Sistema de Lembretes

Maria Eduarda Amaral Muniz

dudaamaralmap3110@gmail.com

1. Organização do sistema

O sistema está estruturado no diretório 'codigo', que é subdividido em 'front-end' e 'back-end'.

Na pasta 'front-end', encontra-se o projeto React. Dentro dela, o diretório 'components' contém os componentes responsáveis pelo cadastro de lembretes, bem como o componente de listagem de lembretes, acompanhado de uma opção para exclusão. Dentro do diretório 'hooks', estão os arquivos dos hooks personalizados para os lembretes. O diretório 'services' contém o arquivo de serviço dos lembretes, que encapsula os métodos para recuperar dados da API. Já no diretório 'styles', estão os arquivos de estilização dos componentes, utilizando pré-processadores CSS. A página 'utils' engloba a lógica de validação dos dados e manipulação de datas.

Na pasta 'back-end', encontramos o diretório 'Controllers', que abriga o controlador responsável pelos lembretes. Também há o diretório 'Models', onde reside o modelo dos lembretes e a conexão com o banco de dados. A pasta 'Validations' contém as regras de validação dos dados. Por fim, o diretório 'Views' contém uma página HTML, atualmente não utilizada, uma vez que o front-end está separado do back-end.

2. Premissas assumidas

Frontend em React: O projeto utiliza o framework React para o desenvolvimento do frontend da aplicação.

Gestão de Lembretes: O objetivo principal da aplicação é permitir que os usuários criem e gerenciem lembretes.

Componente LembreteForm: Existe um componente chamado LembreteForm que é responsável por renderizar um formulário para adicionar novos lembretes.,

Componente LembreteList: Existe um componente chamado LembreteList que é responsável por renderizar uma lista dos lembretes cadastrados.

Validação de Campos: O componente LembreteForm realiza validações nos campos de nome e data antes de enviar o lembrete para o servidor.

Feedback ao Usuário: Caso haja algum erro durante o envio do lembrete ou se os campos não forem preenchidos corretamente, o componente LembreteForm fornece feedback ao usuário exibindo mensagens de erro na interface.

Backend em .NET: O backend foi desenvolvido em .NET.

Hooks Customizados: O projeto utiliza hooks customizados, como useLembretes e useForm, para gerenciar o estado e a lógica de negócios de forma modularizada.

API RESTful: O backend da aplicação segue o estilo arquitetural RESTful, fornecendo endpoints HTTP para realizar operações read, create e delete.

Banco de dados: O SGBD utilizado foi o MySQL, junto com a ferramenta de administração HeidiSQL.

3. Decisões de projeto

Optou-se por utilizar o React para desenvolver a interface do usuário (UI) devido à especificação do teste, bem como pela sua eficiência, reatividade e extenso ecossistema de bibliotecas e ferramentas. Essa escolha facilitou significativamente a construção de uma UI dinâmica e responsiva, atendendo aos requisitos modernos de interatividade e fluidez.

Decidiu-se empregar hooks personalizados, como useLembretes e useForm, para gerenciar o estado e a lógica de negócios do frontend. Essa abordagem permitiu uma organização mais estruturada do código, propiciando uma manutenção mais eficiente e simplificando a extensão da aplicação. Além disso, tornou-a modular e reutilizável, aspectos fundamentais para o desenvolvimento ágil e escalável.

A validação dos dados no frontend antes do envio para o servidor foi uma escolha estratégica para assegurar a integridade dos dados e proporcionar uma experiência mais fluida ao usuário. Essa prática não apenas reduz a necessidade de requisições desnecessárias ao servidor, mas também oferece feedback imediato ao usuário, promovendo uma interação mais clara e intuitiva.

A modularização do código em arquivos separados, como os hooks personalizados e as funções de validação, foi cuidadosamente planejada visando à separação de responsabilidades e à organização estruturada do código-fonte. Essa abordagem contribui para uma manutenção mais escalável.

Além disso, o padrão arquitetural Model-View-Controller (MVC) foi adotado para estruturar a aplicação de forma mais coerente e escalável. O MVC separa a aplicação em três componentes principais: o Modelo (Model), que gerencia os dados e a lógica de negócios, a Visão (View), responsável pela interface do usuário (que neste caso, está no diretório front-end), e o Controlador (Controller), que coordena as interações entre o Modelo e a Visão. Essa abordagem facilita a manutenção, o teste e a evolução da aplicação, promovendo uma arquitetura mais robusta e flexível.