Nome: João Marcos Melo Monteiro

01817269127

Estrutura Inicial do Projeto

1. **Configuração do Ambiente de Trabalho**
   - Instale a versão necessária do Node.js (v22.x).
   - Configure o AWS CLI e as credenciais para autenticação com os serviços da AWS.
   - Instale o DynamoDB local (ou utilize o serviço real durante os testes).
2. **Estrutura do Repositório**
   - Um diretório principal com a seguinte estrutura inicial:

```
scss
Copiar código
├── src/
│   ├── handlers/
│   ├── services/
│   ├── models/
│   ├── utils/
│   └── tests/
├── .gitignore
├── package.json
├── tsconfig.json
├── README.md
└── serverless.yml (caso inclua o CloudFormation ou
Serverless Framework)
```

Pontos-Chave para o Desenvolvimento

1. **CRUD de Clientes**
   - **EndPoints básicos**:
     - `POST /clientes` → Criar cliente.
     - `GET /clientes/{id}` → Consultar cliente.
     - `PUT /clientes/{id}` → Atualizar cliente.
     - `DELETE /clientes/{id}` → Remover cliente.
     - `GET /clientes` → Listar todos os clientes (opcional, mas interessante).
   - **Regras de Negócio**:
     - Validação de campos (e.g., um contato principal obrigatório).
     - O status (`ativo/inativo`) deve ser armazenado como booleano.
2. **Modelos de Dados**
   - **Tabela DynamoDB**:
     - `PK` → `Cliente#<id>` (partition key).
     - `SK` → `Metadata` (para armazenar dados principais) ou `Endereco#<id>` / `Contato#<id>`.
   - **Classe Cliente**:

        - Propriedades: `nomeCompleto`, `dataNascimento`, `status`, `enderecos`, `contatos`.
        - Métodos: `criar`, `atualizar`, `deletar`, etc.
3. **Arquitetura Serverless**
    - **AWS Lambda**:
        - Uma função para cada operação (ou uma função única com roteamento interno).
    - **API Gateway**:
        - Configuração com proxy integration para facilitar o roteamento para o Lambda.
    - **DynamoDB**:
        - Armazene os clientes, endereços e contatos de forma eficiente.
4. **Testes**
    - **Unitários**:
        - Teste classes e métodos.
    - **Integração**:
        - Teste chamadas da API com o DynamoDB local.
    - **Cobertura de Testes**:
        - Use ferramentas como `nyc` para medir.
5. **CloudFormation**
    - Descreva a configuração dos recursos:
        - Tabela DynamoDB.
        - Funções Lambda.
        - API Gateway.

Diferenciais

- **Template CloudFormation**: Inclua um arquivo `serverless.yml` ou `cloudformation.yml` para configurar recursos.
- **Cobertura de Teste**: Foque em garantir alta cobertura de testes, incluindo casos de erro.

Exemplo de Estrutura para o README.md

````
markdown
Copiar código
# Desafio Backend - Node.JS + AWS

## **Como Rodar o Projeto**
1. Instale as dependências:
   ```bash
   npm install
````

2. Inicie o DynamoDB local:

   ```
   bash
   Copiar código
   ```
   ```

```
docker run -p 8000:8000 amazon/dynamodb-local
```

3. Configure as variáveis de ambiente (veja `.env.example`).
4. Execute o projeto:

```
bash
Copiar código
npm start
```

Testes

- Execute os testes com:

```
bash
Copiar código
npm test
```

Recursos Utilizados

- **AWS Lambda**: Processamento serverless.
- **API Gateway**: Interface de acesso.
- **DynamoDB**: Armazenamento de dados.

Diferenciais Implementados

- ☑ Template CloudFormation.
- ☑ Cobertura de testes.

```
yaml
Copiar código

---

## **Próximos Passos**
Se precisar de ajuda para começar o código ou com alguma parte
específica, avise! Posso ajudar com exemplos de classes, configuração
de Lambda ou testes. ☺
```
4o