Stack

**Stack confirmado (actualizado)**

* **Lenguaje: Python.**
* **Frameworks:** 
  + **FastAPI (para endpoints).**
  + **Pandas (para manipulación y limpieza de datos).**
  + **Psycopg2 (para interactuar con Supabase PostgreSQL y PGVector).**
* **LLM/Embeddings:** 
  + **Hugging Face Transformers (modelo sentence-transformers como all-MiniLM-L6-v2 para embeddings).**
  + **Análisis de sentimiento:** 
    - **Hugging Face Transformers (modelo preentrenado distilbert-base-uncased-finetuned-sst-2-english).**
    - **VADER (análisis basado en reglas, rápido y efectivo para textos cortos).**
    - **TextBlob (análisis basado en patrones, simple y con polaridad/subjetividad).**
  + **Análisis de tópicos:** 
    - **Gensim (modelo LDA para extracción de tópicos).**
* **Almacenamiento vectorial: Supabase PGVector.**
* **Base de datos: Supabase PostgreSQL.**
* **Control de versiones: GitHub.**
* **Otras herramientas:** 
  + **Supabase Python Client (para interactuar con Supabase).**
  + **Matplotlib/Seaborn (para visualizaciones, como ranking de tópicos).**
  + **VS Code (entorno de desarrollo con scripts .py).**
  + **Scikit-learn (para preprocesamiento o utilidades, si es necesario).**
  + **Python-dotenv (para manejar credenciales).**

**Propuesta de funcionalidades**

Dado el límite de 48 horas y la instrucción de elegir 1 o 2 funcionalidades, propongo enfocarnos en:

1. **Generación y Guardado de Embeddings** (Funcionalidad 1):
   * **Por qué**: Es crítica para consultas RAG y análisis semántico, formando la base para futuros análisis. Implementar esto demuestra competencia en manejo de embeddings y almacenamiento vectorial, que son tecnologías avanzadas.
   * **Alcance**:
     + Procesar textos (mensajes, feedback, transcripciones).
     + Generar embeddings con sentence-transformers.
     + Guardar en Supabase PGVector con metadatos (userId, teamId, simulaciónId, tipo de interacción, timestamp).
     + Crear un endpoint FastAPI para verificar duplicados.
     + **Bonus**: Implementar limpieza básica de texto (por ejemplo, eliminar stopwords, normalizar texto).
   * **Impacto**: Muestra habilidades en integración con Supabase, manejo de embeddings y desarrollo de APIs.
2. **Análisis de Sentimiento y Tópicos Recurrentes** (Funcionalidad 2):
   * **Por qué**: Permite demostrar versatilidad con un modelo de ML preentrenado (Hugging Face) para análisis de sentimiento, combinado con una técnica simple como LDA para tópicos. Esto resalta tu capacidad para integrar enfoques modernos y escalables.
   * **Alcance**:
     + Procesar texto libre (feedback, transcripciones).
     + Analizar sentimiento con un modelo preentrenado de Hugging Face (distilbert-base-uncased-finetuned-sst-2-english).
     + Extraer tópicos con LDA (Gensim).
     + Guardar resultados en Supabase PostgreSQL (userId, simulaciónId, tipo de texto, resultado).
     + **Bonus**: Visualizar un ranking de tópicos por equipo con Matplotlib/Seaborn.
   * **Impacto**: Destaca tu conocimiento en NLP, ML preentrenado, y visualización, alineándose con el objetivo de evaluar habilidades blandas.

**Razonamiento**:

* Elegir ambas funcionalidades cubre un espectro amplio: **Funcionalidad 1** muestra competencia en almacenamiento vectorial y APIs, mientras que **Funcionalidad 2** destaca análisis avanzado con ML y visualización.
* El modelo preentrenado de Hugging Face es rápido de implementar (no requiere entrenamiento) y añade un componente moderno que impresionará en la prueba técnica.
* Priorizamos los **bonuses** (limpieza de texto y ranking de tópicos) porque son alcanzables en 48 horas y añaden valor al entregable.

**Cómo manejaremos las comparaciones en la Funcionalidad 2**

Para la **Funcionalidad 2 (Análisis de Sentimiento y Tópicos Recurrentes)**, implementaremos el análisis de sentimiento con tres métodos (**Hugging Face**, **VADER**, **TextBlob**) y compararemos sus resultados. Esto se reflejará en el código, la documentación (README), y posiblemente en una visualización. Aquí está el plan:

1. **Análisis de sentimiento**:
   * **Hugging Face Transformer**:
     + Usaremos distilbert-base-uncased-finetuned-sst-2-english para clasificar textos en positivo/negativo (con probabilidad).
     + Ventaja: Alta precisión, especialmente en contextos complejos.
   * **VADER**:
     + Proporciona una puntuación compuesta (compound) que clasifica el texto como positivo, negativo o neutral.
     + Ventaja: Rápido, optimizado para textos cortos (como mensajes o feedback).
   * **TextBlob**:
     + Devuelve polaridad (de -1 a 1) y subjetividad (de 0 a 1).
     + Ventaja: Simple, incluye subjetividad para evaluar cuán opinativo es el texto.
   * **Comparación**:
     + Procesaremos los mismos textos (por ejemplo, feedback o transcripciones) con los tres métodos.
     + Guardaremos los resultados en Supabase PostgreSQL con columnas para cada método (por ejemplo, hf\_sentiment, vader\_compound, textblob\_polarity).
     + Generaremos un resumen comparativo (por ejemplo, una tabla en el README o una visualización con Matplotlib/Seaborn) mostrando las diferencias en clasificación.
2. **Análisis de tópicos**:
   * Usaremos **LDA con Gensim** para extraer tópicos recurrentes (sin cambios aquí, ya que no mencionaste alternativas para tópicos).
   * **Bonus**: Visualizaremos un ranking de tópicos por equipo con Matplotlib/Seaborn.
3. **Impacto en la prueba técnica**:
   * Comparar tres métodos de análisis de sentimiento demuestra un enfoque analítico profundo y versatilidad, lo cual será bien valorado.
   * Documentaremos en el README las fortalezas y limitaciones de cada método, justificando su inclusión.

Supabase: kjfkmdklfmgslñmñlsmf´ls6464654sdkksdf

Supabase url :

NEXT\_PUBLIC\_SUPABASE\_URL=https://icxgavjswwlwnshvnhga.supabase.co

NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImljeGdhdmpzd3dsd25zaHZuaGdhIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NDY2MjgwNjIsImV4cCI6MjA2MjIwNDA2Mn0.jcqUd8YVfVdaQtg1w0apdLSjx3MReJNjnk4HkutONlM