OC PIZZA

SYSTEME DE GESTION DE PIZZERIAS

Dossier de conception technique

Version 1.0

Auteur
Crispin Pigla

A Faire :

- Remplacer manuellement les valeurs entre <>
- Les valeurs entre {{}} sont à renseigner dans les propriétés du document

IT Consulting & Development

S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A

Table des matières

1 - Versions

Auteur	Date	Description	Version
Crispin Pigla	12/12/2020	Création du document	1.0

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier de conception fonctionnelle de l'application OC pizzeria.

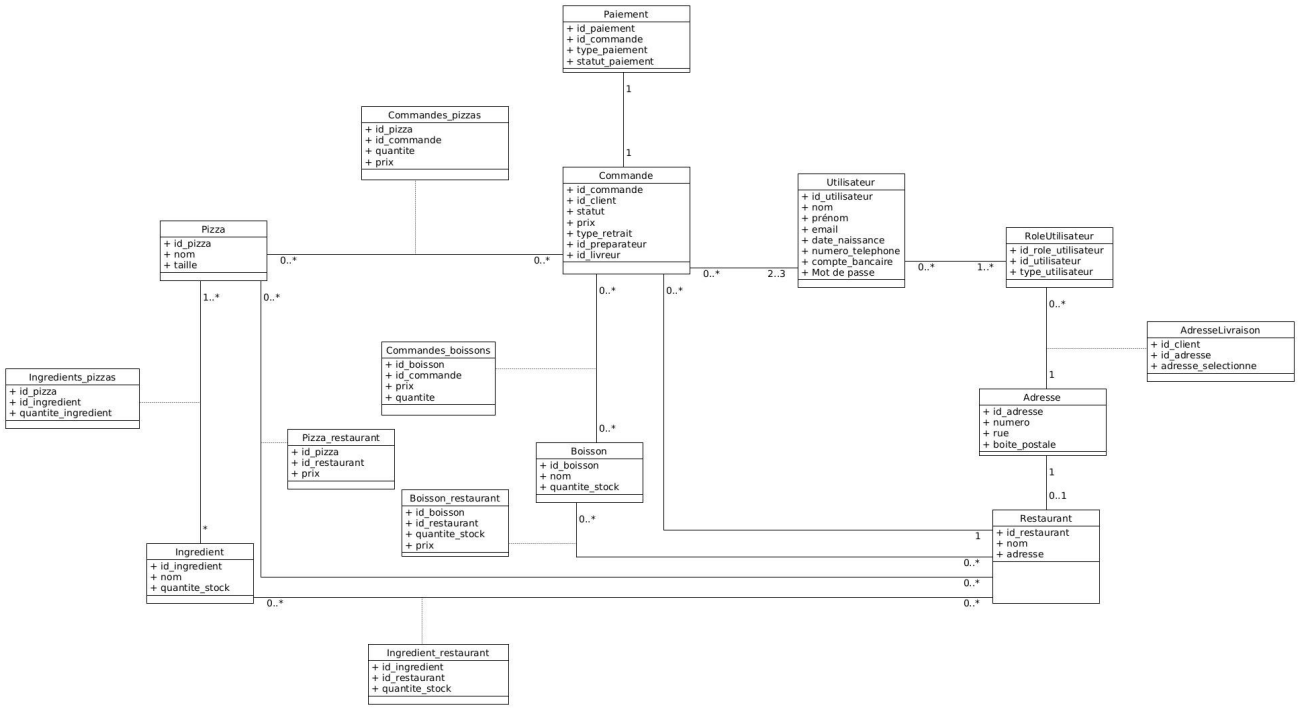
Les éléments du présents dossiers découlent :

• des discussions avec le client

• du cahier des charges

3 - Le domaine fonctionnel

3.1 - Référentiel



Ce diagramme représente le diagramme de classe des différentes tables de la base de données du projet .

3.1.1 - Règles de gestion

Description textuelle du diagramme de classe:

Le diagramme de classe ci-dessus est composé des classes suivantes :

- Pizza: représente la classe des pizzas des restaurants. Cette classe possède plusieurs attributs dont:
  - id_pizza: qui représente l'identifiant unique de la pizza dans la base de donnée,
  - nom: qui représente le nom de la pizza,

- taille: qui représente la taille de pizza (petit, moyen, grand).

- Ingrédient: représente la classe des ingrédients de pizzas des restaurants. Cette classe possède plusieurs attributs dont:
  - id_ingredient: qui représente l'identifiant unique de l'ingredient,
  - nom: qui représente le nom de l'ingrédient,
  - quantite_stock: qui représente la quantité d'ingrédients présents en stock dans tous les restaurants.
- Paiement: représente la classe des paiements de commandes effectuées. Cette classe possède plusieurs attributs dont:
  - id_paiement: qui représente l'identifiant unique du paiement,
  - id_commande: qui représente la commande dont le paiement fait référence,
  - type_paiement: qui représente le mode de paiement utilisé lors du règlement (en ligne, par catre au retrait, en espece au retrait).
  - statut_paiement: qui représente le statut du paiement(effectuée, pas effectué).
- Commande: représente la classe des commandes passées. Cette classe possède plusieurs attributs dont:
  - id_commande: qui représente l'identifiant unique de la commande,
  - id_client: qui représente l'identifiant du client qui a passé la commande,
  - statut: qui représente le statut de la commande(en attente de préparation, en cours de préparation, commande prépapée, en cours de livraison, commande livrée),
  - prix: qui représente le prix de la commande,
  - type_retrait: qui représente le type de retrait choisi par le client (sur place, livraison) ,
  - id_preparateur: qui représente l'identifiant du préparateur de la commande,
  - id_livreur: qui représente l'identifiant du livreur de la commande.
- Boisson: représente la classe des boissons dans le stock des restaurants . Cette classe possède plusieurs attributs dont:
  - id_boisson: qui représente l'identifiant unique de la boisson,
  - nom: qui représente le nom de la boisson
  - quantite_stock: qui représente la quantité dans le stocks de tous les restaurants.
- Utilisateur: représente la classe des utilisateurs de l'application. Cette classe possède plusieurs attributs dont:
  - id_utilisateur: qui représente l'identifiant unique de l'utilisateur,
  - nom: qui représente le nom de l'utilisateur,
  - prenom: qui représente le prenom de l'utilisateur,
  - email: qui représente l'adresse email de l'utilisateur,

- date_naissance: qui représente la date de naissance de l'utilisateur,
- numero_telephone: qui représente le numero de téléphone de l'utilisateur,
- compte_bancaire: qui représente le numero de compte bancaire de l'utilisateur,
- mot de passe: qui représente le mot de passe de l'utilisateur.

- RoleUtilisateur: représente la classe des different roles que peut avoir un utilisateur (client, preparateur, livreur, responsable, directeur). Cette classe possède plusieurs attributs dont:
  - id_utilisateur: qui représente l'identifiant de l'utilisateur,
  - type_utilisateur: qui représente le role associé à l'utilisateur(client, preparateur, livreur, responsable, directeur)
- Adresse: représente la classe des adresses des clients et des restaurants. Cette classe possède plusieurs attributs dont:
  - id_adresse: qui représente l'identifiant unique de l'adresse,
  - numero: qui représente le numero de rue lié à l'adresse,
  - rue: qui représente le nom de la rue liée à l'adresse,
  - boite_postale: qui représente la boite postale liée à l'adresse.
- Restaurant: représente la classe des restaurants du groupe de la pizzeria. Cette classe possède plusieurs attributs dont:
  - id_restaurant: qui représente l'identifiant unique du restaurant,
  - nom_restaurant: qui représente le nom du restaurant,
  - adresse: qui représente l'identifiant de l'adresse liée au restaurant.
- Ingrédients_pizzas: représente la classe des ingrédients qui constituent une pizza. Cette classe possède plusieurs attributs dont:
  - id_pizza: qui représente l'identifiant de la pizza,
  - id_ingredient: qui représente l'identifiant de l'ingredient associé à la pizza,
  - quantite_ingredient: qui représente la quantité d'ingrédients utilisés pour la préparation de la pizza,
- Commandes_pizzas: représente la classe des pizzas qui constituent une commande. Cette classe possède plusieurs attributs dont:
  - id_pizza: qui représente l'identifiant de la pizza associée à une commande,
  - id_commande: qui représente l'identifiant de la commande,
  - quantite: qui représente le nombre d'instance de la pizza dans la commande,
  - prix: qui représente le prix de la pizza en fonction de sa quantité dans la commande.
- Commandes_boissons: représente la classe des boissons qui constituent une commande . Cette classe possède plusieurs attributs dont:
  - id_boisson: qui représente l'identifiant de la boisson associée à une commande,
  - id_commande: qui représente l'identifiant de la commande,

    - prix: qui représente le prix de la boisson en fonction de sa quantité dans la commande,
    - quantite: qui représente le nombre d'instance de la boisson dans la commande.
- AdresseLivraison: représente la classe des adresses de livraison des commandes. Cette classe possède plusieurs attributs dont:
    - id_client: qui représente l'identidiant du client
    - id_adresse: qui représente l'identifiant de l'adresse liée à l'utilisateur
    - adresse_selectionne: qui indique si l'adresse de livraison est celle sélectionné pour la prochaine livraine livraison.
- Boisson_restaurant: représente la classe des boissons dans un restaurant. Cette classe possède plusieurs attributs dont:
    - id_boisson: qui représente l'identifiant de la boisson,
    - id_restaurant: qui représente l'identifiant du restaurant,
    - quantite_stock: qui représente la quantité de la boisson présente dans le stock du restaurant,
    - prix: qui représente le prix de la boisson dans le restaurant.
- Ingredient_restaurant: représente la classe des ingrédients présents dans les stocks d'un restaurant . Cette classe possède plusieurs attributs dont:
    - id_ingredient: qui représente l'identifiant de l'ingrédient,
    - id_restaurant: qui représente l'identifiant du restaurant,
    - quantite_stock: qui représente la quantité d'ingredients en stocks,
- Pizza_restaurant: représente la classe des pizzas préparées dans un restaurant . Cette classe possède plusieurs attributs dont:
    - id_pizza: qui représente l'identifiant de la pizza,
    - id_restaurant: qui représente l'identifiant du restaurant dans lequel la pizza est préparée,
    - prix: qui représente le prix de la pizza dans le restaurant.

Description textuelle des relations entre les classes :

- Dans la relation ingredient-pizza, une pizza est composée de plusieurs ingrédients et un ingrédient pourra être contenu dans une ou plusieurs pizzas. Nous avons donc une relation plusieurs à plusieurs.
- Dans la relation pizza-commande, une pizza pourra être contenu dans 0 ou plusieurs commandes et une commande pourra contenir entre 0 et plusieurs pizzas. Nous avons donc une relation plusieurs à plusieurs.
- Dans la relation boisson-commande, une boisson pourra être contenu dans 0 ou plusieurs commandes et une commande pourra contenir entre 0 et plusieurs boissons. Nous avons donc une relation plusieurs à plusieurs.
- Dans la relation paiement-commande, une commande sera payée une seule fois et paiement ne sera rattaché qu'à une seule commande. Nous avons donc une relation un à un.
- Dans la relation commande-restaurant, une commande sera passée dans un

restaurant et un restaurant pourra produire entre 0 et plusieurs commandes . Nous avons donc une relation un à plusieurs.

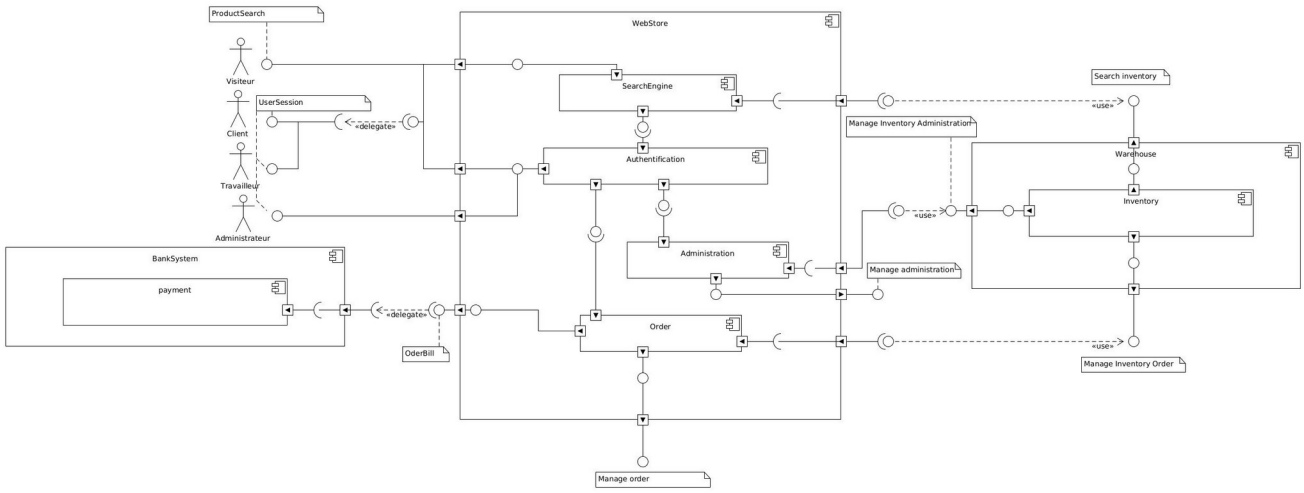
- Dans la relation commande-utilisateur, une commande sera passée par un utilisateur, préparée un autre utilisateur et (dans le cas ou elle est livrée) livrée par un autre utilisateur. Un utilisateur pourra passer (préparer ou livrer) entre 0 et plusieurs commandes. Nous avons donc une relation plusieurs à plusieurs.
- Dans la relation utilisateur-adresse, un utilisateur pourra avoir 0 (si l'utilisateur est un travailleur), 1 ou plusieurs adresses (si l'utilisateur est un client) et une adresse pourra etre affectée à 0 ou plusieurs utilisateurs. Nous avons donc une relation plusieurs à plusieurs.
- Dans la relation adresse-restaurant, une adresse pourra être rattachée à 0 ou un restaurant et un restaurant n'aura qu'une seule adresse. Nous avons donc une relation de un à un.
- Dans la relation utilisateur-role, un utilisateur pourra avoir un ou plusieurs roles et un role pourra etre affecté à 0 ou plusieurs utilisateurs. Nous avons donc une relation plusieurs à plusieurs.
- Dans la relation ingrédient-restaurant, un ingrédient pourra être utilisé dans 0 ou plusieurs restaurants et un restaurant pourra utiliser 0 (à la création du restaurant) ou plusieurs ingrédients. Nous avons donc une relation plusieurs à plusieurs.
- Dans la relation pizza-restaurant, une pizza pourra être preparée dans 0 ou plusieurs restaurants et un restaurant pourra preparer 0 ou plusieurs pizzas. Nous avons donc une relation plusieurs à plusieurs.
- Dans la relation boisson-restaurant, une boisson pourra être vendue dans 0 ou plusieurs restaurants et un restaurant pourra vendre 0 ou plusieurs boissons. Nous avons donc une relation plusieurs à plusieurs.

4 - Architecture Technique

4.1 - Application Web

La pile logicielle est la suivante :

- **Python 3.8.5 :**
  - Par la simplicité de son intégration au projet il nous permettra d'accorder plus d'attention au contenu de la solution
  - Les applications pythons sont simples à maintenir.
- **Serveur d'application Gunicorn 20.0.4 :**
  - Permet de gérer un grand nombre de connexions avec très peu de processeur et de mémoire.
  - Par son efficacité il forme avec nginx une base performante de serveur.
- **Serveur web NGINX 1.18.0 :**
  - Est l'un des serveurs les plus populaires
  - Peut être configurer comme un serveur Web classique pour servir des fichiers statiques et pour des requêtes dynamiques
  - Peut être utilisé avec un ou plusieurs serveurs applicatifs avec un mécanisme de répartition de charge.
- **POSTGRESQL 13.1 :**
  - Etant la base de données open source la plus populaire au monde, elle offre plus de flexibilité pour la maintenance.
- **Django 3.1.1 :**
  - Permet de créer des applications sécurisées et complètes.
  - Utilise l'architecture Modèle/Vue/Controller (MVC),ce qui permet d'avoir une application bien structurée



4.1.1 - Composant BankSystem

Ce composant représente le systeme bancaire chargé d'effectuer les transactions financière entre le client et le restaurant.

Ce composant prend en son interface requise la facture de la commande et la transmet au composant paiement.

4.1.2 - Composant Payment

Ce composant est chargé d'effectuer les transactions financière entre le client et le restaurant.

Ce composant prend en son interface requise la facture de la commande et execute le paiement.

4.1.3 - Composant WebStore

Ce composant est chargé de traiter les differentes operations demandées par l'utilisateur de l'application.

Ce composant prend en son interface requise un inventaire de recherche, un gestionnaire de commande et un gestionnaire de restaurant et rend en son interface fournie une session de connexion, un produit recherché par un visiteur et un gestionnaire de commande

4.1.4 - Composant SearchEngine

Ce composant est le moteur de recherche de l'application. Il est chargé de rechercher un produit dans la base de donnée.

Ce composant prend en son interface fournie un inventaire de recherche et rend en son interface fournie un produit recherché.

4.1.5 - Composant Authentification

Ce composant est chargé d'exécuter la connexion d'un utilisateur au système.

Ce composant prend en son interface requise un produit recherché et rend en son interface fournie une session d'utilisateur.

4.1.6 - Composant Administration

Ce composant est chargé de traiter les demandes d'utilisateur relatives à l'administration de restaurants.

Ce composant prend en son interface requise une session d'utilisateur et un gestion d'inventaire de restaurant et rend en son interface fournie un gestionnaire de restaurant.

4.1.7 - Composant Commande

Ce composant est chargé de traiter les demandes d'utilisateur relatives aux commandes.

Ce composant prend en son interface requise une session d'utilisateur et un gestion d'inventaire de commande et rend en son interface fournie un gestionnaire de commande.

4.1.8 - Composant Warehouse

Ce composant est chargé de fournir, modifier, restituer ou supprimer des éléments dans la base de donnée.

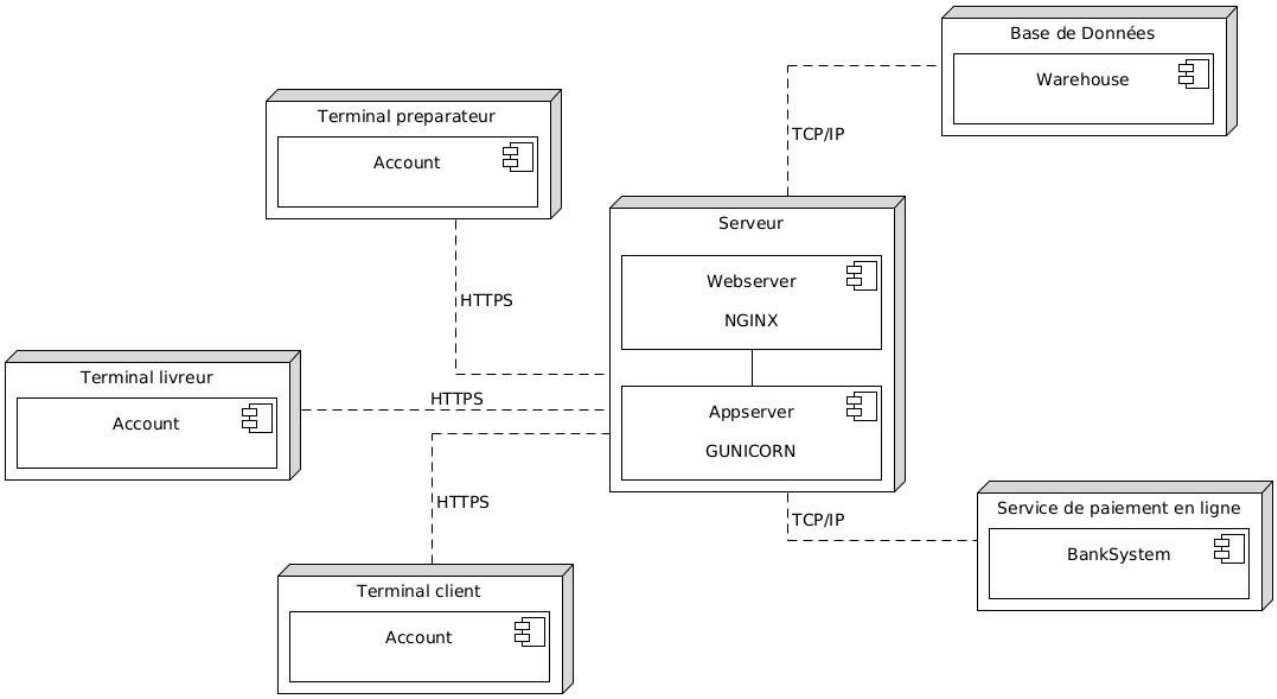
Ce composant rend en ses interfaces fournies les gestionnaires d'inventaires de restaurant, de commandes et un inventaire de recherche.

4.1.9 - Composant Inventory

Ce composant est chargé d'executer les operations de lecture, modification, restitution ou suppression des éléments dans la base de donnée.

Ce composant rend en ses interfaces fournies les gestionnaires d'inventaires de restaurant, de commandes et un inventaire de recherche.

5 - Architecture de Déploiement



Le diagramme ci-dessus représente le diagramme de déploiement de la solution.

5.1 - Serveur de Base de données

Etant la base de données open source la plus populaire au monde, elle offre plus de flexibilité pour la maintenace.

Les applications pythons sont simples à maintenir.

5.2 - Serveur Gunicorn

Permet de gérer un grand nombre de connexions avec très peu de processuer et de mémoire.

Par son efficaité il forme avec nginx une base performante de serveur.

5.3 - Serveur NGINX

Est l'un des serveurs les plus populaires.

Peut etre configurer comme un serveur Web classique pour servir des fichiers statiques et pour des requêtes dynamiques.

Peut etre utilisé avec un ou plusieurs serveurs applicatifs avec un mécanisme de répartition de charge.

6 - Glossaire

Domaine fonctionnel	Ensemble des tables et des relations dans une base de donnée
Serveur	Dispositif qui offre des services à un ou plusieurs clients