Distributed Systems (H0N08A)

Report: Developing a locally deployed distributed application

Lomis Orestis *(r0735890)*
Lommaert Stefan *(r0737707)*

November 25, 2022

Question 1 The Booking Platform allows multiple clients to ask about available flights via REST. The Booking Platform gets this info from the airlines via REST and passes it to the client. When a client wants to confirm the quotes and make a booking, the Booking Platform passes the info to the BackGroundWorker via Pub/Sub. Once the booking is done, this info is saved on the Firestore server. The user can then see their bookings on the front-end. See Figure 1 for the deployment diagram.

Question 2 The whole application is built upon the Maven Springboot middleware. This allows us to more easily create a REST API. Then we also used the Google Cloud Pub/Sub and Firestore middleware to implement the indirect communication and to store data to a database respectively.

Question 3 The quotes are created and collected in the cart. Before the actual booking is created, at the confirmation of the quotes, the indirect communication kicks in. Then we allow background workers to confirm the quotes and check whether or not this is even possible. If not, no quotes are confirmed.

Question 4 We only sent the customer email and the quotes as a byte string, which contain all the necessary information. The background worker then makes a new Quote object from this data.

Question 5 Responsiveness actually goes down with indirect communication. This is a trade-off with the big scalability improvements. By using indirect communication we can be agnostic about the connections within the network, which would otherwise be a huge overhead in an internet network. The responsiveness being slightly worse is not too bad of a trade-off for this massive gain.

Question 6 When creating the bookings, i.e. browsing the site and picking flights and seats, we do not want the user experience to be bad. So this must be responsive and would not involve any indirect communication. When the tickets are being booked this is not such a big deal.

Question 7 This should not be possible. Double bookings by themselves are ruled out by the airlines. Our implementation of ACID properties simply deletes all booked flights when 1 booking fails. These also don't get stored in the Firestore.

Question 8 This makes it very easy to filter based on roles. We can do a quick check on the role of the user and allow or disallow them access to certain funcionalities.

Question 9 We would need to change the WebSecurityConfig. This configures the authentication provider. And we would also need to change the security filter in SecurityFilter, because it depends on the WebSecurityConfig.

Question 10 We give the Unreliable Airline 5 tries to send its info. If the unreliable airline still fails to connect after 5 tries, we don't send any data from the Unreliable Airline back to the client. The client just sees the data from the Reliable Airline. Or if the client specifically asks data from Unreliable Airline, like a seat, we just return an empty list. The client then just gets a blank page and doesn't crash, he can still go back to the previous menu.

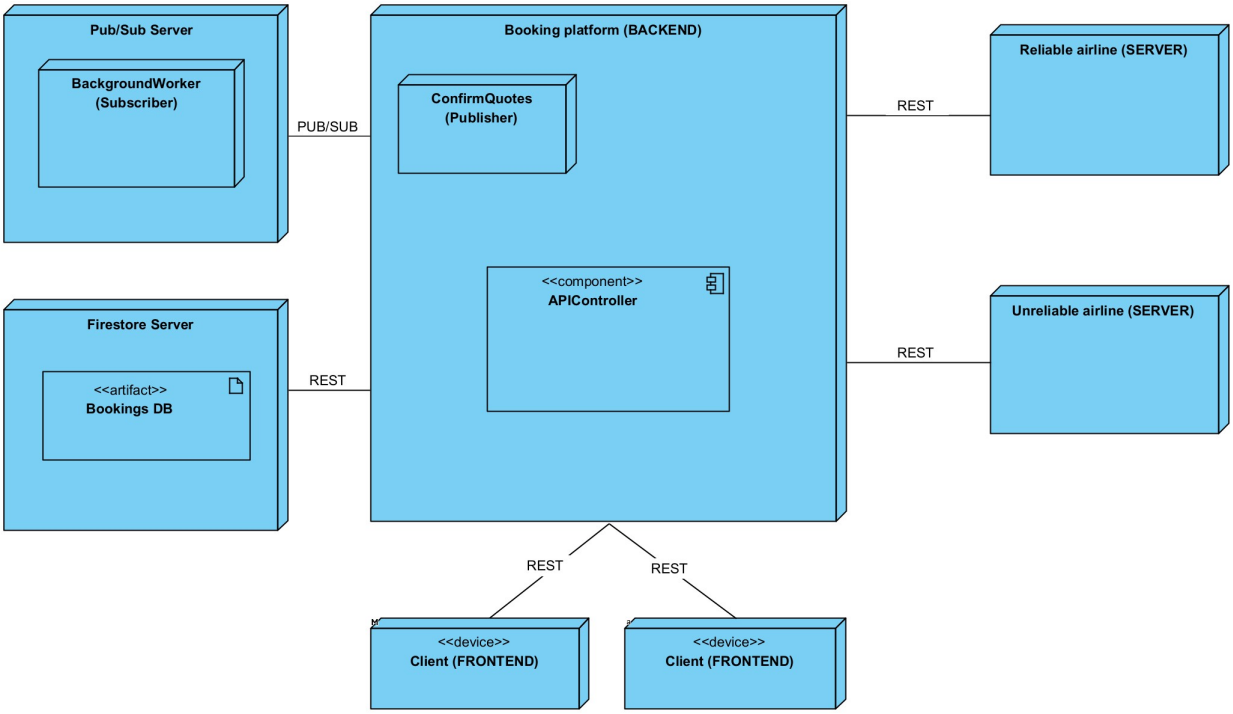

Figure 1: Deployment Diagram