Zaawansowany system rezerwacji zasobów

Twoim zadaniem jest stworzenie zaawansowanego systemu rezerwacji zasobów (np. sal konferencyjnych, sprzętu IT, czy pojazdów służbowych) dla średniej wielkości firmy. Aplikacja powinna obsługiwać różne typy zasobów, harmonogramy rezerwacji, ograniczenia czasowe i zaawansowane raportowanie. Projekt ma być stworzony z wykorzystaniem FastAPI lub Django, w zależności od Twojego doświadczenia i preferencji.

Wymagania funkcjonalne

1. **Zarządzanie zasobami:**
    - CRUD dla zasobów, z następującymi atrybutami:
        - Nazwa zasobu (String).
        - Typ zasobu (np. sala konferencyjna, pojazd, sprzęt IT).
        - Lokalizacja (String).
        - Dostępność (dni i godziny, np. "Poniedziałek-Piątek, 9:00-18:00").
    - Możliwość przypisywania ograniczeń specyficznych dla zasobów (np. minimalny czas rezerwacji, maksymalny czas użytkowania).
2. **Rezerwacje:**
    - Tworzenie rezerwacji z następującymi atrybutami:
        - ID użytkownika.
        - ID zasobu.
        - Data i godzina rozpoczęcia.
        - Data i godzina zakończenia.
    - Walidacja:
        - Rezerwacja nie może kolidować z istniejącą rezerwacją.
        - Rezerwacja musi być zgodna z dostępnością zasobu.
        - Rezerwacja powinna mieć minimalny i maksymalny czas trwania (np. min. 30 minut, maks. 8 godzin).
3. **Raportowanie i statystyki:**
    - Endpoint API zwracający statystyki, np.:
        - Liczba rezerwacji na dzień/tydzień/miesiąc.
        - Najczęściej rezerwowane zasoby.
        - Średni czas rezerwacji.
    - Eksport danych do pliku CSV lub JSON.
4. **Uwierzytelnianie i autoryzacja:**
    - Użytkownicy muszą się logować, aby korzystać z systemu (np. JWT w FastAPI lub Django Authentication).
    - Role użytkowników:
        - **Admin:** Pełny dostęp (zarządzanie użytkownikami, zasobami i rezerwacjami).
        - **Użytkownik:** Możliwość przeglądania zasobów i tworzenia własnych rezerwacji.
5. **Notyfikacje:**
    - Automatyczne powiadomienia e-mail dla użytkowników o zbliżającej się rezerwacji (np. 1 godzinę wcześniej).
    - Powiadomienia o anulowaniu lub modyfikacji rezerwacji.

Wymagania techniczne

1. **Backend:**
    - Framework: **FastAPI** lub **Django (Rest Framework)**.
    - Obsługa baz danych: **PostgreSQL** lub **SQLite**.
    - Użycie relacji w bazie danych:
        - Relacja użytkownik -> rezerwacje.
        - Relacja zasób -> rezerwacje.
2. **Testy:**
    - Testy jednostkowe i integracyjne:
        - Walidacja danych wejściowych.
        - Obsługa konfliktów rezerwacji.
        - Testy endpointów API.
    - Framework do testowania:
        - **FastAPI**: pytest + httpx.
        - **Django**: unittest + Django Test Client.
3. **API:**
    - Dokumentacja API:
        - **FastAPI**: automatyczna dokumentacja OpenAPI.
        - **Django**: integracja z **drf-spectacular** lub **Swagger**.
    - Wymagana paginacja wyników na endpointach zwracających listy (np. zasoby, rezerwacje).
4. **Obsługa błędów:**
    - Zwrot odpowiednich kodów HTTP (np. 400 dla błędnej walidacji, 404 dla nieistniejących zasobów).
    - Czytelne komunikaty o błędach.