

OC Pizza

OCPizzapp

Dossier de conception technique

Version v0.1

Auteur
Amaury Bois
Developpeur

TABLE DES MATIÈRES

1 -Versions.....	3
2 -Introduction.....	4
2.1 -Objet du document.....	4
2.2 -Références.....	4
3 -Architecture Technique.....	5
3.1 -Composants.....	5
3.2 -Application Web.....	7
4 -Architecture de Déploiement.....	8
4.1 -Modèle physique de donnée.....	9
5 -Architecture logicielle.....	19
5.1 -Principes généraux.....	19
5.1.1 -Les couches.....	19
5.1.2 -Les modules.....	19
5.1.3 -Structure des sources.....	19
6 -Points particuliers.....	21
6.1 -Ressources.....	21
6.2 -Environnement de développement.....	21
6.3 -Procédure de packaging / livraison.....	21
7 -Glossaire.....	22

1 - VERSIONS

Auteur	Date	Description	Version
Amaury Bois	18/05/20	Création du document	V0.0
Amaury Bois	23/09/20	Mise à jour du document	V0.1

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application **OCPizzapp**.

L'objectif du document est de présenter les outils, les technologies et les méthodes mises en oeuvre pour réaliser l'application.

Les éléments du présents dossiers découlent :

- de l'entretien sur la version précédente de ce document (le 18/05/2020)

2.2 - Références

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

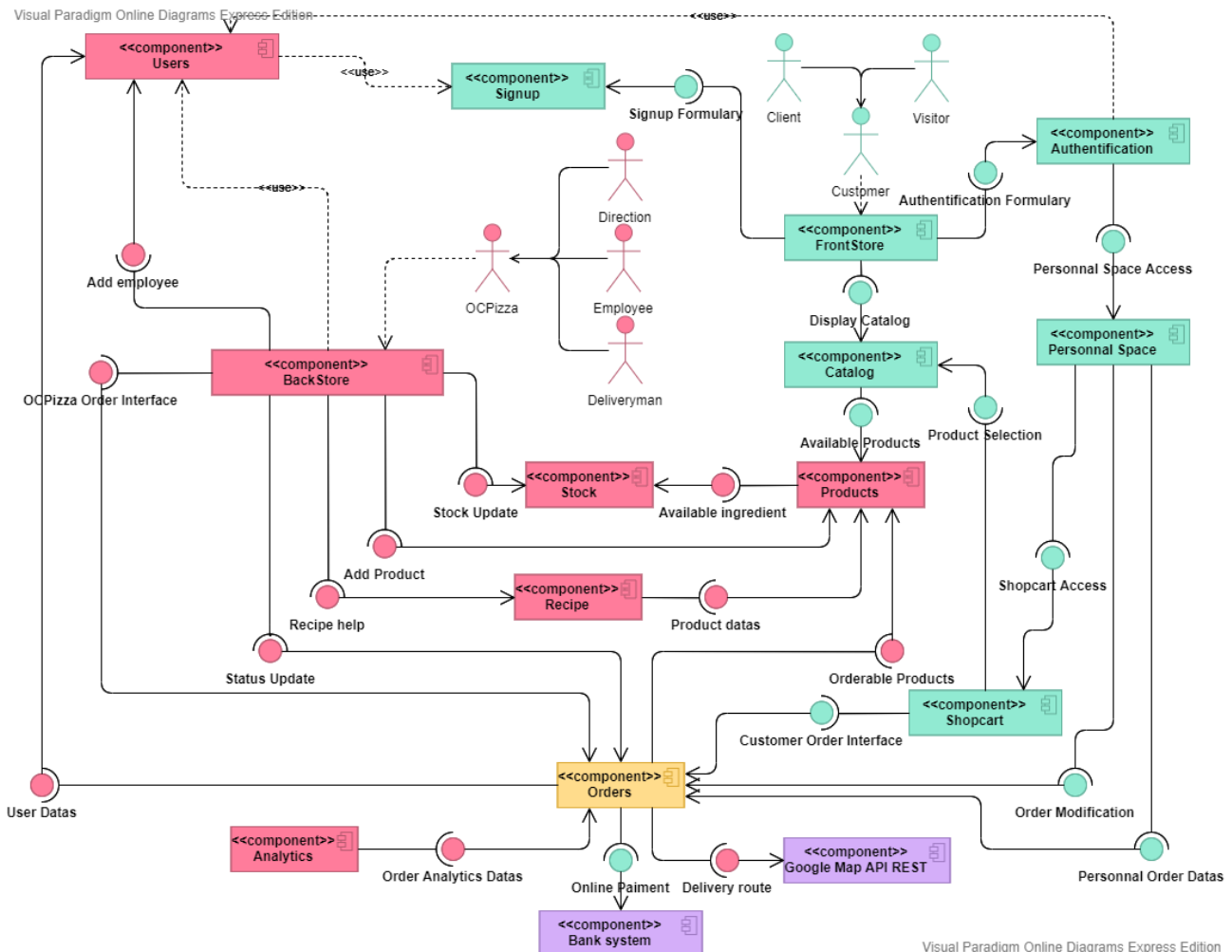
1. **DCF – v0.1** : Dossier de conception fonctionnelle de l'application
2. **DE – V0.0** : Dossier d'exploitation de l'application

3 - ARCHITECTURE TECHNIQUE

3.1 - Composants

Afin de décrire l'organisation du programme d'un point de vue logiciel, nous allons utiliser des diagrammes de composant. Le premier abordera le système d'un point de vue général tandis que les deux suivants aborderons, respectivement, l'API Google map et le système de paiement bancaire.

3.1.1 – Général



Notes :

en Vert : composants et interface pouvant être utilisée par les utilisateurs hors-OC Pizza

en Jaune : composants et interface pouvant être utilisée par tous les utilisateurs

en Rouge : composants et interface pouvant être utilisée par les utilisateurs OC Pizza

en Violet : composants extérieurs

Décrivons brièvement son fonctionnement en abordant chaque composants :

Signup : La clientèle (Customer) l'utilisera afin de créer un compte client sur le système, il est utilisé par le composant **Users** qui recueille les données saisies par l'utilisateur.

Authentication : La clientèle l'utilisera afin de s'authentifier sur le site, ce composant utilisera les données du composant **Users** afin de s'assurer de l'identité du client. S'authentifier donne également accès à l'espace personnel (**Personnal space**).

FrontStore : Ce composant correspond au site web que la clientèle pourra consulter. Il permet au client d'accéder aux composants **Signup** et **Authentication** (afin d'accéder à son espace personnel) mais sert également à afficher le catalogue (**Catalog**) des produits.

Catalog : Ce composant propose les produits disponibles aux clients. Ces derniers peuvent en sélectionner pour en ajouter à leur paniers (**Shopcart**) via **Product Selection**.

Personnal space : Ce composant est l'espace personnel du client. Il peut y passer pour accéder à son panier (**Shopcart**) afin de passer commande, modifier/annuler une commande en cours avec **Order Modification** ou consulter ses commandes présentes et passées via l'interface **Personnal Order Datas**.

Shopcart : Ce composant correspond au panier du client. Il est accessible via l'espace personnel (**Personnal space**) et permet de commander (via **Customer order interface**) les produits ayant été ajouté depuis le catalogue (**Catalog**).

Users : Ce composant rassemble les informations sur les utilisateurs (nom, adresse etc.). Ces informations peuvent être récupérées par le composant **Orders**.

BackStore : Ce composant représente une partie du système uniquement accessible par les employés d'OC Pizza (qui peuvent s'authentifier directement via ce composant). Il permet d'accéder aux interface de commande, de modification de statut d'une commande, d'ajout/modification d'un produit, de consulter une recette, de mettre à jour le stock. Un membre de la direction pourra également ajouter un employé.

Stock : Ce composant rassemble les informations sur le stock d'un restaurant. Ses données peuvent être modifiées via le composant **BackStore**. Il est également lié au composant **Products** afin de déterminer si les ingrédients nécessaires sont en stock afin de pouvoir préparer un produit.

Products : Ce composant rassemble les informations sur les produits proposés par OC Pizza. Il est utilisé par le composant **Catalog** qui récupère les données afin de les afficher aux client. Si les ingrédients nécessaires au produit sont indisponibles, un message sera affiché à l'utilisateur. Des produits peuvent être ajouté/modifié depuis le composant **BackStore**. Enfin, il est lié au composant **Recipe**, auquel il envoi des données sur la composition des produits.

Recipe : Ce composants rassemble les recettes des produits proposés par OC Pizza. Il utilise les données envoyées par le composant **Products** et est consultable via l'**interface Recipe Help** pour les employés ayant besoin d'un aide-mémoire pour une préparation.

Orders : Ce composant rassemble les informations des commandes. Il récupère les produits commandés, les données utilisateurs telle que l'identité du client ou l'adresse de livraison, le statut de la commande, modifiable via l'**interface Status Update**.

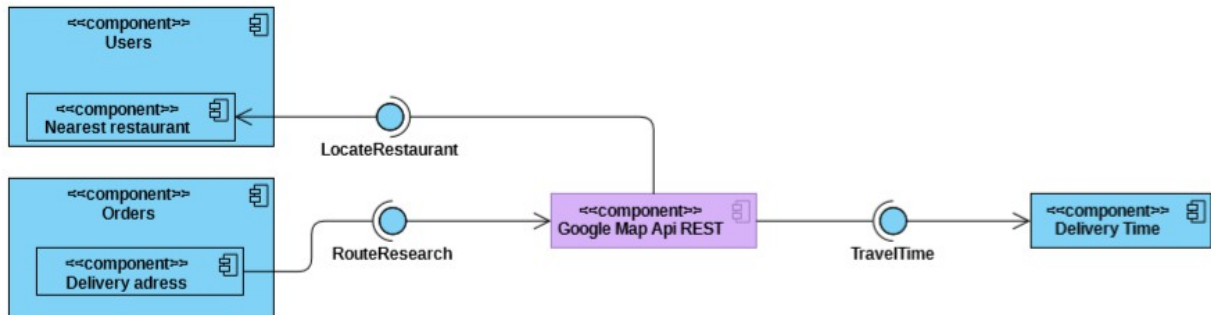
Analytics : Ce composant récupère les informations contenues dans le composant **Orders** afin de générer des données analytiques consultables par la direction d'OC Pizza, que ce soit les produits favoris des clients, les délais de préparation des commandes etc.

Google Map API REST : Ce composant extérieur gère les besoins de géolocalisation du système, nous l'aborderons un peu plus en détail par la suite.

Bank system : Ce composant extérieur gère la partie paiement en ligne du système, comme pour le précédent composant, nous l'aborderons plus en détail.

3.1.2 – API REST Google map

Ci-dessous, le diagramme de composant de l'API REST Google map :

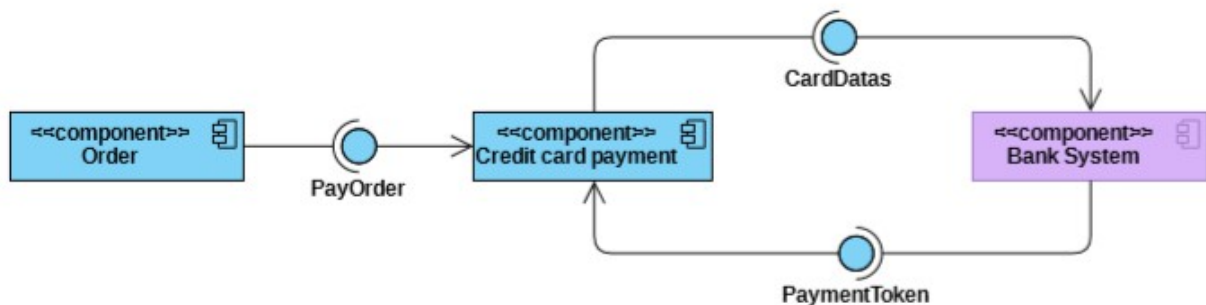


Voici brièvement les raisons de l'utilisation de ce composant extérieur :

- La possibilité selon l'adresse d'un utilisateur de localiser le restaurant le plus proche.
- Afficher l'itinéraire vers une adresse de livraison à un livreur.
- Déterminer le temps de trajet de la livraison afin d'indiquer un délai d'attente au client.

3.1.3 – Diagramme de composant système bancaire

Ci-dessous, le diagramme de composant du système bancaire :



Décrivons l'utilisation de ce composant avec un court exemple : un utilisateur décide de payer sa commande en ligne, cela implique l'utilisation d'une carte bancaire. Les données de cette carte sont envoyées au système bancaire qui procède au paiement puis renvoi un token de validation du paiement.

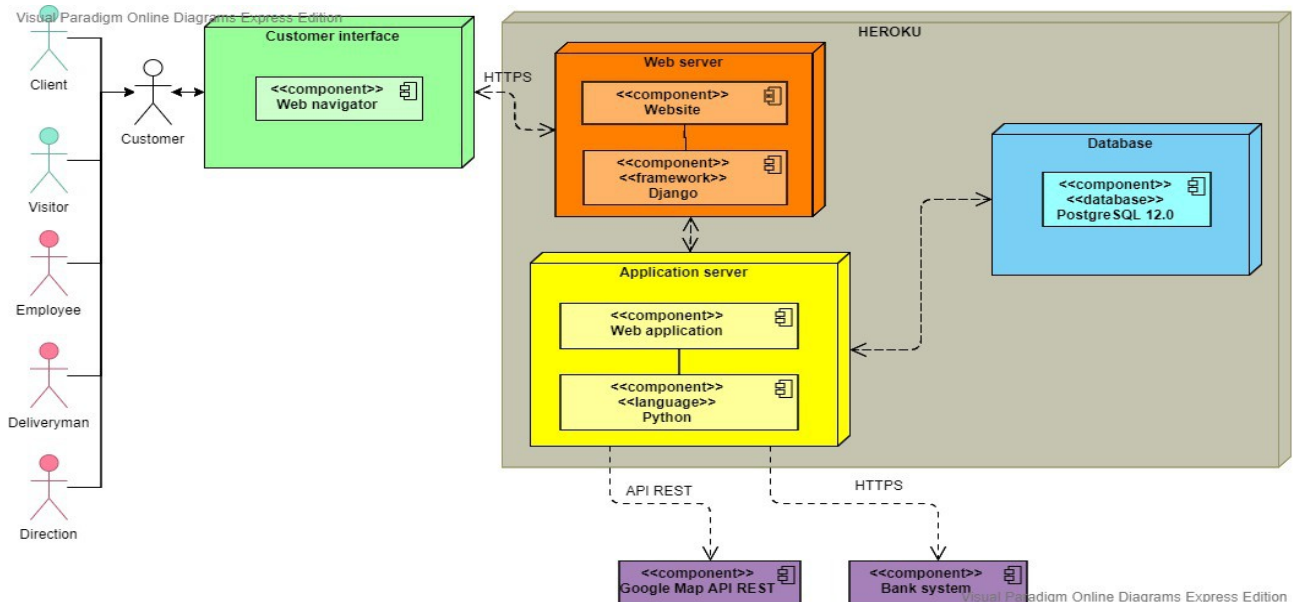
3.2 - Application Web

L'application sera développée en langage Python (version 3.8) avec l'utilisation du framework Django (version 3.1.1). Elle sera hébergée sur un serveur Heroku.

4 - ARCHITECTURE DE DÉPLOIEMENT

Nous allons utiliser un diagramme de déploiement afin d'identifier les éléments matériels nécessaire à notre solution, ainsi que leurs connexions entre eux. Chacun de ces éléments est appelé « nœud ».

Ci-dessous, le diagramme de déploiement :



Nous pouvons apercevoir quatre parties distinctes :

Customer interface : C'est le moyen d'accès au système d'un utilisateur, il peut s'agir de n'importe quoi du moment que le support permet d'accéder à un navigateur internet (**Web Navigator**). Ainsi les employés pourront utiliser des postes fixes ou des tablettes intégrés dans le restaurant afin d'accéder au système, les livreurs préféreront probablement utiliser un smartphone, les clients pourront utiliser n'importe lequel de ces moyens, etc.

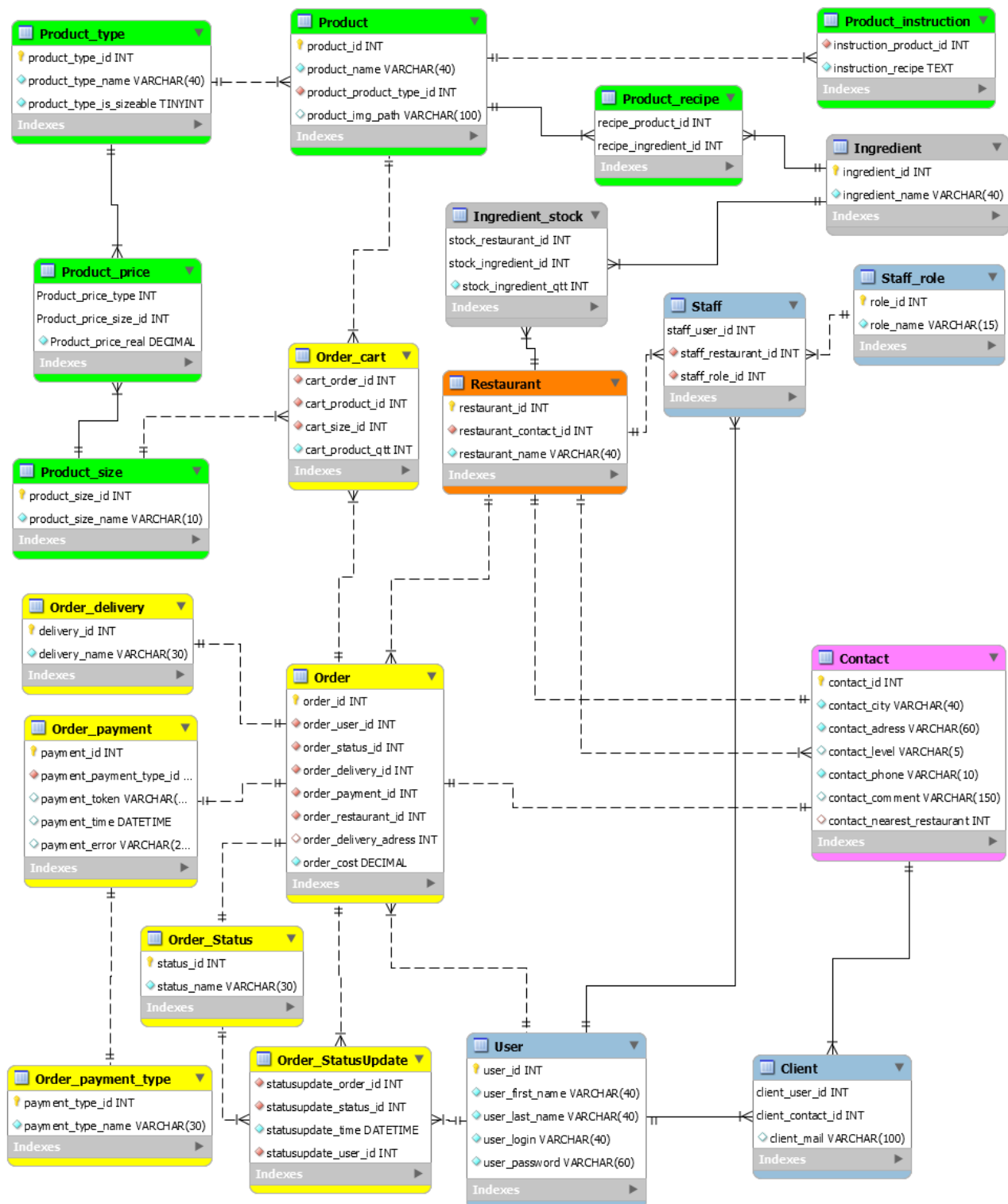
Web Server : C'est le serveur décentralisé sur lequel se trouve le site internet (**Website**) qui utilise le **framework Django**. Il permet de faire le lien entre les nœuds **Customer interface** et **Application server**.

Application server : C'est le serveur contenant l'application (**Web application**) qui fera fonctionner le système. Cette application sera développée en **Python**. Ce nœud est également relié aux composants extérieurs **Google Map API REST** et **Bank system** dont nous déjà vu les raisons d'utilisations.

Database : Pour terminer, c'est le serveur qui contient la base de donnée PostgreSQL. Ce nœud est directement relié au nœud **Application server**.

4.1 - Modèle physique de donnée

Ci-dessous, la représentation du modèle de donnée :



Description des tables

Restaurant



Cette table regroupe les informations communes des restaurants OCPizza.

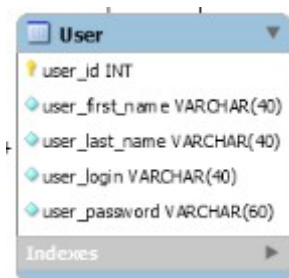
Toutes les colonnes doivent être renseignées (**NOT NULL**)

restaurant_id : clé primaire en **AUTO_INCREMENT** de type **INT**.

restaurant_name : nom du restaurant, type **VARCHAR(40)**

restaurant_contact_id : clé étrangère se référant à **contact_id** de la table **Contact**, qui permet de retrouver l'adresse du restaurant.

User



Cette table regroupe les informations communes des utilisateurs.

Toutes les colonnes doivent être renseignées (**NOT NULL**)

user_id : clé primaire en **AUTO_INCREMENT** de type **INT**

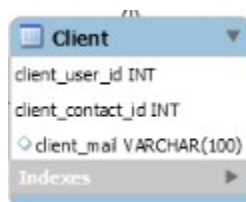
user_first_name : prénom de l'utilisateur, type **VARCHAR(40)**

user_last_name : nom de l'utilisateur, type **VARCHAR(40)**

user_login : login de l'utilisateur, type **VARCHAR(40)**

user_password : mot de passe de l'utilisateur, sera encrypté avec bcrypt (un module existant pour Python), type **VARCHAR(60)**

Client



Client	
client_user_id	INT
client_contact_id	INT
client_mail	VARCHAR(100)
Indexes	

Cette table fait le lien entre un utilisateur client, son adresse et son adresse mail.

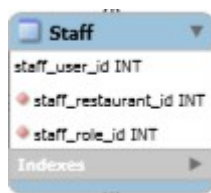
Toutes les colonnes doivent être renseignées (**NOT NULL**)

client_user_id : **clé étrangère** se référant à **user_id** de la table **User**, un utilisateur se retrouvant dans cette table sera donc considéré comme un client par le système.

client_contact_id : **clé étrangère** se référant à **contact_id** de la table **Contact**, le client ayant forcément une (voire plusieurs) adresse(s) pour recevoir une livraison.

client_mail : adresse mail du client, type **VARCHAR(100)**

Staff



Staff	
staff_user_id	INT
staff_restaurant_id	INT
staff_role_id	INT
Indexes	

Cette table fait le lien entre un utilisateur OCPizza, son rôle et son restaurant d'affiliation.

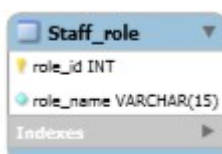
Toutes les colonnes doivent être renseignées (**NOT NULL**)

staff_user_id : **clé étrangère** se référant à **user_id** de la table **User**, un utilisateur inscrit dans cette table sera considéré comme une personne travaillant pour OC Pizza par le système.

staff_restaurant_id : **clé étrangère** se référant à **restaurant_id** de la table **Restaurant**, elle permet de déterminer dans quel restaurant travaille l'employé.

staff_role_id : **clé étrangère** se référant à **role_id** de la table **Staff_role**, permet de renseigner quel est le rôle de l'employé (livreur, pizzaiolo etc.)

Staff_role



Staff_role	
role_id	INT
role_name	VARCHAR(15)
Indexes	

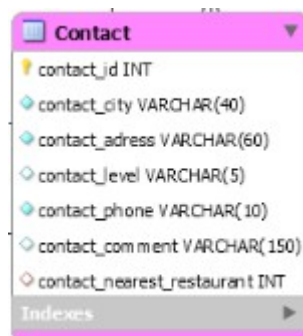
Cette table liste les rôles possibles pour les employés d'OCPizza.

Toutes les colonnes doivent être renseignées (**NOT NULL**)

role_id : **clé primaire** en **AUTO_INCREMENT** de type **INT**

role_name : nom du rôle (employé, livreur etc.), type **VARCHAR(15)**

Contact



Contact	
contact_id	INT
contact_city	VARCHAR(40)
contact_address	VARCHAR(60)
contact_level	VARCHAR(5)
contact_phone	VARCHAR(10)
contact_comment	VARCHAR(150)
contact_nearest_restaurant	INT
Indexes	

Cette table regroupe les informations communes concernant les coordonnées des clients et restaurants.

Mise à part les colonnes **contact_level** et **contact_comment**, toutes les colonnes doivent être renseignées (**NOT NULL**)

contact_id : clé primaire en **AUTO_INCREMENT** de type **INT**

contact_city : ville de l'entité, type **VARCHAR(40)**

contact_address : adresse de l'entité, type **VARCHAR(60)**

contact_level : étage de l'entité, type **VARCHAR(5)**

contact_phone : numéro de téléphone de l'entité, type **VARCHAR(10)**

contact_comment : commentaire de l'entité, type **VARCHAR(150)**

contact_nearest_restaurant : clé étrangère se référant à **restaurant_id** de la table **Restaurant**, cela sert à déterminer quel restaurant sera le plus proche de l'adresse d'un client (et ce afin qu'il récupère automatiquement la commande).

Ingredient



Ingredient	
ingredient_id	INT
ingredient_name	VARCHAR(40)
Indexes	

Cette table liste les ingrédients utilisés par OCPizza.

Toutes les colonnes doivent être renseignées (**NOT NULL**)

ingredient_id : clé primaire en **AUTO_INCREMENT** de type **INT**

ingredient_name : nom de l'ingrédient, type **VARCHAR(40)**

Ingredient_stock



Cette table fait le lien entre les ingrédients et les restaurants afin d'établir le stock de chaque enseigne OCPizza.

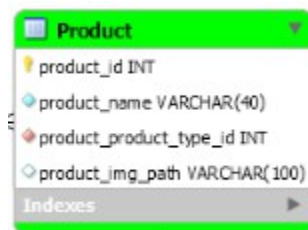
Toutes les colonnes doivent être renseignées (**NOT NULL**)

stock_restaurant_id : **clé étrangère** se référant à **restaurant_id** de la table **Restaurant**, permettant de savoir à quel restaurant appartient le stock.

stock_ingredient_id : **clé étrangère** se référant à **ingredient_id** de la table **Ingredient**, nous renseigne sur la nature de l'ingrédient en stock.

stock_ingredient_qtt : quantité de l'ingrédient en stock, type **INT**

Product



Cette table regroupe les informations communes des produits.

Mise à part la colonne **product_img_path**, toutes les colonnes doivent être renseignées (**NOT NULL**)

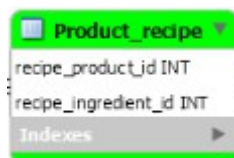
product_id : **clé primaire** en **AUTO_INCREMENT** de type **INT**

product_name : nom du produit, type **VARCHAR(40)**

product_product_type_id : **clé étrangère** se référant à **product_type_id** de la table **Product_type**, cela permet de savoir à quelle catégories appartient le produit, cela sera utile pour déterminer son prix.

product_img_path : chemin vers image/représentation du produit, type **VARCHAR(100)**

Product_recipe



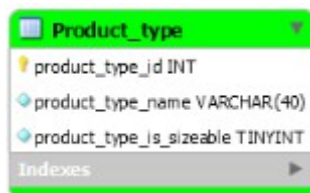
Cette table fait le lien entre les ingrédients et les produits, afin d'établir la recette d'un produit.

Toutes les colonnes doivent être renseignées (**NOT NULL**)

recipe_product_id : **clé étrangère** se référant à **product_id** de la table **Product** : le produit concerné.

recipe_ingredient_id : **clé étrangère** se référant à **ingredient_id** de la table **Ingredient** : les ingrédients présent dans le produit.

Product_type



Cette table liste les catégories possibles des produits OCPizza.

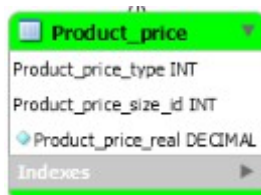
Toutes les colonnes doivent être renseignées (**NOT NULL**)

product_type_id : **clé primaire** en **AUTO_INCREMENT** de type **INT**

product_type_name : nom de la catégorie, type **VARCHAR(40)**

product_type_is_sizeable : détermine si un produit affilié à cette catégorie peut avoir un attribut de taille, type **TINYINT**

Product_price



Cette table fait le lien entre une catégorie de produit et une taille afin de déterminer un prix.

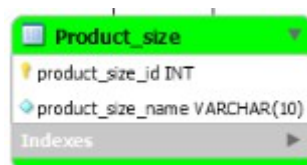
Toutes les colonnes doivent être renseignées (**NOT NULL**)

Product_price_type_id : **clé étrangère** se référant à **product_type_id** de la table **Product_type**, le premier élément pour déterminer le prix étant la catégorie d'un produit.

Product_price_size_id : **clé étrangère** se référant à **product_size_id** de la table **Product_size**, le second élément étant sa taille (si le produit peut avoir différents formats).

Product_price_real : prix d'un produit selon sa catégorie et sa taille, type **DECIMAL**

Product_size



Product_size	
product_size_id	INT
product_size_name	VARCHAR(10)
Indexes	

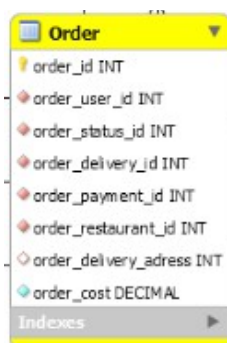
Cette table liste les tailles possibles pour les produits OCPizza

Toutes les colonnes doivent être renseignées (**NOT NULL**)

Product_size_id : clé primaire en **AUTO_INCREMENT** de type **INT**

Product_size_name : nom de la taille du produit, une entrée « none » existe pour les produits aux formats fixe, type **VARCHAR(40)**

Order



Order	
order_id	INT
order_user_id	INT
order_status_id	INT
order_delivery_id	INT
order_payment_id	INT
order_restaurant_id	INT
order_delivery_adress	INT
order_cost	DECIMAL
Indexes	

Cette table regroupe les informations communes des commandes.

Mise à part la colonne **order_delivery_adress**, toutes les colonnes doivent être renseignées (**NOT NULL**)

order_id : clé primaire en **AUTO_INCREMENT** de type **INT**

order_user_id : clé étrangère se référant à **user_id** de la table **User**, afin de savoir qui a passé la commande, cela peut être un client comme un employé.

order_status_id : clé étrangère se référant à **status_id** de la table **Order_status**, permet de connaître le statut actuel de la commande.

order_delivery_id : clé étrangère se référant à **delivery_id** de la table **Order_delivery**, renseigne le mode de livraison de la commande.

order_payment_id : clé étrangère se référant à **payment_id** de la table **Order_payment**, donne les informations sur le paiement de la commande.

order_restaurant_id : clé étrangère se référant à **restaurant_id** de la table **Restaurant**, permet de savoir quel restaurant s'occupe de la commande.

order_delivery_adress : clé étrangère se référant à **contact_id** de la table **Contact**, donne l'adresse de livraison si la commande doit être livrée.

order_cost : coût total de la commande au moment où elle a été passée, type **DECIMAL**

Order_cart



Order_cart	
cart_order_id	INT
cart_product_id	INT
cart_size_id	INT
cart_product_qtt	INT
Indexes	

Cette table fait le lien entre une commande, un produit et sa taille, et détermine la quantité de produit sélectionné par un utilisateur.

Toutes les colonnes doivent être renseignées (**NOT NULL**)

cart_order_id : **clé étrangère** se référant à **order_id** de la table **Order**, permet de savoir à quelle commande appartient le panier.

cart_product_id : **clé étrangère** se référant à **product_id** de la table **Product**, renseigne sur le (ou les) produit commandé.

cart_size_id : **clé étrangère** se référant à **product_size_id** de la table **Product_size**, indique le format du (ou des) produit commandé.

cart_product_qtt : quantité d'un produit dans le panier, type **INT**

Order_delivery



Order_delivery	
delivery_id	INT
delivery_name	VARCHAR(30)
Indexes	

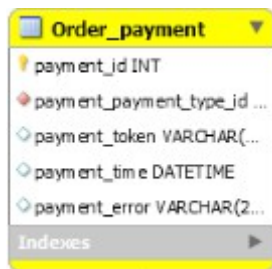
Cette table liste tout les mode de livraison possibles.

Toutes les colonnes doivent être renseignées (**NOT NULL**)

delivery_id : **clé primaire** en AUTO_INCREMENT de type **INT**

delivery_name : nom du mode de livraison, type **VARCHAR(30)**

Order_payment



Order_payment
payment_id INT
payment_payment_type_id ...
payment_token VARCHAR(...)
payment_time DATETIME
payment_error VARCHAR(200)
Indexes

Cette table rassemble les informations sur le paiement d'une commande.

Seules les colonnes **payment_id** et **payment_type_id** doivent être renseignées (**NOT NULL**)

payment_id : clé primaire en **AUTO_INCREMENT** de type **INT**

payment_payment_type_id : clé étrangère se référant à **payment_type_id** de la table **Payment_type**, permet de savoir à quel mode de paiement a été choisi pour la commande.

payment_token : Token de paiement renvoyé par le système bancaire, encrypté, type **VARCHAR(16)**

payment_time : heure et date du paiement de la commande (dès le moment où ce champ est rempli, la commande est considérée comme payée), type **DATETIME**

payment_error : si une erreur se produit au moment du paiement, le message sera stocké ici, type **VARCHAR(200)**

Order_status



Order_Status
status_id INT
status_name VARCHAR(30)
Indexes

Cette table liste tout les statuts possibles.

Toutes les colonnes doivent être renseignées (**NOT NULL**)

status_id : clé primaire en **AUTO_INCREMENT** de type **INT**

status_name : nom du mode de paiement, type **VARCHAR(30)**

Order_StatusUpdate



Order_StatusUpdate	
statusupdate_order_id	INT
statusupdate_status_id	INT
statusupdate_time	DATETIME
statusupdate_user_id	INT
Indexes	

Cette table fait le lien entre une commande, un statut et un utilisateur afin de savoir qui a modifié le statut d'une commande, et quand.

Toutes les colonnes doivent être renseignées (**NOT NULL**)

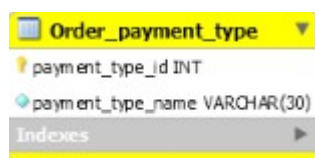
statusupdate_order_id : **clé étrangère** se référant à **order_id** de la table **Order**

statusupdate_status_id : **clé étrangère** se référant à **status_id** de la table **Order_status**

statusupdate_time : heure et date de la mise à jour du statut d'une commande, type **DATETIME**

statusupdate_user_id : **clé étrangère** se référant à **user_id** de la table **User**

Order_payment_type



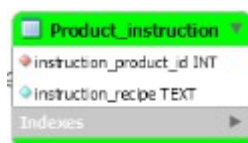
Order_payment_type	
payment_type_id	INT
payment_type_name	VARCHAR(30)
Indexes	

Cette table liste les modes de paiement disponibles.

payment_id : **clé primaire** en **AUTO_INCREMENT** de type **INT**

payment_name : nom du mode de paiement, type **VARCHAR(25)**

Product_instruction



Product_instruction	
instruction_product_id	INT
instruction_recipe	TEXT
Indexes	

Cette table contient les instructions des recettes des produits OC Pizza

instruction_product_id : **clé étrangère se référant à product_id de la table Product**

instruction_recipe : instruction de la recette du produit, type **TEXT**

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

OCPizzaapp sera développé via le framework Django. Un pattern model view template sera utilisé.

5.1.1 - Les couches

L'architecture applicative est la suivante :

- une couche **business** : responsable de la logique métier du composant
- une couche **model** : implémentation du modèle des objets métiers
- une couche **view** : responsable de l'interface de l'application

5.1.2 - Les modules

Deux modules seront créés :

- **webapp** : Gèrera l'interface des clients et celle des employés d'OC Pizza
- **userapp** : Gèrera toute la partie authentification de l'application.

5.1.3 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

```
Racine
├── manage.py
├── requirements.txt
├── .gitignore
├── README.md
├── OCPizzapp_project
│   ├── __init__.py
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── webapp
│   ├── static
│   │   ├── webapp
│   │   │   ├── css
│   │   │   │   ├── index.css
│   │   │   │   └── ...
│   │   │   ├── js
│   │   │   │   ├── script.js
│   │   │   │   └── ...
│   │   └── img
│   └── template
│       ├── webapp
│       │   ├── index.html
│       │   └── ...
│       └── 404.html
```

```

    |
    |   L ...
    |
    |   ├── __init__.py
    |   ├── admin.py
    |   ├── apps.py
    |   ├── forms.py
    |   ├── models.py
    |   ├── urls.py
    |   └── views.py
    |
    └─ userapp
        ├── static
        |   └─ userapp
        |       ├── css
        |       |   └─ ...
        |       ├── js
        |       |   └─ ...
        |       └─ img
        ├── template
        |   └─ userapp
        |       └─ ...
        ├── __init__.py
        ├── admin.py
        ├── apps.py
        ├── forms.py
        ├── models.py
        ├── urls.py
        └── views.py

```

6 - POINTS PARTICULIERS

6.1 - Ressources

Les ressources graphiques utilisées par l'application seront fournies par OC Pizza.

Les données à implémenter dans la base de données seront également fournies par OC Pizza.

6.2 - Environnement de développement

Nous préconisons l'utilisation de Visual Studio Code sous environnement virtuel Virtualenv.

6.3 - Procédure de packaging / livraison

L'application sera déployée sur son serveur d'hébergement Heroku au moment de la livraison finale.

Un dossier d'exploitation de l'application sera également remis.

7 - GLOSSAIRE
