Trabalhando com arquivos em Next.js

Introdução

Nesta aula, focaremos na criação da página de exibição de arquivo, um componente essencial para a interação completa com os arquivos em nossa aplicação web. Após explorar a estrutura do projeto, a configuração da aplicação e a criação da página de upload, avançaremos para visualizar os arquivos gerados, compreendendo como o Next.js gerencia o acesso e a exibição de arquivos através de URLs. Essa etapa não só consolida nosso entendimento sobre o fluxo de trabalho com arquivos no Next.js, mas também destaca a integração entre front-end e back-end, preparando-nos para implementações práticas e eficientes em contextos reais de desenvolvimento web.

Estrutura de projeto para arquivos

Iniciamos nossa jornada explorando a gestão de arquivos em Next.js, uma etapa crucial na evolução de nossa aplicação web. Até o momento, focamos no manuseio de arquivos no back-end, mas agora estendemos essas práticas ao ambiente Next.js, buscando uma integração harmoniosa e funcional entre as duas partes da aplicação.

Ao longo do texto, abordaremos a estrutura de projeto específica para o manuseio de arquivos, configurando a aplicação para alinhar as práticas do back-end com o front-end Next.js. Este módulo é dividido em quatro segmentos principais, começando pela definição da estrutura de projeto, seguida pela configuração necessária para o gerenciamento de arquivos, práticas de upload e, por fim, a exibição dos arquivos enviados.

Nosso foco inicial, a estrutura de projeto para arquivos, é uma base que precisamos entender e estabelecer corretamente. No Next.js, a organização do projeto e o roteamento são intimamente ligados, com a estrutura de diretórios influenciando diretamente como as rotas são resolvidas. Essa abordagem requer atenção na organização dos diretórios e arquivos, garantindo que o mapeamento faça sentido no contexto da aplicação e suas funcionalidades.

É essencial perceber que, embora o Next.js ofereça flexibilidade na estruturação do projeto, devemos seguir padrões que facilitam a manutenção e a clareza do código. Estruturas como `controller`, `image`, ou `model` podem ser incorporadas ao projeto, permitindo que o Next.js as utilize para operações como upload, leitura e armazenamento de dados. O objetivo é que, independentemente da complexidade da estrutura, o resultado seja uma interface funcional e integrada ao restante da aplicação.

Ao trabalhar em ambiente colaborativo ou corporativo, a definição e a adesão a uma estrutura de projeto padronizada são cruciais. Isso evita confusões e erros decorrentes de mapeamentos incorretos ou redundantes, facilitando a integração e a manutenção do código entre o front-end e o back-end. Por exemplo, se no back-end temos uma pasta `prescriptions` para arquivos, faz sentido espelhar essa organização no front-end, promovendo uma integração intuitiva e eficiente.

Portanto, a análise e ajuste da estrutura de projeto são passos iniciais vitais. É necessário revisar as estruturas existentes, verificar sua lógica e adequação, e ajustar conforme necessário para assegurar uma gestão eficaz de arquivos e rotas.

Nas próximas partes deste módulo, mergulharemos na implementação, explorando como realizar o upload e a visualização de arquivos no Next.js, aprimorando nossas habilidades em desenvolvimento web e preparando-nos para desafios reais no mercado de trabalho.

Configuração da aplicação para arquivos

Prosseguindo com nossa aula, vamos entender mais sobre a configuração da aplicação para lidar com arquivos. Esse segmento é fundamental para garantir que o front-end esteja preparado para realizar o upload de arquivos, integrando-se harmoniosamente ao back-end que já configuramos para essa finalidade.

A funcionalidade de upload de arquivos no front-end não é apenas uma extensão estética do back-end, mas uma parte integral da aplicação que requer uma configuração cuidadosa e precisa. Verificar a funcionalidade do back-end é o primeiro passo, assegurando que ele esteja apto a receber arquivos. Essa preparação envolve, entre outros aspectos, a revisão do código e das configurações estabelecidas anteriormente, para evitar qualquer desalinhamento entre as duas partes da aplicação.

No front-end, a configuração se concentra no mapeamento e na criação de páginas específicas para o gerenciamento de arquivos. A recomendação é reservar uma área dedicada para essas funções, não apenas por uma questão de organização, mas também para facilitar a manutenção e a escalabilidade da aplicação.

Na configuração da aplicação para arquivos, é crucial que o front-end esteja preparado para o upload. Por exemplo, na configuração do Next.js, precisamos especificar no arquivo 'next.config.js' quais tipos de arquivos serão aceitos, usando o 'fileExtensions' para permitir formatos como '.pdf' ou '.jpg'. Ao criar a página de upload, podemos adicionar um componente 'FileUploader' que interage com a API do back-end para salvar arquivos no servidor.

Para ilustrar, na nossa interface inicial, podemos adicionar um link ou componente que direcione para a funcionalidade de upload de arquivos, sugerindo a possibilidade de uma organização em submenus para uma navegação intuitiva e eficiente.

Nesta fase, estabelecemos uma estrutura básica que inclui a página de upload de arquivos. A estrutura reflete a organização já presente no back-end, utilizando, por exemplo, a pasta `prescriptions` para alinhar as operações de arquivos entre as duas frentes da aplicação. Isso demonstra uma continuidade e consistência entre o front-end e o back-end, simplificando o processo de integração e manutenção.

Durante a implementação, a tela de upload é concebida como uma área onde poderemos gerenciar os arquivos, ainda que inicialmente seja apresentada de forma simplificada. A estrutura no front-end mimetiza a organização do back-end, onde cada entidade possui sua respectiva seção, como a pasta `prescriptions` para o upload de arquivos. A interface é projetada para facilitar a navegação e a interação com a aplicação, integrando funcionalidades como a visualização de tabelas vazias, que posteriormente serão populadas com os dados correspondentes.

Para o desenvolvimento da aplicação, utilizamos o modelo híbrido do Next.js, que permite a manipulação tanto do lado do servidor quanto do cliente, mantendo a flexibilidade e a eficiência no manuseio dos arquivos. A implementação no front-end é planejada para incluir funcionalidades essenciais como a navegação e o gerenciamento de estados, utilizando ferramentas como `useRouter` e `useState` para facilitar o upload e a listagem de arquivos.

Concluindo esta parte da aula, estabelecemos a base para a gestão de arquivos no front-end, preparando o terreno para as próximas etapas, onde realizaremos o upload de arquivos e sua subsequente visualização na aplicação. Continuaremos a expandir e aprofundar nossas habilidades na próxima aula, focando na implementação efetiva e na operacionalização do upload de arquivos, enriquecendo nossa experiência prática e conhecimento técnico em desenvolvimento web com Next.js.

Criação da página de upload de arquivo

Avançaremos agora na criação da página de upload de arquivo, com o objetivo de estruturar essa funcionalidade na interface da aplicação, assegurando a integração com o back-end. Vamos considerar um exemplo prático onde a funcionalidade de upload é essencial: um sistema para um consultório médico onde é possível fazer upload dos históricos médicos dos pacientes.

Na nossa aplicação Next.js, criaremos uma página chamada `upload.tsx` dentro de `pages/medical-records`. Essa página conterá um formulário com campos para inserir informações do paciente e um campo de upload de arquivo. O envio desses dados ao back-end será feito usando a API `fetch`.

A estrutura básica do arquivo `upload.tsx` poderia ser assim:

```
import React, { useState } from 'react';

function UploadPage() {

    const [file, setFile] = useState(null);

    const handleSubmit = async (event) => {

        event.preventDefault();

        const formData = new FormData();

        formData.append('file', file);

        try {

            const response = await fetch('/api/upload', {

                method: 'POST',

                body: formData,
```

```
            });

            if (!response.ok) {

                        throw new Error('Erro no upload do
arquivo');

            }

            alert('Arquivo enviado com sucesso!');

        } catch (error) {

                 console.error('Erro ao enviar arquivo:',
error);

        }

    };

    const handleFileChange = (event) => {

        setFile(event.target.files[0]);

    };

    return (

        <form onSubmit={handleSubmit}>

            <label>

                Arquivo:

                            <input type="file" onChange=
{handleFileChange} />
```

```
            </label>

            <button type="submit">Enviar</button>

        </form>

    );

}

export default UploadPage;
```

Neste código, utilizamos o hook `useState` para gerenciar o estado do arquivo selecionado. O método `handleSubmit` trata o envio do formulário, preparando o `FormData` e realizando a requisição POST para o endpoint `/api/upload`. A função `handleFileChange` atualiza o estado quando o usuário seleciona um arquivo.

Considerando a integração com o banco de dados, é crucial que a estrutura do banco suporte as novas funcionalidades. Se o upload de arquivos foi adicionado posteriormente ao design inicial do sistema, devemos garantir que as entidades relevantes no banco de dados sejam atualizadas para suportar o armazenamento de referências aos arquivos enviados.

Além disso, a interface do usuário na página de upload deve ser clara e funcional, com botões e mensagens que orientem o usuário durante o processo de upload, garantindo uma boa usabilidade.

Concluindo, a criação de uma página de upload de arquivo exige uma abordagem metódica, considerando a experiência do usuário, a integração com o back-end e a consistência dos dados no banco de dados. Este exemplo prático demonstra a importância de planejar cuidadosamente a implementação de novas funcionalidades em sistemas existentes.

Criação da página de exibição de arquivo

Finalizando nossa aula, abordaremos a criação da página de exibição de arquivo. Após estabelecer a funcionalidade de upload, o próximo passo é implementar a maneira pela qual os arquivos, como as prescrições médicas em nosso projeto, são visualizados.

A exibição de arquivos no Next.js é conduzida através de URLs, seguindo a mesma lógica de requisição e resposta que já vimos. O front-end inicia um processo que solicita ao back-end o arquivo desejado, armazenado e referenciado por uma URL específica no banco de dados. Assim que o back-end processa essa solicitação, ele fornece o arquivo, permitindo ao front-end exibir ou baixar o conteúdo para o usuário.

A visualização de arquivos envolve uma interação direta com o back-end, onde o front-end desempenha o papel de solicitar e apresentar o arquivo com base em sua localização identificada pela URL. Esta localização pode variar, dependendo da configuração do sistema, seja em um ambiente local ou em nuvem.

Na tela de upload, integramos a funcionalidade `showfile`, que permite a visualização do arquivo. Esta função utiliza o método GET para recuperar o caminho do arquivo no banco de dados, oferecendo ao usuário a opção de download. Esse mecanismo é consistente com outras implementações de integração com o back-end, com a particularidade de não necessitar de um corpo na requisição, já que é um GET.

O processo segue uma estrutura padrão de erro e tratamento de resposta, com a exibição de erros no front-end em caso de falhas no back-end. A resposta obtida é então transformada em um blob, um tipo de objeto que facilita a manipulação de arquivos binários, e convertida em uma URL para download.

No aspecto visual da implementação no Next.js, a funcionalidade é acionada por um botão que invoca `showfile`, passando o ID da prescrição como referência. A lógica condicional determina se o botão de visualização

ou upload será exibido, baseando-se na existência do arquivo na prescrição.

Este tópico conclui a aula sobre manipulação de arquivos em Next.js, fornecendo uma compreensão de como implementar e gerenciar a funcionalidade de arquivos em uma aplicação web. Os detalhes e o código usado estarão disponíveis no GitHub da disciplina, permitindo aos alunos revisar, comparar e implementar essas funcionalidades em seus próprios projetos.

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 aqueles interessados em aprofundar seus conhecimentos sobre o gerenciamento de arquivos em Next.js, recomendo o vídeo “Uploading Files With Next.js Just Got Way Easier” do canal Ravi - Perfect Base no YouTube. Este material oferece uma visão prática e detalhada sobre as técnicas avançadas de upload de arquivos, mostrando as últimas atualizações e facilitações oferecidas pelo Next.js.

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