Projet OC PIZZA :
Création d'un système informatique de gestion et de vente en ligne.

Dossier de conception technique

MC info, 7 rue Plop, 77777 JESAISPAS

Suivi des versions

1 Historique du document

VERSION	DATE	REDACTEUR	MODIFICATION

2 Diffusion

DESTINATAIRE	DATE	POUR INFORMATION	POUR VALIDATION

3 Validation

REDACTEUR	CONTROLEUR	APPROBATEUR	CLIENT

Sommaire

Présentation du document

4 Objet du document

Le présent document constitue le dossier de conception technique du projet de création d'un système informatique de gestion et de vente en ligne de l'entreprise OC Pizza.

5 Contexte actuel

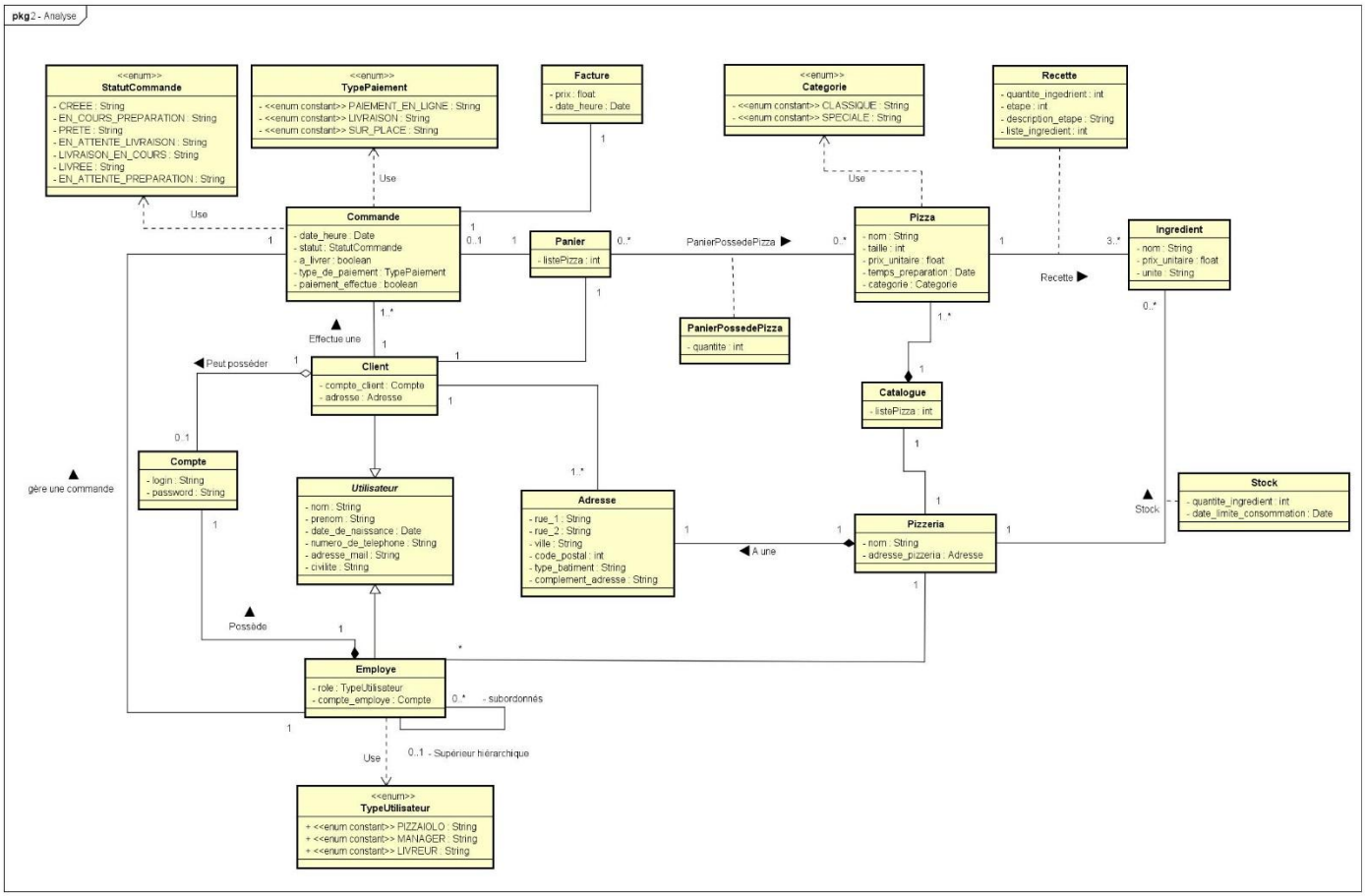
« OC Pizza » est un jeune groupe de pizzeria spécialisé dans les pizzas livrées ou à emporter.
A l'heure actuelle, le groupe compte déjà 5 points de vente mais ne possède pas de système informatique et ne réalise pas de vente en ligne.

Présentation du dossier de conception

6 Domaine fonctionnel

Le domaine fonctionnel du système OC Pizza établit l'ensemble des classes qui serviront de support au développement de l'application, à la création du Modèle Physique de Données.

6.1 Diagramme de classe UML



6.2 Description des classes

6.2.1 Classe « Utilisateur »

Cette classe regroupe les attributs communs à tous les « Utilisateurs » de l'application.

Cette classe mère est associée aux classes filles **Client** et **Employe.**

Attribut	Description
nom	Nom de l'utilisateur
prenom	Prénom de l'utilisateur
date_de_naissance	Date de naissance de l'utilisateur
numero_de_telephone	Numéro de téléphone de l'utilisateur
adresse_mail	Adresse mail de l'utilisateur
civilite	Civilité de l'utilisateur

6.2.2 Classe « Client »

Cette classe regroupe les attributs communs à tous les « Client » de l'application.

Cette classe fille est associée à la classe mère **Utilisateur** dont elle hérite les attributs.

La classe est associée à la classe « Compte ». Chaque Client peut posséder 0 ou 1 « Compte » pour réaliser ses commandes. La classe est associée à la classe « Adresse » binaire (1, 1..*). Chaque client possède 1 ou plusieurs adresses de livraison.

Attribut	Description
compte_client	Un objet compte qui peut être associé au client
adresse	Un objet adresse qui est associé au client

6.2.3 Classe « Employe »

Cette classe regroupe les attributs communs à tous les « Employe » de l'application.

Cette classe fille est associée à la classe mère **Utilisateur** dont elle hérite les attributs.

La classe est associée à la classe « Compte ». Chaque Employé doit posséder 1 « Compte » pour réaliser les commandes. Cette classe possède une association binaire à elle-même car un employé « manager » peut-être associé à 0 ou une infinité d'employé « subordonnés ».

Attribut	Description
role	Rôle de l'utilisateur. Cet attribut est obligatoire.
Compte_employe	Le compte de l'employé. Cet attribut est obligatoire

6.2.4 Classe « Compte »

Cette classe regroupe les attributs « login » et « password » de chaque compte.

Cette classe est associée aux classes filles **Client** et **Employe.** Un compte ne peut appartenir qu'à un employé et un employé doit avoir un compte obligatoirement. En revanche, un compte ne peut appartenir qu'à un client mais il n'est pas indispensable que le client est un compte pour réaliser des commandes.

Attribut	Description
login	Login du client ou de l'employé
password	Le password du client ou de l'employé

6.2.5 Classe « TypeUtilisateur »

Cette classe de type ENUM regroupe les attributs correspondant aux rôles des « Employe ».

Cette classe est utilisée par la classe **Employe.** Chaque attribut permet de définir les droits associés au type d'utilisateur.

Attribut	Description
PIZZAIOLO	Correspond au type d'utilisateur PIZZAIOLO
MANAGER	Correspond au type d'utilisateur MANAGER
LIVREUR	Correspond au type d'utilisateur LIVREUR

6.2.6 Classe « Commande »

Cette classe regroupe les attributs communs à toutes les « Commande » de l'application.

Cette classe est associée aux classes **Facture**, **Panier**, **Client** et **Employe** par des associations binaires. Un employé ne gère qu'une commande à la fois et un client peut avoir une ou plusieurs commandes.

La commande peut-être associé à un seul panier et mais un panier n'est pas obligatoirement associé à une commande. Chaque commande est associée à une facture. La classe **Commande** utilise les classes **StatuCommande** et **TypePaiement**.

Attribut	Description
date_heure	La date et l'heure de la commande
statut	Le statut de la commande
a_livrer	Un booléen définissant si la commande doit être livrée
type_de_paiement	Le type de paiement choisi
paiement_effectue	Un booléen définissant si le paiement a été fait

6.2.7 Classe « StatutCommande »

Cette classe de type ENUM regroupe les attributs correspondant aux statuts de la commande.

Cette classe est utilisée par la classe **Commande.** Chaque attribut permet de définir le statut de la commande.

Attribut	Description
CREEE	Lorsque la commande est créée
EN_ATTENTE_PREPARATION	Lorsque la commande est en attente de préparation
EN_COURS_PREPARATION	Lorsque la commande est en cours de préparation
PRETE	Lorsque la commande est prête
EN_ATTENTE_LIVRAISON	Lorsque la commande est en attente de livraison
LIVRAISON_EN_COURS	Lorsque la commande est en cours de livraison
LIVREE	Lorsque la commande est livrée

6.2.8 Classe « TypePaiement »

Cette classe de type ENUM regroupe les attributs correspondant aux types de paiement disponible pour chaque commande.

Cette classe est utilisée par la classe **Commande.** Chaque attribut permet de définir le type de paiement de la commande.

Attribut	Description
PAIEMENT_EN_LIGNE	Lorsque la commande est payée en ligne
LIVRAISON	Lorsque la commande est payée à la livraison
SUR_PLACE	Lorsque la commande est payée sur place

6.2.9 Classe « Facture »

Cette classe regroupe les attributs communs à toutes les factures de l'application.

Cette classe est associée à la classe **Commande** par une association binaire un pour un. Une facture correspond à une commande et inversement.

Attribut	Description
prix	Le prix total de la commande
date_heure	La date et l'heure de la facture

6.2.10 Classe « Panier »

Cette classe est associée à la classe **Commande** et par la classe d'association **PanierPossedePizza** à la classe **Pizza**. Cette classe possède l'attribut « listePizza » représentant la liste des pizzas contenues dans le panier. Un panier est associé à zéro ou une commande. Un panier peut-être associé à zéro ou une infinité de pizza et inversement.

Attribut	Description
ListePizza	La liste des pizzas contenues dans le panier.

6.2.11 Classe « PanierPossedePizza »

Cette classe est une classe d'association entre la classe **Panier** et **Pizza**. Cette classe possède l'attribut « quantité » représentant la quantité de pizza ajoutée au panier.

Attribut	Description
quantité	La quantité de pizza ajoutée au panier.

6.2.12 Classe « Pizza »

Cette classe regroupe les attributs communs à toutes les pizzas.

Cette classe est associée aux classes **Panier, Catalogue** et **Ingrédient.** Elle utilise la classe **Catégorie**. Chaque pizza est associée à ses ingrédients par la classe d'association **Recette**. Pour faire une pizza il faut au moins trois ingrédients. La classe Pizza est reliée à la classe Catalogue par une association de composition car le catalogue est composé d'une à plusieurs pizzas.

Attribut	Description
nom	Le nom de la pizza
taille	La taille de la pizza
prix_unitaire	Le prix de la pizza
temps_preparation	Le temps de préparation de la pizza
categorie	La catégorie de la pizza

6.2.13 Classe « Categorie »

Cette classe de type ENUM regroupe les attributs correspondant aux catégories de pizza disponible.

Cette classe est utilisée par la classe **Pizza.** Chaque attribut permet de définir la catégorie de la pizza.

Attribut	Description
CLASSIQUE	Catégorie CLASSIQUE
SPECIALE	Catégorie SPECIALE

6.2.14 Classe « Catalogue »

Cette classe est associée aux classes **Pizza** et **Pizzeria.** Un catalogue est composé de pizzas d'où la relation de composition entre les classes correspondantes. Un catalogue est possédé par une pizzeria et une pizzeria possède un unique catalogue.

Attribut	Description
listePizza	La liste des pizzas du catalogue

6.2.15 Classe « Ingredient »

Cette classe regroupe les attributs communs à tous les ingrédients.

Cette classe mère est associée aux classes filles **Client** et **Employe.**

Attribut	Description
nom	Nom de l'ingrédient
prix_unitaire	Le prix unitaire d'un ingrédient
unite	L'unité de conditionnement de l'ingrédient

6.2.16 Classe « Recette »

Cette classe est une classe d'association entre la classe **Pizza** et **Ingredient**. Une recette est associée à une pizza et au moins trois ingrédients.

Attribut	Description
quantite_ingredient	La quantité d'un ingrédient
etape	Le numéro d'étape de la recette
description_etape	La description de l'étape de la recette
liste_ingredient	La liste des ingrédients nécessaire à la recette

6.2.17 Classe « Pizzeria »

Cette classe regroupe les attributs communs à toutes les pizzerias.

Cette classe est associée aux classes **Employe**, **Ingredient**, **Catalogue** et **Adresse.** La pizzeria possède obligatoirement une adresse. Elle possède un catalogue. La pizzeria est associée à une infinité d'employé. La classe **Pizzeria** est associée à la classe **Ingredient** par une classe d'association **Stock.**

Attribut	Description
nom	Nom de la pizzeria
adresse_pizzeria	L'adresse de la pizzeria

6.2.18 Classe « Stock »

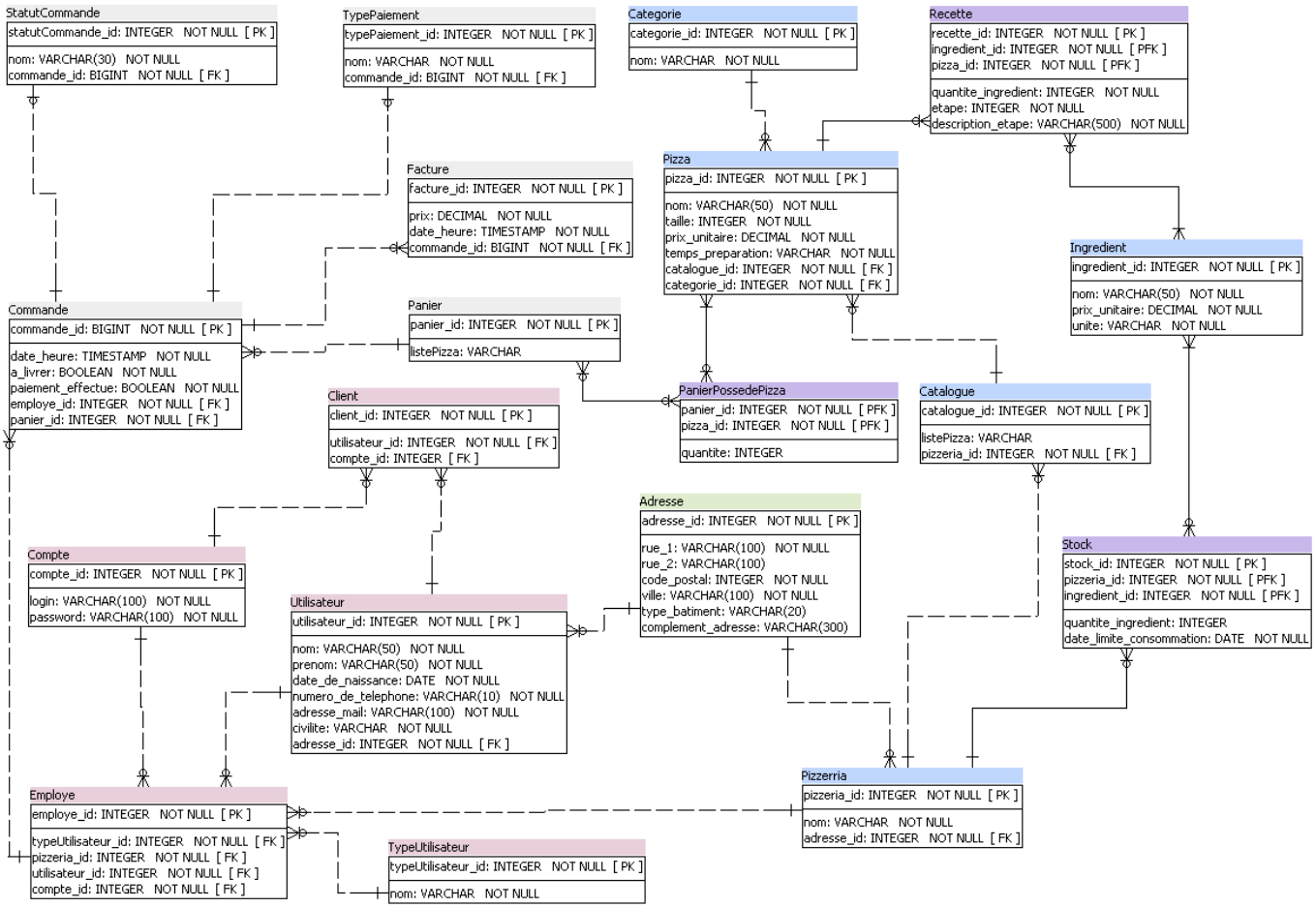
Cette classe est une classe d'association entre la classe **Pizzeria** et **Ingredient**. Un stock est associé à une pizzeria et à zéro ou une infinité d'ingrédients.

Attribut	Description
quantite_ingredient	La quantité de l'ingrédient
date_limite_consommation	La date limite de consommation de l'ingrédient

7 Modèle physique de données

Grâce au diagramme UML du domaine fonctionnel, nous avons pu établir le Modèle Physique de Données (MPD) de l'application. Ce modèle va nous permettre de modéliser la base de données de l'application OC PIZZA.

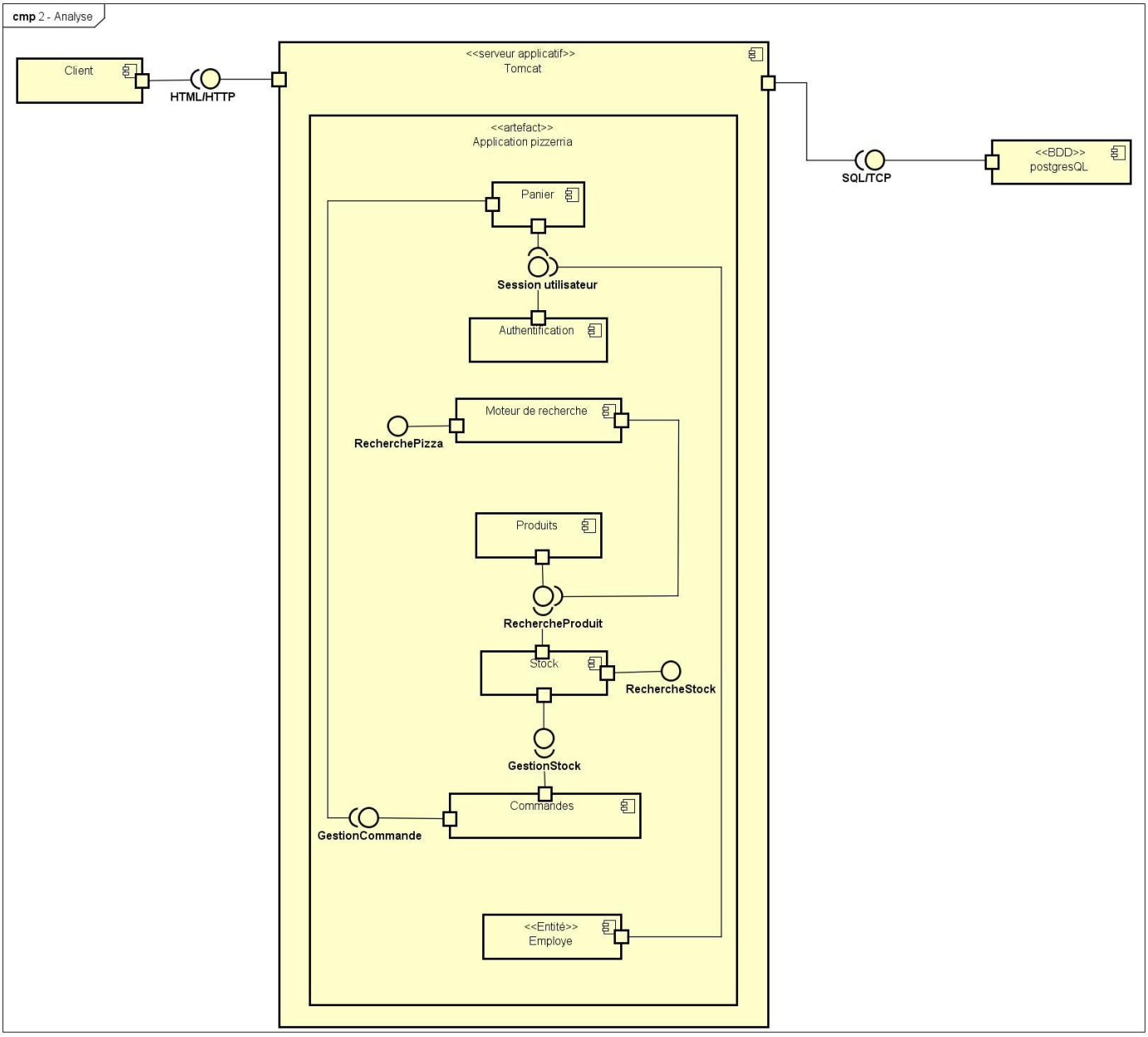
7.1.1 Description du MPD



7.2 Architecture de l'application

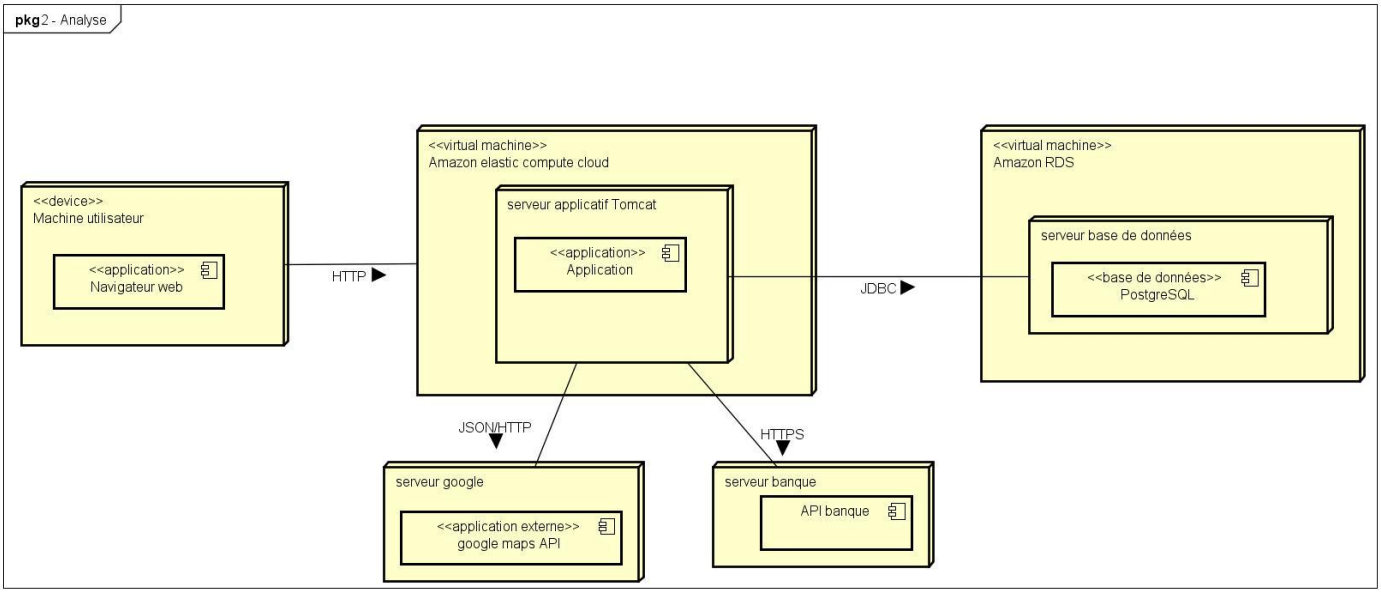
7.2.1 Diagramme de composant

Le diagramme de composant décrit l'organisation du système du point de vue des éléments logiciels. Il met en évidence les dépendances entre composants et décrit les interfaces entre les composants internes et externes.



7.2.2 Diagramme de déploiement

Le diagramme UML de déploiement vient compléter la description de l'architecture du système en identifiant les éléments matériels. Le diagramme fait apparaître les types de connexions, la répartition des composants et les chemins de communication entre les nœuds du système.



Nous avons décidé de déployer l'application sur une solution AMAZON Web Services (AWS). L'application sera hébergée sur une machine virtuelle Amazon Elastic Compute Cloud (Amazon EC2). Cette solution offre une capacité de calcul évolutive dans le cloud AWS. L'utilisation d'Amazon EC2 vous dispense d'investir à l'avance dans du matériel et, par conséquent, vous de développer et déployer les applications plus rapidement. Il est possible d'utiliser Amazon EC2 pour lancer autant de serveurs virtuels que nécessaire, configurer la sécurité et la mise en réseau, et gérer le stockage. Amazon EC2 permet également de monter ou descendre en puissance rapidement 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.

La partie Base De Données (BDD) sera déployée sur une machine virtuelle Amazon Relational Database Service (Amazon RDS). Amazon RDS permet d'installer, de gérer et de mettre à l'échelle facilement la base de données relationnelle dans le cloud. Ce service offre une capacité économique et ajustable ainsi qu'une automatisation des tâches administratives chronophages, telles que l'allocation de matériel, le paramétrage de bases de données, l'application de correctifs et les sauvegardes.