Distributed Systems
Homework task #1

Spring Boot application

Description

The aim of this homework project is to create a Spring Boot application, which will be used in other upcoming homework projects as the basis.

Steps

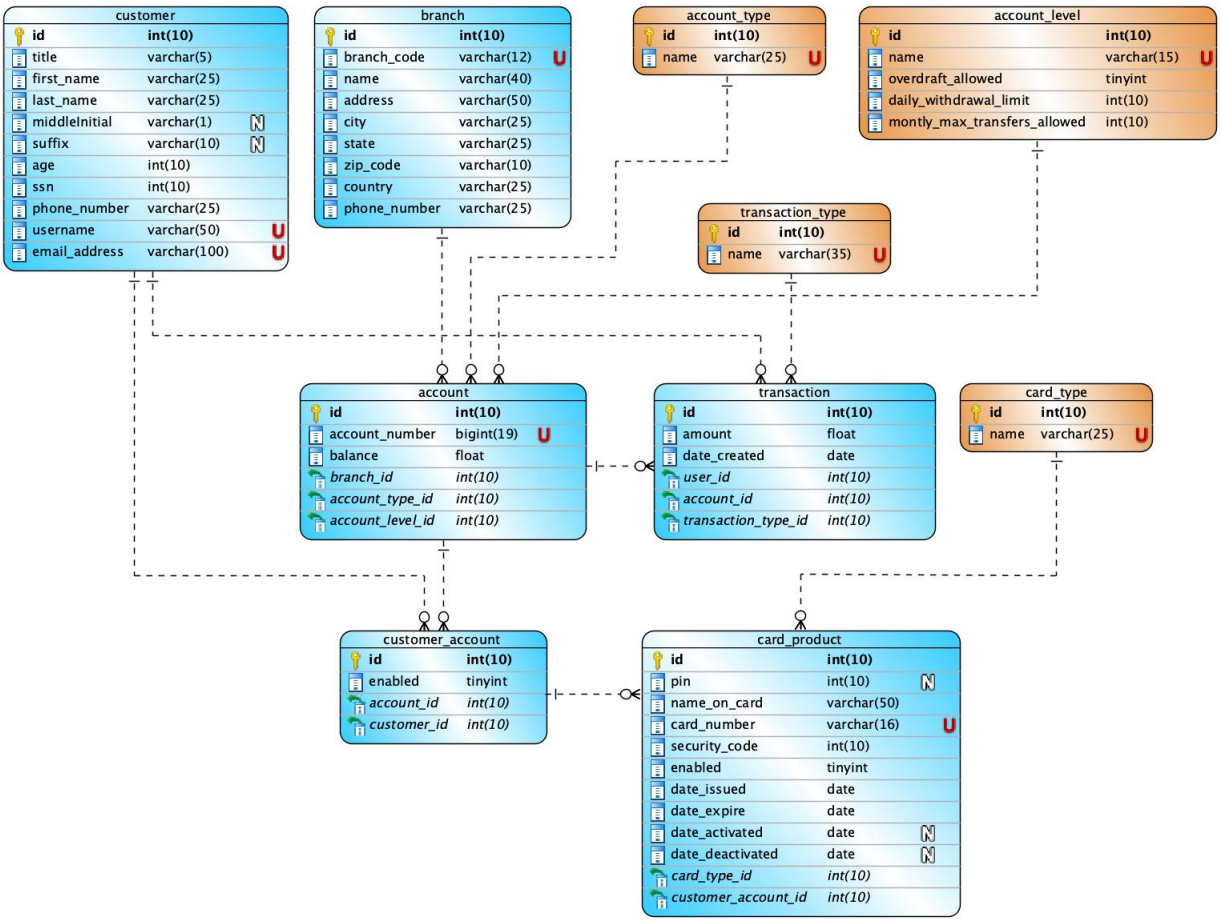
1. Choose any business area, which has enough data entities for this project. That could be any kind of business, like banking, medicine, commerce, industry etc.
2. Define the ER (Entity-Relationship) diagram. This could be just a sketch of the business data model, from which the model Java classes will be created. The example ER diagram is shown below (Picture 1).
3. Create Spring Boot application from Spring Initializr web page: https://start.spring.io/. Select Java as language, Maven as build system (optionally you may choose Gradle as build system, Groovy or Kotlin based modern Java projects build system). Select Java version from 17 to 23. For older Java versions this Spring Initializr page can be used https://springinitializrjava8.cc/.
4. Define the Group name for your project and Artifact name. The Group name can be the same for all your projects, as this is the root directory where Maven will store all your projects in local repository. Only Artifact name may differ. **Follow the naming conventions for package names!** (See references section for more information).
5. On Spring Initializr web page add dependencies for at least **Spring Web** and **Spring Data JPA.** This allows you to add JPA and Hibernate support for your application and create REST endpoints.
6. Add actuators to your Web application. After application startup, Spring Boot will launch the default embedded Tomcat Application Server and actuators should be available on the path actuator/health and actuator.
7. Add configuration for any Database of your choice, e.g. H2 In-Memory DB, MySQL, Oracle etc.
8. Add Java classes that correspond to your defined data model. Put corresponding JPA annotations. **Note that when using JPA and Hibernate in Java Enterprise projects (like Spring Boot based project), Data Entity tables will created for your by Hibernate framework, provided you added proper annotations on each of your model class.**
9. Create application startup script to load initial data to the Database. This also could be an SQL script.
10. Create Java service class which takes the data from Database and transforms it to XML document. JAXB framework and corresponding annotations should be used.
11. Create single REST endpoint which responds to HTTP GET request and accepts some Request Query Parameters. Filter data from your database by the values of these parameters, using Repository class fetch some data and transform it to XML representation. **Save this XML to local directory and return this local path as endpoint response. Do not send the generated XML!**
12. Test the endpoint using any API Testing Tool (Postman, Insomnia, SoapUI etc.)

General Requirements

1. The entire code should be properly formatted.
2. The package/class/field/method names should conform to the naming conventions.
3. The Unit tests for all classes should present.
4. The entire code should be properly documented with JavaDoc comments.
5. The entire code should conform to S.O.L.I.D principles.

References

1. Building an Application with Spring Boot
2. Naming Conventions
3. Guide to Java Packages
4. JPA - Introduction
5. Hibernate. Everything data
6. Hibernate Tutorial
7. Postman Tutorial
8. Spring @RequestParam Annotation
9. Unmarshalling XML Data into Java Objects using Spring Boot
10. How to Use JAXB for XML Processing with Java
11. S.O.L.I.D: The First 5 Principles of Object Oriented Design
12. How to Write Doc Comments for the Javadoc Tool



Picture 1. Example ER diagram