Distributed Systems (H0N08A)

# *Report:* Distributed Cloud Applications

Jules Verbessem *(r0957436)*
Gert-Jan Gillis *(r0674083)*

November 17, 2023

**1. Imagine you were to deploy your application to a real cloud environment, so not a lab deployment where everything runs on the same machine. Which hosts/systems would then execute which processes, i.e., how are the remote objects distributed over hosts? Clearly outline which parts belong to the front and back end, and annotate with relevant services. Create a component/deployment diagram to illustrate this: highlight where the client(s) and server(s) are.** Als wij de applicatie zouden deployen in een echte cloud omgeving zouden wij de frontend laten hosten op een aparte server (Firebase Host, zie Figure 1). Ook zouden wij de backend opsplitsen in 2 delen. De Java Backend API die het meeste processen afhandelt. En een TicketStore Microservice. De backend gebruikt PubSub om te communiceren met de TicketStore omdat het soms langer duurt om een transactie in de TicketStore af te ronden.

**2. Where in your application were you able to leverage middleware to hide complexity and speed up development?** Als de REST Controller een request ontvangt om het winkelmandje af te rekenen (confirm quotes). Dan stuurt hij een indirect bericht naar de TicketStore controller, gebruik makend van PubSub. Deze handelt de transactie af en stuurt een confirmatie als alles in orde is. Door de functionaliteit van de booking te verplaatsen van de WebClient naar de TicketStore verminderen we de complexiteit. Ook laat het ons toe om apart te werken. Eén persoon werkt aan de WebClient terwijl de andere werkt aan de TicketStore. Dit versnelt het development proces.

**3. At which step of the booking workflow (create quote, collect cart, create booking) would the indirect communication between objects or components kick in? Describe the steps that trigger the indirect communication, and what happens afterwards.** Bij collect cart zal de REST Controller de request ontvangen om alle quotes af te handelen. Dan stuurt de Controller een indirect bericht. Er start een transactie. Alle gereserveerde seats moeten allemaal available zijn, anders worden de plaatsen niet geboekt en de quotes weer vrijgegeven. Als alle seats geboekt zijn wordt er een booking object aangemaakt en opgeslagen in de Firestore. Als de transactie beëindigd wordt, wordt er een bevestiging gestuurd naar de REST Controller en antwoordt die op zijn beurt met een response met status code 200.

**4. Which kind of data is passed between the application and the background worker when creating a booking? Does it make sense to persist data and only pass references to that data?** Er wordt een Booking DTO doorgestuurd op een indirecte manier naar de background worker. Deze bestaat uit een lijst van Quotes en informatie over de user. Het is niet logisch de lijst van quotes door te sturen "by reference". Als de communicatie tussen deze 2 componenten fragile is, kan dit een mogelijkheid zijn om minder "data transfer overhead" te hebben. Maar dit kan ervoor zorgen dat er data duplicatie en concurrency is. Indirecte communicatie gebruikt (bijna) altijd "pass by value".

**5. How does your solution to indirect communication improve scalability and responsiveness (especially in a distributed/cloud context)?** Door de scalability te verhogen, verhoog je automatisch ook de responsiveness. Als er indirecte communicatie is samen met load balancing, dan is er geen verschil met de scalability van de indirecte communicatie. Maar zonder load balancing is indirecte communicatie beter omdat het "federated brokers" gebruikt. Indirecte communicatie verbetert de responsiveness van de applicatie. Door het feit dat de applicatie niet moet wachten op de response van de subsciber, en gewoon verder kan gaan met zijn proces.

**6. Can you give an example of a user action where indirect communication would not be beneficial in terms of the trade-off between scalability and usability?** Als een user de lijst wil zien van welke treinen er zijn. Dan wil hij/zij deze direct. Het is niet gebruiksvriendelijk om de request te gaan queueën en pas af te handelen wanneer het de beurt is van de gebruiker. Dit soort request moet direct gebeuren. Als we hiervoor indirecte communicatie gaan gebruiken worden de requests gequeueëd en zal er maar één instantie van de applicatie alle requests afhandelen. Met directe communicatie zullen de requests sneller kunnen afgehandeld worden, wat dus gebruiksvriendelijker is, maar minder scalable.

**7. Is there a scenario in which your implementation of ACID properties may lead to double bookings of one seat?** Nee want als je ACID implementeert, bouw je Atomicity in. Wat ervoor zorgt dat als één seat reservation failt, ze allemaal moeten failen. Waardoor het niet mogelijk is om één seat in meerdere bookings te hebben. De opgave stelt dat de Train Company's correct kunnen omgaan met gelijktijdige reservaties.

**8. How does role-based access control simplify the requirement that only authorized users can access manager methods?** Door role-based access control kan je specificeren welke endpoints niet bereikbaar zijn voor gebruikers zonder de manager-role. Ook zorgt dit ervoor dat je in de frontend sommige pagina's niet zichtbaar maakt voor gebruikers zonder de manager-role. Als je niet met rollen zou werken, zou je een nieuwe applicatie moeten maken met enkel de functionaliteit van de manager, waarbij enkel die kunnen inloggen. Dit is veel werk.

**9. Which components would need to change if you switch to another authentication provider? Where may such a change make it more/less difficult to correctly enforce access control rules, and what would an authentication provider therefore ideally provide?** De authentication provider controleert de user credentials en geeft een JWT token terug. Deze wordt gebruikt in de frontend om door te sturen naar de backend als er een API request is. In de backend wordt deze token gedecodeerd en worden de user credentials eruit gehaald. Als we van authentication provider gaan wisselen heeft dit geen effect op de backend, zolang er maar een JWT token wordt terug gegeven die gedecodeerd kan worden. Idealiter zou de authentication provider de complexiteit van het inloggen/aanmaken van een gebruiker moeten overnemen van de applicatie.

**10. How does your application cope with failures of the Unreliable Train Company? How severely faulty may that train company become before there is a significant impact on the functionality of your application?** De Unreliable Train Company geeft soms een 429 (Too many request) of een 500 (Server Error) terug. Onze applicatie laat toe om de data van de Reliable Train Company wel te tonen als de Unreliable Train Company een fout geeft. Er kunnen in dat geval ook enkel maar seats gebooked worden voor de Reliable Train Company. Wanneer de Unreliable Train Company terug werkt, dan zal terug de data van beide getoond worden. Wanneer de gebruiker op de pagina staat om een stoel te boeken bij de Unreliable Train Company. En deze dan een probleem geeft. Dan zou de applicatie moeten redirecten naar de Home pagina. Waar de gebruiker dan enkel de data kan zien van de Reliable Train Company. Maar hiervoor moeten we de frontend kunnen aanpassen. Wanneer er tickets worden geboekt en het boeken van een ticket bij de Unreliable Train Company failt. Dan gaan we de eerder gemaakte

bookings terug verwijderen. Heel de transactie failt. De gebruiker zal dan opnieuw zijn stoel plaatsen moeten reserveren op de applicatie.

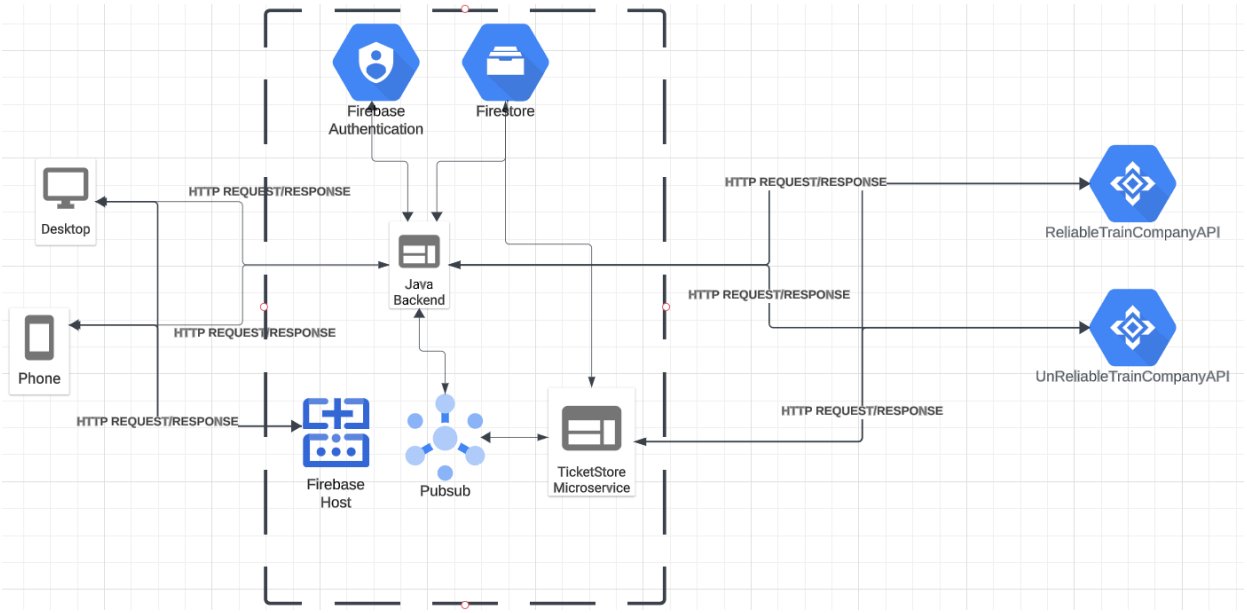


Figure 1: Deployment diagram