Teste Técnico

Responsável	Leve Saúde
Verificação	Vazio
Tags	Guias e Processos Dev
Última edição	quinta-feira 10:43
Data	Vazio

Ocultar 3 propriedades

Teste Técnico - Desenvolvedor Backend

Entrega:

Por favor, submeta o código fonte no seu repositório Git contendo as instruções para teste local no arquivo README.

Objetivo:

Desenvolver uma API utilizando **Node.js** com **TypeScript**, **Serverless Framework** e **AWS Lambda**. A API terá dois endpoints:

1. **Buscar agendas e horários dos médicos**.

2. **Marcar agendamento do paciente**.

Os dados retornados deverão ser **mockados** (não precisam ser integrados a uma base de dados real). O foco é avaliar sua capacidade técnica, domínio de boas práticas e uso correto de ferramentas e tecnologias.

Instruções:

- Utilize **Node.js** (v14 ou superior) e **TypeScript**.
- A API deve ser criada utilizando o **Serverless Framework com o plugin serveless-offline para teste local**.
- As funções devem ser implementadas como **AWS Lambda e as triggers Rest API (AWS Api Gateway)**.
- Mockar as respostas dos endpoints com dados fictícios, mas em formato realista.
- Siga boas práticas de codificação e arquitetura.
- Entregar o código em um repositório Git (público), com instruções no README para rodar o projeto.

Descrição da API:

1. **Endpoint: Buscar agendas e horários dos médicos**
   - **Rota**: `GET /agendas`
   - **Descrição**: Retorna uma lista de médicos com suas respectivas agendas e horários disponíveis.
   - **Resposta esperada**:

```
{ "medicos": [ { "id": 1, "nome": "Dr. João Silva", "especialidad
e": "Cardiologista", "horarios_disponiveis": [ "2024-10-05 09:00",
"2024-10-05 10:00", "2024-10-05 11:00" ] }, { "id": 2, "nome": "Dr
a. Maria Souza", "especialidade": "Dermatologista", "horarios_disp
oniveis": [ "2024-10-06 14:00", "2024-10-06 15:00" ] } ] }
```

2. **Endpoint: Marcar agendamento do paciente**
   - **Rota**: `POST /agendamento`
   - **Descrição**: Permite que o paciente marque um horário de consulta com um médico.
   - **Payload esperado**:

```
{ "medico_id": 1, "paciente_nome": "Carlos Almeida", "data_horario": "2024-10-05 09:00" }
```

   - **Resposta esperada**:

```
{ "mensagem": "Agendamento realizado com sucesso", "agendamento": { "medico": "Dr. João Silva", "paciente": "Carlos Almeida", "data_horario": "2024-10-05 09:00" } }
```

Requisitos Técnicos:

Funcionais:

- **Endpoints REST** implementados como **funções Lambda**.
- Respostas mockadas, sem necessidade de interação com banco de dados real.
- Seguir boas práticas de RESTful APIs.
- Implementar **testes unitários** para a lógica de negócio e validações com o **Jest**.
- Incluir **validação de payloads** nas requisições, com retorno de erros adequados (ex: 400 para requisições inválidas).

Boas Práticas:

- Utilizar **TypeScript** em todo o projeto.
- Criar uma **documentação clara** no README, incluindo como rodar o projeto localmente e como fazer deploy.

Estrutura de Pastas (sugestão):

```
bash /src /utils # Funções utilitárias /agenda # Domínio de negócio /cont
roller /service /dto /interface /mocks # Dados mockados /agendamento # Do
mínio de negócio /controller /service /dto /interface /mocks # Dados mock
ados
```

Extras:

- Utilizar **prettier** e **eslint** para garantir a qualidade do código.

Critérios de Avaliação:

1. **Qualidade do código**: Estruturação, modularidade, legibilidade e aderência a boas práticas.
2. **Domínio de TypeScript**: Tipagem correta, uso de interfaces, classes, etc.
3. **Conhecimento de Serverless**: Configuração e implementação de funções Lambda utilizando o Serverless Framework.
4. **Boas práticas REST**: Design de rotas e tratamento de erros.
5. **Documentação**: Clareza das instruções e documentação do projeto.
6. **Testes**: Implementação de testes unitários e validação da lógica.
