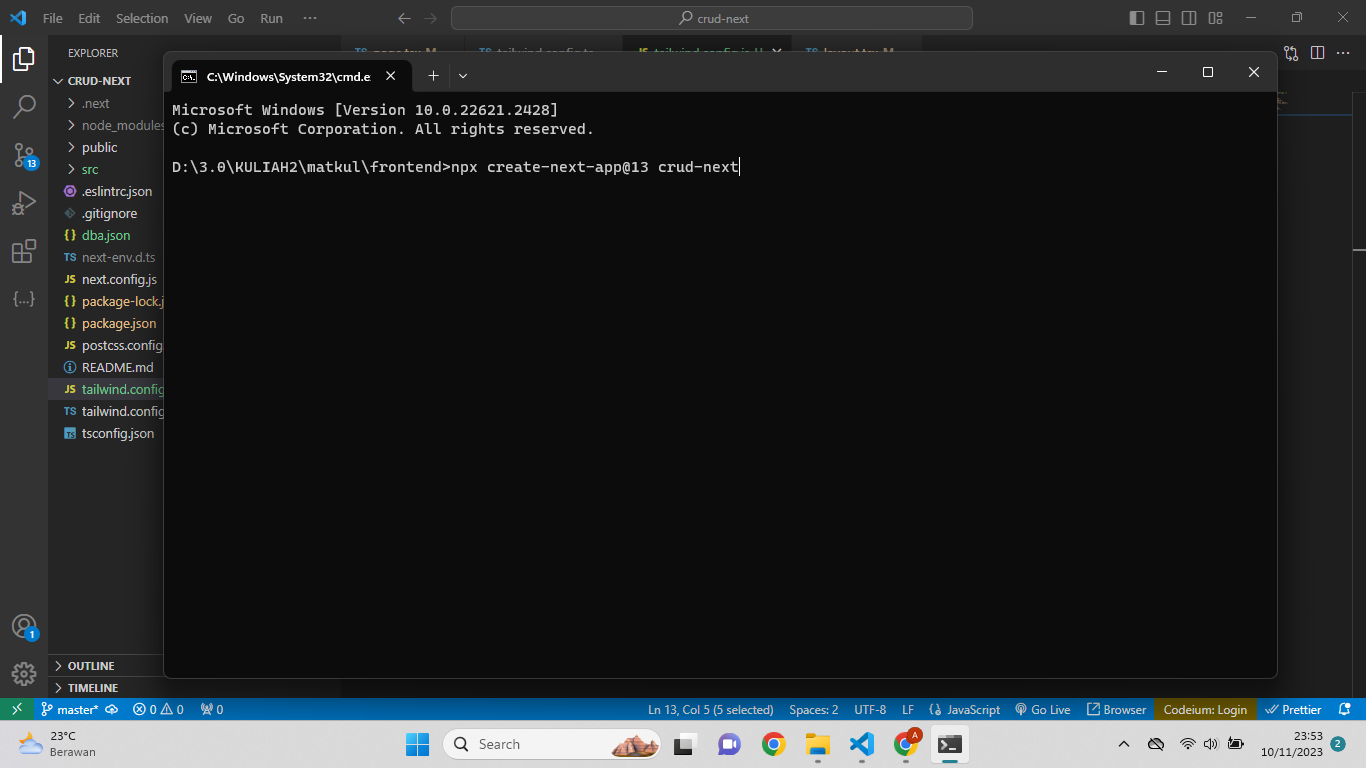
**Next JS CRUD**

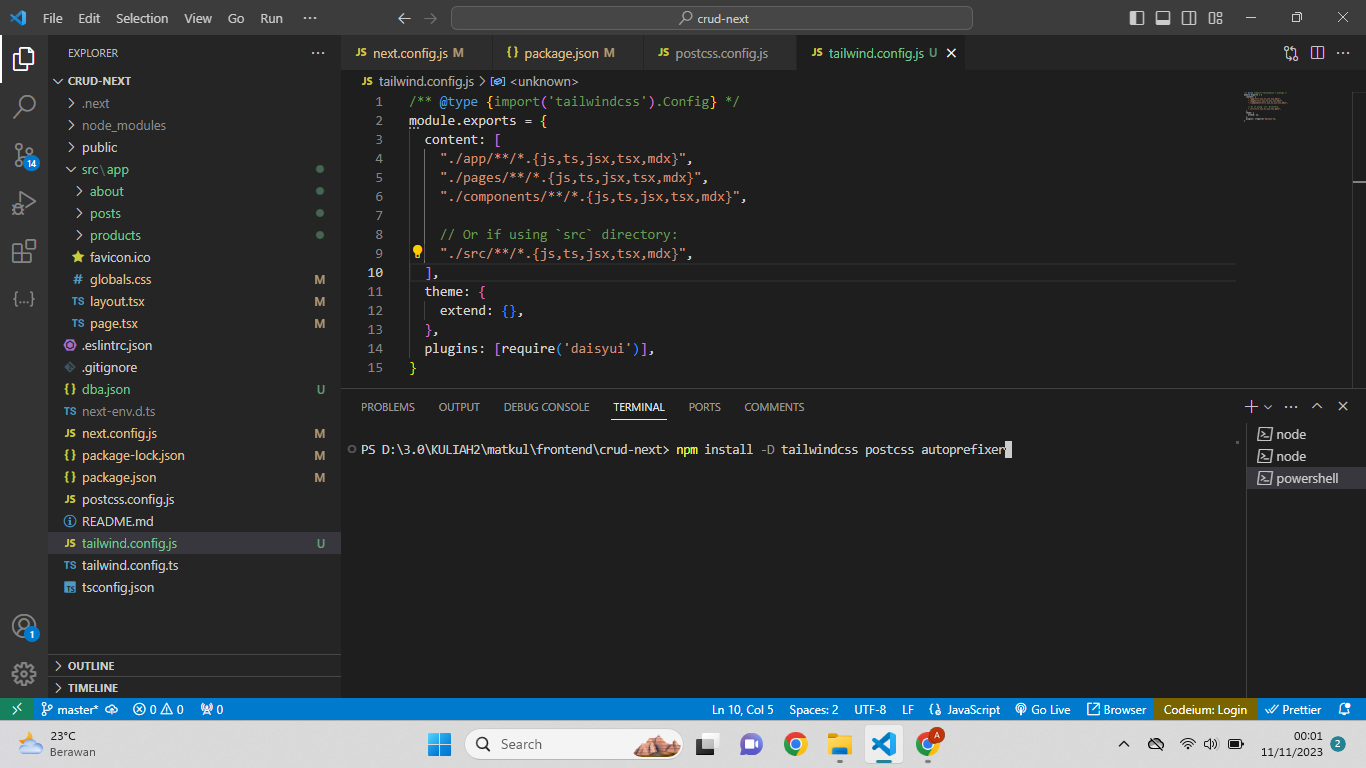
*Silva Tria Alfares / PPLSB / 6*

Step by step membuat CRUD dengan next JS :

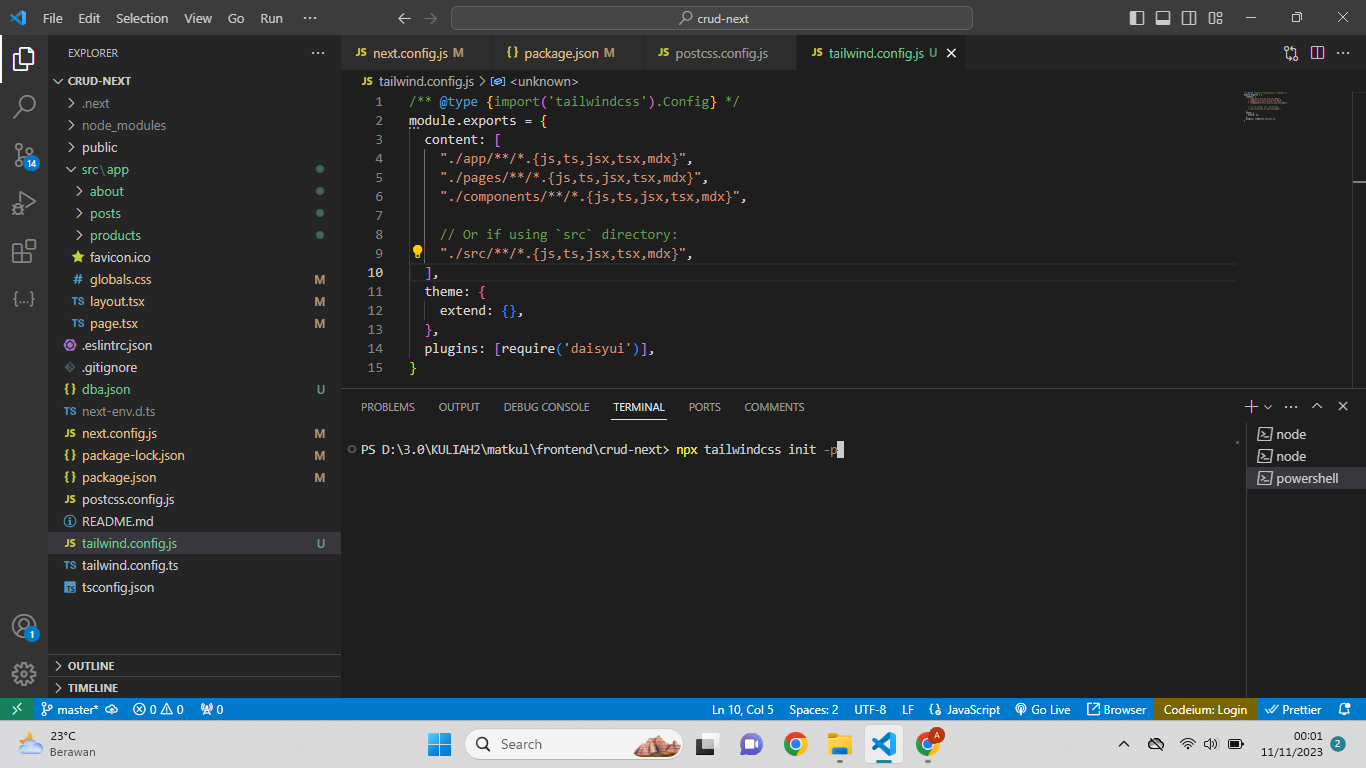
1. Install Next versi 13 ( pastikan telah menginstall node yang mumpuni )



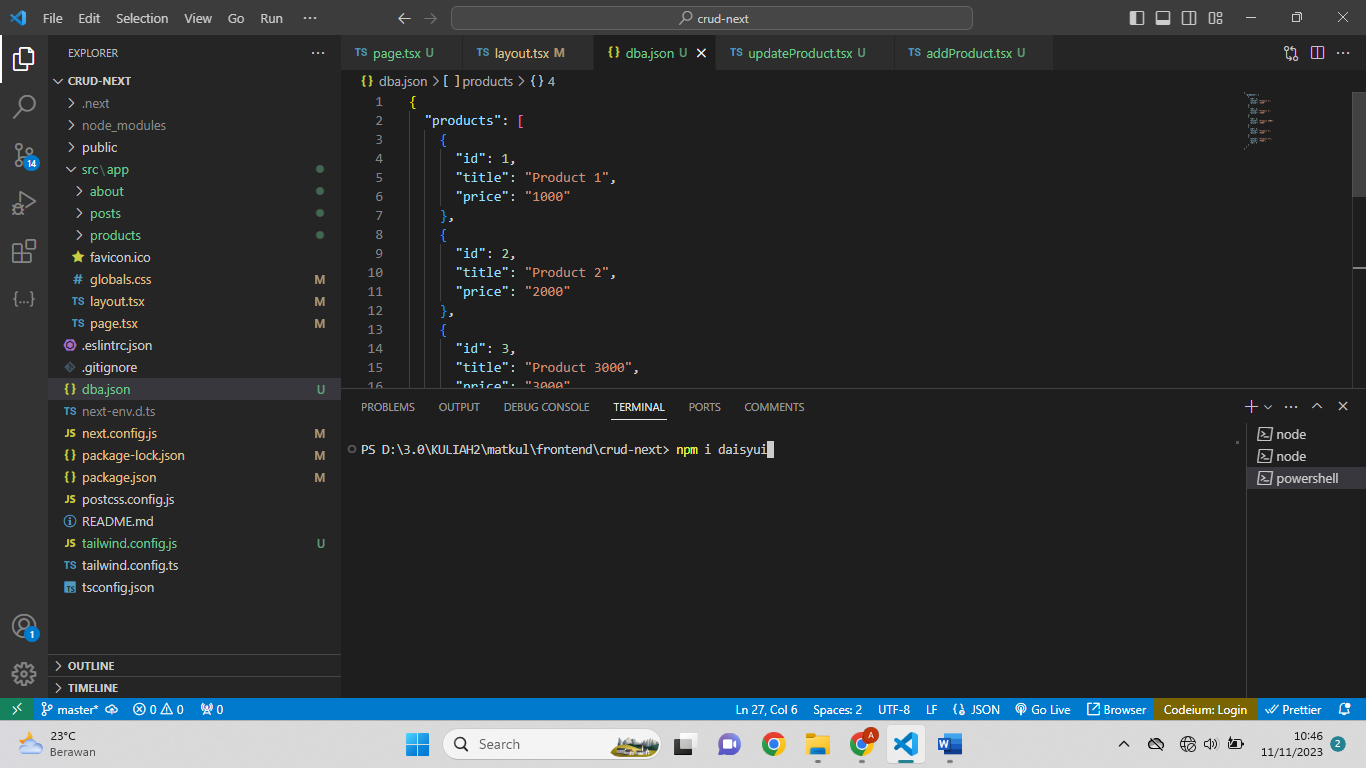
1. Install tailwind css untuk membantu kita membuat tampilan



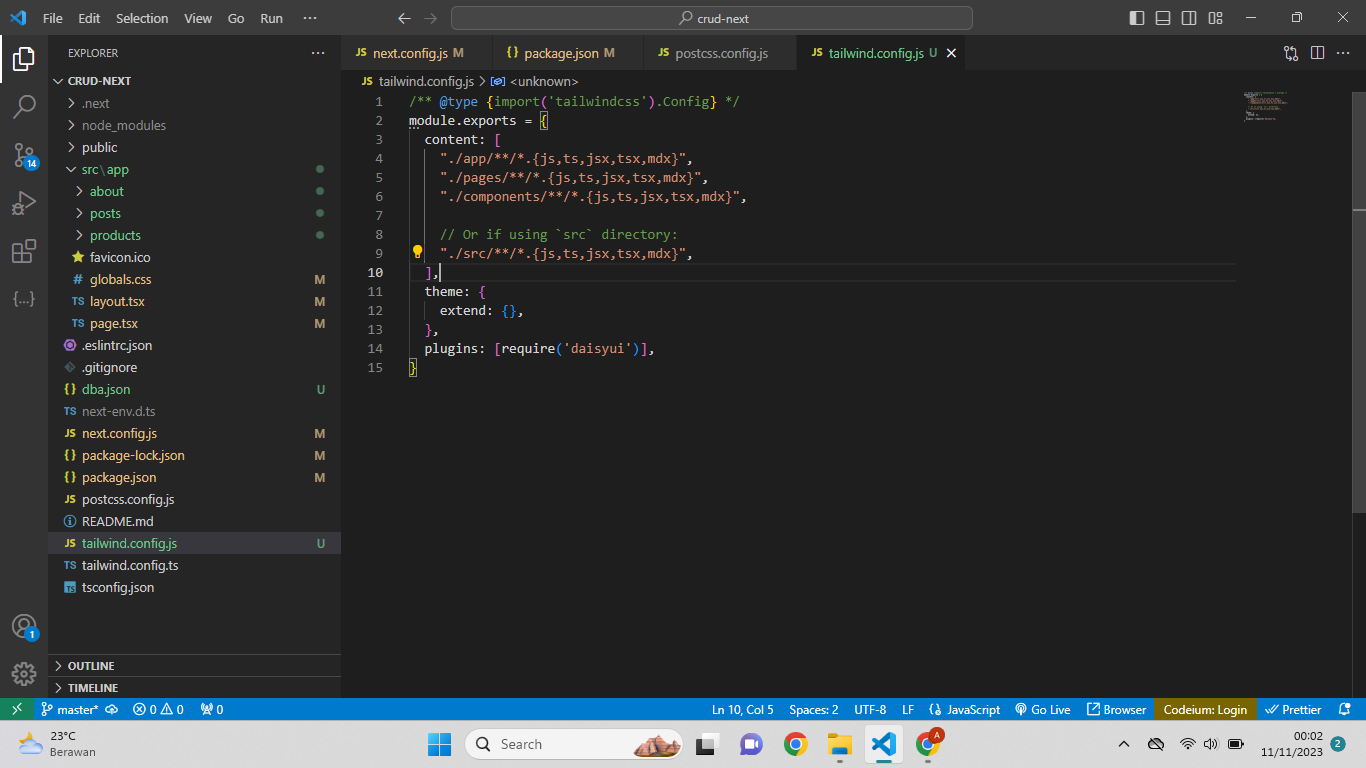
1. Inisialisasi tailwindcss untuk membuat file tailwind.config.js



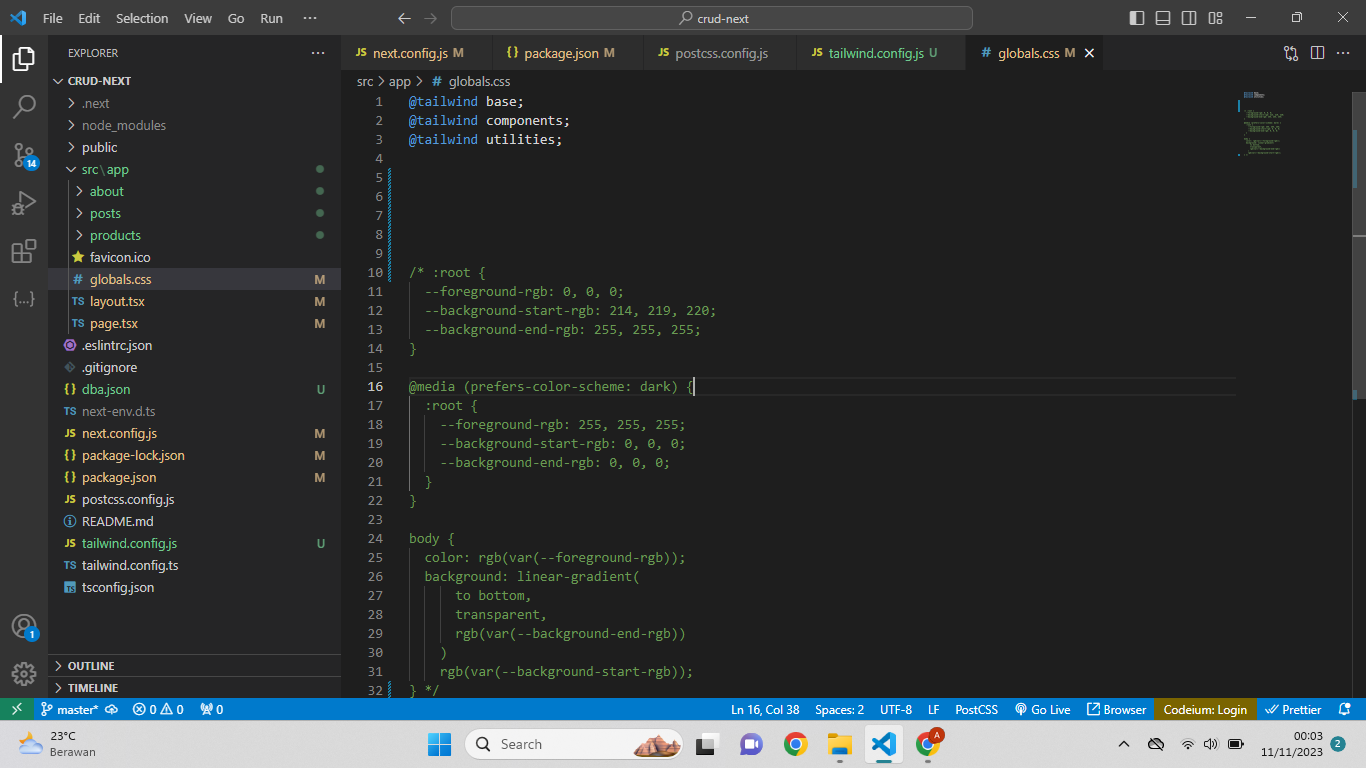
1. Kita juga akan menggunakan daisyui yang nantinya juga memudahkan kita untuk membuat tampilan



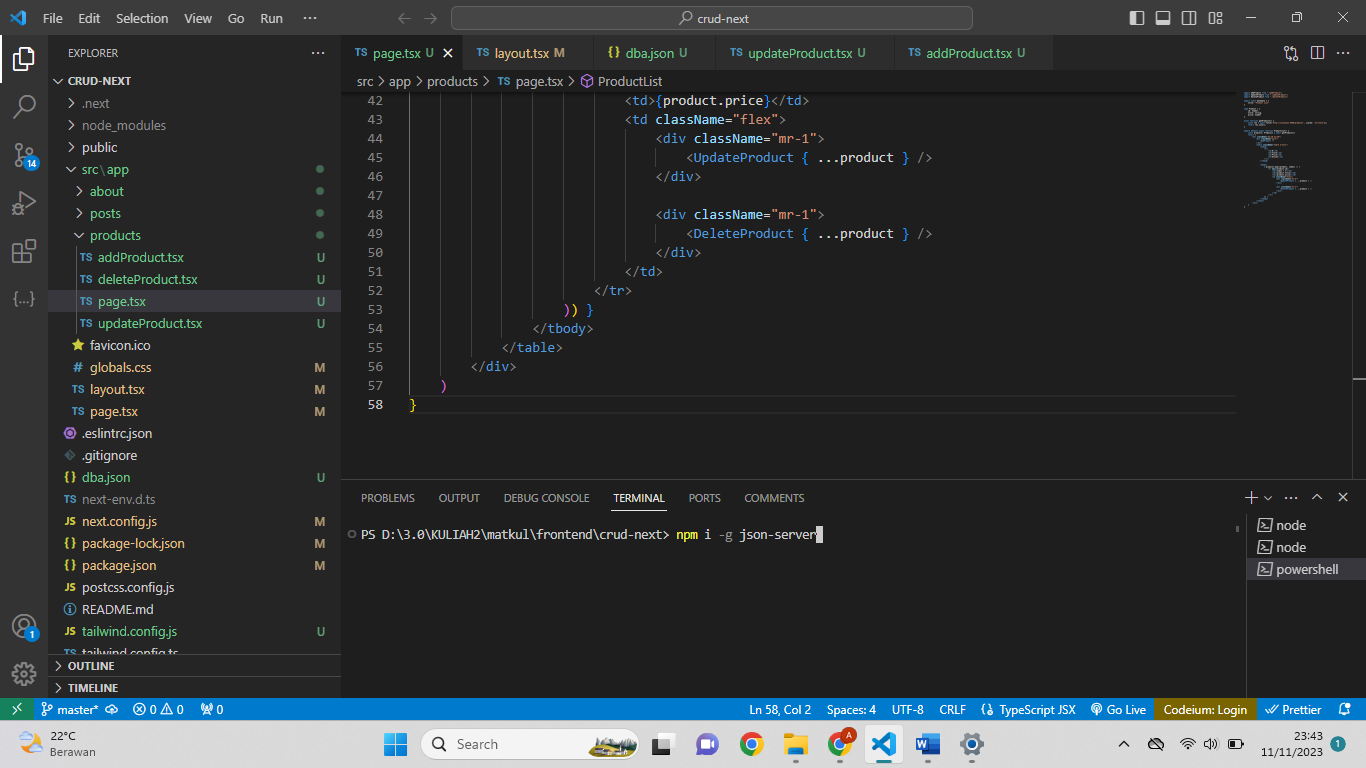
1. Konfigurasi file tailwind.config.js agar otomatis file2 yang ada pada framework next mendapatkan style tailwind ( setup tailwind untuk next )



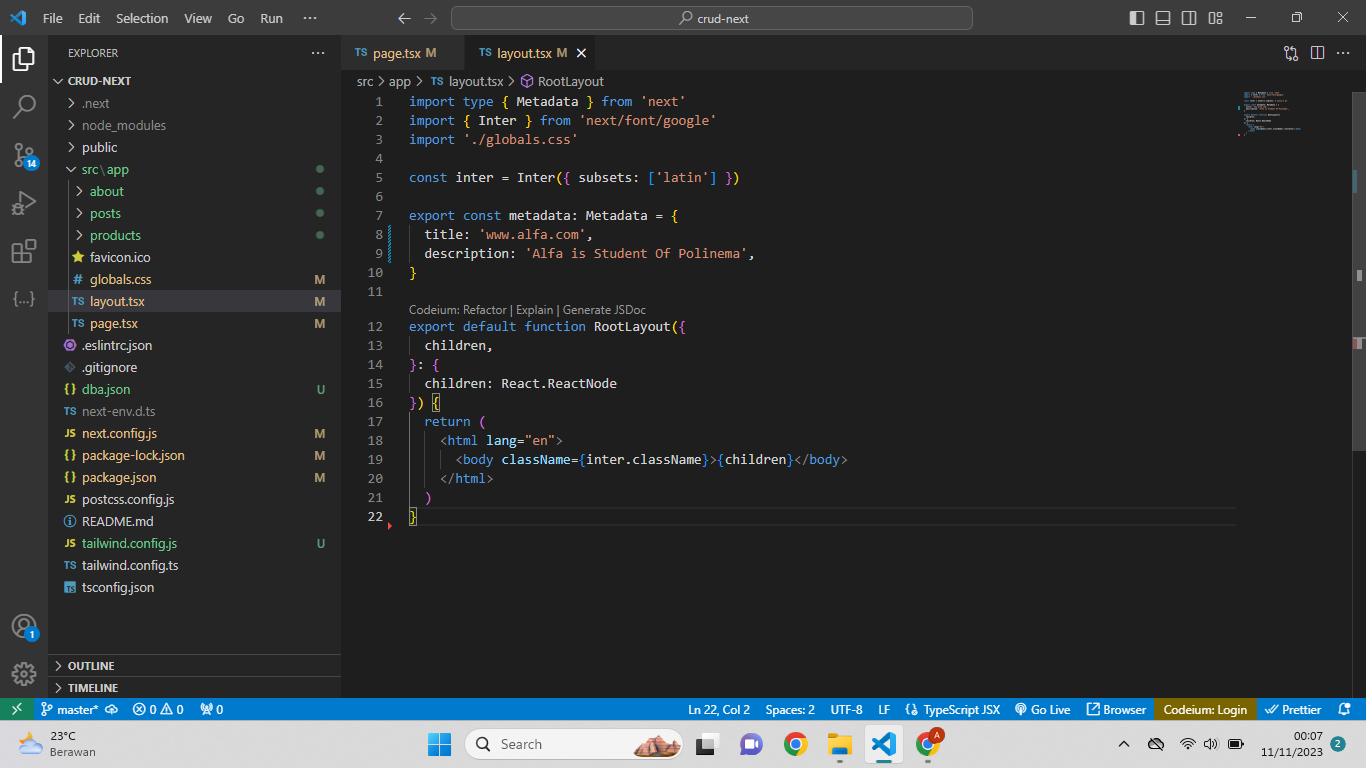
1. Tambahkan directive tailwind agar style tailwind dapat berjalan pada file2 yang sudah ditentukan sebelumnya ( jangan lupa import lokasi file global.csss pada file layout.tsx )



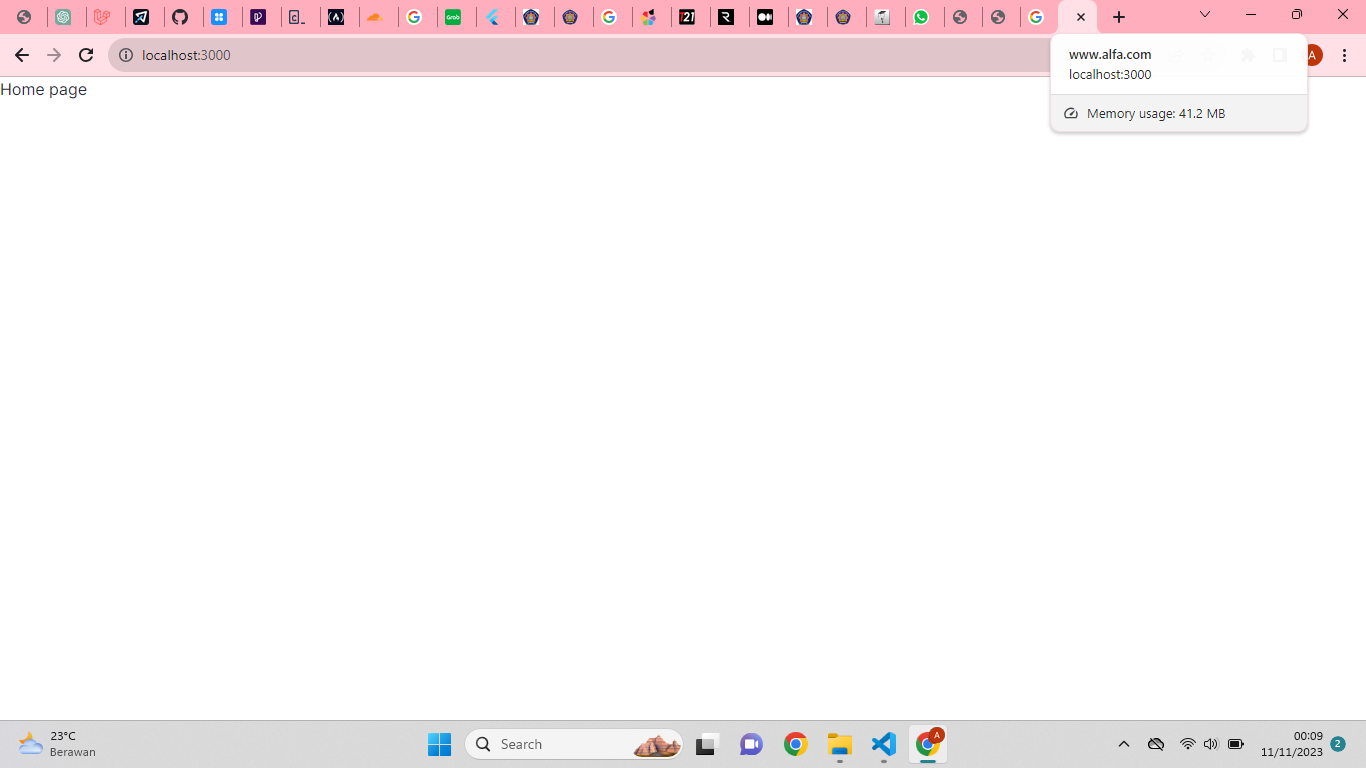
1. Kita juga akan install json-server untuk membuat API ala2 database kita



1. Pada layout.tsx terdapat export metadata, misal atribut title akan merubah judul website



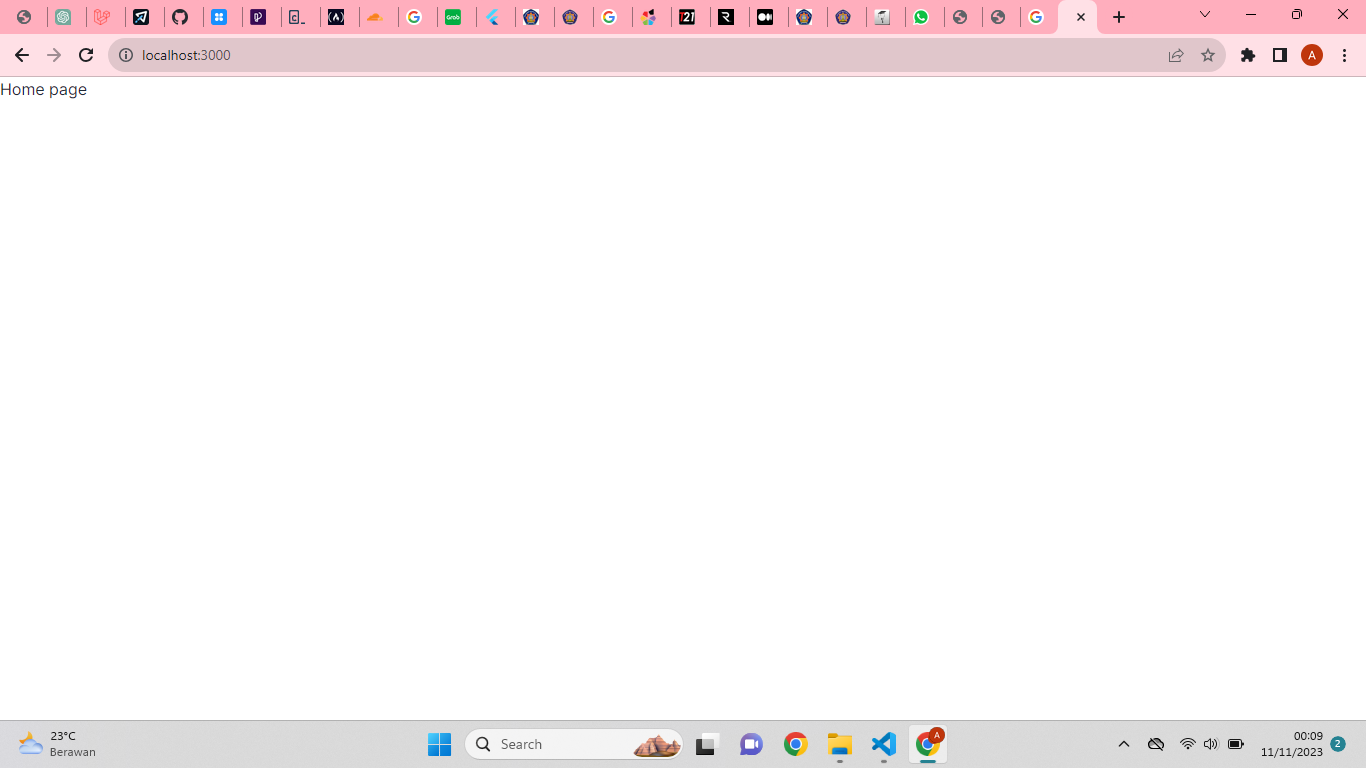
Hasil :



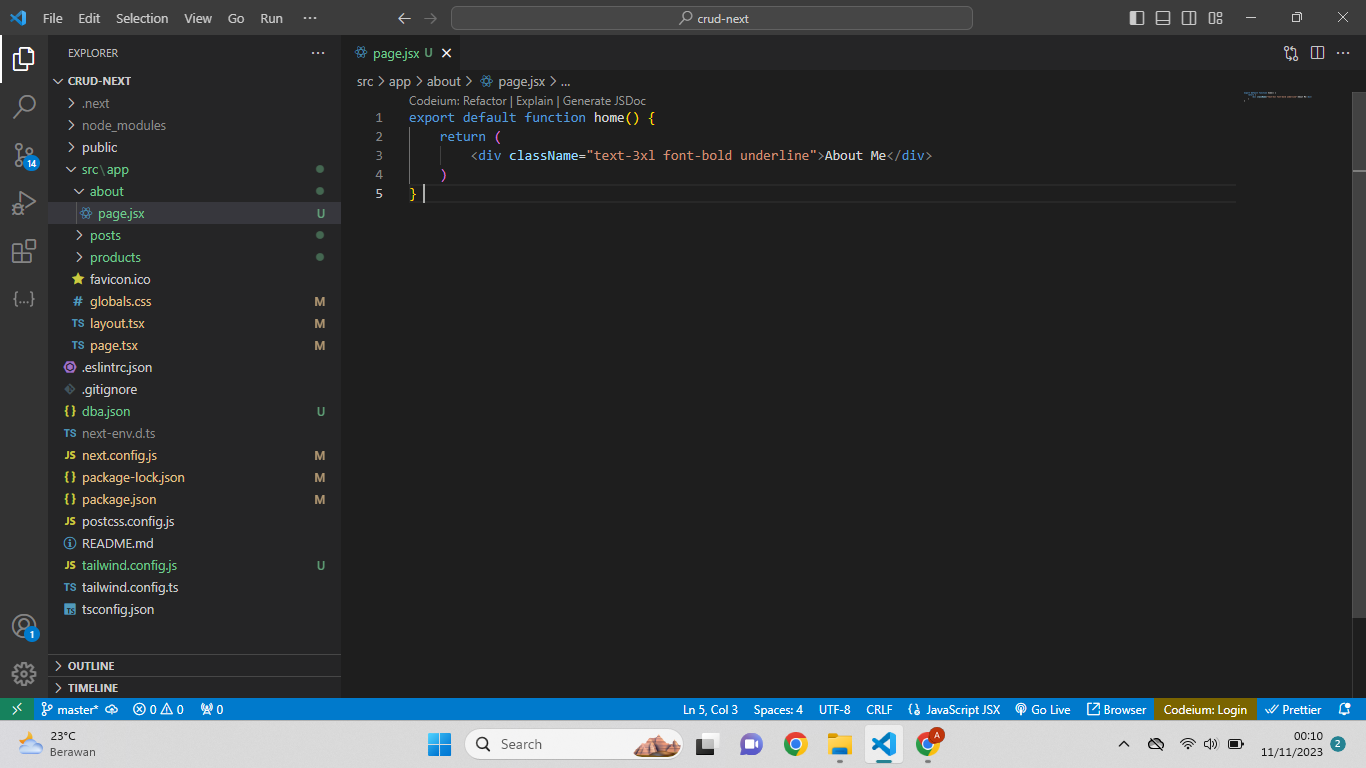
1. Untuk halaman awal atau root akan otomatis menapilkan page.tsx pada folder src/app.

Keyword export default function digunakan untuk menjalankan suatu function ketika file tersebut dipanggil

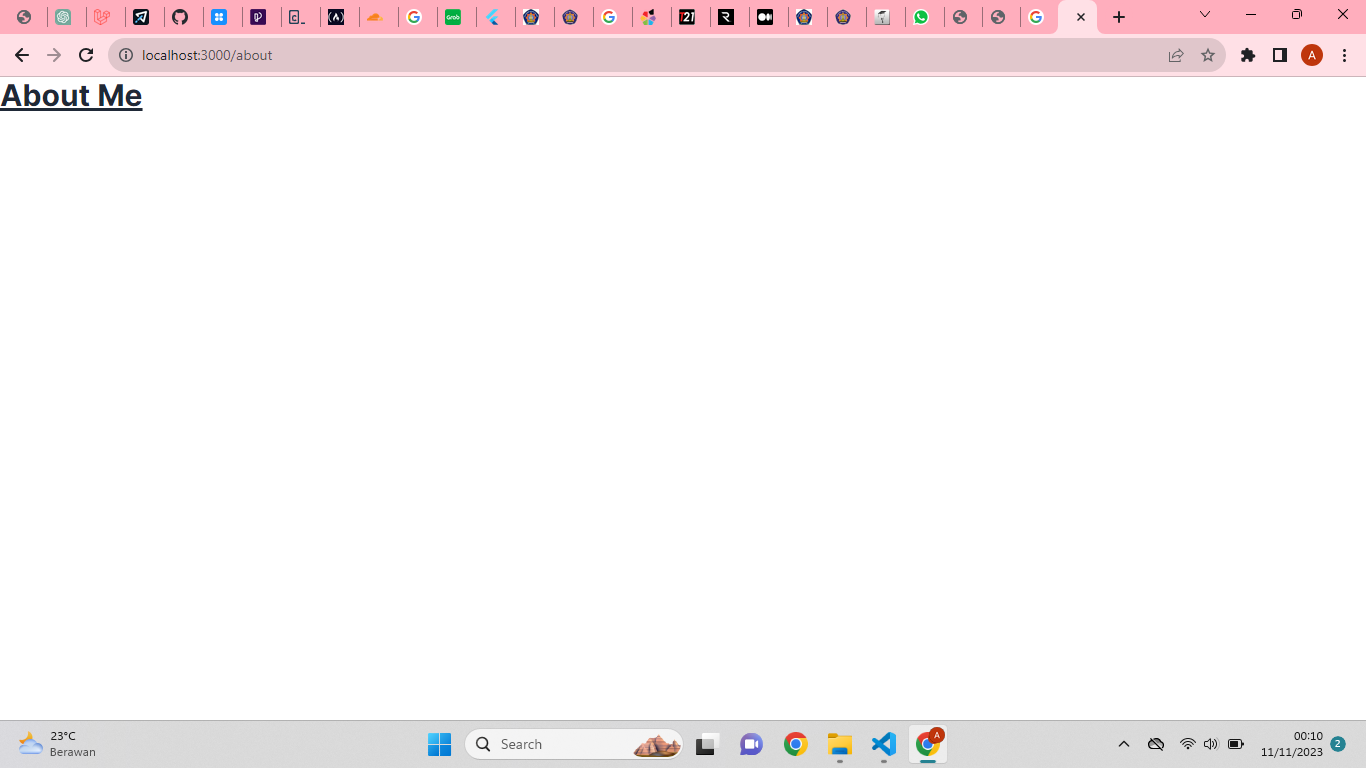
Hasil ( halaman root ) :



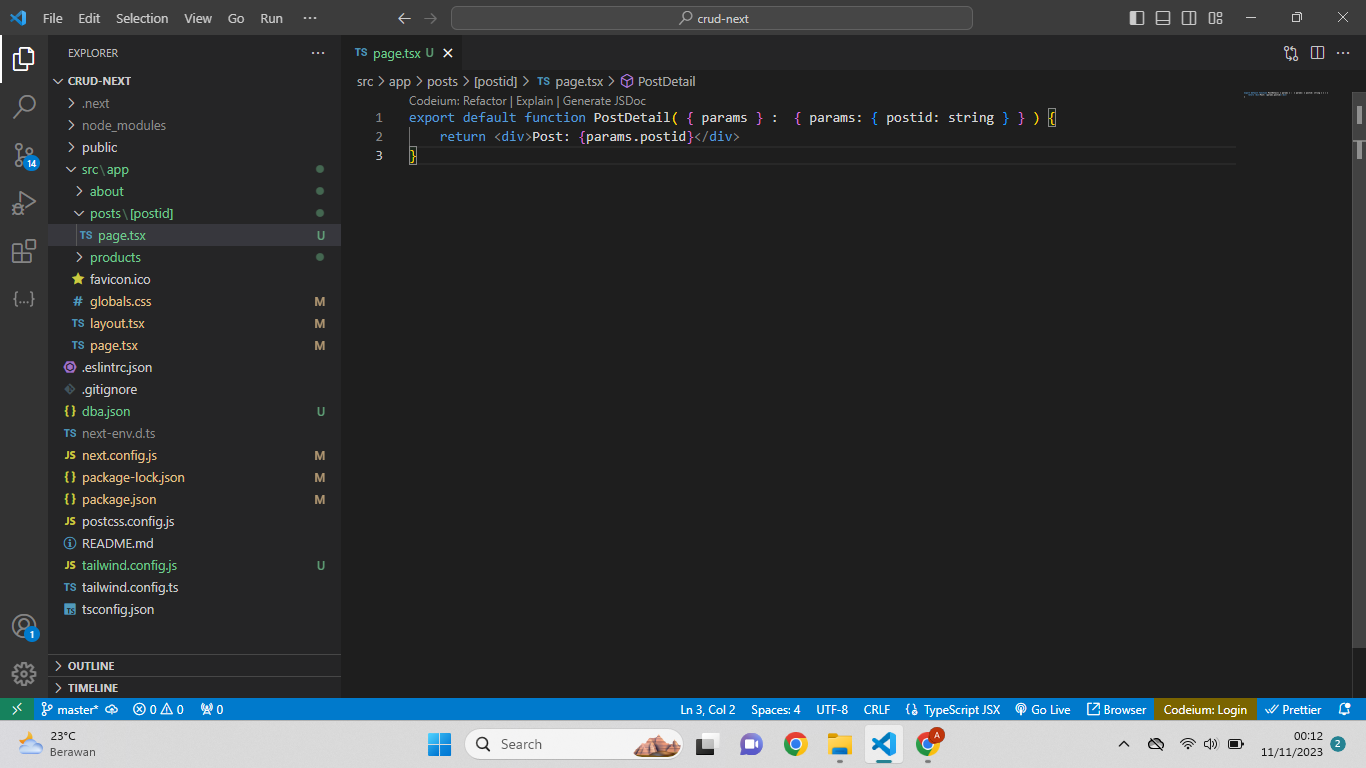
1. Next JS sudah sangat memudahkan kita untuk membuat routing. Semisal kita ingin membuat router /about, kita hanya perlu menambahkan folder about pada src/app maka otomatis akan menjadi sebuah route. Lalu saat ingin menampilkan halaman kita lakukan cara yang sama dengan sebelumnya dengan membuat file page.jsx



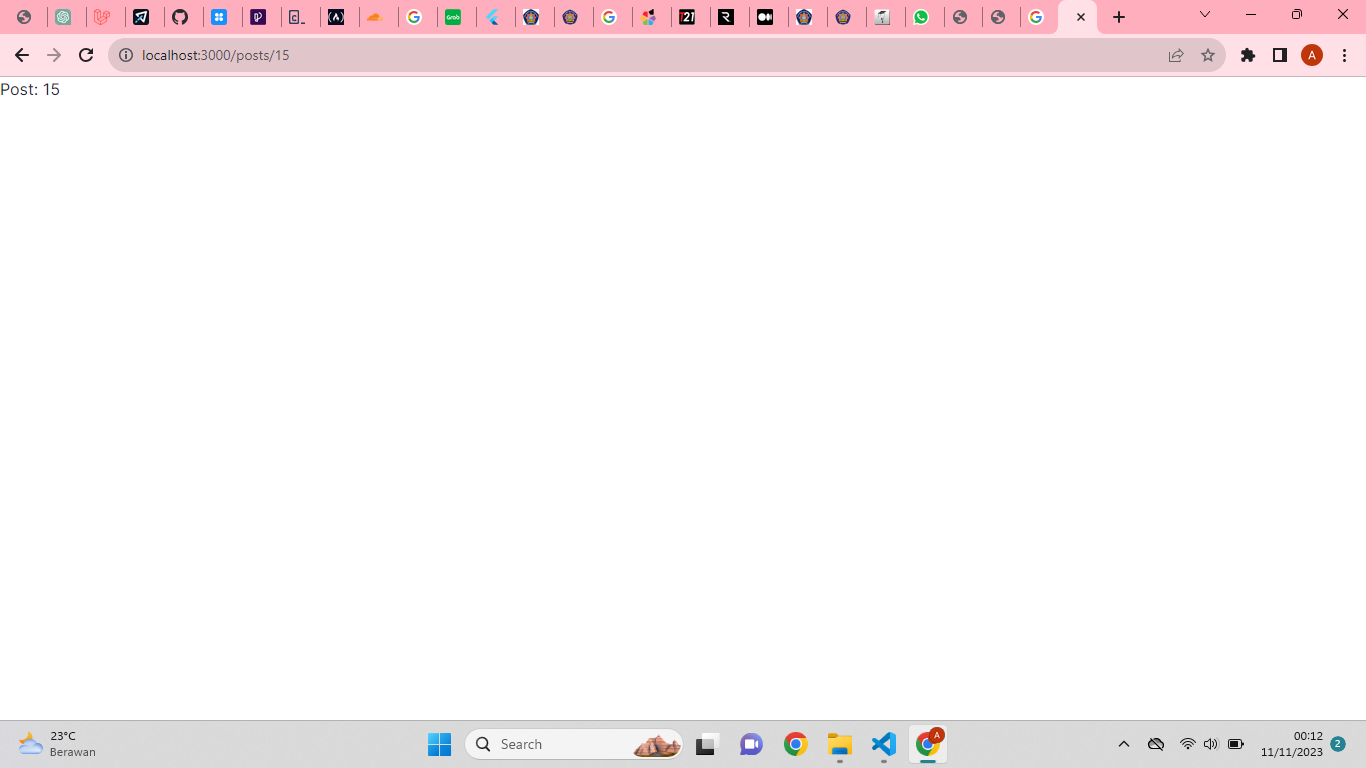
Hasil :



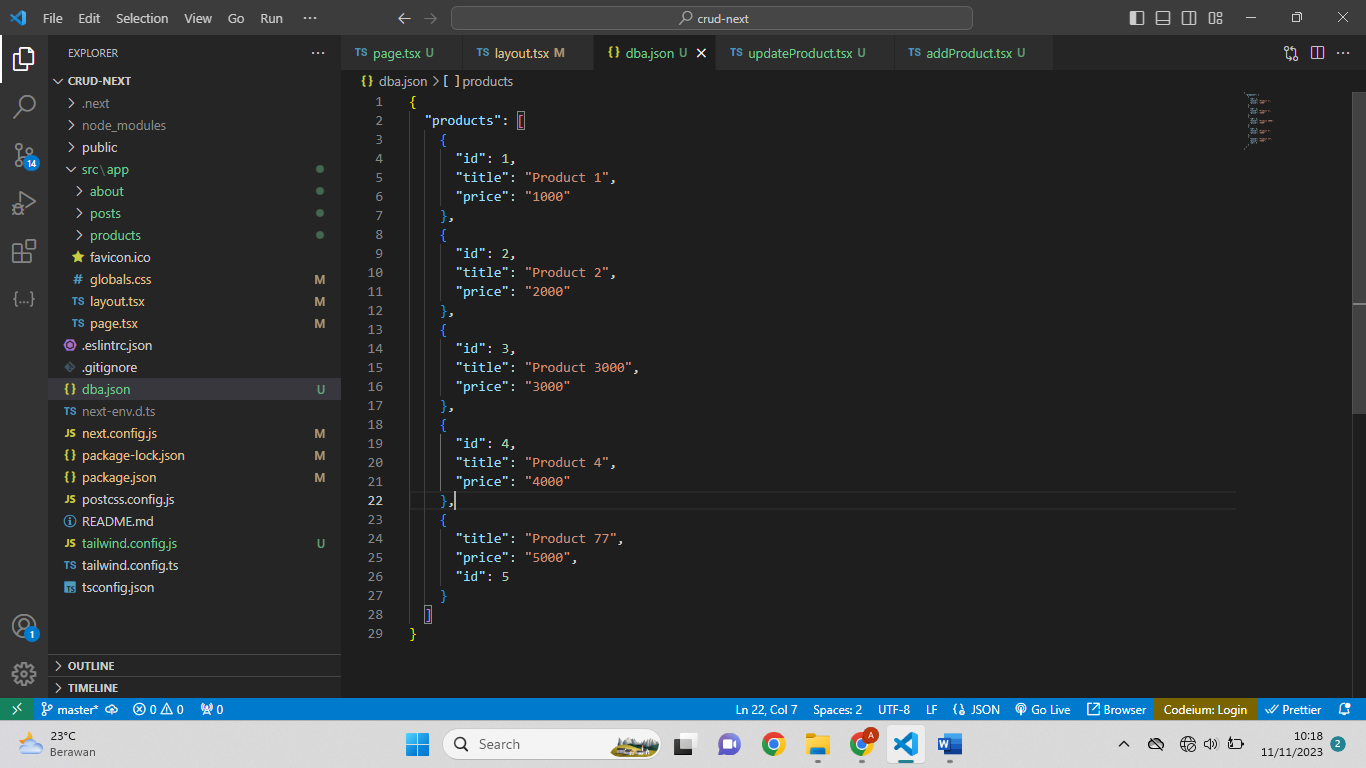
1. Kita juga bisa menerima parameter pada function di file yang terhubung dengan route /posts/[postid] pada kasus ini. yang artinya terdapat route posts dan sub route [postid] sebagai tempat menerima parameter dan ditangkap pada function untuk ditampilkan.



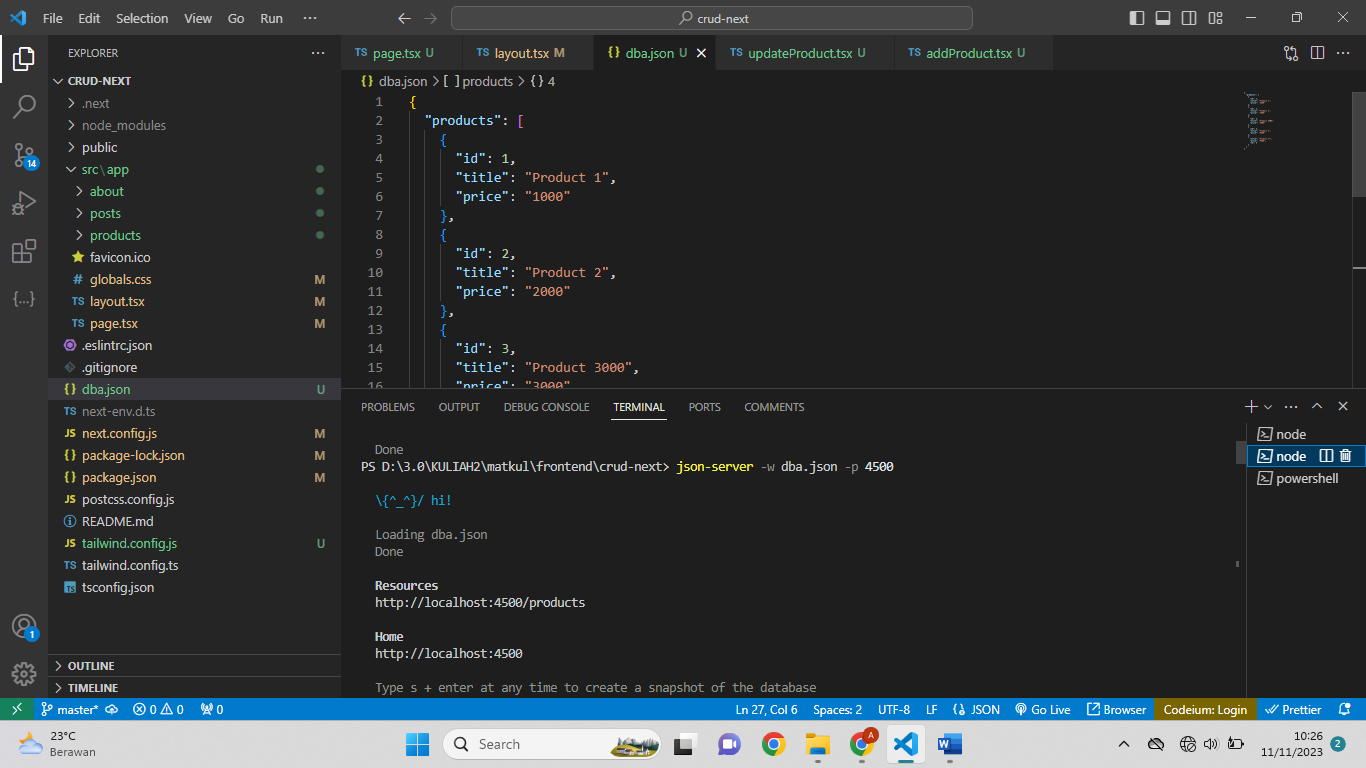
Hasil ( nilai 15 akan tergantung dengan sub route di belakang route /posts ):



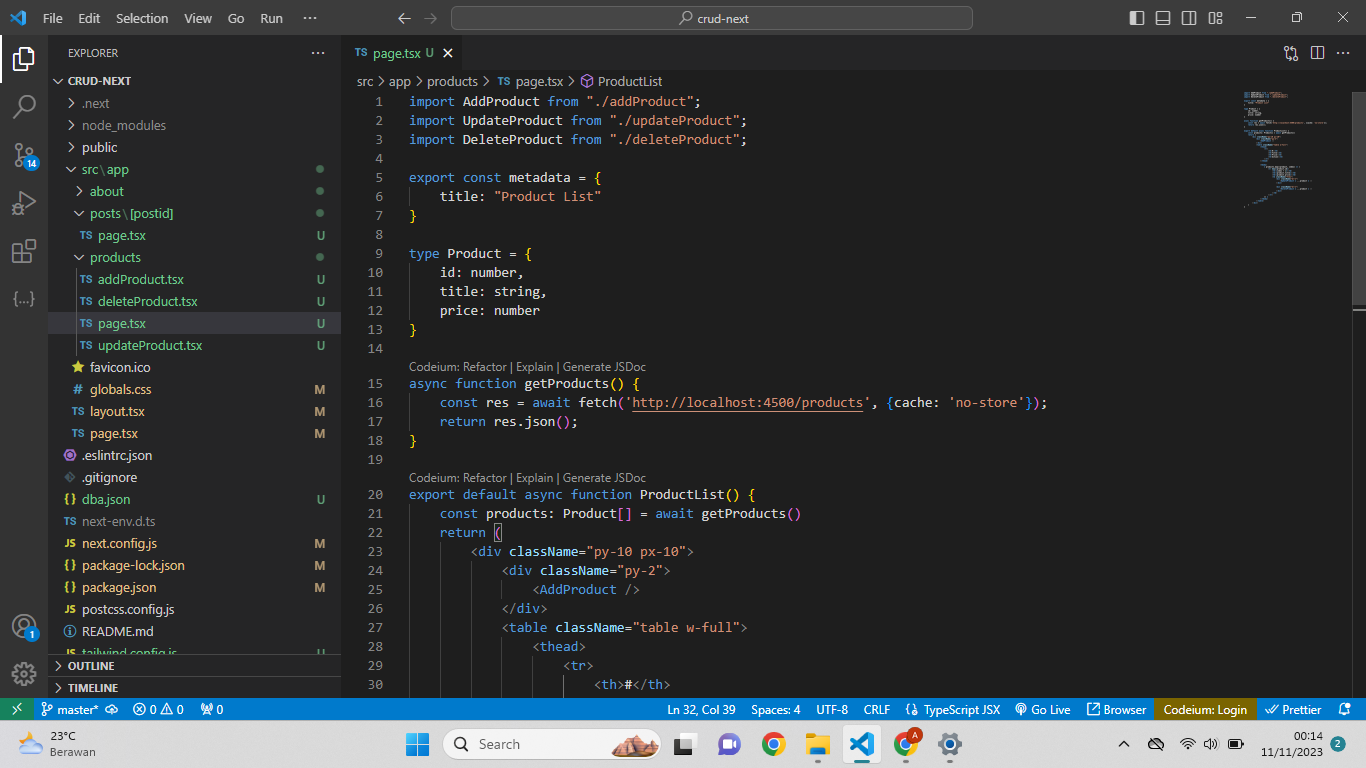
1. Dibawah ini adalah fake API dba.json dengan route /products dengan atribut id, title, dan price dan kita coba buat beberapa data di dalmanya



1. Berikut adalah sintaks untuk menjalankan server API.
   * + -w untuk menentukan file api yang ingin dijalankan
     + -p untuk membuat port angka agar tidak bentrok

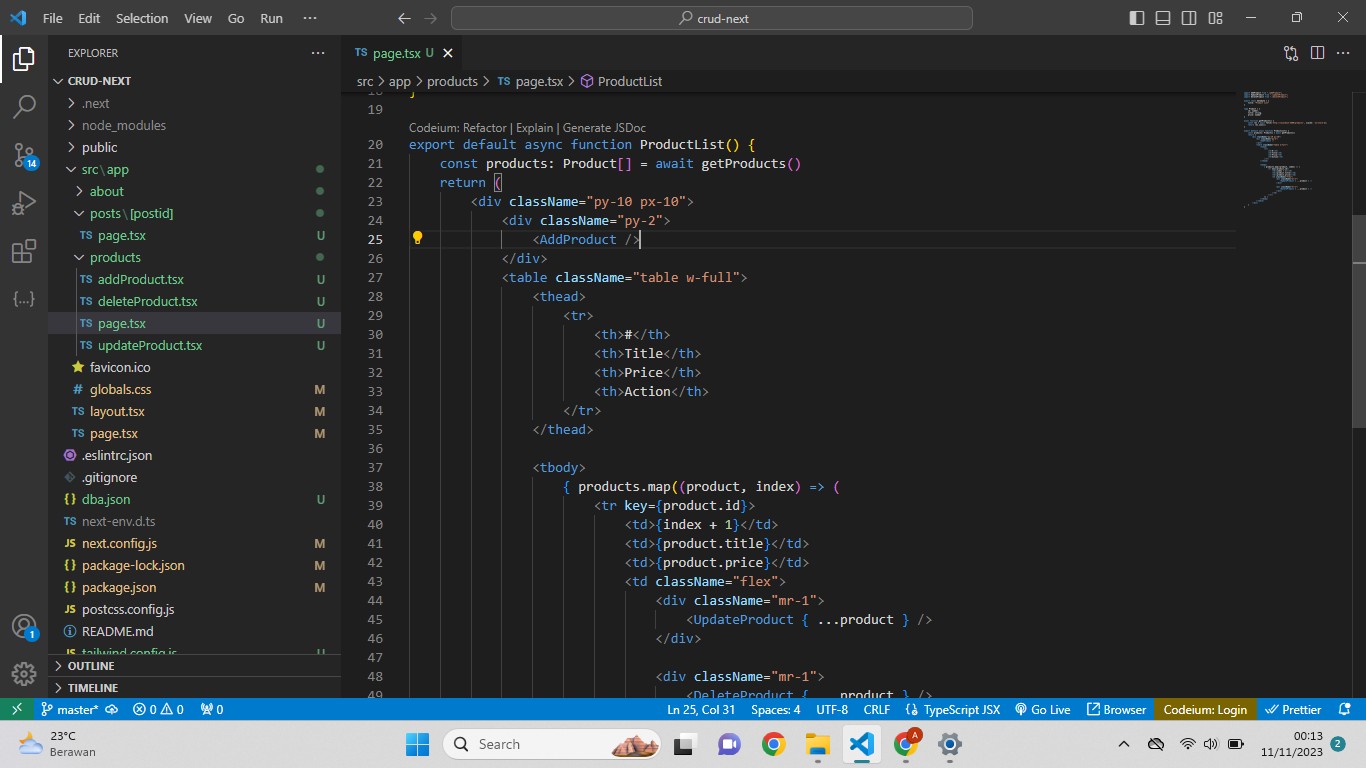


1. Kita mulai membuat halaman untuk menampilkan data dari API yang telah kita buat dengan function getProducts() pada file page.jsx, artinya halaman ini akan menjadi halaman awal ketika akses route products

* type Product untuk menentukan jenis data yang akan kita tampilkan
* fetch untuk mengambil data API yang telah kita buat ( default GET )
* cache dengan value ‘no-store’ bertujuan agar ketika API berubah maka data fetch juga akan berubah  
  

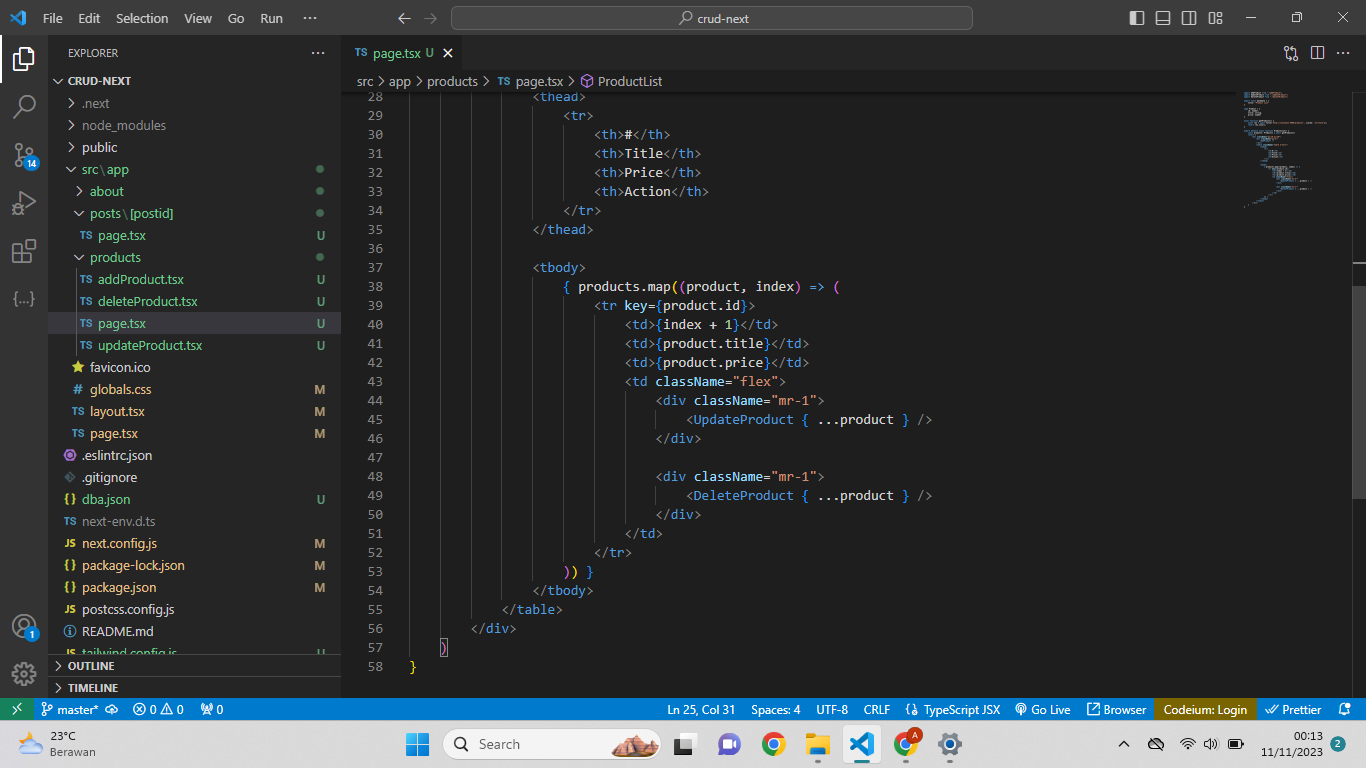
1. Kita akan membuat tampilannya dalam bentuk table yang kita beri style tailwind dan daisy

const products berisi semua data product hasil fetch dari getProducts()



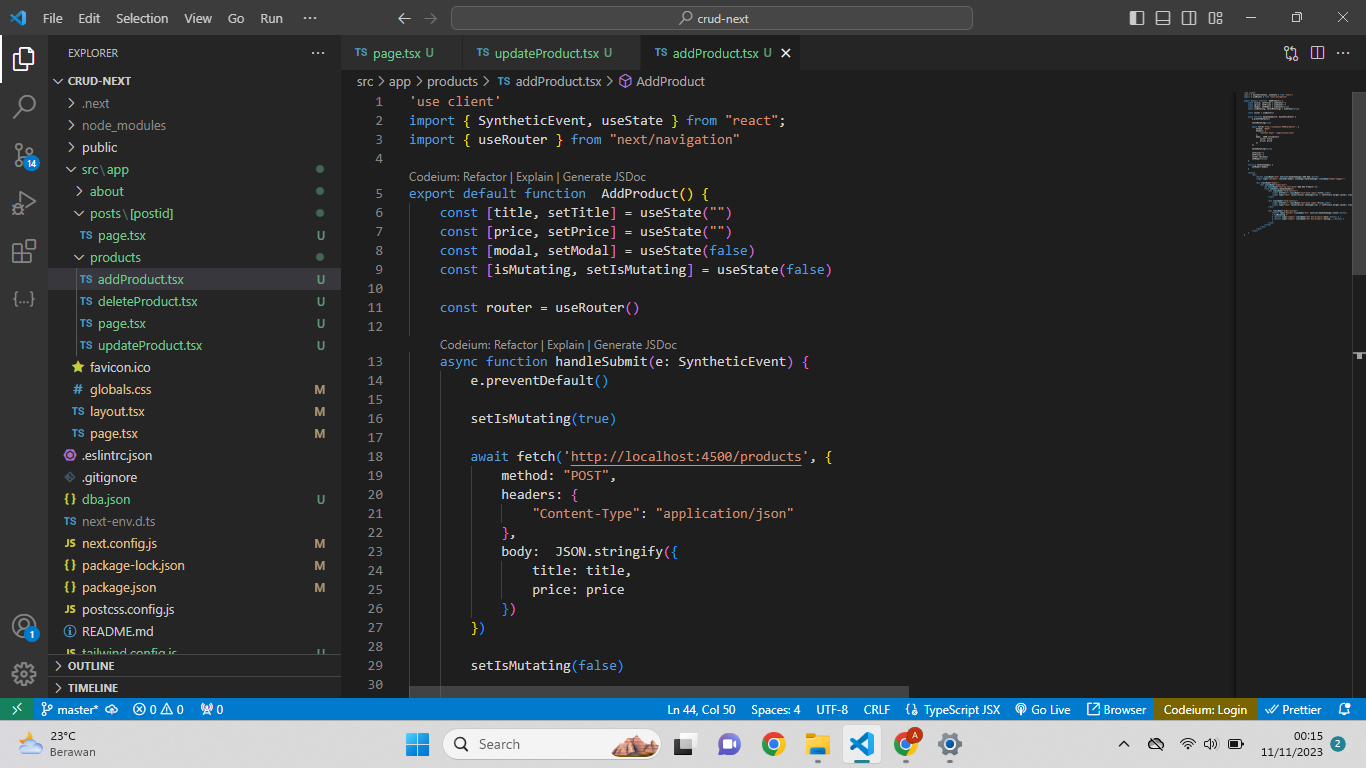
1. Pada tiap baris pada tabel kita looping dengan map

* map merupakan fungsi untuk iterasi object dan mengambil nilai di dalamnya
* Setiap looping baris <tr> perlu wajib diberi atribut key dan diisi data yang uniq yaitu id dari tiap object sebagai syarat untuk loop
* Di kolom terakhir kita panggil component UpdateProduct dan DeleteProduct yang telah kita import sebelumnya dan akan kita buat setelah ini

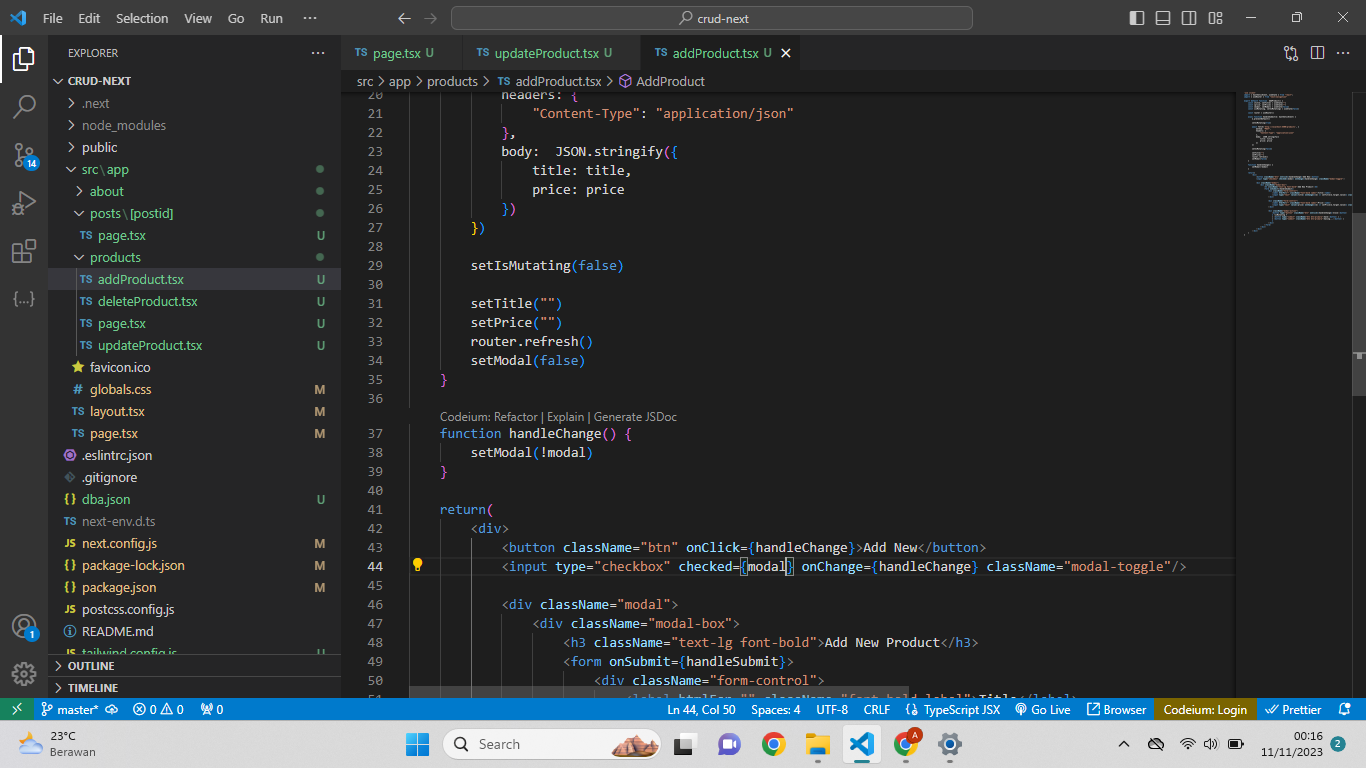


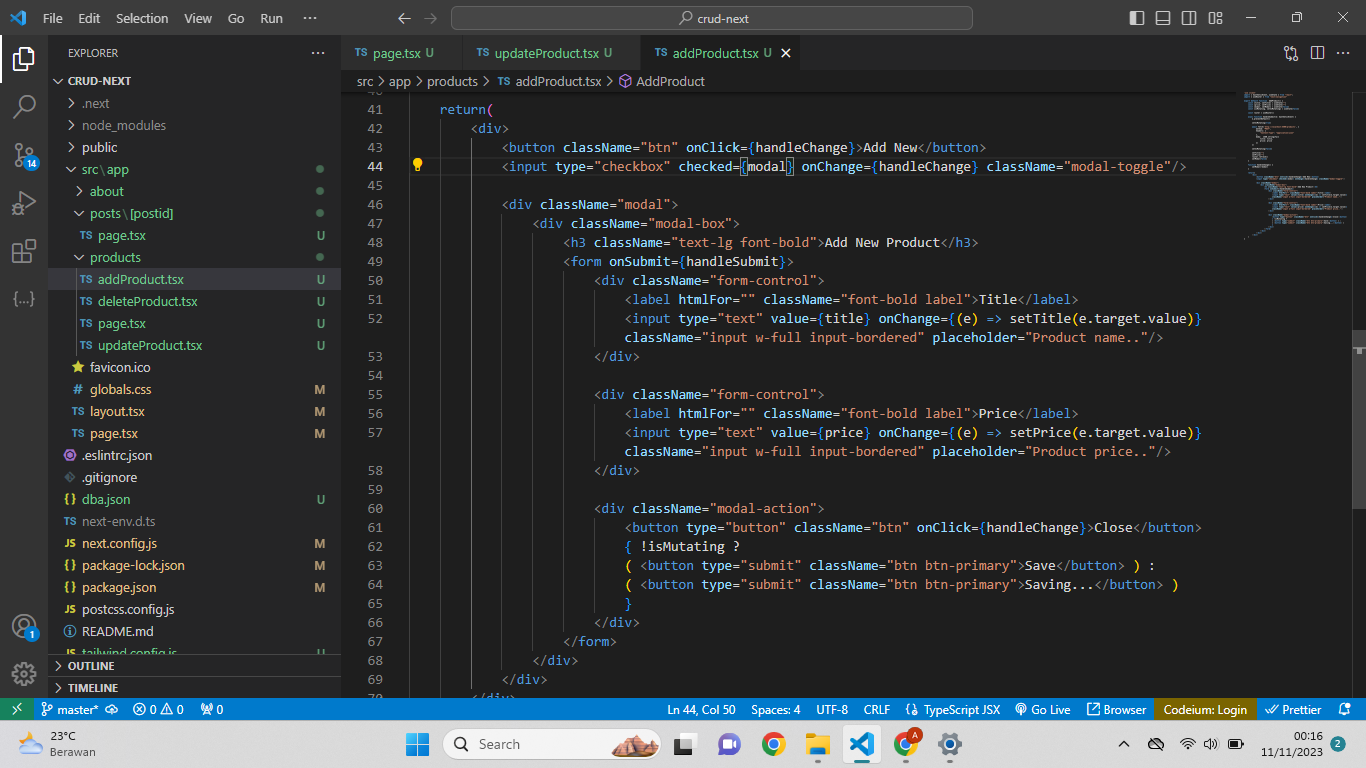
1. Kita lanjutkan membuat component AddProduct() dengan import beberapa package Atribut title dan price adalah state dengan inisialisasi “ ” atau string kosong

* Atribut modal dan isMutating adalah state dengan inisialisasi false
* Di dalam AddProduct terdapat handleSubmit dengan menggunakan SyntheticEvent untuk handle event
* Terdapat fungsi fetch untuk akses API dengan metode POST yang artinya mengambil semua data dari url API yaitu /products
* Atribut body menangkap data title dan price lalu mengubahnya menjadi bentuk string dengan JSON.stringify
* setMutating pada awalnya bernilai true artinya masih melakukan proses pengambilan data dari API, jika Sudah selesai maka di set menjdi false



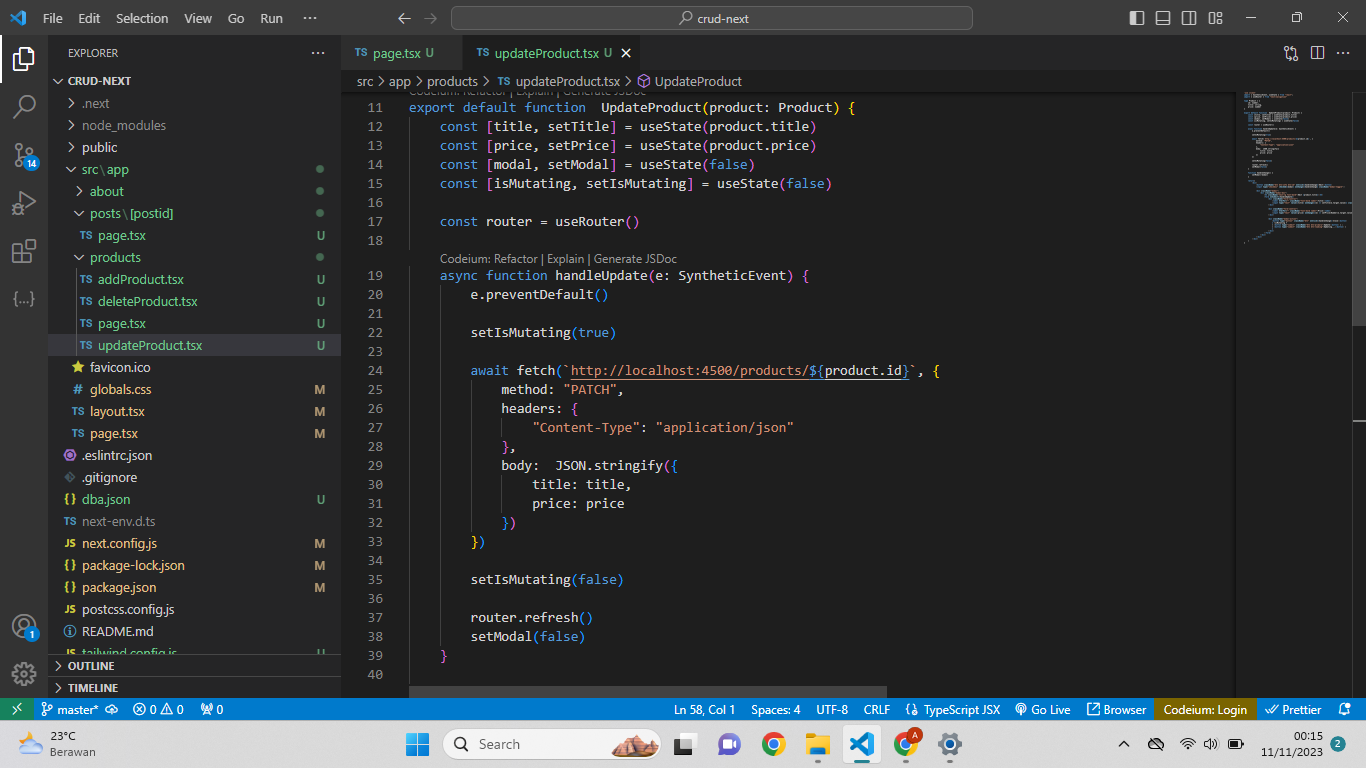
* function handleChange digunakan saat klik tombol Add New akan memunculkan sebuah modal yang berisi form input data yang kita kirimkan ke API
* Setelah selesai mengirim data lewat method POST, kita ubah state setTitle, setPrice dan setModal ke bentuk awal. Karena atribut state akan berubah sesuai dengan keadaan secara realtime dan perlu di set kembali keadaanya
* handleSubmit yang telah kita buat sebelumnya dieksekusi saat form dikirm
* di tiap bagian input masing2 memiliki value dan setValue yang bersumber dari state yang mereka miliki semisal title mengambil dari state title dan untuk mengubah nilainya menggunakan setTitle. Terdapat dua inputan yaitu title dan price



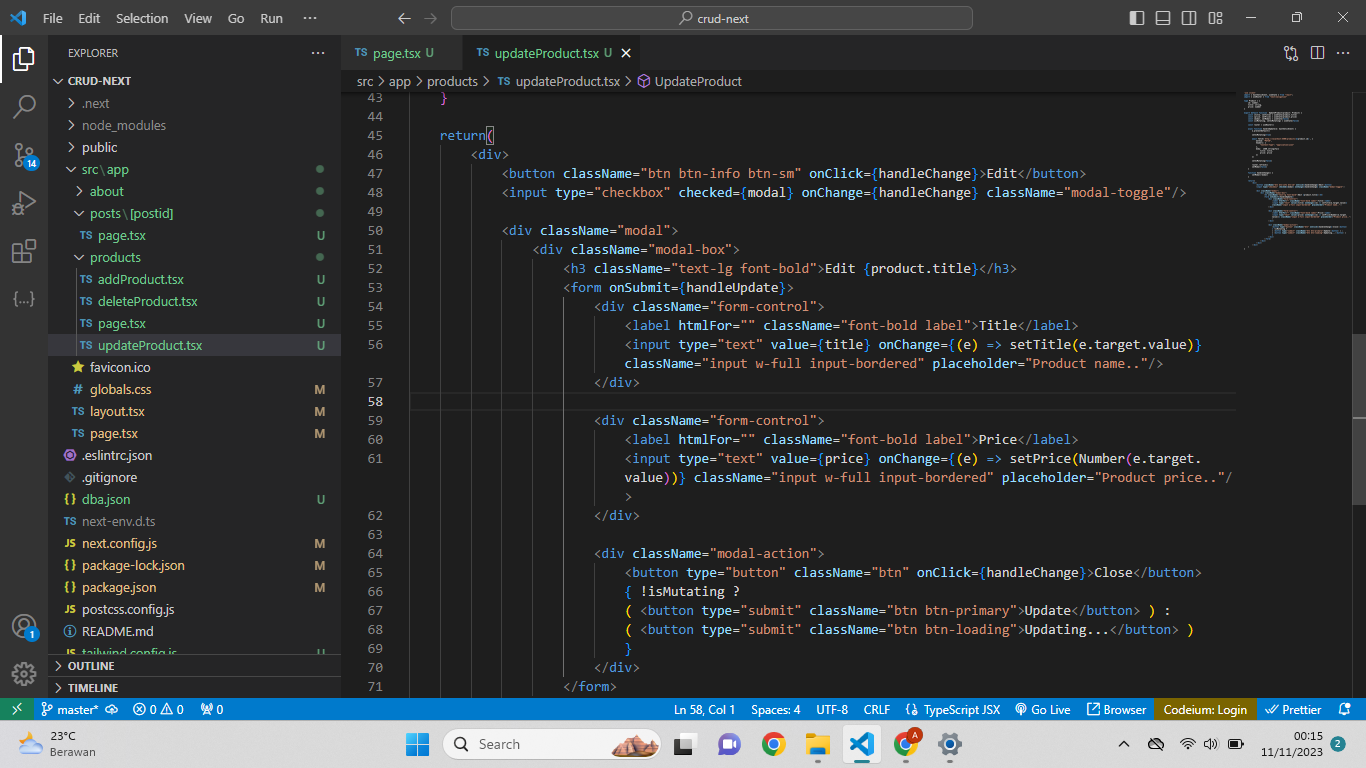


1. Lanjut membuat component UpdateProduct yang memiliki mekanisme mirip dengan AddProduct sebelumnya namun sedikit berbeda

* Saat ingin update data kita wajib menambahkan parameter di url API fetch yang dalam kasus ini adalah /products berupa id untuk mengetahui product mana yang ingin di update. Contoh: /products/1
* Nilai awal dari state title dan price kita set dengan nilai yang mereka miliki
* Method yang kita gunakan adalah PATCH yang berarti untuk update suatu data pada API

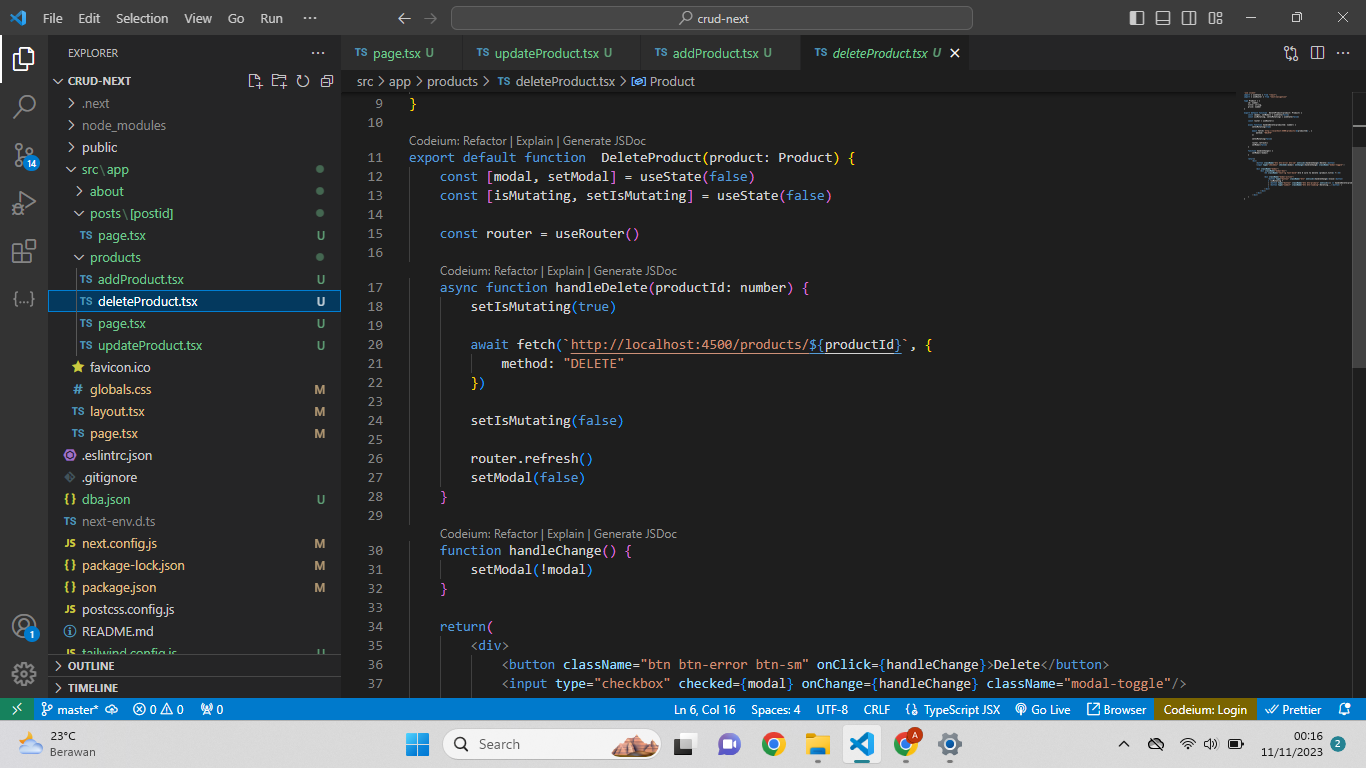


* Karena telah memiliki nilai data maka kita tampilkan pada bagian judul dan value inputan
* Pada bagian onChange() mengambil value yang diinputkan dan merubahnya state yang masing2 inputan miliki

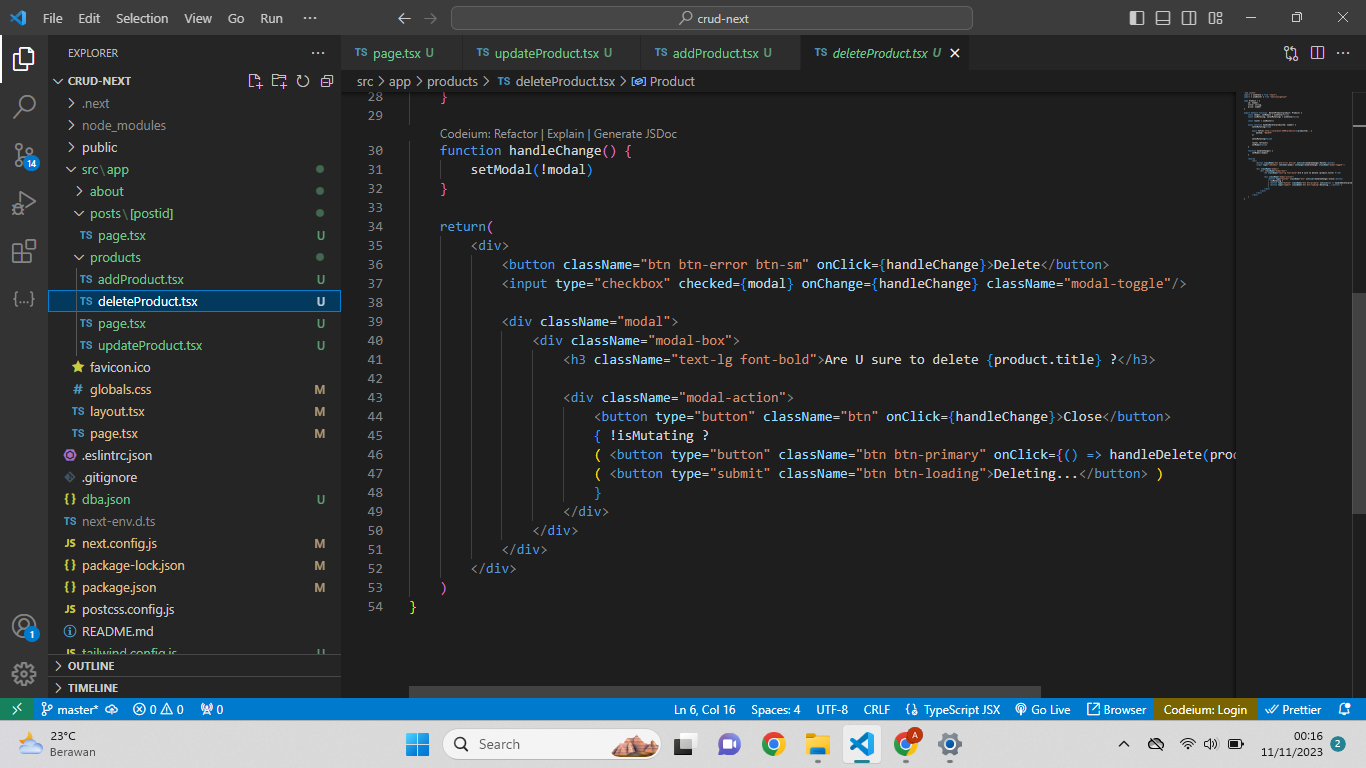


1. Terakhir kita buat deleteProduct untuk menghapus data dari product

* Untuk menghapus kita juga memerlukan parameter di URL sama seperti update sebelumnya
* Parameter DeleteProduct cukup menggunakan data product tanpa SyntheticEvent
* State yang digunakan hanyalah modal dan isMutating
* Terdapat handleDelete yang memerlukan parameter productId
* Metode yang digunakan adalah DELETE yang berarti menghapus suatu data pada API

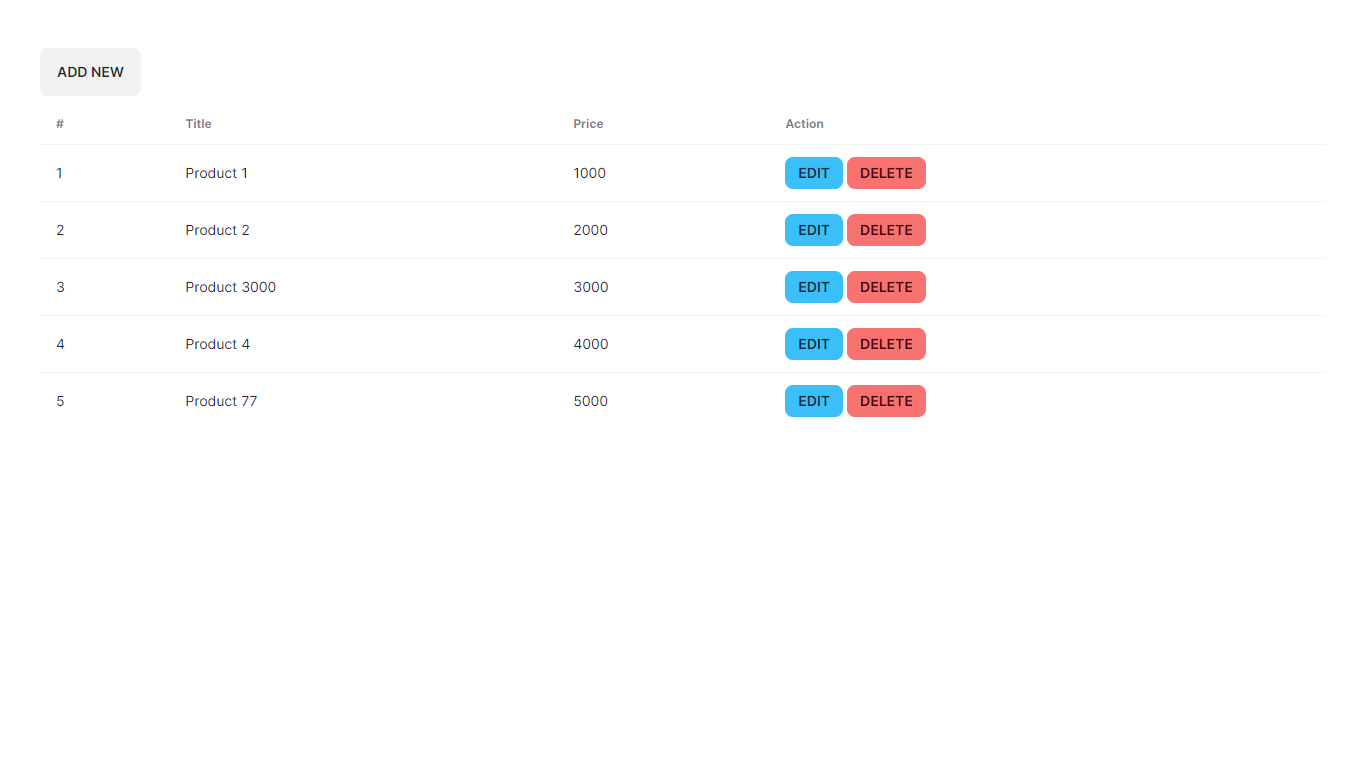


* Modal hanya berisi peringatan apakah yakin ingin menghapus dengan menampilkan data yang ingin dihapus



Hasil :

* Coba untuk menjalan operasi CRUD



**SELAMAT MENCOBA!**