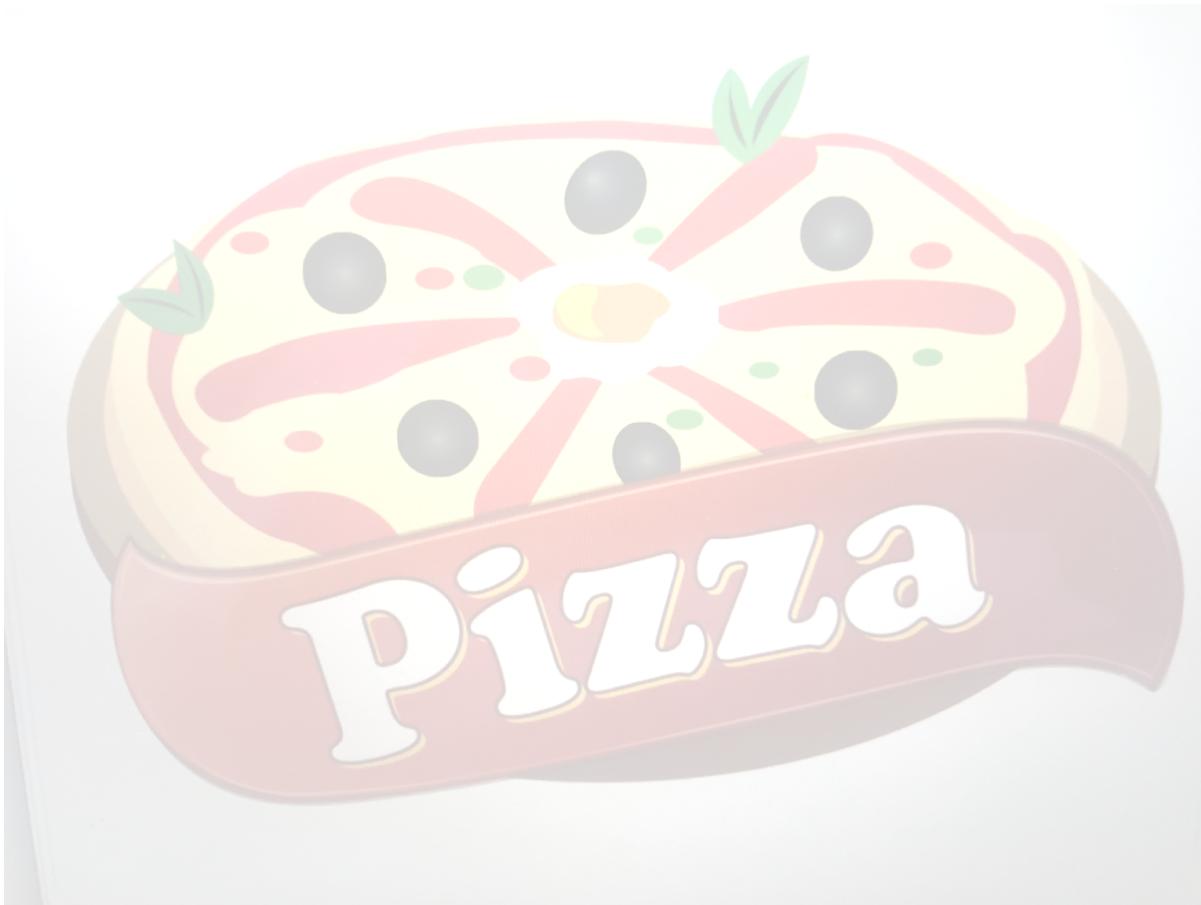


Projet 6

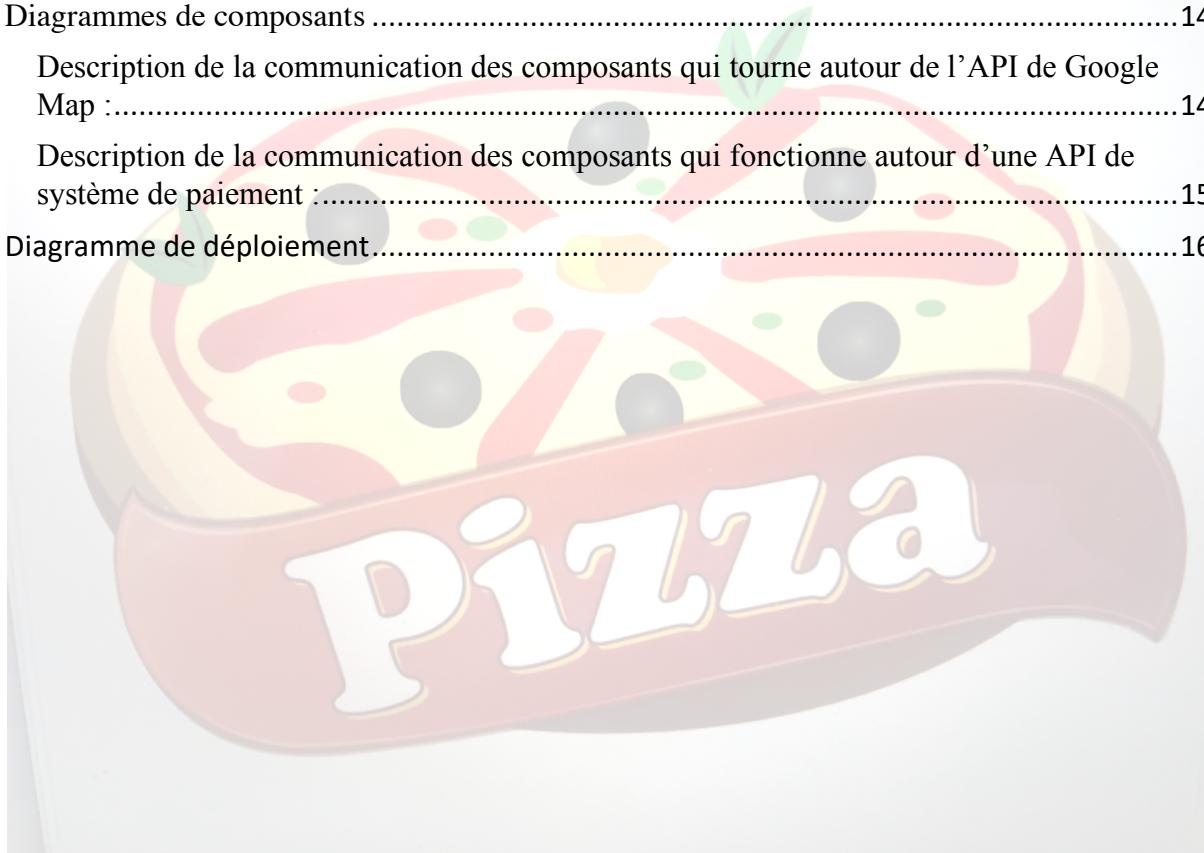
Concevoir la solution Technique d'un système de Gestion de Pizzeria



Auteur : YANN HAMDI

Table des matières

Besoin du client	3
Contexte :	3
Diagramme de classe.....	4
MODELE PHYSIQUE DE DONNEES	7
Exemples De Requête :	13
Deuxième exemple :	13
Troisième exemple :	13
Quatrième exemple :	13
Dernier exemple :	13
Diagrammes de composants	14
Description de la communication des composants qui tourne autour de l'API de Google Map :	14
Description de la communication des composants qui fonctionne autour d'une API de système de paiement :	15
Diagramme de déploiement	16



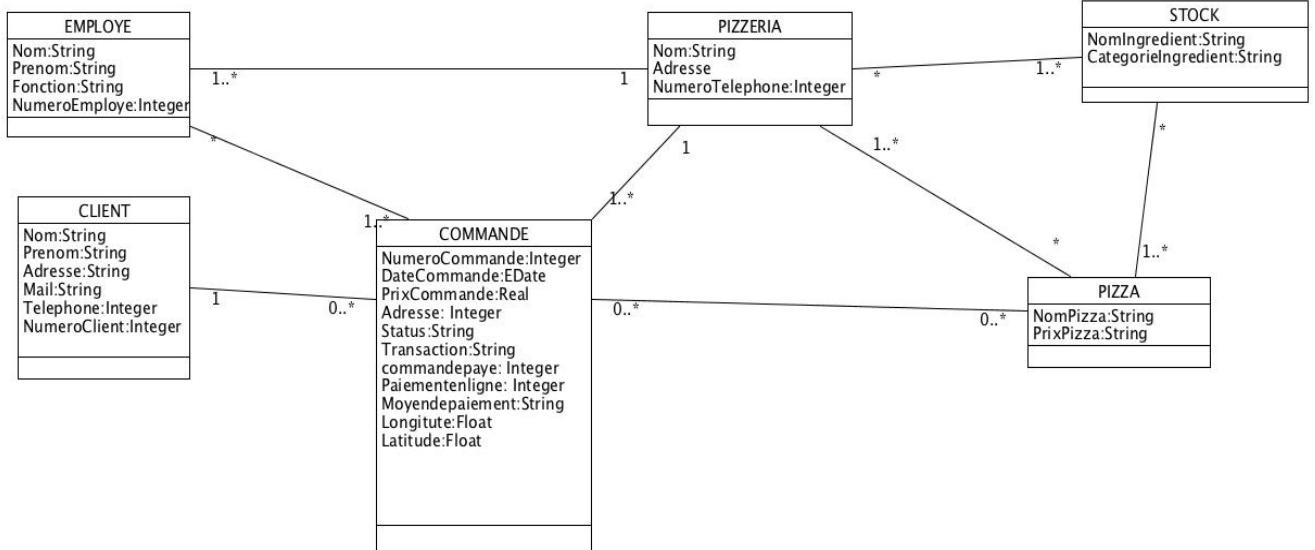
Besoin du client

Contexte :

Votre enseigne « **OC Pizza** » est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et vous prévoyez d'en ouvrir au moins 3 de plus d'ici la fin de l'année. Vous avez pris contact avec nous afin de mettre en place un système informatique, déployé dans toutes vos pizzerias qui 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 à priori déjà fait une prospection et les logiciels existants n'ont pas répondu à vos attentes. Je vais donc essayer de faire au mieux pour répondre à vos besoins de manière efficace. Ce document est donc une proposition de solution technique de votre système de gestion

Diagramme de classe



Nous pouvons donc voir notre diagramme de classe représenté ci-dessus avec 8 différentes classes, je vais donc vous faire une présentation de chaque classe ainsi que leur rôle.

1.Classe "EMPLOYE" :

Nous avons en premier lieu la classe « EMPLOYE » qui représentera tous les employés des pizzerias, cette classe est liée avec la classe pizzeria avec une relation 1..* et 1 puisqu'un employé ne peut travailler qu'à une seule pizzeria par contre une pizzeria peut avoir 1 ou plusieurs employés qui y travaille.

Il y a également une relation entre la classe employé et la classe commande avec une relation *, 1..* puisqu'il y peut y avoir plusieurs employés qui travaillent sur la même commande (livreur et pizzaiolo par exemple) et un employé peut avoir travaillé sur plusieurs commandes.

2.Classe "PIZZERIA" :

Nous avons ensuite la classe « PIZZERIA » qui représente les différentes pizzerias du groupe. Cette classe est reliée à la classe « EMPLOYE »

comme expliqué plus haut puis nous avons d'autre relation.

Elle est ensuite reliée au stock afin d'avoir un suivi du stock des différentes pizzerias, elle est reliée par une relation *, 1..* puisque que le même aliment du stock peut être présent dans plusieurs pizzeria à la fois et une pizzeria a plusieurs aliments du stock.

Ensuite, nous avons une relation entre la classe « **Pizzeria** » et la classe « **Commande** » avec une relation 1, 1..* puisqu'une commande ne peut appartenir qu'à une seule pizzeria et une pizzeria peut avoir plusieurs commandes.

Puis également une relation entre la classe « **PIZZA** » et la classe « **PIZZERIA** » ce qui correspondrait au menu par pizzeria, nous avons une relation 1..*, * puisque qu'une pizza peut appartenir à toutes les pizzerias et qu'une pizzeria peut avoir plusieurs pizzas au menu.

3.Classe "STOCK":

Cette classe correspond au stock disponible par pizzeria, elle joue le rôle de suivi du stock des aliments.

Elle est reliée à la classe « **PIZZERIA** » comme déjà expliqué précédemment puis à la classe « **PIZZA** » puisque qu'une pizza est composé de différents ingrédients disponible dans le stock, nous avons une relation de cardinalité *, 1..*, effectivement un ingrédient du stock peut être présent sur plusieurs pizzas et une pizza peut avoir différents aliments par sorte de pizza.

4.Classe "PIZZA" :

Cette classe correspond aux différentes pizzas disponibles au menu et cela par pizzeria. Elle est liée à la classe « **PIZZA** » et la classe « **STOCK** » (déjà mentionnée dans la classe pizza et la classe stock)

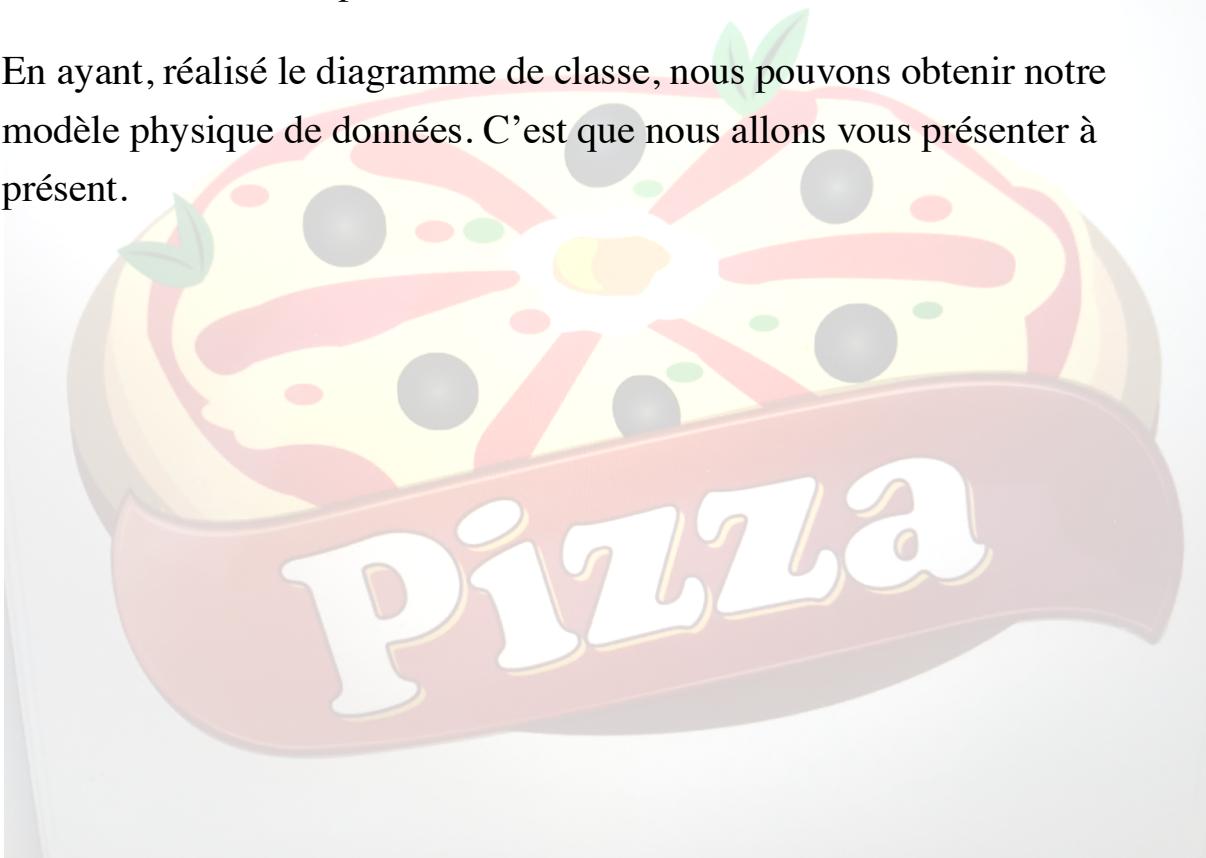
Elle est également liée à la classe « **COMMANDE** » puisqu'une commande va être composée d'aucune ou plusieurs pizzas d'où la relation 0..*, 0..*. Et une pizza peut faire partie d'aucune commande ou bien de plusieurs commandes.

5.Classe "COMMANDÉ":

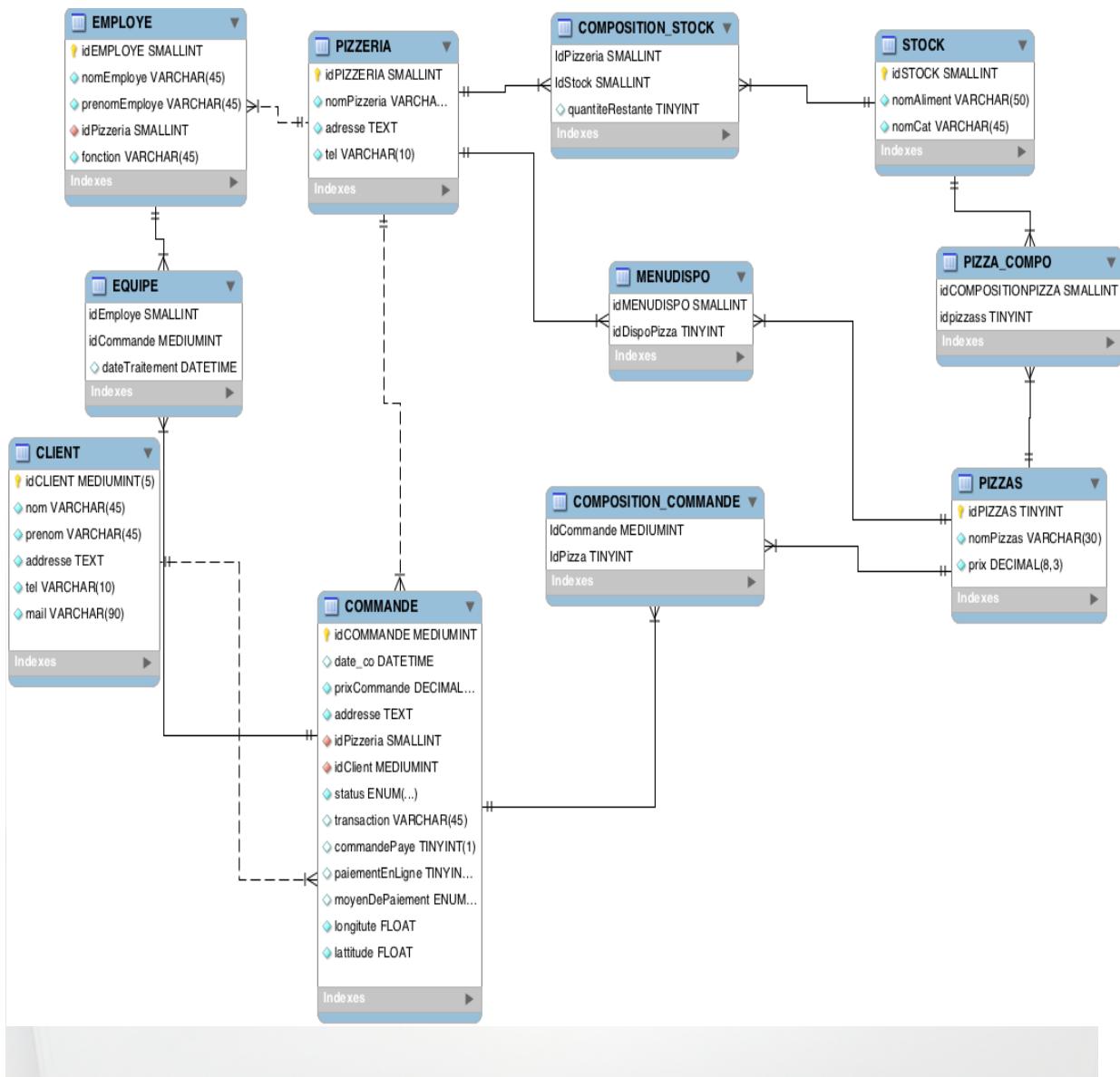
La classe « **COMMANDÉ** » représente le détail de l'ensemble commandes présentes pour chaque pizzeria, la relation entre la classe « **PIZZERIA** » et la classe « **COMMANDÉ** » a déjà été traitée dans la classe « **PIZZERIA** ».

Nous avons une relation entre la classe « **COMMANDÉ** » et la classe « **CLIENT** » et cette relation est $1, 0..*$, évidemment, une commande ne peut appartenir qu'à un seul client à la fois tandis qu'un client peut avoir aucune commande à plusieurs commandes.

En ayant, réalisé le diagramme de classe, nous pouvons obtenir notre modèle physique de données. C'est que nous allons vous présenter à présent.



MODELE PHYSIQUE DE DONNEES



1. Table "EMPLOYEE" :

Cette table va contenir les informations sur les employés avec leur fonction et savoir dans quel pizzeria ils travaillent.

Toutes les colonnes seront « **NOT NULL** » puisqu'il faudra renseigner chaque colonne.

IDEMPLOYEE est un **SMALLINT AUTO-INCREMENT** cette colonne est la clé primaire de la table, cela permettra de reconnaître de manière unique les employés

Les colonnes nom et prénom seront de type **VARCHAR(45)** pour 45 caractère maximum

La colonne fonction pour déterminer le rôle de l'employé sera également de type **VARCHAR(45)**

Puis la colonne **IdPizzeria** qui sera de type **SMALLINT NON NULL UNSIGNED** et qui sera la clé étrangère vers la table « **PIZZERIA** » ce qui permettra d'identifier sur quel pizzeria l'employé travaille.

2. Table " PIZZERIA " :

Nous avons **idPIZZERIA** qui est de type **SMALLINT NON NULL UNSIGNED AUTO-INCREMENT** puisqu'il sera incrémenté automatiquement lors de l'insertion, ne peut être nul et il s'agit également de clé primaire de la table pour permettre d'identifier de manière unique une pizzeria.

De plus, j'ai rajouté un index unique sur la colonne **nomPizzeria** afin d'éviter que l'on rentre plusieurs fois le même nom de pizzeria.

Nous avons la colonne « **ADRESSE** » de type **TEXT** car assez long du fait du nom de la rue, le numéro, le code postal et la ville également **NON NULL** puisque les informations doivent obligatoirement être fournies

3. Table "STOCK" :

Cette table va stocker les ingrédients disponibles par pizzeria, classé par catégorie.

Nous avons donc la colonne **IdSTOCK** de type **SMALLINT UNSIGNED NOTNULL AUTO-INCREMENT**, qui sera la clé primaire de la table afin d'identifier de manière unique.

Puis les colonnes nomAliment et nomCat qui seront de types **VARCHAR NOTNULL** puisque les colonnes ne peuvent rester vides.

4. Table "COMPOSITION STOCK" :

Il s'agit d'une table de composition du fait de la cardinalité de la relation, nous sommes obligés de créer cette table intermédiaire pour représenter la relation des différentes pizzeria avec les différents aliments.

Nous avons ici une clé primaire composite avec la colonne idPizzeria qui est de type identique à sa colonne de référence de la table « **PIZZERIA** » puisqu'il y s'agit également d'une clé étrangère qui identifiera la pizzeria concernée par le stock.

Nous avons également la colonne idStock qui sera également de même type que la colonne qu'elle référence puisqu'encore une fois il s'agit de la clé étrangère relié à l'identifiant de la table « **STOCK** ».

Le couple de colonne va donc nous donner les informations sur la pizzeria concernée avec l'aliment correspondant.

Puis nous avons la colonne quantitéRestante qui sera de type **TINYINT UNSIGNED**, cette colonne n'est pas **NOT NULL** puisqu'elle peut être égal à zéro s'il n'y a plus de stock disponible, cette colonne est décomptée en unité de pizza.

5. Table "PIZZA" :

Cette table va stocker les différentes pizzas disponibles dans chaque

pizzeria.

Nous avons encore la clé primaire sur la colonne **idPIZZAS** de type **TINYINT UNSIGNED NOT NULL AUTO-INCREMENT** qui permettra d'identifier les pizzas de manières uniques.

Nous avons la colonne nomPizzas qui correspond au nom de la pizza de type **VARCHAR(30)**, nous n'avons pas besoin de nombreux caractère pour cette colonne puisque les noms de pizzas ne sont jamais très long généralement.

Puis une colonne prix de type **REAL UNSIGNED NOT NULL** puisque le prix de la pizza peut ne pas être un nombre entier, doit être renseigné et ne peut pas être nul.

6. Table "MENUDISPO" :

Il s'agit d'une table intermédiaire qui représente la relation entre la table « **PIZZERIA** » et « **PIZZA** ».

Cette table est en quelque sorte le menu de pizza disponible par pizzeria.

Nous avons une clé primaire composite, il s'agit du couple de colonne qui caractérisera l'unicité.

Nous avons dans cette table la colonne **idMENUDISPO** qui est de type identique à la colonne **idPIZZERIA** de la table « **PIZZERIA** » puisque cette référence cette dernière avec la clé étrangère.

Puis la colonne idDispoPizza qui est de même type que la colonne **idPIZZAS** puisqu'il s'agit également de la clé étrangère référencée par cette dernière.

7. Table "PIZZA COMPO" :

Cette table servira de recette pour chaque pizza et montrera les ingrédients disponibles pour chaque pizza.

Il s'agit également d'une table intermédiaire pour représenter la relation entre la table « **STOCK** » et la table « **PIZZA** ».

Comme pour les autres tables intermédiaires, on y trouve un clé primaire composite, puisque le couple de colonne fera son unicité. Et les colonnes respectent le type de données par lesquelles elles sont référencés par le biais de la clé étrangère.

8. Table "COMMANDÉ" :

Cette table va stocker toutes les informations de la commande et qui permettra un suivi de toutes les commandes.

Nous avons d'abord la colonne **idCOMMANDÉ** qui sera de type **MEDIUMINT UNSIGNED NOT NULL AUTO-INCREMENT**, puisque la colonne devra être renseigné, sera positive puis s'incrémentera automatiquement. Il s'agit de clé primaire de table, elle permettra d'identifier de manière unique chaque ligne de cette table.

On y trouvera les colonnes de date et heure de prise de commande. Cette colonne sera de type **DATETIME** cela permettra d'avoir l'heure exacte de la commande.

Nous avons une colonne pour le prix total de la commande (à noter que nous n'avons pas respecter la forme normal 3 puisque le prix pourrait se calculer à partir de la table pizza, cela dit si le prix des pizzas change , cela perturberait le montant total de notre commande, du coup il est préférable d'avoir en base de données le montant total de la commande sur notre table commande)

Nous avons également la colonne adresse de type **TEXT NON NULL**, nous précisons cette information malgré l'adresse dans la table client de cette manière, le client peut donner l'adresse où il se trouve et le système pourra lui proposer la pizzeria la plus proche grâce à un système de géolocalisation (l'api Google map dans notre cas).

IdPizzeria qui sera la clé étrangère vers la table pizzeria afin

d'associer la pizzeria rattachée à la commande.

idCLIENT qui sera aussi une clé étrangère vers la table « **CLIENT** » afin d'identifier le client qui a passé la commande.

Nous avons la colonne « statut » qui est de type ENUM car les champs sont connus, la valeur sera soit en cours, soit livrée en attente ou bien annulée.

Nous avons un booléen pour les colonnes commande Paye et paiement En Ligne, avec 1 pour vrai et 0 pour faux.

Puis le moyen de paiement avec une colonne ENUM pour carte ou cash, on peut éventuellement ouvrir d'autres moyens de paiement.

Cette colonne restera vide s'il y a paiement en ligne puisqu'en ligne, manifestement il n'y aura pas de paiement en espèce.

9. Table "COMPOSITION COMMANDE" :

Cette table servira de table intermédiaire entre la table commande et la table pizza, elle permettra de savoir la composition de la commande (quelle pizza le client a t'il commandé ?)

Elle sera de type identique aux autres tables intermédiaires avec une clé primaire composite et 2 clés étrangères sur les tables commande et pizza.

10. Table "EQUIPE"

On retrouve également une table intermédiaire qui va expliciter les employés qui ont traité la commande. Nous avons encore une clé primaire composite et 2 clés étrangères qui relient les tables « **EMPLOYEE** » et « **COMMANDE** ».

Puis une colonne dateTraitement de type **DATETIME** cela permettra de savoir l'heure exacte de traitement par le pizzaiolo et le livreur.

11. Table "CLIENT":

Cette table stockera les informations du client tel que son adresse son téléphone ainsi que son mail.

Exemples De Requête :

Je vais présenter plusieurs exemples de requêtes à partir de SQL qui montreront comment fonctionne notre base de données et les requêtes disponibles pour les utilisateurs.

Premier exemple :

Disons que l'on veuille voir les commandes passées et livrées pour la pizzeria avec id 1 soit dans notre base de données la pizzeria « chez Luigi » nous allons avoir une double jointure (voir requête numéro 1 du fichier request.sql)

Deuxième exemple :

Disons que nous voulons obtenir le détail des commandes en cours de préparation dans la pizzeria « Chez Mario » sachant que nous n'avons pas le numéro id de la pizzeria en question(voir requête numéro 2 du fichier)

Troisième exemple :

L'aide mémo pour le pizzaiolo pour les recettes, disons par exemple que le pizzaiolo recherche la recette de« la pizza orientale »(voir requête numéro 3)

Quatrième exemple :

On souhaite savoir toutes les commandes passées et leur composition du client « Hamdi »(voir requête numéro 4)

Dernier exemple :

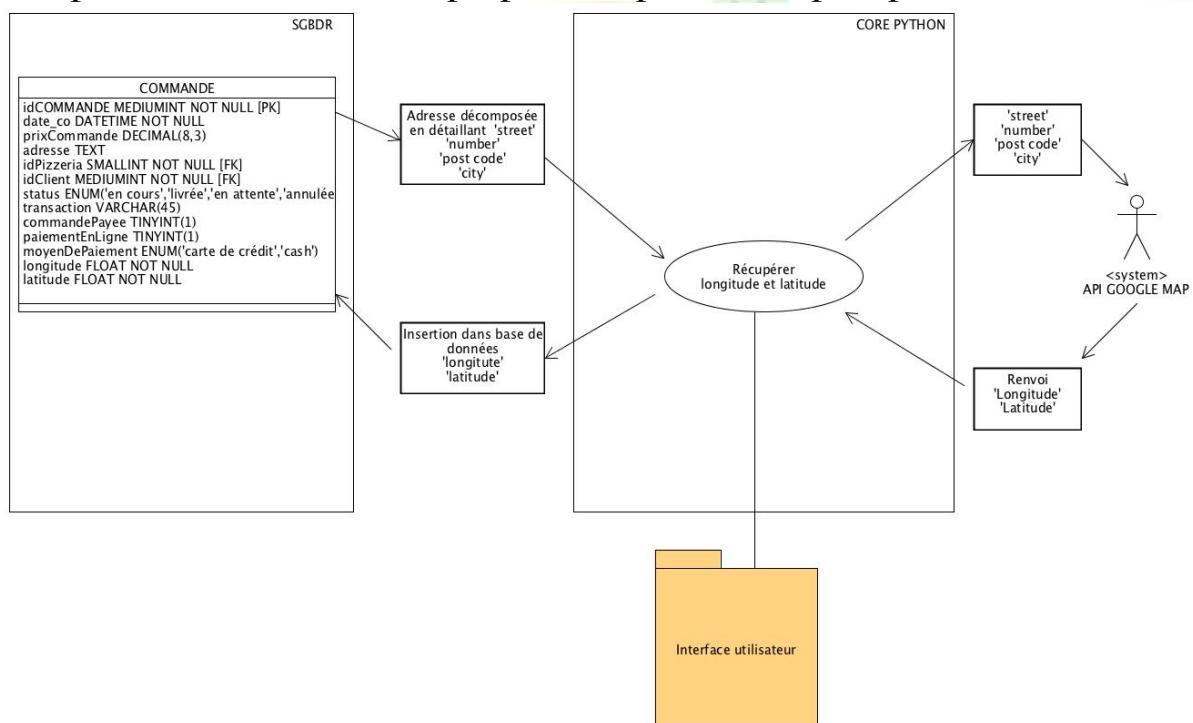
on souhaite connaître les pizzas disponibles selon le stock pour la pizzeria dont l'id=1 (voir requête numéro 5)

Diagrammes de composants

Nous allons à présent nous intéresser aux diagrammes de composants et détailler le fonctionnement de différents acteurs secondaires c'est à dire les acteurs qui communiquent avec notre application mais en font partie seulement de manière indirecte.

Description de la communication des composants qui tourne autour de l'API de Google Map :

En effet, lors de l'enregistrement de sa commande le client va fournir une adresse de livraison ce qui permettra par le biais de l'api de Google Map de comparer les adresses et de proposer la pizzeria la plus proche.



Description de la communication des composants qui fonctionne autour d'une API de système de paiement :

Lorsque le client passe une commande et choisit le paiement en ligne, tout un processus se met en place afin de procéder à la validation du paiement et cela grâce à un API de paiement.

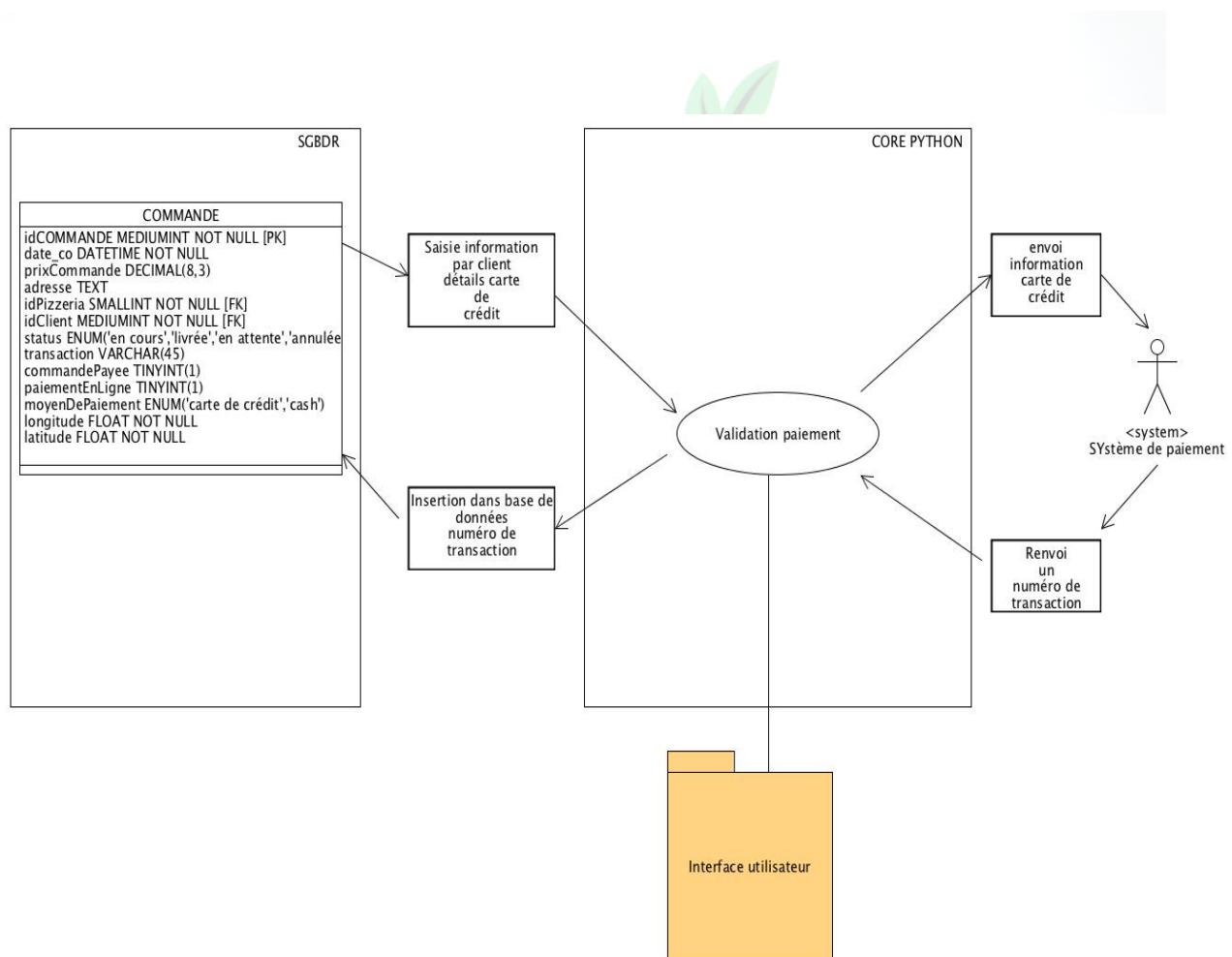
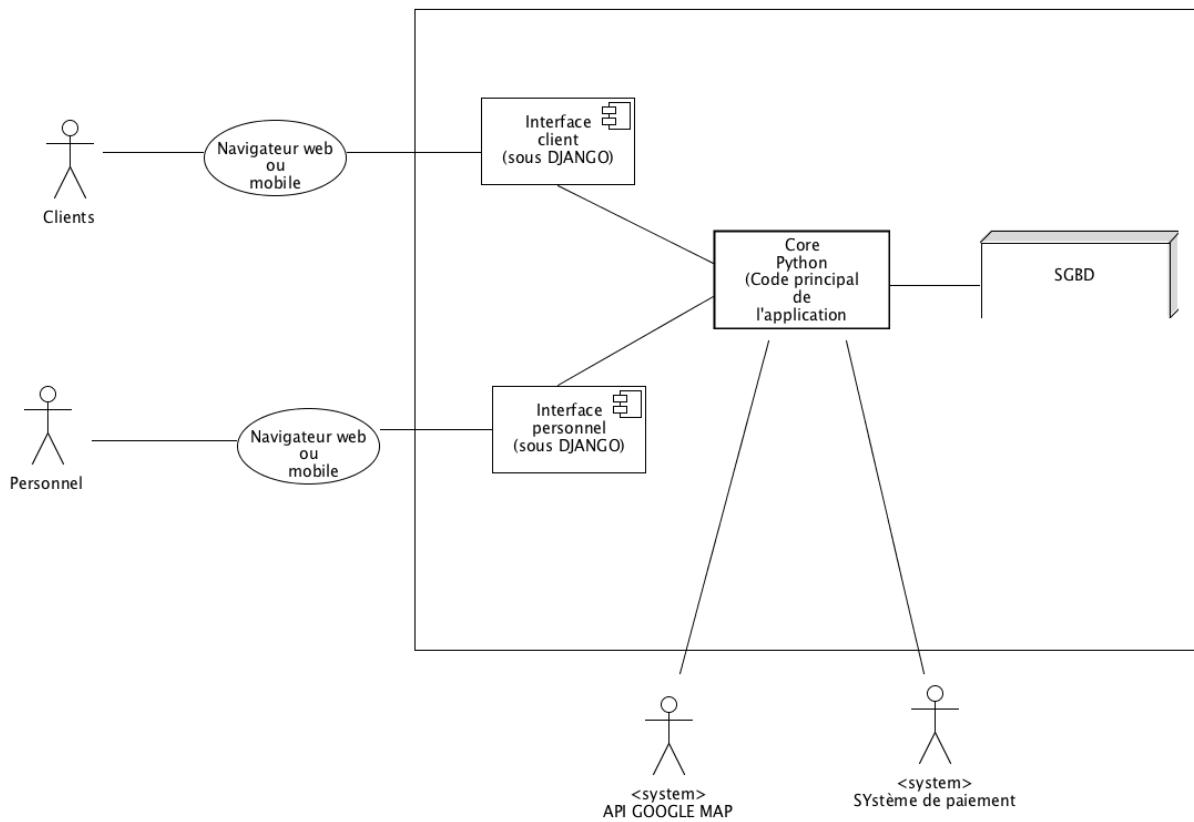


Diagramme de déploiement



Ici, nous avons la représentation de l'architecture du logiciel, avec les utilisateurs qui utiliseront une application web par le biais d'une interface mise en place à l'aide du « Framework DJANGO », qui est le plus utilisé et le plus adéquat avec le langage de programmation « Python ». Nous avons le cœur du programme qui tourne sous python et qui contrôlera les différents composants de l'application ainsi que l'écriture sur notre base de données. Puis les acteurs secondaires de l'application, dans notre cas l'api de Google Map et un Api pour le système de paiement.