Assignment 1 Made by Group 11b

(Context diagram)

As you can see in the context diagram above, the bounded contexts we have derived are users, reservations, rooms and authentication. We chose these bounded contexts to be our microservices, which is why we mapped them directly into microservices. So we will have 4 microservices in our project. We have chosen these microservices, since we think that we can cover all functionality of the program in these microservices. We can use the user microservice for handling everything concerning the user of the program, such as defining a role and navigating through the system. In addition, we use the reservation microservice for handling the situation when a user wants to make a reservation. Next to that, we use the room microservice for handling everything concerning a specific room, such as checking the availability, or changing the available equipment. Lastly, the authentication microservice handles the authentication of several types of users (employees, secretaries and admins). The other 3 microservices are connected to this microservice to ensure security. The user, room and reservation microservice are all core microservices since these services contain the main parts of the program. These services depend on authentication, which is why the authentication microservice is a generic microservice.

Microservices:

**Users**

The user microservices will serve as a service where the user will find the most appropriate functionality for using the system. The microservice will be connected to the other three microservices to fully connect the whole program, while the user microservice on its own will provide the service of selecting the appropriate role in the system, which will communicate to the authentication microservice through the API to authenticate the specific role.

Furthermore, this service makes sure all the stated rules are followed, such as providing a reason for booking a room and making sure an employee does not book a room more than two weeks in advance. If this does happen, this microservice should provide a clear error message to help the user to see what went wrong.

Every request from this microservice will need at least one of the other three microservices to provide the necessary information, with an example mentioned above. Therefore, the user microservice is a service which helps the user to get to the information they want faster, since the user microservice contains the information of the specific user, which makes communicating to the other microservices easier, since the specifics of the user do not need to be looked up anywhere anymore.

We have chosen to make this microservice a separate microservice, because the user is one of the most important parts of the system. We need to make sure that the data of each user is (safely) stored and that actions can be done under their name. We saw no other option than making a separate microservice for users.

The overall responsibility of this microservice is to make sure that everything runs smoothly for the user, that their data is stored safely and making sure that the data is edited correctly when applicable following the ACID principle (atomicity, consistency, isolation, durability).

**Reservations**

The reservation microservice will serve as a service which handles the making of the reservation and everything else that has to do with reservations, such as editing or cancelling it. This microservice will be working in between the user and room microservice, since if a reservation is made, the request will go from the user microservice, through the reservation microservice and stored with the specific room which is handled by the room microservice. It also works the other way around, the room microservice might request information about certain reservations, because the user might want to know when a specific room is booked and when there are openings in the schedule.

We have chosen to make the reservation microservice a separate microservice, because we felt that it was one of the most important parts of the entire system, which could not be merged with another microservice. We thought about merging it with the room microservice (which is mentioned below), but in the end we decided that the system would be clearer and more efficient by making a separate microservice for reservations.

The overall responsibility of this microservice is to handle all actions regarding reservations, such as generating, editing, cancelling and storing.

**Rooms**

The room microservice will serve as a service which handles all the requests which have to do with a specific or multiple rooms. The room microservice will be connected to the user as well as the reservation microservice since the info of a room can be requested directly requested by a user as well as indirectly via making a reservation.

Furthermore, this microservice will make sure that every rule of the system regarding a room is adhered to, such as that a room which is under maintenance cannot be booked . If this is not the case, the microservice will provide an error to the user microservice, which will convert it to a clear error message for the user.

Regarding other functionalities of this microservice, it will deal with storing data about the different buildings where the rooms are located, together with the equipment and the maximum capacity of a room.

Other than that, it will also allow system admins to change/add certain elements to one of the fields mentioned above.

We have chosen to make rooms a separate microservice, because we thought that there would be enough requests during the use time of the system which could be handled specifically by the room microservice and where combining it with other entities of the system would create less clarity and division of functionality in the system. Next to that, we plan to include the building entity in the room microservice, since the building entity seems to be less frequently used by the system, therefore it would make more sense to make the rooms a priority over the buildings where they are located.

The overall responsibility of the microservice is to make sure that everything regarding the rooms and the schedule is when a room is booked is handled correctly and in an efficient manner.

**Authentication**

This microservice is a generic microservice in contrast to the other three core microservices, which means that the system depends on this service. This microservice will provide the authentication of the system and is connected to the user microservice, which is also the responsibility of the microservice, since when the user opens the system, he/she should authenticate themselves immediately. After a successful authentication, a token is sent with every request, of which the other microservices can check the existence without needing the authentication microservice again. We are going to use JWT tokens to ensure that the users of the system are authenticated. We have chosen for this method, since a part of the group is already familiar working with JWT tokens and this method should fit the requirements that we want for the authentication system.

We have chosen to make a separate authentication microservice, because we need authentication in each of the other three core microservices which are mentioned above. It would create code duplication if we would create an authentication system of each microservice, which is why we chose to make a dedicated microservice for the purpose of authentication. The other microservices can make use of this microservice to ensure the security of the system, which is also the responsibility of this microservice.

**Architecture**

Since we are using microservices, we have a separate database for each microservice. For the password of a user specifically we will use encryption to make sure the passwords cannot be leaked easily.