IT Consulting & Dev

OC Pizza

Système informatique OC Pizza

Dossier de conception technique

Version 1.1

Auteur
Jacques Colin
Analyste-programmeur

OC Pizza

IT Consulting & Dev

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Jacques Colin	13/08/2020	Création du document	1.0
Jacques Colin	26/08/2020	Ajout d'un complément d'information	1.1

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application OC Pizza.

Ce document a pour but de présenter les technologies et architectures mises en œuvre afin de développer l'application OC Pizza.

2.2 - Références

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

1. **DCF – 1.1**: Dossier de conception fonctionnelle de l'application
2. **DE – 1.1**: Dossier d'exploitation de l'application

3 - Architecture Technique

3.1 - Composants généraux

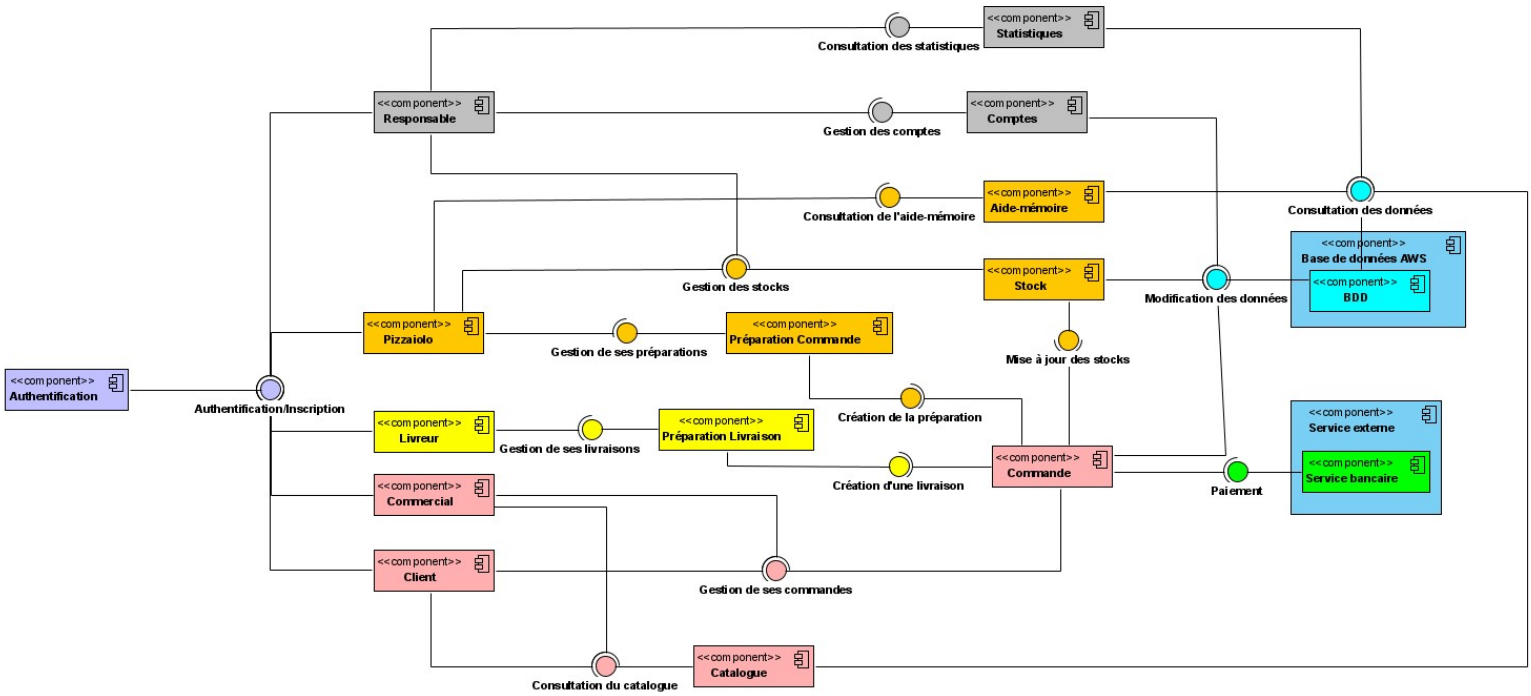


Diagramme de composants

Le commercial et le client peuvent consulter le catalogue en vue de créer une commande, et gérer ses commandes pour les consulter , annuler ou modifier.
Le livreur et le pizzaiolo peuvent gérer leurs préparations respectives. Le pizzaiolo peut consulter son aide-mémoire en cas d'oubli.
Le responsable peut consulter les statistiques et gérer les stocks.
Une commande peut procéder au paiement via le service bancaire, créer une préparation pour le pizzaiolo et pour le livreur, mettre à jour les stocks enfin modifier les données dans la base de données.

3.2 - Application Web

La pile logicielle est la suivante :

- Application **Angular 10.0.0** / **PostgreSQL 12.4 / Java 8**
- Serveur d'application **AWS**

3.3 - Application Android

Application en Java 8 ne servant qu'à faire une redirection vers l'interface web.

L'application contient une Webview sans ActionBar, sera disponible en format portrait et paysage et est compatible avec les téléphones portables et tablettes Android.

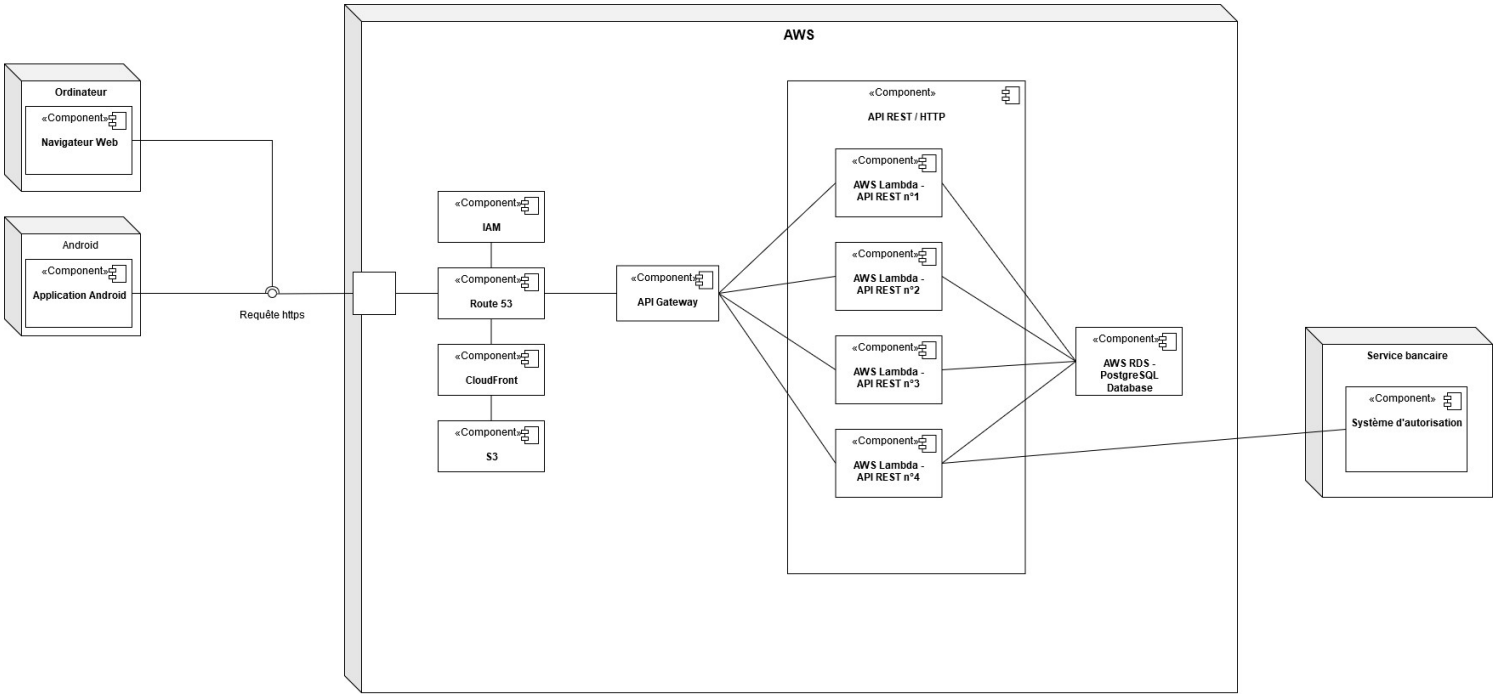
3.4 - API RESTful

Les API RESTful seront développées en Java 8 avec Spring Boot. La configuration du build.gradle se trouve dans la partie 6.2 - Implémentation / build. Les API permettront de questionner la base de données.

3.5 - Base de données

Le moteur de base de données sera PostgreSQL. Le client SQL peut être pgAdmin ou SQL Workbench. La base de données sera gérée avec Amazon RDS.

4 - Architecture de Deploiement



La route 53 est un système de noms de domaine.

Amazon IAM est un service de management des utilisateurs et de leurs permissions.

Amazon S3 est un service de stockage qui va nous permettre de stocker le site web.

Amazon CloudFront est un service nous permettant d'optimiser le S3.

Amazon API Gateway est un service qui permet de créer des API RESTful en utilisant des fonctions d'Amazon Lambda.

Amazon Lambda permet d'exécuter du code sans avoir à mettre en service ou gérer des serveurs.

Amazon RDS est un système nous permettant de gérer une base de données notamment PostgreSQL.

4.1 - Serveur de Base de données

Le serveur de base de données est PostgreSQL 12.4.

PostgreSQL est un puissant SGBD relationnel orienté objet et open source. Avec ses 30 ans de développement actif, PostgreSQL s'est forgé une réputation de SGBD puissant, robuste, fiable et performant.

Ce serveur sera géré via AWS RDS et les identifiants seront stockés avec AWS Secret Manager.
Les tables seront déployées via un script SQL

4.2 - Application Android

L'application sera compilée en APK et disponible dans le Play store.

4.3 - API RESTful

Les API RESTful seront déployées dans AWS Lambda afin de questionner la base de données PostgreSQL via le proxy Amazon RDS Proxy.

4.4 - Application Web

L'application web sera hébergée dans un compartiment du Amazon S3.

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

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

5.1.1 - Les couches

L'architecture applicative est la suivante :

- Une couche **model** : : implémentation du modèle des objets métiers
- Une couche **view** : implémentation de l'interface utilisateur
- Une couche **controller**: implémentation du traitement des données

5.1.2 - Structure des sources

La structuration des répertoires de l'application Angular suit la logique suivante :

```
Racine
 ├─e2e
 ├─ src
 ¦    ├─ app
 ¦        ├─html
 ¦        ├─ ts
 ¦        ├─ css
 ¦    ├─ assets
 ¦    ├─ environment
```

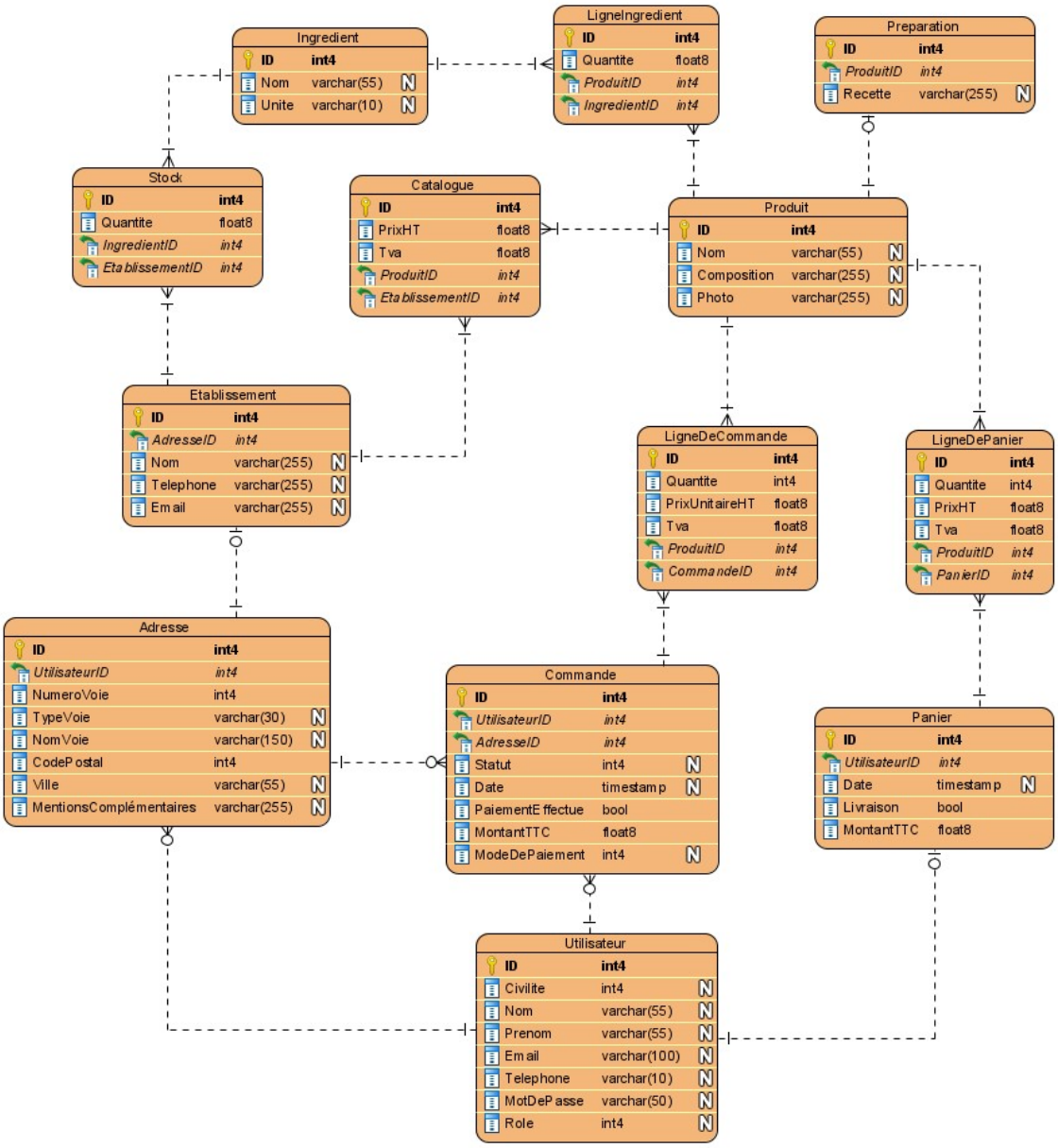
La structuration des répertoires des API RESTful suit la logique suivante :

```
Racine
 ├─ gradle
 ├─ build.gradle
 ├─ settings.gradle
 ├─ gradle.properties
 ├─ src
 ¦    ├─ app
 ¦        ├─ jar
 ¦        ├─ build.gradle
```

La structuration des répertoires de l'application Android suit la logique suivante : (Auto-générée par Android Studio)

```
Racine
 ├─ app
 ¦    ├─ build
 ¦    ├─ libs
 ¦    ├─ src
 ¦        ├─ androidTest
 ¦        ├─ debug
 ¦        ├─ main
 ¦            ├─ java
 ¦            ¦   ├─ MainActivity.java
 ¦            ├─ res
 ¦            ├─ AndroidManifest.xml
 ¦        ├─ release
 ¦        ├─ test
 ¦        ├─ build.gradle
 ├─ build
 ├─ gradle
 ├─ build.gradle
 ├─ gradle.properties
 ├─ gradlew
 ├─ local.properties
 ├─ settings.gradle
```

5.1.3 - Modèle physique des données



Modèle physique des données

Remarque : Le mot de passe de l'utilisateur est crypté via la fonction MD5.

5.1.4 - Diagramme de classes

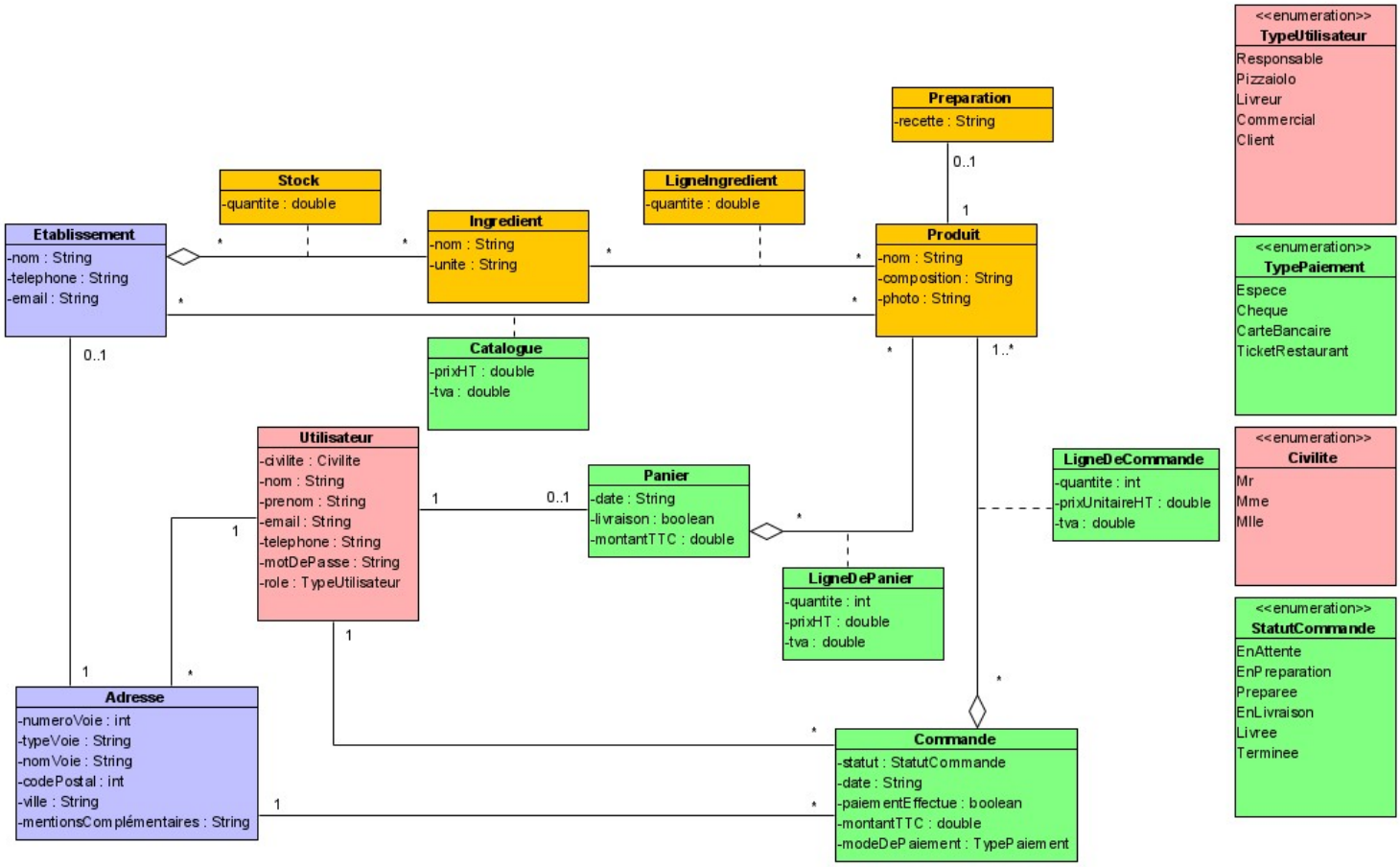


Diagramme de classes

Un utilisateur peut avoir plusieurs ou aucune adresses et commandes et a au plus un panier.

Un établissement a obligatoirement une adresse pour pouvoir être choisi comme point de vente et catalogue plusieurs produits et stocke plusieurs ingrédients.

Une commande possède une adresse pour livrer ou récupérer la commande et comporte au moins une ligne de commande afin d'éviter de faire une commande sans produits.

6 - POINTS PARTICULIERS

6.1 - Environnement de développement

Le développement de l'application Angular se fera sur l'IDE Webstorm d'Intellij.

Pour développer l'application Android nous utiliserons Android Studio.

Enfin les API RESTful seront développées avec Eclipse.

6.2 - Implémentations / build

Afin de développer les API RESTful il faudra ajouter ces lignes dans le fichier « build.gradle » dans le fichier « app ».

```
plugins {
	id 'org.springframework.boot' version '2.3.2.RELEASE'
	id 'io.spring.dependency-management' version '1.0.8.RELEASE'
	id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
	mavenCentral()
}

dependencies {
	implementation 'org.springframework.boot:spring-boot-starter-
web'
	testImplementation('org.springframework.boot:spring-boot-
starter-test') {
		exclude group: 'org.junit.vintage', module: 'junit-
vintage-engine'
	}
}
```

```
implementation 'com.amazonaws:aws-lambda-java-core:1.2.1'
implementation 'com.amazonaws:aws-lambda-java-events:2.2.9'
runtimeOnly 'com.amazonaws:aws-lambda-java-log4j2:1.2.0'

test {
       useJUnitPlatform()
}
```

Ainsi que ces lignes pour build un fichier ZIP à déployer dans AWS Lambda.

```
task buildZip(type: Zip) {
    from compileJava
    from processResources
    into('lib') {
        from configurations.runtimeClasspath
    }
}
```

6.3 - Identifiants

Le nom de compte AWS est : OCPIZZA

Le mot de passe du compte AWS est : XXXX

7 - GLOSSAIRE

SGBD	Système de gestion de base de données
MPD	Modèle physique de données
IDE	Environnement de développement intégré