CLOUD COMPUTING

FINAL PROJECT

Integrantes:
Luis Méndez
Mauricio Álvarez
Jeffry Quintana

CONTENIDO

CASO DE USO

Smart Album es una aplicación serverless Multi-tenancy con Arquitectura basada en eventos capaz de automáticamente organizar fotografías en base a objetos, escenas y personas. Este utiliza Amazon Rekognition para etiquetar y categorizar las imágenes subidas en un bucket de Amazon S3 y guardar la metadata en DynamoDB cuando el usuario lo requiera.

SERVICIOS UTILIZADOS

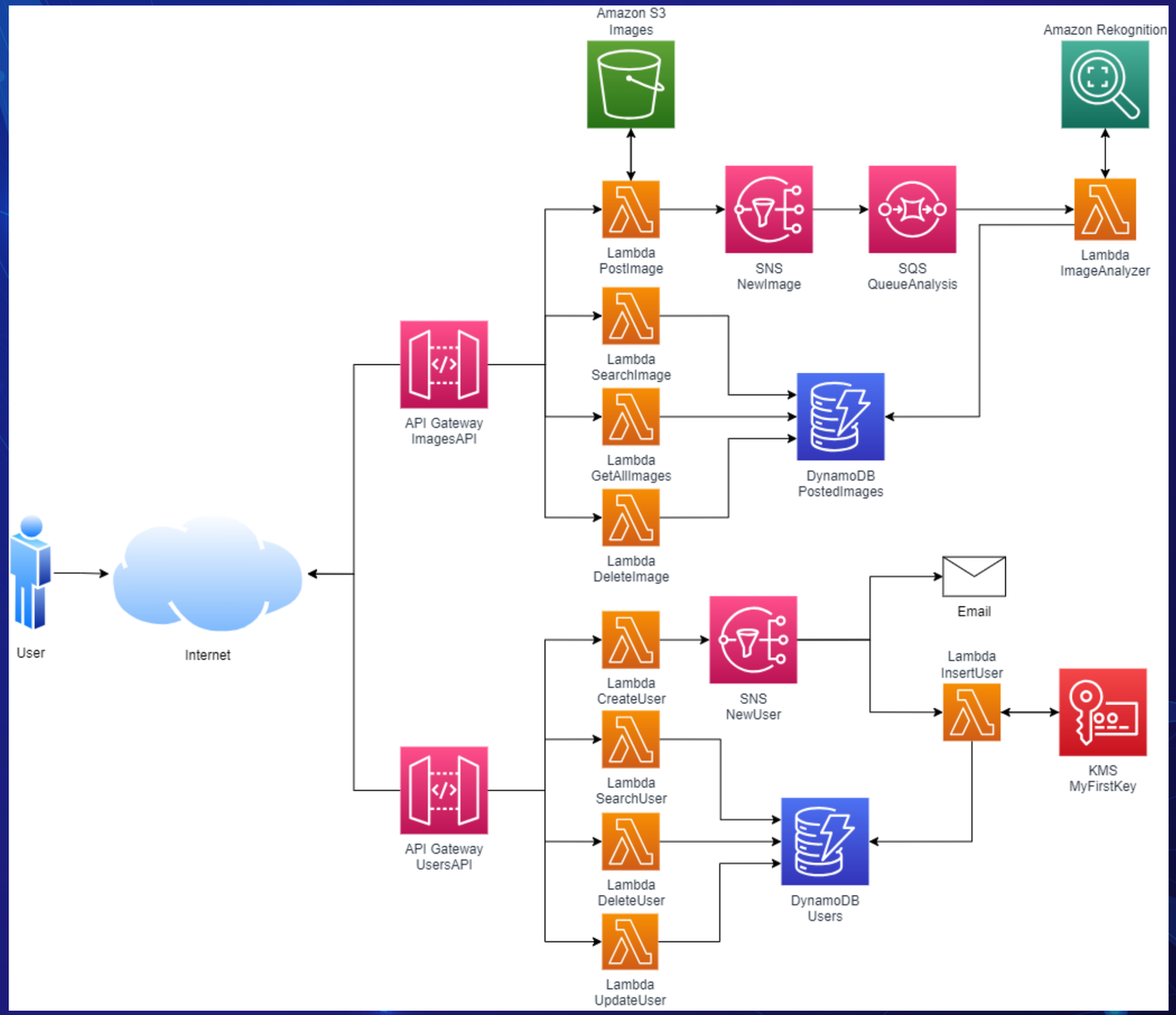
01	02	03	04	05
DEVELOPMENT	STORAGE	DATABASE	MESSAGING	AI AND SECURITY
AWS Lambda Amazon API Gateway	Amazon S3	Amazon DynamoDB	Amazon SQS Amazon SNS	Amazon Rekognition KMS



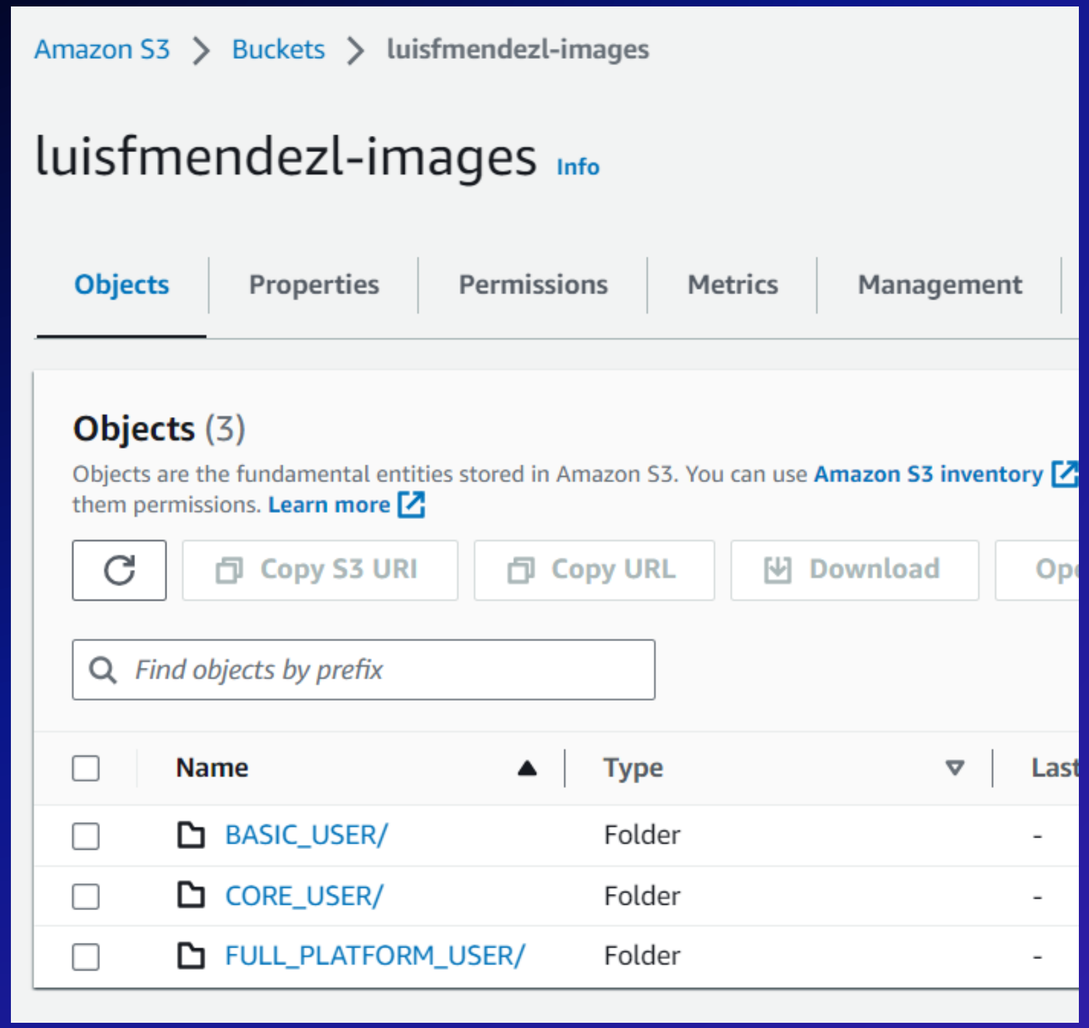


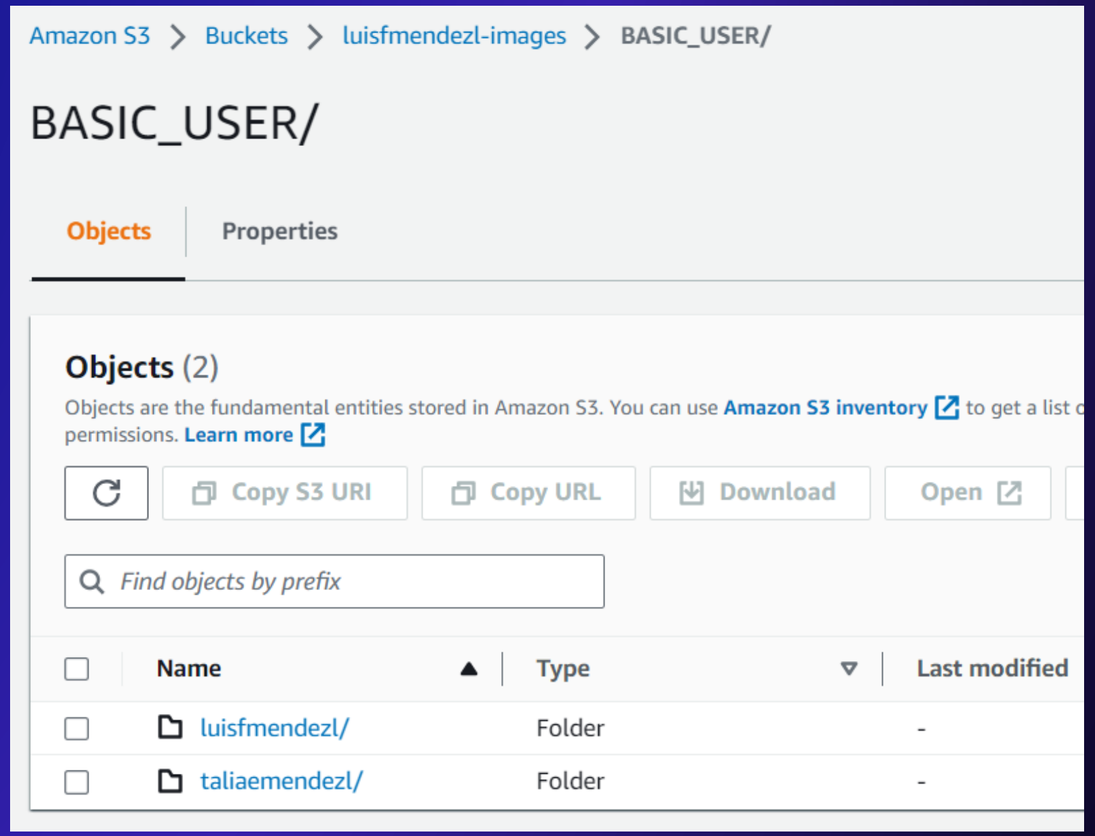



DIAGRAMA DE SOLUCIÓN "SMART ALBUM" SERVERLESS APP



AMAZON S3 BUCKET





TABLAS DYNAMODB

Users

Autopreview | View table details

Scan or query items
Expand to query or scan items.

Items returned (1) | Actions | Create item

1

user_type	user_name	user_data
BASIC_USER	luisfmendezl	{ "last_name" : { "S" : "Méndez" }, "password" : { "S" : "AQICAHiNiqbnCDGgOK+DP…

PostedImages

Autopreview | View table details

Scan or query items
Expand to query or scan items.

Items returned (3) | Actions | Create item

1

image_user	image_name	image_details	tags
BASIC_USER/luisfmen…	birthday.jpg	{ "posting_date"…	[{ "S" : "People" }, { "S" : "Person" }, { "S" : "Fun" }, { "S" :…
BASIC_USER/luisfmen…	family.jpg	{ "posting_date"…	[{ "S" : "People" }, { "S" : "Person" }, { "S" : "Face" }, { "S"…
BASIC_USER/luisfmen…	messi-CR.jpg	{ "posting_date"…	[{ "S" : "Adult" }, { "S" : "Male" }, { "S" : "Man" }, { "S" : "P…

TABLAS DYNAMODB

Attributes View DynamoDB JSON


```
{
 "user_type": {
  "S": "BASIC_USER"
 },
 "user_name": {
  "S": "luisfmendezl"
 },
 "user_data": {
  "M": {
   "email": {
    "S": "luis.mendez.l@utec.edu.pe"
   },
   "first_name": {
    "S": "Luis"
   },
   "last_name": {
    "S": "Méndez"
   },
   "password": {
    "S": "AQICAHiNiqbnCDGgOK+DPPwWU8f0DjrVj/ac
        /pOU1Sf/ou3Rr29NAgEQgCNBaSIIs5VVEiRRgD
   },
   "registration_date": {
    "S": "2023-07-07"
```

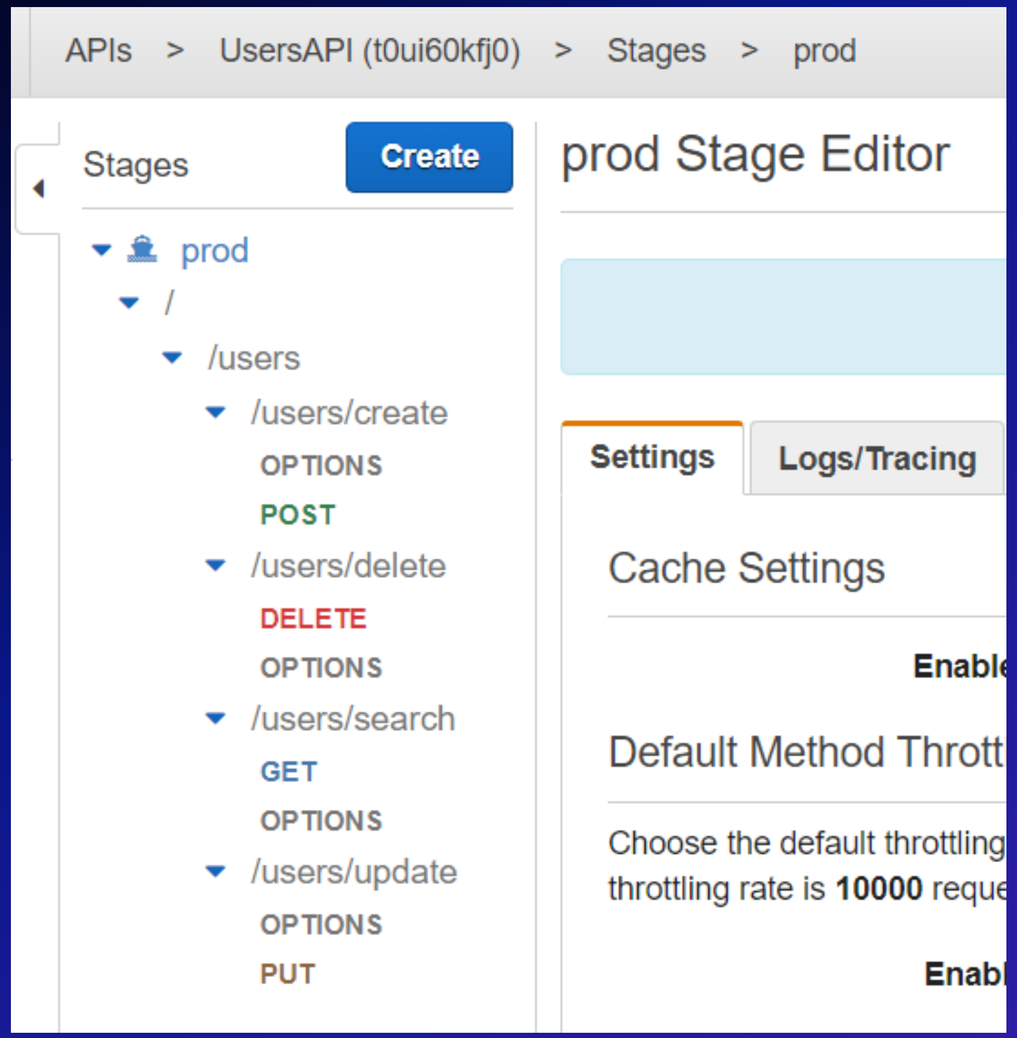

User instance

Attributes View DynamoDB JSON

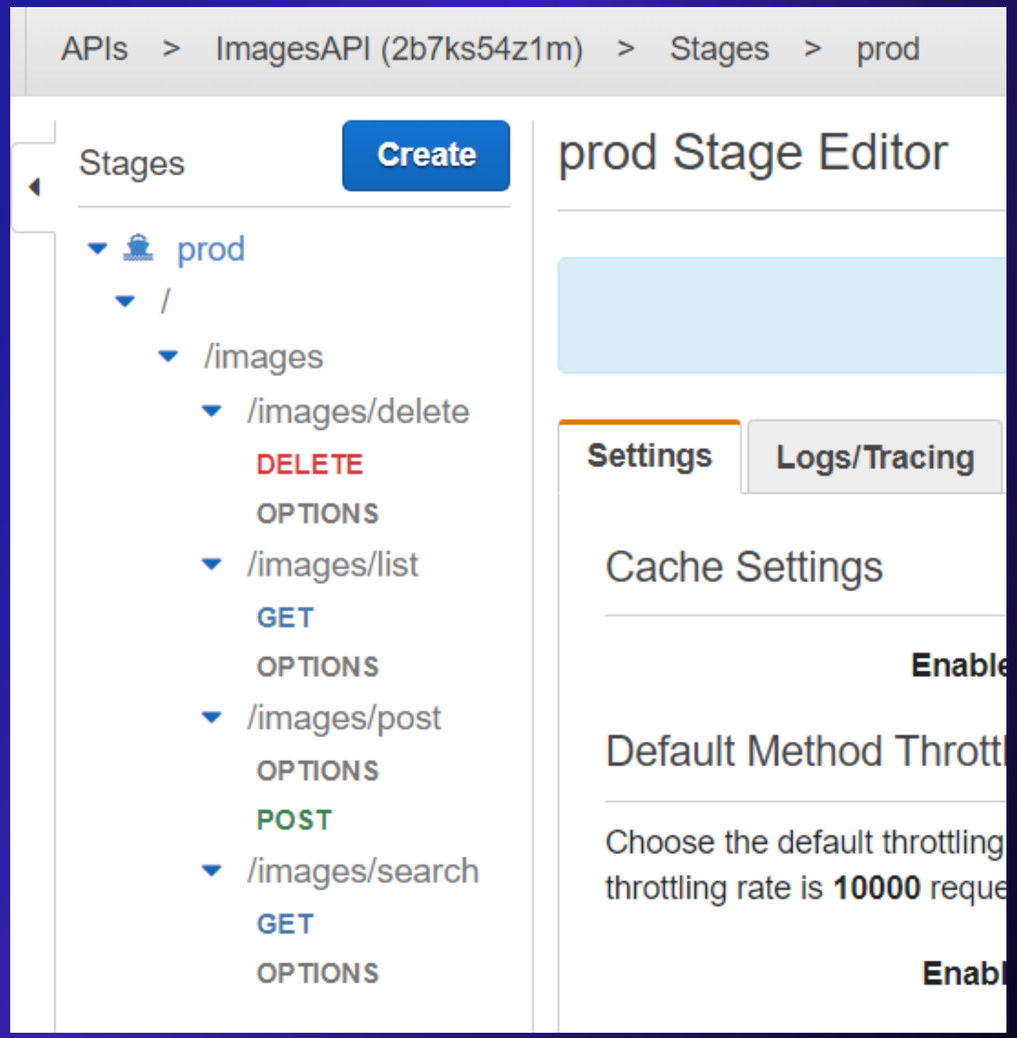

```
{
  "image_user": {
    "S": "BASIC_USER/luisfmendezl"
  },
  "image_name": {
    "S": "messi-CR.jpg"
  },
  "image_details": {
    "M": {
      "posting_date": {
        "S": "2023-07-08"
      },
      "posting_time": {
        "S": "01:55:29.998089"
      }
    }
  },
  "tags": {
    "L": [
      {
        "S": "Adult"
      },
      {
        "S": "Male"
```


Image instance

AMAZOND APIS GATEWAY



UsersAPI



ImagesAPI

LAMBDAS

CreateUser.py

UsersAPI > CreateUser.py > ...

```python
import json
import boto3

def lambda_handler(event, context):

    # JSON processing
    user_type = event['user_type']
    user_name = event['user_name']
    user_data = event['user_data']

    user = {
        'user_type': user_type,
        'user_name': user_name,
        'user_data': user_data
    }

    # S3 bucket creation
    s3_client = boto3.client('s3')
    bucket_name = 'luisfmendezl-images'
    folder_name = f'{user_type}/{user_name}/'

    s3_response = s3_client.put_object(
        Bucket=bucket_name,
        Key=(folder_name)
    )
    print(s3_response)

    # SNS publication
    sns_client = boto3.client('sns')
    sns_response = sns_client.publish(
        TopicArn = 'arn:aws:sns:us-east-1:8968238055
        Subject = 'New user',
        Message = json.dumps(user),
```

CreateUser

InsertUser.py

UsersAPI > InsertUser.py > ...

```python
import json
import boto3
import base64
from datetime import date

def lambda_handler(event, context):

    Message = event["Records"][0]["Sns"]['Message']
    Message = json.loads(Message)

    user_type = Message['user_type']
    user_name = Message['user_name']
    user_data = Message['user_data']

    user_data['password'] = encrypt_password(user_data['password

    user_data['registration_date'] = str(date.today())

    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('t_users')

    user = {
        'user_type': user_type,
        'user_name': user_name,
```

```python
def encrypt_password(password):

    kms_client = boto3.client('kms')
    response = kms_client.encrypt(
        KeyId='fa48f908-27d4-4949-9052-7340cec268fa',
        Plaintext=password
    )
    encrypted_password = response['CiphertextBlob']

    return base64.b64encode(encrypted_password).decode('utf-8')

```

InsertUser

DeleteUser.py

UsersAPI > DeleteUser.py

```python
import boto3

def lambda_handler(event, context):

    user_type = event['user_type']
    user_name = event['user_name']

    # Delete user from Users
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('Users')

    dynamodb_response = table.delete_item(
        Key={
            'user_type': user_type,
            'user_name': user_name
        }
    )

    # Delete folder from the Bucket
    bucket_name = 'luisfmendezl-images'
    prefix = f'{user_type}/{user_name}/'

    s3 = boto3.client('s3')
    s3_response = s3.list_objects_v2(Bucket=bucket_name, Pr

    print(s3_response)

    if 'Contents' in s3_response:
        objects = [{'Key': obj['Key']} for obj in s3_respon
        s3.delete_objects(Bucket=bucket_name, Delete={'Obje

    # Delete their images from PostedImages
    dynamodb = boto3.client('dynamodb')
    image_user = f'{user_type}/{user_name}'

    scan_params = {
        'TableName': 'PostedImages',
```

DeleteUser

LAMBDAS

PostImage.py

ImagesAPI > PostImage.py > ...

```python
import json
import boto3

def lambda_handler(event, context):

    user_type = event['user_type']
    user_name = event['user_name']
    image_name = event['image_name']

    s3_client = boto3.client('s3')
    bucket_name = 'luisfmendezl-images'
    image_path = f'{user_type}/{user_name}/{image_name}'

    try:
        s3_response = s3_client.head_object(Bucket=bucket_name, K

        # SNS publication
        sns_client = boto3.client('sns')

        sns_response = sns_client.publish(
            TopicArn = 'arn:aws:sns:us-east-1:896823805584:NewImag
            Subject = 'New Image',
            Message = json.dumps(image_path),
            MessageAttributes = {
                'user_type': {'DataType': 'String', 'StringValue'
            }
        )

        return {
            'statusCode': 200,
            'body': sns_response
        }

    except Exception as e:

        print("Image does not exist:", str(e))
        return {
```

PostImage

ImageAnalizer.py

ImagesAPI > ImageAnalizer.py > lambda_handler

```python
import json
import boto3
from datetime import datetime

def lambda_handler(event, context):

    body = json.loads(event['Records'][0]['body'])
    image_path = json.loads(body['Message'])
    bucket_name = 'luisfmendezl-images'

    rekognition = boto3.client('rekognition')

    # Labels
    response = rekognition.detect_labels(
        Image={
            'S3Object': {
                'Bucket': bucket_name,
                'Name': image_path
            }
        },
        MaxLabels=10
    )

    labels = response['Labels']
    labels = [label['Name'] for label in labels]

    # DynamoDB insertion
    user_type, image_id = image_path.split('/', 1)
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('PostedImages')

    now = datetime.now()

    image = {
        'user_type': user_type,
        'image_id': image_id,
        'image_details': {
```

ImageAnalyzer

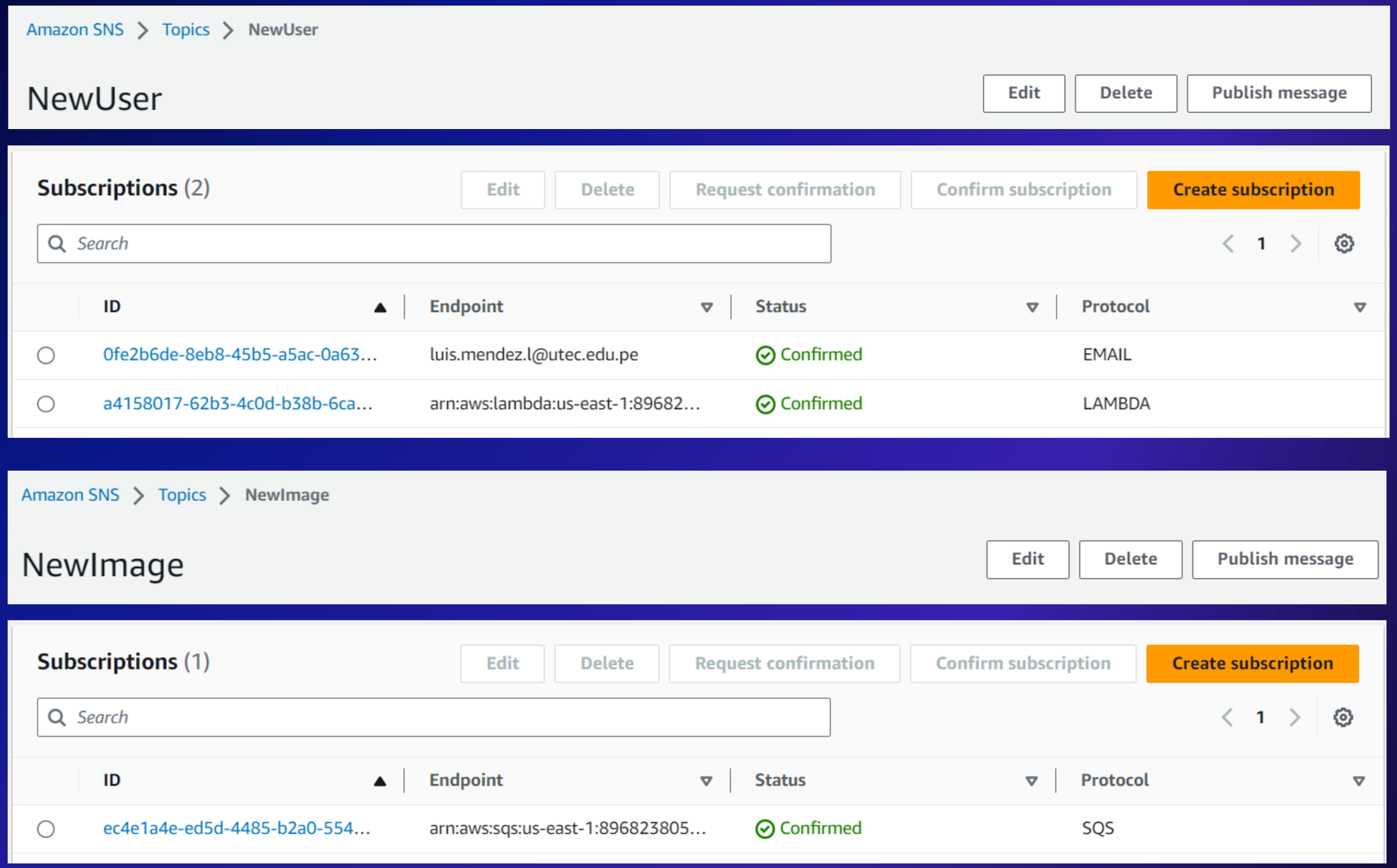
DeleteImage.py

ImagesAPI > DeleteImage.py > ...

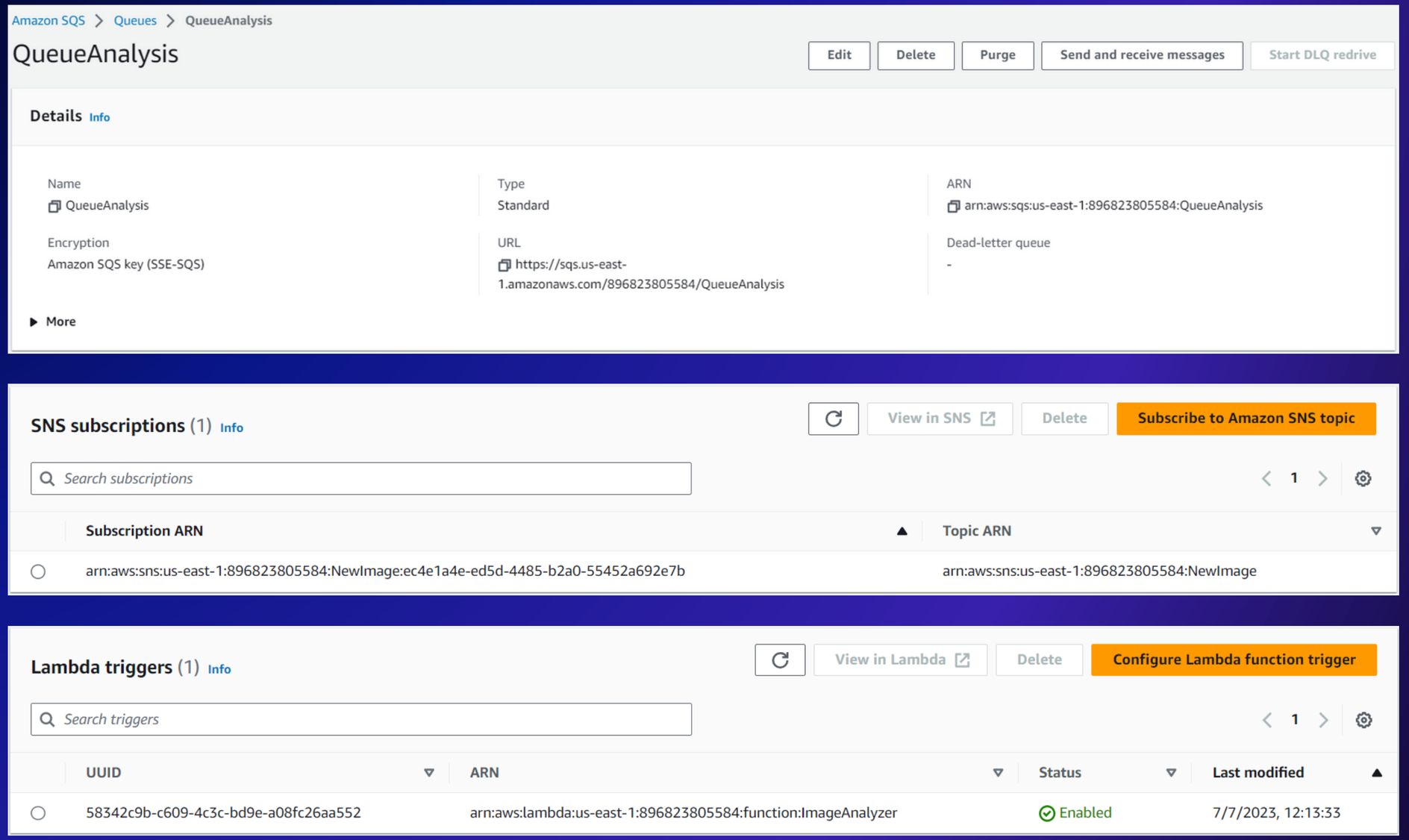
```python
import boto3

def lambda_handler(event, context):

    user_type = event['user_type']
    user_name = event['user_name']
    image_user = f'{user_type}/{user_name}'

    image_name = event['image_name']

    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('PostedImages')

    dynamodb_response = table.delete_item(
        Key={
            'image_user': image_user,
            'image_name': image_name
        }
    )

    return {
        'statusCode': 200,
        'response': dynamodb_response
    }

```

DeleteImage

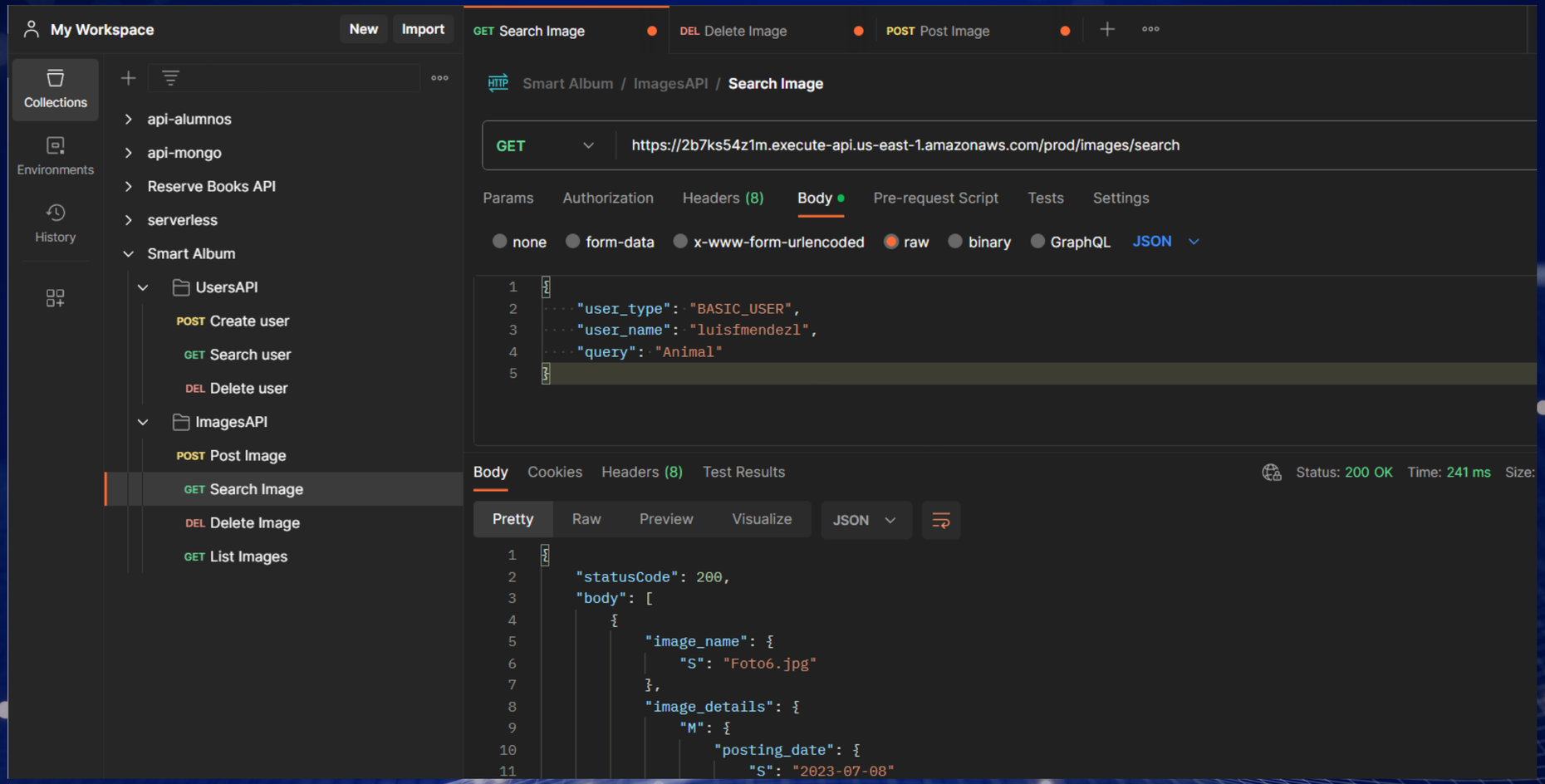
SIMPLE NOTIFICATION SERVICE (SNS)



SIMPLE QUEUE SERVICE (SQS)



DEMO :)


My Workspace
New
Import
Collections
Environments
History
api-alumnos
api-mongo
Reserve Books API
serverless
Smart Album
UsersAPI
POST Create user
GET Search user
DEL Delete user
ImagesAPI
POST Post Image
GET Search Image
DEL Delete Image
GET List Images
GET Search Image
DEL Delete Image
POST Post Image
Smart Album / ImagesAPI / Search Image
GET
https://2b7ks54z1m.execute-api.us-east-1.amazonaws.com/prod/images/search
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"user_type": "BASIC_USER",
"user_name": "luisfmendezl",
"query": "Animal"
}
Body
Cookies
Headers (8)
Test Results
Status: 200 OK
Time: 241 ms
Size:
Pretty
Raw
Preview
Visualize
JSON
{
"statusCode": 200,
"body": [
{
"image_name": {
"S": "Foto6.jpg"
},
"image_details": {
"M": {
"posting_date": {
"S": "2023-07-08"