|  |
| --- |
| **OC Pizza**  Système de gestion pour pizzeria  Dossier de conception technique  Version 1.2 |
| **Auteur**  Mathieu Lemaire  *Développeur d’application Android* |

Table des matières

1 -Versions 3

2 -Introduction 4

2.1 -Objet du document 4

2.2 -Références 4

3 -Architecture Technique 5

3.1 -Composants généraux 5

3.1.1 -Package A 5

3.1.1.1 -Composant X 5

3.1.1.2 -Composant Y 5

3.1.2 -Package B 5

3.1.2.1 -Composant Z 5

3.2 -Application Web 5

3.2.1 -Composants X 5

3.2.2 -Composants Y et Z 5

3.3 -Application XXX... 5

4 -Architecture de Déploiement 6

4.1 -Serveur de Base de données 6

4.2 -Serveur XXX 6

5 -Architecture logicielle 7

5.1 -Principes généraux 7

5.1.1 -Les couches 7

5.1.2 -Les modules 7

5.1.3 -Structure des sources 7

5.2 -Application Web 8

5.3 -Application Xxx 8

6 -Points particuliers 9

6.1 -Gestion des logs 9

6.2 -Fichiers de configuration 9

6.2.1 -Application web 9

6.2.1.1 -Datasources 9

6.2.1.2 -Fichier xxx.yyy 9

6.2.2 -Application Xxx 9

6.3 -Ressources 9

6.4 -Environnement de développement 9

6.5 -Procédure de packaging / livraison 9

6.6 -XXX 9

7 -Glossaire 10

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Mathieu  Lemaire | 25/09/2025 | Création du document | 1.1 |
| Mathieu  Lemaire | 27/09/2025 | Modification du document | 1.2 |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Objet du document

Le présent document constitue le dossier de conception fonctionnelle de l'application OC Pizza pour la maîtrise d’ouvrage (MOA) et de la maîtrise d’œuvre (MOE).

Objectif du document est de définir le fonctionnement et les fonctionnalités du système, en tenant compte des objectifs de celui-ci et les besoins de ses utilisateurs.

Les éléments du présent dossier découlent :

* D’un premier entretient de présentation des spécifications fonctionnelles produites par IT Consulting & Development
* D’un second entretient de présentation de solution technique

## Références

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

1. **Projet 10 – Dossier de conception technique** : Dossier de conception technique de l'application
2. **Projet 10 – Dossier d’exploitation** : Dossier d’exploitation de l’application
3. **Projet 10 – PV de Livraison** : Procès-verbal de la livraison finale

# Architecture Technique

## Composants généraux

La modélisation et l’identification des éléments composant le système d’information à mettre en place et leurs interactions.

### Package Services

#### Google Maps Geolocation API

L’API de géolocalisation renvoie un rayon autour du client qui permet de récupérer avec précision, en réseau cellulaire ou en réseau WIFI, les informations qu’il souhaite récupérer. Dans ce cas précis c’est une solution pour permettre de trouver les restaurants le proche et aussi récupérer sa localisation pour une livraison éventuelle.

#### Google Maps Directions API

L’API Directions est un service Web qui utilise une requête pour renvoyer des directions / trajets à l’appareil et l’emplacements que l’on souhaite sélectionner. Plusieurs moyens de transport sont disponibles.

Cette API permettra l’acheminement des commandes en livraison et do connaître la position du livreur en temps réel.

### Package Back Office Restaurant

Ce package à pour but de gérer le restaurant.

#### Order Manager

Gestion des commandes qui ont été passées, avec les informations client, le statut en temps réel.

**3.1.2.2 – Administration**

Une interface Web permettant aux employés la prise et la gestion de commandes. Il permet aussi gérant de récupérer des statistiques clients et d’avoir un suivi complet sur la gestion du restaurant.

### 3.1.3 – Package Payment

Ce package regroupe la gestion des différents moyens de paiement de l’application.

3.1.3.1 – API Bank

Une API de paiement en ligne est un service qui permet la transmission des informations de paiement d’un client à l’application. Cette API a pour but de vérifier si les informations sont conformes et de validé le paiement.

## Application Web

La pile logicielle est la suivante :

* Serveur d'application sera TomCat Embedded fourni automatiquement par Spring Boot

### Composants Base de données - PostgreSQL

La base de données est sur un serveur PostgreSQL, qui permettra la gestion des données de l’application en back end. Son rôle sera le stockage de des données utilisateurs et le stockage des données des restaurants que ce soit pour l’identification ou la gestion.

### Composants Front end – Angular

Pour la gestion Front End ce sera Angular. Cela va permettre d’avoir une interface qui sera adaptée à chaque rôles utilisateurs. Bien sûr une personne travaillant dans le restaurant aura plus de fonctionnalités qu’un client.

*3.2.3 – Composants Back End – Java*

Pour la gestion Back End, ce sera principalement la gestion des actions utilisateurs et le fonctionnement de l’application avec sa structure et ses différents modules. Ceci va permettre l’envoi des demandes que l’utilisateur souhaite avoir et renvoyer les informations à l’utilisateur via l’affichage.

Java 21avec le Framework Spring Boot 3.5 permet de répondre aux besoins.

## Application Mobile

Pour la partie application mobile, ce sera développé en React Native pour permettre un développement cross-plateforme (iOS + Android) avec une seule base de code.

# Architecture de Déploiement

Diagramme UML de déploiement

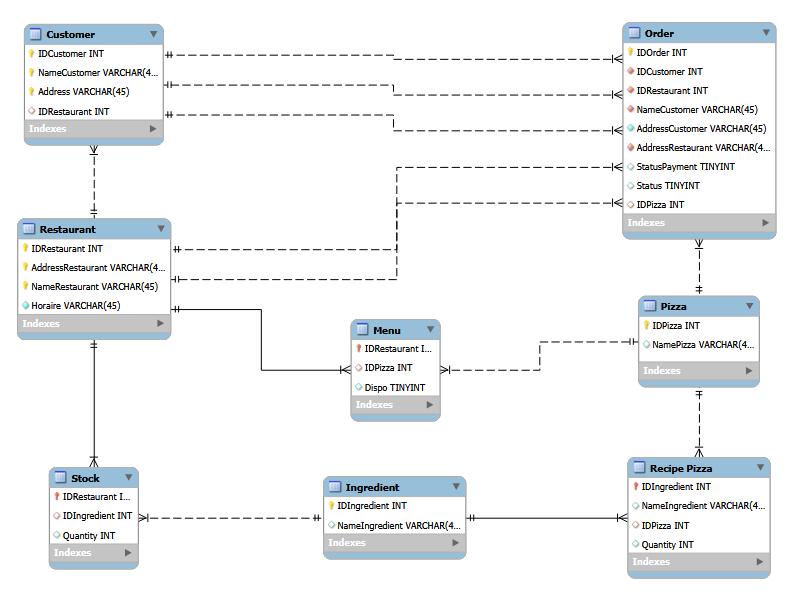
Explication / commentaires si nécessaires...

## Serveur de Base de données

Caractéristiques techniques :

* Ubuntu Server 22.04 LTS
* PostgreSQL 15.6 (Plus stable)

Les données seront stockées sur une base de données PostgreSQL hébergé en ligne. Voici le schéma du modèle physique de base de données prévu pour la solution technique.



# Architecture logicielle

## Principes généraux

L’hébergement et le management des versions du projet seront gérées sur Github. De plus, les dépendances ainsi que le packaging se feront via Apache Maven. Pour la persistance de données en base de données utilisation de Supabase Storage.

La solution s’articulera autour de plusieurs modules autonomes exposants des API Rest utilisables par les autres modules de l’application.

Les modules seront documentés via Swagger. Cette interface permettra aux développeurs de tester l’application.

### Les couches

L'architecture applicative est une architecture MVVM :

* Unecouche **Model** : responsable de la logique métier du composant et de la manipulation des données en base de données.
* Unecouche **View** : responsable de l’interface perçue par l’utilisateur, elle est la couche qui interagit directement avec l’utilisateur en recevant ses actions.
* Une couche **ViewModel** : responsable de gestion entre la vue et le modèle. Il contient la logique de présentation de l’application, transforme les données du modèle en un format que la vue peut facilement afficher et il interprète les actions de la vue pour la transmettre au modèle.

### Les modules

Ex: modules Maven dans le cas d’application multi-module...

### Structure des sources

La structuration des répertoires du projet suit la logique suivante :

/src

├── /assets

│ ├── /images # Images pour les pizzas, les points de vente, etc.

│ ├── /icons # Icônes (ex : panier, profil, etc.)

│ └── /styles # Styles globaux, couleurs, typographies

├── /components

│ ├── /Button # Composant bouton réutilisable

│ ├── /PizzaCard # Composant pour afficher une pizza dans une liste

│ └── /Header # Composant pour l'en-tête (menu, logo)

├── /models

│ ├── /PizzaModel # Modèle pour la gestion des pizzas

│ ├── /StoreModel # Modèle pour la gestion des points de vente

│ ├── /OrderModel # Modèle pour la gestion des commandes

│ └── /db.js # Fichier pour gérer l'accès à PostgreSQL

├── /services

│ ├── /api # Services pour les appels API

│ └── /auth # Services pour l'authentification

├── /views

│ ├── /HomeScreen # Écran d'accueil avec les pizzas populaires

│ ├── /StoreScreen # Écran des détails d'un point de vente

│ ├── /CartScreen # Écran du panier pour passer commande

│ └── /LoginScreen # Écran de connexion

├── /config

│ ├── dbConfig.js # Configuration de la connexion à PostgreSQL

└── /test # Tests unitaires et d'intégration

# Points particuliers

## Gestion des logs

Concernant la gestion des logs, d’après notre architecture nous pouvons utiliser une solution comme ELK Stack. Cette solution permet de centraliser les logs et les actions des différents utilisateurs avec Tomcat Embedded et PostgreSQL, selon le niveau de criticité défini.

## Fichiers de configuration

La description de l’ensemble des fichiers de configurations cités ci-dessous est décrite dans le Dossier d’exploitation

### Application web

Web.xml (Section du fichier Dossier Exploitation)

### PostgreSQL

Pg\_ident.conf

Pg\_hba.conf

Postgresql.conf

*6.2.3 – Tomcat Embedded*

server.xml

web.xml

context.xml

## Ressources

<https://www.postgresql.org/download>

<https://tomcat.apache.org/download-10.cgi>

Serveur pour la base de données PostgreSQL :

* 8-16 Go RAM
* Disque SSD supérieur à 100Go (Plus rapide)

Service externe pour le système de paiement :

* Stripe / Paypal (Pas de serveur à gérer, utilisation via leur API)

## Environnement de développement

Pour le développement de l’application Intellij IDEA (Backend Java)

PostgreSQL pour la gestion de base de données

VS Code pour le développement Front End

Docker pour la simplification des tests et le déploiement

## Procédure de packaging / livraison

Le packaging est fait par Maven. La procédure de packaging est alors lancée avec la commande « mvn clean package », il en résultera un fichier exécutable java « .jar ».

Le déploiement se fera avec la commande « mvn clean install » qui permettra de supprimer les anciennes compilations de la classe java ainsi que les fichiers ressources.

# Glossaire

|  |  |
| --- | --- |
|  |  |
|  |  |