Web para buscar partes de los Simpsons por medio de frases. POr ejemplo ingreso: Ahora va a ser una sorpresa la plancha y con IA realiza esta búsqueda y devuelve un link de youtube o otra web donde ver ese capitulo o esa parte del capitulo

1) Procesamiento del texto y búsqueda de frases:

Utilizar modelos de lenguaje como OpenAI GPT, BERT, o vector embeddings para buscar coincidencias de frases o similares en una base de datos.

2) Base de datos de episodios/escenas:

Necesitarías tener acceso a una base de datos que relacione frases con capítulos o partes específicas de los episodios.

Esto puede hacerse con:

Transcripciones completas de los capítulos.

Metadatos de videos.

Sistemas de indexación y búsqueda.

3) Enlace a contenido (YouTube u otra plataforma):

Utilizar la API de YouTube para buscar videos relacionados con las frases.

Alternativamente, puedes almacenar clips cortos (dentro de límites legales) en tu servidor.

¿Cómo hacerlo?

1) Crear una base de datos de frases y capítulos

Buscar transcripciones de los capítulos de Los Simpsons. Muchas webs tienen subtítulos o scripts públicos. Procesa estos scripts y relaciónalos con episodios y minutos específicos.

Guarda esta información en una base de datos como PostgreSQL o MongoDB.

2) Procesar la búsqueda con IA

Utiliza embeddings de texto (OpenAI API o Hugging Face) para buscar frases similares aunque no sean exactas.

Por ejemplo, si el usuario escribe: "Ahora será sorpresa la plancha", el sistema entiende que está buscando la frase original "Ahora va a ser una sorpresa la plancha".

3) Devolver enlaces de video

Con la API de YouTube, puedes buscar videos relacionados y filtrarlos por palabras clave y duración. <https://youtu.be/ID_DEL_VIDEO?t=SEGUNDOS>

*Web parecida: Frinkiac (*[*https://frinkiac.com/*](https://frinkiac.com/)*)*

¿Por qué es una buena idea?

Nicho de fanáticos: Los Simpsons tienen una comunidad gigantesca y una demanda constante de contenido.

Innovación: Combinar IA con un buscador de frases y videos aún no está bien explotado.

Monetización: Podrías monetizar con publicidad, afiliación a plataformas de streaming o incluso mediante suscripciones premium.

**Tecnologías sugeridas para el proyecto**

Backend: Go (Golang): Para la API y manejo de la lógica del servidor.

GORM: ORM para manejar la base de datos.

API de YouTube: Para buscar y generar enlaces a videos.

Base de datos: PostgreSQL o MongoDB: Para almacenar frases y metadatos.

Procesamiento de IA: OpenAI API: Para embeddings de texto y búsqueda de frases similares.

Hugging Face (alternativa open-source): Implementar modelos de búsqueda semántica.

Frontend: Vue.js o React: Para crear una interfaz intuitiva donde el usuario escriba frases y vea los resultados.

Despliegue:Docker para contenedores. Heroku o AWS para hosting.

**Arquitectura sugerida BackEnd. Clean Architecture.**

1. Backend RESTful API

Dividimos el backend en microservicios lógicos que interactúen mediante endpoints REST.

Entrada: La frase del usuario.

Procesos:

Búsqueda semántica en una base de datos de frases/episodios.

Generación del enlace al video correspondiente (YouTube API o un enlace prealmacenado).

Salida: Información del episodio (nombre, temporada, minuto) y enlace.

Instalar Dependencias Necesarias

Gin (Framework HTTP para crear la API):

go get -u github.com/gin-gonic/gin

GORM (ORM para manejar la base de datos):

go get -u gorm.io/gorm gorm.io/driver/postgres

DOTENV (Cargar variables de entorno, como claves o conexiones):

go get github.com/joho/godotenv

-------------------------

Secret key Open IA: sk-proj-MHRqAIcj4uDYslLLckgJH90ZMT-ED-DIwIryoFactWWxeK-nkb19J8WJAF\_Y6u42v0U2MlnqcAT3BlbkFJwzXozj8imWbUzz8BytN5oL0GRgxtDgknUPh49IerpJYOGIO7VdgefCUfttBjJ5KcsK0qeJQyAA

**Para revisar la base de datos usar PgAdmin4 (PosgreSQL).**

**EndPoints disponibles:**

1) GET /api/search - para buscar episodios

2) POST /api/episodes - para crear nuevos episodios

3) POST /api/update-embeddings - para actualizar los embeddings manualmente cuando sea necesario

**Correr proyecto**:

1) go run main.go

2) Luego por postman le pego a los metodos.

Method: get

URL: <http://localhost:8080/api/search?phrase=Ahora%20va%20a%20ser%20una%20sorpresa%20la%20plancha>

3) La idea de este get es obtener informacion del registro guardado en base de dato, por una frase similiar a la guardada.

4) Para probar la web: [http://localhost:8080](http://localhost:8080/)

**Casos de prueba buscando frases:**

1) <http://localhost:8080/api/search?phrase=no> pueden demostrarlo --> [

{

"id": 2,

"created\_at": "2024-12-17T20:21:22.469197-03:00",

"phrase": "Yo no fui, nadie me vio, no pueden demostrarlo",

"episode": "S06E01",

"title": "Bart of Darkness",

"timestamp": "08:21",

"youtube\_url": "<https://youtu.be/dummy_video2?t=501>"

}

]

2) <http://localhost:8080/api/search?phrase=sorpresa> la plancha -->

[

{

"id": 1,

"created\_at": "2024-12-17T20:21:22.469197-03:00",

"phrase": "Ahora va a ser una sorpresa la plancha",

"episode": "S05E02",

"title": "Cape Feare",

"timestamp": "12:34",

"youtube\_url": "<https://youtu.be/dummy_video?t=754>"

}

]

3) <http://localhost:8080/api/search?phrase=no> fui yo -->

[

{

"id": 2,

"created\_at": "2024-12-17T20:21:22.469197-03:00",

"phrase": "Yo no fui, nadie me vio, no pueden demostrarlo",

"episode": "S06E01",

"title": "Bart of Darkness",

"timestamp": "08:21",

"youtube\_url": "<https://youtu.be/dummy_video2?t=501>"

}

]

Analisis hasta el momento:

Mediante el post, se lee la base de datos y si hay registros con embedding=null, se actualizan con el codigo de embedding. Luego con el get se busca la frase que tiene un parecido y se devuelve el registro. Logica de grabar, convertir el embedding y leer, funciona OK.

10 casos creados.

**Proximos pasos:**

1) Subirla a algun hosting.

2) A la web ponerle un nombre tipo: simonalacacalisa, diantres, kawabonga, o alguna otra frase que no tenga web, ver con chatgpt. Sino algun nombre rancio como: sini, simpsi, o algo asi.

3) Mejorar los detalles de capitulos, texto informativo, etc.

4) Buscar pagina con frases de los simpson y buscar alrededor de 50 frases como carga inicial para probar la pagina.

5) Agregar FAQ y poner algo mio si lo veo conveniente.

6) Subirlo a github con las tecnologias usadas.