IT CONSULTING
& DEVELOPMENT

OC Pizza

Description du domaine fonctionnel du nouveau système de gestion des commandes de OC Pizza

1. Description du projet

OC Pizza est un jeune groupe de pizzeria en plein essor. Le groupe est spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 4 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici 6 mois.

OC Pizza a besoin d'un système de gestion pour ses restaurants car aucune solution n'a été trouvée parmi les logiciels existants.

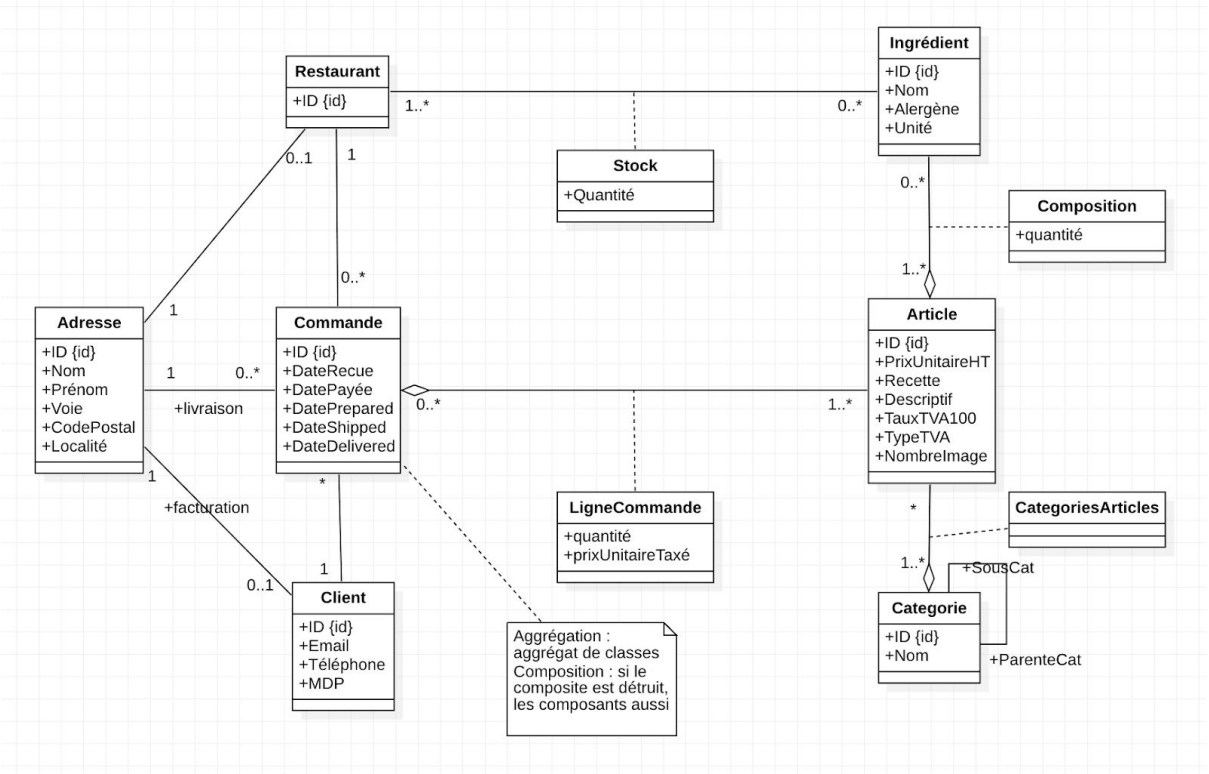
L'objectif confié à IT consulting development est d'en développer un sur-mesure.

Rappel des besoins exprimés par le client.

2. Description du domaine fonctionnel

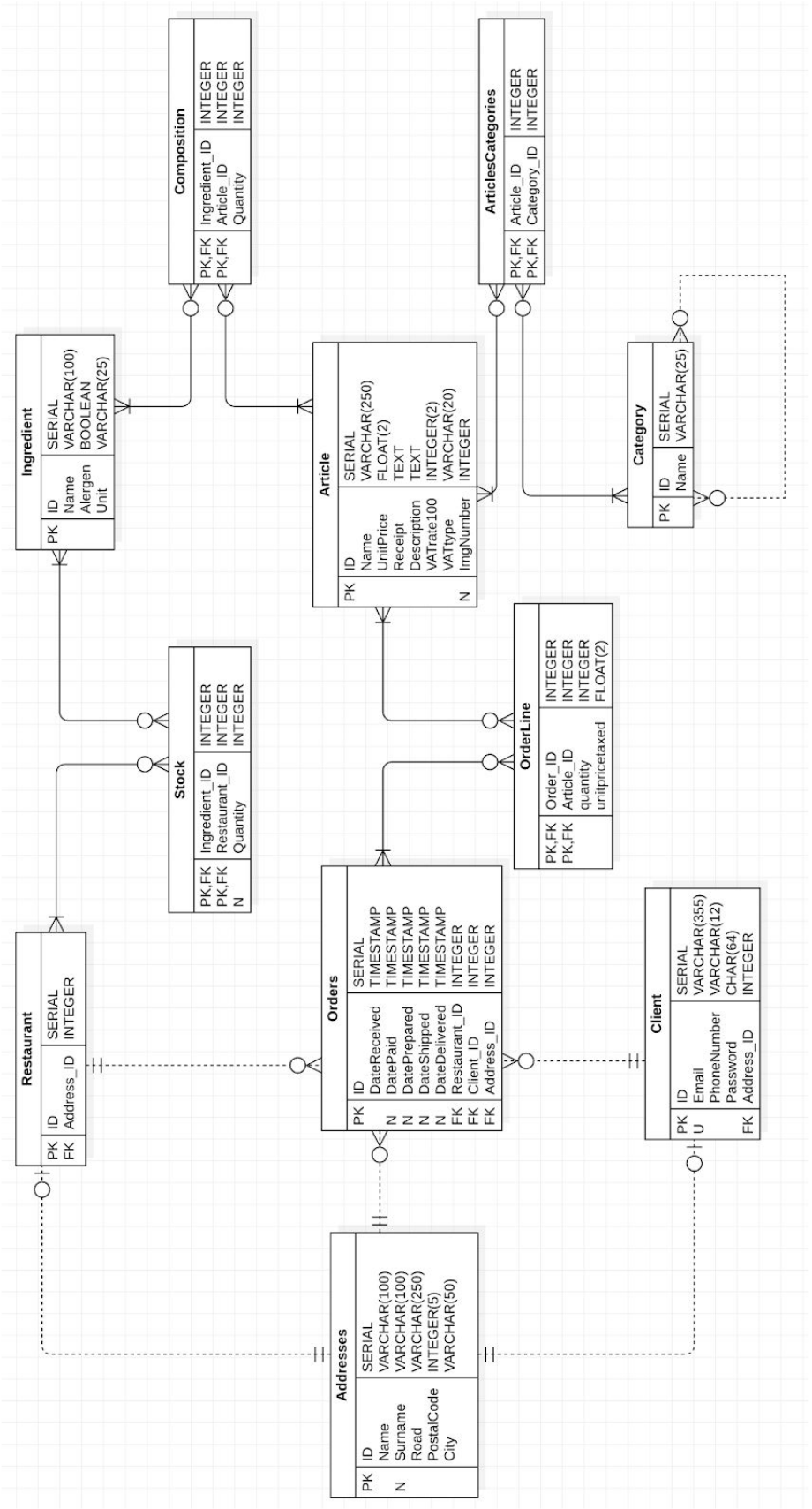
2.1. Diagramme de classe

Le diagramme de classe est le suivant :



2.2. Modèle physique de données

Le domaine physique de données associé est reporté ci-dessous.


Composition
Ingredient_ID INTEGER
Article_ID INTEGER
Quantity INTEGER
PK,FK
PK,FK
ArticlesCategories
Article_ID INTEGER
Category_ID INTEGER
PK,FK
PK,FK
Ingredient
ID SERIAL
Name VARCHAR(100)
Alergen BOOLEAN
Unit VARCHAR(25)
PK
Article
ID SERIAL
Name VARCHAR(250)
UnitPrice FLOAT(2)
Receipt TEXT
Description TEXT
VATrate100 INTEGER(2)
VATtype VARCHAR(20)
ImgNumber INTEGER
PK
N
Category
ID SERIAL
Name VARCHAR(25)
PK
Stock
Ingredient_ID INTEGER
Restaurant_ID INTEGER
Quantity INTEGER
PK,FK
PK,FK
N
OrderLine
Order_ID INTEGER
Article_ID INTEGER
quantity INTEGER
unitpricetaxed FLOAT(2)
PK,FK
PK,FK
Restaurant
ID SERIAL
Address_ID INTEGER
PK
FK
Orders
ID SERIAL
DateReceived TIMESTAMP
DatePaid TIMESTAMP
DatePrepared TIMESTAMP
DateShipped TIMESTAMP
DateDelivered TIMESTAMP
Restaurant_ID INTEGER
Client_ID INTEGER
Address_ID INTEGER
PK
N
N
N
N
FK
FK
FK
Client
ID SERIAL
Email VARCHAR(355)
PhoneNumber VARCHAR(12)
Password CHAR(64)
Address_ID INTEGER
PK
U
FK
Addresses
ID SERIAL
Name VARCHAR(100)
Surname VARCHAR(100)
Road VARCHAR(250)
PostalCode INTEGER(5)
City VARCHAR(50)
PK
N

Le tables créées et peuplées de données de test sont les suivantes :

```
ocpizza=# \dt
              List of relations
 Schema |        Name        | Type  | Owner
--------+--------------------+-------+-------
 public | address            | table | vft
 public | article            | table | vft
 public | articlescategories | table | vft
 public | category           | table | vft
 public | client             | table | vft
 public | composition        | table | vft
 public | ingredient         | table | vft
 public | orderline          | table | vft
 public | orders             | table | vft
 public | restaurant         | table | vft
 public | stock              | table | vft
(11 rows)
```

Les données de test ont été générées à l'aide de la webapp Moockaroo. Les difficultés rencontrées ont consisté en l'ordre de **création des tables qui était faite dans l'ordre alphabétique au lieu de l'ordre logique de création des table selon les clés étrangères** et l'ordre des **attributs qui n'était pas le même que celui des données de test en CSV**.

Un court script python a été écrit pour regénérer les requêtes SQL rapidement il est disponible dans le projet git.

Les données insérées sont numériquement les suivantes :

TABLE	**\i data_moockaroo_insert.sql**
addresses	COPY 100
category	COPY 4
ingredient	COPY 35
article	COPY 30
client	COPY 80
restaurant	COPY 5
orders	COPY 150
orderline	COPY 200
composition	COPY 80
stock	COPY 140
articlescategories	COPY 30

2.3. Tests (T1 = test1.sql, T2 = test2.sql...)

Les questions posées en terme de test sont listées ci-dessous avec leur transcription en SQL et le résultat de la requête. Les fichiers SQL sont en ligne sur Github.

Les résultats peuvent être vérifiés à l'aide des **données de tests qui ont été reportées dans ce [tableau partagé]**.

T1. ma commande peut-elle contenir plusieurs pizzas ?

```
SELECT COUNT(*) FROM (
   SELECT
       -- compte les articles des commandes
       COUNT(article_ID), order_ID
   FROM
       orderline
   GROUP BY
       order_ID
   HAVING
       -- compte les commandes à plusieurs articles
       COUNT(article_ID) > 1)
AS
   toto;
```

donne le nombre de commandes avec plusieurs articles :

```
 count
-------
    49
(1 row)
```

Résultat conforme aux attentes : les couples de orderline sont parfois répété 2 fois sauf un qui l'est 3 fois.

T2. puis-je retrouver le contenu d'une commande ?

```
SELECT
   orderline.Order_ID, orderline.quantity, article.name
FROM
   orderline
LEFT JOIN article
   ON orderline.article_ID = article.ID
WHERE Order_ID = 14;
```

donne :

```
order_id | quantity |           name
----------+----------+--------------------------
       14 |        2 | Grisebach's Bristlegrass
       14 |        3 | Rinodina Lichen
```

```
ocpizza=# SELECT id, name FROM article ORDER BY ID;
 id |             name
----+------------------------------
  1 | Singlestem Leather-root
  2 | Greene's Goldenbush
  3 | Limestone Adderstongue
  4 | Red Valerian
  5 | Earth Loosestrife
```

```
             6 | Mono Milkvetch
 7 | Hybrid Oak
 8 | Reverchon's Palafox
 9 | Johnson's Blazing Star
10 | Climbing Bedstraw
11 | Grisebach's Bristlegrass
12 | Golden Eye Saxifrage
13 | Summer Holly
14 | Twining Snoutbean
15 | Ionopsidium
16 | Lean Flatsedge
17 | Trans-pecos Croton
18 | Reverchon's Rosinweed
19 | Spotted Joe Pye Weed
20 | Rinodina Lichen
21 | Puerto Rico Swallow-wort
22 | Prettyface
23 | Siberian Crab Apple
24 | Tree Brake
25 | Western Meadow-rue
26 | Tropical Puff
27 | Broad Fleabane
28 | Checkered Rattlesnake Plantain
29 | Cynodontium Moss
30 | Sideoats Grama
```

T3. puis-je afficher les commandes non livrées dans un restaurant particulier ?

Pour trouver le nom du restaurant :

```
SELECT
   Restaurant.ID, Addresses.name
FROM
   Restaurant
   LEFT JOIN Addresses
   ON Addresses.ID = Restaurant.Address_ID
WHERE Restaurant.ID = 3;
```

Pour trouver les commandes non livrées dans ce restaurant :

```
SELECT
   Orders.ID
FROM
   Orders
WHERE (Orders.DateDelivered IS NULL AND Orders.Restaurant_ID = 3);
```

T4. puis-je afficher le prix d'une commande donnée ?

```
WITH Summary AS (
    SELECT
        ROUND((Article.UnitPrice *
        (1+Article.VATrate100::NUMERIC/100))::NUMERIC, 2) AS Price,
        OrderLine.quantity
    FROM
```

```
                OrderLine
        LEFT JOIN Article
            ON OrderLine.article_ID = Article.id
        WHERE
            OrderLine.order_id = 13
        )
    SELECT
        SUM(Price * quantity)
    FROM
        Summary;
```

Donne :

```
ocpizza=# \i test_4.sql
  sum
-------
 20.04
(1 row)

 id | article_id | id | price
----+------------+----+-------
 13 |         17 | 17 |  2.62
 13 |         24 | 24 | 17.42
```

La commande 13 qui contient les articles 17 et 24 coûte bien :
1,36*(1+0,93)*1 + 9,12*(1+0,91)*1 = 20,04

T5. puis-je afficher les commandes en attente d'un client ?

Pour trouver les commandes en attente d'un client :

```
SELECT
   Orders.ID
FROM
   Orders
WHERE (Orders.DateDelivered IS NULL AND Orders.Client_ID = 4);
```

Donne :

```
 id
-----
  53
 102
(2 rows)
```

Autrement pour afficher en plus le nom :

```
WITH NotDelivered AS (
   SELECT
      ARRAY_AGG(Orders.ID) AS orders
   FROM
      Orders
   WHERE (Orders.DateDelivered IS NULL AND Orders.Client_ID = 4)
   )
   SELECT
      Addresses.Name, NotDelivered.orders
   FROM
```

```
            NotDelivered,
        Addresses
           RIGHT JOIN Client
           ON Addresses.id = Client.address_ID
     WHERE Client.ID = 4;
```

Donne les commande non livrées correspondant au client 4 :

```
ocpizza=# \i tests/test_6.sql
   name     |  orders
------------+----------
 Aujouanet | {53,102}
(1 row)
```

T6. puis-je afficher l'adresse de livraison d'une commande terminée même après que le client a changé son adresse ?

La requête suivante donne l'adresse 13 :

```
SELECT
   Orders.ID, Addresses.Road, Addresses.PostalCode, Addresses.City
FROM
   Orders
   LEFT JOIN Addresses
      ON Addresses.ID = Orders.Address_ID
WHERE
   Orders.ID = 53;
```

Adresse 13 (qui n'est pas celle du client qui a changé d'adresse) :

```
ocpizza=# \i tests/test_6.sql
 id |        road         | postalcode |  city
----+---------------------+------------+--------
 53 | 582 Monterey Circle |      76531 | Rennes
(1 row)
```

Ce qui est cohérent avec les données :

```
id     restaurant_ID    client_ID    Address_ID
53     1                4            13
```

T7. puis-je lister les pizzas pour lesquelles tous les ingrédients sont en stock dans un restaurant donné ?

Pour être sûr d'avoir au moins un article avec le bon stock, on update la table Composition (test/test_7_alter_compo.sql) :

```
UPDATE Composition SET
   Quantity = 2
WHERE Article_ID = 10;
```

On créé un tableau intermédiaire NotEnough pour trouver les articles qui n'ont pas assez d'ingrédients en stock dans le restaurant 3 :

```
SELECT
```

```
      Composition.article_id, Composition.ingredient_ID, Stock.quantity AS
   available, Composition.quantity AS needed
FROM
   Composition, Stock
WHERE
   Composition.quantity > Stock.quantity
   AND Stock.ingredient_ID = Composition.ingredient_ID
   AND Stock.restaurant_ID = 3
ORDER BY Composition.article_id;
```

Puis avec un FULL JOIN pour avoir tous les articles même ceux qui ont assez d'ingrédients, on trouve le résultat recherché :

```
WITH NotEnough AS(
       SELECT
       Composition.article_id, Composition.ingredient_ID, Stock.quantity AS
       available, Composition.quantity AS needed
   FROM
       Composition, Stock
   WHERE
       Composition.quantity > Stock.quantity
       AND Stock.ingredient_ID = Composition.ingredient_ID
       AND Stock.restaurant_ID = 3
   ORDER BY Composition.article_id
   )
   SELECT
       Article.id, COUNT(NotEnough.ingredient_ID)
   FROM
       Article
       FULL JOIN NotEnough
           ON Article.id = NotEnough.article_id
   GROUP BY
       Article.id
   HAVING
       COUNT(NotEnough.ingredient_ID) = 0
   ORDER BY Article.id;
```

Soit :

```
 id | count
----+-------
 10 |     0
 27 |     0
```

Les articles 10 et 27 ont assez d'ingrédients en stock pour être réalisés.

T8. puis retrouver le prix payé pour une pizza dans une commande terminé même si le prix a changé depuis ?

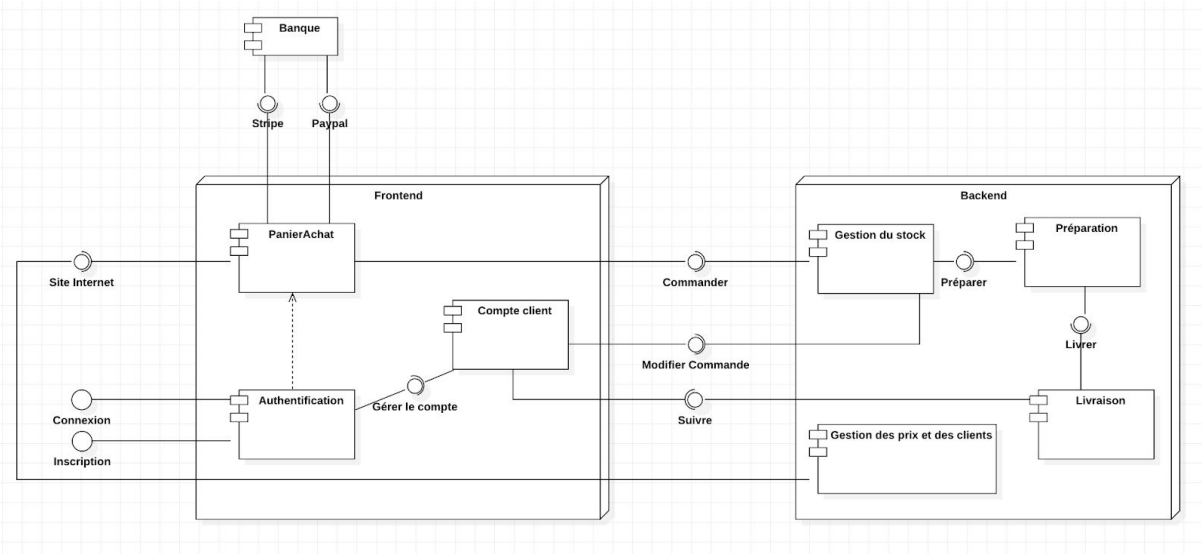
À la base non. Ajout d'une colonne unitpricetaxed dans la table Orderline.

```
UPDATE
   Orderline
SET
   unitpricetaxed = ROUND((Article.unitprice *
   (1+Article.vatrate100::NUMERIC/100))::NUMERIC, 2)
FROM Article
WHERE Orderline.article_ID = Article.ID;
```

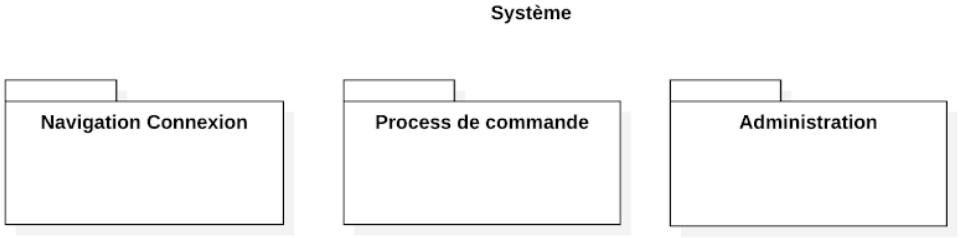
Désormais, le prix de chaque article de orderline est indépendant de celui de l'article qui aurait été modifié.

3. Les différents composants du système et les composants externes utilisés par celui-ci et leur interaction

Les composants du système sont les suivants :



Pour rappel, lors du projet 4, les composantes du système avaient été identifiées comme suit :



4. Description de l'organisation physique de ces composants (déploiement)

Le site internet utilise les port 80 pour le protocole http et le 443 pour le protocole https.

L'ordinateur "client" décrit à la fois le backoffice des restaurants et l'ordinateur d'un client au sens classique du terme.