

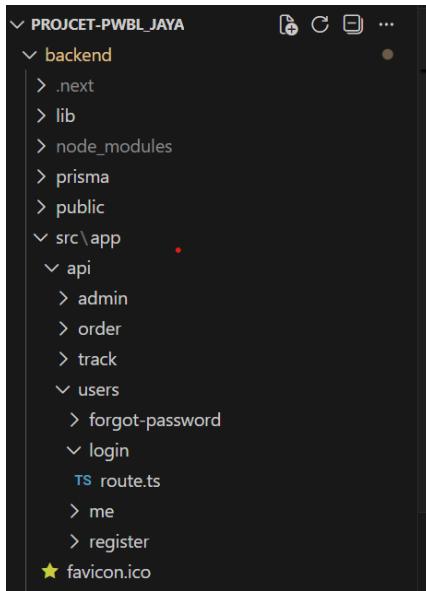
Laporan Dokumentasi project pwbl_team_jaya

Nama : Ferdian Alfarizi

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

```
schema.prisma
1 // This is your Prisma schema file,
2 // learn more about it in the docs: https://www.prisma.io/docs/concepts/components/prisma-schema/prisma-schema-file
3
4 generator client {
5   provider = "prisma-client-js"
6 }
7
8 datasource db {
9   provider = "postgresql"
10  url     = env("DATABASE_URL")
11 }
12
13 model Admin {
14   id      Int    @id @default(autoincrement())
15   username String @unique
16   password String
17   createdAt DateTime @default(now())
18 }
19
20 model Order {
```

```
.env
1 DATABASE_URL="postgresql://postgres:postgres@localhost:5432/prisma"
2 JWT_SECRET= "rahasia_super_aman"
```

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.

```

PROJECT-PWBL_JAYA
└── backend
    ├── .next
    └── lib
        ├── TS auth.ts
        ├── TS cors.ts
        ├── TS edge-auth.ts
        ├── TS hashes.ts
        ├── TS pricing.ts
        ├── node_modules
        ├── prisma
        ├── migrations
        └── schema.prisma
    ├── seed.ts
    └── src\app
        ├── api
        │   ├── admin
        │   ├── order
        │   └── track
        ├── users
        │   ├── forgot-password
        │   └── login
        └── route.ts

```

```

TS auth.ts
import jwt from "jsonwebtoken";
interface TokenPayload {
  id: number;
  username: string;
  role: string;
}
// Fungsi untuk memverifikasi token admin dari header Authorization
export function verifyAdmin(req: Request): TokenPayload | null {
  const auth = req.headers.get("authorization");
  // Jika tidak ada header Authorization, kembalikan null
  if (!auth) return null;
  // Ekstrak token dari header (format: "Bearer")
  const token = auth.split(" ")[1];
  if (!token) return null;
  // Verifikasi token JWT
  try {
    const payload = jwt.verify(
      token,
      process.env.JWT_SECRET!
    );
    return payload;
  } catch (err) {
    console.error(err);
    return null;
  }
}

PROBLEMS DEBUG CONSOLE PORTS TERMINAL
PS D:\project-pwbl_jaya> git push origin main
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 959 bytes | 479.00 KiB/s, done.
Total 9 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/nazwasyakuru/project-pwbl_jaya.git
  
```

4. Membuat filde seed untuk membuat akun super admin

```

PROJECT-PWBL_JAYA
└── backend
    ├── prisma
    ├── migrations
    └── schema.prisma
    ├── seed.ts
    └── src\app
        ├── api
        │   ├── admin
        │   └── dashboard
        ├── orders
        ├── packages
        ├── order
        ├── track
        └── users
            ├── forgot-password
            ├── login
            └── me

```

```

TS seed.ts
import { PrismaClient } from "@prisma/client";
import bcrypt from "bcryptjs";
const prisma = new PrismaClient();
async function main() {
  const password = await bcrypt.hash("adminjaya123", 10);
  await prisma.admin.create({
    data: {
      username: "admin",
      password: password
    }
  });
  console.log("Admin created!");
}
main()
  .catch((err) => console.error(err))
  
```

```

PROBLEMS DEBUG CONSOLE PORTS TERMINAL
PS D:\project-pwbl_jaya> git push origin main
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 959 bytes | 479.00 KiB/s, done.
Total 9 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/nazwasyakuru/project-pwbl_jaya.git
  
```

5. Membuat folder admin dalam api, menambahkan beberapa folder untuk admin seperti login admin untuk login, dashboard admin unutuk monitoring pesanan, order untuk konfirmasi pesanan user, dan packages untuk menambahkan paket pesanan seperti ekspres atau regular.

```

PROJECT-PWBL_JAVA
└── backend
    ├── prisma
    ├── public
    └── src
        └── app
            ├── api
            │   └── admin
            │       ├── dashboard
            │       └── login
            └── orders\[id]\confirm
                └── TS route.ts
                    └── TS route.ts
            └── packages
                └── [id]
                    └── TS route.ts

```

```

TS route.ts ...\\adminLogin M TS route BLACKBOX
backend > src > app > api > admin > login > TS route.ts > ...
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     // Admin login route
13     export async function POST(req: Request) {
14         try {
15             const body = await req.json();
16             const { username, password } = body as {
17                 username: string;
18                 password: string;
19             };
20             const user = await prisma.user.findFirst({
21                 where: {
22                     email: username,
23                     password: bcrypt.hashSync(password, 10),
24                 },
25             });
26             if (!user) {
27                 return NextResponse.json(
28                     { message: "User not found" },
29                     { status: 404 }
30                 );
31             }
32             const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET);
33             return NextResponse.json(
34                 { token },
35                 { status: 200 }
36             );
37         } catch (error) {
38             console.error(error);
39             return NextResponse.json(
40                 { message: "Internal server error" },
41                 { status: 500 }
42             );
43         }
44     }
45     // CEGAH DELETE JIKA SUDAH BAYAR
46     if (order.isPaid === true) {
47         return NextResponse.json(
48             { message: "Order sudah dibayar, tidak bisa dihapus. Silakan hubungi CS." },
49             { status: 403 }
50         );
51     }
52     // USER HANYA BISA HAPUS ORDER MILIKNYA SENDIRI (kecuali admin)
53     if (user.role !== "admin" && user.id !== order.userId) {
54         return NextResponse.json(
55             { message: "Kamu tidak punya akses ke order ini" },
56             { status: 403 }
57         );
58     }
59     // DELETE DIPERBOLEHKAN
60     await prisma.order.delete({ where: { id } });
61
62     return NextResponse.json(
63         { message: "Order dihapus" },
64         { status: 200 }
65     );
66 }

```

6. Buat folder order dan slug id (user bisa update order ketika belum payment) agar user bisa hit api untuk menambahkan order, dan jika order duplikat tidak dapat membuat order

```

PROJECT-PWBL_JAVA
└── backend
    ├── prisma
    ├── migrations
    ├── schema.prisma
    ├── seed.ts
    ├── public
    └── src
        └── app
            ├── api
            │   └── admin
            │       └── order
            └── order
                └── [id]
                    └── TS route.ts
                        └── TS route.ts
            └── track
            └── users
                ├── forgot-password
                ├── login
                ├── me
                └── register
            └── TS middleware.ts
            └── TS route.ts

```

```

PROJECT-PWBL_JAVA
└── backend
    ├── prisma
    ├── migrations
    ├── schema.prisma
    ├── seed.ts
    ├── public
    └── src
        └── app
            ├── api
            └── order
                └── TS route.ts
                    └── TS route.ts
            └── track
            └── users
                ├── forgot-password
                ├── login
                ├── me
                └── register
            └── TS middleware.ts
            └── TS route.ts

```

```

TS route.ts ...\\adminLogin M TS route.ts ...\\order M TS edge-auth.ts M TS middlewares M TS appts M TS n BLACKBOX
backend > src > app > api > order > TS route.ts > POST > existingOrder
20 export async function POST(req: Request) {
21     const body = await req.json();
22     const { name, phone, address, serviceType, weight, totalPrice } = body;
23     const existingOrder = await prisma.order.findFirst({
24         where: {
25             phone,
26             serviceType,
27             isPaid: false, // aturan: belum dibayar tidak boleh buat duplikat
28         },
29     });
30
31     if (existingOrder) {
32         return NextResponse.json(
33             { message: "Order dengan layanan dan nomor ini sudah ada" },
34             { status: 409 }
35         );
36     }
37
38     const newOrder = await prisma.order.create({
39         data: {
40             name,
41             phone,
42             address,
43             serviceType,
44             weight,
45             totalPrice: totalPrice ?? 0, // default
46         },
47     });
48
49     return NextResponse.json(
50         { message: "Order berhasil dibuat" },
51         { status: 201 }
52     );
53 }

```

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

```

    EXPLORER
    ...
    PROJECT-PWBL_JAYA
    ...
    backend > src > app > api > track > [id] > TS route.ts > ...
    ...
    TS route.ts
    ...
    6
    7 //CHECK ADMIN TOKEN
    8
    9 function verifyAdmin(req: Request) {
    10   const auth = req.headers.get("authorization");
    11   if (!auth) return null;
    12
    13   const token = auth.split(" ")[1];
    14   try {
    15     return jwt.verify(token, process.env.JWT_);
    16   } catch {
    17     return null;
    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.findFirst({
    29       where: { id: trackId },
    30       include: {
    31         order: true,
    32       },
    33     });
    34
    35     if (!tracking) {
    36       return cors(
    37         NextResponse.json(
    38           { message: "Tracking not found" },
    39           { status: 404 }
    40         )
    41       );
    42     }
    43
    44     return cors(
    45       NextResponse.json(
    46         { tracking },
    47         { status: 200 }
    48       )
    49     );
    50   }
    51 }
    ...
    TS route.ts > ...
    ...
    1 import { NextResponse } from "next/server";
    2 import { PrismaClient } from "@prisma/client";
    3 import jwt from "jsonwebtoken";
    4
    5 const prisma = new PrismaClient();
    ...
    7 // verify JWT token
    8 function verifyToken(req: Request) {
    9   const auth = req.headers.get("authorization");
    10  if (!auth) return null;
    11  const token = auth.split(" ")[1];
    12  try {
    13    return jwt.verify(token, process.env.JWT_);
    14  } catch {
    15    return null;
    16  }
    17
    18 // tambah tracking admin
    19 export async function POST(req: Request) {
    20   try {
    21     const user = verifyToken(req);
    22     if (!user) {
    23       return NextResponse.json({ message: "Unauthorized" }, { status: 401 });
    24     }
    25     const data = (await req.json()) as {
    26       orderId: number;
    27       status: string;
    28     };
    29     const tracking = await prisma.tracking.create({
    30       data,
    31     });
    32     return cors(
    33       NextResponse.json(
    34         { tracking },
    35         { status: 201 }
    36       )
    37     );
    38   }
    39 }
    ...
  
```

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

```

    EXPLORER
    ...
    PROJECT-PWBL_JAYA
    ...
    backend > src > app > api > users > register > TS route.ts > ...
    ...
    TS route.ts
    ...
    1 import { NextResponse } from "next/server";
    2 import { PrismaClient } from "@prisma/client";
    3 import bcrypt from "bcryptjs";
    4 import { cors } from "@lib/cors";
    ...
    6 const prisma = new PrismaClient();
    ...
    8 export async function OPTIONS(req: Request) {
    9   return cors(new NextResponse(null, { status: 200 }));
    10 }
    ...
    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 diisi" },
    25           { status: 400 }
    26         )
    27       );
    28     }
    29   }
    ...
    TS route.ts > ...
    ...
    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";
    ...
    7 const prisma = new PrismaClient();
    ...
    9 export async function OPTIONS() {
    10   return cors(new NextResponse(null, { status: 200 }));
    11 }
    ...
    14 export async function POST(req: Request) {
    15   try {
    16     const data = await req.json() as {
    17       email: string;
    18       password: string;
    19     };
    20     const { email, password } = data;
    21
    22     if (!email || !password) {
    23       return cors(
    24         NextResponse.json(
    25           { message: "Email dan password wajib diisi" },
    26           { status: 400 }
    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

```

PROJECT-PWBL_JAYA
└── backend
    ├── .next
    ├── lib
    ├── node_modules
    ├── prism
    ├── public
    ├── src
    └── .env
        └── .gitignore
    └── eslint.config.mjs
TS middleware.ts
TS next-env.d.ts
JS next.config.js
(I) package-lock.json
(I) package.json
(I) README.md
tsconfig.json
└── Dokumentasi
    └── Dokumentasi_final_Nazwa_Syakuru_233130...
        └── README.md
    └── frontend
        └── ...
TS middleware.ts
TS routes.ts ...\\order
TS routes.ts ...\\id
TS page.tsx ...\\loginadmin
TS edge-auth.ts
TS middleware.ts X
TS api.ts ...
PROBLEMS DEBUG CONSOLE PORTS TERMINAL
PS D:\project-pwbl_jaya>

```

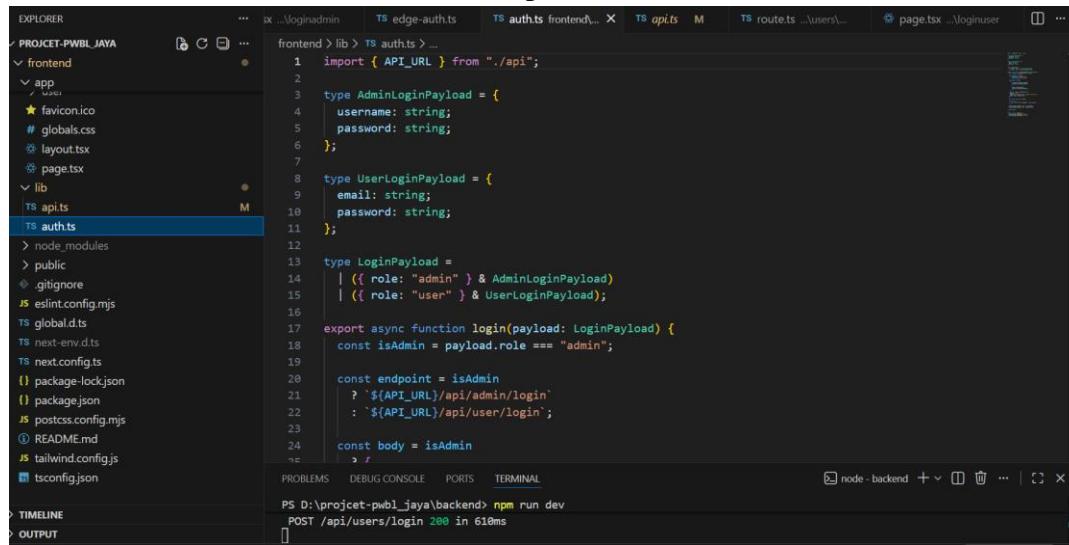
10. Api.ts membuat function api url, pusat konfigurasi dan fungsi API untuk menghubungkan frontend dengan backend. File ini berperan dalam mengatur base URL backend serta menyediakan fungsi auth dan request HTTP (GET, POST, PUT, DELETE) agar pengambilan dan pengiriman data menjadi lebih terstruktur dan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

```

PROJECT-PWBL_JAYA
└── frontend
    ├── lib
    │   └── TS api.ts M
    ├── app
    ├── favicon.ico
    ├── # globals.css
    ├── layout.tsx
    └── page.tsx
TS api.ts M
TS auth.ts
TS lib
TS node_modules
TS public
└── .gitignore
JS eslint.config.mjs
TS global.d.ts
TS next-env.ts
TS next.config.ts
(I) package-lock.json
(I) package.json
JS postcss.config.mjs
(I) README.md
JS tailwind.config.js
tsconfig.json
└── ...
PROBLEMS DEBUG CONSOLE PORTS TERMINAL
PS D:\project-pwbl_jaya\backend> npm run dev
POST /api/users/login 200 in 610ms

```

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.



```

PROJECT-PWBL_JAVA
  ✓ frontend
    ✓ app
      favicon.ico
      globals.css
      layout.tsx
      page.tsx
    ✓ lib
      TS api.ts
      TS auth.ts
        > node_modules
        > public
        .gitignore
        JS eslint.config.js
        TS global.d.ts
        TS next-env.d.ts
        TS next.config.ts
        () package-lock.json
        () package.json
        JS postcss.config.js
        README.md
        JS tailwind.config.js
        tsconfig.json

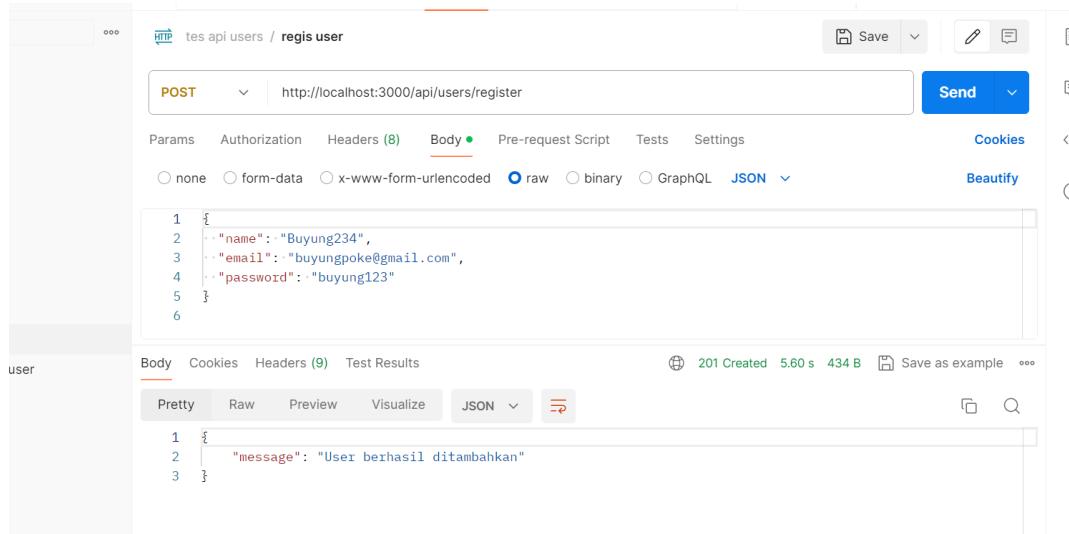
TS auth.ts
  import { API_URL } from "./api";
  type AdminLoginPayload = {
    username: string;
    password: string;
  };
  type UserLoginPayload = {
    email: string;
    password: string;
  };
  type LoginPayload =
    | ({ role: "admin" } & AdminLoginPayload)
    | ({ role: "user" } & UserLoginPayload);
  export async function login(payload: LoginPayload) {
    const isAdmin = payload.role === "admin";
    const endpoint = isAdmin
      ? `${API_URL}/api/admin/login`
      : `${API_URL}/api/user/login`;
    const body = isAdmin
      > /
    return fetch(endpoint, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(payload),
    }).then((res) => res.json());
  }

```

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

USER

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



tes api users / regis user

POST http://localhost:3000/api/users/register

Params Authorization Headers (8) Body Body Pre-request Script Tests Settings Cookies </>

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

```

1: {
2:   "name": "Buyung234",
3:   "email": "buyungpoke@gmail.com",
4:   "password": "buyung123"
5: }
6

```

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```

1: {
2:   "message": "User berhasil ditambahkan"
3: }

```

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

The screenshot shows a POST request to `http://localhost:3000/api/users/register`. The request body is a JSON object with fields `name`, `email`, and `password`. The response status is 409 Conflict, indicating that the email address is already registered.

```
POST http://localhost:3000/api/users/register
{
  "name": "Buyung",
  "email": "buyungpoke@gmail.com",
  "password": "buyung123"
}
```

Body Cookies Headers (9) Test Results
Pretty Raw Preview Visualize JSON
1 {
2 "message": "Email sudah terdaftar"
3 }

409 Conflict 40 ms 431 B Save as example

Login Users

Login Users registrasi tadi dan mendapatkan token auth

The screenshot shows a POST request to `http://localhost:3000/api/users/login`. The request body is a JSON object with fields `email` and `password`. The response status is 200 OK, indicating successful login, and includes a JSON object with `message` and a long `token`.

```
POST http://localhost:3000/api/users/login
{
  "email": "buyungpoke@gmail.com",
  "password": "buyung123"
}
```

Body Cookies Headers (9) Test Results
Pretty Raw Preview Visualize JSON
1 {
2 "message": "Login berhasil",
3 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6Nyicm9sZSI6InVzZXIiLCJpYXQiOjE3Njg2NTI5NDAsImV4cCI6MTc2ODczOTM0MH0.
xk-zuqfw2mw2evmXWBCd2YQ-9DWlNqi7_pKx50ryi80"
4 }

200 OK 619 ms 585 B Save as example

Login Admin

Login admin berhasil dan mendapat token

The screenshot shows a Postman request to `http://localhost:3000/api/admin/login` with a POST method. The body contains the following JSON:

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

The response status is 200 OK, with a response time of 473 ms and a size of 612 B. The response body is:

```
1 {
2   "message": "Login berhasil",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidXNlcm5hbWUiOiJhZG1pbisInJvbGUiOiJhZG1pbisImhlhdCI6MTc2ODY1NTMzMNSwiZXhwIjoxNzY4NzQxNzM1fQ.-_mKtsIJcoGJs20f6BJeffeHgCSFHy6iazi09BpnSjrc"
4 }
```

Order

order berhasil di buat, api order aman

The screenshot shows a Postman request to `http://localhost:3000/api/order` with a POST method. The body contains the following JSON:

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

The response status is 201 Created, with a response time of 267 ms and a size of 477 B. The response body is:

```
1 {
2   "message": "Order dibuat",
3   "order": {
4     "id": 1,
5     "name": "Buyung",
6     "phone": "088706553307",
7     "address": "Jl. sultan agung",
8     "serviceType": "reguler",
9   }
10 }
```

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

The screenshot shows the Postman interface with a successful GET request to `http://localhost:3000/api/order`. The response body is a JSON object containing an array of orders, with the first order's details visible:

```
1 {
2   "orders": [
3     {
4       "id": 1,
5       "name": "Buyung",
6       "phone": "088706553307",
7       "address": "Jl. sultan agung",
8       "serviceType": "reguler",
9     }
10 }
```

update order sebelum payment

The screenshot shows the Postman interface with a successful PUT request to `http://localhost:3000/api/order/1`. The response body is a JSON object indicating the order was updated:

```
1 {
2   "message": "Order diperbarui",
3   "order": {
4     "id": 1,
5     "name": "Buyung",
6     "phone": "088706553307",
7     "address": "Jl. sultan agung",
8     "serviceType": "reguler",
9   }
10 }
```

Track

Lihat Tracking pesanan user, tidak bisa melihat pesanan user lain

The screenshot shows the Postman interface with a collection named 'rk'. A new request is being created with the URL `http://localhost:3000/api/track`. The method is set to GET. The response status is 200 OK with a response time of 64 ms and a body size of 536 B. The response content is a JSON object representing a tracking entry:

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

Tracking di tambahkan admin ketika user melakukan payment

The screenshot shows the Postman interface with a collection named 'rk'. A new request is being created with the URL `http://localhost:3000/api/track`. The method is set to POST. The response status is 201 Created with a response time of 500 ms and a body size of 348 B. The response content is a JSON object indicating success:

```
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     }
```