Microservice Mini Project

A microservice based solution for an online retailer

This mini project is a compulsory assignment in System Integration. Your solution must be approved by the teacher before you can attend exam.

Background

An online retailer, who sells various products over the Internet, has asked you to develop a new microservice-based solution for the back-end. Before they commit themselves to invest in a fully functional solution, they want you to develop a "proof of concept".

Before customers can submit orders, they must register themselves with information such as company name and registration number, email address, phone number, billing and shipping address. After the initial registration, the customer should be able to change email address, phone number, billing and shipping address. Customers should at any time be able to get an overview of their past orders (incl. the order status). An order which have not yet been dispatched (i.e. shipped) may be cancelled by the customer on request.

An order may consist of several items of different products. For each product, the following information should be available: Product registration number, name, price, category, the number of items in stock, and the number of items reserved. When a new order is created, the number of items ordered for each product are reserved in the system. Later, when the ordered items are shipped, the reservations are removed, and the number of items in stock are decremented for each ordered product. Because customers pay on delivery, no credit card information is needed.

When the online retail system receives an order, the following actions should take place:

- If the customer does not exist, the order should be rejected, and the client, who sent the request, should be notified.
- The retail system should check the customer's credit standing and the inventory level before accepting the order.
- If the customer has outstanding bills, the order should be rejected, and the customer should be notified.
- Similarly, if some items in the order are not in stock, the order should be rejected, and the customer should be notified.
- If the customer is in good credit standing and the order items are all in stock, the order should be accepted, and an order confirmation should be returned to the customer.

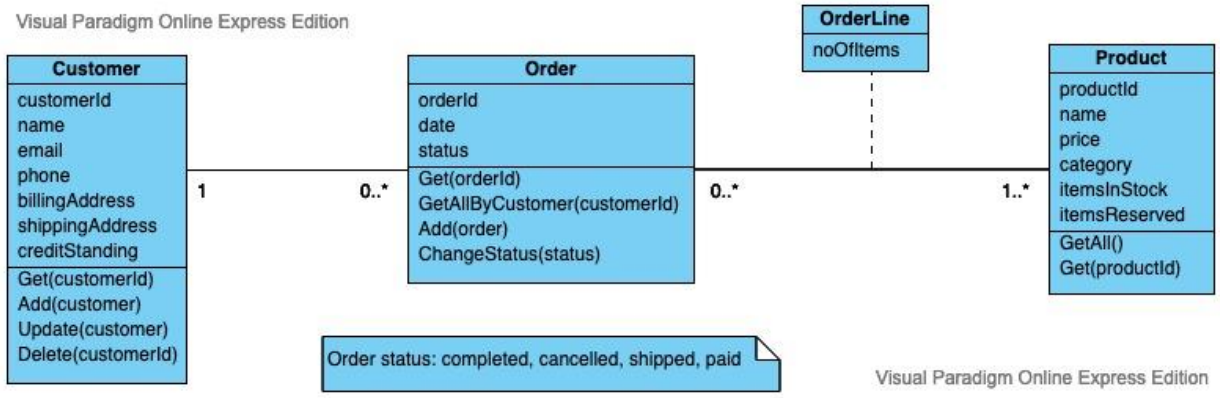
Because we are aiming at an initial solution with a limited scope, no login or shopping cart is required, and we will assume that an order is shipped as a single package.

When the order is shipped, an invoice should be emailed to the customer. You are not required to implement real email functionality. A mock is sufficient.

Part 1: Design and implementation (Week 8)

You should only develop the online retailer's part of the solution (i.e. not the front-end). The solution must be based on RESTful web services. You should use a *microservice-based* approach. Use ***bounded contexts*** to ensure that the coupling between services is minimal.

Here is a model of the system that you can use as a starting point:



I have published a partial solution, named "OnlineRetailer_Partial" on GitHub:

https://github.com/henrikkyhl/OnlineRetailer_Partial

I recommend that you use this solution as a starting point. It implements a significant part of the Product and Order services, although the design could and should be improved. You will also see that the partial solution uses an in-memory database. I strongly recommend that you continue to do that instead of using a SQL Server database. Otherwise, it will be more difficult to deploy the solution using Docker.

Part 2: Deployment using Docker (Week 9)

You should deploy the solution as a multi-container solution using Docker.

Part 3: Use messaging for inter-service communication (Week 10-11)

You should refactor the solution so that the microservices communicate with each other using messaging, when it makes sense.

Terms and conditions

It is highly recommended that you solve the assignment in teams. A team size of 3-4 members is recommended.

You should demonstrate your solution for me Friday, Week 12. Each team has 10-15 minutes for demonstration. You should demonstrate that you can run the solution outside Visual Studio using Docker containers. You should also demonstrate that two services can communicate with each other. You can use Postman to invoke the services. Each team member must actively participate in the presentation.

Team members who do not participate in the demonstration may be asked to write a report (5-10 pages), which covers the requirements for the demonstration.