Projet numéro 6:

Concevez la solution technique d’un système de gestion d’une pizzeria.

Elève: Vincent Cailly
Mentor: Stéphane Nédélec
Examinateur: Benoit Prieur

Contexte du devoir:

« OC Pizza » est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici la fin de l'année. Un des responsables du groupe a pris contact avec vous afin de mettre en place un système informatique, déployé dans toutes ses pizzerias et qui lui permettrait notamment :

1. d'être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;
2. de suivre en temps réel les commandes passées et en préparation ;
3. de suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas sont encore réalisables ;
4. de proposer un site Internet pour que les clients puissent :
    a. passer leurs commandes, en plus de la prise de commande par téléphone ou sur place
    b. payer en ligne leur commande s'ils le souhaitent – sinon, ils paieront directement à la livraison
    c. modifier ou annuler leur commande tant que celle-ci n'a pas été préparée
5. de proposer un aide mémoire aux pizzaiolos indiquant la recette de chaque pizza
6. d'informer ou notifier les clients sur l'état de leur commande
7. Le client a déjà fait une petite prospection et les logiciels existants qu'il a pu trouver ne lui conviennent pas.

Ce devoir est la suite du projet numéro 4: " Analysez les besoins de votre client pour son groupe de pizzerias".
https://github.com/Totobriac/OC-Pizzeria

Travail demandé:

---

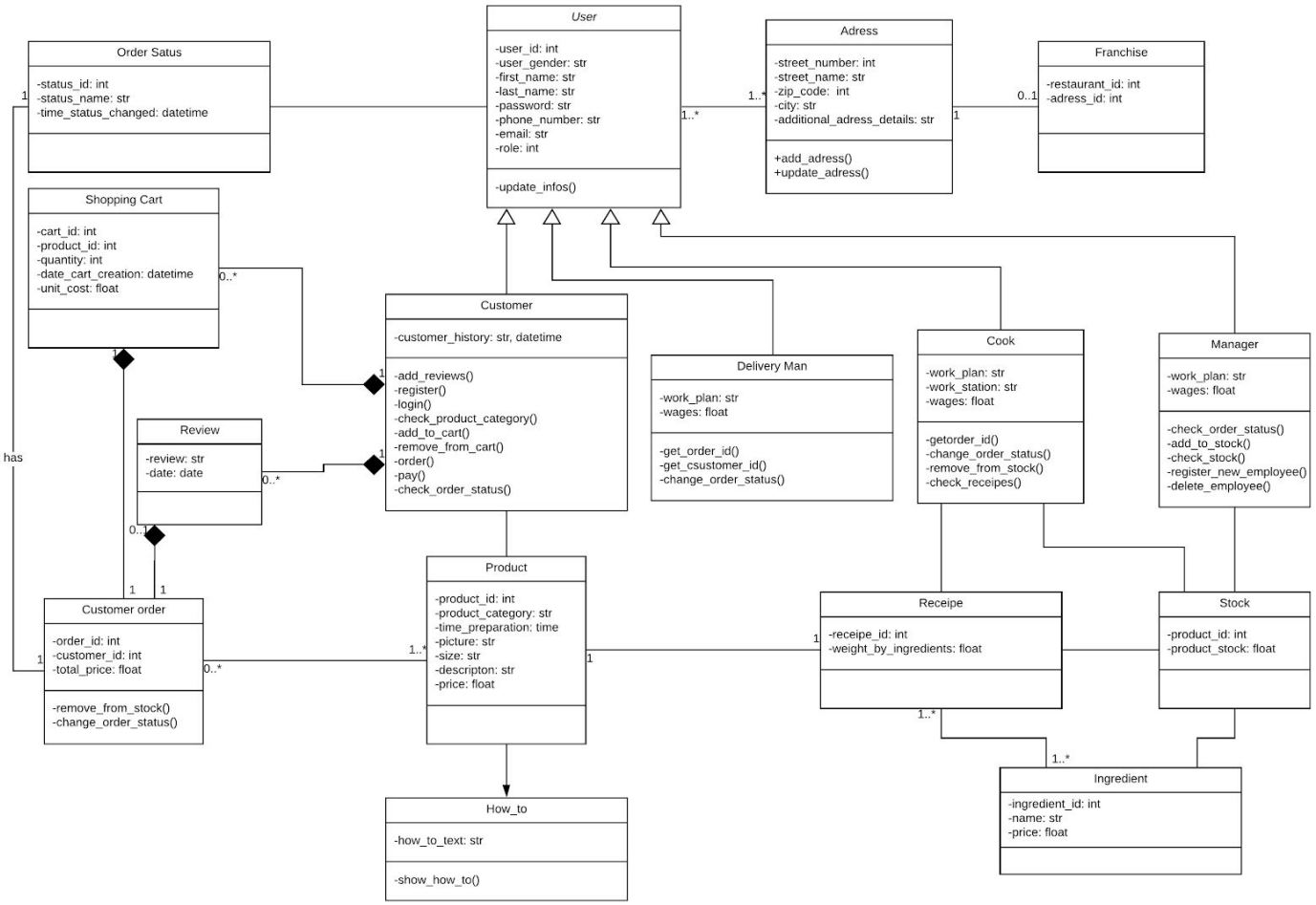
1. modéliser les objets du domaine fonctionnel.
2. identifier les différents éléments composant le système à mettre en place et leurs interactions.
3. décrire le déploiement des différents composants que vous envisagez.
4. élaborer le schéma de la ou des bases de données que vous comptez créer.

Modélisation des objets du domaine fonctionnel:

A travers ce diagramme nous décrivons toutes les classes que nous allons créer lors de la programmation du projet.

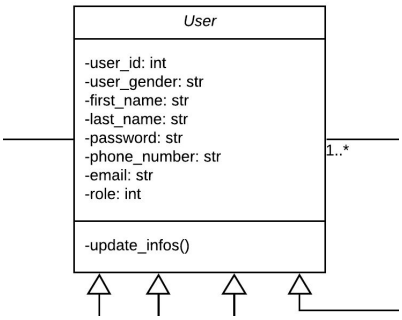
Nous notons aussi les relations qu'elles entretiennent l'une avec l'autre.

Enfin, ce diagramme sera la base de la création du "modèle physique de données".



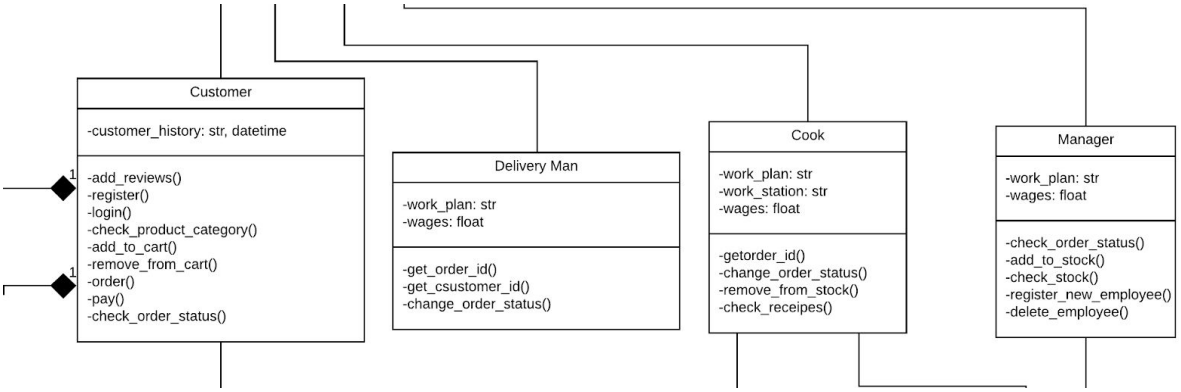
Principales classes:

Classe User



Classe mère, elle regroupe les attributs communs à tous les utilisateurs de notre application (numéro de téléphone, nom, prénom...). C'est le coeur de notre application.

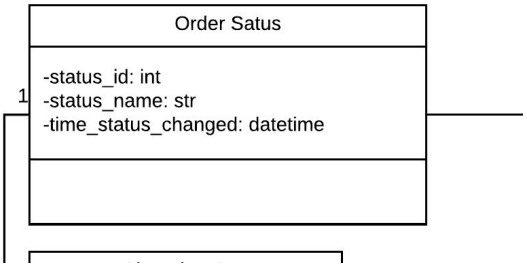
Ses classes filles héritent toutes de ces attributs:



Nous observons ainsi que chaque classe fille à ses propres méthodes et attributs.

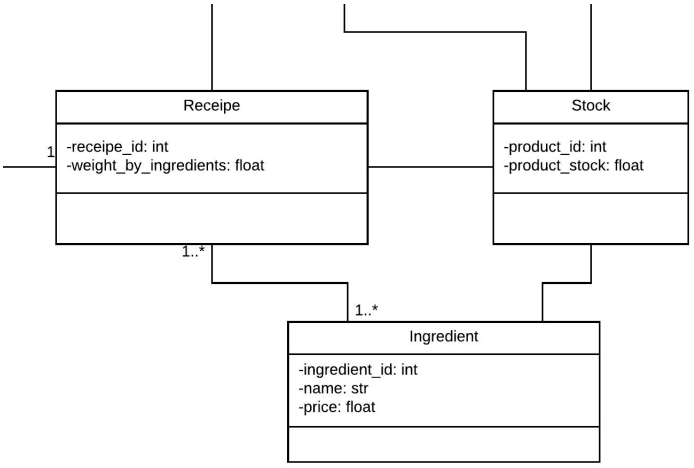
Ainsi il sera plus facile dans le futur de rajouter de nouvelles classes d'utilisateur si besoin est avec ses propres spécificités (serveurs, agents de sécurité...).

Classe Order Status



Une de la fonctionnalité demandée par le client est le suivis en temps réel de l'état d'une commande. Avec cette classe nous pouvons suivre l'état d'avancement de la commande et aussi à quelle heure le changement d'état à eut lieu.

Classe Stock



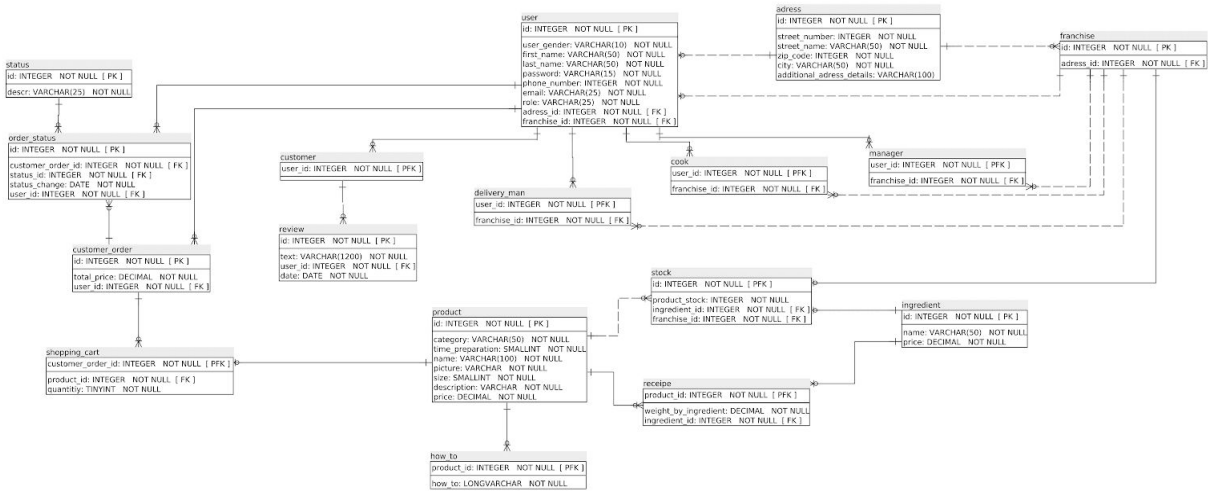
Aussi il nous fut demandé de pouvoir suivre en direct les stocks de produits que les pizzeria peuvent proposer.

Pour ceci nous décomposons chaque pizza par ingrédients et par le grammage d'ingrédients nécessaire à sa préparation, puis nous le comparerons au stock final d'ingrédients.

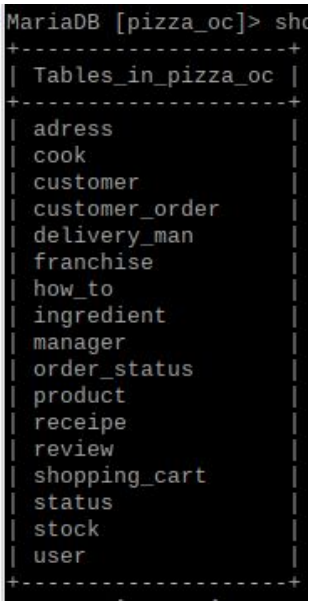
Pour avoir une vision du stock plus poussée, chaque stock est mis à jour par franchise, chaque client, selon quel restaurant il choisit, aura donc un résultat personnalisé.

Modèle physique de données:

A l’aide du domaine fonctionnel, nous élaborons un modèle physique de donnés. Ce diagramme décrit toutes les bases de données nécessaires au bon fonctionnement de notre application et toutes les interactions entre elles.



L’idée est que dans un futur proche l'application puisse évoluer , que nous puissions ajouter des fonctionnalitées. Aussi mettre à jour les recettes de pizza, ou plus simplement en créer de nouvelles se doit d'êtres simple.



(différentes tables de notre base de données)

Principales bases de données:

Bd User

```
MariaDB [pizza_oc]> describe user;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| id           | int(11)     | NO   | PRI | NULL    |       |
| user_gender  | varchar(10) | NO   |     | NULL    |       |
| first_name   | varchar(50) | NO   |     | NULL    |       |
| last_name    | varchar(50) | NO   |     | NULL    |       |
| password     | varchar(15) | NO   |     | NULL    |       |
| phone_number | int(11)     | NO   |     | NULL    |       |
| email        | varchar(25) | NO   |     | NULL    |       |
| role         | varchar(25) | NO   |     | NULL    |       |
| adress_id    | int(11)     | NO   | MUL | NULL    |       |
| franchise_id | int(11)     | NO   | MUL | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

Cette base de données regroupe les informations de chaque utilisateur, qu'il soit consommateur, cuisinier, livreur ou manager.

Deux contraintes sont imposées afin de garantir l'intégrité des données:

-adress_id: un 'user' ne peut exister sans adresse

-franchise_id: un 'user' doit avoir un magasin attitré. Que cela soit le magasin le plus proche pour un client, ou son lieu de travail pour un employé.

Customer order

```
MariaDB [pizza_oc]> describe customer_order;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| user_id     | int(11)      | NO   | MUL | NULL    |                |
| total_price | decimal(5,2) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

Cette base de données regroupe les commandes des clients. La contrainte sur user_id permet que aucune commande ne soit créé sans user-id.

Order_status

```
MariaDB [pizza_oc]> describe order_status;
+-------------------+----------+------+-----+---------+----------------+
| Field             | Type     | Null | Key | Default | Extra          |
+-------------------+----------+------+-----+---------+----------------+
| id                | int(11)  | NO   | PRI | NULL    | auto_increment |
| customer_order_id | int(11)  | NO   | MUL | NULL    |                |
| status_id         | int(11)  | NO   | MUL | NULL    |                |
| status_change     | datetime | NO   |     | NULL    |                |
| user_id           | int(11)  | NO   | MUL | NULL    |                |
| franchise_id      | int(11)  | NO   | MUL | NULL    |                |
+-------------------+----------+------+-----+---------+----------------+
```

Cette base de données nous permet de suivre le changement de statut de nos commandes. Beaucoup de contraintes sont imposées pour garantir la bonne “traçabilité” des commandes. Ceci est une des fonctionnalité demandée par notre client.

Ainsi pour un changement de statut nous devons impérativement avoir un numéro de commande qui correspond, un responsable du changement et un magasin dans lequel la commande est réalisée.

Stock

```
MariaDB [pizza_oc]> describe stock;
+---------------+---------+------+-----+---------+----------------+
| Field         | Type    | Null | Key | Default | Extra          |
+---------------+---------+------+-----+---------+----------------+
| id            | int(11) | NO   | PRI | NULL    | auto_increment |
| ingredient_id | int(11) | NO   | MUL | NULL    |                |
| franchise_id  | int(11) | NO   | MUL | NULL    |                |
| product_stock | int(11) | NO   |     | NULL    |                |
+---------------+---------+------+-----+---------+----------------+
```

Comme convenu avec notre client, nous implémenterons un système pointu de suivi des stocks. Pour ce faire nous créons une base de données ou chaque ‘stock’ devra correspondre à un produit existant (ingredient_id) et “appartenir” à un magasin donné.

Diagramme de déploiement:

Ce diagramme décrit les différents composants de notre projet.

Nous observons que l'interface client, tout comme l'interface pizzeria sera accessible grâce à un web-browser. Cela permet de pouvoir utiliser notre produit sur une infinité d'appareils différents (PC, téléphone, tablette...)

L'application principale elle sera reliée à une base de données Mysql.

Bien sûr, nous aurons besoin d'API externes pour procéder au paiement ainsi que pour géolocaliser les clients.