<!--Desarrollo de MarathonBot-->

Trabajo practico integrador {

<Por="Giuliana Bressi Vagliviello, Justo Perez, Martina Roldan, Federico Solanes"/>

}

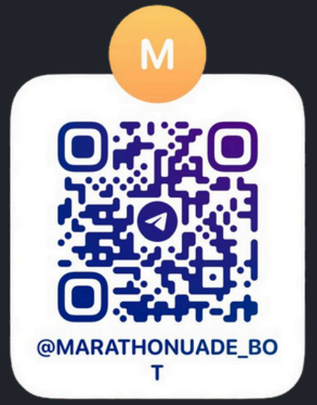
Introducción {

<--Problemática-->

Creamos el MarathonBot para resolver una duda que cualquier fan de las series/peliculas alguna vez tuvo: ¿Cuánto dura realmente un maratón de series o películas?. El tiempo oficial de pantalla no refleja la realidad, ignora las pausas necesarias para ir al baño, buscar comida o simplemente estirar las piernas.

<--Solución-->

Nuestro Bot de Telegram
para la planificación realista
de maratones de series y películas.



}

Nuestra Solución (El MVP):

Del dato bruto al plan ejecutable.

< Conexión a API en tiempo real: Utiliza la API de TMDB (The Movie Database) para buscar títulos de películas o series y obtener sus duraciones al instante.

< Cálculo Realista: Suma automática de pausas, aplicando una regla de negocio de 10 minutos por episodio/película (self.pausa_por_unidad = 10).

< Visualización: Genera un Gráfico de Pastel (torta) con matplotlib que compara el tiempo viendo vs. el tiempo de descanso.

Arquitectura Técnica (POO) {

bot.py "El jefe" ---> Es el punto de entrada que recibe comandos de Telegram, instancia las otras clases y ensambla la respuesta final.

config.py "Las llaves" ---> se encarga de la gestión de claves secretas y la configuración centralizada del proyecto.

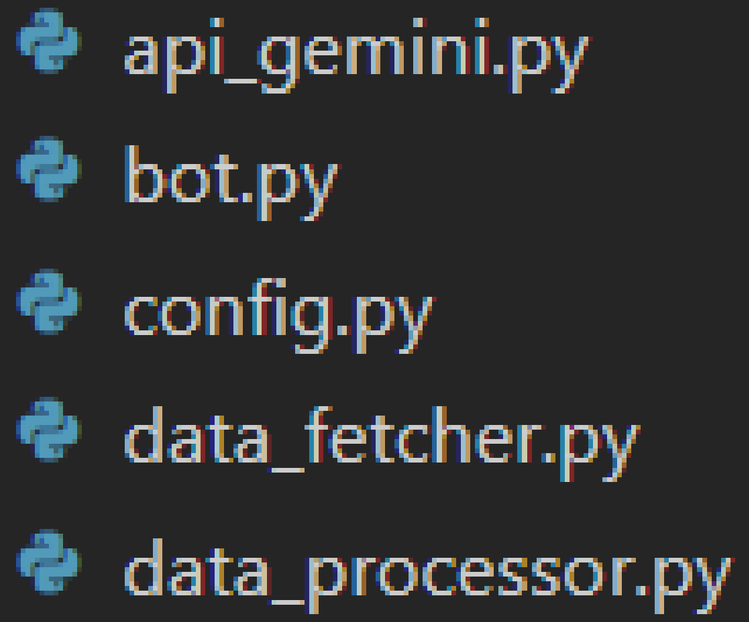
DataFetcher.py "El investigador" ---> Se conecta a la API de TMDB para buscar el título y obtener sus detalles técnicos (duración y cantidad de episodios)

DataProcessor.py "El analista" ---> Realiza el Cálculo Realista de tiempo y usa Matplotlib para generar el gráfico de torta (grafico_temporal.png)

api_gemini.py "El consultor de IA" ---> Este es el archivo que conecta con Google Gemini.

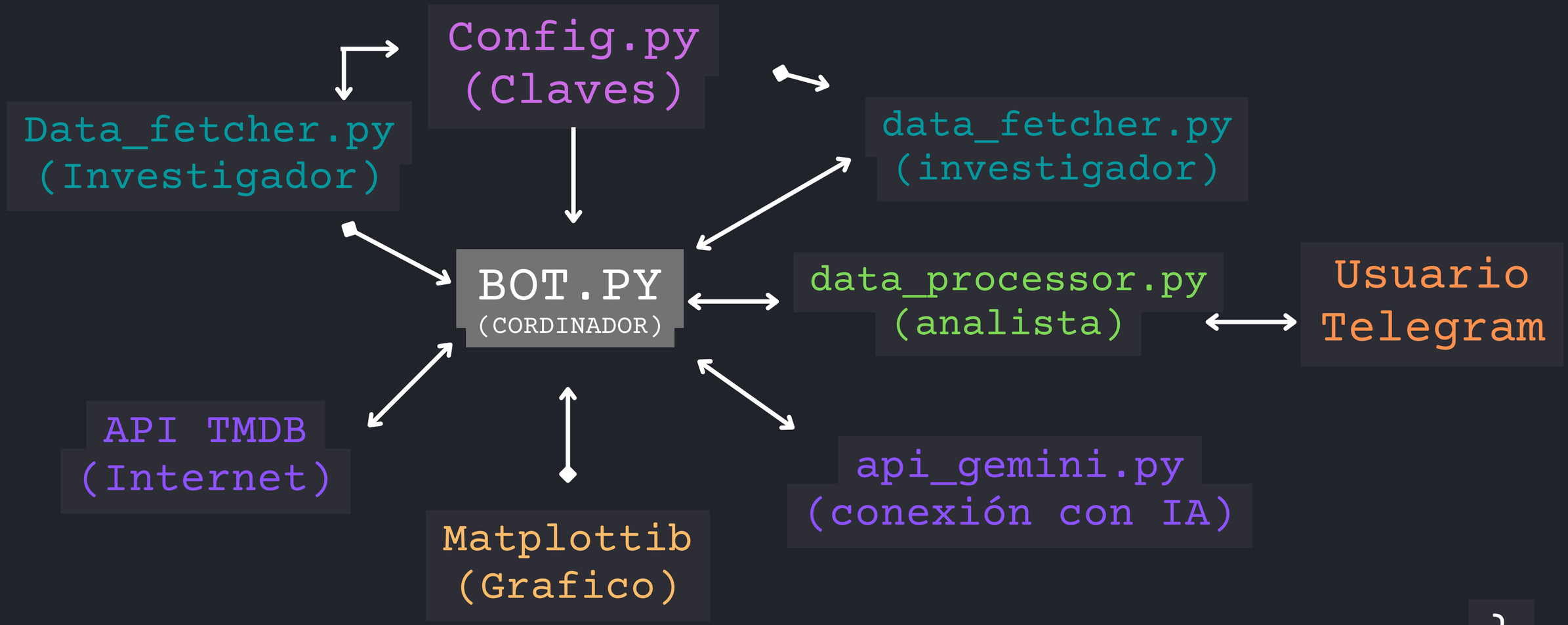
README.md "El manual" ---> Este es el archivo que le permite al usuario leer las instrucciones de uso y funcionamiento.

Requirements.txt "La librería" ---> Este es el archivo que lista las librerías externas (dependencias) para que el proyecto funcione.



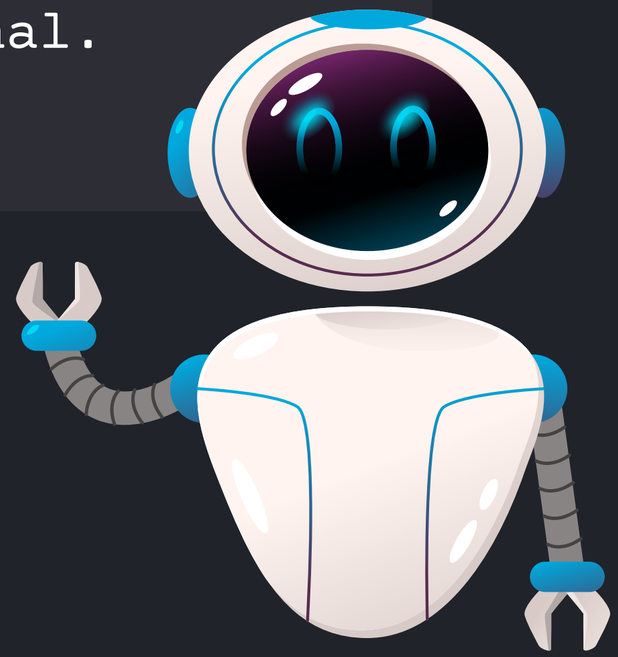
}

Estructura POO {



}

El Usuario inicia el proceso en el Bot; el Bot delega la búsqueda al DataFetcher; el DataFetcher trae los datos y el Bot se los pasa al DataProcessor para el análisis. Finalmente, el Bot le da al Usuario la Respuesta Final.



Tecnologías Utilizadas {

Python:

Lenguaje principal de desarrollo y lógica del negocio.

Telegram API:

Utilizada a través de la librería python-telegram-bot para la interfaz de usuario y la gestión de comandos.

TMDB (The Movie Database):

Fuente de datos para buscar información técnica de películas y series.

Tecnologías Utilizadas {

Matplotlib:

Librería de Python utilizada para la Visualización de Datos, específicamente para generar el Gráfico de Torta.

Requests:

Librería auxiliar para realizar la conexión y las peticiones HTTP a la API de TMDB.

Gemini:

Herramienta de Inteligencia Artificial Generativa utilizada como soporte integral del proyecto.

}

Comandos Utilizados {

/sinopsis [título]:

Conecta con la API de TMDB para buscar el contenido y utiliza Google Gemini para generar una descripción narrativa.

/detalle [título]:

Recupera metadatos técnicos (duración, cantidad de episodios) y estructura una ficha técnica limpia mediante IA.

/planear [título]:

Calcula el tiempo real de maratón, genera un gráfico de torta visual y ofrece consejos de organización personalizados por IA

/ayuda:

Muestra una lista de todos los comandos disponibles.

}

Aspectos de mejora {

```
Probando buscador con 'Breaking Bad'...
Error buscando titulo: 401 Client Error: Unauthorized for url: https://api.themoviedb.org/3/search/multi?api_key=eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIyN
DRlZDkwOTM5MmYzMTZmOWI5NzI0Yjg4NDVkZmUzZiIsIm5iZiI6MTc2MzkxMjU3MC43NTQsInN1YiI6IjY5MjMyYjdhNTg0YmYyMzc5NmEzYjQxOSIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2
ZXJzaW9uIjoxfQ.7P71GXKUxilL2SCmNgkQLF8EpF4cdC43eN3l3CWrJSM&query=Breaking+Bad&language=es-ES
No se encontró nada.
```

- Problema (Error 401): Al probar por primera vez el data_fetcher.py, recibimos un 401 Client Error: Unauthorized
- Causa: Se había utilizado inicialmente el código de API convencional en lugar del token de autenticación correcto.
- Solución: Se identificó la necesidad de usar la API Key (v3 auth) para la autenticación correcta y se reemplazó en el archivo de configuración (config.py).

Aprendizaje:

- Importancia de la gestión segura de claves y el uso del módulo config.py para separar las credenciales del código principal.
- Lectura atenta de la documentación técnica de la API para diferenciar los tipos de token de autenticación.

}

Aspectos de mejora {

```
from data_processor import DataProcessor
from ai_client import AiClient # Importamos el nuevo módulo de IA    Import "ai_client" could not be resolved
```

- Problema (import "ai_client" could not be resolved): archivo bot.py está buscando un archivo llamado ai_client.py pero no lo encuentra en la carpeta.
- Causa: Faltó crear el archivo nuevo para la Inteligencia Artificial.
- Solución: Se creó el archivo ai_client.py, que se encargue de OpenAI en la clase AiClient y se verificó la instalación de la librería openai. Esto reforzó la importancia de la consistencia en la arquitectura modular.

Aprendizaje:

- Entendimos que escalar el proyecto implica gestionar nuevas dependencias. Al integrar IA, no solo tuvimos que escribir la lógica, sino preparar el entorno de ejecución con las librerías adecuadas.

}

ai_client.py

```python
import google.generativeai as genai
from config import GEMINI_API_KEY

class AIClient:
    """
    Clase dedicada exclusivamente a la interacción con la Inteligencia Artificial (Gemini).
    """
    def __init__(self):
        # Configuramos la API Key una sola vez al iniciar
        genai.configure(api_key=GEMINI_API_KEY)
        # Usamos el modelo flash que es rápido para respuestas de chat
        self.model = genai.GenerativeModel('gemini-1.5-flash')

    def analizar_obra(self, titulo):
        """
        Genera un análisis de tono, género y temas clave para una película o serie.
        """
        try:
            prompt = (
                f"Actúa como un crítico de cine experto. Analiza brevemente el tono, "
                f"género y los temas clave de la obra '{titulo}'. "
                f"Responde en un solo párrafo conciso en español."
            )
            response = self.model.generate_content(prompt)
            return response.text
        except Exception as e:
            return f"Error al consultar a la IA: {str(e)}"
```

}

api_gemini.py

```python
import google.generativeai as genai
import config

genai.configure(api_key=config.GEMINI_API_KEY)

def crear_modelo(instrucciones):
    model = genai.GenerativeModel(model_name='gemini-2.5-flash', system_instruction=instrucciones)
    return model

def preguntar_gemini(pregunta, instrucciones=None):
    model = crear_modelo(instrucciones)
    response = model.generate_content(pregunta)
    return response.text

```

Reemplazamos código y renombramos archivo:

- Optimizamos el código del archivo ai_client.py y lo reemplazamos por api_gemini.py.

}

Resultados {

Comando

Muestra la captura de pantalla del chat con el comando /planear Breaking Bad (o el ejemplo que usaste)..

Reporte de texto

Explica que el bot genera un resumen con:

- Datos técnicos (Episodios, Duración unitaria).
- El TIEMPO TOTAL REAL en horas, resultado del cálculo realista.

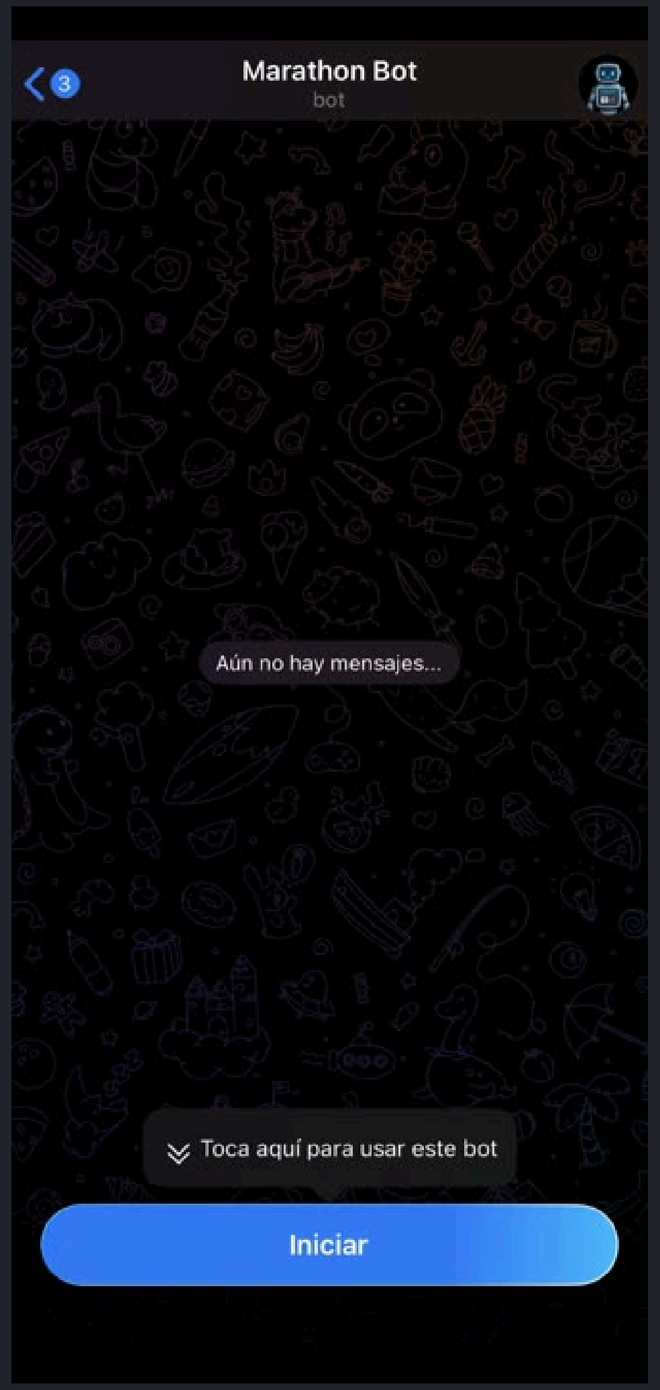
Gráfico de Torta

Muestra la captura del gráfico (grafico_temporal.png)

- Explicación: Este gráfico compara visualmente el tiempo de 'Tiempo Viendo' (en azul) contra el 'Tiempo de Pausa' (en gris). Es la visualización clave para la toma de decisiones.

}

Veamos como funciona {



MarathonBot {

}

<Próximos Pasos y Conclusión>

Futuro

Análisis Avanzado: Integración de la librería Gemini para un posible análisis de sentimientos o resumen de la sinopsis.

Mayor Flexibilidad: Permitir al usuario definir su propio tiempo de pausa (más de 10 minutos) o incluir datasets locales.

<!--MarathonBot-->

Gracias {

<Por="Giuliana Bressi Vagliviello, Justo Perez, Martina Roldan, Federico Solanes"/>

}