Gift Certificates

Task 4. JPA + Security. Estimated time: 10 days

2019

# Functional Requirements

This task is an extension of the RESTful web-service from Task 1 for Gift Certificates system.

Application should support *user-based authentication*. This means a user is stored in a database with some basic information and a password.

The system should be *extended* to expose the following REST APIs:

1. Change price of a Gift Certificate.
2. Buy Gift Certificate for a User.
3. Get information about User’s certificates.
4. Get information about a User’s certificate purchase: cost and timestamp of a purchase.
5. Get the most popular Tag of a User with the highest cost of all certificates.
   1. Demonstrate SQL execution plan for this query.
6. Search Gift Certificates by several Tags (“and” condition).
7. Sign up.
8. Log in.
9. All *read* operations should support pagination. Please create a flexible and non-erroneous solution. Handle all exceptional cases.

# User Permissions

Guest:

1. Read operations for Gift Certificates.
2. Sign up.
3. Log in.

User:

1. Buy Gift Certificates.
2. All read operations.

Administrator (can be added only via database call):

1. All operations, including addition and modification of entities.

# Technical Requirements

## General Requirements

1. Code should be clean and should not contain any “developer-purpose” constructions.
2. Application should be designed and written with respect to OOD and SOLID principles.
3. Code should contain valuable comments where appropriate.
4. Public APIs should be documentedusing Javadoc.
5. A clear layered structure should be used: responsibilities of each application layer should be defined.
6. JSON should be used as a format of client-server communication messages.
7. Convenient error and exception handling mechanism should be implemented: all errors should be meaningful and localized on backend side.
8. Abstraction should be used to avoid code duplication.

## Tools and Implementation Requirements

**Please note that only GA versions of tools, frameworks, and libraries are allowed.**

1. JDK version: 8. Use Streams, java.time.\*, an etc. where it is appropriate.
2. Application packages root: com.epam.esm.
3. Spring Boot.
4. Custom Connection pool should be changed to a well-known implementation of DataSource (i.e. Hikari).
5. JPA / Hibernate should be used for data access.
   1. Audit data should be populated using JPA features.
6. Java Code Convention is mandatory. Exception: margin size – 120 characters.
7. Build tool: Gradle 4.+. Multi-module project.
8. Spring Framework 4.+
9. Application container: Spring IoC.
10. Database: PostgreSQL 9.+ or 10.+
11. Testing: JUnit 4.+ or 5.+. Mockito.
12. Service layer should be covered with unit tests not less than 80%.
13. Spring Transaction testing should be performed in scope of this task.
14. Spring Security should be used as a security framework.
15. Server should support only stateless user authentication and verify integrity of JWT token.
16. Use OAuth2 as an authorization protocol.
17. OAuth2 scopes should be used to restrict data.
18. Implicit grant and Resource owner credentials grant should be implemented.
19. Implement CSRF protection.
20. For demo, generate at least:
21. 1000 users,
22. 1000 tags,
23. 10 000 gift certificates .

Gift certificates should be linked with tags and users. All values should look meaningful: random words, but not random letters.

1. APIs should be demonstrated using Postman tool.
2. For demo, prepare Postman collections with APIs.
3. *Optional: use HTTPS protocol.*

## Restrictions

It is forbidden to use:

1. Spring Data Repositories,
2. Lombok,
3. Powermock,
4. Any Hibernate specific features.

# Recommended Materials

This section contains links to materials recommended for self-study.

1. [Spring in Action, 4th edition](https://www.manning.com/books/spring-in-action-fourth-edition)
2. [Foundations of RESTful architecture & REST practices](https://dzone.com/refcardz/rest-foundations-restful)
3. [Best Practices for Designing a Pragmatic RESTful API](http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api)
4. [Data Transfer Object](https://martinfowler.com/eaaCatalog/dataTransferObject.html)
5. [Understanding Spring Web Application Architecture: The Classic Way](https://www.petrikainulainen.net/software-development/design/understanding-spring-web-application-architecture-the-classic-way/)
6. [PowerMock examples and why better not to use them](https://automationrhapsody.com/powermock-examples-better-not-use/)
7. [Аутентификация при помощи JWT и Spring Security](https://habrahabr.ru/post/337600/)
8. [JWT](https://jwt.io/)
9. [OAuth 2.0 простым и понятным языком](https://habrahabr.ru/company/mailru/blog/115163/)
10. [Spring Boot and OAuth2](https://spring.io/guides/tutorials/spring-boot-oauth2/)
11. [OAuth2 Grants](https://alexbilbie.com/guide-to-oauth-2-grants/)
12. [Spring Transaction Testing](https://docs.spring.io/spring/docs/current/spring-framework-reference/testing.html#testcontext-tx-programmatic-tx-mgt)
13. [Random Words Generator](https://randomwordgenerator.com/)
14. [Java/JPA Developer's Guide](https://www.objectdb.com/java/jpa)
15. [CSRF](https://docs.spring.io/spring-security/site/docs/current/reference/html/csrf.html)

My links:

Gradle

<https://www.tutorialspoint.com/gradle/>

<https://www.youtube.com/watch?v=JwPYjnhah3g>

multi module - <https://habr.com/ru/post/225189/>

Paste maven dependencies <https://sagioto.github.io/maven2gradle/>

<https://habr.com/ru/company/redmadrobot/blog/275515/>

Setting up Gradle web project in Eclipse (on Tomcat Server) <https://medium.com/@wkrzywiec/setting-up-gradle-spring-project-in-eclipse-on-tomcat-server-77d68454fd8d>

<https://o7planning.org/ru/11247/create-a-gradle-java-web-application-and-run-on-gradle-tomcat-plugin>

dependency <https://github.com/bmuschko/gradle-tomcat-plugin>

dependency <https://plugins.gradle.org/plugin/com.bmuschko.tomcat>

JPA

<https://easyjava.ru/data/jpa/jpa-i-svyazi-mezhdu-obektami/>

inheritance <https://thoughts-on-java.org/complete-guide-inheritance-strategies-jpa-hibernate/>

<https://easyjava.ru/data/jpa/jpa-i-svyazi-mezhdu-obektami/>

<https://www.baeldung.com/jpa-many-to-many>

<https://www.callicoder.com/hibernate-spring-boot-jpa-many-to-many-mapping-example/>

<https://hellokoding.com/jpa-many-to-many-relationship-mapping-example-with-spring-boot-maven-and-mysql/>

entity-and-dto

<https://hellokoding.com/mapping-jpa-hibernate-entity-and-dto-with-mapstruct/>

<https://breakpoint.digital/auto-map-object-to-dto-spring-rest/>

CP <https://www.concretepage.com/spring-boot/spring-boot-hikaricp>

<https://docs.spring.io/spring-boot/docs/current/reference/html/howto-data-access.html>

CriteriaQuery examples

<https://www.boraji.com/hibernate-5-criteria-query-example>

<https://howtodoinjava.com/hibernate/hibernate-criteria-queries-tut>

<https://www.tutorialspoint.com/hibernate/hibernate_criteria_queries> +++

<https://discourse.hibernate.org/t/org-hibernate-annotationexception-use-of-onetomany-or-manytomany-targeting-an-unmapped-class/897>

<https://www.dineshonjava.com/hibernate/introduction-to-criteria-api/>

<https://www.programcreek.com/java-api-examples/?class=javax.persistence.criteria.CriteriaBuilder&method=createQuery>

<https://spring.io/blog/2011/04/26/advanced-spring-data-jpa-specifications-and-querydsl/> specification

<https://spring.io/blog/2011/02/10/getting-started-with-spring-data-jpa>

<https://www.logicbig.com/tutorials/spring-framework/spring-data/specifications.html>

. HQL Select Query Example <https://www.mkyong.com/hibernate/hibernate-query-examples-hql/>

<https://www.javatpoint.com/hql>

*EntityManager*

<https://www.programcreek.com/java-api-examples/?class=javax.persistence.criteria.CriteriaBuilder&method=createQuery>

<https://www.programcreek.com/java-api-examples/?api=javax.persistence.criteria.CriteriaQuery>

session

<http://www.java2novice.com/hibernate/session-interface-methods/>

time

<https://habr.com/ru/post/273177/>