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 4](#_Toc2248987)

[a. Rappel du contexte 4](#_Toc2248988)

[b. Rappel des besoins attendus 4](#_Toc2248989)

[II. Solution fonctionnelle 5](#_Toc2248990)

[III. Domaine fonctionnel 5](#_Toc2248991)

[a. Le diagramme de classes 5](#_Toc2248992)

[b. Les objets 6](#_Toc2248993)

[La classe « PizzaStore » 6](#_Toc2248994)

[La classe « Address » 7](#_Toc2248995)

[La classe « Team » 8](#_Toc2248996)

[La classe « Client » 9](#_Toc2248997)

[La classe « OrderPizzeria » 10](#_Toc2248998)

[La classe « Billing » 11](#_Toc2248999)

[La classe « Ingredient » 11](#_Toc2249000)

[La classe « Pizza » 12](#_Toc2249001)

[La classe « Recipe » 13](#_Toc2249002)

[La classe « Supplier » 13](#_Toc2249003)

[La classe d’association « ClientDetails » 13](#_Toc2249004)

[La classe d’association « TeamDetails » 13](#_Toc2249005)

[La classe d’association « Stock » 14](#_Toc2249006)

[La classe d’association « OrderSupplier » 14](#_Toc2249007)

[La classe d’association « CommandLine » 14](#_Toc2249008)

[La classe d’association « Composition » 15](#_Toc2249009)

[IV. Le modèle physique de données (MPD) 16](#_Toc2249010)

[a. Le MPD d’OC Pizza 16](#_Toc2249011)

[b. Les tables 16](#_Toc2249012)

[La table « PizzaStore » 16](#_Toc2249013)

[La table « Address » 17](#_Toc2249014)

[La table « Team » 17](#_Toc2249015)

[La table « Client » 18](#_Toc2249016)

[La table « OrderPizzeria » 18](#_Toc2249017)

[La table « Billing » 19](#_Toc2249018)

[La table « Ingredient » 20](#_Toc2249019)

[La table « Pizza » 20](#_Toc2249020)

[La table « Recipe » 21](#_Toc2249021)

[La table « Supplier » 21](#_Toc2249022)

[La table d’association « ClientDetails » 21](#_Toc2249023)

[La table d’association « TeamDetails » 21](#_Toc2249024)

[La table d’association « Stock » 22](#_Toc2249025)

[La table d’association « OrderSupplier » 22](#_Toc2249026)

[La table d’association « CommandLine » 22](#_Toc2249027)

[La table d’association « Composition » 23](#_Toc2249028)

[V. Les différents composants du système et les composants externes utilisés par le domaine fonctionnel et leur interaction 23](#_Toc2249029)

[a. Le diagramme de composants 23](#_Toc2249030)

[b. Descriptif des composants 24](#_Toc2249031)

[Le navigateur web 24](#_Toc2249032)

[Le catalogue 24](#_Toc2249033)

[Le tableau de bord 24](#_Toc2249034)

[Le panier 25](#_Toc2249035)

[L’authentification 25](#_Toc2249036)

[Les notifications 25](#_Toc2249037)

[Les commandes 25](#_Toc2249038)

[Les clients 25](#_Toc2249039)

[Les comptes 25](#_Toc2249040)

[L’inventaire 25](#_Toc2249041)

[Le paiement 26](#_Toc2249042)

[La base de données MySQL 26](#_Toc2249043)

[VI. L’architecture de déploiement 27](#_Toc2249044)

[a. Le diagramme de déploiement 27](#_Toc2249045)

[b. Environnements utilisateurs 27](#_Toc2249046)

[Les matériels des utilisateurs 27](#_Toc2249047)

[Le serveur d’application 28](#_Toc2249048)

[Le serveur de base de données 28](#_Toc2249049)

[Le serveur bancaire 28](#_Toc2249050)

[Les marchés financiers 28](#_Toc2249051)

[Google Host 29](#_Toc2249052)

# 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

Le diagramme de classe est une représentation des éléments qui composent la base de données et ses relations internes.

**Une image contenant carte, texte

Description générée automatiquement**

## 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 d’entre eux. 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 l’horaire d’ouverture de la pizzeria.
* **closingHour**est l’attribut décrivant l’horaire 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 ou membre de cette équipe travaille (works) dans un, et un seul point de vente à la fois. Chaque équipe est affectée 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 à la fois.

### La classe « Address »



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

* *roadCategory* désigne le type de route (rue, avenue, boulevard, impasse etc...).
* *streetNumber*est le numéro associé au catégorie de route ci-dessus.
* *complement* correspond au complément d’adresse ou une instruction particulière concernant la livraison. Il est facultatif.
* *zipCode*est le code postal.
* *city*est 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 **Client** (Client), en effet un client peut avoir (has) une ou plusieurs adresses (domicile, lieu de travail, site universitaire etc..) et chaque adresse ne permet pas d’identifier un client particulier. En effet, dans le cas de membres d’un même foyer, ceux-ci fourniront la même adresse. Il en est de même pour des collègues d’une même entreprise désirant se faire livrer sur leur lieu de travail. La cardinalité est 1..\*, 1..\*. Il est donc nécessaire de créer une classe d’association. Celle-ci s’appelle **ClientDetails.**

Il en est de même pour son association avec la classe **Team** (Équipe), car un membre de l’équipe peut avoir (has) plusieurs adresses et une adresse ne permet pas forcement d’identifier un membre de l’équipe. En effet, il est possible que des collègues du même point de vente soient colocataires, et donc qu’ils aient la même adresse. La cardinalité est 1..\*, 1..\*. Il est donc nécessaire de créer une classe d’association. Celle-ci s’appelle **TeamDetails.**

Enfin, **Address** 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 (donc plusieurs adresses). En revanche, l’adresse de celui-ci permet de l’identifier. Il ne partage pas son adresse avec un autre fournisseur (cardinalité 1..\*, 1).

### La classe « Team »

**Une image contenant capture d’écran

Description générée automatiquement**

La classe **Team** (Équipe) 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.

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

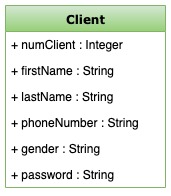
* *employee* (employée).
* *delivery guy*(livreur-se).
* *pizza chef*(pizzaiolo-a).
* *manager*(manager).
* *boss* (patron-ne).

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

**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 par une seule équipe (cardinalité 0..\*, 1).

Enfin, la classe a une association avec la classe **Address** (Adresse), détaillée précédemment.

### La classe « Client »



La classe **Client** (Client) possède les attributs suivants :

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

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

Son association avec la classe **Address** (Adresse) a été explicitée ci-dessus.

### La classe « OrderPizzeria »

Une image contenant capture d’écran, texte

Description générée automatiquement

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

Elle est également associée à la classe **Pizza** (Pizza), association many-to-many de cardinalité (1..\*, 1..\*). Chaque commande est constituée (constitutes) d’une ou plusieurs pizzas et chaque pizza se retrouve dans une ou plusieurs commandes. Il est donc nécessaire de créer une classe d’association. Celle-ci s’appellera **CommandLine**.

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 « 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.

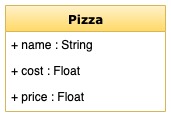
La classe a trois associations.

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 est composée (makes up) de minimum trois ou plusieurs ingrédients, et un ingrédient peut être présent dans une ou plusieurs recettes (cardinalité 3..\*, 1..\*). Il est donc nécessaire de créer une classe d’association (en l’occurrence ici **Composition)**.

Sa troisième association se fait avec la classe **Supplier** (Fournisseur). Vos fournisseurs vous approvisionnent (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..\*). Il est donc nécessaire de créer une classe d’association (en l’occurrence ici **OrderSupplier)**.

### La classe « Pizza »



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

* *name*, le nom qui lui est attribué, le nom sous lequel elle sera commercialisée.
* *cost,* un cout variable.
* *price*, un prix variable.

Elle a une association avec la classe **OrderPizzeria**, qui a été décrite ci-dessus.

Elle est également associée à la classe **Recipe** (recette). Cette association est de type one-to-one, en effet, chaque pizza possède (has) une recette qui est lui est unique. Par conséquent, on peut identifier la pizza par la recette, et inversement (cardinalité 1,1).

### La classe « Recipe »



La classe **Recipe** (recette) est une classe qui possède l’attribut *comment* (instructions concernant la réalisation de la pizza). Elle contient les instructions destinées au pizzaiolo-a.

La classe a des associations avec les classes **Pizza** et **Ingredient** qui ont été brièvement décrites précédemment.

### 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.

### La classe d’association « ClientDetails »



**ClientDetails** est une classe d’association qui ajoute l’attribut *membershipStartDate* (date d’adhésion du client)à l’association many-to-many des classes **Client** (Client) et **Address** (Adresse). L’association a été précédemment détaillée.

### La classe d’association « TeamDetails »



**ClientDetails** est une classe d’association qui ajoute l’attribut *membershipStartDate* (date d’adhésion du salarié)à l’association many-to-many des classes **Team** (Équipe) et **Address** (Adresse). L’association a été précédemment détaillée.

### La classe d’association « 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 d’association « OrderSupplier »

Une image contenant capture d’écran

Description générée automatiquement

La classe **OrderSupplier** (CommandeFournisseur) est une classe d’association qui ajoute l’attribut *date* (date de la livraison de la matière première) et *quantity* (quantite de matière première commandée) à l’association many-to-many des classes **Ingredient** (Ingrédient) et **Supplier** (Fournisseur). L’association a été précédemment détaillée.

### La classe d’association « CommandLine »

Une image contenant capture d’écran

Description générée automatiquement

**CommandLine** (ligne de commande) est une classe d’association qui ajoute les attributs *quantity* (quantité), *pizzaSize* (taille de la pizza choisie), *pizzaCrust* (type de croute) choisi à l’association many-to-many des classes **OrderPizzeria** (Commande) et **Pizza** (Pizza). L’association a été précédemment détaillée.

La table **enumeration** définit les seules valeurs possibles de l’attribut *pizzaSize*, soient :

* *1*
* *1.5*
* *2*

Nous justifierons ce choix dans la table éponyme à cette classe dans le MPD.

La table **enumeration** définit également les seules valeurs possibles de l’attribut *pizzaCrust*, soient :

* thin (croute fine)
* thick (croute épaisse)
* crusty (croute croustillante)

### La classe d’association « Composition »



**Composition** (composition) est une classe d’association qui nait de la relation many-to-many des classes **Ingredient** et **Recipe**. Elle ajoute l’attribut *quantity* (quantité nécessaire à la réalisation de la recette).

L’association entre les deux classes a été précédemment détaillée.

# Le modèle physique de données (MPD)

## Le MPD d’OC Pizza

À partir du diagramme de classes, nous pouvons élaborer votre MPD.

Le modèle physique des données permet d'avoir une représentation graphique de la structure d’une base de données et de mieux comprendre les relations entre les différentes tables.

Une image contenant capture d’écran

Description générée automatiquement

## 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 possède 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 127 salarié-es.
  + TIME, pour *opening\_hour* et *closing\_hour*, car ici nous n’avons pas besoin d’un type DATETIME, l’heure suffit.

Elle possède également une clé étrangère de type MEDIUMINT, *address*\_*id* se référant à l’*iD* de la table **Address**.

Tous les attributs de la table **PizzaStore** doivent être renseignés (NOT NULL), sauf *revenue*, et *team\_size*, des informations qui peuvent être considérées comme confidentielles pour être omises.

### 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 MEDIUMINT, 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*.

Elle possède la clé étrangère *supplier\_id*, se référant à l’*iD* de la table **Supplier**. On crée une clé étrangère dans la table correspondant à la cardinalité maximum.

*supplier\_id* est de type TINYINT.

Tous les attributs de la table **Address** doivent être renseignés (NOT NULL) sauf *complement* et *supplier\_id*.

### 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 100 caractères pour *first\_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.

L'option de mettre la clé étrangère *pizza\_store\_id* de type TINYINT et se référant à l’*iD* de la table **PizzaStore** dans la table **Team** se justifie par le fait qu’on puisse considérer ici plus « naturel » de placer les points de vente autour de l’équipe que de faire rentrer un point de vente dans un membre de l’équipe.

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.

Outre ce MEDIUMINT pour la clé primaire, elle dispose des types suivants :

* + VARCHAR, limité à 100 caractères pour *first\_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.

Ces attributs doivent tous être renseignés (NOT-NULL).

### La table « OrderPizzeria »

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

La **table** dispose d’une clé primaire *order\_number*, et elle est de type MEDIUMINT, un identifiant unique avec AUTO INCREMENT 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 vérifier 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 au *num\_client* 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’un seul *attribut*, dont le type est :

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

Elle possède donc une clé primaire étrangère *order\_id* de type MEDIUMINT se référant à *order\_number* de la table **OrderPizzeria.**

Cette table nait de la fusion des tables **OrderPizzeria**, **CommandLine** et **Pizza**. Pour calculer le montant total on doit récupérer l’*order\_id*. A partir de l’order\_id, on peut récupérer la quantité désirée ainsi que la *pizza\_id*. Et à partir de la *pizza\_id*, on peut récupérer le prix de vente de la pizza concernée. Mais on ne peut pas créer immédiatement la table dans **create\_bdd.sql** car on ignore à cet instant la valeur du montant. Elle sera créée dans le fichier **fill\_bdd.sql**.

La requête ci-dessous crée la table **Billing** et récupère les informations nécessaires :

CREATE TABLE Billing AS SELECT order\_id, (quantity\*price) AS amount

FROM CommandLine inner join Pizza ON CommandLine.pizza\_id = Pizza.iD;

Puis on crée la contrainte :

ALTER TABLE Billing ADD CONSTRAINT order\_pizzeria\_billing\_fk

FOREIGN KEY (order\_id)

REFERENCES OrderPizzeria (order\_number)

ON DELETE NO ACTION

ON UPDATE NO ACTION;

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

### 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*.

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

### 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 127 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*.
  + DECIMAL, limité à 4 chiffres pour la partie entière et 2 chiffres pour la partie décimale, pour *cost* et *price*.

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

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

### La table « Recipe »

Cette **table** dispose d’un *iD*, identifiant unique avec AUTO INCREMENT et elle regroupe les informations de toutes les recettes de pizzas.

La **table** dispose d’un *attribut* dont le type est :

* + MEDIUMTEXT pour *comment*.

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

### 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* (nom).

Il doit être renseigné (NOT NULL).

### La table d’association « ClientDetails »

Cette **table** d’association regroupe deux clés primaires étrangères [PFK] *client\_id* se référant à *num\_client* de la table **Client**, et *address\_id* se référant à l’*iD* de la table **Address** qui sont toutes les deux de type MEDIUMINT.

Outre ces *attributs*, le type est :

* + DATETIME, pour *membership\_start\_date*, la date d’adhésion.

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

### La table d’association « TeamDetails »

**TeamDetails** est une tabled’association regroupant deux clés primaires étrangères [PFK] *team\_id* se référant à l’*iD* de la table **Team**, et *address\_id* se référant à l’*iD* de la table **Address**. Elles sont de type MEDIUMINT.

Outre ces *attributs*, le type est :

* + DATETIME, pour *membership\_start\_date, la* date d’adhésion.

Tous les attributs de cette table d’association doivent être renseignés (NOT NULL).

### La table d’association « Stock »

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

La **table** dispose de deux clés primaires étrangères [PFK], qui 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 :

* + DECIMAL(10, 2) 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 d’association « OrderSupplier »

Cette **table** regroupe les informations concernant la livraison de matières premières par vos fournisseurs.

La **table** dispose de deux *attributs* dont le type est :

* + DATETIME pour *date*.
  + DECIMAL(10,2) pour *quantity*.

Les attributs *ingredient\_id et supplier\_id* sont des clés primaires étrangères se référant respectivement à l’*iD* de la table **Ingredient et** à l’*iD* de la table **Supplier**. Ces attributs sont de type TINYINT.

### La table d’association « CommandLine »

Cette **table** regroupe deux clés primaires étrangères [PFK] *order\_id* se référant à *order\_number* de la table **OrderPizza**, et *pizza\_id* se référant à l’*iD* de la table **Pizza** qui sont respectivement de type MEDIUMINT et TINYINT.

Outre ces *attributs*, le type est :

* + ENUM, pour *pizza\_size* et *pizza\_crust*, qui peuvent prendre les valeurs énoncées plus haut. Pour *pizza\_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...
  + TINYINT pour *quantity*.

### La table d’association « Composition »

Cette table d’association stocke deux clés primaires étrangères [PFK] *recipe\_id* se référant à *iD* de la table **Recipe**, et *ingredient\_id* se référant à l’*iD* de la table **Ingredient** qui sont de type TINYINT.

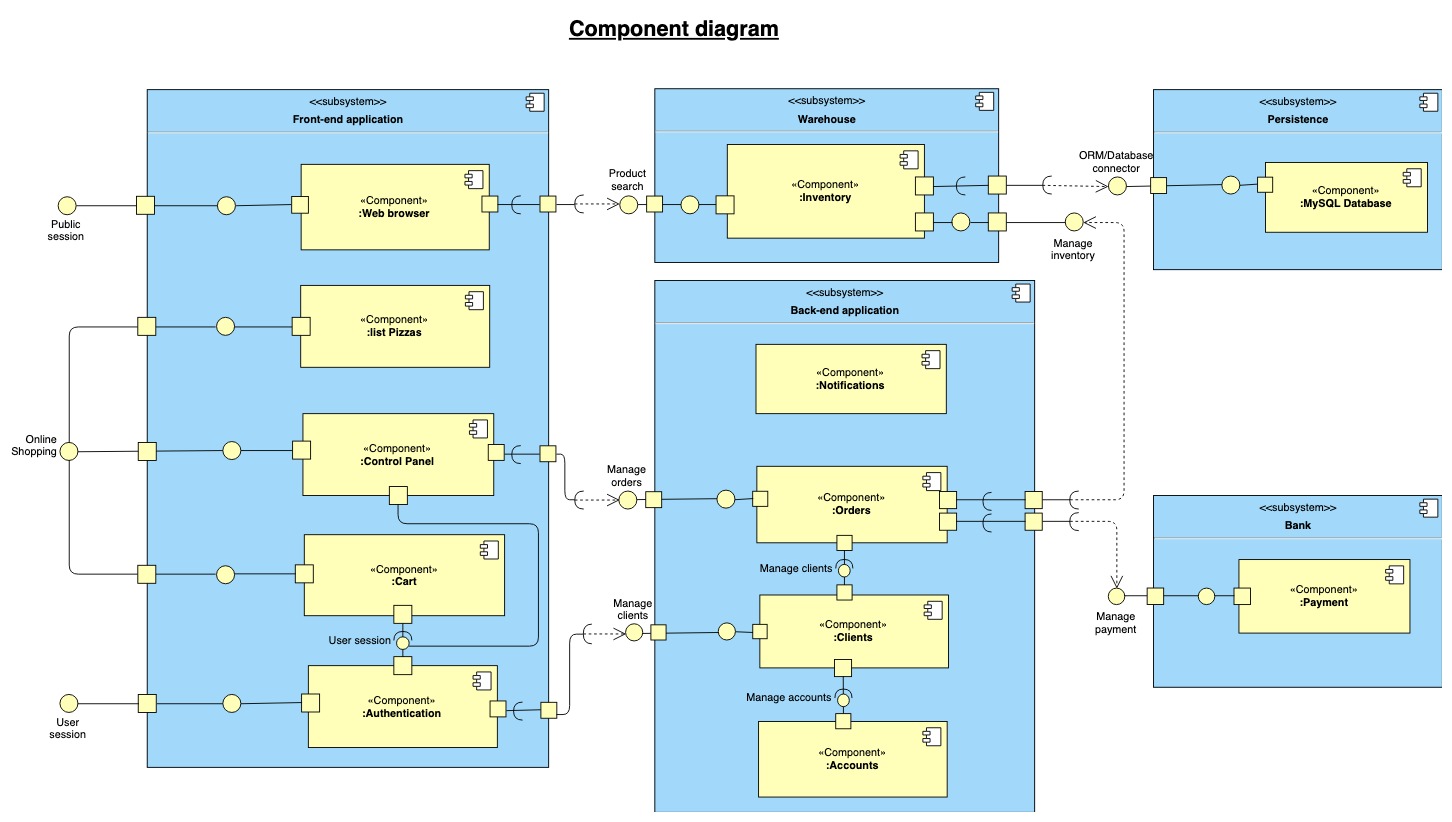
Elle possède également l’attribut *quantity*, de type DECIMAL (10, 2).

Tous les attributs doivent être renseignés (NOT NULL).

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

## Le diagramme de composants

Un diagramme de composants représente la structure du système logiciel, en décrivant les composants du logiciel, ses interfaces entre composants internes externes ainsi que ses dépendances.



## Descriptif des composants

Nous avons identifié, pour votre application, 12 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

L’intérêt du composant **Control Panel** est d 'avoir accès pour tout l’équipe a toutes les informations en instantané. Il permet également de suivre la performance des différents points de ventes pour le patron et d’être alerté en cas d’anomalies.

### 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 **(Notifications)** 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)** est utile aux employés pour assurer le bon fonctionnement des différentes étapes, de la validation de la commande à sa livraison.

### Les clients

La gestion des clients **(Clients)** est le plus utile au pizzaiolo qui doit suivre les instructions concernant la commande ainsi qu’au livreur qui doit trouver l’adresse de livraison, si le client a choisi de se faire livrer.

### Les comptes

Le composant comptes **(Accounts)** gère les comptes clients et équipe. Le manager ainsi que le patron peuvent s’en charger. Par exemple, il se peut que l’équipe s’agrandisse, il sera alors nécessaire d’ajouter ce membre a votre équipe. Dans le cas contraire, lorsqu’un membre démissionne, il sera nécessaire de le supprimer.

### L’inventaire

Le composant **Inventory** (Inventaire) 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** (Paiement)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.

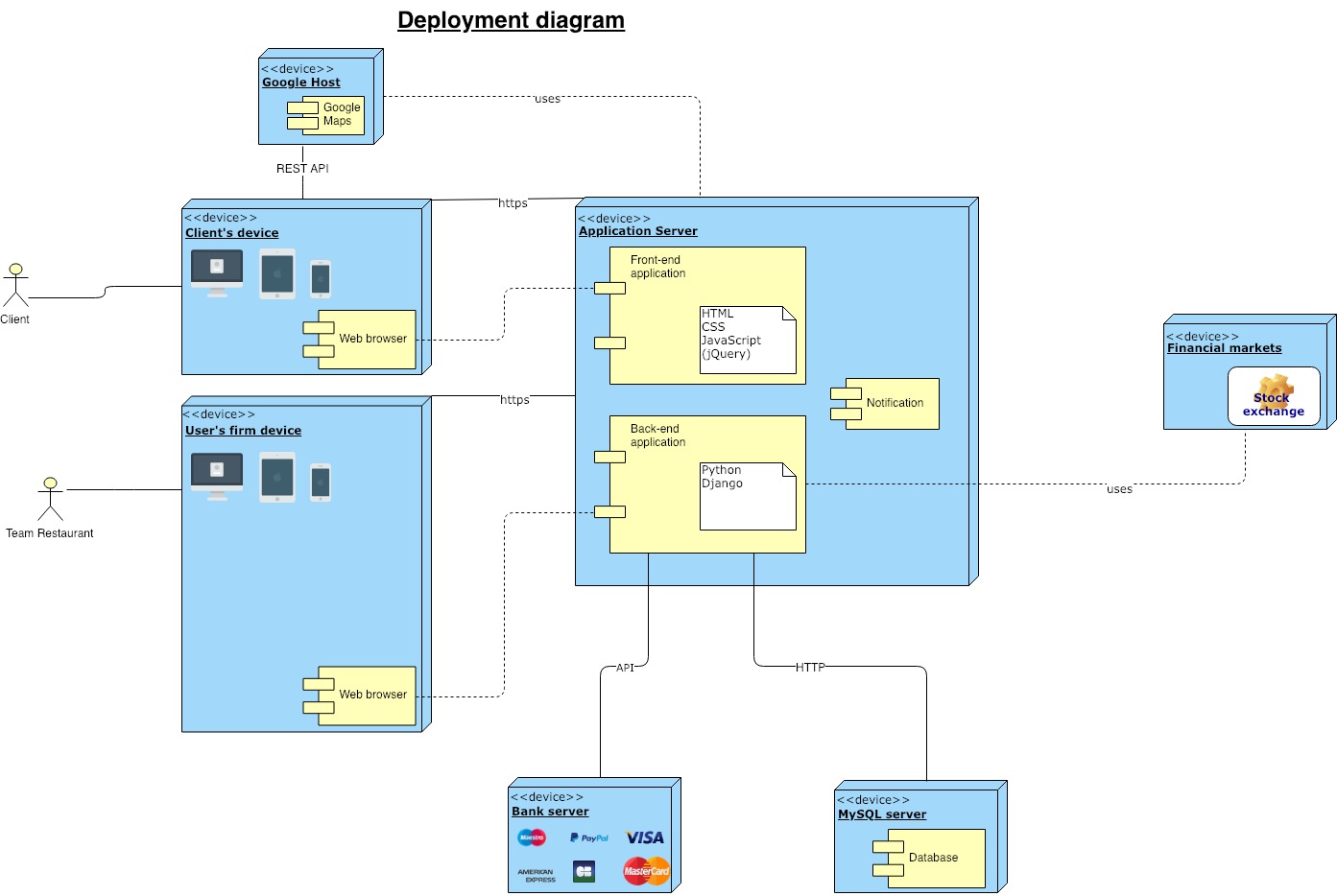
### La base de données MySQL

La base de données MySQL sera déployée sur un serveur tournant sous Debian 9. Elle stocke toutes les tables citées précédemment lors de l’analyse du MPD.

# 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

Nous avons identifié six environnements utilisateurs.

### Les matériels des utilisateurs

Il s’agit d’ordinateurs, tablettes ou smartphones que le client et l’équipe utilisent pour se connecter à votre application (site web).

Le personnel et les clients s’y rendent pour passer des commandes. Les l’utilisent pour passer commande sans intermédiaire. L’équipe peut passer une commande pour le client lorsque celui-ci souhaite récupérer sa commande sur place, ou lorsque celui-ci téléphone.

Les cadres utilisent l’application principalement pour gérer les pizzerias d’un point de vue financier.

Les clients et l’équipe utilisent un navigateur web qui, va lancer des requêtes HTTPS et permettre de se connecter au deuxième nœud, le serveur d’application **(Application Server)**.

### Le serveur d’application

Il s’agit du site web divisé en une partie front-end (partie visible) et une partie back-end (partie logique).

Le front-end utilise comme ressources les langages de programmations HTML, CSS, et JavaScript (librairie jQuery).

Quant au back-end, il sera construit en langage Python, jumelé à son Framework Django.

Des notifications peuvent être envoyées au client ou à l’équipe, résultant de divers évènements liés au processus de commande.

Votre serveur communiquera avec des services externes comme l’hôte Google **(Google Host)**, les marchés financiers **(Financial markets)**, la base de données **(MySQL server)** et le serveur bancaire **(Bank)**. Ces communications sont détaillées dans chacun des services cités.

### Le serveur de base de données

Un protocole HTTP permet à votre nœud serveur d’application de communiquer avec le nœud serveur de base de données qui est considéré comme un service externe.  
Ce serveur héberge votre base de données MySQL 5.2.

### Le serveur bancaire

Un protocole API permet à votre nœud serveur d’application de communiquer avec le nœud serveur bancaire qui est considéré comme un service externe.  
Son rôle est gérer les opérations de paiement.

### Les marchés financiers

Sur les marchés, les échanges concernent également les matières premières. Le rôle des cadres de votre entreprise est de négocier celles-ci.

Le prix de ces matières premières fluctue en fonction de la demande. Le rôle du patron est d’anticiper une éventuelle hausse des prix, et éventuellement trouver un substitut à un ingrédient d’une pizza.

Ce service communique avec votre serveur d’application.

### Google Host

Ici, il est fort probable que vos clients découvrent votre pizzeria via un système de navigation comme Google MAPS, celui-ci lui renvoyant une liste des pizzerias à proximité.

Le protocole REST API permet à votre application serveur de communiquer avec le matériel du client.