Software Architecture

CSE2115 Software Engineering Methods Group 08b

Y2Q2 2021/2022

Domains Types

We identified four core domains. These are the main focus of the system. In order to be able to book a room we need the following:

- Users
- Rooms
- Buildings
- Bookings

We identified one generic domain that has to be implemented withing the entirety of our system:

- Authentication

We also identified one supporting domain. This helps support one of our core domains(User) but it is not critical and the system can function without it:

- Research groups

Bounded contexts

Applying the Domain Driven Design approach, we identified the following bounded contexts:

- Users
  - In our application, we have different types of users: admin, secretary and regular user.
  - Within the users context, these types of users are modeled as the same entity (with different roles).
  - Within different contexts, these types of users can be mapped to different entities(E.g. in the booking context, users are either admins, secretaries or regular users).

  - Users can also be part of a group. One user can be a member of multiple groups.

- Rooms
  - Rooms can be booked by users
  - Entities with a set of features on which they can be filtered.
  - Within all contexts, each room is modeled as the same entity(only one type).

- Buildings
  - Buildings host the rooms which can be booked.
  - They determine the opening/closing times of the rooms inside them.
  - Within all contexts, each building is modeled as the same entity(only one type).

- Bookings
  - Bookings provide the relationship between a user who created the booking and the booked.
  - A Booking contains a list of users, the participants, specified by the creator of the booking. A Booking has a reference to specific room and start/end times.
  - Within all contexts, each booking is modeled as the same entity (only one type).

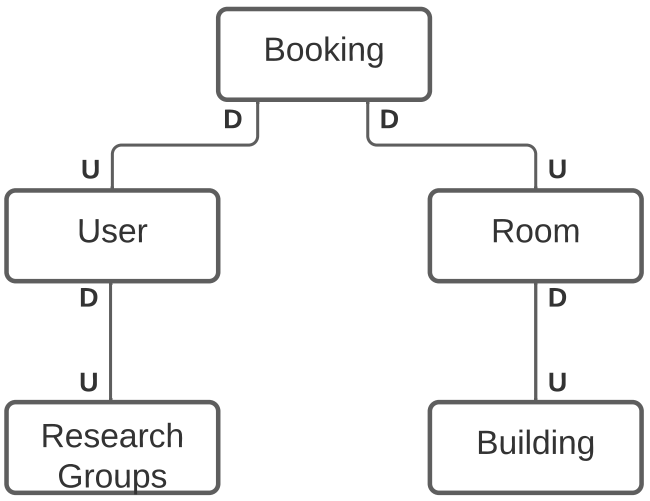

Figure 1: Context map

- Partnership: Partnerships between contexts have been described above in Figure 1, where the upstream context has a direct effect on the downstream context. For example, the building will affect the room, as a building's opening hours will effect the availability of the room. Furthermore, the room effects the booking, as the opening hours of the building is equal to the opening hours of the room. Thus, the booking can only be within this period and is downstream from both room and building.
- Conformism: None
- Anti-Corruption Layer(ACL): None

Microservices

The four bounded contexts are mapped to three microservices, each with their own database:

- User Microservice: This microservice handles anything to do with the users. It helps the system identify users and notifies the main gateway. This helps the user in question to perform actions within the system. In addition, it also has a layer of security that helps validate user logins in a safe and secure way. In this microservice we also store information about groups and their members.
- Room Microservice: This microservice manages the rooms and the buildings the rooms belong to. It also deals with figuring out the availability of the rooms. In addition, each room has several attributes from equipment to capacity, which are all stored in the database that this microservice takes care of.
- Booking Microservice: This microservice manages the making of bookings and the list of bookings overall. Each booking is tied to a user, so when a user requests to make a booking or see their own booking, this microservice takes care of it.

Component Diagram

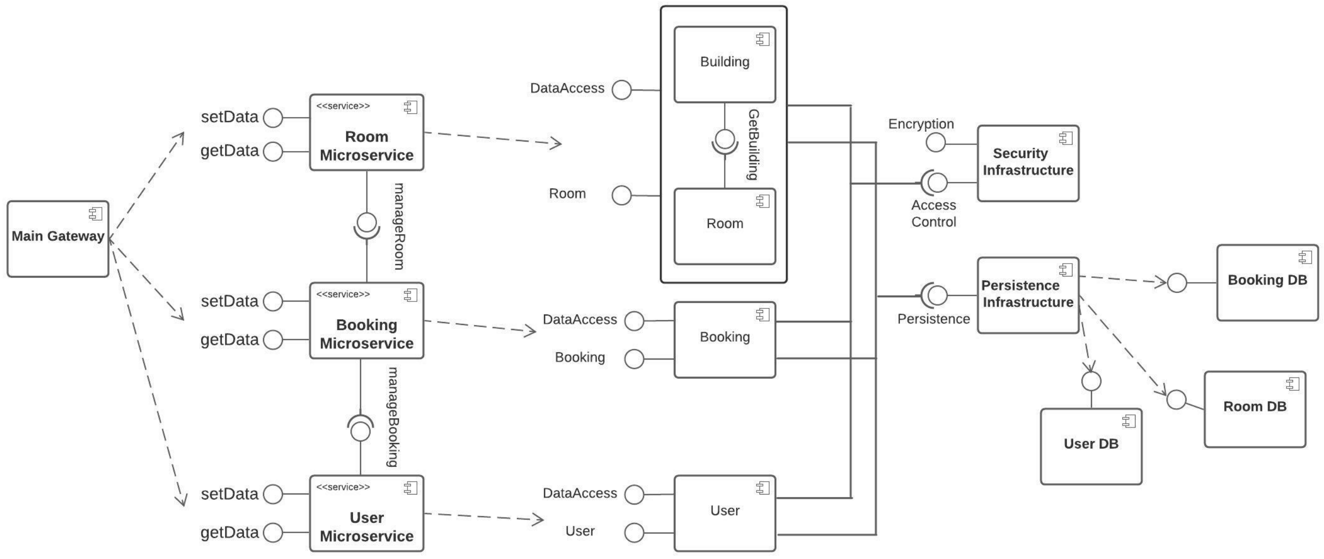

Figure 2: Component diagram showcasing interaction between microservices

This diagram shows how the microservices communicate with each other. The main gateway routes all the data between the microservices as required. We can see that the Booking microservice requires both the User microservice and Room microservice to manage bookings and rooms. The methods manageBooking and manageRoom refer to all the methods that are to do with making, editing and deleting rooms and bookings. It also includes any other methods we may have to implement when it comes to the interaction between these microservices. The diagram also shows how each entity is implemented within the microservices and all of them will have a security layer and a persistence layer. The security layer is responsible for encrypting sensitive data and the persistence layer is responsible for storing the data in databases (one database for each service).