Data requirements

Table of contents

Introduction

This document is going to identify the required data to fulfill the planned functionality of the application. What is more, details about data storage and security are going to be described.

Based on the architecture design, the application is planned to consist of 7 micro-services and 2 API gateways in total. Gateways will not require a database since their sole purpose is operating with the rest of the microservices and their functionality. Each microservice will have a database of its own, containing relevant data for the service. Depending on the data load, different database types could be utilized in the services to ensure performance.

Data needs

An overview of the required data for each service is described in the list below:

- Customer Authentication Service:
  - Authentication credentials – username, password, e-mail address, salt, authentication tokens;
- Customer Account Service:
  - Customer data not needed for authentication but related to their account – names, age, payment methods, preferences, account settings;
- Customer Payment Service:
  - Details about a customer's payment – for the current development of the application, it is going to store only mock data since no payment will be implemented;
- Orders Service:
  - Details about a customer's order – product identification, customer identification, timestamp, payment status, payment reference,
- Product Service:
  - Details about each product – product name, description, business ID, contents, price
  - Details about product stock – amount of available product
- Business Account Service:
  - Data related to a business account – business name, chamber of commerce details, contact details, account settings;
- Business Authentication Service:
  - Authentication credentials – username, password, e-mail address, salt, authentication tokens.

Data Storage and Load

In this section, the expected load and storage requirements for each service are going to be taken into account. What is more, a database type will be selected for each service:

- Customer Authentication Service:
  - Read/Write = 90%/10%;
  - A lot of small read operations for authenticating users;
  - No complexities in queries required;
  - Apache Cassandra;
- Customer Account Service:
  - Read/Write = 80%/20%;
  - Low load expected;
  - Entity relations required;
  - PostgreSQL database;
- Customer Payment Service:
  - Read/Write = 10%/90%;
  - Data relations not required;
  - No complexities in queries required;
  - Apache Cassandra;
- Orders Service:
  - Read/Write = 10%/90%;
  - High load expected;
  - Data relations not required;
  - No complexities in queries required;
  - Apache Cassandra;
- Products Service:
  - Read/Write = 70%/30%
  - Moderate load expected;
  - Entity relations required;
  - PostgreSQL database;

Security and Compliance

In order to make sure that all data stored by the application is secure, some measures have to be applied beforehand.

Encryption

In order to ensure that data is safe within the database, it needs to be encrypted. Storing sensitive information encrypted in the database ensures that even if a hacker manages to gain access to the given database, they still need to get through the encryption to access the real data. Furthermore, data should be encrypted during transmission as well, since communication

between services and databases could be intercepted as well. More details on the encryption are going to be provided once a research is done on the topic.

Authentication and Access control

In order to ensure that unauthenticated users cannot access the databases, every request at the API gateway is going to be authenticated before being allowed to access any resources. Following the authentication, the user's permissions are going to be validated for the request they are trying to make. In case the permissions of the user do not match the required permissions for accessing the requested resource, the user will not be granted access to the data. For example, regular customers are going to be able to only read from the products database while business accounts would also be allowed to add and modify products in it. This access control is going to be implemented in the API gateways.

Logging

In order to monitor all operations executed in the databases, event logging is going to be implemented. It will log every attempt to access, change, delete data, both authorized and unauthorized. Using this method will allow a person who is monitoring the system to identify possible data leaks and breaches.

GDPR

GDPR is a legal framework to control the processing of personal data. It covers all businesses which handle personal data of EU citizens. The goal of GDPR is to protect people's right to privacy while allowing the safe processing of their data. Processing a person's demographic data, personal preferences, health data, etc. are all considered as sensitive by GDPR. In order to apply these regulations most efficiently, the development would be focused on minimizing the sensitive data stored in the system and apply security measures where possible. A mechanism for deleting a user's data has to be implemented in case some user withdraws their consent to contain to keep their data.