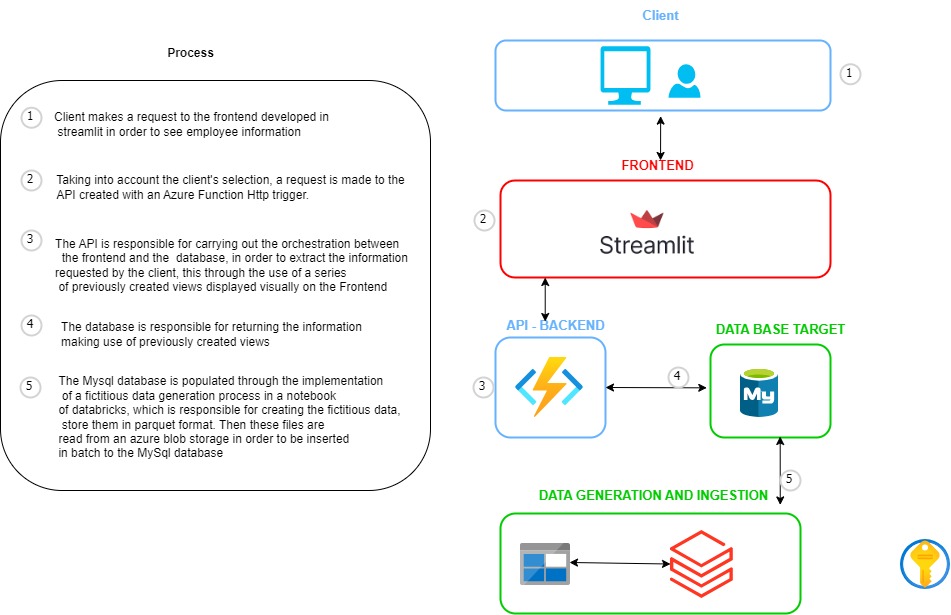
**Arquitectura de la Solución Planteada**



**DESAFIO #1:**

Construya un script que genere de forma automática los datos de: departamentos, puestos de trabajo, y empleados.

Solución:

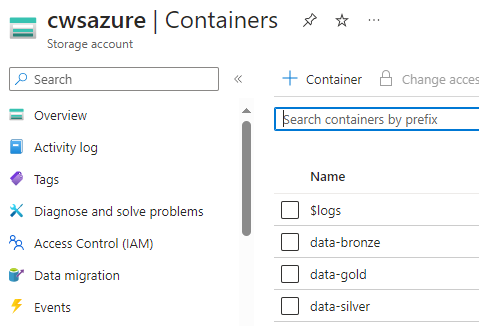
El presente desafío se soluciona en un notebook de Databricks llamado “TestMVM”, este notebook es creado en Databricks comunnity edition.

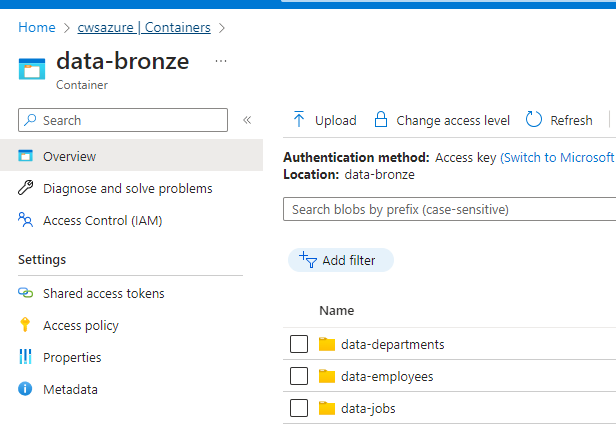
**DESAFIO #2**

Guarde los datos simulados en archivos con formato CSV/Parquet. Explique el porqué de la escogencia del formato. No descarte usar la capa gratuita de algún servicio de almacenamiento tipo cloud, será considerado un plus.

Solución :

Los datos generados son almacenados en un Azure Blob storage, se crea la gestión de capas en blob storage de bronze, silver y gold.





Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

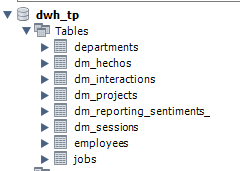
Los datos son almacenados en la capa bronze, en formato parquet ya que este formato es optimizado para el almacenamiento de datos tabulares, este formato realiza una compresión de los datos de forma eficiente permitiendo tener un menor tamaño en comparación a otros formatos. Permitiendo asi tener una reducción en los costos de almacenamiento en el blob storage.

**DESAFIO #3:**

Implemente un proceso batch para migrar los datos a una base de datos SQL/NoSQL, o si lo desea, a un Datawarehouse o bucket analítico  
de un Datalake. No descarte usar la capa gratuita de algún servicio de almacenamiento tipo cloud, será considerado un plus.

Solución:

Se realiza una lectura de los archivos tipo parquet desde un notebook en databricks, el cual lee los archivos y los almacena en batch a una base de datos Mysql, en la cual se crean 3 tablas (employees, Jobs, departments) con el fin de almacenar la información que proviene del blob storage.



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**DESAFIO #4:**

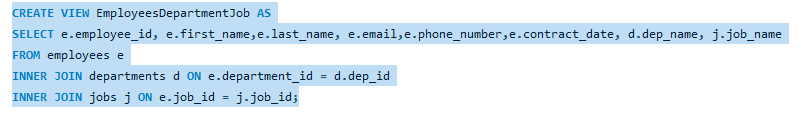
Dependiendo si escoge una base de datos SQL/NoSQL, un Datawarehouse, o un Datalake, entonces desarrolle una view/query/report a  
partir del modelo de datos

Solución:

Se crean 5 vistas con el fin de generar una serie de reportes para ser invocadas desde el frontend de streamlit.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

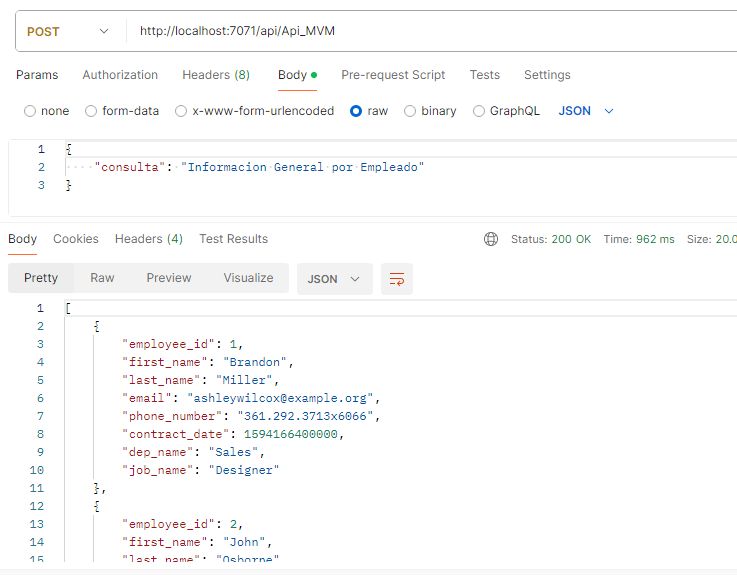
**DESAFIO #5:**

Desarrolle una API REST para consultar la view/query/report. Para el desarrollo de la API considere algún framework de Python, C#/.Net.

Solución :

Se crea un azure Function Http Trigger como orquestador de servicios, el cual es el encargado de orquestar las peticiones realizadas por el frontend desarrollado en streamlit, y gestionar el acceso a la base de datos para el consumo de las vistas desarrolladas anteriormente para la generación de reportes.

Se selecciona este servicio ya que es un servicio brindado por azure el cual es aprovisionado, escalable y mantenido de manera automática brindando un aprovisionamiento serverles evitando asi la administración de servidores o infraestructura.



**DESAFIO #6:**

Mejore la implementación de la API realizando un despliegue que use contenedores (valide las distintas opciones que le brinda su nube).  
Considere una prueba de consumo a la API implementando o activando algún front de acceso para ejecutar la invocación a la view/query/report.

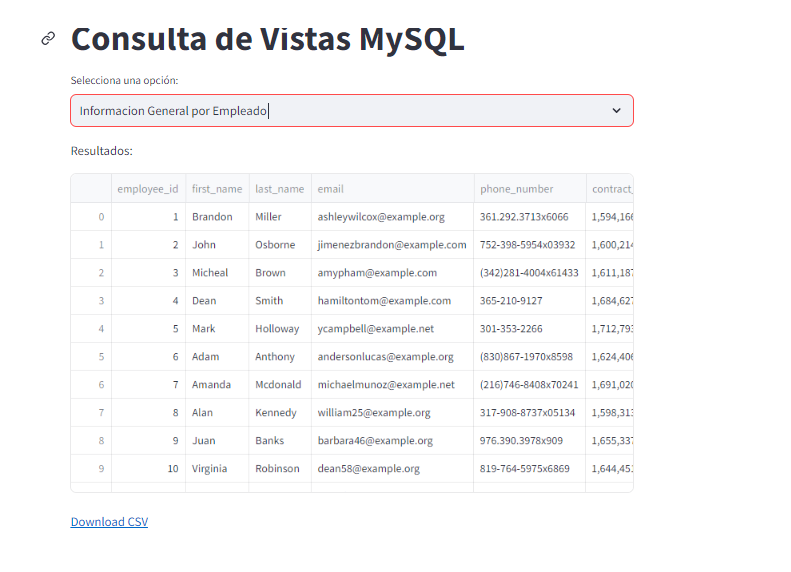
Solución:

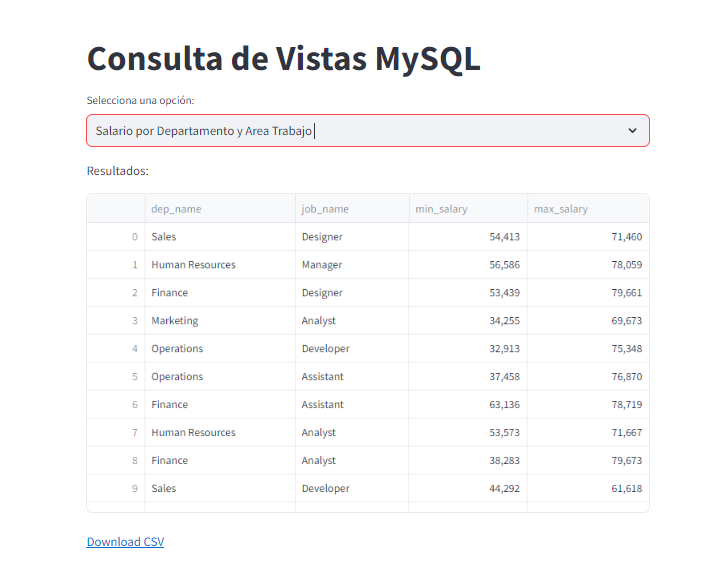
Se crea un frontend en streamlit el cual es el encargado de consumir el API desarrollada con un Azure Function Http Trigger.

Este frontend cuenta con un combo box el cual tiene definida una serie de opciones las cuales permite la generación de reportes teniendo en cuenta las vistas generadas en la base de datos.



Una vez se selecciona alguna de las opciónes, este se encarga de realizar una petición al api, teniendo en cuenta la vista correspondiente del reporte de ña base de datos mysql. Generando como resultado un reporte, el cual puede ser descargado como un archivo CSV por parte del usuario





Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente