OC Pizza

Système de Gestion Informatique

Dossier de conception technique

Version 1.0.0

Auteur
Damien Gironnet
Analyste-Programmeur

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Damien Gironnet	13/10/2020	Création du document	1.0.0

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique du système de gestion de pizzérias pour le groupe OC-PIZZA.

L'objectif de ce document est de présenter aux développeurs l'architecture du système et ses composants. Le document contient notamment :

- La définition des différents composants du système et leurs interactions (partie 3).
- La description de la base de données relationnelle et de sa structure (section 4.2).
- Le diagramme de déploiement du système (section 4.3).
- La présentation de l'architecture logicielle (partie 5).
- Les ressources nécessaires à la phase de développement (section 6.3).

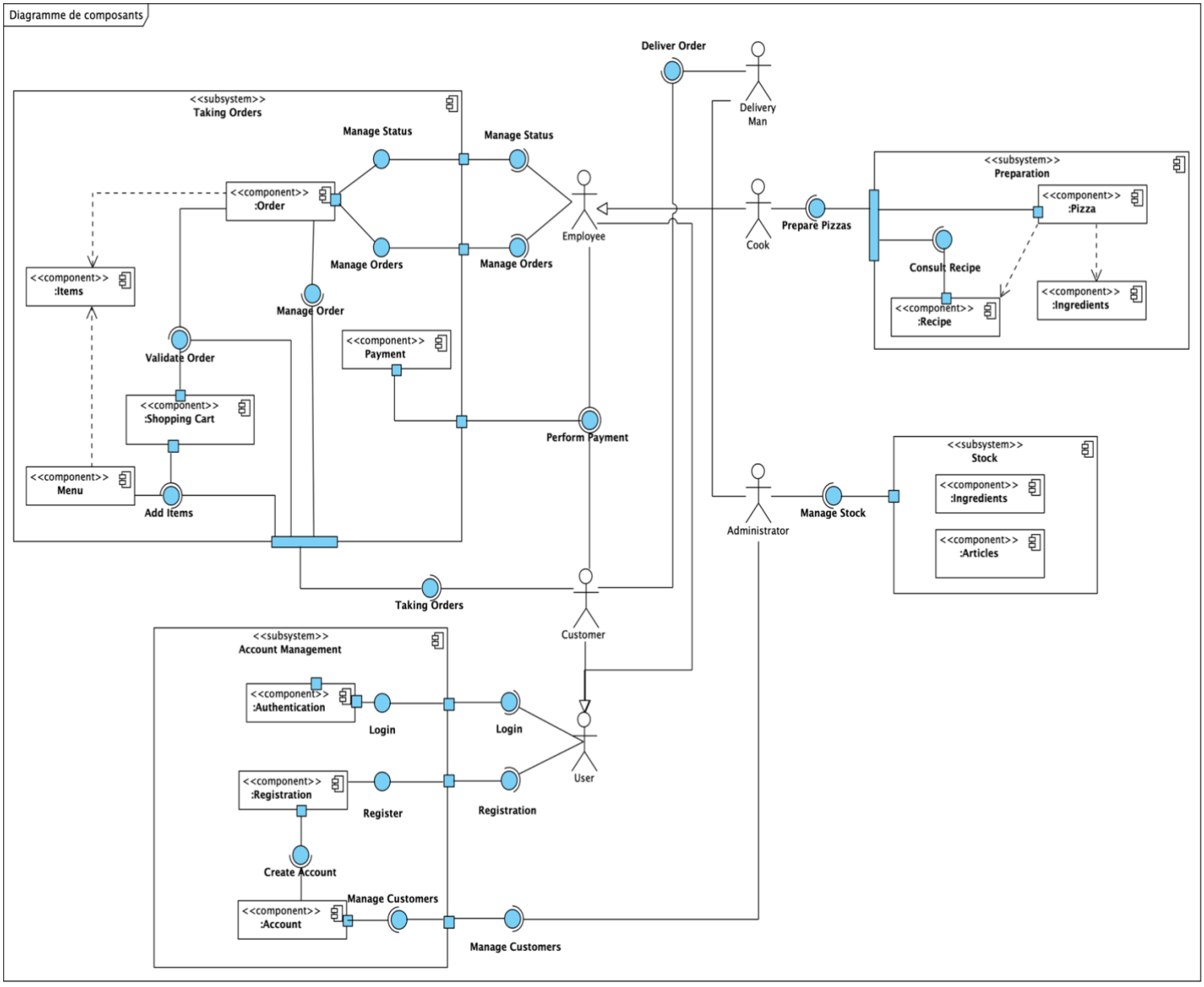
2.2 - Références

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

1. **DCF – 1.0.0** : Dossier de conception fonctionnel de l'application
2. **DE – 1.0.0** : Dossier d'exploitation de l'application
3. **PVL – 1.0.0** : Procès-verbal de livraison de l'application

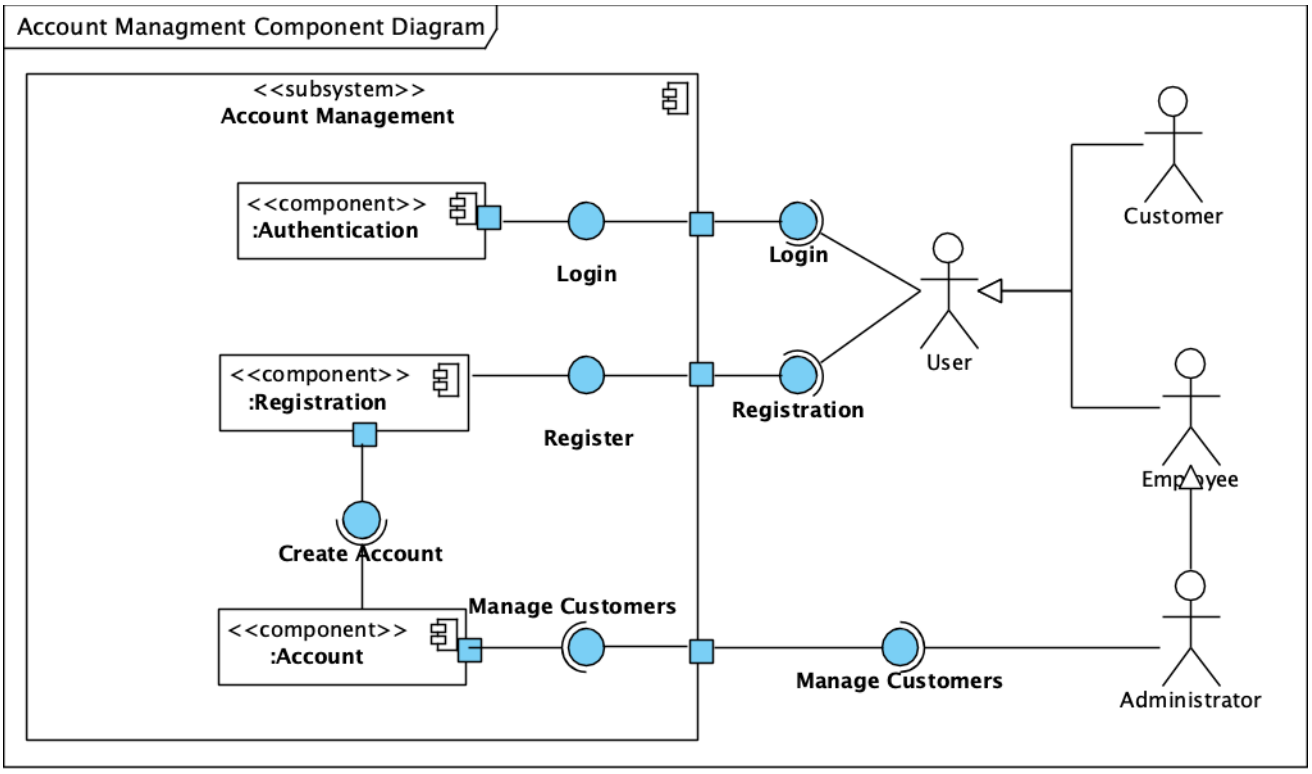
3 - Architecture Technique

3.1 - Composants généraux



3.1.1 - Subsystem Account Management

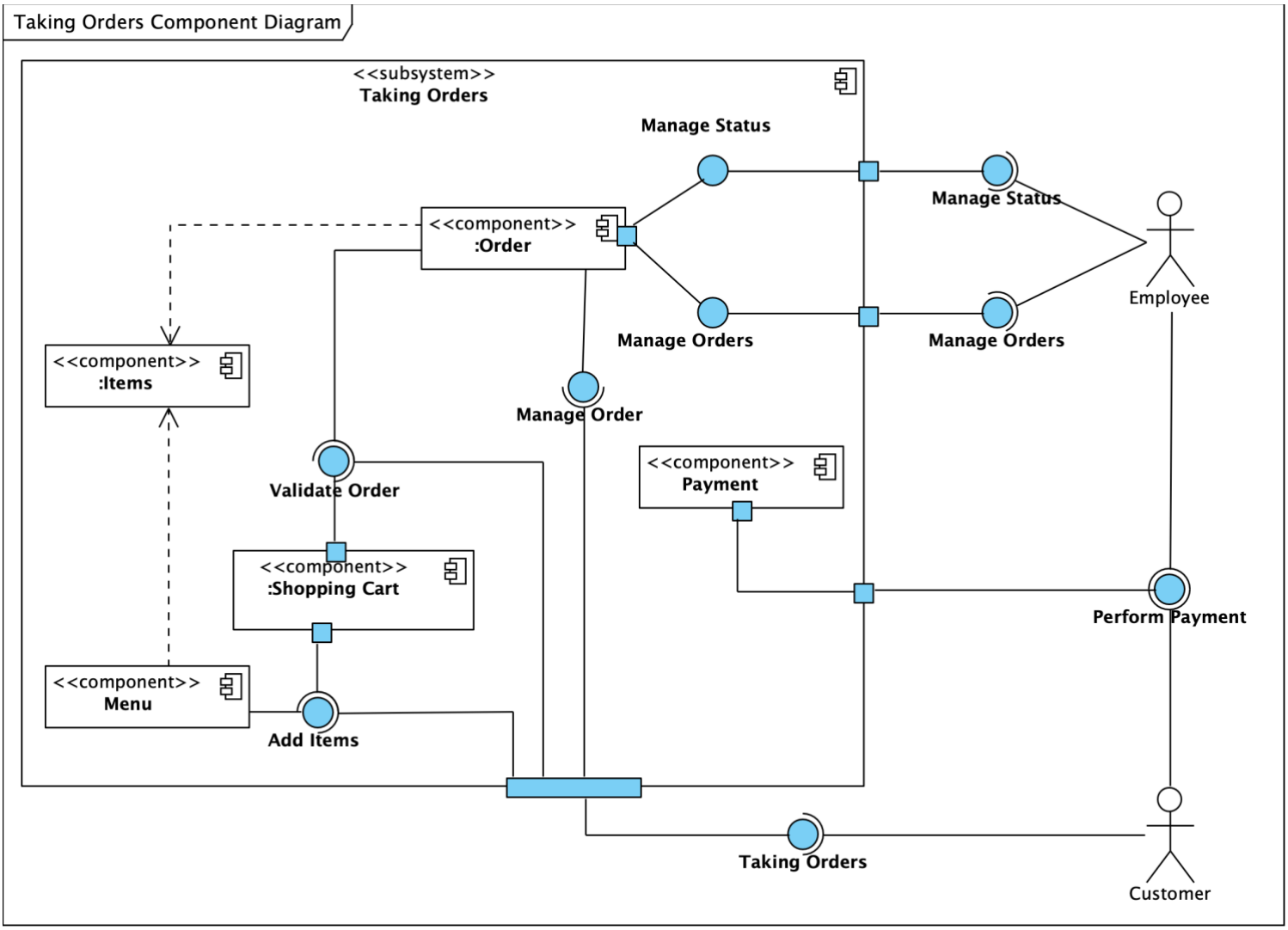
Le diagramme ci-dessous est la représentation des composants du sous-système Account Managment.



L'interface Login permet à un utilisateur de s'authentifier. Lorsque L'utilisateur n'est pas enregistré, il doit créer un compte. L'interface Create Account permet de créer un compte. L'interface Manage Customers permet à l'administrateur de gérer les comptes utilisateurs.

3.1.2 - Subsystem Taking Orders

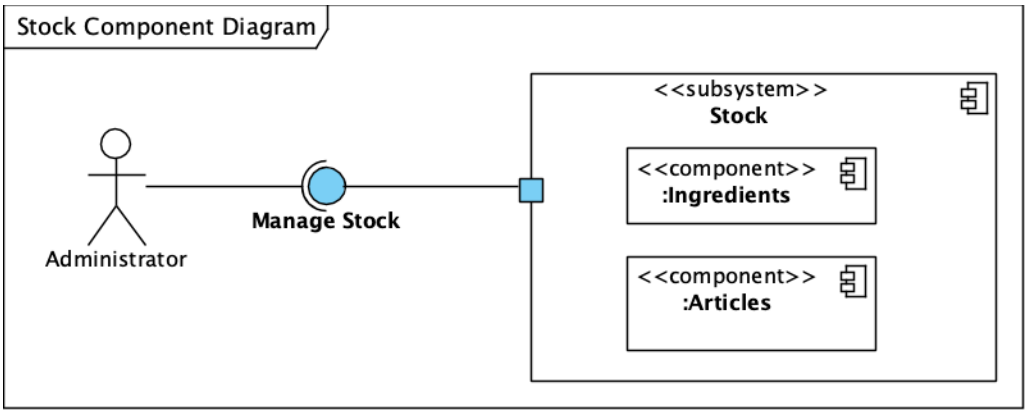
Le diagramme ci-dessous est la représentation des composants du sous-système Taking Orders.



L'interface Taking Orders permet au client de passer une commande. L'interface Add Items permet au client d'ajouter des produits à son panier représenté par le composant Shopping Cart. L'interface Validate Order permet au client de valider sa commande. L'interface Manage Order permet au client de modifier la commande. L'interface Manage Status permet aux employées de faire évoluer le statut de la commande tout au long de son cycle de vie. L'interface Perform Payment permet d'effectuer le paiement de la commande.

3.1.3 - Subsystem Stock

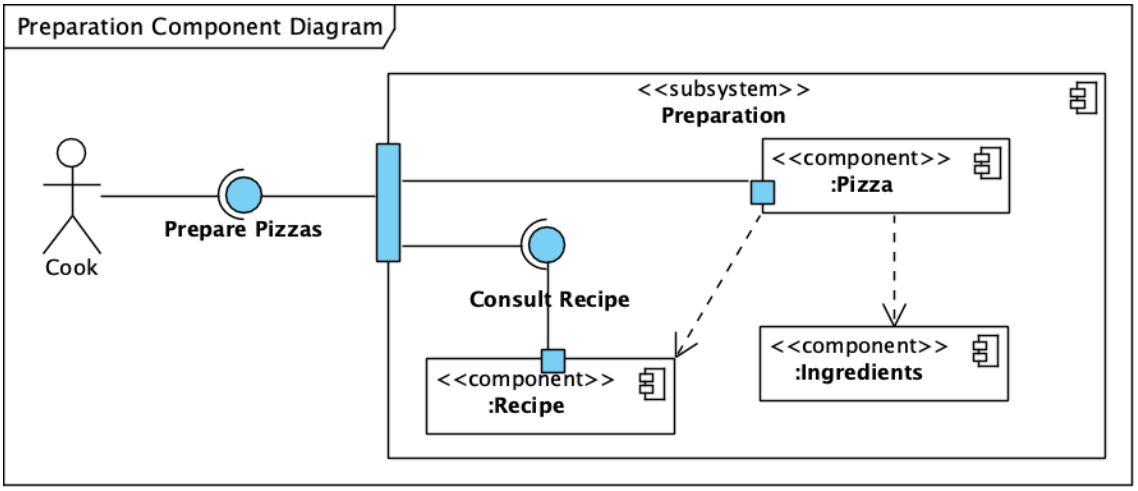
Le diagramme ci-dessous est la représentation des composants du sous-système Stock.



L'interface Manage Stock permet à l'administrateur de gérer le stock de la pizzeria dont il est responsable.

3.1.4 - Subsystem Preparation

Le diagramme ci-dessous est la représentation des composants du sous-système Preparation.



L'interface Prepare Pizzas est l'interface dédiée à la préparation des pizzas par le cuisinier. L'interface Consult Recipe permet au cuisinier de consulter la recette des pizzas qu'il prépare.

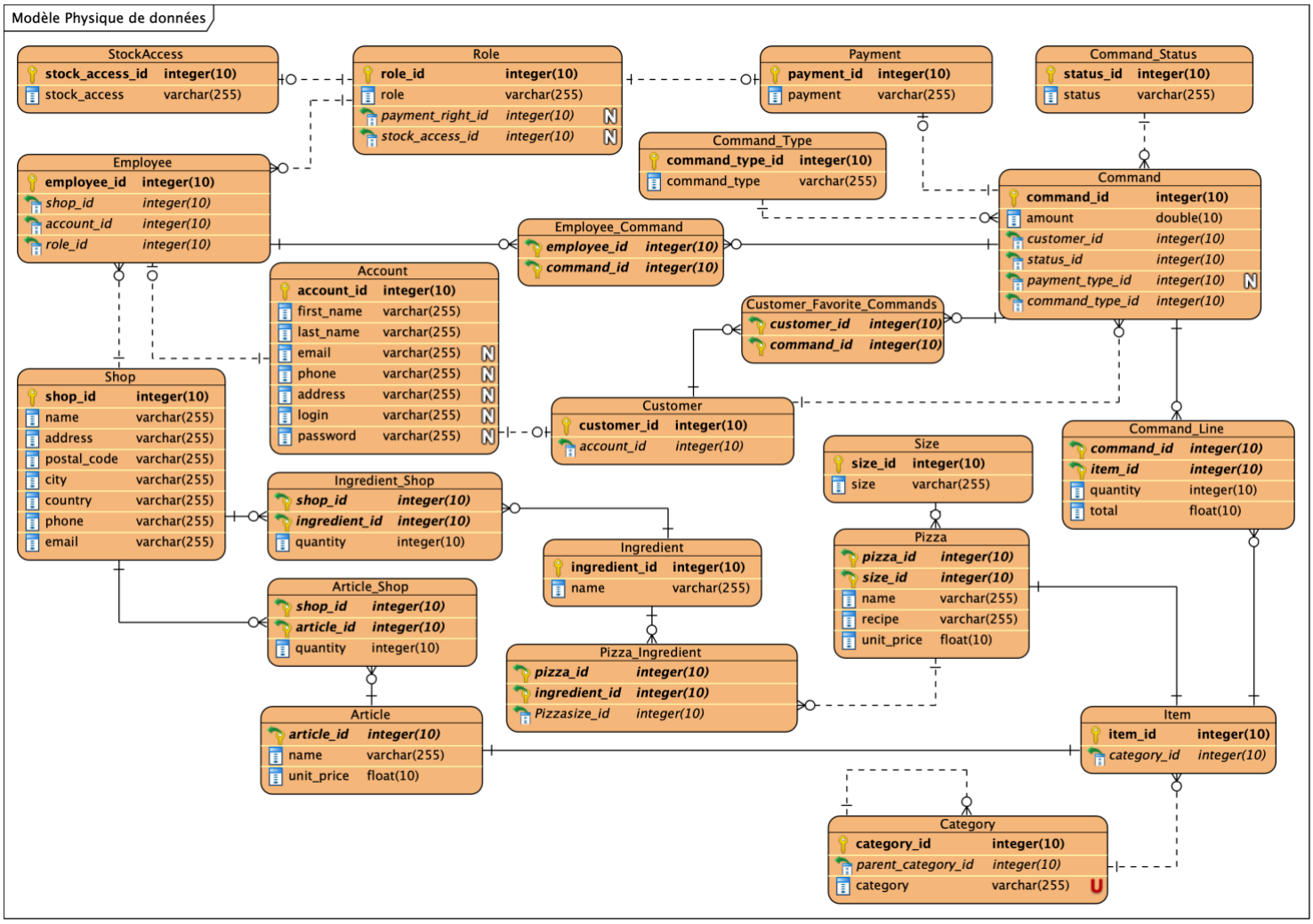
4 - ARCHITECTURE DE DÉPLOIEMENT

4.1 - Serveur de base de données

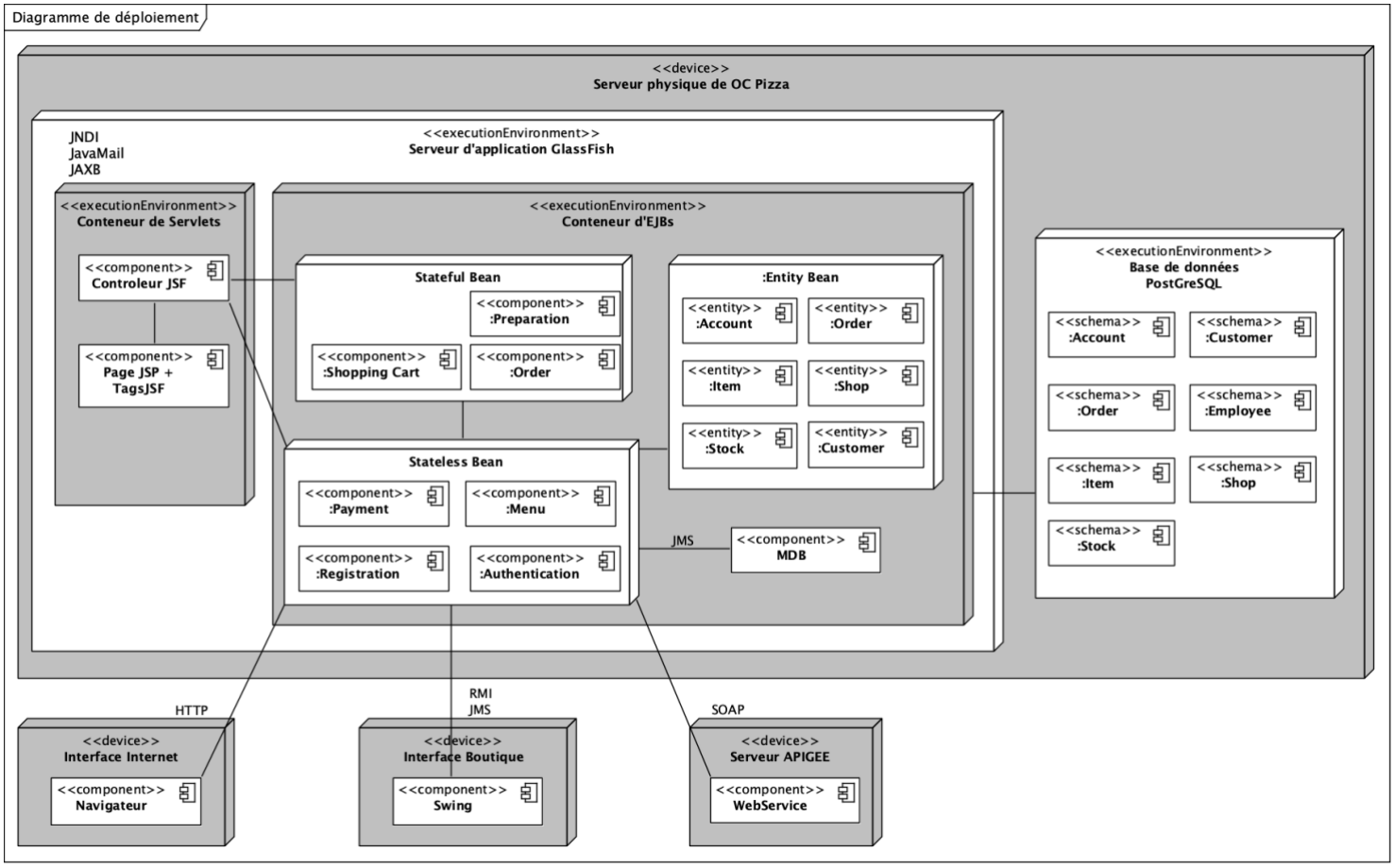
Le serveur de base de données sera un serveur hébergé dans les locaux techniques du siège d'OC-PIZZA de type PostgreSQL.

4.2 - Structure de la base de données

La base de données sera composée des tables suivantes, possédant les relations spécifiées sur le diagramme. Les scripts SQL de créations seront fournis pour la phase de déploiement.



4.3 - Déploiement



Serveur physique de OC Pizza

L'élément « Serveur physique de OC Pizza » est le nœud qui représente la structure de la partie back-end du système proposé pour la solution technique de ce document.
Il est composé d'un serveur d'application GlassFish et d'une base de données PostGreSQL.

Serveur d'application Glassfish

Un serveur GlassFish est un serveur d'application pouvant à la fois contenir un conteneur web et un conteneur métier de type EJB.

Conteneur d'EJBs

Le Conteneur EJB (Entreprise Java Bean) est l'élément qui gère la partie métier de l'application. Elle est composée de 4 parties:

- Stateful Bean
- Stateless Bean
- Message Driven Bean
- Entity Bean

Stateless Bean

Les EJB Stateless sont des EJB sans état qui ne fonctionne que de manière éphémère. Une fois qu'un client a demandé et reçu une fonctionnalité du composant, l'interaction avec ce composant prend fin, ne laissant aucune trace de ce qui s'est passé. L'utilisation standard d'un EJB sans état réside dans le fait qu'une application cliente le contacte en lui transmettant des paramètres. L'EJB accède alors à une base de données, effectue plusieurs traitements, appelle d'autres EJB, puis retransmet un résultat au client. C'est ce qui se produit pour les composants Payment, Menu, Registration et Authentication.

Stateful Bean

Par opposition au composant sans état, les EJB stateful associent les requêtes à un client spécifique, unissant client et EJB dans une relation un-un. Ce type de composant fournit un ensemble de traitement via des méthodes, mais il a la possibilité de conserver des données entre les différents appels de méthodes d'un même client, qui sollicite ses services et ce, tout au long du dialogue.

Les données conservées par le bean sont donc conservées en mémoire.

Message Driven Bean

Les précédents types d'EJB offrent des services de manières synchrone. Le client émet une requête, puis attend que l'EJB lui envoie un résultat.

Pour les Message Driven Bean (MDB), le comportement est complètement différent. Les clients n'appellent pas directement des méthodes mais utilisent JMS pour produire un message et le publier dans une file d'attente. A l'autre bout, le MDB est à l'écoute de cette file d'attente et se « réveille » à l'arrivée du message. Il extrait ce dernier de la file d'attente, en récupère le contenu puis exécute un traitement. Le client n'a donc pas besoin de figer son exécution durant le traitement du MDB. Le traitement est asynchrone.

Entity Bean

Les EJB stateful sont détruits lorsque la session du client se termine. Ils ne peuvent donc pas être utilisés pour stocker de façon permanente les informations de l'application. Les EJB entités peuvent répondre à ce besoin puisqu'ils sont persistants. En effet, leur état est sauvegardé sur un support de stockage externe, comme une base de données. Les entities représentent des données, ou plus exactement des objets métier, qui perdurent après la fin d'une session.

Base de données PostgreSQL

La base de données choisi pour cette application est une base de données PostGreSQL.

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git** et déposé sur le compte GitHub de la société It Consulting, les dépendances et le packaging par **Apache Maven.**

5.1.1 - Les modules

L'application est une application Maven multi-module. Elle est composée de plusieurs modules :

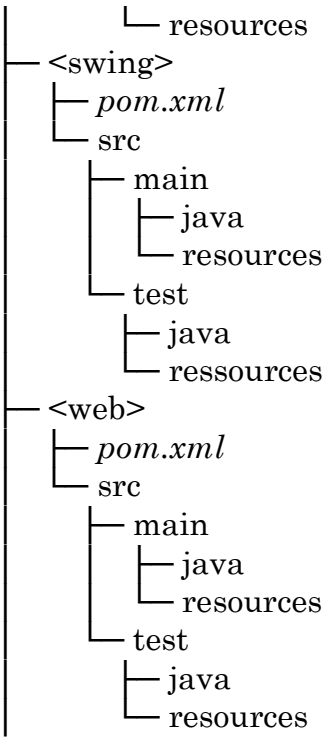
- Module « ear » : le module pour générer le fichier à déployer sur le serveur d'application GlassFish qui contiendra la couche métier (module « business ») et la couche web (module « web ») de l'application ;
- Module « business » : qui contient la logique métier utilisée aussi bien par l'interface internet que l'interface boutique ;
- Module « web » : qui contient le code nécessaire à l'affichage de pages jsp/jsf pour l'interface internet ;
- Module « swing » : qui contient le code pour la création de l'interface boutique en client lourd Java Swing ;

5.1.2 - Structure des sources

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 »)

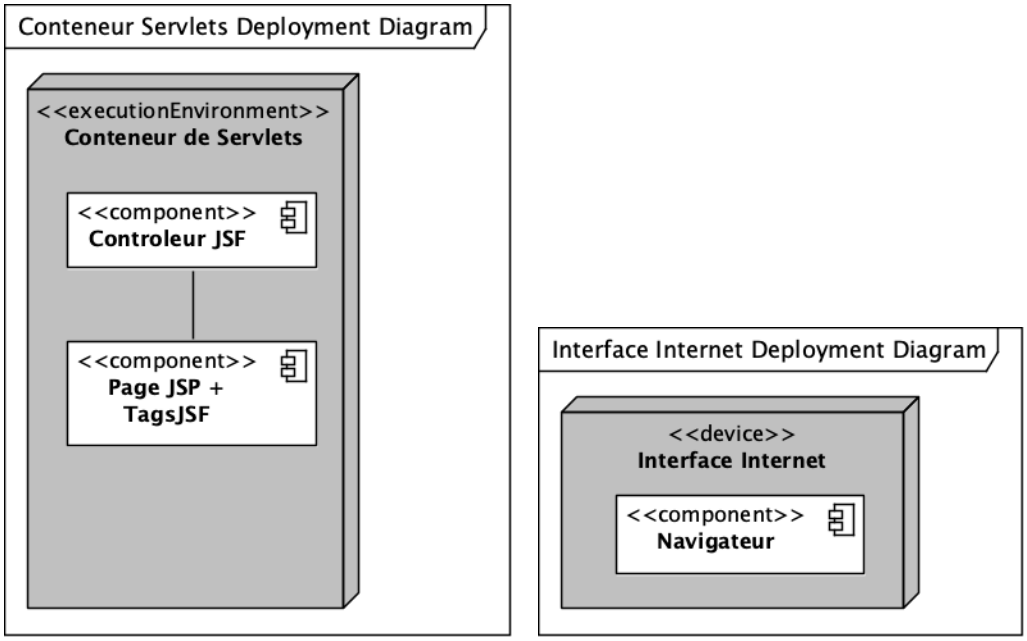
racine
- *pom.xml*
- <ear>
  - *pom.xml*
- <business>
  - *pom.xml*
  - src
    - main
      - java
      - resources
    - test
      - java



5.2 - Interface Internet

La pile logicielle est la suivante :

- Application **J2EE / JSF** (JDK version 1.8)
- Serveur d'application **GlassFish**

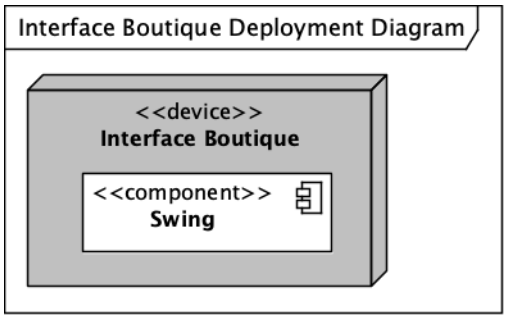


Le Conteneur de Servlets utilisé possède deux composants : un contrôleur JSF et un composant contenant les page JSP et les tags JSF. Les pages JSP et les tags JSF servent à l'affichage dans le navigateur de l'utilisateur du site Web de OC Pizza. Le contrôleur est la partie qui permet la gestion de la partie métier en reliant les pages JSP à la partie métier du conteneur EJB.

5.3 - Interface Boutique

La pile logicielle est la suivante :

- Application **JAVA Swing** (JDK version 1.8)
- Serveur d'application **GlassFish**



L'interface Boutique est écrite en Swing, une API permettant de construire des interfaces graphiques sophistiquée en langage Java.

6 - POINTS PARTICULIERS

6.1 - Gestion des logs

Des logs de toutes les actions réalisées sont stockés sur le serveur GlassFish et consultable sur la page Web d'administration du serveur. Ces logs sont collectés à des fins statistiques et de sécurité. Les logs seront conservés un an, puis écrasés.

6.2 - Ressources

Au niveau matériel, OCPizza devra disposer dans les locaux techniques de son siège, d'un serveur d'application GlassFish et un serveur de base de données PostgreSQL déployé sur un serveur physique Windows Server 2019.

6.3 - Environnement de développement

Le développement sera réalisé dans les locaux d'**It Consulting.** Le projet sera développé à l'aide de l'IDE IntelliJ IDEA.

6.4 - Procédure de packaging / livraison

Les sources des différentes applications seront stockées et mises à disposition sur GitHub.