

Next.js Full Beginner Course

[2023] TypeScript

→ Next.js

→ Framework em cima de React

→ Escreve-se ~~no~~ os componentes e assim tem a bela interface de usuário declarativa

→ Pode-se instalar vários pacotes

→ ! de React é que no Next.js é possível obter, para cada página individual, se queremos carregar os dados do lado do cliente ou do lado do servidor

→ Em React os dados só são carregados do lado do cliente

→ Basicamente apenas uma única página da web, uma página HTML que não contém quaisquer dados dinâmicos (dados do banco), estes dados são carregados depois executando JavaScript.

→ Abordagem popular, porque faz sites parecerem aplicativos modernos, como aplicativos móveis, onde a página carrega praticamente instantaneamente e então temos algum tipo de controle gerador de progresso enquanto nós esperamos por mais dados para carregar

→ **Problema:** O motor de busca Chrome obtém esta página HTML vazia e não executa qualquer JavaScript, ou seja, as páginas que seu site não

nenhum underbrace no Google ou em qualquer motor de busca, o que é terrível se o SEO é importante para o seu site. Hoje em dia, os rastreadores do Google podem executar Js e underbrace as páginas do site, mas outros rastreadores não.

mas demanda para atualizar as páginas

→ Outros benefícios do Next.js

- Otimizações automáticas de imagem
 - Coloca a imagem no tamanho adequado.
 - Adiciona carregamento lento
- Sistema integrado de roteamento → != React que usa pacotes para navegar entre páginas diferentes
- API routes
- Funções Serverless para poder construir todo o nosso back-end em next.js
- Serverless é um pouco diferente de Anonymous
- Servidor Express

→ Criando um projeto

- rode no terminal: `npx create-next-app@latest`
- Dê um nome para o projeto
- Escolha usar TypeScript, porque JavaScript não tem uma digitação forte, o que deixa difícil construir projetos maiores com ele.
- Escolha usar ESLint, porque nos mostra alguns erros e enganos no nosso código.
- Escolha não usar Tailwind CSS
- Escolha não usar 'src' directory, porque é bom manter o padrão de seleção, que não é esse.
- Escolha usar "App Router"
↳ não

→ Configure o "import alias" com "@/*", porque assim
começa com a vantagem de sempre apontar para a
pasta raiz.

→ `tsconfig.json`: Contém todas configurações do TS

→ `package.json`: Onde ficam as dependências

→ Novas dependências

→ No terminal do VS Code digite:

→ `npm i bootstrap react-bootstrap nextjs-progressbar`

→ Entend

→ Para rodar o projeto: `npm run dev`

→ Clique no link depois de "url:" segurando o cursor

→ Estrutura de páginas e Proteamento

→ As páginas ficam dentro da pasta "pages"

→ Nome dos arquivos incluem a URL

→ Arquivo "index.tsx" é a home

↳ (é um arquivo com
o nome "search" dentro
de "pages")

Instale a extensão "Simple React Snappers"

→ No arquivo "search" digite "sfc" para criar um componente

↳ O componente não precisa ter o mesmo nome do
arquivo

↳ Dentro de "return" coloque o HTML

→ No home:

- Retire todo o HTML dentro da tag "main"
- Retire o "className" da tag main
- O "Inter({...})" é uma classe de fonte do google
- Quando sera usado a fonte é baixada para seu servidor local, o que deixa mais rápido e muito mais prático ao seu usuário, porque não há um pedido aos servidores da google, e qual envia o IP do usuário.

→ Para aplicar algo em todos os arquivos mexto no arquivo - app.tsx

→ "<Component (... pageProps) />" renderiza todo o aplicativo

→ É substituído por outras páginas enquanto navegamos

→ Pode colocar coisas que não são comuns em cada página, como um roteiro

→ Coloca ~~(...)~~ parênteses enrolado de "<comp..." e depois também o enrolado com uma <div>

→ Recorta o "const Inter..." da "index.tsx" e cole em cima de "export..." da "seo - app.tsx"

→ Para tornar ela global coloque na <div> "className = {Inter.className}"

* para baixar outras fontes do google é só ir no "import type { Inter, ... } from..."

↳ aperta ctrl + enter que ele sugere outras

→ Retire a Tag <head> do "index.tsx" e coloque no "app.tsx"

em cima da <div> "<Component ... />"

↳ se a página não tiver seu próprio head, vai usar esse

- Arquitetura da página fica em: `App - document.TSX`
- O "global.css" só pode ser importado em "app.tsx"
- Pode apagar "Home.module.css" porque é o estilo da página Home que não vamos usar

→ Retirando os clichês do "global.css"

→ max-width

→ border-radius

→ --primary-glow } brilhos

→ --secondary-glow

→ Tudo alusivo de "--secondary-glow" até a "3"

→ @media (prefers-color-scheme...) → porque com Bootstrap não precisa de mudanças

→ Substitua "overflow-x: hidden" por "min-height: 100vh"

→ @media...

→ -app.tsx

→ Envolver "<Component...>" com <Container> do Bootstrap

→ Verifique se no import do Container está "react-bootstrap" exatamente, se não estiver podeclar depois.

→ Cria o arquivo "App.module.css" na pasta "style" e importa em "-app.tsx"

↳ import styles from '@/styles/App.module.css'

↳ pode ser qualquer nome

→ Coloque em <Container> → <Container className={styles.pageContent}>

classe css

→ Importe o css do Bootstrap com "import 'bootstrap/dist/css/bootstrap.min.css';"

↳ <https://github.com/react-bootstrap/getting-started/introduction>

X27 → Coloca o import do css do Bootstrap quebrado, porque
existem problemas sobreescrita-lo com nosso css.

→ News API

→ API de notícias

→ newsapi.org

→ Têm que criar conta

→ Minha API Key: [REDACTED]

→ Crie na raiz do projeto um arquivo .env.local

↳ nele contém
as variáveis de
projeto

→ Dentro do .env.local coloque NEWS_API_KEY= sua key

→ Em newsapi.org/docs/endpoints/top-headlines mostra o que ele retorna quando fazemos uma busca pelas últimas notícias

→ Crie na raiz uma pasta "models" e dentro "NewsArticles.ts"

→ export interface NewsArticle {

author: string,

} pego os dados que a

} API vai me mandar

e coloco os

mais
tipos que

ss

export interface NewsResponse {

articles: NewsArticle[],

} além dos artigos, a API
mandou um "status" e
o total de artigos

}

→ index.tsx

→ Novamente a tag <head>, sobreescrivendo a da
app.tsx, mas só sobre escreve o título

tilibra

→ Para ter certeza que os <title> não se substituir, adicionamos "key = "title"" no <title> da index.js da app

→ Exporte a função getServerSideProps, que o Next.js usa para pegar dados do ~~server~~ server render quando a página carrega, e armazene em uma variável.
export const getServerSideProps: GetServerSideProps<>
em cima de "export default ..."

→ Para criar o tipo que vamos passar para o componente criamos, em cima do "export const...":

```
interface BreakingNewsPageProps {  
    newsArticles: NewsArticle[],  
}
```

→ Adiciona dentro do () de "export default...":
{newsArticles}: BreakingNewsPageProps

→ Volta no "export const..." e incrementa:

```
export const getServerSideProps: GetServerSideProps<BreakingNewsPageProps> = async () => {  
    ;  
}
```

}

* Se uma chave do .env for criada com "NEXT_PUBLIC_" na frente, ela aparecerá ~~no~~ no front, se não, será protegida e não podemos usá-la

→ Adicione a pasta "components" na raiz do projeto e crie o arquivo "NewsArticleEntry.jsx"

* Em NewsArticleEntity, ~~tex~~ para não ter que usar "article" na frente de todos os propririedades dele, podemos colocar article: { ... }

Propiedades que vamos usar, tipo "title"

→ Search.tsx

→ Em uma da "Resets return" coloque:

```
const [ searchResults, setSearchResults ] = useState( )
```

variáveis de estado para manter os dados que vamos conegar ↳ configuração dos resultados

```
const [searchResults, setSearchResults] = useState<NewsArticle[]>[  
  null>(null);
```

Só com isso e sem o argumento de tipo explícito "< >" o TypeScript não sabe que valor "SearchResults" deve ter

argumento de tipo explícito
pode ser NewsArticle ou null

→ conegamento dos resultados de pesquisa

```
const [searchResultsLoading, setSearchResultsLoading] = useState(false);
```

↳ boolean

```
async function handleSubmit(e: FormEvent<HTMLFormElement>){  
    e.preventDefault();  
    const { value: name } = document.querySelector('input')!
```

* react formhooks é um pacote que ajuda quando se tem vários campos

descobrir
a API key
então é bom
fazer **tilde**
na pasta "app"-mento.
para pegar
os dados do formulário
const formDara = new FormData(e.target as HTMLFormElement);

→ /api/search-news.ts

const searchQuery = req.query.q?.toString(); → pega o que foi mandado na requisição

Rotas dinâmicas

↳ caminho: /category/business

↳ vai ser
uma pasta

↳ como isso sempre vai estar mudando,
não compensa criar um arquivo para
cada categoria, então o arquivo
criado fica assim:
[category].tsx

<></> - se chama anti-fragmento

→ [category].tsx

→ getStaticProps

→ Com isso os dados serão buscados somente 1x, quando
compila o código e não quando carrega a página
→ gera um HTML estático

export const getStaticProps: GetStaticProps<CategoryNewsPageProps> = async ({ params }) => { ... }

↳ retirado do "context" que é um pacote-
zinho com muitas coisas, mas a URL vem
do "params" então colocamos ele entre
{} para usar
↳ para desestruturar o context

→ getStaticPaths

→ Usado para criar caminhos estáticos. Neste caso preci-
samos usar isso, porque para usar o getStaticProps, ele
precisa saber quais categorias já existem, afinal, ele
não é executado quando carrega a página.

slugs = caminhos relativos

fallback → se a pessoa inserir outra URL diferente do array definido, mostrará uma página 404 ("não encontrado")

* No lugar do array poderia ser uma consulta para trazer todos os categoriais

→ Desvantagem: endereços estáticos não são atualizados após a compilação

→ Incremental static regeneration

→ Atualiza os dados estáticos com um tempo pré-definido quando um usuário abre a página

→ Volta no return do `getStaticProps` e adiciona: `revalidate: 5 * 60`:

↳ faz isso pq tem que ser em seg.

→ `const router = useRouter();`, → possibilidade pegar parâmetros da URL

`router.query.category?.toString()`, → pegando a categoria

404/500 páginas

→ é só criar os arquivos em "pages" com o código do erro, tipo "404.tsx", e personalizar

* `getServerSide` ou `getStaticProps` caso deem problemas, façam diretamente na página 500.

→ `next/link` → Atualiza a página, mas não a NavBar, o menu

tilibra

→ `next/image` → ~~remove~~ otimiza a imagem

↳ para funcionar tem que permitir a URL em: `next.config.js` de onde vem a img