OC Pizza

Développement du système informatique d'un groupe de pizzerias

Dossier de conception technique

Version 1

Auteur
J. Rouzic
Développeur

Table des matières

1 - Versions

Auteur	Date	Description	Version
J. Rouzic	06/05/2019	Création du document	1
J. Rouzic	09/06/2019	Finalisation du document	1
J.Rouzic	17/06/2019	Reprise du document	1

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier de conception technique du système informatique. Il décrit l'architecture de l'ensemble d'application qui constistuent ce SI.

2.2 - Références

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

1. **Doc 1 - Dossier de conception fonctionnelle.pdf** : Dossier de conception fonctionnelle de l'application
2. **Doc 3 - Dossier d_exploitation.pdf**
3. **Doc 4 - PV Livraison.pdf**

3 - Architecture Technique

3.1 - Composants généraux

Pour vous figurer l'architecture générale du système, deux annexes sont à votre disposition :

Doc 2 – Annexe 1.pdf : une version simplifiée.

Doc 2 – Annexe 2.pdf : version complète et détaillée.

3.2 - Application Webservice REST

La pile logicielle est la suivante :

- Application **Spring Boot** (JDK version 1.8)
- Serveur d'application **Tomcat 9.18** (inclus dans le .JAR de l'application)
- Spring Web – Spring Data/JPA connecté à une base de données PostgreSQL 9.6.

Elles se composent principalement des modules suivants :

- Un module d'application API REST

  Ce module a pour rôle de récupérer les requêtes GET/POST d'appel à l'API, ces requêtes sont émises par l'application Web (décrite au 3.3) et par l'application Angular.

  Il renvoie des réponses au format JSON.

- Un module Consumer

  Basé principalement sur Spring Data / JPA il fera la jonction entre le module métier du Webservice et la base de données.

- Un module Authentication helper component

  Une authentification réussie dans le module Spring MVC doit permettre d'être automatiquement authentifiée au sein du module API REST, c'est ce que fait cet helper.

3.3 - Application Web Spring Boot

La pile logicielle est la suivante :

- Application **Spring Boot** (JDK version 1.8)
- serveur d'application **Tomcat 9.18** (inclus dans le .JAR de l'application)
- Spring Boot – Spring MVC

Elle se compose principalement des modules suivants :

- Un module d'application Webapp Spring MVC

  C'est sur ce module que l'utilisateur est redirigé lorsqu'il accède à l'URL du site.

  Ce module a pour rôle de proposer une page d'index au contenu statique ainsi qu'un

formulaire d’authenfication / inscription, il reçoit un login / mot de passe et l’adresse IP (on reconnaitra les adresses IP fixes des structures d’Ocpizza).

Il renvoie ensuite une application Web de type “single page application” (décrite au 3.4 et 3.5).

- Un module Consumer

  Son rôle est d’appeler / consommer l’API REST pour récupérer des données, authentifier et inscrire les utilisateurs.

3.4 - Application Angular Professionnelle

Elle est basée sur le framework **Angular 8.0.**

Elle émet des requêtes GET/POST destinées à l’API REST et consomme le JSON délivré par l’API REST.

Elle fournit les fonctionnalités requises pour une utilisation professionnelle du système informatique, de la gestion de commande à la gestion des stocks.

3.5 - Application Angular Cliente

Identique à l’application Angular professionnelle, elle dispose cependant de moins de fonctionnalités.

Elle fournit toutes les fonctionnalités requises pour l’utilisation du magasin en ligne par le client.

4 - Architecture de Déploiement

4.1 - Serveur de Base de données

On privilégie un hebergement de type “Platform as a Service” (P.A.A.S) afin de déléguer les problèmatiques de mise à jour et de sécurité.

4.2 - Applications Angular

Deux application de type “single-page applications” : une **professionnelle** et une **cliente**. Elle permettent de couvrir la plupart des cas d’utilisation. Elles sont déployées sur le terminal du client : quand le client quitte la page d’index, la webapp Spring MVC redirige alors vers la **“single-page application” cliente** ; après connexion, si les identifiants et l’adresse IP apparaissent comme étant ceux d’un professionnel, la Webapp redirige celle une nouvelle **“single-page application”**, celle des **professionnels**.

Elles sont déployées chez un “Platform as a service”.

4.3 - Serveurs d’applications Spring MVC et webservice REST

Le Webservice REST et la Webapp sont également chez un hébergeur de type “Platform as a Service”.

4.4 - Diagramme de déploiement

Se référer à l’annexe **Doc 2 – Annexe 3.pdf**

5 - Architecture logicielle

Les sources et versions du projet sont gérées par **Git**, les dépendances et le packaging par **Apache Maven** pour les projets Spring Boot. Pour les projets Angular on utilisera **npm** et **Angular CLI**.

5.1 - Les couches

5.1.1 - Couche de la Webapp Spring MVC

- une couche **business** : responsable de la logique métier du composant.
- une couche **model** : implémentation du modèle des objets métiers.
- une couche **consumer** : couche d'accès au Webservice REST.
- une couche **webapp** : la couche qui réalise le modèle MVC pour les pages d'index et d'authentification / inscription.
- une couche **technical** : contient des élements techniques.

5.1.2 - Couche du Webservice REST

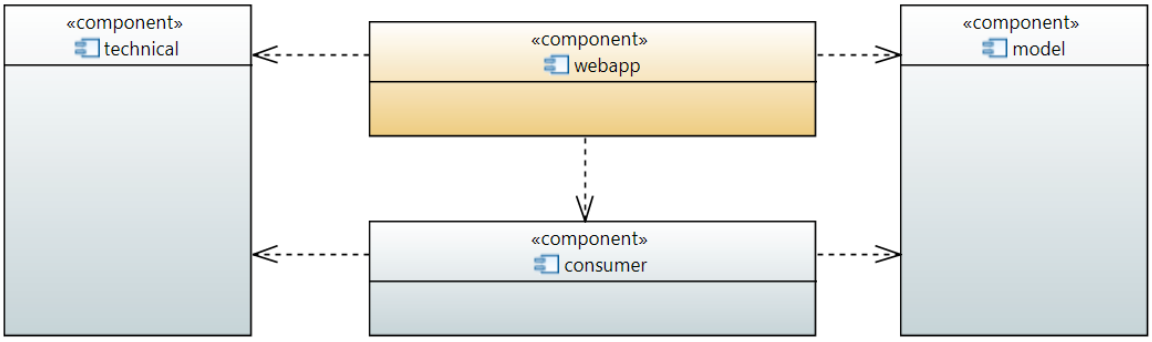
- une couche **business** : responsable de la logique métier du composant.
- une couche **model** : implémentation du modèle des objets métiers.
- une couche **consumer** : couche d'accès à la base de données.
- une couche **webservice-rest** : le web service REST qui fonctionne de paire avec la "single page application".
- une couche **technical** : contient des élements techniques.

5.1.3 - Couche des applications Angular cliente et professionnelle

- une couche **app** : la couche de base d'une application angular, elle contient les autres couches à savoir
  - une couche **core** : responsible de la vue et de la logique, elle contient les composants qui constituent la "single-page application"
  - une couche **consumer** : elle contient les composants qui communiquent avec le Webservice REST
  - une couche **model** : elle contient les elements du model et est partagée.

5.2 - Les modules

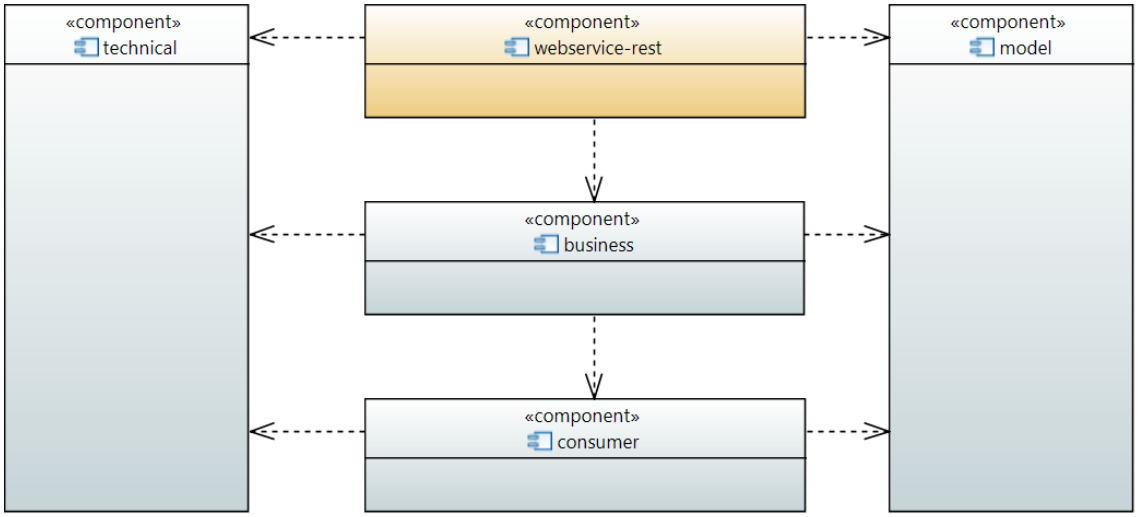
5.2.1 - Modules de la Webapp spring MVC



Le module Webapp propose les pages web statiques à l'utilisateur.

Le module Consumer se connecte au module Webservice-rest (ci-dessous)

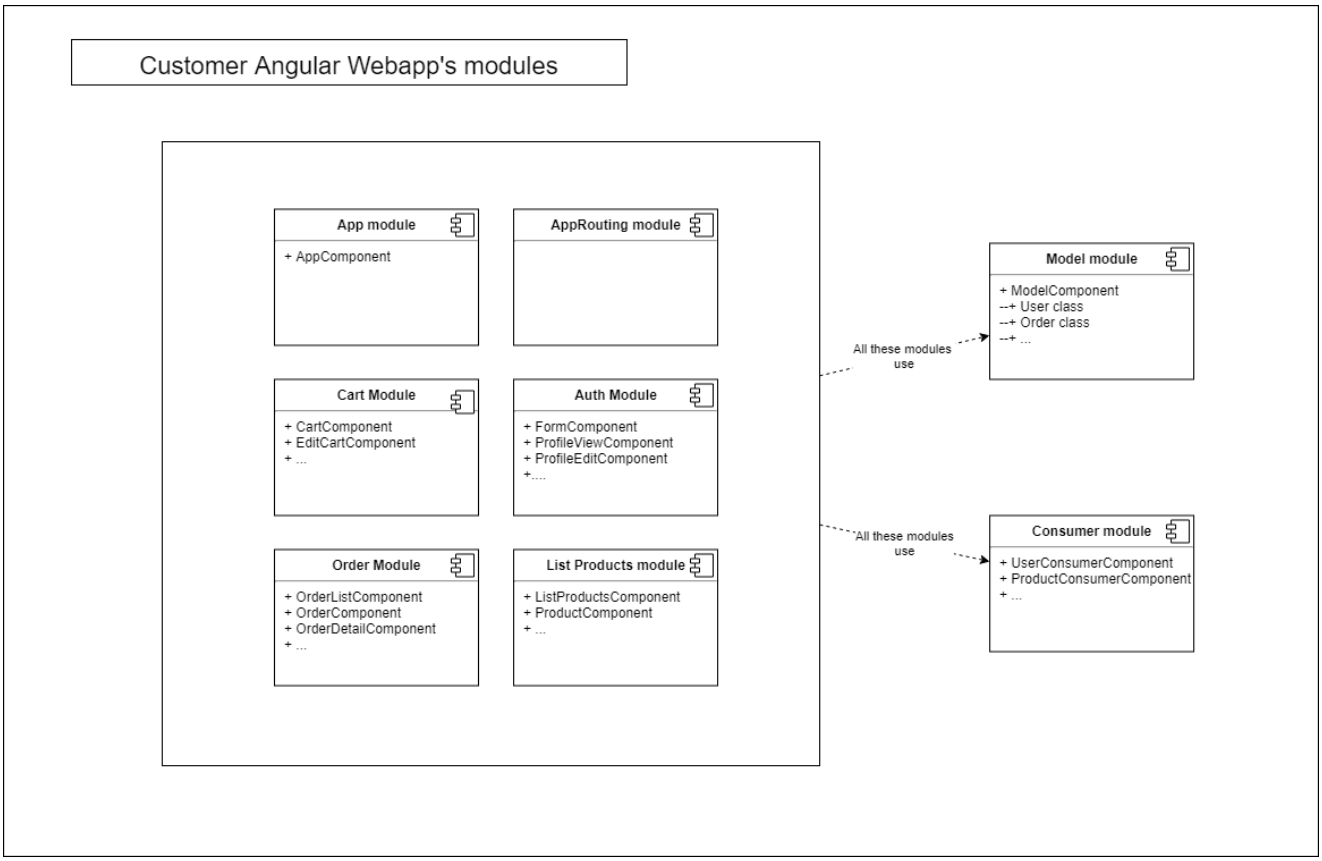
5.2.2 - Modules du Service web REST



Le module Webservice-rest expose l'API aux applications du SI.

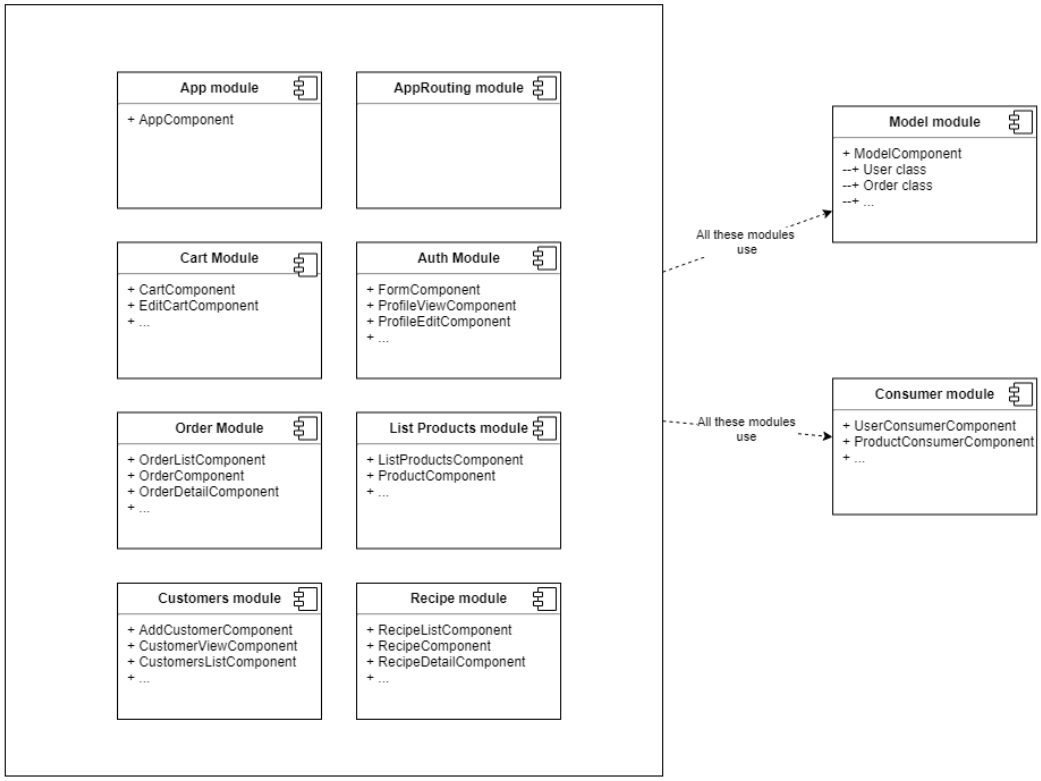
Le module Consumer accède au serveur de base de données.

5.2.3 - Modules de l'application Angular cliente



5.2.4 - Modules de l'application Angular professionnelle

Professional Angular Webapp's modules

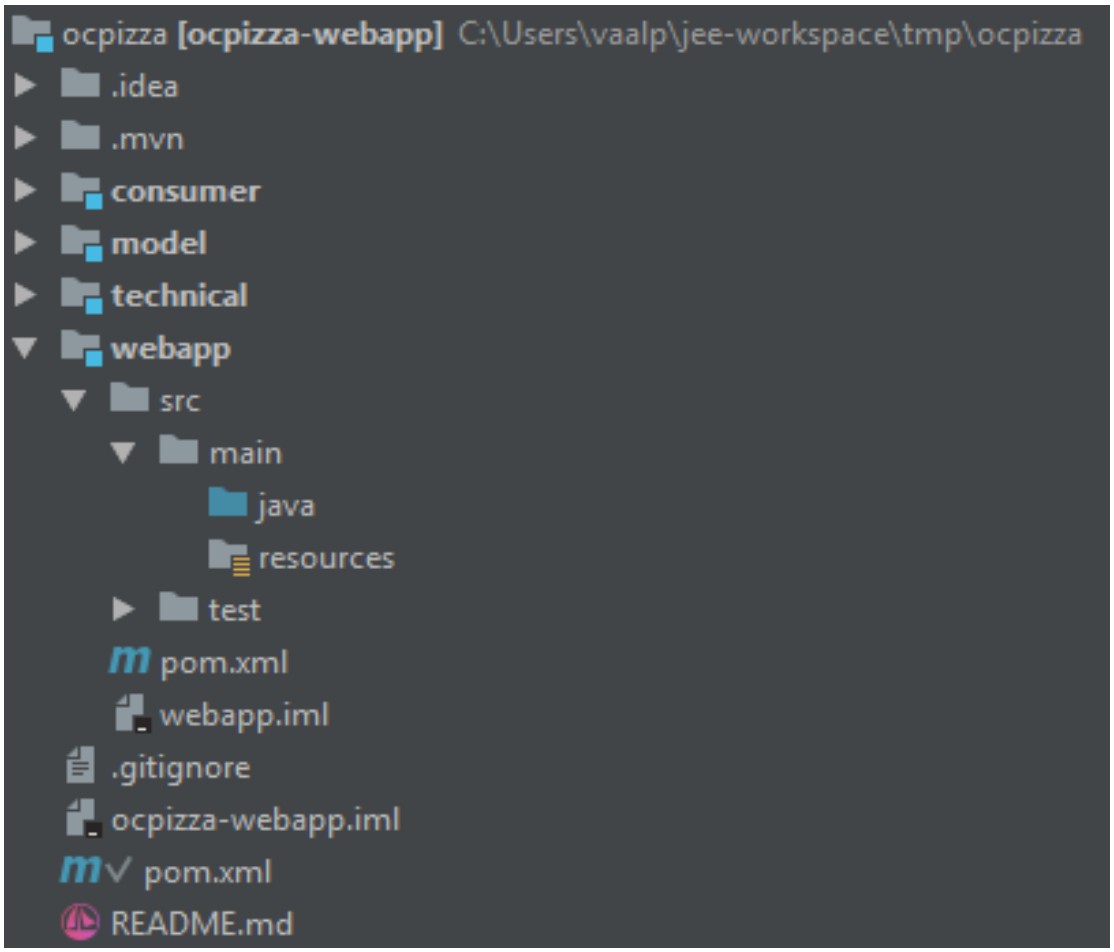


5.3 - Structure des sources

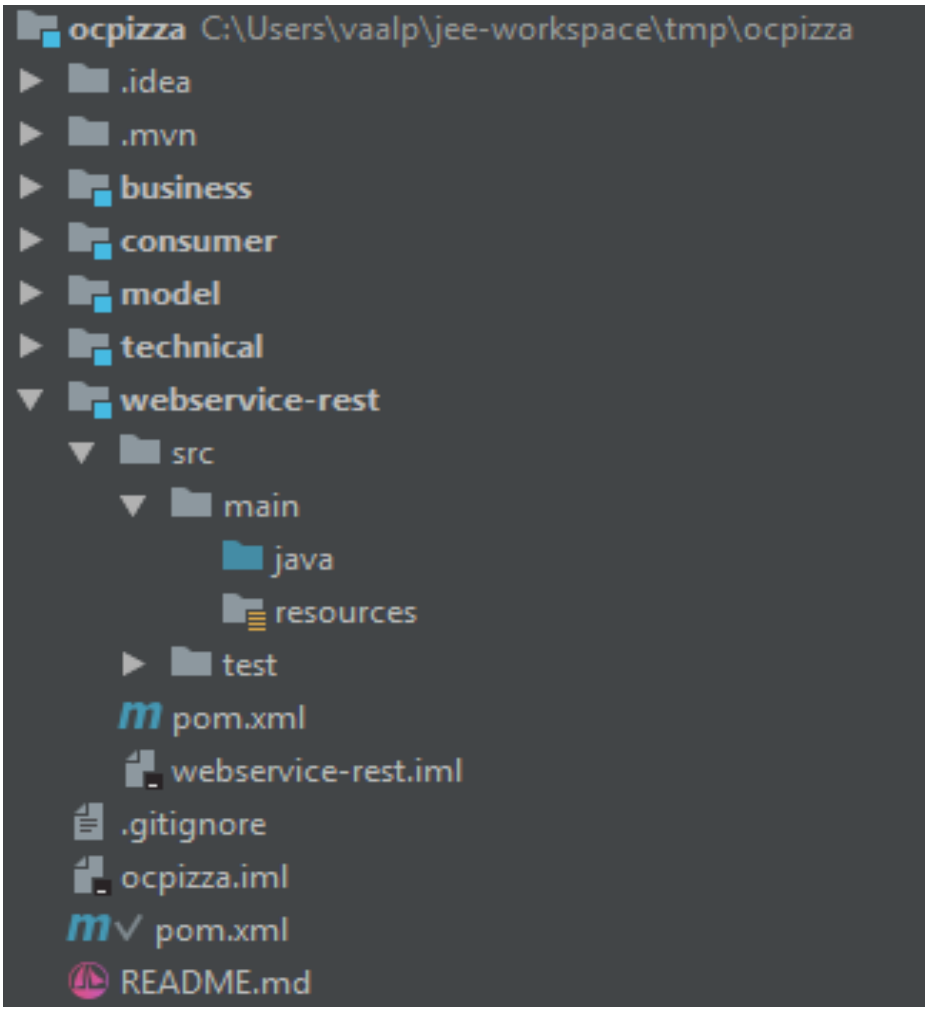
La structuration des répertoires du projet suit la logique suivante :

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

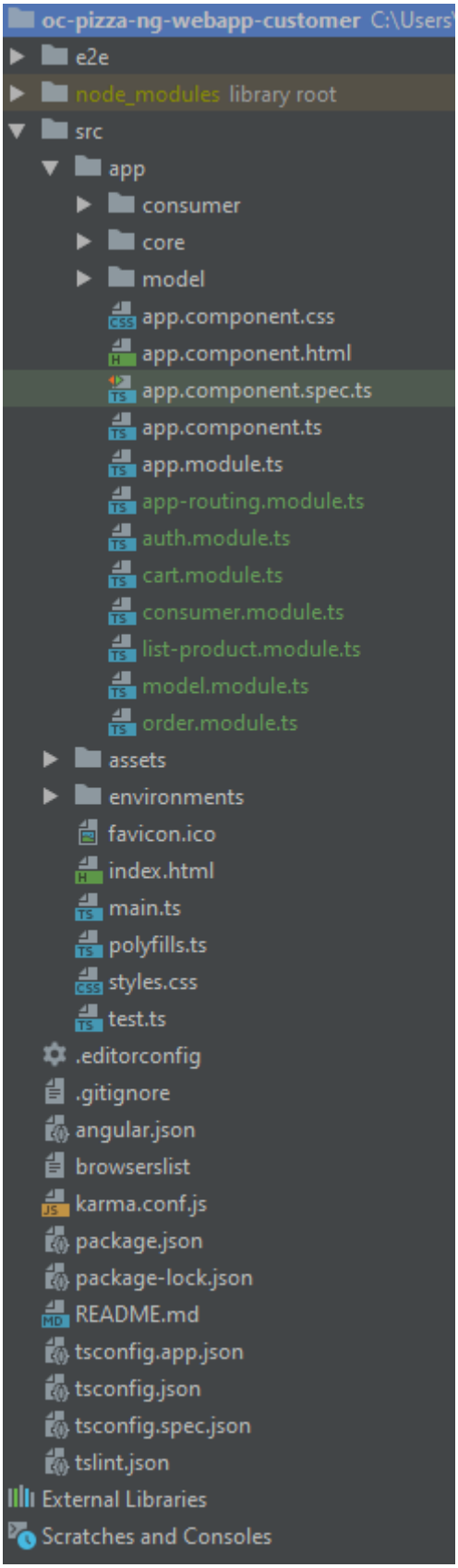
5.3.1 - Structure de la Webapp Spring MVC



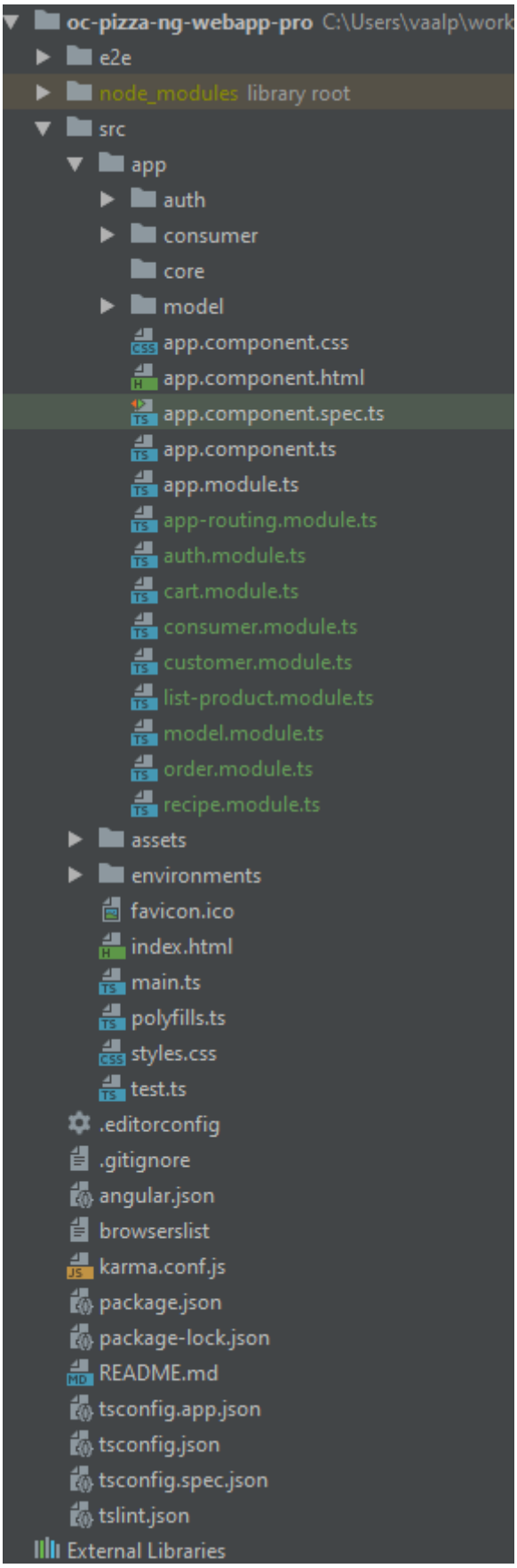
5.3.2 - Structure du Webservice REST



5.3.3 - Structure de l'application Angular cliente



5.3.4 - Structure de l'application Angular professionnelle



6 - Points particuliers

6.1 - Gestion des logs

Les logs sont externalisés dans un fichier dont le *path* est défini dans application.properties (cf 6.2.1.2).

6.2 - Fichiers de configuration

6.2.1 - Application Webservice REST

L'application Webservice REST possède un fichier application.properties situé dans {project-folder}\webservice-rest\src\main\resources.

6.2.1.1 - Datasources (interne au fichier application.properties)

datasource.driverClassName=org.postgresql.Driver

datasource.url=postgresql://oc-system:change-it@db-ocpizza-postgresql.services.clever-cloud.com:5432/db-ocpizza

datasource.username=oc-system

datasource.password=change-it

6.2.1.2 - Fichier application.properties

#Embedded tomcat server port

server.port=8080

Logs settings

logging.level.fr.ocpizza.web=INFO

logging.file=/C:/env/logs/oc-webservice-rest.log

logging.level.org.springframework=INFO

Custom properties defining the URL of the single-page applications

spa.url.client= https://ocpizza,firebaseapp.com

spa.url.pro= https://ocpizza-pro,firebaseapp.com

6.2.2 - Application Spring MVC

L'application web possède un fichier application.properties situé dans {project-folder}\webapp-spring-mvc\src\main\resources.

6.2.2.1 - Fichier application.properties

#Embedded tomcat server port
server.port=8080

# Logs settings
logging.level.fr.ocpizza.web=INFO
logging.file=/C:/env/logs/oc-webapp.log
logging.level.org.springframework=INFO

# i18n for internationalisation
spring.messages.basename=messages
spring.messages.encoding=UTF-8

6.2.3 - Applications Angular cliente et professionnelle

6.2.3.1 - Fichier environment.prod.ts

Nous avons le fichier d'environnement : **environment.prod.ts** dont le contenu est :

```
export const environment = {
  production: true,
  apiLocation : 'api.ocpizza.com',
  logOutput : 'console'
};

```

6.3 - Environnement de développement

Intellij 2019.1 pour développer le Webservice REST et la Webapp.

Webstrom 2019.1 pour développer les deux applications Angular.

6.4 - Procédure de packaging / livraison

Utilisez la commande Maven suivante : **mvn package** pour packager le Webservice REST et la Webapp.

Utilisez la commande : **ng build –prod** pour build les applcations Angular.