Dossier d'analyse

Projet PUBIO

Table des matières

1 Introduction

PUBLIO a pour objectif la conception d'un système d'information permettant d'aider à la gestion de la bibliothèque de chaque enfant de la famille et plus précisément, aider à la gestion des prêts, des demandes de prêt et retours des livres.

Le logiciel, qui fera suite à cette analyse, sera installé sur un ordinateur dans un réseau local. Il devra permettre la gestion des livres en prêt ainsi que des utilisateurs. Le détenteur d'une bibliothèque pourra autoriser l'accès à sa bibliothèque à un utilisateur, accepter les demandes de prêt et évaluer l'emprunteur sur le retour des prêts (état du livre et respect des délais de retour de livre). Une option est installée pour qu'un utilisateur, puisse partager lui aussi sa bibliothèque.

Au cour de ce dossier, nous allons présenter différents diagrammes modélisés selon la méthode UML (Unified Modeling Language). Chaque modélisation sera annotée, la démarche que nous aurons suivit, les différentes hypothèses soulevées et les différents choix effectués seront expliqués, jusqu'au résultat final de notre travail.

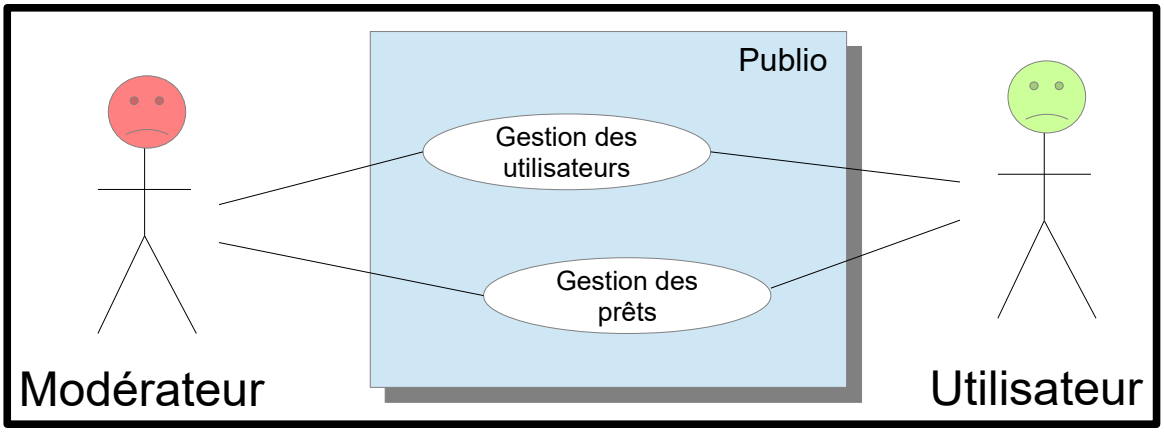
2 Modélisation UML

2.1 Cas d'utilisation

2.1.1 Approche simple

Dans un premier niveau, On retrouve l'utilisateur (le copain), le modérateur (un membre de la famille), un système comprenant :

- la création et la modification d'un compte utilisateur, qu'on appelle la gestion des utilisateurs,
- la demande de prêt, la mise en prêt, la prise en charge du retour, appelons cela la gestion des prêts.



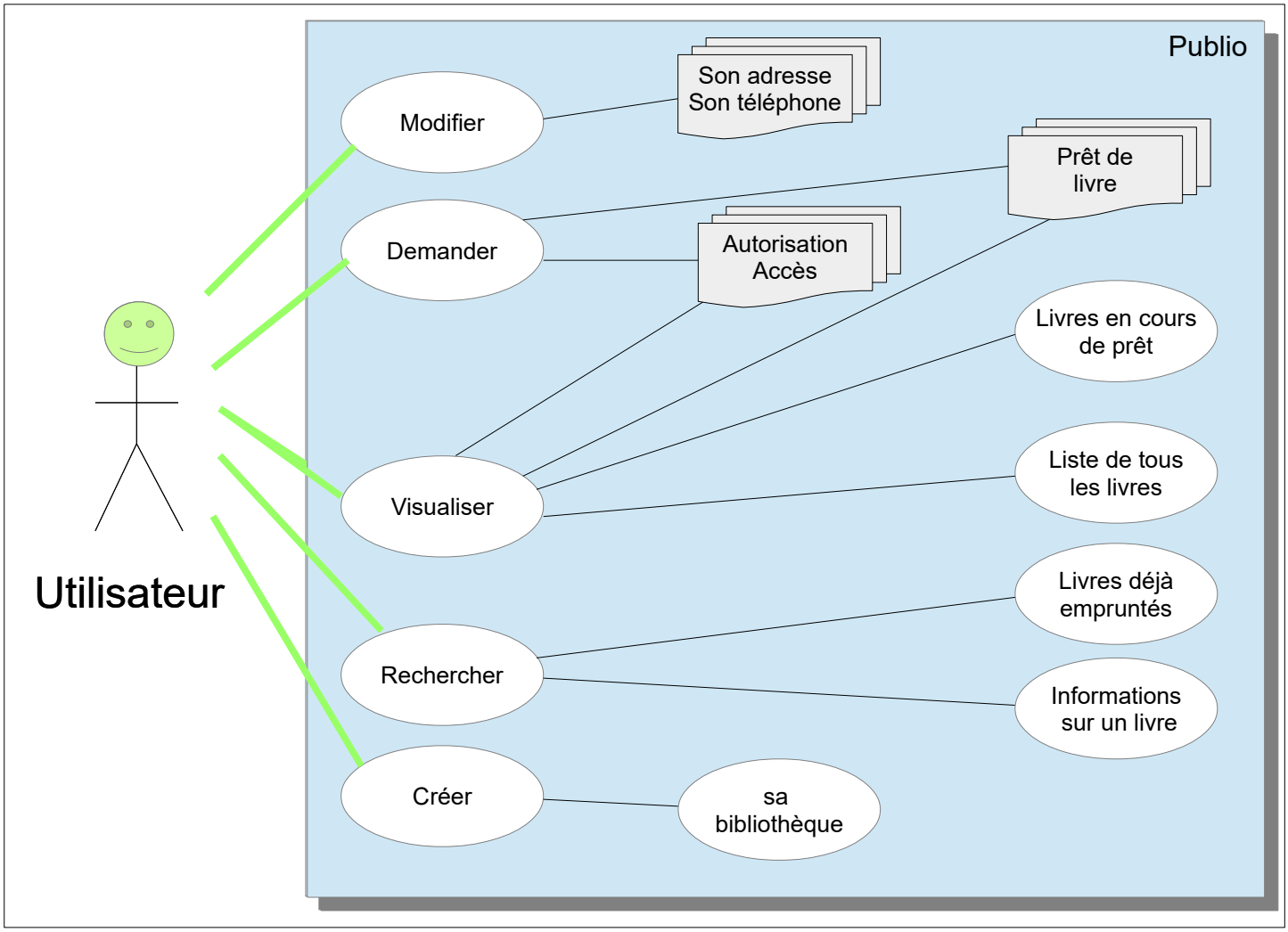
Cette vision nous apporte que trop peu d'informations. Il nous faut une représentation plus détaillée des relations existantes pour chacun des acteurs dans le système, afin de mieux appréhender le modèle conceptuel.

2.1.2 Cas d'utilisation avec la modélisation des acteurs

2.1.2.1 Vue de l'utilisateur

Si on reprend le paragraphe 4.2.2 du cahier des charges, la liste des actions que peut mener un utilisateur est :

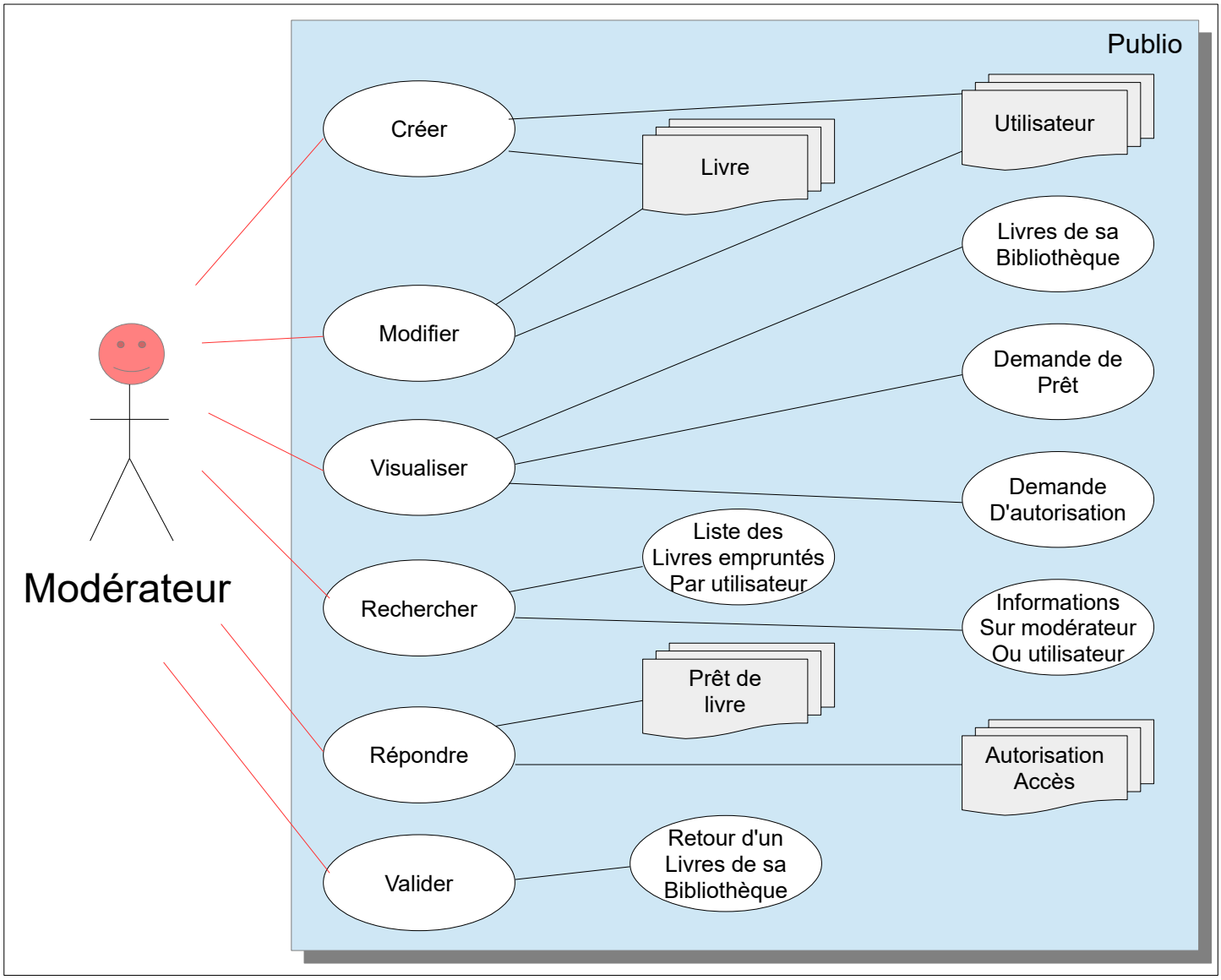
- modifier son compte (adresse, téléphone),
- visualiser l'ensemble des livres de toutes les bibliothèques avec une arborescence si plusieurs livres portent le même titre, indiquer le nombre,
- visualiser les livres qu'il emprunte (max 2),
- rechercher les livres qu'il a déjà emprunté,
- rechercher les informations d'un livre,
- demander le prêt d'un livre à son modérateur,
- visualiser la demande de prêt,
- demander une autorisation à un modérateur pour emprunter dans sa bibliothèque,
- visualiser la demande d'autorisation,
- créer une bibliothèque.



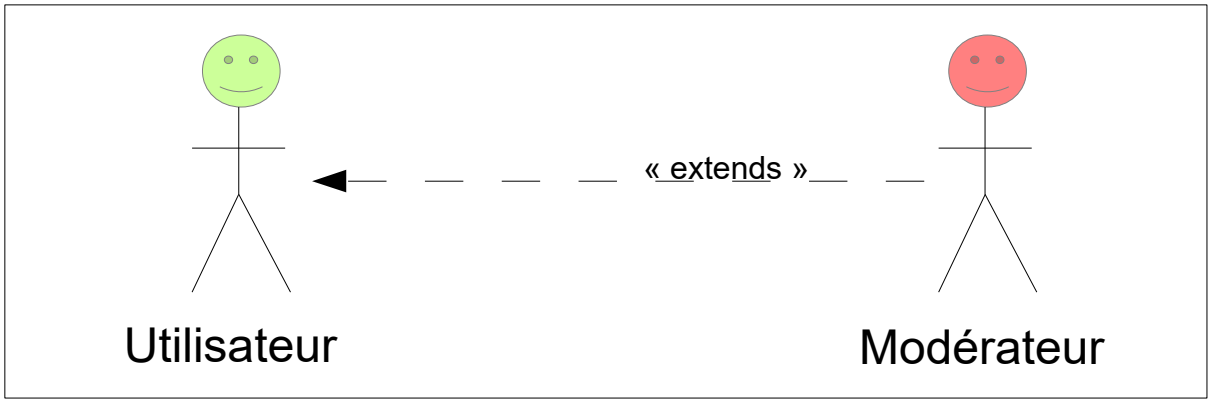
2.1.2.2 Vue du modérateur

la liste des actions que peut mener un modérateur est :

- créer un compte utilisateur,
- créer un livre,
- modifier dans un compte utilisateur (adresse, téléphone),
- modifier dans un livre de sa bibliothèque, ainsi que la date de prêt,
- visualiser les livres de sa bibliothèque,
- rechercher la liste des livres qu'un des ses copains à emprunter,
- rechercher les informations d'un utilisateur, d'un modérateur,
- visualiser les demandes de prêt à traiter,
- répondre à un utilisateur sur le prêt d'un livre (accepter ou refuser),
- visualiser les demandes d'autorisation à traiter,
- répondre à une demande d'autorisation de prêt d'un utilisateur, ou d'un autre modérateur,
- valider le retour d'un livre (état du livre et disponibilité).



2.1.2.3 Relation entre le modérateur et l'utilisateur



On définit le modérateur comme étant un utilisateur à la seule différence de celui-ci, c'est qu'il possède une bibliothèque. Part cela on part du principe que les actions que peut mener un utilisateur seront réalisées aussi par un modérateur.

Dans l'entité « utilisateur », il y a un attribut se nommant : détient une bibliothèque c'est un booléen, avec 0 est égal à non, et 1 qui est égale à oui. Si l'utilisateur détient une bibliothèque, il a enregistré au moins un livre, alors il devient un modérateur.

Dans l'entité « livre », il y a un attribut se nommant : état du livre, c'est un booléen, avec 0 est égal à le livre est bon, et 1 qui est égale à le livre est en mauvais état.

2.2 Les règles de gestion *(§4.2.3 du cahier des charges)*

Cas d'utilisation : modifier données utilisateur		
Acteur :	Utilisateur	Modérateur
Description	• Saisir son adresse et son numéro de téléphone • Modification dans la table de la BDD	• Choisir l'utilisateur parmi la liste des utilisateurs liés • Saisir son adresse et son numéro de téléphone • Modification dans la table de la BDD

Cas d'utilisation : demander le prêt d'un livre	
Acteur :	Utilisateur
Description	• Vérifier le Nombre de prêt en cours de l'utilisateur • Choisir le livre à emprunter parmi la liste de tous les livres Demander le prêt d'un livre à un modérateur, agit dans la table « demander » : • vérifier dans la table « lier » que l'utilisateur soit rattaché au modérateur ◦ si oui : incrémente la table « demander » avec les informations du livres, du demandeur et le détenteur du livre. ◦ si non : signale à l'utilisateur qu'il n'est pas encore autorisé dans la bibliothèque

Cas d'utilisation : demander autorisation accès	
Acteur :	Utilisateur
Description	• Choisir le modérateur parmi la liste de tous les modérateurs Demander une autorisation à un modérateur, agit sur « autoriser » : • incrémente « autoriser » avec le nom et prénom de l'utilisateur et du modérateur

Cas d'utilisation : visualiser les autorisations accès		
Acteur :	Utilisateur	Modérateur
Description	• Liste de toutes les autorisations d'accès de l'utilisateur	• Liste de toutes les autorisations d'accès du modérateur

Cas d'utilisation : Visualiser les prêts de livre		
Acteur :	Utilisateur	Modérateur
Description	• Liste de tous les prêts de livre de l'utilisateur	• Liste de tous les prêts de livre du modérateur

Cas d'utilisation : visualiser livre en cours de prêt	
Acteur :	Utilisateur
Description	• Liste au maximum 2 livres Méthode : lorsqu'un livre est prêté, il y a une incrémentation dans « retourner » avec le titre, le sous-titre et le tome, la date n'est pas renseignée. On aura le résultat par la jointure entre « prêter » et « retourner »

Cas d'utilisation : visualiser liste de tous les livres	
Acteur :	Utilisateur
Description	• Liste des livres dans « livre »

Cas d'utilisation : rechercher livres déjà emprunter	
Acteur :	Utilisateur
Description	• Liste des livres que l'utilisateur a emprunté et rendu Méthode : jointure entre « prêter » et « retourner »

Cas d'utilisation : rechercher informations sur un livre	
Acteur :	Utilisateur
Description	• Sélectionner un livre parmi la liste de tous les livres • visualiser l'ensemble des informations

Cas d'utilisation : créer sa bibliothèque	
Acteur :	Utilisateur
Description	Créer une bibliothèque, agit sur « utilisateur » : • modifie l'attribut « détient une bibliothèque » par oui au lieu de non • démarre l'action créer un livre

Cas d'utilisation : créer un livre	
Acteur :	Modérateur
Description	agit sur « livre » : • insérer le titre, le sous-titre, le tome, l'auteur, le nombre de pages, état du livre (bon ou mauvais)

Cas d'utilisation : créer un utilisateur	
Acteur :	Modérateur
Description	agit sur la table « utilisateur » : • insérer nom, prénom, adresse, numéro de téléphone • met à zéro automatiquement, les avertissements retard et mauvais état rendu • met automatiquement, non sur « détient une bibliothèque » • création d'un lien entre le nouvel utilisateur et le modérateur

Cas d'utilisation : modifier un livre	
Acteur :	Modérateur
Description	agit sur « livre » : choisir quel élément est à modifier • modifie 1 des élément : le titre, le sous-titre, le tome, l'auteur, le nombre de pages, ainsi que la date de retour de prêt

Cas d'utilisation : visualiser la liste des livres de sa bibliothèque	
Acteur :	Modérateur
Description	agit sur « livre » : • liste les livres ayant le nom et prénom du modérateur

Cas d'utilisation : Rechercher la liste des livres empruntés par un de ses utilisateurs	
Acteur :	Modérateur
Description	• Sectionner un des utilisateurs parmi la liste Méthode : jointure entre « prêter » et « retourner »

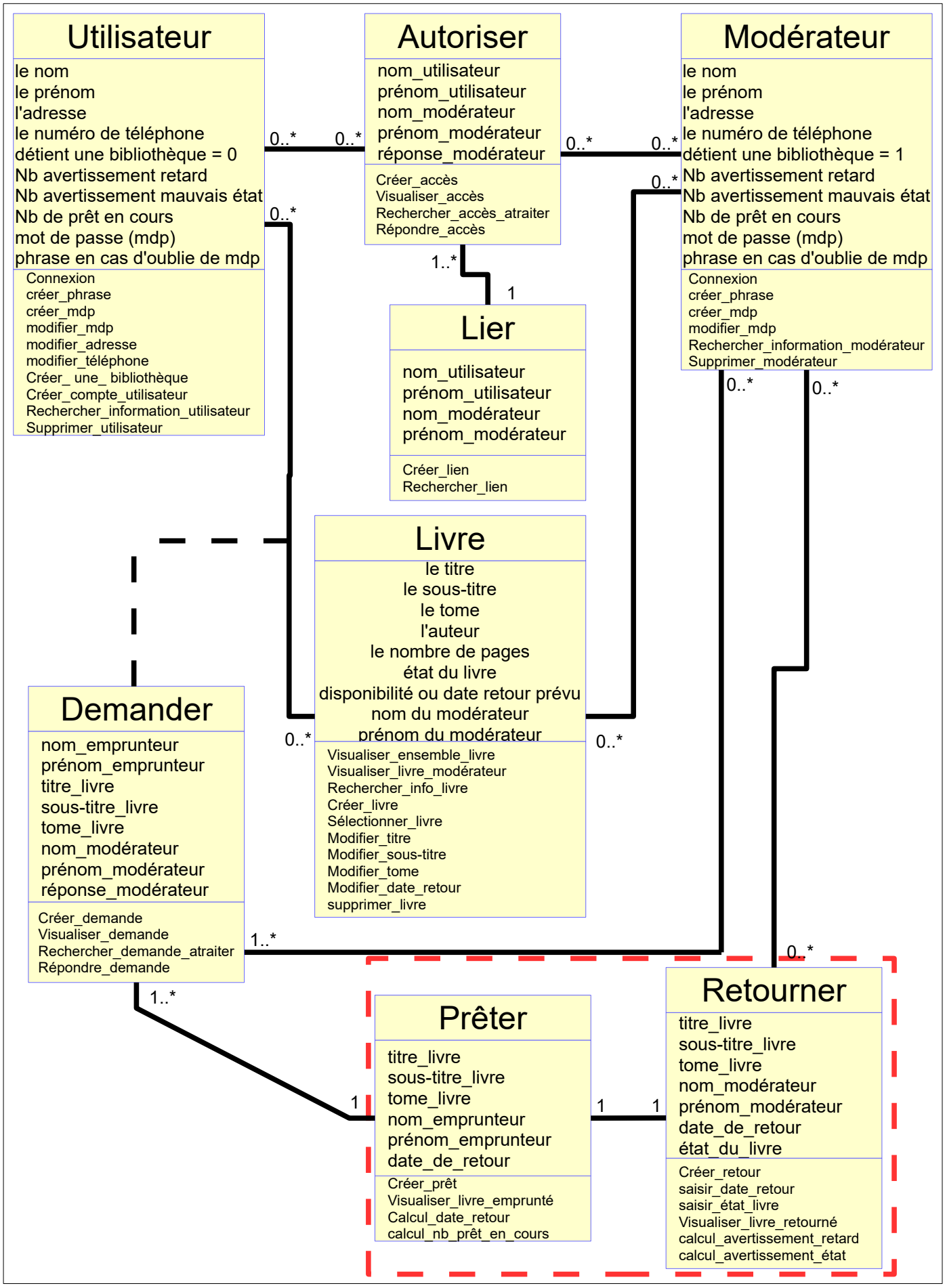
Cas d'utilisation : rechercher informations sur un utilisateur / modérateur	
Acteur :	Modérateur
Description	Agit sur « utilisateur » • Sélectionner un utilisateur parmi la liste de tous les utilisateurs • visualiser l'ensemble des informations

Cas d'utilisation : répondre prêt de livre	
Acteur :	Modérateur
Description	Inclus à l'activité « demander » : • Affichage de la liste des demandes à traiter • Sélectionner une demande parmi la liste • Saisir oui ou non • si oui : ◦ insertion dans la table« prêter » : titre, sous-titre, tome du livre, le nom du utilisateur, le prénom d'utilisateur, calculer la date de retour ◦ ajouter date de retour dans le « livre » ◦ insertion dans la table « retourner » : le nom, prénom du modérateur du livre, titre, sous-titre, tome du livre ◦ incrémentation de +1 dans Nombre de prêt en cours de l'utilisateur • si non : retirer de la liste des demandes de prêts à traiter

Cas d'utilisation : répondre autorisation d'accès	
Acteur :	Modérateur
Description	Inclus à l'activité « autoriser » : • Affichage de la liste des demandes à traiter • Sélectionner une demande parmi la liste • Saisir oui ou non • si oui : ◦ insertion dans la table « lier » : le nom et le prénom de l'utilisateur, le nom et le prénom du modérateur • si non : retirer de la liste des demandes d'autorisation à traiter

Cas d'utilisation : valider le retour d'un livre	
Acteur :	Modérateur
Description	Inclus dans la relation « retourner » : • Affichage de la liste des retours pas encore saisie • Sélectionner le livre à retourner • Insertion de l'état du livre et de la date de retour • Calcul des points d'avertissement • Modifie le « livre » retrait de la date de retour par disponible • Modifie le « livre » sur l'état si changement • Décrémenter de 1 dans « utilisateur » : nb de prêt en cours

2.3 Diagrammes de classes

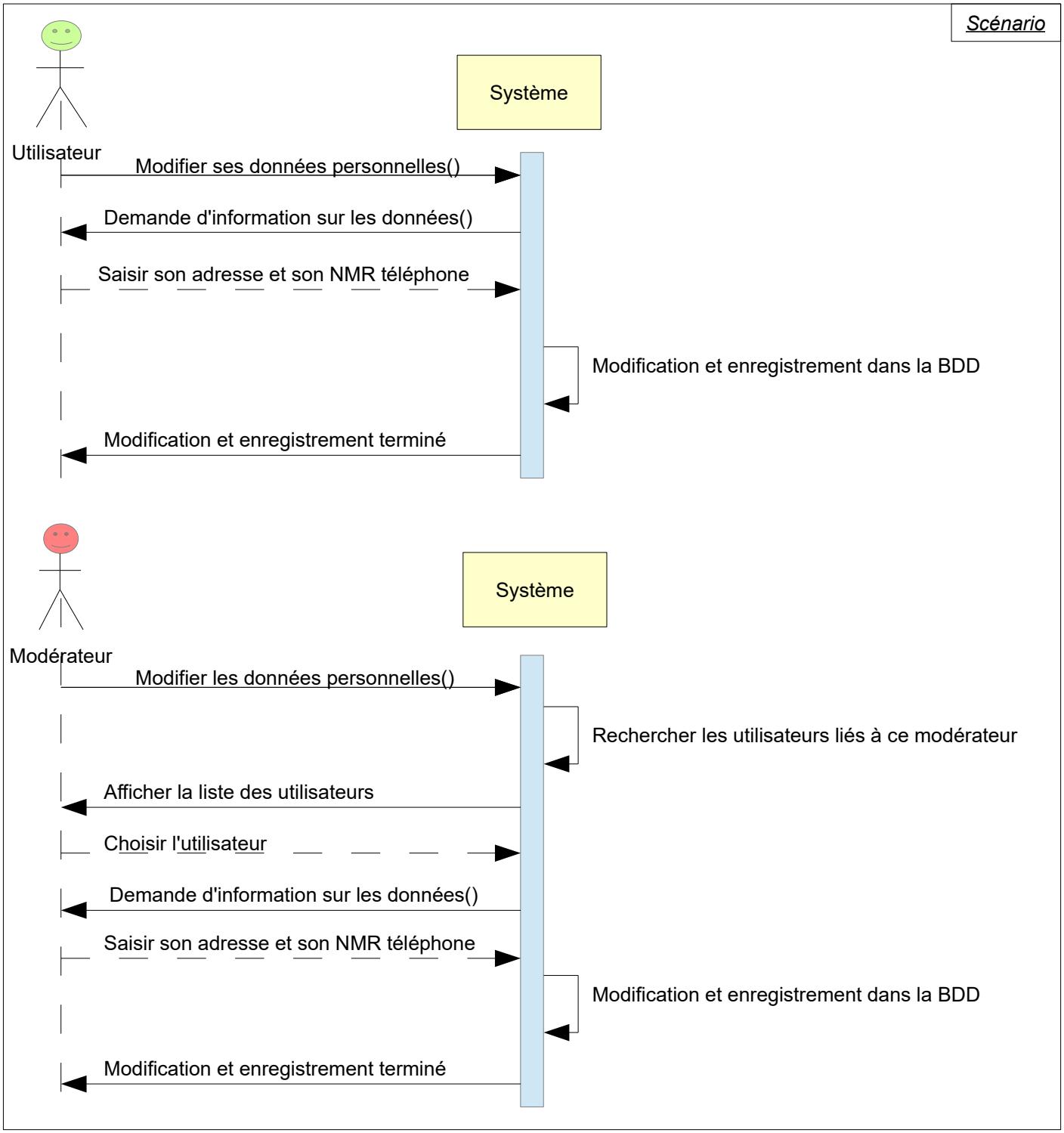


Observation : On remarque que les classes prêter et retourner sont de cardinalité de 1 de chaque côté. Il sera alors plus judicieux de ne faire qu'une seule classe. Ainsi, lors de la création des tables il sera plus facile de gérer le nombre de livre en cours de prêts si Prêter et Retourner sont dans la même table.

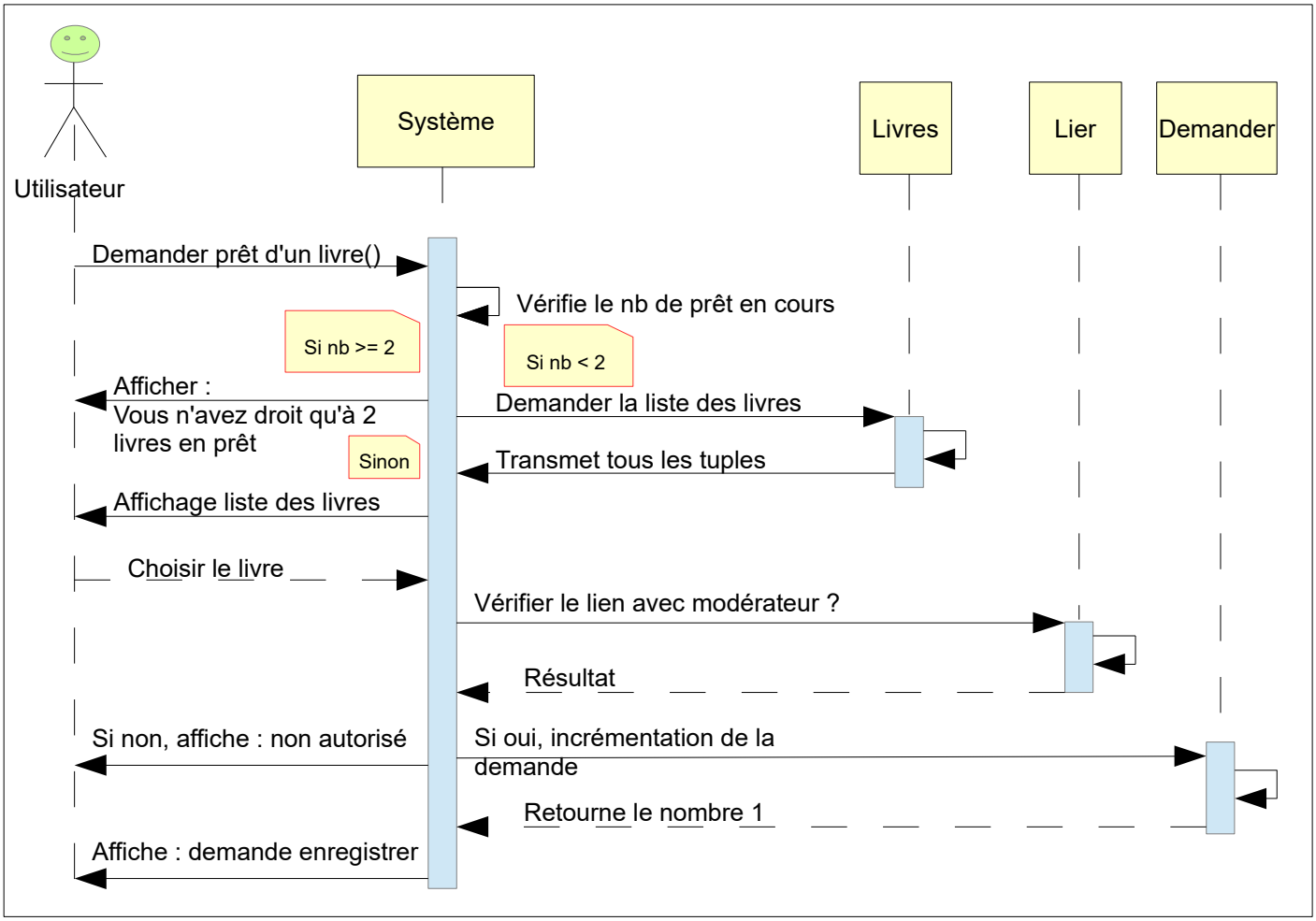
2.4 Diagramme de séquences

Les différents scénarios et diagrammes de séquences découlent des cas d'utilisation. Ici nous présentons chaque cas d'utilisation avec ses fonctions et ses exceptions.

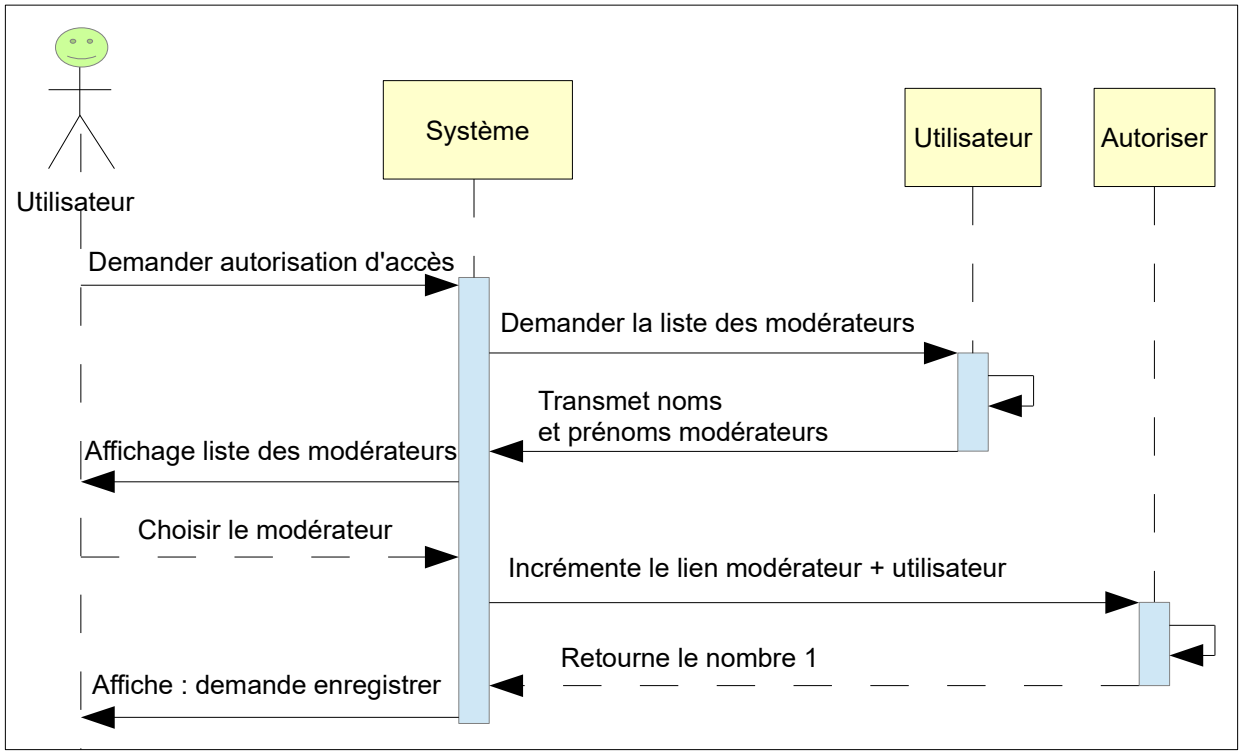
2.4.1 modifier données utilisateur



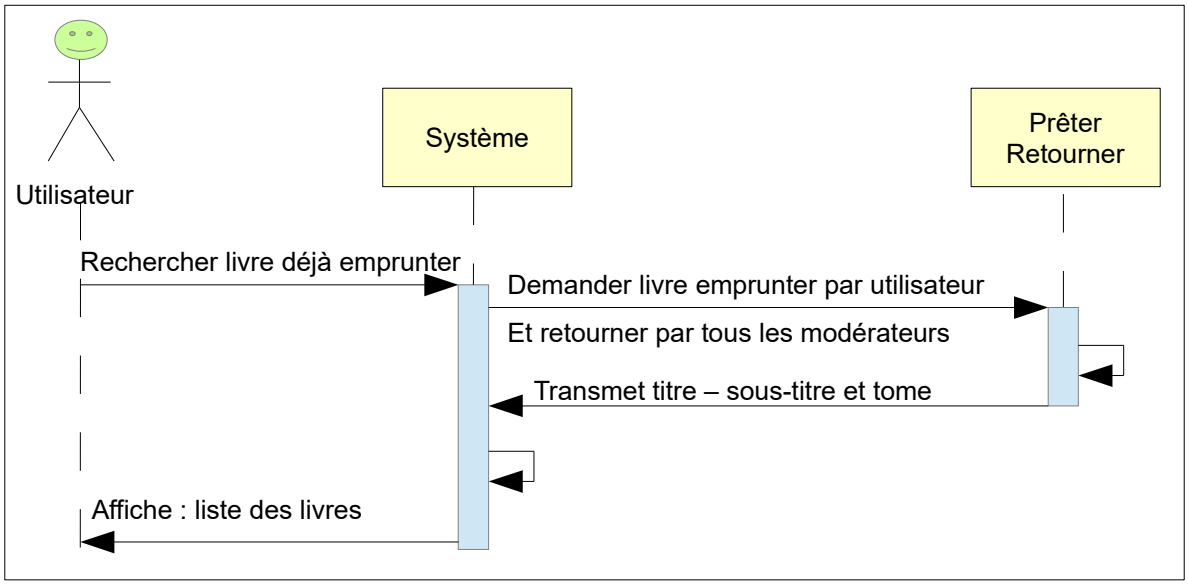
2.4.2 demander le prêt d'un livre



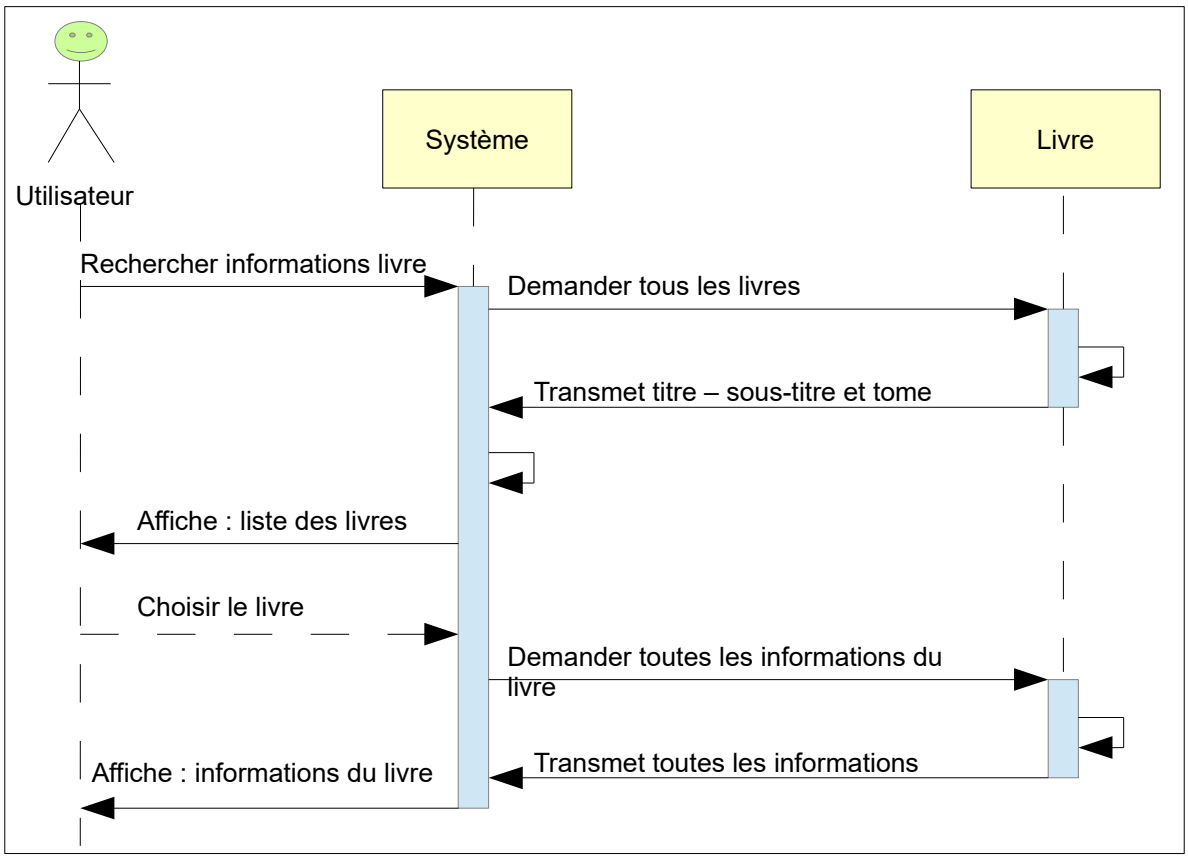
2.4.3 demander autorisation accès



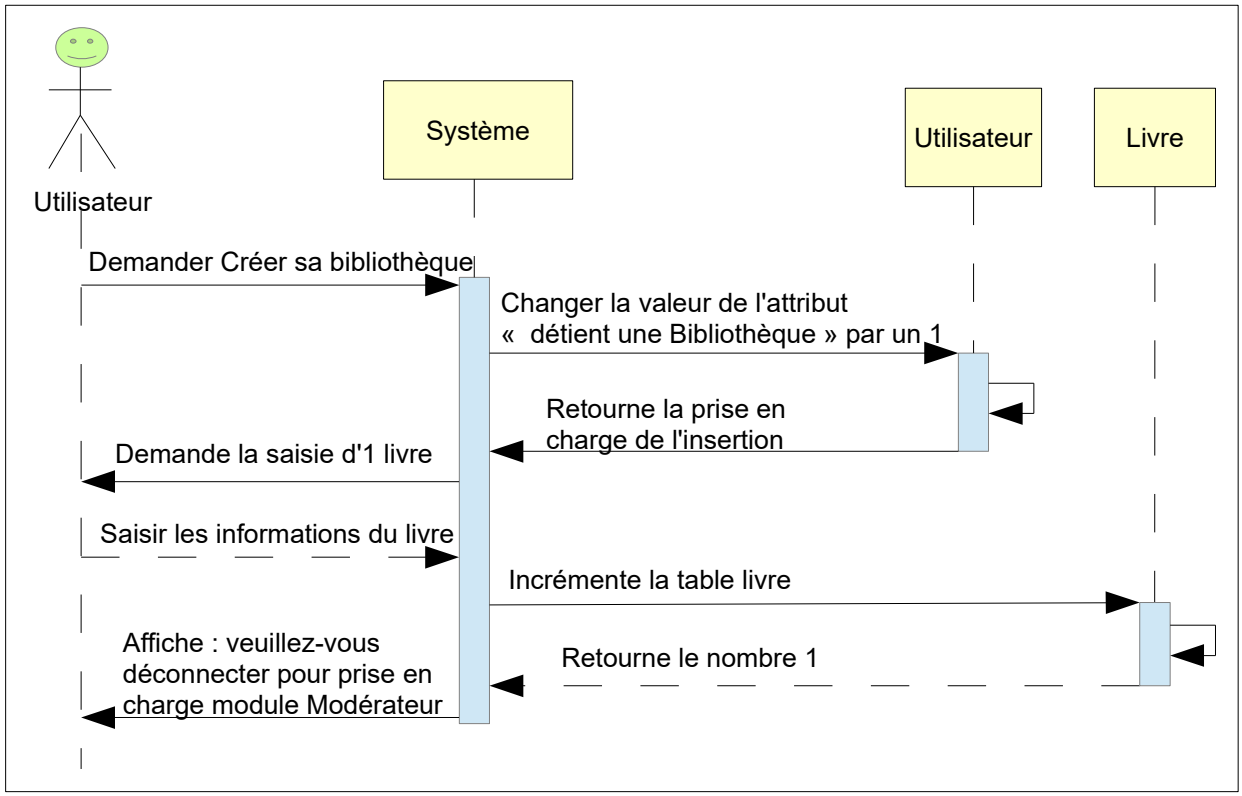
2.4.4 rechercher livres déjà emprunter



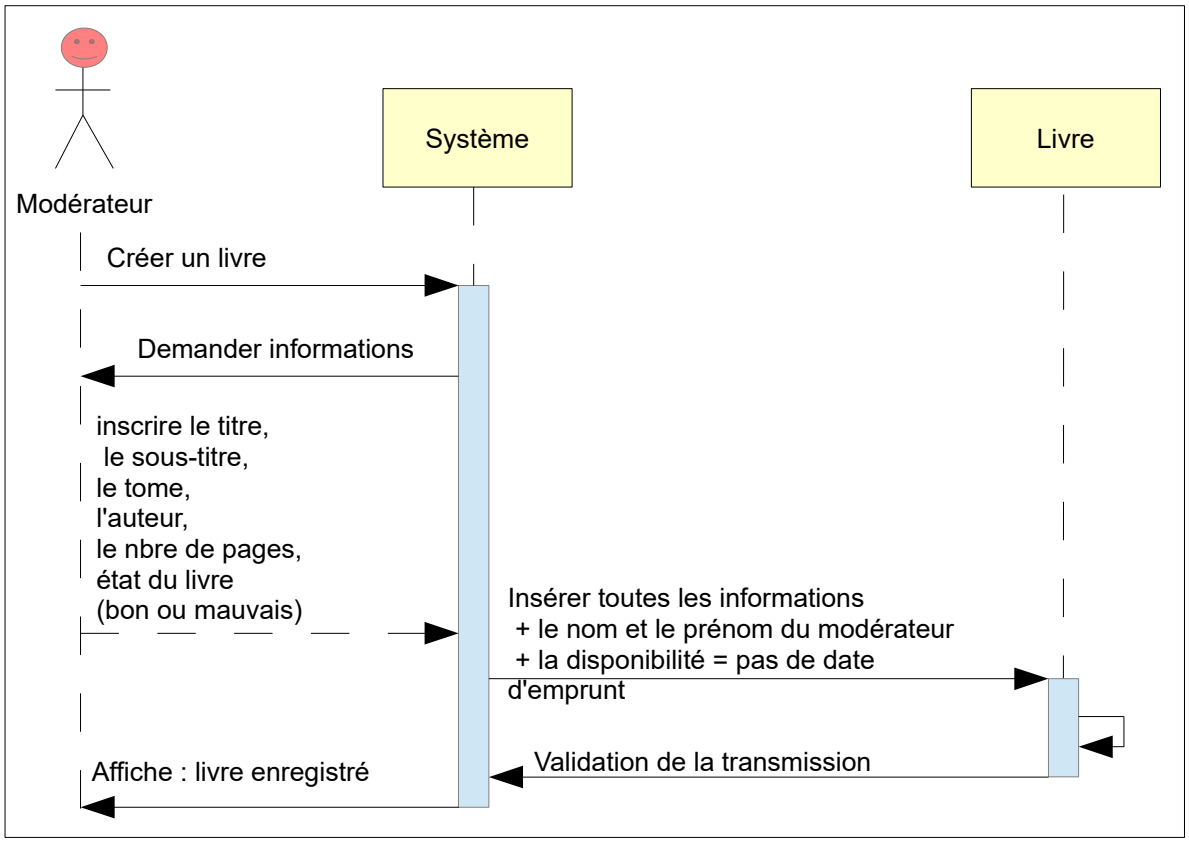
2.4.5 rechercher informations sur un livre



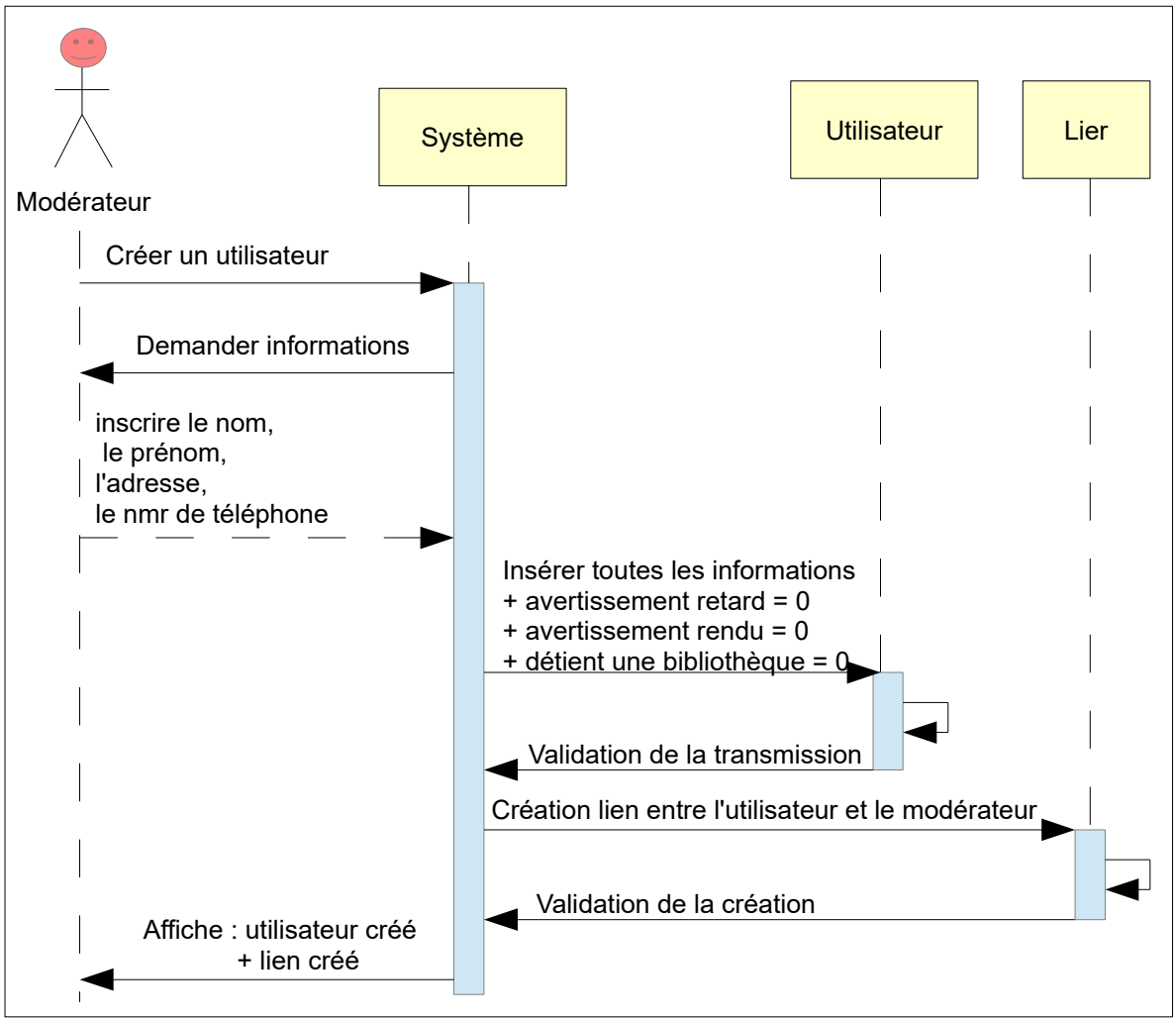
2.4.6 créer sa bibliothèque



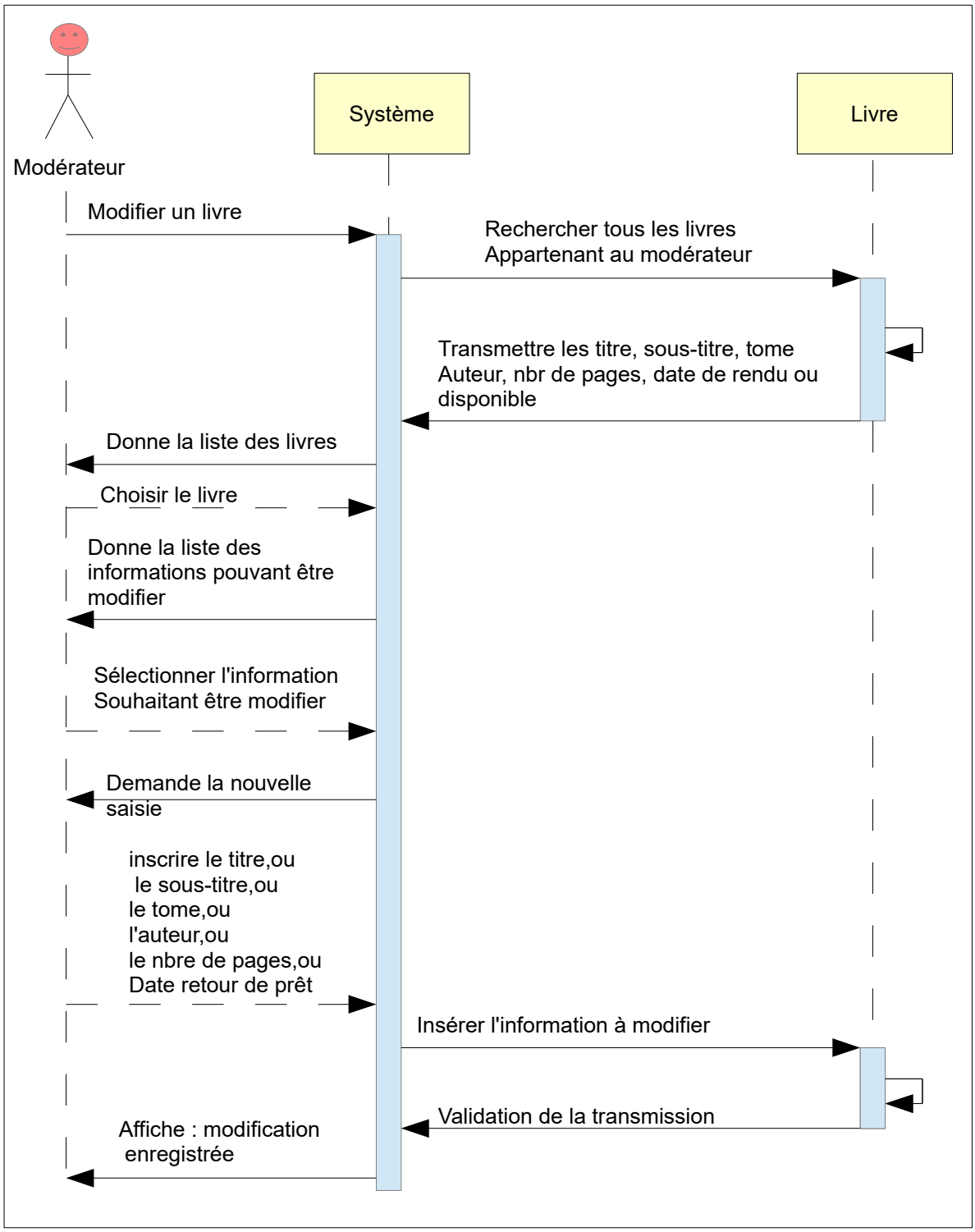
2.4.7 créer un livre



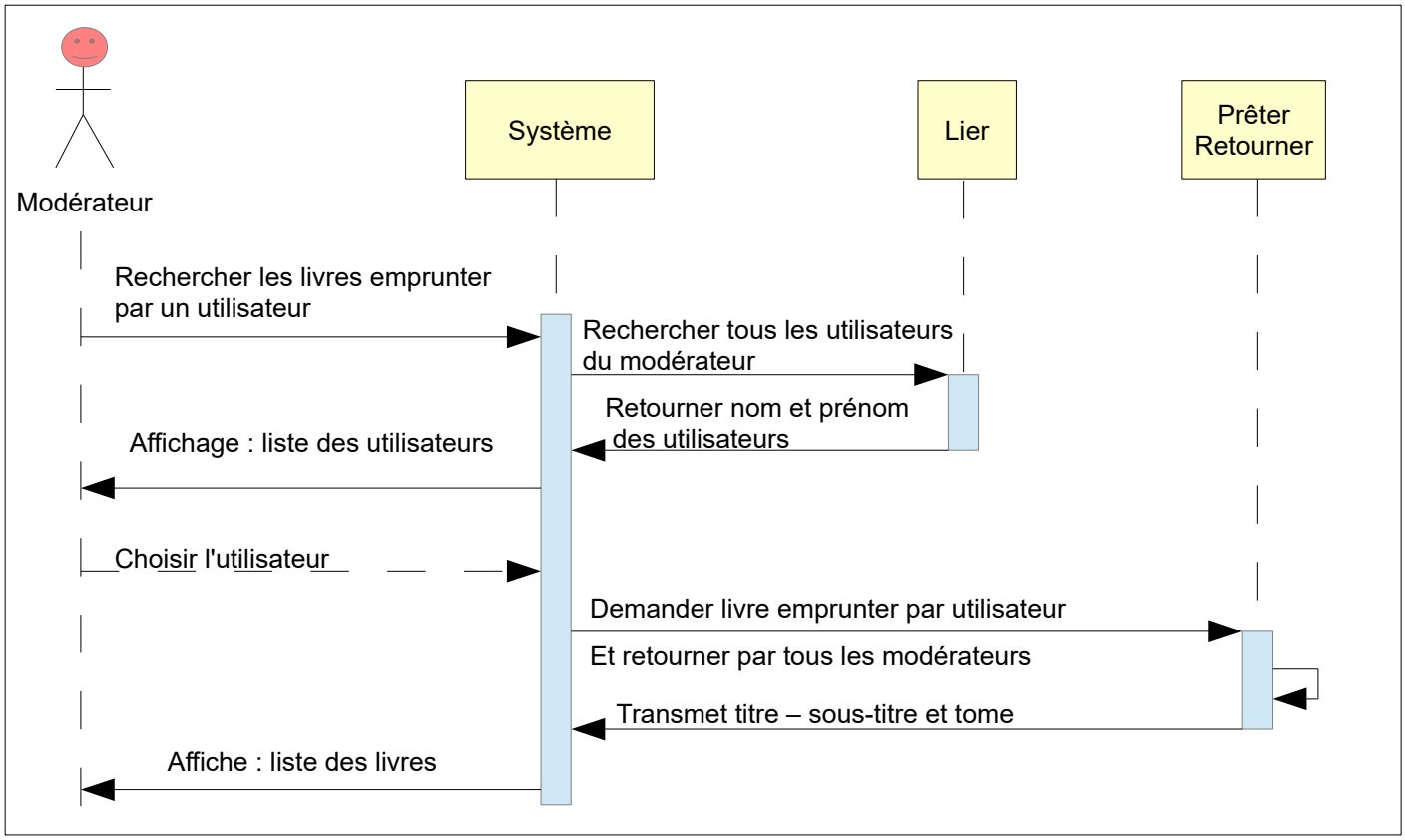
2.4.8 créer un utilisateur



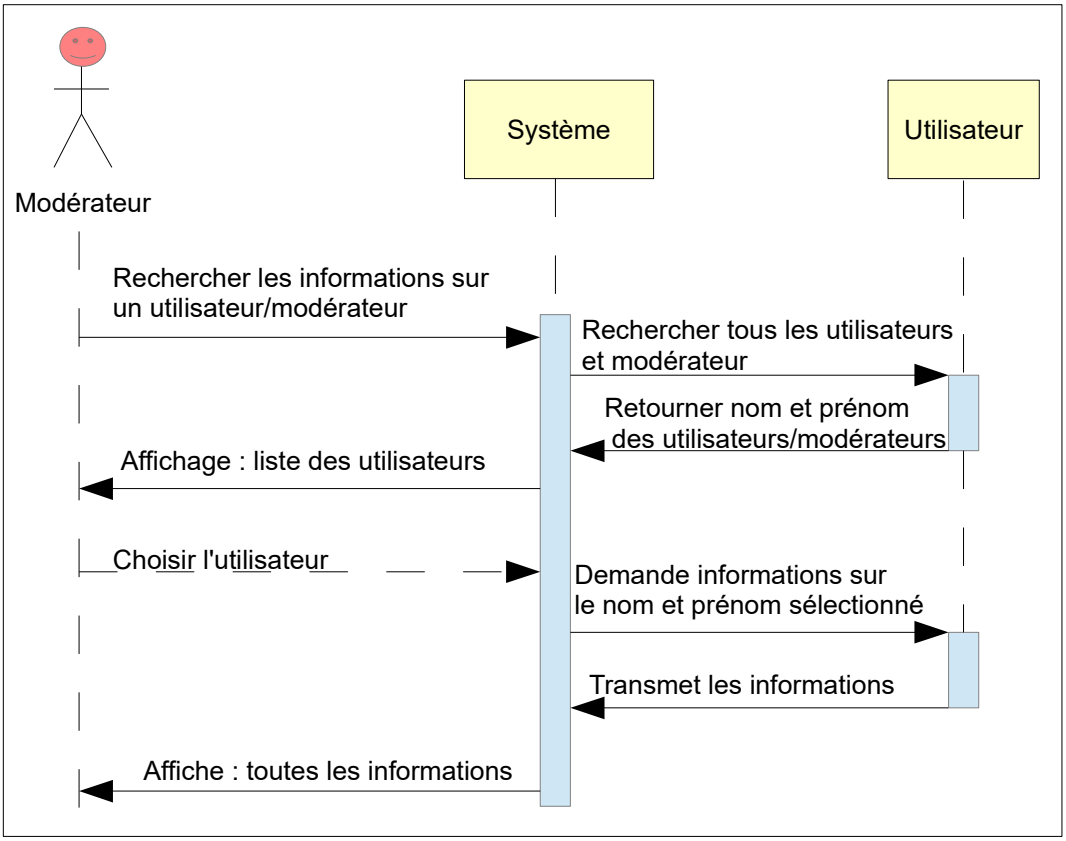
2.4.9 modifier un livre



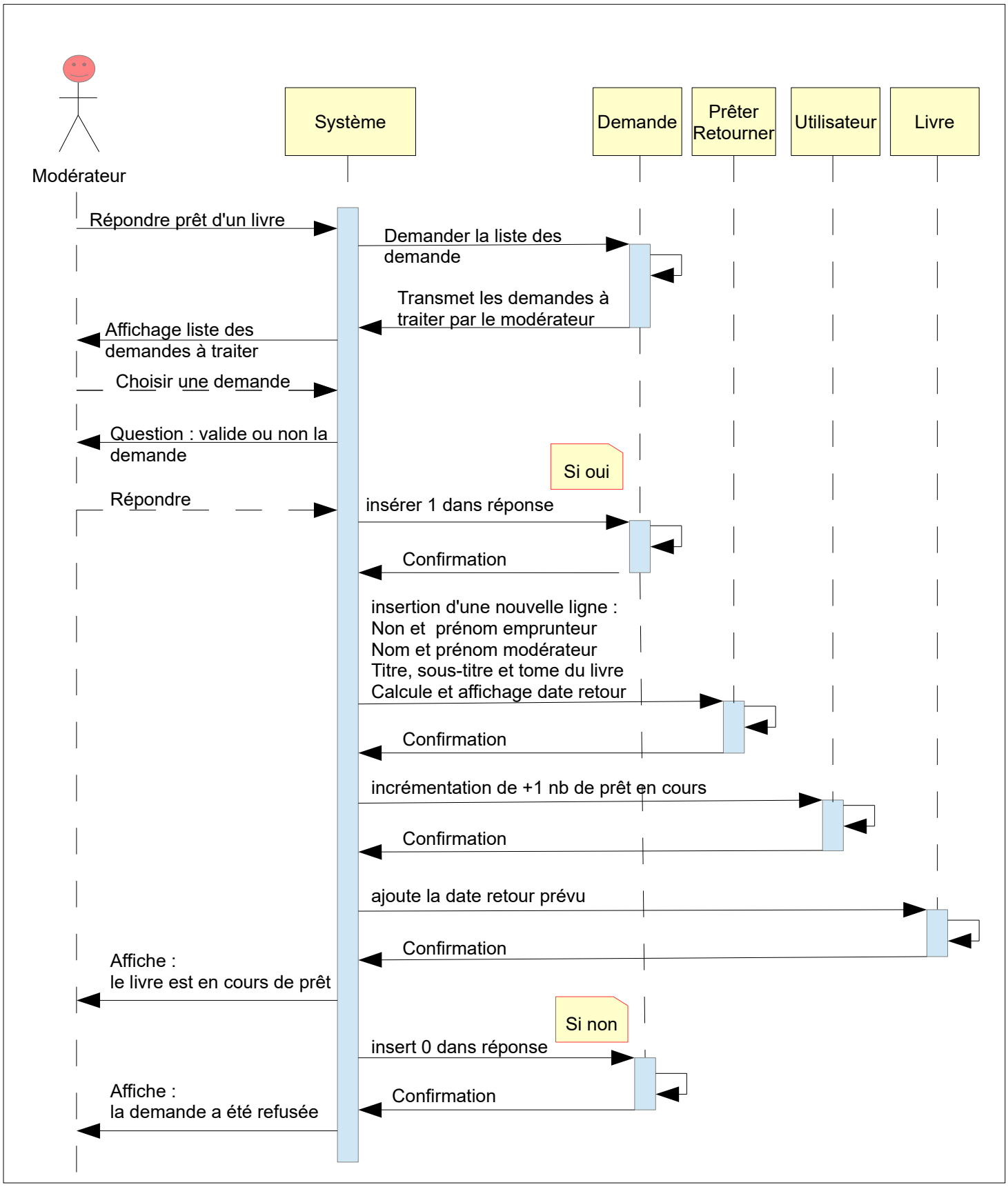
2.4.10 Rechercher la liste des livres empruntés par un de ses utilisateurs



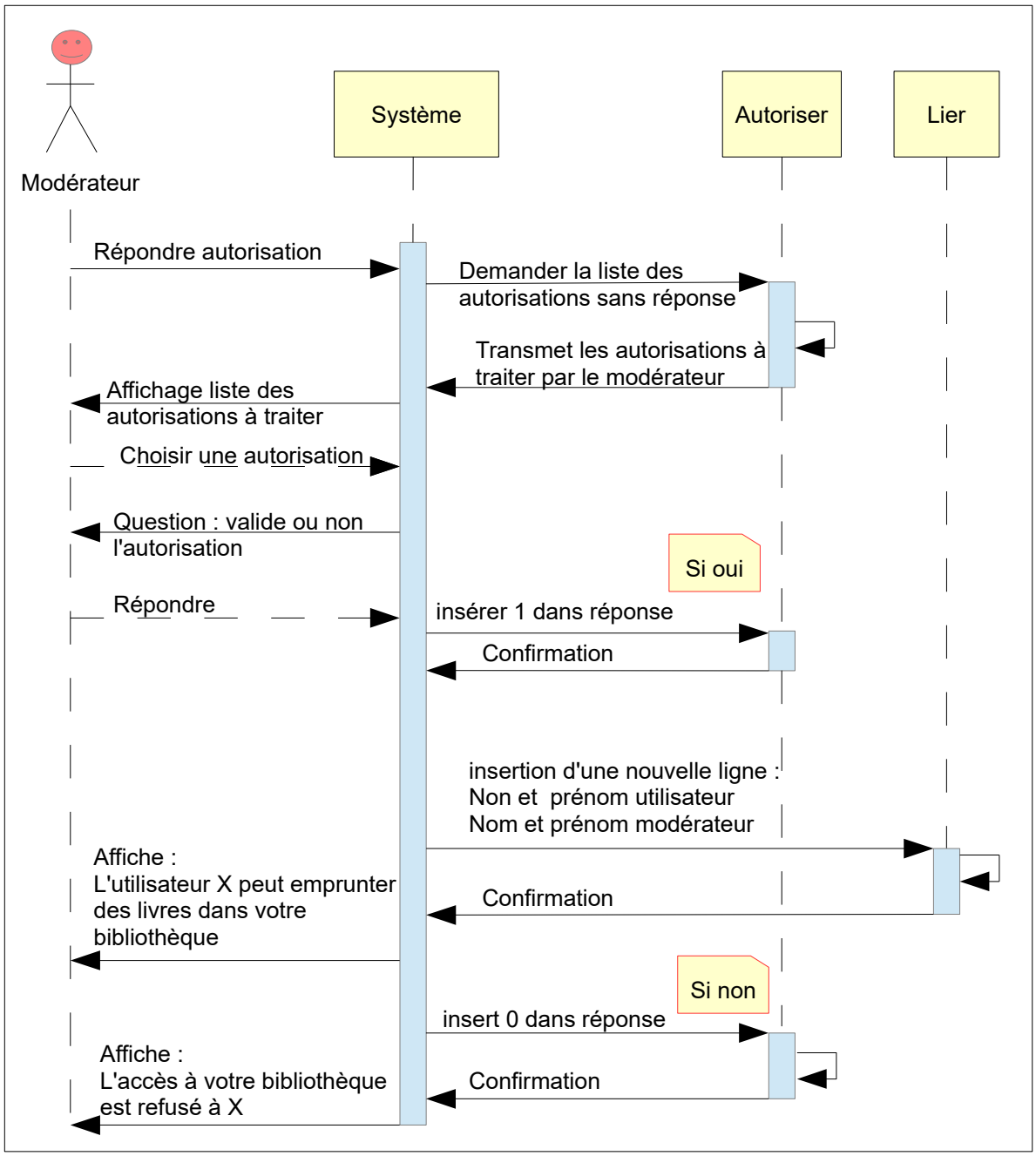
2.4.11 Rechercher informations sur un utilisateur / modérateur



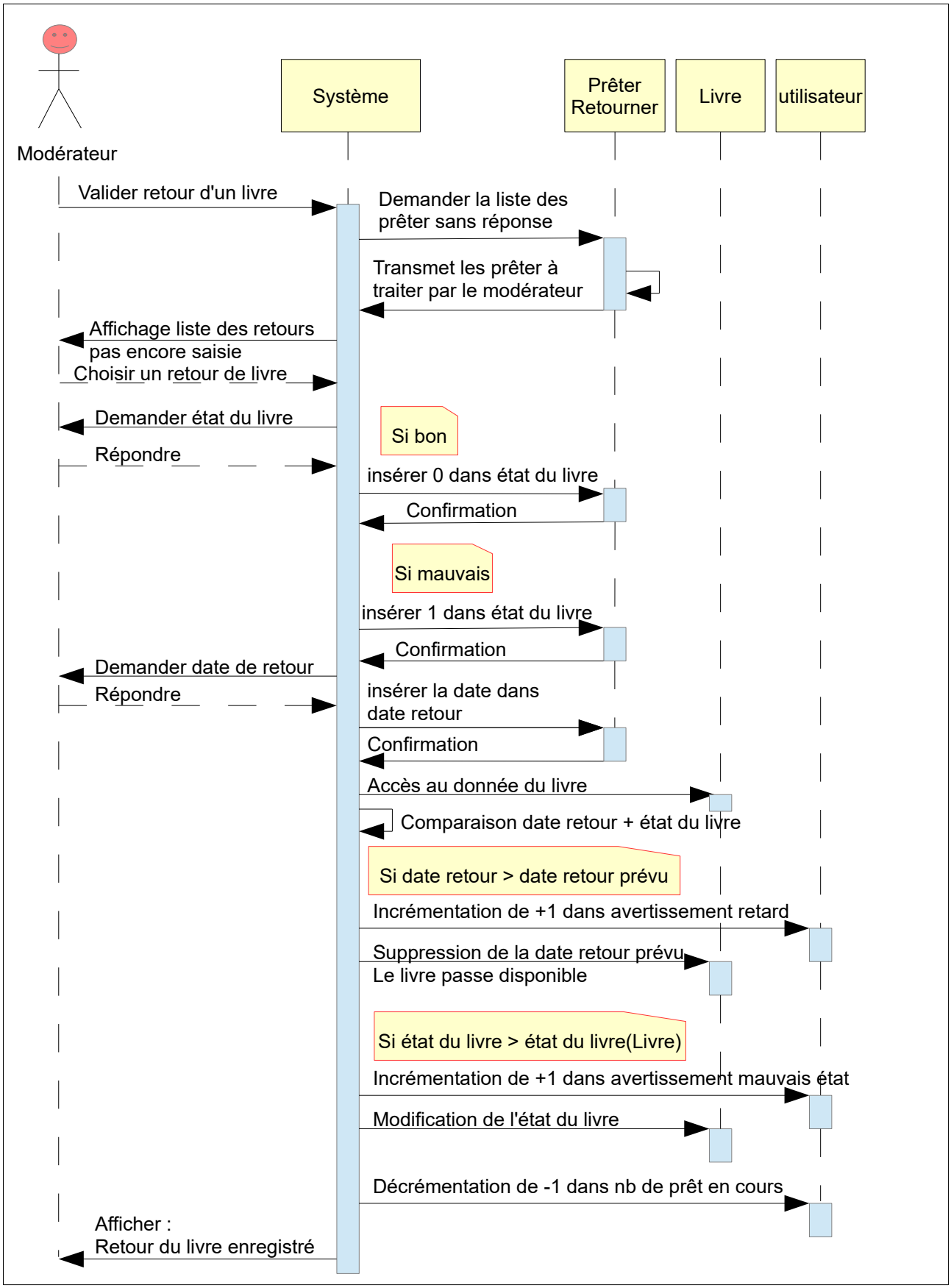
2.4.12 Répondre prêt de livre



2.4.13 Répondre autorisation d'accès



2.4.14 valider le retour d'un livre



2.5 Les avertissements

2.5.1 Les retards

Suite aux cahier des charges, nous savons que Lenaïc ne supporte pas les retards de prêt. Pour remédier à cela, il sera mis en place un contrôle sur les dates de retour des livres. Le but, maintenant, est de modéliser les sanctions dû au retard.

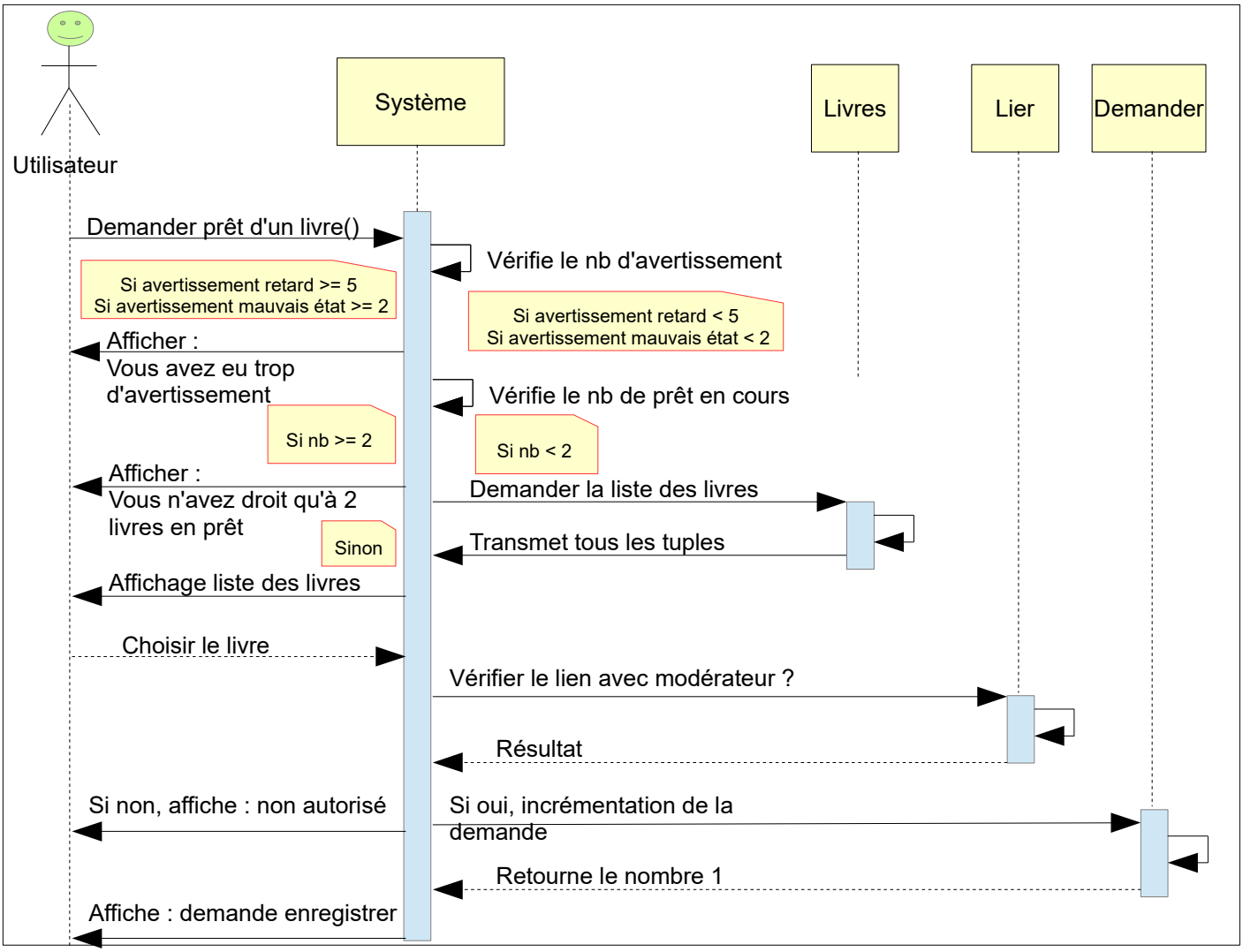
Lorsque l'emprunteur arrivera à 5 avertissements retard, il ne lui sera plus possible d'emprunter de nouveau livre. Mais, si l'emprunteur est un modérateur, ce qui implique qu'il détient une bibliothèque. On ne va pas prendre la décision de le supprimer sèchement de la base de données. Sinon, on va se retrouver à gérer des bugs surtout si des livres de chez cette personne sont en cours de prêt. La manière la plus simple, c'est de rajouter une ligne de contrôle lorsqu'un utilisateur souhaite demander le prêt d'un livre.

2.5.2 Le mauvais état du livre

Suite aux cahier des charges, nous savons que Madame Guillaume s'insurge sur l'état de ses livres prêtés. Pour remédier à cela, il sera mis en place un contrôle sur l'état des livres lors du retour. Le but, maintenant, est de modéliser les sanctions dû à la maltraitance des livres.

Lorsque l'emprunteur arrivera à 2 avertissements mauvais état, on procédera de la manière que pour les retards en rajoutant une ligne de contrôle lorsqu'un utilisateur souhaite demander le prêt d'un livre.

2.5.3 Le diagramme de séquence



3 Spécifications de l'interface

3.1 Interface Homme Machine

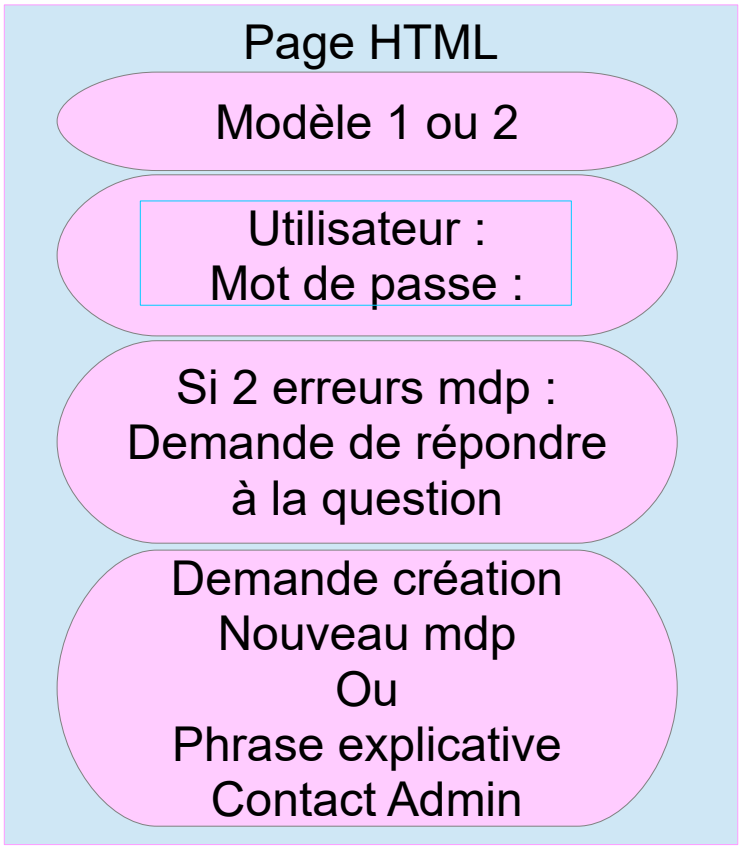
3.1.1 Le Logo

3.1.1.1 Modèle 1

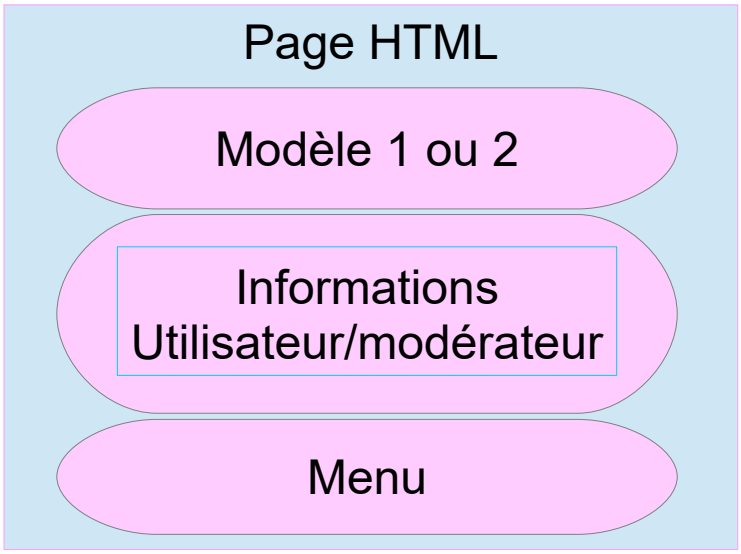
3.1.1.2 Modèle 2

3.1.2 Le prototypage

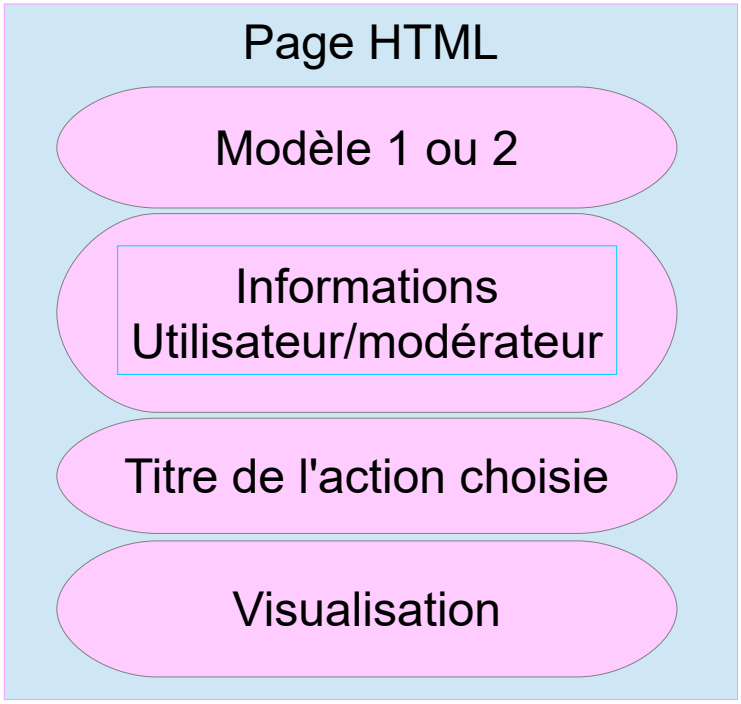
3.1.2.1 La connexion Utilisateurs ou Modérateurs



3.1.2.2 L'accès au menu



3.1.2.3 Les actions choisis du menu



3.2 Interface Logiciel / Logiciel

3.2.1 Interface d'utilisation

Java *Enterprise Edition* (Java EE) a été choisi par notre entreprise afin de faciliter le travail en équipe sur un même projet : l'application est découpée en couches, et le serveur sur lequel tourne l'application est lui-même découpé en plusieurs niveaux. Pour faire simple, Java EE fournit un ensemble d'extensions au Java standard afin de faciliter la création d'applications centralisées.

1. Pour la conception de l'application **Publio**, nous utiliserons **Eclipse *Oxygen*** pour Java EE. Lien de direction sur le site officiel : ici.
2. Pour le stockage de données, nous utiliserons **PostgreSQL** et nous créerons une base de données nommé : *Publio* avec un nouveau rôle : *child*.
3. Pour le déploiement de l'application, nous utiliserons comme serveur **GlassFish 4**. Pour ce serveur d'application JEE, il faut définir un domaine,on traduit cela par un espace de travail. Il sera nommé : *domainPublio*

3.2.2 Interface de communication

Nous utiliserons un modèle de communication standard dans la conception d'applications Java EE : **le modèle MVC (Modèle – Vue – Contrôleur)**

Il découpe littéralement l'application en couches distinctes, et de ce fait impacte très fortement l'organisation du code ! Voici dans les grandes lignes ce qu'impose MVC :

- tout ce qui concerne le traitement, le stockage et la mise à jour des données de l'application doit être contenu dans la couche nommée "Modèle" (le M de MVC) ;
- tout ce qui concerne l'interaction avec l'utilisateur et la présentation des données (mise en forme, affichage) doit être contenu dans la couche nommée "Vue" (le V de MVC) ;
- tout ce qui concerne le contrôle des actions de l'utilisateur et des données doit être contenu dans la couche nommée "Contrôle" (le C de MVC).

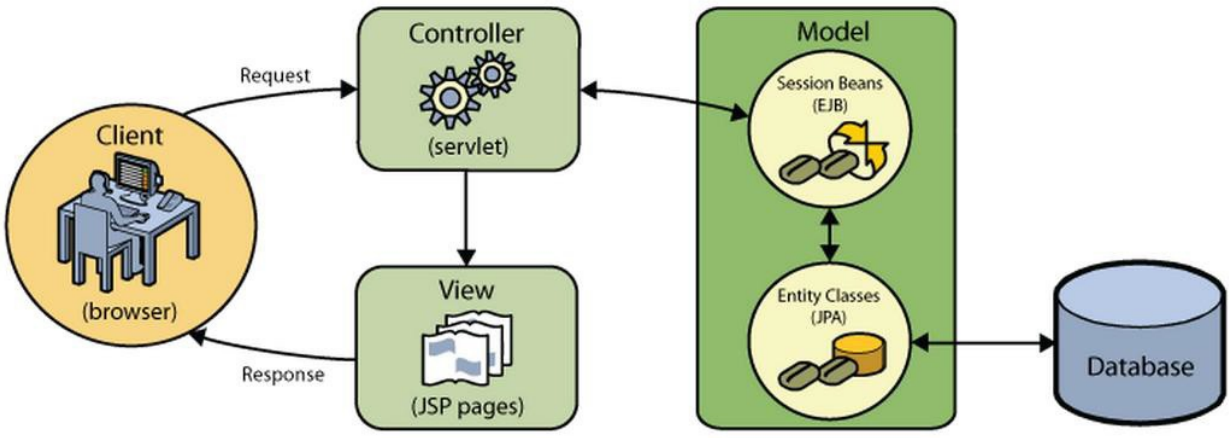


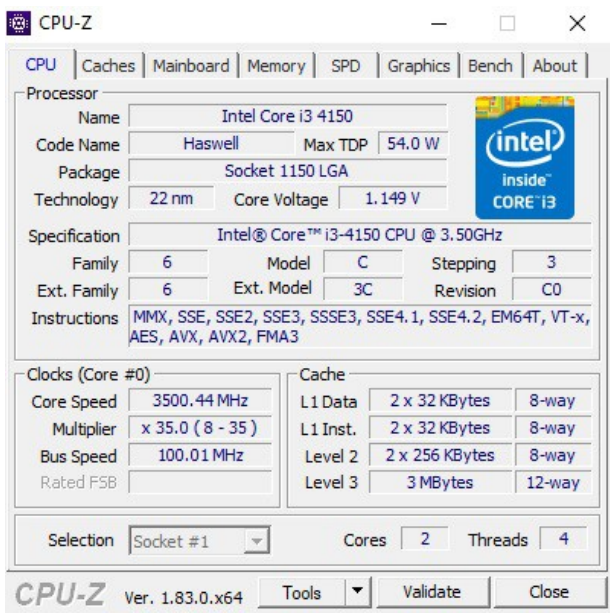
Illustration : communication interne de l'application

3.3 Interface Logiciel / Matériel

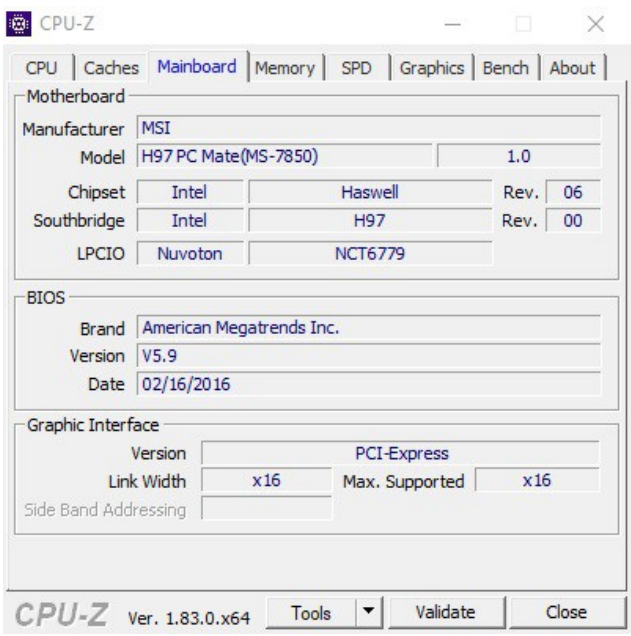
3.3.1 Ordinateur

Les caractéristiques techniques de l'ordinateur utilisé, pour la création de l'application Publio, sont les suivantes :

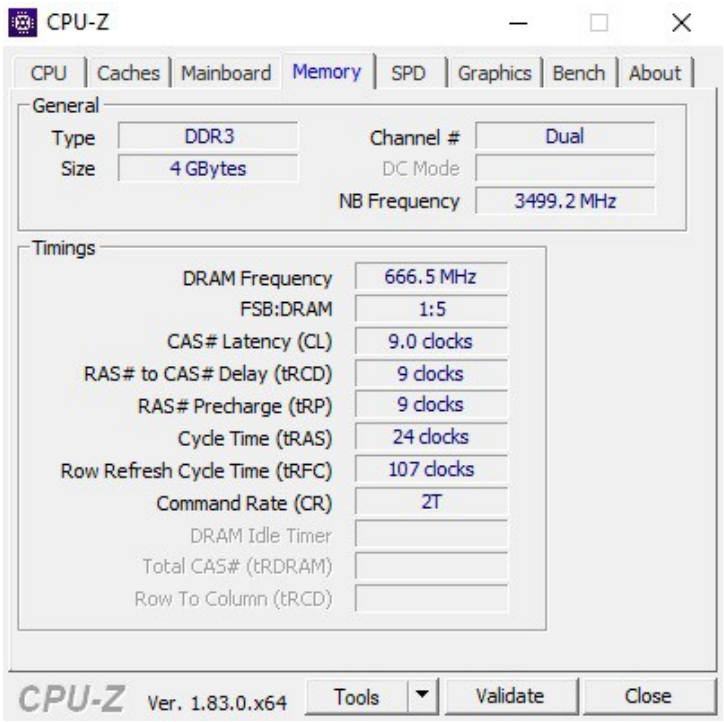
Microprocesseur



Carte Mère



Mémoire Centrale



3.3.2 Système d'exploitation

Le travail sera réalisé dans l'environnement : Windows 10 64 bit.

3.3.3 Test d'implémentation

Avant d'envoyer au tuteur l'ensemble des données de Publio, on vérifiera les instructions d'intégration dans un nouvel environnement. Pour cela, nous effectuerons un test sur une autre machine : EeePC, avec un OS sous Linux MX16 32 bit.

4 Planification

4.1 Tableau issu du cahier des charges

Type	Nom	Durée	Date de début	Date de fin
	phase 1 : Base de données	4	27/04/18	30/04/18
	Création d'un administrateur	1	27/04/18	27/04/18
	Création nom de la base de données	1	28/04/18	28/04/18
	Création des tables	1	29/04/18	29/04/18
	Création des droits sur les tables	1	30/04/18	30/04/18
	phase 2 : vues utilisateur+connexion	4	01/05/18	04/05/18
	Création des vues utilisateur, modérateur(§4.2)	1	01/05/18	01/05/18
	Création des vues utilisateur, modérateur(§4.2)	1	02/05/18	02/05/18
	Tests unitaires (connexion - affichage vues)	1	03/05/18	03/05/18
	Essai avec Lenaïc	0	04/05/18	04/05/18
	Rédaction manuel utilisateur partie 1 + Dossier Technique	1	04/05/18	04/05/18
	phase 3 : les sorties	13	05/05/18	17/05/18
	Incrémentation des tables pour valider les tests	1	05/05/18	05/05/18
	Création des vues de sortie (§5.1.3)	4	06/05/18	09/05/18
	Lier les actions visualiser (§4.2.2) avec les vues (§5.1.3)	4	10/05/18	13/05/18
	Tests unitaires de l'apparence pour les sorties	2	14/05/18	15/05/18
	Essai avec Yvain	0	16/05/18	16/05/18
	Rédaction manuel utilisateur partie 2 + Dossier Technique	2	16/05/18	17/05/18
	phase 4 : les interrogations	8	18/05/18	25/05/18
	Création de fonctions PL* (les avertissements)	1	18/05/18	18/05/18
	Création des vues d'interrogation (§5.1.4)	2	19/05/18	20/05/18
	Lier les actions rechercher (§4.2.2) avec les vues	2	21/05/18	22/05/18
	Tests unitaires sur le rendu d'une recherche	1	23/05/18	23/05/18
	Essai de Madame GUILLAUME	0	24/05/18	24/05/18
	Rédaction manuel utilisateur partie 3	2	24/05/18	25/05/18
	phase 5 : les entrées	20	26/05/18	14/06/18
	Création des vues de saisie pour les entrées (§5.1.2)	6	26/05/18	31/05/18
	Lier les actions (§4.2.2) avec les vues (§5.1.2)	6	01/06/18	06/06/18
	Tests unitaires sur la saisir des entrées	4	07/06/18	10/06/18
	Vérification des saisies d'entrée venant de l'API	1	11/06/18	11/06/18
	Essai de la famille	0	12/06/18	12/06/18
	Rédaction manuel utilisateur partie 4 + Dossier Technique	3	12/06/18	14/06/18

4.2 Observations

La planification des tâches est claire, le respect des délais est la charge de MOE et du développeur. Au moindre retard, le MOE doit immédiatement avertir le MOA ; car celui-ci doit planifier les rendez-vous d'essai avec les différents membres de la famille.

5 Conclusion

L'objectif de ce projet était de modéliser le système d'information Publio, selon les préceptes de la « méthode » d'analyse UML.

Les différentes tâches fixées ont été réalisées à partir de plusieurs hypothèses. Nous avons modélisé les opérations importantes en respectant les contraintes fixées afin de les conformer les unes aux autres et d'avoir une vision au plus juste possible. Les diagrammes de séquences, cas d'utilisations, diagramme de classes font partie de notre analyse.

Suite réunion entre le MOA de la famille et le MOE de l'entreprise, le modèle 2 a été choisi pour le logo.