



FIAP

Leticia Santiago e Silva RM565799

Liana Lyumi Morisita Fujisima RM565698

Victor William Hwan Cho RM565382

JAVA - CHALLENGE

Doctor Ajudar

São Paulo

2025



Sumário

Link do repositório.....	2
Objetivo.....	3
Finalidade.....	3
Funcionalidade.....	4
Tabela de Endpoints.....	5
Protótipo.....	7
Modelo de Entidade-Relacionamento.....	14
Diagrama de Classes Atualizado.....	15

Link do repositório

https://github.com/santiago-leticia/Java_challenge_sp4.git

Objetivo

No conhecimento comum, é de se esperar que um sistema, que pode ser tanto por banco, lojas online ou até hospitais, possua os seus próprios dados presentes, armazenados em um sistema. Com isso, é de se esperar que se pense que, dentro dessas plataformas ou lugares, possuem os seus próprios bancos de dados, nos quais guardam a informação de seus usuários e, sem comentar, a facilidade que oferecem para buscar informação de um usuário específico.

Com essa ideia em mente, foi concordado ao grupo que o projeto de Java seria um meio de os usuários mandarem as suas informações e, a longo prazo, seriam armazenadas no sistema. Isso iria facilitar bastante para saber quando será a próxima consulta dos usuários, quem será o médico responsável pela consulta ou até a capacidade de alterar informações ou adicionar novos pacientes.

Mas o objetivo do projeto é poder ajudar tanto o usuário quanto o funcionário, para facilitar obter informações, deletar, criar em uma forma fácil ou apenas ver quando será a próxima consulta de tal paciente.

Finalidade

Contudo, qual é a finalidade do projeto? Como sabemos, um dos principais problemas presentes no challenge é a dificuldade de pessoas mais velhas de utilizar o site do hospital das clínicas, com isso a finalidade do projeto está junta com essa ideia.

A finalidade presente no projetor possui como sentido facilitar tanto o meio de adicionar ou acessar informações de uma forma simples que o usuário possa entender melhor, ou saber de forma simples, em vez de fazer um sistema demorado ou ter dificuldade para navegar dentro do site.



Funcionalidade

As funcionalidades presentes no projeto, estão relacionadas com as funções simples, como cadastrar, ler relatório de um usuário, excluir ou atualizar

- Adicionar informações de usuário, funcionário ou consulta.
- Excluir informações. Contudo precisa do id, email e senha
- Atualizar informações de tanto por paciente, funcionário ou consultas
- Sistema de reconhecimento de email e senha dentro das funções: excluir, gerar relatório ou atualizar informações.
- Ver suas informações no relatório, contudo precisa do email e senha do usuário
- Capacidade de informar para um usuário ou funcionário:
 - Quando será a próxima consulta que ele terá.
 - Apresentar a Data da consulta ou informações básicas.
 - Informar quem será o responsável que iria resultar a consulta .
 - O Dia da consulta.
 - Informar o horário da consulta.
 - Informações básicas da consulta para o usuário entender o objetivo da consulta.

Tabela de Endpoints

Operações	Método HTTP	URL principal	URL do Endpoint	Descrição	Status de Sucesso	Exceções Comuns
Criar Paciente	POST	/doctorAjuda	/cadastrar/paciente	Cadastrar novo paciente no banco de dados, dentro dela, tem nome_usuario, cpf, telefone, email_usuario e senha_usuario	201 CREATED	500 Internal Server Error, 422 Unprocessable Entity
Criar Funcionário	POST	/doctorAjuda	/cadastrar/funcionario	Cadastrar novo Funcionário no banco de dados, dentro dela, tem nome_funcionario, tipo_funcionario, email_funcionario e senha_funcionario.	201 CREATED	500 Internal Server Error, 422 Unprocessable Entity
Criar Consulta	POST	/doctorAjuda	/cadastrar/consulta	Cadastrar nova consulta no banco de dados, dentro dela, tem id_paciente, id_funcionario, data_consulta, horas_consultas e informacao_consulta	201 CREATED	500 Internal Server Error, 422 Unprocessable Entity
Buscar usuário	GET	/doctorAjuda	/relatorio/paciente	A para saber informação do usuário, apenas vai precisar colocar o email e senha do usuário para ver as informações. Essa parte é importante, porque aí vai conseguir o id.	200 OK	404 Not Found, 500 Internal Server Error
Buscar Funcionário	GET	/doctorAjuda	/relatorio/funcionario	A para saber informação do Funcionário, apenas vai precisar colocar o email e senha do Funcionário para ver as informações. Essa parte é importante, porque aí vai conseguir o id.	200 OK	404 Not Found, 500 Internal Server Error
Buscar Consulta	GET	/doctorAjuda	/relatorio/consulta	A para saber informação do Consulta, apenas vai precisar colocar o email e senha do usuário para ver as informações. Essa parte é importante, porque aí vai conseguir o id.	200 OK	404 Not Found, 500 Internal Server Error
Remover Paciente	DELETE	/doctorAjuda	/deletar/paciente	Para deletar o usuário do paciente, apenas precisa do id, email e senha do usuário. Essa parte é importante, porque aí vai conseguir o id.	200 OK	404 Not Found, 500 Internal Server Error
Remover Funcionário	DELETE	/doctorAjuda	/deletar/funcionario	Para deletar o funcionário, apenas precisa do id, email e senha do funcionário.	200 OK	404 Not Found, 500 Internal Server Error



Remover Consulta	DELETE	/doctorAjuda	/deletar/consulta	Para deletar o funcionário, apenas precisa do id da consulta, email e senha do usuário.	200 OK	404 Not Found, 500 Internal Server Error
Atualizar Paciente	PUT	/doctorAjuda	/atualizar/paciente	Para atualizar as informações, precisa saber do id do usuário que primeiro precisa ver o relatório do usuário para saber qual é.	200 OK	400 Bad Request, 500 Internal Server Error
Atualizar Funcionário	PUT	/doctorAjuda	/atualizar/funcionario	Para atualizar as informações, precisa saber do id do funcionário que primeiro precisa ver o relatório do funcionário para saber qual é.	200 OK	400 Bad Request, 500 Internal Server Error
Atualizar Consulta	PUT	/doctorAjuda	/atualizar/consulta	Para atualizar as informações, precisa saber do id da consulta que primeiro precisa ver o relatório da consulta para saber qual é.	200 OK	400 Bad Request, 500 Internal Server Error

Protótipo

No Postman

The screenshot shows a Postman interface with a POST request to `http://localhost:8080/doctorAjuda/cadastrar/paciente`. The Body tab is selected and contains the following JSON payload:

```

1  {
2    "nome_usuario": "Letícia S e S",
3    "cpf": "31231231",
4    "telefone": "1153453",
5    "email_usuario": "le@gmail.com",
6    "senha_usuario": "12345"
7
8  }
9
10

```

Below the request, the response is shown with a status of `201 Created`, a duration of `280 ms`, and a size of `374 B`. The response body is:

```

1 Criando com sucesso

```

Na parte da criação em todas as partes será necessário que coloque corretamente as informações corretas. Mas, caso queira ver as informações do usuário deve ir no relatório para assim conseguir saber do id, utilizando email e senha.

The screenshot shows a Postman interface with a GET request to `http://localhost:8080/doctorAjuda/relatorio/paciente`. The Body tab is selected and contains the following JSON payload:

```

1  {
2    "email_usuario": "le@gmail.com",
3    "senha_usuario": "12345"
4  }
5
6

```

Below the request, the response is shown with a status of `200 OK`, a duration of `58 ms`, and a size of `493 B`. The response body is:

```

1 [
2   {
3     "id_usuario": 1,
4     "nome_usuario": "Letícia S e S",
5     "cpf": "31231231",
6     "telefone": "1153453",
7     "email_usuario": "le@gmail.com",
8     "senha_usuario": "12345"
9   }
10 ]

```



Mas caso ou email ou cpf, já esteja no sistema vai ser avisado ao usuário que já existe.

The screenshot shows a POST request in Postman. The URL is `http://localhost:8080/doctorAjuda/cadastrar/paciente`. The Body tab is selected, showing a JSON payload:

```
1 {  
2   "nome_usuario": "Letícia S e S",  
3   "cpf": "31231231",  
4   "telefone": "1153453",  
5   "email_usuario": "le@gmail.com",  
6   "senha_usuario": ""}  
7  
8 }  
9  
10
```

The response status is **422 Unprocessable Entity**, with a response time of 209 ms and a response size of 387 B. The response body is: `1 Email ja cadastrado`.

Mas também ocorrer um aviso caso se usuário esquecer de colocar algo

The screenshot shows a POST request in Postman. The URL is `http://localhost:8080/doctorAjuda/cadastrar/paciente`. The Body tab is selected, showing the following JSON payload:

```
1 {  
2   "nome_usuario": "Leticia S e S",  
3   "cpf": "",  
4   "telefone": "1153453",  
5   "email_usuario": "le@.com",  
6   "senha_usuario": ""  
7 }  
8  
9  
10
```

The response status is **422 Unprocessable Entity**, with a response time of 432 ms and a response size of 381 B. The response body contains the message: **1 Cpf incorreto**.

Para poder deletar uma informação será apenas necessário o id do usuário ou do funcionário, com email e senha de ambos.

Mas na consulta para deletar será apenas necessário o id da consulta, mas o email e senha do usuário. Como uma forma de calcular uma consulta.

The screenshot shows a Postman interface for a DELETE request to the endpoint `http://localhost:8080/doctorAjuda/deletar/paciente`. The request body is a JSON object with fields: `"id_usuario": 1`, `"email_usuario": "le@gmail.com"`, and `"senha_usuario": "12345"`. The response status is `200 OK` with a response message `1 Removido com sucesso`.

```
1 {  
2   "id_usuario": 1,  
3   "email_usuario": "le@gmail.com",  
4   "senha_usuario": "12345"  
5 }  
6  
1 Removido com sucesso
```

Na parte de atualizar informações, será apenas pedido o id que será só mostrando no relato

The screenshot shows a Postman interface with the following details:

- HTTP Method:** PUT
- URL:** `http://localhost:8080/doctorAjuda/atualizar/paciente`
- Body (JSON):**

```
1 {  
2     "id_usuario":2,  
3     "nome_usuario":"dasda",  
4     "cpf":"fsfs",  
5     "telefone":"4234324",  
6     "email_usuario":"santiagoleticia",  
7     "senha_usuario":"1242"  
8 }  
9  
10
```
- Response Status:** 200 OK
- Response Time:** 60 ms
- Response Size:** 367 B
- Response Headers:** (not explicitly shown)
- Response Body:** (empty)

O relato da consulta

The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** <http://localhost:8080/doctorAjuda/relatorio/consulta>
- Body (JSON):**

```
1  {
2    "email_usuario": "santiagoleticia",
3    "senha_usuario": "1242"
4  }
5
```
- Response Status:** 200 OK
- Response Body (JSON):**

```
1  [
2    {
3      "id_usuario": 0,
4      "nome_usuario": "2",
5      "cpf": "dasda",
6      "telefone": "fsfs",
7      "email_usuario": "4234324",
8      "senha_usuario": "santiagoleticia",
9      "id_consulta": 2,
10     "id_funcionario": 1242,
11     "nome_funcionario": "1",
12     "data_consulta": "0002-10-20 00:00:00",
13     "horas_consulta": "15:00",
14     "informacao_consulta": "Exame de sangue"
15   }
16 ]
```

FIAP

Caso se o erro 500 se mostrar vai apresentar isso

The screenshot shows a Postman interface with a POST request to `http://localhost:8080/doctorAjuda/cadastrar/consulta`. The request body is a JSON object with fields: `"id_paciente":2, "id_funcionario":1, "data_consulta":"2025-10-02", "horas_consultas":"15:00", "informacao_consulta":"Exame de sangue"`. The response is a 500 Internal Server Error with the message `{"erro": "Erro no servidor"}`.

POST http://localhost:8080/doctorAjuda/cadastrar/consulta Send

Params Auth Headers (8) Body Scripts Settings Cookies Beautify

raw JSON

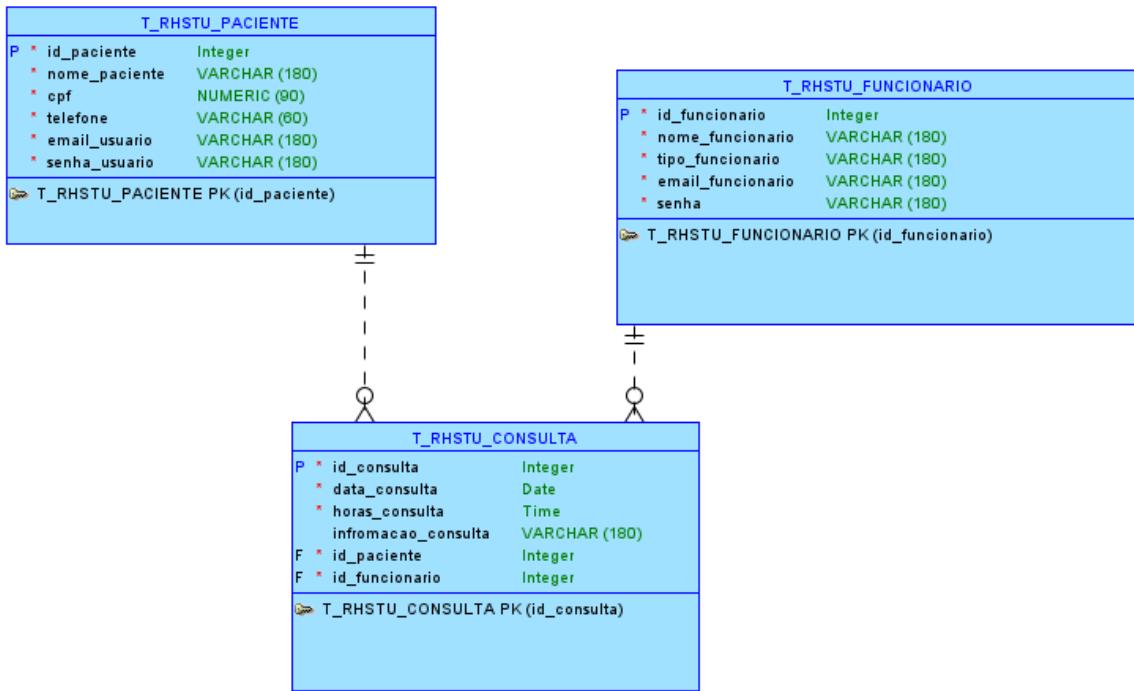
```
1 {  
2     "id_paciente":2,  
3     "id_funcionario":1,  
4     "data_consulta":"2025-10-02",  
5     "horas_consultas":"15:00",  
6     "informacao_consulta":"Exame de sangue"  
7 }  
8  
9
```

Body 500 Internal Server Error 130 ms 396 B [Raw](#) [Copy](#) [...](#)

{ } JSON Preview Debug with AI [Raw](#) [Copy](#) [Link](#)

```
1 {  
2     "erro": "Erro no servidor"  
3 }
```

Modelo de Entidade-Relacionamento



Ativar

FIAP

Diagrama de Classes Atualizado

