OC Pizza
Groupe

OPENCLASSROOMS
Consulting

OC PIZZA

SOLUTION TECHNIQUE D'UN SYSTÈME DE GESTION DE PIZZERIA

Dossier de conception technique

Version 1.1

Auteur
Edouard LOUSSOUARN
Analyste-Programmeur

TABLE DES MATIÈRES

Versions

Auteur	Date	Description	Version
Edouard	11/09/2021	Création du document	0.1
Edouard	12/09/2021	Ajout du MPD et des diagrammes de composants externes	0.2
Edouard	15/09/2021	Rélecture, mise en page et correction	1.1

1 - INTRODUCTION

1.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application OC Pizza. Il a pour objectif de fixer les solutions techniques de l'application web OC Pizza.

Ce document détails les différents composants du domaine fonctionnel. Ainsi que les différentes relations qui lient ses composants.

Les éléments du présents dossiers découlent :

- De la demande du client
- Du développement du sytème

1.2 - Références

Pour de plus amples informations, se référer également aux éléments suivants:

1. **P10_OCPizza_01_ dossier_de_conception_fonctionnelle** : Dossier de conception fonctionnelle
2. **P10_OCPizza_03_ dossier_d_exploitation** : Dossier d'exploitation de l'application

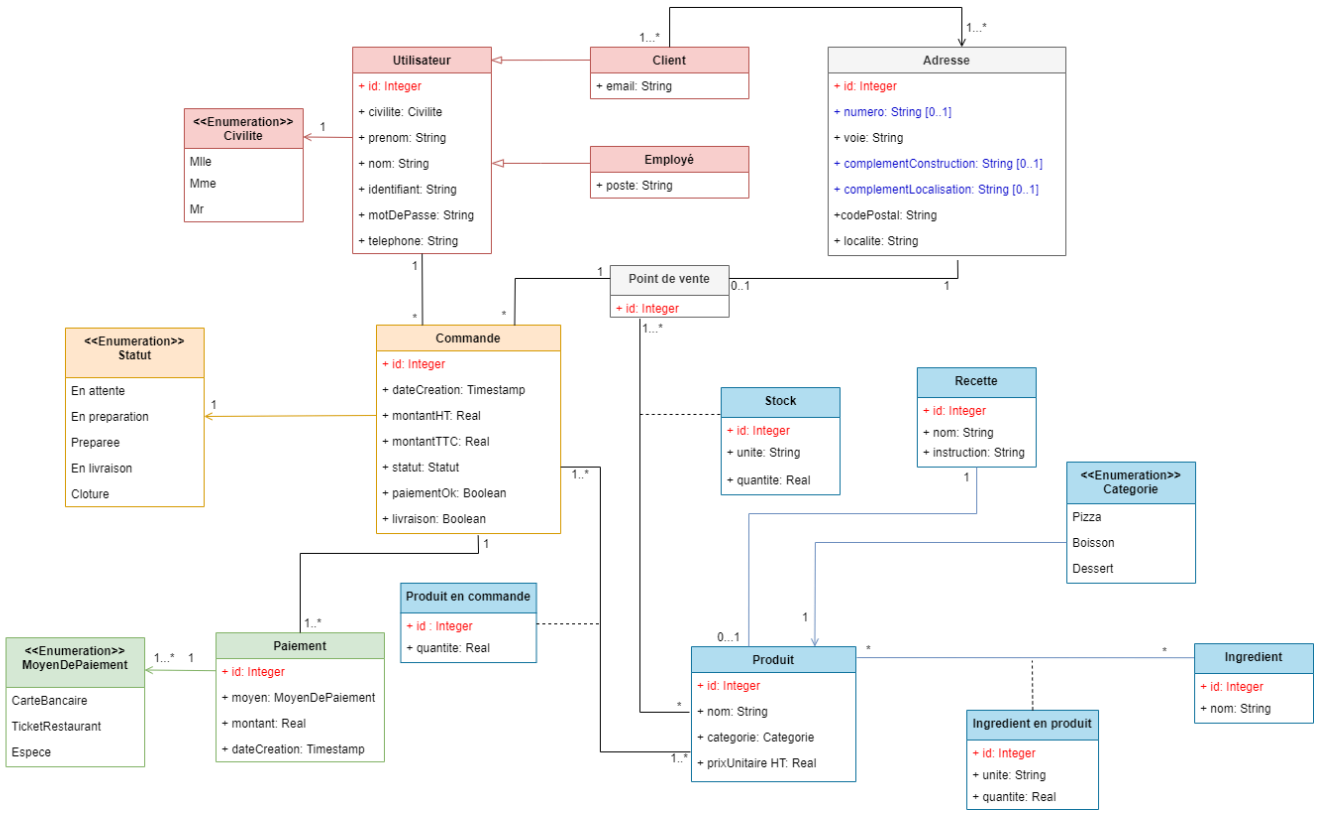
2 - DIAGRAMME DE CLASSE

2.1 - Description

Le **domaine fonctionnel*** d' OC Pizza définit l'ensemble des classes utiles au système. Le domaine est modélisé via le **langage UML*** et un **diagramme de classe*** qui met en évidence les relations entre les différentes classes. Ces classes vont servir de support à la création du modèle physique de données qui sera utilisé pour créer la base de données qui va regrouper les informations du système que l'on veut enregistrer.

Le domaine fonctionnel est constitué de l'ensemble des composants suivants : ceux de la partie « Utilisateur », ceux de la partie « Produit », de la partie « Commande » et de la partie « Paiement ». Il contient aussi deux composants généraux : un composant « Adresse » et un deuxième «Point de vente ».

Diagramme de classe d'OC Pizza



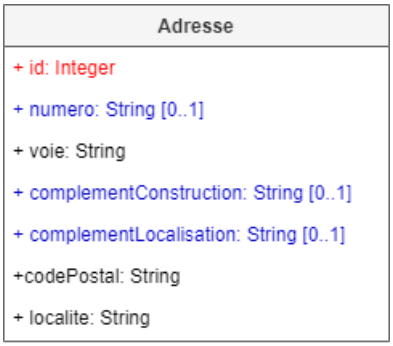
2.1.1 - Les composants généreaux

2.1.1.1 - Composant Adresse

Cette classe est composée des attributs des adresses des clients, des adresses des points de vente et des adresses des commandes.

Un même client peut avoir plusieurs adresses de livraison (domicile, travail..) c'est une relation one-to-many. Une même adresse peut correspondre à différents clients notamment dans le cas où, dans une famille, plusieurs membre de cette famille possède un compte client dans le système (relation one-to-many).

Les champs à renseigner sont les champs qui correspondent aux informations d'une adresse postale classique : le **numéro** et le nom de la **voie**, un complément **d'information** sur la **construction** (cage d'escalier, numéro de bâtiment, code d'accès…) et la **localisation** (nom de résidence…) si nécessaire. Le **code postal** et le nom de la **localité**.



Il existe une relation one-to-one avec la classe **Point de vente** : chaque point de vente possède une adresse.

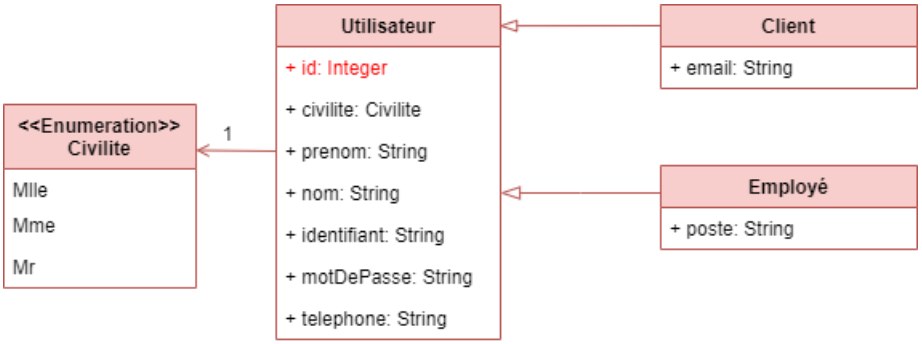
2.1.1.2 - Composant Point de vente

La classe **Point de vente** représente les différentes boutiques du groupe. L'attribut **id** permet de différencier chacun d'entre eux. Cette classe est associées aux classe **Adresse**, **Commande**, **Stock** et **Produit**. Chaque point de vente est associé à un stock.

La relation avec la classe **Commande** est de type one-to-many. Il y a aucune ou plusieurs commandes passées dans un point de vente. Et chaque commande est reliée à un seule point-de-vente.

Idem pour la relation avec la classe **Produit** (cardinalité 0 ..*) , il peut y avoir aucun ou une multitude de produits sur un point de vente.

2.1.2 - Les composants de la partie « Utilisateur »



2.1.2.1 - Composant Utilisateur

La classe **Utilisateur** est la classe mère qui est associée aux classes filles **Client** et **Employé.** Cette classe regroupe les attributs communs à chaque utilisateur. On y trouve sa **civilité**, son **nom**, son **prénom**, son **identifiant**, son **mot de passe** et son numéro de **téléphone**.

2.1.2.2 - Composant Civilité

La classe **Civilité** est une énumération qui propose les choix possibles concernant l'utilisateur : **Mlle**, **Mme** ou **Mr**.

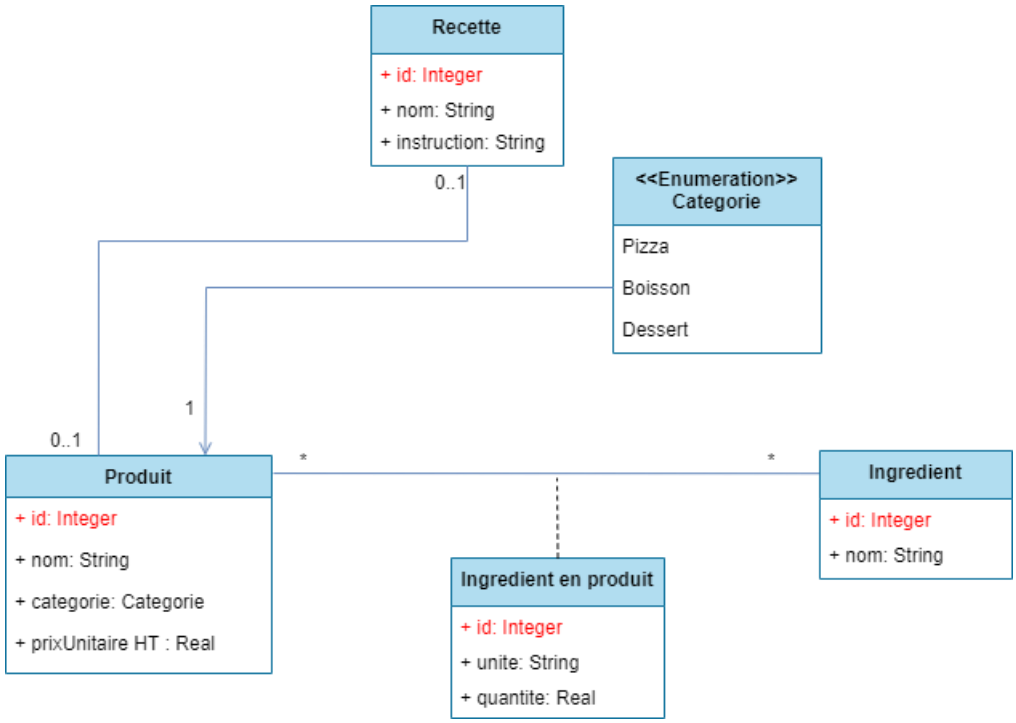
2.1.2.3 - Composant Client

La classe **Client** contient le champ **email** qui est utiliser pour faire parvenir au client les informations relatives aux commandes effectuées.

2.1.2.4 - Composant Employé

La classe **Employé** précise le poste qu'occupe celui-ci dans l'entreprise pour lui affecter des droits d'accès dans le système.

2.1.3 - Les composants de la partie « Produit »



2.1.3.1 - Composant Produit

Chaque produit appartient à une catégorie. Une recette peut être associé à aucun ou un seul produit et réciproquement. Un produit à un attribut **id** qui sert d'identifiant unique, un attribut **nom** et un **prix unitaire HT**.

La relation entre la classe **Produit** et la classe **Ingrédient** est de type many-to-many. Un ingrédient peut constituer aucun, un ou plusieurs produit et inversement un produit peut être constitué d'aucun, un ou plusieurs ingrédients.

2.1.3.2 - Composant Recette

Chaque produit appartient à une catégorie. Une recette peut être associé à aucun ou un seul produit et réciproquement. Un produit à un attribut **id** qui sert d'identifiant unique, un attribut **nom** et un **prix unitaire HT**.

La relation entre la classe **Produit** et la classe **Ingrédient** est de type many-to-many. Un ingrédient peut constituer aucun, un ou plusieurs produit et inversement un produit peut être constitué d'aucun, un ou plusieurs ingrédients.

2.1.3.3 - Composant Ingrédient

Un ingrédient à un **id** unique et un **nom**. On peut avoir aucun ou plusieurs ingrédients par produit. Et un ingrédient peut composer aucun ou plusieurs produits.

2.1.3.4 - Composant Ingrédient en produit

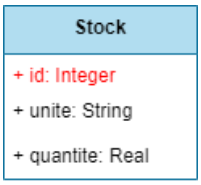
Cette classe est une classe d'association. Elle ajoute un champ **id**, un champ **unité** et un champ **quantité** à l'association entre les tables **Produit** et **Ingrédient**. Cela indique la quantité d'ingrédient qui compose le produit. C'est une relation many-to-many.

2.1.3.5 - Composant catégorie

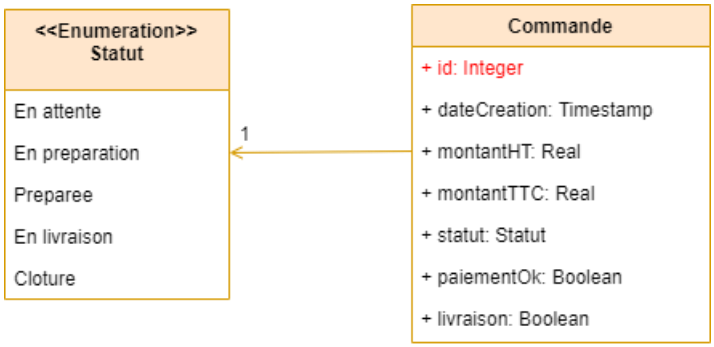
Cette classe permet de préciser si le produit est une **pizza**, une **boisson** ou un **dessert**.

2.1.3.6 - Composant Stock

C'est une classe d'association qui permet d'ajouter les attributs **unité** et **quantité** à l'association entre les tables **Produit** et **Point de vente**. C'est une relation many-to-many.



2.1.4 - Les composants de la partie « Commande »



2.1.4.1 - Composant Commande

La classe **Commande** permet d'enregistrer la **date** et **l'heure** de création d'une commande, les **montants HT** et **TTC** et le **statut** de celle-ci. On précise aussi si la commande est **réglée** et s'il s'agit d'une commande qui doit être **livrée** ou à retirer sur place.

Il existe une relation entre la classe **Commande** et la classe **Utilisateur**. Une commande correspond à un utilisateur et un utilisateur peut passer aucune, une ou une infinité de commande.

Une deuxième relation se déroule avec la classe **Paiement**. Une relation de type one-to-many. A chaque commande correspond un ou plusieurs paiements.

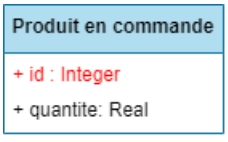
La relation avec le composant **Point de vente** a déjà été évoqué précédemment

2.1.4.2 - Composant Statut

Cette classe précise l'état de la commande si celle-ci est **en attente**, **en préparation**, **préparée**, **en livraison** ou **clôturée**.

2.1.4.3 - Composant Produit en commande

C'est une classe d'association. Elle ajoute un champ **id** et un champ **quantité** à l'association entre les tables **Produit** et **Commande**. Elle précise la quantité d'un type de produit qui compose une commande.



2.1.5 - Les composants de la partie « paiement »

2.1.5.1 - Composant Paiement

Ce composant contient un champ **id**, un champ **moyen de paiement**, le **montant** à payer, une **date** et une **heure** de création. Un paiement peut être effectué avec différents moyen de paiement. La relation entre la classe **Moyen de paiement** et la classe **Paiement** est de type one-to-many.

2.1.5.2 - Composant Moyen de Paiement

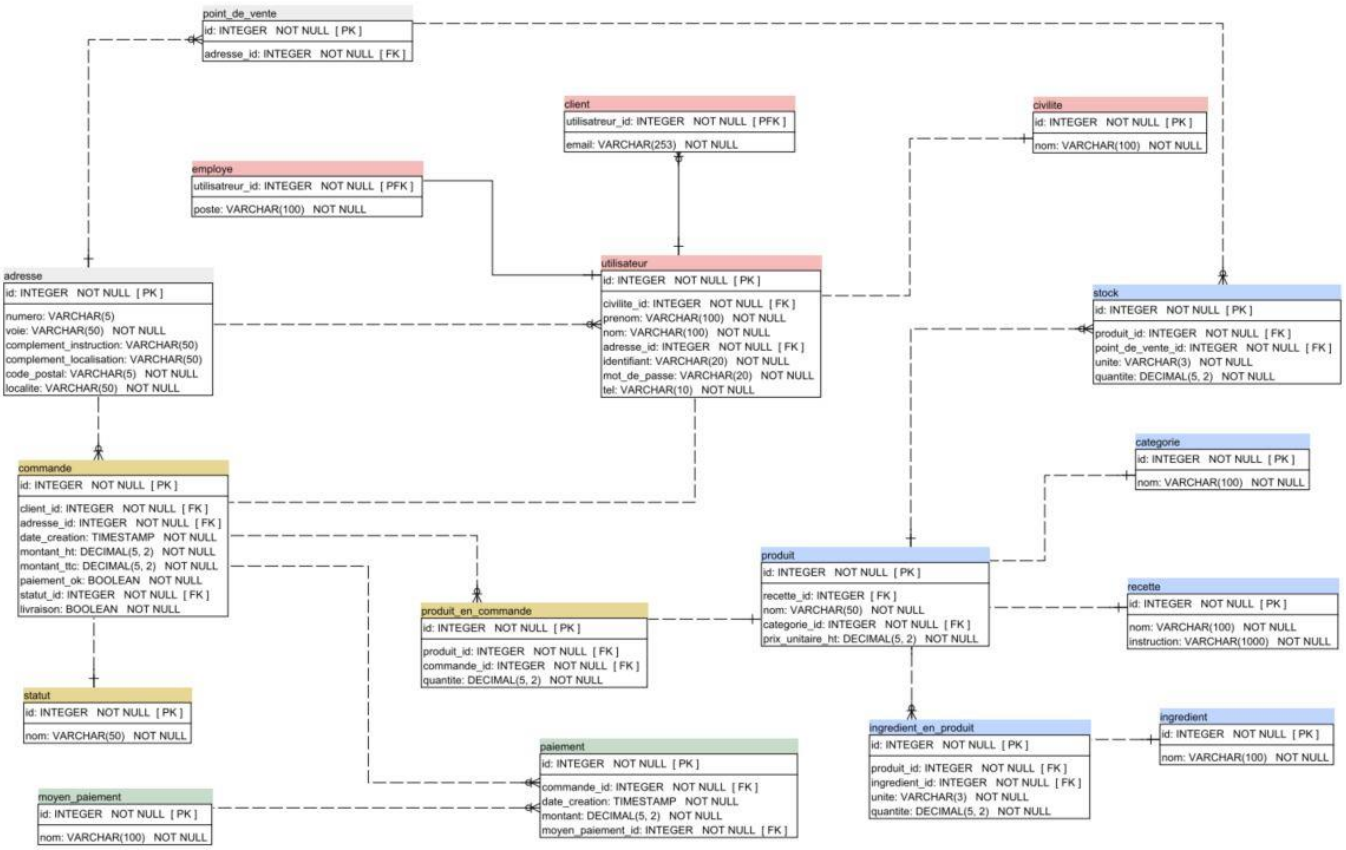
Cette classe contient les champs des différents moyen de paiement : **carte bancaire**, **ticket restaurant** et **espèces**.

3 - Modèle Physique De Données (MPD)

3.1 - Description

Le **model physique de données*** est la dernière étape de l'analyse du système. Il modélise la base de données relationnelle d'OC Pizza. Il décrit les tables et les différents liens entre elles. On trouve les types de données des colonnes qui composent chaque table ainsi que les clés primaires et étrangères.

Model physique de données d'Oc Pizza



3.1.1 - Les tables du MPD

3.1.1.1 - La table Adresse

La table **adresse** regroupe les informations des adresses des clients et des points de vente. Sa clé primaire est **id** et est auto incrémentée. Seul les colonnes **numéro**, **complément d'instruction** et **complément de localisation** acceptent NULL comme valeur.

3.1.1.2 - La table Point de vente

La colonne **id** est la clé primaire et est auto incrémentée. La colonne **adresse_id** est une clé étrangère qui a pour valeur la clé primaire de la colonne **id** de la table **Adresse** qui lui correspond.

3.1.1.3 - La table Utilisateur

Son **id** est la clé primaire et est auto incrémentée. Chacune de ses colonnes ne peut être NULL. La colonne **civilite_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Civilité**. Et la colonne **adresse_id** est une clé étrangère qui a pour valeur la clé primaire de la colonne **id** de la table **Adresse** qui lui correspond.

3.1.1.4 - La table Civilité

Son **id** est la clé primaire et est auto incrémentée. Chacune de ses colonnes ne peut être NULL.

3.1.1.5 - La table Client et employé

Les colonnes **utilisateur_id** sont des clés étrangères qui se réfèrent à l'**id** correspondant de la table **utilisateur**. Chacune des colonnes ne peut être NULL.

3.1.1.6 - La tableProduit

Son **id** est la clé primaire et est auto incrémentée. Chacune de ses colonnes ne peut être NULL sauf la colonne **recette_id** car un produit n'est pas obligatoirement rattaché à une recette. La colonne **recette_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Recette**. Et la colonne **categorie_id** est une clé étrangère qui a pour valeur la clé primaire de la colonne **id** de la table **Catégorie** qui lui correspond.

3.1.1.7 - Les tables Catégorie, recette et ingrédient

Les colonnes **id** sont les clés primaires et sont auto incrémentées. Chacune des colonnes ne peut être NULL.

3.1.1.8 - La table Stock

Son **id** est la clé primaire et est auto incrémentée. La colonne **produit_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Produit**. La colonne **point_de_vente_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Point de vente**. Chacune des colonnes ne peut être NULL.

3.1.1.9 - La table Ingrédient en produit

Son **id** est la clé primaire et est auto incrémentée. La colonne **produit_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Produit**. La colonne **ingrédient_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Ingrédient**. Chacune des colonnes ne peut être NULL.

3.1.1.10 - La table Commande

Son **id** est la clé primaire et est auto incrémentée. La colonne **client_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Utilisateur**. La colonne **adresse_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Recette**. La colonne **statut_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Statut**. Chacune des colonnes ne peut être NULL.

3.1.1.11 - La table Produit en commande

Son **id** est la clé primaire et est auto incrémentée. La colonne **produit_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Produit**. La colonne **commande_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Commande**. Chacune des colonnes ne peut être NULL.

3.1.1.12 - La table Paiement

Son **id** est la clé primaire et est auto incrémentée. La colonne **commande_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Commande**. La colonne **moyen_de_paiement_id** est une clé étrangère qui a pour valeur l'**id** correspondant de la table **Moyen de paiement**. Chacune des colonnes ne peut être NULL.

3.1.1.13 - Les tables Statut et moyen de paiement

Les colonnes **id** sont les clés primaires et sont auto incrémentées. Chacune des colonnes ne peut être NULL.

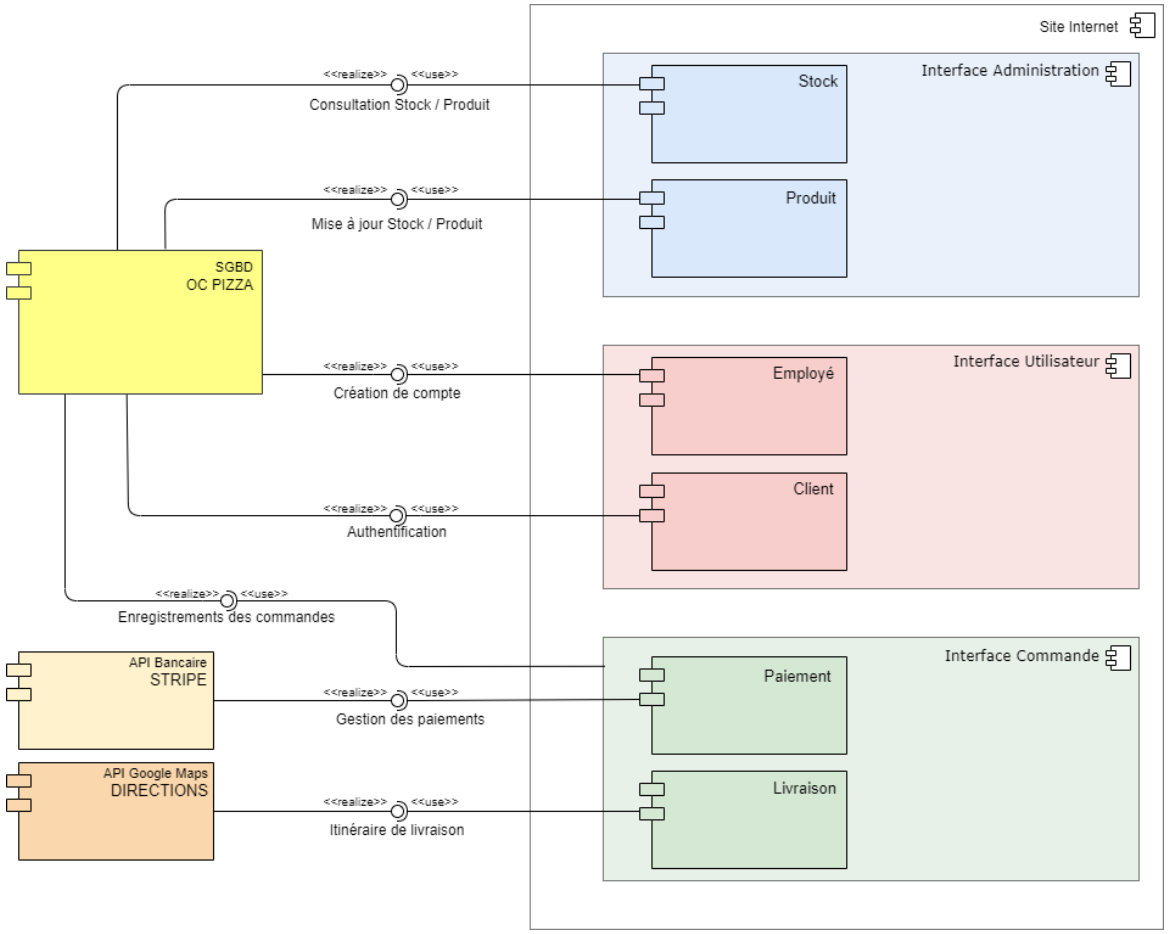
4 - Les Composants Externes

4.1 - Le diagramme de composant

En UML, un composant est un élément logiciel remplaçable et réutilisable qui fournit ou reçoit un service bien précis. Les composants fournissent des services via des interfaces.
On distingue deux types d'interfaces : les interfaces requises qui fournissent un service au composant et dont le composant a besoin pour fonctionner. Et les interfaces fournies pour lesquels le composant fourni lui-même un service.

Le **diagramme de composants*** fait partie des diagrammes structuraux d'UML. Il permet de représenter les différents éléments logiciels (composants) du système et leurs dépendances (relations qui les lient).

Le diagramme de composant



Trois composants externes interviennent dans le système :

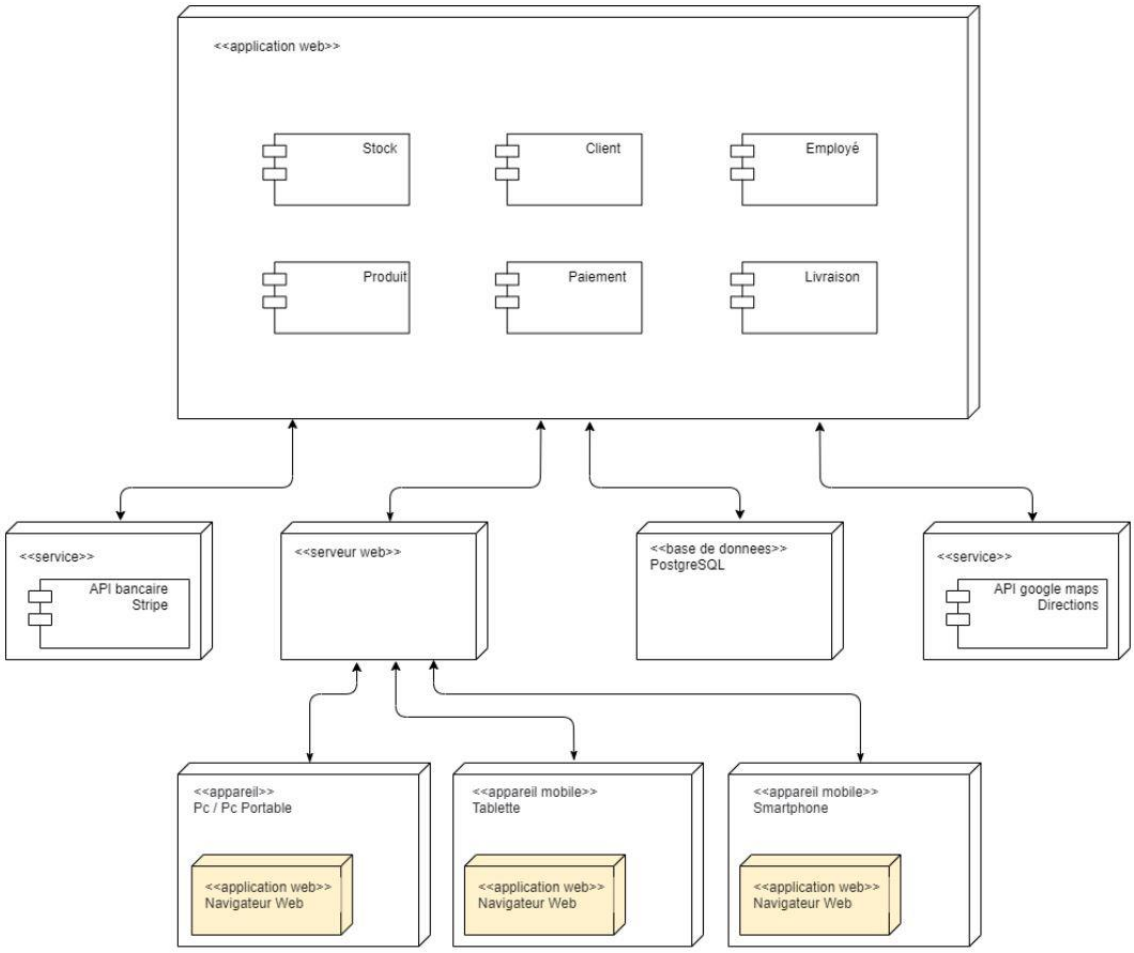
L'API Directions* de **Google Maps*** est le composant utilisé pour calculer le parcours et l'itinéraire que le livreur doit emprunter pour les livraisons à domicile.

L'API **Stripe*** est le système bancaire utilisé pour que les clients puissent régler leurs commandes de manière sécurisée.

La **base de données*** OC Pizza va enregistrer et sauvegarder les informations concernant les commandes, les règlements, les comptes utilisateurs et les informations relatives au stock de produit et d'ingrédients;

4.2 - Le diagramme de déploiement

Le **diagramme de déploiement*** décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. Les nœuds (les boîtes en trois dimensions) représentent les composants du système qu'ils soient logiciels ou matériels.



L'application web est reliée au service bancaire, au service Directions de Google Maps et à la base de données PostgreSQL et au serveur web.

Les utilisateurs accèdent au système à partir d'un Smartphone, d'une tablette, d'un pc ou d'un pc portable.

5 - GLOSSAIRE

Domaine fonctionnel	Il définit l'ensemble des classes utiles au système.
Language UML	Le langage UML (langage de modélisation unifié) a été pensé pour être un langage de modélisation visuelle commun, et riche sémantiquement et syntaxiquement. Il est destiné à l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes par leur structure aussi bien que leur comportement.
Diagramme de classe	Il met en évidence les relations entre les différentes classes. Ces classes vont servir de support à la création du modèle physique de données
Model Physique de données	Il est utilisé pour créer la base de données qui va regrouper les informations du système que l'on veut enregistrer. Il modélise la base de données relationnelle
Diagramme de composant	Il fait partie des diagrammes structuraux d'UML. Il permet de représenter les différents éléments logiciels (composants) du système et leurs dépendances (relations qui les lient).
API	Application Programming Interface : Interface de programmation applicative. Elle permet de "brancher" et d'échanger efficacement des données entre deux applications. Une API permet de développer un langage commun afin que les données d'une application A puissent être transmises et comprises par une application B, toutes les deux programmées dans un langage différent ou situées à deux endroits distincts.
Google Maps	Google Maps est un service de cartographie en ligne.
API Direction	C, est le composant utilisé pour calculer le parcours et l'itinéraire que le livreur doit emprunter pour les livraisons à domicile.
Stripe	Stripe est une plateforme de traitement des paiements qui permet de transférer de l'argent du compte bancaire d'un client vers le compte d'une entreprise au moyen d'une transaction par carte de crédit. C'est un moyen simple d'accepter des paiements en ligne, sans frais d'installation ni frais mensuels. Stripe assure un niveau de sécurité élevé, qui permet de recevoir vos paiements en toute fiabilité.
Le diagramme de déploiement	Il décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. Les nœuds (les boîtes en trois dimensions) représentent les composants du système qu'ils soient logiciels ou matériels.
Base de données	Elle permet de stocker et de retrouver des données structurées, semi-structurées ou des données brutes ou de l'information, souvent en rapport avec un thème ou une activité ; celles-ci peuvent être de natures différentes et plus ou moins reliées entre elles.
PostgreSQL	PostgreSQL est un système de gestion de base de données relationnelle orienté objet puissant et open source qui est capable de prendre en charge en toute sécurité les charges de travail de données les plus complexes.