OC Pizza

Document de spécifications techniques

Projet 7

Maxime Bontemps
09/2019

Sommaire:

I - Introduction

Contexte

« OC Pizza » est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici la fin de l'année.

Pour répondre à ses attentes un des responsables nous demande de mettre en place un système informatique qui serait déployé dans toutes ses pizzerias

Analyse du besoin

Rappel des objectifs

Le nouveau système informatique que nous proposerons à OC Pizza devra permettre de :

-Être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;

-Suivre en temps réel les commandes passées et en préparation ;

-Suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas sont encore réalisables ;

-Proposer un site Internet pour que les clients puissent :

. Passer leurs commandes, en plus de la prise de commande par téléphone ou sur place

. Payer en ligne leur commande s'ils le souhaitent – sinon, ils paieront directement à la livraison

. Modifier ou annuler leur commande tant que celle-ci n'a pas été préparée

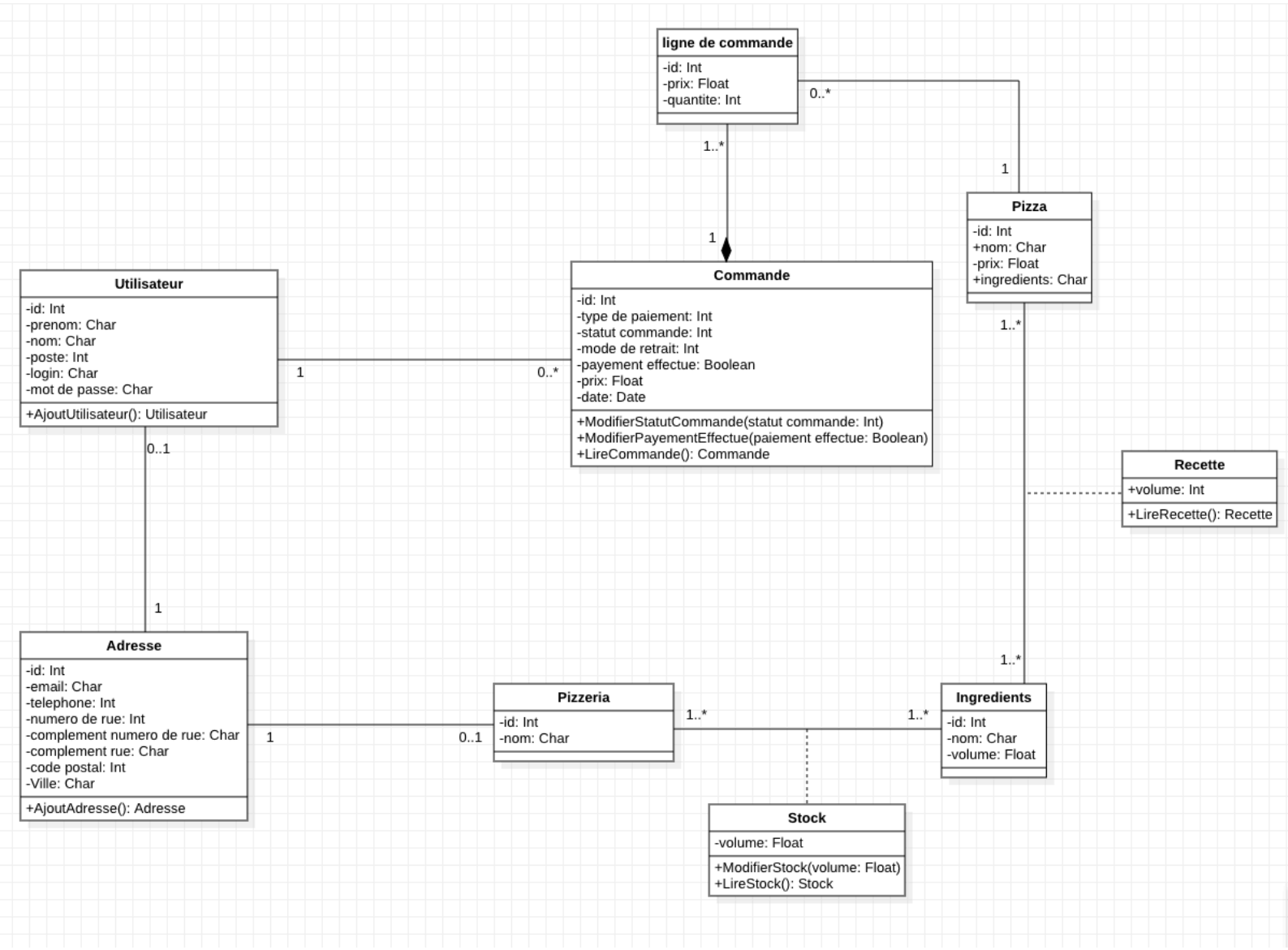
-Proposer un aide-mémoire aux pizzaiolos indiquant la recette de chaque pizza

II - Domaine fonctionnel

Le domaine fonctionnel de l'application sera composé des objets suivants :

- Commande: contient les commandes
- Utilisateurs : contient les coordonnées des utilisateurs (clients et employés)
- Pizzéria : contient les coordonnées des pizzérias
- Adresse : contient les adresses
- Lignes de commandes : contient le détail des pizzas commandées pour une commande
- Pizza : contient les pizzas avec leur nom et leur prix ainsi que les ingrédients qui les composent
- Recette : contient la totalité des ingrédients nécessaires à la réalisation des pizzas
- Ingrédient : contient tous les ingrédients
- Stock : contient le nombre d'ingrédients disponibles

Diagramme de classe



Une base de donnée relationnelle s'appuie sur un modèle relationnel. C'est à dire qu'on va créer des relations entre les tables. UML va créer un formalisme pour modéliser ces relations entre les classes: une association (un trait sur les diagrammes).

On peut poser des contraintes numériques sur les associations ,cela va limiter le nombre minimal et maximal d'instances des classes dans l'association pouvant être liées entre elles. On appelle cela les multiplicités.

Il existe différentes catégories d'association en fonction des multiplicités qui sont apposées sur celle ci :

0..0	0	Aucune instance
0..1		Aucune ou une seule instance
1..1	1	Exactement une instance
0..*	*	Aucune, une ou plusieurs instances (aucune limite)
1..*		Au moins une instance (aucune limite maximum)
x..x	x	Exactement *x* instance(s)
m..n		Au moins *m* et au plus *n* instances

Présentations des différentes classes

Classe Commande

La classe commande contient l'ensemble des commandes. Elle est reliée aux classes Utilisateur (le client qui commande) et Ligne de commande.

Attributs :

- Commande id : numéro de commande attribué par le système lors de la création de la commande.

- Type de paiement : contient le type de paiement:
    - 1: paiement en ligne
    - 2: paiement en boutique
    - 3: paiement à la livraison
- Statut commande : contient le statut de la commande
    - 1: préparation non commencée
    - 2: préparation en cours
    - 3: préparation terminée
    - 4: commande livrée
    - 5: commande annulée
- Mode de retrait : contient le mode de retrait choisi par le client. Le type est Int et non Boolean car OCPizza pourrait rajouter des modes de livraisons.
    - 1: retrait en boutique
    - 2: livraison
    - (3: uber eat)
    - (4: to good to go)
- Paiement effectué : Contient un Boolean
    - True: la commande a été payé
    - false: la commande n'a pas été payé
- Prix : Contient le prix total de la commande

- Date : Contient la date de la commande du type Date: ‘AAAA-MM-JJ’

Classe Utilisateur

La classe utilisateur contient l’ensemble des utilisateurs, c’est a dire à la fois les employés d’OCPizza et les clients. Elle est reliée aux classes Adresse et Commande.

Attributs :

- Utilisateur id : numéro d’utilisateur attribué par le système lors de la création d’un profil utilisateur
- Prénom : prénom de l’utilisateur
- Nom : nom de l’utilisateur
- Poste : poste occupé par l’utilisateur :
    - 1 : gérant
    - 2: pizzaiolo
    - 3: livreur
    - 4: client
- Login : identifiant utilisé par l’utilisateur pour se connecter
- Mot_de_passe : mot de passe utilisé par l’utilisateur pour se connecter

Classe Pizzeria

La classe pizzéria contient la liste des pizzérias. Elle est reliée aux classes Adresse et Stock.

Attributs :

- Pizzeria id : id de la pizzeria attribué par le système lors de la création d’un profil pizzeria
- Nom : nom de la pizzeria

Classe adresse

La classe adresse contient les adresses des utilisateurs et des pizzerias, cet objet contient aussi les emails et les les numéros de téléphone.
Chaque employé est authentifié dans le système par un identifiant :

Attributs:

- Adresse id : id de l'adresse attribué par le système lors de la création d'une adresse
- Email : email de l'utilisateur ou de la pizzeria
- Telephone : telephone de l'utilisateur ou de la pizzeria
- Numéro de rue : numéro de rue de l'adresse de l'utilisateur ou de la pizzeria
- Complement numero de rue : complément du numéro de rue (bis, ter etc…). Cette valeur peut être Null
- Complement rue : complément de l'adresse
- Code postal : code postal de l'adresse
- Ville: ville

Classe ligne de commande

La classe ligne de commande contient une ligne de la commande, cela permet de gérer la commande plus facilement. Elle est reliée aux classes Commande et Pizza

Attributs:

- Ligne de commande id: id de ligne de la commande attribué par le système lors de la création d'une commande
- Prix : prix de la pizza * la quantité
- Quantité : quantité de cette pizza

Classe recette

La classe recette contient le nombre des différents ingrédients pour chaque pizza. C'est une jonction des classes Pizza et Ingrédient.

Attributs:

- Volume : volume de cet ingrédient (c est un entier car ce volume est compté en « dose »)

Classe ingrédient

La classe ingrédient contient tout les ingrédients et leurs quantités pour faire une « dose »

Attributs:

- ingredient id : id de l'ingrédient
- Nom : nom de l'ingrédient
- Volume : volume de l'ingrédient en gramme

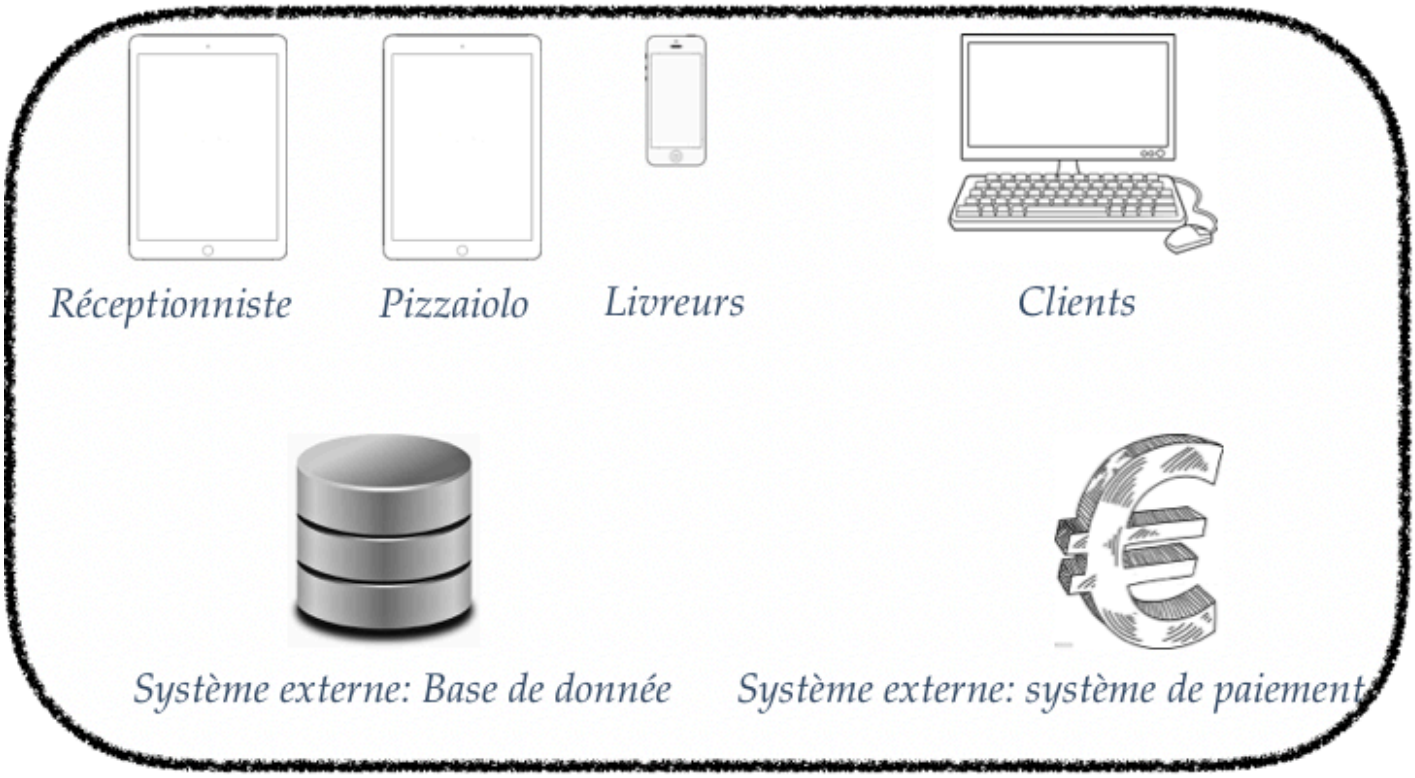
Classe stock

La classe stock contient le nombre d'ingrédients des différents types par pizzeria. C'est une jonction des classes Ingrédient et Pizzeria

Attributs:

- Volume : volume de l'ingrédient en gramme

III - Solution proposée



Choix techniques

La solution que nous vous proposons se fera à l'aide des technologies ci-dessous:

- Une application iOS pour les employés de la pizzeria (les deux iPad et l'iPhone par exemple auront accès à la même application mais pas aux mêmes fonctionnalités). Cette application sera développé en Swift. Swift est un Language de programmation aussi fiable qu'intuitif mis au point par Apple pour développer des apps iOS, Mac, Apple TV et Apple Watch.

- Un site Web adaptatif qui permettra aux clients de s'identifier, créer un compte et passer des commandes. Ce site web sera développé en HTML (qui structure et donne du sens au contenu) et en CSS (qui a pour rôle de mettre en forme le contenu). Ces deux langages ont été choisis car ce sont les plus classiques et ils sont complémentaires

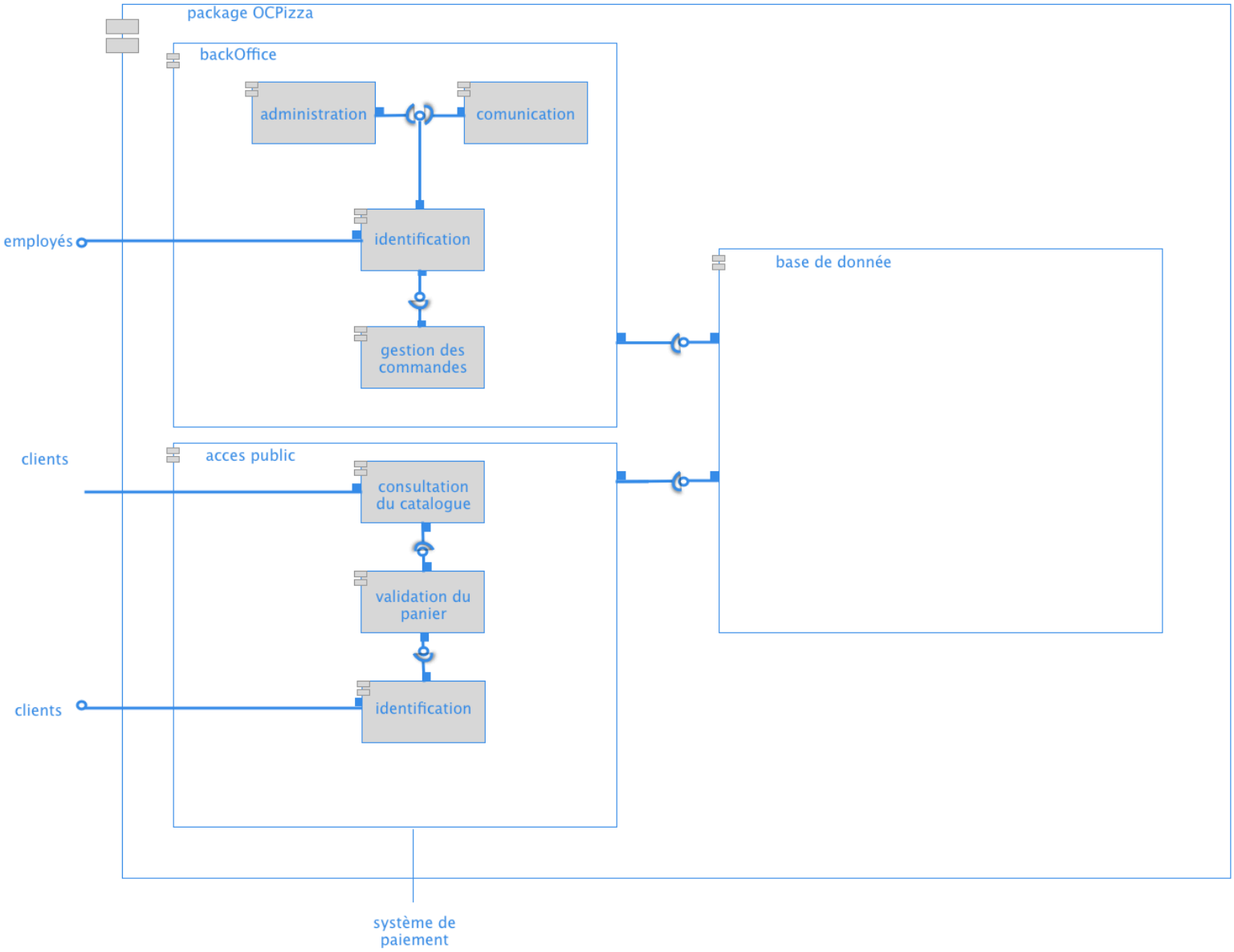
- Pour l'hébergement des données nous utiliserons Amazon de type SQL. Amazon propose un système de gestion de bases de données simplifié, il fait partie des logiciels de gestion de base de données les plus utilisés au monde et est gratuit.

- Pour traiter les paiement et compte tenu des contrainte juridiques et fiscale nous utiliserons un système d'encaissement (carte bleue, ticket restaurant, chèques et espèce) inclus dans un système comptable intégré et agréé du type Cegid.

IV Architecture Technique

Diagramme de composants

Les Diagrammes de composants illustrent les parties du logiciel qui feront partie d'un système. Il permet également de décrire les composants d'un système et les interactions entre ceux-ci.



Notre diagramme est composé de 3 modules: Accès public, backoffice et base de donnée.

Le module **accès public** est composé de 3 composants :

- Identification : Il permet l'inscription des nouveaux clients au système ainsi que leur connexion. C'est une étape obligatoire pour passer une commande.
- Validation du panier: C'est la page du site où se trouvent tout les articles sélectionnés par le client. Une fois le panier validé il devient une commande. Pour valider son panier il faut au préalable s'être identifié.
- Consultation du catalogue: C'est ce module qui permet au client de consulter les pizzas (et entrées, desserts). Il peut les sélectionner pour les faire passer dans le panier. L'identification n'est pas nécessaire pour cette étape.

Le module **backoffice** est composé de 4 composants:

- Identification : Il permet l'inscription et la connexion des employé, c'est une étape obligatoire pour accéder à n'importe quel autre module.
- Gestion des commandes : Il permet de gérer les commandes c'est à dire faire une nouvelle commande dans le cas où un client téléphone, avoir le suivis des commandes, l'affichage des recettes etc.., Toutes les fonctionnalités de ce module ne sont pas disponibles pour tous les employés.
- Administration : Il permet l'administration du système, on peut y modifier les informations liés aux clients ou modifier les stocks.
- Communication : C'est le module qui permet aux employé de communiquer entre eux. Ainsi le pizzaiolo pour informer le livreur et le gérant quand une commande est prête et le livreur pour informer le gérant quand une commande est livrée.

Le module **base de donnée** contient le composant base de donnée qui est relié aux deux autres modules. Tout y est stocké .

Le système de paiement est externe au système que nous vous proposons

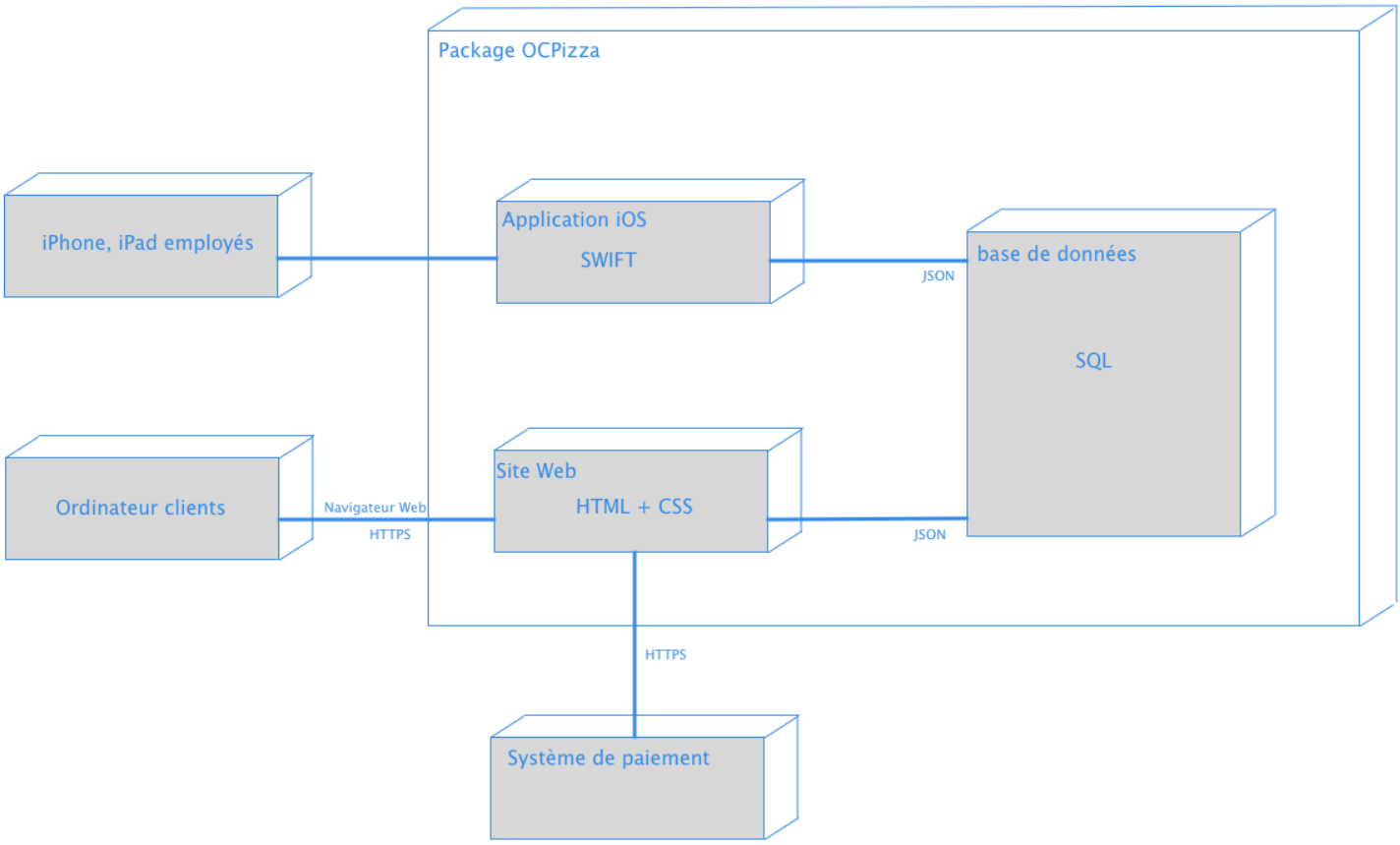
V - Architecture de déploiement

Diagramme de déploiement

Le Diagramme de déploiement est en UML, une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux.

Le diagramme de déploiement précise comment les composants sont répartis sur les nœuds et quelles sont les connexions entre les composants ou les nœuds.

La partie développée est celle nommée : package OCPizza.



VI - Base de donnée

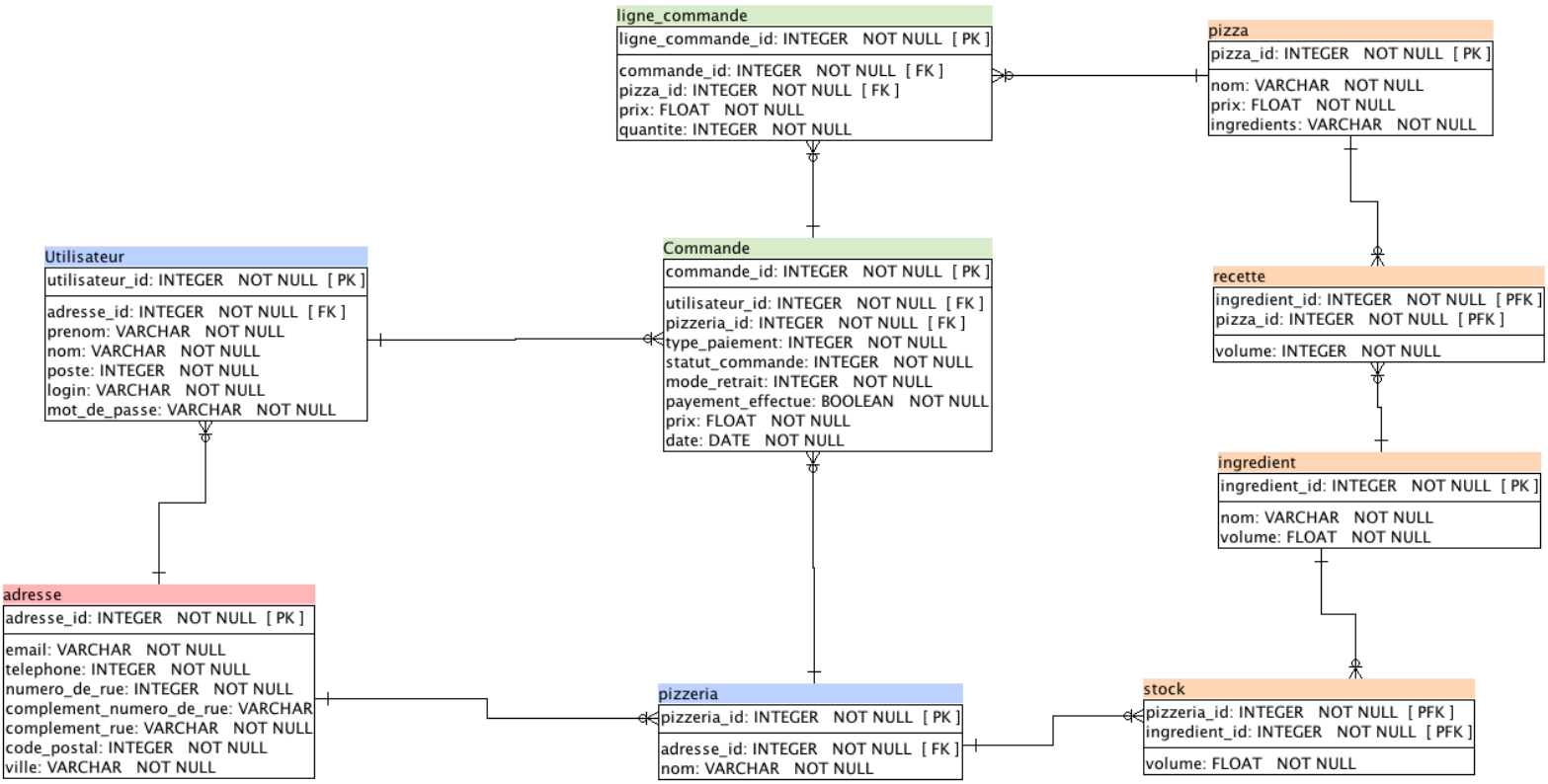
Introduction aux base de données

Techniquement, une base de données n'est que votre ensemble de données. Le système de gestion de base de données (SGBD) est le logiciel que vous utilisez pour structurer et gérer ces données.

Les système de gestion de base de donnée relationnels (SGBD-R) sont le type de SGBD le plus couramment utilisé. Ils sont basés sur un modèle relationnel. Nous utiliserons postgreSQL. PGAdmin est l'outil graphique qui permet d'administrer postgreSQL et de lancer des scripts SQL. Afin d'exécuter des requêtes SQL nous utiliserons le logiciel SQuirrel SQL.

Les étapes pour la création de notre base de donnée :

- Modélisation du domaine fonctionnel avec UML (Diagramme de classe)
- Création du modèle physique de donnée (SQL Power Architect)
- SQL Power Architect génère le script de création du schéma de base de donnée
- Création de la base de donnée sur PGadmin III grâce au script de création
- Implémentation et interrogation des tables de notre base de donnée sur Squirrel SQL



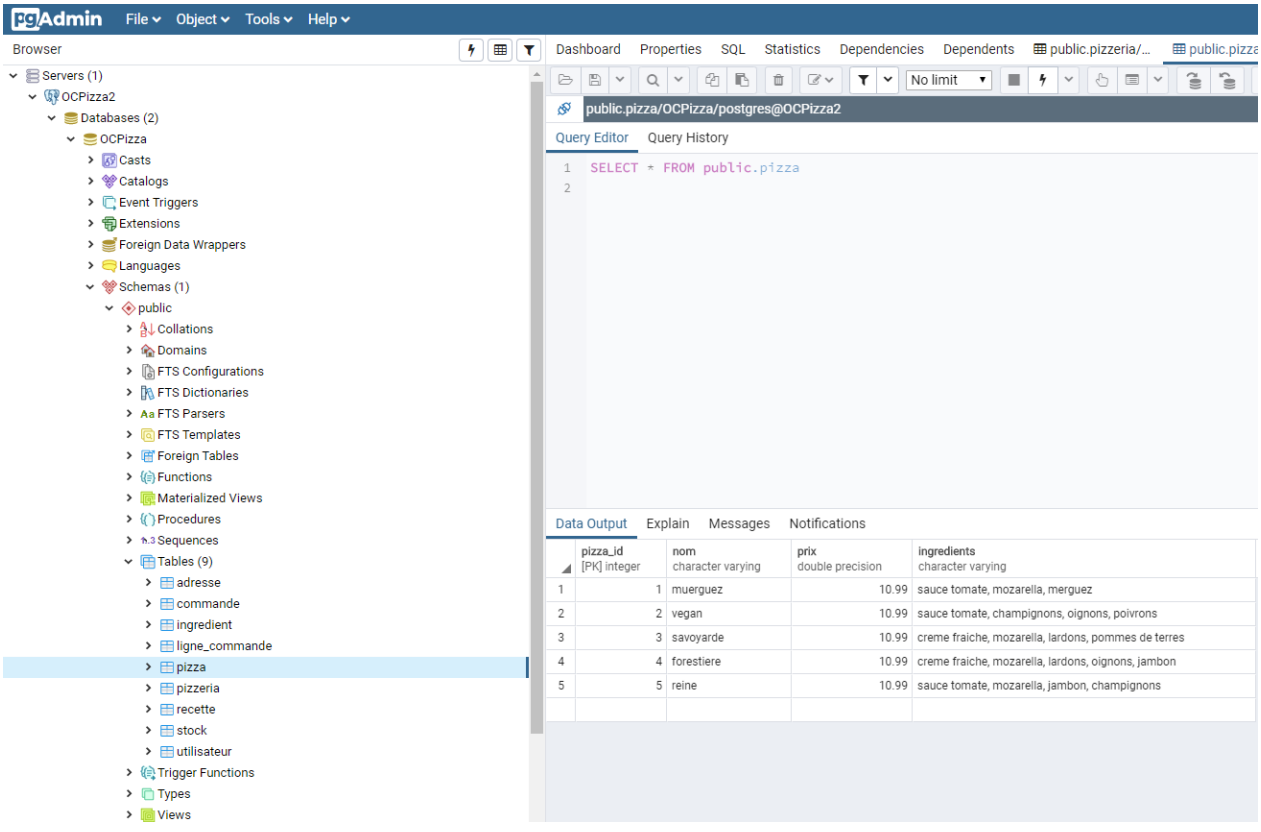
Le modèle Physique des Données ci-dessus a été conçu à partir du diagramme de classe. Il permettra la création de notre base de données relationnelles à partir d'un script issu de ce modèle. J'ai utilisé le logiciel SQL Power Architect pour élaborer ce modèle physique de donnée et pour générer le script de création de notre base de donnée

Grâce au MPD j'ai généré le script de création avec SQL Power Architect

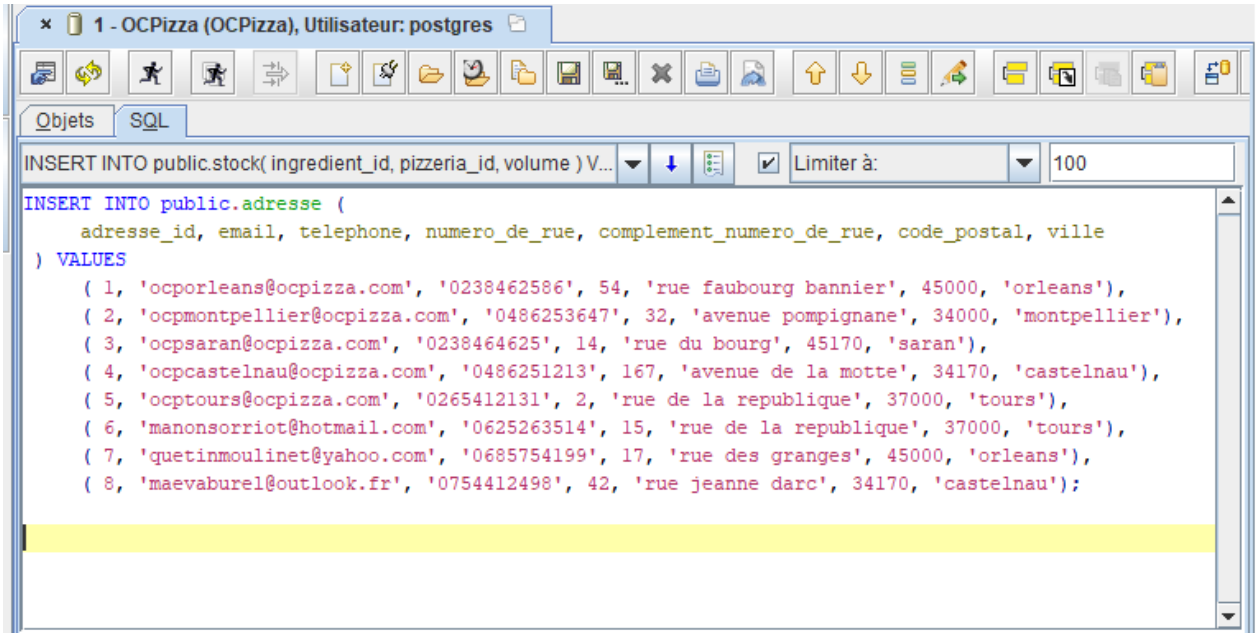
script creation.TXT

```
CREATE TABLE public.ingredient (
                ingredient_id INTEGER NOT NULL,
                nom VARCHAR NOT NULL,
                volume REAL NOT NULL,
                CONSTRAINT ingredient_pk PRIMARY KEY (ingredient_id)
);


CREATE TABLE public.pizza (
                pizza_id INTEGER NOT NULL,
                prix REAL NOT NULL,
                nom VARCHAR NOT NULL,
                ingredients VARCHAR NOT NULL,
                CONSTRAINT pizza_pk PRIMARY KEY (pizza_id)
);


CREATE TABLE public.recette (
                pizza_id INTEGER NOT NULL,
                ingredient_id INTEGER NOT NULL,
                volume REAL NOT NULL,
                CONSTRAINT recette_pk PRIMARY KEY (pizza_id, ingredient_id)
);


CREATE TABLE public.adresse (
                adresse_id INTEGER NOT NULL,
                email VARCHAR NOT NULL,
                telephone VARCHAR NOT NULL,
```

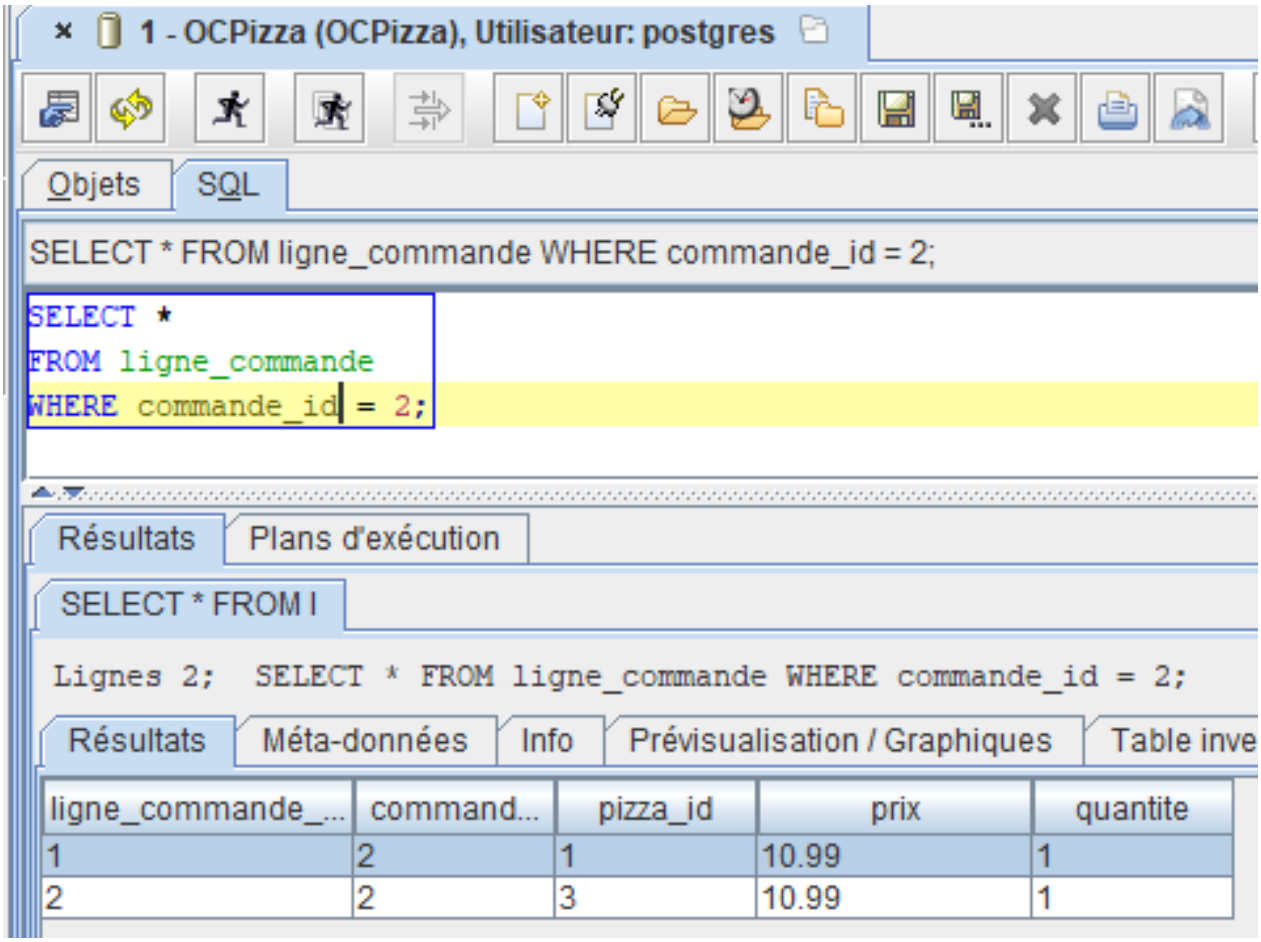
Sur PGAdmin4 j'ai pu exécuté ce script afin de générer nos tables dans notre base de donnée



Dans Squirrel SQL on peut ensuite ajouter des valeurs dans nos tables grâce à la commande « insert ».



Dans Squirrel SQL on peut aussi afficher des valeurs de nos tables grâce à la commande « select».



```
INSERT INTO public.pizza(
       pizza_id, prix, nom, ingredients
 ) VALUES
       ( 1, 10.99, 'muerguez', 'sauce tomate, mozarella, merguez'),
       ( 2, 10.99, 'vegan', 'sauce tomate, champignons, oignons, poivrons'),
       ( 3, 10.99, 'savoyarde', 'creme fraiche, mozarella, lardons, pommes de terres'),
       ( 4, 10.99, 'forestiere', 'creme fraiche, mozarella, lardons, oignons, jambon'),
       ( 5, 10.99, 'reine', 'sauce tomate, mozarella, jambon, champignons');
```

```
INSERT INTO public.stock(
        ingredient_id, pizzeria_id, volume
 ) VALUES
        ( 1, 1, 2000),
        ( 2, 1, 2000),
        ( 3, 1, 2000),
        ( 4, 1, 2000),
        ( 5, 1, 2000),
        ( 6, 1, 2000),
        ( 7, 1, 2000),
        ( 8, 1, 2000),
        ( 9, 1, 2000),
        ( 10, 1, 2000),
        ( 1, 4, 2000),
        ( 2, 4, 2000),
        ( 3, 4, 2000),
        ( 4, 4, 2000),
        ( 5, 4, 2000),
        ( 6, 4, 2000),
        ( 7, 4, 2000),
        ( 8, 4, 2000),
        ( 9, 4, 2000),
        ( 10, 4, 2000);
```

```
INSERT INTO public.ingredient(
        ingredient_id, nom, volume
 ) VALUES
        ( 1, 'sauce tomate', 100),
        ( 2, 'creme fraiche', 100),
        ( 3, 'lardons', 60),
        ( 4, 'jambon', 60),
        ( 5, 'merguez', 60),
        ( 6, 'champignons', 60),
        ( 7, 'oignons', 40),
        ( 8, 'poivrons', 60),
        ( 9, 'mozarella', 60),
        ( 10, 'pomme de terre', 60);
```

pizza_id	prix	nom	ingredients
1	10.99	muerguez	sauce tomate, mozarella, merguez
2	10.99	vegan	sauce tomate, champignons, oignons, poivrons
3	10.99	savoyarde	creme fraiche, mozarella, lardons, pommes de terres
4	10.99	forestiere	creme fraiche, mozarella, lardons, oignons, jambon
5	10.99	reine	sauce tomate, mozarella, jambon, champignons

ingredient_id	pizzeria_id	volume
1	1	2000.0
2	1	2000.0
3	1	2000.0
4	1	2000.0
5	1	2000.0
6	1	2000.0
7	1	2000.0
8	1	2000.0
9	1	2000.0
10	1	2000.0
1	4	2000.0
2	4	2000.0
3	4	2000.0
4	4	2000.0
5	4	2000.0
6	4	2000.0
7	4	2000.0
8	4	2000.0
9	4	2000.0
10	4	2000.0

ingredient_id	nom	volume
1	sauce tomate	100.0
2	creme fraiche	100.0
3	lardons	60.0
4	jambon	60.0
5	merguez	60.0
6	champignons	60.0
7	oignons	40.0
8	poivrons	60.0
9	mozarella	60.0
10	pomme de terre	60.0