DOCUMENTACION DEL PROYECTO

Análisis de Popularidad y Éxito Financiero de Contenidos en Netflix utilizando Datos de IMDb.

INTEGRANTES:

Diego Fernando Díaz Hernández

PONTIFICIA UNIVERSIDAD JAVERIANA BOGOTÁ D.C.

FACULTAD DE INGENIERÍA

Tópicos Avanzados De Bases De Datos

BOGOTÁ D.C.

MAYO DE 2024

Contenido

Introducción

Descripción del proyecto

El proyecto tiene como objetivo analizar la popularidad y el éxito financiero de las películas y series en Netflix, utilizando datos adicionales de IMDb. Se busca identificar tendencias y proporcionar insights valiosos para los creadores de contenido y las plataformas de streaming.

Contexto

En la actualidad, las plataformas de streaming recopilan una gran cantidad de datos sobre el contenido que ofrecen y su desempeño. La integración de datos externos, como las calificaciones y reseñas de IMDb, puede enriquecer el análisis y proporcionar insights más profundos.

Alcance

El proyecto abarca la extracción, transformación y análisis de datos de Netflix e IMDb. Se implementa una arquitectura que incluye almacenamiento, procesamiento de datos y una capa de API para acceder a los resultados del análisis.

Objetivos

Objetivos Generales

El proyecto se centra en el análisis de la popularidad y el éxito financiero de los contenidos disponibles en Netflix. Este análisis se realiza mediante la integración y comparación de datos internos de Netflix con datos externos provenientes de IMDb.

Los datasets utilizados en este análisis son los siguientes:

- **Netflix**: Datos disponibles hasta el año 2021.
- **IMDb**: Datos disponibles hasta el año 2023.

1. **Analizar la popularidad y el éxito financiero de los contenidos en Netflix**:
    - Este objetivo busca entender qué hace que ciertos contenidos sean más populares y financieramente exitosos que otros.
2. **Proporcionar insights valiosos sobre las tendencias y preferencias del público**:
    - Al analizar los datos, el proyecto pretende descubrir las preferencias del público y cómo estas han evolucionado con el tiempo.

Objetivos Específicos

Para alcanzar los objetivos generales, se han establecido los siguientes objetivos específicos

1. **Identificar cómo han cambiado las tendencias en los géneros populares a lo largo de los años**:
    - Este objetivo se enfoca en analizar la evolución de los géneros populares en Netflix. Se examinan datos históricos para detectar cambios en las preferencias del público y cómo estos han afectado la producción y oferta de contenido.
2. **Evaluar la relación entre las calificaciones de IMDb y la popularidad en Netflix**:

- Este objetivo investiga cómo las calificaciones y reseñas de IMDb, una fuente externa y objetiva, se correlacionan con la popularidad de los contenidos en Netflix. Se analiza si existe una relación significativa entre las dos métricas y cómo esta relación puede influir en las decisiones de marketing y producción.

Atributos de calidad

La arquitectura está diseñada para manejar un aumento en la carga de trabajo mediante el uso de servicios en la nube que permiten una escalabilidad horizontal y vertical. Como es un ambiente educativo y estos servicios no son gratuitos, no se espera una validación con ATAM u otro mecanismo de evaluación de estos atributos de calidad.

Escalabilidad

La escalabilidad es la capacidad de un sistema para manejar un aumento en la carga de trabajo, ya sea mediante la adición de recursos (escalado vertical) o mediante la adición de más nodos (escalado horizontal), sin comprometer el rendimiento.

La arquitectura propuesta está diseñada para escalar tanto horizontal como verticalmente mediante el uso de servicios en la nube.

Implementación:

- **Azure Storage Account y Blob Storage**: Pueden escalar automáticamente para manejar grandes volúmenes de datos sin necesidad de intervención manual.
- **Azure DataFactory**: Permite crear pipelines que procesan grandes cantidades de datos de manera eficiente, adaptándose a un aumento en la carga.
- **MongoDB Atlas**: Ofrece escalabilidad automática y gestiona el crecimiento de los datos mediante la adición de nodos y la distribución de la carga.
- **Redis Cache**: Ayuda a manejar aumentos en la carga de consultas mediante el almacenamiento en caché de respuestas frecuentes.
- **Azure App Service**: La API Client se despliega en contenedores Docker, lo que facilita su escalabilidad y despliegue en múltiples instancias.

Medición:

- Monitoreo continuo del uso de CPU, memoria y ancho de banda en Azure y MongoDB Atlas.
- Ajuste automático de instancias y recursos según sea necesario para mantener el rendimiento óptimo.

Rendimiento

El rendimiento se refiere a la rapidez con la que un sistema puede procesar y recuperar datos, asegurando una experiencia de usuario fluida y eficiente.

Implementación:

- **Redis Cache**: Mejora el tiempo de respuesta de las consultas más frecuentes, reduciendo la latencia.
- **Azure DataFactory**: Optimiza el procesamiento y transformación de datos.
- **API Client dockerizado**: Desarrollado utilizando prácticas de optimización para asegurar respuestas rápidas y desplegado en contenedores Docker para asegurar consistencia y rapidez en el despliegue, además se implementará con un enfoque hexagonal por una separación de capas y principios SOLID asegurando una separación de responsabilidades adecuada.

Medición:

- **Tiempo de respuesta de la API**: Monitoreo de la rapidez con la que la API responde a las solicitudes.
- **Latencia en la consulta de datos**: Medición del tiempo que tarda una consulta en completarse.
- **Tasa de aciertos del caché**: Monitoreo de la efectividad del caché de Redis.

Disponibilidad

La disponibilidad es la medida en que un sistema está operativo y accesible cuando se necesita, minimizando el tiempo de inactividad.

Implementación:

- **Azure SQL y MongoDB Atlas**: Ofrecen alta disponibilidad con opciones de replicación y recuperación ante desastres, multi regiones.
- **API Client dockerizado**: Desplegado en contenedores Docker, facilitando la escalabilidad y redundancia para ser desplegado en diferentes servicios en caso de ser necesario.

Medición:

- **Monitoreo del tiempo de actividad**: Uso de herramientas de monitoreo para asegurar un alto tiempo de actividad como App Insight, como es un ambiente educativo no se implementará en la app service por costos.
- **Implementación de políticas de recuperación ante fallos**: Estrategias para minimizar el impacto de posibles interrupciones, como es un ambiente educativo no se implementará en los servicios como tal por costos.

Seguridad

La seguridad se refiere a las medidas implementadas para proteger los datos y asegurar que solo usuarios autorizados tengan acceso, protegiendo así la integridad y confidencialidad de los datos.

Implementación:

- **SQL Database**: Maneja la autenticación y autorización de usuarios con mecanismos seguros y listas blancas, para este ejercicio tendremos lista blanca de servicios en azure y dirección ip de mi computador.
- **JWT (JSON Web Tokens)**: Utilizados para la autenticación de usuarios en la API.
- **CSV Fuentes**: ya que son públicos no son necesarios utilizar mecanismos de seguridad más allá de la implementación y alojamiento en el Access Storage.

Medición:

- **Auditorías de seguridad regulares**: Evaluaciones periódicas para identificar y mitigar vulnerabilidades, podría utilizarse mecanismos de inspección de código estático para detectar vulnerabilidades de seguridad.

Mantenibilidad

La mantenibilidad es la capacidad de un sistema para ser modificado y actualizado de manera eficiente, permitiendo ajustes y mejoras sin interrumpir el servicio.

Implementación:

- **Uso de contenedores Docker**: Para el despliegue de la API Client, facilitando las actualizaciones y la escalabilidad.
- **Azure DataFactory**: Permite la modificación fácil de pipelines de procesamiento de datos.
- **Documentación clara y completa**: Repositorio para la estructura de las colecciones y scripts para SQL.

Medición:

- **Tiempo necesario para aplicar actualizaciones**: Evaluación del tiempo que toma implementar cambios.
- **Facilidad para agregar nuevos módulos o servicios**: Monitoreo de la adaptabilidad de la arquitectura por la separación clara de responsabilidades, aunque tenemos un trade off negativo hacia el acoplamiento con Microsoft Azure en el caso del Azure data factory.

Confiabilidad

La confiabilidad es la capacidad de un sistema para proporcionar datos precisos y consistentes a los usuarios, asegurando que los servicios siempre estén disponibles y funcionando correctamente.

Implementación:

- **Fuentes de Datos Confiables:** Uso de datos provenientes de fuentes confiables y bien conocidas como Netflix e IMDb, que proporcionan datos precisos y consistentes.

- **MongoDB Atlas**: Uso de replicación y alta disponibilidad para asegurar la integridad de los datos, como es un ambiente educativo y tenemos restricciones económicas no será implementado.

Medición:

- **Verificación de la Precisión de los Datos:** Revisión periódica de la precisión de los datos provenientes de Netflix e IMDb para asegurar la fiabilidad de las fuentes de datos.
- **Monitoreo de la integridad y consistencia de los datos**: Asegurando que los datos no se corrompan y siempre estén disponibles.

Descripción de la arquitectura

Diagrama de la arquitectura

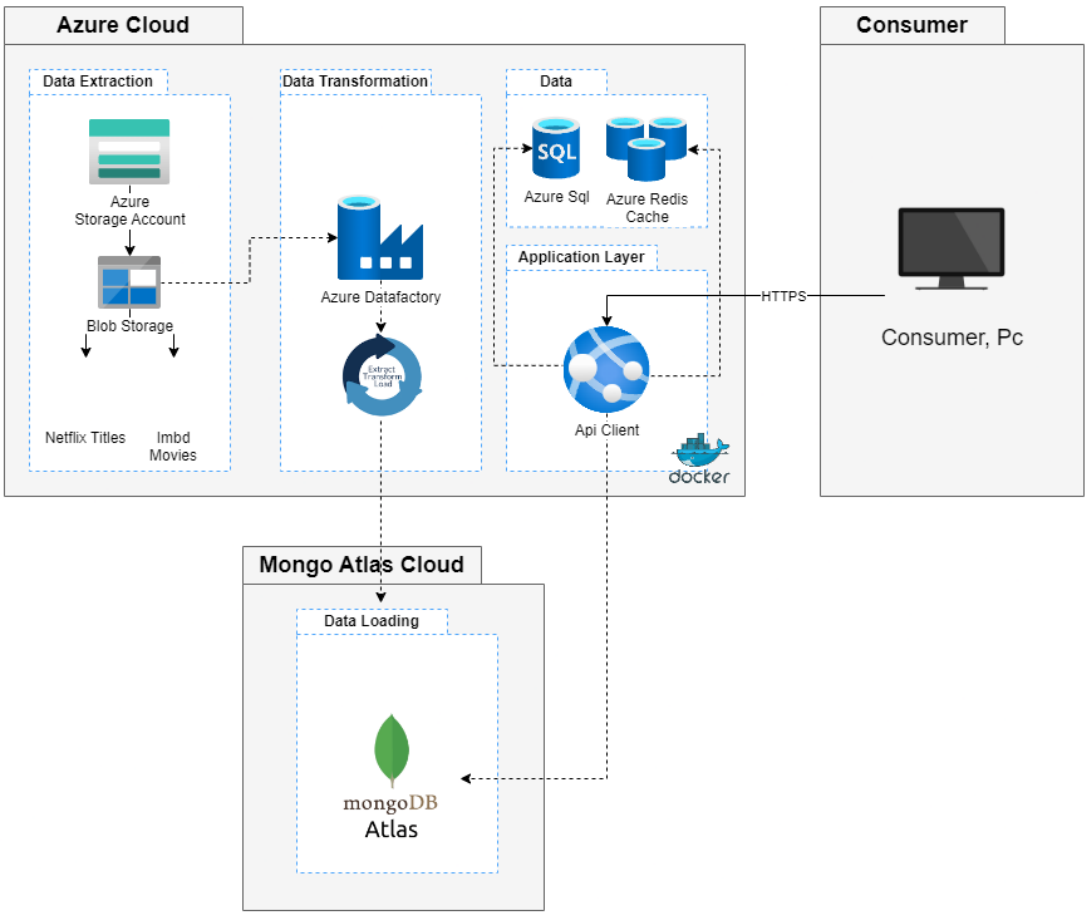

Ilustración 1 de la arquitectura a nivel físico

Componentes

La arquitectura del sistema se compone de varios componentes principales, cada uno con responsabilidades específicas:

1. **Azure Storage Account**
    - **Responsabilidad**: Almacenar los archivos CSV de Netflix e IMDb en Blob Storage.
    - **Función**: Proporciona un almacenamiento escalable y duradero para los datos brutos.

- **Diccionario de datos Netlix**:

  Incluye tanto películas como programas de televisión, lo que permite analizar la diversidad del contenido en la plataforma.

  Contiene datos sobre el elenco, el país de origen y las calificaciones, proporcionando una visión completa del contenido disponible.

Campo	Traducción	Tipo de Dato	Descripción	Ejemplo
show_id	Identificador del Show	string	Identificador único del show	"s1"
type	Tipo de Contenido	string	Tipo de contenido (Movie o TV Show)	"Movie"
title	Título	string	Título del contenido	"Dick Johnson Is Dead"
director	Director	string	Director del contenido	"Kirsten Johnson"
cast	Elenco	string	Elenco del contenido	"Ama Qamata, Khosi Ngema, etc."
country	País	string	País de origen del contenido	"United States"
date_added	Fecha de Adición	date	Fecha en que el contenido se añadió a Netflix	"September 25, 2021"
release_year	Año de Lanzamiento	number	Año de lanzamiento del contenido	2020
rating	Calificación	string	Calificación del contenido	"PG-13"
duration	Duración	string	Duración del contenido	"90 min"
listed_in	Géneros/Categorías	string	Géneros y categorías en que el contenido está listado	"Documentaries"
description	Descripción	string	Descripción del contenido	"As her father nears the end of his life..."

Tabla 1. Diccionario de datos Dataset Netflix

- **Diccionario de datos complementarios IMDb**:

  Incluye datos detallados sobre películas, como presupuesto y ganancia, que no están presentes en el dataset de Netflix.

  Proporciona una perspectiva sobre la popularidad y el éxito financiero de las películas.

Campo	Traducción	Tipo de Dato	Descripción	Ejemplo
names	Título del contenido	string	Título del contenido	"Creed III"
date_x	Fecha de lanzamiento	date	Fecha de lanzamiento	"03/02/2023"
score	Calificación en IMDb	number	Calificación en IMDb	73
genre	Géneros del contenido	string	Géneros del contenido	"Drama, Action"
overview	Resumen del contenido	string	Resumen del contenido	"After dominating the boxing world..."
crew	Elenco y equipo del contenido	string	Elenco y equipo del contenido	"Michael B. Jordan, Tessa Thompson"
orig_title	Título original del contenido	string	Título original del contenido	"Creed III"
status	Estado del contenido	string	Estado del contenido (Released,	"Released"
orig_lang	Idioma original del contenido	string	Idioma original del contenido	"English"
budget_x	Presupuesto del contenido	number	Presupuesto del contenido	75000000
revenue	Ingresos del contenido	number	Ingresos del contenido	271616668
country	País de origen del contenido	string	País de origen del contenido	"AU"

Tabla 2. Diccionario de datos Dataset IMDb

2. **Azure DataFactory**

   - **Responsabilidad**: Procesar y transformar los datos CSV en el formato JSON adecuado para MongoDB.
   - **Función**: Facilita la creación de pipelines que automatizan la extracción, transformación y carga (ETL) de datos.

3. **MongoDB Atlas**

   - **Responsabilidad**: Almacenar los documentos JSON con los datos combinados de Netflix e IMDb.

- **Función**: Proporciona una base de datos NoSQL escalable y de alto rendimiento que almacena los datos transformados.
- **Json almacenado en la colección después de la transformación**:

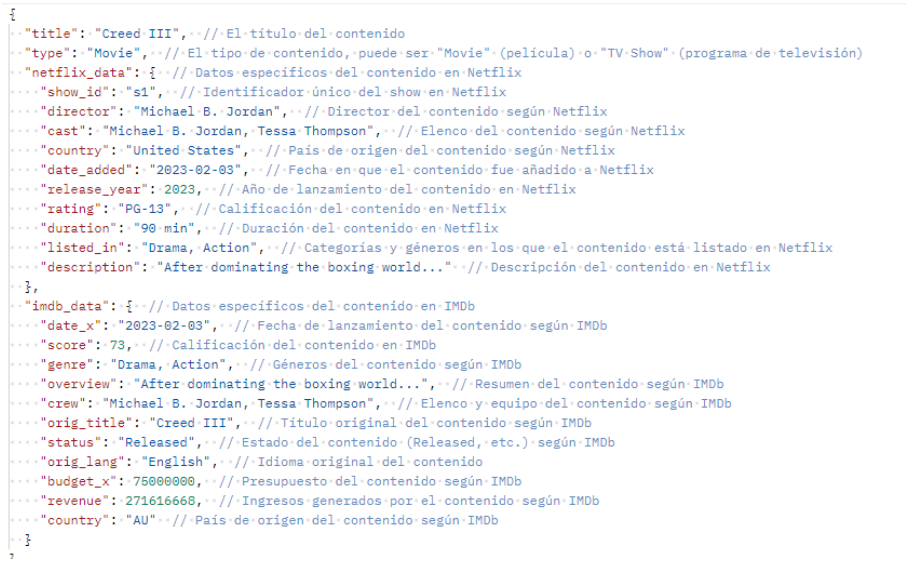

Ilustración 2. Json resultado después de la transformación

4. **API Client (App service contenerizado)**
    - **Responsabilidad**: Desarrollar una API que expone métodos para acceder a los datos de MongoDB.
    - **Función**: Facilita el acceso a los datos mediante solicitudes HTTP, proporcionando endpoints para diversas consultas.
5. **SQL Database**
    - **Responsabilidad**: Manejar la autenticación y autorización de usuarios.
    - **Función**: Almacena la información de los usuarios y gestiona sus credenciales y permisos.
6. **Redis Cache**
    - **Responsabilidad**: Mejorar el rendimiento cacheando respuestas de las API más solicitadas.
    - **Función**: Reduce la carga en MongoDB al almacenar en caché las respuestas de consultas frecuentes, mejorando así el tiempo de respuesta.

Flujo de datos

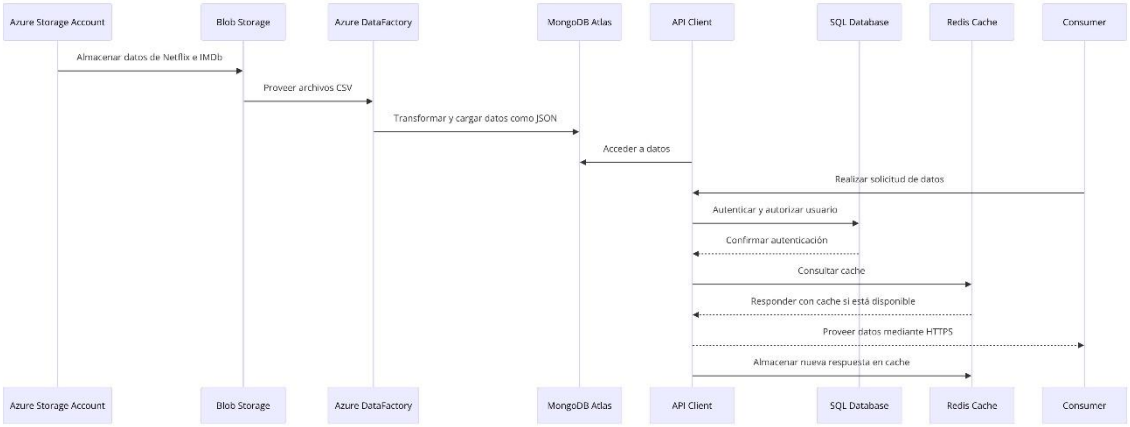

Ilustración 3 de flujo de datos con diagrama de secuencia

El flujo de datos a través del sistema sigue estos pasos:

1. **Almacenamiento de Datos**:
    - Los datos de Netflix e IMDb se almacenan inicialmente en **Azure Storage Account** como archivos CSV en Blob Storage.
2. **Procesamiento y Transformación de Datos**:
    - **Azure DataFactory** toma los archivos CSV almacenados y los procesa, transformándolos en documentos JSON adecuados para MongoDB.
3. **Carga de Datos**:
    - Los documentos JSON generados se cargan en **MongoDB Atlas**. Este paso asegura que los datos están listos para ser consultados.
4. **Acceso a los Datos**:
    - La **API Client**, desplegada en contenedores Docker, accede a los datos almacenados en MongoDB Atlas y los expone mediante endpoints RESTful.
5. **Autenticación de Usuarios**:
    - Los usuarios que acceden a la API son autenticados y autorizados utilizando la **SQL Database**. Esto asegura que solo usuarios autorizados pueden acceder a los datos.
6. **Cacheo de Respuestas**:
    - **Redis Cache** almacena respuestas de consultas frecuentes para mejorar el rendimiento y reducir la latencia en el acceso a los datos.
7. **Consumo de Datos**:
    - Los consumidores (usuarios y aplicaciones) acceden a los datos a través de HTTPS, utilizando los endpoints proporcionados por la API.

Tecnologías Utilizadas

Lenguajes de programación

- **C#**: Utilizado para el desarrollo del API Client y la lógica del servidor. C# es un lenguaje de programación moderno y de propósito general que es parte del ecosistema .NET, ideal para aplicaciones de alto rendimiento y gran escala.
- **SQL (Structured Query Language)**: Utilizado para interactuar con bases de datos relacionales, como Azure SQL Database. SQL es un lenguaje estándar para gestionar y manipular bases de datos relacionales.
- **MongoDB Query Language (MQL)**: Utilizado para interactuar con MongoDB. MQL es el lenguaje de consulta específico de MongoDB, utilizado para realizar operaciones CRUD (Create, Read, Update, Delete) y consultas en bases de datos NoSQL.

Frameworks y librerías

Estas fueron los siguientes frameworks y librerías seleccionadas:

- **.NET 8**: Utilizado como el framework principal para desarrollar la aplicación. .NET 8 proporciona una plataforma unificada para crear aplicaciones de alta calidad, ofreciendo soporte para diferentes tipos de aplicaciones y servicios.
- **Entity Framework Core**: Una ORM (Object-Relational Mapper) para .NET, utilizada para interactuar con bases de datos relacionales de manera más intuitiva y productiva.
- **MongoDB Driver**: Librería oficial de MongoDB para .NET, utilizada para interactuar con la base de datos MongoDB Atlas.
- **StackExchange.Redis**: Librería para interactuar con Redis en .NET, utilizada para implementar el almacenamiento en caché.
- **AutoMapper**: Librería utilizada para mapear objetos entre diferentes tipos. Facilita la conversión de objetos de dominio a DTOs (Data Transfer Objects) y viceversa.
- **Swagger (Swashbuckle)**: Utilizado para documentar y probar la API. Swagger genera documentación interactiva que facilita a los desarrolladores entender y consumir la API.

Plataformas y servicios

Estas fueron las siguientes plataformas y servicios seleccionados:

- **Azure Storage Account y Blob Storage**: Utilizados para el almacenamiento de archivos CSV de Netflix e IMDb. Blob Storage proporciona almacenamiento de objetos escalable y duradero para datos no estructurados.

- **Azure DataFactory**: Servicio de integración de datos basado en la nube que permite crear, programar y orquestar flujos de trabajo de datos. Utilizado para procesar y transformar los datos CSV en JSON.
- **MongoDB Atlas**: Servicio de base de datos en la nube que proporciona una solución de base de datos NoSQL escalable y de alto rendimiento. Utilizado para almacenar los documentos JSON con los datos combinados de Netflix e IMDb.
- **Azure SQL Database**: Base de datos relacional en la nube utilizada para manejar la autenticación y autorización de usuarios. Proporciona características de alta disponibilidad y recuperación ante desastres.
- **Docker**: Plataforma utilizada para desplegar la API Client en contenedores. Docker permite empaquetar aplicaciones y sus dependencias en un contenedor portátil que puede ejecutarse en cualquier entorno.
- **Redis Cache**: Utilizado para mejorar el rendimiento cacheando respuestas de las API más solicitadas. Redis es conocido por su alta velocidad y eficiencia en el manejo de datos en memoria.

Configuración e instalación

Requisitos previos

Nota: Es importante contar con experiencia previa media-baja para estas configuraciones en la plataforma de Azure, se puede detallar pero los servicios cambian constantemente, con esto se sugiere que se tengan conocimientos previos en la plataforma, como aun no tengo servicios montados esta sección va con estos pasos iniciales como “pasos” pero serán alimentados en la siguiente entrega.

Antes de la instalación y configuración del proyecto, es necesario asegurarse de tener las siguientes herramientas y software instalados:

1. **Herramientas y Software Necesarios:**
    - **.NET 8 SDK**: Para desarrollar y ejecutar la aplicación.
    - **Docker**: Para desplegar la API Client en contenedores.
    - **Azure Account**: Para utilizar servicios como Azure Storage Account, Azure DataFactory, Azure Rediscache y Azure SQL Database.
    - **MongoDB Atlas Account**: Para gestionar la base de datos MongoDB en la nube.
    - **Redis**: Corriendo para el almacenamiento en caché.
    - **Git**: Para clonar el repositorio del proyecto.
    - **Visual Studio o Visual Studio Code**: Como entorno de desarrollo.

Instrucciones de instalación

Se deben seguir estos pasos para instalar y configurar el proyecto:

1. **Clonar el Repositorio:**

```
git clone <URL_del_repositorio>
cd <nombre_del_proyecto>
```

2. **Instalar Dependencias:**
    - Navega al directorio del proyecto y ejecuta el siguiente comando para restaurar las dependencias del proyecto:

```
dotnet restore
```

3. **Configurar Docker:**
    - Asegúrate de que Docker esté instalado y en ejecución.
    - Construye y ejecuta los contenedores Docker:

```
docker-compose up --build
```

4. **Configurar Azure Services:**
    - **Azure Storage Account**:
        - Crea una cuenta de almacenamiento en Azure y configura un Blob Storage para almacenar los archivos CSV.
    - **Azure DataFactory**:
        - Configura un Data Factory en Azure para procesar y transformar los datos. (Acá se adjuntaran los pipelines en json para la ejecución, tanto ingesta como transformación)
    - **Azure SQL Database**:
        - Crea una base de datos SQL en Azure y configura las tablas necesarias para la autenticación de usuarios. (acá se adjuntara scritps para la SQL)
5. **Configurar MongoDB Atlas:**
    - Crea una cuenta en MongoDB Atlas y configura una base de datos.
    - Asegúrate de obtener la cadena de conexión para MongoDB Atlas.
6. **Configurar Redis:**
    - Configura la conexión a Redis en el archivo de configuración del proyecto.

Configuración inicial

Antes de ejecutar el proyecto, realiza las siguientes configuraciones iniciales:

1. **Agregar Archivos en Blob Storage:**
    - Sube los archivos CSV de Netflix e IMDb al Blob Storage en Azure Storage Account.
2. **Configurar Conexiones en Azure DataFactory:**
    - **Origen**: Configura Azure DataFactory para que el origen de datos sea los archivos CSV almacenados en Blob Storage.
    - **Destino**: Configura el destino de los datos procesados para que se almacenen en MongoDB Atlas.
3. **Ejecutar Pipelines en DataFactory:**
    - Crea y ejecuta los pipelines en Azure DataFactory para procesar y transformar los datos desde los archivos CSV a JSON y cargar los datos en MongoDB Atlas.
4. **Configurar el archivo appsettings.json:**
    - Asegurarse de que las configuraciones en el archivo appsettings.json estén alineadas con las variables de entorno definidas y las configuraciones de Azure Sql, Azure Rediscache y MongoDB Atlas.

Una vez completadas las configuraciones iniciales, puedes ejecutar el proyecto:

```
dotnet run
```

Uso del proyecto

Guía del usuario

Acceder a la Documentación de la API:

- Una vez que la aplicación esté en ejecución, puedes acceder a la documentación interactiva de la API generada por Swagger en la siguiente URL en el portal de Azure o en el ambiente local:
- http://localhost:2324/swagger
- http://<azurewebsitecontainer>/swagger aun no esta desplegado, cuando lo este se modifica este apartado.

Autenticación:

- Para interactuar con la API, primero debes autenticarte. Utiliza el endpoint de login para obtener un token JWT.
    - POST /api/auth/login
    - Request Body:

```
{
  "username": "your_username",
  "password": "your_password"
}
```

    - Response:

```
{
  "token": "your_jwt_token"
}
```

Acceder a los datos:

- Una vez autenticado, utiliza el token JWT para acceder a los datos mediante los endpoints de la API. Añade el token en el encabezado de autorización de tus solicitudes.
- Authorization: Bearer your_jwt_token

Ejemplos de uso

Obtener Todos los Títulos:

- Endpoint: Get/api/titles
- Ejemplo de solicitud: Get/api/titles con header de **Authorization**: Bearer your_jwt_token
- Ejemplo de respuesta:

```
[
  {
    "title": "Creed III",
    "type": "Movie",
    "netflix_data": {
      "show_id": "s1",
      "director": "Michael B. Jordan",
      "cast": "Michael B. Jordan, Tessa Thompson",
      "country": "United States",
      "date_added": "2023-02-03",
      "release_year": 2023,
      "rating": "PG-13",
      "duration": "90 min",
      "listed_in": "Drama, Action",
      "description": "After dominating the boxing world..."
    },
    "imdb_data": {
      "date_x": "2023-02-03",
      "score": 73,
      "genre": "Drama, Action",
      "overview": "After dominating the boxing world...",
      "crew": "Michael B. Jordan, Tessa Thompson",
      "orig_title": "Creed III",
      "status": "Released",
      "orig_lang": "English",
      "budget_x": 75000000,
      "revenue": 271616668,
      "country": "AU"
    }
  },
  ...
]
```


Obtener por búsqueda de género y puntuación

- Endpoint: GET /api/titles/search?genre={genre}&score={score}
- Ejemplo de solicitud: GET /api/titles/search?genre=Drama&score=75 con header de **Authorization**: Bearer your_jwt_token
- Ejemplo de respuesta:

```
[
  {
    "title": "Creed III",
    "type": "Movie",
    "netflix_data": {
      "show_id": "s1",
      "director": "Michael B. Jordan",
      "cast": "Michael B. Jordan, Tessa Thompson",
      "country": "United States",
      "date_added": "2023-02-03",
      "release_year": 2023,
      "rating": "PG-13",
      "duration": "90 min",
      "listed_in": "Drama, Action",
      "description": "After dominating the boxing world..."
    },
    "imdb_data": {
      "date_x": "2023-02-03",
      "score": 73,
      "genre": "Drama, Action",
      "overview": "After dominating the boxing world...",
      "crew": "Michael B. Jordan, Tessa Thompson",
      "orig_title": "Creed III",
      "status": "Released",
      "orig_lang": "English",
      "budget_x": 75000000,
      "revenue": 271616668,
      "country": "AU"
    }
  },
  ...
]
```

Pruebas y validación

Estrategia de pruebas

Estas son las siguientes pruebas para realizar:

1. **Pruebas Unitarias:**
    - Las pruebas unitarias se realizarán en componentes individuales del sistema para asegurarse de que cada uno funcione correctamente de manera aislada.
    - Se utilizará xUnit junto con Moq para crear y ejecutar estas pruebas en el entorno .NET 8.
    - Se cubrirán casos como validación de datos y lógica de negocio.
2. **Pruebas de Integración:**
    - Las pruebas de integración se centrarán en verificar que diferentes módulos del sistema interactúen correctamente entre sí.
    - Estas pruebas se realizarán para asegurar la correcta integración entre el API Client, MongoDB Atlas, Redis Cache y Azure SQL Database.
    - Se utilizarán bases de datos y servicios simulados para crear un entorno de pruebas controlado.
3. **Pruebas de Aceptación:**
    - Las pruebas de aceptación se llevarán a cabo para asegurar que el sistema cumpla con los requisitos y expectativas del usuario final.
    - Incluirán pruebas funcionales completas que simulen escenarios de uso real.
    - Se utilizará Postman para ejecutar y automatizar las pruebas de los endpoints de la API.

Casos de prueba

Estos son algunos de los ejemplos de casos de prueba que se utilizarán para asegurar la calidad del sistema:

1. **Pruebas Unitarias:**
    - **Validación de Datos:**
        - **Descripción:** Verificar que los datos recibidos en las solicitudes sean válidos.
        - **Caso de Prueba:** Enviar una solicitud con datos de entrada inválidos y verificar que se retorne un error adecuado.
    - **Lógica de Negocio:**

        - **Descripción:** Verificar la lógica de negocio implementada en el API Client.
        - **Caso de Prueba:** Ejecutar una operación de negocio específica y verificar que el resultado sea el esperado.

2. **Pruebas de Integración:**
    - **Interacción con MongoDB Atlas:**
        - **Descripción:** Verificar que el API Client interactúe correctamente con MongoDB Atlas.
        - **Caso de Prueba:** Crear, leer, actualizar y eliminar documentos en MongoDB Atlas y verificar que las operaciones se realicen correctamente.
    - **Interacción con Redis Cache:**
        - **Descripción:** Verificar que el almacenamiento en caché funcione correctamente.
        - **Caso de Prueba:** Realizar una consulta a la API, almacenar la respuesta en Redis y verificar que las consultas posteriores utilicen el caché.
    - **Autenticación con Azure SQL Database:**
        - **Descripción:** Verificar que la autenticación de usuarios funcione correctamente.
        - **Caso de Prueba:** Autenticar un usuario y verificar que se retorne un token JWT válido.
3. **Pruebas de Aceptación:**
    - **Consulta de Títulos:**
        - **Descripción:** Verificar que los usuarios puedan consultar títulos correctamente.
        - **Caso de Prueba:** Realizar una solicitud para obtener todos los títulos y verificar que se retorne la lista completa.
    - **Búsqueda de Títulos por Género:**
        - **Descripción:** Verificar que la funcionalidad de búsqueda funcione correctamente.
        - **Caso de Prueba:** Realizar una solicitud para buscar títulos por género y verificar que se retornen los resultados correctos.
    - **Acceso Autenticado:**

- **Descripción:** Verificar que solo usuarios autenticados puedan acceder a los datos protegidos.
- **Caso de Prueba:** Intentar acceder a un endpoint protegido sin autenticación y verificar que se retorne un error de acceso denegado.

Resultados esperados

Los resultados esperados de las pruebas son los siguientes:

1. **Pruebas Unitarias:**
    - Todos los casos de prueba deben pasar sin errores.
    - Las funciones individuales deben comportarse según lo esperado en todos los escenarios probados.
2. **Pruebas de Integración:**
    - La interacción entre los diferentes módulos debe ser correcta.
    - No debe haber errores de comunicación entre el API Client, MongoDB Atlas, Redis Cache y Azure SQL Database.
3. **Pruebas de Aceptación:**
    - El sistema debe cumplir con los requisitos y expectativas del usuario final.
    - Los endpoints de la API deben funcionar correctamente y retornar los resultados esperados.
    - Los mecanismos de autenticación y autorización deben ser efectivos.

Mantenimiento y soporte

Guía de mantenimiento

Para asegurar que el sistema se mantenga operativo y eficiente, se deben seguir procedimientos y mejores prácticas regulares, sugiero algunas directrices para el mantenimiento del sistema:

1. **Actualizaciones Regulares:**
    - **Dependencias y Librerías:** Mantener todas las dependencias y librerías del proyecto actualizadas para aprovechar las últimas mejoras y parches de seguridad.
    - **Frameworks y Servicios:** Asegurarse de que los frameworks (como .NET 8) y los servicios en la nube (como Azure y MongoDB Atlas) estén actualizados.
2. **Monitoreo y Alertas:**

- **Herramientas de Monitoreo:** Utilizar herramientas de monitoreo como Azure Monitor para supervisar el rendimiento del sistema y detectar posibles problemas.
- **Alertas:** Configurar alertas para recibir notificaciones sobre cualquier problema crítico que pueda afectar la disponibilidad o el rendimiento del sistema.

3. **Copia de Seguridad y Recuperación:**
    - **Copia de Seguridad Regular:** Realizar copias de seguridad periódicas de las bases de datos y del almacenamiento en Blob para prevenir la pérdida de datos.
    - **Planes de Recuperación:** Desarrollar y probar planes de recuperación ante desastres para asegurar una rápida recuperación en caso de fallos.
4. **Optimización del Rendimiento:**
    - **Revisión de Código:** Realizar revisiones de código regularmente para identificar y corregir posibles cuellos de botella en el rendimiento.
    - **Pruebas de Carga:** Realizar pruebas de carga para asegurarte de que el sistema puede manejar el tráfico esperado.

Soporte

Para obtener ayuda y soporte para el proyecto, sugiero utilizar las siguientes opciones:

1. **Documentación del Proyecto:**
    - Consulta la documentación del proyecto para obtener información detallada sobre la configuración, instalación y uso del sistema, en caso especifico me refiero al Readme del repositorio.
2. **Soporte en Línea:**
    - **Repositorio de GitHub:** Utilizar el repositorio de GitHub del proyecto para reportar problemas, solicitar nuevas características y contribuir al desarrollo del proyecto.
3. **Contacto Directo:**
    - **Equipo de Desarrollo:** Contactarme a través del correo electrónico institucional. diegofdiazh@javeriana.edu.co

Contribuciones

Guía de contribución

Para contribuir al proyecto, se deben seguir las siguientes directrices (aun no tengo el repositorio github, pero será agregado acá):

1. **Realizar Cambios:**
    - Realiza los cambios necesarios en la rama especifica diferente a la main.
2. **Enviar un Pull Request:**
    - Envía un pull request desde tu rama al repositorio principal rama main y describe los cambios realizados.

Políticas de código

- **Estilo de Código y patrón hexagonal:** Seguir el estilo de código definido en la documentación del proyecto.
- **Pruebas:** Asegurarse de que todos los cambios estén cubiertos por pruebas unitarias y de integración.
- **Documentación:** Actualizar la documentación según sea necesario para reflejar cualquier cambio realizado.

Licencia

Agradecimientos

Un agradecimiento especial a mi familia por su apoyo incondicional, a los profesores que me impartieron la materia de Tópicos Avanzados de Bases de Datos, y a la Pontificia Universidad Javeriana por proporcionarme la formación y los recursos necesarios para llevar a cabo este proyecto.