OC Pizza

Projet 08

Dossier de conception technique

Version 1.0

Auteur
Laurent Cordier
Développeur d'application

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Laurent Cordier	24/12/2019	Création du document	0.1
Laurent Cordier	01/01/2020	Version livrable	1.0

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application OC Pizza portant sur la mise en place d'un nouveau système informatique pour l'ensemble des pizzerias du groupe.

2.2 - Références

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

1. **DCF – Projet08** : Dossier de conception fonctionnelle de l'application.
2. **DEX – Projet08** : Dossier d'exploitation.

3 - ARCHITECTURE TECHNIQUE

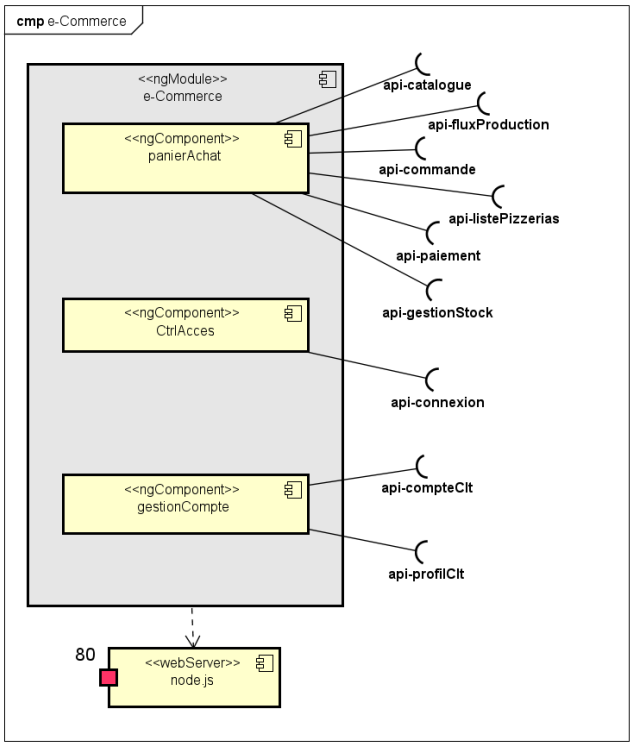
3.1 - Application Web e-Commerce

3.1.1 - La pile logicielle

La pile logicielle est la suivante :

- Angular : 8.2.14, Angular/Material : 8.2.3, Node.js : 12.14.0 LTS, Typescript : 3.5.3

3.1.2 - Diagramme UML de Composants



3.1.3 - Composant e-Commerce

Le composant e-Commerce représente un module au sens Angular qui est servi par le composant UML node.js sur le port 80.

Note : le fonctionnement d'une application Angular et du serveur node.js ne sont pas décrits ici.

3.1.4 - Composant ctrlAccès

Le composant ctrlAccès représente un service et un composant, au sens Angular. Ce composant utilise l'interface api-connexion exposée par le service Frontal-Web. Il produit les fonctions suivantes :

- Masque de saisie pour s'identifier avec contrôle des champs obligatoires,
- Bouton d'action pour se déconnecter.

3.1.5 - Composant panierAchat

Le composant panierAchat représente un service et un composant, au sens Angular. Il utilise les apis

- api-catalogue pour obtenir le catalogue des recettes disponibles de cette pizzeria,
- api-paiement pour transmettre le montant et les informations sur le moyen de paiement,
- api-gestionStock pour diminuer le stock des ingrédients de cette recette de cette commande, pour cette pizzeria
- api-commande pour enregistrer la commande à produire par cette pizzeria,
- api-listePizzerias pour obtenir la liste des pizzerias du groupe OC Pizza,
- api-fluxProduction pour enregistrer la commande dans le flux de production de cette pizzeria,

exposées par le service Frontal-Web pour transmettre ou recevoir les données. Il produit les fonctions suivantes :

- Confirmation de commande en affichant les boutons d'action de confirmation ou d'annulation.
- Choisir le mode paiement en proposant la liste déroulante :
  - Par chèque à la livraison,
  - Par CB.
- Pouvoir annuler la Commande tant qu'elle n'est pas en production en activant le bouton d'action « annulCmde ».
- Consulter le catalogue Groupe OC Pizza en affichant la liste des recettes disponibles avec case de saisie de nombre de pizza en regard de chaque ligne.
- Choisir une pizzeria en affichant la liste des pizzerias du groupe avec un radio bouton en regard de chaque ligne reçue.
- Créer un Panier d'achat en affichant un bouton dont l'activation place les recettes dont le nombre de pizzas est >0 dans le panier d'achat.

3.1.6 - Composant gestionCompte

Le composant gestionCompte représente un service et un composant, au sens Angular. Il utilise les apis :

- api-compteClt (nommage simplifié) pour créer un compte (api_compteCltPost) ou retrouver un compte (api_compteCltGet),
- api-profilClt (nommage simplifié) pour retrouver un profil (api_profilCltGet), modifier un profil (api_profilCltPut), créer un profil (api_profilCltPost),

exposées par le service Frontal-Web pour transmettre ou recevoir les données. Il produit les fonctions suivantes :

- Créer un compte client en proposant un masque de saisie des informations requises pour la création d'un compte client.
- Gérer ou créer ou retrouver le profil client en affichant les informations modifiables du profil client ainsi que les boutons d'action de confirmation ou annulation.

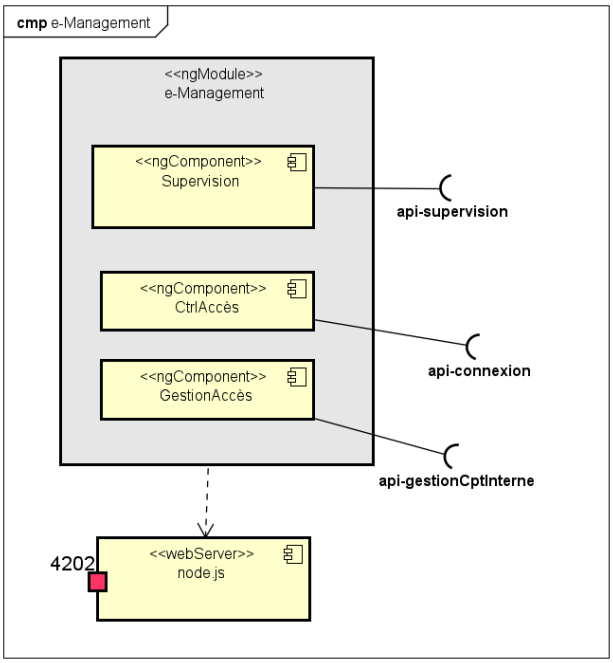
3.2 - Application Web e-Management

3.2.1 - La pile logicielle

La pile logicielle est la suivante :

- Angular : 8.2.14, Angular/Material : 8.2.3, Node.js : 12.13.0, Typescript : 3.5.3

3.2.2 - Diagramme UML de Composants



3.2.3 - Composant e-Management

Le composant e-Management représente un module au sens Angular qui est servi par le composant UML node.js sur le port 4202.

Note : le fonctionnement d'une application Angular et du serveur node.js ne sont pas décrits ici.

3.2.4 - Composant Supervision

Le composant Supervision représente un service et un composant, au sens Angular. Il utilise l'api api-supervision exposée par le service Frontal-Web. Il produit les fonctions suivantes :

- Visualiser les indicateurs d'activité de la pizzeria en affichant les données sous forme de tableau, issues du journal des commandes.

- Choisir une pizzeria pour en consulter l'activité en affichant la liste des pizzerias – disponible pour acteur Direction uniquement.
- Visualiser les indicateurs d'activité du groupe en affichant les données sous forme de tableau, issues de la consolidation des journaux des commandes des pizzerias.

3.2.5 - Composant GestionAccès

Le composant Supervision représente un service et un composant, au sens Angular. Il utilise l'api api-gestionCptInterne exposée par le service Frontal-Web pour transmettre ou recevoir les données. Il produit la fonction suivante :

- Demander ouverture/renouvellement compte collaborateur en proposant le masque de saisie des informations requises pour la création d'un compte collaborateur.

3.2.6 - Composant CtrlAccès

Le composant CtrlAccès représente un service et un composant, au sens Angular. Il utilise l'api api-connexion exposées par le service Frontal- Web pour transmettre ou recevoir les données. Il produit les fonctions suivantes :

- Masque de saisie pour s'identifier avec contrôle des champs obligatoires
- Bouton d'action pour se déconnecter

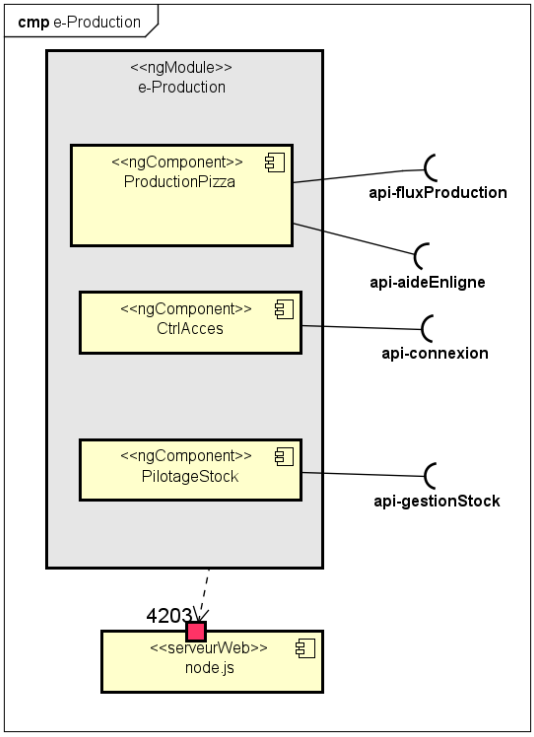
3.3 - Application Web e-Production

3.3.1 - La pile logicielle

La pile logicielle est la suivante :

- Angular : 8.2.14, Angular/Material : 8.2.3, Node.js : 12.13.0, Typescript : 3.5.3

3.3.2 - Diagramme UML de Composants



3.3.3 - Composant e-Production

Le composant e-Production représente un module au sens Angular qui est servi par le composant UML node.js sur le port 4203.

Note : le fonctionnement d'une application Angular et du serveur node.js ne sont pas décrits ici.

3.3.4 - Composant ProductionPizza

Le composant ProductionPizza représente un service et un composant, au sens Angular. Il utilise les apis :

- api-aideEnLigne pour recevoir la liste des ingrédients d'une recette,
- api-fluxProduction pour changer l'état de la commande selon l'étape de son traitement,

exposées par le service Frontal- Web pour transmettre ou recevoir les données. Il produit les fonctions suivantes :

- Liste des pizzas à livrer sous forme de tableau avec bouton radio en regard de chaque ligne,
- Liste des pizzas à produire sous forme de tableau avec bouton radio en regard de chaque ligne,

3.3.5 - Composant CtrlAccès

Le composant CtrlAccès représente un service et un composant, au sens Angular. Il utilise l'api api-connexion exposées par le service Frontal- Web pour transmettre ou recevoir les données. Il produit les fonctions suivantes :

- Masque de saisie pour s'identifier avec contrôle des champs obligatoires
- Bouton d'action pour se déconnecter

3.3.6 - Composant GestionStock

Le composant GestionStock représente un service et un composant, au sens Angular. Il utilise l'api

- api-gestionStock pour créer ou augmenter le stock d'un ingrédient, pour cette pizzeria,

exposée par le service Frontal- Web pour transmettre ou recevoir les données. Il produit les fonctions suivantes :

- Liste des ingrédients de quantité inférieur au seuil avec liste déroulante de fournisseur et champ de saisie de quantité libellé avec l'unité (Kg, litre…)
- Bouton d'action pour confirmer l'impression des bons de commandes.

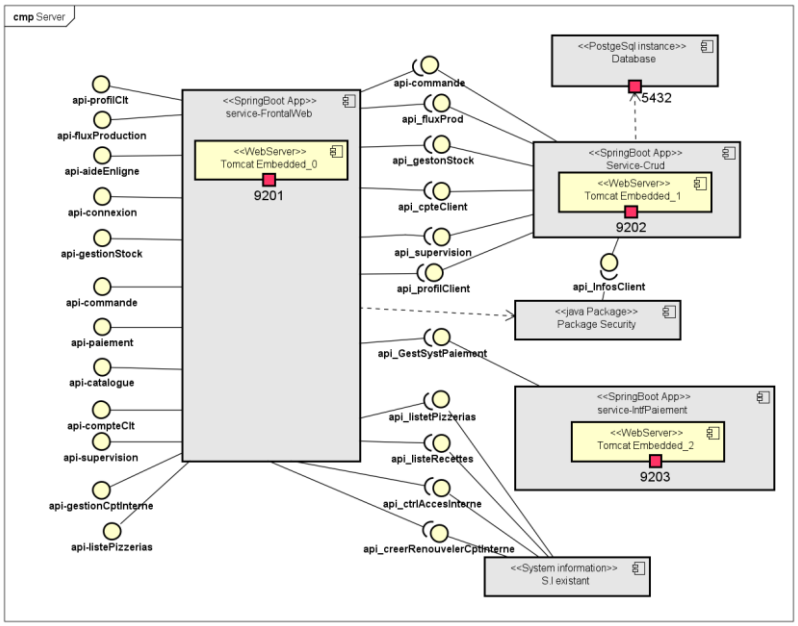
3.4 - Application Serveur

3.4.1 - La pile logicielle

La pile logicielle est la suivante :

- SpringBoot 2.2.0 Release
- JDK version 11.04,
- PostgreSQL 12.1,
- Tomcat Embedded 9.0.27
- Hibernate 5.4.6 Final
- Spring Security 5.2.0 Release

3.4.1.1 - Diagramme UML de Composants



3.4.2 - Composant Service-FrontalWeb

Le composant Service-FrontalWeb représente un service hébergé par le gestionnaire de container Tomcat Embedded, au sens Spring Boot, accessible via le port TCP : 9201.

Il réalise les apis :

- api-fluxProduction qui transmet/reçoit les données du/au Composant Service-Crud pour la pizzeria dont l'idPizzeria est passé en paramètre de la requête.
- api-aideEnligne qui retrouve la recette dans le paramètre de la requête et renvoie l'aide en ligne reçu du composant S.I Existant.
- api-profilClt qui consomme l'api api_profilClient du composant Service-Crud- voir le composant Service-Crud pour le détail de cette api (GET/POST/PUT).
- api-compteClt qui consomme l'api api_cpteClient du composant Service-Crud pour retrouver ou créer un compte client - voir le composant Service-Crud pour le détail de cette api (GET/POST).
- api-connexion qui selon l'attribut de requête
    - route les informations de connexions vers le composant S.I Existant s'il s'agit d'un collaborateur,
    - utilise les méthodes du composant package security pour identifier/authentifier un compte client,
- api-gestionStock qui transmet/reçoit les données du/au composant Service-Crud - voir le

composant Service-Crud pour le détail de cette api (GET/POST/PUT).

- api-commande qui transmet/reçoit les données du/au Composant Service-Crud.
- api-paiement qui transmet/reçoit les données du/au Composant Service-IntfPaiement.
- api-listePizzerias qui consomme l'api api_listePizzerias du composant S.I existant.
- api-catalogue qui
    - reçoit le stock de la pizzeria dont l'id est passé dans l'attribut de requête en consommant l'api api_gestionStock du composant Service-Crud
    - reçoit la liste des recettes du groupe en consommant l'api api_listeRecettes du composant S.I existant
    - renvoie le catalogue pizzeria contenant les recettes dont les ingrédients sont en stock dans cette pizzeria
- api-supervision qui renvoie les données reçues du composant Service-Crud
    - pour une pizzeria si l'attribut idPizz est valorisé,
    - pour le groupe, en consolidant l'ensemble du journal.
- api-gestionCptInterne qui consomme l'api api_creerRenouvelerCptInterne du composant S.I Existant pour une demande d'ouverture ou de renouvellement de compte interne collaborateur.

3.4.3 - Composant Service-Crud

Le composant Service-Crud représente un service hébergé par le gestionnaire de container Tomcat Embedded, au sens Spring Boot, accessible via le port TCP : 9202.

Ce composant effectue les requêtes CRUD sur le composant Database.

Il réalise les apis :

- api_profilClient, table ProfilClient, (dénomination simplifiée) qui se décompose en
    - api_profilClientGet pour obtenir le profil Client dont le nom est passé en attribut de la requête.
    - api_profilClientPut pour modifier le profil Client dont le nom est passé en attribut de la requête.
    - api_profilClientPost pour enregistrer un nouveau profil client.
- api_cptClient, table CompteAcheteur, (dénomination simplifiée) qui se décompose en
    - api_cptClientGet pour obtenir les informations du compte client dont le courriel est passé en attribut de la requête.
    - api_cptClientPost pour enregistrer un nouveau compte client.

- api_gestionStock, table Stock Ingrédient, (dénomination simplifiée) qui se décompose en
    - api_gestionStockGet pour obtenir la liste des ingrédients avec leur quantité en stock d'une pizzeria dont l'id est passé en paramètre de la requête.
    - api_gestionStockPut pour mettre à jour le stock d'un ingrédient ; le corps de la requête contenant :
        - l'idIngredient de l'ingrédient à mettre à jour,
        - l'idPizzeria de la pizzeria concernée,
        - la quantité.
    - api_gestionStockPost pour créer le stock d'un ingrédient le corps de la requête contenant :
        - l'idIngredient de l'ingrédient,
        - l'idPizzeria de la pizzeria concernée,
        - la quantité.
- api-commande, tables Commande, PanierAchat, LignePanier, journal,
    - ajoute un enregistrement dans les tables commande et PanierAchat,
    - ajoute autant d'enregistrement dans la table LignePanier qu'il y a de lignes dans le panier d'achat,
    - ajoute un enregistrement dans la table journal.
- api_fluxProd, table FluxProduction, (dénomination simplifiée) qui se décompose en
    - api_fluxProdGet pour obtenir la liste des commandes ainsi que leur état courant, pour la pizzeria dont l'idPizzeria est passé en paramètre de la requête.
    - api_fluxProdPost pour mettre à jour l'état d'une commande.
- api_supervision pour obtenir le contenu du journal des commandes par pizzeria.
- api_InfosClient pour traiter les informations Client

3.4.4 - Composant Package Security

Ce composant gère le tockenId pour la session Web d'un utilisateur en contrôlant que le mot de passe fourni est celui mémorisé. Le mot de passe est enregistré sous forme d'une chaîne de caractères qui représente le résultat de la fonction de « hashage » de type SHA-1024, appliquée à ce mot de passe.

Au sein d'une session, il permet au composant Frontal-Web de connaître l'utilisateur connecté donc son compte et son profil.

Etant réalisé avec le Framework SpringSecurity, sa réalisation repose sur l'implémentation des

interfaces « UserDetail », « UserDetailService », « AuthenticationProvider » et de la dérivation de de la classe « WebSecurityConfigurerAdapter ».

3.4.5 - Composant S.I existant

Ce composant est hors périmètre – Il est présent ici pour la cohérence d'ensemble des documents.

Pour mémoire, ce composant expose au travers d'API, les fonctions suivantes :

- Liste des pizzerias,
- Liste des recettes du groupe,
- Contrôle d'accès à l'existant pour les collaborateurs,
- Ouverture/Fermeture session de travail des collaborateurs.

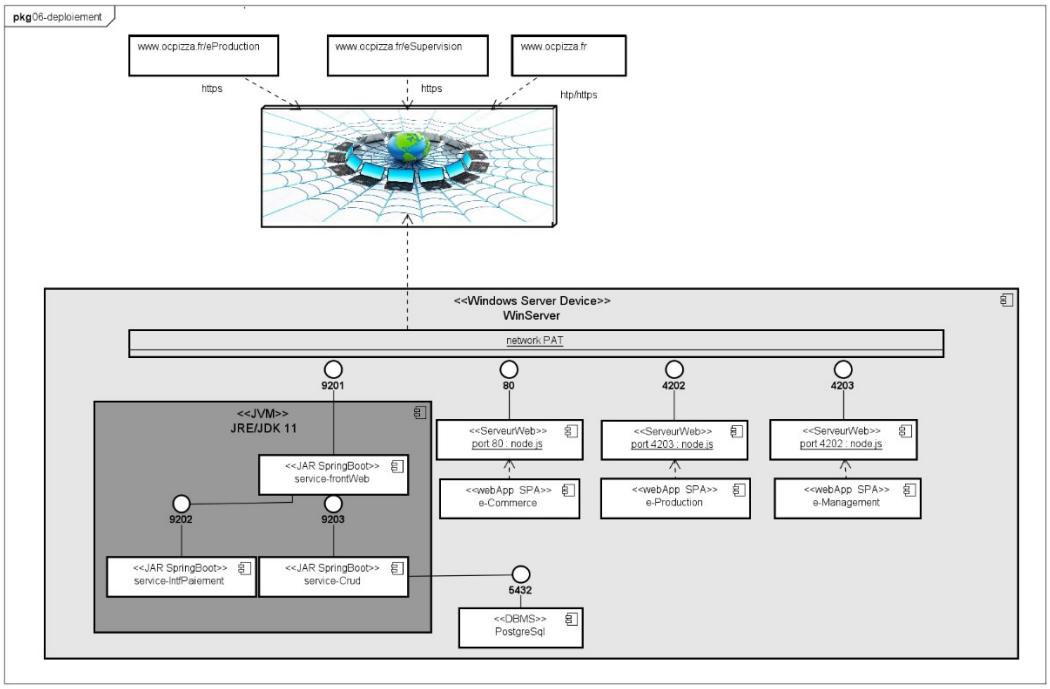
3.4.6 - Composant Service-IntfPaiement

Le composant Service- IntfPaiement représente un service hébergé par le gestionnaire de container Tomcat Embedded, au sens Spring Boot, accessible via le port TCP : 9203.

Ce composant gère la liaison avec le système de paiement en implémentant les API publiées par ce système. Les Api du système de paiement sont hors-périmètre de ce document.

- api_gestSystPaiement pour transmettre les données nécessaires au paiement d'une commande au service d'interface entre le système de paiement et l'application OC Pizza.

4 - ARCHITECTURE DE DEPLOIEMENT



4.1 - Serveur de Base de données

Sans objet – Le DBMS est hébergé par le serveur unique d'OC Pizza.

4.2 - Serveur d'application

Le serveur utilisé est celui de l'existant OC Pizza sous Windows server 2019

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

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

5.1.1 - Les couches des applications Angular

L'architecture applicative est la suivante :

- une couche **business** : responsable de la logique métier du composant
- une couche **model** : implémentation du modèle des objets métiers

5.1.2 - Les couches des applications Serveur

L'architecture applicative est la suivante :

- une couche **business** : responsable de la logique métier du composant.
- une couche **model** : implémentation du modèle des objets métiers.
- une couche **view** : implémentation des entry point représentés par un contrôleur pour chaque API exposée.

5.1.3 - Les modules des applications Angular

Sans objet.

5.1.4 - Les modules des applications Serveur

Chaque application est un module Maven. En revanche, une application est mono-module au sens Java (version > V9).

5.1.5 - Structure des sources des applications Angular

La structure d'une application Angular respecte les règles d'architecture mises en œuvre par le générateur d'application Angular/Cli. Se reporter à la documentation officielle Angular/Cli pour plus de précisions.

5.1.6 - Structure des sources des applications Serveur

Pour chaque application, la structuration générale des répertoires `optionnels` et **`obligatoires`** du projet suit la logique suivante :

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

```
OC Pizza
  ├─ pom.xml
  ├─ <nom du module maven> (ex : Service-FrontalWeb)
  │    ├─ pom.xml
```

```
└── src
    ├── main
    │   ├── java
    │   │   └── ocpizza
    │   │       ├── controller
    │   │       ├── model
    │   │       ├── service
    │   │       ├── authication
    │   │       ├── configuration
    │   │       └── utility
    │   └── resources
    └── test
        ├── java
        └── resources
```

Les fichiers source qui constituent la couche :

- business sont placés dans le répertoire/package `service`.
- model sont placés dans le répertoire/package `model`.
- view sont placés dans le répertoire/package `controller`.

Les fichiers source qui contiennent

- les tests unitaires sont placés dans le répertoire test/java
- la configuration spring et spring security sont placés dans le répertoire `configuration`.
- les règles spring security applicables à l'identification et authentification de l'utilisateur sont placés dans le répertoire `authication`.
- les outils communs à tous les sources de ce module maven sont placés dans le répertoire `utility`.

Les fichiers de configuration

- que sont applications.yml, application.properties et logback-spring.xml sont placés dans le répertoire `resources` et une copie est aussi placée dans `test/resources`.

Note : Toutes les applications n'ont pas nécessairement besoin de l'ensemble de ces répertoires (ex : Service-FrontalWeb n'a pas de fichiers en lien avec spring security – il n'a donc pas les répertoires qui s'y rapportent).

5.2 - Applications Web

L'application Web respecte la structure suivante :

Webapp_xxx

----------→ fichiers de configurations Angular, TypeScript et npm.

----------→src/

----------→ fichiers de base html et typescript

----------→ environnements/

----------→ fichiers pour distinguer la version de développement de celle de production.

----------→ assets/

----------→ images et icones

----------→app/

----------→ fichiers principaux de l'application (du module au sens Angular) :

----------→ xxxx/

----------→ fichiers composants et services du composant xxxx.

6 - POINTS PARTICULIERS

6.1 - Gestion des logs

Chaque application Java embarque la librairie Slf4j /Logback. A la racine de chaque application se trouve le répertoire logs qui contient un fichier du nom de l'application suffixé par « .log ».

6.2 - Fichiers de configuration

6.2.1 - Application web

Aucun – il s'agit d'une application Angular/node packagée non modifiable. Le choix du port d'écoute est effectué en ligne de commande lors du lancement de l'application.

6.2.1.1 - Datasource

Le datasource (paramètres JDBC et paramètres JPA/Hibernate) est défini dans le fichier de configuration du service-crud, main/resources/application.yml.

Ce fichier n'est pas modifiable par l'utilisateur.

6.2.1.2 - Fichier de configuration des Logs

Il s'agit du fichier logback-spring.xml, présent dans le répertoire main/resources de chaque application java. Ce fichier n'est pas modifiable par l'utilisateur.

6.2.2 - Applications serveur

Les fichiers de configuration sont application. properties, application.yml, dans main/resources de chaque application java.

Application.properties contient le paramétrage springtools ainsi que les éléments de configuration applicatif de chaque application java.

Application.yml contient le n° de port sur lequel Tomcat Embedded écoute. Pour l'application Service-Crud y sont définis le datasource et le comportement JPA/Hibernate.

Ces fichiers ne sont pas modifiables par l'utilisateur.

6.3 - Ressources

Sans-objet

6.4 - Environnement de développement

Pour ce qui concerne les développements java : **JetBrains IntelliJ 2019.3.1 ultimate**

Pour ce qui concerne le SGBD : **PgAdmin4**, psql et **JetBrains IntelliJ 2019.3.1 ultimate**

Pour ce qui concerne les développements Angular/TypeScript : **JetBrains IntelliJ 2019.3.1 ultimate**

Pour ce qui concerne les tests des développements Angular/TypeScript : navigateur avec moteur chromium (Edge ou chrome) avec l'extension **JetBrains IDE Support** installée.

6.5 - Procédure de packaging / livraison

Pour ce qui concerne les développements java : fichier JAR complet.

Pour ce qui concerne les développements Angular/TypeScript, il s'agit d'un fichier zip contenant le répertoire /webapp_xxx contenant les sources complets. Ici, xxx est valorisé par le nom de la webapp (e-Commerce, …).

7 - GLOSSAIRE

Angular	Framework de développement d'application web en typescript (et orienté asynchrone – Ajax)
SPA	Single page application
CRUD	Create Read Update et Delete : opérations usuelles sur une base de données
Spring Boot	Framework Spring produisant les fichiers Jar complet et gérant les dépendances des librairies qu'il utilise.