UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

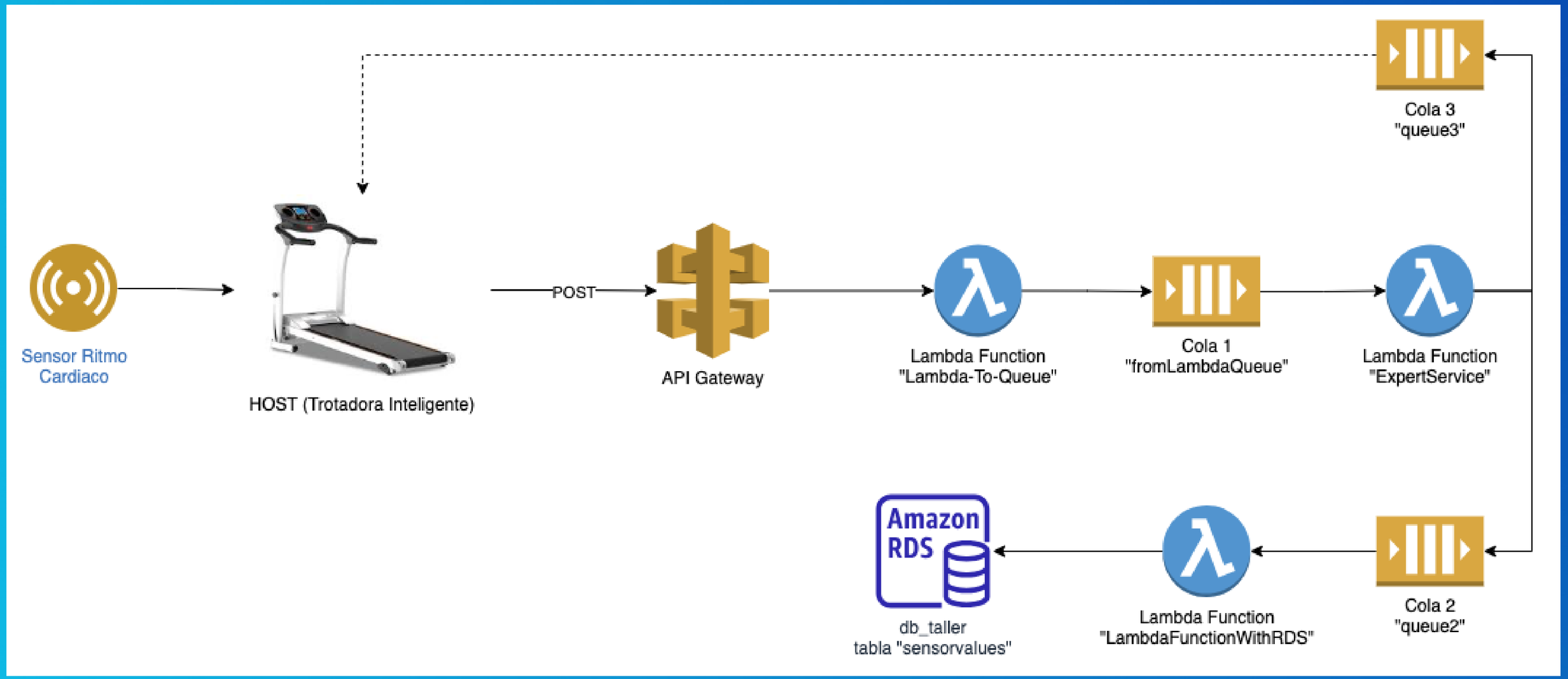
DEPARTAMENTO
DE INFORMÁTICA

TROTADORA INTELIGENTE

ILI140 TALLER DE PROGRAMACIÓN

Nicolás Araya Urrutia

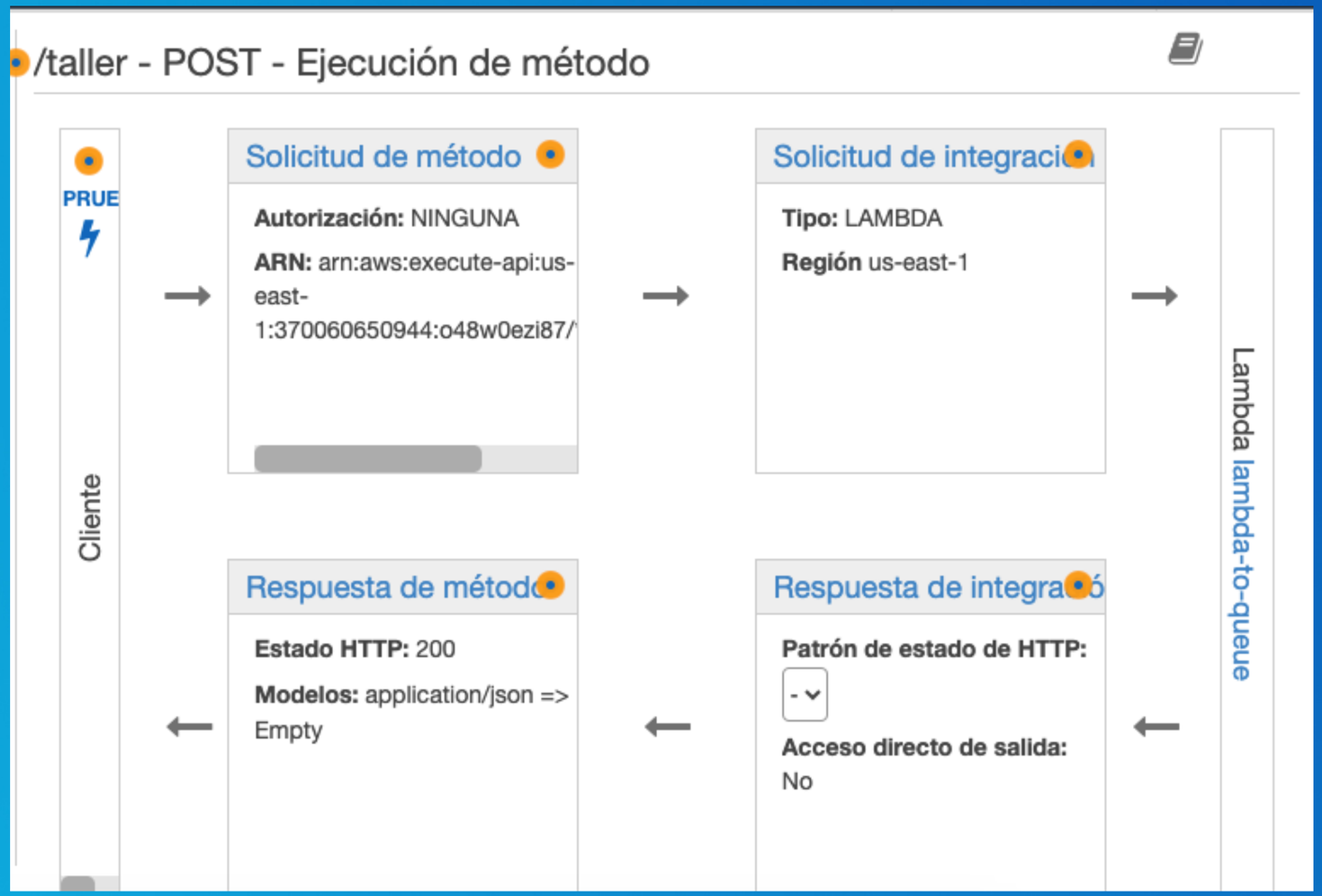
ARQUITECTURA CLOUD (AWS)



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


/taller - POST - Ejecución de método
PRUE
Cliente
Solicitud de método
Autorización: NINGUNA
ARN: arn:aws:execute-api:us-east-1:370060650944:o48w0ezi87/
Solicitud de integraci
Tipo: LAMBDA
Región us-east-1
Lambda lambda-to-queue
Respuesta de métod
Estado HTTP: 200
Modelos: application/json => Empty
Respuesta de integra
Patrón de estado de HTTP:
-
Acceso directo de salida:
No

FUNCTION: "LAMBDA-TO-QUEUE"

lambda_function

```python
import json
import boto3

def lambda_handler(event, context):  #required
    value = event['body-json']
    sqs = boto3.client('sqs')  #client is required to interact with
    sqs.send_message(
        QueueUrl="https://sqs.us-east-1.amazonaws.com/370060650944/fromLambdaQueue",
        MessageBody=json.dumps(value)
    )

    return {
        'statusCode': 200,
        'body': json.dumps(value)
    }
```

COLA 1: "FROMLAMBDAQUEUE"

Amazon SQS > Colas > fromLambdaQueue

fromLambdaQueue

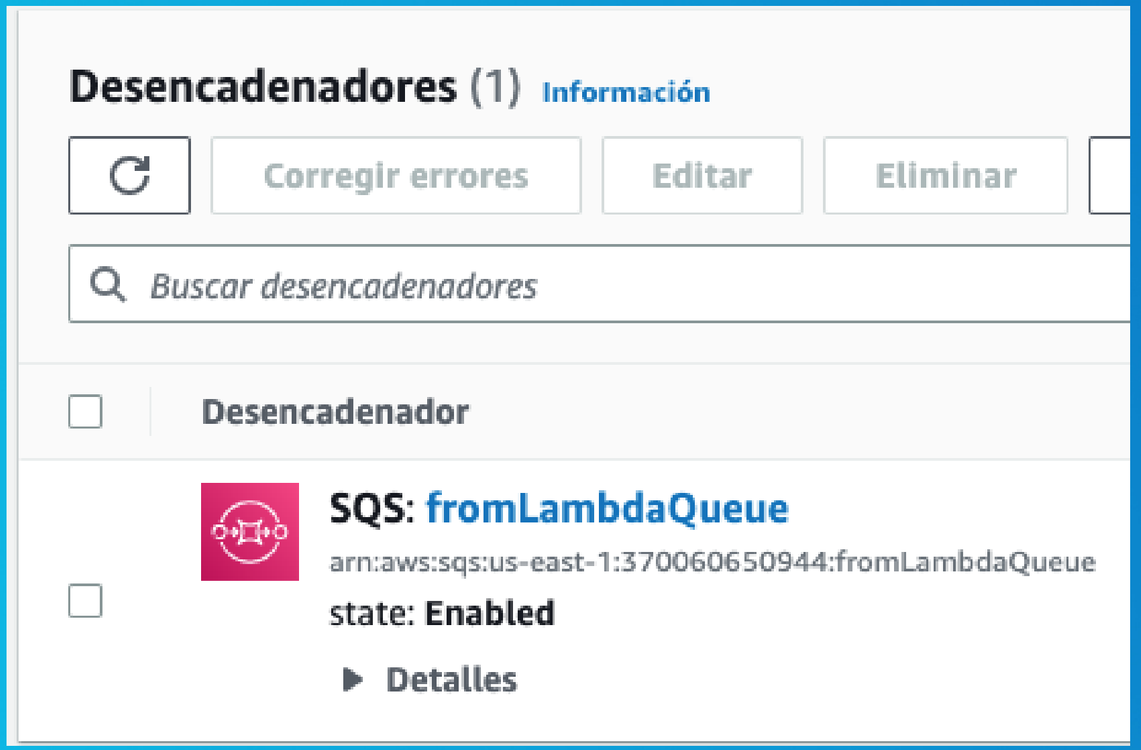
Editar | Eliminar | Purgar | Enviar y recibir mensajes | Iniciar el redireccionamiento de la DLQ

Detalles Información

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

▶ Más

FUNCTION: "EXPERT-SERVICE"



lambda_function × Execution results ×

```python
import json
import boto3

def lambda_handler(event, context):
    message = event['Records'][0]['body']
    data = json.loads(message)
    value = float(data['value'])

    inclination = value * 2
    velocity = value / 2

    dicc_export = {
        "value": str(value),
        "inclination": str(inclination),
        "velocity": str(velocity)
    }

    sqs = boto3.client('sqs')

    sqs.send_message(
        QueueUrl="https://sqs.us-east-1.amazonaws.com/370060650944/queue2",
        MessageBody=json.dumps(dicc_export)
    )
    sqs.send_message(
        QueueUrl="https://sqs.us-east-1.amazonaws.com/370060650944/queue3",
        MessageBody=json.dumps(dicc_export)
    )

    return {
        'statusCode': 200,
        'body': json.dumps(dicc_export)
    }
```

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

▶ Más

Amazon SQS > Colas > queue3

queue3

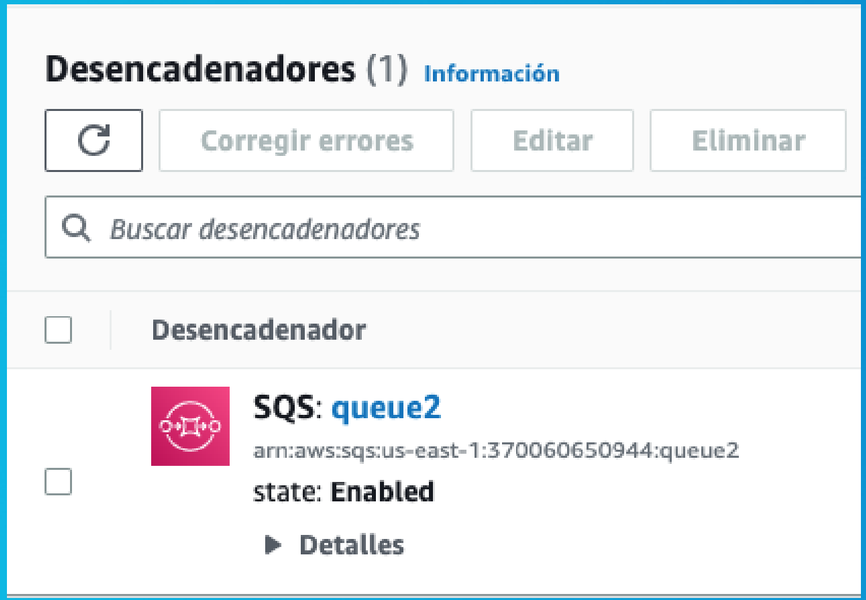
Editar | Eliminar | Purgar | Enviar y recibir mensajes | Iniciar el redireccionamiento de la DLQ

Detalles Información

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

▶ Más

FUNCTION: "LAMBDA-FUNCTION-WITH-RDS"



lambda_function × Execution results ×

```python
import sys
import logging
import pymysql
import json

# rds settings
rds_host  = "tallerprogra2.cunetuyifwaj.us-east-1.rds.amazonaws.com"
user_name = "admin"
password = "admin123456789"
db_name = "db_taller"

logger = logging.getLogger()
logger.setLevel(logging.INFO)

# create the database connection outside of the handler to allow connections to be
# re-used by subsequent function invocations.
try:
    conn = pymysql.connect(host=rds_host, user=user_name, passwd=password, db=db_name, connect_timeout=5)
except pymysql.MySQLError as e:
    logger.error("ERROR: Unexpected error: Could not connect to MySQL instance.")
    logger.error(e)
    sys.exit()

logger.info("SUCCESS: Connection to RDS MySQL instance succeeded")
```

FUNCTION: "LAMBDA-FUNCTION-WITH-RDS"

lambda_function × Execution results ×

```python
    sys.exit()

logger.info("SUCCESS: Connection to RDS MySQL instance succeeded")

def lambda_handler(event, context):
    """
    This function creates a new RDS database table and writes records to it
    """
    message = event['Records'][0]['body']
    data = json.loads(message)
    value = data['value']
    inclinacion = data['inclination']
    velocidad = data['velocity']

    item_count = 0
    sql_string = f"insert into sensorvalues (ritmocardiaco, inclinacion, velocidad) values('{value}', '{inclinacion}

    with conn.cursor() as cur:
        #cur.execute("create table if not exists Customer ( CustID  int NOT NULL, Name varchar(255) NOT NULL, PRIMAF
        cur.execute(sql_string)
        conn.commit()
        cur.execute("select * from sensorvalues")
        logger.info("The following items have been added to the database:")
        for row in cur:
            item_count += 1
            logger.info(row)
    conn.commit()
    return "Added %d items to RDS MySQL table" %(item_count)
```

1:1 Python Spaces: 4

AMAZON RDS: TALLER-PROGRA-2

RDS > Bases de datos > tallerprogra2

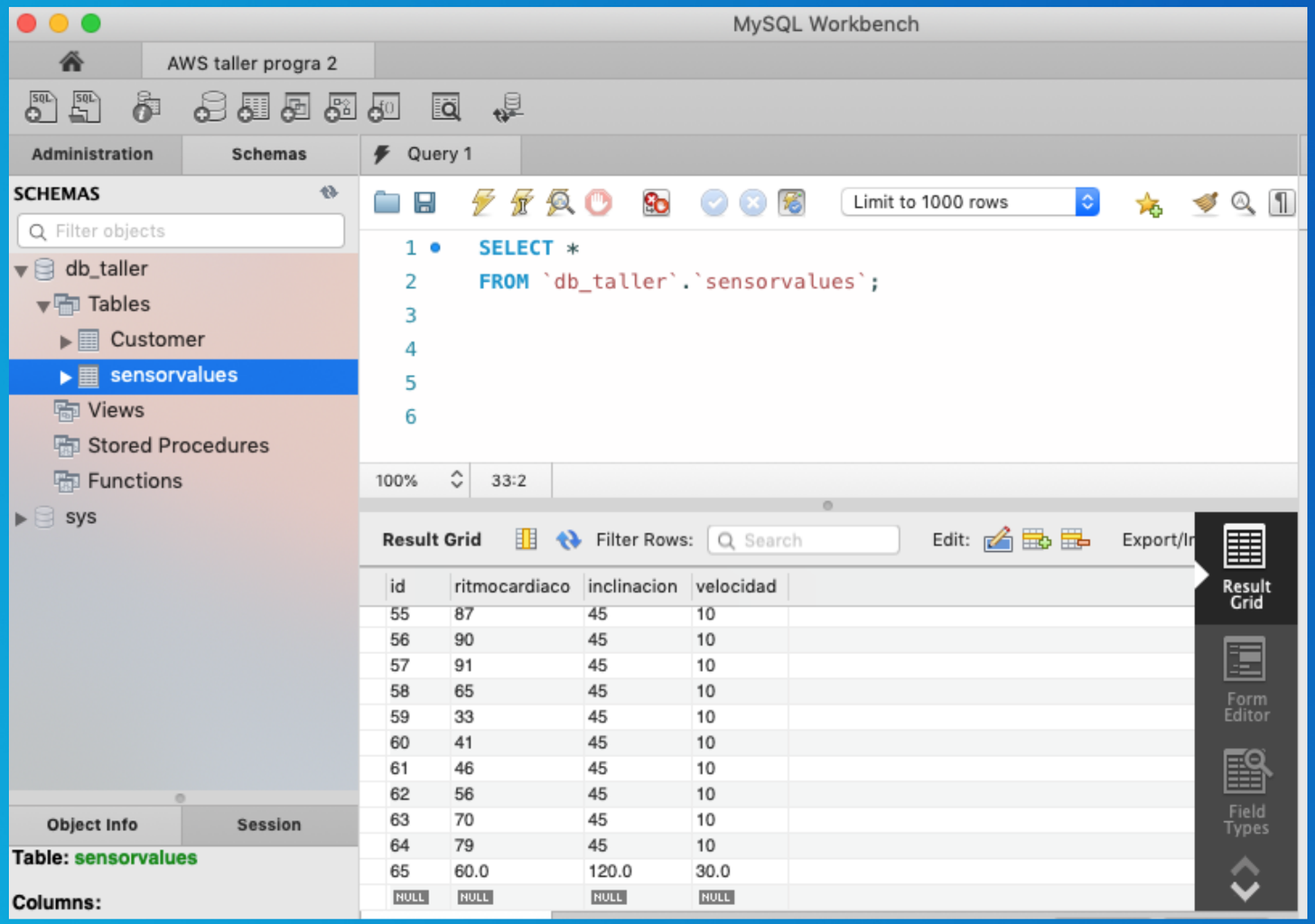
tallerprogra2

Modificar | Acciones

Resumen

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

BD MYSQL: DB_TALLER



DEMO