FORMATION DA JAVA

OPEN CLASSROOMS – 2019

Étudiant : Éric AUBRUN

Projet 5 : Concevez la solution technique d'un système de gestion de pizzeria

https://github.com/RicoBSJ/Projet5_OCR_Rico_2019.git

SOMMAIRE

INTRODUCTION

En tant qu'analyste-programmeur, mon travail vise 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.

Nous conclurons ce document par les solutions fonctionnelles et techniques retenues.

RAPPEL DE LA COMMANDE DU CLIENT

1. Le contexte

« OC Pizza » est un jeune groupe de pizzeria en plein essor 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 6 mois.

2. Les besoins du Client

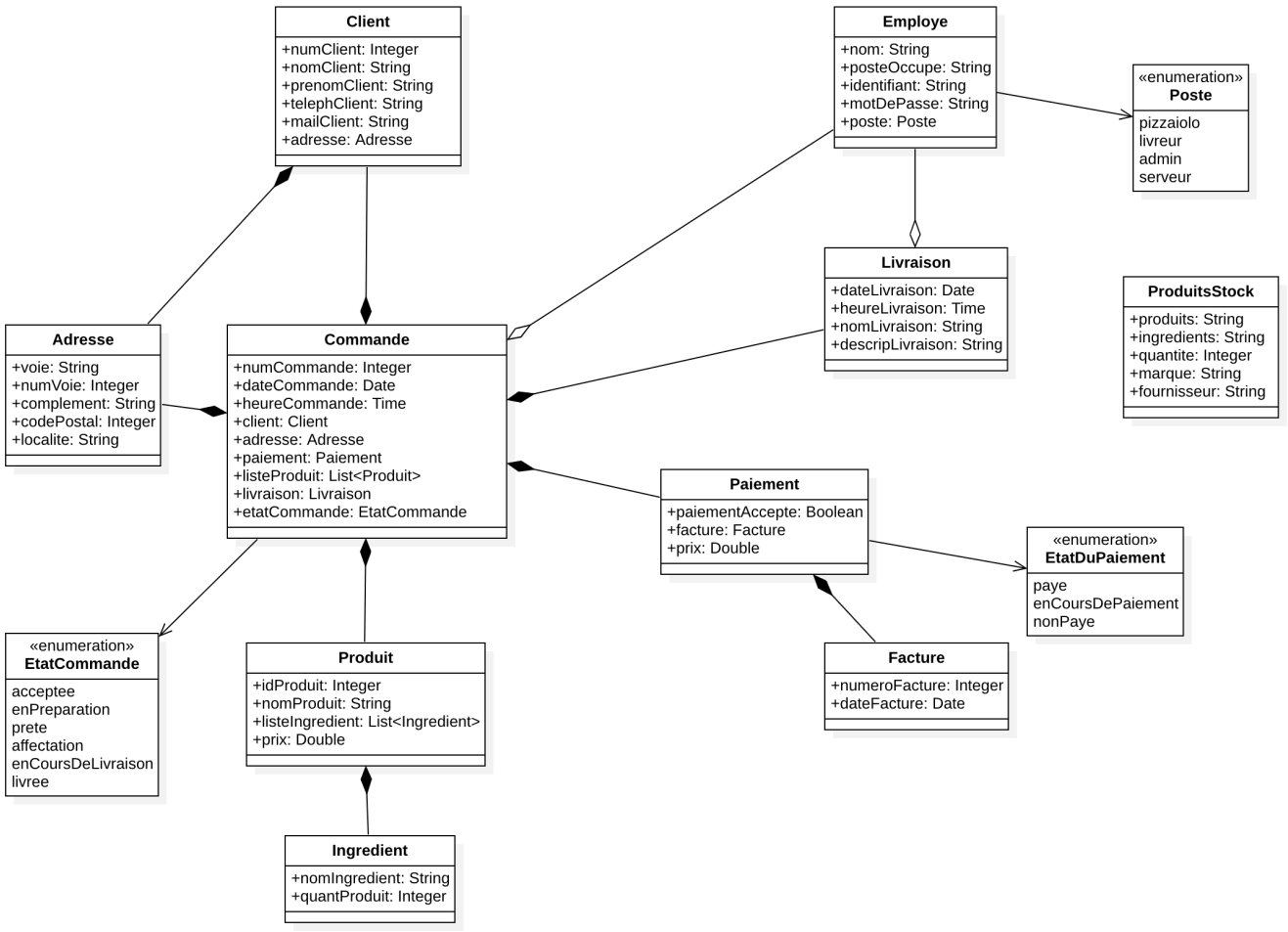
- Être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;
- Suivre en temps réel les commandes passées, en préparation et en livraison ;
- Suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas peuvent encore être réalisées ;
- Proposer un site Internet pour que les Clientes 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.
- Proposer un aide-mémoire aux pizzaiolos indiquant la recette de chaque pizza.

I. LE DOMAINE FONCTIONNEL

1. Description

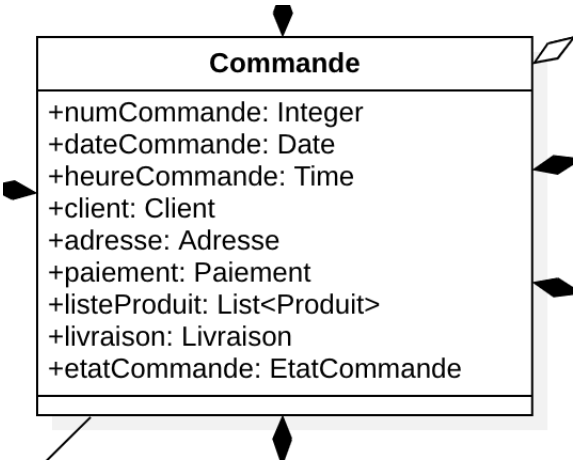
Le domaine fonctionnel est représenté par le diagramme de classe. Ce dernier vise à relier les classes entre elles, de manière à développer une application dynamique et en capacité de satisfaire la commande client.

2. Le diagramme de classe d'OC Pizza



3. Le descriptif des classes et de leurs interactions

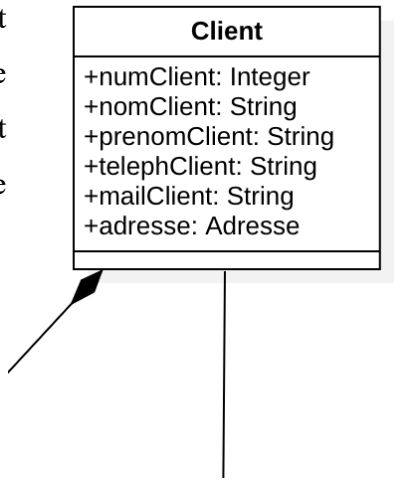
❖ La classe « Commande »

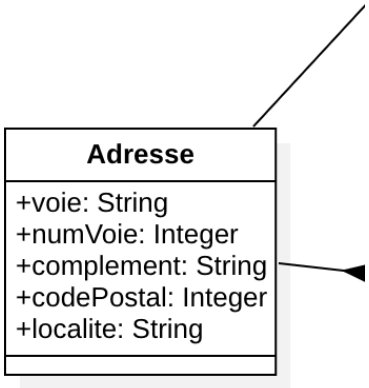


La classe « Commande » est « hyper sollicitée ». Elle est en fait la pierre angulaire de l'application : Les cinq clés étrangères « client », « adresse », « paiement », « livraison » et « etatCommande » permettent respectivement, de relier la commande au client l'ayant produite, la commande à l'adresse où doivent être livrés les produits, la commande en cours au paiement que celle-ci génère, la commande à sa livraison et enfin, la commande à son état, selon qu'elle est acceptée, en préparation, prête, affectée à un livreur ou bien en cours de livraison.

❖ Les classes « Client » et « Adresse »

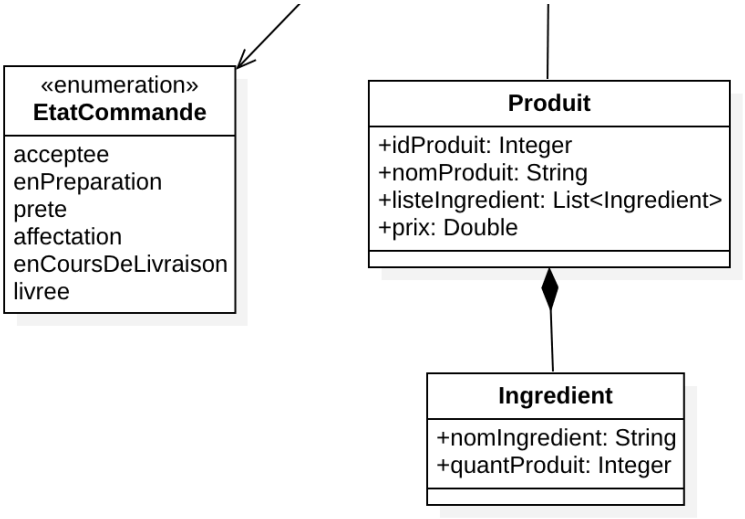
La classe « Client » répertorie : le numéro de client (attribut auto-incrémenté), son nom, son prénom, son téléphone ainsi que son adresse mail. La clé étrangère « adresse » permet de relier la classe « Client » à la classe « Adresse », de sorte que chaque client puisse avoir une adresse de livraison définie.





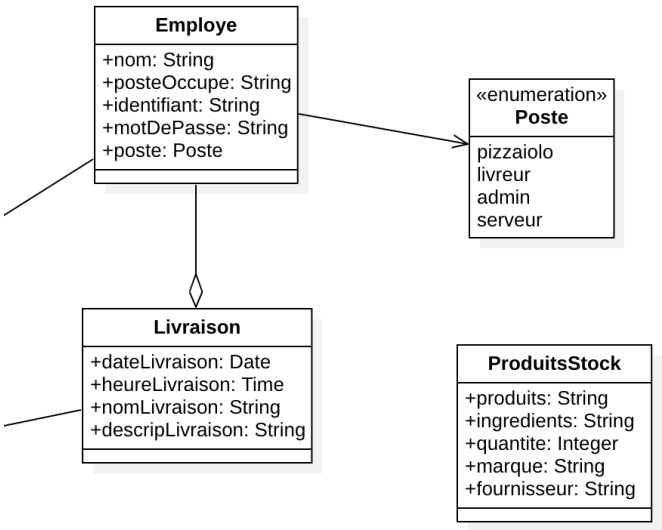
La classe « Adresse » contient pour chaque client une adresse spécifique, détaillée et unique (« one to one »).

❖ Les classes « Produit », « Ingedient » et l'énumération « EtatCommande »



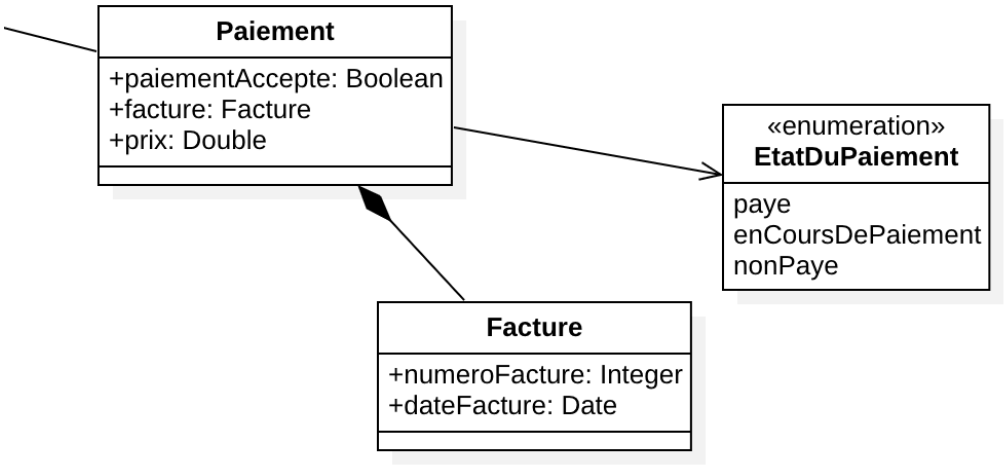
Les classes « Produit » et « Ingredient » permettent d'enregistrer les produits et ingrédients que le client choisit. L'énumération « EtatCommande » paramètre les différents changements d'états de la commande client.

❖ Les classes « Employe », « Livraison », « ProduitsStock » et l'énumération « Poste »



L'énumération « Poste » définit l'employé en charge de la commande et/ou de sa livraison. La classe « Employe » attribue pour chaque employé le nom, le poste occupé, l'identifiant ainsi que le mot de passe : L'ensemble de ces éléments permet, dès la connexion à l'application, une authentification. La classe « Livraison » enregistre la date, l'heure, le nom rattaché à la livraison ainsi que son descriptif. La classe « ProduitsStock » est plus particulièrement utilisée par l'administrateur dans le but d'effectuer une gestion de stock en cohérence avec les besoins et de pouvoir lancer une commande auprès des fournisseurs.

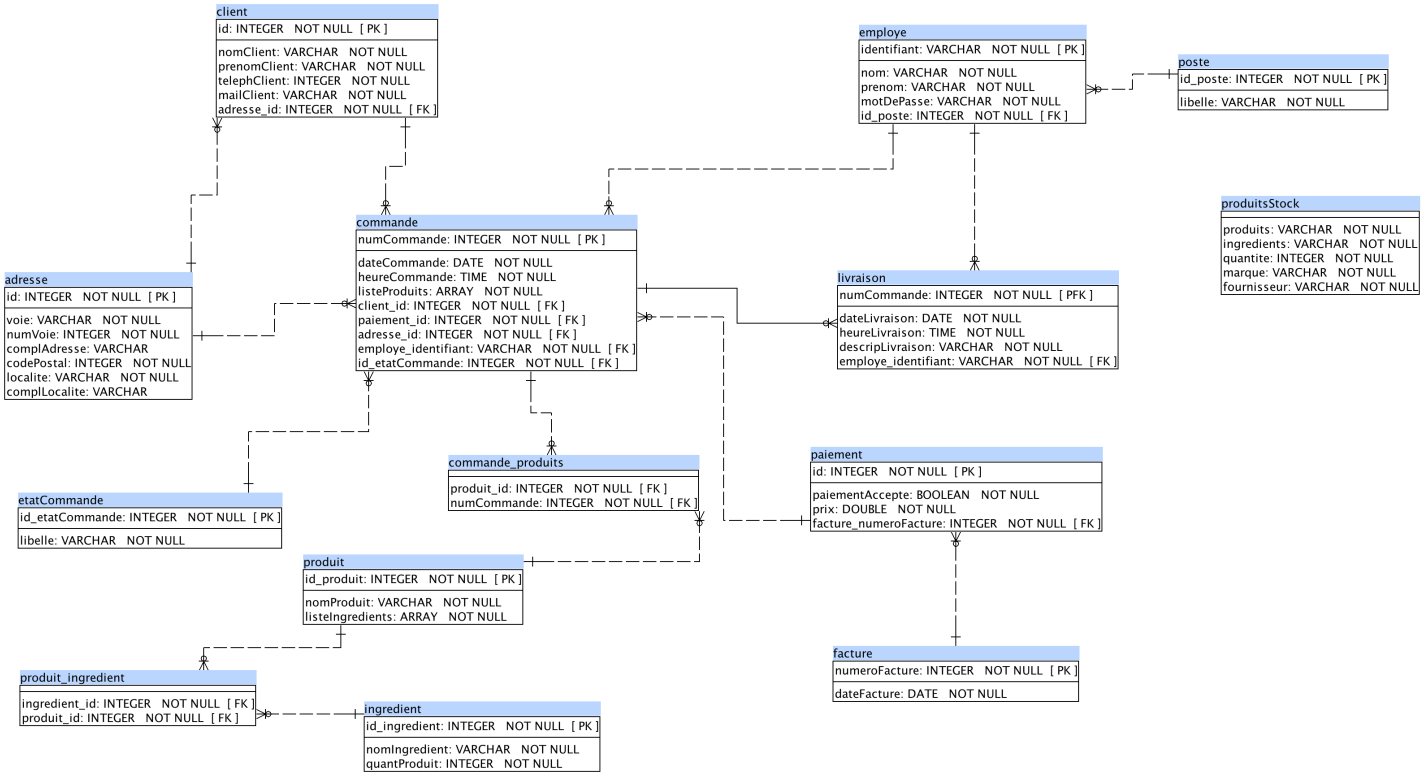
❖ Les classes « Paiement », « Facture » et l'énumération « EtatDuPaiement »



La classe « Paiement » finalise le paiement client dès lors que les différentes étapes du paiement sont validées. Ce paiement produit une facture, celle-ci établissant le coût de la commande. L'énumération « EtatDuPaiement » recense les différents états du paiement.

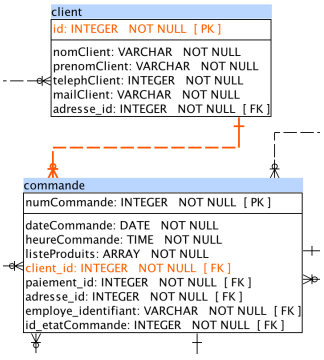
II. LE MODELE PHYSIQUE DE DONNEES

Le modèle physique de données est par excellence la représentation schématique en mesure de générer du code SQL, notamment par l'intermédiaire du logiciel SQL Power Architect, ce code pouvant construire une base de données PostgreSQL. Le modèle physique de données crée pour chacune des classes une clé primaire, celles-ci permettant d'établir des correspondances entre l'ensemble des classes.

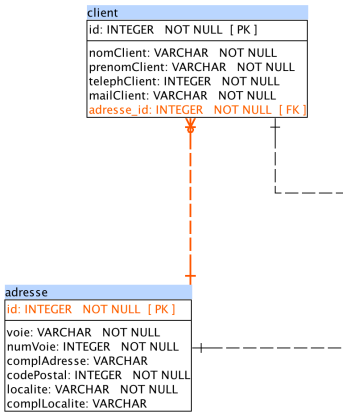


1. La relation entre les classes « client » et « commande »

Une commande possède un client (one to one) aussi, la classe « commande » comporte « client_id » en clé étrangère. Cette clé permet d'associer chaque client à une commande unique.

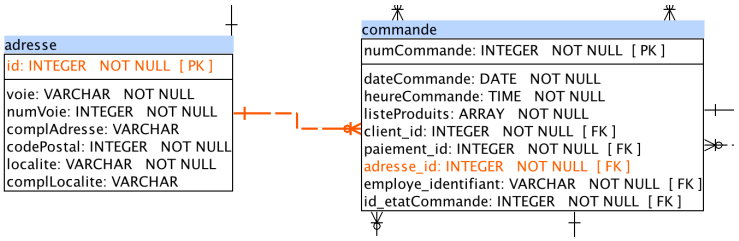


2. La relation entre les classes « client » et « adresse »



Chaque client possède une adresse unique. Celle-ci est enregistrée dans la classe « client » via la clé étrangère « adresse_id ».

3. La relation entre les classes « adresse » et « commande »

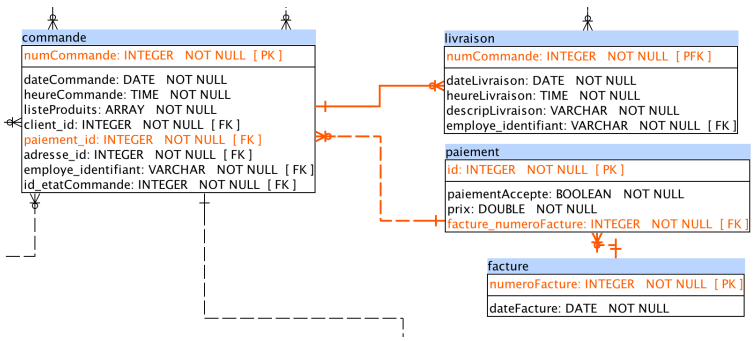


Chaque commande est reliée à une adresse unique. Pour ce faire, la classe « commande » possède la clé étrangère « adresse_id ».

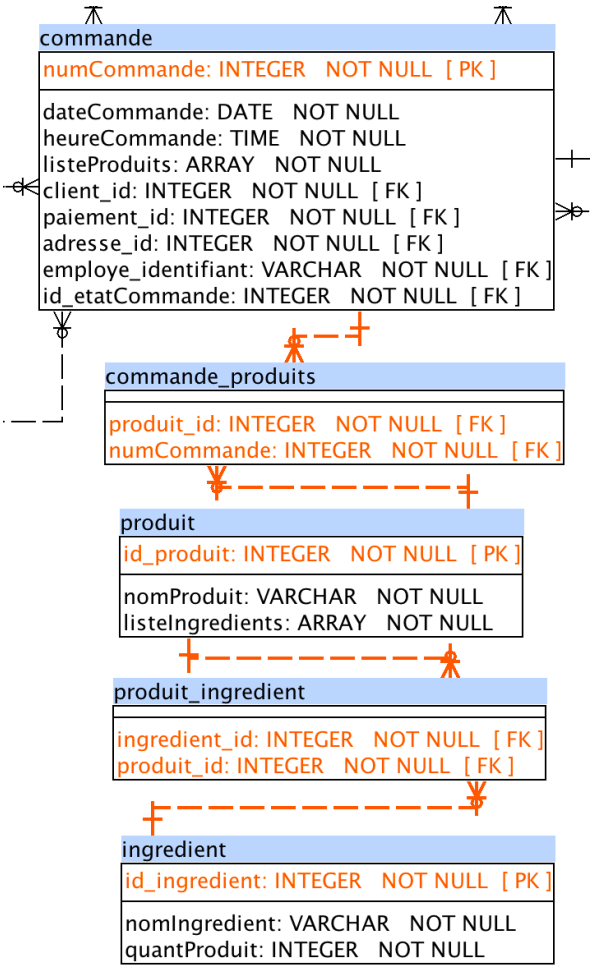
4. La relation entre les classes « commande » et « livraison », « commande » et « paiement » et « paiement » et « facture »

Une commande est en relation avec un paiement en particulier, de même que ce paiement est en référence avec une facture unique.

Dans la mesure où une livraison peut correspondre à plusieurs commandes (« one to many »), la classe livraison possède une clé à la fois primaire et étrangère. Celle-ci permet de regrouper les commandes en correspondance avec une livraison unique.

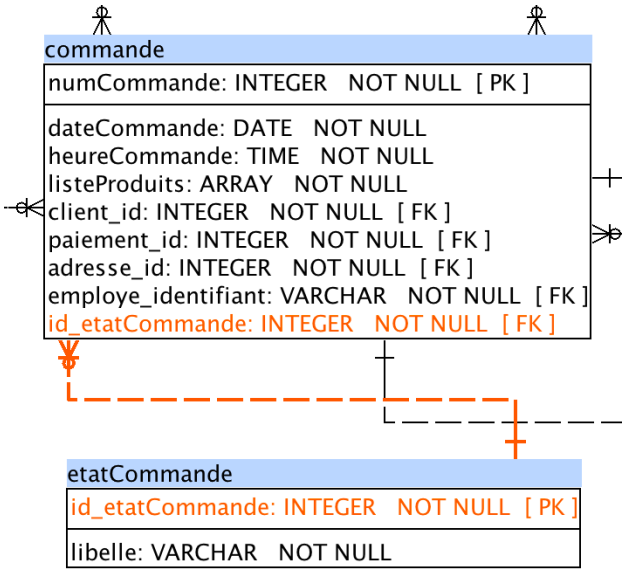


5. Les spécificités des relations entre les classes « commande », « produit » et « ingredient »



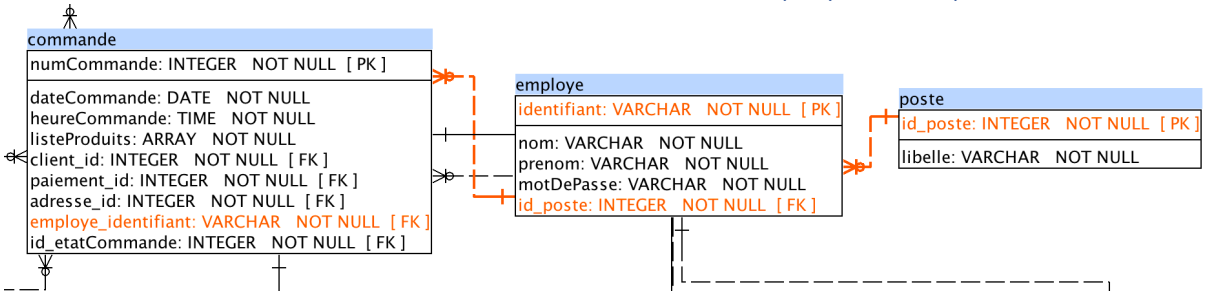
Plusieurs commandes comportent plusieurs produits et plusieurs ingrédients (« many to many »). Afin de mieux structurer les relations entre ces trois tables, il a été nécessaire de créer des tables de liaison : « commande_produits » et « produit_ingredient ». La table de liaison « commande_produits » permet d'enregistrer en clés étrangères les deux clés primaires « numCommande » (clé primaire de la table « commande ») et « produit_id » (clé primaire de la table « produit »). Cette table de liaison définit une première spécificité de la commande client. La seconde table de liaison « produit_ingredient » enregistre en clés étrangères « produit_id » (clé primaire de la table « produit ») et « ingredient_id » (clé primaire de la table « ingredient »). Cette deuxième table de liaison fournit une seconde spécificité de la commande client.

6. La relation entre les classes « etatCommande » et « commande »



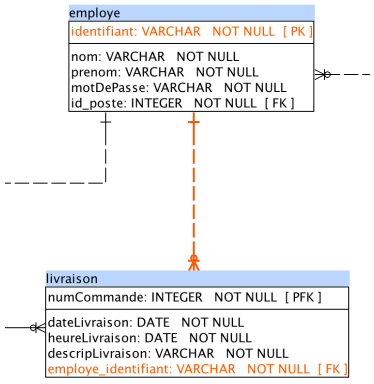
La classe « commande » intègre en clé étrangère « id_etatCommande » afin de spécifier pour chaque commande son état : acceptée ou en préparation ou prête ou affectée à un livreur ou en cours de livraison. Cet état permet au client de connaître la situation de sa commande et est spécifié dans l'attribut « libelle ».

7. La relation entre les classes « commande », « employe » et « poste »



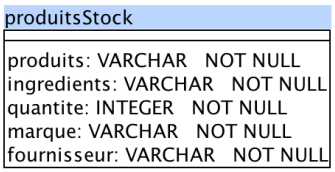
La classe « employe » intègre la clé étrangère « id_poste ». Cette clé étrangère fait référence à la classe « poste ». Cette classe désigne l'employé en charge de la commande ou de sa livraison. De son côté, la classe « commande » comporte la clé étrangère « employe_identifiant » de manière à récupérer l'identifiant de l'employé désigné en amont.

8. La relation entre les classes « employe » et « livraison »



La classe « livraison » possède la clé étrangère « employe_identifiant » afin de connaître l'employé en gestion de la commande.

9. La classe « produitsStock »

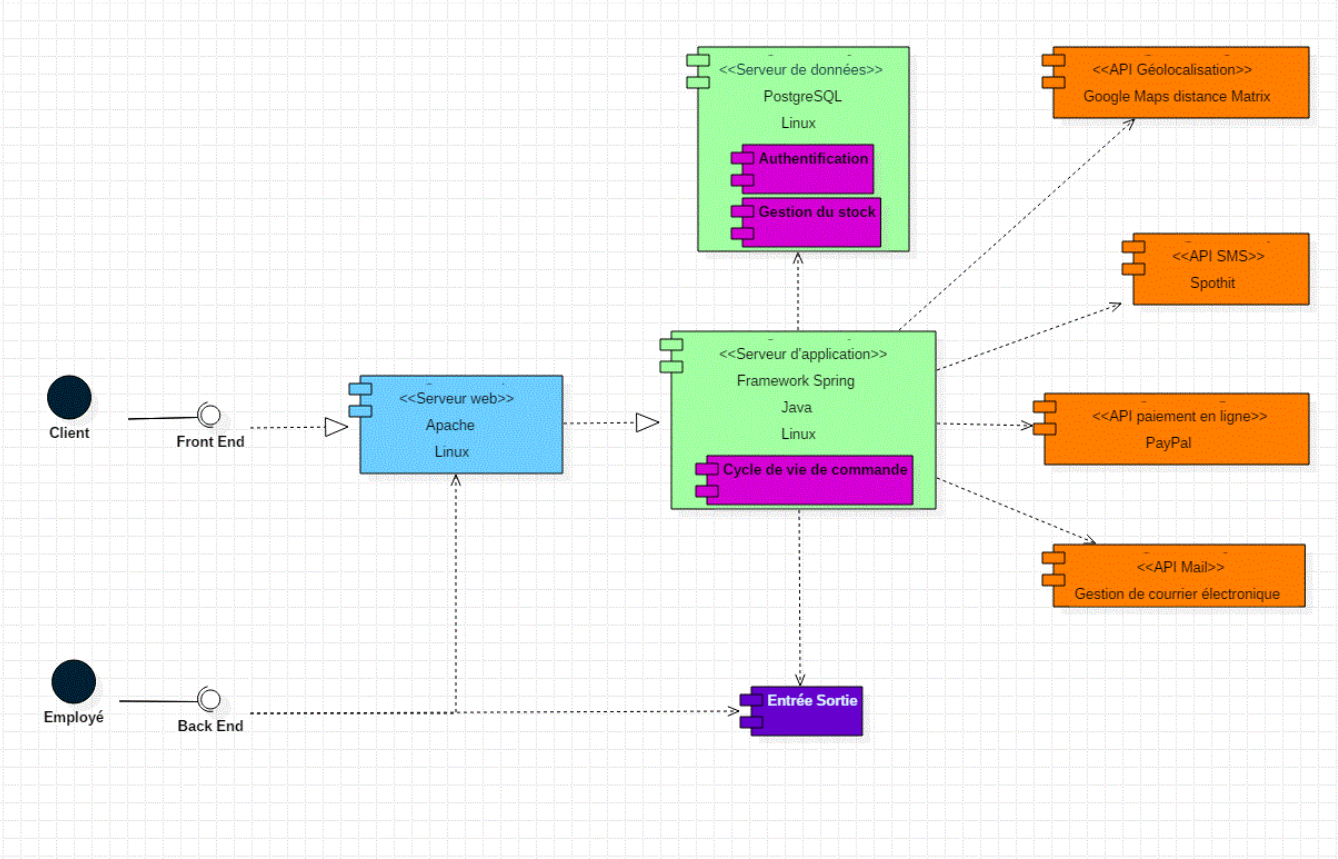


La classe « produitsStock » ne possède aucune clé primaire car celle-ci n'est en relation avec aucune classe. Son unique fonction consiste dans la gestion du stock, de son niveau suffisant et en concordance avec les besoins clients.

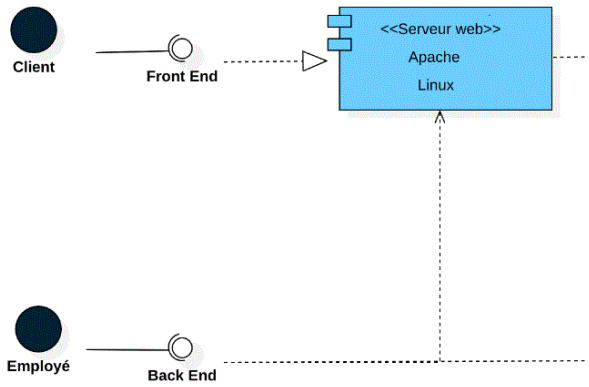
III. LES COMPOSANTS DU SYSTEME

Le diagramme de composant s'efforce de donner une représentation très proche de l'application telle qu'elle sera développée. Ce diagramme permet de commencer à repérer les contraintes de mise en œuvre ainsi que les facteurs facilitants.

1. Le diagramme de composants



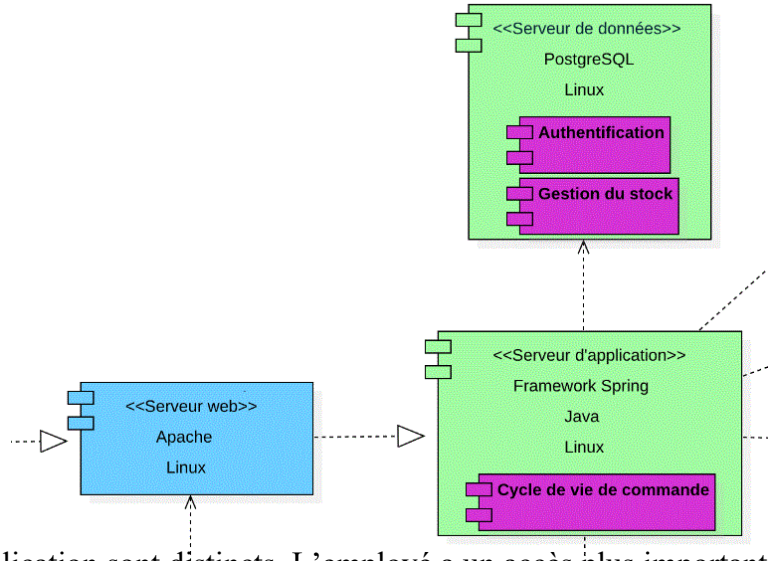
2. La partie connexion du diagramme



Le client se connecte à l'application via l'interface client. Il n'a pas accès au back-end. Le Serveur Web redirige sa connexion.

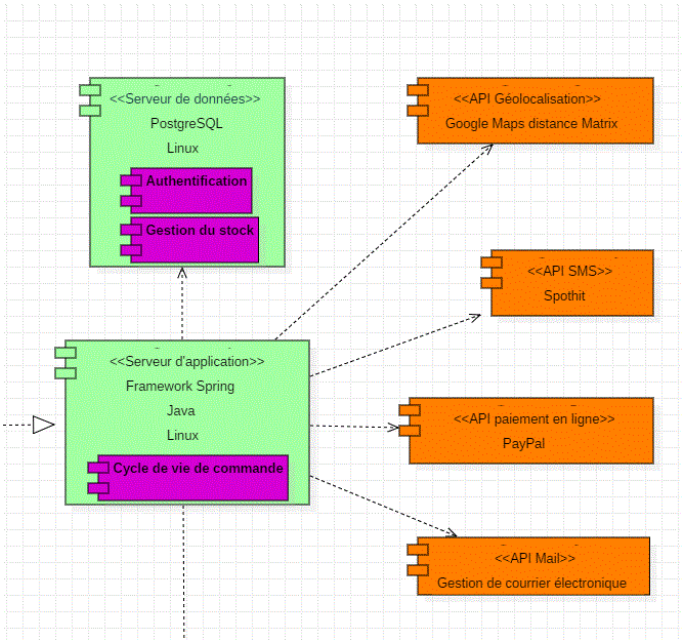
L'employé a un accès élargi à l'application. Cet accès lui permet notamment de contrôler les commandes ainsi que leur acheminement.

3. Les trois serveurs et leur interconnexion



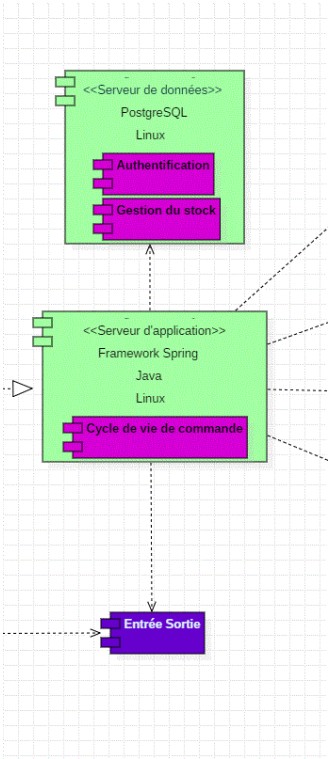
Le Serveur Web redirige la demande de connexion tout d'abord vers le Serveur d'Application. Ce dernier questionne le Serveur de Données, notamment pour contrôler l'identifiant et le mot de passe de l'utilisateur. Selon que l'utilisateur est un client ou un employé, les droits d'accès à l'application sont distincts. L'employé a un accès plus important et peut à tout moment contrôler le processus des commandes, par exemple : Le Pizzaiolo peut accepter une commande dans la mesure où les différentes étapes de paiement ont été franchies avec succès. L'Admin, dès lors qu'une commande est prête pour être livrée, attribue celle-ci à un livreur. L'Admin a également accès à la gestion du stock, contrôler ses niveaux et envoyer, si nécessaire, des commandes aux fournisseurs. Le livreur prend connaissance via son accès des commandes qui lui ont été attribuées.

4. Les APIs



Les **APIs** permettent une gestion flexible de l'application et du traitement des commandes client : *l'API paiement en ligne* gère spécifiquement le paiement des clients, *l'API Mail* permet à l'Admin d'effectuer des commandes auprès des fournisseurs afin de maintenir le stock à un niveau satisfaisant, *l'API Géolocalisation* permet à l'Admin d'attribuer une livraison au livreur à proximité de l'adresse client et enfin, *l'API SMS* permet au livreur de signaler « en live » que le client a réceptionné la commande.

5. L’accès employé

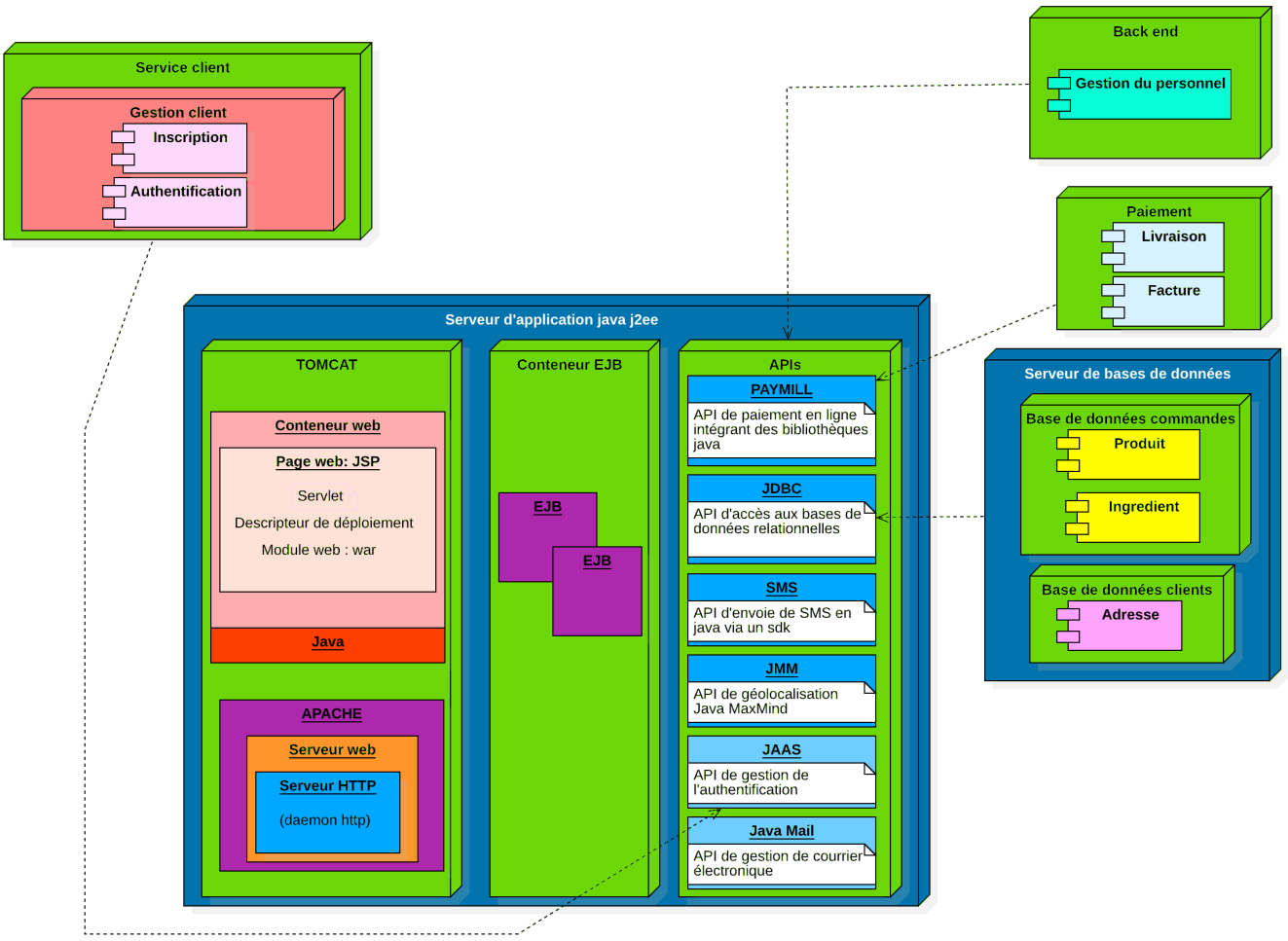


La connexion du composant « Entrée Sortie » en back-end permet aux employés de gérer, via l’application, les commandes clients, leurs états, les changements de ceux-ci, la vérification de la solvabilité du client (rejet du paiement en cas de non-solvabilité). Le serveur peut également établir des statistiques afin de mesurer la qualité des processus de ventes, ceci afin de repérer les points faibles, les améliorer et, plus globalement, améliorer le fonctionnement général de l’application en faisant remonter les observations, questionnements, etc.

IV. L'ARCHITECTURE DE DEPLOIEMENT

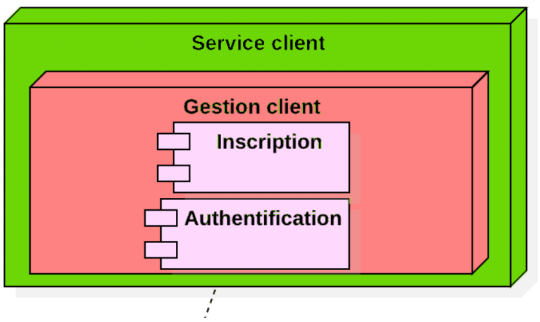
L'architecture de déploiement permet d'avoir un rendu visuel qui soit le plus proche possible de l'application telle qu'elle se présente.

1. Le diagramme de déploiement



❖ L'accès client

Cet accès permet aux clients de se connecter à l'application via une authentification ou une inscription si celle-ci n'a pas déjà été effectuée. Il y a une spécificité d'accès : le client n'aura pas accès aux back-end de l'application mais uniquement au front-end afin de commander les produits qu'il souhaite.



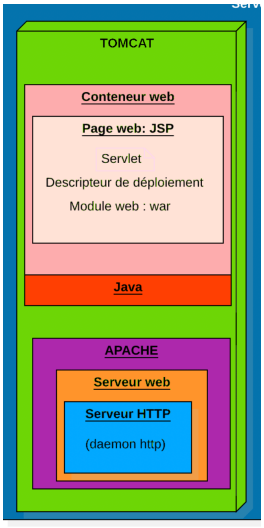
❖ Le serveur TOMCAT

Le serveur regroupe différentes couches, dont :

a. *Le serveur web Http*

Ce serveur qui gère exclusivement des requêtes HTTP. Il a pour rôle d'intercepter les requêtes Http, sur un port qui est par défaut 80, pour les traiter et générer ensuite des réponses Http. Tous les serveurs web embarquent un daemon Http (http) ou équivalent qui s'occupe de cette fonctionnalité.[1]

b. *Le conteneur Web*

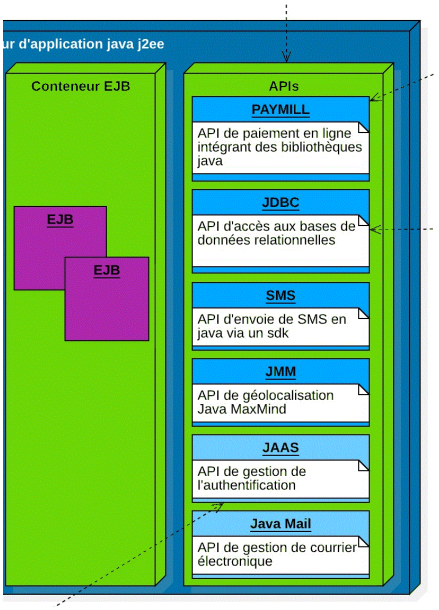


Cette extension va permettre d'avoir la possibilité d'exécuter des programmes écrits avec des langages de programmation (java, PHP, C# ou autres) dans le serveur web : Par exemple le serveur Tomcat n'est autre qu'un serveur Apache couplé avec un moteur web java.[2]

❖ Le serveur d'application

Il est composé de :

1. **Un conteneur EJB** qui encapsule les traitements des Entreprise JavaBeans.
2. Un ensemble de **services** d'infrastructures et de communication :
   i. PAYMILL : API de paiement en ligne intégrant des bibliothèques Java.
   ii. JDBC (Java DataBase Connectivity) API d'accès aux bases de données relationnelles.



[1] Source : https://www.supinfo.com/articles/single/1156-difference-serveur-web
[2] Ibid.

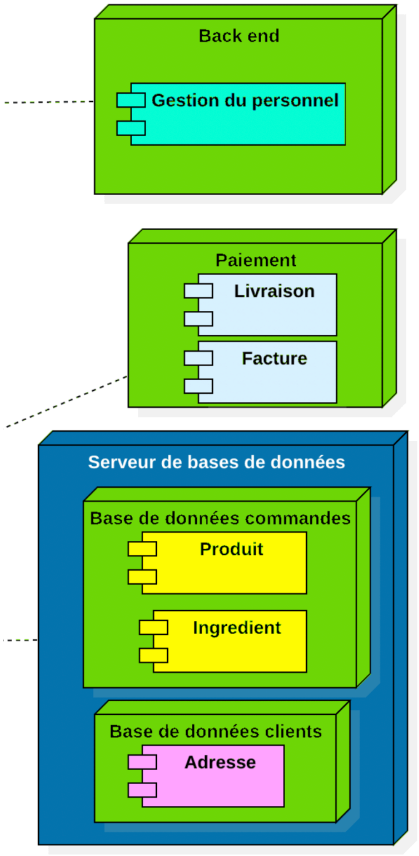
iii. SMS : API d'envoi de SMS en java via un SDK.

iv. JMM : API de géolocalisation « Java MaxMind ».

v. JAAS (Java Authentication and Authorization Service) API de gestion de l'authentification.

vi. JavaMail API pour la gestion de courrier électronique.[3]

❖ Le back-end



La partie cachée et non accessible aux clients permet de gérer leurs commandes, de les enregistrer, de les traiter. Le back-end permet également aux employés de se connecter à l'application, de prendre connaissance de l'avancée du processus de vente, de l'intercepter si nécessaire, d'effectuer un suivi en temps réel des commandes clients, etc.

[3] Ibid.