Nouveau système informatique pour l'ensemble des pizzerias du groupe

Spécifications techniques

Version	Date	Sections	Description
0.1	16/09/2020	1, 2 & 3	Création du document et ajout des sections 1, 2
0.2	17/09/2020	4 & 5	Ajout des sections 3, 4 et 5
0.3	18/09/2020	6 & 7	Ajout de la section 6 et 7
1.0	21/09/2020	Toutes	Relecture globale et ajustement de l'ensemble des sections du document

1 / Objectif du document

L’objectif du présent document intitulé “spécifications techniques” est de définir :

- le domaine fonctionnel
- le diagramme de composants
- les relations entre les différents composants internes et externes du système
- le diagramme de classes correspondant à chaque domaine
- le diagramme de classes dans sa globalité
- le modèle physique de données
- comment le système va être déployé (diagramme de déploiement)

Ce document doit donc décrire des solutions techniques du système informatique mis en place par IT Consulting & Development afin de répondre aux besoins exprimés par le groupe OC Pizza.

Les éléments du présent dossier découlent :
- du recueil des besoins client transmis lors du p6
- des spécifications fonctionnelles établies lors du p6

Ce document devra être mis à jour régulièrement pour tenir compte des évolutions fonctionnelles et techniques du système.

2 / Le domaine fonctionnel

Description de son utilité

Nous allons identifier les éléments et les informations que l'on souhaite enregistrer dans notre base de données pour que cela forme un système cohérent en définissant un domaine fonctionnel global. On utilisera dans ce document de spécifications techniques une base de données relationnelle afin de distinguer au mieux les relations entre les différentes parties qui composent notre domaine fonctionnel.
On utilisera donc une approche orientée objet pour représenter les composants de notre domaine fonctionnel. Le diagramme de classes servira de base à la modélisation du Modèle Physique de Données.

Pourquoi utiliser le diagramme de classes ?
Le diagramme de classes fait partie des diagrammes qui respectent la norme de modélisation graphique UML (Unified Modeling Language).
Chaque objet de notre domaine fonctionnel peut être soit réel (un client, un produit, etc.) ou abstrait (une commande, un stock, etc.).

Le domaine fonctionnel du système OC Pizza décrit l'ensemble des classes qui serviront de support à la création du MPD. Celui-ci établira le schéma des données relationnelles qui sera exploité par le nouveau système. Le domaine fonctionnel sera découpé en trois packages avant d'être présenté dans son intégralité. Les packages sont ceux présentés dans les spécifications fonctionnelles, à savoir :

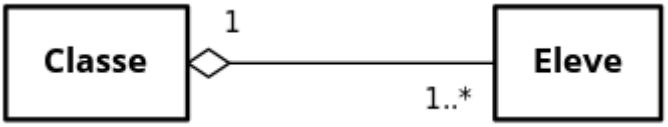
- le domaine de commande
- le domaine d'authentification
- le domaine d'administration

Pour mieux comprendre les diagrammes de classes voici quelques notions importantes à prendre en compte :

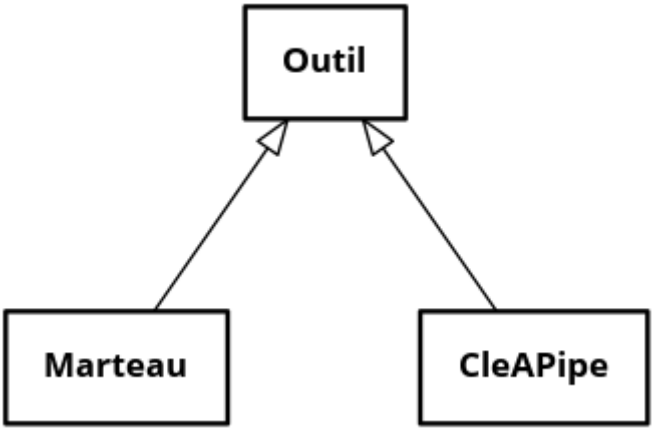
- Pour le nom des classes il est nécessaire qu'il soit :
    - au singulier
    - avec des lettres non accentuées
    - commençant par une majuscule
    - utilisant Pascal Case (chaque mot commence avec une majuscule)

  Ce qui donnerait par exemple ceci : PierrePrecieuse
- Pour les attributs :
    - au singulier
    - avec des lettres non accentuées
    - commençant par une minuscule
    - utilisant Camel Case (chaque mot commence avec une majuscule sauf le premier)

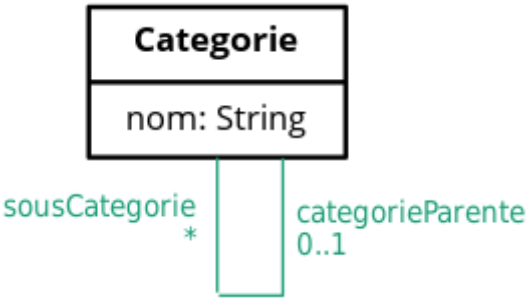
  Ce qui donnerait par exemple ceci : couleurYeux
- les classes sont composées de 3 parties correspondant, de haut en bas : au nom de la classe, aux attributs et aux opérations (méthodes appelables sur le classe). Les opérations sont importantes en AOO pour la conception de l'application)
- les méthodes présentées dans chaque classe sont indicatives et ne sont donc pas exhaustives
- le + devant les attributs signifie qu'il s'agit d'un attribut public
- il existe trois catégories de relations :
    - un à un (one-to-one)
    - un à plusieurs (one-to-many) ou plusieurs à un (many-to-one)
    - plusieurs à plusieurs (many-to-many)
- Quand une classe a un rôle qui correspond à un ensemble ou un regroupement d'objets, il peut être intéressant de mettre en valeur cet aspect. En effet, cela permet de voir au premier coup d'œil qu'il s'agit d'un ensemble, sans avoir à se pencher sur les multiplicités de l'association (qui sont bien évidemment dans ce cas de type un à plusieurs ou plusieurs à plusieurs). Cet aspect d'ensemble est modélisé par une association appelée agrégation et est matérialisée par un losange du côté de la classe jouant le rôle d'ensemble. Par exemple dans l'association Classe/Élève, la classe joue un rôle de regroupement d'élèves.



- l'héritage entre deux classes est représentée avec une flèche non pleine (un marteau et une clé à pipe sont des outils et se distinguent par leur fonctionnalité principale. À savoir pour un marteau le fait de taper avec et pour une clé à pipe de serrer ou desserrer un écrou) :

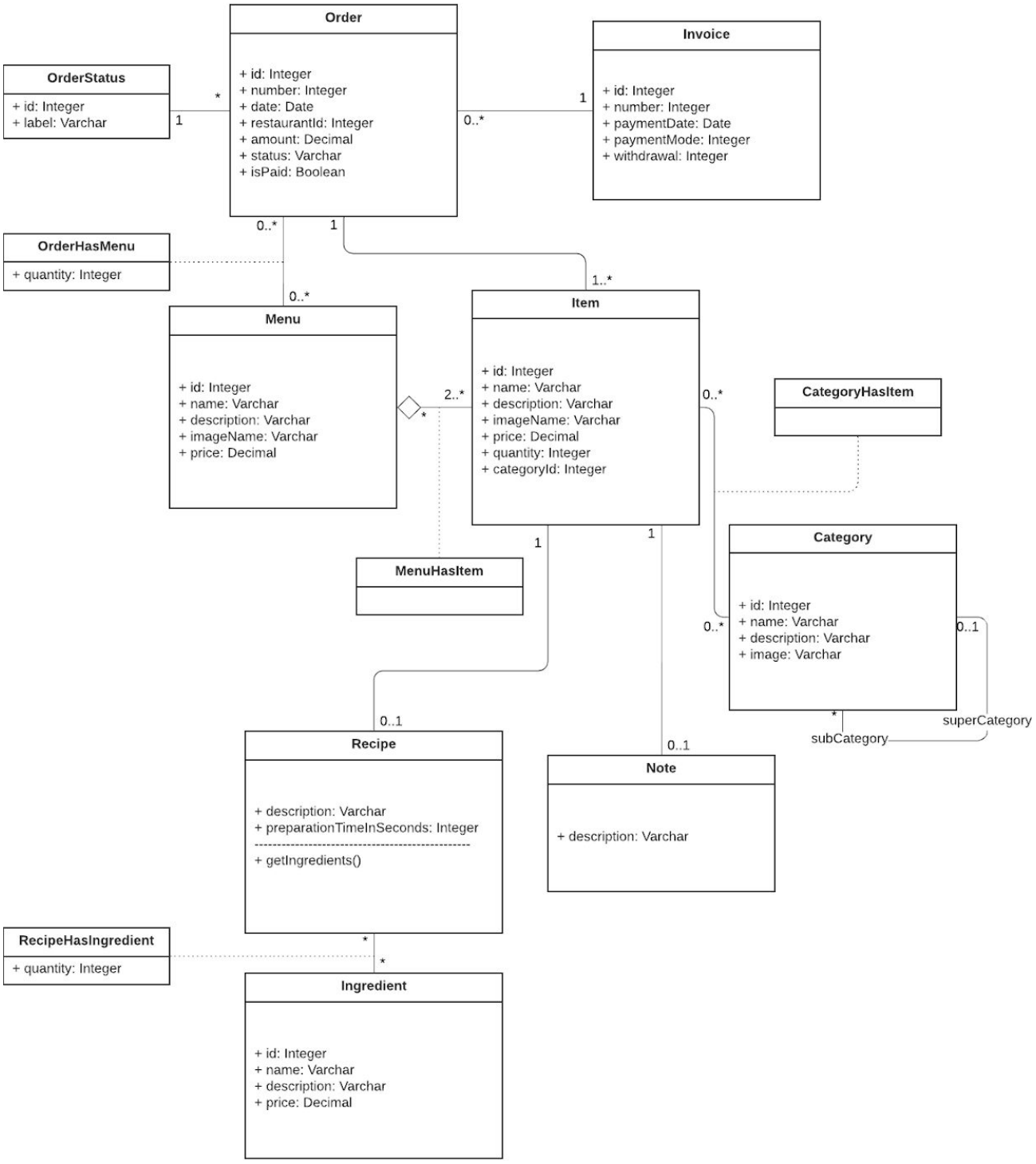


- Une association est dite réflexive quand les deux extrémités de celle-ci pointent sur la même classe. Voici un exemple d'association réflexive pour modéliser le fait qu'une catégorie peut avoir des sous-catégories :



Le but de ces règles est de réduire les possibles erreurs de typographie et d'être compatible avec leur mise en œuvre tant dans le code de l'application que dans la base de données.

Domaine de commande

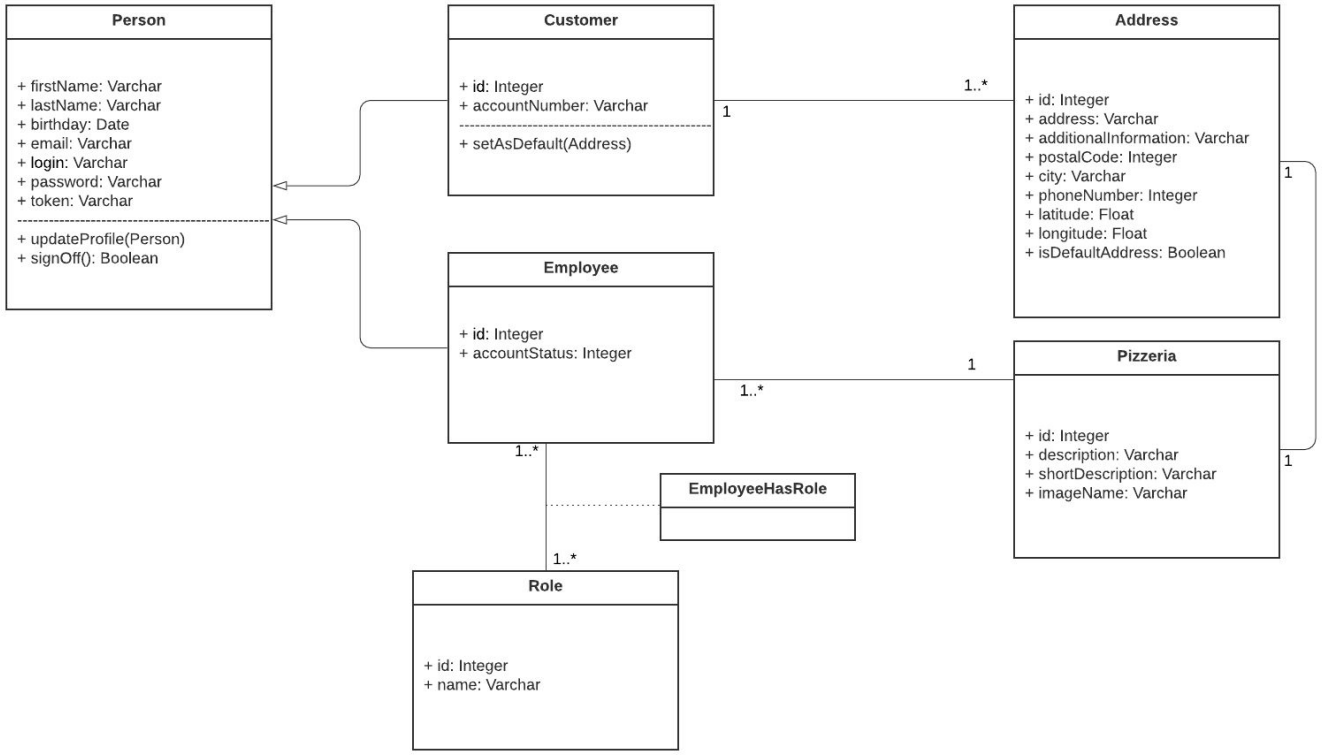


Description du diagramme de classes d'une commande :

Dans ce diagramme une facture "Invoice" est attachée à une commande.
Une commande peut ne pas contenir de menu "Menu" mais devra obligatoirement contenir au moins un produit "Item". Une commande peut contenir également, en plus de contenir un produit, un ou plusieurs menus.
Un élément de menu est défini par aucune ou une recette "Recipe" qui comprend la description et le temps de la préparation en secondes. Une recette est composée de plusieurs ingrédients et il existe plusieurs recettes. Chaque ingrédient possède un nombre le représentant sa quantité en stock dans la classe "Stock" associée.
La classe catégorie "Category" est établie pour faciliter le regroupement et classification des produits proposés aux clients. Sur ce diagramme on peut donc apercevoir une association dont les deux extrémités pointent sur la même classe intitulée "Category". Il s'agit d'une association réflexive. Dans notre cas, elle sert à indiquer qu'une catégorie peut avoir une catégorie parente (et dans l'autre sens, des sous-catégories).
On peut ainsi modéliser une arborescence de catégories comme celle-ci par exemple :

- Boissons
    - Jus de fruit
    - Limonades
    - Sodas

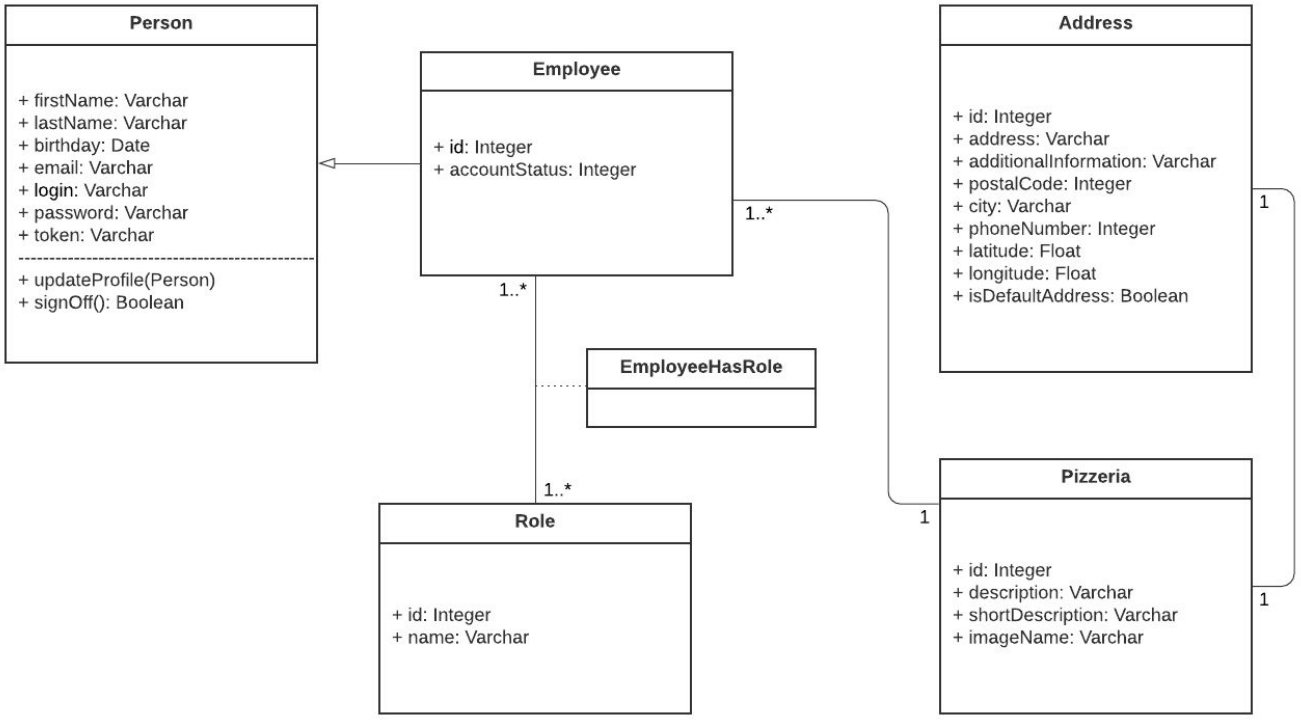
Domaine d’authentification



Description du diagramme de classes d’authentification :

Une personne se spécialise en client ou employé en fonction de son compte. Les classes “Customer” et “Employee” héritent donc toutes deux de la classe mère “Person”.
Qu’on soit connecté en temps que client ou employé on peut déclencher le processus de commande. Dans le cas d’un visiteur (non connecté), il faudra impérativement une connection au compte afin de finaliser la commande.
Une personne peut stocker un nombre illimité d’adresses et il devra en définir une par défaut s’il en possède plus qu’une.
Enfin, il existe une classe “Pizzeria” et une “Role” qui sont reliées respectivement à la classe “Employee” afin que ce dernier puisse avoir une adresse d’exercice et une fonction (caissier, livreur, pizzaïolo, etc.).

Domaine d’administration

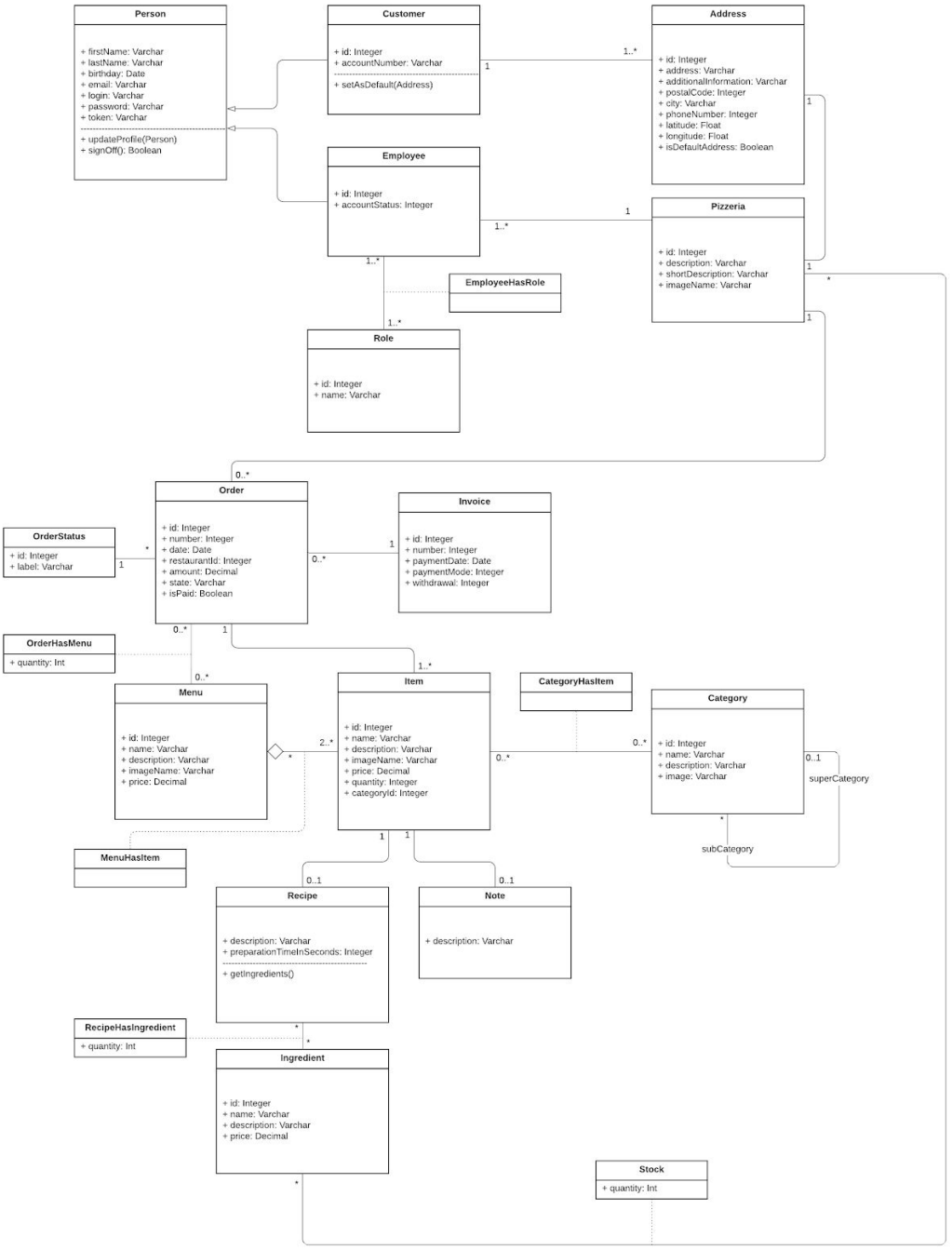


Description du diagramme de classes d’administration :

La classe employé “Employee” hérite de la classe mère “Person” et les sous classes “Manager”, “Cashier”, “Pizzaiolo” et “Deliverer” héritent de ‘Employee”.
La spécialisation de chaque employé (manager, caissier, pizzaïolo, livreur) est déterminée par la relation “Role” dans le diagramme de classes d’authentification.
Chaque employé peut être affecté à une seule pizzeria.
Enfin, chaque pizzeria est rattachée à son lot de commandes effectuées par ses clients par les liaisons entre les classes pizzeria, invoice, order et person. (Cette liaison n’est visible que sur le schéma principal).

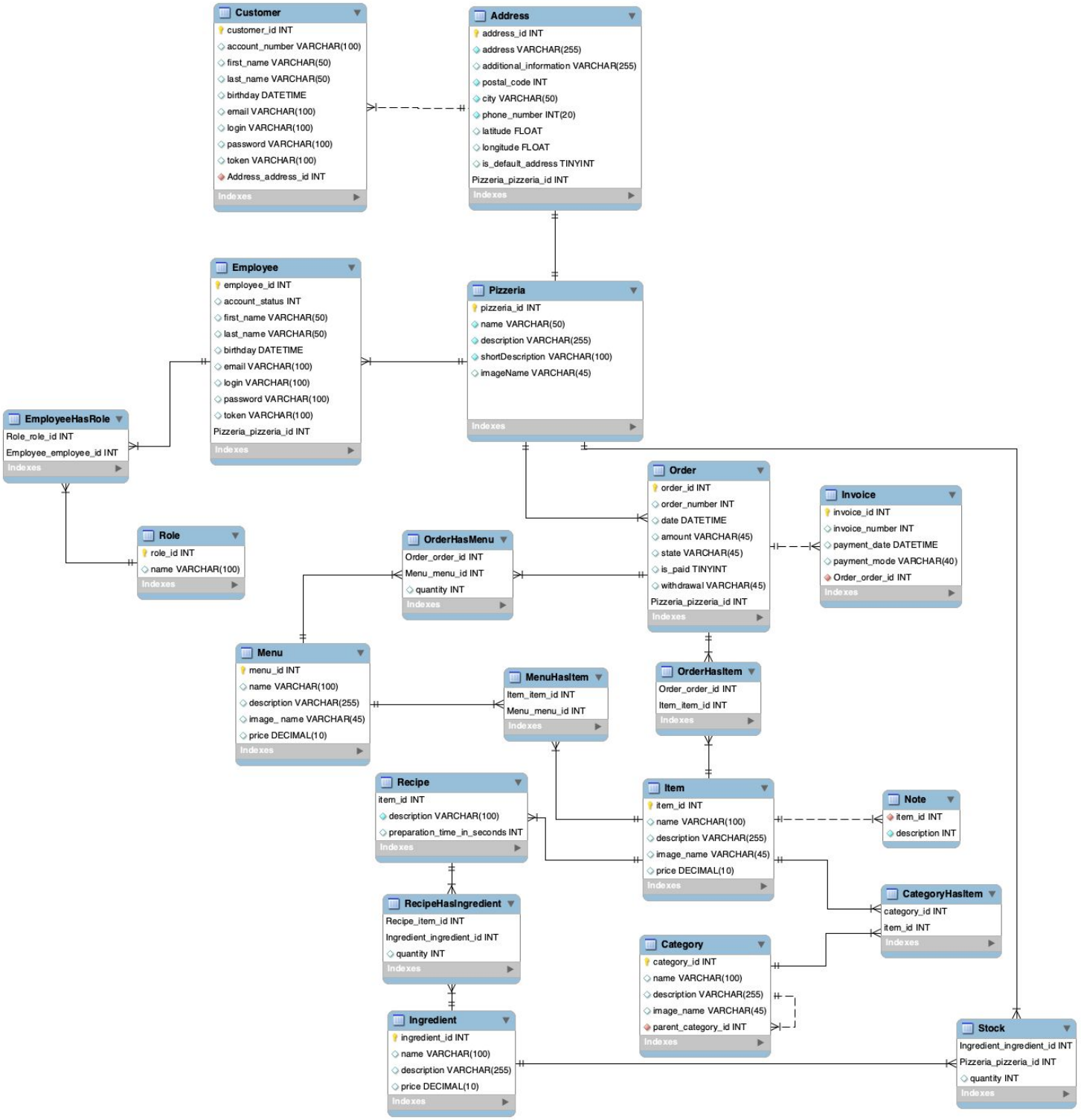
3 / Le diagramme de classes du domaine fonctionnel

Voici donc une représentation globale du diagramme de classes du domaine fonctionnel du groupe OC Pizza regroupant les trois domaines précédemment respectifs “commande”, “authentification” et “administration” :



4 / Le modèle physique de données

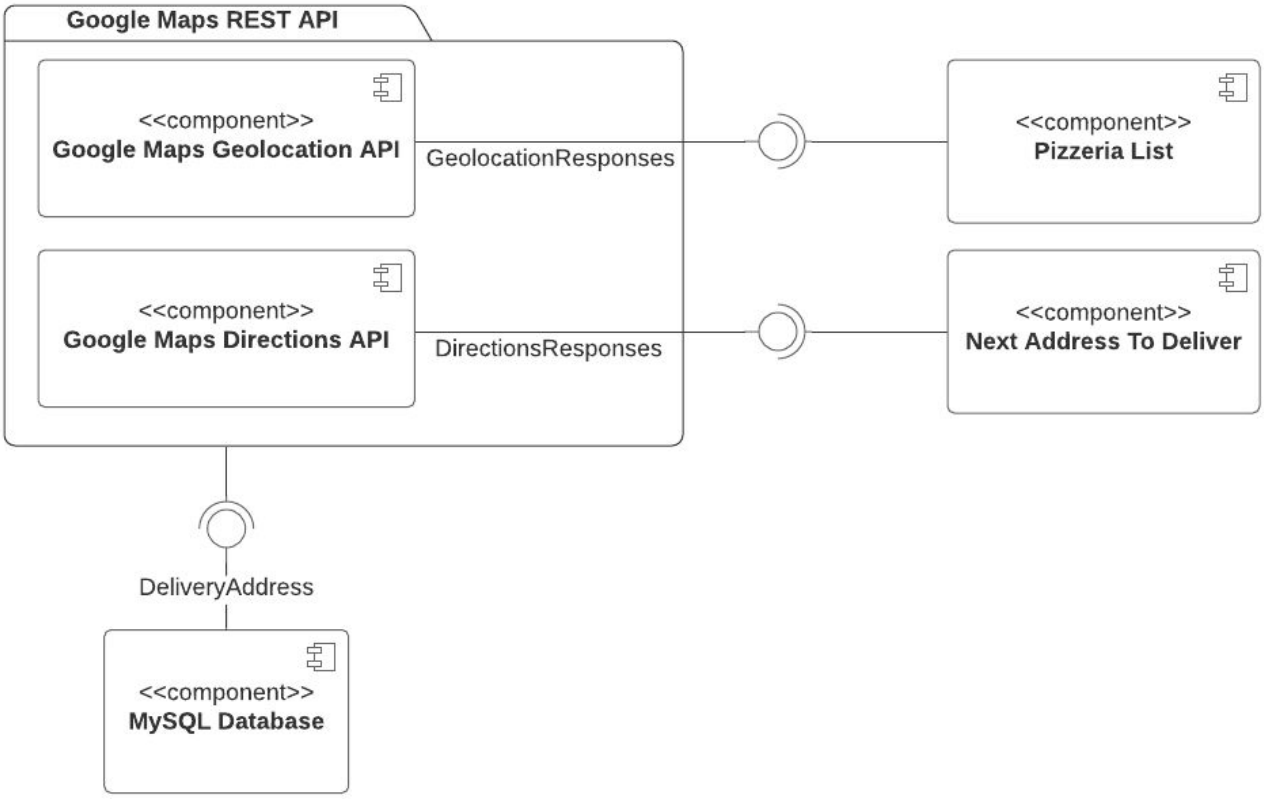
Dans la méthode Merise, le Modèle Physique de Données (MPD) consiste à modéliser dans le détail la base de données relationnelle OC Pizza. Nous décrirons les tables et les liens entre-elles, les types de données des différentes colonnes de chaque table et les clés primaires et étrangères. Le langage SQL est utilisé pour ce type d'opération.



5 / Les diagrammes de composants

L’organisation des éléments logiciels du système du groupe OC Pizza met en évidence les dépendances entre les composants.
Le diagramme de composants décrit l'organisation du système du point de vue des éléments logiciels comme les modules (paquetages, fichiers sources, bibliothèques, exécutables), des données (fichiers, bases de données) ou encore des éléments de configuration (paramètres, scripts, fichiers de commandes). Ce diagramme permet de mettre en évidence les dépendances entre les composants (qui utilise quoi ?).

Diagramme de composants de l’API Google Maps

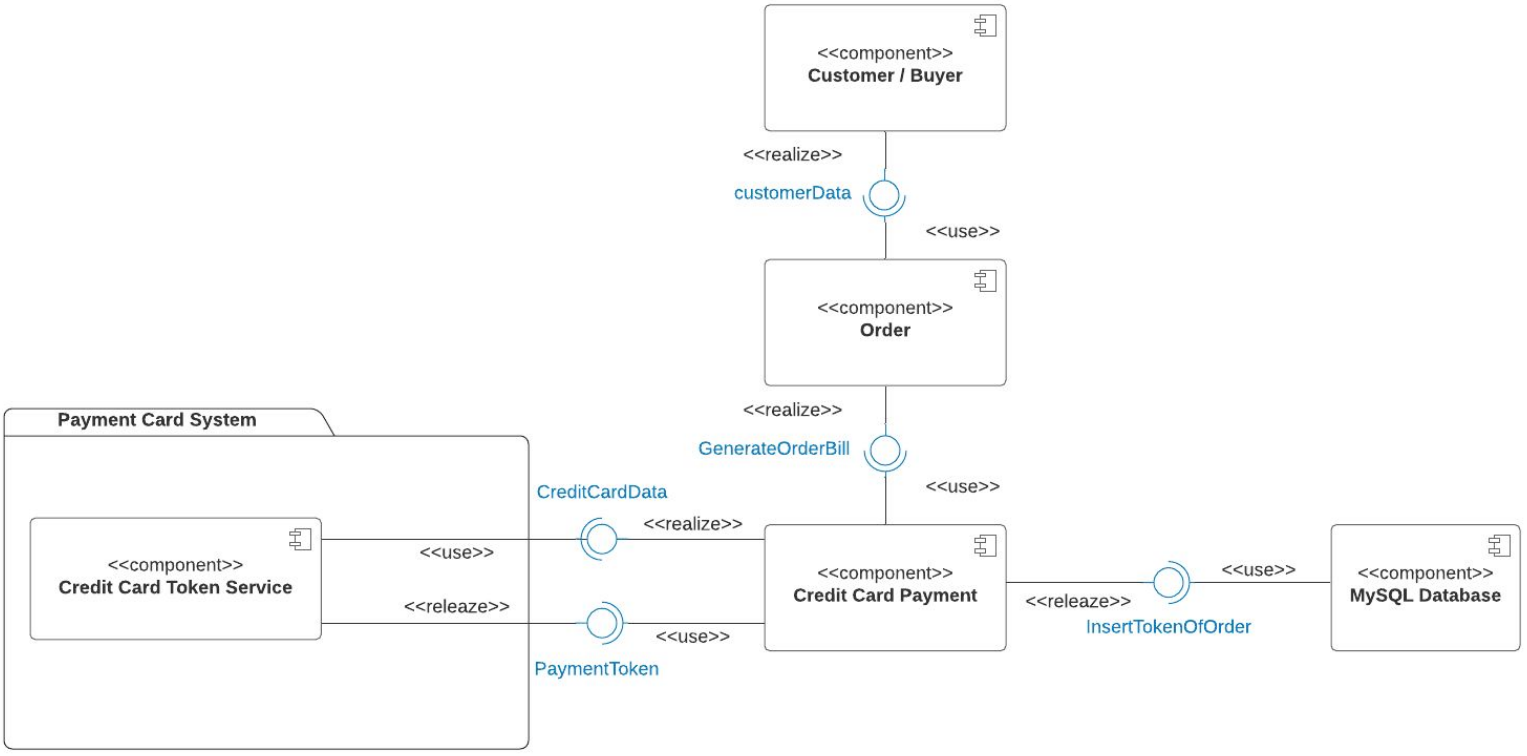


Description du diagramme de composants de l'API Google Maps :

Le composant "**MySQL Database"**, par son interface offerte envoie aux APIs REST Google Maps Directions et Geolocation les informations sur l'adresse de livraison d'un client et l'adresse des différentes pizzerias du groupe OC Pizza. Ces composants externes calculent le temps de parcours et les itinéraires à privilégier, pour un véhicule, entre cette adresse et celles des pizzerias et envoie l'information à l'interface requise du composant **"Next Address To Deliver"** qui va retenir le prochain client à livrer. Cette information sera stockée dans la base de données dans la table "Address" pour l'adresse de livraison donnée.
L'envoi d'information est effectué également au composant **"Pizzeria List"** qui va retenir les coordonnées physique de chaque pizzeria.

Le composant Google Maps Directions API fournit aussi au composant **"Next Address To Deliver**" le temps prévu pour effectuer le trajet. Celui-ci s'additionne au temps d'attente puis au temps prévu de fabrication de la commande en cours et fournira une heure approximative prévue de livraison au client, avant qu'il ne valide sa commande.
D'autre part, le composant Google Maps Geolocation API fournit au composant **"Pizzeria List"** les coordonnées (latitude / longitude) de chacune des six pizzerias afin d'afficher les pizzerias sur une carte et montrer aux clients leur proximité physique avec chacune des pizzerias.

Diagramme de composants du système de paiement par carte bleue

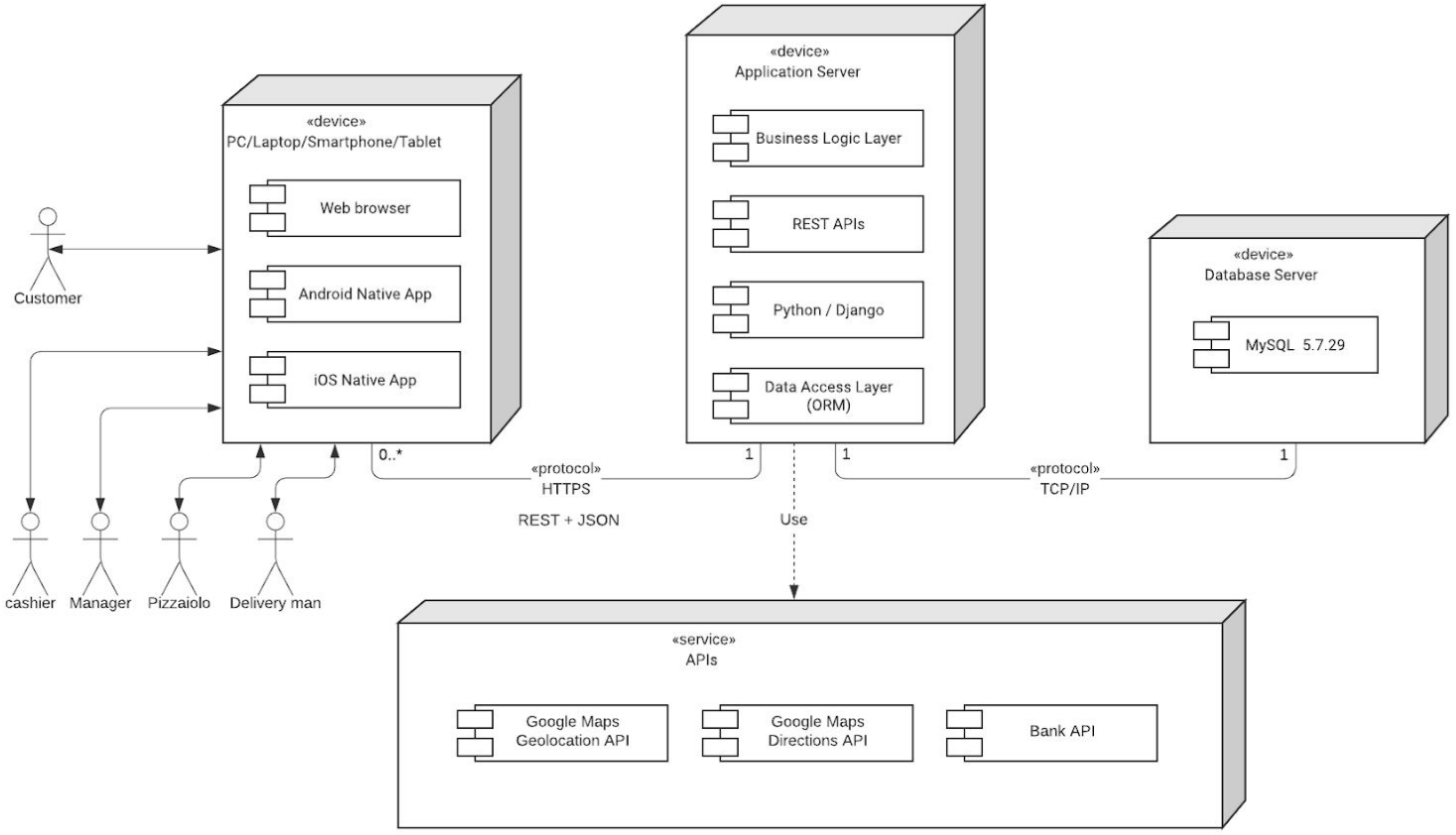


Description du diagramme de composants du système de paiement par carte bancaire :

Le composant **“Customer / Buyer”**, par son interface envoie au composant **“Order”** les informations sur la commande. Ainsi, ce dernier est en mesure générer la facture associée à la commande ainsi que le montant total de celle-ci. Ce montant est requis par le composant **“Credit Card Payment**” qui récupère également les information du client comme son nom et prénom par exemple.
Le composant **“Credit Card Payment**” va échanger avec le système bancaire via le composant **“Credit Card Token Service”** des informations sur le client et le montant à régler. Le client aura directement accès au service de carte bancaire de la banque pour rentrer ses coordonnées personnelles (numéro de carte, date d’expiration, cryptogramme visuel à l’arrière de la carte bancaire).
En retour, ce composant externe renvoie un token au composant interne avec l’information cryptée sur la réalisation du paiement.
C’est cette information que gardera en base de données le système d’OC Pizza via le composant **“MySQL Database”**.

6 / Diagramme de déploiement

En UML, un diagramme de déploiement est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations. Les éléments utilisés par un diagramme de déploiement sont principalement les nœuds, les composants, les associations et les artefacts. Les caractéristiques des ressources matérielles physiques et des supports de communication peuvent être précisées par stéréotype.



Acteurs et noeud des appareils

Le premier nœud **“PC/Laptop/Smartphone/Tablet”** représente les appareils utilisés par les acteurs de l’application OC Pizza afin de passer une commande.
Les différents acteurs sont :

- les clients qui se connectent au site web via leur ordinateur, smartphone ou tablette
- les employés des différentes pizzerias du groupe OC Pizza qui passent commande via le site web pour les clients de passage dans la pizzeria ou qui effectuent une commande par téléphone

Il y a aussi les employés d’OC Pizza occupant les postes de Pizzaïolo, Livreur (Delivery Man), Manager général et Manager de pizzeria (Manager), qui utilisent l’application mobile pour le premier, la tablette pour le deuxième ou l’ordinateur pour les derniers afin d’interagir avec le système OC Pizza.
Tous ces utilisateurs utilisent un navigateur web (Web browser) ou une application mobile via les requêtes HTTPS et se connectent au deuxième nœud nommé **“Application Server”**.

Serveur de base de données

Le chemin de communication TCP/IP permet à la couche d’application de communiquer avec le nœud serveur de base de données intitulé **“Database Server”** qui est situé sur un autre serveur distinct. Ce serveur héberge la base de données qui fonctionne sous la version MySQL 5.7.29.

Serveur web et couche d’application

Le site est lié à la au noeud **“Application Server”** qui, programmée en Python/Django, intégrant les logiques métiers (Business Logic Layer), et interagissant avec les services externes comme Google Maps Directions / Geolocation API et le système bancaire (Bank), permet de gérer l’ensemble du processus de commande.
Le dernier composant logiciel de ce nœud est l’ORM (Data Access Layer) qui permet l’interface avec la base de données. Un mapping objet-relationnel (en anglais object-relational mapping ou ORM) est un type de programme informatique qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de données orientée objet. Ce programme définit des correspondances entre les schémas de la base de données et les classes du programme applicatif. On pourrait le désigner par là, « comme une couche d'abstraction entre le monde objet et monde relationnel ». Du fait de sa fonction, on retrouve ce type de programme dans un grand nombre de frameworks sous la forme de composant ORM qui a été soit développé, soit intégré depuis une solution externe. L'utilisation de la programmation orientée objet avec une base de données relationnelle nécessite de convertir les données relationnelles en objets et vice-versa. Ceci conduit à programmer cette conversion pour chaque objet et donc à dupliquer énormément de code similaire. En fait, le code permettant de réaliser toutes les opérations de CRUD (création, lecture, mise à jour, suppression) se ressemble énormément

d'une table à l'autre et d'un objet à l'autre. Les frameworks de mapping objet-relationnel permettent d'éliminer la duplication de code dans les opérations CRUD.

Pourquoi le framework Django ?

Django est un framework Python de haut niveau, permettant un développement rapide de sites internet, sécurisés, et maintenables. Créé par des développeurs expérimentés, Django prend en charge la plupart des tracas du développement web, on peut donc se concentrer sur l'écriture de notre application sans avoir besoin de réinventer la roue. Il est gratuit, open source, possède une communauté active, une bonne documentation, et plusieurs options pour du support gratuit ou non.