|  |
| --- |
| **Jean-Charles Gibier**  **OC Pizza**  Dossier de conception technique  Version 0.1 |
| **Auteur**  Jean-Charles Gibier  *Chef de projet* |

Table des matières

1 - Versions 3

2 - Introduction 4

2.1 - Objet du document 4

3 - Le domaine fonctionnel 5

3.1 - Référentiel 5

3.1.1 - Règles de gestion 9

4 - Architecture Technique 10

4.1 - Application Web 10

4.1.1 - Pile logicielle 10

4.1.2 - Diagramme de composants 10

4.1.3 - Composants externes 10

4.1.4 - Composants internes 11

5 - Architecture de Déploiement 12

5.1 - Serveur de Base de données 13

6 - Glossaire 14

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Jean-Charles | 15/06/2020 | Création du document | 001 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Objet du document

Le présent document constitue le dossier de conception technique de l'application PSOC (Pizza store of OC). Il a pour objectif de décrire des solutions techniques du système informatique mis en place par IT Consulting pour répondre aux besoins exprimés par le groupe OC Pizza.

Les éléments du présents dossiers découlent :

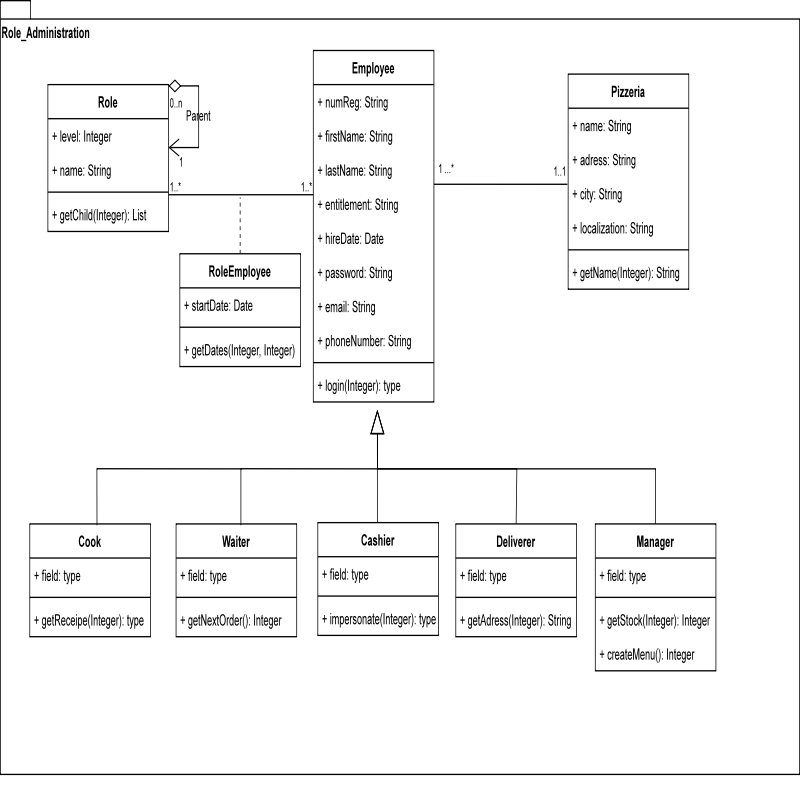
• Du dossier de conception fonctionnelle (V.002) de ce projet

• Du recueil des besoins client effectué par IT consulting & développement

# Le domaine fonctionnel

## Référentiel

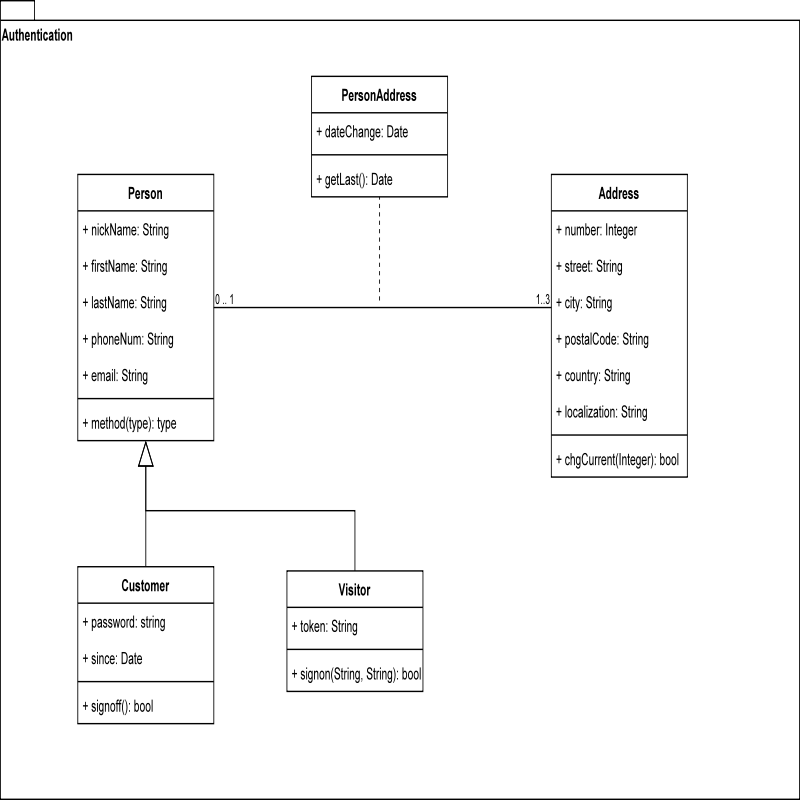
Le domaine fonctionnel du système OC Pizza décrit l’ensemble des classes qui serviront de support à la création du Modèle Physique de Donnée. Celui-ci établira le schéma des données relationnelles qui sera exploité par le programme.  
Ces classes sont réparties en trois sous ensembles fonctionnels définis ci après.  
  
*Notes:*   
- Les liaisons entre ces trois domaines sont définies dans le schéma général situé sur [le dépot Git de ce projet](https://github.com/jean-charles-gibier/OC_pizza/blob/master/docs/CD_P6.svg).   
- Les listes des méthodes présentées dans chaque classe sont indicatives et ne sont pas exhaustives.  
  
  
**Domaine “Roles and administration”:**



*Explications:*

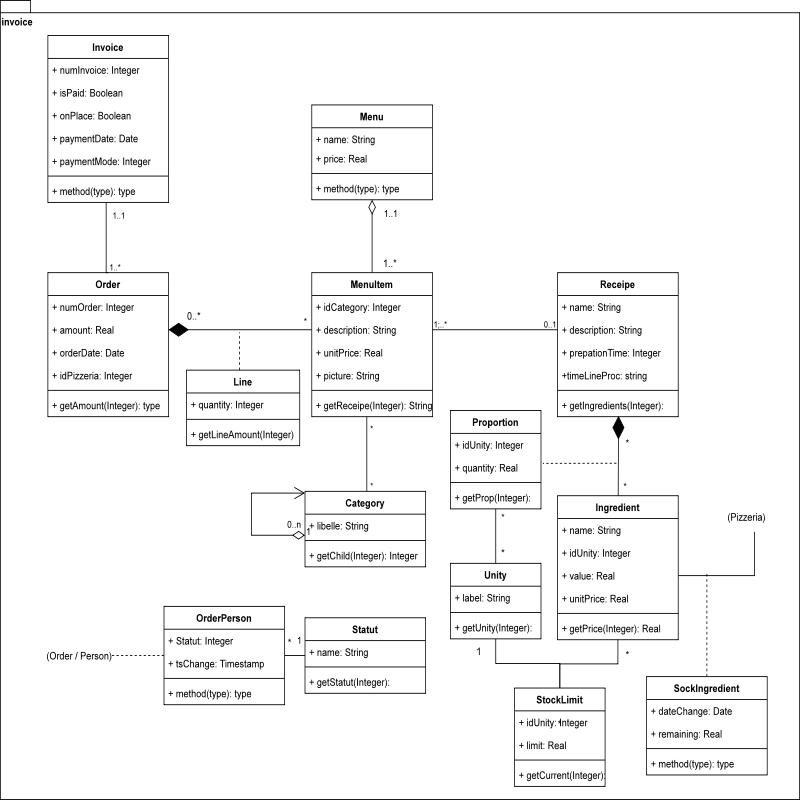
Chaque **employé** est affecté à une **pizzeria**. S’il occupe un poste au sein de plusieurs établissements du groupe, on considérera simplement que l’affectation représente son lieu de travail principal.  
La spécialisation de chaque employé (cuisinier, manager, etc.) peut être déterminée par la relation “employee – role” (affectations multiples hiérarchisées et datées) ou plus simplement par la propriété “entitlement” qui permettra éventuellement de dériver la classe ‘employee’.  
Enfin chaque pizzeria est rattachée à son lot de commandes effectuées par ses clients par les liaisons entre les classes **pizzeria**, **invoice**, **order** et **person**. (Cette liaison n’est visible que sur le schéma principal).

**Domaine “Authentication”:**



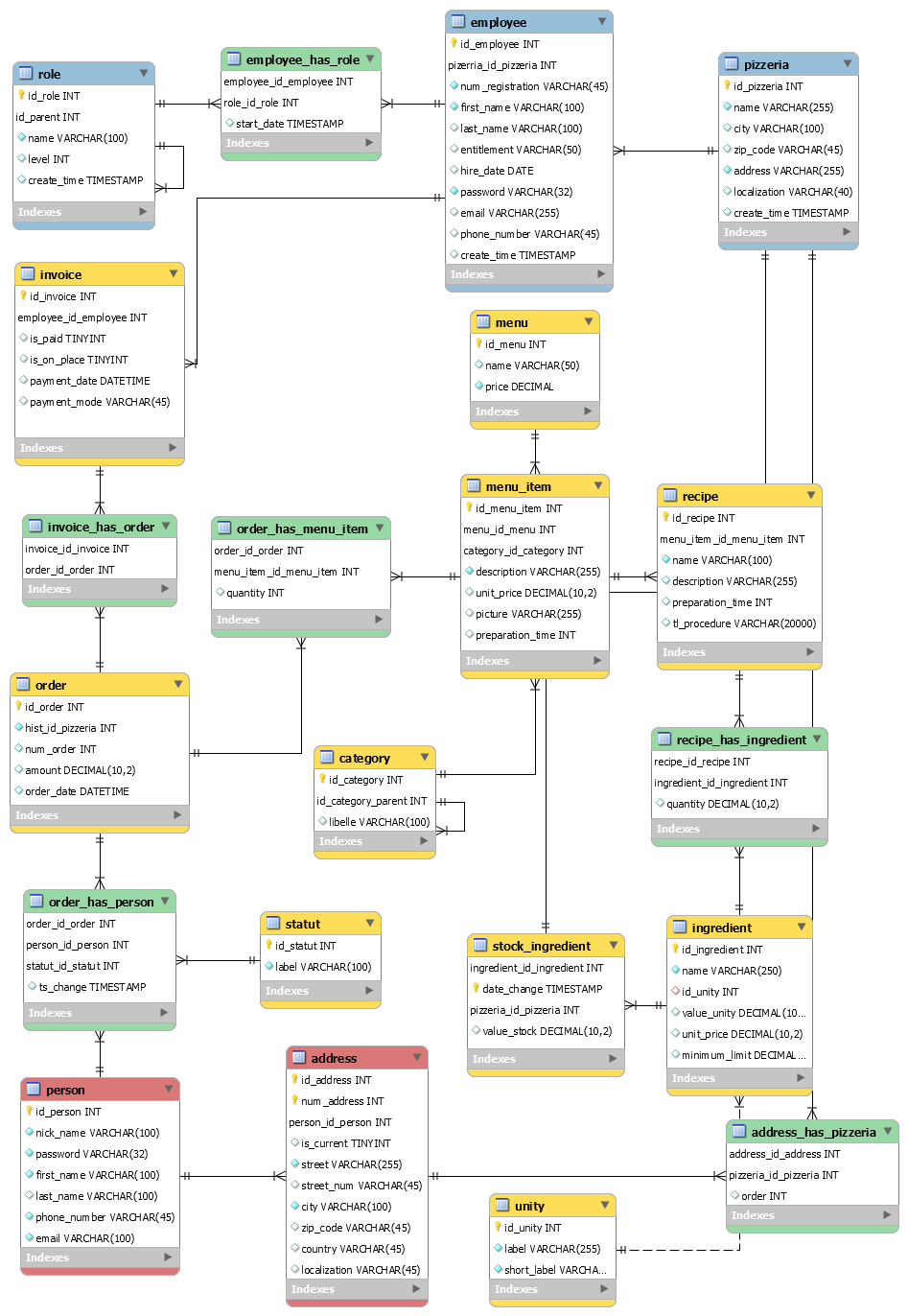
*Explications:*   
Une **personne** se spécialise en **client** ou **visiteur** en fonction des propriétés renseignées ou non dans l’entité ‘**person**’. Quel que soit son état, elle peut déclencher le processus de commande. Dans le cas d’un visiteur, la complétion des propriétés sera requise à la validation de la commande.  
Une personne peut stocker un nombre limité d’**adresses** (ici arbitrairement fixé à 3), dont une adresse de référence (choix par défaut).  
  
*Note:* Les classes ‘**person**’ et ‘**employee**’ auraient pu être regroupées au sein d’une même entité générique. Néanmoins nous avons intentionnellement séparé ces objets dans notre modèle. En effet, les entités caractérisées évoluent dans deux contextes d’exploitation distincts, pour des finalités distinctes. En conséquence, les tables doivent être manipulées indépendamment. Exemple métier : une campagne commerciale promotionnelle n’a pas de sens pour les **employés** d’une **pizzeria**. Par ailleurs, le mix de concept entre le **client** final et l’accédant au back-office (**manager** et autres comptes privilégiés) n’est pas cohérent sur le plan sécuritaire.

**Domaine “Invoices”:**



*Explications:*   
La gestion du flux des **commandes** est au cœur du système. Le document de spécifications fonctionnelles précise que le processus est déclenché par le **client**. Une **facture (invoice**) sera attachée à cette commande (nous laissons la possibilité d’affecter plusieurs **commandes** à une **facture**). Une commande comprend de 0 à plusieurs **éléments de menu** (menus items ou plat).  
Un élément de **menu** est défini par une ou plusieurs **recette**(s) (**recipe**) qui comprend la description et le temps de la préparation. Une **recette** est composée de plusieurs **ingrédients.** Chaque composition est associée à une **proportion** et une **unité**. Chaque **pizzeria** possède un **stock** par **ingrédient** daté (permettant ainsi de tracer la progression des **stocks**).  
  
Les classes **menu** et **catégorie** sont établies pour faciliter le regroupement et classification des plats proposé aux clients. Une **catégorie** distinguera par exemple les types «entrée», «boisson», «dessert» dans les occurrences **menu-item** (la composition réflexive permettant des sous catégories). Le **menu** quand à lui regroupe un set de **catégories**.

**Modèle physique de données:**



### Règles de gestion

Les règles de gestion découlant des relations et des cardinalités présentées :

- un **employé** est affecté à une **pizzeria**, le système ne gère pas par exemple le cas d’employé détaché ou d’un manager s’occupant de plusieurs pizzeria. Néanmoins, un manager peut avoir accès aux données de plusieurs pizzeria.

- la classe ‘**person**’ est liée à 3 **adresses** par défaut mais le modèle permet un nombre illimité d’adresse. L’adresse de référence sera définie en fonction de sa plus grande proximité avec un des **restaurants** du groupe OC.   
- La classe **rôle** étend la notion d’attribution de poste par composition (tel rôle comprend les fonctions de tel autre rôle)

- Une **pizzeria** est reliée à une **personne** en fonction de son adresse ou en fonction d’un choix préalable si le client n’est pas inscrit (dans ce cas la pizzeria sera reliée à une adresse vide en attente de complétion).

- Une occurrence de **stock ingrédient** est relié à un ingrédient, une pizzeria, une date.  
- Une **commande** est relié à une personne avec une liaison «many to many». La classe de liaison est datée afin de pouvoir historiser les états (les **statuts**) successifs d’une commande donnée et d’en faciliter le monitoring.

- Une table de liaison **stock\_ingredient** entre un **ingrédien**t et une **pizzeria** est datée pour suivre la progression des **stocks**.

- Chaque **ingrédient** possède une unité (Litre, Kg, Unité) pour évaluer le volume du **stock** ainsi que la **quantité** nécessaire dans l’application d’une **recette**.

- Une **personne** peut avoir plusieurs **adresses**, si c’est le cas, les adresses seront ordonnées par ordre de priorité pour le choix de proximité des pizzeria.

- Un **menu item** appartient à un ou plusieurs **menu**. (un menu item = un plat). La classe menu regroupe un groupe de catégories de plats (entrée, plat, dessert, boisson) via une liaison «many to many» entre les menus et les plats.

- Un **menu-item** /plat peut avoir plusieurs **recettes** (jusqu’à 3) pour s’adapter en fonction de l’état des **stocks** ou des **demandes** client. Le pizzaiolo pourra choisir une alternative le cas échéant.

- ...

# Architecture Technique

## Application Web

### Pile logicielle

La pile logicielle proposée est la suivante :

* Application **Python 3.7**
* Serveur d'application **Gunicorn 20.x**
* Serveur d'application **Nginx1.18**
* Base de donnée **Mysql 8.020**

### Diagramme de composants

L’organisation des éléments logiciels du système d’ OC Pizza met en évidence les dépendances entre les composants. Ils décrivent les interfaces entre les composants internes, internes et externe .

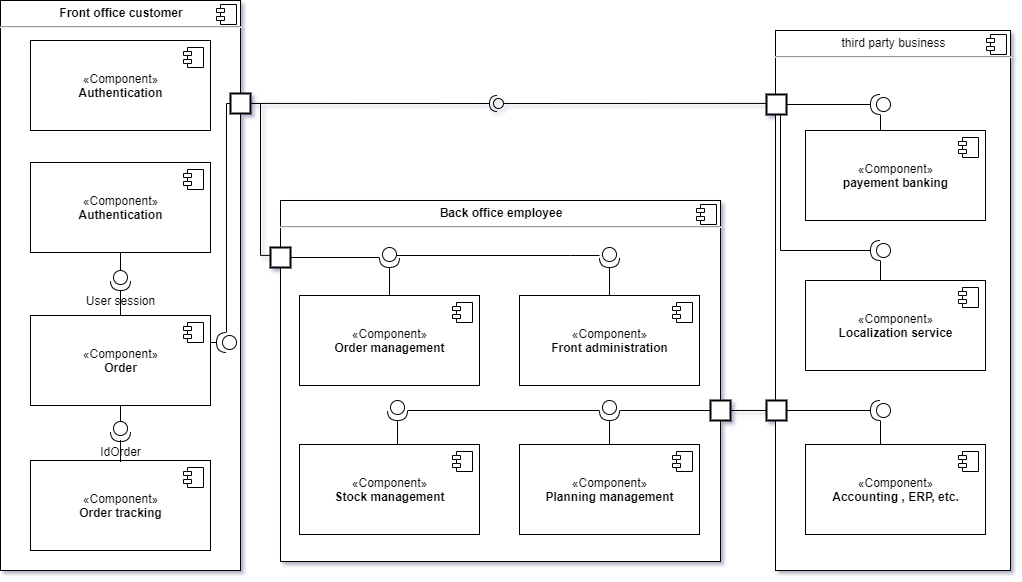


Diagramme UML de Composants

### Composants externes

**Localisation et géolocalisation**: Ce composant intervient dans 2 étapes précises :  
- lorsque le client demande une localisation du restaurant le plus proche en fonction de celle de l’appareil avec lequel il se connecte au service (téléphone ou ordinateur portable, connexion fixe). Le composant devra renvoyer une position plus ou moins précise pour que le système puisse répondre : «vous êtes à tel endroit, nous vous proposons le restaurant X le plus proche».  
- lorsque l’adresse de livraison est définie ou celle du restaurant, le composant devra renvoyer les coordonnées latitude et longitude correspondantes afin de déterminer le temps de livraison probable.

**Service de paiement**: Ce composant va échanger avec le système bancaire via le portail API des réseaux de retrait CB les plus connus (Visa / Master card / autres) L'API va assurer le bon déroulement de la transaction sécurisé par un système de jeton. Les informations sur le client et le montant à régler seront demandées au client et celui ci aura directement accès au service délivrés par sa carte bancaire.

**Service ERP**  : Composant (éventuel) nécessaire à la communication avec divers progiciels de gestion (comptabilité, organisation RH, gestion commerciale etc.). La description de l’interface est fortement dépendante du système choisi. (De manière simplifiée on peut aussi envisager une interface avec des produits tel que Access ou PowerBI ou même un fichier csv pour Excel).

### Composants internes

Les composants internes du système se répartissent principalement en 2 sections : le front office et le back office. Le front office est principalement consacré a servir le client en provenance du web ou physiquement présent sur le point de vente. Le back office va organiser la production et la vente des produits dans ses différentes étapes.  
  
***Sur le front :***

**La consultation** : Le visiteur (connecté ou non) doit avoir accès au site et au choix des menus. Ce composant assure donc le service de navigation préalable à toute prise de commande.  
**L’authentification** : Ce composant à pour but d’identifier ou d’enregistrer le client sur le site pour lui assurer un service continu de la commande à la livraison.   
**La prise de commande** : Ce composant guide le client depuis le choix de sa commande (séléction) jusqu’à sa validation.  
**Le suivi de la commande** : Ce composant gère le flux de la commande client après sa validation.

***Sur back :***  
**Gestion des commandes** : Ce composant surveille le flux général des commandes côté back (monitoring).  
**Administration du front** : Ce composant permet la personnalisation du front, la mise à jour des menus, l’affichage des annonces, la promotion, voir le CRM etc.  
**Gestion des employés** : Ce composant planifie les attribution des taches au quotidien (monitoring).  
**Gestion des stocks** : Ce composant surveille l’état des stock, déclenche les alertes de réapprovisionnement, etc. (monitoring).

# Architecture de Déploiement

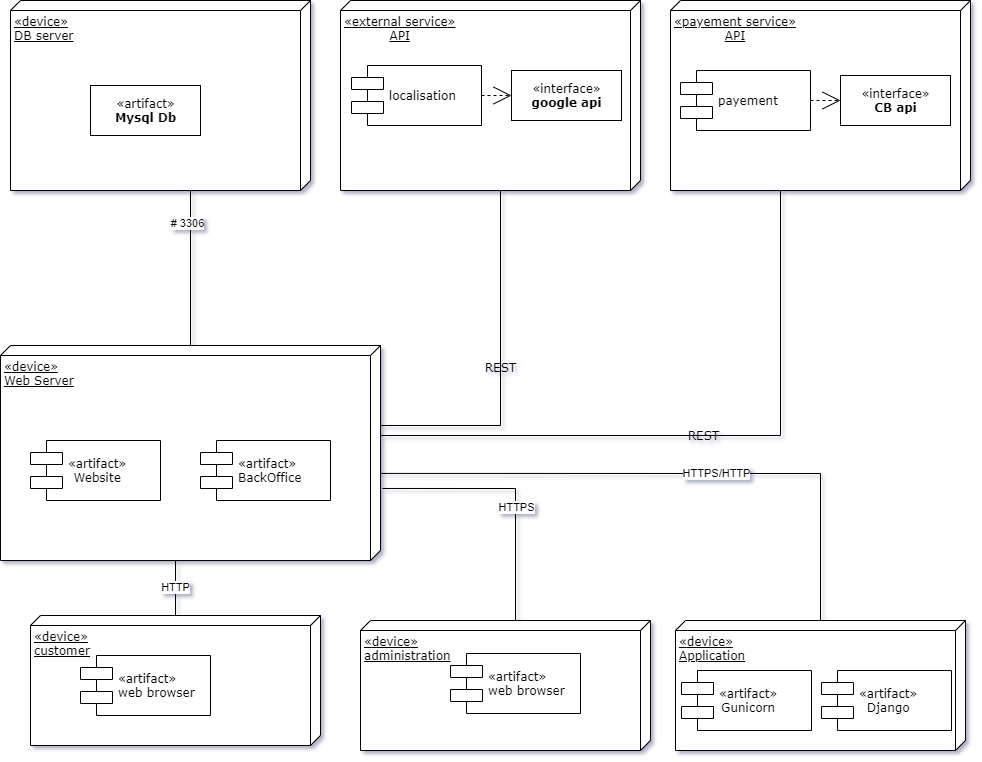


Diagramme UML de déploiement

Les éléments matériels physiques et leurs interconnexions :

- Serveur de base de données Mysql ou Maria DB pour la persistence des données.

- Navigateur (web browser) de l’ utilisateur/client se connecte par connexion http/https via ordinateur, téléphone ou tablette au serveur front office sur le service de commande.

- Navigateur (web browser) de l’administrateur se connecte par connexion https au serveur back office sur le service de gestion.

- Serveur web décentralisé ou frontal Nginx réceptionne les requêtes et les communique via socket

- Gunicorn interface WSGI receptionne les requêtes et les communique au framework Django

- Django , le framework dispatch la requête invoque les objets «callable».

- Un service python (cf le diagramme de composants décrivant sommairement les types de services disponibles) traite la requête et renvoie une réponse au navigateur.

## Serveur de Base de données

Descriptions & caractéristiques techniques :   
Serveur dédié Linux Debian Jessie + 100 Go SDRAM + MySQL 8.020

Informations importantes sur le jeu de test   
  
Le document d’installation des éléments présentés est disponible sur le repository Github à l’adresse suivante :  
 <https://github.com/jean-charles-gibier/OC_pizza/blob/master/doc/document_jeu_de_test.docx>  
Le script d’installation du schéma présenté est disponible sur le repository à l’adresse suivante :  
 <https://github.com/jean-charles-gibier/OC_pizza/blob/master/sql/init_db.sql>

Points particuliers

Parmi les suggestions, axes d’évolution du SI présenté :

La pile technique/materiel/ application sera avantageusement déployée sur un cloud provider tel que Azure ou Gogle cloud ou AWS.

Le serveur de base de donnée pourra être sécurisé par le déploiement d’un serveur de backup en mode slave sur une configuration relativement peu coûteuse.

# Glossaire

|  |  |
| --- | --- |
| **Nom** | **Définition** |
| API | Interface de programmation d’application ou interface de programmation applicative ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. |
| REST | Ensemble de contraintes à utiliser pour créer des services web. Les services web conformes au style d'architecture établissant une interopérabilité entre les ordinateurs sur Internet |
| ERP | « Enterprise Ressource Planning ». en français : PGI (Progiciel de Gestion Intégré) |
| MPD | Modèle Physique de Donnée schéma descriptif de l'implantation de la base de données dans un SGBDR. |
| Point de vente | Désigne l’emplacement de la chaîne ou les pizzas sont retirées par le livreur ou par le client Un point de vente est désigné par une adresse précise et/ou par ses coordonnées GPS. |
| Ingrédient | Composant de la recette de la pizza. |
| Commande | Une commande est composée de plats et boisson contenus dans la carte. |
| Carte | «La carte» est la carte du restaurant, elle est composée de plats et boissons . |
| Plat | Un plat désigne de façon générique un élément de la commande. Par exemple : pizza, entrée, dessert, ingrédients supplémentaires (à déterminer) Dans la version actuelle, le système les Plats n’intégreront que des pizzas. |
| Livraison | Déplacement entre un point de vente et une adresse de livraison (adresse client) Elle contiendra le nombre de Km et l’identifiant du livreur, l’identifiant du client Si l’adresse de livraison égale celle du point de vente, il s’agit d’un retrait sur place. |
| État de la commande | États possibles (et successifs) de la commande entre sa création et la confirmation de sa livraison. Énumération proposée:  «En sélection» => la commande initiée mais en cours de sélection,  «Commandée» => la commande est complète et attribuée à un id client,  «Payée» => la commande est payée,  «En préparation» => la commande est en préparation,  «En livraison» => la commande est en livraison,  «Livrée» => le livreur ou le client indiquent que la commande à été livrée,  «Annulée» => la commande est annulé (raison à préciser). |
| Recette | Liste d’ingrédients associés à une quantité et des conseils de préparation |
| Pizza | Résultat d’une recette préparée par le pizzaiolo. |
| Supplément | Ingrédient supplémentaire proposé au client dans l’exécution de la recette (ex : oeuf fromage) |
| Boisson | Consommation sélectionnée par le client ajoutée à la commande en cours |
| Payement | Transaction par CB ou en liquide qui déclenche la préparation et la livraison d’une commande. |
| Back office | Cf lot back office document de spécifications techniques |
| Stocks | Mesure des ingrédients disponibles dans un point de vente Mis à jour lors de la préparation des pizza et par le responsable lors de l’approvisionnement |
| Préparation | Processus générant la pizza => diminue le stock des ingrédients |
| Front office | Cf découpage lot «front office» document de spécifications techniques |
| Gestion clientèle | Regroupe toutes les actions en relation avec les clients et les commandes associées |
| Suivi | Succession d’états dans le cycle de vie d’une commande |
| Disponibilité | Disponibilité des ingrédients ou rupture de stock |
| Panier | Commande en cours de constitution |
| Dashboard | Tableau des tâches / commandes adressées au caissier, puis au pizzaiolo enfin au livreur. |