OC PIZZA

Application de gestion de pizzerias en ligne

Dossier de conception technique

Version 1.0

Auteur
Stéphanie Mehraik
Analyste-programmeur

TABLE DES MATIÈRES

1. VERSIONS

Auteur	Date	Description	Version
SME	09/07/20	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. Il est destiné aux développeurs, mainteneurs et à l'équipe d'OC Pizza.

L'objectif du document est de lister les contraintes spécifiques dont les intervenants vont devoir tenir compte pour développer et maintenir l'application.

Le présent document présentera les langages et les conventions de développement, l'architecture logicielle et de déploiement ainsi que les logs et monitoring propres à l'application.

Les éléments du présent dossier découlent :

- de notre entretien du 09 juillet 2020
- Du document de spécifications techniques réalisé par IT Consulting & Developement

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. **DE - 1.0** : Dossier d'exploitation de l'application
3. **PVL - 1.0** : Procès verbal de livraison finale

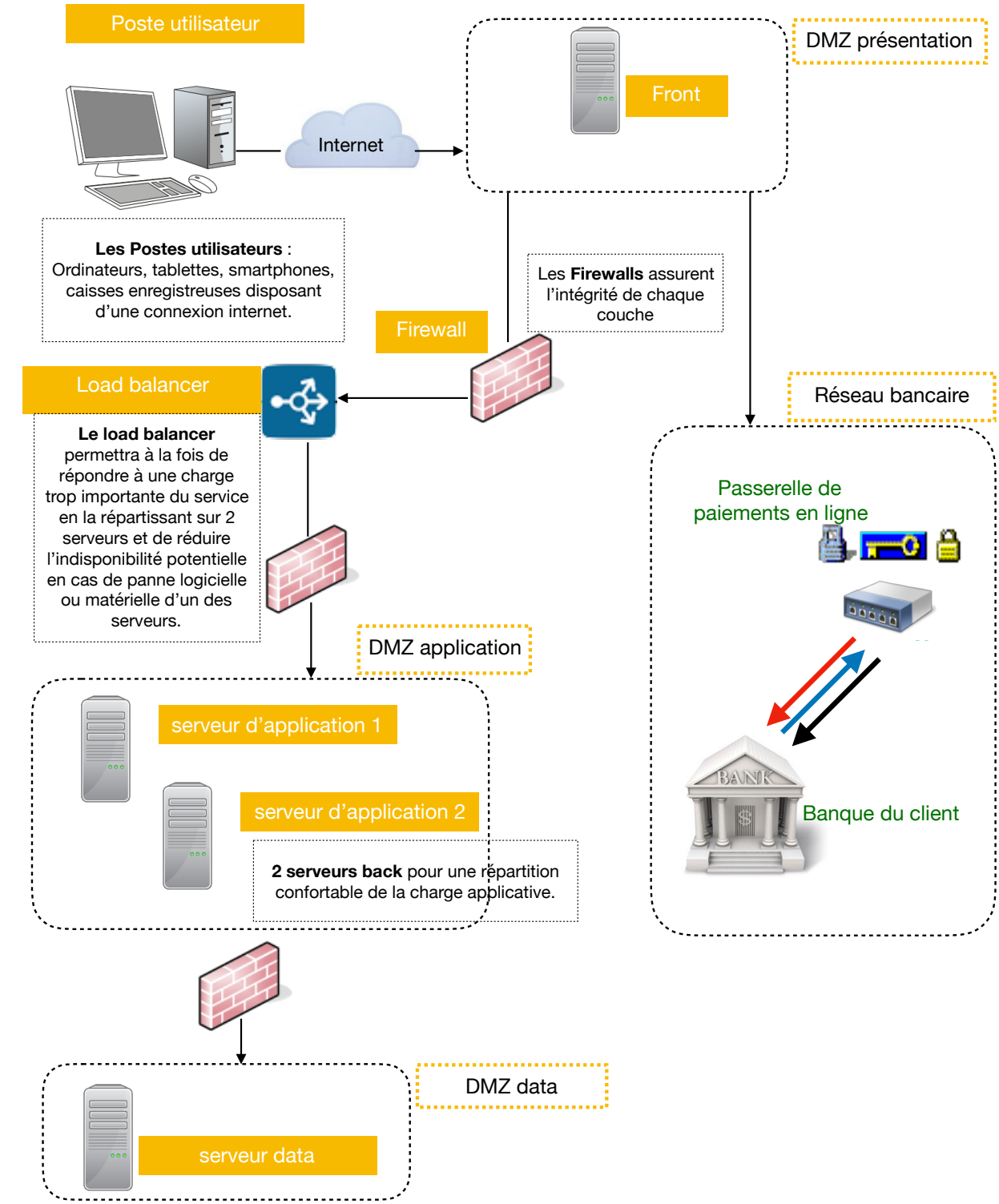
3.ARCHITECTURE TECHNIQUE

Afin de répondre au mieux aux besoins client, nous avons développé l'application en deux parties distinctes : l'application front-end et l'application back-end.

3.1.Application Web

La pile logicielle est la suivante :

- Les postes utilisateurs disposant d'une connexion Internet. Les navigateurs web classiques tels que Google Chrome 83.0.4103.97, Internet explorer 11.0, Safari 5.1.7, Firefox 12, etc., pourront être utilisés pour accéder à l'application.
- Application **J2EE** (JDK version 1.8)
- 1 Serveur d'application front-end, la partie visible du site-web. Nous proposons d'utiliser les langages de programmation standard : HTML, CSS et Javascript.
- 2 serveurs back-end pour la partie logique du site. Le langage proposé est Java, qui est très populaire en entreprise, jumelé à son Framework MVC Spring. Cette interface back-end communiquera avec la base de données MySQL.
- 1 load balancer qui répartira la charge entre les serveur en cas de panne ou d'activité importante.


Poste utilisateur
Internet
Front
DMZ présentation
Les Postes utilisateurs : Ordinateurs, tablettes, smartphones, caisses enregistreuses disposant d'une connexion internet.
Les Firewalls assurent l'intégrité de chaque couche
Firewall
Load balancer
Le load balancer permettra à la fois de répondre à une charge trop importante du service en la répartissant sur 2 serveurs et de réduire l'indisponibilité potentielle en cas de panne logicielle ou matérielle d'un des serveurs.
Réseau bancaire
Passerelle de paiements en ligne
DMZ application
serveur d'application 1
serveur d'application 2
2 serveurs back pour une répartition confortable de la charge applicative.
BANK
Banque du client
DMZ data
serveur data

3.1.1.Composants

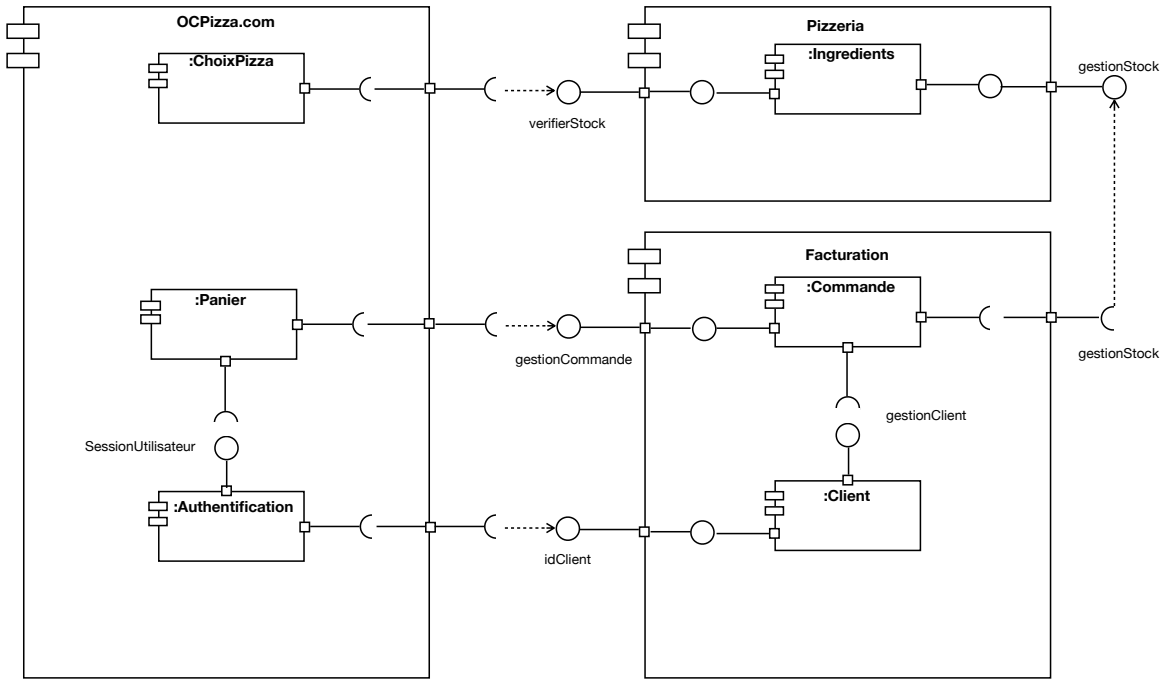


Diagramme UML de Composants

Nous avons identifié 6 composants pour l'application. Dans cette partie, nous décrirons les composants internes et externes utilisés par l'application

Le choix des pizzas:

Le catalogue va permettre au client de consulter la liste des pizzas proposées. C'est à partir de ce composant qu'il fera son choix parmi la gamme de produits et alimenter son panier.

Le panier :

Le panier est un dispositif du site marchand en ligne dans lequel se trouvent tous les articles sélectionnés par un client au cours de sa visite. Il calcule le total à payer. Une fois validé, le panier se transforme en commande.

L'authentification :

Le composant d'authentification permet l'inscription de nouveaux clients à l'application, et permet la connexion des clients existants et du personnel. Cette étape est obligatoire pour passer et gérer une commande.

Les commandes :

La gestion des commandes est nécessaire pour assurer le bon cheminement d'une commande client, de sa validation à la livraison.

Les clients :

La gestion des clients permet au personnel de suivre les instructions concernant la commande, par exemple pour trouver l'adresse de livraison si le client a choisi de se faire livrer.

Inventaire :

Le composant Ingrédients représente la gestion des stocks. Il permet au personnel et aux gérants de contrôler les matières premières reçues, consommées et à commander, sur les différents points de vente.

3.2. Base de données

Le base de données permet de stocker et de retrouver l'intégralité des données et informations en rapport avec la gestion de la chaine de restaurants OC PIZZA.

3.2.1.Système de gestion de base de données

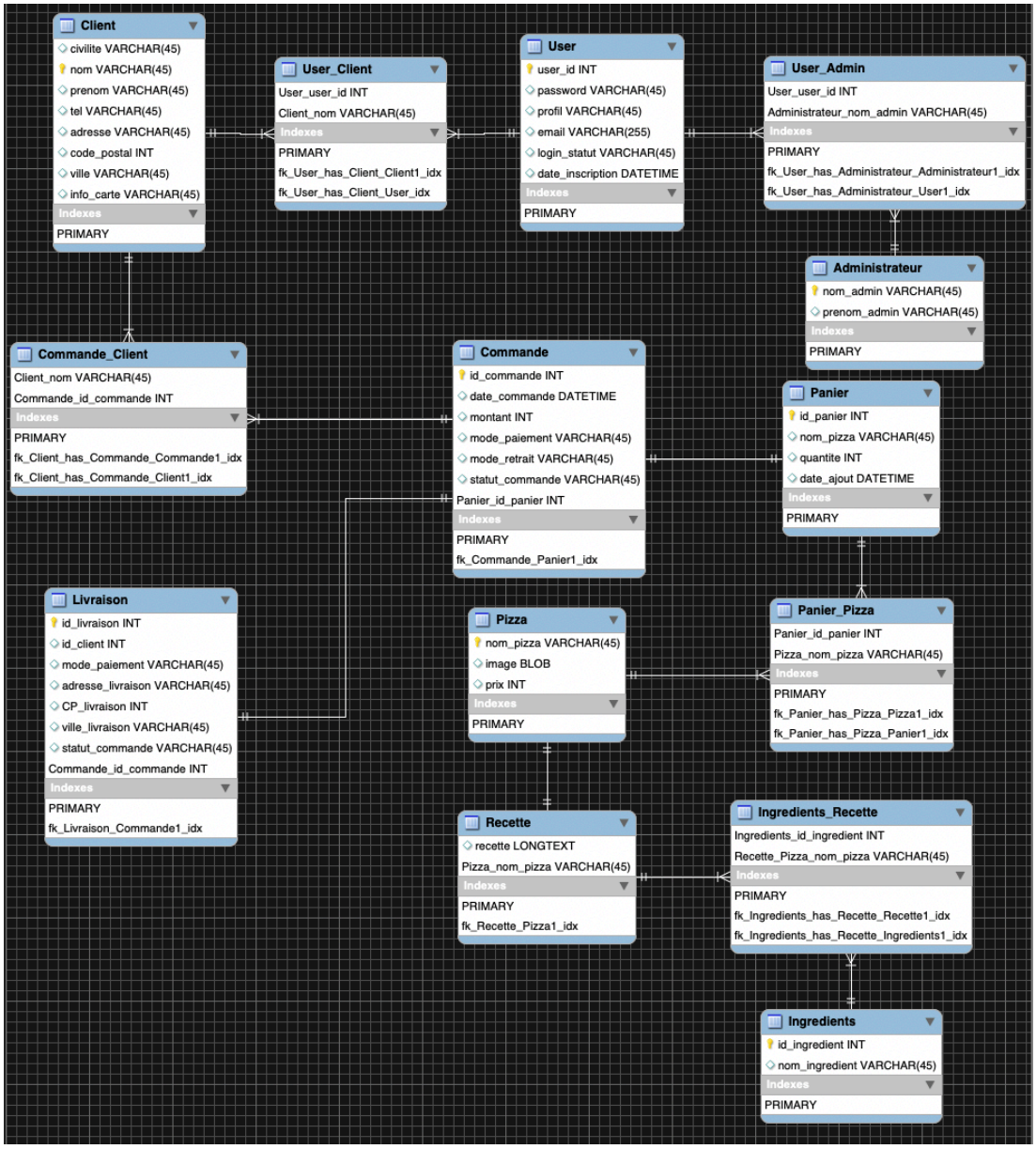
Un système de gestion de base de données est un logiciel qui permet le stockage d'information dans une base de données. On peut y créer, lire, mettre à jour et supprimer des données.

Pour l'application OC PIZZA, nous proposons de gérer la base de données par le SGDBR MySQL qui est open source, son utilisation est répandue, efficace et sécurisée. C'est un système très rapide et parfaitement adapté à la taille du projet.

3.2.2. Modèle physique de données

A partir du diagramme de classes présenté ans le dossier de conception fonctionnelle, nous pouvons élaborer le MPD.

Le MPD permet d'avoir une représentation visuelle de la structure d'une base de données et de mieux comprendre les relations entre les différentes tables. Les parties suivantes auront pour but de détailler les types utilisés pour chaque attribut des tables ainsi que les clés primaires et étrangères.



3.2.3.Présentation des tables

La table « User »

Attribut	Type	Commentaire
user_id	INT(11) NOT NULL, AUTO_INCREMENT	**Clé primaire -** identifiant de l'utilisateur
password	VARCHAR(45)	mot de passe
profil	VARCHAR(45)	niveau de permission
email	VARCHAR(255)	adresse e-mail
login_statut	VARCHAR(45)	état de connexion (connecté / déconnecté)
date_inscription	DATETIME	date d'inscription

La table « Client »

Attribut	Type	Commentaire
nom	VARCHAR(45) NOT NULL	**Clé primaire -** nom du client
civilite	VARCHAR(45)	civilité du client (Madame / Monsieur)
prenom	VARCHAR(45)	prénom du client
tel	VARCHAR(45)	numéro de téléphone
adresse	VARCHAR(45)	numéro et nom de voie
code_postal	INT(11)	code postal
ville	VARCHAR(45)	ville
info_carte	VARCHAR(45)	informations bancaires

La table de correspondance « User_Client »

Attribut	Type	Commentaire
User_user_id	INT(11)	**Clé étrangère -** identifiant de l'utilisateur
Client_nom	VARCHAR(45)	**Clé étrangère** - Nom du client

La table « Administrateur »

Attribut	Type	Commentaire
nom_admin	VARCHAR(45)	**Clé primaire -** nom de l'administrateur
prenom_admin	VARCHAR(45)	prénom de l'administrateur

La table de correspondance « User_Admin »

Attribut	Type	Commentaire
User_user_id	INT(11)	**Clé étrangère -** identifiant de l'utilisateur
Administrateur_nom_admin	VARCHAR(45)	**Clé étrangère** - Nom de l'administrateur

La table « Commande »

Attribut	Type	Commentaire
id_commande	INT(11) NOT NULL, AUTO_INCREMENT	**Clé primaire -** numéro de commande
date_commande	DATETIME	Date de la commande
montant	INT(11)	Montant total de la commande
mode_paiement	VARCHAR(45)	Mode de paiement choisi
mode_retrait	VARCHAR(45)	Sur place, à emporter ou à livrer
statut_commande	VARCHAR(45)	statut de la commande (validée / préparée / livrée)
Panier_id_panier	INT(11)	**Clé étrangère -** identifiant du panier

Table de correspondance « Commande_Client »

Attribut	Type	Commentaire
Commande_id_commande	INT(11)	**Clé étrangère -** numéro de commande
Client_nom	VARCHAR(45)	**Clé étrangère** - Nom du client

La table « Panier »

Attribut	Type	Commentaire
id_panier	INT(11) NOT NULL, AUTO_INCREMENT	**Clé primaire -** identifiant du panier
nom_pizza	VARCHAR(45)	nom de la pizza
quantité	INT(11)	quantité commandée
date_ajout	DATETIME	date d'ajout de la pizza dans le panier

La table « Pizza »

Attribut	Type	Commentaire
nom_pizza	VARCHAR(45) NOT NULL	**Clé primaire -** nom de la pizza
image	BLOB	photo de la pizza
prix	INT(11)	prix unitaire de la pizza
Operation	**Commentaire**	
listePizzaUpdate()	Pour mettre à jour la liste des pizzas	

La table de correspondance « Panier_Pizza »

Attribut	Type	Commentaire
Panier_id_panier	INT(11)	**Clé étrangère -** identifiant du panier
Pizza_nom_pizza	VARCHAR(45)	**Clé étrangère** - Nom de la pizza

La table « Ingredients »

Attribut	Type	Commentaire
id_ingredient	INT(11) NOT NULL, AUTO_INCREMENT	**Clé primaire -** identifiant de l'ingrédient
nom_ingredient	VARCHAR(45)	nom de l'ingrédient

La table « Recette »

Attribut	Type	Commentaire
Pizza_nom_pizza	VARCHAR(45)	**Clé étrangère -** nom de la pizza
recette	LONGTEXT	recette de la pizza

La table de correspondance « Ingredients_Recette »

Attribut	Type	Commentaire
Recette_Pizza_nom_pizza	VARCHAR(45)	**Clé étrangère -** nom de la pizza
Ingredient_id_ingredient	INT(11)	**Clé étrangère -** Identifiant de l'ingrédient

La table « Livraison »

Attribut	Type	Commentaire
id_livraison	INT(11) NOT NULL, AUTO_INCREMENT	**Clé primaire -** identifiant livraison
Commande_id_commande	INT(11)	**Clé étrangère -** numéro de commande
User_user_id	INT(11)	**Clé étrangère -** identifiant client
mode_paiement	VARCHAR(45)	mode de paiement (carte / cash)
adresse_livraison	VARCHAR(45)	Numéro et nom de voie
CP_livraison	INT(11)	Code postal
ville_livraison	VARCHAR(45)	Ville
statut_commande	VARCHAR(45)	statut de la commande (validée / préparée / livrée)

4.ARCHITECTURE DE DÉPLOIEMENT

Le diagramme ci-dessous décrit le déploiement physique environnemental du système ainsi que ces composants matériels

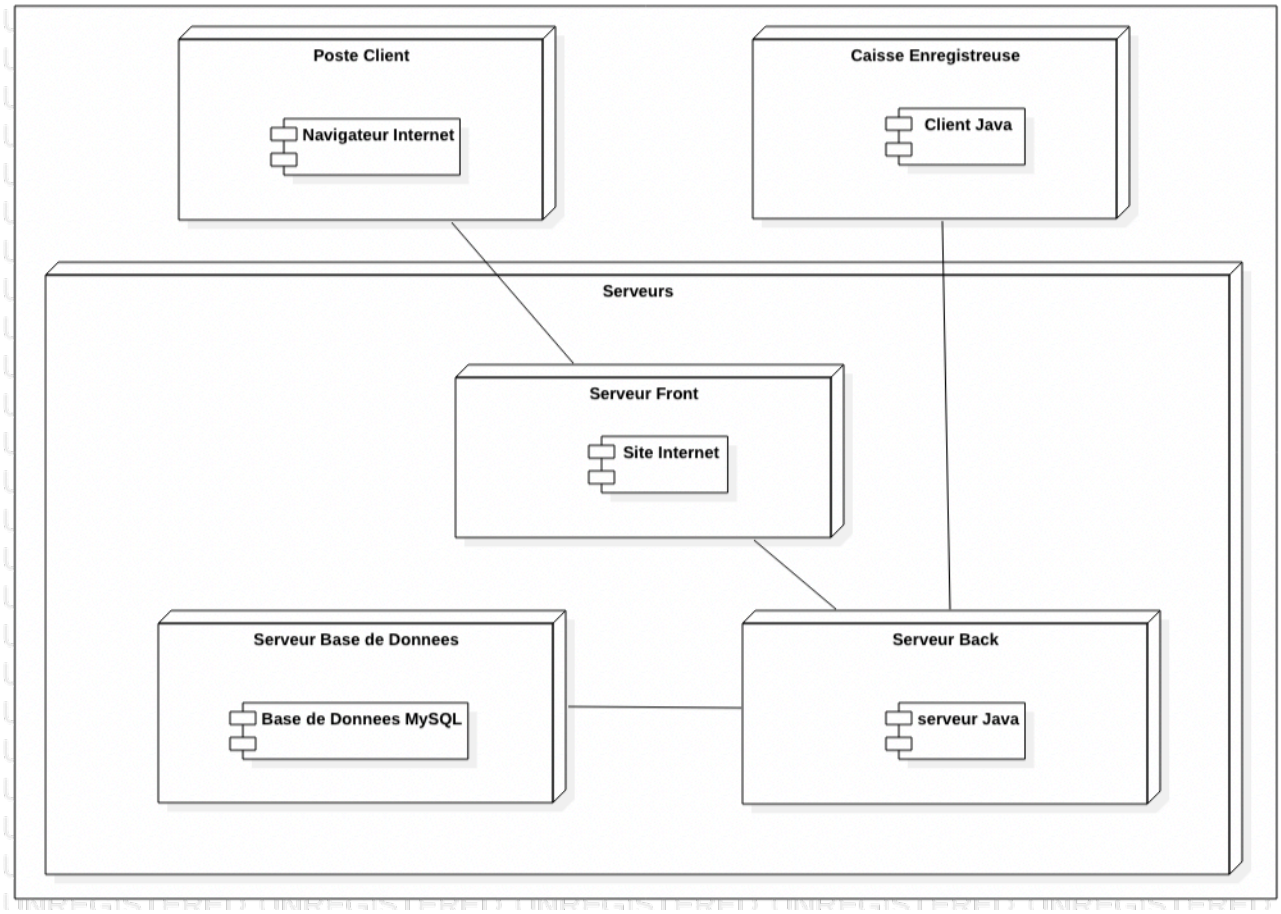


Diagramme UML de déploiement

Poste client :

Il s'agit d'ordinateurs, tablettes ou smartphones, équipés d'une connexion Internet, que les visiteurs et clients utilisent pour se connecter à l'application.

Caisse enregistreuse :

Ordinateur qui équipe les points de vente, sur lequel le personnel gère les commandes et le restaurant.

Serveurs :

Il s'agit du site web divisé en une partie visible (serveur front) et une partie logique (serveur back).

Des notifications peuvent être envoyées au client ou au personnel selon les divers événements liés au processus de commande.

Le serveur communiquera avec des services externes comme la base de données et le système bancaire.

Le serveur de base de données:

Un protocole HTTP permet au noeud serveur d'application de communiquer avec le noeud serveur de base de données qui est considéré comme un service externe. Ce serveur héberge la base de données MySLQL

Le serveur bancaire :

Un protocole API permet au noeud serveur d'application de communiquer avec le noeud serveur bancaire qui est considéré comme un service externe. Son rôle est de gérer les opérations de paiement.

4.1.Serveur de déploiement

L'application OC PIZZA sera déployée sur un serveur OVH, entreprise française implantée dans 19 pays à travers le monde proposant des temps de latence réduits et un réseau ultra-sécurisé.

L'offre d'hébergement Pro au prix de 4,99 euros TTC par mois nous semble une solution adaptée à la taille du projet. Elle comprend :

- 1 nom de domaine,
- 250 Go d'espace disque,
- 100 adresses e-mails,
- Trafic illimité.

5.ARCHITECTURE LOGICIELLE

5.1.Principes généraux

Le développement de l'application se présente sous la forme d'un projet Spring MVC, divisé en 3 parties :

- **Gestion des achats** : contient toutes les fonctionnalités concernant la prise de commande et la livraison.
- **Gestion administrative** : contient toutes les fonctionnalités réservées au Manager et au patron (ex: mise à jours du site et consultation des chiffres).
- **Authentification** : les 2 parties ci-dessus devront utiliser le package Authentification.

Spring utilise un architecture-pattern MCV à 3 tiers :

- **La Vue** représente la couche avec laquelle l'utilisateur interagit.
- **Le contrôleur** peut communiquer avec la vue mais aussi avec **le Modèle** qui représente les données que le système manipule et qui sont stockées dans la base de données.
- Chaque couche ne communique qu'avec les couches adjacentes. Ceci présente l'avantage de préserver les données des modifications des utilisateurs d'une part, mais aussi de modulariser le système en couches qui peuvent être développées chacune indépendamment des autres.

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

5.1.1.Les couches

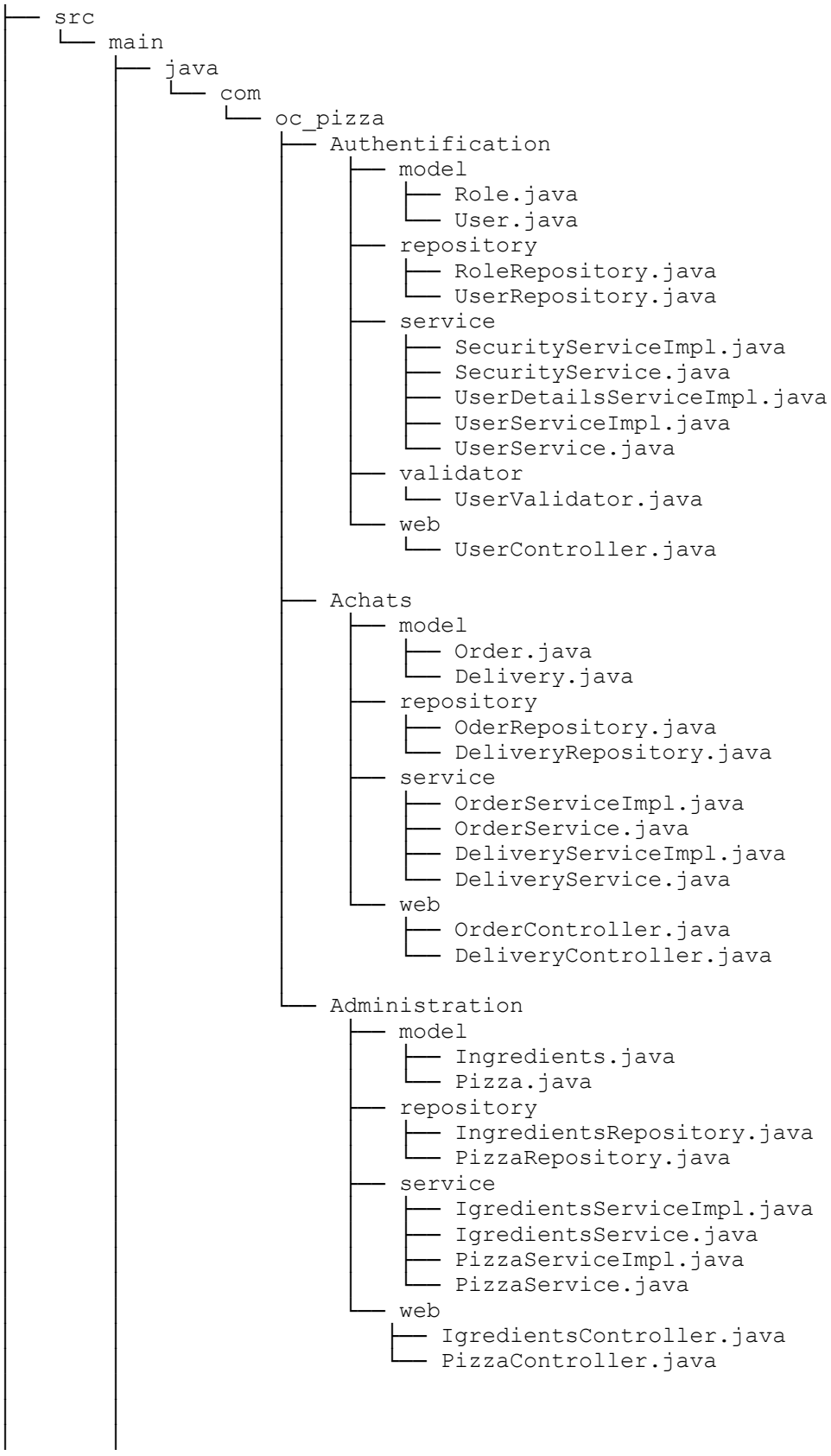
L'architecture applicative est la suivante :

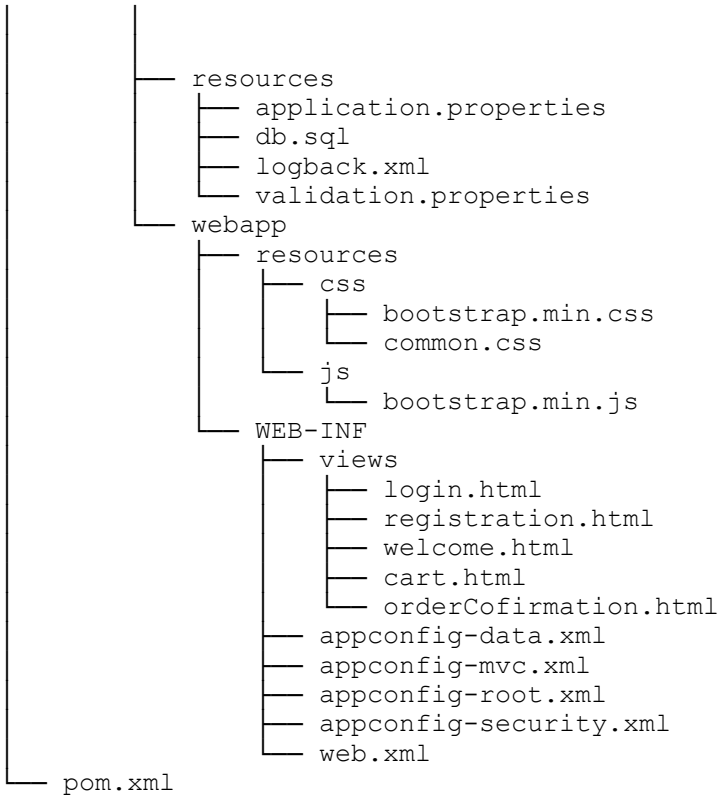
- une couche **business** : responsable de la logique métier du composant,
- une couche **model** : responsable de la représentation des données,
- Une couche **présentation** : responsable de l'interface utilisateur/application.

5.1.2.Structure des sources

La structuration des répertoires du projet suit la logique suivante :

- les répertoires sources sont créés de façon à respecter la philosophie Maven (à savoir : « convention plutôt que configuration »).


src
main
java
com
oc_pizza
Authentification
model
Role.java
User.java
repository
RoleRepository.java
UserRepository.java
service
SecurityServiceImpl.java
SecurityService.java
UserDetailsServiceImpl.java
UserServiceImpl.java
UserService.java
validator
UserValidator.java
web
UserController.java
Achats
model
Order.java
Delivery.java
repository
OderRepository.java
DeliveryRepository.java
service
OrderServiceImpl.java
OrderService.java
DeliveryServiceImpl.java
DeliveryService.java
web
OrderController.java
DeliveryController.java
Administration
model
Ingredients.java
Pizza.java
repository
IngredientsRepository.java
PizzaRepository.java
service
IgredientsServiceImpl.java
IgredientsService.java
PizzaServiceImpl.java
PizzaService.java
web
IgredientsController.java
PizzaController.java


resources
application.properties
db.sql
logback.xml
validation.properties
webapp
resources
css
bootstrap.min.css
common.css
js
bootstrap.min.js
WEB-INF
views
login.html
registration.html
welcome.html
cart.html
orderCofirmation.html
appconfig-data.xml
appconfig-mvc.xml
appconfig-root.xml
appconfig-security.xml
web.xml
pom.xml

6.POINTS PARTICULIERS

6.1.Gestion des logs et monitoring

Le Logs Data Plateform d'OVH offre une solution complète de gestion et d'analyse de logs pour l'entreprise du client. Cela offre une vue sur le comportement des serveurs physiques et virtuels, et permet de prévoir d'éventuels incidents ou dysfonctionnements.

Le service de monitoring d'OVH permet de suivre l'état de la machine et de déclencher automatiquement l'intervention d'un technicien dans le datacenter.

Tous les serveurs ainsi que l'ensemble du réseau sont surveillés 24h/24 et 7j/7 par les équipes techniques d'OVH.

OVH intervient dès le déclenchement d'une alerte (défaut de réponse au ping) afin de limiter au maximum le temps d'indisponibilité des serveurs et du réseau.

6.2.Ressources

Ces ressources désignent la charte graphique ainsi que le jeu de données.

6.2.1.Charte graphique

Le client devra nous fournir un cahier des charges graphique définissant l'identité visuelle de l'entreprise. Il comprendra les caractéristiques des différents éléments graphiques tels que les couleurs; polices, logos, symboles, etc.

6.2.2.Jeu de données

Le client devra nous fournir toutes les données à implémenter dans la base de données de l'application, à savoir : points de vente, pizzas proposées, etc.

6.3.Environnement de développement

Nous conseillons l'utilisation de l'IDE Spring Tool Suite, étendu dans Eclipse car il se spécialise dans les développement des applications Spring qui est un des frameworks les plus répandus pour les développements en Java.

6.4.Procédure de packaging / livraison

L'application sera déployée sur le serveur d'OVH au moment de la livraison finale.

Le Dossier d'exploitation - DE 1.0 - sera alors remis à l'équipe traitante. Ce document liste les informations dont l'équipe d'exploitation a besoin pour pouvoir assurer la continuité de l'utilisation en règle du système, et pouvoir réagir de manière appropriée lorsqu'un problème surgit.

7.GLOSSAIRE

IDE : Environnement de développement

SGBDR : Système de gestion de base de données relationnel

MPD : Modèle physique de données

MVC : Modèle Vue Contrôleur