Table of Contents

1. About the document

The purpose of the document is to give more details about the Code, API design, and design decisions taken during the design and development of the Customer Service.

2. Deliverables

- This document
- Code in GitHub - https://github.com/satheeshkumarmayalagan/qtest
- REST API Contract - https://app.swaggerhub.com/apis/qtest/CustomerService/1.0.0/

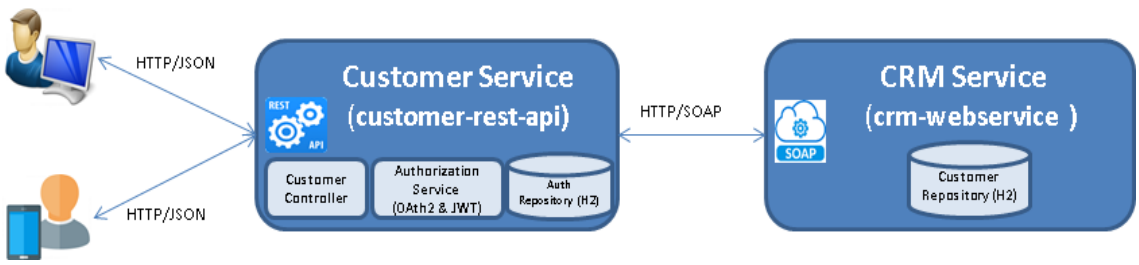
3. Assumptions

- Backend CRM application exposes SOAP-based Web Services

4. High-level Overview

The code contains two applications:

1. **customer-rest-api** – A Spring boot-based application that exposes the REST APIs given in the contract
2. **crm-webservice** – A Spring boot-based application that exposes SOAP-based Web Services used to mock the CRM application. This mocks the CRM service.



customer-rest-api exposes REST APIs to manage the customer profile data. The service also contains the Authorization Server and Resource Server to secure the REST APIs. The user database is stored inside the H2 in-memory database. The API call in turn makes calls to the SOAP service exposed by crm-webservice

crm-webservice exposes SOAP services. The SOAP services manages the customer data stored in the H2 in-memory database

Object-to-object mapping is used which will impact the extensibility and that any changes to the data structure will require code change, build and deployment. It is possible to make it more extensible by using XPath, JsonPath, properties, beanutils and following naming conventions, object-object mapping is preferred when structures are complex and requiring transformations.

Some of the dependencies are:

- java 8
- maven 4
- spring-boot-starter-web
- spring-boot-starter-security
- spring-boot-starter-data-jpa
- spring-boot-starter-web
- com.h2database— Database for storing auth user data
- spring-security-jwt
- spring-security-oauth2
- mockito

5. Real-world Considerations

The following things should be considered for production environment

- Secure the HTTP transport with TLS (**HTTPs**)
- Authorization server will be a separate service with which the customer service will interact to secure the REST APIs
- The users will typically be in an LDAP-based Identity Management system
- Production-scale RDBMS will be used instead of H2

6. Testing

For testing purposes, auth users and customers are loaded into H2 database.

6.1. Unit Testing

Both the application uses JUnits for unit testing.

customer-rest-api has a mock web service (MockCustomerWebService.java) for crm-webservice dependency. application.yml defines the test profile to route to the mock web service instead of the crm-webservice end-point

6.2. Application Testing

Steps for testing the services:

- Build both the applications using maven 4 and java 8

```
mvn clean install
```

- Run the applications

```
java -jar target\customer-rest-api-0.1.0.jar
```

```
java -jar target\crm-webservice-0.1.0.jar
```

- customer-rest-api will expose the REST APIs in port **8080**
- crm-webservice will expose the SOAP services in port **9090 -** http://localhost:9090/crm/ws/customer.wsdl
- Get the JWT access token:

```
POST http://localhost:8080/oauth/token
Header: Authorization as Basic YXBpdXNlcjpGZXJqc2FkZ2ox
Header: content-type as application/x-www-form-urlencoded
Body:
username as admin.admin
password as jwtpass
grant_type as password
```

- Get the Customer details:

```
GET http://localhost:8080/customer/1
Header: Authorization as Bearer <<access_token from above step>>
```