OC Pizza spécifications techniques

OC Pizza

Introduction.

Ce document constitue les spécifications techniques du projet OC Pizza. Le commanditaire veut gagner en efficacité en optimisant le processus de gestion des commandes. Il souhaite un site Web pour pouvoir effectuer, modifier, payer et suivre une commande.

Il veut suivre en temps réel l'évolution des commandes et celui du stock. Les préparateurs doivent avoir à disposition les recettes des pizzas.

La première partie constitue la description du domaine fonctionnel et des différentes entités utilisées dans le projet.

La seconde partie détaille les relations entre les différents composants internes ou externes du système. En effet dans une base de données relationnelle, les données sont regroupées par concept dans des tables et les concepts sont liés les uns aux autres par des relations.

La dernière partie décrit le déploiement du système.

Le domaine fonctionnel.

On doit identifier les éléments et les informations que l'on veut enregistrer dans notre base de données pour que cela forme un système cohérent, c'est le domaine fonctionnel. On utilisera dans notre étude une base de données relationnel car il y a des relations entre les différentes parties qui composent notre domaine fonctionnel.

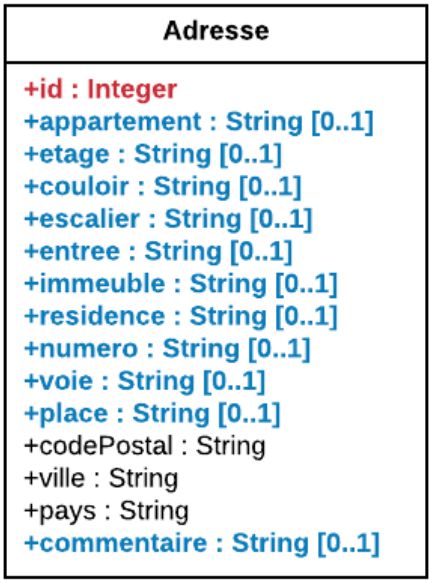
On utilise une approche orientée objet pour représenter les composants de notre domaine fonctionnel. Le diagramme de classes servira de base à la modélisation de celui-ci. Le diagramme de classes fait partie des diagrammes qui respectent la norme de modélisation graphique UML (Unified Modeling Language).

Chaque objet de notre domaine fonctionnel peut être soit réel (un client, un produit...) ou abstrait (commande, stock…). Il possède un identifiant unique et des attributs.

Les composants généraux.

On utilise un composant spécifique pour gérer les adresses car des "Utilisateur" peuvent avoir la même adresse et cela évite les redondances de données enregistrées. Les champs sont basés sur la nomenclature de la Poste. Le composant "Magasin" permet d'identifier un magasin et ses coordonnées.

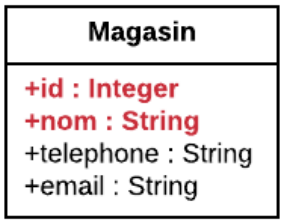
Adresse



La plupart des champs ne sont pas obligatoire comme "voie" car une personne peut habiter sur une "place" et inversement. On ajoute une ligne de commentaire dans l'adresse pour ajouter des informations comme un code de digicode.

Magasin

Un objet "Magasin" permet de classer les différents magasins. On utilise un attribut "id" pour avoir un identifiant unique qui sert de référence et on spécifie aussi que le "nom" est un identifiant pour faire des recherches plus rapidement.

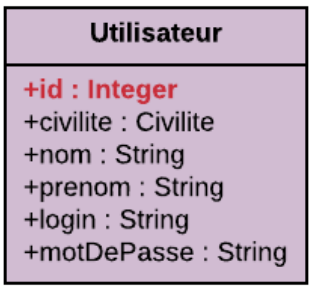


Les composants de la partie utilisateur.

L'utilisateur principal du site web est le client. On doit avoir son identification (nom, prénom, login, mot de passe) et ses coordonnées (téléphone, adresse, email). On doit aussi identifier les employés : accueil, pizzaiolo, livreur, manager, gestionnaire, direction, comptable, direction.

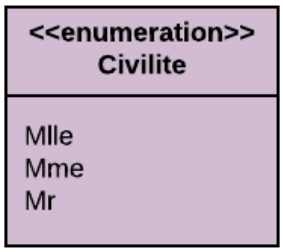
Pour les employés on a juste besoin de connaître leur rôle pour leur affecter des droits sur le système. On va créer donc un composant "Utilisateur" et deux composant qui vont en hérités "Client" et "Employe".

Utilisateur

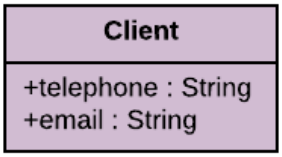


Un champ unique "id" permet d'identifier un utilisateur. Chaque "Utilisateur" a un "Magasin" attitré soit parce que c'est son magasin auprès duquel il effectue ses commandes ou soit parce qu'il y travaille. On fait référence à ce "Magasin" par son identifiant unique "id". On utilise une énumération "Civilite" pour la civilité afin de limiter les choix et d'éviter les erreurs.

Civilite <<enum>>

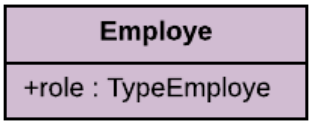


Client



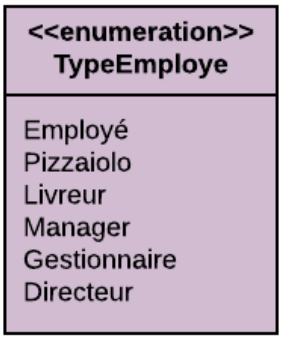
Contrairement à un employé, on doit connaître les coordonnées du "Client" pour pouvoir effectuer la vente et la livraison. L'adresse de livraison est référencée par un identifiant du composant "Adresse".

Employe

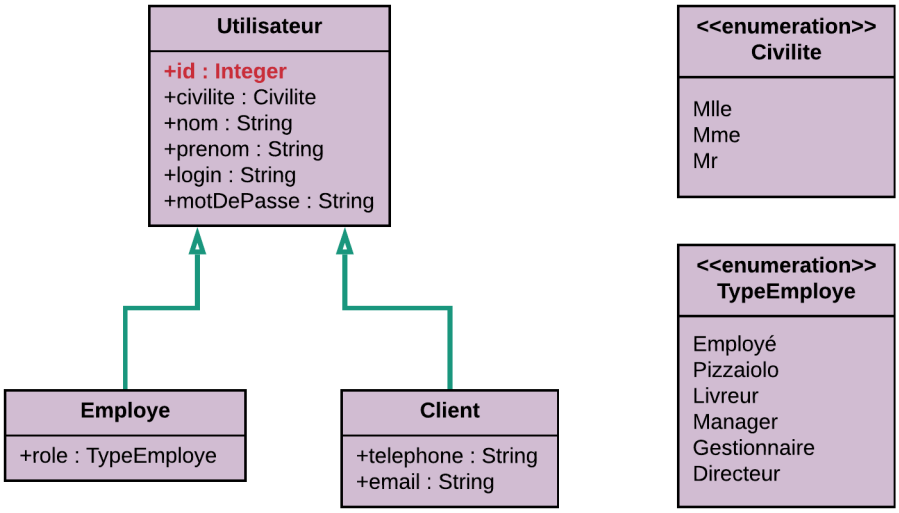


Pour définir le rôle de l'employé donc ses droits sur le système, on utilise une énumération "TypeEmploye".

TypeEmploye <<enum>>



Apperçu de la partie Utilisateur

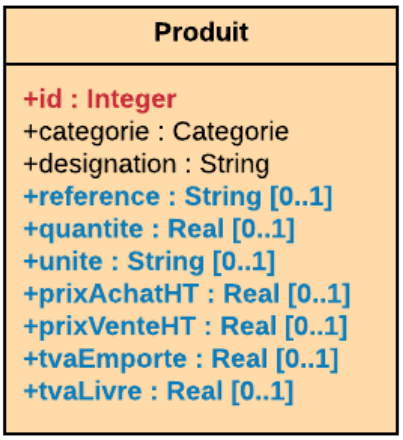


On voit bien l'héritage que font les objets "Employe" et "Client" de "Utilisateur"

Les composants de la partie produit.

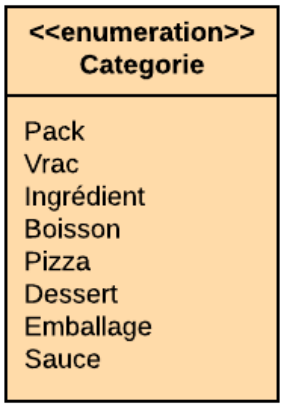
On doit gérer les ingrédients de base, les produits manufacturés et les produits additionnels comme les boissons. On doit enregistrer le prix d'achat et le prix de vente hors taxes pour chaque produit vendable. La TVA est gérée lors du passage de la commande suivant si elle est livrée ou emportée.

Produit



Le fournisseur n'est pas obligatoire si le produit est une référence de pizza. En outre comme une pizza n'a pas de prix d'achat et un ingrédient n'a pas de prix de vente, ces attributs ne sont pas indispensables. On doit aussi avoir l'information sur le taux de tva de chaque produit vendu qui peut différer pour un produit emporté ou livré.

Categorie <<enum>>

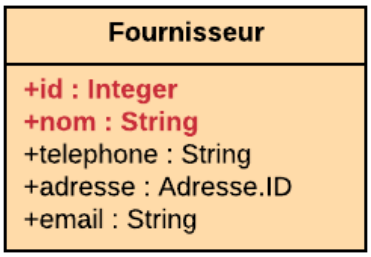


Un composant "Categorie" est une énumération qui regroupe les différentes "categorie" de "Produit". On utilisera :

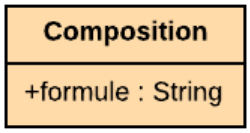
- Pack : pour les produits en groupes livrés par le fournisseur (caisse de 24 canettes),
- Boisson : pour les produits en stock qui sont à l'origine des pack mais vendu à l'unité.
- Ingrédient : pour les produits livrés par le fournisseur, au poids ou au volume et utilisés en vrac (farine, légumes, viande, poisson…)
- Vrac : pour les produits en stock qui sont à l'origine des ingrédients qui ont été livrés.

Fournisseur

Un objet "Fournisseur" permet de classer les différents fournisseurs. On utilise un attribut "id" pour avoir un identifiant unique qui sert de référence et on spécifie aussi que le "nom" est un identifiant pour faire des recherches plus rapidement.



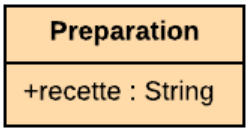
Composition



C'est l'identifiant du produit qui sert de référence pour les "Composition".

On utilise un composant spécifique pour renseigner de la composition d'un produit vendu sur le site Web, qu'il soit manufacturé dans le magasin ou non. Un autre composant "Preparation" servira pour enregistrer les recettes pour les "Pizzaiolo".

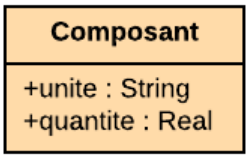
Preparation



C'est l'identifiant du produit qui sert de référence pour les "Preparation".

On doit garder l'information des produits et la quantité nécessaire pour la fabrication de chaque pizza pour pouvoir modifier le stock des ingrédients en cas de vente d'une pizza. On utilise l'objet "Composant".

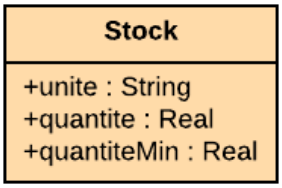
Composant



Une pizza et un ingrédient sont identifiés par leur identifiant dans le composant "Produit". On ajoute un attribut "unite" pour savoir si la quantité est un poids, un volume ou une unité.

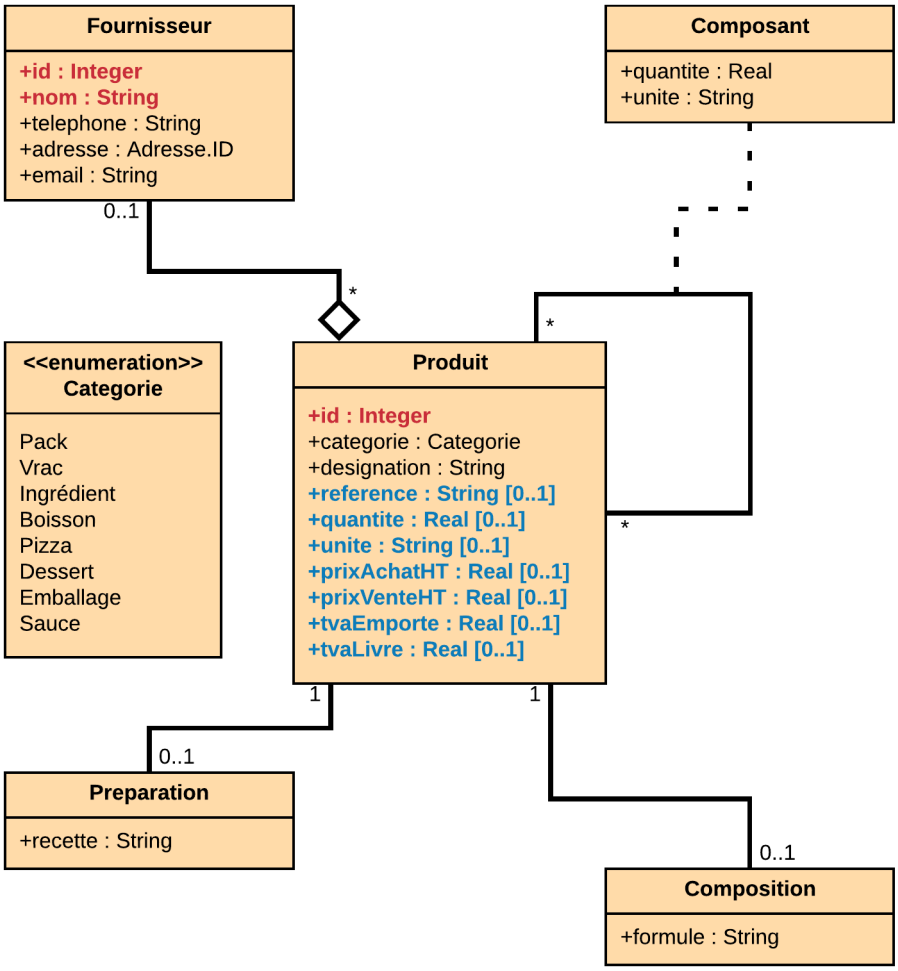
Un objet "Stock" permet de garder les informations sur les stocks de chaque produit dans chaque magasin.

Stock



L'information dépend des identifiants du "Magasin" et du "Produit". On ajoute la notion de quantité minimale dans un attribut pour pouvoir gérer par la suite les commandes automatiques d'un produit.

Apperçu de la partie Produit

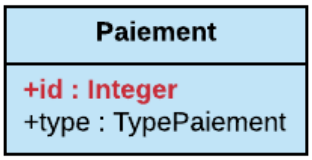


On remarque bien l'objet "Composant" qui comporte l'information entre les composés "Produit" et les composants "Produit".

Les composants de la partie paiement.

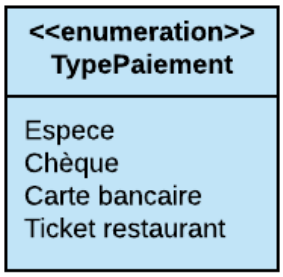
On utilise un objet "Paiement" dont héritera les différents types de paiement : "CarteBancaire", "TicketRestaurant" et "Cheque". Les paiements par espèce n'ont pas d'information spécifique donc ils n'ont pas besoin d'un composant qui hérite de "Paiement".

Paiement

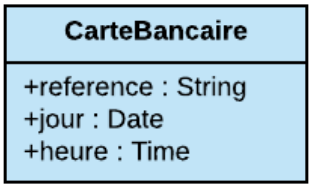


On utilise en attribut un identifiant commun à tous les paiement et un autre pour connaître le type de paiement qui fait référence à une énumération "TypePaiement".

TypePaiement <<enum>>

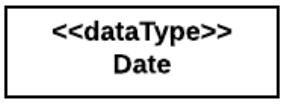


CarteBancaire

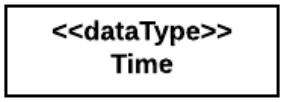


On utilise des <<dataType>> pour le jour et l'heure.

Date <<dataType>>

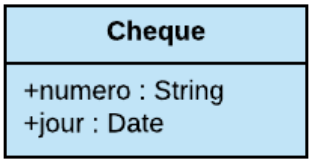


Time <<dataType>>



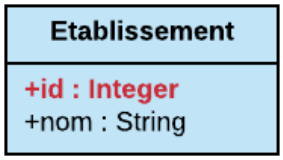
TicketRestaurant

Cheque



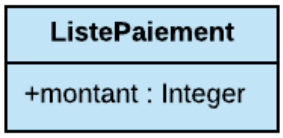
Pour éviter les mauvaises saisies et la redondance on utilise un composant "Etablissement" pour faire référence aux banques émettrices de "Cheque" et aux organismes de gestion des "TicketRestaurant".

Etablissement



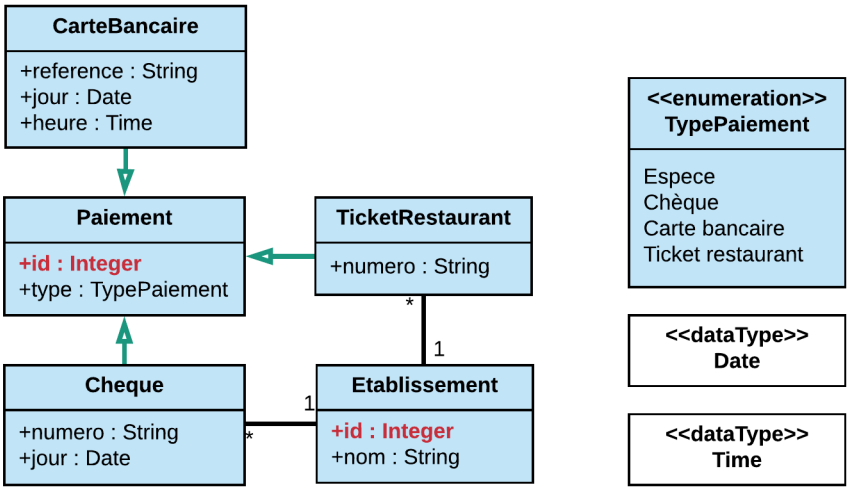
Cet objet contiendra les noms des établissements bancaires et des organismes de titre restaurant.

ListePaiement



C'est lui qui fait la liaison entre une "Commande" et ses différents "Paiement" car un "Client" peut régler sa commande avec plusieurs modes de paiement.

Apperçu de la partie Paiement

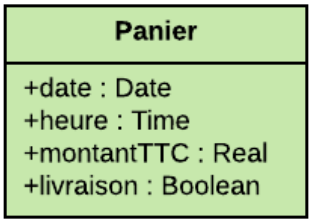


Les objets "CarteBancaire", "Cheque" et "TicketRestaurant" héritent de "Paiement". Le "Paiement" par espèce sera directement dans la classe mère car il n'a pas d'attribut en plus.

Les composants de la partie commande.

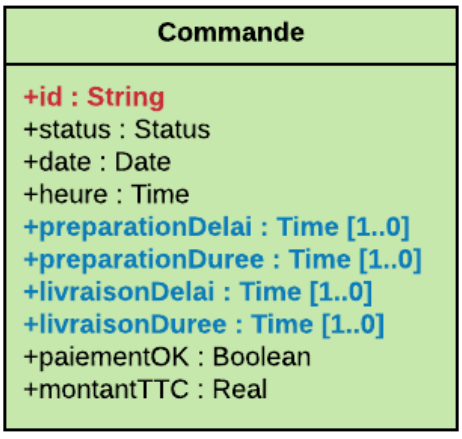
On distingue le panier que le client rempli de produits et la commande qui contient les paniers validés. Le composant "Panier" permet de garder les informations sur les produits que le Client choisi avant de valider sa commande. On doit enregistrer la date est l'heure pour pouvoir effacer les paniers non validés à partir d'un certain moment après la déconnexion du client.

Panier



Chaque panier est lié à un seul "Utilisateur". On utilise aussi les <<dataType>> "Date" et "Time". L'attribut "livraison" permet de savoir si la commande sera livrée ou retirée sur place pour calculer le montant du panier en fonction des taux de TVA.

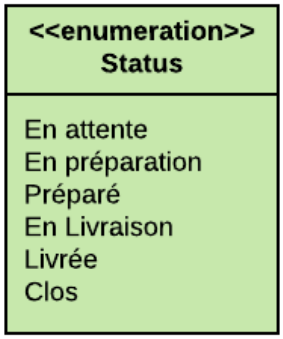
Commande



Le composant "Commande" permet de garder les informations sur le déroulement de la commande comme le "Status" qui est une énumération et les différents champs permettant de suivre les temps de préparations, le montant et la validation du paiement.

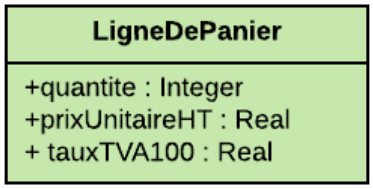
- preparationDelai : est le temps entre la validation de la commande par l'utilisateur et le début de préparation par le "Pizzaiolo".
- preparationDuree : est le temps de préparation de la commande par le "Pizzaiolo".
- livraisonDelai : est le temps mis pour finaliser la commande avant son départ en livraison.
- livraisonDurée : est le temps mis par le "Livreur" pour livrer les produits au client.

Statut <<enum>>



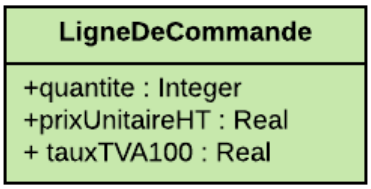
Les informations sur le contenu des "Commande" et des "Panier" sera enregistré dans les objets "LigneDeCommande" et "LigneDePanier" respectivement. On garde les prix en Hors-Taxes car si la vente est à emporter le taux de TVA n'est pas le même.

LigneDePanier



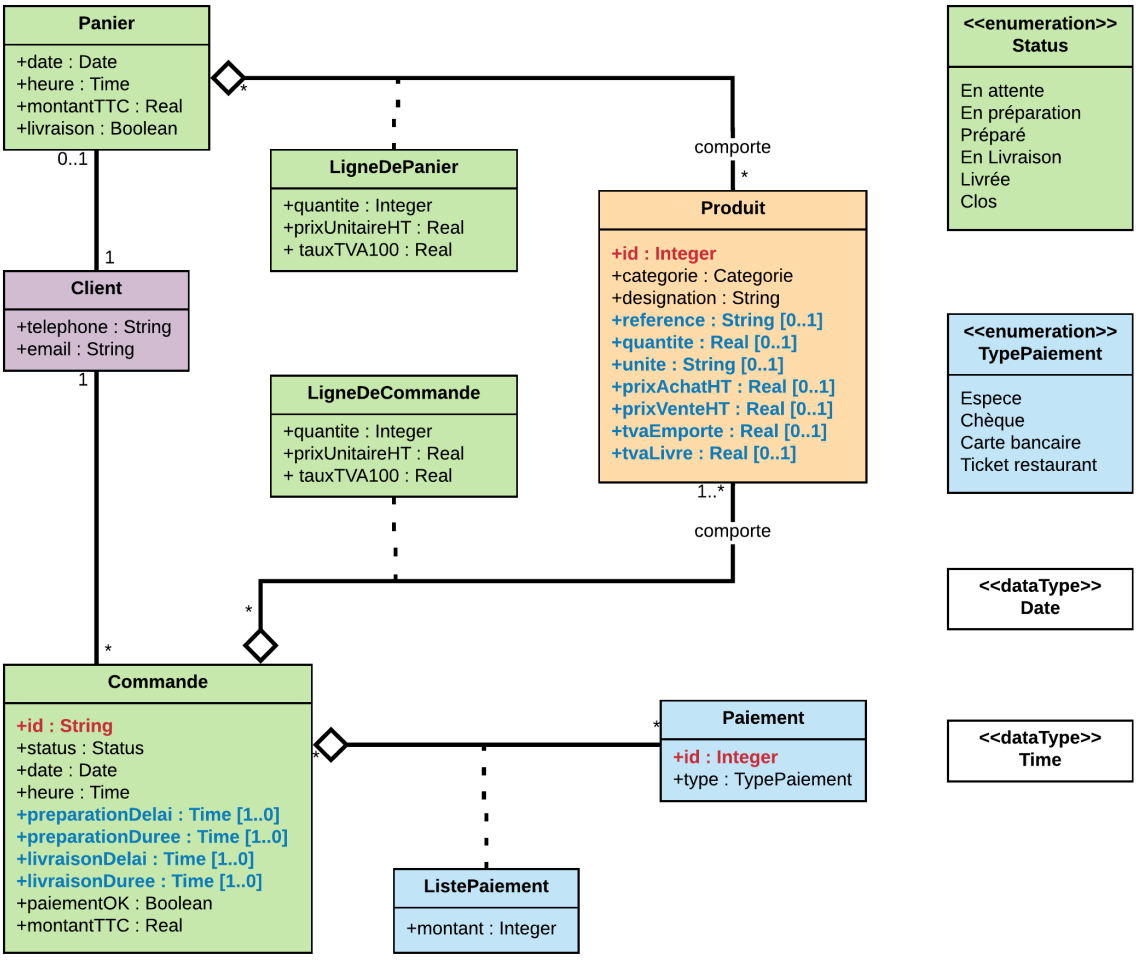
On identifie chaque tuple par l'identifiant du "Panier" donc celui "Utilisateur" et l'identifiant du "Produit".

LigneDeCommande



On identifie chaque tuple par l'identifiant de la "Commande" et celui du "Produit". On ajoute ici l'information sur la TVA du produit qui dépend de l'adresse de livraison. Lors de la validation du "Panier", il suffit de transférer les tuples de "LigneDePanier" à "LigneDeCommande" et de créer un nouvel objet "Commande".

Apperçu de la partie Commande



On remarque la similitude entre "LigneDePanier" et "LigneDeCommande" pour pouvoir déplacer les données de l'un à l'autre en cas de validation du "Panier".

Les relations entre les composants.

Les associations avec Utilisateur.

Utilisateur – Magasin

Un "Utilisateur" est lié à un magasin soit parce qu'il y travaille ou soit parce qu'il est client. Chaque "Utilisateur" est lié à un seul "Magasin". Un "Magasin" a au moins un "Utilisateur", c'est le "Magager".

Les associations avec Client.

Client – Adresse

Un "Client" a une seule "Adresse" et une "Adresse" peut être commune à plusieurs "Client" ou aucun car elle peut être "Adresse" d'un "Magasin".

Client – Panier

Confère l'association Panier-Client.

Client – Commande

Confère l'association Panier-Client.

Les associations avec Panier.

Panier – Client

Chaque "Panier" est lié à un seul "Client" mais si un "Client" est enregistré mais n'a pas encore commandé, il n'a pas de "Panier".

Panier – Produit

Un "Panier" peut contenir aucun ou plusieurs "Produit", c'est une agrégation de "Produit". Si on supprime un "Panier", les produits ne sont pas détruits. Un "Produit" peut être dans aucun ou plusieurs "Panier".

Les associations avec Commande.

Commande-Client

Chaque "Commande" est liée forcément à un seul "Client". Un "Client" peut avoir fait aucune "Commande" ou plusieurs.

Commande – Produit

Une "Commande" doit contenir au moins un ou plusieurs "Produit", c'est une agrégation de "Produit". Si on supprime une "Commande", les produits ne sont pas détruits. Les "Produit" peuvent être dans plusieurs "Commande".

Commande – Adresse

Une "Commande" a obligatoirement une et une seule "Adresse". Un "Client" pouvant commander plusieurs "Commande", on aura plusieurs "Commande" avec la même adresse. C'est aussi le cas ou plusieurs "Client" comme des colocataires habitent à la même "Adresse".

Commande – Paiement

Une "Commande" peut être une agrégation de "Paiement" si le "Client" paie avec différents moyens (ticket restaurant et espèce). Pour des raisons comptables on ne doit pas supprimer les "Paiement" si l'on supprime la "Commande". On devra faire un remboursement. Les "Paiement" ne resteront pas longtemps sans "Commande" associé mais il est clair que des "Paiement" seront dans plusieurs "Commande".

Les associations avec Produit.

Produit – Panier

Confère l'association Panier-Produit.

Produit – Commande

Confère l'association Commande-Produit.

Produit – Magasin

Chaque "Magasin" a un stock de plusieurs "Produit", c'est une agrégation. On ne considère pas comme une composition pour ne pas détruire un stock de "Produit" et éviter le gaspillage. Un "Magasin" pourra ne pas avoir de "Produit" avant son ouverture s'il n'a pas encore reçu sa marchandise. Pour rentabiliser les pizzérias on va utiliser les mêmes "Produits" dans toutes.

Produit - Fournisseur

Un "Produit" manufacturé n'a pas de "Fournisseur" sinon les autres en ont un seul. Un "Fournisseur" peut avoir aucun ou plusieurs "Produit". On retrouve ici une agrégation où l'on ne supprimera pas les "Produit" à la disparition du "Fournisseur".

Produit – Composition

On pourra indiquer pour certain "Produit" leur "Composition" qui sera unique. En effet c'est difficile d'avoir deux sodas avec la même composition et vendre deux pizzas avec les mêmes ingrédients sous des noms différents. On a la possibilité de ne pas indiquer la composition mais c'est mieux envers la clientèle pour se prémunir des allergies ou des restrictions diététiques.

Produit – Preparation

Pour le "Pizzaiolo" on enregistre les recettes des "Produit" manufacturé comme les pizzas dans "Preparation". Chaque "Preparation est liée à un seul "Produit" et un "Produit" n'est pas obligé d'avoir une recette.

Produit - Produit

Cette association démontre le lien entre les "Produit" qui sont des ingrédients et les "Produit" qui sont des produits manufacturés comme les pizzas. Chaque pizza à plusieurs ingrédients et un ingrédient peut être dans plusieurs pizzas comme le fromage.

Les autres associations.

Magasin – Adresse

Un "Magasin" a forcément une seule "Adresse" et une "Adresse" n'est pas forcément celle d'un "Magasin".

Fournisseur - Adresse

Comme pour les "Magasin" un "Fournisseur" a forcément une seule "Adresse" et une "Adresse" n'est pas forcément celle d'un " Fournisseur ".

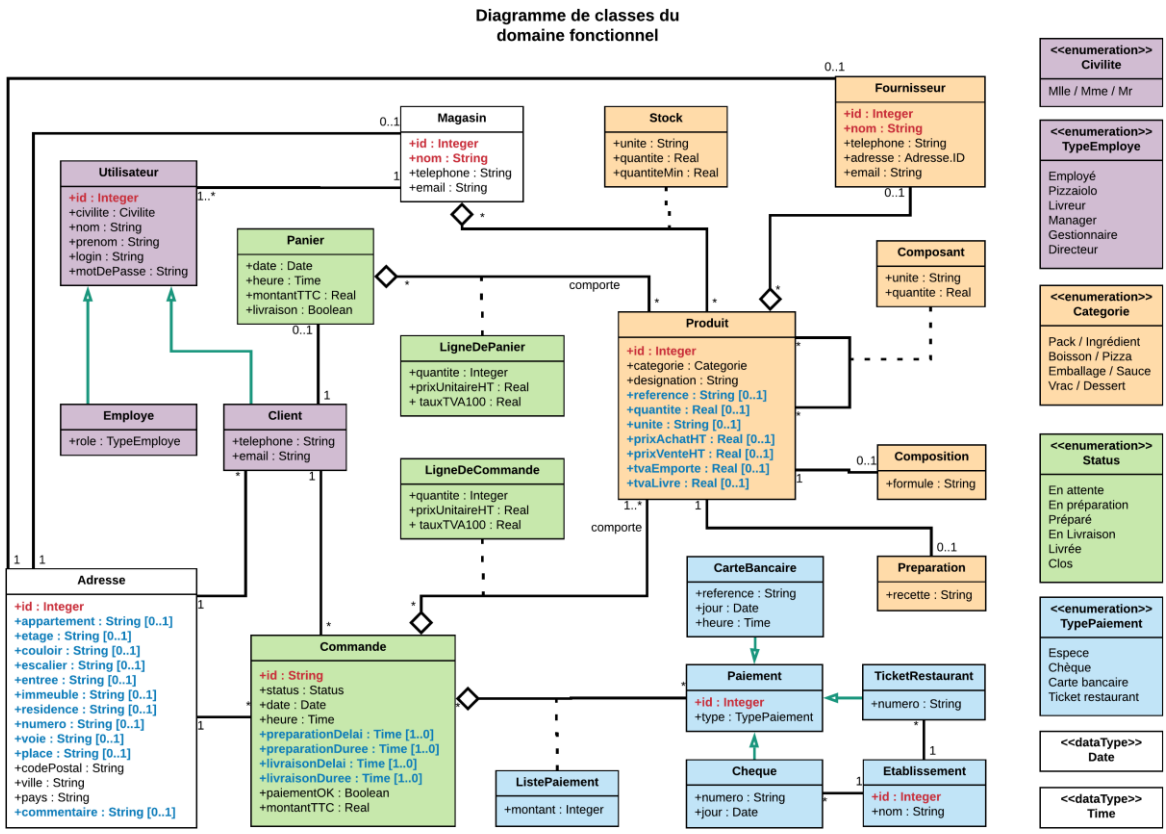
Cheque – Etablissement

Un "Cheque" provient d'un seul "Etablissement" et ce dernier peut émettre aucun ou plusieurs "Cheque".

TicketRestaurant - Etablissement

Un "TicketRestaurant" provient d'un seul "Etablissement" et ce dernier peut émettre aucun ou plusieurs " TicketRestaurant ".

Le diagramme de classes du domaine fonctionnel



Les héritages sont indiqués en vert et les agrégations comportent un losange blanc sur la classe est une agrégation de l'autre classe. Les attributs en rouge sont les identifiants qui doivent être uniques pour l'objet. Les attribut qui ne sont pas obligatoire avec une multiplicité [0..1] sont en bleu et les autres en noir. Sur le côté on a la liste des **<<enumeration>>** et des **<<dataType>>**. On va se baser sur le diagramme de classes du domaine fonctionnel pour élaborer le modèle physique de données MDP. On utilisera **SQL Power Architect** pour élaborer le MDP et générer un script pour générer notre base de données sous **MySQL**.

Le modèle physique de données MDP.

Les types de données.

On transcrit chaque classe et énumération en table de base de données. Les types Integer seront transcrit en :

Type	Bytes	Valeurs	Description
TINYINT UNSIGNED	1	0 à 254	Pour les identifiants avec peu de valeurs
INT UNSIGNED	4	0 à 4294967294	Pour les identifiants avec beaucoup de valeurs
DECIMAL (5,2)	4	-999,99 à 999,99	Pour les attributs monétaires et les quantités avec une précision de 5 chiffres avant la virgule et 2 après.

Les String en :

- VARCHAR(N) pour les chaînes de faible longueur ou N est la taille maximale.
- TEXT pour les grandes chaînes comme les commentaire, recette et formule.

Les <<dataType>> :

- Date en DATE.
- Time en TIME

Dénomination.

- AI (AUTO INCREMENT) : La valeur est automatiquement incrémentée si elle n'est pas spécifiée.
- PK (PRIMARY KEY) : Clé primaire servant d'identifiant aux tuples de la table.
- FK (FOREIGN KEY) : Clé étrangère servant à identifier des tuples d'une autre table.
- NN (NOT NULL) : La valeur ne peut pas être nulle comme pour les clés primaires.

La partie Utilisateur.

Utilisateur

Colonne	Type	Precision	AI	PK	FK	NN
id	**INT UNSIGNED**	**10**	**X**	**X**		**X**
civilite	ENUM ('Mlle','Mme','M')					X
nom	VARCHAR	50				X
prenom	VARCHAR	50				X
login	VARCHAR	50				X
mot_de_passe	VARCHAR	255				X
magasin_id	INT UNSIGNED	10			X	X

Contrainte d'unicité.

On ajoute une contrainte d'unicité "UNIQUE" sur le "login" pour ne pas avoir de problème de connexion.

Clé étrangère sur magasin_id

Table	Colonne	ON DELETE	ON UPDATE
civilite	id	NO ACTION	NO ACTION

Client

Colonne	Type	Precision	AI	PK	FK	NN
utilisateur_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
telephone	VARCHAR	10				X
adresse_id	INT UNSIGNED	10			X	X
email	VARCHAR	255				X

Contrainte d'unicité.

On ajoute une contrainte d'unicité "UNIQUE" sur "email" pour permettre la récupération de mot de passe par email en.

Clé étrangère sur utilisateur_id

Table	Colonne	ON DELETE	ON UPDATE
utilisateur	id	CASCADE	CASCADE

Si on met à jour ou supprime un "utilisateur", "client" sera modifié ou supprimé.

Clé étrangère sur adresse_id

Table	Colonne	ON DELETE	ON UPDATE

adresse	id	NO ACTION	NO ACTION

Employé

Colonne	Type	Precision	AI	PK	FK	NN
utilisateur_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
role	ENUM ('Accueil', 'Pizzaiolo', 'Livreur', 'Manager', 'Gestionnaire', 'Comptable', 'Direction')					X

Clé étrangère sur utilisateur_id

Table	Colonne	ON DELETE	ON UPDATE
utilisateur	id	CASCADE	CASCADE

Si on met à jour ou supprime un "utilisateur", "employe" sera modifié ou supprimé.

La partie Produit.

Produit

Colonne	Type	Precision	AI	PK	FK	NN
id	**INT UNSIGNED**	**10**	**X**	**X**		**X**
designation	VARCHAR	50				X
categorie	ENUM ('pack', 'vrac', 'ingrédient', 'pizza', 'boisson', 'dessert', 'emballage', 'sauce')					X
fournisseur_id	INT UNSIGNED	10			X	
reference	VARCHAR	20				
quantite	DECIMAL	(5,2)				
unite	VARCHAR	3				
prix_achat_ht	DECIMAL	(5,2)				
prix_vente_ht	DECIMAL	(5,2)				
tva_emporte	DECIMAL	(3,1)				
tva_livre	DECIMAL	(3,1)				

Contrainte d'unicité.

On ajoute une contrainte d'unicité "UNIQUE" sur le "designation" pour ne pas avoir deux fois le même produit.

Clé étrangère sur fournisseur_id

Table	Colonne	ON DELETE	ON UPDATE
fournisseur	id	NO ACTION	NO ACTION

Composition

Colonne	Type	Precision	AI	PK	FK	NN
produit_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
formule	TEXT					X

Clé étrangère sur produit_id

Table	Colonne	ON DELETE	ON UPDATE
produit	id	CASCADE	CASCADE

Si on met à jour ou supprime un "produit", "composition" sera modifié ou supprimé.

Préparation

Colonne	Type	Precision	AI	PK	FK	NN

produit_id	INT UNSIGNED	10		X	X	X
recette	TEXT					X

Clé étrangère sur produit_id

Table	Colonne	ON DELETE	ON UPDATE
produit	id	CASCADE	CASCADE

Si on met à jour ou supprime un "produit", "preparation" sera modifié ou supprimé.

Stock

Colonne	Type	Precision	AI	PK	FK	NN
magasin_id	INT UNSIGNED	10		X	X	X
produit_id	INT UNSIGNED	10		X	X	X
quantite	DECIMAL	(5,2)				X
quantite_min	DECIMAL	(5,2)				X
unite	VARCHAR	3				X

Clé étrangère sur produit_id

Table	Colonne	ON DELETE	ON UPDATE
produit	id	NO ACTION	NO ACTION

Clé étrangère sur magasin_id

Table	Colonne	ON DELETE	ON UPDATE
magasin	id	NO ACTION	NO ACTION

Composant

Colonne	Type	Precision	AI	PK	FK	X
produit_id	INT UNSIGNED	10		X	X	X
ingredient_id	INT UNSIGNED	10		X	X	X
quantite	DECIMAL	(5,2)				X
unite	VARCHAR	3				X

Clé étrangère sur produit_id

Table	Colonne	ON DELETE	ON UPDATE
produit	id	NO ACTION	NO ACTION

Clé étrangère sur ingredient_id

Table	Colonne	ON DELETE	ON UPDATE
produit	id	NO ACTION	NO ACTION

La partie Paiement.

Paiement

Colonne	Type	Precision	AI	PK	FK	NN
paiement_id	**INT UNSIGNED**	**10**	**X**	**X**		**X**
type	ENUM ('espèce', 'carte bancaire', 'ticket restaurant', 'chèque bancaire', 'sans')					X

Carte bancaire

Colonne	Type	Precision	AI	PK	FK	NN
paiement_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
reference	VARCHAR	100				X
jour	DATE					X
heure	TIME					X

Clé étrangère sur paiement_id

Table	Colonne	ON DELETE	ON UPDATE
paiement	id	CASCADE	CASCADE

Si on met à jour ou supprime un "paiement", "carte_bancaire" sera modifié ou supprimé.

Chèque

Colonne	Type	Precision	AI	PK	FK	NN
paiement_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
banque	TINYINT UNSIGNED	3			X	X
numero	VARCHAR	100				X
jour	DATE					X

Clé étrangère sur paiement_id

Table	Colonne	ON DELETE	ON UPDATE
paiement	id	CASCADE	CASCADE

Si on met à jour ou supprime un "paiement", "cheque" sera modifié ou supprimé.

Ticket restaurant

Colonne	Type	Precision	AI	PK	FK	NN
paiement_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**

numero	VARCHAR	50				X
etablissement_id	TINYINT UNSIGNED	3			X	X

Clé étrangère sur paiement_id

Table	Colonne	ON DELETE	ON UPDATE
paiement	id	CASCADE	CASCADE

Si on met à jour ou supprime un "paiement", "ticket_restaurant" sera modifié ou supprimé.

Établissement

Colonne	Type	Precision	AI	PK	FK	NN
id	**TINYINT UNSIGNED**	**3**		**X**		**X**
nom	VARCHAR	20				**X**

Liste paiement

Colonne	Type	Precision	AI	PK	FK	NN
commande_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
paiement_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
montant	DECIMAL	(5,2)				**X**

On doit garder une trace des paiements donc on ne peut pas les modifier ou supprimer en cas de modification ou suppression dans les tables "paiement" et "commande".

Clé étrangère sur paiement_id

Table	Colonne	ON DELETE	ON UPDATE
paiement	id	NO ACTION	NO ACTION

Clé étrangère sur commande_id

Table	Colonne	ON DELETE	ON UPDATE
commande	id	NO ACTION	NO ACTION

La partie Commande.

Panier

Colonne	Type	Precision	AI	PK	FK	NN
utilisateur_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
jour	DATE					X
heure	TIME					X
montant_ttc	DECIMAL	(5,2)				X
livraison	TINYINT UNSIGNED					X

Clé étrangère sur utilisateur_id

Table	Colonne	ON DELETE	ON UPDATE
client	id	NO ACTION	NO ACTION

Commande

Colonne	Type	Precision	AI	PK	FK	NN
id	**INT UNSIGNED**	**10**	**X**	**X**		**X**
utilisateur_id	**INT UNSIGNED**	**10**			**X**	**X**
adresse_id	**INT UNSIGNED**	**10**				**X**
status	ENUM ('En attente', 'En préparation', 'Préparée', 'En livraison', 'Livrée', 'Clos')					X
jour	DATE					X
heure	TIME					X
montant_ttc	DECIMAL	(5,2)				X
preparation_delai	TIME					
preparation_duree	TIME					
livraison_delai	TIME					
livraison_duree	TIME					
paiement_ok	TINYINT	1				X

Clé étrangère sur adresse_id

Table	Colonne	ON DELETE	ON UPDATE
adresse	id	NO ACTION	NO ACTION

Clé étrangère sur utilisateur_id

Table	Colonne	ON DELETE	ON UPDATE
utilisateur	id	NO ACTION	NO ACTION

Ligne de panier

Colonne	Type	Precision	AI	PK	FK	NN

utilisateur_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
produit_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
quantite	DECIMAL	(2,0)				X
prix_unitaire_ht	DECIMAL	(5,2)				X
taux_tva	DECIMAL	(3,1)				X

Clé étrangère sur utilisateur_id

Table	Colonne	ON DELETE	ON UPDATE
panier	id	CASCADE	CASCADE

Si le "panier" est supprimé on supprime les tuples correspondant dans la table "ligne_de_panier".

Clé étrangère sur produit_id

Table	Colonne	ON DELETE	ON UPDATE
produit	id	NO ACTION	NO ACTION

Ligne de commande

Colonne	Type	Precision	AI	PK	FK	NN
commande_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
produit_id	**INT UNSIGNED**	**10**		**X**	**X**	**X**
quantite	DECIMAL	(2,0)				X
prix_unitaire_ht	DECIMAL	(5,2)				X
taux_tva	DECIMAL	(3,1)				X

Clé étrangère sur commande_id

Table	Colonne	ON DELETE	ON UPDATE
commande	id	CASCADE	CASCADE

Si la "commande" est supprimée on supprime les tuples correspondant dans la table "ligne_de_commande".

Clé étrangère sur produit_id

Table	Colonne	ON DELETE	ON UPDATE
produit	id	NO ACTION	NO ACTION

Les autres tables.

Adresse

Colonne	Type	Precision	AI	PK	FK	NN
id	**INT UNSIGNED**	**10**	**X**	**X**		**X**
appartement	VARCHAR	4				
etage	VARCHAR	3				
couloir	VARCHAR	3				
escalier	VARCHAR	3				
entree	VARCHAR	3				
immeuble	VARCHAR	10				
residence	VARCHAR	20				
numero	VARCHAR	5				
voie	VARCHAR	50				
place	VARCHAR	50				
code	VARCHAR	5				X
ville	VARCHAR	20				X
pays	VARCHAR	20				X
commentaire	TEXT					

Contrainte d'unicité.

Pour ne pas avoir deux fois la même adresse on implémentera des TRIGGER BEFORE INSERT et BEFORE UPDATE.

Magasin

Colonne	Type	Precision	AI	PK	FK	NN
id	**INT UNSIGNED**	**10**	**X**	**X**		**X**
nom	VARCHAR	50		X		X
telephone	VARCHAR	10				X
email	VARCHAR	255				X
adresse_id	INT UNSIGNED	10			X	X

Clé primaire sur "nom"

Définir une clé primaire sur le "nom" permet d'avoir des noms uniques pour les magasins et d'accélérer les recherches sur le "nom".

Clé étrangère sur "adresse_id"

Table	Colonne	ON DELETE	ON UPDATE
adresse	id	NO ACTION	NO ACTION

Fournisseur

Colonne	Type	Precision	AI	PK	FK	NN
id	**INT UNSIGNED**	**10**	**X**	**X**		**X**
nom	VARCHAR	50		X		X
telephone	VARCHAR	10				X
email	VARCHAR	255				X
adresse_id	INT UNSIGNED	10			X	X

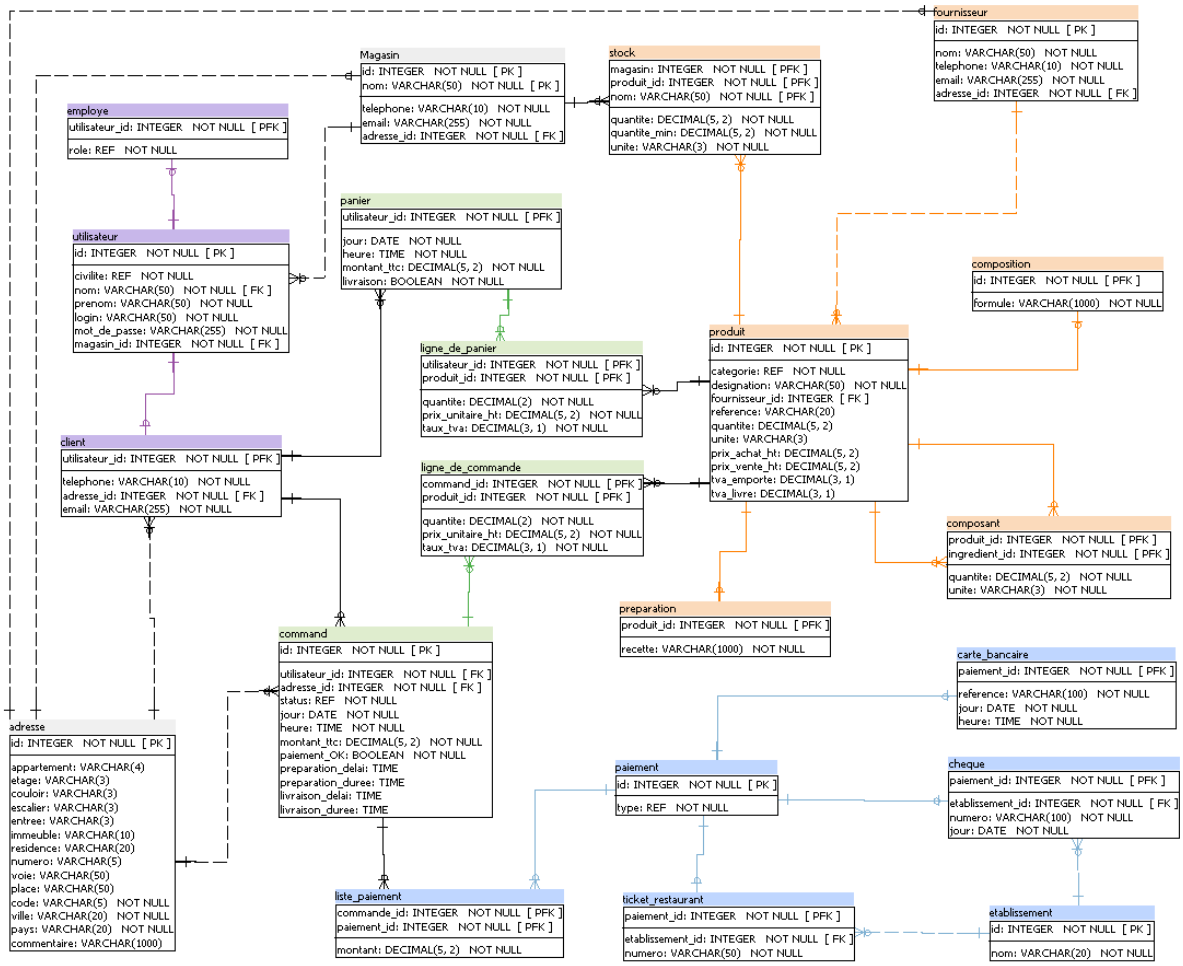
Clé primaire sur "nom"

Définir une clé primaire sur le "nom" permet d'avoir des noms uniques pour les fournisseurs et d'accélérer les recherches sur le "nom".

Clé étrangère sur "adresse_id"

Table	Colonne	ON DELETE	ON UPDATE
adresse	id	NO ACTION	NO ACTION

Diagramme du Modèle Physique de Données.



L'outil de création du modèle physique de données ne contient pas tous les types de MySQL.

- Les VACHAR(1000) sont remplacés par des type TEXT.
- Les BOOLEAN sont remplacés par des TINYINT.

Une table a été ajoutée dans la base pour afficher les messages d'erreur. Elle n'apparait pas dans le diagramme.

Étude de déploiement.

L'étude du déploiement de l'application **OC Pizza** est basée sur **AWS (Amazon Web Services)**.

La partie utilisateur.

Les utilisateurs doivent pouvoir se connecter au site web **OC Pizza** à partir de n'importe quel navigateur web sur un ordinateur, une tablette ou un smartphone. On doit avoir un site responsive pour s'adapter aux trois résolutions différentes de ces médias.

Pour une meilleure intégration avec les smartphones on développera des applicatifs spécifiques. Surtout pour le livreur qui devra valider la livraison lors de son déplacement chez le client.

- **APK** pour **Android**,
- **APP** pour **Apple**.

La partie base de données.

La base de données sera hébergée par **AWS** avec **Amazon RDS (Relational Database Service)**. Cela permet de gérer facilement la base de données relationnelle dans le cloud avec une grande souplesse d'évolution. On prendra le mode **Multi-AZ** pour avoir une copie de notre base de données, dans une autre région, synchronisée qui pourra prendre le relai en cas de maintenance ou de défaillance de la base principale.

Les services annexes.

Le DNS.

Pour permettre une meilleure connexion des utilisateurs à notre application, on utilise **Amazon Route 53** qui sert de **Domain Name Server**. Il converti les adresses nominatives en adresse **IP**. En plus de rendre le site web toujours visible par tout le monde, on a la possibilité de modifier à la volée l'adresse **IP** de nos services en toute transparence pour l'utilisateur, c'est **Route 53** qui assure la liaison avec la bonne adresse. **Route 53** dispatche les demandes soit vers **CloudFront** soit vers l'**API Gateway**.

La zone de mémoire cache.

Amazon CloudFront est un service web qui accélère la distribution de vos contenus web statiques et dynamiques, tels que des fichiers **.html**, **.css**, **.js**, **multimédias** et **image**, à vos utilisateurs. **CloudFront** diffuse votre contenu à travers un réseau mondial de centres de données appelés emplacements périphériques. Lorsqu'un utilisateur demande le contenu que vous proposez avec **CloudFront**, il est dirigé vers l'emplacement périphérique qui fournit la latence la plus faible et, par conséquent, le contenu est remis avec les meilleures performances possibles.

Si le contenu se trouve déjà dans l'emplacement périphérique avec la plus faible latence, **CloudFront** le remet immédiatement.

Si le contenu ne se trouve pas à cet emplacement périphérique, **CloudFront** va le chercher dans un **Bucket** comme **Amazon S3** ou le demande à un serveur **HTTPS** que l'on a identifié comme étant la source originale du contenu de notre application.

Le stockage des données publiques.

On utilise un simple **Bucket** comme **Amazon Simple Storage Service** pour stocker tous les fichiers de notre site Web qui peuvent être publiques. **Amazon S3** offre une interface simple de services Web qui permet de stocker et de récupérer n'importe quelle quantité de données, à tout moment, de n'importe où sur le Web. Il permet aux développeurs d'accéder à la même infrastructure de stockage de données hautement évolutive, fiable, rapide, peu coûteuse qu'**Amazon** utilise pour faire fonctionner son propre réseau mondial de sites. Ce service vise à maximiser les avantages d'échelle et à en faire bénéficier les développeurs.

Gestion de la sécurité.

Pour exposer les **APIs REST** des **microservices** on utilise **Amazon API Gateway**. **Amazon API Gateway** est un service qui permet de créer, de publier, de maintenir, de surveiller et de sécuriser les **API REST** à n'importe quelle échelle. Les développeurs **d'API** peuvent créer des **API** qui accèdent à **AWS** ou à d'autres services web, ainsi qu'aux données stockées dans le cloud **AWS**. En tant que développeur **d'API API Gateway**, on peut créer des **API** en vue de les utiliser dans nos applications client ou celles de développeurs d'applications tiers.

On couple **l'API Gateway** avec le service **Amazon Identity Access Management** pour garantir la sécurité de notre site web. **IAM** contrôle l'accès aux ressources **AWS**. On utilise **IAM** pour contrôler les personnes qui s'authentifient et sont autorisées à utiliser les ressources.

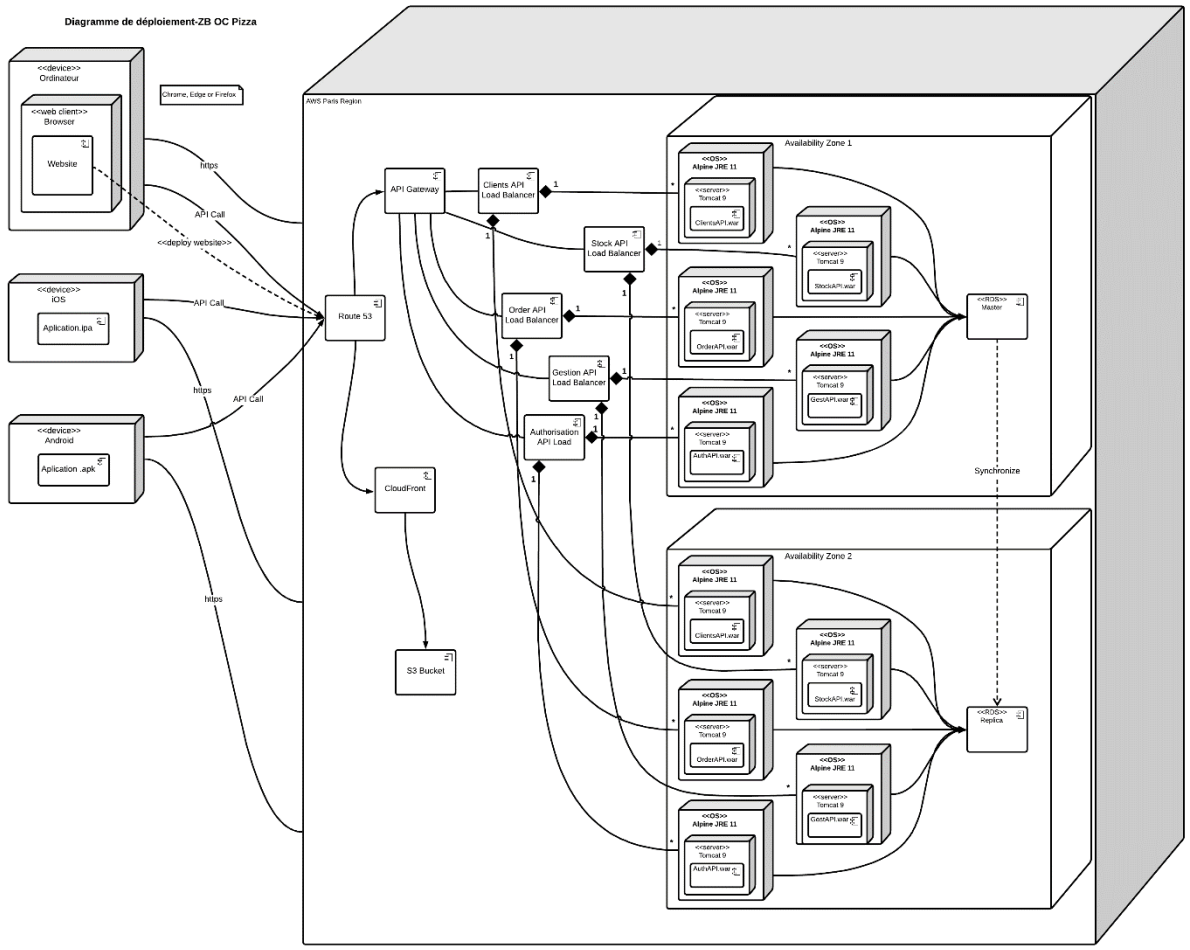
Les microservices.

On a découpé l'application en **Microservice** pour une plus grande simplicité et une facilité à scalabilité pour répondre à l'accroissement de la demande des utilisateurs. Les **Microservices** sont des **API REST** qui sont exposée par **l'API Gateway**. Les **Microservices** sont gérés chacun par un **Load Balancer** qui permet de dupliquer l'**API REST**.

Les APIs sont installées sur des instances **Amazon EC2 (Elastic Compute Cloud)**. **Amazon EC2** offre une capacité de calcul évolutive dans le cloud **AWS**. L'utilisation d'**EC2** dispense d'investir à l'avance dans du matériel et, par conséquent, on peut développer et déployer les applications plus rapidement. On peut utiliser **Amazon EC2** pour lancer autant de serveurs virtuels que nécessaire, configurer la sécurité et la mise en réseau, et gérer le stockage. Il permet également de monter ou descendre en puissance rapidement, avec les **Load Balancer**, afin de gérer l'évolution des exigences ou des pics de popularité, ce qui permet de réduire la nécessité de prévoir le trafic du serveur.

On utilise **Alpine** une version légère de **Linux** avec le runtime **Java JRE11** pour faire fonctionner nos **API** qui sont déployées sur un serveur **Tomcat**.

Diagramme de déploiement.



Les composants.

La partie client.

Navigateur web.

Le client doit accéder au site web avec la plupart des navigateurs depuis :

- Un ordinateur sous **Microsoft Windows 10**
- Un ordinateur sous **Apple OS X 10.14.6**
- Une tablette sous **Apple iOS 11**
- Une tablette sous **Android Nougat 7.1**
- Un smartphone sous **Apple iOS 11**
- Un smartphone sous **Android Nougat 7.1**

L'application doit être responsive pour s'adapter aux différents écrans. La communication se fait en utilisant le protocole sécurisé **HTTPS** sur internet.

Applicatif APP/APK.

Une application simplifiée sur les deux principaux **OS** des smartphones permet de mieux répondre aux attentes des clients.

- Une **APP** pour les smartphones sous **Apple iOS 11**
- Une **APK** pour les smartphones sous **Android Nougat 7.1**

Il faut faire aussi une application **APK** dédié pour les livreurs sur **Android Nougat 7.1** pour valider la réception des commandes et le paiement.

Les services Amazon.

On utilise le service **Route 53** pour orienter les requêtes **HTTPS** de l'utilisateur vers le bon service sans qu'il ne se soucie de l'adresse **IP**. Le service **CloudFront** agit comme un cache qui va chercher directement les données disponibles qui sont directement accessibles. L'**API Gateway** oriente les requêtes vers le bon **microservice** et gère la sécurité avec **IAM**.

La partie publique.

Les fichiers **HTML**, **CSS**, **JS** et images qui n'ont pas besoin d'être sécurisé sont stocké dans un bucket **S3 (Simple Storage Service)**. L'accès se fait via une connexion sécurisée **HTTPS**.

La partie sécurisée.

La base de données.

Le Système de Gestion de Base de Données **MySQL 8.0.16** sera déployé sur une instance de serveur **RDS (Relational Database Service)** sous **UNIX**. La communication se fait en **HTTPS** via le port **3306**. Une autre instance de base de données est synchronisée avec la première pour servir de sauvegarde ou de remplacement en cas de maintenance sur la base master. Cela est inclus dans l'option **Multi A-Z** du produit **Amazon RDS**.

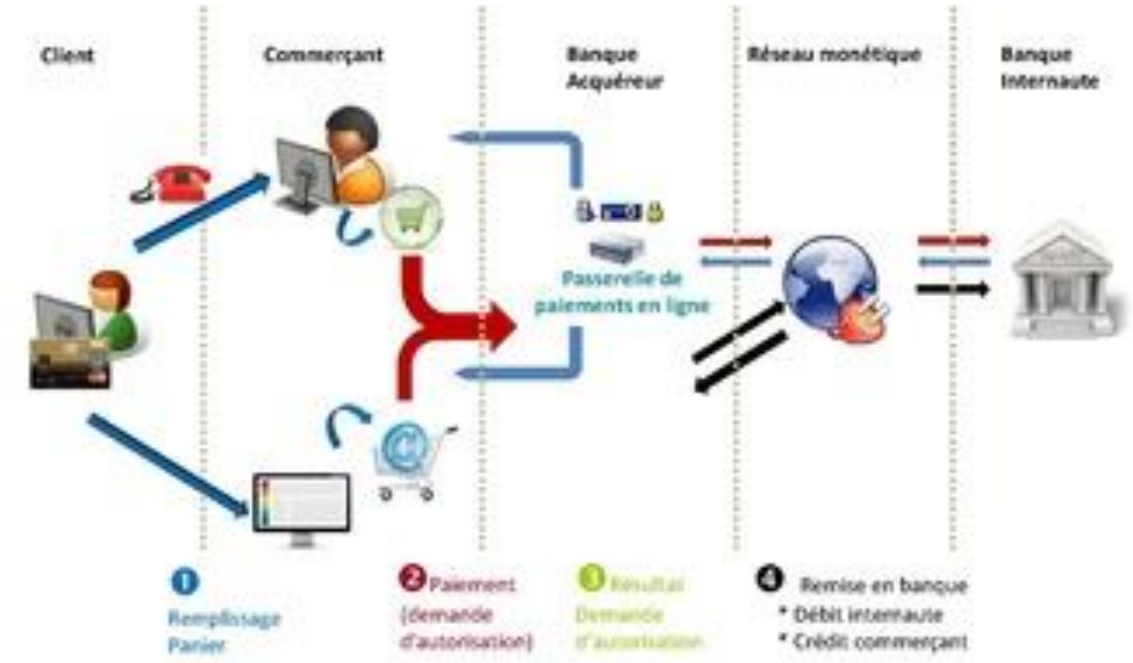
Les Microservices

Ils seront implémentés en **Java** avec **SPRING** et packagé en **WAR** avec **MAVEN** pour faciliter la mise à jour et le déploiement. L'archive **WAR** sera déployée sur un serveur **Tomcat 9.0.24** qui tourne sur une instance **Amazon EC2 (Elastic Cloud Computing)** sous **Alpine** une version légère de **Linux** avec **Java JRE 11**. L'accès se fait par requête **REST** via **HTTPS** via le port **8080**. Les **Microservices** sont des **API REST** exposées par l'**API Gateway**. Les **Microservices** sont les seuls à pouvoir accéder à la base de données Master.

La partie externe.

Le système bancaire.

La banque doit nous fournir l'interface avec son système pour pouvoir effectuer les paiements via notre application. La communication se fera par le protocole sécurisé **HTTPS** et en utilisant un cryptage des données par clé publique et privée.



Le diagramme de composant.

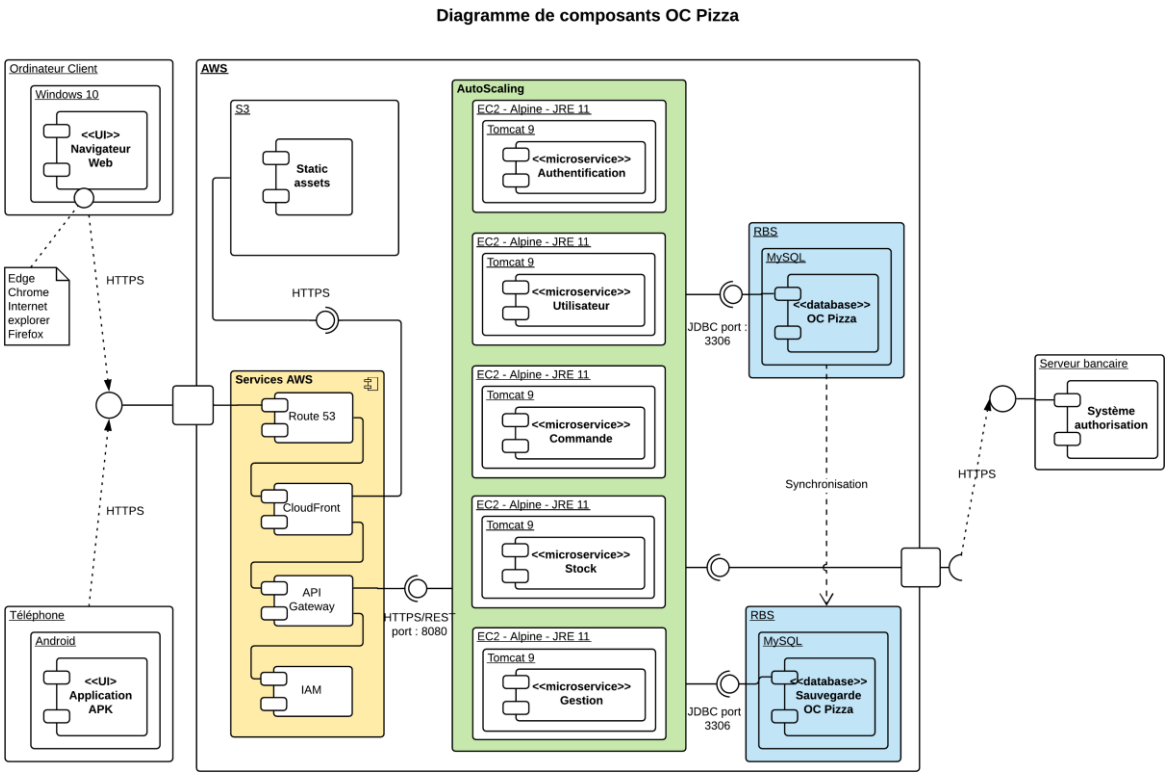


Table des matières.

Introduction. 2
Le domaine fonctionnel. 3
Les composants généraux. 3
Adresse 3
Magasin 4
Les composants de la partie utilisateur. 5
Utilisateur 5
Civilite <<enum>> 5
Client 6
Employe 6
TypeEmploye <<enum>> 6
Apperçu de la partie Utilisateur 7
Les composants de la partie produit. 8
Produit 8
Categorie <<enum>> 8
Fournisseur 9
Composition 9
Preparation 10
Composant 10
Stock 10
Apperçu de la partie Produit 11
Les composants de la partie paiement. 12
Paiement 12
TypePaiement <<enum>> 12
CarteBancaire 12
Date <<dataType>> 13
Time <<dataType>> 13
TicketRestaurant 13
Cheque 13

Etablissement....13
ListePaiement....14
Apperçu de la partie Paiement....14
Les composants de la partie commande.....15
Panier....15
Commande....15
Statut <<enum>>....16
LigneDePanier....16
LigneDeCommande....17
Apperçu de la partie Commande....17
Les relations entre les composants....18
Les associations avec Utilisateur.....18
Utilisateur – Magasin....18
Les associations avec Client.....18
Client – Adresse....18
Client – Panier....18
Client – Commande....18
Les associations avec Panier.....18
Panier – Client....18
Panier – Produit....19
Les associations avec Commande.....20
Commande-Client....20
Commande – Produit....20
Commande – Adresse....20
Commande – Paiement....20
Les associations avec Produit.....22
Produit – Panier....22
Produit – Commande....22
Produit – Magasin....22
Produit - Fournisseur....22
Produit – Composition....22
Produit – Preparation....23

Produit - Produit 23
Les autres associations. 24
Magasin – Adresse 24
Fournisseur - Adresse 24
Cheque – Etablissement 24
TicketRestaurant - Etablissement 24
Le diagramme de classes du domaine fonctionnel 25
Le modèle physique de données MDP. 26
Les types de données. 26
Dénomination. 26
La partie Utilisateur. 27
Utilisateur 27
Client 27
Employé 28
La partie Produit 29
Produit 29
Composition 29
Préparation 29
Stock 30
Composant 30
La partie Paiement. 32
Paiement 32
Carte bancaire 32
Chèque 32
Ticket restaurant 32
Établissement 33
Liste paiement 33
La partie Commande. 34
Panier 34
Commande 34
Ligne de panier 34
Ligne de commande 35

Les autres tables. 36
Adresse 36
Magasin 36
Fournisseur 37
Diagramme du Modèle Physique de Données. 38
Étude de déploiement. 39
La partie utilisateur. 39
La partie base de données. 39
Les services annexes. 40
Le DNS. 40
La zone de mémoire cache. 40
Le stockage des données publiques. 40
Gestion de la sécurité. 41
Les microservices. 42
Diagramme de déploiement. 43
Les composants. 44
La partie client. 44
Navigateur web. 44
Applicatif APP/APK. 44
Les services Amazon. 45
La partie publique. 45
La partie sécurisée. 45
La base de données. 45
Les Microservices 45
La partie externe. 46
Le système bancaire. 46
Le diagramme de composant. 47
Table des matières. 48