OC Pizzas

oc-pizzas.com

Dossier de conception technique

Version 1.0

Auteur
Gino Ladowitch
Directeur technique

Gino Ladowitch
https://gil-web.com

3-7 rue Albert Marquet – 06 60 48 06 54 – contact@gil-web.com
SCOPA au capital variable enregistrée au RCS de Paris – SIREN 448 762 526 – Code APE : 7022Z

Table des matières

1 - Introduction....3
1.1 - Objet du document....3
1.2 - Références....3
2 - Architecture Technique....4
2.1 - Composants généraux....4
2.2 - Composants généraux....4
2.2.1 - Schéma général....4
2.2.2 - Package account_manager....4
2.2.2.1 - Composant Authentification....4
2.2.2.2 - Composant Authentification....5
2.2.2.3 - Composant MailManager....5
2.2.3 - Package order_manager....5
2.2.3.1 - Composant PizzaManager....5
2.2.4 - Package content_manager....5
2.2.4.1 - Composant QueueManager....5
2.2.4.2 - Composant ManageRestaurant....5
2.2.4.3 - Composant StockManager....5
2.2.5 - Package stat_manager....5
2.3 - Diagramme de classes....6
2.3.1 - Diagramme....6
2.3.2 - Remarques sur le diagramme....7
3 - Architecture de Déploiement....8
3.1 - Serveur de Base de données....8
3.2 - Modèle physique de données....8
4 - Architecture logicielle....9
4.1 - Principes généraux....9
4.1.1 - Structure des sources....9

1 - Introduction

1.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application Python / Django OC Pizzas pour répondre aux fonctionnalités définies en amont dans le projet de création d'un site web de gestion de pizzeria pour le groupe OC Pizza.

1.2 - Références

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

1. **OC Pizzas - Dossier d'exploitation** : Dossier d'exploitation de l'application
2. **OC Pizzas - Dossier de conception fonctionnelle** : Dossier de conception fonctionnelle de l'application

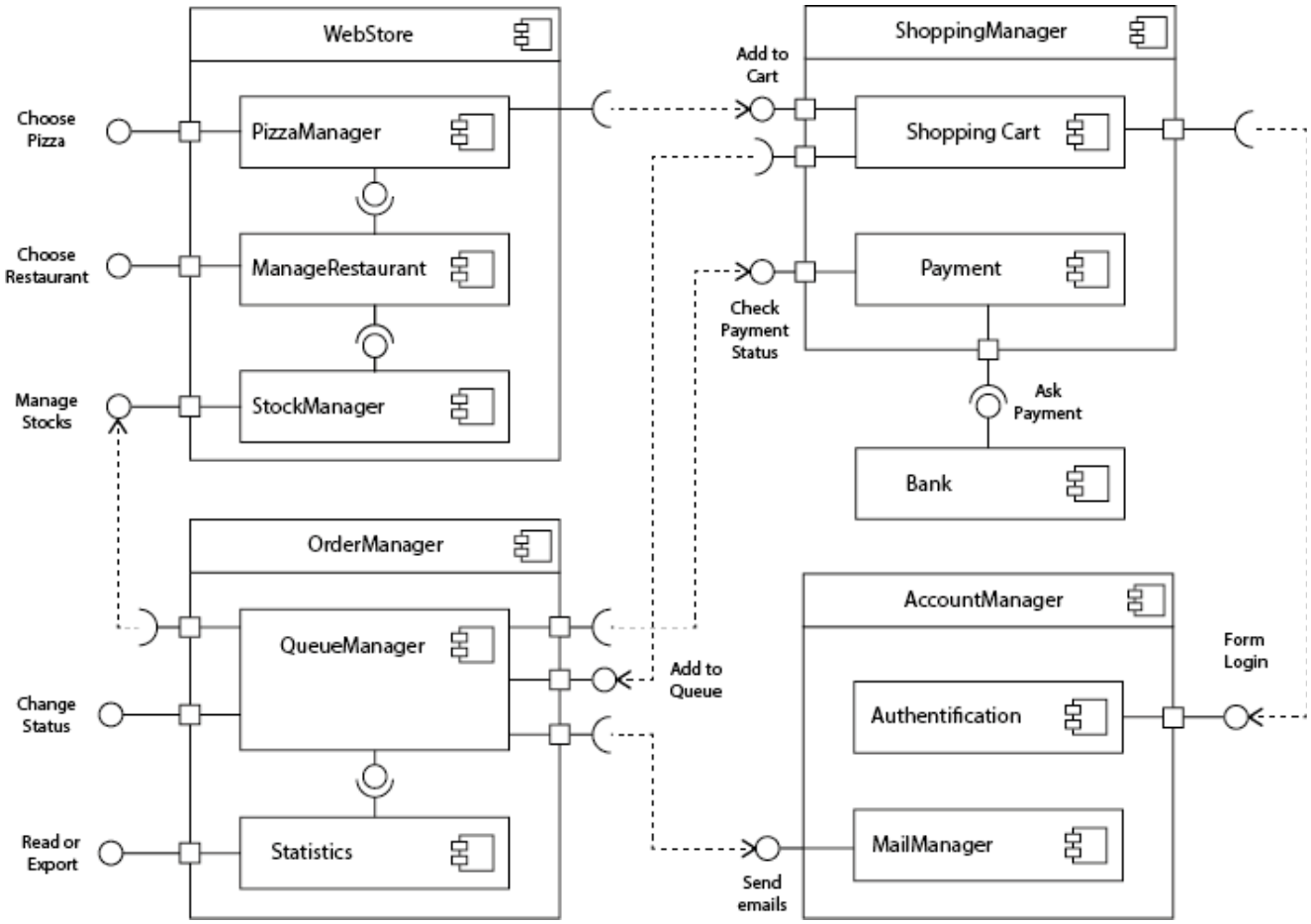
2 - Architecture Technique

2.1 - Composants généraux

L'application Django 1.11 classique, composée de 4 module fonctionnels décrits en détails ci-dessous et d'un module maître contenant les réglages principaux de l'application.

2.2 - Composants généraux

2.2.1 - Schéma général



2.2.2 - Package account_manager

2.2.2.1 - Composant Authentification

Description et rôle/objectif...

2.2.2.2 - Composant Authentification

Gère les vérifications nécessaires à l'authentification des utilisateurs clients et membre d'OC Pizzas.

2.2.2.3 - Composant MailManager

Gère l'envoi d'emails issus des processus d'authentification et de commandes.

2.2.3 - Package order_manager

2.2.3.1 - Composant PizzaManager

Gère la file de commandes.

2.2.4 - Package content_manager

2.2.4.1 - Composant QueueManager

Gère les pizzas et leurs composition (ingrédients)

2.2.4.2 - Composant ManageRestaurant

Gère les contenus restaurant.

2.2.4.3 - Composant StockManager

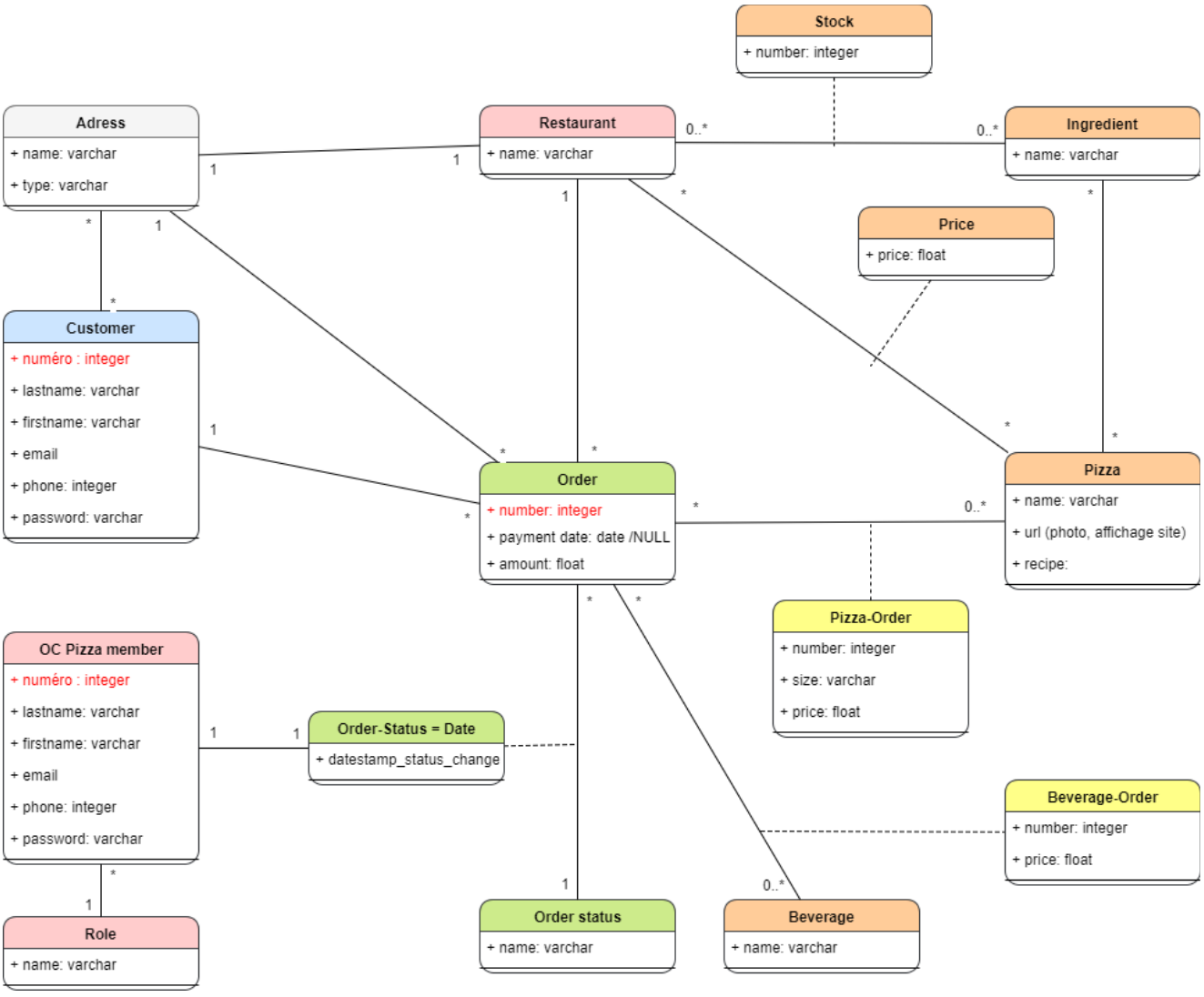
Gère la gestion des stocks (ingrédients).

2.2.5 - Package stat_manager

Interroge la base de données en vue de fournir un document téléchargeable des statistiques de un ou plusieurs restaurant (s) : ventes, stocks, cumul, etc. Comme défini dans le cahier des charges.

2.3 - Diagramme de classes

2.3.1 - Diagramme



Code couleurs :

Afin de faciliter la lecture des schémas, le code couleur suivant sera utilisé :

- Rouge : éléments propres à OC Pizza : gestion des restaurants et staff ;
- Orange : éléments liés aux produits alimentaires : pizzas, ingrédients, boisson, stocks d'ingrédients ;
- Vert : tous les éléments liés au processus de commande ;
- Bleu : éléments de gestion des comptes clients ;
- Gris : éléments utilisés dans divers contextes.

2.3.2 - Remarques sur le diagramme

Customer & OC Pizza Member :
Il s'agit de comptes classique de type utilisateurs (nom, prénom, email, password, etc.).
L'identification se fera par la combinaison email/password, aucun nom d'utilisateur ne parait nécessaire.
Il est fort probable que l'identification des clients et des membres d'équipes soit faits dans une nomenclature spécifique au groupe OC Pizza, qui garantit l'unicité, c'est pour cela que c'est directement le terme "number" qui est utilisé et non "id".
Les mots de passes seront hachés, donc très longs, 255 caractères sont prévus à cet usage.
Le téléphone est mémorisé dans un champ texte limité à 12 caractères, ce qui est suffisant pour retenir 10 chiffres plus deux d'indicatifs.
Pour des raisons de clarté dans la structure des données, les classes des clients et des membres du staff seront totalement séparées et pas héritées d'une classe mère telle qu'un utilisateur générique.
Dans ce modèle chaque membre de staff est associé à un seul restaurant, toutefois nous pouvons imaginer qu'un rôle (exemple : administrateur) permette d'accéder à tous le restaurants. C'est l'application qui permettra de le faire.

Pizzas :
Ce n'est pas précisé dans le brief, mais souvent les pizzas sont proposées en plusieurs tailles, ce sera indiqué dans un simple champ texte au niveau de la table PizzaOrder.

Les pizzas ont un prix fixe dans leur tables, mais dans la table PizzaOrder un champ sell_price permettra de gérer d'éventuelles réductions.

NB : afin de ne pas alourdir les données de test certains champs ont été omis, dans la réalité il y aura probablement plusieurs URL d'images, de même la recette a été traitée comme un simple champ texte, mais il pourrait être imaginable que ce contenu soit géré de manière plus poussée, et dans ce cas il y aurait une table vers laquelle pointerait une clé étrangère.

Addresses :
La relation Customer/Adress est many to many afin que chaque client puisse avoir plusieurs adresses, tel que "maison", "bureau", etc.
En revanche les restaurants n'ont qu'une seule adresse.

CustomerOrder & PizzaOrder & BeverageOrder :
La table portant la commande. Connectée à un client unique, une adresse unique et exactement un restaurant. En revanche chaque commande peut contenir plusieurs pizzas, boissons, et inversement pizzas et boissons seront dans plusieurs commandes à la fois.
Remarque : la relation pizza-customer est à 0 ou plus, mais dans la pratique – c'est l'application qui l'imposera – il ne sera certainement pas possible de commander sans au moins une pizza (exemple : uniquement des boissons).

OrderStatusChange :
Cette table va servir à tenir un historique des changements d'état de la commande, qui pourront être, par exemple : "pending", "in preparation", "prepared", "waiting", "running", "taken away", "delivered".

3 - Architecture de Déploiement

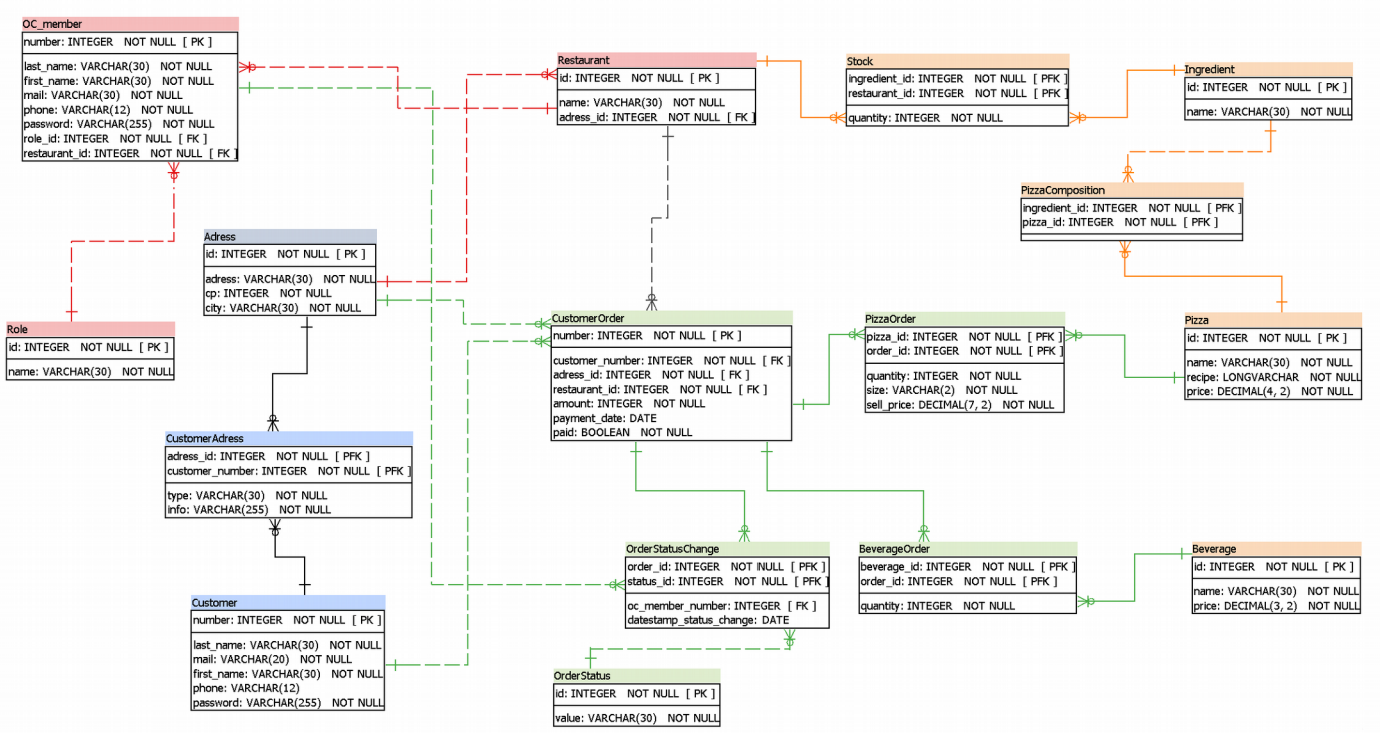
3.1 - Serveur de Base de données

Serveur de base de données hébergeant la base de données de l'application.

Système : Linux.

Logiciel : PostgreSQL v11 (supportée jusqu'au 09/11/2023)

3.2 - Modèle physique de données



4 - Architecture logicielle

4.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git**.

Les modules complémentaires sont contenus dans l'environnement virtuel de Python.

4.1.1 - Structure des sources

```
ROOT
├── account_manager
│   ├── migrations
│   │   └── __init__.py
│   ├── templates
│   │   └── account_manager
│   ├── __init__.py
│   ├── admin.py
│   ├── app.py
│   ├── forms.py
│   ├── models.py
│   ├── tests.py
│   ├── tokens.py
│   ├── urls.py
│   └── views.py
├── command_manager
│   ├── migrations
│   │   └── __init__.py
│   ├── templates
│   │   └── command_manager
│   ├── __init__.py
│   ├── admin.py
│   ├── app.py
│   ├── forms.py
│   ├── models.py
│   ├── tests.py
│   ├── tokens.py
│   ├── urls.py
│   └── views.py
├── content_manager
│   ├── migrations
│   │   └── __init__.py
│   ├── templates
│   │   └── content_manager
│   ├── __init__.py
│   ├── admin.py
│   ├── app.py
│   ├── forms.py
│   ├── models.py
│   ├── tests.py
```

```
-   │   ├── urls.py
-   │   └── views.py
-   │
-   ├── stat_manager
-   │   ├── migrations
-   │   │   └── __init__.py
-   │   ├── templates
-   │   │   └── content_manager
-   │   ├── __init__.py
-   │   ├── admin.py
-   │   ├── app.py
-   │   ├── forms.py
-   │   ├── models.py
-   │   ├── tests.py
-   │   ├── urls.py
-   │   └── views.py
-   │
-   ├── oc_pizzas
-   │   ├── settings
-   │   │   ├── __init__.py
-   │   │   └── production.py
-   │   ├── static
-   │   └── src
-   │       ├── main
-   │       │   ├── java
-   │       │   └── resources
-   │       └── test
-   │           ├── java
-   │           └── ressources
-   ├── .gitignore
-   ├── .travis.yml
-   ├── manage.py
-   ├── README.md
-   ├── requirements.txt
-   └── templates
```