Projet 6

Concevez la solution technique d’un système de gestion de pizzeria

Document de spécifications techniques

Clémence ROBIN

HISTORIQUE DES RÉVISIONS DU DOCUMENT

Version	Date	Modifications	Auteur
0.1	17/04/20	Rédaction des spécifications techniques	Clémence ROBIN

DIFFUSION

Destinataires	Pour validation	Pour information

VALIDATION

Date	Validateur	Représentant

Table des matières

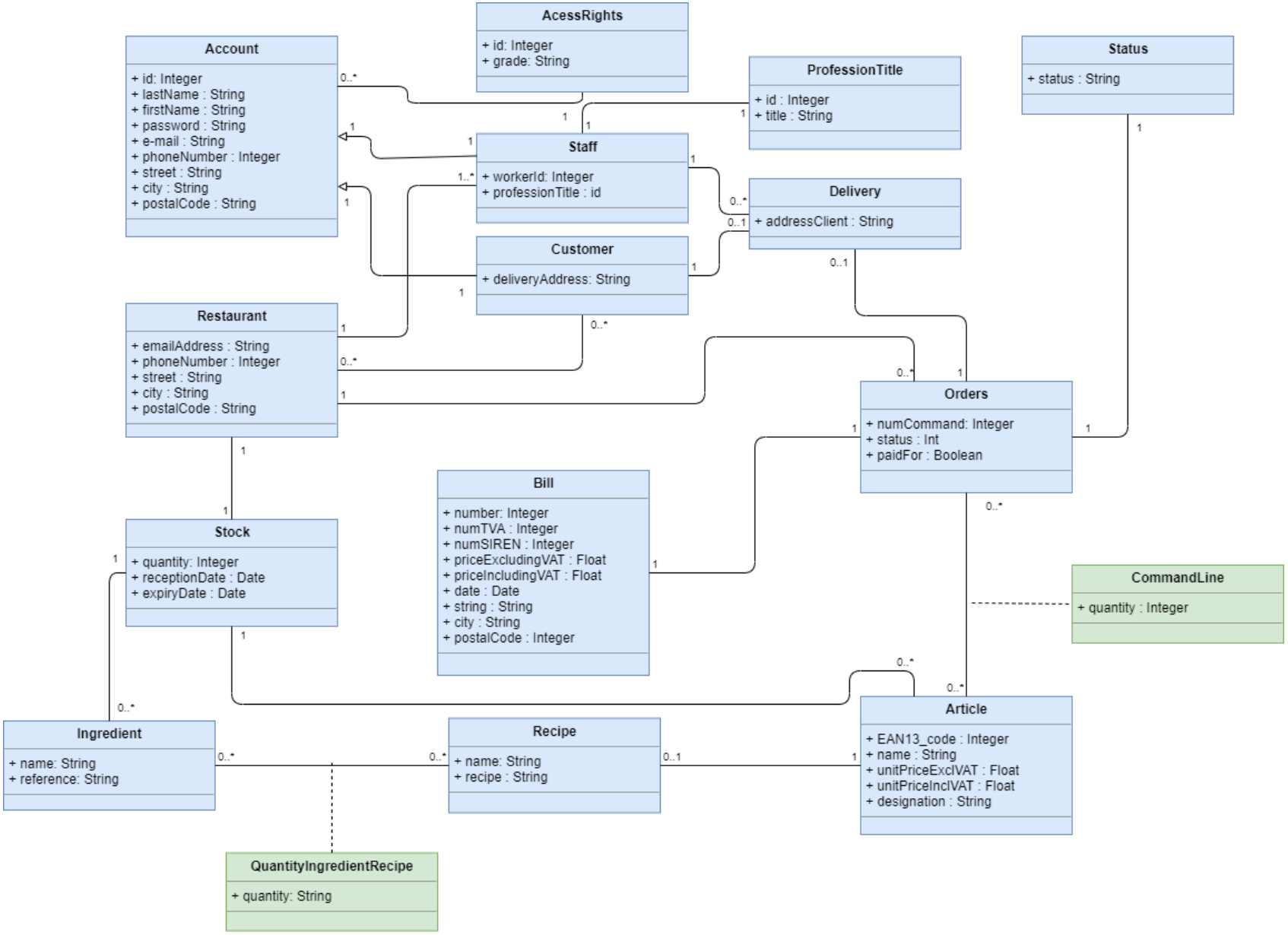
1 – Objectif du document

Ce document détaille l’architecture technique du système de gestion de pizzeria. Il s’adresse :

- Aux développeurs du projet
- Au chargé de projet

2 – Description des données

2.1 – Diagramme de classe



2.2 – Description du diagramme de classe

Nous revenons ici sur les classes principales et clarifions les relations qu'elles peuvent avoir entre elles. Lorsqu'une relation est explicitée dans la description d'une classe A envers une classe B, nous ne reviendrons par sur cette relations lors de la descriptions de la classe B.

Restaurant

Représente les restaurants de OC Pizza.

Chaque stock peut varier selon le restaurant : ainsi, chaque instance de la classe restaurant est associée à une instance de la classe Stock par le biais d'une relation One-to-one.

De même, chaque commande doit être associée à un restaurant : on obtient donc une association One-to-many de Restaurant vers Orders.

Un client peut passer commande dans plusieurs restaurants différents, et un restaurant peut avoir plusieurs clients : on obtient donc une association Many-to-many entre une instance de la classe Restaurant et une instance de la classe Customer.

Un restaurant aura plusieurs membres du personnel associés. En revanche, chaque membre du personnel est associé à un restaurant (exception du Directeur). On obtient donc une association de type One-to-many entre une instance de la classe Restaurant et une instance de la classe Staff.

Stock

Représente les stocks d'articles et d'ingrédients disponibles.

Les stocks peuvent contenir plusieurs ingrédients et plusieurs articles (nous entendons par ingrédient une matière première et par article un article prêt à être consommer (ex : Bouteille d'eau). Pour ce faire, on obtient une relation de type One-to-many d'une instance de la classe Stock vers une instance de la classe Ingredient ou une instance de la classe Article.

Article

Représente les articles vendus par l'entreprise.

Un article peut être acheté et revendu (ex : cannettes de boisson) ou être préparé par le cuisinier : une instance de la classe Article aura ainsi une relation de type One-to-one avec une instance de la classe Recette.

Un article peut ne pas être commandé, être commandé par une ou plusieurs commandes. Elle obtient ainsi une relatuon de type Many-to-many par l'intermédiaire de la classe d'association CommandLine.

CommandLine (classe d'association)

Classe d'association entre la classe Orders et la classe Article

Cette classe d'association sert de lien entre la classe Orders et la classe Article, notamment en précisant la quantité commandée d'un article pour une commande donnée.

Orders

Représente les commandes passées par les clients au restaurant

Chaque commande aura un statut d'avancement de préparation (« En cours de prépatation », « Livrée » …). Chaque instance de la classe Orders a donc une relation de type One-to-one avec une instance de la classe Status.

De même, chaque commande ne peut être associée qu'à une livraison. Elle a aura donc une relation de type One-to-one vers une instance de la classe Delivery.

Enfin, chaque commande aura sa facture correspondante. Nous avons donc besoin d'une relation de type One-to-one avec une instance de la classe Bill.

Account

Représente les comptes du personnel et des clients

Un compte appartiendra soit à un client, soit à un membre du personnel. Une instance de la classe Account aura donc une relation de type One-to-tone avec les instances de la classe Staff ou Customer.

Les comptes pouvant donc appartenir soit à un membre du personne, soit à un client, les droits d'utilisateurs ne seront donc pas les mêmes. Mais ils peuvent aussi différer d'un membre du personnel à l'autre (Manager ou Cuisinier). Nous avons donc créé une classe AccessRights donc les instances seront associées aux instances de la classe Account.

En effet, un compte sera associé à un droit, mais un droit peut être possédé par plusieurs comptes. Nous obtenons donc une relation de type Many-to-one.

Customer

Hérite de la classe Account

Représente le client

Le client ayant passé commande aura donc une livraison associée : on obtient une relation One-to-tone entre une instance de la classe Customer et une instance de la classe Delivery.

Staff

Hérite de la classe Account

Représente le personnel de l'entreprise

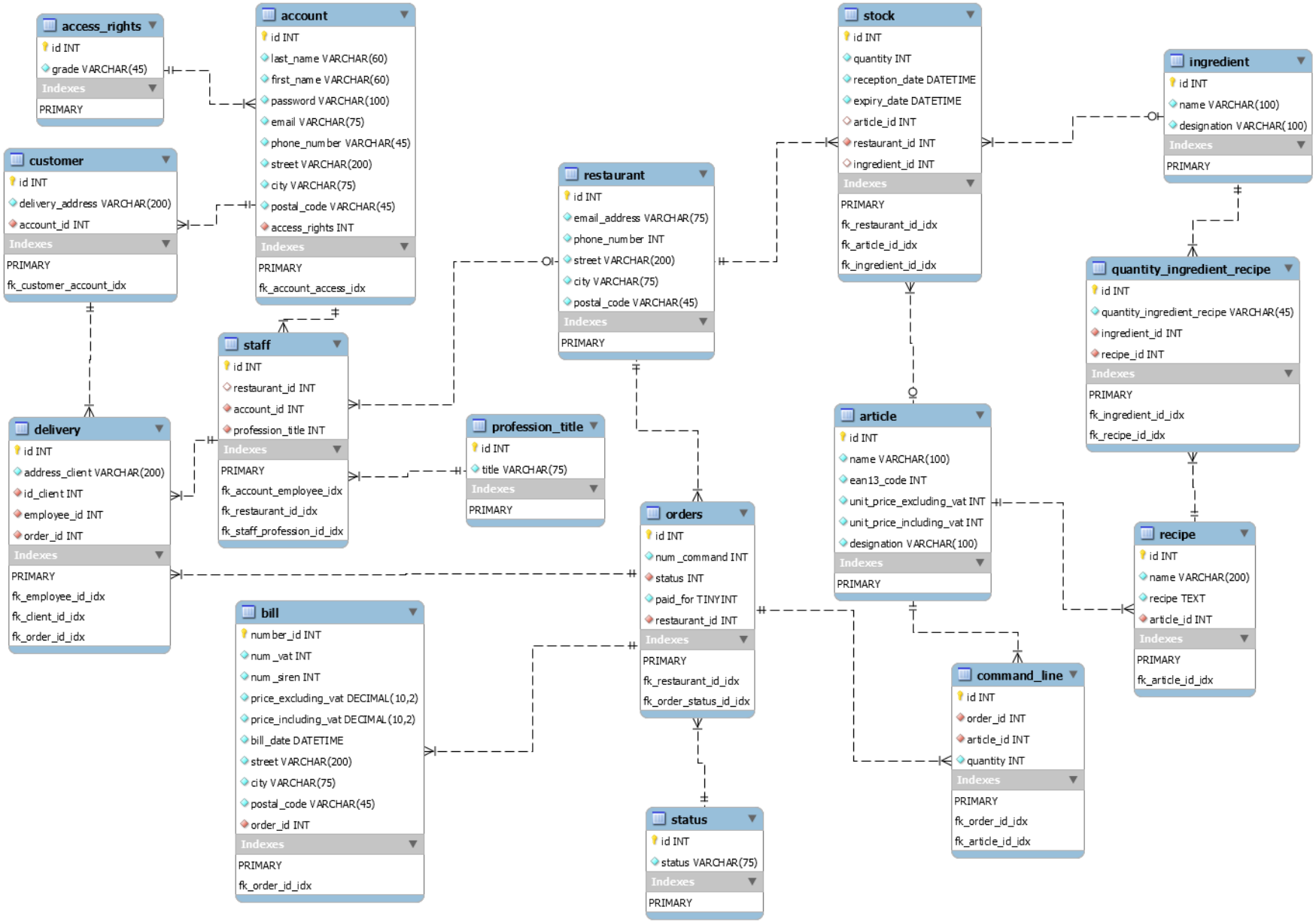
Un livreur peut avoir une ou plusieurs livraisons à effectuer, mais chaque livraison sera faite par un livreur uniquement : on obtient donc une relation de type One-to-many entre une instance de la classe Staff et une instance de la classe Delivery.

QuantityIngredientRecipe

Classe d’association entre la classe Ingredients et la classe Recipe

Cette d’association sert à spécifier les quantités nécessaires d’un ingrédient donné pour une recette donnée.

2.3 – Modèle physique de données



2.4 – Descriptions du modèle physique de données

Nous revenons ici sur les tables principales et clarifions les clés qui les lient entre elles. Lorsqu'une clé est abordée dans la description d'une table A envers une table B, nous ne reviendrons par sur cette relations lors de la description de la table B.

Stock

Les champs articles_id et ingredient_id peuvent tous les deux être *null*. En effet, une instance du stock peut concerner un ingredient (matière première) ou un article (ex :cannette de soda). L'un des deux champs sera donc *null*.

La table contient trois clés étrangères :

- **fk_stock_restaurant_id** : entre le champ restaurant_id de la table Stock et le champ id de la table Restaurant
- **fk_stock_article_id** : entre le champ article_id de la table Stock et le champ id de la table Article
- **fk_stock_ingredient_id** : entre le champs ingredient_id de la table Stock et le champ id de la table Ingredient

Recipe

Une recette doit correspondre à un article préparé par le restaurant, donc à un instance de la classe Article. Elle aura donc une clé étrangère entre le champ article_id et le champ id de la table Article :

- **fk_recipe_article_id** : entre le champ article_id de la table Recipe et le champ id de la table Article

Orders

Chaque commande doit être associée à un restaurant et chaque commande doit être associée à un statut. La table Orders contient donc deux clés étrangères :

- **fk_order_restaurant_id** : entre le champ restaurant_id de la table Orders et le champs id de la table Restaurant
- **fk_order_status_id** : entre le champ status de la table Orders et le champs id de la table Status

Command_line

La table Command_line sert à préciser le nombre d'article commandé pour une commande donnée. Elle doit donc être associée à l'id de l'article et l'id de la commande. La table contient par conséquent deux clés étrangères:

- **fk_command_order_id** *:* entre le champ order_id de la table Command_line et le champ id de la table Orders

- **fk_command_article_id** *:* entre le champ article_id de la table Command_line et le champ id de la table Article

Bill

Chaque facture devant être associée à une commande, la table Bill contient donc une clé étrangère :

- **fk_bill_order_i**d : entre le champ order_id de la table Bill et le champ id de la table Orders

Account

Les comptes pouvant être associés à des clients ou bien à des employés ayant des droits différents, chacun d'entre eux doit être associé à un droit spécifique. Nous avons donc une clé étrangère dans la table Account :

- **fk_account_acce**ss : entre le champ access_rights de la table Account et le champ id de la table AccessRights

Quantity_ingredient_recipe

Cette table servant à indiquer les quantités d'un ingrédient donné pour une recette donnée, nous avons choisi de donner au champ quantit_ingredient_recipe le type VARCHAR : cela permet d'insérer d'autres types de données que des chiffres, par exemple : « Un zeste de » ou « Une cuillère à soupe de ... ». La table quantiy_ingredient_recipe faisant le lien entre un ingrédient et une recette, elle contient par conséquent deux clés étrangères :

- **fk_quantity_ingredient_id** : entre le champ ingredient_id de la table Quantity_ingredient_recipe et le champ id de la table Ingredient
- **fk_quantiry_recipe_id** : entre le champ recipe_id de la classe Quantity_ingredient_recipe et le champ id de la table Recipe

Customer

Chaque client passant commande doit avoir un compte associé. Nous avons donc une clé étrangère dans la table Customer :

- **fk_customer_account_id** : entre le champ account_id de la table Customer et le champ id de la table Account

Staff

Nous avons choisi de laisser la possibilité au champ restaurant_id d'avoir la valeur NULL, selon si un employé, comme un livreur, travaille dans un ou plusieurs restaurants. De plus, le directeur n'est pas associé à un restaurant en particulier.

Chaque membre du personnel possède un compte et est associé à un intitulé de profession. De même, si le champ restaurant_id n'est pas *null*, il doit être associé à un restaurant donné. Nous avons donc trois clés étrangères :

- **fk_staff_account_id** : entre le champ account_id de la table Staff et le champ id de la table Account
- **fk_staff_restaurant_id** : entre le champ restaurant_id de la table Staff et le champ id de la table Restaurant
- **fk_staff_profession_id** : entre le champ profession_id de la table Staff et le champ id de la table Profession_title

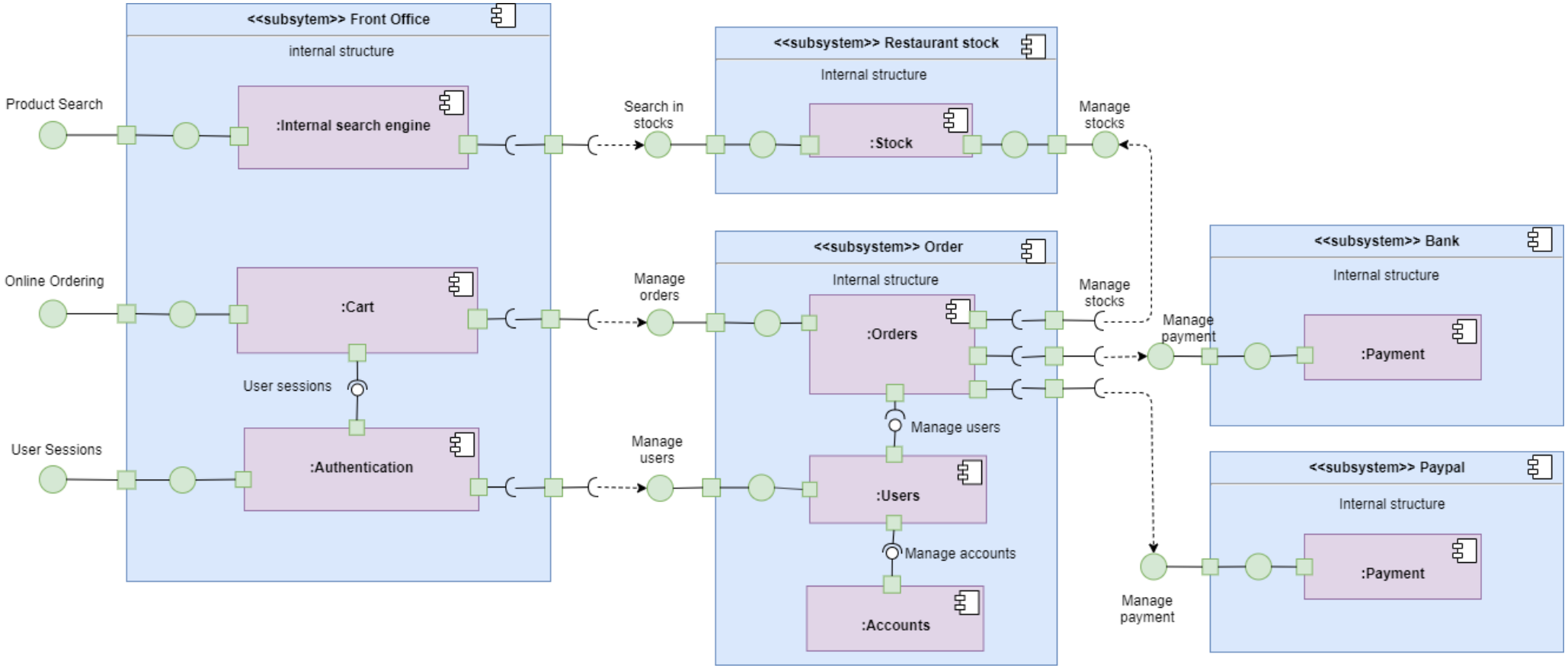
Delivery

Chaque livraison correspondant à une commande étant livrée à un client par un livreur donné, nous avons trois clés étrangères dans la table Delivery :

- **fk_delivery_employee_id** : entre le champ employee_id de la table Delivery et le champ id de la table Staff
- **fk_delivery_client_id** : entre le champ id_client de la table Delivery et le champ id de la table Customer
- **fk_delivery_order_id** : entre le champd order_id de la table Delivery et le champ id de la table Orders

3 – Composants

3.1 – Diagramme de composants



3.2 – Description du diagramme de composants

Le site web est représenté par un sous-système contenant lui même trois composants : *Internal search engine*, *Cart* et *Authentication*. Le composant *Internal search engine* expose l'interface *Product search*, permettant ainsi de rechercher les produits en utilisant l'interface *Search in stocks* fournie par le composant *Stock* du sous-système *Restaurant Stock*.

Le composant *Cart* utilise l'interface *Manage Orders* founie par le composant *Orders* lors du paiement et fournit l'interface *Online Ordering*.

Le composant *Authentication* permet quant à lui aux utilisateurs de créer un compte, s'authentifier ou se déconnecter en fournissant l'interface *User sessions*.

Le sous-système *Order* fournit deux interfaces, *Manage users* et *Manage orders*. Les connecteurs de délégation lient les objets externes à ce sous-système aux composants *Orders* et *Users* appartenant à ce système.

Le sous-système *Restaurant Stock* fournit deux interfaces *Search in stocks* et *Manage Stocks*.

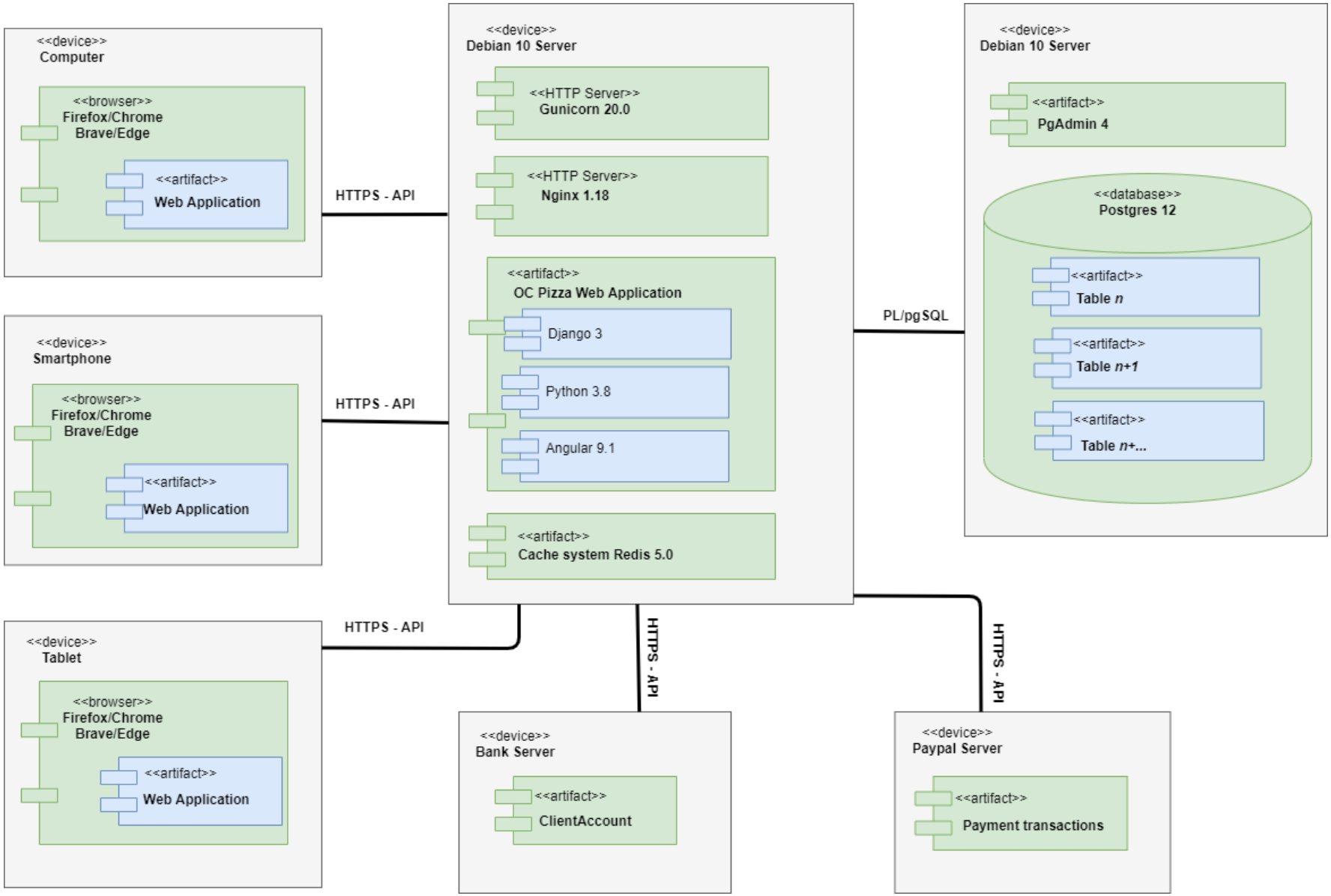
Les deux sous-système *Bank* et *Paypal* fournissent chacun une interface *Manage Payment* : l'interface *Manage Orders* exposée par le composant *Users* peut ainsi joindre les interfaces *Manage Payment*.

Le parcours d'une commande pourrait donc être le suivant :

- L'utilisateur est sur le site web (Front office).
- Il recherche des produits : le moteur de recherche fait appel aux stocks pour vérifier la disponibilité de ces derniers.
- L'utilisateur rajoute des produits dans le panier.
- Il passe commande : il n'est pas authentifié donc la commande ne peut pas être liée à un utilisateur.
- Il s'authentifie.
- Il tente à nouveau de passer commande.
- Le système de commande lui demande ses données bancaires ou les informations de son compte Paypal.
- En cas de succès, les produits commandés sont retirés de l'inventaire du stock.

4 – Déploiement

4.1 – Diagramme de déploiement



4.2 – Description du diagramme de déploiement

L'utilisateur peut avoir accès à l'application par le biais de différents appareils devant être connectés à internet et posséder un navigateur web pour que ce dernier puisse faire les requêtes au serveur qui contient l'application par le biais du langage de protocole HTTPS et l'API de l'application web.

Nous aurons deux serveurs fonctionnant avec le système d'exploitation Debian 10 :

1. Un serveur Debian qui hébergera :
    - La base de données PostgreSQL 12
    - La plateforme d'administration de base de données Pgadmin 4, qui facilitera la gestion de la base de donnée PostgreSQL.
2. Un autre serveur Debian qui hébergera :
    - Le serveur web HTTP Nginx (low-client) pour traiter les contenus statiques et agir en tant que proxy invserse. Il reçoit les requêtes HTTP et les transfère au serveur d'application Gunicorn. Il fait aussi parvenir les réponses HTTP de l'application Django au navigateur web du client.
    - Le serveur d'application HTTP Gunicorn (fast-client) qui travaillera en association avec Nginx. Lorsque Nginx recevra une requête dynamique, il transférera la requête à Gunicorn via un socket.
    - L'application web du restaurant, fonctionnant avec le framework Django 3. Django reçoit les requêtes de Nginx et lui renverra les réponses HTTP. Angular pourra être intégré au sein de Django pour déployer une interface *front-end*.
    - Le système de cache Redis, qui servira à réduire le temps de chargement de l'application lorsque la machine aura déjà les informations en mémoire cache.

Les deux serveurs communiqueront avec le langage de procédure chargeable PL/pgSQL pour le système de base de données PostgreSQL.

Le serveur hébergeant l'application pourra communiquer avec deux autres serveurs afin d'effectuer les transactions bancaires :

- Le serveur de la banque cible qui effectuera les procédures de vérifications et de transactions, par le le biais du langage de protocole HTTPS et de l'API fournie par la banque
- Le serveur de Paypal qui effectuera les procédures de vérifications et de transactions, par le le biais du langage de protocole HTTPS et de l'API fournie par Paypal