

Relatório Exercício Prático - Backend Development

Otávio Augusto *

Viridis, Belo Horizonte - MG

14 de maio de 2018



*otavio.augusto@outlook.com

Sumário

1	Objetivo	3
2	Dependências utilizadas	4
3	Exemplos de uso da API	5
3.1	Equipamentos	6
3.2	Manutenção	8
4	Testes Unitários	14
5	Documentação da API	15

1 Objetivo

O objetivo desse exercício é construir um serviço RESTful que implemente o escopo simplificado de gestão de ativos industriais, cuja API deve expor os seguintes recursos e métodos:

- Equipamentos
 1. Listar o inventário com todos os equipamentos;
 2. Consultar um equipamento específico através de seu id.
- Ordens de manutenção (indicam a data em que um equipamento será submetido a um determinado serviço de manutenção)
 1. Criar uma ordem de manutenção;
 2. Alterar uma ordem de manutenção existente;
 3. Consultar todas as ordens de manutenção existentes;
 4. Consultar uma ordem específica através de seu id.

Por se tratar de um escopo simplificado da regra de negócio, não foram contemplados outros relacionamentos e entidades que entrariam em uma gestão de ativos industriais. Tudo o que foi criado está em conformidade com as solicitações deste exercício.

2 Dependências utilizadas

As seguintes ferramentas e dependências utilizadas no projeto foram:

- Java JDK 8;
- Maven 4.0.0;
- Spring Data JPA;
- Banco de Dados PostgreSQL 9.4;
- SLF4J para guardar os logs;
- Spring Tool Suite 3.9.3;
- Spring Boot 2.0.2;
- Lombok para diminuir a verbosidade das entidades;
- Spring boot test;
- Swagger 2.7.0;
- Jackson;
- Mockito.

3 Exemplos de uso da API

As seções a seguir mostram cada um dos métodos solicitados no escopo dos objetivos¹ para a realização do exercício. Para cada um dos métodos realizados foi implementado o `HttpServletRequest` que é uma extensão da interface `ServletRequest`. Ele é utilizado para fazer a autenticação e validar o acesso à API através de um parâmetro Header e uma Hash SHA-1.

Para persistência de dados foi utilizado o Spring Data que facilita a criação de repositórios. Ele faz isso livrando o programador de ter que implementar as interfaces referentes aos repositórios (ou DAOs), e também já deixando pré-implementadas algumas funcionalidades como, por exemplo, de ordenação das consultas e de paginação de registros. Para isso, foi utilizada a interface `JpaRepository`.

A figura 1 mostra como deve ser adicionado o parâmetro Header e o Hash de autenticação ao Postman (programa utilizado para os testes) em todos os métodos implementados. Ambos estão disponíveis na classe `AuthenticateService` do pacote `security`. No Postman, na aba Builder, vá até a aba Header e insira os dados em `key(header)` e `value(hash)` respectivamente.

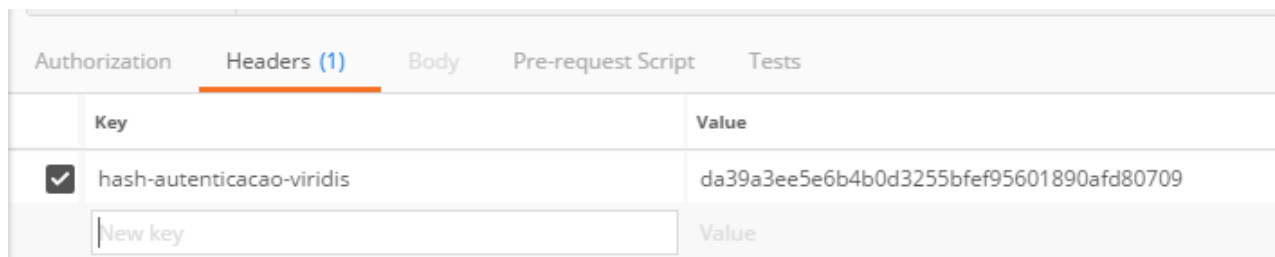
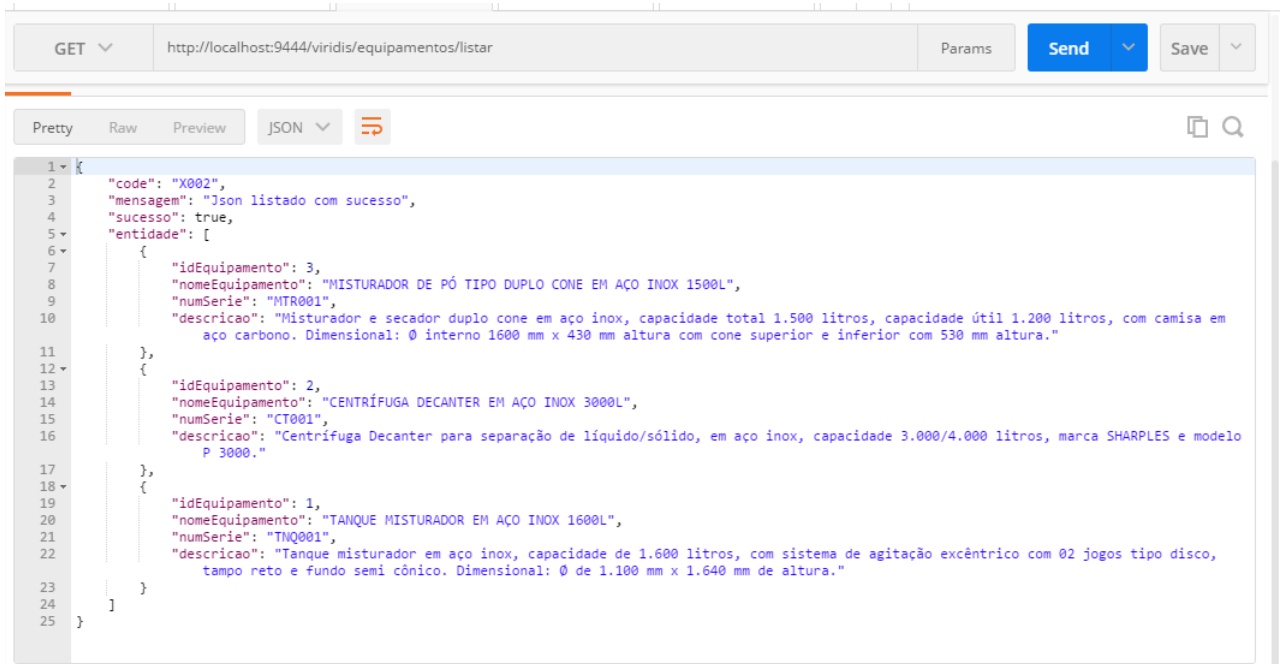


Figura 1: Autenticação Http

3.1 Equipamentos

- Listar o inventário com todos os equipamentos.



```
1 {
2   "code": "X002",
3   "mensagem": "Json listado com sucesso",
4   "sucesso": true,
5   "entidade": [
6     {
7       "idEquipamento": 3,
8       "nomeEquipamento": "MISTURADOR DE PÓ TIPO DUPLO CONE EM AÇO INOX 1500L",
9       "numSerie": "MTR001",
10      "descricao": "Misturador e secador duplo cone em aço inox, capacidade total 1.500 litros, capacidade útil 1.200 litros, com camisa em aço carbono. Dimensional: Ø interno 1600 mm x 430 mm altura com cone superior e inferior com 530 mm altura."
11    },
12    {
13      "idEquipamento": 2,
14      "nomeEquipamento": "CENTRÍFUGA DECANter EM AÇO INOX 3000L",
15      "numSerie": "CT001",
16      "descricao": "Centrifuga Decanter para separação de líquido/sólido, em aço inox, capacidade 3.000/4.000 litros, marca SHARPLES e modelo P 3000."
17    },
18    {
19      "idEquipamento": 1,
20      "nomeEquipamento": "TANQUE MISTURADOR EM AÇO INOX 1600L",
21      "numSerie": "TNQ001",
22      "descricao": "Tanque misturador em aço inox, capacidade de 1.600 litros, com sistema de agitação excêntrico com 02 jogos tipo disco, tampo reto e fundo semi cônico. Dimensional: Ø de 1.100 mm x 1.640 mm de altura."
23    }
24  ]
25 }
```

Figura 2: Método para listar todos os Equipamentos

- Consultar um equipamento específico através de seu id.

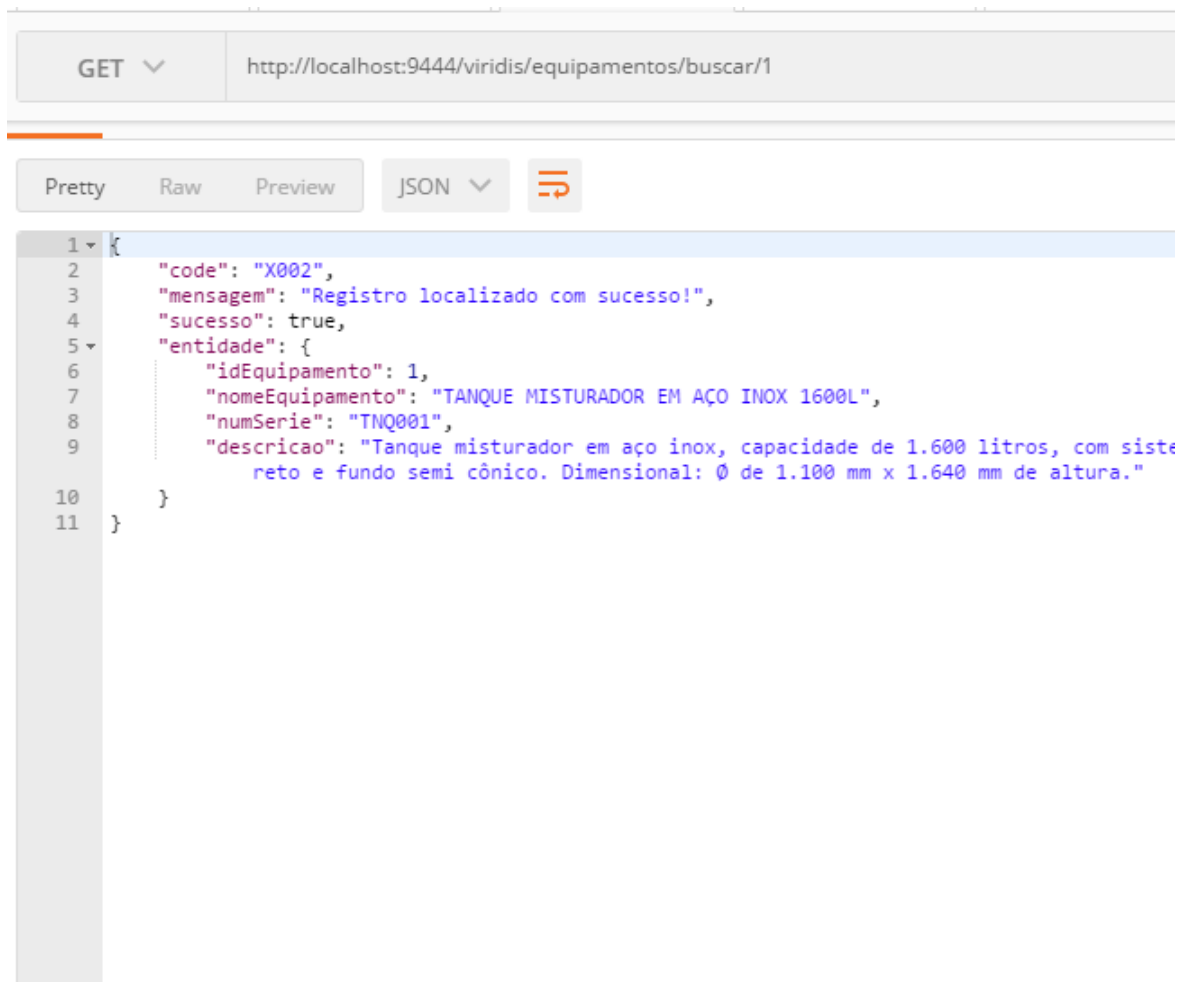
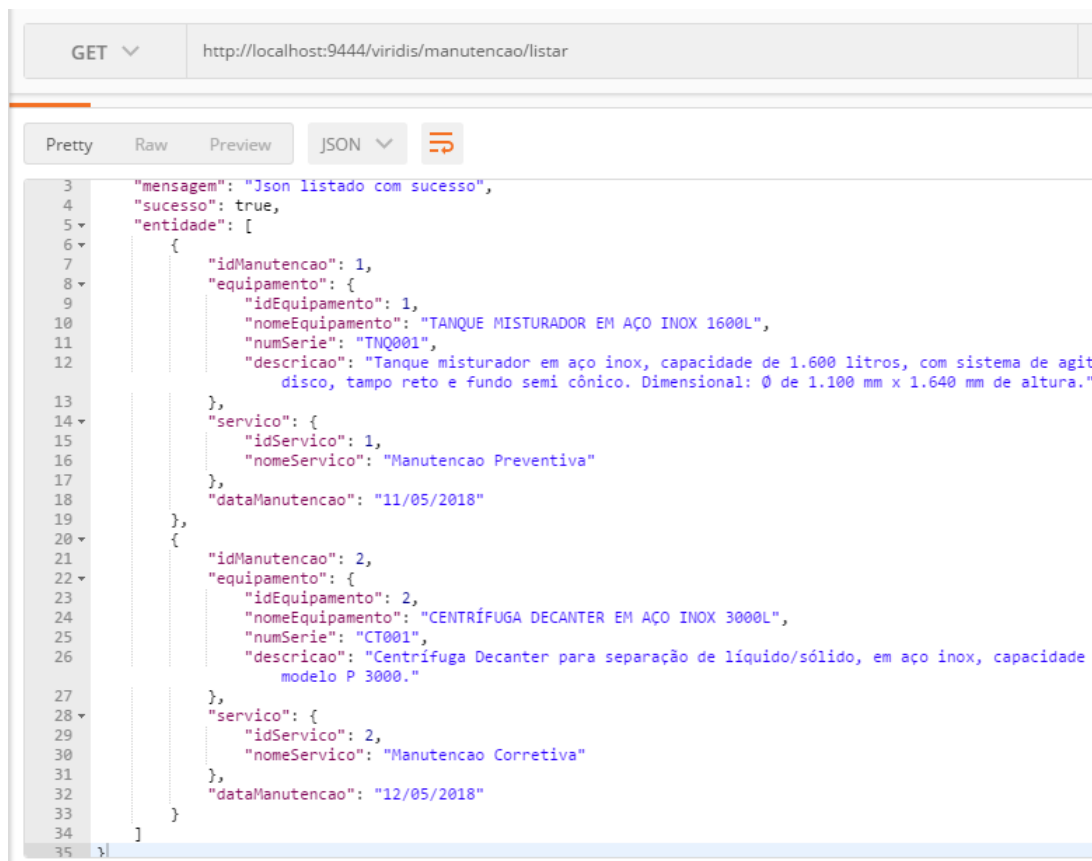


Figura 3: Método para consultar um equipamento através de um id

3.2 Manutenção

As Ordens de manutenção indicam a data em que um equipamento será submetido a um determinado serviço de manutenção.

- Consultar todas as ordens de manutenção existentes.

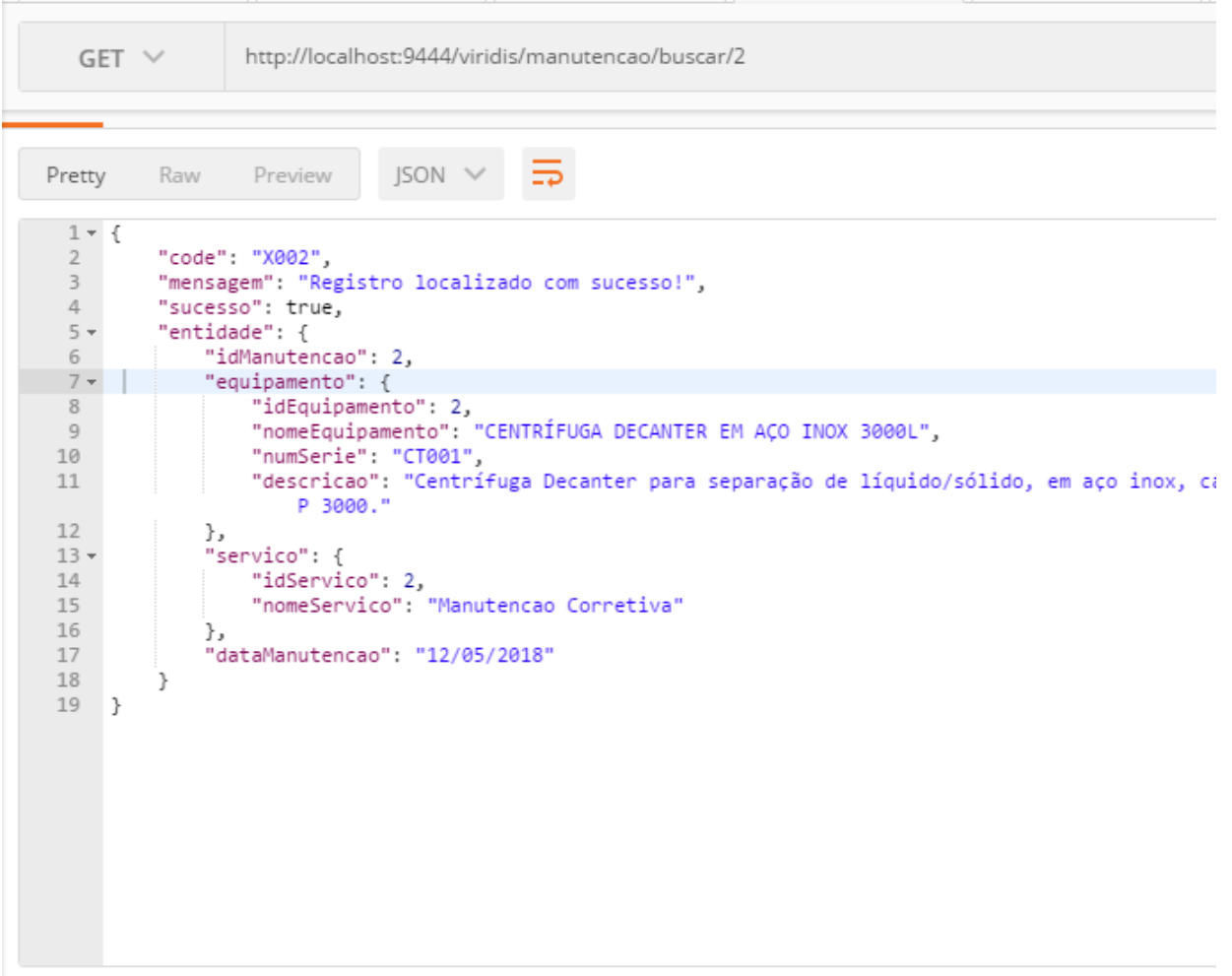


```
GET http://localhost:9444/viridis/manutencao/listar

{"mensagem": "Json listado com sucesso",
  "sucesso": true,
  "entidade": [
    {
      "idManutencao": 1,
      "equipamento": {
        "idEquipamento": 1,
        "nomeEquipamento": "TANQUE MISTURADOR EM AÇO INOX 1600L",
        "numSerie": "TNQ001",
        "descricao": "Tanque misturador em aço inox, capacidade de 1.600 litros, com sistema de agit
          disco, tempo reto e fundo semi cônico. Dimensional: Ø de 1.100 mm x 1.640 mm de altura."
      },
      "servico": {
        "idServico": 1,
        "nomeServico": "Manutencao Preventiva"
      },
      "dataManutencao": "11/05/2018"
    },
    {
      "idManutencao": 2,
      "equipamento": {
        "idEquipamento": 2,
        "nomeEquipamento": "CENTRÍFUGA DECANTER EM AÇO INOX 3000L",
        "numSerie": "CT001",
        "descricao": "Centrífuga Decanter para separação de líquido/sólido, em aço inox, capacidade
          modelo P 3000."
      },
      "servico": {
        "idServico": 2,
        "nomeServico": "Manutencao Corretiva"
      },
      "dataManutencao": "12/05/2018"
    }
  ]
}
```

Figura 4: Método para listar todas as ordens de Manutenções do banco de Dados. Existem duas linhas já cadastradas no banco.

- Consultar uma ordem específica através de seu id.



```
1 {
2   "code": "X002",
3   "mensagem": "Registro localizado com sucesso!",
4   "sucesso": true,
5   "entidade": {
6     "idManutencao": 2,
7     "equipamento": {
8       "idEquipamento": 2,
9       "nomeEquipamento": "CENTRÍFUGA DECANter EM AÇO INOX 3000L",
10      "numSerie": "CT001",
11      "descricao": "Centrífuga Decanter para separação de líquido/sólido, em aço inox, c
12      P 3000."
13    },
14    "servico": {
15      "idServico": 2,
16      "nomeServico": "Manutencao Corretiva"
17    },
18    "dataManutencao": "12/05/2018"
19  }
```

Figura 5: Método para buscar uma ordem de Manutenção específica pelo id

- Criar uma ordem de manutenção.

Como visto acima no método para listar Manutenções(Figura 4), existem duas linhas já pré cadastradas no banco. Portanto ao persistir uma nova será gerado o `id_manutenção = 3`. O Hibernate é capaz de perceber quando um conjunto de dados é novo e serializar a entidade para gerar um novo registro, com isso, não há necessidade de passar o parâmetro no Json.

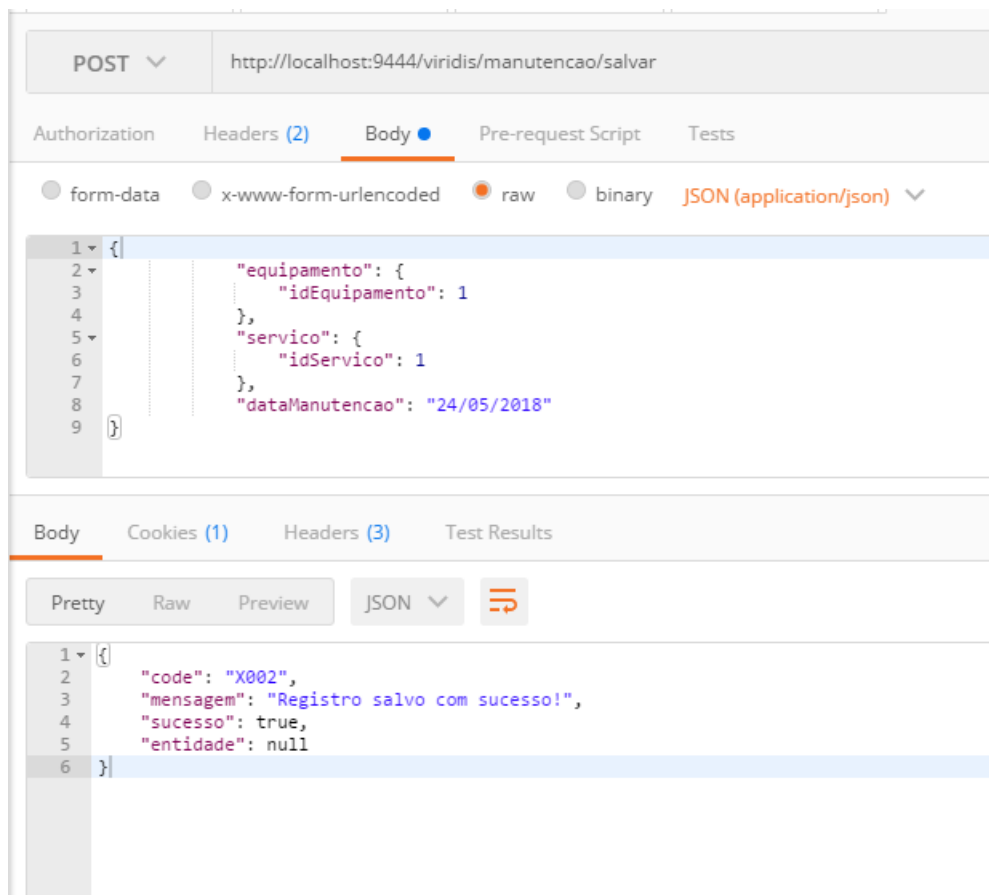
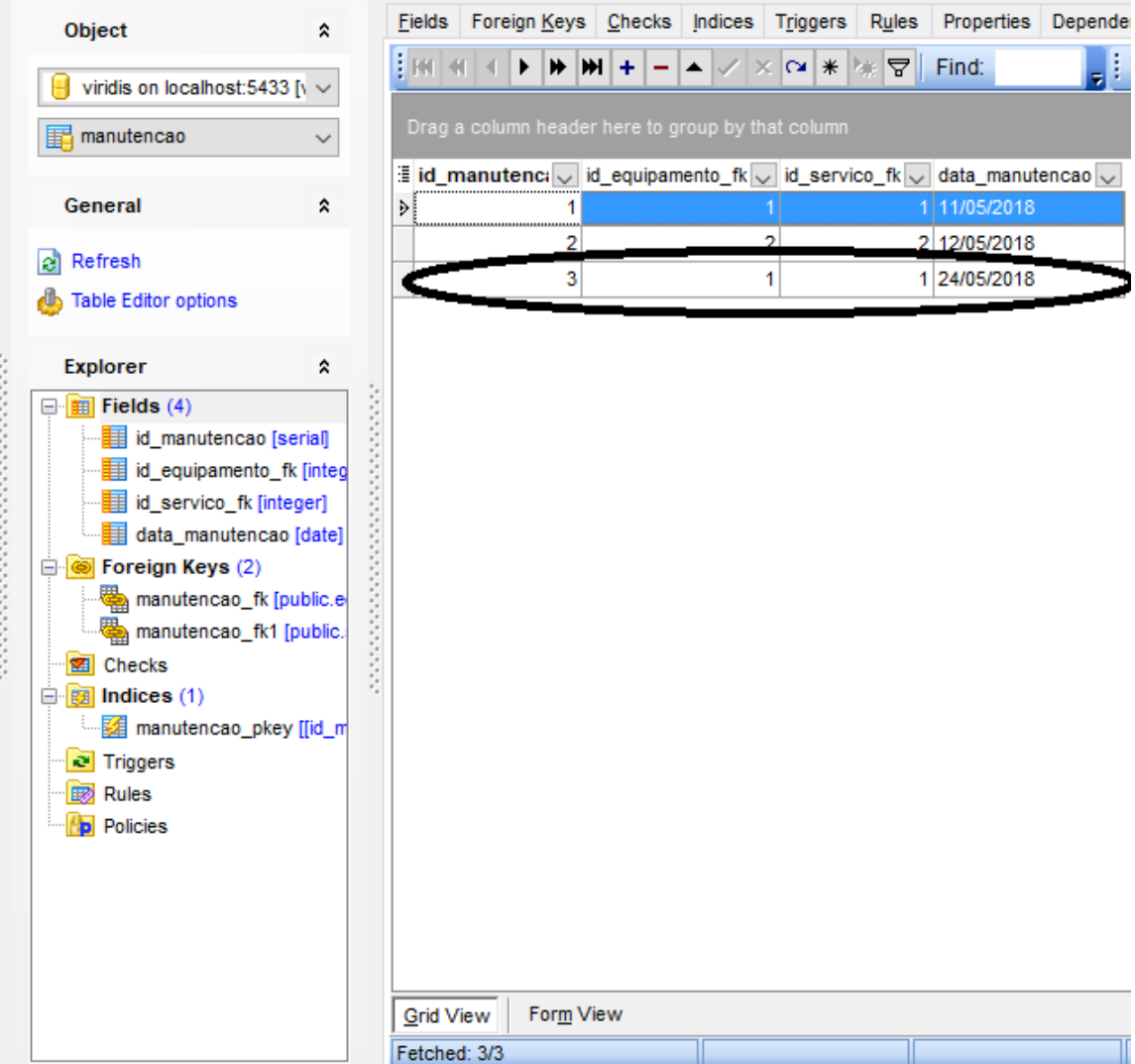


Figura 6: Método para criar nova ordem de Manutenção

Verificando o Banco de Dados vemos que a linha foi inserida corretamente.



The screenshot shows a database management interface with the following components:

- Object Explorer:** Shows the connection 'viridis on localhost:5433' and the database 'manutencao'.
- General Tab:** Includes 'Refresh' and 'Table Editor options' buttons.
- Explorer:** Lists database objects for 'manutencao':
 - Fields (4):** id_manutencao [serial], id Equipamento_fk [integer], id_servico_fk [integer], data_manutencao [date].
 - Foreign Keys (2):** manutencao_fk [public.e], manutencao_fk1 [public.e].
 - Checks:**
 - Indices (1):** manutencao_pkey [[id_m
 - Triggers:**
 - Rules:**
 - Policies:**
- Table Editor:** Displays the 'manutencao' table data in Grid View. The table has 4 columns: id_manutencao, id Equipamento_fk, id_servico_fk, and data_manutencao. It contains 3 rows. The third row is circled.

id_manutencao	id Equipamento_fk	id_servico_fk	data_manutencao
1	1	1	11/05/2018
2	2	2	12/05/2018
3	1	1	24/05/2018

Grid View | Form View
Fetched: 3/3

Figura 7: Dados persistidos no banco Postgres

- Alterar uma ordem de manutenção existente.

Agora iremos alterar o registro inserido anteriormente, passando uma nova data. Desta vez para que já possível concluir a tarefa, a coluna `id_manutencao` deverá ser passada pelo Json, para que o Hibernate saiba qual é a linha que vai sofrer mudanças. Sendo assim passaremos o `id = 3` que foi criado recentemente.

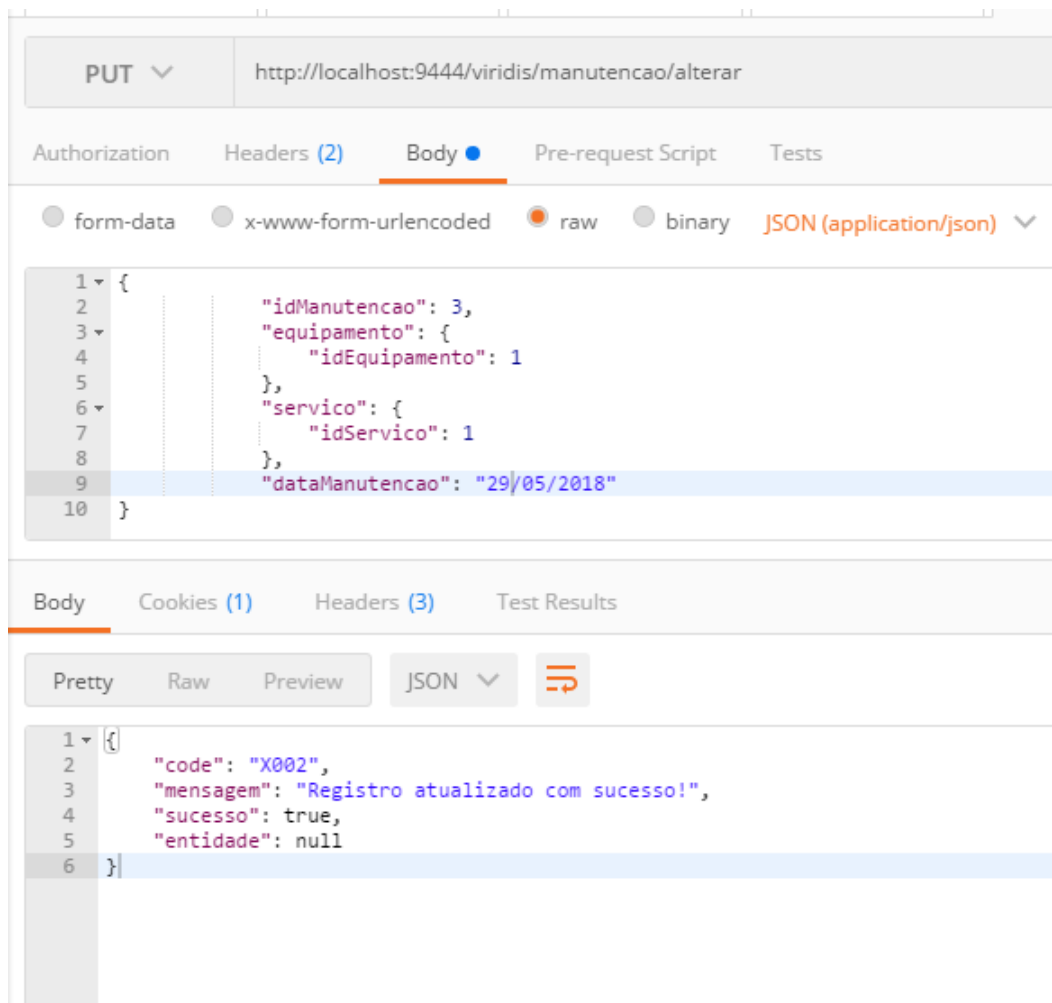


Figura 8: Método para alterar uma ordem de Manutenção existente

E agora, verificando novamente o Banco de Dados vemos que a linha foi alterada.

id_manutencao	id_equipamento	id_servico	data_manutencao
1	1	1	11/05/2018
2	2	2	12/05/2018
3	1	1	29/05/2018

Figura 9: Dados alterados na linha com id = 3

4 Testes Unitários

Foram implementados e utilizados os teste unitários, utilizando o Spring Test, JUnit e Mockito. Segue abaixo a imagem dos testes realizados.

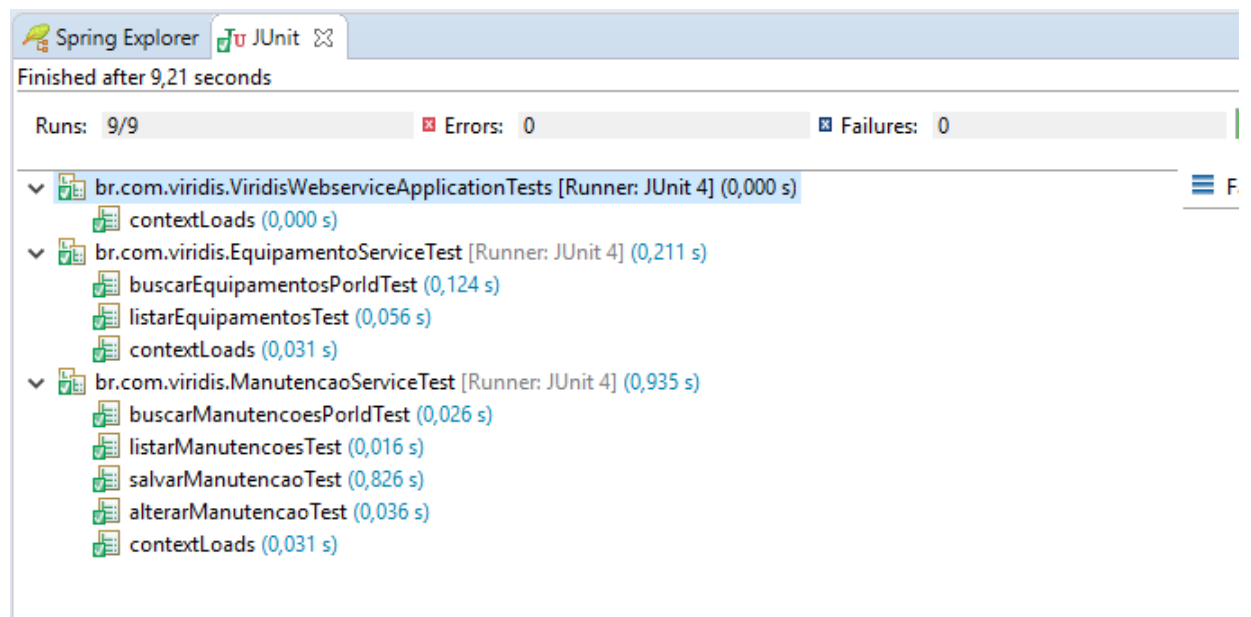


Figura 10: Testes Unitários

5 Documentação da API

Além deste documento descritivo está disponível no Github o código da aplicação, o README para verificar como rodar a aplicação, bem como os scripts para gerar o Banco de Dados. Além disso, pode-se ver uma documentação da API criada através do Swagger, que especifica a lista de recursos que estão disponíveis na API REST e as operações que podem ser chamadas nesses recursos. O mesmo fica disponível assim que a aplicação é inicializada no servidor, seja ele local ou remoto. Além disso, os logs da aplicação gerados pelo SLF4J são salvos na pasta D:\application.log para quem usa o Windows e para alterar o local, basta alterar a linha 14 do "application.properties"

logging.file= D:\application.log

The screenshot shows the Swagger UI for an API named 'Api-Viridis'. The interface has a green header with the Swagger logo, a dropdown menu set to 'default (/v2/api-docs)', and an 'Explore' button. Below the header, the API title 'Api-Viridis' is displayed, followed by a description: 'API Rest com dados para o serviço Viridis.' It also credits 'Created by Otávio Augusto' and provides links to the GitHub repository, contact information, and license. The main content area lists two services: 'equipamento-service : Equipamento Service' and 'manutencao-service : Manutencao Service'. Each service has a 'Show/Hide', 'List Operations', and 'Expand Operations' link. Under 'equipamento-service', there are two GET operations: one for searching equipment by ID and another for listing all equipment. Under 'manutencao-service', there are three operations: a PUT operation to update an existing maintenance order, a GET operation to search for a maintenance order by ID, and a POST operation to register a new maintenance order. At the bottom, it shows the base URL as '/' and the API version as '1.0'.

equipamento-service : Equipamento Service Show/Hide | List Operations | Expand Operations

Method	Path	Description
GET	/viridis/equipamentos/buscar/{id}	Consultar um equipamento através do seu id
GET	/viridis/equipamentos/listar	Listar todos os equipamentos

manutencao-service : Manutencao Service Show/Hide | List Operations | Expand Operations

Method	Path	Description
PUT	/viridis/manutencao/alterar	Alterar uma ordem de manutenção existente
GET	/viridis/manutencao/buscar/{id}	Consultar um manutencao através do seu id
GET	/viridis/manutencao/listar	Listar todos os manutencaos
POST	/viridis/manutencao/salvar	Cadastrar uma nova ordem de manutenção

[BASE URL: / , API VERSION: 1.0]

Figura 11: Documentação da API via Swagger