|  |
| --- |
| **OC Pizza**  **Projet 8**  Dossier de conception technique  Version 2 |
| **Auteur**  Alexandre Lardon  *Analyste-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 - Composants généraux 5

3.1.1 - Package Authentification 5

3.1.2 - Package Gestion des commandes 5

3.1.3 - Package Gestion des stocks / Points de vente 5

3.1.4 - Package Gestion des stocks / Centrale d’achat 5

3.1.5 - Package Gestion des employés 6

3.2 - Diagramme de Composants 6

4 - Architecture de Déploiement 7

4.1 - Serveur de Base de données 8

4.2 - Serveur Tomcat 8

5 - Architecture logicielle 9

5.1 - Présentation générale 9

5.2 - Microservice 9

5.2.1 - Les couches d’un microservice 9

5.2.2 - Les ressources 9

5.2.3 - Structure d’un microservice 10

5.3 - Architecture Globale 11

6 - Outils Technique 12

7 - Points particuliers 14

7.1 - Gestion des logs 14

7.2 - Ressources 15

7.3 - Environnement de développement 15

7.4 - Procédure de packaging / livraison 15

8 - Glossaire 16

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Alexandre | 24/10/20 | Création du document | 0.1.0 |
| Alexandre | 24/10/20 | Ajout des 4 premiers chapitres | 0.1.0 |
| Alexandre | 25/10/20 | Document terminé | 1.0 |
|  |  |  |  |

# Introduction

## Objet du document

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

Objectif du document est de justifié le choix des technologies et librairies utilisées, suivant l’architecture sur l’ensemble de l’application mise en place.

## Références

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

1. [**DCF – OC Pizza**](file:///C:\Users\alexa\Desktop\Soutenance%208\Projet%20OC%20Pizza%20-%20Dossier%20de%20conception%20fonctionnelle%20.docx) : Dossier de conception fonctionnelle de l’application
2. [**DE – OC Pizza**](file:///C:\Users\alexa\Desktop\Soutenance%208\Projet%20OC%20Pizza%20-%20Dossier%20d_exploitation.docx) : Dossier d’exploitation de l'application
3. [**PV – OC Pizza**](file:///C:\Users\alexa\Desktop\Soutenance%208\Projet%20OC%20Pizza%20-%20PV%20Livraison.docx) : PV de livraison de l’application

# Architecture Technique

## Composants généraux

Pour la mise en place de l’architecture technique, nous concevons d’abord le diagramme de package que l’on retrouve dans le [dossier de conception fonctionnelle](Projet%20OC%20Pizza%20-%20Dossier%20de%20conception%20fonctionnelle%20.docx#Diagramme_de_Package). Nous retrouvons à l’intérieur les éléments mettant en place l’architecture du projet et de l’application.

### Package Authentification

Le package authentification regroupe tous les composants permettant aux différents utilisateurs de se connecter.

### Package Gestion des commandes

Il représente le système dans son ensemble de la gestion commandes pour les différents points de vente. Il est le cœur de l’application.

Cela va de la consultation de la carte et la prise de commande, jusqu’à la livraison finale des produits.

Il est en liaison avec tous les acteurs de l’entreprise et le client, mais également avec le système bancaire pour le règlement des commandes.

### Package Gestion des stocks / Points de vente

La gestion des stocks sur les points de vente est regroupée dans ce package.

Il gère l’ensemble des stocks sur les points de ventes, cela concerne les produits servant à élaborer les pizzas et autres, de la réception de la commande du fournisseur jusqu’à l’entrée physique et informatique du stock.

Mais également à toutes les sorties de stocks comprenant toutes les boissons et les autres produits vendus.

Chaque point de vente gère son stock, et fait l’inventaire de celui-ci afin de pouvoir réconcilier le stock physique et informatique.

### Package Gestion des stocks / Centrale d’achat

La centrale d’achat regroupe les actions permettant le réapprovisionnement produit des points de vente.

Au départ chaque responsable point de vente passe une commande, elles sont regroupées et accessible au gérant pour qu’il effectue une commande globale, demandant une liaison avec le système bancaire.

La commande est passé auprès d’un fournisseur par le biais d’une liaison.

Une fois la commande livrée, elle est contrôlée puis dispatchée sur les différents points de vente.

### Package Gestion des employés

Le package est accessible uniquement au gérant.

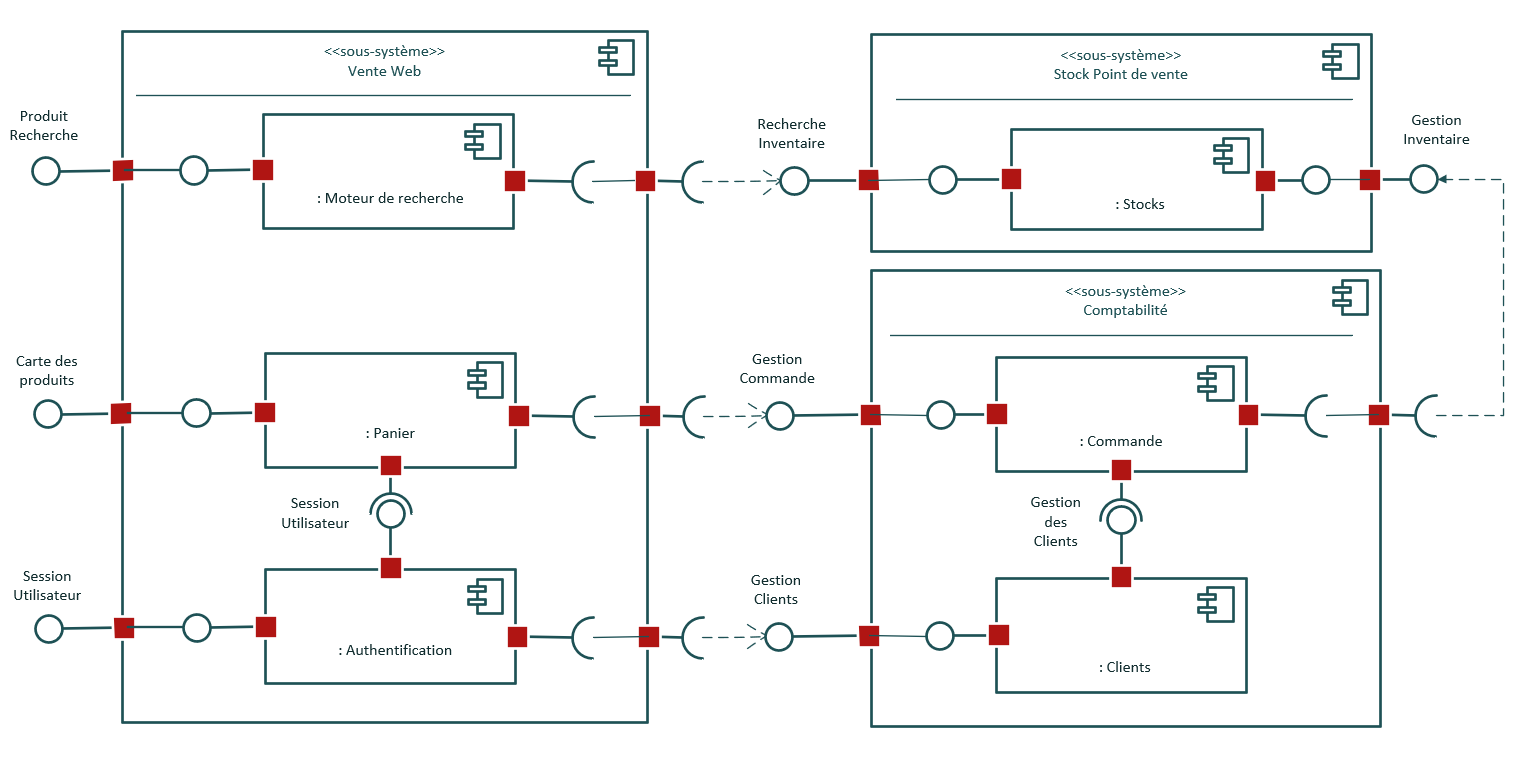
Il permet la gestion complète des employés, l’ajout, la suppression de ceux-ci.

Toutes les informations concernant l’employés, nécessaire à l’entreprise.

## Diagramme de Composants

Le diagramme de composants permet d'obtenir une vue d'ensemble du système logiciel.

Diagramme UML de Composants



Le client effectue une action dans le sous-système ‘vente web’ :

* Il recherche un produit, le composant ‘Moteur de recherche’ demande au ‘Stocks’ par l’intermédiaire de l’interface si le produit recherché par l’utilisateur est encore en stock. Si le stock est différent de nul alors le client peut l’ajouter à son panier si celui-ci est connecté.
* Il consulte la carte des produits et souhaite en ajouter au panier. Dans ce cas nous voyons que l’interface est requise (‘Session Utilisateur’) et que l’utilisateur doit être authentifier pour effectuer cette action.

Une fois authentifié le client accède à la gestion de son compte et des anciennes commandes.

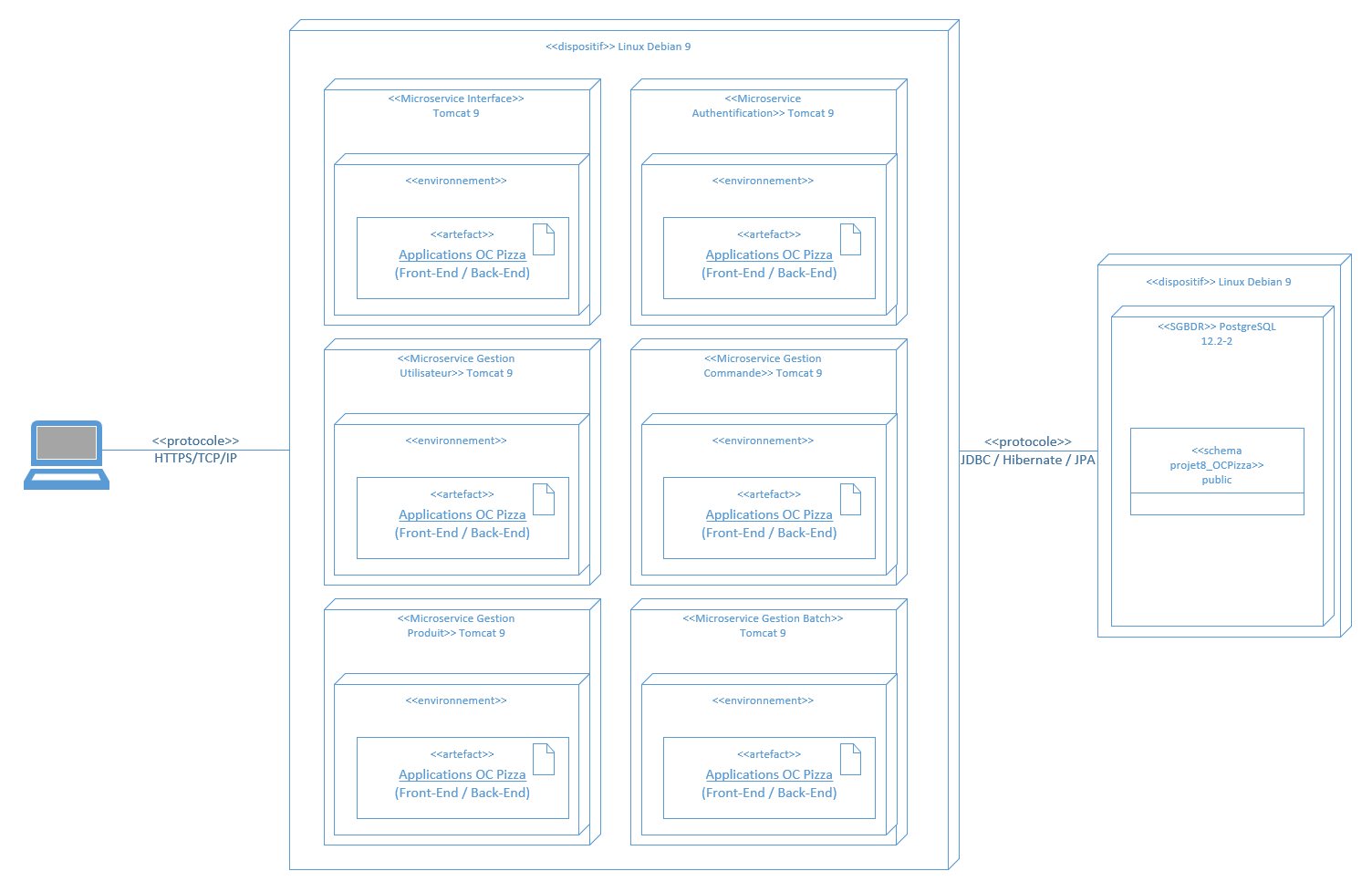
Mais également à sa commande actuelle, et la possibilité de la modifié ou de la validé.

Le composant commande est également en liaison avec le ‘Stocks’ pour pouvoir l’interroger et savoir si la commande du client peut être validé.

# Architecture de Déploiement

Le diagramme de déploiement décrit le déploiement physique des informations générées par le logiciel sur des composants matériels.

Diagramme UML de déploiement



L’IHM échange avec le serveur Linux Debian 9 par l’intermédiaire de requête disposant du protocole Https.

Le serveur contient 6 serveurs Tomcat 9 contentant nos différents microservices.

Nous retrouvons pour chaque microservice un environnement permettant de faire fonctionner l’application aussi bien du côté Back-End que Front-End.

Nous retrouvons également notre SGBDR Postgresql 12.2-2 contenant notre base de données ‘projet8\_OCPizza ‘ se trouvant dans un autre serveur Debian 9.

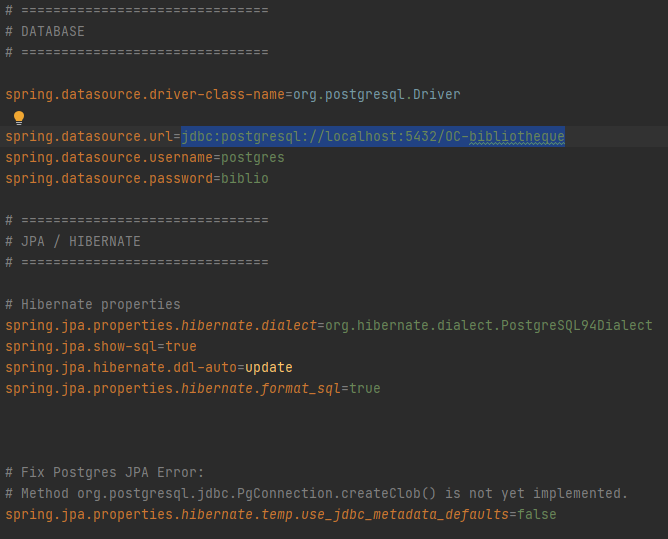
Celle-ci est en relation avec les microservices par l’intermédiaire de l’API JDBC créant une connexion à partir d’un URL. Elle est en relation avec l’ORM Hibernate qui lui génère les requêtes postgresql permettant de pouvoir interroger la base de données.

## Serveur de Base de données

La base de données est hébergée sur un serveur sous Linux Debian 9.

La mise en place de celle-ci est effectuée comme décrite dans ce [lien](https://www.activpart.com/installer-postgresql-11-sur-debian-8-jessie/).

Une fois installé, nous configurons le fichier application.properties, en modifiant l’URL.



## Serveur Tomcat

Les microservices sont déployé par le biais des serveurs Tomcat.

Pour installer le serveur Tomcat sur Linux Debian 9, suivons la procédure décrite sur ce [lien](https://howto.biapy.com/fr/debian-gnu-linux/serveurs/http/installer-tomcat-sur-debian).

# Architecture logicielle

## Présentation générale

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

Nous retrouvons comment avoir accès à Git et comment restaure le projet [ici](Projet%20OC%20Pizza%20-%20Dossier%20d_exploitation.docx#git).

Apache Maven sert à packager le projet et notamment chaque microservice en générant un fichier WAR.

Nous retrouvons à l’intérieure de chaque fichier pom le type du packaging souhaité pour qu’ensuite Maven puisse packager le projet.

Maven sert également à gérer les dépendances du projet. Il récupère les dépendances mise en place dans le fichier pom, ainsi que les dépendances requises à celles dont on a besoin grâce à la gestion de la transitivité des dépendances.

## Microservice

### Les couches d’un microservice

L'architecture applicative est la suivante :

* unecouche **controller** : contient les méthodes permettant de récupérer, d’envoyer, de sauvegarder les données et d’interagir avec les objets pour y avoir accès. Elle fait le lien entre la couche dao et la couche model.
* unecouche **model** : implémentation du modèle des objets métiers.
* une couche **dao** : interface permettant d’interroger la base de données.
* une couche **service** : cette couche est utilisé dans certains microservice. Elle permet d’ajouter et de faciliter la manipulation des objets suivant le résultat attendu, lorsque cela est compliqué avec la couche controller.

### Les ressources

Nous retrouvons nos fichiers .properties détaillés [ici](Projet%20OC%20Pizza%20-%20Dossier%20d_exploitation.docx#ressources).

Pour le microservice interface, nous retrouvons également deux sous-dossiers :

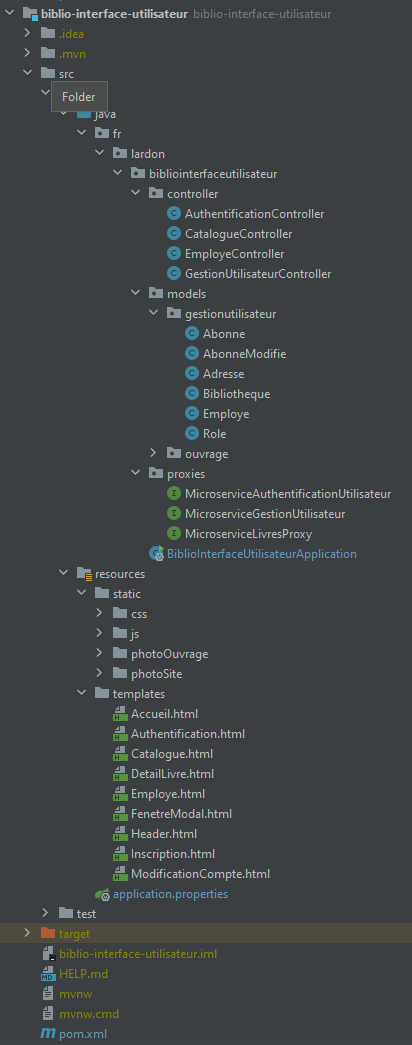
* Static : contenant les fichiers CSS, Javascript et également les photos du site.
* Template : contenant toutes les pages HTML de l’interface du projet.

Pour le microservice Batch, nous retrouvons les mêmes sous-dossiers :

* Static : contenant les photos affichés dans les emails
* Template : où l’on trouve les différentes pages HTML contenant les emails.

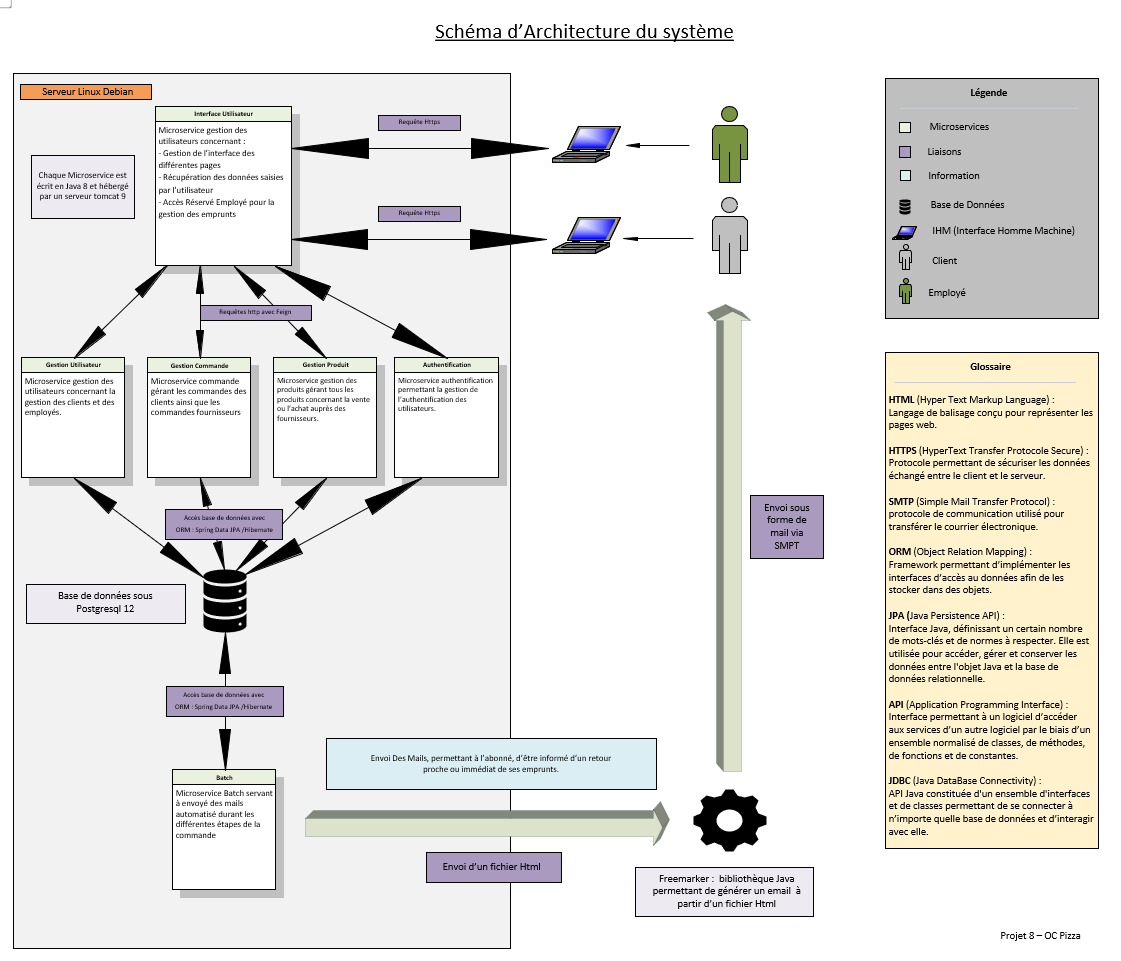
### Structure d’un microservice

La structuration du microservice est la suivante :



## Architecture Globale

L’architecture globale du projet est expliquée à travers ce schéma :



Ce schéma permet de comprendre l’échange entre l’application et le client ou l’employé via des requêtes https.

Ensuite les microservices échangent entre eux grâce au client http Feign, les données nécessaires récupérées dans la base de données, via Spring Data JPA / Hibernate.

Le microservice batch a également accès à ces données, pour ensuite les ajoutés dans les fichiers HTML grâce à Thymeleaf.

Et pour l’envoi, FreeMarker s’occupe de générer le mail à partir du fichier html.

L’envoi du mail est fait via SMTP au client ou à l’employé.

# Outils Technique

Les outils techniques utilisés dans le projet sont tous répertoriés dans ce tableau :

|  |  |
| --- | --- |
| **Technologie** | **Utilisation Projet** |
| Java 1.8.0\_281 | L'utilisation du langage java permet la mise en place des microservices. |
| Maven 3.6.3 | Utilisé pour la gestion des dépendances et la compilation du projet. |
| Spring 5.2.5 | Framework Java permettant de faciliter la programmation du code et de l'envoi des batchs |
| Spring Boot 2.2.6 | Le framework a permis de faciliter le développement de l'application en gérant automatiquement les dépendances et en ajoutant dans les fichiers pom, uniquement l'ensemble des dépendances utiles, par l'ajout des starters. |
| Spring Boot Starter Web 2.2.6 | Spring Boot Starter Web permet d'ajouter les dépendances permettant le fonctionnement du microservice(Tomcat, Hibernate…) |
| Feign 2.2.2 | Client http se trouvant dans le starter Spring Cloud. Il permet de scanner le microservice puis de créer les requêtes Http, afin de communiquer avec le microservice souhaité. |
| Lombok 1.18.16 | Libraire Java qui permet par l’intermédiaire d’annotation, de générer les getters, setters, constructeur, toString, equals, Hashcode des objets Java. |
| Thymeleaf 3.0.11 | Moteur de template qui simplifie la syntaxe, par l'intérmediaire de tags HTML via des attributs, qui permettent d'accéder aux variables et objets à l'intérieur des controllers, pour ensuite les afficher dans les pages web. |
| FreeMarker 2.3.30 | Librairie Java permettant à partir d'un fichier HTML de pouvoir générer un mail. |
| Postgresql 12.2-2 | PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO) permettant d'interagir avec la base de données. |
| pgAdmin 4.19 | Interface permettant de rédiger et appliquer les requêtes postgresql |
| Tomcat 9.0.33 | Le serveur tomcat permet de gérer les requêtes entre le client et le serveur, et de les rendre dynamique. |
| HTML 5 / CSS 3 | Le langage HTML 5 et CSS 3 permettent de créer et de représenter le contenu d'une page web et sa structure. |
| Bootstrap 4.0.0-2 | Librairie CSS et javascript, permettant de faciliter la conception de l’interface du site et sa responsivité. |
| Javascript | Le langage javascript permet de créer du contenu dynamique sur les pages web. |

La première partie du tableau sont les outils utilisés pour le coté Back-End de l’application.

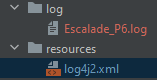
La deuxième partie regroupe les outils utilisés pour le coté Front-End de l’application.

# Points particuliers

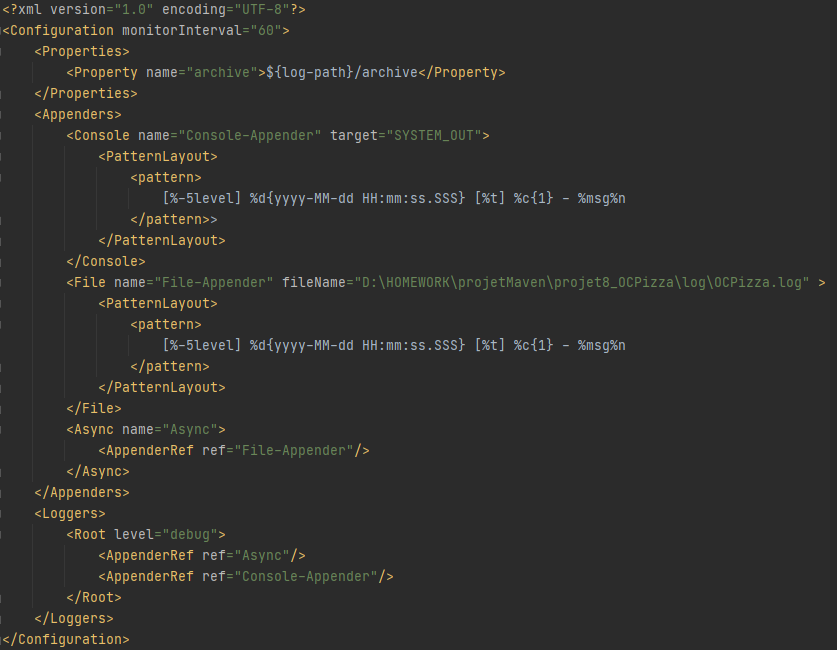
## Gestion des logs

La gestion des logs est gérée par log4j2.

La gestion des logs et le paramétrage se trouvent dans le fichier ‘log4j2.xml’ dans le dossier ‘OCPizza-technical’ :



Le détail du fichier :



Dans le fichier, la sortie des logs se fait sur la console mais également dans un fichier ‘OCPizza.log’.

Cela permet d’avoir une trace des logs, pour vérifier le bon fonctionnement de l’application et de gérer en cas de problème.

## Ressources

Les ressources sont détaillées [ici](Projet%20OC%20Pizza%20-%20Dossier%20d_exploitation.docx#ressources).

## Environnement de développement

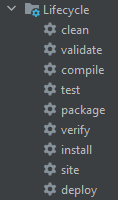
Le projet est réalisé sur l’environnement de développement ‘IntelliJ IDEA 2020.1.0’

Les informations de l’IDE se trouve [ici](https://www.jetbrains.com/fr-fr/idea/).

Le téléchargement de l’environnement se trouve dans ce [lien](https://www.jetbrains.com/fr-fr/idea/download/#section=windows).

## Procédure de packaging / livraison

Maven s’occupe de packager les microservices par l’intermédiaire de l’interface de l’IDE :



La livraison du projet comprends :

* Le code de l’application.
* Les fichiers WAR de tous les microservices.
* Les sauvegardes data et structure de la base de données.
* Le dossier de conception fonctionnelle.
* Le dossier d’exploitation.
* Le dossier de conception technique.
* Le PV de livraison.

# Glossaire

|  |  |  |
| --- | --- | --- |
| **Log4j2** | Log4J2 est une API de log proposée par la fondation Apache, pour la gestion des logs. | |
| **Postgresql** | PostgreSQL est un système de gestion de base de données relationnelle et objet permettant la gestion de la base de données. | |
| **Maven** | Maven est un outil de gestion et d’automatisation de production des projets logiciels Java en général et JavaEE en particulier. | |
| **Tomcat** | Serveur permettant d’exécuter des applications Web développées avec les technologies Java. | |
| **Hibernate** | Hibernate est un ORM (Object Relation Mapping). C’est un framework permettant d’implémenter les interfaces d’accès aux données afin de les stocker dans des objets. | |
| **IHM** | IHM (Interface-Homme-Machine) sont les moyens et outils mis en œuvre afin qu'un humain puisse contrôler et communiquer avec une machine. | |
| **JDBC** | JDBC (Java DataBase Connectivity) est une API Java constituée d'un ensemble d'interfaces et de classes permettant de se connecter à n’importe quelle base de données et d’interagir avec elle. | |
| **JPA** | JPA **(**Java Persistence API) Interface Java, définissant un certain nombre de mots-clés et de normes à respecter. Elle est utilisée pour accéder, gérer et conserver les données entre l'objet Java et la base de données relationnelle. | |
| **API** | API (Application Programming Interface) Interface permettant à un logiciel d’accéder aux services d’un autre logiciel par le biais d’un ensemble normalisé de classes, de méthodes, de fonctions et de constantes. | |
| **SMTP** | SMTP (Simple Mail Transfer Protocol) protocole de communication utilisé pour transférer le courrier électronique. | |
| **HTTPS** | HTTPS(HyperText Transfer Protocole Secure) Protocole permettant de sécuriser les données échangées entre le client et le serveur. | |
| **HTML** | HTML(Hyper Text Markup Language) Langage de balisage conçu pour représenter les pages web. | |
| **WAR** | Le fichier WAR est une archive Java. Elle permet de structurer et de paramétrer l’application en elle-même. Il détermine également les dépendances et les variables lors du déploiement d’une application web. | |
| **GitHub** | GitHub est un service web d’hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. | |
| **FreeMarker** | **FreeMarker** est un moteur de template, il est fourni par **Apache** comme une bibliothèque **Java** source ouverte. **FreeMarker** lit des fichiers modèle et les combine avec des objets **Java** pour générer un texte de sortie (Html, email, ..). | |
|  | |  | |
|  | |  | |