1. Za pomocą Spring Initializera stworzyć nową aplikację z zależnością **Spring Web.**
2. Stworzyć dwa kontrolery :
   1. **UserController** ( /api/user ) z metodami:
      1. **getUser** - zwraca **tylko** wiadomość “This is user!”
      2. **updateUser** - zwraca **tylko** wiadomość “User updated!”
   2. **AdminController** ( /api/admin ) z metodami:
      1. **getAdmin** - zwraca **tylko** wiadomość “This is admin!”
      2. **createUser** - zwraca **tylko** wiadomość “User created!”
      3. **deleteUser** - zwraca **tylko** wiadomość “User deleted!”
3. Upewnić się, że działa
4. Dodać spring security do **pom.xml:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

1. w pliku **application.properties** nadpisać domyślnego użytkownika :

* username: user
* password: pass
* role : user

1. Zabezpieczyć :
   1. **UserController** tak, żeby używać go mógł tylko zalogowany użytkownik z rolami : “user” lub “admin”
   2. **AdminController** tak, żebym mógł go używać tylko zalogowany użytkownik z rolą “admin”

W tym momencie mając domyślnego użytkownika ( application.properties ) z rolą : “user” tylko UserController powinien być osiągalny.

1. Zaktualizować domyślnego użytkownika i zmienić mu rolę na “admin”
2. Przetestować czy oba kontrolery działają dla zalogowanego użytkownika z rolą **admin**
3. Nadpisać UserDetailsService żeby zwracał 2 użytkowników :
   1. username: **admin**; password: **admin**: rola: **admin**
   2. username: **user**; password: **user**: rola: **user**
4. Stworzyć i zastosować PasswordEncoder ( BCryptPasswordEncoder )
5. użyć passwordEncodera w tworzeniu użytkowników z punktu 33
6. Usunąć użytkownika z **application.properties**
7. Przetestować aplikację czy zachowuje się tak samo dla obu użytkowników.
8. Zmienić zabezpiecznia dla endpointów:
   1. **UserController** 
      1. **getUser** - (GET) dostępne dla wszystkich
      2. **updateUser** - (PUT) dostępne tylko dla roli user i admin
   2. **AdminController**
      1. **getAdmin** - (GET) dostępne tylko dla roli user i admin
      2. **createUser** - (POST) dostępne tylko dla roli admin
      3. **deleteUser** - (DELETE) dostępne tylko dla roli admin
9. Zmienić tworzenie użytkowników. Usunąć rolę i dodać uprawnienia :
   1. user : **USER\_READ**, **USER\_EDIT**
   2. admin: **ADMIN**
10. Dodać nowego użytkownika:
    1. username: **spectator**; password: **spectator**: uprawnienia: **USER\_READ**
11. Dostosować security do obsługi sprawdzania uprawnień ( nie roli )
    1. **UserController** 
       1. **getUser** - dostępne dla **USER\_READ i ADMIN**
       2. **updateUser** - dostępne dla **USER\_EDIT i ADMIN**
    2. **AdminController**
       1. **getAdmin** - dostępne dla **ADMIN**
       2. **createUser** - dostępne dla **ADMIN**
       3. **deleteUser** - dostępne dla **ADMIN**
12. Stworzyć osobny projekt i wykonać w nim : <https://spring.io/guides/gs/messaging-stomp-websocket/>
13. Zmodyfikować ten projekt dodając konfigurację **SchdeluerConfig** wraz z adnotacją @EnableScheduling. Następnie za pomocą **SimpleMessageTemplate** wysyłać automatycznie co 1 sekundę wiadomość z jednym losowym imieniem.
14. [**Zadanie dla chętnych - ta wiedza nie będzie później sprawdzana**] Jeżeli skończyłeś zachęcam Cię do zaimplementowania **JWT** w zadaniu z punktów **25) - 41)**.

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

# **Kurs Spring Boot #07 - Spring Security i JWT Token:** <https://www.youtube.com/watch?v=and2DR_N6tE>

# **Spring Security | FULL COURSE** : <https://www.youtube.com/watch?v=her_7pa0vrg>