

# Teste Prático – DSIN

Vaga: Desenvolvimento – Estágio

Nome: Mathäus Lazarus

Email: [lazarusmathaus@gmail.com](mailto:lazarusmathaus@gmail.com)

- O Código do projeto também está no meu Github: <https://github.com/lazarusms>
- Esse PDF contém os prints das requisições e das respostas da aplicação realizados pelo Postman (como solicitado)
- Optei por colocar tudo em um PDF para ficasse mais organizado e pudesse ter uma explicação em cada requisição de como ela se relaciona com o que foi pedido na atividade '**Cabeleira Leila Salão de Beleiza**'.
- Os endpoints também podem ser acessados pelo Swagger, ao rodar a aplicação enviada, através da URL:  
**<http://localhost:8080/swagger-ui/index.html>**
- Utilizei Java e o SpringBoot. O banco de dados utilizado foi o H2 por praticidade.

Transformei os pedidos da Leila em pequenas tarefas e fiz o código em cima disso.

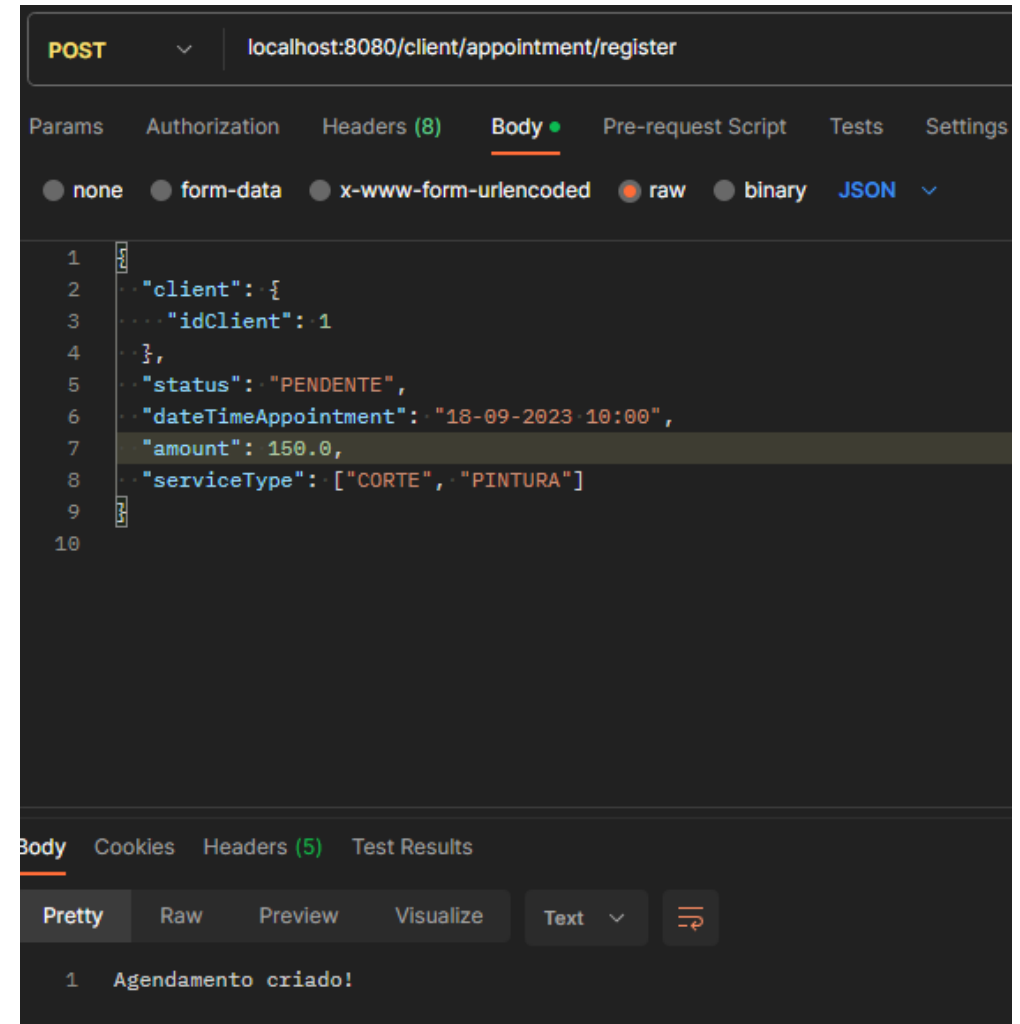
A aplicação inicia com **dois clientes** e o manager '**Leila**' cadastrados no banco de dados

```
@Configuration
public class StdConfiguration {

    @Bean
    CommandLineRunner commandLineRunner(ClientRepository clientRepository, ManagerRepository managerRepository) {
        return args -> {
            Client newClient = new Client(
                firstName: "Mathaus", Typo: In word 'Mathaus'.
                lastName: "Lazarus",
                LocalDate.of( year: 2000, MARCH, dayOfMonth: 5),
                Role.CLIENTE,
                email: "mathaus@email.com",
                phoneNumber: "119895119232");
            clientRepository.save(newClient);
            Client newClientNew = new Client(
                firstName: "Victoria",
                lastName: "Lazarus",
                LocalDate.of( year: 2000, AUGUST, dayOfMonth: 15),
                Role.CLIENTE,
                email: "victoria@email.com",
                phoneNumber: "119895119232");
            clientRepository.save(newClientNew);
            Manager manager = new Manager(
                Role.ADMIN,
                name: "Leila");
            managerRepository.save(manager);
        };
    }
}
```

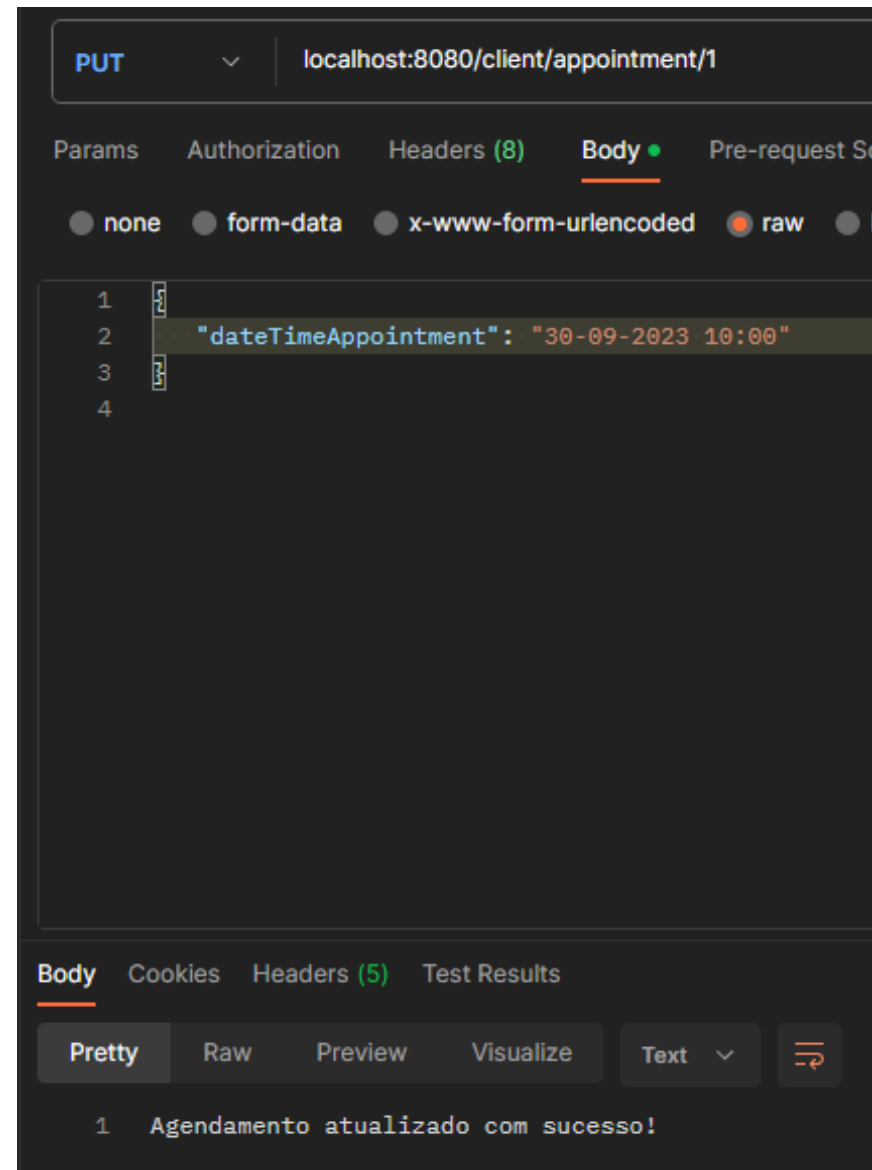
**Tarefa:** Agendamento de um ou mais serviços, sendo possível agendar mais de um tipo de serviço por vez

- Isso é realizado através do **POST** `/cliente/register-appointment`
- Podem ser passados mais de um “ServiceType” por vez.



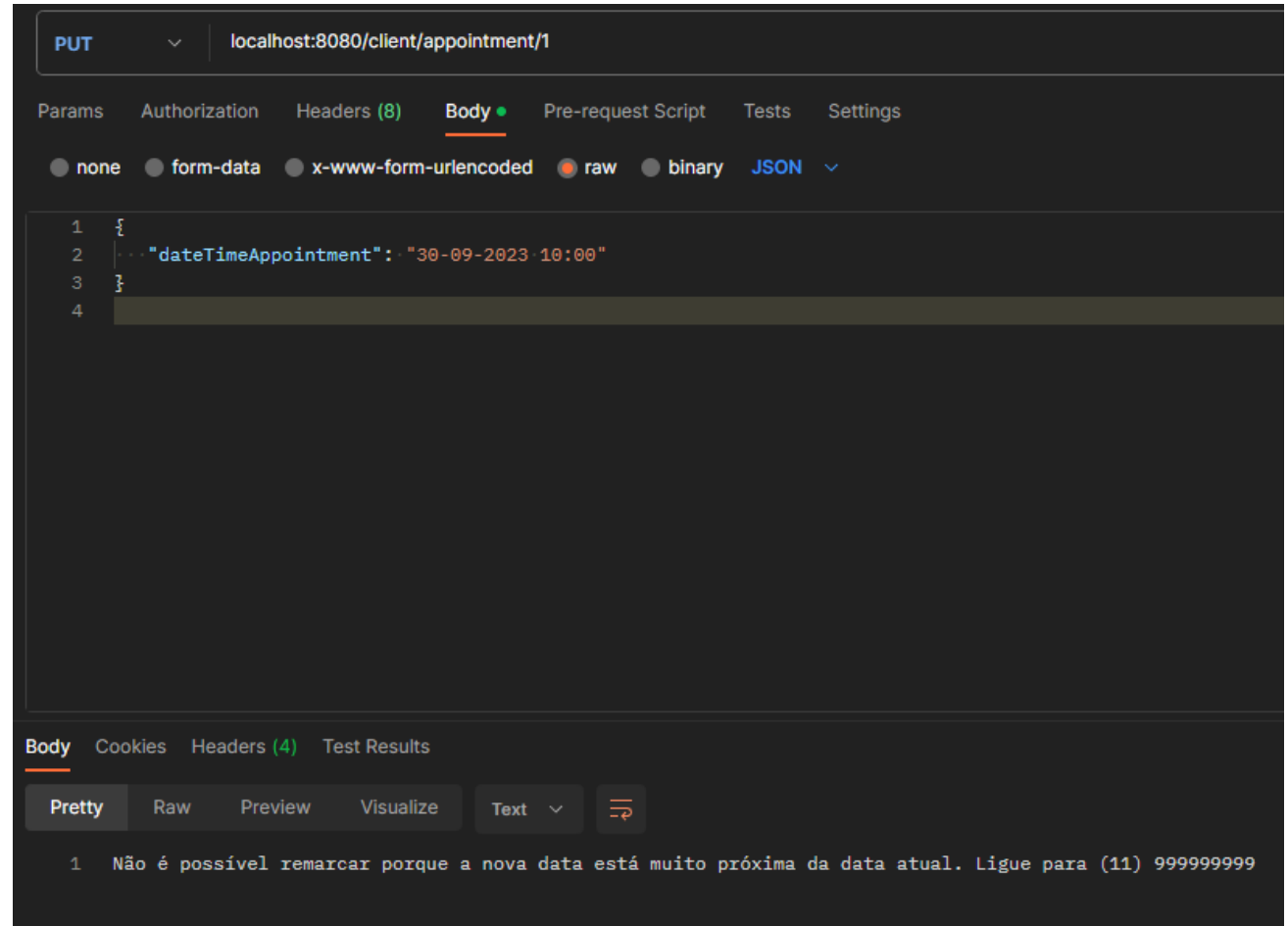
## Tarefa: Realizar alterações nesses agendamentos

- Isso é realizado através do **PUT** `/client/appointment/{appointmentId}`
- Podem ser alterados qualquer informação do agendamento, inclusive o STATUS, assim que realizado o procedimento.



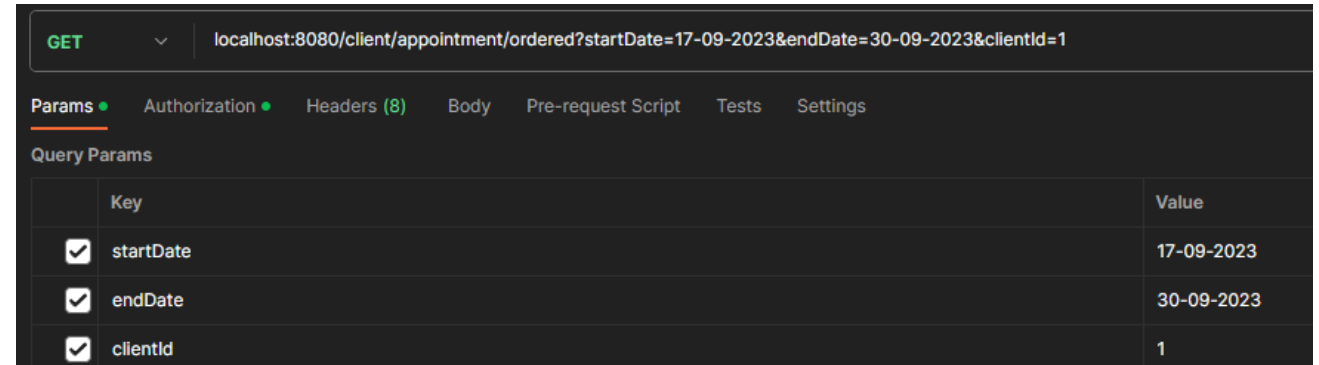
## Tarefa: Permitir alterações apenas 2 dias antes do agendamento

- É realizado um **PUT** `/client/appointment/{appointmentId}`
- Como pedido, só é possível alterar o dia de um agendamento se ele estiver há mais de dois dias de diferença do dia atual. Por exemplo, hoje é dia 17-09, e o agendamento estava marcado para amanhã, dia 18-09. Por isso, a alteração não pode ser realizada pelo site, mesmo que a nova data seja mais, por exemplo, dia 30-09, como na imagem. É necessário ligar.
- Eu realizei essa validação apenas para remarcar o agendamento, mas é possível, por exemplo, incluir outros serviços.



## Tarefa: Histórico de agendamentos realizados em determinado período (por cliente)

- **GET /client/appointment/ordered**
- Como a ideia é filtrar por período, é necessário passar o dia inicial, o último dia e o Id do cliente
- Também fiz um endpoint para o manager para filtrar, **em toda a lista de agendamentos**, os do período informado. Diferente desse que só filtra para o cliente informado.



GET	localhost:8080/client/appointment/ordered?startDate=17-09-2023&endDate=30-09-2023&clientId=1
Params	Authorization Headers (8) Body Pre-request Script Tests Settings
Query Params	
Key	Value
<input checked="" type="checkbox"/> startDate	17-09-2023
<input checked="" type="checkbox"/> endDate	30-09-2023
<input checked="" type="checkbox"/> clientId	1

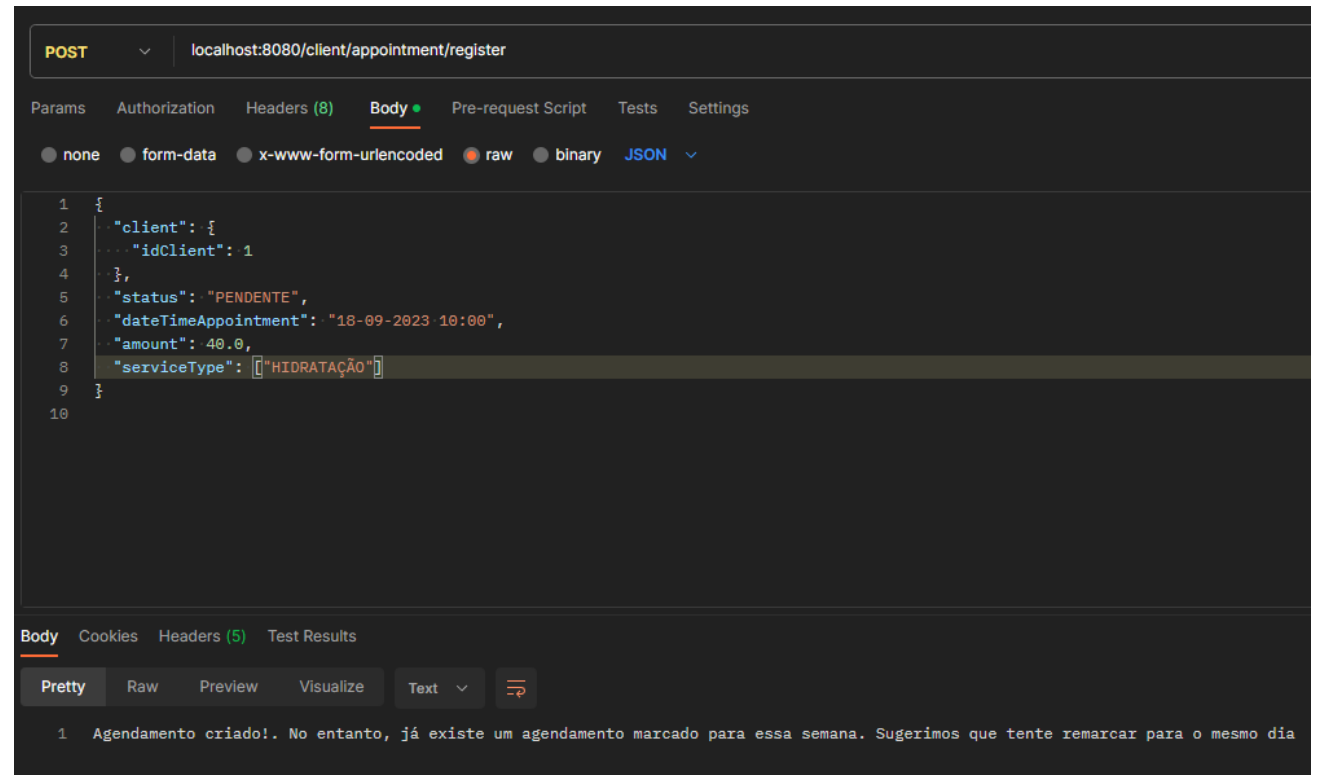
- Retorna todos os agendamentos do cliente informado no período solicitado

```
{
  "idAppointment": 1,
  "client": {
    "idClient": 1,
    "firstName": "Mathaus",
    "lastName": "Lazarus",
    "dateOfBirth": "2000-02-05",
    "role": "CLIENTE",
    "email": "mathaus@email.com",
    "phoneNumber": "119895119232"
  },
  "status": "PENDENTE",
  "dateTimeAppointment": "17-09-2023 10:00",
  "amount": 150.00,
  "serviceType": [
    "CORTE",
    "PINTURA"
  ]
}
```

```
{
  "idAppointment": 2,
  "client": {
    "idClient": 1,
    "firstName": "Mathaus",
    "lastName": "Lazarus",
    "dateOfBirth": "2000-02-05",
    "role": "CLIENTE",
    "email": "mathaus@email.com",
    "phoneNumber": "119895119232"
  },
  "status": "PENDENTE",
  "dateTimeAppointment": "27-09-2023 10:00",
  "amount": 70.00,
  "serviceType": [
    "CORTE"
  ]
}
```

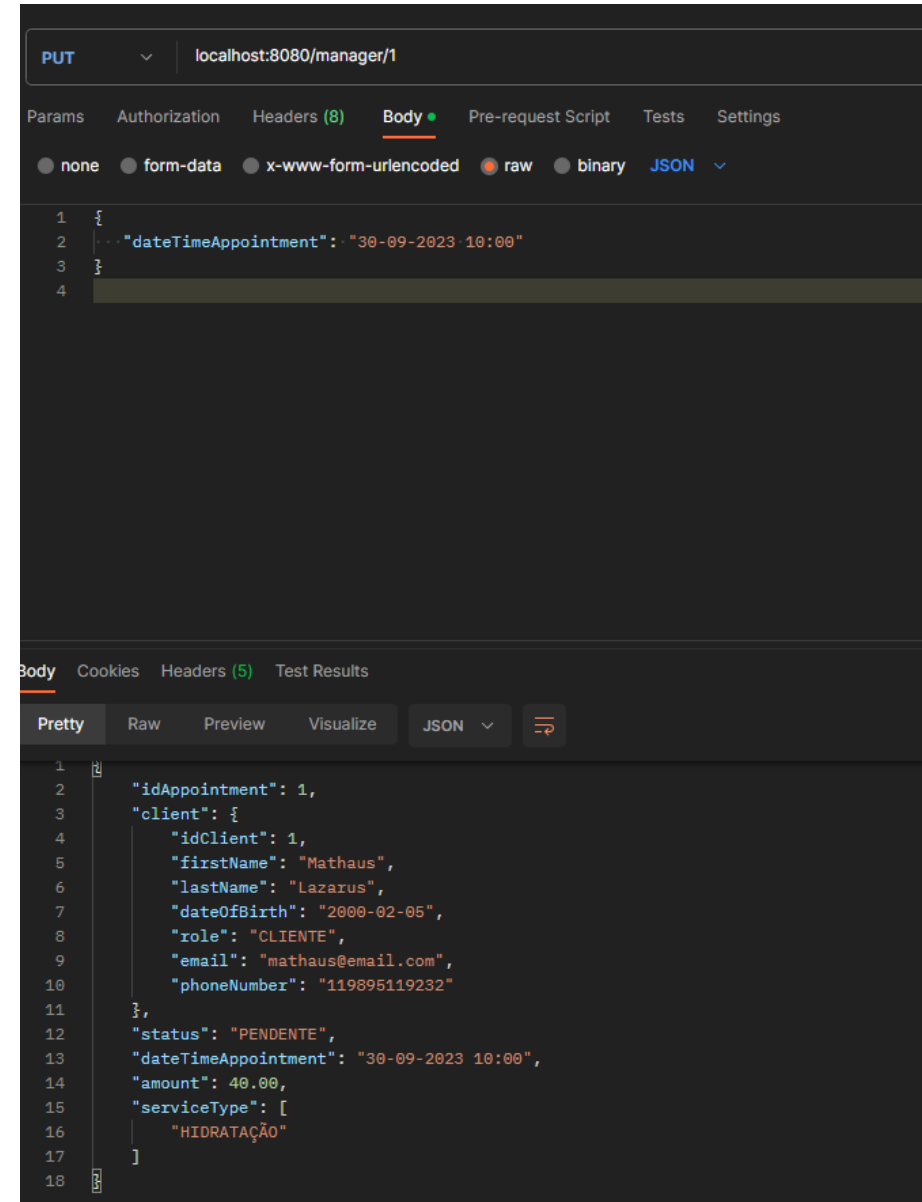
**Tarefa:** Identificar caso já exista um agendamento do cliente para aquela semana e sugerir que o novo agendamento seja marcado para mesma data (do primeiro agendamento)

- No caso, já existia, um agendamento para o dia **17-09-2023**, então o sistema sugere ao cliente que seja alterado para o mesmo dia.
- No entanto, optei por deixar o agendamento ser criado e apenas passar a sugestão ao cliente.
- Isso é feito através de uma validação do **POST client/appointment/register**



## Tarefa opcional: Na parte operacional, Leila gostaria de ter acesso para alterar os agendamentos

- Isso é feito através de uma requisição **PUT** **/manager/{appointmentId}**
- Diferente do método do cliente, esse método não tem a validação dos 'dois dias'. Para que Leila possa alterar a data quando um cliente ligar solicitando



The screenshot displays a REST client interface with a PUT request to `localhost:8080/manager/1`. The 'Body' tab is active, showing a JSON payload. Below the request, the 'Response' tab is also active, displaying the returned JSON data in a pretty-printed format.

```
1 {
2   ... "dateTimeAppointment": "30-09-2023 10:00"
3 }
4
```

```
1 {
2   "idAppointment": 1,
3   "client": {
4     "idClient": 1,
5     "firstName": "Mathaus",
6     "lastName": "Lazarus",
7     "dateOfBirth": "2000-02-05",
8     "role": "CLIENTE",
9     "email": "mathaus@email.com",
10    "phoneNumber": "119895119232"
11  },
12  "status": "PENDENTE",
13  "dateTimeAppointment": "30-09-2023 10:00",
14  "amount": 40.00,
15  "serviceType": [
16    "HIDRATAÇÃO"
17  ]
18 }
```



## Tarefa opcional: Uma listagem dos agendamentos recebidos

- Isso é feito através de uma requisição **GET** `/manager/upcoming-list`
- Nesse caso, eu deixei a lista em ordem de próximo atendimento (o que pode ser alterado depois). Para que Leila saiba qual é o próximo cliente.

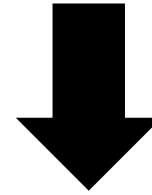
```
GET localhost:8080/manager/upcoming-list

[
  {
    "idAppointment": 3,
    "client": {
      "idClient": 1,
      "firstName": "Mathaus",
      "lastName": "Lazarus",
      "dateOfBirth": "2000-02-05",
      "role": "CLIENTE",
      "email": "mathaus@email.com",
      "phoneNumber": "119895119232"
    },
    "status": "PENDENTE",
    "dateTimeAppointment": "20-09-2023 10:00",
    "amount": 40.00,
    "serviceType": [
      "HIDRATAÇÃO"
    ]
  },
  {
    "idAppointment": 2,
    "client": {
      "idClient": 1,
      "firstName": "Mathaus",
      "lastName": "Lazarus",
      "dateOfBirth": "2000-02-05",
      "role": "CLIENTE",
      "email": "mathaus@email.com",
      "phoneNumber": "119895119232"
    },
    "status": "PENDENTE",
    "dateTimeAppointment": "23-09-2023 10:00",
    "amount": 40.00,
    "serviceType": [
      "HIDRATAÇÃO"
    ]
  }
]
```

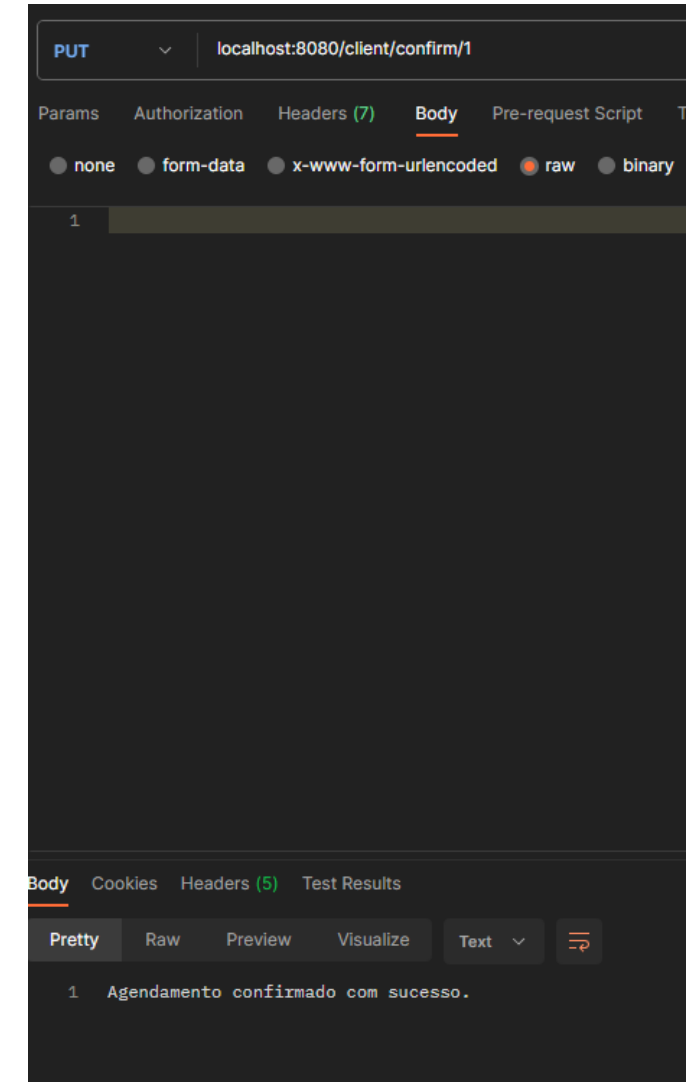
## Tarefa opcional: Possibilitar a confirmação do agendamento ao cliente

- Isso é feito através de uma requisição **PUT** `/client/confirm/{appointmentId}`
- O status do agendamento é confirmado através de uma requisição PUT

```
{
  "idAppointment": 1,
  "client": {
    "idClient": 1,
    "firstName": "Mathaus",
    "lastName": "Lazarus",
    "dateOfBirth": "2000-02-05",
    "role": "CLIENTE",
    "email": "mathaus@email.com",
    "phoneNumber": "119895119232"
  },
  "status": "PENDENTE",
  "dateTimeAppointment": "30-09-2023 10:00",
  "amount": 40.00,
  "serviceType": [
    "HIDRATAÇÃO"
  ]
}
```

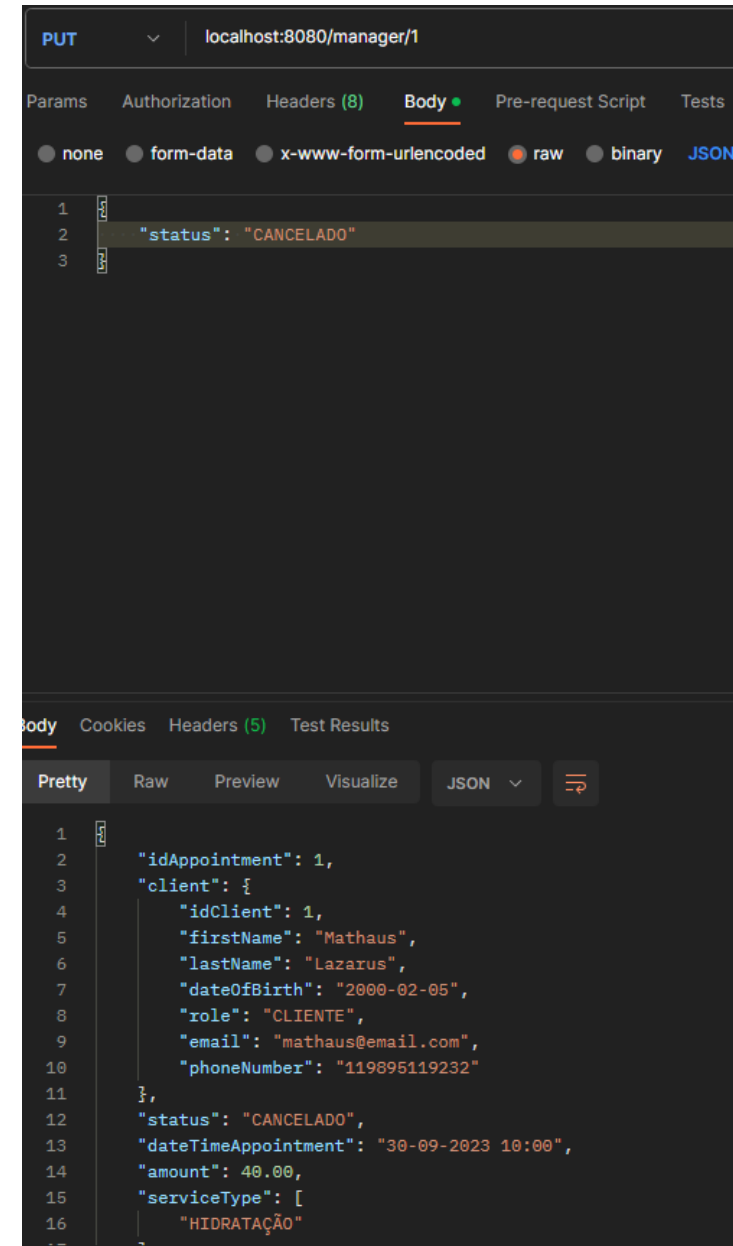


```
{
  "idAppointment": 1,
  "client": {
    "idClient": 1,
    "firstName": "Mathaus",
    "lastName": "Lazarus",
    "dateOfBirth": "2000-02-05",
    "role": "CLIENTE",
    "email": "mathaus@email.com",
    "phoneNumber": "119895119232"
  },
  "status": "CONFIRMADO",
  "dateTimeAppointment": "30-09-2023 10:00",
  "amount": 40.00,
  "serviceType": [
    "HIDRATAÇÃO"
  ]
}
```



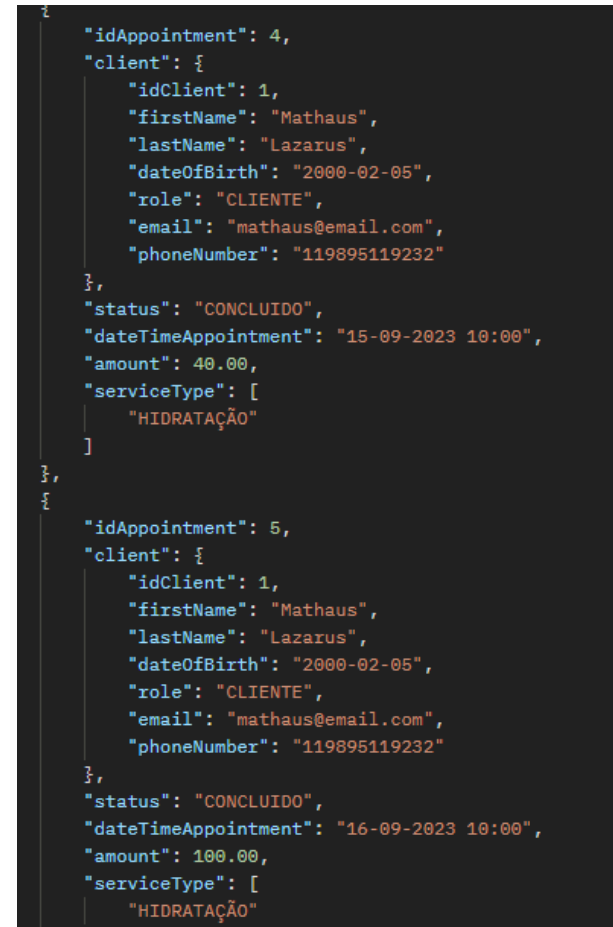
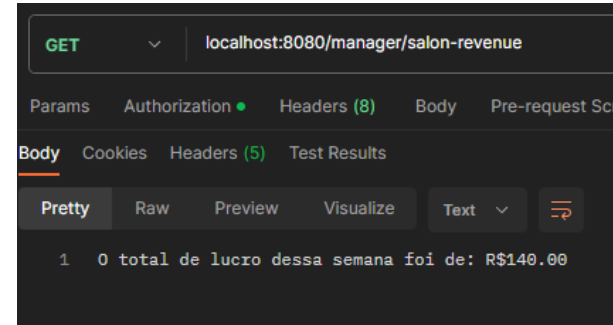
## Tarefa opcional: Gerenciamento do status dos serviços solicitados (manager)

- Isso é feito através de uma requisição **PUT** `/manager/{appointmentId}`
- Nesse caso, esse gerenciamento pode ser feito pelo mesmo método de update dos agendamentos, alterando assim o status do agendamento em questão.



## Tarefa opcional: Ferramenta que possibilite o acompanhamento do negócio, por desempenho semanal

- Isso é feito através de uma requisição **GET** `/manager/salon-revenue`
- Essa ferramenta vai pegar apenas os agendamentos **CONCLUIDOS** de uma semana atrás, a partir da data local, e então somar e apresentar lucro semanal.
- Por esse motivo, de precisar passar agendamentos “passados”, foi que eu realizei esse tipo de validação no método de `registerAppointment`
- No caso apresentado, seria a soma apenas dos agendamentos 4 e 5. Os outros não estão como concluídos.



# Swagger Documentação

É possível acessar e testar os endpoints pelo Swagger.

Rodando a aplicação do Spring Boot e acessando:

**<http://localhost:8080/swagger-ui/index.html>**

manager-controller ^	
GET	/manager/{appointmentId} v
PUT	/manager/{appointmentId} v
DELETE	/manager/{appointmentId} v
POST	/manager/register-appointment v
GET	/manager/upcoming-list v
GET	/manager/salon-revenue v
GET	/manager/list v
GET	/manager/list/ordered v
appointment-client-controller ^	
PUT	/client/confirm/{appointmentId} v
GET	/client/appointment/{appointmentId} v
PUT	/client/appointment/{appointmentId} v
DELETE	/client/appointment/{appointmentId} v
POST	/client/appointment/register v
GET	/client/info/{clientId} v
GET	/client/appointment/ordered v
client-controller ^	
GET	/client/config/{clientId} v
PUT	/client/config/{clientId} v
DELETE	/client/config/{clientId} v
POST	/client/config/register v
GET	/client/config v