Distributed Systems (H0N08A)

Report: Project

Thibaut Mertens *(r0762702)*
Ignasi Juez Guirao *(r0974979)*

November 17, 2023

Question 1 Figure 1 represents a diagram of the deployment of our application and its components. The client located on the left of the figure represented as a laptop connects with the Firebase authentication and the Front-end app, which provides us with a connection between the front-end and the backend. The Front-end app makes multiple connections with different train companies, a reliable one and an unreliable one for the laboratory implementation where more train companies could be added in the future. To not saturate the front-end and worsen the user experience we implemented the pub/sub cloud, this receives the bookings from the front-end and pushes them as quotes to the worker which will store them in the Cloud Firestore Database. And last, the Cloud Firestore Database where all data is stored is connected with the front-end and the backend to achieve our goal of providing all the information to the customer in the front-end.

Question 2 With the Firebase emulator suite, we were able to use the Firebase database without any configuration beforehand, it also provided us with an authenticator emulator which has provided us with security measures in our application and the Pubsub emulator for indirect communication.

Question 3 Indirect communication between components takes action in createbooking. The procedure that triggers indirect communication begins with the front-end app receiving the quotes from the customer. Through the Securityfilter we obtain the client's email and create a data entity that includes the client's email and quotes. This will be published to the cloud pub/sub and received by the bgWorker subscription. All the client's quotes will be attempted to reserve their tickets and create the client booking, in case the reservation of a ticket fails, all the previous ones will be undone by a rollback mechanism.

Question 4 The data passed between the application and the bgWorker when creating a booking is the client's email and his requested quotes. Being a quote an object composed of trainCompany, trainId and seatId. We think that could be a good idea since it is a large amount of information exchanged after all.

Question 5 Our pub/sub solution for indirect communication introduces a significant enhancement in terms of scalability and responsiveness within a distributed/cloud context. By adopting a pub/sub architecture, we achieve a crucial decoupling in both time and space. Scalability is achieved by moving the most intensive work to the backend where the infrastructure can effortlessly handle a heavier workload without overburdening the front-end. This adaptability allows us to scale our solution horizontally by adding more background workers to accommodate growing demands. The decoupling of tasks ensures that the front-end remains responsive even when dealing with bigger operations, preventing any noticeable delays or disruptions in the user interface.

Question 6 A user action where indirect communication would not be beneficial in terms of the trade-off between scalability and usability is that if only one request is processed at a time, utilizing Pub/Sub may introduce additional overhead compared to having the Front End handle the task directly. Even when booking a single ticket, the absence of indirect communication could prove more advantageous.

Question 7 Due to our implementation, a double booking is not possible in normal working conditions. This is because a ticket of a seat is only to a booking when it successfully recieved a ticket from the train company server. However, it should be emphasized that if there is a bug in the train company server that gives two tickets, a double booking can take place.

Question 8 RBAC simplifies the requirement by providing a structured and flexible approach to managing access permissions based on roles. Where roles provide a logical and organized way to group related permissions, it allows us to add more roles or modify existing ones without making any major changes to the application.

Question 9 When switching to another authentication provider, several components in the application may need to change as authentication mechanism, user identity management and security configuration that will have to be suitable for the new authentication provider. Changing to a non-role-based authentication provider would make it less easy to correctly enforce access control rules since role-based gave us a lot of ease. The characteristics we should look for on an ideal authentication provider could be standardized protocols, customization options and clear documentation.

Question 10 We modified the system to be more fault tolerant and handle failures appropriately when one component or one connection between two components fails. The procedure we implemented is based on retries when the connection fails, the error is propagated to the retry where we implement a backoff strategy where the time between retries will increase exponentially. It stops after a limited amount of retries to make sure the frontend is not blocked for too long. If the train company's faults are occasional and do not severely impact essential functionalities, the application may continue to operate smoothly. However, if the faults are frequent or critical like a service downtime the impact can be more significant.

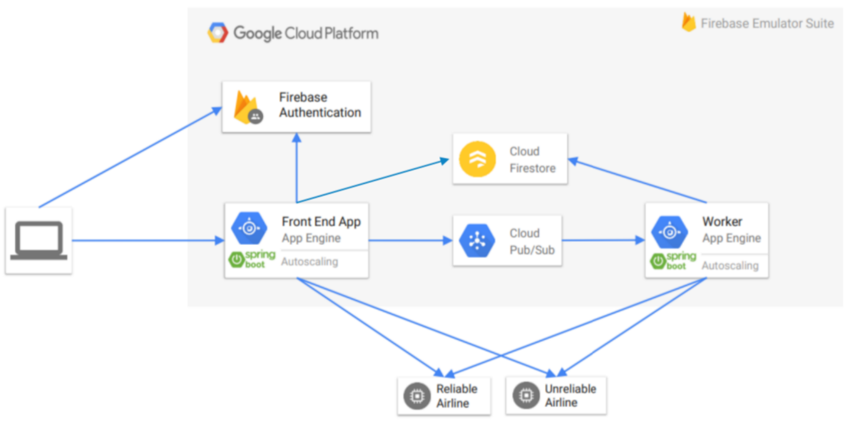

Figure 1: Deployment diagram.