

# Stock System

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

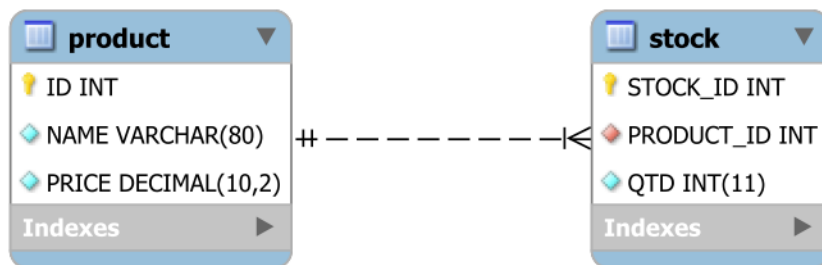
## Requisitos Funcionais

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

## 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 construído 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 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 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.
- Utilizou-se a biblioteca React Native Paper com a premissa de não se preocupar tanto com o design da aplicação.
- 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, simultaneamente:

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

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

## Back-End:

- Utiliza as tecnologias **Node JS** e **Express**

- O banco de dados escolhido foi o [MySQL \(v5.7\)](#).
- 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:

- Utiliza as seguintes tecnologias:
  - [Expo](#)
  - [React Native](#)
  - [Redux](#)
  - [React Native Paper](#)
- 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!