



CLOUD COMPUTING FINAL PROJECT

Integrantes:
Luis Méndez
Mauricio Álvarez
Jeffry Quintana



CONTENIDO

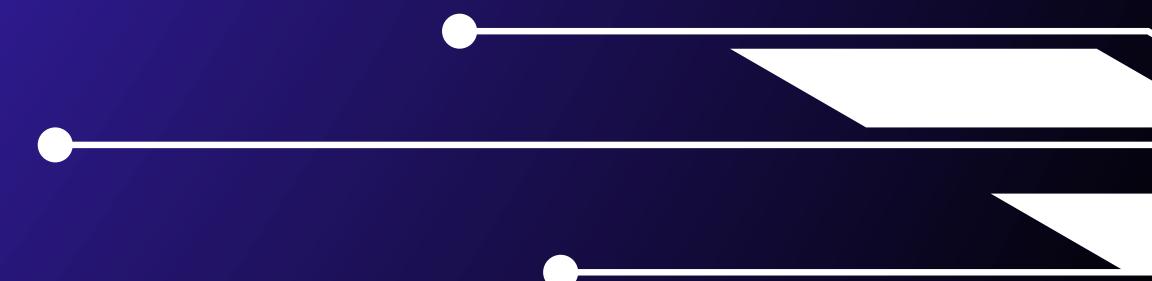
CASO DE USO

SERVICIOS UTILIZADOS

DIAGRAMA DE SOLUCIÓN

ESPECIFICACIONES

DEMO



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.

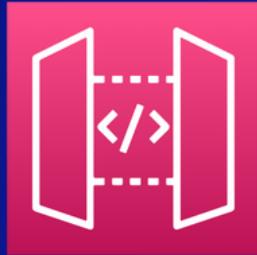


SERVICIOS UTILIZADOS

01

DEVELOPMENT

AWS Lambda
Amazon API Gateway



02

STORAGE

Amazon S3



03

DATABASE

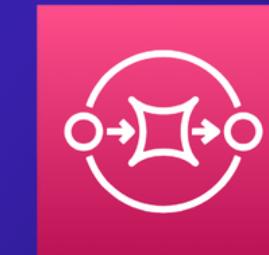
Amazon DynamoDB



04

MESSAGING

Amazon SQS
Amazon SNS



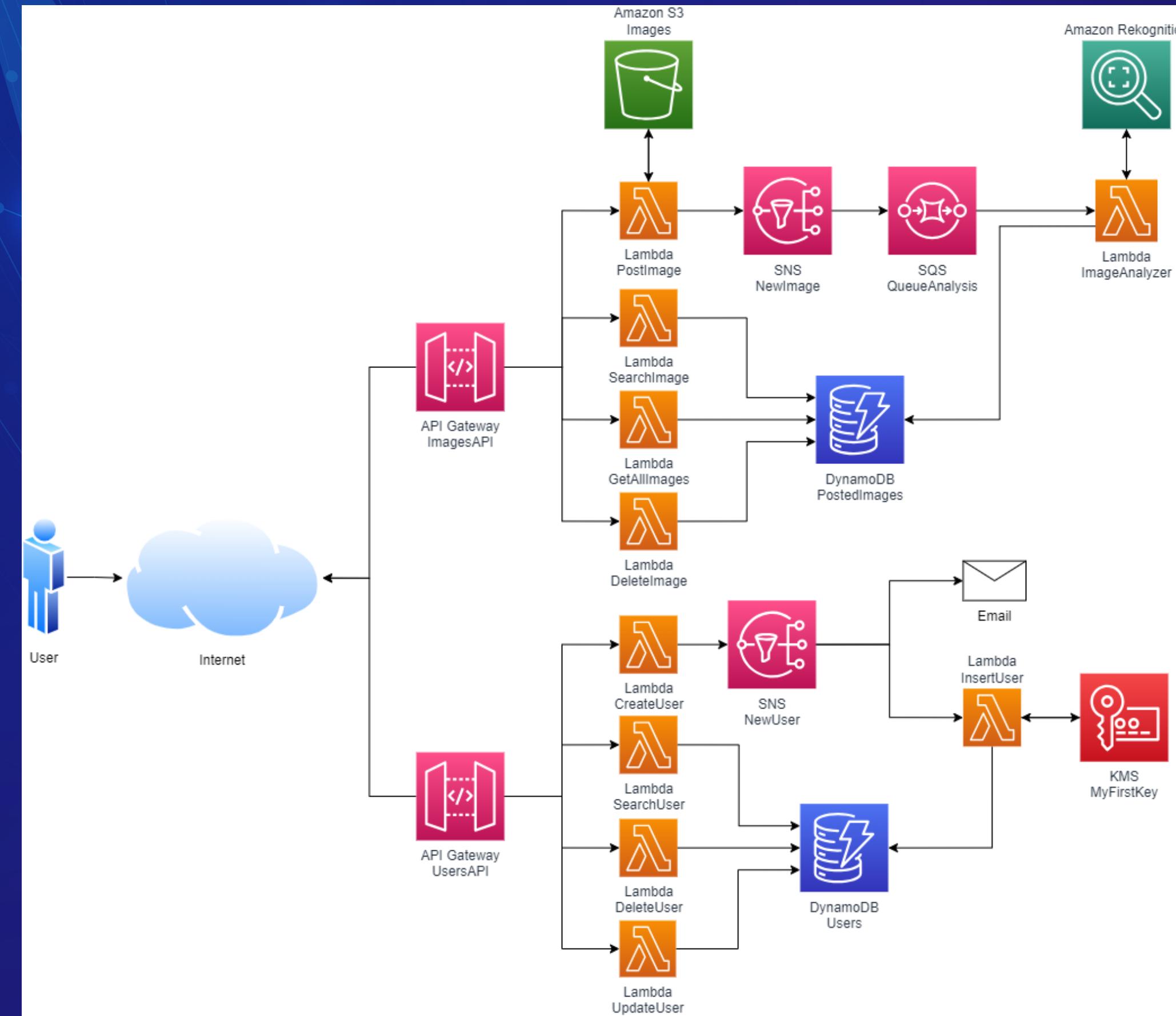
05

AI AND SECURITY

Amazon Rekognition
KMS



DIAGRAMA DE SOLUCIÓN “SMART ALBUM” SERVERLESS APP



AMAZON S3 BUCKET

Amazon S3 > Buckets > luisfmendezl-images

luisfmendezl-images Info

Objects Properties Permissions Metrics Management

Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	BASIC_USER/	Folder	-
<input type="checkbox"/>	CORE_USER/	Folder	-
<input type="checkbox"/>	FULL_PLATFORM_USER/	Folder	-

Amazon S3 > Buckets > luisfmendezl-images > BASIC_USER/

BASIC_USER/

Objects Properties

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	luisfmendezl/	Folder	-
<input type="checkbox"/>	taliaemendezl/	Folder	-

TABLAS DYNAMODB

Users

Autopreview [View table details](#)

Scan or query items
Expand to query or scan items.

Items returned (1)

C Actions ▾ Create item

< 1 > ⚙️ ✖

	user_type	user_name	user_data
<input type="checkbox"/>	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)

C Actions ▾ Create item

< 1 > ⚙️ ✖

	image_user	image_name	image_details	tags
<input type="checkbox"/>	BASIC_USER/luisfmendezl	birthday.jpg	{ "posting_date": { "S": "2023-10-15T12:00:00Z" }, "tags": [{ "S": "People" }, { "S": "Person" }, { "S": "Fun" }, { "S": "Celebration" }] }	[{ "S": "People" }, { "S": "Person" }, { "S": "Fun" }, { "S": "Celebration" }]
<input type="checkbox"/>	BASIC_USER/luisfmendezl	family.jpg	{ "posting_date": { "S": "2023-10-15T12:00:00Z" }, "tags": [{ "S": "People" }, { "S": "Person" }, { "S": "Family" }, { "S": "Relationships" }] }	[{ "S": "People" }, { "S": "Person" }, { "S": "Family" }, { "S": "Relationships" }]
<input type="checkbox"/>	BASIC_USER/luisfmendezl	messi-CR.jpg	{ "posting_date": { "S": "2023-10-15T12:00:00Z" }, "tags": [{ "S": "Sports" }, { "S": "Football" }, { "S": "Lionel Messi" }, { "S": "Barcelona" }] }	[{ "S": "Sports" }, { "S": "Football" }, { "S": "Lionel Messi" }, { "S": "Barcelona" }]

TABLAS DYNAMODB

Attributes View DynamoDB JSON

```
1 ▼ {
  2 ▼   "user_type": {
  3     "S": "BASIC_USER"
  4   },
  5 ▼   "user_name": {
  6     "S": "luisfmendezl"
  7   },
  8 ▼   "user_data": {
  9     "M": {
 10       "email": {
 11         "S": "luis.mendez.l@utec.edu.pe"
 12       },
 13       "first_name": {
 14         "S": "Luis"
 15       },
 16       "last_name": {
 17         "S": "Méndez"
 18       },
 19       "password": {
 20         "S": "AQICAHiNiqb.../pOU1sf/ou3Rr29NAgEQgCNBaSII...5VVEiRRgD"
 21     },
 22     "registration_date": {
 23       "S": "2023-07-07"
 24     }
 25   }
}
```

User instance

Attributes View DynamoDB JSON

```
1 ▼ {
  2 ▼   "image_user": {
  3     "S": "BASIC_USER/luisfmendezl"
  4   },
  5 ▼   "image_name": {
  6     "S": "messi-CR.jpg"
  7   },
  8 ▼   "image_details": {
  9     "M": {
 10       "posting_date": {
 11         "S": "2023-07-08"
 12       },
 13       "posting_time": {
 14         "S": "01:55:29.998089"
 15       }
 16     },
 17   },
 18 ▼   "tags": {
 19     "L": [
 20       {
 21         "S": "Adult"
 22       },
 23       {
 24         "S": "Male"
 25       }
 26     ]
 27   }
}
```

Image instance

AMAZOND APIS GATEWAY

APIs > UsersAPI (t0ui60kfj0) > Stages > prod

Stages Create

- prod
 - /
 - /users
 - /users/create
 - OPTIONS
 - POST
 - /users/delete
 - DELETE
 - OPTIONS
 - /users/search
 - GET
 - OPTIONS
 - /users/update
 - OPTIONS
 - PUT

prod Stage Editor

Settings Logs/Tracing

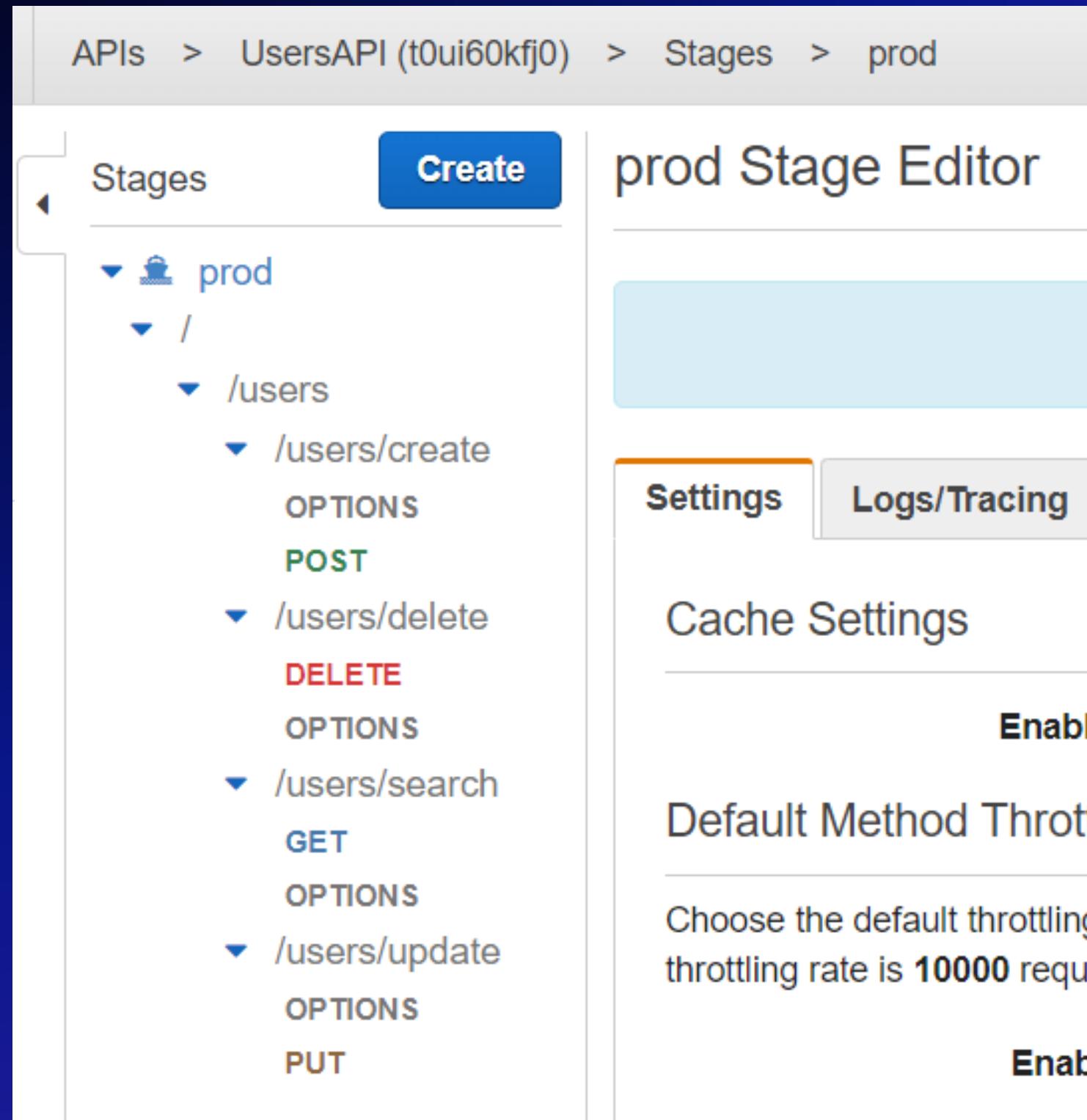
Cache Settings

Enabled

Default Method Throttling

Choose the default throttling rate is **10000** requests per second

Enabled



UsersAPI

APIs > ImagesAPI (2b7ks54z1m) > Stages > prod

Stages Create

- prod
 - /
 - /images
 - /images/delete
 - DELETE
 - OPTIONS
 - /images/list
 - GET
 - OPTIONS
 - /images/post
 - OPTIONS
 - POST
 - /images/search
 - GET
 - OPTIONS

prod Stage Editor

Settings Logs/Tracing

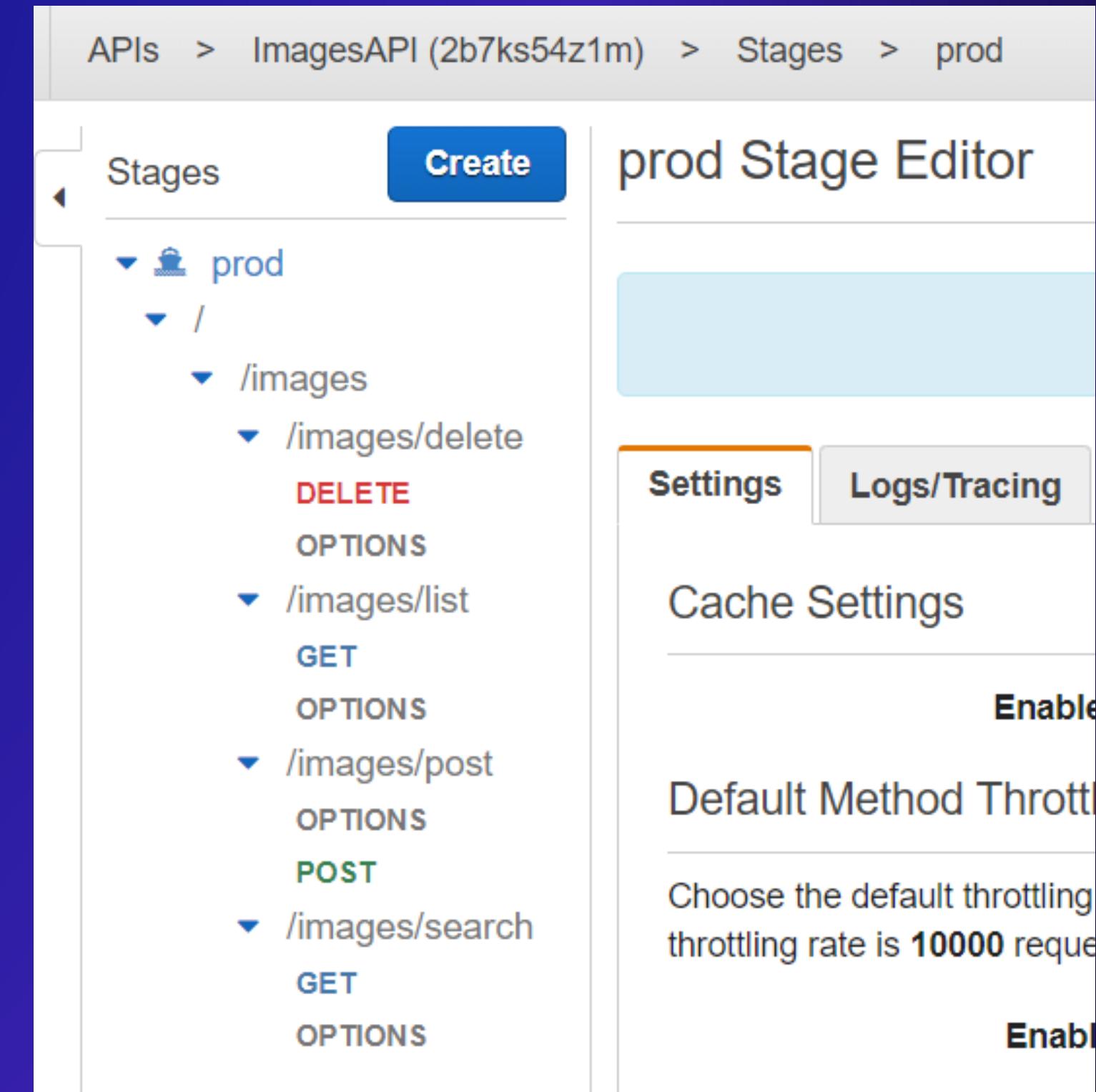
Cache Settings

Enabled

Default Method Throttling

Choose the default throttling rate is **10000** requests per second

Enabled



ImagesAPI

LAMBDAS

```
1 import json
2 import boto3
3
4 def Lambda_handler(event, context):
5
6     # JSON processing
7     user_type = event['user_type']
8     user_name = event['user_name']
9     user_data = event['user_data']
10
11     user = {
12         'user_type': user_type,
13         'user_name': user_name,
14         'user_data': user_data
15     }
16
17     # S3 bucket creation
18     s3_client = boto3.client('s3')
19     bucket_name = 'luisfmendezl-images'
20     folder_name = f'{user_type}/{user_name}/'
21
22     s3_response = s3_client.put_object(
23         Bucket=bucket_name,
24         Key=folder_name
25     )
26     print(s3_response)
27
28     # SNS publication
29     sns_client = boto3.client('sns')
30     sns_response = sns_client.publish(
31         TopicArn = 'arn:aws:sns:us-east-1:8968238055',
32         Subject = 'New user',
33         Message = json.dumps(user),
```

CreateUser

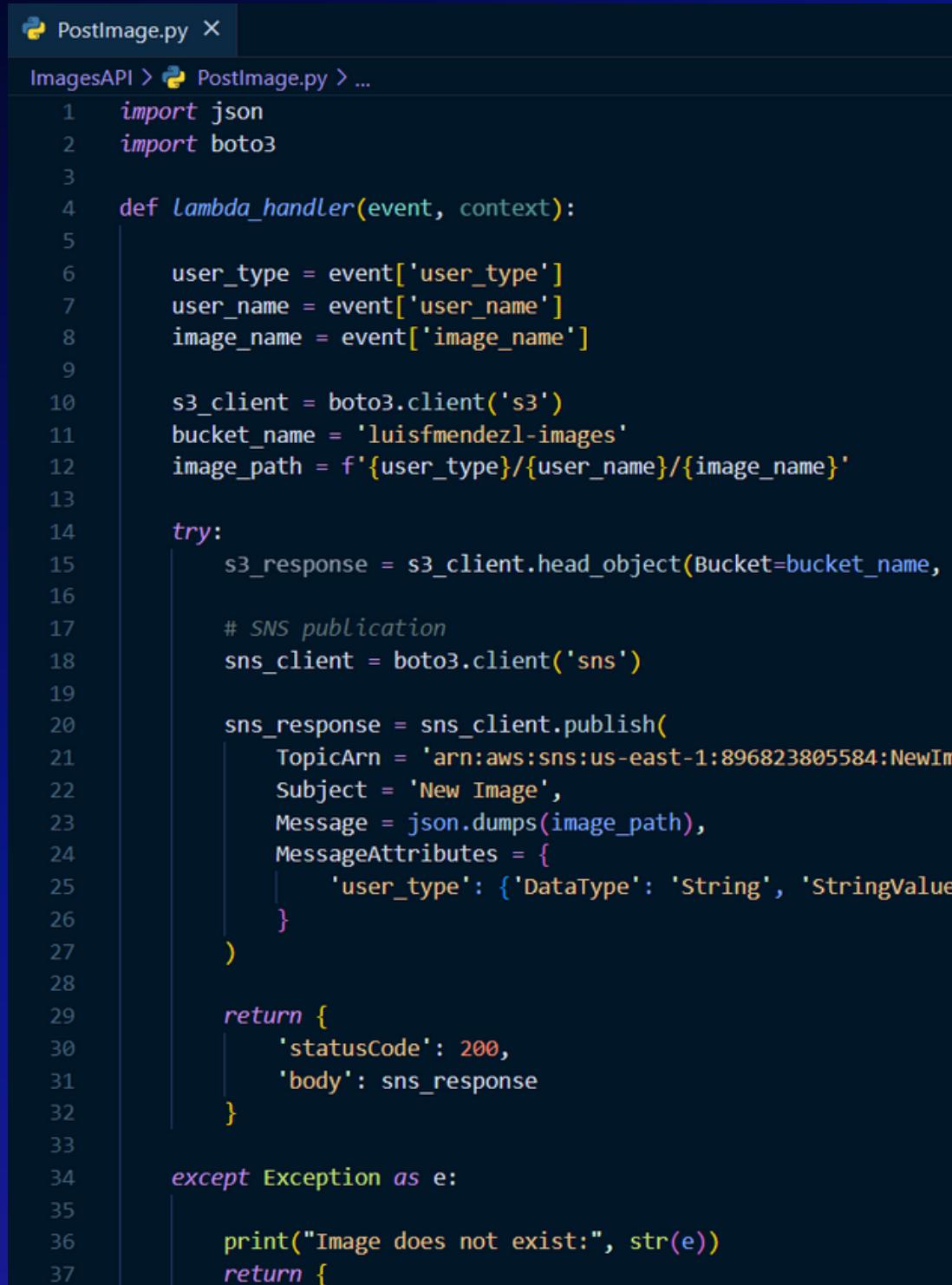
```
1 import json
2 import boto3
3 import base64
4 from datetime import date
5
6 def Lambda_handler(event, context):
7
8     Message = event["Records"][0]["Sns"]["Message"]
9     Message = json.loads(Message)
10
11     user_type = Message['user_type']
12     user_name = Message['user_name']
13     user_data = Message['user_data']
14
15     user_data['password'] = encrypt_password(user_data['password'])
16
17     user_data['registration_date'] = str(date.today())
18
19     dynamodb = boto3.resource('dynamodb')
20     table = dynamodb.Table('t_users')
21
22     user = {
23         'user_type': user_type,
24         'user_name': user_name,
25
26         def encrypt_password(password):
27
28             kms_client = boto3.client('kms')
29             response = kms_client.encrypt(
30                KeyId='fa48f908-27d4-4949-9052-7340cec268fa',
31                 Plaintext=password
32             )
33             encrypted_password = response['CiphertextBlob']
34
35             return base64.b64encode(encrypted_password).decode('utf-8')
```

InsertUser

```
1 import boto3
2
3 def Lambda_handler(event, context):
4
5     user_type = event['user_type']
6     user_name = event['user_name']
7
8     # Delete user from Users
9     dynamodb = boto3.resource('dynamodb')
10    table = dynamodb.Table('Users')
11
12    dynamodb_response = table.delete_item(
13        Key={
14            'user_type': user_type,
15            'user_name': user_name
16        }
17
18    # Delete folder from the Bucket
19    bucket_name = 'luisfmendezl-images'
20    prefix = f'{user_type}/{user_name}/'
21
22    s3 = boto3.client('s3')
23    s3_response = s3.list_objects_v2(Bucket=bucket_name, Prefix=prefix)
24
25    print(s3_response)
26
27    if 'Contents' in s3_response:
28        objects = [{ 'Key': obj['Key']} for obj in s3_response['Contents']]
29        s3.delete_objects(Bucket=bucket_name, Delete={'Objects': objects})
30
31    # Delete their images from PostedImages
32    dynamodb = boto3.client('dynamodb')
33    image_user = f'{user_type}/{user_name}'
34
35    scan_params = {
36        'TableName': 'PostedImages',
37        'KeyConditionExpression': 'ImageUser = :user',
38        'ValueFilter': {':user': { 'S': image_user }}
```

DeleteUser

LAMBDAS



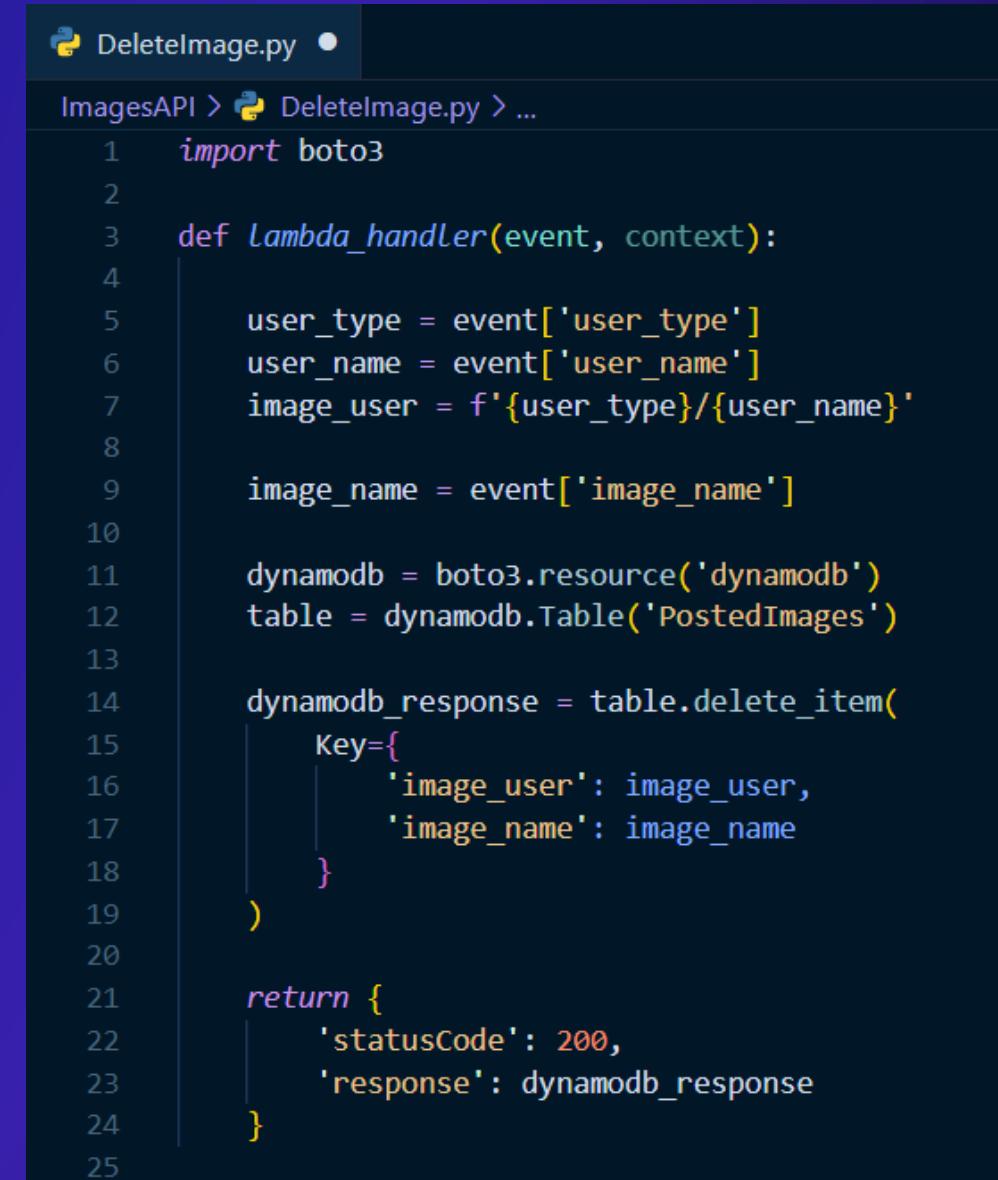
```
PostImage.py X
ImagesAPI > PostImage.py > ...
1 import json
2 import boto3
3
4 def Lambda_handler(event, context):
5
6     user_type = event['user_type']
7     user_name = event['user_name']
8     image_name = event['image_name']
9
10    s3_client = boto3.client('s3')
11    bucket_name = 'luisfmendezl-images'
12    image_path = f'{user_type}/{user_name}/{image_name}'
13
14    try:
15        s3_response = s3_client.head_object(Bucket=bucket_name, Key=image_path)
16
17        # SNS publication
18        sns_client = boto3.client('sns')
19
20        sns_response = sns_client.publish(
21            TopicArn = 'arn:aws:sns:us-east-1:896823805584:NewImage',
22            Subject = 'New Image',
23            Message = json.dumps(image_path),
24            MessageAttributes = {
25                'user_type': {'DataType': 'String', 'StringValue': user_type}
26            }
27        )
28
29        return {
30            'statusCode': 200,
31            'body': sns_response
32        }
33
34    except Exception as e:
35
36        print("Image does not exist:", str(e))
37        return {
```

PostImage



```
ImageAnalyzer.py X
ImagesAPI > ImageAnalyzer.py > lambda_handler
1 import json
2 import boto3
3 from datetime import datetime
4
5 def Lambda_handler(event, context):
6
7     body = json.loads(event['Records'][0]['body'])
8     image_path = json.loads(body['Message'])
9     bucket_name = 'luisfmendezl-images'
10
11    rekognition = boto3.client('rekognition')
12
13    # Labels
14    response = rekognition.detect_labels(
15        Image={
16            'S3Object': {
17                'Bucket': bucket_name,
18                'Name': image_path
19            }
20        },
21        MaxLabels=10
22    )
23
24    labels = response['Labels']
25    labels = [label['Name'] for label in labels]
26
27    # DynamoDB insertion
28    user_type, image_id = image_path.split('/', 1)
29    dynamodb = boto3.resource('dynamodb')
30    table = dynamodb.Table('PostedImages')
31
32    now = datetime.now()
33
34    image = {
35        'user_type': user_type,
36        'image_id': image_id,
37        'image_details': {
```

ImageAnalyzer



```
DeleteImage.py •
ImagesAPI > DeleteImage.py > ...
1 import boto3
2
3 def Lambda_handler(event, context):
4
5    user_type = event['user_type']
6    user_name = event['user_name']
7    image_user = f'{user_type}/{user_name}'
8
9    image_name = event['image_name']
10
11    dynamodb = boto3.resource('dynamodb')
12    table = dynamodb.Table('PostedImages')
13
14    dynamodb_response = table.delete_item(
15        Key={
16            'image_user': image_user,
17            'image_name': image_name
18        }
19    )
20
21    return {
22        'statusCode': 200,
23        'response': dynamodb_response
24    }
25
```

DeleteImage

SIMPLE NOTIFICATION SERVICE (SNS)

Amazon SNS > Topics > NewUser

NewUser

Edit Delete Publish message

Subscriptions (2)

Edit Delete Request confirmation Confirm subscription Create subscription

ID	Endpoint	Status	Protocol
0fe2b6de-8eb8-45b5-a5ac-0a63...	luis.mendez.l@utec.edu.pe	Confirmed	EMAIL
a4158017-62b3-4c0d-b38b-6ca...	arn:aws:lambda:us-east-1:89682...	Confirmed	LAMBDA

Amazon SNS > Topics > NewImage

NewImage

Edit Delete Publish message

Subscriptions (1)

Edit Delete Request confirmation Confirm subscription Create subscription

ID	Endpoint	Status	Protocol
ec4e1a4e-ed5d-4485-b2a0-554...	arn:aws:sqs:us-east-1:896823805...	Confirmed	SQS

SIMPLE QUEUE SERVICE (SQS)

Amazon SQS > Queues > QueueAnalysis

QueueAnalysis

Edit Delete Purge Send and receive messages Start DLQ redrive

Details Info

Name	Type	ARN
QueueAnalysis	Standard	arn:aws:sqs:us-east-1:896823805584:QueueAnalysis
Encryption	URL	Dead-letter queue
Amazon SQS key (SSE-SQS)	https://sqs.us-east-1.amazonaws.com/896823805584/QueueAnalysis	-

► More

SNS subscriptions (1) Info

C View in SNS Delete Subscribe to Amazon SNS topic

Search subscriptions < 1 > 🔍

Subscription ARN	Topic ARN
arn:aws:sns:us-east-1:896823805584:NewImage:ec4e1a4e-ed5d-4485-b2a0-55452a692e7b	arn:aws:sns:us-east-1:896823805584:NewImage

Lambda triggers (1) Info

C View in Lambda Delete Configure Lambda function trigger

Search triggers < 1 > 🔍

UUID	ARN	Status	Last modified
58342c9b-c609-4c3c-bd9e-a08fc26aa552	arn:aws:lambda:us-east-1:896823805584:function:ImageAnalyzer	Enabled	7/7/2023, 12:13:33

DEMO :)

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections, environments, and history. The 'Collections' section shows 'api-alumnos', 'api-mongo', 'Reserve Books API', 'serverless', and 'Smart Album'. Under 'Smart Album', there are two folders: 'UsersAPI' containing 'POST Create user', 'GET Search user', and 'DEL Delete user'; and 'ImagesAPI' containing 'POST Post Image', 'GET Search Image' (which is selected), 'DEL Delete Image', and 'GET List Images'. At the top, a navigation bar has tabs for 'New' and 'Import'. Below the navigation, there are four buttons: 'GET Search Image' (highlighted in green), 'DEL Delete Image' (orange), 'POST Post Image' (yellow), and a '+' button. To the right of these buttons is a 'More' icon. The main workspace shows an 'HTTP Smart Album / ImagesAPI / Search Image' request. The method is 'GET' and the URL is 'https://2b7ks54z1m.execute-api.us-east-1.amazonaws.com/prod/images/search'. The 'Body' tab is selected, showing a JSON payload:

```
1  {
2    "user_type": "BASIC_USER",
3    "user_name": "luisfmendezl",
4    "query": "Animal"
5 }
```

Below the body, the 'Status' is 200 OK, 'Time' is 241 ms, and 'Size' is 1.53 KB. The 'Body' tab is also selected here, showing the response body:

```
1  {
2    "statusCode": 200,
3    "body": [
4      {
5        "image_name": {
6          "S": "Foto6.jpg"
7        },
8        "image_details": {
9          "M": {
10            "posting_date": {
11              "S": "2023-07-08"
12            }
13          }
14        }
15      }
16    ]
17  }
```