***Livrable 2***

Table des matières

[Base de données 1](#_Toc262899162)

[Dictionnaire de données 1](#_Toc1155849233)

[Modèle conceptuel de données 3](#_Toc377815361)

[Modèle logique de données 4](#_Toc891129104)

[Diagrammes UML 2.0 5](#_Toc907117339)

[Diagramme de cas d’utilisation 5](#_Toc1209909566)

[Diagrammes d’activité 7](#_Toc103114602)

[Diagramme de séquence 8](#_Toc36883541)

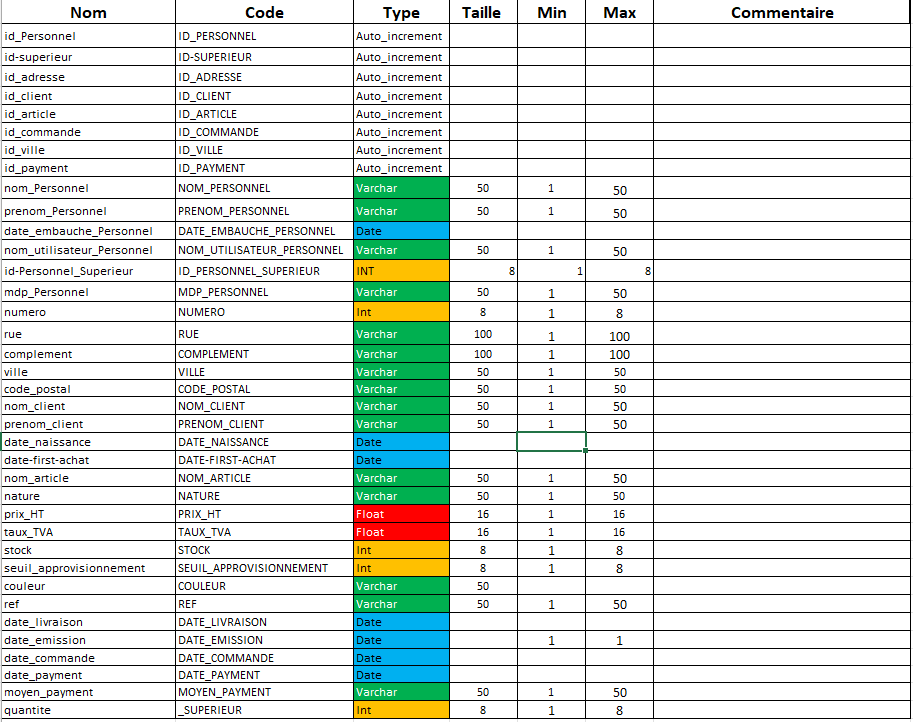
[Diagramme de classe 10](#_Toc991012733)

# Base de données

Pour réaliser la base de données nous avons procédé par étapes en commençant par le dictionnaire de données qui nous permet de regrouper toutes les données avec leurs différentes caractéristiques (Type, taille, Minimum, Maximum et description) puis la matrice de dépendance correspondante. Ensuite nous avons réalisé le modèle conceptuel de données (MCD) qui permet d’avoir une vue d’ensemble sur la base de données avec les entités et ses relations puis nous l’avons transformé en modèle logique de données (MLD) qui nous permet de transformer nos entités et relations en tables et d’avoir nos clés étrangères.

## Dictionnaire de données

Nous pouvons voir notre dictionnaire de données avec toutes les données qui seront intégré dans notre base de données avec leurs différentes Type, Taille, Min, Max et leurs commentaires. Nous pouvons constater notre matrice de dépendances avec les différents id et les données qui en dépendent.



## Modèle conceptuel de données

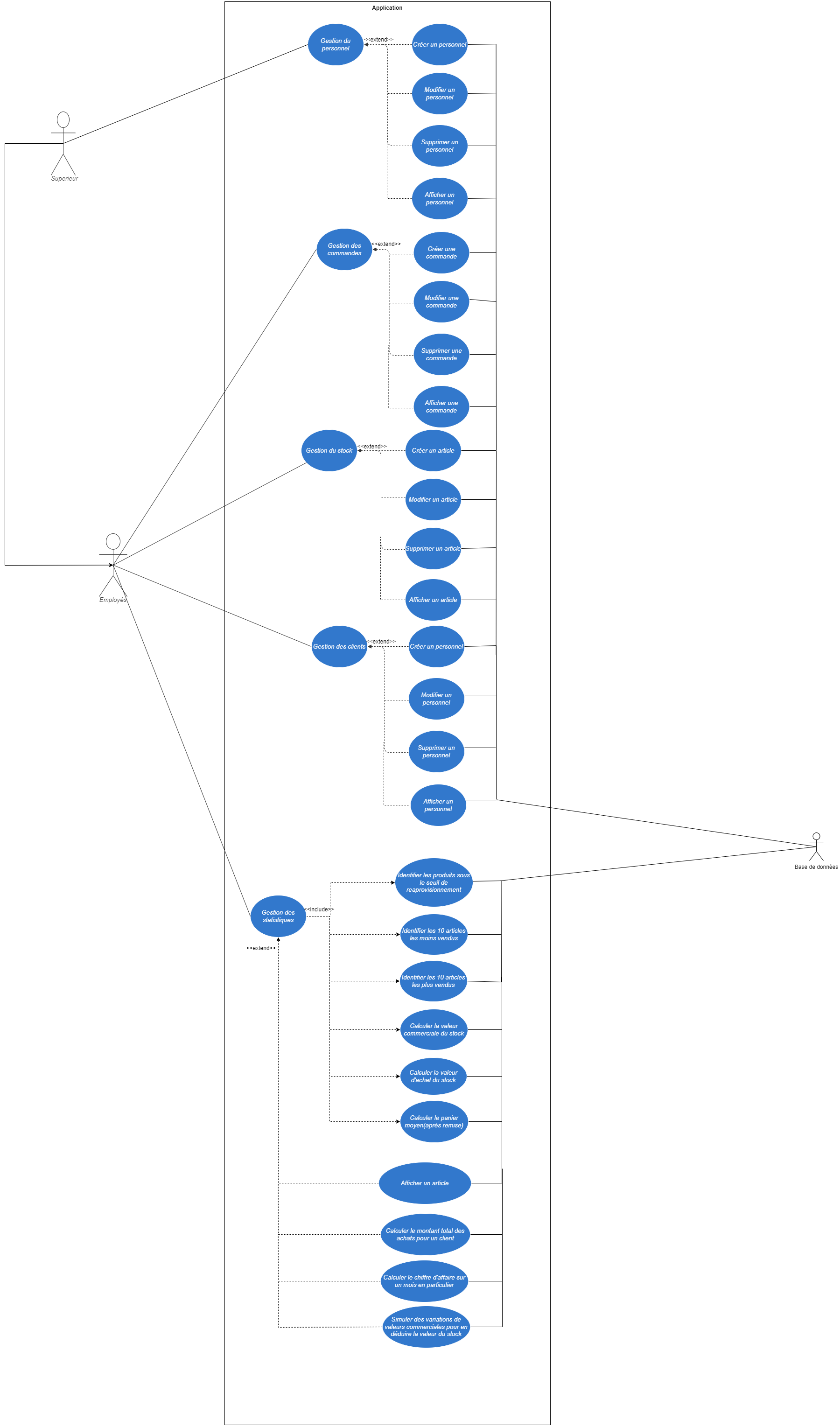
Le modèle conceptuel de données permet de voir l’ensemble de nos entités avec leurs relations ainsi que leurs cardinalités et leurs clé primaire. Nous avons décidé de créer une table par correspondance des données.

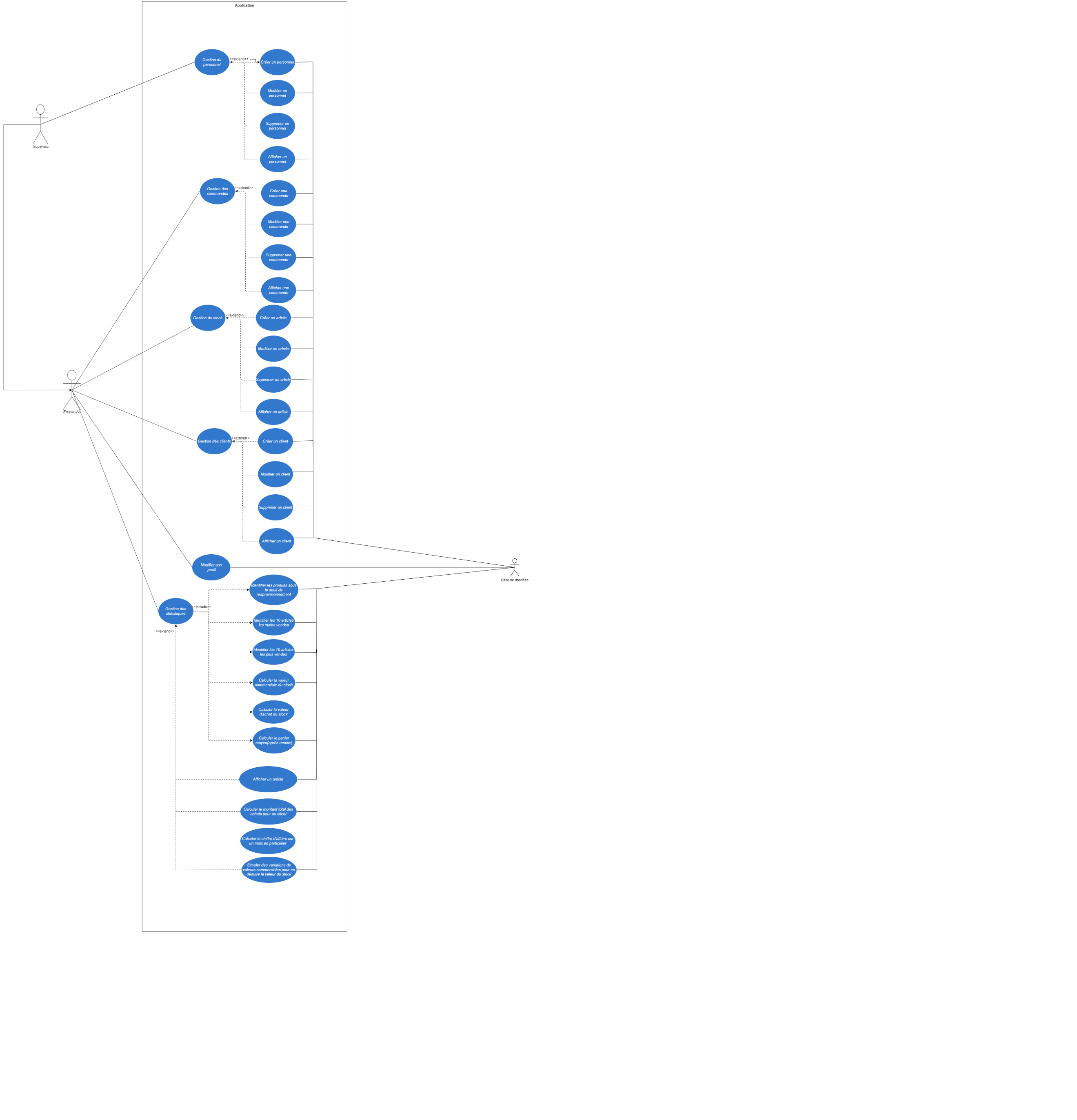
## Modèle logique de données

Le modèle logique de données permet de transformer nos entités en table ainsi que les relations et de créer nos clés étrangères, grâce à ce MLD nous pourrons faire l’implémentation de notre SGBDR. Notre base de données sera alors constituée de douze tables.

# Diagrammes UML 2.0

## Diagramme de cas d’utilisation



Nous pouvons voir notre diagramme de cas d’utilisation. Il représente l’ensemble des possibilités que peut réaliser un utilisateur sur notre logiciel. Nous avons distingué deux utilisateurs possibles car les rôles de ces derniers sont différents sur notre application.

## Diagrammes d’activité

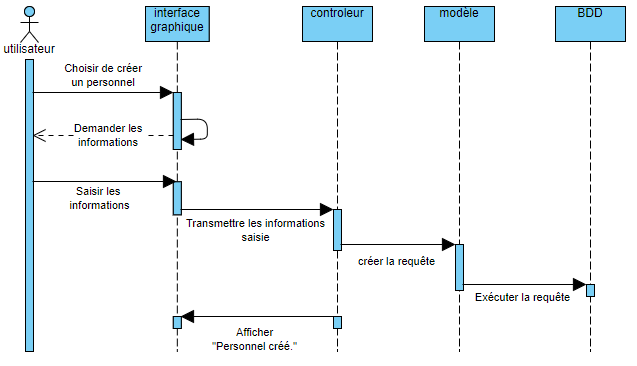
Notre diagramme d’activité présente le déroulement détaillé de tous les cas d’utilisation vu précédemment. Pour plus de simplicité nous avons représentés uniquement pour la gestion de personnel car le principe reste le même pour chaque item.

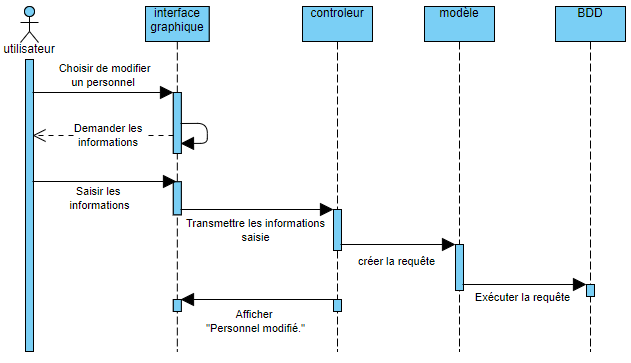
## Diagramme de séquence

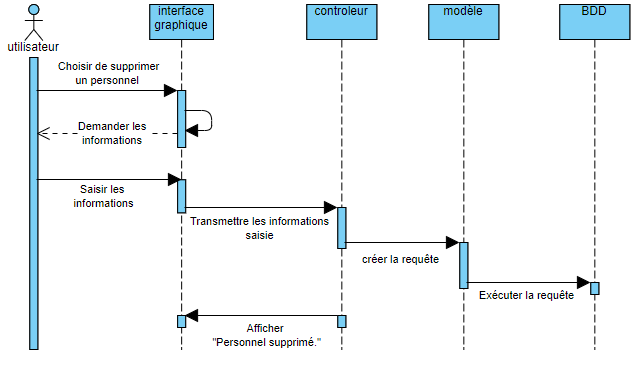
Un diagramme de séquence permet de voir l’aspect temporel de l’utilisation de notre logiciel.

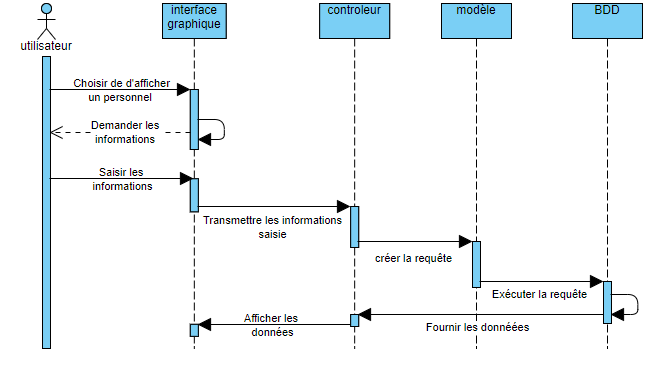
Pour plus de visibilité, nous avons fait 4 diagrammes de séquence décrivant chacun une action.

Nous pouvons voir ci-après le diagramme pour la création d’un personnel.

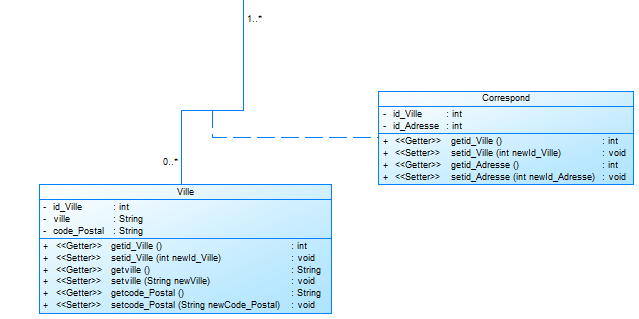


Ensuite nous voyons le digramme concernant la modification : 

Nous allons maintenant voir la suppression d’un personnel : 

Et pour finir l’affichage : 

## Diagramme de classe

Une image contenant texte, capture d’écran, intérieur

Description générée automatiquementNous pouvons voir ci-dessous le diagramme de classe, il représente l’ensemble des classes qui seront dans notre code avec pour chacune l’ensemble des attributs et méthodes. Nous pouvons aussi identifier la viabilité avec un – pour private, + pour public et # pour protected. 