

Informe Tecnico: Clase 04 Mayo

Nicolás Ortiz¹, Juan Camilo morales²

¹⁻²Dpto. de Ingeniería,
Universidad Central
Maestría en Analítica de Datos
Curso de Automatización e integración de datos
Bogotá, Colombia
`{1nortizv,2jmoralesr10}@ucentral.edu.co`,³

May 13, 2024

1 Resumen o resumen ejecutivo

Presentación Apache Hadoop

Introducción

Apache Hadoop es un marco de software que permite la computación distribuida de grandes conjuntos de datos a través de grupos de computadoras utilizando modelos de programación simples. Es conocido por su capacidad para escalar desde un solo servidor hasta miles de máquinas, cada una ofreciendo almacenamiento y procesamiento local.

Contenidos del Informe

¿Qué es Apache Hadoop?

Apache Hadoop es una plataforma de código abierto para la automatización e integración de datos, diseñada para soportar aplicaciones de big data y analítica avanzada, incluyendo la inteligencia artificial. Diferencia entre Hadoop y Apache Hadoop Mientras que "Hadoop" se refiere generalmente a la plataforma de código abierto para el almacenamiento y procesamiento de big data, "Apache Hadoop" se refiere específicamente al proyecto desarrollado y mantenido por la Apache Software Foundation. Incluye una variedad de subproyectos que complementan la funcionalidad básica de Hadoop.

Generalidades

Hadoop fue diseñado inicialmente para superar las limitaciones de los sistemas tradicionales de gestión de bases de datos (DBMS) al manejar datos no estructurados y semiestructurados. Su arquitectura se basa en la distribución y replicación de datos a través de múltiples nodos, lo que mejora la fiabilidad y el rendimiento.

Componentes Principales

HDFS (Hadoop Distributed File System): Sistema de archivos distribuido que almacena datos de manera redundante para asegurar la disponibilidad y fiabilidad.
MapReduce: Modelo de programación que permite el procesamiento paralelo de grandes conjuntos de datos.
YARN (Yet Another Resource Negotiator): Gestor de recursos y planificación de trabajos que permite a múltiples aplicaciones utilizar Hadoop.
Hadoop Common: Conjunto de utilidades y bibliotecas comunes que soportan los otros módulos de Hadoop.

Ventajas de Hadoop

Escalabilidad Horizontal: Capacidad de aumentar el rendimiento añadiendo más nodos en lugar de mejorar las especificaciones de los nodos existentes.

Alta Disponibilidad y Tolerancia a Fallos: Replicación de datos y capacidades de recuperación que aseguran la disponibilidad continua de los datos y la integridad en caso de fallos.

Costo-Efectividad: Utilización de hardware común y de bajo costo para construir clusters grandes y potentes.

Flexibilidad: Capacidad para manejar datos estructurados y no estructurados de diversas fuentes.

Paralelismo Masivo: Procesamiento de grandes volúmenes de datos de manera paralela, acelerando significativamente el tiempo de análisis.

Ecosistema Robusto: Amplio conjunto de herramientas y proyectos complementarios como Hive, Pig, HBase, y Spark que extienden las capacidades de Hadoop.

Conclusión

Apache Hadoop se ha consolidado como una solución poderosa y flexible para el manejo de grandes volúmenes de datos. Su arquitectura distribuida y su ecosistema robusto lo hacen una opción preferida para empresas que buscan aprovechar el big data para obtener insights valiosos y mejorar sus procesos de negocio.

Proyecto Chatbot

Introducción

Este proyecto tiene como objetivo desarrollar un chatbot para la Universidad Politécnico Grancolombiano, utilizando Peticiones, Quejas, Reclamos y Sugerencias (PQRS) internas como fuente de información. El chatbot se basa en un Large Language Model (LLM) combinado con Retrieval Augmented Generation (RAG) para proporcionar respuestas precisas y contextualizadas.

Contenidos del Informe

Título del Trabajo de Grado “Desarrollo de un Chatbot para la Universidad Politécnico Grancolombiano, basado en las PQRS internas, mediante Large Language Model (LLM) con Retrieval Augmented Generation (RAG)”.

Introducción y Justificación

En Colombia, la gestión de PQRS es fundamental para mantener la confianza y satisfacción de la comunidad educativa. La Universidad Politécnico Grancolombiano enfrenta el desafío de manejar un volumen creciente de PQRS de manera eficiente. La propuesta busca desarrollar un chatbot que aproveche la tecnología LLM y RAG para mejorar la gestión de estas interacciones, reduciendo la carga administrativa y mejorando la calidad del servicio.

Pregunta de Investigación

¿Cómo se puede mejorar la eficiencia, precisión y satisfacción en el proceso de respuesta a los usuarios en la Universidad Politécnico Grancolombiano?

Objetivo General

Desarrollar un Chatbot para la Universidad Politécnico Grancolombiano, basado en las PQRS internas, mediante Large Language Model (LLM) con Retrieval Augmented Generation (RAG).

Objetivos Específicos

Describir las características y frecuencias de las PQRS para identificar las demandas más comunes.

Diseñar una estructura de base de datos para la recuperación de información relevante. Evaluar la efectividad de las respuestas proporcionadas por el chatbot utilizando LLM y RAG.

Antecedentes

El manejo eficiente de PQRS en el contexto educativo es crucial para mejorar la experiencia estudiantil. Estudios previos han demostrado que la integración de RAG en LLM mejora significativamente la precisión y relevancia de las respuestas generadas. Estos modelos combinan información de bases de datos externas para evitar errores comunes conocidos como "alucinaciones".

Fuente de los Datos Los datos serán anonimizados y suministrados por el departamento de servicio y permanencia de la universidad. Se utilizarán datos históricos de PQRS de los años 2021, 2022 y 2023, en formatos CSV, PDF y HTML.

Aplicación y/o Aporte Específico al Campo

El desarrollo de este chatbot mejorará la gestión de PQRS en la Universidad Politécnico Grancolombiano, optimizando la eficiencia operativa y reduciendo los tiempos de respuesta. Este proyecto también puede servir como modelo replicable para otras instituciones educativas.

Metodología o Actividades Específicas

Fase 1: Diseño y Planificación Definición de requisitos, selección de tecnologías y diseño de arquitectura.

Fase 2: Desarrollo y Configuración Configuración del entorno de desarrollo, implementación de la base de datos y desarrollo del chatbot.

Fase 3: Pruebas y Evaluación Pruebas funcionales, evaluación de la interfaz y validación del contenido de las respuestas.

Fase 4: Implementación y Despliegue Integración del sistema, despliegue del chatbot y capacitación de usuarios.

Fase 5: Monitoreo y Mejora Continua Monitoreo del desempeño, recopilación de feedback y actualizaciones periódicas.

Recursos

Humanos: Director de proyecto, desarrolladores de software.

Tecnológicos: PC para servidores, software de desarrollo, plataformas de pruebas.

Datos: Acceso a bases de datos de PQRS y documentación relevante.

Financieros: Presupuesto operativo cubierto por los estudiantes.

Cronograma de Actividades

Definición de requisitos (Semanas 1-2).

Selección de tecnologías y diseño de arquitectura (Semanas 3-4).

Configuración del entorno de desarrollo (Semanas 3-5).

Implementación de la base de datos (Semanas 5-6).

Desarrollo del chatbot (Semanas 7-10).

Pruebas funcionales y evaluación de la interfaz (Semanas 11-12).
Validación del contenido de respuestas y ajustes (Semanas 13-14).
Implementación, despliegue y capacitación de usuarios (Semanas 15-16).

Resultados Esperados

Automatización de respuestas a PQRS, mejorando la eficiencia operativa y la satisfacción de los usuarios. Optimización de la carga administrativa y mejora continua del sistema basado en el feedback de los usuarios.

Conclusión

La implementación de un chatbot basado en LLM y RAG promete transformar la gestión de PQRS en la Universidad Politécnico Grancolombiano, ofreciendo respuestas precisas y relevantes que mejoran la experiencia del usuario y optimizan los procesos administrativos. Este proyecto establece un precedente importante en la utilización de tecnologías avanzadas para la automatización y mejora de tareas administrativas en entornos educativos.

Taller Integración de datos con REST

Objetivo

Aprender los conceptos básicos de REST y cómo consumir un servicio RESTFUL usando un cliente HTTP.

¿Qué es APIs?

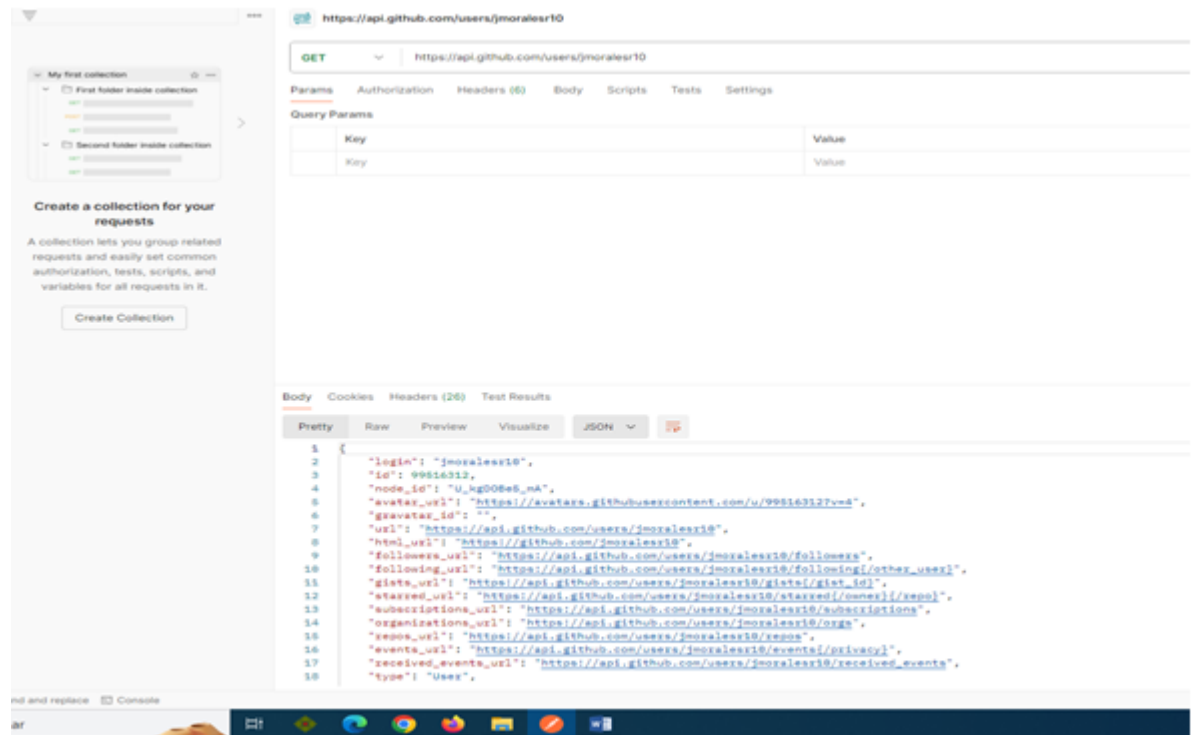
Una API, o Interfaz de Programación de Aplicaciones, es un conjunto de reglas y protocolos que permiten que diferentes aplicaciones se comuniquen entre sí. Se utilizan para acceder a servicios web, integrar software, automatizar tareas y desarrollar aplicaciones, permitiendo la interacción controlada entre sistemas de software. Las API pueden tomar diversas formas, como API REST, API SOAP o API GraphQL, y son esenciales en la informática moderna.

¿Qué es RestFulApi? Un RESTful API (Application Programming Interface) es un conjunto de reglas y convenciones que permite a los sistemas interactuar entre sí a través de la web de una manera sencilla y eficiente. Se basa en el estilo arquitectónico REST (Representational State Transfer), que utiliza métodos HTTP estándar (como GET, POST, PUT y DELETE) para realizar operaciones en recursos identificados por URLs. Un RESTful API se caracteriza por ser stateless, lo que significa que cada solicitud contiene toda la información necesaria para comprenderla, sin necesidad de mantener un estado entre solicitudes. Esto lo convierte en una opción popular para la creación de servicios web que son fáciles de entender, escalables y ampliamente utilizados en el desarrollo de aplicaciones web y móviles.

Taller

Primera parte:

- Instala Postman.
- Abre Postman y crea una nueva solicitud GET.
- En la URL, escribe `https://api.github.com/users/jtunombre` .
- Haz clic en "Send".
- La respuesta de la solicitud debería ser un JSON con la información de tu perfil de GitHub.



Segunda parte:

- Abre tu navegador y ve a Postman.
- Si no tienes una cuenta en Postman, puedes registrarte de forma gratuita.
- Una vez registrado e iniciado sesión, verás el entorno de trabajo de Postman.
- En la sección de "Crear" en la parte izquierda, selecciona "Request".
- Asigna un nombre a tu solicitud, por ejemplo, "Obtener datos de una API".
- En la sección de "Request", selecciona el método "GET".

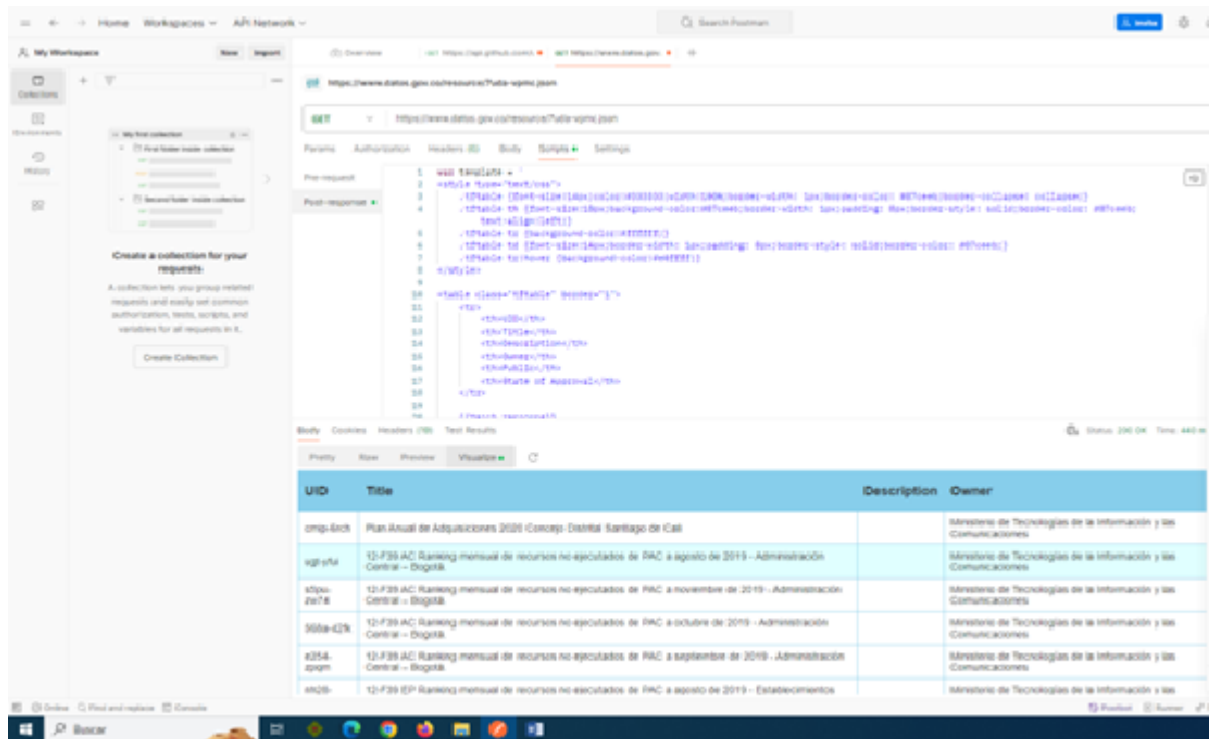
Ahora, necesitamos una API pública para hacer la solicitud. Utilizaremos la API JSONPlaceholder, que proporciona datos de ejemplo. Puedes realizar una solicitud GET a <https://jsonplaceholder.typicode.com/posts/1> para obtener un post de ejemplo.

En la barra de URL de Postman, ingresa

<https://jsonplaceholder.typicode.com/posts/1>.

<https://jsonplaceholder.typicode.com/comments>

<https://jsonplaceholder.typicode.com/photos>



Tercera parte

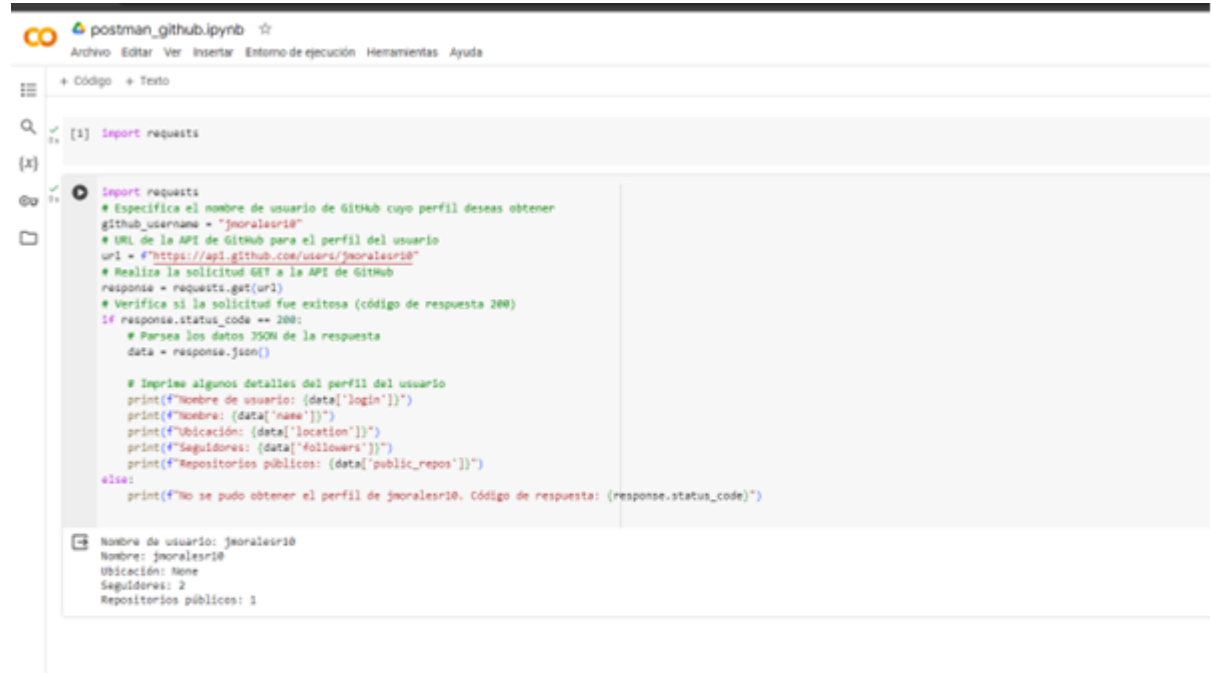
Utiliza la biblioteca requests en Python para realizar una solicitud GET a la API de GitHub y obtener los datos de perfil de una cuenta pública.

```
import requests
Especifica el nombre de usuario de GitHub cuyo perfil deseas obtener
github_usuario = "nombredeusuario"
URLdeAPIdeGitHubparaelfildelusuario
url = f"https://api.github.com/users/{github_usuario}"
RealizasolicitudGETaAPIdeGitHubresponse = requests.get(url)
Verificasilasolicitudfueexitosa(códigoderespuesta200)ifresponse.status_code == 200 :
```

```
ParsealosdatosJSONdelarespuestadata = response.json()
```

```
Imprime algunos detalles del perfil del usuario
print(f'Nombre de usuario: {data["login"]}')
print(f'Nombre: {data["name"]}')
print(f'Ubicación: {data["location"]}')
print(f'Seguidores: {data["followers"]}')
print(f'Repositorios públicos: {data["public_repos"]}')
else :
print(f'No se pudo obtener el perfil de {github_usuario}. Código
```

derespuesta : response.status_code”)



The screenshot shows a Jupyter Notebook interface with a file named 'postman_github.ipynb'. The notebook contains a single cell with a REST client request to the GitHub API. The request is a GET to 'https://api.github.com/users/jmoralesr10'. The response is a JSON object containing user information. The output of the cell shows the parsed JSON data.

```
[1] Import requests

Import requests
# Especifica el nombre de usuario de GitHub cuyo perfil deseas obtener
github_username = "jmoralesr10"
# URL de la API de GitHub para el perfil del usuario
url = f"https://api.github.com/users/{github_username}"
# Realiza la solicitud GET a la API de GitHub
response = requests.get(url)
# Verifica si la solicitud fue exitosa (código de respuesta 200)
if response.status_code == 200:
    # Parsea los datos JSON de la respuesta
    data = response.json()

    # Imprime algunos detalles del perfil del usuario
    print(f"Nombre de usuario: {data['login']}")
    print(f"Nombre: {data['name']}")
    print(f"Ubicación: {data['location']}")
    print(f"Seguidores: {data['followers']}")
    print(f"Repositorios públicos: {data['public_repos']}")
else:
    print(f"No se pudo obtener el perfil de {github_username}. Código de respuesta: {response.status_code}")
```

Nombre de usuario: jmoralesr10
Nombre: jmoralesr10
Ubicación: None
Seguidores: 2
Repositorios públicos: 1