Projeto Unificado - Fase 2 (Revisado): Análise de Engajamento de Mídias Globo com Orientação a Objetos 🚀

Objetivo Principal: Aplicando os princípios e técnicas da Programação Orientada a Objetos na base de código da Fase 1. Assim, o resultado será um sistema mais robusto, modular, extensível e com maior integridade dos dados.

Módulo Foco: DS-PY-02 (Programação Orientada a Objetos - Python)

1. Modelagem de Classes Centrais

1.1. Classe Plataforma

Representa uma plataforma onde o conteúdo é consumido ou a interação ocorre.

- **Atributos (privados com properties):**
  - __id_plataforma (int): Identificador único da plataforma (pode ser gerado internamente pela classe gerenciadora).
  - __nome_plataforma (str): Nome da plataforma (e.g., "Globoplay", "G1").
- **Construtor (__init__):**
  - Recebe nome_plataforma e, opcionalmente, id_plataforma.
  - Valida se o nome não está vazio.
- **Properties (@property e @*.setter):** Para acesso e validação dos atributos.
- **Métodos Mágicos:**
  - __str__(self): Retorna o nome da plataforma.
  - __repr__(self): Retorna uma representação como Plataforma(nome='...').
  - __eq__(self, other) e __hash__(self): Para permitir que objetos Plataforma sejam comparados e usados em coleções como sets ou chaves de dicionários (baseado no nome, por exemplo).

1.2. Classe Base Conteudo

Representa um item de conteúdo consumível.

- **Atributos (protegidos ou privados com properties):**
  - _id_conteudo (int).
  - _nome_conteudo (str).
  - _interacoes (list): Lista para armazenar objetos Interacao associados a este conteúdo.
- **Construtor (__init__):**
  - Recebe id_conteudo e nome_conteudo.
  - Inicializa _interacoes como uma lista vazia.
- **Properties:** Para id_conteudo e nome_conteudo.

- **Métodos:**
  - adicionar_interacao(self, interacao): Adiciona um objeto Interacao à lista _interacoes.
  - calcular_total_interacoes_engajamento(self): Calcula o total de 'like', 'share', 'comment'.
  - calcular_contagem_por_tipo_interacao(self): Retorna um dicionário com a contagem de cada tipo de interação.
  - calcular_tempo_total_consumo(self): Soma watch_duration_seconds das interações.
  - calcular_media_tempo_consumo(self): Calcula a média de watch_duration_seconds > 0.
  - listar_comentarios(self): Retorna uma lista dos textos dos comentários.
- **Métodos Mágicos:** __str__(self) e __repr__(self).

1.3. Classes Derivadas de Conteudo (Herança e Polimorfismo)

- **Classe Video(Conteudo):**
  - **Atributos Adicionais:** __duracao_total_video_seg (int, privado).
  - **Construtor (__init__):** Chama super().__init__() e inicializa __duracao_total_video_seg.
  - **Property:** Para duracao_total_video_seg.
  - **Métodos Sobrescritos/Novos:**
    - calcular_percentual_medio_assistido(self): Calcula (tempo médio de consumo / duracao_total_video_seg) * 100. Retorna 0 se duracao_total_video_seg for 0.
- **Classe Podcast(Conteudo):**
  - **Atributos Adicionais:** __duracao_total_episodio_seg (int, privado, opcional para maior detalhe).
- **Classe Artigo(Conteudo):**
  - **Atributos Adicionais:** __tempo_leitura_estimado_seg (int, privado).

1.4. Classe Interacao

Representa uma única interação de um usuário com um conteúdo em uma plataforma.

- **Atributos (privados com properties):**
  - __interacao_id (int, opcional, gerado internamente).
  - __conteudo_associado (Conteudo): Referência ao objeto Conteudo relacionado.
  - __id_usuario (int).
  - __timestamp_interacao (datetime).

    - __plataforma_interacao (Plataforma): Referência ao objeto Plataforma onde ocorreu.
    - __tipo_interacao (str).
    - __watch_duration_seconds (int).
    - __comment_text (str).
- **Atributo de Classe:**
    - TIPOS_INTERACAO_VALIDOS = {'view_start', 'like', 'share', 'comment'} (ou outros definidos).
- **Construtor (__init__):**
    - Recebe os dados brutos (e.g., de uma linha do CSV), um objeto Conteudo e um objeto Plataforma.
    - Valida e atribui os valores:
        - Converte id_usuario para int.
        - Converte timestamp_interacao para datetime.
        - Valida tipo_interacao contra Interacao.TIPOS_INTERACAO_VALIDOS. Se inválido, pode definir um padrão ou levantar um ValueError.
        - Converte watch_duration_seconds para int (padrão 0, não negativo).
        - comment_text (strip, padrão string vazia).
- **Properties:** Para acesso e validação dos atributos.
- **Métodos Mágicos:** __str__ e __repr__.

1.5. Classe Usuario

Representa um usuário da plataforma.

- **Atributos (privados com properties):**
    - __id_usuario (int).
    - __interacoes_realizadas (list): Lista de objetos Interacao.
- **Construtor (__init__):** Recebe id_usuario.
- **Properties:** Para acesso aos atributos.
- **Métodos:**
    - registrar_interacao(self, interacao: Interacao): Adiciona à lista __interacoes_realizadas.
    - obter_interacoes_por_tipo(self, tipo_desejado: str) -> list: Filtra __interacoes_realizadas.
    - obter_conteudos_unicos_consumidos(self) -> set: Retorna um set de objetos Conteudo (ou seus IDs).
    - calcular_tempo_total_consumo_plataforma(self, plataforma: Plataforma) -> int: Calcula o tempo total de consumo para uma plataforma específica.
    - plataformas_mais_frequentes(self, top_n=3) -> list: Retorna as N plataformas mais utilizadas pelo usuário.

- **Métodos Mágicos:** __str__, __repr__.

2. Classe de Orquestração e Análise: SistemaAnaliseEngajamento

- **Atributos (privados):**
  - __plataformas_registradas: Dicionário mapeando nome_plataforma (str) para objetos Plataforma. Usado para o "CRUD" em memória.
  - __conteudos_registrados: Dicionário mapeando id_conteudo (int) para objetos Conteudo (ou suas subclasses).
  - __usuarios_registrados: Dicionário mapeando id_usuario (int) para objetos Usuario.
  - __proximo_id_plataforma (int): Para gerar IDs para novas plataformas.
- **Construtor (__init__):** Inicializa os dicionários e o contador de ID.
- **Métodos de Gerenciamento de Plataforma ("CRUD" em memória):**
  - cadastrar_plataforma(self, nome_plataforma: str) -> Plataforma: Cria e adiciona uma nova plataforma se não existir. Retorna o objeto Plataforma.
  - obter_plataforma(self, nome_plataforma: str) -> Plataforma: Retorna uma plataforma pelo nome. Se não existir, pode cadastrá-la.
  - listar_plataformas(self) -> list: Retorna uma lista de todas as plataformas cadastradas.
- **Métodos de Carga e Processamento:**
  - _carregar_interacoes_csv(self, caminho_arquivo: str) -> list: Carrega dados brutos do CSV.
  - processar_interacoes_do_csv(self, caminho_arquivo: str):
    - Chama _carregar_interacoes_csv.
    - Ao invés de gerar um dicionario com os dados brutos, será adaptado para armazenar os dados como objetos da classe adequada:
      - Obtém/Cria o objeto Plataforma.
      - Obtém/Cria o objeto Conteudo.
      - Obtém/Cria o objeto Usuario.
      - Tenta instanciar Interacao, lidando com ValueError para validações.
      - Se Interacao válida, registra-a nos objetos Conteudo e Usuario.
- **Métodos de Análise e Relatório:**
  - gerar_relatorio_engajamento_conteudos(self, top_n: int = None): Itera por __conteudos_registrados, usa os métodos de cada objeto Conteudo para calcular métricas e as exibe.
  - gerar_relatorio_atividade_usuarios(self, top_n: int = None): Similar, para usuários.
  - identificar_top_conteudos(self, metrica: str, n: int): (e.g., metrica='tempo_total_consumo').

- **Atributo de Classe/Método de Classe (opcional):**
  - SistemaAnaliseEngajamento.VERSAO_ANALISE = "2.0"

3. Estrutura de Módulos e Pacotes

O projeto deve ser organizado com estrutura de módulos como aprendido ao longo do módulo

- **projeto_engajamento_fase2/** (Pacote principal)
  - __init__.py
  - **entidades/** (Sub-pacote)
    - __init__.py
    - plataforma.py (Classe Plataforma)
    - conteudo.py (Classes Conteudo, Video, Podcast, Artigo)
    - interacao.py (Classe Interacao)
    - usuario.py (Classe Usuario)
  - **analise/** (Sub-pacote)
    - __init__.py
    - sistema.py (Classe SistemaAnaliseEngajamento)
  - main.py (Script principal)
  - interacoes_globo.csv (Arquivo de dados)

4. Enunciado Detalhado para os Alunos (Resumo)

1. **Desenvolver Classes de Entidade:**
   - Implementar Plataforma, Conteudo (e suas subclasses Video, Podcast, Artigo), Interacao, e Usuario.
   - Focar em construtores robustos com validação de dados, properties para acesso controlado e métodos mágicos (__str__, __repr__).
   - A classe Interacao deve validar o tipo_interacao contra uma lista de tipos permitidos.
2. **Implementar SistemaAnaliseEngajamento:**
   - Gerenciar coleções de objetos Plataforma, Conteudo, e Usuario.
   - Implementar o carregamento de dados do CSV, transformando-os nos objetos definidos.
   - Realizar a vinculação entre os objetos (e.g., uma interação se refere a um conteúdo e uma plataforma específicos).
   - Desenvolver métodos para gerar relatórios de métricas, delegando os cálculos para os métodos das classes de entidade.
3. **Organização e Execução:**
   - Estruturar o código em pacotes e módulos conforme sugerido.
   - O main.py deve instanciar SistemaAnaliseEngajamento, carregar os dados e apresentar as métricas.

4. **Apresentação (único entregável além do código):**
    - Demonstrar o funcionamento do sistema.
    - Explicar as decisões de design OO, como as classes se relacionam, e como os princípios da POO foram aplicados.
    - Discutir desafios e aprendizados.