Perroquet

Franck AUDIGER, Ahmed AISSA, Valérian DAMM

I. Introduction

- Description
  - Le projet Perroquet est un logiciel client lourd (c’est-à-dire installé sur la machine). Qui permet de gérer des cours en ligne, les enseignants peuvent déposer un cours et les étudiants peuvent en suivre.
- Contexte
  - Ce projet est réalisé dans le cadre de la formation de la Licence Informatique de l'Université de lorraine. Ce projet fait partie du cours de Conception et Programmation Objet Avancée.

II. Expression du besoin

- Fonction
  - Proposer un cours
    - être un enseignant
  - Suivre un cours
    - être un étudiant inscrit dans le cours
  - Ouvrir inscription a un cours
    - être un enseignant
    - le cours doit avoir été accepté par l'administrateur
  - Accepter/refuser un cours
    - être un administrateur
  - Inscription d'un étudiant à un cours
    - être un étudiant
    - nombre de place du cours est limité
    - système de file d'attente
    - l’étudiant ne doit pas avoir atteint le nombre maximum de cours
  - Compléter cours avec des ressources

        - être un enseignant
    - Remettre un travail
        - être un étudiant
    - Ouvrir un dépôt
        - être un enseignant
        - doit avoir une date de dépôt
    - Noter un devoir
        - être enseignant
    - Retirer son inscription
        - être étudiant
        - désinscription juste pendant la période d'inscription
    - Publier des notes
        - être enseignant
    - Voir notes
        - être étudiant
        - les notes sont celle de l'étudiant
    - Création de compte
        - être étudiant ou enseignant
        - doit être accepté par l'administrateur pour être valide
    - Accepter/refuser compte
        - être administrateur

III. Solution au besoin (scénario)

- Se connecter
    - Un utilisateur (administrateur, enseignant ou étudiant) peut se connecter via un identifiant et un mot de passe.
    - L'utilisateur doit attendre la validation du compte par l'administrateur. Le

système affichera donc à l'utilisateur si son compte est en attente.

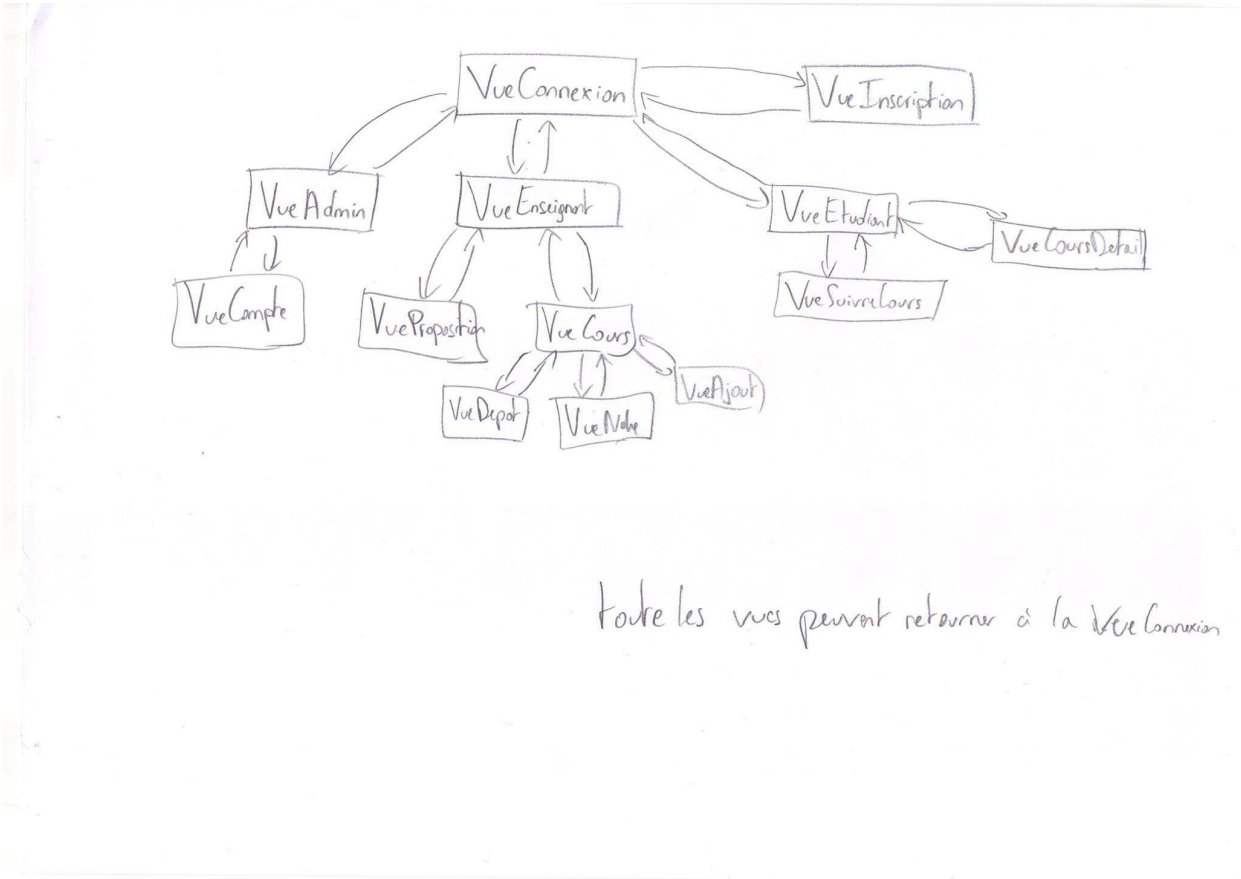
- Un utilisateur (enseignant ou étudiant) peut faire une demande de création de compte avec un formulaire.

- Connexion en tant qu'étudiant.
  - Vue de tous les cours suivis par celui-ci.
  - Il peut s'inscrire/se désinscrire d'un cours tant que l'inscription à ce cours est possible.
  - En cliquant sur l'intitulé du cours, l'étudiant accède aux ressources, devoirs ou dépôts proposés par l'enseignant.
  - L'étudiant peut demander à voir tous les cours proposés et ouverts par les différents enseignants si le nombre de places maximum dans ce cours n'est pas atteint.
- Connexion en tant qu'enseignant
  - Vue sur tous les cours enseignés.
  - L'enseignant peut ouvrir ou fermer l'inscription du cours aux étudiants.
  - Via des boutons, l'enseignant peut ajouter une ressource, un dépôt, un devoir d'un cours. Le système prévient l'utilisateur (enseignant) de la réussite de l'opération.
  - En cliquant sur l'intitulé d'un cours, l'enseignant accède à une vue où toutes les ressources du cours seront présentées. Il pourra effectuer les mêmes opération que précédemment via un menu ainsi qu'ouvrir/fermer un dépôt, noter ou même publier les notes d'un devoir et retirer une ressource, un devoir ou un dépôt.
  - L'enseignant peut proposer un nouveau cours et ainsi remplir un formulaire concernant le cours qui devra être accepter par l'administrateur. Le système prévient l'utilisateur (enseignant) de la réussite de l'opération.
- Connexion en tant qu'administrateur
  - Vue sur toutes les propositions de cours (avec bouton accepter/refuser)
  - Le système supprime la demande de la liste et envoie une notification à l'enseignant. Le système prévient l'utilisateur (administrateur) de la réussite de l'opération.

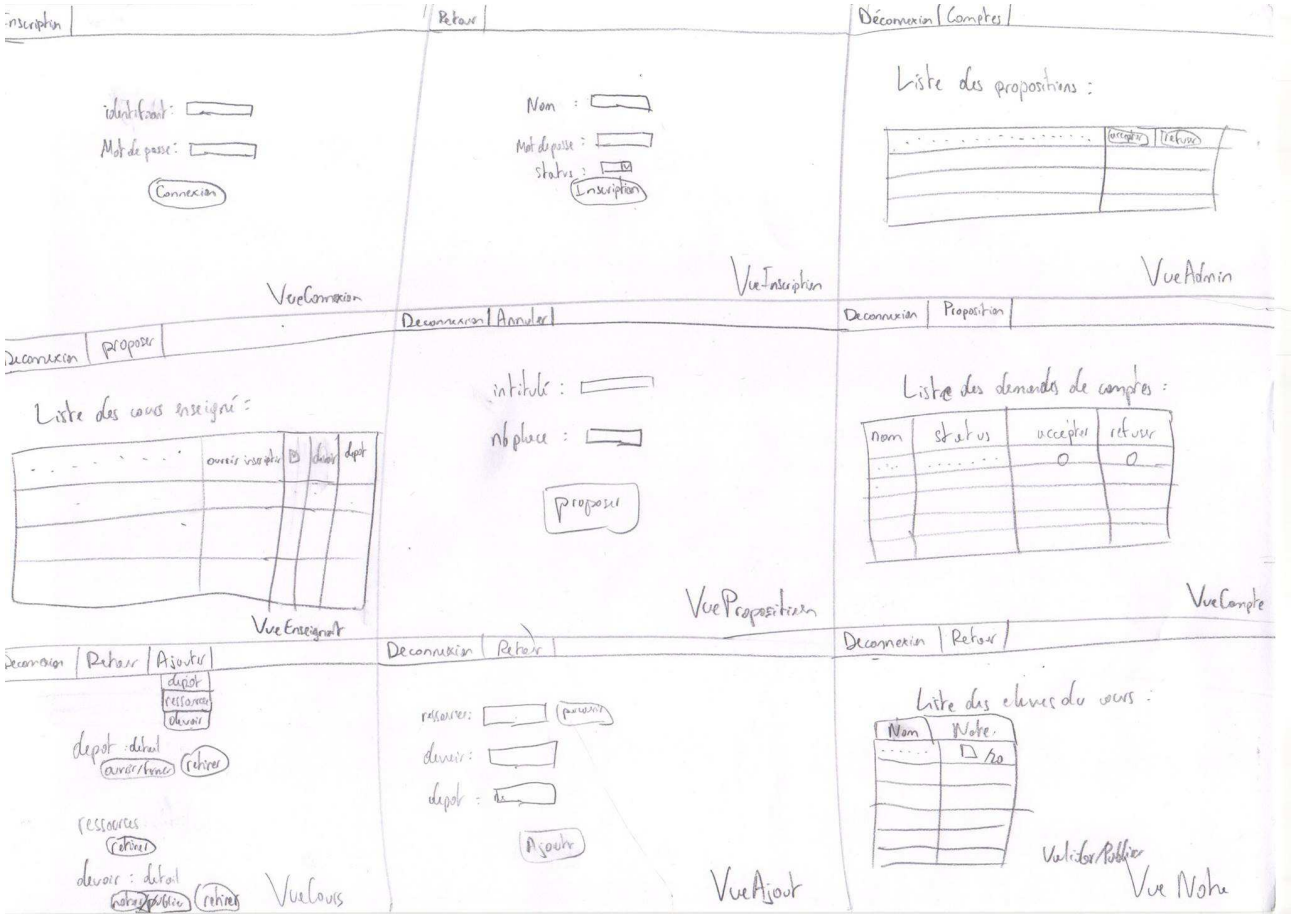
- L'administrateur peut accéder, via un bouton à la liste des demandes des comptes en attente. En cas de refus, le système crée un compte refusé, sinon crée un nouveau compte. Le système prévient l'utilisateur (administrateur) de la réussite de l'opération.

IV. Diagramme cinématique

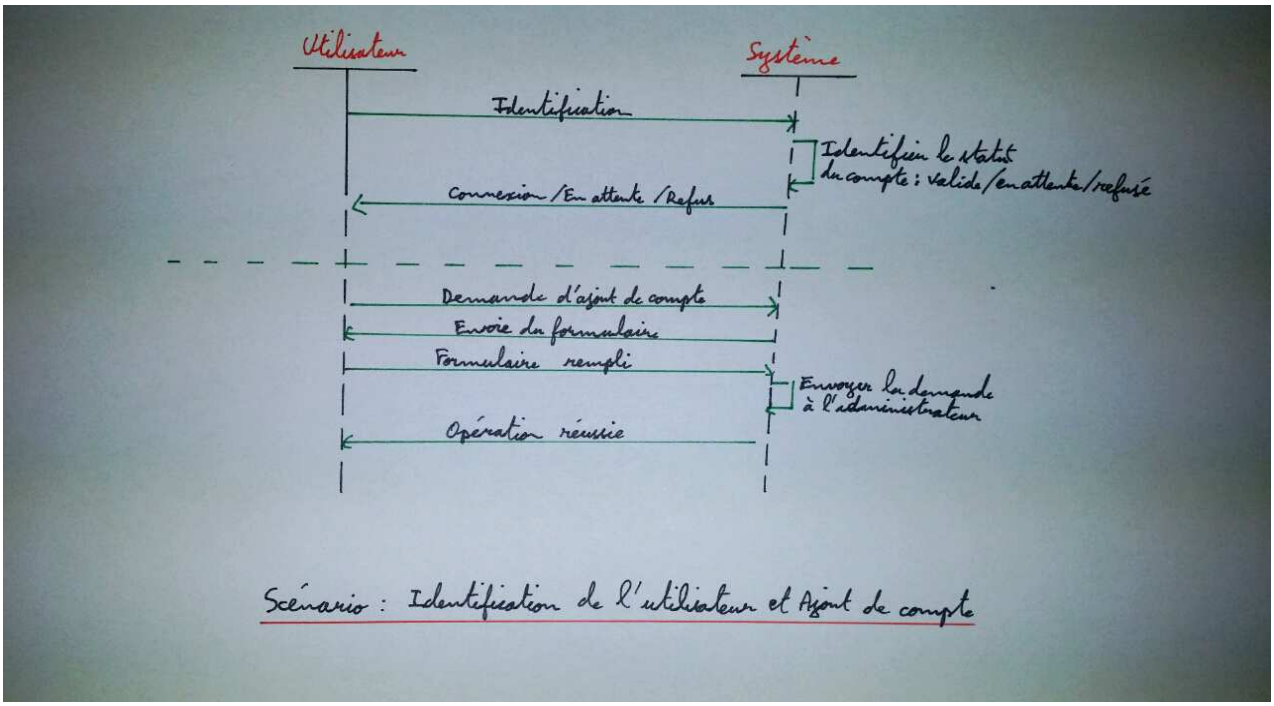
- cinématique



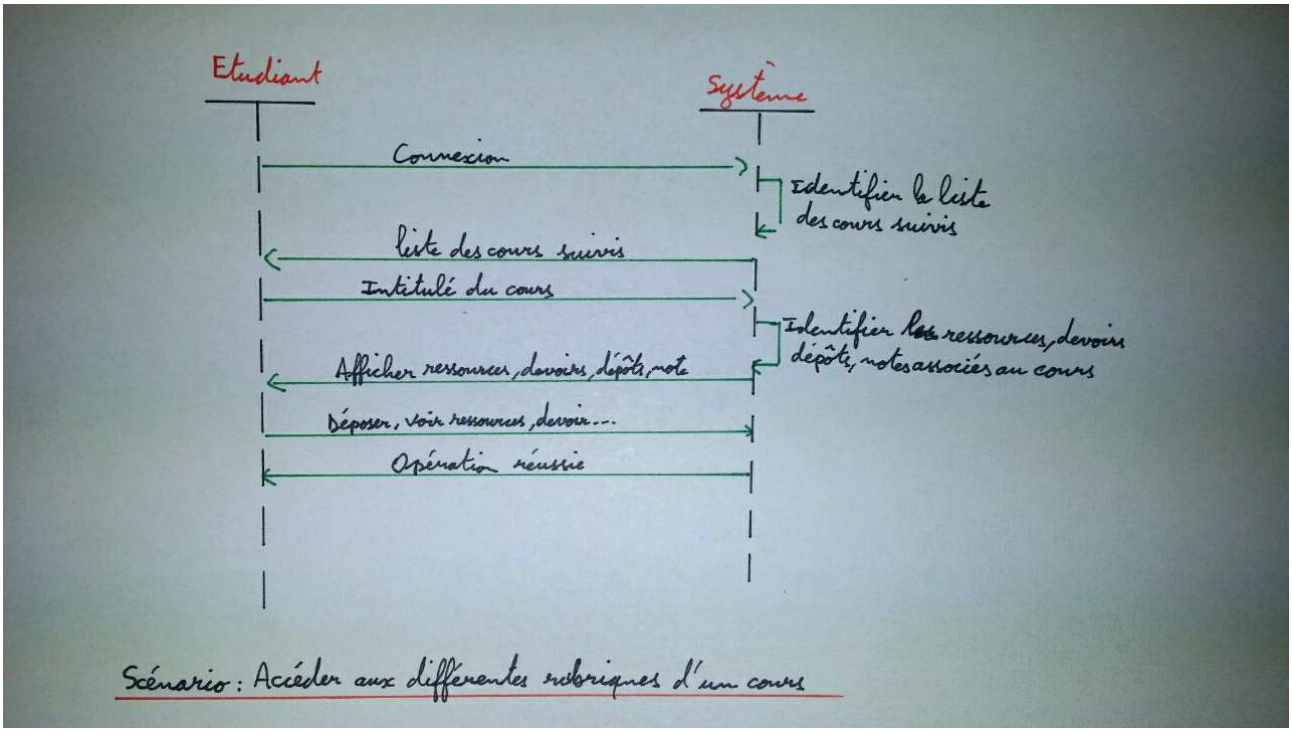
- vues


Inscription
identifiant :
Mot de passe :
Connexion
VueConnexion
Retour
Nom :
Mot de passe :
status :
Inscription
VueInscription
Déconnexion | Comptes
Liste des propositions :
accepter
refuser
VueAdmin
Déconnexion | proposer
Liste des cours enseigné :
ouvrir inscription
dépot
VueEnseignant
Déconnexion | Annuler
intitulé :
nb place :
proposer
VueProposition
Déconnexion | Proposition
Liste des demandes de comptes :
nom
status
accepter
refuser
VueCompte
Déconnexion | Retour | Ajouter
dépot
ressource
devoir
dépot : détail
ouvrir/fermer
retirer
ressources
retirer
devoir : détail
noter/publier
retirer
VueCours
Déconnexion | Retour
ressource:
parcourir
devoir :
dépot :
Ajouter
VueAjout
Déconnexion | Retour
Liste des eleves du cours :
Nom
Note
/20
Valider/Publier
Vue Note
Déconnexion | Retour
état : ouvrir/fermer
Liste des dépot :
Nom
Fichier
VueDepot
Déconnexion | Propositions
Cours suivie :
Intitulé
désinscription
VueEtudiant
Déconnexion | Retour
Proposition de cours :
Intitulé
Suivre
VueSuivreCours
Déconnexion | Retour
dépot : intitulé
déposer
ressource : intitulé
voir
devoir : intitulé
(la note si publié)
VueCoursDetail

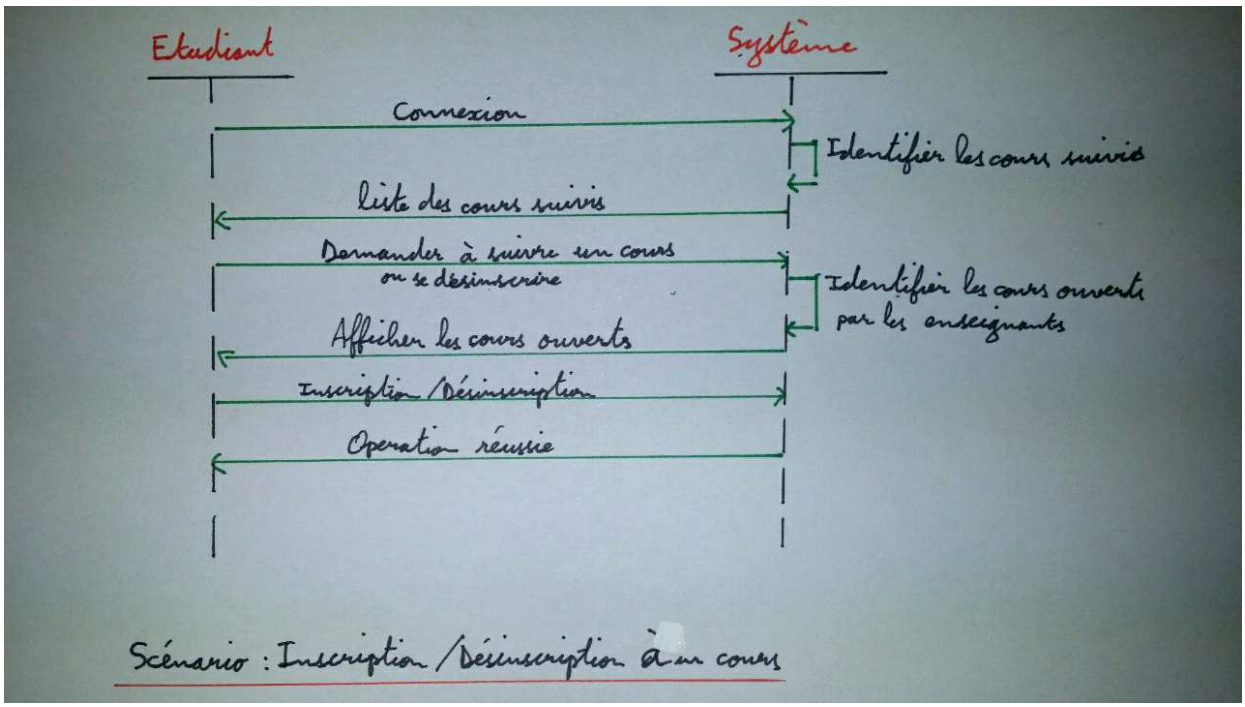
v. Diagramme de séquence



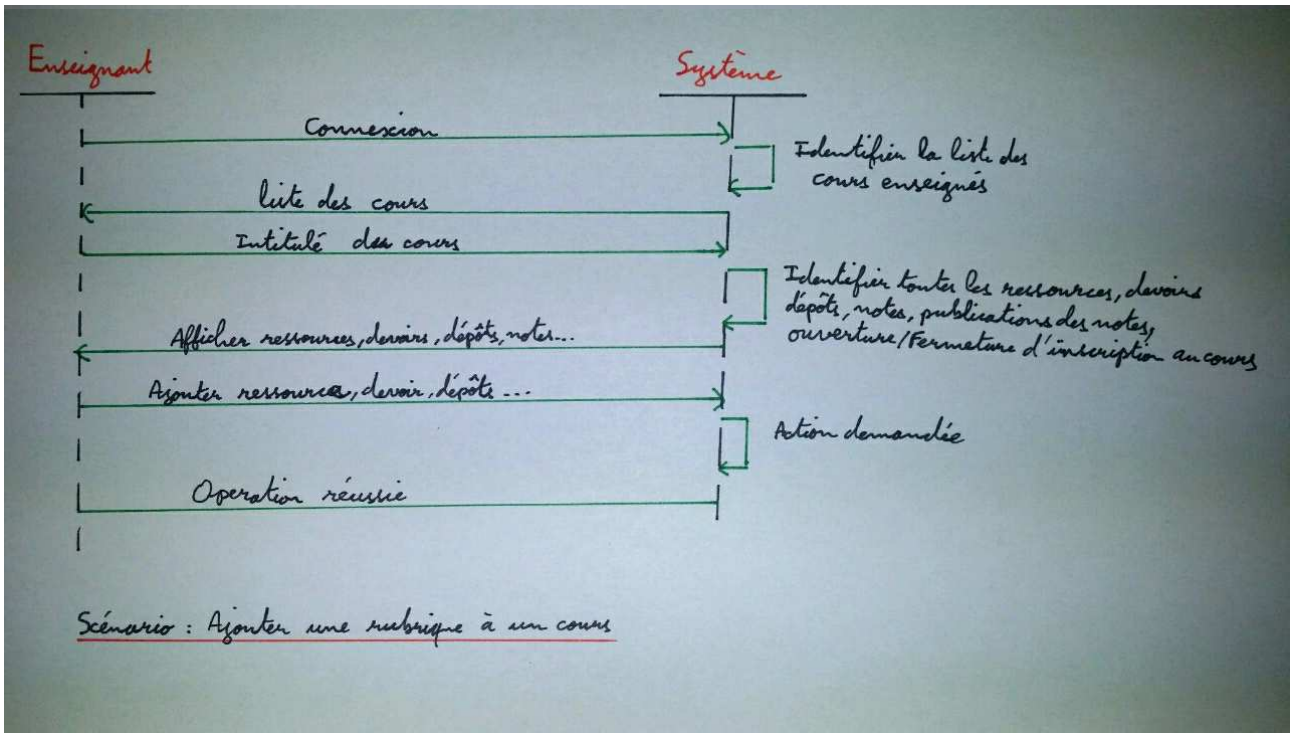
Scénario : Identification de l'utilisateur et Ajout de compte



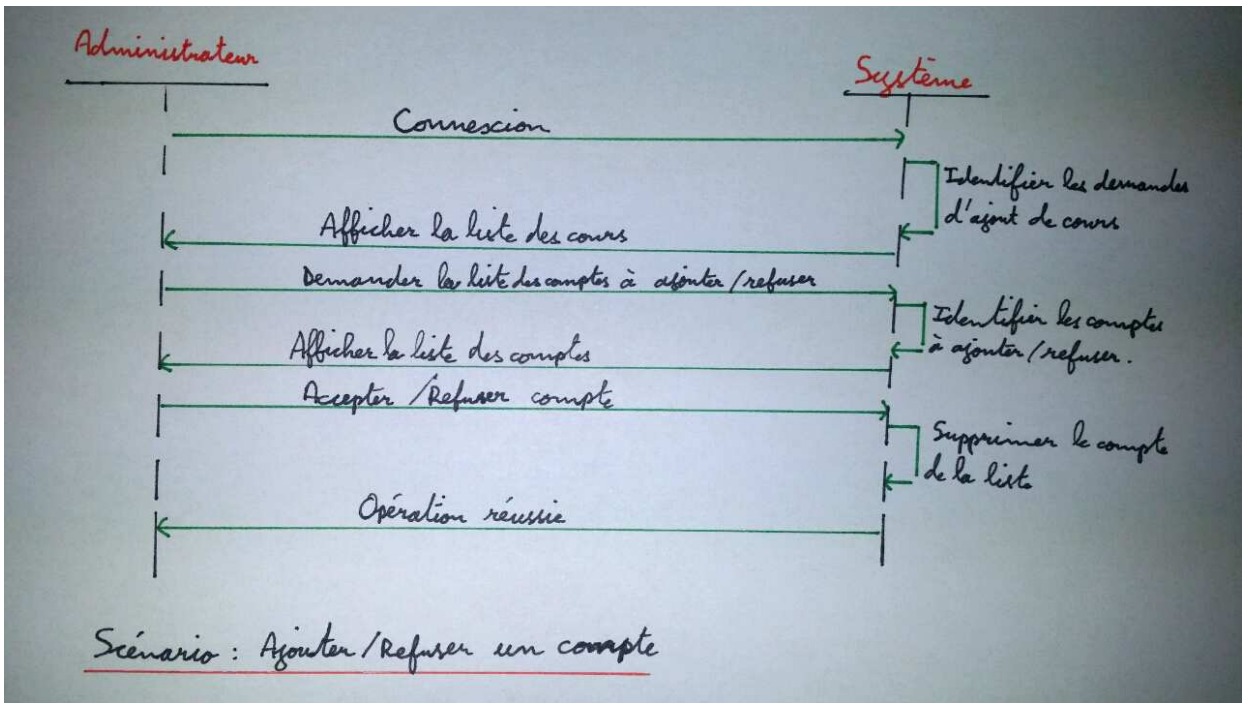
Scénario : Accéder aux différentes rubriques d'un cours



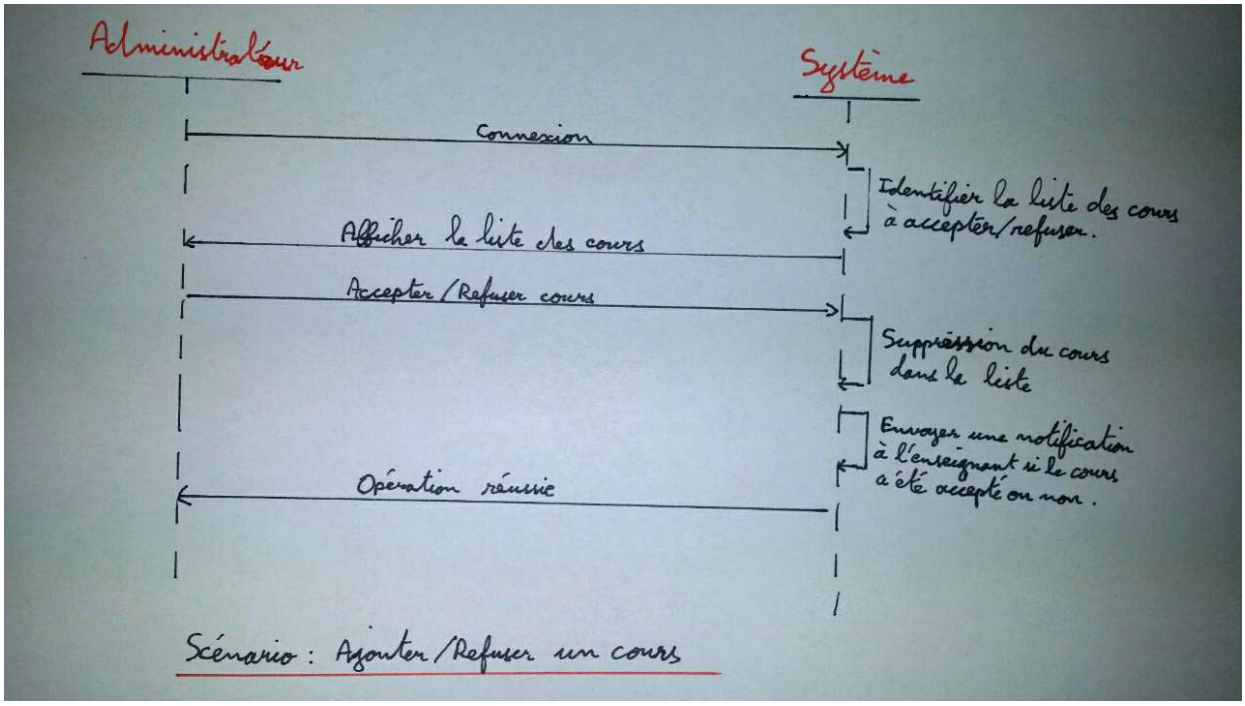
Scénario : Inscription / Désinscription à un cours



Scénario : Ajouter une rubrique à un cours

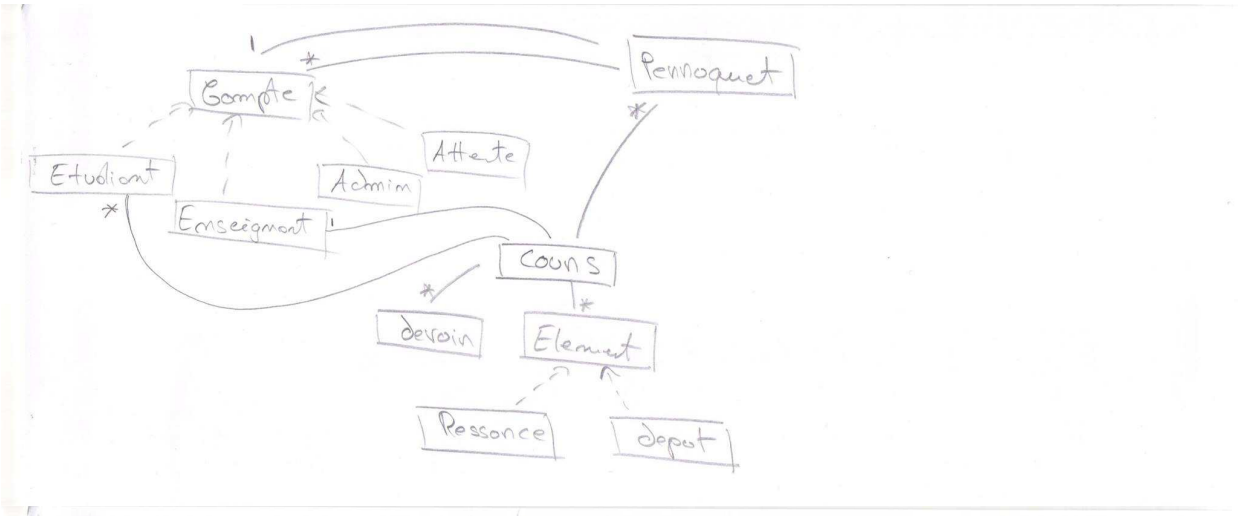


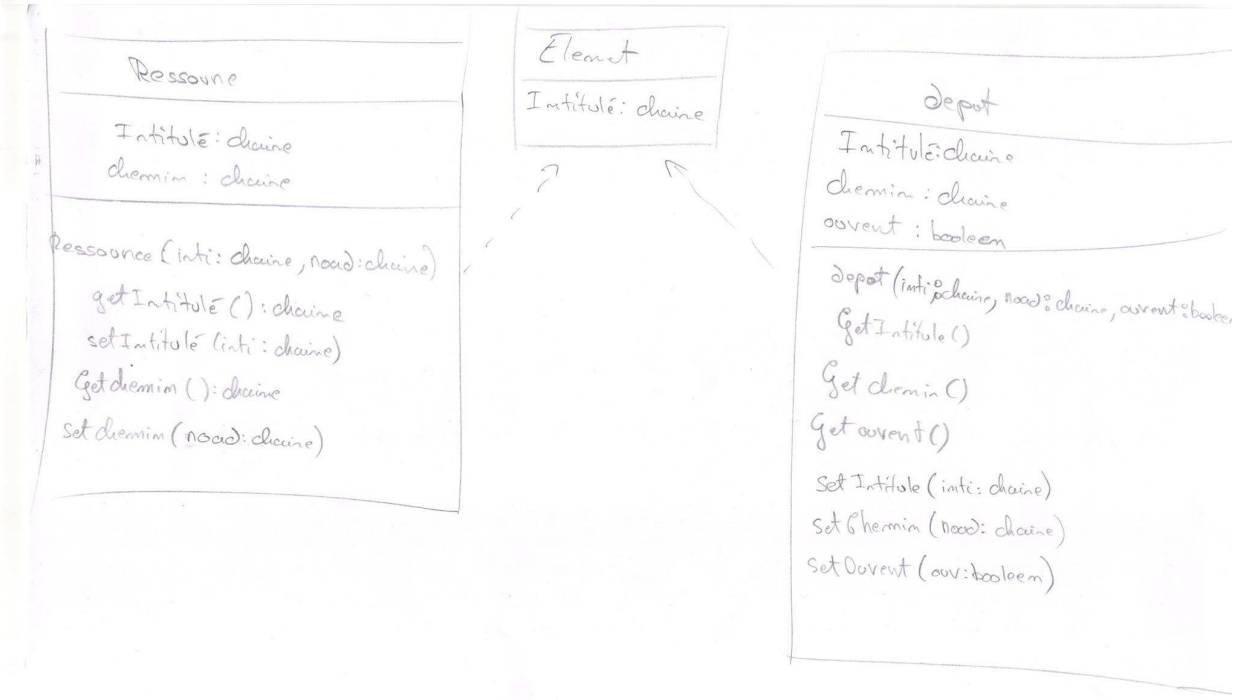
Scénario : Ajouter/Refuser un compte

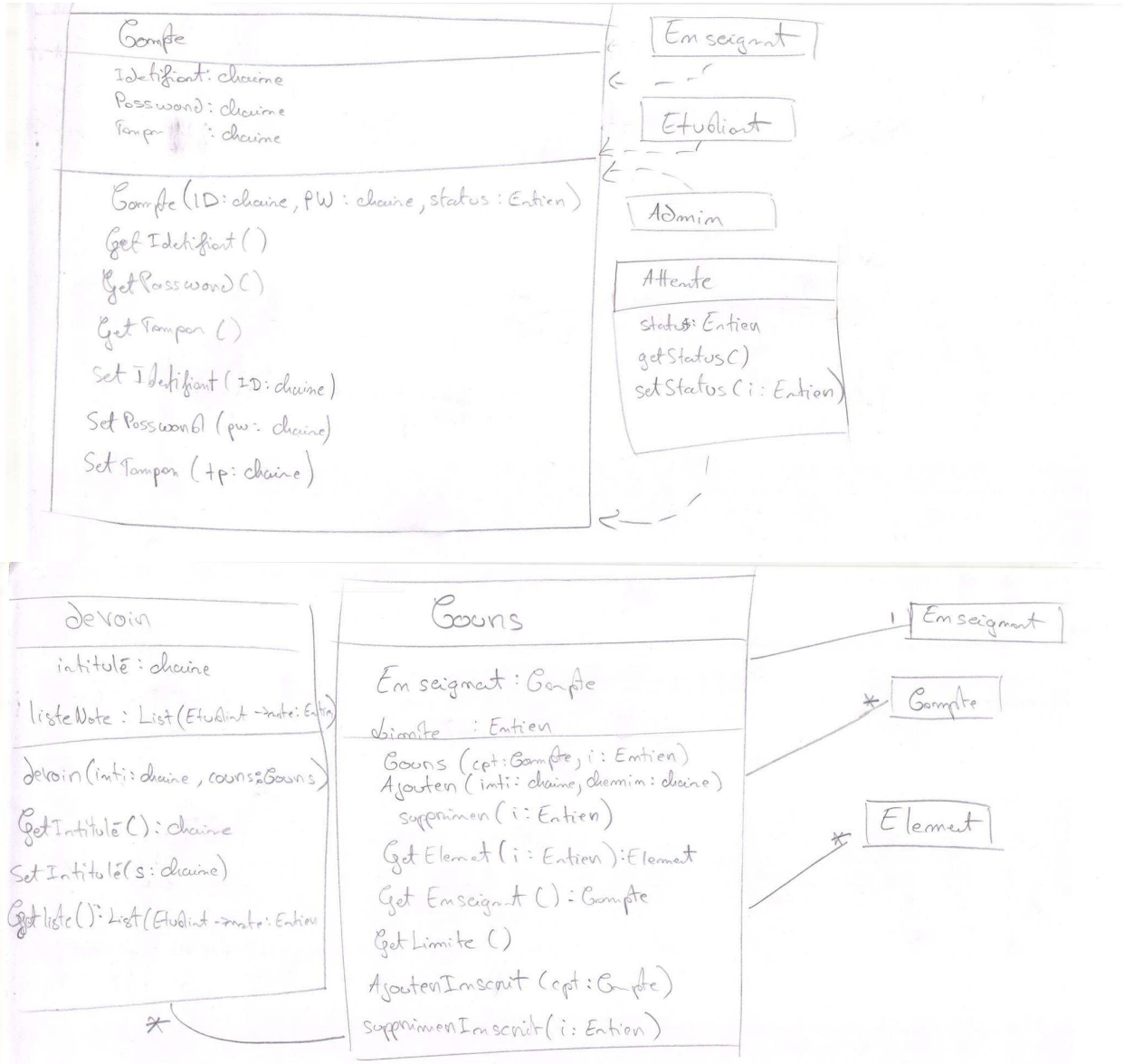


Scénario : Ajouter/Refuser un cours

VI. Diagramme de classe






Compte
Identifiant: chaine
Password: chaine
Tampon : chaine
Compte (ID: chaine, PW : chaine, status : Entier)
Get Identifiant ()
Get Password ()
Get Tampon ()
Set Identifiant (ID: chaine)
Set Password (pw: chaine)
Set Tampon (tp: chaine)
Enseignant
Etudiant
Admin
Attente
status: Entier
getStatus()
setStatus (i: Entier)
Devoir
intitulé : chaine
listeNote : List (Etudiant -note: Entier)
Devoir (inti: chaine, cours: Cours)
GetIntitulé (): chaine
SetIntitulé (s: chaine)
Getliste (): List (Etudiant -note: Entier
Cours
Enseignant : Compte
Limite : Entier
Cours (cpt: Compte, i : Entier)
Ajouter (inti: chaine, chemin: chaine)
supprimer (i : Entier)
Get Element (i : Entier): Element
Get Enseignant (): Compte
Get Limite ()
AjouterInscrit (cpt: Compte)
supprimerInscrit (i: Entier)
Enseignant
Compte
Element
1
*
*
*

Pennoquet

typeAjout : Entier session : Compte

connexion (identifiant : chaine) :
inscription (identifiant : chaine, mdp : chaine, status : Entier) :
accepter (nb : Entier) :
refuser (nb : Entier) :
ouvrirInscription (nbCours : Entier) :
fermerInscription (nbCours : Entier)
ajouterRessource (nbCours : Entier, elem : Element) :
ajouterDevoir (nbCours : Entier, dev : Element) :
ajouterDepot (nbCours : Entier, elem : Element) :
proposerCours (c : Cours).
accepterCompte (nb : Entier) :
refuserCompte (nb : Entier) :
ouvrirDepot (nb : Entier, c : Cours) :
fermerDepot (nb : Entier, c : Cours) :
retirerElement (nb : Entier, c : Cours).
noterDevoir (listNote : list (Entier), d : Devoir).
publierDevoir (d : devoir) :
setTypeAjout (n : Entier) :
getTypeAjout () : entier
getDepotFile (d : depot) :
SetDepotFile (d : depot, chemin : chaine)
getRessourceFile (d : ressource) :
setRessourceFile (d : ressource, chemin : chaine)
getNoteDevoir (d : devoir) : chaine
inscriptionCours (nb : Entier, e : Etudiant)
desinscriptionCours (nb : Entier, e : Etudiant)
getCoursEnseigne () : List (Cours)
getCoursSuivie () : List (Cours)

getListDepot (nb : Entier, c : cours) : List (Depot)
getListNote (nb : Entier, c : cours) : List (Entier)
getListDemande () : List (Compte)
getCoursEnseigne (e : Enseignant) : List (Cours)
getListPropositionCours () : List (Cours)
getSession () : Compte
setSession (s : Compte) :
getListEtud () : List (Compte)
getListEnseignant () : List (Compte)
getListAttente () : List (Compte)
getListAdmin () : List (Compte)
AjouterCompte (ID : chaine, mdp : chaine, status : Entier)
AjouterCompte (numero : Entier, b : boolean)
SupprimerCompte (perso : Entier)