

# **Spécifications techniques**

Guillaume BOURLART

## **Sommaire**

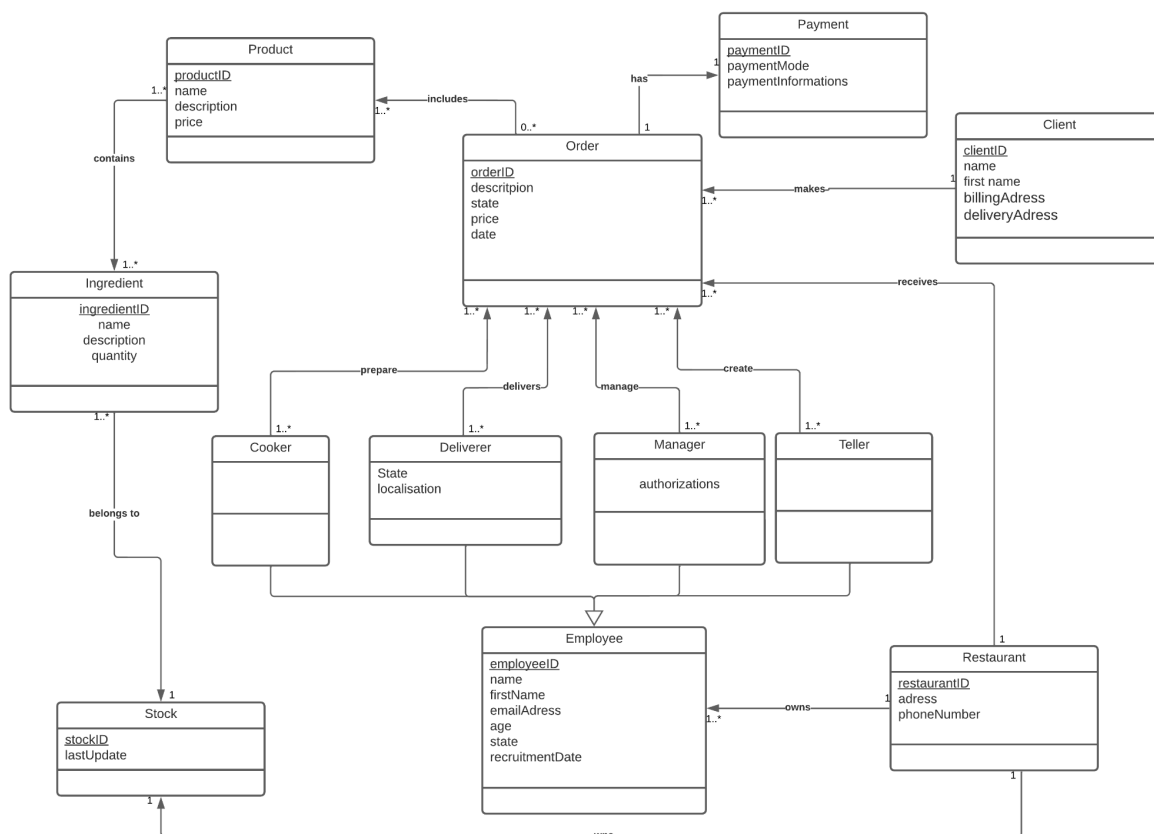
I.	Base de données.....	3
	A. Modélisation.....	3
	B. Base de données.....	5
II.	Partie client.....	6
	A. Version web.....	6
	B. Version mobile.....	7
III.	Partie restaurant.....	8
	A. Version pour le système POS.....	8
	B. Version mobile.....	8
IV.	Équipements.....	8
V.	Architecture finale du système informatique.....	9

# I - Base de données

## A. Modélisation

Qui dit système informatique, dit évidemment base de données pour le stockage de données. Commençons donc par la modéliser. Cette base de données sera reliée à toutes les applications conçues pour le système.

### ❖ UML



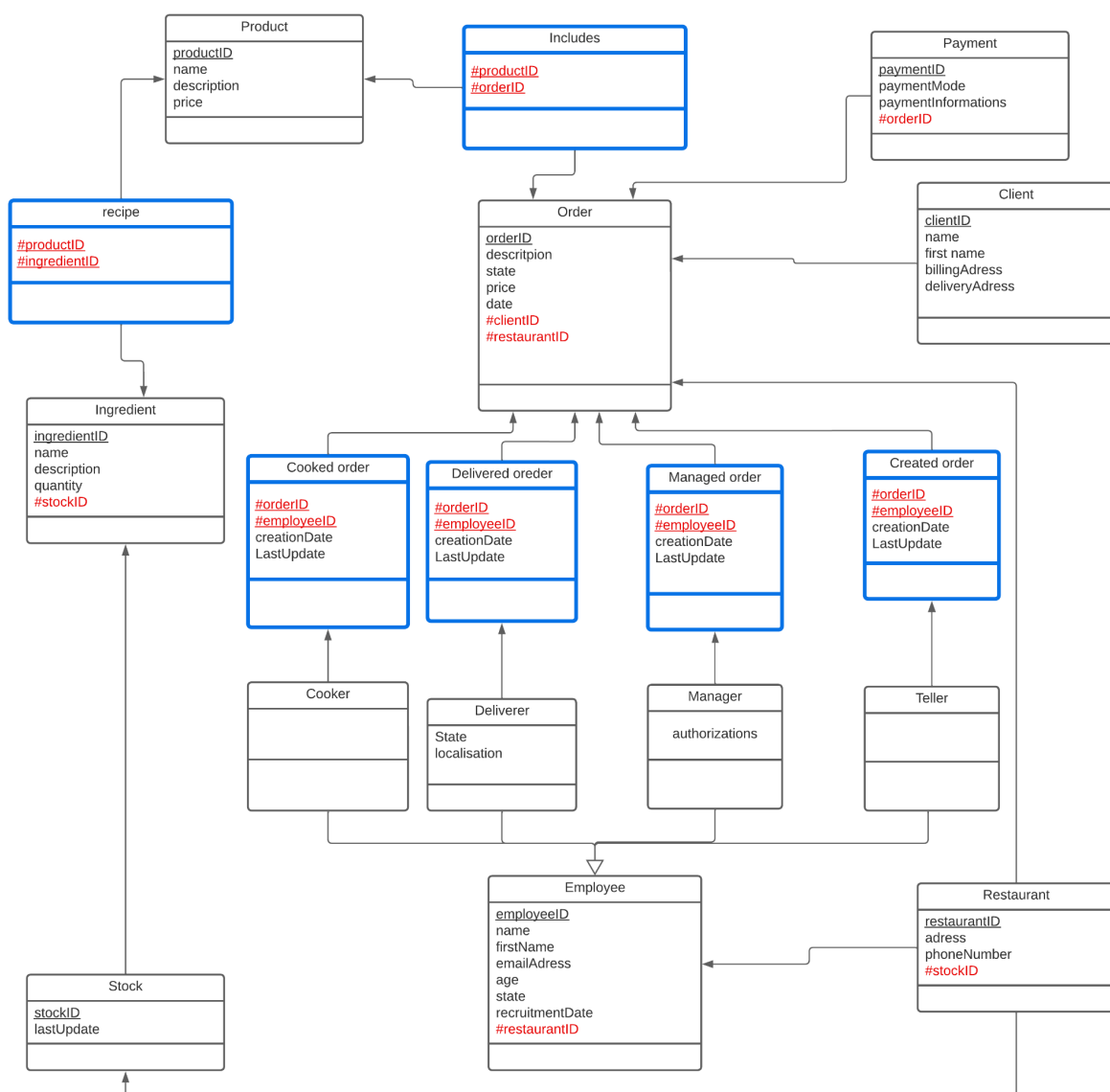
<u>soulignement</u>	clé primaire
1..*	cardinalité "Un ou plusieurs"
0..*	cardinalité "Zéro ou plusieurs"
1	cardinalité "Un"

explication des différentes table :

Un restaurant possède des employés qui peuvent être cuisinier, livreur ou caissier. Ces derniers peuvent respectivement préparer, livrer ou gérer une commande passée par un client. Il y a aussi le gérant qui, lui, a tous les droits. La commande possède un paiement et des produits, constitués pour certains d'ingrédients entreposés dans le stock du restaurant.

Pour le suivi de l'état de la commande et de sa livraison, nous allons récupérer d'une part la localisation du livreur et la mettre à jour dans la base de données, et d'autre part l'état de la commande qui sera mis à jour en temps réel.

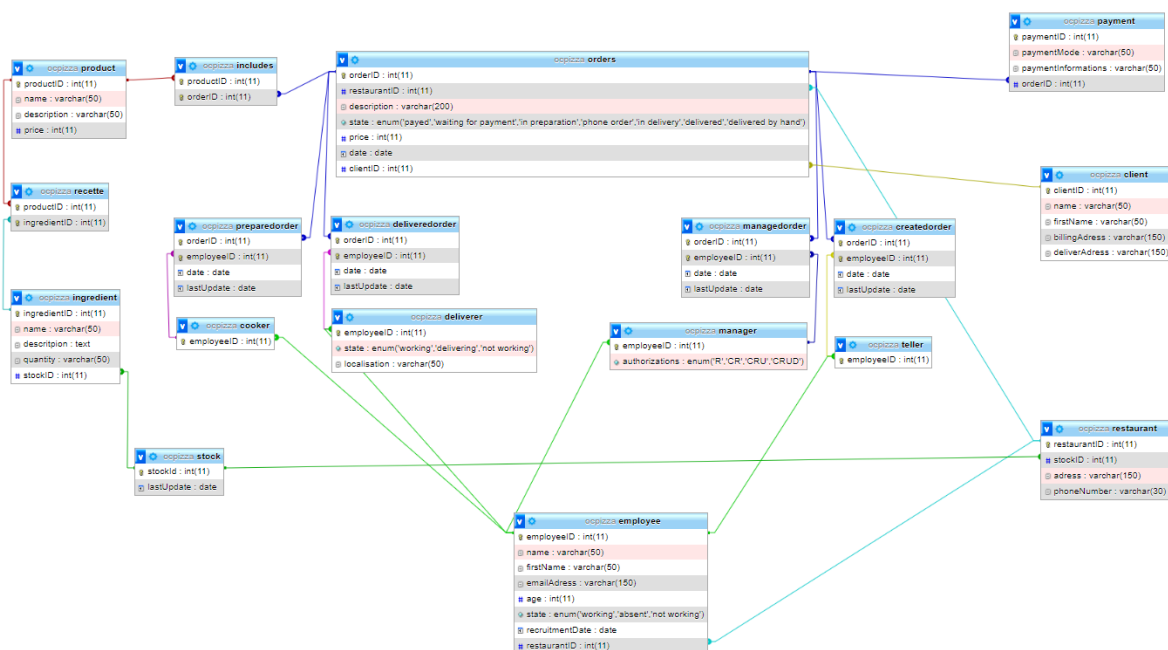
### ❖ MPD :







<u>soulignement</u>	clé primaire
#	clé étrangère
<u>#soulignement</u>	clé primaire et étrangère

Dans ce **Modèle physique de Données**, toutes les cardinalités **1..\* - 1..\*** ont donné naissance à de nouvelles tables (ici en bleu) récupérant chacune les clé primaires des tables avec lesquelles elles sont en relation.

### ❖ Export de la BDD :



	Date
	Varchar, text
	Integer
	Primary key

## B. Base de données

Deux types de base de données existent : relationnelle et non relationnelle.

### ❖ Base de données relationnelle (SQL)

#### ➤ Avantages

- Données structurées
- Cohérente
- Sécurisée
- Possibilité de gérer des droits d'accès différents à la base de données et aux données
- Base de données scalable

#### ➤ Inconvénients

- Pas adaptée aux données non structurées
- Le fait de structurer ses données peut rendre difficile et très complexe l'évolution de votre base de données dans le futur
- Le fait de connaître le langage SQL pour pouvoir utiliser ce moteur de base de données

### ❖ Base de données non relationnelle (NoSQL)

#### ➤ Avantages

- Non structurée et semi-structurée
- Flexibilité de votre base de données et de vos données
- Rapidité de prise en main
- Rapidité plus forte dans l'écriture des données que le relationnel

#### ➤ Inconvénients

- La sécurité et la cohérence des données sont sacrifiées même si une équipe de développeurs peut le minimiser, il y aura toujours un risque.
- Le regroupement et le croisement de données sont plus difficiles à réaliser qu'en SQL

Puisque nous avons des données structurées, dont nous connaissons les tables et caractéristiques et que nous voulons opter pour la sécurité, nous choisirons une base de données SQL, donc relationnelle.

Pour que les programmes que nous allons développer puissent interagir avec la base de données, nous avons besoin d'un SGBD (Système de Gestion de Base de Données).

Voici donc les SGBD les plus connus et utilisés :

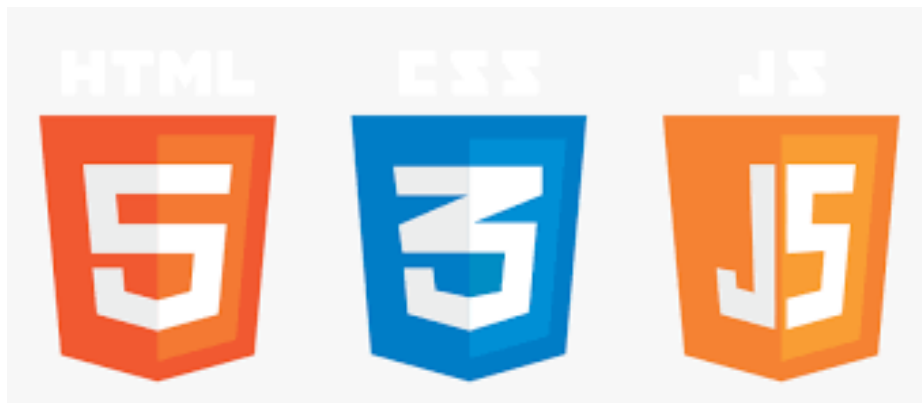
	Oracle	MySQL	PostgreSQL	SQL Server	MongoDB
--	--------	-------	------------	------------	---------

Difficulté d'utilisation	Difficile	Facile	Facile	Facile	Facile
Open source	✗	✓	✓	✗	✓
Performances	Très bonnes	Bonne performances (jusqu'à un certain nombre de données)	Mauvaises	Manque de compatibilité	Très bonnes, mais plus grosse utilisation mémoire

Puisque MySQL est la base de données la plus utilisée au monde, qu'elle est facile à installer, à utiliser et surtout qu'elle est gratuite ; c'est cette dernière qui sera utilisée malgré des capacités réduites au bout d'un certain nombre de données. Il ne s'agit pas là de lancer une énorme base de données mais une base de données plutôt simple pour des pizzerias, c'est-à-dire avec une quantité raisonnable de données stockées.

## II - Partie client

### A) Version web



Pour le site web, les technologies incontournables que sont HTML, CSS et Javascript seront utilisées pour le développement. Au lieu d'utiliser ces technologies en Vanilla, c'est-à-dire sans framework, nous allons utiliser un framework.

Parmi les meilleurs frameworks se trouvent : **Laravel, Vue JS, React JS, Angular, Symfony, Ruby on rails, Django, et Express**, etc.

Chaque framework possède ses avantages et inconvénients, mais certains sont plus faciles d'apprentissage comme Express par exemple.

Certains sont aussi plus adaptés à du front end tandis que d'autres sont orientés back end.

Le mieux serait donc d'opter pour un framework orienté front end (design des applications), car nous n'aurons pas besoin d'une gestion poussée pour le back end (côté serveur du système).

**Angular, Vue** ou **react** seraient donc des choix judicieux pour un système performant et optimisé pour le design côté client.

Pour la mise à jour en temps réel, nous utiliserons **AJAX**:

AJAX (Asynchronous JavaScript + XML) est un ensemble de technologies existantes qui, une fois combinées, rendent les applications Web capables de réaliser des mises à jour rapides de l'interface utilisateur sans devoir recharger la page entière du navigateur.



## **B) Version mobile**

3 possibilités s'offrent à nous pour le développement de cette partie sur mobile :

1. Les applications natives, qui sont développées spécifiquement pour un des systèmes d'exploitation utilisés par les smartphones et tablettes (iOS, Android, etc.)
2. Les applications hybrides, qui sont multiplateforme grâce à une combinaison de langage web et natif.
3. Les applications web, qui sont multiplateforme grâce à un développement via langage web, accessible depuis n'importe quel mobile.

Le plus simple et le moins coûteux en temps et en ressources serait simplement d'adapter le site web en application web.

Il sera quelque peu moins performant, mais cela sera minime et peu important pour ce type d'application et, pour un lancement, c'est suffisant.

Par la suite, en fonction du budget et du besoin, nous pourons changer et améliorer l'application web.

## **III - Partie restaurant**

### **A) Version pour le système POS**

En ce qui concerne la partie gestion destinée au restaurant, un développement natif sur ios ou android (respectivement Swift ou Kotlin) permettrait d'avoir un logiciel très rapide et performant, mais cela serait aussi beaucoup plus coûteux.

La deuxième option serait donc de faire ce système de gestion sous forme de site web, afin d'utiliser les mêmes technologies que pour le site web dédié aux clients.

Cela règlera deux problèmes :

- Liberté de choix dans la sélection d'un système POS
- les coût seront moins élevés (maintenance, développement)

### **B) Version mobile**

De la même façon que pour la partie client, le site sera adapté en application web pour les terminaux tels que les téléphones ou les tablettes.

Cela permettra notamment au gérant d'accéder au système à distance en cas de besoin, et aux livreurs d'accéder aux systèmes pour travailler depuis leur téléphone.

## **IV - Equipements**

Pour les équipements du système POS, il y aura besoins de :

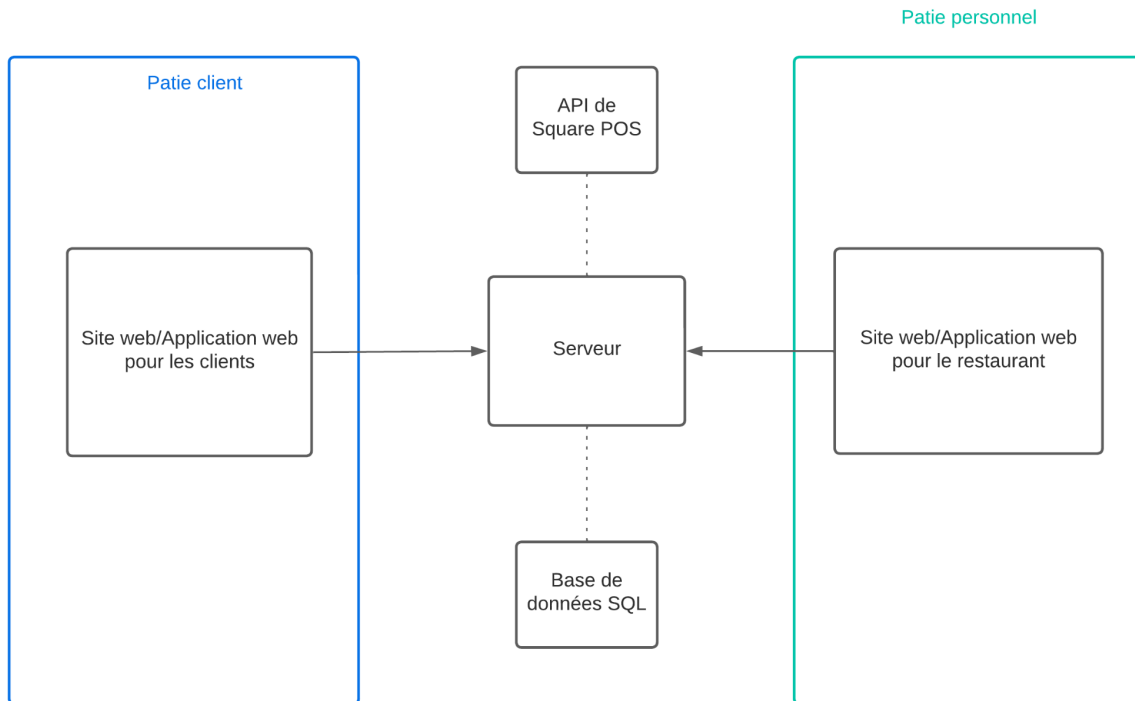
- Tablettes pour les prise de commandes
- Caisses
- Moyen de paiement (carte bleue, etc...)
- Ecran pour le menu (destiné aux clients)
- Tablette pour le cuisinier

Square POS est une marque qui met à disposition des équipements de système POS, notamment la caisse pour les paiements sur place ainsi qu'une API permettant de lier l'application web de gestion au système de paiement de Square POS.

Cela permet d'avoir un écosystème permettant d'accepter les paiements physique et en ligne.

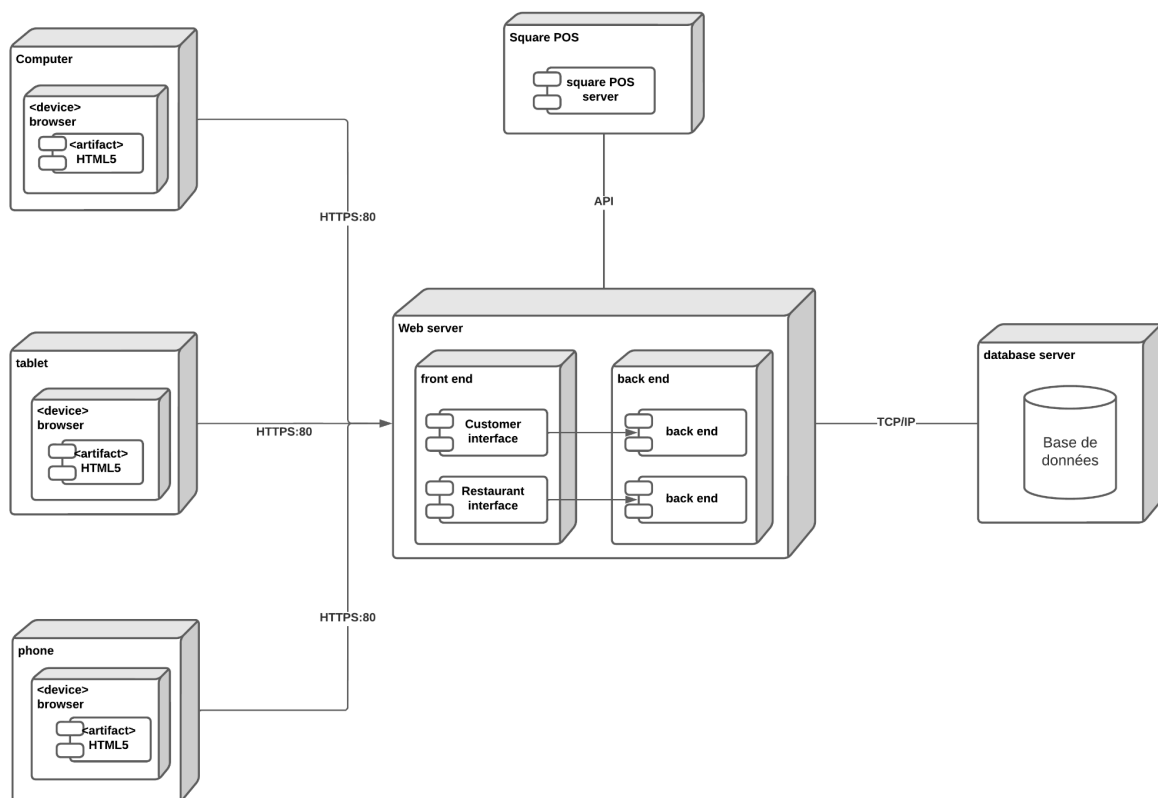
Les écrans et tablettes pourront être android ou iOS puisqu'il s'agira simplement d'accéder au site web du restaurant. Seule une connexion internet sera donc requise.

## V - Architecture finale du système informatique



Le réseau sera donc doté d'une base de données, qui communiquera avec les sites et applications web des deux côtés (client et restaurant).

Elle communique aussi avec la base de données et l'API de Square pour les paiements afin d'assurer le bon déroulement du système informatique.



Pour finir, voici donc le diagramme de déploiement. C'est sur un serveur web que seront stockées les applications web dédiées au restaurant et aux clients. Depuis leurs terminaux, les utilisateurs auront accès à la partie front de ces dernières, tandis que la partie back communiquera avec la ou les potentielles API utilisées ainsi qu'avec la base de données.