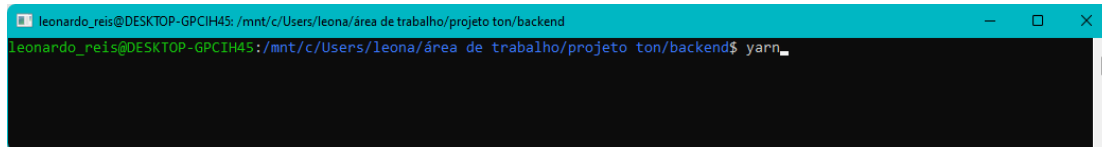


DOCS – PROJETO – TON

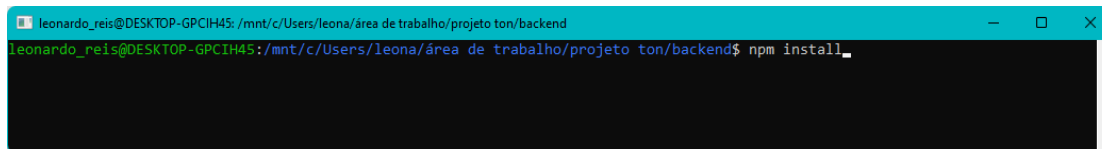
EXECUTANDO O BACKEND:

Siga os seguintes passos para a instalação da aplicação backend

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

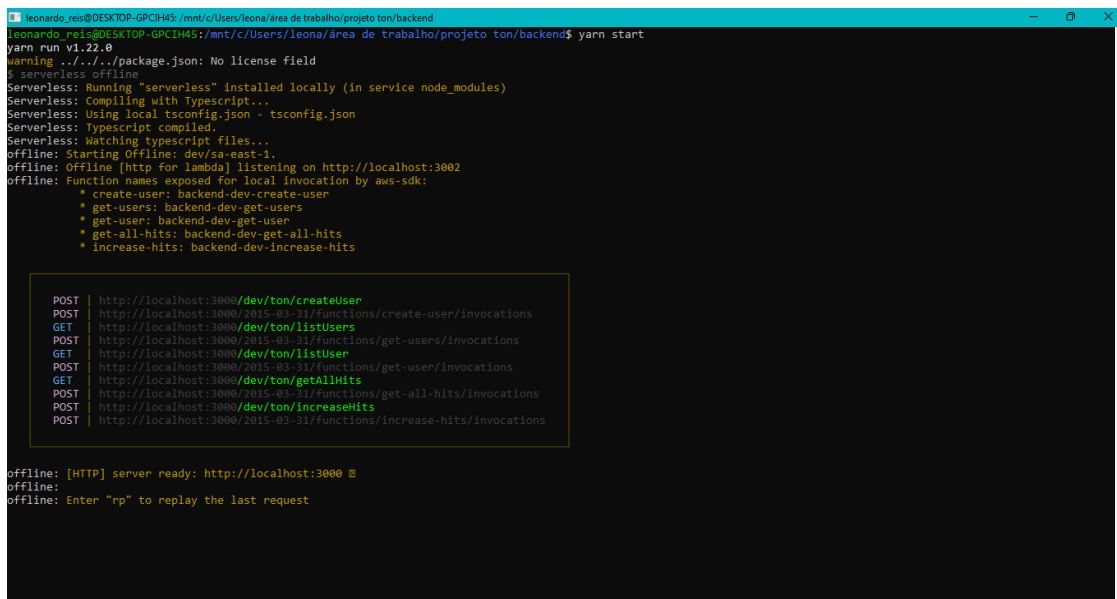
A terminal window with a blue title bar. The prompt is 'leonardo_reis@DESKTOP-GPCIH45: /mnt/c/Users/leona/área de trabalho/projeto ton/backend'. The command 'yarn' has been entered and the cursor is at the end of the line.

```
leonardo_reis@DESKTOP-GPCIH45: /mnt/c/Users/leona/área de trabalho/projeto ton/backend$ yarn
```

A terminal window with a blue title bar. The prompt is 'leonardo_reis@DESKTOP-GPCIH45: /mnt/c/Users/leona/área de trabalho/projeto ton/backend'. The command 'npm install' has been entered and the cursor is at the end of the line.

```
leonardo_reis@DESKTOP-GPCIH45: /mnt/c/Users/leona/área de trabalho/projeto ton/backend$ npm install
```

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>).

A terminal window with a blue title bar. The prompt is 'leonardo_reis@DESKTOP-GPCIH45: /mnt/c/Users/leona/área de trabalho/projeto ton/backend'. The command 'yarn start' has been entered. The output shows serverless offline mode, local lambda listening on http://localhost:3002, and a list of exposed function names. A yellow box highlights a list of HTTP requests. The terminal ends with 'server ready: http://localhost:3000' and a prompt to enter 'rp' to replay the last request.

```
leonardo_reis@DESKTOP-GPCIH45: /mnt/c/Users/leona/área de trabalho/projeto ton/backend$ yarn start
yarn run v1.22.0
warning ../../package.json: No license field
$ serverless offline
Serverless: Running "serverless" installed locally (in service node_modules)
Serverless: Compiling with Typescript...
Serverless: Using local tsconfig.json - tsconfig.json
Serverless: Typescript compiled.
Serverless: Watching typescript files...
offline: Starting Offline: dev/sa-east-1
offline: Offline [http for lambda] listening on http://localhost:3002
offline: Function names exposed for local invocation by aws-sdk:
  * create-user: backend-dev-create-user
  * get-users: backend-dev-get-users
  * get-user: backend-dev-get-user
  * get-all-hits: backend-dev-get-all-hits
  * increase-hits: backend-dev-increase-hits

POST http://localhost:3000/dev/ton/createUser
POST http://localhost:3000/2015-03-31/functions/create-user/invocations
GET http://localhost:3000/dev/ton/listUsers
POST http://localhost:3000/2015-03-31/functions/get-users/invocations
GET http://localhost:3000/dev/ton/listUser
POST http://localhost:3000/2015-03-31/functions/get-user/invocations
GET http://localhost:3000/dev/ton/getAllHits
POST http://localhost:3000/2015-03-31/functions/get-all-hits/invocations
POST http://localhost:3000/dev/ton/increaseHits
POST http://localhost:3000/2015-03-31/functions/increase-hits/invocations

offline: [HTTP] server ready: http://localhost:3000
offline:
offline: Enter "rp" to replay the last request
```

ROTAS:

1. CreateUser

1.1 Requisição –

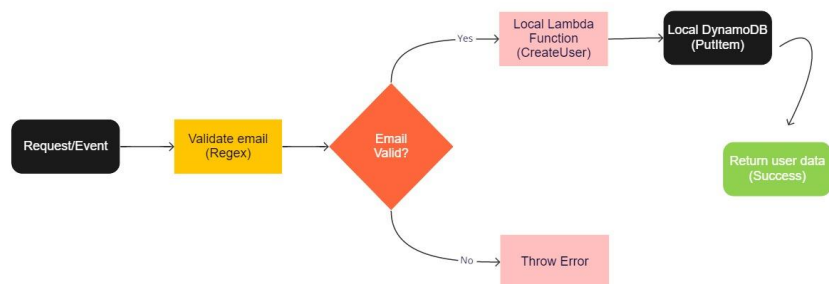
Passa os dados { name, email }, como corpo da requisição JSON

```
{
  "name": "leo",
  "email": "leonardoreismeelo@outlook.com"
}
```

1.2 Resposta

```
{
  "Item": {
    "name": {
      "S": "leo"
    },
    "createdAt": {
      "S": "2021-10-31T02:08:27.242Z"
    },
    "id": {
      "S": "8d1e38c1-549b-4f33-bb58-a9b5b8dea0cb"
    },
    "email": {
      "S": "leonardoreismeelo@outlook.com"
    },
    "updatedAt": {
      "S": "2021-10-31T02:08:27.243Z"
    }
  }
}
```

1.3 Fluxograma da rota –



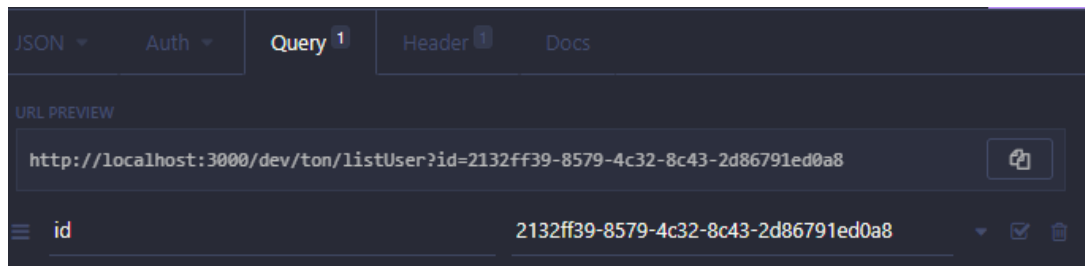
2. ListUser

2.1 Requisição –

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

Formato:

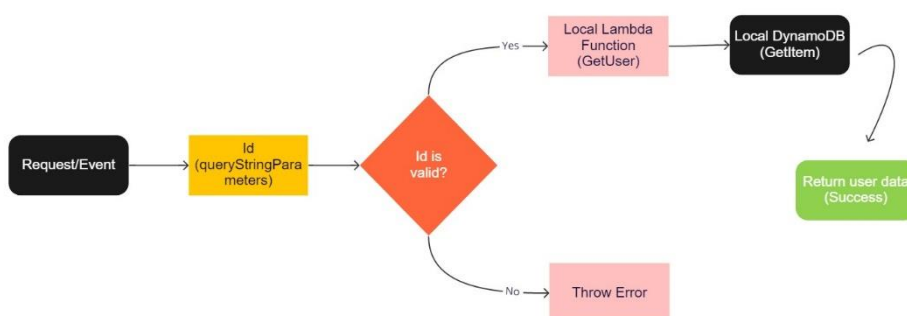
http://localhost:3000/dev/ton/listUser?id=<aquivocedigitalid>



2.2 Resposta

```
{
  "Item": {
    "name": {
      "S": "leo"
    },
    "createdAt": {
      "S": "2021-10-31T01:57:36.087Z"
    },
    "id": {
      "S": "2132ff39-8579-4c32-8c43-2d86791ed0a8"
    },
    "email": {
      "S": "leonardoreismeelo@outlook.com"
    },
    "updatedAt": {
      "S": "2021-10-31T01:57:36.088Z"
    }
  }
}
```

2.3 Fluxograma da rota –



3. ListUsers

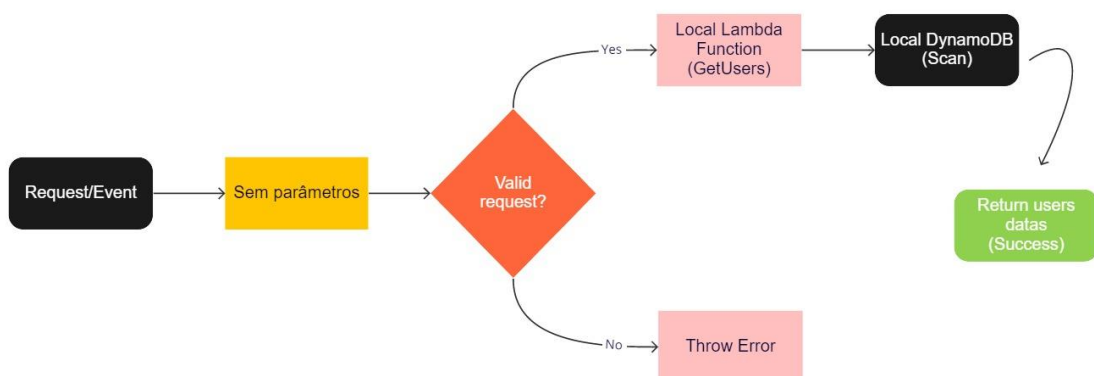
3.1 Requisição –

Sem parâmetros necessários nessa requisição

3.2 Resposta –

```
{
  "Items": [
    {
      "name": {
        "s": "leo"
      },
      "createdAt": {
        "s": "2021-10-31T02:08:27.242Z"
      },
      "id": {
        "s": "8d1e38c1-549b-4f33-bb58-a9b5b8dea0cb"
      },
      "email": {
        "s": "leonardoreismeelo@outlook.com"
      },
      "updatedAt": {
        "s": "2021-10-31T02:08:27.243Z"
      }
    },
    {
      "name": {
        "s": "Leonardo"
      },
      "createdAt": {
        "s": "2021-10-30T02:08:46.300Z"
      },
      "password": {
        "s": ""
      },
      "id": {
        "s": "b3058646-4fe2-4176-9971-95ddb151fcbd"
      },
      "email": {
```

3.3 Fluxograma da rota –



4. CreateKey

4.1 Requisição –

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

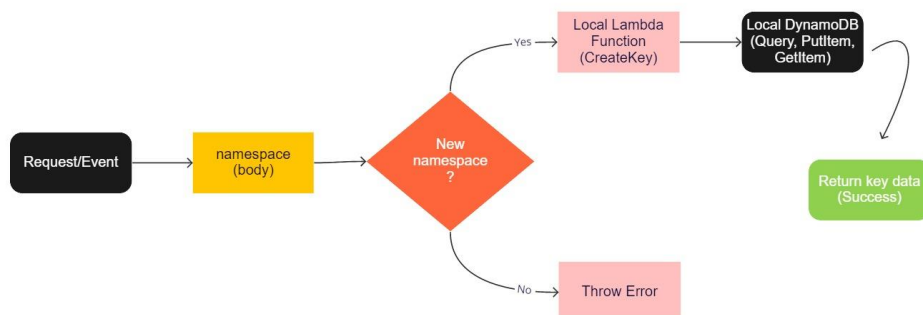
OBS: No campo namespace, não pode conter caracteres especiais, ex: / ou :, etc.

```
{  
  "namespace": "testando.com.br"  
}
```

4.2 Resposta –

```
{  
  "Item": {  
    "namespace": {  
      "S": "testando.com.br"  
    },  
    "createdAt": {  
      "S": "2021-11-01T04:58:31.155Z"  
    },  
    "id": {  
      "S": "709d0584-25de-4710-8a8d-2ff3fc9f7507"  
    },  
    "value": {  
      "N": "0"  
    },  
    "key": {  
      "S": "561dc394-cf3f-4cf6-a8d9-640e58eb8eef"  
    },  
    "updatedAt": {  
      "S": "2021-11-01T04:58:31.157Z"  
    }  
  }  
}
```

4.3 Fluxograma da rota –



5. IncreaseHits

5.1 Requisição –

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

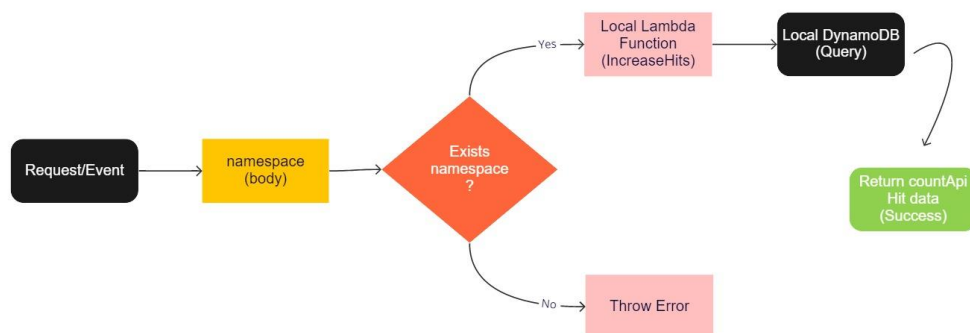
OBS: No campo namespace, não pode conter caracteres especiais, ex: / ou :, etc

```
{  
  "namespace": "testando.com.br"  
}
```

5.2 Resposta –

```
1 {  
2   "status": 200,  
3   "path": "testando.com.br/561dc394-cf3f-4cf6-a8d9-  
4     640e58eb8eef",  
5   "value": 4  
}
```

5.3 Fluxograma da rota –



6. GetAllHits

6.1 Requisição –

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

Formato:

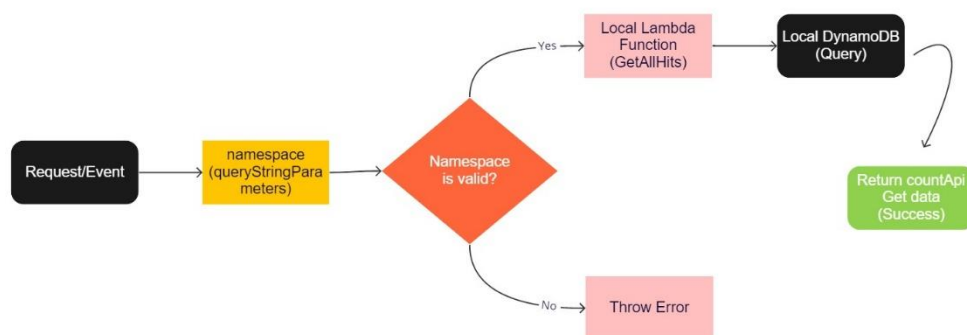
http://localhost:3000/dev/ton/GetAllHits?namespace=<quivocedigitalid>



6.2 Resposta –

```
{
  "status": 200,
  "path": "testando.com.br/561dc394-cf3f-4cf6-a8d9-640e58eb8eef",
  "value": 4
}
```

6.3 Fluxograma da rota –



miro

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 específico, 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 específica, utilizando como parâmetros a própria url e a sua chave única utilizada na rota anterior

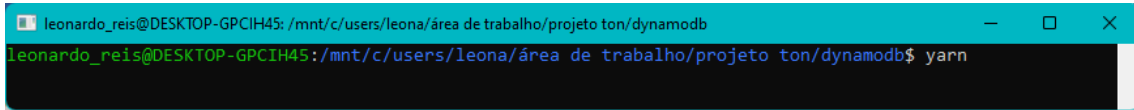
GetAllHits – para buscarmos os acessos/clicks de uma página específica, sem a necessidade de passarmos a sua chave, somente a sua url, trazendo assim a quantidade total de acessos/clicks até o momento na página indicada

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

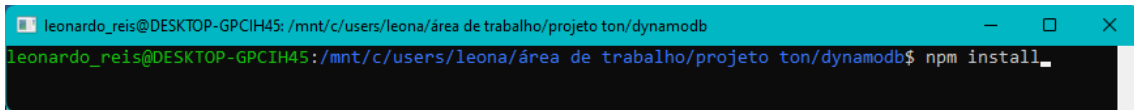
EXECUTANDO O DYNAMODB:

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

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

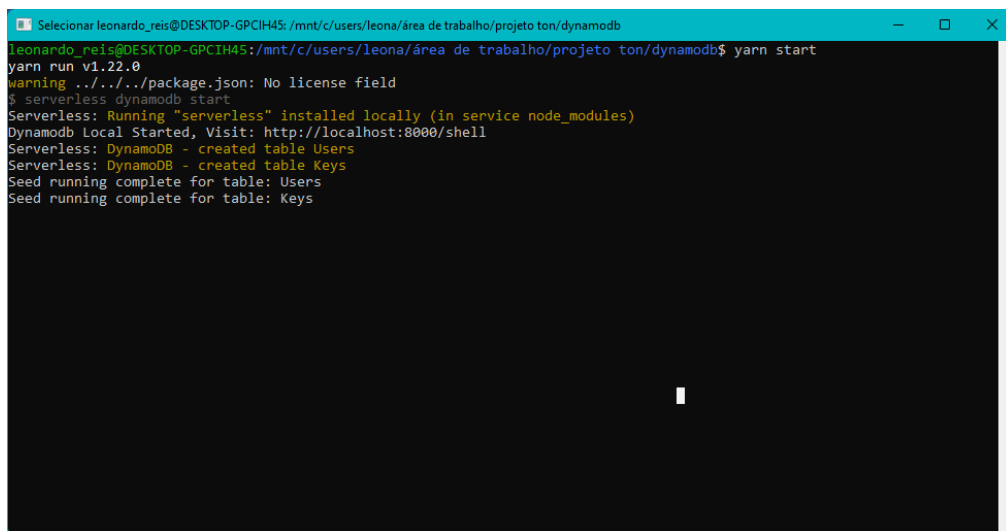


```
leonardo_reis@DESKTOP-GPCIH45: /mnt/c/users/leona/área de trabalho/projeto ton/dynamodb$ yarn
```



```
leonardo_reis@DESKTOP-GPCIH45: /mnt/c/users/leona/área de trabalho/projeto ton/dynamodb$ npm install
```

3. 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>).



```
Selecione leonardo_reis@DESKTOP-GPCIH45: /mnt/c/users/leona/área de trabalho/projeto ton/dynamodb$ yarn start
yarn run v1.22.0
warning ../../package.json: No license field
$ serverless dynamodb start
Serverless: Running "serverless" installed locally (in service node_modules)
Dynamodb Local Started, Visit: http://localhost:8000/shell
Serverless: DynamoDB - created table Users
Serverless: DynamoDB - created table Keys
Seed running complete for table: Users
Seed running complete for table: Keys
```

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.

2. Keys

A tabela 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)

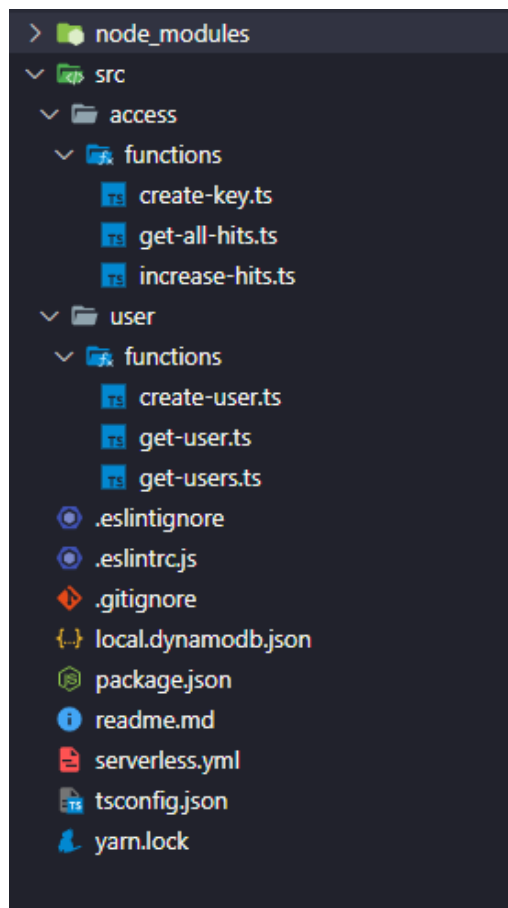
package.json(arquivo principal do projeto, com os scripts)

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)

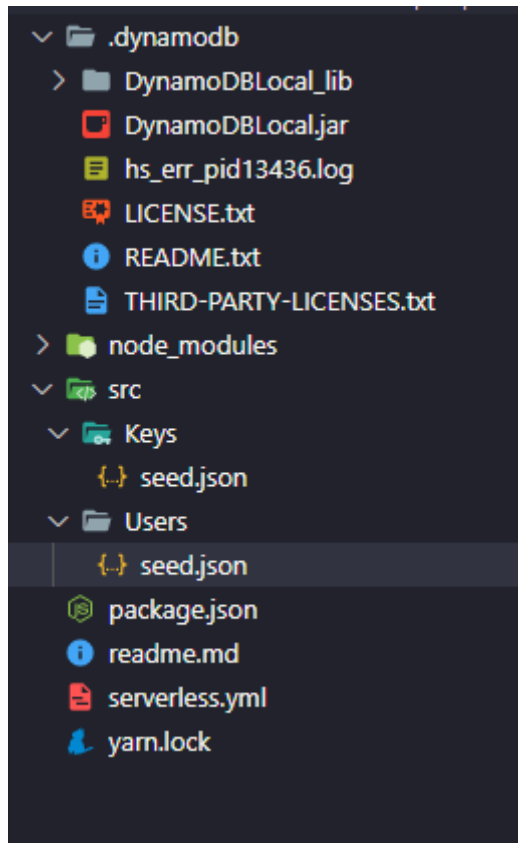
Users(pasta contendo a seed utilizada para popular a tabela Users)

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>