mcode
https://www.mcode.com

Réf	MO-001
Date	03/09/2019
Pages	16
Date de création	01/09/2019

Rédigé par :	Approuvé par :

MODIFICATIONS DU DOCUMENT

VERSION	DATE	PAGE	DESCRIPTION
01.0	01/09/2019		Création du document
02.0	03/09/2019		Modification

TABLE DES MATIERES

1 - INTRODUCTION

1.1 - Objet du document

Le présent document constitue le dossier de conception technique de la future solution développée pour OC Pizza.

Il met en évidence le modèle fonctionnel du système ainsi que le modèle physique de données qui servira à la conception de la base de données OC Pizza.

Les composants internes et externes seront aussi analysés afin d'expliquer les liens du système avec les applications externes (Google Maps, Banque).

L'architecture de déploiement sera abordée pour identifier les éléments matériels et les éléments logiciels qui leurs sont rattachés.

Ces spécifications permettent de décrire comment la solution proposée sera implémentée d'un point de vue technique.

1.2 - Contexte

Les éléments du présent dossier se réfèrent au cahier des charges que nous avons reçu, ci -après.

Cahier des charges OC Pizza

« 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 prévoit d'en ouvrir au moins 3 de plus d'ici la fin de l'année.

Mettre en place un système informatique, déployé dans toutes ses pizzerias et qui lui permettrait :

- d'être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;
- de proposer un aide-mémoire aux pizzaiolos indiquant la recette de chaque pizza ;
- 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.

1.3 - Enjeux et Objectifs

Ce document est la deuxième étape d'une démarche qui se réalise, en collaboration avec OC Pizza, dans l'optique de répondre au plus près à ses besoins et à ceux de ses clients utilisateurs.

Les réunions à venir nous permettront de confirmer les spécifications techniques de la solution, en accord avec les spécifications fonctionnelles validées (cf. Dossier de spécifications fonctionnelles joint).

2 - LE DOMAINE FONCTIONNEL

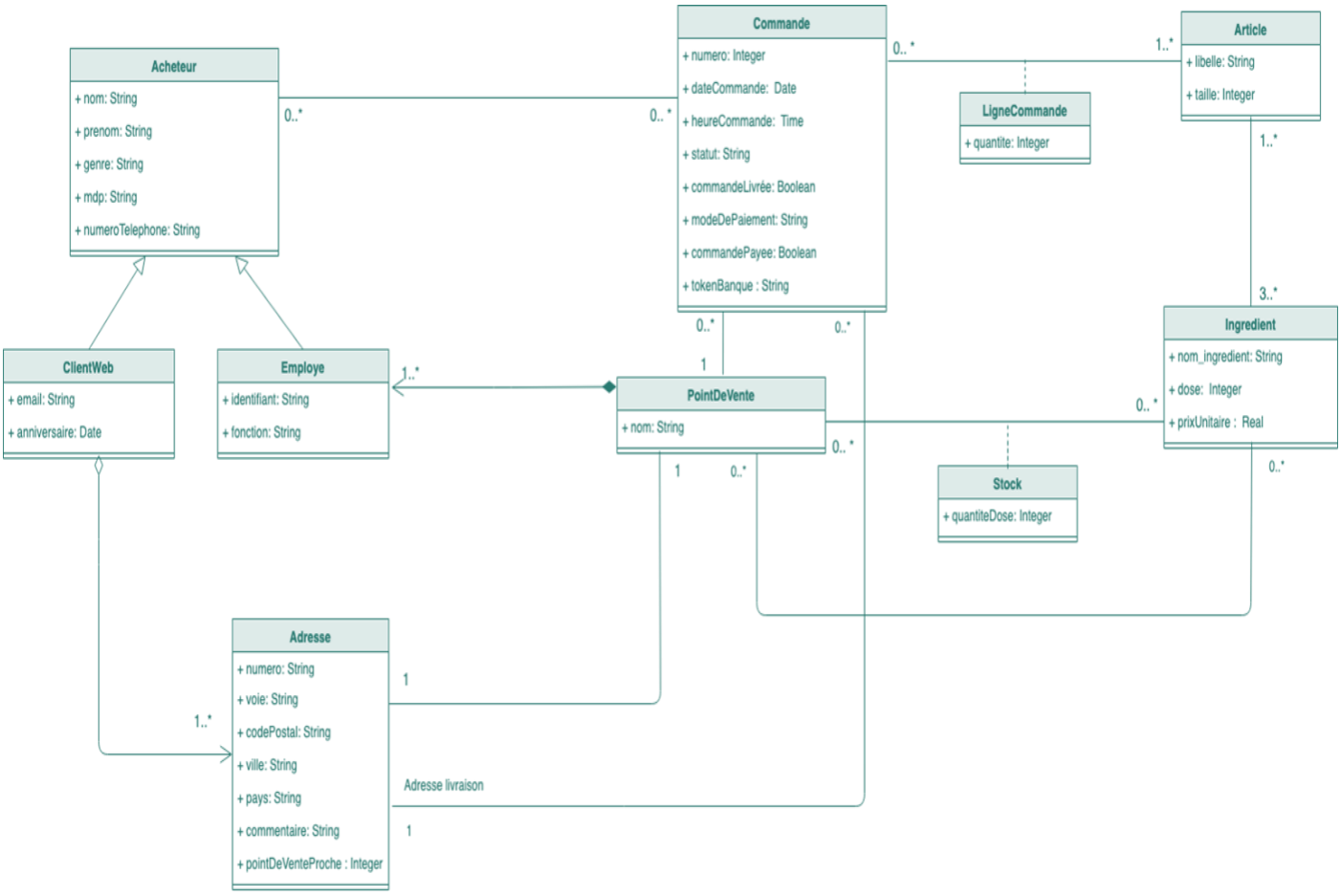
2.1 - Description

Le domaine fonctionnel du système OC Pizza établit l'ensemble des classes liées entre elles qui serviront de support à la programmation en Python et à la création du Modèle Physique de Données.

Le domaine fonctionnel est représenté par un diagramme UML, ici le Diagramme de classes.

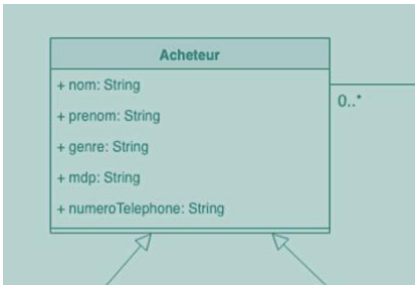
2.2 - Diagramme de classes UML d'OC Pizza

Le diagramme de classes UML (Class Diagram) représente ainsi l'organisation de l'information grâce aux différentes classes et aux liens entre-elles.



2.3 - Descriptif des classes

2.3.1 - Classe « Acheteur »



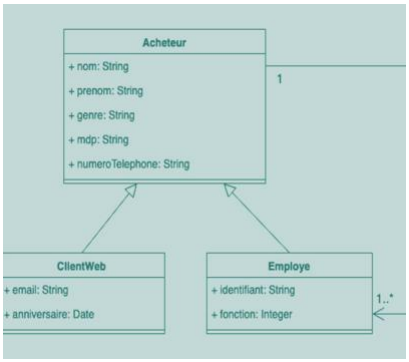
Cette classe regroupe les attributs communs à tous les « Acheteurs ».

Cette **classe mère est associée aux classes filles ClientWeb et Employé.**

Elle est aussi associée à la classe Commande (many-to-many), car l'«Acheteur» saisit une commande. Chaque Acheteur peut faire zéro ou une infinité de commandes (cardinalité 0..*) et chaque commande peut être attribuée à un ou aucun Acheteur dans le cas d'une suppression de compte (cardinalité 0..*)

ATTRIBUT	DESCRIPTION
Prenom	Prénom de l'Acheteur
Nom	Nom de l'Acheteur
Genre	Genre de l'Acheteur
Mot_de_passe	Mot de passe de l'Acheteur
NumeroTelephone	Numéro de téléphone de l'Acheteur

2.3.2 - Classes « Employé » et « ClientWeb »



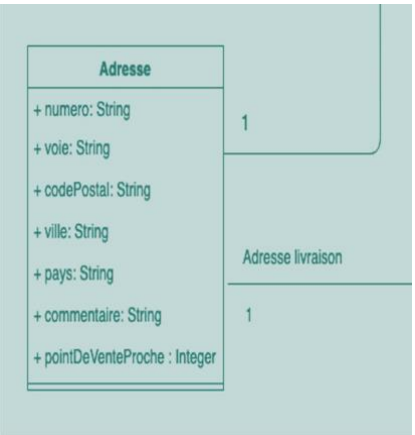
Ces deux classes héritent de la classe Acheteur. Ce sont des **spécialisations de l'« Acheteur »**, association one-to-one, avec dans le premier cas les attributs des **ClientWeb** et dans le deuxième les attributs des **Employés** (utilisateurs salariés d'OC Pizza).

ATTRIBUT	DESCRIPTION
Email	Email du client Internet
Anniversaire	Date de naissance du client Internet

La classe **Employé** est associée à la classe **PointDeVente** par une association de composition (one to many) : les utilisateurs salariés des différentes pizzerias du groupe OC Pizza sont affectés à un point de vente.

ATTRIBUT	DESCRIPTION
Nom	Nom de l'employé
Fonction	Fonction de l'employé

2.3.3 - Classe « Adresse »

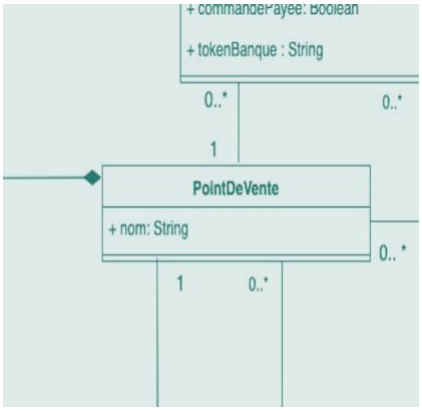


Cette classe regroupe tous les attributs des adresses soit des commandes soit des points de vente.

La classes **Adresse** a une association one to one avec la classe **PointDeVente** (chaque pizzeria possédant une adresse).

ATTRIBUT	DESCRIPTION
numero	Numero de l'adresse
voie	Voie de l'adresse
codePostal	Code postal de l'adresse
ville	Ville de l'adresse
pays	Pays de l'adresse
Commentaire	Complément lié à l'adresse
pointDeVenteProche	Indication de la pizzeria la plus proche de l'adresse indiquée

2.3.4 - Classe « PointDeVente »



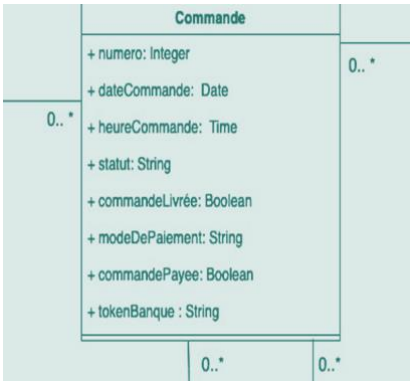
Cette classe a pour seul attribut nom. Elle a des associations avec les classes **Employé, Adresse, Commande et Ingredient**.

Les deux premières associations ont été décrites ci-dessus. L'association avec **Commande**, de type one-to-many, relie la commande à un point de vente qui effectuera ladite commande.

Il y a donc aucune ou une infinité de commandes passées à une pizzeria donnée du groupe (cardinalité 0..*) et chaque commande n'est affectée qu'a un seul point de vente (cardinalité 1).

ATTRIBUT	DESCRIPTION
nom	Nom du point de vente

2.3.5 - Classe « CommandeClient »



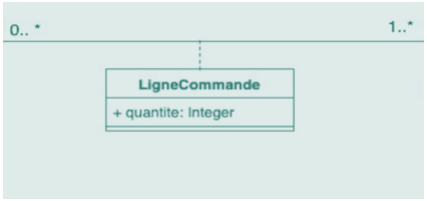
Cette classe regroupe les attributs nécessaires pour une commande de pizza(s). Elle a des associations avec les classes **Acheteur, PointDeVente, Adresse et Article**.

Les trois premières associations ont été décrites ci-dessus.

L'association avec Article (pizza) est de type many-to-many, avec la cardinalité 0..*, car pour un article il peut y avoir 0 ou une infinité de commandes ; et la cardinalité 1..*, car pour une commande il y a forcément au moins un article.

ATTRIBUT	DESCRIPTION
numero	Numéro de la commande
dateCommande	Date de la commande
heureCommande	Heure de la commande
statut	Statut de la commande
commandeLivree	Commande à livrer ou non : vrai ou faux
modePaiement	Mode de paiement choisi
commandePayee	Paiement effectué ou non : vrai ou faux
tokenBanque	Token renvoyé à la Banque après le paiement par CB (attribut non obligatoire car un autre mode peut être choisit)

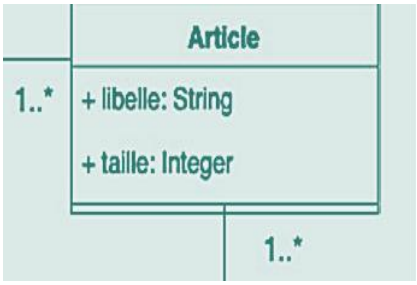
2.3.6 - Classe « LigneCommande»



LigneCommande est une classe d'association, permettant d'ajouter l'attribut quantité à l'association entre les classes **Commande et Article** (type many-to-many).
Cette classe permet de comptabiliser pour une commande combien d'articles de tel type font partie du panier d'achat.

ATTRIBUT	DESCRIPTION
quantite	Quantité commandée pour un article donné

2.3.7 - Classe « Article »



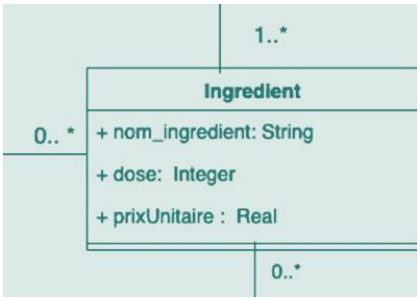
Cette classe regroupe les attributs d'une pizza : le libellé et la taille. Elle a des associations avec les classes **Commande et Ingredient.** La première association a été décrite ci-dessus.

L'association avec **Ingredient** est de type many-to-many, avec la cardinalité 1..*, car un ingrédient se retrouve dans au moins une pizza ou article ; et la cardinalité 1..*, car une pizza ou article contient au moins 1 ingrédient.

Pour la taille (size) des pizzas, la taille « Grand format » est le double de la taille « Format normal » et aura donc le double de dose de tous les ingrédients.

ATTRIBUT	DESCRIPTION
libelle	Nom de l'article
taille	Format de l'article

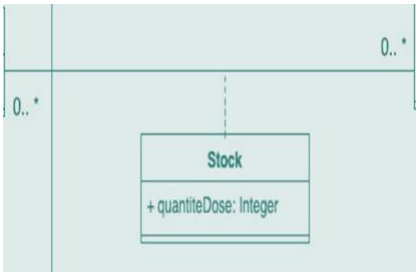
2.3.8 - Classe « Ingredient»



Cette classe regroupe les attributs d'un article. Elle a des associations avec les classes **Article** (décrite ci-dessus) et **PointDeVente.**
L'association avec **PointDeVente** est de type **many-to-many** avec la **cardinalité 0..* dans les deux sens**.
Chaque pizzeria (5 aujourd'hui, potentiellement plus dans le futur) stocke (Stocks) tous les types d'ingrédients qui sont nécessaires à la préparation des commandes.
Un ingrédient peut n'être dans aucune pizzeria en cas de rupture de stock générale, de même, dans une pizzeria il peut n'y avoir aucun ingrédient.

ATTRIBUT	DESCRIPTION
nom_ingredient	Nom de l'ingrédient
dose	Quantité de chaque ingrédient
prix_unitairettc	Prix TTC de chaque ingrédient

2.3.9 - Classe « Stock »



Stock est une classe d'association, permettant d'ajouter l'attribut quantiteDose à l'association entre les classes **Ingredient** et **PointDeVente** (type many-to-many).

Cette classe permet de comptabiliser la quantité de doses de chaque ingrédient dans chaque pizzeria. Cette quantité peut être nulle en cas de rupture de stock.

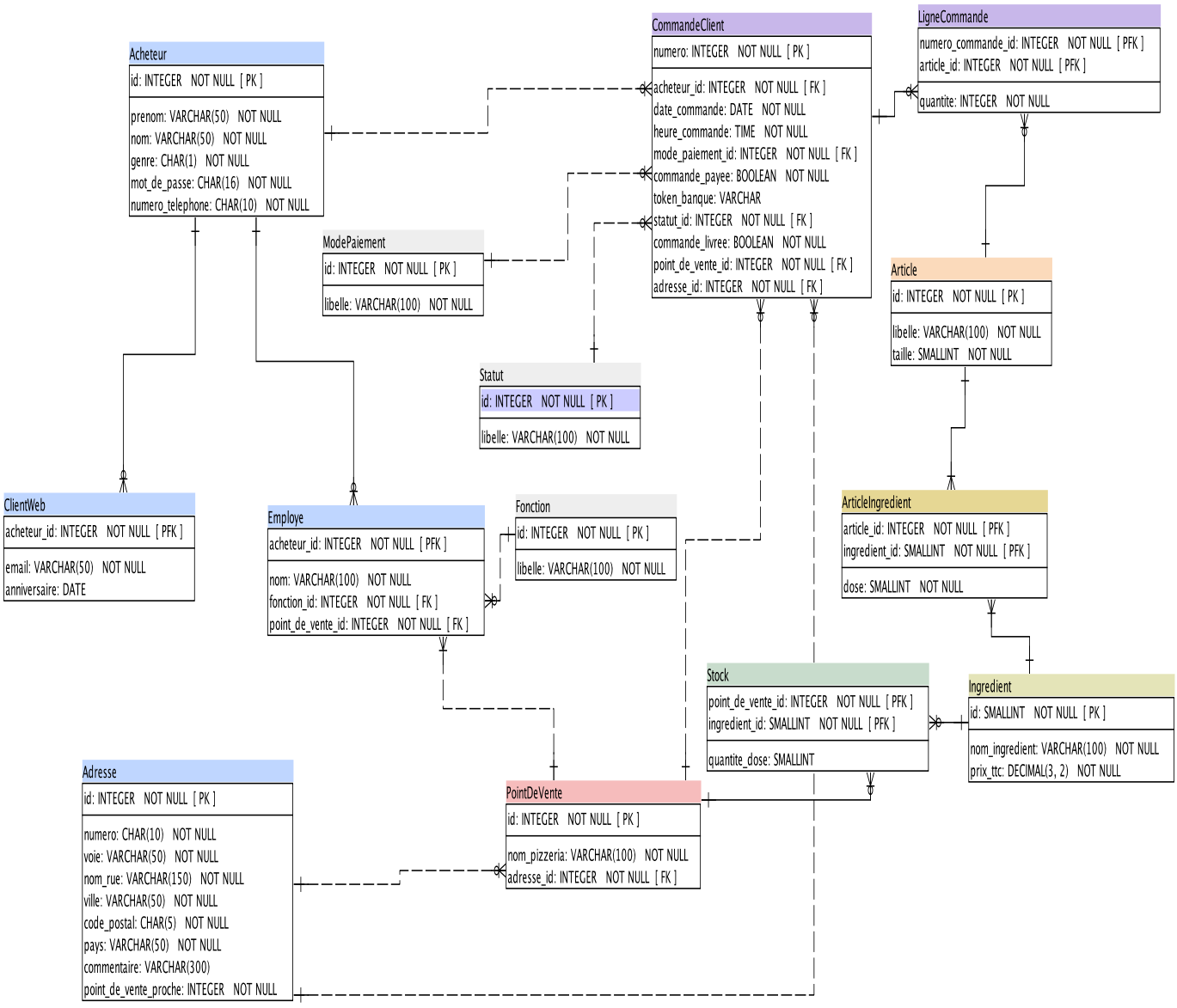
ATTRIBUT	DESCRIPTION
Quantite_dose	Quantité de doses d'un ingrédient dans chaque pizzeria du groupe

3 - LE MODELE PHYSIQUE DE DONNEES

3.1 - Description

Le **Modèle Physique de Données** (Physical Data Model) va permettre de modéliser dans le détail la base de données relationnelle d'OC Pizza.
Nous décrirons les tables et les liens entre-elles, les types de données des différentes colonnes de chaque table et les clés primaires et étrangères.

3.2 - Modèle Physique de Données d'OC Pizza



3.3 - Descriptif des tables principales

3.3.1 - Table « Acheteur »

Cette table regroupe les informations de tous les « Acheteurs », en général.
Toutes les colonnes de cette table sont renseignées **(NOT NULL)**.
La clé primaire est id, avec un **AUTO_INCREMENT**, et de type **INTEGER**.
Pour les autres colonnes, le type est :

- **VARCHAR**, avec **50** caractères pour le **nom** et le **prenom** ;
- **CHAR**, avec **10** caractères pour **numero_telephone**, **1** pour **genre** (« H » ou « F »), et **16** caractères pour le **mot_de_passe**.

3.3.2 - Table « ClientWeb »

Cette table regroupe les informations spécifiques d'une **spécialisation** des « Acheteurs » que sont les « Clients Internet ».
La colonne **email (VARCHAR, 100**) est obligatoirement renseignée **(NOT NULL).**
La colonne **anniversaire** est de type **DATE.**
La colonne **acheteur_id est une clé étrangère** se référant à l'**id** correspondant dans la table **Acheteur**.

3.3.3 - Table « Employe »

Cette table regroupe les informations spécifiques d'une **spécialisation** des « Acheteurs » que sont les employés d'OC Pizza.
Ici, **toutes les colonnes sont à renseigner (NOT NULL).**
Le nom est un VARCHAR (50) et la fonction un INTEGER.
La colonne **acheteur_id** est **une clé étrangère** se référant à l'**id** correspondant dans la table **Acheteur.**

3.3.4 - Table « Adresse »

Les colonnes numero, voie, nom_rue, ville, code_postal, pays et point_de_vente_proche doivent être renseignées (NOT NULL).
La colonne **commentaire** n'est pas à renseigner obligatoirement. Elle permet d'ajouter des précisions par rapport à l'adresse (étage, code...).
La **clé primaire est id**, avec un **AUTO_INCREMENT**, et de type **INTEGER**.
Pour les colonnes, le type de données est :

- **VARCHAR**, avec **150** caractères pour **nom_rue** ;
- **VARCHAR**, avec **50** caractères pour **voie, ville, pays** ;
- **CHAR**, avec **10** caractères pour **numero** et **5** pour **code_postal** ;
- **VARCHAR**, avec **300** caractères pour **commentaire** ;

3.3.5 - Table « PointDeVente »

La colonne nom, de type **VARCHAR**, avec **100** caractères, doit être renseignée **(NOT NULL)**.
La **clé primaire** est **id**, avec un **AUTO_INCREMENT**, de type **INTEGER.**
La colonne **adresse_id est une clé étrangère** se référant à l'**id** correspondant dans la table **Adresse**, afin de récupérer l'adresse de chaque pizzeria.

3.3.6 - Table « Commande »

Toutes les colonnes, à part bank_token (car le paiement n'est pas forcément par carte bancaire), doivent être renseignées **(NOT NULL).**

La clé primaire est numero, avec un **AUTO_INCREMENT**, et de type **INTEGER**.
Pour les colonnes, le type de données est :

- **DATE**, pour **date_commande** ;
- **TIME**, pour **heure_commande** ;
- **INTEGER**, pour **statut** ;
- **BOOLEAN**, pour **commande_livree** (true ou false si la commande est à livrer à domicile ou non) ;
- **INTEGER**, pour **mode_paiement**;
- **BOOLEAN**, pour **commande_payee** (true ou false si le paiement a été effectué ou non) ;
- **VARCHAR**, avec pour **bank_token**.

Il y a **3 clés étrangères** :

- la colonne **acheteur_id** se référant à l'**id** correspondant dans la table **Acheteur**, afin de relier chaque commande à un « acheteur ».
- la colonne **point_de_vente_id** se référant à l'**id** correspondant dans la table **PointDeVente**, afin de relier chaque commande à une pizzeria du groupe. Le choix de la pizzeria dépend du résultat du calcul de géolocalisation entre le lieu de livraison et les différentes pizzerias : la plus proche sera choisie.
- la colonne **adresse_id** se référant à l'**id** correspondant dans la table **Adresse**, afin de relier chaque commande à l'adresse du client.

3.3.7 - Table « LigneCommande »

La table **LigneCommande** est une **table d'association** (association many-to-many).
La colonne **quantite**, de type **INTEGER**, doit être renseignée (**NOT NULL**). C'est la quantité de chaque type d'article commandé.
Il y a **2 clés étrangères** :

- la colonne **numero_commande** de la table **Commande**, afin de relier chaque ligne de commande à une commande.
- la colonne **article_id** se référant à l'**id** de la table **Article**, afin de relier chaque ligne de commande à un article.

3.3.8 - Table « Article »

La colonne **libelle**, de type **VARCHAR**, avec **100** caractères, doit être renseignée **(NOT NULL).**
La colonne **taille**, de type **SMALLINT**, doit être renseignée **(NOT NULL)**. Elle prend la valeur **1** pour les articles de taille « Format standard» et **2** pour les articles de taille « Grand Format».
La clé primaire est id, avec un **AUTO_INCREMENT**, et de type **INTEGER**.

3.3.9 - Table « Ingredient »

La colonne **nom_ingredient**, de type **VARCHAR**, avec **100** caractères, doit être renseignée **(NOT NULL).**
La colonne **prix_ttc**, de type **DECIMAL**, de **3 chiffres dont 2 après la virgule**, doit être renseignée **(NOT NULL)**.
La clé primaire est id, avec un **AUTO_INCREMENT**, et de type **INTEGER**.

3.3.10 - Table « ArticleIngredient »

La table **ArticleIngredient**, n'apparaît pas dans le Diagramme de classes. C'est une **table d'association** qui doit être créée du fait de l'association many-to-many entre les tables **Article** et **Ingredient**.
La colonne dose, de type **SMALLINT**, doit être renseignée **(NOT NULL)** et prend la valeur **1** pour les articles « Format normal», et **2** pour les articles « Grand Format ». C'est le nombre de dose de chaque ingrédient que nécessite chaque article.

Il y a 2 clés étrangères :

- la colonne **article_id** se référant à l'id correspondant dans la table **Article**, afin de relier chaque ingrédient à un article donné.
- la colonne **ingredient_id** se référant à l'id correspondant dans la table **Ingredient**, afin de relier chaque article à un ingrédient donné.

3.3.11 - Table « Stock »

La table **Stock** est une table d'association (association many-to-many).
La colonne **quantite_dose**, de type **SMALLINT**, doit être renseignée mais peut être de valeur nulle (fin de stock d'un ingrédient). C'est la quantité de doses que chaque pizzeria possède de chaque ingrédient.
Il y a 2 clés étrangères :

- la colonne **point_de_vente_id** se référant à l'**id** correspondant dans la table **PointDeVente**, afin de relier chaque pizzeria donnée à un stock d'ingrédient.
- la colonne **ingredient_id** se référant à l'**id** correspondant dans la table **Ingredient**, afin de relier un stock d'ingrédient donné à chaque pizzeria.

3.3.12 - Liens

Documentation PostgreSQL : https://docs.postgresql.fr/11/

Fichiers SQL
- Création de la base de données :
https://github.com/MelanieObr/OC_PIZZA_P7/blob/master/create_db_oc_pizza_v3.sql

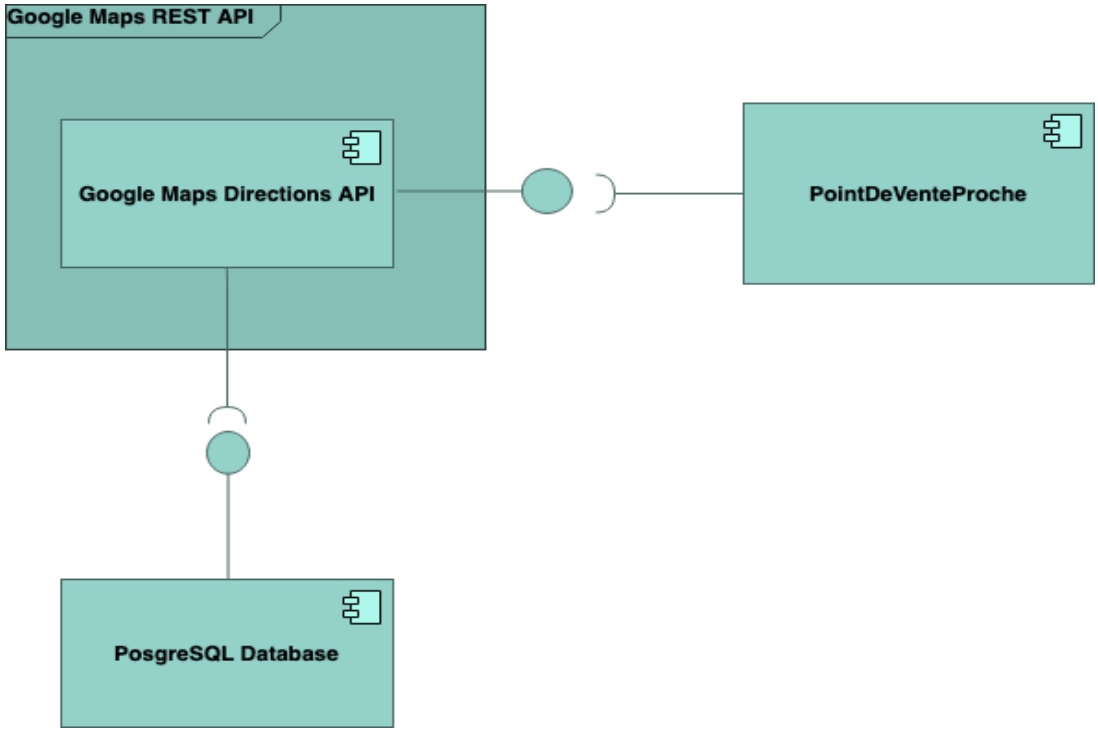
- Fichier de démo :
https://github.com/MelanieObr/OC_PIZZA_P7/blob/master/oc_pizza_demo_v3.sql

4 - COMPOSANTS EXTERNES DU SYSTEME

4.1 – Description

Les Diagrammes de composants (Component Diagrams) décrivent l'organisation du système du point de vue des éléments logiciels. Ils mettent en évidence les dépendances entre les composants, et décrivent ici les interfaces entre les composants internes du système OC Pizza et les composants externes (Banque, Google Maps).

4.2 - La géolocalisation (diagramme et description)



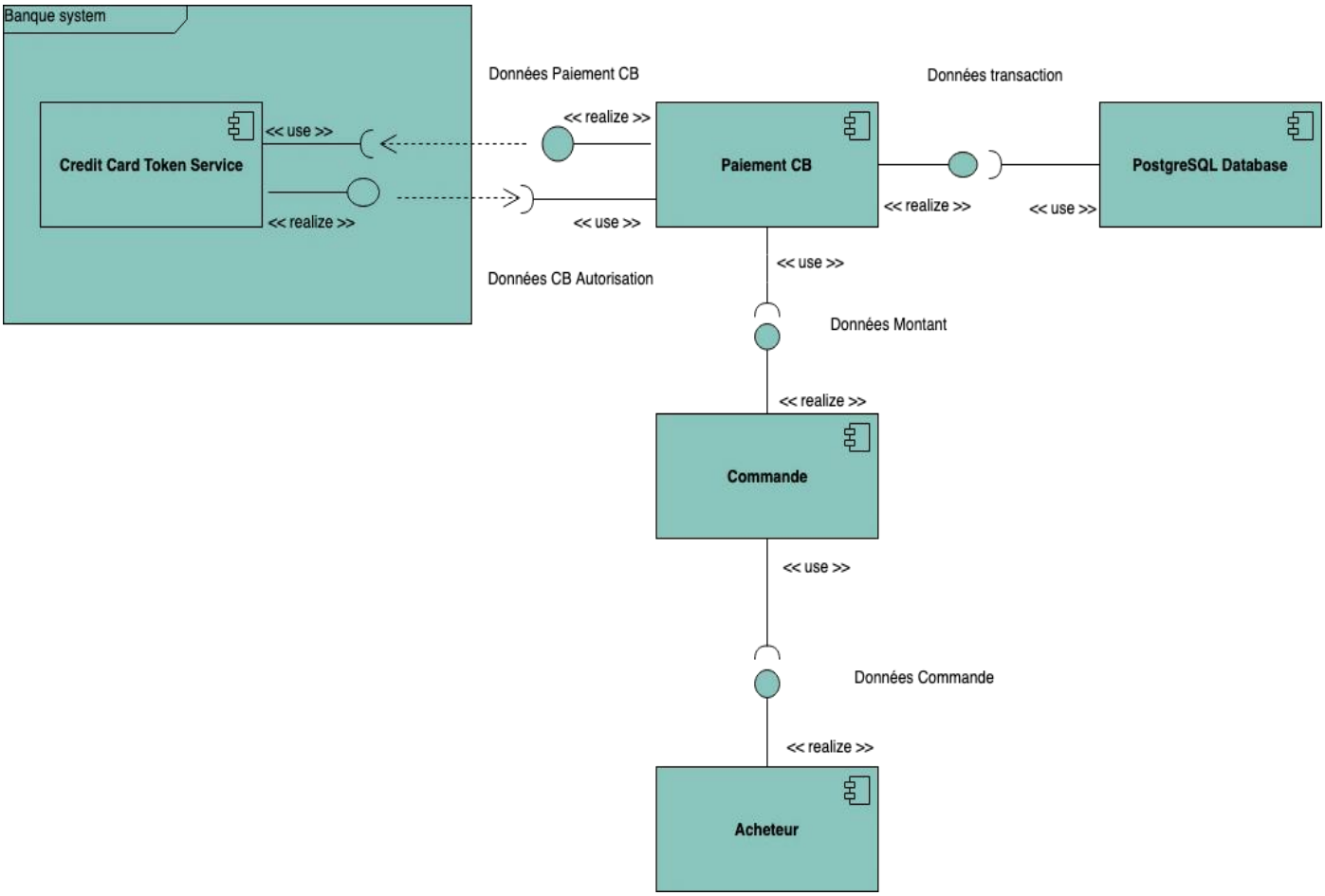
Le composant PostgreSQL Database envoie à l'API REST Google Maps Directions les informations sur l'adresse de livraison d'un client et l'adresse des différentes pizzerias du groupe OC Pizza.

Ce composant externe calcule le temps de parcours et l'itinéraire à privilégier pour un véhicule, entre cette adresse et celles des points de vente et envoie l'information à l'interface requise du composant PointDeVenteProche qui va retenir la pizzeria la plus proche du lieu de livraison.
Cette information (la pizzeria la plus proche) sera stockée dans la base de données dans la colonne point_de_vente_proche de la table Adresse pour l'adresse de livraison donnée.

Documentation : https://developers.google.com/maps/documentation/directions/start?hl=fr

4.3 - Le paiement en ligne (diagramme et description)

Tokeniser une carte revient à prendre le **numéro de la carte** et à le **transformer en jeton (token)** afin que les données ne soient plus sensibles. Ce qui garantit la sécurité des données, quel que soit le mode de paiement, en remplaçant les informations initiales du compte par une série de chiffres.



Le **composant Acheteur envoie au composant Commande les informations sur la commande** et celui-ci calcule le montant total. Ce montant est requis par le **composant Paiement CB** qui récupère aussi le nom du client.

Il échange avec le système bancaire via le composant Credit Card Token Service des informations sur le client et le montant à régler. Le **client** aura directement accès au service de carte bancaire de la banque pour rentrer ses **coordonnées personnelles (numéro de carte, date d'expiration…).**
En retour, le **composant Credit Card Token Service** renvoie un **token** au composant interne avec l'information cryptée sur la réalisation du paiement.

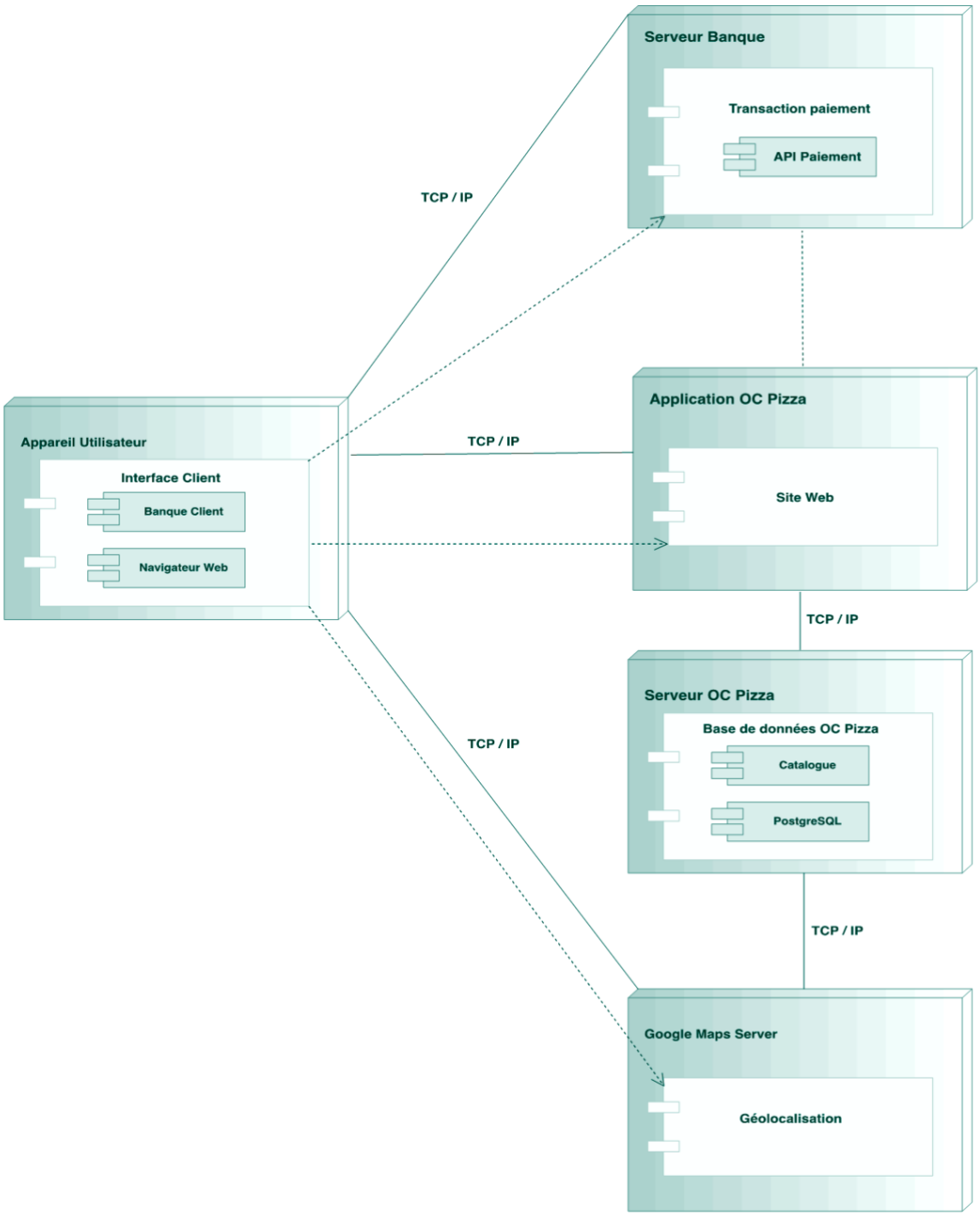
Cette information sera **gardée en base de données, via le composant PostgreSQL** Database, le système d'OC Pizza.

Exemples d'API : Mangopay et Lemonway. Les 2 services proposent un paiement via CB, visa, Mastercard et de nombreux autres moyens de paiement.

5 - ARCHITECTURE DE DEPLOIEMENT

5.1 – Diagramme de déploiement et description

Sur le diagramme, il y a **5 nœuds** qui représentent les **éléments hardware** ou **software** faisant partie du **système.**



-> **Appareil Utilisateur** indique l'ordinateur avec lequel l'utilisateur se connecte. L'**Interface Client** représente l'interface utilisateur, à travers laquelle il peut accéder au site web d'OC Pizza (via le **Navigateur Web**) et payer en ligne (via la **Banque Client**).

-> **Serveur Banque** indique le serveur qui gère les requêtes pour les paiements en ligne, effectués grâce aux APIs des banques (**API Paiement**).
Le transfert des données entre les deux nœuds se fait grâce au protocole TCP/IP. La dépendance entre les composants **Interface Client** et **Transaction Paiement** est marquée par la flèche en pointillés.

-> **Application OC Pizza** indique l'ensemble de pages web qui composent le site web Oc Pizza, comme décrit par le composant **Site Web**.
Le transfert de données entre ce nœud et le nœud **Appareil Utilisateur** se fait grâce au protocole TCP/IP.
La dépendance entre les composants **Interface Client** et **Site Web** est marquée par la flèche en pointillés.

-> **Serveur OC Pizza** indique le serveur chargé de gérer les requêtes des pages du site web.
Il repère les informations grâce à la base de données dédiée **Base de données OC Pizza** et les affiche à l'utilisateur sous forme de **Catalogue**. Les interactions avec la base de données sont assurées par le SGBDR **PostgreSQL**. L'échange des données entre les nœuds **Application OC Pizza** et **Serveur OC Pizza** se fait toujours avec TCP/IP.

-> **Serveur Google Maps** indique le serveur Google chargé de gérer les requêtes de géolocalisation.
Le transfert des données entre ce nœud et **Serveur OC Pizza** est réalisé encore à l'aide du protocole TCP/IP.
Le **Serveur Google Maps** est relié via TCP/IP aussi au nœud **Appareil Utilisateur** et il y a une relation de dépendance entre les composants **Interface Client** et **Géolocalisation**.