Dossier de spécifications techniques

Auteur : Maylis Baschet
Version : 30/10/2019 (1ère version)

https://github.com/MayBaMay/OCP_P6

TABLE DES MATIÈRES

VERSIONS :

Auteur	Date	Description	Version
Maylis Baschet	30/10/2019	Première rédaction	1.0

1. Introduction

1.1 Objet du document

Le présent document a pour but de présenter le dossier de spécifications techniques de la solution proposée par *IT Consulting Development* à OC Pizza.

Ce document met en évidence :
- le modèle fonctionnel de la solution illustré par une représentation statique des éléments qui composent le système et leurs relations,
- le modèle physique de données ou modèle relationnel de la base de données, pierre angulaire de la solution qui illustrera plus précisément l'architecture de celle-ci,
- les différents composants internes et externes du système et les liens et interactions entre eux,
- l'architecture de déploiement décrivant les éléments matériels et logiciels nécessaires.

1.2 Contexte

Dans le cadre d'un développement économique conséquent, OC Pizza souhaite disposer d'un système performant de gestion.
L'entreprise ouvrant de plus en plus de points de vente (5 actuellement et très prochainement 8), elle a donc besoin d'une gestion centralisée et performante qui permet de diminuer les temps de réalisation de ses prestations et renseigner les responsables sur l'activité tant globale que spécifique à chaque point de vente.
Le site devra être agréable, utile et facile d'utilisation tant pour les clients que pour les membres du staff des différents points de vente.

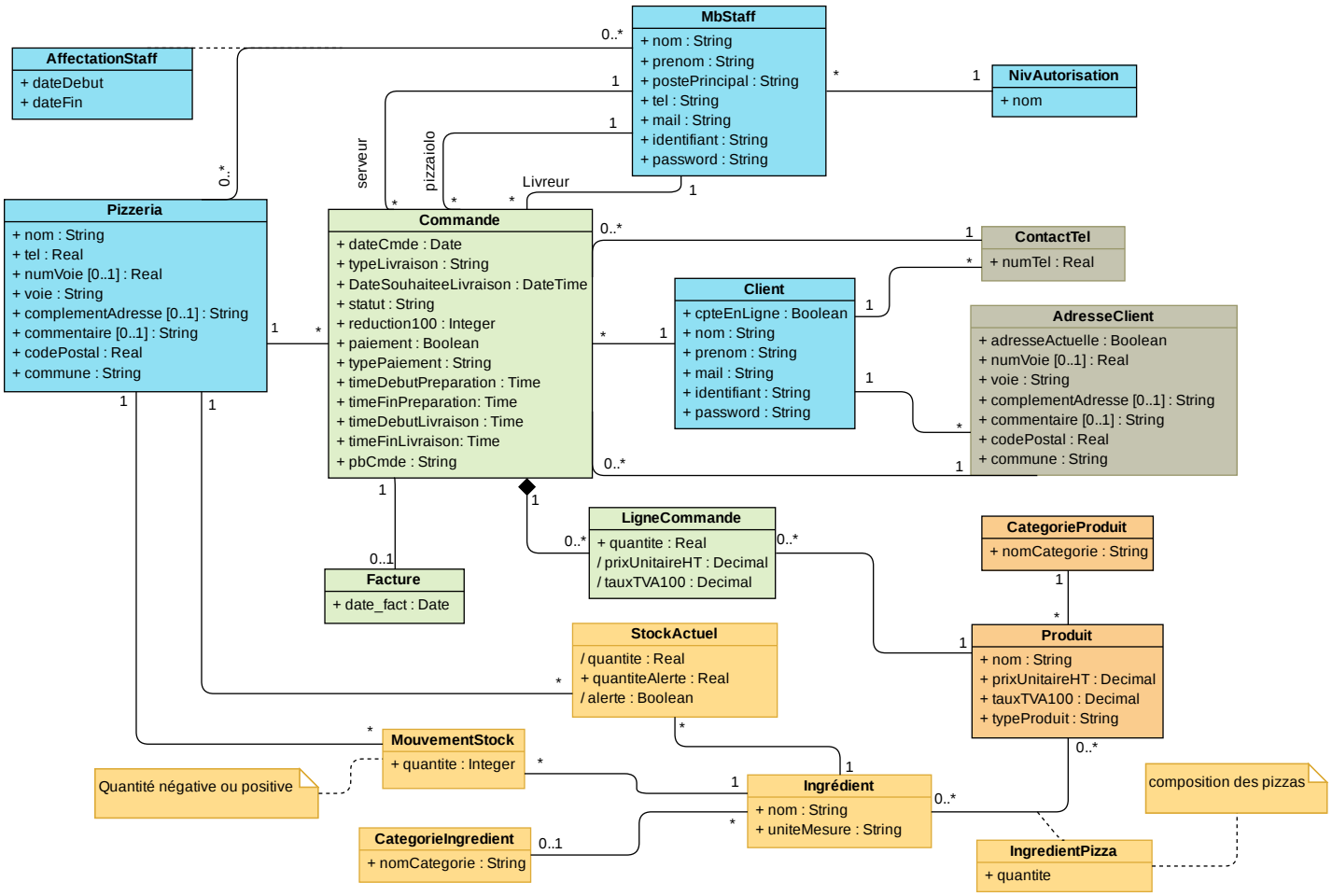
Besoins exprimés :
- être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;
- suivre en temps réel les commandes passées, en préparation et en livraison ;
- suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas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

1.3 Enjeux et objectifs

Ce document constitue la deuxième étape de cette collaboration suivant la proposition de spécifications fonctionnelles présentée le 16 juillet 2019 et aux échanges qui l'ont suivi. Il présente donc les différentes réponses techniques adaptées au mieux aux besoins exprimés. Il conviendra bien évidemment de les préciser encore davantage avec OCPizza afin de répondre pleinement à toutes les attentes. À cette fin, les différents diagrammes présentés dans le présent dossier ont été rédigés autant que possible en français.

2. Domaine fonctionnel

Le domaine fonctionnel, illustré par le diagramme de classe UML ci-dessous, représente l'organisation de l'information dans le système grâce aux différentes classes et aux liens existants entre-elles.
La conceptualisation de ce domaine fonctionnel servira de base à la création du Modèle Physique de données, socle de la modélisation de la base de données relationnelle MySQL ainsi qu'à la programmation à venir en Python.

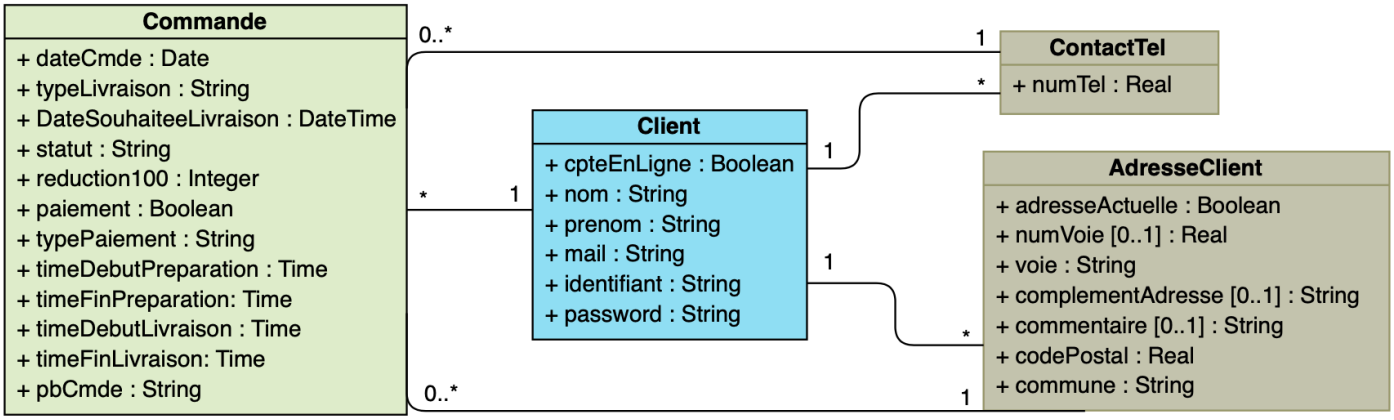


2.1 Diagramme de classe UML

Ce diagramme a pour place centrale la classe Commande qui directement ou indirectement permettra d'accéder à toutes les informations concernant l'activité réalisée par l'entreprise en temps réel. Cette classe est en effet en lien avec les informations des clients, des différentes boutiques, de l'équipe d'OCPizza et des produits vendus. Elle viendra également, par l'intermédiaire de la classe de ligne de commande et des classes concernant les produits, ingrédients et stocks, permettre le suivi en temps réel du stock de chaque pizzeria.

2.2 Descriptif des classes et de leurs relations

2.2.1 Le traitement des informations clients :

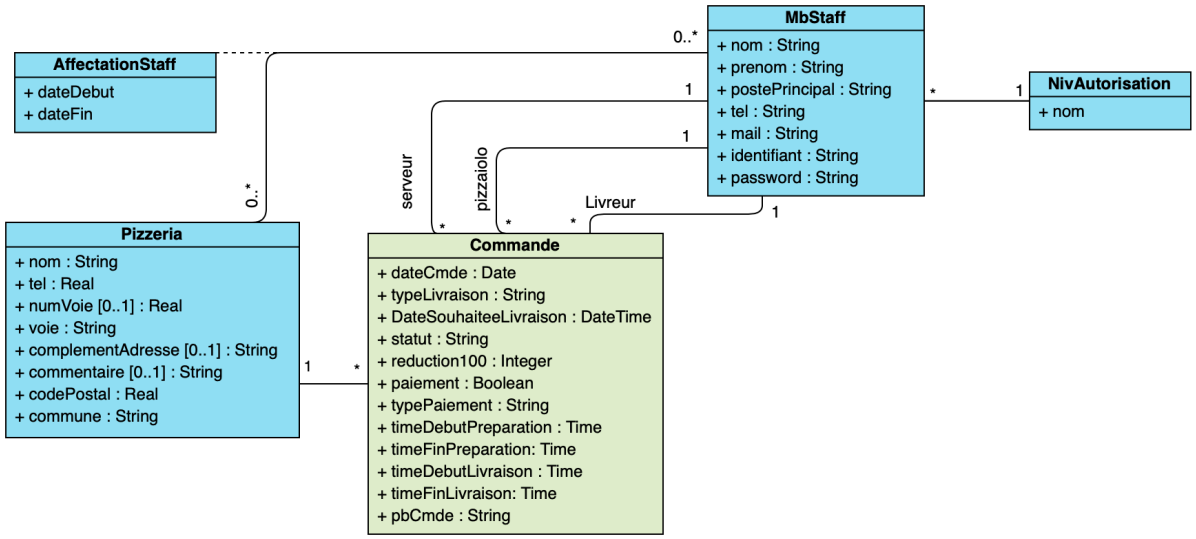


Les informations des clients sont centralisées dans une classe ***Client*** contenant les informations générales les concernant ainsi que les identifiant et mot de passe d'accès aux comptes clients du système. Elle comprends également un attribut de type booléen ***cpteEnLigne*** indiquant si un compte en ligne a déjà été créé par le client. Ainsi, les commandes et informations d'un même client seront basées sur un même identifiant que la commande soit passée en ligne par le client ou par un employé pour son compte. Le programme pourra également vérifier avant de créer un nouveau compte si un compte client a déjà été créé pour le client, évitant ainsi les doublons.

Les coordonnées des clients sont quand à elles contenues dans les classes ***AdresseClient*** et ***ContactTel***. Un même client peut avoir une ou plusieurs adresses passées, présentes ou à venir. Il en va de même pour les numéros de téléphone. Il s'agit donc d'associations ***one-to-many***. L'attribut ***adresseActuelle*** permettra d'accéder rapidement à la dernière adresse renseignée par le client.

La classe ***Client*** est liée avec la classe ***Commande*** par une association ***one-to-many.*** Un(e) client(e) peut en effet passer plusieurs commandes, mais une commande ne peut être passée que par un seul client.
La classe ***Commande*** permet également d'identifier l'adresse et le contact téléphonique du client spécifique à une commande particulière par une association ***one-to-many*** aux classes ***AdresseClient*** et ***ContactTel***. Dans le cas d'un changement d'adresse, une nouvelle instance de la classe ***AdresseClient*** doit être créée afin de garder en mémoire les adresses utilisées dans les commandes passées.
L'attribut ***typeLivraison*** renseignera quand à lui s'il s'agit d'une commande à emporter avec retrait en boutique ou à livrer au domicile du client.
Nous incluons à cette classe un attribut ***dateSouhaiteeLivraison*** renseignant le cas échéant une date et une heure spécifique indiquée par le client pour se faire livrer (la mi-temps d'un match par exemple). Par défaut, cet attribut sera renseigné par le système par une estimation et une moyenne des temps de préparation selon qu'il s'agisse d'une livraison en boutique ou d'une livraison à domicile.

2.2.2 Le traitement des informations des pizzerias et membres de l'équipe d OCPizza



Comme l'on peut s'y attendre, chaque pizzeria est renseignée dans une instance de la classe ***Pizzeria*** et chaque membre de l'équipe, dans une instance de la classe ***MbStaff***. Concernant cette dernière, seules les informations utiles à l'application ont été retenues. L'adresse d'un(e) salarié(e) n'est pas nécessaire dans une solution de gestion des commandes, seuls ses contacts téléphonique et électronique pourront éventuellement servir. Nous retrouvons également ici les identifiant et mot de passe de chaque membre de l'équipe à son compte personnel.

Une association de classe permet de créer la classe ***AffectationStaff*** qui permettra de savoir pour quelle(s) pizzeria(s) travaille ou a travaillé un(e) salarié(e). Nous partons en effet du principe qu'un(e) salarié(e) pourrait dans l'absolu être amené à travailler dans plusieurs pizzeria du groupe, pour par exemple remplacer un(e) salarié(e) malade au pied levé, parer à un surcroit d'activité dans une pizzeria, dans le cas où les livreurs assurerai des livraisons de différentes pizzeria en fonction de leur position géographique… Nous reviendrons plus en détail sur l'association qui les lie dans le chapitre suivant (cf 3.2.4)

Afin de répondre aux fonctionnalités attendues, la table ***Commande*** renseignera l'attribut ***statut*** de la commande qui sera l'un des statuts suivants : « En attente de préparation », « En cours de préparation », « En attente de livraison », « En cours de livraison » et « Livrée ».

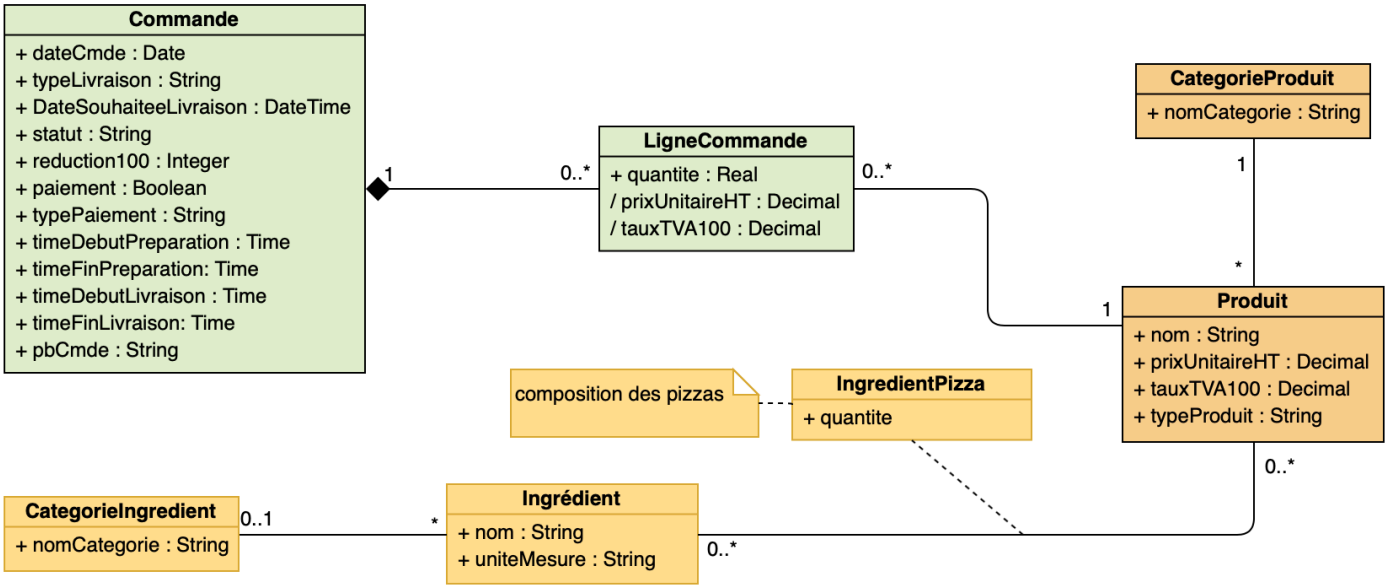
Une triple association à la classe ***MbStaff*** permettra de renseigner qui des salariés prendra en charge la prise de commande (indiqué comme ***« serveur »*** dans notre diagramme), étant renseigné NULL pour les commandes en ligne, qui prendra en charge la préparation (***« pizzaiolo »***) et qui prendra en charge la livraison.
L'attribut ***DateCmde*** sera renseigné automatiquement par le programme aux date et heure de prise en compte de la commande par le système soit à l'initiative d'un(e) employé(e) pour le compte d'un client soit par un client depuis le site web.

Un pizzaiolo renseignera la prise en charge de la commande en utilisant l'application avec son compte ce qui aura pour effet de renseigner l'attribut ***« timeDebutPreparation »*** qui servira à automatiser le passage au ***statut*** « En cours de préparation » qui met fin à la possibilité pour le(la) client(e) de modifier sa commande. De même lorsqu'un pizzaiolo renseignera avoir fini la préparation d'une commande, l'attribut ***« timeFinPreparation »*** sera renseigné et le statut pourra ainsi passer à « En attente de livraison » et être signalé comme tel aux différents livreurs.

Le même mécanisme s'enchaîne avec la livraison : un(e) livreur(euse) déclare prendre en charge la commande ce qui renseigne l'attribut ***« timeDebutLivraison »*** et lorsque la livraison aura été effectuée, le(la) livreur(euse) le signalera à l'application qui renseignera l'attribut « ***timeFinLivraison »*** et le ***statut*** de la commande passera ainsi en « Livrée ».

Enfin, entre la direction, qui peut accéder à toutes les fonctionnalités et données du système et le(la) livreur(euse) qui n'a vocation qu'à accéder aux fonctionnalités et données concernant les livraisons, les niveaux d'autorisation doivent être pris en compte et définis clairement. Nous devrons donc définir avec OCPizza les besoins de sécurisation à mettre en place dans notre programme en tenant compte des profils multi-postes afin d'éviter toute situation de blocage. La classe ***MbStaff*** sera ainsi associée à la classe ***NivAutorisation*** par une association ***one-to-many***.

2.2.3 Le traitement des informations des produits



La classe ***LigneCommande*** permet de renseigner le contenu des commandes avec une association de composition ***one-to-many*** à la classe ***Commande***. Outre les quantités de chaque produit de la ligne de la commande, seront automatiquement renseignés dans la classe ***LigneCommande,*** les prix et taux de TVA applicables au produit au moment du passage de la commande et qui sont contenu dans la classe ***Produit***. Nous pourrons ainsi garder une trace figée du prix et du taux de TVA sans craindre que ceux-ci soient modifiés au moment d'un changement de tarifs global.

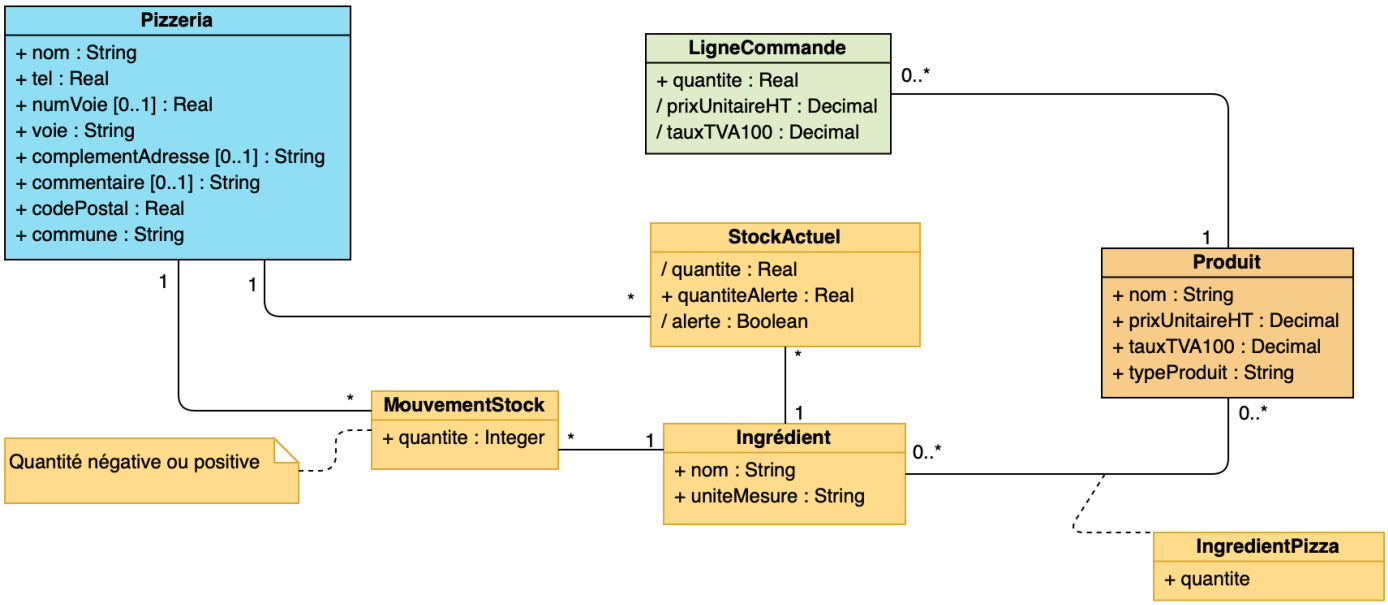
Les produits proposés par OCPizza sont de plusieurs types que nous retrouverons renseignés dans l'attribut ***« typeProduit »*** de la classe ***Produit*** notamment :

- les produits « Transformés » tels que les pizzas, préparés par les pizzaiolo à partir d'ingrédients
- Les produits de « Revente » tels que les boissons ne nécessitant aucun ingrédient à leur préparation.
- Les « suppléments » qui ne sont autres que des ingrédients ajoutés à un produit à la demande d'un client.

La classe d'association ***IngredientPizza*** fait le lien entre les classes ***Produit*** et ***Ingrédient*** nécessaires à la confection des produits transformés et des suppléments. Il s'agit en effet des recettes que nous utiliserons pour l'aide-mémoire des pizzaiolo ainsi que pour ajuster le stock d'ingrédients découlant de chaque commande comme nous le verrons plus en détail ci-dessous.

Une classe ***CategorieProduit*** est prévue afin de faciliter les recherches et l'affichage des produits dans notre application. Une association ***one-to-many*** devrait suffire, un produit n'ayant a priori pas vocation à appartenir à plusieurs catégories. De même, une classe ***CategorieIngredient*** est prévue.

2.2.4 Le traitement des informations des stocks d'ingrédients :



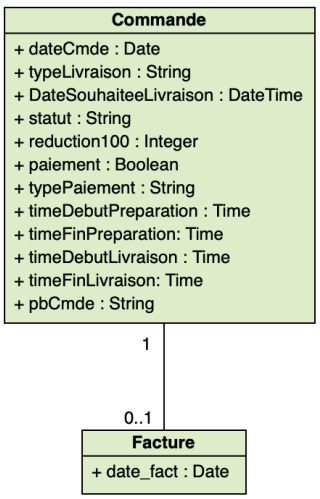
La gestion des stocks d'ingrédients et donc des produits qui en proviennent se fera grâce à la classe ***MouvementStock*** qui renseignera une instance par modification d'un stock d'ingrédient.

Celles-ci peuvent provenir d'une mise à jour manuelle effectuée par la direction ou un(e) salarié(e) suite à l'achat de nouveaux ingrédients ou à un ajustement du stock existant, ou par une mise à jour automatique qui sera déclenchée pour chaque ligne de commande dont le ***statut*** sera passé en « En cours de préparation ». Le système viendra prendre les informations de quantité du produit et la référence du produit dans la classe ***LigneCommande***, les ingrédients et la quantité nécessaire pour réaliser le produit concerné dans la classe ***IngredientPizza*** et viendra répercuter toutes ces informations dans la classe ***MouvementStock***.

Pourront ainsi être renseignés par addition (quantités positives ou négatives) du programme à venir les stocks actuels de chaque ingrédient dans une classe ***StockActuel.***

Les deux classes relatives aux stocks d'ingrédients seront bien évidemment associés à la classe ***Pizzeria*** afin d'identifier leurs stocks respectifs.

2.2.5 De la commande à la facture :



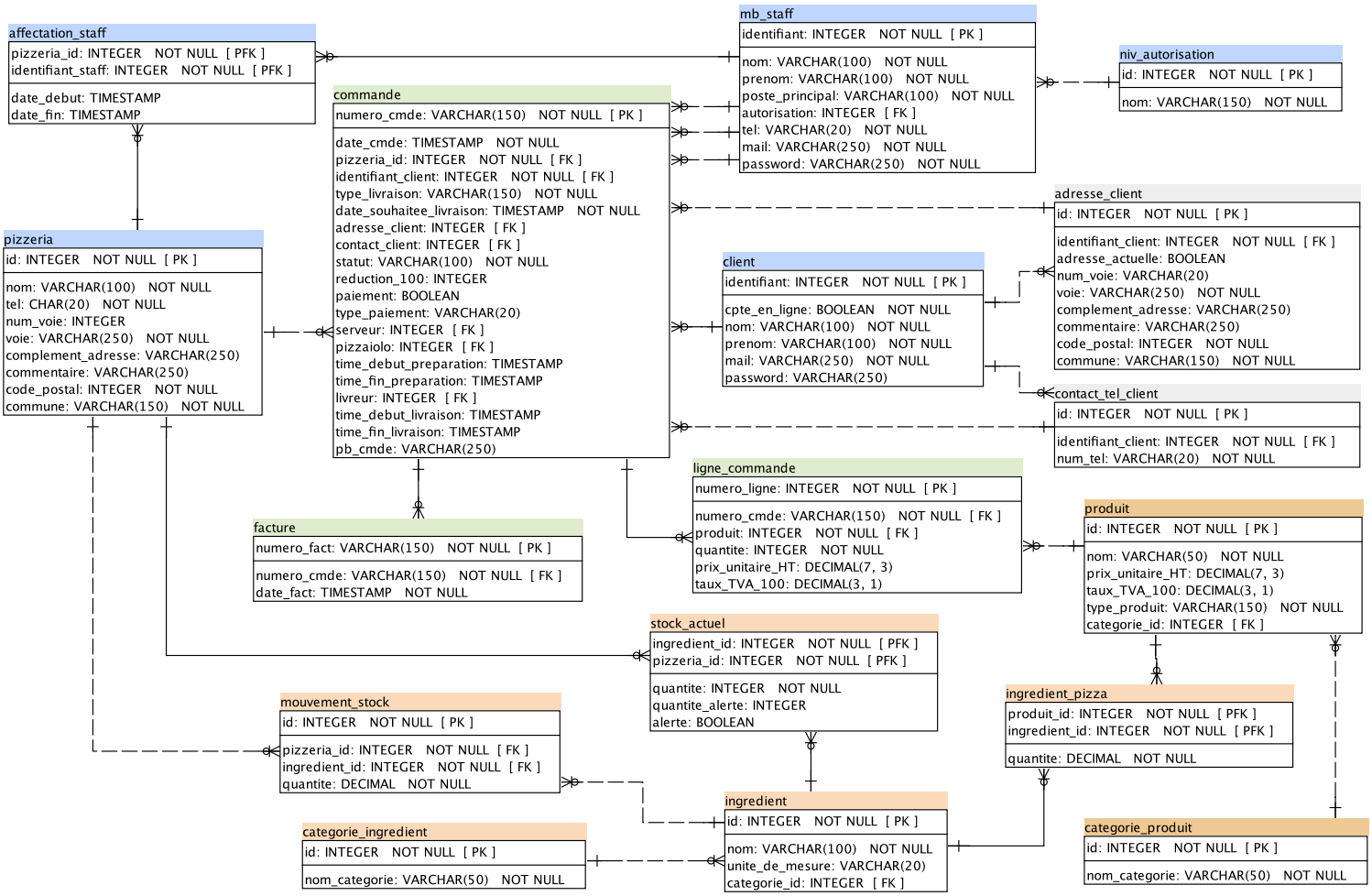
Lorsqu'une commande a pu être menée à son terme, autrement dit lorsque le(la) livreur(euse) confirme la livraison, le programme que nous réaliserons fera passer automatiquement le ***statut*** de la commande à « Livrée » et génèrera une instance de la classe ***Facture*** associée à la classe ***Commande*** par une association ***one-to-one***. Une commande ne peut aboutir qu'à une facture et inversement une facture ne peut faire référence qu'à une commande.

Ainsi pour retrouver les éléments constitutifs d'une facture, il suffira de remonter à la commande à laquelle elle fait référence.

3. Architecture de la base de données

Dérivé du diagramme de classe UML précédent, le modèle Physique de données ci-dessous permet de modéliser la base de données relationnelle nécessaire au système d'OCPizza.

3.1 Modèle Physique de Données



Nous retrouvons donc ici les différentes classes rencontrées dans le chapitre précédent mais sous forme de tables dans lesquelles sont définies les clés primaires [PK] et étrangères [FK] reliant les tables entre elles ainsi que les formats et caractéristiques des différentes colonnes de ces tables.

3.2 Description des tables

3.2.1 Tables client, adresse_client et contact_tel_client :

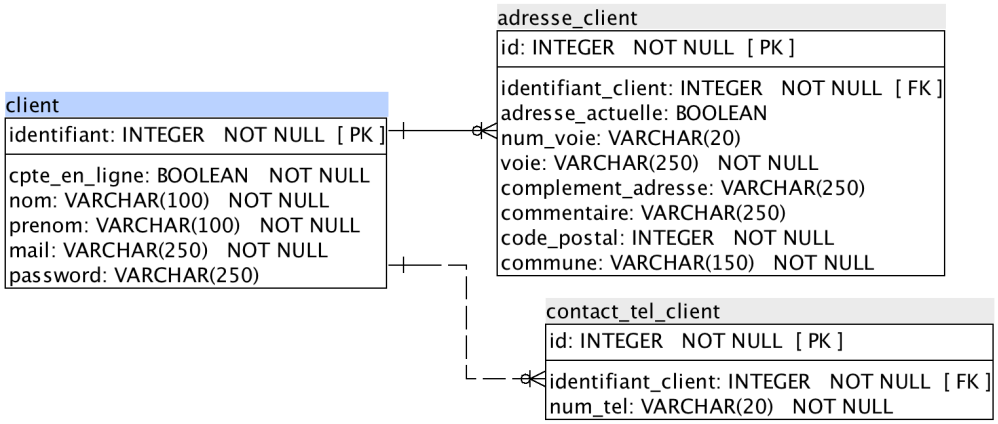


Table client :

identifiant : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouveau client. Concernant les ventes à emporter, nous envisageons la création d'un compte spécifique qui permettra de ne pas renseigner de client particulier.

cpte_en_ligne : de type booléen, cette colonne passera à la valeur « true » lorsque le(la) client(e) créera son compte en ligne. Le programme pourra ainsi vérifier s'il n'existe pas déjà un compte client en ligne avec les mêmes informations (nom et prénom, mail) tout en permettant de créer des comptes pour ce(tte) même client(e) dans les pizzerias.

nom, ***prenom*** et ***mail*** : renseignés par l'employé(e) prenant en charge la commande pour le compte du client ou par l'intermédiaire du compte client ayant été utilisé.

password : de type VARCHAR et de longueur suffisante pour prendre en compte les mots de passe cryptés, celui-ci ne sera exigé par le système que lorsqu'un(e) client(e) créera son compte personnel en ligne et non lors de la prise de commande par l'intermédiaire d'un membre du staff pour le compte de ce(tte) client(e).

Table adresse_client :

id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouvelle adresse d'un client.

identifiant_client : clé étrangère de la table, elle ne peut faire référence qu'à un ***identifiant*** de la table ***client.***

adresse_actuelle : de type booléen, cette colonne indique « true » pour la dernière adresse renseignée pour un client. Elle aura par conséquent comme valeur par défaut « true » à chaque création d'une nouvelle adresse.

num_voie : de type VARCHAR(20), des informations comme les compléments de numéros (bis, ter…) seront supportés.

voie : cette colonne est prévue comme devant être obligatoirement renseignée (NOT NULL). Cependant, nous pouvons tout à fait concevoir que lors d'une vente à emporter notamment, seuls le code postal ou la commune soient des informations réellement importantes à conserver.

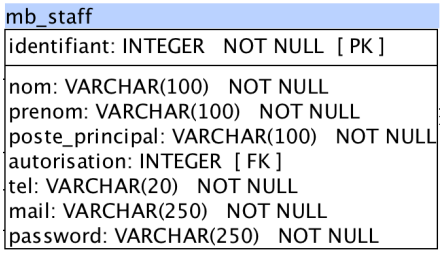
complement_adresse et ***commentaire*** : ces deux colonnes permettent de recueillir des informations annexes mais pouvant être importante pour la livraison comme le numéro d'appartement, le nom sur l'interphone …

code_postal et ***commune*** : ces colonnes doivent obligatoirement être renseignées pour justifier la création d'une ligne de la table adresse_client.

Table contact_tel_client :

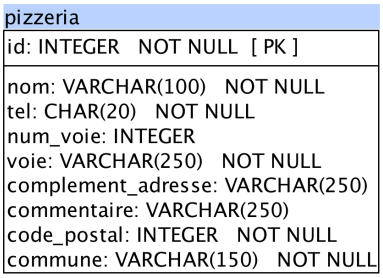
id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouveau numéro d'un client.
identifiant_client : clé étrangère de la table, elle ne peut faire référence qu'à un identifiant de la table ***client.***
num_tel : de type VARCHAR(20) il devrait supporter les différents type de formats de numéro de téléphone comme les préfixes internationaux(ex : +33)

3.2.2 Table mb_staff :



identifiant : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouveau membre de l'équipe enregistré dans le système.
nom, ***prenom*** et ***poste_principal*** : ces informations sont à renseigner obligatoirement pour la création d'un compte employé.
autorisation : clé étrangère de la table, elle ne peut faire référence qu'à un identifiant de la table ***niv_autorisation*** et ne pourra être renseignée que par la direction ou un membre de l'équipe d'OCPizza disposant d'un niveau d'autorisation suffisant (cf 3.2.5)
tel et ***mail*** : de type VARCHAR et de capacité suffisante pour couvrir les différents formats, ces informations sont obligatoirement renseignées
password : mot de passe obligatoirement à générer à la création du compte.

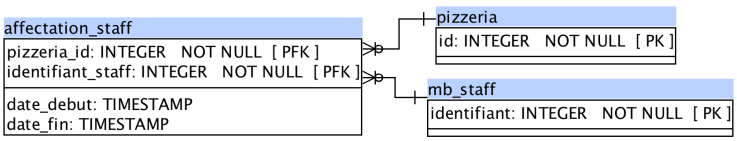
3.2.3 Table pizzeria :



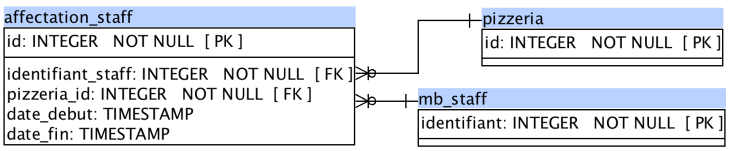
id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouvelle pizzeria enregistrée dans le système.
nom : dénomination de chaque pizzeria cette colonne peut renseigner le nom du quartier éventuellement si les pizzeria n'ont par de nom distincts.
tel : de type VARCHAR et de capacité suffisante pour couvrir les différents formats, cette information est obligatoirement renseignée.
num_voie : de type VARCHAR(20), des informations comme les compléments de numéros (bis, ter…) seront supportés.
voie : cette colonne est prévue comme devant être obligatoirement renseignée (NOT NULL).
complement_adresse et ***commentaire*** : ces deux colonnes permettent de recueillir des informations annexes.
code_postal et ***commune*** : ces colonnes doivent obligatoirement être renseignées.

3.2.4 Table affectation_staff :

Proposition 1



Proposition 2



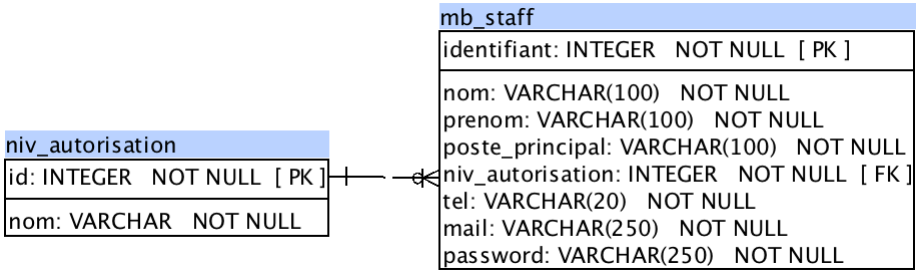
pizzeria_id : clé étrangère de la table, elle ne peut faire référence qu'à un identifiant de la table ***pizzeria.***
identifiant_staff : clé étrangère de la table, elle ne peut faire référence qu'à un identifiant de la table ***mb_staff.***
date_debut : de type DATETIME, cette colonne renseigne la date et l'heure de la prise de fonction de l'employé(e).
date_fin : de type DATETIME, cette colonne renseigne la date et l'heure de la fin de la mission de l'employé(e).
Cette table peut nous amener à discuter ensemble des besoins d'OCPizza.
Le but est-il :

- de retrouver aisément l'affectation actuelle de chaque salarié(e), de lui signaler son affectation (*Proposition 1*)
- ou de conserver l'historique des affectations? (*Proposition 2*)

Pour répondre à cette question, nous devons considérer dans quelle mesure les salariés sont amenés à changer de pizzeria d'affectation?

Remarque : Comme nous le verrons ci-dessous, nous saurons dans quelle pizzeria un salarié a travaillé dans les informations de la table commande qui renseignera qui a pris la commande, qui l'a préparée et qui l'a livrée. Pour retrouver simplement où a travaillé un employé à tel date, il nous suffirait de retrouver les commandes dans lesquelles le salarié est intervenu et de constater quelle pizzeria a pris en charge ces commandes.

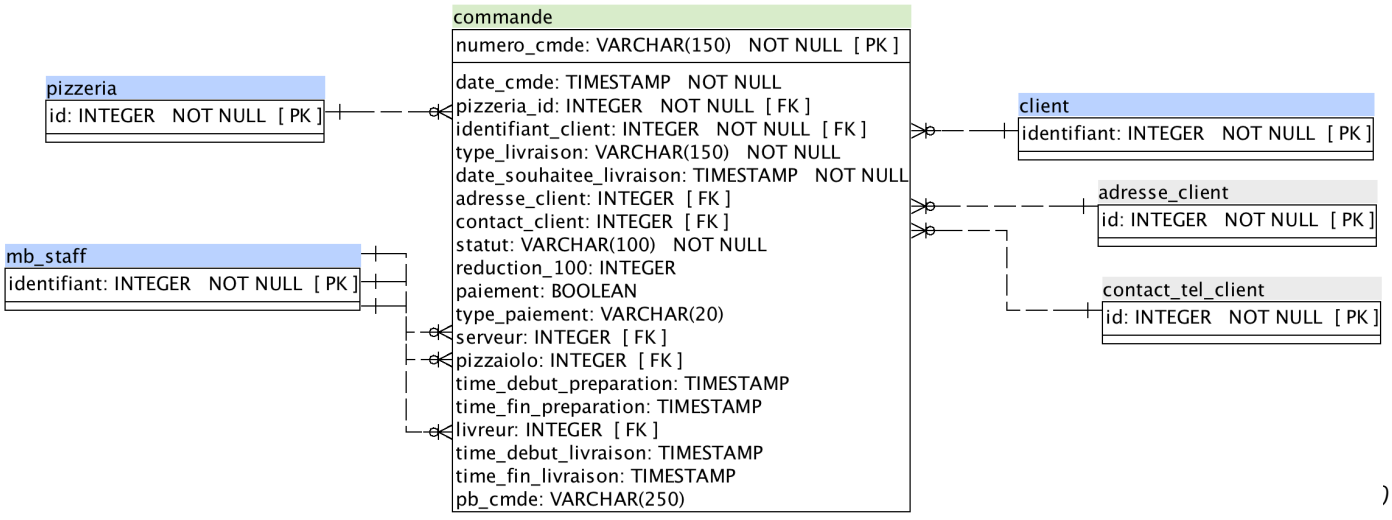
3.2.5 Table niv_autorisation :



Cette table renseigne les différents niveaux d'autorisation accordés aux membre de l'équipe d'OCPizza pour l'utilisation du système.

id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouveau niveau d'autorisation enregistrée dans le système
nom : cette colonne renseigne la dénomination du niveau d'autorisation.

3.2.6 Table commande :



INFORMATIONS D'IDENTIFICATION DE LA COMMANDE

numero_cmde : clé primaire de la table elle assure l'unicité des commandes. Nous vous proposons de générer des numéros de commandes spécifiques à chaque pizzeria, en utilisant le format VARCHAR et en chargeant le programme d'incrémenter les numéros au fur et à mesure des créations des commandes. Ainsi, nous pourrons définir les numéros de commande avec une référence à la pizzeria concernée et/ou par exemple l'année , le mois ou toute autre information qui pourrait paraitre utile pour faciliter l'information.

pizzeria_id : clé étrangère de la table, elle ne peut faire référence qu'à une instance de la table ***pizzeria.***

Identifiant_client : clé étrangère de la table, elle ne peut faire référence qu'à une instance de la table ***client*** et sera renseignée soit par l'intermédiaire du compte client ayant été utilisé soit renseigné par l'employé(e) prenant acte de la commande pour le client.

statut : information que le programme modifiera à chacune des étapes de la commande :

- « En attente de préparation » —> création d'une commande
- « En cours de préparation » —> prise en charge de la commande par le pizzaiolo
- « En attente de livraison » —> renseignement de la fin de la préparation par le pizzaiolo
- « En cours de livraison » —> renseignement de la prise en charge par le(a) livreur(euse)
- « Livrée » —> renseignements du succès de la livraison par le(la) livreur(euse)

INFORMATIONS CONCERNANT LE PAIEMENT DE LA COMMANDE

reduction_100 : des réductions pourraient éventuellement être prévues (pour les salariés, clients réguliers, coupons…), elles sont ici renseignées en nombre entier à diviser par 100 par le programme.

paiement : de type booléen, cette colonne renverra l'information sur le règlement effectif de la commande communiqué par l'API de paiement en ligne (cf 4.1) ou par la prise en compte du règlement par le(la) livreur(euse).

type_paiement : information renseignée par l'intermédiaire du compte client ayant été utilisé ou renseigné par l'employé prenant acte de la commande pour le client, les possibilités seront :
« Chèque » / « Espèce » /« CB » / « En ligne »
Une table d'énumération sera créée afin de permettre l'ajout éventuel de nouveaux types de paiement (tickets restaurants par exemple) et faciliter leur sélection.

PRISE D'ACTE DE LA COMMANDE

serveur : clé étrangère de la table, elle ne peut faire référence qu'à une instance de la table ***mb_staff*** et sera renseignée par l'intermédiaire du compte de l'employé(e) déclarant enregistrer la commande ou sera renseignée comme NULL.

date_cmde : de type DATETIME, elle est définie par défaut comme étant la date et l'heure de création de la commande.

INFORMATIONS SUR LA PRÉPARATION DE LA COMMANDE

pizzaiolo : clé étrangère de la table, elle ne peut faire référence qu'à une instance de la table ***mb_staff*** et sera renseignée par l'intermédiaire du compte de l'employé(e) déclarant prendre en charge la préparation de la commande.

time_debut_preparation : de type TIME, elle est renseignée automatiquement par le programme à la date et heure de prise en charge de la commande par le pizzaiolo soit lorsque le précédent attribut est renseigné.

time_fin_preparation : de type TIME, elle est renseignée par l'intermédiaire du compte de l'employé(e) déclarant avoir fini la préparation de la commande.

INFORMATIONS SUR LA LIVRAISON DE LA COMMANDE

type_livraison : information renseignée par le client en ligne ou par le membre de l'équipe enregistrant la commande et indiquant s'il s'agit d'une commande à emporter avec retrait en boutique ou à livrer au domicile du client. Une table d'énumération peut éventuellement être créée si OCPizza pense faire évoluer ses offres notamment en ajoutant la possibilité de dégustation sur place.

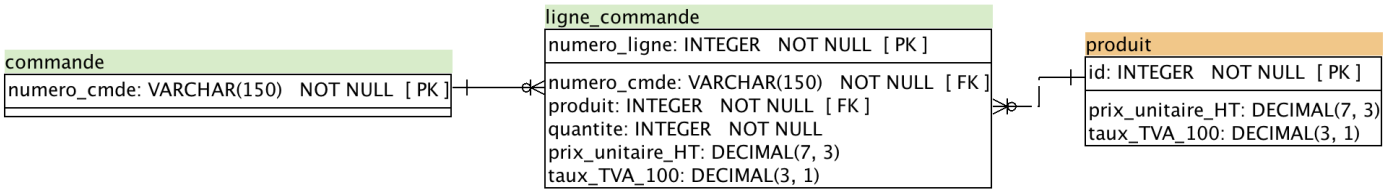
date_souhaitee_livraison : de type DATETIME, cette information est renseignée par le client lors de la prise de commande. Il peut s'agir d'une livraison « dès que possible », dans ce cas le système renseignera automatiquement une estimation du temps de livraison de la commande, ou d'une livraison pour une date et heure spécifique indiquée par le client.

livreur : clé étrangère de la table, cette colonne ne peut faire référence qu'à une instance de la table ***mb_staff***. Il s'agit ici en effet de renseigner l'identité de l'employé(e) ayant pris en charge la livraison de la commande. Le livreur peut être l'employé(e) transportant la commande au domicile du client comme celui remettant la commande au client en boutique selon le type de commande indiqué dans la colonne précédente.
id_adresse **:** clé étrangère de la table, cette colonne ne peut faire référence qu'à une instance de la table ***adresse_client.*** Elle n'est pas obligatoirement à remplir notamment lorsque la commande est de type « À emporter ».
contact_client **:** clé étrangère de la table, cette colonne ne peut faire référence qu'à une instance de la table ***contact_tel_client.*** Il est en effet important pour le livreur de disposer directement du numéro de téléphone sur le quel le client est joignable pour la commande en question. De même que pour l'adresse, cette colonne peut ne pas être renseignée notamment pour les commandes de type « À emporter ».
time_debut_livraison : de type TIME, elle est renseignée automatiquement par le système lorsque le livreur déclare prendre en charge la livraison, cette colonne indique la date et l'heure de la prise en charge de la livraison.
time_fin_livraison : de type TIME, elle est renseignée automatiquement par le système lorsque le livreur déclare avoir effectué avec succès la livraison. Elle indique la date et l'heure de la remise effective de la commande au client. Pour les commandes de type « livraison à domicile », il s'agira donc du temps de trajet du livreur et pour les commandes de type « à emporter », il sera automatiquement défini comme la colonne précédente correspondant au début de la livraison.

REMARQUE OU PROBLÈME RENCONTRÉ SUR LA COMMANDE

pb_cmde : de format TEXT, cette colonne est renseignée par l'un des employés pour signaler toute difficulté à faire remonter à la direction. En effet, comme indiqué dans le dossier de spécifications fonctionnelles, il nous semble important de conserver les informations concernant les difficultés rencontrées lors de la livraison des commandes notamment et de permettre ainsi leur communication à la direction. Il peut en effet s'agir de l'absence du client au domicile, d'un refus de paiement, de problème du moyen de locomotion. Afin de faciliter le renseignement de cette information par le livreur, nous pouvons envisager la création d'une table d'énumération renseignant les différentes possibilités.

3.2.8 Table ligne_commande :



numero_ligne **:** clé primaire de la table, elle se génère en auto-incrémentation à chaque nouvelle ligne de commande enregistrée dans le système.
numero_cmde **:** clé étrangère de la table, elle ne peut faire référence qu'à une instance de la table ***commande***
produit **:** clé étrangère de la table, elle ne peut faire référence qu'à une instance de la table ***produit***
quantite **:** cette colonne renseigne le nombre de produit identique sur une même commande
prix_unitaire_HT **:** cette colonne renseigne automatiquement par requête SQL le prix HT applicable à un produit au moment du passage de la commande. Cette information n'est pas associée à la colonne prix_unitaire_HT de la table produit d'où elle provient afin de garder une trace sécurisée des tarifs au moment de la commande. Nous lui allouons une capacité de 3 décimales afin d'assurer la précision des prix après application du taux de TVA.
taux_TVA_100 **:** de même, cette colonne est automatiquement renseignée par le système à partir de la colonne taux_TVA_100 de la table produit sans y être directement associée.

3.2.9 Tables produit et categorie_produit :

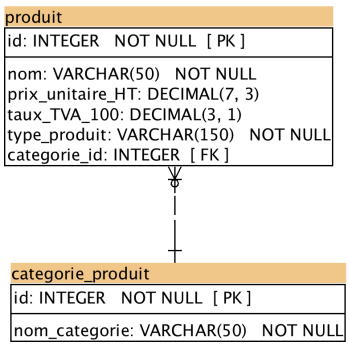


Table produit :

id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouveau produit enregistré dans le système.
nom : cette colonne renseigne l'appellation du produit. Il peut s'agir de pizzas 4 fromages, d'une canette de coca-cola, d'un pot de glace individuel ou encore de supplément champignon sur une pizza. La capacité de 50 caractères devrait suffire pour ce type d'information.
prix_unitaire_HT : cette colonne est à renseigner lors de la création d'un produit ou lors d'une évolution des prix des produits. Nous vous proposons de ne pas rendre cette information obligatoire afin de permettre d'insérer éventuellement des produits sans valeur marchande
taux_TVA_100 : de même, colonne à renseigner à création et évolution du taux, non obligatoire.
type_produit : de manière obligatoire, elle doit être renseignée et pour l'instant, nous visualisons plusieurs types de produits envisageables :

- « transformés » : différentes pizzas, plats de pâtes, desserts fait maison…
- « revente » : boissons, glaces, produits non préparés sur place…
- « suppléments » : ingrédients supplémentaires à ajouter à une recette.
- Autres : à définir avec OCPizza

Une table d'énumération pourra éventuellement être créée si OCPizza envisage d'autre types de produits.
categorie_id : clé étrangère de la table, elle ne peut faire référence qu'à une instance de la table ***categorie_produit.***

Table categorie_produit :

id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouvelle catégorie de produits enregistrée dans le système.
nom_categorie : cette colonne, obligatoire (NOT NULL), renseigne la désignation de la catégorie. Elle reste à définir avec OCPizza. Elle pourrait contenir différentes catégories de type :
« Pizzas » /« Pâtes » /« Plats chauds » / « Salades » / « Boissons » / « Dessert » …

3.2.10 Tables ingredient et categorie_ingredient :

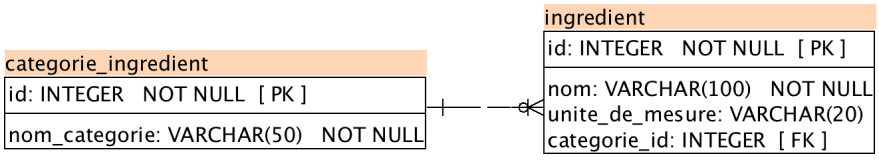


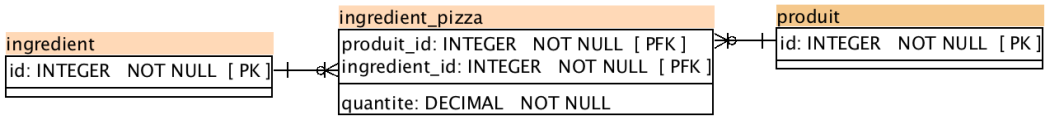
Table ingredient :

id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouvel ingrédient enregistré dans le système.
nom : cette colonne correspond à la dénomination de l'ingrédient. Il peut s'agir de champignons, de fromage, de canette de coca-cola ou de pot de glace individuel. Elle est à renseigner obligatoirement (NOT NULL)
unite_de_mesure : cette colonne est à renseigner de manière facultative et correspond à l'unité de mesure de l'ingrédient concerné (grammes, centilitres, unité…)
categorie_ingredient : clé étrangère de la table, elle ne peut faire référence qu'à une instance de la table ***categorie_ingredient.***

Table categorie_ingredient :

id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouvelle catégorie d'ingrédient enregistrée dans le système.
nom : cette colonne, obligatoire (NOT NULL), renseigne la désignation de la catégorie. Elle reste à définir avec OCPizza. Elle pourrait contenir différentes catégories de type :
« bases » / « pâtes » / « viandes & poissons » / « fromages » / « condiments » /« legumes »/
« boissons » /« snacks » /« desserts » …

3.2.11 Table ingredient_pizza :



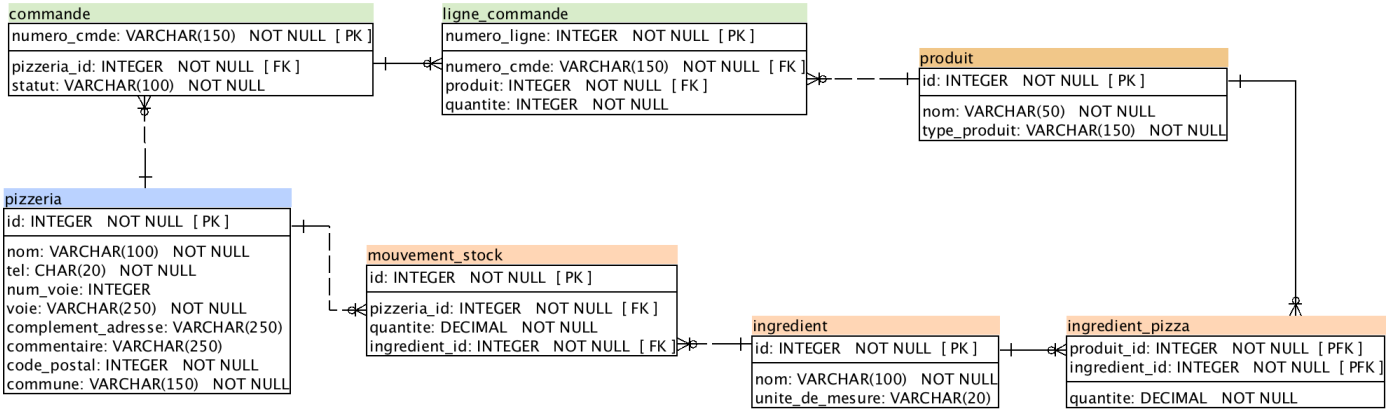
La table ingredient_pizza prend comme clé primaire le couple ***produit_id*** et ***ingredient_id***
produit_id : clé primaire de la table mais également clé étrangère, elle ne peut faire référence qu'à une instance de la table ***produit***.
ingredient_id : clé primaire de la table mais également clé étrangère, elle ne peut faire référence qu'à une instance de la table ***ingredient***.
quantite : cette colonne à renseigner de manière obligatoire (NOT NULL), correspond à la quantite nécessaire d'un ingrédient pour réaliser la recette d'un produit.

NB :
*Les produits de type revente seront donc forcément en doublons dans les tables **produit** et **ingredient** :*
- *en tant qu'ingrédients afin de permettre la gestion de leur stock dans les différentes pizzerias*
- *en tant que produits afin de permettre leur vente et leur insertion dans les commandes des clients*

Quant aux suppléments, ils apparaissent :
- *en tant qu'ingrédients dans la table **ingredient***
- *en tant qu'ingrédients d'une recette de supplément dans la table **ingredient_pizza** puisque nous avons besoin de la quantité nécessaire pour réaliser le supplément en question*
- *En tant que produit dans la table **produit** en vue de pouvoir les insérer dans les commandes.*

Concernant des ingrédients comme la farine, nécessaire à la préparation des pâtes à pizza, nous considérerons les pâtes à pizza elle-même en tant qu'ingrédients et non les éléments dont elles sont constituées. En effet, nous partons du principe que le pizzaiolo préparera un certain nombre de pâtes (commune à priori à toutes les pizzas) et c'est ce stock dont nous devons tenir compte de notre point de vue satisfaction client.

3.2.12 Table mouvement_stock :



Comme indiqué dans la description de notre diagramme de classe, les mouvements de stock peuvent provenir d'une mise à jour manuelle effectuée par la direction ou un(e) salarié(e) suite à l'achat de nouveaux ingrédients ou à un ajustement du stock existant, mais également par une mise à jour automatique qui sera déclenchée pour chaque ligne de commande dont le statut sera passé en « En cours de préparation ». Le système viendra prendre les informations de quantité du produit et la référence du produit dans la classe ***ligne_commande***, les ingrédients et la quantité nécessaire pour réaliser le produit concerné dans la classe ***ingredient_pizza*** et viendra répercuter toutes ces informations dans la classe ***mouvement_stock***.

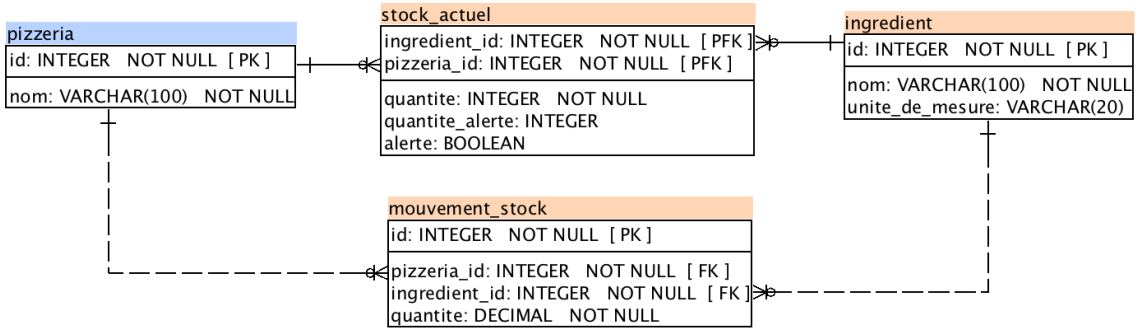
Toutes les colonnes de cette tables devront être obligatoirement renseignée (NOT NULL) pour générer un nouveau mouvement de stock.

id : clé primaire de la table, elle se génère en auto-incrémentation à chaque nouveau mouvement de stock enregistré dans le système.
pizzeria_id : clé étrangère de la table, cette colonne ne peut faire référence qu'à une instance de la table ***pizzeria*** et sera renseignée :
- soit à l'initiative de la personne mettant à jour le stock d'ingrédients (renouvellement ou ajustement)
- Soit automatiquement par le programme avec l'identifiant de pizzeria contenu dans la table ***commande***

ingredient_id : cette colonne est donc renseignée :
- soit à l'initiative de la personne mettant à jour le stock d'ingrédients (renouvellement ou ajustement)
- Soit automatiquement par le programme à partir de la colonne ***ingredient_id*** de la table ***ingredient_pizza***

quantite : de même, cette colonne est renseignée :
- soit à l'initiative de la personne mettant à jour le stock d'ingrédients (renouvellement ou ajustement)
- Soit automatiquement par le programme à partir des colonnes quantite des tables ***ingredient_pizza*** et ***ligne_commande***

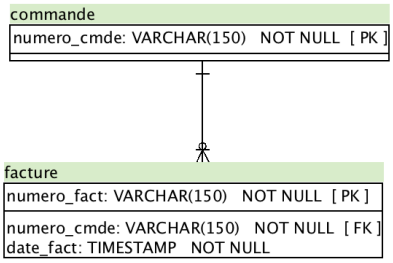
3.2.13 Table stock_actuel :



La table stock_actuel prend comme clé primaire le couple **pizzeria_id** et ***ingredient_id.***
Les trois premières colonnes de cette table seront automatiquement modifiées par le système à chaque mouvement de stocks. De la table ***mouvement_stock.***
pizzeria_id : clé primaire de la table mais également clé étrangère, elle ne peut faire référence qu'à une instance de la table ***pizzeria*** et sera renseignée automatiquement par le programme avec l'identifiant de pizzeria contenu dans la table ***mouvement_stock.***
ingredient_id : clé primaire de la table mais également clé étrangère, elle ne peut faire référence qu'à une instance de la table ***ingredient*** et sera renseignée automatiquement par le programme avec l'identifiant de l'ingrédient concerné ***.***

quantite : cette colonne est mise à jour en ajoutant la quantité (positive ou négative) indiquée dans la table ***mouvement_stock***.
quantite_alerte : Afin d'éviter une rupture de stock impromptue, nous vous proposons de définir des niveaux d'alerte rappelant à la personne en charge des achats de renouveler le stock d'un ingrédient.
alerte : cette colonne indiquera automatiquement « true » dans le cas où la quantité du stock actuel est inférieur à la quantité alerte définie.

3.2.14 Table facture :



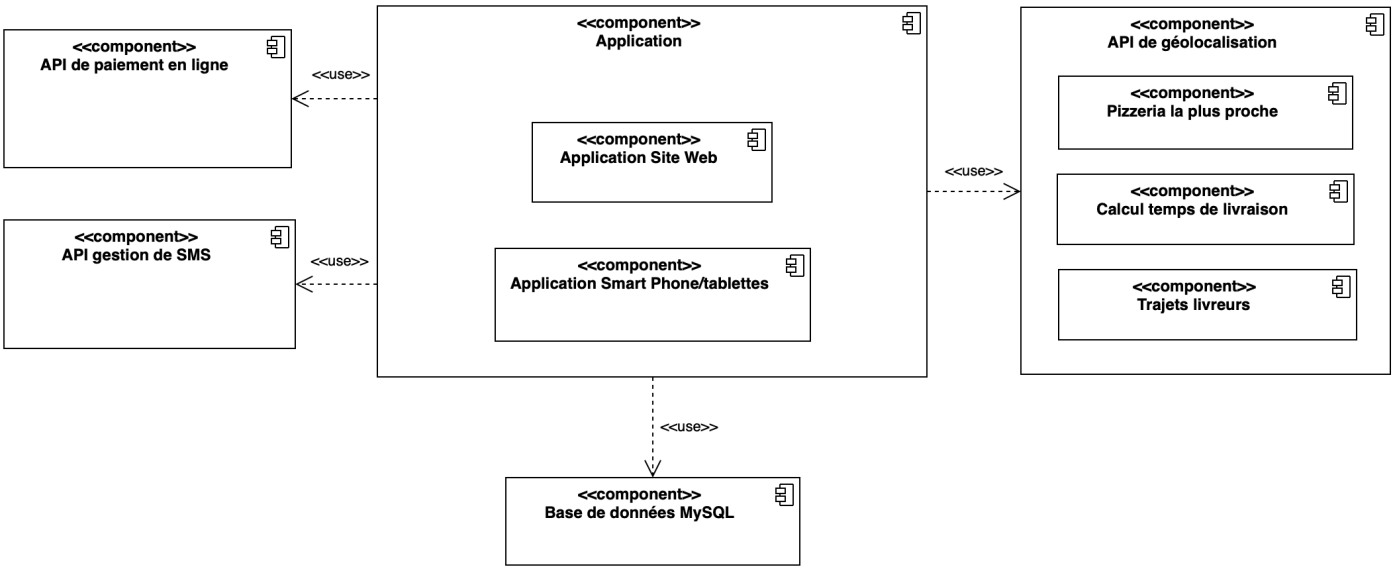
numero_fact : clé primaire de la table elle assure l'unicité des commandes. Afin de générer des numéros de factures spécifiques à chaque pizzeria, nous vous proposons d'utiliser le format VARCHAR et nous chargerons le programme d'incrémenter les numéros au fur et à mesure de la création des factures.
Ainsi, nous pourrons définir nos numéros de factures avec une référence à la pizzeria concernée et par exemple l'année , le mois ou toute autre information qui pourrait paraitre utile pour faciliter l'information.

numero_cmde : clé étrangère de la table, cette colonne ne peut correspondre qu'à une instance de la table commande. Le ***numéro_cmde*** ici indiqué est donc également de type VARCHAR.
date_fact : de type DATETIME, cette colonne sera par défaut définie comme la date et l'heure de la livraison effective.

4. Composants et déploiement du système

4.1 Organisation des composants du système

Le diagramme de composant ci-dessous décrie l'organisation du système du point de vue des éléments logiciels sous forme de composants. Il met en évidence les relations et dépendances entre ceux-ci, internes comme externes.



Le système devra faire appel à différents composants externes pour répondre aux fonctionnalités souhaitées. L'application utilisera la base de données MySQL pour récupérer les données nécessaires à l'utilisation des différentes API.

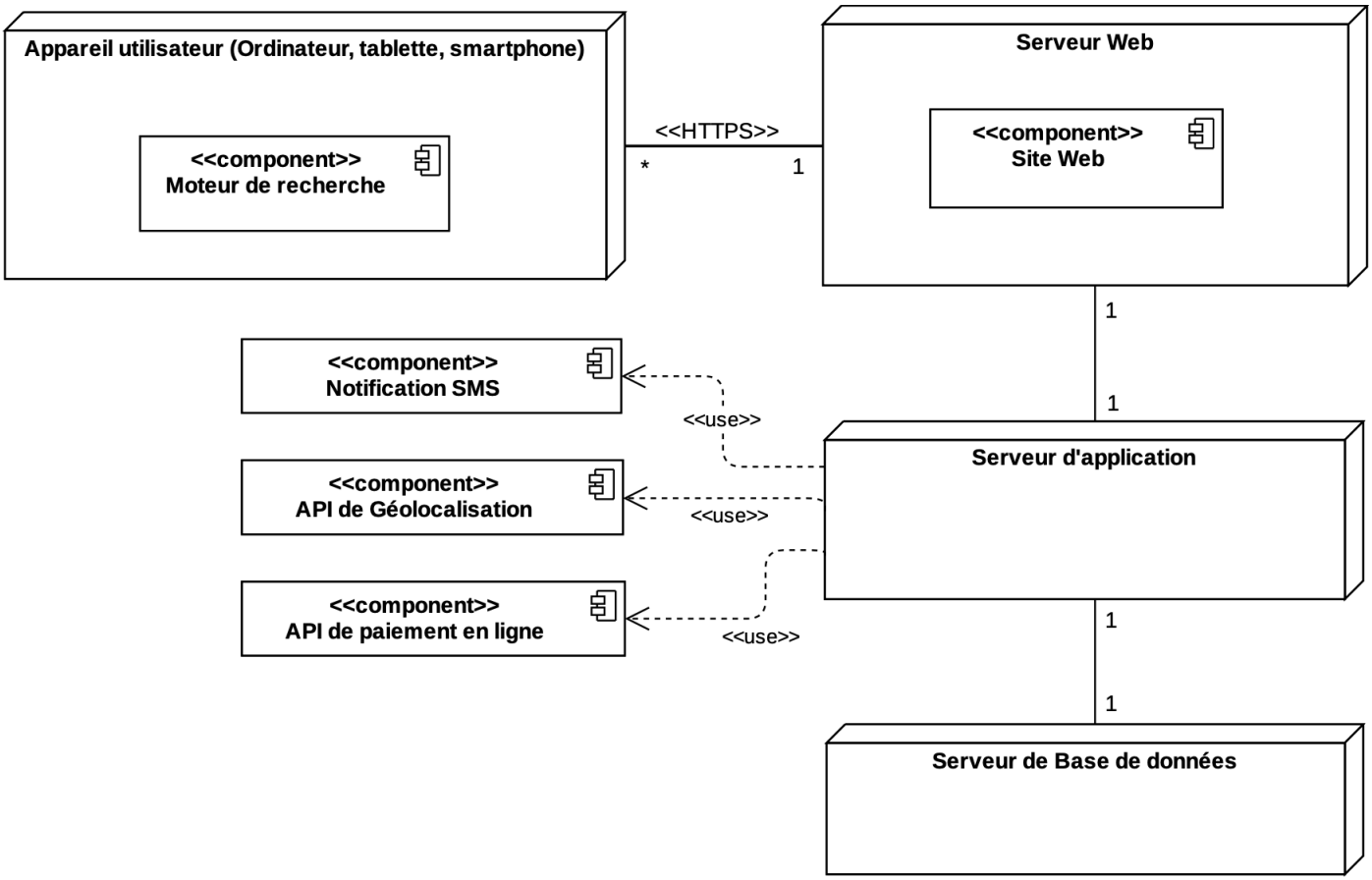
La base de données contient les adresses des pizzerias ainsi que les adresses de livraisons des clients. En utilisant une ***API de géolocalisation***, l'application permettra la recherche de la pizzeria à proximité de l'adresse client, permettra de calculer les temps de livraison nécessaires pour une commande et de proposer des itinéraires de trajets aux livreurs.

De même l'application utilisera une ***API de gestion des paiement en ligne***. Les informations d'identification client ainsi que du montant total TTC des factures pourront ainsi être calculées par l'application grâce aux informations contenues dans la base de données MySQL (table facture). Puis cette API permettra d'échanger directement avec le service de banque en ligne et de renvoyer au système l'information de la réalisation effective du paiement.

Enfin, nous pouvons envisager l'ajout d'un appel à un service de ***gestion d'envoi automatique de SMS*** aux clients afin de les informer des différentes informations concernant l'avancement du traitement de leur commande.

Pour ces différents services, de nombreuses API sont envisageables et nous reviendrons ultérieurement sur le choix de chacun d'eux pour analyser leur adaptabilité au système et le coût qu'ils représentent.

4.2 Architecture de déploiement



Le diagramme de déploiement se rapproche encore plus de la réalité physique, puisqu'il identifie les éléments matériels (PC, Modem, Station de travail, Serveur, etc.), leur disposition physique (connexions) et la disposition des exécutables (représentés par des composants) sur ces éléments matériels.
Autrement dit, le diagramme de déploiement décrit la disposition des ressources matérielles, appelés noeuds, composant le système et montre la répartition des composants sur ces matériels, ainsi que les chemins de communications entre les différents noeuds.

Ainsi, le noeud ***appareil utilisateur,*** correspondant aux différents supports d'utilisation du système par les différents acteurs (clients comme membre du staff d'OCPizza), utilisera un composant ***moteur de recherche*** qui via des ***requêtes HTTPS*** se connecteront au second noeud, le ***serveur web***.
Ce dernier correspond au matériel décentralisé où se situe le site internet et qui utilisera le framework ***Django***.
Le site web est alors lié à une ***couche d'application*** contenant tout le code de toutes les fonctionnalités du système. Celui-ci, comme proposé dans notre dossier de spécifications fonctionnelles sera programmé en ***Python***.
Ce serveur d'application sera relié à la ***base de données MySQL*** décrite ci-dessus et utilisera des ***API de géolocalisation, paiement en ligne et de notifications SMS.***