OC Pizza

Projet 9

Dossier de conception technique

Version 1.0

Auteur

yohan solon

Développeur

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Yohan solon	01/05/2018	Création du document	1.0

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application OC Pizza Objectif du document détaillé la structure interne des composants utilisé dans l'application web.

L'application devra fournir aux clients la possibilité de consulter l'ensemble des produits que sont destinés à la vente en ligne, celui-ci pourra constituer sa commande, soit par une action en ligne avec un système de panier ou par consultation avec une prise de commande par téléphone.

La gestion en temps réel des stocks et leur suivi permettra de faire une analyse précise de la consommation quotidienne.

Il sera mis à disposition des outils permettant d'optimiser les livraisons ce qui a pour effet d'augmenter l'efficacité de gestion du circuit de commande.

2.2 - Références

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

DCF - projet 9: Dossier de conception fonctionnelle de l' application

Diagramme de classe

Modèle logique de données

3 - Architecture Technique

3.1 - Composants généraux

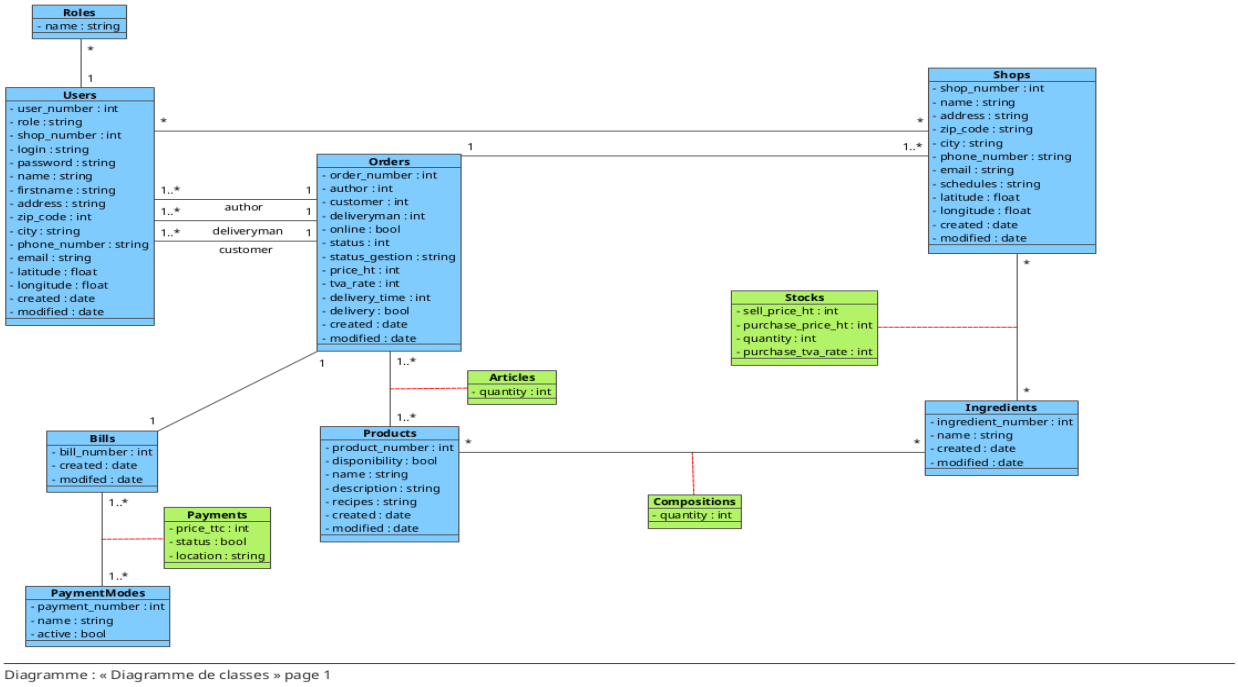


Diagramme : « Diagramme de classes » page 1

3.2 - Composant et interactions

- Association Roles / Users :

Définitions des classes :

- Roles : Représente le rôle qu'un utilisateur peut avoir
- Users : Représente un utilisateur de l'application

Interprétation de la relation :

- Un rôle peut être assigné à un ou plusieurs utilisateur(s)
- Un utilisateur a un rôle

Table de correspondance :

Users	
Id [PK]	name
1	Jean
2	Elise
3	Pierre

Roles
name [PK]
Client
Employé

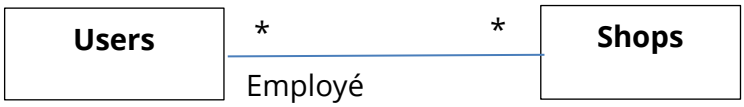
Tuples des tables “users” et “roles”

Pour définir quel rôle possède chaque utilisateur nous ajoutons un nouvel attribut “role_name” en tant que clé étrangère FK dans la table Users et on y ajoute la valeur de la clé primaire du rôle correspondant

Users		
Id [PK]	name	role_name [FK]
1	Jean	Client
2	Elise	Employé
3	Pierre	Employé

Ajout de la clé étrangère “role_name” dans la table Users

- Association Users / Shops, relation plusieurs-à-plusieurs



Définitions des classes :

- Shops : Représente un magasin
- Users : Représente un utilisateur “employé” de l’application

Interprétation de la relation :

- Un magasin peut avoir plusieurs employé-e(s)
- Un / des employé-e(s) peut être assigné-e(s) dans plusieurs magasins

Table de correspondance :

Users	
Id [PK]	name
1	Jean
2	Elise
3	Pierre

Shops	
Id [PK]	name
1	Mag_1
2	Mag_2

Tuples des tables “users” et “shops”

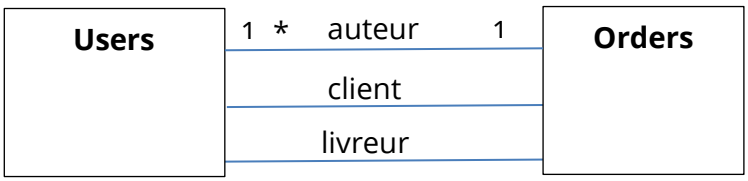
Pour répondre à la multiplicité des informations, nous mettrons en place une table qui matérialisera les différentes liaisons, afin de définir les employées d’un magasin.

La clé primaire de de cette table sera composé de clé étrangère point sur les tables users et shops.

Employees	
user_id [PFK]	shop_id [PFK]
1	1
2	2
3	2
3	1

Ajout d'une table employees matérialisant la relation « plusieurs à plusieurs » entre les tables users et shops

- Association Users / Orders, relations un-à-plusieurs



Définitions des classes :

- Orders : Représente une commande
- Users : Représente un utilisateur de l'application

Interprétation des relations :

- La première association indique l'auteur de la commande
- La deuxième association le client de la commande
- La troisième association l'employé en charge de donner la livraison

Table de correspondance :

Users	
Id [PK]	name
1	Jean
2	Elise
3	Pierre

Orders
Id [PK]
1
2

Tuples des tables "users" et "orders"

Pour définir dans quelle association est un utilisateur nous ajoutons de nouveaux attributs "author","customer","deliveryman" en tant que clé étrangère FK dans la table Orders et on y ajoute la valeur de la clé primaire de l'utilisateur correspondant

Orders			
Id [PK]	author [FK]	customer [FK]	deliveryman [FK]
1	2	1	3
2	2	..	..

Ajout des clefs étrangères dans la table Orders

- Association Orders / Shops relation un-à-plusieurs

Définitions des classes :

- Orders : Représente une commande
- Shops : Représente un magasin

Interprétation des relations :

- Une commande est assignée à un magasin
- Un magasin peut avoir 0 ou plusieurs commandes

Table de correspondance :

Orders
Id [PK]
1
2

Shops	
Id [PK]	name
1	Mag_1
2	Mag_2

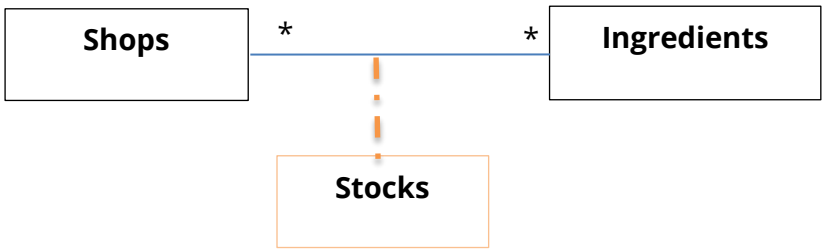
Tuples des tables “orders” et “shops”

Pour définir dans quel magasin est associer une commande nous ajoutons un nouvel attribut “shop_id” en tant que clé étrangère FK dans la table Orders et on y ajoute la valeur de la clé primaire du magasin correspondant.

Orders	
Id [PK]	shop_id [FK]
1	2
2	2

Ajout de la clef étrangère dans la table Orders

- Associations Shops / Ingrédients relation plusieurs-à-plusieurs



Définitions des classes :

- Shops : Représente un magasin
- Ingredients : Représente un ingrédient
- Classe d'association stocks

Interprétation de la relation :

- Un magasin peut contenir un stock d'ingrédients
- Un ingrédient peut être dans plusieurs stocks

Shops	
Id [PK]	name
1	Mag_1
2	Mag_2

Ingrédients	
Id [PK]	name
1	Ingredient 00
2	Ingredient 01

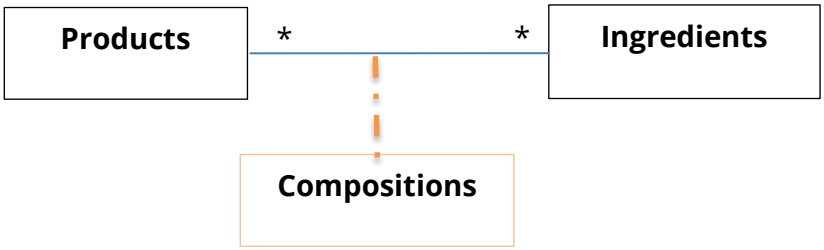
Tuples des tables "shops" et "ingredients"

Pour définir l'association de nos 2 tables la table stocks est constitué d'une clef primaire composé de deux clefs étrangères des valeurs des clefs primaires des table Shops et Ingredients et indique ça quantité.

Stocks		
shop_Id [PFK]	Ingredient_id [PFK]	quantity
1	1	10
2	1	50

Classe d'association stocks sur une base de d'une table de relation plusieurs-à-plusieurs

- Association Products / Ingredients relation plusieurs-à-plusieurs



Définitions des classes :

- Products : Représente un produit
- Ingredients : Représente un ingrédient

Interprétation des relations :

- Un produit est composé de plusieurs ingrédients
- Un ingrédient peut composer différents produits
- La classe d'association compositions contient les quantités d'ingrédients pour un produit

Products	
Id [PK]	name
1	product_1
2	product_2

Ingrédients	
Id [PK]	name
1	Ingredient 00
2	Ingredient 01

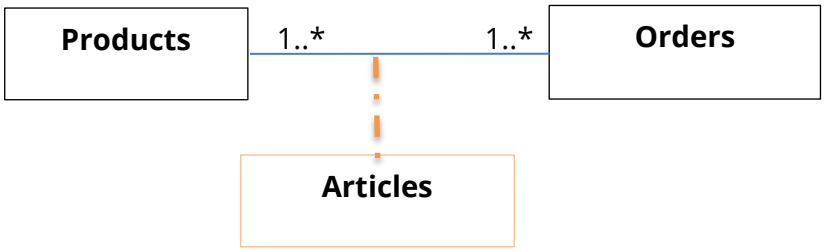
Tuples des tables "products" et "ingredients"

La classe d'association compositions est constitué d'une clef primaire composé des clefs étrangère des tables products et ingrédients ayant comme valeur leurs clefs primaires et la quantité d'ingrédients par association.

Compositions		
product_Id [PFK]	Ingredient_id [PFK]	quantity
1	1	4
2	2	5

Classe d'association compositions

- Association Products / Orders, relation plusieurs-à-plusieurs



Définitions des classes :

- Products : Représente un produit
- Orders : Représente une commande

Interprétation des relations :

- Une commande est composée de 1 à plusieurs articles
- Un produit peut composer 1 à commandes
- La classe d'association articles contient les quantités de produit que compose une commande

Products	
Id [PK]	name
1	product_1
2	product_2

Orders
Id [PK]
1
2

Tuples des tables "products" et "orders"

La classe d'association articles est chargé de faire la jonction entre les tables products et orders, constituant ainsi la liste de la quantité d'ingrédients nécessaire dans la constitution d'un produit.

Articles		
product_Id [PFK]	order_id [PFK]	quantity
1	1	4
2	1	5

Classe d'association articles

Contient la clef primaire composé des clefs étrangère product_id et order_id, ayant la valeur des clefs primaire de la table products et orders.

- Associations Orders / Bills, relation un-à-un

Définitions des classes :

- Orders : Représente une commande
- Bills : Représente une facture

Interprétation des relations :

- Une commande à une facture
- Une facture correspond à une commande

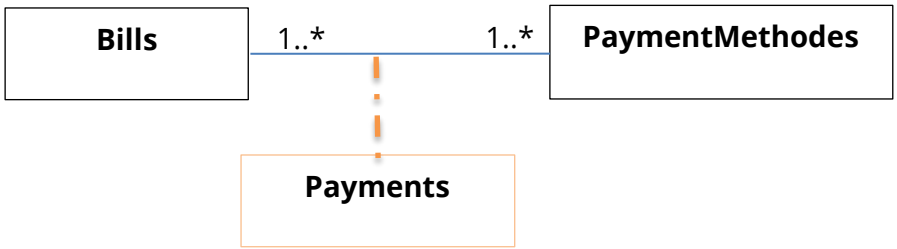
Tuples des tables “orders” et “Bills”

Pour savoir à quelle commande correspond une facture nous allons ajouter une clef étrangère dans la table Bills qui prendra comme valeur la clef primaire de la table orders.

Bills	
number [PK]	order_id [FK]
1	1
2	2

Ajout de la clef étrangère dans la table Bills

- Association Bills / PaymentMethodes, relation plusieurs-à-plusieurs6,3



Définitions des classes :

- Bills : Représente une facture
- PaymentMethodes : Représente les moyens de paiements

Interprétation des relations :

- Une facture a au minimum 1 à plusieurs modes de paiement
- Une méthode de paiement peut apparaître dans 1 à plusieurs factures
- La classe d'association payments contient les différents paiements effectués pour une facture

Bills	
number [PK]	order_id [FK]
1	1
2	2

PaymentMethodes
id [PK]
1
2

Tuples des tables "bills" et "PaymentMethodes"

La classe d'association payments est chargé de faire la jonction entre les tables bills et paymentMethodes, constituant ainsi la liste des moyens de paiement utilisés par le client.

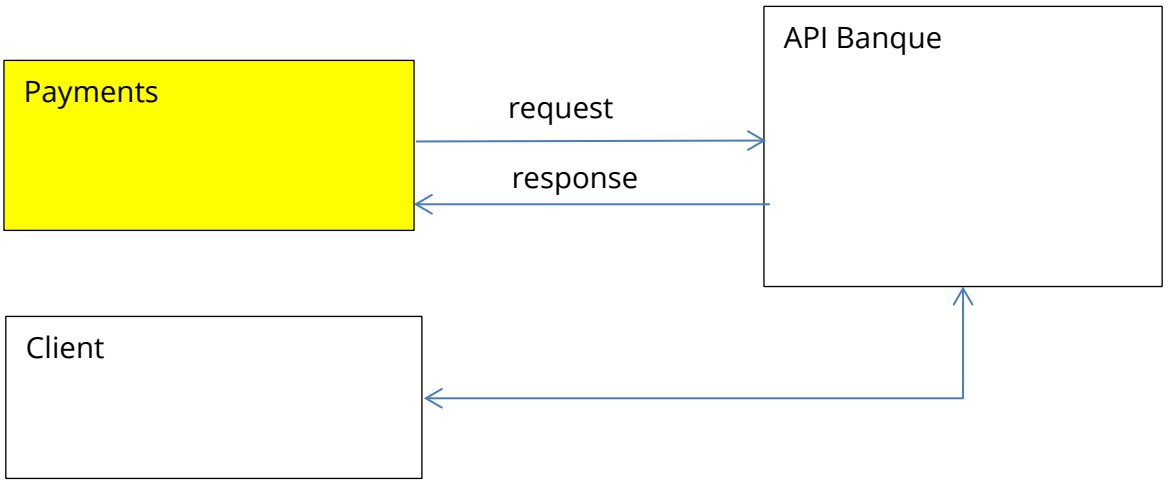
Payments	
bill_Id [PFK]	payment_methode_id [PFK]
1	1
2	1

Classe d'association Payments

Contient la clef primaire composé des clefs étrangère bill_id et payment_methode_id, ayant la valeur des clefs primaire de la table bills et paymentMethodes.

3.3 - Composants externes

3.3.1 - API Banque

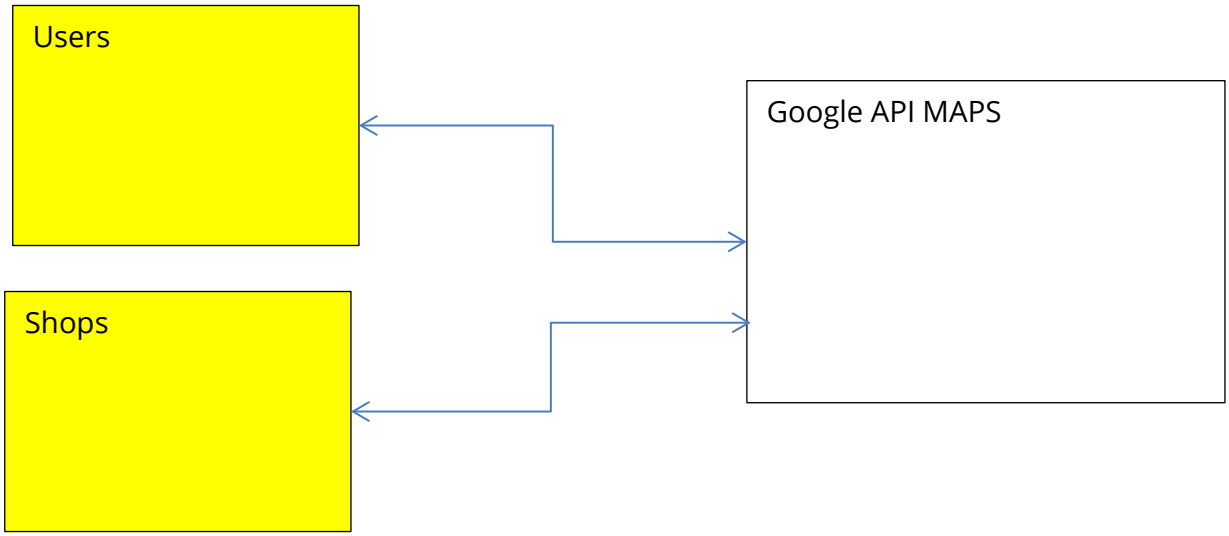


Concernant le règlement par carte bancaire, la classe payments procèdera selon les apis utilisées à la connexion vers les différent APIs.

Le client sera invité à renseigner ses identifiants, soit sur la page de paiement de l'application soit sur la page de sa banque qui ensuite seront transmis à l'api de la banque.

L'api demandera alors une confirmation au client pour le règlement, en fonction de la validation une réponse sera retourné à notre application qui traitera l'information changeant ainsi la valeur de la colonne status et avertira le client.

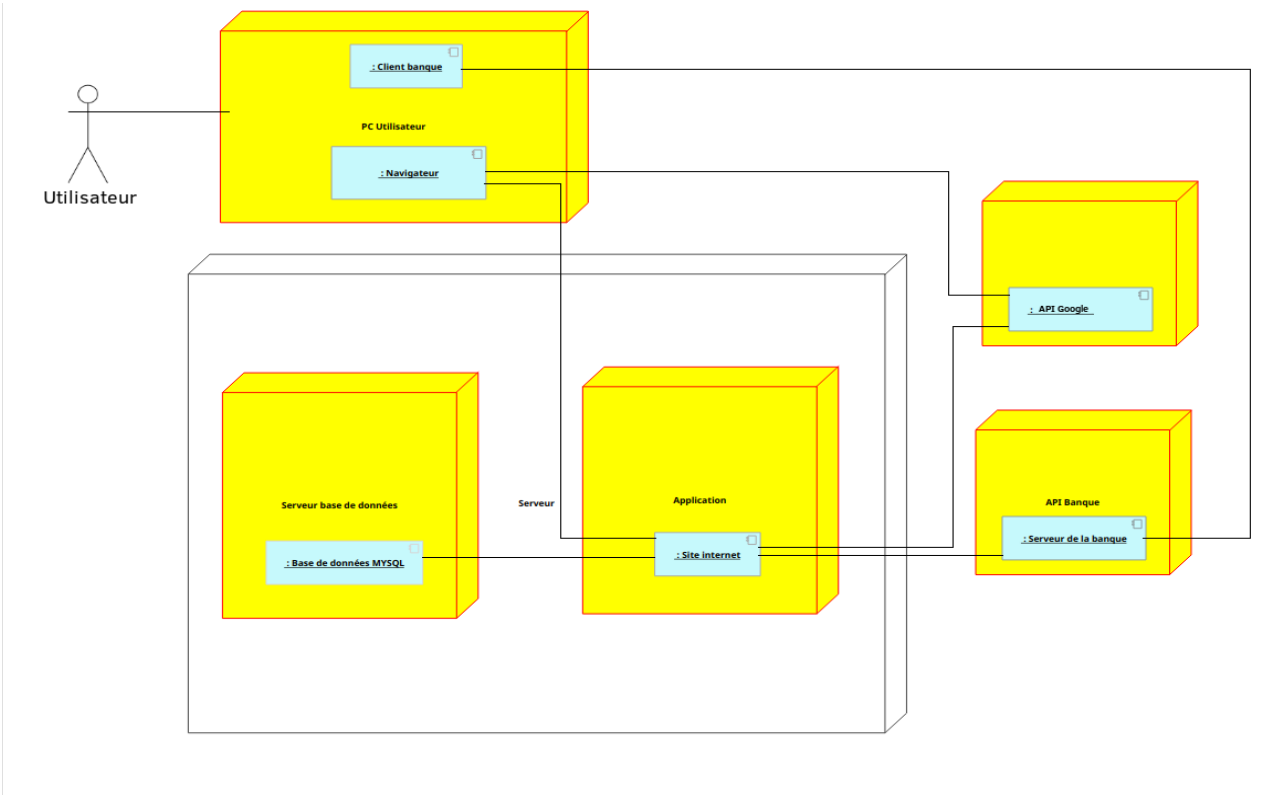
3.3.2 - API Google MAP



Les classes users et shops transmettent une adresse à l'api de google qui retourne les coordonnées gps, permettant ainsi de compléter les informations de localisation Latitude et Longitude de chaque table.

4 - Architecture de Deploiement

Diagramme UML de déploiement



4.1 - Serveur de Base de données

Les données seront sauvegardées grâce à PostgreSQL 9.6

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git**, l'intégration continu se fera grâce à travis.

6 - POINTS PARTICULIERS

6.1 - Fichiers de configuration

6.1.1 - Application web

Le fichier de configuration sera décomposé en 2 fichier une pour le développement et une pour la production.

Chacun de ses fichiers contiendront les configurations pour chaque environnement de développement.

Le fichier de configuration dit de production sera remonté sur le serveur par le biais d'une connexion ftp.

6.2 - Environnement de développement

L'application dans sa phase de développement sera réalisée sur un serveur local et versionné sur serveur git mis à disposition par nos services.

6.3 - Livraison

L'application sera livrée dans sa version de production depuis le serveur git.