I.S

OC-Pizza

Application Web

Dossier de conception technique

Version 1.0

Auteur
ihsan salman

Table des matières

1 - Versions

Auteur	Date	Description	Version
IH	05/08/2021	Création du document	V 0.1

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application web du client OC-pizza qui souhaite remplacer le système informatique actuel, il découle du précédent document Dossier de spécifications techniques.

Ce document a pour but de :

- modéliser le domaine fonctionnel et définir le modèle physique de données afin de concevoir la base de données du futur SI d'OC-Pizza
- représenter les interactions entre le systèmes et les composants internes et externes du SI
- Décrire le déploiement des composants
- Décrire l'architecture logicielle et ses modules

Les éléments du présents dossiers découlent :

- du besoin du client
- l'analyse des exigences du client et des implications de chaque demande

2.2 - Références

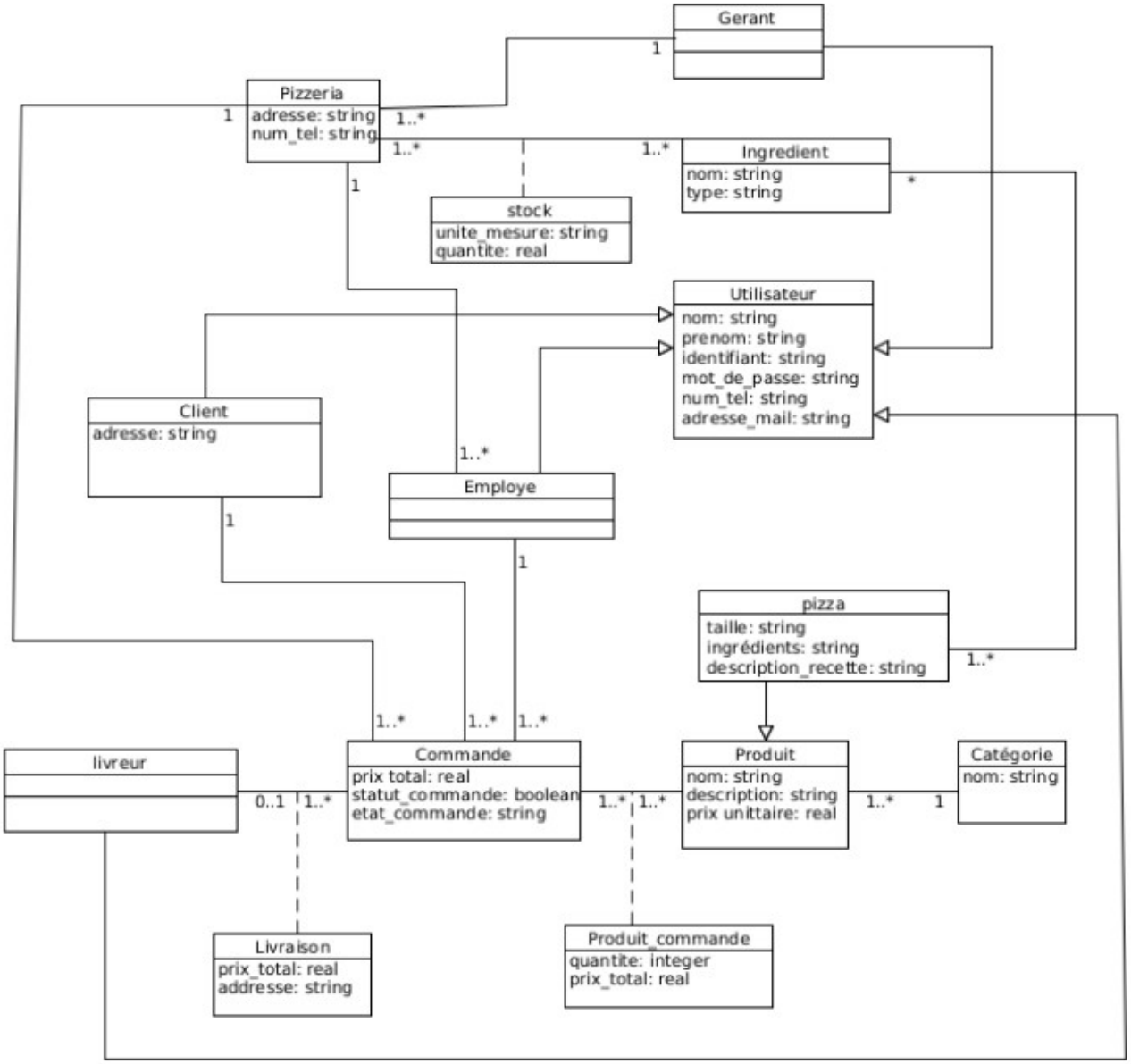
Pour de plus amples informations, se référer également aux éléments suivants:

1. **P9 - dossier de conception technique** : Dossier de conception technique de l'application
2. **P9 - dossier d'exploitation** : dossier d'exploitation de l'application
3. **P9 - Livraison** : Livraison de l'application

3 - LE DOMAINE FONCTIONNEL

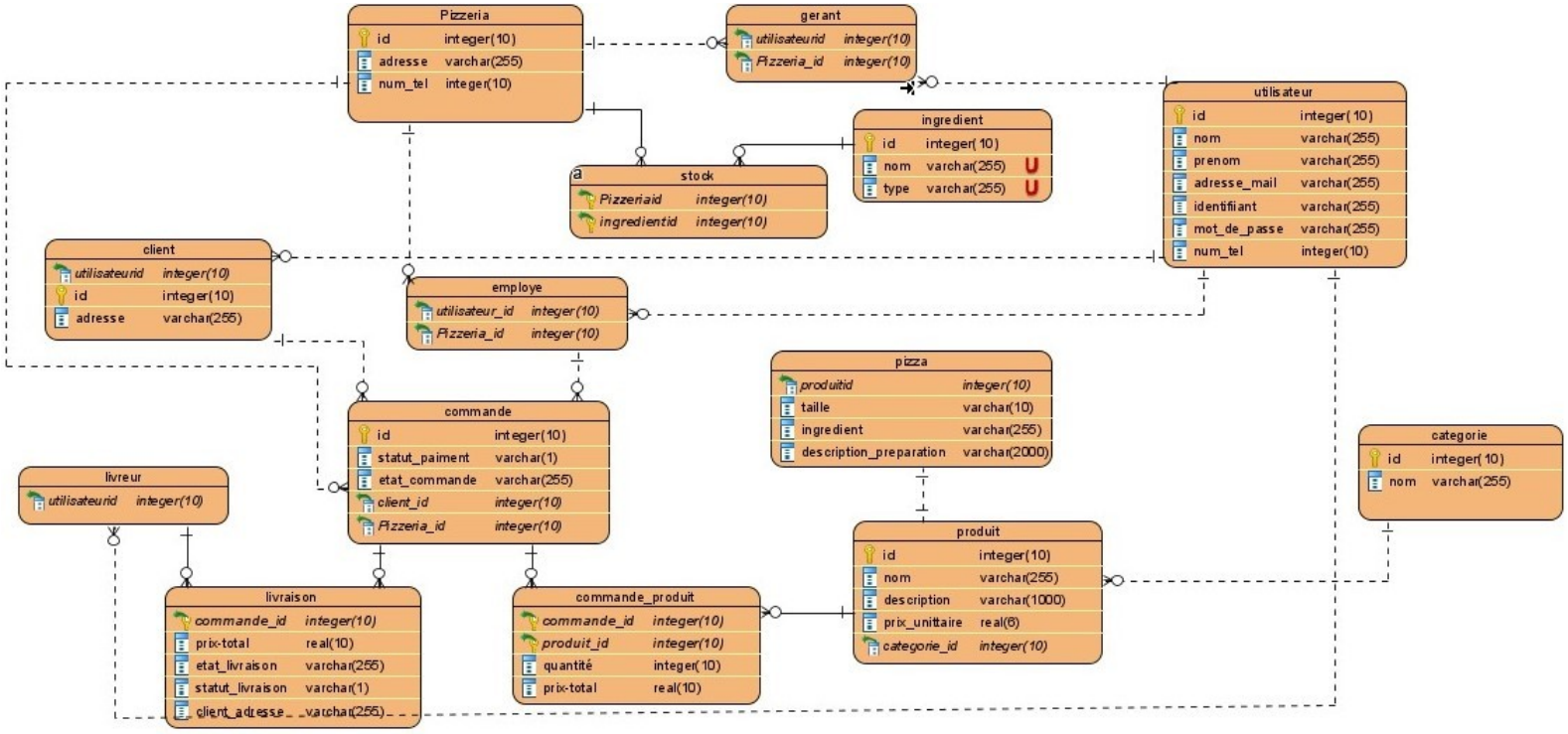
Le domaine fonctionnel permet en deux étapes de représenter la base de données sous forme de schéma. Tout d'abord le diagramme de classe introduit les éléments phares de la base de données en simplifiant grandement les relations afin de créer un dialogue simple et compréhensible entre le développeur et le client. Ensuite, une fois que le client valide les choix du diagramme de classe, le modèle physique de données est élaboré avec comme base le diagramme de classe. Les attributs de chaque classe se transposent en des tables de données avec des attributs plus détaillés facilitant l'élaboration de la base de données.

3.1 - Diagramme de classe



3.2 - Modèle physique de données

A partir du diagramme de classe, on peut détailler l'ensemble des éléments qui vont constituer la base de données. En effet, même si les relations sont simplifiées, le diagramme de classe fournit assez d'informations pour compléter le modèle physique de données. L'ensemble des éléments de la base de données y sont représentés.



3.3 - Descriptifs des classes et des tables

3.3.1 - Classe/Table Catégorie

La classe Catégorie recense l'ensemble des noms de catégorie de produits, avec un identifiant unique pour chaque nom.

3.3.2 - Classe/Table Pizzeria

La classe Pizzeria regroupe les informations de chaque enseigne du groupe OC-Pizza; l'adresse, le numéro de téléphone et un identifiant unique compose la classe afin de renseigner la localisation et contacter les différentes enseignes. Elle occupe notamment une place centrale dans le mise en place de la base de données.

Relations

Classe : gérant – Cardinalité : 1 à 1,*

Une pizzeria ne peut avoir qu'un seul gérant, mais un gérant peut être à la tête de plusieurs enseignes.
Classe : ingrédient – Cardinalité : 1,* à 1,*
Une pizzeria stocke forcément plus d'un seul ingrédient, et un même ingrédient peut se retrouver dans le stock de plusieurs pizzerias.
Classe : commande – Cardinalité : 1 à 1,*
Une commande est faite par une seule pizzeria ; mais une pizzeria représente forcément plusieurs commandes.
Classe : employé – Cardinalité : 1,* à 1
Un employé travaille dans une seule pizzeria, mais une pizzeria regroupe plusieurs employés.

3.3.3 - Classe/Table Utilisateur

La classe utilisateur donne les informations sur les utilisateurs de l'application ; notamment le nom, le prénom, l'adresse-mail, l'identifiant et le mot de passe pour la connexion, et un numéro de téléphone. Un identifiant unique permet de dissocier deux utilisateurs avec des informations identiques comme le nom et prénom.

Relations

Les classes gérant, employé, client et livreur héritent des attributs de la classe utilisateur, en tant qu'utilisateur de l'application.

3.3.4 - Classe/Table Produit

La classe produit regroupe les informations sur les produits de la pizzeria mis en vente, avec un nom, une description, un prix unitaire (taxes compris). Un identifiant unique permet de différencier les produits aux informations similaires et un identifiant est associé à la table catégorie.

Relations

La classe Pizza hérite des attributs de la classe produit,
Classe : Catégorie – Cardinalité : 1 à 1,*
Une catégorie représente plus d'un produit mais un produit appartient à une seule catégorie.
Classe : Commande – Cardinalité : 1,* à 1,*
Une commande est composée d'un ou plusieurs produit et un produit peut être dans plusieurs commande.

3.3.5 - Classe/Table d'associations

La classe stock est issu de l'associations des classes pizzeria et ingrédient, elle permet de donner les quantités de stock restant pour un ingrédient donné.
La classe livraison est issue de l'association des classes livreur et commande, elle donne les détails sur chaque livraison avec le prix total et l'adresse.
La classe produit_commandé est issu de l'association des classes commande et produit, elle précise la quantité de chaque produit commandé et le prix total.

3.3.6 - Le mot de passe

Concernant le mot de passe, il est crypté directement par une librairie python (crypt en mode SHA256) et que si sa longueur en texte est de maximum 30 caractères alphanumériques, le mot de passe crypté fait plus de 100 caractères de long. De plus, il est impossible de récupérer un mot de passe perdu, il faudra en recréer un !

4 - Diagramme de packages

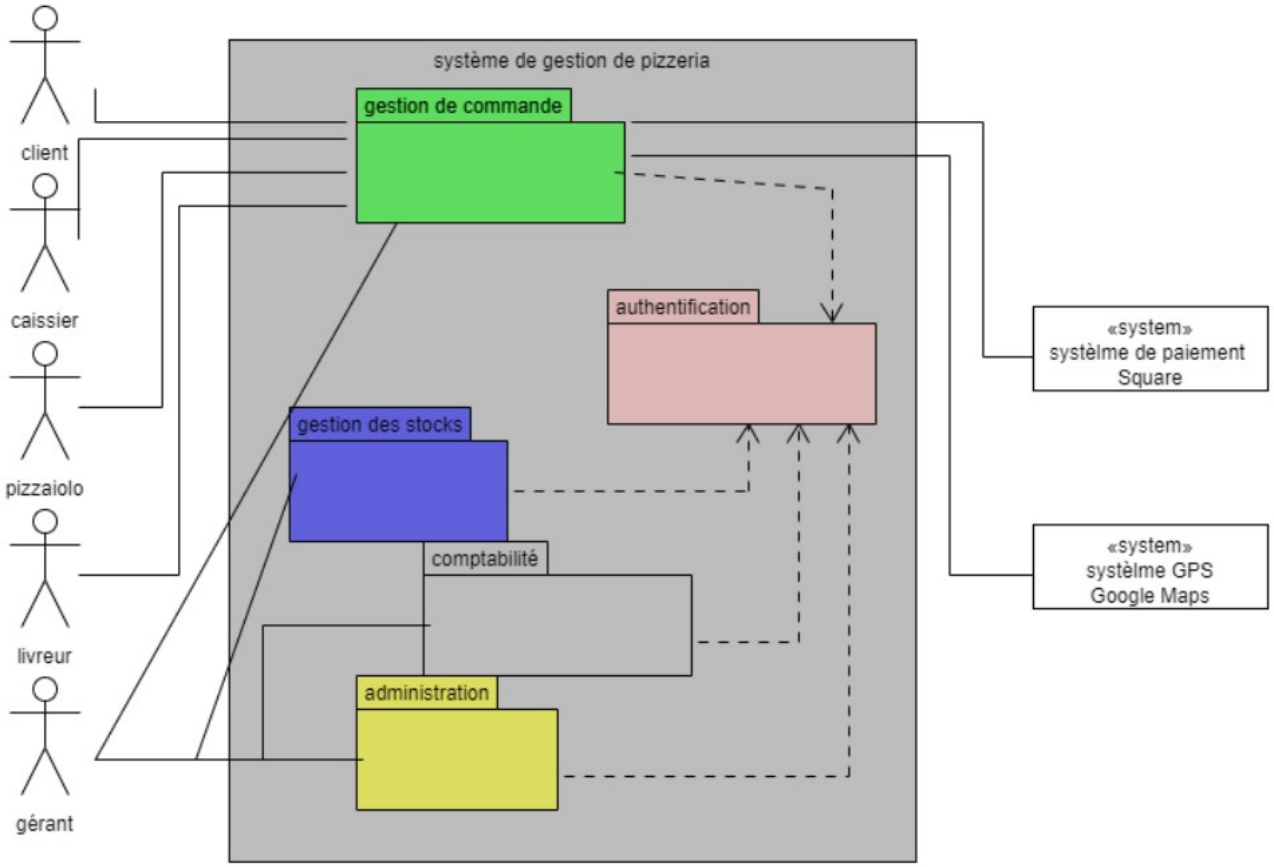


Diagramme de packages

Sur ce diagramme, on peut observer l'ensemble des fonctionnalités du SI. La gestion des commandes comprend de multiples autres actions réalisables par les utilisateurs notamment la prise de commande ou bien la livraison etc… Les autres packages représentent des fonctionnalités plus précises que seul le gérant pourra effectuer sauf l'authentification qui concerne l'ensemble des utilisateurs.

5 - Architecture Technique

L'application est de type web, fragmenté en plusieurs partie qui constitut le serveur de l'application, elle permettra la communication entre les utilisateurs et les composants externes du système.

Les composants externes sont schématisés par des diagrammes de composants, et les composants internes mettent en évidence les interfaces nécessaires. Il y a 2 deux composants externes :

- Google Maps pour la géolocalisation
- Square pour le paiement en ligne sur le site

5.1 - L'application WEB

L'application web sera une application web responsive, c'est-à-dire qu'elle devra supporter le changement de support de connexion (PC, tablette et smartphone). Cette application devra supporter notamment la navigation fluide de l'utilisateur pour répondre au besoin du client.
L'application sera composée des logicielles suivants :

-Système d'Exploitation : Linux Ubuntu 20.04
La distribution Linux Ubuntu existe maintenant depuis plus de 15 ans. Cette distribution est très utilisée tant pour les postes de travail que pour les serveurs. Les versions X.04 sont des versions Long Time Support et vous garantissent un support gratuit de la part de l'éditeur de 5 ans.

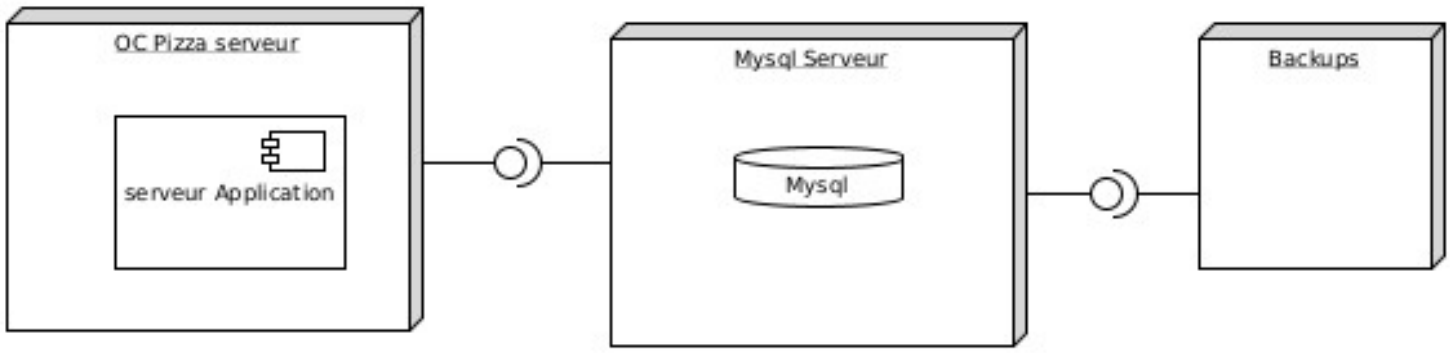
- Langage utilisé : Python 3.8
Le langage Python est très utilisé de nos jours pour sa rapidité et les nombreuses bibliothèques qu'il offre.

- Framework : Django 3.0
Django est une infrastructure d'application côté serveur extrêmement populaire et dotée de beaucoup de fonctionnalités, écrite en Python.

- Serveur Web : Apache 2.4
Le serveur web apache existe depuis 1995. Sa réputation n'est donc plus à faire et sa robustesse est bien connue dans le monde du Web. De plus, tous les modules, librairies ou autres fonctionnent sous Apache ce qui n'est pas toujours le cas de ses concurrents.

- Serveur de base de données : MySql
MySQL est un service de base de données entièrement géré pour déployer des applications natives du cloud en utilisant la base de données open source la plus populaire au monde.

5.2 - base de données



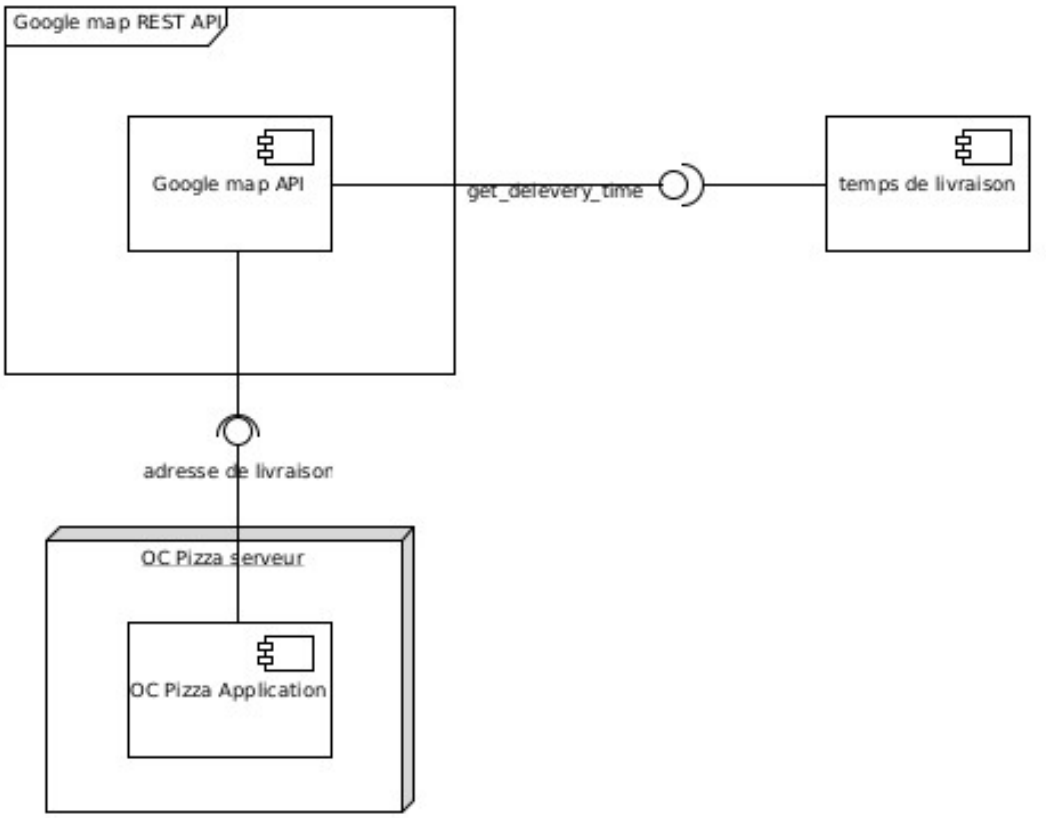
La base de données sera un serveur MySql et son système d'exploitation sera Linux. Les sauvegardes seront effectuées la nuit afin d'éviter de surcharger le serveur pendant son utilisation de la journée.
Afin de stocker les données, un ordinateur lambda suffit car les données à enregistrer ne représente pas une quantité énorme. On évite ainsi la perte de données si le datacenter rencontre des problèmes et les programmeurs pourront facilement accéder aux données afin de régler d'éventuels problèmes de sauvegardes. Les coûts de la base de données resteront minimes par rapport à un hébergement dans un datacenter.
Un système « Backup » permet aussi d'avoir une sauvegarde supplémentaire pour la base donnée afin de ne pas perdre définitivement les données enregistrer, une sécurité supplémentaire.

5.3 - Composants externes

Les composants externes nécessaire au système seront détaillés de manière plus précise dans les pages suivantes sous formes de diagramme de composant. Elles sont indispensables à la réalisation du projet tout en respectant le cahier des charges du client et le bon fonctionnement du système.

5.3.1 - Google Maps - Géolocalisation

Le serveur d'OC-Pizza fournit à l'API de Google Map l'adresse de livraison, ce dernier renvois le temps du parcours et le meilleur itinéraire pour la livraison.
La localisation des différentes adresses de la pizzeria ne demande pas de fournir des données à l'API, une carte sera directement consultable sur le site.
Le livreur pourra utiliser un GPS sur son téléphone afin d'effectuer la livraison.
Le diagramme ci-dessous illustre le fonctionnement de ce composant.



5.3.2 - Square – Paiement en ligne

Le règlement par internet doit répondre à des normes de sécurité et de confidentialité très importantes. Seuls quelques acteurs proposent une offre répondant à ces obligations tout en mettant à la disposition de leurs clients des outils de suivi, d'anti-fraude évolues et la possibilité d'avoir plusieurs moyens de paiement tels que les cartes VISA et MasterCard, PayPal, Applepay ou Googlepay, bitcoins…
L'API Square nécessite de mettre en œuvre un dialogue à trois (client – serveur OC-Pizza – serveur Square). Cet échange permet de garantir que la transaction correspond bien aux propriétés ACID et qu'elle s'est donc bien exécutée.

ACID : Atomicité – Cohérence – Isolation – Durabilité

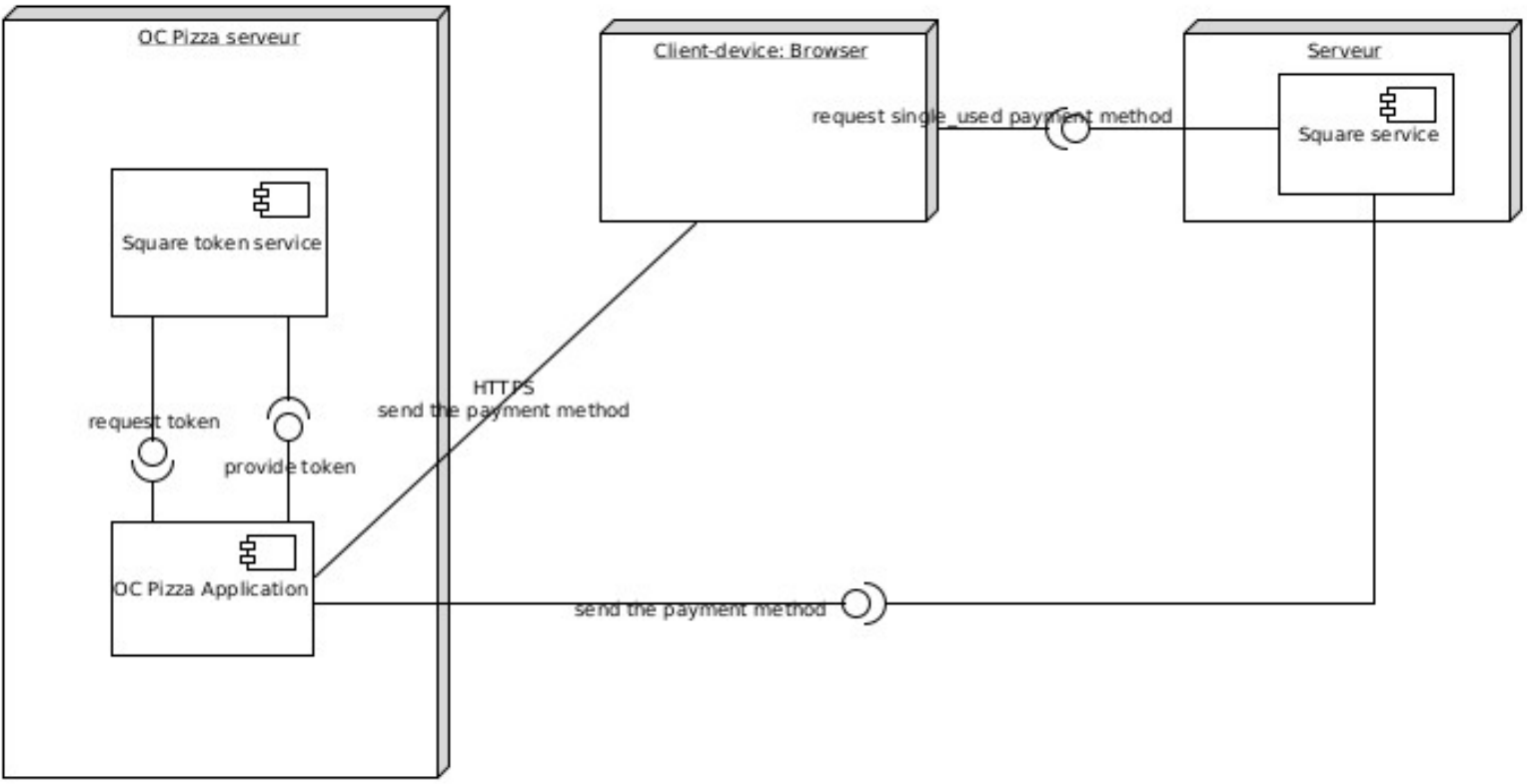
Atomicité : Garantie que la transaction soit faite complètement ou pas du tout.

Cohérence : L'état du système avant et après la transaction est valide

Isolation : La transaction ne dépend d'aucune autre.

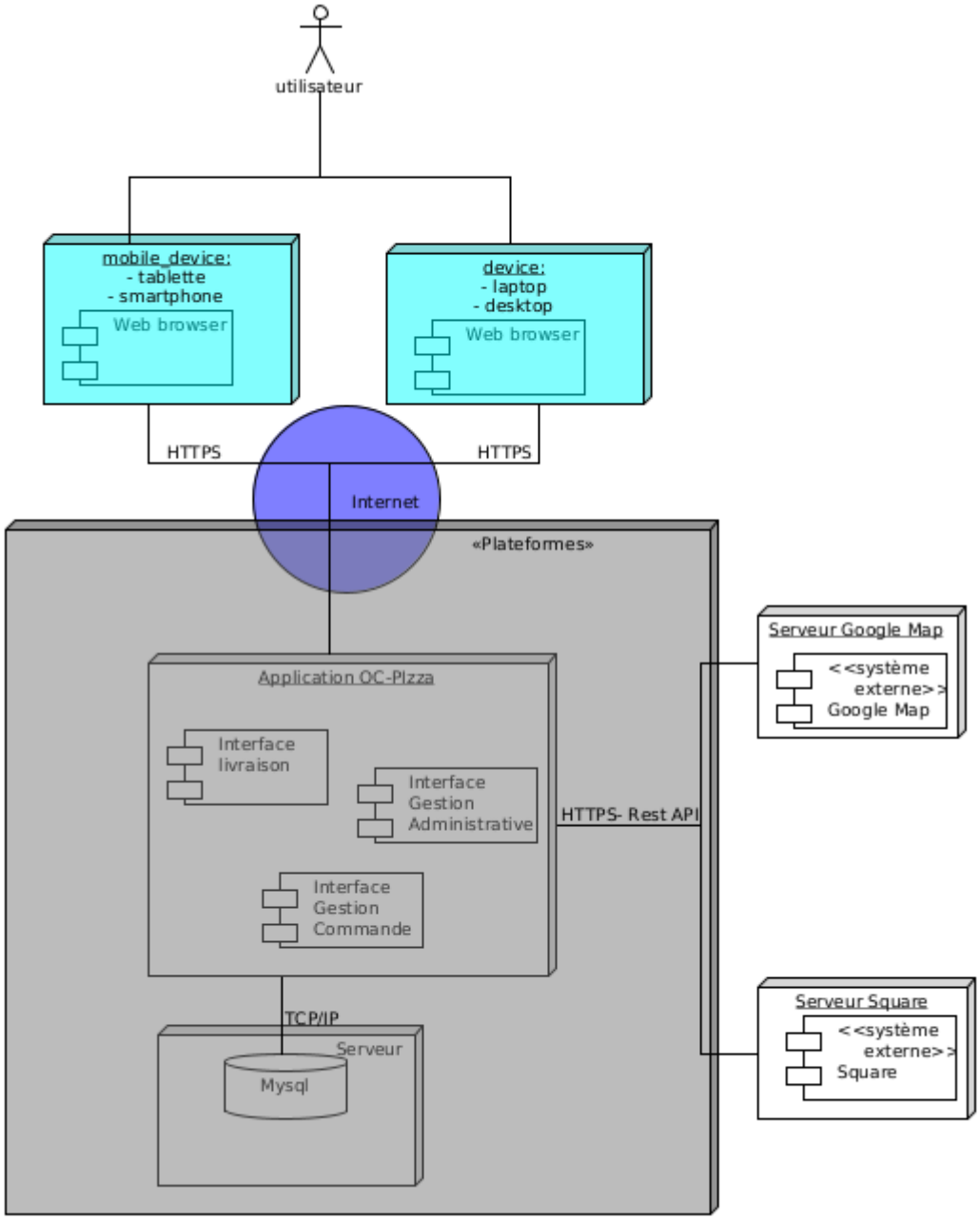
Durabilité : Garantie que la transaction a bien été enregistrée de façon permanente.

Le diagramme suivant illustre le principe de fonctionnement du composant.



Ce système permet donc de sécuriser la transaction de bout en bout et dispose d'un système anti-fraude élaboré.
L'API est largement documentée et Square met à disposition un SDK avec de nombreux exemples.

6 - Architecture de Déploiement



Le diagramme de déploiement uml permet de visualiser le déploiement de l'application sur le serveur, les interactions entre les acteurs et le système et les différentes connexions entre les composants.
Ce diagramme peut être divisé en trois partie :

- Les utilisateurs
- Serveurs
- Composants externes

6.1 - Les utilisateurs

Les utilisateurs pourront avoir accès au site de manière classique grâce au lien à partir d'un appareil ayant accès à internet et un navigateur standard ; donc les téléphones, les tablettes et les ordinateurs permettront de naviguer sur le site OC-Pizza.
Les vendeurs pourront avoir un ordinateur avec un clavier et un écran tactile pas trop grand afin de saisir facilement une commande par téléphone et pour pouvoir avoir une interaction forte avec le système. Les pizzaiolos pourront eux utiliser un écran tactile type tablette car ils n'auront rien à taper au clavier. Les livreurs devront eux utiliser un smartphone pour des simples question d'encombrement. Les administrateurs devront utiliser un écran plus grand pour pouvoir une meilleure vue de l'ensemble.

6.2 - Les serveurs

Le choix de la mise en place du serveur dépendra du client. Si le choix se porte sur un hébergement « clef en main » par un hébergeur du marché, il faudra alors séparer le serveur de base de données du serveur d'application afin que les données soient sécurisées directement par l'hébergeur.

7 - Architecture logicielle

7.1 - Principes généraux

Le code source de l'application est géré Git et les dépendances par pip de python.

Afin de structurer correctement l'application, nous allons séparé en 3 applications Django le programme principale :

- app_admin : gestion administrative
- app_commandes : gestion des commandes
- app_livraisons

En utilisant cette fonctionnalité de Django, le code de l'application sera facilement réutilisable lorsque le client voudra ajouter une nouvelle fonction à l'application ou étendre les possibilités marketing.

7.1.1 - Les couches

L'architecture applicative est de type MVT (model vue template) :

- la couche **Model** : Django possède la fonctionnalité particulière de gérer la base de données directement depuis les strict python par la création des modèles et des méthodes qui vont permettre d'utiliser les objets
- la couche **Vue** : Responsable du front-end, c'est-à-dire l'affichage de l'interface utilisateur, récupère les données de la couche model et la couche template afin de renvoyer une requête HTTP selon le besoin de la page internet chargé
- la couche template ; Responsable de l'affichage des données sur les pages du site internet.

7.1.2 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

- les répertoires sources sont crées de façon à respecter la philosophie Maven (à savoir : « convention plutôt que configuration »)

```
oc_pizza
├── app_admin                          (ici 3 applications donc 3x)
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── __init__.py
│   ├── migrations
│   ├── models.py
│   ├── static
│   │   └── app_admin
│   │       ├── css
│   │       │   └── style.css
│   │       ├── img
│   │       │   └── image.jpeg
│   │       └── js
│   │           └── script.js
│   ├── templates
│   │   └── app_admin
│   │       ├── home.html
│   │       └── mentions.html
│   ├── urls.py
│   └── views.py
├── manage.py
├── oc_pizza
│   ├── asgi.py
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── README.md
├── requirements.txt
├── settings.py
├── staticfiles
├── static
│   └── img
│       ├── favicon.png
│       └── logo
│
├── templates
│   ├── errors
│   │   └── errors.html
│   └── layouts
│       ├── base.html
│       └── partials
│           ├── _back.html
│           ├── _footer.html
│           ├── _nav.html
│           ├── _paginator.html
│           └── _scripts.html
```

8 - Points particuliers

8.1 - Ressources

Tous les éléments graphiques utilisés lors de la production de l'application OC-Pizza tels que les images, logo, icônes, couleurs et polices ont été fournies par le client avant de commencer la programmation.
Les esquisses des pages du site ont été déterminés à l'avance pour ne pas interférer avec la phase de développement.

8.2 - APIs

Les APIs comme expliqué dans la démarche de cette documentation à plusieurs reprises, permettent de renvoyer les données de l'utilisateur pour récupérer des informations précises que la fonctionnalité désiré doit afficher.

Deux APIs sont nécessaires à l'application :

- Google Maps API pour gérer la géolocalisation de l'utilisateur
- Square pour la gestion de transaction bancaire

8.3 - Environnement de développement

Le développement de l'application est distribué au sein de l'équipe qui se charge de gérer chaque besoin du client. Python fourni beaucoup de ressources gratuite pour mener à bien l'ensemble des tâches requises et réduit fortement la difficulté de la production avec Django comme Framework.

8.4 - Procédure de packaging / livraison

Lors de livraison de l'application, l'équipe se chargera de minimiser au maximum les tâches à effectuer pour le client afin de garantir la sécurité et le bon fonctionnement du projet. Dans cette documentation, le client aura assez de ressources pour faciliter la réutilisation ou le changement du code source.

Dès que le client aura réceptionner les données de connexion, il pourra commencer à utiliser le serveur comme souhaité.

9 - Glossaire

SI	Système informatique: ensemble des composants de l'unité de travail ou de consultation des données utilisé par l'utilisateur afin d'effectuer une tâche ou de recouvrir un besoin personnel.