

COMPUTAÇÃO NA NUVEM

Segundo trabalho prático



Curso: Eng.^a Informática, Redes e Telecomunicações

Autores:

a21 801 188 – Hércules Semedo

a21 900 252 – Armando Carruagem

2020/2021

Índice

Tópicos	Páginas
Objectivos	3
Caso de estudo para trabalho	4
Introdução	5
Enquadramento Teórico	6
Serverless e hosting tradicional	7
Criação da Amazon DynamoDB	8
Criação das funções Amazon Lambda	10
Criação da API Gateway (HTTP)	12
Testes de implementação	22
Customização	27
Testes e demonstrações	29

Objectivos

Este último trabalho da disciplina tem por objectivo a elaboração de uma solução Web sem servidores – Serverless – tirando partido dos recursos disponibilizados pela tecnologia cloud.

Essencialmente, para a resolução dos exercícios propostos, somente são necessários os seguintes serviços:

- DynamoDB;
- AWS Lambda;
- AWS API Gateway;
- IAM.

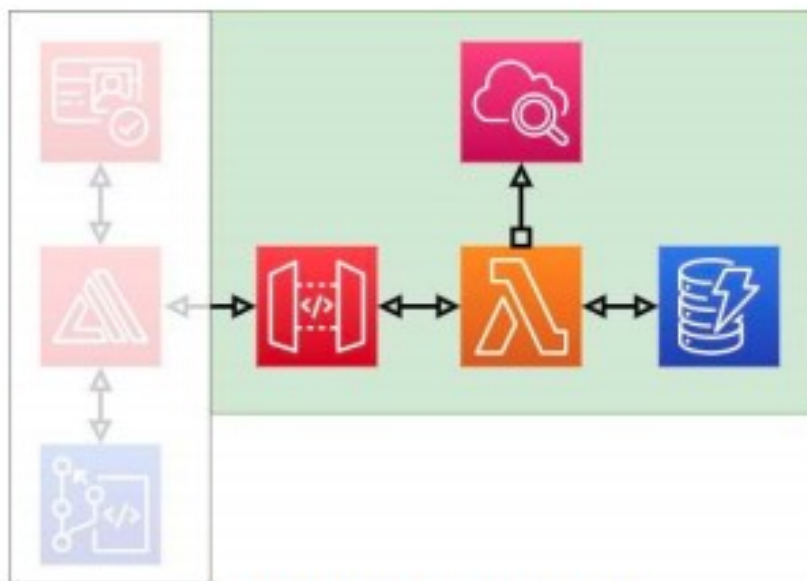


FIGURA 1 – Modelo *serverless* AWS

Caso de estudo para trabalho

O trabalho consiste na elaboração de pauta de avaliação para unidades curriculares de ensino superior, com as condições que abaixo se detalham:

1. Estrutura de dados essenciais

- a. Lista de alunos:
 - i. Nome do aluno
 - ii. Número de aluno
 - iii. Turma
- b. Resultados de avaliação a uma disciplina
 - i. Código da disciplina
 - ii. Número de aluno
 - iii. Resultado na disciplina

2. Dados obrigatórios para o trabalho

- a. Alunos do grupo
- b. Avaliações de cada aluno em Sistemas de Informação na Nuvem

3. Resultados a produzir

- a. Alunos de uma turma
- b. Avaliações de um aluno à disciplina existente
- c. Pauta completa de uma turma

O trabalho base pode ser acrescentado com elementos adicionais que complementem a avaliação. Sugerem-se os elementos abaixo, podendo cada grupo acrescentar outros que considere relevantes.

4. Dados adicionais

- a. Disciplinas:
 - i. Código da disciplina
 - ii. Nome da disciplina
 - iii. Ano curricular e semestre (pode ser no mesmo atributo)
 - iv. ECTS

5. Resultados adicionais

- a. Lista de disciplinas
- b. Avaliações de um aluno a todas as disciplinas
- c. Pauta completa, apresentando o nome da disciplina em vez do código

Introdução

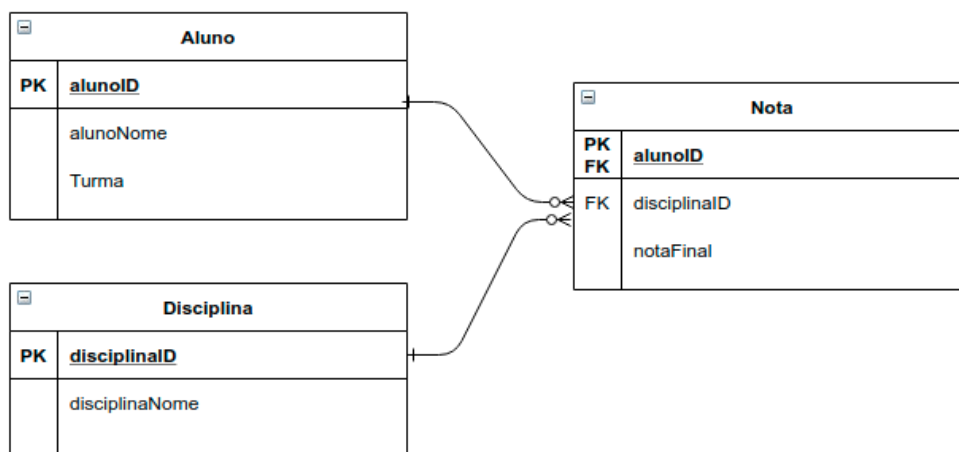
Atualmente o serverless tem ganhado bastante aderência em cloud computing, sendo assim, é importante para estudantes na área de informática, e não só, o domínio desta matéria. Atendendo a isso iremos realizar neste segundo trabalho prático, um relatório duma solução web em serverless, utilizando recursos disponibilizado pela tecnologia cloud. No severless os desenvolvedores não gerem os servidores onde os aplicativos estão a ser executados, desta forma os aplicativos podem ganhar escala automaticamente.

Para solucionar os problemas propostos utilizaremos como instrumento de trabalho a plataforma Amazon, na qual utilizaremos especificamente os seguintes serviços; DynamoDB, AWS Lambda, AWS API Gateway, e o IAM.

O desenvolvimento do trabalho também foi baseado em um tutorial fornecido pela aws, cujo link se encontra indicado abaixo:

https://docs.aws.amazon.com/pt_br/apigateway/latest/developerguide/http-api-dynamo-db.html

Também para atingir os requisitos a serem cumpridos, foi considerado o seguinte diagrama relacional no DynamoDB:



Enquadramento teórico

Como foi dito na introdução do relatório, utilizemos os seguintes serviços para a realização deste trabalho:

DynamoDB

Segundo AWS o DynamoDB é um serviço de banco de dados NoSQL totalmente gerenciado, e fornece um rápido desempenho, também previsível com escalabilidade integrada. Este serviço permite a transferência de encargos administrativos de operação e escalabilidade de um banco de dados distribuído. O serviço oferece também criptografia em repouso, o qual elimina a carga e a complexidade operacionais envolvidas na proteção de dados confidenciais.

É possível também, no DynamoDB, criar tabelas de banco de dados que armazenam e recuperam qualquer quantidade de dados e atendem a todos os níveis de tráfego solicitados. Podemos aumentar ou diminuir a capacidade de throughput da tabela sem tempo de inatividade ou degradação do desempenho. Há também o recurso de backup sob demanda. Ele permite que a criação de backups completos das tabelas para retenção e arquivo de longo prazo de modo a atender às necessidades de conformidade regulamentar.

AWS Lambda

O lambda permite executar código sem fornecer ou gerenciar servidores. O código é executado numa infraestrutura de computação de alta disponibilidade juntamente com toda a administração dos recursos computacionais, inclusive a manutenção do servidor e do sistema operativo. Também é possível executar o código em ambiente virtual, para praticamente qualquer tipo de aplicação ou serviço de back-end.

AWS API Gateway

É um serviço que pertence a AWS que permite a criação, publicação, manutenção, monitorização, e proteção de APIs REST e WebSocket em qualquer escala. Os desenvolvedores podem criar APIs que acessem serviços da AWS ou outros serviços da Web, bem como dados armazenados na nuvem. É possível criar APIs para uso em aplicações pessoais, ou disponibiliza-las para outros desenvolvedores de aplicações.

IAM

O IAM permite o controle em segurança de acesso aos recursos da AWS de forma segura. Permite identificar quem é autenticado, e autorizado para utilização dos recursos.

Serverless e hosting tradicional

No Serverless os criadores das aplicações concentram-se mais em desenvolver e melhorar as suas aplicações, uma vez que não têm de fazer o gerenciamento dos servidores, o que não acontece no hosting tradicional.

O hosting tradicional é uma forma de oferecer recursos em um ambiente que hospeda aplicações, soluções de TI ou ativos. consiste em hospedar ou manter aplicações, sistemas ou até mesmo estruturas inteiras em um data center terceiro, funciona por meio do modelo de contratação de infraestrutura de TI (como hardware, licenças, serviços gerenciados e especializados) junto a um provedor de Data Center.

Em empresas onde se exige uma elevada taxa de disponibilidade do ambiente computacional e que precisam contar com uma infraestrutura escalável, não é recomendado a utilização do hosting tradicional. Já em negócios cujo o fluxo de tráfego é previsível, pode-se optar pelo uso do fazer o hosting tradicional e seus ativos de TI, desde que seja de forma compartilhada ou dedicada. Isso deve-se ao facto de que muitas vezes no modelo compartilhado, o desempenho pode ficar comprometido por conta de picos de tráfego de dados

De lembrar que este modelo não tem elasticidade, e surgir a necessidade de utilizar um volume maior de recursos, a empresa deverá comprar espaço adicional no servidor e podem acabar por pagar um serviço sem estar a utilizá-lo.

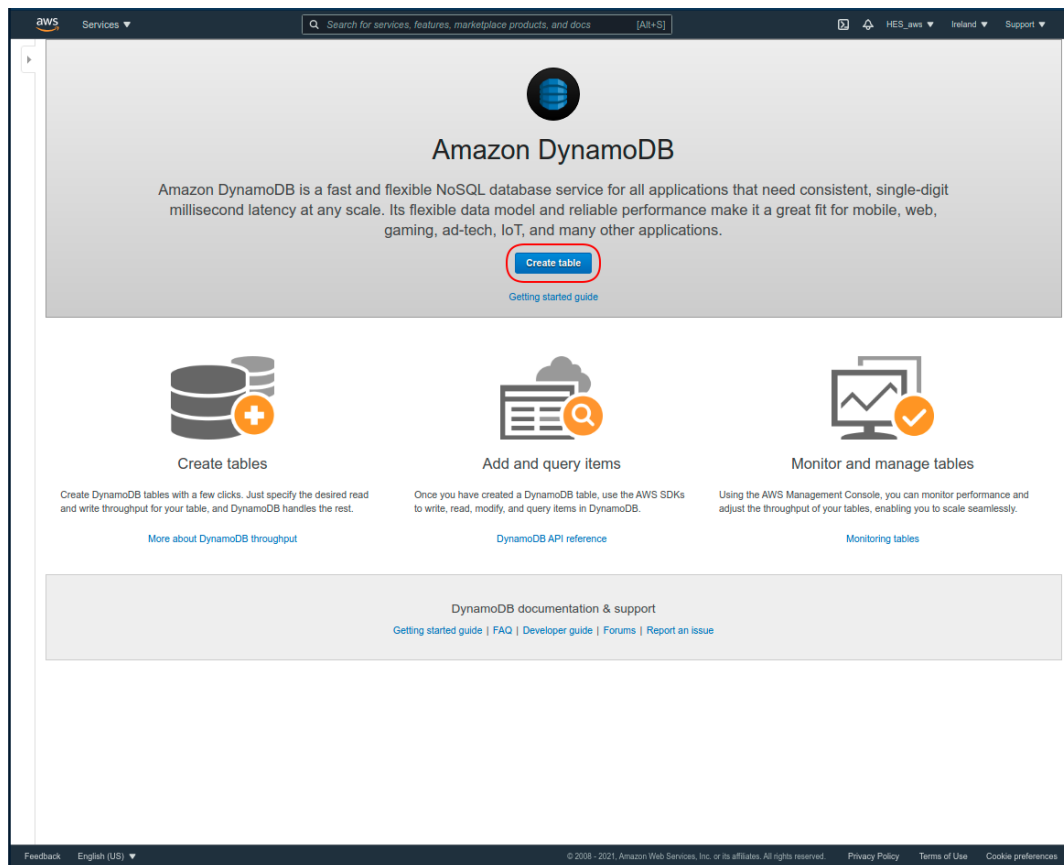
com a utilização do serverless, a TI consegue realizar testes e encontrar soluções para possíveis problemas de infraestrutura com eficiência, automatizar tarefas repetitivas para focar no crescimento da empresa, otimizar a experiência dos usuários finais, tudo isso com segurança e redundância. Também a capacidade de aumentar a capacidade de processamento e armazenamento de dados conforme a necessidade do utilizador, de forma simples e rápida. Assim, é possível garantir que qualquer infraestrutura de TI esteja preparada para se adaptar a qualquer cenário, sem necessidade de fazer grandes investimentos de uma vez só, garantindo escalabilidade operacional.

Outra vantagem do serverless em relação ao hosting tradicional é o modelo de pagamento dos serviços, visto que o serverless permite uma alocação mais saudável de investimento, o que já não acontece no hosting tradicional, em que o cliente deve pagar por uma quantidade limitada de recursos.

Criação da amazon DynamoDB

Os primeiros passos do trabalho foram de teste. Deste modo, inicialmente foi criado e foi utilizado apenas uma tabela da Amazon DynamoDB, que é um serviço de base de dados não relacional (noSQL database) da Amazon.

A criação foi feita acedendo ao DynamoDB na parte Database na lista de serviços da amazon e selecionando a opção *Create table*, a opção apresentada na imagem abaixo:



Na criação foi especificado o nome da tabela, e o nome escolhido foi “Alunos”. Logo na criação também é necessário especificar uma chave primária à tabela, que no caso da tabela Alunos é o número de cada aluno (alunoID).

Depois de especificar a tabela e a chave primária, foi selecionado a opção *create*, parecido ao da página anterior, como indica a imagem indicada abaixo:

aws

Services

Search for services, features, marketplace products, and docs

[Alt+S]

HES_aws

Ireland

Support

Create DynamoDB table

Tutorial

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*

Alunos

Primary key*

Partition key

alunoID

(String)

☐ Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☒ Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type.

You do not have the required role to enable Auto Scaling by default.
Please refer to [documentation](#).

+ Add tags

NEW

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Cancel

Create

Feedback

English (US)

© 2018 - 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

Cookie preferences

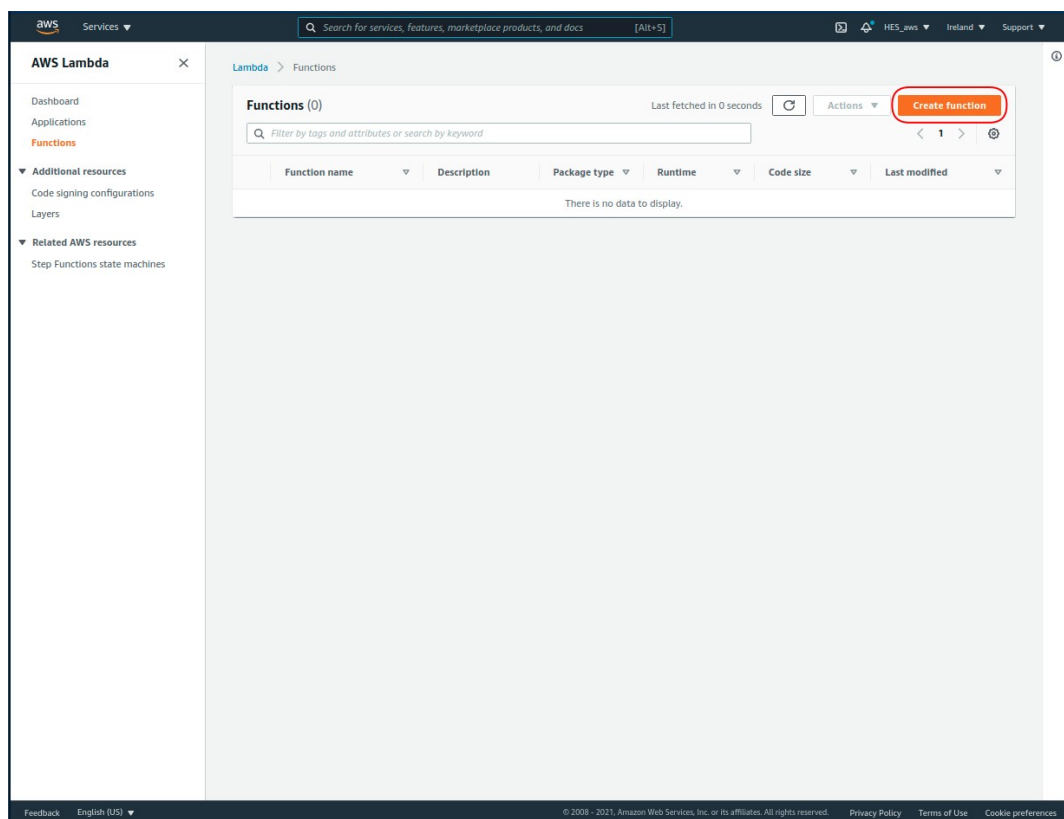
Criação de funções com amazon Lambda

De seguida, foi criado uma função Lambda onde foram implementadas as funções de CRUD. Para criar a função foi acedido a consola das funções Lambdas do Amazon, e seleccionando a opção **Create function**.

Após seleccionar a opção, foi seleccionada a opção *Author from scratch* e também foi necessário especificar o nome da função Lambda. O nome escolhido foi “**avaliacoes_CRUD**”. O Runtime escolhido foi o Node.js 14.

Na parte das permissões, foi escolhido a opção *Change default execution Role* e seleccionando a opção *Create new role from AWS policy templates*. O nome escolhido para o novo role criado foi “**avaliacoes_CRUD_role**”. Na parte Policy templates, foi escolhido o *Simple microservice permissions*, que permite à função Lambda aceder e interagir com a tabela criada no DynamoDB.

E por fim foi seleccionado a opção *Create function*. Estes passos estão resumidos nas imagens abaixo indicadas:



The screenshot shows the AWS Lambda console interface for creating a new function. The 'Author from scratch' tab is active. The 'Basic information' section includes a 'Function name' field with the value 'avaliacoes_CRUD' and a 'Runtime' dropdown set to 'Node.js 14.x'. The 'Permissions' section shows the 'Change default execution role' step, where 'Create a new role from AWS policy templates' is selected. The 'Role name' field contains 'avaliacoes_CRUD_role' and the 'Policy templates' dropdown is set to 'Simple microservice permissions'. The 'Create function' button at the bottom right is highlighted with a red circle.

Após criar a função Lambda, foi carregado o código exemplo (do tutorial) para o ficheiro `index.js`, da função. Foi necessário fazer algumas alterações no código para que funcionasse com a tabela Alunos criada no DynamoDB (em especial os parâmetros que apontavam para a tabela ou à sua chave primária). Após efetuar as edições necessárias no código, foi selecionado a opção de *Deploy* para atualizar o código.

Criação da API Gateway (HTTP)

De seguida, foi criado uma API HTTP, que iria fornecer um endpoint para a função Lambda criada.

A criação foi feita primeiro acedendo à consola API Gateway e seleccionando as opções *Create API*. Foi criado um API HTTP, e na criação foi necessário escolher um nome para API. Seguindo a lógica das etapas anteriores, foi escolhido o nome “**avaliacoes_CRUD_api**”.

De seguida foram criadas as rotas da API. Para a criação das rotas é necessário especificar o seu caminho e o método HTTP a ser usado. Repetindo este mesmo processo para cada rota, inicialmente, foram criadas quatro rotas que são as seguintes:

```
+ GET /alunos/{alunoID}
+ GET /alunos
+ PUT /alunos
+ DELETE /alunos/{alunoID}
```

De seguida foi criada uma integração da API, que basicamente ligará cada rota com a função lambda criada. Isto foi feito na consola da API, criada na parte *Integrations > Manage integrations < Create*. Na parte em que se especifica o tipo de API, foi seleccionado o tipo **Lambda function** e foi indicado o nome da função que era para ser associado às rotas (avaliacoes_CRUD).

Após criar a integração, só foi necessário anexá-la às rotas criadas, algo que foi feito seleccionando a opção *Attach integration*, para cada rota anteriormente criada.

Estes passos, para criar e configurar a API HTTP criada, estão representados nas imagens abaixo indicadas:

aws

Services

Search for services, features, marketplace products, and docs

[Alt+S]

HES_aws

Ireland

Support

API Gateway

APIs

Custom domain names

VPC links

Step 1

Create an API

Step 2

Configure routes

Step 3

Define stages

Step 4

Review and create

Create an API

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda Integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP Integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations

Info

Add integration

API name

An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

avalliacoes_CRUD_api

Cancel

Review and Create

Next

Feedback

English (US)

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

Cookie preferences

aws

Services

Search for services, features, marketplace products, and docs

[Alt+S]

HES_aws

Ireland

Support

API Gateway

APIs

Custom domain names

VPC links

API: avalliacoes_CRUD...

(f7ftisc06h)

Develop

Routes

Authorization

Integrations

CORS

Reimport

Export

Deploy

Stages

Protect

Throttling

Monitor

Metrics

Logging

Successfully created API avalliacoes_CRUD_api (f7ftisc06h)

API Gateway > Routes > Create

Create a route

Route and method

Info

Choose a method and enter a path to create a route. You can also specify one \$default route per API. The \$default route is invoked when the request to the API matches no other routes.

GET

/alunos/{alunoID}

Cancel

Create

Feedback

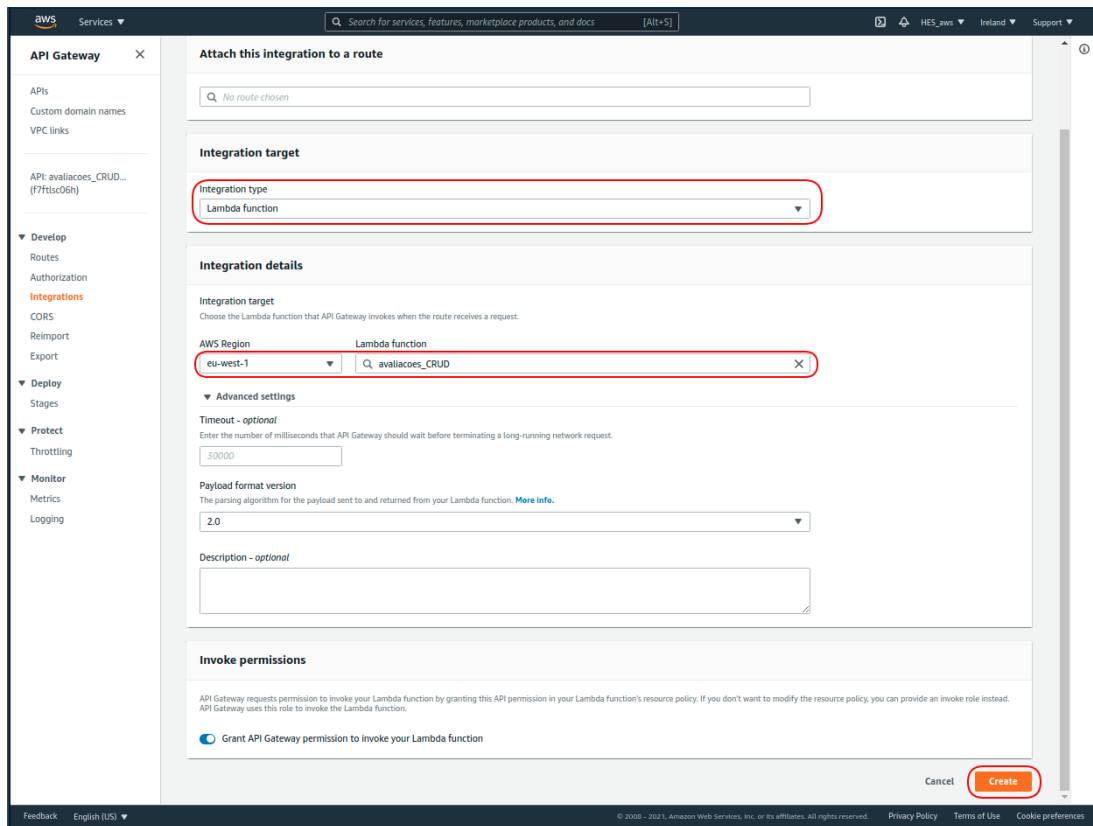
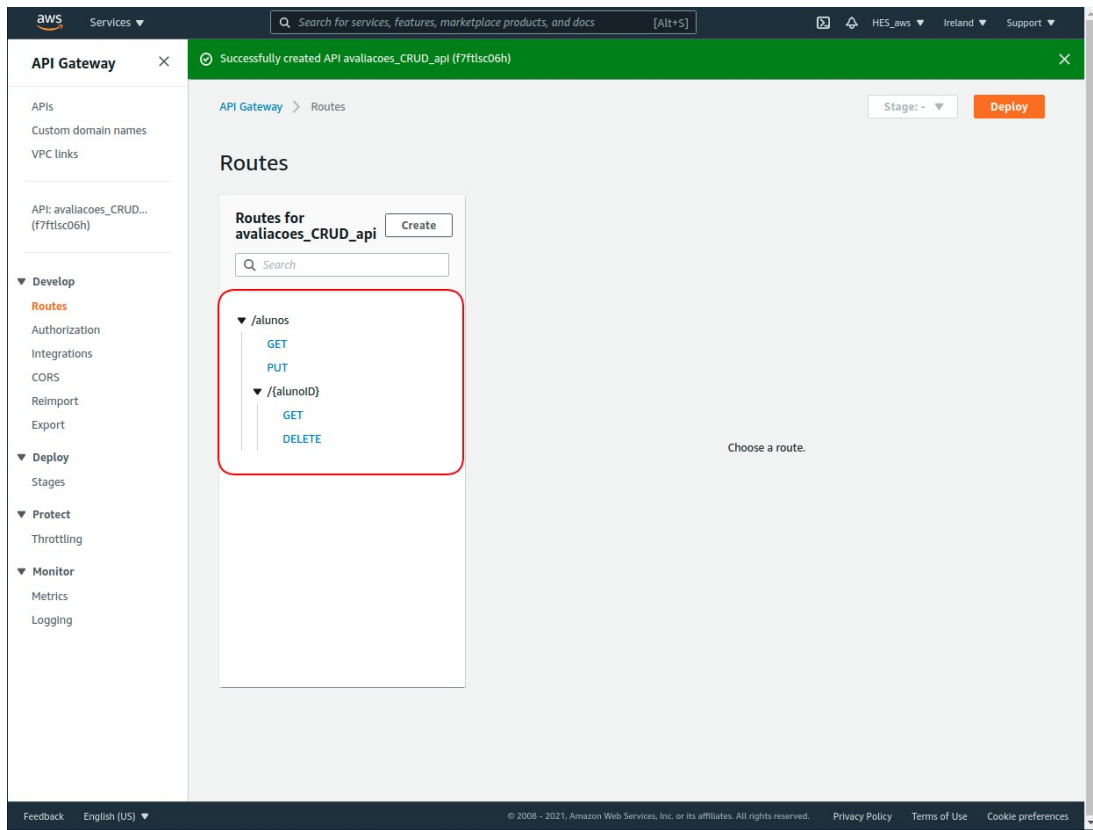
English (US)

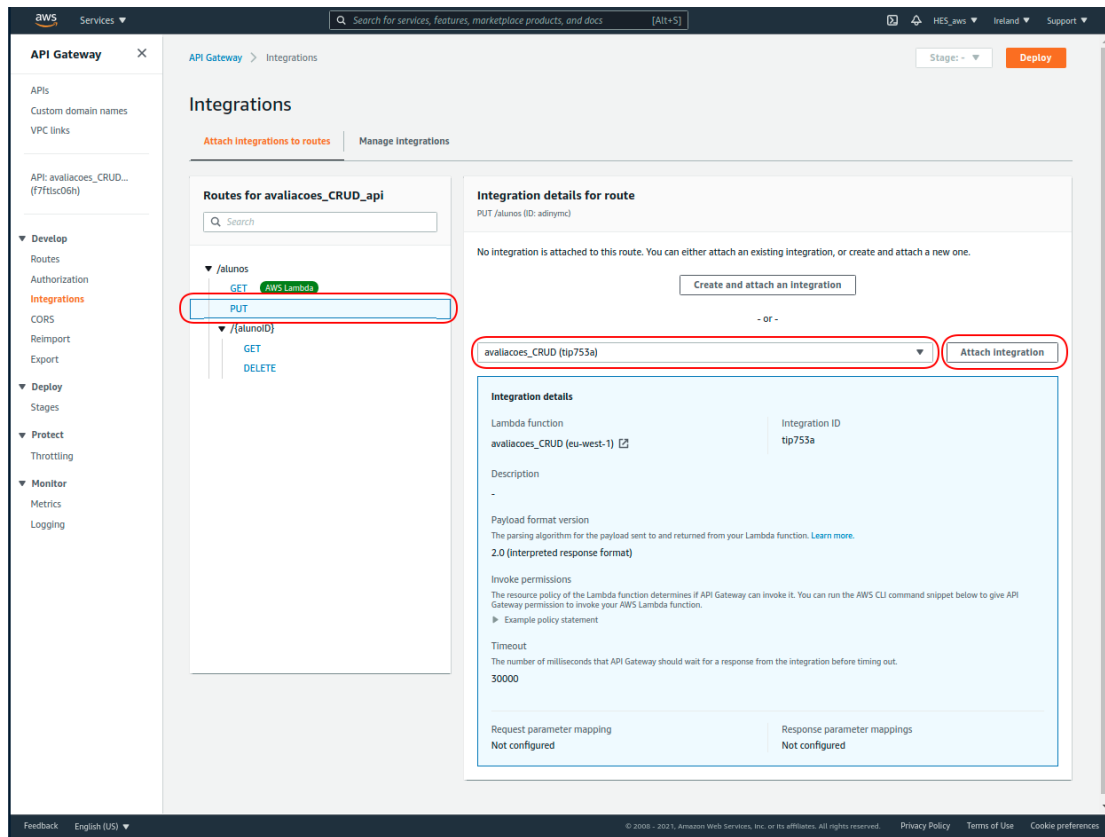
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

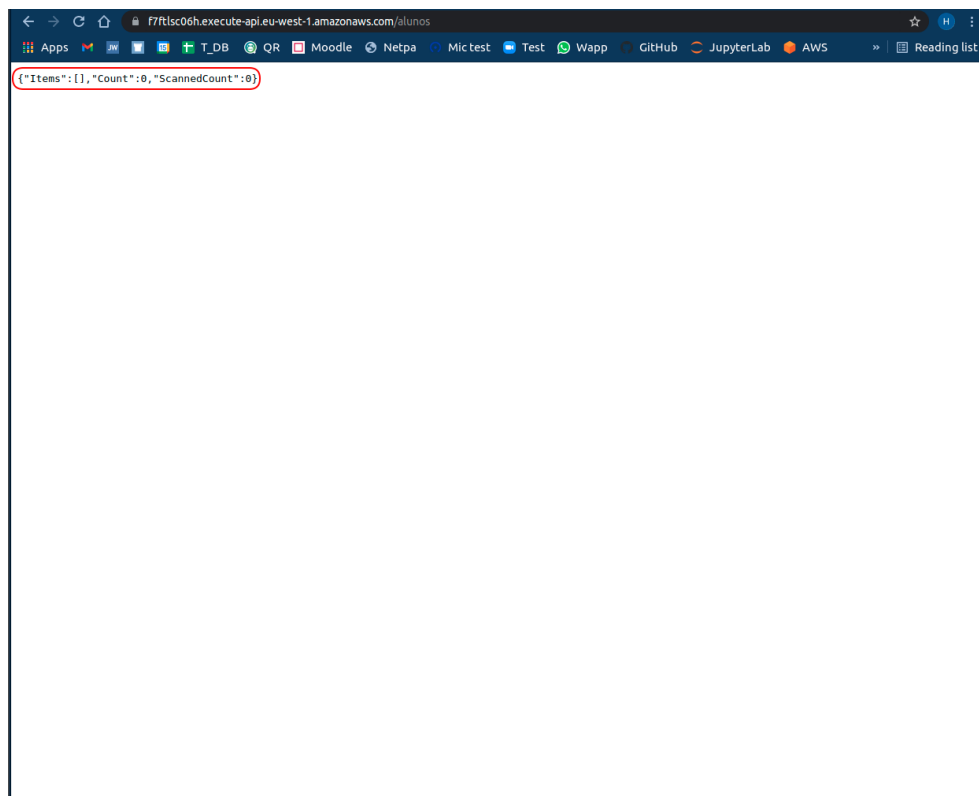
Terms of Use

Cookie preferences





Após ter associado todas as rotas à integração, foi realizado o teste da API criada. Primeiro foi simplesmente acessado ao URL da API mais a rota **/alunos** e como se pode verificar na imagem indicada abaixo, não houve nenhum erro ao aceder à rota:



Também a implementação foi testada usando os comandos CURL, como se pode verificar no exemplo apresentado abaixo:

1º) Foi adicionado um aluno à lista dos alunos (cujo alunoID = 110 022):

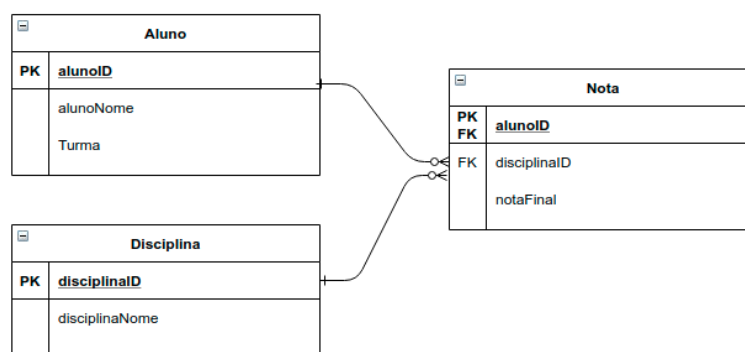
```
hes@HES-PC:~$ curl -v https://f7f7lsc06h.execute-api.eu-west-1.amazonaws.com/alunos
* Trying 54.77.94.45:443...
* TCP_NODELAY set
* Connected to f7f7lsc06h.execute-api.eu-west-1.amazonaws.com (54.77.94.45) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
*   CAPath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server accepted to use h2
* Server certificate:
*   subject: CN=.execute-api.eu-west-1.amazonaws.com
*   start date: Oct 22 00:00:00 2020 GMT
*   expire date: Nov 20 23:59:59 2021 GMT
*   subjectAltName: host "f7f7lsc06h.execute-api.eu-west-1.amazonaws.com" matched cert's *.execute-api.eu-west-1.amazonaws.com
*   issuer: C=US, O=Amazon, OU=Server CA 1B, CN=Amazon
*   SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x55a1e4849ea0)
* GET /alunos HTTP/2
* Host: f7f7lsc06h.execute-api.eu-west-1.amazonaws.com
* user-agent: curl/7.68.0
* accept: */*
*
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
* HTTP/2 200
* date: Thu, 17 Jun 2021 05:32:14 GMT
* content-type: application/json
* content-length: 78
* apigw-requestid: BDeukhg9j0EEFjw=
* Connection #0 to host f7f7lsc06h.execute-api.eu-west-1.amazonaws.com left intact
hes@HES-PC:~$
```


2º) Foi listado todos os alunos (Apresenta o aluno adicionado anteriormente):

```
hes@HES-PC: ~  
hes@HES-PC:~$ curl -v https://f7f1isc06h.execute-api.eu-west-1.amazonaws.com/alunos  
* Trying 54.77.94.45...  
* TCP_NODELAY set  
* Connected to f7f1isc06h.execute-api.eu-west-1.amazonaws.com (54.77.94.45) port 443 (#0)  
* ALPN, offering h2  
* ALPN, offering http/1.1  
* successfully set certificate verify locations:  
* CAfile: /etc/ssl/certs/ca-certificates.crt  
* CAPath: /etc/ssl/certs  
* TLSv1.3 (OUT), TLS handshake, Client hello (1):  
* TLSv1.3 (IN), TLS handshake, Server hello (2):  
* TLSv1.2 (IN), TLS handshake, Certificate (11):  
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):  
* TLSv1.2 (IN), TLS handshake, Server finished (14):  
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):  
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):  
* TLSv1.2 (OUT), TLS handshake, Finished (20):  
* TLSv1.2 (IN), TLS handshake, Finished (20):  
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256  
* ALPN, server accepted to use h2  
* Server certificate:  
* subject: CN=*.execute-api.eu-west-1.amazonaws.com  
* start date: Oct 22 00:00:00 2020 GMT  
* expire date: Nov 20 23:59:59 2021 GMT  
* subjectAltName: host "f7f1isc06h.execute-api.eu-west-1.amazonaws.com" matched cert's "*.execute-api.eu-west-1.amazonaws.com"  
* issuer: C=US; O=Amazon; OU=Server CA 1B; CN=Amazon  
* SSL certificate verify ok.  
* Using HTTP2, server supports multi-use  
* Connection state changed (HTTP/2 confirmed)  
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0  
* Using Stream ID: 1 (easy handle 0x55a1e4849ea0)  
* GET /alunos HTTP/2  
* Host: f7f1isc06h.execute-api.eu-west-1.amazonaws.com  
* User-Agent: curl/7.68.0  
* accept: */*  
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!  
* HTTP/2 200  
* date: Thu, 17 Jun 2021 05:32:14 GMT  
* content-type: application/json  
* content-length: 74  
* apigw-requestid: BDeukhg9j0EEPjw=  
* Connection #0 to host f7f1isc06h.execute-api.eu-west-1.amazonaws.com left intact  
hes@HES-PC:~$
```

Criação das outras tabelas amazon DynamoDB

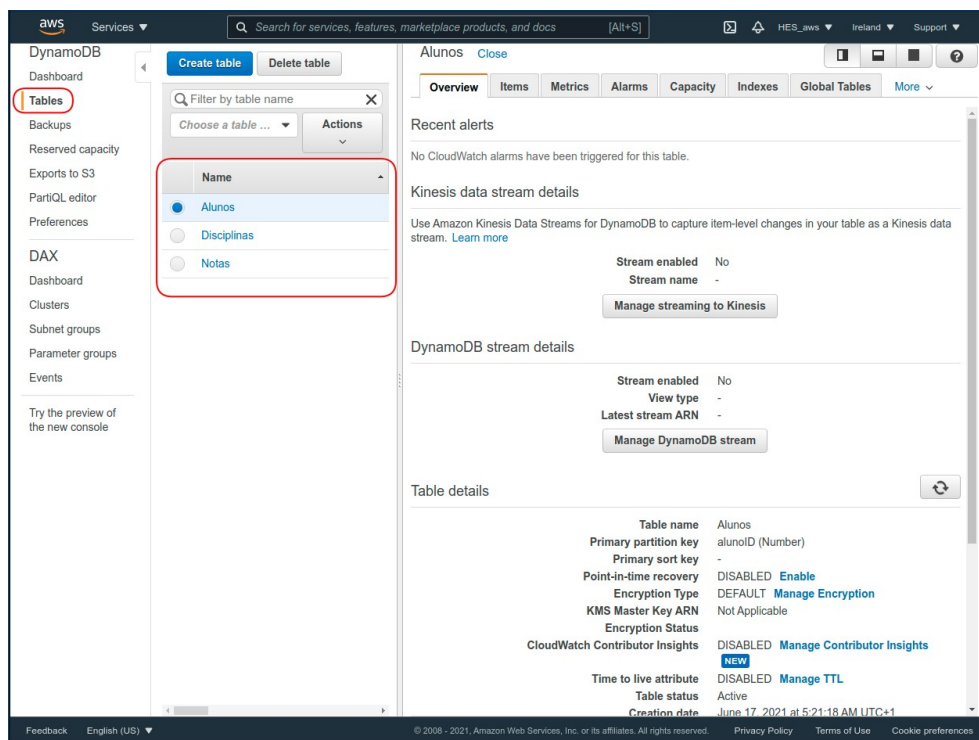
Como já referido, para implementar a arquitetura do trabalho, foi considerado o seguinte diagrama relacional:



Deste modo, além da tabela *Alunos*, foi necessário criar as tabelas *Disciplinas* e a tabela *Notas*, que como os nomes indicam, guardarão todas as disciplinas e todas as notas associadas aos alunos.

Para criar as tabelas no DynamoDB, foi usado o mesmo método apresentado anteriormente na criação da tabela *Alunos*.

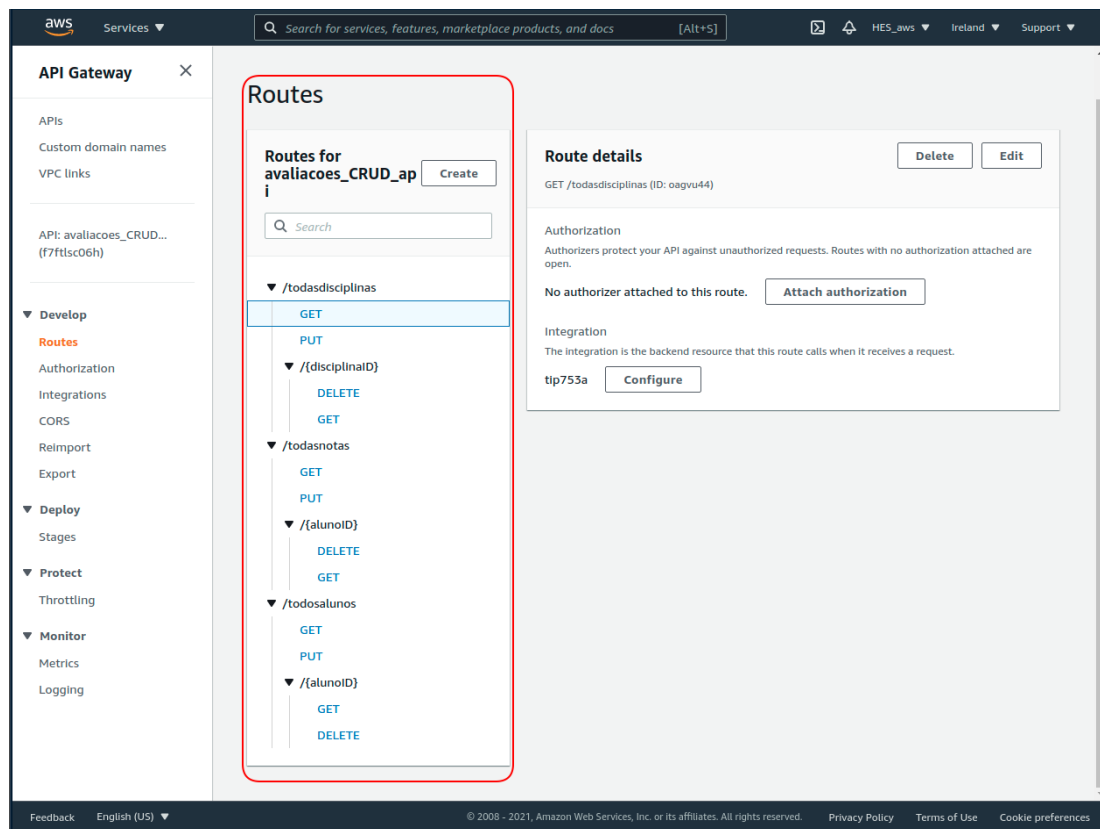
Uma particularidade da tabela *Notas*, é o facto de ter uma chave primária que resulta da combinação de um identificador de aluno e identificador da disciplina, permitindo assim que haja apenas uma nota para cada combinação de aluno e disciplina. Na imagem abaixo, pode-se verificar a criação das tabelas *Disciplinas* e *Notas*:



Criação de novas rotas da API

Após adicionar as novas tabelas ao DynamoDB, foi criado e/ou melhorado algumas rotas da API HTTP, para assim implementar também os métodos CRUD, às novas tabelas.

Após as alterações aplicadas, a API ficou com a seguinte lista de rotas:

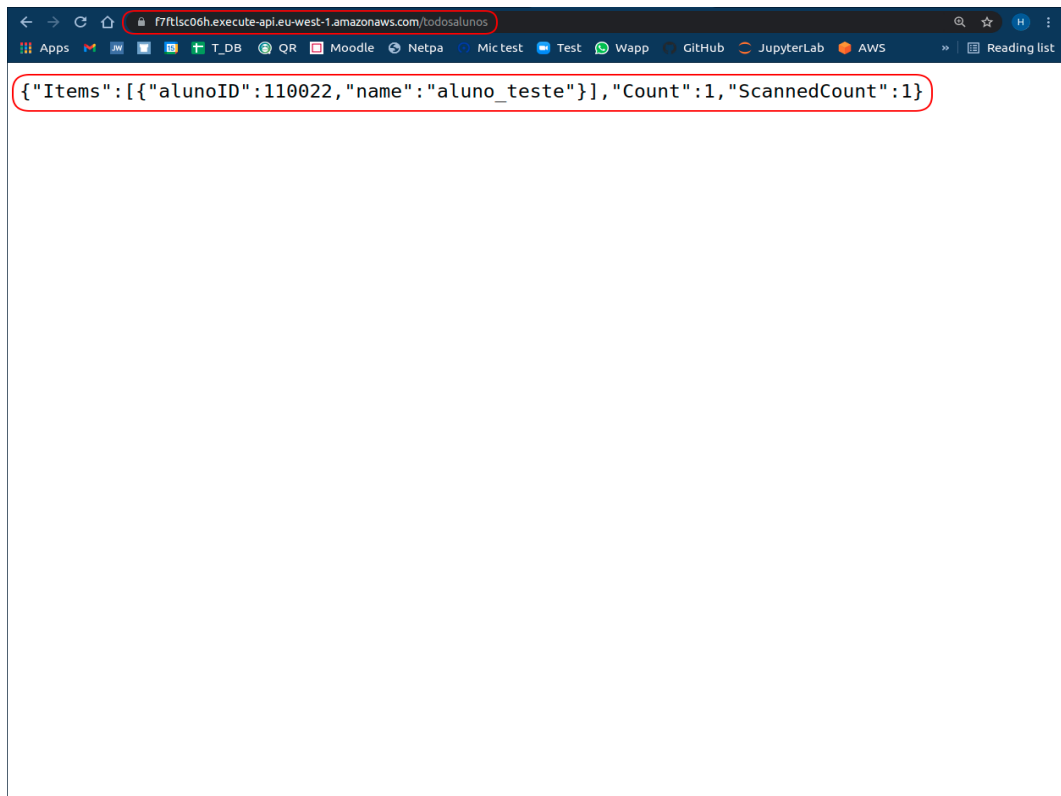


Algo importante para o funcionamento das rotas, foi adicioná-las à integração anteriormente criada e assim associa-las com a função Lambda.

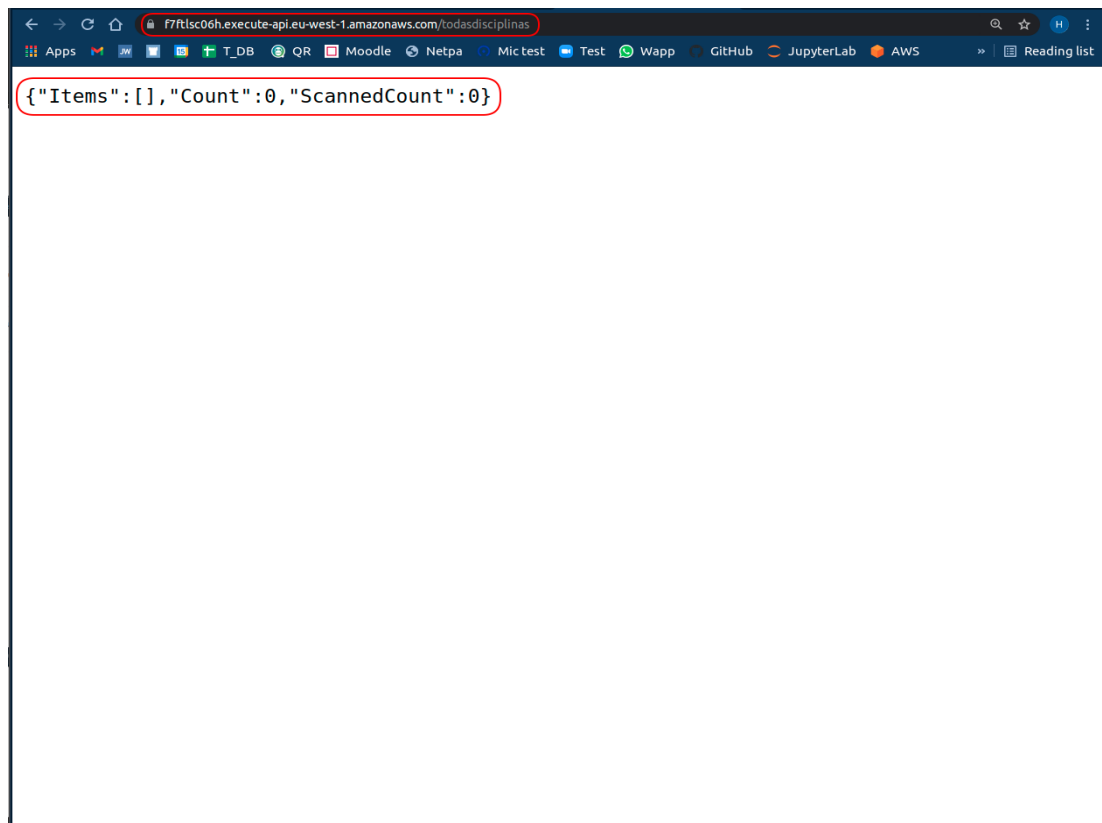
O passo seguinte à criação das rotas, foi editar código da função Lambda de forma a funcionar com as essas rotas.

Depois de efetuar as alterações necessárias, foi necessário fazer os primeiros testes às rotas criadas. As imagens abaixo indicam que é possível obter resposta HTTP com o URL e rotas da API.

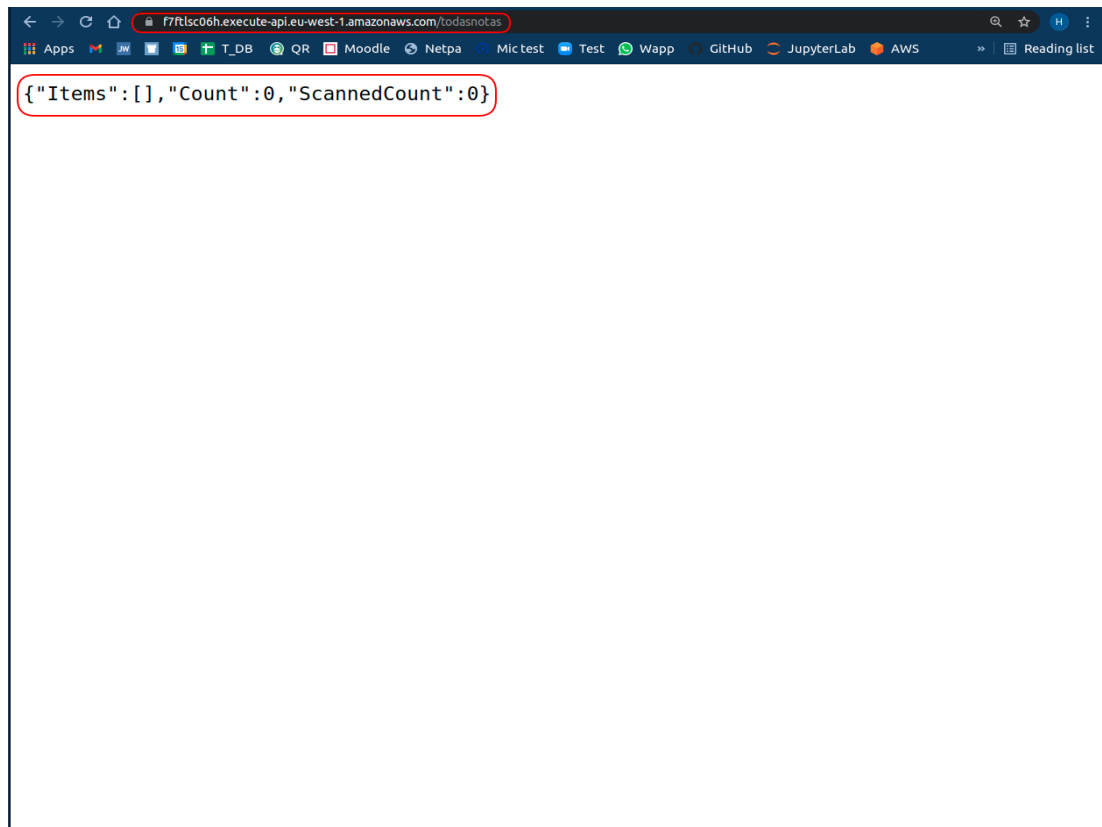
Lista de todos alunos registrados:



Lista de todas as disciplinas:



Lista de todas as notas:



Testes de implementação

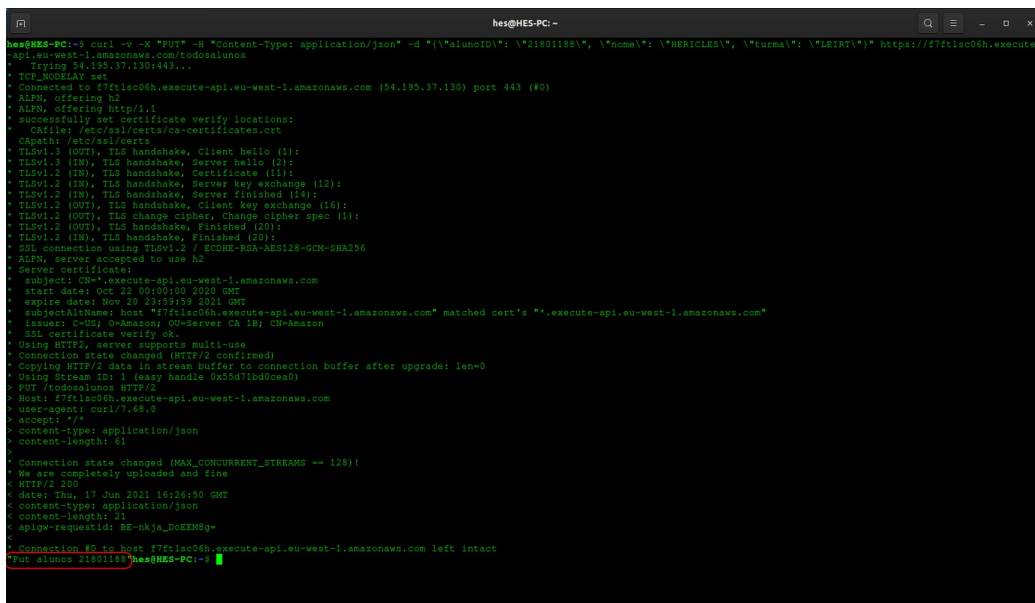
Como nas etapas anteriores, os testes da implementação foram feitos utilizando o recurso CURL.

1. Inserir alunos:

O primeiro teste feito é o de inserção de alunos. Foi realizado utilizando o seguinte comando:

```
curl -v -X "PUT" -H "Content-Type: application/json" -d
{"alunoID": "<alunoID>", "nome": "<nome>", "turma": "<turma>"}
https://<API_id>.execute-api.us-west-1.amazonaws.com/todosalunos
```

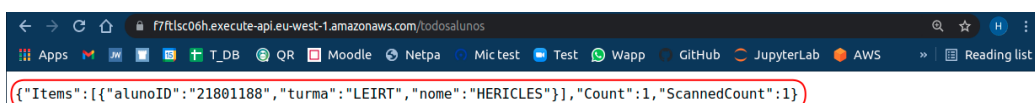
O resultado da execução pode ser verificado nas imagens apresentadas abaixo:



```
hes@HES-PC:~$ curl -v -X "PUT" -H "Content-Type: application/json" -d '{"alunoID": "21801188", "nome": "HERICLES", "turma": "LEIRT"}' https://f7ftlsc06h.execute-api.us-west-1.amazonaws.com/todosalunos
Trying 54.195.37.130:443...
TCP_NODELAY set
Connected to f7ftlsc06h.execute-api.us-west-1.amazonaws.com (54.195.37.130) port 443 (#0)
ALPN, offering http/1.1
successfully set certificate verify locations:
  CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
TLSv1.3 (OUT), TLS handshake, Client hello (1):
TLSv1.3 (IN), TLS handshake, Server hello (2):
TLSv1.2 (IN), TLS handshake, Certificate (11):
TLSv1.2 (IN), TLS handshake, Server key exchange (12):
TLSv1.2 (IN), TLS handshake, Server finished (14):
TLSv1.1 (OUT), TLS handshake, Client key exchange (16):
TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.2 (OUT), TLS handshake, Finished (20):
TLSv1.2 (IN), TLS handshake, Finished (20):
SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
ALPN, server accepted to use h2
Server certificate:
  subject: CN=*.execute-api.us-west-1.amazonaws.com
  start date: Oct 22 00:00:00 2020 GMT
  expire date: Nov 20 23:59:59 2021 GMT
  subjectAltName: host "f7ftlsc06h.execute-api.us-west-1.amazonaws.com" matched cert's "*.execute-api.us-west-1.amazonaws.com"
  issuer: C=US; O=Amazon; OU=Server CA 1B; CN=Amazon
SSL certificate verify ok.
Using HTTP2, server supports multi-use
Connection state changed (HTTP/2 confirmed)
Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
Using Stream ID: 1 (easy handle 0x555471bd0ca0)
PUT /todosalunos HTTP/2
Host: f7ftlsc06h.execute-api.us-west-1.amazonaws.com
user-agent: curl/7.68.0
accept: */*
content-type: application/json
content-length: 61

Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
We are completely uploaded and fine
HTTP/2 200
date: Thu, 17 Jun 2021 14:26:50 GMT
content-type: application/json
content-length: 21
apigw-requestid: BE-nkja_DoEEM0g=

{"Items": [{"alunoID": "21801188", "turma": "LEIRT", "nome": "HERICLES"}], "Count": 1, "ScannedCount": 1}
Connection #0 to host f7ftlsc06h.execute-api.us-west-1.amazonaws.com left intact
hes@HES-PC:~$
```

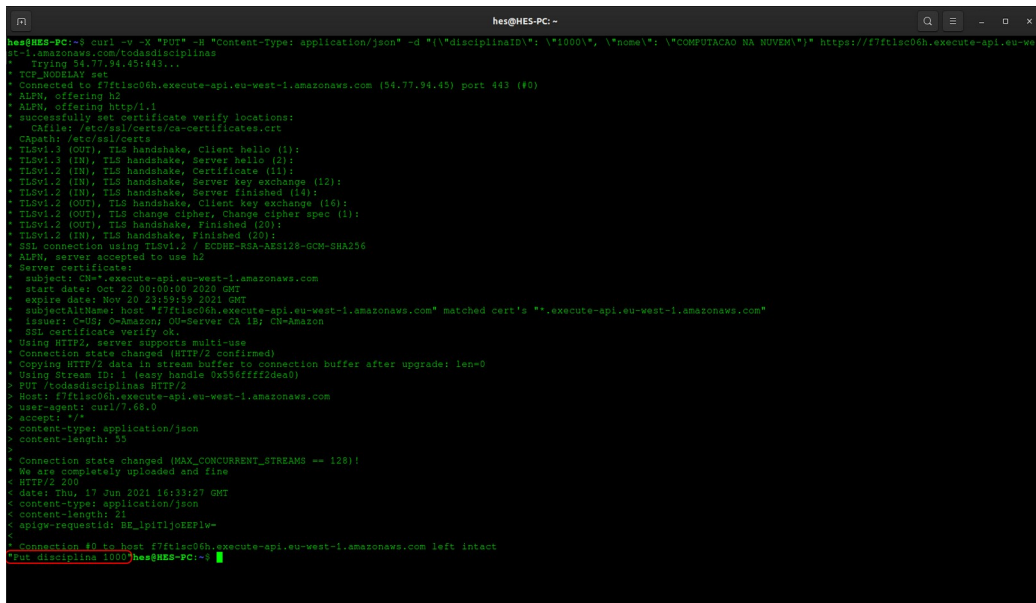


2. Inserir disciplinas:

Um teste idêntico foi feito com a inserção de disciplinas. Foi utilizado o seguinte comando:

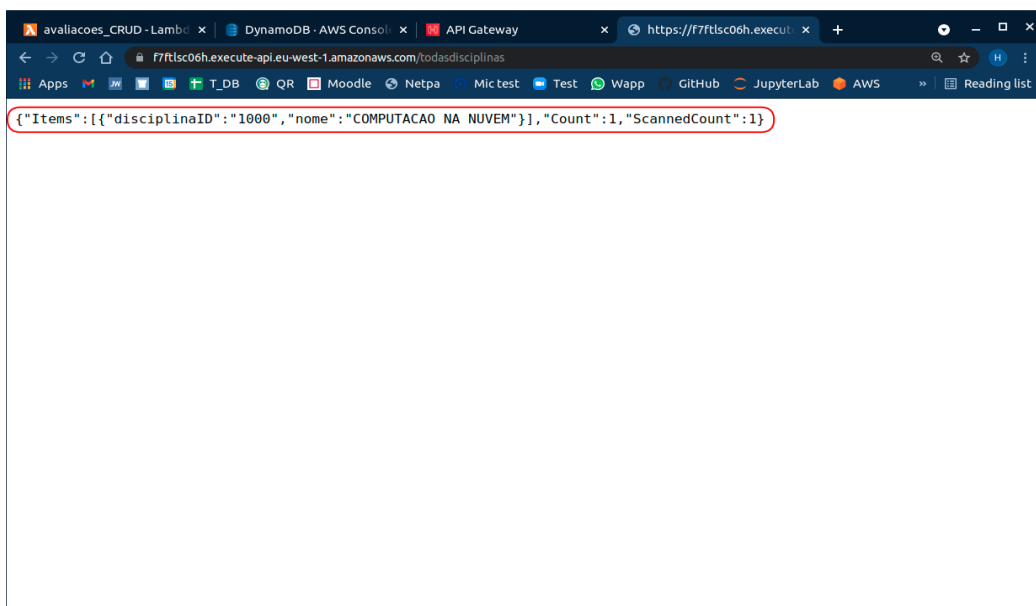
```
curl -v -X "PUT" -H "Content-Type: application/json" -d '{"disciplinaID": "<disciplinaID>", "nome": "<disciplinaNome>"}' https://<API_id>.execute-api.us-west-1.amazonaws.com/todasdisciplinas
```

O resultado da execução pode ser verificado nas imagens apresentadas abaixo:



```
hes@HES-PC:~$ curl -v -X "PUT" -H "Content-Type: application/json" -d '{"disciplinaID": "1000", "nome": "COMPUTACAO NA NUVEU"}' https://f7f1sc06h.execute-api.eu-west-1.amazonaws.com/todasdisciplinas
Trying 54.77.94.45...
Connected to f7f1sc06h.execute-api.eu-west-1.amazonaws.com (54.77.94.45) port 443 (#0)
ALPN, offering h2
ALPN, offering http/1.1
successfully set certificate verify locations:
  CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
TLSv1.3 (OUT), TLS handshake, Client hello (1):
TLSv1.3 (IN), TLS handshake, Server hello (2):
TLSv1.2 (IN), TLS handshake, Certificate (11):
TLSv1.2 (IN), TLS handshake, Server key exchange (12):
TLSv1.2 (IN), TLS handshake, Server finished (14):
TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.2 (OUT), TLS handshake, Finished (20):
TLSv1.2 (IN), TLS handshake, Finished (20):
SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
ALPN, server accepted to use h2
Server certificate:
  subject: CN=.execute-api.eu-west-1.amazonaws.com
  start date: Oct 22 00:00:00 2020 GMT
  expire date: Nov 20 23:59:59 2021 GMT
  subjectAltName: host "f7f1sc06h.execute-api.eu-west-1.amazonaws.com" matched cert's *.execute-api.eu-west-1.amazonaws.com
  issuer: C=US; O=Amazon; OU=Server CA 18; CN=Amazon
SSL certificate verify ok.
Using HTTP2, server supports multi-use
Connection state changed (HTTP/2 confirmed)
Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
Using Stream ID: 1 (easy handle 0a36ffff12e0)
PUT /todasdisciplinas HTTP/2
Host: f7f1sc06h.execute-api.eu-west-1.amazonaws.com
user-agent: curl/7.68.0
accept: */*
content-type: application/json
content-length: 55

Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
We are completely uploaded and fine
HTTP/2 200
date: Thu, 17 Jun 2021 18:33:27 GMT
content-type: application/json
content-length: 21
spigw-requestid: BGlp1IjoEEFlw-
Connection #0 to host f7f1sc06h.execute-api.eu-west-1.amazonaws.com left intact
*Put disciplina 1000*hes@HES-PC:~$
```



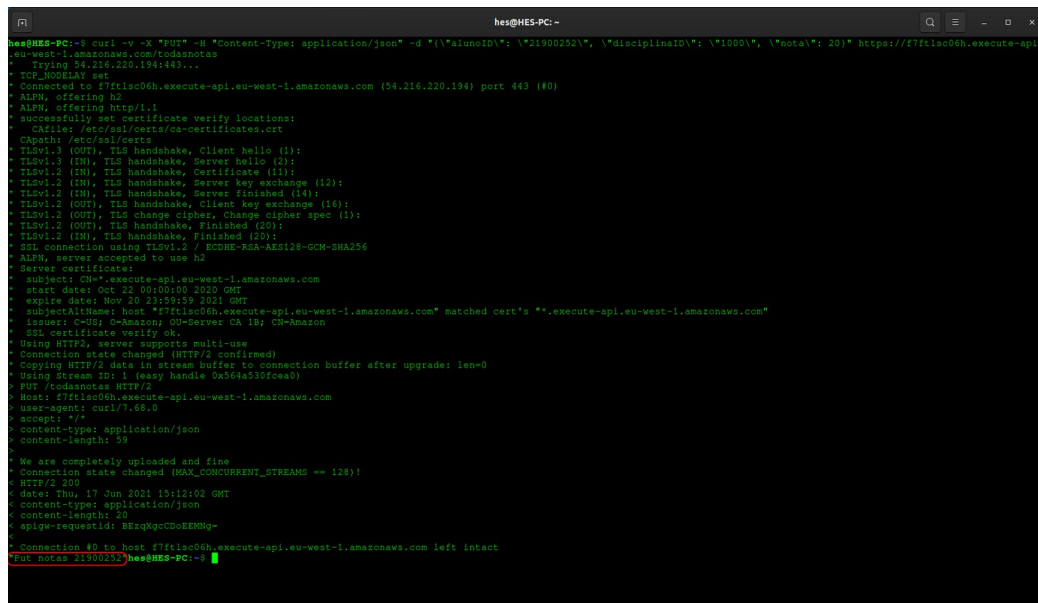
```
avaliacoes_CRUD - Lambda x DynamoDB - AWS Console x API Gateway x https://f7f1sc06h.execute-api.eu-west-1.amazonaws.com/todasdisciplinas
{"Items":[{"disciplinaID":"1000","nome":"COMPUTACAO NA NUVEU"}], "Count":1, "ScannedCount":1}
```

3. Inserir notas:

Um teste idêntico foi feito com a inserção de disciplinas. Foi utilizado o seguinte comando:

```
curl -v -X "PUT" -H "Content-Type: application/json" -d '{"alunoID": "<alunoID>", "disciplinaID": "<disciplinaID>", "nota": "<nota>"}' https://<API_id>.execute-api.us-west-1.amazonaws.com/todasNotas
```

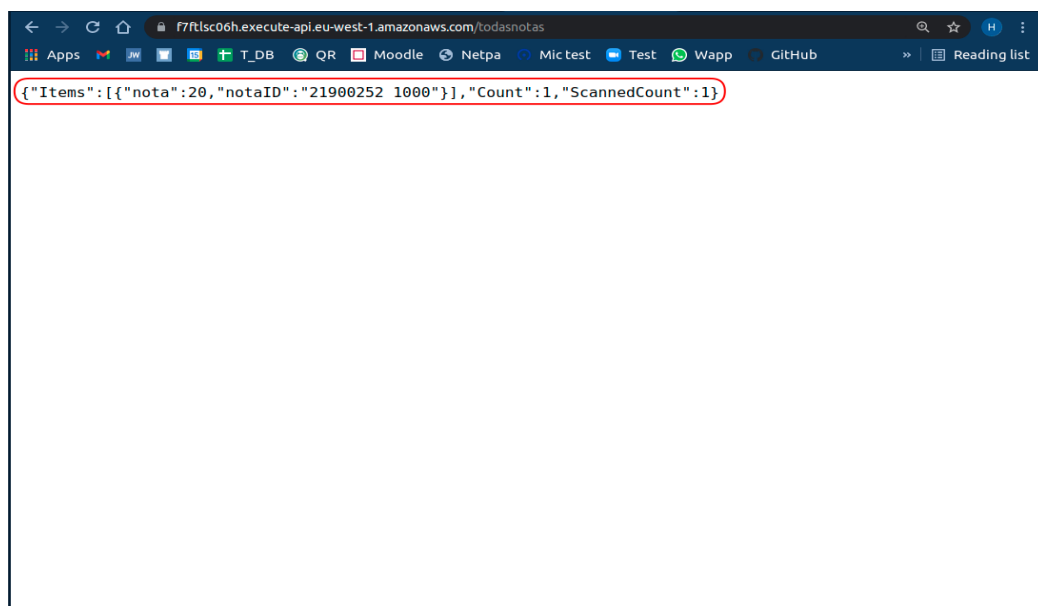
O resultado da execução pode ser verificado nas imagens apresentadas abaixo:



```
hes@HES-PC:~$ curl -v -X "PUT" -H "Content-Type: application/json" -d '{"alunoID": "21900252", "disciplinaID": "1000", "nota": 20}' https://f7f7tisc06h.execute-api.us-west-1.amazonaws.com/todasNotas
Trying 54.216.220.194:443...
TCP_NODELAY set
Connected to f7f7tisc06h.execute-api.us-west-1.amazonaws.com (54.216.220.194) port 443 (#0)
ALPN, offering h2
ALPN, offering http/1.1
successfully set certificate verify locations:
  CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
TLSv1.3 (OUT), TLS handshake, Client hello (1):
TLSv1.3 (IN), TLS handshake, Server hello (2):
TLSv1.2 (IN), TLS handshake, Certificate (11):
TLSv1.2 (IN), TLS handshake, Server key exchange (12):
TLSv1.2 (IN), TLS handshake, Server finished (14):
TLSv1.2 (OUT), TLS handshake, Client key exchange (15):
TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.2 (OUT), TLS handshake, Finished (20):
TLSv1.2 (IN), TLS handshake, Finished (20):
SSL connection using TLSv1.3 / ECDHE-RSA-AES128-GCM-BR0356
ALPN, server accepted to use h2
server certificate:
subject: CN=*.execute-api.us-west-1.amazonaws.com
start date: Oct 22 00:00:00 2020 GMT
expire date: Nov 20 23:59:59 2021 GMT
subjectAltName: host "f7f7tisc06h.execute-api.us-west-1.amazonaws.com" matched cert's *.execute-api.us-west-1.amazonaws.com
issuer: C=US; O=Amazon; OU=Server CA 1B; CN=Amazon
PEM certificate verify ok.
Using HTTP2, server supports multi-use
Connection state changed (HTTP2 confirmed)
Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
Using Stream ID: 1 (easy handle 0x564e330f0e40)
PUT /todasNotas HTTP/2
Host: f7f7tisc06h.execute-api.us-west-1.amazonaws.com
user-agent: curl/7.68.0
accept: */*
content-type: application/json
content-length: 59

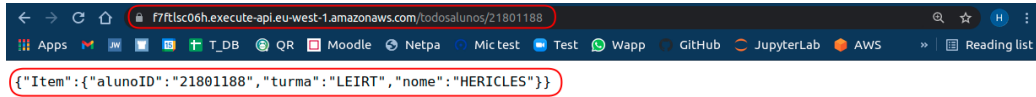
We are completely uploaded and fine
Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
HTTP/2 200
date: Thu, 17 Jun 2021 15:12:02 GMT
content-type: application/json
content-length: 20
spiqw-requestid: HEqXgcDcEEMNg=

Connection #0 to host f7f7tisc06h.execute-api.us-west-1.amazonaws.com left intact
PUT notas 21900252 hes@HES-PC:~$
```



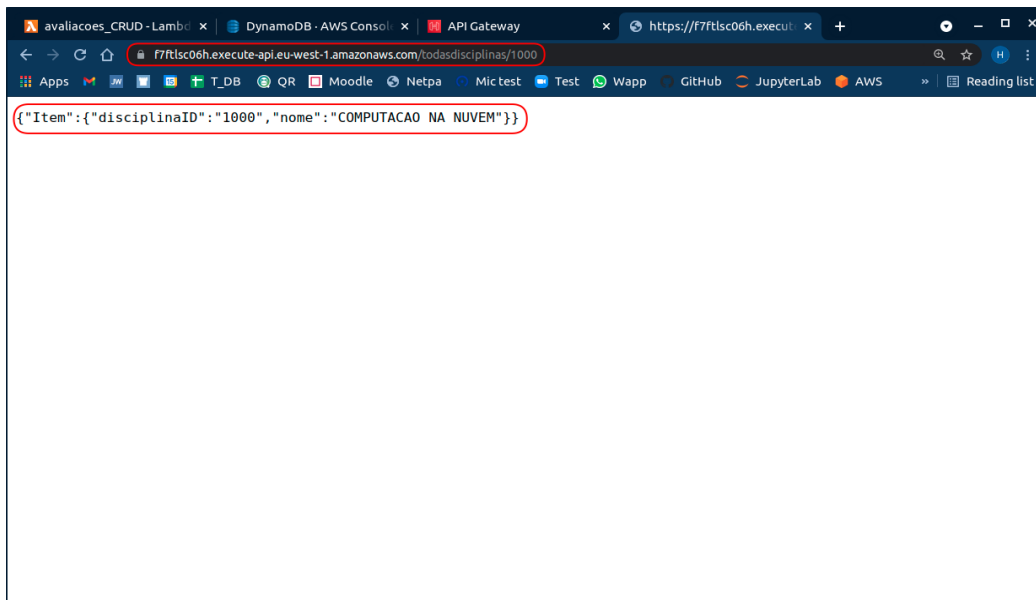
4. Consulta de alunos:

O teste de consulta pode-se verificar através do CURL, ou utilizando diretamente o URL da API. A imagem abaixo apresenta a consulta feita a partir do browser:



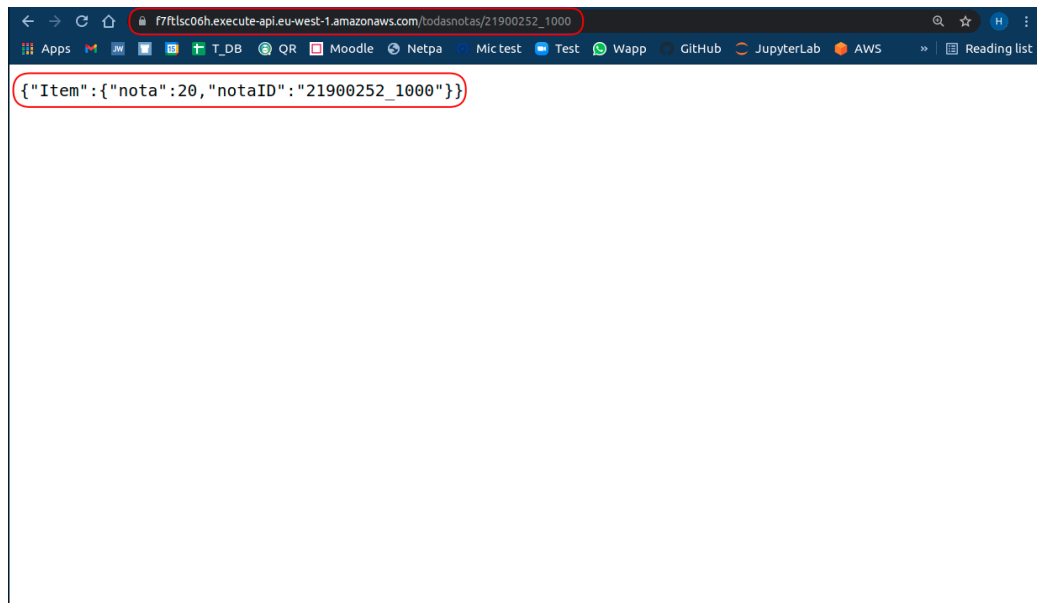
5. Consulta de disciplinas:

A imagem abaixo apresenta a consulta de uma disciplina feita a partir do browser:



6. Consulta de notas:

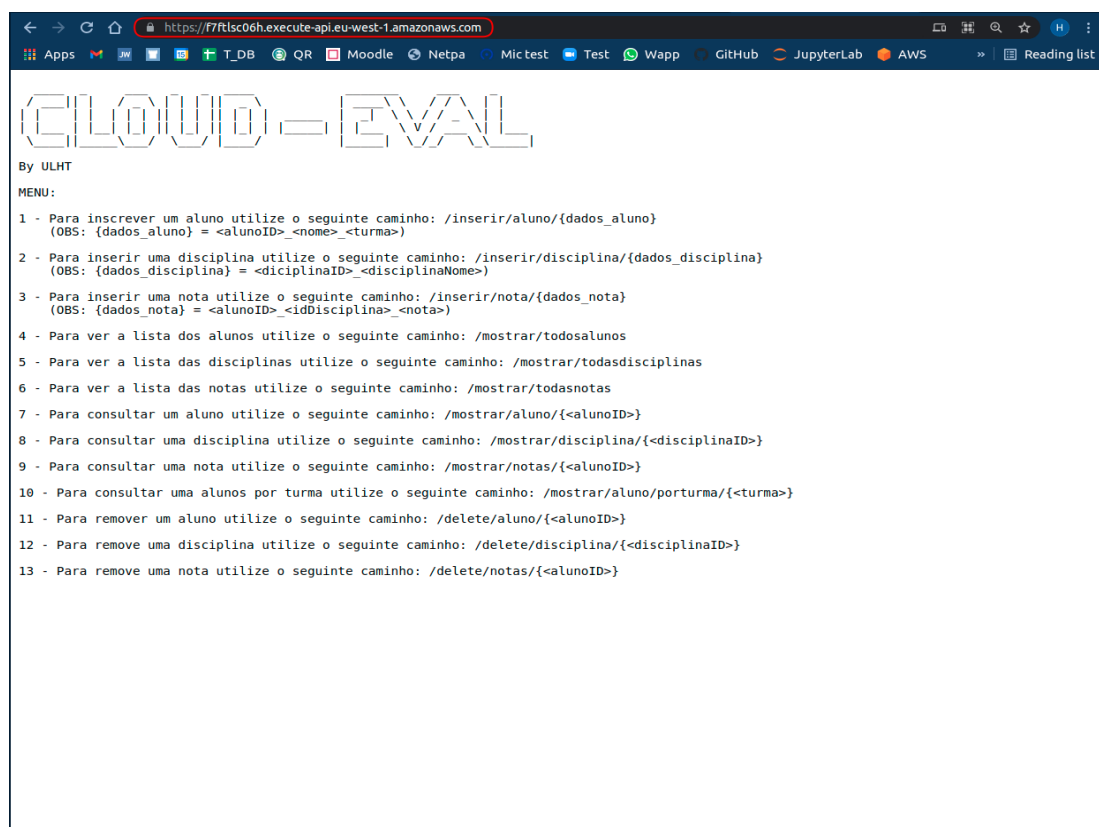
A imagem abaixo apresenta a consulta de uma nota feita a partir do browser:



Customização

Para não usar sempre os comandos CURL, para manipular as tabelas do DynamoDB, foi implementado um menu, que embora seja rudimentar, permite a interação com o backend apenas utilizando o browser.

O menu principal é apresentado na raiz do site. Para isso foi necessário criar uma rota na API do seguinte tipo: **"ANY /"**. A imagem abaixo permite observar os aspectos do menu e as operações implementadas:



A ideia de implementação, foi usar o próprio url da página como identificador das operações (CRUD) e passagem de parâmetros. O único problema foi o facto da implementação funcionar apenas com o método GET do HTTP, mesmo se a operação efetiva for escrita ou remoção de objectos no DynamoDB.

Deste modo, foi necessário criar e editar novas rotas da API Gateway. A imagem apresentada abaixo, indica as rotas usadas para esta implementação:

API Gateway

Search for services, features, marketplace products, and docs [Alt+S]

HE3_vms Ireland Support

API: avaliacaoes_CRUD... (77fbc09a)

Develop

- Routes
- Authorization
- Integrations
- CORS
- Reimport
- Export

Deploy

- Stages

Protect

- Throttling

Monitor

- Metrics
- Logging

Search

ANY

- ▼ /delete
 - ▼ /aluno
 - ▼ /{id_aluno} GET
 - ▼ /notas
 - ▼ /{id_aluno} GET
 - ▼ /disciplina
 - ▼ /{id_disciplina} GET
- ▼ /insistir
 - ▼ /nota
 - ▼ /{dados_nota} GET
 - ▼ /disciplina
 - ▼ /{dados_disciplina} GET
 - ▼ /aluno
 - ▼ /{dados_aluno} GET
- ▼ /mostrar
 - ▼ /todasnotas GET
 - ▼ /aluno
 - ▼ /porturma
 - ▼ /{turma} GET
 - ▼ /{id_aluno} GET
 - ▼ /disciplina
 - ▼ /{id_disciplina} GET
 - ▼ /notas
 - ▼ /{id_aluno} GET
 - ▼ /todasdisciplinas GET
 - ▼ /todosalunos GET

GET /mostrar/aluno/porturma/{turma} (ID: apcwn8)

Authorization

Authorizers protect your API against unauthorized requests. Routes with no authorization attached are open.

No authorizer attached to this route. [Attach authorization](#)

Integration

The integration is the backend resource that this route calls when it receives a request.

tip753a [Configure](#)

Feedback English (US)

© 2008 - 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Testes e demonstrações

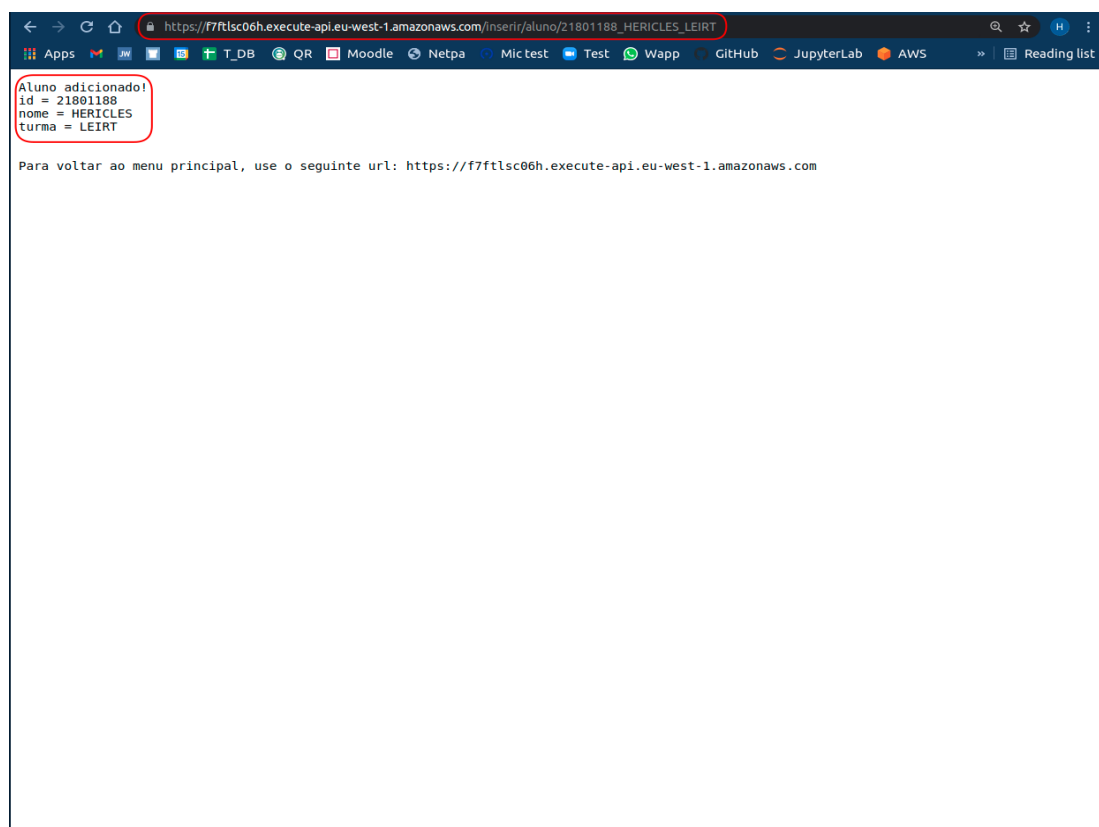
Esta parte do relatório mostra os testes realizados na nova implementação, de acordo com os requisitos do trabalho.

1. Inserir alunos:

O primeiro teste a ser apresentado é a de inserção de alunos à base de dados (DynamoDB). Antes de efetuar os testes, o DynamoDB foi esvaziado para assim não conter itens dos testes anteriores. Para testar, foi inserido valores reais dos alunos do grupo, assim como valores apenas representativos.

Como é indicado no menu principal, estando no url da API, deve-se usar o seguinte caminho para inserção de alunos: `/inserir/aluno/{dados_aluno}`. A variável **`dados_alunos`** deve ter o seguinte formato: `<alunoID>_<nome>_<turma>`.

Após a inserção, é apresentado o seguinte output, indicado na figura abaixo:

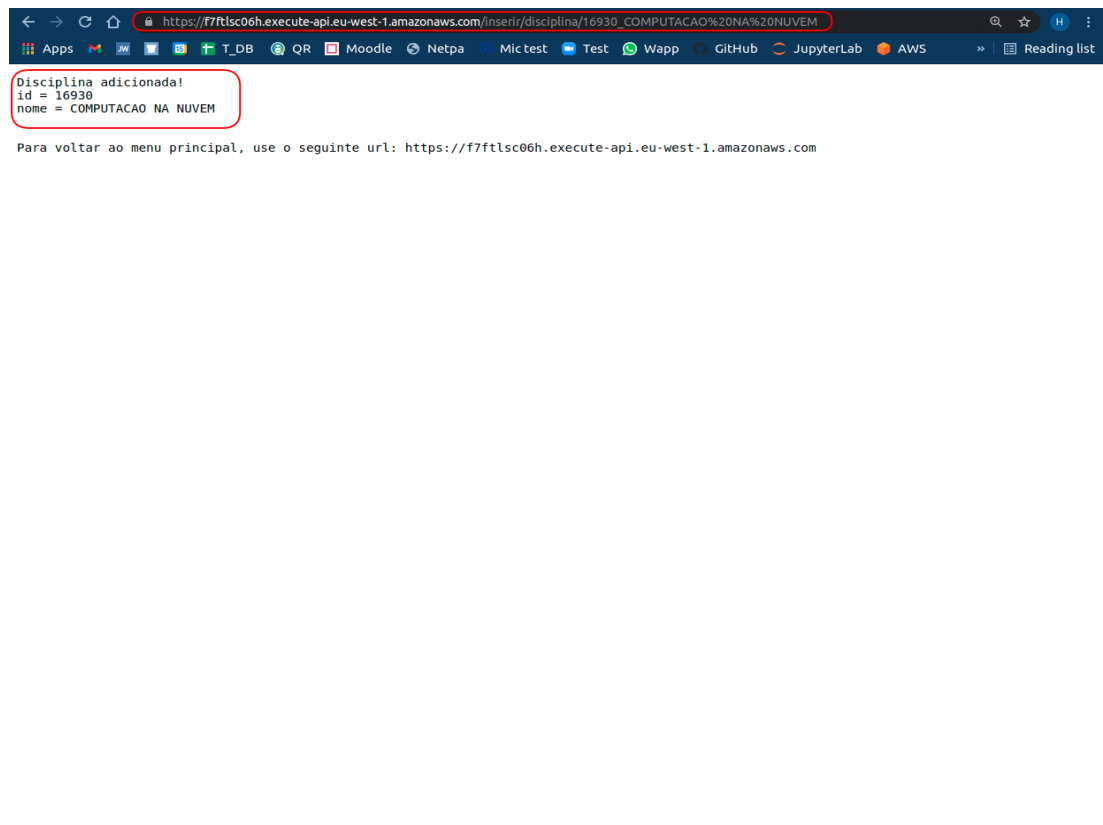


2. Inserir disciplinas:

A lógica de inserção de disciplinas é semelhante à lógica usada para inserir alunos à lista.

Como é indicado no menu principal, deve-se usar o seguinte caminho para inserção de disciplinas: **/inserir/disciplina/{dados_disciplina}**. A variável **dados_disciplina** deve ter o seguinte formato: **<disciplinaID>_<disciplinaNome>**.

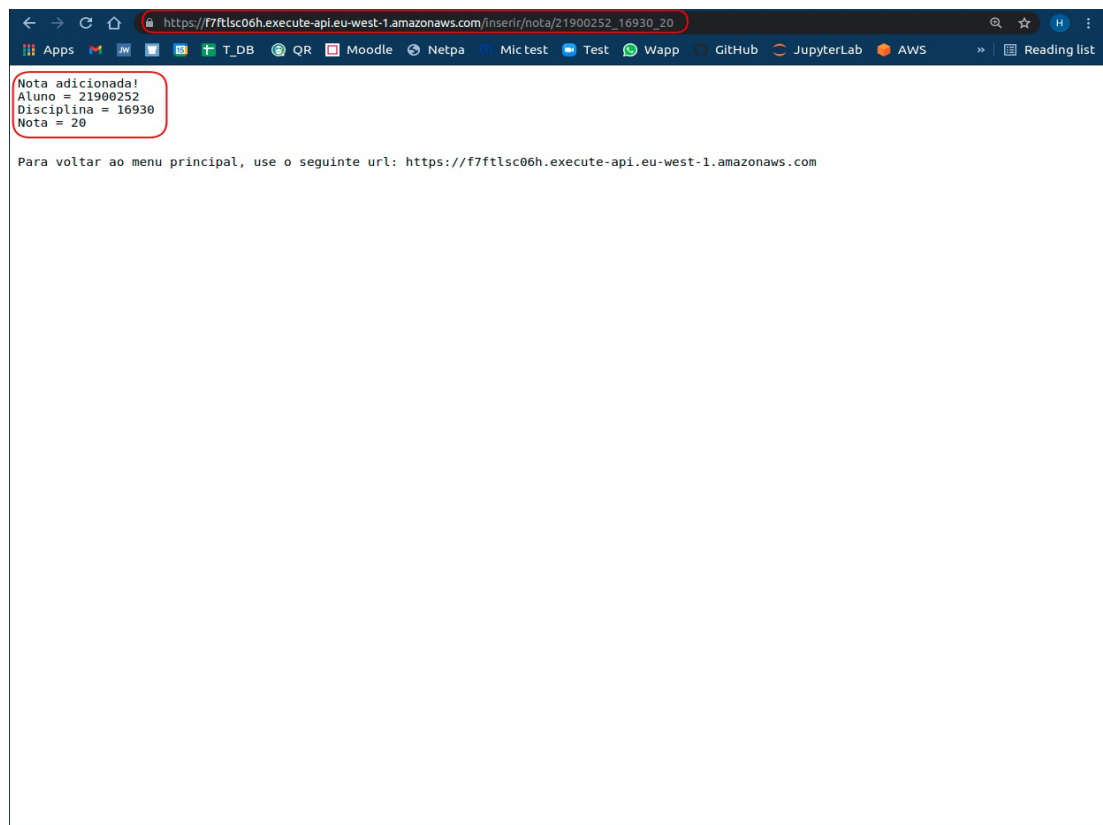
Após a inserção, é apresentado o seguinte output, indicado na figura abaixo:



3. Inserir uma nota:

Para inserir uma nota à base de dados deve-se usar o seguinte caminho para inserção de disciplinas: **/inserir/nota/{dados_nota}**. A variável **dados_nota** deve ter o seguinte formato: **<alunoID>_<idDisciplina>_<nota>**.

Após a inserção, é apresentado o seguinte output, indicado na figura abaixo:



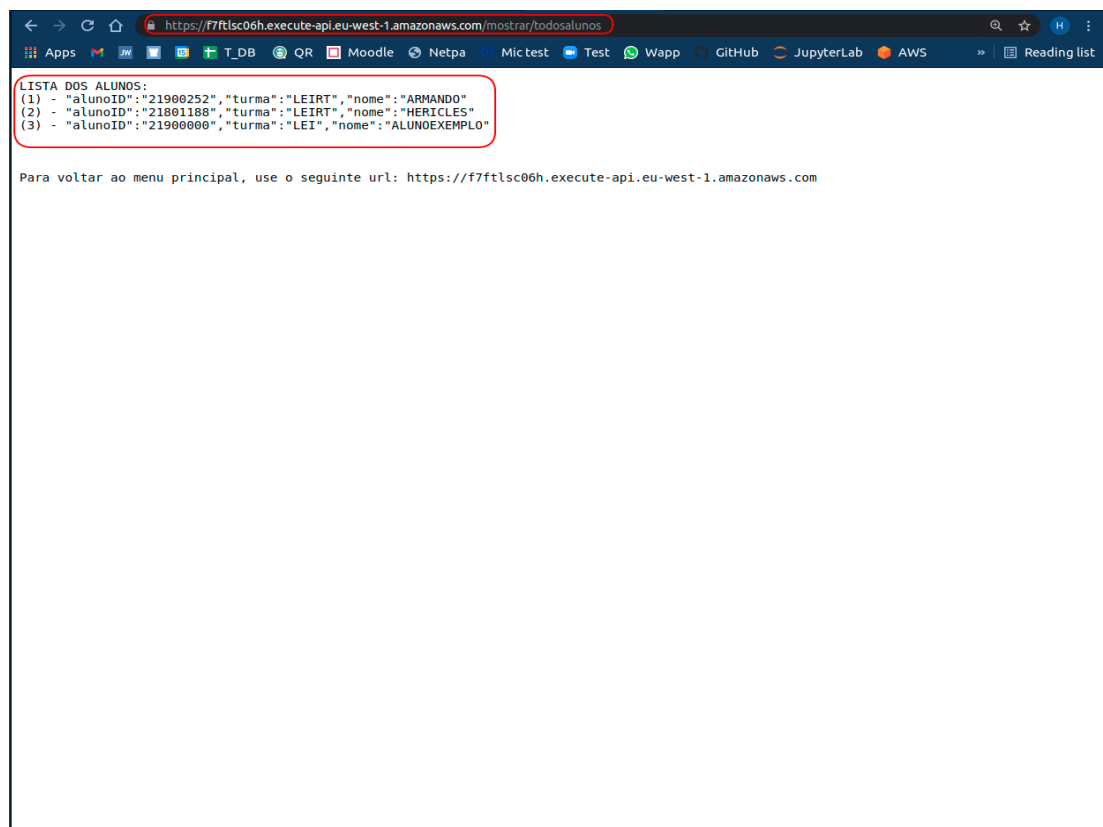
4. Consultas gerais:

Também foi implementado a consulta geral dos itens de cada categoria, guardados na base de dados. Como é apresentado no menu principal, para consultas genéricas devem ser usados os seguintes caminhos:

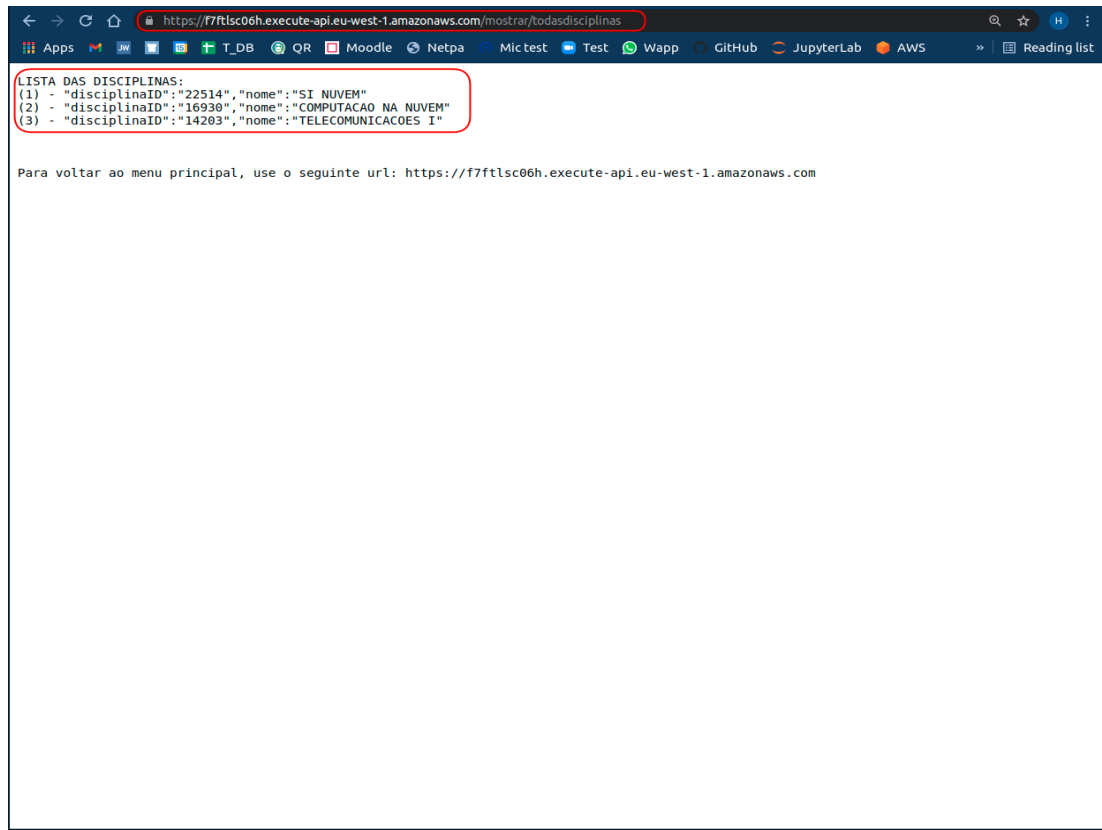
- + **/mostrar/todosalunos**
- + **/mostrar/todasdisciplinas**
- + **/mostrar/todasnotas**

As imagens abaixo indicam como é apresentado o output das consultas gerais para cada uma das entidades das tabelas do DynamoDB.

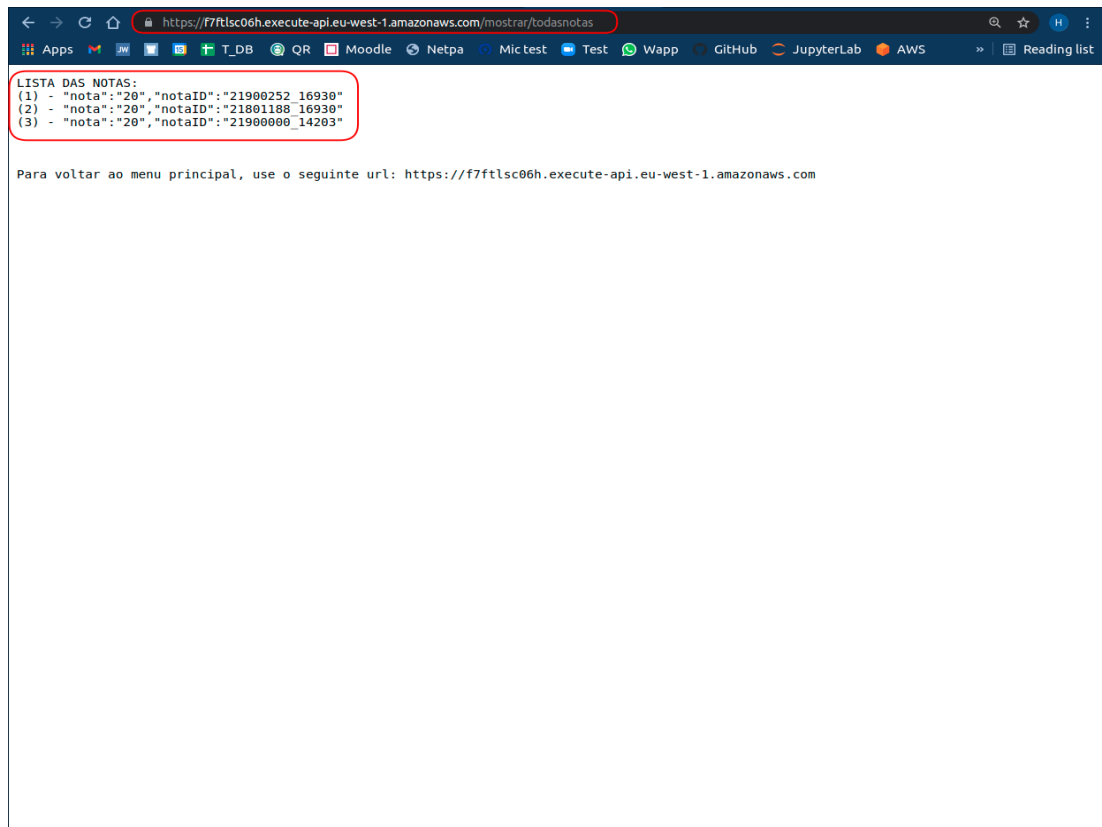
Lista dos Alunos:



Lista das Disciplinas:



Lista das Notas:



5. Consultas específicas:

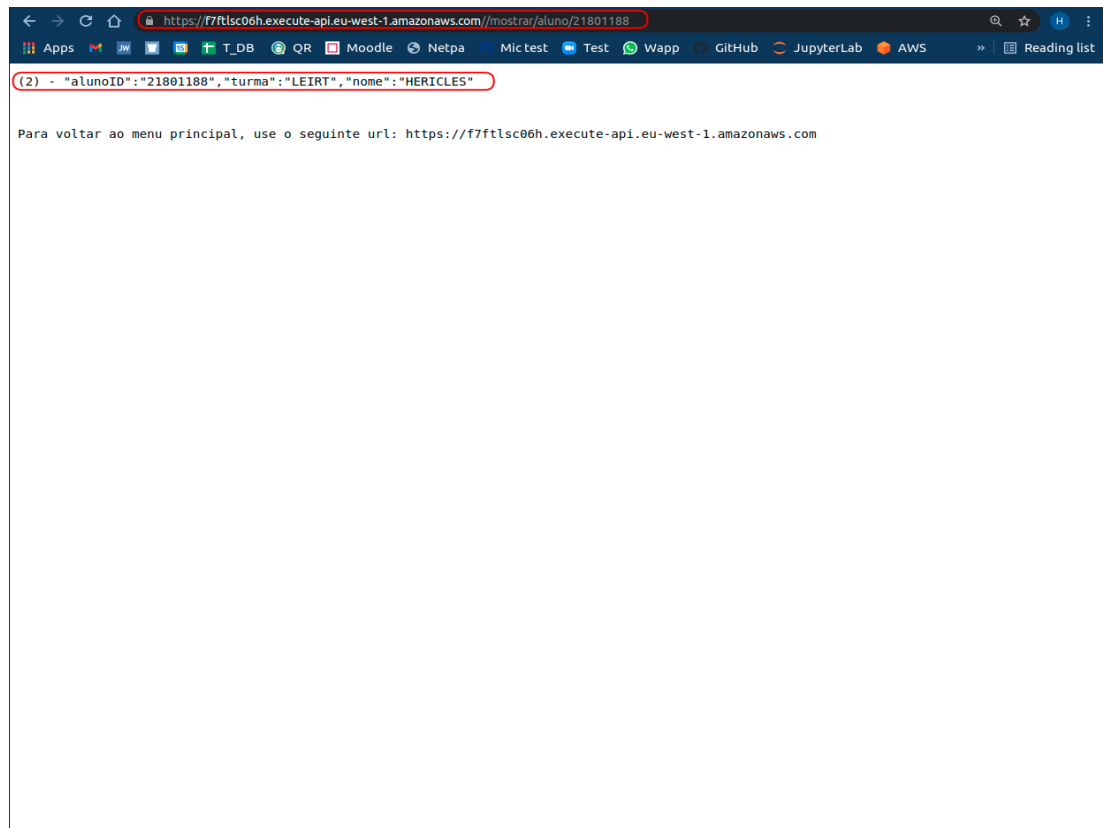
Foi também implementado operações de consultas específicas, utilizando o número identificador dos itens.

De acordo com o menu principal, devem ser usados os seguintes caminhos:

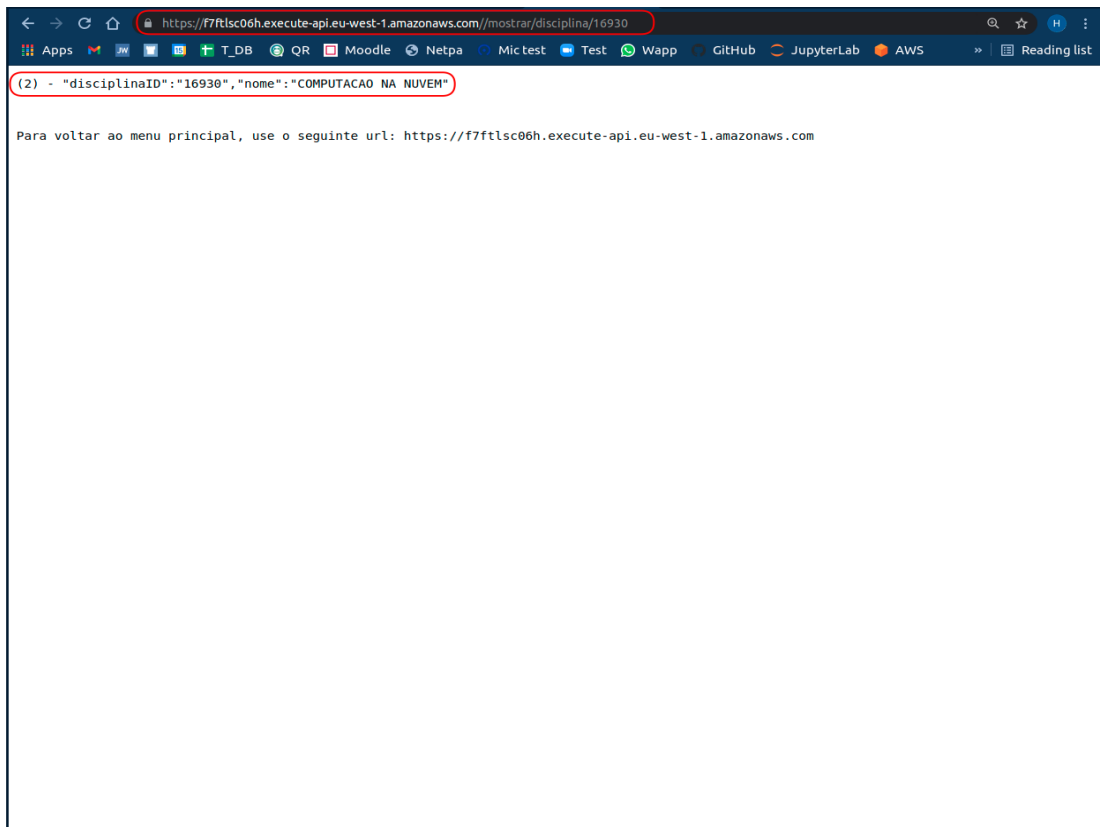
```
+ /mostrar/aluno/{alunoID}
+ /mostrar/disciplina/{<disciplinaID>}
+ /mostrar/notas/{<alunoID>}
```

As imagens abaixo indicam qual é o output das consultas indicadas acima:

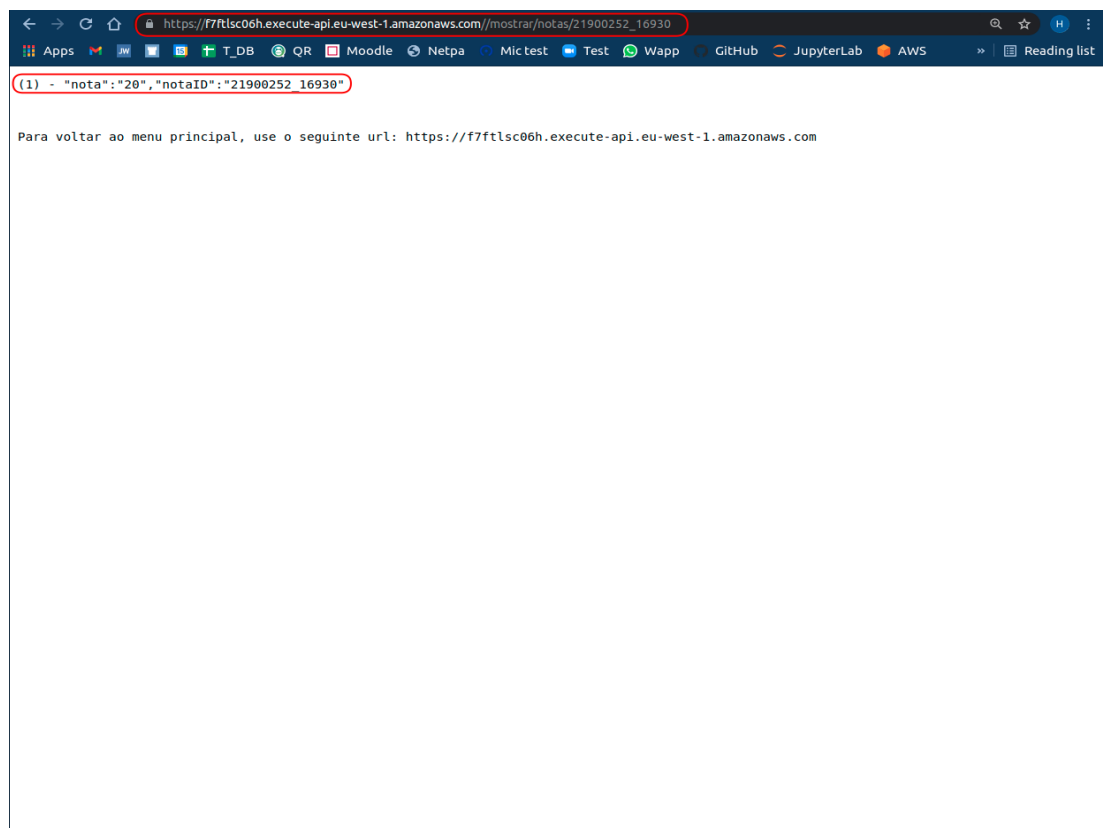
Consulta de um aluno:



Consulta de uma disciplina:



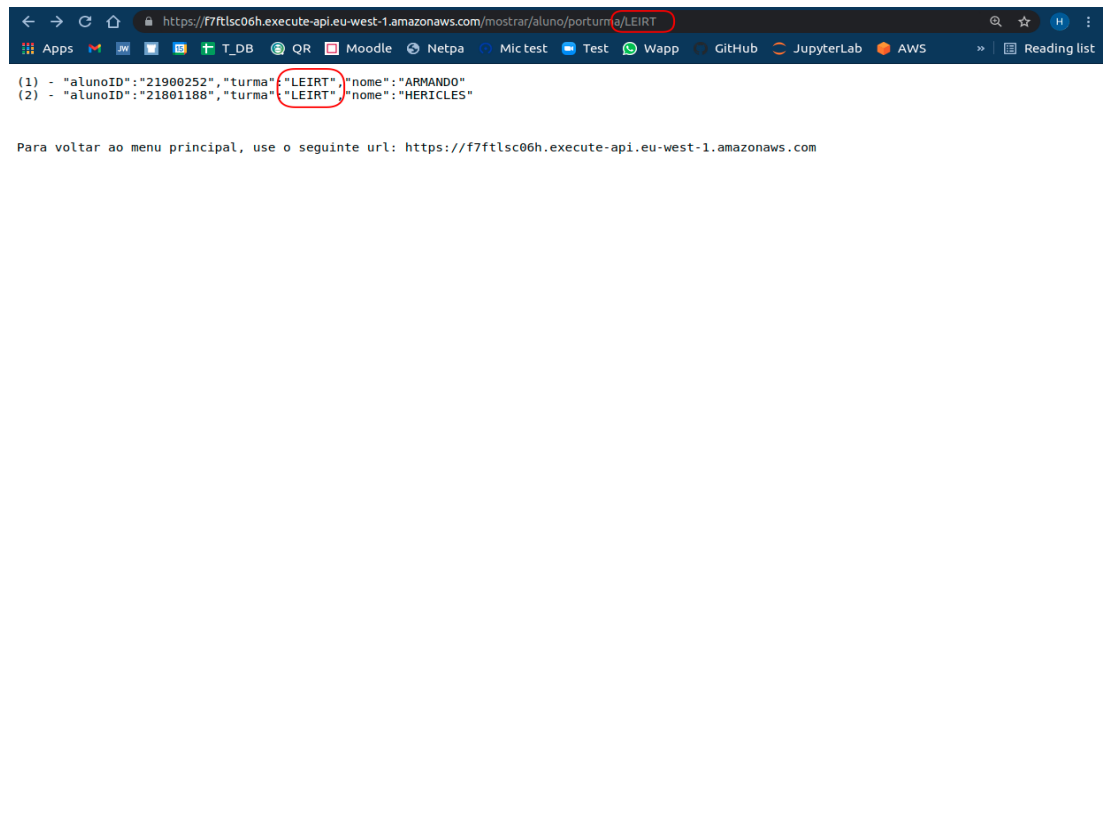
Consulta de uma nota:



Também foi implementada uma operação de consulta que permite filtrar a lista dos alunos de uma só turma. Para efetuar esta consulta, deve-se usar o seguinte caminho:

`/mostrar/aluno/porturma/{<turma>}`

A imagem indicada abaixo, apresenta o resultado de uma busca de alunos de acordo com a turma:



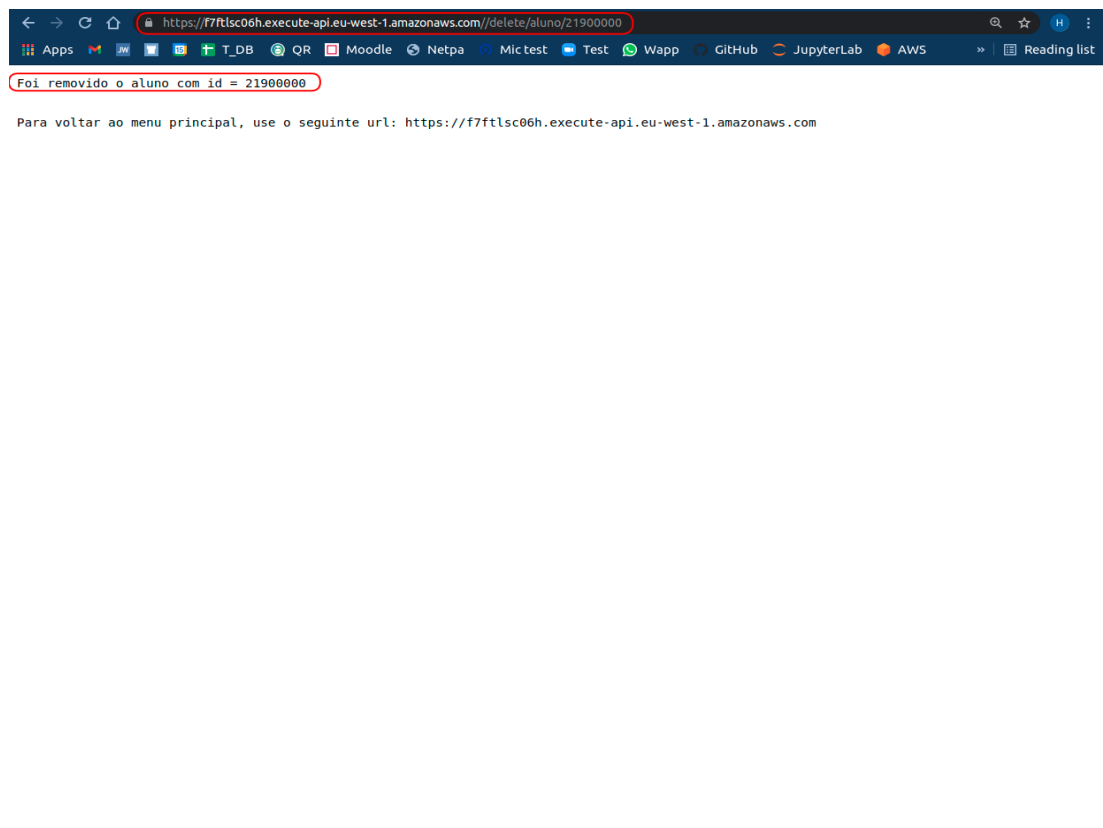
6. Remover itens:

A última operação da conhecida CRUD, é o remover, e por isso foi também implementado operações que permitem remover itens do DynamoDB.

Para isso deve-se usar os seguintes caminhos:

```
+ /delete/aluno/{<alunoID>}  
+ /delete/disciplina/{<disciplinaID>}  
+ /delete/notas/{<alunoID>}
```

Já que o comportamento é semelhante para os vários itens a imagem abaixo representa apenas a remoção de um aluno da base de dados:



Para concluir, as imagens abaixo, apresentam o estado de cada tabela do DynamoDB, após as operações efetuadas:

Amazon AWS Management Console - DynamoDB

Search for services, features, marketplace products, and docs [Alt+S]

Services: HES_aws Ireland Support

DynamoDB

- Dashboard
- Tables
 - Backups
 - Reserved capacity
 - Exports to S3
 - PartiQL editor
 - Preferences
- DAX
 - Dashboard
 - Clusters
 - Subnet groups
 - Parameter groups
 - Events
- Try the preview of the new console

Alunos

Create table Delete table

Filter by table name

Choose a table ... Actions

Name

- Alunos
- Disciplinas
- Notas

Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Contributor Insights Triggers Access control Tags

Create item Actions

Scan: [Table] Alunos: alunoID Viewing 1 to 2 items

Scan [Table] Alunos: alunoID

Add filter

Start search

alunoID	nome	turma
21801188	HERICLES	LEIRT
21900252	ARMANDO	LEIRT

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Amazon AWS Management Console - DynamoDB

Search for services, features, marketplace products, and docs [Alt+S]

Services: HES_aws Ireland Support

DynamoDB

- Dashboard
- Tables
 - Backups
 - Reserved capacity
 - Exports to S3
 - PartiQL editor
 - Preferences
- DAX
 - Dashboard
 - Clusters
 - Subnet groups
 - Parameter groups
 - Events
- Try the preview of the new console

Disciplinas

Create table Delete table

Filter by table name

Choose a table ... Actions

Name

- Alunos
- Disciplinas
- Notas

Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Contributor Insights Triggers Access control Tags

Create item Actions

Scan: [Table] Disciplinas: disciplinaID Viewing 1 to 3 items

Scan [Table] Disciplinas: disciplinaID

Add filter

Start search

disciplinaID	nome
14203	TELECOMUNICACOES I
16930	COMPUTACAO NA NUVEM
22514	SI NUVEM

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

aws Services Search for services, features, marketplace products, and docs [Alt+S]

DynamoDB Dashboard

Tables

Backups

Reserved capacity

Exports to S3

PartiQL editor

Preferences

DAX

Dashboard

Clusters

Subnet groups

Parameter groups

Events

Try the preview of the new console

Create table Delete table

Filter by table name

Choose a table ...

Actions

Name

Alunos

Disciplinas

Notas

Notas Close

Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Contributor Insights Triggers Access control Tags

Create item Actions

Scan: [Table] Notas: notalD

Viewing 1 to 3 items

Scan [Table] Notas: notalD

Add filter

Start search

notalID	nota
21901188_16930	20
21900000_14203	20
21900252_16930	20

Feedback English (US)

© 2008 - 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences