OC PIZZA

Vente de pizzas en ligne

Dossier de conception technique

Version 1.0

Auteur
Maxime Jacquette
Analyste-programmeur

TABLE DES MATIERES

1 - Versions **3**
2 - Introduction **4**
2.1 - Objet du document 4
2.2 - Références 4
3 - Architecture Technique **5**
3.1 - Application Web 5
3.2 - Base de données 5
3.2.1 - SGBD utilisé *5*
3.2.2 - MPD *6*
3.2.3 - Présentation des tables *7*
4 - Architecture de Déploiement **13**
4.1 - Déploiement de l'application 13
4.2 - Serveur de déploiement 15
5 - Architecture logicielle **16**
5.1 - Principes généraux 16
5.1.1 - Les couches *16*
5.1.2 - Le diagramme de composants *16*
5.1.3 - Structure des sources *19*
6 - Points particuliers **21**
6.1 - Ressources 21
6.1.1 - Charte graphique *21*
6.1.2 - Jeu de données *21*
6.2 - Environnement de développement 21
6.3 - Procédure de packaging et de livraison 21
7 - Logs et monitoring **22**
7.1 - Système de logs 22
7.2 - Monitoring 22
8 - Glossaire **23**

1 - VERSIONS

Auteur	Date	Description	Version
Maxime J.	09/02/2019	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 à l'attention des développeurs, mainteneurs et de l'équipe technique d'OC PIZZA.

L'objectif de l'analyse technique étant de lister les contraintes spécifiques dont les développeurs vont devoir tenir compte pour coder l'application, le présent document présentera les langages et les conventions de développement, les conventions de développement, l'architecture logicielle et de déploiement ainsi que les logs, le monitoring propres à l'application.

Les éléments du présent dossier découlent :

- de notre premier entretien datant du 11 décembre 2018.
- du document de spécifications techniques réalisé par notre entreprise Python Circles.

2.2 - Références

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

1. **Projet 9 - Dossier de conception fonctionnelle.pdf – 1.0** : Dossier de conception fonctionnelle de l'application
2. **Projet 9 - Dossier d_exploitation.pdf – 1.0** : Dossier d'exploitation de l'application.
3. **Projet 9 - PV Livraison.pdf – 1.0** : Procès-verbal de livraison finale.

3 - Architecture Technique

Afin de répondre au mieux aux besoins du client, nous avons découpé l'application en deux parties, **l'application front-end** et **back-end**.

3.1 - Application Web

Le **front-end** est la partie visible du site-web (partie où il y a les interactions client). Nous proposons de le réaliser à l'aide des langages de programmation HTML, CSS et JavaScript (bibliothèque JavaScript). Pourquoi ? Car ce sont de véritables standards qui sont à la base de tout projet de développement web.

Le **back-end** est la partie logique du site-web. Nous proposons de le réaliser à l'aide du langage de programmation Python, jumelé à son Framework Django. Cette interface back-end communiquera avec la base de données MySQL. Pourquoi Python ? Car Python est un langage polyvalent, multiplateforme et open source. Le système est fiable, énormément utilisé, standard de l'industrie (Google, Instagram ou encore Dropbox ont sont développés en Python). Pourquoi recourir à un MVT comme Django ? Car grâce à la séparation des données de la vue et du contrôleur, sa conception devient plus claire et efficace, car cela permet un gain de temps de maintenance et d'évolution du site et car de nombreux sites internet recourent à un MVT pour modifier facilement les contenus.

3.2 - Base de données

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

3.2.1 - SGBD utilisé

Un Système de Gestion de Base de Données (SGBD) est un logiciel qui permet le stockage d'informations dans une base de données. Il permettra de créer, lire, mettre à jour, et supprimer les données qui sont contenues dans la base de données.

Nous proposons de stocker ces données dans le SGBDR MySQL. Pourquoi MySQL ? Car MySQL est la base de données open source la plus populaire au monde. MySQL, par le fait qu'il ne gère pas l'intégrité référentielle par exemple, se révèle plus rapide que d'autres SGBDR comme PostgreSQL puisqu'il ne doit pas faire les tests d'intégrité (qui permettent de vérifier qu'une base de données est cohérente). L'utilisation de MySQL est plus répandue que celle de d'autres SGBDR, tels que PostgreSQL.

3.2.2 - MPD

À partir du diagramme de classes présenté dans le dossier de conception fonctionnelle, nous pouvons élaborer le MPD.
Le modèle physique des données 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.

MPD

3.2.3 - Présentation des tables

La table « PizzaStore »

La **table** dispose d'une clé primaire *iD*, avec un AUTO_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacun des points de vente. Pour le moment, l'entreprise n'en possède que cinq, il est donc inutile de lui affecter un type MEDIUMINT ou SMALLINT.

Pour les autres *attributs*, le type est :

- VARCHAR, limité à 150 caractères pour *name*.
- DECIMAL, limité à 15 chiffres pour la partie entière et deux chiffres pour la partie décimale, pour *revenue*.
- TINYINT, pour *team_size* car il est peu probable, ne serait-ce qu'une des pizzerias, compte plus de 127 salarié-es.
- TIME, pour *opening_hour* et *closing_hour*, car ici nous n'avons pas besoin d'un type DATETIME, l'heure suffit.

Elle possède également une clé étrangère de type MEDIUMINT, *address_id* se référant à l'*iD* de la table **Address**.

Tous les attributs de la table **PizzaStore** doivent être renseignés (NOT NULL), sauf *revenue*, et *team_size*, des informations qui peuvent être considérées comme confidentielles pour être omises.

La table « Address »

La **table** dispose d'une clé primaire *iD*, avec un AUTO_INCREMENT, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacune des adresses.

Pour les autres *attributs*, le type est :

- VARCHAR, limité à 50 caractères pour *road_category, street_number,* et *city.*
- VARCHAR, limité à 150 caractères pour *complement* et *email.*
- CHAR, limité à 5 caractères pour *zipcode*.

Elle possède la clé étrangère *supplier_id*, se référant à l'*iD* de la table **Supplier**. On crée une clé étrangère dans la table correspondant à la cardinalité maximum.

supplier_id est de type TINYINT.

Tous les attributs de la table **Address** doivent être renseignés (NOT NULL) sauf *complement* et *supplier_id*.

La table « Team »

La **table** dispose d'une clé primaire *iD*, avec un AUTO_INCREMENT, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacun des salariés. Nous pensons qu'un TINYINT serait insuffisant, au vu de la taille des équipes, et devant l'éventualité que cette équipe change.

Pour les autres *attributs*, le type est :

- ENUM pour *position* qui peut prendre comme valeur l'un des cinq postes occupés par le salarié.
- VARCHAR, limité à 150 caractères pour *workplace.*
- VARCHAR, limité à 100 caractères pour *first_name* et *last_name*.
- CHAR, limité à 11 caractères pour *phone_number*.
- CHAR, limité à 1 caractère pour *gender* (H ou F).
- VARCHAR, limité à 32 caractères pour *password,* un mot de passe crypté.

L'option de mettre la clé étrangère *pizza_store_id* de type TINYINT et se référant à l'*iD* de la table **PizzaStore** dans la table **Team** se justifie par le fait qu'on puisse considérer ici plus « naturel » de placer les points de vente autour de l'équipe que de faire rentrer un point de vente dans un membre de l'équipe.

Tous les attributs de la table **Team** doivent être renseignés (NOT NULL).

La table « Client »

Client dispose d'une clé primaire *num_client*, avec un AUTO_INCREMENT, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacun des clients.

Outre ce MEDIUMINT pour la clé primaire, elle dispose des types suivants :

- VARCHAR, limité à 100 caractères pour *first_name* et *last_name*.
- CHAR, limité à 11 caractères pour *phone_number*.
- CHAR, limité à 1 caractère pour *gender* (H ou F).
- VARCHAR, limité à 32 caractères pour *password,* un mot de passe crypté.

Ces attributs doivent tous être renseignés (NOT-NULL).

La table « OrderPizzeria »

La **table** dispose d'une clé primaire *order_number*, et elle est de type MEDIUMINT, un identifiant unique avec AUTO INCREMENT permettant de différencier chacune des commandes.

Pour les autres *attributs*, le type est :

- ENUM pour *status* et *payment_method* qui peuvent prendre respectivement comme valeurs les statuts de commande et les moyens de paiement précédemment listés.
- DATETIME, pour *date*. Ici il est primordial d'avoir l'heure, le jour etc. de la commande.
- BOOLEAN, pour *payment made*, qui se charge de vérifier si le paiement a eu lieu ou non (True ou False).

Les attributs *client_id* et *team_id* sont des clés étrangères de type MEDIUMINT se référant respectivement au *num_client* de la table **Client** et à l'*iD* de la table **Team**.

Tous les attributs de la table **OrderPizzeria** doivent être renseignés (NOT NULL).

La table « Billing »

La **table** dispose d'une clé primaire *iD*, et elle est de type MEDIUMINT, un identifiant unique permettant de différencier chacune des factures.

Pour les autres *attributs*, le type est :

- DECIMAL, limité à 6 chiffres pour la partie entière et 2 chiffres pour la partie décimale, pour *amount*, le montant.

Outre ces attributs, elle possède une clé étrangère de type MEDIUMINT se référant à *order_number* de la table **OrderPizzeria.**

Tous les attributs de la table **Billing** doivent être renseignés (NOT NULL).

La table « Ingredient »

La **table** dispose d'une clé primaire *iD*, avec AUTO_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacun des ingrédients.

Pour les autres *attributs*, le type est :

- VARCHAR, limité à 150 caractères pour *name*.

Tous les attributs de la table **Ingredient** doivent être renseignés (NOT NULL).

La table « Pizza »

La **table** dispose d'une clé primaire *iD*, avec AUTO_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacune des pizzas. Nous pensons que le client OC_PIZZA ne commercialise pas plus de 127 pizzas, il est donc inutile de lui affecter un type MEDIUMINT ou SMALLINT.

Pour les autres *attributs*, le type est :

- VARCHAR, limité à 150 caractères pour *name*.
- DECIMAL, limité à 4 chiffres pour la partie entière et 2 chiffres pour la partie décimale, pour *cost* et *price*.

L'attribut *recipe_id* est une clé étrangère se référant à l'*iD* de la table **Recipe**. Elle est de type TINYINT.

Tous les attributs de la table **Pizza** doivent être renseignés (NOT NULL).

La table « Recipe »

Cette **table** dispose d'un *iD*, identifiant unique avec AUTO INCREMENT et elle regroupe les informations de toutes les recettes de pizzas.

La **table** dispose d'un *attribut* dont le type est :

- MEDIUMTEXT pour *comment*.

Tous les attributs de la table **Recipe** doivent être renseignés (NOT NULL).

La table « Supplier »

La **table** dispose d'une clé primaire *iD*, avec un AUTO_INCREMENT, et elle est de type TINYINT, un identifiant unique permettant de différencier chacun des fournisseurs.

La **table** possède un autre *attribut*, le type est :

- VARCHAR, limité à 150 caractères pour *name*.

Il doit être renseigné (NOT NULL).

La table d'association « ClientDetails »

Cette **table** d'association regroupe deux clés primaires étrangères [PFK] *client_id* se référant à *num_client* de la table **Client**, et *address_id* se référant à l'*iD* de la table **Address** qui sont toutes les deux de type MEDIUMINT.

Outre ces *attributs*, le type est :

- DATETIME, pour *membership_start_date*, la date d'adhésion.

Tous les attributs de la table doivent être renseignés (NOT NULL).

La table d'association « TeamDetails »

TeamDetails est une table d'association regroupant deux clés primaires étrangères [PFK] *team_id* se référant à l'*iD* de la table **Team**, et *address_id* se référant à l'*iD* de la table **Address**. Elles sont de type MEDIUMINT.

Outre ces *attributs*, le type est :

- DATETIME, pour *membership_start_date, la* date d'adhésion.

Tous les attributs de cette table d'association doivent être renseignés (NOT NULL).

La table d'association « Stock »

La **table** dispose de deux clés primaires étrangères [PFK], qui sont de type TINYINT :

- ingredient_id, se référant à l'*iD* de la table **Ingredient**.
- pizza_store_id, se référant à l'*iD* de la table **PizzaStore**.

Outre ces *attributs*, le type est :

- DECIMAL(10, 2) pour *quantity,* la quantité. L'unité de mesure sera par défaut le kilogramme.

Tous les attributs de la table **Stock** doivent être renseignés (NOT NULL), sauf *quantity* car il se peut qu'un ingrédient soit en rupture de stock.

La table d'association « OrderSupplier »

La **table** dispose de deux *attributs* dont le type est :

- DATETIME pour *date*.
- DECIMAL(10,2) pour *quantity*.

Les attributs *ingredient_id et supplier_id* sont des clés primaires étrangères se référant respectivement à l'*iD* de la table **Ingredient et** à l'*iD* de la table **Supplier**. Ces attributs sont de type TINYINT.

La table d'association « CommandLine »

Cette **table** regroupe deux clés primaires étrangères [PFK] *order id* se référant à *order number* de la table **OrderPizza**, et *pizza_id* se référant à l'*iD* de la table **Pizza** qui sont respectivement de type MEDIUMINT et TINYINT.

Outre ces *attributs*, le type est :

- ENUM, pour *pizza_size* et *pizza_crust*, qui peuvent prendre les valeurs énoncées plus haut. Pour *pizza_size*, nous proposons de travailler avec des échelles. Ainsi la valeur 1 correspondra à une pizza de taille standard, 1.5 à une pizza medium, 2 à une pizza familiale etc...
- TINYINT pour *quantity*.

La table d'association « Composition »

Cette table d'association stocke deux clés primaires étrangères [PFK] *recipe_id* se référant à *iD* de la table **Recipe**, et *ingredient_id* se référant à l'*iD* de la table **Ingredient** qui sont de type TINYINT.

Elle possède également l'attribut *quantity*, de type DECIMAL (10, 2).

Tous les attributs doivent être renseignés (NOT NULL).

4 - Architecture de Deploiement

4.1 - Déploiement de l'application

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

Diagramme UML de déploiement

Nous avons identifié six environnements utilisateurs.

Les matériels des utilisateurs

Il s'agit d'ordinateurs, tablettes ou smartphones que le client et l'équipe utilisent pour se connecter à l' application (site web).

Le personnel et les clients s'y rendent pour passer des commandes. Les l'utilisent pour passer commande sans intermédiaire. L'équipe peut passer une commande pour le client lorsque celui-ci souhaite récupérer sa commande sur place, ou lorsque celui-ci téléphone.

Les cadres utilisent l'application principalement pour gérer les pizzerias d'un point de vue financier. Les clients et l'équipe utilisent un navigateur web qui, va lancer des requêtes HTTPS et permettre de se connecter au deuxième nœud, le serveur d'application (Application Server).

Le serveur d'application

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

Le front-end utilise comme ressources les langages de programmations HTML, CSS, et JavaScript (librairie jQuery).
Quant au back-end, il sera construit en langage Python, jumelé à son Framework Django.

Des notifications peuvent être envoyées au client ou à l'équipe, résultant de divers évènements liés au processus de commande.

Le serveur communiquera avec des services externes comme l'hôte Google (Google Host), les marchés financiers (Financial markets), la base de données (MySQL server) et le serveur bancaire (Bank). Ces communications sont détaillées dans chacun des services cités.

Le serveur de base de données

Un protocole HTTP permet au nœud serveur d'application de communiquer avec le nœud serveur de base de données qui est considéré comme un service externe.
Ce serveur héberge la base de données MySQL 5.2.

Le serveur bancaire

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

Les marchés financiers

Sur les marchés, les échanges concernent également les matières premières. Le rôle des cadres de l'entreprise est de négocier celles-ci.

Le prix de ces matières premières fluctue en fonction de la demande. Le rôle du patron est d'anticiper une éventuelle hausse des prix, et éventuellement trouver un substitut à un ingrédient d'une pizza.

Ce service communique avec le serveur d'application.

Google Host
Ici, il est fort probable que les clients découvrent la pizzeria via un système de navigation comme Google MAPS, celui-ci lui renvoyant une liste des pizzerias à proximité.

Le protocole REST API permet à l'application serveur de communiquer avec le matériel du client.

4.2 - Serveur de déploiement

L'application 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.

Nous pensons que 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 et proposé par OVH semble une solution adaptée à la taille du projet.

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

Le développement de l'application se présente sous la forme d'un projet Django, divisé en trois parties :

- Ventes
- Administration Restaurant
- Gérance Cadres

Django utilise un architecture-pattern proche de MVC : le MVT. MVT signifiant « Modèle - Vue – Template ».

Le modèle permet de faire de requêtes à la base de données. La vue reçoit des requêtes HTTP et va lire dans la base l'élément correspondant, pour générer le Template. Le Template est un fichier HTML qui reçoit des objets Python et qui est lié à une vue.

5.1.1 - Les couches

L'architecture applicative est la suivante :

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

5.1.2 - Le diagramme de composants

Un diagramme de composants représente la structure du système logiciel, en décrivant les composants du logiciel, ses interfaces entre composants internes externes ainsi que ses dépendances.

Diagramme UML de composants

Nous avons identifié, pour l'application, 12 composants.
Dans cette partie nous décrirons les composants internes et externes utilisés par l'application.

Le navigateur web

Le navigateur web **(Web browser)** est un logiciel ou interface graphique permettant de naviguer sur le World Wide Web.
Les navigateurs web classiques tels que Firefox 65.0, Google Chrome 72.0.3626.81, Internet explorer 11.0.101, Safari 12.0 ou Ecosia 4.0.4 seront ainsi utilisables pour accéder à l'application (site web).

Le catalogue

Le composant **List pizzas** va permettre au client de consulter le catalogue des pizzas commercialisées. C'est à partir de ce composant qu'il fera son choix parmi la gamme de produits.

Le tableau de bord

L'intérêt du composant **Control Panel** est d 'avoir accès pour tout l'équipe a toutes les informations en instantané. Il permet également de suivre la performance des différents points de ventes pour le patron et d'être alerté en cas d'anomalies.

Le panier

Il s'agit [**Cart**] du dispositif du site marchand en ligne dans lequel se trouvent tous les articles sélectionnés durant la visite par les clients. Il calcule également le total à régler. Le panier se transforme en commande au moment de la validation.

L'authentification

Le composant **Authentication** permet l'inscription de nouveaux clients à l'application ainsi que les connexions de ceux-ci et de l'équipe. Pour le client, cette étape est nécessaire pour pouvoir passer une commande. Elle est également obligatoire pour l'équipe, pour avoir accès à la gestion des commandes notamment.

Les notifications

Le système de notification **(Notifications)** permettra par exemple de transmettre des informations au client sur sa commande. Il pourra s'agir d'alertes push ou de SMS reçus. Concernant l'équipe, il s'agira d'alertes push envoyés sur le tableau des commandes pour avertir l'équipe d'une nouvelle commande à gérer etc.

Les commandes

La gestion des commandes **(Orders)** est utile aux employés pour assurer le bon fonctionnement des différentes étapes, de la validation de la commande à sa livraison.

Les clients

La gestion des clients **(Clients)** est le plus utile au pizzaiolo qui doit suivre les instructions concernant la commande ainsi qu'au livreur qui doit trouver l'adresse de livraison, si le client a choisi de se faire livrer.

Les comptes

Le composant comptes **(Accounts)** gère les comptes clients et équipe. Le manager ainsi que le patron peuvent s'en charger. Par exemple, il se peut que l'équipe s'agrandisse, il sera alors nécessaire d'ajouter ce membre à l'équipe. Dans le cas contraire, lorsqu'un membre démissionne, il sera nécessaire de le supprimer.

L'inventaire

Le composant **Inventory** (Inventaire) permet à l'équipe ainsi qu'aux cadres d'OC PIZZA de pouvoir contrôler les matières premières reçues par les fournisseurs, les marchandises consommées par la cuisine des différents points de vente ainsi que celles actuellement en stock.

Le paiement

Le composant **Payment** (Paiement) permet au client de régler le montant de sa commande. Il gèrera aussi le transfert de cet argent sur le compte bancaire de l'entreprise.

La base de données MySQL

La base de données MySQL sera déployée sur un serveur tournant sous Debian 9. Elle stocke toutes les tables citées précédemment lors de l'analyse du MPD.

5.1.3 - 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 »)

```
Projet_OC_PIZZA/
├── docs
├── requirements.txt
├── README.md
├── OC_PIZZA/
│   ├── manage.py
│   ├── settings/
│   │   ├── __init__.py
│   │   └── settings.py
│   ├── urls.py
│   └── wsgi.py
├── ventes/
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations/
│   │     └── __init__.py
│   ├── models.py
│   ├── views.py
│   └── tests.py
├── restaurants/
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations/
│   │     └── __init__.py
│   ├── models.py
│   ├── views.py
│   └── tests.py
├── cadres/
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations/
│   │     └── __init__.py
│   ├── models.py
```

```
│   ├── views.py
│   └── tests.py
├── static/
│   ├── style.js
│   ├── images/
│   └── style.css
└── templates/
    └── base.html
```

6 - POINTS PARTICULIERS

6.1 - Ressources

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

6.1.1 - Charte graphique

Le groupe de pizzerias OC PIZZA doit nous faire parvenir un cahier des charges graphique (couleurs, polices, logos, symboles...) définissant l'identité visuelle (caractéristiques des différents éléments graphiques) de l'entreprise.

6.1.2 - Jeu de données

Les données de base (pizzas commercialisées, points de vente, coordonnées des différents salariés, des fournisseurs etc.) implémentées dans la base de données de l'application sont également du ressort d'OC pizza.

6.2 - Environnement de développement

L'utilisation d'un IDE (environnement de développement) est recommandée tel PyCharm. Vous pouvez aussi développer en utilisant un éditeur de texte au choix des développeurs. Néanmoins, nous recommandons l'installation de l'assistant de programmation Kite, un système de complétion de code Python.

6.3 - Procédure de packaging et de livraison

L'application fera l'objet d'un déploiement sur le serveur OVH au moment de la livraison finale.

À cette occasion le dossier d'exploitation **Projet 9 - Dossier d_exploitation.pdf – 1.0** permettant de lister tout ce dont le client a besoin pour assurer la continuité de l'utilisation de l'application lui sera remis.

7 - LOGS ET MONITORING

7.1 - Système de logs

Le Logs Data Platform d'OVH offre une plateforme 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.

7.2 - Monitoring

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 datacentre.

Tous les serveurs de nos clients 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 (non réponse aux pings) afin de limiter au maximum le temps d'indisponibilité des serveurs et du réseau.

8 - GLOSSAIRE

SGBDR	Système de Gestion de Bases de Données Relationnelles
MPD	Modèle physique de données