Software Integration in a Trailer Rental System Part 2

Scenario: MyTrailer is a company that rents car trailers so that people can move things or take garden waste to the genbrugsplads. The trailers are owned by the company but are placed in locations such as Jem og Fix or Fog so that they are always close to where people want them. The trailers carry both the MyTrailer and the location's branding but all the rental and payments are done via the MyTrailer system. In return for parking the trailers on their site the companies who give space to MyTrailer get their advertising on the trailers and pay MyTrailer for providing the loan service (which helps attract customers to their stores).

Customers access the trailer system over an app. They can select a location and book a trailer. All of the trailers are identified by a location ID and a number (1,2,3,4 etc). The user books a specific trailer. Rentals are for a maximum 24 hour period but always finish at midnight at the latest. This means a trailer rented at 9pm could only be rented for a maximum three hours. Overnight rental counts as long-term rental and needs to be booked from the website and collected from specialist locations and is not part of the app. They are not part of the normal MyTrailer offer which is for short term rental. If a trailer is returned late there is an excess rental fee added to the customer's bill. The fee charged for a trailer is zero but most (over 80%) customers buy insurance for the trailer that costs 50 Kr. MyTrailer makes most of its money from the company partnerships and the insurance fees (the number of claims is very small).

Tasks: You are being asked to implement the system using the domain driven design approach. This may include using requirements gathering techniques you have learned and Enterprise Architecture design ideas. It should include at least two services and include documentation about the codebase. It is recommended that you develop a monolithic application that you split into services. The application does not have to be production ready

- Using the bounded contexts you have already designed
- Show how you used your model as a base for the implementation of your system– use https://ddd-practitioners.com/home/ as a guide.
- Prepare documentation to show how you built the system – for example how the ports and adapters pattern was implemented.
- Make sure you have an iteration of the system that can be demonstrated
- Email me a link or upload it to the upload folder for OLA 4 of the github repo containing your implementation.

  Note: you can design the system in your own way just be sure to document the tech stack and the application architecture. We have not yet looked at message brokers or event driven architecture but feel free to use tools like Apache Kafka and/or RabbitMQ in your application and any serialisation/deserialization libraries to handle streams of data.