Estudiante: Diego Mateo Laguna Levy

Asignatura: Tecnologías Web II

Docente: Ing. Miguel Ángel Pacheco

Microservicio en GraphQL para el seguimiento de oyentes a creadores

1. Análisis del Proyecto Integrador (15 pts)

- Identificación clara del endpoint (5 pts)

Endpoint a consumir:

- POST usuarios/seguirCreador

Justificación:

- Se escogió el endpoint debido a que se encargará de almacenar los seguidos de los oyentes por medio de los ID de Usuarios y Creadores, lo cual sucederá de manera constante y ha de ser efectiva.
- Se eligió GraphQL debido a su flexibilidad de consultas.

Descripción Técnica

Método	POST
Url	Graphql/
Formato de datos (seguir)	JSON (usuarios_idusuarios, creadores_idcreador)
Formato de respuesta	JSON
Autenticación	**JWT (JSON Web Tokens)** mediante header Authorization: Bearer <token>.

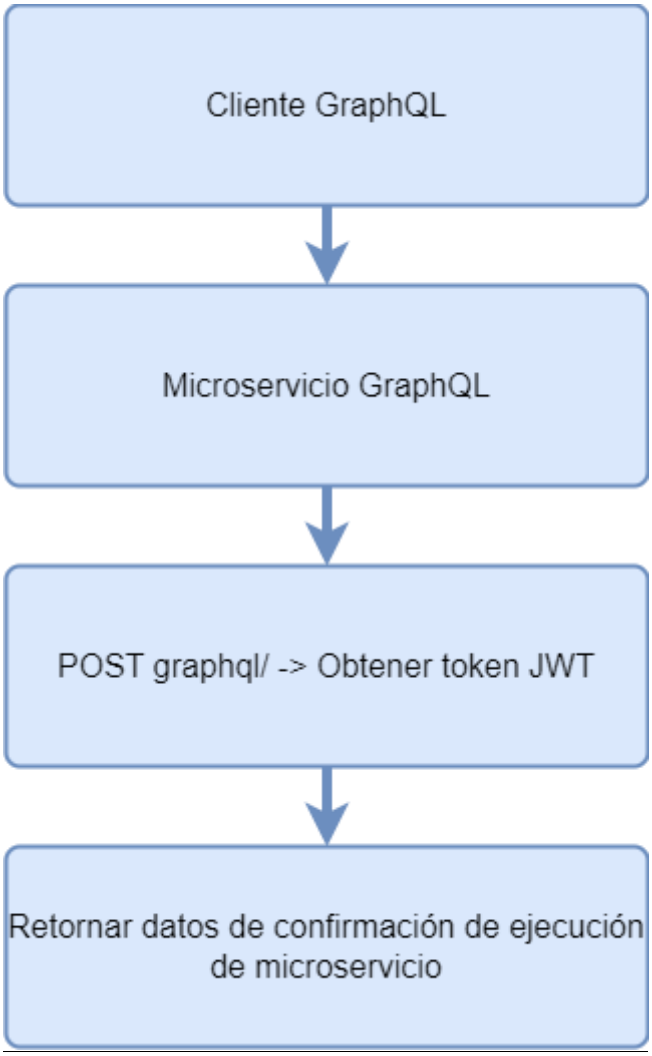
2. Diseño del microservicio

- Objetivo del microservicio claramente definido

Objetivo: Construir un microservicio **GraphQL** que permita a usuarios poder seguir a un creador de contenido ofreciendo una consulta rápida y efectiva.

- **Elección y justificación de la tecnología de comunicación**
    - GraphQL
        - Para las consultas solo requiere los campos necesarios.
        - Es ideal para front end exigentes debido a su capacidad de estructurar respuestas personalizadas.

- Diagrama del flujo de integración (5 pts)



3. Implementación Técnica (40 pts)

Estructura del proyecto

```
backend/
├── backend/
|   ├── __pycache__/
|   ├── models/
|   |   ├── __pycache__/
|   |   ├── Creadores.py
|   |   └── Usuarios.py
|   ├── templates/
|   ├── __init__.py
|   ├── .env
|   ├── apps.py
|   ├── asgi.py
|   ├── db_connection.py
|   ├── forms.py
|   ├── schema.py
|   ├── serializers.py
|   ├── settings.py
|   ├── urls.py
|   ├── views.py
|   ├── wsgi.py
|   └── db.sqlite3
├── manage.py
frontend/
```

Codigo base

- Schema.py

```
import graphene
from django.db import connection
from graphql import GraphQLError
import jwt
from django.conf import settings
from datetime import datetime, timedelta

class SeguirCreadorPayload(graphene.ObjectType):
    success = graphene.Boolean(required=True)
    usuarios_idusuario = graphene.Int()
    creadores_idcreador = graphene.Int()
```

```
class HealthCheckType(graphene.ObjectType):
  status = graphene.String()
  server_time = graphene.String()

class SeguirCreadorInput(graphene.InputObjectType):
  usuarios_idusuario = graphene.Int(required=True)
  creadores_idcreador = graphene.Int(required=True)

def get_user_id_from_token(context):
  auth_header = context.META.get('HTTP_AUTHORIZATION', '')
  if not auth_header.startswith('Bearer '):
    raise GraphQLError("Token inválido o faltante")

  token = auth_header.split(' ')[1]
  try:
    payload = jwt.decode(token, settings.SECRET_KEY,
algorithms=['HS256'])
    return payload.get('user_id')
  except jwt.ExpiredSignatureError:
    raise GraphQLError("Token expirado")
  except jwt.InvalidTokenError:
    raise GraphQLError("Token inválido")

class SeguirCreador(graphene.Mutation):
  class Arguments:
    input = SeguirCreadorInput(required=True)

  Output = SeguirCreadorPayload

  @staticmethod
  def mutate(root, info, input):
    try:
      user_id = get_user_id_from_token(info.context)

      if not input.usuarios_idusuario or not input.creadores_idcreador:
        raise GraphQLError("Faltan campos requeridos")

      with connection.cursor() as cursor:
        cursor.execute(
          """
```

```
            INSERT INTO backend_listaseguidos
            (usuarios_idusuario, creadores_idcreador)
            VALUES (%s, %s)
            RETURNING creadores_idcreador
            """,
            [input.usuarios_idusuario, input.creadores_idcreador]
          )
          cursor.fetchone()

        return SeguirCreadorPayload(
          success=True,
          usuarios_idusuario=input.usuarios_idusuario,
          creadores_idcreador=input.creadores_idcreador
        )

      except Exception as e:
        return SeguirCreadorPayload(
          success=False,
          usuarios_idusuario=None,
          creadores_idcreador=None
        )

class Query(graphene.ObjectType):
  health = graphene.Field(HealthCheckType, description="Verifica estado
del servicio")
  hola = graphene.String(description="Endpoint de prueba")

  def resolve_health(self, info):
    return HealthCheckType(
      status="OPERATIONAL",
      server_time=datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    )

  def resolve_hola(self, info):
    return "Mundo"

class Mutation(graphene.ObjectType):
  seguir_creador = SeguirCreador.Field()

schema = graphene.Schema(
  query=Query,
```

```
    mutation=Mutation,
    auto_camelcase=True
)

def generate_test_token(user_id=1, expires_in=24):
    """Genera un token JWT para pruebas manuales"""
    payload = {
        'user_id': user_id,
        'exp': datetime.utcnow() + timedelta(hours=expires_in)
    }
    return jwt.encode(payload, settings.SECRET_KEY, algorithm='HS256')

TEST_TOKEN = generate_test_token()
print(f"\n 🔑 Token de prueba (JWT):\n{TEST_TOKEN}\n")
```

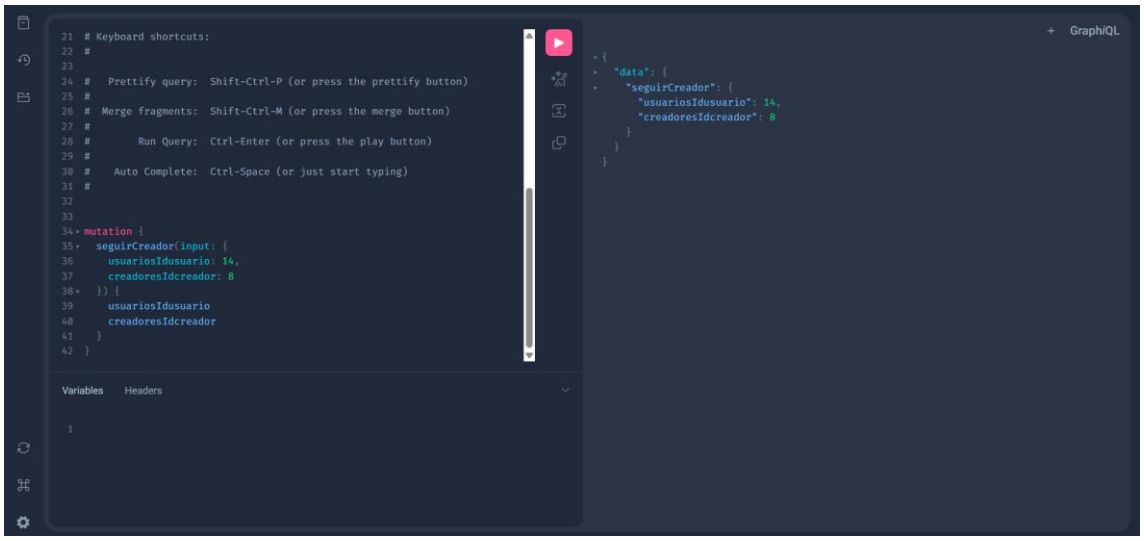
- Urls.py

```
urlpatterns = [
    path('graphql/', csrf_exempt(GraphQLView.as_view(graphiql=True))),
]
```

Librerías necesarias:

4. Pruebas y Documentación

- Evidencias funcionales



- **Pruebas manuales**
    - **Acceso sin token JWT**
      Query sin autenticación JWT

Query con autenticación JWT

- **Documentacion clara**
  - **Instrucciones de instalación**
    1. Clonar el repositorio:
    2. Ingresar a la carpeta cd backend/backend
    3. Instalar Django y dependencias principales
       pip install django==4.2.*
       graphene-django==3.1.*

django-cors-headers==4.3.*

4. Para iniciar el proyecto se ha de ejecutar el comando *Python manage.py runserver* en la diraccion backend/backend>
5. Para ejecutar el microservicio se ha de ingresar a la dirección local que levanta el backend con el endpoint *graphql/*

- **Ejemplo de consulta:**

```
mutation {
 seguirCreador(input: {
  usuariosIdusuario: 14,
  creadoresIdcreador: 7
 }) {
  success
 }
}
```

En la parte de Headers se ha de ingresar el token JWT generado:

```
{
 "Authorization":"Bearer  token"
}
```

- **Respuesta esperada**

```
{
 "data": {
  "seguirCreador": {
   "success": true
  }
 }
}
```