Maxime Jacquette

C Pizza



Document (PDF) des spécifications techniques.

##### **Projet 6 Openclassrooms : Concevez la solution technique d’un système de gestion de pizzeria**

Lien Github : <https://github.com/Maximedu13/Concevez-la-solution-technique-d-un-syst-me-de-gestion-de-pizzeria>

Table des matières

[I. Votre entreprise : contexte et besoins 3](#_Toc336681)

[a. Rappel du contexte 3](#_Toc336682)

[b. Rappel des besoins attendus 3](#_Toc336683)

[II. Solution fonctionnelle 5](#_Toc336684)

[III. Domaine fonctionnel 5](#_Toc336685)

[a. Le diagramme de classes 5](#_Toc336686)

[b. Les objets 6](#_Toc336687)

[La classe « PizzaStore » 6](#_Toc336688)

[La classe « Address » 7](#_Toc336689)

[La classe « Person » 8](#_Toc336690)

[La classe « Team » 8](#_Toc336691)

[La classe « Client » 9](#_Toc336692)

[La classe « OrderPizzeria » 10](#_Toc336693)

[La classe « Billing » 11](#_Toc336694)

[La classe « Stock » 11](#_Toc336695)

[La classe « Ingredient » 11](#_Toc336696)

[La classe « Recipe » 12](#_Toc336697)

[La classe « Pizza » 13](#_Toc336698)

[La classe « Supplier » 13](#_Toc336699)

[IV. Le modèle physique de données (MPD) 14](#_Toc336700)

[a. Le MPD d’OC Pizza 14](#_Toc336701)

[b. Les tables 14](#_Toc336702)

[La table « PizzaStore » 14](#_Toc336703)

[La table « Address » 15](#_Toc336704)

[La table « Person » 15](#_Toc336705)

[La table « Team » 16](#_Toc336706)

[La table « Client » 16](#_Toc336707)

[La table « OrderPizzeria » 17](#_Toc336708)

[La table « Billing » 17](#_Toc336709)

[La table « Stock » 18](#_Toc336710)

[La table « Ingredient » 18](#_Toc336711)

[La table « Recipe » 18](#_Toc336712)

[La table « Pizza » 19](#_Toc336713)

[La table « Supplier » 19](#_Toc336714)

[V. Les différents composants du système et les composants externes utilisés par le domaine fonctionnel et leur interaction 20](#_Toc336715)

[a. Le diagramme de composants 20](#_Toc336716)

[b. Descriptif des composants 20](#_Toc336717)

[Le navigateur web 20](#_Toc336718)

[Le catalogue 20](#_Toc336719)

[Le tableau de bord 21](#_Toc336720)

[Le panier 21](#_Toc336721)

[L’authentification 21](#_Toc336722)

[Les notifications 21](#_Toc336723)

[Les commandes 21](#_Toc336724)

[Les clients 21](#_Toc336725)

[Les comptes 21](#_Toc336726)

[L’inventaire 21](#_Toc336727)

[Le paiement 21](#_Toc336728)

[La base de données MySQL 21](#_Toc336729)

[VI. L’architecture de déploiement 21](#_Toc336730)

[a. Le diagramme de déploiement 21](#_Toc336731)

[b. Environnements utilisateurs 22](#_Toc336732)

# Votre entreprise : contexte et besoins

## Rappel du contexte

« OC Pizza » est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter.

Vous comptez déjà 5 points de vente et prévoyez d’en ouvrir au moins 3 de plus d’ici la fin de l’année. Un des responsables du groupe a pris contact avec nous afin de mettre en place un système informatique.

## Rappel des besoins attendus

Celui-ci déployé dans toutes ses pizzerias vous permettrait notamment :

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

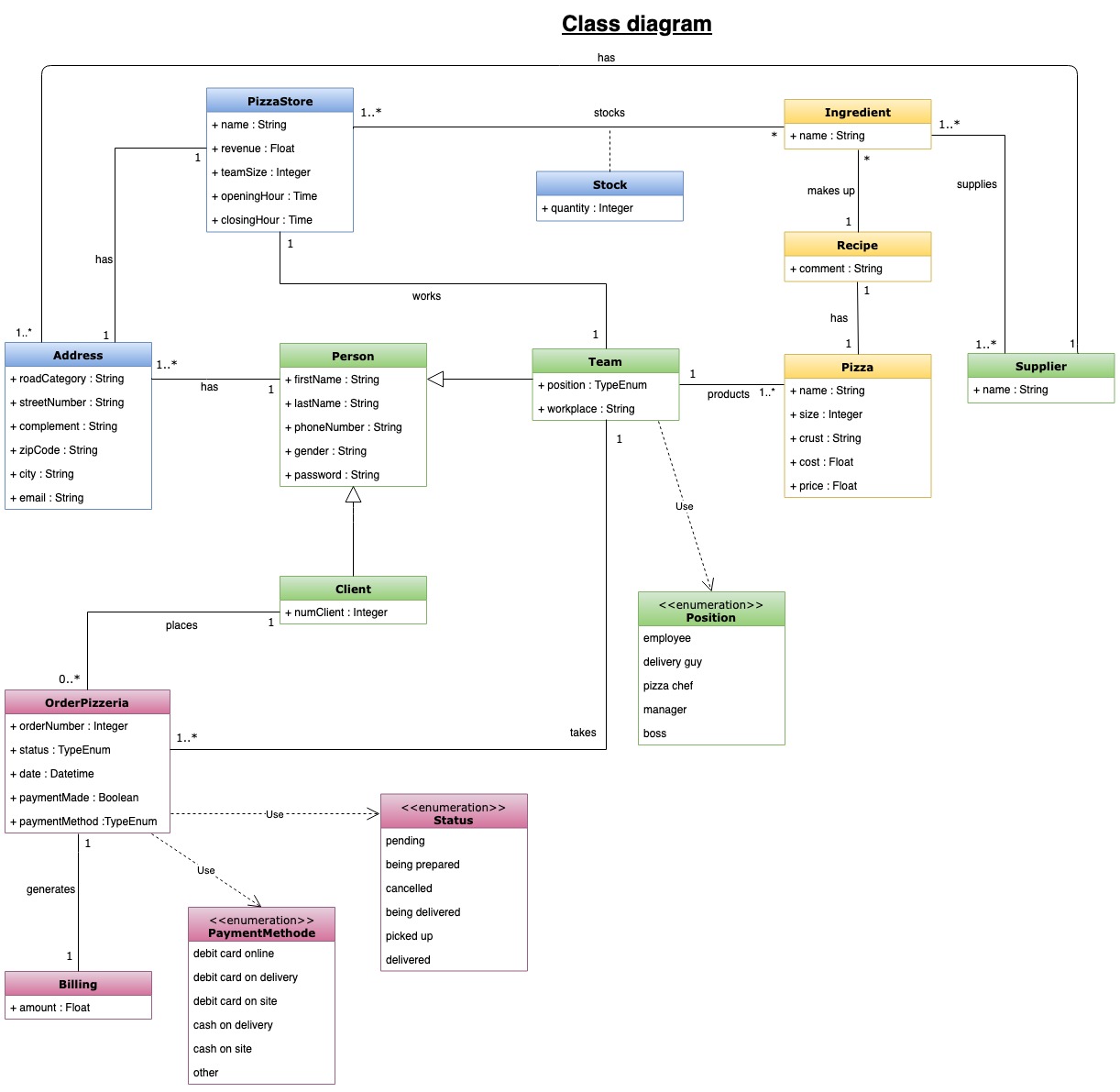
Vous avez déjà fait une petite prospection et les logiciels existants que vous avez pu trouver ne vous conviennent pas.

# Solution fonctionnelle

La solution a été présentée lors du projet 4 « Analysez les besoins de votre client pour son groupe de pizzerias ». Elle est consultable via le lien suivant : <https://github.com/Maximedu13/Analysez-les-besoins-de-votre-client-pour-son-groupe-de-pizzerias>

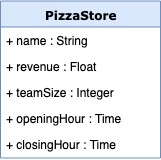
# Domaine fonctionnel

## Le diagramme de classes

****

## Les objets

### La classe « PizzaStore »



Votre groupe « OC Pizza » compte actuellement cinq points de vente, mais vous comptez en ouvrir plusieurs autres d’ici la fin de l’année et cela au vu de la bonne rentabilité de votre entreprise. Cette classe [**PizzaStore**(Point de vente)] regroupe les caractéristiques de chacun de ceux-ci. Elle possède cinq attributs :

* **name**qui correspond au nom du point de vente et qui permet de différencier les pizzerias géographiquement.
* **revenue**est un attribut qui différencie les différents points de vente selon les revenus de ceux-ci. Ils permettront ainsi de voir lesquels de vos points de vente sont les plus et moins rentables.
* **teamSize**est un attribut représentant la taille de l’équipe (nombre de salariés) qui travaille dans le point de vente
* **openingHour** est l’attribut décrivant les horaires d’ouverture de la pizzeria.
* **closingHour**est l’attribut décrivant les horaires de fermeture de la pizzeria.

La classe a une association one- to-one avec la classe **Address** (Adresse),chaque pizzeria possédant (has) une adresse, et chaque adresse permettant de différencier géographiquement les différents points de vente (cardinalité 1,1).

Elle possède également une association one- to-one avec la classe **Team** (Équipe), car une équipe travaille (works) dans un, et un seul point de vente à la fois. Chaque équipe est affecté à un point de vente unique (cardinalité 1, 1).

Sa troisième relation, avec la classe **Ingredient** (Ingrédient), est une association many-to-many (cardinalité \*, 0..\*) car le point de vente peut stocker (stocks) une infinité d’ingrédients, mais un ingrédient particulier peut être stocké dans 0 ou plusieurs points de vente.

### La classe « Address »



La classe **Address** regroupe les attributs d’adresses des points de vente et de l’utilisateur (personnel et clients), soient les attributs suivants :

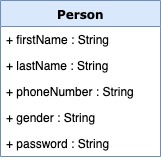
* *roadCategory* désigne le type de route. Il peut s’agir d’une rue, d’une avenue, d’un boulevard, d’une impasse etc...
* *streetNumber*est le numéro associé au catégorie de route ci-dessus.
* *complement* correspond au complément d’adresse. Il est facultatif.
* *zipCode*est le code postal.
* *city*est l’attribut de la ville.
* *email* est le courriel.

Cette classe a une association one- to-one avec la classe **PizzaStore** (Point de vente), association expliquée précédemment.

Elle possède également une association dite many-to-many avec la classe **Person** (Personne), en effet une personne peut avoir (has) une ou plusieurs adresses et chaque adresse peut être reliée à une ou plusieurs personnes vivant sous le même toit (un foyer par exemple) (cardinalité 1..\*, 1..\*).

Enfin, elle est reliée à la classe **Supplier** (Fournisseur) par une association one-to-many car selon la localisation de vos points de vente, il est fort probable que le même fournisseur possède (has) plusieurs entrepôts de stockage. En revanche, l’adresse de celui-ci permet de l’identifier. Il ne partage pas son adresse avec un autre fournisseur par exemple (cardinalité 1..\*, 1).

### La classe « Person »

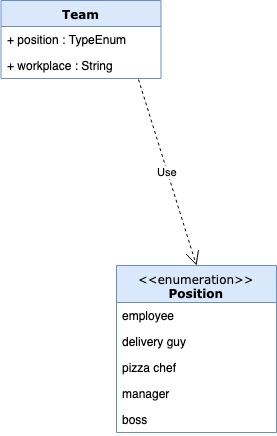


Cette classe est une généralisation des classes **Team** (Équipe) et **Client** (Client), en effet chaque membre de l’équipe (dit personnel), et client possède les attributs suivants :

* *firstName***,** un prénom.
* *lastName,*un nom de famille.
* *phoneNumber***,** un numéro de téléphone.
* *gender***,** un sexe.
* *password,*un mot de passe.

Elle est aussi associée à la classe **Address** (Adresse)**,** association many-to-many, détaillée précédemment.

### La classe « Team »



La classe **Team** (Équipe) est une spécialisation de la classe **Person** (Personne).

La table **enumeration** définit les seules valeurs possibles de l’attribut *Position*(*poste occupé*), soient :

* *employee* (employée).
* *delivery guy*(livreur-se).
* *pizza chef*(pizzaiolo-a).
* *manager*(manager).
* *boss* (patron-ne).

La classe possède également un autre attribut :

* *workplace*, le lieu de travail, permettant d’identifier une équipe à son point de vente.

La classe possède une association one-to-one avec la classe **PizzaStore** (Point de vente) expliquée précédemment.

**Team** (Équipe) est reliée à la classe **Pizza** (Pizza) par une association many-to-many (cardinalité 0..\*, 0..\*). En effet, l’équipe produit (products) zéro ou plusieurs pizzas en même temps et une pizza donnée peut être produite par zéro ou plusieurs équipes à un instant t.

**Team** (Équipe) possède également une association avec la classe **OrderPizzeria** (Commande). Cette association est de type one-to-many : l’équipe prend (takes) zéro ou plusieurs commandes à la fois. La commande en question n’est prise que par une équipe (cardinalité 0..\*, 1).

### La classe « Client »

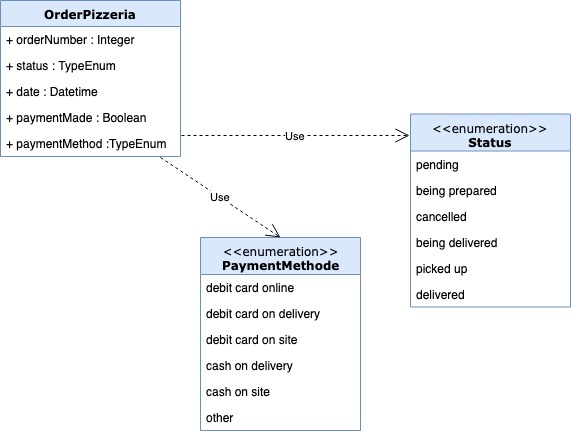


La classe **Client** (Client), tout comme la classe **Team** (Équipe) est une spécialisation de la classe **Person** (Personne). Outre des attributs de cette classe dont elle hérite, elle possède un attribut :

* *numClient*, un numéro client unique.

Elle a une association de type one-to-many avec la classe **OrderPizzeria** (Commande). Un client peut passer (places) zéro ou plusieurs commandes et une commande n’est passée que par un seul client (cardinalité 0..\*, 1).

### La classe « OrderPizzeria »



La classe **OrderPizzeria** (Commande) correspond aux commandes passées par les clients du groupe OC Pizza, ou par l’équipe OC Pizza.

Elle possède les attributs suivants :

* *OrderNumber,* un numéro de commande unique.
* *Date,* la date de la commande.
* *paymentMade,* un booléen vérifiant si le paiement a été effectué ou non.

La table **enumeration** définit les seules valeurs possibles de l’attribut *Status* (Statut de la commande), soient :

* *pending* (en attente)
* *being prepared* (en préparation)
* *cancelled* (annulée)
* *being delivered* (en cours de livraison)
* *picked up* (récupérée sur place)
* *delivered* (livrée)

La table **enumeration** définit également les seules valeurs possibles de l’attribut *PaymentMethod* (moyen de paiement), soient :

* *debit card online*(paiement en carte bancaire en ligne)
* *debit card on delivery*(paiement en carte bancaire lors de la livraison)
* *debit card on site*(paiement en carte bancaire sur place)
* *cash on delivery*(paiement en liquide lors de la livraison)
* *cash on site*(paiement en liquide sur place)
* *other*(autre [par ex. : tickets resto...])

La classe est associée aux classes **Client** (Client) et **Team** (Équipe)*,* associations décrites précédemment.

Outre celles-ci, elle en possède une, dite one-to-one avec la classe **Billing** (Facture). En effet, chaque commande génère (generates) une et une seule facture à la fois. Une facture (Billing) est générée par une seule commande à la fois (cardinalité 1, 1).

### La classe « Billing »



La classe **Billing** (Facture) ne possède qu’un seul attribut :

* *amount*, son montant.

Elle a une association avec la classe **OrderPizzeria** (commande), expliquée ci-dessus.

### La classe « Stock »



**Stock** est une classe d’association qui ajoute l’attribut *quantity* (quantité)à l’association many-to-many des classes **PizzaStore (Point de vente)** et **Ingredient** (Ingrédient). L’association a été précédemment détaillée.

### La classe « Ingredient »



La classe **Ingredient** (Ingrédient) permettra à la pizzeria de lister tous les ingrédients utiles à la fabrication des pizzas commercialisées.

**Ingredient** (Ingrédient) ne possède qu’un seul attribut :

* *name*, le nom de l’ingrédient.

La classe a trois associations.

Celle avec la classe **PizzaStore** (Point de vente) a été précédemment expliquée.

Elle possède deux autres associations. L’une d’entre elles, associe **Ingredient** à la classe **Recipe** (Recette). Cette association est de type many-to-many. Chaque recette contient (makes up) plusieurs ingrédients, et un ingrédient peut être présent dans une ou plusieurs recettes (cardinalité 1..\*, \*).

Sa troisième association se fait avec la classe **Supplier** (Fournisseur). Vos fournisseurs vous approvisionne (supplies) d’un ou plusieurs ingrédients à la fois, et ce même ingrédient peut être fourni par un ou plusieurs de vos fournisseurs. L’association est donc de type many-to-many (cardinalité 1..\*, 1..\*)

### La classe « Recipe »



Cette classe contient les instructions destinées au pizzaiolo-a. Elle ne contient qu’un seul attribut :

* *comment*, un commentaire particulier concernant la réalisation de la pizza.

Elle est associée avec la classe **Ingredient** (Ingrédient). L’association a été decrite ci-dessus.

Elle a également une association de type one-to-one avec la classe **Pizza** (Pizza). En effet, une pizza a (has) une et une seule recette [celle de la pizzeria], et une recette est spécifique à une et une seule pizza (cardinalité 1, 1).

### La classe « Pizza »



La classe **Pizza** (Pizza) regroupe tous les attributs caractéristiques d’une pizza :

* *name*, un nom attribué.
* *size*,une taille.
* *crust,* un type de croute (fine, épaisse, croustillante etc...)
* *cost,* un cout.
* *price*, un prix.

Elle a des associations avec les classes **Team** (Équipe) et **Recipe** (Recette), décrites ci-dessus.

### La classe « Supplier »



La classe **Supplier** (Fournisseur) fournit l’attribut caractéristique d’un fournisseur :

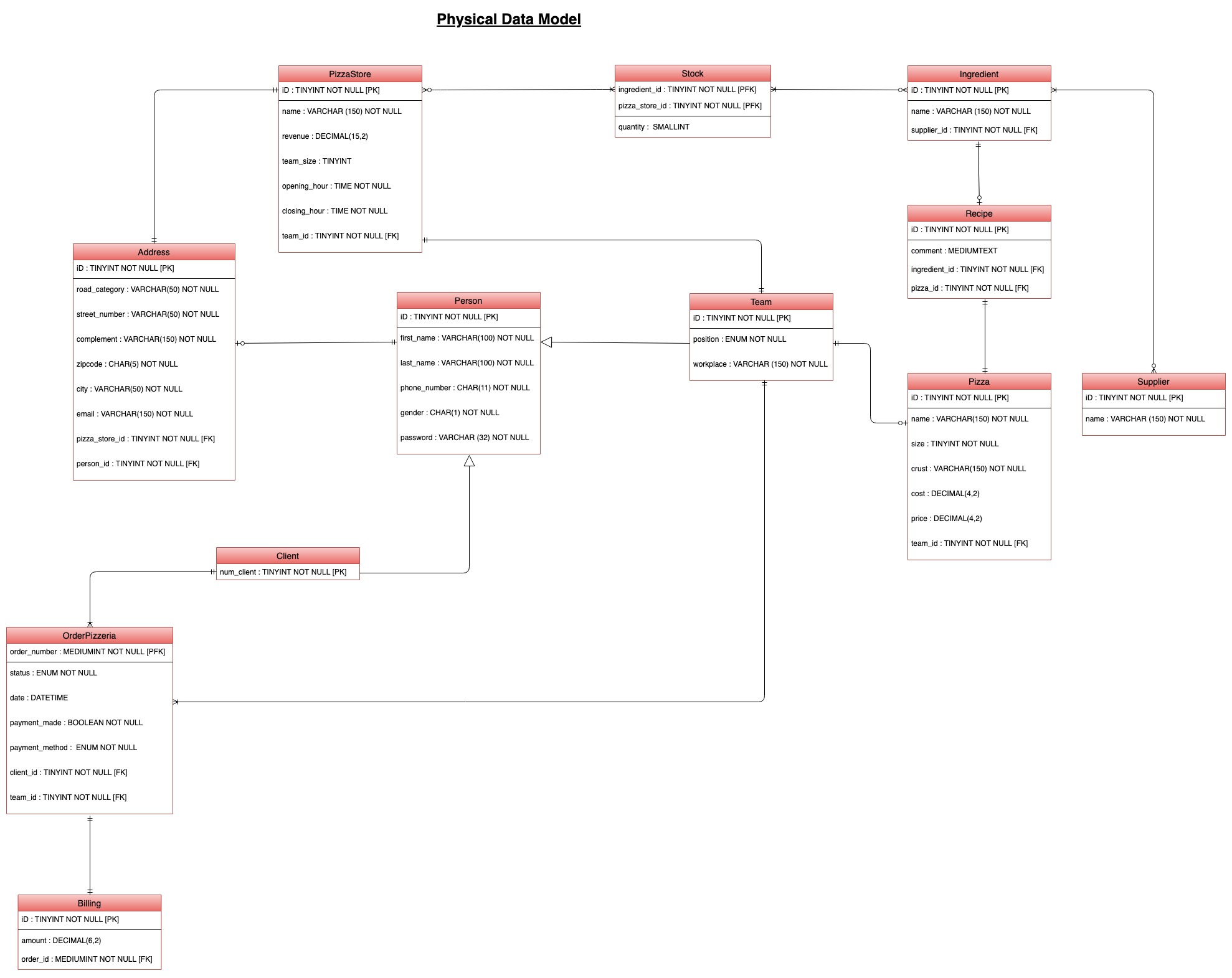
* *name*, le nom du fournisseur, de l’entreprise qui vous approvisionne en matières premières.

Cette classe est associée aux classes **Address** (Adresse) et **Ingredient** (Ingrédient), associations expliquées précédemment.

# Le modèle physique de données (MPD)

## Le MPD d’OC Pizza

À partir du diagramme de classes, nous pouvons élaborer votre MPD. Nous justifierons ici les types de données utilisés



## Les tables

### La table « PizzaStore »

Cette **table** regroupe les informations de tous vos points de vente.

La **table** dispose d’une clé primaire *iD*, avec un AUTO\_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacun de vos points de vente. Pour le moment votre entreprise n’en possédez que cinq, il est donc inutile de lui affecter un type MEDIUMINT ou SMALLINT.

Pour les autres *attributs*, le type est :

* + VARCHAR, limité à 150 caractères pour *name*.
  + DECIMAL, limité à 15 chiffres pour la partie entière et deux chiffres pour la partie décimale, pour *revenue*.
  + TINYINT, pour *team\_size* car il est peu probable, ne serait-ce qu’une de vos pizzerias, compte plus de 255 salarié-es.
  + TIME, pour *opening\_hour* et *closing\_hour*, car ici nous n’avons pas besoin d’un type DATETIME, l’heure suffit.

L’attribut *team\_id* est une clé étrangère se référant à l’*iD* correspondant de la table **Team**.

Tous les attributs de la table **PizzaStore** doivent être renseignés (NOT NULL), sauf *revenue*, et *team\_size*.

### La table « Address »

Cette **table** regroupe toutes les adresses de vos points de ventes, de vos clients, de vos fournisseurs ainsi que des salariés de votre entreprise.

La **table** dispose d’une clé primaire *iD*, avec un AUTO\_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacune des adresses.

Pour les autres *attributs*, le type est :

* + VARCHAR, limité à 50 caractères pour *road\_category, street\_number,* et *city.*
  + VARCHAR, limité à 150 caractères pour *complement* et *email.*
  + CHAR, limité à 5 caractères pour *zipcode*.

Les attributs *pizza\_store\_id* et *person\_id* sont des clés étrangères se référant respectivement à l’*iD* de la table **PizzaStore et** à l’*iD* de la table **Person**.

*pizza\_store\_id* est de typeTINYINT *et person\_id* de type MEDIUMINT.

Tous les attributs de la table **Address** doivent être renseignés (NOT NULL).

### La table « Person »

Cette **table** regroupe les informations de toutes les personnes, en tant qu’individus.

La **table** dispose d’une clé primaire *iD*, avec un AUTO\_INCREMENT, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacune des personnes. Nous pensons qu’un TINYINT serait insuffisant, au vu de la taille de vos équipes et du nombre de clients.

Pour les autres *attributs*, le type est :

* + VARCHAR, limité à 100 caractères pour *name* et *last\_name*.
  + CHAR, limité à 11 caractères pour *phone\_number*.
  + CHAR, limité à 1 caractère pour *gender* (H ou F).
  + VARCHAR, limité à 32 caractères pour *password,* un mot de passe crypté.

Tous les attributs de la table **Person** doivent être renseignés (NOT NULL).

### La table « Team »

Cette **table** regroupe les informations des membres de toutes les équipes.

La **table** dispose d’une clé primaire *iD*, avec un AUTO\_INCREMENT, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacun des salariés. Nous pensons qu’un TINYINT serait insuffisant, au vu de la taille de vos équipes, et devant l’éventualité que cette équipe change.

Pour les autres *attributs*, le type est :

* + ENUM pour *position* qui peut prendre comme valeur l’un des cinq postes occupés par le salarié.
  + VARCHAR, limité à 150 caractères pour *workplace.*

Tous les attributs de la table **Team** doivent être renseignés (NOT NULL).

### La table « Client »

Cette **table** regroupe les informations de chacun de vos clients.

**Client** dispose d’une clé primaire *num\_client*, avec un AUTO\_INCREMENT, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacun de vos clients.

Cet attribut doit être renseigné (NOT NULL).

### La table « OrderPizzeria »

Cette **table** regroupe toutes les informations concernant les commandes.

La **table** dispose d’une clé primaire et étrangère *order\_number*, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacune des commandes.

Pour les autres *attributs*, le type est :

* + ENUM pour *status* et *payment\_method* qui peuvent prendre respectivement comme valeurs les statuts de commande et les moyens de paiement précédemment listés.
  + DATETIME, pour *date*. Ici il est primordial d’avoir l’heure, le jour etc. de la commande.
  + BOOLEAN, pour *payment\_made*, qui se charge de verifier si le paiement a eu lieu ou non (True ou False).

Les attributs *client\_id* et *team\_id* sont des clés étrangères de type MEDIUMINT se référant respectivement à l’*iD* de la table **Client** et à l’*iD* de la table **Team**.

Tous les attributs de la table **OrderPizzeria** doivent être renseignés (NOT NULL).

### La table « Billing »

La **table** regroupe toutes les informations concernant les factures générées par les commandes et envoyées aux clients.

La **table** dispose d’une clé primaire *iD*, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacune des factures.

Pour les autres *attributs*, le type est :

* + DECIMAL, limité à 6 chiffres pour la partie entière et 2 chiffres pour la partie décimale, pour *amount*.

Outre ces attributs, elle possède une clé étrangère de type MEDIUMINT se référant respectivement à *order\_number* de la table **OrderPizzeria.**

Tous les attributs de la table **Billing** doivent être renseignés (NOT NULL).

### La table « Stock »

Cette **table** regroupe les informations des stocks de votre entreprise.

La **table** dispose de deux clés primaires étrangères [PFK], et sont de type TINYINT :

* Ingredient\_id, se référant à l’*iD* de la table **Ingredient**.
* Pizza\_store\_id, se référant à l’*iD* de la table **PizzaStore**.

Outre ces *attributs*, le type est :

* + SMALLINT pour *quantity,* la quantité. L’unité de mesure sera par défaut le kilogramme.

Tous les attributs de la table **Stock** doivent être renseignés (NOT NULL), sauf *quantity* car il se peut qu’un ingrédient soit en rupture de stock.

### La table « Ingredient »

Cette **table** regroupe les informations des ingrédients nécessaires à la fabrication de vos pizzas.

La **table** dispose d’une clé primaire *iD*, avec AUTO\_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacun des ingrédients.

Pour les autres *attributs*, le type est :

* + VARCHAR, limité à 150 caractères pour *name*.

L’attribut *supplier\_id* est une clé étrangère se référant à l’*iD* de la table **Supplier**. Elle est de type TINYINT.

Tous les attributs de la table **Ingredient** doivent être renseignés (NOT NULL).

### La table « Recipe »

Cette **table** regroupe les informations de toutes les recettes de pizzas.

La **table** dispose d’une clé primaire *iD*, avec un AUTO\_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacune des recettes.

Pour les autres *attributs*, le type est :

* + MEDIUMTEXT pour *comment*.

Les attributs *ingredient\_id et pizza\_id* sont des clés étrangères se référant respectivement à l’*iD* de la table **Ingredient et** à l’*iD* de la table **Pizza**.

Tous les attributs de la table **Recipe** doivent être renseignés (NOT NULL), sauf *comment*, car il se peut que vous estimiez qu’une pizza n’ait pas de recette.

### La table « Pizza »

La table **Pizza** regroupe les informations des pizzas.

La **table** dispose d’une clé primaire *iD*, avec AUTO\_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacune de vos pizzas. Nous pensons que vous ne commercialisez pas plus de 255 pizzas, il est donc inutile de lui affecter un type MEDIUMINT ou SMALLINT.

Pour les autres *attributs*, le type est :

* + VARCHAR, limité à 150 caractères pour *name* et *crust*.
  + DECIMAL, limité à 4 chiffres pour la partie entière et 2 chiffres pour la partie décimale, pour *cost* et *price*.
  + DECIMAL, limité à 1 chiffre pour la partie entière et 1 chiffre pour la partie décimale, pour *size*. Nous vous proposons de travailler avec des échelles. Ainsi la valeur 1 correspondra à une pizza de taille standard, 1.5 à une pizza medium, 2 à une pizza familiale etc...

L’attribut *team\_id* est une clé étrangère se référant à l’*iD* de la table **Team**. Elle est de type TINYINT.

Tous les attributs de la table **Pizza** doivent être renseignés (NOT NULL), sauf *cost*, et *price*.

### La table « Supplier »

Cette **table** regroupe les informations concernant vos fournisseurs.

La **table** dispose d’une clé primaire *iD*, avec un AUTO\_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacun de vos fournisseurs.

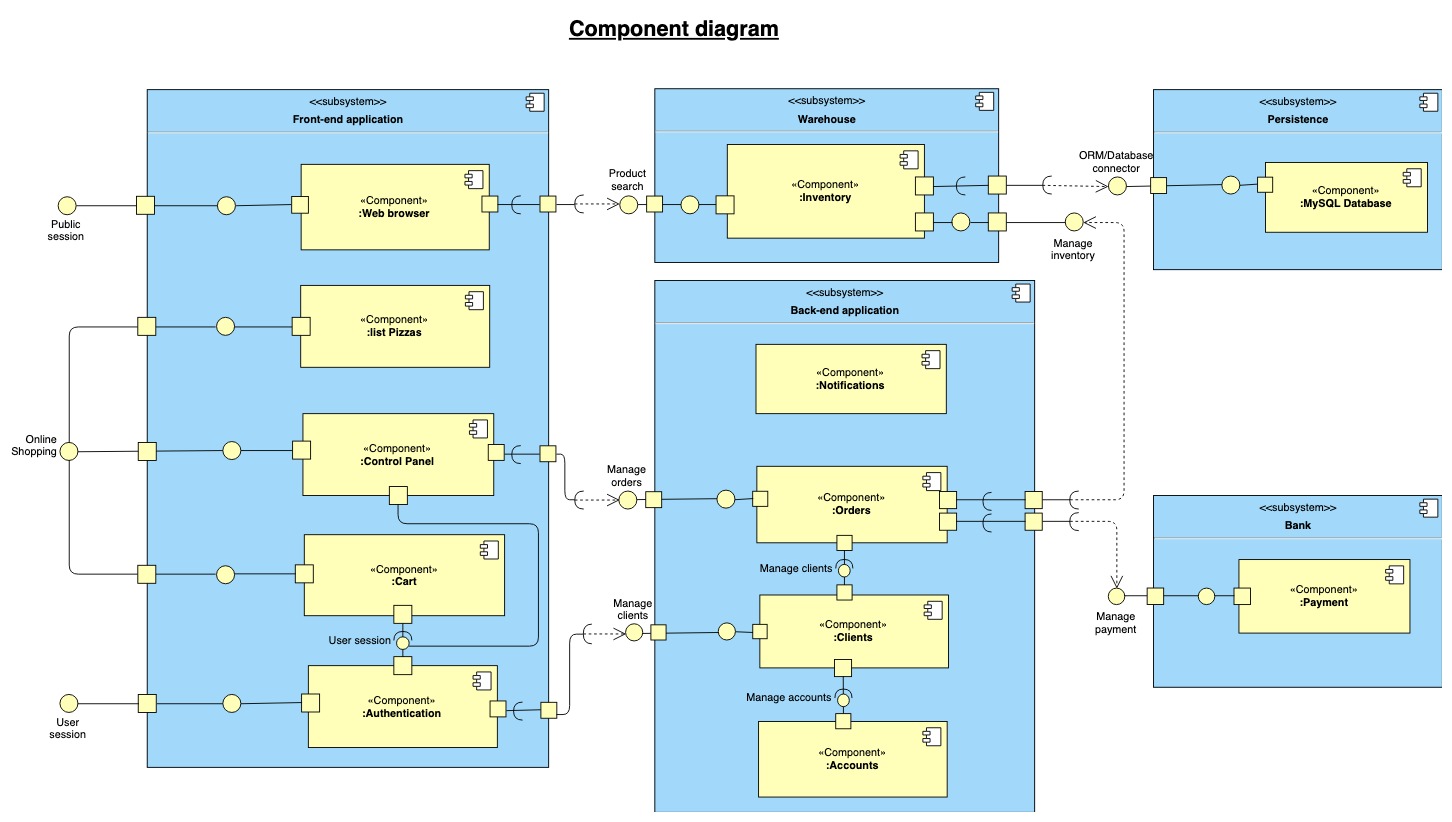
La **table** possède un autre *attribut*, le type est :

* + VARCHAR, limité à 150 caractères pour *name*.

Il doit être renseigné (NOT NULL).

# Les différents composants du système et les composants externes utilisés par le domaine fonctionnel et leur interaction

## Le diagramme de composants



## Descriptif des composants

Dans cette partie nous décrirons les composants internes et externes utilisés par l’application.

### Le navigateur web

Le navigateur web **(Web browser)** est un logiciel ou interface graphique permettant de naviguer sur le World Wide Web.  
Les navigateurs web classiques tels que Firefox 65.0, Google Chrome 72.0.3626.81, Internet explorer 11.0.101, Safari 12.0 ou Ecosia 4.0.4 seront ainsi utilisables pour accéder à votre application (site web).

### Le catalogue

Le composant **List pizzas** va permettre à votre client de consulter le catalogue des pizzas que vous commercialisez. C’est à partir de ce composant qu’il fera son choix parmi la gamme de produits.

### Le tableau de bord

### Le panier

Il s’agit [**Cart**] du dispositif de votre site marchand en ligne dans lequel se trouvent tous les articles sélectionnés durant la visite par vos clients. Il calcule également le total à régler. Le panier se transforme en commande au moment de la validation.

### L’authentification

Le composant **Authentication** permet l’inscription de nouveaux clients à a votre application ainsi que les connexions de ceux-ci et de l’équipe. Pour le client, cette étape est nécessaire pour pouvoir passer une commande. Elle est également obligatoire pour l’équipe, pour avoir accès à la gestion des commandes notamment.

### Les notifications

Le système de notification permettra par exemple de transmettre des informations au client sur sa commande. Il pourra s’agir d’alertes push ou de SMS reçus. Concernant l’équipe, il s’agira d’alertes push envoyés sur le tableau des commandes pour avertir l’équipe d’une nouvelle commande à gérer etc.

### Les commandes

La gestion des commandes **(Orders)**

### Les clients

### La gestion des clients **(Clients)**

### Les comptes

Le composant **Accounts**

### L’inventaire

Le composant **Inventory** permet à votre équipe et à vous-même de pouvoir contrôler les matières premières reçues par vos fournisseurs, les marchandises consommées par la cuisine de vos différents points de vente ainsi que celles actuellement en stock.

### Le paiement

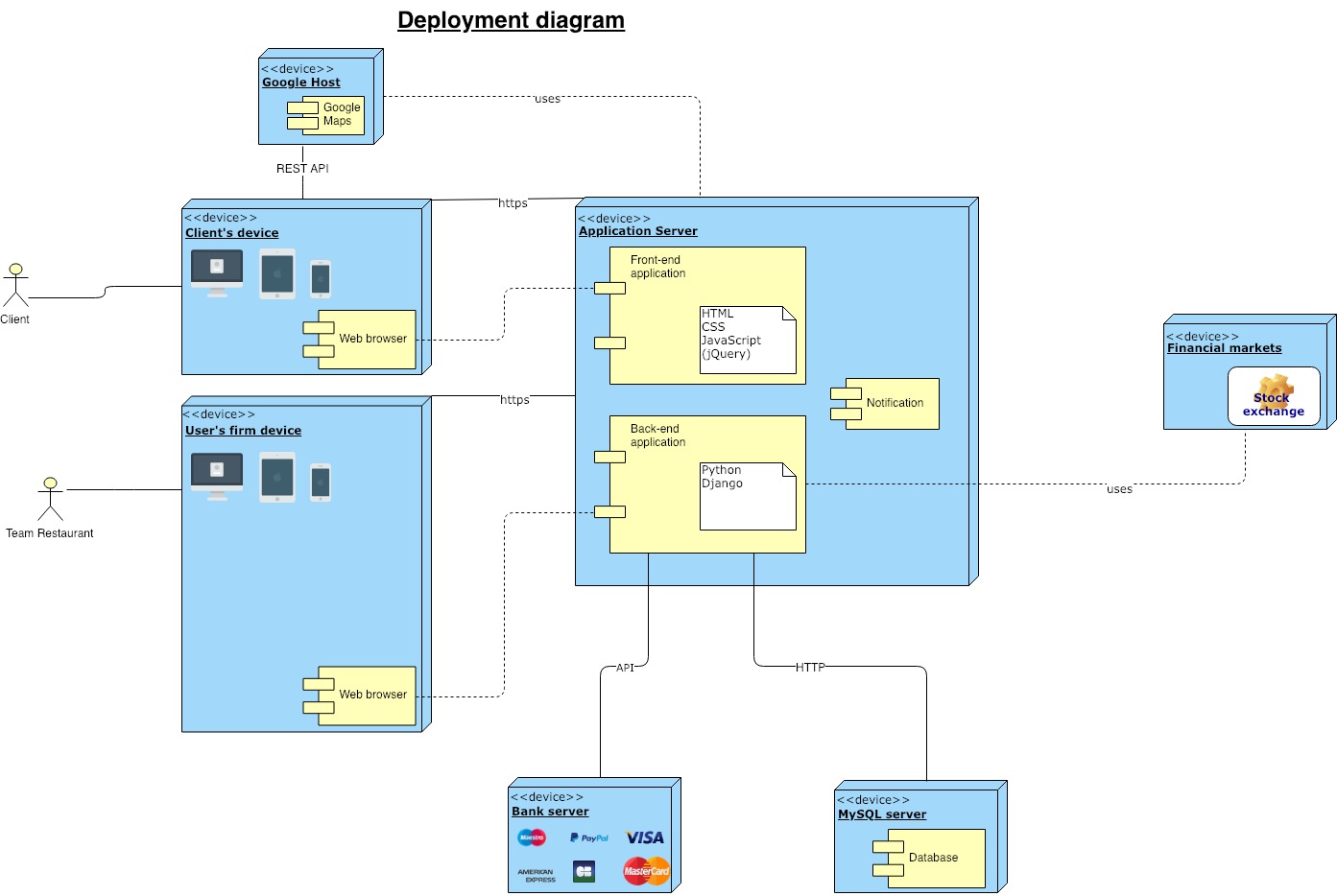
Le composant **Payment** permet au client de régler le montant de sa commande. Il gèrera aussi le transfert de cet argent sur le compte bancaire de l’entreprise.

### La base de données MySQL

# L’architecture de déploiement

## Le diagramme de déploiement

Un diagramme de déploiement décrit le déploiement physique environnemental du système ainsi que des composants matériels.



## Environnements utilisateurs

Protocole

Pourquoi celui la utilise

Le chemin de communication **TCP/IP** permet à la couche d’application de communiquer avec le nœud serveur de base de données (**Database Server**) qui est situé sur un autre matériel, externe.  
Ce serveur héberge la Base de données (*Database*) qui tourne sous **MySQL 5.7.20**, version actuelle.