# User-service documentation

1. Пишем регистрацию пользователя.
   1. Создаем entity Users
   2. В таблицу пользователь добавляется токен для того, что бы проверить подтвержден ли e-mail
2. public String generateEmailVerificationToken(String userId) {  
    String token = Jwts.*builder*()  
    .setSubject(userId)  
   // time of email verification = 864000000; // 10 days  
    .setExpiration(new Date(System.*currentTimeMillis*() + SecurityConstants.*EXPIRATION\_TIME*))  
    .signWith(SignatureAlgorithm.*HS512*, SecurityConstants.*getTokenSecret*())  
    .compact();  
    return token;  
    }

для того, что бы получить SecurityConstants.*getTokenSecret*())

в public class SecurityConstants {

создается метод

public static String getTokenSecret()  
{  
 AppProperties appProperties = (AppProperties) SpringApplicationContext.*getBean*("AppProperties");  
 return appProperties.getTokenSecret();  
}

для которого необходимо создать

@Component  
public class AppProperties {  
  
 @Autowired  
 private Environment env;  
  
 public String getTokenSecret()  
 {  
 return env.getProperty("tokenSecret");  
 }  
}

Который получает environment свойство tokenSecret (из application.properties)

@SpringBootApplication  
public class UserServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(UserServiceApplication.class, args);  
 }  
  
 @Bean  
 public BCryptPasswordEncoder bCryptPasswordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public SpringApplicationContext springApplicationContext() {  
 return new SpringApplicationContext();  
 }  
  
 @Bean(name = "AppProperties")  
 public AppProperties getAppProperties() {  
 return new AppProperties();  
 }  
}

+

*/\*\*  
 \* благодаря этому классу у UsersServiceImpl есть доступ к CONTEXT  
 \*/*public class SpringApplicationContext implements ApplicationContextAware {  
 private static ApplicationContext *CONTEXT*;  
  
 @Override  
 public void setApplicationContext(ApplicationContext context) throws BeansException {  
 *CONTEXT* = context;  
 }  
  
 public static Object getBean(String beanName) {  
 return *CONTEXT*.getBean(beanName);  
 }  
}

Эти классы необходимы для получения ApplicationContext и всех свойств

1. Меняем security так, что бы при неподтвержденном e-mail токен не создавался

Меняем метод ы классе UserServiceImpl

@Override

public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {

UserEntity userEntity = userRepository.findByEmail(email);

if (userEntity == null)

throw new UsernameNotFoundException(email);

/\*\*

\* если userEntity.getEmailVerificationStatus() == false -> login вернет error

\*/

return new User(userEntity.getEmail(), userEntity.getEncryptedPassword(), userEntity.getEmailVerificationStatus(),

true, true, true, new ArrayList<>());

}