OC PIZZA

Système de gestion Pizzeria

Dossier de conception technique

Version 1.0

Auteur
Fardi ISSIHAKA
Analyste-programmeur

TABLE DES MATIÈRES

1. VERSIONS

Auteur	Date	Description	Version
ISSIHAKA Fardi	01/11/2021	Création 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, destiné à l'équipe technique qui aura pour mission de mettre en place la structure logiciel et développement de l'application.

Le document a pour objectif d'apporter une vision claire dans la mise en place de l'application notamment les technologies utilisés et les manières dont elles vont interagir ensemble.

Les éléments du présents dossiers découlent :

- Du cahier des charges du client OC Pizza
- Des besoins exprimés par le clients
- Du dossier de conception fonctionnelle

2.2.Références

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

1. **DCT - PDOCPizza_01_fonctionnelle** : Dossier d'exploitation fonctionnel du projet
2. **DCT - PDOCPizza_O3_exploitation** : Dossier d'exploitation du projet

3.ARCHITECTURE TECHNIQUE

La mise en place de notre application va déprendre de deux architectures techniques différentes :

- Le Front-end : C'est à dire l'interface visible par le client.
- Le back-end : La partie immergé piloté par Node JS

3.1.Application Web

L'application web fait partie du package « **Gestion des ventes** ». Elle est utilisé par les visiteurs, les clients et les employés pour effectuer les commandes. C'est la partie front-end de notre application. Le point d'entrée de ce package se fait par l'authentification des utilisateurs au préalable avec une adresse électronique et un mot de passe.

Ensuite l'application interagit avec le back-end pour créer un nouveau client (utilisateur) et sauvegarder ce dernier dans la base de donnée avec ses informations.

Une fois l'authentification effectif, les produits sont affichés dans l'application à l'aide de stock-api et par la même occasion mettre à jour le stock des produits dans l'inventaire.

Une interface avec le système de paiement (**Stripe** / **PayPal**) permet d'effectuer les autorisations et valider les paiements par carte bancaire sur l'application.

La pile logicielle est la suivante :

- Application **HTML** (5) / **CSS** (3) / **Javascript** (ES6) / **BootStrap (4.4)** / **MySQL** (5.6)
- Serveur d'application **OVH**

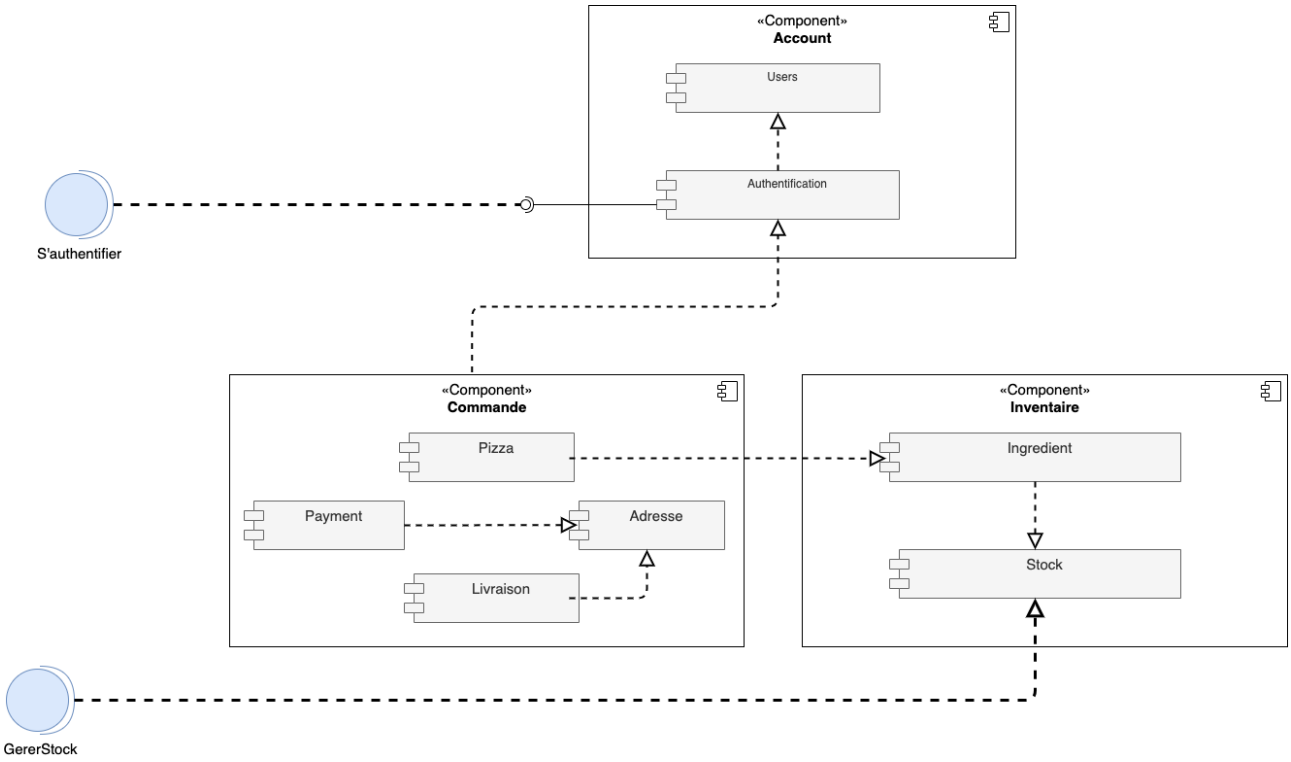
Toujours dans le package; « **Gestion des ventes** », les employées vont aussi pouvoir interagir avec l'application pour ; effectuer une commande devant le client, consulter et suivre les commandes, consulter les recettes (aide mémoire), valider ou annuler une commande. Le livreur peut aussi consulter les commandes en attente de livraison, utiliser le GPS pour se rendre chez le client, encaisser le client (si nécessaire) et valider ou annuler une commande.

Les requêtes passent par une API qui récupère la liste des commandes en cours ensuite effectue la mise à jours des statuts des commandes directement depuis le back-end.

Une API Google Map est intégrer dans l'application pour permettre au livreur d'utiliser le GPS lors des livraisons de commande.

Diagramme UML de Composants

Le diagramme de composant est un model static qui permet d'obtenir une vue d'ensemble de notre système du point de vue des éléments logiciels. Ainsi de bien comprendre le comportement du service fourni par chacun des composants de notre système. Ce diagramme ici va nous permettre de mettre en évidence les dépendances entre les composants (Account, , commande et inventaires).



3.1.1.Composants « Account »

Le composant « **Account** » permet à l'utilisateur de se créer un compte utilisateur ou de se connecter à l'application en utilisant l'interface d'authentification qui expose ce composant. Dans le composant on retrouve le composant « **users** » qui fait référence à l'utilisateur et le composant **authentification**. Le composant « **users** » dépend du composant « **Authentification** ».

3.1.2.Composants « Commande »

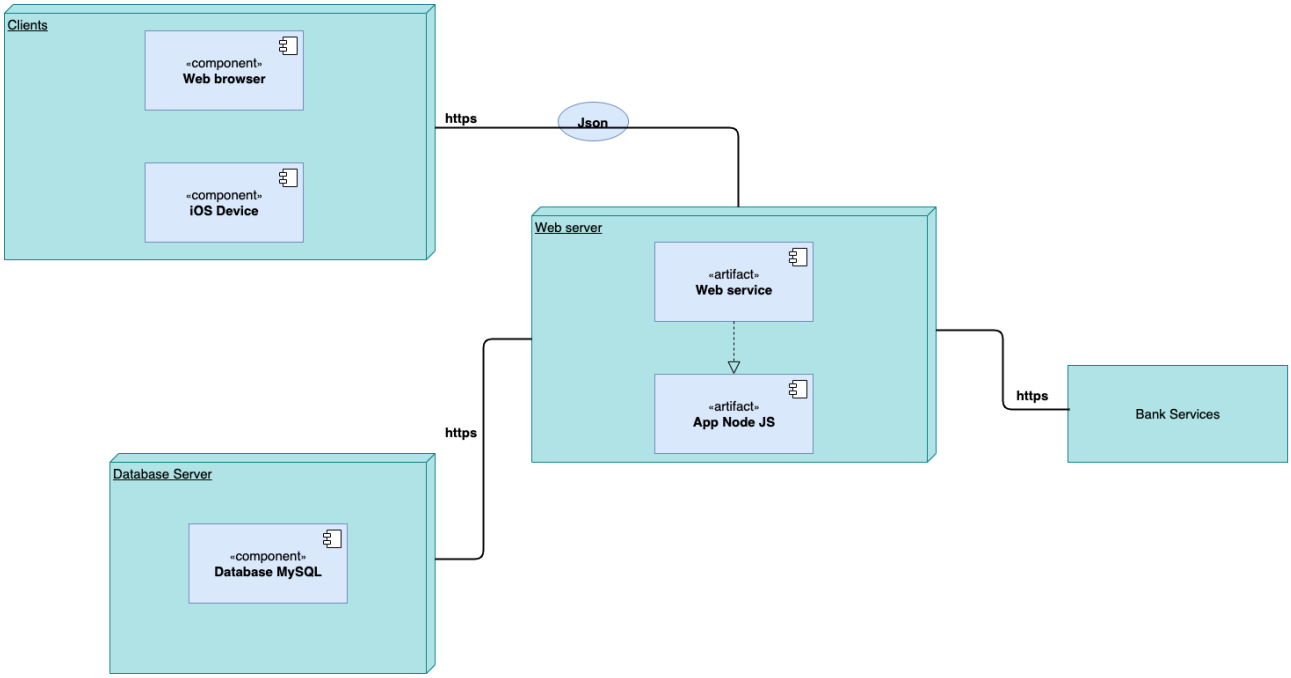
Le composant commande dépend du composant **account** pour exister. Car c'est grâce à son compte que l'utilisateur peut consulter ou effectuer une commande. Ce composant contient Quatre autres composant (**Pizza**, qui dépend d'un autre composant **'ingrédient'**, composant **paiement** qui dépend du composant **'adresse'** du composant **livraison** qui lui aussi dépend du composant **adresse** et enfin le composant **adresse** qui permet la livraison de la **commande**.

3.1.3.Composants « Inventaire »

Le composant **inventaire** permet la gestion de stock, pour cela il contient deux autres composant (**ingrédient** qui dépend du composant **stock** et le composant **stock**) le composant stock expose l'interface « **Gérer stock** ».

4. ARCHITECTURE DE DÉPLOIEMENT

Diagramme UML de déploiement



- Le Noeud container « **Client** » : corresponds à l'interface front-end coté client, il contient le composant **web browser** qui représente le navigateur web ou sera consulté le site internet et le composant « **iOS Devices** » qui représente le téléphone ou tablette sur laquelle sera consulté l'application.
- Les clients communiquent avec le web serveur par le biais du protocole HTTPS (HyperText Transfert Protocol Secure). Plus sécuriser et qui permet aux visiteurs de vérifier l'identité du site web auquel ils accèdent.
- Le Noeud container « Web Server » : représente le serveur qui héberge et qui stock les éléments de notre application. Il contient le composant web service qui correspond aux APIs qui seront utilisés dans l'application et le composant App Mode JS pour la mise en forme de notre back-end.
- Le noeud container « Database server » : Représente la base de donnée dans laquelle est stocké les données de notre application. Il contient le composant Database MySQL qui correspond au système de gestion de base de données relationnelle que nous utiliserons.
- Le Noeud « Bank service » : représente le service bancaire utilisé dans notre application pour accéder au paiement en ligne via le service de 'Stripe' ou 'PayPal'

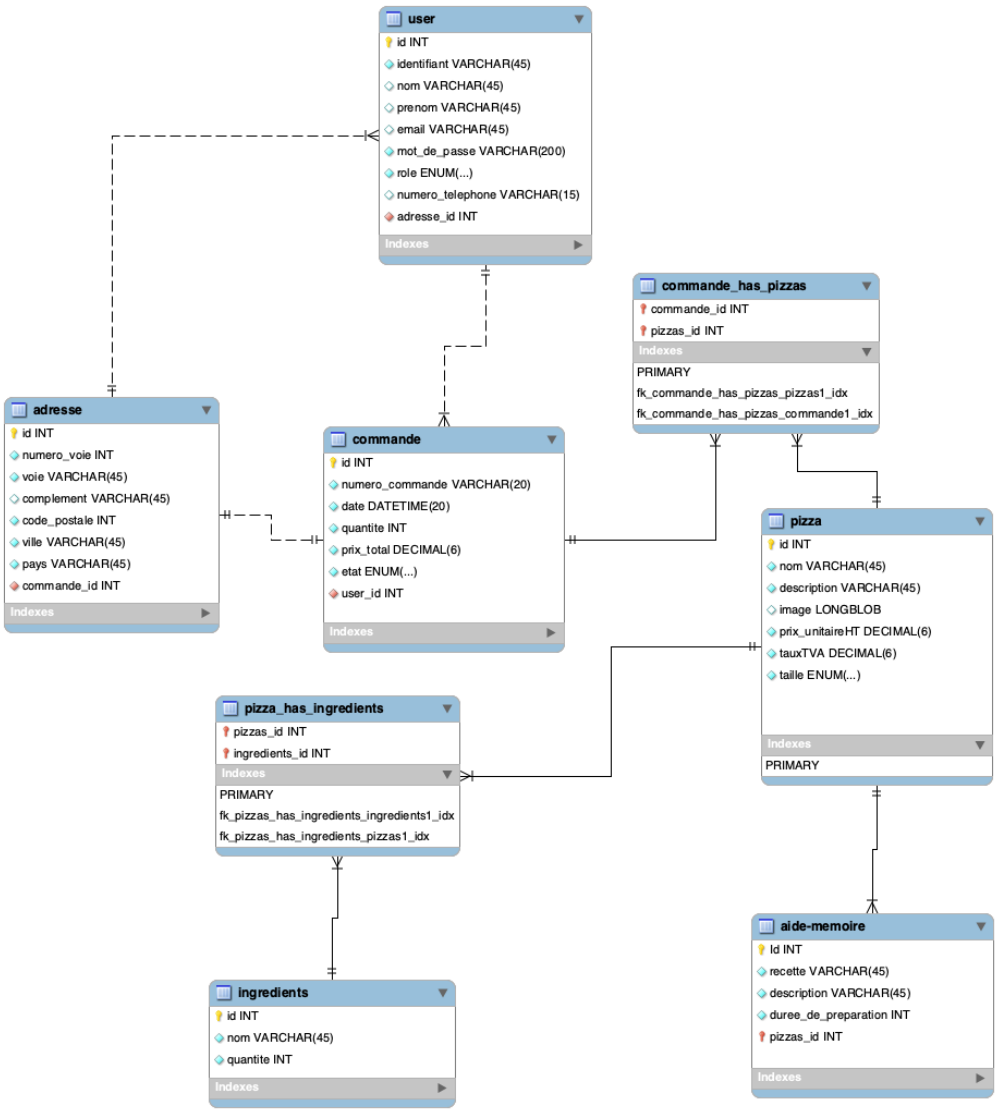
4.1.Serveur de Base de données

Pour le déploiement de notre serveur de base de données, on utilise MySQL qui est un serveur de base de données relationnelles SQL. C'est le plus populaire en hébergement web et facile d'intégrer avec d'autres services Microsoft.

On l'associe avec le SGBD: MySQL Workbench un utilitaire de conception et d'administration de base de données pour la conception et la modélisation des différents tables de notre système et la mise en relation de ces tables.

Les informations stockés et manipulés dans notre base de données sont en outre, les utilisateurs, les commandes, les pizzas, les ingrédients etc…

Ils sont présentés sous formes de tables dans lesquelles ont retrouve les attributs, les identifiants et les clés primaires.



Présentation des tables de la base de données

La table « user » :

La table englobe tous les utilisateurs du système, elle dispose d'une clé primaire « id » de type INT et Auto-incrémentable, cette clé « id » permet de différencier chacun des utilisateurs existants dans la base de données de notre système.

La table est composé d'attribut de type (VARCHAR) avec une limitation de caractère.

Elle est composée aussi d'attribut « Enum » pour la sélection du rôle de l'utilisateur.

Et un Foreign Key « adresse_id » s'y ajoute pour définir la relation entre les deux tables

La table « adresse » :

Dispose d'une clé primaire « id » de type INT et Auto-incrémentable, cette clé « id » permet de différencier les adresses existantes dans la base de données de notre système.

La table est composé d'attribut de type (VARCHAR) et (INTEGER) avec une limitation de caractère.

Un Foreign Key « commande_id » s'y ajoute pour définir la relation entre les deux tables

La tables « commande » :

Elle dispose d'une clé primaire « id » de type INT et Auto-incrémentable, cette clé « id » permet de différencier les commandes de chaque utilisateur existant dans la base de données de notre système.

La table est composé d'attributs de type (VARCHAR) pour le numéro de commande, (DATE) pour la date de la commande, (INT) pour la quantité, (DECIMAL) pour le prix de la commande, (ENUM) pour l'état de la commande.

Un Foreign Key « user_id » s'y ajoute pour définir la relation entre les deux tables

La table « pizza » :

Elle dispose d'une clé primaire « id » de type INT et Auto-incrémentable, cette clé « id » permet de différencier les pizzas existantes dans la base de données de notre système.

La table est composée d'attributs de type (VARCHAR) pour le nom et la description, (LONGBLOG) pour le format de l'image, (DECIMAL) pour le prix et (ENUM) pour la taille de la pizza.

Une table « commande_has_pizza », avec « commande_id » et « pizza_id » est rajouté pour faire la relation entre la table pizza et la table commande.

La table « ingredient » :

Elle dispose d'une clé primaire « id » de type INT et Auto-incrémentable, cette clé « id » permet de différencier les ingrédients existant dans la base de données de notre système.

La table est composée d'attribut de type (VARCHAR) pour le nom de l'ingrédient et (INT) pour la quantité de chaque ingrédient.

Une table pizza_has_Ingredient, avec « ingredient_id » et « pizza_id » est rajoutée pour faire la relation entre la table pizza et la table ingrédient.

La table « aidé-mémoire » :

Elle dispose d'une clé primaire « id » de type INT et Auto-incrémentable, cette clé « id » permet d'identifié l'aide mémoire.

La table est composée d'attribut de type (VARCHAR) pour la recette et la description et (INT) pour la durée de préparation.

Et un Foreign Key « pizza_id » s'y ajoute pour définir la relation entre les deux tables

4.2.Serveur web

Pour la mise en ligne de notre application web (site internet), nous utilisons le service **OVH Cloud** qui permet d'héberger dans une DataCenter notre système très facilement sans avoir à le stocké localement sur notre machine.

D'autant plus que **OVH Cloud** propose des offres assez riche en fonction des besoins.

Pour le transfère des fichiers du système vers le serveur cloud, on utilise le client FTP (Filezilla) qui est compatible avec **OVH Cloud**

5.ARCHITECTURE LOGICIELLE

5.1.Principes généraux

Les sources et versions du projet sont gérées par **Git**, ensuite le code est stocké à distance dans un repository privé sur **GitHub**. La structure du projet, elle, est implémentée en **HTML**, enjolivée par du **CSS** et du **Javascript** avec l'utilisation de la librairie **Bootstrap**. Le projet est développé dans l'éditeur de code; **VScode**, un IDE très complet et très populaire qui a l'avantage d'embarquer plusieurs extensions avec la possibilité de débuger en direct sur le logiciel et la mise en ligne est fait avec **Filezilla**.

5.1.1.Les couches

L'architecture applicative est la suivante :

- une couche **Model** : Responsable de la logique métier du composant
- une couche **View** : Responsable de l'affichage de l'interface utilisateur
- une couche **Contrôler** : Responsable de la communication entre le Model et la View qui eux ne communique pas directement.

5.1.2.Structure des sources

La structuration des répertoires du projet suit la logique suivante :

– les répertoires sources sont crées de façon à respecter le modèle MVC

```
racine
├── model
│       ├── classes
│       │     ├── user
│       │     ├── employee
│       │     ├── order
│       │     ├── commande
│       │     ├── pizza
│       │     └── ingredient
│       └── other
├── view
│     └── src
│         ├── main
│         └── test
├── controller
│     └── src
│         ├── main
│         └── test
├── unit tests
└── src
    └── assets
```

6.GLOSSAIRE

Bootstrap	Bootstrap est une collection d'outils utiles à la création du design de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option.
MySQL	MySQL est un système de gestion de bases de données relationnelles
OVH Cloud	OVH est notre service d'hébergement et serveur qu'on vas utiliser pour stocker et mettre en ligne notre application. OVH est une entreprise française. Elle pratique initialement de l'hébergement de serveur, et est un fournisseur d'accès à Internet, puis opérateur de télécommunications pour les entreprises
API	interface de programmation applicative est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.
UML	Le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language, est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet
Stripe	Stripe est une société américaine d'origine irlandaise, destinée au paiement par internet pour professionnels
Paypal	PayPal est une entreprise américaine offrant un système de service de paiement en ligne dans le monde entier. La plateforme sert d'alternative au paiement par chèque ou par carte bancaire
Filezilla	FileZilla est notre outil de migration que nous allons utiliser. FileZilla Client est un client FTP, FTPS et SFTP, développé sous la licence publique générale GNU. Il est intégré à la liste des logiciels libres préconisés par l'État français dans le cadre de la modernisation globale de ses systèmes d'informations