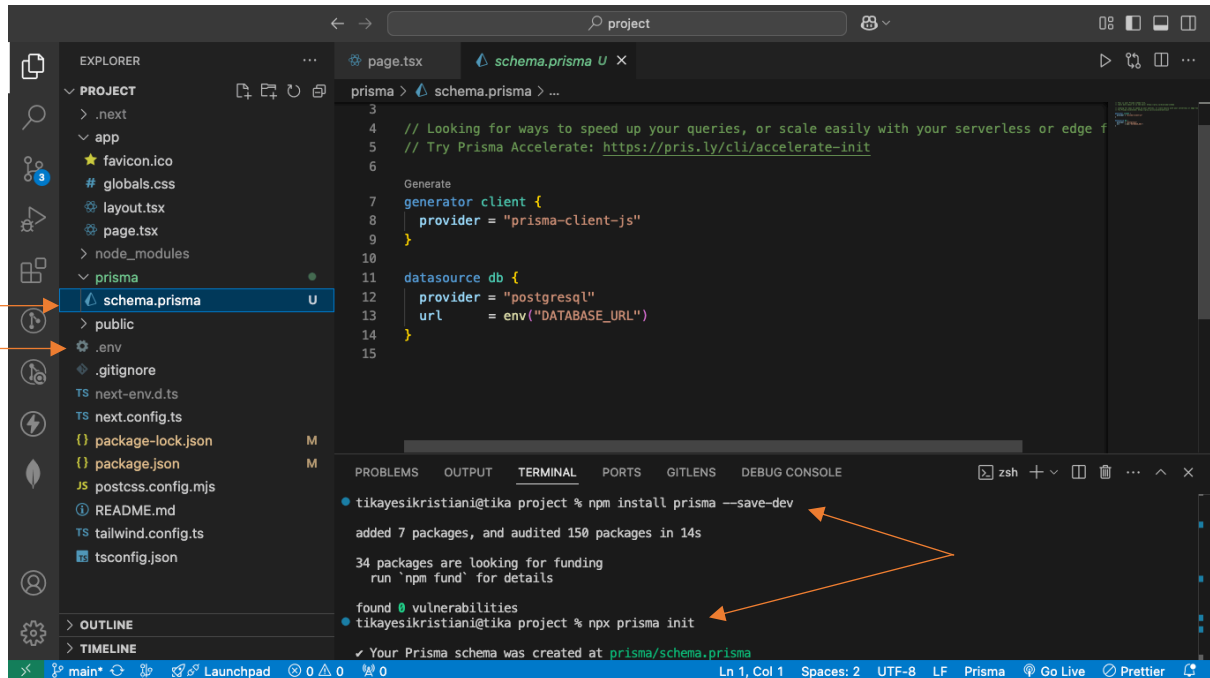
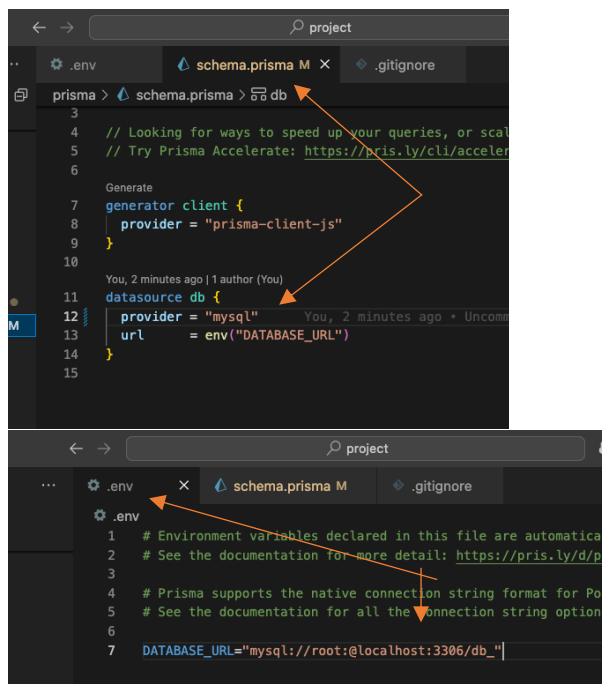


Nama : Deo Cahyo Anggoro  
NPM : 22312168

## Dokumentasi Project



Menginstall prisma dengan cara mengetik npm install prisma --save-dev di terminal dan npx prisma init unntuk membuat file schema.prisma dan .env



Karena Saya menggunakan mysql ganti postgresql ke mysql di file schema.prisma dan .env untuk menghubungkan ke database mysql  
Penjelasan di file .env root adalah usernya dan localhost:3306 adalah portnya

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  GITLENS  DEBUG CONSOLE  zsh + - [ ] [ ] ... ^ X

found 0 vulnerabilities
tikayesikristiani@tika project % npm i -D daisyui@latest

added 4 packages, and audited 154 packages in 4s

35 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
tikayesikristiani@tika project %
```

Menginstall DaisyUI dengan cara mengetik `npm i -D daisyui@latest` di terminal, `@latest` maksudnya adalah versi terbaru dari daisyui

```
ts tailwind.config.ts > [ ] default
3   export default {
15     // ...
16   },
17   plugins: [
18     require('daisyui'),
19   ],
```

Lalu masukan `require('daisyui')`, kedalam plugins di file `tailwind.config.ts` untuk menambahkan daisyui

```
layout.tsx M •
app > layout.tsx > RootLayout
6   export default function RootLayout({
11     return [
12       <html lang="en" data-theme="lemonade">
13         <body>
14           { /* Area Header */ }
15           <header></header>
16
17           { /* Area Conten */ }
18           <section>
19             {children}
20           </section>
21
22           { /* Area Footer */ }
23           <footer></footer>
24         </body>
25       </html>
26     ]
27   }
You, 3 minutes ago • Uncommitted changes
```

Menambahkan Header, Section, dan Footer bertujuan untuk memisahkan area header, konten, dan footer untuk menciptakan struktur yang lebih terorganisir dan semantic

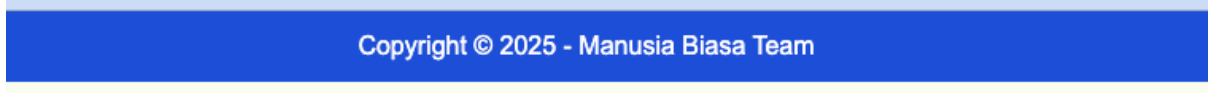
```
{/* Area Footer */}  
<footer>  
Copyright &copy; 2025 - Manusia Biasa Team  
</footer>
```

Menambahkan text copyright di footer, &copy; adalah logo copyright

```
> layout.tsx > ...  
You, 1 minute ago | 3 authors (You and others)  
// import file "globals.css"  
import "./globals.css"
```

Mengimport globals.css di layout.tsx

```
{/* Area Footer */}  
<footer className="flex justify-center bg-blue-700 text-white py-2.5">  
Copyright &copy; 2025 - Manusia Biasa Team  
</footer>
```



Memperbagus tampilan footer

`flex justify-center`

agar textnya berada ditengah

`bg-blue-700`

membuat background berwarna biru

`text-white`

membuat text berwarna putih

`py-2.5`

memberikan padding sebesar 10 px

```

<html lang="en" data-theme="dark">
  <body>
    { /* Area Header */ }
    <header>
      <Link href="/">Ecommerce</Link>
      <nav>
        <Link href="/">Home</Link>
        <Link href="/">About</Link>
        <Link href="/">Products</Link>
        <Link href="/">Contact</Link>
      </nav>
    </header>
  </body>
</html>

```

Menambahkan Link yang berguna untuk tombol navigasi

```

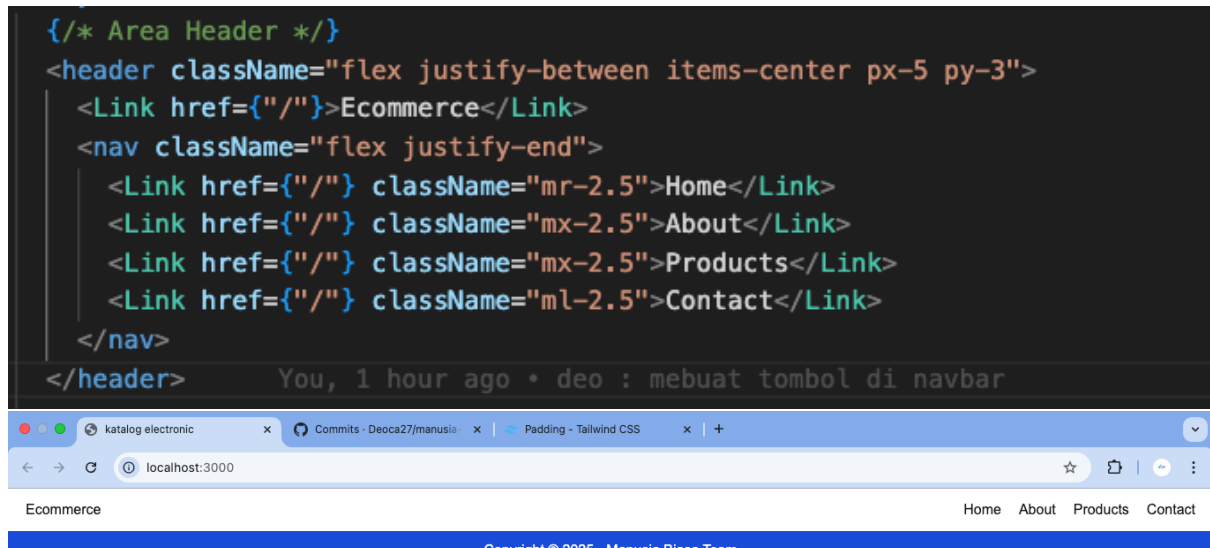
{ /* Area Header */ }
<header>
  <Link href="/">Ecommerce</Link>
  <nav className="flex justify-end">
    <Link href="/" className="mr-2.5">Home</Link>
    <Link href="/" className="mx-2.5">About</Link>
    <Link href="/" className="mx-2.5">Products</Link>
    <Link href="/" className="ml-2.5 mr-5">Contact</Link>
  </nav>
</header>

```

Menambahkan flex justify-end agar tombol berada di sebelah kanan

Home About Products Contact

Dan menambahkan margin agar tombolnya memiliki jarak  
 mr untuk margin kanan  
 ml untuk margin kiri  
 dan 2.5 adalah ukuran marginya sebesar 10px



`flex justify-between items-center px-5 py-3`

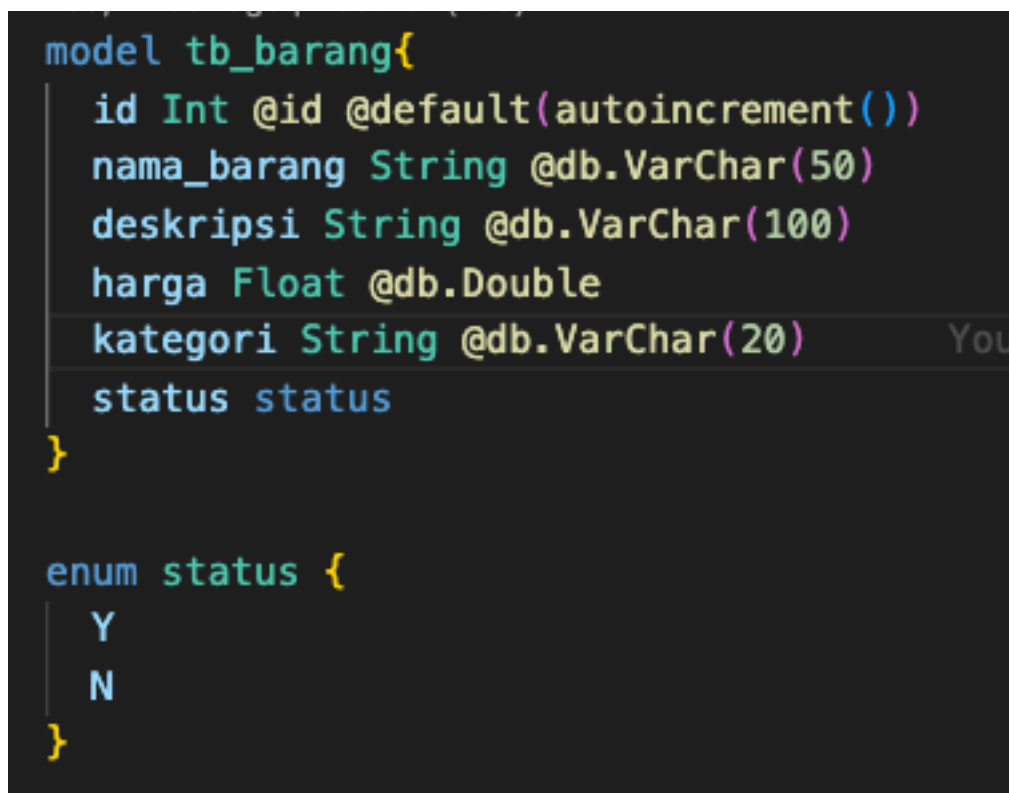
Berguna agar tata letaknya sejajar

`flex justify-between` Mengatur jarak antar flex items secara horizontal sehingga item pertama berada di sisi kiri, item terakhir di sisi kanan

`items-center` Menyelaraskan (align) flex items secara vertikal di tengah-tengah flex container

`px-5` untuk mengatur padding kiri dan kanan sebesar 20px

`py-3` untuk mengatur padding atas dan bawah sebesar 12px



Menambah tabel barang di schema.prisma

Int adalah tipe data integer untuk bilangan bulat

@id menandai bahwa itu adalah primary key

@default(autoincrement()) untuk membuat nilai secara otomatis

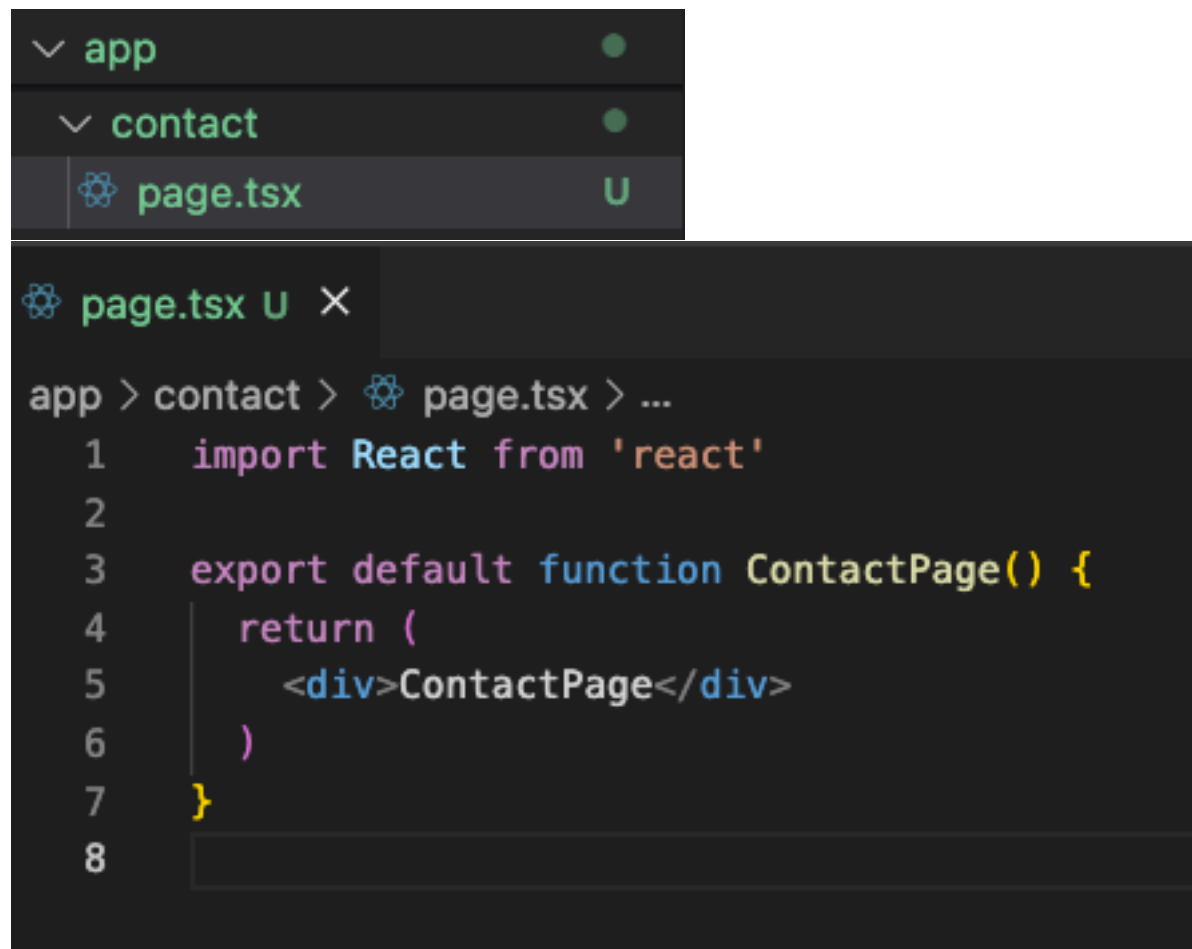
String adalah tipe data untuk teks

@db.VarChar(50) untuk tipe data ini di database adalah varchar dan untuk nilainya adalah panjang maksimum karakter

Float adalah tipe data untuk bilangan pecahan

@db.Double untuk tipe data di database adalah double

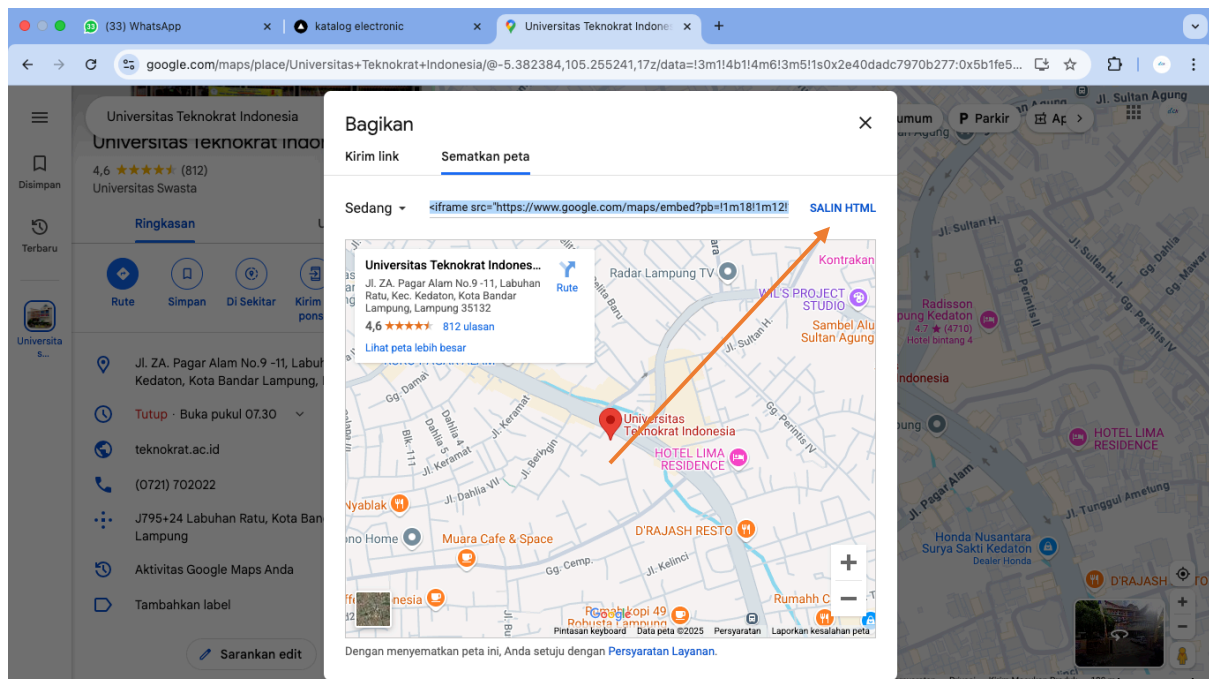
Untuk melakukan migrate ketikkan npx prisma migrate dev di terminal lalu beri nama lalu prisma akan membuat migration file



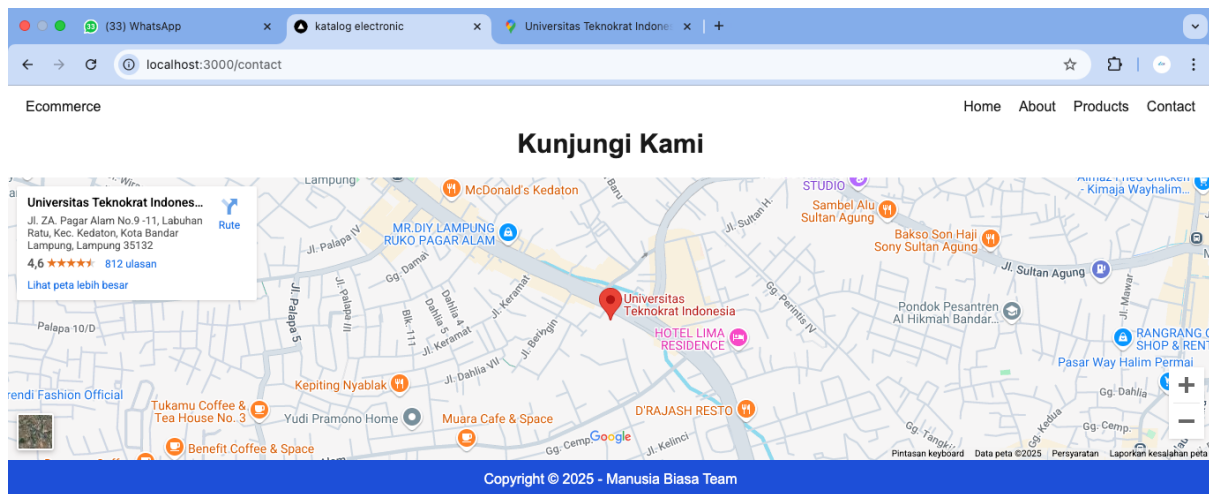
The image shows a code editor interface. At the top, there is a file explorer with a tree view showing a folder structure: 'app' (expanded), 'contact' (expanded), and 'page.tsx'. The 'page.tsx' file is selected. Below the file explorer, the code editor shows the content of 'page.tsx'. The code is as follows:

```
app > contact > page.tsx > ...
1  import React from 'react'
2
3  export default function ContactPage() {
4    return (
5      <div>ContactPage</div>
6    )
7  }
8
```

Buat folder baru didalam folder app dengan nama contact lalu buat file dengan nama page.tsx  
Lalu ketikkan rfc lalu enter dan ganti page menjadi ContactPage



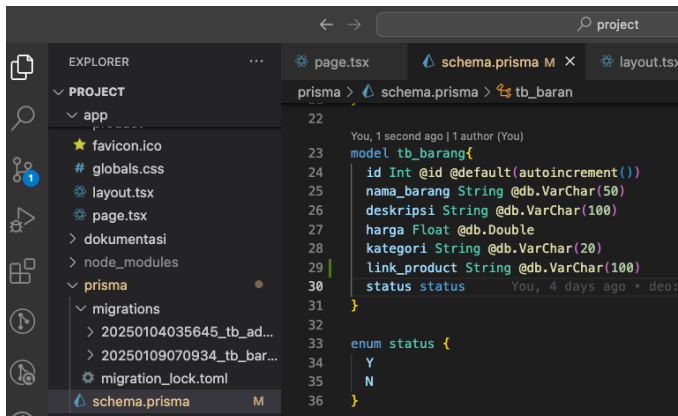
Untuk menambah map dapat langsung mengcopynya dari google map  
 Lalu atur ukuran mapnya menggunakan tailwind dengan cara mengganti defaultnya width dan height dengan className="" w dan h untuk width dan height  
 Saya menggunakan w-full h-80 untuk width full dari kanan ke kiri atau 100% dan height 20rem atau 320px



Lalu menambah text judul menggunakan <h1></h1> dan diatur menggunakan tailwind

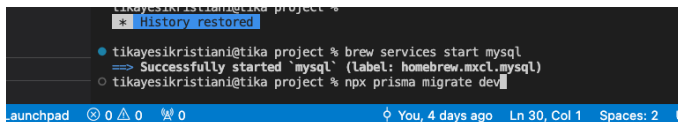
```
<h1 className="text-3xl font-bold mb-5 text-center">Kunjungi Kami</h1>
```

text-3xl untuk mengatur font size sebesar 30px dan line height sebesar 36px  
 font bold untuk menebalkan textnya  
 mb-5 untuk memberikan margin dibawah sebesar 20px  
 text center agar text berada ditengah

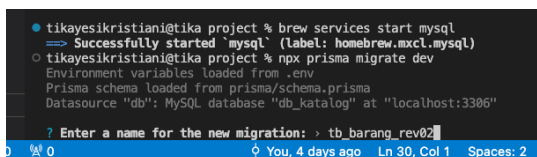


Untuk menambah field di prisma pertama masuk ke file schema.prisma lalu tambahkan field ke dalam tabel

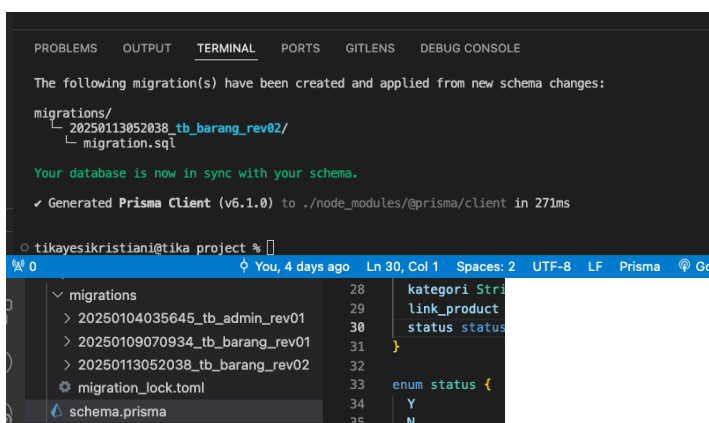
Disini saya menambahkan link\_product dengan tipe data string varchar sebanyak dengan max 100 huruf



Setelah itu lakukan migrate dengan cara mengetikan npx prisma migrate dev

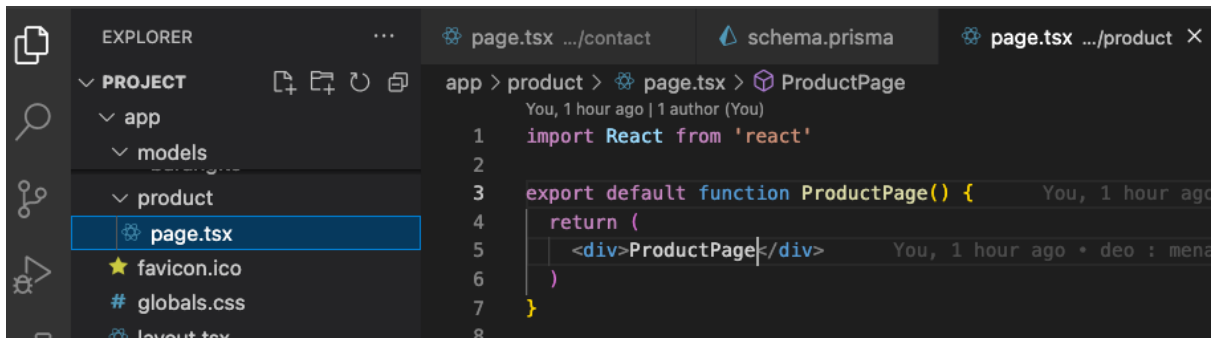


Lalu beri nama migration datanya, disini saya menamainya tb\_barang\_rev02 karena ini adalah revisi ke 2



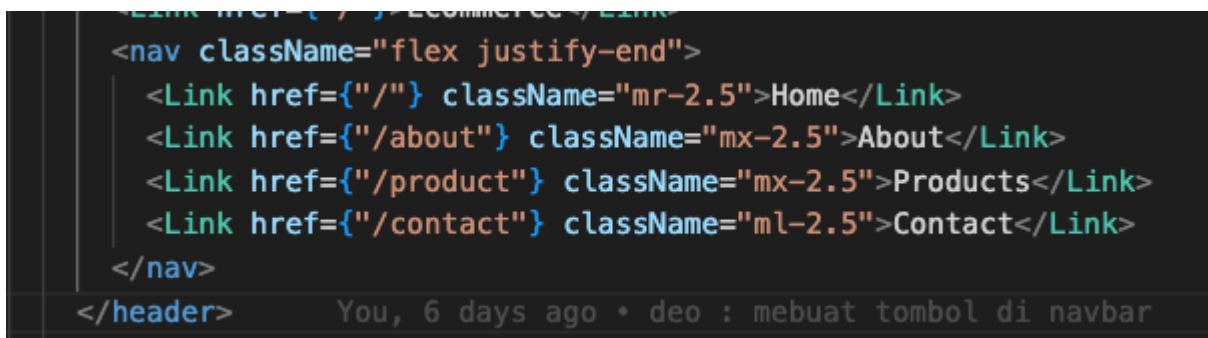
Setelah selesai maka akan muncul file migration yang dibuat tadi dan fieldnya akan masuk ke database



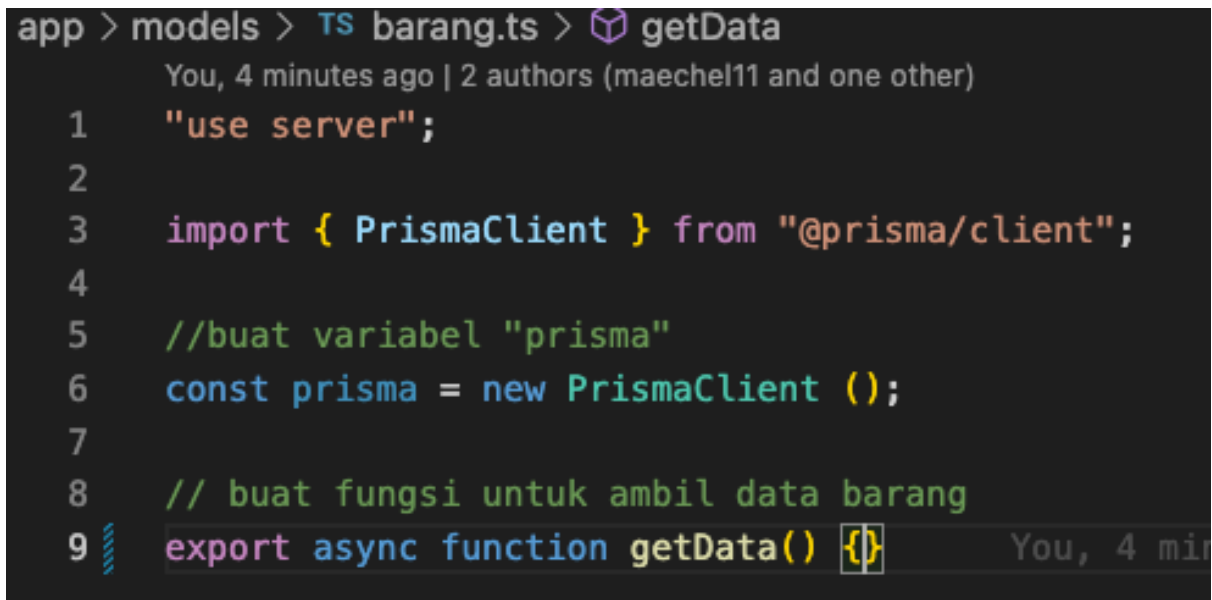


Buat folder baru dengan nama product dan file page.tsx untuk nantinya dijadikan halaman product

Lalu ketikkan rfc untuk mempermudah membuat export default function



Lalu tambahkan /product di rootlayout agar navigasinya mengarah ke file page yang ada di dalam folder product



Buka file barang.ts lalu membuat export async function getData() {}

Export digunakan agar file ini dapat di impor di file lain

Async menunjukkan bahwa fungsi ini bersifat asynchronous

function getData ini digunakan untuk mengambil data dari database

```

app > models > TS barang.ts > [🔗] getData > [🔗] barang
You, 4 minutes ago | 2 authors (You and one other)
1  "use server";
2
3  import { PrismaClient } from "@prisma/client";
4
5  //buat variabel "prisma"
6  const prisma = new PrismaClient ();
7
8  // buat fungsi untuk ambil data barang
9  export async function getData() {
10   // buat variabel "barang"
11   const barang = await prisma.tb_barang.findMany({});
12 }

```


Membuat variable barang `const barang = await prisma.tb_barang.findMany({});` digunakan untuk mengambil semua data dari tabel `tb_barang` dalam database menggunakan Prisma

```

app > models > TS barang.ts > [🔗] getData > [🔗] barang > [🔗] where
You, 12 minutes ago | 2 authors (You and one other)
1  "use server";
2
3  import { PrismaClient } from "@prisma/client";
4
5  //buat variabel "prisma"
6  const prisma = new PrismaClient ();
7
8  // buat fungsi untuk ambil data barang
9  export async function getData() {
10   // buat variabel "barang"
11   const barang = await prisma.tb_barang.findMany({
12     where: {
13       status: "Y",
14     },
15   });
16 }

```

Where: {status:"Y"},} ini maksudnya data yang akan ditampilkan hanya data yang enum statusnya adalah Y

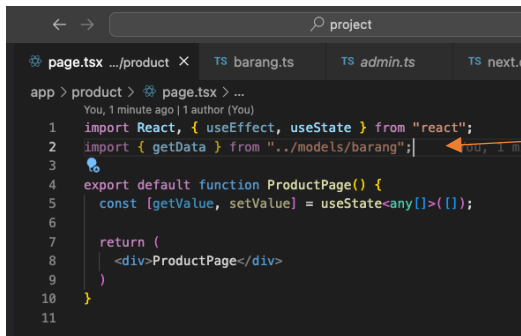
```
app > models > TS barang.ts >  getData
// const prisma = new PrismaClient()

6   const prisma = new PrismaClient ();
7
8   // buat fungsi untuk ambil data barang
9   export async function getData() {
10      // buat variable "barang"
11      const barang = await prisma.tb_barang.findMany({
12         where: {
13            status: "Y",
14         },
15      });
16
17      return barang;
18   }
```

Return barang; digunakan untuk mengembalikan data yang telah diambil dari database  
Dan kenapa harus dikembalikan? Karena data ini nantinya akan dibuat untuk ditampilkan

```
← → project
page.tsx .../product • TS barang.ts TS admin.ts TS next.co
app > product > page.tsx > ...
You, 5 minutes ago | 1 author (You)
1   import React, { useEffect, useState } from "react";
2
3   export default function ProductPage() {
4     const [getValue, setValue] = useState<any[]>([]);
5
6     return (
7       <div>ProductPage</div>
8     )
9   }
10
```

Bagian kode ini adalah bagian dari komponen React yang menggunakan Hooks (useState dan useEffect)



```
1 import React, { useEffect, useState } from "react";
2 import { getData } from "../models/barang";
3
4 export default function ProductPage() {
5   const [getValue, setValue] = useState<any[]>([]);
6
7   return (
8     <div>ProductPage</div>
9   )
10 }
11
```

Mengimport get data dari file barang.ts yang ada didalam folder models



```
2 import React, { useEffect, useState } from "react";
3 import { getData } from "../models/barang";
4
5 export default function ProductPage() {
6   //buat hook dengan "use state"
7   const [getValue, setValue] = useState<any[]>([]);
8
9   // buat fungsi untuk panggil "getData"
10  async function fetchData() {
11    const data = await getData();
12    setValue(data);
13  }
14
```

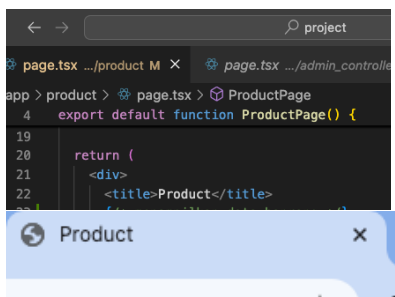
Bagian ini adalah fungsi asynchronous yang mengambil data menggunakan getData(), menunggu hasilnya dengan await, dan kemudian memperbarui state getValue menggunakan setValue(data) dengan data yang diperoleh.



```
// buat fungsi untuk panggil "getData"
async function fetchData() {
  const data = await getData();
  setValue(data);
}

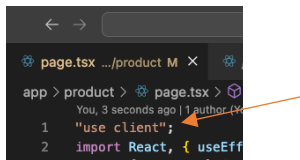
// hook dengan "use effect"
useEffect(() => {
  // panggil fungsi "getData"
  fetchData();
}, []);
```

Bagian ini memanggil fungsi fetchData sekali saat komponen pertama kali dirender, karena array dependency kosong ([]), yang berarti efek hanya dijalankan sekali.



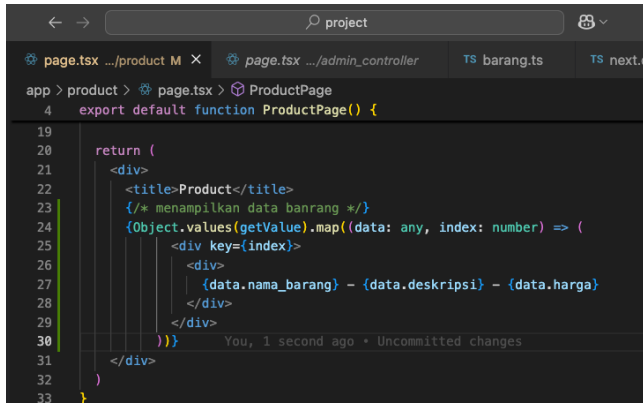
```
4 export default function ProductPage() {
19
20   return (
21     <div>
22       <title>Product</title>
```

Menambahkan title agar saat di menu product di bagian title web tertulis title



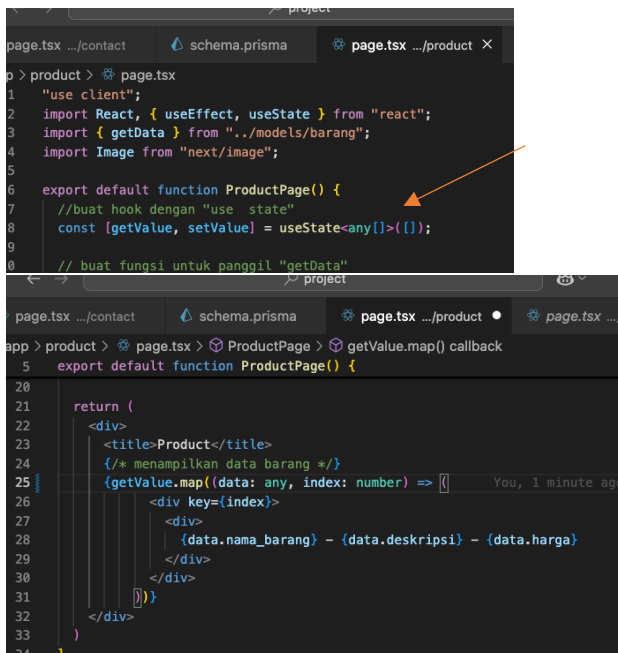
```
1 "use client";
2 import React, { useEffect
```

"use client" memastikan komponen dirender di browser, bukan di server, biasanya untuk menggunakan fitur React seperti useState atau useEffect.



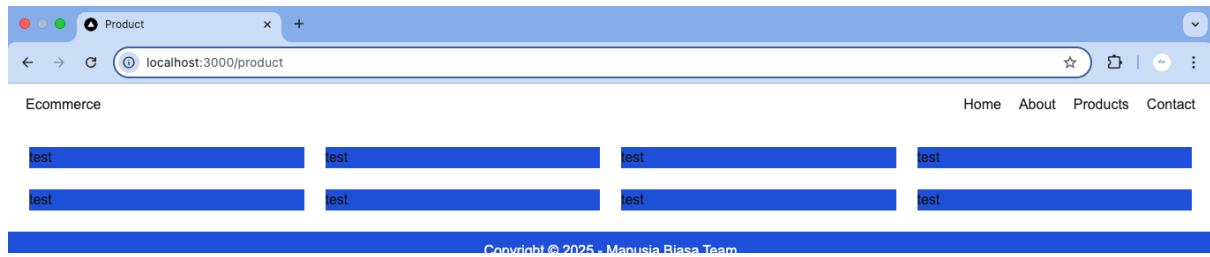
```
export default function ProductPage() {
  return (
    <div>
      <title>Product</title>
      { /* menampilkan data barang */ }
      { Object.values(getValue).map((data: any, index: number) => (
        <div key={index}>
          <div>
            {data.nama_barang} - {data.deskripsi} - {data.harga}
          </div>
        </div>
      )) }
    </div>
  )
}
```

Kode ini digunakan untuk menampilkan data barang dari state getValue. Data diakses dengan Object.values(getValue) lalu di-loop menggunakan map, menampilkan nama\_barang, deskripsi, dan harga setiap item dalam elemen <div>.



```
1 "use client";
2 import React, { useEffect, useState } from "react";
3 import { getData } from "../models/barang";
4 import Image from "next/image";
5
6 export default function ProductPage() {
7   //buat hook dengan "use state"
8   const [getValue, setValue] = useState<any[]>([]);
9
10  // buat fungsi untuk panggil "getData"
11
12  // ...
13
14  return (
15    <div>
16      <title>Product</title>
17      { /* menampilkan data barang */ }
18      { getValue.map((data: any, index: number) => (
19        <div key={index}>
20          <div>
21            {data.nama_barang} - {data.deskripsi} - {data.harga}
22          </div>
23        </div>
24      )) }
25    </div>
26  )
27 }
```

Kode ini tidak memerlukan Object.values() karena getValue sudah diinisialisasi sebagai array dengan useState<any[]>([])  
Jika Object.values() tetap ada meskipun getValue sudah berupa array, itu tidak akan menyebabkan error tetapi menjadi redundan dan tidak efisien



Membuat grid menggunakan tailwind.css

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 p-6">
```

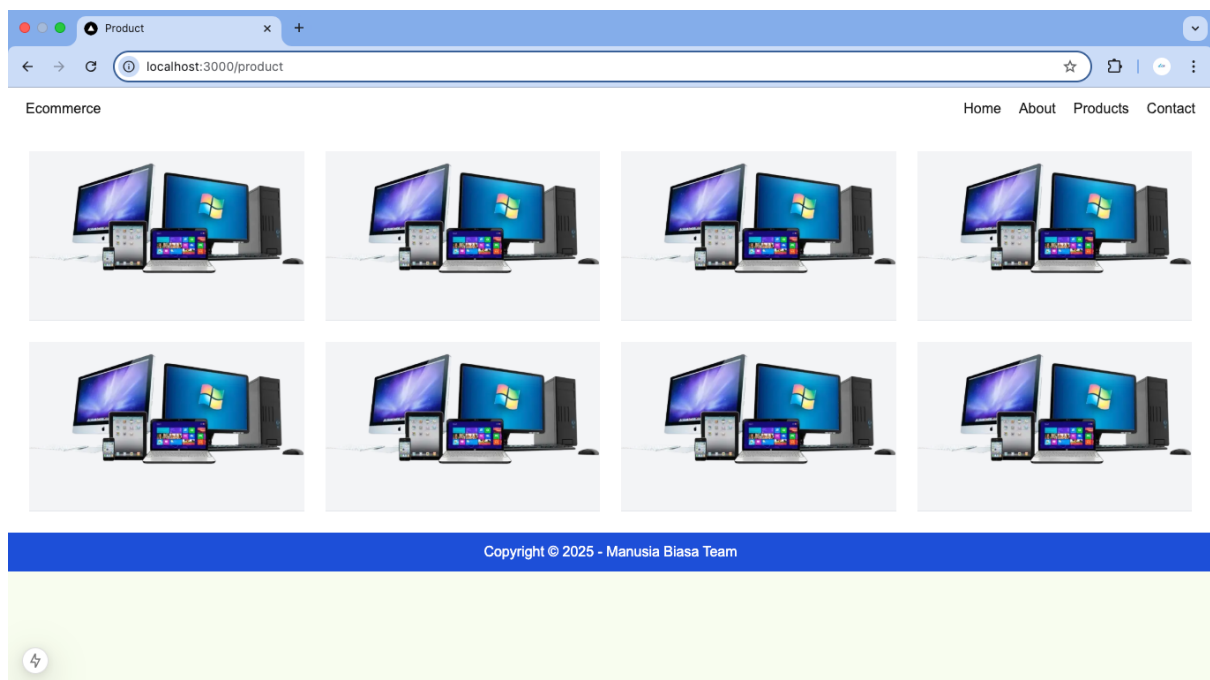
**grid-cols-1:** Mengatur grid untuk memiliki 1 kolom pada layar kecil (default).

**md:grid-cols-2:** Pada layar berukuran medium (tablet) atau lebih besar, grid akan memiliki 2 kolom.

**lg:grid-cols-4:** Pada layar berukuran large (desktop) atau lebih besar, grid akan memiliki 4 kolom.

**gap-6:** Mengatur jarak antara elemen-elemen grid (baik baris maupun kolom) sebesar 1.5rem (atau 24px).

**p-6:** Memberikan padding di seluruh sisi container sebesar 1.5rem (atau 24px). I



Buat tampilan untuk display

```
<div className="w-full h-48 bg-gray-100 flex items-center justify-center border-b border-gray-200">
```

```
<Image
```

```
src="/images/foto.png"
```

```
alt={data.nama_barang}
```

```
width={500}
```

```
height={300}
```

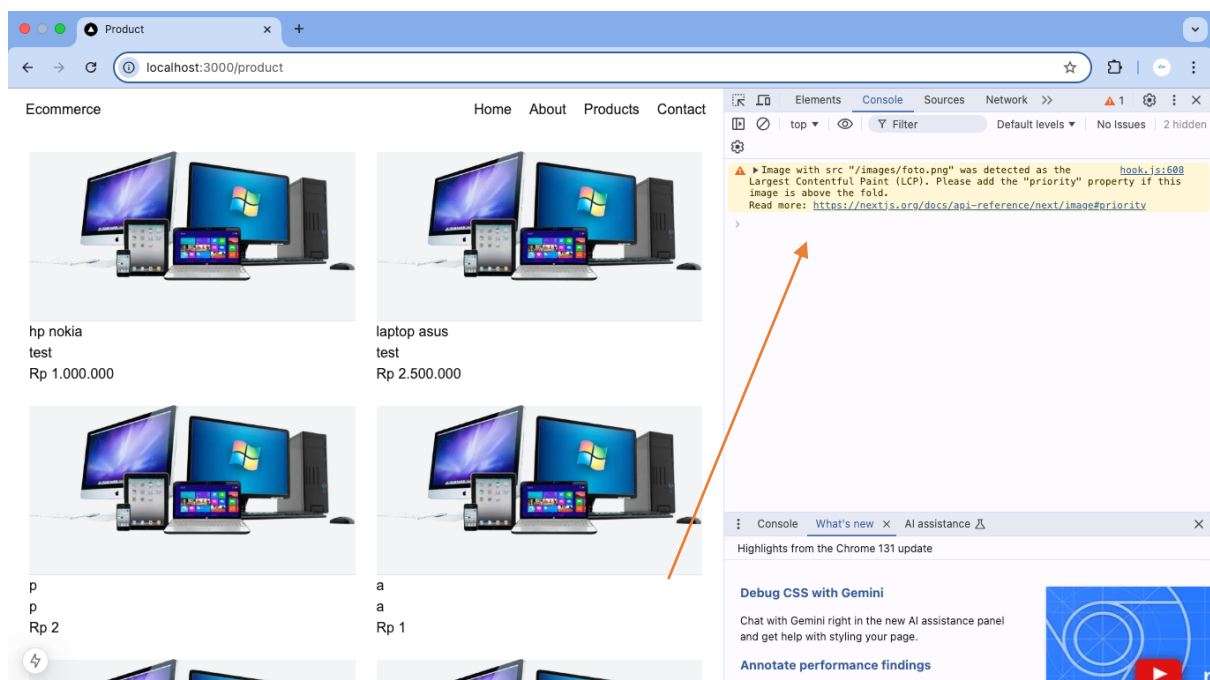
```


```


Disini menggunakan tailwind  
w-full = width: 100%  
h-48 = height: 12rem / 192px  
bg-gray-100 = background-color: (abu-abu terang)  
flex = display: flex  
items-center = align-items: center  
justify-center = justify-content: center  
border-b = border-bottom: 1px solid  
border-gray-200 = border-color: (abu-abu terang)

```
<div>
  { /* Nama Barang */ }
  <h2>
    {data.nama_barang}
  </h2>
  { /* Deskripsi */ }
  <p>
    {data.deskripsi}
  </p>
  { /* Harga */ }
  <p>
    Rp {data.harga.toLocaleString("id-ID")}
  </p>
</div>
```

Menampilkan nama barang, deskripsi, dan harga yang diambil dari database  
Untuk bagian harga menggunakan toLocaleString("id-ID"),  
toLocaleString("id-ID") digunakan untuk format angka agar sesuai dengan format lokal  
Indonesia  
misalnya Rp 1000000000 menjadi Rp 1.000.000.000



Tambahkan Priority pada komponen <Image>

```
<Image  
src="/images/foto.png"  
alt={data.nama_barang}  
width={500}  
height={300}  
priority   
>
```

Di Next.js, atribut `priority` pada komponen `<Image>` digunakan untuk memberi tahu Next.js agar memprioritaskan pemuatan gambar tersebut.