Projet 6

Dossier de spécifications techniques.

LEKISHVILI Rati
(Analyste-programmeur)

Table des matières

I. VERSIONS

Auteur	Date	Description	Version
LEKISHVILI Rati	01/02/2021	Création du document de spécifications techniques	1.0

II. INTRODUCTION

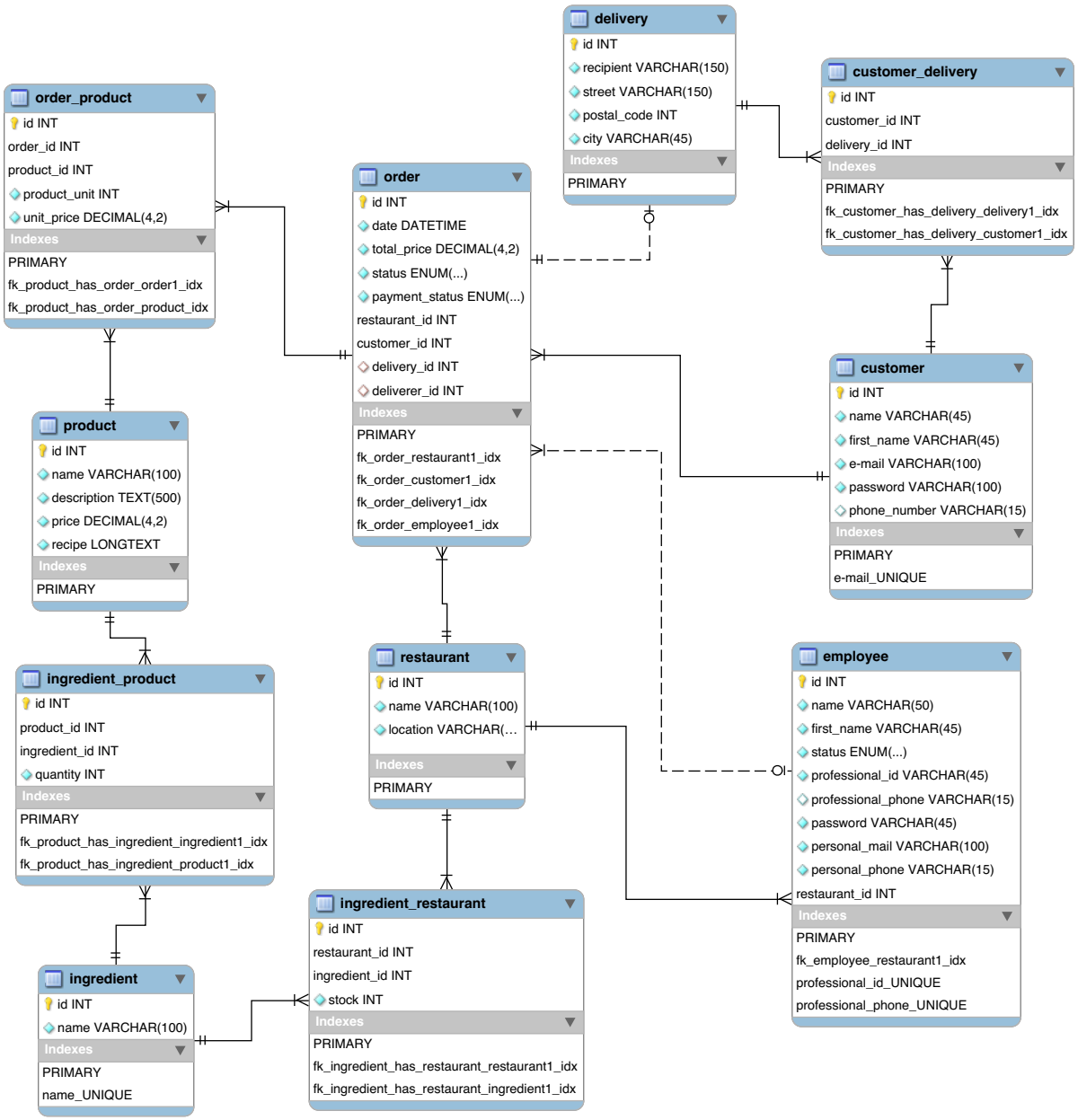
A. Rappel du contexte

OC Pizza est un groupe spécialisé dans les pizzas livrées ou à emporter. Les responsables souhaitent mettre en place un nouveau système informatique dans tous leurs restaurants, pour être plus efficace dans la gestion des commandes ainsi que dans le suivi et la traçabilité. Le groupe souhaite aussi développer un site internet, pour permettre aux clients de commander en ligne, suivre leurs commandes et effectuer les règlements en toute sécurité.

B. Objet du document

Le présent document constitue le dossier de spécifications techniques de la prochaine application d'OC Pizza. L'objectif du document est de définir le domaine fonctionnel et de concevoir l'architecture technique de la solution répondant aux besoins du client.

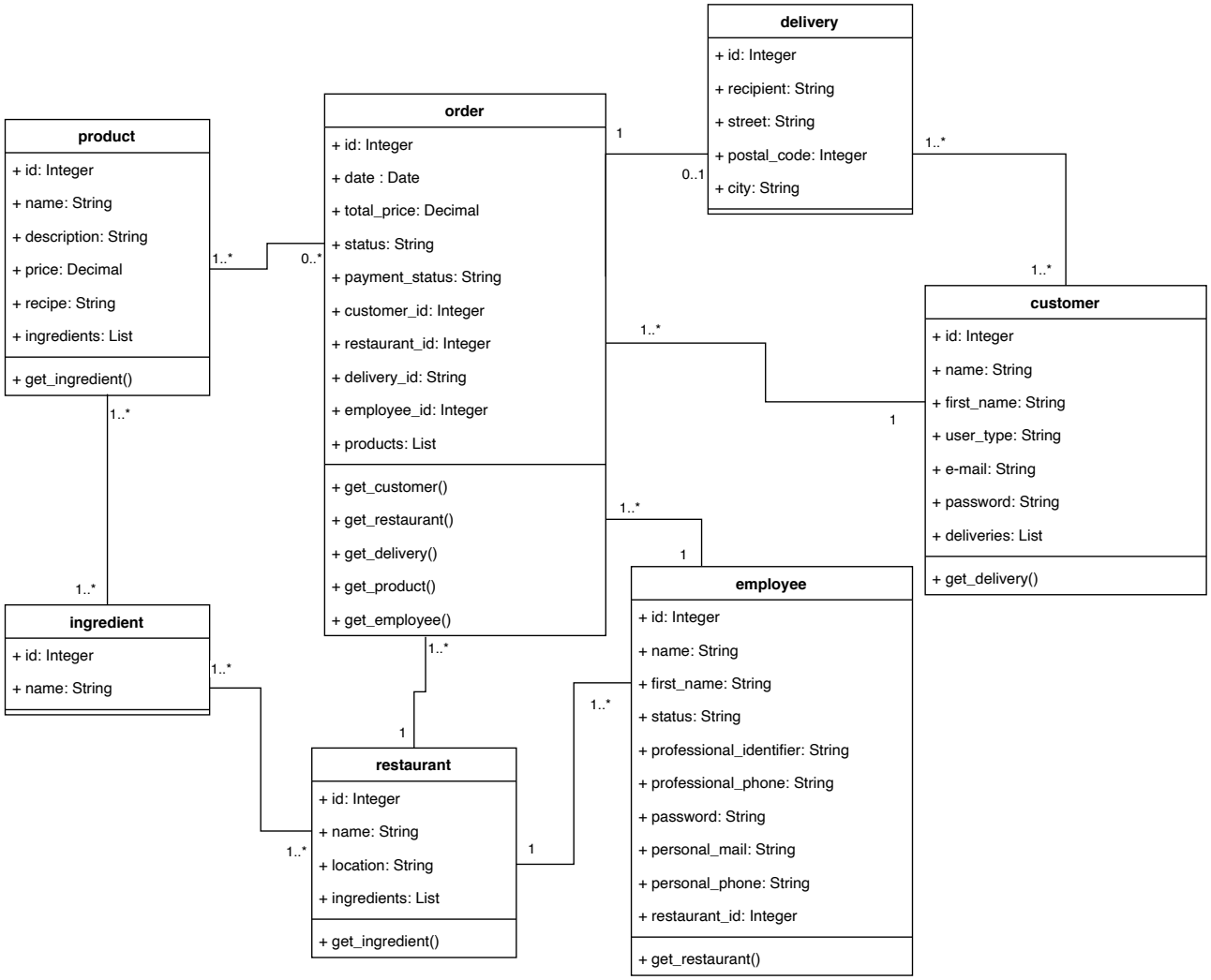
III. MODÈLE PHYSIQUE DE DONNÉES



IV. LE DOMAINE FONCTIONNEL

A. Diagramme de classes

Diagramme UML de classes



Le diagramme de classes (UML) est utilisé lors de la conception d'un logiciel pour représenter les classes ainsi que leurs relations et cardinalités. Ce dernier permet de montrer la structure interne du système, en fournissant une représentation abstraite des différents objets composants le domaine fonctionnel.

1. Règles de gestion

Définition des classes

* **order** : représente les commandes passées par un client.
* **product** : désigne les produits disponibles au sein des pizzérias.
* **ingredient** : regroupe tous les produits disponibles au sein des restaurants.
* **restaurant** : représente les restaurants d'OC Pizza.
* **employee** : désigne les employés des restaurants du groupe.
* **customer** : regroupe les clients.
* **delivery** : représente toutes les adresses de livraisons renseignées.

Relations entre les classes

- *Relation (n:n) entre « **product** » et « **order** ».*
  Un produit peut être associé à zéro ou plusieurs commandes, et une commande peut regrouper un ou plusieurs produits.

- *Relation (n:n) entre « **product** » et « **ingredient** ».*
  Un produit peut être composé de un ou de plusieurs ingrédients. De même, un ingrédient peut être présent dans un ou plusieurs produits.

- Relation (n:n) entre « **restaurant** » et « **ingredient** ».
  Un restaurant aura en sa disposition plusieurs ingrédients pour préparer ses pizzas. Un ingrédient peut se retrouver dans plusieurs restaurants.

- *Relation (1:n) entre « **employee** » et « **restaurant** ».*
  Un employé ne peut travailler que dans un seul restaurant, mais un restaurant emploiera en général plusieurs salariés.

- *Relation (n:1) entre « **restaurant** » et « **order** ».*
  Un restaurant peut cumuler plusieurs commandes, tandis qu'une commande aura un seul restaurant associé à elle.

- *Relation (1:n) entre « **order** » et « **customer** ».*
  Une commande aura un seul client associé à elle. Les clients peuvent avoir plusieurs commandes.

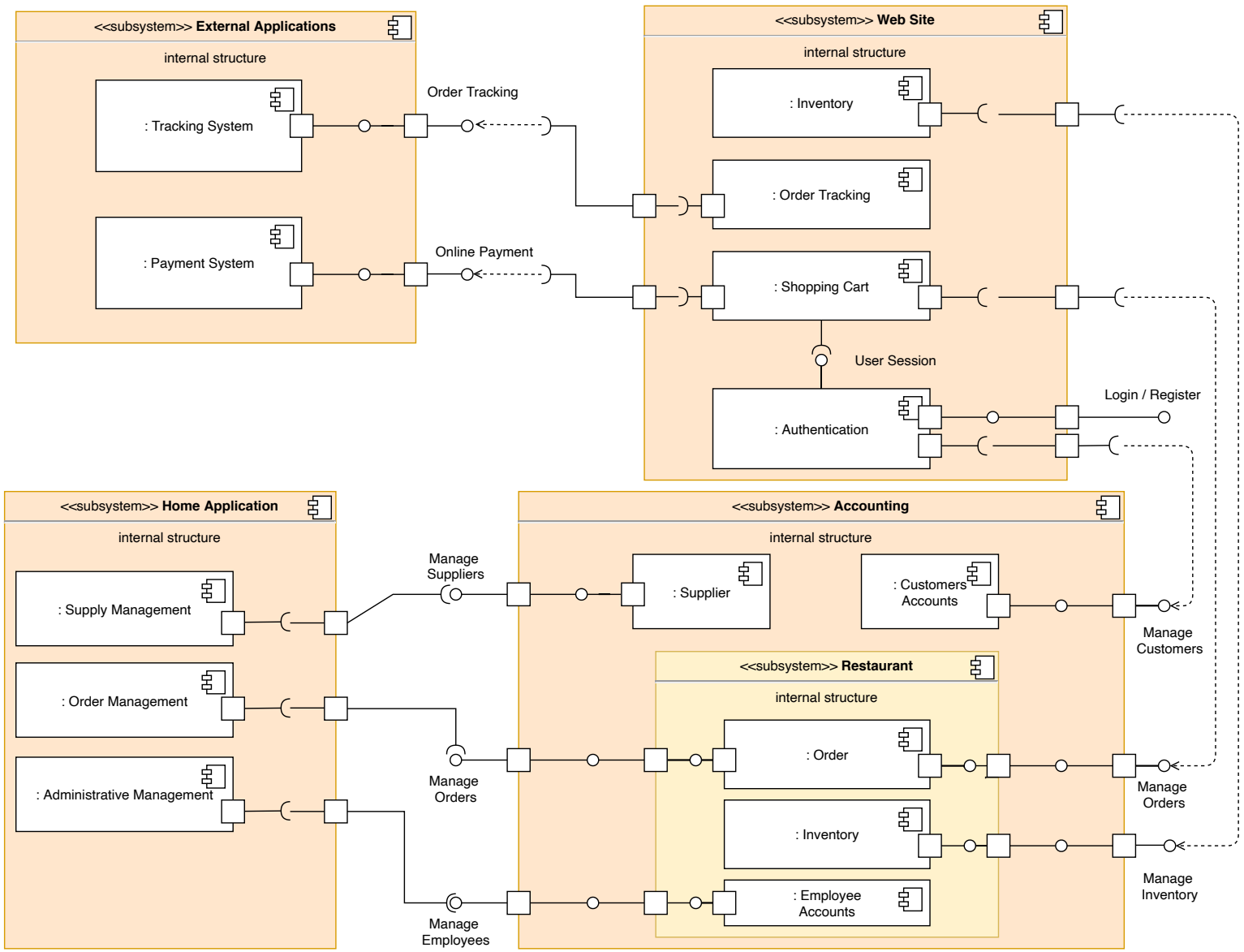
- *Relation (n:n) entre « **customer** » et « **delivery** ».*
  Un client peut avoir plusieurs adresses de livraisons. De même une adresse de livraison peut être associée à plusieurs clients en même temps.

- *Relation (0,1:n) entre « **order** » et « **delivery** ».*
  Une commande aura zéro (si le client récupère sa commande sur place) ou une adresse de livraison. Une adresse de livraison peut être associée à plusieurs commandes.

V. ARCHITECTURE TECHNIQUE

A. Diagramme de composants

Diagramme UML de Composants



Un diagramme de composants a pour objectif d'illustrer les relations entre les différents composants d'un système. Les composants font référence aux modules de classes, pouvant s'interfacer avec les autres composants du système.

Les « subsystem » représentent les grandes familles de composants.

Les composants :

1. Web Site

Le composant « Web Site » fournit aux différents utilisateurs (employés et clients), l'interface permettant de se connecter ou de créer un compte pour accéder à l'application web.

Pour afficher au client l'état d'une commande ou proposer les produits pour lesquels les stocks sont suffisants dans les restaurants, ce composant va avoir besoin des interfaces de gestion de commandes et gestion de l'inventaire.

Afin de permettre au client d'effectuer le règlement de sa commande en ligne, ainsi que de géolocaliser celle-ci, les interfaces de gestion de paiement en ligne et de gestion de géolocalisation seront nécessaires.

2. External Applications

Les applications externes fournissent les interfaces permettant la géolocalisation des commandes et la gestion des paiements en ligne au composant « web site ».

3. Accounting

Le composant « comptabilité » procure les informations sur l'état des commandes, de l'inventaire pour le composant « web site », et valide l'existence d'un utilisateur pour lui permettre d'ouvrir une session utilisateur.

Ce composant fournit aussi les interfaces employés, fournisseurs et commandes à l'application interne du groupe.

4. Home Application

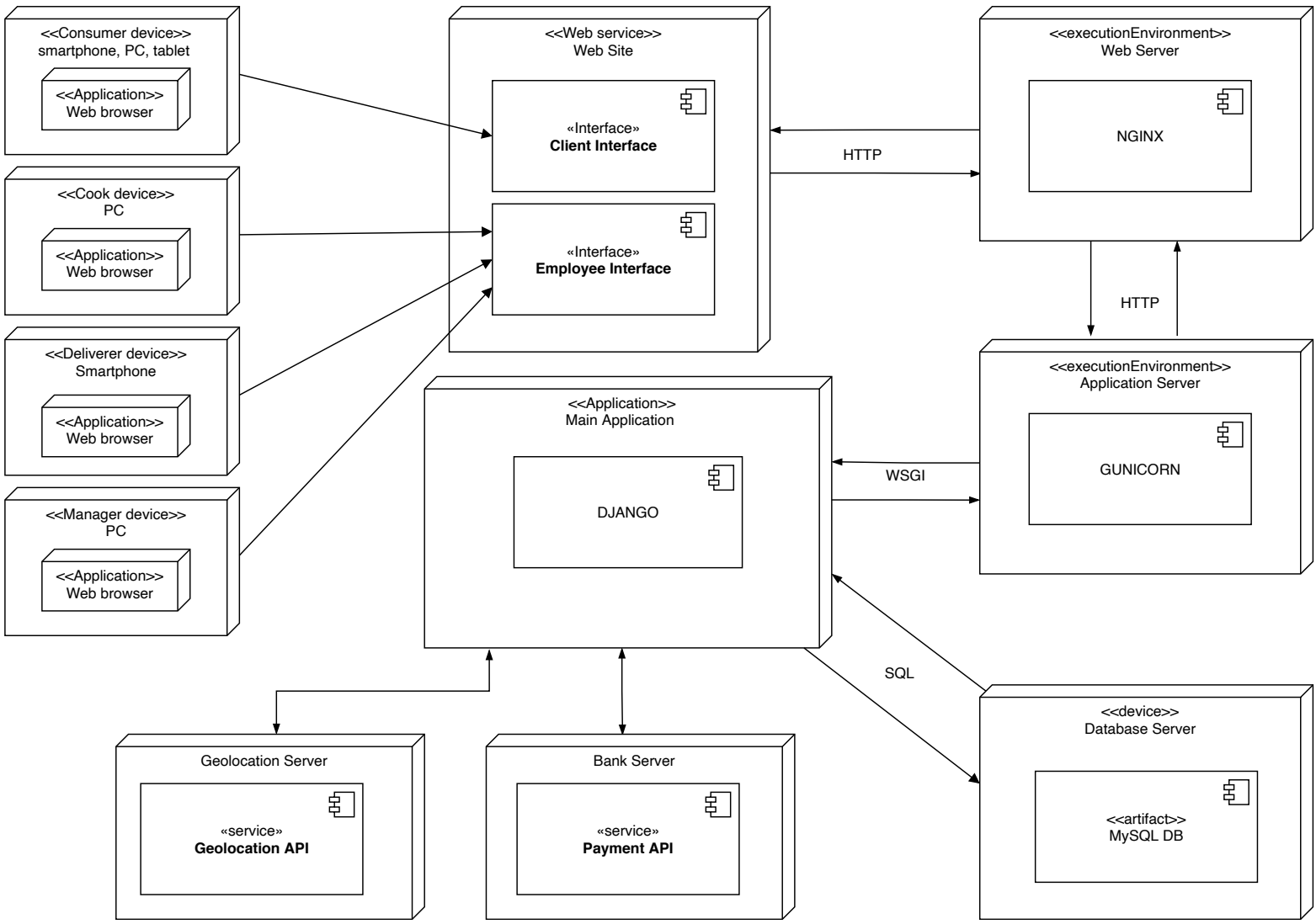
Le composant « Home Application » représente l'application interne pour les salariés, qui leurs permettra de changer le statut d'une commande, de gérer les fournisseurs, et aux responsables de gérer le personnel. Ce composant requiert les informations employés, commandes et fournisseurs.

VI. ARCHITECTURE DE DÉPLOIEMENT

La pile de logiciels est la suivante :

- Application : **Python 3.9.1** (https://www.python.org/downloads/)
- Framework : **Django 3.1.6** (https://www.djangoproject.com/download/)
- Serveur d'application : **Gunicorn 20.0.4** (https://gunicorn.org)
- Serveur web : **NGINX 1.18.0** (http://nginx.org/en/download.html)
- Serveur base de données : **MySQL 8.0** (https://dev.mysql.com/downloads/mysql/)

Diagramme UML de déploiement



Le diagramme de déploiement décrit la disposition physique des ressources matérielles qui composent le système. Une ressource est représenté par un noeud, le diagramme nous aide à visualiser une vue statique de ses ressources ainsi que les connexions qui existent entre elles.

A. Application

L'application sera développée à l'aide de Python 3, et de son framework web le plus populaire **Django**.

B. Serveur de base de données

Le choix du serveur de la base de données s'est porté sur MySQL, qui est un système de gestion de base de données relationnelle très populaire et reconnu pour sa fiabilité. MySQL utilise le langage SQL pour les opérations CRUD sur ses bases de données.

C. Serveur d'application Gunicorn

Gunicorn (Green Unicorn), est un serveur d'application HTTP python, qui communiquera avec notre application à l'aide de spécifications WSGI. WSGI ou Web Server Gateway Interface est une spécification qui définira une interface entre le serveur web et notre application web.

D. Serveur web NGINX

Le serveur web NGINX traite les requêtes HTTP envoyées par les clients. Associé à Gunicorn les deux serveurs permettent un **traitement plus rapide des demandes clients**, en raison de leurs complémentarité. NGINX se chargera de traiter les fichiers statiques et laissera Gunicorn s'occuper des fichiers dynamiques.

E. Serveur de Géolocalisation

Afin de localiser les livreurs, on utilisera une API de géolocalisation. **Ipstack** est l'API recommandée en raison de sa popularité et de la confiance que lui accordent des entreprises comme HubSpot par exemple.

F. Serveur de Banque

Pour des règlements sécurisés en ligne, l'API de PayPal est le choix recommandé, en raison d'un taux de confiance élevé des utilisateurs.