

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA



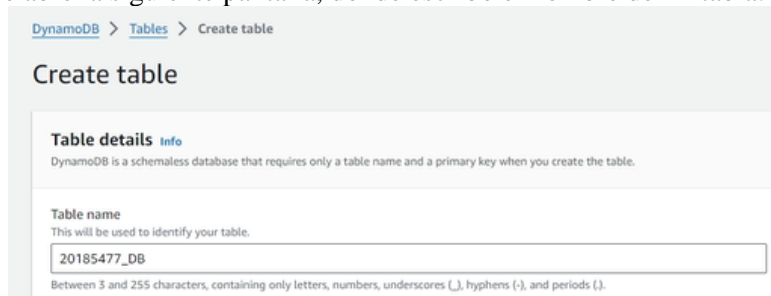
Arquitectura de Software AWS services

Autor: Kevin Javier Lázar Salomé (20185477)

Lima-2024
Perú

1. AWS DynamoDB:

- I. La creación de la base de datos en DynamoDB empieza, dando click en Create Table que abre la siguiente pantalla, donde escribo el nombre de mi tabla:



DynamoDB > Tables > Create table

Create table

Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

20185477_DB

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

- II. Luego se configura el parámetro partition key, el cual es el identificador de mi tabla, en este caso, le asigne id_sede con tipo de dato Number:



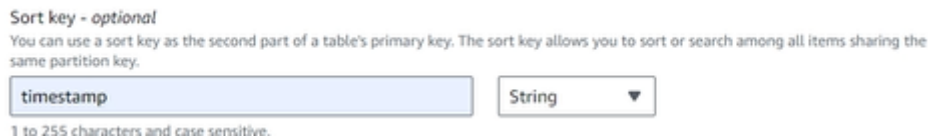
Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

id_sede Number

1 to 255 characters and case sensitive.

- III. Lo mismo con sort key, el cual me permite almacenar múltiples ítems con la misma partition key y los ordena, le asigne timestamp con tipo de dato String:



Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

timestamp String

1 to 255 characters and case sensitive.

- IV. Asimismo, configuro la tabla con configuraciones personalizadas, poniendo la clase de la tabla en estándar y On-demand para que use recursos cuando se le solicite.

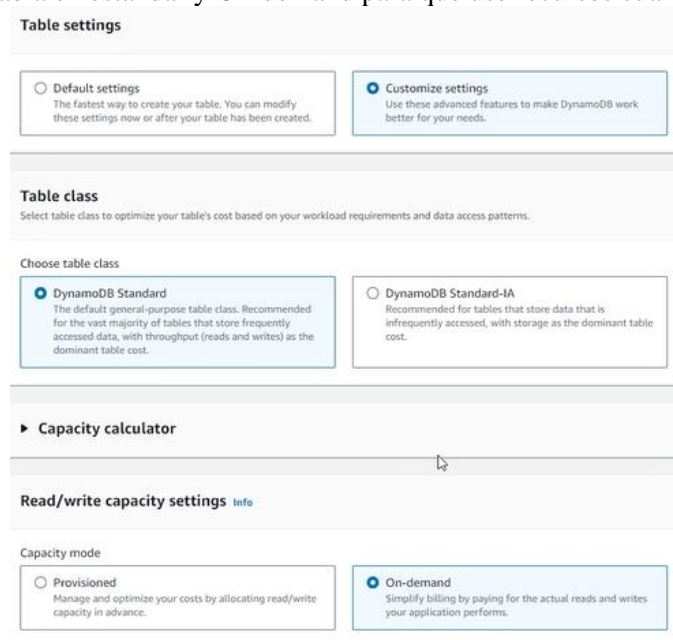


Table settings

☐ Default settings
The fastest way to create your table. You can modify these settings now or after your table has been created.

☒ Customize settings
Use these advanced features to make DynamoDB work better for your needs.

Table class

Select table class to optimize your table's cost based on your workload requirements and data access patterns.

Choose table class

☒ **DynamoDB Standard**
The default general-purpose table class. Recommended for the vast majority of tables that store frequently accessed data, with throughput (reads and writes) as the dominant table cost.

☐ **DynamoDB Standard-IA**
Recommended for tables that store data that is infrequently accessed, with storage as the dominant table cost.

► Capacity calculator

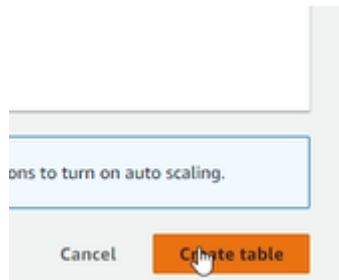
Read/write capacity settings [Info](#)

Capacity mode

☐ **Provisioned**
Manage and optimize your costs by allocating read/write capacity in advance.

☒ **On-demand**
Simplify billing by paying for the actual reads and writes your application performs.

- V. Para finalizar se da click en create Table:



- VI. Ahora, con la tabla ya creada se entra a la misma y se va al menú de Actions para darle click en Create Item. En esta parte, se le agrega los atributos (esquema de datos) que poseerá la tabla. Se asigna un nombre, valor y tipo de dato:

Create item Form JSON view

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

Attributes Add new attribute

Attribute name	Value	Type	
id_sede - Partition key	2	Number	
timestamp - Sort key	2024-05-20 19:23:13	String	
estado_robot	Operativo	String	Remove
cantidad_fallas	4	Number	Remove
posicion_critica	120	Number	Remove

Cancel Create item

- VII. Al final se le da click en Create Item y se muestra en la tabla los atributos ingresados con sus valores:

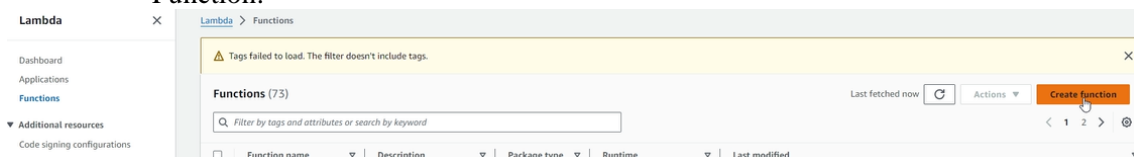
Items returned (1) Refresh Actions Create item

< 1 > Settings Grid

	id_sede (Number)	timestamp (String)	cantidad_fallas	estado_robot	posicion_critica
<input type="checkbox"/>	2	2024-05-20 19:23:13	4	Operativo	120

2. AWS Lambda:

- I. En el buscador de AWS, escribo Lambda y lo seleccionó para darle click en Create Function:



- II. Me aparece una ventana para configurarlo, donde le seleccionó Author from Scratch, le asignó un nombre de 20185477_POST, el lenguaje a usar es Node.js 16.x, la arquitectura es x86_64 y, en rol de ejecución, se escoge usar rol existente como AdministratorRole:

Function name
Enter a name that describes the purpose of your function.

20185477_POST

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 16.x

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ x86_64

☐ arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **Change default execution role**

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

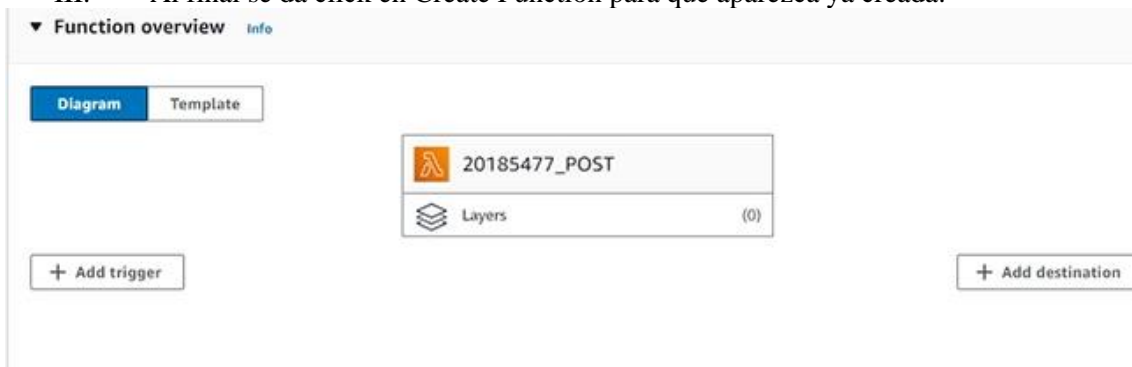
AdministratorRole

View the AdministratorRole role [on the IAM console](#).

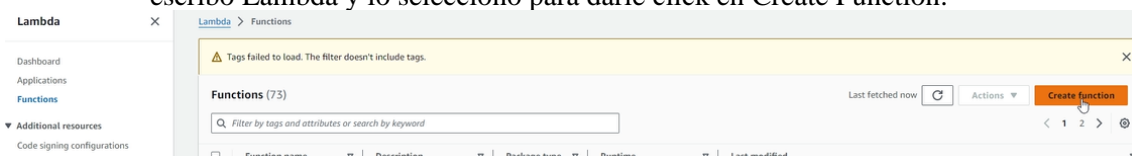
► **Advanced settings**

Cancel Create function

III. Al final se da click en Create Function para que aparezca ya creada:



IV. Los pasos se repiten para crear la Función GET. Primero, en el buscador de AWS, escribo Lambda y lo seleccioné para darle click en Create Function:



V. Me aparece una ventana para configurarlo, donde le seleccionó Author from Scratch, le asignó un nombre de 20185477_GET, el lenguaje a usar es Node.js 16.x, la arquitectura es x86_64 y, en rol de ejecución, se escoge usar rol existente como AdministratorRole:

Function name
Enter a name that describes the purpose of your function.

20185477_GET

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 16.x

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ x86_64

☐ arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.


AdministratorRole


[View the AdministratorRole role](#) on the IAM console.


VI. Al final se da click en Create Function para que aparezca ya creada:
20185477_GET

▼ Function overview [Info](#) Ex

Diagram **Template**

 20185477_GET

 Layers (0)


 API Gateway (2)

[+ Add trigger](#)

[+ Add destination](#)

Description
-

Last modified
21 hours ago

Function ARN
 arn:aws:lambda:

Function URL [Info](#)

- Configuración y ejecución de prueba de eventos.

20185477_POST

- Para esta función, se le da click en Test y se crea un nuevo evento de Test, se le asigna un nombre y se le pone Private con un evento Json como se muestra a continuación:

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event

☐ Edit saved event

Event name

Test_lambda_20185477

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Event JSON

Format JSON

```
1 {  
2   "id_sede": 6,  
3   "estado_robot": "Operativo",  
4   "timestamp": "2024-02-40 09:23:16",  
5   "cantidad_fallas": 10,  
6   "posicion_critica": 10  
7 }
```

Cancel

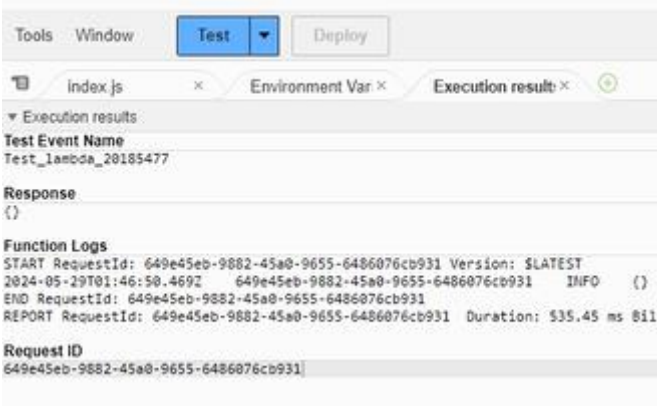
Invoke

Save

- II. Al final se le da click en Save y con eso estaría listo para hacerse pruebas en el mismo Lambda. Le agrego el Code Source para conectar el Lambda con el Dynamo, tal que escriba los valores ingresados y los registre en la tabla:

```
const AWS = require('aws-sdk');  
const dynamodb = new AWS.DynamoDB({region: 'eu-central-1', apiVersion: '2012-08-10'});  
  
exports.handler = (event, context, callback) => {  
  const params = {  
    Item: {  
      "id_sede": {  
        "N": "" + event.id_sede  
      },  
      "estado_robot": {  
        "S": "" + event.estado_robot  
      },  
      "timestamp": {  
        "S": "" + event.timestamp  
      },  
      "cantidad_fallas": {  
        "N": "" + event.cantidad_fallas  
      },  
      "posicion_critica": {  
        "N": "" + event.posicion_critica  
      }  
    },  
    TableName: "20185477_DB"  
  }  
  
  dynamodb.putItem(params, function(err, data) {  
    if (!err) {  
      console.log(data);  
      callback(null, data);  
    } else {  
      console.log(err);  
      callback(err);  
    }  
  })  
}
```

III. Luego, le doy click a Deploy y le doy click a Test para ejecutar la prueba, dándome como resultado esto:



IV. Se ve que se ha agregado a la tabla la información:

<input type="checkbox"/>	id_sede (Number)	timestamp (String)	cantidad_fallas	estado_robot	posicion_critica
<input type="checkbox"/>	3	2024-01-40 09:23:13	3	Operativo	420
<input type="checkbox"/>	2	2024-05-20 19:23:13	4	Operativo	120
<input type="checkbox"/>	4	2024-12-22 12:23:14	0	Inoperativo	0
<input type="checkbox"/>	6	2024-02-40 09:23:16	10	Operativo	10
<input type="checkbox"/>	1	2024-02-27 14:23:19	5	Operativo	53

20185477_GET

V. Para esta función, se le da click en Test y se crea un nuevo evento de Test, se le asigna un nombre y se le pone Private con un evento Json como se muestra a continuación:

Test event action

☒ Create new event ☐ Edit saved event

Event name

TEST_GET_20185477

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

TEST_GET_20185477

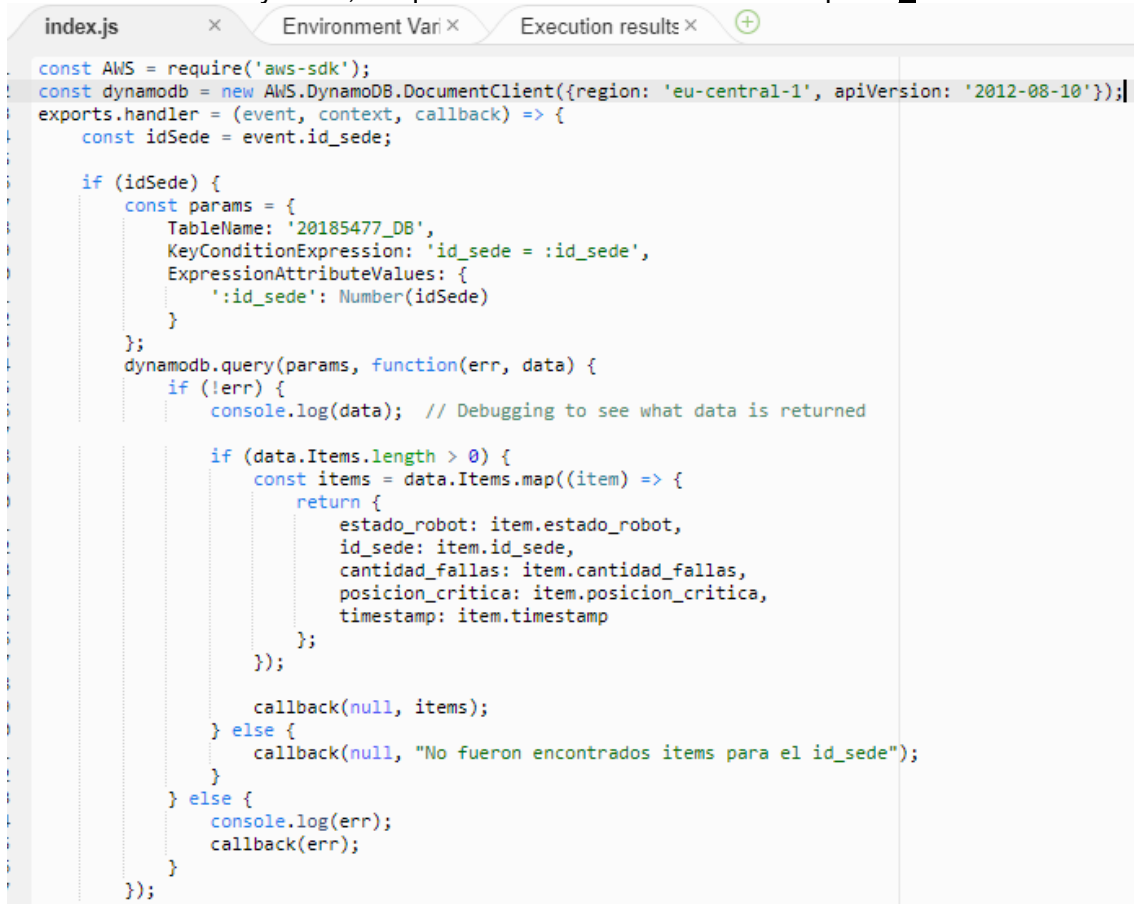
Event JSON

Format JSON

```
1 {  
2   "id_sede": 1  
3 }
```

VI. Al final se le da click en Save y con eso estaría listo para hacerse pruebas en el mismo Lambda. Le agrego el Code Source para conectar el Lambda

con el Dynamo, tal que me devuelva la información por id_sede:



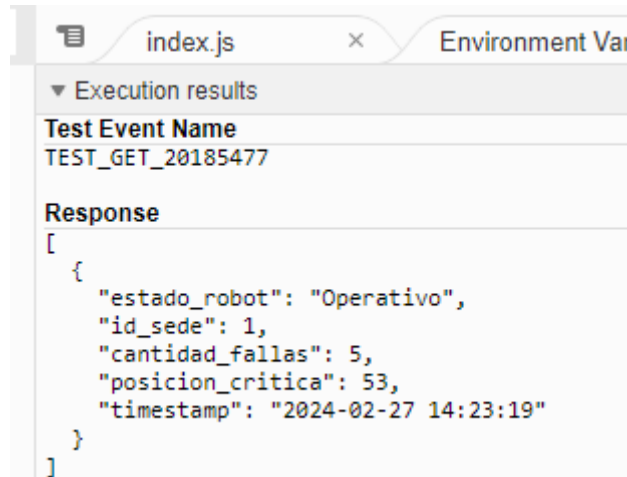
```
index.js x Environment Vari Execution results +
const AWS = require('aws-sdk');
const dynamodb = new AWS.DynamoDB.DocumentClient({region: 'eu-central-1', apiVersion: '2012-08-10'});
exports.handler = (event, context, callback) => {
  const idSede = event.id_sede;

  if (idSede) {
    const params = {
      TableName: '20185477_DB',
      KeyConditionExpression: 'id_sede = :id_sede',
      ExpressionAttributeValues: {
        ':id_sede': Number(idSede)
      }
    };
    dynamodb.query(params, function(err, data) {
      if (!err) {
        console.log(data); // Debugging to see what data is returned

        if (data.Items.length > 0) {
          const items = data.Items.map((item) => {
            return {
              estado_robot: item.estado_robot,
              id_sede: item.id_sede,
              cantidad_fallas: item.cantidad_fallas,
              posicion_critica: item.posicion_critica,
              timestamp: item.timestamp
            };
          });

          callback(null, items);
        } else {
          callback(null, "No fueron encontrados items para el id_sede");
        }
      } else {
        console.log(err);
        callback(err);
      }
    });
  }
};
```

VII. Luego, le doy click a Deploy y le doy click a Test para ejecutar la prueba, dándome como resultado esto:



```
index.js x Environment Vari
Execution results
Test Event Name
TEST_GET_20185477
Response
[
  {
    "estado_robot": "Operativo",
    "id_sede": 1,
    "cantidad_fallas": 5,
    "posicion_critica": 53,
    "timestamp": "2024-02-27 14:23:19"
  }
]
```

3. AWS API Gateway:

I. Se empieza, buscando Api Gateway en el buscador y se le da click en Create API:



II. De todas las opciones, se selecciona el REST API y se da click en build:

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:
Lambda, HTTP, AWS Services

Import

Build

III. Se le asigna un nombre, en mi caso, le asigne compare_yourself_20185477, una descripción y un tipo de endpoint Regional:

Create REST API

API details

☒ New API
Create a new REST API.

☐ Clone existing API
Create a copy of an API in this AWS account.

☐ Import API
Import an API from an OpenAPI definition.

☐ Example API
Learn about API Gateway with an example API.

API name

compare_yourself_20185477

Description - optional

Permite a los usuarios compararse entre ellos mismos

API endpoint type
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

Regional

Cancel

Create API

IV. Se le da click en Create Api y se mostrará en la tabla de APIs:

<input type="radio"/>	compare_yourself_20185477	Permite a los usuarios compararse entre ellos mismos	rhmsvxmu64	REST	Regional	2024-05-28

- Configuración de los recursos y métodos asociados a la(s) funcion(es) Lambda.

I. Se selecciona la API creada y se le da click en Create Resource:

Create resource

Resource details

Path
/

Resource ID
9etx96qpjj

II. Pongo el nombre al recurso como compare-yourself y le activo la casilla para crear el método Options y doy click en Create Resource:

Create resource

Resource details

☐ Proxy resource [Info](#)
 Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

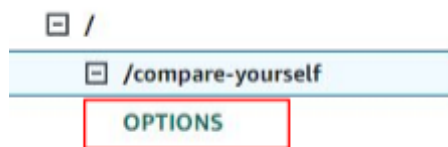
Resource path

Resource name

☒ CORS (Cross Origin Resource Sharing) [Info](#)
 Create an OPTIONS method that allows all origins, all methods, and several common headers.

[Cancel](#)
[Create resource](#)

III. Dando el siguiente resultado:



IV. Luego, con el recurso creado se le crea un método en el botón Create Method:

Methods (3)

[Delete](#)
[Create method](#)

Method type	Integration type	Authorization	API key
20185477_POST			

20185477_POST

Throttle

Copy ARN

Actions

Function overview

Diagram

Template

20185477_POST

Layers (0)

+ Add trigger

+ Add destination

Export to Application Composer

Download

Description

Last modified 32 minutes ago


Function ARN


Function URL


VI. En esta ventana se muestra la configuración completa:


Method type
POST


Integration type

☒ **Lambda function**
Integrate your API with a Lambda function.


☐ **HTTP**
Integrate with an existing HTTP endpoint.


☐ **Mock**
Generate a response based on API Gateway mappings and transformations.


☐ **AWS service**
Integrate with an AWS Service.


☐ **VPC link**
Integrate with a resource that isn't accessible over the public internet.


☐ **Lambda proxy integration**
Send the request body to the Lambda function.

Lambda function
Provide the Lambda function ARN.
eu-central-1

Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

☒ **Default timeout**
The default timeout is 29 seconds.

VII. Al final se le da click en Create Method para que se vea este resultado:

Resources

Create resource

/compare-yourself - POST - Method execution

ARN: am:aws:execute-api:eu-central-1:003829283967:rhmsvsmu64:POST/compare-yourself

Resource ID: cyo67x

Client → Method request → Integration request →

← Method response ← Integration response ←

Update documentation

VIII. Luego, se ingresa a Integration request para asignarle un Mapping Template al darle click en Edit:

Method request | **Integration request** | Integration response | Method response

Integration request settings Edit

Integration type **Lambda**

Region **eu-central-1**

Lambda proxy integration **False**

Lambda function **cv-store-data**

- IX. Se le cambia a cuando no hay templates definidos y se coloca para tipo de contrnido application/json y con un cuerpo de plantilla igual al que se muestra:

☒ When there are no templates defined (recommended)

☐ When no template matches the request content-type header

☐ Never

► URL path parameters

► URL query string parameters

► URL request headers parameters

▼ Mapping templates

Content type

application/json

Remove

Generate template

Template body

```
1 {  
2  
3   "id_sede": $input.json('$.id_sede'),  
4   "estado_robot": $input.json('$.estado_robot'),  
5   "timestamp": $input.json('$.timestamp'),  
6   "cantidad_fallas": $input.json('$.cantidad_fallas'),  
7   "posicion_critica": $input.json('$.posicion_critica')  
8  
9 }
```

- X. Al darle Save, ya estaría listo el método para ser usado. Se le agrega un validador de data para que siga una estructura con un modelo al ir a models y darle click en Create Model:

Models (2)

Use models to define the format for the body of different requests and responses used by your API.

Delete Edit Update documentation Create model

- XI. Se procede a configurarlo como se muestra en la siguiente imagen y se agrega un esquema modelo que tiene las propiedades del modelo:

Create model

Model details

Name

The model name must have 1-1024 characters. Valid characters: A-Z, a-z, and 0-9.

Content type

Description - optional

Model schema

```

4
5   "title": "CompareData",
6
7   "type": "object",
8
9   "properties": {
10
11     "id_sede": {
12       "type": "integer"
13     },
14   },
15
16
17   "estado_robot": {
18

```

Cancel Create

XII. Le doy a Create para tener este resultado:

Models (3) Delete Edit Update documentation Create model

Use models to define the format for the body of different requests and responses used by your API.

Name	Content type	Description
CompareData	application/json	

XIII. Finalmente, me voy al método Post creado y busco method request para darle Edit y lo configuro como la siguiente imagen, teniendo el método POST listo:

Method request settings

Authorization

Request validator

☐ API key required

Operation name - optional

► URL query string parameters

► HTTP request headers

▼ Request body

Content type

Model
 Remove

Add model

Cancel Save

XIV. Para crear el método GET, primero se crea un recurso en compare-yourself, dando click en Create resource con el nombre {id_sede}:

Create resource

Resource details

☐ Proxy resource **Info**
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy*}.

Resource path:

Resource name:

☒ CORS (Cross Origin Resource Sharing) **Info**
Create an OPTIONS method that allows all origins, all methods, and several common headers.

Cancel

- XV. Luego, se crea un método, dando click a Create Method y aparecerá una ventana, donde se escoge el tipo de método, en este caso, GET, el tipo de integración Lambda, el lugar junto al ARN obtenido de:

20185477_GET

Layers (0)

(2)

+ Add destination

Description

Last modified: 22 hours ago

Function ARN: `arn:aws:lambda:eu-central-1:003829283967:function:20185477_GET`

- XVI. En esta ventana se muestra la configuración completa:

Method type

GET

Integration type

☒ Lambda function
Integrate your API with a Lambda function.

☐ HTTP
Integrate with an existing HTTP endpoint.

☐ Mock
Generate a response based on API Gateway mappings and transformations.

☐ AWS service
Integrate with an AWS Service.

☐ VPC link
Integrate with a resource that isn't accessible over the public internet.

☐ Lambda proxy integration

Send the request to: `arn:aws:lambda:eu-central-1:003829283967:function:20185477_GET`

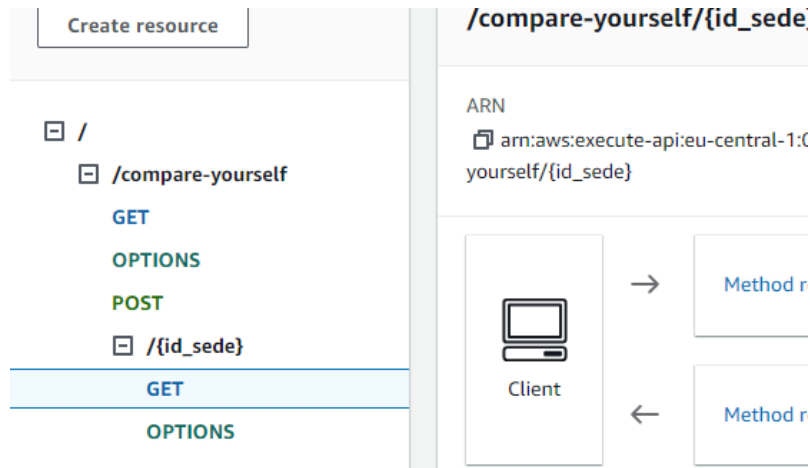
Lambda function: `arn:aws:lambda:eu-central-1:003829283967:function:20185477_GET`

Provide the Lambda function: `eu-central-1`

Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

☒ Default timeout
The default timeout is 29 seconds.

- XVII. Al final se le da click en Create Method para que se vea este resultado:



XVIII. Después de ello, se va a integration request y a Edit para agregar un Mapping Template:

The screenshot shows the AWS API Gateway console's Mapping Template editor. The `Content type` is set to `application/json`. The `Generate template` dropdown is empty. The `Template body` is a text area containing the following JSON:

```

1 {
2
3   "id_sede": "$input.params('id_sede')"
4
5 }

```

At the bottom of the editor, there is a link to [Learn more](#) and a button labeled `Add mapping template`. At the bottom right of the console, there are `Cancel` and `Save` buttons.

Finalmente, le doy a save y el método GET estaría listo.

- Creación de un Stage para producción.
- I. Para convertir mi programa a tipo público se le da click en Deploy API y se le crea un Stage, donde mi API será desplegada con el nombre Production y una descripción:

Deploy API

Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. [Learn more](#)

Stage

New stage

Stage name

Production

A new stage will be created with the default settings. Edit your stage settings on the **Stage** page.

Deployment description

Despliegue de producción

Cancel

Deploy

- II. Se le da click en Deploy y se ve al Stage Production como una de las etapas de la API desplegada:

Stages

Production

/

/compare-yourself

GET

OPTIONS

POST

/ (id_sede)

GET

OPTIONS

Stage details Info

Stage name

Production

Cache cluster

Inactive

Default method-level caching

Inactive

Invoke URL

https://rhmsvxxmu64.execute-api.eu-central-1.amazonaws.com/Production

Rate

Info

Burst

Info

Web ACL

-

Client certificate

-

Edit

4. Pruebas de concepto e integración:

- I. Se realizó las pruebas en Postman para el método POST, creando una nueva colección y copiando el URL correspondiente:

Stages

Production

/

/compare-yourself

GET

OPTIONS

POST

/ (id_sede)

Method overrides

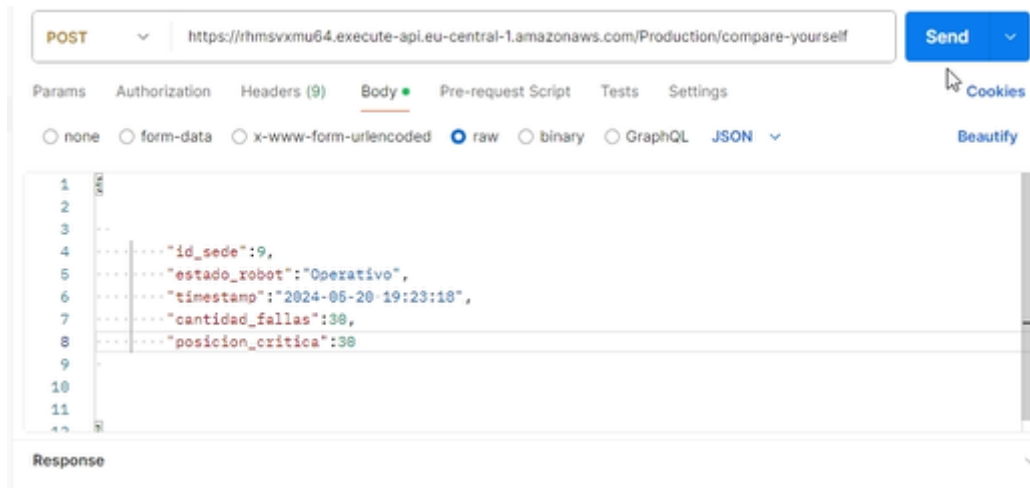
By default, methods inherit stage-level settings. To customize settings for a method, configure method overrides.

This method inherits its settings from the 'Production' stage.

Copied

https://rhmsvxxmu64.execute-api.eu-central-1.amazonaws.com/Production/compare-yourself

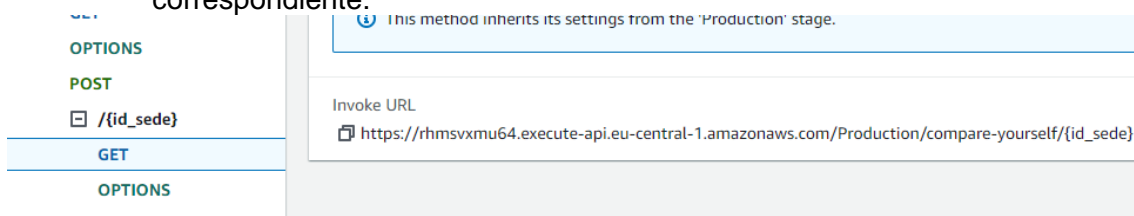
II. Lo pego, seleccionó el método POST y le agrego un Body para darle SEND:



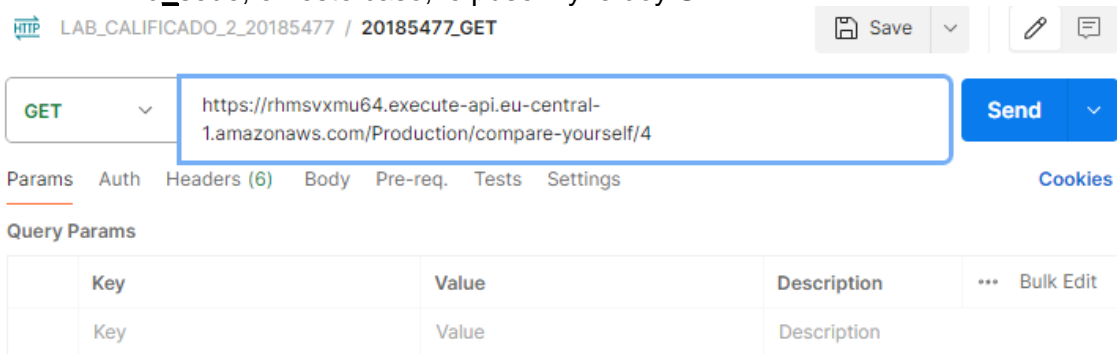
III. Sale el resultado esperado, pues se ve el registro:

<input type="checkbox"/>	id_sede (Number)	timestamp (String)	cantidad_fallas	estado_robot	posicion_critica
<input type="checkbox"/>	3	2024-01-40 09:23:13	3	Operativo	420
<input type="checkbox"/>	2	2024-05-20 19:23:13	4	Operativo	120
<input type="checkbox"/>	9	2024-05-20 19:23:18	30	Operativo	30
<input type="checkbox"/>	4	2024-12-22 12:23:14	0	Inoperativo	0
<input type="checkbox"/>	6	2024-02-40 09:23:16	10	Operativo	10
<input type="checkbox"/>	1	2024-02-27 14:23:19	5	Operativo	53
<input type="checkbox"/>	5	2024-05-25 19:23:13	20	Operativo	20

IV. Se realizó las pruebas en Postman para el método GET al copiar el URL correspondiente:



V. Lo pego, seleccionó el método GET y cambio el {id_sede} por el valor de id_sede, en este caso, le puse 4 y le doy SEND:



VI. Obteniendo como resultado todos los datos que tienen el id_sede seleccionado:

Body ▾

 200 C

Pretty

Raw

Preview

Visualize

JSON ▾



```
1  [
2    {
3      "estado_robot": "Operativo",
4      "id_sede": 4,
5      "cantidad_fallas": 200,
6      "posicion_critica": 200,
7      "timestamp": "2024-05-20 19:23:15"
8    },
9    {
10     "estado_robot": "Inoperativo",
11     "id_sede": 4,
12     "cantidad_fallas": 0,
13     "posicion_critica": 0,
14     "timestamp": "2024-12-22 12:23:14"
15   }
16 ]
```