

Nama : Diva Afirlia Maheswari

NPM : 23313022

Kelas : TI 23 A

Laporan Dokumentasi Proyek

1. File route.ts ini digunakan untuk membuat **API endpoint** pada Next.js (App Router) yang berfungsi **mengambil seluruh data produk dari database** menggunakan Prisma ORM

```
@@ -0,0 +1,19 @@
+ import { NextResponse } from "next/server";
+ import { PrismaClient } from "@prisma/client";
+
+ const prisma = new PrismaClient();
+
+ // =====
+ // GET: Ambil Semua Produk
+ // =====
+ export async function GET() {
+   try {
+     const produk = await prisma.produk.findMany();
+     return NextResponse.json(produk);
+   } catch (error) {
+     return NextResponse.json(
+       { message: "Gagal mengambil data produk" },
+       { status: 500 }
+     );
+   }
+ }
```

Kode route.ts tersebut juga digunakan untuk membuat API endpoint **GET** pada Next.js yang berfungsi mengambil seluruh data produk dari database. Pada awal kode, NextResponse digunakan untuk mengatur response API, sedangkan PrismaClient berfungsi sebagai penghubung ke database. Prisma diinisialisasi agar aplikasi dapat menjalankan query database, kemudian fungsi GET dibuat untuk menangani permintaan pengambilan data produk. Data diambil menggunakan prisma.produk.findMany() dan dikembalikan ke client dalam bentuk JSON. Jika terjadi kesalahan saat proses pengambilan data, sistem akan menampilkan pesan error dengan status HTTP 500 sebagai penanda kesalahan pada sisi server.

2. Potongan kode ini digunakan untuk melakukan **validasi username** sebelum data pengguna diproses lebih lanjut. Sistem mengecek apakah nilai name yang dikirim sudah terdaftar di database dengan menggunakan fungsi prisma.user.findFirst() pada tabel user. Jika data dengan username tersebut ditemukan, maka API akan langsung menghentikan proses dan mengembalikan response JSON berisi pesan "*Username sudah digunakan*" dengan status HTTP **409 (Conflict)**, yang menandakan adanya bentrok data karena username harus bersifat unik. Validasi ini bertujuan untuk menjaga konsistensi data serta mencegah duplikasi akun pengguna dalam sistem.

```

+
+ // 4. Cek username (name) sudah ada atau belum
+ const nameExist = await prisma.user.findFirst({
+   where: { name }
+ });
+
+ if (nameExist) {
+   return NextResponse.json(
+     { message: "Username sudah digunakan" },
+     { status: 409 }
+   );
+ }
+
+ }
```

3. DEETE : Hapus Produk

```
+  
+ / =====  
+ // DELETE: Hapus Produk per ID  
+ // =====  
+ export async function DELETE(req: Request) {  
+   try {  
+     const url = new URL(req.url);  
+     const id = Number(url.searchParams.get("id"));  
+  
+     if (!id) {  
+       return NextResponse.json(  
+         { message: "ID produk tidak ditemukan" },  
+         { status: 400 }  
+       );  
+     }  
+  
+     await prisma.produk.delete({  
+       where: { id },  
+     });  
+   }  
+ }
```

Kode ini berfungsi sebagai **API DELETE** untuk menghapus data produk berdasarkan **ID** yang dikirim melalui parameter URL. Pada awal fungsi, sistem mengambil URL dari request dan mengekstrak nilai id menggunakan `searchParams`, kemudian mengonversinya menjadi tipe angka. Jika ID tidak ditemukan atau tidak valid, API akan mengembalikan response JSON dengan pesan *"ID produk tidak ditemukan"* dan status HTTP **400 (Bad Request)**. Apabila ID valid, sistem akan menghapus data produk dari database menggunakan perintah `prisma.produk.delete()` dengan kondisi berdasarkan ID tersebut. Implementasi ini memastikan proses penghapusan data berjalan terkontrol dan hanya dilakukan pada data yang valid.

4. Menambahkan response di DELETE

```
      where: { id },  
    });  
+    return NextResponse.json(  
+      { message: "Produk berhasil dihapus" },  
+      { status: 200 }  
+    );  
+  } catch (error) {  
+    return NextResponse.json(  
+      { message: "Gagal menghapus produk" },  
+      { status: 500 }  
+    );  
+  }  
+ }
```

Setelah proses penghapusan data produk berhasil dilakukan menggunakan Prisma, API akan mengembalikan response JSON dengan pesan *"Produk berhasil dihapus"* dan status HTTP **200 (OK)** sebagai penanda bahwa operasi berjalan sukses. Namun, jika terjadi kesalahan selama proses penghapusan, misalnya ID tidak ditemukan di database atau terjadi error pada server, maka blok `catch` akan dijalankan dan API mengirimkan pesan *"Gagal menghapus produk"* dengan status HTTP **500 (Internal Server Error)**. Mekanisme ini memastikan sistem memberikan informasi yang jelas terkait keberhasilan maupun kegagalan proses penghapusan data.

```
PROJECT-PWIR_JAVA  [26] [C] [B]
  backend
    src\app
      api
        admin
        users
        order
        [jd]
        TS routes
        TS routes.ts
        track
        [jd]
        TS routes
        TS routes.ts
        users
        favicon.ico
        # globals.css
        layout.tsx
        page.module.css
        page.tsx
        api
        @gltone
        # eslint.config.mjs
        TS middleware.ts
        # next.config.ts
        OUTLINE
        TIMELINE
        backend > src\app > api > order > TS routes > ...
        38 export async function POST(req: Request) {
        39
        37   const { name, phone, address, serviceType, weight, totalPrice } = data;
        38
        39   if (!name || !phone || !serviceType || !weight) {
        40     return NextResponse.json(
        41       { message: "Data wajib belum lengkap" },
        42       { status: 400 }
        43     );
        44   }
        45
        46   // cegah order duplikat: cek apakah ada order dengan name, phone, dan address yang sama dalam 1
        47   const existingOrder = await prisma.order.findFirst({
        48     where: {
        49       phone,
        50       serviceType,
        51       isPaid: false, // aturan: belum dibayar tidak boleh buat duplikat
        52     },
        53   });
        54
        55   if (existingOrder) {
        56     return NextResponse.json(
        57       { message: "Order dengan layanan dan nomor ini sudah ada" },
        58       { status: 400 }
        59     );
        60   }
        61
        62   const newOrder = await prisma.order.create({
        63     data: {
```

7. Error Handling

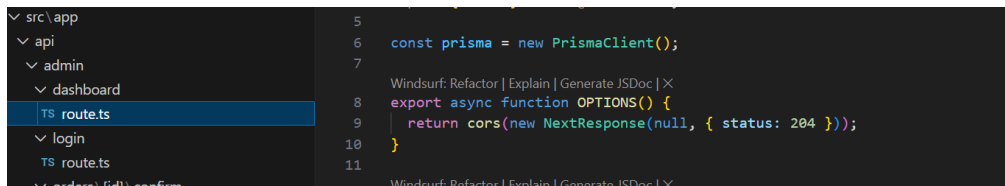
kode ini berfungsi untuk **mengambil dan memvalidasi data input** yang diterima oleh sistem. Data seperti name, phone, address, serviceType, weight, dan totalPrice diambil dari objek data menggunakan teknik *destructuring*. Selanjutnya, sistem melakukan pengecekan terhadap data wajib, yaitu name, phone, serviceType, dan weight. Jika salah satu dari data tersebut tidak diisi atau bernilai kosong, maka proses akan dihentikan dan API mengembalikan response JSON berisi pesan *“Data wajib belum lengkap”* dengan status HTTP **400 (Bad Request)**. Validasi ini bertujuan untuk memastikan bahwa data penting telah diisi sebelum diproses lebih lanjut.

```
code/backend/src/app/order/order/route.ts 14 2
```

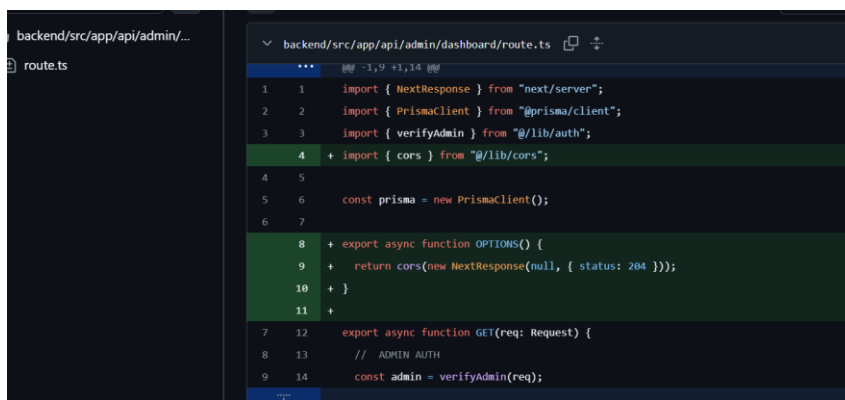
```

+
@@ -38,7 +38,7 @@ export async function POST(req: Request) {
38 38     if (!name || !phone || !serviceType || !weight) {
39 39         return NextResponse.json(
40 40             { message: "Data wajib belum lengkap" },
41 41             { status: 400 }
42 42         );
43 43     }
44 44
+
@@ -51,4 +51,16 @@ export async function POST(req: Request) {
51 51     weight,
52 52     totalPrice: totalPrice ?? 0, // default
53 53 },
54 54 - });
55 55 + });
56 56 +     return NextResponse.json(
57 57         { message: "Order dibuat", order: newOrder },
58 58         { status: 201 }
59 59     );
60 60 }
61 61
+
@@ -63,4 +63,4 @@ export async function GET(req: Request) {
63 63     const orders = await prisma.order.findMany();
64 64     return NextResponse.json(orders, { status: 200 });
65 65 }
66 66
+
@@ -68,4 +68,4 @@ export async function PUT(req: Request) {
68 68     const { id, name, phone, serviceType, weight, totalPrice } = req.json();
69 69     const order = await prisma.order.update({
70 70         where: { id },
71 71         data: { name, phone, serviceType, weight, totalPrice }
72 72     });
73 73     return NextResponse.json(order, { status: 200 });
74 74 }
75 75
+
@@ -77,4 +77,4 @@ export async function DELETE(req: Request) {
77 77     const { id } = req.json();
78 78     const order = await prisma.order.delete({
79 79         where: { id }
80 80     });
81 81     return NextResponse.json({ message: "Order deleted" }, { status: 200 });
82 82 }
83 83
+
@@ -84,4 +84,4 @@ export async function PATCH(req: Request) {
84 84     const { id, name, phone, serviceType, weight, totalPrice } = req.json();
85 85     const order = await prisma.order.update({
86 86         where: { id },
87 87         data: { name, phone, serviceType, weight, totalPrice }
88 88     });
89 89     return NextResponse.json(order, { status: 200 });
90 90 }
91 91
+
@@ -92,4 +92,4 @@ export async function OPTIONS(req: Request) {
92 92     return NextResponse.json({ message: "Options" }, { status: 200 });
93 93 }
94 94
+
@@ -95,4 +95,4 @@ export async function HEAD(req: Request) {
95 95     return NextResponse.json({ message: "Head" }, { status: 200 });
96 96 }
97 97
+
@@ -98,4 +98,4 @@ export async function TRACE(req: Request) {
98 98     return NextResponse.json({ message: "Trace" }, { status: 200 });
99 99 }
100 100
+
@@ -101,4 +101,4 @@ export async function POST(req: Request) {
101 101     return NextResponse.json({ message: "Post" }, { status: 200 });
102 102 }
103 103
+
@@ -104,4 +104,4 @@ export async function PUT(req: Request) {
104 104     return NextResponse.json({ message: "Put" }, { status: 200 });
105 105 }
106 106
+
@@ -107,4 +107,4 @@ export async function DELETE(req: Request) {
107 107     return NextResponse.json({ message: "Delete" }, { status: 200 });
108 108 }
109 109
+
@@ -110,4 +110,4 @@ export async function PATCH(req: Request) {
110 110     return NextResponse.json({ message: "Patch" }, { status: 200 });
111 111 }
112 112
+
@@ -113,4 +113,4 @@ export async function OPTIONS(req: Request) {
113 113     return NextResponse.json({ message: "Options" }, { status: 200 });
114 114 }
115 115
+
@@ -116,4 +116,4 @@ export async function HEAD(req: Request) {
116 116     return NextResponse.json({ message: "Head" }, { status: 200 });
117 117 }
118 118
+
@@ -119,4 +119,4 @@ export async function TRACE(req: Request) {
119 119     return NextResponse.json({ message: "Trace" }, { status: 200 });
120 120 }
121 121
+
@@ -122,4 +122,4 @@ export async function POST(req: Request) {
122 122     return NextResponse.json({ message: "Post" }, { status: 200 });
123 123 }
124 124
+
@@ -125,4 +125,4 @@ export async function PUT(req: Request) {
125 125     return NextResponse.json({ message: "Put" }, { status: 200 });
126 126 }
127 127
+
@@ -128,4 +128,4 @@ export async function DELETE(req: Request) {
128 128     return NextResponse.json({ message: "Delete" }, { status: 200 });
129 129 }
130 130
+
@@ -131,4 +131,4 @@ export async function PATCH(req: Request) {
131 131     return NextResponse.json({ message: "Patch" }, { status: 200 });
132 132 }
133 133
+
@@ -134,4 +134,4 @@ export async function OPTIONS(req: Request) {
134 134     return NextResponse.json({ message: "Options" }, { status: 200 });
135 135 }
136 136
+
@@ -137,4 +137,4 @@ export async function HEAD(req: Request) {
137 137     return NextResponse.json({ message: "Head" }, { status: 200 });
138 138 }
139 139
+
@@ -140,4 +140,4 @@ export async function TRACE(req: Request) {
140 140     return NextResponse.json({ message: "Trace" }, { status: 200 });
141 141 }
142 142
+
@@ -143,4 +143,4 @@ export async function POST(req: Request) {
143 143     return NextResponse.json({ message: "Post" }, { status: 200 });
144 144 }
145 145
+
@@ -146,4 +146,4 @@ export async function PUT(req: Request) {
146 146     return NextResponse.json({ message: "Put" }, { status: 200 });
147 147 }
148 148
+
@@ -149,4 +149,4 @@ export async function DELETE(req: Request) {
149 149     return NextResponse.json({ message: "Delete" }, { status: 200 });
150 150 }
151 151
+
@@ -152,4 +152,4 @@ export async function PATCH(req: Request) {
152 152     return NextResponse.json({ message: "Patch" }, { status: 200 });
153 153 }
154 154
+
@@ -155,4 +155,4 @@ export async function OPTIONS(req: Request) {
155 155     return NextResponse.json({ message: "Options" }, { status: 200 });
156 156 }
157 157
+
@@ -158,4 +158,4 @@ export async function HEAD(req: Request) {
158 158     return NextResponse.json({ message: "Head" }, { status: 200 });
159 159 }
160 160
+
@@ -161,4 +161,4 @@ export async function TRACE(req: Request) {
161 161     return NextResponse.json({ message: "Trace" }, { status: 200 });
162 162 }
163 163
+
@@ -164,4 +164,4 @@ export async function POST(req: Request) {
164 164     return NextResponse.json({ message: "Post" }, { status: 200 });
165 165 }
166 166
+
@@ -167,4 +167,4 @@ export async function PUT(req: Request) {
167 167     return NextResponse.json({ message: "Put" }, { status: 200 });
168 168 }
169 169
+
@@ -170,4 +170,4 @@ export async function DELETE(req: Request) {
170 170     return NextResponse.json({ message: "Delete" }, { status: 200 });
171 171 }
172 172
+
@@ -173,4 +173,4 @@ export async function PATCH(req: Request) {
173 173     return NextResponse.json({ message: "Patch" }, { status: 200 });
174 174 }
175 175
+
@@ -176,4 +176,4 @@ export async function OPTIONS(req: Request) {
176 176     return NextResponse.json({ message: "Options" }, { status: 200 });
177 177 }
178 178
+
@@ -179,4 +179,4 @@ export async function HEAD(req: Request) {
179 179     return NextResponse.json({ message: "Head" }, { status: 200 });
180 180 }
181 181
+
@@ -182,4 +182,4 @@ export async function TRACE(req: Request) {
182 182     return NextResponse.json({ message: "Trace" }, { status: 200 });
183 183 }
184 184
+
@@ -185,4 +185,4 @@ export async function POST(req: Request) {
185 185     return NextResponse.json({ message: "Post" }, { status: 200 });
186 186 }
187 186
+
@@ -188,4 +188,4 @@ export async function PUT(req: Request) {
188 188     return NextResponse.json({ message: "Put" }, { status: 200 });
189 188 }
190 188
+
@@ -191,4 +191,4 @@ export async function DELETE(req: Request) {
191 191     return NextResponse.json({ message: "Delete" }, { status: 200 });
192 191 }
193 191
+
@@ -194,4 +194,4 @@ export async function PATCH(req: Request) {
194 194     return NextResponse.json({ message: "Patch" }, { status: 200 });
195 194 }
196 194
+
@@ -197,4 +197,4 @@ export async function OPTIONS(req: Request) {
197 197     return NextResponse.json({ message: "Options" }, { status: 200 });
198 197 }
199 197
+
@@ -200,4 +200,4 @@ export async function HEAD(req: Request) {
200 200     return NextResponse.json({ message: "Head" }, { status: 200 });
201 200 }
202 200
+
@@ -203,4 +203,4 @@ export async function TRACE(req: Request) {
203 203     return NextResponse.json({ message: "Trace" }, { status: 200 });
204 203 }
205 203
+
@@ -206,4 +206,4 @@ export async function POST(req: Request) {
206 206     return NextResponse.json({ message: "Post" }, { status: 200 });
207 206 }
208 206
+
@@ -209,4 +209,4 @@ export async function PUT(req: Request) {
209 209     return NextResponse.json({ message: "Put" }, { status: 200 });
210 209 }
211 209
+
@@ -212,4 +212,4 @@ export async function DELETE(req: Request) {
212 212     return NextResponse.json({ message: "Delete" }, { status: 200 });
213 212 }
214 212
+
@@ -215,4 +215,4 @@ export async function PATCH(req: Request) {
215 215     return NextResponse.json({ message: "Patch" }, { status: 200 });
216 215 }
217 215
+
@@ -218,4 +218,4 @@ export async function OPTIONS(req: Request) {
218 218     return NextResponse.json({ message: "Options" }, { status: 200 });
219 218 }
220 218
+
@@ -221,4 +221,4 @@ export async function HEAD(req: Request) {
221 221     return NextResponse.json({ message: "Head" }, { status: 200 });
222 221 }
223 221
+
@@ -224,4 +224,4 @@ export async function TRACE(req: Request) {
224 224     return NextResponse.json({ message: "Trace" }, { status: 200 });
225 224 }
226 224
+
@@ -227,4 +227,4 @@ export async function POST(req: Request) {
227 227     return NextResponse.json({ message: "Post" }, { status: 200 });
228 227 }
229 227
+
@@ -230,4 +230,4 @@ export async function PUT(req: Request) {
230 230     return NextResponse.json({ message: "Put" }, { status: 200 });
231 230 }
232 230
+
@@ -233,4 +233,4 @@ export async function DELETE(req: Request) {
233 233
```

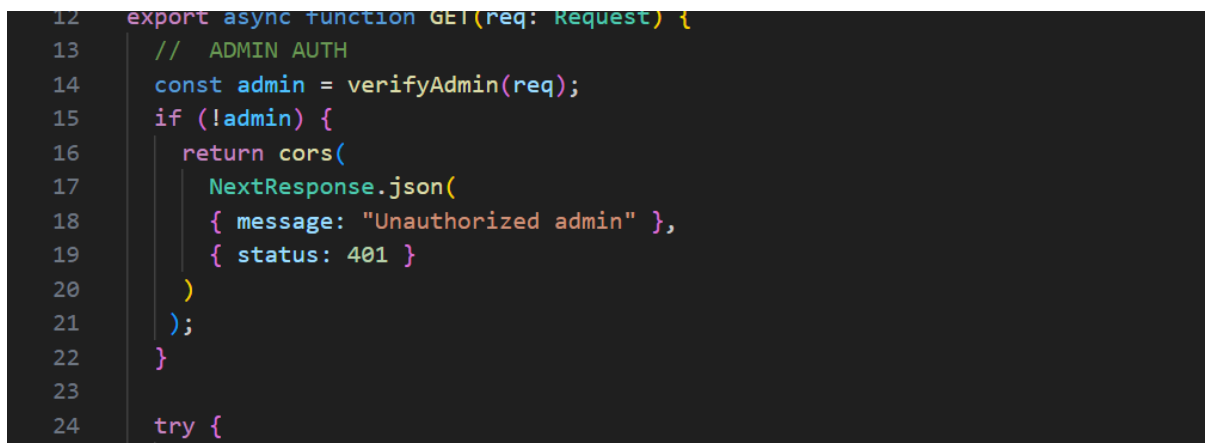
8. Update Admin



kode ini digunakan untuk **menginisialisasi koneksi database dan menangani permintaan CORS** pada API. PrismaClient diinisialisasi agar aplikasi dapat berkomunikasi dengan database menggunakan Prisma ORM. Sementara itu, fungsi OPTIONS dibuat untuk menangani *preflight request* dari browser yang berkaitan dengan CORS (Cross-Origin Resource Sharing). Fungsi ini mengembalikan response kosong dengan status HTTP **204 (No Content)** yang menandakan bahwa permintaan diperbolehkan, sehingga request dari domain lain dapat diproses dengan lancar oleh API.



9. Update admin AUTH



kode ini berfungsi untuk melakukan **autentikasi admin** sebelum suatu proses dijalankan. Fungsi `verifyAdmin(req)` digunakan untuk memeriksa apakah request yang masuk berasal dari pengguna dengan hak akses admin. Jika hasil verifikasi gagal atau admin tidak valid, maka API akan langsung menghentikan proses dan mengembalikan response JSON berisi pesan *"Unauthorized admin"* dengan status HTTP **401 (Unauthorized)** melalui mekanisme CORS. Validasi ini bertujuan untuk menjaga keamanan sistem dengan memastikan bahwa hanya admin yang berwenang yang dapat mengakses atau menjalankan fitur tertentu.

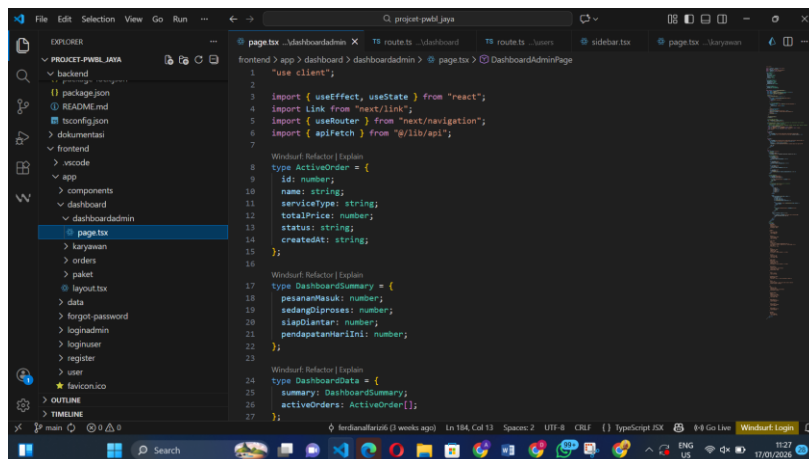
10. Update Middleware



```
4 // Fungsi pembantu CORS eksplisit di dalam middleware untuk memastikan header selalu ada.
5
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
```

Kode tersebut merupakan **fungsi pembantu untuk mengatur CORS (Cross-Origin Resource Sharing)** pada middleware agar setiap response API selalu memiliki header CORS. Fungsi setCorsHeaders menerima objek NextResponse, lalu menambahkan header Access-Control-Allow-Origin untuk mengizinkan akses dari domain `http://localhost:3001`, Access-Control-Allow-Methods untuk menentukan metode HTTP yang diperbolehkan (GET, POST, PUT, PATCH, DELETE, dan OPTIONS), serta Access-Control-Allow-Headers untuk mengizinkan penggunaan header Content-Type dan Authorization. Fungsi ini bertujuan mencegah pemblokiran request oleh browser dan memastikan komunikasi antara frontend dan backend berjalan dengan baik.

11. Update page dashboard admin



```
1 "use client";
2
3 import { useEffect, useState } from "react";
4 import Link from "next/link";
5 import { useRouter } from "next/navigation";
6 import { apiFetch } from "e/lib/api";
7
8 type ActiveOrder = {
9   id: number;
10  name: string;
11  serviceType: string;
12  totalPrice: number;
13  status: string;
14  createdAt: string;
15 };
16
17 type DashboardSummary = {
18   pesanMasuk: number;
19   sedangProses: number;
20   siapDiantar: number;
21   pendapatanIni: number;
22 };
23
24 type DashboardData = {
25   summary: DashboardSummary;
26   activeOrders: ActiveOrder[];
27 };
28
```

Jadi dashboard admin tidak bisa di akses begitu saja, tetapi ada fitur log yang mengharuskan admin login terlebih dahulu untuk mengakses dashboard tersebut.

12. Update tampilan order pada page admin agar data yang masuk di dapatkan dari page user nya

```
15
16 /* FORMAT RUPIAH */
Windsurf: Refactor | Explain | X
17 const formatRupiah = (value: number) =>
18   "Rp " + value.toLocaleString("id-ID");
19
Windsurf: Refactor | Explain | Generate JSDoc | X
20 export default function OrdersPage() {
21   const router = useRouter();
22   const [orders, setOrders] = useState<Order[]>([]);
23   const [loading, setLoading] = useState(true);
24   const [error, setError] = useState("");
25
26   /* FETCH ORDERS */
27   useEffect(() => {
Windsurf: Refactor | Explain | Generate JSDoc | X
28     const fetchOrders = async () => {
29       try {
30         const token = localStorage.getItem("admin_token");
31         if (!token) {
32           router.push("/loginadmin");
33           return;
34         }
35
36         const data = await apiFetch("/api/admin/orders", {
37           headers: {
38             Authorization: `Bearer ${token}`,
39           },
40         });

```

13. Login User Fix

```
try {
  const data = await apiFetch("/api/users/login", {
    method: "POST",
    body: JSON.stringify({ email, password }),
  });

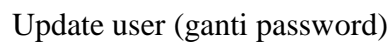
  console.log("LOGIN SUCCESS:", data);

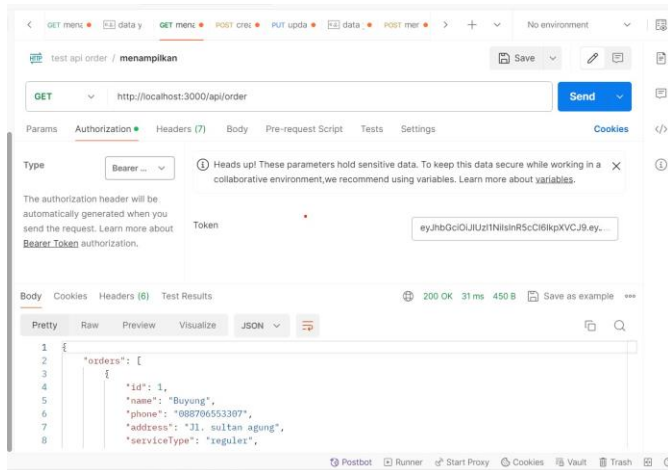
  if (data.token) {
    localStorage.setItem("user_token", data.token);
    router.push("/user");
  } else {
    throw new Error("Token tidak diterima dari server");
  }

} catch (err: any) {
  console.error("LOGIN ERROR:", err);
  setError(err.message || "Gagal terhubung ke server");
} finally {
  setLoading(false);
}
};

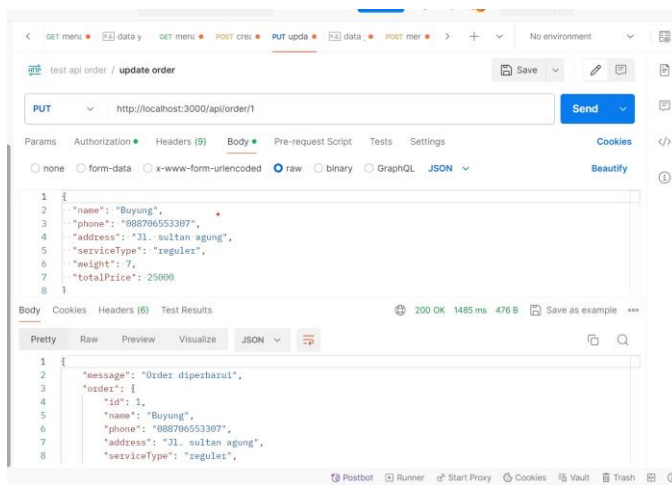
return (
```

digunakan untuk **menangani proses login yang berhasil pada sisi client**. Ketika server mengembalikan data login, sistem menampilkan pesan *“LOGIN SUCCESS”* beserta data ke console untuk keperluan debugging. Selanjutnya, kode melakukan pengecekan apakah server mengirimkan token. Jika token tersedia, token tersebut disimpan ke dalam localStorage dengan nama token dan user_token sebagai bukti autentikasi pengguna, kemudian pengguna diarahkan ke halaman /user. Namun, jika token tidak diterima dari server, sistem akan melempar error dengan pesan *“Token tidak diterima dari server”* untuk menandakan adanya kegagalan pada proses autentikasi.

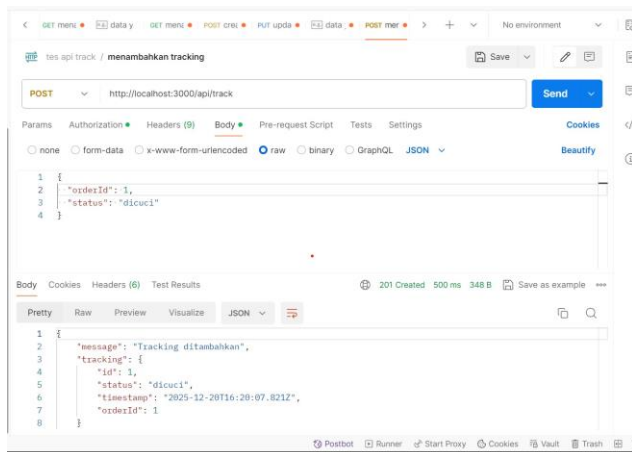




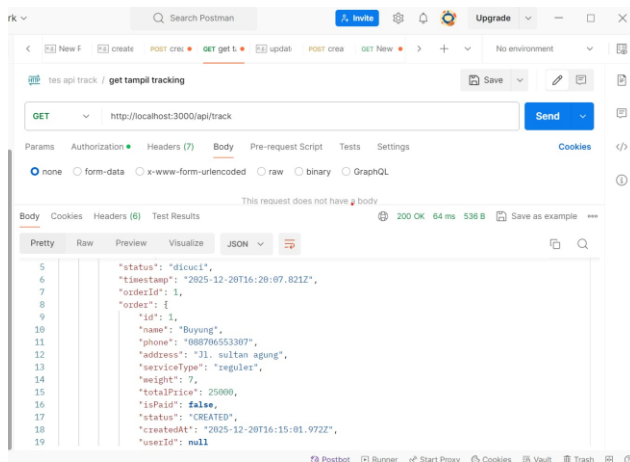
Lihat order



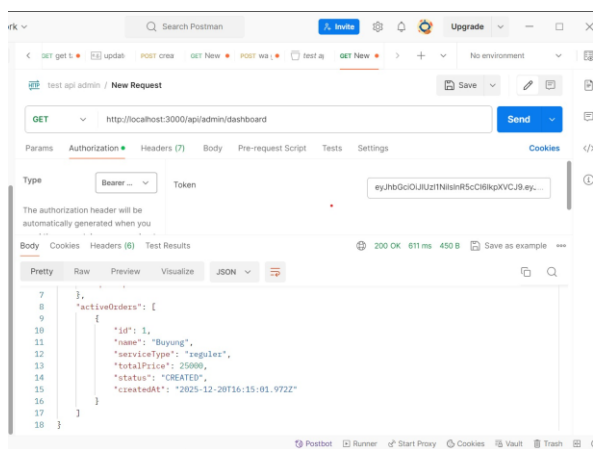
Update order before payment



Add tracking pada ordered



Tampil tracking



Admin lihat pesanan masuk