TUTORIAL

Tutorial Explicado: Código page.tsx

Este arquivo é escrito em TypeScript com React e é usado para construir uma página de site. Vamos explicar tudo passo a passo, como se fosse para quem está começando.

1. Importações

Logo no início, o código importa algumas coisas:

```
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
```

Explicação:

import é como se fosse "chamar" algo de outro lugar.

Aqui ele está trazendo componentes prontos: Button, Input, Label.

Esses componentes vêm de uma pasta chamada components/ui/.

Eles são usados para construir botões, campos de texto e etiquetas (labels) no site.

Imagine: em vez de criar um botão do zero, você importa um pronto.

2. Função Principal (Page)

Depois das importações, o código cria uma função chamada Page:

```
export default function Page() {
  return (
    <div className="flex items-center justify-center h-screen">
      <div className="w-full max-w-md p-8 space-y-6">
```

```
      {/* Conteúdo vai aqui */}
    </div>
  </div>
 )
}
```

Explicação:

A função Page é um componente React. É o que vai ser mostrado na tela.

export default significa que esse é o arquivo principal que vai ser usado em outro lugar.

Dentro do return:

 é uma caixa na tela (tipo uma "área").

A primeira centraliza tudo no meio da tela (flex, items-center, justify-center, h-screen).

A segunda define o tamanho do conteúdo (w-full, max-w-md, p-8, space-y-6).

Essas classes (flex, w-full, p-8 etc.) são do Tailwind CSS — um jeito rápido de colocar estilo.

3. O Conteúdo da Página

Dentro da segunda , está o que aparece na tela:

```
<div>
  <h1 className="text-2xl font-bold">Login</h1>
  <form className="space-y-4">
    <div className="space-y-2">
      <Label htmlFor="email">Email</Label>
      <Input id="email" type="email" required />
    </div>
    <div className="space-y-2">
      <Label htmlFor="password">Password</Label>
```

```
      <Input id="password" type="password" required />
    </div>
    <Button type="submit" className="w-full">
      Login
    </Button>
  </form>
</div>
```

Vamos quebrar isso:

<h1>: Título principal. Escrito "Login".

<form>: Um formulário — onde o usuário vai preencher as informações.

Dentro do <form>, temos dois campos de texto:

Um para o email.

Outro para a senha.

Cada campo tem:

Um Label (nome do campo).

Um Input (caixinha para digitar).

No final, tem um botão chamado "Login".

O formulário está organizado com espaços (space-y-4, space-y-2) para não ficar tudo colado.

4. O Que Esse Código Faz?

Esse código cria uma página de login simples, onde:

Você digita seu email e sua senha.

Clica no botão "Login".

IMPORTANTE:

Esse código ainda não faz o login de verdade. Ele só mostra a tela bonita.

Para realmente logar (tipo mandar dados para um servidor), teria que adicionar mais código.

Resumo Visual (Simplificando)

Tutorial parte 2 - Projeto Pikachu (Mew)

```
'use client'
import Link from "next/link";
import Image from "next/image";
```

- `'use client'`: avisa ao Next.js que essa página será executada no navegador (é interativa).
- `import`: é como "chamar" algo de outro lugar.
- `Link`: permite fazer links entre páginas no Next.js.
- `Image`: carrega imagens de forma otimizada (melhor para desempenho e SEO).

```
const dogs = [
  "/dogs/dog1.jpg",
  "/dogs/dog2.jpg",
  "/dogs/dog3.jpg",
  "/dogs/dog4.jpg",
  "/dogs/dog5.jpg",
  "/dogs/dog6.jpg",
];
```

Aqui temos uma lista com os caminhos das imagens dos cachorros.

Essas imagens devem estar na pasta `public/dogs/` do seu projeto.

```
export default function Doacoes() {
  return (
    <main className="px-4 sm:px-6 lg:px-8 py-10 max-w-7xl mx-auto">
      {/* conteúdo */}
    </main>
  );
}
```

`export default function Doacoes()` cria o componente principal da página.
`<main>`: é a parte principal da tela. Usa Tailwind CSS para colocar espaçamento (padding), largura e centralizar.

```
<h1 className="text-3xl font-bold text-center mb-10">DOAÇÕES</h1>
```

Mostra o título "DOAÇÕES".
`text-3xl`: deixa o texto grande.
`font-bold`: texto em negrito.
`text-center`: centraliza.
`mb-10`: dá um espaço embaixo (margin-bottom).

```
<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-8 mb-12">
  {dogs.map((src, index) => (
    <div
      key={index}
      className="rounded-xl bg-gray-400 p-2 flex justify-center items-center shadow-lg"
    >
      <Image
        src={src}
        alt={`Cachorro ${index + 1}`}
        width={300}
        height={300}
        className="rounded-md object-cover w-full h-full max-h-[300px]"
      />
    </div>
  ))}
</div>
```

A `<div>` usa uma **grade (grid)** para organizar as imagens em colunas:

- `1 coluna` em telas pequenas.
- `2 colunas` em telas médias.
- `3 colunas` em telas grandes.

`dogs.map(...)`: percorre cada imagem da lista e cria um card para ela.

Cada card tem:

- Fundo cinza (`bg-gray-400`), cantos arredondados (`rounded-xl`), sombra (`shadow-lg`) e padding.
- A imagem é exibida com tamanho e corte ajustado para não deformar (`object-cover`).

```
<div className="flex justify-center">
  <Link href="/">
    <button className="bg-blue-600 text-white px-6 py-3 rounded hover:bg-blue-700 transition">
      Voltar para Home
    </button>
  </Link>
</div>
```

`<Link href="/">`: cria um botão que leva de volta para a página inicial.

O botão tem:

- Cor de fundo azul (`bg-blue-600`), texto branco, padding e bordas arredondadas.
- `hover:bg-blue-700`: muda a cor ao passar o mouse.
- `transition`: deixa a troca de cor mais suave.

Tutorial parte 3 - Projeto Pikachu (Mew)

```
export async function GET() {
  const dogs = [
    { id: 1, name: "Rex", imageUrl: "https://placedog.net/400/400?id=1", breed: "Labrador" },
    { id: 2, name: "Bella", imageUrl: "https://placedog.net/400/400?id=2", breed: "Poodle" },
    { id: 3, name: "Thor", imageUrl: "https://placedog.net/400/400?id=3", breed: "Beagle" },
    { id: 4, name: "Luna", imageUrl: "https://placedog.net/400/400?id=4", breed: "Bulldog" },
    { id: 5, name: "Max", imageUrl: "https://placedog.net/400/400?id=5", breed: "Golden Retriever"
    { id: 6, name: "Mel", imageUrl: "https://placedog.net/400/400?id=6", breed: "Pastor Alemão" }
  ];

  return new Response(JSON.stringify(dogs), {
    status: 200,
    headers: { "Content-Type": "application/json" },
  });
}
```

- Define uma função `GET()` para responder a requisições do tipo GET.
- Cria uma lista de cães com `id`, `name`, `imageUrl` e `breed`.
- Retorna os dados como JSON na resposta da API.
- Define o cabeçalho da resposta como `"Content-Type": "application/json"` para que o front-end reconheça o tipo de dado.

```
import Link from 'next/link';
import Image from 'next/image';
import { useEffect, useState } from 'react';
```

- Link: cria links entre páginas do site.
- Image: carrega imagens de forma otimizada (mais leve e rápida).
- useEffect: executa algo quando a página carrega.
- useState: guarda informações enquanto o usuário usa a página.

```
const [dogs, setDogs] = useState<Dog[]>([]);
```

- Cria um espaço (estado) para guardar a lista de cachorros.
- O tipo `Dog` tem: `id`, `name`, `imageUrl`, `breed`.

```
useEffect(() => {
  async function fetchDogs() {
    const res = await fetch('/api/dogs');
    const data = await res.json();
    setDogs(data);
  }
  fetchDogs();
}, []);
```

- Quando a página é carregada, a função `fetchDogs()` é chamada.
- Ela busca os dados na URL `/api/dogs`.
- Depois, guarda os dados na variável `dogs` com `setDogs()`.

```
dogs.map((dog) => (
  <div key={dog.id} className="...">
    <Image src={dog.imageUrl} alt={`Cachorro ${dog.name}`} width={300} height={300} />
  </div>
))
```

- `dogs.map(...)` percorre cada cachorro da lista e cria um card.
- Cada card tem:
    - Imagem do cachorro (`Image`)
    - Fundo cinza, cantos arredondados, sombra, padding
    - Tamanho fixo com corte ajustado (`object-cover`)

```
<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3">
```

Responsividade

1 coluna em telas pequenas (celular)
2 colunas em telas médias (tablet)
3 colunas em telas grandes (computador)

```jsx
<Link href="/">
  <button className="bg-blue-600 text-white ...">Voltar para Home</button>
</Link>
```

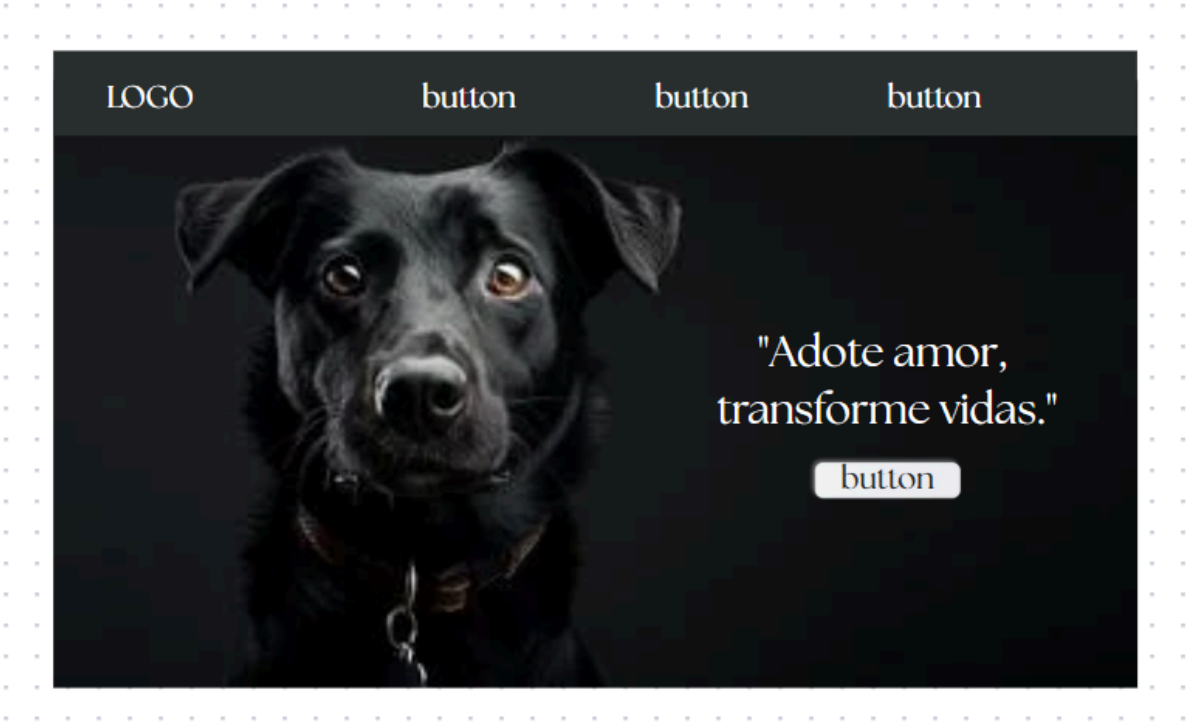
Cria um botão com:

- Cor azul
- Texto branco
- Arredondado
- Animação ao passar o mouse (`hover:bg-blue-700` e `transition`)

Escolhemos desenvolver o **Amigo de Patas** porque acreditamos no poder transformador da adoção de animais — tanto na vida dos cães quanto na das pessoas que os acolhem. Também percebemos que muitas iniciativas de adoção ainda não têm a visibilidade que merecem, seja por falta de recursos ou de alcance na internet. Nosso site é uma forma prática e afetiva de usar a tecnologia para um propósito social, reunindo informações de maneira organizada, acessível e convidativa para toda a comunidade.

Além disso, é uma maneira de unir nossa formação acadêmica em tecnologia à necessidade de impacto real na sociedade, promovendo o bem-estar animal e reforçando o compromisso de cidadãos mais conscientes e solidários.

Wireframe da página inicial



Wireframe segunda página

Inspirações: