


Задание1. Необходимо создать простое приложение SpringBoot и настроить для него SpringActuator.

```
<dependencies>  Add Starters...  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-web</artifactId>  
  </dependency>  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-data-jpa</artifactId>  
  </dependency>  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-actuator</artifactId>  
  </dependency>  
  <dependency>  
    <groupId>io.micrometer</groupId>  
    <artifactId>micrometer-registry-prometheus</artifactId>  
  </dependency>
```

Задание2. Настроить отображение healthcheck СУБД и места на диске.

```
1  server.port=8080  
2  spring.application.name=PrepareToProd  
3  
4  management.endpoints.web.exposure.include=health,metrics,prometheus  
5  management.endpoint.health.show-components=always  
6  management.endpoint.health.show-details=always  
7  
8  management.health.ping.enabled=false  
9  management.health.ssl.enabled=false  
10
```

Задание 3. Добавить пользовательские метрики «Количество заказов» и «Средний чек».

```
8
9 @Configuration
10 public class MetricsConfig {
11     @Bean
12     public MeterBinder counterMeterBinder(OrderRepo orderRepo) {
13         return MeterRegistry registry ->
14             Gauge.builder(name: "count.orders", orderRepo::getCountOfOrders)
15                 .register(registry);
16     }
17
18     @Bean
19     public MeterBinder averageSum(OrderRepo orderRepo) {
20         return MeterRegistry registry ->
21             Gauge.builder(name: "average.sum", orderRepo::getAverage)
22                 .register(registry);
23     }
24 }
```

Задание 4. Дополнительная задача: подключить и настроить SpringSecurity чтобы ограничить доступ к пользовательским метрикам.

```
9 @Configuration
10 @EnableWebSecurity
11 public class SecurityConfig {
12
13     @Bean
14     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
15
16         http
17             .csrf(AbstractHttpConfigurer::disable)
18             .authorizeHttpRequests( AuthorizationManagerRequestMat... requests -> requests
19                 .requestMatchers(⊗ "/actuator/metrics/count.orders", ⊗ "/actuator/metrics/average.sum") AuthorizedUri
20                 .hasRole("ADMIN") AuthorizationManagerRequestMat...
21                 .anyRequest().permitAll()
22             ).httpBasic(Customizer.withDefaults())
23             .logout( LogoutConfigurer<HttpSecurity> logout-> logout
24                 .logoutUrl("/logout")
25                 .logoutRequestMatcher(new AntPathRequestMatcher(⊗ "/logout", httpMethod: "GET"))
26                 .logoutSuccessUrl("/actuator/metrics")
27                 .invalidateHttpSession(true)
28                 .deleteCookies( ...cookieNamesToClear: "JSESSIONID")
29                 .clearAuthentication(true)
30             )
31             .headers( HeadersConfigurer<HttpSecurity> headers -> headers
32                 .cacheControl(HeadersConfigurer.CacheControlConfig::disable)
33             );
34
35         return http.build();
36     }
37 }
```

```
@Bean  👤 patroN
public UserDetailsService userDetailsService(PasswordEncoder passwordEncoder) {
    UserDetails admin = User.builder()
        .username("admin")
        .password(passwordEncoder.encode(rawPassword: "root"))
        .roles("ADMIN")
        .build();

    UserDetails user = User.builder()
        .username("user")
        .password(passwordEncoder.encode(rawPassword: "1234"))
        .roles("USER")
        .build();

    return new InMemoryUserDetailsManager(admin, user);
}
```

```
@Bean  👤 patroN
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```