UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

Facultad de Ingenierias y Arquitectura
Carrera de Computación

Prácticum 1.1 – Proyecto Integrador
Octubre 2024 – Febrero 2025

Profesores:
Phd. Nelson Piedra, MsC. Audrey Romero, MsC. María Belén Mora,
Phd. Jorge López, Mgtr.Santiago Quiñonez

Mgtr. Karla Romero
Mgtr. Omar Ruiz

Integrantes (apellidos y nombres de los estudiantes):

- ALAO HUIRACOCHA BRYAN JOAQUIN
- VICENTE ABAD KEVIN CRISTOPHER
- CUEVA VALDIVIESO JUAN SEBASTIAN

Índice de contenidos

1 Introducción

El propósito del Proyecto Integrador en el componente Practicum 1.1 tiene como objetivo de desarrollar una base de datos a partir de un DataSet propuesto aplicando conocimientos de Fundamentos de Bases de Datos y Programación Funcional y Reactiva.
Es importante analizar los documentos compartidos en el aula virtual:

- Plan de estudio del componente.
- Anexo de planificación del Proyecto integrador.

Previo a este proyecto los estudiantes del tercer ciclo en la materia de Practicum 1.1 conformaron grupos de trabajo de un número 4 estudiantes. Los estudiantes deben cumplir con los siguiente prequisitos dentro del grupo:

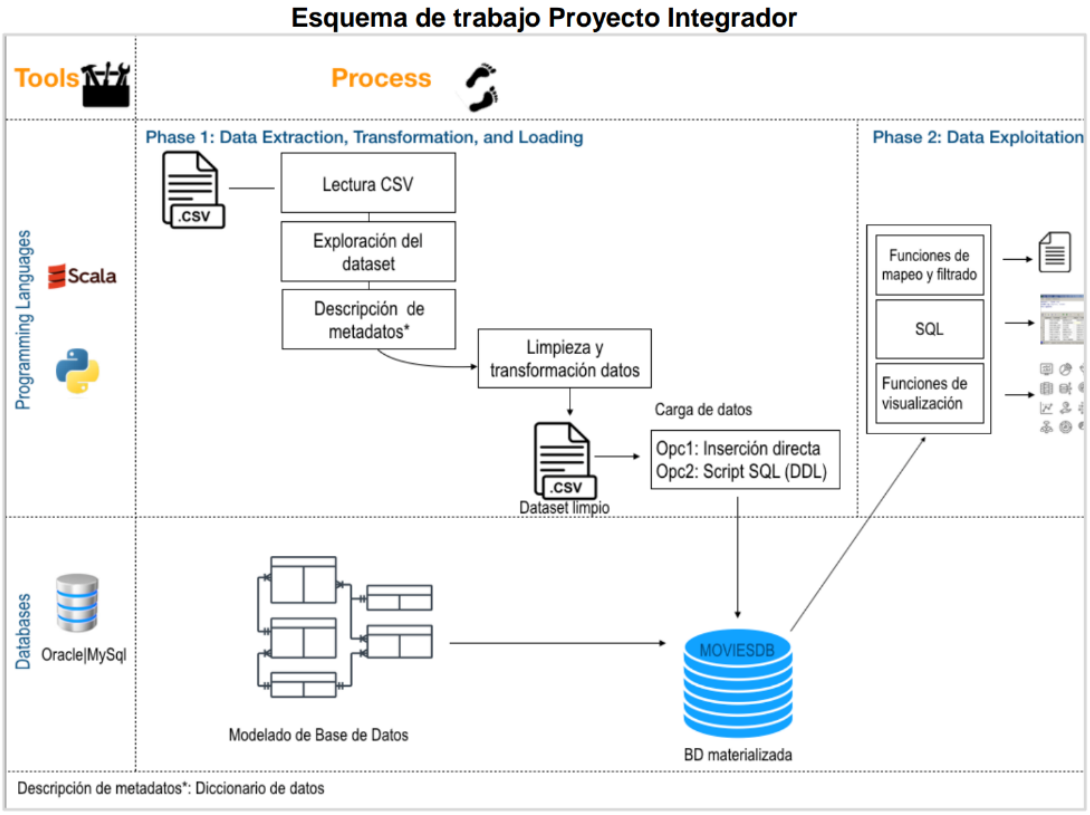
- Estar matriculados en la materia Base de Datos y Programación Funcional y Reactiva.
- Estar matriculados solo en la materia de Base de Datos o Programación Funcional y Reactiva.
- En casos espciales y bajo la responsabilidad propia del estudiante, no estar matriculado en ninguna de las dos materias, al ser así, dentro de un grupo solo puede haber un integrante con este perfil.

El Dataset a trabajar será entregado a los estudiantes siguiendo el cronograma de planificación.

2 Generalidades

Para el desarrollo de este proyecto integrador a partir de la semana 7 los estudiantes deberán realizar entregables respecto al Dataset especificado todas las actividades propuestas en el documento de Anexo de Planificación del Proyecto Integrador. Deberán en un solo entregable presentar lo que deben desarrollar una vez obtenido el DataSet, semana a semana a partir de la semana 7 de estudio.
El esquema general del proyecto se lo puede apreciar en la figura 1.1

Figura 1.1



3 Marco Teórico

3.1 Introducción

El análisis y desarrollo de una base de datos y el uso de la programación son primordiales dentro de los sistemas modernos referentes a la gestión de datos. Este marco teórico abarca los conceptos básicos para el manejo de bases de datos y programación funcional y reactiva, ambos son cruciales para cumplir con el objetivo de realizar un análisis descriptivo y exploratorio de un dataset, proporcionado, referente a películas.

3.2 Fundamentos de bases de datos

Una base de datos es el cumulo u conjunto de datos organizados almacenados dentro de un equipo computacional. La creación de estas nos ayuda al acceso, gestión y manipulación de los datos de una manera mas eficiente y ordenada, de la misma forma garantizamos la integridad. Nos apoyamos de un sistema de gestión de bases de datos (DBMS) para la interacción y modificación de los datos. Existen una variedad de DBMS que podemos hacer uso, tales como MySQL, PostgreSQL, Oracle, etc. Estos trabajan mediante consultas, también llamadas “querrys”, con el fin de hacer contacto u administrar los datos almacenos.

Tenemos diferentes modelos con los cuales podemos representar una base de datos, algunos de una manera más gráfica, algunos que veremos dentro del proyecto integrador son:

- Modelo conceptual: representación de un sistema basado en conceptos específicos, ayudando a los demás a comprender mediante entidades importantes y como estas se relacionan entre sí.
- Modelo relacional: haciendo uso de tablas, compuestas por registros y atributos, y relaciones entre ellas nos da a comprender el flujo de datos entre tablas.
- Modelo lógico: un modelo mas preciso que el modelo conceptual, igualmente, basado en tablas “contiene representaciones de entidades y atributos, relaciones, identificadores exclusivos, subtipos y supertipos y restricciones entre relaciones” (IBM).
- Modelo físico: modelo que representa las tablas, las cuales van a tener las claves primarias, foráneas y demás atributos, y sus relaciones. “Un modelo de datos físicos se puede utilizar para generar sentencias DDL que, después, se pueden desplegar en un servidor de base de datos” (IBM).

3.3 Lenguaje SQL

El lenguaje SQL (Structured Query Language) es el lenguaje utilizado por los DNMS, siendo el estándar para la comunicación con las bases de datos. Permite la realización de consultas, actualizaciones, inserciones y creación de esquemas (tablas u bases de datos).

Comandos:

- SELECT: extraer información.
- INSERT: modificación de los datos.
- UPDATE: modificación de los datos.
- CREATE: definición de esquemas.

3.4 Normalización

4 Desarrollo del Proyecto Integrador

Dominio - Base de Datos.

4.1 Entregable 1

4.1.1 Actividades

- Estudio y comprensión de los datos proporcionados.
- Obtención de datos primarios sobre el dataset de trabajo.

4.1.2 Análisis del Dataset.

4.1.3 Descripción general del dataset.

El listado de datos dentro del dataset, que lleva de nombre “pi_movies_complete”, podemos encontrar un cumulo de películas con su nombre, actores, fecha de lanzamiento, etc. Este nos ayuda a mantener una organización y un almacenamiento de las películas estrenadas y/o proyectadas, ya sea en un cine o página de reproducción de películas.

4.1.4 Análisis descriptivo

Nombre del Dataset	pi_movies_small
Objetivo del Dataset	Clasificar las películas según algunos puntos
Cuántas películas están incluidas	100
Cuáles son los géneros más comunes y menos presentados	El más común es Acción y comedia El menos común es TvMovie
Cuántas películas se producen por año	2 películas por año
Se define si hay más de un director de películas incluido	Si
Cuáles otros datos o metadatos se	Idiomas, donde fue el país donde se produjo, compañía y popularidad

incluyen en el Dataset	
Qué información podría analizar con este Dataset (al menos 3 ideas)	Cuántas películas hay en cada idioma Español INGLES
Con qué datos de películas que encuentran en sitios como - Netflix, IMBD, TMBD- se relaciona el dataset	ACTORES, título, idioma, producción

4.1.5 Descripción de Datos del data set

COLUMNA	TIPO DE DATO	DESCRIPCION
adult	String	Pelicula para adultos
belongs_to	JSON	Si la pelicula pertenece a una colecccion
budget	double	Presupuesto para la pelicula
genres	JSON	Lista de generos de la pelicula
homopage	string	URL de la pagina web oficial de la pelicula
id	int	Identificador unico de la pelicula
imdb_id	string	Identificador unico de la pelicula en la base de datos IBM
original_lenguage	string	Idioma original de la pelicula
original_title	string	Titulo original de la pelicula
overview	string	Sinopsis de la pelicula
popularity	double	Popularidad de la pelicula
poster_path	string	Imagen de la pelicula (URL)
production_companies	JSON	Compañias asociadas a la pelicula
production-countries	JSON	Paises donde se estreno la pelicula
release_date	fecha	Fecha de lanzamiento
revenue	double	Ingresos ganados
runtime	double	Duracion de la pelicula
spoken_languages	JSON	Lista de idiomas de la pelicula
status	string	Estado pelicula
tagline	string	Lema de la pelicula
title	string	Titulo de la pelicula
video	boolean	Video asociado a la pelicula
vote_average	double	Votos promedio

vote_count	int	Numero de votos recibidos
keywords	JSON	Palabras claves
cast	JSON	Actores participantes
crew	JSON	Empleados del equipo tecnico
ratings	double	Calificacion de la pelicula

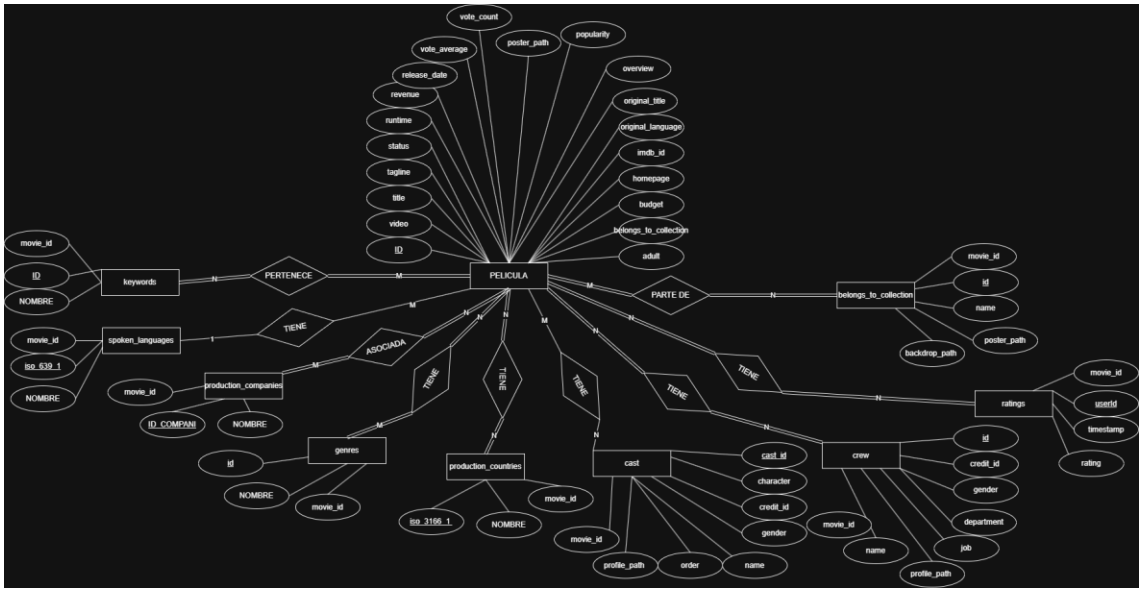
4.2 Entregable 2.

4.2.1 Modelo Conceptual

Estidades claves	genres
	production_companies
	production_countries
	spoken_languages
	keywords
	cast
	crew
	rating
	PELICULA
	Belongs_to_collection
ATRIBUTOS EN LAS ENTIDADES	
PELICULA	adults
	budget
	homepage
	id
	imdb_id
	original_language
	original_title
	overview
	popularity
	poster_path
	release_date
	revenue
	runtime
	status
	tagline
	title
	video
	vote_average
	vote_count
KEYWORDS	movie_id
	id
	name

SPOKEN_LANGUAGES	movie_id
	iso_639_1
	name
PRODUCTION_COMPANIES	movie_id
	name
	id
GENRES	movie_id
	id
	name
PRODUCTION_COUNTRIES	movie_id
	iso_3166_1
	name
CAST	cast_id
	character
	movie_id
	credit_id
	gender
	name
	order
	profile_path
CREW	credit_id
	department
	movie_id
	gender
	id
	job
	name
	profile_path
RATINGS	movie_id
	userId
	timestamp
	rating
BELONGS_TO_COLLECTION	movie_id
	id
	name
	poster_path
	backdrop_path

4.2.2 Esquema relacional



https://drive.google.com/file/d/1u7yxDhC9MUKOCClceWNRnN9yBtFnzYdd/view?usp=sharing

4.2.3 Diccionario de datos

Nombre de la columna	Tipo de dato	Descripción	Restricciones	Obligatoriedad
adult	String	Indica si la película es para adultos.		No
belongs_to	JSON	Indica si la película pertenece a una colección.		No
budget	double	Presupuesto asignado para la película.		No
genres	JSON	Lista de géneros de la película.		No
homepage	string	URL de la página web oficial de la película.		No
id	int	Identificador único de la película.	Clave primaria	Si
imdb_id	string	Identificador único en la base de datos IMDB.	Único	No
original_language	string	Idioma original de la película.		Si

original_title	string	Título original de la película.		Si
overview	string	Sinopsis de la película.		No
popularity	double	Popularidad de la película.		No
poster_path	string	URL de la imagen del póster de la película.		No
production_companies	JSON	Lista de compañías asociadas con la película.		No
production_countries	JSON	Lista de países donde se estrenó la película.		No
release_date	fecha	Fecha de lanzamiento de la película.		Si
revenue	double	Ingresos generados por la película.		No
runtime	double	Duración de la película en minutos.		No
spoken_languages	JSON	Lista de idiomas hablados en la película.		No
status	string	Estado de la película (e.g., lanzada, planeada).		Si
tagline	string	Lema promocional de la película.		No
title	string	Título de la película.		Si
video	boolean	Indica si hay un video asociado a la película.		No
vote_average	double	Promedio de calificaciones de los usuarios.		No

vote_count	int	Número total de votos recibidos.		No
keywords	JSON	Palabras clave asociadas con la película.		No
cast	JSON	Lista de actores que participaron en la película.		No
crew	JSON	Lista de empleados técnicos de la película.		No
ratings	double	Calificación global de la película.		No

4.2.4 Ejemplos de relaciones o muestras

Ejemplo 1:

Película: "El Asedio Final"

Atributos de la Película

- **Título Original**: El Asedio Final
- **Idioma Original**: Español
- **Fecha de Lanzamiento**: 2020-11-15
- **Géneros**: Acción, Drama
- **Presupuesto**: $2,500,000
- **Ingresos**: $10,000,000
- **Popularidad**: 85.7
- **Duración**: 130 minutos
- **Sinopsis**: Un grupo de soldados lucha para defender su territorio durante una invasión inesperada.
- **Estado**: Lanzada
- **Palabras Clave**: guerra, soldados, invasión

Relaciones de la Película

1. **Relación con Actores**:
    - Actores:
        - Juan Pérez (Protagonista)
        - Ana López (Secundaria)
        - Carlos Ruiz (Villano)
2. **Relación con Compañías de Producción**:
    - Compañías:
        - CineMundo
        - Latam Films
3. **Relación con Géneros**:
    - Géneros:
        - Acción
        - Drama

Ejemplo 2:
Película: "Ant-Man"
Atributos de la Película
- Título Original:Ant-Man
- Idioma Original: Inglés
- Fecha de Lanzamiento: 2015-07-14
- Géneros: Ciencia Ficción, Acción, Aventura
- Presupuesto: $130,000,000
- Ingresos: $519,311,965
- Popularidad: 102,899
- Duración: 117 minutos
- Sinopsis: Armado con la asombrosa capacidad de reducirse en tamaño pero aumentar en fuerza, el maestro ladrón Scott Lang debe abrazar su héroe interior y ayudar a su mentor, el Doctor Hank Pym, a proteger el secreto detrás de su espectacular traje de Ant-Man de una nueva generación de amenazas gigantescas. Contra obstáculos aparentemente insuperables, Pym y Lang deben planear y llevar a cabo un atraco que salvará al mundo.
- Estado: Lanzada
- Palabras Clave: superhéroe, atraco, tecnología avanzada
Relaciones de la Película
1. Relación con Actores:
  - Actores:
    - Paul Rudd (Scott Lang / Ant-Man)
    - Michael Douglas (Hank Pym)
    - Evangeline Lilly (Hope van Dyne)
2. Relación con Compañías de Producción:
  - Compañías:
    - Marvel Studios
3. Relación con Géneros:
  - Géneros:
    - Ciencia Ficción
    - Acción
    - Aventura

Ejemplo 3:
Película: "Dumb and Dumber To"
Atributos de la Película
- Título Original: Dumb and Dumber To
- Idioma Original: en
- Fecha de Lanzamiento: 12/11/2014
- Géneros: 'Comedy
- Presupuesto: $40000000
- Ingresos: $169837010
- Popularidad: 15402597
- Duración: 110 minutos
- Sinopsis: 20 years after the dimwits set out on their first adventure, they head out in search of one of their long lost children in the hope of gaining a new kidney.
- Estado: Released
- Palabras Clave: friendship – sequel - road movie - buddy comedy
Relaciones de la Película

1. Relación con Actores:
   - Actores:
     - Jim Carrey (Lloyd Christmas)
     - Jeff Daniels (Harry Dunne)
     - Rachel Melvin (Penny)
2. Relación con Compañías de Producción:
   - Compañías:
     - New Line Cinema
     - Universal Pictures
3. Relación con Géneros:
   - Géneros:
     - Comedy

4.3 Entregable 3.

4.3.1 Modelado Lógico (figura del modelo lógico final)

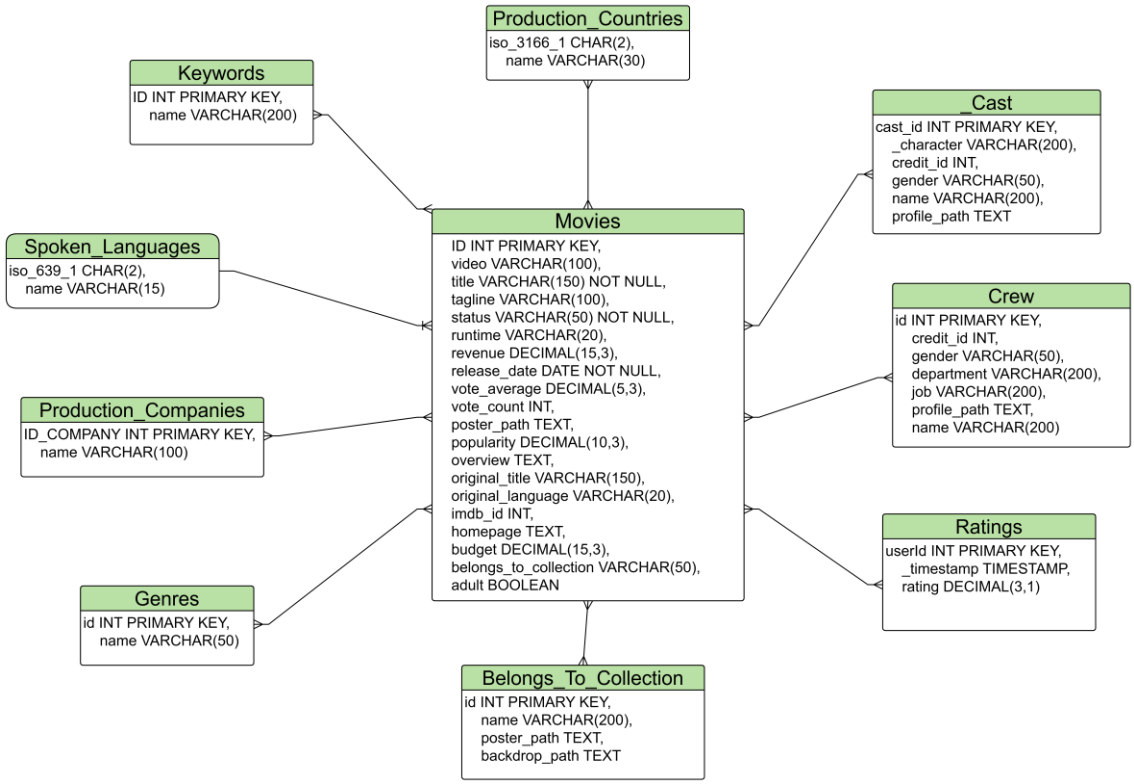
4.3.2 Proceso del modelado lógico

El proceso de transformación del modelo conceptual al modelo lógico consiste en convertir los elementos abstractos del modelo conceptual en estructuras más formales y detalladas que pueden implementarse en un sistema de bases de datos relacionales.

1. **Identificación de entidades y relaciones**: Se analizan los elementos del modelo conceptual (entidades, atributos y relaciones) y se definen como tablas en el modelo lógico.
2. **Definición de claves primarias y foráneas**: Se establecen los identificadores únicos (claves primarias) y las referencias entre tablas (claves foráneas) para mantener la integridad referencial.
3. **Especificación de tipos de datos**: Se asignan tipos de datos adecuados a cada atributo, considerando restricciones y optimización del almacenamiento.
4. **Normalización**: Se eliminan redundancias y dependencias innecesarias para optimizar la estructura de la base de datos.

Según Connolly y Begg (2015), la transformación del modelo conceptual al modelo lógico permite organizar la información en una estructura eficiente y coherente, facilitando la implementación en un SGBD.

Referencia:
Connolly, T., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Pearson.



4.3.3 Diccionario de datos

Movies				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
ID	INT	Número	Identificador único de la película	PRIMARY KEY
video	VARCHAR(100)	Texto	Información sobre el video	NULL permitido
title	VARCHAR(150)	Texto	Título de la película	NOT NULL
tagline	VARCHAR(100)	Texto	Lema o eslogan de la película	NULL permitido
status	VARCHAR(50)	Texto	Estado de la película (ej. "Released")	NOT NULL
runtime	VARCHAR(20)	Texto	Duración de la película en minutos	NULL permitido
revenue	DECIMAL(15,3)	Número	Recaudación en taquilla	NULL permitido
release_date	DATE	Fecha	Fecha de estreno	NOT NULL
vote_average	DECIMAL(5,3)	Número	Promedio de votos	NULL permitido
vote_count	INT	Número	Número de votos	NULL permitido
poster_path	TEXT	Texto	URL del póster de la película	NULL permitido
popularity	DECIMAL(10,3)	Número	Índice de popularidad	NULL permitido
overview	TEXT	Texto	Resumen de la película	NULL permitido
original_title	VARCHAR(150)	Texto	Título original de la película	NULL permitido
original_language	VARCHAR(20)	Texto	Idioma original de la película	NULL permitido
imdb_id	INT	Número	Identificador en IMDb	NULL permitido

homepage	TEXT	Texto	Página web oficial	NULL permitido
budget	DECIMAL(15,3)	Número	Presupuesto de la película	NULL permitido
belongs_to_collection	VARCHAR(50)	Texto	Colección a la que pertenece la película	NULL permitido
adult	VARCHAR(10)	Texto	Indica si la película es para adultos	NULL permitido

Keywords				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
ID	INT	Número	Identificador único de la palabra clave	PRIMARY KEY
name	VARCHAR(200)	Texto	Nombre de la palabra clave	NULL permitido

Spoken_Languages				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
iso_639_1	CHAR(2)	Código	Código ISO del idioma	PRIMARY KEY
name	VARCHAR(15)	Texto	Nombre del idioma	NULL permitido

Production_Companies				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
ID_COMPANY	INT	Número	Identificador único de la compañía	PRIMARY KEY
name	VARCHAR(100)	Texto	Nombre de la compañía	NULL permitido

Genres				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
id	INT	Número	Identificador único del género	PRIMARY KEY
name	VARCHAR(50)	Texto	Nombre del género	NULL permitido

Production_Countries				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
iso_3166_1	CHAR(2)	Código	Código ISO del país	PRIMARY KEY
name	VARCHAR(30)	Texto	Nombre del país	NULL permitido

_Cast				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
cast_id	INT	Número	Identificador único del actor en la película	PRIMARY KEY
_character	VARCHAR(200)	Texto	Nombre del personaje interpretado	NULL permitido
credit_id	INT	Número	ID del crédito	NULL permitido
gender	VARCHAR(50)	Texto	Género del actor	NULL permitido
name	VARCHAR(200)	Texto	Nombre del actor	NULL permitido
profile_path	TEXT	Texto	URL de la foto de perfil	NULL permitido

Crew				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
id	INT	Númerico	Identificador único del miembro del equipo	PRIMARY KEY
credit_id	INT	Númerico	ID del crédito	NULL permitido
gender	VARCHAR(50)	Texto	Género del miembro del equipo	NULL permitido
department	VARCHAR(200)	Texto	Departamento en el que trabaja	NULL permitido
job	VARCHAR(200)	Texto	Puesto dentro de la producción	NULL permitido
profile_path	TEXT	Texto	URL de la foto de perfil	NULL permitido
name	VARCHAR(200)	Texto	Nombre del miembro del equipo	NULL permitido

Ratings				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
userId	INT	Númerico	Identificador único del usuario	PRIMARY KEY
_timestamp	TIMESTAMP	Fecha y hora	Fecha y hora del rating	DEFAULT CURRENT_TIMESTAMP
rating	DECIMAL(3,1)	Númerico	Puntuación otorgada a la película	NULL permitido

Belongs_To_Collection				
Atributo	**Tipo de Dato**	**Dominio**	**Descripción**	**Restricciones**
id	INT	Númerico	Identificador único de la colección	PRIMARY KEY
name	VARCHAR(200)	Texto	Nombre de la colección	NULL permitido
poster_path	TEXT	Texto	URL de la imagen de la colección	NULL permitido
backdrop_path	TEXT	Texto	URL del fondo de la colección	NULL permitido

4.4 Entregable 4.

4.4.1 Modelado Físico

EXPLICACION DEL PROCESO DE CREACION DE LA BASE DE DATOS

- **Movies**: Contiene los datos principales de cada película, como título, fecha de estreno, presupuesto, recaudación, idioma original, votaciones y popularidad.
- **Keywords**: Almacena palabras clave asociadas a las películas, útiles para clasificaciones y búsquedas.
- **Spoken_Languages**: Guarda los idiomas hablados en las películas con su código ISO.
- **Production_Companies**: Registra las compañías productoras de las películas.
- **Genres**: Define los distintos géneros cinematográficos, como acción, drama, comedia, etc.

- **Production_Countries**: Lista los países donde se ha producido cada película.
- **_Cast**: Contiene información sobre los actores que participan en las películas, incluyendo el personaje que interpretan.
- **Crew**: Registra al equipo técnico de las películas, con detalles como departamento, cargo y género.
- **Ratings**: Almacena las calificaciones dadas por los usuarios a las películas.
- **Belongs_To_Collection**: Agrupa películas dentro de una misma colección, como sagas o franquicias.

```sql
CREATE DATABASE modelo_fisico;
USE modelo_fisico;

CREATE TABLE Movies(
    ID INT PRIMARY KEY,
    video VARCHAR(100),
    title VARCHAR(150) NOT NULL,
    tagline VARCHAR(100),
    status VARCHAR(50) NOT NULL,
    runtime VARCHAR(20),
    revenue DECIMAL(15,3),
    release_date DATE NOT NULL,
    vote_average DECIMAL(5,3),
    vote_count INT,
    poster_path TEXT,
    popularity DECIMAL(10,3),
    overview TEXT,
    original_title VARCHAR(150),
    original_language VARCHAR(20),
    imdb_id INT,
    homepage TEXT,
    budget DECIMAL(15,3),
    adult BOOLEAN
);
```

```sql
CREATE TABLE Keywords(
    ID INT PRIMARY KEY,
    id_movie INT,
    name VARCHAR(200)
);

CREATE TABLE Spoken_Languages(
    iso_639_1 CHAR(2),
    id_movie INT,
    name VARCHAR(15)
);

CREATE TABLE Production_Companies(
    ID_COMPANY INT PRIMARY KEY,
    id_movie INT,
    name VARCHAR(100)
);

CREATE TABLE Genres(
    id INT PRIMARY KEY,
    id_movie INT,
    name VARCHAR(50)
);


CREATE TABLE Production_Countries(
    iso_3166_1 CHAR(2),
    id_movie INT,
    name VARCHAR(30)
);

CREATE TABLE _Cast(
    cast_id INT PRIMARY KEY,
    id_movie INT,
    _character VARCHAR(200),
    credit_id INT,
    gender VARCHAR(50),
    name VARCHAR(200),
    profile_path TEXT
);
```

```sql
CREATE TABLE Crew(
    id INT PRIMARY KEY,
    id_movie INT,
    credit_id INT,
    gender VARCHAR(50),
    department VARCHAR(200),
    job VARCHAR(200),
    profile_path TEXT,
    name VARCHAR(200)
);

CREATE TABLE Ratings(
    userId INT PRIMARY KEY,
    id_movie INT,
    _timestamp TIMESTAMP, -- Formato 'YYYY-MM-DD HH:MM:SS'
    rating DECIMAL(3,1)
);

CREATE TABLE Belongs_To_Collection(
    id INT PRIMARY KEY,
    id_movie INT,
    name VARCHAR(200),
    poster_path TEXT,
    backdrop_path TEXT
);
```

4.4.2 Inserción de datos

La inserción de datos es el proceso mediante el cual se añaden nuevos registros a una base de datos. En el contexto de una base de datos relacional, este proceso implica agregar información a las tablas correspondientes, manteniendo la integridad y las relaciones entre los datos.

Explicación breve del proceso de inserción de datos:

- Identificación de las tablas: Primero, se deben identificar todas las tablas que contienen los diferentes tipos de información. Por ejemplo, en una película, puede haber tablas para la película en sí (título, presupuesto, etc.), los actores, los géneros, las compañías de producción, entre otras.
- Definición de los datos a insertar: Luego, se deben definir los valores específicos que se van a insertar en cada tabla. Por ejemplo, el título de la película, la fecha de lanzamiento, el género, los actores, y otros detalles relevantes.
- Uso de la sentencia SQL INSERT: Para cada tabla, se utiliza la sentencia INSERT INTO seguida del nombre de la tabla y los valores que se desean agregar. Este comando debe especificar las columnas y los valores correspondientes a cada una de ellas.
- Relaciones entre tablas: Las bases de datos relacionales a menudo contienen relaciones entre tablas. Por ejemplo, una película puede estar asociada con varios géneros o actores, y es necesario asegurarse de insertar estos datos en las tablas correspondientes de manera coherente para que las relaciones entre las tablas sean correctas.

- Verificación de la integridad referencial: Al insertar datos en tablas relacionadas, es importante verificar que los valores de clave primaria y clave externa sean correctos, es decir, que los registros relacionados existan en las tablas correspondientes para evitar violaciones de integridad referencial.

Ejemplo de inserción:

Si estás insertando una película, podrías comenzar con la tabla Movies para ingresar los detalles básicos de la película, y luego insertar los actores, géneros, y otros datos relacionados en sus respectivas tablas.

4.4.5 Consultas

Descripción de las Subconsultas:

```
-- SUBCONSULTAS
-- Consulta 1: Películas con mayor recaudación por género
SELECT g.name AS Genre, m.title, m.revenue
FROM Movies m
JOIN Genres g ON m.ID = g.id_movie
WHERE m.revenue = (SELECT MAX(revenue) FROM Movies WHERE ID = g.id_movie);

-- Consulta 2: Película con mayor calificación promedio
SELECT m.title, m.vote_average
FROM Movies m
WHERE m.vote_average = (SELECT MAX(vote_average) FROM Movies);

-- Consulta 3: Películas con un presupuesto mayor al promedio
SELECT m.title, m.budget
FROM Movies m
WHERE m.budget > (SELECT AVG(budget) FROM Movies);
```

Consulta 1:

- Esta consulta obtiene el título y la recaudación de las películas con mayor recaudación dentro de cada género.
- JOIN se usa para asociar la tabla de películas (Movies) con la de géneros (Genres).
- La subconsulta (SELECT MAX(revenue) FROM Movies WHERE ID = g.id_movie) obtiene la película con la mayor recaudación dentro de cada género.

Consulta 2:

- Esta consulta selecciona la película con la calificación promedio más alta.
- Subconsulta: (SELECT MAX(vote_average) FROM Movies) busca la calificación promedio máxima de todas las películas.

Consulta 3:

- Aquí se listan las películas cuyo presupuesto es superior al presupuesto promedio de todas las películas.
- Subconsulta: (SELECT AVG(budget) FROM Movies) calcula el presupuesto promedio.

Descripción de consultas GROUP BY:

```sql
-- Consulta 4: Número de películas por compañía productora
SELECT p.name AS Company, COUNT(m.ID) AS MovieCount
FROM Movies m
JOIN Production_Companies p ON m.ID = p.id_movie
GROUP BY p.name;

-- Consulta 5: Cantidad de películas por idioma
SELECT s.name AS Language, COUNT(m.ID) AS MovieCount
FROM Movies m
JOIN Spoken_Languages s ON m.ID = s.id_movie
GROUP BY s.name;

-- Consulta 6: Actores y su cantidad de películas
SELECT c.name, COUNT(cast_id) AS MovieCount
FROM _Cast c
GROUP BY c.name;

-- Consulta 7: Películas con un elenco de más de 5 actores
SELECT m.title, COUNT(c.cast_id) AS ActorCount
FROM Movies m
JOIN _Cast c ON m.ID = c.id_movie
GROUP BY m.title
HAVING COUNT(c.cast_id) > 5;

-- Consulta 8: Películas con más de un género
SELECT m.title, COUNT(g.id) AS GenreCount
FROM Movies m
JOIN Genres g ON m.ID = g.id_movie
GROUP BY m.title
HAVING COUNT(g.id) > 1;

-- Consulta 9: Películas con más de un país de producción
SELECT m.title, COUNT(p.iso_3166_1) AS CountryCount
FROM Movies m
JOIN Production_Countries p ON m.ID = p.id_movie
GROUP BY m.title
HAVING COUNT(p.iso_3166_1) > 1;
```

Consulta 4:

- Se obtiene la cantidad de películas producidas por cada compañía.
- Se utiliza GROUP BY para agrupar los resultados por compañía productora (p.name) y COUNT para contar el número de películas en cada grupo.

Consulta 5:

- Esta consulta muestra cuántas películas existen en cada idioma.
- GROUP BY agrupa por el idioma (s.name) y COUNT cuenta las películas por idioma.

Consulta 6:

- Se obtiene la cantidad de películas en las que ha participado cada actor.
- GROUP BY agrupa por nombre de actor (c.name) y COUNT cuenta las películas de cada actor.

Consulta 7:

- Aquí se listan las películas que tienen más de 5 actores en su elenco.
- GROUP BY agrupa las películas por título (m.title) y HAVING COUNT(c.cast_id) > 5 filtra aquellas con más de 5 actores.

Consulta 8:

- Esta consulta muestra las películas que tienen más de un género asociado.
- GROUP BY agrupa por título de película y HAVING COUNT(g.id) > 1 filtra aquellas películas con más de un género.

Consulta 9:

- Similar a la anterior, esta consulta selecciona las películas que fueron producidas en más de un país.
- GROUP BY agrupa por el título de la película y HAVING COUNT(p.iso_3166_1) > 1 filtra aquellas que tienen más de un país de producción.

Descripción de Consulta Multitabla

```
-- CONSULTA MULTITABLA(Combinar al menos 3 tablas)
SELECT m.title AS Movie, g.name AS Genre, c.name AS Actor
FROM Movies m
JOIN Genres g ON m.ID = g.id_movie
JOIN _Cast c ON m.ID = c.id_movie
WHERE c.cast_id = (SELECT MIN(cast_id) FROM _Cast WHERE id_movie = m.ID);
```

- Esta consulta muestra las películas junto con su género y el primer actor en el elenco (basado en el cast_id más bajo).
- JOIN se utiliza para combinar tres tablas: Movies, Genres y _Cast.
- Subconsulta: (SELECT MIN(cast_id) FROM _Cast WHERE id_movie = m.ID) selecciona al actor con el menor cast_id para cada película.

4.4.6 Inserción de datos

```
INSERT INTO Movies (ID, video, title, tagline, status, runtime, revenue, release_date, vote_average, vote_count, poster_path, popularity, over
VALUES (1, NULL, 'Ant-Man', NULL, 'Released', '117 min', 519311965.000, '2015-07-14', NULL, NULL, NULL, 102899.000,
        'Armado con la asombrosa capacidad de reducirse en tamaño pero aumentar en fuerza,
        el maestro ladrón Scott Lang debe abrazar su héroe interior y ayudar a su mentor,
        el Doctor Hank Pym, a proteger el secreto detrás de su espectacular traje de Ant-Man de una nueva generación de amenazas gigantescas.
        Contra obstáculos aparentemente insuperables, Pym y Lang deben planear y llevar a cabo un atraco que salvará al mundo.',
        'Ant-Man', 'en', 285, NULL, 130000000.000, FALSE);

INSERT INTO Keywords (ID, id_movie, name)
VALUES (1, 1, 'superhéroe'),
       (2, 1, 'atraco'),
       (3, 1, 'tecnología avanzada');

INSERT INTO Spoken_Languages (iso_639_1, id_movie, name)
VALUES ('en', 1, 'Inglés');

INSERT INTO Production_Companies (ID_COMPANY, id_movie, name)
VALUES (1, 1, 'Marvel Studios');

INSERT INTO Genres (id, id_movie, name)
VALUES (1, 1, 'Ciencia Ficción'),
       (2, 1, 'Acción'),
       (3, 1, 'Aventura');
```

```sql
INSERT INTO Production_Countries (iso_3166_1, id_movie, name)
VALUES ('US', 1, 'United States');

INSERT INTO _Cast (cast_id, id_movie, _character, credit_id, gender, name, profile_path)
VALUES (1, 1, 'Scott Lang / Ant-Man', 1, 'Male', 'Paul Rudd', NULL),
       (2, 1, 'Hank Pym', 2, 'Male', 'Michael Douglas', NULL),
       (3, 1, 'Hope van Dyne', 3, 'Female', 'Evangeline Lilly', NULL);

INSERT INTO Crew (id, id_movie, credit_id, gender, department, job, profile_path, name)
VALUES (1, 1, 1, 'Male', 'Directing', 'Director', NULL, 'Peyton Reed'),
       (2, 1, 2, 'Male', 'Writing', 'Screenplay', NULL, 'Edgar Wright'),
       (3, 1, 3, 'Male', 'Production', 'Producer', NULL, 'Kevin Feige');

INSERT INTO Ratings (userId, id_movie, _timestamp, rating)
VALUES (1, 1, '2025-02-03 12:00:00', 8.0);

INSERT INTO Belongs_To_Collection (id, id_movie, name, poster_path, backdrop_path)
VALUES (1, 1, 'Marvel Cinematic Universe', NULL, NULL);
```

Dominio – Programación Funcional y Reactiva.

4.5 Entregable 1.

4.5.1 Análisis Exploratorio de Datos

Descripción del diccionario de datos sobre todo enfocados a los objetos JSON encontrados.

COLUMNA	TIPO DE DATO	DESCRIPCION	Observación (descripción de algo particular, por ejemplo, si hay errores en las estructuras JSON, etc.)
belongs_to_collection	JSON	Contiene los detalles de la colección a la que pertenece una película. El objeto incluye los campos: ***id (identificador único de la colección), name (nombre de la colección), poster_path (ruta de la imagen del cartel) y backdrop_path (ruta de la imagen de fondo).***	Si la película no pertenece a una colección, este campo será null. El poster_path y backdrop_path son rutas relativas que deben combinarse con una URL base para mostrar imágenes.
genres	JSON	Representa un género y contiene las siguientes claves: id: Un identificador numérico único para el género. name: El nombre del género en formato de texto.	Algunos generos estan dentro de apostrofes ('...') en vez de doble comilla ("...."). Si la pelicula no tiene genero se determinara una lista vacia.
production_companies	JSON	Representa una compañía de producción y contiene las siguientes claves: name: El nombre de la compañía de producción en formato de texto. id: Un identificador	Algunos generos estan dentro de apostrofes ('...') en vez de doble comilla ("...."). Si la pelicula no tiene genero se determinara una lista vacia.

		numérico único para la compañía.	
production_countries	JSON	Representa a un país relacionado con la producción de una pelicula y tiene las siguientes claves: iso_3166_1: Código del país name: El nombre completo del país en texto.	Algunos "production_countries" estan dentro de apostrofes ('...') en vez de doble comilla ("...."). iso_3166_1 sera representado en dos letras Si la pelicula no tiene genero se determinara una lista vacia.
spoken_languages	JSON	Representa un idioma hablado en la película y tiene las siguientes claves: iso_639_1: Código del idioma representado por dos letras name: El nombre completo del idioma en texto.	Algunos "spoken_languages" estan dentro de apostrofes ('...') en vez de doble comilla ("...."). En iso_639_1 sera representado en dos letras Si la pelicula no tiene genero se determinara una lista vacia.

keywords	JSON	representa una palabra clave o tema relacionado con la película y tiene las siguientes claves: id: Un identificador numérico único. name: El nombre o descripción de la palabra en texto.	Algunos "keywords" estan dentro de apostrofes ('...') en vez de doble comilla ("...."). Si la pelicula no tiene genero se determinara una lista vacia.
cast	JSON	Representa a un miembro del casting de una película, detalles sobre su papel y información personal y tiene las siguientes claves: cast_id: Un identificador único para cada miembro del elenco dentro de la producción. character: El nombre del personaje que interpreta credit_id: Un identificador único de crédito. gender: Representa el género del actor. id: Identificador único del actor. name: El nombre del actor. order: Representa el orden de aparición. profile_path: La ruta de la imagen de perfil del actor.	Algunos "cast" estan dentro de apostrofes ('...') en vez de doble comilla ("...."). El profile_path es una ruta relativa que debe combinarse con una URL base para mostrar imágenes. Si la pelicula no tiene genero se determinara una lista vacia.

crew	JSON	Representa un miembro del equipo y tiene las siguientes claves:. credit_id: Identificador único del crédito del miembro. department: Departamento al que pertenece el miembro. gender: Género del miembro. id: Identificador único del miembro. job: El trabajo específico realizado por la persona. profile_path: Ruta de la imagen de perfil del miembro.	Algunos "crew" estan dentro de apostrofes ('...') en vez de doble comilla ("...."). El profile_path es una ruta relativa que debe combinarse con una URL base para mostrar imágenes. Si la pelicula no tiene genero se determinara una lista vacia.
ratings	JSON	Representan las calificaciones asignadas por diferentes usuarios a la película. userId: Identificador único del usuario quien realizó la calificación. rating: La calificación otorgada a la película. timestamp: Marca de tiempo.	Si la pelicula no tiene genero se determinara una lista vacia. Algunos "rating" estan dentro de apostrofes ('...') en vez de doble comilla (".....").

4.5.2 Resultados de estadística descriptiva

```
--- Análisis de la columna 'title' ---
Número total de títulos: 99
Número de títulos únicos: 98
Título más largo: Lock, Stock and Two Smoking Barrels
Longitud promedio de los títulos: 72.00

Frecuencia de los 5 títulos más comunes:
Unicorn City                                                    2
Unguarded                                                       1
Eddie: The Sleepwalking Cannibal                                1
Follow Me: The Yoni Netanyahu Story                             1
Quints                                                          1
```

```
--- Estadísticas para 'Runtime' ---
Conteo: 99
Media: 99.17
Mínimo: 0.00
Máximo: 360.00
Desviación Estándar: 43.71

--- Estadísticas para 'Vote Average' ---
Conteo: 99
Media: 5.43
Mínimo: 0.00
Máximo: 9.50
Desviación Estándar: 2.37

--- Estadísticas para 'Vote Count' ---
Conteo: 99
Media: 257.89
Mínimo: 0.00
Máximo: 6656.00
Desviación Estándar: 1034.90
```

```
C:\Users\User\.jdks\openjdk-23.0.1\bin\java.exe ...
--- Estadísticas para 'Budget' ---
Conteo: 99
Media: 3588282.83
Mínimo: 0.00
Máximo: 130000000.00
Desviación Estándar: 18723357.91

--- Estadísticas para 'Popularity' ---
Conteo: 99
Media: 2.40
Mínimo: 0.00
Máximo: 26.88
Desviación Estándar: 5.00

--- Estadísticas para 'Revenue' ---
Conteo: 99
Media: 16625218.92
Mínimo: 0.00
Máximo: 847423452.00
Desviación Estándar: 100131385.84
```

```
--- Análisis de la columna 'original_language' ---
Número total de idiomas: 99
Número de idiomas únicos: 14
Frecuencia de los 5 idiomas más comunes:
en          75
fr          7
da          3
it          2
es          2


Process finished with exit code 0
```

4.5.3 Informe técnico

https://github.com/PFR-Computacion-O24F25/b2-proyecto-integrador-b-cotopaxi.git

4.5.4 Proyecto Final

https://github.com/JU4NSCV/PRACTICUM1.1.git

4.6 Entregable 2.

4.6.1 Proceso de limpieza de datos.

```
val belongsToCollectionData = datosFiltrados.flatMap { fila =>
  for {
    movieId <- Try(fila("id").trim.toInt).toOption
    jsonStr = fila.getOrElse("belongs_to_collection", "").trim if jsonStr.nonEmpty && jsonStr != "\"\""
    jsonLimpio = limpiarJsonCrew(jsonStr).replaceAll("'", "\"")
    jsonObj <- Try(Json.parse(jsonLimpio).validate[JsObject]).toOption.collect { case JsSuccess(obj, _) => obj }
    id <- (jsonObj \ "id").asOpt[Int]
    name = (jsonObj \ "name").asOpt[String].filter(_.nonEmpty).getOrElse("null")
    posterPath = (jsonObj \ "poster_path").asOpt[String].filter(_.nonEmpty).getOrElse("null")
    backdropPath = (jsonObj \ "backdrop_path").asOpt[String].filter(_.nonEmpty).getOrElse("null")
  } yield (movieId, id, name, posterPath, backdropPath)
}
```

```
val belongsToCollectionData = datosFiltrados.flatMap { fila =>
  for {
    movieId <- Try(fila("id").trim.toInt).toOption
    jsonStr = fila.getOrElse("belongs_to_collection", "").trim if jsonStr.nonEmpty && jsonStr != "\"\""
    jsonLimpio = limpiarJsonCrew(jsonStr).replaceAll("'", "\"")
    jsonObj <- Try(Json.parse(jsonLimpio).validate[JsObject]).toOption.collect { case JsSuccess(obj, _) => obj }
    id <- (jsonObj \ "id").asOpt[Int]
    name = (jsonObj \ "name").asOpt[String].filter(_.nonEmpty).getOrElse("null")
    posterPath = (jsonObj \ "poster_path").asOpt[String].filter(_.nonEmpty).getOrElse("null")
    backdropPath = (jsonObj \ "backdrop_path").asOpt[String].filter(_.nonEmpty).getOrElse("null")
  } yield (movieId, id, name, posterPath, backdropPath)
}
```

```
datosFiltrados.foreach { fila =>
  for {
    movieId <- Try(fila("id").trim.toInt).toOption
    jsonStr = fila.getOrElse("keywords", "").trim if jsonStr.nonEmpty && jsonStr != "\"\""
    jsonLimpio = limpiarJson(jsonStr).replaceAll("'", "\"")
    jsonArray <- Try(Json.parse(jsonLimpio).as[JsArray]).toOption
  } jsonArray.value.foreach { obj =>
    obj.validate[Keyword] match {
      case JsSuccess(keyword, _) =>
        keywordsList += keyword
        movieKeywordsList += ((movieId, keyword.id, keyword.name))
      case JsError(errors) =>
        println(s"Error en el JSON de keywords: $errors")
    }
  }
}
```

4.7 Entregable 3.

4.7.1 Script funcional

```
def convertPeliculaData(data: List[Map[String, String]]): List[  ⚠ JU4NSCV
  (String, Long, String, Long, String, String, String, String, Double, String, String, Long, Int, String, String, String, String, Double, Int)
] = {
  data.map { row =>
    (
      row.get("adult").filter(_.nonEmpty).getOrElse("").trim, // Strings: Asignar "" si es None o está vacio
      row.get("budget").flatMap(s => Try(s.toLong).toOption).getOrElse(0L), // Long: Asignar 0 si es None o no es un número válido
      row.get("homepage").filter(_.nonEmpty).getOrElse("").trim,
      row.get("id").flatMap(s => Try(s.toLong).toOption).getOrElse(0L),
      row.get("imdb_id").filter(_.nonEmpty).getOrElse("").trim,
      row.get("original_language").filter(_.nonEmpty).getOrElse("").trim,
      row.get("original_title").filter(_.nonEmpty).getOrElse("").trim,
      row.get("overview").filter(_.nonEmpty).getOrElse("").trim,
      row.get("popularity").flatMap(s => Try(s.toDouble).toOption).getOrElse(0.0), // Double: Asignar 0.0 si es None o no es un número válido
      row.get("poster_path").filter(_.nonEmpty).getOrElse("").trim,
      row.get("release_date").filter(_.nonEmpty).getOrElse("").trim,
      row.get("revenue").flatMap(s => Try(s.toLong).toOption).getOrElse(0L),
      row.get("runtime").flatMap(s => Try(s.toInt).toOption).getOrElse(0), // Int: Asignar 0 si es None o no es un número válido
      row.get("status").filter(_.nonEmpty).getOrElse("").trim,
      row.get("tagline").filter(_.nonEmpty).getOrElse("").trim,
      row.get("title").filter(_.nonEmpty).getOrElse("").trim,
      row.get("video").filter(_.nonEmpty).getOrElse("").trim,
      row.get("vote_average").flatMap(s => Try(s.toDouble).toOption).getOrElse(0.0),
      row.get("vote_count").flatMap(s => Try(s.toInt).toOption).getOrElse(0)
    )
  }
}
```

Método donde recibo una lista de mapas de String para poder trabajar con la inserción de datos, creando un script en y limpiando la lista para evitar errores. Este concepto se llevo a

cabo en todas las inserciones, recibiendo en otros casos una lista de tuplas, limpiando y generando el script donde se encuentran los datos listos para la inserción en la base de datos.

Referencias Bibliográficas

IBM. (s.f.). *Modelado de modelos de datos lógicos*. IBM Documentation. Recuperado el 26/01/2025 de https://www.ibm.com/docs/es/ida/9.1.2?topic=modeling-logical-data-models