OC Pizza

Mise en place d'un système de gestion de pizzeria

Dossier de conception technique

Version 1

Auteur
Kevin Bertrand
Développeur d'applications

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Kevin Bertrand	11/05/2022	Création du document	0.1
Kevin Bertrand	12/05/2022	Finalisation du document	1

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application **OC Pizza**. Ce document est destiné aux développeurs, aux mainteneurs et à l'équipe technique du client.

L'objectif de ce document est d'énumérer les contraintes à respecter lors de la réalisation des applications. De plus, ce document recense également les spécifications de développement tels que les langages de programmation, les conventions, l'architecture des applications ainsi que le déploiement.

Les éléments du présent dossier découlent :

- Du premier entretien avec les équipes de OC Pizza ;
- De l'analyse des besoins et de la conception technique préalablement effectuées.

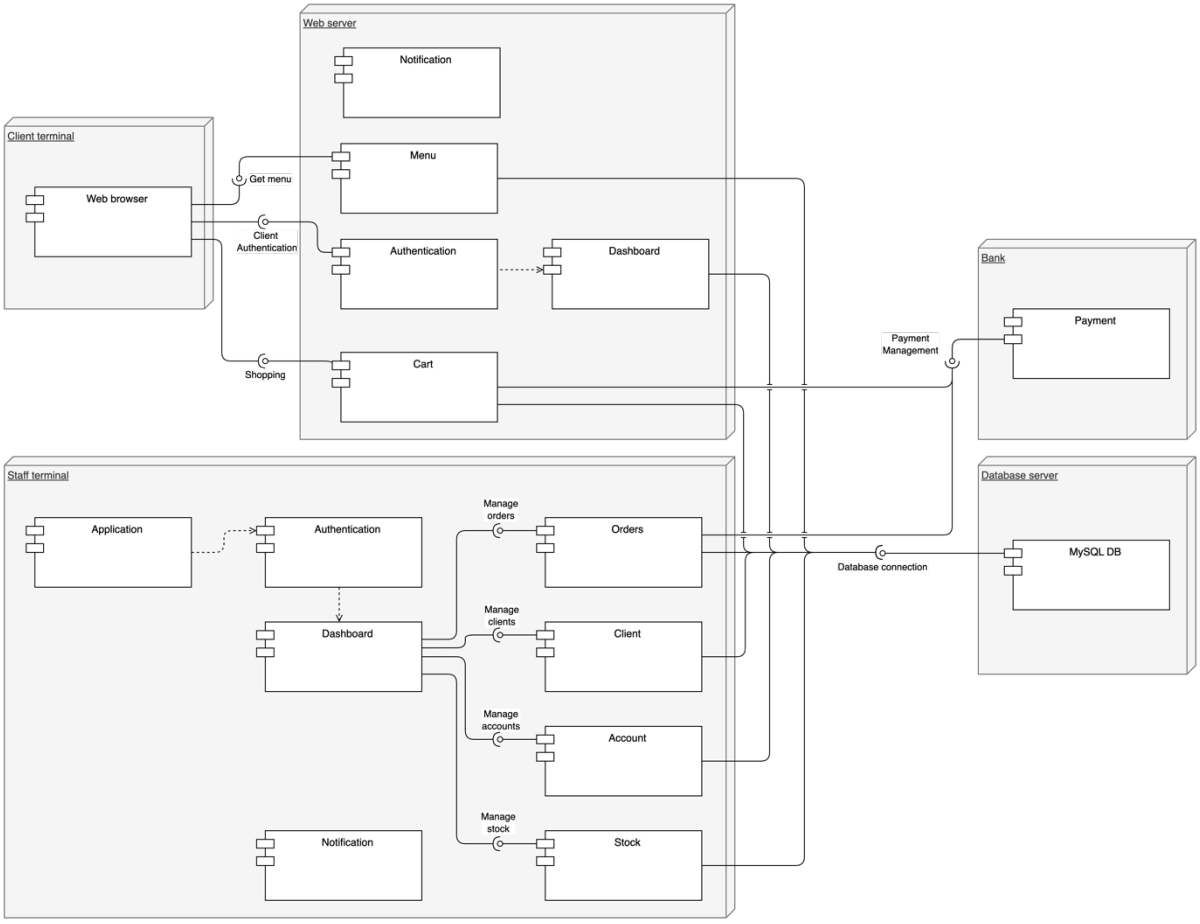
2.2 - Références

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

1. **Bertrand_Kevin_1_dossier_fonctionnelle_052022.pdf** : Dossier de conception fonctionnelle de l'application
2. **Bertrand_Kevin_3_dossier_d'exploitation_052022.pdf** : Dossier d'exploitation de l'application
3. **Bertrand_Kevin_4_PV_de_livraison_finale_052022.pdf** : Procès-verbal de livraison finale

3 - Architecture Technique

3.1 - Diagramme des composants



3.2 - Composants généraux

3.2.1 - Payement

Le composant payement est externe au système. Il permettra aux applications de gérer le payement des clients en vérifiant les données bancaires des clients et de sécuriser les transactions.

3.2.2 - Système de gestion de base de données

3.2.2.1 - Description

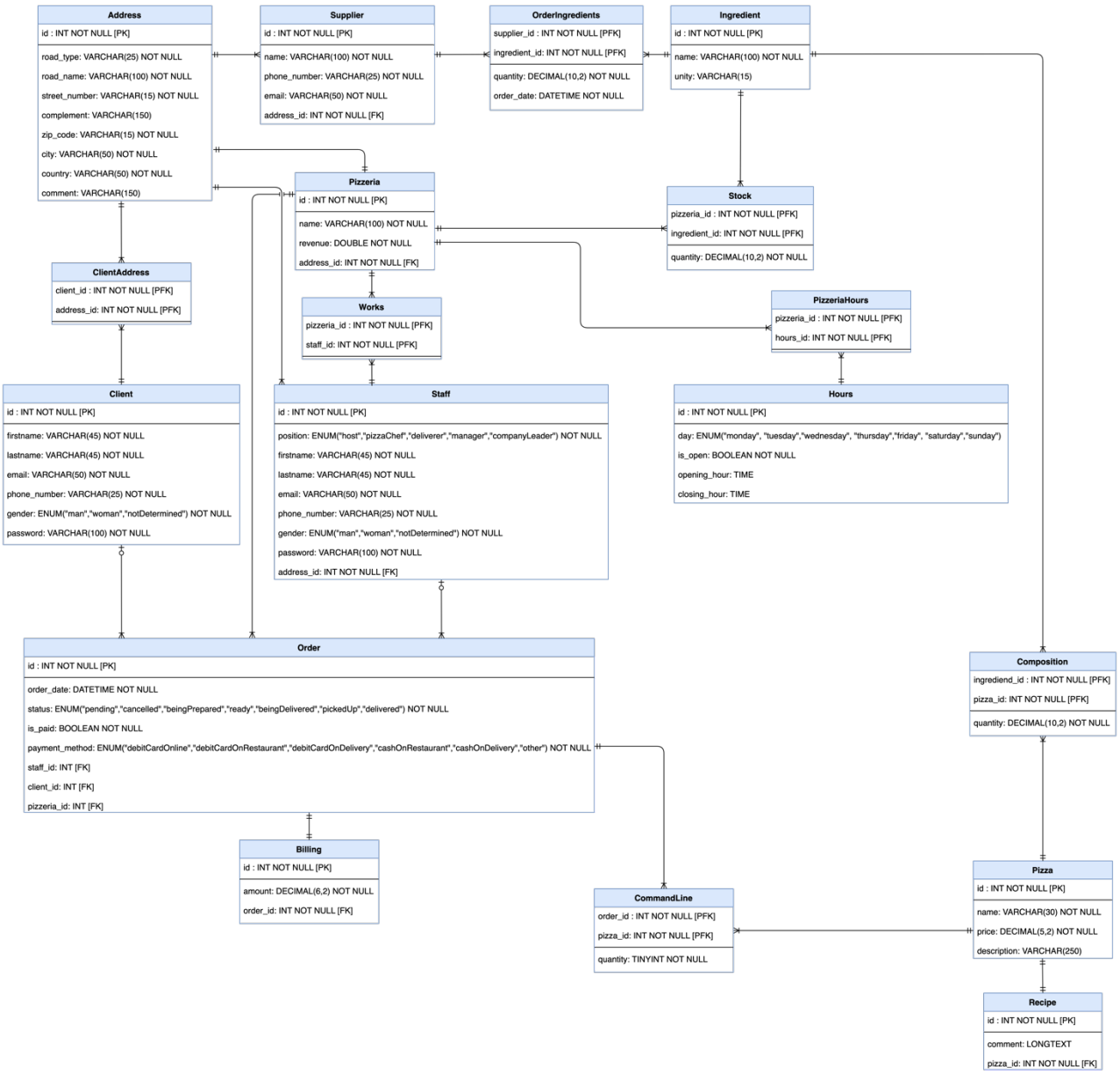
Le **S**ystème de **G**estion de **B**ase de **D**onnées (SGBD) est un logiciel de gestoin de base de données servant à stocker, à manipuler et à gérer des données dans une base de données.

3.2.2.2 - Choix du système

Le choix du SGBD s'est dirigé vers **MySQL** car ce SGBD est open source et est également un des plus populaire. MySQL possède également l'avantage d'avoir des données structures et, de ce fait, propose une rapidité de gestion des données rapide.

3.2.2.3 - Modèle physique de données

Le diagramme ci-dessous représente le **M**odèle **P**hysique de **D**onnées (MPD). Ce diagramme permet de visualiser graphiquement la structure et les relations de la base de données.



3.2.2.4 - Présentation des tables

3.2.2.4.1 Address

La table « **Address** » regroupe les informations relatives aux adresses des clients, du staff, des fournisseurs et des pizzérias.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
road_type	VARCHAR(25)	- Non-nul
road_name	VARCHAR(100)	- Non-nul
street_number	VARCHAR(15)	- Non-nul
complement	VARCHAR(150)	
zip_code	VARCHAR(15)	- Non-nul
city	VARCHAR(50)	- Non-nul
country	VARCHAR(50)	- Non-nul
comment	VARCHAR(150)	

3.2.2.4.2 Supplier

La table « **Supplier** » regroupe les informations relatives aux fournisseurs.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
name	VARCHAR(100)	- Non-nul
phone_number	VARCHAR(25)	- Non-nul
email	VARCHAR(50)	- Non-nul
address_id	INT	- Non-nul - Clé étrangère de la table **Address**

3.2.2.4.3 OrderIngredients

La table « **OrderIngredients** » est une table de correspondance entre les tables « **Ingredient** » et « **Supplier** »

Champ	Type	Caractéristiques
supplier_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Supplier**
ingredient_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Ingredient**
quantity	DECIMAL(10,2)	- Non-nul
order_date	DATETIME	- Non-nul

3.2.2.4.4 Ingredient

La table « **Ingredient** » regroupe tous les attributs des ingrédients utiles à la confection des pizzas.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
name	VARCHAR(100)	- Non-nul
unity	VARCHAR(15)	- Non-nul

3.2.2.4.5 Pizzeria

La table « **Pizzeria** » regroupe tous les attributs des pizzérias du groupe.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
name	VARCHAR(100)	- Non-nul
revenue	DOUBLE	- Non-nul
address_id	INT	- Clé étrangère de la table **Address**

3.2.2.4.6 Hours

La table « **Hours** » regroupe tous les attributs relatifs aux horaires d'ouverture des pizzérias.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
day	ENUM("monday", "tuesday","wednesday", "thursday","friday", "saturday","sunday")	- Non-nul
is_open	BOOLEAN	- Non-nul
opening_hour	TIME	- Non-nul
closing_hour	TIME	- Non-nul

3.2.2.4.7 PizzeriaHours

La table « **PizzeriaHours** » est une table de correspondance entre les tables « **Pizzeria** » et « **Hours** ».

Champ	Type	Caractéristiques
pizzeria_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Pizzeria**
hours_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Hours**

3.2.2.4.8 Stock

La table « **Stock** » est une table de correspondance entre les tables « **Pizzeria** » et « **Ingredient** ».

Champ	Type	Caractéristiques
pizzeria_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Pizzeria**
ingredient_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Ingredient**
quantity	DECIMAL(10,2)	- Non-nul

3.2.2.4.9 Staff

La table « **Staff** » regroupe tous les attributs relatifs aux employés du groupe de pizzéria.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
position	ENUM("host", "pizzaChef", "deliverer","manager", "companyLeader")	- Non-nul
firstname	VARCHAR(45)	- Non-nul
lastname	VARCHAR(45)	- Non-nul
email	VARCHAR(50)	- Non-nul - Unique
phone_number	VARCHAR(25)	- Non-nul - Unique
password	VARCHAR(45)	- Non-nul
gender	ENUM("man","woman", "notDetermined")	- Non-nul
address_id	INT	- Non-nul - Clé étrangère de la table **Address**

3.2.2.4.10 Works

La table « **Works** » est une table de correspondance entre les tables « **Pizzeria** » et « **Staff** ».

Champ	Type	Caractéristiques
staff_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Staff**
pizzeria_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Pizzeria**

3.2.2.4.11 Client

La table « **Client** » regroupe tous les attributs relatifs aux clients des pizzérias.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
firstname	VARCHAR(45)	- Non-nul
lastname	VARCHAR(45)	- Non-nul
email	VARCHAR(50)	- Non-nul - Unique
phone_number	VARCHAR(25)	- Non-nul - Unique
password	VARCHAR(45)	- Non-nul
gender	ENUM("man","woman", "notDetermined")	- Non-nul

3.2.2.4.12 ClientAddress

La table « **ClientAddress** » est une table de correspondance entre les tables « **Client** » et « **Address** ».

Champ	Type	Caractéristiques
client_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Client**
address_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Address**

3.2.2.4.13 Order

La table « **Order** » regroupe tous les attributs relatifs aux commandes passées par les clients.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
order_date	DATETIME	- Non-nul
status	ENUM("pending", "cancelled", "beingPrepared", "ready", "beingDelivered", "pickedUp", "delivered")	- Non-nul
is_paid	BOOLEAN	- Non-nul
payment_method	ENUM("debitCardOnline", "debitCardOnRestaurant", "debitCardOnDelivery", "cashOnRestaurant", "cashOnDelivery", "other")	- Non-nul
staff_id	INT	- Clé étrangère de la table **Staff**
client_id	INT	- Clé étrangère de la table **Client**
pizzeria_id	INT	- Clé étrangère de la table **Pizzeria**

3.2.2.4.14 Billing

La table « **Billing** » regroupe tous les attributs relatifs aux factures des commandes.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
amount	DECIMAL(6,2)	- Non-nul
order_id	INT	- Non-nul - Clé étrangère de la table **Order**

3.2.2.4.15 Pizza

La table « **Pizza** » regroupe tous les attributs relatifs aux pizzas.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
name	VARCHAR(30)	- Non-nul
price	DECIMAL(5,2)	- Non-nul
description	VARCHAR(250)	

3.2.2.4.16 CommandLine

La table « **CommandLine** » est une table de correspondance entre les tables « **Order** » et « **Pizza** ».

Champ	Type	Caractéristiques
order_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Order**
pizza_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Pizza**
quantity	TINYINT	- Non-nul

3.2.2.4.17 Recipe

La table « **Recipe** » regroupe tous les attributs relatifs aux recettes des pizzas.

Champ	Type	Caractéristiques
id	INT	- Clé primaire - Non-nul - Unique - Auto-incrémental
comment	LONGTEXT	- Non-nul
pizza_id	INT	- Non-nul - Clé étrangère de la table **PIZZA**

3.2.2.4.18 Composition

La table « **Composition** » est une table de correspondance entre les tables « **Pizza** » et « **Ingredient** ».

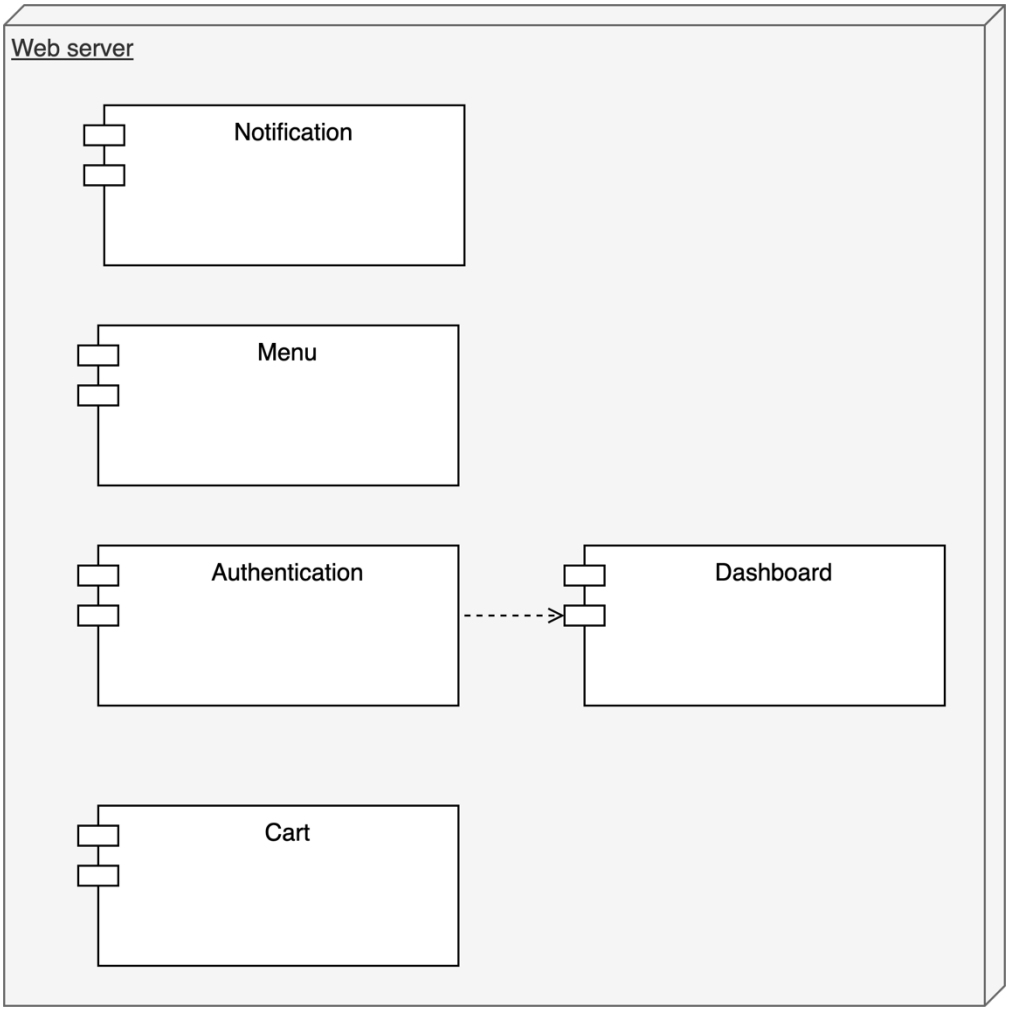
Champ	Type	Caractéristiques
pizza_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Pizza**
ingredient_id	INT	- Clé primaire - Non-nul - Clé étrangère de la table **Ingredient**
quantity	DECIMAL(10,2)	- Non-nul

3.3 - Application Web

L'application Web est séparée en 2 parties :

- Le **Front-end** : Partie visible par l'utilisateur accessible via un navigateur web.
  Cette partie sera développé en HTML, CSS et Javascript car ce sont des standards de la programmation Front-end.

- Le **Back-end** : Partie invisible gérant la logique. Cette partie sera développée en Python avec le Framework Django. Le choix de ce langage et de ce Framework s'est basé sur sa popularité, sa flexibilité et sa maintenabilité. C'est cette partie qui communiquera avec la base de données et les autres composants externes au système.



3.3.1 - Composant « Notification »

Ce composant permet au système d'envoyer des notifications aux client pour les informer sur l'avancement de leurs commandes.

3.3.2 - Composant « Menu »

Ce composant permet au système d'afficher au client la liste des personnes encore disponible dans la pizzéria sélectionnée par le client.

3.3.3 - Composant « Authentication »

Ce composant permet au client de s'inscrire ou de se connecter sur l'application Web.

3.3.4 - Composant « Dashboard »

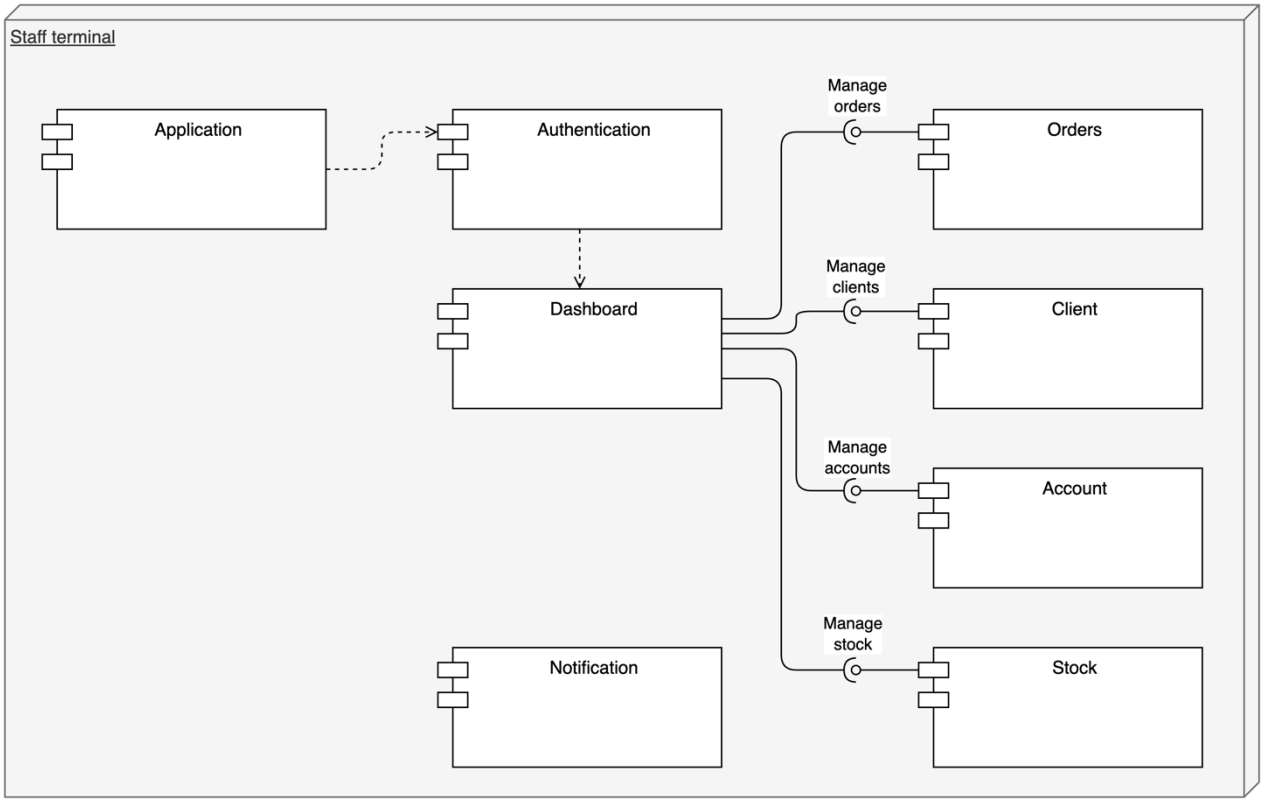
Ce composant permet au client de visualiser l'historique de ses commandes, visualiser le statut d'une commande et modifier ses informations personnelles.

3.3.5 - Composant « Cart »

Ce composant permet d'afficher le panier du client. Le montant de celui-ci sera mis à jour automatiquement dès que le client ajoutera/supprimera une pizza ou modifiera le nombre de pizza.

3.4 - Application OC Pizza

L'application OC Pizza sera disponible sur les plateformes iOS et iPadOS de chez Apple. Elle sera développée en Swift avec UIKit. Nous avons fait le choix de ces plateformes en raison de leur stabilité, leur sécurité mais aussi en raison de la flexibilité et de l'exigence demandée pour réaliser les applications.



3.4.1 - Composant « Application »

Ce composant reprend le code général de l'application avec son design.

3.4.2 - Composant « Authentication »

Ce composant permet aux différents utilisateurs (personnels du groupe) de se connecter à leur compte respectif.

3.4.3 - Composant « Dashboard »

Ce composant permet d'afficher les actions autorisées aux différentes personnes en fonction de leur rôle dans le groupe.

3.4.4 - Composant « Notification »

Ce composant permet d'envoyer des notifications au personnel pour indiquer qu'une nouvelle commande vient d'être passée (pour les pizzaiolos) ou encore qu'une commande est en attente de livraison (pour les livreurs).

3.4.5 - Composant « Orders »

Ce composant permet de visualiser et gérer les commandes de la pizzéria.

3.4.6 - Composant « Client »

Ce composant permet d'obtenir les informations du client relatives à la livraison de ses commandes.

3.4.7 - Composant « Account »

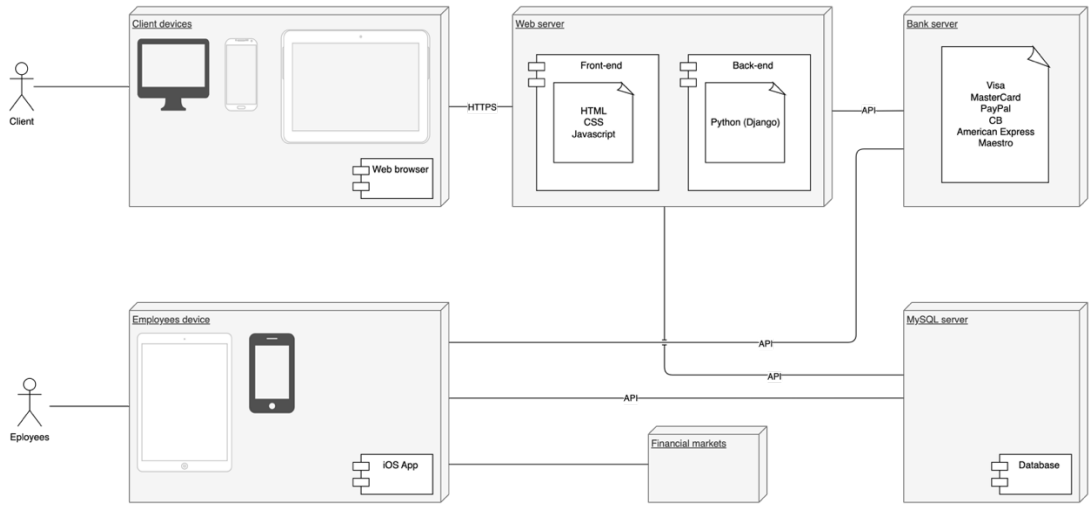
Ce composant permet de gérer les comptes des employés du groupe.

3.4.8 - Composant « Stock »

Ce composant permet de visualiser et gérer le stock des pizzerias.

4 - Architecture de Déploiement

Le diagramme ci-dessous représente l'environnement physique du système ainsi que les composants matériels.



4.1 - Marchés financiers

Les marchés financiers sont externes au système et permettront de consulter le prix des matières premières et de les négocier. Ces services utiliseront un système d'API pour communiquer avec l'application OC Pizza.

4.2 - Serveur bancaire

Le serveur bancaire est externe au système et permettra de gérer et sécuriser les payements. Un protocole API sera mis en place afin de faire communiquer les applications (web server et application OC Pizza) et ce serveur.

4.3 - Serveur de Base de données

Le serveur de base de données est externe au système et permettra d'héberger les données des applications. Ce serveur embarquera la base de données et le système d'API qui permettra d'établir la communication avec les applications (Web serveur et application OC Pizza).

La base de données sera de type MySQL 8.0.29 et sera hébergé sur un serveur Ubuntu Server 22.04 LTS. Le système d'API sera programmé en Swift avec le Framework Vapor.

4.4 - Serveur Web

Le serveur Web permettra de :

- Stocker et exécuter le code du Back-End qui représente la partie logique du site web et ne sera pas visible par les utilisateurs.
- Transmettre le code du Front-End aux utilisateurs.

La partie Front-End sera programmée en HTML 5, CSS 3 et Javascript ES6 tandis que la partie Back-End sera programmée en Python 3.10 avec le Framework Django 4.0.4. Tous ces scripts seront hébergés sur un serveur Ubuntu Server 22.04 LTS.

4.5 - Matériel Client

Le matériel client sera soit un ordinateur (de bureau ou portable), un smartphone ou une tablette et devra au minimum respecter les spécifications relatives aux navigateurs web suivantes :

- **Chrome 88.0.4324**
- **Safari MacOs 15.4**
- **Safari iOS 12.1.2**
- **Microsoft Edge 88.0.705.50**
- **Mozilla Firefox 91**
- **Opera 74**

4.6 - Matériel équipe OC Pizza

Le matériel de l'équipe d'OC Pizza sera des iPhones et iPad qui devront au minimum embarquer la version d'iOS 12.5 minimum. Ceux-ci embarqueront l'application développée en Swift avec UIKIT.

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git**.

L'application OC Pizza sera développée suivant le pattern MVC, c'est-à-dire **M**odèle-**V**ue-**C**ontrôleur.

L'application Web sera développée quant-à-elle suivant le pattern MVT utilisé par Django, c'est-à-dire **M**odèle-**V**ue-**T**emplate.

- **Modèle :** Le modèle est la partie de l'application qui exécute toute la partie logique de l'application comme les appels réseaux ou encore les opérations arithmétiques.
- **Vue :** C'est la partie graphique de l'application, elle n'a pour but que d'afficher des éléments à l'écran.
- **Controller :** Cette partie permet de relier le modèle à la vue.
- **Template :** Correspond à un fichier HTML recevant des objets Python. Le Template est toujours lié à la vue.

5.1.1 - Les couches

L'architecture applicative est la suivante :

- Une couche **business** : responsable de la logique métier du composant
- Une couche **model** : implémentation du modèle des objets métiers
- Une couche **présentation** : responsable de l'interface graphique de l'application.

5.1.2 - Structure des sources

5.1.2.1 - Application OC Pizza

La structuration des répertoires de l'application OC Pizza suit la logique suivante :

```
oc_pizza_ios
├─ README.md
├─ requirements.txt
├─ docs/
├─ OcPizzaTests/
├─ OcPizzaUITests /
├─ OcPizza.xcodeproj /
└─ OcPizza/
    ├─ SupportingFiles/
    │   ├─ Assets.xcassets/
    │   │   ├─ AppIcon.appiconset/
    │   │   ├─ AccentColor.colorset/
    │   │   └─ Contents.json
    │   ├─ LaunchScreen.storyboard
    │   ├─ Info.plist
    │   ├─ AppDelegate.swift
    │   └─ SceneDelegate.swift
    ├─ Model/
    │   ├─ NetworkManager.swift
    │   ├─ AuthenticationManager.swift
    │   ├─ OrderManager.swift
    │   ├─ AccountManager.swift
    │   ├─ ClientManager.swift
    │   └─ StockManager.swift
    ├─ View/
    │   └─ Main.storyboard
    └─ Controller/
        ├─ AuthenticationController.swift
        ├─ OrderController.swift
        ├─ AccountController.swift
        ├─ ClientController.swift
        └─ StockController.swift
```

5.1.2.2 - Application Web

La structuration des répertoires de l'application web suit la logique suivante :

```
oc_pizza_web
├─ configure.sh
├─ README.md
├─ requirements.txt
├─ docs/
├─ static/
│   ├─ script.js
│   ├─ style.css
│   └─ assets/
│       ├─ img/
│       └─ fonts/
├─ templates/
│   └─ base.html
├─ application/
    ├─ app.py
    ├─ models.py
    ├─ __init__.py
    ├─ view.py
    └─ tests.py
```

5.1.2.3 - Web service de la base de données

La structuration des répertoires du web service de la base de données suit la logique suivante :

```
oc_pizza_db
├─ configure.sh
├─ README.md
├─ requirements.txt
├─ docs/
├─ Package.resolved
├─ Package.swift
├─ Sources/
│   ├─ App/
│   │   ├─ configure.swift
│   │   ├─ routes.swift
│   │   ├─ Controllers/
│   │   ├─ Migrations/
│   │   └─ Models/
│   └─ Run/
│       └─ main.swift
├─ Tests/
    └─ AppTests/
```

6 - POINTS PARTICULIERS

6.1 - Gestion des logs

Les logs des applications seront stockés sur leurs serveurs respectifs dans le dossier « /var/logs/oc-pizza/ ». Ces logs seront enregistrés avec les standards qui les composent, c'est-à-dire que chacune des actions inscrites dans les logs seront horodatés et ces logs seront enregistrés dans des fichier .log.

6.2 - Fichiers de configuration

6.2.1 - Application web

Afin de faciliter le déploiement et la configuration de l'application Web, un fichier de configuration a été créer.

6.2.1.1 - Fichier configure.sh

Cet unique fichier de configuration permet d'installer les composants nécessaires au fonctionnement de l'application tels que Nginx. Il configurera également tous les composants de telle sorte que le déploiement soit le plus automatique possible. Pour ce faire, plusieurs questions seront posées afin de rentrer des informations confidentielles qui ne peuvent être mises en clair sur Github.
Le script installera les librairies suivantes :

- Nginx et ses dépendances.

Le script configurera les composants suivants :

- Nginx avec l'URL du site web ;
- Cron pour archiver automatiquement à minuit les logs de la veille ;
- Les variables d'environnement pour le fonctionnement de l'app (url de la base de données et son port de communication).

Une fois la fin de la configuration, un message s'affiche pour indiquer que la configuration s'est correctement achevée.

6.2.2 - Base de données

Afin de faciliter le déploiement et la configuration de la base de données et son web service, un fichier de configuration a été créer.

6.2.2.1 - Fichier configure.sh

Cet unique fichier de configuration permet d'installer les composants nécessaires au fonctionnement de la base de données et de son web service tels que Nginx et MySQL. Il configurera également tous les composants de telle sorte que le déploiement soit le plus automatique possible. Pour ce faire, plusieurs questions seront posées afin de rentrer des informations confidentielles qui ne peuvent être mises en clair sur Github.
Le script installera les librairies suivantes :

- Nginx et ses dépendances ;
- MySQL et ses dépendances.

Le script configurera les composants suivants :

- Nginx avec l'URL et le port de communication du web service ;
- Cron pour archiver automatiquement à minuit les logs de la veille ;
- Les variables d'environnement pour le fonctionnement de la base de données (nom d'utilisateur, mot de passe et port de communication).

Une fois la fin de la configuration, un message s'affiche pour indiquer que la configuration s'est correctement achevée.

6.2.3 - Application OC Pizza

L'application OC Pizza étant une application iOS/iPadOS et étant disponible sur l'AppStore, il n'existe aucun fichier de configuration.

6.3 - Ressources

Vous trouverez aux liens suivants, la documentation officielle de toutes les technologies nécessaires pour la réalisation du projet :

- Swift : https://www.swift.org
- UIKit : https://developer.apple.com/documentation/uikit
- Vapor : https://vapor.codes
- Python : https://www.python.org
- Django : https://www.djangoproject.com
- Ubuntu : https://ubuntu.com
- Apache : https://httpd.apache.org
- Git : https://git-scm.com

6.4 - Environnement de développement

6.4.1 - Base de données

La base de données sera développée sur XCode et sera dans un premier temps testée en local sur le Mac de développement. Ensuite, la base de données sera transférée sur un serveur de développement interne à l'entreprise pour effectuer les tests avec les applications.

6.4.2 - Application Web

Le développement de l'application Web recommande l'utilisation d'un environnement de développement (IDE) tel que Visual Studio Code.

6.4.3 - Application OC Pizza

Le développement de l'application OC Pizza nécessite l'utilisation de XCode et donc d'au minimum un Mac.

6.5 - Procédure de packaging / livraison

Tous les livrables nécessaire au développement du projet seront stockés sur GitHub.

L'application OC Pizza sera mise en ligne sur l'AppStore d'Apple accessible uniquement via un lien de téléchargement afin de restreindre l'accès à l'application.

Les serveurs Web et de base de données seront déployés sur des serveurs de chez Scaleway lors de la mise en production du système.

7 - GLOSSAIRE

DATETIME	Date et heure
DECIMAL(n,m)	Nombre décimal de n chiffre avec m décimales
DOUBLE	Nombre réel
INT	Nombre entier
LTS	**L**ong **T**ime **S**upport
MPD	**M**odèle **P**hysique de **D**onnées
MVC	**M**odel **V**iew **C**ontroller
MVT	**M**odel **V**iew **T**emplate
SGBDR	**S**ystème de **G**estion de **B**ases de **D**onnées
VARCHAR(n)	Chaine de caractère de taille variable avec un maximum de n caractères