Zadanie Rekrutacyjne

Zadanie:

Twoim zadaniem jest stworzenie systemu rezerwacji książek w bibliotece, składającego się z kilku komponentów:

1. **Główny Backend w Django** (Python 3) - odpowiedzialny za zarządzanie danymi książek, użytkowników, oraz rezerwacji.
2. **Serwis API w Flask** (Python 2) - który będzie udostępniał dane o statusie książek z zewnętrznych bibliotek oraz przyjmował informacje o dokonanych rezerwacjach.
3. **Baza danych MySQL** - do przechowywania danych o książkach, użytkownikach, oraz rezerwacjach.
4. **Redis** - do przechowywania sesji użytkowników, cache'owania wyników oraz kolejkowania zadań asynchronicznych.
5. **Integracja z zewnętrznymi serwisami** - aplikacja powinna komunikować się z zewnętrznymi serwisami przez REST API.

Wymagania:

1. **Główny Backend w Django:**
    - Stwórz aplikację Django z modelami `Book`, `User`, `Reservation`.
    - Implementuj widoki oraz API endpoints dla CRUD operacji na książkach, użytkownikach i rezerwacjach.
    - Zaimplementuj logikę rezerwacji książek, uwzględniając sprawdzanie dostępności w zewnętrznych serwisach za pomocą Flask API.
    - Dodaj autoryzację i uwierzytelnianie użytkowników przy użyciu tokenów JWT.
    - Stwórz zadania asynchroniczne (np. przy użyciu Celery) do odświeżania statusu książek oraz wysyłania powiadomień do użytkowników.
2. **Serwis API w Flask:**
    - Stwórz serwis, który będzie nasłuchiwał na określonym porcie i komunikował się z Django backendem.
    - Endpoint `/status/<book_id>` - powinien zwracać informacje o statusie książki w zewnętrznych bibliotekach.
    - Endpoint `/reserve` - powinien przyjmować informacje o rezerwacjach i synchronizować dane z główną bazą.
    - Dodaj obsługę logowania oraz prostą walidację danych wejściowych.
3. **Baza danych MySQL:**
    - Skonfiguruj schemat bazy danych w MySQL, który obsłuży dane o książkach, użytkownikach, rezerwacjach oraz logi.
    - Dodaj przykładowe dane do testowania.
4. **Redis:**

    - Skonfiguruj Redis do przechowywania sesji użytkowników oraz cache'owania wyników zapytań do zewnętrznych API.
    - Zaimplementuj kolejkowanie zadań asynchronicznych, np. do odświeżania statusu książek lub wysyłania powiadomień.
5. **Integracja z zewnętrznymi serwisami:**
    - Zaimplementuj logikę komunikacji z zewnętrznymi serwisami przez REST API. Symuluj odpowiedzi tych serwisów za pomocą mocków.
    - Zadbaj o odpowiednią obsługę błędów oraz retry w przypadku problemów z komunikacją.

Dodatkowe wymagania:

- Przygotuj dokumentację API przy użyciu Swaggera lub innego narzędzia.
- Stwórz kilka testów jednostkowych i integracyjnych, które sprawdzą poprawność działania kluczowych funkcji aplikacji.
- Zaimplementuj logowanie błędów i operacji, zarówno w Django, jak i Flask, oraz przechowuj je w MySQL.

Dostarczenie zadania:

1. Utwórz repozytorium na GitHub i umieść tam cały kod projektu.
2. Dołącz plik README z instrukcjami dotyczącymi uruchomienia aplikacji lokalnie (w tym wymagane zależności, konfiguracja środowiska, oraz uruchamianie zadań asynchronicznych).
3. Dołącz diagram architektury, który przedstawia komunikację między komponentami systemu.
4. Prześlij link do repozytorium wraz z opisem swojego podejścia do realizacji zadania.

Czas na wykonanie zadania:

Masz 10 dni na ukończenie zadania od momentu jego otrzymania.

Powodzenia! Czekamy na Twoje zgłoszenie i mamy nadzieję, że dołączysz do naszego zespołu!