OC Pizza

Création du nouveau système informatique

Dossier de conception technique

Version 1.0

Auteur
Sylla Mohamadou
Developpeur d'application ioS

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Xxx	JJ/MM/AAAA	Création du document	XXX

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application de gestion d'OC Pizza.
L'objectif du document est d'informer et aider les développeurs pour la conception, la maintenance et l'évolution de l'application.
Les éléments du présent dossier découlent :

- du domaine fonctionnel
- des spécifications techniques
- du modèle physique de donnée

2.2 - Références

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

1. **Projet_8_Dossier_d_exploitation.pdf**: Dossier d'exploitation
2. **Projet_8_Dossier_de_Conception-fonctionnel.pdf** : Dossier de conception fonctionnel
3. **Projet_8_Dossier_de_Conception-technique.pdf** : Dossier de conception technique

3 - ARCHITECTURE TECHNIQUE

Afin de répondre au mieux aux besoins du client, nous avons découpé l'application en deux parties, L'interface client et l'interface OC PIZZA

3.1 - Application OC Pizza

Le front-end est la partie visible du site-web (partie où il y a les interactions client).
Il sera réalisé à l'aide des langages de programmation *HTML, CSS et JavaScript* car ce sont de véritables standards qui sont. La base de tout projet de développement web.

Nous utiliseront le **Bootstrap** pour le front end.

Le back-end est la partie logique du site-web. Il sera réalisé avec le langage de programmation *PHP, jumelé à son Framework Symfony*. Cette interface back-end communiquera avec la base de données Mysql.

3.2 - La base de données

La base de données permet de stocker et de retrouver l'intégralité des données et informations en rapport avec le groupe de pizzerias.

3.2.1 - Base de données utilisé

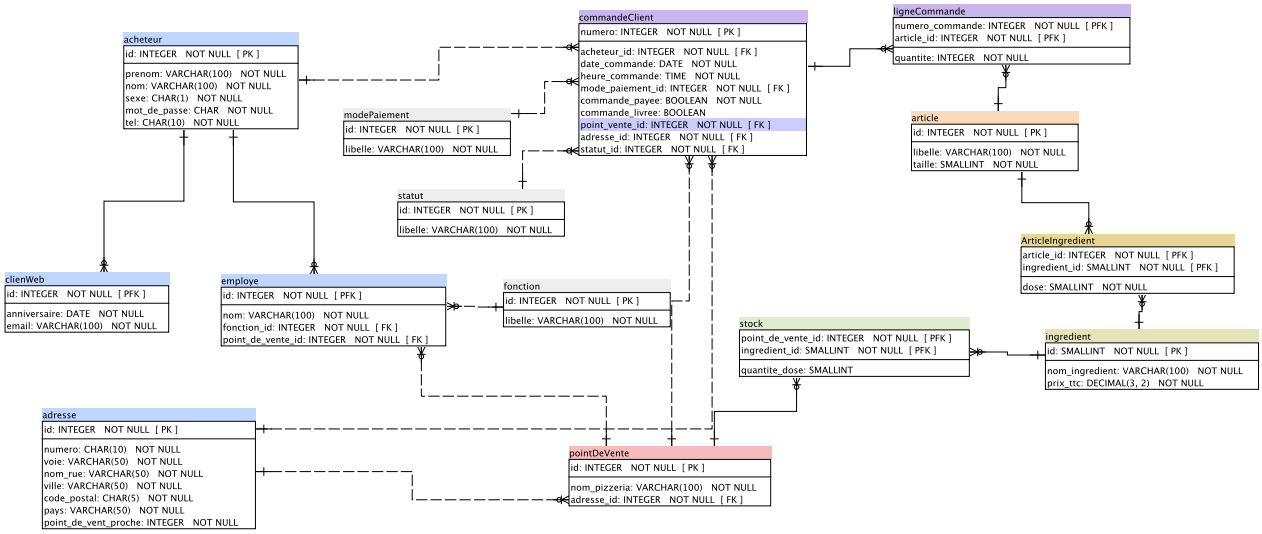
Un Système de Gestion de Base de Données est un logiciel qui permet le stockage d'informations dans une base de données. Il permet de créer, lire, mettre à jour, et supprimer les données qui sont contenues dans la base de données. Ces données seront stockées grace a mysql.

3.2.2 - MPD

A partir du diagramme de classes présenté dans le dossier de conception fonctionnelle, nous avons élaboré le MPD (Modèle Physique de Données).

Il permet d'avoir une **représentation graphique** 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 de table ainsi que les clés étrangères et primaires.**



3.2.3 - Descriptif des tables principales

Table « Acheteur »

Cette table regroupe les informations de tous les « Acheteurs », en général. **Toutes les colonnes** de cette table sont renseignées **(NOT NULL)**. **La clé primaire est id**, avec un **AUTO_INCREMENT**, et de type **INTEGER**. Pour les autres colonnes, le type est :

• **VARCHAR**, avec **50** caractères pour le **nom** et le **prenom** ; • **CHAR**, avec **10** caractères pour **numero_telephone**, **1** pour **genre** (« H » ou « F »), et **16** caractères pour le **mot_de_passe**.

Table « ClientWeb »

Cette table regroupe les informations spécifiques d'une **spécialisation** des « Acheteurs » que sont les « Clients Internet ». La colonne **email (VARCHAR, 100**) est obligatoirement renseignée **(NOT NULL).** La colonne **anniversaire** est de type **DATE.**

La colonne **acheteur_id est une clé étrangère** se référant à l'**id** correspondant dans la table **Acheteur**.

Table « Employe »

Cette table regroupe les informations spécifiques d'une **spécialisation** des « Acheteurs » que sont les employés d'OC Pizza. Ici, **toutes les colonnes sont à renseigner (NOT NULL).** Le nom est un VARCHAR (50) et la fonction un INTEGER.

La colonne **acheteur_id** est **une clé étrangère** se référant à l'**id** correspondant dans la table **Acheteur.**

Table « Adresse »

Les colonnes numero, voie, nom_rue, ville, code_postal, pays et point_de_vente_proche doivent être renseignées (NOT NULL). La colonne **commentaire** n'est pas à renseigner obligatoirement. Elle permet d'ajouter des précisions par rapport à l'adresse (étage, code...).

La **clé primaire est id**, avec un **AUTO_INCREMENT**, et de type **INTEGER**. Pour les colonnes, le type de données est :

• **VARCHAR**, avec **150** caractères pour **nom_rue** ; • **VARCHAR**, avec **50** caractères pour **voie, ville, pays** ; • **CHAR**, avec **10** caractères pour **numero** et **5** pour **code_postal** ; • **VARCHAR**, avec **300** caractères pour **commentaire** ;

Table « PointDeVente »

La colonne nom, de type **VARCHAR**, avec **100** caractères, doit être renseignée **(NOT NULL)**. La **clé primaire** est **id**, avec un **AUTO_INCREMENT**, de type **INTEGER.** La colonne **adresse_id est une clé étrangère** se référant à l'**id** correspondant dans la table **Adresse**, afin de récupérer l'adresse de chaque pizzeria.

Table « Commande »

Toutes les colonnes, doivent être renseignées **(NOT NULL).**

La clé primaire est numero, avec un **AUTO_INCREMENT**, et de type **INTEGER**. Pour les colonnes, le type de données est :

• **DATE**, pour **date_commande** ; • **TIME**, pour **heure_commande** ; • **INTEGER**, pour **statut** ; • **BOOLEAN**, pour **commande_livree** (true ou false si la commande est à livrer à domicile ou non) ; • **INTEGER**, pour **mode_paiement**; • **BOOLEAN**, pour **commande_payee** (true ou false si le paiement a été effectué ou non) ;

Il y a **3 clés étrangères** : • la colonne **acheteur_id** se référant à l'**id** correspondant dans la table **Acheteur**, afin de relier chaque commande à un « acheteur ». • la colonne **point_de_vente_id** se référant à l'**id** correspondant dans la table **PointDeVente**, afin de relier chaque commande à une pizzeria du groupe. Le choix de la pizzeria dépend du résultat du calcul de géolocalisation entre le lieu de livraison et les différentes pizzerias : la plus proche sera choisie. • la colonne **adresse_id** se référant à l'**id** correspondant dans la table **Adresse**, afin de relier chaque commande à l'adresse du client.

Table « LigneCommande »

La table **LigneCommande** est une **table d'association** (association many-to-many). La colonne **quantite**, de type **INTEGER**, doit être renseignée (**NOT NULL**). C'est la quantité de chaque type d'article commandé. Il y a **2 clés étrangères** :

• la colonne **numero_commande** de la table **Commande**, afin de relier chaque ligne de commande à une commande. • la colonne **article_id** se référant à l'**id** de la table **Article**, afin de relier chaque ligne de commande à un article.

Table « Article »

La colonne **libelle**, de type **VARCHAR**, avec **100** caractères, doit être renseignée **(NOT NULL).** La colonne **taille**, de type **SMALLINT**, doit être renseignée **(NOT NULL)**. Elle prend la valeur **1** pour les articles de taille « Format standard» et **2** pour les articles de taille « Grand Format». **La clé primaire est id**, avec un **AUTO_INCREMENT**, et de type **INTEGER**.

Table « Ingredient »

La colonne **nom_ingredient**, de type **VARCHAR**, avec **100** caractères, doit être renseignée **(NOT NULL).** La colonne **prix_ttc**, de type **DECIMAL**, de **3 chiffres dont 2 après la virgule**, doit être renseignée **(NOT NULL)**. **La clé primaire est id**, avec un **AUTO_INCREMENT**, et de type **INTEGER**.

Table « ArticleIngredient »

La table **ArticleIngredient**, n'apparaît pas dans le Diagramme de classes. C'est une **table d'association** qui doit être créée du fait de l'association many-to-many entre les tables **Article** et **Ingredient**. La colonne dose, de type **SMALLINT**, doit être renseignée **(NOT NULL)** et prend la valeur **1** pour les articles « Format normal», et **2** pour les articles « Grand Format ». C'est le nombre de dose de chaque ingrédient que nécessite chaque article.

Il y a 2 clés étrangères : • la colonne **article_id** se référant à l'id correspondant dans la table **Article**, afin de relier chaque ingrédient à un article donné. • la colonne **ingredient_id** se référant à l'id correspondant dans la table **Ingredient**, afin de relier chaque article à un ingrédient donné.

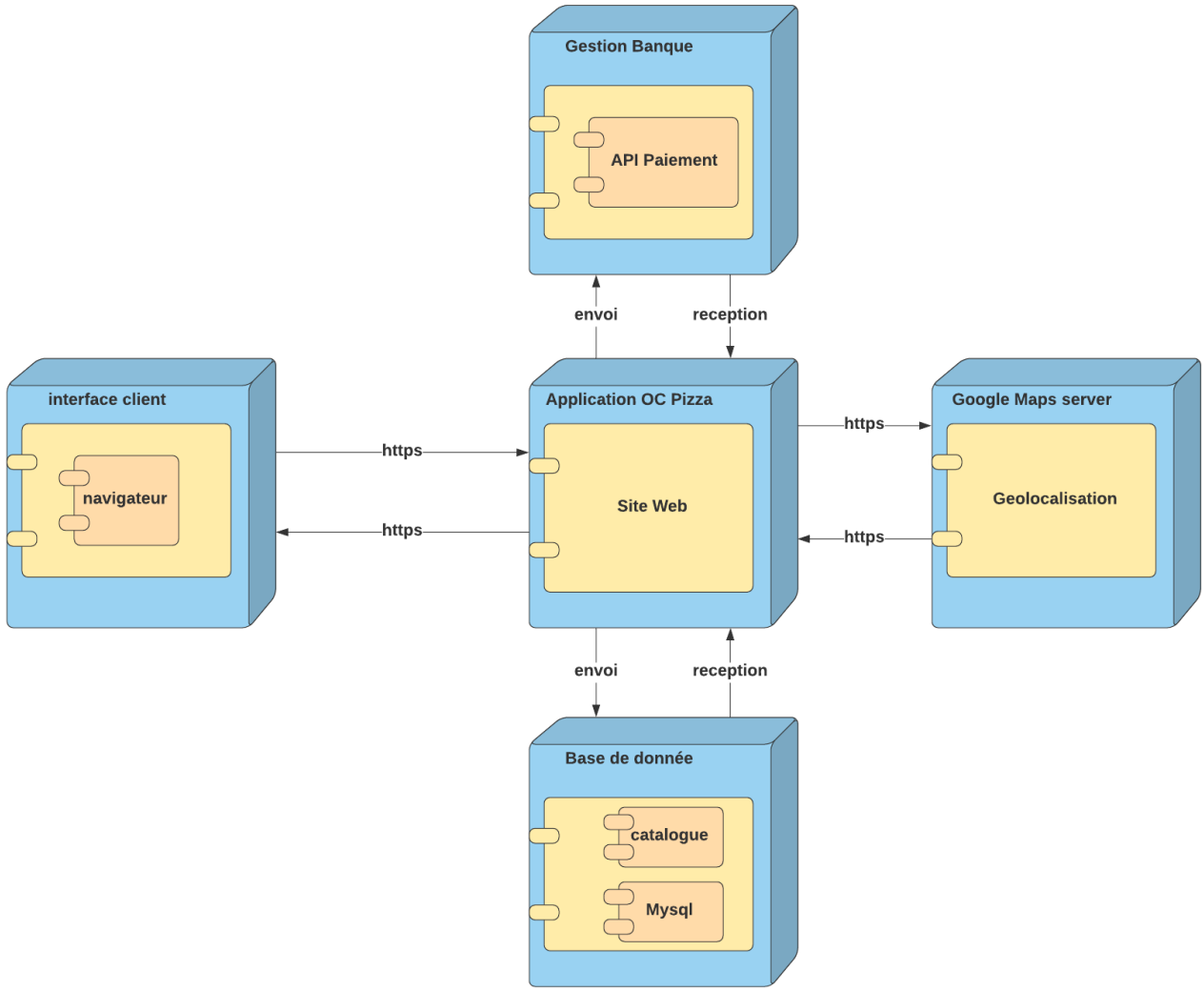
Table « Stock »

La table **Stock** est une table d'association (association many-to-many). La colonne **quantite_dose**, de type **SMALLINT**, doit être renseignée mais peut être de valeur nulle (fin de stock d'un ingrédient). C'est la quantité de doses que chaque pizzeria possède de chaque ingrédient. Il y a 2 clés étrangères :

• la colonne **point_de_vente_id** se référant à l'**id** correspondant dans la table **PointDeVente**, afin de relier chaque pizzeria donnée à un stock d'ingrédient. • la colonne **ingredient_id** se référant à l'**id** correspondant dans la table **Ingredient**, afin de relier un stock d'ingrédient donné à chaque pizzeria.

4 - Architecture de Déploiement

4.1 - Diagramme de deploiement



Comme on peut le voir dans le diagramme, il y a 5 nœuds qui représentent les éléments hardware ou software faisant partie du système.

-> ***Appareil Utilisateur*** indique l'ordinateur avec lequel l'utilisateur se connecte. L'**Interface Client** représente l'interface utilisateur, à travers laquelle il peut accéder au site web d'OC Pizza (via le **Navigateur Web**).

-> ***Serveur Banque*** indique le serveur qui gère les requêtes pour les paiements en ligne, effectués grâce a ***STRIPE*** (**API Paiement**). Le transfert des données entre les deux nœuds se fait grâce au protocole https.

-> ***Application OC Pizza*** indique l'ensemble de pages web qui composent le site web Oc Pizza, comme décrit par le composant **Site Web**. Le transfert de données entre ce nœud et le nœud **Appareil Utilisateur** se fait grâce au protocole HTTPS.

-> ***Serveur OC Pizza*** indique le serveur chargé de gérer les requêtes des pages du site web. Il repère les informations grâce à la base de données dédiée **Base de données OC Pizza** et les affiche à l'utilisateur sous forme de **Catalogue**. Les interactions avec la base de données sont assurées par **Mysql**. L'échange des données entre les nœuds **Application OC Pizza** et **Serveur OC Pizza** se fait toujours avec HTTPS.

-> **Serveur Google Maps** indique le serveur Google chargé de gérer les requêtes de géolocalisation.
Le transfert des données entre ce nœud et **Serveur OC Pizza** est réalisé encore à l'aide du protocole HTTPS.
Le **Serveur Google Maps** est relié via HTTPS aussi au nœud **Appareil Utilisateur** et il y a une relation de dépendance entre les composants **Interface Client** et **Géolocalisation**.

4.2 - Serveur de deploiement

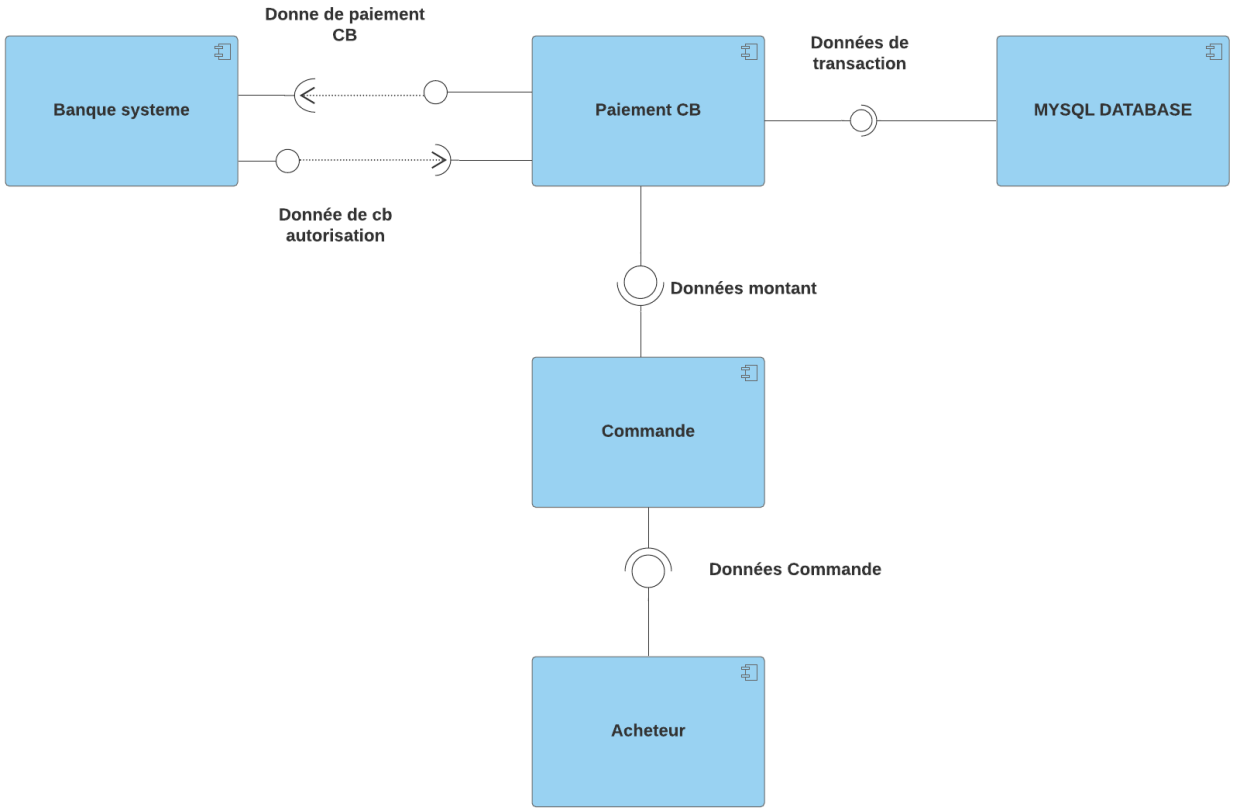
OC PIZZA sera déployée sur le serveur OVH, entreprise française implantée à l'étranger proposant des temps de latence réduits et un réseau ultra-sécurisé.

L'offre d'hébergement Pro (1 nom de domaine, 250 Go d'espace disque, 100 adresses e-mails, Trafic illimité) dont le prix est de 7,19 euros TTC par mois proposée par OVH est une solution adaptée à la taille du projet.

5 - COMPOSANT

5.1.1 - Diagramme de composant paiement

Les Diagrammes de composants (Component Diagrams) décrivent l'organisation du système du point de vue des éléments logiciels. Ils mettent en évidence les dépendances entre les composants, et décrivent ici les interfaces entre les composants internes du système OC Pizza et les composants externes (Banque)



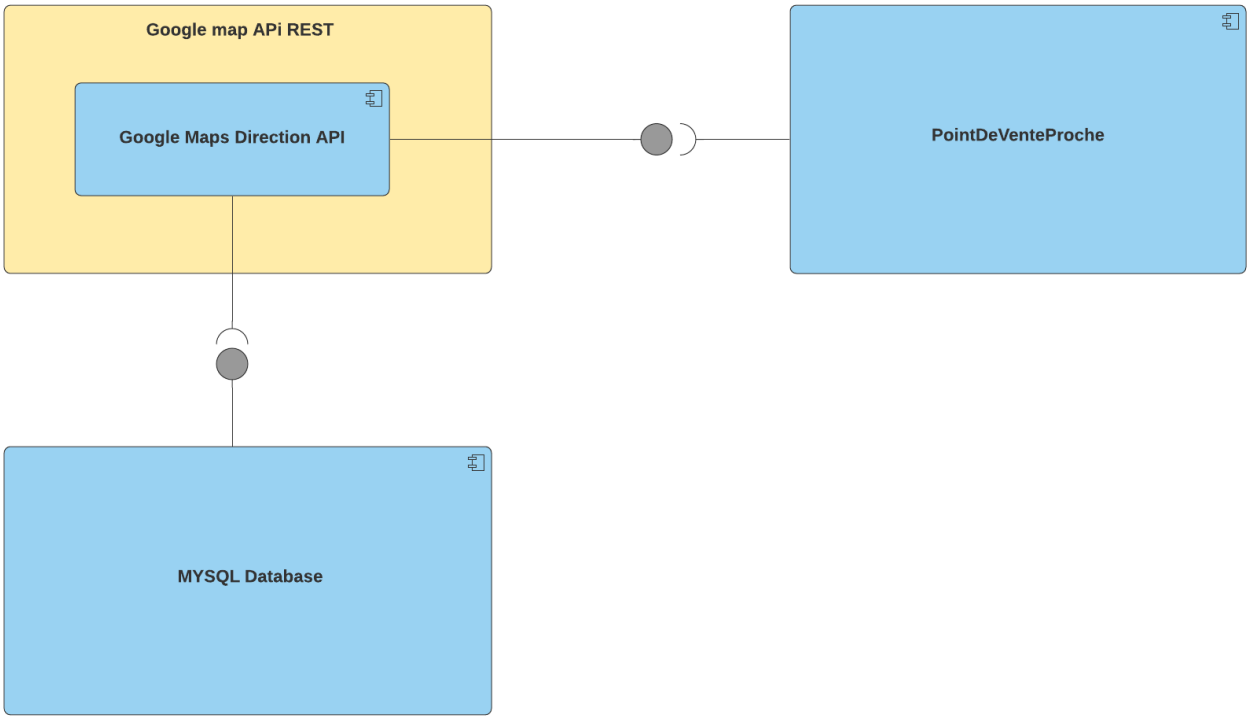
5.1.2 - Description diagramme de composant

Le **composant Acheteur envoie au composant Commande les informations sur la commande** et celui-ci calcule le montant total. Ce montant est requis par le **composant Paiement CB** qui récupère aussi le nom du client.

Il échange avec le système bancaire des informations sur le client et le montant à régler. Le **client** aura directement accès au service de carte bancaire de la banque pour rentrer ses **coordonnées personnelles (numéro de carte, date d'expiration…).** En retour, le **composant Banque système** renvoie au composant interne les l'information sur la réalisation du paiement.

Cette information sera **gardée en base de données, via le composant Mysql** Database, le système d'OC Pizza.

5.1.3 - La geolocalisation



5.1.4 - Description diagramme Geolocalisation

Le composant Mysql Database envoie à l'API REST Google Maps Directions les informations sur l'adresse de livraison d'un client et l'adresse des différentes pizzerias du groupe OC Pizza.

Ce composant externe **calcul le temps de parcours** et l'itinéraire à privilégier, pour un véhicule, entre cette adresse et celles des points de vente et **envoie l'information à** l'interface requise du **composant PointDeVenteProche** qui va **retenir la pizzeria la plus proche du lieu de livraison**.
Cette information (la pizzeria la plus proche) sera **stockée dans la base de données** dans la colonne **point_de_vente_proche de la table Adresse** pour l'adresse de livraison donnée.

5.1.5 - Structure des sources Symfony

La structuration des répertoires du projet suit la logique suivante :

```
OC Pizza
 config/
├─ services.yaml
├─ ...
└─ packages/
    ├─ framework.yaml
    ├─ ...
    ├─ **dev/**
    │   ├─ monolog.yaml
    │   └─ ...
    ├─ **prod/**
    │   └─ monolog.yaml
    └─ **test/**
        ├─ framework.yaml
        └─ ...
└─ routes/
    ├─ annotations.yaml
    └─ **dev/**
        ├─ twig.yaml
        └─ web_profiler.yaml
```

6 - POINTS PARTICULIERS

6.1 - Gestion des logs

Pour accéder aux statistiques du site, aller sur :

https://logs.ovh.net/ocpizza

La connexion se fait via l'identifiant OVH et le mot de passe associé.
Cliquez sur le lien généré automatiquement dans le **Manager** pour accéder aux statistiques et aux logs. Il faudra s'identifier avec la référence client (Nic-handle) et le mot de passe.

Espace statistiques

Une fois connecté à l'espace de statistiques, deux possibilités :
- Accéder aux statistiques du site via urchin v6.
- Consulter les logs bruts en temps réel ou sur une période antérieure

Urchin v6

Ces statistiques donnent des renseignements sur le trafic du site

- Le nombre de visiteurs,
- Le nombre de pages visualisées,
- Le "poids" des pages visualisées,
- Le nombre de requêtes http.
- Les durées moyennes de connexion à l'ensemble de du site ou une page particulière
- Comment les visiteurs du site l'ont-ils connu ?
- Par quels moteurs de recherche ont-ils trouvé l'URL de du site ?
- Quels mots-clés ont-ils utilisé lors de leur recherche ?

Logs bruts

Il est possible de visualiser les logs du site pratiquement en direct en moins de 15 minutes, ce qui permet de vérifier le bon fonctionnement du site presque en temps réel.
Différents types de logs sont à disposition :

- Logs Web : trouvez ici les différents logs de consultation du site, ainsi que les différentes actions réalisées à partir du site. Cela permet par exemple de repérer des tentatives de hacks.
- Logs FTP : les différentes connexions FTP seront enregistrées et conservées dans ces logs.
- Logs erreur : les différentes erreurs générées par le site.
- Logs CGI : les différents appels aux scripts cgi.bin qui ont été réalisés.
- Logs out : les statistiques de l'hébergement sur les différents appels externes réalisés.
- Logs SSH : ces logs indiquent les différentes connexions réalisées avec le protocole SSH.
- Logs cron : le résultat de l'exécution des tâches planifiées

6.2 - Base de données

Un fichier SQL est fournis avec les livrable, ce fichier contient les script pour la creation de la base de données.

6.3 - Environnement de développement

L'IDE pour le projet symfony est visual studio code.

6.4 - Procédure de packaging / livraison

L'application fera l'objet d'un déploiement sur le serveur OVH au moment de la livraison finale.
Les identifiants et mots de passe seront donnés au client après la mise en ligne et le paiement de ce dernier.

7 - GLOSSAIRE

MPD	Modele physique de données