Assignment 1 part 1

CSE2115: Software Engineering Methods

Group 06b

Assignment 1 part 1

by

Group 06b

Instructor:	Dr. A. Panichella & F. Mulder
Teaching Assistant:	M. Segers
Institution:	Delft University of Technology
Place:	Faculty of Electrical Engineering, Mathematics and Computer Science
Date:	16-12-2022

Summary

This report provides an overview of the proposed architecture for Anni's Pizza online portal, which is further explained in Part 1, and also highlights two design patterns that were used when building the portal, which is elaborated in Part 2.

Part 1

In order to properly determine the architecture of the system, we first wrote the requirements from the scenario using the MoSCoW method, then extracted the bounding context from the requirements. We have decided to have all the bounding contexts as microservices, each microservice having a clearly defined role and was carefully chosen to take into account the scalable parts of the system. The interaction between microservices is explained in the last section of this part.

Contents

Summary **i**

1 Bounding Contexts **1**

2 Microservices individually explained **3**

3 How they work together **6**

List of Figures **7**

1
Bounding Contexts

The following core bounding contexts were identified: **Users**, **Menu**, **Stores**, **Orders**. **Authentication** was identified as a generic domain context, while **Coupons** being a supporting domain.

The core bounding context **Users** contains the information about specific user's allergies. It is upstream to the Menu context as it is important for the Menu to be aware what are the users allergies such that it can filter the list of pizzas for the user. Users will be queried for allergies by Menu, so that it can warn the Order if an item with allergies has been added so that Order can warn the user that is placing the order.

The core context **Menu** contains information on the available assortments of pizzas and configurable toppings for each given pizza. It also contains the list of possible allergens for each item on the menu in order to filter out information based on User requests. It is downstream to the Users context as it receives the allergies of the user sending the request. It is upstream to the Order context as it provides the context with the verification and pricing of items from the menu that will be stored in the user's "shopping cart".

The core context **Stores** contains information about the stores in the chain and is able to notify the stores on events. Each store has a location, a name and an inbox to simulate dummy emails. This context can be queried to present the list of the locations available to place an order to.The Stores context is in bidirectional stream with the Order since Stores receives and stores messages about the orders of the respective store and Order queries the Stores to see if the selected location is valid.

The core context **Orders** handles the information about the currently active orders. For each order there are 4 identified stages: **ongoing**, **placed**, **finished**, **canceled**. While the order is in the ongoing stage new pizzas and coupons can be added/edited/removed from it. This context is downstream to the Menu, Users and Coupons contexts. From the Menu it validates a pizza and gets the price of it. From Users it requests the preferred location if the location has not been provided in the order request. To Coupons it hands over details of the order such that the coupons can be verified and can be applied to the order in an optimal manner, it receives back the price and the coupon applied. Order is in bidirectional stream with the Stores, since the Order gets the validation if the store selected is a valid store and the Store gets a notification when the order is placed or there are any other changes to the order.Once the ongoing stage is done the order is considered placed and cannot be edited anymore. The user that placed the order can still cancel their order up to half an hour before their selected time, after this the order can no longer be canceled by the user.

Admins/regional managers can cancel the order up to the selected time. A successful order (that goes through all stages) will be marked as finished, otherwise it will be marked as canceled.

The generic context **Authentication** handles the authentication of the user within the system and provides them with a token that contains their privileges within the system and uniquely identifies them.

The supporting context **Coupons** is upstream to the Orders context, it contains the currently available coupons and the information about what each coupon entails (its effects, expiry time). It receives the information about the order (coupon codes and prices of items ordered) and decides which coupon to apply for the best price and what is the price after application. It also serves to verify the coupons added to an order. Coupons can be used by the customers and added by admins.

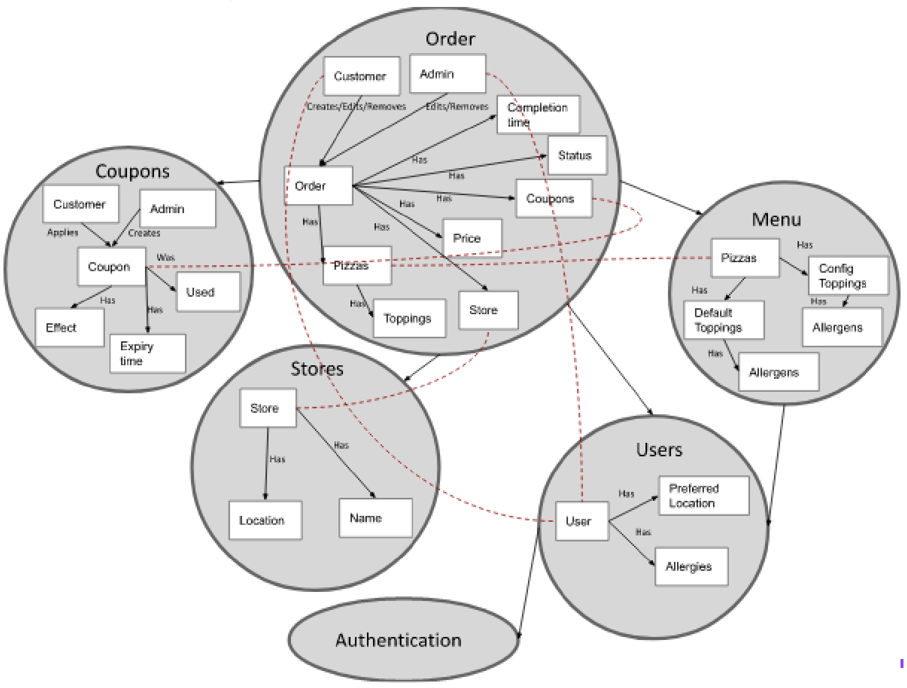

Figure 1.1: Bounding context diagram

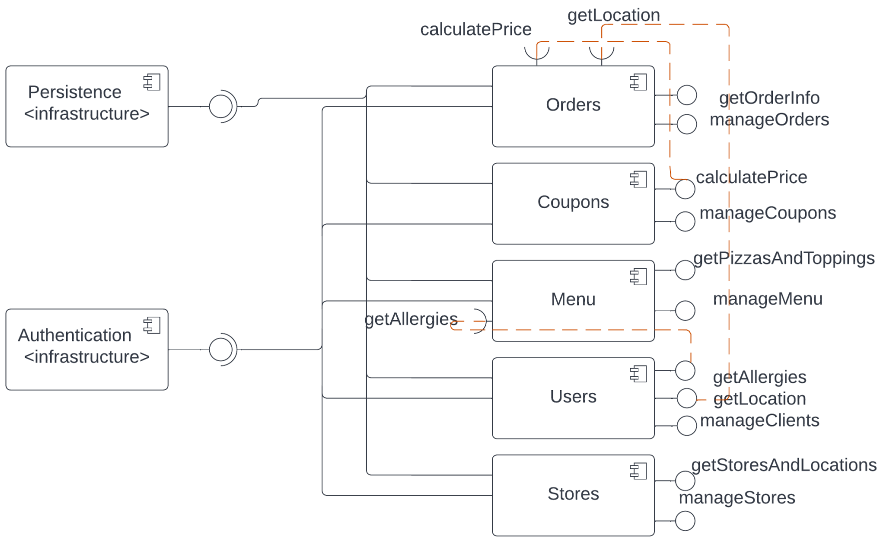

Figure 1.2: UML diagram

2
Microservices individually explained

1. Authentication

The Authentication microservice is responsible for the security of the application. Anyone will be able to perform the following operations: account registration, authentication, and changing passwords. The regional manager can also add roles and change the roles of different users.

The role of this microservice is to enhance the security of the service by reducing the risk of user impersonation, while also contributing to creating role-based access to the system.

We have decided to have a separate microservice for authentication to allow an increase in the security of the service (e.g. adding more proxies just for this part) without adding overhead to the rest of the business logic.

2. User

The User microservice keeps track of allergies that the user has and the location of their preferred store. The user is authenticated using the token given from the authentication microservice. The allergens are queried by the menu and order microservices in order to filter out items or notify the user that they might be allergic to the item.

The User microservice takes in a token that is used to fetch the user's information from a database (by using the memberID provided by the token).

We have created a microservice for the users because we feel that it is best to store sensitive information about a customer in a secure way (using authentication). In this way the allergens and the preferred location are stored separately from the username and password of a user.

3. Menu

The Menu microservice provides all available assortments of pizza that are available for ordering, while allowing configuring items with customizable toppings and also filtering out or warning users in case of allergens present.

The role of this microservice is to provide the users with the menu of the store chain, so they know what they can order from and also store pizzas, custom toppings for pizza types, and all the different allergens to make sure its customers are safe. The regional manager can also add or remove pizzas or toppings from the menu.

We have created a microservice specifically for the Menu to better encapsulate what people can order from the actual ordering business logic. Moreover, it allows the restaurant to scale better in case they want to extend their assortment to more than pizza, while also providing a way to order from them while the Menu microservice does not provide the desired output (eg. it is being updated or has an outage).

4. Order

The order microservice is responsible for handling orders. It stores ongoing, placed, finished and canceled orders in a database.
The order microservice also collects order details. First it collects the order time and location. If the location is not provided the Order will query the Users for the preferred location.

When the pizzas are added to the order the Order queries the Menu to see if the pizzas are valid. If any of the pizzas contain an allergen the Menu will indicate that to the Order and the Order will warn the user.

If the user applies coupons, the order is sent to the coupon microservice and gets back the final price of the order.

The order can be edited in the ongoing stage by the user. Once the user places the order the order can no longer be edited, however the user that placed the order can still cancel their order up to half an hour before their selected time, after this the order can no longer be canceled by the user. Once the order is placed or when there are changes to the order the respective store is notified. Admins/regional managers can cancel any order up until the order is finished. Admins/regional managers can also see all the orders.

The role of the Order microservice is to encapsulate all business logic relating to an order, while also keeping track of the history of operations and also of the current shopping carts, allowing the user to still have their data after an outage in the system.

The reasoning behind this microservice is to allow separation of ordering related logic because it needs to scale fast if Anni's pizza will be a successful business, so everything in the order microservice must be able to scale (from the maximum QPS it can handle to the bandwidth and real-time storage) without affecting the other microservices.

5. Coupon

The Coupon microservice is responsible for validating coupon codes and for choosing the best coupon code for minimizing order type. This microservice allows new coupon codes to be added and also to take a possible order and to check if a coupon code gives a better price and return the new price.

The role of this microservice is to contain the business logic for coupon codes and also store the available coupon codes.

We decided to have a separate microservice for handling coupon codes because over time multiple types of coupon codes might be added to the service. If the coupon logic is encapsulated outside of the system it will improve deployment times, reliability and robustness. If something fails when adding the logic and it makes the Coupon service crash, Ordering will still be possible so it would not affect that many consumers. Moreover, having a separate system with the business logic will make the system easier to test.

6. Store

The store microservice is responsible for keeping track of possible store locations. This microservice allows the regional manager to add new or remove existing stores. The store microservice is also responsible for encapsulating dummy email functionality.

The role of this microservice is to provide users a list of stores in which they can choose one to collect their pizza. It also provides the possibility for the admin to check notifications regarding orders in their store. The regional manager will be able to check all notifications from all stores.

We have decided to keep Store in another microservice to have a better overview for each individual store, while also allowing the users to order from their preferred location or to an already-known location in case of failure of the Store system.

7. Gateway

The gateway microservice is responsible for propagating requests through to the system. Its role is to hide the ports the microservices are running, providing extra security against cyberattacks. Moreover, it allows Anni's Pizza to handle a lot of clients, allowing load balancing and having multiple server locations.

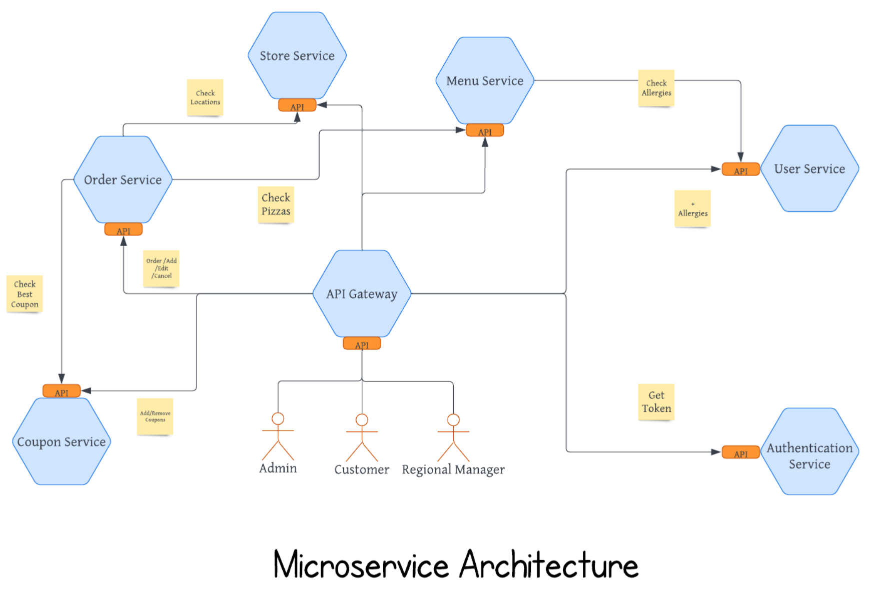

Figure 2.1: Microservice architecture

3
How they work together

Through the system, at this point there are 3 possible roles: customer, store manager and regional manager. Each role has the same or more privileges than the previous (store manager has customer privileges plus its own and regional manager has store manager privileges plus their own). Each role privileges are elaborated in the next part of this section.

All the microservices propagate through the gateway. A customer will use the Authentication microservice to create an account and receive a token to authenticate themselves. Once they have logged in, they will be able to set up their allergens and their preferred store location through the User microservice. The customer will also be able to select at which store they would prefer to collect their pizza through the Order microservice. The Menu microservice will help them fetch all available pizzas from the pizza database so they can decide which pizza to order. Pizzas that contain the allergens can be filtered out during the fetching process. The customer will be able to add pizzas to the cart through the Order microservice and the system will notify them if there are any allergies. Furthermore, they can also use some coupons, which is done through the Coupon microservice. After making their choice, they can edit or remove pizzas from their order through the Order microservice before they send their order. Once they have determined everything, they will check out through the Order microservice again. If they choose to cancel their order, they can do so 30 minutes before the selected time and the system will notify the store when there are any changes.

An admin will use the authentication service to create an account and receive a token to authenticate themselves. They can edit the set of coupons through the coupon service by adding new coupons or deleting existing ones. In addition, they are able to manage the existing orders, and if needed, they can cancel them. This is done through the Order microservice.

A regional manager will use the authentication service to create an account and receive a token to authenticate themselves. They will be able to see all the currently placed orders and cancel any selection of orders if needed through the Order microservice. Moreover, they can see the inboxes of every single store in the chain to have a better grasp on every single store event through the Store microservice.

List of Figures

1.1 Bounding context diagram 2
1.2 UML diagram 2

2.1 Microservice architecture 5