

# OC PIZZA

## Nouveau Système Informatique

Dossier de conception technique

Version 1.0

**Auteur**

Khadija CHTIOUI

*Analyste Programmeur*

# TABLE DES MATIERES

<b>1 - Versions.....</b>	<b>4</b>
<b>2 - Introduction.....</b>	<b>5</b>
2.1 - Objet du document.....	5
2.2 - Références .....	5
<b>3 - Architecture Technique .....</b>	<b>6</b>
<b>3.1 - Composants généraux.....</b>	<b>6</b>
3.1.1 - Diagramme de composant UML.....	7
3.1.2 - Package Boutique en ligne .....	8
3.1.2.1 - Composant Authentification .....	8
3.1.2.2 - Composant panier :.....	8
3.1.3 - Package Commandes .....	8
3.1.3.1 - Composant Commande en cours.....	8
3.1.3.2 - Composant Suivi de la commande.....	8
3.1.3.3 - Composant Livraison.....	8
3.1.4 - Package pizzeria .....	8
3.1.4.1 - Composant Menu .....	8
3.1.4.2 - Composant Stock .....	9
3.1.4.3 - Composant Recette.....	9
3.1.5 - Package Administration.....	9
3.1.5.1 - Composant Back-Office : .....	9
3.1.6 - Package externe La base de données (BDD) .....	9
3.1.7 - Package externe Système de paiement.....	9
<b>3.2 - Application Web &amp; Mobile .....</b>	<b>10</b>
3.4.1 – Back End .....	10
3.4.2 – Front End .....	10
3.4.3 - Mobile .....	10
<b>4 - Architecture de Déploiement .....</b>	<b>11</b>
4.1 - Diagramme de déploiement.....	11
4.2 - Serveur de Base de données .....	12
4.2.1 - Modèle Physique de Données (MPD).....	12
4.2.2 - Création de la Base de Données (SQL) .....	13
4.3 - Serveur de base de données .....	21
4.4 - Serveur Web .....	21
<b>5 - Architecture logicielle .....</b>	<b>22</b>
5.1 - Principes généraux.....	22
5.1.1 - Structure des sources.....	23

<b>6 - Points particuliers.....</b>	<b>24</b>
<b>6.1 - Gestion des logs.....</b>	<b>24</b>
<b>6.2 - Fichiers de configuration .....</b>	<b>25</b>
<b>6.3 - Ressources .....</b>	<b>25</b>
6.3.1 - Carte graphique.....	25
6.3.2 - Données .....	25
<b>6.4 - Environnement de développement .....</b>	<b>26</b>
<b>6.5 - Procédure de packaging / livraison.....</b>	<b>26</b>
<b>7 - Glossaire .....</b>	<b>27</b>

# 1 - VERSIONS

Auteur	Date	Description	Version
Kh.CHTIOUI	01/03/2022	Creation du document	1.0

## 2 - INTRODUCTION

### 2.1 - Objet du document

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

Objectif du document :

Présenter les fonctionnalités du Système « OC Pizza » telles qu'elles ont pu être définies avec le client lors des échanges et réunions préparatoires.

Les éléments du présent dossier découlent :

- De la présentation au client des spécifications fonctionnelles lors de l'entrevue de lancement du projet
- Du recueil des besoins et du cahier des charges donné par le client
- De la présentation de la solution technique

### 2.2 - Références

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

1. **DCF – 1.0** : Dossier de conception fonctionnelle de l'application
2. **CE – 1.0** : Dossier d'exploitation de l'application

## 3 - ARCHITECTURE TECHNIQUE

### 3.1 - Composants généraux

Le diagramme de composants illustre la relation entre les différents composants du système.

On a deux types d'accès celui du client (Front Office) et celui du personnel de la Pizzeria (Back Office).

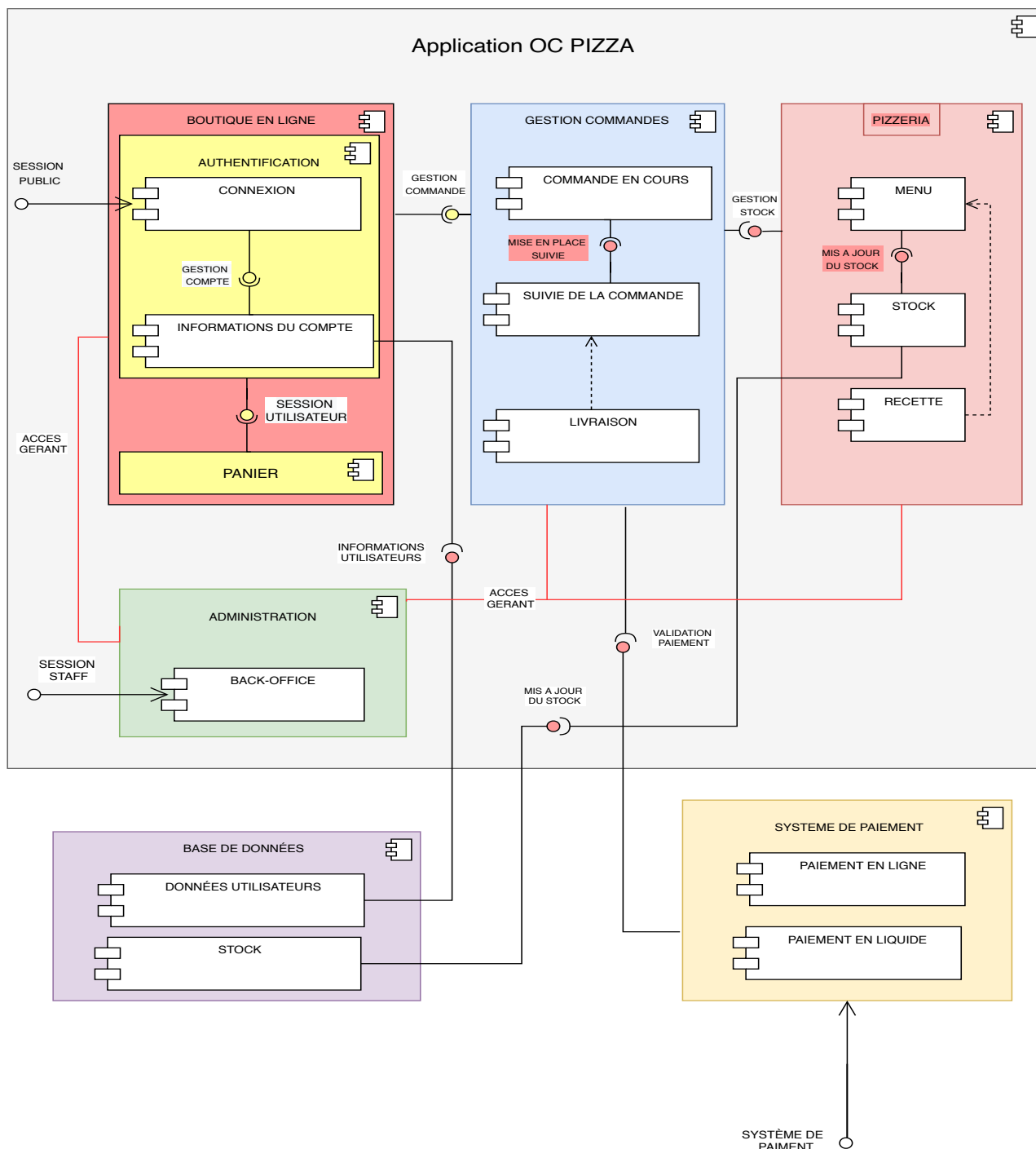
Avant de pouvoir commander le client doit se connecter.

Son panier sera lié à la composante « Commande » qui elle-même sera liée à la composante « Pizzeria » afin d'obtenir la disponibilité des produits.

Un panier validé par le client devient alors une commande en cours et active le suivi de commande. On peut aussi constater des dépendances comme la Livraison qui va dépendre du Suivi ou la Recette qui dépend du Produit.

### 3.1.1 - Diagramme de composant UML

Diagramme de composant



### **3.1.2 - Package Boutique en ligne**

On regroupe les fonctionnalités nécessaires pour la connexion au compte et à la prise d'une commande.

#### **3.1.2.1 - Composant Authentification**

Il permet l'Authentification de l'utilisateur. Il va permettre l'inscription ou la connexion d'un utilisateur et va contenir les informations au compte (Adresse, Numéro de téléphone, adresse, email, etc.)

Par défaut lors de la création d'un compte, le rôle est défini sur client.

#### **3.1.2.2 - Composant panier :**

Il permet la création d'une commande par l'utilisateur.

Les options seront plus ou moins poussées selon le rôle de l'utilisateur.

### **3.1.3 - Package Commandes**

Regroupe les fonctionnalités nécessaires au suivi des commandes pour un client ou une pizzeria.

#### **3.1.3.1 - Composant Commande en cours**

Il permet d'afficher une commande et son détail avec les lignes de commandes et le montant de la commande. Il va donc permettre le suivi des commandes actuelles.

#### **3.1.3.2 - Composant Suivre de la commande**

Il permet de donner les informations liées à une commande et de l'avancement de son état, il dépend de la commande en cours.

L'avancement du statut va permettre d'annuler ou modifier une commande quand cela est encore possible.

#### **3.1.3.3 - Composant Livraison**

Il permet la gestion des informations liées à la livraison et renvoie les informations à l'artefact suivi de la commande.

### **3.1.4 - Package pizzeria**

Ce composant va contenir les informations d'un pizzeria et donc les produits disponibles dans celui-ci.

#### **3.1.4.1 - Composant Menu**

Il permet l'affichage des produits proposés par un pizzeria.



Il contient les informations détaillées des différents produits ainsi que leur prix de vente.  
Il est relié au stock afin d'afficher la disponibilité des produits.

### **3.1.4.2 - Composant Stock**

Il permet la mise à jour en temps réel des produits qui sont encore disponibles dans le stock.

### **3.1.4.3 - Composant Recette**

Il permet l'affichage d'une recette et des composants pour le Pizzaiolo.

Il est relié au Menu afin d'afficher la recette correspondante ainsi que la quantité nécessaire à la création de celui-ci.

## **3.1.5 - Package Administration**

Ce package est spécifique au gérant.

Il permet des manipulations avancées sur les comptes et un visuel global du groupe.

### **3.1.5.1 - Composant Back-Office :**

Ce composant permet au gérant d'avoir une vue en temps réel sur l'activité du groupe (commande stock livraison).

Il va permettre de créer des comptes utilisateurs pour la pizzeria ou le cas échéant de faire des modifications.

## **3.1.6 - Package externe La base de données (BDD)**

Elle va contenir les informations nécessaires à la partie Authentification, Commande et Pizzeria

Chaque composant a recours à la BDD pour fournir les informations nécessaires que ce soit pour vérifier les informations d'un compte utilisateur, d'afficher le détail d'une commande ou d'informer sur les produits disponibles ainsi que leur disponibilité.

## **3.1.7 - Package externe Système de paiement**

Le paiement d'une commande passera par le système de paiement qui va enregistrer le règlement de la commande.

## 3.2 - Application Web & Mobile

La pile logicielle est la suivante :

- Nginx version 1.21.1 ou ultérieure
- Python version 3.9.5 ou ultérieure
- Django version 3.2.3 ou ultérieure
- HTML version 5.3 ou ultérieure
- CSS version 4.15 ou ultérieure
- JavaScript version ECMAScript2020 ou ultérieure
- PostgreSQL version 9.6 ou ultérieure
- Java version 8 ou ultérieure

Serveur Web  
Back End  
Back End  
Front End  
Front End  
Front End  
Base de données  
Application Android

### 3.4.1 – Back End

Le Back-End est la partie invisible du site en charge du côté logique du site  
Celui-ci sera codé en python avec un Framework Django  
Il sera en lien direct avec notre BDD.

### 3.4.2 – Front End

Le Front End est la partie visible du site web (notre interface client et équipe pizzeria).  
Pour ce faire, nous utiliserons les standards que sont HTML, CSS et JavaScript.

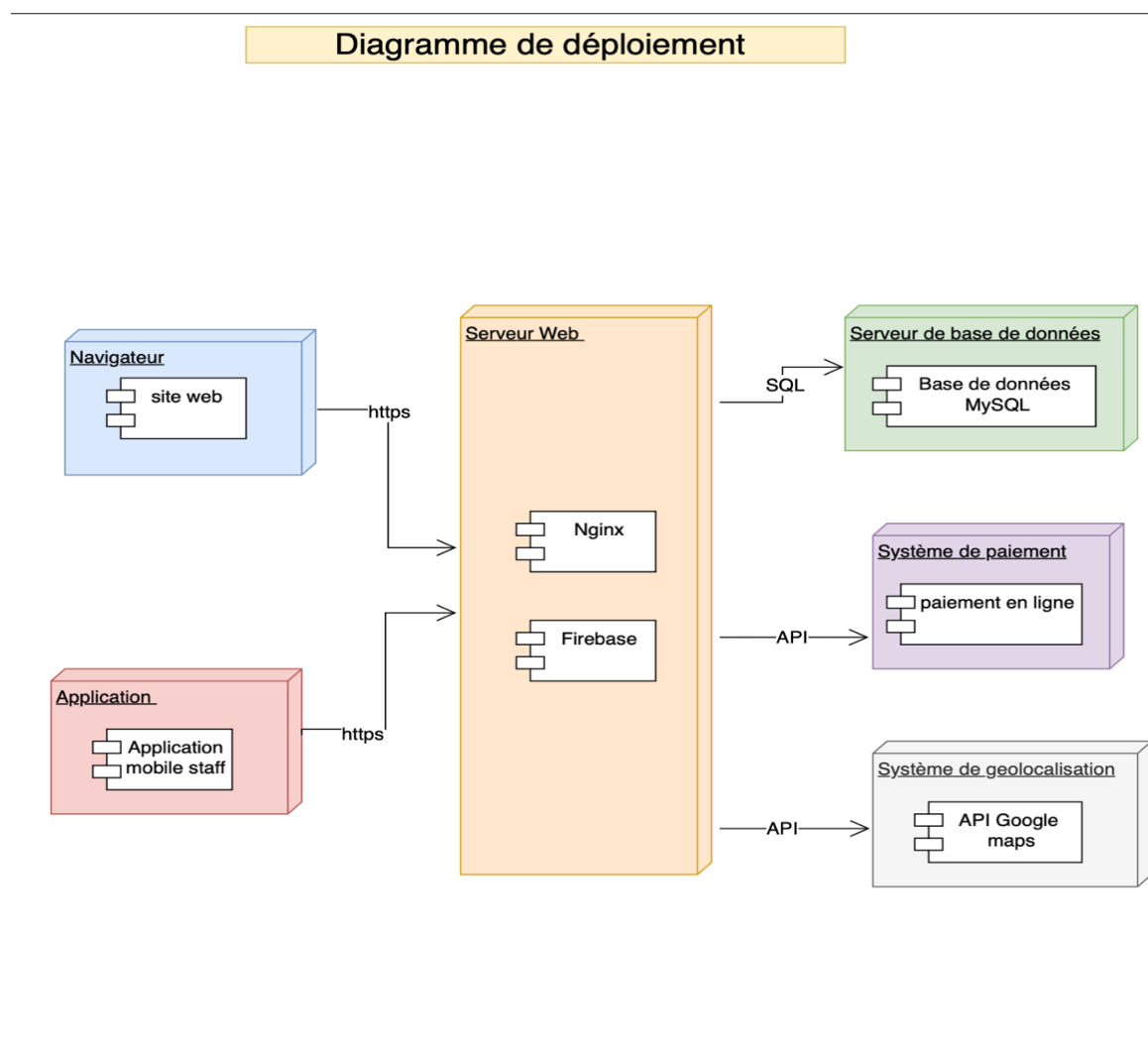
### 3.4.3 - Mobile

L'application mobile étant sous Android, nous utiliseront le langage Java pour la développer.

## 4 - ARCHITECTURE DE DÉPLOIEMENT

### 4.1 - Diagramme de déploiement

Notre diagramme de déploiement explique la disposition des composants du système sur les infrastructures physiques.



Notre serveur est composé d'un Front-end et d'un Back-end qui vont communiquer ensemble sur notre serveur.

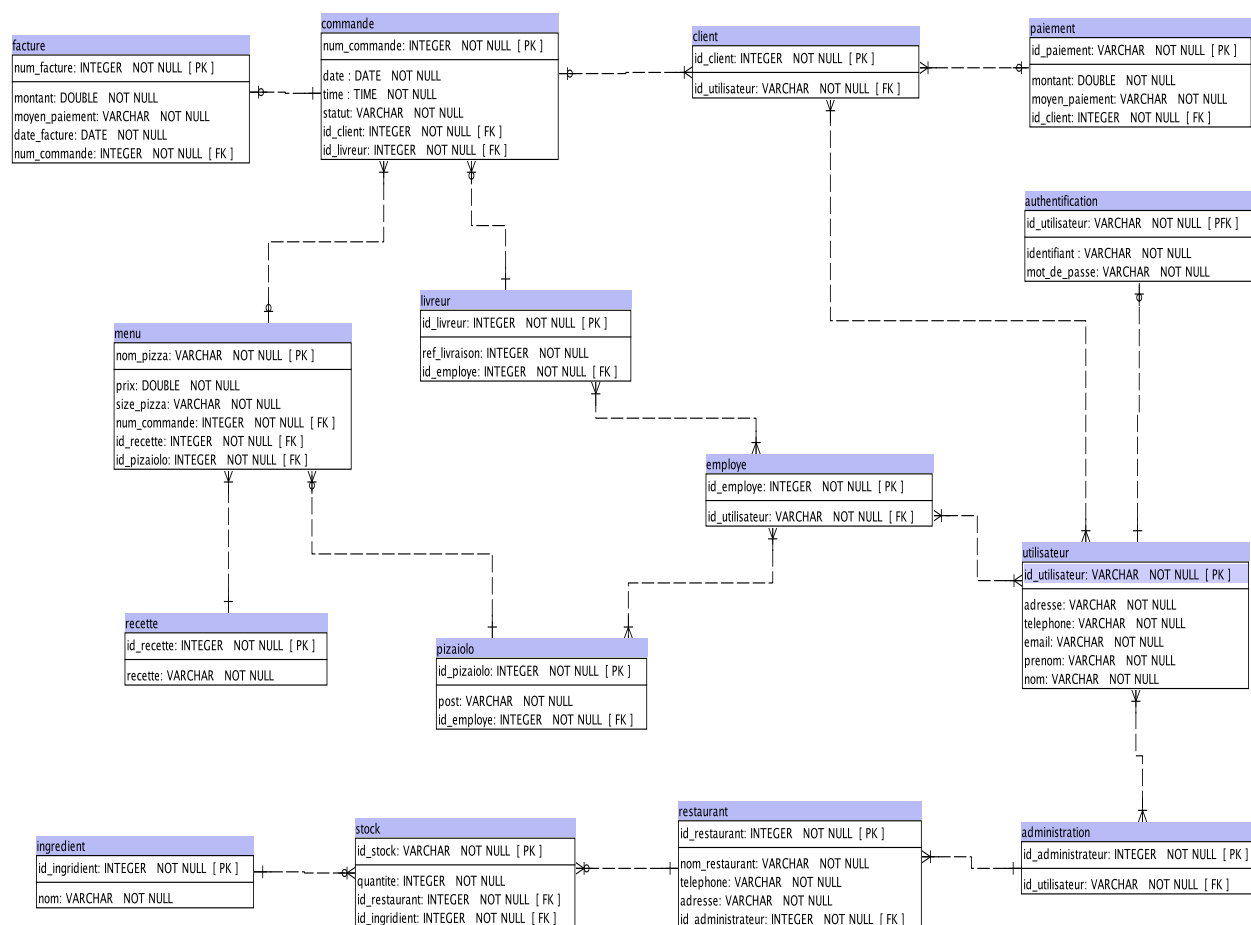
Nos utilisateurs accèdent à la partie Front End du Site Web qui va permettre d'afficher les informations voulues.

Notre partie Back End quant à elle, va communiquer avec la base de données via des

Enfin nous avons recours à des API pour la partie système de paiement et le système de localisation.

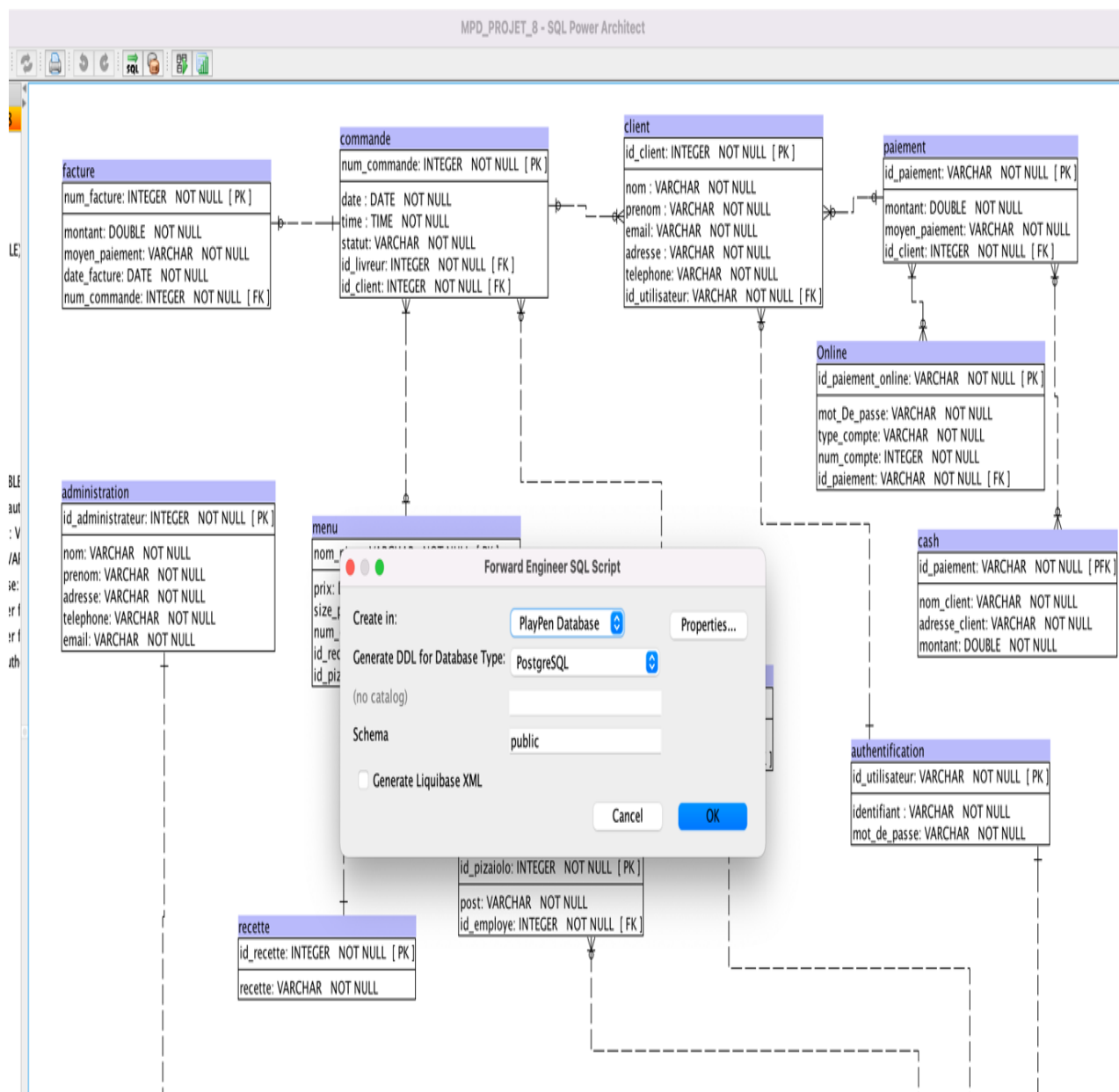
Notre choix pour le système de gestion de base de données s'est tourné sur PostgreSQL.

Le modèle va représenter les relations entre les différentes tables.

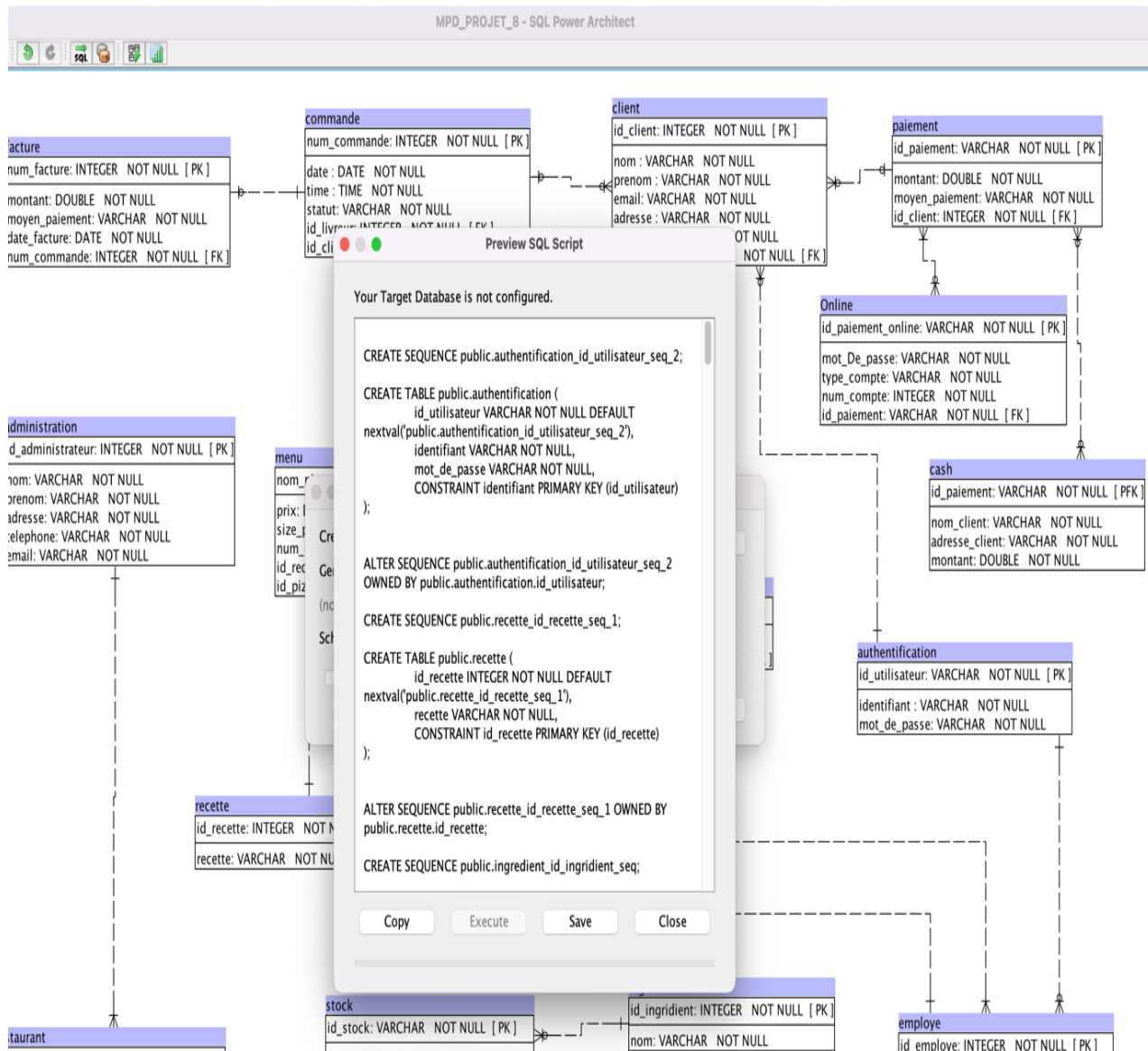


## 4.2.2 - Création de la Base de Données (SQL)

Dans "Power Architect", je génère la base de données à partir de mon modèle physique de données, en cliquant sur l'icône SQL situé en dessous du menu.

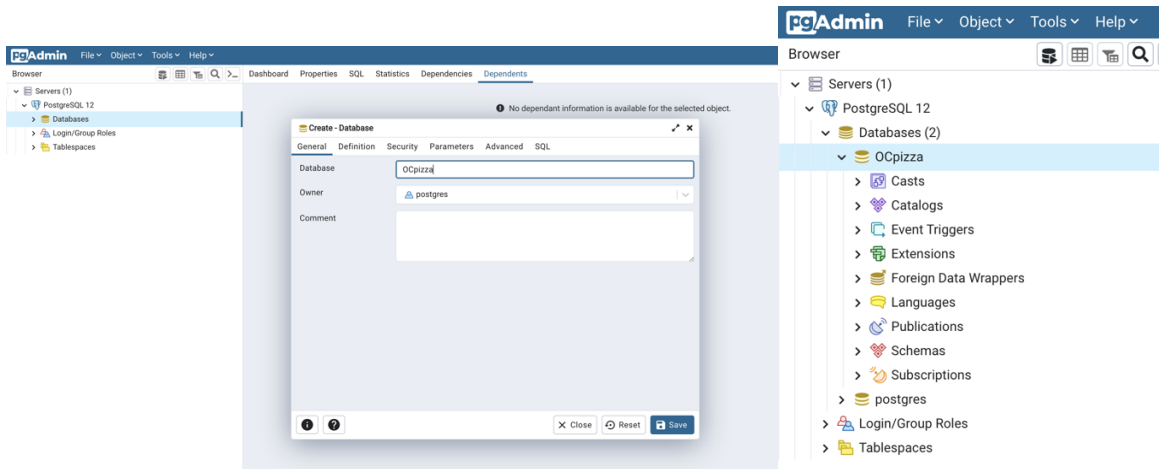


\_ Ensuite je copie le code crée en SQL, à l'aide du bouton "Copy".

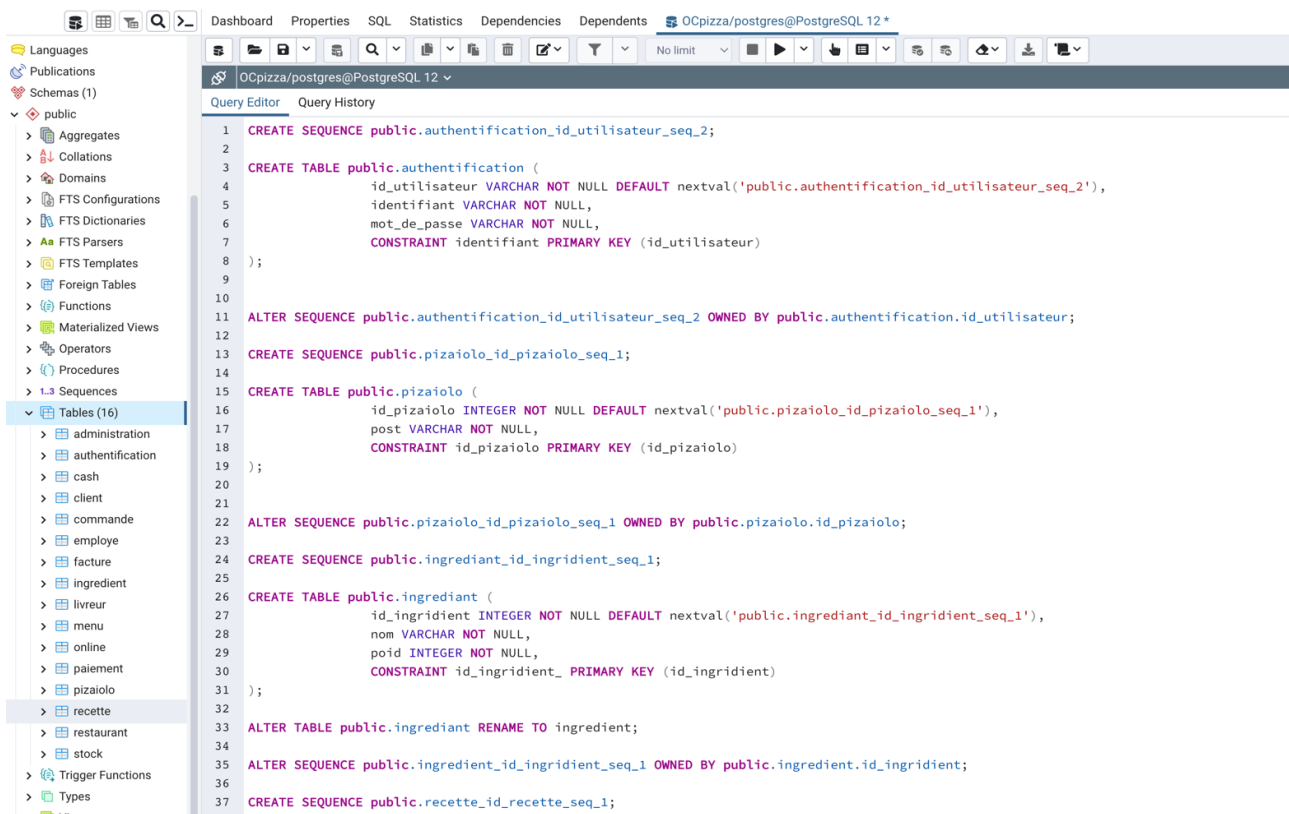


\_ Je me connecte dans "pgAdmin4" et je crée une nouvelle base de données.

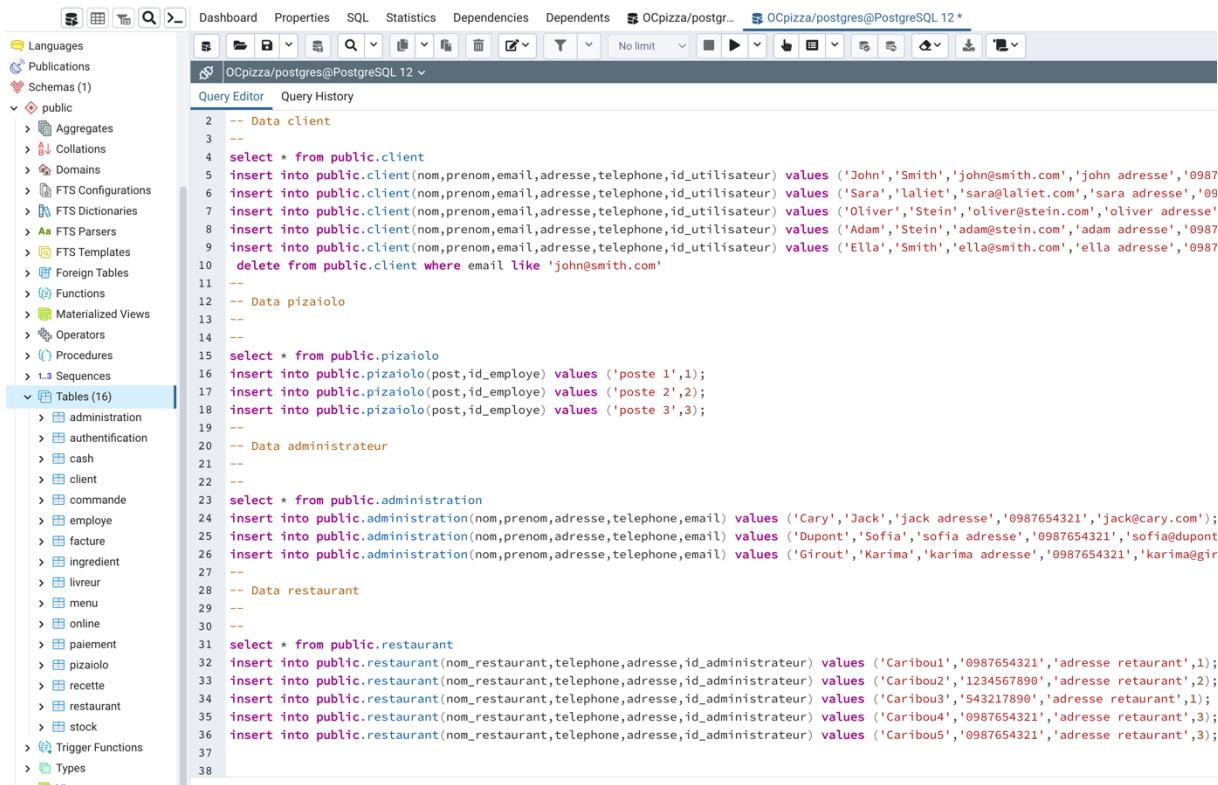
\_ Je rentre le nom de ma base de données, je l'enregistre, et ma base de données apparaît dans le navigateur.



\_ Je clique sur 'Query Editor', Je colle le code crée en SQL sur "Power Architect" dans cette partie et je valide. Les tables du groupe "OC Pizza" sont maintenant dans la base données.



\_j'ai effectué quelque insertion dans la base de données :



```

-- Data client
select * from public.client
insert into public.client(nom,prenom,email,adresse,telephone,id_utilisateur) values ('John','Smith','johnsmith.com','john adresse','0987
insert into public.client(nom,prenom,email,adresse,telephone,id_utilisateur) values ('Sara','laliet','sara@laliet.com','sara adresse','09
insert into public.client(nom,prenom,email,adresse,telephone,id_utilisateur) values ('Oliver','Stein','oliver@stein.com','oliver adresse'
insert into public.client(nom,prenom,email,adresse,telephone,id_utilisateur) values ('Adam','Stein','adam@stein.com','adam adresse','0987
insert into public.client(nom,prenom,email,adresse,telephone,id_utilisateur) values ('Ella','Smith','ella@smith.com','ella adresse','0987
delete from public.client where email like 'johnsmith.com'

-- Data pizzaiolo
select * from public.pizzaiolo
insert into public.pizzaiolo(post,id_employe) values ('poste 1',1);
insert into public.pizzaiolo(post,id_employe) values ('poste 2',2);
insert into public.pizzaiolo(post,id_employe) values ('poste 3',3);

-- Data administrateur
select * from public.administration
insert into public.administration(nom,prenom,adresse,telephone,email) values ('Cary','Jack','jack adresse','0987654321','jack@cary.com');
insert into public.administration(nom,prenom,adresse,telephone,email) values ('Dupont','Sofia','sofia adresse','0987654321','sofia@dupont
insert into public.administration(nom,prenom,adresse,telephone,email) values ('Girout','Karima','karima adresse','0987654321','karima@gir

-- Data restaurant
select * from public.restaurant
insert into public.restaurant(nom_restaurant,telephone,adresse,id_administrateur) values ('Cariboul','0987654321','adresse restaurant',1);
insert into public.restaurant(nom_restaurant,telephone,adresse,id_administrateur) values ('Caribou2','1234567890','adresse restaurant',2);
insert into public.restaurant(nom_restaurant,telephone,adresse,id_administrateur) values ('Caribou3','543217890','adresse restaurant',1);
insert into public.restaurant(nom_restaurant,telephone,adresse,id_administrateur) values ('Caribou4','0987654321','adresse restaurant',3);
insert into public.restaurant(nom_restaurant,telephone,adresse,id_administrateur) values ('Caribou5','0987654321','adresse restaurant',3);

```

### Data client :

	id_client [PK] integer	id_utilisateur character varying
1	1	1
2	2	2
3	3	2
4	4	4
5	5	5
6	6	6

### Data pizzaiolo :

	id_pizzaiolo [PK] integer	post character varying	id_employe integer
1	2	poste 1	1
2	3	poste 2	2
3	4	poste 3	3



### Data utilisateur:

	<b>id_utilisateur</b> [PK] character varying	<b>adresse</b> character varying	<b>telephone</b> character varying	<b>email</b> character varying	<b>prenom</b> character varying	<b>nom</b> character varying
1	1	sara adresse	0987654321	sara@laliel.com	Sara	laliel
2	2	oliver adresse	0987654321	oliver@stein.com	Oliver	Stein
3	3	oliver adresse	0987654321	oliver@stein.com	Oliver	Stein
4	4	john adresse	0987654321	john@smith.com	John	Smith
5	5	adam adresse	0987654321	adam@stein.com	Adam	Stein
6	6	ella adresse	0987654321	ella@smith.com	Ella	Smith
7	7	jack adresse	0987654321	jack@cary.com	Jack	Cary
8	8	sofia adresse	0987654321	sofia@dupont.com	Sofia	Dupont
9	9	karima adresse	0987654321	karima@girout.com	Karima	Girout
10	10	karima adresse	0987654321	karima@girout.com	Karima	Girout
11	11	sofia adresse	0987654321	sofia@dupont.com	Sofia	Dupont
12	12	lopez adresse	0987654321	lopez@martin.com	Martin	Lopez
13	13	Guillot adresse	0987654321	thomas@guillot.com	Thomas	Guillot
14	14	Hubert adresse	0987654321	bernard@hubert.com	Bernard	Hubert
15	15	Dupuis adresse	0987654321	Laurent@Dupuis.com	Laurent	Dupuis

### Data administrateur:

	<b>id_administrateur</b> [PK] integer	<b>id_utilisateur</b> character varying
1	1	7
2	2	8
3	3	9

### Data restaurant:

	<b>id_restaurant</b> [PK] integer	<b>nom_restaurant</b> character varying	<b>telephone</b> character varying	<b>adresse</b> character varying	<b>id_administrateur</b> integer
1	1	Caribou1	0987654321	adresse restaurant	1
2	2	Caribou2	1234567890	adresse restaurant	2
3	3	Caribou3	543217890	adresse restaurant	1
4	4	Caribou4	0987654321	adresse restaurant	3
5	5	Caribou5	0987654321	adresse restaurant	3

Data employé:

	<b>id_employe</b> [PK] integer	<b>id_utilisateur</b> character varying
1		
2	2	13
3	3	14
4	4	15

Data livreur:

	<b>id_livreur</b> [PK] integer	<b>ref_livraison</b> integer	<b>id_employe</b> integer
1	1	88665544	3
2	2	88665544	4

Data commande:

	<b>num_commande</b> [PK] integer	<b>date</b> date	<b>time</b> time without time zone	<b>statut</b> character varying	<b>id_client</b> integer	<b>id_livreur</b> integer
1	1	2020-10...	10:00:00	commande validée	6	2
2	3	2020-10...	10:00:00	commande anulée	1	2
3	5	2020-10...	10:00:00	commande prête à ê...	2	2
4	6	2020-10...	10:00:00	commande en prépa...	3	1
5	7	2020-10...	10:00:00	commande en attente	4	2
6	8	2020-10...	10:00:00	commande en attent...	5	2
7	9	2020-10...	10:00:00	commande finalisée	3	1
8	10	2020-10...	10:00:00	commande en cours...	2	1
9	11	2020-10...	10:00:00	commande livrée et ...	5	2

### Data authentication:

	<b>id_utilisateur</b> [PK] character varying	<b>identifiant</b> character varying	<b>mot_de_passe</b> character varying
1	1	john@smith.com	kjjhggtr
2	2	sara@lallet.com	kjjhggtr
3	3	oliver@stein.com	kjjhggtr
4	4	ella@smith.com	kjjhggtr
5	5	adam@stein.com	kjjhggtr
6	6	Lopez@Martin.fr	kjjhggtr
7	7	Guillot@Thomas.fr	kjjhggtr
8	8	Hubert@Bernard.fr	kjjhggtr
9	9	Dupuis@Laurent.fr	kjjhggtr
10	10	Vasseur@Charles.fr	kjjhggtr

### Data recette:

	<b>id_recette</b> [PK] integer	<b>recette</b> character varying
1	1	(Ingrédients : 400 g / 14 oz pâte à pizza fraîche,45 ml / 3 c. à soupe huile olive,2 oignons moyens émincés, 2grosses tomates en tranches fines
2	2	(Les quantités sont données pour une petite pizza, multipliez par 1.5 pour une moyenne et par deux pour une grande.)Prenez un paton et étalez la pâte, jusqu'à la...
3	3	(400 g / 14 oz pâte à pizza fraîche,45 ml / 3 c. à soupe huile olive,2 oignons moyens émincés, 2grosses tomates en tranches fines
4	4	(Les quantités sont données pour une petite pizza, multipliez par 1.5 pour une moyenne et par deux pour une grande.)Prenez un paton et étalez la pâte, jusqu'à la...

### Data ingrédient:

	<b>id_ingredient</b> [PK] integer	<b>nom</b> character varying
1	1	tomate
2	2	Oignon
3	3	olive
4	4	poivron
5	5	fromage

### Data Menu:

	<b>nom_pizza</b> [PK] character varying	<b>prix</b> double precision	<b>size_pizza</b> character varying	<b>num_commande</b> integer	<b>id_recette</b> integer	<b>id_pizaiolo</b> integer
1	Margarita	7	small	2	1	3
2	frommage	10	moyenne	5	2	2
3	Vegetarian	10	grande	8	3	3
4	pizza minute	7	small	10	4	3

### Data stock:

	<b>id_stock</b> [PK] character varying	<b>quantite</b> integer	<b>id_restaurant</b> integer	<b>id_ingridient</b> integer
1	1	30	2	1
2	2	50	3	2
3	7	100	4	3
4	8	9	3	1
5	9	200	5	5
6	11	70	4	3
7	12	59	2	1

### Data facture:

	<b>num_facture</b> [PK] integer	<b>montant</b> double precision	<b>moyen_paiement</b> character varying	<b>date_facture</b> date	<b>num_commande</b> integer
1	2	30	carte de credit	2020-10-12	2
2	3	100	cash	2020-10-12	4
3	4	300	carte de credit	2020-10-12	5
4	5	50	carte de credit	2020-10-12	7
5	6	10	cash	2020-10-12	10
6	7	99	cash	2020-10-12	8
7	9	567	carte de credit	2020-10-12	12
8	10	234	cash	2020-10-12	11

Data paiement:

	id_paiement [PK] character varying	montant double precision	moyen_paiement character varying	id_client integer
1	1	30	carte de credit	6
2	3	30	carte de credit	6
3	4	100	cash	3
4	5	300	carte de credit	4
5	6	60	cassh	5
6	7	90	carte de credit	6

### 4.3 - Serveur de base de données

Point central de notre système la base de données sera sous PostgreSQL.

La base de données sera hébergée sur un serveur OVH doté de :

- 2 CPU
- 4 Go de RAM

### 4.4 - Serveur Web

Pour le serveur Web nous avons choisi l'offre d'OVH « Cloud Web », qui inclus :

- 2 CPU
- 4 Go de RAM

A cela s'ajoute d'autres fonctions utiles comme :

- Un nom de domaine personnalisé
- Le **SSL Let's Encrypt** qui inclus la certification du nom de domaine en activant le protocole HTTPS

Le coût mensuel est de 22.80€ TTC par mois.

L'offre est sécurisée et facilement modifiable.

## 5 - ARCHITECTURE LOGICIELLE

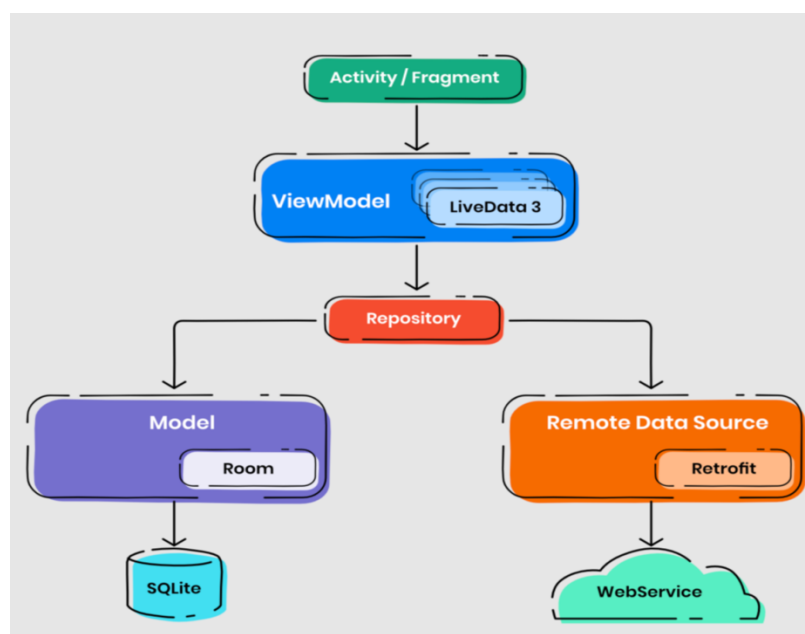
### 5.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git**, les dépendances et le packaging par **Apache Maven**.

Les couches

L'architecture applicative est la suivante :

- Une couche **model** : implémentation du modèle des objets métiers
- Une couche **ViewModel** : contient les méthodes permettant de récupérer, d'envoyer, de sauvegarder les données et d'interagir avec les objets pour y avoir accès. Elle fait le lien entre la couche dao et la couche model.
- Une couche **dao** : interface permettant d'interroger la base de données.
- Une couche **View** : gestion de l'interface utilisateur



### 5.1.1 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

- les répertoires sources sont créés de façon à respecter l'architecture MVVM.



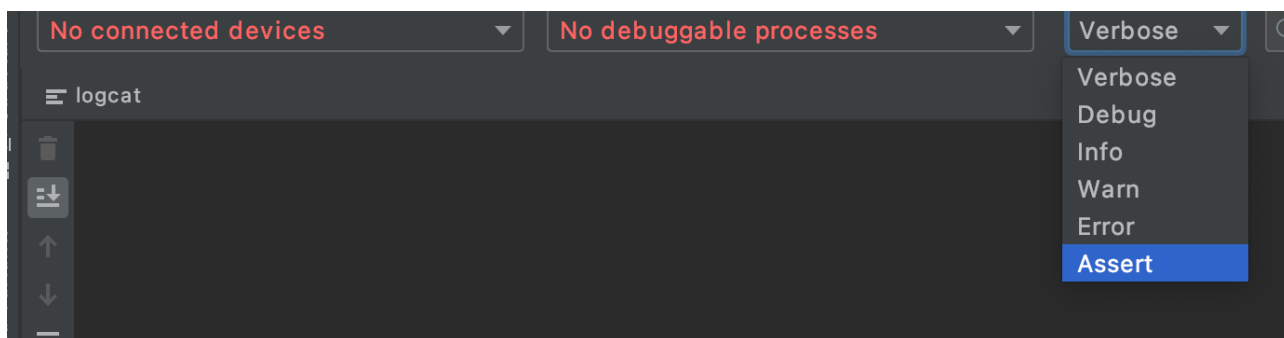
## 6 - POINTS PARTICULIERS

### 6.1 - Gestion des logs

Le système de logs est indispensable dans le développement Android. Il est utile pour afficher des messages possédant différents niveaux d'importance. Ces messages seront visibles à l'aide du **logcat**.

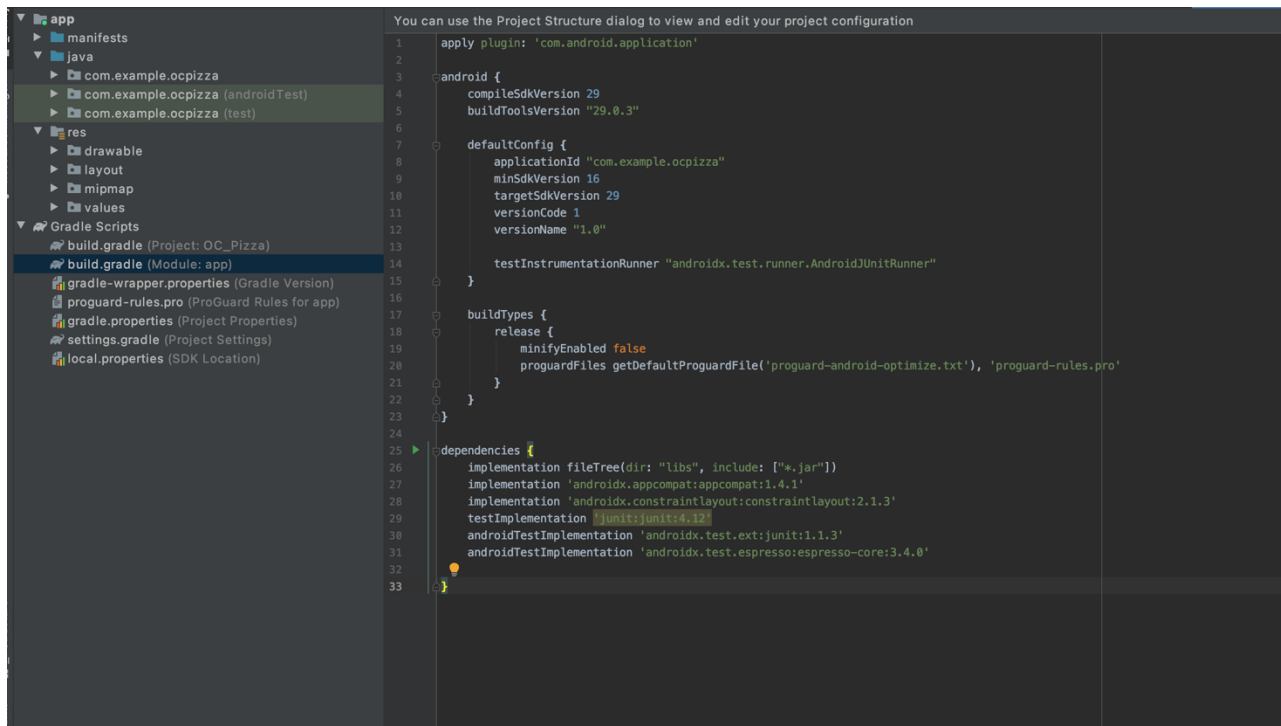
À l'aide de cet outil, vous pouvez :

- Voir les messages de logs affichés par l'appareil cible.
- Filtrer les messages à afficher selon l'application, l'identifiant ou le tag d'une application.
- Recherche un message.
- Filtrer les messages par importance (verbose, debug, error, etc.).
- Sauvegarder les logs sélectionnés dans un fichier.





## 6.2 - Fichiers de configuration



## 6.3 - Ressources

### 6.3.1 - Carte graphique

Les éléments graphiques (polices, logos, photos des produits, jeux de couleur, ...) sont fournis par OC Pizza.

### 6.3.2 - Données

Les données pour la création de la base de données sont-elles aussi fournies par OC Pizza, cela comprend notamment : les produits, les données utilisateurs, les adresses, ...  
Mais aussi les références bancaires pour la partie encaissement.

## 6.4 - Environnement de développement

Le choix de l'environnement reste libre à chaque développeur néanmoins il existe des Prérequis et des préconisations.

- L'IDE pour le développement de l'application Android sera Android studio.
- L'IDE recommandé pour le développement de la partie Site Web est Visual Studio Code
- L'outil de gestion pour PostgreSQL reste libre mais pgAdmin est préférable
- Logiciel PowerArchitect permet de modéliser une base de données puis de générer automatiquement le schéma dans le système de gestion de base de données de notre choix.
- Le versionning passera par un client Git qui sera interfacé avec GitHub

## 6.5 - Procédure de packaging / livraison

Le système fera l'objet d'un déploiement sur OVH et sur Google Play au moment de la livraison finale.

**La livraison du projet comprends :**

- Le code source du site sera donnée dans un zip au client.
- Le code de l'application qui aura son propre zip.
- Les sauvegardes data et structure de la base de données.
- Le dossier de conception fonctionnelle.
- Le dossier d'exploitation.
- Le dossier de conception technique.
- Le PV de livraison.

## 7 - GLOSSAIRE

<b>API</b>	API (Application Programming Interface) Interface permettant à un logiciel d'accéder aux services d'un autre logiciel par le biais d'un ensemble normalisé de classes, de méthodes, de fonctions et de constantes.
<b>CSS</b>	Les CSS ( <i>Cascading Style Sheets</i> en anglais, ou « feuilles de style en cascade ») sont le code utilisé pour mettre en forme une page web.
<b>GitHub</b>	GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.
<b>HTML</b>	HTML (Hyper Text Markup Language) Langage de balisage conçu pour représenter les pages web.
<b>SGBD</b>	Système de Gestion de Base de données
<b>HTTPS</b>	HTTPS (HyperText Transfer Protocole Secure) Protocole permettant de sécuriser les données échangées entre le client et le serveur.
<b>Logs</b>	Le terme log désigne un type de fichier, ou une entité équivalente, dont la mission principale consiste à stocker un historique des événements.
<b>Serveur</b>	<p>Un serveur désigne un élément de logiciel ou de matériel informatique qui fournit des fonctionnalités à d'autres dispositifs ou programmes, appelés clients. Les serveurs peuvent offrir diverses fonctionnalités, comme le partage de ressources ou de données avec plusieurs clients, l'exécution de calculs pour un client, etc.</p> <p>Dans le cas des applications, un serveur est un framework qui héberge des applications. Il permet de créer des applications web et un environnement serveur pour les exécuter de manière transparente.</p>
<b>Hébergement en cloud</b>	Hébergement en cloud utilise différents serveurs pour maximiser le temps de fonctionnement et équilibrer la charge. Au lieu d'utiliser un seul serveur, votre site web ou votre application peut s'appuyer sur un cluster qui utilise les ressources d'un pool centralisé. Ainsi, si un serveur tombe en panne, un autre prend le relais et tout se passe bien.
<b>IDE</b>	Un environnement de développement informatique est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels.

<b>Maven</b>	Maven est un outil de gestion et d'automatisation de production des projets logiciels Java en général et JavaEE en particulier.
<b>Postgresql</b>	PostgreSQL est un système de gestion de base de données relationnelle et objet permettant la gestion de la base de données.