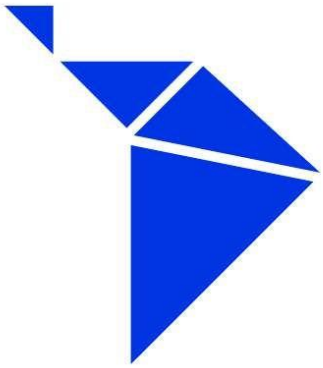
APLATAM



APLATAM

EVALUACIÓN DE HABILIDADES EN UN DESARROLLO DE UN FLUJO ETL

LUIS ARMANDO FLORES CRUZ

09 DE FEBRERO DE 2025

Objetivo

Evaluar tus habilidades en el desarrollo de un flujo ETL, utilizando como herramientas Python, SQL y Github

Esquematización y Diseño

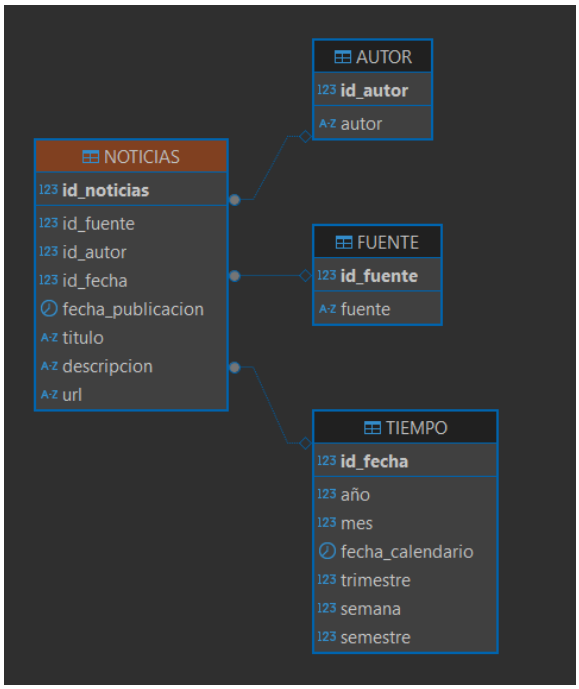
Para esta evaluación consideré utilizar railway, MySQL y Pyhton ya que las herramientas ya que las herramientas son flexibles, ergonómicas para el desarrollo de un reporte de noticias.

Extracción de datos desde una API Publica

Opté por utilizar NewsAPI por que vi que el manejo de documentación y la construcción de la consulta es más rápida, consideré usar el servicio gratuito para cumplir con la evaluación.

Solución

Modelo Entidad Relación



El modelo estrella es adecuado para un reporte o indicador de noticias.

Noticias es la tabla histórica o transaccional y las constelaciones conectadas son las de autor, fuente y tiempo.

Con este modelo controlamos en una tabla de hechos los IDs y fechas y el rendimiento de la tabla será adecuado para un gran volumen de información.

Las dimensiones de autor y fuentes las construí tomándolo de la API, así aseguramos tener un catálogo de datos maestros.

La dimensión tiempo es para poder filtrar la información por cualquier segmento de fecha (año, mes , trimestre , semana), para el diseño de un reporte o indicador es mas ergonómico incluir estos campos.

Explicación del Data pipeline

- Conexión a la Base de Datos
  - Archivo: conexion.py
  - Configura la conexión con MySQL en Railway.
- Preparación de entorno
  - Archivo: Preparacion_Entorno.py
  - Crea las dimensiones AUTOR, FUENTE, EXTRACTOR_NOTICIAS
  - Creación y llenado de información de la tabla TIEMPO
  - Creación de la tabla NOTICIAS
- Extracción de Datos
  - Archivo: Extractor.py
  - Obtiene noticias de la API NewsAPI sobre Tesla.
  - Guarda los datos en la tabla EXTRACTOR_NOTICIAS.
  - También inserta datos en las tablas FUENTE y AUTOR.
- Transformación y Carga
  - Archivo: Trans_Carga.py
  - Toma los datos de EXTRACTOR_NOTICIAS, limpia y normaliza títulos y descripciones.
  - Carga los datos en la tabla final NOTICIAS.
- Ejecución del Proceso Completo
  - Archivo: Main.py
  - Crea las tablas necesarias y borra datos antiguos.
  - Extrae, transforma y carga los datos ejecutando todas las funciones en orden.

Scripts incluyendo comentarios:

Main.py:

```python
from datetime import datetime
from conexion import conectar_db  # Importamos la función de conexión a la base de datos
from Preparacion_Entorno import (
    crear_tabla_extractor, crear_tabla_NOTICIAS, crear_tabla_tiempo,
    insertar_datos_tiempo, eliminar_datos_tiempo, crear_tabla_fuente,
    crear_tabla_autor
)
from Extractor import obtener_noticias, guardar_noticias, dimension_fuentes, dimension_autor
from Trans_Carga import transformador_carga_noticias

# Punto de entrada del script
if __name__ == "__main__":

    # Configuración de fechas para cargas iniciales de información
    FECHA_INICIO = "2025-02-08"
    FECHA_FIN = "2025-02-09"

    # Cuando se completen las cargas iniciales, estas líneas se comentarán y se activarán las cargas incrementales
    # FECHA_INICIO = datetime.today().strftime("%Y-%m-%d")
    # FECHA_FIN = datetime.today().strftime("%Y-%m-%d")

    # ------------------------------------------
    # Preparación del entorno: Creación de tablas
    # ------------------------------------------
    crear_tabla_autor()
    crear_tabla_fuente()
    crear_tabla_extractor()
    crear_tabla_tiempo()
    eliminar_datos_tiempo()
    insertar_datos_tiempo()
    crear_tabla_NOTICIAS()

    # ------------------------------------------
    # Extracción, transformación y carga de datos
    # ------------------------------------------
    noticias = obtener_noticias(FECHA_INICIO, FECHA_FIN)

    if noticias:
        guardar_noticias(noticias)  # Guarda las noticias extraídas en la base de datos
        dimension_fuentes()  # Procesa la dimensión de fuentes de noticias
        dimension_autor()  # Procesa la dimensión de autores
        transformador_carga_noticias(FECHA_INICIO, FECHA_FIN)  # Transformación y carga final de los datos
```

APLATAM

Extractor.py:

```python
import requests
from datetime import datetime
from conexion import conectar_db  # Importamos la función de conexión a la base de datos

# Obtener noticias de la API
def obtener_noticias(FECHA_INICIO, FECHA_FIN):
    """
    Obtiene noticias desde la API de NewsAPI dentro del rango de fechas especificado.
    """
    API_KEY = "6c3523005d044a9bbdc713b1edccb524"
    URL = f"https://newsapi.org/v2/everything?q=tesla&from={FECHA_INICIO}&to={FECHA_FIN}&sortBy=publishedAt&apiKey={API_KEY}"
    response = requests.get(URL)

    if response.status_code == 200:
        return response.json()["articles"]
    else:
        print("Error al obtener noticias:", response.status_code)
        print("Respuesta:", response.text)  # Imprime el contenido completo de la respuesta
        return []

# Guardar noticias en la base de datos
def guardar_noticias(noticias):
    """
    Guarda las noticias obtenidas en la base de datos.
    """
    conn = conectar_db()
    if conn:
        cursor = conn.cursor()

        # Eliminar todos los registros existentes en la tabla
        cursor.execute("DELETE FROM EXTRACTOR_NOTICIAS")
        conn.commit()
        print("🗑️ Todos los registros han sido eliminados de la tabla EXTRACTOR_NOTICIAS.")

        # Insertar nuevas noticias
        query = """
            INSERT INTO EXTRACTOR_NOTICIAS (fuente, autor, titulo, descripcion, url, fecha_publicacion)
            VALUES (%s, %s, %s, %s, %s, %s)
        """

        for noticia in noticias:
            fuente = noticia["source"]["name"]
            autor = noticia["author"] if noticia["author"] else "Desconocido"
            titulo = noticia["title"]
            descripcion = noticia["description"]
            url = noticia["url"]
            fecha_publicacion = datetime.strptime(noticia["publishedAt"], "%Y-%m-%dT%H:%M:%SZ")

            cursor.execute(query, (fuente, autor, titulo, descripcion, url, fecha_publicacion))

        # Confirmar los cambios en la base de datos
        conn.commit()
        print("✅ Noticias de Tesla guardadas correctamente en extractor noticias.")

        cursor.close()  # Cerrar cursor
        conn.close()  # Cerrar conexión

# Procesar la dimensión de fuentes
def dimension_fuentes():
    """
    Elimina e inserta registros en la tabla de fuentes.
    """
    conn = conectar_db()
    if conn:
        try:
            cursor = conn.cursor()

            cursor.execute("DELETE FROM FUENTE")
            conn.commit()

            cursor.execute("INSERT INTO FUENTE (fuente) SELECT DISTINCT UPPER(fuente) AS fuente FROM railway.EXTRACTOR_NOTICIAS")
            conn.commit()
            print("✅ Todos los registros han sido eliminados e insertados en la tabla fuentes")
        except Exception as e:
            print(f"❌ Error al eliminar e insertar datos en fuentes: {e}")
        finally:
            cursor.close()
            conn.close()
```

APLATAM

```python
# Procesar la dimensión de autores
def dimension_autor():
    """
    Elimina e inserta registros en la tabla de autores.
    """
    conn = conectar_db()
    if conn:
        try:
            cursor = conn.cursor()

            cursor.execute("DELETE FROM AUTOR")
            conn.commit()

            cursor.execute("INSERT INTO AUTOR (autor) SELECT DISTINCT UPPER(autor) AS autor FROM railway.EXTRACTOR_NOTICIAS")
            conn.commit()
            print("✅ Todos los registros han sido eliminados e insertados en la tabla autor")
        except Exception as e:
            print(f"❌ Error al eliminar e insertar datos en autor: {e}")
        finally:
            cursor.close()
            conn.close()
```

APLATAM

Trans_Carga.py

```python
import requests
from datetime import datetime
from conexion import conectar_db  # Importamos la función para conectar con la base de datos

def transformador_carga_noticias(FECHA_INICIO, FECHA_FIN):
    """
    Transforma y carga datos en la tabla NOTICIAS eliminando registros existentes en el rango de fechas
    especificado y luego insertando nuevos datos desde la tabla EXTRACTOR_NOTICIAS.

    Parámetros:
        FECHA_INICIO (str): Fecha de inicio en formato 'YYYY-MM-DD'.
        FECHA_FIN (str): Fecha de fin en formato 'YYYY-MM-DD'.
    """

    conn = conectar_db()  # Establece la conexión con la base de datos
    if conn:
        try:
            cursor = conn.cursor()

            # Eliminar registros dentro del rango de fechas especificado
            query_delete = f"""
                DELETE FROM NOTICIAS
                WHERE DATE(fecha_publicacion) BETWEEN '{FECHA_INICIO}' AND '{FECHA_FIN}'
            """
            cursor.execute(query_delete)
            conn.commit()

            # Insertar nuevos datos transformados en la tabla NOTICIAS
            query_insert = """
                INSERT INTO NOTICIAS (id_fuente, id_autor, id_fecha, fecha_publicacion, titulo, descripcion, url)
                SELECT
                    f.id_fuente AS id_fuente,
                    a.id_autor AS id_autor,
                    t.id_fecha AS id_fecha,
                    n.fecha_publicacion,
                    UPPER(n.titulo) AS titulo,
                    UPPER(n.descripcion) AS descripcion,
                    n.url
                FROM
                    railway.EXTRACTOR_NOTICIAS AS n
            LEFT JOIN
                FUENTE AS f ON n.fuente = f.fuente
            LEFT JOIN
                AUTOR AS a ON n.autor = a.autor
            LEFT JOIN
                TIEMPO AS t ON DATE(n.fecha_publicacion) = t.fecha_calendario
        """
        cursor.execute(query_insert)
        conn.commit()

        print("✅ Todos los registros han sido eliminados e insertados en la tabla NOTICIAS")

    except Exception as e:
        print(f"❌ Error al eliminar e insertar datos en NOTICIAS: {e}")

    finally:
        cursor.close()  # Cerrar el cursor
        conn.close()    # Cerrar la conexión con la base de datos
```

Preparacion_entorno.py

```python
from datetime import datetime, timedelta
from conexion import conectar_db  # Importamos la función de conexión desde el archivo de conexión

# Función para crear la tabla AUTOR si no existe
def crear_tabla_autor():
    conn = conectar_db()  # Conectamos a la base de datos
    if conn:
        try:
            cursor = conn.cursor()  # Creamos un cursor para ejecutar la consulta
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS AUTOR (
                    id_autor INT AUTO_INCREMENT PRIMARY KEY,  # ID único para cada autor
                    autor VARCHAR(255)                        # Nombre del autor
                )
            """)
            conn.commit()  # Confirmamos los cambios en la base de datos
            print("✅ Tabla `AUTOR` verificada.")
        except Exception as e:
            print(f"❌ Error al crear/verificar la tabla AUTOR: {e}")
        finally:
            cursor.close()  # Cerramos el cursor
            conn.close()    # Cerramos la conexión

# Función para crear la tabla FUENTE si no existe
def crear_tabla_fuente():
    conn = conectar_db()
    if conn:
        try:
            cursor = conn.cursor()
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS FUENTE (
                    id_fuente INT AUTO_INCREMENT PRIMARY KEY,  # ID único para cada fuente
                    fuente VARCHAR(255)                        # Nombre de la fuente
                )
            """)
            conn.commit()
            print("✅ Tabla `FUENTE` verificada.")
        except Exception as e:
            print(f"❌ Error al crear/verificar la tabla FUENTE: {e}")
```

```
        finally:
            cursor.close()
            conn.close()

# Función para crear la tabla EXTRACTOR_NOTICIAS si no existe
def crear_tabla_extractor():
    conn = conectar_db()
    if conn:
        try:
            cursor = conn.cursor()
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS EXTRACTOR_NOTICIAS (
                    id INT AUTO_INCREMENT PRIMARY KEY,       # ID único para cada noticia
                    fuente VARCHAR(255),                     # Fuente de la noticia
                    autor VARCHAR(255),                      # Autor de la noticia
                    titulo TEXT,                             # Título de la noticia
                    descripcion TEXT,                        # Descripción de la noticia
                    url TEXT,                                # URL de la noticia
                    fecha_publicacion DATETIME               # Fecha y hora de publicación
                )
            """)
            conn.commit()
            print("✅ Tabla `EXTRACTOR_NOTICIAS` verificada.")
        except Exception as e:
            print(f"❌ Error al crear/verificar la tabla EXTRACTOR_NOTICIAS: {e}")
        finally:
            cursor.close()
            conn.close()

# Función para crear la tabla TIEMPO si no existe
def crear_tabla_tiempo():
    conn = conectar_db()
    if conn:
        try:
            cursor = conn.cursor()
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS TIEMPO (
                    id_fecha INT AUTO_INCREMENT PRIMARY KEY,  # ID único para cada fecha
                    año INT,                                  # Año de la fecha
```

```
                    año INT,                                # Año de la fecha
                    mes INT,                                # Mes de la fecha
                    fecha_calendario DATE,                  # Fecha en formato calendario
                    trimestre INT,                          # Trimestre de la fecha
                    semana INT,                             # Semana del año
                    semestre INT                            # Semestre del año
                )
            """)
            conn.commit()
            print("✅ Tabla `TIEMPO` verificada.")
        except Exception as e:
            print(f"❌ Error al crear/verificar la tabla TIEMPO: {e}")
        finally:
            cursor.close()
            conn.close()

### Llenado de información para la tabla TIEMPO ###

# Función para generar los datos para la tabla TIEMPO
def generar_datos_tiempo():
    fecha_inicio = datetime(2025, 1, 1)  # Fecha de inicio para los datos
    fecha_fin = datetime(2030, 12, 31)  # Fecha final para los datos
    delta = timedelta(days=1)            # Incremento de un día para cada iteración
    datos = []                           # Lista para almacenar los datos generados

    while fecha_inicio <= fecha_fin:
        # Obtención de los valores para cada columna
        año = fecha_inicio.year
        mes = fecha_inicio.month
        fecha_calendario = fecha_inicio.strftime('%Y-%m-%d')
        trimestre = (mes - 1) // 3 + 1
        semana = fecha_inicio.isocalendar()[1]
        semestre = 1 if mes <= 6 else 2

        datos.append((año, mes, fecha_calendario, trimestre, semana, semestre))  # Añadimos los datos a la lista
        fecha_inicio += delta  # Avanzamos un día

    return datos
```

```python
# Función para eliminar todos los registros de la tabla TIEMPO
def eliminar_datos_tiempo():
    conn = conectar_db()

    if conn:
        try:
            cursor = conn.cursor()
            cursor.execute("DELETE FROM TIEMPO")  # Eliminamos todos los registros de la tabla TIEMPO
            conn.commit()
            print(f"✅ Se han eliminado los registros en la tabla TIEMPO.")
        except Exception as e:
            print(f"❌ Error al eliminar datos en la tabla TIEMPO: {e}")
        finally:
            cursor.close()
            conn.close()

# Función para insertar los datos generados en la tabla TIEMPO
def insertar_datos_tiempo():
    datos = generar_datos_tiempo()  # Generamos los datos
    conn = conectar_db()

    if conn:
        try:
            cursor = conn.cursor()
            cursor.executemany(
                "INSERT INTO TIEMPO (año, mes, fecha_calendario, trimestre, semana, semestre) VALUES (%s, %s, %s, %s, %s, %s)",
                datos  # Insertamos todos los datos generados
            )
            conn.commit()
            print(f"✅ Se han insertado {len(datos)} registros en la tabla TIEMPO.")
        except Exception as e:
            print(f"❌ Error al insertar datos en la tabla TIEMPO: {e}")
        finally:
            cursor.close()
            conn.close()

# Función para crear la tabla NOTICIAS con las claves foráneas relacionadas
def crear_tabla_NOTICIAS():
    conn = conectar_db()
```

```
if conn:
    try:
        cursor = conn.cursor()
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS NOTICIAS (
                id_noticias INT AUTO_INCREMENT PRIMARY KEY,  # ID único para cada noticia
                id_fuente INT,                               # ID de la fuente (clave foránea)
                id_autor INT,                                # ID del autor (clave foránea)
                id_fecha INT,                                # ID de la fecha (clave foránea)
                fecha_publicacion DATETIME,                  # Fecha y hora de publicación
                titulo TEXT,                                 # Título de la noticia
                descripcion TEXT,                            # Descripción de la noticia
                url TEXT,                                    # URL de la noticia
                CONSTRAINT fk_noticias_fuente FOREIGN KEY (id_fuente) REFERENCES railway.FUENTE (id_fuente) ON DELETE CASCADE ON UPDATE CASCADE,
                CONSTRAINT fk_noticias_autor FOREIGN KEY (id_autor) REFERENCES railway.AUTOR (id_autor) ON DELETE SET NULL ON UPDATE CASCADE,
                CONSTRAINT fk_tiempo FOREIGN KEY (id_fecha) REFERENCES TIEMPO(id_fecha) ON DELETE SET NULL ON UPDATE CASCADE
            )
        """)
        conn.commit()
        print("✅ Tabla `NOTICIAS` verificada.")
    except Exception as e:
        print(f"❌ Error al crear/verificar la tabla NOTICIAS: {e}")
    finally:
        cursor.close()
        conn.close()
```

Conexión.py

```
import mysql.connector

# Configuración de la base de datos en Railway
# Aquí se guardan los detalles de conexión a la base de datos
DB_CONFIG = {
    "host": "monorail.proxy.rlwy.net",  # Dirección del host de la base de datos
    "port": 36716,                      # Puerto de conexión a la base de datos
    "user": "root",                     # Usuario de la base de datos
    "password": "AMQvzlgsvrCRqqRBWDycqlmtDaAeEBlf",  # Contraseña para la conexión
    "database": "railway"               # Nombre de la base de datos a la que nos conectamos
}

# Función para conectar a la base de datos
# Intenta establecer una conexión con la base de datos usando los parámetros de DB_CONFIG
def conectar_db():
    try:
        # Intentamos realizar la conexión a la base de datos
        conn = mysql.connector.connect(**DB_CONFIG)
        return conn  # Retorna la conexión si es exitosa
    except mysql.connector.Error as err:
        # En caso de error, imprimimos un mensaje con la descripción del error
        print("❌ Error en la base de datos:", err)
        return None  # Retorna None si la conexión falla
```

README

Proyecto de Base de Datos con MySQL

Este proyecto maneja la creación y gestión de tablas en una base de datos MySQL para almacenar información relacionada con autores, fuentes, noticias, y tiempos. Está diseñado para ejecutarse con Python, utilizando la biblioteca `mysql-connector` para interactuar con MySQL.

Requisitos previos

Antes de comenzar, asegúrate de tener instalado lo siguiente en tu entorno:

- **Python 3.x**: Asegúrate de tener una versión reciente de Python instalada. Puedes verificar tu versión ejecutando el siguiente comando en tu terminal o consola:

  ```bash

  python --version

###MySQL Server: Debes tener una instancia de MySQL en ejecución. Si estás utilizando servicios en la nube, asegúrate de tener las credenciales correctas para acceder a la base de datos.

Instalación

Clona el repositorio (o descarga los archivos del proyecto). Si estás usando Git, ejecuta:

bash

Copiar

Editar

git clone <URL_DEL_REPOSITORIO>

cd <nombre_del_repositorio>

Crea un entorno virtual (opcional pero recomendado): Si no tienes un entorno virtual, puedes crear uno con los siguientes comandos:

bash

Copiar

Editar

python -m venv env

Activa el entorno virtual:

En Windows:

bash

Copiar

Editar

.\env\Scripts\activate
  ```

En macOS/Linux:

bash

Copiar

Editar

source env/bin/activate

Instala las dependencias: Asegúrate de tener el archivo requirements.txt en el proyecto (si no lo tienes, puedes crear uno con las dependencias necesarias). Ejecuta el siguiente comando para instalar las bibliotecas necesarias:

bash

Copiar

Editar

pip install -r requirements.txt

Si no tienes el archivo requirements.txt, instala mysql-connector directamente con:

bash

Copiar

Editar

pip install mysql-connector

Configuración de la Base de Datos

Conexión a la Base de Datos: Asegúrate de modificar las credenciales de conexión a la base de datos en el archivo conexion.py (o donde se gestionen las configuraciones de conexión). Aquí es donde se especifican los datos como el host, usuario, contraseña y base de datos de MySQL.

python

Copiar

Editar

```
DB_CONFIG = {
    "host": "tu_host",
    "port": 3306,  # Cambia si usas otro puerto
    "user": "tu_usuario",
    "password": "tu_contraseña",
    "database": "tu_base_de_datos"
}
```

Ejecución del Proyecto

solo ejecuta el archivo Main.py , dicho archivo hará todo el procedimiento de :

*creación de tablas

*Extracción tranformación y carga de datos

Estructura de la Base de Datos

El proyecto creará las siguientes tablas en la base de datos:

AUTOR: Almacena información sobre los autores.

id_autor: Identificador único del autor (clave primaria).

autor: Nombre del autor.

FUENTE: Almacena información sobre las fuentes.

id_fuente: Identificador único de la fuente (clave primaria).

fuente: Nombre de la fuente.

EXTRACTOR_NOTICIAS: Almacena las noticias extraídas.

id: Identificador único de la noticia (clave primaria).

fuente: Fuente de la noticia.

autor: Autor de la noticia.

titulo: Título de la noticia.

descripcion: Descripción de la noticia.

url: URL de la noticia.

fecha_publicacion: Fecha de publicación de la noticia.

TIEMPO: Almacena información sobre fechas y tiempos.

id_fecha: Identificador único de la fecha (clave primaria).

año: Año de la fecha.

mes: Mes de la fecha.

fecha_calendario: Fecha en formato calendario (YYYY-MM-DD).

trimestre: Trimestre de la fecha.

semana: Semana del año.

semestre: Semestre del año.

NOTICIAS: Relaciona noticias con fuentes, autores y fechas.

id_noticias: Identificador único de la noticia (clave primaria).

id_fuente: Clave foránea que referencia a FUENTE.

id_autor: Clave foránea que referencia a AUTOR.

id_fecha: Clave foránea que referencia a TIEMPO.

fecha_publicacion: Fecha y hora de publicación de la noticia.

titulo: Título de la noticia.

descripcion: Descripción de la noticia.

url: URL de la noticia.

Errores Comunes

Error de conexión: Si hay un error en la conexión a la base de datos, asegúrate de que las credenciales de la base de datos son correctas y que el servidor de MySQL está en ejecución.

Tablas ya existentes: Si las tablas ya existen en la base de datos, el código las verificará y no las recreará.