Leonardo de Paula Fraga Geronymo, RA: 10390279

Kemuel Áquila de Matos, RA: 10322341

Projeto Pikachu

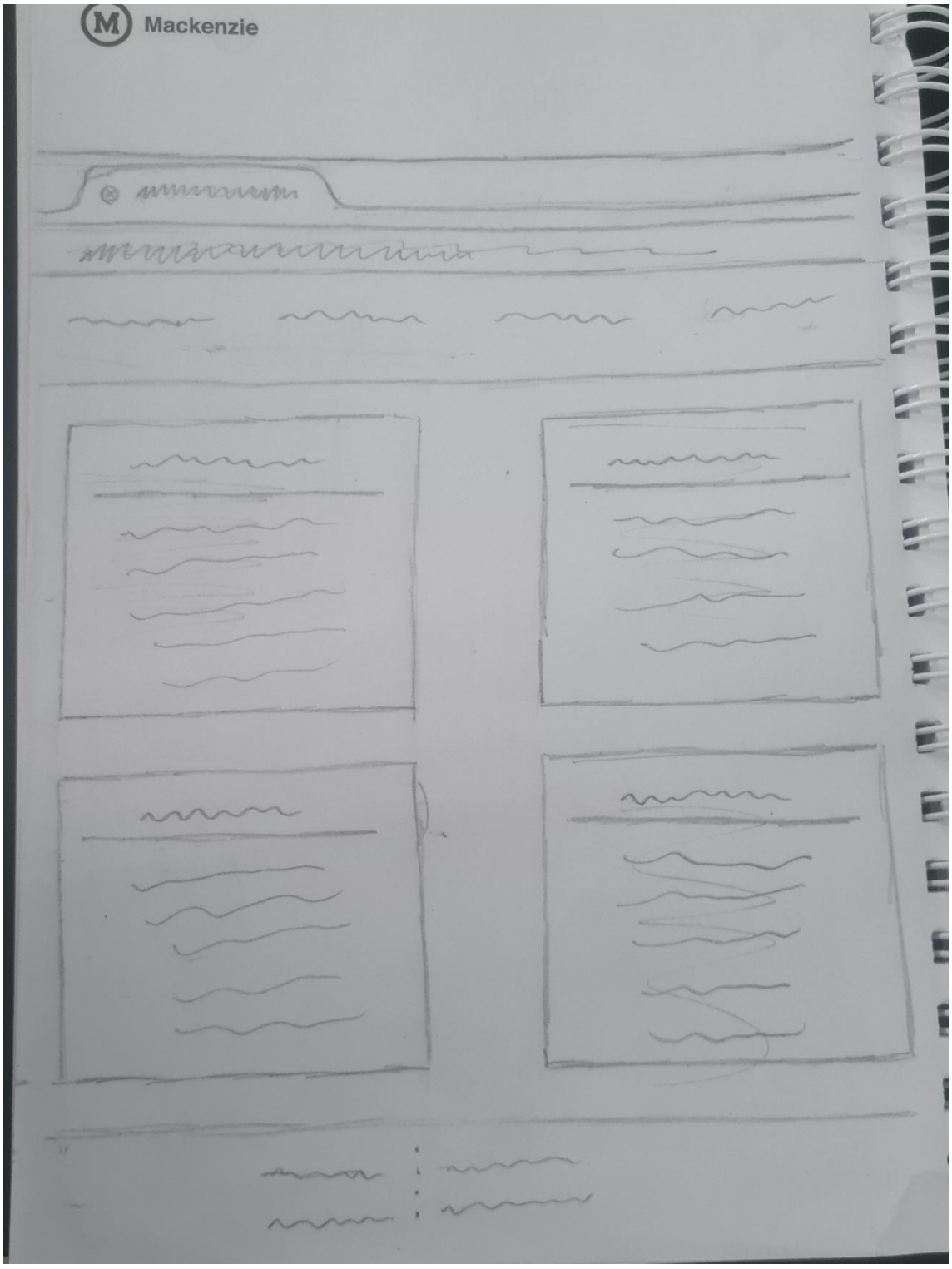
Web Mobile 02J12

Tutorial

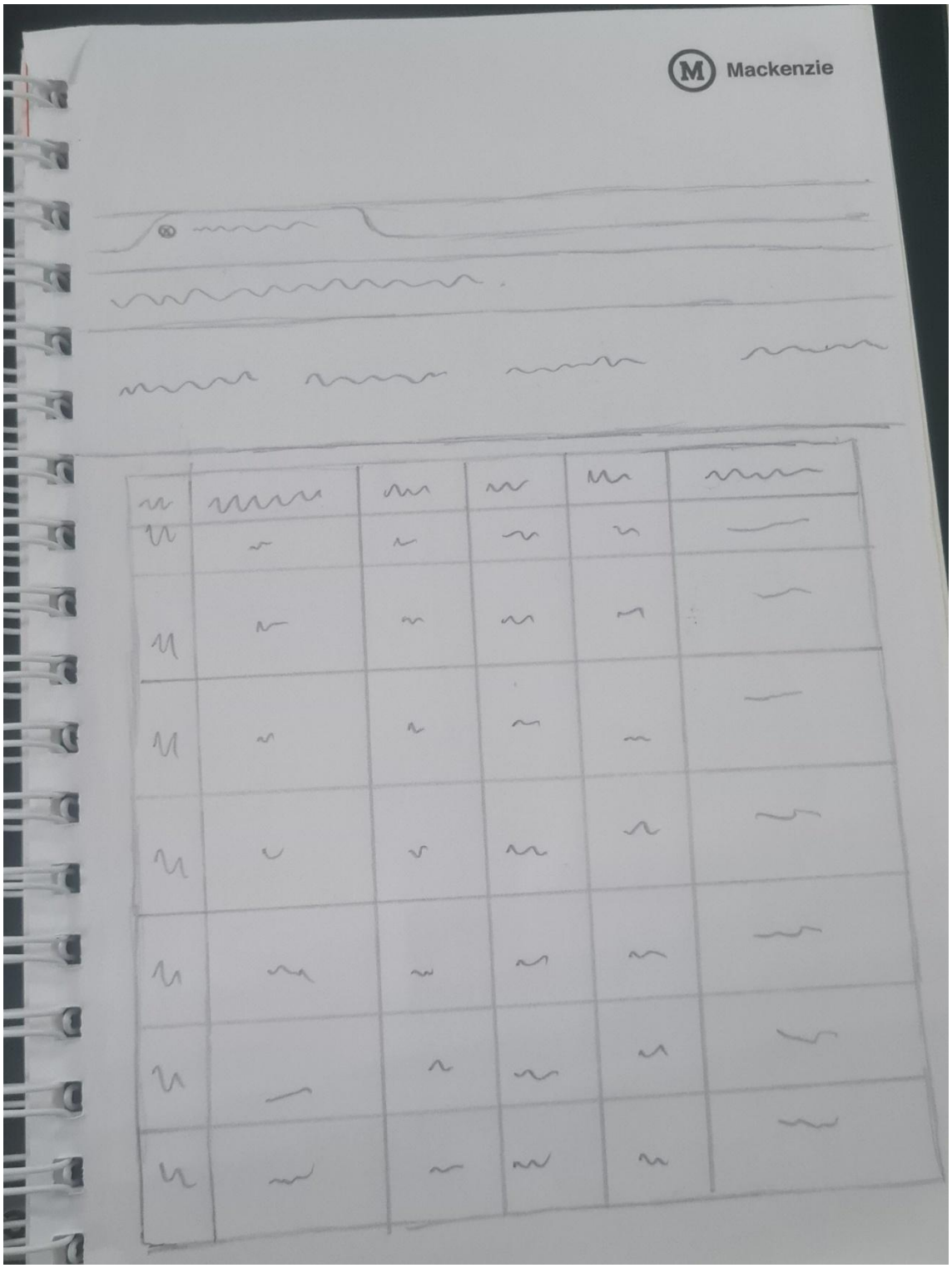
Este projeto tem como objetivo um site desenvolvido em NextJS, com o tema o campeonato Brasileirão Série B, no menu principal teremos um header onde, teremos 4 opções: Menu, Brasileirão B, Artilharia e Tabela cada um desses será um link, onde será redirecionado para cada uma das paginas referentes ao nome, também teremos 3 cards com cada um deles explicando um pouco sobre um tema do campeonato, e no rodapé teremos a identificação dos integrantes com nome e RA.

O restante das páginas serão feitos no mesmo modelo, onde a um header semelhante ao do menu, afim de que o usuário possa voltar as páginas ou avançar para as seguintes, já no body será apresentado uma tabela do campeonato, em outra página será mostrado os maiores artilheiros deste ano e naoutra informações simples da competição, as informações seram expostas através da “ API FUTEBO L”: https://www.api-futebol.com.br/ .

Página Principal:



Páginas Campeonatos:



Página Principal

```
import Link from 'next/link';
```

Está importando o componente Link da biblioteca next/link, que é usado para criar links de navegação entre páginas dentro do projeto Next.js

```
export default function Home() {
```

Define e exporta a função Home, que representa a página principal.

```
const cardStyle = {
  border: '1px solid #ccc',
  padding: '1rem',
  borderRadius: '8px',
  maxWidth: '400px',
  margin: '1rem',
  flex: '1 1 400px',
};
```

Estilo dos "cards" (caixas de conteúdo). Define borda, espaçamento interno e externo, arredondamento dos cantos e ajuste de layout responsivo com flex.

```
const titleStyle = {
  fontWeight: 'bold',
  marginBottom: '0.5rem',
};
```

Estilo aplicado aos títulos (h2) dos cards. Define negrito e margem inferior.

```
<nav style={{ display: 'flex', justifyContent: 'space-around', padding: '1rem 0', borderBottom: '1px solid #ccc', flexWrap: 'wrap' }}>
  <div>MENU</div>
  <Link href="/brasileirao">BRASILEIRÃO B</Link>
  <Link href="/tabela">TABELA</Link>
```

<Link href="/artilharia">ARTILHARIA</Link>

</nav>

Um menu de navegação com estilo flexível e links para outras páginas.

flexWrap: 'wrap' permite que os itens quebrem linha em telas menores.

<main style={{ display: 'flex', justifyContent: 'space-around', padding: '2rem', flexWrap: 'wrap' }}>

Container para os três cards, com layout em Flexbox que se adapta à tela.

<section style={cardStyle}>

<h2 style={titleStyle}>Formato</h2>

** ... **

</section>

Cada section representa um card com conteúdo específico:

Card 1: Formato

- História da Série B e explicação sobre o sistema de disputa.
- Destaque para o ano de 2006 com a introdução dos pontos corridos.

Card 2: Destaques e Curiosidades

- Fala sobre os clubes com mais títulos.
- Menciona recordes como o Cruzeiro em 2022 e o Corinthians em 2008.

Card 3: Maiores Artilheiros

- Lista os jogadores com mais gols em uma única edição da Série B.

Página Brasileirao

'use client';

Informa ao Next.js que este é um componente do lado do cliente.

import { useEffect, useState } from 'react';

import Link from 'next/link';

useEffect: executa código após o componente ser montado.

useState: cria uma variável de estado.

Link: permite navegação entre páginas sem recarregar.

const [campeonato, setCampeonato] = useState(null);

Cria um estado chamado campeonato que começa com null.

```
useEffect(() => {
  const fetchDados = async () => {
    try {
      const res = await fetch('https://api.api-futebol.com.br/v1/campeonatos/10', {
        headers: {
          Authorization: 'Bearer test_d669ab27a3819608318d1439cccfbf'
        }
      });
      const data = await res.json();
      setCampeonato(data);
    } catch (err) {
      console.error('Erro ao buscar dados da Copa do Brasil:', err);
    }
  };
```

```
  fetchDados();
}, []);
```

Faz uma requisição HTTP para pegar dados do campeonato com ID 10(significa serie B na API FUTEBOL).

Coloca os dados na variável campeonato.

Em caso de erro, exibe no console.

```
<nav>
  <Link href="/">MENU</Link>
  <Link href="/brasileirao">BRASILEIRÃO B</Link>
  <Link href="/tabela">TABELA</Link>
  <Link href="/artilharia">ARTILHARIA</Link>
</nav>
```

Barra de navegação com links para outras páginas da aplicação.

```
<main className="p-6 max-w-4xl mx-auto">
```

Usa classes do Tailwind CSS para estilização:

- p-6: padding
- max-w-4xl: largura máxima
- mx-auto: centraliza o conteúdo

```
{campeonato ? (
  <div>...</div>
) : (
  <p>Carregando dados do Brasileirão Série B...</p>
)}
```

Enquanto os dados ainda não foram carregados, mostra "Carregando...".

Quando campeonato estiver preenchido, exibe os dados do campeonato.

```
<img src={campeonato.logo} alt="Logo campeonato" />
<h2>{campeonato.nome_popular}</h2>
<p>Edição: {campeonato.edicao_atual?.nome_popular}</p>
<p>Fase Atual: {campeonato.fase_atual?.nome}</p>
<p>Tipo: {campeonato.tipo}</p>
<p>Status: {campeonato.status}</p>
```

Exibe o logo e as informações principais do campeonato.

Usa ?. para evitar erros se algum dado estiver undefined.

Página Artilharia

```
const [artilheiros, setArtilheiros] = useState([]);
const [loading, setLoading] = useState(true);
```

artilheiros: guarda os dados vindos da API (lista de jogadores).

loading: indica se os dados ainda estão sendo carregados.

```
 useEffect(() => {
   const fetchArtilharia = async () => {
     try {
       const response = await fetch('https://api.api-futebol.com.br/v1/campeonatos/10/artilharia', {
         headers: {
           Authorization: 'Bearer test_d669ab27a3819608318d1439cccfbf'
         }
       });
       const data = await response.json();
       console.log(data);
       setArtilheiros(data.filter(item => item !== null));
```

```
    } catch (error) {
      console.error('Erro ao buscar artilharia:', error);
    } finally {
      setLoading(false);
    }
  };

  fetchArtilharia();
}, []);
```

Esse useEffect é executado uma vez só, ao montar o componente.

Dentro dele:

- Faz uma requisição GET para a API da artilharia da Série B.
- Adiciona o token de autorização no cabeçalho.
- Transforma a resposta da API em um objeto JavaScript com await response.json().
- setArtilheiros(data.filter(item => item !== null)): salva no estado, mas remove valores nulos que podem vir da API.
- setLoading(false): informa que a requisição terminou.

```
<nav style={{ display: 'flex', justifyContent: 'space-around', padding: '1rem 0', borderBottom: '1px solid #ccc', flexWrap: 'wrap' }}>
  <Link href="/">MENU</Link>
  <Link href="/brasileirao">BRASILEIRÃO B</Link>
  <Link href="/tabela">TABELA</Link>
  <Link href="/artilharia">ARTILHARIA</Link>
</nav>
```

Barra de navegação com links para outras páginas do site.

** **

Usa classe do Tailwind CSS para estilização:

- mt-10: margin-top de 2.5rem (40px).

<h2 className="text-2xl font-semibold mb-4">Artilharia</h2>

Usa classes do Tailwind CSS:

- text-2xl: tamanho da fonte grande (2× extra-large).
- font-semibold: espessura da fonte média (semi-negrito).
- mb-4: margin-bottom de 1rem (16px).

```
{loading ? (
  <p>Carregando artilheiros...</p>
) : artilheiros.length === 0 ? (
  <p className="text-gray-400">Nenhum artilheiro disponível no momento.</p>
) : (
  <ul className="space-y-2">
    {artilheiros.map((jogador, index) => (
      <li key={index} className="bg-gray-900 p-4 rounded-lg flex items-center justify-between">
        <div>
          <p className="text-lg font-medium">{jogador.atleta.nome_popular}</p>
          <p className="text-sm text-gray-400">{jogador.time.nome_popular}</p>
        </div>
        <span className="text-xl font-bold">{jogador.gols}</span>
      </li>
    ))}
  </ul>
)}
```

- Se loading for true: mostra “Carregando...”
- Se a lista estiver vazia: mostra “Nenhum artilheiro...”
- Senão: exibe a lista dos artilheiros.

Cada item da lista mostra:

- Nome do jogador → jogador.atleta.nome_popular
- Nome do time → jogador.time.nome_popular
- Número de gols → jogador.gols

Página Tabela

```
const [tabela, setTabela] = useState([]);
const [carregando, setCarregando] = useState(true);
```

tabela: armazena os dados da API.

carregando: controla se os dados ainda estão sendo carregados.

```
const thStyle = { ... }
const tdStyle = { ... }
```

Estilos definidos para <th> e <td>, com cores escuras e texto claro.

```
useEffect(() => {
  const fetchTabela = async () => {
    const response = await fetch('https://api.api-futebol.com.br/v1/campeonatos/10/tabela', {
      headers: {
        Authorization: 'Bearer test_d669ab27a3819608318d1439cccfbf',
      },
    });
```

```
    const data = await response.json();
    setTabela(data);
    setCarregando(false);
  };
  fetchTabela();
}, []);
```

Quando o componente é carregado, a função fetchTabela faz uma chamada à API do Campeonato Brasileiro Série B (/campeonatos/10/tabela).

O Authorization precisa de um token da API-Futebol.

Após receber os dados, armazena em tabela e desativa o carregando.

```
<nav>
  <Link href="/">MENU</Link>
  ...
</nav>
```

Links de navegação para outras páginas.

```
{carregando ? (
  <p>Carregando tabela...</p>
) : Array.isArray(tabela) ? (
  <table>...</table>
) : (
  <p>Erro ao carregar a tabela.</p>
)}
```

Se ainda está carregando → mostra texto de carregamento.

Se a tabela é um array → renderiza a <table>.

Se falhou ao carregar → mostra mensagem de erro.

```
<thead>
  <tr>
    <th>Posição</th>
    ...
  </tr>
</thead>
<tbody>
  {tabela.map((time, index) => (
    <tr key={time.time.time_id}>
      <td>{time.posicao}</td>
      <td>
        <img src={time.time.escudo} />
        {time.time.nome_popular}
      </td>
      ...
    </tr>
  ))}
</tbody>
```

Cada linha da tabela representa um time com:

- Posição, nome, escudo
- Pontos, Jogos, Vitórias, Empates, Derrotas
- Gols Pró, Gols Contra, Saldo de Gols
- Aproveitamento (%)
- Últimos Jogos (ultimos_jogos.join(' ') mostra uma sequência de símbolos, como V E D

Conclusão

O desenvolvimento do projeto proporcionou uma experiência prática e enriquecedora na construção de aplicações web utilizando Next.js. Durante a execução, foi possível aplicar e consolidar conhecimentos fundamentais sobre componentes React, navegação entre páginas com roteamento dinâmico, e o consumo de APIs externas como a API-Futebol, que forneceu dados atualizados sobre o campeonato.

A construção do layout com estilo responsivo, utilizando flexbox e estilizações tanto com Tailwind CSS quanto com estilos inline, permitiu a criação de uma interface acessível e funcional, adaptável a diferentes tamanhos de tela. O uso de useEffect e useState no lado do cliente reforçou a compreensão da ciclo de vida dos componentes e do gerenciamento de estado no React.

Além disso, aprendemos a lidar com dados assíncronos, tratar erros de requisições, e interpretar estruturas JSON de uma API externa. A criação de diferentes páginas como a de artilharia, tabela e informações permitiu exercitar boas práticas de componentização, organização e reutilização de código.