Integração do projeto Next.js com a aplicação

Introdução

Bem-vindos à aula sobre "Integração do projeto Next.js com a aplicação", onde completaremos a construção e configuração das páginas da nossa aplicação Next.js. Esta etapa crucial consolida a integração com o back-end, focando em operações essenciais como criação, listagem, edição e exclusão de registros nas entidades definidas, como `doctor`, `patient`, `appointment` e `prescription`. Abordaremos desde a configuração inicial de rotas até a implementação de interações dinâmicas com o back-end. Utilizaremos técnicas de programação avançadas, como gerenciamento de estado e ciclo de vida dos componentes React, para desenvolver uma aplicação web robusta e funcional. Este módulo visa equipar os alunos com habilidades para criar interfaces de usuário integradas e eficientes, preparando-os para enfrentar desafios reais de desenvolvimento no mundo da programação.

Criação da primeira página

Iniciamos nossa exploração sobre a integração do projeto Next.js, focando na criação da primeira página e na estrutura necessária para integrar o front-end Next.js com nosso back-end em Node.js. Abordaremos o mapeamento de rotas no Next.js, um conceito essencial para evitar erros de roteamento. Posteriormente, integraremos as rotas do back-end ao projeto front-end, culminando na criação das páginas da aplicação.

Ilustrando o mapeamento de rotas, onde criamos o arquivo 'about.tsx' dentro do diretório 'pages', acessível através do endereço '/about' no

navegador. Este arquivo representará a página “Sobre” do seu site:

```
// pages/about.tsx

import React from 'react';

const About = () => {

  return (

    <div>

      <h1>Sobre Nós</h1>

      <p>Esta é a página sobre nós do nosso site.</p>

    </div>

  );

};

export default About;
```

Este exemplo ajudará a compreender como o Next.js transforma a estrutura de diretórios em rotas navegáveis.

Optamos por concentrar nossos esforços em uma única entidade para simplificar o processo de aprendizagem, dada a complexidade visual e estilística envolvida no front-end. Nossa escolha recai sobre a entidade do médico, permitindo-nos focar na autenticação e nas funcionalidades principais, como login e listagem de médicos.

Para criar uma nova página no Next.js, iniciamos com a base já configurada pelo comando `npx create-next-app`, ajustando-a conforme a necessidade

do projeto. Utilizaremos a extensão TSX, característica do Next.js, para estruturar nossa página. É crucial entender o sistema de roteamento automático do Next.js e manter a nomenclatura padrão do projeto para um roteamento eficaz.

Daremos os primeiros passos práticos acessando o projeto via VS Code, ajustando a porta do back-end para 3001 para evitar conflitos com o front-end que, por padrão, opera na porta 3000. No front-end, dentro do diretório `pages`, criaremos nossa primeira página, `home`, como um ponto de partida simples, sem focar em estilização detalhada nesta fase.

Ao configurar o ambiente de desenvolvimento, é importante realizar testes para garantir que a aplicação funcione corretamente em localhost na porta especificada. Esse processo nos familiarizará com o roteamento automático do Next.js e a geração dinâmica de páginas, estabelecendo uma base sólida para as próximas etapas do projeto. Conforme avançamos, nosso foco será criar uma aplicação que demonstre a integração eficiente entre o front-end e o back-end, proporcionando uma experiência de aprendizado prática e aplicada.

Exemplos de Rotas sem o backend

Nesta parte da nossa aula, focaremos em exemplos de rotas sem o back-end, demonstrando como iniciar com uma página na nossa aplicação Next.js. Selecionamos uma entidade específica para facilitar o entendimento e a transição da autenticação do backend para uma integração completa da aplicação.

Utilizaremos o ABP Router, que auxilia na criação de componentes de servidor no React e suporta layouts, incluindo layouts compartilhados. No Next.js, o roteamento das telas é estruturado pela organização de pastas do projeto, onde cada página terminal é nomeada como `page.tsx`. Esta nomenclatura de arquivo e a estrutura de pastas direcionam o roteamento no Next.js.

O uso do ABP Router pode ser demonstrado configurando uma rota para 'doctor/list', que exibe uma lista simulada de médicos, mesmo sem uma conexão com o backend. Isso exemplifica como gerenciar rotas e visualizar conteúdo estático durante o desenvolvimento inicial.

Na prática, criaremos um diretório `doctor` dentro do diretório `pages` do nosso projeto Next.js, ilustrando como esse roteamento é refletido na URL da aplicação. Cada subdiretório dentro de `pages`, como `doctor`, `appointment`, `prescription`, e `patient`, representará uma entidade distinta na nossa aplicação, cada uma com sua própria página `page.tsx`.

Por exemplo, para criar uma rota para a página de criação de novos médicos, adicionaremos um link na página inicial que direciona para `/doctor/create`, usando o componente `Link` do Next.js para gerenciar a navegação interna.

Ao criar essas páginas, utilizaremos a extensão TSX do Next.js para integrar TypeScript e JSX, proporcionando uma estrutura robusta para a construção da interface. Isso nos permitirá visualizar como o Next.js organiza o roteamento e prepara o terreno para uma integração mais aprofundada com o back-end nas etapas subsequentes do curso.

Integração das rotas com o backend

Na terceira parte de nossa aula, exploraremos como integrar as rotas do front-end com o back-end. Essa etapa é fundamental para estabelecer uma comunicação eficiente entre as interfaces de usuário e o servidor, permitindo que as ações realizadas no front-end reflitam no back-end e vice-versa.

Primeiramente, é essencial entender que a integração das rotas envolve o processo de requisição e resposta, onde o front-end faz solicitações específicas e o back-end retorna os dados ou resultados correspondentes. Para isso, utilizaremos o conceito de Cross-Origin Resource Sharing (CORS)

para permitir que nossa aplicação Next.js interaja com o back-end sem enfrentar restrições de segurança. A inclusão do middleware CORS no back-end é crucial para evitar erros de acesso negado, possibilitando que recursos sejam compartilhados entre diferentes origens.

Na prática, integraremos uma rota '/doctor/detail/:id' para realizar uma requisição GET e obter detalhes de um médico específico, demonstrando a dinâmica de requisição e resposta, além da utilização de parâmetros na rota.

Vamos iniciar a prática revisando o back-end para garantir que o CORS esteja corretamente configurado, adicionando o necessário `npm install cors` e implementando o middleware no código do servidor. Isso assegura que as requisições originadas do front-end sejam aceitas pelo back-end.

Em seguida, no front-end, concentraremos nossa atenção em estabelecer rotas que se comunicam com o back-end. Examinaremos como o Next.js facilita essa integração através de seu sistema de roteamento baseado na estrutura de pastas. Criaremos rotas específicas para as entidades do nosso projeto, como `doctor`, `patient`, `appointment`, e `prescription`, e implementaremos páginas correspondentes dentro de cada diretório.

Para exemplificar, vamos desenvolver uma página de criação para a entidade `doctor`, onde o usuário poderá inserir informações que serão enviadas ao back-end para processamento. Isso envolve a criação de formulários no front-end, que coletam dados e os enviam ao back-end através de métodos HTTP, como POST.

Utilizaremos `useState` e `useEffect` do React para gerenciar o estado e o ciclo de vida dos componentes, respectivamente, enquanto `fetch` ou `axios` podem ser usados para realizar as requisições HTTP ao back-end. Essas requisições serão configuradas para interagir com as APIs definidas no back-end, completando o fluxo de dados entre as duas partes da aplicação.

Ao aplicar esses direcionamentos, teremos uma compreensão de como as rotas no Next.js podem ser integradas ao back-end, permitindo uma interação dinâmica e responsiva na nossa aplicação web. Esse conhecimento é essencial para o desenvolvimento de aplicações completas, que exigem uma comunicação constante e eficiente entre o front-end e o back-end.

Criação das páginas da aplicação

Na quarta e última parte da nossa aula, focaremos na criação e configuração das páginas da nossa aplicação Next.js, consolidando a integração com o back-end. Neste estágio, aprofundaremos no conceito de como estruturar e vincular as interfaces do usuário com as operações do servidor, garantindo uma aplicação web dinâmica e funcional.

Partiremos do princípio que cada pasta ou arquivo no diretório `pages` em nosso projeto Next.js se transforma automaticamente em uma rota navegável na aplicação. Após configurar a página de login e a homepage, avançaremos para criar páginas específicas que interagem com cada entidade do nosso sistema, como `doctor`, `patient`, `appointment` e `prescription`.

Aprimoraremos a página 'doctor' adicionando a funcionalidade de edição através da rota 'doctor/[id]/edit', onde '[id]' é um parâmetro dinâmico. Esta abordagem prática ilustrará a implementação de operações CRUD em nossa aplicação.

Nesse contexto, a primeira tarefa será aprimorar a página `doctor`, que servirá como modelo para as demais entidades. Vimos anteriormente como estruturar essa página para permitir a criação (ou registro) de novos médicos. Agora, expandiremos sua funcionalidade para incluir visualizações, como a listagem completa dos médicos (`get all`), e ações como editar e deletar registros específicos.

Executaremos, então, o login na aplicação para testar a integração do front-end com o back-end. Corrigiremos quaisquer problemas relacionados à autenticação, especialmente os ligados à transmissão do token no cabeçalho das requisições HTTP. Isso é crucial para garantir que as operações realizadas no front-end sejam autorizadas e processadas corretamente pelo back-end.

Ao acessar a homepage, introduziremos um novo caminho de rota para a listagem de médicos (`list all doctors`). Essa ação nos levará à página específica que exibirá todos os registros médicos disponíveis, permitindo a interação direta com cada um deles.

Exploraremos como configurar cada página para realizar operações específicas de CRUD (Criar, Ler, Atualizar, Deletar) através de formulários e requisições HTTP, fazendo uso dos hooks `useState` e `useEffect` do React para gerenciar o estado e o ciclo de vida dos componentes. A funcionalidade de listagem (`get all`) será implementada usando `fetch` para recuperar dados do back-end e exibi-los na interface do usuário.

Por fim, detalharemos o processo de criação de interfaces para a edição e exclusão de registros, enfatizando a importância de gerenciar corretamente os identificadores (IDs) e tokens de autenticação para assegurar a segurança e integridade da aplicação.

Concluindo esta aula, os alunos terão adquirido conhecimentos sobre como integrar plenamente o Next.js com o back-end, desenvolvendo uma aplicação web interativa que reflete as operações de um sistema real.

GitHub da Disciplina

Você pode acessar o repositório da disciplina no GitHub a partir do seguinte link: https://github.com/FaculdadeDescomplica/ProgramacaoII. Esse repositório

tem como principal objetivo guardar os códigos das aulas práticas da disciplina para aprimorar suas habilidades em vários tópicos, incluindo a criação e consumo de APIs com controle de autenticação utilizando Node.js e utilizando boas práticas de programação e mercado.

Conteúdo Bônus

Para complementar o aprendizado do conteúdo abordado na aula, recomendo o vídeo "Curso Next.js: Criando páginas e roteamento", apresentado por Matheus Battisti no canal "Hora de Codar", disponível no YouTube. Esse vídeo oferece uma exploração prática sobre a criação de páginas e o sistema de roteamento no Next.js, elementos essenciais para a integração de front-end e back-end em aplicações web.

Referências Bibliográficas

Bibliografia Básica:

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados.** 7.ed. Pearson: 2018.

MEDEIROS, L. F. de. **Banco de dados:** princípios e prática. Intersaberes: 2013.

VICCI, C. (Org.). **Banco de dados.** Pearson: 2014.

Bibliografia Complementar:

CARDOSO, L. da C. **Design de aplicativos.** Intersaberes: 2022.

JOÃO, B. do N. **Usabilidade e interface homem-máquina.** Pearson: 2017.

LEAL, G. C. L. **Linguagem, programação e banco de dados:** guia prático de aprendizagem. Intersaberes: 2015.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de dados:** implementação em SQL, PL/SQL e Oracle 11g. Pearson: 2013.

SETZER, V. W.; SILVA, F. S. C. **Bancos de dados.** Blucher: 2005.

Ir para exercício