Crear y desplegar un **Chatbot de Preguntas y Respuestas (Q&A)** .El chatbot debe ser capaz de consultar una base de conocimiento vectorial alojada en **Pinecone** para responder preguntas específicas, y funcionar dentro de una interfaz web de **Streamlit**.

🛠️ **Requerimientos Técnicos y Fases del Proyecto**

El proyecto se divide en tres fases principales que cubren todo el ciclo de vida del despliegue de una aplicación de IA Generativa.

Fase 1: Ingesta de Datos y Vectorización (LangChain & Hugging Face & Pinecone)

En esta fase, los estudiantes prepararán los datos y crearán la base de conocimiento vectorial que usará el chatbot.

1. **Fuente de Datos (Documentos):**
    - **Acción:** Seleccionar una fuente de documentación pública (ej. un manual de usuario, la documentación de una librería de Python, o un conjunto de archivos PDF/TXT) y guardarla en una carpeta local.
2. **Preparación de LangChain:**
    - **Acción:** Utilizar un **DocumentLoader** (ej. DirectoryLoader) para cargar los documentos.
    - **Acción:** Aplicar un **TextSplitter** (ej. RecursiveCharacterTextSplitter) para dividir los documentos grandes en fragmentos (chunks) con un tamaño y solapamiento adecuados.
3. **Embeddings y Pinecone:**
    - **Acción:** Configurar un modelo de **Embeddings** utilizando la librería **Hugging Face** (ej. HuggingFaceEmbeddings apuntando a un modelo como *all-MiniLM-L6-v2 o ADA*).
    - **Acción:** Inicializar la conexión con **Pinecone** (índice y *namespace* específicos).
    - **Acción:** Realizar el proceso de **upsert** (carga) de todos los *chunks* vectorizados en el índice de Pinecone, usando el modelo de *embeddings* de Hugging Face.

Fase 2: Construcción del Chatbot RAG (LangChain & Streamlit)

En esta fase, los estudiantes orquestarán la lógica del chatbot y construirán la interfaz de usuario.

1. **Orquestación con LangChain:**
    - **Acción:** Crear un **VectorStoreRetriever** a partir del índice de Pinecone que alimentaron en la Fase 1.
    - **Acción:** Construir la **Chain** principal de RAG (ej. RetrievalQA.from_chain_type o ConversationalRetrievalChain si se requiere historial).
    - **Acción:** Definir un **Prompt Template** que instruya al LLM a comportarse como un experto en la documentación cargada.
2. **Interfaz de Streamlit:**

- **Acción:** Diseñar la interfaz principal usando el layout de Streamlit.
- **Acción:** Utilizar **st.chat_message** y **st.chat_input** para la interacción conversacional.

3. **Optimización y Persistencia:**
    - **Acción:** Utilizar **@st.cache_resource** para cargar el modelo de *embeddings* y la conexión de Pinecone **solo una vez** al inicio de la aplicación.
    - **Acción:** Utilizar **st.session_state** para **mantener el historial de la conversación** completo entre las interacciones del usuario.

Fase 3: Despliegue y Validación (Streamlit)

1. **Configuración de Despliegue:**
    - **Acción:** Configurar las claves API (Pinecone y Hugging Face Token) de forma segura utilizando el archivo **secrets.toml** de Streamlit (o variables de entorno).
    - **Acción:** Crear un archivo **requirements.txt** con todas las dependencias exactas.
2. **Pruebas Funcionales:**
    - **Acción:** Probar el chatbot en preguntas que **requieren** la información del documento cargado en Pinecone. La respuesta debe citar o reflejar fielmente el contexto recuperado.
    - **Acción:** Probar el chatbot en preguntas de conocimiento general (fuera de la documentación) para verificar que el LLM funciona correctamente.