OC-Pizza

Gestion de pizzeria

Dossier de conception technique

Version V1.0.0

Auteur
Faure Quentin
Analyste-programmeur

Table des matières

1 - Versions

Auteur	Date	Description	Version
Faure Quentin	19/04/23	Création du document	1.0.0

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application Gestion de pizzeria.

L'objectif du document est de décrire les choix techniques et les composants logiciels utilisés pour le développement de l'application. Il s'addresse aux développeurs, auc mainteneurs et à l'équipe technique du client.

Les éléments du présents dossiers découlent :

- de l'analyse des besoins de la maîtrise d'ouvrage,
- des choix de conception fonctionnelle qui ont été réalisés.

2.2 - Références

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

1. **DCF – Gestion de pizzeria** : Dossier de conception fonctionnelle de l'application.
2. **DE – Gestion de pizzeria** : Dossier d'exploitation.

3 - Architecture Technique

3.1 - Composants généraux

3.1.1 - Interface utilisateur

L'interface utilisateur permet aux utilisateurs de naviguer dans l'application: visualiser les recettes pour le personnel, les résultats de chaque pizzeria pour les gérants, ou passer des commandes pour les clients; Il est composé des langages HTML,CSS et JavaScript.

3.1.2 - Framework front-end

Un framework front-end est un ensemble d'outils et de bibliothèques qui permettent de développer et de structurer l'interface utilisateur d'une application web. Son objectif est de faciliter le développement et la maintenance d'interfaces utilisateur complexes, en fournissant des fonctionnalités telles que la gestion de l'état, le routage, la gestion des événements, la manipulation du DOM, etc. Les frameworks front-end offrent également une structure pour organiser les composants d'interface utilisateur, les styles et les scripts en modules réutilisables. De cette manière, les développeurs peuvent créer des interfaces utilisateur cohérentes et évolutives avec un minimum de répétition de code.

3.1.3 - Framework back-end

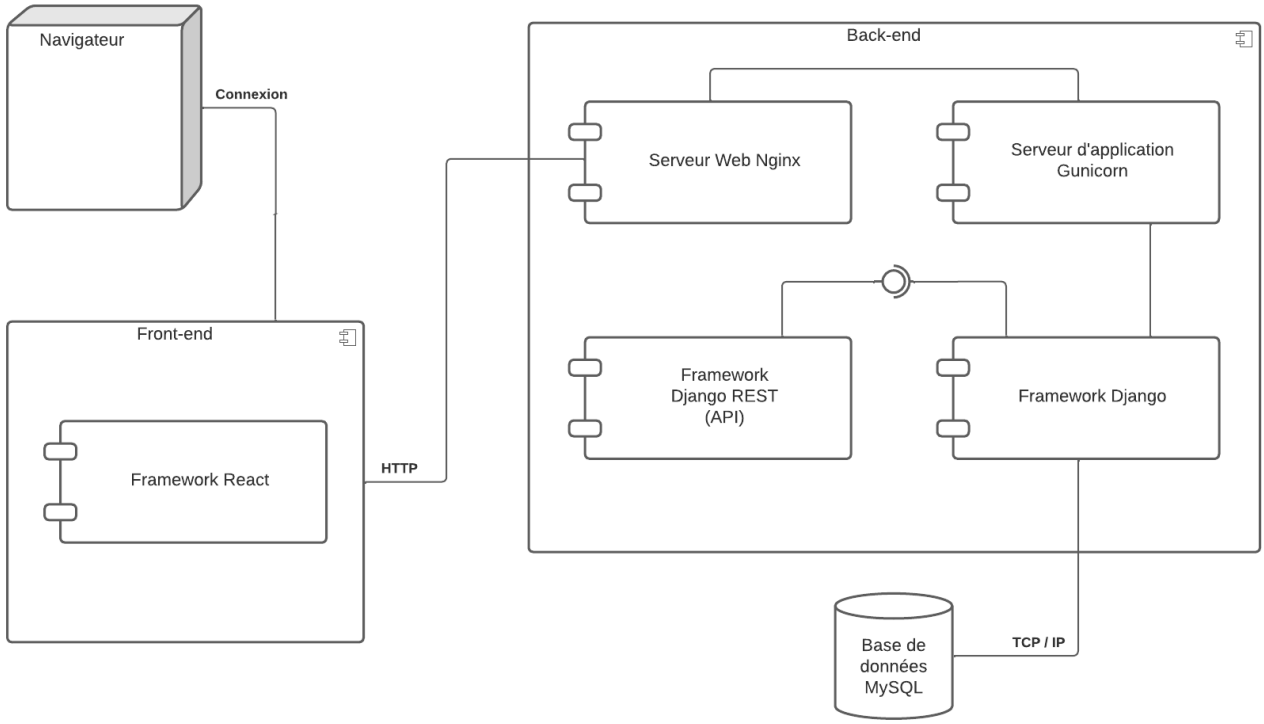
C'est un ensemble d'outils et de bibliothèques logicielles préconstruites qui permettent de créer des applications web en se concentrant sur le développement de fonctionnalités principales. Un framework back-end fournit souvent des outils pour faciliter la maintenance, le déploiement et l'extensibilité de l'application.

3.2 - Application Web

La pile logicielle est la suivante :

- Application **Python** (Version 3.11)
- Framework web : **Django** (Version 4.2)
- Framework front-end : **React** (Version 18.2.)
- Bas de données: **MySQL** (v8.0.33)
- Serveur d'application: **Gunicorn** (Version 20.1.0)
- Serveur web: **Nginx** (1.24.0)

Diagramme de composants



3.2.1 - Framework React

Le framework front-end React est utilisé pour construire les intrerfaces utilisateur dynqamiques de l'application.Il permet de créer des composants réutilisables et de les combiner pour construire des interfaces utilisateurs plus complexes.

3.2.2 - Framework Django

Framework web open-source en Python, Django est utilisé pour créer des application web complexes sur le modèle MVC, ce qui signifie que les applications sont organisées en trois parties distinctes: les modèles qui definissent les données, les vues qui définisssent la logique de l'application et les controleurs qui gèrent les intéractions entre les modèles et les vues.

Le framework web Django est utilisé pour créer l'architecture de l'application web. Il fonctionne en conjointement avec le framework front-end React pour créer une application web à page unique qui permettant une expérience utilisateur plus fluide et intéractive. Django offre des fonctionnalités telles que la gestion des URL, l'authentification utilisateur, la gestion de base de données, des sessions, etc. qui facilitent le développement et la maintenance de l'application web.

3.2.3 - Serveur d'application Gunicorn

Gunicorn est un serveur d'application Python open-source qui permet de déployer des applications Python sur un serveur web. Gunicorn est souvent utilisé avec des frameworks web comme Django ou Flask. Gunicorn est conçu pour être rapide, fiable et facile à utiliser. Il permet de gérer les connexions entrantes et sortantes de l'application web et d'optimiser les performances.

3.2.4 - Framework Django REST

Django REST est un framework qui permet de créer des API RESTful en utilisant Django. Les API RESTful permettent aux clients d'interagir avec une application web en utilisant des requêtes HTTP standard, telles que GET, POST, PUT, DELETE. Django REST fournit une couche d'abstraction pour faciliter la création d'API RESTful avec Django.

3.2.5 - Serveur web Nginx

Nginx est un serveur web open-source léger et rapide qui peut être utilisé comme serveur web principal ou en tant que serveur proxy. Nginx est souvent utilisé pour servir des fichiers statiques, des pages web dynamiques ou des applications web. Nginx est capable de gérer des connexions entrantes et sortantes et d'optimiser les performances de l'application web.

3.2.6 - Base de données MySQL

MySQL est un système de gestion de base de données relationnelles open-source. Il est utilisé pour stocker et récupérer des données à partir d'une application web. MySQL est très populaire en raison de sa fiabilité, de sa flexibilité et de sa rapidité. Il peut être utilisé pour stocker des données structurées ou non structurées. MySQL offre également de nombreuses fonctionnalités avancées telles que la réplication, la sauvegarde, la sécurité, etc.

4 - Architecture de Déploiement

Diagramme UML de déploiement

Explication / commentaires si nécessaires...

4.1 - Serveur de Base de données

Description : Le serveur de base de données héberge la base de données MySQL qui stocke toutes les informations relatives aux commandes, recettes, pizzerias, etc.

Caractéristiques techniques :

- Système d'exploitation : Ubuntu Server 20.04 LTS
- Version de MySQL : 8.0.26
- Capacité de stockage : 500 Go SSD
- RAM : 16 Go
- Processeur : Intel Xeon E5-2670

Informations importantes / points particuliers :

- La base de données est régulièrement sauvegardée sur un serveur distant pour éviter toute perte de données en cas de panne ou de défaillance du serveur principal.
- Des mesures de sécurité ont été mises en place pour protéger la base de données contre les attaques externes, telles que l'utilisation de pare-feu et de protocoles d'authentification sécurisés.

4.2 - Serveur d'application

Description : Le serveur d'application héberge l'application web et traite les requêtes des utilisateurs.

Caractéristiques techniques :

- Système d'exploitation : Ubuntu Server 20.04 LTS
- Version de Gunicorn : 20.1.0
- Version de Python : 3.9.7
- Capacité de stockage : 1 To SSD
- RAM : 32 Go
- Processeur : Intel Xeon E5-2670

Informations importantes / points particuliers :

- Le serveur d'application est configuré pour fonctionner avec le serveur web Nginx en utilisant une interface WSGI.
- Les requêtes des utilisateurs sont traitées de manière asynchrone pour garantir une meilleure performance de l'application.
- Des mesures de sécurité ont été mises en place pour protéger le serveur d'application contre les attaques externes, telles que l'utilisation de pare-feu et de protocoles d'authentification sécurisés.

5 - Architecture logicielle

5.1 - Principes généraux

Dans le cadre de ce projet, l'architecture logicielle est basée sur les principes suivants :

1. Séparation des préoccupations (Separation of Concerns) : chaque couche de l'application est responsable de tâches spécifiques, avec une logique de fonctionnement clairement définie.
2. Architecture orientée services (Service-Oriented Architecture) : l'application est construite autour de services, qui fournissent des fonctionnalités réutilisables à l'ensemble de l'application.
3. Utilisation d'une API RESTful : les services communiquent via une API RESTful pour assurer l'interopérabilité et la communication entre les différentes couches de l'application.

5.1.1 - Les applications

Le projet comprend les application fonctionnelles suivantes:

- une application de gestion des utilisateurs;
- une application de gestion des commandes;
- une application de gestion des pizzeria.

Le projet est organisé en modules Django, avec une structure en couches pour chaque module. Chaque module est construit autour de services spécifiques, qui fournissent des fonctionnalités réutilisables.

5.1.2 - Structure des sources

```
racine
├── backend
│   ├── authentication
│   │   ├── __init__.py
│   │   ├── admin.py
│   │   ├── apps.py
│   │   ├── models.py
│   │   ├── serializers.py
│   │   ├── urls.py
│   │   └── views.py
│   ├── command
│   │   ├── __init__.py
│   │   ├── admin.py
│   │   ├── apps.py
│   │   ├── models.py
│   │   ├── serializers.py
│   │   ├── urls.py
│   │   └── views.py
│   ├── pizzeria
│   │   ├── __init__.py
│   │   ├── admin.py
│   │   ├── apps.py
│   │   ├── models.py
│   │   ├── serializers.py
│   │   ├── urls.py
│   │   └── views.py
│   └── oc_pizza_management
│       ├── __init__.py
│       ├── asgi.py
│       ├── settings.py
│       ├── urls.py
│       └── wsgi.py
├── frontend
│   ├── public
│   │   ├── index.html
│   │   └── ...
│   ├── src
│   │   ├── App.js
│   │   ├── index.js
│   │   └── ...
│   ├── package.json
│   └── ...
├── static
│   ├── css
│   │   └── ...
│   ├── js
│   │   └── ...
│   └── images
│       └── ...
├── templates
│   ├── base.html
│   └── ...
├── .gitignore
├── .travis.yml
├── README.md
├── requirements.txt
└── manage.py
```

6 - Points particuliers

6.1 - Gestion des logs

L'application Django utilisera la bibliothèque Python "logging" pour gérer les logs. Les logs seront stockés dans des fichiers distincts en fonction de leur niveau de gravité et transmis à Sentry pour une meilleure gestion des erreurs. Sentry est un outil de surveillance et d'analyse des erreurs qui permet de détecter, diagnostiquer et résoudre les problèmes liés aux applications en temps réel.

6.2 - Fichiers de configuration

6.2.1 - Configuration du backend

Le backend contient plusieurs fichiers de configuration pour différents aspects de l'application.

6.2.1.1 - Fichier settings.py

Le fichier settings.py contient les paramètres de configuration principaux du projet, tels que la configuration de la base de données, les clés secrètes, les paramètres d'authentification et de sécurité, etc.

6.2.1.2 - Fichier urls.py

Les fichier urls.py contiennent la configuration des URLs de l'application, qui permettent de mapper les requêtes HTTP aux vues correspondantes.

6.2.1.3 - Fichier serializers.py

Les fichiers serializers.py contiennent les définitions de sérialiseurs pour les modèles de l'application pour la gestion de l'API REST.

6.2.2 - Configuration du frontend

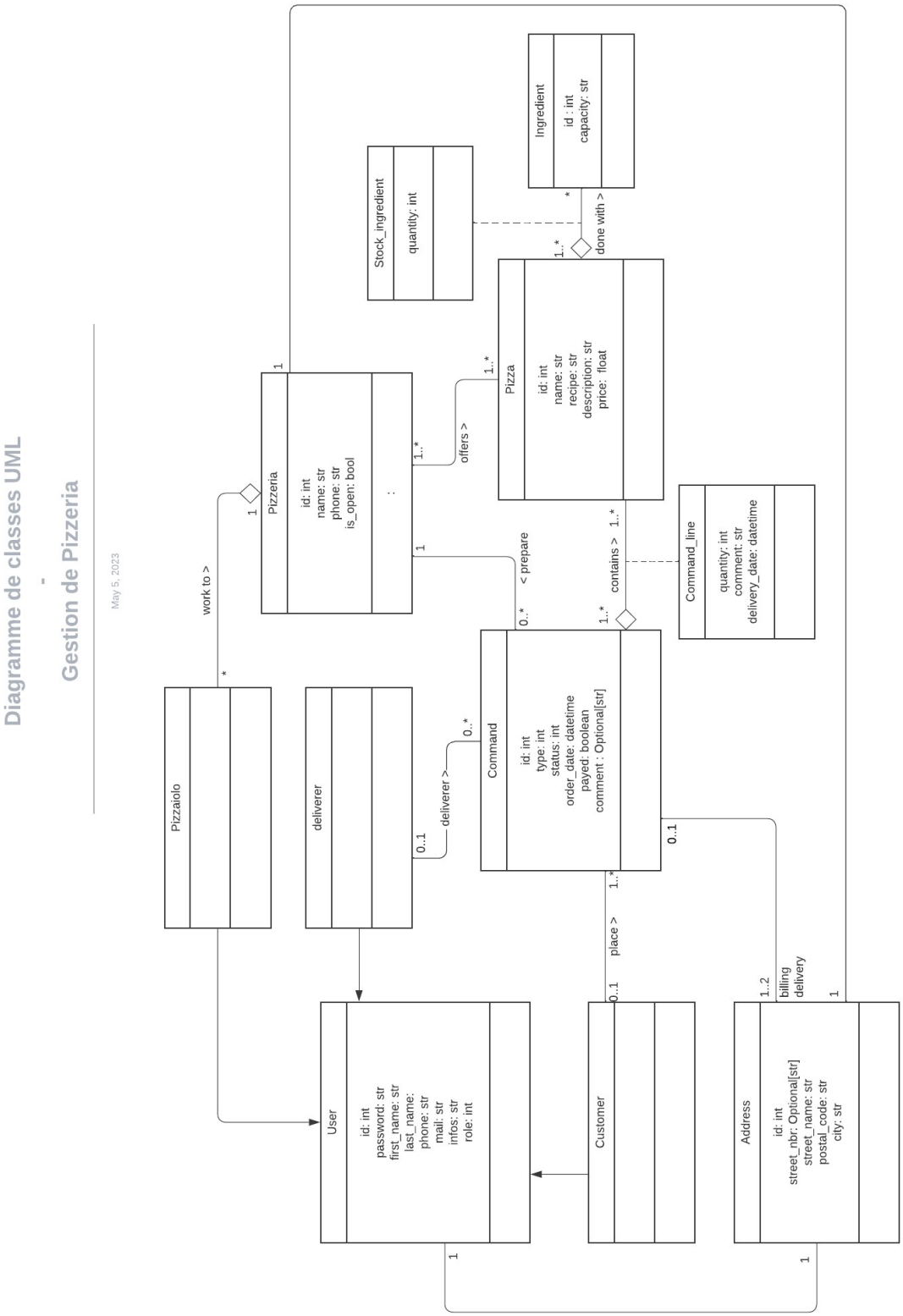
Le frontend contient également un fichier de configuration, situé dans le répertoire racine de l'application frontend.

6.2.2.1 - Fichier package.json

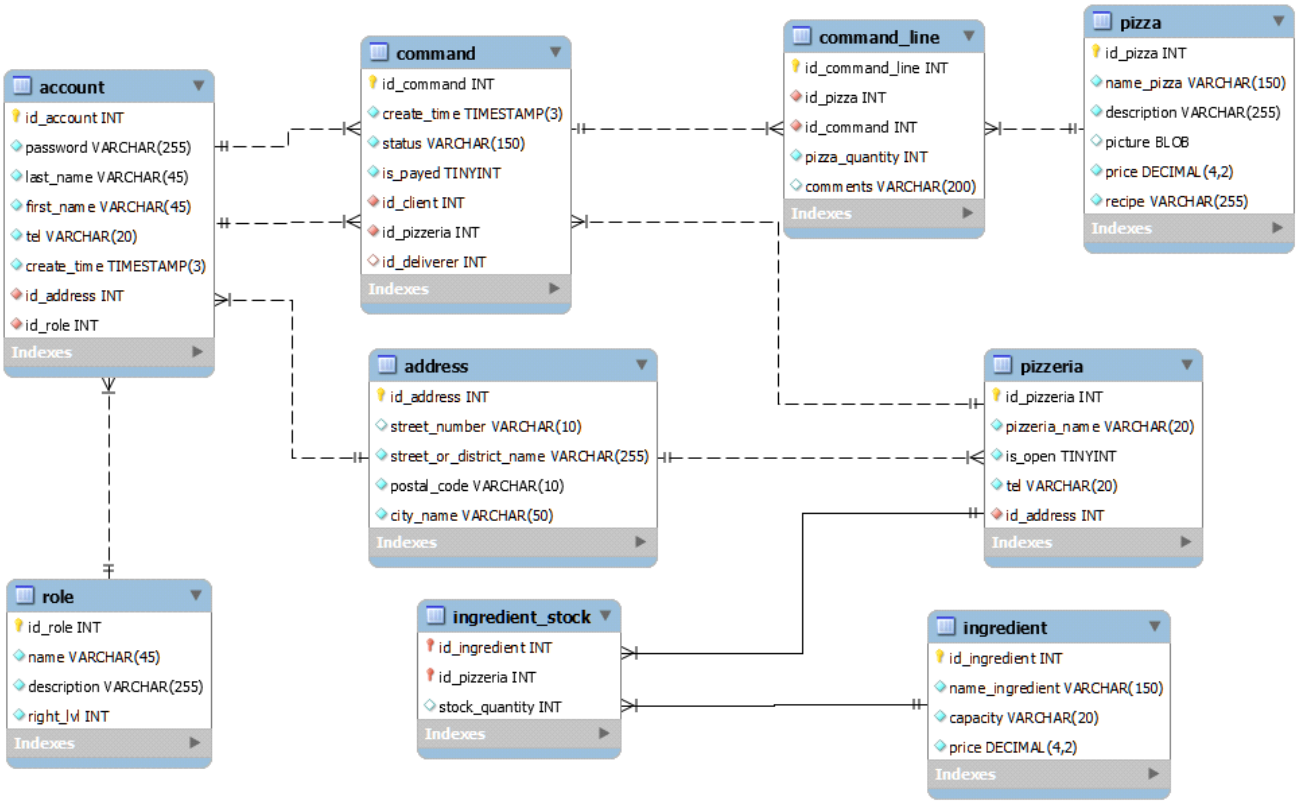
Le fichier package.json contient les dépendances et les scripts pour le développement et le déploiement de l'application frontend. On y définit également des variables d'environnement pour la configuration de l'application.

6.3 - Ressources

6.3.1 - Diagramme de classe



6.3.2 - Modèle physique de données



6.4 - Environnement de développement

L'application sera développée sur des machines locales à l'aide de l'environnement virtuel Python créé avec le module "virtualenv" et de l'éditeur de code VS Code. La gestion de versions sera assurée par Git et les modifications seront publiées sur GitHub. Les dépendances Python seront gérées à l'aide du fichier "requirements.txt" situé à la racine du projet. Pour le frontend, les dépendances seront gérées à l'aide de "npm".

6.5 - Procédure de packaging / livraison

Le processus de packaging et de livraison de l'application sera automatisé grâce à l'utilisation de Travis CI pour l'intégration continue. Les modifications apportées au code source seront automatiquement vérifiées et testées à chaque "push" sur le dépôt GitHub. Si les tests échouent, l'équipe de développement sera immédiatement informée par e-mail ou sur une plateforme de messagerie. Si les tests réussissent, Travis CI déclenchera automatiquement une procédure de déploiement sur l'environnement de production.

En cas de modification des dépendances du projet, les fichiers de configuration des dépendances (tel que package.json pour le frontend et requirements.txt pour le backend) seront automatiquement mis à jour lors de l'intégration continue avec Travis CI. Cela garantira que les environnements de développement, de test et de production sont toujours à jour avec les mêmes dépendances.

7 - Glossaire
