Na wstępie uczciwie przyznam że samo API jest trywialne i mocno niedopracowane - niepoprawne wykorzystanie większości endpointów skutkuje błędem serwera (500); brak kodów odpowiedzi.

Włożyłem jednak dużo pracy w zaimplementowanie (głównie zrozumienie i rozwiązywanie problemów) dodatkowych funkcjonalności - mam nadzieję że wady i zalety nieco się zbilansują.

Dodatkowe elementy projektu:

1. **Zabezpieczenie endpointów:**
    a. **Wykorzystanie tokenów JWT**
    b. **Dostęp w oparciu o role** (@PreAuthorize("hasAnyAuthority('ADMIN')") itp.)
    c. W jednym przypadku dodatkowo wykorzystałem utworzony **PermissionEvaluator** (@PreAuthorize("hasPermission(#id, 'updateDefender')"))
2. **Token JWT przechowywany i przesyłany jest w formie Cookie** (wymagało to dodanie filtru pobierającego wartość ciasteczka i jego umieszczenie w nagłówku - modyfikacja przesyłanego żądania).
3. **HATEOAS** - niestety, z wykorzystaniem spring-boot-starter-hateoas, co zmusiło mnie do tworzenia RepresentationModelAssemblerów. Finalnie dodane linki umożliwiają jedynie sprawne nawigowanie po udostępnianych zasobach, jednak nie udało mi się wykorzystać ich w funkcjach POST/PUT (tak jak ma to miejsce przy wykorzystaniu @RepositoryRestResource; PagingAndSortingRepository dostępnych przez inne zależności) (Zamiast tego, klucze obce przekazywane są jako ID).
4. **Zastosowanie DTO**, które w kilku przypadkach ukrywają i dodają pola udostępnianych zasobów.