**Documentación del proyecto:**

**Aspirante:** Fabián Andrés Ramírez Mensa

**Repositorio GitHub:** <https://github.com/magnusalexander1/repositorio_glb>

**Enlace API:** https://container-service-1.km3mrqvdrhfpm.us-east-1.cs.amazonlightsail.com/empleados

**Diagrama de arquitectura de la solución:**

Amazon S3

Amazon RDS

Job Glue

Job Glue

**Icono

Descripción generada automáticamenteIcono

Descripción generada automáticamenteIcono

Descripción generada automáticamente**

Base de datos Postgres

**Icono

Descripción generada automáticamente**

LightSail

Se ingestan los datos en una BBDD postgres

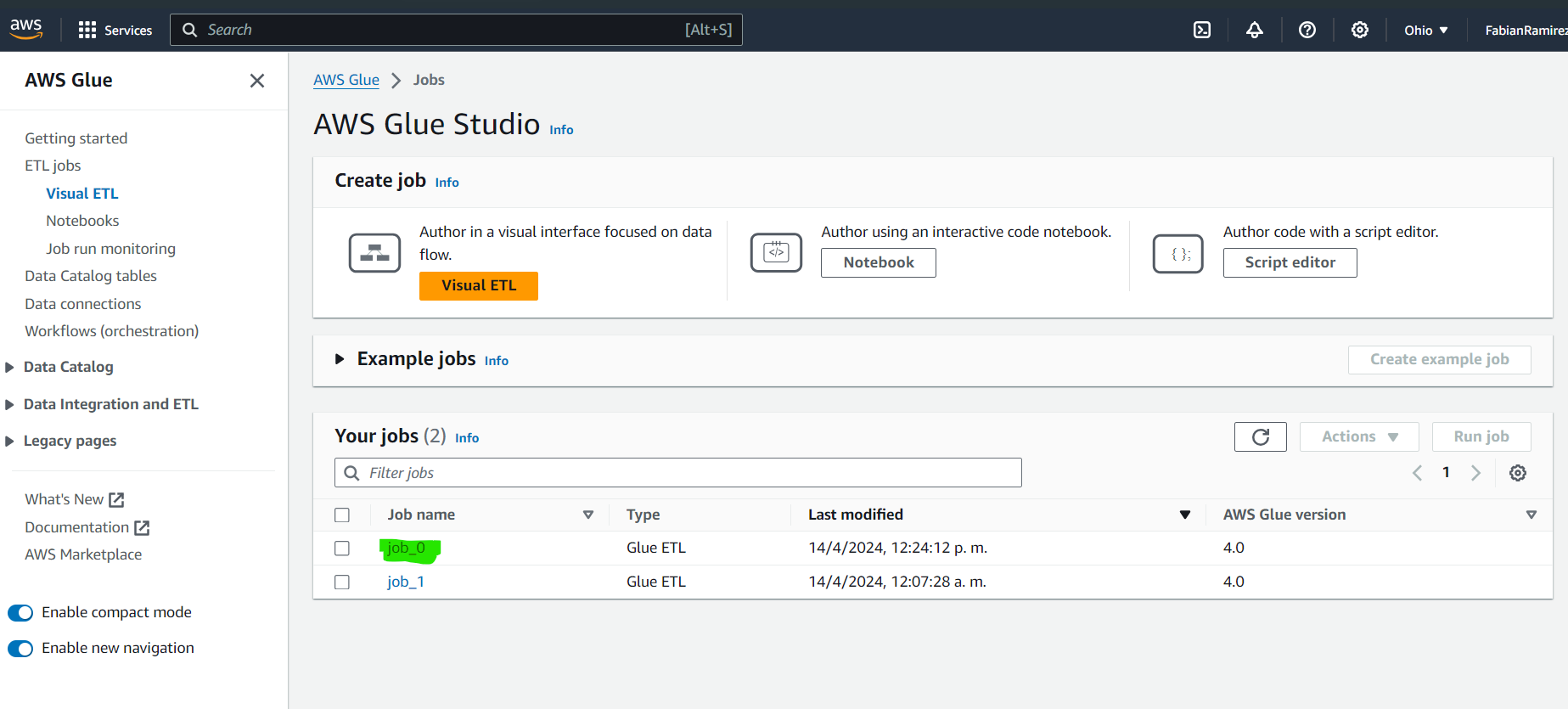
Se almacenan los CSVs en S3

Creación CSVs

Se disponibiliza la APP con Docker

Los archivos CSV empleados, departamentos y puestos\_de\_trabajo fueron creados con un Job de Amazon Glue escrito en Python y disponibilizados en un bucket Amazon S3 para ser más tarde ingestados en una base de datos relacional de Postgres (Amazon RDS) con otro job de Glue. Por último se construyó una vista sencilla en Postgres la cual es consultada a través de una api desarrollada con el framework de Flask, el api se disponibilizó utilizando una imagen de Docker con LightSail.

1. **Creación del Job Glue para generar CSVs:** Para la creación de los archivos CSVs empleados, departamentos y puestos\_de\_trabajo se utiliza un Job de Amazon Glue, el cual está escrito en Python utilizando pandas. Teniendo por JobName “Job\_0”. El script Python se sube al repositorio GitHub (<https://github.com/magnusalexander1/repositorio_glb/blob/main/script_pandas.py>). Se hubiese podido hacer con pyspark también y con la librería Faker, no se hizo con Faker la generación de los nombres porque no estaba disponible en el entorno Glue, no obstante con un poco más de tiempo es posible instalarla en el entorno.



Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Podemos observar las ejecuciones exitosas:  
Una captura de pantalla de una computadora

Descripción generada automáticamente

**La elección del CSV** se hace por simplicidad, al ser un volumen pequeño de datos no es necesario un formato parquet el cual se puede particionar y está optimizado para consultas.

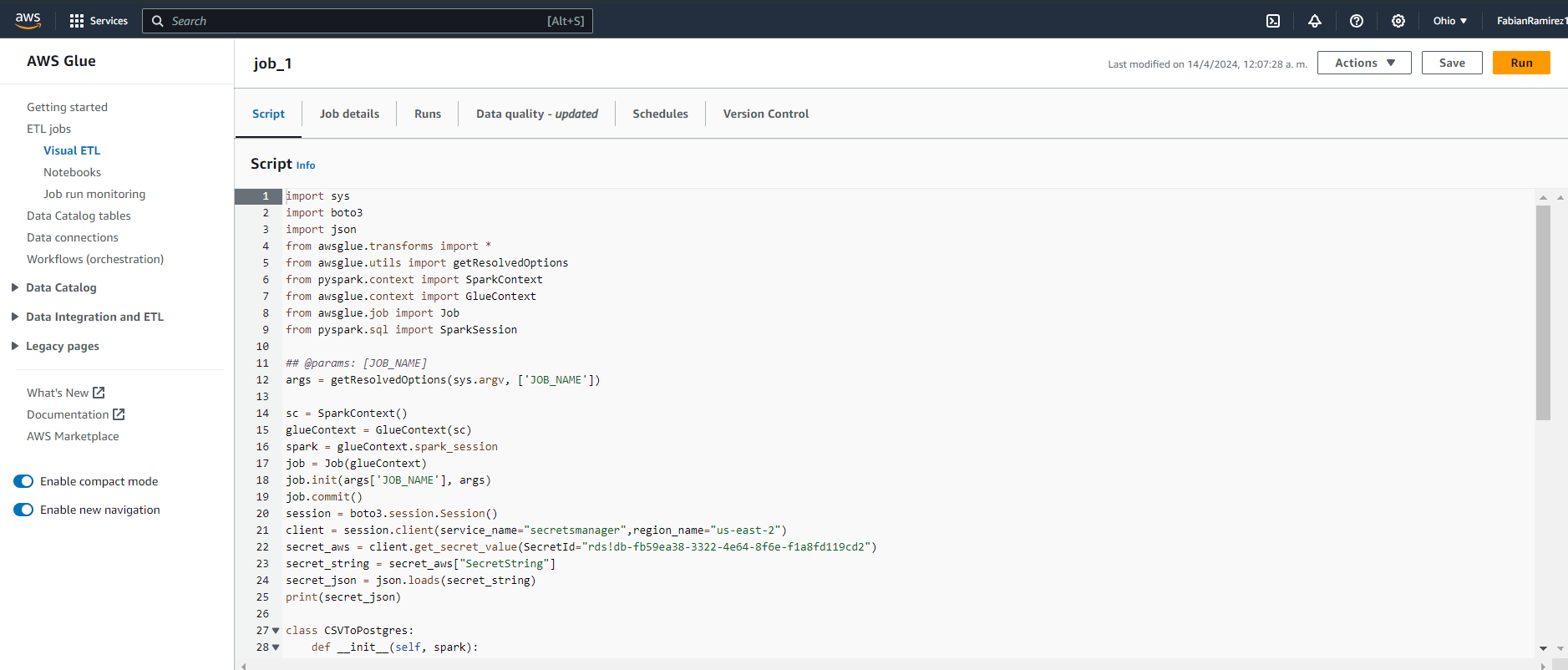
1. **Almacenamiento en capa gratuita en un servicio cloud:** Para el almacenamiento de los CSVs se eligió S3, el servicio de almacenamiento de AWS

En la imagen de abajo se puede observar los CSVs almacenados en S3

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

1. **Ingesta de datos en base de datos postgres.** Para hacer la ingesta de los datos a una base de datos relacional se eligió un Job de Amazon Glue, el cual lee los datos con pyspark y los guarda en una base de datos Postgres utilizando un servicio de AWS llamado RDS. En 3 siguientes imágenes se observa el script del job en glue.

****

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

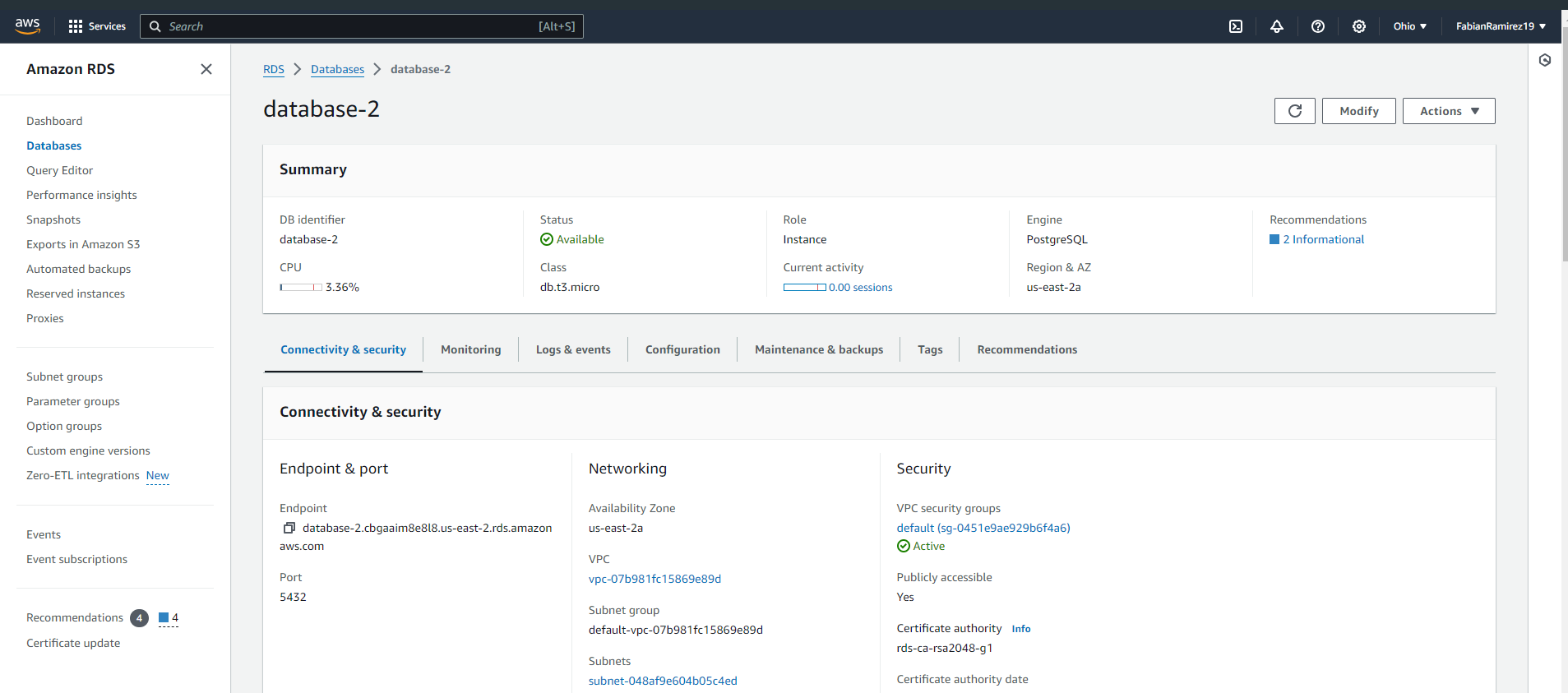
Descripción generada automáticamente**

En la siguiente imagen se puede observar la correcta ejecución del job:

Una captura de pantalla de una computadora

Descripción generada automáticamente

1. **Almacenamiento en RDS:** RDS es un servicio de AWS que nos permite disponibilizar bases de datos relacionales con algunos motores populares como Postgres.

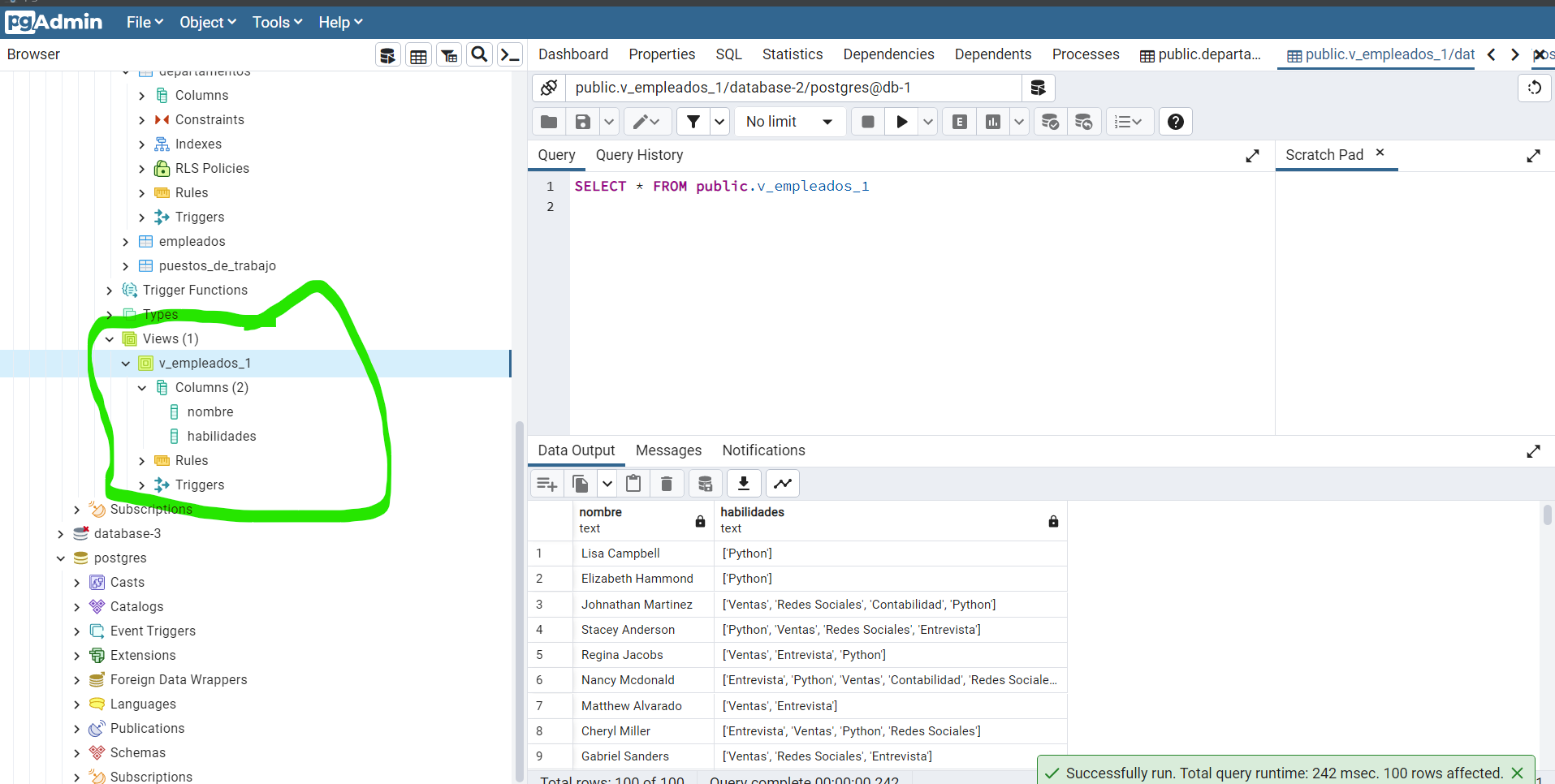


En la siguiente imagen podemos ver a través de PgAdmin las tablas ingestadas en postgres:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Adicionalmente se creó una vista sencilla en Postgres para ser consultada desde una API. A continuación, se puede ver la vista:



1. **Consulta de la vista a través de una API:** El desarrollo de la API se hizo en Flask un framework de Python, el script del API se encuentra en el repositorio como app.py (<https://github.com/magnusalexander1/repositorio_glb/blob/main/app.py>)

Para disponibilizar la aplicación con Docker se utilizó LigthSail. El DockerFile utilizado lo pueden encontrar en el repositorio de GitHub en el siguiente enlace: <https://github.com/magnusalexander1/repositorio_glb/blob/main/Dockerfile>

Para consumir la vista desde la API se puede acceder desde el siguiente enlace:

https://container-service-1.km3mrqvdrhfpm.us-east-1.cs.amazonlightsail.com/empleados

En la imagen de abajo se puede observar el recurso en AWS.

