

Reporte de Proyecto Cloud con AWS

Estudiante: Inti Luna Avilés

Programa: Ingeniería de datos.
DATAHACK

2024-04-08

Resumen.....	3
Deployment.....	4
Detalle de proyecto.....	5
Diseño.....	5
Diagramas.....	5
Servicios usados al hacer deploy.....	7
Endpoints - Funciones.....	8
POST anuncios.....	8
GET anuncios.....	9
GET anuncios/id.....	9
DELETE anuncios/id.....	10
Endpoints - Pruebas.....	10
POST - postman.....	10
GET anuncios.....	11
GET anuncios/id.....	12
DELETE anuncios/id.....	12
Integrar endpoint en HTML.....	12
Página inicial.....	17
Publicar anuncio.....	18
Ver anuncios.....	18
Ver detalle de anuncio.....	19
CHAT.....	19
Deployment inicial.....	19
Integración a endpoint de anuncios.....	19
Resultado final.....	22
Bash.....	23
Recursos usados.....	24

Resumen

Se ha creado una aplicación para crear, llamar y eliminar anuncios. Adicionalmente se ha utilizado el código de simple-chat con modificaciones menores para adaptarlo al proyecto y tener todo el código con definición de endpoints en un solo fichero y ser llamado por el código HTML que define la interfase gráfica y realiza las llamadas a los endpoints. Para desplegar, se ha utilizado serverless que interactúa con los proveedores de servicio (AWS en este caso) para definir y desplegar infraestructura como código y de manera muy rápida. Por último se ha creado fichero bash (.sh) para desplegar con serverless, insertar registros de prueba a la tabla de anuncios y abrir el navegador automáticamente con la URL del despliegue.

Deployment

Solo se tiene que ejecutar fichero bash

```
>./deploy_and_fill_db.sh
```

Este fichero:

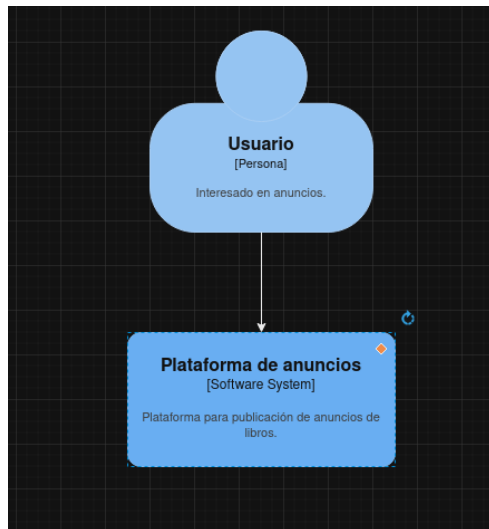
- hace despliegue en AWS la infraestructura usando serverless
- inserta registros de prueba
- abre el navegador con URL apropiada

Detalle de proyecto

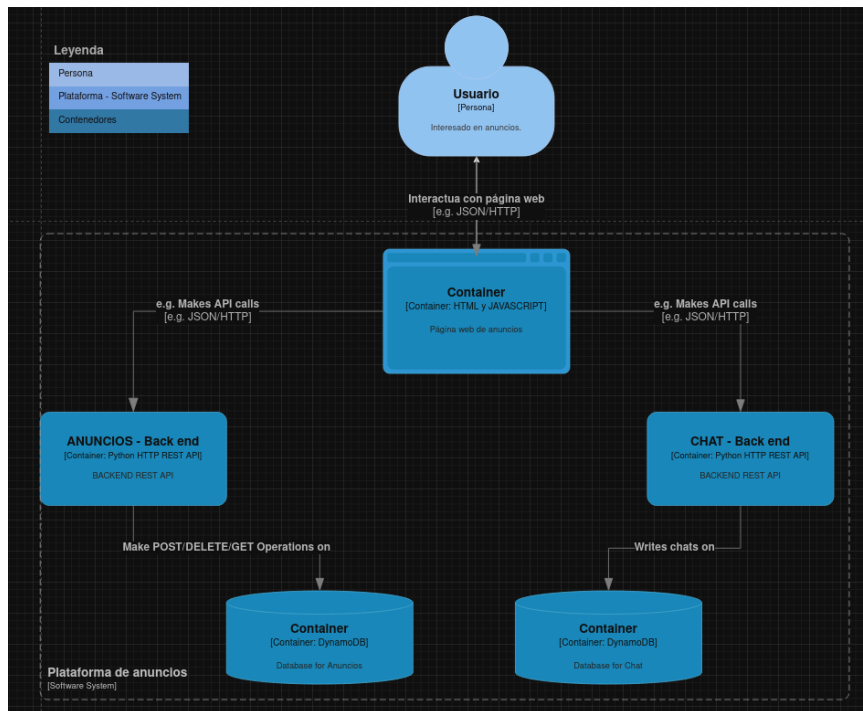
Diseño

Diagramas

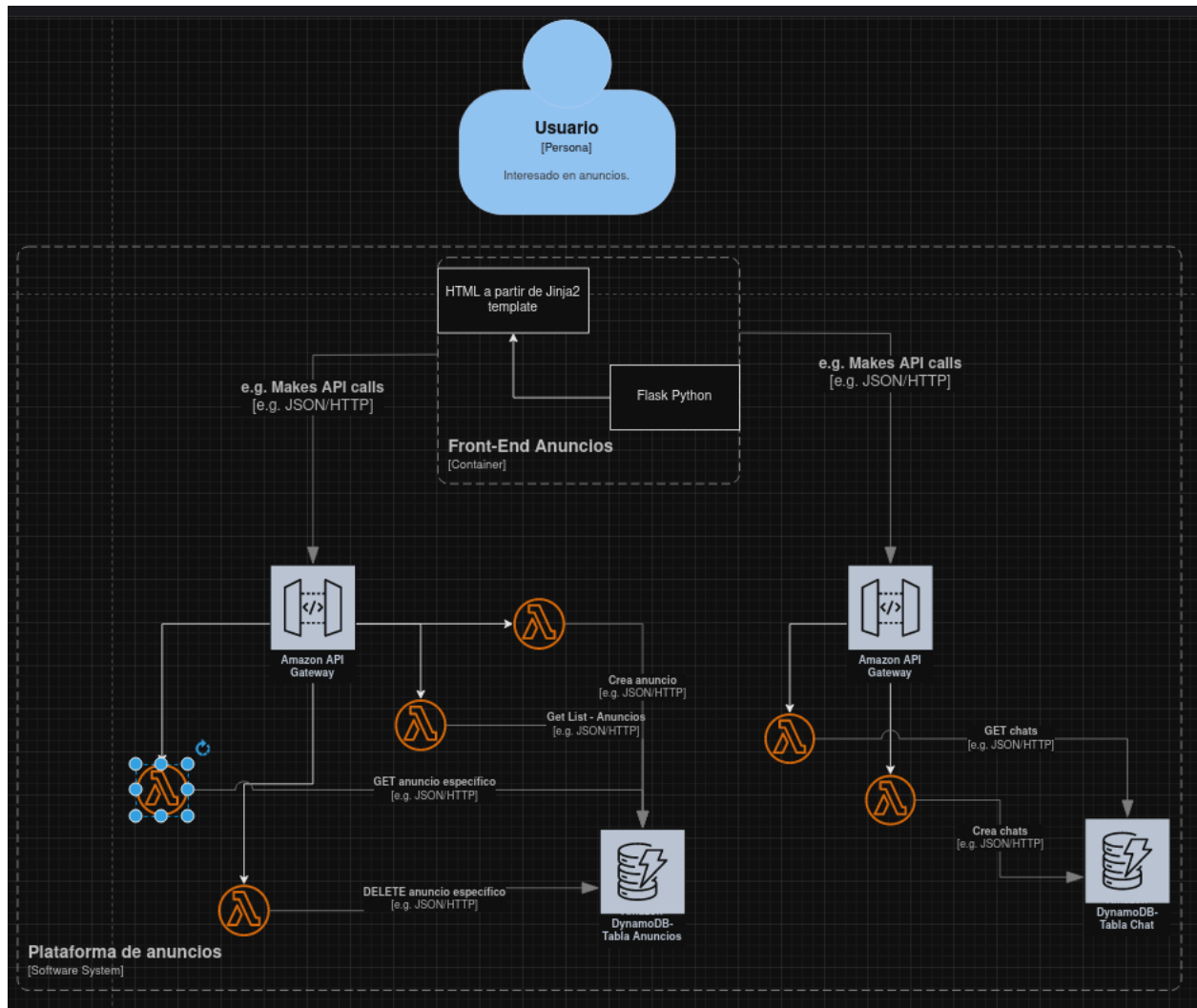
Nivel 1



Nivel 2



Nivel 3



Anuncios - Variables

Variables:

- Anuncio_id
- Titulo
- Precio
- Email

Servicios usados al hacer deploy

Funciones Lambda: Es un servicio serverless y se cobra por número de llamadas a la función y por el tiempo que toma ejecutar el código. A la fecha (2024-04-08), el costo es de \$0.20 per 1M requests y de \$0.0000166667 for every GB-second. Limitante: tiempo máximo de ejecución de 15 minutos.

API Gateway: Gestiona las tareas relacionadas a las llamadas de API incluyendo administración de tráfico, autorizaciones y control de versiones entre otras.

DynamoDB: Es una base de datos NoSQL serverless. La base de datos puede estar en uso o no, cuando no está en uso el costo es casi cero. Se cobra por almacenamiento. Y cuando está en uso, se cobra por llamadas a la base de datos para escribir y leer. Otros costos a tomar en cuenta son backups, duplicidad de tablas multi-regiones entre otros.

Role IAM: Es creado para poder manipular dynamodb.

En la página de estimación de costos (<https://calculator.aws/#/>) se puede ver el detalle que es prácticamente cero para el ejercicio del proyecto en cuestión.

Endpoints - Funciones

POST anuncios

```
@app.route('/anuncios', methods=['POST'])
def create_anuncio():
    anuncio_id = request.json.get('anuncioId')
    titulo = request.json.get('titulo')
    precio = request.json.get('precio')
    email = request.json.get('email')
    if not anuncio_id or not titulo or not precio:
        return jsonify({'error': 'Por favor brinde: "anuncioId", "titulo",
"precio", "email"'}), 400

    dynamodb_client.put_item(
        TableName=ANUNCIOS_TABLE, Item={'anuncioId': {'S': anuncio_id},
'titulo': {'S': titulo}, 'precio': {'S': precio}, 'email': {'S': email}}
    )

    return jsonify({'anuncioId': anuncio_id, 'titulo': titulo}),200
```


GET anuncios

```
@app.route('/anuncios', methods=['GET'])
def get_list_anuncios():
    try:
        response = dynamodb_client.scan(
            TableName=ANUNCIOS_TABLE
        )
        items = response.get('Items', [])
        if not items:
            return jsonify({'message': 'No users found in the table'}), 404

        anuncios = []
        for item in items:
            anuncio = {
                'anuncioId': item.get('anuncioId').get('S'),
                'titulo': item.get('titulo').get('S')
            }
            anuncios.append(anuncio)

        return jsonify(anuncios), 200

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

GET anuncios/id

```
@app.route('/anuncios/<string:anuncio_id>')
def get_anuncio_detalle(anuncio_id):
    result = dynamodb_client.get_item(
        TableName=ANUNCIOS_TABLE, Key={'anuncioId': {'S': anuncio_id}}
    )
    item = result.get('Item')
    if not item:
        return jsonify({'error': 'No pude encontrar anuncio con ese "anuncioID"'}), 404

    return jsonify(
        {'anuncioID': item.get('anuncioId').get('S'), 'titulo':
item.get('titulo').get('S'), 'precio': item.get('precio').get('S'), 'email':
item.get('email').get('S')}
    )
```

DELETE anuncios/id

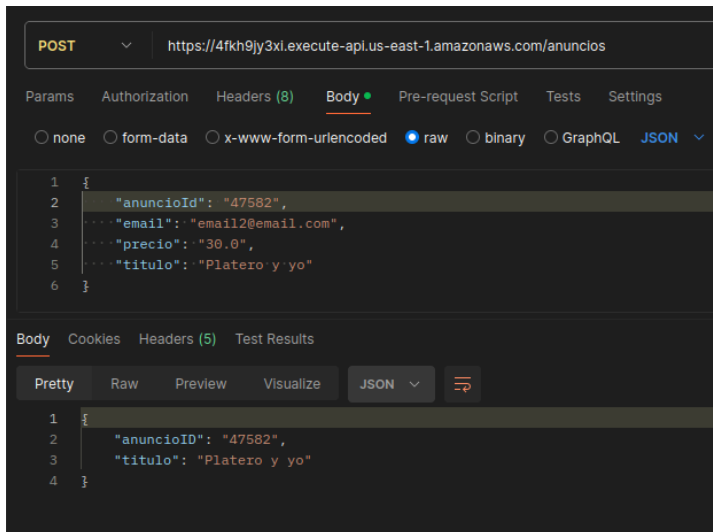
```
# Endpoint para eliminar un anuncio por su ID
@app.route('/anuncios/<string:anuncio_id>', methods=['DELETE'])
def delete_anuncio(anuncio_id):
    # Primero, verifica si el anuncio existe
    result = dynamodb_client.get_item(
        TableName=ANUNCIOS_TABLE, Key={'anuncioId': {'S': anuncio_id}}
    )
    item = result.get('Item')
    if not item:
        return jsonify({'error': 'No pude encontrar anuncio con ese "anuncioID"' }), 404

    # Si el anuncio existe, procede a eliminarlo
    dynamodb_client.delete_item(
        TableName=ANUNCIOS_TABLE, Key={'anuncioId': {'S': anuncio_id}}
    )

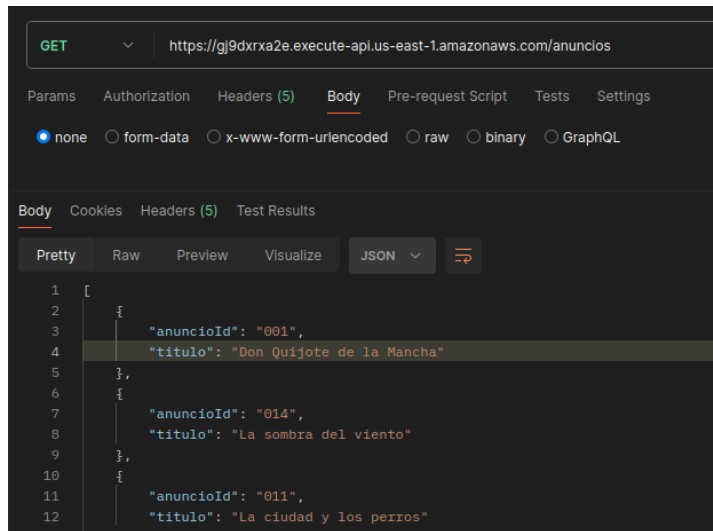
    return jsonify({'message': 'Anuncio eliminado exitosamente'}), 200
```

Endpoints - Pruebas

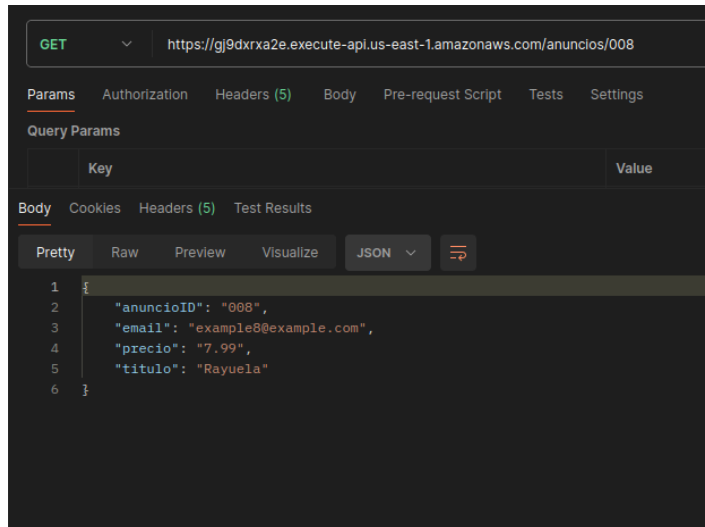
POST - postman



GET anuncios



GET anuncios/id



Registramos varios anuncios usando un script que llama a endpoint-post

Probamos en terminal

```
(serverless) vant@Agile:~/Documents/datahack/cloud/proyecto3_cloud/aws-python-flask-dynamodb-api-project$ python preload_using_endpoint.py https://nof8lfduk6.execute-api.eu-west-1.amazonaws.com
https://nof8lfduk6.execute-api.eu-west-1.amazonaws.com/anuncios
anuncio id: 001, titulo: Don Quijote de la Mancha
{'anuncioId': '001', 'titulo': 'Don Quijote de la Mancha'}
anuncio id: 002, titulo: Cien años de soledad
{'anuncioId': '002', 'titulo': 'Cien años de soledad'}
anuncio id: 003, titulo: El amor en los tiempos del cólera
{'anuncioId': '003', 'titulo': 'El amor en los tiempos del cólera'}
anuncio id: 004, titulo: La sombra del viento
{'anuncioId': '004', 'titulo': 'La sombra del viento'}
anuncio id: 005, titulo: Crónica de una muerte anunciada
{'anuncioId': '005', 'titulo': 'Crónica de una muerte anunciada'}
anuncio id: 006, titulo: El laberinto de los espíritus
{'anuncioId': '006', 'titulo': 'El laberinto de los espíritus'}
anuncio id: 007, titulo: La casa de los espíritus
{'anuncioId': '007', 'titulo': 'La casa de los espíritus'}
anuncio id: 008, titulo: Rayuela
{'anuncioId': '008', 'titulo': 'Rayuela'}
anuncio id: 009, titulo: El perfume: historia de un asesino
```

DELETE anuncios/id

Revisamos anuncios

Anuncio ID	Título
001	magia
011	La ciudad y los perros
007	La casa de los espíritus

Eliminamos anuncio

Integrar endpoint en HTML

Se crea html con botón para llamar a endpoint pero ocurre un problema con cors.

Mostrar Anuncios

Mostrar Anuncios

Ocurrió un error al cargar los anuncios.

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Filter Output

Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at <https://q19d4rx92e.execute-api.us-east-1.amazonaws.com/anuncios>. (Reason: CORS header "Access-Control-Allow-Origin" missing). Status code: 200.

HTML jinja2 template

Creamos un template para mostrar los anuncios:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Anuncios</title>
  <style>
    table {
      border-collapse: collapse;
      width: 100%;
    }

    th, td {
      border: 1px solid #dddddd;
      text-align: left;
      padding: 8px;
    }

    th {
      background-color: #f2f2f2;
    }
  </style>
</head>
<body>
  <h1>Anuncios</h1>
  <table>
    <thead>
      <tr>
        <th>Anuncio ID</th>
        <th>Título</th>
      </tr>
    </thead>
    <tbody>
      {% for anuncio in anuncios %}
      <tr>
        <td>{{ anuncio.anuncioId }}</td>
        <td>{{ anuncio.titulo }}</td>
      </tr>
      </tbody>
    </table>
  </body>
</html>
```

```

        </tr>
        {% endfor %}
    </tbody>
</table>
</body>
</html>

```

Ajustamos la función para enviar datos al template html:

```

@app.route('/anuncios', methods=['GET'])
def get_list_anuncios():
    try:
        response = dynamodb_client.scan(
            TableName=ANUNCIOS_TABLE
        )
        items = response.get('Items', [])
        if not items:
            return jsonify({'message': 'No users found in the table'}), 404

        anuncios = []
        for item in items:
            anuncio = {
                'anuncioId': item.get('anuncioId').get('S'),
                'titulo': item.get('titulo').get('S')
            }
            anuncios.append(anuncio)

        #return jsonify(anuncios), 200
        return render_template('anuncios.html', anuncios=anuncios), 200

    except Exception as e:
        return jsonify({'error': str(e)}), 500

```

Resultado

Anuncios

Anuncio ID	Título
001	Don Quijote de la Mancha
014	La sombra del viento
011	La ciudad y los perros
007	La casa de los espíritus
010	Los renglones torcidos de Dios
008	Rayuela
013	La historia interminable
020	Memorias de una geisha
003	El amor en los tiempos del cólera
009	El perfume: historia de un asesino
015	La hoguera de las vanidades
002	Cien años de soledad
017	Los juegos del hambre
004	La sombra del viento
005	Crónica de una muerte anunciada
016	El túnel
012	La fiesta del chivo
006	El laberinto de los espíritus
018	El prisionero del cielo
019	La elegancia del erizo

Se integran resultados de endpoints a html templates y se obtiene:

Página inicial

 <https://kvjtodhkb5.execute-api.eu-west-1.amazonaws.com>

Proyecto para módulo cloud en Datahack por Inti Luna

Plataforma de venta de libros

Usando Python: Flask, Jinja2 Template, y Serverless para crear infraestructura con AWS

código disponible en: [repositorio de github](#)

Ver anuncios

Publicar anuncio

Publicar anuncio

https://kvjtodhkb5.execute-api.eu-west-1.amazonaws.com/nuevo-anuncio

Crear Anuncio

ID de Anuncio:

Título:

Precio:

Email de contacto:

Crear Anuncio

Reset

Inicio

Ver anuncios

Anuncios	
Anuncio ID	Título
001	magia
011	La ciudad y los perros
010	Los renglones torcidos de Dios
013	La historia interminable
020	Memorias de una geisha
003	El amor en los tiempos del cólera
009	El perfume: historia de un asesino
015	La hoguera de las vanidades
007	El gran libro de magia de Isadora Moon y Mirabella
0033	Cuando el sol despierta
002	Cien años de soledad
017	Los juegos del hambre
004	La sombra del viento
005	Crónica de una muerte anunciada
016	El túnel
012	La fiesta del chivo
006	El laberinto de los espíritus
018	El prisionero del cielo
019	La elegancia del erizo

Inicio

Publicar anuncio

Ver detalle de anuncio



El laberinto de los espíritus

ID de Anuncio: 006
Precio: 8.99
Email de contacto: example6@example.com

- Ver anuncios
- Nuevo anuncio
- Eliminar anuncio

CHAT

Deployment inicial

Se prueba en un deployment independiente con algunos cambios menores:

Antes	Después
<pre>provider: name: aws runtime: python3.8 region: eu-west-1 environment: DYNAMODB_MESSAGES_TABLE: simple-chat-messages iamRoleStatements: - Effect: "Allow" Action: - dynamodb:Query - dynamodb:PutItem Resource: - "arn:aws:dynamodb:\${opt:region, self:provider.region}:*:table/\${self:provider.environment.DYNAMODB_MESSAGES_TABLE}"</pre>	<pre>environment: DYNAMODB_MESSAGES_TABLE: simple-chat-messages iamRoleStatements: - role: role: statement: - Effect: "Allow" Action: - dynamodb:Query - dynamodb:PutItem Resource: - "arn:aws:dynamodb:\${opt:region, self:provider.region}:*:table/\${self:provider.environment.DYNAMODB_MESSAGES_TABLE}"</pre>

Se puede enviar y recibir mensajes.

Integración a endpoint de anuncios

- Se copia handler.py de repositorio github simple-chat
- Se modifica serverless.yml para tener ambos servicios en un deployment

Resultados

```
Packaging Python WSGI handler...
Warning: Serverless Framework observability features do not support the following runtime: python3.9
✓ Your AWS account is now integrated into Serverless Framework Observability
✓ Serverless Framework Observability is enabled

✓ Service deployed to stack aws-python-flask-dynamodb-api-project-dev (61s)

dashboard: https://app.serverless.com/intiluna/apps/dh-geopython/aws-python-flask-dynamodb-api-project/dev/eu-west-1
endpoints:
  GET - https://7l565l25ei.execute-api.eu-west-1.amazonaws.com/dev/{chat_id}
  POST - https://7l565l25ei.execute-api.eu-west-1.amazonaws.com/dev/{chat_id}
  ANY - https://kvjtodhkb5.execute-api.eu-west-1.amazonaws.com
functions:
  api: aws-python-flask-dynamodb-api-project-dev-api (19 MB)
  get_messages: aws-python-flask-dynamodb-api-project-dev-get_messages (19 MB)
  send_message: aws-python-flask-dynamodb-api-project-dev-send_message (19 MB)
o (base) vant@Agile:~/Documents/datahack/cloud/proyecto3_cloud/aws-python-flask-dynamodb-api-project$
```

Envio mensajes

The screenshot shows a REST client interface with a POST request to `https://7l565l25ei.execute-api.eu-west-1.amazonaws.com/dev/1`. The request body is a JSON object: `{ "user_id": "1", "text": "hola, para 9 de abril por favor" }`. The response is also in JSON format: `{ "status": 201, "title": "OK", "detail": "New message posted into chat 1" }`.

```
GET https://7l565l25ei.execute-api.eu-west-1.amazonaws.com/dev/1
POST https://7l565l25ei.execute-api.eu-west-1.amazonaws.com/dev/1
DEL https://kvjtodhkb5.execute-api.eu-west-1.amazonaws.com/dev/1
```

POST `https://7l565l25ei.execute-api.eu-west-1.amazonaws.com/dev/1`

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

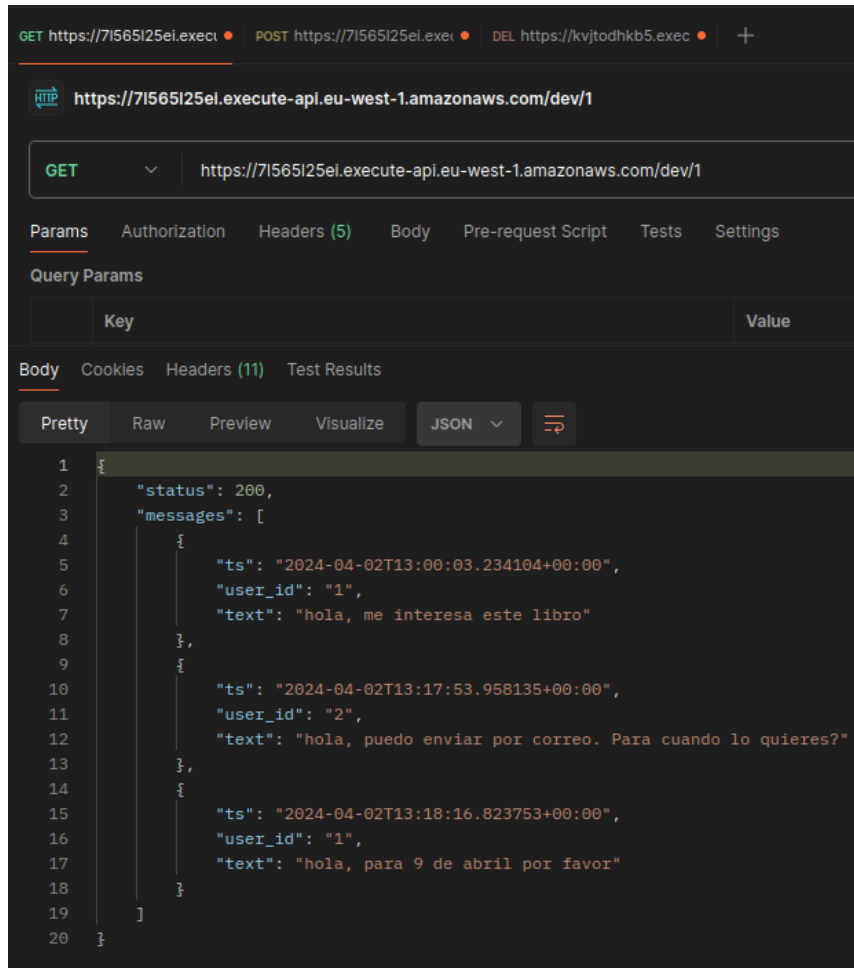
```
1 {
2   "user_id": "1",
3   "text": "hola, para 9 de abril por favor"
4 }
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize **JSON**

```
1 {
2   "status": 201,
3   "title": "OK",
4   "detail": "New message posted into chat 1"
5 }
```

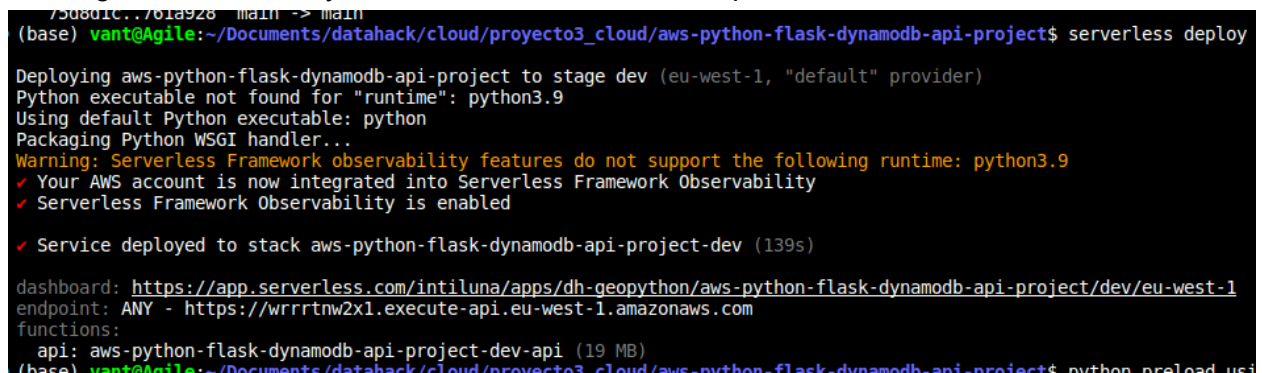
Obtener mensajes



The screenshot shows a web browser with a GET request to `https://71565125ei.execute-api.eu-west-1.amazonaws.com/dev/1`. The response is a JSON object with a status of 200 and an array of three messages. Each message contains a timestamp, a user ID, and a text field.

```
1 {
2   "status": 200,
3   "messages": [
4     {
5       "ts": "2024-04-02T13:00:03.234104+00:00",
6       "user_id": "1",
7       "text": "hola, me interesa este libro"
8     },
9     {
10      "ts": "2024-04-02T13:17:53.958135+00:00",
11      "user_id": "2",
12      "text": "hola, puedo enviar por correo. Para cuando lo quieres?"
13     },
14     {
15      "ts": "2024-04-02T13:18:16.823753+00:00",
16      "user_id": "1",
17      "text": "hola, para 9 de abril por favor"
18     }
19   ]
20 }
```

Se integra las funciones y ahora se tienen todos los endpoint en un mismo URL



The screenshot shows a terminal window with the following output:

```
7508d1c7761a928 main -> main
(base) vant@Agile:~/Documents/datahack/cloud/proyecto3_cloud/aws-python-flask-dynamodb-api-project$ serverless deploy

Deploying aws-python-flask-dynamodb-api-project to stage dev (eu-west-1, "default" provider)
Python executable not found for "runtime": python3.9
Using default Python executable: python
Packaging Python WSGI handler...
Warning: Serverless Framework observability features do not support the following runtime: python3.9
✓ Your AWS account is now integrated into Serverless Framework Observability
✓ Serverless Framework Observability is enabled

✓ Service deployed to stack aws-python-flask-dynamodb-api-project-dev (139s)

dashboard: https://app.serverless.com/intiluna/apps/dh-geopython/aws-python-flask-dynamodb-api-project/dev/eu-west-1
endpoint: ANY - https://wrrrtnw2x1.execute-api.eu-west-1.amazonaws.com
functions:
  api: aws-python-flask-dynamodb-api-project-dev-api (19 MB)
(base) vant@Agile:~/Documents/datahack/cloud/proyecto3_cloud/aws-python-flask-dynamodb-api-project$ python preload.py
```

Resultado final

Resultado de ajustar html para llamar endpoints del chat

https://wrrrtnw2x1.execute-api.eu-west-1.amazonaws.com/anuncios/014

Detalle de anuncio

La sombra del viento

ID de Anuncio: 014

Precio: 10.99

Email de contacto: example14@example.com

Ver anuncios

Nuevo anuncio

Eliminar anuncio

Mensajes del chat

[2024-04-02T14:28:25.023034+00:00] 1: Muy buen libro

[2024-04-02T14:28:34.271796+00:00] 1: No lo pude entender

[2024-04-02T14:28:43.065805+00:00] 1: Rapidp

[2024-04-02T14:28:45.432372+00:00] 1: sdsdsdsd

[2024-04-02T14:28:48.302390+00:00] 1: dsddewwe

[2024-04-02T14:28:50.909292+00:00] 1: cdcdfertertr

[2024-04-02T14:28:53.524714+00:00] 1: wwerwewe

[2024-04-02T14:28:56.388143+00:00] 1: deddsxc

Escribe un nuevo mensaje...

Enviar

Bash

Se crea un fichero con los códigos requeridos para evitar múltiples operaciones manuales.

Fichero:

```
#!/bin/bash

# 1. Deploy de la aplicación en AWS utilizando Serverless
echo "Desplegando proyecto de Inti Luna en AWS usando serverless..."
serverless deploy

# 2 Esperar 3 segundos para que la URL de la API HTTP esté lista
echo "Esperando que la URL de la API HTTP esté disponible..."
sleep 3

# 3. Llamar a una función para rellenar la base de datos con registros de prueba
echo "Insertando registros de prueba en la base de datos..."
#api_url=$(sls info --verbose | grep HttpApiUrl | awk '{print $2}')
api_url=$(sls info --verbose | awk '/HttpApiUrl/{print $2}')
python preload_using_endpoint.py "$api_url"

# 4. Abriendo pagina web
echo "Abriendo la página web con la URL de la API HTTP: $api_url..."
echo "En caso de que no se abra automáticamente, abra manual en browser: $api_url..."
```

Se aplica permisos de ejecución:

```
>chmod +x deploy_and_fill_db.sh
```

Y se ejecuta:

```
> ./deploy_and_fill_db.sh
```

```

(base) vant@Agile: ~/Documents/datahack/cloud/proyecto3_cloud/aws-python-flask-dynamodb-api-project$ ./deploy_and_fill_db.sh
Desplegando proyecto de Inti Luna en AWS usando serverless...

Deploying aws-python-flask-dynamodb-api-project to stage dev (eu-west-1, "default" provider)
Python executable not found for "runtime": python3.9
Using default Python executable: python
Packaging Python WSGI handler...
Warning: Serverless Framework observability features do not support the following runtime: python3.9
✓ Your AWS account is now integrated into Serverless Framework Observability
✓ Serverless Framework Observability is enabled

✓ Service deployed to stack aws-python-flask-dynamodb-api-project-dev (125s)
  Follow link (ctrl+click)
dashboard: https://app.serverless.com/intiluna/apps/dh-geopython/aws-python-flask-dynamodb-api-project/dev/eu-west-1
endpoint: ANY - https://625yi0ili5.execute-api.eu-west-1.amazonaws.com
functions:
  api: aws-python-flask-dynamodb-api-project-dev-api (19 MB)
Esperando que la URL de la API HTTP esté disponible...
Insertando registros de prueba en la base de datos...
https://625yi0ili5.execute-api.eu-west-1.amazonaws.com/anuncio-nuevo
anuncio_id: 001, titulo: Don Quijote de la Mancha
{'anuncioId': '001', 'titulo': 'Don Quijote de la Mancha'}
anuncio_id: 002, titulo: Cien años de soledad
{'anuncioId': '002', 'titulo': 'Cien años de soledad'}
anuncio_id: 003, titulo: El amor en los tiempos del cólera
{'anuncioId': '003', 'titulo': 'El amor en los tiempos del cólera'}
anuncio_id: 004, titulo: La sombra del viento
{'anuncioId': '004', 'titulo': 'La sombra del viento'}
anuncio_id: 005, titulo: Crónica de una muerte anunciada
{'anuncioId': '005', 'titulo': 'Crónica de una muerte anunciada'}
anuncio_id: 006, titulo: El laberinto de los espíritus
{'anuncioId': '006', 'titulo': 'El laberinto de los espíritus'}

```

Recursos usados

Serverless

<https://www.serverless.com/framework/docs/getting-started/>
<https://www.serverless.com/framework/docs/tutorial>
<https://www.serverless.com/plugins/serverless-hooks>
<https://www.serverless.com/blog/cors-api-gateway-survival-guide>

DynamoDB - Python

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html>
<https://www.fernandomc.com/posts/ten-examples-of-getting-data-from-dynamodb-with-python-and-boto3/>
<https://dynobase.dev/dynamodb-python-with-boto3/#scan>