

Projet 07

Suite du projet 06
OC Pizza

OC Pizza

Spécifications Techniques



Angélique Fourny 12.2020

Sommaire

Le document concerne la conception techniques de notre futur solution **OCPizza**, qui fait suite au contenu du **projet 06** qui nous a aidé à mettre en place les documents que nous allons abordés :

Le modèle fonctionnel avec un diagramme de classe ainsi qu'un modèle physique de données qui va nous servir dans la conception de notre base de données. Ainsi qu'un diagramme de déploiement et de composant

CAHIER DES CHARGES

OC Pizza ? 1

Le Domaine Fonctionnel :

- Diagramme de classes 2
- Descriptif 3 5

Le Modèle Physique de Donnée :

- MPD 6
- Descriptif 7 9

La base de donnée 10

Le Diagramme de composants 11

Le Diagramme de déploiement 12



OCPizza est un jeune groupe de pizzeria en plein essor. *Créé par Franck et Lola*, le groupe est 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 6 mois.

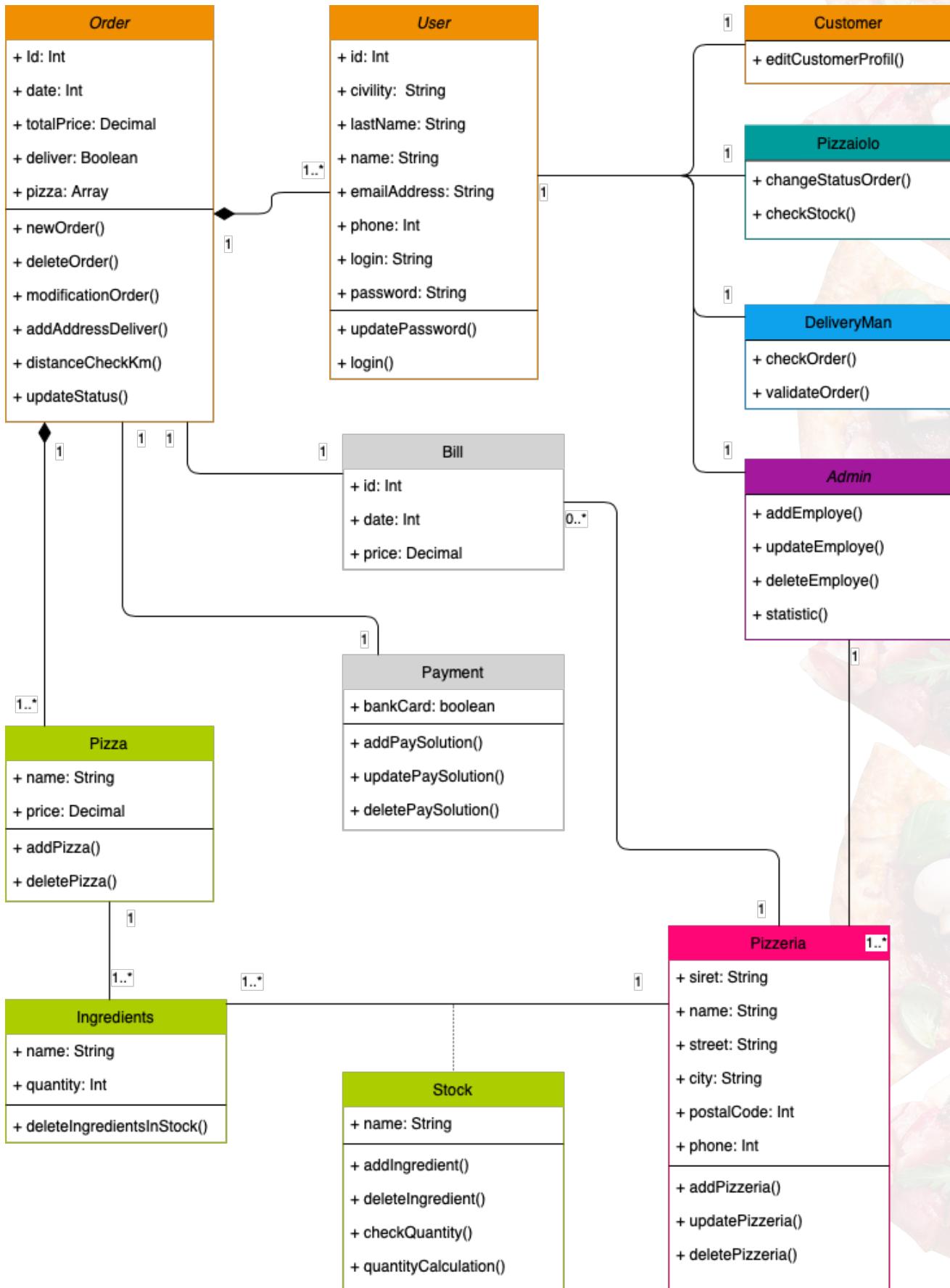
Le système informatique actuel ne correspond plus aux besoins du groupe car il ne permet pas une gestion centralisée de toutes les pizzerias.

De plus, il est très difficile pour les responsables de suivre ce qui se passe dans les points de ventes.

Les besoins d'OCPizza

- être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;
- suivre en temps réel les commandes passées, en préparation et en livraison ;
- suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas peuvent encore être réalisées ;
- proposer un aide-mémoire aux pizzaiolos indiquant la recette de chaque pizza
- proposer un site Internet pour que les clients puissent :
 1. passer leurs commandes, en plus de la prise de commande par téléphone ou sur place ;
 2. payer en ligne leur commande s'ils le souhaitent ou sinon, ils paieront directement à la livraison ;
 3. modifier ou annuler leur commande tant que celle-ci n'a pas été préparée.

Diagramme de classes UML d'OCPizza

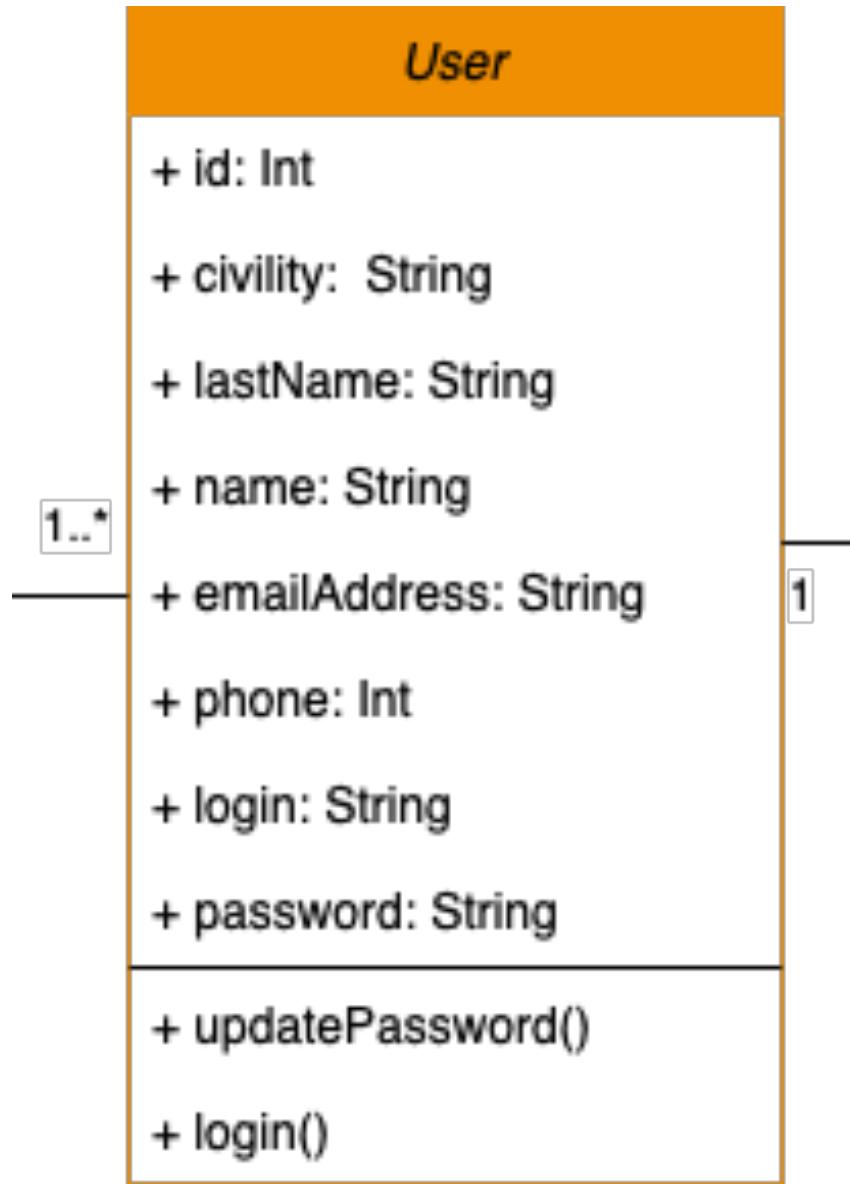


Le Domaine Fonctionnel : Descriptif

Cahier des charges

3

2 & 5 - Le domaine fonctionnel / 6 & 9 - Le modèles physique de donnée / 10 - le base de donnée / 11 - Le diagramme de composants / 12 - Le diagramme de déploiement

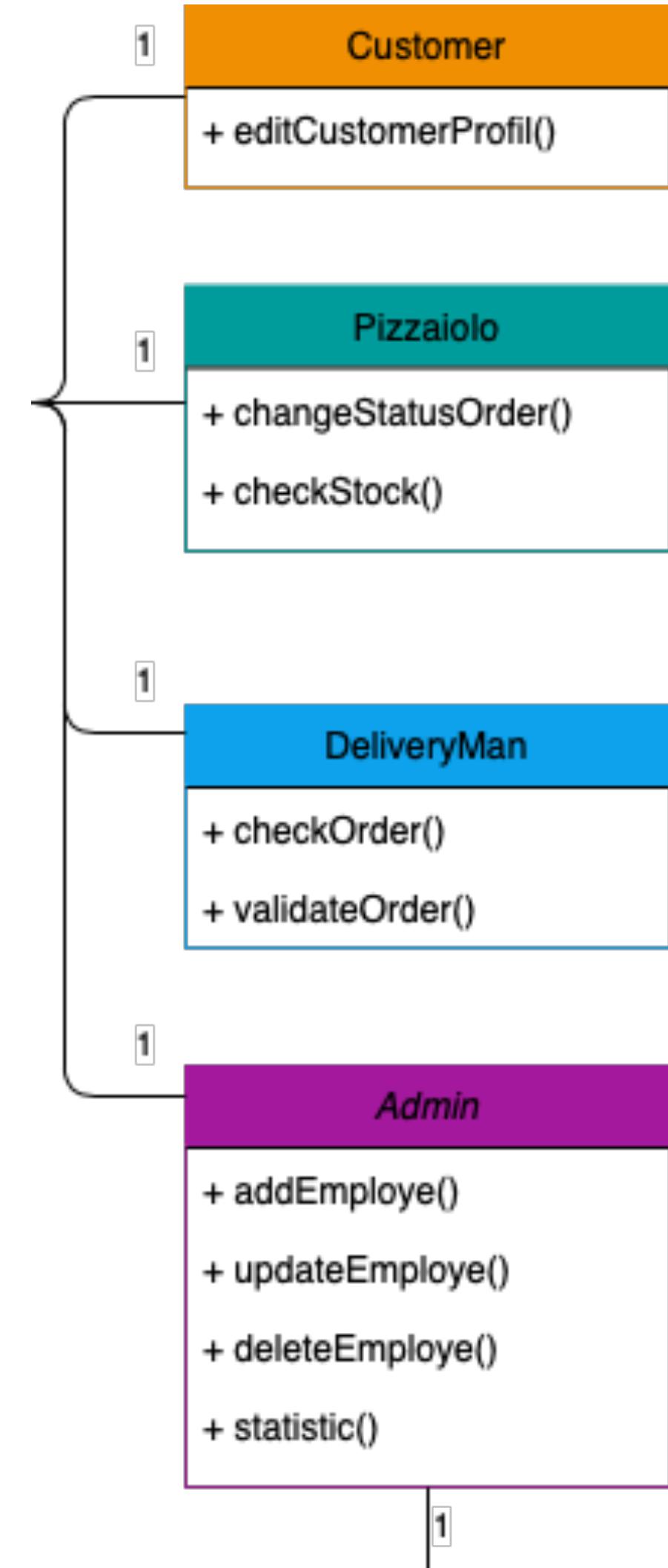


Cette classe **User** est associée aux classes filles , , et .

Elle est aussi associée à la classe **Order**. Plusieurs User peuvent avoir accès à une même commande. Chaque User a un identifiant et mot de passe unique.

Les Attributs de la classe **User** sont :

- civility : le genre Monsieur ou Madame
- lastName : le nom de famille
- name : le prénom
- emailAddress : l'adresse email
- phone : numéro de téléphone
- login : identifiant
- password : mot de passe



Customer
 Pizzaiolo
 DeliveryMan
 Admin.

Les classes filles , , et héritent de tous les attributs de la classe **User**.

Ils sont associés à la classe **User**.

Avec pour une fonctionnalités pour éditer son profil.

Pour la possibilité de changer les statuts des commandes et de vérifier les stock

Pour la possibilité de vérifier les commandes et de les valider

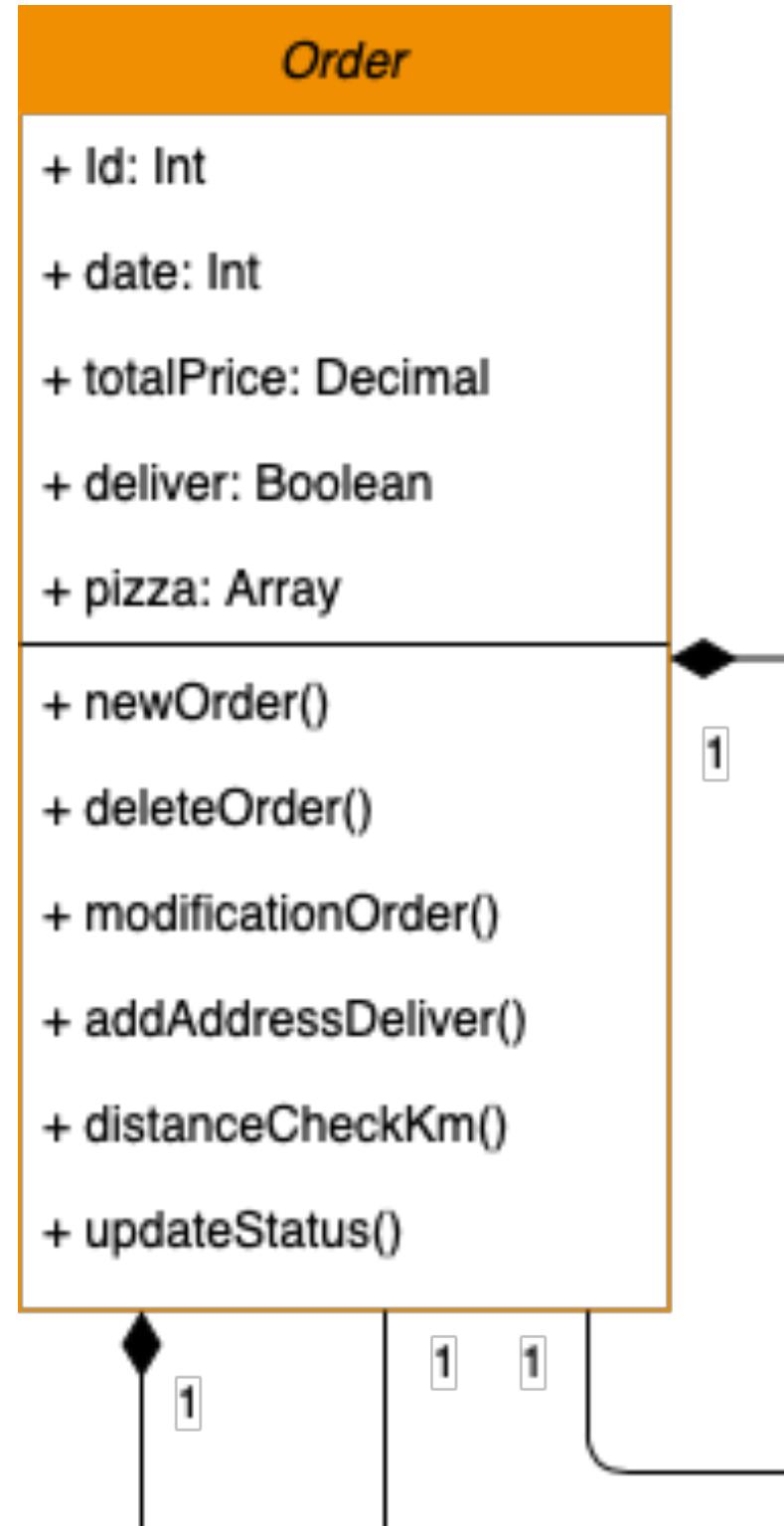
Pour la possibilité d'ajouter des employés, les modifier les supprimer.
Et faire des statistiques

Le Domaine Fonctionnel : Descriptif

Cahier des charges

4

2 & 5 - Le domaine fonctionnel / 6 & 9 - Le modèles physique de donnée / 10 - le base de donnée / 11 - Le diagramme de composants / 12 - Le diagramme de déploiement

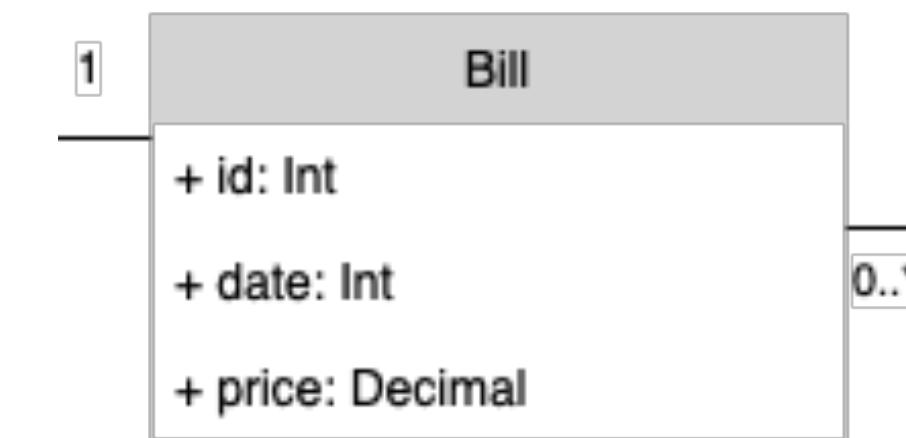


La Classe **Oder = commande** est associée aux classes **Bill**, **Payment**, **Pizza**.

Les Attributs de cette Classe sont :

- date : date de la **commande**
- totalPrice : le montant total de la **commande**
- deliver : la livraison
- pizza : les différentes pizzas

Il sera possible de créer, supprimer ou modifier les commandes. De choisir un livreur pour la livraison. De vérifier que la distance entre la pizzeria et l'adresse de livraison est bien inférieur ou égal à 20 km. Et de pourvoir afficher le statut de la commande en temps réel

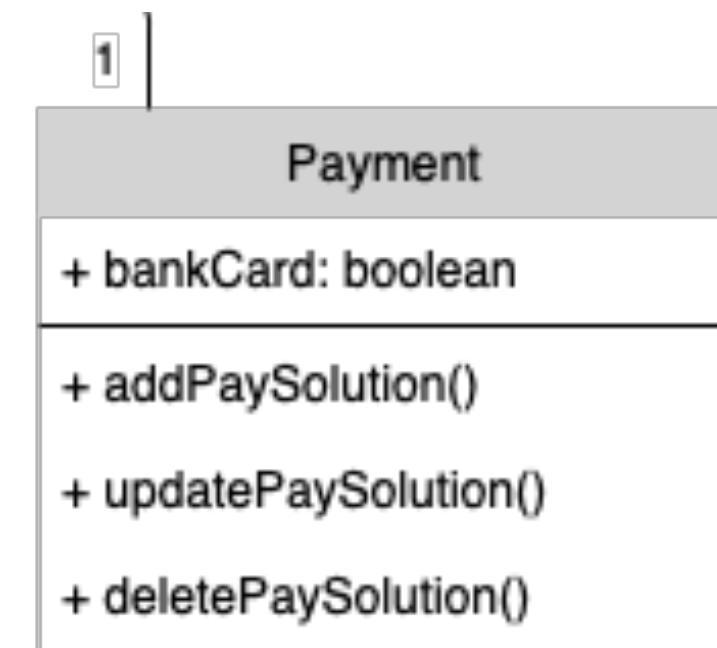


La Classe **Bill = facture**.

Les Attributs de cette Classe sont :

- date : date de la **facture**
- price : le montant **facture**

La facture sera associé à la Classe **Pizzéria**



La Classe **Payment = paiement**.

Les Attributs de cette Classe sont :

- bankCard : proposer le paiement par CB ou dans le cas contraire en espèce

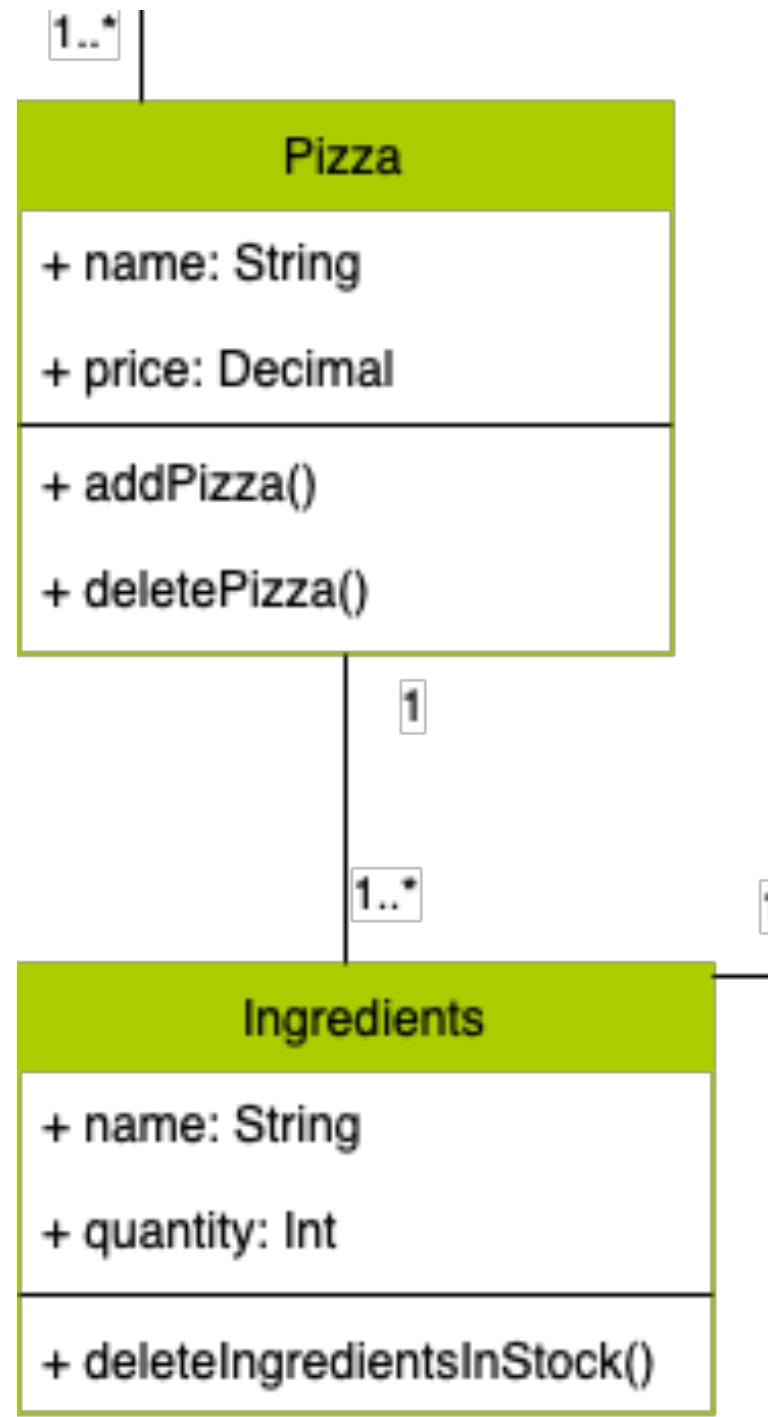
Il sera possible d'ajouter le moyen de paiement de la modifier ou de le supprimer

Le Domaine Fonctionnel : Descriptif

6 & 9 - Le modèles physique de donnée / 10 - le base de donnée / 11 - Le diagramme de composants / 12 - Le diagramme de déploiement

Cahier des charges

5

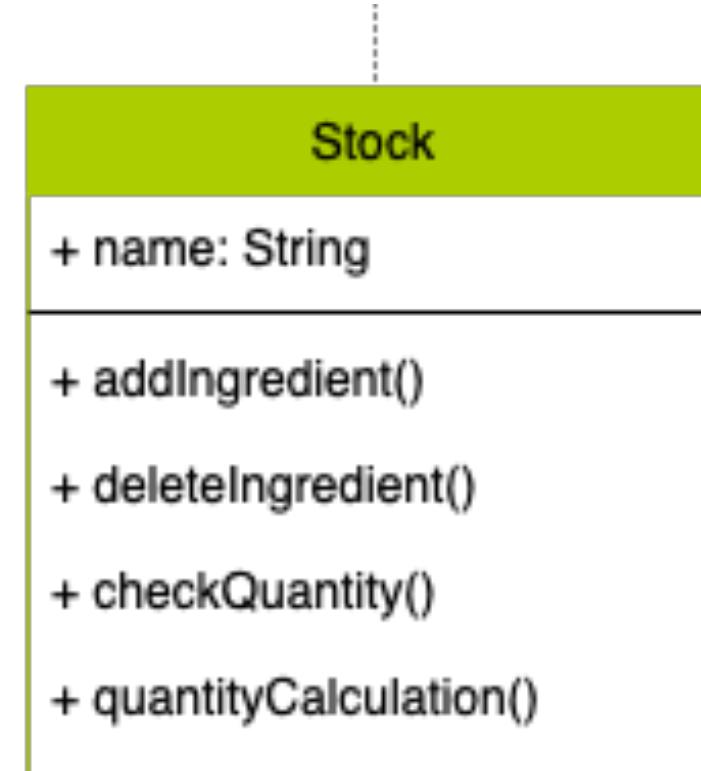


La Classe **Pizza** est associée aux classes **Oder** et **Ingrédients**.

Les Attributs de cette Classe sont :

- name : nom de la **Pizza**
- Price : le prix de la **Pizza**

Il sera possible d'ajouter des pizzas ou de les supprimer

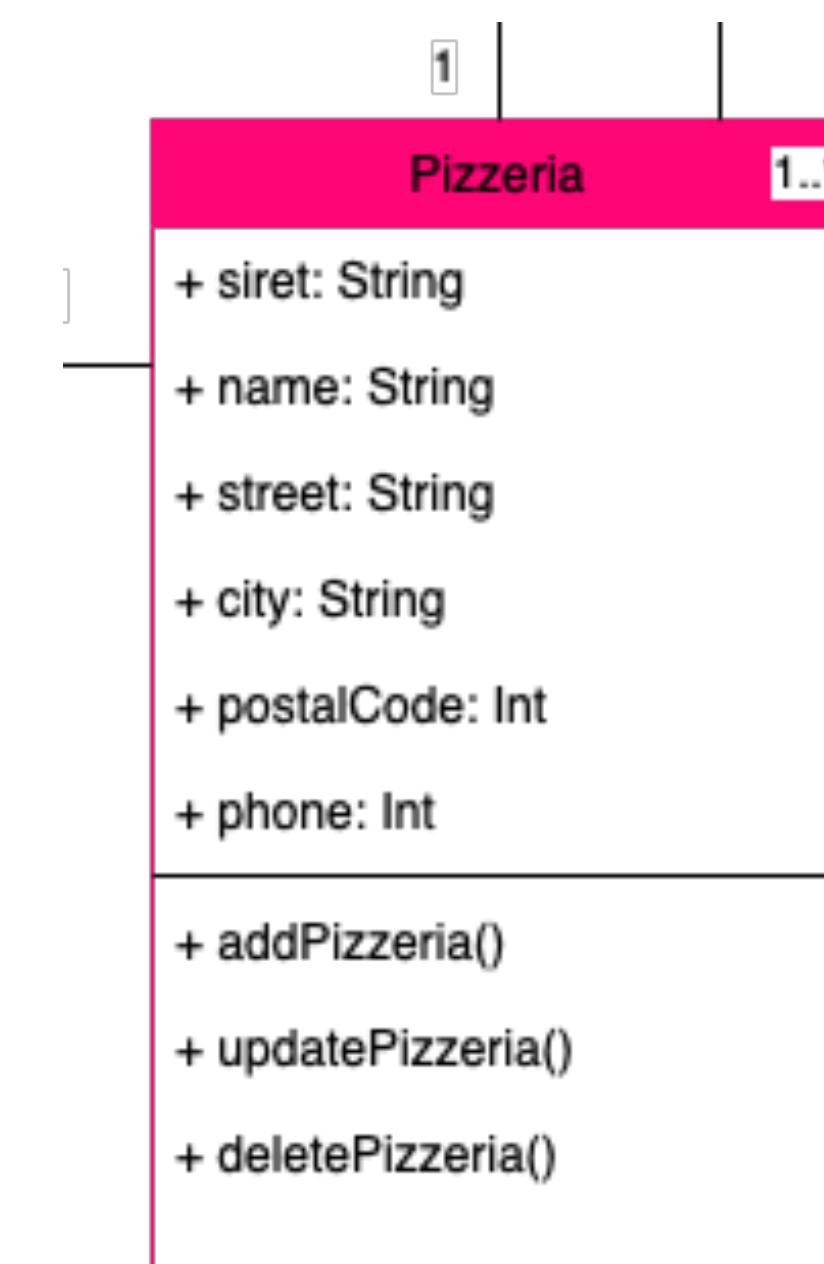


La Classe **Stock** est une Classe d'association entre **Ingrédients** et **pizzeria**

Les Attributs de cette Classe sont :

- name : date de la facture

Il sera possible d'ajouter, supprimer et vérifier le **stock**



La Classe **Pizzeria**

Les Attributs de cette Classe sont :

- Siret : numero de siret
- Name : un nom
- Street, city, postalCode : une adresse
- Phone : numéro de téléphone

Il sera possible d'ajouter, de modifier ou de supprimer des **Pizzéries**

La Classe **Ingrédients** est associée aux classes **Pizza** et **Stock**

Les Attributs de cette Classe sont :

- name : nom de l'**ingrédient**
- Quantité : la quantité **l'ingrédient**

Une fonction sera ajouté pour que dès qu'une commande est passé les ingrédients utilisé seront supprimer du stock. Ce qui évitera de proposer des pizzas ou il manquera des ingrédients

Le modèle physique de donnée : MPD

Cahier des charges

6

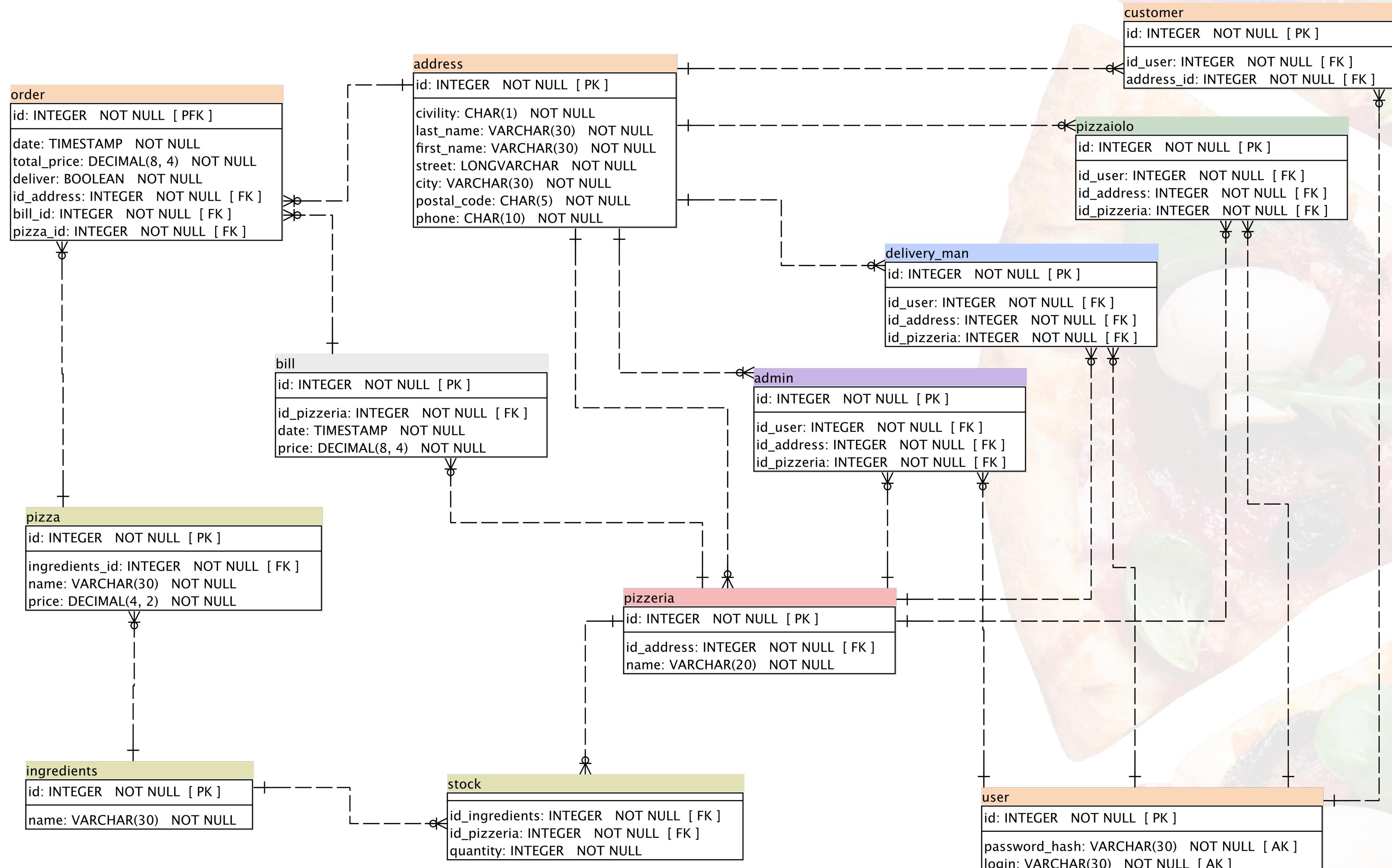
6 & 9 - Le modèles physique de donnée / 10 - le base de donnée / 11 - Le diagramme de composants / 12 - Le diagramme de déploiement

Le MPD

(modèle physique de donnée)

est un outil de conception de
base de données.

Il va nous permettre de
schématiser notre futur base
de données avec des Tables
et des colonnes et des
associations de Tables



Le modèle physique de donnée : Descriptif

Cahier des charges

7

6 & 9 - Le modèles physique de donnée / 10 - le base de donnée / 11 - Le diagramme de composants / 12 - Le diagramme de déploiement

La Table **address** :

Elle regroupe toutes les informations pour **les adresses**. Toute la Table sera NOT NULL.

Une clé primaire PK « id », en auto-incrémentation et de type INTEGER (Int)

- Civility de type CHAR(1) = 1 caractère.
- last_name, first_name et city VARCHAR(30) <= 30 caractères
- street LONGVARCHAR = caractères illimités
- postal_code CHAR(5) = 5 caractères (ex 75000)
- Phone CHAR(10) = 10 caractères (0231961919)

Les Table **customer**, **pizzaiolo**, **delivery_man**, et **admin** :

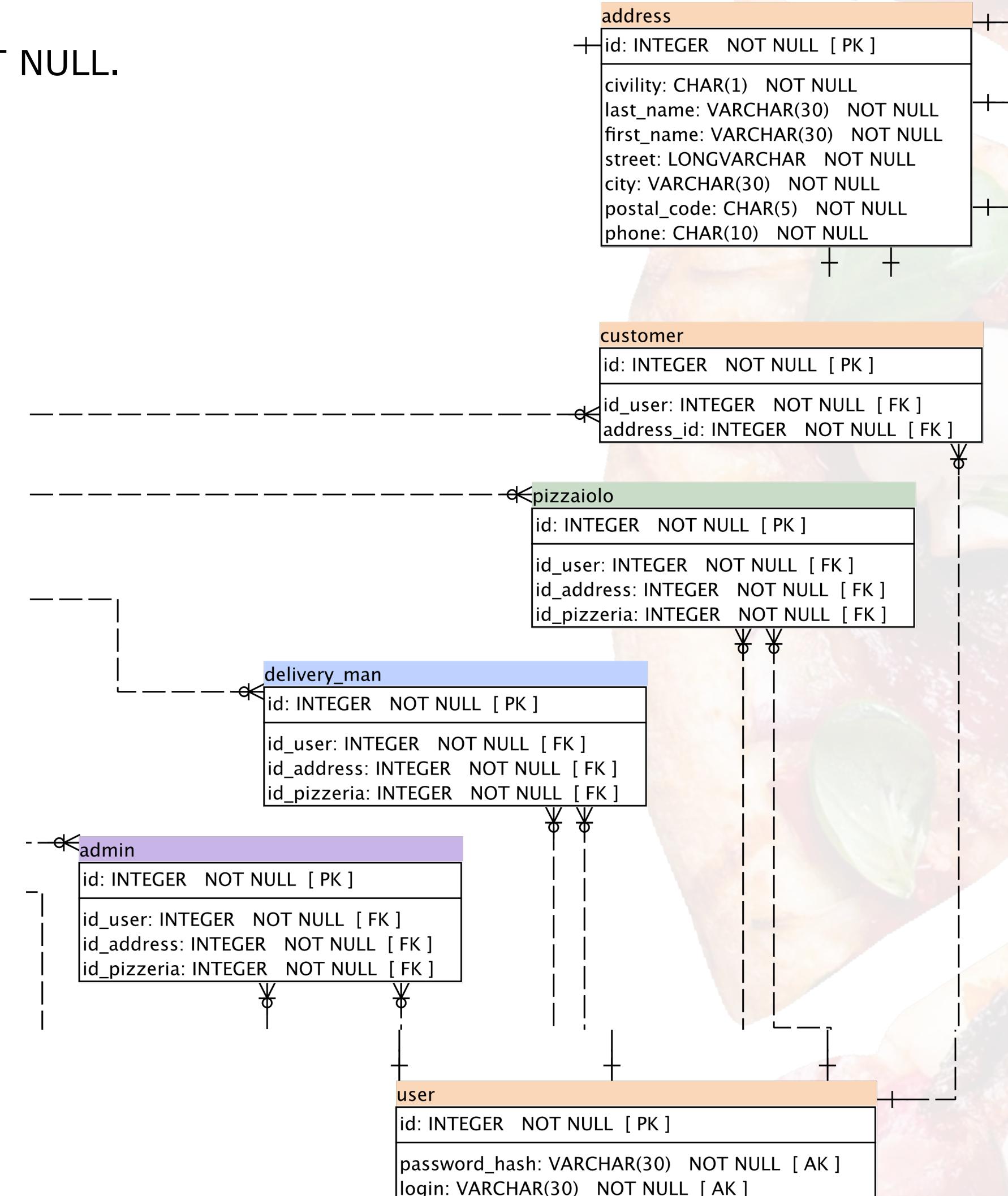
Elles auront toutes une clé primaire « id », en auto-incrémentation et de type INTEGER (Int) NOT NULL.

Avec des associations clés étrangères FK sur **id_user**, **id_address**, et pour tous sauf **customer** une association avec **id_pizzeria**

La Table **user** :

Elle regroupe toutes les informations pour la connexion de chaque utilisateur. Toute la Table sera NOT NULL. La PK « id », en auto-incrémentation et de type INTEGER (Int)

- password_hash de style VARCHAR(30) = le mot de passe <= 30 caractères
- Login VARCHAR(30) = identifiant <= 30 caractères



Le modèle physique de donnée : Descriptif

Cahier des charges

8

6 & 9 - Le modèles physique de donnée / 10 - le base de donnée / 11 - Le diagramme de composants / 12 - Le diagramme de déploiement

La Table **order** :

Elle regroupe toutes les informations concernant les **commandes**. Toute la Table sera NOT NULL.

Une clé primaire PK « id », en auto-incrémentation et de type INTEGER (Int)

- date de type TIMESTAMP = pour la date et l'heure
- total_price DECIMAL(8, 4) = il y aura 8 chiffres en tout
- Deliver BOOLEAN = pour avoir le choix du livreur

Avec des associations FK **id_address**, **bill_id**, **pizza_id**

| order |
|-------------------------------------|
| id: INTEGER NOT NULL [PFK] |
| date: TIMESTAMP NOT NULL |
| total_price: DECIMAL(8, 4) NOT NULL |
| deliver: BOOLEAN NOT NULL |
| id_address: INTEGER NOT NULL [FK] |
| bill_id: INTEGER NOT NULL [FK] |
| pizza_id: INTEGER NOT NULL [FK] |

La Table **bill** :

Elle regroupe toutes les informations de chacune des **factures**. Toute la Table sera NOT NULL. La PK

« id », en auto-incrémentation et de type INTEGER (Int)

- date de type TIMESTAMP = pour la date et l'heure
- price DECIMAL(8, 4) = il y aura 8 chiffres en tout pour le montant de la facture

Elle aura en association FK **id_pizzeria**

| bill |
|--------------------------------------|
| id: INTEGER NOT NULL [PK] |
| id_pizzeria: INTEGER NOT NULL [FK] |
| date: TIMESTAMP NOT NULL |
| price: DECIMAL(8, 4) NOT NULL |

La Table **pizzeria** :

Elle regroupe toutes les informations des **pizzéries**. Toute la Table sera NOT NULL. La PK « id », en auto-incrémentation et de type INTEGER (Int)

- name VARCHAR(20) <= 20 caractères pour le nom

Elle aura en association FK **id_address**

| pizzeria |
|-------------------------------------|
| id: INTEGER NOT NULL [PK] |
| id_address: INTEGER NOT NULL [FK] |
| name: VARCHAR(20) NOT NULL |

Le modèle physique de donnée : Descriptif

Cahier des charges

9

10 - la base de donnée / 11 - Le diagramme de composants / 12 - Le diagramme de déploiement

La Table **pizza** :

Elle regroupe toutes les informations pour les **pizzas**. Toute la Table sera NOT NULL.

Une clé primaire PK « id », en auto-incrémentation et de type INTEGER (Int)

- name VARCHAR(30) <= 30 caractères pour le nom de chacune des pizzas
- price DECIMAL(4, 2) = il y aura 4 chiffres pour le prix de chacune des pizzas

Avec une association clé étrangère FK **ingredients_id**

| + | pizza | + |
|---|---|---|
| | id: INTEGER NOT NULL [PK] | |
| | ingredients_id: INTEGER NOT NULL [FK] | |
| | name: VARCHAR(30) NOT NULL | |
| | price: DECIMAL(4, 2) NOT NULL | |

La Table **ingredients** :

Elle regroupe les informations pour les **ingrédients** utilisés. Toute la Table sera NOT NULL. La PK « id », en auto-incrémentation et de type INTEGER (Int)

- name VARCHAR(30) <= 30 caractères pour le nom des ingrédients

| + | ingredients | + |
|---|-----------------------------|---|
| | id: INTEGER NOT NULL [PK] | |
| | name: VARCHAR(30) NOT NULL | |

La Table **stock** :

C'est une Table association entre la Table **ingrédients** et **pizzeria**

- quantity INTEGER = quantité en int. Associée à l'**id_ingredients** et **id_pizzeria**

| + | stock | + |
|---|---|---|
| - | id_ingredients: INTEGER NOT NULL [FK] | - |
| - | id_pizzeria: INTEGER NOT NULL [FK] | - |
| | quantity: INTEGER NOT NULL | |

Créations des tables

```
CREATE TABLE public.address (
    id integer NOT NULL,
    civility character(1) NOT NULL,
    last_name character varying(30) NOT NULL,
    first_name character varying(30) NOT NULL,
    street text NOT NULL,
    city character varying(30) NOT NULL,
    postal_code character(5) NOT NULL,
    phone character(10) NOT NULL
);
```

Dans mon DUMP (sauvegarde de ma base de données) au format SQL.
Nous pouvons lire les créations des tables.

L'exemple ci dessus est la table adresse avec :

Un ID qui est autoincrémenté, NOT NULL qui ne peut pas être vide.
Il y a un nombre de caractères allant jusqu'à 30 (varying) et pour d'autres un nombre strictement égale à 5 pour le code postal, 10 pour le n° de téléphone. Et pour cette table toutes les informations sont en NOT NULL

Intégrations des données

```
INSERT INTO public.address (civility, last_name,
first_name, street, city, postal_code, phone)
VALUES
('m', 'pan', 'peter', '5 rue de la grande ours',
'paris', '75000', '0123456789'),
('m', 'capitaine', 'crochet', '2 rue de la piraterie',
'caen', '14000', '0123456780'),
('f', 'fée', 'clochette', '1 chemin de la foret',
'ouistreham', '14150', '0123456781'),
('f', 'darling', 'wendy', '8 impasse de la poussière de
fée', 'hérouville saint clair', '14200', '0123456782'),
('m', 'rebelle', 'merida', '10 chemin du chateau',
'caen', '140009', '0123456784'),
('m', 'rebelle', 'merida', '10 chemin du chateau',
'caen', '140009', '0123456784'),
('m', 'prince', 'ali', '2 rue de la dune', 'hérouville
saint clair', '14200', '0123456799');
```

Quand je rentre des données dans la bases je renseigne tous ce qui était demandé dans la table.

À savoir : civility, last_name, first_name, street, city, postal_code et phone. Vous constaterez que l'ID n'apparait pas. C'est normal puisqu'il est autoincrémenté et donc pour chaque nouvelle ligne, il lui sera attribué un N° ID unique

Le Diagramme de composants

12 - Le diagramme de déploiement

Cahier des charges

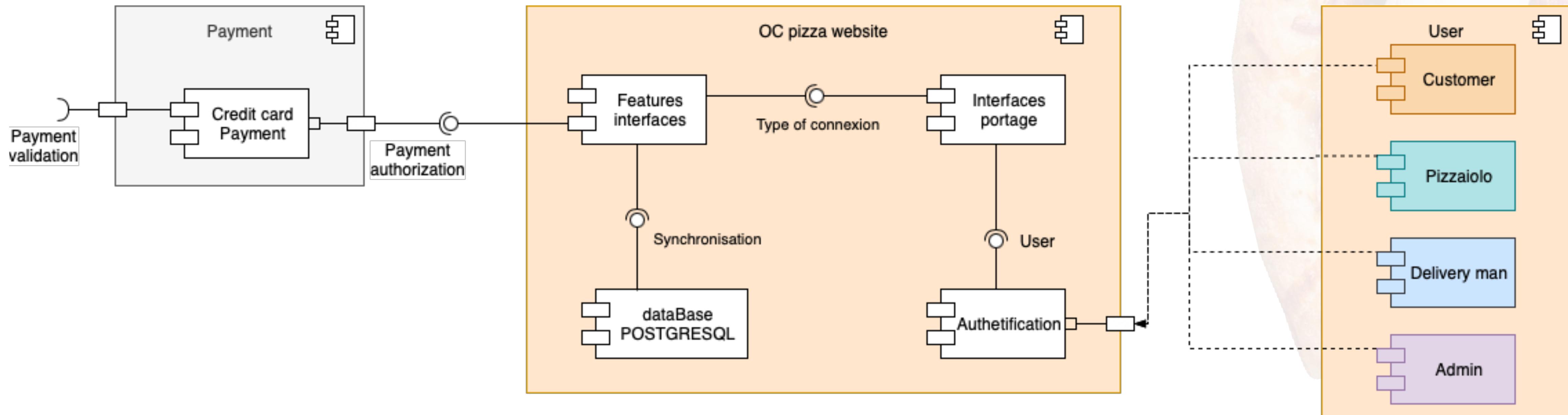
11

Le Diagramme de composants

Décrit l'organisation entre les différents composants d'un système.

Les packages sont :

Payment, website et user



Le Diagramme de déploiement

FIN

Cahier des charges

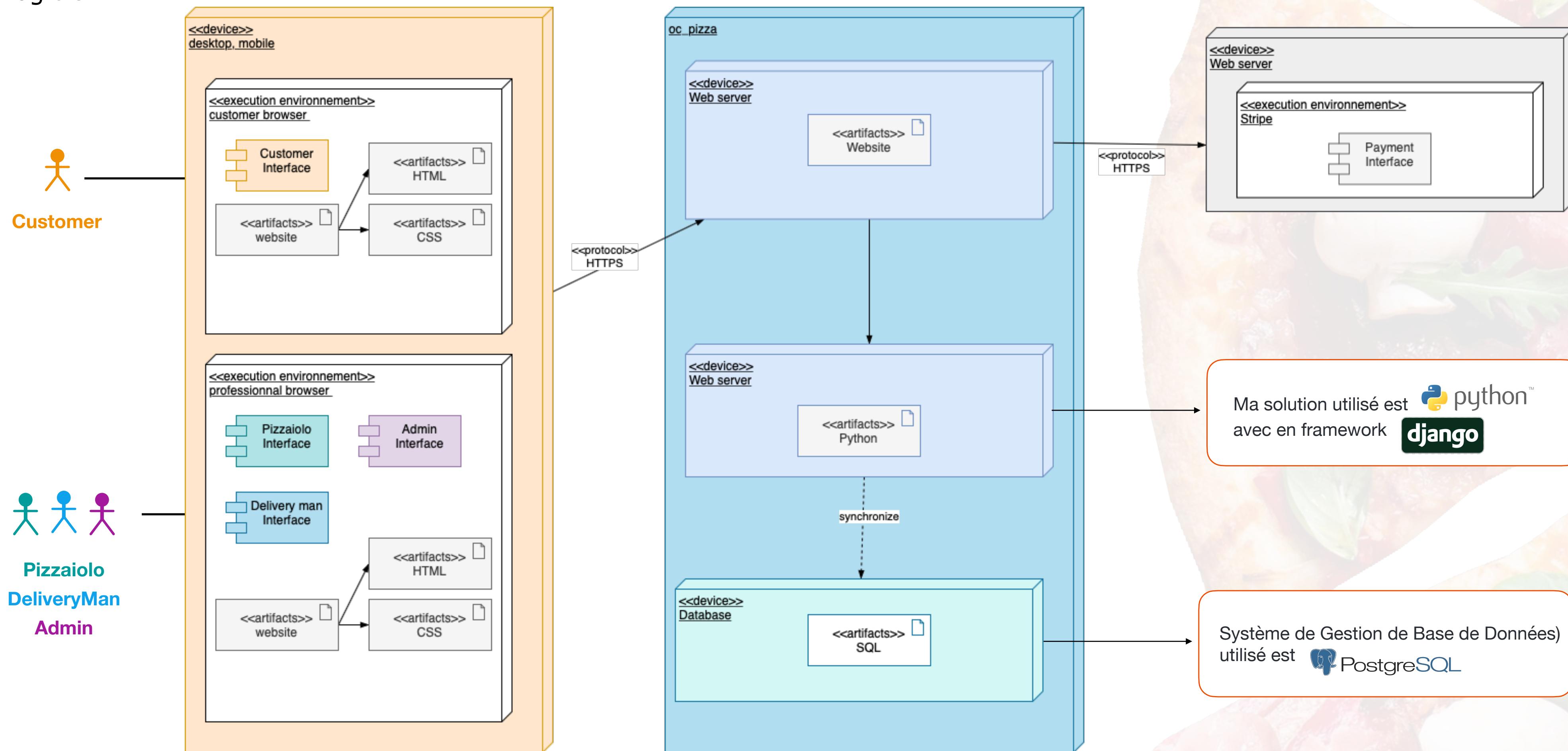
12

Le Diagramme de déploiement

Il décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. On appelle artefact l'information qui est générée par le logiciel

Composant :
rectangle avec deux onglets indiquant un élément logiciel.

Artefact : produit développé par le logiciel, symbolisé par un rectangle avec le nom et le mot « artefact » entourés de flèches doubles.



OCPizza

