Universidad Católica de Honduras

"Nuestra Señora Reina de la Paz"

Reporte Técnico

Asignatura:

Bases de Datos Organizacionales

Proyecto Capstone:

Sistema de Análisis de Reseñas y Sentimientos de Usuarios

Integrantes Equipo #3:

Julio César Hernández Santiago
David Eliud Hernández Vidal
Keneth Uriel Chinchilla
Fernando Mejía Pinto
José Inestroza

Docente:

Servio Palacios, Ph.D.

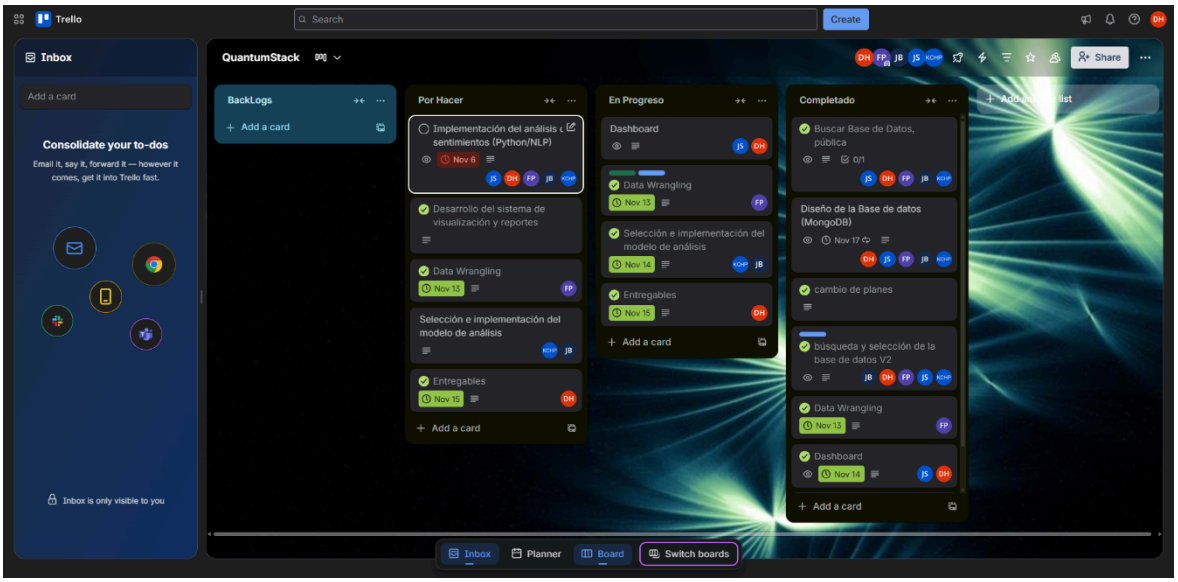
Fecha:

Sábado 15 de noviembre, 2025

Tabla Contenido

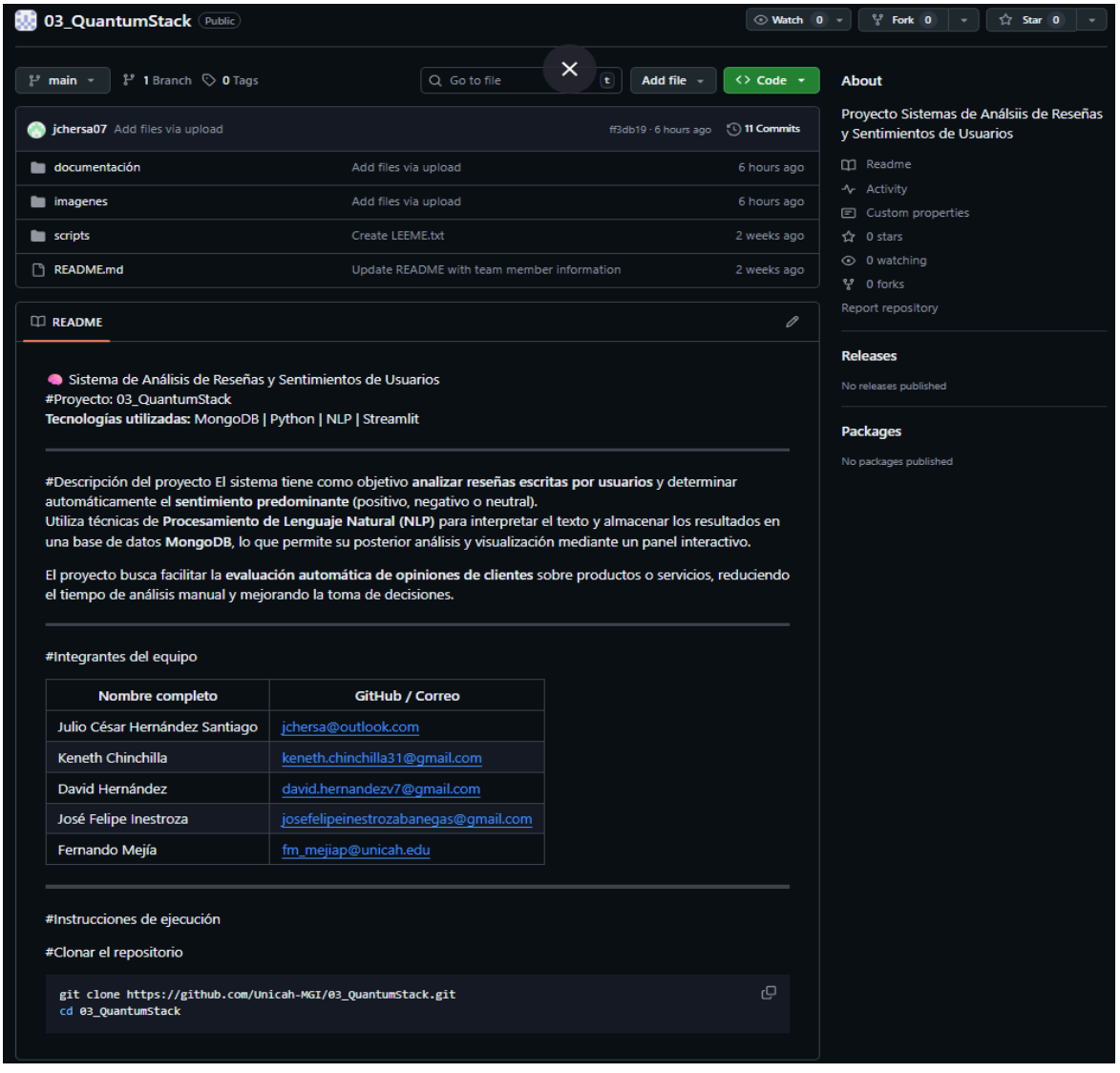
Enlace al tablero Trello

https://trello.com/b/RlXpRi4I/quantumstack



Enlace al repositorio GitHub

https://github.com/Unicah-MGI/03_QuantumStack



Diligencia Técnica

Descripción del problema

Hoy en día las empresas necesitan de herramientas sofisticadas para hacerle seguimiento a lo que la gente opina de sus productos, servicios y marca. La proliferación de las plataformas digitales ha ampliado las maneras en que los individuos manifiestan lo que piensan y sienten en la red, creando así un cúmulo de datos no estructurados que encierran información para la toma de decisiones estratégicas. En este contexto, las empresas que aprovechan el análisis de sentimientos pueden obtener información valiosa de estos datos de forma rápida y eficiente, descubriendo tendencias de satisfacción, quejas o áreas de mejora. Es por ello que The Walt Disney Company quiere conocer qué piensan sus clientes acerca de las experiencias en sus parques temáticos para separar opiniones positivas, neutras y negativas, mejorar la satisfacción del cliente y desarrollar estrategias de marketing más efectivas y personalizadas.
El sistema de análisis de reseñas y sentimientos está diseñado para recopilar, procesar y visualizar opiniones de los usuarios sobre productos o servicios de una organización. Su propósito es transformar datos no estructurados (comentarios, reseñas, publicaciones) en conocimiento útil para la toma de decisiones estratégicas, orientadas a mejorar la experiencia del cliente y la reputación de marca.
El sistema desarrollado tiene como finalidad analizar reseñas y opiniones de usuarios provenientes de diferentes fuentes digitales (plataformas web, redes sociales, formularios, etc.) con el objetivo de determinar el sentimiento expresado en los comentarios positivo, negativo o neutro. A partir de este análisis, la organización puede evaluar la percepción de los clientes y ajustar estrategias de marketing, atención al cliente y desarrollo de productos.
El sistema captura reseñas provenientes de fuentes como redes sociales o formularios web, las procesa mediante algoritmos de *Natural Language Processing* (NLP) y clasifica los textos según su polaridad (positiva, neutra o negativa). Los resultados son almacenados y visualizados en tiempo real a través de un frontend interactivo.

Objetivo General

Desarrollar un sistema de base de datos organizacional capaz de capturar, almacenar y analizar grandes volúmenes de datos no estructurados en formato JSON, provenientes de múltiples fuentes, con el fin de llevar a cabo análisis de sentimientos sobre las reseñas de usuarios y apoyar la toma de decisiones estratégicas orientadas a la satisfacción del cliente.
Aplicar técnicas de Procesamiento de Lenguaje Natural (NLP) para clasificar y evaluar los sentimientos expresados en las reseñas, identificando patrones y tendencias relevantes sobre la percepción de los usuarios.

Objetivos específicos

- Integrar el sistema con herramientas analíticas y visuales basadas en Python, como Streamlit o Power BI, para ofrecer un frontend interactivo que facilite la interpretación de los resultados mediante gráficos y métricas de sentimiento.
- Evaluar la eficiencia, escalabilidad y desempeño del modelo propuesto, asegurando su capacidad para procesar datos en tiempo real y adaptarse al crecimiento del volumen de información y consultas.
- Diseñar un modelo de base de datos NoSQL que permita almacenar de forma flexible los comentarios de los usuarios y los resultados derivados de los algoritmos de análisis de sentimientos (NLP).
- Implementar un sistema con alto rendimiento en operaciones de lectura y escritura, garantizando eficiencia en el procesamiento de datos y en la inserción de resultados analíticos.
- Integrar la base de datos con herramientas analíticas como **Python** y **Power BI**, para la visualización e interpretación de la información procesada.
- Establecer un flujo de trabajo que facilite la extracción, transformación y carga (ETL) de los datos desde fuentes externas, incluyendo datasets públicos y datos sintéticos.
- Evaluar el desempeño del modelo y su escalabilidad frente a volúmenes crecientes de información y consultas frecuentes.

Volumen de datos estimado y tipo de operaciones frecuentes

Se espera tener un volumen de datos de aproximadamente 40,000 registros en la tabla de reseñas, los cuales nos servirán como base para poder realizar el análisis de sentimiento.

Operaciones frecuentes:

- Inserciones masivas de los resultados analizados por los algoritmos NLP.
- adición de datos para realizar operaciones como filtración y transformación de datos.
- Consultas de texto para recopilar la información necesaria a analizar.
- Exportación de datos hacia herramientas de visualización mediante dashboards.

Fuente de datos a utilizar

El sistema emplea un dataset público y validado disponible en kaggle (Chillar, 2020) de donde realizaremos el proceso de Extracción, Transformación y Carga de los datos que contienen las reseñas de los usuarios, complementado con la creación de datos sintéticos para llenar las demás tablas de la base de datos.

Justificación tecnológica del modelo elegido

a) MongoDB como sistema de base de datos:

MongoDB fue seleccionado por su naturaleza NoSQL documental**,** ideal para datos no estructurados como reseñas en formato texto. Entre sus ventajas:

- Permite almacenar documentos JSON flexibles**,** adaptables a cambios en el esquema sin afectar el rendimiento.
- Soporta operaciones de agregación y búsqueda de texto (full-text search), esenciales para analizar palabras clave o frases frecuentes.
- Ofrece escalabilidad horizontal en entornos distribuidos**,** lo que permite manejar millones de reseñas con alta disponibilidad.
- Se integra fácilmente con herramientas de análisis y visualizacion como Python, Streamlit, Power BI y frameworks de análisis, facilitando la conexión entre la base de datos y los modelos de NLP.

El tipo de base de datos elegida es debido a la necesidad de manipular y analizar grandes cantidades de datos no estructurados (textos, reseñas, comentarios, etc.) provenientes de distintas fuentes y se justifica por los siguientes factores:

1. Escalabilidad horizontal

Este sistema generaría un gran volumen de datos contante por lo cual se debe de mantener un rendimiento a medida que crece la cantidad de datos.

2. Flexibilidad de esquema

La variabilidad de los datos puede llegar a ser un problema, pero con esta herramienta podemos mitigar ese problema ya que cuenta con una excelente flexibilidad en cuestión de datos, ya que el formato que maneja es BSON, que es una extensión de JSON, a lo cual podemos guardar documentos con estructura variable, lo que nos permite adaptar el modelo de datos sin rediseñar la base de datos, siendo un esquema ideal si necesitamos que los datos evolucionen o se integre otras fuentes de datos.

3. Alto rendimiento en consultas específicas

El rendimiento de esta herramienta para lectura y escritura es perfecto para la cantidad de datos que se desean manejar en esta base de datos, nos resulta clave para este tipo de sistemas por lo que podemos obtener información de manera rápida y eficiente con consultas complejas.

b) Python como lenguaje de procesamiento

Python es la columna vertebral del procesamiento analítico, debido a:

- Su extenso ecosistema de librerías para **machine learning y NLP** (como *spaCy*, *NLTK*, *transformers* de Hugging Face).
- Conectividad directa con MongoDB mediante *PyMongo* o *Motor*, lo que permite insertar resultados analíticos de manera eficiente.
- Capacidad para construir pipelines de **ETL (Extracción, Transformación y Carga)** que limpian, procesan y transforman los datos antes del análisis.

c) NLP (Procesamiento de Lenguaje Natural)

El módulo NLP se encarga de interpretar el contenido textual de las reseñas, utilizando técnicas de:

- **Tokenización** y **lemmatización** para normalizar el texto.
- **Análisis de sentimiento** mediante modelos *BERT* o *RoBERTa*.
- **Extracción de palabras clave (keywords)** para identificar temas recurrentes.

Los resultados se almacenan junto con la reseña original, enriqueciendo los datos con metainformación semántica.

d) Frontend y visualización

El sistema cuenta con una interfaz construida con tecnologías compatibles con el ecosistema Python, como **Streamlit** o **Dash**, que permiten:

- Visualizar métricas clave (porcentaje de reseñas positivas/negativas).
- Generar **gráficos interactivos** (nubes de palabras, barras, tendencias temporales).
- Consultar y filtrar reseñas directamente desde el navegador.

Arquitectura general del sistema

Flujo de trabajo:

1. **Ingesta de datos**: recopilación de reseñas desde APIs, archivos o formularios web.
2. **Preprocesamiento (Python)**: limpieza del texto y detección del idioma.
3. **Análisis de sentimiento (NLP)**: clasificación de polaridad y extracción de keywords.
4. **Almacenamiento (MongoDB)**: los resultados se guardan en documentos JSON estructurados por usuario, reseña y análisis.
5. **Visualización (Frontend)**: dashboards interactivos en Power BI donde se muestran los resultados de manera intuitiva.

La arquitectura del sistema de análisis de reseñas y sentimientos se compone de **cuatro capas principales**, cada una con responsabilidades específicas: **captura**, **procesamiento**, **almacenamiento** y **visualización**. A continuación se detalla la arquitectura completa.

1. Capa de Ingesta de Datos (Data Ingestion Layer)

Aquí se reúnen todas las reseñas que serán procesadas. Las fuentes pueden ser:

- Formularios web
- APIs (por ejemplo, redes sociales o plataformas de reseñas)
- Archivos CSV/JSON
- Bases de datos externas

Esta capa entrega los datos sin procesar al motor de análisis.

2. Capa de Procesamiento y Análisis (Processing & NLP Layer)

Implementada principalmente en **Python**, esta capa contiene el pipeline de procesamiento:
a) Preprocesamiento

- Limpieza del texto
- Eliminación de ruido
- Normalización (lowercase, lematización, tokenización)
- Detección de idioma

b) Análisis de Sentimiento (NLP)
Utiliza modelos como:

- BERT
- RoBERTa
- Modelos de HuggingFace
- SpaCy o NLTK para utilidades

Genera:

- Polaridad (positivo, negativo, neutro)
- Keywords
- Score de sentimiento
- Categorías o temas

c) ETL hacia la base de datos
Una vez analizadas, las reseñas se transforman y envían hacia MongoDB mediante librerías como **PyMongo**.

3. Capa de Almacenamiento (Database Layer)

El sistema utiliza **MongoDB** como base de datos documental por su:

- Flexibilidad para datos no estructurados (JSON)
- Escalabilidad horizontal
- Excelente integración con Python
- Compatibilidad con búsquedas de texto y agregaciones

Colecciones principales:

- **users**
- **reviews**
- **sentiments**
- **parks**

MongoDB almacena tanto la reseña original como los resultados del análisis NLP, formando documentos enriquecidos.

4. Capa de Visualización y Consumo (Frontend / Visualization Layer)

Esta capa presenta los resultados del análisis a los usuarios finales.
Desarrollada con tecnologías compatibles con Python:

- **Streamlit**
- **Dash (Plotly)**
- **Flask + HTML/CSS/JS**
- **Power BI** integrado mediante exportación

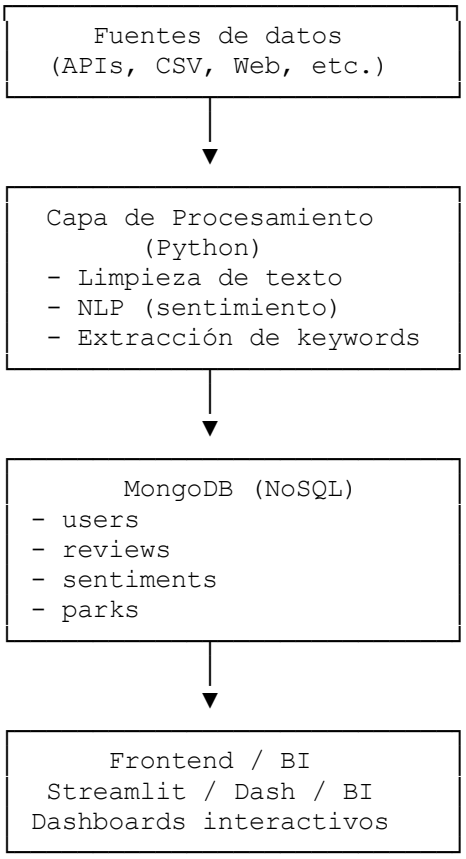
Funcionalidades típicas:

- Dashboard de polaridad global
- Gráficos de tendencia por fecha
- Nube de palabras (keywords)
- Filtros por usuario, parque o sentimiento
- Búsqueda de reseñas individuales

Flujo de la arquitectura (Resumen)

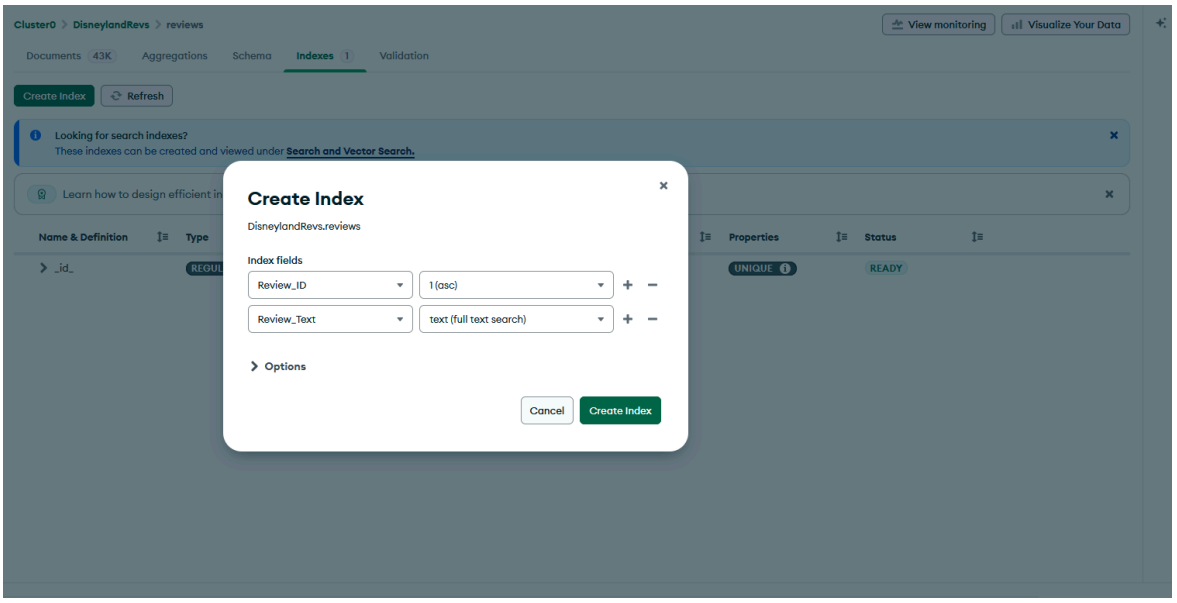
1. **Ingreso de datos:** las reseñas llegan desde diversas fuentes.
2. **Procesamiento (Python):** el texto se limpia, analiza y clasifica con NLP.
3. **Almacenamiento (MongoDB):** se guarda el documento enriquecido con sentimiento, keywords y metadatos.
4. **Visualización (Frontend):** dashboards permiten explorar resultados en tiempo real.

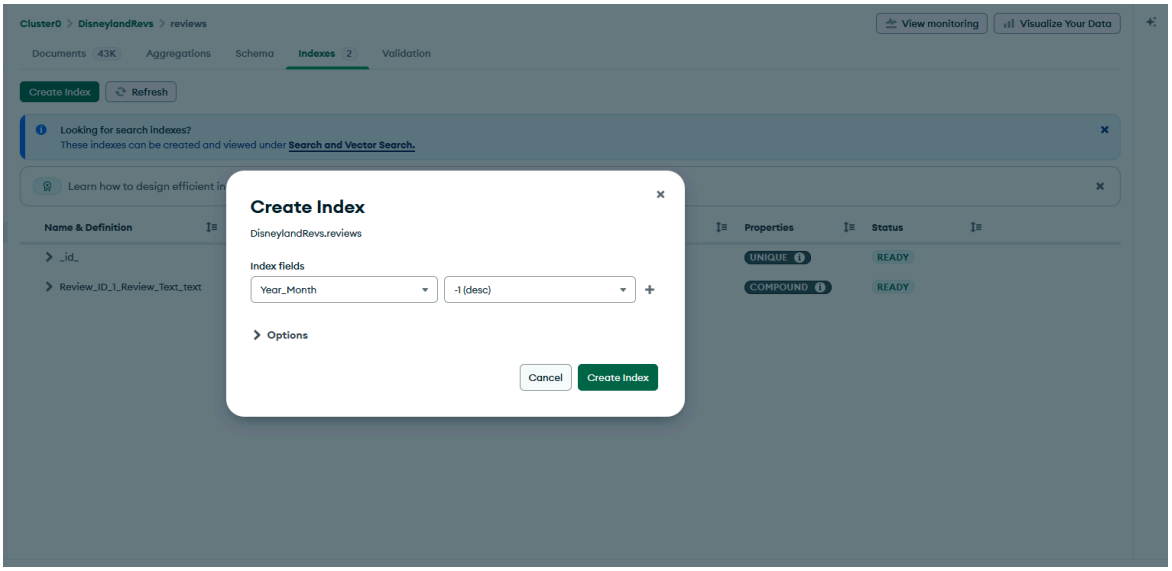
Arquitectura en formato gráfico (resumen)

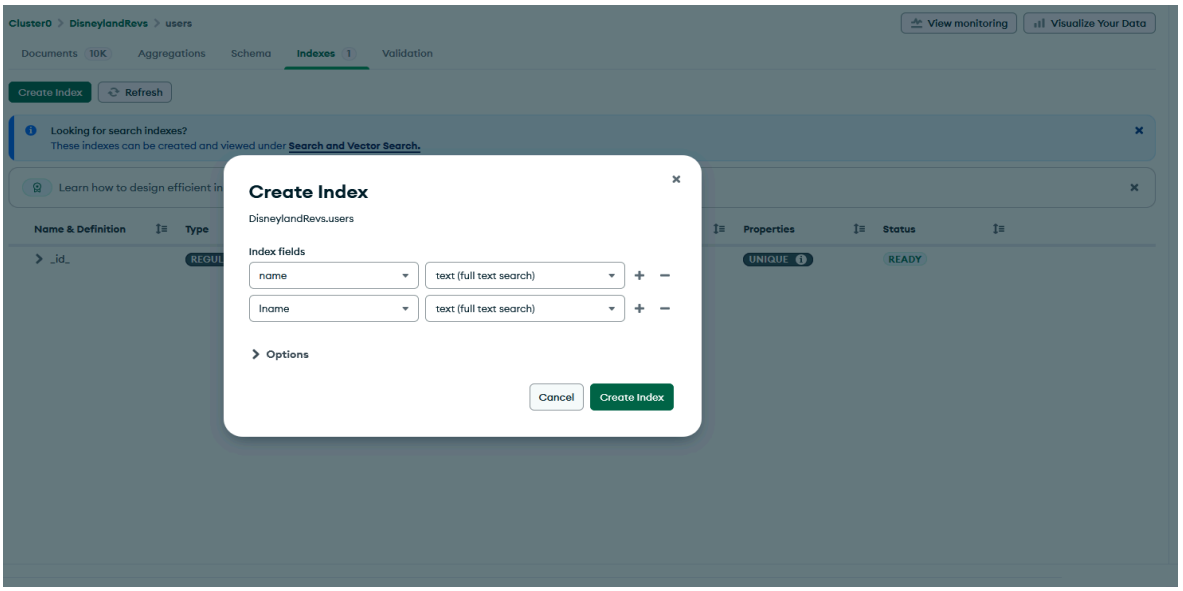


Indices MongoDB

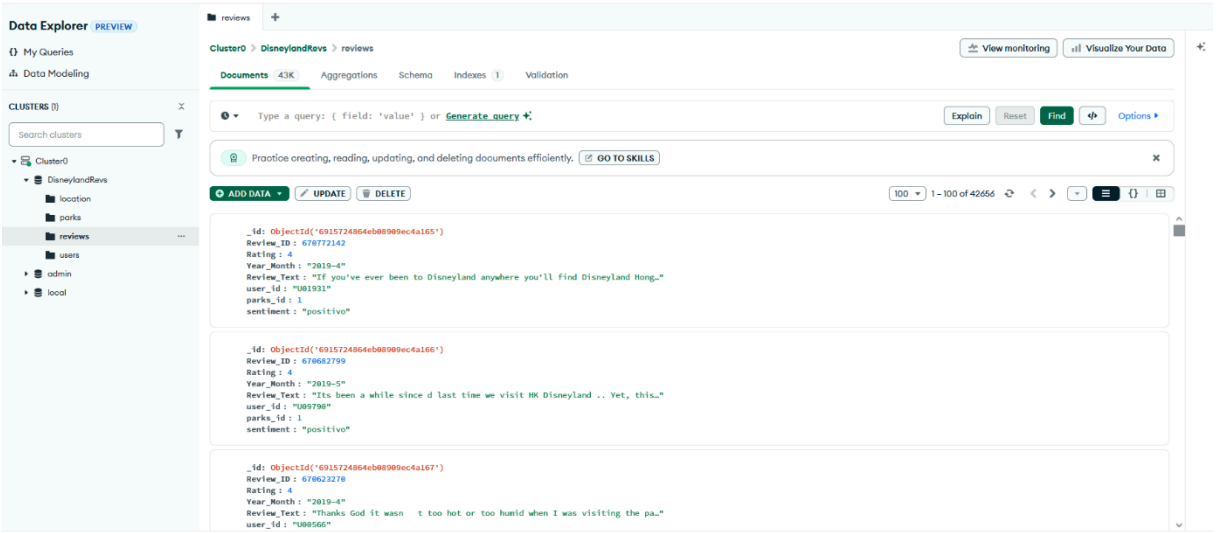
Optimizar los tiempos de consultas en MongoDB para la coleccions de reviews







Cluster MongoDB




```
#Importacion de librerias necesarias
from pymongo import MongoClient
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Obtencion de IP publica del notebook
!curl ipinfo.io/ip
```

```
34.74.139.251
```

```
#Conexion a Mongo DB
uri = "mongodb+srv://fmmejiap_db_user:Fer99194805@cluster0.172zje5.mongodb.net/"
client = MongoClient(uri)
db = client["DisneylandRevs"]
collection = db["reviews"]
print("Conexión establecida con MongoDB")
```

```
Conexión establecida con MongoDB
```

```
#Creacion de un DatFrame con los datos del CSV
df = pd.read_csv('DisneylandReviews.csv', encoding="ISO-8859-1")
df.head()
```

```
Show hidden output
```

```
#Verificamos que no existan valores nulos
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42656 entries, 0 to 42655
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Review_ID          42656 non-null  int64 
 1   Rating             42656 non-null  int64 
 2   Year_Month         42656 non-null  object
 3   Reviewer_Location  42656 non-null  object
 4   Review_Text        42656 non-null  object
 5   Branch             42656 non-null  object
dtypes: int64(2), object(4)
memory usage: 2.0+ MB
```

```
#Creamos un DataFrame para realizar cambios dentro de la estructura sin afetar el DF principal
df1 = pd.DataFrame(df)
#Eliminamos columnas que no van a ser necesarias dentro de la tabla Reviews
df1 = df1.drop(['Reviewer_Location','Branch'], axis=1)
df1.head()
```

	Review_ID	Rating	Year_Month	Review_Text
0	670772142	4	2019-4	If you've ever been to Disneyland anywhere you...
1	670682799	4	2019-5	Its been a while since d last time we visit HK...
2	670623270	4	2019-4	Thanks God it wasn t too hot or too humid wh...
3	670607911	4	2019-4	HK Disneyland is a great compact park. Unfortu...
4	670607296	4	2019-4	the location is not in the city, took around 1...

```
#Creamos un DataFrame con 3 registros con la informacion de los parques de Disneyland
data = {
    "id": [1, 2, 3],
    "name": ["Hong Kong Disneyland", "California Disneyland", "Paris Disneyland"],
    "thematic": ["Magic Kingdom", "Magic Kingdom", "Magic Kingdom"],
    "location": ["Hong Kong", "California", "Paris"]
}

df_Parks = pd.DataFrame(data)
print(df_Parks)
```

```
   id                 name       thematic    location
0   1   Hong Kong Disneyland  Magic Kingdom   Hong Kong
1   2  California Disneyland  Magic Kingdom  California
2   3       Paris Disneyland  Magic Kingdom       Paris
```

```
#Creamos un DataFrame con 3 registros con la informacion de la locacion de los parques
data = {
    "address": ["Boulevard de Parc, Coupvray","1313 Disneyland Drive","Lantau Island"],
    "city": ["Paris","Anaheim","Lantau Island"],
    "state": ["Paris","California","Hong Kong"],
    "country": ["Francia","United States","China"]
}

df_LocationParks = pd.DataFrame(data)
print(df_LocationParks)
```

```
                      address          city       state        country
0  Boulevard de Parc, Coupvray         Paris       Paris        Francia
1       1313 Disneyland Drive        Anaheim  California  United States
2               Lantau Island  Lantau Island   Hong Kong          China
```

```
#Creamos un nuevo dataframe en el cual almacenamos los datos de los nombres de los parques del dataframe principal de reviews
df3 = pd.DataFrame(df['Branch'])
df3.info()

#Reemplazamos los nombres de los parques por el id del parque
df3['Branch'] = df3['Branch'].replace({'Disneyland_HongKong': '1'})
df3['Branch'] = df3['Branch'].replace({'Disneyland_California': '2'})
df3['Branch'] = df3['Branch'].replace({'Disneyland_Paris': '3'})
#Cambiamos el tipo de datos de Objeto por int
df3[['Branch']] = df3[['Branch']].astype("int")
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42656 entries, 0 to 42655
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Branch  42656 non-null  object
dtypes: object(1)
memory usage: 333.4+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42656 entries, 0 to 42655
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Branch  42656 non-null  int64
dtypes: int64(1)
memory usage: 333.4 KB
```

```
#Agregamos la nueva columna parks_id con los id de los parques
df1['parks_id'] = df3['Branch'].values
df1.tail()
```

	Review_ID	Rating	Year_Month	Review_Text	user_id	parks_id
42651	1765031	5	missing	i went to disneyland paris in july 03 and thou...	U00158	3
42652	1659553	5	missing	2 adults and 1 child of 11 visited Disneyland ...	U02674	3
42653	1645894	5	missing	My eleven year old daughter and myself went to...	U00327	3
42654	1618637	4	missing	This hotel, part of the Disneyland Paris compl...	U09648	3
42655	1536786	4	missing	I went to the Disneyparis resort, in 1996, wit...	U03480	3

```
#Una vez tenemos terminamos la manipulacion de los datos procedemos a subirlos a MongoDB
#Insercion de datos em la coleccion Reviews
collection.delete_many({})
records = df1.to_dict('records')
collection.insert_many(records)
print(f"Datos insertados: {len(records)} registros")
```

```
Datos insertados: 42656 registros
```

```
#Seleccionamos la coleccion usuarios
collection1 = db["users"]

#Insercion de datos en la coleccion users
collection1.delete_many({})
records = df2.to_dict('records')
collection1.insert_many(records)
print(f"Datos insertados: {len(records)} registros")
```

```
Datos insertados: 10000 registros
```

```
#Seleccionamos la coleccion parks
collection2 = db["parks"]

#Insercion de datos en la coleccion parks
collection2.delete_many({})
records = df_Parks.to_dict('records')
collection2.insert_many(records)
print(f"Datos insertados: {len(records)} registros")
```

```
InsertManyResult([ObjectId('6915738564eb08909ec56f15'), ObjectId('6915738564eb08909ec56f16'), ObjectId('6915738564eb08909ec56f17')], acknowledged=True)
```

```
#Seleccionamos la coleccion location
collection3 = db["location"]

#Insercion de datos en la coleccion location
collection3.delete_many({})
records = df_LocationParks.to_dict('records')
collection3.insert_many(records)
print(f"Datos insertados: {len(records)} registros")
```

```
Datos insertados: 3 registros
```

Modelo API del Sistema de Análisis de Reseñas y Sentimientos

El sistema expone una **API RESTful** desarrollada típicamente en **Python (FastAPI o Flask)**, que sirve como intermediario entre:

- El **frontend**
- Los **módulos de NLP**
- La **base de datos MongoDB**

Esto asegura desacoplamiento, escalabilidad y acceso seguro a los datos.
Arquitectura del Modelo API está compuesta por tres capas:

- Capa de Controlador (Endpoints REST)
  Define las rutas y métodos HTTP.

- Capa de Servicio (Lógica de negocio y NLP)
  Procesos como clasificación, análisis y transformación de datos.

- Capa de Acceso a Datos (MongoDB) Consultas, inserciones y agregaciones.

Modelo estructurado

A. Gestión de Usuarios
1) Crear usuario
POST /api/users

```
{
 "user_id": "0504200198071",
 "name": "John",
 "lname": "Doe",
 "email": "john@gmail.com",
 "phone": "+50495483127"
}
```

2) Obtener usuario por ID
GET /api/users/{user_id}
Respuesta:

```
{
 "user_id": "0504200198071",
 "name": "John",
 "lname": "Doe",
 "email": "john@gmail.com"
}
```

B. Gestión de Reseñas
1) Crear reseña
POST /api/reviews

```
{
 "review_id": "45928",
 "user_id": "0504200198071",
 "park_id": "06",
 "text": "Fue un excelente paseo por los castillos. perfecto día.",
 "date": "2024-06"
}
```

2) Obtener reseñas por usuario
GET /api/reviews/user/{user_id}

3) Obtener todas las reseñas
GET /api/reviews

C. Procesar Análisis de Sentimiento (NLP)
1) Analizar una reseña
POST /api/sentiment/analyze
La API envía el texto a un modelo NLP (por ejemplo BERT).
Body:

```
{
 "text": "Fue un excelente paseo por los castillos."
}
```

Respuesta:

```
{
 "polarity": 1,
 "score": 0.93,
 "model": "BERT",
 "keywords": ["excelente", "perfecto"]
}
```

2) Guardar análisis de sentimiento vinculado a una reseña
POST /api/sentiment/save

```
{
 "review_id": "45928",
 "polarity": 1,
 "model": "BERT",
 "keywords": ["excelente", "perfecto"]
}
```

D. Consultas analíticas
1) Estadísticas globales
GET /api/analytics/summary
Respuesta:

```
{
 "total_reviews": 40000,
 "positive": 60,
 "neutral": 25,
 "negative": 15
}
```

2) Tendencia por fecha
GET /api/analytics/trends

3) Palabras clave más frecuentes
GET /api/analytics/keywords

Flujo de Operación del modelo API
Frontend envía reseña → API
API guarda reseña en MongoDB
API envía texto al módulo NLP
NLP procesa y regresa polaridad + keywords
API almacena el análisis en MongoDB
Frontend solicita métricas y visualizaciones
API envía datos agregados o filtrados

Beneficios del Modelo API

- Permite integrar móvil, web y dashboards fácilmente.
- Desacopla el frontend y el procesamiento NLP.
- Escalabilidad horizontal del backend.
- Seguridad mediante JWT o API Keys.
- Integración óptima con MongoDB gracias a JSON nativo.

Modelo de datos relacional (Entidades y Relaciones)

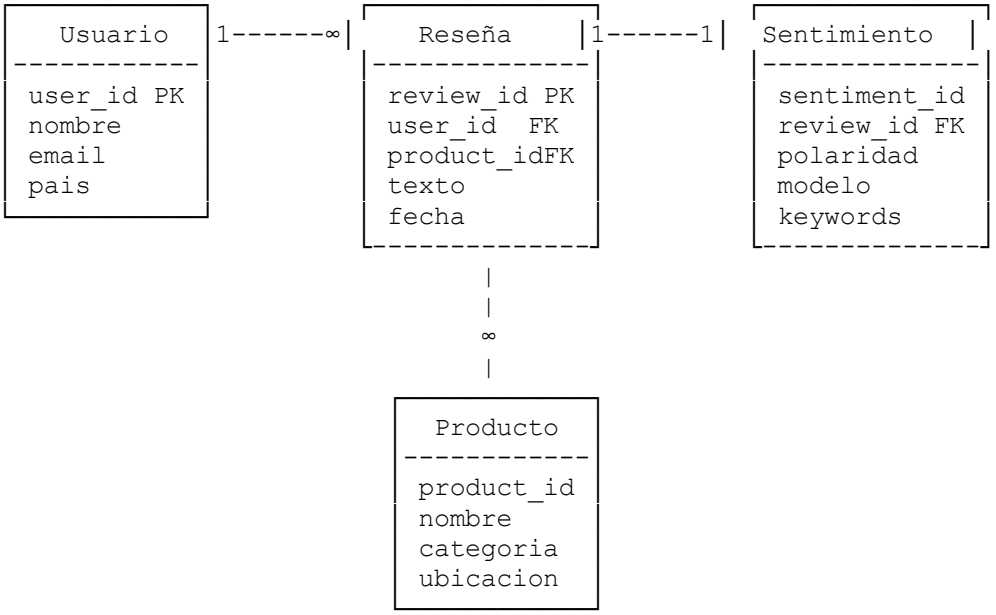
Aunque MongoDB emplea un modelo documental, su equivalente **relacional** puede representarse de la siguiente manera para propósitos conceptuales:

- **Usuario (User)**
  - *user_id (PK)*
  - *nombre*
  - *correo*
  - *país*
- **Reseña (Review)**
  - *review_id (PK)*
  - *user_id (FK)*
  - *texto*
  - *fecha*
  - *parque_id (FK)*
- **Parque (Park)**
  - *parque_id (PK)*
  - *nombre*
  - *ubicación*
- **Análisis (Sentiment)**
  - *analysis_id (PK)*
  - *review_id (FK)*
  - *polaridad* (1=positiva, 0=neutra, -1=negativa)
  - *modelo_usado*
  - *keywords*

Relaciones:

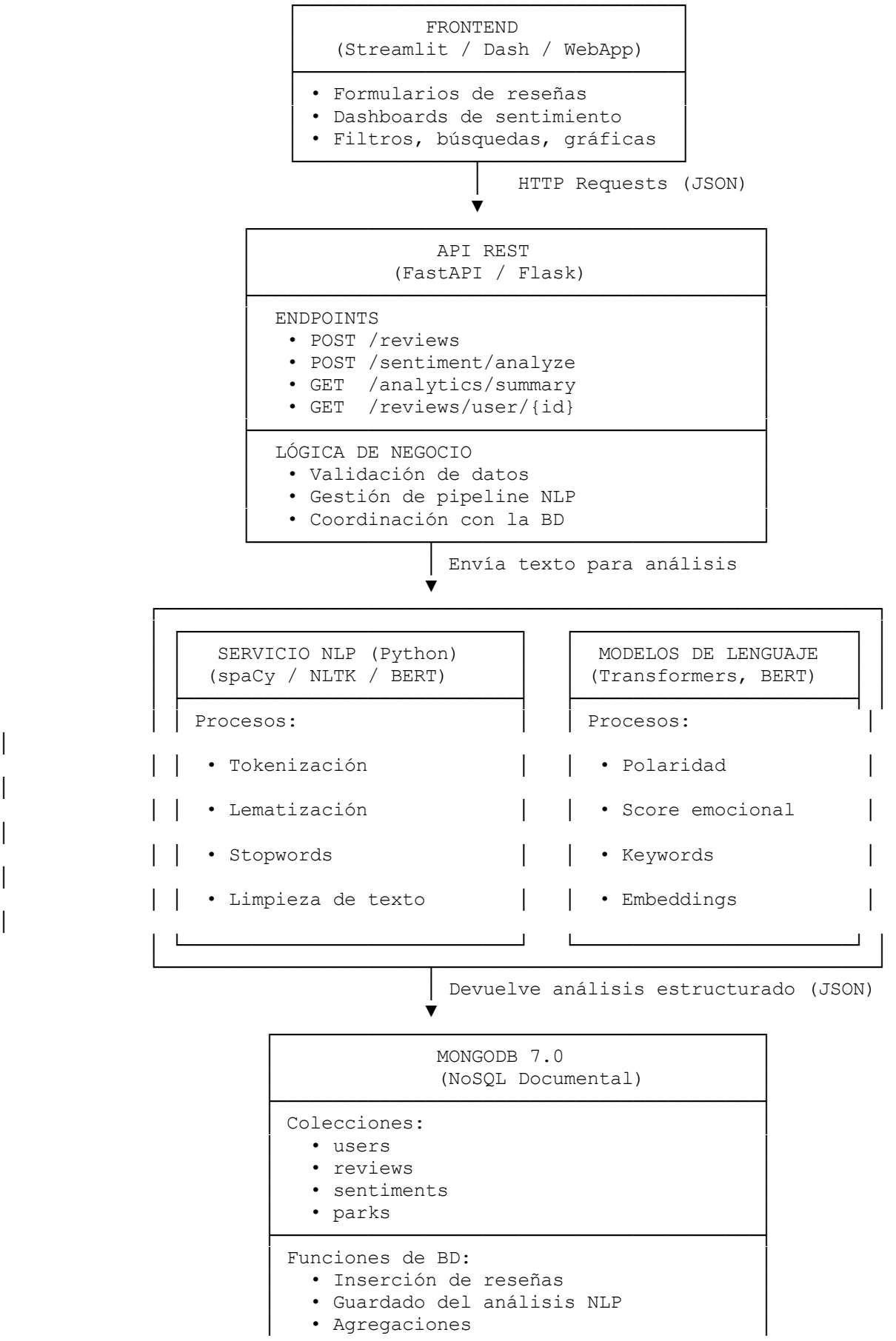
- Un **usuario** puede escribir muchas **reseñas** → Relación *1 a N*.
- Cada **reseña** pertenece a un **producto o parque** → Relación *N a 1*.
- Cada **reseña** tiene un **análisis de sentimiento asociado** → Relación *1 a 1*

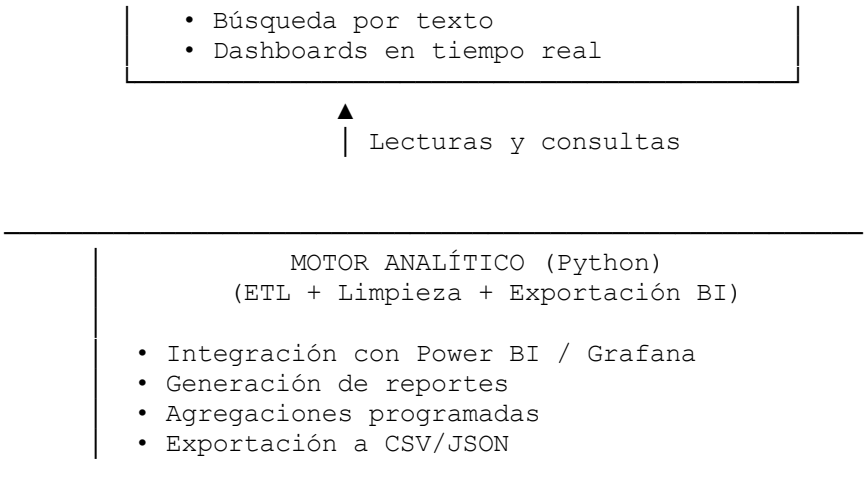
Diagrama Entidad-Relación (Versión Conceptual)



```
{
“users”: {
        “id”: “0504200198071”,
        “name”: “John”,
        “lname”: “Doe”,
        “phone”: “+50495483127”
        “email”: “johndoe@gmail.com”
        },
“reviews”: {
        “id”: “45928”,
        “rate”: 3,
        “text”: “Fue un excelente paseo por los castillos. perfecto día.”,
        “Date”: “2024-06”,
        “userid”: “0504200198071”,
        “Parks”: {
                “id”: 06,
                “name”: “Hong Kong Disneyland”,
                “thematic”: “Magic Kingdom”,
                “Location”: {
                                “address”: “123 Main Street”,
                                “city”: “Lantau Island”,
                                “state”: “Hong Kong”,
                                “country”: “China”
                        }
                },
        “sentimientos”: {
                “id”: 52,
                “polaridad”:  1,
                “modelo”: “Bert”,
                “Keywords”: [“excelente”, “perfecto”]
                }
        }
}
```

Diagrama de la Arquitectura - Sistema de Análisis de Reseñas y Sentimientos





1. Frontend
Interfaz donde el usuario:
- Ingresa reseñas
- Visualiza análisis
- Consulta dashboards

2. API REST
Punto central de comunicación:
- Recibe reseñas
- Llama al módulo NLP
- Guarda resultados en MongoDB

3. Servicio NLP
Procesamiento semántico:

- Limpia texto
- Analiza sentimiento
- Extrae palabras clave

4. MongoDB
Base de datos documental:

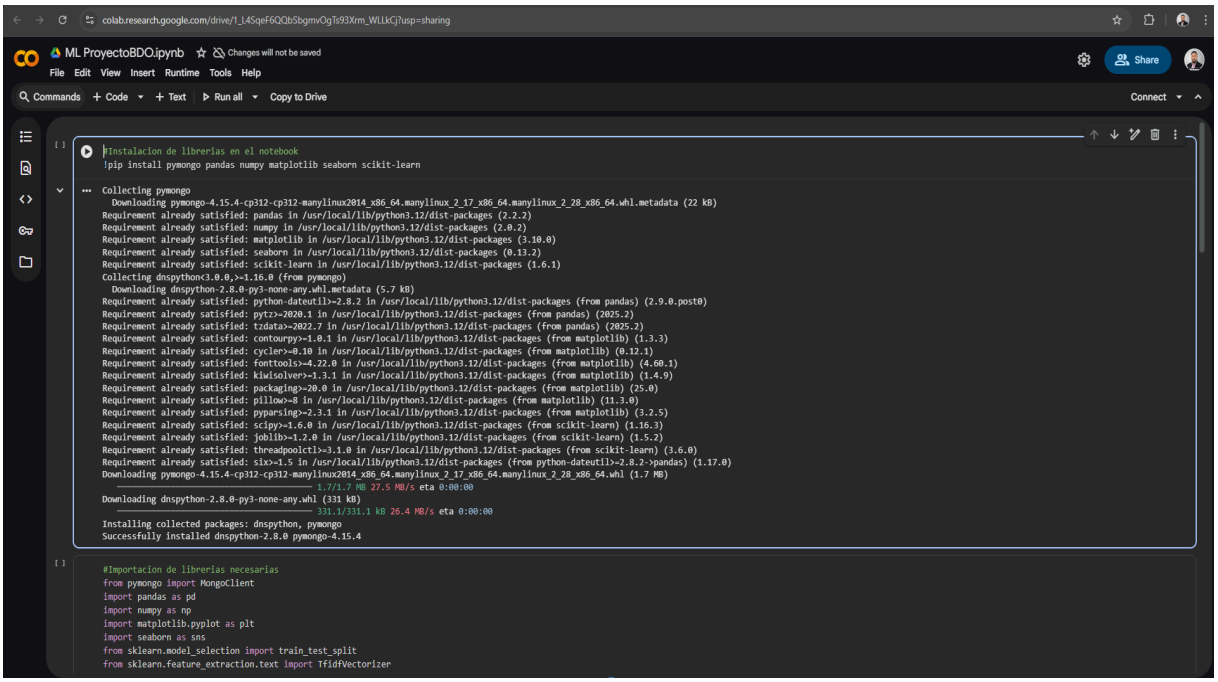
- Guarda reseñas originales
- Guarda análisis de sentimiento
- Permite agregaciones y búsquedas

5. Motor analítico
Conexión con BI:
- Dashboards
- Reportes
- ETL automatizado

Colab Machine Learning Aplicado

ML ProyectoBDO.ipynb

Ejecución de codigo del análisis

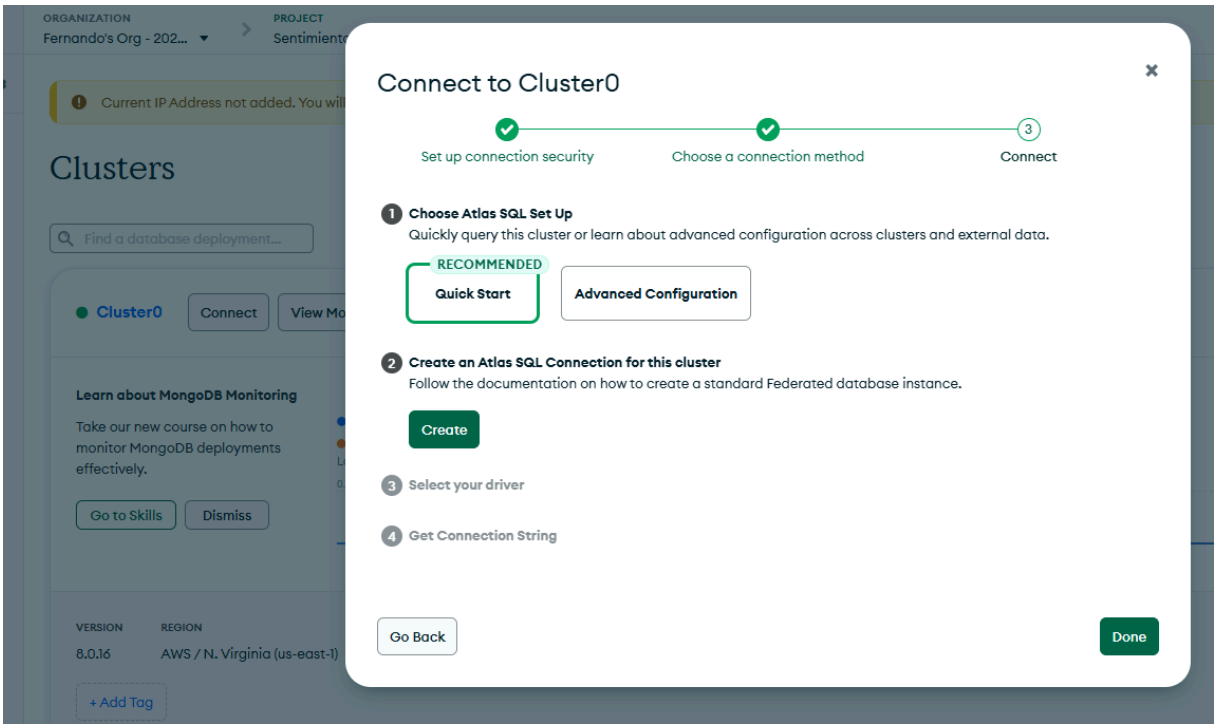


Resultados en tabla de sentimientos

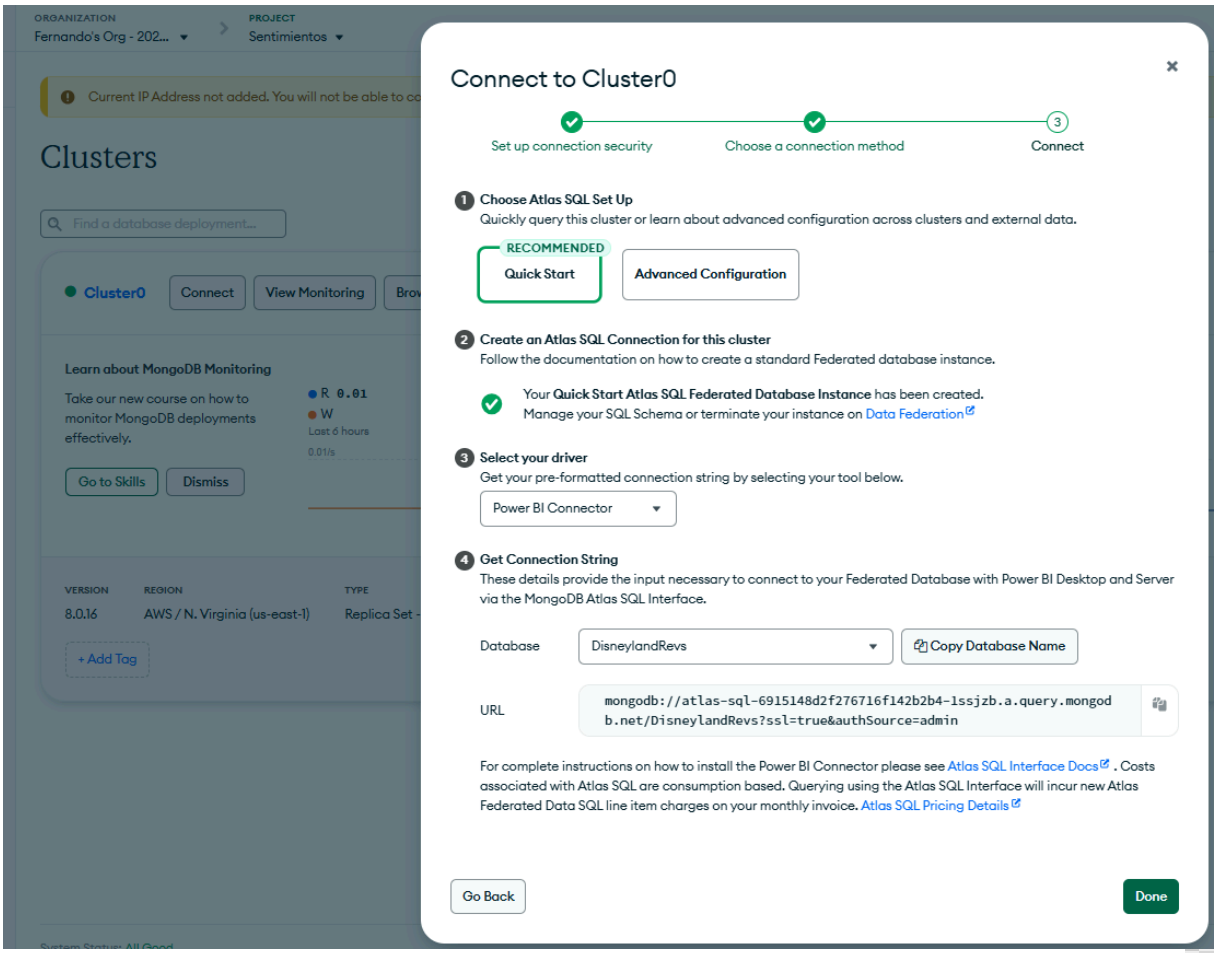

```
_id: ObjectId('69181793dcdc70768af7fe6f')
Review_ID : 670591897
sentiment : "positive"
model_used : "Logistic Regression"
▸ keywords : Array (5)

_id: ObjectId('69181793dcdc70768af7fe70')
Review_ID : 670585330
sentiment : "positive"
model_used : "Logistic Regression"
▸ keywords : Array (5)

_id: ObjectId('69181793dcdc70768af7fe71')
Review_ID : 670574142
sentiment : "negative"
model_used : "Logistic Regression"
▸ keywords : Array (5)

_id: ObjectId('69181793dcdc70768af7fe72')
Review_ID : 670571027
sentiment : "negative"
model_used : "Logistic Regression"
▾ keywords : Array (5)
  0: "signage"
  1: "knows"
  2: "building"
  3: "rooms"
  4: "terrible"
```

Dashboard de salida - Power BI

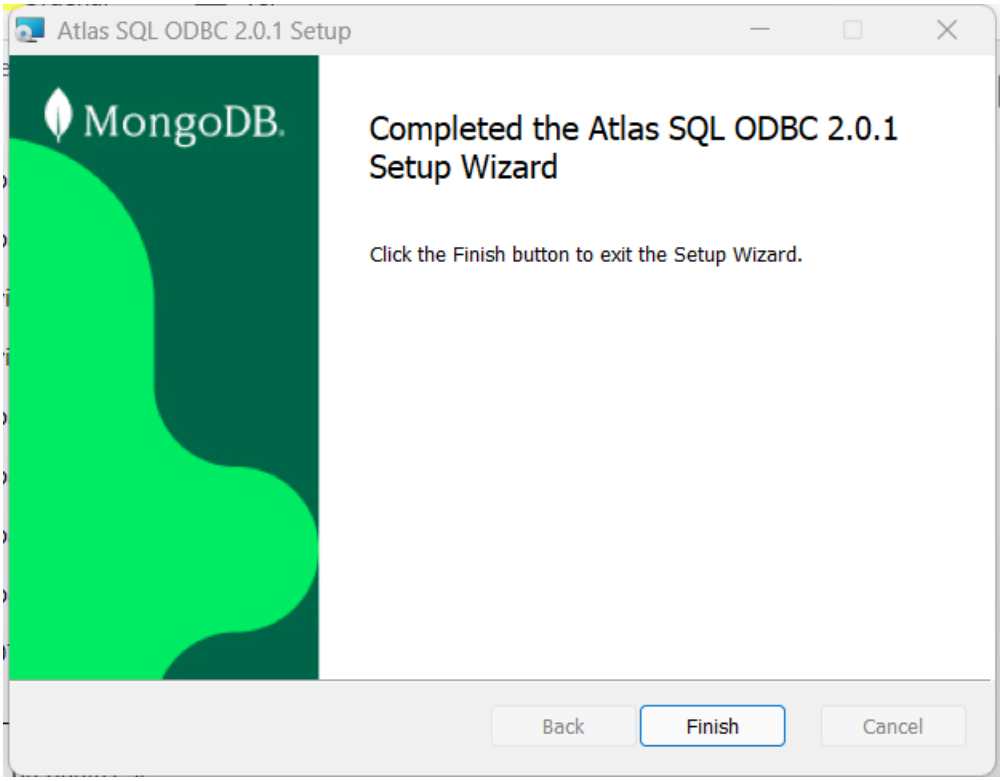
Paso 1 Create Atlas SQL Connection



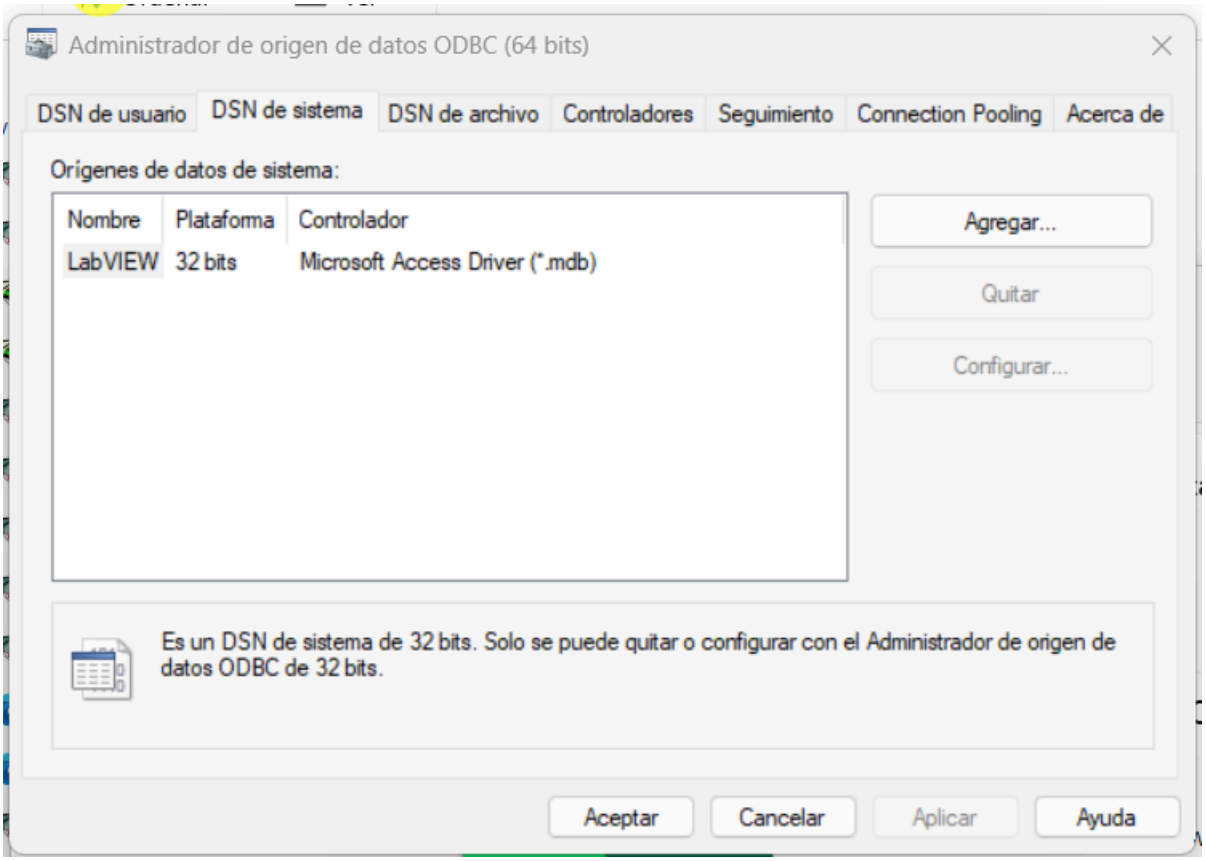
Paso 2 Elegir un método de conexión



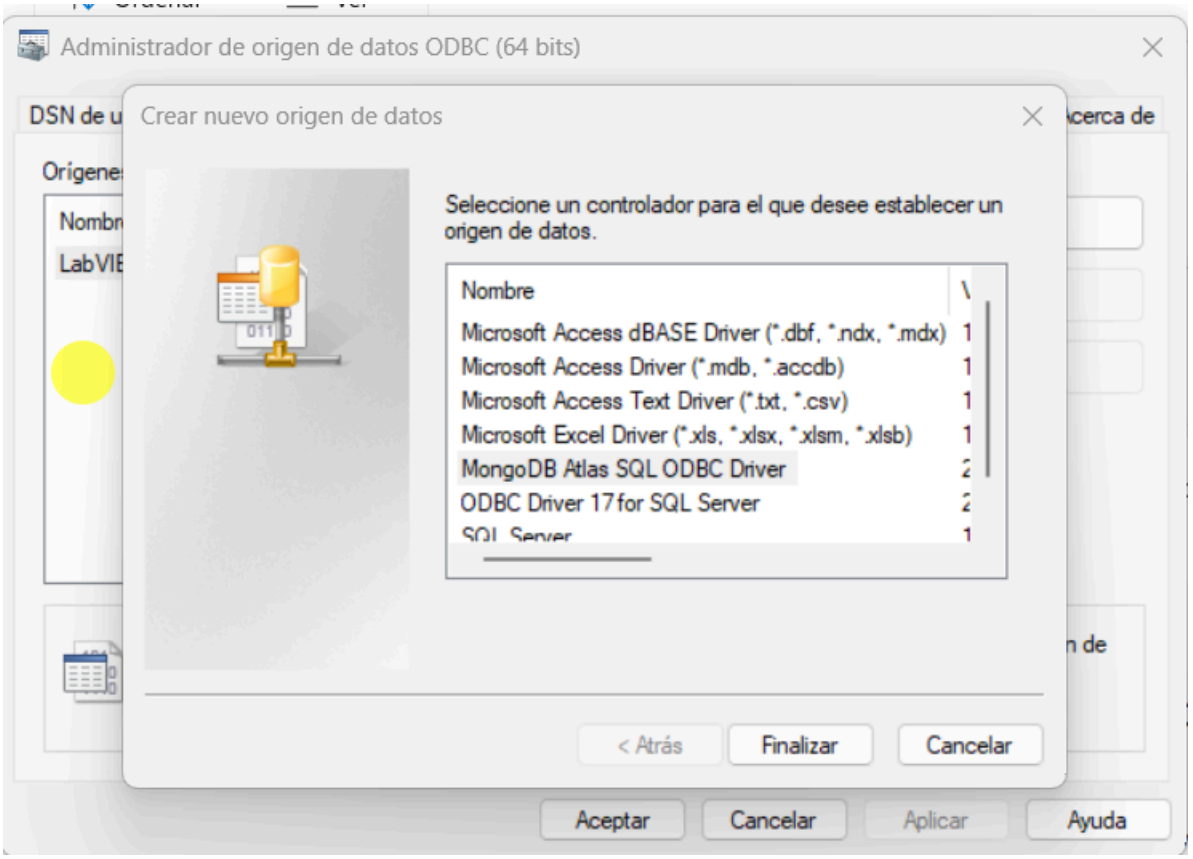
Paso 3 Instalación Atlas SQL OBDC



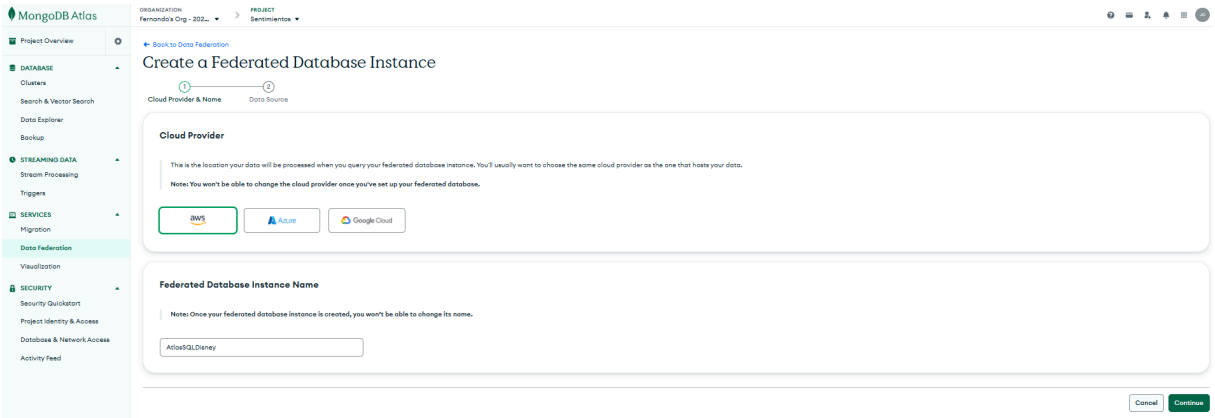
Paso 4 Administrador de Origen de Datos



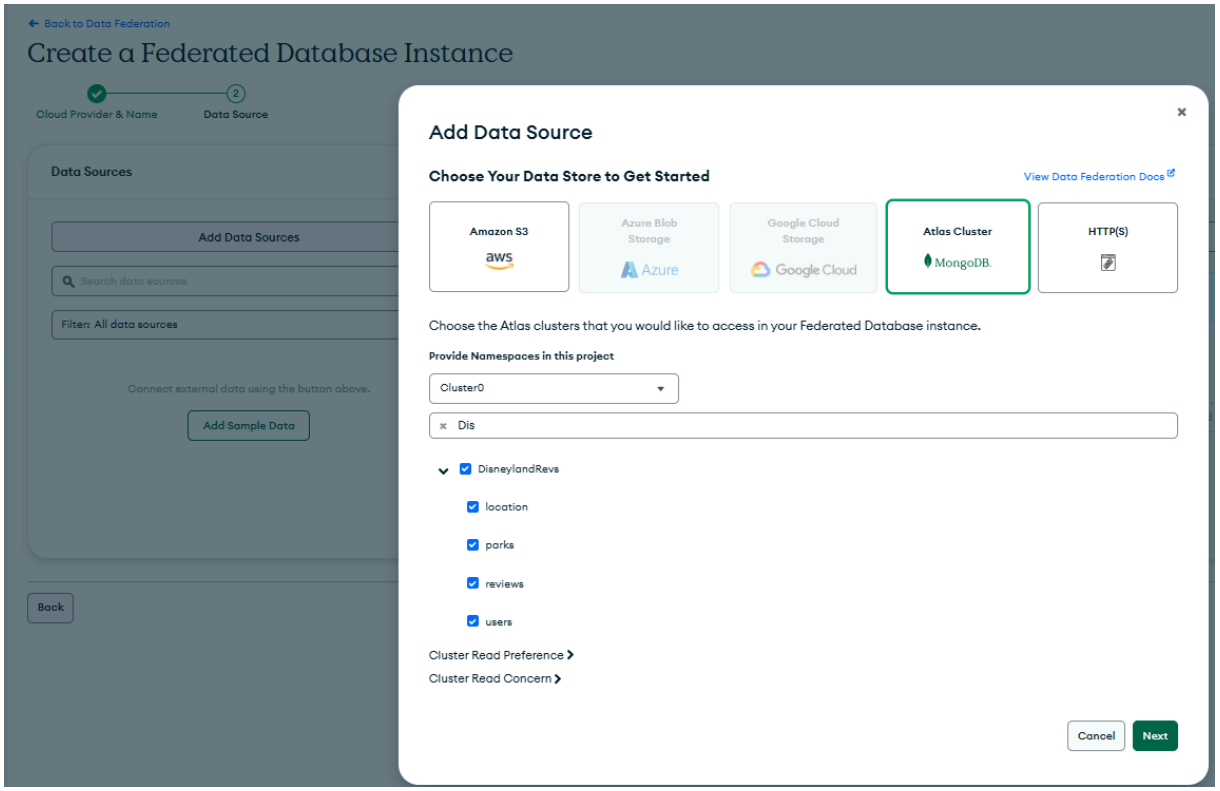
Paso 5 clic en agregar MongoDB Atlas SQL ODBC Driver



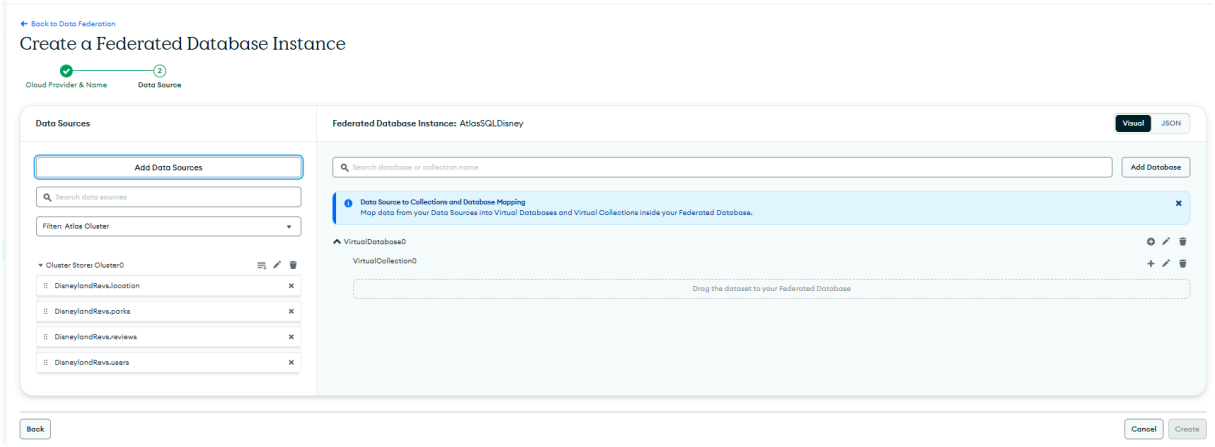
Paso 6 Create a Federal Databaase Instance



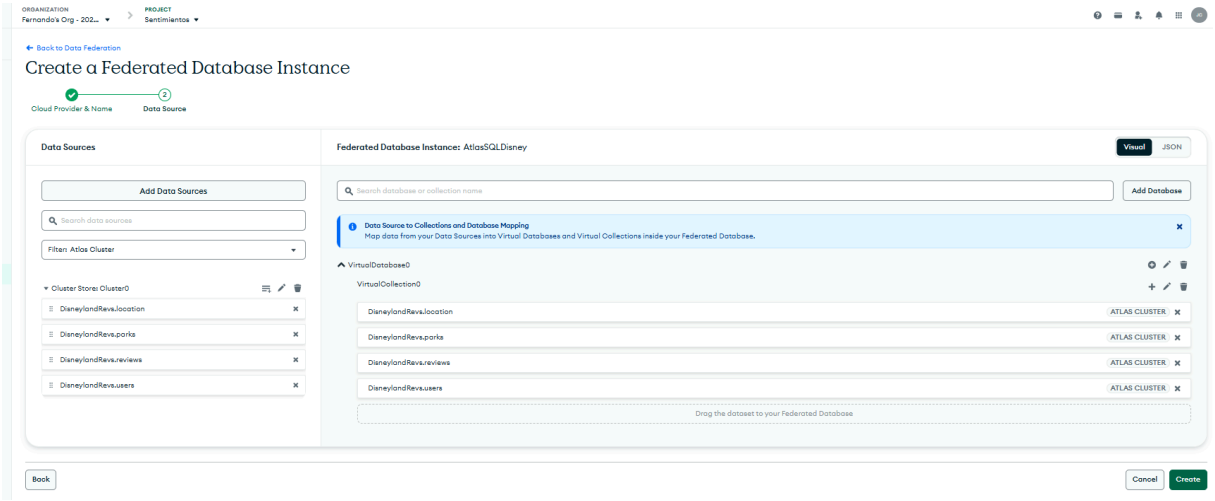
Paso 7 Add Data Source



Paso 8 Muestra el cluster con la base de datos



Paso 9 Enviar los datasets del Cluster Store hacia el VirtualCollections para Power BI



Dashboard Power BI

Conclusión

El presente proyecto permitió aplicar los conocimientos adquiridos en la asignatura Base de Datos Organizacionales mediante el diseño, implementación y justificación de un sistema de almacenamiento y análisis orientado a la gestión de grandes volúmenes de datos no estructurados.

La elección de MongoDB como sistema de gestión de base de datos documental demostró ser una decisión técnica acertada, ya que su estructura flexible basada en documentos JSON facilita el almacenamiento, consulta e integración de datos con herramientas analíticas como Python y Power BI.

En términos conceptuales, el proyecto integra los principios de gestión de datos, modelado NoSQL y análisis relacional, mostrando una visión integral sobre cómo las bases de datos pueden contribuir a mejorar la experiencia del cliente y optimizar la toma de decisiones en el ámbito empresarial.

La solución sugerida no solo alcanza los objetivos establecidos, sino que también prepara el camino para futuras expansiones, como la inclusión de modelos predictivos más avanzados, análisis en varios idiomas o integraciones en tiempo real. Este trabajo representa un ejemplo claro de cómo la tecnología de datos puede transformar la manera en que las organizaciones interpretan y valoran la voz de sus usuarios

El uso combinado de MongoDB**,** Python y proporciona una plataforma robusta, escalable y flexible para analiz ar grandes volúmenes de reseñas de usuarios. Esta arquitectura favorece la integración con herramientas de inteligencia de negocio y aprendizaje automático, permitiendo a las organizaciones comprender mejor las percepciones de sus clientes y actuar estratégicamente sobre la base de datos reales.

MongoDB responde a la necesidad de manejar información semiestructurada y en crecimiento constante, con una integración fluida hacia el análisis de sentimientos.
Aunque el modelo relacional explica las dependencias entre entidades, la adopción del modelo documental permite mayor agilidad en el desarrollo, escalabilidad y facilidad de integración con entornos de machine learning y NLP.

.

Referencias

- Chillar, R. (2020). *Disneyland Reviews Dataset*. Kaggle. https://www.kaggle.com/datasets/arushchillar/disneyland-reviews

- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL). https://doi.org/10.48550/arXiv.1810.04805

- Feldman, R. (2013). *Techniques and Applications for Sentiment Analysis*. Communications of the ACM, 56(4), 82–89. https://doi.org/10.1145/2436256.2436274

- MongoDB Inc. (2024). *MongoDB Documentation*. MongoDB. https://www.mongodb.com/doc

- Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00416ED1V01Y201204HLT016