Caixa Eletrônico

O projeto consiste basicamente em 3 funções: deposito, saque, extrato. Ao executar qualquer uma das 3 funções seus dados serão automaticamente cadastrados pelo sistema e ao mesmo tempo executando a função chamada. Por isso não haverá tela de cadastro ou login, pois não será necessário.

Caso informe o CPF ou um valor não permitido, o sistema exibirá uma mensagem de erro informando o que foi feito de errado. Caso a execução ocorra de acordo uma mensagem de sucesso será exibida ou os dados exigidos.

Executar o Projeto

1. Precisa ter instalado em sua máquina
    - Gerenciador de pacotes como NPM ou YARN
    - Node.js
    - Git

2. Execute o código:
    git clone https://github.com/Alyfer-Reis/caixaEletronico.git

3. entre na pasta caixa eletrônico e execute o arquivo sql.sql em seu gerenciador de banco de dados MYSQl.

4. Depois entre na pasta back-end.

5. crie um arquivo .env dentro da pasta back-end com o seguinte código

```
    module.exports = {

  db: {
    host : 'localhost',
    database: 'cash_machine',
    user: 'nome do ususario',
    port: porta,
    password: 'senha do usuário'
  }
}
```

6. abra seu terminal na pasta back-end e execute os seguintes códigos.
    - npm i
    - npm start

7. se der tudo certo o back-end deve estar executando corretamente.

8. Entre na pasta front-end e execute os mesmo códigos do passo 6.

9. Se tudo dê certo o projeto irá abrir na porta 3000, e pronto para testes.

Tecnologias

- **Front-end**
    1. **React:** Biblioteca Javascript

    2. **Styled-components:** Pre-processador de css, garantindo a mesma experiencia do css tradicional mais a possibilidade de manipular o css com Javascript.

- **Back-end**
    1. **Express:** Framework utilizado para criação de APIs utilizando Javascript.

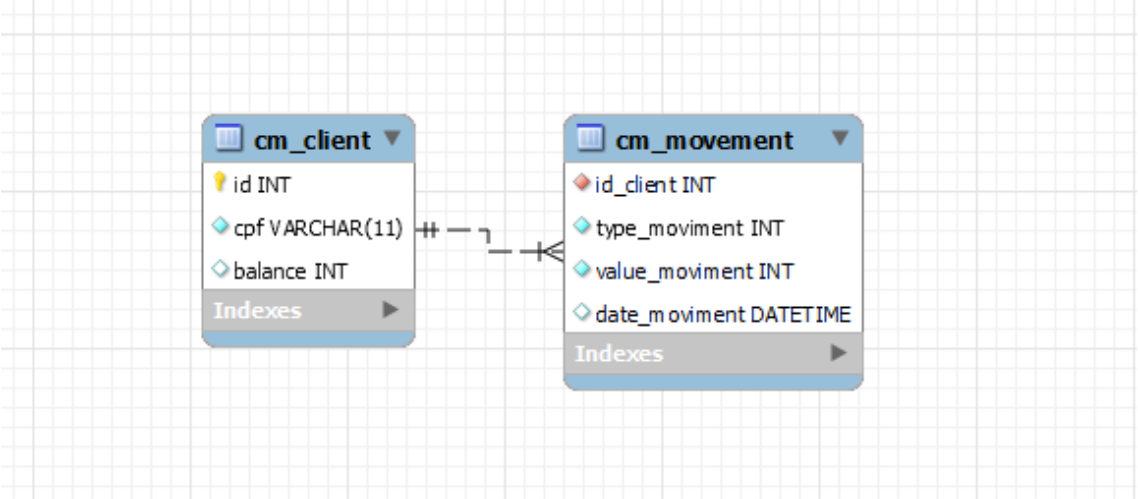
    2. **Consign:** biblioteca que ajuda na organização do código, facilitando o gerenciamento de rotas no express.

    3. **Serverless-MYSQL:** biblioteca que realiza aconexão com o banco de dados mysql e facilitando na criação e execução de query, de uma forma mais manual.

    4. **Nodemon:** execução de API automática sem precisar reiniciar a aplicação quando uma alteração e feita.

    5. **MYSQL:** banco de dados relacional**.**

Banco de dados



Tabelas

1. Cm_client

Campo	Tipo/Tamanho	Obrigatório	Comentário
id	INT	S	Campo destinado para identificação do cliente (chave primaria)
CPF	VARCHAR(11)	S	Cpf do cliente
balance	INT	N	Saldo da conta do cliente
RELACIONAMENTOS			
Tabela	**Descrição**		
N/A	N/A		

2. Cm_movement

Campo	Tipo/Tamanho	Obrigatório	Comentário
Id_client	INT	S	Campo destinado para identificação do cliente (chave primaria)
Type_movement	INT	S	Tipo de movimentação 1. Para deposito 2. Para saque
Value_movement	INT	S	Valor depositado ou sacado da movimentação
Date_movement	Datetime	S	Data da movimentação
RELACIONAMENTOS			
Tabela	**Descrição**		
Cm_client	Id_client vinculado com id do client		

Teste de API

1. **Deposito**
    - **url:** localhost:3000/deposito
    - **method:** POST
    - **body:**

```
{
  "cpf":"somente os numeros do cpf",
  "money": valor do deposito
}
```

2. **Saque**
    - **url:** localhost:3000/saque
    - **method:** POST
    - **body:**

```
{
  "cpf":"somente os numeros do cpf",
  "money": valor do deposito
}
```

3. **Extrato**
    - **url:** localhost:3000/extrato
    - **method:** POST
    - **body:**

```
{
  "cpf":"somente os numeros do cpf"
}
```