



DOSSIER DES SPECIFICATIONS TECHNIQUES



IT CONSULTING & DEVELOPMENT

SOMMAIRE

I - LE CONTEXTE

II – DOMAINE FONCTIONNEL

- 1 – Diagramme de classe
- 2 – Description des classes
 - a – Restaurant
 - b – User
 - c – Client
 - d – OCPizza
 - e – Contact
 - f – Ingredient
 - g – Stock
 - h - Recipe
 - i – Product
 - j – Order
 - k - Basket
 - l - Order_statusupdate

III – DIAGRAMMES

- 1 – Diagramme de classe
- 2 – Description des classes
 - a – Restaurant
 - b – User
 - c – Client
 - d – OCPizza
 - e – OCPizza_role
 - f – Contact
 - g – Ingredient
 - h - Ingredient_stock
 - i – Product
 - j – Product_recipe
 - k - Product_type
 - l – Product_price
 - m – Product_size
 - n – Order
 - o – Order_basket
 - p – Order_delivery
 - q – Order_payment
 - r – Order_status
 - s – Order_statusupdate
 - t – Order_payment_type

IV – DIAGRAMME DE COMPOSANTS

- 1 – Général
- 2 – API REST Google map
- 3 – Système bancaire

V – DIAGRAMME DE DEPLOIEMENT

I – CONTEXTE

La société :

OC Pizza est une société spécialisée dans la vente de pizza en livraison et à emporter. Elle compte actuellement cinq points de vente et trois supplémentaires sont prévus.

L'objectif :

L'objectif est de mettre en place un système informatique pour l'ensemble des pizzerias du groupe

Fonctionnalités demandées :

- Être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation
- Suivre en temps réel les commandes passées, en préparation et en livraison
- Suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas peuvent encore être réalisées
- 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.
- Proposer un aide-mémoire aux pizzaiolos indiquant la recette de chaque pizza

b – User



Cette classe possède les attributs communs à chaque utilisateurs du programme (à savoir les clients ou les employés d'OCPizza). Ses attributs sont :

User_firstname : le prénom de l'utilisateur

User_lastname : le nom de l'utilisateur

User_login : le login d'authentification de l'utilisateur

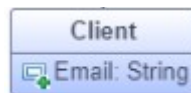
User_password : le mot de passe de l'utilisateur

Les classes **Client** et **OCPizza** lui sont associées par **héritage**, la complétant et permettant d'ajouter des attributs non-communs à ces deux classes.

La classe est associée à la classe **Order**.

Order : c'est une association **one-to-many**, un utilisateur pouvant interagir avec une ou plusieurs commande (que ce soit passer une commande, modifier le statut d'une commande etc.)

c – Client

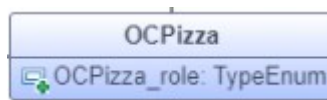


Elle ne possède qu'un attribut (**Email**) qui correspond à l'adresse mail du client.

Cette classe est associée par **héritage** à la classe **User**, qu'elle vient compléter. Elle est également associée avec la classe **Contact**.

Contact : c'est une association **one-to-many**, un client pouvant avoir une ou plusieurs adresses enregistrées (par exemple, si le client souhaite se faire livrer à une autre adresse que celle de son domicile).

d – OCPizza



Elle ne possède qu'un attribut (**Role**) qui correspond à l'affectation d'un employé OCPizza.

Cette classe est associée par **héritage** à la classe **User**, qu'elle vient compléter. Elle est également associée avec la classe **Restaurant** dont elle est un **composant**.

Restaurant : l'association a déjà été commentée précédemment.

e – Contact



Cette classe possède les attributs suivant :

City : nom de ville de l'entité

Address : adresse de l'entité

Level : Si appartement, étage de l'entité

Phone : Numéro de téléphone de l'entité

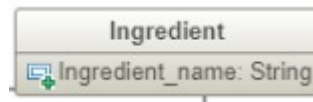
Comment : Commentaires de l'entité

La classe est associée aux classes **Restaurant**, **User** et **Order**.

Les deux premières associations ont été commentées précédemment.

Order : c'est une association **one-to-one**, dans le cas où la commande doit être livrée, celle-ci ne peut être livrée qu'à une seule adresse.

f – Ingredient



Elle ne possède qu'un attribut (**Ingredient_name**) qui correspond au nom de l'ingrédient.

La classe est associée aux classes **Restaurant** et **Product**.

Restaurant : l'association a déjà été commentée précédemment..

Product : c'est une association **many-to-many**, un (ou plusieurs) produit pouvant contenir un certain nombre d'ingrédient dans sa recette.

g – Stock



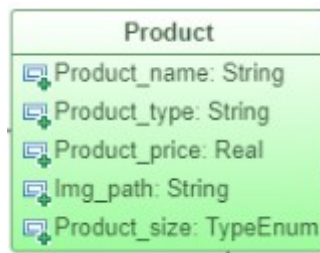
C'est une **classe d'association** qui permet d'ajouter l'attribut **Ingredient_quantity** à l'association entre les classes **Restaurant** et **Ingredient** (**many-to-many**). Cela permet de connaître la quantité d'un ingrédient dans le stock d'un restaurant.

h – Recipe



C'est une **classe d'association** entre les classes **Ingredient** et **Product** (**many-to-many**). Cela permet de connaître les ingrédients nécessaire à la préparation d'un ou plusieurs produits, en bref, la recette.

i – Product



Cette classe possède les attributs suivant :

Product_name : le nom du produit

Product_type : la catégorie du produit (pizza, boisson etc.)

Product_price : la valeur du produit

Product_size : la taille du produit

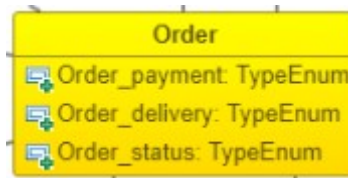
Image_path : un chemin vers une image/représentation du produit

La classe est associée aux classes **Ingredient** et **Order**.

Ingredient : l'association a déjà été commentée précédemment

Order : c'est une association **many-to-many**, un (ou plusieurs) produit pouvant être contenu dans une (ou plusieurs) commande.

j – Order



Cette classe possède les attributs suivant :

Order_payment : le mode de paiement de la commande

Order_delivery : le mode de livraison de la commande

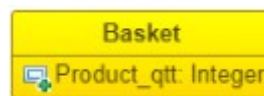
Order_status : le statut de la commande (en préparation, en livraison etc.)

La classe est associée aux classes **Restaurant**, **User**, **Contact**, **Product** et **Order_statusupdate**

Les quatre premières associations ont été commentées précédemment.

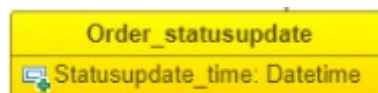
Order_statusupdate : c'est une association **one-to-many**, une commande ayant pu avoir une (ou plusieurs) mise à jour de statut.

k – Basket



C'est une **classe d'association** qui permet d'ajouter l'attribut **Product_qtt** à l'association entre les classes **Product** et **Order** (**many-to-many**). Cela permet de connaître la quantité d'un produit dans une commande.

l – Order_statusupdate



Elle ne possède qu'un attribut (**Statusupdate_time**) qui correspond à la date et l'heure de mise à jour de statut d'une commande.

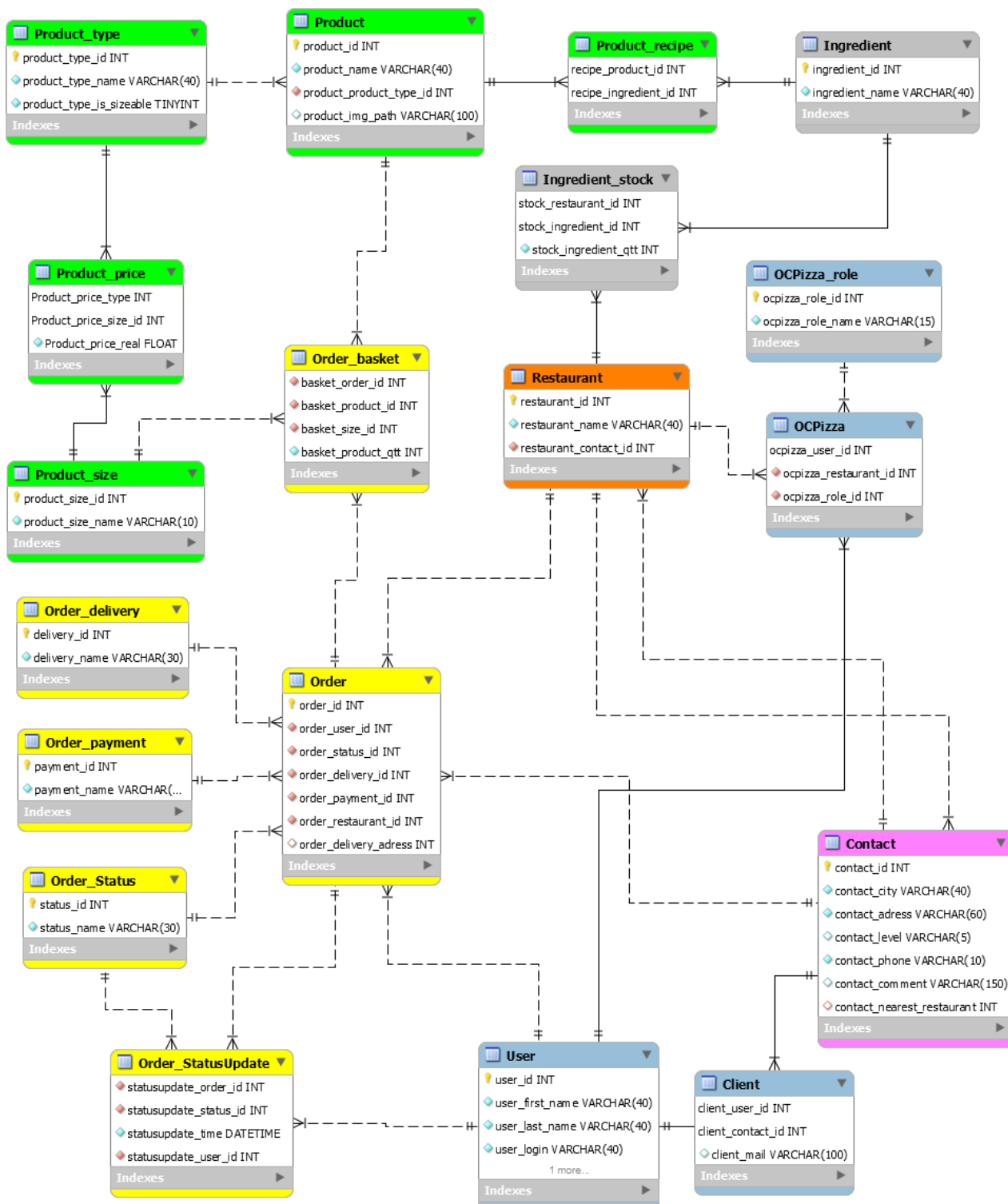
La classe est associée à la classe **Order**.

Order : l'association a déjà été commentée précédemment.

III – MODELE DE DONNEES

1 – Modèle de donnée

Ci-dessous, la représentation du modèle de donnée :



2 – Description des tables

a – 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.

b – 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**

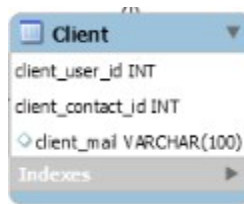
user_first_name : prénom de l'utilisateur, type **VARCHAR(40)**

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)**

c – 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.

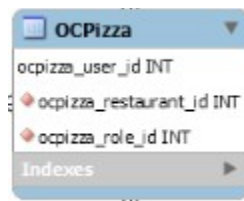
Toutes les colonnes doivent être renseignées (**NOT NULL**)

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)**

d – Staff



OCPizza	
ocpizza_user_id	INT
ocpizza_restaurant_id	INT
ocpizza_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.)

e – Staff_role



OCPizza_role	
ocpizza_role_id	INT
ocpizza_role_name	VARCHAR(15)
Indexes	

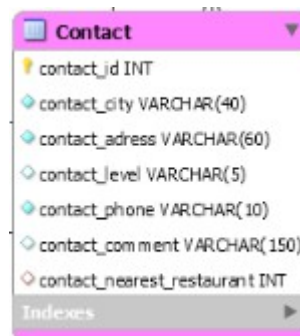
Cette table liste les rôles possibles pour les employés d'OCPizza.

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)**

f – Contact



Contact	
contact_id	INT
contact_city	VARCHAR(40)
contact_adress	VARCHAR(60)
contact_level	VARCHAR(5)
contact_phone	VARCHAR(10)
contact_com ment	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_adress : 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).

g – 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)**

h – Ingredient_stock



Ingredient_stock	
stock_restaurant_id	INT
stock_ingredient_id	INT
stock_ingredient_qtt	INT
Indexes	

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

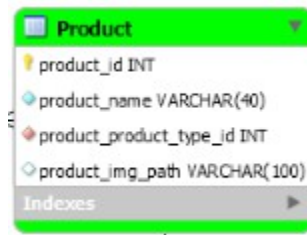
Toutes les colonnes doivent être renseignées (**NOT NULL**)

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**

i – Product



Product	
product_id	INT
product_name	VARCHAR(40)
product_product_type_id	INT
product_img_path	VARCHAR(100)
Indexes	

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**

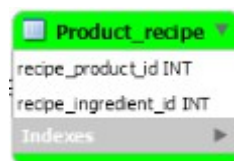
product_name : nom du produit, type **VARCHAR(40)**

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)**

j – Product_recipe



Product_recipe	
recipe_product_id	INT
recipe_ingredient_id	INT
Indexes	

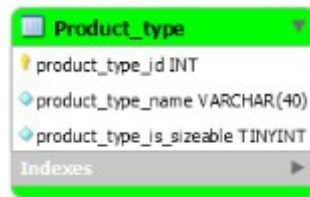
Cette table fait le lien entre les ingrédients et les produits, afin d'établir la recette d'un produit.

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.

k – Product_type



Product_type	
product_type_id	INT
product_type_name	VARCHAR(40)
product_type_is_sizeable	TINYINT
Indexes	

Cette table liste les catégories possibles des produits OCPizza.

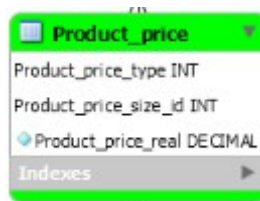
Toutes les colonnes doivent être renseignées (**NOT NULL**)

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**

l – Product_price



Product_price	
Product_price_type_id	INT
Product_price_size_id	INT
Product_price_real	DECIMAL
Indexes	

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

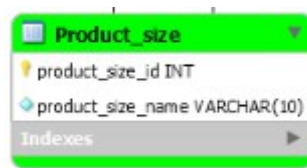
Toutes les colonnes doivent être renseignées (**NOT NULL**)

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**

m – 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)**

n – 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**

o – Order_basket



Order_basket	
•	basket_order_id INT
•	basket_product_id INT
•	basket_size_id INT
•	basket_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**)

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

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

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

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

p – Order_delivery



Order_delivery	
•	delivery_id INT
•	delivery_name VARCHAR(30)
Indexes	

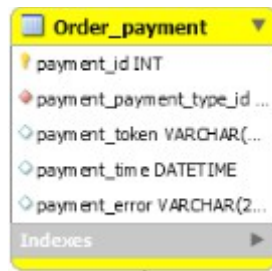
Cette table liste tout les mode de livraison possibles.

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)**

q – 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)

r – Order_status



Order_Status	
status_id	INT
status_name	VARCHAR(30)
Indexes	

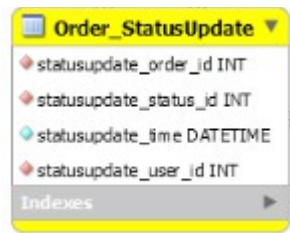
Cette table liste tout les statuts possibles.

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)

s – 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**)

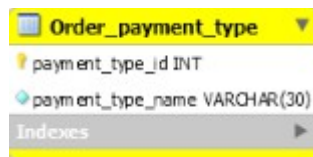
statusupdate_order_id : clé étrangère se référant à **order_id** de la table **Order**

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**

t – Order_payment_type



Order_payment_type	
payment_type_id	INT
payment_type_name	VARCHAR(30)
Indexes	

Cette table liste les modes de paiement disponibles.

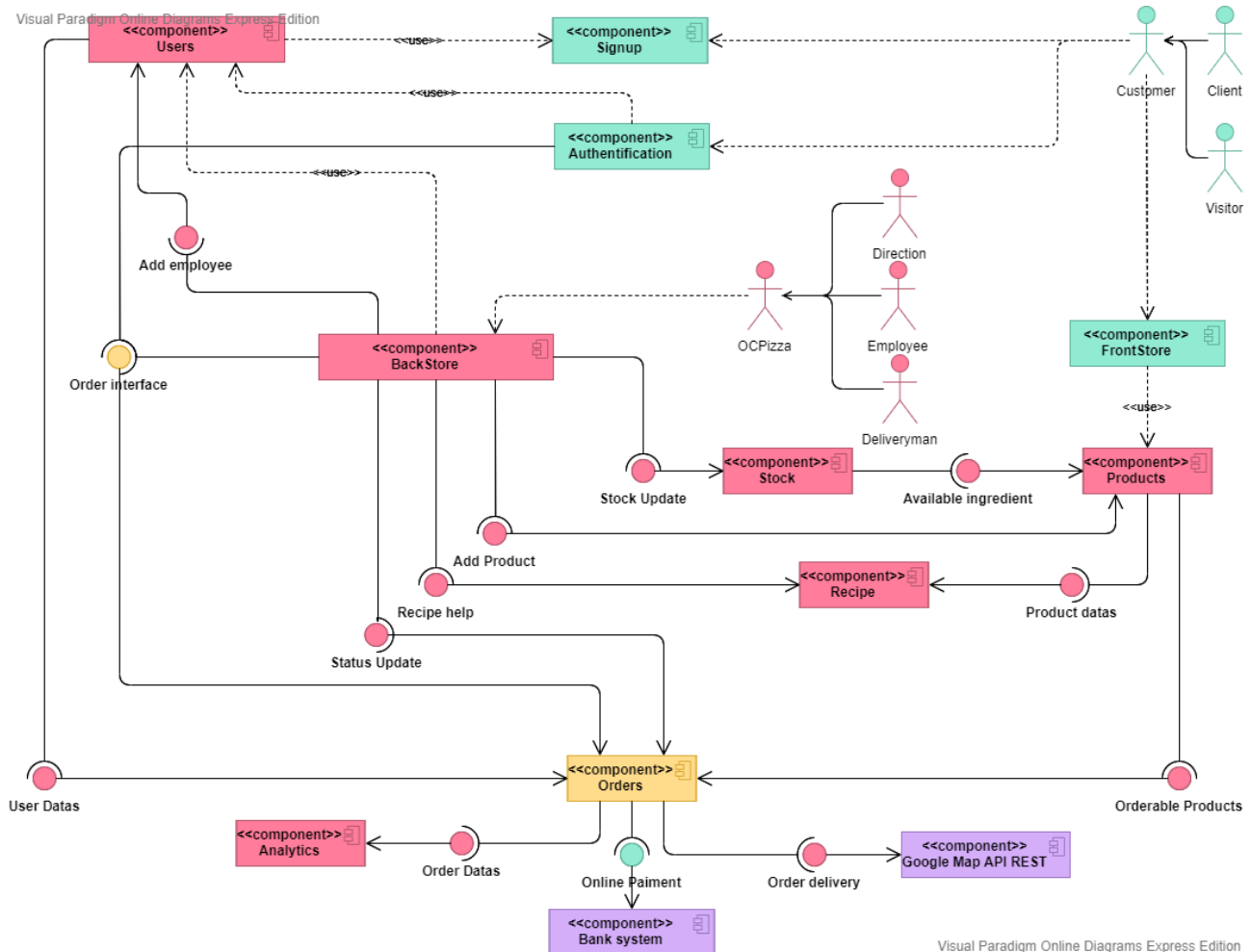
payment_id : clé primaire en **AUTO_INCREMENT** de type **INT**

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

IV – DIAGRAMME DE 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.

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'interface de commande permettant, comme son nom l'indique, de passer une commande,

ou d'interagir avec.

FrontStore : Ce composant correspond plus ou moins au site web que la clientèle pourra consulter. Il utilise le composant **Product** afin d'afficher aux clients quels produits sont disponibles à la commande, et les informations sur ces derniers (prix, ingrédients etc.)

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 **FrontStore** 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 envoie 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**.

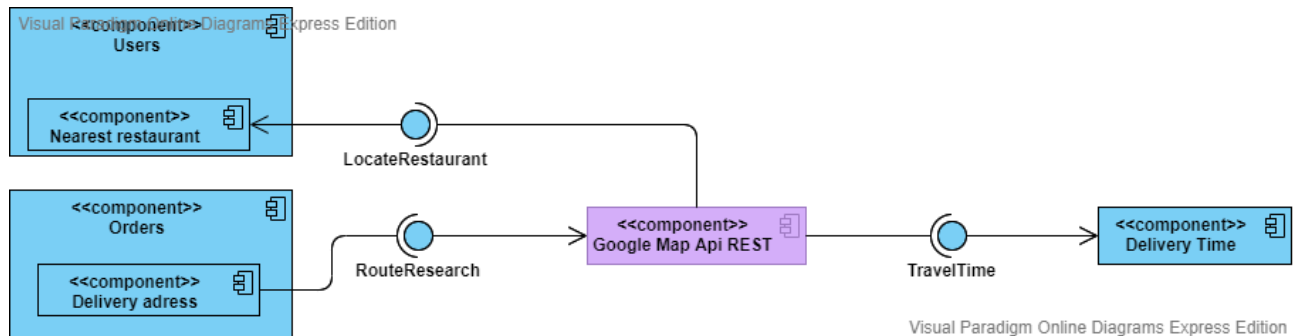
Analytics : Ce composant récupère les informations contenues dans le composant **Orders** afin de générer des données analytiques consultables par la direction d'OC Pizza, que ce soit les produits favoris des clients, les délais de préparation des commandes etc.

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.

2 – API REST Google map

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

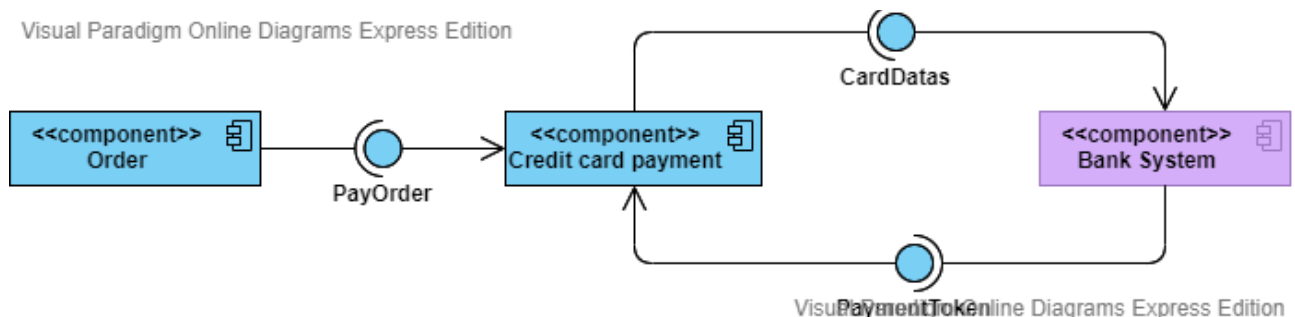


Voici brièvement les raisons de l'utilisation de ce composant extérieur :

- 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 – Diagramme de composant système bancaire

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

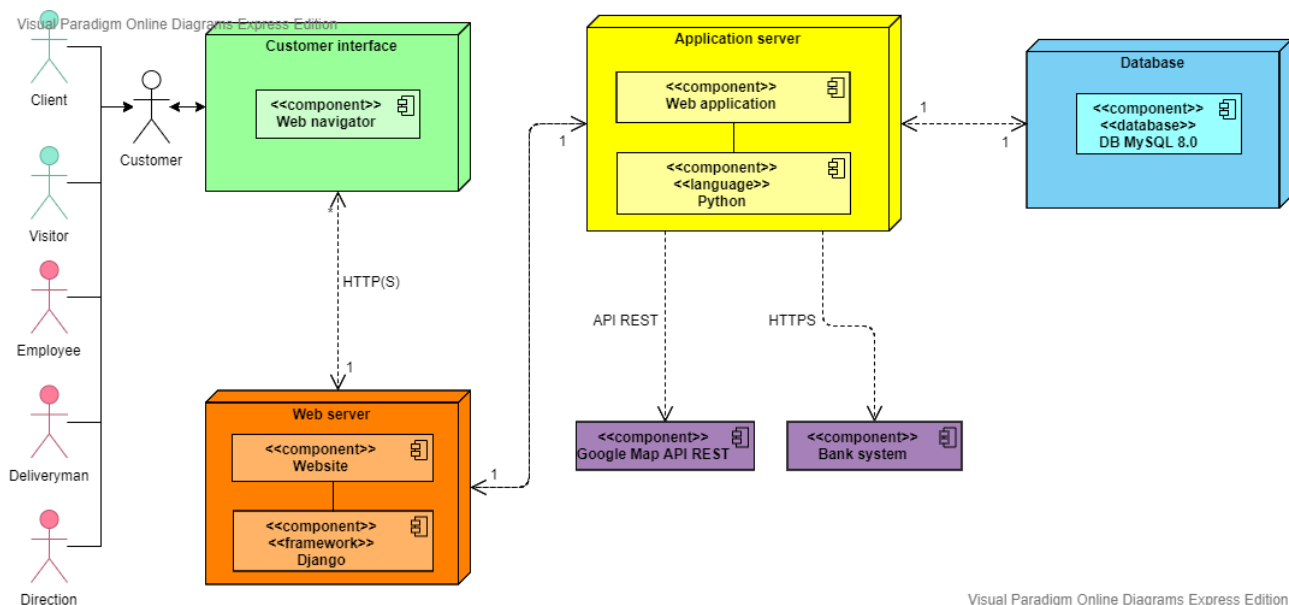


Décrivons l'utilisation de ce composant avec un court exemple : un utilisateur décide de payer sa commande en ligne, cela implique l'utilisation d'une carte bancaire. Les données de cette carte sont envoyées au système bancaire qui procède au paiement puis renvoi un token de validation du paiement.

V – DIAGRAMME DE DEPLOIEMENT

Après nos diagrammes de composants, 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 « noeud ».

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 MySQL (**DB MySQL 8.0**). Ce nœud est directement relié au nœud **Application server**.