# Projet 6 : Concevez la solution technique d'un système de gestion de pizzeria

Dossier des spécifications techniques

**Client : OC PIZZA** **Auteur : Yannick Driever** **Date : 26/12/2020**

# SOMMAIRE

SOMMAIRE 2

INTRODUCTION 3

RAPPEL DU CONTEXTE 3

DEFINITION DES BESOINS CLIENT 3

CONCEPTION DE L'ARCHITECTURE 4

DESCRIPTION DU DOMAINE FONCTIONNEL 6

MODELE PHYSIQUE DE DONNEES 7

BASE DE DONNEES 8

# INTRODUCTION

L'objet de ce document est de définir les spécifications techniques détaillées de la future application.
Les spécifications techniques détaillées ont pour but de définir le domaine fonctionnel afin de modéliser ses objets, mais également de décrire l'architecture technique :

- identifier les différents éléments qui vont composer le système ainsi que leurs interactions
- décrire le déploiement des différents composants
- élaborer le schéma de la base de données

# RAPPEL DU CONTEXTE

« OC Pizza » est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà cinq points de ventes et prévoit d'en ouvrir au moins trois de plus d'ici la fin de l'année.
Cependant, le système actuel est devenu obsolète et ne réponds plus aux besoins croissants de notre client, qui est en recherche de modernité.
Les responsables du groupe ont déjà prospectés des logiciels existants sur le marché mais n'ont pas trouvé leur bonheur.
Ils nous demandent donc de concevoir une solution personnalisée, adaptée à leurs nouveaux besoins.

# DEFINITION DES BESOINS CLIENT

Notre client nous demande de déployer dans toutes ses pizzerias un système informatique capable :

- d'être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation
- de suivre en temps réel les commandes passées et en préparation
- de suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas sont encore réalisables
- de proposer un site Internet pour que les clients puissent passer leur commande et la payer en ligne (ou à la livraison)
- de modifier ou annuler leur commande tant que celle-ci n'a pas été préparée
- d'informer ou notifier les clients sur l'état de leur commande
- de proposer un aide mémoire aux pizzaïolos indiquant la recette de chaque pizza

# CONCEPTION DE L'ARCHITECTURE

En adéquation avec les besoins client et à l'aide du diagramme de composant, nous avons décidé de découper le système en 6 composants principaux, qui contiennent eux-mêmes des sous composants qui vont interagir entre eux et mettre en évidence leur dépendance.
Les composants Client, Employé et Administration sont dépendants du composant Authentification pour accéder au système.
Le composant Employé reçoit les commandes du composant Client, il envoie ensuite les informations nécessaires au composant Administration afin de pouvoir mettre à jour le composant Stock.
Le composant Système bancaire fournit les interfaces de paiement requises par les composants Client et Employé.

**Figure 1 : Diagramme de composants**

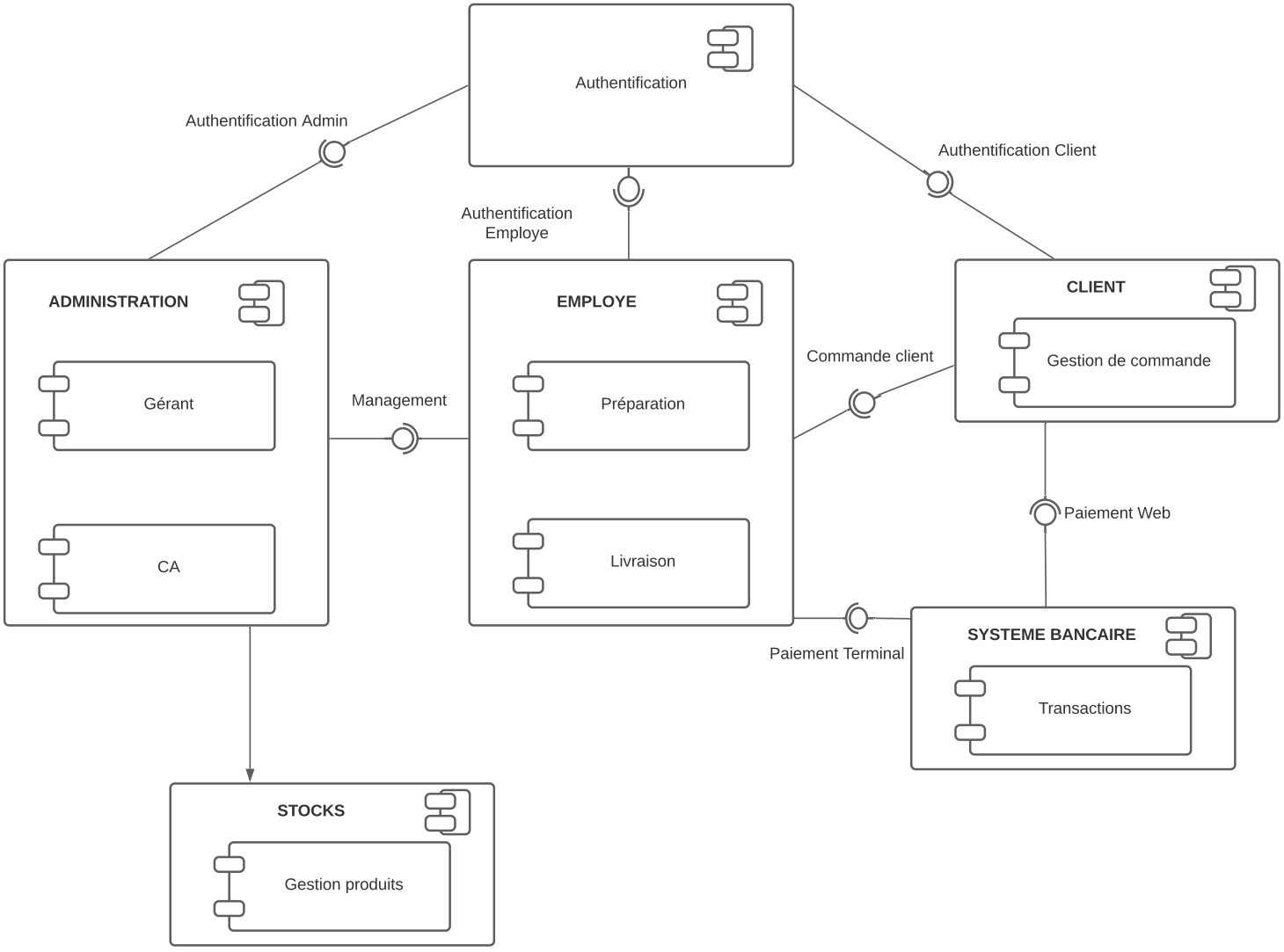
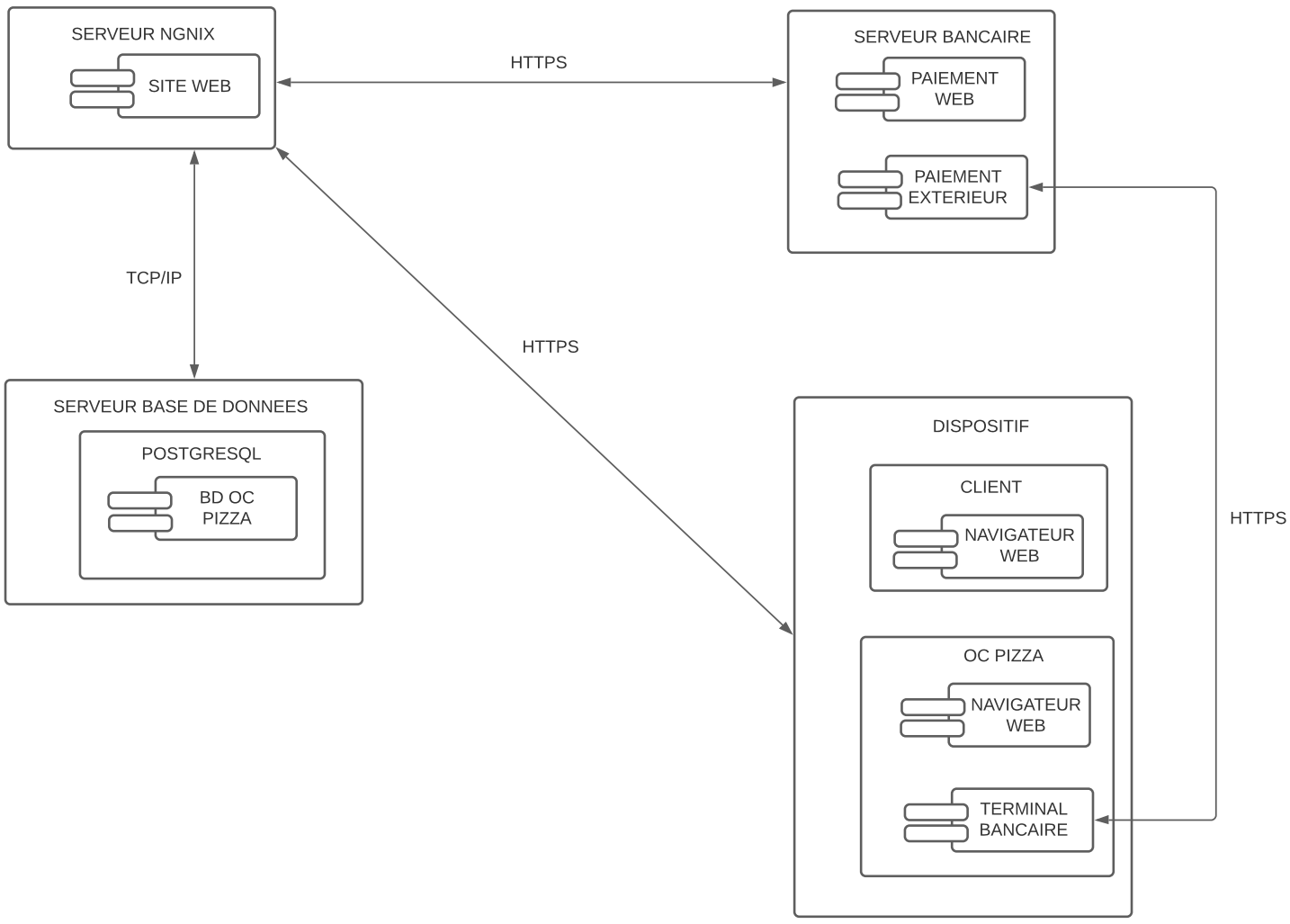


**Diagramme de déploiement**

Nous avons décidé de créer 4 nœuds sur lesquelles déployer le futur système de gestion de pizzerias :

- un serveur NGNIX qui sera chargé d'héberger le futur site web OCPizza.fr
- un serveur POSTGRESQL pour la base de données OCPizza - un serveur bancaire pour effectuer les transactions de paiement
- un dispositif de déploiement comprenant un navigateur web pour les clients ainsi que le personnel OCPizza, sans oublier d'ajouter un terminal bancaire au livreur pour le paiement à la livraison. Les différents nœuds communiqueront ensemble par liaison HTTPS (liaison web sécurisée pour les utilisateurs), seul la liaison entre le serveur NGINX et la base de données POSTGRESQL sera de type TCP/IP (échange de données entre machines). Lors de la conception de l'application, on prévoira un déploiement multi-supports (PC, tablettes, smartphones).

**Figure 2 : Diagramme de déploiement**



# DESCRIPTION DU DOMAINE FONCTIONNEL

A l'aide du diagramme de classe, nous avons identifié les parties qui composeront l'application mais surtout leurs relations ainsi que leurs cardinalités ou multiplicités.
Nous avons répertorié un total de **11 classes** :

- un **point de vente** emploie du **personnel** (3 ou plus), il est composé d'un stock qui est lui même composé de plusieurs **produits** et **ingrédients**. Il vend les commandes au client
- une **commande** est payée par le **client**, elle contient **l'adresse** du client, mais aussi les **produits** sélectionnés qui seront listés en **ligne de commande**. Elle est envoyée en **livraison**
- le **client** paye une **commande**, possède une **adresse** et reçoit une **facture**
- les classes **client** et **personnel** hérite de la classe **utilisateur**

Il existe une correspondance entre les classes UML et les tables du MPD, ce qui nous amènera naturellement à la réalisation du modèle physique de données

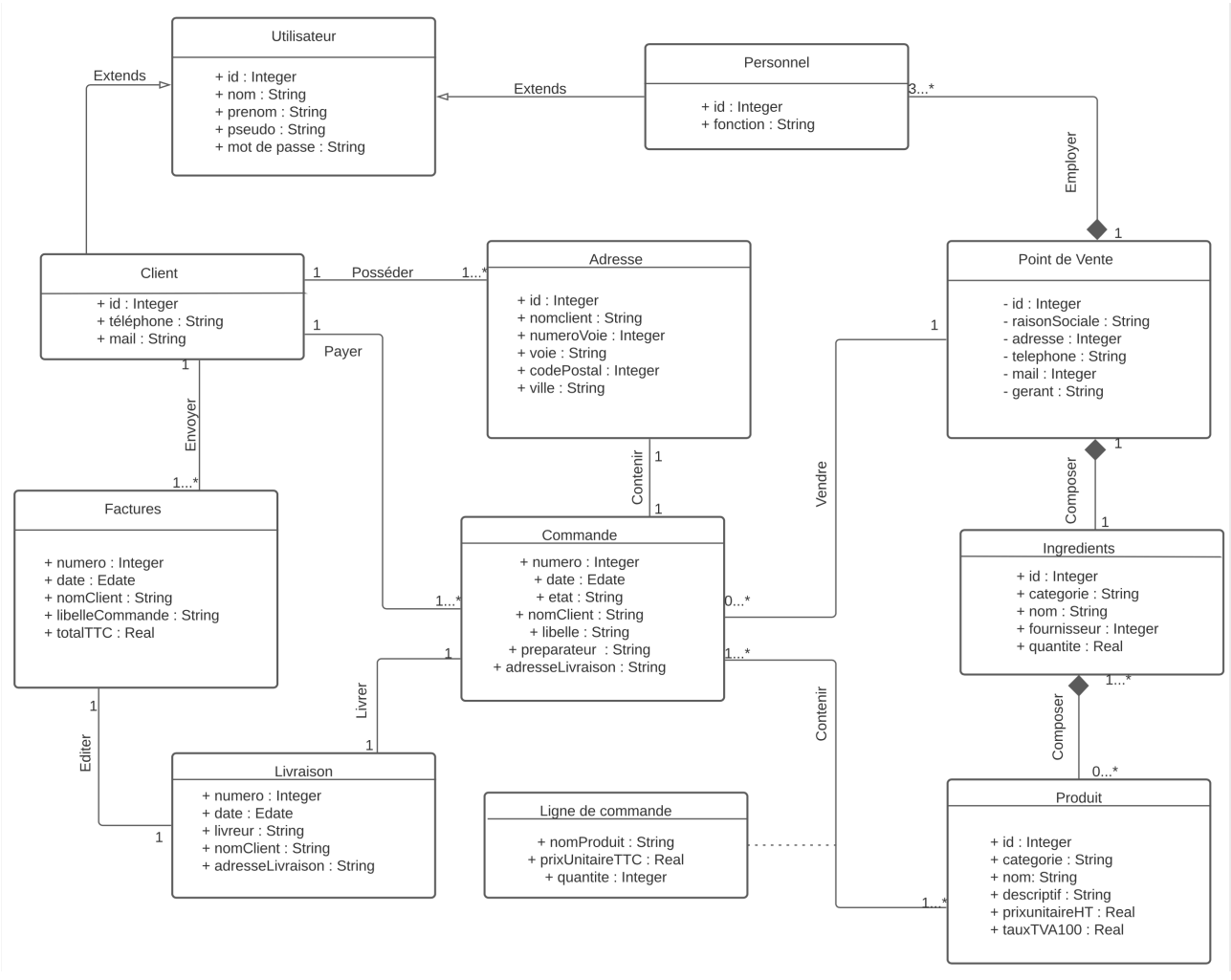
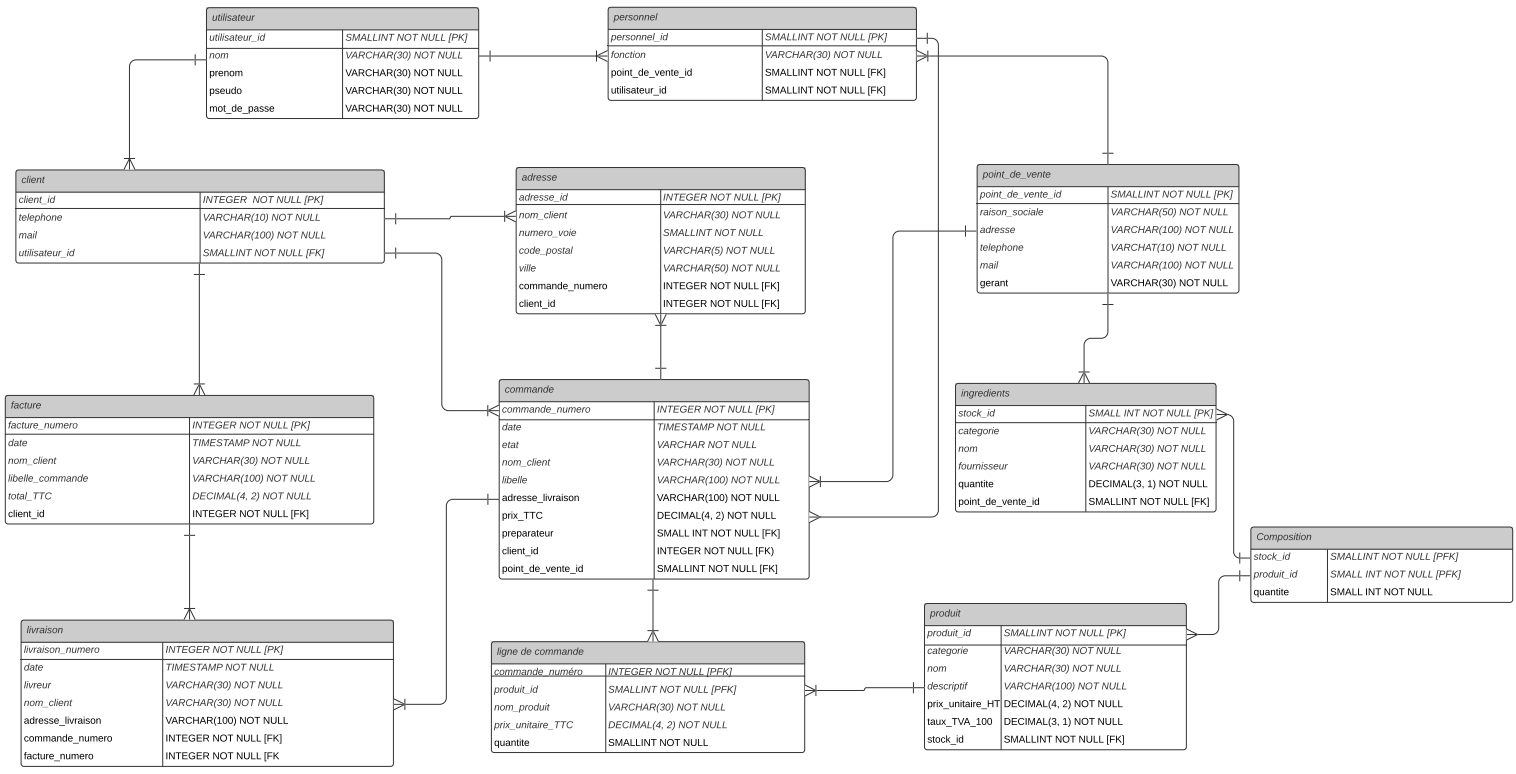


**Figure 3 : Diagramme de classes**

# MODELE PHYSIQUE DE DONNEES

Le modèle physique de données (MPD) permet d'avoir une représentation graphique de la structure d'une base de données et ainsi mieux comprendre les relations entre les différents tables grâce aux clés primaires et étrangères.
Il va également mettre en évidence les différentes données ainsi que leur typage.
Cette étape de réalisation graphique est indispensable pour l'implémentation de la base de données dans un SGBD-R comme POSTGRESQL par exemple.

**Figure 4 : Modèle physique des données**



# BASE DE DONNEES

## Présentation

Notre choix s'est orienté vers la base de données POSTGRESQL, c'est un système de gestion de base de données relationnel (SGBD-R), c'est à dire que les données seront stockées dans des tableaux pouvant être lies entre eux, à la différence des SGBD NoSQL où les données sont structurées en clé-valeur ou orienté graphe.

## Avantages

POSTGRESQL possèdent des outils riches et variés (interface graphique, bibliothèques, API …) ce qui en fait un SGBD-R de qualité, robuste et puissant. Il présente également l'avantage non négligeable d'être libre, c'est-à-dire gratuit et dont les sources sont disponibles.

## Fonctionnement

Il fonctionne en architecture client/serveur :

- une partie serveur constituée d'une application interne (PostMaster), capable de traiter et répondre aux requêtes SQL
- une partie client, installée sur les machines devant accéder au serveur de la base de données, afin de permettre à notre client de l'interroger à l'aide de requêtes SQL

Un client peut éventuellement fonctionner sur le serveur lui-même.