|  |
| --- |
| **OC PIZZA**  **Documentez votre système de gestion de pizzeria**  Dossier de conception technique  Version 1.0 |
| **Auteur**  AUBRUN Eric  *Analalyste-programmeur* |

Table des matières

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

3.1.1 - Composant Authentification 6

3.1.2 - Composant Gestion du stock 6

3.1.3 - Composant Cycle de vie de commande 7

3.1.4 - Les API 7

3.1.4.1 - L’API Géolocalisation 7

3.1.4.2 - L’API SMS 7

3.1.4.3 - L’API Paiement en ligne 7

3.1.4.4 - L’API Mail 7

3.2 - Application XXX... 7

4 - Architecture de Déploiement 8

4.1 - Serveur de Base de données 8

4.2 - Serveur XXX 8

5 - Architecture logicielle 9

5.1 - Principes généraux 9

5.1.1 - Les couches 9

5.1.2 - Les modules 9

5.1.3 - Structure des sources 9

5.2 - Application Web 10

5.3 - Application Xxx 10

6 - Points particuliers 11

6.1 - Gestion des logs 11

6.2 - Fichiers de configuration 11

6.2.1 - Application web 11

6.2.1.1 - Datasources 11

6.2.1.2 - Fichier xxx.yyy 11

6.2.2 - Application Xxx 11

6.3 - Ressources 11

6.4 - Environnement de développement 11

6.5 - Procédure de packaging / livraison 11

6.6 - XXX 11

7 - Glossaire 12

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Eric AUBRUN | 01/05/2021 | Création du document | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## 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 des documents suivants :

* Mise en place d’un nouveau système informatique pour l’ensemble des pizzerias du groupe OC Pizza
* Concevez la solution technique d’un système de gestion de pizzeria

## Références

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

1. **PDOCPizza\_02\_fonctionnelle.pdf – 1.0**: Le Dossier de conception technique de l'application
2. **PDOCPizza\_03\_exploitation.pdf – 1.0** : Le Dossier d'exploitation de l’application
3. **PDOCPizza\_04\_livraison.pdf – 1.0 :** Le PV de livraison finale

# Architecture Technique

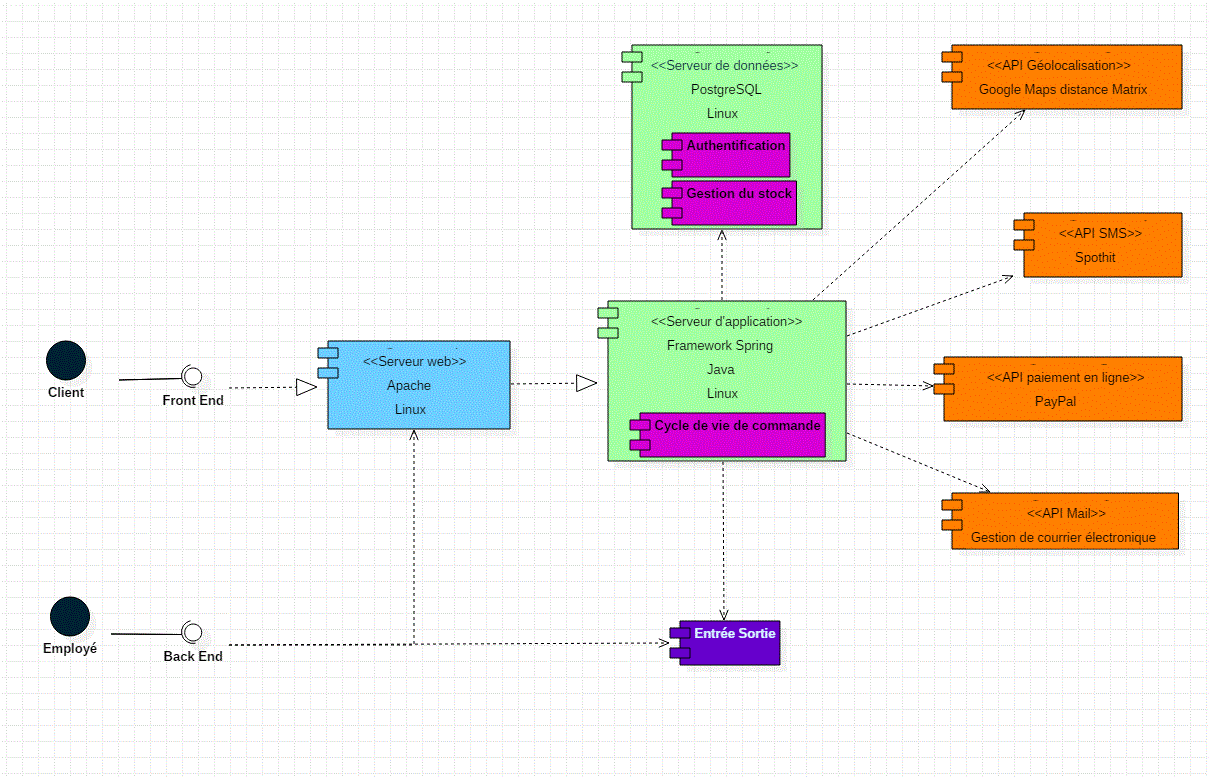
Nous proposons d’architecturer le produit en deux applications :

* Une application et son serveur traitant spécifiquement le front-end :« *Nous faisons le choix de développer le front-end en ayant recours au Framework Angular[[1]](#footnote-1). Cet outil sera accompagné et complété de Framework CSS, de librairies de composants Angular ainsi que de librairies JavaScript facilitant le développement en général*. »[[2]](#footnote-2)
* Une seconde application installée sur un autre serveur, Angular pilotant la communication entre les deux applications à partir de requêtes HTTP : « *Nous trouverons, côté back-end, une base de données contenant les informations relatives aux utilisateurs et aux produits proposés à la vente par OC Pizza. Nous utiliserons une API d’authentification afin de sécuriser les informations d’accès. Une API/web services permettra de gérer les paiements bancaires et enfin, dans le but d’améliorer les performances de l’application, nous pourrons exécuter côté serveur, Angular Universal.*»[[3]](#footnote-3)

## Application Web

La pile logicielle est la suivante :

* Pour la partie back-end :
* PostgreSQL installé sur un serveur de données sous Linux, gérant l’authentification ainsi que le stock de produits et ingrédients, nécessaires à la confection des pizzas.
* Le Framework Spring installé sur un serveur d’application tournant sous Linux, développé en Java, gérant plus particulièrement les différentes phases du cycle de vie d’une commande.
* Pour la partie front-end :
* Le Framework Angular installé sur un serveur Web Apache, lui aussi sous Linux, accompagné d’un Framework CSS comme Bootstrap, permettant notamment d’obtenir un effet « Responsive Web Design »[[4]](#footnote-4), des librairies de composants Angular comme Ng-Bootstrap, en mesure par exemple de construire des fenêtres modales, des librairies générant des graphiques de type « camemberts » comme Chart.Js ou encore Lodash, une bibliothèque Javascript contenant des fonctions permettant par exemple de tester et manipuler des valeurs, créer des fonctions composites ou utiles pour des tableaux, objets ou chaînes de texte.



**DIAGRAMME 1 : DIAGRAMME DE COMPOSANTS**

### Composant Authentification

Description et rôle/objectif :

### Composant Gestion du stock

### Composant Cycle de vie de commande

### Les API

#### L’API Géolocalisation

#### L’API SMS

#### L’API Paiement en ligne

#### L’API Mail

## Application XXX...

# Architecture de Déploiement

Diagramme UML de déploiement

Explication / commentaires si nécessaires...

## Serveur de Base de données

Description

Caractéristiques techniques (ex: Serveur Linux Debian Jessie + PostgreSQL 9.6…)

Informations importantes / points particuliers

## Serveur XXX

...

# Architecture logicielle

## Principes généraux

Les sources et versions du projet sont gérées par **Git**, les dépendances et le packaging par **Apache Maven / Grunt / ...**.

...

### Les couches

L'architecture applicative est la suivante :

* unecouche **business** : responsable de la logique métier du composant
* unecouche **model** : implémentation du modèle des objets métiers
* …
  + …
* ...

### Les modules

Ex: modules Maven dans le cas d’application multi-module...

### Structure des sources

La structuration des répertoires du projet suit la logique suivante :

* les répertoires sources sont crées de façon à respecter la philosophie Maven (à savoir : « convention plutôt que configuration »)

racine  
 ├─ *pom.xml*  
 ├─ <moduleX>  
 │ ├─ *pom.xml*  
 │ └─ src  
 │ ├─ main  
 │ │ ├─ java  
 │ │ └─ resources  
 │ └─ test  
 │ ├─ java  
 │ └─ resources  
 ├─ <moduleY>  
 │ ├─ *pom.xml*  
 │ └─ src  
 │ ├─ main  
 │ │ ├─ java  
 │ │ └─ resources  
 │ └─ test  
 │ ├─ java  
 │ └─ ressources  
 └─ src  
 └─ lib

* ...

## Application Web

…

Si besoin, diagramme UML de composants pour monter les différents modules et leur inter-dépendances

## Application Xxx

…

# Points particuliers

## Gestion des logs

…

## Fichiers de configuration

### Application web

...

#### Datasources

...

#### Fichier xxx.yyy

...

### Application Xxx

...

## Ressources

...

## Environnement de développement

## Procédure de packaging / livraison

## XXX

…

# Glossaire

|  |  |
| --- | --- |
|  |  |
|  |  |

1. Source : https://www.angularchef.com/recette/2/ [↑](#footnote-ref-1)
2. Extrait de : « Mise en place d’un nouveau système informatique pour l’ensemble des pizzerias du groupe OC Pizza », page 39 [↑](#footnote-ref-2)
3. Ibid. page 39 [↑](#footnote-ref-3)
4. Le Responsive Design ou plus précisément le Responsive Web Design (RWD) est une technique de conception d’interface digitale qui fait en sorte que l’affichage d’une quelconque page d’un site s’adapte de façon automatique à la taille de l’écran du terminal qui le lit. Source : [https://www.seo.fr/definition/responsive-design#:~:text=Le%20Responsive%20Design%20ou%20plus,du%20terminal%20qui%20le%20lit](https://www.seo.fr/definition/responsive-design" \l ":~:text=Le%20Responsive%20Design%20ou%20plus,du%20terminal%20qui%20le%20lit). [↑](#footnote-ref-4)