

Loïc ROMERO

Analyste-programmeur

loic.romero1@gmail.com

**Résumé**

L’objectif du document est de proposer une description du domaine fonctionnel, de décrire les différents composants du système, les composants externes utilisés par celui-ci, leurs interactions et de faire la description de l’organisation physique de ces composants.

*Version 2.1 – 03/04/2021*

Dossier de SPECIFICATIONS TECHNIQUES

**« Projet 9 : Documentez votre système de gestion de pizzeria »**



**Table des matières**

[1 - Versions 2](#_Toc68358209)

[2 - Introduction 3](#_Toc68358210)

[2.1 - Objet du document 3](#_Toc68358211)

[2.2 - Références 3](#_Toc68358212)

[2.3 - Besoin du client 4](#_Toc68358213)

[2.3.1 - Contexte 4](#_Toc68358214)

[2.3.2 - Enjeux et Objectifs 4](#_Toc68358215)

[3 - Architecture Technique 5](#_Toc68358216)

[3.1 - Application 5](#_Toc68358217)

[3.1.1 - Back-end : 5](#_Toc68358218)

[3.1.2 - Front-end : 5](#_Toc68358219)

[3.2 - Modèle physique de données 6](#_Toc68358220)

[3.2.1 - Synthèse 6](#_Toc68358221)

[3.2.2 - Schéma du modèle physique de données 6](#_Toc68358222)

[3.2.3 - Détail des tables 8](#_Toc68358223)

[4 - Architecture de déploiement 17](#_Toc68358224)

[4.1 - Synthèse 17](#_Toc68358225)

[4.2 - Diagramme de déploiement 17](#_Toc68358226)

[4.3 - Détail des terminaux 19](#_Toc68358227)

[4.3.1 - « External » 19](#_Toc68358228)

[4.3.2 - « Group » 19](#_Toc68358229)

[4.3.3 - « Users » 19](#_Toc68358230)

[5 - Architecture logicielle 20](#_Toc68358231)

[5.1 - Principes généraux 20](#_Toc68358232)

[5.1.1 - Les couches 20](#_Toc68358233)

[5.1.2 - Diagramme de composants 20](#_Toc68358234)

[5.1.3 - Structure des sources 23](#_Toc68358235)

[5.2 - Application Web 23](#_Toc68358236)

[6 - Points particuliers 24](#_Toc68358237)

[6.1 - Gestion des logs 24](#_Toc68358238)

[6.2 - Monitoring 25](#_Toc68358239)

[6.3 - Ressources 26](#_Toc68358240)

[6.3.1 - Charte graphique 26](#_Toc68358241)

[6.3.2 - Maquettes 26](#_Toc68358242)

[6.3.3 - Base de données de développement et finale. 26](#_Toc68358243)

[6.4 - Environnement de développement 26](#_Toc68358244)

[6.5 - Procédure de packaging / livraison 26](#_Toc68358245)

[7 - Glossaire 27](#_Toc68358246)

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Loïc ROMERO | 07/09/2020 | Création du document. | 1.0 |
| Loïc ROMERO | 21/09/2020 | Ajout du plan. | 1.1 |
| Loïc ROMERO | 23/09/2020 | Ajout des schémas. | 1.2 |
| Loïc ROMERO | 24/09/2020 | Rédaction (Chap. : 2.1 – 2.2 – 2.3 – 3.1) | 1.3 |
| Loïc ROMERO | 25/09/2020 | Rédaction (Chap. : 3.3) | 1.4 |
| Loïc ROMERO | 26/09/2020 | Rédaction (Chap. : 4.1 – 4.3) | 1.5 |
| Loïc ROMERO | 28/09/2020 | Rédaction (Chap. : 5 – 6) | 1.6 |
| Loïc ROMERO | 14/03/2021 | Remaniement de la version 1.6 du Projet 6 | 2.0 |
| Loïc ROMERO | 03/04/2021 | Relecture | 2.1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Objet du document

Le présent document constitue le **dossier de spécifications technique** du « Projet 9 : Documentez votre système de gestion de pizzeria ».

Il fait suite au dossier de spécifications fonctionnelles. Il est issu des conversations avec le client.

**L’objectif** du document est de **décrire les méthodes, procédés et technologies** sélectionnées pour faire face aux contraintes de réalisation du projet. Il s’agit d’obtenir l’accord du client **avant le début de la phase de réalisation**.

Toute modification en cours de réalisation, devra au préalable être approuvée par les parties concernées et tracées dans ce même document.

Les éléments présents dans le dossier, permettrons de comprendre la **nature, l’organisation et les relations entre les composants** de la solution informatique précédemment décrite dans le dossier de conception fonctionnel.

Un rappel du contexte, des enjeux et des objectifs issus du besoin client, OC Pizza, sera rédigé.

Puis, par le biais de la méthodologie **UML** seront présentés :

* *Le diagramme de représentation du modèle physique de données (MPD).*
* *Le diagramme de composants.*
* *Le diagramme de déploiement.*

*Ces schémas seront détaillés avec précision.*

## Références

Le présent *document, «***PDOCPizza\_02\_technique.pdf »** *est disponible à l’adresse suivante :*

* [*https://github.com/ROL-1/P9-Documentez*](https://github.com/ROL-1/P6-Solution-Technique)

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

1. **PDOCPizza\_01\_fonctionnel.pdf – [version 3.1] :** Dossier de conception fonctionnel
2. **PDOCPizza\_03\_exploitation.pdf – [version 1.1] :** Dossier d’exploitation
3. **PDOCPizza\_04\_PV\_livraison.pdf :** PV de livraison

## Besoin du client

### Contexte

Le groupe « **OC Pizza** » (client) a contacté notre entreprise, « **IT Consulting & Development** » (prestataire) pour signifier son souhait de faire développerun système de gestion informatique pour l’ensemble des restaurants du groupe. Il s’agit de restaurants spécialisés dans les pizzas livrées ou à emporter.

La motivation de cette démarche est liée à la croissance du groupe. Possédant déjà 5 points de vente, il prévoit d’en ouvrir au moins 3 de plus d’ici **6 mois**.

Le système informatique actuel ne correspond plus aux besoins du groupe car il ne permet pas une gestion **centralisée** de toutes les pizzerias.

Les responsables ont besoin de **suivre** facilement ce qui se passe dans les points de vente.

Les livreurs ne peuvent pas indiquer « en live » que la livraison est effectuée.

### Enjeux et Objectifs

Le système informatique devra répondre aux besoins suivants :

**Centraliser** :

* Comptes clients uniques pour tous les types d’achat (en ligne – par téléphone – sur place).
* Interface d’achat reliée en ligne ou à la livraison.

Offrir de la **visibilité** :

* Site internet dédié aux clients (passer commande).

**Organiser**, **homogénéiser** :

* Gestion des commandes (réception – préparation – livraison).
* Relier les indicateurs de stocks à l’interface d’achat (savoir si une pizza est encore réalisable).
* Un client peut modifier ou annuler sa commande (avant la « préparation »)
* Proposer un aide-mémoire aux pizzaïolos indiquant la recette de chaque pizza.
* Définir un cycle de vie des commandes.
* Gestion administrative (comptes utilisateurs, indicateurs).

Offrir un **suivi** :

* En temps réel du statut des commandes.
* En temps réel du stock d’ingrédients.

Enfin, il sera impératif de proposer une solution pouvant être concrétisée dans un délai raisonnable pour permettre une prise en main par les responsables **avant** l’ouverture des nouveaux points de ventes d’ici **6 mois**.

# Architecture Technique

## Application

### Back-end :

* CMS/Langage :

Utiliser un CMS offre une simplicité de mise en place mais des contraintes de conception. En séparant le fond de la forme cette solution limite les réponses envisageables. Notre entreprise étant sollicitée pour développer intégralement la solution, il est préférable de partir sur un langage de programmation qui nous est connu et qui nous permettra de concrétiser intégralement les réponses souhaitées. Nous proposons donc mettre à profit notre expérience du langage **Python 3** pour avoir une maitrise totale du projet et plus de flexibilité.

* Framework :

**Django** s’impose de par sa conception optimisée pour le langage python. C’est un Framework très populaire pour la création de sites web (utilisé par Instagram ou Pinterest, par exemple). Il permettra un développement rapide d’un site dynamique et pourra nous permettre de générer une interface d’administration. En comparaison avec Flask ou d’autres, il est le plus exhaustif, permet d’automatiser un bon nombre de tâches et de disposer d’une très grande communauté.

* Base de données (BDD) :

Nous pourrons utiliser un logiciel qui permet de gérer des bases de données (comptes utilisateurs, etc…) et donc de gérer de grosses quantités d'informations par le langage SQL, avec **MySQL**. Opensource, il est une fois encore un des systèmes de gestion parmi les plus connus, (davantage que PostgreSQL, par exemple) et utilisés (par Youtube ou Wordpress, par exemple).

* Hébergeur :

Il sera également nécessaire de choisir un hébergeur. Les hébergeurs gratuits pouvant engendrer des problèmes de maintenance, il sera nécessaire d’estimer le budget allouable pour obtenir un service de qualité satisfaisante pour ce site visant à accueillir un publique large. Les solutions sont nombreuses (OVH, IONOS, LIKUID.COM, par exemple). La même réflexion pourra également se porter sur le nom de domaine.

### Front-end :

Nous proposons de mettre en place une **arborescence hiérarchique** du fait que les liens entre les différents modules seront plus intuitifs et que cette structure s’accorde parfaitement avec ce projet de type commercial.

De plus les usages des clients pouvant être sur différents **devices** nous mettrons en place une solution « **responsive** » : le site d’adaptera et se redimensionnera automatiquement selon les contraintes de résolution des navigateurs (et supports : smartphone/tablette/ordinateur).

Les technologies retenues seront les standards du web, à savoir : **HTML 5,** **CSS 3** et **JavaScript**.

## Modèle physique de données

### Synthèse

Inspiré du diagramme de classes, le modèle physique de données (**MPD**) présente les **tables** utilisées pour **stocker les données** du système informatique. Il s’agit de la représentation de la **base de données (BDD)**. Le MPD sert à implanter la base de données (grâce au reverse engineering) et lui est donc fidèle. Les tables sont constituées d’**attributs**, correspondant aux colonnes d’un tableau, et d’**instances**, correspondant aux lignes. Les relations entre les tables sont identifiées par la notation « patte d’oie ».

Des **différences** apparaissent par rapport au diagramme de classes lorsque l’expression fonctionnelle ne correspond pas aux **besoins techniques** de la création de cette architecture de stockage de données.

Notamment, le panier (« ShoppingCart ») n’apparait pas directement car sa gestion est déléguée à la partie site web (type cookies/API JavaScript). On notera également la fusion des classes « Customer » et « Employee » dans une unique table « User », la différentiation pouvant se faire par l’attribution de rôle spécifiques.

Chaque table sera détaillée dans la partie 4.3.

### Schéma du modèle physique de données

*Schéma visible page suivante.*

*Réalisé avec MySQL Workbench.*

*Lien vers la documentation :* [*Documentation MySQL Workbench*](https://dev.mysql.com/doc/workbench/en/)





### Détail des tables

Chaque table est détaillée, ci-dessous, dans deux tableaux :

* Colonne : décrit les types de chaque colonne. La colonne « PK » indique si l’attribut est une clef primaire. La colonne « NN » indique que la valeur NULL n’est pas autorisée. La colonne « AI » précise que la colonne est auto-incrémentée. La mention « FK » précise qu’il s’agit d’une clef étrangère (faisant référence à une autre table).
* Relation : indique la multiplicité (ou cardinalité) de la relation entre la table et celle indiquée dans la colonne « Table », décrit la relation en précisant éventuellement sa nature (jonction).
* Les tables de jonction sont détaillées avec les tables auxquelles elles sont liées.

#### Table « Adress »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | adress\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | number | *VARCHAR : 5 charactères autorisés.* |  | ✓ |  | | street | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | | zip | *VARCHAR : 10 charactères autorisés.* |  | ✓ |  | | city | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | | comment | *VARCHAR : 255 charactères autorisés.* |  |  |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 0..1 – 1 | *Order* | *Une adresse est liée à aucune ou plusieurs commandes.*  *Une commande est liée à une unique adresse.* |
| 0..1 – 1 | *Restaurant* | *Une adresse est liée à aucun ou plusieurs restaurants.*  *Un restaurant est lié à une unique adresse postale.* |
| 1 - 1 | *User* | *Une adresse de facturation est liée à un unique utilisateur.*  *Un utilisateur est lié à une unique adresse de facturation.* |

* Logique d’utilisation : Avant de créer son panier, le client doit choisir son restaurant. La vérification de l’adresse de livraison se fera lorsque le restaurant devra accepter la commande.

#### Table « Bill »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | bill\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | Order\_order\_id | *INT : référence de la table.* | ✓ | ✓ |  | | date | *DATETIME : date de création.* |  | ✓ |  | | total | *DECIMAL : montant total.* |  | ✓ |  | | Payement\_Method\_  Payement\_method\_id | *FK : renvoie à la table « Payement\_Method* *» et son attribut « Payement\_method\_id ». Pour identifier le moyen de paiement.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 1 – 0..1 | *Order* | *Une facture est liée à une unique commande.*  *Une commande est liée à aucune ou une facture.* |
| 1 - 1 | *Payement\_Method* | *Une facture est liée à un unique moyen de paiement.*  *Un moyen de paiement est lié à aucune ou une facture.* |

* Logique d’utilisation : Chaque facture doit impérativement avoir un numéro unique non réutilisable. La facture est créée au moment du paiement. Eventuellement cette tâche est prise en charge par le service de livraison, une facture est tout de même éditée par le restaurant.

#### Table « Category »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | category\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | name | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | | Category\_  category\_id | *FK : renvoie à la table « Category » et son attribut «*category\_id *».* |  |  |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 0..1 - 1..\* | *Category* | *Une catégorie est liée a aucune ou plusieurs sous-catégories.*  *Une sous-catégorie est liée à une unique catégorie.* |
| Junction | *Ingredient\_has\_Category* | *Ingredient\_has\_Category est une table de jonction entre Ingredient et Category*  *Une catégorie est liée à aucun ou plusieurs ingrédients.*  *Un ingrédient est lié à une ou plusieurs catégories.* |
| Junction | *Product\_has\_Category* | *Product\_has\_Category est une table de jonction entre Product et Category*  *Une catégorie est liée à aucun ou plusieurs produits.*  *Un produit est lié à aucune ou plusieurs catégories.* |

* Logique d’utilisation : Les catégories « produits » et « ingrédients » permettent d’identifier les objets pouvant apparaitre ou non dans le catalogue produit.

#### Table « Ingredient »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | code | *INT : référence de la table.* | ✓ | ✓ | ✓ | | name | *INT : référence de la table.* |  | ✓ |  | | priceExclTax | *DECIMAL : prix Hors taxe.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 0..\* - 1 | *FormOrderSupplier* | *Un ingrédient est lié à aucun ou plusieurs bons de commande.*  *Une ligne de bon de commande est lié à un ingrédient.* |
| Junction | *Ingredient\_has\_Recipe* | *Ingredient\_has\_Recipe est une table de jonction entre Ingredient et Recipe.*  *Un ingrédient est lié à aucune ou plusieurs recettes.*  *Une recette est liée à un ou plusieurs ingrédients.*  *La table précise la quantité (en grammes) par ingrédient.* |
| Junction | *Ingredient\_has\_Category* | *Ingredient\_has\_Category est une table de jonction entre Ingredient et Category.*  *Un produit est lié à aucune ou plusieurs catégories.*  *Une catégorie est liée à aucun ou plusieurs produits.* |
| Junction | *Restaurant\_has\_Ingredient* | *Restaurant\_has\_Ingredient est une table de jonction entre Restaurant et Ingredient.*  *Un restaurant est lié à aucun ou plusieurs ingrédients.*  *Un ingrédient est lié à un ou plusieurs restaurants.*  *La table précise la quantité (en grammes) par ingrédient.* |

#### Table « FormOrderSupplier »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | number\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | date | *DATETIME : date de création du bon.* |  | ✓ |  | | amount(grams) | *DECIMAL(10,2) : quantité (en grammes) d’ingrédient à commander.* |  | ✓ |  | | Restaurant\_siret | *FK : renvoie à la table «*Restaurant *» et son attribut «*siret *». Pour identifier le restaurant à l’origine du bon de commande.* |  | ✓ |  | | Ingredient\_code | *FK : renvoie à la table « User » et son attribut « user\_id ». Pour identifier l’utilisateur à l’origine de l’avis.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 1 – 0..\* | *Ingredient* | *Un bon de commande est lié à un unique ingrédient.*  *Un ingrédient est lié à aucun ou plusieurs bons de commande.* |
| 1 – 0..\* | *Restaurant* | *Un bon de commande est lié à un unique restaurant.*  *Un restaurant est lié à aucun ou plusieurs bons de commande.* |

* Logique d’utilisation : Le restaurant transmet ce bon au système « fournisseurs » (logiciel).

#### Table « Opinion »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | opinion\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | text | *TEXT : champ texte de longueur variable.* |  | ✓ |  | | date | *DATETIME : date de création de l’avis.* |  | ✓ |  | | User\_user\_id | *FK : renvoie à la table « User » et son attribut « user\_id ». Pour identifier l’utilisateur à l’origine de l’avis.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 1 – 0..\* | *User* | *Un avis est lié à un unique utilisateur (client).*  *Un utilisateur (client) est lié à aucun ou plusieurs avis.* |

* Logique d’utilisation : Dans cette proposition, les avis ne sont pas liés à un restaurant ou une commande spécifique.

#### Table « Order »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | order\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | validate | *TINYINT : Type booléen (statut).* |  | ✓ |  | | paid | *TINYINT : Type booléen (statut).* |  | ✓ |  | | shipped | *TINYINT : Type booléen (statut).* |  | ✓ |  | | delivred | *TINYINT : Type booléen (statut).* |  | ✓ |  | | date\_created | *DATETIME : date de création.* |  | ✓ |  | | date\_paid | *DATETIME : date de paiement.* |  |  |  | | date\_shipped | *DATETIME : date de prise en charge par le service de livraison.* |  |  |  | | date\_delivred | *DATETIME : date de livraison.* |  |  |  | | Restaurant\_  siret | *FK : renvoie à la table « Restaurant » et son attribut « siret ». Pour identifier le restaurant sollicité.* |  | ✓ |  | | User\_  user\_id | *FK : renvoie à la table « User » et son attribut « user\_id ». Pour identifier l’utilisateur à l’origine de la commande.* |  | ✓ |  | | Status\_  status\_id | *FK : renvoie à la table « Status » et son attribut « status\_id ». Pour identifier le statut de la commande.* |  | ✓ |  | | Adress\_  adress\_id | *FK : renvoie à la table « Adress » et son attribut « adress\_id ». Pour identifier l’adresse de livraison.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 1 – 0..\* | *Adress* | *Une commande est liée à une unique adresse.*  *Une adresse est liée à aucune ou plusieurs commandes.* |
| 0..1 – 1 | *Bill* | *Une commande est liée à aucune ou une unique facture.*  *Une facture est liée à une unique commande.* |
| Junction | *Order\_has\_Product* | *Order\_has\_Product est une table de jonction entre Order et Product.*  *Une commande est liée à aucun ou plusieurs produits.*  *Un produit est lié à aucune ou plusieurs commandes.*  *La table précise la quantité, le prix hors taxes et l’ajout d’un commentaire, par produit.* |
| 1 – 0..\* | *Restaurant* | *Une commande est liée à un unique restaurant.*  *Un restaurant est lié à aucune ou une commande.* |
| 1 – 1 | *Status* | *Une commande est liée à un unique statut.*  *Un statut est lié à aucune ou plusieurs commandes.* |
| 1 – 0..\* | *User* | *Une commande est liée à un unique utilisateur.*  *Un utilisateur est lié à aucune ou plusieurs commandes.* |

* Logique d’utilisation : La gestion du panier est dédié au site web.

#### Table « Payement\_Method »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | payement\_method\_id | *INT : référence de la table.* | ✓ | ✓ |  | | title | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 0..\* - 1 | *Bill* | *Un moyen de paiement est lié à aucune ou une facture.*  *Une facture est liée à un unique moyen de paiement.* |

#### Table « Privilege »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | privilege\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | title | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| Junction | *Role\_has\_Privilege* | *Role\_has\_Privilege est une table de jonction entre Role et Privilege.*  *Un role est lié à aucun ou plusieurs utilisateurs.*  *Un utilisateur est lié à au moins un ou plusieurs utilisateurs.* |
| Junction | *Role\_has\_User* | *Role\_has\_User est une table de jonction entre Role et User.*  *Un utilisateur est lié à un ou plusieurs rôles.*  *Un rôle est lié à aucun ou plusieurs utilisateurs.* |

#### Table « Product »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | code | *INT : référence de la table.* | ✓ | ✓ | ✓ | | name | *INT : référence de la table.* | ✓ | ✓ |  | | description | *DATETIME : date de création.* |  | ✓ |  | | priceExclTax | *DECIMAL : montant total.* |  | ✓ |  | | Recipe\_recipe\_id | *FK : renvoie à la table «*Recipe *» et son attribut «*recipe\_id *».* |  |  |  | | Vat\_vat\_id | *FK : renvoie à la table «*Vat *» et son attribut «*vat\_id *».* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| Junction | *Order\_has\_Product* | *Order\_has\_Product est une table de jonction entre Order et Product.*  *Un produit est lié à aucune ou plusieurs commandes.*  *Une commande est liée à aucun ou plusieurs produits.*  *La table précise la quantité, le prix hors taxes et l’ajout d’un commentaire, par produit.* |
| Junction | *Product\_has\_Category* | *Product\_has\_Category est une table de jonction entre Product et Category.*  *Un produit est lié à aucune ou plusieurs catégories.*  *Une catégorie est liée à aucun ou plusieurs produits.* |
| 0..1 – 1 | *Recipe* | *Un produit est lié à aucune ou une recette.*  *Une recette est liée à un unique produit.* |
| Junction | *Restaurant\_has\_Product* | *Restaurant\_has\_Product est une table de jonction entre Restaurant et Product.*  *Un produit est lié à un ou plusieurs restaurants.*  *Un restaurant est lié à aucun ou plusieurs produits.*  *La table précise la quantité (en grammes) par produit.* |
| 1 – 0..\* | *Vat* | *Un produit est lié à une unique T.V.A.*  *Une T.V.A. est liée à aucun ou plusieurs produits.* |

#### Table « Recipe »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | recipe\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | name | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | | protocol | *TEXT : champ texte de longueur variable.* |  |  |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| Junction | *Ingredient\_has\_Recipe* | *Ingredient\_has\_Recipe est une table de jonction entre Ingredient et Recipe.*  *Un ingrédient est lié à aucune ou plusieurs recettes.*  *Une recette est liée à un ou plusieurs ingrédients.*  *La table précise la quantité (en grammes) par recette.* |
| 1 – 1 | *Product* | *Une recette est liée à un unique produit.*  *Un produit est lié à une unique recette.* |
| Junction | *Recipe\_has\_Recipe* | *Recipe\_has\_Recipe est une table de jonction reliant Recipe a elle-même.*  *Une recette est liée à aucune ou plusieurs recettes.* |

#### Table « Restaurant »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | siret | *BIGINT : le SIRET comporte 11 chiffres.* | ✓ | ✓ |  | | name | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | | url | *VARCHAR : 255 charactères autorisés.* |  | ✓ |  | | email | *VARCHAR : 100 charactères autorisés.* |  | ✓ |  | | phone | *VARCHAR : 20 charactères autorisés.* |  | ✓ |  | | Adress\_  adress\_id | *FK : renvoie à la table « Adress » et son attribut « adress\_id ». Adresse postale.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 1 – 1 | *Adress* | *Un restaurant est lié à une unique adresse postale.*  *Une adresse postale est liée à aucun ou un unique restaurant.* |
| 0..\* - 1 | *FormOrderSupplier* | *Un restaurant est lié à aucun ou plusieurs bon de commande.*  *Un bon de commande est lié à un unique restaurant.* |
| 0..\* - 1 | *Order* | *Un restaurant est lié à aucune ou plusieurs commandes.*  *Une commande est liée à un unique restaurant.* |
| Jonction | *Restaurant\_has\_Ingredient* | *Restaurant\_has\_Ingredient est une table de jonction entre Restaurant et Ingredient.*  *Un restaurant est lié à aucun ou plusieurs ingrédients.*  *Un ingrédient est lié à un ou plusieurs restaurants.*  *La table précise la quantité (en grammes) par ingrédient.* |
| Jonction | *Restaurant\_has\_Product* | *Restaurant\_has\_Product est une table de jonction entre Restaurant et Product.*  *Un restaurant est lié à aucun ou plusieurs produits.*  *Un produit lié à un ou plusieurs restaurants.*  *La table précise la quantité par produit.* |
| Jonction | *User\_has\_Restaurant* | *User\_has\_Restaurant est une table de jonction entre User et Restaurant.*  *Un restaurant est lié à aucun ou plusieurs utilisateurs (employé, manager).*  *Un employé est lié à un unique restaurant.* |

#### Table « Role »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | role\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | title | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| Junction | *Role\_has\_Privilege* | *Role\_has\_Privilege est une table de jonction entre Role et Privilege.*  *Un role est lié à aucun ou plusieurs utilisateurs.*  *Un utilisateur est lié à au moins un ou plusieurs utilisateurs.* |
| Junction | *Role\_has\_User* | *Role\_has\_User est une table de jonction entre Role et User.*  *Un utilisateur est lié à un ou plusieurs rôles.*  *Un rôle est lié à aucun ou plusieurs utilisateurs.* |

#### Table « Status »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | status\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | title | *VARCHAR : 45 charactères autorisés.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
| 0..\* – 1 | *Order* | *Un statut est lié à aucune ou plusieurs commandes.*  *Une commande est lié à un unique statut.* |

#### Table « User »

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Colonne | Type | PK | NN | AI | | user\_id | *INT : référence de la table.* | ✓ | ✓ | ✓ | | first\_name | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | | last\_name | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | | phone | *VARCHAR : 20 charactères autorisés.* |  | ✓ |  | | password | *VARCHAR : 55 charactères autorisés.* |  | ✓ |  | | create\_time | *VARCHAR : 255 charactères autorisés.* |  | ✓ |  | | email | *TIMESTAMP : date de création.* |  |  |  | | Adress\_  Adress\_id | *FK : renvoie à la table « Adress » et son attribut « adress\_id ». Pour la facturation.* |  | ✓ |  | |

|  |  |  |
| --- | --- | --- |
| Relation | Table | Description |
|  | *Adress* | *Un utilisateur est lié à une unique adresse de facturation.*  *Une adresse de facturation est liée à un unique utilisateur.* |
|  | *Opinion* | *Un utilisateur est lié à aucun ou plusieurs avis.*  *Un avis est lié à un unique utilisateur.* |
|  | *Order* | *Un utilisateur est lié à aucune ou plusieurs commandes.*  *Une commande est liée à un unique utilisateur.*  *Le protocole de réalisation d’une commande en cas de prise en charge par un employé (ex : commande passée directement au restaurant) est du ressort de la logique métier (à définir avec le client).* |
| Junction | *Role\_has\_User* | *Role\_has\_User est une table de jonction entre Role et User.*  *Un utilisateur est lié à un ou plusieurs rôles.*  *Un rôle est lié à aucun ou plusieurs utilisateurs.* |
| Junction | *User\_has\_Restaurant* | *User\_has\_Restaurant est une table de jonction entre User et Restaurant.*  *Un restaurant est lié à aucun ou plusieurs utilisateurs (employé, manager).*  *Un employé est lié à un unique restaurant.* |

* Logique d’utilisation : Les clients et employés sont rassemblés dans une unique table « User », la différentiation pouvant se faire par l’attribution de rôle spécifiques. Chaque rôle obtenant des privilèges il sera possible de créer de nouveaux rôles pour répondre à des besoins spécifiques, même ponctuels.

# Architecture de déploiement

## Synthèse

Le **diagramme de déploiement** est une vue « statique**»** des diagrammes **UML**. Il illustre l’utilisation de l’**infrastructure** **physique** par le système et ses acteurs, ainsi que la nature de leurs **relations**. Il identifie les **éléments** **matériels** (serveurs) nécessaires et précise, au cœur des nœuds, par le biais de ses composants, quels **logiciels** seront utilisés.

La **pile logicielle** est la suivante :

|  |  |  |
| --- | --- | --- |
|  | Logiciel | Version |
| Application | Python | 3.8 |
| Framework | Django | 3.1 |
| Serveur d’application (WSGI) | Gunicorn | 20.0 |
| Serveur web | Nginx | 1.19 |
| Serveur base de données | PostgreSql | 13.2 |

*Les versions proposées correspondent à celles envisagée au moment de la rédaction du présent document mais sont sujettes à évoluer.*

## Diagramme de déploiement

*Schéma visible page suivante.*

*Les terminaux (« devices ») présentés sont regroupés dans trois groupes, seuls ceux identifiés par « group » (couleur bleue) font partit du système que nous développons.*



## Détail des terminaux

### « External »

Notre système sera amené à communiquer avec des terminaux externes tels que :

* « **Banking System** » : **API** qui sera l’interface avec le système bancaire pour obtenir des **paiements** ou des autorisations de paiement.
* « **Delivery System** » : qui sera l’interface avec le système de livraison, donc selon notre proposition une entreprise externe. Nous utiliserons une application (et/ou API) avec laquelle nous pourrons lui transmettre et recevoir les informations nécessaires à la **prise en charge des commandes** et à la gestion des **paiements à la livraison**. Elle permettra également de suivre en temps réel la **position des livreurs**.
* « **Supplier System** » : ce terminal désigne un logiciel à part entière auquel les restaurants transmettront simplement leurs besoins en ingrédients par le biais de « bon de commandes ». La **gestion des factures** **« fournisseurs »** et de leurs montant sera effectuée par ce logiciel.

### « Group »

Notre solution proposera donc de mettre en place :

* Un serveur web, **Nginx**: qui sera en charge de recueillir les requêtes **HTTP**(s) transmises par les utilisateurs et de les transmettre à notre serveur d’application.
* Un serveur d’application, **Gunicorn** : qui hébergera les composants logiciels que nous développerons, par le biais de la communication « WSGI » (Web Server Gateway Interface). Il contient la logique métier, cœur de notre programme, rédigé avec le langage **Python** et son framework **Django**.
* Un serveur de base de données, **MySQL**: qui servira à stocker, gérer et extraire les données dans une base de données.
* Le tout sera mis à disposition grâce aux services de « droplets » de **DigitalOcean** (cf. 5.2)

### « Users »

Les usages des utilisateurs pouvant être sur **différents** **terminaux** nous mettrons en place une solution « **responsive** » : le site d’adaptera et se redimensionnera automatiquement selon les contraintes de résolution des **navigateurs** et de leurs supports : smartphone/tablette/ordinateur. Leurs requêtes seront transmises généralement en HTTP ou HTTPS notamment lors de la connexion.

.

# Architecture logicielle

## Principes généraux

Les sources et versions du projet sont gérées par **Git**, les dépendances et le packaging par **pip.** Il s’agit d’un projet développé avec Python et **Django**. Django propose un ORM pour communiquer avec la base de données permettant de s’abstraire de l’écriture SQL. Django créé le projet sous la forme du design pattern « MVT » (Model, View, Template). Les modèles servent à construire la base de données, les vues à définir les requêtes http et leur parcours sur le site et les « templates » structurent les pages en HTML.

### Les couches

L'architecture applicative est la suivante :

* Unecouche **business** : responsable de la logique métier du composant.
* Unecouche **model** : gère l’implémentation du modèle des objets métiers.
* Une couche **interface** : responsable des interfaces utilisateurs (client/employés).

### Diagramme de composants

Le **système informatique** proposé dans le dossier de dossier de conception fonctionnelle est composé de **parties logicielles** et de **données** qui sont décrites dans la partie suivante et illustrées dans le **diagramme de composants**. L’intérêt du découpage par composants est d’identifier des parties relativement indépendantes. Le diagramme de composants illustre également les **relations** entre ces composants au travers de la visualisation des ports et des dépendances. Il permet d’avoir une vue d’ensemble « statique » sur le projet de développement en cours.

Pour guider la lecture une différentiation est proposée par un code couleur selon les usagers concernés. Cette segmentation n’est pas absolue : certaines fonctionnalités peuvent être transverses.

*Schéma visible page suivante.*



#### Détail des composants

##### Package « Administration »

Le package « **Administration** » est destiné principalement à la direction du groupe et aux responsables de point de vente. Il rassemble le composant « **Accounts** » permettant de gérer les comptes utilisateurs, le composant « **Dashboard** » représentant l’outil affichant les indicateurs et le composant « **Receips** » permettant de gérer l’aide-mémoire.

##### Package « Authentification »

Le package « **Authentification** » est le premier à être sollicité lorsqu’un utilisateur souhaite se connecter. Il permet à ce dernier d’obtenir son rôle, donc ses privilèges associés et ainsi de pouvoir accéder à son interface et ses options dédiées.

##### Package « Restaurant Management »

Le package « **Restaurant Management** » est destiné aux employés et responsables de point de vente. Il contient l’interface de gestion des stocks, « **Stock** ». Ce package différentie la gestion du stock de produits, « **Product Inventory** » de celle du stock d’ingrédients, « **Ingredient** **Inventory** », la disponibilité des premiers dépendant des derniers. Ce package contient aussi l’interface de gestion des commandes « **Order Management** ». Il permet aux employés de mettre à jour les stocks, de gérer les commandes (composant « **Order** ») et d’effectuer un encaissement (composant « **Payement** »). Le retrait des produits est fait par le système dès qu’une commande est passée. Ce package communique avec les composants externes pour transmettre un bon de commande d’ingrédients, obtenir une autorisation de paiement et notifier le système de livraison de l’était de la commande. La mise à jour du catalogue produit est faite par le système en fonction de la disponibilité des produits. Une commande devient active lorsque l’information est transmise à partir d’un panier.

##### Package « Shopping »

Le package « **Shopping** » correspond à l’interface d’achat et par extension l’interface **client**. Les clients y ont accès à la possibilité de proposer un avis, par le composant « **Opinion** », de consulter le catalogue produit, par le composant « **Catalog** » et en parcourant celui-ci de remplir un panier par le composant « **ShoppingCart** ». Une fois le panier validé, l’information de création de commande est transmise au package « **Restaurant** **Management** ».

##### Composants « Externes »

N’étant pas pris en charge par notre système, on note la présence de trois **composants externes** :

* « **Supplier System** » : logiciel gérant les bons de commande d’ingrédients.
* « **Bank System** » : Api bancaire pour les autorisations de paiement.
* « **Delivery System** » : représentant l’Api de l’entreprise gérant la livraison.

### Structure des sources

|  |  |
| --- | --- |
| Projet\_OC\_PIZZA/  ├─ manage.py  ├─ pipfile  ├─ README.md  ├─ Pipfile.lock  ├─ *docs/*  ├─ ocpizza/  │ ├─ management/  │ │ └─ commands/  │ │ └─ customcommand.py  │ ├─ *asgi.py*  │├─ settings.py  │ ├─ urls.py  │ └─ wsgi.py  ├─ shopping/  │ ├─ migrations/  │ │ └─ 0001\_initial.py  │ ├─ static/shopping/  │ │ ├─ assets/  │ │ ├─ css/  │ │ └─ js/  │ ├─ templates/  │ │ ├─ shopping/  │ │ └─ 404.html  │ ├─ *admin.py*  │├─ apps.py  │ ├─ forms.py  │ ├─ models.py  │ ├─ urls.py  │ └─ views.py | ├─ restaurant/  │ ├─ migrations/  │ │ └─ 0001\_initial.py  │ ├─ static/restaurant/  │ │ ├─ assets/  │ │ ├─ css/  │ │ └─ js/  │ ├─ templates/  │ │ ├─ restaurant/  │ │ └─ 404.html  │ ├─ *admin.py*  │├─ apps.py  │ ├─ forms.py  │ ├─ models.py  │ ├─ urls.py  │ └─ views.py  ├─ administration/  │ ├─ migrations/  │ │ └─ 0001\_initial.py  │ ├─ static/administration/  │ │ ├─ assets/  │ │ ├─ css/  │ │ └─ js/  │ ├─ templates/  │ │ ├─ administration/  │ │ └─ 404.html  │ ├─ *admin.py*  │├─ apps.py  │ ├─ forms.py  │ ├─ models.py  │ ├─ urls.py  │ └─ views.py  └─ tests/  ├─ *functionnal/*  ├─ integration/  └─ unit/ |

## Application Web

L’application sera mise en ligne sur un serveur web hébergé par **DigitalOcean.** DigitalOcean proposent d’utiliser des « **Droplets** » : ce sont des machines virtuelles basées sur Linux permettant de créer, d’interagir et de monitorer le **serveur** mis à disposition. C’est un service **payant** dont OC PIZZA devra déterminer le plan approprié (avec nos recommandations) selon les usages et le trafic attendu. Ce plan pourra être aisément modifié en cas **d’évolutions** des besoins (des alertes peuvent être crées).

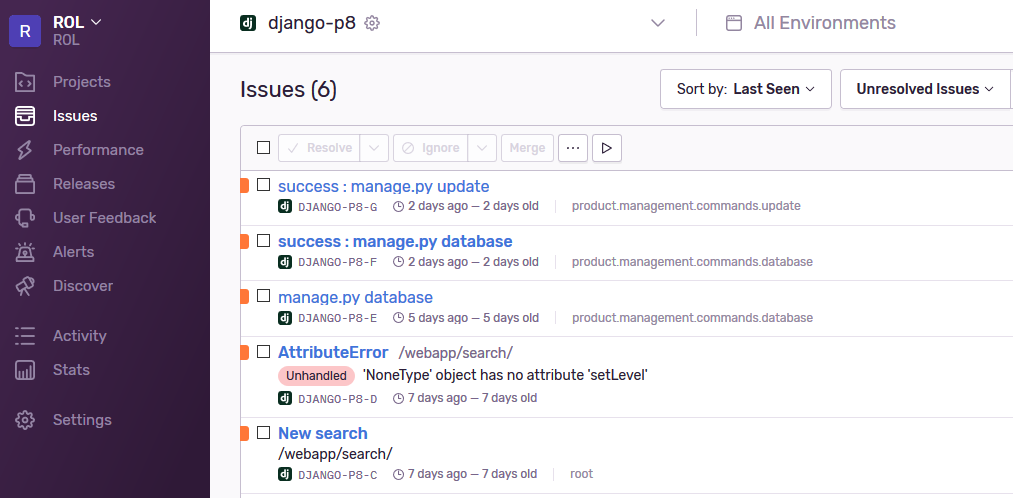
Il est possible de retrouver la **pile logicielle** au paragraphe 4.1.

# Points particuliers

## Gestion des logs

Nous avons créé des **logs** pouvant captés par l’application **Sentry**. Cette plateforme de **monitoring** permet de tracker les erreurs en temps réel. Des logs ont également étés créés au niveau « info » pour permettre de suivre l’exécution de certaines tâches (mise à jour hebdomadaire de la base de données ).

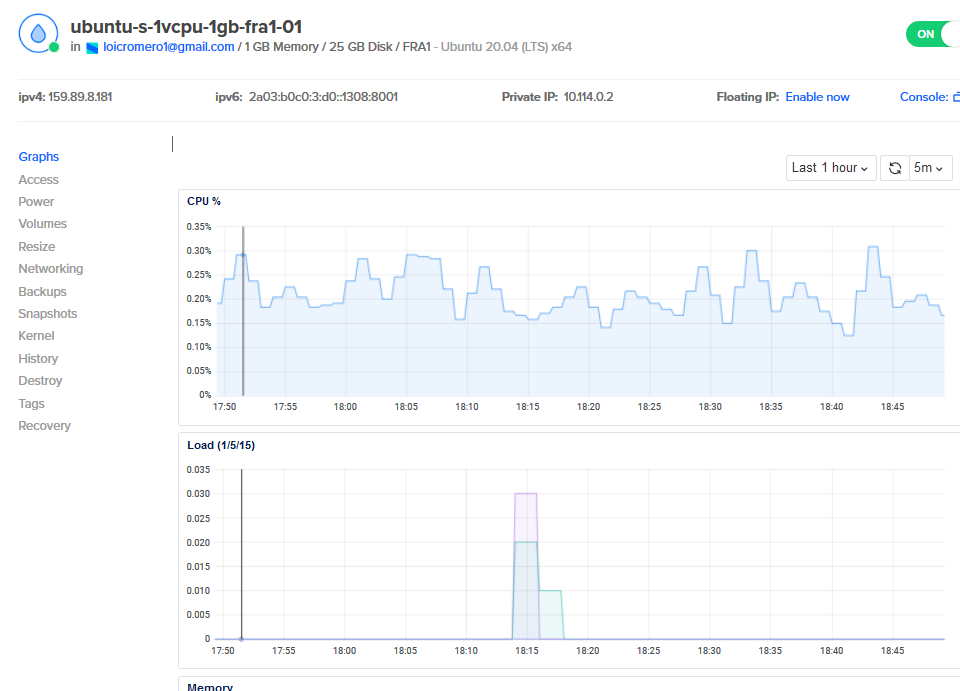
Exemple de l’interface Sentry, ci-dessous :



## Monitoring

**DigitalOcean** (cf. 5.2) intègre des outils de monitoring.

Cela donne accès à des **graphiques** permettant de suivre l’état du serveur : Disk Usage, CPU usage, Bandwidth, Memory, etc…



Dans la rubrique « Monitoring » il est également possible de créer des alertes sur ces mêmes critères, avec « alert policies ». Les alertes seront transmises à l’adresse mail de référence.

## Ressources

Il s’agit des éléments que doit fournir la société OC PIZZA pour que nous puissions développer la solution informatique.

### Charte graphique

Il convient à l’entreprise OC PIZZA de nous faire parvenir une charte graphique permettant de définir l’ambiance souhaitée pour le site. Soit par des consignes précises, soit par des lignes directrices que nous adapterons nous aurons besoin de définir : les couleurs, les icones, les polices, les logos, les éléments responsifs, éventuellement des images ou photographies libres de droits ou dont les droits ont étés acquis.

### Maquettes

Il convient à l’entreprise OC PIZZA de nous faire parvenir des maquettes illustrant au moins à l’état d’ébauche de premières idées de dispositions de mise en pages du site.

### Base de données de développement et finale.

Il convient à l’entreprise OC PIZZA de nous mettre à disposition un jeu de données nous permettant de développer les modèles des bases de données et de les tester (adresses des restaurants, informations sur les employés, etc…). A terme, les données complètes devront nous être fournies avant la mise en production de la solution informatique.

## Environnement de développement

L’environnement de développement choisit par IT Consulting & Development est l’IDE (environnement de développement intégré) **Visual Studio Code** (v.1.54).

Il est recommandé d’utiliser un environnement virtuel à l’aide de **pipenv** ; création et activation :

pipenv install

pipenv shell

Pipenv ne produit pas de requirements.txt, mais des fichiers Pipfiles et Pipfile.lock.

## Procédure de packaging / livraison

Il est possible d’accéder à l’application livrée en se rendant à l’adresse suivante :

*www.ocpizza.rol1.fr*

Les informations nécessaires à l’installation et l’utilisation de l’application sont disponibles dans le **dossier** **d’exploitation** fournis au moment de la livraison.

# Glossaire

|  |  |
| --- | --- |
| AI | « Auto Incremental » : attribut auto-incrémenté d’une table SQL. |
| BDD | Base De Données |
| CI | Intégration continue |
| Composant | Un composant UML est une unité autonome fournissant un service précis. Sa vocation est d’être réutilisable |
| Employés | Désigne généralement à la fois, les employés, les responsables de points de vente et la direction du groupe. |
| FK | « Foreign Key » : clef secondaire d’une table SQL. |
| Framework | Désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d’une partie d'un logiciel. |
| PK | « Primary Key » : clef primaire d’une table SQL. |
| Pizzaïolo | Préparateur de pizza |