

## Microservicio con *gRPC* para la gestión de Usuarios y Creadores

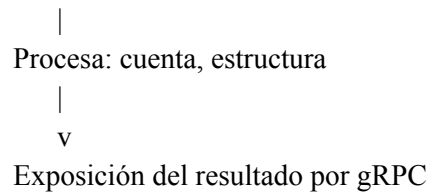
### 1. Análisis del Proyecto Integrador

- Identificación clara del endpoint
  - POST usuario/seguirCreador  
Endpoint para hacer que un usuario siga a un creador (solo para pruebas)
  - GET usuario/seguimientos\_procesados/[id]  
Este endpoint se utiliza para obtener los seguimientos de un usuario
- Justificación de la elección del endpoint  
Se eligió este endpoint porque:
  - Permite realizar aplicaciones para recomendaciones de otros creadores a usuarios
  - Resulta útil para generar estadísticas por creador
  - Es un endpoint que está disponible para uso y que este permite utilizar el id del usuario para probar el microservicio
- Descripción técnica del endpoint (estructura, método, formato de datos)

Característica	Descripción
Método	GET
URL	/usuarios/seguimientos_procesados
Formato de datos	Query/URL (usuarios_idusuario)
Formato de respuesta	JSON

### 2. Diseño del Microservicio

- Objetivo del microservicio claramente definido  
Desarrollar un microservicio que permita hacer consumo del endpoint usuarios/obtenerSeguimiento, al cual se le aplique escalabilidad, mejora para el envío del dato en un nuevo endpoint para facilidad del desarrollo, mostrando el total de creadores seguidos y el id de los creadores que sigue.
- Elección y justificación de la tecnología de comunicación  
gRPC fue elegido ya que:
  - Es eficiente y rápido en la conexión de servicios
  - Utiliza archivos *.proto* que permiten una manipulación más clara de la estructura
  - Serializa los datos más ligeros que en un JSON
- Diagrama del flujo de integración  
[gRPC Microservicio]
  - |
  - |---> Hace petición HTTP GET a --> [Django Backend: usuarios/seguimientos\_procesados/[id]]
  - |
  - |<--- Respuesta con JSON de creadores seguidos



### 3. Implementación Técnica

- Desarrollo del microservicio funcional
- Consumo correcto del endpoint externo
- Procesamiento y transformación de los datos
- Exposición de endpoint propio/documentado

```
backend/
├── grpc_client/
│   ├── __init__.py
│   ├── seguimientos_pb2.py          # Generado
│   ├── seguimientos_pb2_grpc.py    # Generado
│   └── seguimientos.proto          # Archivo .proto
├── servidor_grpc.py
├── urls.py
├── views.py
└── README.md
```

#### **seguimientos.proto:**

```
syntax = "proto3";
```

```
package seguimientos;
```

```
service SeguimientosService {
    rpc ObtenerSeguimientosProcesados(SeguimientosRequest) returns
    (SeguimientosResponse);
}
```

```
message SeguimientosRequest {
    string usuarios_idusuario = 1;
}
```

```
message SeguimientosResponse {
    int32 total = 1;
    repeated string creadores = 2;
    string error = 3;
}
```

#### **servidor\_grpc.py:**

```
import grpc
from concurrent import futures
import os
```

```

import sys
import django

sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'backend.settings')
django.setup()

from grpc_client import seguimientos_pb2, seguimientos_pb2_grpc
from django.db import connection

class SeguimientosService(seguimientos_pb2_grpc.SeguimientosServiceServicer):
    def ObtenerSeguimientosProcesados(self, request, context):
        try:
            user_id = request.usuarios_idusuario
            print(f"Llamada gRPC recibida con usuarios_idusuario={user_id}")

            with connection.cursor() as cursor:
                cursor.execute("""
                    SELECT creadores_idcreador
                    FROM backend_listaseguidos
                    WHERE usuarios_idusuario = %s
                """, [user_id])
                rows = cursor.fetchall()
                creadores = [str(row[0]) for row in rows]

            return seguimientos_pb2.SeguimientosResponse(
                total=len(creadores),
                creadores=creadores,
                error=""
            )
        except Exception as e:
            return seguimientos_pb2.SeguimientosResponse(
                total=0,
                creadores=[],
                error=str(e)
            )

def serve():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))

seguimientos_pb2_grpc.add_SeguimientosServiceServicer_to_server(SeguimientosService(),
server)
server.add_insecure_port('[::]:50051')
server.start()
print("Servidor gRPC activo en puerto 50051...")
server.wait_for_termination()

```

```
if __name__ == '__main__':  
    serve()
```

#### **urls.py:**

```
from django.contrib import admin  
from django.urls import path
```

```
from django.urls import path  
from . import views
```

```
urlpatterns = [  
    path('usuarios/crear/', views.crear_usuario, name='crear_usuario'),##con Postman  
    path('usuarios/listar/', views.listar_usuarios, name='listar_usuarios'),##con Postman  
    path('creadores/listar/', views.listar_creadores, name='listar_creadores'),  
    path('creadores/registrar', views.mostrar_formulario_registro, name='registrar'),  
    path('registro', views.registro, name='registro'),  
    path('creadores/mostrar', views.mostrar_creadores, name='mostrar_creadores'),  
    path('usuarios/seguirCreador', views.seguirCreador, name='seguir_creador'),##con Postman  
    path('usuarios/obtenerSeguimientos/', views.obtenerSeguimientos,  
name='obtener_seguimientos'),##con Postman  
    path('usuarios/seguimientos_procesados/<int:usuarios_idusuario>/',  
(views.seguimientos_procesados), name='seguimientos'),]
```

#### **views.py:**

```
# backend/views.py  
from django.http import JsonResponse  
from .db_connection import obtener_conexion  
from django.http import JsonResponse  
from django.views.decorators.csrf import csrf_exempt  
import json  
from .models.Usuarios import Usuario  
# backend/views.py  
from django.views.decorators.csrf import csrf_exempt  
from django.http import JsonResponse  
from .models import Creadores  
from django.utils.crypto import get_random_string  
from django.views import View  
import datetime  
from django.shortcuts import render  
from django.http import JsonResponse, HttpResponse  
import os  
  
import grpc  
from django.http import JsonResponse  
from backend.grpc_client import seguimientos_pb2, seguimientos_pb2_grpc
```

```

from django.shortcuts import render
from .settings import SUPABASE_URL, SUPABASE_KEY, DEBUG
from supabase import create_client, Client
# views.py

from django.http import JsonResponse
from django.contrib.auth.hashers import make_password
import os
import uuid # Importar uuid para generar nombres únicos

# Configurar la conexión con Supabase
url = SUPABASE_URL
key = SUPABASE_KEY
supabase: Client = create_client(url, key)
# views.py

from datetime import datetime

import logging

logger = logging.getLogger(__name__)

def seguimientos_procesados(request, usuarios_idusuario):
    if request.method != 'GET':
        return JsonResponse({'error': 'Método no permitido'}, status=405)

    try:
        with grpc.insecure_channel('localhost:50051') as channel:
            stub = seguimientos_pb2_grpc.SeguimientosServiceStub(channel)
            grpc_response = stub.ObtenerSeguimientosProcesados(
seguimientos_pb2.SeguimientosRequest(usuarios_idusuario=str(usuarios_idusuario))
            )

        return JsonResponse({
            'total': grpc_response.total,
            'creadores': list(grpc_response.creadores),
            'error': grpc_response.error
        })

    except Exception as e:

```

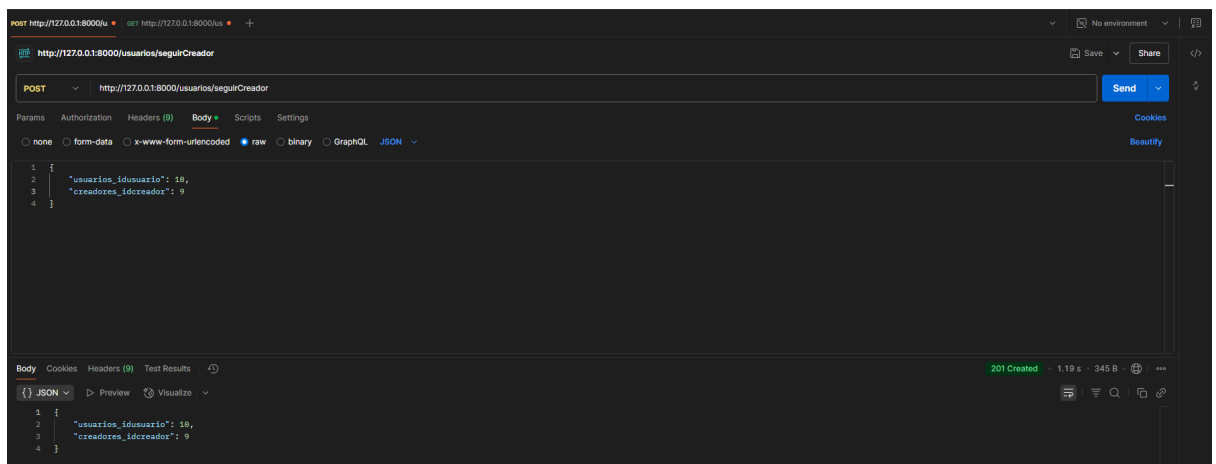
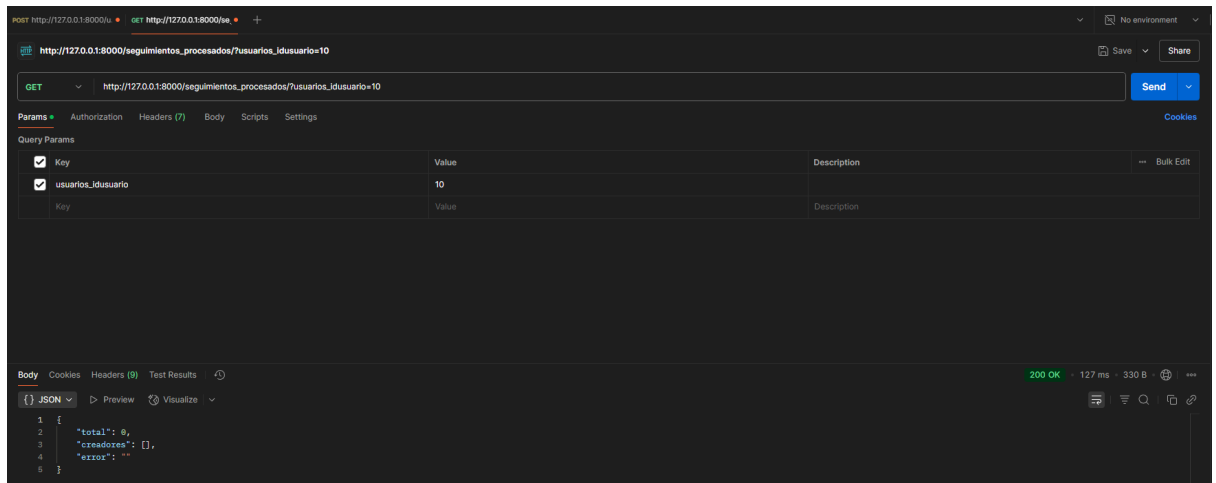
```
return JsonResponse({'error': str(e)}, status=500)
```

**Librerías necesarias:** pip install django grpcio grpcio-tools protobuf

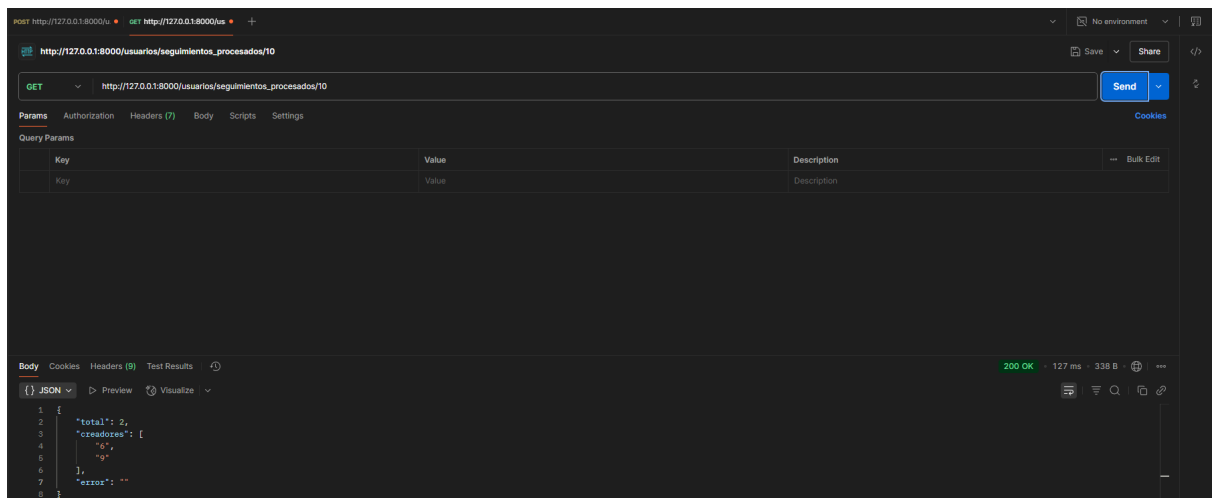
#### 4. Pruebas y Documentación

Evidencias funcionales (capturas, video, Postman, etc.)

Cargado de seguimientos



Resultado del endpoint con Postman



- Pruebas unitarias o manuales explicadas  
Pruebas manuales

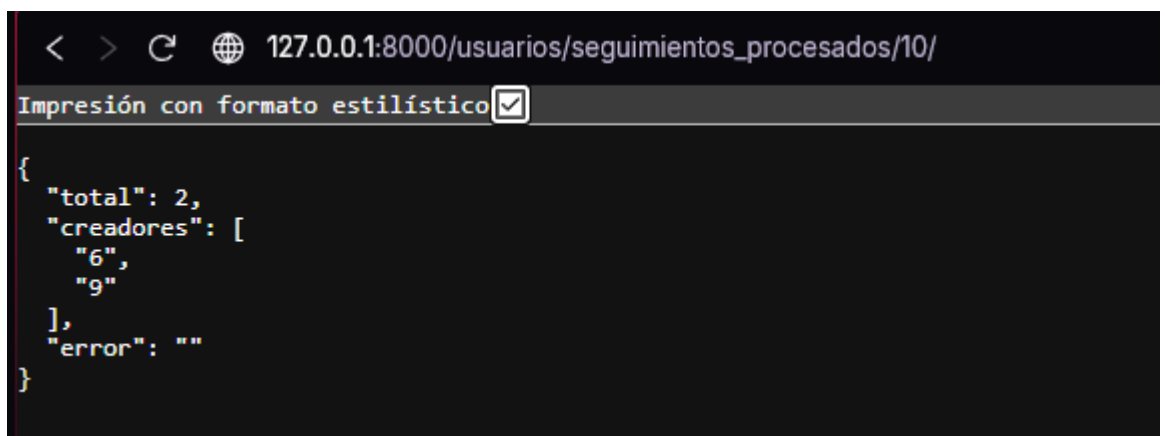
1. Iniciar el microservicio con: `python grpc\_server.py` en la carpeta AdrianGonzales\_2daEvaluacionBackend/backend/backend/backend`
2. Iniciar el backend con: `python manage.py runserver`
3. Realizar una petición GET a:  
`http://127.0.0.1:8000/usuarios/seguimientos\_procesados/10`
4. Validar que el resultado sea:

```
```json
{
  "total": 2,
  "creadores": ["6", "9"],
  "error": ""
}
```

```
PS C:\Users\Adrian\Documents\GitHub\AdrianGonzales_2daEvaluacionBackend> python backend/backend/backend/servidor_grpc.py
Servidor gRPC activo en puerto 50051...

C:\Users\Adrian\Documents\GitHub\AdrianGonzales_2daEvaluacionBackend\backend\backend\backend\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 03, 2025 - 23:30:46
Django version 5.2, using settings 'backend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



The screenshot shows a web browser window with the address bar displaying the URL `127.0.0.1:8000/usuarios/seguimientos_procesados/10/`. Below the address bar, there is a checkbox labeled "Impresión con formato estilístico" which is checked. The main content area of the browser displays a JSON response in a dark-themed editor:

```
{
  "total": 2,
  "creadores": [
    "6",
    "9"
  ],
  "error": ""
}
```

- Documentación clara del microservicio (README.md o PDF)
5. Presentación Final
    - Explicación técnica del microservicio y su integración en PDF adjunto al repositorio
    - Demostración funcional clara