Conception du Projet

# La Modélisation Conceptuelle

Avant – Propos :

Maintenant que le contexte est posé, nous allons nous pencher sur la modélisation de la base de données qui nous permettra de stocker et ré exploiter les données nécessaires à l’application.

Le Modèle Conceptuel de Données (MCD) : est la représentation abstraite et simplifiée des données d'un système d'information, mettant en avant les entités, leurs attributs et les relations entre elles, sans dépendance technologique.

# 

# 1.1 Modèle de dictionnaire à fournir

Différentes catégories du Dictionnaire des données :

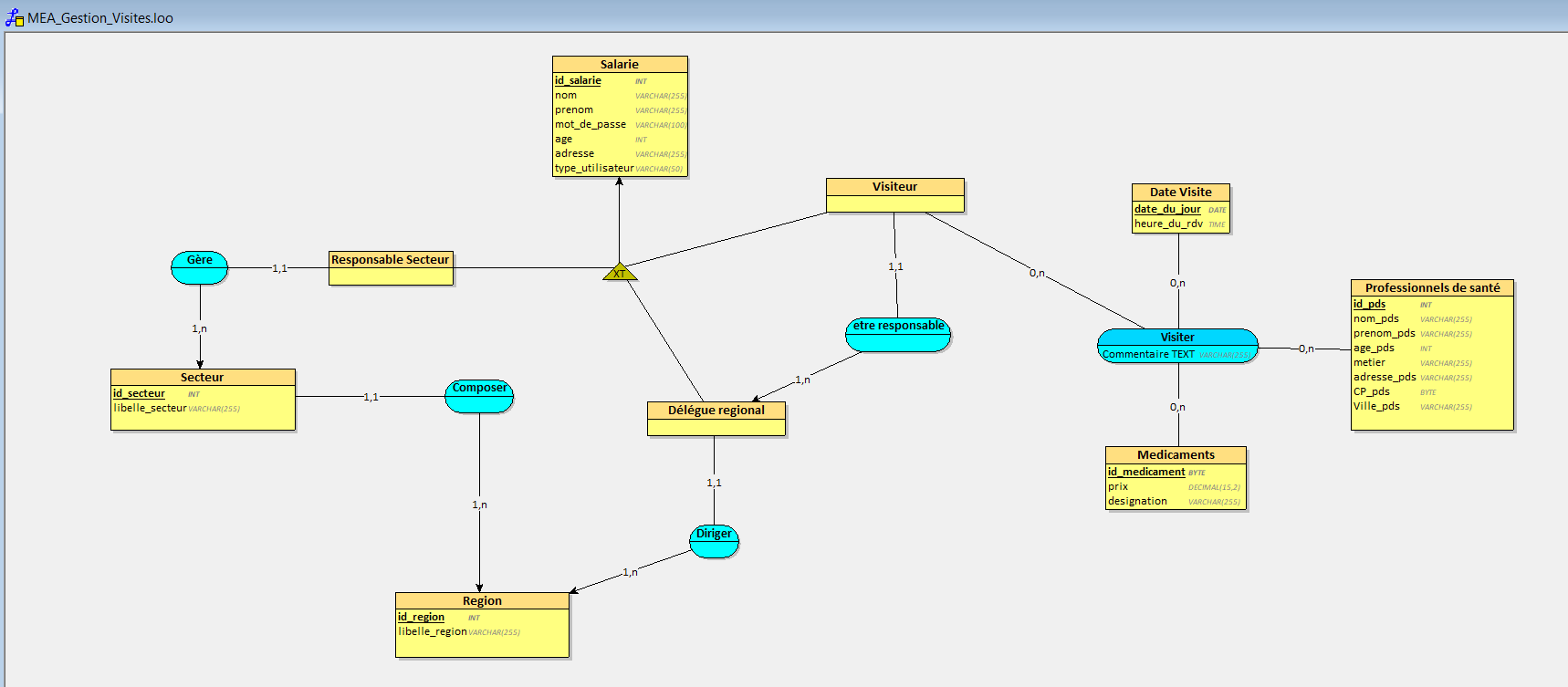
|  |
| --- |
|  Nom **Conceptuel** :   * Il s'agit du **nom** que l'on donne à une entité ou à un attribut dans la phase de conception, souvent exprimé en termes fonctionnels ou métier. Ce nom est plus compréhensible pour les utilisateurs finaux ou les analystes. Par exemple, "Nom du Visiteur", "Date de la Visite".    Nom **Logique** :   * C'est le **nom réel** que prendra l'attribut ou l'entité dans la base de données, c'est-à-dire la manière dont il sera défini dans la table.    Type **(E, Ca, Co)** :   * + **E** : **Entité** – Un objet principal du système.   + **Ca** : **Caractéristique** – Un attribut d'une entité   + **Co** : **Contrainte** – Une relation ou une règle entre deux entités    Nature :   * La **nature** de la donnée détermine quel type de données elle représente dans la base. Les types courants sont :   + **Entier** (INT)   + **Chaîne de caractères** (VARCHAR)   + **Texte** (TEXT)   + **Date** (DATE ou DATETIME)   + **Énumération** (ENUM)   + **Booléen** (BOOLEAN)    Longueur :   * La longueur maximale que l'attribut peut prendre. * N/A -> Non applicable (pour un texte ou une clé primaire)    Identifiant :   * Est-ce que cet attribut est un **identifiant unique** dans la base de données ?   + **Oui** (PK pour Primary Key)   + **FK** (Foreign Key) – S'il s'agit d'une clé étrangère, c'est-à-dire une référence à une autre table.   + **Non** – S'il s'agit d'un attribut standard.    Exemple **de Valeur** :   * Un exemple de valeur qui pourrait être stockée dans cet attribut. Cela permet de mieux comprendre la nature des données. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nom Logique | Nom Conceptuel | Type (E, CA,Co) | Nature | Longueur | Identifiant | Exemple de Valeur |
| Identifiant Utilisateur | id\_salarie | E | INT (Auto-incrémenté) | N/A | PK | 3 |
| Nom Utilisateur | nom | CA | VARCHAR | 255 | / | BLANCA |
| Prénom Utilisateur | prenom | CA | VARCHAR | 255 | / | Liz |
| Mot de Passe Utilisateur | mot\_de\_passe | CA | VARCHAR | 100 | / | Qoej\*24d |
| Age de l’Utilisateur | age | CA | INT | 3 | / | 19 |
| Adresse de l’Utilisateur | adresse | CA | VARCHAR | 255 | / | 24 rue de Belfort |
| Type de l’Utilisateur | type\_utilisateur | CA | Varchar | 255 | / | Visiteur |
| Identifiant du secteur | id\_secteur | E | INT (Auto-incrémentation | N/A | PK | 98 |
| Libelle du secteur | libelle\_secteur | CA | VARCHAR | 255 | / | Secteur Nord |
| Identifiant de la région | id\_region | E | INT (Auto-incrémentation) | N/A | PK | 56 |
| Date du jour de la Visite | date\_du\_jour | CA | DATE | 10 | / | 18/10/2024 |
| Heure du rendez-vous de la visite | heure\_rdv | CA | TIME | 5 | / | 15 :20 |
| Identifiant du Medicament | id\_medicaments | E | INT (Auto-incrémentation) | N/A | PK | 954 |
| Designation du médicament | designation | CA | VARCHAR | N/A | / | A.S.L PANPHARMA |
| Identifiant du professionnel de santé | id\_pds | CA | INT (Auto-incrémention) | N/A | PK | 653 |
| Nom du professionnel de santé | nom\_pds | CA | VARCHAR | 255 | / | M. Jambon |
| Prénom du professionnel de santé | prenom\_pds | CA | VARCHAR | 255 | / | Jean |
| Age du professionnel | age\_pds | CA | INT | 3 | / | 62 |
| Metier du professionnel | metier | CA | VARCHAR | 255 | / | Médecin généraliste |
| Adresse du professionnel | adresse\_pds | CA | VARCHAR | 255 | / | 24 rue de Test |
| Code Postal | CP\_pds | CA | INT | 10 | / | 90000 |
| Ville du professionnel de santé | Ville\_pds | CA | VARCHAR | 255 | / | Belfort |

# 1 .2 Modèle entité association

# Le modèle Entité-Association (MEA) ou modèle conceptuel de données, est une représentation graphique qui permet de visualiser et de modéliser les entités d'un système d'information ainsi que leurs relations. Chaque entité représente un objet important du système (par exemple, un utilisateur, un médicament, une visite), et chaque association (ou relation) montre comment ces entités interagissent entre elles. Le MEA est utilisé pour comprendre la structure de la base de données à un niveau conceptuel, avant de passer à l'implémentation.

Le MEA permet d'avoir une vue d'ensemble des données du système gérera et comment elles sont liées. Il aide à identifier les duplications de données et à les corriger, assurant ainsi une base de données efficace. Le MEA est la première étape pour créer la base de données physique (tables, clés primaires et étrangères).



**Explications** :

|  |
| --- |
| Cette modélisation permet d’attribuer facilement à chaque Salarié un rôle et des droits spécifiques. Chaque rôle est lié à un lieu où l’action auquel il possède des droits et devoirs.   * Chaque Salarié peut-être soit visiteur, soit Délégué régional soit Responsable de Secteur. Ces trois tables héritent des attributs de la table Salariés. * Le Responsable de Secteur Gère un unique secteur tandis qu’un secteur peut être gérer par un ou plusieurs responsables. * Chaque Secteur se compose d’une unique Région tandis qu’une région est composé d’un ou plusieurs secteurs. * Une région est dirigée par un ou plusieurs Déléguées de région tandis qu’un délégué ne s’occupe que d’une unique région. * Chaque délégué est responsable d’un ou plusieurs visiteurs et chaque visiteur possède un délégué comme responsable. * Un visiteur réalise 0 ou plusieurs visites et chaque visite possèdent un ou plusieurs visiteurs. * Chaque visite possède 0 ou plusieurs date, professionnelles de santé et médicaments. |

# 2. Modèle Logique

|  |
| --- |
| À ce stade, la fonction principale du modèle de données est de visualiser les éléments de données et la façon dont ils sont liés les uns aux autres. La modélisation logique des données permet également de détailler les attributs associés à un élément de données. Par exemple, un modèle logique de données spécifierait la nature d’un élément de données, c’est-à-dire le nom du compte (chaîne de caractères), le numéro de compte (nombre entier).  Cette étape plus structurée de la modélisation des données est plus pertinente pendant la conception de l’application, lorsqu’elle peut servir de mécanisme de communication dans les environnements plus techniques où travaillent les analystes et les concepteurs de bases de données. Elle nous aide à comprendre les détails des données dans une plus large mesure que les modèles conceptuels de données, sans toutefois fournir de perspective sur la façon dont elle devrait être mise en œuvre.  On notera les noms des « tables » (‘’entités’’ dans le MCD) en minuscule (lowercase) et il est possible de traduire les termes en anglais pour rendre le MLD plus universel (non réalisé dans ce projet).  Les différentes étapes sont :   * + 1. Réécriture des entités en tant que tables sur chaque ligne (à l’exception des tables relationnelles     2. On spécifie pour chaque table ses attributs entre parenthèses en adoptant une casse pour nos attributs (en général la snake\_case)     3. On ajoute pour chaque table sa clé primaire en première position     4. On ajoute les relations via les clés étrangères selon les cardinalités. |

***MLD*** *(textuel)*

|  |
| --- |
| * salarie (id\_salarie, nom, prenom, mot\_de\_passe, age, adresse, type\_utilisateur) * responsable secteur (#id\_salarie, #id\_secteur) * delegue\_regional (#id\_salarie, #id\_region) * visiteur (#id\_salarie) * secteur (id\_secteur, libelle\_secteur) * region (id\_region, libelle\_region) * date\_visite (date\_du\_jour, heure\_du\_rdv) * medicaments (id\_medicaments, prix, designation) * professionnels de sante (id\_pds, nom\_pds, prenom\_pds, age\_pds, metier, adresse\_pds, CP\_pds, Ville\_pds) * visiter( #date\_du\_jour, #id\_medicament, #id\_pds, #id\_salarie) |

* La Table « visiter » découle des cardinalités 0:n , elle devient une table de liaison. La clé primaire est formé par un groupe de clé étrangères (4 tables liés).
* Les tables delegue regional, responsable secteur et visiteur hérite de la table « salarie ».

# 3. Script de création de la base de données

Voici le script SQL pour la création des tables en tenant compte des clés primaires, des clés étrangères et des index :

|  |
| --- |
| -- Création de la table salarie  CREATE TABLE salarie (  id\_salarie INT AUTO\_INCREMENT PRIMARY KEY,  nom VARCHAR(255) NOT NULL,  prenom VARCHAR(255) NOT NULL,  mot\_de\_passe VARCHAR(100) NOT NULL,  age INT NOT NULL,  adresse VARCHAR(255) NOT NULL,  type\_utilisateur VARCHAR(50) NOT NULL  );  -- Création de la table secteur  CREATE TABLE secteur (  id\_secteur INT AUTO\_INCREMENT PRIMARY KEY,  libelle\_secteur VARCHAR(255) NOT NULL  );  -- Création de la table region  CREATE TABLE region (  id\_region INT AUTO\_INCREMENT PRIMARY KEY,  libelle\_region VARCHAR(255) NOT NULL  );  -- Création de la table responsable\_secteur  CREATE TABLE responsable\_secteur (  id\_salarie INT,  id\_secteur INT,  PRIMARY KEY (id\_salarie, id\_secteur),  FOREIGN KEY (id\_salarie) REFERENCES salarie(id\_salarie) ON DELETE CASCADE,  FOREIGN KEY (id\_secteur) REFERENCES secteur(id\_secteur) ON DELETE CASCADE  );  -- Création de la table delegue\_regional  CREATE TABLE delegue\_regional (  id\_salarie INT,  id\_region INT,  PRIMARY KEY (id\_salarie, id\_region),  FOREIGN KEY (id\_salarie) REFERENCES salarie(id\_salarie) ON DELETE CASCADE,  FOREIGN KEY (id\_region) REFERENCES region(id\_region) ON DELETE CASCADE  );  -- Création de la table visiteur  CREATE TABLE visiteur (  id\_salarie INT PRIMARY KEY,  FOREIGN KEY (id\_salarie) REFERENCES salarie(id\_salarie) ON DELETE CASCADE  );  -- Création de la table date\_visite  CREATE TABLE date\_visite (  date\_du\_jour DATE,  heure\_du\_rdv TIME,  PRIMARY KEY (date\_du\_jour)  );  -- Création de la table medicaments  CREATE TABLE medicaments (  id\_medicaments INT AUTO\_INCREMENT PRIMARY KEY,  prix DECIMAL(15, 2) NOT NULL,  designation VARCHAR(255) NOT NULL  );  -- Création de la table professionnels\_de\_sante  CREATE TABLE professionnels\_de\_sante (  id\_pds INT AUTO\_INCREMENT PRIMARY KEY,  nom\_pds VARCHAR(255) NOT NULL,  prenom\_pds VARCHAR(255) NOT NULL,  age\_pds INT NOT NULL,  metier VARCHAR(255) NOT NULL,  adresse\_pds VARCHAR(255) NOT NULL,  CP\_pds INT NOT NULL,  ville\_pds VARCHAR(255) NOT NULL  );  -- Création de la table visiter  CREATE TABLE visiter (  date\_du\_jour DATE,  id\_medicament INT,  id\_pds INT,  id\_salarie INT,  commentaire TEXT,  PRIMARY KEY (date\_du\_jour, id\_medicament, id\_pds, id\_salarie),  FOREIGN KEY (date\_du\_jour) REFERENCES date\_visite(date\_du\_jour) ON DELETE CASCADE,  FOREIGN KEY (id\_medicament) REFERENCES medicaments(id\_medicaments) ON DELETE CASCADE,  FOREIGN KEY (id\_pds) REFERENCES professionnels\_de\_sante(id\_pds) ON DELETE CASCADE,  FOREIGN KEY (id\_salarie) REFERENCES visiteur(id\_salarie) ON DELETE CASCADE  );  -- Index sur les colonnes fréquemment recherchées  CREATE INDEX idx\_nom\_salarie ON salarie(nom);  CREATE INDEX idx\_prenom\_salarie ON salarie(prenom);  CREATE INDEX idx\_nom\_pds ON professionnels\_de\_sante(nom\_pds);  CREATE INDEX idx\_prenom\_pds ON professionnels\_de\_sante(prenom\_pds);  CREATE INDEX idx\_designation\_medicament ON medicaments(designation); |

### Explications :

1. **Clés Primaires et Auto-increment :**
   * Pour les tables comme salarie, secteur, region, medicaments, et professionnels\_de\_sante, j'ai défini une clé primaire avec un type AUTO\_INCREMENT pour générer automatiquement les identifiants uniques.
2. **Clés Étrangères et Relations :**
   * Les tables comme responsable\_secteur, delegue\_regional, visiteur, et visiter utilisent des **clés étrangères** pour maintenir l'intégrité référentielle. J'ai ajouté la clause ON DELETE CASCADE pour supprimer en cascade les enregistrements liés lorsque l'enregistrement parent est supprimé.
3. **Table visiter :**
   * La table visiter contient des clés étrangères reliant date\_visite, medicaments, professionnels\_de\_sante, et visiteur. Toutes ces relations sont modélisées avec des clés primaires composites pour garantir que chaque combinaison soit unique.
4. **Index :**
   * J'ai ajouté des **index** sur les colonnes fréquemment recherchées telles que nom, prenom, nom\_pds, prenom\_pds, et designation pour optimiser les performances des recherches.

### Vérification des contraintes d'intégrité :

* Les **clés primaires** sont correctement définies dans toutes les tables.
* Les **clés étrangères** sont utilisées pour garantir l'intégrité référentielle entre les tables.
* Les **index** sont créés pour optimiser les recherches sur les champs les plus souvent interrogés, comme les noms, prénoms et désignations.

# Modélisation UML

### Le Diagramme de Contexte : Explication et Construction

Le **diagramme de contexte** est une représentation simple du système dans son environnement. Il est souvent considéré comme une **vue globale** qui montre comment le système interagit avec les **acteurs externes** (utilisateurs, autres systèmes, etc.). Son principal objectif est de donner une **vue d'ensemble du système** sans entrer dans les détails techniques ou les processus internes.

#### ****Pourquoi le Diagramme de Contexte est Important ?****

1. **Vue d'ensemble** : Il montre rapidement **les limites du système**, c'est-à-dire ce qui se trouve **à l'intérieur** (les fonctionnalités principales) et ce qui se trouve **à l'extérieur** (les acteurs externes).
2. **Communication** : C’est un outil essentiel pour **communiquer facilement** avec des personnes non techniques, car il ne contient pas de détails complexes. Il aide à comprendre **qui** interagit avec le système et **comment**.
3. **Définir les acteurs** : Il clarifie quels sont les **acteurs** et leurs **interactions** avec le système. Les acteurs peuvent être des personnes, des groupes, ou même d'autres systèmes logiciels.

#### ****Étapes pour Construire le Diagramme de Contexte****

1. **Identifier le Système** : Il faut d'abord **définir le système** lui-même, c'est l'élément central du diagramme. Dans notre cas, ce serait le système de **gestion des visites médicales**. On représentera le système par une boîte ou un rectangle central.
2. **Identifier les Acteurs Externes** : Un acteur est tout ce qui **interagit avec le système**, mais qui **ne fait pas partie du système**. Ce sont souvent :
   * **Utilisateurs humains** : Par exemple, les visiteurs, les professionnels de santé, les responsables secteur, les délégués régionaux.
   * **Autres systèmes** : Si ton système communique avec des **bases de données externes** ou d'autres systèmes informatiques, ils doivent aussi être représentés comme des acteurs.
3. **Représenter les Interactions** : L’on dessinera des flèches ou des lignes entre les **acteurs** et le **système** pour représenter les **interactions**. Ces interactions décrivent **ce que fait chaque acteur** avec le système, comme une action ou un flux de données.

Le Diagrammes de contexte est donc composé de :

|  |
| --- |
| * Le système (représenté par une boîte centrale)   + Nommé « système de gestion des visites médicales * Les acteurs (autour du système), représenté par des « bonhommes ». * **Visiteur** : Planifie et enregistre les visites, ajoute des commentaires et des informations. * **Responsable de secteur** : Supervise les visiteurs et les affecte à des secteurs. * **Délégué régional** : Supervise les responsables secteur. * Les interactions représentées par des flèches liant les utilisateurs aux systèmes. * Le **visiteur** interagit avec le système pour enregistrer les visites, ajouter des commentaires sur les professionnels de santé et les médicaments. * Les **responsables de secteur** gèrent les visiteurs dans leur zone d'affectation. * Les **délégués régionaux** supervisent les opérations à un niveau plus large, mais ne traitent pas directement avec les professionnels de santé. |

Diagrammes de Contexte :

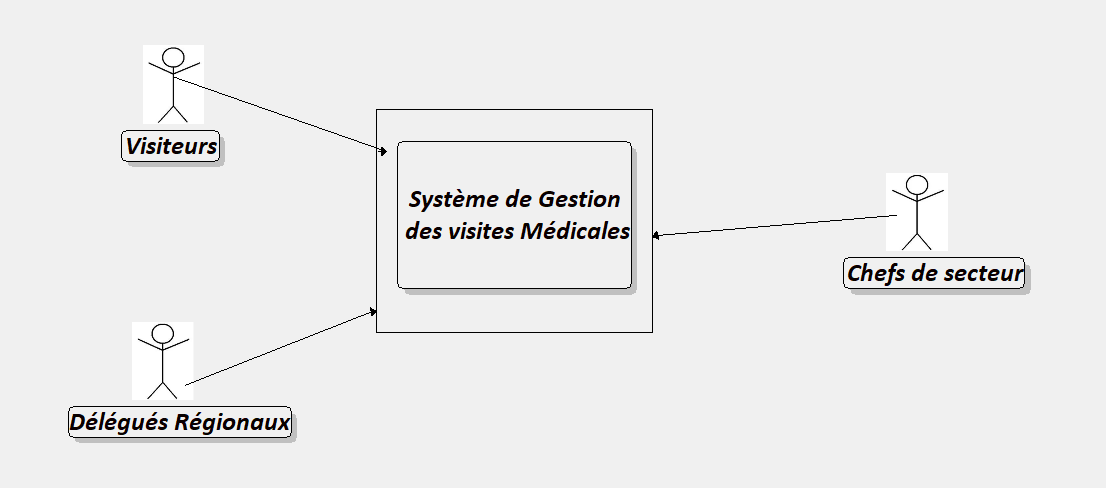


Diagramme des cas d’utilisations :

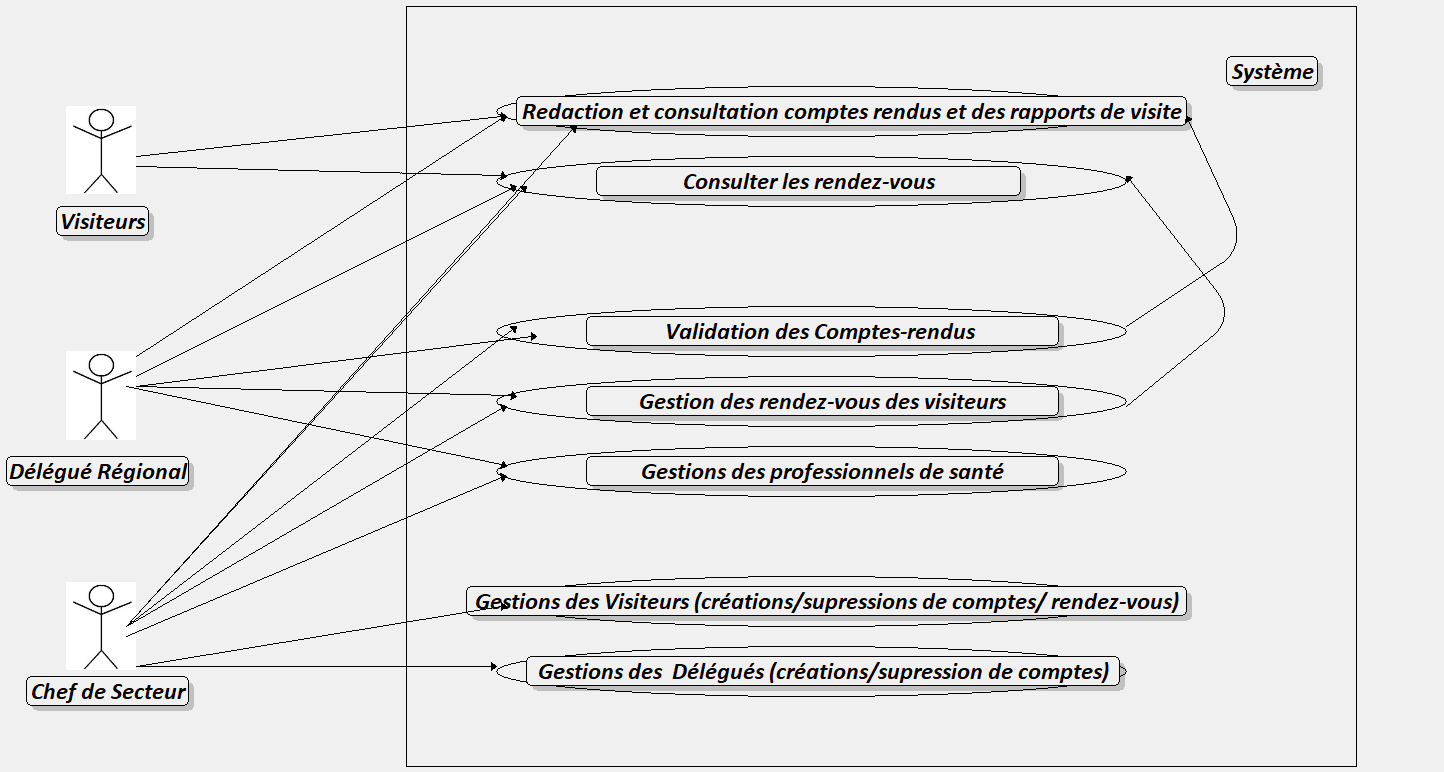
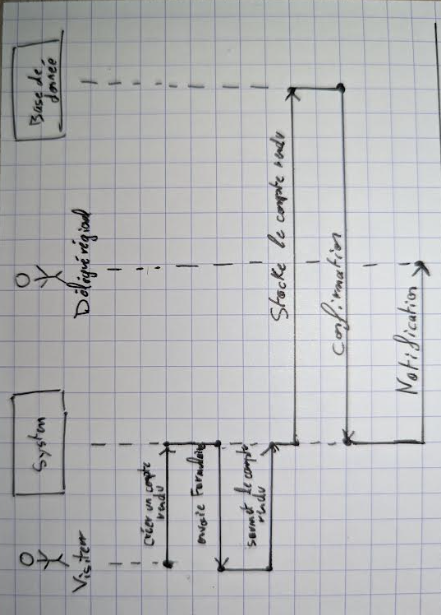


Diagramme de Séquence :

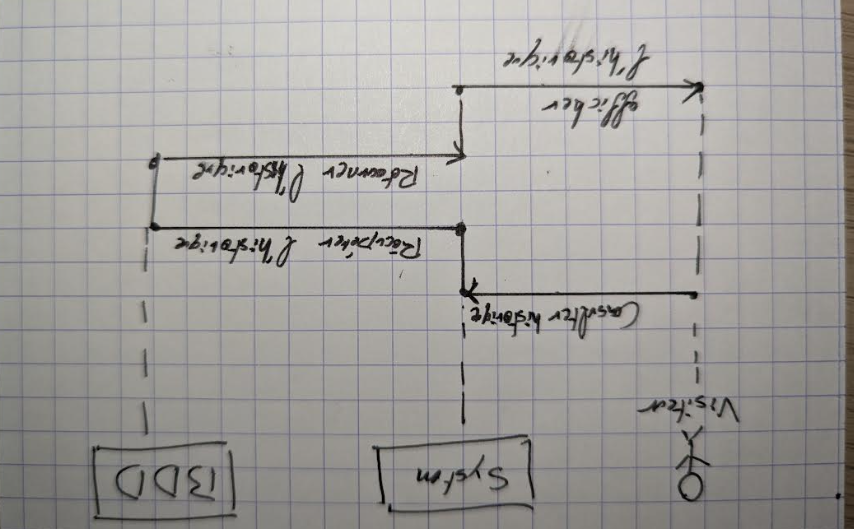
Exemples : Réalisation d’un compte-rendu :



Explications :

|  |
| --- |
| * Le visiteur désigne l’action de créer un compte-rendu (suite à une visite) via un bouton de l’application. * L’application récupère l’information et lui envoie le formulaire à compléter. * Le visiteur remplis puis soumet le compte-rendu * L’application stocke le compte-rendu dans la base de donnée * La base de donnée envoie confirmation du stockage à l’application. * Le système (l’application) notifie le délégué régional de la présence d’un nouveau compte-rendu. |

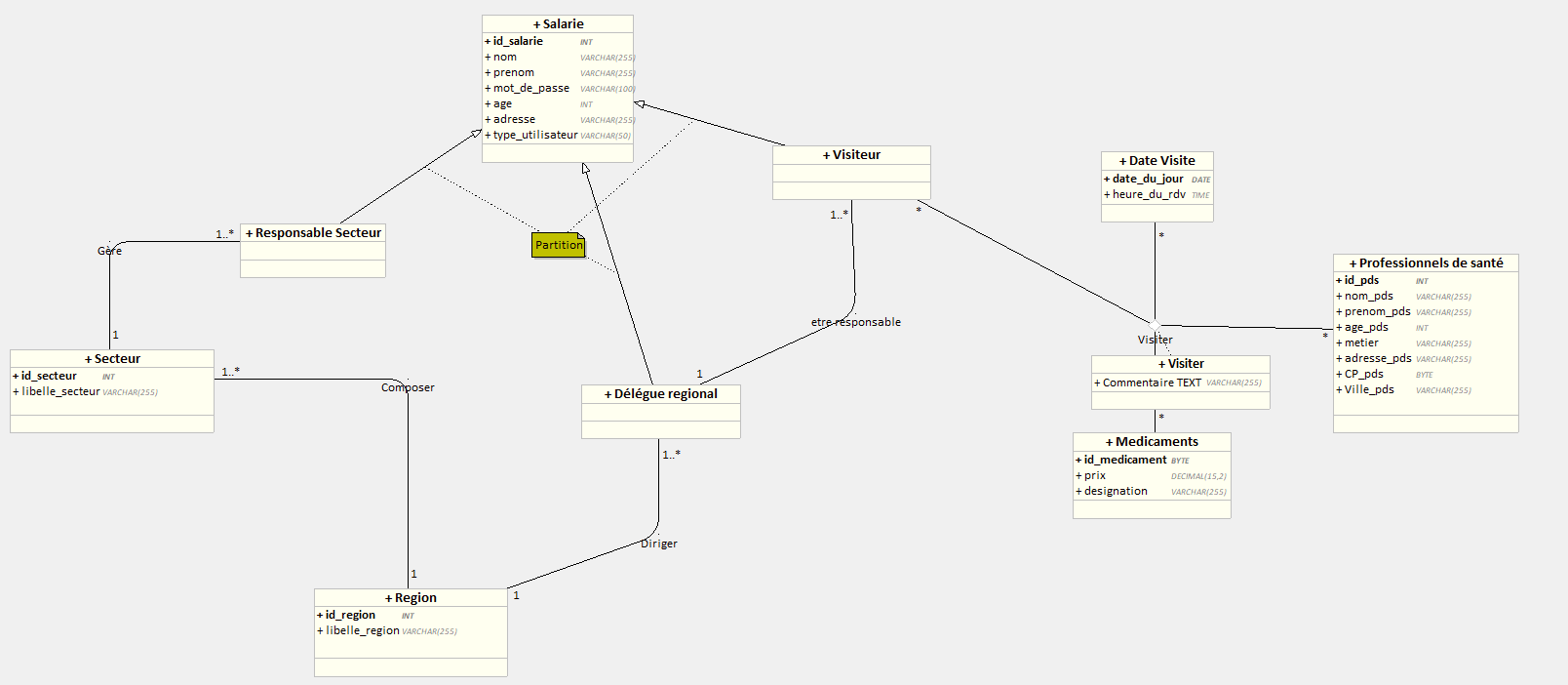
Diagramme de Séquence – Consultation de l’historique:



Explications :

|  |
| --- |
| * Le visiteur désigne l’action de visionner son historique (suite à une visite) via un bouton de l’application. * Le système récupère l’historique sur la Base de Donnée * La base de donnée renvoie au système l’historique * L’application affiche l’historique chez l’utilisateur. |

Diagramme de Class :



Le logiciel ne me permettant pas d’entrer les fonctions de chaque classe dans le second encadré de chaque table en voici une partie :

+----------------------+ | Salarie | +----------------------+ |

+ creerSalarie(nom : String, prenom : String, mot\_de\_passe : String, age : Int, adresse : String, type\_utilisateur : String) : void |

| + mettreAJourSalarie(id\_salarie : Int, nom : String, prenom : String, age : Int, adresse : String) : void |

| + supprimerSalarie(id\_salarie : Int) : void |

| + seConnecter(email : String, mot\_de\_passe : String) : boolean | | + verifierDroits(type\_utilisateur : String, action : String) : boolean |

| + seDeconnecter(email : String, mot\_de\_passe : String) : boolean | | + verifierDroits(type\_utilisateur : String, action : String) : boolean |

+----------------------+

+----------------------+ | Visiteur | +----------------------+ |

planifierVisite(date : DATE, heure : TIME, id\_pds : INT, id\_medicament : INT) : void

annulerVisite(id\_visite : INT) : void

consulterHistoriqueVisites(id\_salarie : INT) : List<Visite>

+----------------------+ | Secteur| +----------------------+ |

creerSecteur(libelle\_secteur : String) : void

modifierSecteur(id\_secteur : Int, libelle\_secteur : String) : void

supprimerSecteur(id\_secteur : Int) : void

+----------------------+ | Region| +----------------------+ |

creerRegion(libelle\_region : String) : void

modifierRegion(id\_region : Int, libelle\_region : String) : void

supprimerRegion(id\_region : Int) : void

+----------------------+ | Medicaments| +----------------------+ |

ajouterMedicament(designation : String, prix : Decimal) : void

modifierMedicament(id\_medicament : Int, designation : String, prix : Decimal) : void

supprimerMedicament(id\_medicament : Int) : void

+----------------------+ | Visiter | +----------------------+ |

enregistrerVisite(date : DATE, id\_medicament : INT, id\_pds : INT, id\_salarie : INT) : void

modifierVisite(date\_du\_jour : DATE, id\_medicament : INT, id\_pds : INT, id\_salarie : INT) : void

supprimerVisite(date\_du\_jour : DATE, id\_salarie : INT) : void