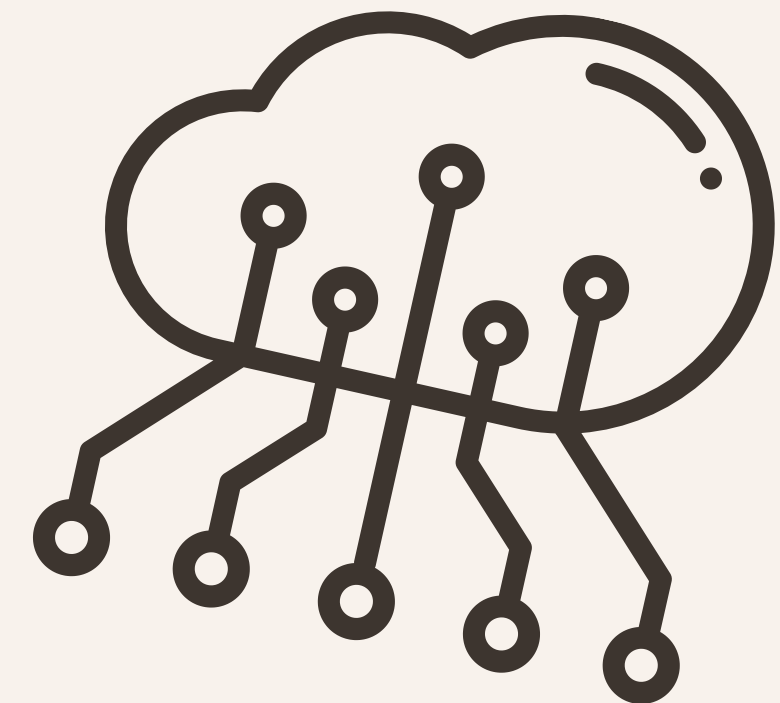


**Analisis de Datos**

# PROYECTO FINAL



**Presentado por:**  
**Alvarado Erika**  
**Coronado Sebastian**  
**Marcatoma Dustin**  
**Minga Anthony**  
**Rios Alan**



# Definición del caso de estudio

- Resumen administrativo
- Declaración de problema clave
- Exploración del problema
- Resolución de la propuesta

# Objetivos Generales y específicos

## General

- Abordar los cinco casos de estudios.

## Específico

- Selección y justificación
- Integración de las fuentes
- Configuración de las bases de datos

# **Descripción del equipo de trabajo y actividades realizadas por cada uno**

Definición del alcance y requisitos – Todos

Diseño de la arquitectura del Data Lake – Todos

Configuración y conexión de fuentes de datos – Rios, Alvarado

Power BI y concentrador de datos – Rios, Marcatoma

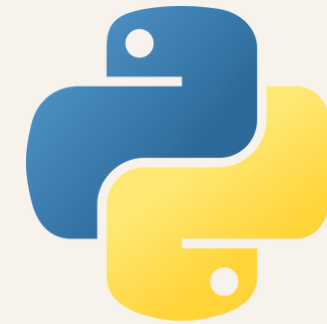
Creación de los dashboards – Marcatoma, Coronado

Documentación final – Alvarado, Minga

# Recursos y herramienta utilizadas



- Jupyter Notebook



- Python



- SQLite



- MySQL

# Recursos y herramienta utilizadas



- SQL server



- MongoDB compass

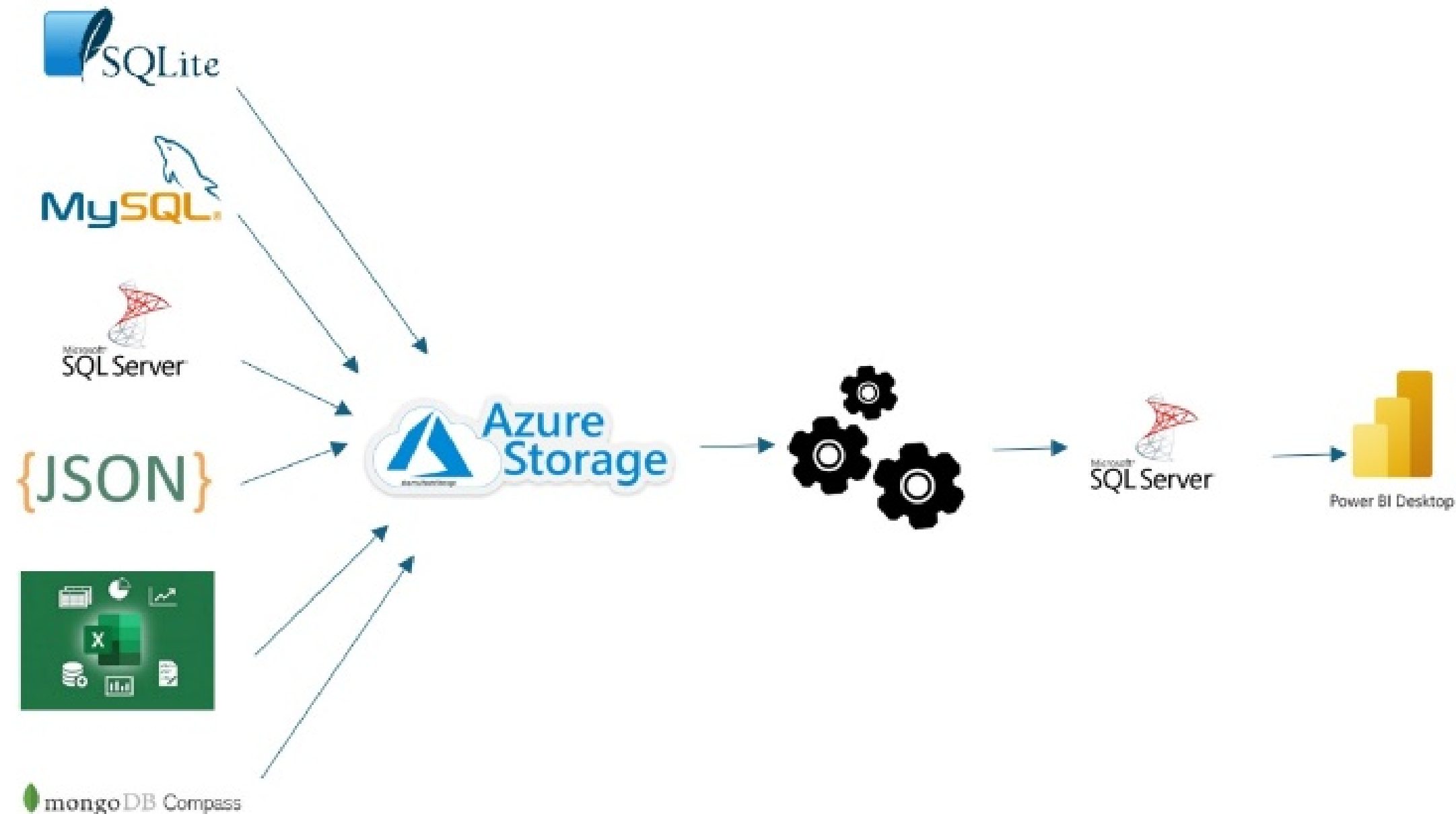


- Azure Storage



- Power BI

# Arquitectura de la solución



# Extracción de datos

## Fuentes

Kaggle.com

UC Irvine Machine Learning  
Repository.

opendatanetwork.com

data.fivethirtyeight.com

reddit.com/r/datasets/

beta.data.gov.sg/datasets

concertful.com

metacritic.com



# Extracción de datos

```
import pandas as pd

# Cargar el archivo CSV
df = pd.read_csv('RankingVideoGames.csv')

# Eliminar filas duplicadas
df = df.drop_duplicates()

# Rellenar los valores nulos en columnas de texto con 'sin información'
df.loc[:, df.dtypes == object] = df.loc[:, df.dtypes == object].fillna('sin información')

# Rellenar los valores nulos en columnas numéricas con 0
df.loc[:, df.dtypes != object] = df.loc[:, df.dtypes != object].fillna(0)

# Guardar los datos limpios en un nuevo archivo CSV
df.to_csv('JuegosL1.csv', index=False)
```

Limpieza de datos data set juegos.

# Extracción de datos

```
import pandas as pd

df = pd.read_csv('IMDb movies.csv', encoding='latin-1')

df = df.drop_duplicates()

# Rellenar los valores nulos en columnas de texto con 'sin información'
df.loc[:, df.dtypes == object] = df.loc[:, df.dtypes == object].fillna('sin información')

# Rellenar los valores nulos en columnas numéricas con 0
df.loc[:, df.dtypes != object] = df.loc[:, df.dtypes != object].fillna(0)

df.to_csv('PelículasL1.csv', index=False)
```

Limpieza de datos data set películas.

```
import pandas as pd

df = pd.read_csv('articles3.csv')
df = df.drop_duplicates()

# Rellenar los valores nulos en columnas de texto con 'sin información'
df.loc[:, df.dtypes == object] = df.loc[:, df.dtypes == object].fillna('sin información')

# Rellenar los valores nulos en columnas numéricas con 0
df.loc[:, df.dtypes != object] = df.loc[:, df.dtypes != object].fillna(0)

df.to_csv('NoticiasL1.csv', index=False)
```

Limpieza de datos data set noticias.

# Extracción de datos

```
|: import pandas as pd

|: df = pd.read_csv('concerts.csv')

df = df.drop_duplicates()

# Rellenar los valores nulos en columnas de texto con 'sin información'
df.loc[:, df.dtypes == object] = df.loc[:, df.dtypes == object].fillna('sin información')

# Rellenar los valores nulos en columnas numéricas con 0
df.loc[:, df.dtypes != object] = df.loc[:, df.dtypes != object].fillna(0)

df.to_csv('ConciertosL1.csv', index=False)
```

Limpieza de datos data set conciertos.

```
import pandas as pd

df = pd.read_csv('IMDb movies.csv', encoding='latin-1')

df = df.drop_duplicates()

# Rellenar los valores nulos en columnas de texto con 'sin información'
df.loc[:, df.dtypes == object] = df.loc[:, df.dtypes == object].fillna('sin información')

# Rellenar los valores nulos en columnas numéricas con 0
df.loc[:, df.dtypes != object] = df.loc[:, df.dtypes != object].fillna(0)

df.to_csv('PelículasL1.csv', index=False)
```

Limpieza de datos data set ciencia.

# Análisis de información

```
•[39]: import mysql.connector as mysql #Conector necesario para MySQL
import pandas as pd #Libreria para leer los datasets
from datetime import datetime #Libreria para cambiar el formato de fechas de los datasets en caso de ser necesario
from sqlalchemy import create_engine #Libreria para enviar data frames a MySQL

•[2]: x1=mysql.connect(host="localhost",user="root", passwd="Tsuki1904") #Credenciales de conexion
x1

[2]: <mysql.connector.connection.MySQLConnection at 0x1c419bf3e30>

•[3]: x2=x1.cursor() #Variable encargada de ejecutar instrucciones en MySQL
x2.execute("create database DLVideojuegos1")

[5]: x2.execute("use DLVideojuegos1")

•[7]: df=pd.read_csv('JuegosL1.csv') # Leemos el datasets

[8]: df
```

Conexión a worbench y creación de base de datos.

# Análisis de información

```
•[14]: fechas=df['release_date'] # Guardamos en una variable la columna de las fechas
fechas
```

```
[14]: 0      November 23, 1998
      1      September 20, 2000
      2      April 29, 2008
      3      September 8, 1999
      4      April 29, 2008
      ...
      18795     May 21, 2013
      18796     November 3, 2003
      18797     September 22, 2015
      18798     October 15, 2012
      18799     March 31, 2009
      Name: release_date, Length: 18800, dtype: object
```

```
•[18]: fechas=pd.to_datetime(fechas) #Convertimos las fechas en un formato de tipo "Date"
```

```
•[19]: fechas=fechas.dt.strftime('%Y-%m-%d') # Cambiamos la fecha al formato establecido
```

```
•[20]: df['release_date']=fechas #Aplicamos el nuevo formato a la columna de fecha de nuestro data frame
```

```
[21]: df
```

Modificación de la columna fecha

# Análisis de información

```
•[22]: df.to_csv('JuegoL1.csv') # Transformamos el data frame a un archivo CSV con Los nuevos cambios

•[43]: enviar_datos=create_engine('mysql+pymysql://root:Tsuki1904@localhost/DLVideojuegos1')
      df.to_sql('datosxd',con=enviar_datos, if_exists='replace',index=False) # Enviamos el data frame a MySQL

[43]: 18800

[ ]:
```

Inserción en MySQL.

# Análisis de información

```
[45]: #DATASET4 A SQLITE
# Carga tu dataset en un DataFrame
data = pd.read_csv('JuegosL3.csv')

# Crea una conexión a la base de datos SQLite
conn = sqlite3.connect('JuegosL3.db')

# Escribe el DataFrame a una tabla en SQLite
data.to_sql('Datos', conn, if_exists='replace', index=False)

# No olvides cerrar la conexión
conn.close()
```

```
[ ]:
```

```
|
```

Inserción en SQLite.

# Análisis de información

```
import pandas as pd
import sqlalchemy
import urllib
import pyodbc

df = pd.read_csv('ConciertosL1.csv')

# Conectar a SQL Server con autenticación de Windows
params = urllib.parse.quote_plus("DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-9KJETOV;DATABASE=ANALISIS;Trusted_Connection=yes;")
engine = sqlalchemy.create_engine("mssql+pyodbc:///?odbc_connect=%s" % params)

# Enviar el DataFrame a SQL Server
df.to_sql('CONCIERTOS', con=engine, if_exists='append', index=False)
```

Inserción en SQL Server.



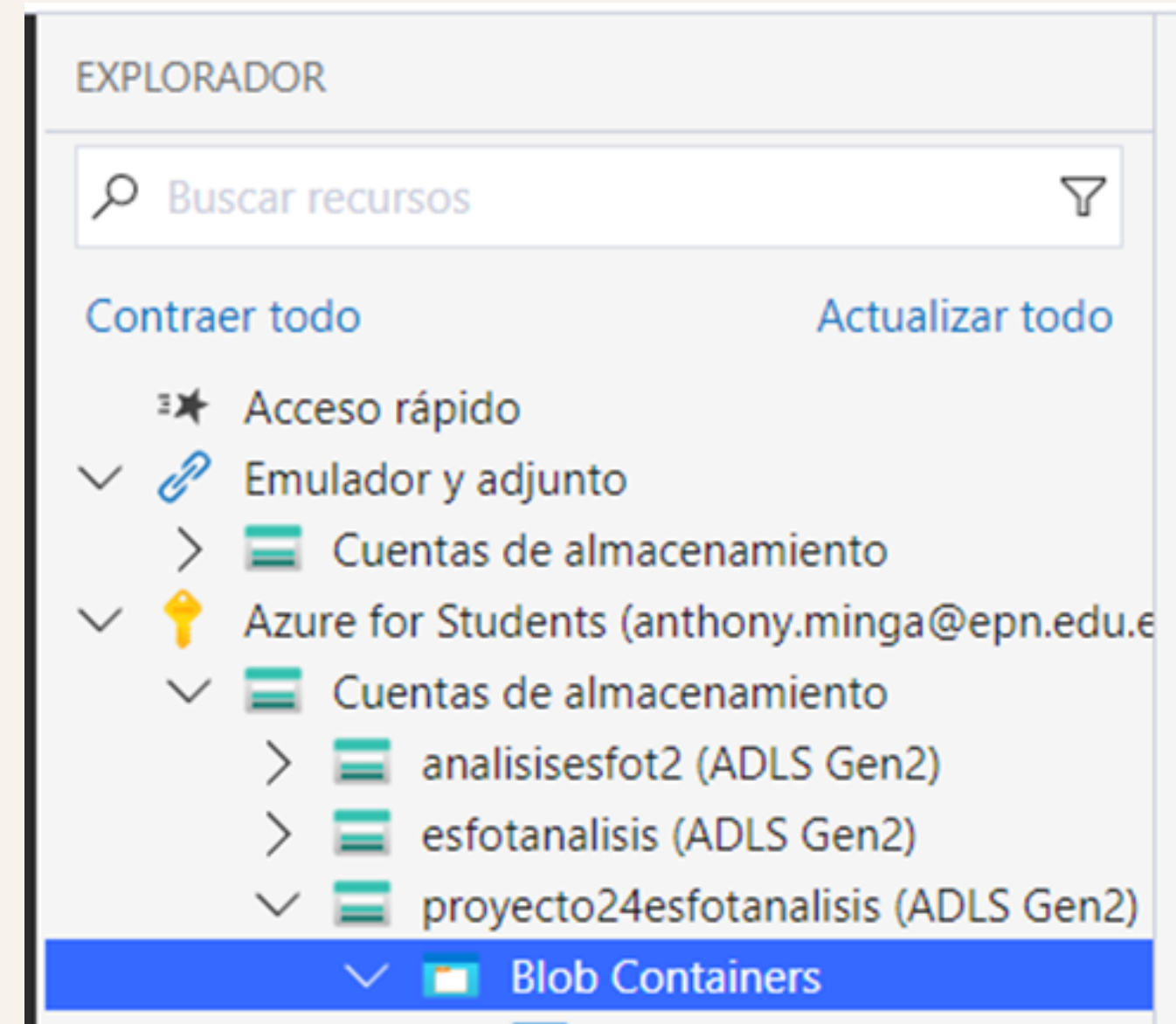
# Análisis de información

```
1  const cheerio = require("cheerio");
2  const axios = require("axios");
3  const j2cp = require("json2csv").Parser;
4  const fs = require("fs");
5
6  let baseUrl = "https://www.metacritic.com/browse/movie/?releaseYearMin=1910&releaseYearMax=2024&page=";
7
8  let movies = [];
9  let num_pags = 2;
10
11 async function getMovies(url) {
12   try {
13     const response = await axios.get(url);
14     const $ = cheerio.load(response.data);
15     const movie = $(".c-finderProductCard_info");
16
17     movie.each(function () {
18       const spans = $(this).find(".c-finderProductCard_titleHeading");
19       const id = $(spans[0]).text();
20       const title = $(spans[1]).text();
21       const date = $(this).find(".u-text-uppercase").text().trim();
22       const description = $(this).find(".c-finderProductCard_description span").text();
23       const metascor = $(this).find(".c-siteReviewScore span").text();
24
25       movies.push({ id, title, date, description, metascor });
26     });
27
28     if (num_pags < 669) {
29       const next_page = baseUrl + num_pags;
30       getMovies(next_page);
31       num_pags += 1;
32
33       console.log(next_page);
34     } else {
35       const parser = new j2cp();
36       const csv = parser.parse(movies);
37
38       fs.writeFileSync("./moviesxd.csv", csv, "utf-8");
39     }
40   } catch (error) {
41     console.error(error);
42   }
43 }
44
45
46 getMovies("https://www.metacritic.com/browse/movie/?releaseYearMin=1910&releaseYearMax=2024&page=1");
```

# Análisis de información

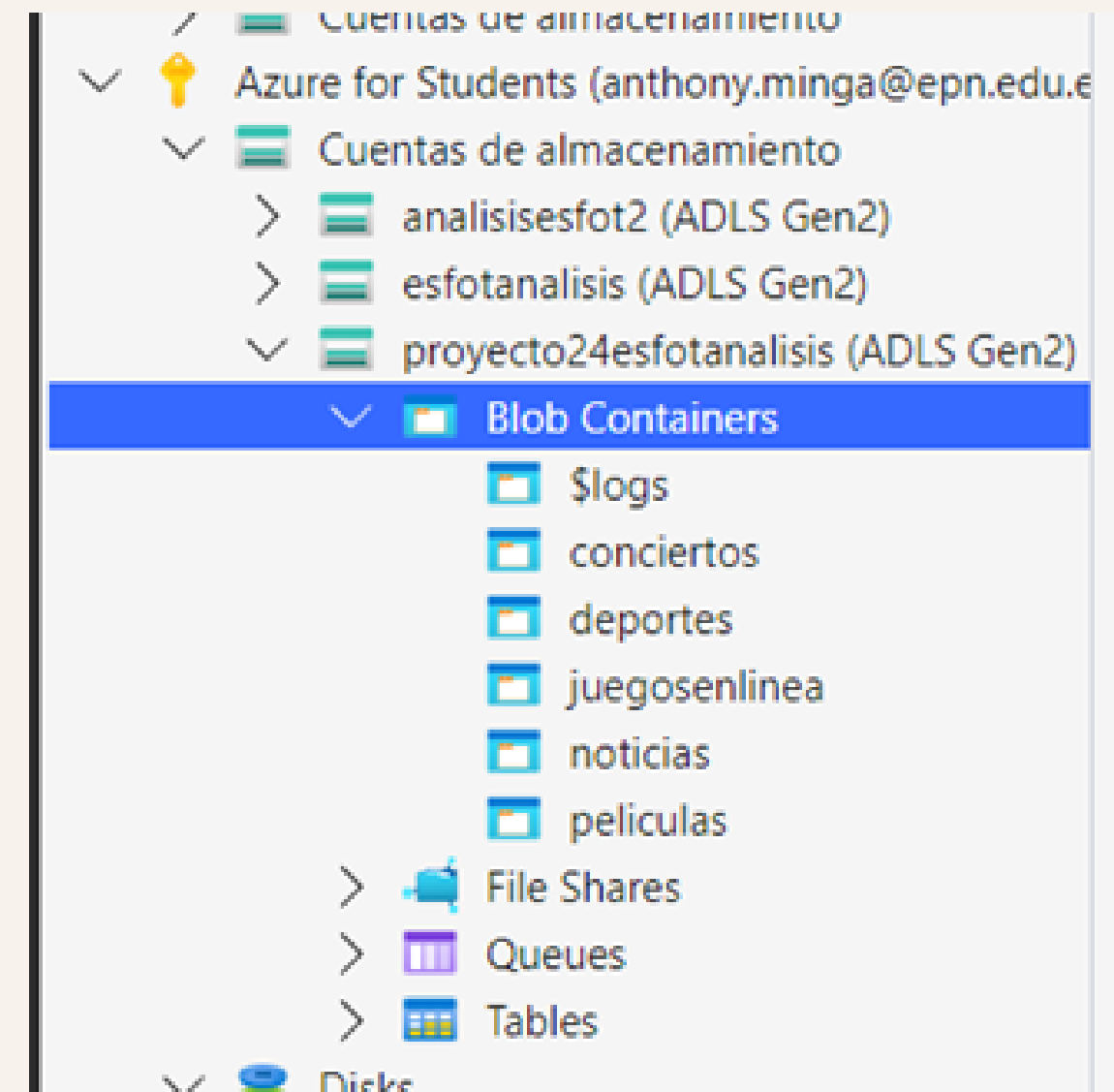
```
1 const cheerio = require("cheerio");
2 const axios = require("axios");
3 const j2cp = require("json2csv").Parser;
4 const fs = require("fs");
5
6 let baseUrl = "https://www.metacritic.com/browse/game/?releaseYearMin=1958&releaseYearMax=2024&page=";
7
8 let games = [];
9 let num_pags = 2;
10
11 async function getGames(url) {
12   try {
13     const response = await axios.get(url);
14     const $ = cheerio.load(response.data);
15     const game = $(".c-finderProductCard_info");
16
17     game.each(function () {
18       const spans = $(this).find(".c-finderProductCard_titleHeading span");
19       const id = $(spans[0]).text();
20       const title = $(spans[1]).text();
21       const date = $(this).find(".u-text-uppercase").text().trim();
22       const description = $(this).find(".c-finderProductCard_description span").text();
23       const metascore = $(this).find(".c-siteReviewScore span").text();
24
25       games.push({ id, title, date, description, metascore });
26     });
27
28     if (num_pags < 549) {
29       const next_page = baseUrl + num_pags;
30       getGames(next_page);
31       num_pags += 1;
32
33       console.log(next_page);
34     } else {
35       const parser = new j2cp();
36       const csv = parser.parse(games);
37
38       fs.writeFileSync("./equisde.csv", csv, "utf-8");
39     }
40   } catch (error) {
41     console.error(error);
42   }
43 }
44
45 getGames("https://www.metacritic.com/browse/game/?releaseYearMin=1958&releaseYearMax=2024&page=1");
46
```

# Data lake



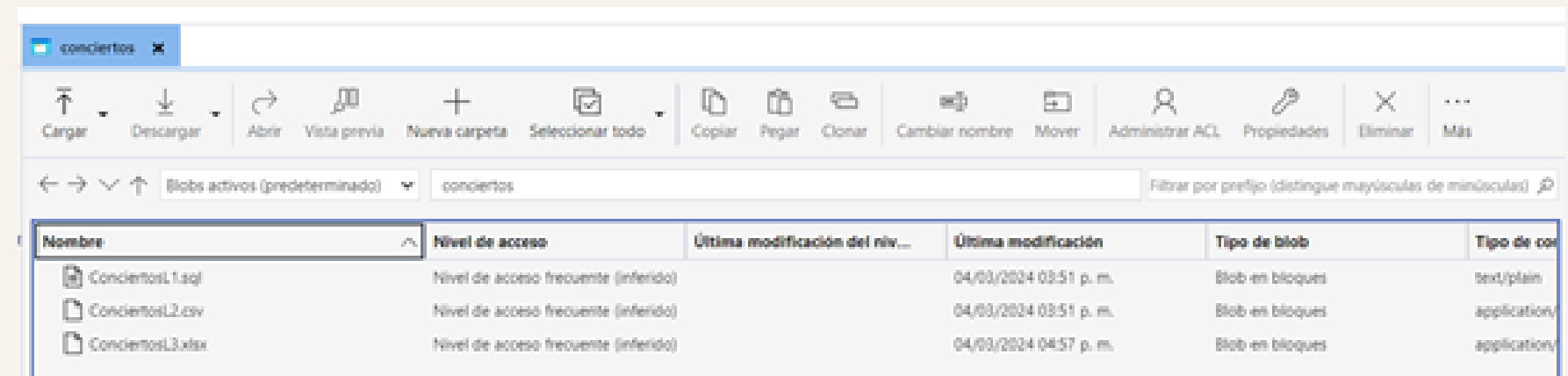
Interfaz Data Lake

# Data lake



Datos introducidos en Azure Storage

# Data lake



Nombre	Nivel de acceso	Última modificación del niv...	Última modificación	Tipo de blob	Tipo de co
Conciertos1.sql	Nivel de acceso frecuente (inferido)		04/03/2024 03:51 p. m.	Blob en bloques	text/plain
Conciertos2.csv	Nivel de acceso frecuente (inferido)		04/03/2024 03:51 p. m.	Blob en bloques	application/
Conciertos3.xlsx	Nivel de acceso frecuente (inferido)		04/03/2024 04:57 p. m.	Blob en bloques	application/

Ejemplo de ingreso de datos

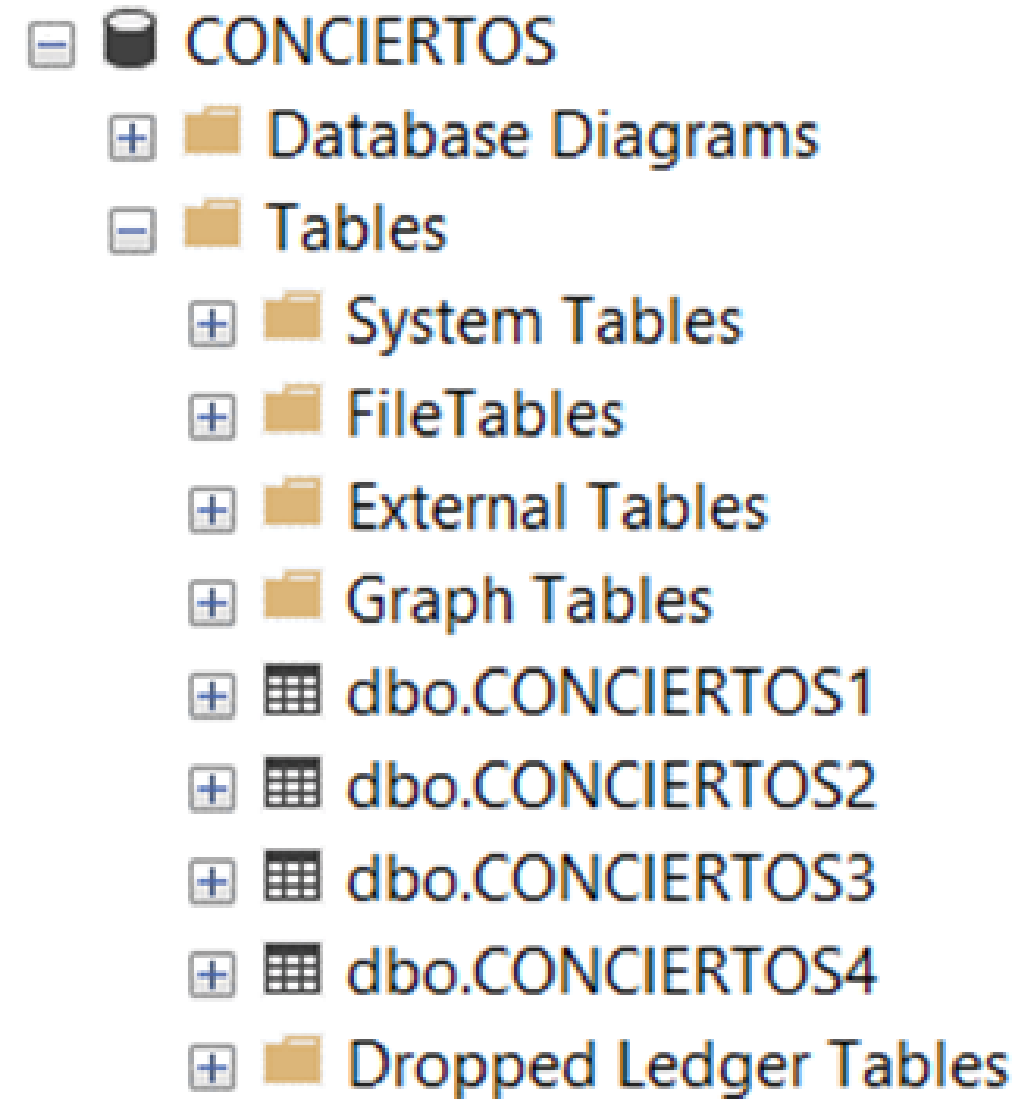
```
Click here to ask Blackbox to help you code faster
df = pd.read_csv("Conciertos1.csv")

# Conectar a SQL Server con autenticación de Windows
params = urllib.parse.quote_plus("DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-9K3JETOV;DATABASE=CONCIERTOS;Trusted_Connection=yes;")
engine = sqlalchemy.create_engine("mssql+pyodbc:///?odbc_connect=%s" % params)

# Enviar el DataFrame a SQL Server
df.to_sql('CONCIERTOS1', con=engine, if_exists='append', index=False)
```

Codigo de conexión a SQL y envio de data frame

# Concentración de datos



Datos introducidos en SQL Server

# Visualización de la información

[https://app.powerbi.com/view?  
r=eyJrIjoibm4YzQxMzEtY2Q0ZC00MjRhLWI4Mz  
Q0YmJjMmI4MjlkZDZlIiwidCI6IjY4MmE0ZTZhLWE  
3N2YtNDk1OC1hM2FjLTllMjY2ZDE4YWEzNyIsIm  
MiOiR9](https://app.powerbi.com/view?r=eyJrIjoibm4YzQxMzEtY2Q0ZC00MjRhLWI4MzQ0YmJjMmI4MjlkZDZlIiwidCI6IjY4MmE0ZTZhLWE3N2YtNDk1OC1hM2FjLTllMjY2ZDE4YWEzNyIsImMiOiR9)

## Suma de home\_team\_fifa\_rank por country y home\_team

**home\_team** ● Albania ● Algeria ● American Samoa ● Andorra ● Angola ● Anguilla





## Suma de Rank por $\tilde{A}$ Location



# **Resultados obtenidos**

# Conclusiones

- La arquitectura y diseño de Data Lake demostró ser eficaz en la integración de datos.
- Power BI como herramienta de visualización de datos proporciono dashboards interactivos y comprensibles.
- Durante el desarrollo del proyecto se identificaron áreas de mejora en la documentación de procesos.

# Recomendaciones

- Se recomienda una planificación detallada y la gestión efectiva del tiempo
- La documentación completa de cada etapa del proyecto es esencial

# Desafíos y problemas encontrados

El principal desafío dentro de la realización de este proyecto fue conseguir los data sets de los diferentes temas, ya que la mayoría que están disponibles gratuitamente llegan a ser de poca o nada de ayuda en el estudio de caso