SANCHEZ Pablo, THERET Guillaume

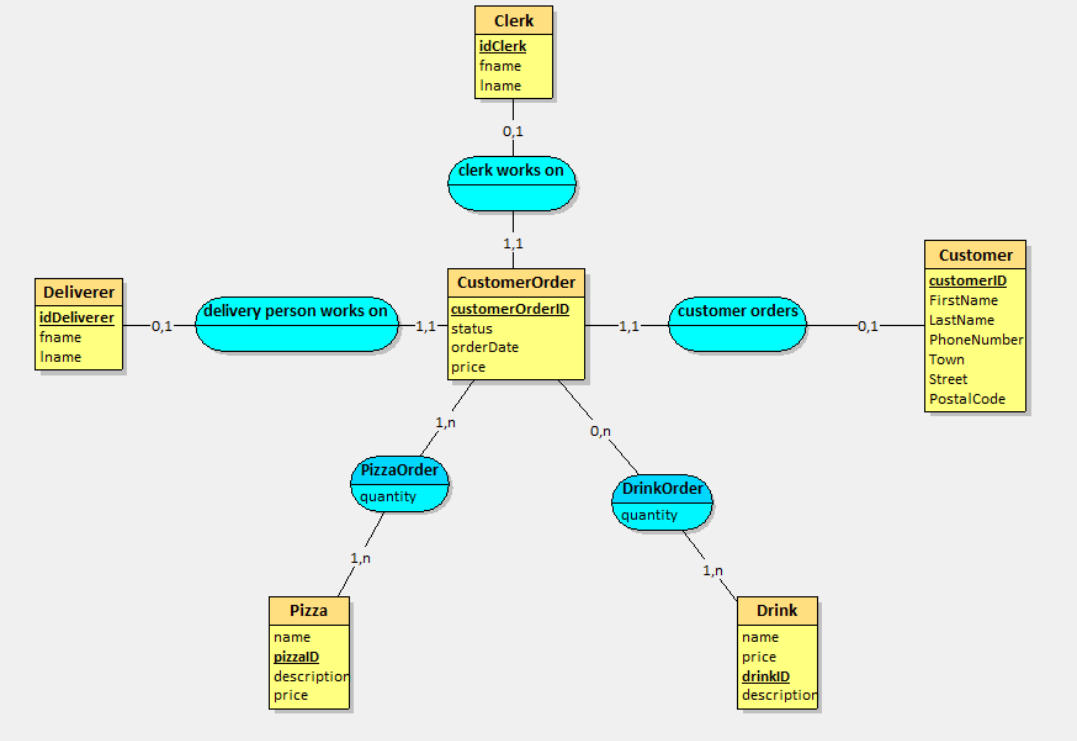
Group SE2

Intégration de systèmes : applications

Problem (MOM)

Our UML Diagrams:

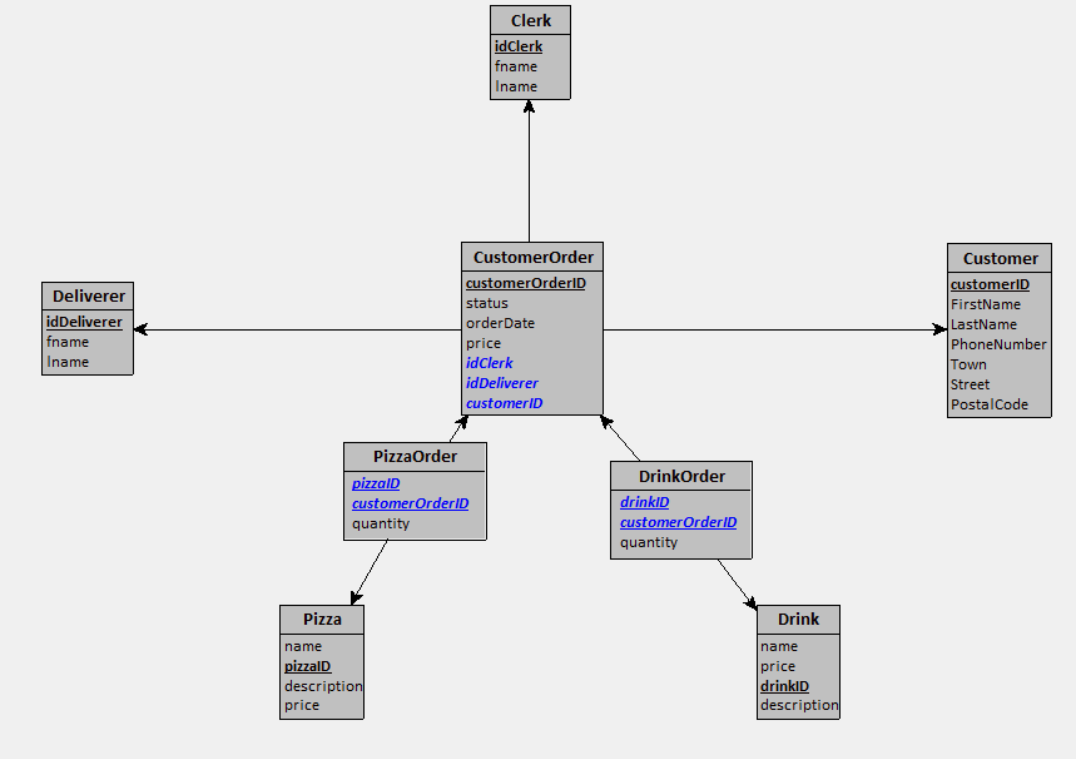
Relational diagram:



We decided to build our database structure around a main entity called “CustomerOrder”. This entity represents an order and is defined by an ID, a string “status” allowing us to say if an order is “closed”, “in preparation”, or “in delivery”. An order is also defined by a date with “orderDate”, and a “Price” integer, corresponding to the overall price of the order. It’s said that an order is composed of one to several pizzas, and none to several drinks. “Pizza” and “Drink” are two different entities, but are defined the same way: an id, a name, a description, and a price for the unit. There both associated to “CustomerOrder” with two associations “PizzaOrder” and “DrinkOrder”, with both attributes’ quantity. To each order, a clerk and a delivery person are in charge. They are two entities “Clerk” and “Deliverer”, defined also by the same attributes: an id, a first name, and a last name. Finally, an order is made by a customer represented by an “entity” Customer”, defined by an id, a first name, a last name, a phone number, a town, a street, and a postal code. The cardinalities are made so that an order is given one customer, one clerk, one delivery person, 1 to several pizzas, and à to several drinks.

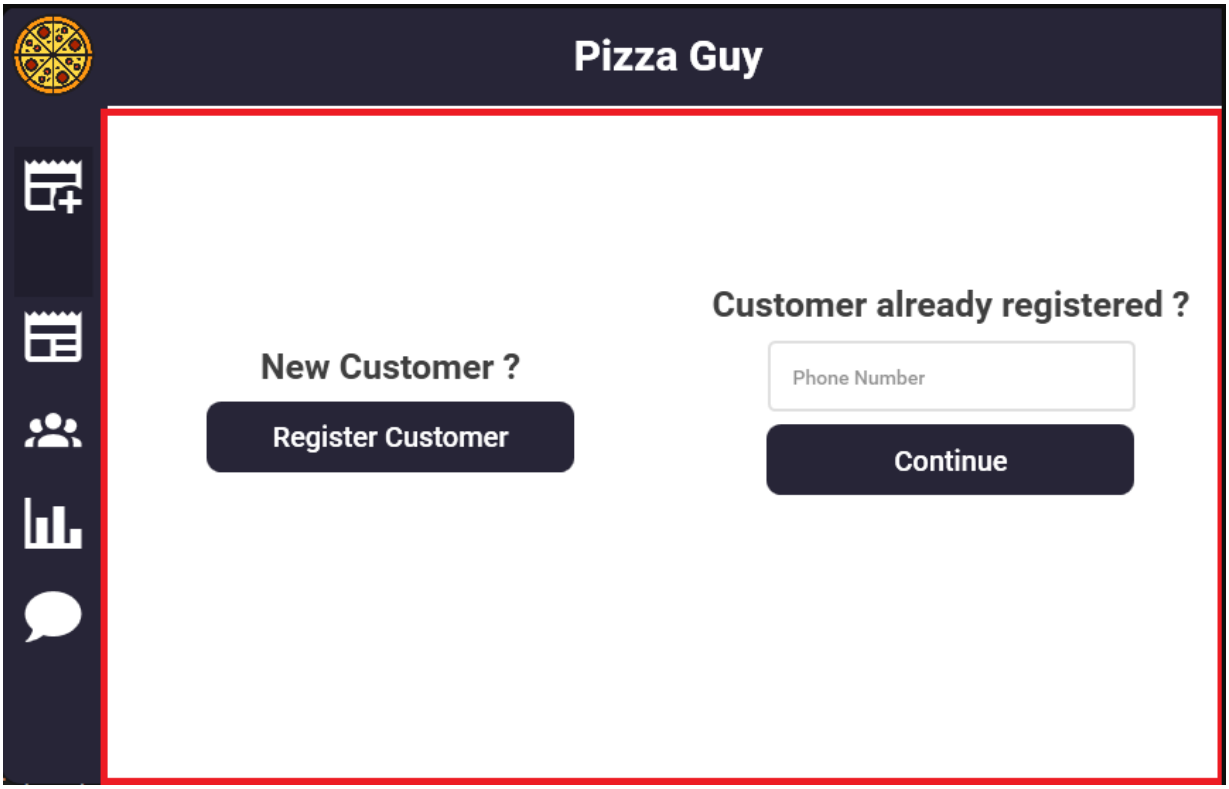
Based on this relational diagram, we build the following logical diagram, in which cardinalities are erased, and in which we can see that “CustomerOrder” is also defined by foreign keys “customerID”, “idClerk”, and “idDeliverer”, and that associations “PizzaOrder” and “DrinkOrder” must be tables in our database, with a pair of two foreign keys and one additional attribute quantity.

Logical Diagram:

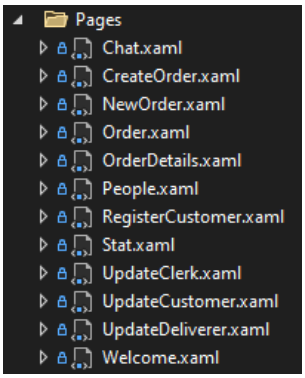


Our C# structure :

In order to build our application and its graphical interface, we use WPF on visual studio, with C#. We also used Docker and RabbitMQ to build the Message Broker System. Our application is made of one main app, and several Pages to navigate in, as we can see circle in red:

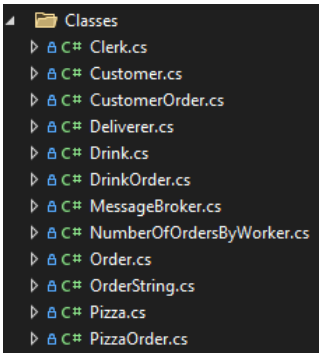


In our project, we have the following Pages:



The first Page, “Chat”, was meant to host our message system, but we ended up not using it. “CreateOrder” allows the user to create a new order, “NewOrder” hosts the login system, “Order” displays all ongoing orders, and allows us to navigate to “OrderDetails”, where we can see the details of the chosen order. “People” page allows us to see all customers / workforce and sort them. On this page, we can remove a user, or navigate to “UpdateClerk”, “UpdateCustomer”, or “UpdateDeliverer” pages, to update the chosen user. “RegisterCustomer” is a page to add a new customer, and “Stat” page allows us to see all the functionalities from Statistics module. Finally, “Welcome” page is a template of a welcome page for our app.

And we have the following Classes:



For our classes, we first have all the classes represented by the logical diagram’s tables, thus “Clerk”, “Customer”, “CustomerOrder”, “Deliverer”, “Drink”, “DrinkOrder”, “Pizza”, and “PizzaOrder”. To these classes, we added a class “MessageBroker” in which we built our message system, based on sending of notifications, and a class “NumberOfOrderByWorker” defined by 2 strings first and last name, and 2 integers for ID and the number of orders, which is used in page “Stat”. We also added “Order” defined by 4 integers as an ID, a customer ID, a price, and a quantity, and 1 string to hold a name, as parent class of “DrinkOrder” and “PizzaOrder”, that are only defined by a Drink ID and a Pizza ID. Finally, we have a class “OrderString” defined by a string and an integer to display a name and a quantity in “OrderDetails” page.