Desarrollar un servicio de proxy en Python que pueda manejar y analizar datos de conversaciones, con énfasis en el rendimiento, la escalabilidad y el uso de buenas prácticas de programación.

# Requisitos Funcionales

## 1. Servicio de Proxy

- Implementar un servicio que actúe como proxy para un API ficticio `cualquierapi.com`.
- El servicio debe poder manejar peticiones como: `curl 127.0.0.1:8080/categories/MLA97994` y redirigirlas a `cualquierapi.com/categories/MLA97994`.
- Utilizar FastAPI como framework web para el servicio.

## 2. API de Conversaciones

- Crear endpoints para recuperar datos de conversaciones basados en filtros.
- Implementar un endpoint como: `GET /api/conversations?company=microsoft&tags=competition,orientation`

La respuesta debe tener un formato similar a:
json
Copy

```
{
  "conversation_ids": ["conv-123", "conv-456", "conv-789"],
  "total": 3,
  "page": 1,
  "per_page": 50

}
```

## 3. Procesamiento de Datos

- Implementar un sistema para procesar datos en paralelo, aprovechando las capacidades asíncronas de FastAPI.
- Crear una cola para manejar tareas que no se pueden procesar en paralelo.

## 4. Estadísticas y Monitoreo

- Almacenar y visualizar estadísticas de uso del proxy.

- Registrar tasas de éxito de las llamadas y tiempos de respuesta.
- Utilizar las capacidades de FastAPI para generar documentación automática de la API.

### 5. Rendimiento

- La solución debe estar diseñada para manejar una carga media de 50,000 solicitudes por segundo.
- Aprovechar las características de alto rendimiento de FastAPI y su soporte para operaciones asíncronas.

## Requisitos Técnicos

1. Usar static typing en todas las funciones (se recomienda el uso de Pydantic, que está integrado con FastAPI).
2. Implementar manejo de excepciones, utilizando el  manejador de errores de FastAPI.
3. Proporcionar documentación clara del código y aprovechar la documentación automática de FastAPI.
4. Incluir unit test para las funciones principales, utilizando herramientas compatibles con FastAPI como TestClient.

## Entregables

1. Código fuente del proyecto.
2. Un README con instrucciones de instalación y ejecución.
3. Diagrama de arquitectura explicando el diseño, funcionamiento y escalabilidad del sistema.
4. Explicación escrita de las decisiones de diseño y arquitectura tomadas, incluyendo cómo se ha aprovechado FastAPI para cumplir con los requisitos de rendimiento.

## Datos de Prueba

Se adjunta un zip con datos de muestra en formato Parquet con una estructura similar a: company, transcript, conversation, tags.

https://drive.google.com/file/d/1j4sXWoj5vS-as_d9GmQQ3S6YC26I6_9x/view?usp=sharing

## Evaluación

Se evaluará:

- Funcionalidad completa según los requisitos.
- Calidad y legibilidad del código.

- Eficiencia y escalabilidad de la solución, especialmente en el uso de características asíncronas de FastAPI.
- Manejo de errores y casos límite.
- Claridad en la documentación y explicaciones de arquitectura.

## Notas Adicionales

- Aunque se prefiere una implementación completa, se aceptará cualquier nivel de completitud.
- Si no es posible conectarse al API real de `cualquierapi.com`, se puede usar un mock o cualquier otra alternativa que demuestre el funcionamiento del proxy.
- La interfaz para estadísticas y control puede soportar REST, aprovechando las capacidades de FastAPI para crear APIs RESTful.
- Se valorará el uso efectivo de las características de FastAPI, como dependencias, validación de datos con Pydantic, y operaciones asíncronas.