



UNIVERSIDAD TÉCNICA
FEDERICO SANTA MARÍA

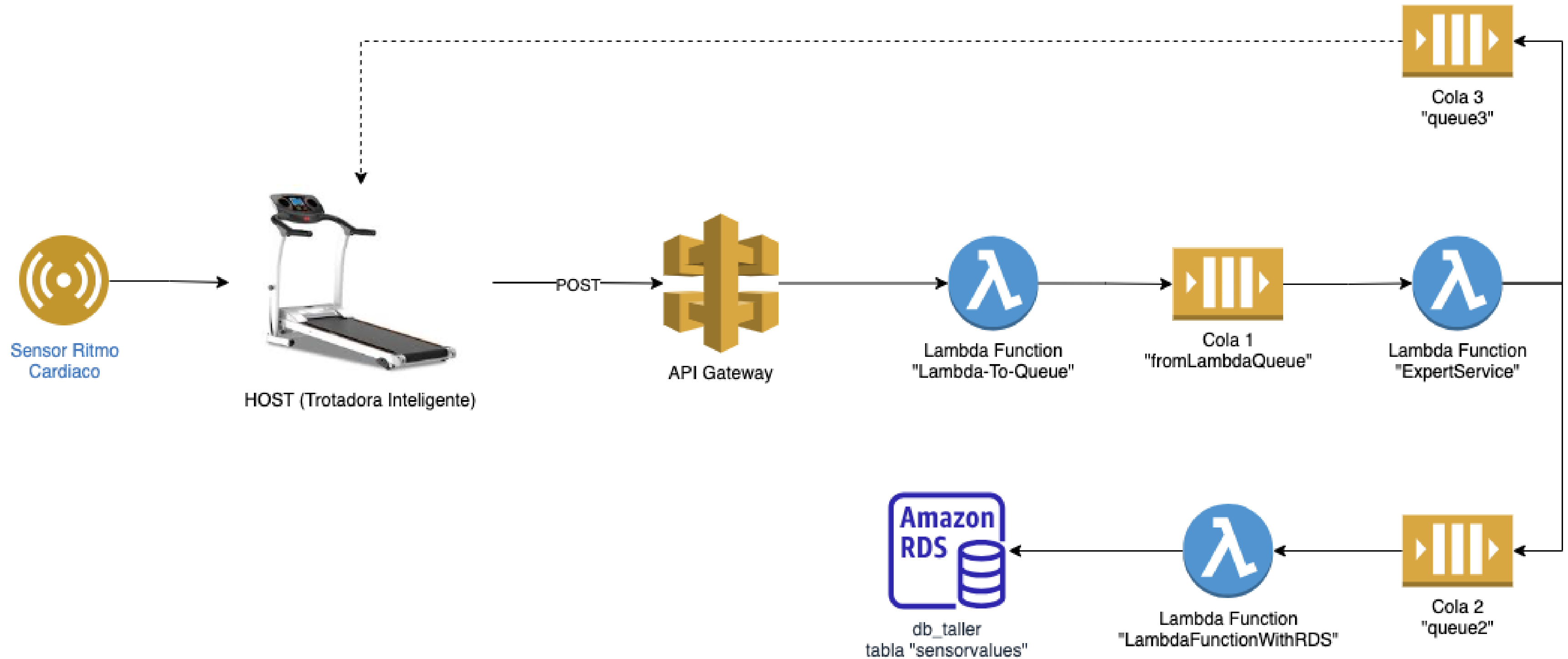
DEPARTAMENTO
DE INFORMÁTICA

TROTADORA INTELIGENTE

ILI140 TALLER DE PROGRAMACIÓN

Nicolás Araya Urrutia

ARQUITECTURA CLOUD (AWS)

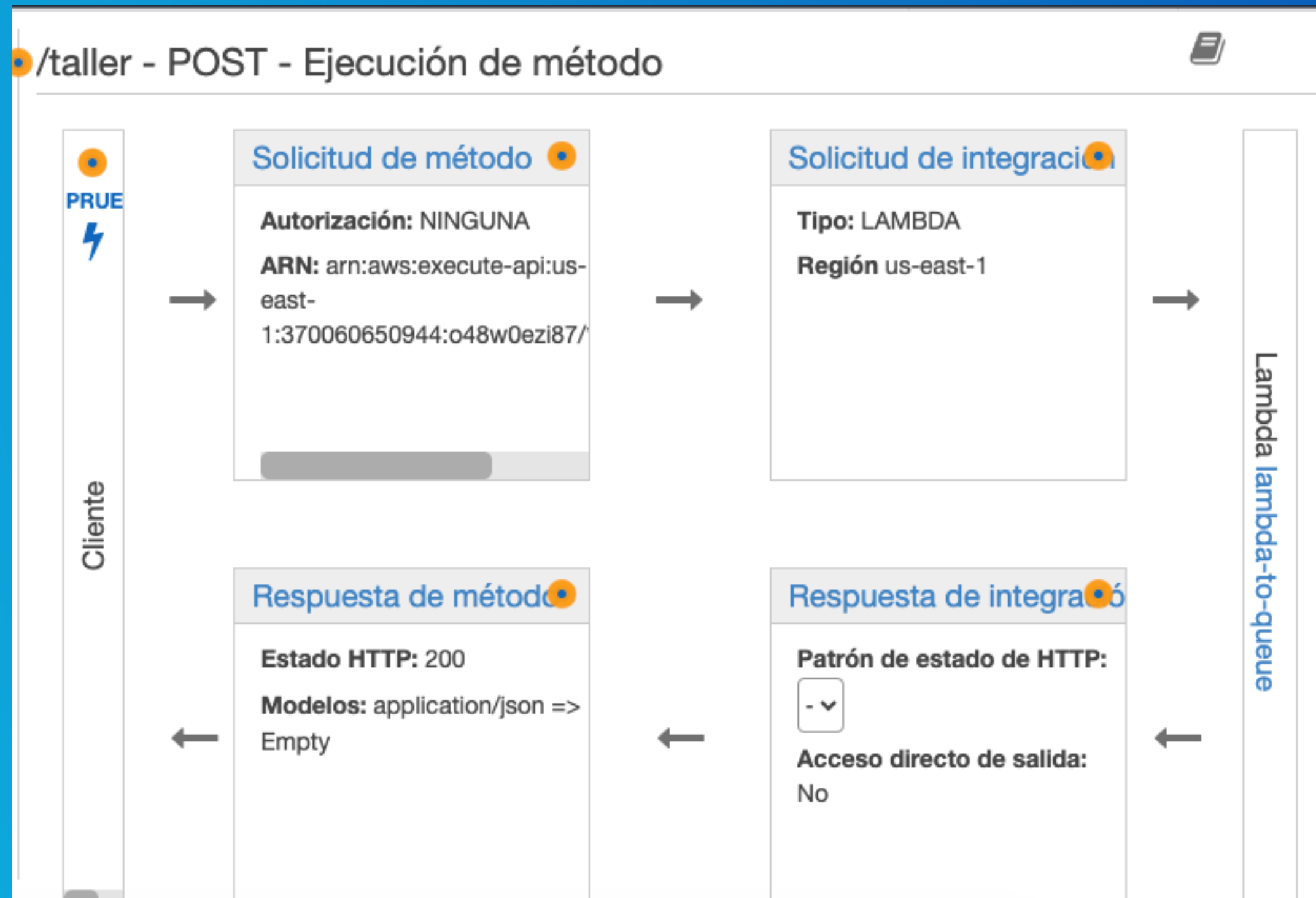


HOST: TROTADORA INTELIGENTE



```
fetch('https://o48w0ezi87.execute-api.us-east-1.amazonaws.com/desarrollo/taller', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(payload)
})
.then(response => {
  if (response.ok) {
    // La llamada a la API fue exitosa
    console.log('Número aleatorio enviado correctamente');
  } else {
    // La llamada a la API falló
    console.log('Error al enviar el número aleatorio');
  }
})
.catch(error => {
  // Error en la llamada a la API
  console.log('Error en la llamada a la API:', error);
});
};
```

API GATEWAY



FUNCTION: "LAMBDA-TO-QUEUE"

```
lambda_function × (+)
1 import json
2 import boto3
3
4 def lambda_handler(event, context): #required
5     value = event['body-json']
6     sqs = boto3.client('sqs') #client is required to interact with
7     sqs.send_message(
8         QueueUrl="https://sqs.us-east-1.amazonaws.com/370060650944/fromLambdaQueue",
9         MessageBody=json.dumps(value)
10    )
11
12    return {
13        'statusCode': 200,
14        'body': json.dumps(value)
15    }
```

COLA 1: "FROMLAMBDAQUEUE"

Amazon SQS > Colas > fromLambdaQueue

fromLambdaQueue

Editar

Eliminar

Purgar

Enviar y recibir mensajes

Iniciar el redireccionamiento de la DLQ

Detalles Información

Nombre

fromLambdaQueue

Tipo

Estándar

ARN

arn:aws:sqs:us-east-1:370060650944:fromLambdaQueue

Cifrado

Clave de Amazon SQS (SSE-SQS)

URL

https://sqs.us-east-1.amazonaws.com/370060650944/fromLambdaQueue





Cola de mensajes fallidos


-

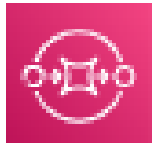
► Más

FUNCTION: "EXPERT-SERVICE"

Desencadenadores (1) Información

 *Buscar desencadenadores*

<input type="checkbox"/>	Desencadenador
<input type="checkbox"/>	<div>SQS: fromLambdaQueue arn:aws:sqs:us-east-1:370060650944:fromLambdaQueue state: Enabled ▶ Detalles</div>

```
lambda_function × Execution results × +
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     message = event['Records'][0]['body']
6     data = json.loads(message)
7     value = float(data['value'])
8
9     inclination = value * 2
10    velocity = value / 2
11
12    dicc_export = {
13        "value": str(value),
14        "inclination": str(inclination),
15        "velocity": str(velocity)
16    }
17
18    sqs = boto3.client('sqs')
19
20    sqs.send_message(
21        QueueUrl="https://sqs.us-east-1.amazonaws.com/370060650944/queue2",
22        MessageBody=json.dumps(dicc_export)
23    )
24    sqs.send_message(
25        QueueUrl="https://sqs.us-east-1.amazonaws.com/370060650944/queue3",
26        MessageBody=json.dumps(dicc_export)
27    )
28
29    return {
30        'statusCode': 200,
31        'body': json.dumps(dicc_export)
32    }
33
```

COLA 2 Y COLA 3

Amazon SQS > Colas > queue2

queue2

Editar

Eliminar

Purgar

Enviar y recibir mensajes

Iniciar el redireccionamiento de la DLQ

Detalles Información

Nombre queue2	Tipo Estándar	ARN arn:aws:sqs:us-east-1:370060650944:queue2
Cifrado Clave de Amazon SQS (SSE-SQS)	URL https://sqs.us-east-1.amazonaws.com/370060650944/queue2	Cola de mensajes fallidos -

Más

Amazon SQS > Colas > queue3

queue3

Editar

Eliminar

Purgar

Enviar y recibir mensajes

Iniciar el redireccionamiento de la DLQ


Detalles Información


Nombre queue3	Tipo Estándar	ARN arn:aws:sqs:us-east-1:370060650944:queue3
Cifrado Clave de Amazon SQS (SSE-SQS)	URL https://sqs.us-east-1.amazonaws.com/370060650944/queue3	Cola de mensajes fallidos -

Más

FUNCTION: "LAMBDA-FUNCTION-WITH-RDS"

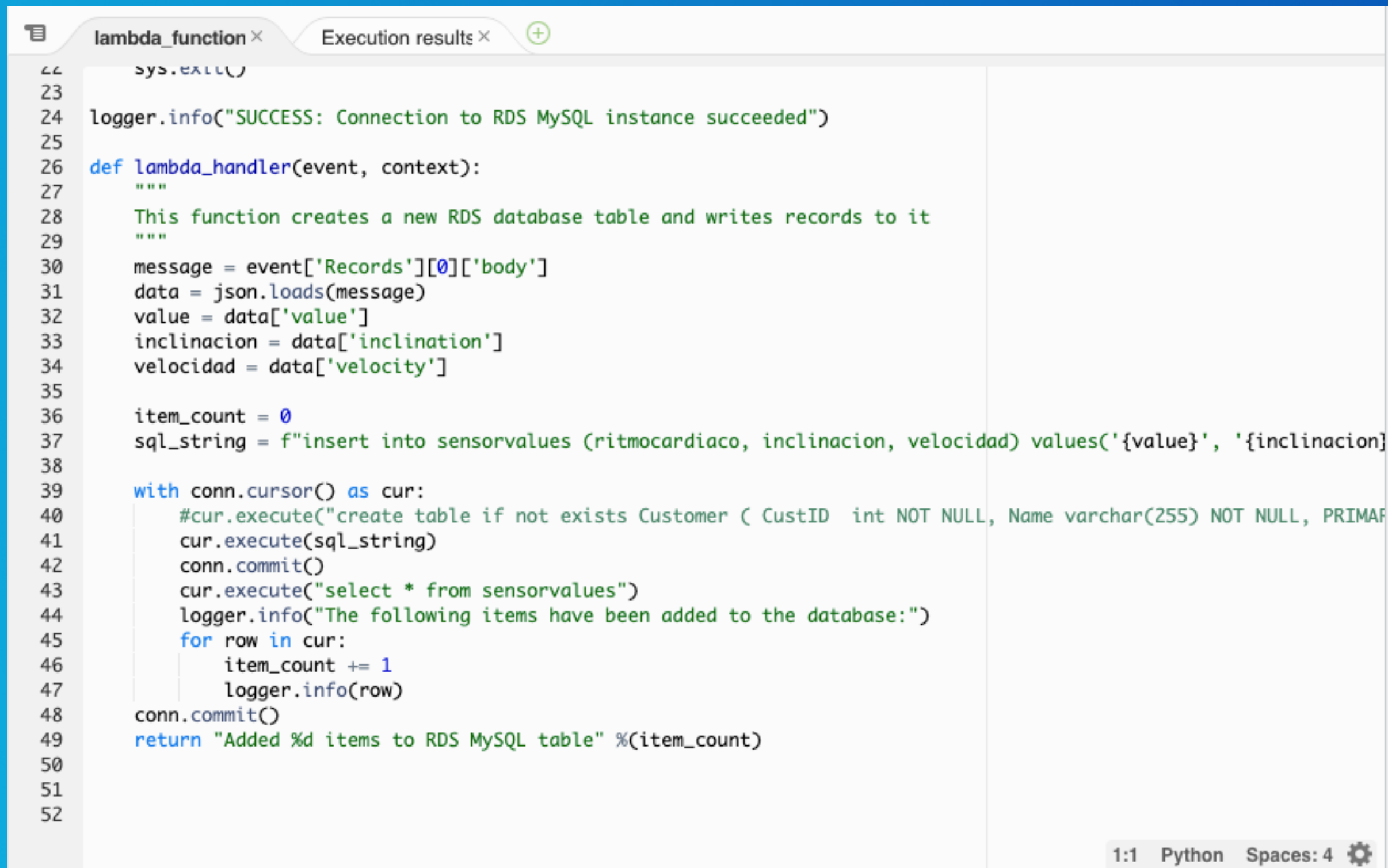
Desencadenadores (1) [Información](#)

 [Corregir errores](#) [Editar](#) [Eliminar](#)

<input type="checkbox"/>	Desencadenador
<input type="checkbox"/>	<div> SQS: queue2 arn:aws:sqs:us-east-1:370060650944:queue2 state: Enabled ► Detalles</div>

```
lambda_function x Execution results x
1 import sys
2 import logging
3 import pymysql
4 import json
5
6 # rds settings
7 rds_host = "tallerprogra2.cunetuyifwaj.us-east-1.rds.amazonaws.com"
8 user_name = "admin"
9 password = "admin123456789"
10 db_name = "db_taller"
11
12 logger = logging.getLogger()
13 logger.setLevel(logging.INFO)
14
15 # create the database connection outside of the handler to allow connections to be
16 # re-used by subsequent function invocations.
17 try:
18     conn = pymysql.connect(host=rds_host, user=user_name, passwd=password, db=db_name, connect_timeout=5)
19 except pymysql.MySQLError as e:
20     logger.error("ERROR: Unexpected error: Could not connect to MySQL instance.")
21     logger.error(e)
22     sys.exit()
23
24 logger.info("SUCCESS: Connection to RDS MySQL instance succeeded")
25
```

FUNCTION: "LAMBDA-FUNCTION-WITH-RDS"



```
22 sys.exit()
23
24 logger.info("SUCCESS: Connection to RDS MySQL instance succeeded")
25
26 def lambda_handler(event, context):
27     """
28     This function creates a new RDS database table and writes records to it
29     """
30     message = event['Records'][0]['body']
31     data = json.loads(message)
32     value = data['value']
33     inclinacion = data['inclination']
34     velocidad = data['velocity']
35
36     item_count = 0
37     sql_string = f"insert into sensorvalues (ritmocardiaco, inclinacion, velocidad) values('{value}', '{inclinacion}')"
38
39     with conn.cursor() as cur:
40         #cur.execute("create table if not exists Customer ( CustID int NOT NULL, Name varchar(255) NOT NULL, PRIMARY KEY (CustID))")
41         cur.execute(sql_string)
42         conn.commit()
43         cur.execute("select * from sensorvalues")
44         logger.info("The following items have been added to the database:")
45         for row in cur:
46             item_count += 1
47             logger.info(row)
48     conn.commit()
49     return "Added %d items to RDS MySQL table" %(item_count)
50
51
52
```

1:1 Python Spaces: 4

AMAZON RDS: TALLER-PROGRA-2

RDS > Bases de datos > tallerprogra2

tallerprogra2

ModificarAcciones ▼

Resumen

Identificador de base de datos tallerprogra2	CPU <div>2.13%</div>	Estado ✔ Disponible	Clase db.t3.micro
Rol Instancia	Actividad actual <div>2 Conexiones</div>	Motor MySQL Community	Región y AZ us-east-1a

BD MYSQL: DB_TALLER

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'db_taller' expanded, showing 'Tables' (Customer, sensorvalues) and 'Views', 'Stored Procedures', and 'Functions'. The 'Query 1' tab is active, showing a SQL query: `SELECT * FROM `db_taller`.`sensorvalues`;`. The 'Result Grid' is displayed below the query, showing 10 rows of data. The columns are 'id', 'ritmocardiaco', 'inclinacion', and 'velocidad'. The bottom status bar indicates 'Table: sensorvalues' and 'Columns:'. The right sidebar contains icons for 'Result Grid', 'Form Editor', and 'Field Types'.

MySQL Workbench

AWS taller progra 2

Administration Schemas Query 1

SCHEMAS

Filter objects

db_taller

Tables

Customer

sensorvalues

Views

Stored Procedures

Functions

sys

100% 33:2

Result Grid Filter Rows: Search Edit: Export/Import

id	ritmocardiaco	inclinacion	velocidad
55	87	45	10
56	90	45	10
57	91	45	10
58	65	45	10
59	33	45	10
60	41	45	10
61	46	45	10
62	56	45	10
63	70	45	10
64	79	45	10
65	60.0	120.0	30.0
NULL	NULL	NULL	NULL

Table: sensorvalues

Columns:

Result Grid

Form Editor

Field Types

The slide features a light blue gradient background. In the top-left and bottom-right corners, there are intricate, abstract line art patterns in a darker blue. These patterns consist of a dense network of interconnected lines, dots, and circular motifs, resembling a complex circuit board or a molecular structure. The word "DEMO" is centered in the middle of the slide in a bold, dark blue, sans-serif font.

DEMO