🚀 Meu Projeto Next.js

📦 PASSO 1 — Criar um projeto com Next

Abre o terminal na pasta onde queres adicionar a pasta com o teu projeto (para adicionar projetos react nunca deve ser feito numa cloud: google drieve, oneDrive, ...):

```
npx create-next-app@latest my-next-project --tailwind --no-app --eslint --
src-dir --import-alias "@/*"
```

- O `npx create-next-app@latest` — Cria o projeto com a versão mais recente do Next.js
- O `my-next-project` — cria uma pasta chamada 'my-next-project' e gera o projeto lá dentro (podes escrever outro nome qualquer para a pasta).
- O `--tailwind` — instala o Tailwind CSS automaticamente (framework de CSS).
- O `--no-app` — força a usar o Pages Router (pasta pages/), não o novo app/ router.
- O `--eslint` — instala o ESLint que aponta erros/"más práticas" do teu código automaticamente.
- O `--src-dir` — divide o código na pasta src/ (ex.: src/pages, src/styles) para melhor organização.
- O `--import-alias "@/*"` — cria um @ para apontar para src/, tornando os imports mais limpos (ex.: '@/components/Button' ao invés de '../../components/Button').

Atenção: eu criei a app com isto tudo porque quis! Voces apenas precisam dos dois primeiros para criar uma app com next, eu é que gosto de instalar já com tudo o que sei que vou precisar.

Depois:

- ✔ Would you like to use TypeScript? … No
- ✔ Would you like to use Turbopack? (recommended) … No

Mais uma vez eu aqui escolhi `não` nas duas opções porque quis, o Typescript porque ainda não demos (mas mais para a frente vamos dar e vai ser muito útil!) e o Turbopack porque não vamos precisar.

Depois de criares a tua app executa:

```
cd my-next-project
npm run dev
```

O primeiro comando para entrares dentro da pasta do projeto (se tiveres dado outro nome á pasta faz `cd outro-nome`) O `npm run dev` é o comando que deves executar sempre para correr o projeto. Sempre que quiseres parar de correr faz Control + C no terminal.

📁 PASSO 2 — Entender a Estrutura de Ficheiros

Agora que criaste o projeto, vamos perceber para que serve cada ficheiro e pasta:

🔧 Ficheiros de Configuração (Raiz do Projeto)

`package.json`

- **O QUE É:** O "bilhete de identidade" do teu projeto
- **PARA QUE SERVE:**
  - Lista todas as dependências (bibliotecas) que o projeto precisa
  - Define os comandos que podes correr (`npm run dev`, `npm run build`, etc.)
  - Tem informações básicas do projeto (nome, versão)

`next.config.mjs`

- **O QUE É:** Configurações específicas do Next.js
- **PARA QUE SERVE:** Personalizar como o Next.js funciona (redirecionamentos, otimizações, etc.)

`eslint.config.mjs`

- **O QUE É:** Configurações do ESLint
- **PARA QUE SERVE:** Define que regras o ESLint deve seguir para apontar erros/más práticas no código

`jsconfig.json`

- **O QUE É:** Configurações do JavaScript para o VS Code
- **PARA QUE SERVE:**
  - Faz o VS Code perceber a estrutura do projeto
  - Define o import alias `@/*` para apontar para `src/`

`postcss.config.mjs`

- **O QUE É:** Configurações do PostCSS
- **PARA QUE SERVE:** Processa o CSS (especialmente para o Tailwind funcionar)

📂 Pastas Principais

`src/` (Source - Código Fonte)

Esta é a pasta onde vais escrever o teu código:

`src/pages/`

- **O QUE É:** Onde crias as páginas da tua aplicação
- **COMO FUNCIONA:** Cada ficheiro `.js` torna-se automaticamente uma página

🔥 Ficheiros Especiais (não são páginas normais):

`_app.js`

- **O QUE É:** O componente "pai" de TODAS as páginas
- **PARA QUE SERVE:**
  - Executa em todas as páginas (como um layout global)

    - Importa estilos globais (`globals.css`)
    - Aqui podes adicionar coisas que queres em todas as páginas (ex: navbar, footer)
- ⚠ **NÃO É UMA PÁGINA:** Os utilizadores não acedem a `/_app`
- 🤔 **MAS POSSO FAZER ISSO NO index.js?**
    - **SIM, mas só na página inicial!** Se criares `about.js`, terias que repetir o código
    - **VANTAGEM:** Com `_app.js` escreves uma vez, aparece em TODAS as páginas automaticamente

`_document.js`

- **O QUE É:** Template HTML "esqueleto" da aplicação
- **PARA QUE SERVE:**
    - Define a estrutura HTML básica (`<html>`, `<head>`, `<body>`)
    - Aqui podes adicionar meta tags, fontes do Google, ou scripts que carregam antes de tudo
- ⚠ **RARAMENTE PRECISAS DE MEXER AQUI**
- ⚠ **NÃO É UMA PÁGINA:** Os utilizadores não acedem a `/_document`
- 🚨 **Para já NÃO mexer!**
- O Next.js já configurou tudo perfeitamente
- Só mexerias aqui mais tarde para coisas avançadas (Google Analytics, fontes especiais, etc.)
- **Foca-te no `index.js` e outras páginas normais!**

📄 Páginas Normais (que os utilizadores acedem):

`index.js`

- **O QUE É:** A página inicial do teu site (`/`)
- **PARA QUE SERVE:**
    - É a primeira página que os utilizadores veem
    - Aqui vais criar o conteúdo da homepage
- 👉 **ESTE É O FICHEIRO QUE VAIS MEXER MAIS!**

Outras páginas que podes criar:

- `src/pages/about.js` → `/about` (página sobre)
- `src/pages/contact.js` → `/contact` (página de contacto)
- `src/pages/services.js` → `/services` (página de serviços)
- E assim por diante...

`src/pages/api/` ⚠ NÃO VAMOS USAR PARA JÁ

- **O QUE É:** Onde crias APIs (backend)
- **EXEMPLO:** `src/pages/api/hello.js` → `/api/hello` (API endpoint)
- ⚠ **IMPORTANTE:** Para já **NÃO vamos mexer com backend**, por isso **ignora esta pasta** e o ficheiro `hello.js` que está lá dentro. Vamos focar-nos apenas no frontend (páginas React)!

`src/styles/`

- **O QUE É:** Onde colocas os ficheiros CSS
- **`globals.css`:** Estilos globais que se aplicam a toda a aplicação

`public/`

- **O QUE É:** Ficheiros estáticos (imagens, ícones, etc.)
- **ACESSO:** Podes aceder diretamente via URL (ex: `public/logo.png` fica apenas `/logo.png`)

`node_modules/`

- **O QUE É:** Todas as dependências/bibliotecas instaladas
- ⚠ **NUNCA MEXER:** Esta pasta é gerida automaticamente pelo npm e nunca deve ser mexida!!

`.next/`

- **O QUE É:** Pasta gerada automaticamente pelo Next.js
- ⚠ **NUNCA MEXER:** Contém o código compilado/otimizado e nunca deve ser mexida!!

📋 Comandos Úteis

```
# Correr o projeto em desenvolvimento
npm run dev

# Compilar o projeto para produção
npm run build

# Correr o projeto compilado
npm run start

# Verificar erros de código
npm run lint
```

✨ Próximos Passos

1️⃣ Editar a página inicial e testar Tailwind

Para personalizar a homepage e testar se o tailwind está a funcionar, modifica o `src/pages/index.js` para isto:

```
export default function Home() {
  return (
    <div className="min-h-screen bg-blue-500 flex flex-col items-center
justify-center">
      <h1 className="text-7xl font-bold text-white">Hello! 👋</h1>
    </div>
  )
}
```

RESULTADO: Uma página com fundo azul e um texto a branco a dizer "Hello! 👋 " **SE O TAILWIND ESTIVER A FUNCIONAR:** Vês os estilos aplicados (cores, espaçamentos, etc.)

2️⃣ Criar pasta components e primeiro componente

Cria a pasta `src/components/` e adiciona o teu primeiro componente `PrimeiroComponente.jsx` com isto:

```
export default function PrimeiroComponente() {
  return (
    <div className="bg-green-100 border border-green-400 text-green-700
px-4 py-3 rounded my-4">
      <h2 className="text-xl font-semibold">🎉 O meu primeiro componente!
</h2>
      <p>Este componente foi criado em
src/components/PrimeiroComponente.js</p>
    </div>
  )
}
```

Agora chama o componente no `index.js`. Adiciona o import no topo do ficheiro:

```
import PrimeiroComponente from '@/components/PrimeiroComponente'
```

E usa o componente dentro do return:

```
import PrimeiroComponente from '@/components/PrimeiroComponente'

export default function Home() {
  return (
    <div className="min-h-screen bg-blue-500 flex flex-col items-center
justify-center">
      <h1 className="text-7xl font-bold text-white">Hello! 👋</h1>
      <PrimeiroComponente />  {/* ← O teu componente aqui */}
    </div>
  )
}
```

RESULTADO: Vês a caixa verde do componente aparecer na página! ✅ **SUCESSO:** Se vires o componente, já sabes criar e usar componentes! Agora sempre que quiseres adicionar um componente novo crias o ficheiro dentro da pasta `components` e fazes import do ficheiro no index.js (para já é a única página que temos) para poderes chamá-lo e usá-lo.

📚 PASSO 3 — Componentes de Estudo React

AGORA JÁ SABES TUDO SOBRE COMO MEXER NA TUA APP NEXT! VAMOS CRIAR COMPONENTES PARA PRATICAR TODA A MATÉRIA DE REACT! VAMOS AGORA A UM RESUMO DA MATÉRIA DADA DE REACT? APRENDEMOS A:

- Criar um componente normal `ComponenteNormal.jsx` (já está feito)
- Criar um componente com uma prop `ComponenteComUmaProp.jsx`
- Criar um componente com várias props `ComponenteComVariasProps.jsx`
- Criar um componente com condicional (ternário e &&) `ComponenteComCondicional.jsx`
- Usar o map dentro de um componente (com chave). `ComponenteComMap.jsx`
- Event handler dentro de um componente (function ou arrow function, com callback ou sem, passar parametros como argumento ou não) `ComponenteComEventos.jsx`
- Usar o useState dentro de um componente `ComponenteComState.jsx`
- Usar o useEffect dentro de um componente `ComponenteComUseEffect.jsx`
- Criar um formulário num componente com formik `ComponenteComFormularioFormik.jsx`

Cria cada um destes componentes na pasta `src/components/` para praticares todos os conceitos.