



OC PIZZA

Application web de vente en ligne et logistique

Dossier de conception technique

Version 1.0

Auteur

Nicolas Deroussen

Analyste développeur

TABLE DES MATIÈRES

1 – VERSIONS.....	3
2 – INTRODUCTION.....	4
2.1 - OBJET DU DOCUMENT.....	4
2.2 – REFERENCES.....	4
3 – ARCHITECTURE TECHNIQUE.....	5
3.1 – APPLICATION OC PIZZA.....	5
3.2 – LA BASE DE DONNEES.....	6
3.2.1 – SGBD UTILISE.....	6
3.2.2 – MPD.....	6
3.2.3 – PRESENTATION DES TABLES.....	8
4 – ARCHITECTURE LOGICIELLE.....	14
4.1 – PRINCIPES GENERAUX.....	14
4.1.1 – LES COUCHES.....	14
4.1.2 – CREATION DE L'APPLICATION.....	15
4.1.3 – LE DIAGRAMME DE COMPOSANTS.....	21
5 – ARCHITECTURE DE DEPLOIEMENT.....	22
5.1 – DIAGRAMME DE DEPLOIEMENT.....	22
5.2 – SERVEUR DE DEPLOIEMENT.....	23
6 – GLOSSAIRE.....	24

1 – VERSIONS

AUTEUR	DATE	DESCRIPTION	VERSION
Nicolas DEROUSSEN	02/03/2020	Création du document	1.0

2 – INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application de gestion d'OC Pizza. L'objectif du document est d'informer et aider les développeurs pour la conception, la maintenance et l'évolution de l'application.

Les éléments du présent dossier découlent :

- du domaine fonctionnel
- des spécifications techniques
- du modèle physique de donnée

2.2 – Références

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

Projet 8 – Dossier d'exploitation

**Projet 8 Dossier de conception
fonctionnelle**

3 – ARCHITECTURE TECHNIQUE

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

3.1 - Application OC Pizza

Le front-end est la partie visible du site-web (partie où il y a les interactions client). Il sera réalisé à l'aide des langages de programmation HTML, CSS, Thymleaf, Bootstrap et JavaScript car ce sont de véritables standards qui sont à la base de tout projet de développement web. Bootstrap sera utilisé pour la partie responsive du site.

Le back-end est la partie logique du site-web. Il sera réalisé avec le langage de programmation Java EE et du puissant framework : Spring, ce framework permet de simplifier grandement le développement et de se concentrer sur la partie métier plutôt que technique. Le back-end sera en mesure de communiquer avec des web-services extérieur tel que google map.

Le back-end communiquera avec la base de données MySQL. La technologie MySQL est fiable et très utilisé par de grande entreprise tel que Ebay, Amazon et de grandes banques.

3.2 - La base de données

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

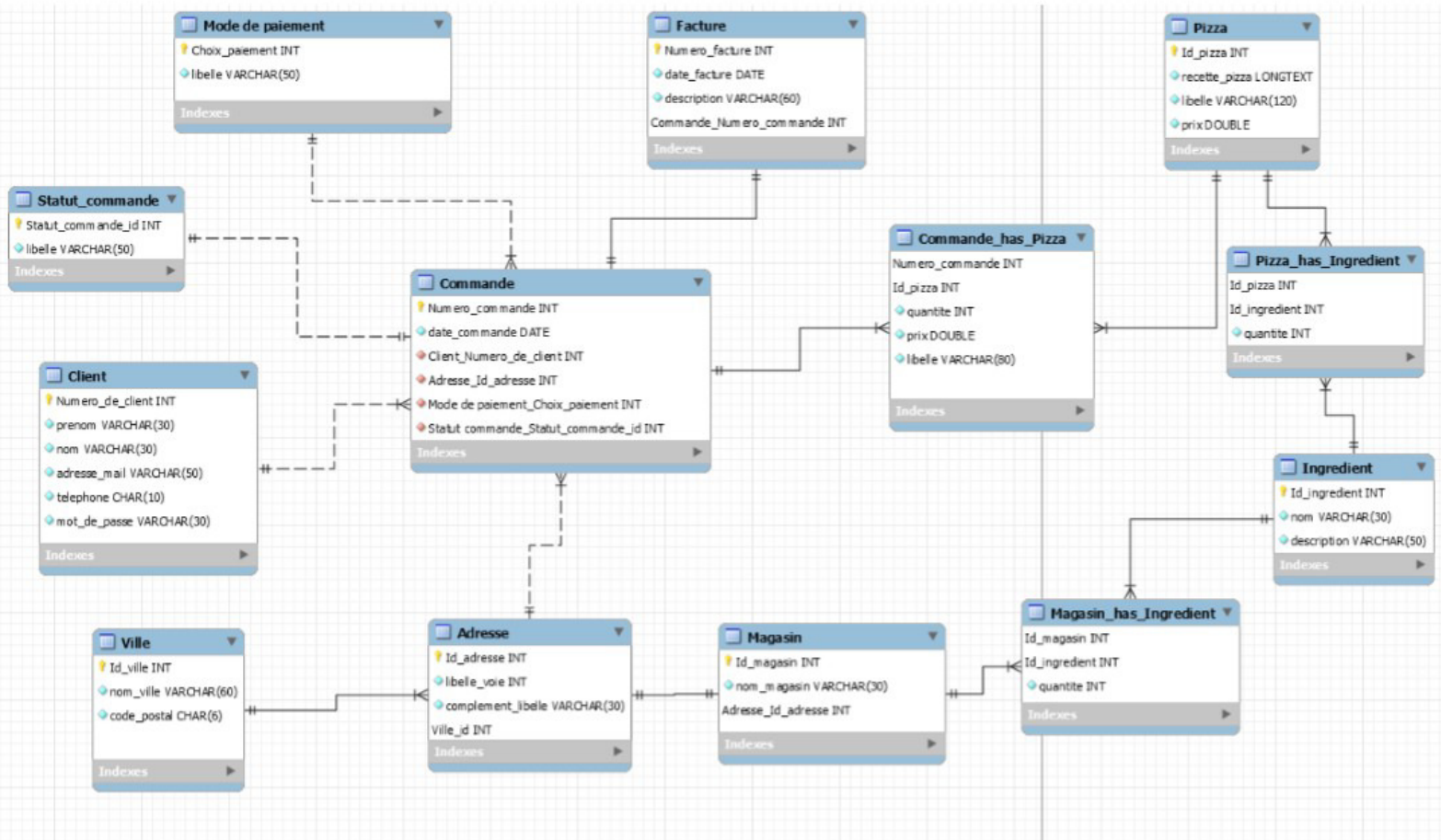
3.2.1 - SGBD utilisé

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

3.2.2 – MPD

A partir du diagramme de classes présenté dans le dossier de conception fonctionnelle, nous avons élaboré le MPD (Modèle Physique de Données). Il 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.

Le modèle physique de donnée



3.2.3 – Présentation des tables

Client

Cette table regroupe les informations des « Clients », en général.

Sa clé primaire est Numero_de_client avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- VARCHAR, avec 30 caractères pour le nom, le prenom et le mot de passe ;
- VARCHAR, avec 50 caractères pour l'adresse e-mail ;
- CHAR, avec 10 caractères pour le telephone.

Commande

Cette table regroupe les informations des « Commandes », en général.

Sa clé primaire est Numero_commande avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- DATE, pour la date de la commande ;
- INTEGER, pour les clefs étrangères suivantes :
 - Numero_de_client, venant de la table «Client»
 - Id_adresse, venant de la table «Adresse»
 - Mode_de_paiement, venant de la table «Mode_de_paiement»
 - Statut_commande_id, venant de la table «Statut_commande»

Adresse

Cette table regroupe les informations des « Adresses », en général.

Sa clé primaire est Id_adresse avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- VARCHAR, avec 30 caractères pour complement_libelle ;
- INTEGER, pour libelle_voie et pour la clef étrangère suivante :
 - Ville_id, venant de la table «Ville»

Ville

Cette table regroupe les informations des « Villes », en général.

Sa clé primaire est Id_ville avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- VARCHAR, avec 60 caractères pour nom_ville et 6 caractères pour code_postal;

Magasin

Cette table regroupe les informations des « Magasins », en général.

Sa clé primaire est Id_magasin avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- VARCHAR, avec 30 caractères pour nom_magasin ;
- INTEGER, pour la clef étrangère suivante :
 - Id_adresse , venant de la table «Adresse»

Magasin_has_Ingredient

Il s'agit de la table intermédiaire entre Magasin et Ingredient puisque ces deux tables sont reliées avec une relation ManyToMany.

Cette table est utilisée pour voir la quantité d'ingredient que chaque magasin possède séparément.

Ses deux clés étrangères de type INTEGER sont :

- Id_magasin, venant de la table «Magasin»
- Id_ingredient, venant de la table «Ingredient»

Elle possède un attribut propre à cette table de type INTEGER : quantite.

Ingredient

Cette table regroupe les informations des « Ingredients », en général.
Sa clé primaire est Id_ingredient avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- VARCHAR, avec 30 caractères pour nom et 50 caractères pour description;

Pizza_has_Ingredient

Il s'agit de la table intermédiaire entre Pizza et Ingredient puisque ces deux tables sont reliées avec une relation ManyToMany.

Cette table est utilisée pour voir le nombre d'ingredient contenu dans chaque pizza.

Ses deux clés étrangères de type INTEGER sont :

- Id_pizza, venant de la table «Pizza»
- Id_ingredient, venant de la table «Ingredient»

Elle possède un attribut propre à cette table de type INTEGER : quantite.

Pizza

Cette table regroupe les informations des « Pizzas », en général.

Sa clé primaire est Id_pizza avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- LONGTEXT, pour recette_pizza;
- VARCHAR, avec 120 caractères pour libelle;
- DOUBLE, pour le prix;

Commande_has_Pizza

Il s'agit de la table intermédiaire entre Commande et Pizza puisque ces deux tables sont reliées avec une relation ManyToMany.

Cette table est utilisée pour voir le nombre de pizzas contenu dans chaque commande.

Ses deux clés étrangères de type INTEGER sont :

- Numero_commande, venant de la table «Commande»
- Id_pizza, venant de la table «Pizza»

Elle possède trois attributs propre :

- INTEGER, avec quantite.
- DOUBLE, avec prix.
- VARCHAR, avec libele.

Facture

Cette table regroupe les informations des « Factures », en général.

Sa clé primaire est Numero_facture avec un AUTO_INCREMENT, et de type INTEGER.

Elle possède une clé étrangère de type INTEGER : Numero_commande venant de la table «Commande».

Pour les autres colonnes, le type est :

- DATE, pour date_facture;
- VARCHAR, avec 60 caractères pour description;
- DOUBLE, pour le prix;

Mode_de_paiement

Cette table regroupe les informations des « Mode_de_paiements », en général.
Sa clé primaire est Choix_paiement avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- VARCHAR, avec 50 caractères pour libelle;

Statut_commande

Cette table regroupe les informations des « Statut_commande », en général.
Sa clé primaire est Statut_commande avec un AUTO_INCREMENT, et de type INTEGER.

Pour les autres colonnes, le type est :

- VARCHAR, avec 50 caractères pour libelle;

4 – ARCHITECTURE LOGICIELLE

4.1 – PRINCIPES GENERAUX

Les sources et versions du projet sont gérées par Git et les dépendances par Maven.

L'IDE utilisé sera Eclipse.

Le développement de l'application se présente sous la forme d'un projet SpringBoot en utilisant le principe MVC.

L'utilisation de Spring aidera à la conception et facilitera grandement la tâche grâce à l'inversion de contrôle (IoC)

4.1.1 – Les couches

La structure des couches se base sur une architecture n-tiers. Cette architecture est principalement composée de quatre couches :

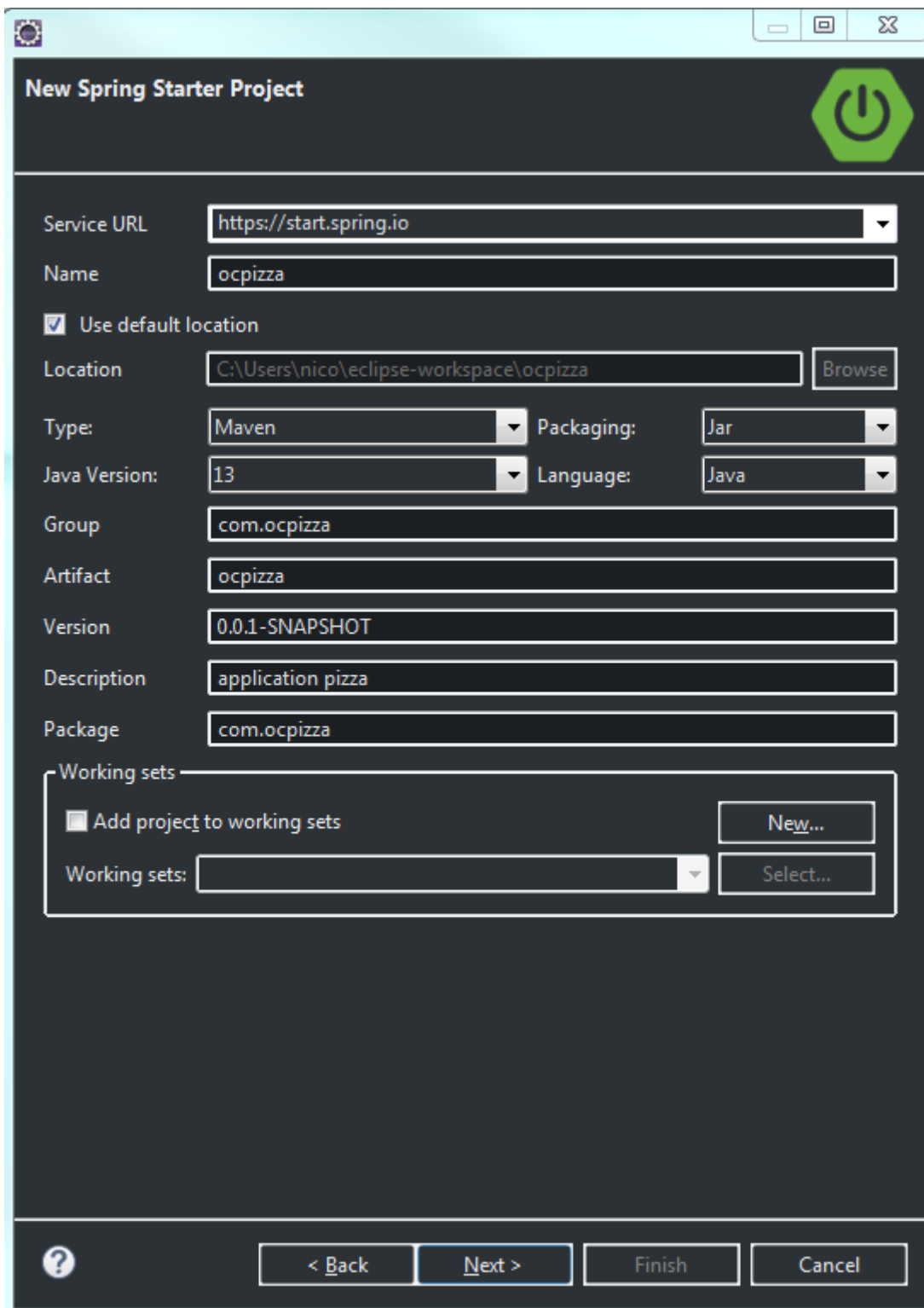
- présentation
- business
- consumer
- model

Les couches sont représentées par les différents packages de l'application :

Nom du package	Description du package
com.ocpizza.configuration	Configuration de spring security.
com.ocpizza.controller	Les contrôleurs permettent de rediriger vers les différentes vues selon les url demandées. Ce package est associé à la couche présentation.
com.ocpizza.dao	La couche d'accès aux données via la base de données. Ce package est associé à la couche consumer.
com.ocpizza.entities	Les différents objets du projet. Ce package est associé à la couche model.
com.ocpizza.service	Les services et leurs implémentations. Ce package est associé à la couche business.

4.1.2 – Création de l'application

Une fois dans la partie Java JEE sur Eclipse, aller dans File puis New puis Other puis trouver Spring Boot. Une fois cela fait cliquer sur Spring Starter Project puis next.



The image shows a 'New Spring Starter Project' dialog box with a dark theme. It features a title bar with standard window controls and a green power button icon. The form contains several input fields and dropdown menus for configuring a new project. At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel', along with a help icon.

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

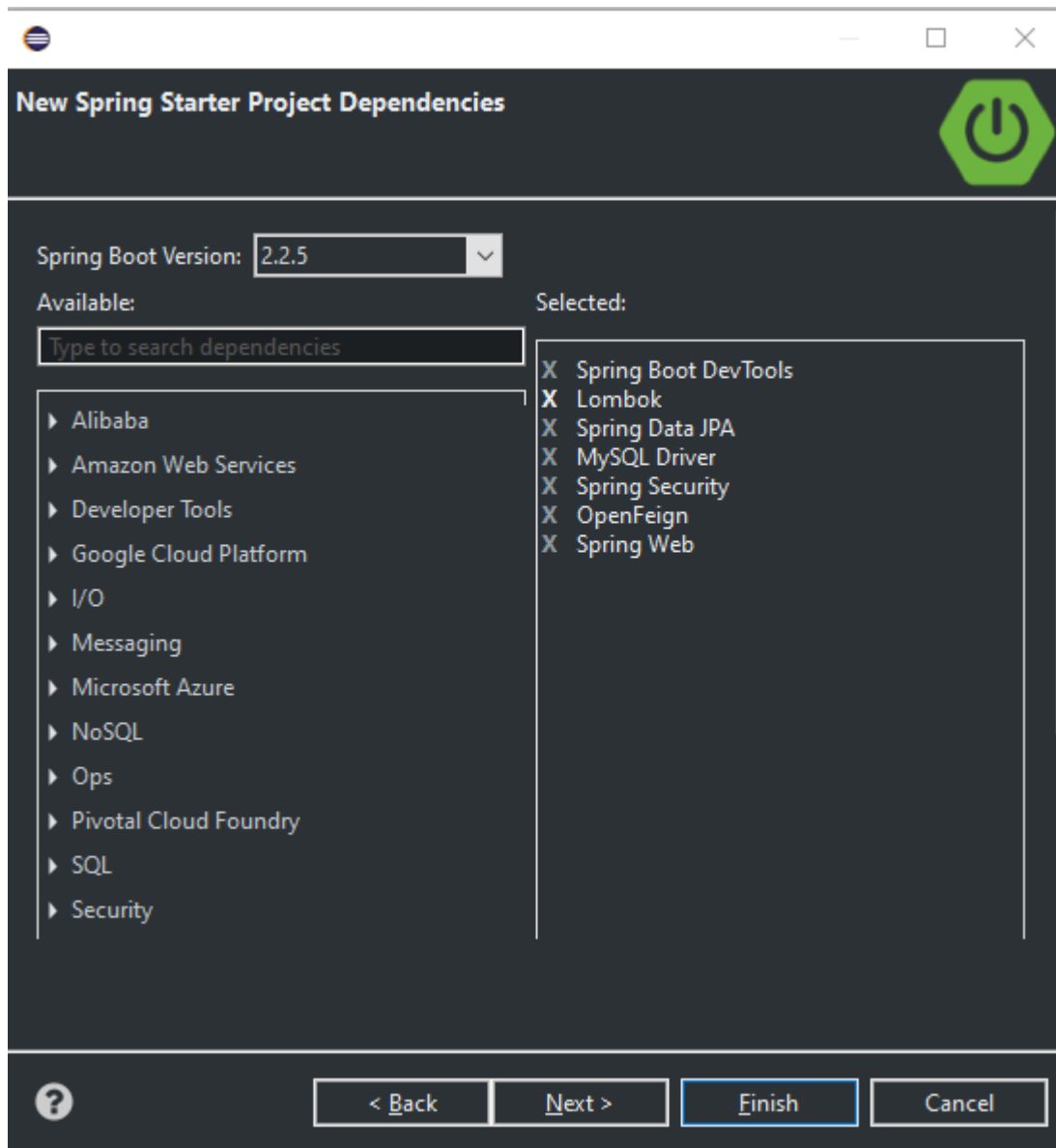
Package:

Working sets

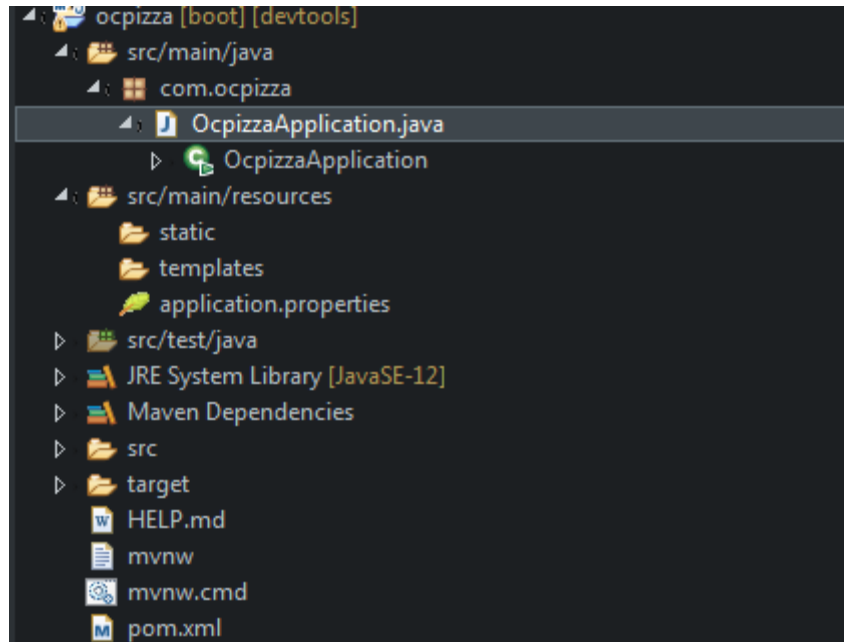
☐ Add project to working sets

Working sets:

Appuyer ensuite sur next.

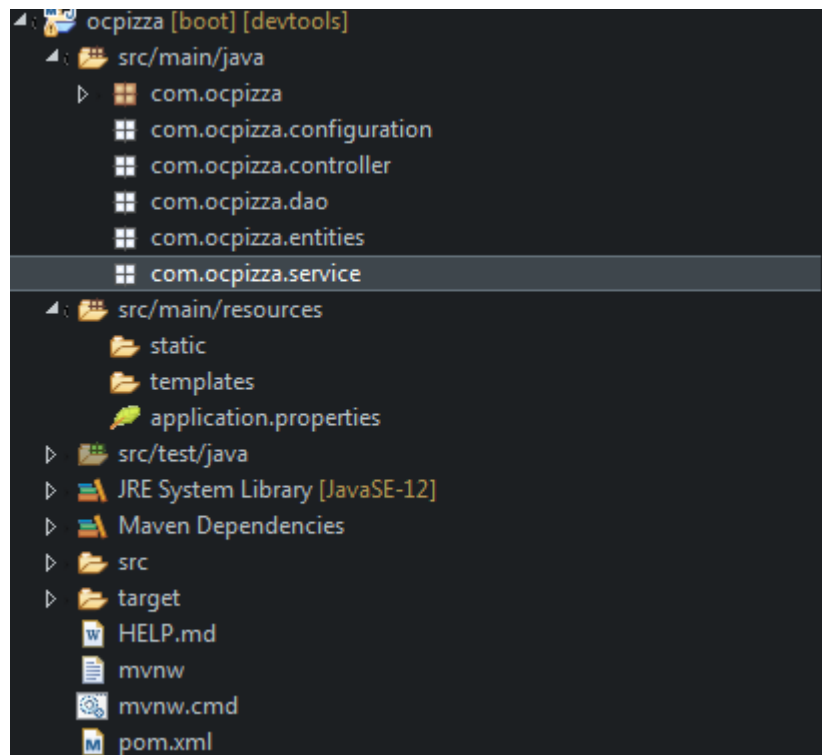


Une fois les dépendances ajoutées appuyer sur finish.



Voici à quoi devra ressembler le projet fraîchement créé. La méthode Main de l'application sera dans le package com.ocpizza et dans la classe OcpizzaApplication.java.

Il faudra ensuite créer les différents packages de l'application :



La partie src/main/resources sera utilisée pour les fichiers static comme les images et également pour le css et Bootstrap.

La partie templates stockera les différentes pages html du projet.

Le fichier application.properties.

Le fichier application.properties situé dans src/main/resources servira pour définir les données requises pour l'accès à la base de donnée ainsi que des données importantes comme le port du serveur ou de la base de donnée. Voici les données à mettre dedans :

database configuration

spring.datasource.url = \${source.database.url.value}

spring.datasource.username = \${username.database.value}

spring.datasource.password = \${password.database.value}

port configuration

server.port = \${port.server.value}

hibernate configuration

spring.jpa.show-sql = true

spring.jpa.hibernate.ddl-auto = update

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

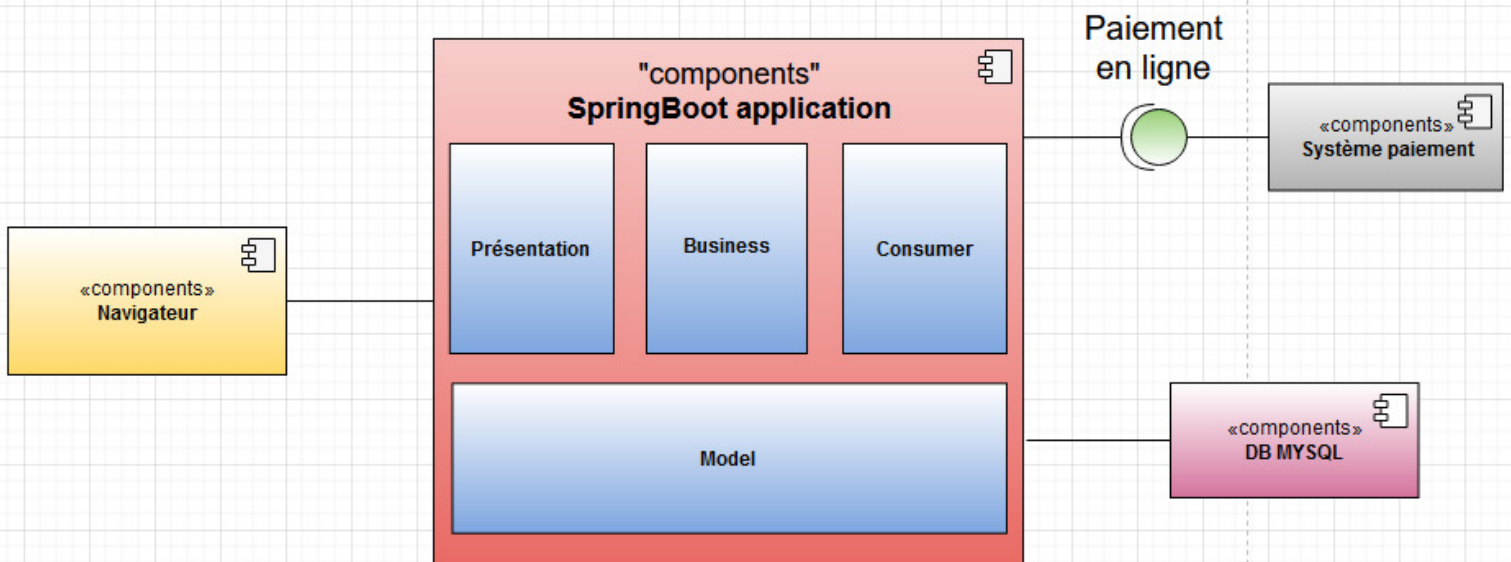
thymeleaf configuration

spring.thymeleaf.mode = LEGACYHTML5

spring.thymeleaf.cache = false

4.1.3 – Le diagramme de composants

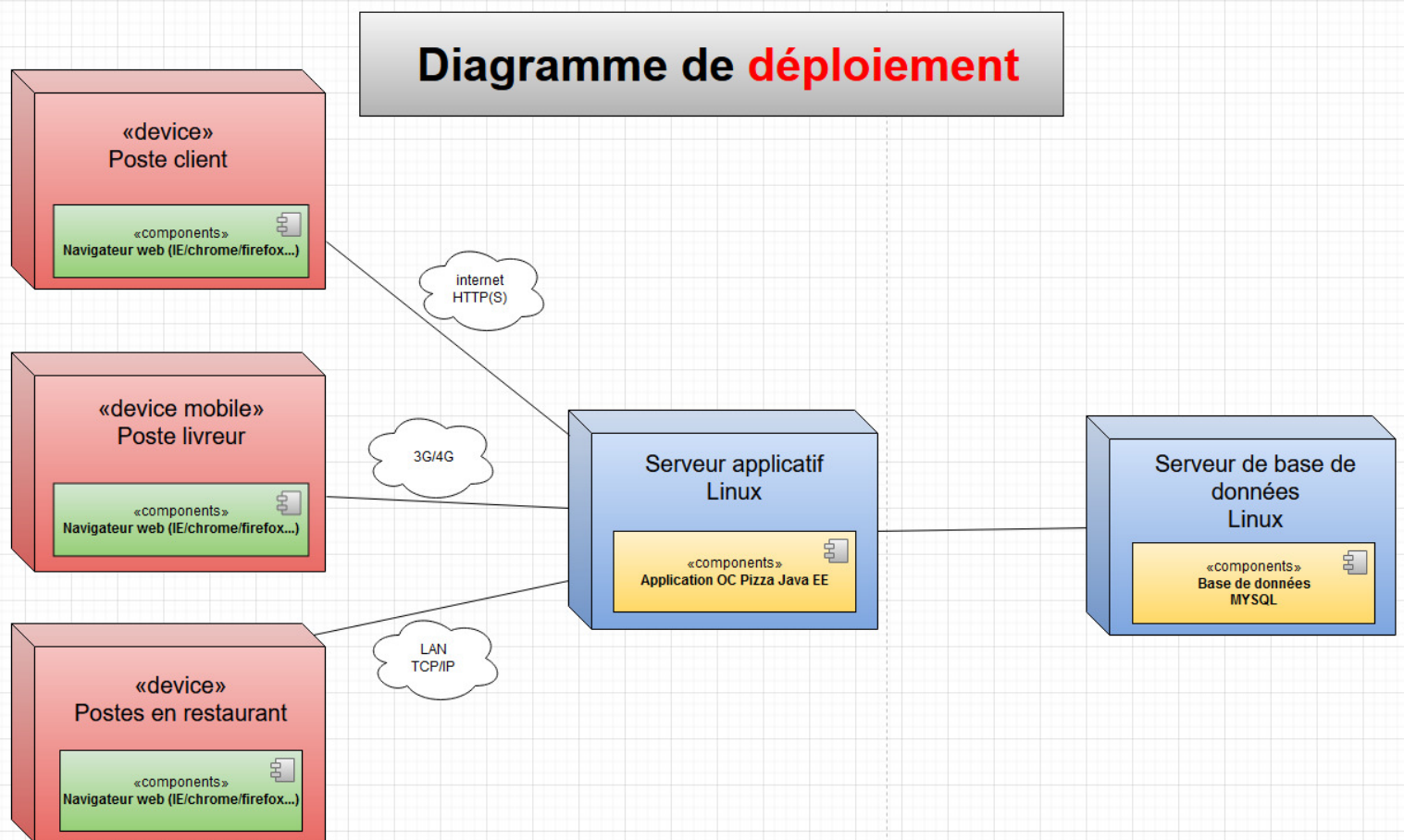
Diagramme de composants



5 – ARCHITECTURE DE DEPLOIEMENT

5.1 – LE DIAGRAMME DE DEPLOIEMENT

Apache tomcat 9 sera utilisé comme serveur HTTP pour l'application.



5.2 – SERVEUR DE DEPLOIEMENT

L'application développée sous le Framework Spring est déployée sur un serveur dédié virtuel (VPS) Daily Razor (Spring MVC Hosting)

L'offre d'hébergement sera **JAVA-PT32** HOSTING à 3,36\$.

JAVA-PT32 HOSTING

Starting at

\$3.36 /monthly

15% OFF (was \$3.95)



Private Tomcat Server



32 MB Private JVM & PermGen

- ✓ 2 Websites/Domains
- ✗ Free Domain Name
- ✓ Free SSL Security
- ✓ Free Website Builder
- ✓ 5 GB Storage
- ✓ 100 GB Transfer
- ✓ 10 Databases
- ✓ Tomcat version 9, 8.5, 8, 7 & 6 with JDK 11/10/9/8/7/6
- ✓ 1-Click Tomcat Restart or via SSH

6 – GLOSSAIRE

MPD (Modèle Physique de Données) consiste à implanter une base de donnée dans un SGBD*.

SGBD* est un système de gestion de base de données, c'est un logiciel système servant à stocker, à manipuler ou gérer, et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations.