Desenvolvimento de um sistema vendas: Front-end em ReactJS e API desenvolvida com o spring framework

Dezembro 2018

Contents

1 Introdução

1.1 Propósito deste documento

Fornecer informações sobre o funcionamento e desenvolvimento da API e do front-end.

1.2 Descrição do projeto

O projeto se desenvolveu a partir de um cenário onde existia uma empresa de vendas que possuía vendedores e gerentes e produtos a serem vendidos para os clientes e havia a necessidade do sistema realizar vendas e gerar relatórios.

A proposta dessa atividade visa a complementação do plano de formação de estágio.

2 Objetivos

Utilizando o Spring Framework e seus subprojetos, desenvolver uma API REST para permitir a exposição de dados de vendas, clientes produtos e vendedores, também a inserção e atualização desses dados.

Para fazer o consumo dessa API, também era necessário criar uma aplicação front-end, por onde fosse possível realizar vendas, cadastrar novos clientes, cadastrar vendedores, gerar relatórios, etc.

3 Topologia

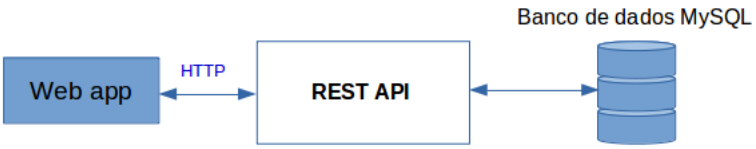

Figure 1: Visão geral do projeto

4 Métodos da API

4.1 getUser

Esse método permite retornar um usuário de acordo com o contexto. Utilizando o basicAuth do Spring Security, simplesmente informamos o username e password

- Get Request URL: http://localhost:8080/api/user/
- Success Response Body


```
{
    “password”: null,
    “username”: “larissac”,
    “authorities”: [
        {
            “authority”:“ROLE_EMPLOYEE”
        },
        {
            “authority”: “ROLE_MANAGER”
        }
    ],
```

"accountNonExpired": true,
"accountNonLocked": true,
"credentialsNonExpired": true,
"enabled": true
}

- Error Reponse Body:
  É retornado uma resposta vazia com código HTTP 401.

4.2 getUserByUsername

Retorna um usuário passando o username.

- Get Request URL: http://localhost:8080/api/user/username/{username}
- Success Response Body para username jsalencar
  {
  "cpf": "09809809800",
  "username": "jsalencar",
  "password": *"encrypted password"*,
  "manager": false
  }
- Error Response Body para username lucas:
  {
  "timestamp": 1545668579511,
  "status": 404,
  "error": "Not Found",
  "message": "Element not found",
  "path": "/api/user/username/lucas"
  }

4.3 addUser

Adiciona um usuário

- Post Request URL: http://localhost:8080/api/user/
- Request Body
  {
  "cpf": "string",
  "username": "string",
  "password": "string",
  "isManager": boolean
  }
- Success Response Body. Exemplo com alguns parâmetros passados de acordo com o request body
  {
  "cpf": "09809809801",
  "username": "josesl",
  "password": "joses",
  "isManager": false
  }
- Error Response Body. Caso username ou CPF já exista.
  {
  "timestamp": 1545668579511,
  "status": 409,

```
    "error": "Conflict",
    "message": "Element already exists",
    "path": "/api/user"
}
```


4.4 updateUser

Atualiza um usuário de determinado CPF

- PUT Request URL: http://localhost:8080/api/user/cpf/{cpf}
- Request Body


```
{
    "cpf": "string",
    "username": "string",
    "password": "string",
    "isManager": boolean
}
```


- Success Response Body. Exemplo com alguns parâmetros passados de acordo com o request body


```
{
    "cpf": "09809809801",
    "username": "josesl",
    "password": "joses",
    "isManager": false
}
```


- Error Response Body. Caso username ou CPF já exista.


```
{
    "timestamp": 1545668579511,
    "status": 409,
    "error": "Conflict",
    "message": "Element already exists",
    "path": "/api/user"
}
```


4.5 getAllCustomers

Lista todos os clientes

- GET Request URL: http://localhost:8080/api/customer
- Success Response Body


```
[
    {
        "id": 1,
        "name": "Fernando Calvacante",
        "cpf": "64598753400",
        "fone": "5516999999999",
        "email": "fernandoc@outlook.com"
    },
    {
        "id": 2,
        "name": "Amilton Calvacante",
        "cpf": "64598753401",
        "fone": "5516999999999",
        "email": "amilton@outlook.com"
```

```
    }
]
```

- Error Response Body
  É retornado uma lista vazia

4.6 getCustomerByName

Busca um cliente pelo nome

- GET Request URL: http://localhost:8080/api/customer/name/{name}
- Success Response Body. Exemplo para name **calvacante**

```
[
    {
        "id": 1,
        "name": "Fernando Calvacante",
        "cpf": "64598753400",
        "fone": "5516999999999",
        "email": "fernandoc@outlook.com"
    },
    {
        "id": 2,
        "name": "Amilton Calvacante",
        "cpf": "64598753401",
        "fone": "5516999999999",
        "email": "amilton@outlook.com"
    }
]
```

- Error Response Body. Exemplo para name **Luana**

```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Not Found",
    "message": "Element not found",
    "path": "/api/customer/name/Luana"
}
```

4.7 getCustomerByCpf

Busca um cliente pelo CPF

- GET Request URL: http://localhost:8080/api/customer/cpf/{cpf}
- Success Response Body. Exemplo para cpf **64598753400**

```
{
    "id": 1,
    "name": "Fernando Calvacante",
    "cpf": "64598753400",
    "fone": "5516999999999",
    "email": "fernandoc@outlook.com"
}
```

- Error Response Body. Exemplo para cpf **64598753410**

```
{
    "timestamp": 1545668579511,
    "status": 404,
```

```
    "error": "Not Found",
    "message": "Element not found",
    "path": "/api/customer/cpf/64598753410"
}
```


4.8 addCustomer

Adiciona um novo cliente

- POST Request URL: http://localhost:8080/api/customer/
- Request Body


```
    "name": "string",
    "cpf": "string",
    "fone": "string",
    "email": "string"
}
```


- Success Response Body


```
{
    "id": 1,
    "name": "Fernando Calvacante",
    "cpf": "64598753400",
    "fone": "5516999999999",
    "email": "fernandoc@outlook.com"
}
```


- Error Response Body. Acontece se tentar adicionar um cliente com CPF já existente


```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Conflict",
    "message": "Element already exists",
    "path": "/api/customer/"
}
```


4.9 updateCustomer

Atualiza os dados de um cliente passando o ID

- PUT Request URL: http://localhost:8080/api/customer/id/{id}
- Request Body


```
    "name": "string",
    "cpf": "string",
    "fone": "string",
    "email": "string"
}
```


- Success Response Body. Exemplo para id **1**


```
{
    "id": 1,
    "name": "Fernando Calvacante",
    "cpf": "64598753400",
    "fone": "5516999999999",
```

```
    "email": "fernandoc@outlook.com"
}
```


- Error Response Body. Acontece se tentar atualizar um cliente com CPF já existente


```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Conflict",
    "message": "Element already exists",
    "path": "/api/customer/id/1"
}
```


4.10 getAllEmployees

Lista todos os empregados

- GET Request URL: http://localhost:8080/api/employee
- Success Response Body


```
[
    {
        "id": 1,
        "cpf": "09809809800",
        "name": "José de Alencar Mendes",
        "fone": "5516999999999",
        "email": "joseam@outlook.com"
        "position": "vendedor"
    },
    {
        "id": 2,
        "cpf": "09809809803",
        "name": "Larissa do Carmo",
        "fone": "5516999999999",
        "email": "larissadc@outlook.com"
        "position": "gerente"
    }
]
```


- Error Response Body
  É retornado uma lista vazia

4.11 getEmployeeByName

Busca um funcionário pelo nome

- GET Request URL: http://localhost:8080/api/employee/name/{name}
- Success Response Body. Exemplo para name **Larissa**


```
[
    {
        "id": 2,
        "cpf": "09809809803",
        "name": "Larissa do Carmo",
        "fone": "5516999999999",
        "email": "larissadc@outlook.com"
        "position": "gerente"
    }
]
```

- Error Response Body. Exemplo para name **Luana**

```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Not Found",
    "message": "Element not found",
    "path": "/api/employee/name/Luana"
}
```

4.12 getEmployeeByCpf

Busca um funcionário pelo CPF

- GET Request URL: http://localhost:8080/api/employee/cpf/{cpf}
- Success Response Body. Exemplo para cpf **09809809803**

```
{
    "id": 2,
    "cpf": "09809809803",
    "name": "Larissa do Carmo",
    "fone": "5516999999999",
    "email": "larissadc@outlook.com"
    "position": "gerente"
}
```

- Error Response Body. Exemplo para cpf **09809809813**

```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Not Found",
    "message": "Element not found",
    "path": "/api/employee/cpf/09809809813"
}
```

4.13 addEmployee

Adiciona um novo funcionário

- POST Request URL: http://localhost:8080/api/employee/
- Request Body

```
{
    "cpf": "string",
    "name": "string",
    "fone": "string",
    "email": "string"
    "position": "string"
}
```

- Success Response Body. Exemplo passando alguns parâmetros conforme solicitado no Request Body

```
{
    "id": 2,
    "cpf": "09809809803",
    "name": "Larissa do Carmo",
    "fone": "5516999999999",
    "email": "larissadc@outlook.com"
```

```
    "position": "gerente"
}
```


- Error Response Body. Acontece se tentar adicionar um funcionário com CPF já existente


```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Conflict",
    "message": "Element already exists",
    "path": "/api/employee/"
}
```


4.14 updateEmployee

Atualiza os dados de um funcionário passando ID

- PUT Request URL: http://localhost:8080/api/employee/id/{id}
- Request Body


```
{
    "cpf": "string",
    "name": "string",
    "fone": "string",
    "email": "string"
    "position": "string"
}
```


- Success Response Body. Exemplo passando alguns parâmetros conforme solicitado no Request Body


```
{
    "id": 2,
    "cpf": "09809809803",
    "name": "Larissa do Carmo",
    "fone": "5516999999999",
    "email": "larissadc@outlook.com"
    "position": "gerente"
}
```


- Error Response Body. Acontece se tentar atualizar um funcionário com CPF já existente


```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Conflict",
    "message": "Element already exists",
    "path": "/api/employee/"
}
```


4.15 getAllProducts

Lista todos os produtos

- GET Request URL: http://localhost:8080/api/product
- Success Response Body


```
[
    {
        "id": 1,
        "name": "Caixa de Som",
```

```
        "resume": "Cor preta, conexão bluetooth e usb",
        "price": 100,
        "quantity": 1000
    },
    {
        "id": 2,
        "name": "Fone de Ouvido",
        "resume": "Alta qualidade de som",
        "price": 57.9,
        "quantity": 100
    }
]
```


- Error Response Body
  É retornado uma lista vazia

4.16 getProductById

Busca um produto pelo ID

- GET Request URL: http://localhost:8080/api/product/id/{id}
- Success Response Body. Exemplo para id **1**


```
{
    "id": 1,
    "name": "Caixa de Som",
    "resume": "Cor preta, conexão bluetooth e usb",
    "price": 100,
    "quantity": 1000
}
```


- Error Response Body. Exemplo para id **11**


```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Not Found",
    "message": "Element not found",
    "path": "/api/product/id/11"
}
```


4.17 getProductByName

Busca um produto pelo nome

- GET Request URL: http://localhost:8080/api/product/name/{name}
- Success Response Body. Exemplo para name **Som**


```
[
    {
        "id": 1,
        "name": "Caixa de Som",
        "resume": "Cor preta, conexão bluetooth e usb",
        "price": 100,
        "quantity": 1000
    }
]
```

- Error Response Body. Exemplo para name **Led**


```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Not Found",
    "message": "Element not found",
    "path": "/api/product/name/Led"
}
```


4.18 getProductByResume

Busca um produto pela descrição

- GET Request URL: http://localhost:8080/api/product/resume/{resume}
- Success Response Body. Exemplo para resume **sem fio**


```
[
    {
        "id": 4,
        "name": "Teclado",
        "resume": "Preto, retroiluminado sem fio",
        "price": 130,
        "quantity": 104
    }
]
```


- Error Response Body. Exemplo para resume **gamer**


```
{
    "timestamp": 1545668579511,
    "status": 404,
    "error": "Not Found",
    "message": "Element not found",
    "path": "/api/product/resume/gamer"
}
```


4.19 addProduct

Adiciona um novo produto

- POST Request URL: http://localhost:8080/api/product/
- Request Body


```
{
    "name": "string",
    "resume": "string",
    "price": double,
    "quantity": integer
}
```


- Success Response Body. Exemplo passando os parâmentros de acordo com o Request Body


```
{
    "id": 1,
    "name": "Teclado",
    "resume": "Preto, retroiluminado sem fio",
    "price": 130,
    "quantity": 104
}
```

- Error Response Body. Acontece se tentar adicionar um produto sem o campo **name**


```
{
    "timestamp": 1545668579511,
    "status": 400,
    "error": "Bad Request",
    "message": "Bad Request",
    "path": "/api/product/"
}
```


4.20 updateProduct

Atualiza um produto passando o ID

- PUT Request URL: http://localhost:8080/api/product/id/{id}
- Request Body


```
{
    "name": "string",
    "resume": "string",
    "price": double,
    "quantity": integer
}
```


- Success Response Body. Exemplo passando os parâmentros de acordo com o Request Body


```
{
    "id": 1,
    "name": "Teclado",
    "resume": "Preto, retroiluminado sem fio",
    "price": 130,
    "quantity": 104
}
```


- Error Response Body. Acontece se tentar atualizar um produto sem o campo **name**


```
{
    "timestamp": 1545668579511,
    "status": 400,
    "error": "Bad Request",
    "message": "Bad Request",
    "path": "/api/product/"
}
```


4.21 getAllSales

Busca todas as vendas

- GET Request URL: http://localhost:8080/api/sale/
- Success Response Body.


```
[
    {
        "id": 1,
        "productPrice": 100,
        "salePrice": 200,
        "productQuantity": 2,
        "descount": 0,
        "paymentMethod": "debito"
        "dateTimeSale": "21-12-2018"
```

```
        "product": "Caixa de Som"
        "customer": "Amilton Calvacante"
        "employee": "José de Alencar Mendes"
    },
    {
        "id": 2,
        "productPrice": 130,
        "salePrice": 130,
        "productQuantity": 1,
        "descount": 0,
        "paymentMethod": "debito"
        "dateTimeSale": "21-12-2018"
        "product": "Fone de Ouvido"
        "customer": "Fernando Calvacante"
        "employee": "Larissa do Carmo"
    }
]
```


4.22 getSaleByEmployee

Busca todas as vendas de um determinado vendedor passando o CPF

- GET Request URL: http://localhost:8080/api/sale/employee/cpf/{cpf}
- Success Response Body. Exemplo para cpf **09809809803**


```
[
    {
        "id": 2,
        "productPrice": 130,
        "salePrice": 130,
        "productQuantity": 1,
        "descount": 0,
        "paymentMethod": "debito"
        "dateTimeSale": "21-12-2018"
        "product": "Fone de Ouvido"
        "customer": "Fernando Calvacante"
        "employee": "Larissa do Carmo"
    },
    {
        "id": 3,
        "productPrice": 57.9,
        "salePrice": 115.8,
        "productQuantity": 2,
        "descount": 0,
        "paymentMethod": "debito",
        "dateTimeSale": "21-12-2018",
        "product": "Fone de Ouvido",
        "customer": "Amilton Calvacante",
        "employee": "Larissa do Carmo"
    }
]
```


4.23 addSale

Adiciona uma nova venda

- POST Request URL: http://localhost:8080/api/sale/
- Success Response Body


```
{
    "productPrice": 130,
    "salePrice": 130,
    "productQuantity": 1,
    "descount": 0,
    "paymentMethod": "debito",
    "idProduct": 1,
    "idCustomer": 2,
    "idEmployee": 4
}
```


4.24 getAllReports

Lista os relatórios

- GET Request URL: http://localhost:8080/api/report/
- Success Response Body


```
[
    {
        "id": 1,
        "dateTimeSale": "2018-12-25",
        "paymentMethod": "debito",
    },
    {
        "id": 2,
        "dateTimeSale": 2018-12-26,
        "paymentMethod": "dinheiro",
    }
]
```


5 Banco de dados

O banco de dados foi modelado em MySQL utilizando para esse fim a ferramenta MySQL Workbench.

Foram criadas as tabelas abaixo para representar as entidades produto, venda, cliente, funcionário e usuário. Essas tabelas se relacionam conforme mostra a figura abaixo.

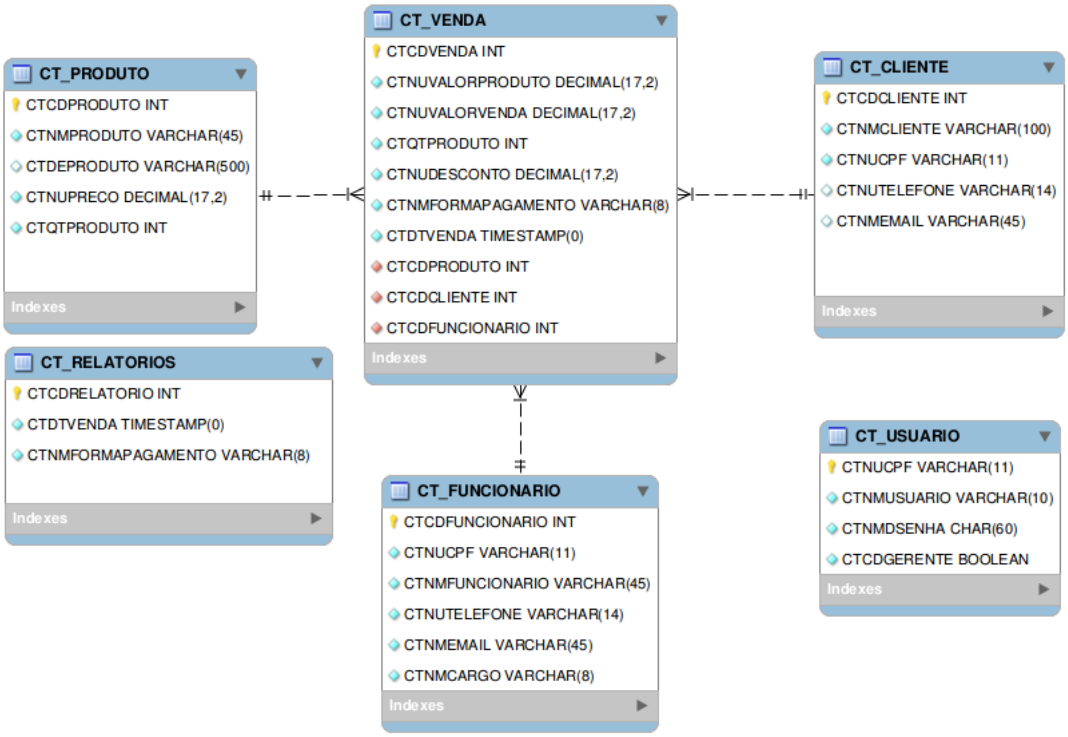

Figure 2: Diagrama de entidade e relacionamento

6 Versionamento

O projeto está versionado no link abaixo:

- git clone https://rochards@bitbucket.org/rochards/projeto-plataformas-02.git