« OC Pizza » est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici la fin de l'année. Un des responsables du groupe a pris contact avec vous afin de mettre en place un système informatique, déployé dans toutes ses pizzerias.

Concevez la solution technique d'un système de gestion de pizza

OpenClassrooms

Damien Gironnet

Sommaire

1 Contexte du document

« OC Pizza » est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici la fin de l'année. Un des responsables du groupe a pris contact avec vous afin de mettre en place un système informatique, déployé dans toutes ses pizzerias et qui lui permettrait notamment :

- d'être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;
- de suivre en temps réel les commandes passées et en préparation ;
- de suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas sont encore réalisables ;
- de proposer un site Internet pour que les clients puissent :
    - passer leurs commandes, en plus de la prise de commande par téléphone ou sur place
    - payer en ligne leur commande s'ils le souhaitent – sinon, ils paieront directement à la livraison
    - modifier ou annuler leur commande tant que celle-ci n'a pas été préparée
- de proposer un aide-mémoire aux pizzaiolos indiquant la recette de chaque pizza
- d'informer ou notifier les clients sur l'état de leur commande

Le client a déjà fait une petite prospection et les logiciels existants qu'il a pu trouver ne lui conviennent pas.

2 Travail Demandé

En tant qu'analyste-programmeur, votre travail consiste, à ce stade, à définir le domaine fonctionnel et à concevoir l'architecture technique de la solution répondant aux besoins du client, c'est-à-dire :

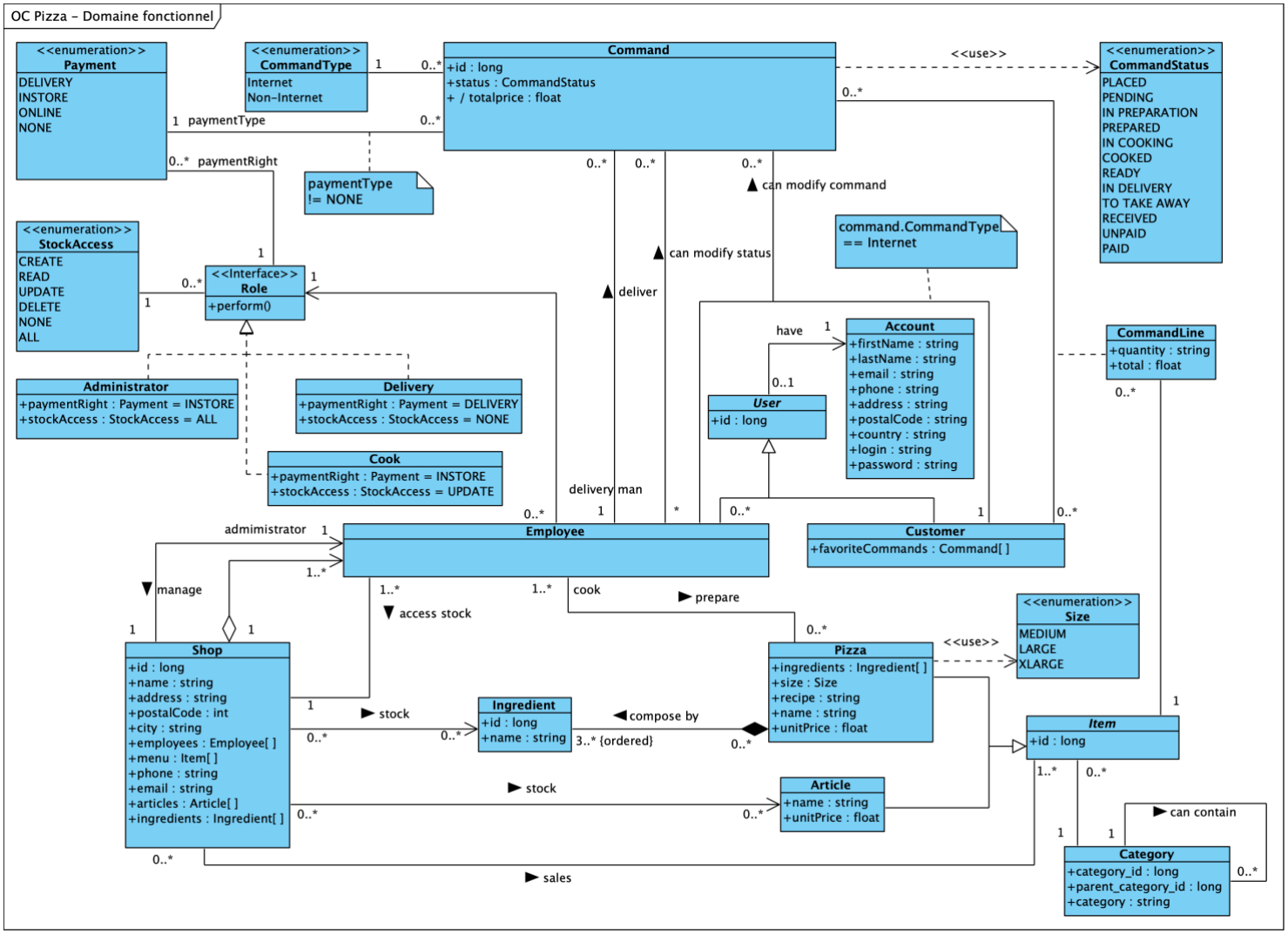
- modéliser les objets du domaine fonctionnel
- identifier les différents éléments composant le système à mettre en place et leurs interactions
- décrire le déploiement des différents composants que vous envisagez
- élaborer le schéma de la ou des bases de données que vous comptez créer

Votre travail sera validé par un des développeurs expérimentés de votre société (ce rôle est assuré par le mentor qui vous fera passer la soutenance du projet).
Vous utiliserez UML pour réaliser cette conception.

3 Domaine Fonctionnel

3.1 Diagramme de classe globale du domaine fonctionnel

Le diagramme de classe ci-dessous représente le domaine fonctionnel dans sa totalité.



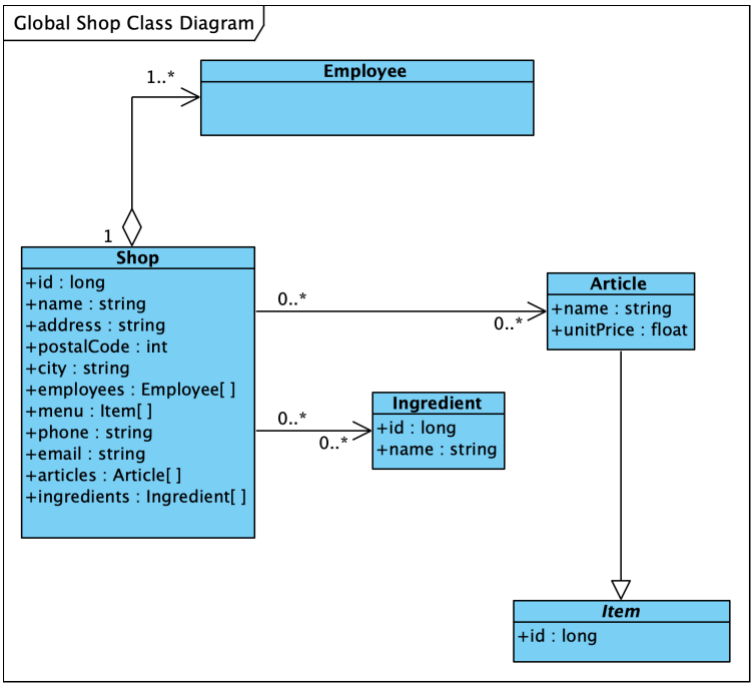
A partir de ce diagramme, nous pouvons distinguer les différentes parties qui composent la solution proposée :

- Shop
- Command
- User
- Product

3.2 Description des classes de la partie Shop

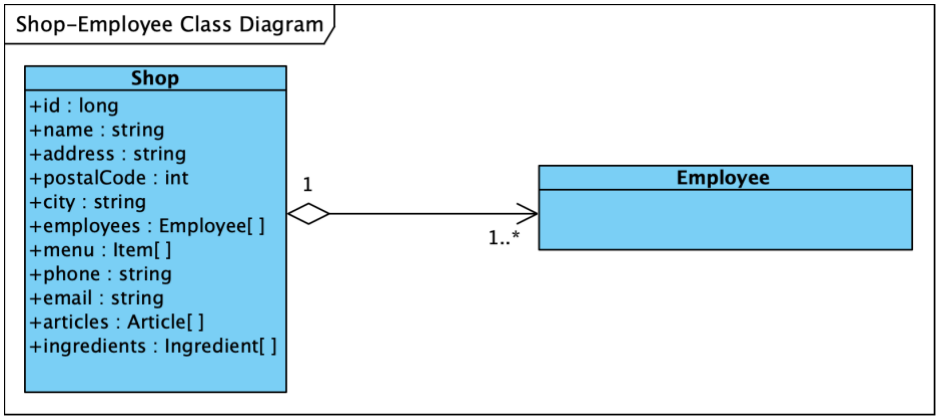
3.2.1 Diagramme de la partie Shop

Le diagramme ci-dessous est la représentation des classes de la partie Shop.



3.2.2 Association Shop – Employee

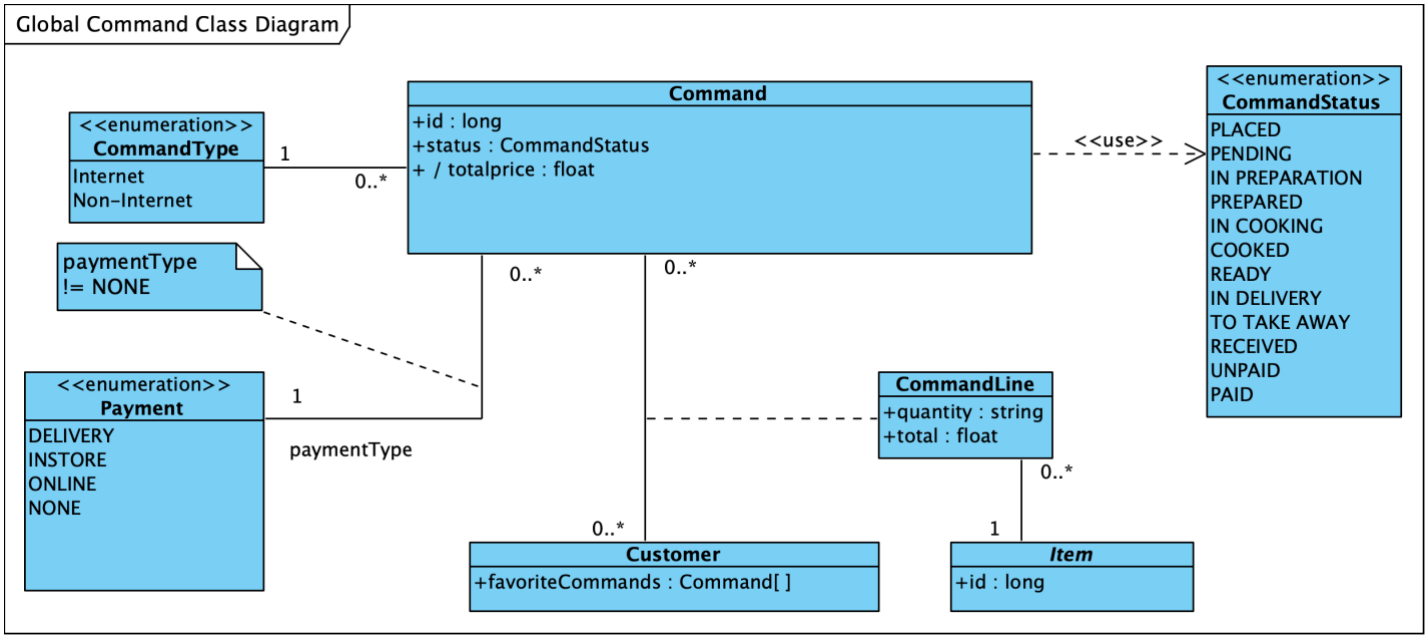
Le diagramme ci-dessous représente l'association entre les classes Shop et Employee. Chaque Pizzeria possède un ensemble d'employés. cet ensemble ne peut pas être vide. Il possède au minimum un employé.



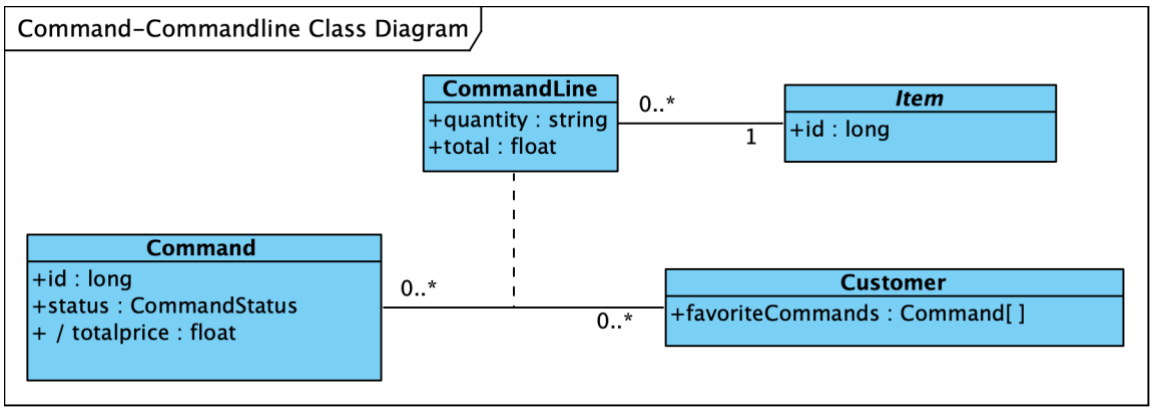
3.3 Description des classes de la partie Command

3.3.1 Diagramme de la partie Command

Le diagramme ci-dessous est la représentation des classes de la partie Command.



3.3.2 Association Command – Command Line

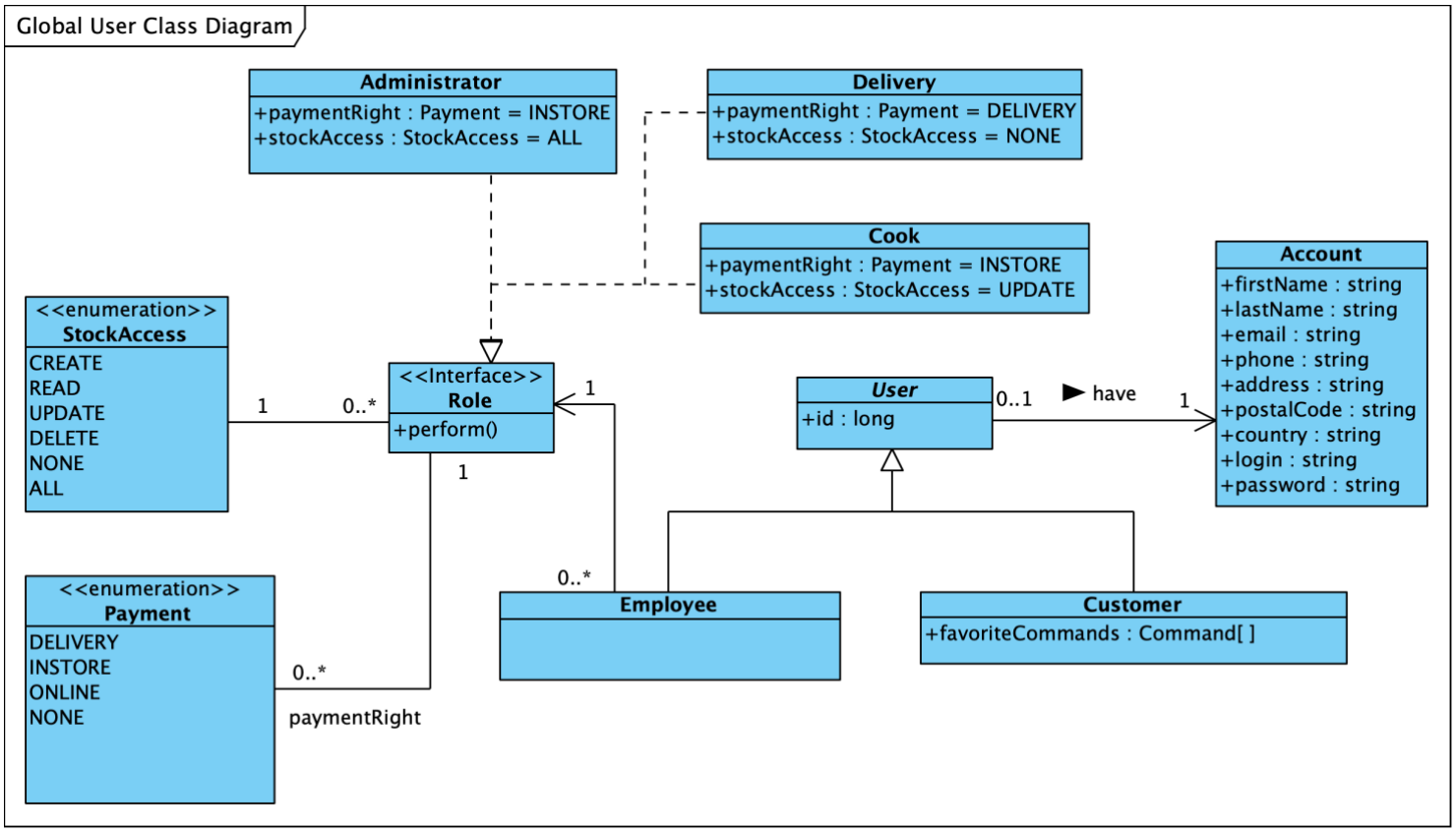


L'association entre Command et CommandLine est un peu particulière car La Classe CommandLine est une classe dite « d'association ». Ceci signifie que la classe Command contient un ensemble d'élément de type CommandLine. On peut aussi remarquer que la classe CommandLine possède une association d'un à un avec la classe Item.

3.4 Description des classes de la partie User

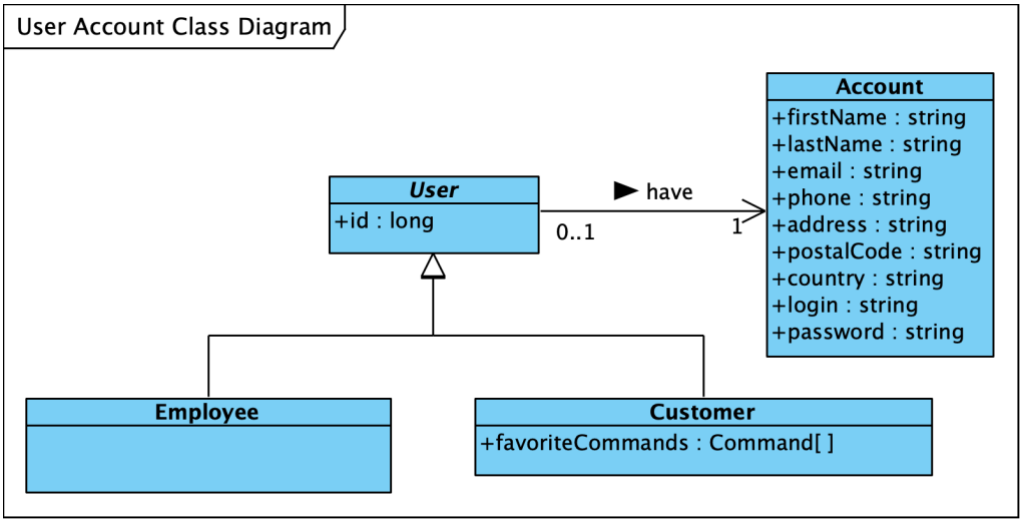
3.4.1 Diagramme de la partie User

Le diagramme ci-dessous est la représentation des classes de la partie User.



3.4.2 Association User – Account

Le diagramme ci-dessous représente l'association entre les classes User et Account. La classe Account contient l'ensemble des informations des sous-classes de la classe abstraite User c'est-à-dire « Customer » et « Employee ».

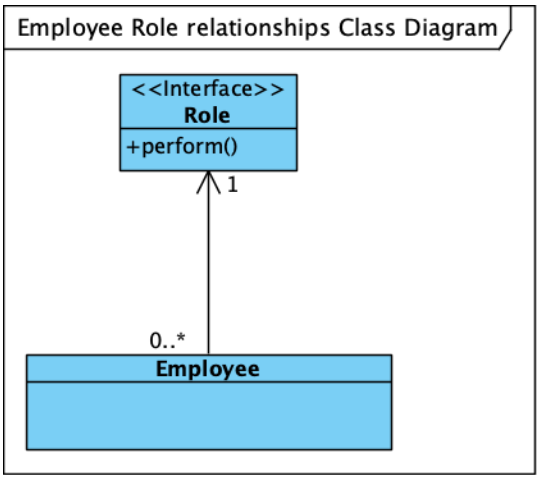


3.4.3 Classe Employee

La classe Employee représente un employé de la pizzeria. Elle ne possède pas directement de champs propres mais elle hérite dans champs de sa super-classe User et possède aussi les champs de l'objet héritant de l'interface Role qui lui est attribué.

3.4.4 Association Employee – Role

Le diagramme ci-dessous représente l'association entre les classes Employee et Role. L'élément Role est une interface. elle permet d'attribuer des valeurs par défaut et des implémentations différentes aux classes qui l'implémentent.



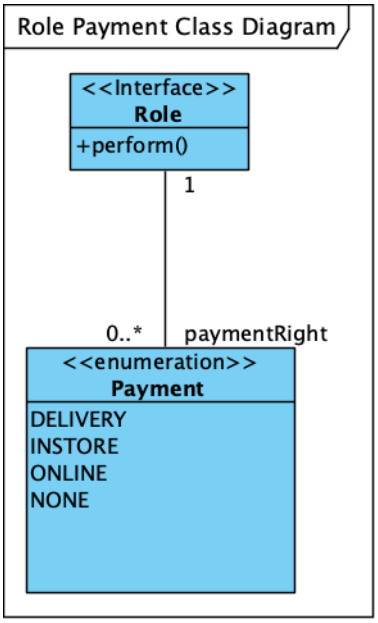
3.4.5 Interface Rôle de l'employé

L'élément Role est une interface. Elle permet de mettre en place le design pattern « **Strategy** » en attribuant dans les classes qui implémente l'interface des valeurs par

défaut aux champs qu'elle contient et des implémentations spécifiques aux méthodes selon la classe implémentant l'interface.

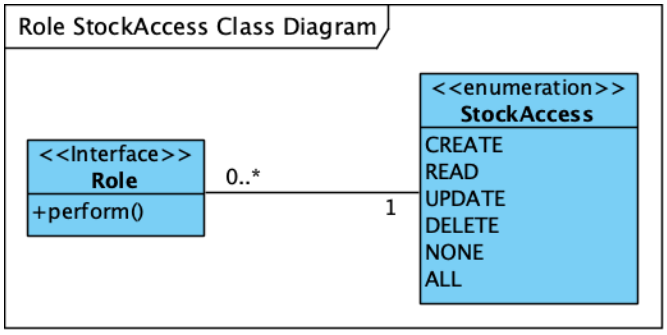
3.4.6 Association Role – Payment Right

Le diagramme ci-dessous représente l'association entre les classes Role et l'énumération Payment ayant pour rôle le droit de paiement (« paymentRight ») qui peut être soit à la livraison, en magasin.



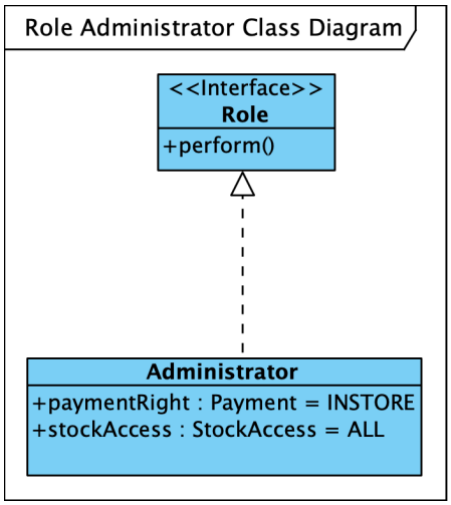
3.4.7 Association Role – Stock Access

Le diagramme ci-dessous représente l'association entre les classes Role et l'énumération StockAccess. L'élément StockAccess est une **énumération**. Elle permet d'attribuer un droit d'accès au stock dans le système avec la possibilité d'attribuer le même droit à différentes classes implémentant l'interface Role



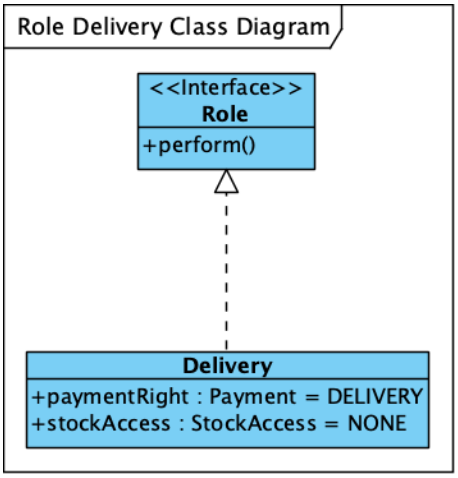
3.4.8 Classe Administrator

La classe Administrator est la classe qui détermine le rôle de l'administrateur d'une pizzeria. Il a pour rôle d'administrer la pizzeria. Il a les droits pour effectuer des paiements en magasin et a tous les droits d'accès au stock.



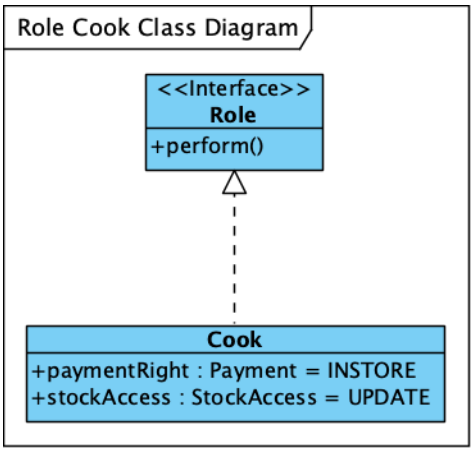
3.4.9 Classe Delivery

La classe Delivery est la classe qui détermine le rôle de livreur d'une pizzeria. Il a pour rôle de livrer les commandes. Il a les droits pour effectuer des paiements en livraison et n'a aucuns droits d'accès au stock.



3.4.10 Classe Cook

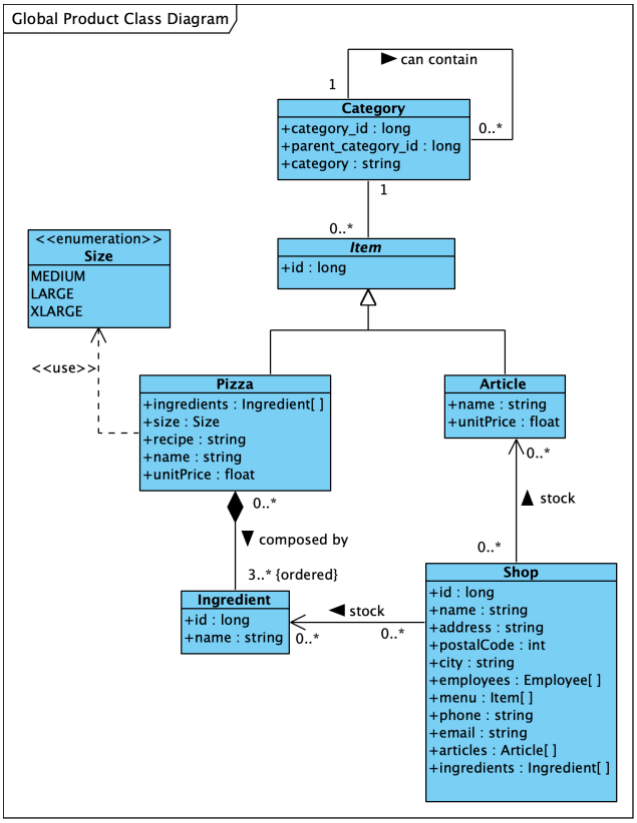
La classe Delivery est la classe qui détermine le rôle de cuisinier d'une pizzeria. Il a pour rôle de préparer les pizzas. Il a les droits pour effectuer des paiements en magasin et a le droit de mettre à jour le stock.



3.5 Description des classes de la partie Product

3.5.1 Diagramme de la partie Product

Le diagramme ci-dessous est la représentation des classes de la partie Product.

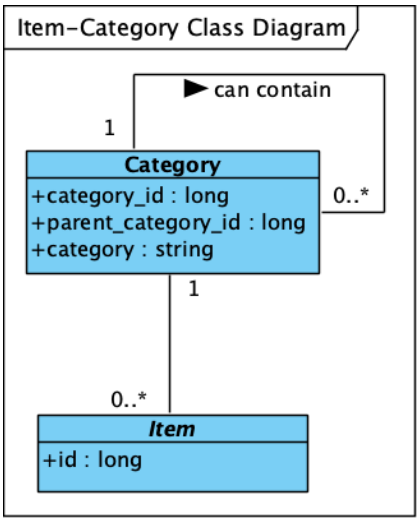


3.5.2 Classe Item

La classe Item est une classe abstraite ce qui signifie qu'elle ne peut être instancié uniquement par ses sous-classes que sont les classes « Pizza » et « Article ». L'utilité principale d'utiliser cette classe abstraite est de permettre de regrouper par le biais de leur identifiant les classes Pizza et Article qui modélise deux types d'objets très différents. Les pizzas sont comme leur nom l'indique le produit premier de la pizzeria alors que les articles peuvent être toutes sortes de produits additionnelles comme des boissons, entrée ou dessert qui sont vendu par la pizzeria mais pas conçu par elle.

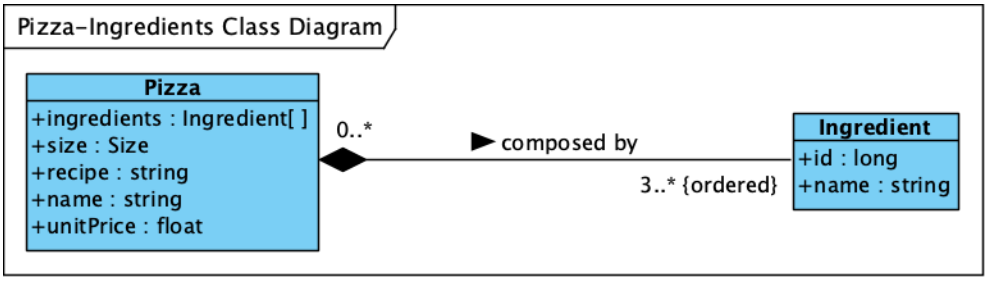
3.5.3 Association Item – Category

Le diagramme ci-dessous représente l'association entre les classes Item et Category. La classe Category représente comme son nom l'indique la catégorie de produit dans laquelle l'item est. Elle a une **association réflexive** sur elle-même ce qui signifie qu'elle possède une référence sur un objet du même type qui a un rôle d'objet parent. Ainsi une catégorie peut posséder une catégorie parente et une liste de sous-catégorie.



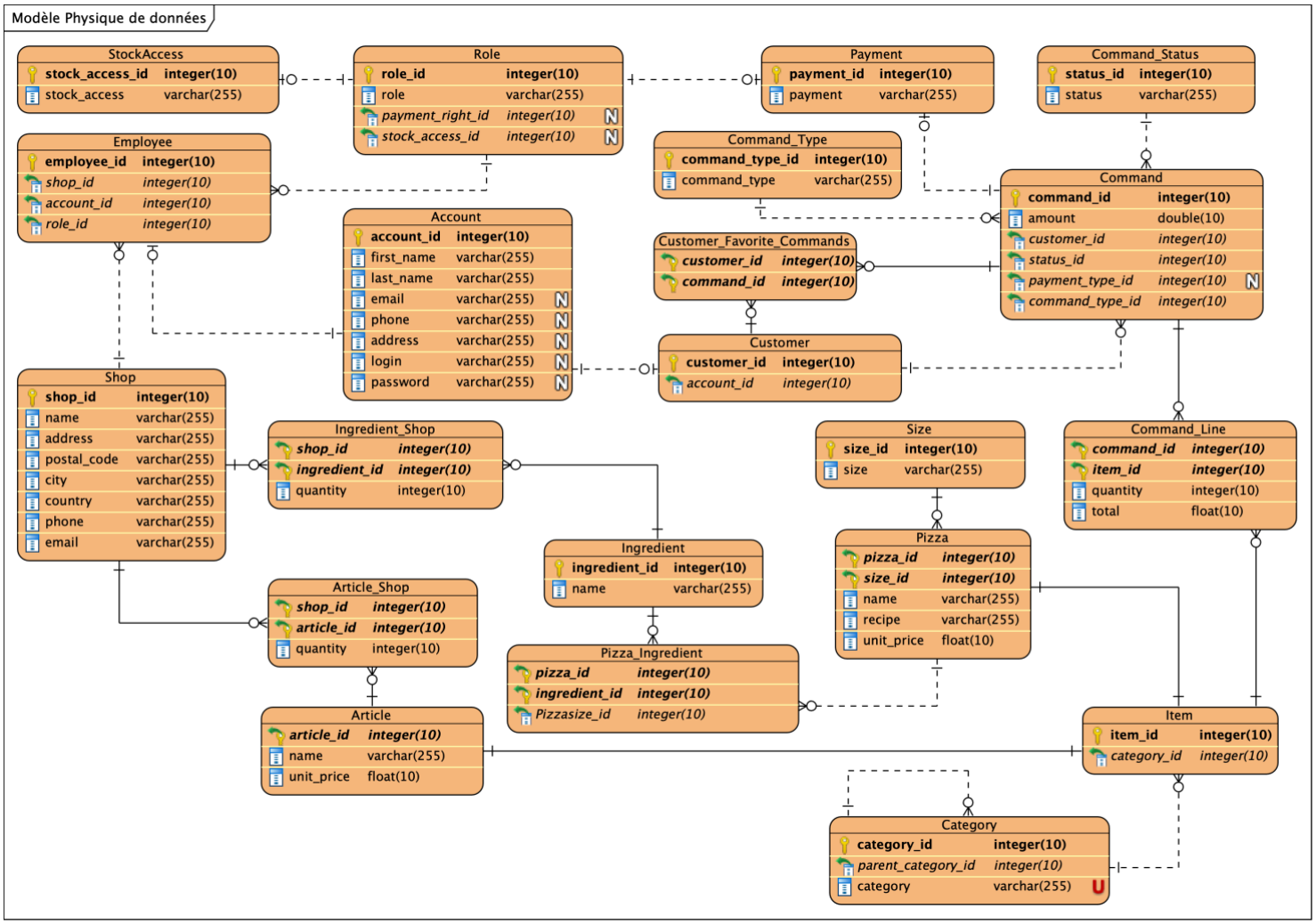
3.5.4 Association Pizza - Ingredient

Si l'on observe l'association reliant la classe Pizza et la classe Ingredient, on peut remarquer la présence d'un losange plein à l'extrémité de la classe Pizza. Cela signifie que chaque ingrédient de la pizza sont des composants de la pizza.



4 Modèle Physique de données

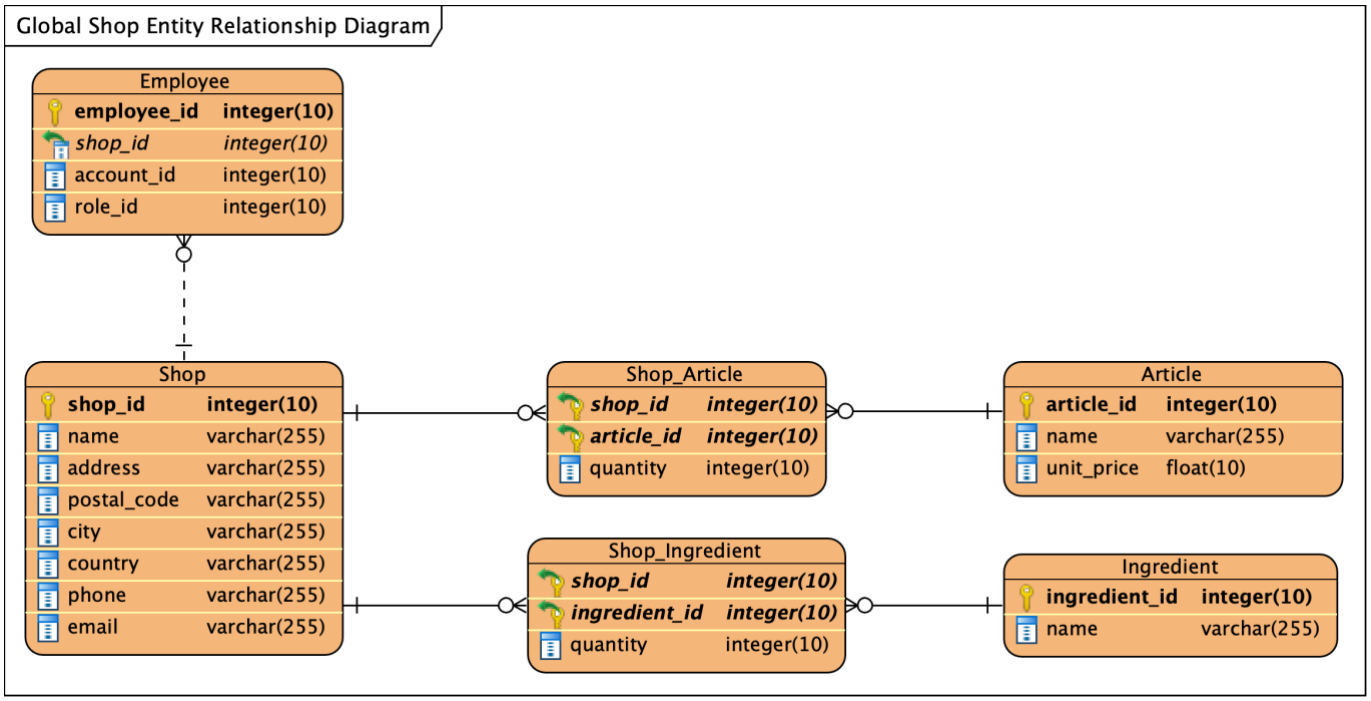
4.1 Diagramme globale du modèle physique de données



4.2 Description des entités de la partie Shop

4.2.1 Diagramme de la partie Shop

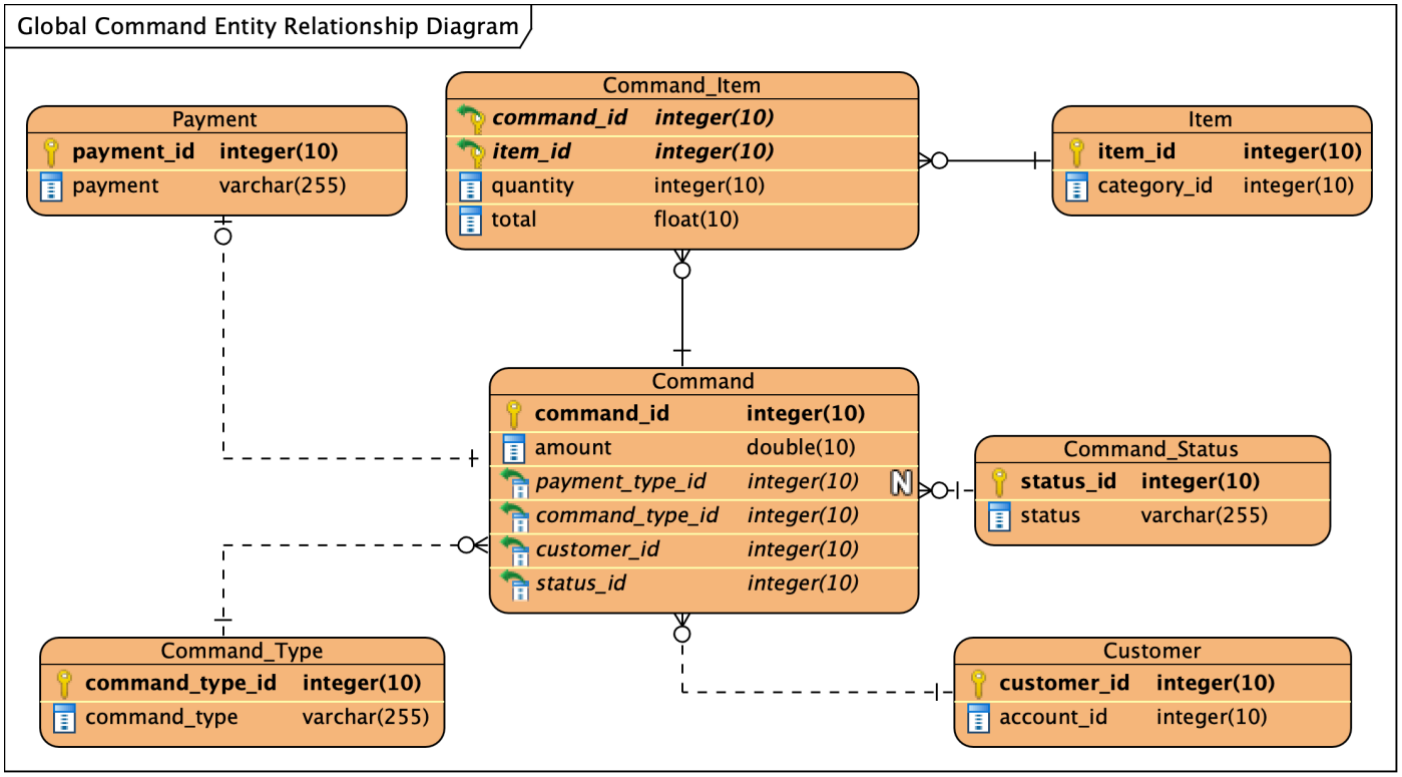
Le diagramme ci-dessous est la représentation des entités de la partie Shop résultant de la partie Shop du domaine fonctionnel.



4.3 Description des entités de la partie Command

4.3.1 Diagramme de la partie Command

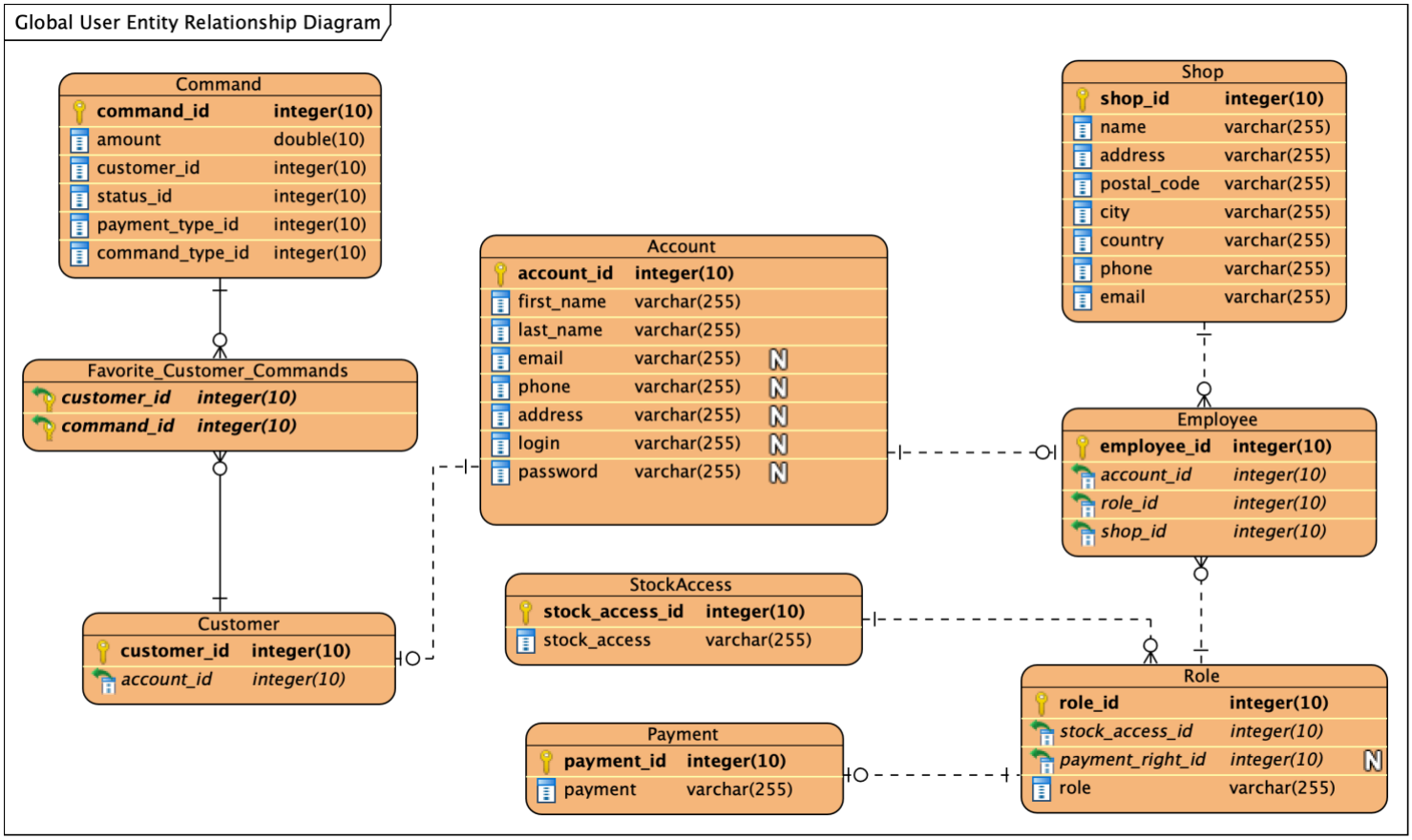
Le diagramme ci-dessous est la représentation des entités de la partie Command résultant de la partie Command du domaine fonctionnel.



4.4 Description des entités de la partie User

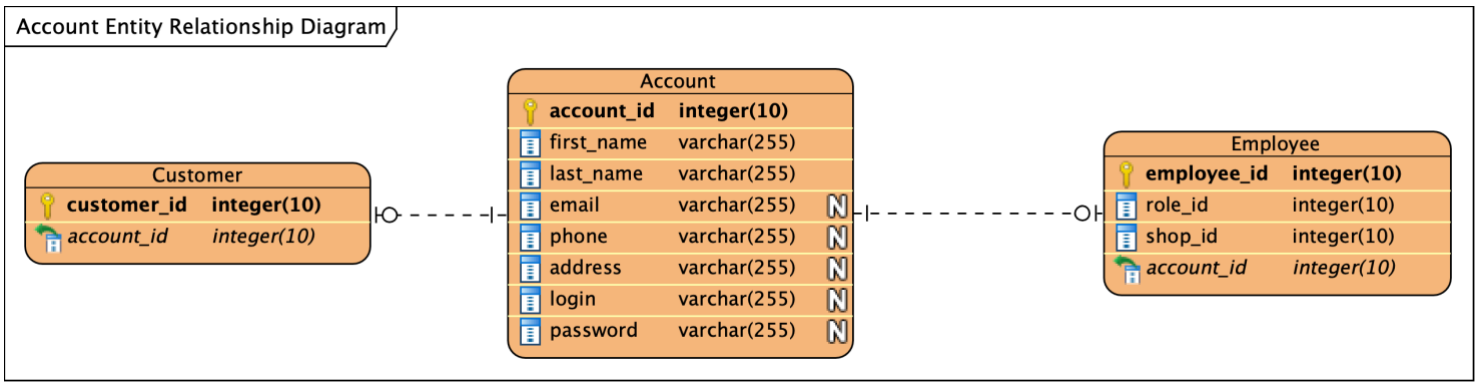
4.4.1 Diagramme de la partie User

Le diagramme ci-dessous est la représentation des entités de la partie User résultant de la partie User du domaine fonctionnel.



4.4.2 Relation Customer – Account - Employee

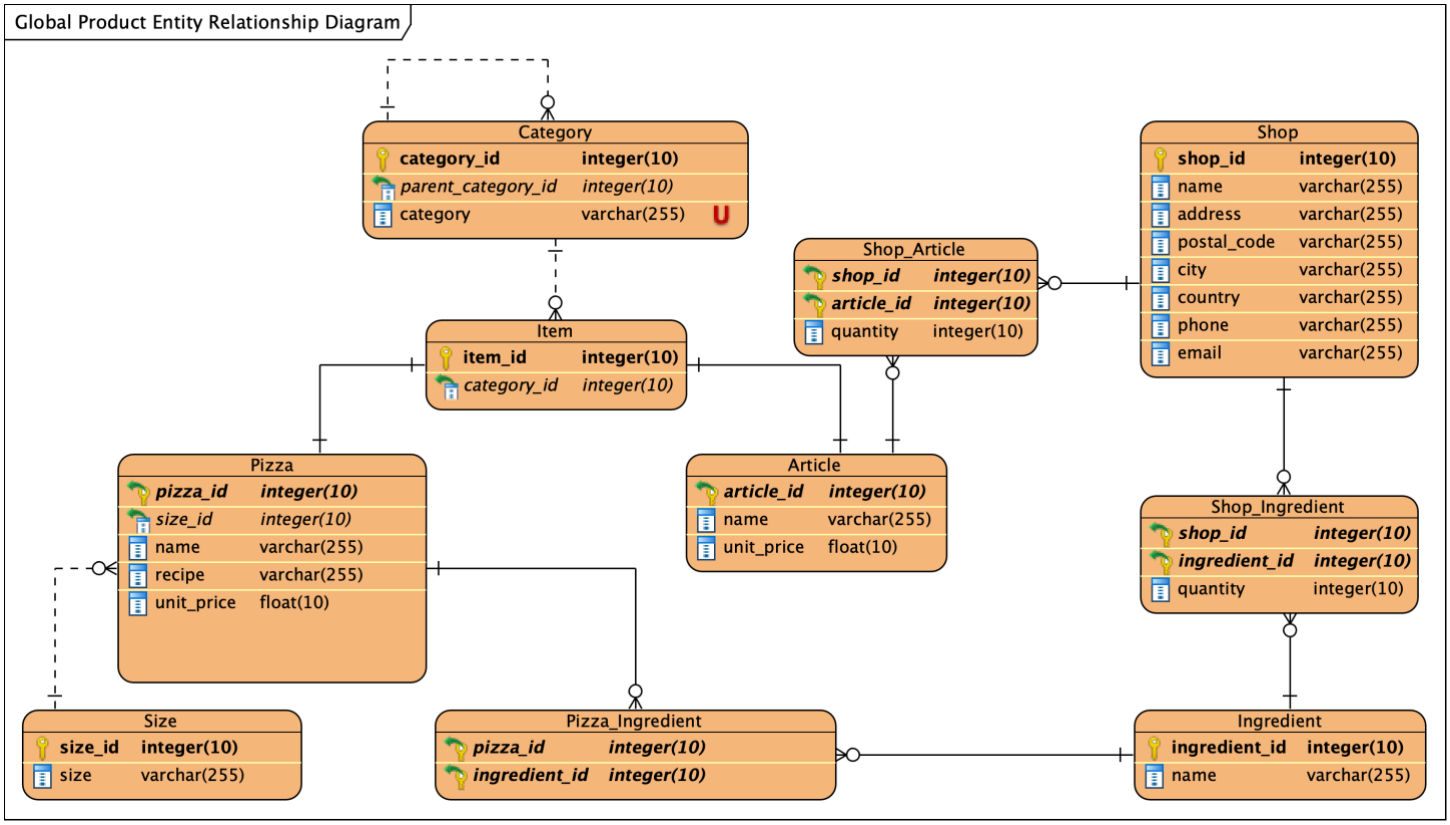
Contrairement au domaine fonctionnel qui possède un élément User dont les classes Employee et Customer héritent, le modèle physique de données ne possède pas d'entité User et attribué directement un identifiant aux entités Customer et Employee.



4.5 Description des entités de la partie Product

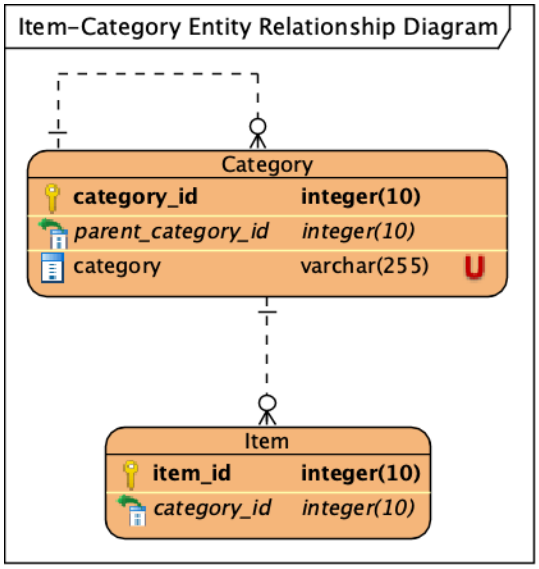
4.5.1 Diagramme de la partie Product

Le diagramme ci-dessous est la représentation des entités de la partie Product résultant de la partie Product du domaine fonctionnel.



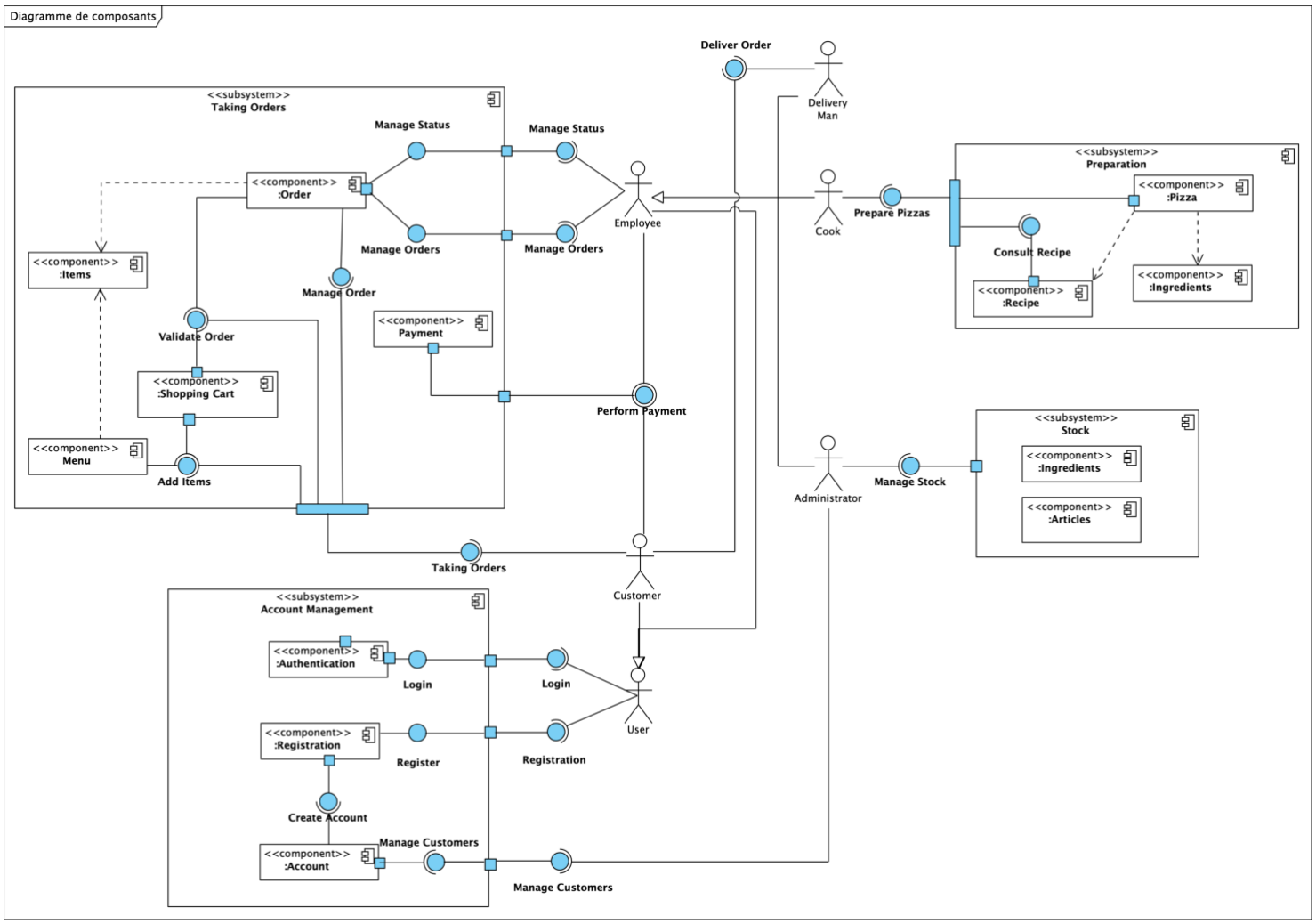
4.5.2 Relation Item – Category

L'entité Category est le résultat de la classe Category du domaine fonctionnel dans le diagramme ERD. Elle représente comme son nom l'indique la catégorie de produit dans laquelle l'item est. Elle a une association réflexive ce qui signifie qu'elle peut possède une référence sur un objet du même type qui a un rôle d'objet parent. Ainsi une catégorie peut posséder une catégorie parente et une liste de sous-catégorie.



5 Diagramme de composants

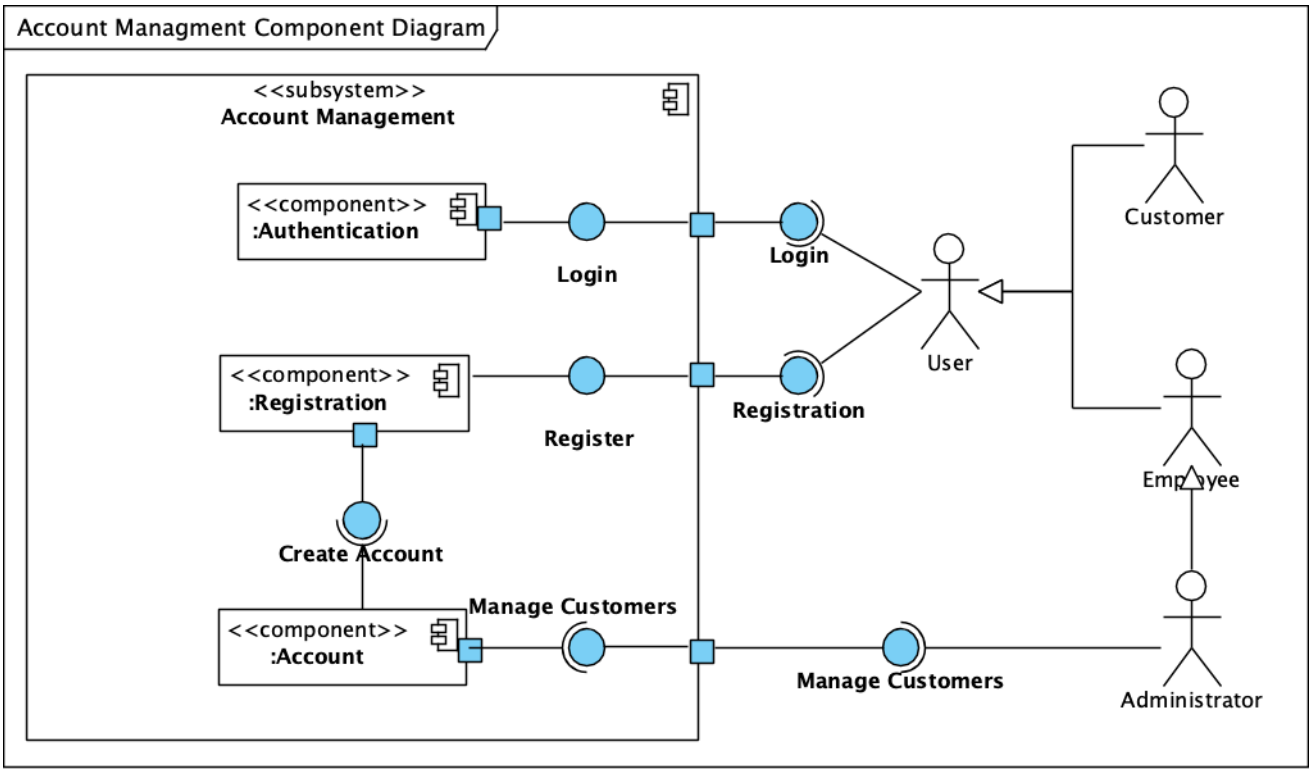
5.1 Diagramme globale de composants



5.2 Description du subsystem Account Managment

5.2.1 Diagramme du subsystem Account Managment

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



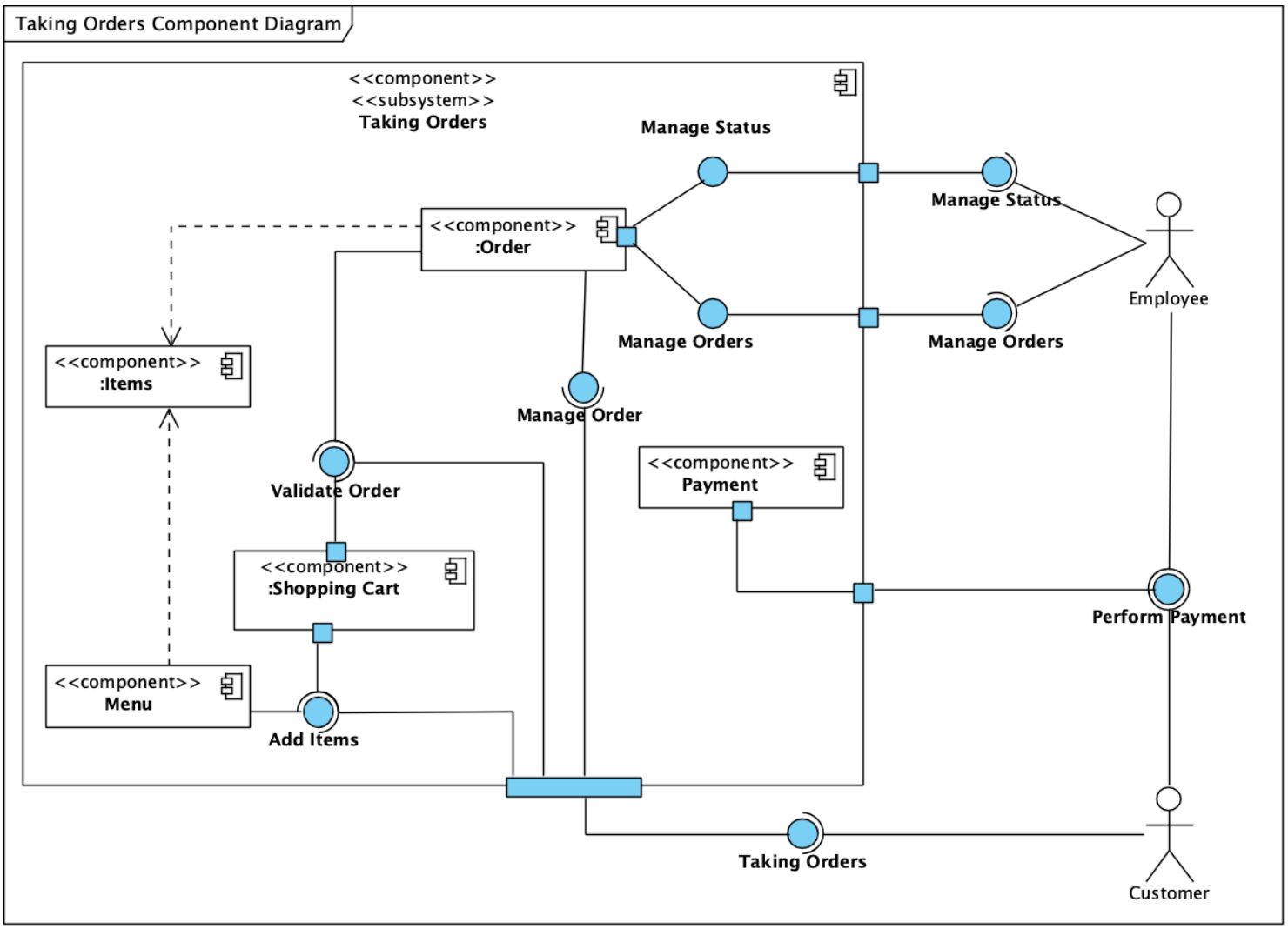
5.2.2 Description

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.

5.3 Description du Subsystem Taking Orders

5.3.1 Diagramme du subsystem Taking Orders

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



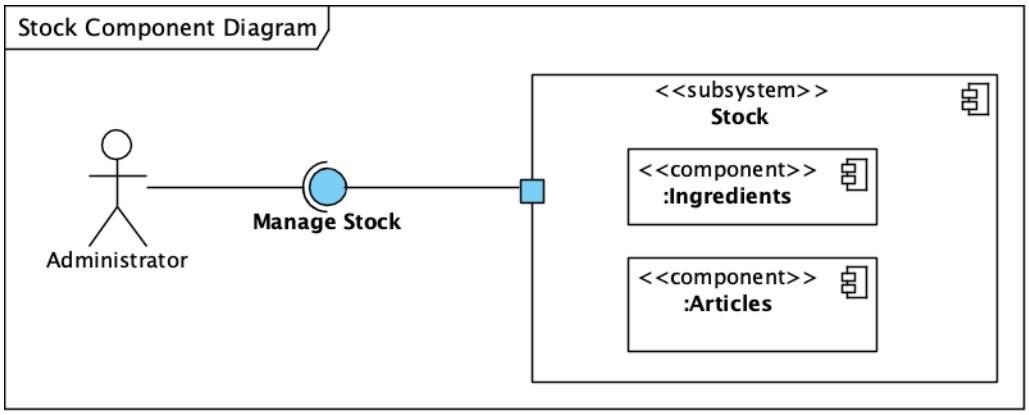
5.3.2 Description

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.

5.4 Description du Subsystem Stock

5.4.1 Diagramme du subsystem Stock

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



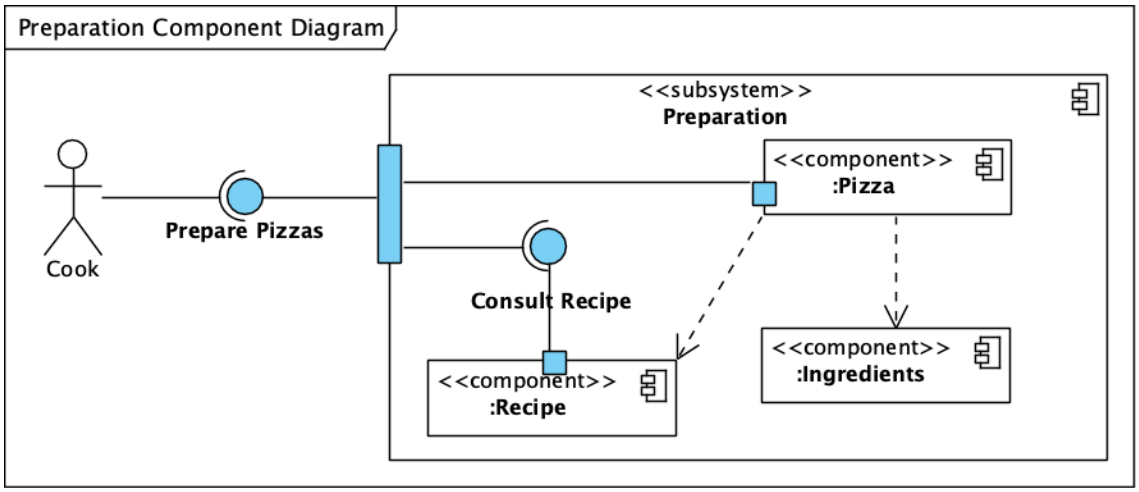
5.4.2 Description

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

5.5 Description du Subsystem Preparation

5.5.1 Diagramme du subsystem Preparation

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

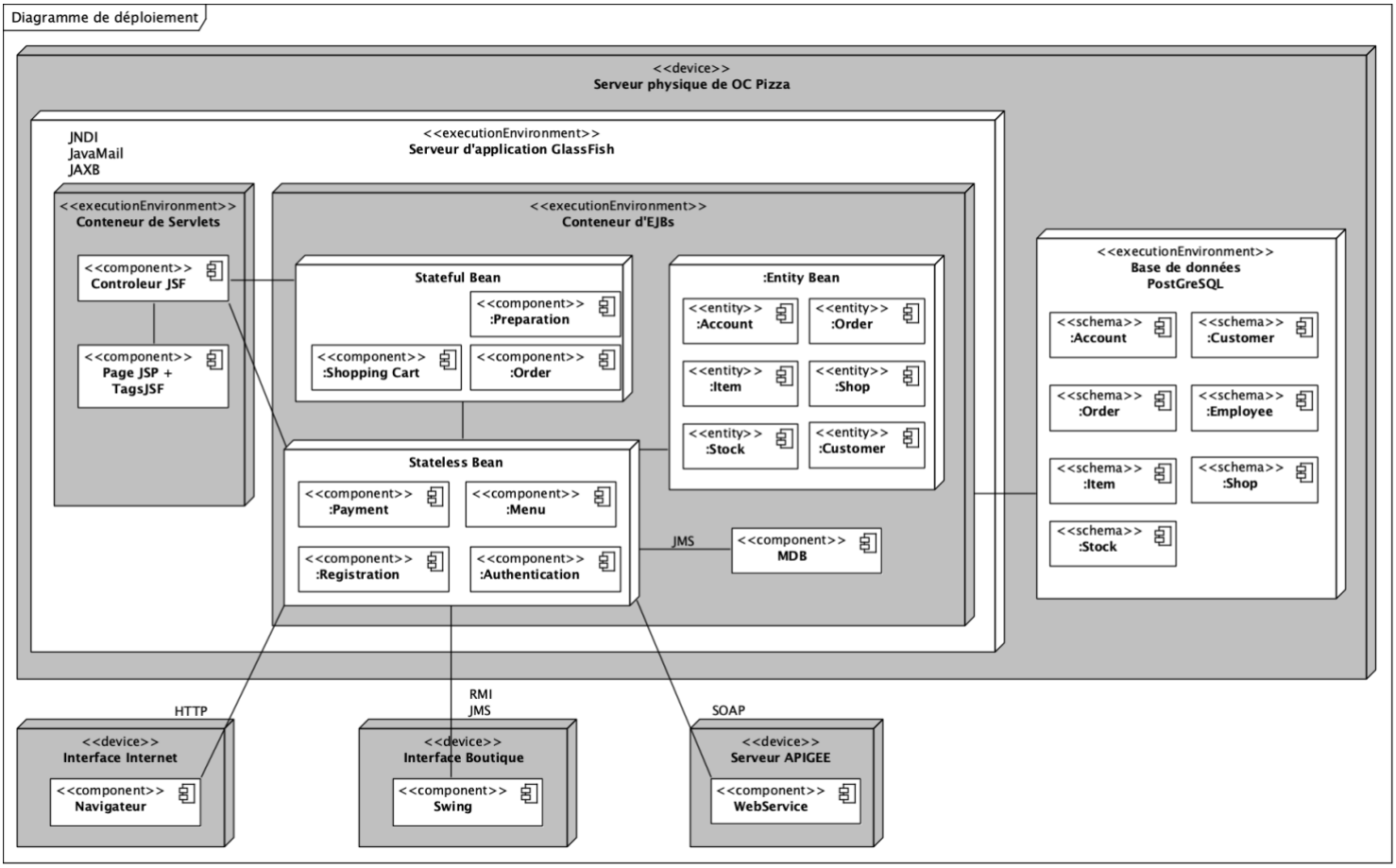


5.5.2 Description

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.

6 Diagramme de déploiement

6.1 Diagramme globale de déploiement



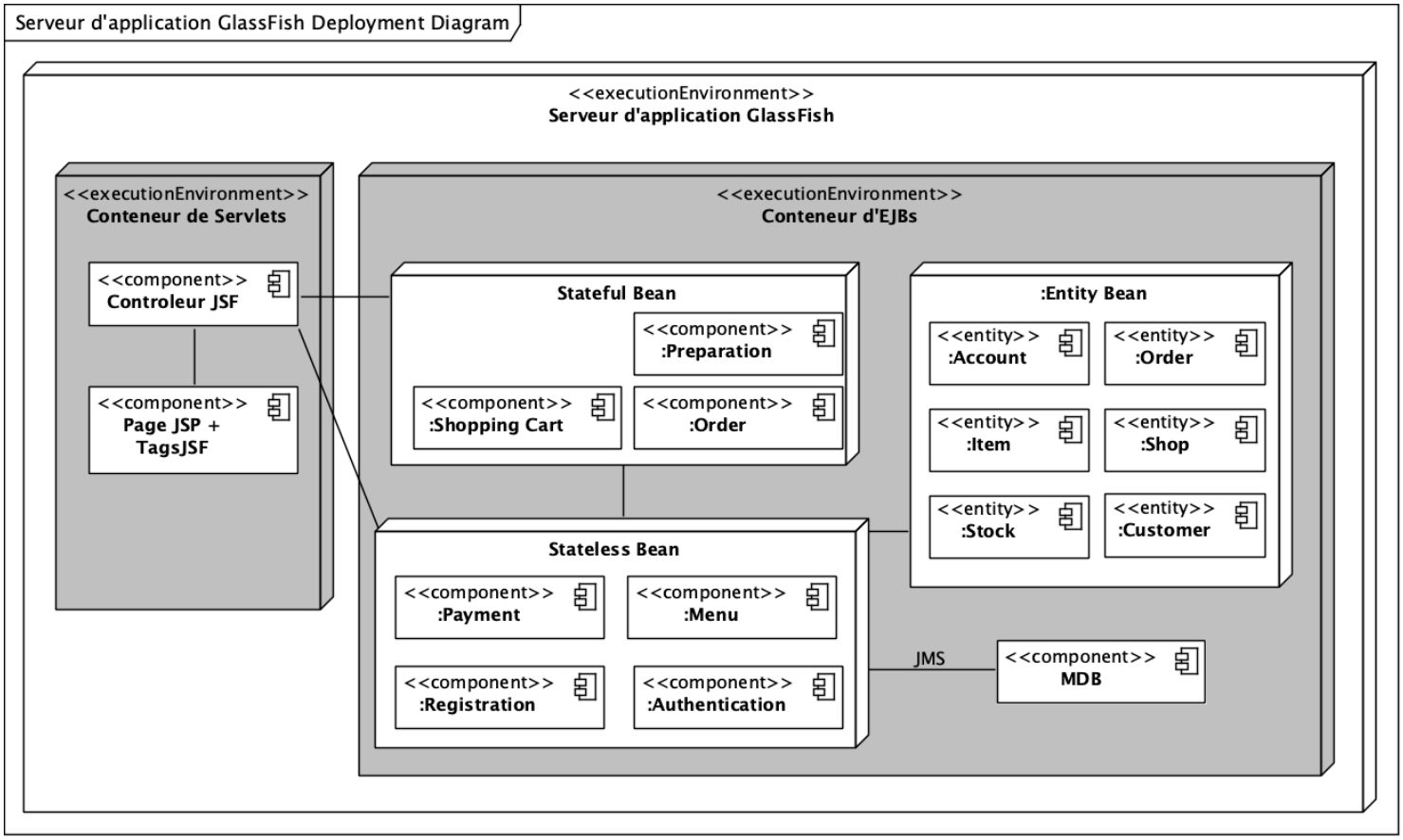
6.2 Description des éléments du diagramme

6.2.1 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.

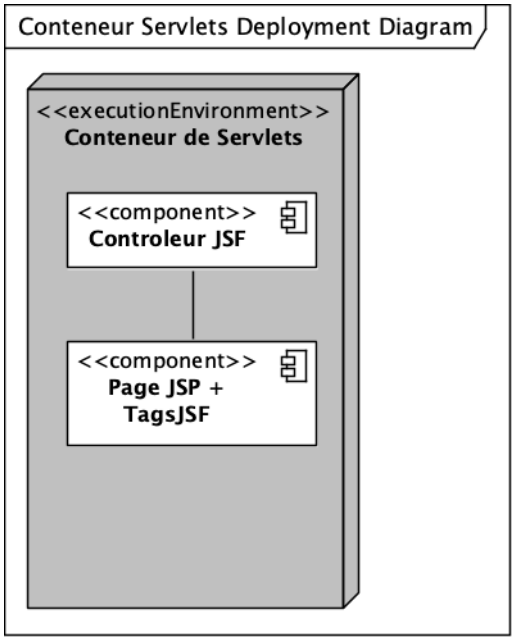
6.2.1.1 Serveur d'application Glassfish

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



6.2.1.2 Conteneur de Servlets

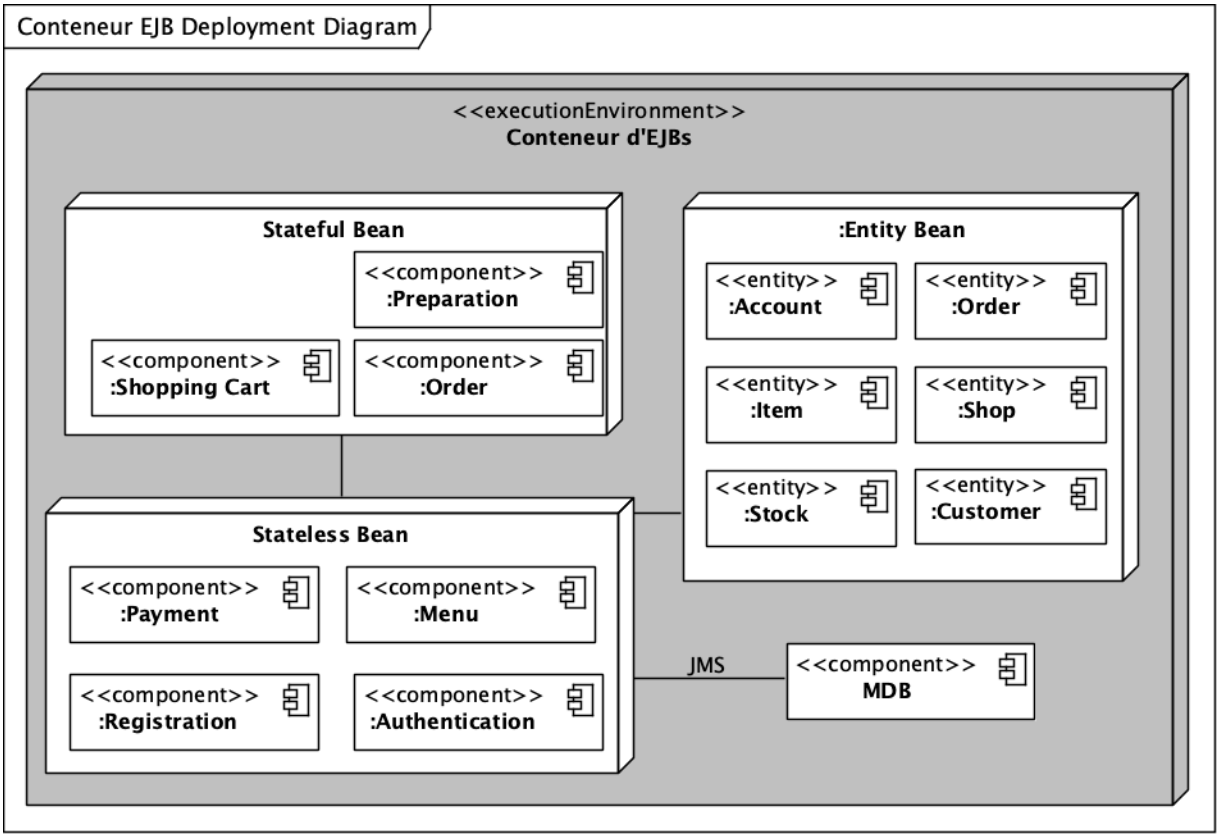
Le Conteneur de Servlets utilisé possède deux composants : un contrôleur JSF et un composant contenant les page JSP et les tags JSF. Les pages JSP et les tags JSF servent à l'affichage dans le navigateur de l'utilisateur du site Web de OC Pizza. Le contrôleur est la partie qui permet la gestion de la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



6.2.1.3 Conteneur d'EJBs

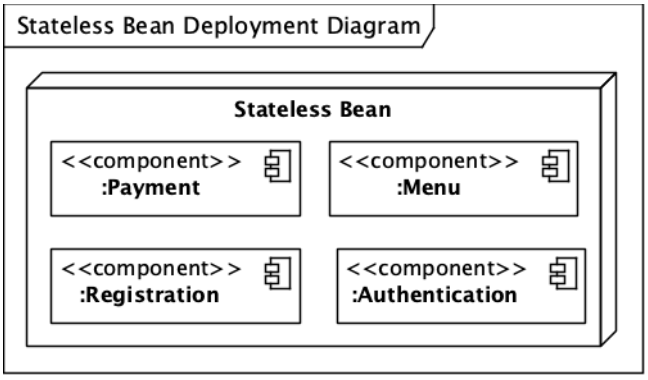
Le Conteneur EJB (Entreprise Java Bean) est l'élément qui gère la partie métier de l'application. Elle est composée de 4 parties :

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



Stateless Bean

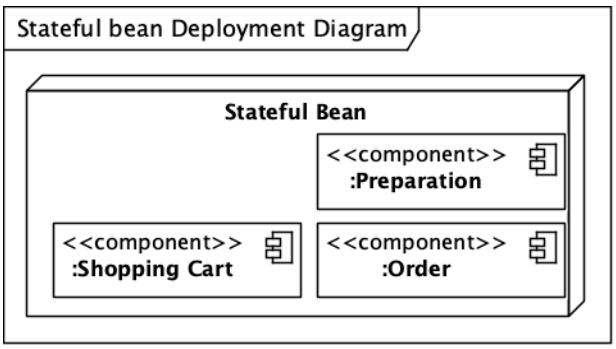
Les EJB Stateless sont des EJB sans état qui ne fonctionne que de manière éphémère. Une fois qu'un client a demandé et reçu une fonctionnalité du composant, l'interaction avec ce composant prend fin, ne laissant aucune trace de ce qui s'est passé. L'utilisation standard d'un EJB sans état réside dans 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.

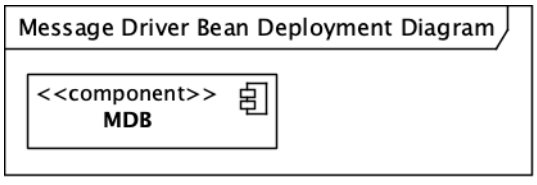
Les données conservées par le bean sont donc conservées en mémoire.



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.

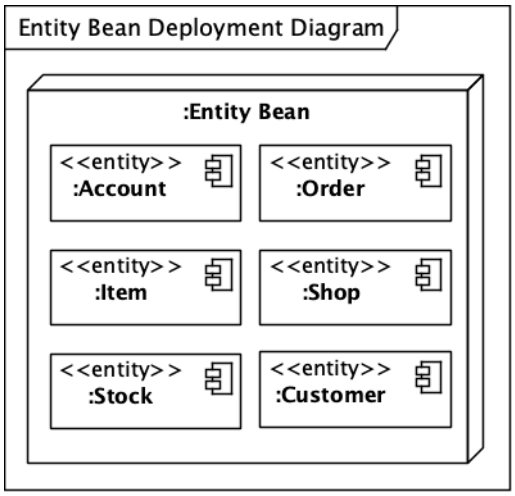
Pour les Message Driven Bean (MDB), le comportement est complètement différent. Les clients n'appellent pas directement des méthodes mais utilisent JMS pour produire un message et le publier dans une file d'attente. A l'autre bout, le MDB est à l'écoute de cette file d'attente et se « réveille » à l'arrivée 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

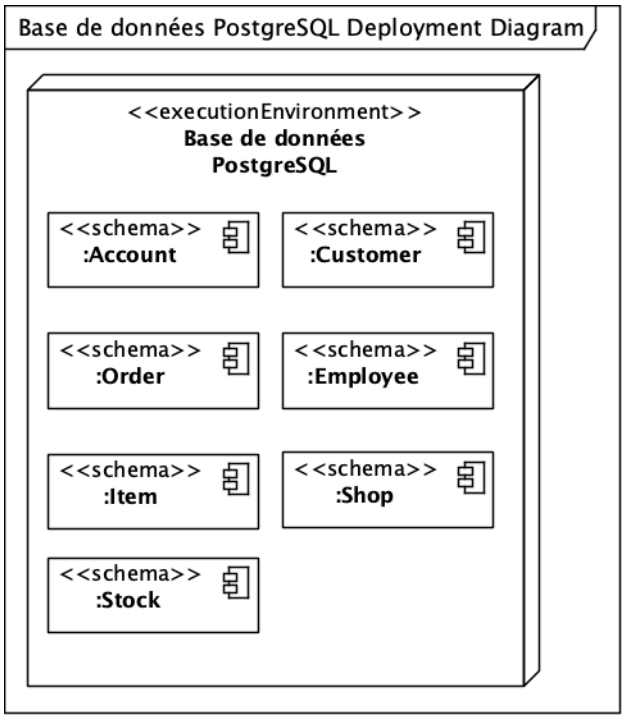
Les EJB stateful sont détruits lorsque la session du client se termine. Ils ne peuvent donc pas être utilisés pour stocker de façon permanente les informations de l'application. Les EJB entités peuvent répondre à ce besoin puisqu'ils sont persistants. En effet, leur état est sauvegardé sur un support 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.



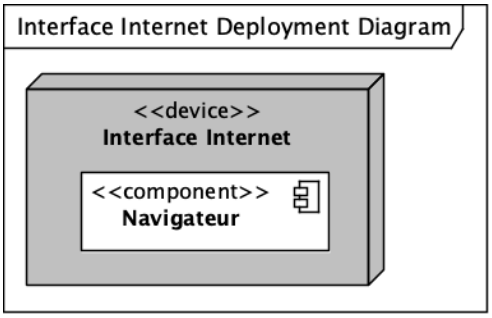
6.2.1.4 Base de données PostgreSQL

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



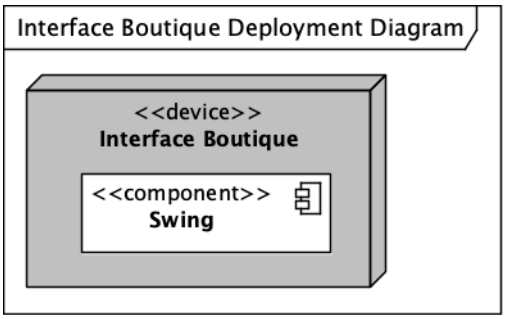
6.2.2 Interface Internet

L'interface Internet est l'interface web utilisé par les internautes et clients pour effectuer des commandes.



6.2.3 Interface Boutique

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



6.2.4 Serveur APIGEE

Le serveur APIGEE est le serveur qui permet par biais de requête SOAP la gestion des paiements.