書籍管理API 開発課題

1. 課題概要

この課題では、書籍とその著者を管理するためのシンプルなREST APIを開発していただきます。
本番環境での運用を想定し、将来の機能拡張やチームでの開発を考慮した、メンテナンス性と拡張性の高い設計を期待しています。

- 開発言語: Python
- フレームワーク: FastAPIを推奨しますが、Flaskやその他フレームワークの利用も可能です。
- データベース: MySQLまたはPostgreSQLを使用してください。
- 開発環境: Dockerを使用してください。
- 開発期間: 約4~6時間を想定しています。
- 今後のフロー:

2. 要件

2.1. 機能要件

以下の2つのリソース（著者、書籍）を管理するAPIを実装してください。

リソース定義:

- 著者 **(Author)**
  - id: UUID (Primary Key)
  - 著者名: string (必須, 50文字以内)
- 書籍 **(Book)**
  - id: UUID (Primary Key)
  - タイトル: string (必須, 100文字以内)
  - 著者ID: UUID (著者のID, 外部キー)

APIエンドポイント:

必須/任意	**Method**	**Path**	説明
必須	POST	/authors	新しい著者を登録します。
必須	POST	/books	新しい書籍を登録します。登録時には、既存の著者IDを指定する必要があります。
必須	GET	/books	登録されている書籍の一覧を取得します。レスポンスには、各書籍に対応する著者名も含むようにしてくださ

			い。
任意	GET	/books/{book_id}	指定されたIDの書籍情報を取得します。レスポンスには、書籍に対応する著者名も含むようにしてください。(オプション要件)
任意	DELETE	/books/{book_id}	指定されたIDの書籍を削除します。(オプション要件)

2.2. 非機能要件

- 環境構築の容易性:
  - 他の開発者があなたのリポジトリをクローンした後、最小限のコマンドで開発環境をセットアップし、アプリケーションを実行できるようにしてください。
- データベース管理:
  - アプリケーションのバージョンアップに伴うデータベースのテーブル構造の変更(スキーママイグレーション)を、コードで管理できるようにしてください。
  - パフォーマンスを意識したテーブル設計(適切なインデックス設定など)を行ってください。
- 堅牢性:
  - リクエストに含まれるデータが不正な場合(例: 文字数オーバー、必須項目の欠如など)は、クライアントにその旨が伝わる適切なエラーレスポンス(HTTPステータスコード 4xx系)を返してください。
  - 存在しないリソースへのアクセスがあった場合も同様に、適切なエラーレスポンス(HTTPステータスコード 404など)を返してください。
- 設計:
  - 全体を通して、関心事の分離(Separation of Concerns)を意識した、見通しが良く変更に強いコード構造を目指してください。

3. 提出物

1. ソースコード一式を格納した**Git**リポジトリの**URL**
   - GitHub、GitLabなど、サービスの指定はありません。
   - ※難しければzip化したファイルでも可能
2. **README.md** ファイル
   - リポジトリ(フォルダ)のルートに README.md を配置し、以下の内容を必ず記載してください。
     - 開発環境のセットアップ手順(起動コマンドなど)
     - 実装したAPIエンドポイントの一覧と、curlなどを用いた簡単な実行例
     - アーキテクチャ等で意識した点

4. 評価の観点（ご参考）

この課題では、単に要件を満たすコードを書くだけでなく、以下のような点を総合的に評価します。

- ソフトウェアアーキテクチャの設計能力
- コードの可読性、保守性
- 適切なツールやライブラリの選定能力
- 問題解決能力
- ドキュメンテーション能力