Global AI Hub Python 202 Bootcamp Proje Dosyası

Genel Bakış

Bu proje, **Python 202 Bootcamp**'inde öğreneceğiniz üç temel konuyu (**OOP**, **Harici API Kullanımı**, **Kendi API'nizi FastAPI ile Yazma**) birleştirerek somut bir ürün ortaya çıkarmanızı amaçlamaktadır. Proje üç aşamadan oluşur ve her aşama bir önceki üzerine inşa edilir.

Amacımız, basit bir komut satırı uygulamasından başlayarak, onu harici verilerle zenginleştirmek ve son olarak bir web servisi haline getirmektir.

Temel Zorunlu Gereksinimler

1. Tüm proje kodları **public** bir **GitHub** repositorisinde bulunmalıdır.
2. Reponuzda, projenin nasıl kurulacağını ve çalıştırılacağını açıklayan detaylı bir README.md dosyası bulunmalıdır.
3. Kodunuzun, Python dilinde yazılmış, *.py uzantılı dosyalarda (veya tek bir dosyada) olması zorunludur.

Aşama 1: OOP ile Terminalde Çalışan Kütüphane

Amaç

Nesne Yönelimli Programlama (OOP) prensiplerini kullanarak modüler ve yönetilebilir bir konsol uygulaması oluşturmak.

Yapılacaklar Listesi

1. **Book Sınıfı Oluşturma:**
   - Her bir kitabı temsil edecek bir Book sınıfı oluşturun.
   - Bu sınıf title, author, ve ISBN (benzersiz kimlik olarak) niteliklerine (attributes) sahip olmalıdır.
   - __str__ metodunu override ederek kitabın bilgilerini okunaklı bir şekilde ("Ulysses by James Joyce (ISBN: 978-0199535675)" gibi) yazdırın.
2. **Library Sınıfı Oluşturma:**
   - Tüm kütüphane operasyonlarını yönetecek bir Library sınıfı oluşturun.
   - Bu sınıfın başlangıçta (__init__) kitapların tutulacağı bir listesi olmalıdır. Ayrıca verilerin saklanacağı dosyanın adını (library.json) almalıdır. **Not:** Txt değil, **JSON** kullanmanız beklenmektedir.
   - **Örnek Metodlar:**
     - add_book(book): Yeni bir Book nesnesini kütüphaneye ekler ve dosyayı günceller.

        - remove_book(isbn): ISBN numarasına göre bir kitabı kütüphaneden siler ve dosyayı günceller.
        - list_books(): Kütüphanedeki tüm kitapları listeler.
        - find_book(isbn): ISBN ile belirli bir kitabı bulur ve Book nesnesini döndürür.
        - load_books(): Uygulama başladığında library.json dosyasından kitapları yükler.
        - save_books(): Kütüphanede bir değişiklik olduğunda (ekleme/silme) tüm kitap listesini library.json dosyasına yazar.
3. **Ana Uygulama Döngüsü (main.py):**
    - Kullanıcıya bir menü sunan bir döngü oluşturun. Örnek bir menü yapısı:
        1. Kitap Ekle
        2. Kitap Sil
        3. Kitapları Listele
        4. Kitap Ara
        5. Çıkış
    - Kullanıcının seçimine göre ilgili Library metodunu çağırın.

Minimum Kabul Kriterleri

- Program komut satırından hatasız bir şekilde çalışmalıdır.
- Kitap ekleme, silme ve listeleme işlevleri beklendiği gibi çalışmalıdır.
- Uygulama kapatılıp açıldığında eklenen kitaplar kaybolmamalıdır (veriler library.json dosyasında kalıcı olmalıdır).
- Kod, Book ve Library sınıfları kullanılarak OOP prensiplerine uygun şekilde yapılandırılmış olmalıdır.
- **Pytest** ile metodların doğru çalıştığının testleri için ilgili python script(ler)i yazılmış olmalıdır.

Aşama 2: Harici API ile Veri Zenginleştirme

Amaç

Mevcut uygulamayı, bir harici API'ye bağlanarak daha akıllı ve kullanışlı hale getirmek. Kitap bilgilerini manuel girmek yerine, ISBN ile otomatik olarak çekmek.

Kullanılacak API: Open Library Books API (ücretsiz ve anahtar gerektirmez).

Teknik Gereksinimler

- httpx kütüphanesi.
- JSON verisini anlama ve işleme.
- Hata yönetimi (try-except).

Yapılacaklar Listesi

1. **httpx Kütüphanesini Projeye Ekleme:**
   - pip ile httpx kütüphanesini kurun.
   - Projeniz için bir requirements.txt dosyası oluşturup httpx'i içine ekleyin.
2. **Kitap Ekleme İşlevini Güncelleme:**
   - Kullanıcıdan artık tüm kitap bilgilerini (başlık, yazar) istemeyin. Sadece **ISBN** numarasını isteyin.
   - Library sınıfındaki add_book metodunu, ISBN numarasını alacak şekilde değiştirin.
   - Metodun içinde, Open Library API'sine şu formatta bir GET isteği gönderin: https://openlibrary.org/isbn/{isbn}.json
3. **API Cevabını İşleme:**
   - Gelen JSON cevabını parse edin.
   - Cevabın içinden title ve authors (yazar isimleri) gibi gerekli bilgileri ayıklayın.
   - Bu bilgilerle yeni bir Book nesnesi oluşturun ve kütüphaneye ekleyin.
4. **Hata Yönetimi:**
   - API isteği başarısız olursa (örn: internet yok) veya verilen ISBN ile kitap bulunamazsa (API 404 dönerse) programın çökmediğinden emin olun. Kullanıcıya "Kitap bulunamadı." gibi anlamlı bir mesaj gösterin.
5. **Pytest Kullanımı:**
   - pytest kütüphanenizi projeye dahil ederek, bu adımda eklediğiniz işlevlerin test(ler)i için ayrı script(ler) oluşturun.

Minimum Kabul Kriterleri

- Aşama 1'in tüm işlevleri çalışmaya devam etmelidir.
- Kullanıcı bir ISBN girdiğinde, program Open Library API'sinden kitap başlığını ve yazarını başarıyla çekmelidir.
- Geçersiz veya bulunamayan bir ISBN girildiğinde program çökmeyip kullanıcıyı bilgilendirmelidir.
- requirements.txt dosyası oluşturulmuş ve gerekli kütüphaneleri içermelidir.
- **Pytest** ile metodların doğru çalıştığının testleri için ilgili python script(ler)i yazılmış olmalıdır.

Aşama 3: FastAPI ile Kendi API'nizi Oluşturma

Amaç

Kütüphane uygulamanızın mantığını bir web servisi haline getirerek, verilerinize bir web arayüzü (API) üzerinden erişim sağlamak.

Öneri: FastAPI, modern yapısı ve otomatik dokümantasyon özellikleri nedeniyle bu görev için zorunludur.

Yapılacaklar Listesi

1. **FastAPI Kurulumu:**
   - Gerekli paketleri (fastapi, uvicorn) kurun ve requirements.txt dosyanızı güncelleyin.
   - Yeni bir api.py dosyası oluşturun.
2. **API Endpoint'leri Oluşturma:**
   - Aşama 1'de yazdığınız Library sınıfının mantığını API endpoint'lerinizde kullanın.
   - Aşağıdaki endpoint'leri (path operations) oluşturun:
     - GET /books: Kütüphanedeki tüm kitapların listesini JSON olarak döndürür.
     - POST /books: Request body'sinde bir ISBN alır ({"isbn": "978-0321765723"} gibi), Open Library'den verileri çeker (Aşama 2'deki mantık) ve kitabı kütüphaneye ekler. Başarılı olursa eklenen kitabı, başarısız olursa hata mesajını döndürür.
     - DELETE /books/{isbn}: Belirtilen ISBN'e sahip kitabı kütüphaneden siler.
3. **Veri Modelleri (Pydantic):**
   - API'nizin döndüreceği Book verisini ve POST isteğinde alacağı ISBN verisini modellemek için Pydantic BaseModel'lerini kullanın. Bu, kodunuzu daha güvenli ve dokümantasyonunuzu daha net hale getirir.
4. **API'yi Test Etme:**
   - Uygulamanızı doğru komutla (uvicorn api:app --reload) başlatın.
   - Tarayıcınızda /docs adresine giderek FastAPI'nin oluşturduğu interaktif dokümantasyon üzerinden tüm endpoint'lerinizi test edin.
   - pytest kütüphanesini kullanarak API testleriniz için ayrı script(ler) oluşturun.

Minimum Kabul Kriterleri

- API sunucusu hatasız bir şekilde başlamalıdır.
- /docs adresindeki interaktif arayüz erişilebilir olmalıdır.
- GET /books, POST /books, ve DELETE /books/{isbn} endpoint'lerinin tamamı beklendiği gibi çalışmalıdır.
- POST endpoint'i, Aşama 2'deki API çağırma mantığını doğru bir şekilde

tetiklemelidir.
- Tüm bağımlılıklar requirements.txt dosyasında listelenmelidir.

Proje Teslimi ve Değerlendirme

Projeniz değerlendirilirken aşağıdaki maddeler dikkate alınır:

1. **GitHub Reposu:**
   - Proje, public bir GitHub reposuna yüklenmelidir.
   - **Bonus:** Commit geçmişiniz, projenin aşamalarını yansıtmalıdır (örn: "Aşama 1 tamamlandı", "API entegrasyonu eklendi" gibi anlamlı commit mesajları).
2. **README.md Dosyası:**
   - **Proje Başlığı ve Açıklaması:** Projenin ne yaptığını kısaca özetleyin.
   - **Kurulum:**
     - Reponun nasıl klonlanacağı.
     - Bağımlılıkların nasıl kurulacağı (pip install -r requirements.txt).
   - **Kullanım (Usage):**
     - Aşama 1 ve 2'deki terminal uygulamasının nasıl çalıştırılacağı (python main.py).
     - Aşama 3'teki API sunucusunun nasıl başlatılacağı (uvicorn api:app --reload).
   - **API Dokümantasyonu (Aşama 3 için):**
     - Oluşturduğunuz endpoint'lerin bir listesi, ne işe yaradıkları ve örnek bir POST isteği için body yapısı. (Örn: POST /books - Body: {"isbn": "..."})
3. **Test Senaryoları (Tüm Aşamalar İçin)**

İleri Seviye Fikirler

Projeniz, Bootcamp'in ardından da reponuzda sizinle birlikte kalacak. Bu yüzden, bazı ek geliştirmelerle güzelleştirmek isteyebilirsiniz:

- Verileri JSON dosyası yerine **SQLite** gibi basit bir veritabanında saklayın.
- API'ye PUT metodu ile kitap güncelleme özelliği ekleyin.
- Basit bir **HTML/CSS/JavaScript** arayüzü hazırlayarak kendi API'nizi tüketen bir web sayfası oluşturun.
- Projenizi **Docker** ile container haline getirin.