Documentação referente a aplicação “Inventory Control”

Autor: Ekistoclecio Heleno Duarte de Lima

Visão Geral do Projeto:

O projeto tem como objetivo implementar uma aplicação de controle de estoque onde o usuário possa realizar o cadastro fornecendo uma quantidade limitada de dados pessoais e após isso, possa realizar o login na aplicação, onde ele terá acesso a um conjunto de recursos que lhe possibilitem cadastrar uma lista de produtos e gerenciá-la.

1. **Cadastro de Usuário:**

O usuário poderá se cadastrar fornecendo 5 informações básicas: nome, sobrenome, e-mail, senha e repetir senha. O sistema possui um mecanismo de validação em duas etapas, onde algumas verificações são realizadas do lado do cliente e outras no lado do servidor. Essas verificações tem como objetivo garantir a integridade dos dados armazenados e a consistência das informações no banco de dados.

Verificações no lado do Cliente:

- Nome: Permite qualquer string diferente de vazio.
- Sobrenome: Permite qualquer string diferente de vazio.
- E-mail: Permite qualquer string diferente de vazio desde que possua um formato de e-mail válido.
- Senha: A senha deverá conter pelo menos 6 caracteres.
- Repetir de senha: É verificado se os dados digitados no campo de “senha” e “repetir senha” são iguais.

Verificações do lado do Servidor:

- E-mail: Faz uma verificação de modo a garantir que todo e-mail cadastrado no sistema seja único.

Caso o e-mail que o usuário está tentando cadastrar já pertença a outro usuário, o servidor retorna a mensagem de erro “O e-mail informado já foi cadastrado.”

2. **Login:**

Para acessar a aplicação, o usuário deve realizar o login fornecendo um e-mail e senha previamente cadastrados. No lado do cliente, verifica-se se o usuário

preencheu os campos de e-mail e senha antes de enviar a solicitação de login para o servidor. No servidor, a solicitação é processada da seguinte maneira:

- O servidor verifica se o e-mail fornecido está cadastrado no banco de dados.
- Se o e-mail não estiver associado a nenhum usuário cadastrado, ou se a senha fornecida não corresponder à senha registrada no banco de dados, o servidor retornará a mensagem de erro "E-mail ou senha inválidos".
- Se o e-mail estiver cadastrado e a senha fornecida corresponder à senha armazenada no banco de dados, o servidor gerará um token de autenticação.
- Esse token é enviado de volta ao cliente, juntamente com algumas informações do usuário, como nome, sobrenome e e-mail.
- O cliente armazena o token e as informações do usuário no armazenamento local, para autenticação subsequente nas páginas protegidas da aplicação.

3. **Gerenciamento de Produtos:**

O usuário pode gerenciar sua lista de produtos realizando a adição, edição ou exclusão dos registros de cada um dos produtos.

- **Adição**: O usuário pode cadastrar um novo produto em sua lista clicando no botão "Adicionar produto" e, em seguida, preenchendo o formulário de cadastro com duas informações básicas: o nome do produto e a quantidade dele. Após isso, a visualização do produto cadastrado estará disponível na tabela de produtos.
- **Edição**: Na tabela de produtos, cada linha representa um produto específico e inclui dois botões. O botão com o ícone de um "lápis" abre um modal onde o usuário pode alterar o nome e a quantidade do produto.
- **Exclusão**: O segundo ícone na tabela é o de uma "lixeira". Clicar nele exclui o produto do sistema.

Ao tentar efetuar uma "Adição" ou "Edição" de um produto, o sistema realiza uma verificação para determinar se o nome do produto já está cadastrado no sistema. Esse processo tem como objetivo evitar que um usuário cadastre dois produtos com o mesmo nome. A verificação é realizada no servidor da aplicação, garantindo a integridade dos dados e a consistência do banco de dados.

4. **Exibição dos Produto:**

A tela de “Produtos” da aplicação exibe uma tabela com três colunas: o nome do produto, a quantidade e uma coluna de ações que disponibiliza botões para editar e excluir o produto. A tabela utiliza um sistema de scroll infinito, o que significa que os dados são carregados progressivamente à medida que o usuário realiza o scroll para baixo. Isso otimiza o desempenho, evitando o carregamento completo dos dados no início da aplicação.

5. **Pesquisa:**

A tela de “Produtos” apresenta uma barra de busca na qual o usuário pode pesquisar pelo nome do produto. O sistema realiza automaticamente a filtragem dos produtos exibidos na tabela com base no nome digitado. Além disso, a barra de pesquisa possui um recurso de autocompletar, que é baseado nos produtos já cadastrados. Esse recurso ajuda o usuário a encontrar o produto desejado com maior facilidade.

Quando o usuário digita um novo caractere na barra de busca, o sistema envia uma requisição para o servidor. O servidor, por sua vez, consulta o banco de dados em busca de produtos cadastrados pelo usuário cujas iniciais do nome coincidam com os caracteres pesquisados. Em seguida, o servidor retorna a lista de produtos correspondentes. Esses produtos são exibidos dinamicamente na tabela de produtos.

6. **Edição dos Dados do Usuários:**

O sistema permite que um usuário logado altere seus próprios dados cadastrais por meio da tela de "Usuário". Nessa tela, é possível editar diretamente o nome, sobrenome e e-mail clicando no ícone ao lado de cada campo correspondente. Depois de realizar as alterações desejadas, o usuário pode salvar as mudanças clicando novamente no ícone ao lado do campo modificado.

Além disso, a tela de "Usuário" oferece a opção de alterar a senha cadastrada. Para isso, o usuário deve fornecer a senha atualmente cadastrada no sistema, bem como a nova senha desejada.

É importante destacar que o processo de edição de dados do usuário segue os mesmos processos de validação de dados que são realizados durante o cadastro de um novo usuário. Isso é feito para garantir a integridade e consistência dos dados armazenados pela aplicação no banco de dados.

Segurança:

A aplicação incorpora recursos básicos de segurança essenciais, seguindo as práticas padrão para garantir a integridade dos dados dos usuários. Essas medidas incluem:

- **Armazenamento Criptografado de Senhas:** As senhas dos usuários são armazenadas no banco de dados de forma criptografada. Isso significa que as senhas originais são transformadas em uma forma irreversível antes de serem armazenadas, tornando-as seguras contra acessos não autorizados.
- **Sistema de Autenticação JWT (JSON Web Token):** A aplicação utiliza um sistema de autenticação baseado em JWT. Isso permite que os usuários autenticados recebam tokens de autenticação, que são usados para verificar sua identidade em solicitações subsequentes. Esses tokens são assinados

digitalmente e incluem informações sobre o ID do usuário. Isso ajuda a proteger as rotas e recursos da aplicação contra acesso não autorizado.

Essas práticas de segurança são fundamentais para proteger as informações dos usuários e garantir que apenas pessoas autorizadas tenham acesso aos dados e funcionalidades da aplicação.

Design:

A aplicação adota um design simples e intuitivo, que destaca de forma clara as funcionalidades disponíveis no sistema. Isso foi deliberadamente planejado para simplificar a navegação e tornar o uso dos recursos acessível aos usuários. Além disso, foram aplicados cuidados para garantir que o design seja responsivo, mantendo sua simplicidade e clareza mesmo em dispositivos com telas menores, como smartphones.

Para alcançar esse resultado, a aplicação faz uso da biblioteca de estilo Material UI. Essa escolha foi feita porque a Material UI oferece componentes simples, elegantes e de fácil integração. Esses componentes contribuem para a criação de um layout agradável e amigável ao usuário, melhorando a experiência geral de uso da aplicação.

Estrutura da Aplicação:

A aplicação possui 2 pastas principais “backend” e “frontend” elas tem como objetivo dividir de forma clara os arquivos responsáveis pelo funcionamento do backend e frontend. Facilitando na organização do projeto e eventuais manutenções ou modificações.

1. **Backend:**

O backend foi elaborado com o auxílio do express para a criação de uma API REST e foi estruturado em 9 rotas, essas rotas estão descritas em “backend/src/routes/index.ts”. E possui um banco de dados PostgreSQL para armazenamento de dados, para facilitar a execução do backend foi utilizado docker/docker compose.

Rotas disponíveis:

Rotas Públicas

- Criação de usuário (POST): “/user/create”
- Login de usuário (POST): “/user/login”

Rotas Protegidas

- Edição de dados do usuário (PUT): “/user/changeUserData”
- Mudança de senha do usuário (PATCH): “/user/changeUserPassword”
- Pegar lista de produtos do usuário (GET): “/user/products/:page”
- Adicionar produto (POST): “/product/create”
- Deletar produto (DELETE): “/product/delete/:name”
- Editar produto (PUT): “/product/update”
- Pesquisar produtos (GET): “/product”

A aplicação conta com 2 tipos de rotas, user e product, cada tipo possui seu próprio arquivo dentro da pasta “backend/src/controller” responsável por implementar as funções que irão tratar as requisições para cada uma das rotas.

Além disso, o servidor conta com o middleware responsável por verificar a autenticação do usuário antes de completar as requisições para rotas protegidas.

Banco de dados:

Todos os arquivos responsável pela configuração do banco de dados encontram-se na pasta “backend/src/database”, onde é possível encontrar a pasta entities responsável por armazenar os arquivos que descrevem as duas entidades centrais do sistema: usuário(nome, sobrenome, email e senha) e produto(nome, quantidade).

2. Frontend:

O frontend foi elaborado com o auxílio da ferramenta Vite(React). Com duas páginas centrais, a de login e registro do usuário, e a página “Home” para quando o usuário estiver logado.

- **Componentes**

Os componentes React responsáveis por renderizar as páginas da aplicação encontram-se na pasta “frontend/src/pages”. Essas páginas são:

- Login e Registro: Pagina responsável pelo login e registro do usuário
- Home: Página responsável por exibir os elementos da aplicação após o usuário efetuar login.
- Router Error: Pagina para caso o usuário tente acessar uma rora não disponível na aplicação.
- Loading: Tela de carregamento simples para ser exibida enquanto o sistema carrega os dados de sessão do usuário.

Os componentes auxiliares responsáveis por renderizar determinados elementos em tela podem ser encontrados dentro da pasta “frontend/src/components”.

- **Hooks Personalizados**

    Toda a lógica relacionada ao funcionamento dos componentes foi modularizada e movida para hooks personalizados, que estão disponíveis na pasta "frontend/src/hooks". Esses hooks personalizados têm como objetivo armazenar qualquer lógica associada ao funcionamento de um componente específico.

    Cada hook recebe um nome que é composto pela palavra "use" seguida pelo nome do componente ao qual o hook está relacionado. Essa convenção de nomenclatura torna fácil identificar a qual componente um determinado hook personalizado pertence.

    Essa organização ajuda a melhorar a legibilidade e a manutenção do código, permitindo que a lógica associada a cada componente seja isolada em um local específico e reutilizável. Além disso, a separação da lógica dos componentes torna o código mais limpo e mais fácil de entender

- .**Services**

As funções relacionadas a requisições no servidor podem ser encontradas na pasta “frontend/services/api”, cada arquivo é responsável pela requisição em uma rota específica do servidor.

*Tais requisições foram feitas com o auxílio da biblioteca Axios.

- **Validação de formulário**

A validação de qualquer formulário do lado do cliente foi feita com o auxílio da biblioteca Zod, responsável por automatizar as validações de forma simples e prática.

Os esquemas utilizados para a validação de cada formulário podem ser encontrados na pasta “frontend/src/utils/zod”.

- **Contextos**

Dentro da pasta “frontend/src/providers/context” é possível encontrar três contextos personalizados criados com o intuito de fornecer para toda a aplicação funções e estados essenciais para o funcionamento de múltiplos componentes.

    - **SessionContext**: Contexto responsável por prover funções e estados relacionados a manipulação da sessão atual do usuário.
    - **AlertContext**: Contexto responsável por prover funções e estados necessários para a exibição de alertas personalizados para o cliente.
    - **ProductsContext**: Contexto responsável por prover funções e estados necessários para a manipulação da lista de produtos do usuário do lado do cliente.

Conclusão

É evidente que as regras de negócio implementadas no projeto são de natureza simples, e a aplicação foi intencionalmente projetada como uma base sólida para futuras aplicações de controle de estoque mais complexas. O projeto oferece um esquema simples e robusto, com código de fácil entendimento e manutenção. Essa abordagem proporciona a flexibilidade necessária para adicionar novas funcionalidades com relativa facilidade, tornando-o adaptável para atender às necessidades em constante evolução.

Além desta documentação, é importante destacar que o código do projeto é enriquecido com comentários para auxiliar no entendimento e na manutenção das funções, garantindo a transparência e a clareza na implementação.