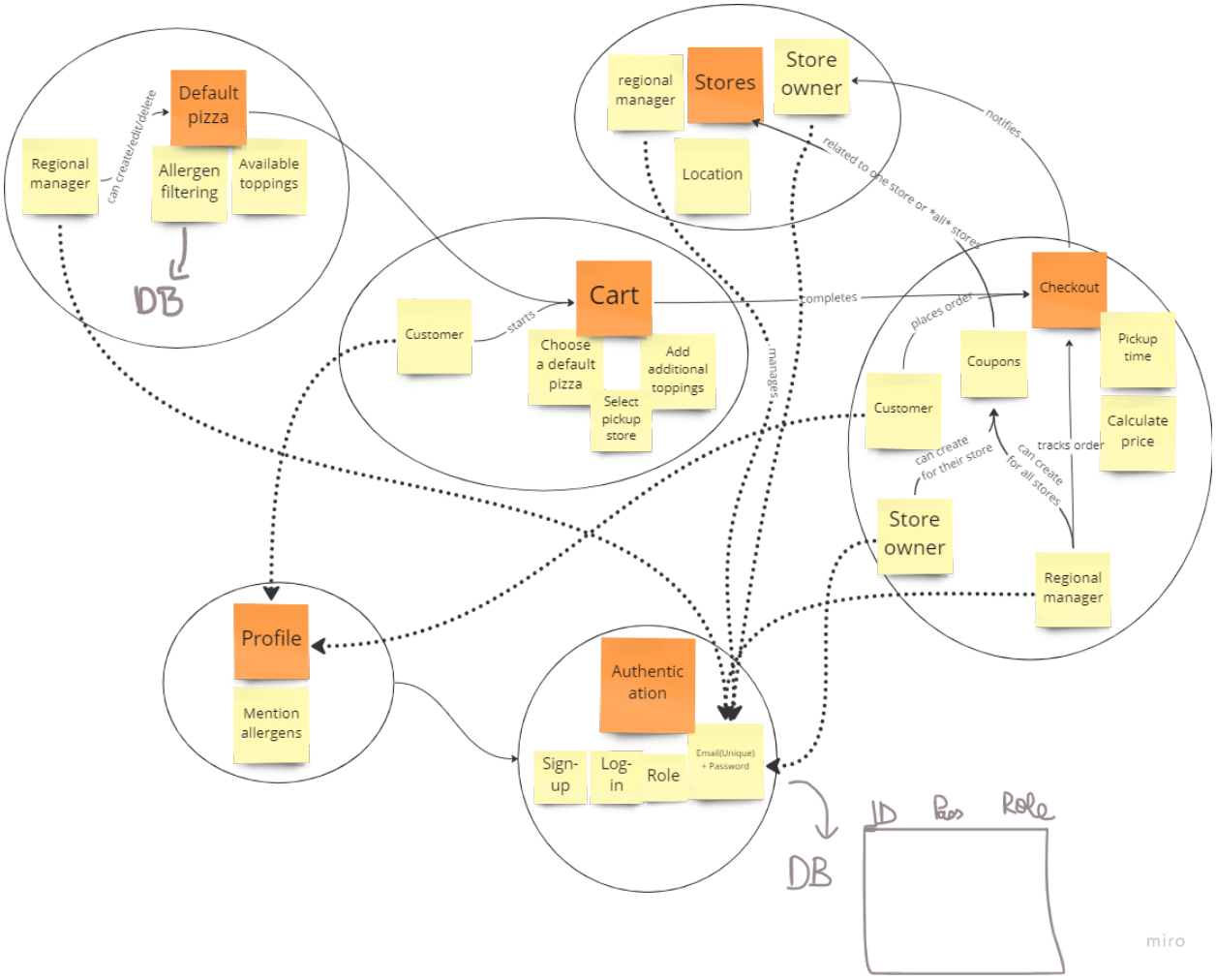
Task 1: Software Architecture

Bounded contexts



Straight lines indicate relations between bounded contexts, as well as relations between objects within a bounded context. Dotted lines represent objects in a bounded context that come from another bounded context.

Authentication

The authentication bounded context should allow customers to create an account with an email and a password. It should generate a JWT token that takes into account the role of the user. The JWT token will be used to access all the other endpoints in the API. Users can be normal customers, regional managers or store owners.
The authentication service should also persist users to the database. Because of security issues we do not allow regional or store managers to create accounts, their accounts will be prefilled in the database manually when we start the application. The authentication microservice interacts

with all the other microservices in order to verify the identity and permissions of users. It is a generic microservice.

Customer

The customer bounded context contains a list of allergens and a list of used coupons. When registering an account, the default role will be *customer*, as regional managers don't go through a normal registration process, they will just log in and this is taken care of in the Authentication bounded context. The allergens are a list of toppings that a user can complete, which can later be used to filter the available pizzas to exclude the ones which contain the aforementioned allergens. The used coupons list is a list of coupons that the user has already used at checkout. This list gets updated when the checkout is done and the order is completed/placed by the customer, then if a coupon has been used, it is added to the customer's list. If an order is canceled, either by the user or the regional manager, the coupon is removed from this list, as it is refunded to the user.

Store

The bounding context of the store contains a corresponding regional manager, a location and a store owner. The Store will be linked to a regional manager in a n-1 relation. The ID of the manager will be saved in the store object, to verify whether the person has permissions at this store. This way we can make sure that only the regional manager is able to make changes.
The store also contains a location where customers are able to pick up their pizzas.
Finally the store will be linked to a store owner in a strict 1-1 relation, the store owner is able to add coupons that only work on their own store.

Default Pizza

We decided on the default pizza bounded context because there are a lot of requirements that relate to setting up toppings and default pizza's. The regional managers should be able to manage the available toppings in the store. This includes adding, updating, and removing toppings. For the toppings we will need to know the name of the topping and the price it costs to add it to your pizza.
The regional manager should also be able to create, edit and remove the default pizza's. A default pizza simply consists of a set of toppings that are on the pizza.
This bounded context also contains the functionality for the customer to retrieve a list of the default pizza's and the pizza toppings that are available. When retrieving either the list of default pizza's or toppings, the customer should have the option to filter out results that contain an allergen as mentioned in their account.

Cart

The bounded context of the Cart consists of functionality related to the process of making an order from the user's perspective. This involves every step of the ordering process prior to checkout. Specifically, Cart first provides the customer the choice of pickup location, where they can choose the store from which they'd want to order from. Following this selection, Cart handles the customer's ability to select a default pizza from the store's selection or create a custom pizza by adding a custom configuration of toppings (selected from a list of toppings the store provides which is the same for all stores). The customer can edit/remove any and all toppings on any pizza (custom or default). The Cart also involves the option to filter the default selection of pizzas the customer sees based on allergens that they have set in their profile, in order to make the order process easier for such dietary restrictions. The actual filtering logic happens in the Default Pizza bounded context which has access to the database of default pizzas, therefore the Cart interacts with the Default Pizza microservice in this way. The customer should also get a warning when choosing a pizza that contains a topping they have listed as an allergen, regardless if they have filtered the selection of pizzas or not. This is done through the interaction of Cart and Customer, where the allergens are listed for the specific customer.

Checkout

In the bounded context of checkout all the logic and management before the completion of the order is held. We decided to separate this part of making an order from the actual order context, as there are multiple aspects to be managed from all the user's, the store owner's, the regional manager's and the application's side.
Firstly, the user has the option to select his pickup time for his order at the store he earlier selected.
Additionally, the functionality for coupons for both sides is included in these. Specifically, the user can try out multiple coupons to reduce the price of his purchase. The coupons are either specific to the selected store, or exist in the whole region the store belongs to. The application will be responsible for checking whether the requested coupon is valid at that point in time, and also whether the specific customer can make use of it, since all users can only apply a given coupon code once. Furthermore, it will have to apply to the order only the coupon code that alone gives the lowest final price, the biggest discount. In case the order gets canceled, the coupon will be refunded to the user, meaning he can apply it to a future order.
Finally, during checkout the customer can view the price of his order by seeing how much each pizza cost, and the coupon's contribution. From the regional manager's perspective, the ability to see, track, edit, or cancel any incoming orders assigned to him is provided. When checkout is completed, the store to prepare the order gets notified and receives all the relative information.

Mapping to microservices

To map the bounded contexts to microservices, we decided to decompose by subdomain.
This follows naturally from the Domain Driven Design principle that the program should adhere

to. The reason we made this decision was to have each microservice be relatively autonomous, and to deliver a different logical part of the project.
Each bounded context is mapped into a microservice of its own. Thus, we have 6 microservices:

- Customer → Customer microservice
- Authentication → Authentication microservice
- Store → Store microservice
- Cart → Cart microservice
- Default Pizza → Default Pizza microservice
- Checkout → Checkout microservice.

Microservice Interactions

Authentication - Customer
Authentication - Stores
Authentication - Default Pizza
Authentication - Checkout
Authentication - Cart

Cart - Default pizza
Cart - Customer
Cart - Checkout

Checkout - Store
Checkout - Customer