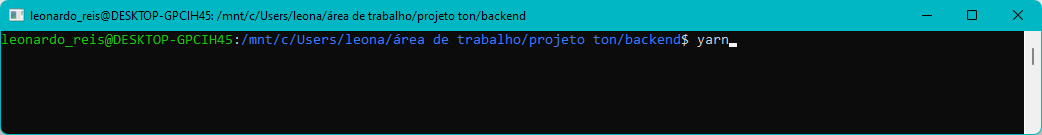
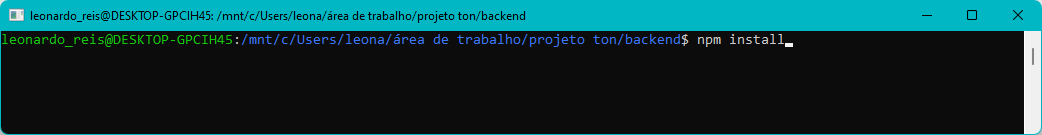
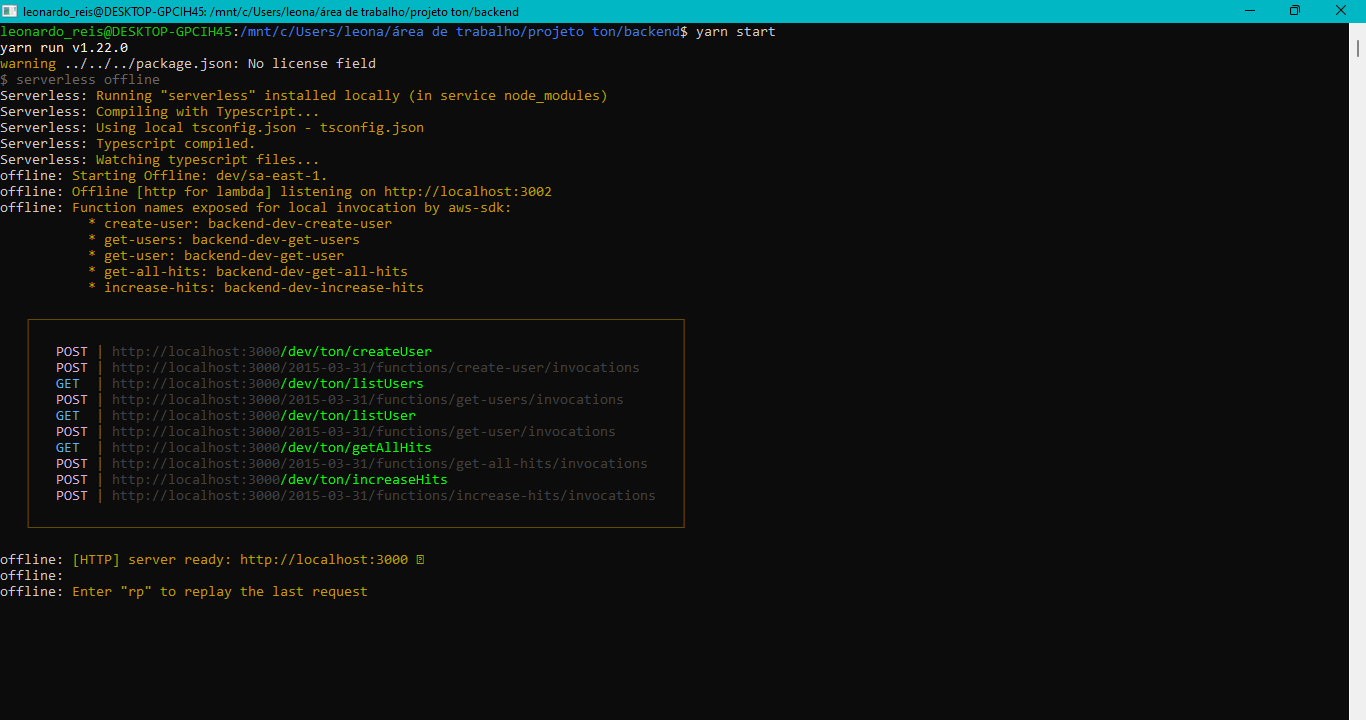
**DOCS – PROJETO – TON**

**EXECUTANDO O BACKEND:**

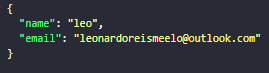
Siga os seguintes passos para a instalação da aplicação backend

1. Execute **yarn** ou **npm install** no seu terminal.
2. Execute **yarn start** ou **npm run start** no seu terminal, após o termino de instalação das dependências. (A aplicação será executada em <http://localhost:3000>).

**ROTAS:**

1. **CreateUser**
   1. Requisição –

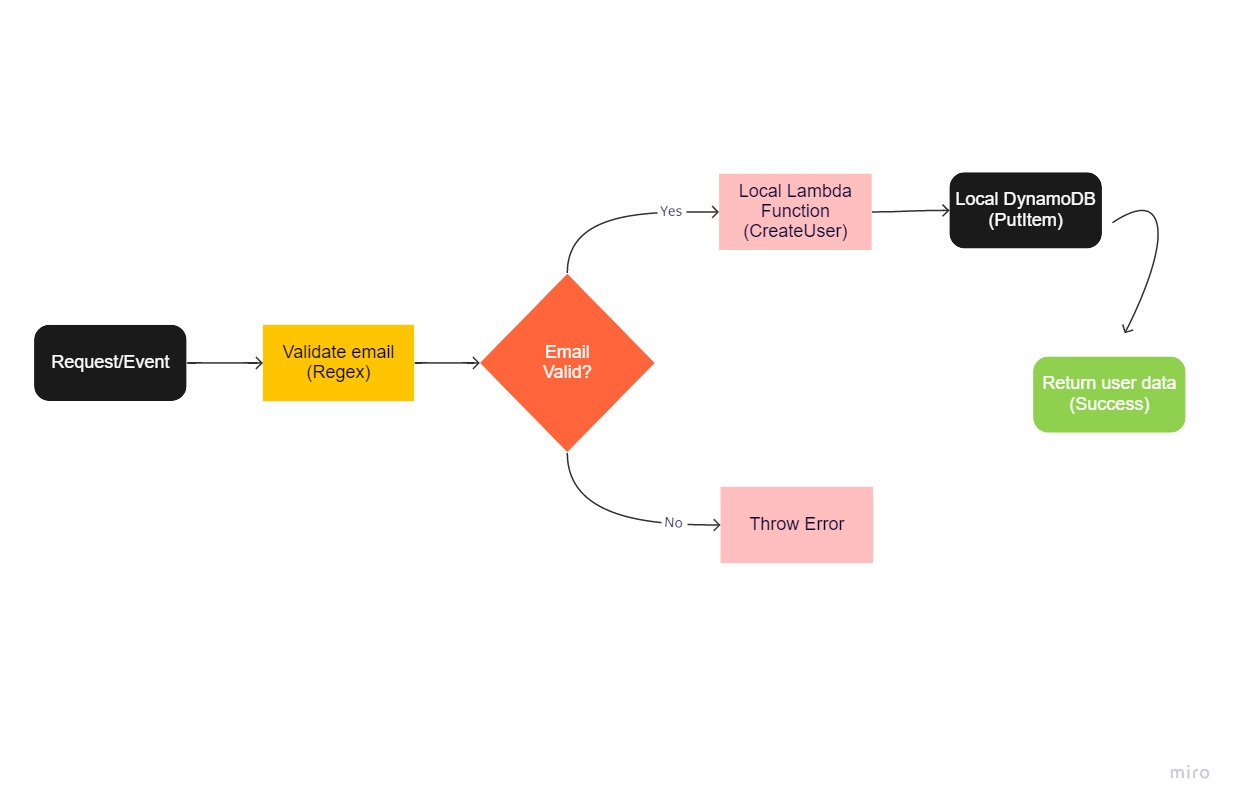
Passe os dados { name, email }, como corpo da requisição JSON



* 1. Resposta



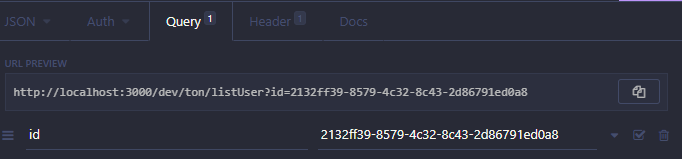
1.3 Fluxograma da rota –



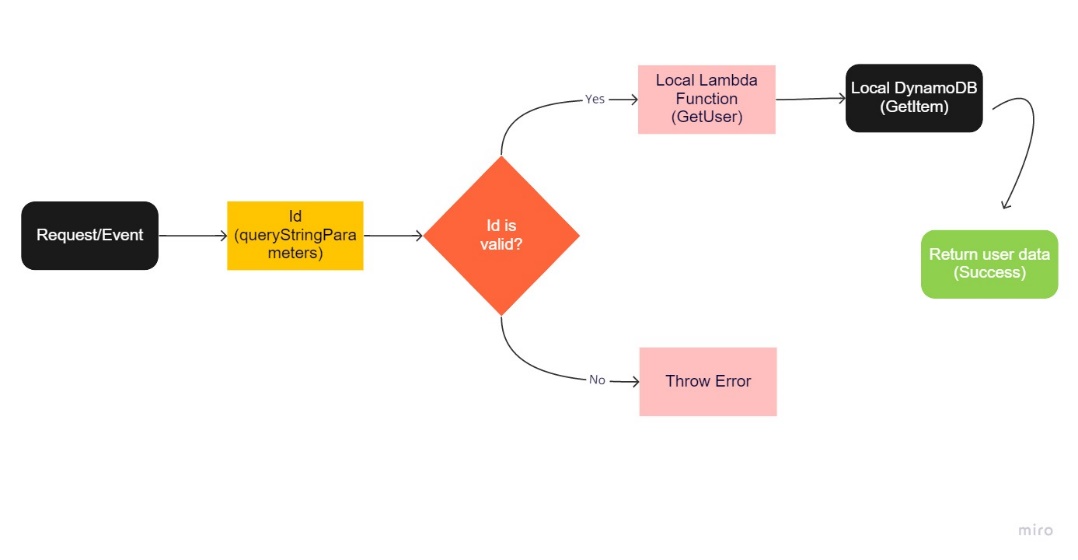
1. **ListUser**
   1. Requisição –

Passe o id do usuário desejado, como **Query Param** na url da requisição

Formato: [http://localhost:3000/dev/ton/listUser?id=<aqui](http://localhost:3000/dev/ton/listUser?id=%3caqui)vocedigitaoid>



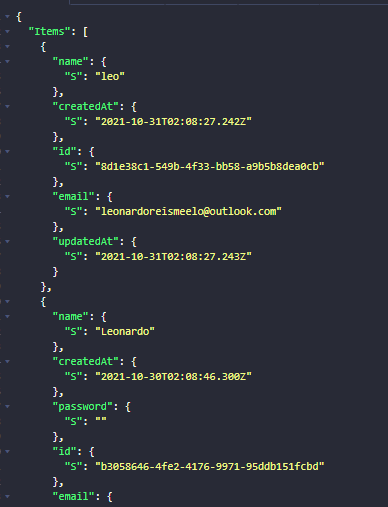
* 1. Resposta
  2. Fluxograma da rota –

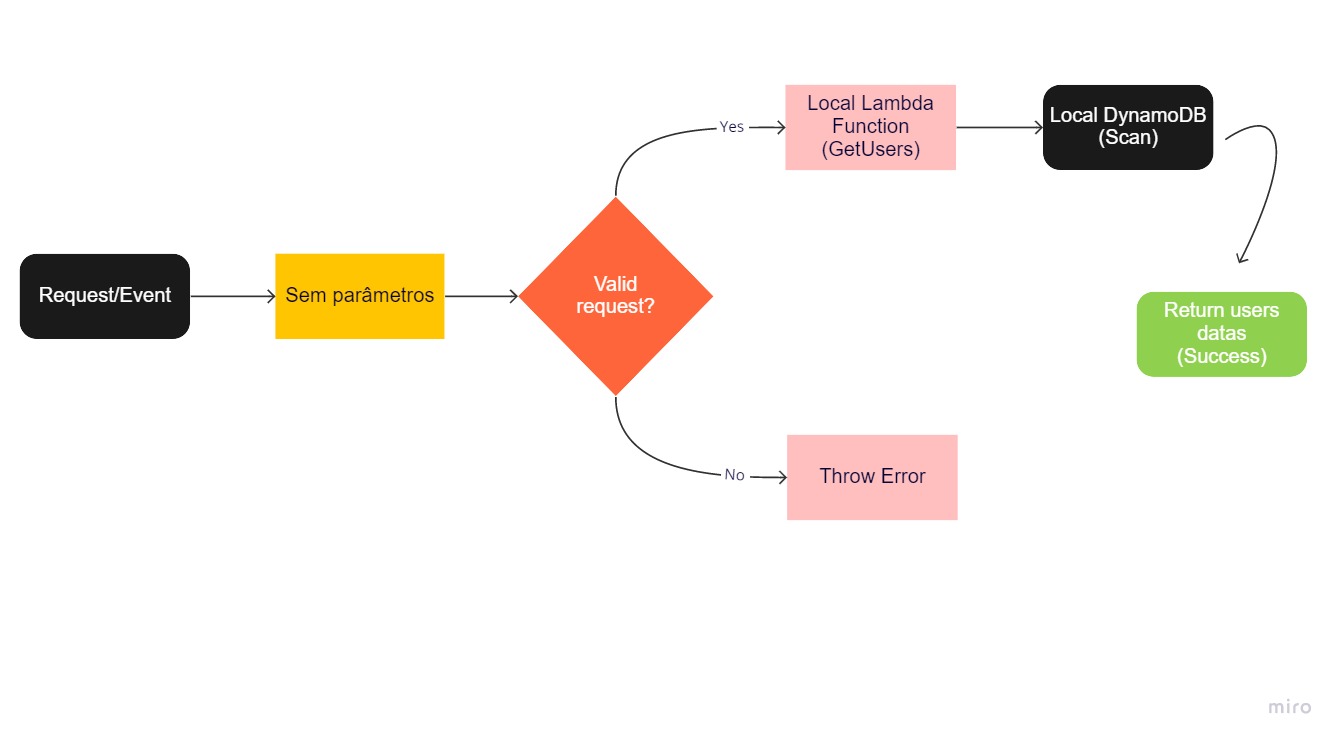


1. **ListUsers**
   1. Requisição –

Sem parâmetros necessários nessa requisição

* 1. Resposta –



* 1. Fluxograma da rota –

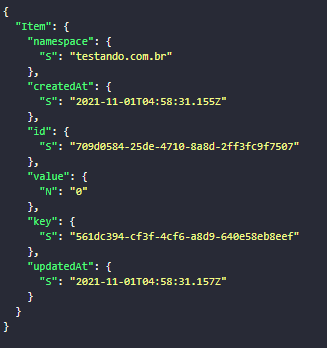
1. **CreateKey**
   1. Requisição –

Passe o dado { **namespace**(url da página) }, como corpo da requisição JSON

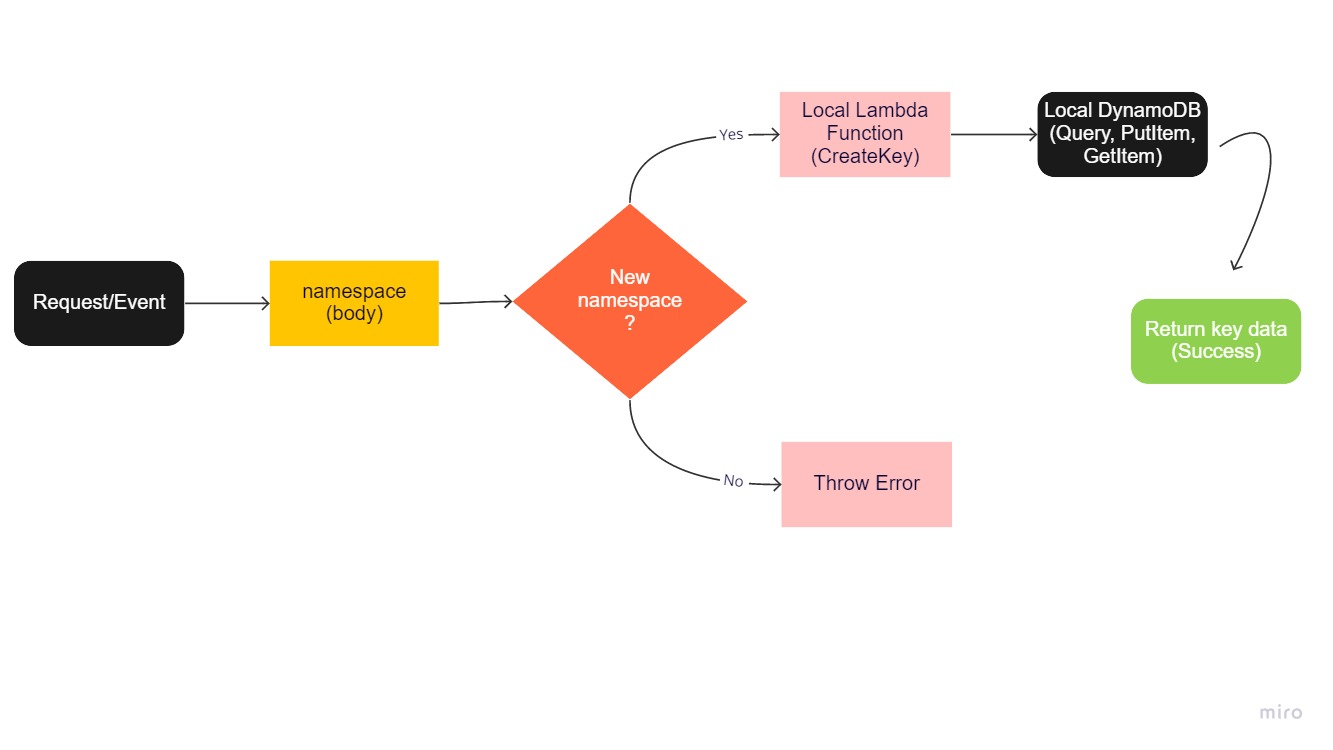
OBS: No campo namespace, não pode conter caracteres especiais, ex: / ou :, etc.



* 1. Resposta –



* 1. Fluxograma da rota –



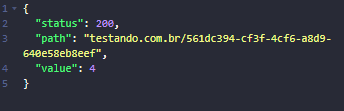
1. **IncreaseHits**
   1. Requisição –

Passe o dado { namespace(Url da página) }, como corpo da requisição JSON

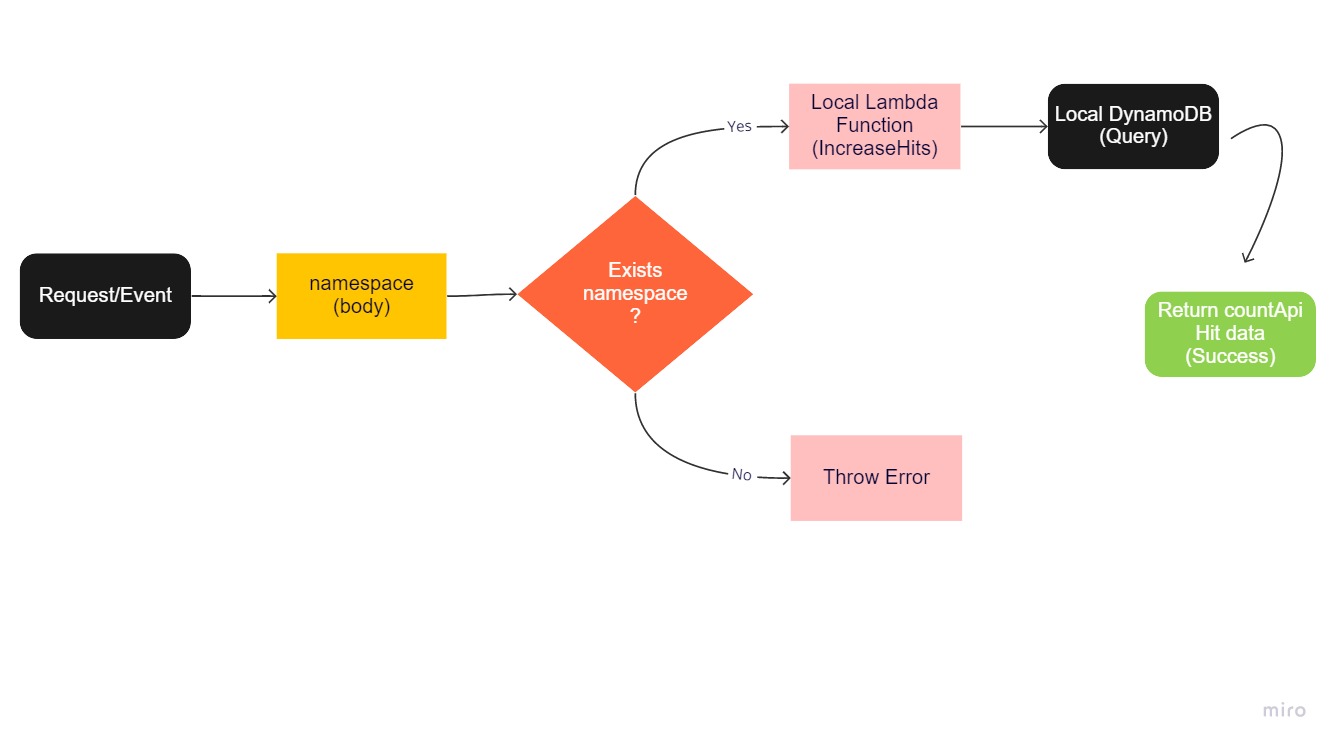
OBS: No campo namespace, não pode conter caracteres especiais, ex: / ou :, etc



* 1. Resposta –



* 1. Fluxograma da rota –

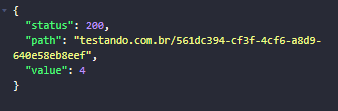


1. **GetAllHits**
   1. Requisição –

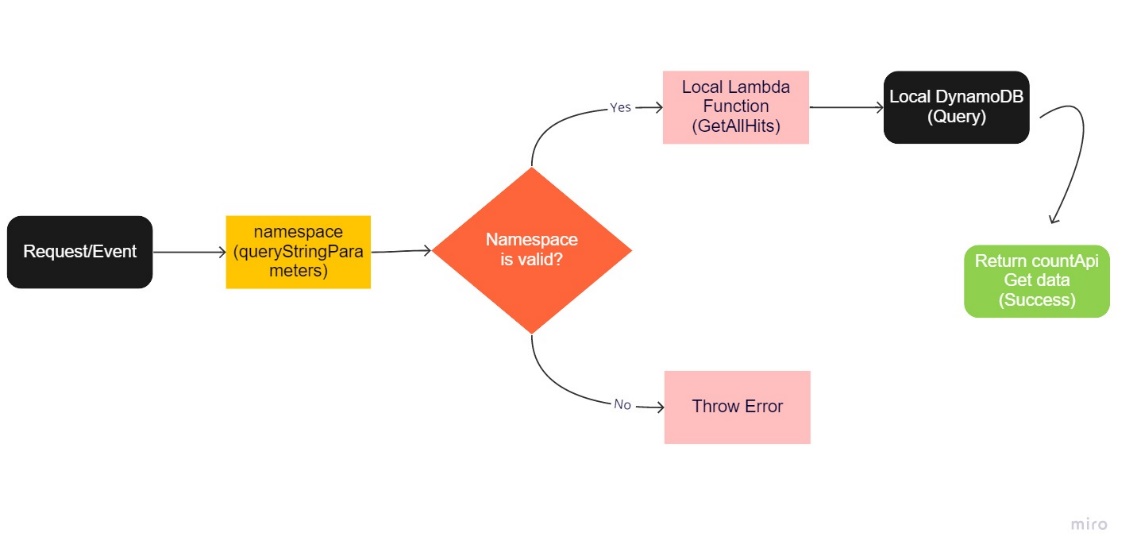
Passe o id do usuário desejado, como **Query Param** na url da requisição

Formato: [http://localhost:3000/dev/ton/GetAllHits?namespace=<aquivocedigitaoid](http://localhost:3000/dev/ton/GetAllHits?namespace=%3caquivocedigitaoid)>

* 1. Resposta –



* 1. Fluxograma da rota –



**EXPLICANDO A SOLUÇÃO DESENVOLVIDA:**

A solução foi desenvolvida, utilizando como referências, a documentação do serverless offline, da aws(dynamodb) e da countAPI(npm).

Ela foi pensada de forma que um usuário pudesse ser cadastrado localmente de maneira fácil e simplificada, além de que também possui duas funções, sendo elas:

ListUser – sendo utilizada para listar os dados de um usuário especifico, baseado no seu id

ListUsers – sendo utilizada para lista os dados de todos os usuários cadastrados no banco de dados

Passando para a parte de utilização da countAPI, foram criadas 3 rotas, sendo elas:

CreateKey – para a criação de uma chave única para cada url desejada

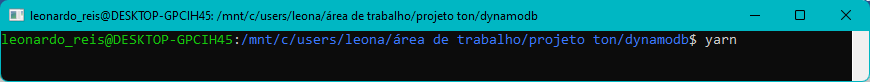
IncreaseHits – para incrementar a quantidade de acessos/clicks naquela url especifica, utilizando como parâmetros a própria url e a sua chave única utilizada na rota anterior

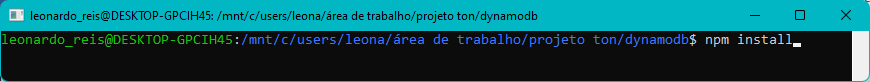
GetAllHits – para buscarmos os acessos/clicks de uma página especifica, sem a necessidade de passarmos a sua chave, somente a sua url, trazendo assim a quantidade total de acessos/clicks até o momento na página indicada

Essa solução foi pensada, de forma que cada url pudesse tem uma contagem individual dos seus acessos, sendo identificada por uma chave de acesso única, evitando assim incoerência de dados.

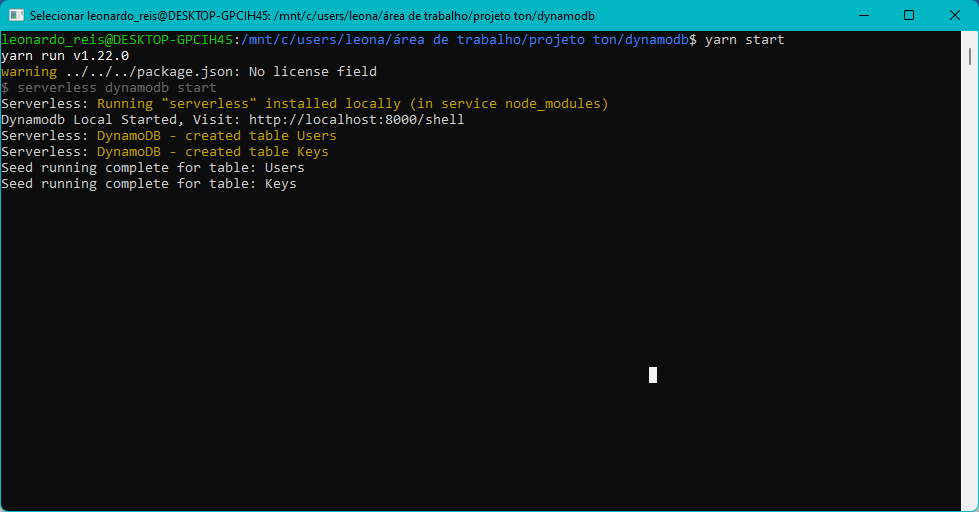
**EXECUTANDO O DYNAMODB:**

Siga os seguintes passos para a instalação do banco de dados dynamodb

1. Execute **yarn** ou **npm install** no seu terminal.



1. Execute **yarn start** ou **npm run start** no seu terminal, após o termino de instalação das dependências. (O banco de dados será executado em <http://localhost:8000>).



**TABELAS:**

1. **Users**

A tabela usuário é composta pelos campos:

Id: string(uuid())

name: string

email: string

createdAt: string(moment().toString())

updatedAt: string(moment().toString())

Sendo esses dados previamente alimentados, por um arquivo seed.json, que já contém algumas informações para popular a tabela.

1. **Keys**

A tabele chaves é composta pelo campos:

Id: string(uuid())

namespace: string()

key: string(uuid())

value: number(int)

createdAt: string(moment().toString())

updatedAt: string(moment().toString())

Sendo esses dados previamente alimentados, por um arquivo seed.json, que já contém algumas informações para popular a tabela.

**ESTRUTURA DA APLICAÇÃO BACKEND**

Nos arquivos do dynamodb temos a seguinte estrutura formada por:

**node\_modules**(dependências utilizadas)

**src**(pasta raiz com os arquivos utilizados)

**access**(pasta contendo a pasta functions)

**functions(access)**(pasta contendo as funções relacionadas a parte de acessos/clicks ao site(countApi))

**user**(pasta contendo a pasta functions)

**functions(user)**(pasta contendo as funções relacionadas a parte de usuários)

**.eslintignore**(arquivo contendo informações de quais locais o eslint deve ignorar)

**.eslintrc.js**(arquivo com as configurações do eslint)

**.gitignore**(arquivo com os locais, que não devem ser subidos junto com o commit do git)

**local.dynamodb.json**(arquivo com o endpoint do dynamodb local)

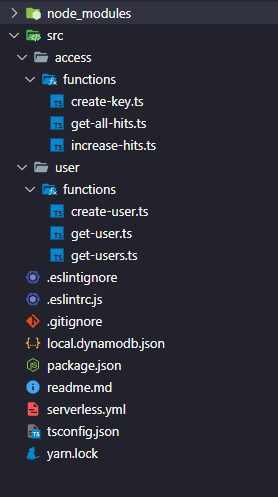
**package.json**(arquivo principal do projeto, com os scripts)

**readme.md**(arquivo com instruções para a instalação do dynamodb local)

**serverless.yml**(arquivo para a execução do dynamodb local)

**tsconfig.json**(arquivo com as configurações para executar a aplicação, utilizando typescript)

**yarn.lock**(arquivo com as informações da dependências, instaladas com yarn)



**ESTRUTURA DO BANCO DE DADOS – DYNAMODB**

Nos arquivos do dynamodb temos a seguinte estrutura formada por:

**.dynamodb**(arquivos do dynamod local)

**node\_modules**(dependências utilizadas)

**src**(pasta raiz com os arquivos utilizados)

**Keys**(pasta contendo a seed utilizada para popular a tabela Keys)

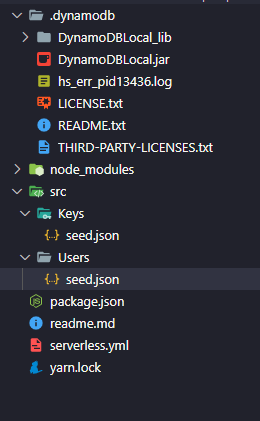
**Users**(pasta contendo a seed utilizada para popular a tabela Users)

**package.json**(arquivo principal do projeto, com os scripts)

**readme.md**(arquivo com instruções para a instalação do dynamodb local)

**serverless.yml**(arquivo para a execução do dynamodb local)

**yarn.lock**(arquivo com as informações da dependências, instaladas com yarn)



**TECNOLOGIAS UTILIZADAS**

<https://www.serverless.com/>

<https://eslint.org/>

<https://nodejs.org/en/>

<https://git-scm.com/>

<https://www.typescriptlang.org/>

<https://www.json.org/json-en.html>

<https://yarnpkg.com/>

<https://www.npmjs.com/>

<https://aws.amazon.com/pt/dynamodb/>

**DEPENDÊNCIAS UTILIZADAS**

<https://www.npmjs.com/package/uuid>

<https://www.npmjs.com/package/serverless-offline>

<https://www.npmjs.com/package/serverless-plugin-typescript>

<https://www.npmjs.com/package/typescript>

<https://www.npmjs.com/package/aws-sdk>

<https://www.npmjs.com/package/countapi-js>

<https://www.npmjs.com/package/dotenv>

<https://www.npmjs.com/package/eslint>

<https://www.npmjs.com/package/moment>

<https://www.npmjs.com/package/serverless-dynamodb-local>