|  |
| --- |
| **OC Pizza**  **Concevez la solution technique d’un système de gestion de pizzeria**  Dossier de conception technique  Version 1.1 |
| **Auteur**  Baptiste  *Analyste* |

Table des matières

1 - Versions 3

2 - Introduction 4

2.1 - Objet du document 4

3 - Domaine fonctionnel 5

3.1 - Description des données 5

3.1.1 - Commande 6

3.1.2 - Contact 6

3.1.3 - Employe 6

3.1.4 - Pizzaiolo 6

3.1.5 - Vendeur 7

3.1.6 - Livreur 7

3.1.7 - Gestionnaire de commande 7

3.1.8 - Compte 7

3.1.9 - Client 7

3.1.10 - Ville 7

3.1.11 - Pays 8

3.1.12 - Adresse 8

3.1.13 - Pizza 8

3.1.14 - Panier (association commande-pizza) 8

3.1.15 - Category 9

3.1.16 - Ingredient 9

3.1.17 - Recette (association ingredient-pizza) 9

3.1.18 - Restaurant 9

3.1.19 - Stock (association ingredient-restaurant) 9

4 - Architecture technique 10

4.1 - Application Web 10

4.1.1 - Composant Panier 10

4.1.2 - Composant Authentification 11

4.1.3 - Composant Accueil 11

4.1.4 - Composant Commande 11

4.1.5 - Composant utilisateur 11

4.1.6 - Composant Pizza 11

4.1.7 - Composant Ingrédient 11

5 - Architecture de Déploiement 12

5.1 - Database server 12

5.2 - Webserver 12

5.3 - Banque 12

5.4 - Ordinateurs (client/employee) 13

6 - Glossaire 14

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Baptiste Simon | 08/03/2019 | Création du document | 1.0 |
| Baptiste Simon | 11/03/2019 | Mise à jour des champs | 1.0.1 |
| Baptiste Simon | 12/08/2019 | Mise à jour Diagramme de Classe | 1.1 |
|  |  |  |  |

# Introduction

## Objet du document

Le présent document constitue le dossier de conception fonctionnelle de l'application "Pizza Easy Order".

Le but du document est de présenter les fonctionnalités techniques de l'application et les différents acteurs.

Les éléments du présent dossier découlent :

* des échanges avec le client
* de l'analyse des besoins du client
* de la représentation UML découlant de l'analyse des besoins

# Domaine fonctionnel

## Description des données

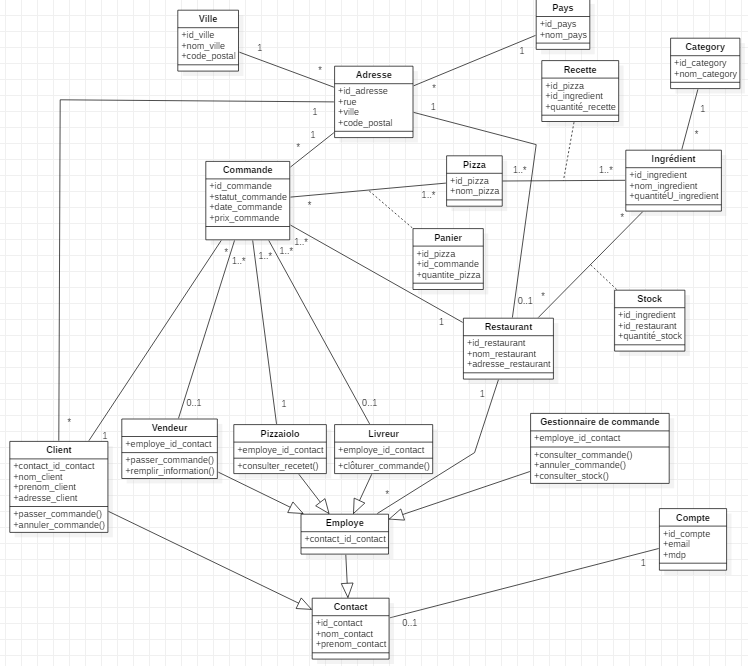


Diagramme de classe

### Commande

La classe commande regroupe les commandes en cours et passées.

Elle est composée de l'id, du statut de la commande, de la date de la commande et du prix de la commande et d'un ensemble de relation :

* Commande - Client : une commande a 1 et 1 seul client (et un client peut avoir 0 ou plusieurs commandes)
* Commande – Pizzaiolo : une commande a 1 et 1 seul pizzaiolo (et un pizzaiolo a 1 ou plusieurs commandes)
* Commande – Vendeur : une commande a 0 ou 1 vendeur (et un vendeur a 1 ou plusieurs commandes)
* Commande – Livreur : une commande a 0 ou 1 livreur (et un livreur a 1 ou plusieurs commandes)
* Commande - Adresse : une commande a 1 et 1 seul adresse (et une adresse a 0 ou plusieurs commandes)

Commande a une relation many-to-many avec la table Pizza.

### Contact

La classe contact rassemble les informations d'un contact / une personne utilisant l'application.

Elle est composée de l'id, du nom et du prénom ainsi qu'une relation :

* Contact - Compte : un employé a 1 et 1 seul compte (et un compte est à 0 ou 1 contact)

### Employe

La classe employe hérite de contact et représente les contacts employés par le restaurant.

Elle est composée de l'id du contact et d'une relation :

* Employé - Restaurant : un employé appartient à 1 et 1 seul restaurant (et un restaurant a 1 ou plusieurs employés)

### Pizzaiolo

La classe Pizzaiolo hérite d'Employé et représente les employés qui sont des pizzaiolos.

Elle est composée de l'id du contact de l'employé (cf. Commande).

### Vendeur

La classe Pizzaiolo hérite d'Employé et représente les employés qui sont des vendeurs.

Elle est composée de l'id du contact de l'employé (cf. Commande).

### Livreur

La classe Pizzaiolo hérite d'Employé et représente les employés qui sont des livreurs.

Elle est composée de l'id du contact de l'employé (cf. Commande).

### Gestionnaire de commande

La classe Pizzaiolo hérite d'Employé et représente les employés qui sont des gestionnaires.

Elle est composée de l'id du contact de l'employé et d'une relation (cf. Commande).

### Compte

La classe compte comporte les informations de connexion de chaque utilisateur.

Elle est composée de l'id, d'un email et d'un mot de passe et de d'une relation (cf. Contact).

### Client

La classe client contient les informations clés d'un client.

Elle est composée de l'id, du nom, du prénom et d'une relation :

* Client - Adresse : un client a 1 et 1 seule adresse (et une adresse est à 0 ou plusieurs clients)

### Ville

La classe ville comporte les informations nécessaire pour identifier une ville pour une adresse.

Elle est composée de l'id, du nom et du code postal. Elle a une relation :

* Ville – Adresse : une adresse appartient à 1 et 1 seule ville (et une ville a 0 ou plusieurs adresses)

### Pays

La classe Pays comporte les informations nécessaires pour identifier un pays pour une adresse.

Elle est composée de l'id, du nom et d'une relation :

* Pays – Adresse : une adresse appartient à 1 et 1 seul pays (et un pays a 0 ou plusieurs adresses)

### Adresse

La classe adresse comporte l'ensemble des adresses nécessaire pour la pizzeria.

Elle est composée de l'id, de la rue, du numéro et de trois autres relations (en complément des 2 précédentes) :

* Adresse – Client : une adresse appartient à 0 ou plusieurs client (et un client a 1 et 1 seule adresse).
* Adresse – Commande : une adresse appartient à 0 ou plusieurs adresses (et une commande a 1 et 1 seule adresse).
* Adresse – Restaurant : une adresse appartient à 0 ou 1 restaurant (et un restaurant a 1 et 1 seule adresse).

### Pizza

La classe pizza comporte l'ensemble des pizzas proposé par le groupe.

Elle est composée de l'id, du nom et du prix.

Pizza a aussi 2 relations many-to-many avec Commande et une avec Ingredient

### Panier (association commande-pizza)

La classe panier sert de relation many-to-many entre commande et pizza.

Elle est donc composée des id de ces deux classes et d'une quantité de pizza.

Une pizza est dans 0 ou plusieurs commandes.

Une commande a 1 ou plusieurs pizzas.

### Category

La classe Category comporte l'ensemble des catégories nécessaire pour classifier les ingrédients.

Elle est composée de l'id et du nom ainsi qu'une relation :

* Category – Ingredient : une category a 0 ou plusieurs ingrédient (et un ingrédient a 1 et 1 seule category)

### Ingredient

La classe ingredient comporte l'ensemble des ingrédients nécessaire pour le groupe.

Elle est composée de l'id, du nom, de la quantité unitaire et du seuil minimal.

Ingredient a une relation many-to-many avec Pizza et Restaurant.

### Recette (association ingredient-pizza)

La classe recette sert de relation many-to-many entre Pizza et Ingredient.

Elle est donc composée des id de ces deux classes et de la quantité d'ingrédient par recette.

Un ingrédient est utilisé dans 1 ou plusieurs pizzas.

Une pizza a 1 ou plusieurs ingrédients.

### Restaurant

La classe restaurant sert à rassembler l'ensemble des établissements du groupe.

Elle est composée de l'id, du nom et d'une relation :

* Restaurant - Adresse : un restaurant a 1 et 1 seule adresse (et une adresse a 0 ou 1 restaurant)

Restaurant a aussi une relation many-to-many avec Ingredient

### Stock (association ingredient-restaurant)

La classe stock sert de relation many-to-many entre la classe Restaurant et Ingredient.

Elle est donc composée des id de ces deux classes et d'une quantité d'ingrédient en stock par restaurant.

Un restaurant a 0 ou plusieurs ingrédients.

Un ingrédient est dans 0 ou plusieurs restaurants.

# Architecture technique

## Application Web

La pile logicielle est la suivante :

* Application Python(django)
* Serveur d'application Ngnix
* Base de données PostgreSQL

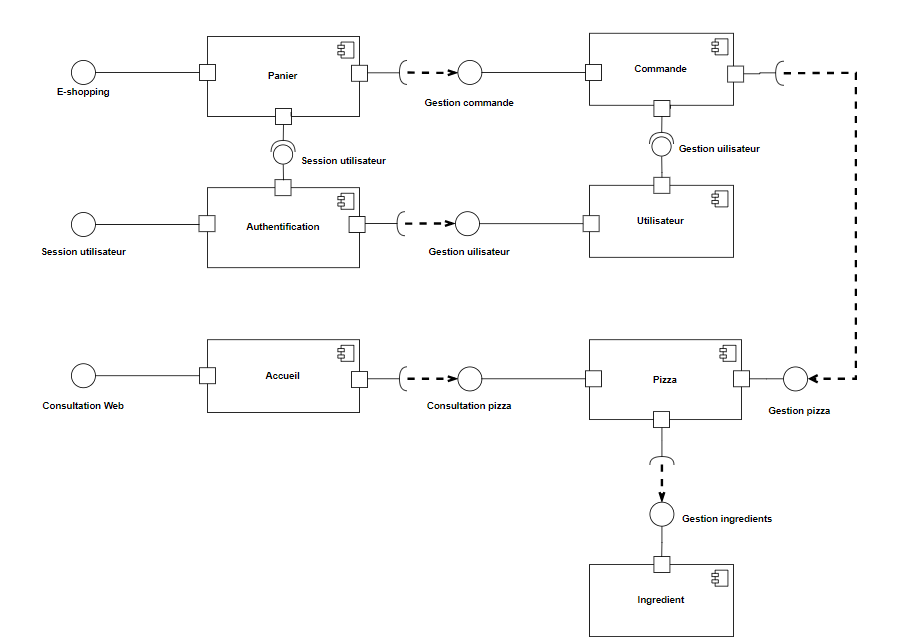


Diagramme UML de Composants

### Composant Panier

Le rôle de ce composant est de rassembler un ensemble d'information afin de préparer une commande.

Il est composé de l'interface de e-shopping qui permet la création de liste de pizza.

Le composant utilise la session utilisateur et est dépendant de l'interface gestion commande.

### Composant Authentification

Le rôle de ce composant est de gérer les connections et inscription sur la solution web.

Il est composé de l'interface session utilisateur qui permet d'initier une session d'utilisateur.

Le composant est dépendant de l'interface gestion utilisateur.

### Composant Accueil

Le rôle de ce composant est de rassembler les informations nécessaires à afficher sur la page d'accueil : les différentes pizzas, les promotions du moment et autres informations.

Il est composé de l'interface consultation web qui permet de consulter les éléments de l'application web.

Le composant est dépendant de l'interface consultation pizza.

### Composant Commande

Le rôle de ce composant est de gérer les commandes en cours et passées.

Il est composé de l'interface gestion commande qui permet de générer et gérer les commandes.

Le composant est dépendant de l'interface gestion pizza.

### Composant utilisateur

Le rôle de ce composant est de gérer les fonctions disponibles suivant l'utilisateur et d'y gérer les paramètres du compte.

Il est composé de l'interface gestion utilisateur qui permet la modification de l'utilisateur et de gérer ses accès suivant ses droits.

### Composant Pizza

Le rôle de ce composant est de gérer les pizzas suivant leurs disponibilités et recette.

Il est composé des interfaces consultation pizza qui permet de consulter les pizza et gestion pizza qui permet de gérer les pizzas.

### Composant Ingrédient

Le rôle de ce composant est de gérer les ingrédients nécessaires pour les pizzas et de gérer les stocks.

Il est composé de l'interface gestion ingredients.

# Architecture de Déploiement

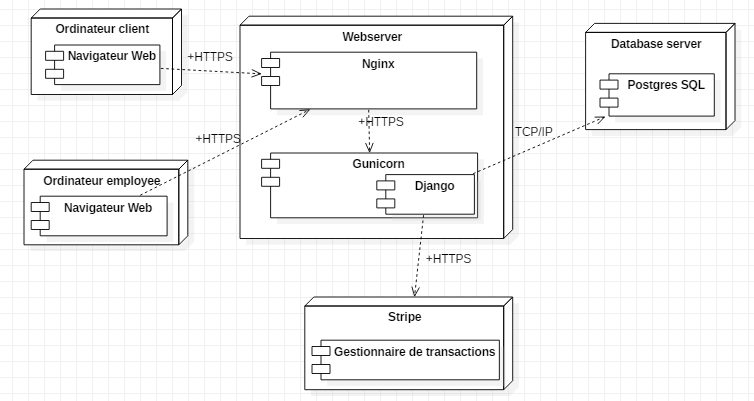


Diagramme UML de déploiement

## Database server

Ce serveur contient la base de données nécessaire à la gestion des commandes des pizzeria.

Les données de toutes les pizzerias sont réunies sur ce serveur.

La Base de données est gérée en PostregSQL.

## Webserver

Ce serveur contient :

* Nginx qui gère les requêtes depuis internet. Il décide aussi si ces requêtes doivent être transféré à Gunicorn
* Gunicorn qui reçoit les requêtes http transféré par Nginx. Il exécute directement les requêtes ou utilise le code de Django. La réponse est envoyée vers Nginx.

## Stripe

Son rôle est de gérer les transactions financières de façon sécurisée.

Il reçoit des requêtes directement du webserver et permet d'interagir avec différent mode de paiements.

## Ordinateurs (client/employee)

Le rôle est d'utiliser la solution grâce à un navigateur web.

Les clients et les employées sont séparés car leur utilisation n'est pas la même.

# Glossaire

|  |  |
| --- | --- |
| **UML** | Unified Modeling Language |
|  |  |