System Integration
Ola 3
Domain Driven Design Part 1

By Kristofer, Mads, Michael and Søren

Event storming session:

In this section, will we go over the event storming session we held where we vent over the business case scenario. We used a Discord whiteboard activity so that we could, as a group, set up sticky notes based on what we discovered about event storming during our research on the topic.

We started by going over the business case and first set up a couple of domain events based on what we could read. We then started expanding on the event by filling out the relevant command and the actors making them. Then we added policies/invariants to expand on the requirements for the commands/events. Finally we added domain events, and filled them out, based on what we saw as requirements for the system to work.

The snapshot below shows our final result. You start at the top left corner, going left to right and one row at a time until you reach the end at the bottom row “Process payment” domain event.

Domain events/Commands:

To help explain our events/commands and what revolves around them, will we take each command and resulting domain event.

App startup Data retrieved:
This command is automatically given by the system when the application starts up and triggers an event to get the data regarding the available locations and trailers from our backend and database.

Select location - Location is selected:
When the data has been loaded and made visible to the user, then the next part of the process is for the user to select what location they would like to see the trailers for.

Select trailer - Trailer is selected:
When a location is selected the user will then need to select a specific trailer that they would like to book.

Book trailer - Trailer is booked:
When a trailer has been selected the user can book the selected trailer, but an invariant has been created that results in only trailers that have not already been booked can be booked.

Add insurance - Added insurance:
When booking a trailer, the user can choose to add insurance to the booking.

Return trailer - Trailer is returned:
When the user is done using the trailer they booked they can return which will remove the booked status from the trailer.

Apply late fee - Late fee is applied:
A policy has been made where if the booked trailer is returned the next day, no matter the time between booked and return, the system will trigger and apply a late fee.

Pay - Process payment:
The user will pay for the booking using a selected payment method.

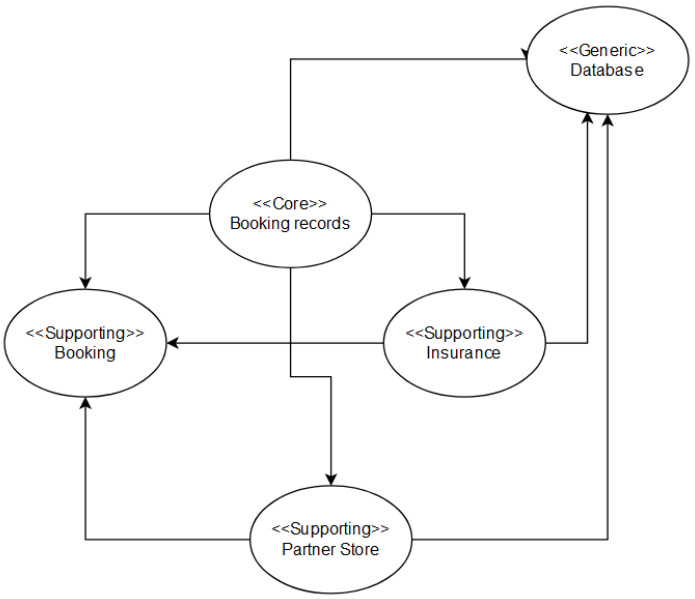
External systems:

We came to two different external systems that the application would need. We view external systems as systems we have no direct control over, so any database we ourselves set up is not considered an external system. Therefore the two external systems are the insurance companies system and the payment system. It makes sense that selection of insurance is made alongside the booking of the trailer but it's not specified that two different payments are to be made during the process, on booking and on return, we have therefore only one payment in the end of the flow, with the insurance being added to the booking. The business case makes reference to a browser website that we suspect would make use of the same database for the locations and trailers as we should, but since it was not specified, we decided the database should be considered part of the application.

Strategy design:

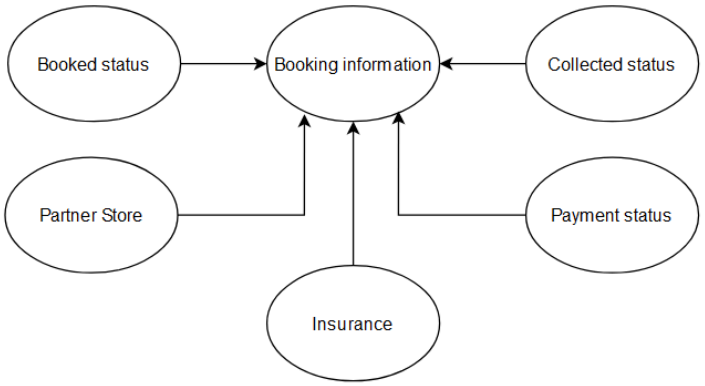
Identifying and classifying subdomains:

We have identified five different subdomains based on our event storming session.
The five being: Booking records, Scheduling, Database, Insurance, Partner store.
They are based on the process we create during the event storming with the booking records being our core subdomain as our entire application revolves around the booking of trailers.
The database is a generic domain as it's not specific to our organization. While both the insurance and partner store subdomain refers to third party organizations, they will contain specific information that we will store and retrieve from the database.
Lastly there is the booking subdomain which is necessary for the organization to succeed but isn't core and it isn't generic because it requires some specialization to work.



Bounded Contexts:

With bounded contexts we take a closer look at one of the subdomains we have identified, as making a big ambiguous model of the entire application and our organization, so that we gain a better understanding of what the subdomain model actually represents.
We should note that they aren't meant to be one to one as the subdomain model could be considered part of the problem space and the bounded contexts belong to the solution space.



Ambiguous language:

Booking:
Refer to the same aspect of our system. In the description of the business and the scenario there are references using “booking”, “rening” and “loading”, but they all refer to the same thing that we have chosen to mark as booking. The system revolves around being able to select and mark a trailer as booking in our system. How the customer interacts with the trailer physically before, during and after booking is not part of our work area.

Partner Store:
Refers to a variety of physical stores wherein one of their own customers may need to transport their goods. While we refer to them generally as partner stores, they are called company partnerships in the business case. They are not part of the company MyTrailer and how they profit or any other agreement made between MyTrailer and the various partner companies are not part of our work area.

Insurance:
Insurance refers to an optional fee that the customer may apply doing the booking process for 50 kr. It is not specified if the insurance comes from a third party insurance company or if the insurance is handled directly by MyTrailer. We treat it as if the insurance comes from outside the system. This way it can work both with MyTrailer and any third party insurance company at the same time.

Late return:
When the customer books a trailer they can only do so for the rest of the day. If they book the trailer kl. 09:00 or 18:00 and return it the next day at the same time, 10:00, it has no effect on the fee. The fee is solely based on the date difference from booking to returning.

Tactical design:

We have drawn an aggregate diagram based on our event storming session.
Within our diagram we have four different aggregates, all containing entities but not all containing value objects.