Stock System

Stock System é uma aplicação desenvolvida utilizando React Native e NodeJS para gereneciar um controle de estoque simples.

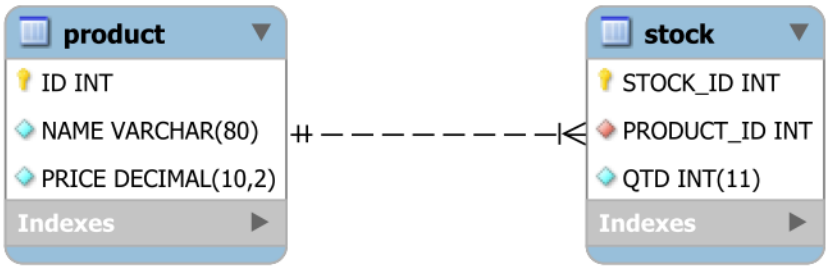
Requisitos Funcionais

- O sistema deve permitir a um usuário gerenciar produtos.
- O sistema deve permitir a um usuário buscar produtos.
- O sistema deve permitir a um usuário adicionar produtos.

Requisitos Não Funcionais

- O sistema deve ser construído em React Native.
- O sistema deve ser acessível em pelo menos uma das seguintes plataformas: iOS ou Android.
- O sistema deve ser contruído com um Back-End REST WebAPI.
- O sistema deve utilizar um banco de dados relacional.
- O sistema deve utilizar Redux para gerenciamento do estado.

Modelagem do Banco de Dados



Comentários

- A modelagem foi feita para atender exclusivamente o escopo proposto, mas está aberta para a expansão.
- A modelagem não caso haja uma tabela de preço para os produtos. Contudo foi a melhor alternativa para não fugir do escopo inicial.
- A separação da quantidade na tabela `STOCK`, ocorre devido ao caráter expansionista do projeto. É possível adicionar uma nova coluna e identificar entradas e saídas do estoque. A modelagem também não foi tão explorada neste aspecto, devido ao escopo proposto.

Estrutura da aplicação:

- Adotou-se o MVC como padrão arquitetural da aplicação como um todo, por sua segregação de responsabilidades em camadas. Desta forma, facilitando a escalabilidade do projeto, reutilização de modelos e facilidade de manutenção do código.
- O backend é uma API REST, na qual seus métodos foram documentados abaixo.
- As políticas de CORS foram "desabilitadas" durante o desenvolvimento.
- O desenvolvimento preferiu utilizar a biblioteca React Native Paper com a premissa de ser portável para ambos as plataformas. No entanto, o app foi executado apenas na plataforma Android.

Instalação e Execução:

- É preciso instalar os pacotes Nodemon (Back-end) e Expo-Cli (Front-end) globalmente.
- Dentro da pasta do projeto, execute os seguintes comandos em dois terminais distintos, simultâneamente:

```
$ cd stockProject;
$ npm i;
$ npm start;
```

```
$ cd stockBack;
$ npm i;
$ npm start;
```

Back-End:

As seguintes rotas foram desenvolvidas:

Add Product

POST: `scheme://host:port/path/to/stockBack/product/add` Body:

```
{
    "name": "Banana",
    "qtd": 23,
    "price": 0.78
}
```

Response:

```
{
    "message": "Produto cadastrado com sucesso!"
}
```

Edit Product

POST: `scheme://host:port/path/to/stockBack/product/edit` Body:

```
{
    "productId": "12",
    "name": "Banana",
    "qtd": 23,
    "price": 0.78
}
```

Response:

```
{
    "message": "Produto editado com sucesso!"
}
```

Remove Product

POST: `scheme://host:port/path/to/stockBack/product/rm` Body:

```
{
    "productId": "12"
}
```

Response:

```
{
    "message": "Produto removido com sucesso!"
}
```

Get Products

GET: `scheme://host:port/path/to/stockBack/products` Response:

```
{
    "products": [
        {
            "id": "12",
            "name": "Batata",
            "qtd": 23,
            "price": 12.23
        }
    ]
}
```

Front-End:

O Front-End foi segregado na seguinte estrutura de pastas:

```
- stockProject
    - components
    - config
    - redux
        - actions
        - reducers
    - screens
    - services
```

- A biblioteca Redux-Navigation foi utilizada para transitar entre as telas.
- Hooks foram utilizados em todos os componentes necessários.
- Hooks da biblioteca React-Redux também foram utilizados.

Contribuições

São bem vindas! Basta mandar um pull request!