

# AWS Serverless Cloud Native Java RESTful API

Disciplina: Serverless Architecture

Turma 36SCJ  
2020

Aruna Fernanda Martins	– RM 338577
Ayrton Henrique Gomes Silva	– RM 337089
Carlos Eduardo Roque da Silva	– RM 338866
Sara Regina Pires	– RM 338142
Willian Yoshiaki Kazahaya	– RM 338950

## SUMÁRIO

1. INTRODUÇÃO E APRESENTAÇÃO .....	3
2. LINGUAGEM E CÓDIGO-FONTE.....	3
3. ESTRUTURA BÁSICA DA APLICAÇÃO .....	4
4. BANCO DE DADOS, ARQUIVO DE CONFIGURAÇÃO E DEPENDÊNCIAS DO PROJETO. ....	5
5. BIBLIOTECAS DE DEPENDÊNCIA .....	7
6. MICROSERVIÇO DA APLICAÇÃO .....	7
7. EXECUÇÃO DA SOLUÇÃO LOCAL, AWS CONSOLE, PARA VISUALIZAÇÃO E TESTES DA API .....	8
8. IMPLANTANDO A SOLUÇÃO NO AWS CONSOLE .....	11
9. REFERÊNCIAS .....	14

# 1. Introdução e apresentação

O Trabalho final da disciplina Serverless Achitecture de tem como objetivo desenvolver uma solução no AWS Serverless Envinronment seguindo práticas baseados nos princípios de uma arquitetura REST. Neste trabalho apresentaremos as ferramentas da Amazon Services AWS SAM<sup>1</sup>, Amazon API Gateway<sup>2</sup>, AWS Lambda<sup>3</sup> e Amazon DynamoDB. Ele também usa a estrutura ORM do DynamoDBMapper para mapear itens de uma viagem em uma tabela do DynamoDB para uma API RESTful para gerenciar as viagens cadastradas na base.

## 2. Linguagem e Código-Fonte

### Linguagem e código-fonte

A solução foi implementada utilizando a linguagem Java utilizando as bibliotecas da Amazon(AWS) Servless para exportar os endpoints configurados no arquivo Template.yaml



**GitHub** <https://github.com/AyrtonHenrique/fiap-aws-serverless>

---

<sup>1</sup> O Modelo de aplicativo sem servidor da AWS (AWS SAM) é uma estrutura de fonte aberta que pode usar para construir aplicações sem servidor na AWS

<sup>2</sup> O Amazon API Gateway é um serviço gerenciado que permite que desenvolvedores criem, publiquem, mantenham, monitorem e protejam APIs em qualquer escala com facilidade.

<sup>3</sup> O AWS Lambda é um serviço de computação que permite que você execute o código sem provisionar ou gerenciar servidores

### 3. Estrutura Básica da Aplicação

#### Desenho da Solução

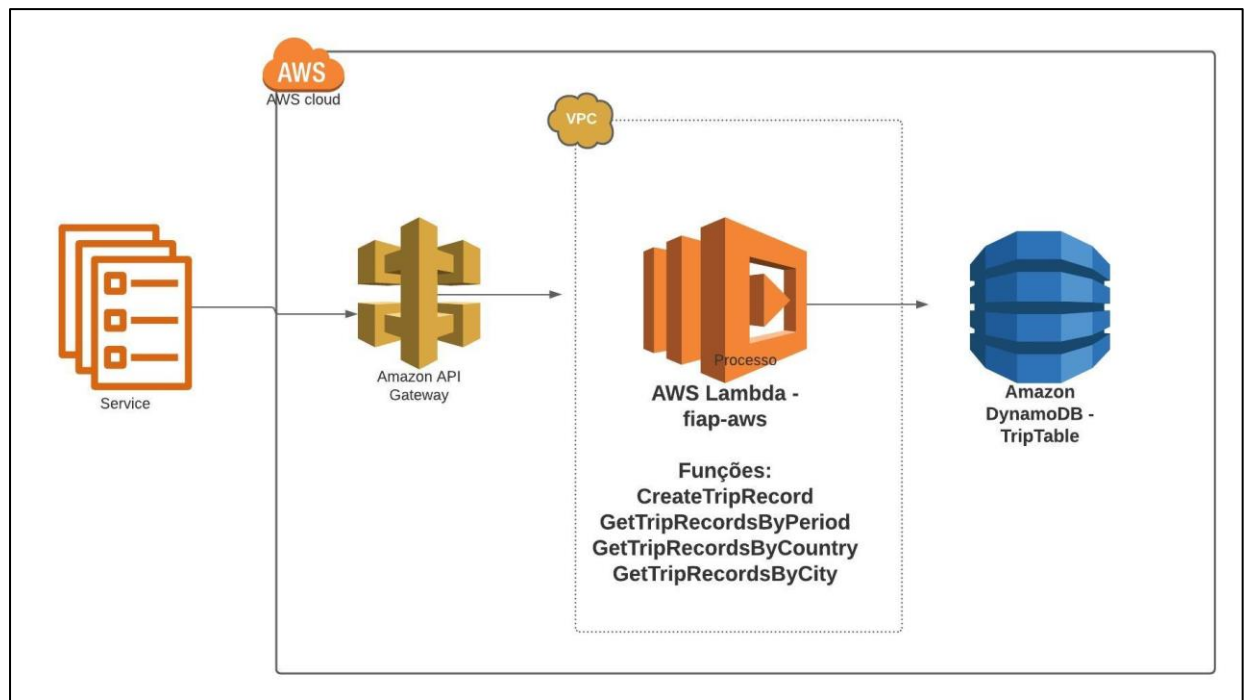


Figura 1 - Desenho da Arquitetura da Solução

#### Entidade

Foi criada apenas uma tabela Trip(Viagem) para exemplificar a solução exposta na AWS API.Gateway

Trip – Representada pela tabela: TripTable.

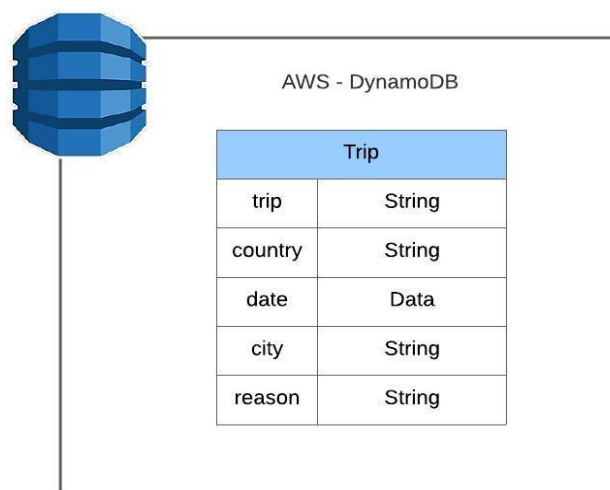


Figura 2 - Tabela NoSQL - Criada no DynamoDB

## 4. Banco de dados, arquivo de configuração e dependências do projeto.

### Banco de Dados NoSQL (Não Relacional) Amazon DynamoDB

A solução utiliza o banco de dados Dynamond DB. Segundo a documentação da AWS a definição de banco de dados NoSQL(Não relacional) são para dados específicos, ou seja, não relacionais onde temos esquemas flexíveis para criar aplicativos mais modernos. O Amazon DynamoDB é o banco de dados NoSQL de escalabilidade infinita da AWS, segundo a documentação da AWS em <<https://aws.amazon.com/dynamodb/>>. A biblioteca de conexão utilizada no projeto está referenciada no arquivo e o acesso ao banco é feito através da biblioteca <**aws-java-sdk-dynamodb**>.

O DynamoDB utiliza a estrutura de arquivo no formato JSON, para cadastro de dados.

```

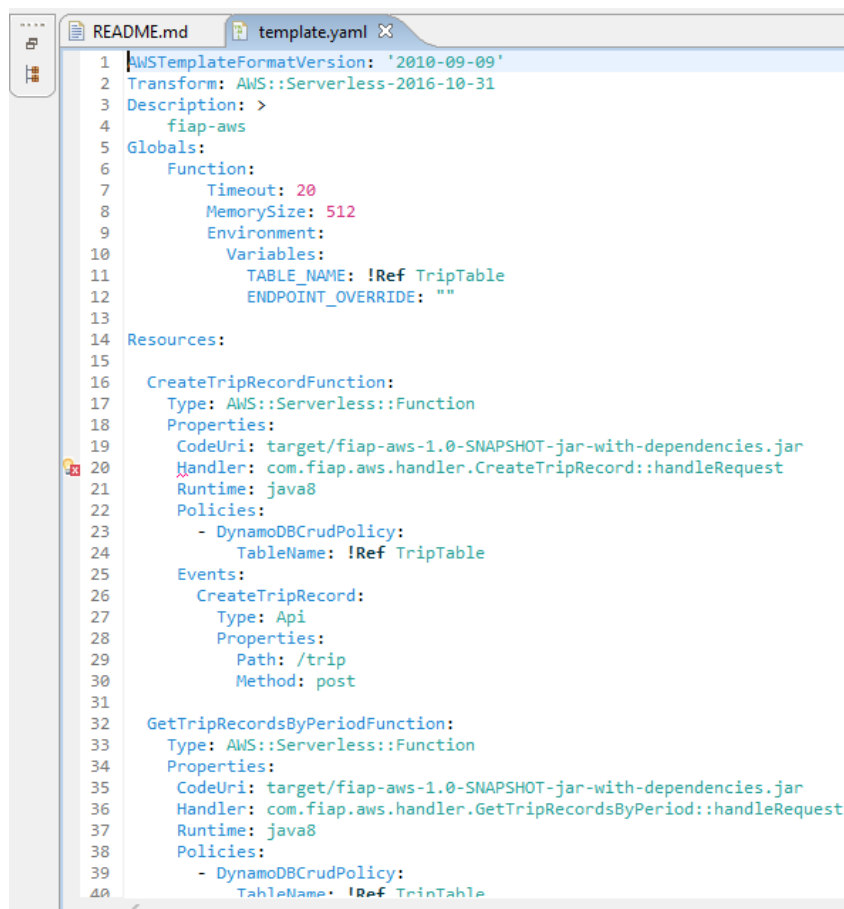
1  aws dynamodb put-item \
2      --table-name Music \
3      --item '{ \
4          "Artist": {"S": "Acme Band"}, \
5          "SongTitle": {"S": "Happy Day"}, \
6          "AlbumTitle": {"S": "Songs About Life"} }' \
7      --return-consumed-capacity TOTAL

```

Figura 3 - Exemplo cadastro DynamoDB - JSON

## Arquivo de Configuração

A aplicação possui uma configuração através de um arquivo de propriedades (*Template.yml*) onde serão descritas as funcionalidades específicas para cada endpoint que será implantado na AWS Management Console.



```

1 AWSTemplateFormatVersion: '2010-09-09'
2 Transform: AWS::Serverless-2016-10-31
3 Description: >
4   fiap-aws
5 Globals:
6   Function:
7     Timeout: 20
8     MemorySize: 512
9     Environment:
10      Variables:
11        TABLE_NAME: !Ref TripTable
12        ENDPOINT_OVERRIDE: ""
13
14 Resources:
15
16   CreateTripRecordFunction:
17     Type: AWS::Serverless::Function
18     Properties:
19       CodeUri: target/fiap-aws-1.0-SNAPSHOT-jar-with-dependencies.jar
20       Handler: com.fiap.aws.handler.CreateTripRecord::handleRequest
21       Runtime: java8
22       Policies:
23         - DynamoDBCrudPolicy:
24             TableName: !Ref TripTable
25     Events:
26       CreateTripRecord:
27         Type: Api
28         Properties:
29           Path: /trip
30           Method: post
31
32   GetTripRecordsByPeriodFunction:
33     Type: AWS::Serverless::Function
34     Properties:
35       CodeUri: target/fiap-aws-1.0-SNAPSHOT-jar-with-dependencies.jar
36       Handler: com.fiap.aws.handler.GetTripRecordsByPeriod::handleRequest
37       Runtime: java8
38       Policies:
39         - DynamoDBCrudPolicy:
40             TableName: !Ref TripTable
  
```

Figura 4 - Exemplo Template arquivo YAML

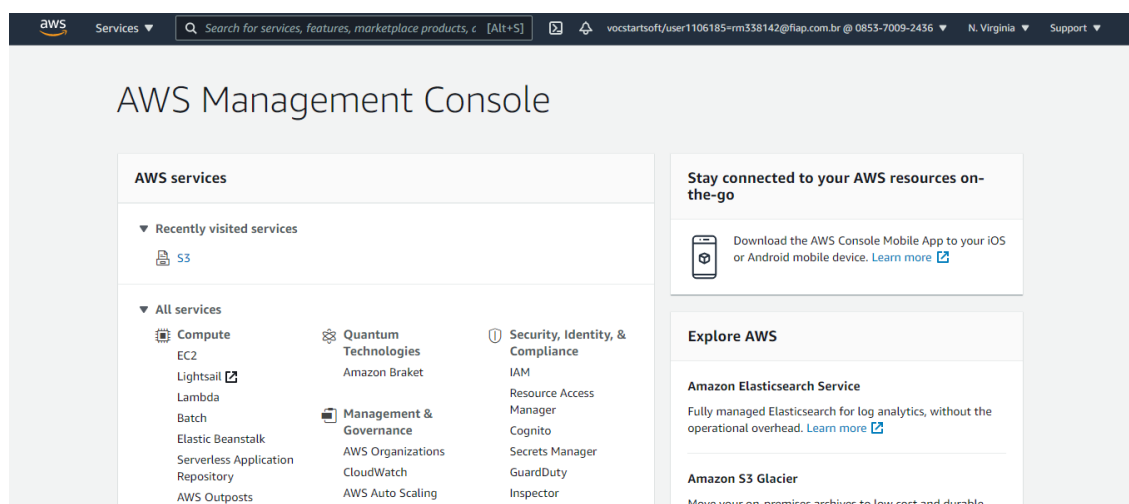


Figura 5- Console de Gerenciamento AWS

## 5. Bibliotecas de dependência

Foram utilizadas nesta solução as seguintes bibliotecas:

**aws-lambda-java e aws-java-sdk-core** – define as interfaces do método do manipulador e o objeto de contexto que o tempo de execução transmite ao manipulador.

**aws-lambda-java-events** – tipos de entrada para eventos de serviços que invocam funções do Lambda.

**aws-java-sdk-dynamodb** – Aws Dynamodb para acesso ao banco de dados.

## 6. Microserviço da Aplicação

### Microserviço: **fiap-aws**

Este microserviço inicia uma aplicação AWS Serverless Function que expõe as funcionalidades codificadas na AWS Lambda.

- CreateTripRecord : Lambda responsável para cadastro da Viagem(Trip);
- GetTripRecordsByPeriod – Lambda responsável consultar as Viagens(Trip) por um determinado período.
- GetTripRecordsByCountry – Lambda responsável por consultar as viagens por um país.
- GetTripRecordsByCity - Lambda responsável por consultar as viagens por um país por cidade.

## 7. Execução da solução local, AWS Console, para visualização e testes da API

Todos os microsserviços que expõe APIs descritos acima tiveram as mesmas documentadas e testadas através do AWS Console

Abaixo estão todas as interfaces das API:

### 1) Pré-requisitos

Para que a solução funcione no ambiente local, devemos instalar AWS CLI e o AWS SAM para executar os comandos no terminal do Windows(CMD).

Executar o comando no command windows: **aws configure**

Pegar as credenciais na conta de estudante na AWS CLI

Colocar os dados AWS Access Key, AWS Secret Access Key nos campos: , conforme imagem abaixo.

```
D:\git>aws configure
AWS Access Key ID [*****AI2X]: ASIARHYDUHOKJLKTVRQ5
AWS Secret Access Key [*****5fNT]: 3UL96RPISzOEXTUD4F8Yy25E7qdoF0J0qxv1Ufi3
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

### 2) Criar a tabela no DynamoDB

```
aws dynamodb create-table
--table-name trip
--attribute-definitions AttributeName=trip,AttributeType=S
                        AttributeName=country,AttributeType=S
                        AttributeName=city,AttributeType=S
                        AttributeName=date,AttributeType=S
                        AttributeName=reason,AttributeType=S
--key-schema AttributeName=trip,KeyType=HASH AttributeName=date,KeyType=RANGE
--local-secondary-indexes "IndexName=countryIndex,KeySchema=[{AttributeName=trip,KeyType=HASH},{AttributeName=date,KeyType=RANGE}],Projection={ProjectionType=ALL}"
                        "IndexName=cityIndex,KeySchema=[{AttributeName=trip,KeyType=HASH},{AttributeName=date,KeyType=RANGE}],Projection={ProjectionType=ALL}"
                        "IndexName=reasonIndex,KeySchema=[{AttributeName=trip,KeyType=HASH},{AttributeName=date,KeyType=RANGE}],Projection={ProjectionType=ALL}"
--billing-mode PAY_PER_REQUEST
--endpoint-url http://localhost:8000
```

### 3) Baixar o código fonte no repositório

<https://github.com/AyrtonHenrique/fiap-aws-serverless/tree/main/fiap-aws>



#### 4) Simulação de Uso dos Endpoints

##### a) Criar um novo registro de Trip(Viagem) - Post

**http://localhost:0000/trip**

Body da Requisição:

```
{
  "trip": "acampa",
  "country": "Canada",
  "date": "2020-12-26 22:30:53.260",
  "city": "toronto",
  "reason": "ferias"
},
```

Retorno da Resposta:

```
{
  "trip": "acampa",
  "country": "Canada",
  "date": "2020-12-26 22:30:53.260",
  "city": "toronto",
  "reason": "ferias"
},
```

##### b) Consultando uma viagem por período - GET

**http://localhost:0000/trip?start=2020-12-01&end=2021-01-31**

```
[
  {
    "trip": "acampa",
    "country": "Canada",
    "date": "2020-12-26 22:30:53.260",
    "city": "toronto",
    "reason": "ferias"
  },
  {
    "trip": "acampa",
    "country": "EUA",
    "date": "2021-01-01 22:30:53.260",
    "city": "New york",
    "reason": "trabalho"
  }
]
```

c) Consultando os dados da viagem por país - **GET**

**http://localhost:0000/trip/{country}**

Retorno da Resposta:

```
[
  {
    "trip": "acampa",
    "country": "Canada",
    "date": "2020-12-26 22:30:53.260",
    "city": "toronto",
    "reason": "ferias"
  }
]
```

d) Consultando os dados da viagem por país e cidade - **GET**

**http://localhost:0000/trip/{country}?city=<City>**

Retorno da Resposta:

```
[
  {
    "trip": "acampa",
    "country": "Canada",
    "date": "2020-12-26 22:30:53.260",
    "city": "toronto",
    "reason": "ferias"
  }
]
```

## 8. Implantando a Solução no AWS Console

### 1) Efetuando deploy da solução no AWS – Execução de Comando.

```
Administrador: Prompt de Comando
D:\git\fiap-aws-serverless\fiap-aws>sam deploy --template-file packaged.yaml --stack-name sara-test --capabilities CAPABILITY_IAM

Deploying with following values
-----
Stack name           : sara-test
Region               : None
Confirm changeset    : False
Deployment s3 bucket  : None
Capabilities          : ["CAPABILITY_IAM"]
Parameter overrides   : {}
Signing Profiles      : {}

Initiating deployment
-----
CreateTripRecordFunction may not have authorization defined.
GetTripRecordsByPeriodFunction may not have authorization defined.
GetTripRecordsByCountryFunction may not have authorization defined.

Waiting for changeset to be created..

CloudFormation stack changeset
-----
Operation      LogicalResourceId      ResourceType      Replacement
-----
+ Add          CreateTripRecordFunctionCreateTripRecordPermissionProd  AWS::Lambda::Permission      N/A
+ Add          CreateTripRecordFunctionRole      AWS::IAM::Role      N/A
+ Add          CreateTripRecordFunction      AWS::Lambda::Function      N/A
+ Add          GetTripRecordsByCountryFunctionGetTripRecordsByCountryPermissionProd  AWS::Lambda::Permission      N/A
+ Add          GetTripRecordsByCountryFunctionRole      AWS::IAM::Role      N/A
+ Add          GetTripRecordsByCountryFunction      AWS::Lambda::Function      N/A
+ Add          GetTripRecordsByPeriodFunctionGetTripRecordsByPeriodPermissionProd  AWS::Lambda::Permission      N/A
+ Add          GetTripRecordsByPeriodFunctionRole      AWS::IAM::Role      N/A
+ Add          GetTripRecordsByPeriodFunction      AWS::Lambda::Function      N/A
+ Add          ServerlessRestApiDeployment84a35344a8  AWS::ApiGateway::Deployment      N/A
+ Add          ServerlessRestApiProdStage      AWS::ApiGateway::Stage      N/A
+ Add          ServerlessRestApi      AWS::ApiGateway::RestApi      N/A
+ Add          TripTable      AWS::DynamoDB::Table      N/A
-----

Changeset created successfully. arn:aws:cloudformation:us-east-1:085370092436:changeSet/samcli-deploy1609550486/f077c63a-efe3-48d8-9fa9-ee2a4383926e
```

Figura 6 - Implantando a solução no AWS Console

## 2) Testando as funcionalidades da Lambida – API :

### - Criar um novo registro de Trip:

Usage Plans	No <a href="#">stage variables</a> exist for this method.
API Keys	Client Certificate
Client Certificates	No client certificates have been generated.
Settings	Request Body

```

1 {
2   "trip": "acampa",
3   "country": "EUA",
4   "date": "2021-01-01 22:30:53.260",
5   "city": "New york",
6   "reason": "trabalho"
7 }

```

[Test](#)

Figura 7 - Testando Função Lambda no API Gateway

### - Consultar viagem por um período:

<ul style="list-style-type: none"> <li>/</li> <li> <ul style="list-style-type: none"> <li>/trip           <ul style="list-style-type: none"> <li>GET</li> <li>POST</li> <li> <ul style="list-style-type: none"> <li>/country               <ul style="list-style-type: none"> <li>GET</li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul>	<p>Make a test call to your method with the provided input</p> <p><b>Path</b></p> <p>No path parameters exist for this resource. You can define path parameters by using the syntax <b>{myPathParam}</b> in a resource path.</p> <p><b>Query Strings</b></p> <p><b>{trip}</b></p> <p>start=2020-12-01&amp;end=2021-01-31</p> <p><b>Headers</b></p> <p><b>{trip}</b></p> <p>Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.</p> <p><b>Stage Variables</b></p> <p>No <a href="#">stage variables</a> exist for this method.</p>	<p>Request: /trip?start=2020-12-01&amp;end=2021-01-31</p> <p>Status: 200</p> <p>Latency: 6372 ms</p> <p><b>Response Body</b></p> <pre> [   {     "trip": "acampa",     "country": "Canada",     "date": "2020-12-26 22:30:53.260",     "city": "toronto",     "reason": "ferias"   },   {     "trip": "acampa",     "country": "EUA",     "date": "2021-01-01 22:30:53.260",     "city": "New york",     "reason": "trabalho"   } ] </pre>
--	--	--

Figura 8 - Testando Função Lambda no API Gateway

## - Consultar viagem para um país:

Resources: Actions ▾

Method Execution /trip/{country} - GET - Method Test

Make a test call to your method with the provided input

Path: {country} (EUA)

Request: /trip/EUA  
Status: 200  
Latency: 6283 ms

Response Body:

```
[
  {
    "trip": "acampa",
    "country": "EUA",
    "date": "2021-01-01 22:30:53.260",
    "city": "New york",
    "reason": "trabalho"
  }
]
```

Query Strings: {country} (param1=value1&param2=value2)

Headers: {country} (Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.)

Response Headers: {"X-Amzn-Trace-Id": "Root=1-5fef5e4-bab4178accfacd9eb95f551a;Sampled=0"}

Stage Variables:

Figura 9 - Testando Função Lambda no API Gateway

## - Consultar viagem para um país e cidade:

Resources: Actions ▾

Method Execution /trip/{country} - GET - Method Test

Make a test call to your method with the provided input

Path: {country} (Canada)

Request: /trip/Canada?toronto  
Status: 200  
Latency: 35 ms

Response Body:

```
[
  {
    "trip": "acampa",
    "country": "Canada",
    "date": "2020-12-26 22:30:53.260",
    "city": "toronto",
    "reason": "ferias"
  }
]
```

Query Strings: {country} (toronto)

Headers: {country} (Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.)

Response Headers: {"X-Amzn-Trace-Id": "Root=1-5fef70a-b6e1fa446c874c7a3340ff4c4;Sampled=0"}

Figura 10 - Testando Função Lambda no API Gateway

## 9. Referências

O que é API REST?. Artigo:, Disponível em:

<<https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>>. Acesso em: 26 de dez. de 2020.

O que é AWS SAM, Guia do Desenvolvedor, Disponível em: <

[https://docs.aws.amazon.com/pt\\_br/serverless-application-model/latest/developerguide/what-is-sam.html](https://docs.aws.amazon.com/pt_br/serverless-application-model/latest/developerguide/what-is-sam.html)>

Amazon API Gateway, Disponível em: < <https://aws.amazon.com/pt/api-gateway/>>

acesso em 26 de dez. de 2020.

Amazon DynamoDB, Disponível em: <<https://docs.aws.amazon.com/index.html>>,

acesso em: 26 de Dez. de 2020

FIAP  
2020