**Gradebook – E-dziennik**

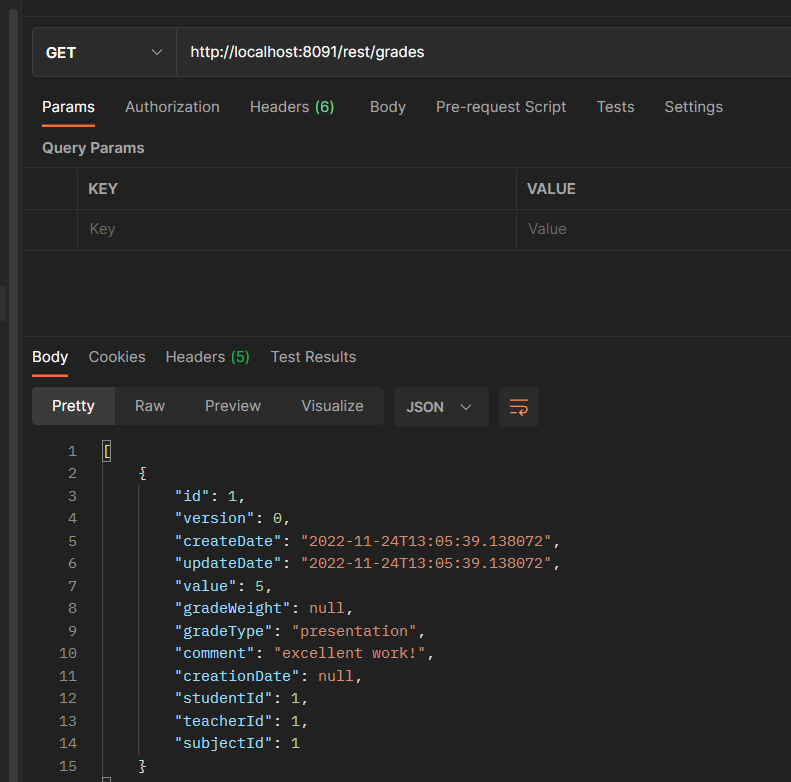
**Istnieje następujący opis dziedzinowy:**

Właściciel prestiżowej sieci szkół podstawowych (chce pozostać anonimowy) pragnie, aby stworzyć dla niego aplikację – E-dziennik. Aplikacja ta ma umożliwiać dodawanie, usuwanie, wyszukiwanie oraz edycję:  
- Nauczycieli  
- Przedmiotów prowadzonych w danych klasach  
- Klas  
- Uczniów  
- Ocen uczniów

**Nauczyciel** przypisany jest do wielu przedmiotów. **Przedmiot** w tej aplikacji rozumiemy jako dana dziedzina prowadzona w danej klasie, np. “Biologia\_3c”. **Klasa** to zbiór uczniów. Każdy przedmiot przypisany jest tylko do jednej klasy – ma to ułatwić poruszanie się po przedmiotach. **Uczniowie** przypisani są do klas, każdy może być przypisany do jednej klasy. **Ocena** to najważniejsza informacja w E-dzienniku i jest przypisana do przedmiotu, ucznia oraz nauczyciela wystawiającego ocenę oraz posiada wartość, wagę, typ (sprawdziań, praca klasowa, odpowiedź ustna, praca domowa itd…) komentarz oraz datę wystawienia.

Poza standardowym rozwiązaniem CRUD’owym aplikacja musi spełniać szereg wymagań biznesowych.  
Otrzymałeś szablon aplikacji z zaimplementową w większości kompletnie domeną “Teacher”. W katalogu “resources” znajduje się model bazy danych, który powinieneś wykorzystać. W tym samym katalogu znajduje się postman’owa kolekcja, którą możesz zaimportować do aplikacji POSTMAN, aby przetestować istniejące kontrollery do “teacher’a”.  
Jesteś odpowiedzialny za zaimplementowanie części backendowej do aplikacji. Twoje zadania to:

1. Przeczytaj dokładnie całość zadania. Jeśli coś będzie dla Ciebie niejasne, zapytaj o to na grupie teamsowej dodając @General, abyśmy zobaczyli powiadomienia.
   1. Jeśli coś nie jest sprecyzowane, to masz dowolność rozwiązania
2. Używając ctrl+shift+f (znajdź po tekście) wyszukaj “dev note” I przeczytaj ich zawartość
3. Używając znajdź po tekście wyszukaj “TODO IMPLEMENT” I postaraj się zaimplementować rzeczy opisane w notatkach. Jeśli którekolwiek sprawi ci trudności, wróć do nich później.
4. Zaimplementuj model encji na podstawie diagramu bazy danych
   1. Usuwanie jakichkolwiek danych nie powinno usuwać kaskadowo usuwać innych danych
   2. Wszystkie encje powinny rozszerzać AbstractEntity
   3. Zamodeluj relacje dwustronnie, pamiętając o mappedby
   4. Przydatne linki: <https://thorben-janssen.com/ultimate-guide-association-mappings-jpa-hibernate/> oraz <https://vladmihalcea.com/the-best-way-to-map-a-onetomany-association-with-jpa-and-hibernate/>
5. Wzorując się na klasach domeny “Teacher” stwórz klasy serwisowe, repozytoria, kontrollery restowe, transferobjecty (ETO) I mappery dla wszystkich encji
   1. Repozytoria niech rozszerzają JPARepository – dzięki temu nie będzie konieczne pisanie podstawowych metod
   2. ETO powinny, w przeciwieństwie do Encji, zawierać ID rodzica oraz ewentualnie listę ID swoich “dzieci”
   3. Zwróć uwagę, które relacje są konieczne do zapisu danej encji – nie można zapisać przedmiotu podania nauczyciela i klasy; nie można zapisać ucznia bez podania klasy, nie można zapisać oceny bez ucznia, nauczyciela i przedmiotu
6. Przygotuj oraz wyeksportuj (i na bieżąco aktualizuj) kolekcję postmanową z operacjami CRUDowymi
7. Napisz Testy jednostkowe (a właściwie integracyjne) dla każdego **serwisu**, testujące wymagane funkcjonalności – czyli zapis, edycję, usuwanie oraz odczyt
   1. Pamiętaj o czyszczeniu bazy danych przed I po każdej metodzie testowej
   2. Testy pisz w postaci given/when/then gdzie given to przygotowanie testu, when to pojedyncze wywołanie metody która będziesz testować, a then to jedna lub wiele asercji testowych
8. Implementując metody serwisowe dodaj poniższe funkcjonalności:
   1. Ocenę można wystawić z danego typu tylko raz danego dnia
   2. Ocena ma przyjmować wartości od 1 do 6
   3. Każda ocena musi mieć wagę od 0 do 9 z dwoma liczbami po przecinku, domyślna wartość to 1 – waga nie musi być podawana przy zapisie /aktualizacji w restApi
   4. Ocena o wartości 1 lub 6 musi mieć komentarz
   5. Typ Oceny ma być ENUMem
   6. Nazwa przedmiotu ma być złączeniem typu (np. BIOLOGY) oraz numeru I nazwy klasy (BIOLOGY\_7b). Zapewnij odpowiednią implementację przy zapisie I aktualizacji, nawet gdy nie nazwa przedmiotu nie jest podana.
9. Napisz testy integracyjne sprawdzające powyższe funkcjonalności
10. Napisz serwisy oraz kontrollery restowe które będą udostępniać poniższe funkcjonalności
    1. Podaj średnią ocenę ważoną danego ucznia z danego przedmiotu (nowy serwis oraz nowy restcontroller)
    2. Wszystkie funkcjonalności przetestuj integracyjne
11. Napisz kwerendy bazodanowe które będą umożliwiać:
    1. Wyszukanie ocen po: wartości od, wartości do, wadze od, wadze do, dacie wystawienia od, dacie wystawienia do, typie, przedmiocie (id), studencie (id)
       1. Wszystkie pola wyszukiwania są opcjonalne
       2. Użyj CriteriaApi aby to zaimplementować, przydatna będzie lista lub tablica predykatów
    2. Kwerenda, która wyszuka wszystkich uczniów, którzy dostali danego dnia ocenę o wartości 1
    3. Kwereda, która jako wynik zwróci liczbę osób, które danego dnia dostały ocenę X
12. Przetestuj powyższe kwerendy testami integracyjnymi
13. Do Kwerend z podpunktu 11 napisz serwisy oraz kontrollery restowe

TODO: resolve nulls, delete ByName method