|  |
| --- |
| **OC Pizza**  **Projet 08**  Dossier de conception technique  Version 1.0 |
| **Auteur**  Laurent Cordier  *Développeur d'application* |

Table des matières

1 - Versions 4

2 - Introduction 5

2.1 - Objet du document 5

2.2 - Références 5

3 - Architecture Technique 6

3.1 - Application Web e-Commerce 6

3.1.1 - La pile logicielle 6

3.1.2 - Diagramme UML de Composants 6

3.1.3 - Composant e-Commerce 6

3.1.4 - Composant ctrlAccès 6

3.1.5 - Composant panierAchat 7

3.1.6 - Composant gestionCompte 7

3.2 - Application Web e-Management 7

3.2.1 - La pile logicielle 7

3.2.2 - Diagramme UML de Composants 8

3.2.3 - Composant e-Management 8

3.2.4 - Composant Supervision 8

3.2.5 - Composant GestionAccès 9

3.2.6 - Composant CtrlAccès 9

3.3 - Application Web e-Production 10

3.3.1 - La pile logicielle 10

3.3.2 - Diagramme UML de Composants 10

3.3.3 - Composant e-Production 10

3.3.4 - Composant ProductionPizza 10

3.3.5 - Composant CtrlAccès 11

3.3.6 - Composant GestionStock 11

3.4 - Application Serveur 11

3.4.1 - La pile logicielle 11

3.4.1.1 - Diagramme UML de Composants 12

3.4.2 - Composant Service-FrontalWeb 12

3.4.3 - Composant Service-Crud 13

3.4.4 - Composant Package Security 13

3.4.5 - Composant S.I existant 13

3.4.6 - Composant Service-IntfPaiement 13

4 - Architecture de Déploiement 14

4.1 - Serveur de Base de données 14

4.2 - Serveur XXX 14

5 - Architecture logicielle 15

5.1 - Principes généraux 15

5.1.1 - Les couches 15

5.1.2 - Les modules 15

5.1.3 - Structure des sources 16

5.2 - Application Web 17

5.3 - Application Xxx 17

6 - Points particuliers 18

6.1 - Gestion des logs 18

6.2 - Fichiers de configuration 18

6.2.1 - Application web 18

6.2.1.1 - Datasources 18

6.2.1.2 - Fichier xxx.yyy 18

6.2.2 - Application Xxx 18

6.3 - Ressources 18

6.4 - Environnement de développement 18

6.5 - Procédure de packaging / livraison 18

6.6 - XXX 18

7 - Glossaire 20

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Laurent Cordier | 19/12/2019 | Création du document | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## 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.

## 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.

# Architecture Technique

## Application Web e-Commerce

### 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

### Diagramme UML de Composants

Une image contenant capture d’écran

Description générée automatiquement

### 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 4201.

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

### 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.

### Composant panierAchat

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

* api-catalogue,
* api-paiement,
* api-gestionStock,
* api-commande,

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.

### 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 son compte client en affichant les informations modifiables du compte client en mode « modification autorisée » ainsi que les boutons d’action de confirmation ou annulation.

## Application Web e-Management

### 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

### Diagramme UML de Composants

Une image contenant capture d’écran

Description générée automatiquement

### 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.

### 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.

### 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.

### 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

## Application Web e-Production

### 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

### Diagramme UML de Composants

Une image contenant capture d’écran

Description générée automatiquement

### 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.

### Composant ProductionPizza

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

* api-aideEnLigne,
* api-fluxProduction,

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,

### 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

### Composant GestionStock

Le composant GestionStock représente un service et un composant, au sens Angular. Il utilise l’api api-gestionStock exposées 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’envoi des bons de commandes.

## Application Serveur

### La pile logicielle

La pile logicielle est la suivante :

* SpringBoot 2.2.0 Release
* JDK version 11.04,
* PostgreSQL 12.1,
* [Tomcat Embedded](https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-core) [9.0.27](https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-core/9.0.27)
* Hibernate 5.4.6 Final
* Spring Security 5.2.0 Release

#### Diagramme UML de Composants

Une image contenant carte, capture d’écran

Description générée automatiquement

### Composant Service-FrontalWeb

Le composant Service-FrontalWeb représente un service hébergé par le gestionnaire de container Tomcat Embedded, au sens Spring Boot.

Il réalise les apis :

* api-fluxProduction qui transmet/reçoit les données du/au Composant Service-Crud.
* api-aideEnligne qui retrouve la recette dans l’attribut 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 pour créer/modifier un profil client
* api-compteClt qui consomme l’api api\_cpteClient du composant Service-Crud pour créer un compte client.
* 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.
* 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.

### Composant Service-Crud

Ce composant effectue les requêtes CRUD sur le composant Database.

Il réalise les apis :

* api\_CptClient (dénomination simplifiée) qui se décompose en
  + api\_CptClientGet pour obtenir les informations Client dont le courriel est passé en attribut de la requête.
  + api\_CptClientPost pour enregistrer un nouveau compte client ou modifier un existant.
* api\_Stock (dénomination simplifiée) qui se décompose en
  + api\_StockGet pour obtenir la liste des ingrédients par pizzeria avec leur quantité en stock
  + api\_StockPut pour mettre à jour le stock d’un ingrédient, par pizzeria.
* api\_FluxProd (dénomination simplifiée) qui se décompose en
  + api\_FluxProdGet pour obtenir la liste des commandes par pizzerias ainsi que leur état courant,
  + api\_FluxProdPost pour mettre à jour la liste des commandes par pizzerias,
* api\_Supervision pour obtenir le contenu du journal des commandes par pizzeria.
* api\_InfosClient pour traiter les informations Client
* en paramètre de la requête :
  + Profil : traite le profil client (création/modification/lecture)
  + Compte : renvoie

### Composant Package Security

Ce composant gére le tocken Id pour la session Web d’un utilisateur en contrôlant

TODO Description et rôle/objectif...

### Composant S.I existant

TODO Description et rôle/objectif...

* En lien avec le S.I Existant que sont : api\_LstPizzeria, api\_LstRecettes et api\_CtrlAccessInterne pour recevoir ou transmettre les informations relatives à :
  + Liste des pizzerias,
  + Liste des recettes du groupe,
  + Ouverture/Fermeture session de travail des collaborateurs.

### Composant Service-IntfPaiement

Description et rôle/objectif...

* api\_GestSystPaiement pour transmettre les données nécessaires au paiement d’une commande au service d’interface entre le système de paiement et l’application OCPizza.

TODO

# 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** pour les sources Java et par **npm** pour les sources TypeScript.

### Les couches des applications Angular TODO

L'architecture applicative est la suivante :

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

### Les couches des applications Serveur TODO

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 des applications Angular TODO

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

### Les modules des applications Serveur TODO

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

### Structure des sources des applications Angular

### Structure des sources des applications Serveur

La structuration des répertoires du projet suit la logique suivante :

* les répertoires sources sont créés 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

Uml component dans l’optique archi logicielle (mes services vu sous forme uml avec interdependances)

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

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