Assignment 1 (Draft)

Our architecture consists of 5 bounded contexts, mapped to 5 microservices. These are the following: Authentication, Users, Basket, Coupons and Orders. We will now discuss each of them in that order.

Authentication

The bounded context for authentication is responsible for registering new customers and authenticating existing users, including stores and regional managers. The functionality is provided by the microservice Authentication Service. The bounded context is mapped 1-to-1 with the microservice.

This microservice will be used upon entering the program/gateway. Users are given a choice to either register or log in. Creating a new account is only available for customers, as the creation of store and regional manager users cannot be completed without the rights and other details required.

A connection with the user database will be made to verify the username-password combination. In case no user with such a username exists, the user will be prompted for invalid credentials. Otherwise, a token will be sent back to the user if the authentication is done successfully. From now on, every time the user interacts with a microservice they will send this token along.

Users

User microservice is used to store and deal with all data that are user-related, such as the user role and allergenic ingredients (in the case of *Customer*). Each user will have a unique *id*, which will be used to retrieve information about them. We will determine through the id which role the user has:

- The *Customers* will have **randomly generated numbers**;
- The *Stores* will have **an S prepended to randomly generated numbers**;
- The *Regional Manager* will have **an R prepended to randomly generated numbers.**

Upon creation, Customer will be asked to select from a list of allergies. These selections will be stored in the User's microservice database as a *List[String].* The method **getAllergies()** retrieves the user's allergies:

- If the user has the role *Store*, this list will be empty.
- If the user is a regional manager, this list will be used to keep track of the stores that the regional manager manages.

If a customer chooses not to add any allergies, they are considered to not have any allergies. If the user has already created his/her account and has forgotten to add all allergies, he/she can still do that using the **addAlergies()** method, which will bring him/her to the selection list again.

The microservice will also store the user's username and password that are used for login.

Basket

The Baskets microservice is used to populate the basket of the customer. It contains *Pizza* entities with their ingredients in its database, listed as *List[String]*. The basket is created through **createOrder()** and the allergens of the customer are requested by the microservice from the User microservice with **getAllergies().** The allergies can be used in two ways based on customer preference:

- Filtering out pizzas with allergenic ingredients before the display
- Checking the selected pizzas for allergens before payment

The basket of the customer is editable, meaning a new pizza can be added, or a pizza can be removed or edited after they are put in the basket before the order is confirmed. The ordering functionality is located in this microservice, and therefore this microservice is the last microservice used by the customer as they select their pizzas and finalize their order (can include additional *coupons*).

Coupons

In the Baskets microservice, a *coupon* can be applied. Coupons are another microservice that searches for valid coupons in the database and calculates the new price of the basket. There are two main functionalities that this microservice provides:

- applying an existing coupon
- adding a new coupon

Coupons microservice mainly works with *Coupon* entities. Class Coupon has:

- the *activationKey* as the key attribute,
- a Boolean field to check if this coupon should be expired after use,
- and the method that calculates the new price after the coupon is applied.

The format of the activation key is 4 alphabetical letters followed by 2 digits, and the specific letters in this key imply the type of the coupon (i.e., discount, buy one get one free, etc).

The coupons are stored in the database, and when the user fills in the activation key, the specific coupon is retrieved from the database. Depending on the type of coupon, the new total price of the basket is calculated differently. A user can also submit another coupon after one is applied. In this case, the new price using the new coupon is calculated and compared with the old price. The coupon with a cheaper price is taken.

For adding a new coupon, a regional manager or worker can give the unique activation key and choose the type of coupon, and the new coupon is stored in the database.

For the combination coupons, The worker/manager can specify which combination of menus this coupon can be applied for. Combination coupon also stores this list of menus, and when a customer uses this kind of coupon the coupon checks if all the menus in this list are in the basket. If they are, it calculates the new price and displays it; if not, it displays an error message.

Orders

The Orders microservice is responsible for handling orders that have been submitted through the aforementioned *Baskets* microservice. The main functionality is to direct the flow of orders, in contrast to Basket which composes the orders. The requests it can receive are related to the flow of orders:

- Enqueue order
- Finish order
- Cancel order
- Show order

To submit any orders, the Enqueue functionality is used. This receives an order and adds it to the database together with an order id. The collection of orders is then accessed by using

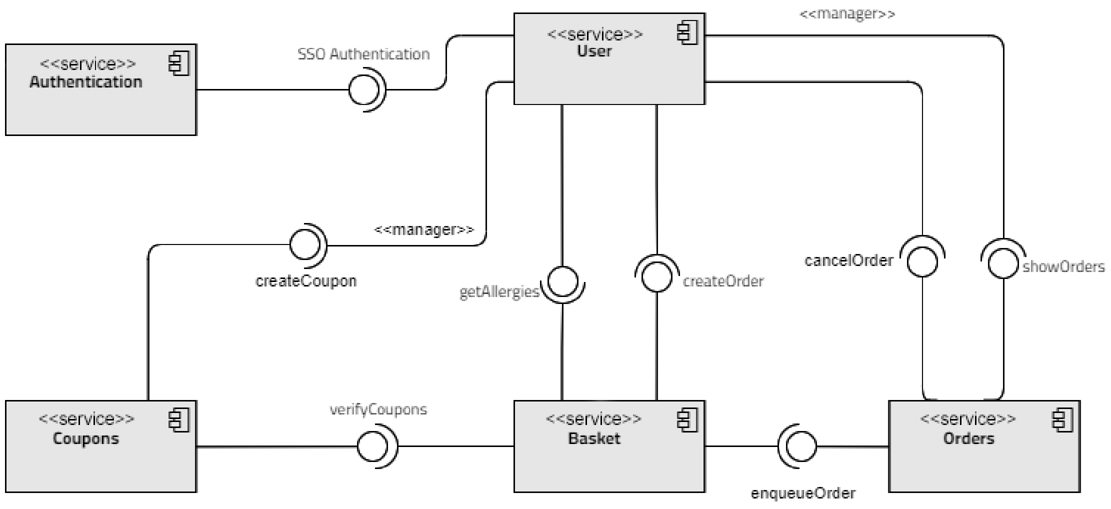
”Show order”. All types of users are authorized to make use of the function, but all get different results:

- **Customer** can see their own order, change anything before submission, and cancel their order until the last 30 minutes before delivery. They can see their order until the delivery.
- **Store** can see all orders(baskets) that are made by customers that selected them as their store of preference. They can check orders off if they are sent.
- **Regional managers** can see all orders made to the stores that lie within their region. They can cancel any order.

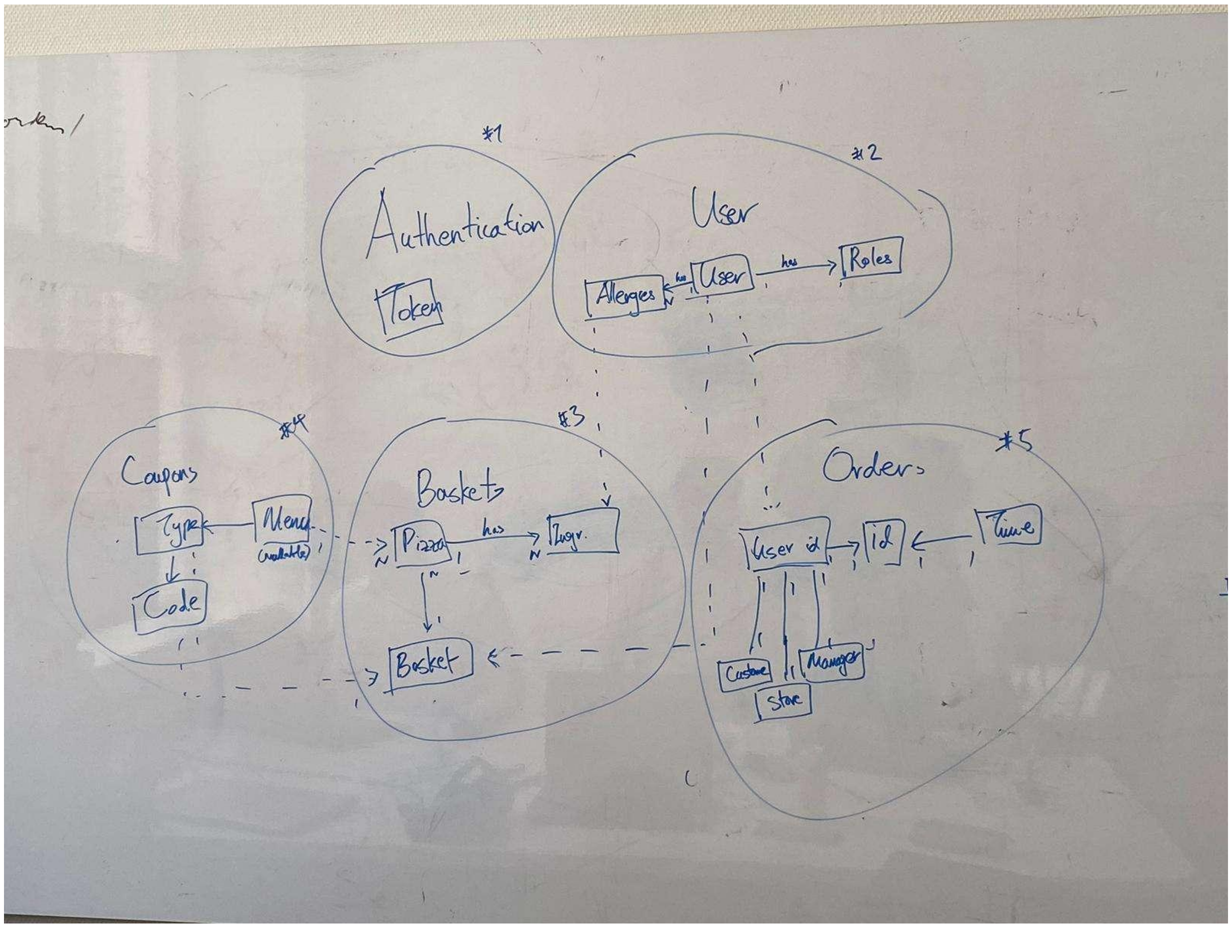
Both canceling and checking off an order remove the order from the collection. In the case that a regional manager cancels an order, the user that submitted the order will be notified.

Diagrams

The following UML component diagram describes our overall architecture, as well as all the interactions between the different microservices:



This context map also provides further insight in the user flow of our application:



#1
Authentication
Token
#2
User
Allerges
has
User
has
Roles
#4
Coupons
Type
Menu
(available)
Code
#3
Baskets
Pizza
has
Ingr.
Basket
#5
Orders
User id
id
Time
Custome
Store
Manager