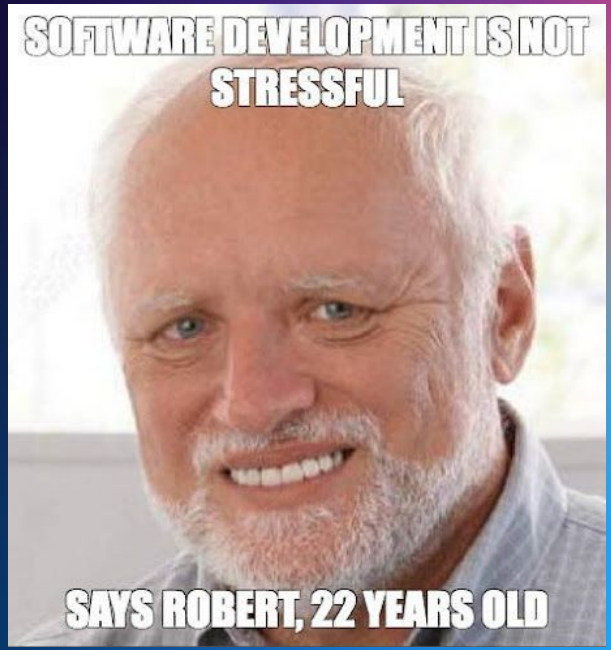
Clase N° 6: NextJS un poco avanzado



Crombie

Objetivos de la clase

- ➔ Darle estilos a nuestra tienda creada previamente
- ➔ Aprender a usar Rutas dinámicas y aplicarlo a nuestro proyecto

Crombie

TailwindCSS

- ➔ Es un framework de CSS que nos permite aplicar estilos directamente en nuestros componentes de React.
- ➔ Ofrece una inmensidad de clases ya definidas para aplicar estilos rápidamente.
- ➔ **Modo JIT (Just-In-Time):** Genera sólo las clases que realmente usas, reduciendo el tamaño del CSS final.

TailwindCSS

- ➔ Responsive: agregando “sm” “md” o “lg” podemos rápidamente darle estilos adaptables a nuestros componentes.
- ➔ Ejemplo:

```
<div class="bg-blue-500 p-4 text-white font-bold text-center">
    ¡Hola, Tailwind!
</div>
```

TailwindCSS

```
<div class="max-w-md mx-auto bg-white rounded-xl shadow-md overflow-hidden
md:max-w-2xl">
  <div class="md:flex ">
    <div class="md:shrink-0">
      <img class="h-48 w-full object-cover md:h-full md:w-48" src="/img/photo.jpg"
alt="Foto">
    </div>
    <div class="p-8">
      <h1 class="text-lg font-bold">¡Bienvenidos a Tailwind!</h1>
      <p class="mt-2 text-gray-500">Desarrolla rápido con clases de utilidad.</p>
    </div>
  </div>
</div>
```

Crombie

Link - useRouter - redirect

- **Link**: es el estándar para navegar entre páginas en la aplicación de NextJs. Cuenta con las ventajas de no recargar toda la página y la precarga de contenido. Ej: ir del Home al About.
- **useRouter**: se usa para el manejo de rutas en respuesta a eventos. Por ejemplo al completar un formulario, queremos que nuestro usuario vuelva al inicio de la página. También permite acceder a parámetros de la URL.
- **redirect**: se utiliza en server side. Útil para bloquear acceso a zonas restringidas o que requieren autenticación.

Introducción a rutas dinámicas

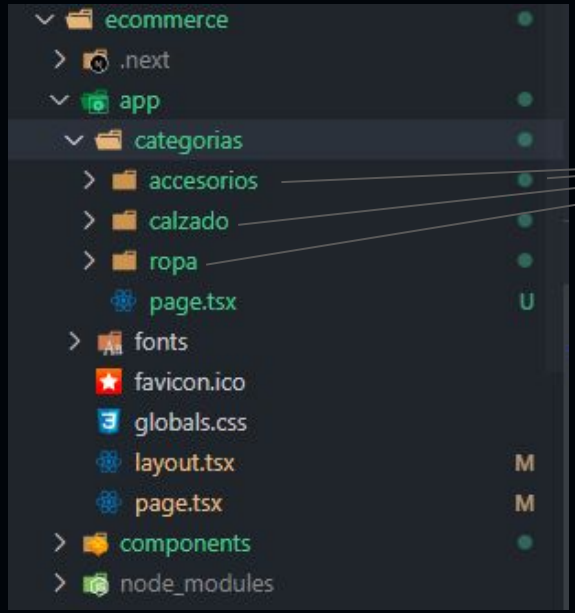
- Permiten crear rutas basadas en parámetros (variables) que no son estáticas.
- Casos de uso: páginas de productos, blogs, perfiles de usuarios.
-

```
/app
  └── usuario
      └── [id]
          ├── page.tsx
          └── layout.tsx (opcional)
```

Crombie

Rutas dinámicas: Uso incorrecto

Basado en el proyecto que creamos la clase pasada, las carpetas podrían lucir algo así:



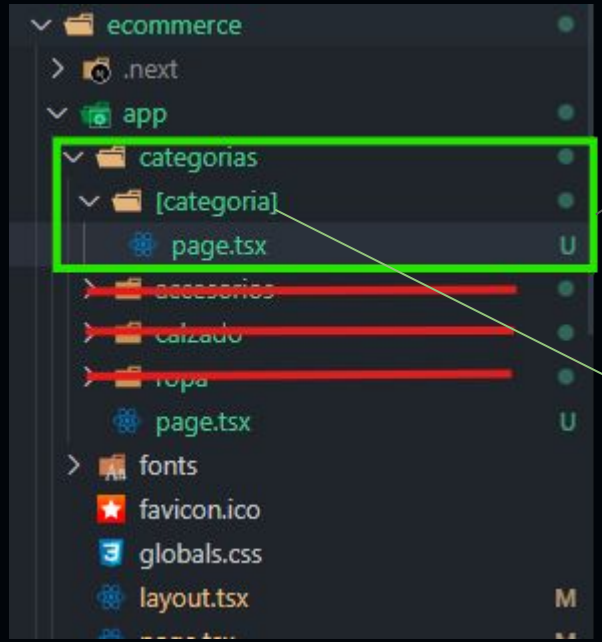
Cada ruta tiene su propia carpeta, y esto funciona a la perfección… pero…

¿Qué problemas podría causar?

No me pidan png, soy backend

Crombie

Rutas dinámicas: Uso correcto



- La carpeta [categoría] nos permite recibir valores dinámicos
- No necesitamos crear una carpeta para cada categoría
- Permite tener una única página de categoría que cambie sus valores dinámicamente

Nuestro componente de categoría leerá el parámetro con el nombre de la carpeta, y nos permitirá saber qué valor hay luego de /categoría/

ecommerce > app > categorias > [categoria] > page.tsx > ...

Complexity is 3 Everything is cool!

```tsx
export default async function Categoria({
  params,
}: {
  params: Promise<{ categoria: string }>;
}) {
  const categoria = (await params).categoria;
  return <div>Viendo {categoria}</div>;
}
```

No me pidan png, soy backend

Crombie

Preguntas?