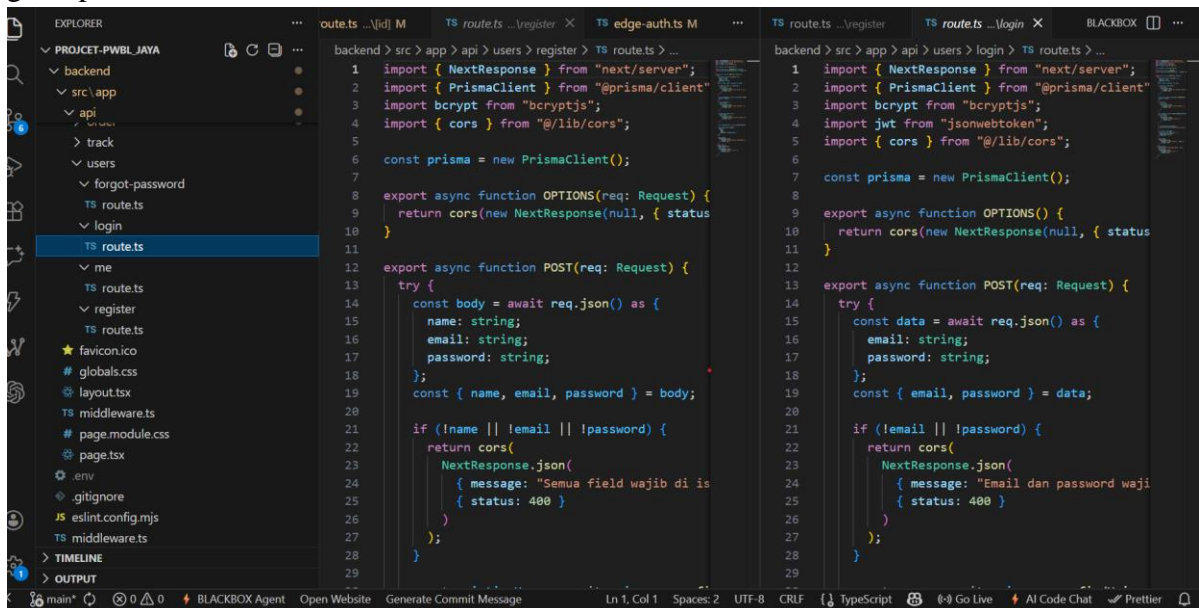
  - File Explorer (Left):** Shows the project structure for 'PROJECT-PWBL\_JAYA'. The 'routes.ts' file is selected under the 'api' directory.
  - Main Editor:** Displays the code for the DELETE endpoint in 'routes.ts'. The code includes a comment in Indonesian: "CEGAH DELETE JIKA SUDAH BAYAR" (Prevent delete if already paid). It checks if the order is paid using 'order.isPaid === true'. If not paid, it returns a 403 status. If paid, it deletes the order using 'prisma.order.delete({ where: { id } })' and returns a 200 status with the message "Order dihapus".
  - Search Bar (Top):** Contains the text 'backendl > src > app > order > [id] > routes.ts > DELETE'.
  - Status Bar (Bottom):** Shows the current position as 'Ln 168, Col 8'.

7. Buat folder track untuk users bisa melihat tracking pesanannya dengan id, dan pada slug admin bisa update status pesanannya, user hanya bisa melihat tetapi admin bisa melihat serta mengupdate



```
6 //CHECK ADMIN TOKEN
7
8 function verifyAdmin(req: Request) {
9   const auth = req.headers.get("authorization")
10   if (!auth) return null;
11
12   const token = auth.split(" ")[1];
13   try {
14     return jwt.verify(token, process.env.JWT_
15   } catch {
16     return null;
17   }
18 }
19
20 // GET TRACKING BY ID
21 export async function GET(
22   req: Request,
23   { params }: { params: { id: string } }
24 ) {
25   try {
26     const trackId = Number(params.id);
27
28     const tracking = await prisma.tracking.fi
29     where: { id: trackId },
30     include: {
31       order: true,
32     },
33   });
34
35   if (!tracking) {
36
37   }
38 }
39
40 // tambah tracking admin
41 export async function POST(req: Request) {
42   try {
43     const user = verifyToken(req);
44     if (!user) {
45       return NextResponse.json({ message
46     }
47
48     const data = (await req.json()) as {
49       orderId: number;
50       status: string;
51     };
52   }
53 }
```

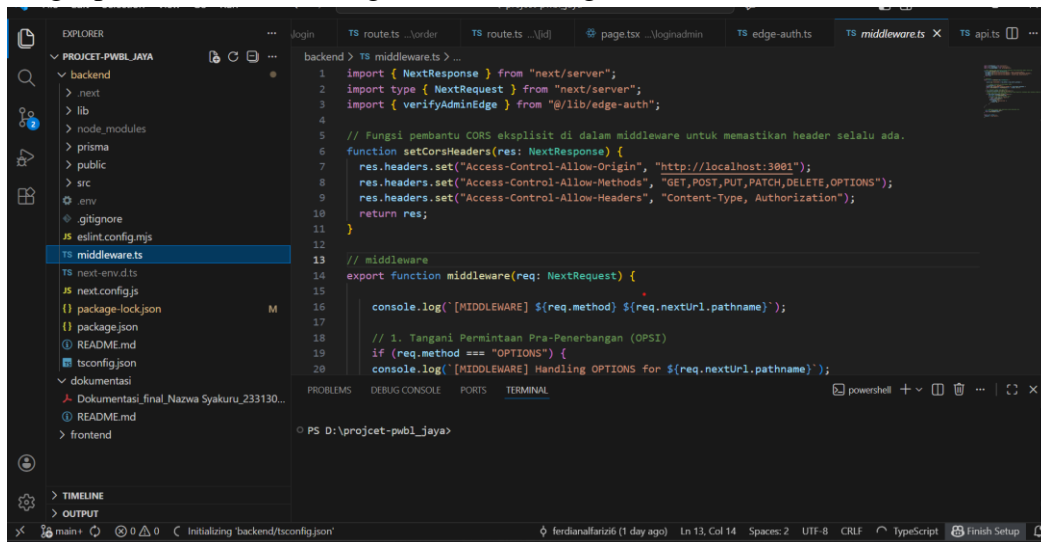
8. Buat folder users, didalamnya ada beberapa folder lagi yaitu register,login, forgot password, me, folder me untuk melihat profile nya dan mengupdate profilnya seperti ganti password



```
1 import { NextResponse } from "next/server";
2 import { PrismaClient } from "@prisma/client"
3 import bcrypt from "bcryptjs";
4 import { cors } from "@lib/cors";
5
6 const prisma = new PrismaClient();
7
8 export async function OPTIONS(req: Request) {
9   return cors(new NextResponse(null, { status
10 }
11
12 export async function POST(req: Request) {
13   try {
14     const body = await req.json() as {
15       name: string;
16       email: string;
17       password: string;
18     };
19     const { name, email, password } = body;
20
21     if (!name || !email || !password) {
22       return cors(
23         NextResponse.json(
24           { message: "Semua field wajib di is
25             ( status: 400 )
26         )
27       );
28     }
29   }
30 }
```

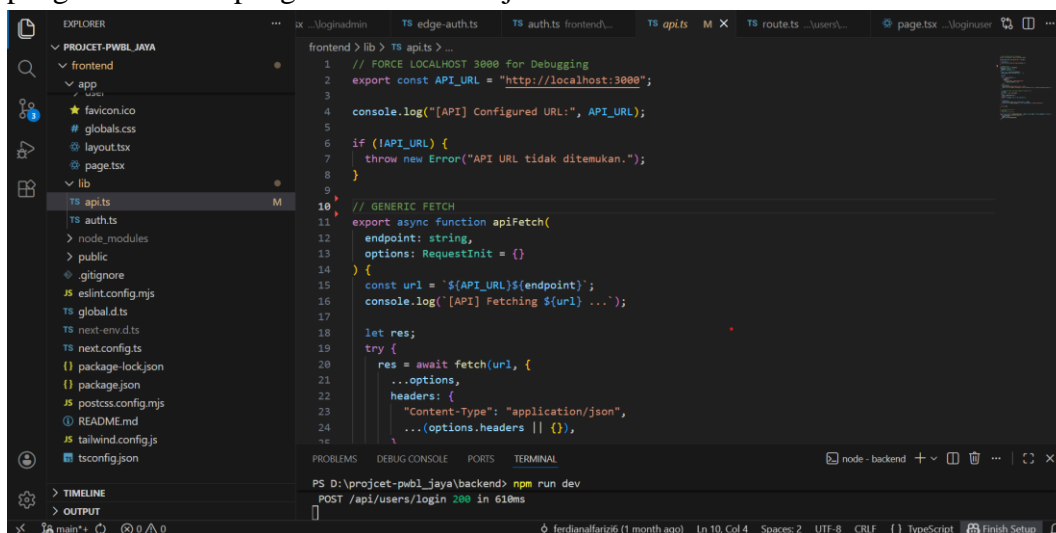
```
1 import { NextResponse } from "next/server";
2 import { PrismaClient } from "@prisma/client"
3 import bcrypt from "bcryptjs";
4 import jwt from "jsonwebtoken";
5 import { cors } from "@lib/cors";
6
7 const prisma = new PrismaClient();
8
9 export async function OPTIONS() {
10   return cors(new NextResponse(null, { status
11 }
12
13 export async function POST(req: Request) {
14   try {
15     const data = await req.json() as {
16       email: string;
17       password: string;
18     };
19     const { email, password } = data;
20
21     if (!email || !password) {
22       return cors(
23         NextResponse.json(
24           { message: "Email dan password waji
25             ( status: 400 )
26         )
27       );
28     }
29   }
30 }
```

9. Buat file middleware untuk membuat jembatan antara backend dan frontend, membuat fungsi pembantu CORS dengan 3 access origin, method, dan headers untuk auth



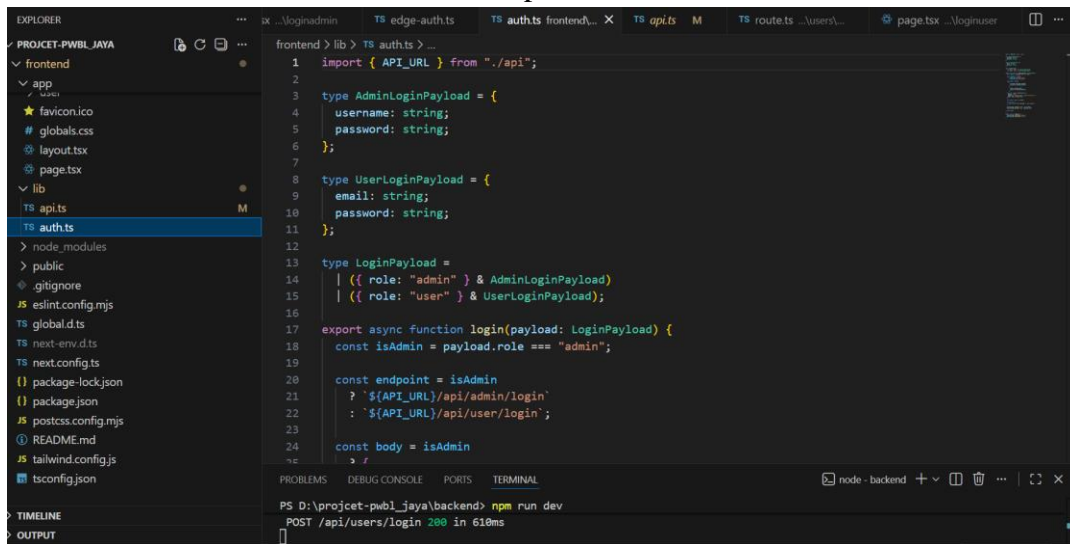
```
1 import { NextResponse } from "next/server";
2 import type { NextRequest } from "next/server";
3 import { verifyAdminEdge } from "@lib/edge-auth";
4
5 // Fungsi pembantu CORS eksplisit di dalam middleware untuk memastikan header selalu ada.
6 function setCorsHeaders(res: NextResponse) {
7   res.headers.set("Access-Control-Allow-Origin", "http://localhost:3001");
8   res.headers.set("Access-Control-Allow-Methods", "GET,POST,PUT,PATCH,DELETE,OPTIONS");
9   res.headers.set("Access-Control-Allow-Headers", "Content-Type, Authorization");
10  return res;
11}
12
13 // middleware
14 export function middleware(req: NextRequest) {
15
16   console.log(`[MIDDLEWARE] ${req.method} ${req.nextUrl.pathname}`);
17
18   // 1. Tangani Permintaan Pra-Penerbangan (OPSI)
19   if (req.method === "OPTIONS") {
20     console.log(`[MIDDLEWARE] Handling OPTIONS for ${req.nextUrl.pathname}`);
21     return setCorsHeaders(new NextResponse());
22   }
23
24   // ... (other logic)
25 }
```

10. Api.ts membuat function api url, pusat konfigurasi dan fungsi API untuk menghubungkan frontend dengan backend. File ini berperan dalam mengatur base URL backend serta menyediakan fungsi auth dan request HTTP (GET, POST, PUT, DELETE) agar pengambilan dan pengiriman data menjadi lebih terstruktur dan konsisten.



```
1 // FORCE LOCALHOST 3000 for Debugging
2 export const API_URL = "http://localhost:3000";
3
4 console.log("[API] Configured URL:", API_URL);
5
6 if (!API_URL) {
7   throw new Error("API URL tidak ditemukan.");
8 }
9
10 // GENERIC FETCH
11 export async function apiFetch(
12   endpoint: string,
13   options: RequestInit = {}
14 ) {
15   const url = `${API_URL}${endpoint}`;
16   console.log("[API] Fetching ${url} ...");
17
18   let res;
19   try {
20     res = await fetch(url, {
21       ...options,
22       headers: {
23         "Content-Type": "application/json",
24         ...(options.headers || {}),
25       },
26     });
27   } catch (error) {
28     console.error("Error fetching data from API:", error);
29   }
30 }
```

11. File auth.ts digunakan untuk menangani proses autentikasi (login) pada sisi frontend. File ini mendefinisikan struktur data login untuk admin dan user, sehingga data yang dikirim ke backend lebih terkontrol dan sesuai tipe.

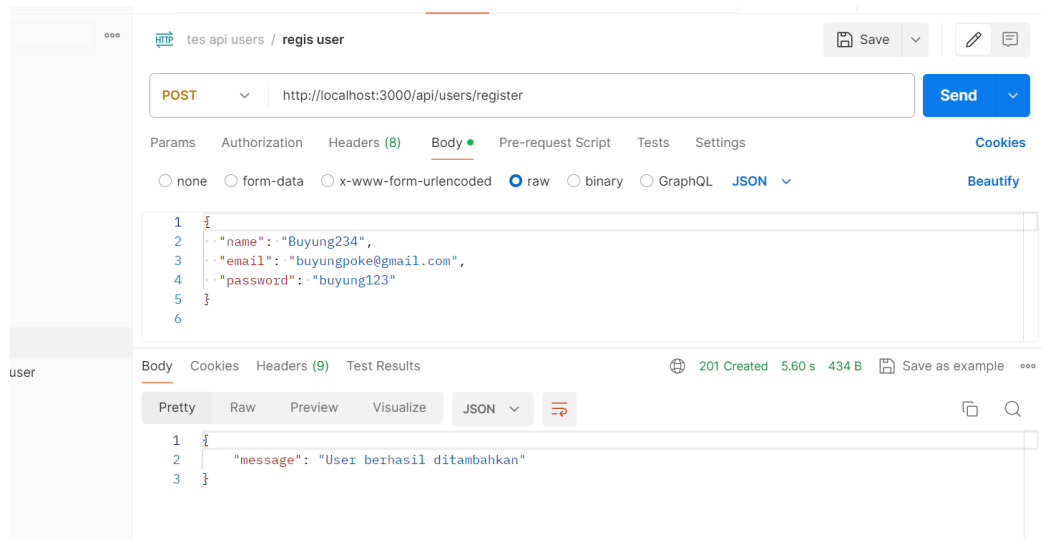


```
1 import { API_URL } from "../api";
2
3 type AdminLoginPayload = {
4   username: string;
5   password: string;
6 };
7
8 type UserLoginPayload = {
9   email: string;
10  password: string;
11 };
12
13 type LoginPayload =
14   | ({ role: "admin" } & AdminLoginPayload)
15   | ({ role: "user" } & UserLoginPayload);
16
17 export async function login(payload: LoginPayload) {
18   const isAdmin = payload.role === "admin";
19
20   const endpoint = isAdmin
21     ? `${API_URL}/api/admin/login`
22     : `${API_URL}/api/user/login`;
23
24   const body = isAdmin
25     ? { username: payload.username, password: payload.password }
26     : { email: payload.email, password: payload.password };
27
28   const response = await fetch(endpoint, {
29     method: "POST",
30     headers: { "Content-Type": "application/json" },
31     body: JSON.stringify(body),
32   });
33
34   if (!response.ok) {
35     throw new Error("Login failed");
36   }
37
38   const data = await response.json();
39   return data;
40 }
```

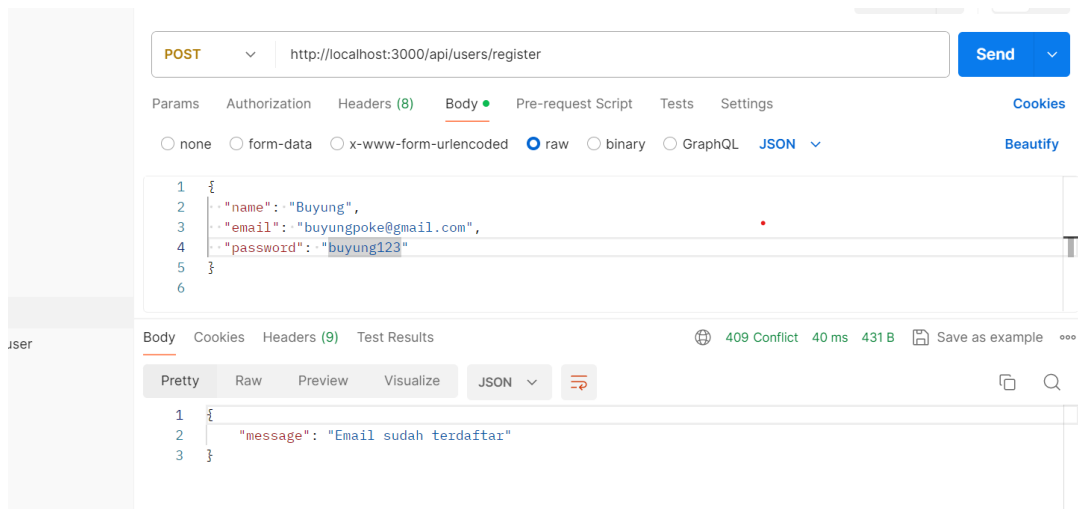
12. Dokumentasi postman, untuk melihat apakah api sudah sesuai dengan semua methodenya dan mendapatkan token auth

## USER

Registrasi dengan Field name, email, password, dan respon berhasil

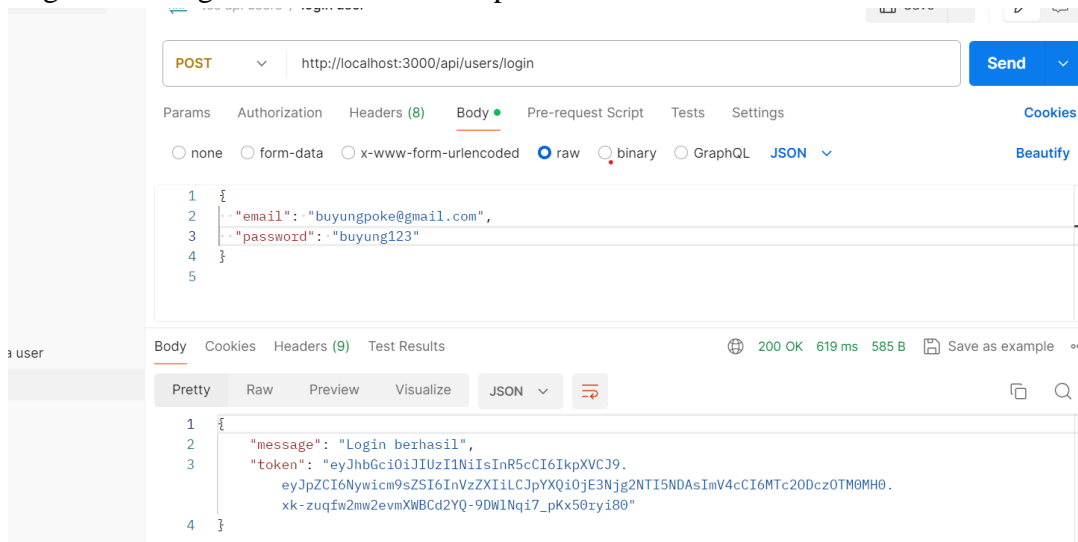


Registrasi ulang users dan melihat respon “email sudah terdaftar ” maka gagal registrasi



## Login Users

Login Users registrasi tadi dan mendapatkan token auth



## Login Admin

Login admin berhasil dan mendapat token

The screenshot shows a REST client interface with a POST request to `http://localhost:3000/api/admin/login`. The request body is a JSON object with `username: "admin"` and `password: "adminjaya123"`. The response status is `200 OK` with a response time of `473 ms` and a body size of `612 B`. The response body is a JSON object containing a success message and a long JWT token.

```
1 {
2   "username": "admin",
3   "password": "adminjaya123"
4 }
```

```
1 {
2   "message": "Login berhasil",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidXN1cm5hbWUiOiJhZG1pb2IsInJvbmUiOiJhZG1pb2IsIm1hdCI6MTc2ODY1NTMzNSwiZXhwIjoxNzY4NzQxNzU1fQ.eyJmKtsIjcoGJs20f6BJefeHgC5FHy6iazi09BpnSjrc"
4 }
```

## Order

order berhasil di buat, api order aman

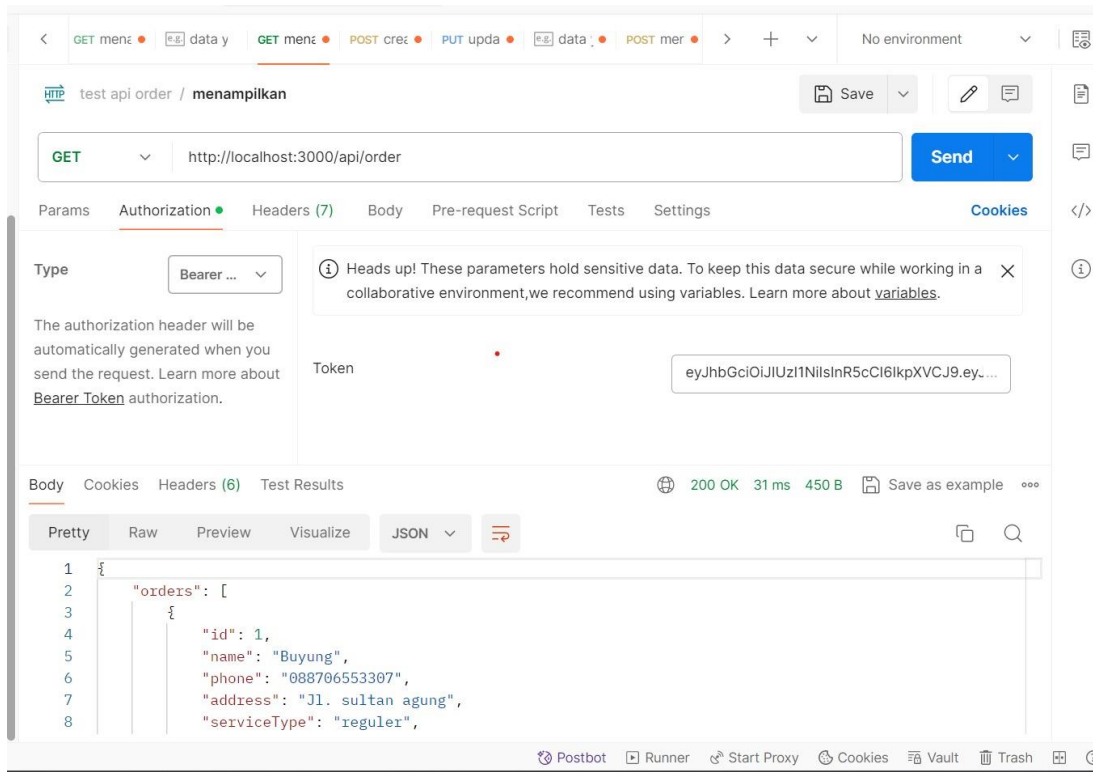
The screenshot shows a REST client interface with a POST request to `http://localhost:3000/api/order`. The request body is a JSON object with fields for name, phone, address, service type, weight, and total price. The response status is `201 Created` with a response time of `267 ms` and a body size of `477 B`. The response body is a JSON object containing a success message and the created order details.

```
1 {
2   "name": "Buyung",
3   "phone": "088706553307",
4   "address": "Jl. sultan agung",
5   "serviceType": "reguler",
6   "weight": 5,
7   "totalPrice": 25000
8 }
```

```
1 {
2   "message": "Order dibuat",
3   "order": {
4     "id": 1,
5     "name": "Buyung",
6     "phone": "088706553307",
7     "address": "Jl. sultan agung",
8     "serviceType": "reguler",
```



Lihat order berdasarkan pesanan user itu sendiri tidak bisa lihat pesanan user yang lain



test api order / menampilkan

GET http://localhost:3000/api/order

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Type Bearer ...

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

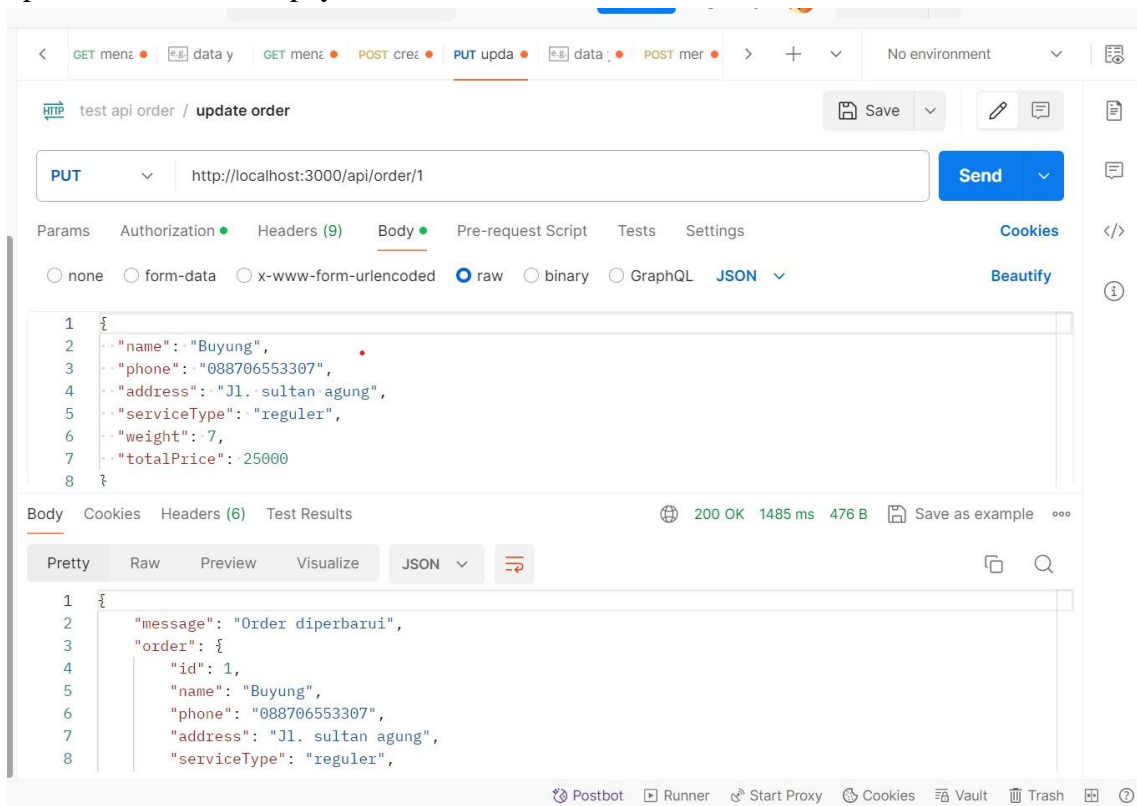
Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

Body Cookies Headers (6) Test Results 200 OK 31 ms 450 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "orders": [
3     {
4       "id": 1,
5       "name": "Buyung",
6       "phone": "088706553307",
7       "address": "Jl. sultan agung",
8       "serviceType": "reguler",
```

update order sebelum payment



test api order / update order

PUT http://localhost:3000/api/order/1

Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautifuly

```
1 {
2   "name": "Buyung",
3   "phone": "088706553307",
4   "address": "Jl. sultan agung",
5   "serviceType": "reguler",
6   "weight": 7,
7   "totalPrice": 25000
8 }
```

Body Cookies Headers (6) Test Results 200 OK 1485 ms 476 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Order diperbarui",
3   "order": {
4     "id": 1,
5     "name": "Buyung",
6     "phone": "088706553307",
7     "address": "Jl. sultan agung",
8     "serviceType": "reguler",
```

## Track

Lihat Tracking pesanan user, tidak bisa melihat pesanan user lain

tes api track / get tampil tracking

GET http://localhost:3000/api/track

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (6) Test Results 200 OK 64 ms 536 B Save as example

Pretty Raw Preview Visualize JSON

```
5 {
6   "status": "dicuci",
7   "timestamp": "2025-12-20T16:20:07.821Z",
8   "orderId": 1,
9   "order": {
10    "id": 1,
11    "name": "Buyung",
12    "phone": "088706553307",
13    "address": "Jl. sultan agung",
14    "serviceType": "reguler",
15    "weight": 7,
16    "totalPrice": 25000,
17    "isPaid": false,
18    "status": "CREATED",
19    "createdAt": "2025-12-20T16:15:01.972Z",
20    "userId": null
21  }
22 }
```

Tracking di tambahkan admin ketika user melakukan payment

tes api track / menambahkan tracking

POST http://localhost:3000/api/track

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "orderId": 1,
3   "status": "dicuci"
4 }
```

Body Cookies Headers (6) Test Results 201 Created 500 ms 348 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Tracking ditambahkan",
3   "tracking": {
4     "id": 1,
5     "status": "dicuci",
6     "timestamp": "2025-12-20T16:20:07.821Z",
7     "orderId": 1
8   }
9 }
```