PROYECTO DESARROLLO DE SOLUCIONES CLOUD
ENTREGA 1 – APLICACIONES WEB ESCALABLES EN UN ENTORNO TRADICIONAL (FRONTEND & BACKEND)

OBJETIVOS

1. Diseñar y desarrollar una aplicación web con arquitectura modular que permita escalabilidad horizontal y vertical en entornos tradicionales.

2. Diseñar escenarios de prueba que permitan medir la capacidad máxima que pueden soportar algunos componentes del sistema. El resultado debe expresarse en el documento del plan de análisis de capacidad con el que se simula el acceso, la carga, el estrés y la utilización de la aplicación.

3. Ejecutar las pruebas de estrés que permitan dimensionar la capacidad de una aplicación y su infraestructura de soporte con base al plan de pruebas.

TIEMPO DE DEDICACIÓN

Esta entrega "Frontend + Backend REST" del proyecto está planeado para tres semanas, en las que cada estudiante deberá invertir las horas definidas en la planeación semanal. Recuerde que para el éxito de la actividad es necesario que conforme un equipo de trabajo según las pautas definidas en el curso.

ESQUEMA DE EVALUACIÓN

Distribución de calificaciones de la entrega:
1. Verificación funcional de los requisitos de la aplicación (Backend): 70%
2. Implementación y funcionalidad del Frontend: 15%
3. Documentación técnica:
    a. Historias de usuario con criterios de aceptación.
    b. Diagramas de despliegue y arquitectura.
    c. Documentación de los servicios del API REST.
    d. Instrucciones para la ejecución y despliegue.
    e. Ponderación total: 15%

LUGAR Y FORMATO DE ENTREGA

La entrega está conformada por:

1. Aplicación en ejecución sobre contenedores Docker basados en GNU/Linux. Debe incluir Dockerfiles para la generación de las imágenes y Docker Compose para el despliegue y ejecución la aplicación en su totalidad.
2. Documentación de la aplicación.
3. Crear un release del código fuente en el repositorio del grupo en GitHub/GitLab.
4. Entregar toda la documentación vía GitHub/GitLab. Puede apoyarse de la Wiki o subir los documentos en PDF al repositorio. El repositorio debe estar ordenado y la información de la entrega y el release debe ser de fácil acceso.

5. Video sustentación, realice un video de máximo 15 minutos donde pruebe cada uno de los endpoints utilizando POSTMAN y cárguelo en su repositorio. Los últimos 3 o 4 minutos del video deben incluir la demostración del frontend funcional.

INFRAESTRUCTURA REQUERIDA PARA EL DESPLIEGUE

En este proyecto se planea utilizar la siguiente infraestructura y servicios:

- Una máquina virtual como servidor único, aprovisionada en AWS Academy. La máquina virtual debe contar con las siguientes especificaciones técnicas 2 vCPU, 4 GiB RAM, 20 GiB en almacenamiento.
- Se recomienda utilizar alguna de las distribuciones mencionadas a continuación: Ubuntu Server, Debian o Alpine Linux (esta es una distro muy liviana). Cada grupo deberá configurar su ambiente con el servidor de aplicaciones, librerías, base de datos y en general todos los componentes necesarios para el despliegue correcto de la aplicación utilizando Docker.
- El despliegue de todos los componentes requeridos por la aplicación. Esto puede incluir servidores como Gunicorn, o Uvicorn, incluso Nginx.

Nota: Los monitores pueden solicitarle una sustentación síncrona, independiente a la video sustentación. Se recomienda dejar el ambiente configurado en una máquina virtual sobre AWS Academy para dicho propósito y las instrucciones para replicar su ambiente de ejecución.

DOCUMENTACIÓN – ARQUITECTURA DE LA APLICACIÓN

Se deberá entregar un documento que describa la arquitectura de la aplicación, así como las consideraciones necesarias para garantizar su escalabilidad y capacidad de atender a cientos de usuarios finales de manera concurrente. Además, el documento deberá incluir una descripción detallada de las limitaciones del desarrollo realizado.

El nombre de este documento deberá ser: "Entrega 1 - Arquitectura, conclusiones y consideraciones.pdf"

RECOMENDACIONES Y CONSIDERACIONES

En este proyecto se deberá desarrollar la aplicación Web con las funcionalidades mencionadas. El backend de la aplicación deberá ser desarrollado en el lenguaje de programación **Python** y el framework de su preferencia**.**

La aplicación deberá contar, como mínimo, con tres componentes para su ejecución en un entorno local (en servidores propios): Ubuntu Server 24.04 LTS como sistema operativo, Nginx y Gunicorn como servidor web y PostgreSQL o MySQL como motor de base de datos. El servidor web debe ser capaz de gestionar solicitudes de forma concurrente. En esta fase, la implementación no se realizará en un proveedor de nube pública; dicha migración se llevará a cabo en la siguiente entrega.

Se recomienda utilizar el siguiente stack tecnológico para el desarrollo de la entrega:

1. **Python 3.12 o superior**.

2. **Fast API o Flask**: Frameworks web.

3. **SQLAlchemy**: ORM SQLAlchemy, mapeador relacional de objetos que simplifica la interacción con una base de datos SQL.

4. **Celery**: es una biblioteca de Python para gestionar colas de tareas distribuidas, es decir, nos permite trabajar con diferentes tareas y distribuirlas en procesos.

5. **Redis o RabbitMQ**: servicios de mensajería que actúan como intermediarios; los brokers enrutan, agregan y permiten crear servicios de publicación/subscripción, comúnmente llamados servicios pub/sub.

6. **JWT**: Uso de JSON Web Tokens (JWT) a para proteger las vistas.

7. **Werkzeug**: biblioteca completa de aplicaciones web WSGI.

8. **PostgreSQL o MySQL:** Motor open source de base de datos SQL.

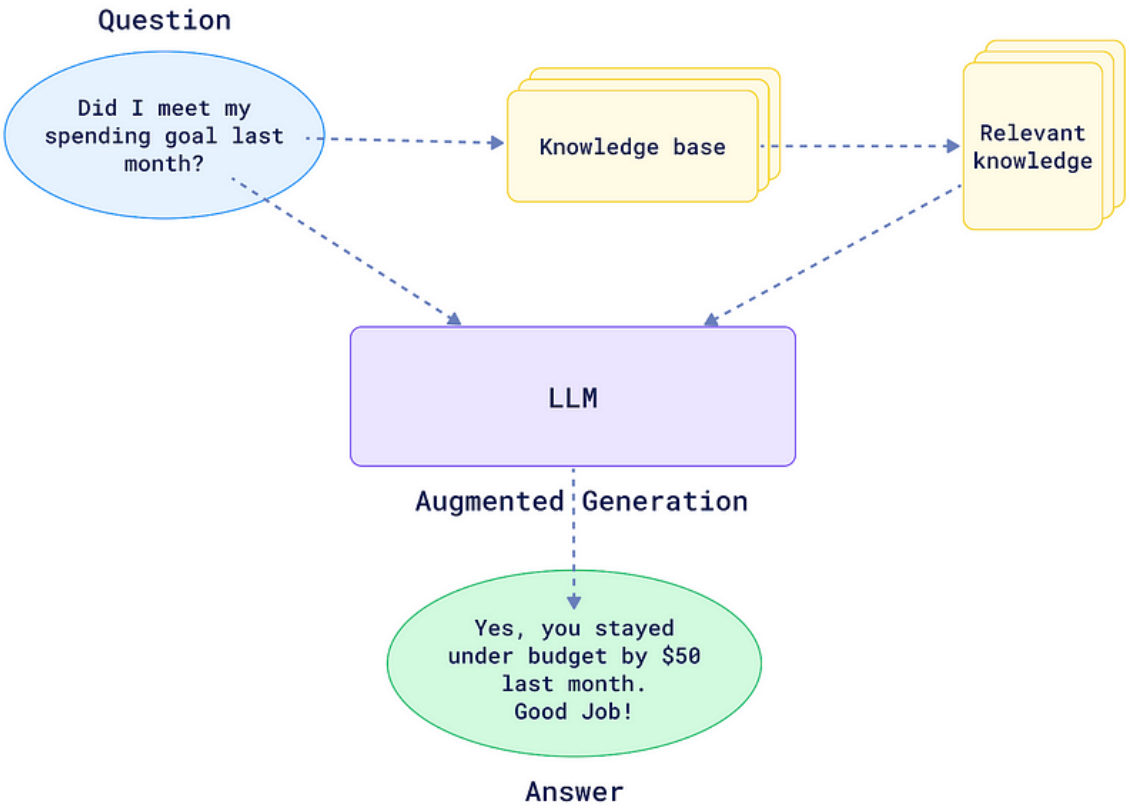
9. **Gunicorn o Uvicorn:** servidor HTTP WSGI para Python y ambientes Unix.

10. **Nginx:** servidor web HTTP de código abierto. Ofrece servicios como proxy inverso, balanceador de carga HTTP y proxy de correo electrónico para IMAP, POP3 y SMTP. Se utiliza en conjunto con Gunicorn o Uvicorn para desplegar en producción aplicaciones web en Python.

11. Para el **fronend** utilice la tecnología de su preferencia.

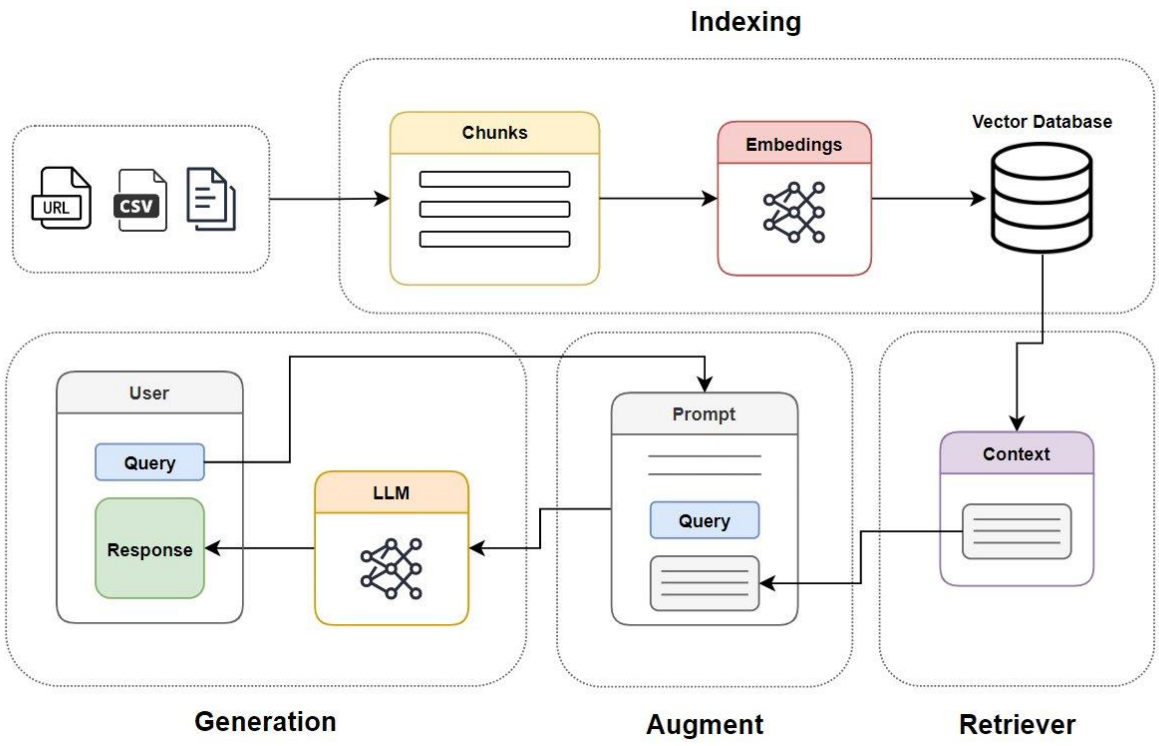
Aplicación: RAG SaaS

En el marco del curso de Desarrollo de Soluciones Cloud, los estudiantes desarrollarán una aplicación SaaS que permita la gestión, análisis y resumen automatizado de documentos mediante inteligencia artificial generativa. Esta aplicación se inspirará en herramientas como NotebookLM, integrando capacidades de procesamiento de lenguaje natural (NLP) para ofrecer a los usuarios la posibilidad de cargar documentos y obtener resúmenes estructurados, explicaciones detalladas y respuestas contextualizadas.



La solución deberá diseñarse con una arquitectura escalable y eficiente en costos, utilizando servicios cloud adecuados para almacenamiento, procesamiento de datos y despliegue de modelos de IA.

La solución es una **aplicación web basada** que utiliza **inteligencia artificial generativa** para asistir en la gestión, análisis y síntesis de información.



Alcance del Proyecto

El alcance de la aplicación incluirá las siguientes funcionalidades principales:

1. **Carga de documentos**: Los usuarios podrán subir archivos en formatos como **PDF, TXT, Word o Markdown**.
2. **Procesamiento de texto con IA**: La aplicación deberá integrar un LLM local para:
   - Generar **resúmenes** de documentos. Organiza la información y genera resúmenes, explicaciones y respuestas contextualizadas.
   - Proporcionar **explicaciones detalladas** sobre conceptos clave.
   - Responder preguntas basadas en el contenido cargado.
3. **Interfaz Web y API**: Se deberá desarrollar una interfaz web funcional y una API REST para el backend de la aplicación.
4. **Gestión de usuarios y sesiones**: Autenticación segura para que los usuarios puedan gestionar su historial de documentos.
5. **Despliegue**: La solución deberá desplegarse utilizando Docker y Docker Compose escalabilidad, disponibilidad y optimización de costos.
6. **Algunas herramientas recomendadas**: LangChain, LlamaIndex, ollama y LLMs pequeños.