OC PIZZA

Dossier de conception technique

Version 1.0

Auteur
Giovanni Gaffé
Maître d'oeuvre

TABLE DES MATIÈRES

1. VERSIONS

Auteur	Date	Description	Version
Giovanni Gaffé	21/04/2020	Description technique de la conception de l'application	1.0

2. INTRODUCTION

2.1. Objet du document

Le présent document constitue le dossier de conception technique de l'application OC Pizza

Objectif du document

Les éléments du présent dossiers découle :

- de l'architecture des composants
- de l'architecture de déploiement
- de l'architecture logicielle

2.2. Références

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

1. **DCF - P6**: spécification Fonctionnelles (Dossier de conception fonctionnel de l'application)

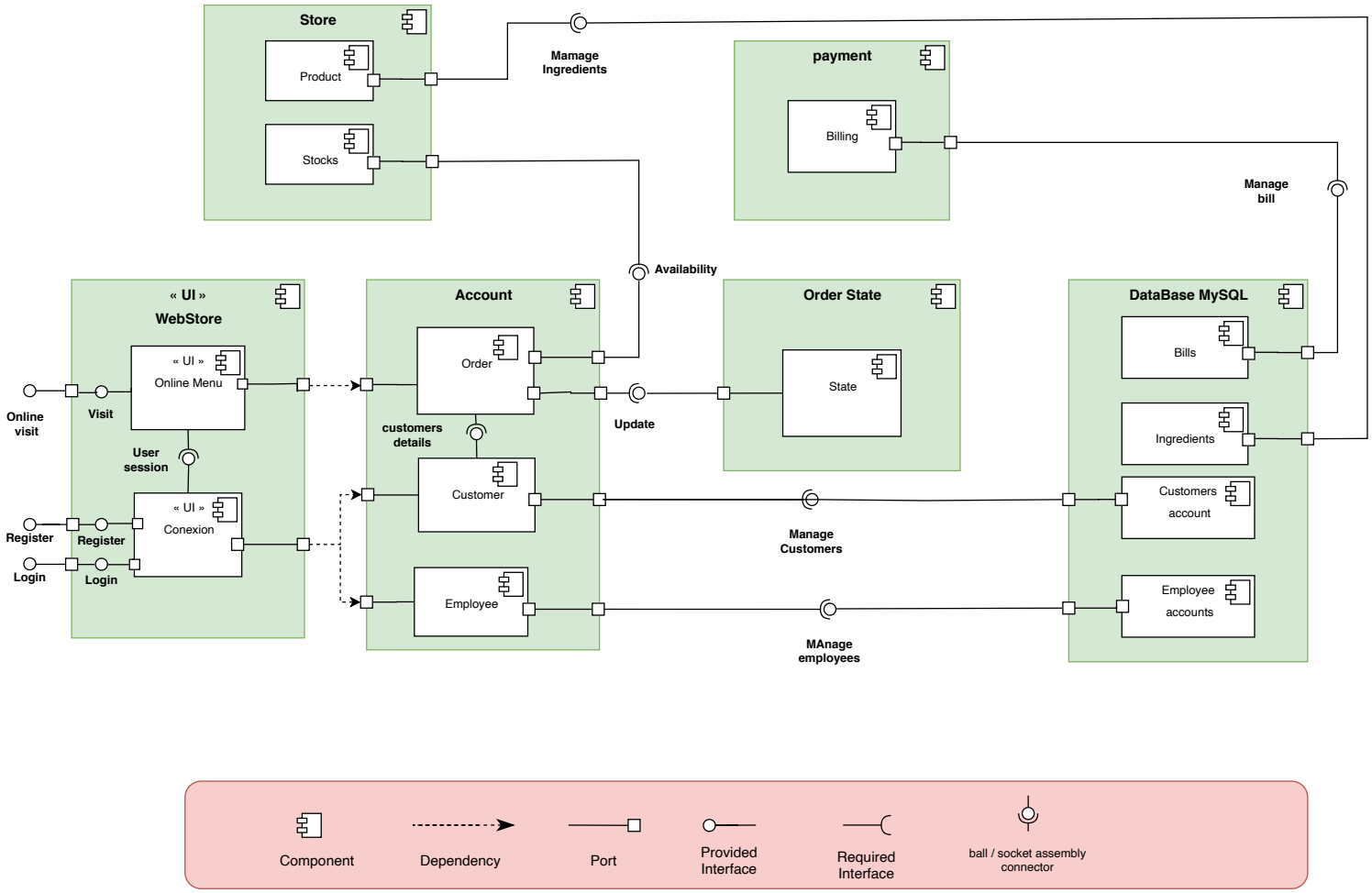
3.Architecture Technique

3.1.Composants généraux

Diagramme UML de composant

Les diagrammes de composant décrivent le système d'un point de vue des éléments logiciels. En insistant et mettant en évidences les dépendances entre composants internet du système.

COMPONENT DIAGRAM



3.1.1.Package WebStore

3.1.1.1.Composant UI OnlineMenu

Interface web Client ainsi que employee. Compatible avec les appareils mobiles

Le point de départ de tout utilisateur qui visitera le site web

3.1.1.2.Composant UI Connexion

Interface permettant la connexion qui donnera lieu aux différentes actions possibles

3.1.2.Package Account

3.1.2.1.Composant Order

Interface permettant de regrouper les commandes faites par l'utilisateur.

3.1.2.2.Composant Employee

Interface permettant de pouvoir gérer les comptes clients et ainsi faire des commandes.

3.1.3.Package Store

3.1.3.1.Composant Products

Interface permettant d'interagir avec la mise à jour d'ingrédients ou de sorties éventuelles.

3.1.3.2.Composant Stocks

Interface permettant de mettre à jour quand une commande est effectuée.

3.1.4.Package Order State

3.1.4.1.Composant State

Interface permettant aux employés de changer l'état d'une commande pour prévenir le client.

3.1.5.Package DataBase My SQL

3.1.5.1.Composant Bills

Interface regroupant les factures dans la base de donnée.

3.1.5.2.Composant Ingredients

Interface regroupant les ingrédients dans la base de donnée.

3.1.5.3.Composant Customers Account

Interface regroupant les Comptes Clients dans la base de donnée.

3.1.5.4.Composant Employee Account

Interface regroupant les Comptes employee dans la base de donnée.

3.1.6.Package Payment

3.1.6.1.Composant Billing

Interface permettant aux bills dans la base de donne d'extraire aisément les factures.

3.2.Application Web

La pile logicielle est la suivante :

- Application Python 3.8 / Django 2.1
- Serveur d'application Gunicorn
- Serveur NGNX

3.2.1.Composant Django

Django est un cadre de développement web open source en Python. Il a pour but de rendre le développement web 2.0 simple et rapide. Pour cette raison, le projet a pour slogan « Le framework pour les perfectionnistes avec des deadlines.Composant PHP

PHP est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamique via un serveur http.

3.2.2.Composants Gunicorn

est un serveur web HTTP WSGI écrit en Python et disponible pour Unix. Son modèle d'exécution est basé sur des sous-processus créés à l'avance, adapté du projet Ruby Unicorn. Le serveur Gunicorn est compatible avec un grand nombre de frameworks web, repose sur une implémentation simple, légère en ressources et relativement rapide

3.2.3.NGNX

NGINX Open Source ou **NGINX** est un logiciel libre de **serveur** Web (ou HTTP) ainsi qu'un proxy inverse écrit par Igor Sysoev, dont le développement a débuté en 2002 pour les besoins d'un site russe à très fort trafic (Rambler).

3.3. Base de données

- MySQL Community Server 8.0.19

3.3.1. MySQL

MySQL est un système de gestion de bases de données relationnelles. Il est distribué sous une double licence GPL et propriétaire.

3.3.1.1.Modele physique de données

3.3.2. MySQL

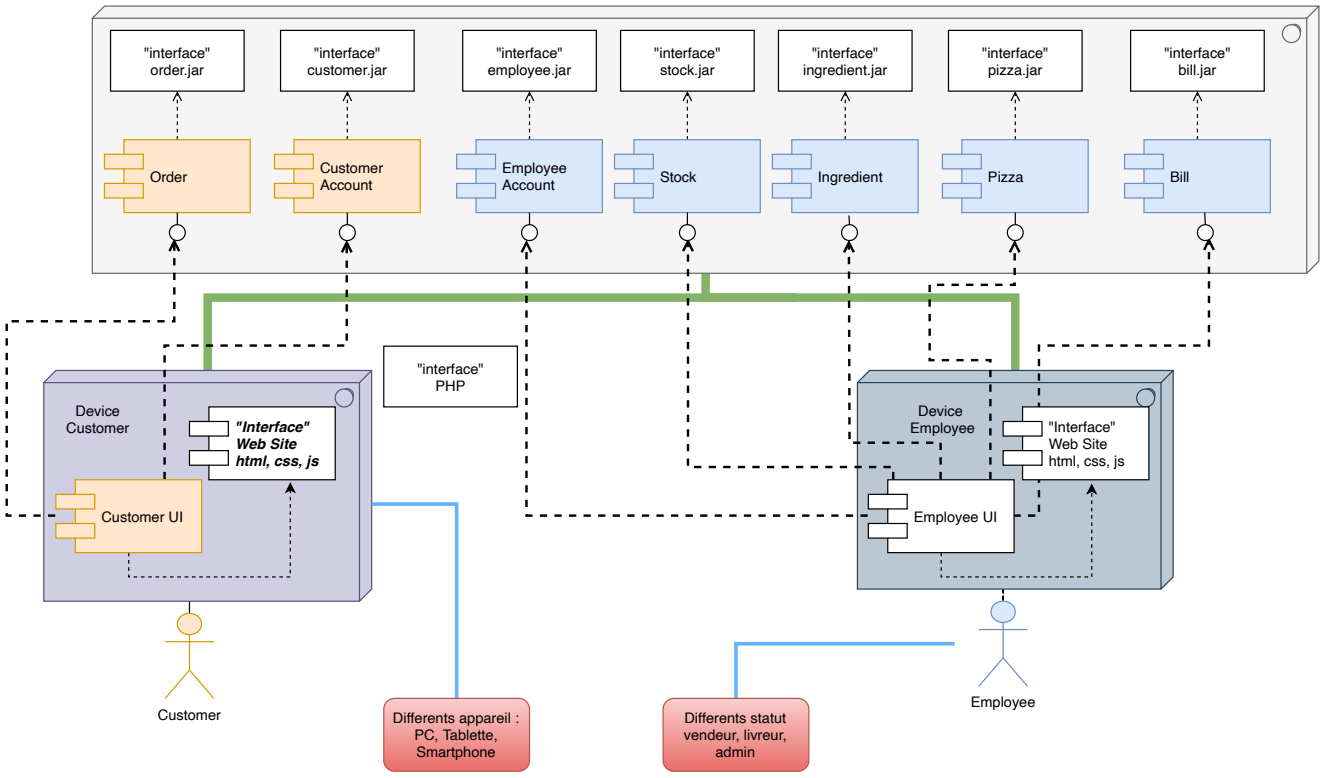
Chrony permet la synchronisation de l'horloge système.

LE serveur MySQL se synchronise avec le serveur Web.

4.Architecture de Déploiement

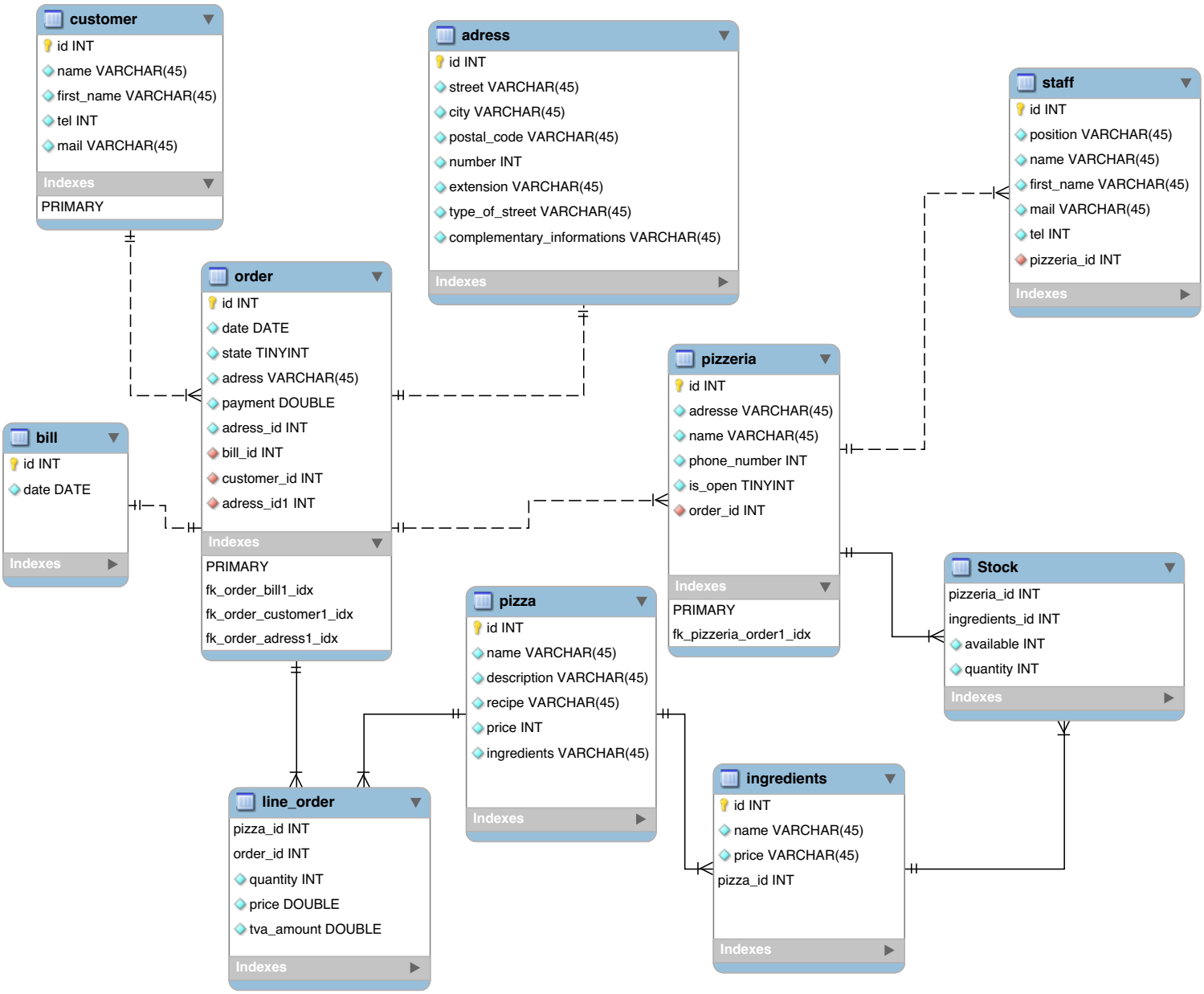
Diagramme UML de déploiement

D'une manière générale un diagramme de déploiement est une vue statique qui sert a représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont repartis ainsi que leurs relation entre eux.



4.1.Serveur Web

- Processeur : Intel Xeon



- Ram : 16 Go
- Disque : 2 x 100 Go en Raid 1
- Système : Linux CentOS 7 64 bits
- Logiciels :
    - Apache
    - PHP
    - Symphony
    - Composer
    - Doctrine
    - Git

4.2.Serveur Base de données

- Processeur : Intel Xeon
- Ram 32 Go
- Disque : 2 x 250 Go en Raid 1
- Système : Linux CentOS 7 64 bits
- Logiciels :
    - MySQL
    - Git

5.Architecture logicielle

5.1.Principes généraux

Les sources et versions du projet sont hébergées sur **Github** et sont gérées avec **Git.** Les dépendances et le packaging par **Pipenv.**

5.1.1. Les couches

L'application utilise le modele MVT (Django) :

- Modèle : Implémentation des objets métiers dans une couche de la base de données
- Template : Préparation du rendu HTML
- Vue : Etabli le lien entre les requêtes web, les modèles et les templates

5.1.2.Les modules

Module **Customer Account** : Gestion des accès utilisateurs

Module **Employee Account** : Gestion de l'administration du site

Module **Pizza** : Gestion des Produits

Module **Ingredients** : Gestion des ventes

Module **Order** : Gestion des Commandes

Module **Bill** : Gestion des ventes

Module **Stock** : Gestion des ingredients

Module **Statistics** : Gestion des ventes

5.1.2. Structure des sources

```
ocpizza-customer/
├── bin
│   └── console
│
├── composer.json
├── composer.lock
├── config
│   ├── bootstrap.php
│   ├── bundles.php
│   ├── packages
│   ├── routes.yaml
│   └── services.yaml
├── controller
├── model
├── public
│   └── index.php
├── src
│   ├── Controller
│   └── Kernel.php
├── symphony.lock
├── var
│   ├── cache
│   └── log
├── vendor
│   ├── autoload.php
│   ├── composer
├──  │   ├── psr
│   └── symfony
└── view
```

La structuration des répertoires du projet suit la logique suivante :

- les répertoires sources sont créés de façon a respecter la philosophie Symphonie 5
- Le design pattern MVC est utilise pour l'organisation du code

/bin : contient les exécutables.

/config : contient la configuration du site.

/public : contient les fichiers destinés aux visiteurs (images, fichiers CSS et Javascript, ...) Il contient également le frontal (index.php).

/src : contient le code source.

/var : contient les logs, le cache et d'autres fichiers nécessaires au bon fonctionnement de l'application.

/vendor : contient les bibliothèques externes (inclus Symphony).

/controller : contient le code qui gère la logique de l'application.

/model : contient le code qui gère l'accès aux données.
/view : contient le code qui gère l'affichage des pages web.

6.Points particuliers

6.1.Gestion des logs

6.1.1.Application web

Localisation des logs : /var/www/ocpizza.fr/

6.1.1.1 - error.log

Contient les erreurs qui surviennent lors des traitements des requêtes.

6.1.1.2 - request.log

Contient les requêtes HTTP destinées au serveur.

6.1.2 - Base de données

Localisation des logs : /var/log/

6.1.2.1 - mysql.log

Contient les évènements et les erreurs MySQL.

6.2.Fichiers de configuration

6.2.1 - Application web

6.2.1.1 - .env

Définit les variables d'environnement de l'application.

6.2.1.1.Datasources

Serveur MySQL : ocpizzadb
Base de données : ocpizza
Encodage : utf8

6.3.Procédure de packaging / livraison

L'ensemble des livrables sont hébergés sur Github.

Les fichiers de configuration et de traitements batch des serveurs seront téléchargeable sous la forme de fichier zip.

- **Serveur web** : webserver-config
- **Serveur de base de données** : dbserver-config
- Pour le site Web, l'installation sera effectuée via des scripts qui effectueront le téléchargement des projets Web sur Github.
- **Site Web clients** : Projet website-customer
- Site Web employés (Point de vente et direction) : Projet website-employee
- **Site Web livreurs** : Projet website-deliverer

7. GLOSSAIRE

Dépendances	Bibliothèque nécessaire au fonctionnement de l'application.
MVT	Représente une architecture orientée autour de trois pôles (le **modèle** , la **vue** et le **template**). Elle s'inspire de l'architecture, très répandu dans les framework web, MVC (un modèle (**Model**) contient les données à afficher, une vue (**View**) contient la présentation de l'interface graphique, un contrôleur (**Controller**) contient la logique concernant les actions effectuer par l'utilisateur).
ORM	Object Relationnel Mapping est un type de programme informatique qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de donnée orientée objet.
Pipenv	Est un outil de packaging pour python qui résout certains problèmes courant associés au flux de travail typique utilisant pop, virtualenv et le bon vieux requirements.txt. En plus de résoudre certains problèmes courants, il consolide et simplifie le processus de développement en un outil de ligne de commande uniquement.