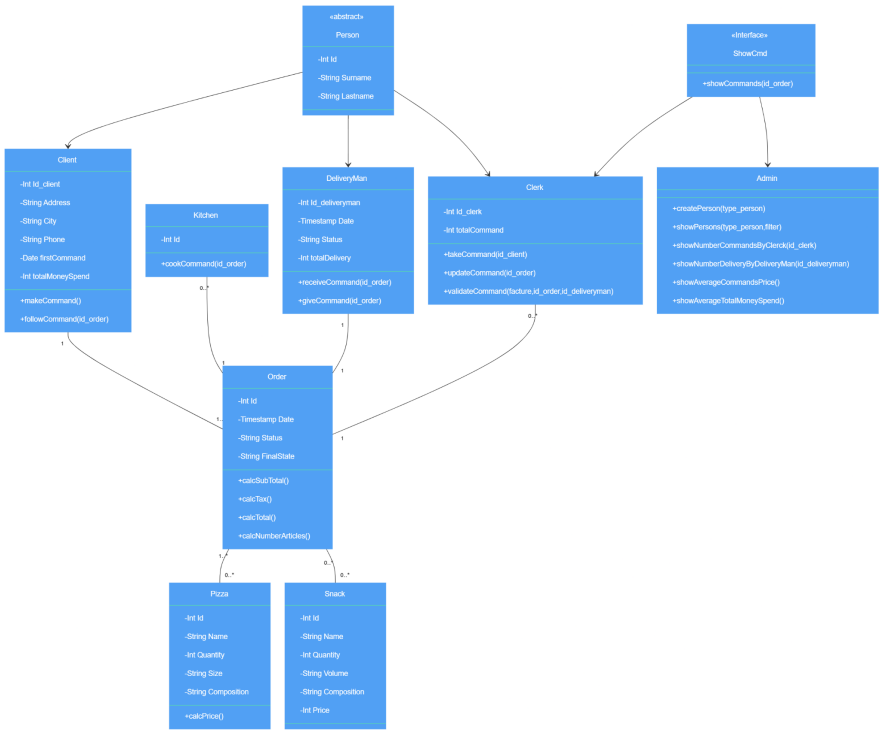
LE LAY Logan, MARGUET Vincent NASS Michael

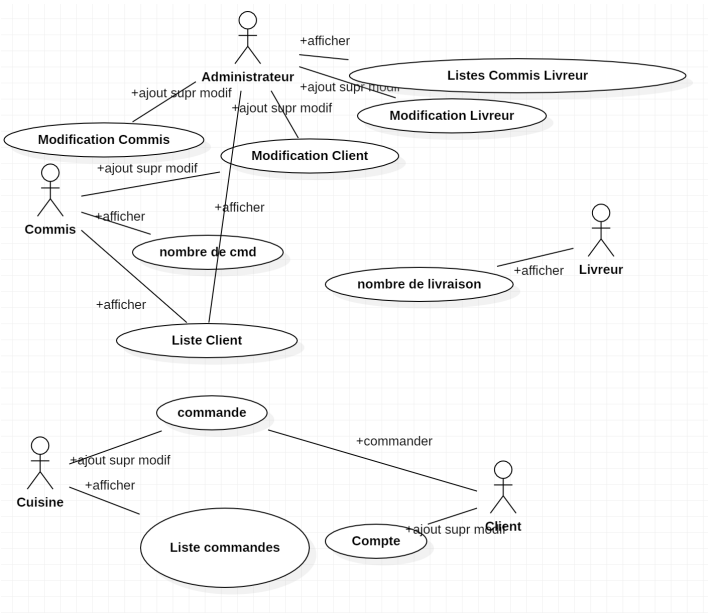
Rapport Application MOM

Pour réaliser notre projet Mom nous avons déjà dû définir nos classes et nos cas d' utilisations en amont.

Diagramme de classe :



Use Case :



On crée un menu on l utilisateur choisit qui il est :

```
0 references
static async Task Main(string[] args)
{
    Console.WriteLine("Welcome in the service of Pizzayolo !\n");
    Console.WriteLine("To start, define who are you ?" +
        "\n 1. Client" +
        "\n 2. Clerk" +
        "\n 3. Kitchen" +
        "\n 4. DeliveryMan" +
        "\nTo quit tap 0.\n");

    bool stop = false;

    while (!stop)
    {
        string chose = Console.ReadLine();

        switch (chose)
        {
            case "1":
                Console.Title = "Client";
                await ClientSide.ActivateClientSide();
                break;

            case "2":
                Console.Title = "Clerk";
                await ClerkSide.ActivateClerkSide();
                break;

            case "3":
                Console.Title = "Kitchen";
```

Et selon la réponse l'instance du service sélectionné sera activée.

Le client déclare son nom et ses infos puis choisit sa commande.
Le commis vérifie si le client en est à sa première commande puis saisit la commande du client. Il l'envoie ensuite à la cuisine.
La cuisine reçoit la commande et la dépose au livreur.
Livreur qui la livre au client, déclare l'état final de la commande et envoie une notification au client.

Module Client

Le clerk enregistre les clients uniques et peut calculer des statistiques.
L'enregistrement d'une nouvelle personne se fait lorsque celle-ci se connecte et entre ses identifiants pour chaque role.

Module Commandes

Non implémenté pour cause de bugs mais le superviseur peut vérifier a tout moment l'état d'une commande. Le calcul du prix se fait automatiquement lors de la saisie de la commande, qui permets donc de l afficher moyennant son numéro.
Non implémenté mais le superviseur peut afficher un commande selon son numéro.

Module stats

Codé mais non implémenté pour raisons de bugs mais le superviseur peut afficher les commis, livreurs et commandes a sa volonté.

Module Communication

Les différentes parties communiquent entre eux.

Client to commis :

```
public override bool SendCommand() {
    return Publisher.Publish<Client>(this, "client-clerk");
}

3 references
public override Order ReceiveCommand<Order>() {
    return Receiver.Receive<Order>("deliveryman-client");
}
```

Commis to cooker:

```
4 references
public override bool SendCommand() {
    return Publisher.Publish<Order>(orderGenerated,"clerk-kitchen");
}

3 references
public override Client ReceiveCommand<Client>() {
    return  Receiver.Receive<Client>("client-clerk");
}
```

Cooker to DeliveryMan :

```
// Methods
4 references
public override bool SendCommand() {
    return Publisher.Publish<Order>(orderGenerated, "kitchen-deliveryman");
}

3 references
public override Order ReceiveCommand<Order>() {
    return Receiver.Receive<Order>("clerk-kitchen");
}
```

DeliveryMan to client :

```
// Methods
4 references
public override bool SendCommand() {
    return Publisher.Publish<Order>(order, "deliveryman-client");
}

3 references
public override Order ReceiveCommand<Order>() {
    return Receiver.Receive<Order>("kitchen-deliveryman");
}
```

Module autre :

La POO a été réalisée avec l'héritage de client et des travailleurs avec la classe personne.
Personne est aussi une classe abstraite.
Kitchen a une interface.
La gestion des messages et l’asynchronisation des tâches se fait avec Message Broker et async/await.