Título del Proyecto: MarathonBot

Asignatura: Taller de Programación II

Integrantes: Justo Pérez Viceconte, Giuliana Bressi Vagliviello, Federico Solanes, Martina Roldán.

Fecha: 27 de Noviembre 2025

1. Resumen

El presente trabajo detalla el desarrollo de un Producto Viable Mínimo (MVP) de un chatbot para Telegram. El objetivo es solucionar la ineficiencia en la planificación del entretenimiento mediante la automatización. El software, desarrollado en Python, consume datos de la API de TMDB y aplica lógica de negocio para transformar duraciones brutas en planes de tiempo realistas, incluyendo visualizaciones gráficas.

2. Objetivos

- **General:** Desarrollar un bot funcional que integre múltiples APIs y conceptos de Programación Orientada a Objetos.
- **Específicos:**
    - Consumir una API REST (TMDB) para obtener datos reales.
    - Generar valor agregado mediante el procesamiento de datos (cálculo de pausas y gráficos con Matplotlib).
    - Implementar un código modular y escalable.

3. Justificación de la Tecnología Seleccionada

- **Python:** Seleccionado por su robustez en el manejo de datos y la amplia disponibilidad de librerías como `requests` y `matplotlib`.
- **Telegram API:** Elegida por su accesibilidad gratuita, documentación clara y facilidad para crear interfaces de usuario conversacionales sin necesidad de desarrollo frontend complejo.
- **Arquitectura Modular (POO):** Se utilizó este paradigma para cumplir con los estándares de la cátedra, permitiendo que la lógica de búsqueda (`DataFetcher`) esté desacoplada de la lógica de presentación (`bot.py`), facilitando futuras expansiones (como la integración de IA).

A diferencia de los calculadores tradicionales, MarathonBot integra **Inteligencia Artificial Generativa (Google Gemini)** para ofrecer análisis cualitativos de las obras, y utiliza la API de **TMDB (The Movie Database)** para obtener datos cuantitativos precisos. El desarrollo se ha realizado íntegramente en Python, aplicando una arquitectura modular orientada a objetos que separa la lógica de negocio, la conexión con APIs externas y la interacción con el usuario.

4. Descripción de la Solución

Funcionalidades Implementadas (Comandos)

El bot responde a 6 comandos requeridos que cubren planificación, analítica y utilidad:

`/planear [título]`: Consulta TMDB, obtiene la duración (o calcula episodios × duración media) y devuelve el tiempo total de inversión necesario.

`/visualizar [título]`: Utiliza `matplotlib` para generar un gráfico de pastel que contrasta el tiempo de visualización vs. los descansos recomendados (regla de 15 min cada 2 horas).

`/ajustar [título] [días]`: Algoritmo iterativo que divide la duración total por la cantidad de días disponibles del usuario, devolviendo una cuota diaria de visualización.

`/analisis [título]`: **Integra la IA.** Envía el título a `ai_client.py`, el cual solicita a Gemini un resumen experto sobre el tono, género y temas de la obra, devolviendo un texto natural y humano.

`/valoracion [título]`: Muestra el rating promedio y la cantidad de votos de la comunidad global.

`/ayuda`: Lista interactiva de las capacidades del bot.

5. Dificultades Encontradas

El principal obstáculo técnico fue la correcta autenticación con servicios de terceros. Específicamente, la distinción entre los diferentes tipos de tokens que ofrece TMDB provocó errores de conexión iniciales (Error 401), los cuales fueron resueltos mediante la depuración del código y revisión de documentación técnica.

6. Conclusiones y Perspectivas Futuras

El MVP cumple con todos los requisitos funcionales establecidos. Se logró transformar datos crudos en información útil para la toma de decisiones del usuario. Como perspectiva futura, se planea integrar la API de OpenAI (ChatGPT) para analizar el "sentimiento" de las sinopsis y

ofrecer recomendaciones personalizadas, funcionalidad que ya está contemplada en la arquitectura actual pero pendiente de implementación en esta fase MVP.