1. Za pomocą Spring Initializera stworzyć aplikację z zależnością **Spring Web.**
2. Stworzyć model dla **Student** zawierającą id, imię, nazwisko, wiek.
3. Stworzyć klasę **StudentService**, stworzyć w niej listę 5 studentów oraz dodać metodę **getAllStudents** zwracająca wszystkich studentów.
4. Stworzyć klasę **StudentController** która działa na uri “**/api/student**” zwracająca wszystkich studentów.
5. Dodać do **StudentService** metody:
   1. addStudent - dodającą studenta,
   2. editStudent - uaktualniając **tylko** wiek i nazwisko
   3. deleteStudent - usuwająca studenta
   4. getStudent - zwracającego studenta o podanym id
6. Dodać do **StudentController** endpointy obsługujące wyżej wymienione metody. Wykorzystaj odpowiednie adnotacje do obsługi zapytań (np. @RequestParam, @RequestBody)
7. Dodaj model **Teacher** zawierający id, imię, nazwisko, unikatową nazwę przedmiotu : [*ALGEBRA, LAW, BIOLOGY*]
8. Rozszerz model **Student** o listę przedmiotów na które chodzi.
9. Dodaj kolejnych 5 studentów do listy
10. Stwórz serwis **TeacherService**, stworzyć w niej listę 3 nauczycieli
11. W **TeacherService** stworzyć metody :
    1. addTeacher - dodaje nauczyciela
    2. deleteTeacher - usuwa nauczyciela
    3. getTeacher - zwraca nauczyciela
    4. getTeacherClass - zwraca wszystkich studentów chodzących na jego zajęcia
    5. deleteStudentFromClassByTeacherId - usuwa podanego studenta z kursu podanego nauczyciela
12. Stworzyć **TeacherController** obsługujący powyższe metody
13. Napisać testy do obu kontrolerów
14. Skonfigurować OpenApi / Swagger i zapoznać się z nim
15. Zmienić nazwę i opis projektu w OpenApi / Swagger
16. Upakować zwracane obiekty z klas **StudentController** i **TeacherController** w obiekt ResponseEntity - zapoznać się z tym obiektem i wykorzystać tak, żeby zarówno poprawne żądanie było obsłużone, jak i takie które jest niepoprawne ( np. nieistniejące id, niepoprawne dane w zapytaniu ). W obu przypadkach dodać nowy header : “successful” : “true” / “false”
17. Stworzyć **AuthorizationStudentFilter** filtr który będzie obsługiwał tylko **URI** dla StudentControllera i który będzie sprawdzał czy w nagłówkach zapytania znajduje się nagłówek

“**role**” : “**STUDENT\_ROLE**” lub

“**role**” : “**TEACHER\_ROLE**”

Jeśli nie to należy zwrócić wiadomość błędu :

**{"errorMessage":"User unauthorized!"}**

ze statusem **401 Unauthorized**

1. Stworzyć kolejny filtr który będzie się wykonywał **przed** *AuthorizationStudentFilter***,** który będzie wyświetlał w logach serwera różnice czasu rozpoczęciem i zakończeniem zapytania. Ten filtr ma działać przy każdym zapytaniu.
2. Stworzyć filtr **AuthorizationTeacherFilter**, który będzie zabezpieczał **URI** dla kontrolera **TeacherController** bazując na nagłówku :

“**role**” : “**TEACHER\_ROLE**”

W przypadku niepowodzenia zwrócić :

**{"errorMessage":"User unauthorized!"}**

ze statusem **401 Unauthorized**

1. Poprawić testy
2. Zakomentować **AuthorizationStudentFilter** i **AuthorizationTeacherFilter**
3. Stworzyć **AuthorizationService** który posiada metodę **isAuthorized** przyjmującą jako parametry URI oraz nagłówek **role (** jeśli istnieje **)** zapytania.
   1. Dla URI **StudentControllera** metoda ma zwracać true jeśli wartość nagłówka role to :

“**STUDENT\_ROLE**” lub “**TEACHER\_ROLE**”

* 1. Dla URI **StudentControllera** metoda ma zwracać true jeśli wartość nagłówka role to : “**TEACHER\_ROLE**”

1. Stworzyć interceptor **AuthorizeInterceptor,** wstrzyknąć do niego **AuthorizationService** i za pomocą metody **isAuthorized** sprawdzić czy podane zapytanie może się wykonać, jeśli nie, zwrócić :

**{"errorMessage":"User unauthorized!"}**

ze statusem **401 Unauthorized**

1. Jeśli trzeba poprawić testy

Jeśli udało Ci się skończyć przed czasem zachęcam Cię do zapoznania się z materiałami, które uważam za przydatne :

* **Jak stworzyć REST API** : <https://www.youtube.com/watch?v=RNLje5-tFco>
* **Spring Boot Tutorial**: <https://www.youtube.com/watch?v=9SGDpanrc8U>

(zachęcamy do obejrzenia całego, ale szczególnie momentów : [23:19](https://www.youtube.com/watch?v=9SGDpanrc8U&t=1399s) API Layer [26:38](https://www.youtube.com/watch?v=9SGDpanrc8U&t=1598s) Business Layer [29:08](https://www.youtube.com/watch?v=9SGDpanrc8U&t=1748s) Dependency Injection )