République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université ABDERAHMANE MIRA de BEJAIA

Faculté des Sciences Exactes
Département d'Informatique

Mémoire

De fin de cycle

En vue de l'obtention du diplôme de Master en Informatique

Option : Génie Logiciel

Thème

Conception et réalisation d'une application d'apprentissage en ligne de la programmation par une évaluation automatisée

Proposé et dirigé par :

M[r] BOUZIDI L' hadi

Jury composé de :
Présidente :
Mme ALOUI Soraya
Examinateur :
Mr : ALOUI A.

Présenté par : MEGRI Zahir
RAHMANI Mohamed

Promotion 2017

Dédicaces

A ceux qui nous ont tout donné sans rien en retour

A ceux qui nous ont encouragés et soutenus dans les moments les plus difficiles

Et ceux à qui nous devons tant

A nos chers parents pour leur amour et leur support continu.

Que ce travail soit le témoignage sincère et affectueux de notre profonde reconnaissance pour tout ce que vous avez fait pour nous.

A nos grands-mères.

A nos frères et sœurs.

A nos oncles et tantes.

A nos amis (es).

Remerciements

Nous remercions le bon dieu le tout puissant de nous avoir donné la force nécessaire et la patience qui nous a permet de mener à bien ce modeste travail ;

Nous tenons à remercier :

Mr BOUZIDI L'hadi

D'avoir accepté de nos encadrer, ça sera pas suffisant pour lui exprimer toute nous reconnaissance pour la confiance et le grand soutien, pour le temps qu'il nous a consacré toute les fois que cela était nécessaire, pour ses conseils précieux qu'il nous a prodigué tout le long de notre travail, et pour son aide.

Mme ALOUI Soraya

De nous avoir fait l'honneur de présider le jury.

Mr ALLOUI A.

D'avoir accepté d'examiner notre travail.

Sans oublier l'ensemble des enseignants ayant contribué à notre formation durant notre cycle d'étude.

Enfin nos remerciements sont dressés plus particulièrement à nos familles et nos amis(es) qui ont su nous soutenir, nous encourager, nous aider et nous supporter tout au long des années.

Liste des figures

Figure 1 : Démarche choisie 6
Figure 2 : Processus de conception complet 8
Figure 3 : Spécification des besoins 15
Figure 4 : Diagramme de contexte 15
Figure 5 : Architecture globale du projet 16
Figure 6 : Diagramme de package des cas d'utilisation 17
Figure 7 : Cas d'utilisation du paquetage "Gestion des utilisateurs" 18
Figure 8 : Cas d'utilisation du paquetage "Gestion de l'environnement d'exécution" 19
Figure 9 : Cas d'utilisation du paquetage « Gestion des parcours d'apprentissage » 20
Figure 10 : Cas d'utilisation du paquetage « Gestion des sessions d'apprentissage » 21
Figure 11 : Cas d'utilisation du paquetage « Participation à une session d'apprentissage » 22
Figure 12 : Gabarit de l'application 23
Figure 13 : Structure des écrans IHM (acteur : Enseignant) 24
Figure 14 : Structure des écrans IHM (acteur : Admin) 25
Figure 15 : Structure des écrans IHM (acteur : Etudiant) 26
Figure 16 : Concepts et attributs liés à la gestion des utilisateurs 36
Figure 17 : Concepts du domaine liés à la gestion des parcours d'apprentissage 37
Figure 18 : Concepts et attributs liés à la participation à une session d'apprentissage 38
Figure 19 : Diagramme de classe d'entité 41
Figure 20 : DCP d'authentification 43
Figure 21 : DCP Gérer les utilisateurs 44
Figure 22 : DCP Gestion des parcours d'apprentissage 44
Figure 23 : Diagramme de classe participante participation à une session d'apprentissage 45
Figure 24 : Notation graphique de base du diagramme d'états 47
Figure 25 : Diagramme de navigation authentification 48
Figure 26 : Diagramme de navigation gestion des utilisateurs 48
Figure 27 : Diagramme de navigation gérer les parcours d'apprentissage 49
Figure 28 : Diagramme de navigation participer à une session d'apprentissage 50
Figure 29 : Passage de l'analyse à la conception préliminaire 52
Figure 30 : DSD « authentification » 53
Figure 31 : DSD « Gestion des utilisateurs » 54
Figure 32 : DSD « gestion des parcours d'apprentissage » 55
Figure 33 : DSD participé à une session d'apprentissage 56
Figure 34 : DCC pour la gestion des utilisateurs 58
Figure 35 : DCC : gestion des parcours 59
Figure 36 : DCC de cas d'utilisation participé à une session d'apprentissage 60
Figure 37 : Modèle de la base de données APEA 63
Figure 38 : Architecture logicielle 65
Figure 39 : La structure du répertoire de cakePHP qui contiendra l'application APEA 66
Figure 40 : La structure de répertoire « src » 67
Figure 41 : Exemple d'un Contrôler 67
Figure 42 : Exemple d'un modèle 68
Figure 43 : Exemple d'une vue 68
Figure 44 : Interface d'accueil avec l'authentification 69
Figure 45 : Interface d'ajouter un enseignant 69

Liste des tableaux

Tableau 1 : Page d'accueil et authentification 27
Tableau 2 : IHM page admin la gestion des utilisateurs 27
Tableau 3 : IHM page admin gestion des langages 28
Tableau 4 : IHM page enseignant gestion des parcours d'apprentissage 28
Tableau 5 : IHM page enseignant gestion des sessions d'apprentissage 29
Tableau 6 : IHM page étudiant découvrir le code secret 29
Tableau 7 : DSS du cas d'utilisation : Authentification (acteur : Anonyme). 31
Tableau 8 : DSS du cas d'utilisation : Gérer les utilisateurs 32
Tableau 9 : DSS du cas d'utilisation : Gérer les parcours 33
Tableau 10 : DSS du cas d'utilisation : Participer à une session d'apprentissage 34
Tableau 11 : Identifier des concepts du domaine 38
Tableau 12 : Identification des attributs 39
Tableau 13 : Identification des attributs (suite) 40

Glossaire

APEA : Apprentissage de la programmation par le biais d'une évaluation automatisée

ESEC : Environnement sécurisé d'exécution de code

UML: Unified Modeling Language

UC: Use case

DSS : Diagramme de séquence système

DCP : Diagramme de classe participante

IHM : Interface Homme-Machine

DCC : Diagramme de classe de conception

DCA : Diagramme de classe d'analyse

MCV : Modèle Vue Contrôleur

DSD : Diagramme de séquence détaillé

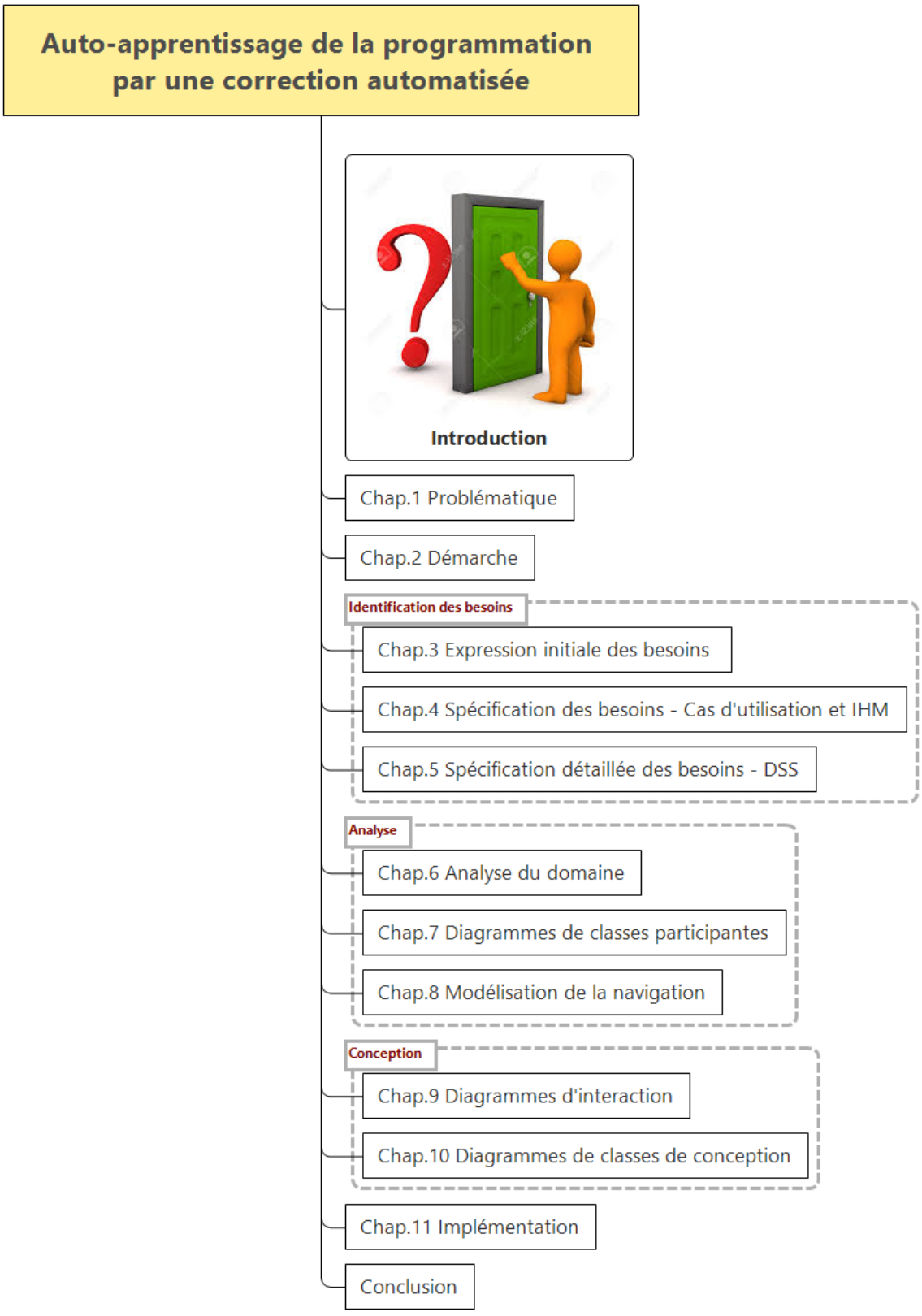
cakePHP : CakePHP est un framework web libre écrit en PHP distribué sous licence MIT. Il suit le motif de conception Modèle-Vue-Contrôleur

Sommaire

Remerciement I
Dédicaces II
Liste des figures III
Liste des tableaux IV
Glossaire V
Introduction 1
Chap.1 – Problématique 3
Chap.2– Démarche 5
Chap.3 – Expression initiale des besoins 9
3-1 PRESENTATION DU PROJET 10
3-2 BESOINS FONCTIONNELS 10
3-3 BESOINS NON FONCTIONNELS 13
Chap.4 –Spécification des besoins 14
4.1 POSITIONNEMENT DE L'ETAPE 15
4.2 DIAGRAMME DE CONTEXTE 15
4.3 DIAGRAMME DE PACKAGE DES CAS D'UTILISATION 17
4.4 DIAGRAMMES DE CAS D'UTILISATION 18
4.5 MAQUETTAGE DE L'IHM 23
Chap.5– Spécification détaillée des besoins 30
5.1 POSITIONNEMENT DE L'ETAPE 31
5.2 LES DIAGRAMMES DE SEQUENCES SYSTEME 31
Chap.6–Analyse du domaine 35
6.1 POSITIONNEMENT DE L'ETAPE 36
6.2 ANALYSE DE DOMAINE POUR CHAQUE CAS D'UTILISATION 36
6.3 DIAGRAMME DE CLASSES D'ENTITE 38
Chap.7–Diagrammes de classes participantes 42
7.1 POSITIONNEMENT DE L'ETAPE 43
7.2 RAFFINER L'ANALYSE : CLASSES PARTICIPANTES 43
Chap.8 – Modélisation de la navigation 46
8.1 POSITIONNEMENT DE L'ETAPE 47
8.2 DIAGRAMME D'ETATS 47
Chap.9 – Diagrammes d'interaction 51
9.1 POSITIONNEMENT DE L'ETAPE 52
9.2 ANALYSE DE L'INTERACTION 53
Chap.10 – Diagrammes de classes de conception 57
10.1 POSITIONNEMENT DE L'ETAPE 58
10.2 CONCEPTION DETAILLEE 58
Chap.11 – Implémentation 61
11.1 POSITIONNEMENT DE L'ETAPE 62
11.2 ENVIRONNEMENT DE DEVELOPPEMENT 62

11.3 SOLUTION TECHNOLOGIQUE 65
11.4 STRUCTURE DE NOTRE APPLICATION 66
11.5 CODAGE 67
11.6 IHM 69
Conclusion 70
Bibliographie 71
Annexes

Introduction



Introduction

L'évaluation continue durant un cours de programmation doit assurer que les étudiants aient de plus en plus de pratiques et autant de rétroactions sur la qualité de leurs solutions. Idéalement, l'étudiant devrait obtenir une rétroaction instantanée à sa solution afin qu'il puisse se rendre compte des erreurs commises et les rectifier et avancer ainsi dans son apprentissage. Même avec des groupes de 10 ou 12 étudiants, un chargé de TP n'est pas capable d'assurer une telle mission ! C'est la raison pour laquelle des systèmes d'évaluation automatisée des programmes ont été créés ces dernières années.

Notre projet est commandité par le département d'informatique de l'université de Bejaia. Il s'inscrit dans le champ des outils d'évaluation automatisée des TP de programmation. Notre but est de mettre à la disposition des Universités (et de façon général des établissements éducatifs), un outil en ligne permettant de faciliter à la fois l'apprentissage de la programmation et l'évaluation des codes produits par les étudiants.

A la différence de la majorité des outils qui ont été réalisés ces dernières années, l'outil que nous voulons réaliser met l'accent sur le concept de parcours d'apprentissage et d'apprentissage par les pairs. Nous avons voulu aussi, nous inscrire dans une démarche ludique et adopter une métaphore de jeux en qualifiant les exercices comme des défis à relever par les étudiants. Nous avons ainsi voulu réunir les ingrédients des jeux comme les scores, les tableaux comparatifs et l'interaction entre les étudiants (qui deviennent des joueurs !).

Plus techniquement, notre but est de concevoir et de réaliser à la fois une application web qui met en avant l'interface avec les différents utilisateurs de notre système (notamment des étudiants et des enseignants), et une plate-forme d'exécution des codes écrits dans plusieurs langages de programmation.

Du point de vue méthodologique, nous nous sommes inspirés de la méthode de conception proposée par Pascal Roques [Roques, 2006]. Il s'agit d'une méthode conception simplifiée et orientée vers le développement d'applications Web tout en se basant sur le langage de modélisation UML2.

Nous présentons, dans ce rapport, d'abord une partie théorique, dans laquelle nous abordons le langage UML et une méthode d'analyse et de conception des systèmes logiciels. Ajouter à cela, nous avons trouvé utile de donner un aperçu sur l'état de l'art en ce qui concerne l'évaluation automatisée des TP de programmation. Nous terminons cette partie théorique la démarche que nous avons suivie tout au long de notre travail.

La suite de ce rapport, retrace l'application de la démarche de conception que nous avons adoptée : Identification des besoins décrite dans les chapitres 2, 3 et 4, l'analyse présentée dans les chapitres 5, 6 et 7, la conception logicielle présentée dans les chapitre 8 et 9 et enfin l'implémentation décrites dans le chapitre 10.

Chap.1 – Problématique

L'apprentissage de la programmation exige énormément de pratique, de l'interactivité et surtout des feedbacks instantanés afin de permettre aux étudiants de comprendre leurs erreurs et de les corriger et de consolider ainsi leurs connaissances théoriques et d'acquérir de nouvelles compétences. Le processus habituellement suivi dans nos universités est accès sur des travaux pratiques sur machine et une assistance de chargés de TP. Malheureusement, même avec des groupes de 12 étudiants, un chargé de TP sera incapable de répondre instantanément aux sollicitations fréquentes des étudiants. De ce fait, certains étudiants perdent énormément de temps et se retrouvent souvent bloqués et ne reçoivent pas rapidement des aides ou des orientations leur permettant d'avancer. C'est pour cela, que l'idée d'une évaluation automatisée des TP de programmation est née il y a un peu plus d'une décennie. Il existe peu d'étude sur ce domaine. L'un des travaux de recherche que nous pouvons citer est celui de PETRI IHANTOLA [IHANTOLA 2010] qui a présenté une étude concernant ce domaine entre 2006 et 2010. Par ailleurs, actuellement, les cours massifs en ligne (MOOC[1]) ont pris de l'ampleur et sont de véritables concurrents aux méthodes d'apprentissage classique. Nous avons découvert que certains de ces MOOC qui traitent des langages de programmation se servent d'une évaluation automatisée des codes soumis par les apprenants. Ce procéder d'évaluation n'a pas été un choix en réalité car il est impossible pour une équipe pédagogique aussi grande soit-elle d'évaluer des milliers de travaux des étudiants. A l'échelle d'une Université comme celle de Bejaïa, une évaluation automatisée des TP de programmation pourrait aider considérablement les chargés de TP. En effet, si l'ordinateur se charge lui-même de délivrer des rapports d'évaluation instantanés sur les codes soumis par les étudiants, ces derniers pourront apprendre plus vite et solliciter moins fréquemment les chargés de TP.

Dans ce projet, c'est à cette problématique de l'évaluation automatisée des TP de programmation que nous allons essayer de donner une réponse.

Chap.2– Démarche

Auto-apprentissage de la programmation par une correction automatisée

- Introduction
- Chap.1 Problématique

Chap.2 Démarche

Identification des besoins

- Chap.3 Expression initiale des besoins
- Chap.4 Spécification des besoins - Cas d'utilisation et IHM
- Chap.5 Spécification détaillée des besoins - DSS

Analyse

- Chap.6 Analyse du domaine
- Chap.7 Diagrammes de classes participantes
- Chap.8 Modélisation de la navigation

Conception

- Chap.9 Diagrammes d'interaction
- Chap.10 Diagrammes de classes de conception
- Chap.11 Implémentation
- Conclusion

Le débat sur les démarches de conception d'applications logicielles n'a jamais cessé et la question n'a jamais été tranchée. L'idéal est d'avoir une méthode à la fois robuste et simple permettant d'aller des besoins au code. Pour la conception d'application Web, plusieurs auteurs [ROQUES 2006] proposent une démarche basée sur le langage de modélisation UML. Ce dernier est devenu un standard pour décrire tout système logiciel. Il constitue une norme qui a été décrite en même temps qu'une méthode d'analyse et de conception des systèmes logiciels, le *Processus Unifié* qui est la méthode de développement pour les logiciels orientés objets. Cette méthode est générique, itérative et incrémentale, contrairement à des méthodes séquentielles comme Merise. Le couple UML et Processus Unifié propose une approche pour conduire la réalisation de systèmes orientés Objet depuis les spécifications jusqu'au déploiement. [ROQUES 2006 ; ROQUES 2007 ; CHARROUX 2008 ; GABAY, 2008]

Pour mener à bien notre projet, nous avons suivi une méthode proposée par Pascal Roques. Ce dernier décrit un processus pour le développement d'applications web se situant à mi-chemin entre UP (Unified Process) un cadre général très complet de processus de développement, et les méthodes agiles en vogue actuellement, telles que XP (eXtreme Programming) et Scrum. Il s'inspire également des bonnes pratiques prônées par les tenants de la modélisation agile (Agile Modeling) [ROQUES 2006].

Plus spécifiquement, nous avons suivi trois phases : identification des besoins (exigences), analyse et enfin conception. Pour chaque phase nous avons entrepris plusieurs activités. Le processus que nous appliqué est :

- conduit par les cas d'utilisation, comme UP, mais beaucoup plus simple ;
- relativement léger et restreint, comme les méthodes agiles, mais sans négliger les activités de modélisation en analyse et conception ;
- fondé sur l'utilisation d'un sous-ensemble du langage UML.

La figure suivante illustre bien le processus que nous avons suivi :

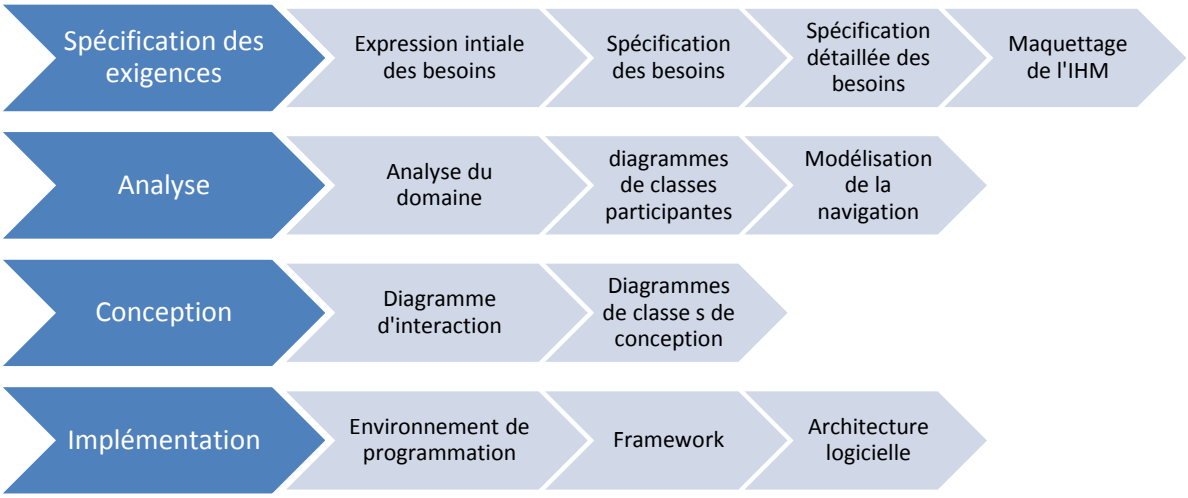


Figure 1 : Démarche choisie

Durant la phase de spécification des exigences, les besoins ont été modélisées au moyen des cas d'utilisation UML et ont été représentés de façon plus concrète par une maquette d'IHM (Interface Homme-Machine). Chaque cas d'utilisation est décrit textuellement de façon détaillée et a donné lieu à un diagramme de séquence simple représentant graphiquement la chronologie des interactions entre les acteurs et le système vu comme une boîte noire. Nous appellerons ce diagramme : «diagramme de séquence système» (DSS).

Durant la phase d'analyse, nous avons procédé à l'analyse du domaine métier. Ainsi, nous avons dégagé les classes « métier » (ou entité) concernant notre problème. Nous avons produits ainsi, un diagramme de classe d'entités. Par la suite, en partant de chaque cas d'utilisation et en se référant aux maquettes d'IHM, nous avons procédé à la modélisation de trois types de classe d'analyse : Les dialogues, les contrôles et les entités. Nous avons produits ainsi, pour chaque cas d'utilisé un diagramme de classe participantes. En nous basant sur les maquettes IHM, nous avons déduit un modèle de navigation pour chaque cas d'utilisation.

Durant la phase de conception, nous nous efforçons à construire des classes de conception logicielle nous permettant de passer directement au code. A cet effet, nous attribuons les responsabilités de comportement, dégagées par le diagramme de séquence système, aux classes d'analyse du diagramme de classes participantes. Les résultats de ce travail abouti aux diagrammes d'interaction. Parallèlement, une première ébauche de la vue statique de conception, c'est-à-dire du diagramme de classes de conception, est construite et complétée. Durant cette phase, l'ébauche du diagramme de classes de conception reste indépendante des choix technologiques qui seront faits ultérieurement.

Durant la phase d'implémentation, nous nous fixons sur l'environnement de développement, le choix du langage de programmation, les *Frameworks* et le modèle d'architecture logicielle. Ensuite, nous passons au codage.

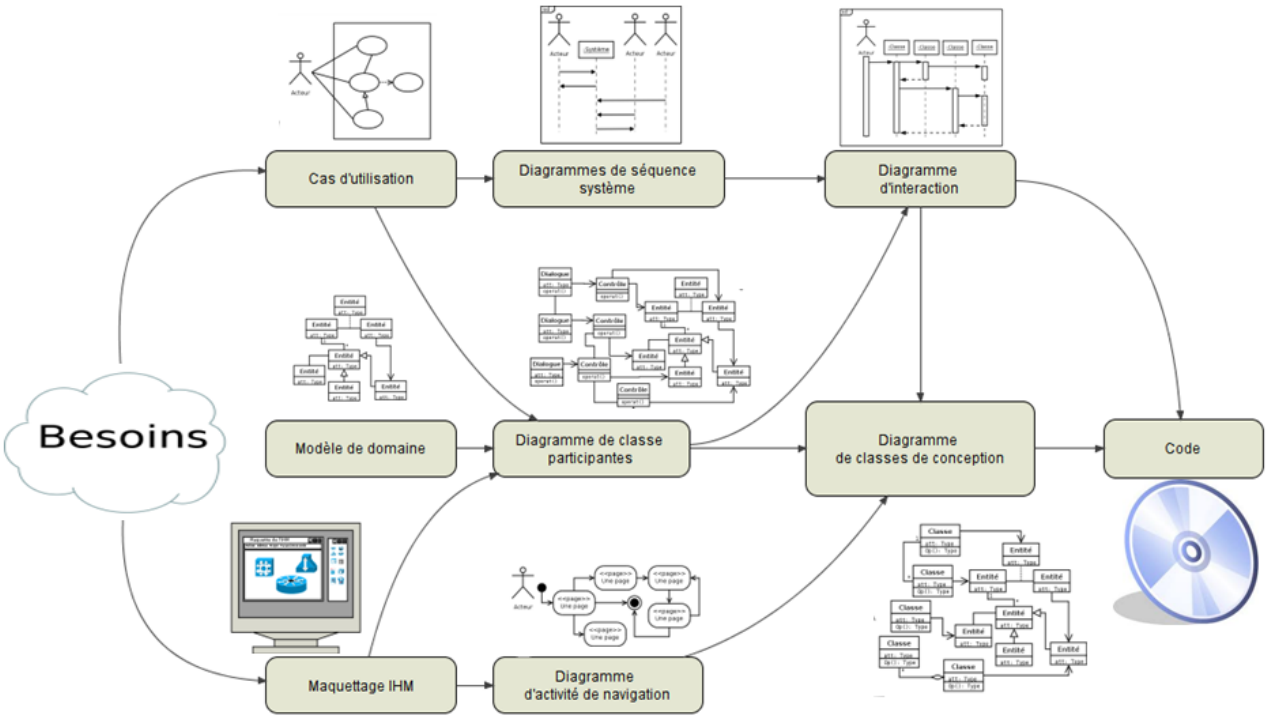
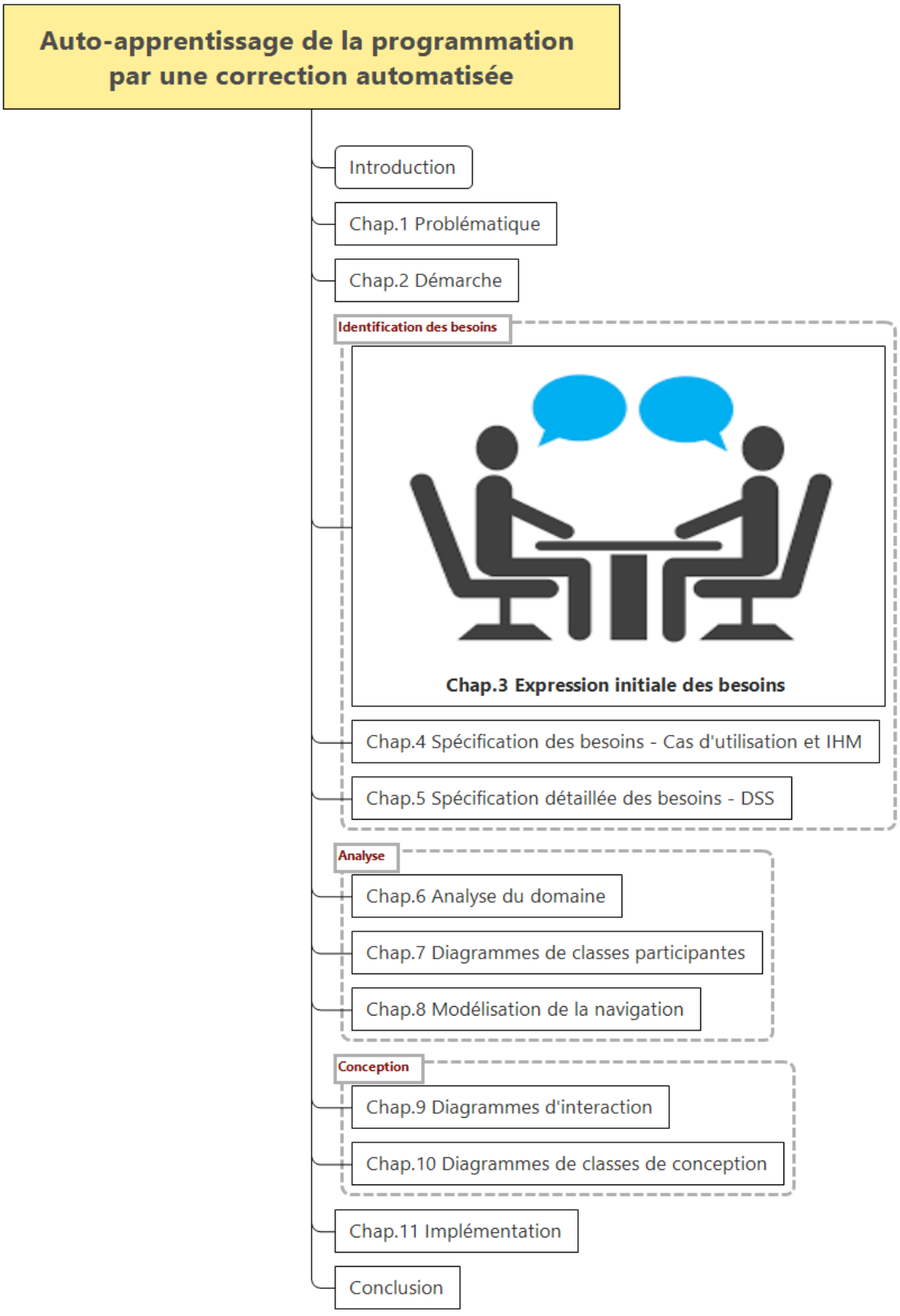


Figure 2 : Processus de conception complet

Chap.3–Expression initiale des besoins



Dans cette partie, nous allons présenter notre projet et l'expression initiale des besoins fonctionnelles et non fonctionnelles tel que formulés pas le commanditaire.

3-1 Présentation du projet

Intitulé du projet : Création d'une plate-forme d'apprentissage en ligne de la programmation par le biais d'une évaluation automatisée.

Organisme d'accueil : Université *Aberahmane Mira* (Département d'informatique)

Public visé : Etudiants qui participent à des tps de programmations et enseignants qui conçoivent ces TP.

Objectifs généraux:

- Permettre à des enseignants de proposer des défis à relever sous forme d'exercices de programmation.
- Permettre à des étudiants de relever ces défis en proposant leurs solutions et en obtenant en réponse des feedbacks « instantanés » sur la qualité de leurs codes.
- Permettre à des étudiants de consulter les solutions de leurs camarades et d'en débattre.

3-2 Besoins fonctionnels

Le système que nous devons réaliser doit permettre un apprentissage de la programmation par le biais d'une évaluation automatisée. Il doit accueillir 3 types d'utilisateurs : des étudiants, des enseignants et des administrateurs.

Les administrateurs doivent pouvoir gérer les utilisateurs de notre système et paramétrer les langages de programmation qui seront pris en charge.

Les enseignants doivent pouvoir gérer des parcours d'apprentissage, gérer des sessions d'apprentissage et superviser les activités de chaque session.

Les étudiants doivent pourvoir suivre des parcours d'apprentissage en résolvant ses défis.

Tous les acteurs de notre système doivent être authentifiés pour pouvoir bénéficier de ses services et doivent être capables de voir et de modifier leurs profils (nom, prénom, mot de passe, courriel et photo)

On définira un parcours d'apprentissage comme un processus que pourrait suivre un étudiant pour apprendre un langage de programmation cible. Voici sa composition :

- un « fil rouge » consistant en une batterie de compétences générales que devrait acquérir l'étudiant.
- un ou de plusieurs paliers.

Un palier doit concerner un sous ensemble de compétences générales du parcours d'apprentissage qui le contient. Il doit être associé à une batterie de compétences ciblées

(spécifiques) et un ensemble de défis à relever par les étudiants. Il doit être caractérisé par un score maximum que l'étudiant pourra obtenir, mais aussi par un seuil minimum que l'étudiant doit avoir pour considérer que le palier est acquis et ainsi permettre le passage au palier suivant.

Un défi est en fait un exercice ou un problème que doit résoudre un étudiant. Il doit couvrir un ensemble de compétences spécifiques parmi la batterie de compétences du palier. Il doit être caractérisé par :

- un énoncé
- un ensemble de tests unitaires
- éventuellement une illustration graphique et une illustration vidéo
- un corrigé
- une liste de ressources pédagogiques (documents téléchargeables)

Voyons maintenant un peu plus en détails les services attendus de notre système au profit de l'administrateur : Gérer des utilisateurs et paramétrage des langages de programmation.

On sous-entend par « *gestion des utilisateurs* » les opérations de création, modification et suppression d'étudiants, d'enseignants et de classes (groupe d'étudiants).

Concernant le paramétrage des langages de programmation autorisés, l'administrateur pourra ajouter de nouveaux langages de programmation. Pour chacun de ces langages, il doit indiquer le nom du langage, un dossier des codes et un dossier des feedback (compte rendu de l'évaluation automatisée). Le dossier des codes va contenir les codes des étudiants qui seront soumis pour une évaluation. Le dossier des feedbacks va contenir les rapports d'évaluation automatisée générés par un environnement sécurisé d'exécution des codes des étudiants.

Voyons maintenant un peu plus en détails les services attendus de notre système au profit de l'enseignant : Gérer des parcours d'apprentissage, gérer des sessions d'apprentissage et superviser les activités de chaque session.

On sous-entend par « *gestion des parcours d'apprentissage* » les opérations de création, modification et suppression de parcours d'apprentissage auxquelles il faut rajouter la gestion des compétences générales du parcours et celle de ses paliers.

Dans le groupe fonctionnel « gestion des compétences générales », l'enseignant doit pouvoir ajouter, modifier et supprimer des compétences générales. Il doit aussi pouvoir éditer (ajout, modification et suppression) de compétences spécifiques pour chaque compétence générale.

Dans ce dernier groupe fonctionnel « gestion des paliers », l'enseignant doit pouvoir créer, modifier et supprimer des paliers. En plus il doit être capable de gérer les défis associés à chaque palier.

Dans le groupe fonctionnel « gestion des défis », l'enseignant doit pouvoir ajouter, éditer et supprimer des défis. L'édition d'un défi doit donner lieu aux fonctionnalités suivantes :

- éditer des informations générales sur le défi (nom, score maximum, nombre de tentatives autorisées)
- rattacher ou retirer des compétences spécifiques au défi,
- ajouter, modifier ou supprimer des indices et des ressources pédagogiques (documents PDF par exemple).
- éditer l'énoncé du défi (texte et éventuellement illustration graphique et vidéo)
- éditer le corrigé du défi (texte et éventuellement illustration graphique et vidéo)
- éditer les tests unitaires (nom, données, poids : pourcentage du score du défi)

Dans le groupe fonctionnel « gestion des sessions d'apprentissage », l'enseignant doit pouvoir, créer, paramétrer, supprimer et superviser des sessions. On définira une session par 4 paramètres : date de début, date de fin, le parcours d'apprentissage concerné et la classe (groupe) d'étudiants inscrits à cette session.

Dans le groupe fonctionnel « supervision de session », l'enseignant pourra consulter le déroulement des activités des étudiants pour une session donnée par palier et par défis. Il doit aussi pouvoir consulter la situation d'un étudiant donnée et vois notamment les défis pour lesquels il a essayé de résoudre. Il doit pouvoir voir notamment les scores obtenus pour chaque défi, le score total en cours obtenu, le classement de l'étudiant par rapport à sa classe par palier et par défis pour une session donnée. En plus, l'enseignant doit pouvoir consulter l'énoncé d'un défi donné, son corrigé et les rapports d'évaluation par défis pour chaque étudiant.

Voyons maintenant un peu plus en détails les services attendus de notre système au profit de l'étudiant : Nous avons regroupé les cas d'utilisation associés à l'étudiant dans un seul groupe fonctionnel que nous avons appelé : « participer à une session d'apprentissage ». En effet, notre application est faite pour l'étudiant et surtout pour qu'il puisse apprendre à programmer. Ce que nous souhaitons que le système rendre comme services à l'étudiant est que ce dernier soit capable de gérer son profil et de participer à une session d'apprentissage. Nous rappelons qu'une session d'apprentissage concerne un parcours qui sera poursuivi durant une période de temps. Ainsi, l'étudiant doit pouvoir consulter toutes les sessions auxquelles il participe. Il doit pouvoir voir son classement à tout moment par rapport à sa classe, le nombre de palier franchis pour une session donnée. Il pourra aussi consulter la liste des défis qu'il doit relever et la liste des défis clôturés. Pour un défi à relever, il doit pouvoir consulter : l'énoncé, le nombre de tentatives qu'il a déjà fait et le nombre totale de tentatives autorisées, les illustrations graphiques et vidéos, les compétences qu'il doit mobiliser. De plus il doit pouvoir résoudre le défi en proposant un code. Une fois qu'il a proposé un code, il va pouvoir consulter un rapport d'évaluation automatisé lui indiquant si son code est syntaxiquement correct. De plus on lui affichera le

nombre de tests unitaires qu'il a réussi et la liste des compétences spécifiques qu'il a acquis.

3-3 Besoins non fonctionnels

Le système que nous devons réaliser doit garantir une bonne réactivité notamment lorsque les étudiants soumettent leurs travaux pour l'évaluation. De plus, sachant que les codes des étudiants seront exécutés dans un environnement d'exécution, il faut prévoir toutes les mesures de sécurité afin d'éviter les conséquences de code malveillants. Par ailleurs, il faut assurer une bonne qualité ergonomique de sorte à rendre l'IHM du futur système attrayant en offrant une charte graphique de qualité et une navigation optimisée.

Chap.4 –Spécification des besoins

Auto-apprentissage de la programmation par une correction automatisée

- Introduction
- Chap.1 Problématique
- Chap.2 Démarche
- **Identification des besoins**
  - Chap.3 Expression initiale des besoins
  - 

    Chap.4 Spécification des besoins - Cas d'utilisation et IHM
  - Chap.5 Spécification détaillée des besoins - DSS
- **Analyse**
  - Chap.6 Analyse du domaine
  - Chap.7 Diagrammes de classes participantes
  - Chap.8 Modélisation de la navigation
- **Conception**
  - Chap.9 Diagrammes d'interaction
  - Chap.10 Diagrammes de classes de conception
- Chap.11 Implémentation
- Conclusion

4.1 Positionnement de l'étape

Rappelons le positionnement de cette activité de spécification des besoins rapport à l'ensemble du processus que nous avons suivi. L'expression intiale des besoins donne lieu d'une part à la délimitation du contexte de notre projet, au maquetage des cas d'utilisation aboutissant à une modélisation par les cas d'utilisation, et d'autre part à une maquette d'interface homme-machine (IHM), commeindiqué sur la figure 3-1.

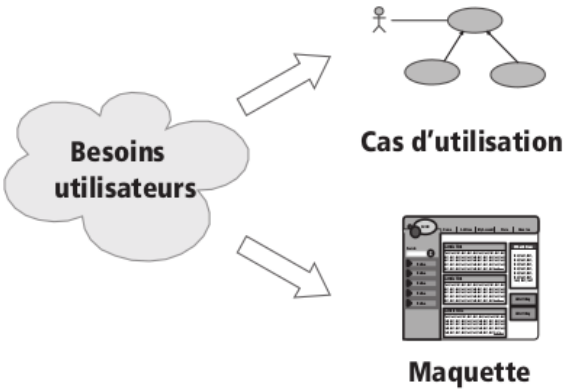


Figure 3 - Spécification des besoins

4.2 Diagramme de contexte

Ce diagramme n'est pas officiellement désigné comme diagramme UML mais il est utile pour la définition des acteurs, avant de commencer à s'intéresser à d'autres aspects, tels que les packages et les cas d'utilisation. Nous définirions notre futur système comme une boite noire rendant des services à des acteurs principaux et pouvant aussi interagir avec des acteurs secondaires. On a identifié 3 acteurs principaux : l'étudiant, l'enseignant et l'administrateur. On a aussi identifié un acteur externe qui va s'occuper d'exécuter les codes des étudiants et rendre des rapports d'évaluation de façon « transparente ».

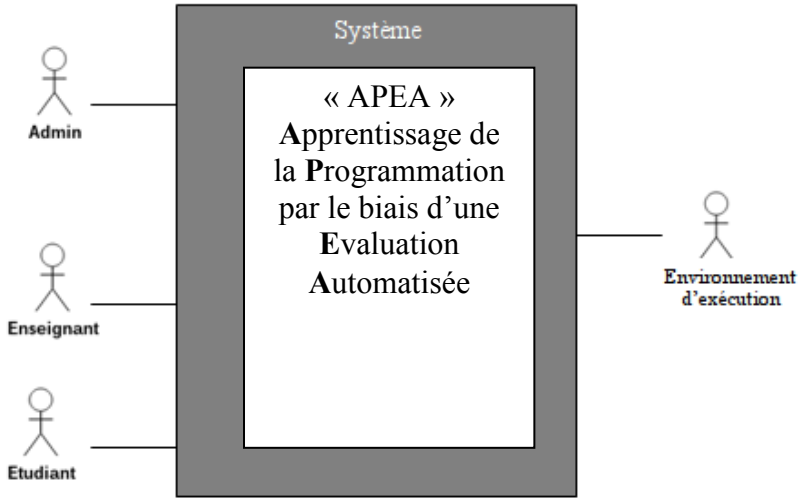


Figure 4 : Diagramme de contexte

En réalité, notre projet consiste à concevoir et réaliser 2 systèmes : le système de d'apprentissage de programmation de la programmation par le biais d'une évaluation automatisée (APEA) et l'environnement sécurisé d'exécution des codes (ESEC). Ce dernier système doit être complètement séparé du premier système (APEA) pour des raisons de sécurité, car on sera amené à exécuter des codes des étudiants. Il pourrait être constitué d'une machine physique (ou virtuelle) indépendant, mais partageant un canal de communication avec le premier système (APEA) pour récupérer les code puis de rendre les rapports d'évaluation).

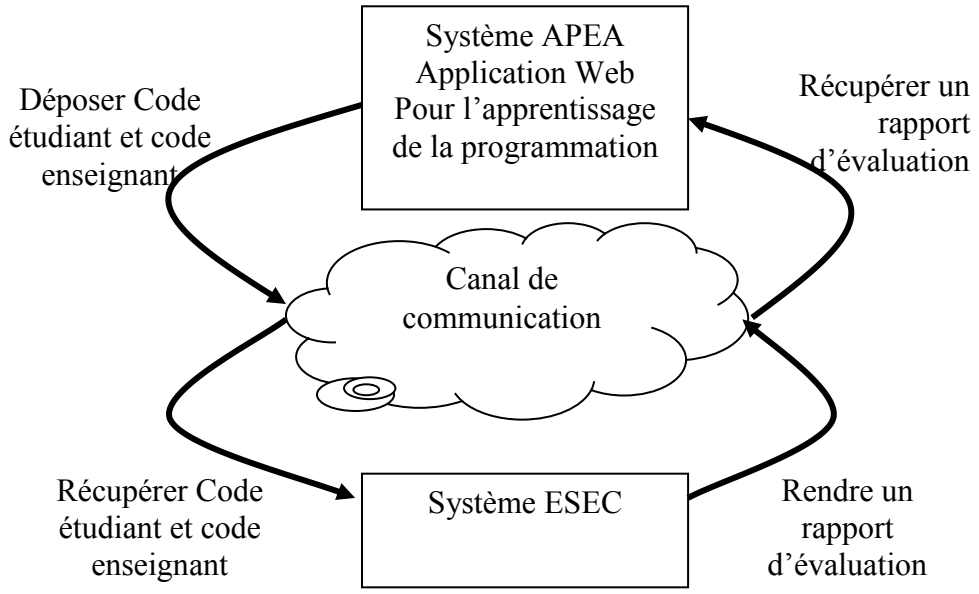


Figure 5 : Architecture globale du projet

Dans la suite de ce projet, nous nous concentrerons surtout sur le premier système (APEA).

Brève description des acteurs :

Administrateur (Admin) : Personne chargée de la gestion des utilisateurs et du paramétrage de l'environnement d'exécution des codes qui seront soumis par les étudiants lors de leurs activités de TP de programmation.

Enseignant : Personne chargé de créer des parcours et des sessions d'apprentissage et de la supervision de ces sessions.

Etudiant : Personne pouvant participer à des sessions d'apprentissage.

Système ESEC : Machine externe (physique ou virtuelle) chargée d'exécuter les codes des étudiants et d'en générer de façon automatisée des rapports d'évaluation.

4.3 Diagramme de package des cas d'utilisation

Les besoins très différents des acteurs et le nombre de fonctionnalités dont notre futur logiciel devra disposer nous semble un peu compliqué. Pour nous faciliter la tâche, nous avons découpé le futur logiciel en parties distinctes, en fonction des familles de fonctionnalités et de façon à pouvoir les analyser séparément. Chacune de ces parties correspond à un domaine fonctionnel ou package. Nos premières discussions avec le commanditaire du projet nous ont permis de dégager 5 groupes fonctionnels regroupant les fonctions attendues du système : gestion des utilisateurs et de l'environnement d'exécution associé à l'acteur « Administrateur », gestion des parcours d'apprentissage et de sessions associé à l'acteur « enseignant » et enfin « Résolution de défis » associé à l'acteur « étudiant ».

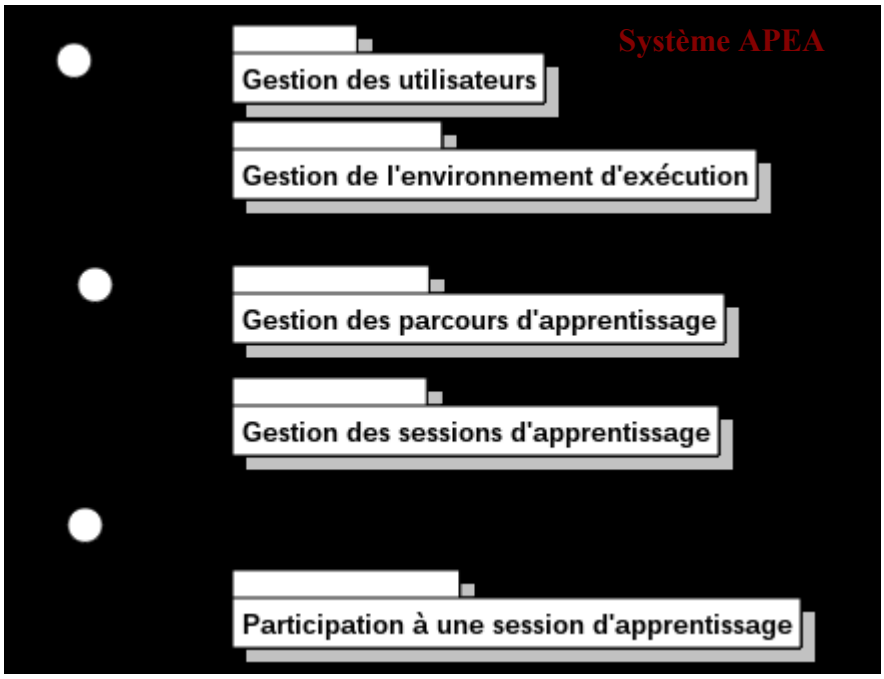


Figure 6 : Diagramme de package des cas d'utilisation

4.4 Diagrammes de cas d'utilisation

Nous avons donc identifié les acteurs, et les grandes familles de fonctionnalités (packages) attendues de notre système. Maintenant, il s'agit de définir plus en détail les besoins de chaque acteur dans ces packages.

Paquetage 1 : Gestion des utilisateurs

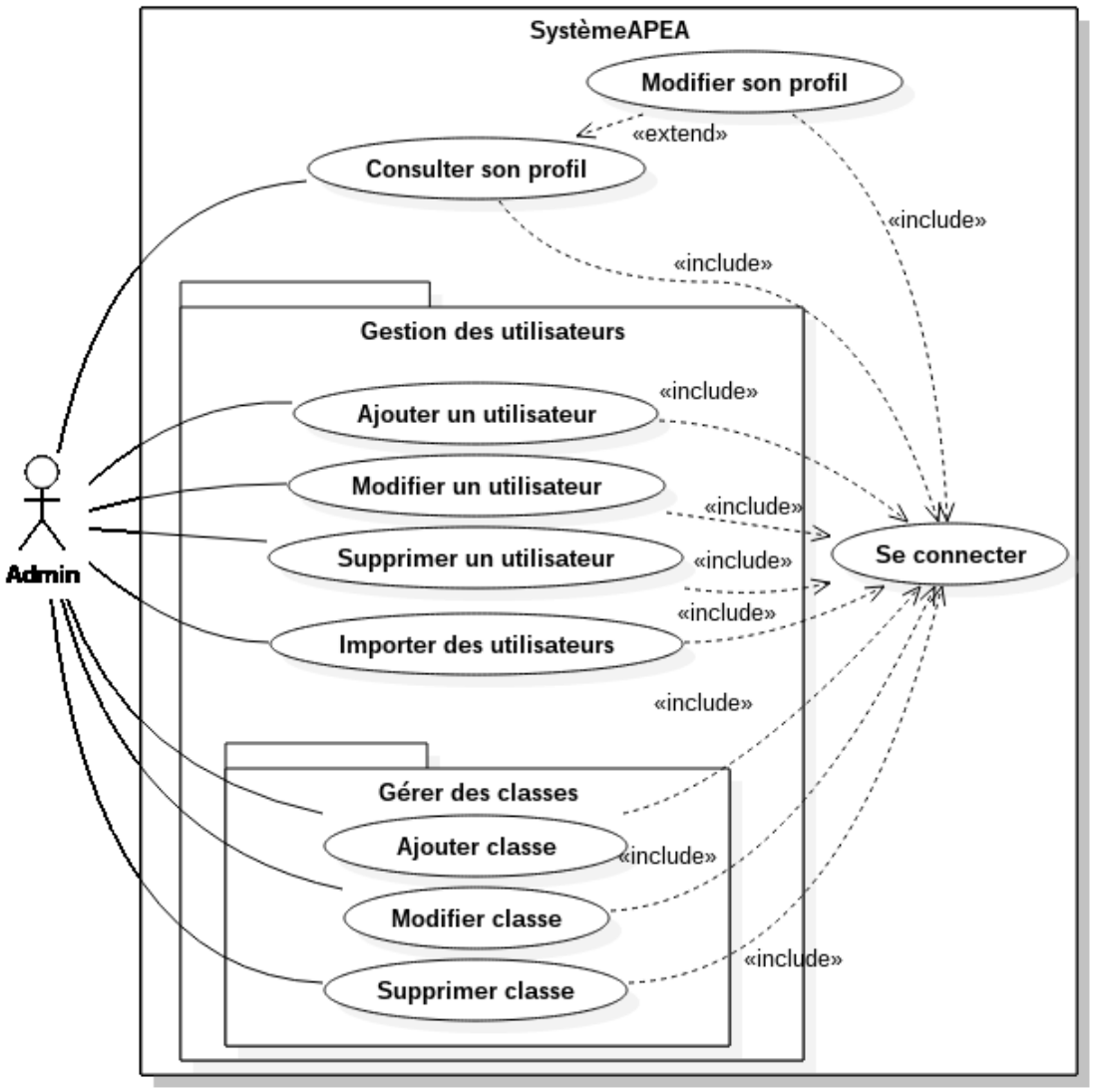


Figure 7 : Cas d'utilisation du paquetage "Gestion des utilisateurs"

Paquetage 2 : Gestion de l'environnement d'exécution

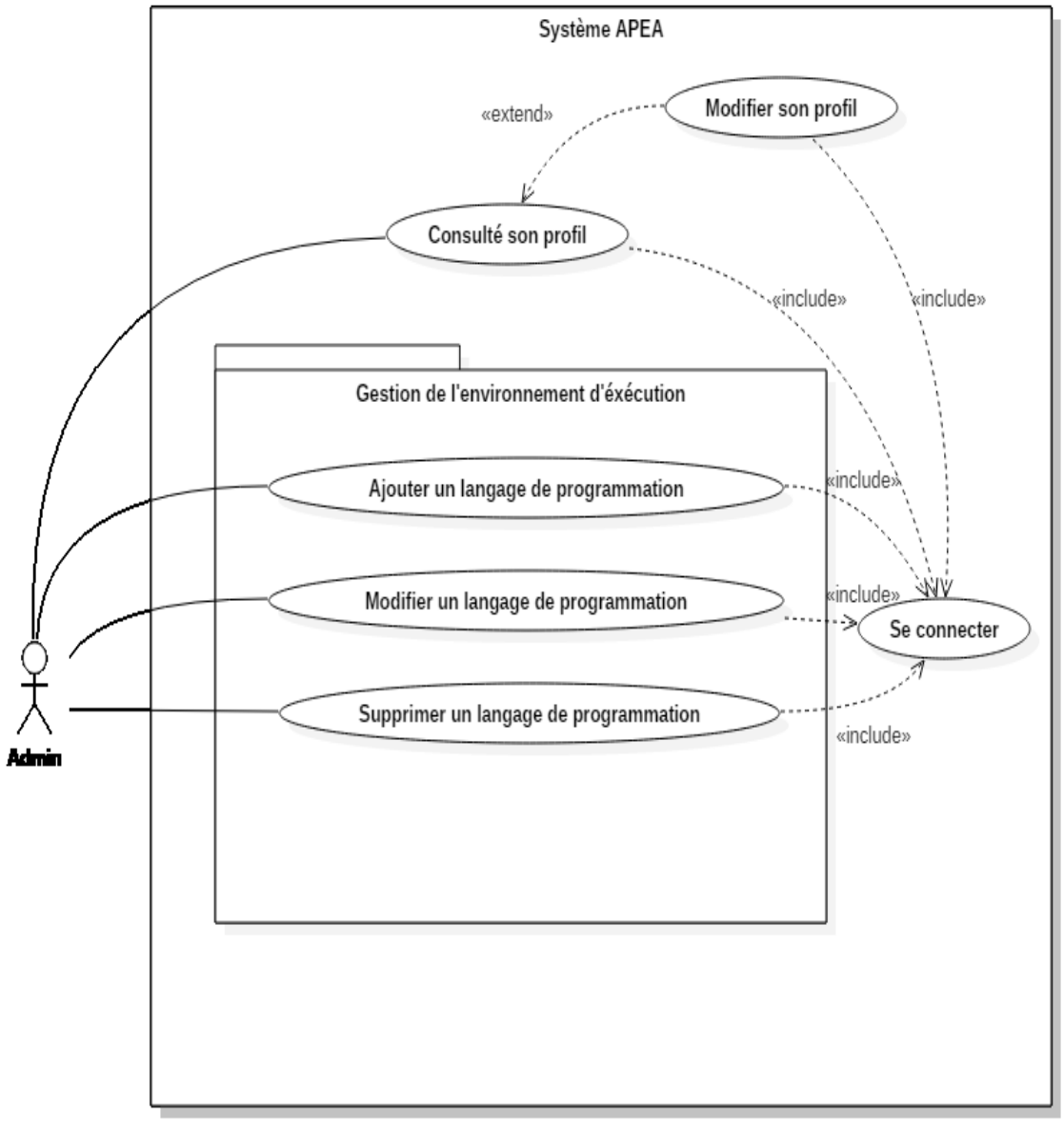


Figure 8 : Cas d'utilisation du paquetage "Gestion de l'environnement d'exécution"

Paquetage 3 : Gestion des parcours d'apprentissage

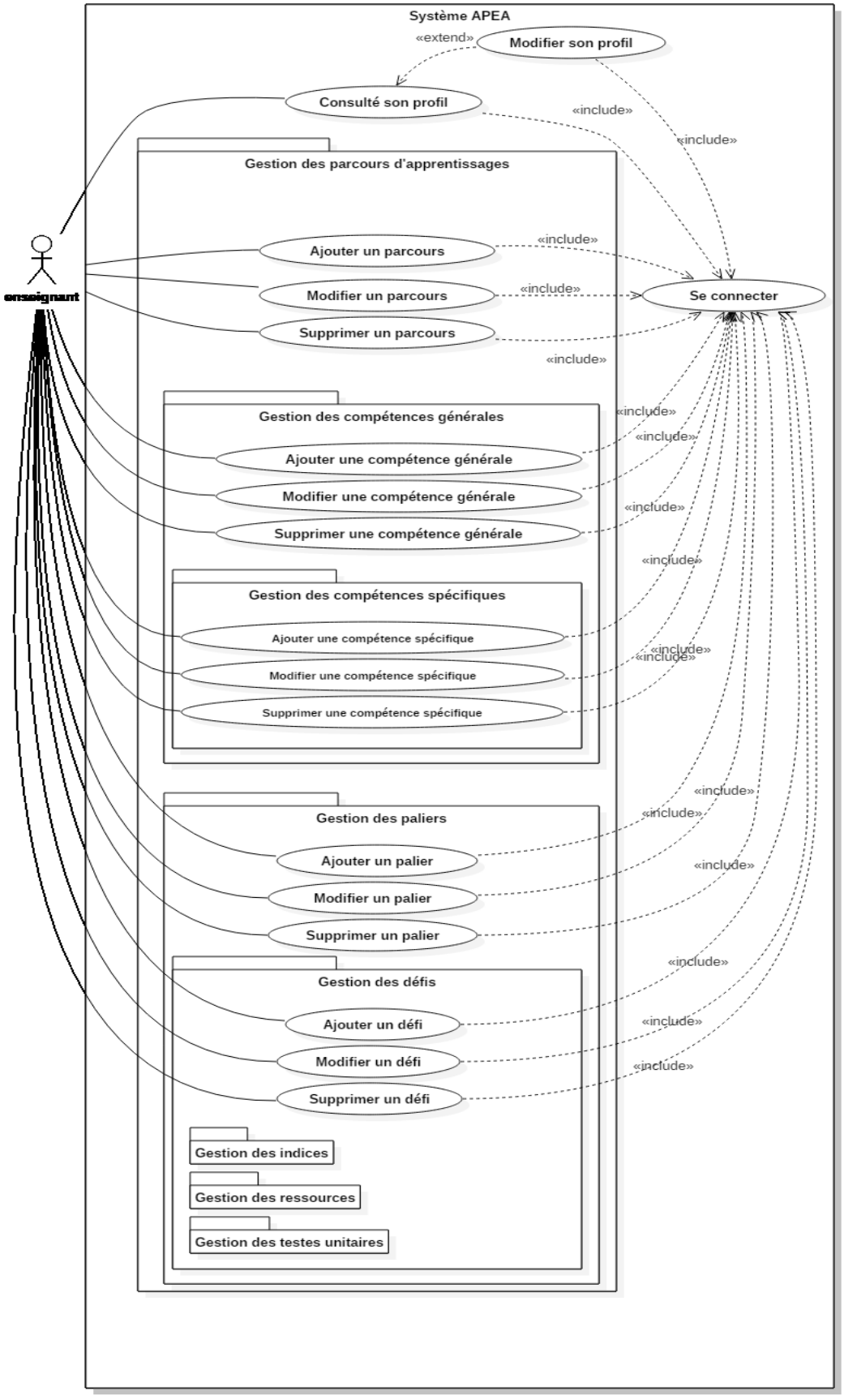


Figure 9 : Cas d'utilisation du paquetage « Gestion des parcours d'apprentissage »

Paquetage 4 : Gestion des sessions d'apprentissage

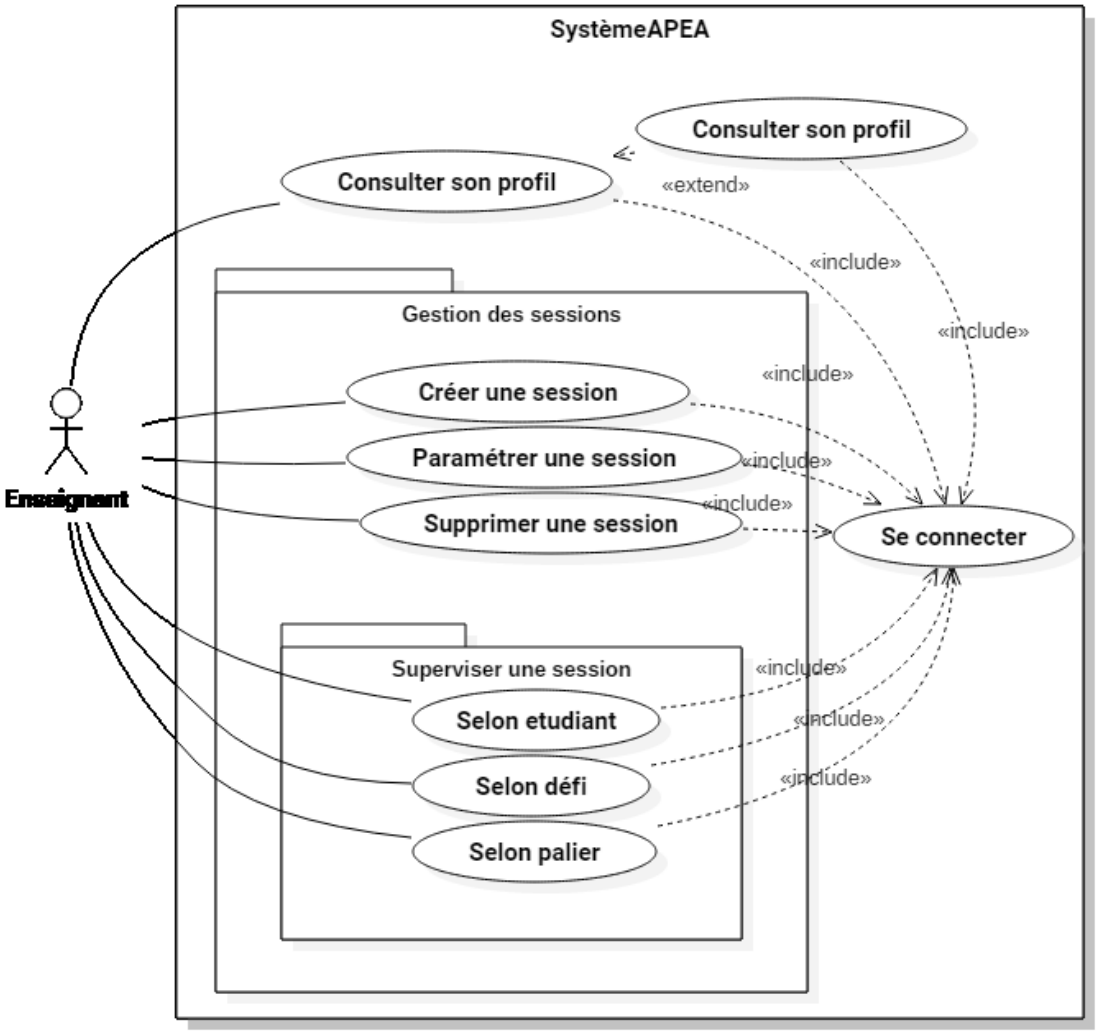


Figure 10 : Cas d'utilisation du paquetage « Gestion des sessions d'apprentissage »

Paquetage 5 : Participation à une session d'apprentissage

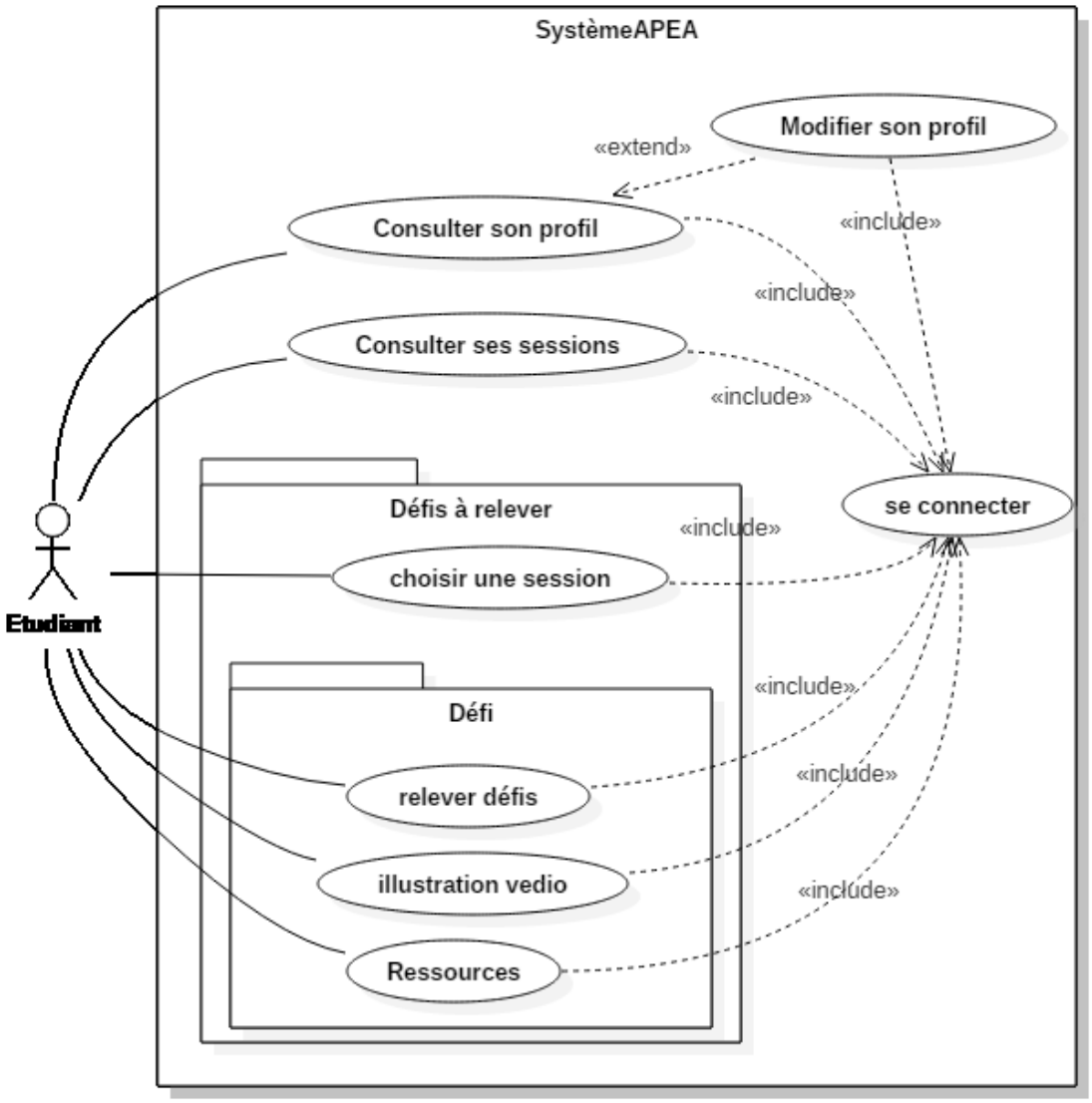


Figure 11 : Cas d'utilisation du paquetage « Participation à une session d'apprentissage »

4.5 Maquettage de l'IHM

Afin d'avoir une idée sur la future application Web que nous devons réaliser, nous avons confectionné en collaboration avec le commanditaire du projet, un maquettage guidé par les cas d'utilisation. Ces écrans sont organisés selon les rôles joués par les utilisateurs.

Gabarit : Afin de garantir une cohérence et une charte d'interaction uniforme, nous avons fixé un gabarit (modèle) de pages comme suit :

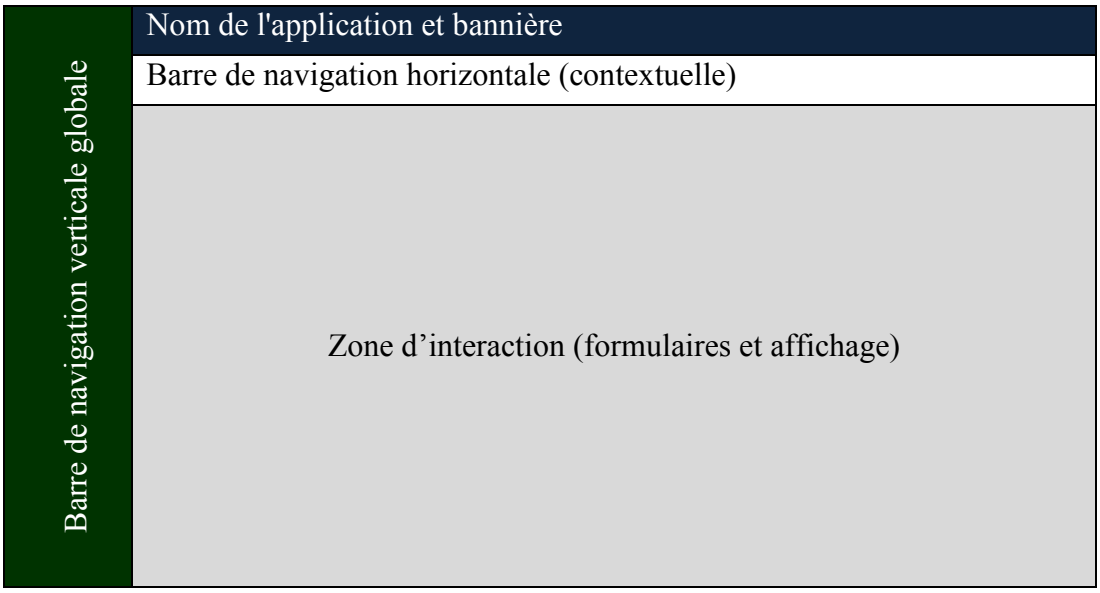


Figure 12 : Gabarit de l'application

La barre de navigation verticale comporte des menus qui seront toujours visible comme le bouton de déconnexion ou d'accès au profil.

La barre de menu horizontale est contextuelle et permet de savoir où on est et de retracer la succession des pages.

La zone centrale est une zone d'interaction où on va avoir des formulaires et des textes d'affichages

Afin d'affiner notre processus de prototypage, en nous basant sur les cas d'utilisation, nous allons d'abord établir l'arborescence des écrans par acteur. Par la suite nous détaillerons chacun des écrans.

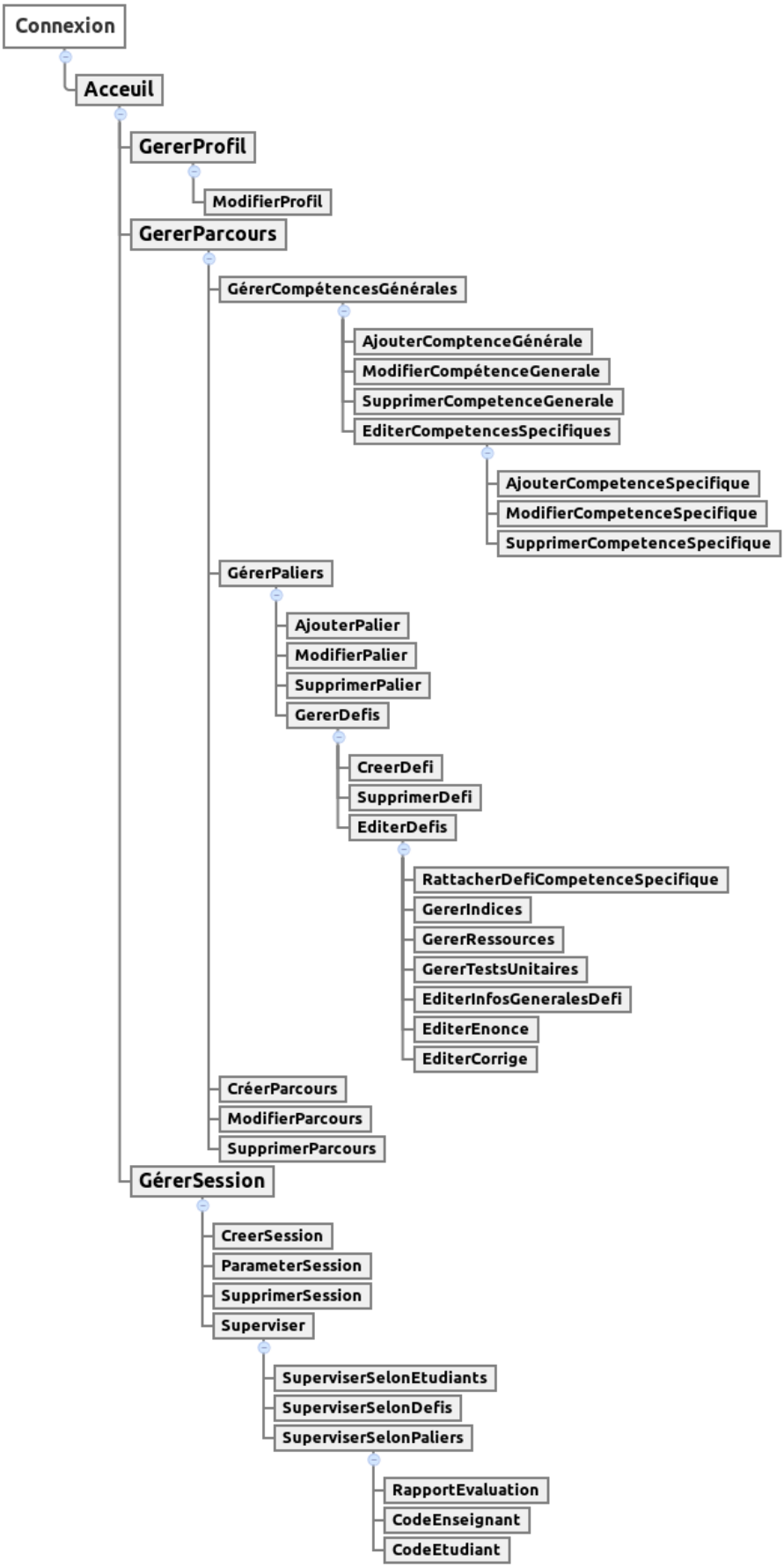


Figure 13 : Structure des écrans IHM (acteur : Enseignant)

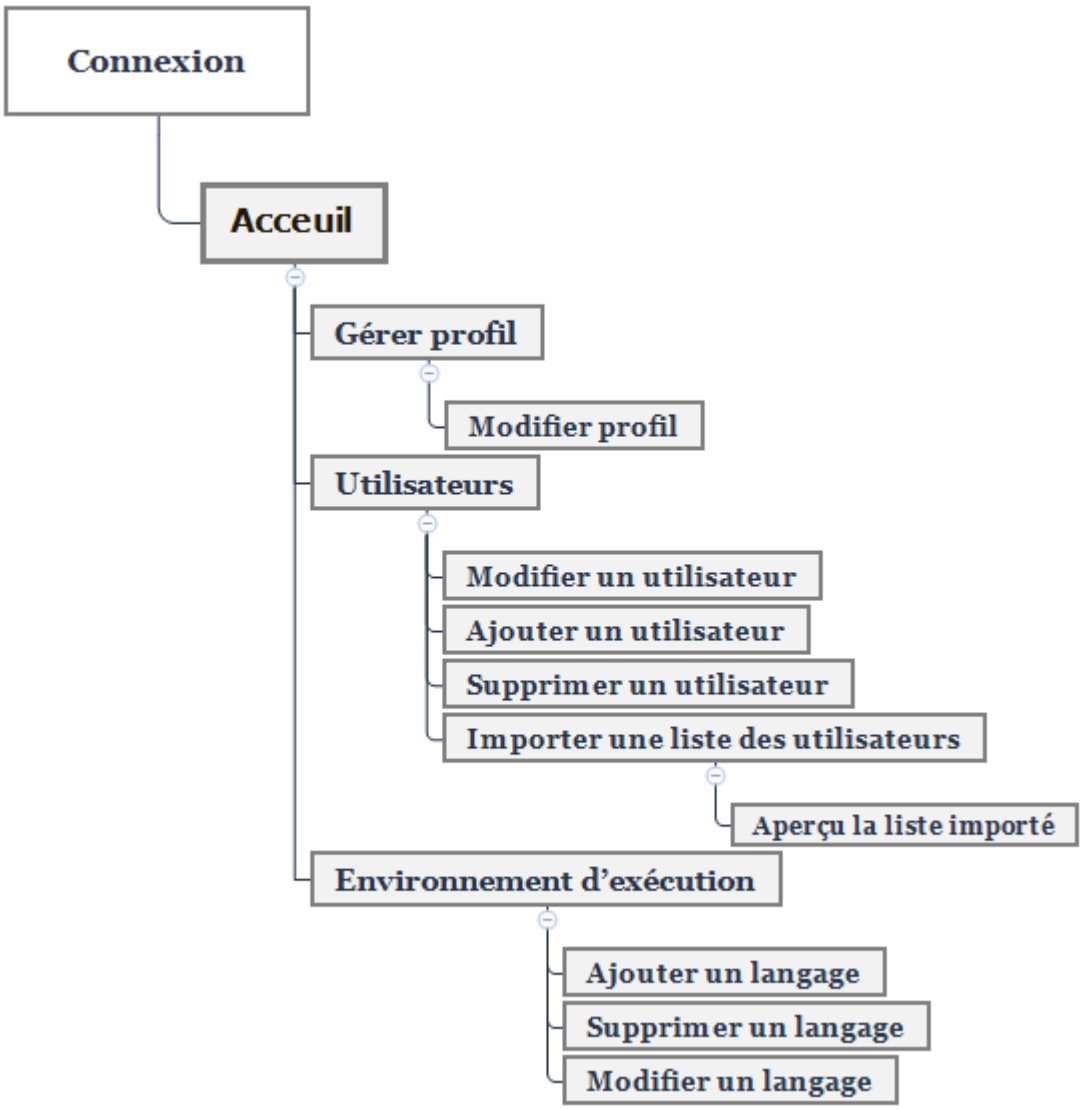


Figure 14 : Structure des écrans IHM (acteur : Admin)

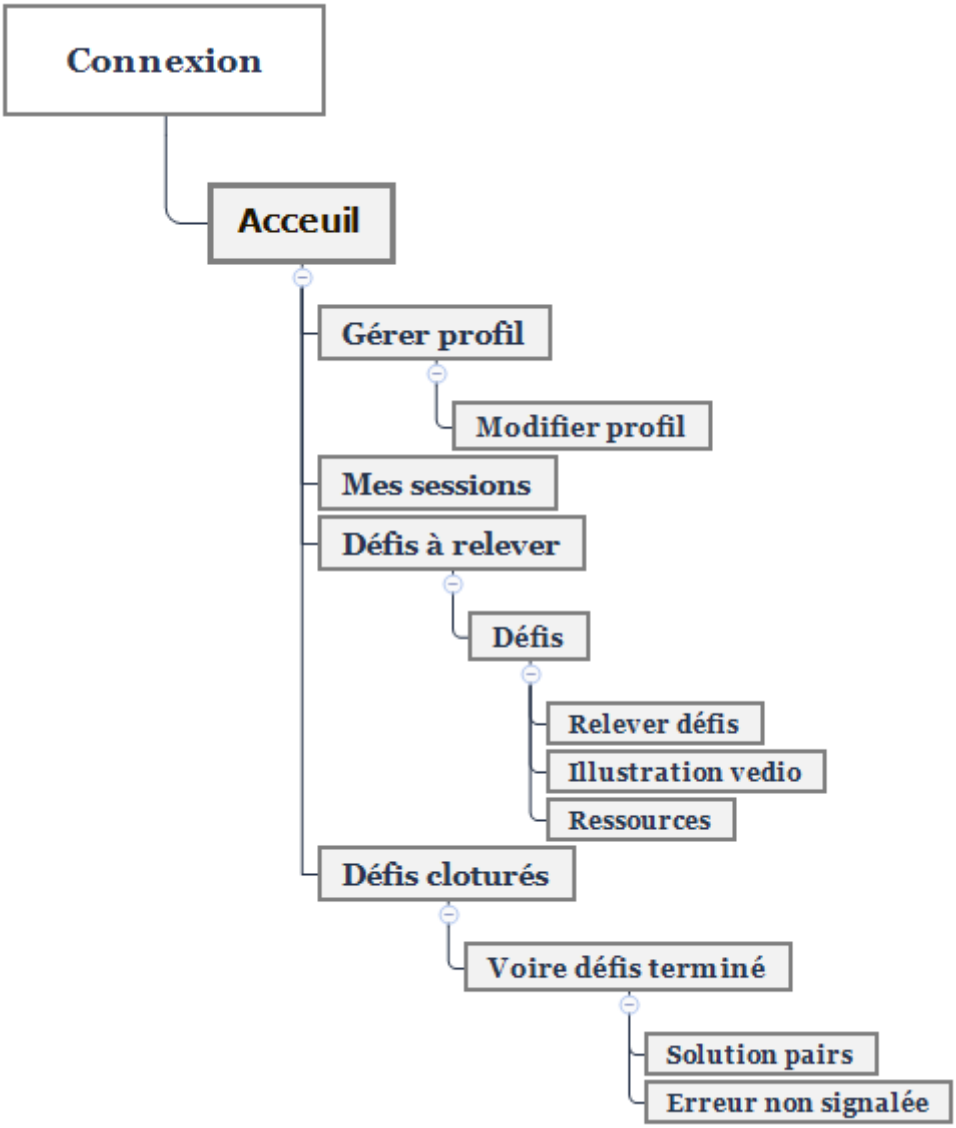


Figure 15 : Structure des écrans IHM (acteur : Etudiant)

Description détaillée de quelques maquettes : Dans ce qui suit, nous allons décrire quelques maquettes IHM que nous avons élaboré. Pour les autres, on renvoi le lecteur aux annexes 1.

La page d'accueil et d'authentification : Cas d'utilisation : « Authentification »

Maquette	Description textuelle
	-Cet écran représente la page d'accueil pour un utilisateur anonyme, elle est organiser en trois zones : menu, affichage et la zone d'interaction permettant l'authentification.

Tableau 1 : Page d'accueil et authentification

Maquettes associées à l'administrateur

Maquette	Description textuelle
	-Cet écran représente la page sur la gestion des utilisateurs. -L'admin accèdera à cette page avec un simple clic sur le boutton Utilisateur dan le menu. -L'admin peut éffectuer une mise a jour sur les utilisateurs et aussi la possibilité d'importer une liste des utilisateurs.

Tableau 2 : IHM page admin la gestion des utilisateurs

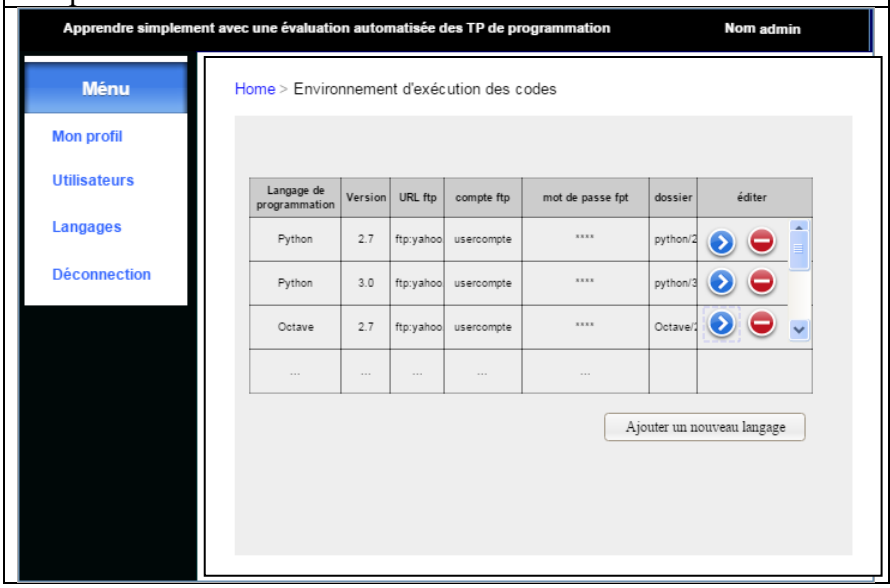
Maquette	Description textuelle
	-Cet écran représente la page sur la gestion des langages. --L'admin accèdera à cette page avec un simple clic sur le boutton langages dans le Menu. -L'admin peut ajouter un nouveau langage ou le modifier ou bien le supprimer.

Tableau 3 : IHM page admin gestion des langages

Maquettes associées à un enseignant

Maquette	Description textuelle
	-Cet écran represente la page sur la gestion des parcours. - l'enseignant accèdera à cette page en survolant le lien « Parcours d'apprentissage ». -Le tableau représente la liste des parcours. -L'enseignant a le choix d'ajouter des parcours ou les supprimer ou bien d'accédé a la gestion des paliers.

Tableau 4 : IHM page enseignant gestion des parcours d'apprentissage

Maquette	Description textuelle
	-Cet écran représente la page sur la gestion des sessions. -L'enseignant accèdera à cette page avec un clic sur le lien Mes session dans le Ménu. -L'enseignant peut éffectuer une mise a jour pour ces sessions avec la possibilité de choisir la liste des étudiants.

Tableau 5 : IHM page enseignant gestion des sessions d'apprentissage

Maquettes associées à un étudiant

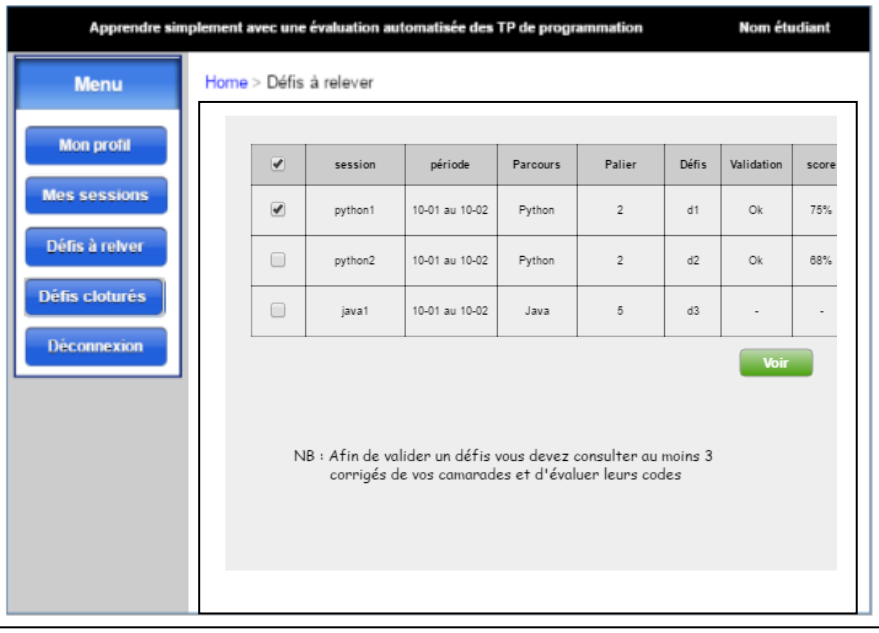
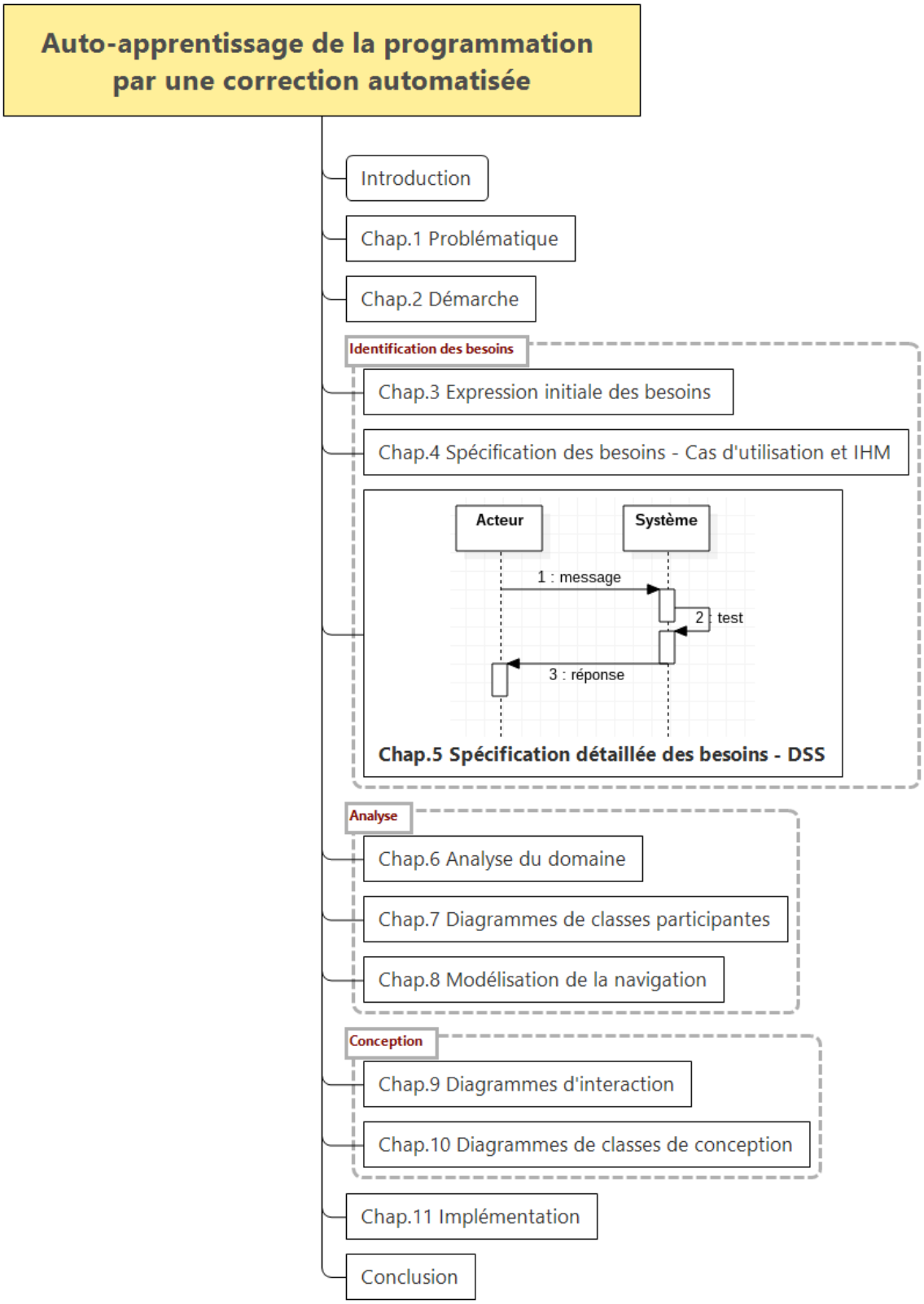
Maquette	Description textuelle
	-Cet écran représente la page défi à relever. -Le tableau représente l'ensemble des parcours ou l'étudiant est déjà inscrit. -L'étudiant doit coché un parcours pour ensuite résoudre un défi

Tableau 6 : IHM page étudiant découvrir le code secret

Chap.5– Spécification détaillée des besoins



5.1 Positionnement de l'étape

Rappelons le positionnement de cette activité de spécification des besoins par rapport à l'ensemble du processus que nous avons suivi : Les cas d'utilisation décrivent les interactions des acteurs avec le site web que nous voulons spécifier et concevoir. Lors de ces interactions, les acteurs produisent des messages qui affectent le système informatique et appellent généralement une réponse de celui-ci. Nous allons isoler ces messages et les représenter graphiquement sur des diagrammes de séquence UML.

5.2 Les diagrammes de séquences système

Dans ce qui suit, nous présenterons, à travers des diagrammes de séquences système, la spécification des besoins pour les cas d'utilisation suivants : Authentification, gérer les utilisateurs, gérer des parcours d'apprentissage et participer à une session d'apprentissage. Pour les autres cas d'utilisation, nous renvoyons le lecteur aux annexes 2.

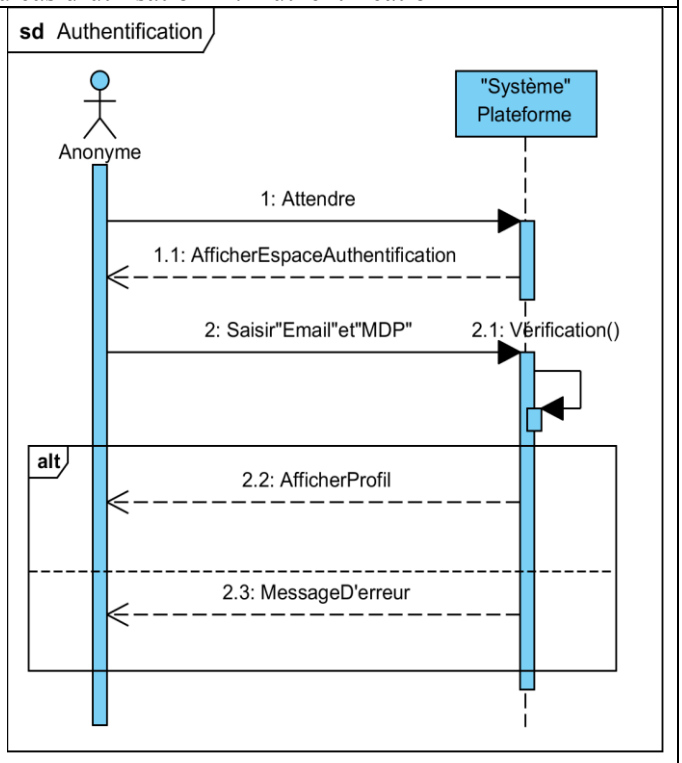
DSS1 - Diagramme de séquence système du cas d'utilisation 1 : Authentification	
Acteurs concernés : Anonyme	
Explication des interactions	
1. L'anonyme accède à la page d'accueil de la plate-forme par le biais d'un navigateur Web. 1.1. Le système répond en affichant (par le biais du navigateur) la page d'accueil, y compris un espace d'authentification. 2. L'anonyme saisit son email et son mot de passe. 2.1. Le système accède à la BDD pour la vérification des données. 2.2 Le système affiche le profil. 2.3 Le système affiche un message d'erreur si l'email et/ou son mot de passe sont incorrects.	

Tableau 7 : DSS du cas d'utilisation : Authentification (acteur : Anonyme).

Remarque :
Sur la figure, la flèche pointillée partant du système à l'étude représente un retour au sens UML. Cela signifie que le message en question (par exemple : Gérer les utilisateurs) est le résultat direct du message précédent par une relation forte de cause à effet.

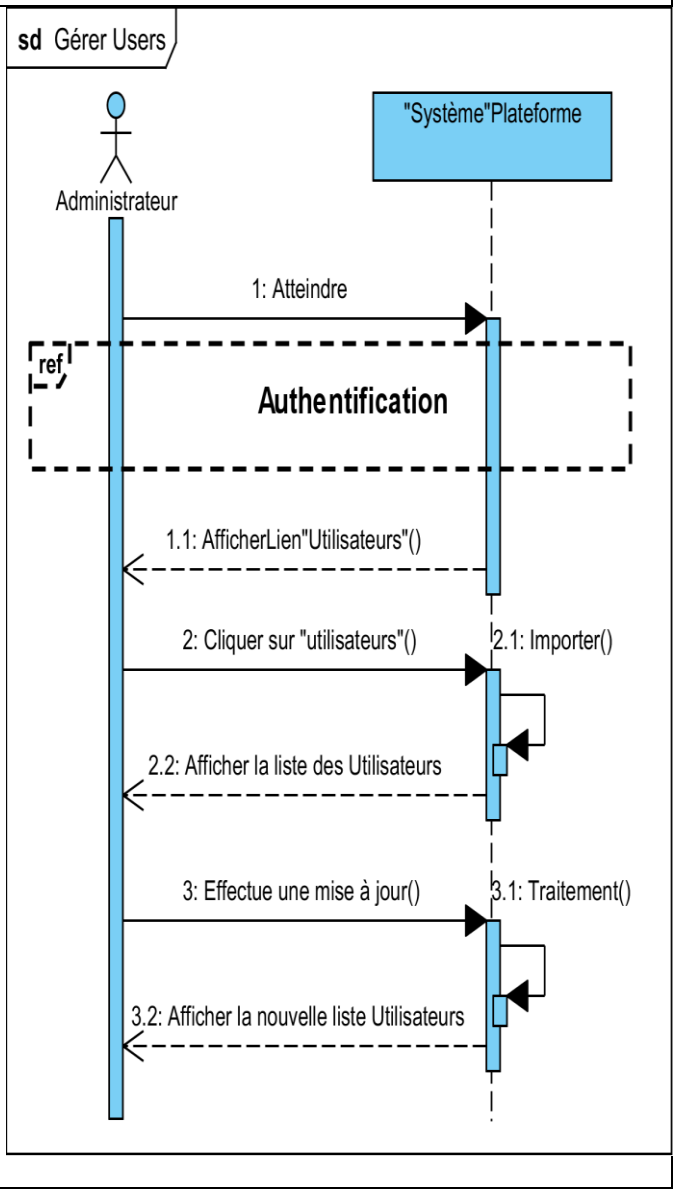
DSS11 - Diagramme de séquence système du cas d'utilisation 11 : Gérer les Utilisateurs.	
Acteurs concernés : Administrateur	
Explication des interactions	
1. L'administrateur accède à la page d'accueil de la plateforme par le biais d'un navigateur Web. 1.1 Le système répond en affichant (par le biais du navigateur) la page d'accueil comportant la zone « profil administrateur » qui à son tour comporte le lien « Gestion utilisateurs ». 2. L'administrateur clique sur le lien gérer les utilisateurs. 2.1. Le système accède à la BDD pour récupérer la liste de tous les utilisateurs. 2.2. Le système affiche les listes des utilisateurs. 3. L'administrateur demande d'effectuer une opération sur les utilisateurs 3.1 Le système accède à la BDD pour mettre à jour la table correspondante. 3.2. Le système affiche les nouvelles listes d'utilisateurs.	

Tableau 8 : DSS du cas d'utilisation : Gérer les utilisateurs

DSS3 - Diagramme de séquence système du cas d'utilisation Gérer Parcours

Acteurs concernés : Enseignant

Explication des interactions

1 : L'enseignant accède à la page d'accueil de la plate-forme.

1.1 Le système répond en affichant (par le biais du navigateur) la page d'accueil comportant la zone « Menu » qui à son tour comporte le lien « Mes parcours ».

2. il va cliquer sur le lien.

2.1. Le système accède à la BDD pour récupérer la liste de tous les parcours.

2.2. Le système affiche la liste des parcours.

3. L'enseignant effectuer les mises à jour.

3.1 Le système accède à la BDD pour mettre à jour ces informations.

3.2. Le système affiche la nouvelle liste des parcours.

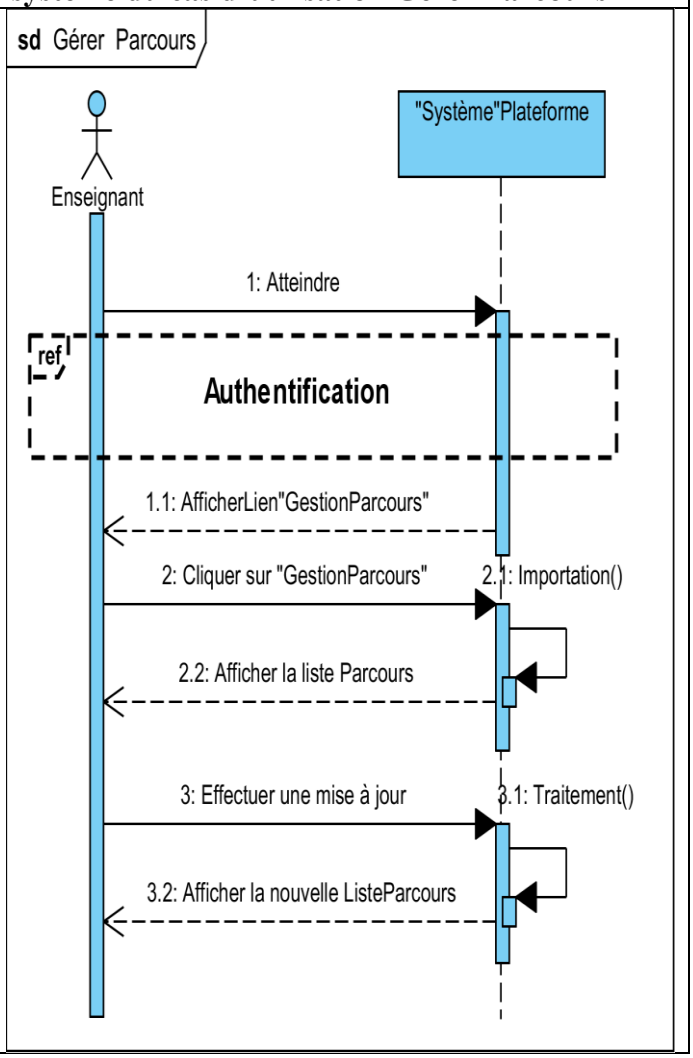


Tableau 9 : DSS du cas d'utilisation : Gérer les parcours

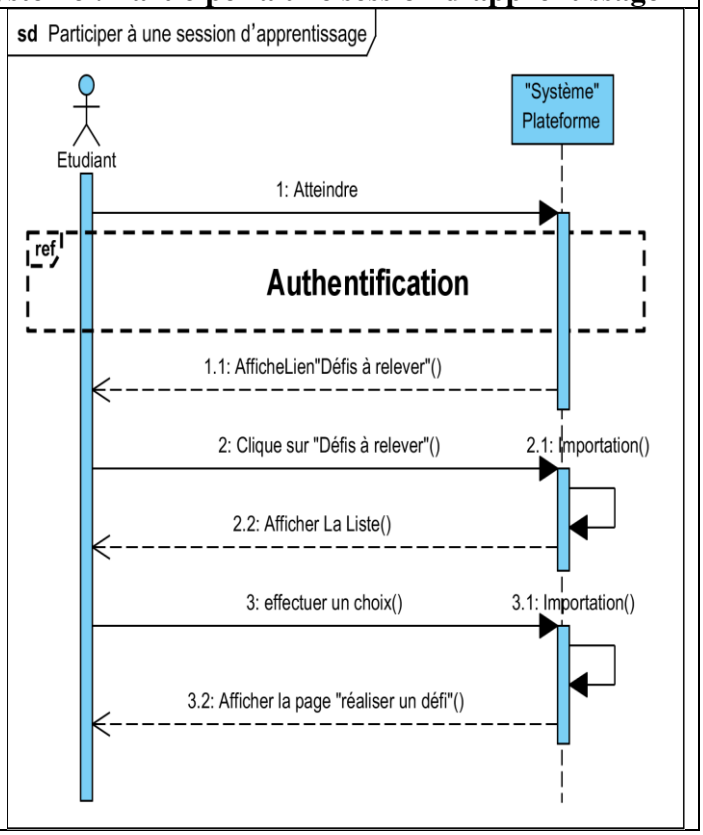
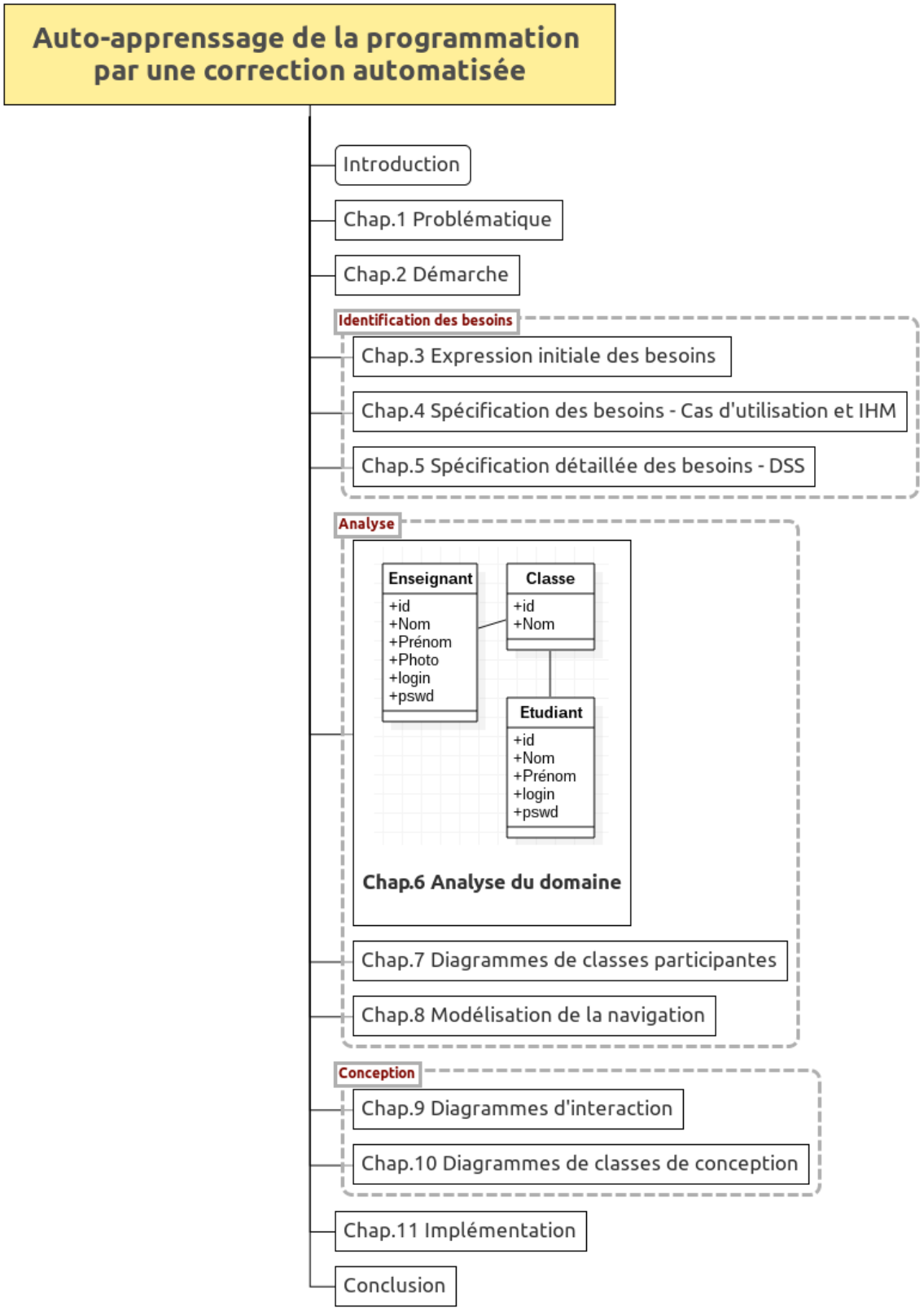
DSS3 - Diagramme de séquence système : Participer à une session d'apprentissage	
Acteurs concernés : Etudiant Explication des interactions 1 : L'enseignant accède à la page d'accueil de la plate-forme. 1.1 Le système répond en affichant (par le biais du navigateur) la page d'accueil comportant la zone « Menu » qui à son tour comporte le lien «Défis à relever ». 2. il va cliquer sur le lien. 2.1. Le système accède à la BDD pour récupérer la liste de ces sessions. 2.2. Le système affiche la liste. 3. l'étudiant effectuer un choix. 3.1 Le système accède à la BDD pour récupérer la liste de choix. 3.2 Le système affiche la page.	

Tableau 10 : DSS du cas d'utilisation : Participer à une session d'apprentissage

Chap.6–Analyse du domaine



6.1 Positionnement de l'étape

Rappelons le positionnement de cette activité par rapport à l'ensemble du processus que nous avons suivi. Nous avons déjà établi l'expression intiales des besoins que nous avons raffiné par la spécification des cas d'utilisation et d'un maquetage d'IHM. Nous allons maintenant procéder à l'analyse du domaine. En d'autres termes, nous allons analyser chaque cas d'utilisation et identifier tous les concepts du domaine métier auquel nous nous intéressons. Dans ce qui suit, nous allons décrire les classes majeures d'analyse de domaine pour chaque cas d'utilisation. En plus détaillé, nous allons identifier (pour chaque cas d'utilisation) les concepts ou classes du domaine (les entités) avec leurs attributs puis les associations entre ces classes.

6.2 Analyse de domaine pour chaque cas d'utilisation

Nous nous limiterons à décrire, ci-dessous, l'analyse de domaines pour 3 cas d'utilisation : « gérer les utilisateurs », « gestion des parcours d'apprentissage » et « participation à une session d'apprentissage ».

On notera que nous avons choisi de décrire un cas d'utilisation impliquant un acteur différent à chaque fois :

- Coté administrateur : gérer les utilisateurs
- Coté enseignant : gestion des parcours d'apprentissage
- Coté étudiant. : Participation à une session d'apprentissage

Concepts et attributs liés à la gestion des utilisateurs :

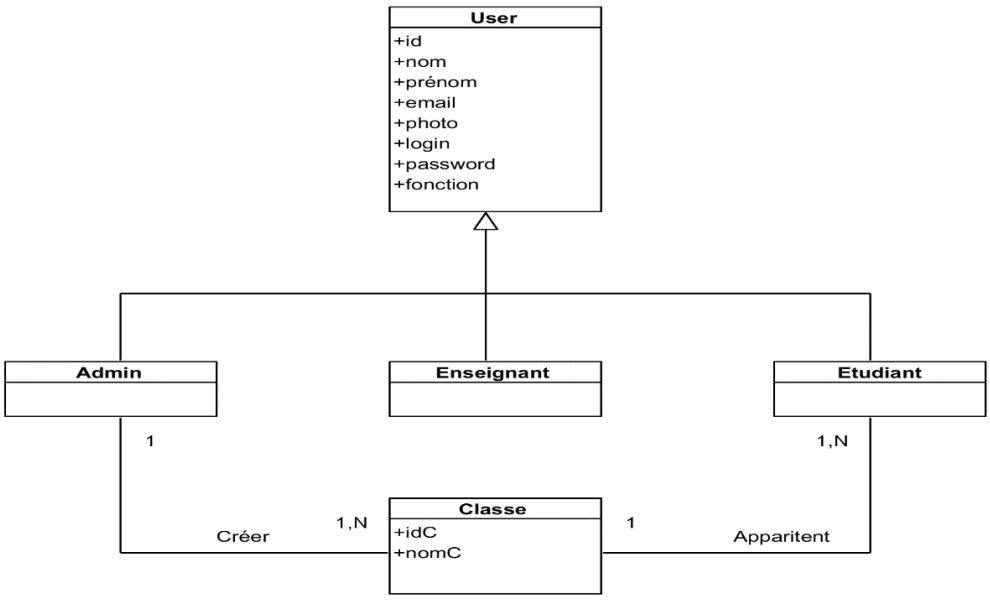


Figure 16 : Concepts et attributs liés à la gestion des utilisateurs

Dans la figure ci-dessus les administrateurs, les enseignants et les étudiants sont des utilisateurs (Relation d'héritage). Ils possèdent tous un identifiant, un nom, un prénom, etc. Comme on peut constater, l'administrateur s'occupe de la création d'une ou plusieurs classes, une classe contient un ou plusieurs étudiants.

Concepts et attributs liés à la gestion des parcours d'apprentissage

Pour ce cas d'utilisation, nous avons identifié plusieurs classes d'entités : *parcours*, *compétences générale*, *compétence spécifique*, *paliers*, *défis*, *indices*, *ressources* et *tests unitaires.* En ajoutant les associations et les attributs entres ces classes, nous avons obtenu le diagramme suivant :

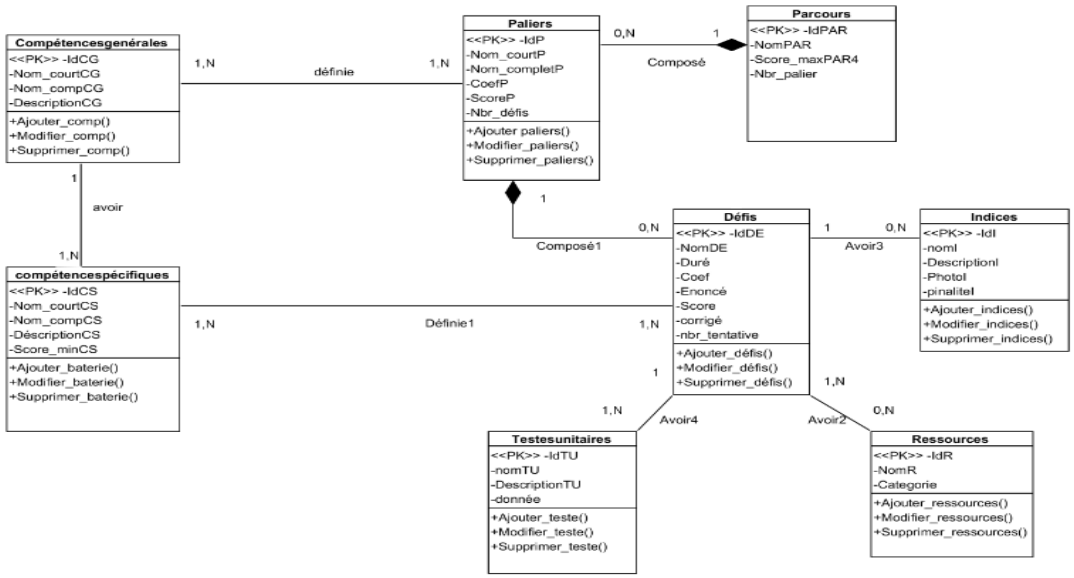


Figure 17 : Concepts du domaine liés à la gestion des parcours d'apprentissage

Comme on peut le constaté dans la figure ci-dessus, un parcours est composé de plusieurs paliers, un palier est associer avec les compétences générales, un palier est composé de plusieurs défis et pour chaque défi l'enseignant peut lui rattacher des indices, des ressources, des tests unitaires et aussi des compétences spécifiques. Comme on peut remarquer que toutes les entités sont identifiées avec un identifiant et quelques attributs qui les décrivent spécifiquement.

Concepts et attributs liés à la participation à une session d'apprentissage

Pour ce cas d'utilisation, nous avons identifié plusieurs classes d'entités : *Sessions*, *défis*, *ressources* et *indices*. L'étudiant pourra participer à une session d'apprentissage pour résoudre les défis de programmation qui ont été proposés par son enseignant. Pour la résolution de ces défis l'étudiant doit consulter les données et l'ensemble des ressources pour ensuite écrire son propre code et le soumettre pour correction automatisée. Avec ce concept si l'étudiant na pas peu résoudre son défi, il pourra le réessayer à nouveau (plusieurs tentative) et aussi il pourra demander des indices.

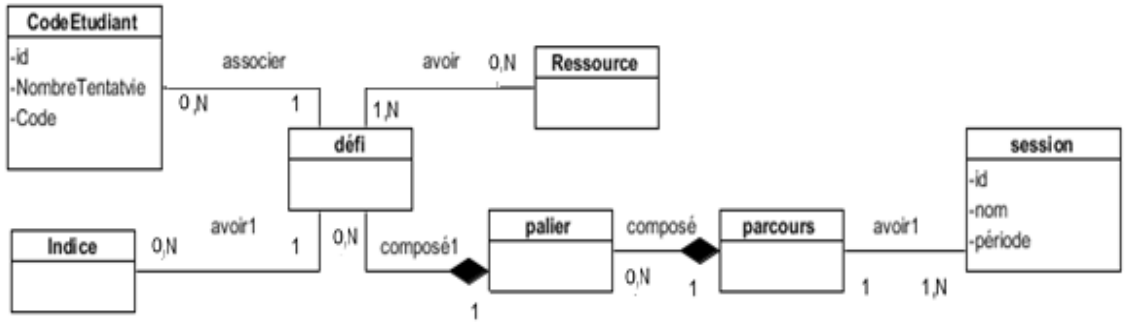


Figure 18 : Concepts et attributs liés à la participation à une session d'apprentissage

6.3 Diagramme de classes d'entité

En nous basons sur l'analyse de domaine associée à chaque cas d'utilisation, nous allons procéder à un raffinement, en établissant un diagramme de classe d'entité synthétique pour toute notre application. On commence par recenser les concepts que nous avons dégagés à travers l'analyse de domaine qu'on a déjà présenté pour chaque cas d'utilisation. Puis nous recensons tous les attributs de chaque entité. Enfin, nous dressons un diagramme de classe d'entité valable pour toute note application.

Identification des concepts du domaine :

Selon les cas d'utilisations	Concepts impliqués
Authentification	Utilisateur, administrateur, enseignant, et étudiant
Gestion des utilisateurs	Enseignant, étudiant, classe et administrateur
Gestion de l'environnement d'exécution	Administrateur et langage
Gestion des parcours d'apprentissage	Enseignant, parcours, compétence générale, compétence spécifique, palier, défi, ressource, indice et les tests unitaires.
Gestion des sessions d'apprentissage	Enseignant, session
Participation a une session d'apprentissage	Etudiant, code, défi, feedback

Tableau 11 : Identifier des concepts du domaine

Identification des attributs :

Nom de la classe	Codification	désignation	Type	Langueur	observation
Classe	IdC	Identifiant de la classe	N	11	Auto-incrémente
	NomC	Nom de la classe	AN	12	/
Sessions	IdS	Identifiant d'une session	N	11	Auto-incrémente
	NomS	Nom de session	A	30	/
	Période_d	Période de début d'une session	D	/	AAAA/MM/JJ
	Periode_fin	Période de fin d'une session	D	/	AAAA/MM/JJ
Parcours	IdPAR	Identifiant d'un parcours	N	11	Auto-incrémente
	NomPAR	Nom de parcours	A	30	/
	Score_maxPAR	Score maximum d'un parcours	N	4	/
	Nbr_palier	Nombre de palier	N	2	/
Paliers	IdP	Identifiant d'un palier	N	11	Auto-incrémente
	Nom_courtP	Nom court de palier	A	50	/
	Nom_compP	Nom complet de palier	A	50	/
	CoefP	Coefficient d'un palier	N	2	/
	ScoreP	Score d'un palier	N	4	/
	Nbr_defi	Nombre de défis	N	2	/
CompétencesGénérale	IdCG	Identifiant d'une compétence	N	11	Auto-incrémente
	Nom_courtCG	Nom court d'une compétence	A	30	/
	Nom_compCG	Nom complet d'une compétence	A	50	/
	DescriptionCG	Description de la compétence	A	60	/
Ressources	IdR	Identifiant d'une ressource	N	11	Auto-incrémente
	NomR	Nom de la ressource	A	30	/
	Catégorie	Catégorie de la ressource	A	30	/
Défis	IdDE	Identifiant d'un défi	N	11	Auto-incrémente
	NomDE	Nom d'un défi	A	30	/
	Duré	Duré d'un défi	AN	11	/
	Coef	Coefecient d'un défi	N	2	/
	enoncé	Enoncé d'un défi	AN	255	/
	score	Score d'un défi	N	4	/
	corrige	Corrigé d'un défi	A	255	/
CompétencesSpécifique	IdCS	Identifiant d'une compétence	N	11	Auto-incrémente
	Nom_courtCS	Nom court d'une compétence	A	30	/
	Nom_compCS	Nom complet d'une compétence	A	50	/
	DescriptionCS	Description d'une compétence	A	60	/
	Score_minCS	Score minimal	N	4	/
Indices	IdI	Identifiant de l'indice	A	11	Auto-incrémente
	DescriptionI	Description de l'indice	AN	60	/
	PhotoI	Lien d'une photo	AN	255	

Tableau 12 : Identification des attributs

Nom de la classe	Codification	désignation	Type	Langueur	observation
Users	Id	Identifiant d'un utilisateur	N	11	Auto-incrémente
	Nom	Nom d'un utilisateur	A	30	/
	Prénom	Prénom d'un utilisateur	A	30	/
	Email	Email d'un utilisateur	AN	50	/
	Photo	Photo d'un utilisateur	A	60	/
	Login	Login d'un utilisateur	AN	30	/
	Password	Mot de passe d'un utilisateur	AN	255	/
	Role	Fonction de l'utilisateur	A	20	/
	Created	La date de création d'un user	D	/	AAAA/MM/JJ
	Modified	La date de modification d'un user	D	/	AAAA/MM/JJ
Admin	IdA	Identifiant d'un administrateur	N	11	Auto-incrémente
Etudiants	IdET	Identifiant d'un étudiant	N	11	Auto-incrémente
Enseignants	IdE	Identifiant d'un enseignant	N	11	Auto-incrémente
CodesEtu	Id_etudiant	Identifiant d'un étudiant	N	11	/
	Id_defis	Identifiant d'un défi	N	11	/
	Etat	Etat de défi résolue ou non	A	4	/
	Nbr_tentative	Nombre tentative	N	11	/
	Code	Code de programme	AN	255	/
Langages	IdL	Identifiant d'un langage	N	11	Auto-incrémente
	NomL	Nom d'un langage	A	30	/
	Version	Version d'un langage	AN	4	/
	Dossier	Chemin d'un dossier	A	60	/
	URLftp	URL de ftp	AN	60	/
	CompteFTP	Compte de l'admin	AN	40	/
	PasswordFTP	Mot de passe de l'admin	AN	40	/
feedbacks	Clarté	La clarté de code	A	30	/
	Performance	La performance de code	A	30	/
	Taille_code	La taille de code	N	11	/
	Classement	Classement de l'étudiant	N	11	/
testeUnitaire	IdTU	Identifiant d'un teste	N	11	Auto-incrémente
	NomTU	Nom de teste	AN	40	/
	DescriptionTU	Description sur le test	AN	60	/
	Donne	Les données de test	AN	60	/

Tableau 13 – Identification des attributs (suite)

Diagramme de classe d'entités :

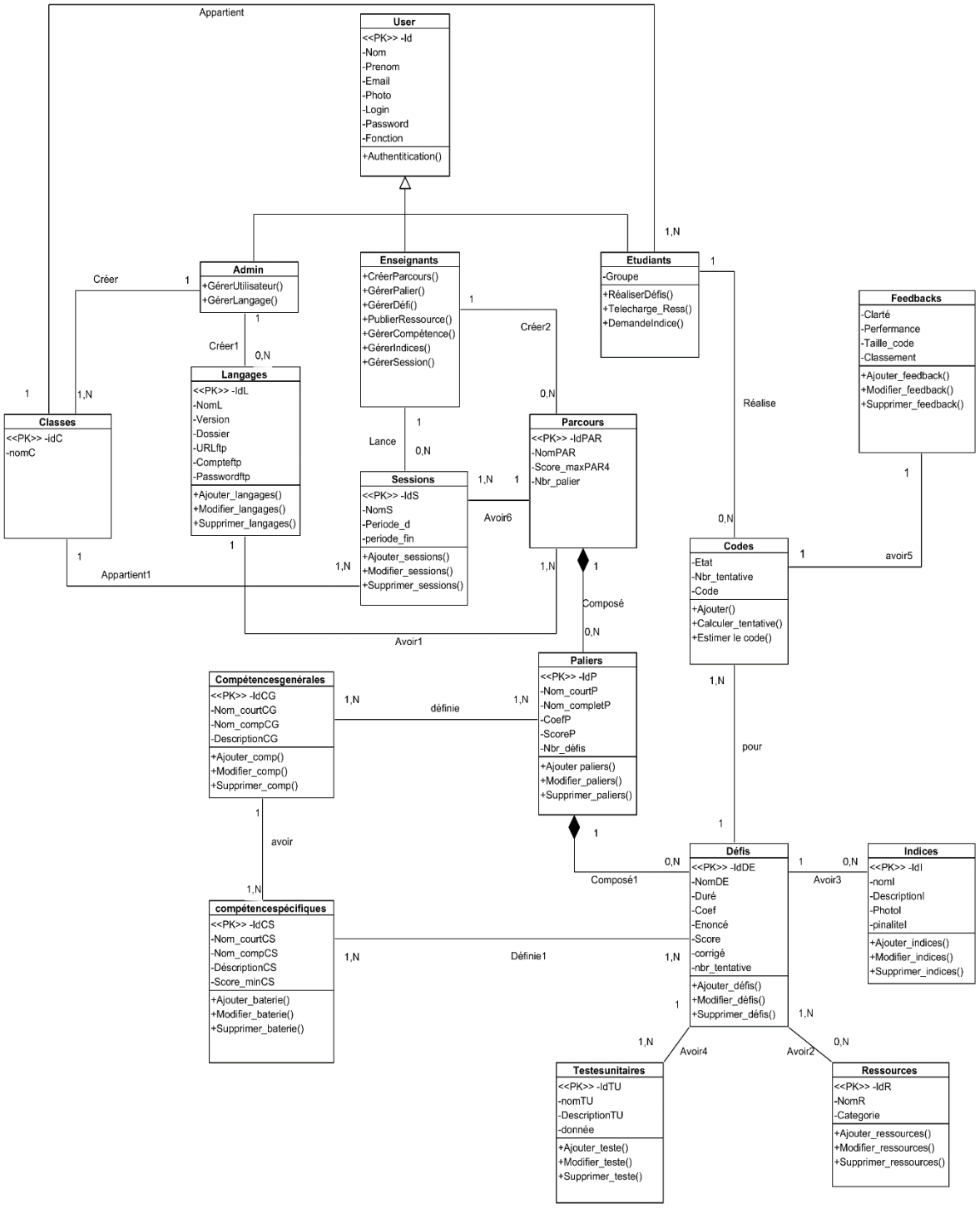
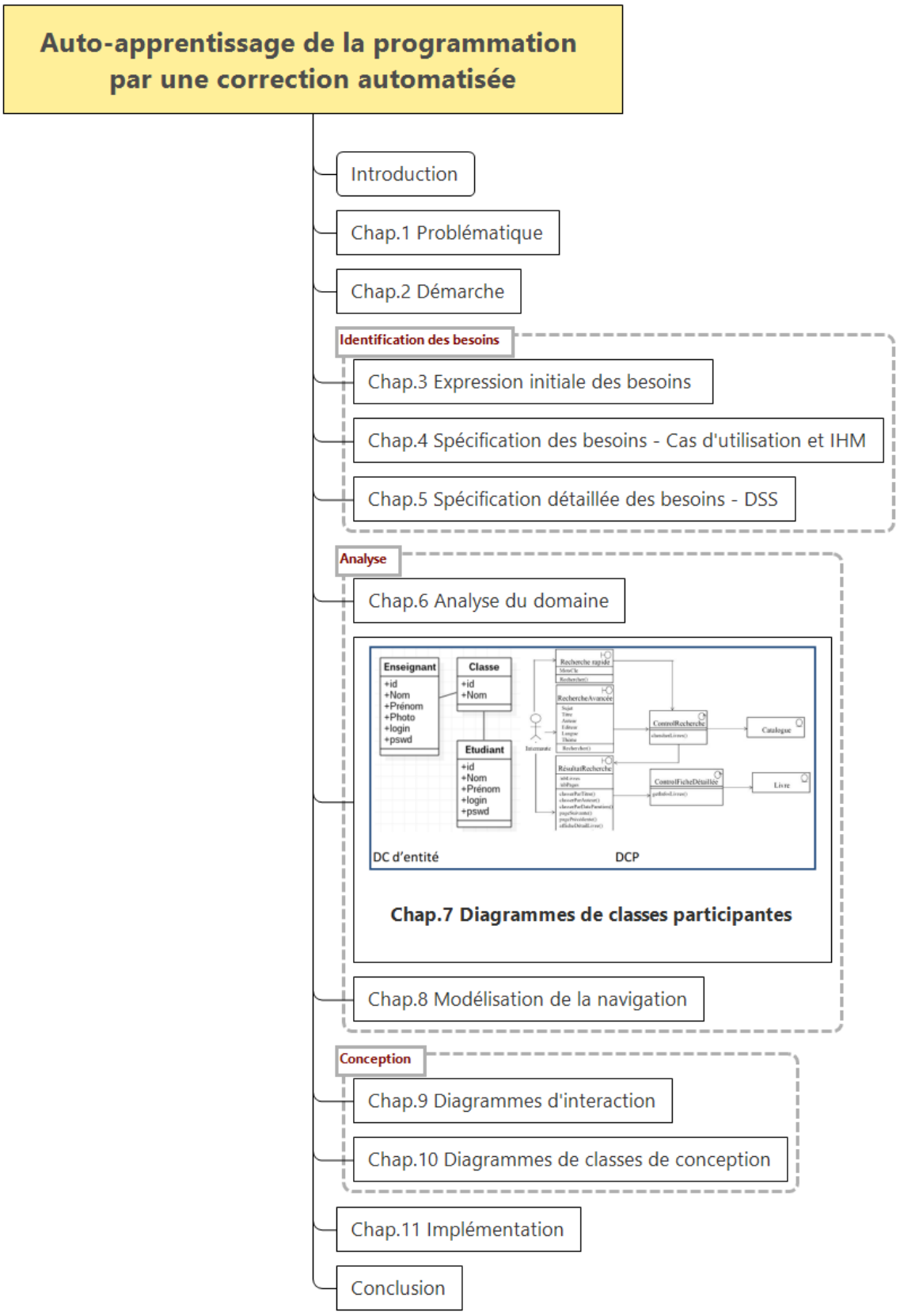


Figure 19 : Diagramme de classe d'entité

Chap.7–Diagrammes de classes participantes



7.1 Positionnement de l'étape

Pour illustrer notre démarche, nous allons identifier les classes d'analyse participantes des principaux cas d'utilisation du notre application. Les entités ayant déjà été répertoriées dans la section traitant des concepts du l'analyse de domaine, nous allons donc ajouter les dialogues et les contrôles et réaliser les diagrammes de classes participantes (DCP) correspondants.

On commence par les dialogues, ce sont les classes qui permettent les interactions entre le site web et ses utilisateurs. Il s'agit typiquement des écrans proposés à l'utilisateur: les formulaires de saisie, les résultats de recherche, etc. Ces classes proviennent directement de l'analyse de la maquette. Il y a au moins un dialogue pour chaque paire acteur-cas d'utilisation.

Les classes qui contiennent la cinématique de l'application sont appelées contrôles. Elles contiennent les règles applicatives et les isolent à la fois des objets d'interface et des données persistantes.

7.2 Raffiner l'analyse : Classes participantes

Dans les figures ci-dessous, nous présenterons les diagrammes de classes participantes (DCP) pour les cas d'utilisation suivants : Authentification « gérer les utilisateurs », « gestion des parcours d'apprentissage » et « participer à une session d'apprentissage ». Pour les autres cas, nous renvoyons le lecteur aux annexes 3.

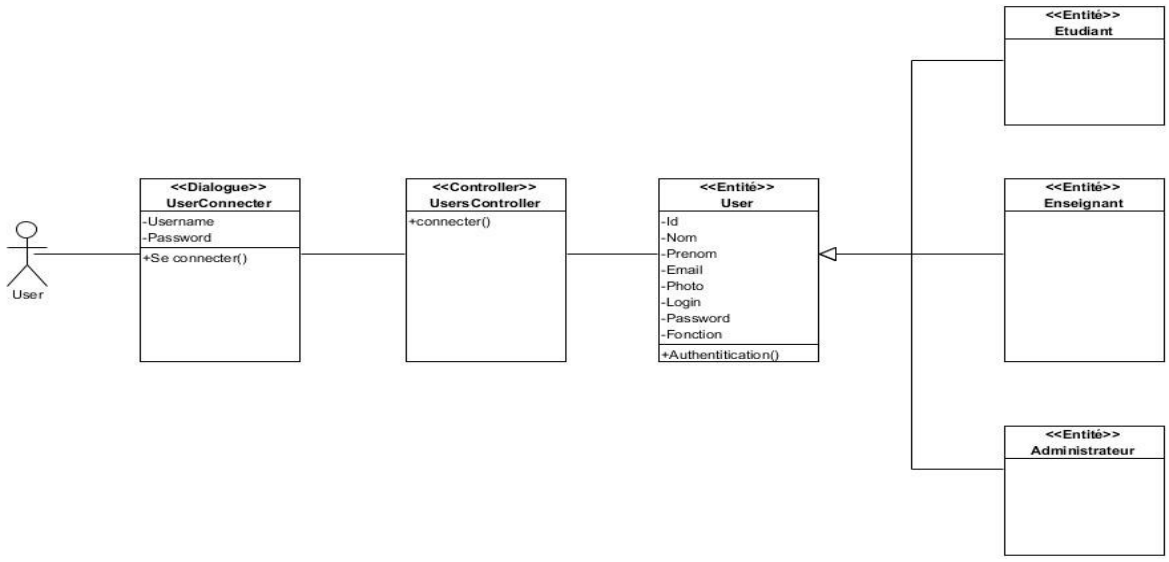


Figure 20 : DCP d'authentification

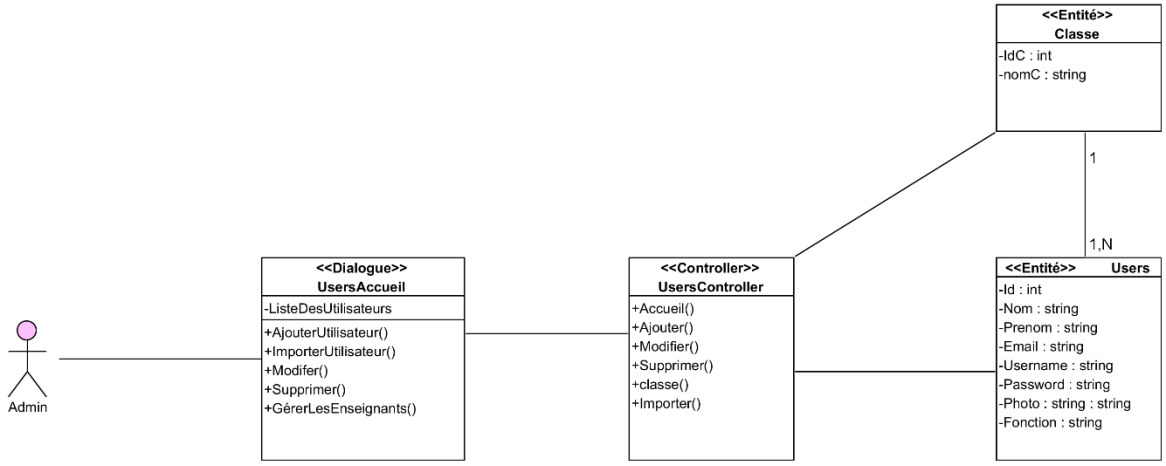


Figure 21 DCP : Gérer les utilisateurs

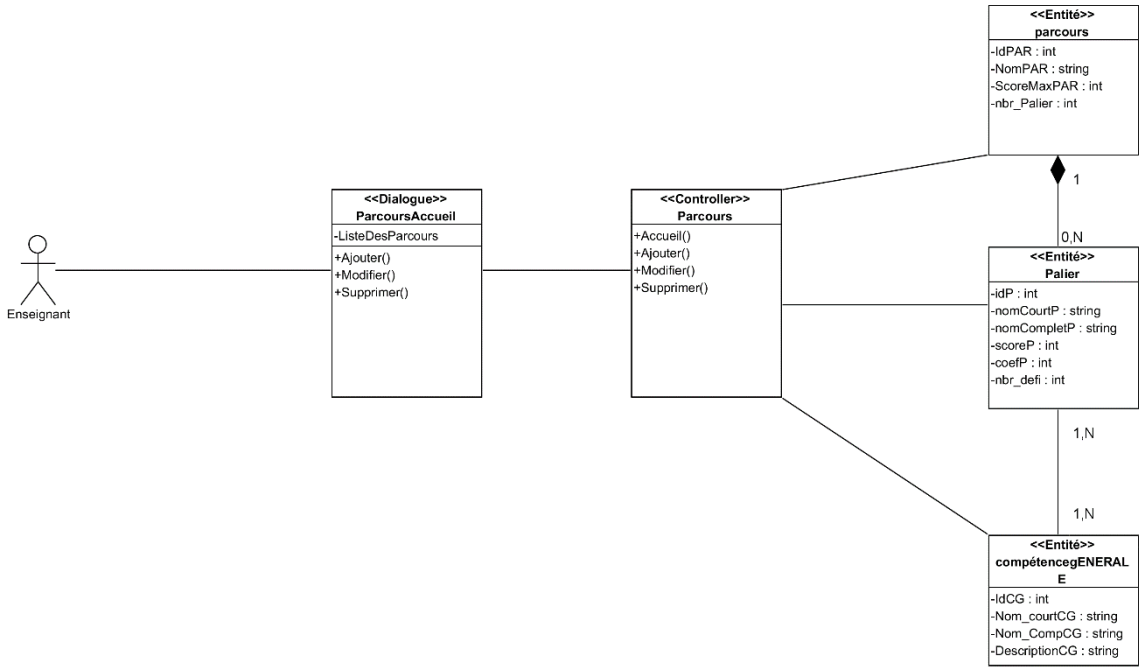


Figure 22 DCP : Gestion des parcours d'apprentissage

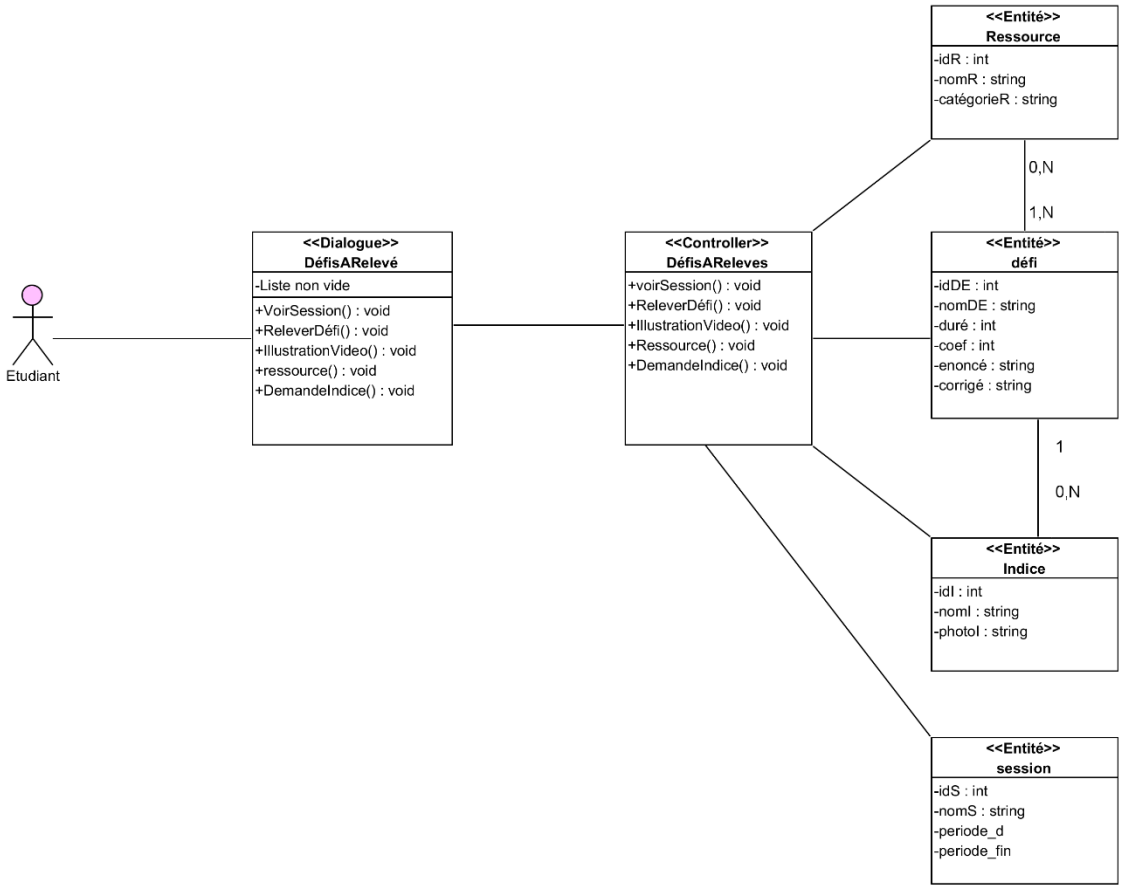


Figure 23 : Diagramme de classe participante participation à une session d'apprentissage

Chap.8 – Modélisation de la navigation

Auto-apprentissage de la programmation par une correction automatisée

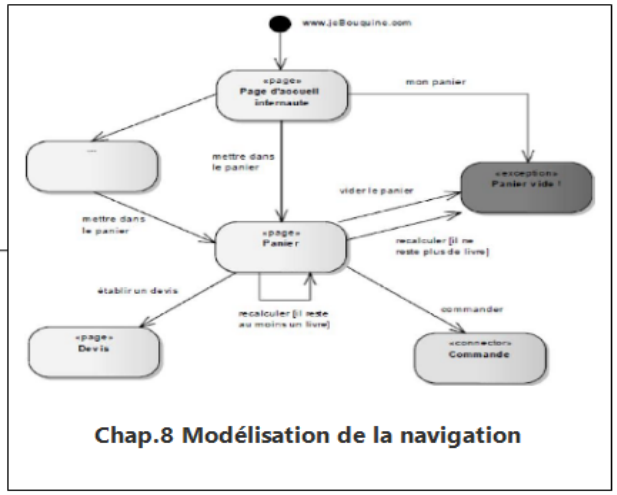
- Introduction
- Chap.1 Problématique
- Chap.2 Démarche

Identification des besoins

- Chap.3 Expression initiale des besoins
- Chap.4 Spécification des besoins - Cas d'utilisation et IHM
- Chap.5 Spécification détaillée des besoins - DSS

Analyse

- Chap.6 Analyse du domaine
- Chap.7 Diagrammes de classes participantes



Chap.8 Modélisation de la navigation

Conception

- Chap.9 Diagrammes d'interaction
- Chap.10 Diagrammes de classes de conception

- Chap.11 Implémentation
- Conclusion

8.1 Positionnement de l'étape

Rappelons le positionnement de cette activité de modélisation de la navigation par rapport à l'ensemble du processus que nous avons suivi : Nous avons identifié les cas d'utilisation et poursuivi leur description détaillée. Nous avons aussi présenté les différents IHM. Il nous reste qu'à détailler une exploitation supplémentaire de la maquettage ou une extension éventuelle de celle-ci. Il s'agit de réaliser des diagrammes dynamiques représentant de manière formelle l'ensemble des chemins possibles entre les principaux écrans proposés à l'utilisateur. Ces diagrammes s'appellent des diagrammes de navigation.

8.2 Diagramme d'états

Pour modéliser la navigation dans notre application web, nous allons utiliser un nombre restreint d'éléments standards, à savoir :

- des états pour représenter les classes dialogues,
- des transitions entre états déclenchées par des événements et pouvant porter des conditions, pour représenter les actions IHM.

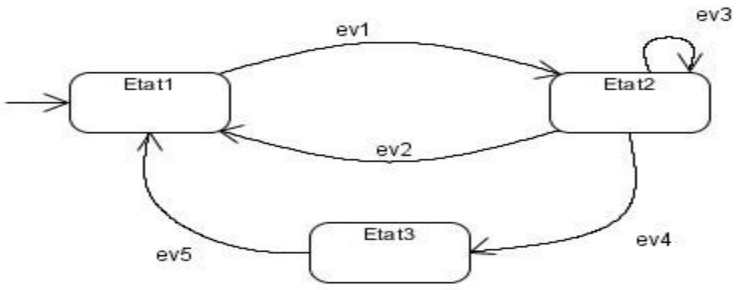


Figure 24 : Notation graphique de base du diagramme d'états

Dans les figures ci-dessous, nous nous présenterons les modèles de navigation uniquement au cas d'utilisation suivant : Authentification, « gérer les utilisateurs », « gestion des parcours d'apprentissage » et « participer à une session d'apprentissage ».

Diagramme de navigation internet d'authentification

Un utilisateur doit saisir son login et son mot de passe dans un frame particulier. Suivant le résultat du contrôle effectuer par le système, il se retrouve soit sur la page d'accueil propre a son profil, soit de nouveau sur le frame d'authentification avec un message d'erreur.

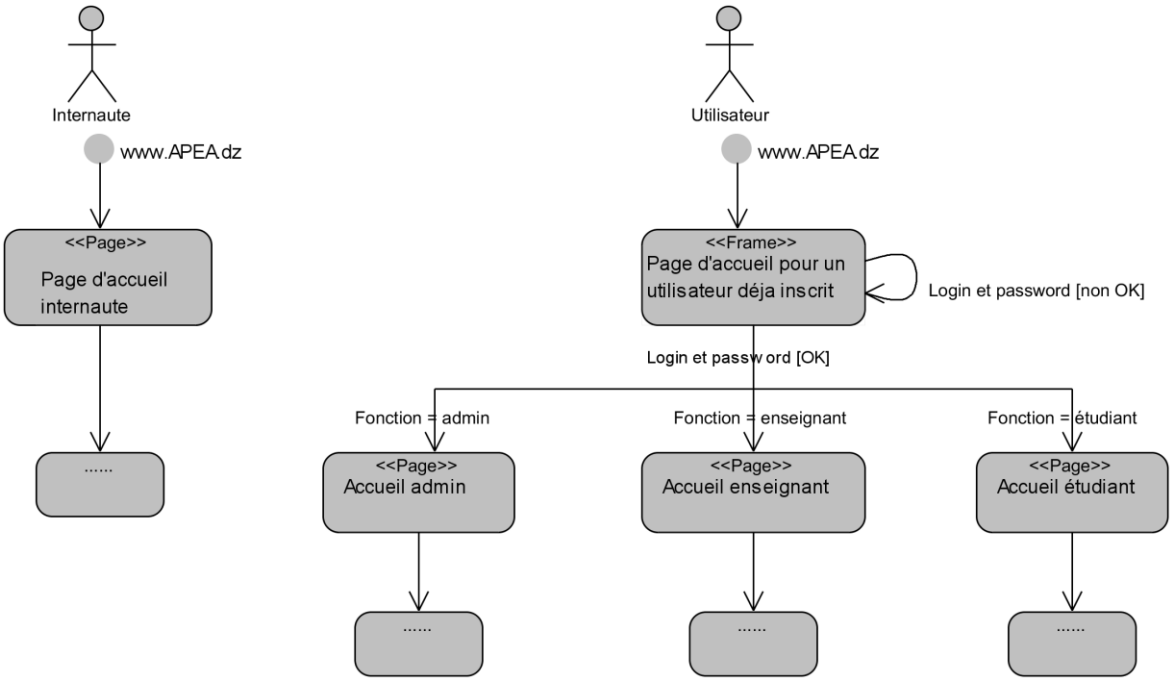


Figure 25 : Diagramme de navigation authentification

Diagramme de navigation internet: « Gérer les utilisateurs »

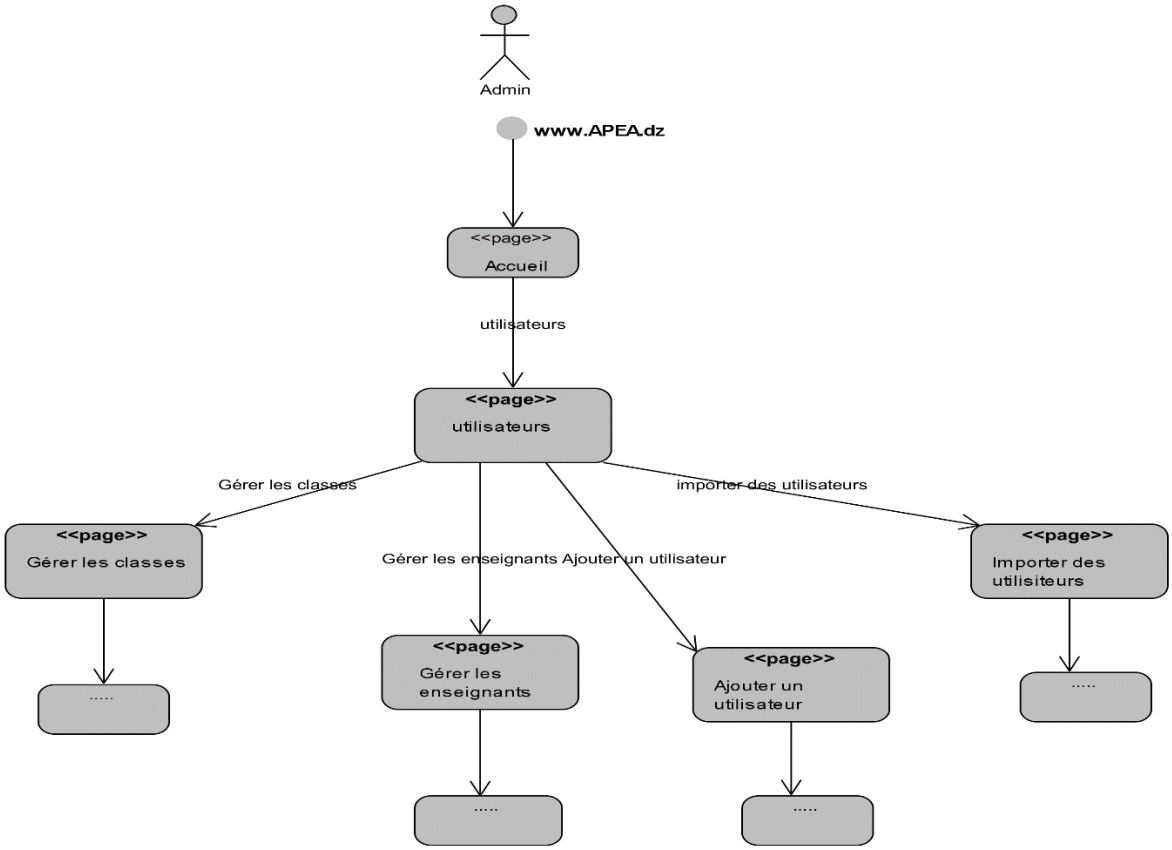


Figure 26 : Diagramme de navigation gestion des utilisateurs

Après l'authentification, l'administrateur se retrouve dans la page d'accueil qui lui convient. Il a la possibilité de gérer des classes, d'effectuer des mises à jour sur les utilisateurs et aussi de gérer des enseignants.

Diagramme de navigation internet: « Gestion des parcours d'apprentissage »

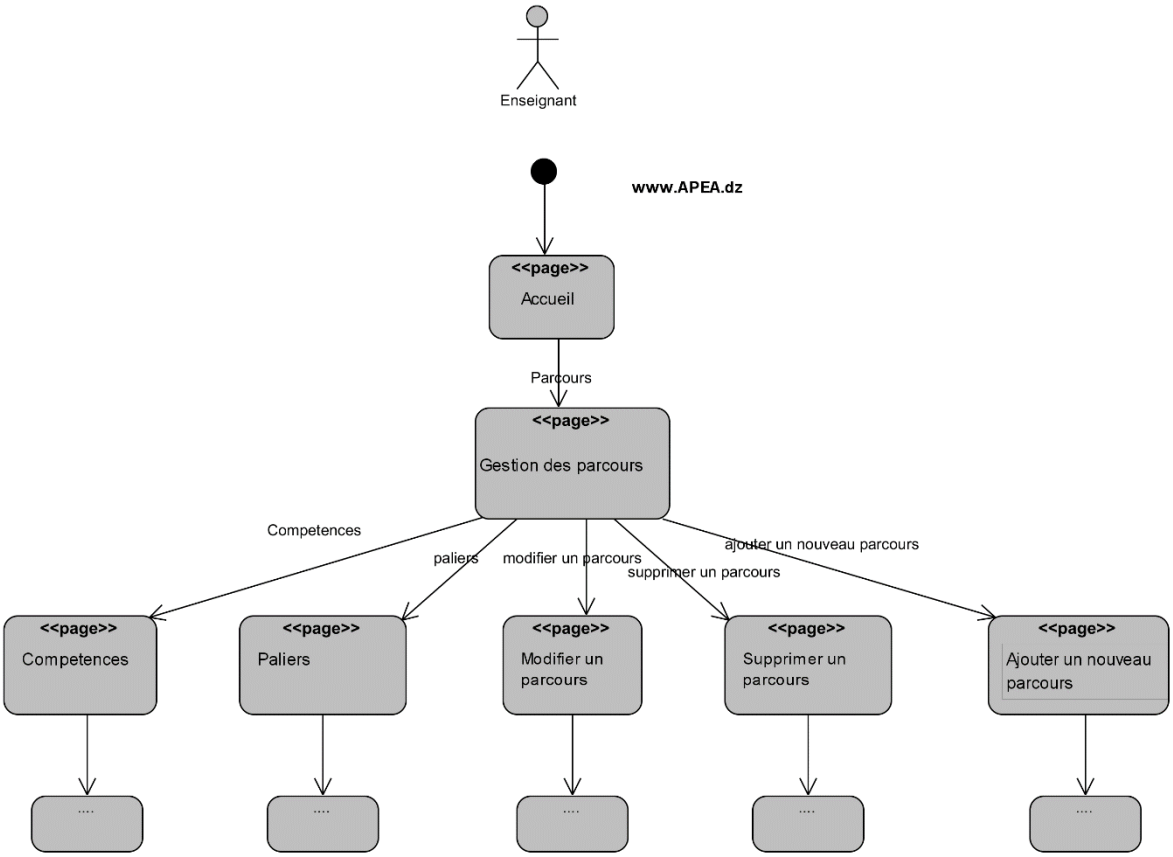


Figure 27 : Diagramme de navigation gérer les parcours d'apprentissage

Après l'authentification, l'enseignant se retrouve dans la page d'accueil qui lui convient. L'enseignant accèdera à la page de la gestion des parcours pour la mise à jours de ces parcours et aussi pour gérer les compétences générales ou les paliers.

Diagramme de navigation internet: participation à une session d'apprentissage

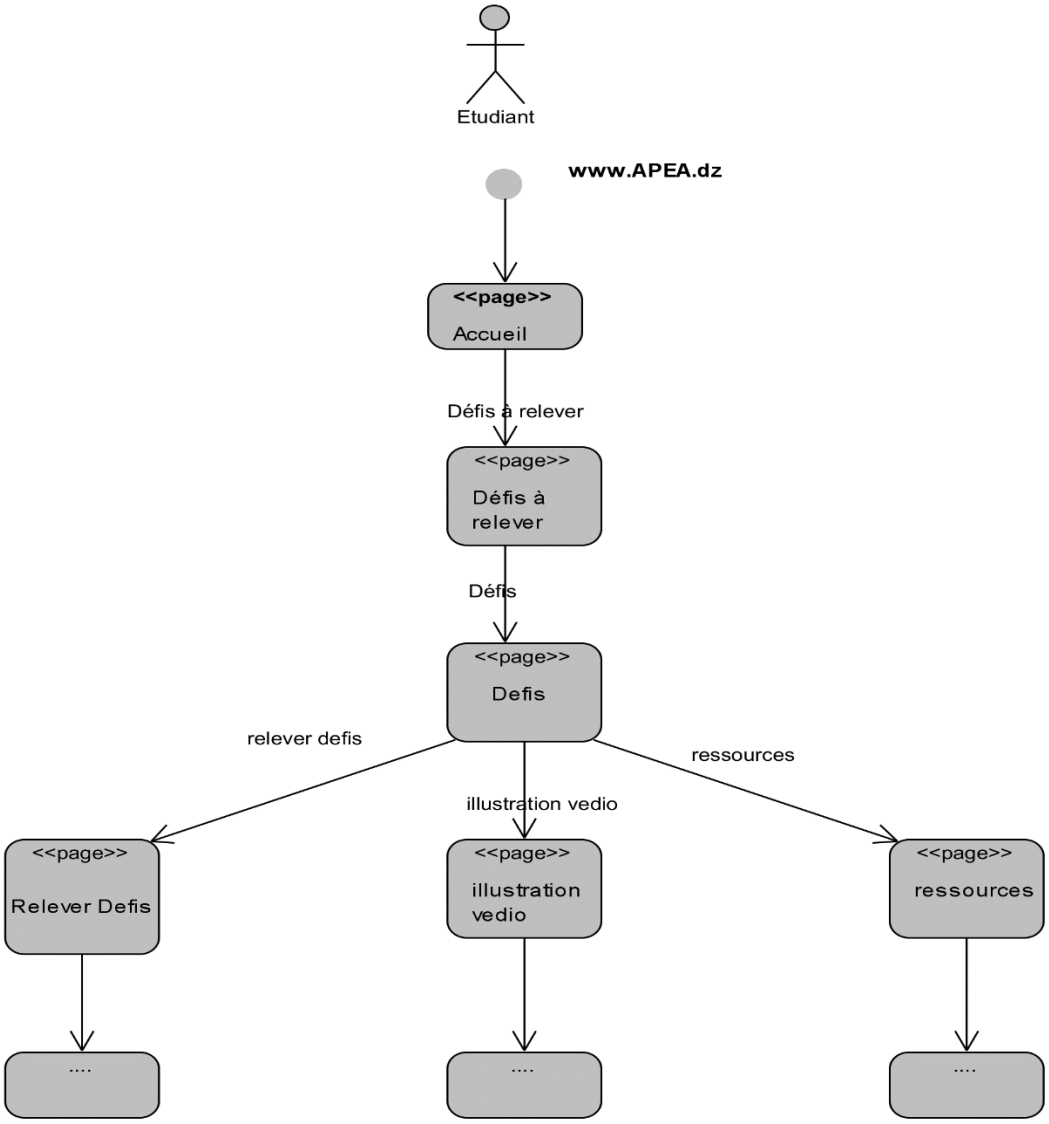


Figure 28 : Diagramme de navigation participer à une session d'apprentissage

Après l'authentification, l'étudiant se retrouve dans la page d'accueil qui lui convient. Si, par exemple, l'étudiant choisit la réalisation d'un défi, il doit passer sur la page défi à relever pour ensuite découvrir le code secret d'un défi, dans cette page l'étudiant a la possibilité de relevé le défi proposé par son enseignant ou consulté les ressources ou bien de demander un indice, comme il peut aussi voir une vidéo de démonstration.

Chap.9 – Diagrammes d'interaction

Auto-apprentissage de la programmation par une correction automatisée

- Introduction
- Chap.1 Problématique
- Chap.2 Démarche

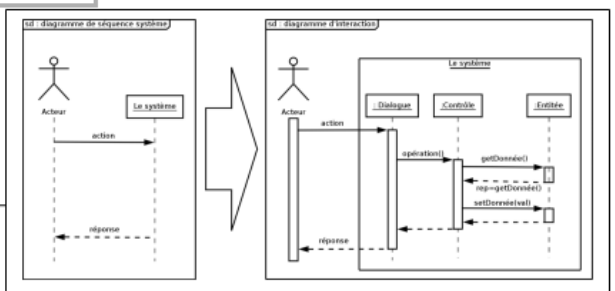
Identification des besoins

- Chap.3 Expression initiale des besoins
- Chap.4 Spécification des besoins - Cas d'utilisation et IHM
- Chap.5 Spécification détaillée des besoins - DSS

Analyse

- Chap.6 Analyse du domaine
- Chap.7 Diagrammes de classes participantes
- Chap.8 Modélisation de la navigation

Conception



Chap.9 Diagrammes d'interaction

- Chap.10 Diagrammes de classes de conception
- Chap.11 Implémentation
- Conclusion

9.1 Positionnement de l'étape

Pour passer maintenant vraiment à la conception, nous allons répartir tout le comportement du système entre ces classes d'analyse et nous décrirons les interactions correspondantes. Nous allons donc utiliser les diagrammes d'interactions (séquence ou communication) qui sont particulièrement utiles pour représenter graphiquement les décisions d'allocation de responsabilités. Chaque diagramme va ainsi représenter un ensemble d'objets de classes différentes collaborant dans le cadre d'un scénario d'exécution du système.

Dans ce genre de diagramme, les objets communiquent en s'envoyant des messages qui invoquent des opérations (ou méthodes) sur les objets récepteurs. Il est ainsi possible de suivre visuellement les interactions dynamiques entre objets, et les traitements réalisés par chacun. Par rapport aux diagrammes de séquence système, nous allons remplacer le système vu comme une boîte noire par un ensemble d'objets en interaction. Pour cela, nous utiliserons trois types de classes d'analyse, à savoir les dialogues, les contrôles et les entités. Nous respecterons également les règles que nous avions fixées sur les relations entre classes d'analyse, mais en nous intéressant cette fois-ci aux interactions dynamiques entre objets :

- Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues.
- Les dialogues peuvent interagir avec les contrôles.
- Les contrôles peuvent interagir avec les dialogues, les entités, ou d'autres contrôles.
- Les entités ne peuvent interagir qu'entre elles.

Le changement de niveau d'abstraction par rapport au diagramme de séquence système peut ainsi se représenter comme sur la figure suivante :

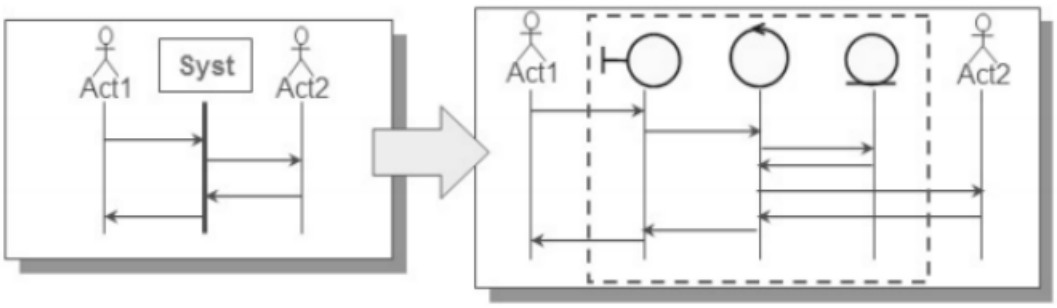
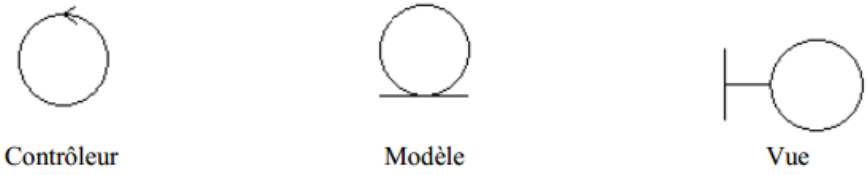


Figure 29 - Passage de l'analyse à la conception préliminaire

Dans les diagrammes qui vont suivre, nous utiliserons trois stéréotypes pour représenter les objets système correspondant aux dialogues, entités et contrôleurs :



Pour chaque cas d'utilisation, nous élaborerons un diagramme de séquence détaillé.

9.2 Analyse de l'interaction

Dans les figures ci-dessous, nous allons présenter les digrammes d'interaction (diagrammes de séquence détaillée ou DSD) uniquement pour les cas d'utilisation d'authentification et de la gestion des utilisateurs. Nous renvoyons le lecteur à l'annexe 4 pour les autres cas d'utilisation.

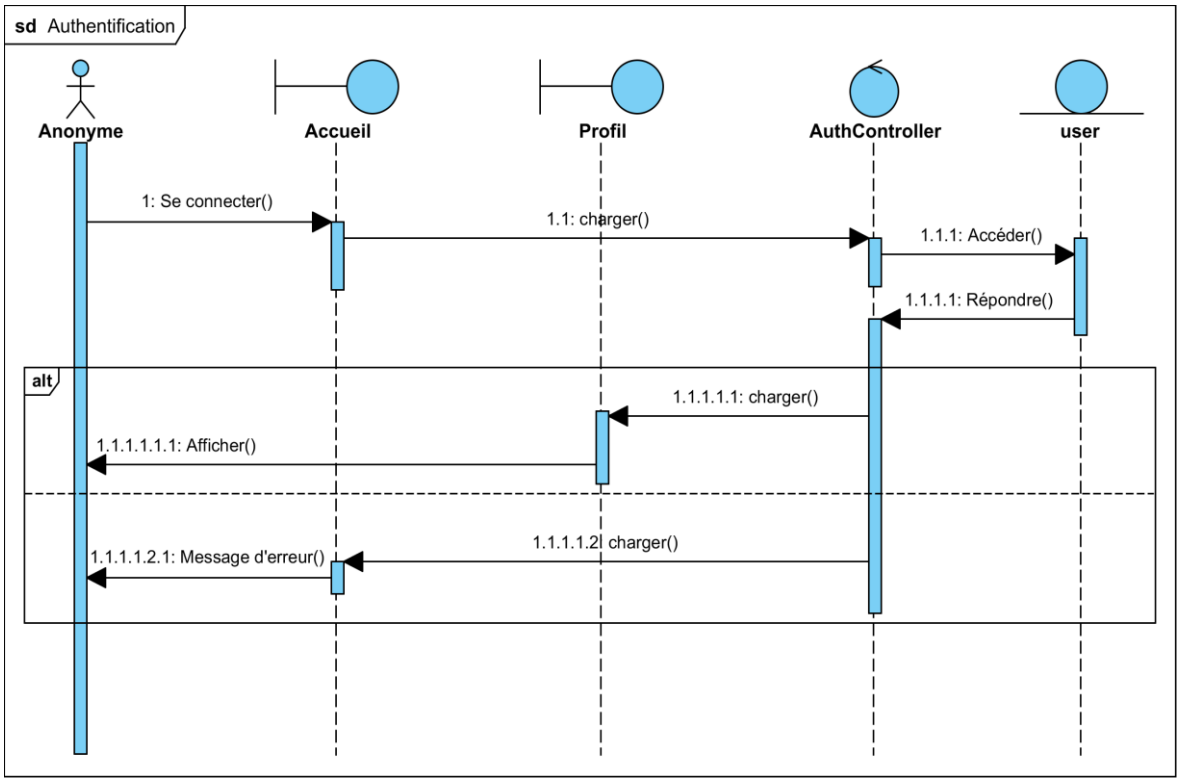


Figure 30 : DSD « authentification »

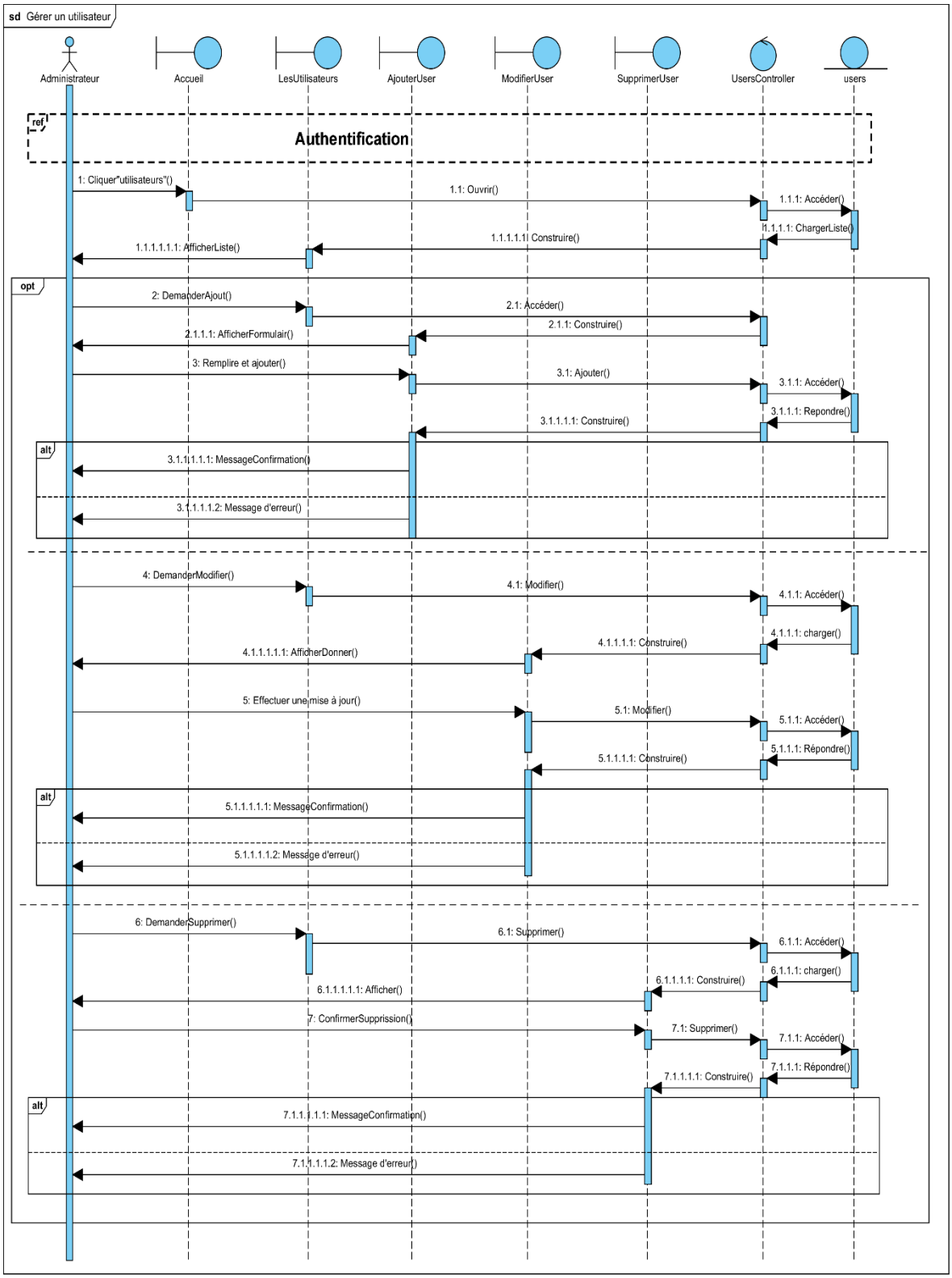


Figure 31 : DSD « Gestion des utilisateurs »

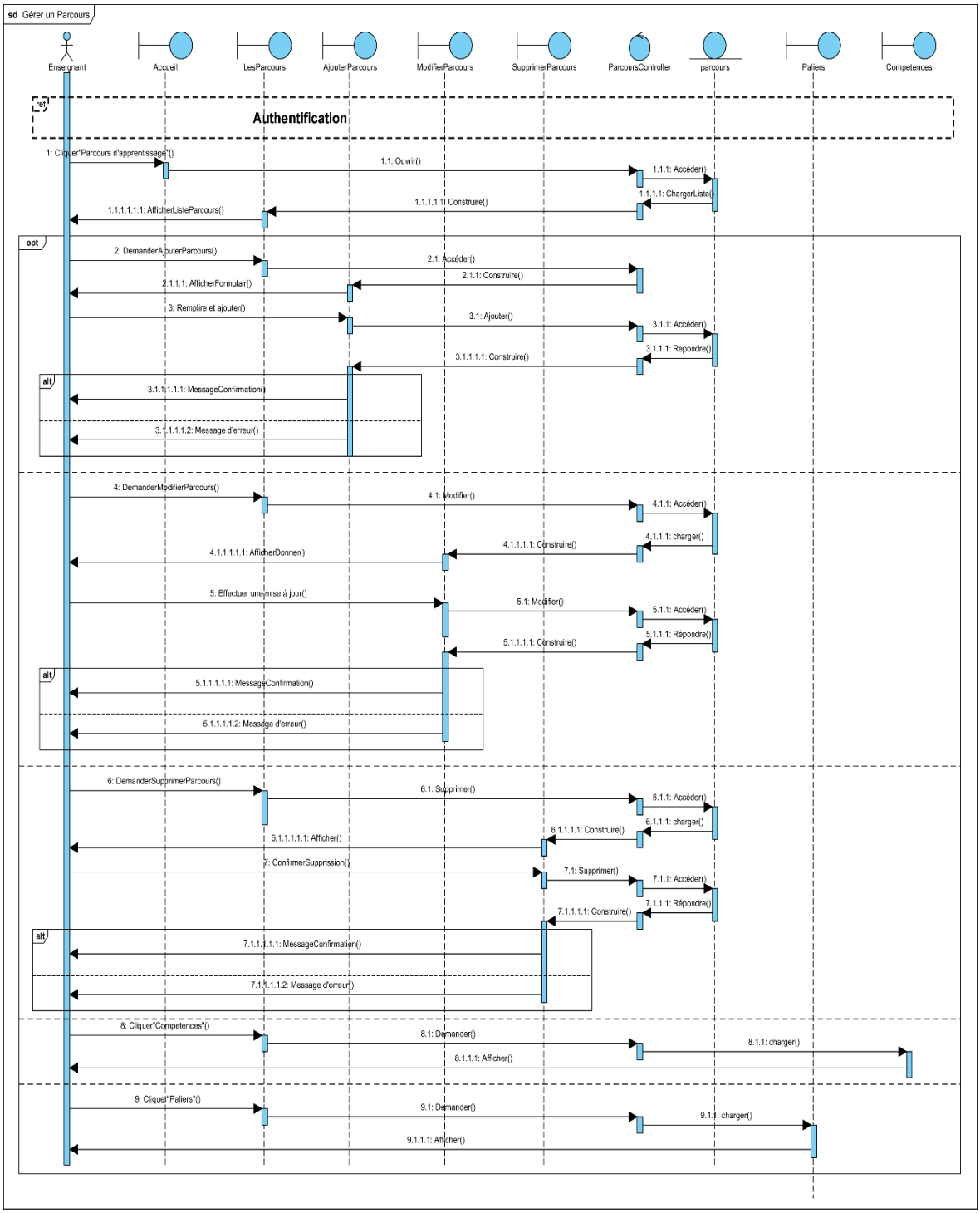


Figure 32 : DSD « gestion des parcours d'apprentissage »

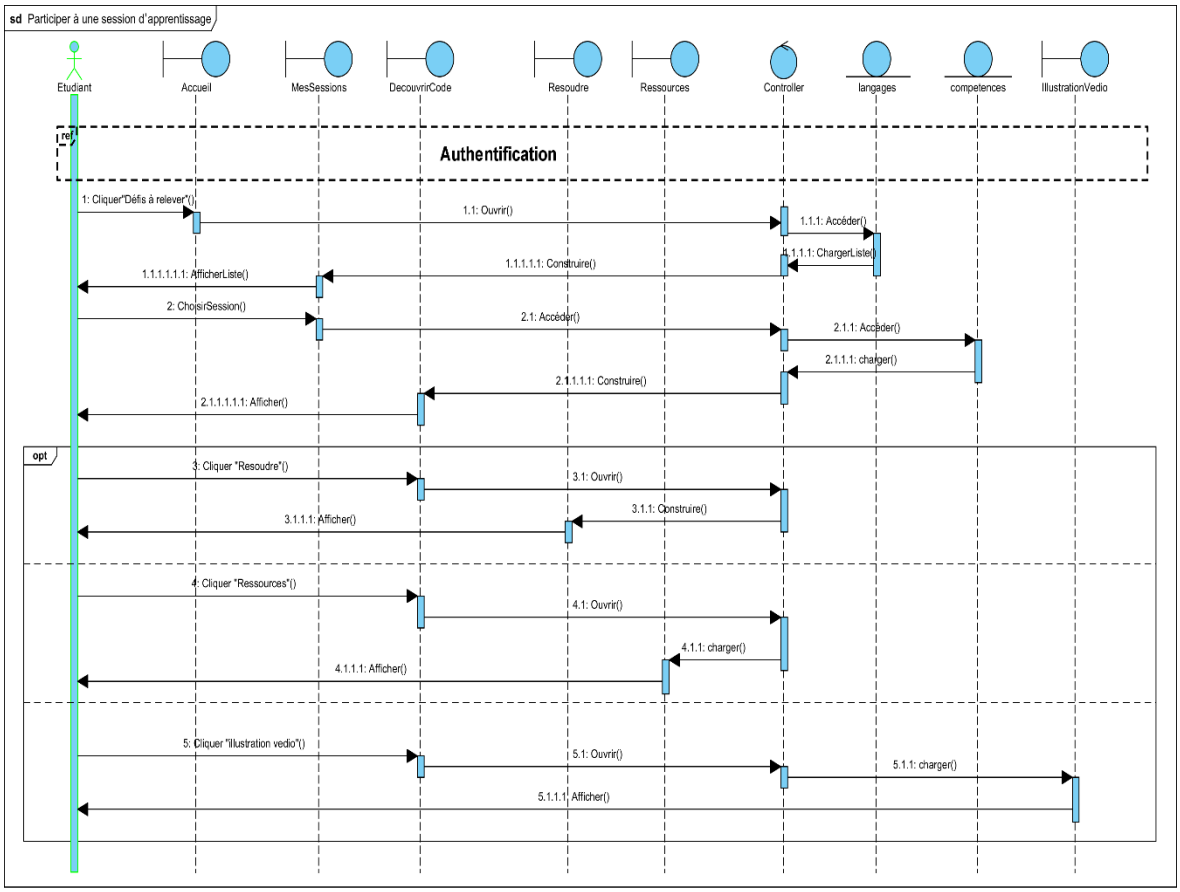
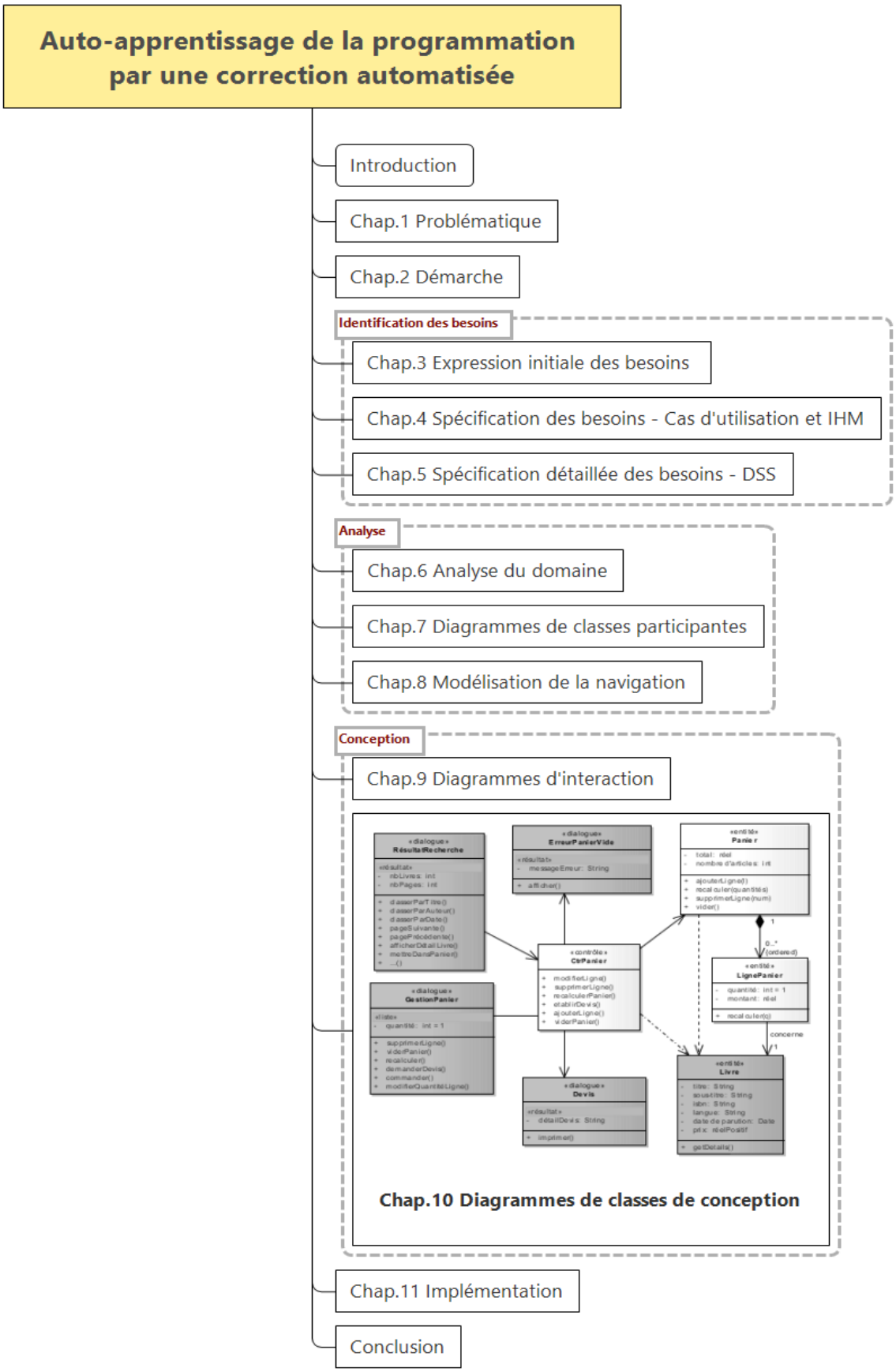


Figure 33 : DSD participé à une session d'apprentissage

Chap.10 – Diagrammes de classes de conception



10.1 Positionnement de l'étape

Rappelons le positionnement de cette activité par rapport à l'ensemble du processus que nous avons suivi. Chaque opération système va donner lieu a une méthode dynamique sous la forme d'un diagramme d'interaction. Les diagramme d'interaction ainsi réalisés vont permettre d'élaborer des diagramme de classes de conception, et ce en ajoutant principalement suivantes aux claasses issues du modèle d'analyse :

- Les opération : un message ne peut etre reçu par un objet que si sa classe a déclaré l'opération publiquecorrespondante ; [ROQUES, 2008]
- La navigabilité des associations, suivant que les liens entre objet sont durables ou temporaire, et en fonction du sens de circulation des messages. [ROQUES, 2008]

10.2 Conception détaillée

En appliquant les règles énoncées plus haut, nous réalisons les diagrammes de classe de conception (DCC). Dans ce qui suit, nous nous limiterons à présenter (à travers les figures ci-dessous) ces classes uniquement pour trois cas d'utilisation :

- Coté administrateur : gérer les utilisateurs
- Coté enseignant : gestion des parcours d'apprentissage
- Coté étudiant. : participation à une session d'apprentissage

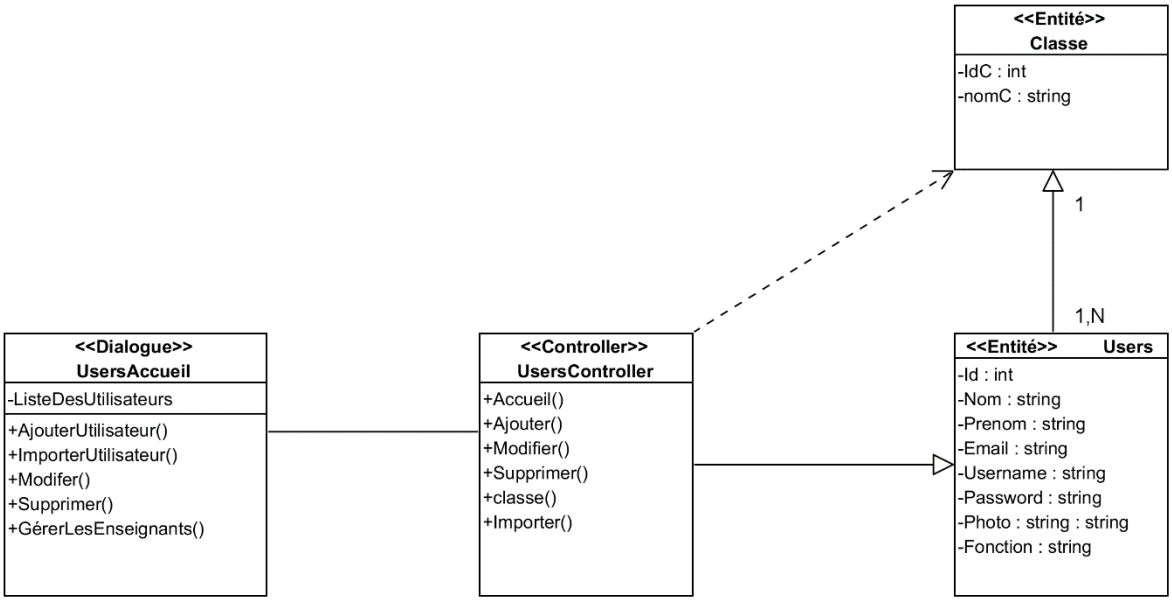


Figure 34 : DCC pour la gestion des utilisateurs

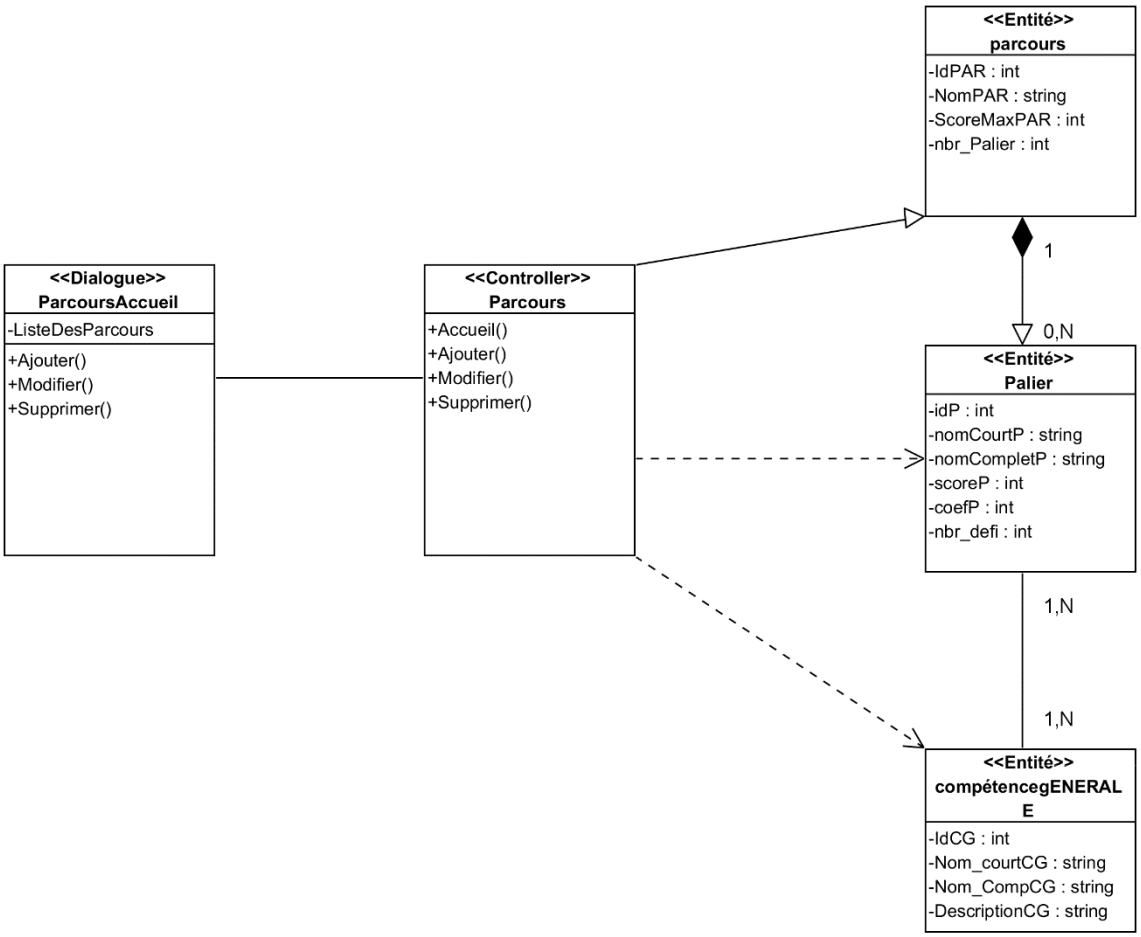


Figure 35 : DCC : gestion des parcours

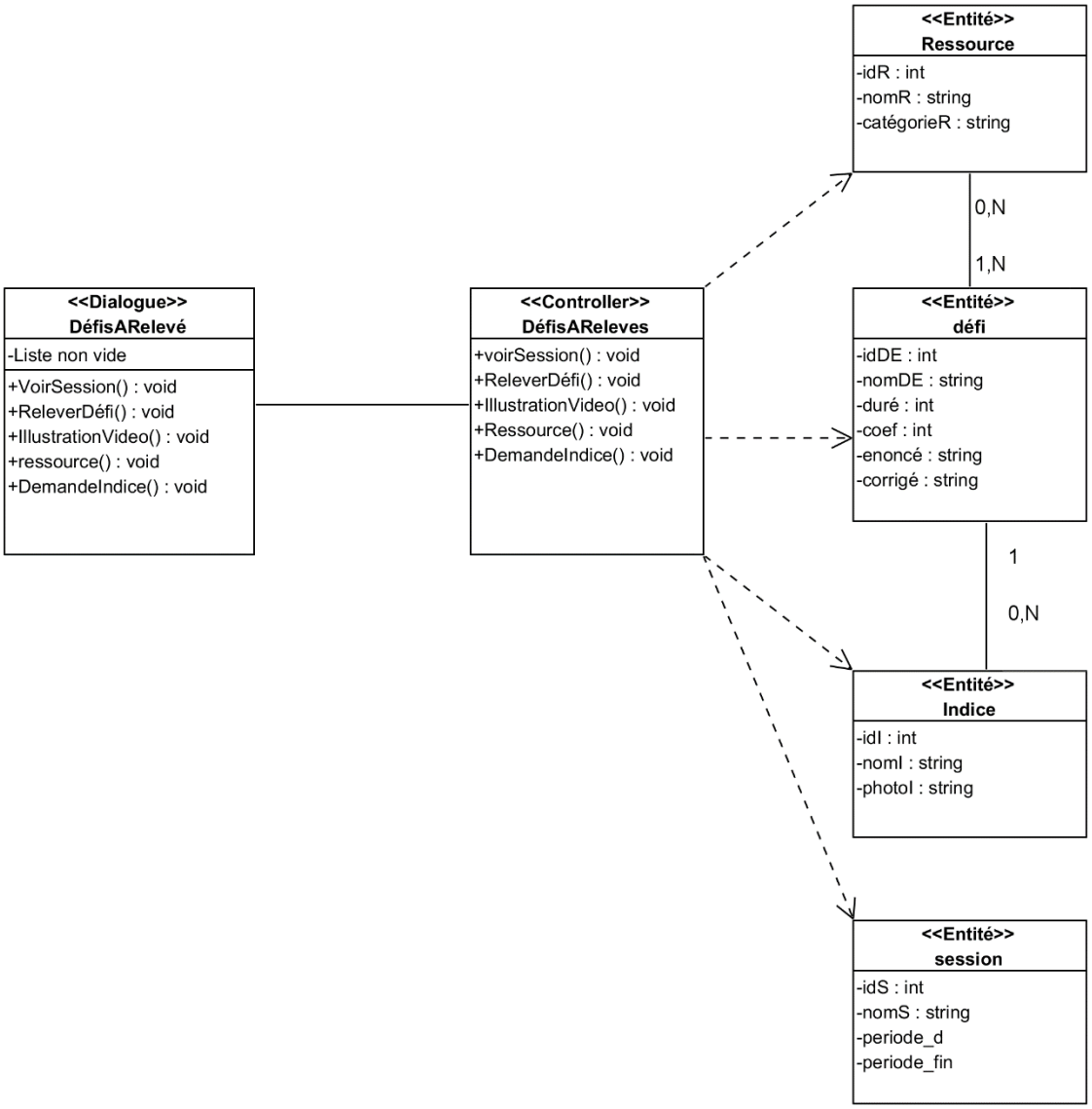
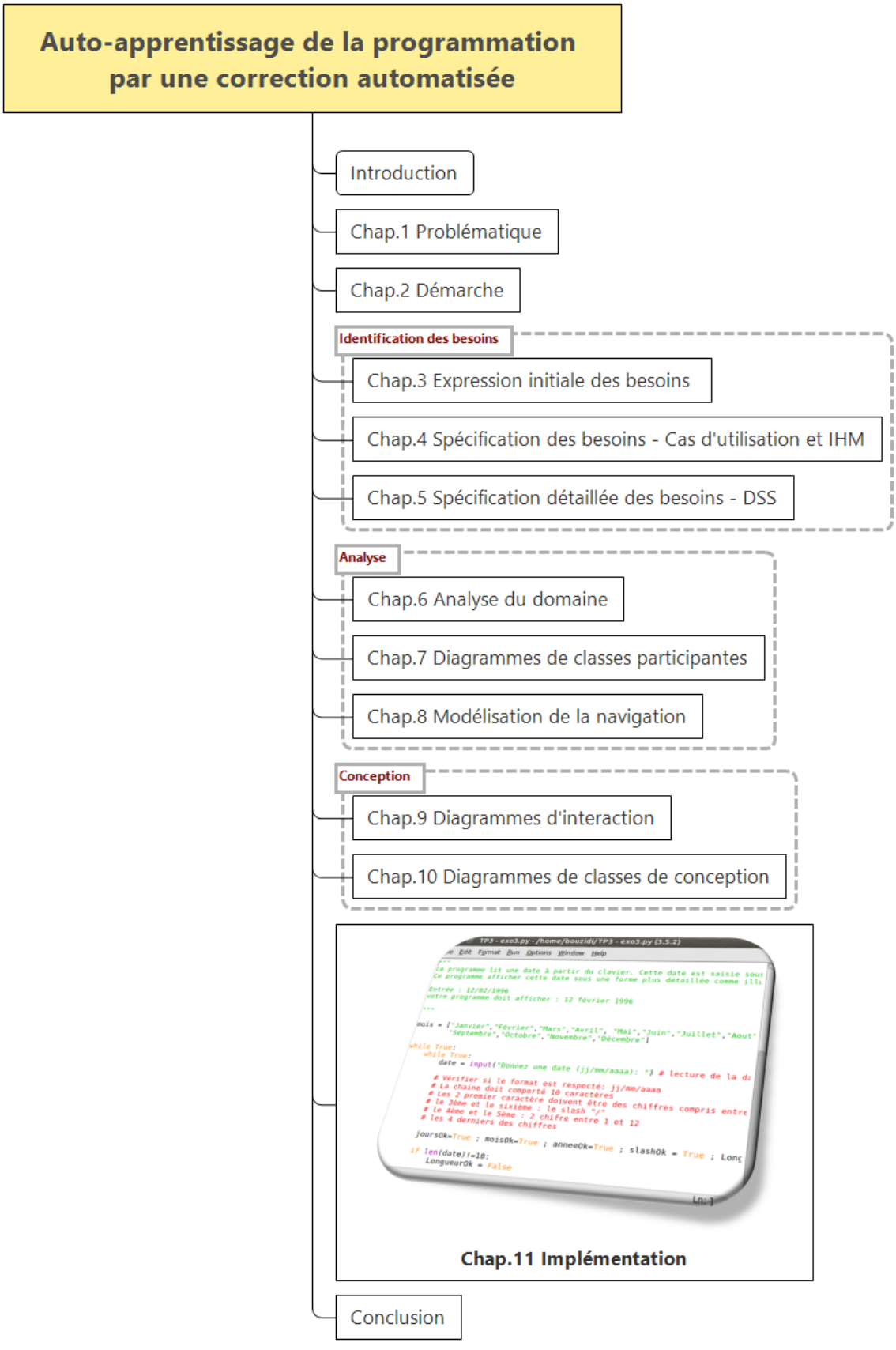


Figure 36 : DCC de cas d'utilisation participé à une session d'apprentissage

Chap.11 – Implémentation



11.1 Positionnement de l'étape

Rappelons le positionnement de cette activité par rapport à l'ensemble du processus que nous avons suivi. Les cas d'utilisation, le maquettage IHM, les diagrammes de séquences système et l'analyse de domaine nous a permet de déduire le diagramme de classe et ensuite le modèle relationnel, les classes d'analyse participantes ont permis d'élaborer les diagrammes de navigations et de séquences détaillées et de classe détaillé. La méthode que nous avons suivi, a permis, non seulement de produire le digramme des classes d'entités, mais a aussi permis d'élaborer des classe d'interface (IHM) et de contrôle. Ces différentes classes peuvent être directement transcrites en code dans un langage de script comme PHP. Par ailleurs, nous avons choisi de travailler avec un modèle de base de données relationnel, il va falloir procéder au passage du diagramme de classe d'entité vers une base de données relationnelle.

11.2 Environnement de développement

- **IDE : Netbeans** : 

NetBeans est un environnement de développement intégré (EDI), placé en *open source* par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de PHP : cet éditeur nous a permet le développement de l'application APEA.

- **Visual Paradigme :** 

Visual Paradigme est un éditeur qui propose une suite logicielle, constituée De plusieurs outils dont on a utilisé : il nous a permet la création des diagrammes UML.

- **Pencil :** 

Pencil est un logiciel de prototypage, il nous a permet de créer les différents IHM.

❖ **MySQLWorkbench :** 

Ce logiciel nous a permet de créer les tables de notre base de données APEA. Mais avant, nous avons d'abord procéder au passage du diagramme de classe vers un modèle relationnel.

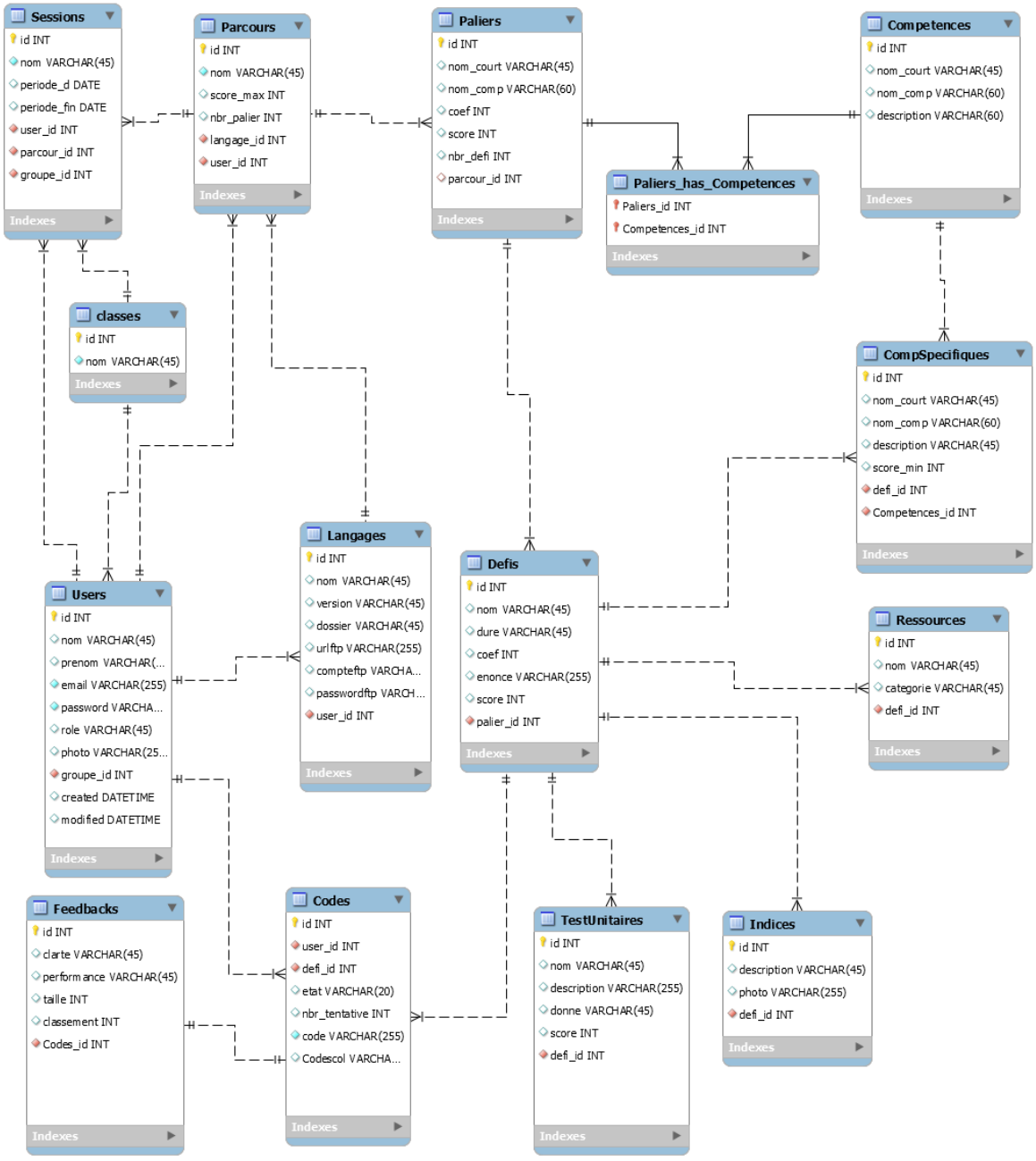


Figure 37 : Modèle de la base de données APEA

❖ **Le Framework cackPHP :**  **[A]**

CakePHP est conçu pour faciliter et simplifier les tâches classiques du développement web, il fournit une structure organisationnelle de base qui comprend les noms de classes, les noms de fichiers, les noms de table de base de données, et d'autres conventions.

- **Conventions essentiels de cackePHP :**

- ✓ Les dialogues seront nommés exactement pareil que les actions (méthodes) des contrôles. **Exemple** : La fonction add() de la classe *ParcoursController* cherchera un gabarit de add dans src/Template/Parcoours/Add.ctp.
- ✓ Les contrôles commenceront par une majuscule et porteront (en général) le nom de l'entité avec laquelle ils sont associés et seront suivis du nom « Controller ». **Exemple** : *ParcoursController*.
- ✓ Les entités seront (de préférence) en anglais, devront être au pluriel, en CamelCaseet finissent par Table. **Exemple** : *ParcoursTable*.
- ✓ **Adapter au modèle MVC :**

-Pour le développement du code de notre application, nous avons préféré de travailler avec ce Framework. Il bâtit sur la base de l'architecture MVC (Modèle Vue Contrôleur). Il s'agit d'une architecture et d'une méthode de conception pour le développement d'applications logicielles qui sépare le modèle de données, l'interface utilisateur et la logique de contrôle.

-Ce modèle d'architecture impose la séparation entre les données, les traitements et la présentation, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.

- o Le Modèle représente le comportement de l'application : traitements des données, interactions avec la base de données, etc. Il décrit les données manipulées par l'application et définit les méthodes d'accès.

- o La Vue correspond à l'interface avec laquelle l'utilisateur interagit. Les résultats renvoyés par le modèle sont dénués de toute présentation mais sont présentés par les vues. Plusieurs vues peuvent afficher les informations d'un même modèle. Elle peut être conçue en html, ou tout autre " langage " de présentation. La vue

n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle, et de permettre à l'utilisateur d'interagir avec elles.

- Le Contrôleur prend en charge la gestion des évènements de synchronisation pour mettre à jour la vue ou le modèle. Il n'effectue aucun traitement, ne modifie aucune donnée, il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondante à la demande.

11.3 Solution technologique

L'application que nous avons conçue est un site web dynamique. Donc, elle est accessible depuis un navigateur Web et son code est hébergé sur un serveur. Ce dernier est doté de trois applications serveur : un serveur web (nous avons opté pour Apache), un serveur de base de données (nous avons opté pour MySQL) et un interpréteur de script coté serveur (nous avons opté pour PHP). Notre application partage un dossier (en local ou en FTP) avec l'environnement sécurisé d'exécution de code « ESEC ».

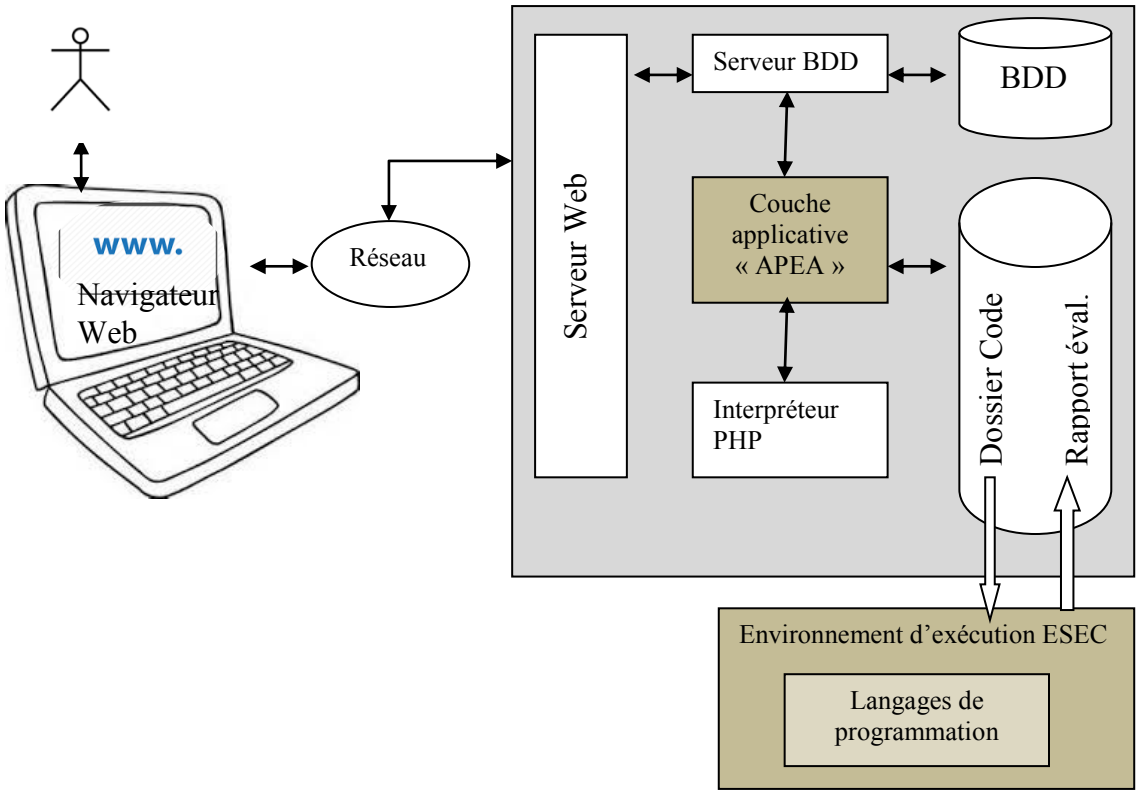


Figure 38 : Architecture logicielle

11.4 Structure de notre application

Notre application est basée sur le framework cakePHP, voici les fichiers et répertoires qui composent ce dernier :

- bin : contient les exécutables de la console Cake.
- config : permettant diverses configurations du Framework.
- plugins : ce dossier contient les plugins cakePHP.
- src : c'est là que nous placerons les fichiers de notre application.
- tests : l'endroit où nous mettons les cas de test pour notre application
- tmp : l'endroit où CakePHP stocke les données temporaires.
- vendor: contient les librairies externe s au Framework s'il y'en a.
- webroot : Il contient tous les fichiers que nous souhaitons voir publiquement.
- Le reste ce sont des fichiers de licence, l'interface principale de cakePHP (index.php), et le dossier qui contient les routages de l'application (.htaccess) c'est ce fichier la qui va définir les chemin et les URLs de cakePHP.

Voici la structure des répertoires de cakePHP :

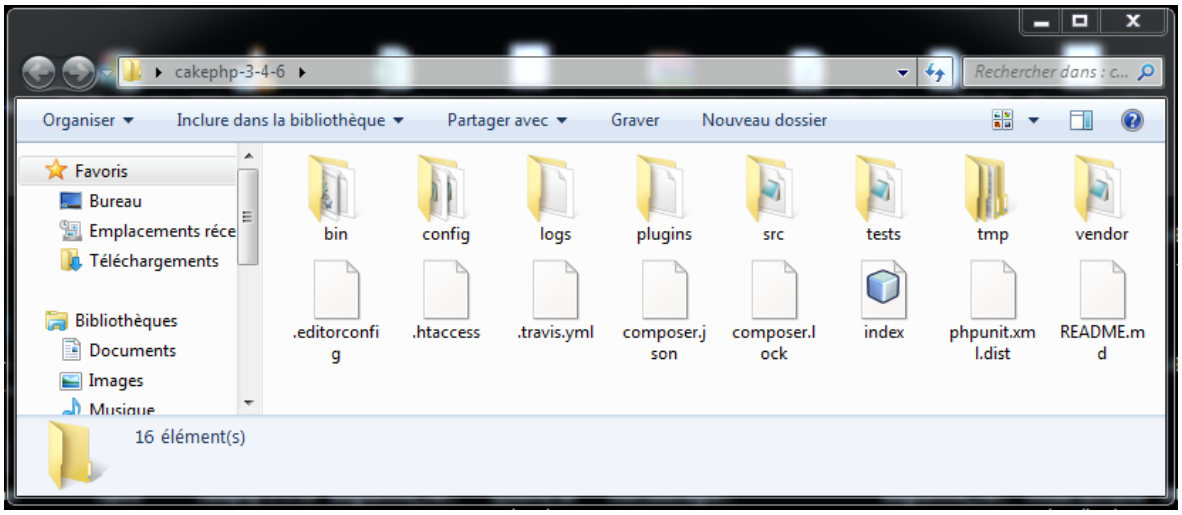


Figure 39 : La structure du répertoire de cakePHP qui contiendra l'application APEA

La structure du répertoire « src » contient 6 sous-répertoires permettant d'organiser chaque partie de notre application :

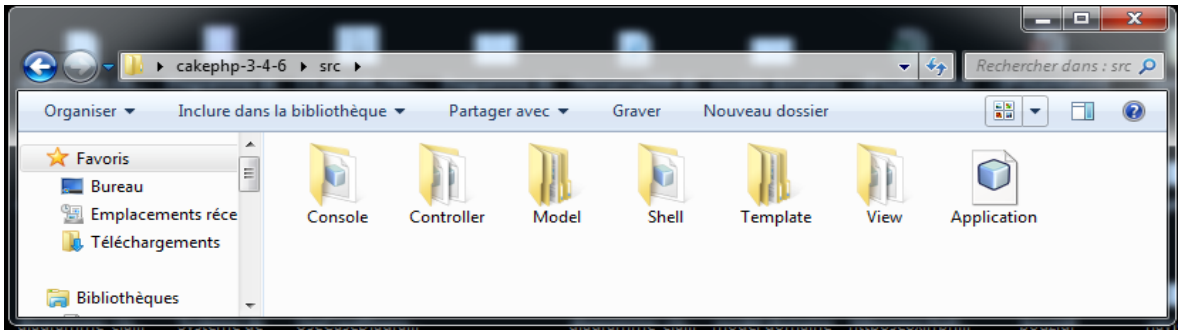


Figure 40 : La structure de répertoire « src »

C'est un dossier essentiel de cakePHP où on peut organiser et stocké le code de notre application.

11.5 Codage

Nous présentons, dans ce qui suit quelques exemples de codes que nous avons produits. Nous nous contenterons de présenter un script « contrôleur », un script « vue » et un script « modèle », ce qui correspond au modèle MVC que nous avons adopté. On se limitera uniquement au cas d'utilisation « ajouter un enseignant » :

```php
<?php
namespace App\Controller;
use Cake\Controller\Controller;
class AppController extends Controller

{
public function Ajouter_enseignant()
{
$this->viewBuilder()->layout('Admin_home');        // Choisir layout (interface)
    $user = $this->Users->newEntity();               // création d'une nouvelle entité
        if ($this->request->is('post'))                  // si on récupère les données
{
 $user = $this->Users->patchEntity($user, $this->request->data); // création de l'objet
  if ($this->Users->save($user))                // si l'entité est bien enregistrer (sans erreur)
 {
  $this->Flash->success(__('L'+"'"+'utilisateur est bien ajouter.'));// message de confirmation
 return $this->redirect(['action' => 'enseignant']); // redirection l'admin vers la liste des enseignants
 }
 $this->Flash->error(__('L'+"'"+'utilisateur na pas été ajouter. SVP, veuillez réssayer un nouveau!'));
}
    $this->set(compact('user'));                        // apres la souvgarde on renvoi les données vers la vue
    $this->paginate = ['conditions' => ['Users.role' =>'Enseignant']]; // fixé la fonction de user a enseignant
    $this->set('users', $this->paginate($this->Users)); // affichage des données dans la vue
    $this->set('_serialize', ['users']);                // serialisation des données de l'utilisateur
}
}
```

Figure 41 : Exemple d'un Contrôler

```php
<?php
namespace App\Model\Table;
use Cake\ORM\Query;
use Cake\ORM\RulesChecker;
use Cake\ORM\Table;
use Cake\Validation\Validator;
class UsersTable extends Table
{
    public function validationDefault(Validator $validator)
    {
        $validator
            ->integer('id')
            ->allowEmpty('id', 'create');
        $validator
            ->allowEmpty('nom');
        $validator
            ->allowEmpty('prenom');
        $validator
            ->email('email')
            ->requirePresence('email', 'create')
            ->notEmpty('email');
        $validator
            ->requirePresence('password', 'create')
            ->notEmpty('password');
        $validator
            ->allowEmpty('role');
        $validator
            ->allowEmpty('photo');
        return $validator;
    }
}
```

Figure 42 : Exemple d'un modèle

```php
<div class="users form large-9 medium-8 columns content">
    <?= $this->Form->create($user) ?>
    <fieldset>
        <legend><strong><?= __('Ajouter un enseignant') ?></strong></legend>
                    <div class="input-group">
                        <span class="input-group-addon">Nom <?php  echo '       ' ?> </span>
                        <?= $this->Form->input('nom',['label'=>'','class'=>'form-control','placeholder'=>'Nom']) ?>
                    </div>
                    <h4 class="blockHeader"></h4>
                    <div class="input-group">
                        <span class="input-group-addon">Prénom <?php  echo '  ' ?></span>
                        <?= $this->Form->input('prenom',['label'=>'','class'=>'form-control','placeholder'=>'Prénom']) ?>
                    </div>
                    <h4 class="blockHeader"></h4>
                    <div class="input-group">
                        <span class="input-group-addon">Email<?php  echo '       ' ?></span>
                        <?= $this->Form->input('email',['label'=>'','class'=>'form-control','placeholder'=>'Email']) ?>
                    </div>
                    <h4 class="blockHeader"></h4>
                    <div class="input-group">
                        <span class="input-group-addon">Password</span>
                        <?= $this->Form->input('password',['label'=>'','class'=>'form-control','placeholder'=>'Password']) ?>
                    </div>
                    <h4 class="blockHeader"></h4>
                    <div class="input-group">
                        <span class="input-group-addon">Role <?php  echo '       ' ?></span><strong>
                        <?= $this->Form->input('role',['label'=>'','class'=>'form-control','placeholder'=>'Role','options'=>['Enseignant'=>'
                            Enseignant']]) ?></strong>
                    </div>
                    <h4 class="blockHeader"></h4>
                    <div class="input-group">
                        <span class="input-group-addon">Photo<?php  echo '      ' ?> </span>
                        <?= $this->Form->input('photo',['label'=>'','class'=>'form-control','placeholder'=>'Photo']) ?>
                    </div>
                    <h4 class="blockHeader"></h4>
    </fieldset>
      <h5 class="blockHeader">
       <input type="submit" class="btn btn-info" value="Ajouter" >
      </h5>
      <?= $this->Form->end() ?>
</div>
```

Figure 43 : Exemple d'une vue

11.6 IHM

Apres avoir présenté la solution technologique et l'environnement de développement, nous présentons les deux interfaces :

Figure 44 : Interface d'accueil avec l'authentification

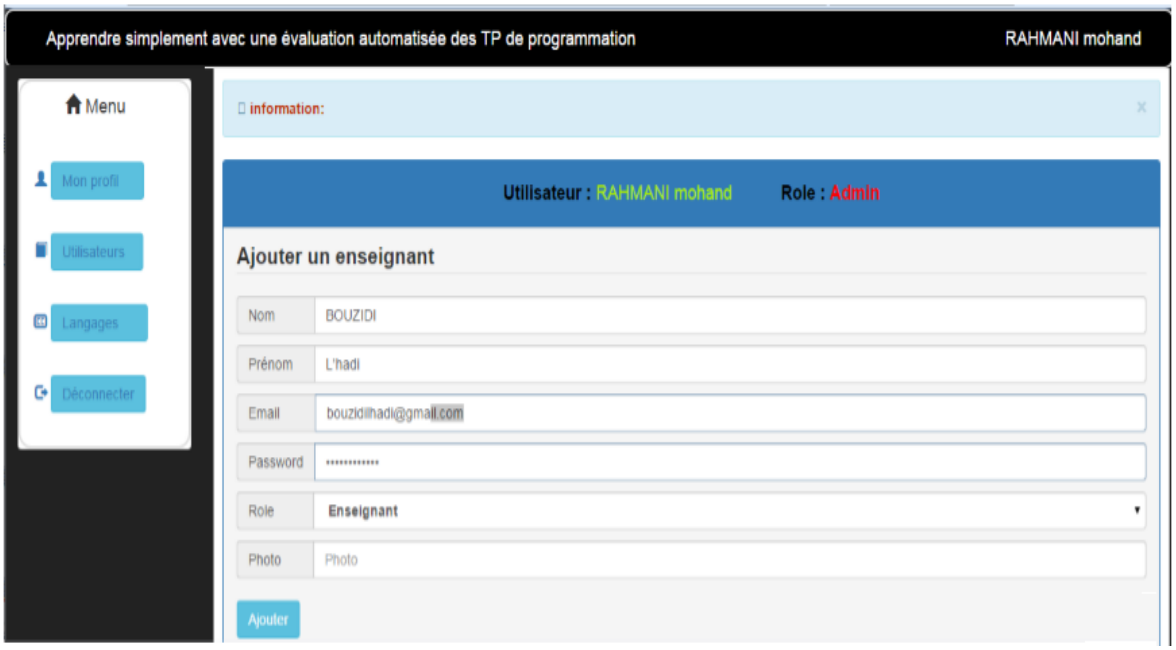


Figure 45 : Interface d'ajouter un enseignant

Conclusion

L'application de l'évaluation automatisée des devoirs de programmation est récente. Elle est appliquée dans divers contextes de formation : MOOC, Université, Lycées et même dans des contextes informels comme l'apprentissage libre sur internet (certains sites Web d'apprentissage de la programmation). Pour une Université comme la nôtre, le bénéfice est certain : réduire la charge de travail des chargés de TP tout en augmentant l'efficacité de l'apprentissage de la programmation grâce à des rapports d'évaluation « instantanés ». A travers, ce projet de fin d'étude de Master, nous avons proposé une solution technique permettant d'implémenter cette méthode d'évaluation. Nous avons essayé de nous démarquer des outils existant en prenant en compte un peu plus les aspects pédagogiques, notamment la possibilité de spécifier des parcours d'apprentissage et d'indiquer des compétences générales et spécifiques. Ainsi, les rapports d'évaluation automatisée qu'il est possible de générer avec notre solution permettent de situer le niveau d'apprentissage des étudiants pas uniquement par un score, mais en citant l'ensemble des compétences qu'il a acquis.

D'un point de vue méthodologie, nous avons appliqué la démarche proposée par Pascal Roques. En particulier, nous nous sommes efforcés de passer en revue tous les aspects analytiques et conceptuels permettant de passer des besoins à l'écriture du code.

Au niveau de l'implémentation, nous avons opté pour le langage PHP et nous nous sommes basés sur un modèle d'architecture MVC en utilisant le framework cakePHP.

Dans notre projet, en réalité il y'avait deux systèmes à réaliser : L'application « APEA » et l'environnement « ESEC ». APEA est une application Web de gestion des TP de programmation (élaboration, soumission, configuration etc.…) et ESEC est un environnement sécurisé d'exécution des codes soumis par les étudiants. Malheureusement, nous n'avions pas eu le temps suffisant pour analyse et concevoir le second système. Bien évidemment, en perspectives futures, il faut approfondir la réflexion sur ce second système tout en améliorant l'ergonomie, et les fonctionnalités du premier système.

Bibliographie

[ROQUES, 2006] : PASCAL ROQUES, UML 2 : MODELISER UNE APPLICATION WEB, 2EME EDITION EYROLLES, 2006, 236 PAGES.

[ROQUES, 2007] : P. ROQUES, F. VALLEE : UML 2 EN ACTION : DE L'ANALYSE DES BESOINS A LA CONCEPTION, 2007, EYROLLES.

[CHARROUX, 2008] : B. CHARROUX, A. OSMANI, Y. THIERRY-MIEG, UML 2, PRATIQUE DE LA MODELISATION, 2E EDITION, PEARSON EDUCATION, 2008

[GABAY, 2008] : J. GABAY, D. GABAY, UML 2: ANALYSE ET CONCEPTION, DUNOD, 2008

[IHANTOLA, 2010] P. IHANTOLA , T. AHONIEMI , V. KARAVIRTA , O. SEPPÄLÄ, REVIEW OF RECENT SYSTEMS FOR AUTOMATIC ASSESSMENT OF PROGRAMMING ASSIGNMENTS, PROCEEDINGS OF THE 10TH KOLI CALLING INTERNATIONAL CONFERENCE ON COMPUTING EDUCATION RESEARCH, P.86-93, OCTOBER 28-31, 2010, KOLI, FINLAND

[CSF, 2017]: CAKE SOFTWARE FOUNDATION, FÉVR. 20, 2017 CAKEPHP COOKBOOK DOCUMENTATION VERSION 3.4

[ROQUES, 2007] : PASCAL ROQUES, FRANCK VALLEE, UML2 EN ACTION, 4E EDITION, EYROLLES, 2007.

[ROQUES, 2008] : PASCAL ROQUES, UML2 PAR LA PRATIQUE, ETUDES DE CAS ET EXERCICES CORRIGES, 6E EDITION.

Annexes

Table des annexes

Annexe 1 Maquettes d'IHM
Annexe 2 Diagrammes de séquence système
Annexe 3 Diagrammes de classe participantes
Annexe 4 Diagrammes d'interaction

Annexe 1
Maquettes d'IHM

✓ Coté Administrateur

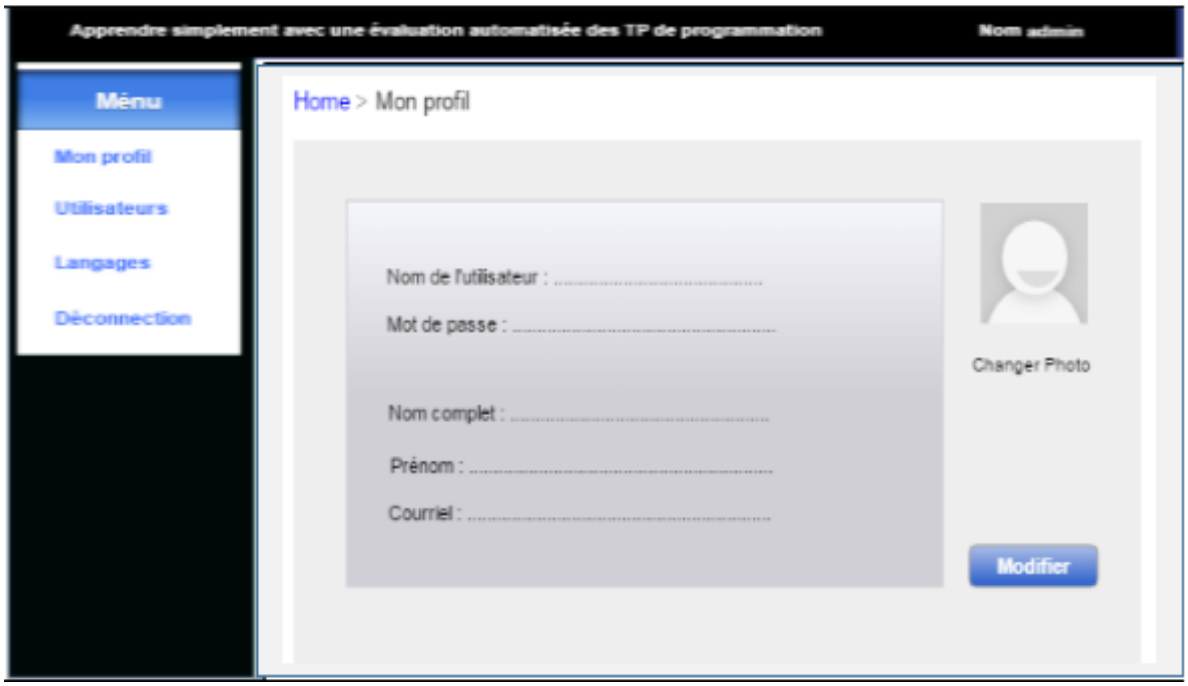


Figure 1 : IHM : Mon Profil

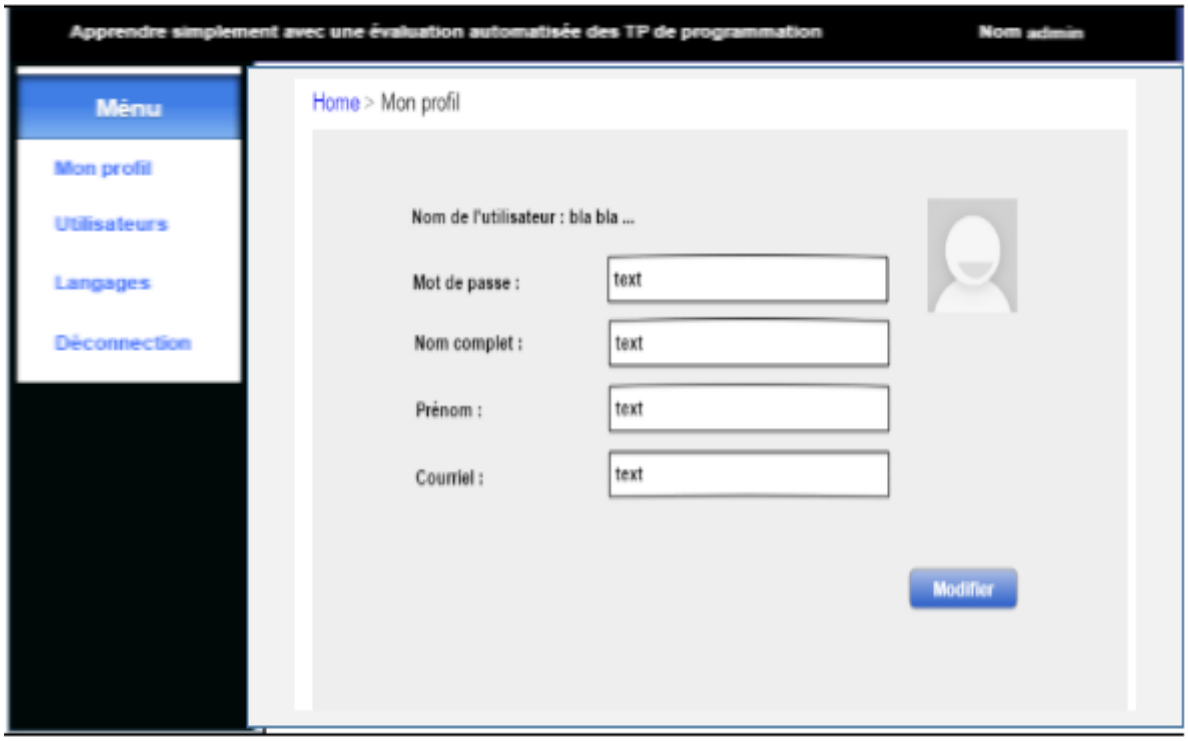


Figure 2 : IHM « Modifier le Mon Profil »

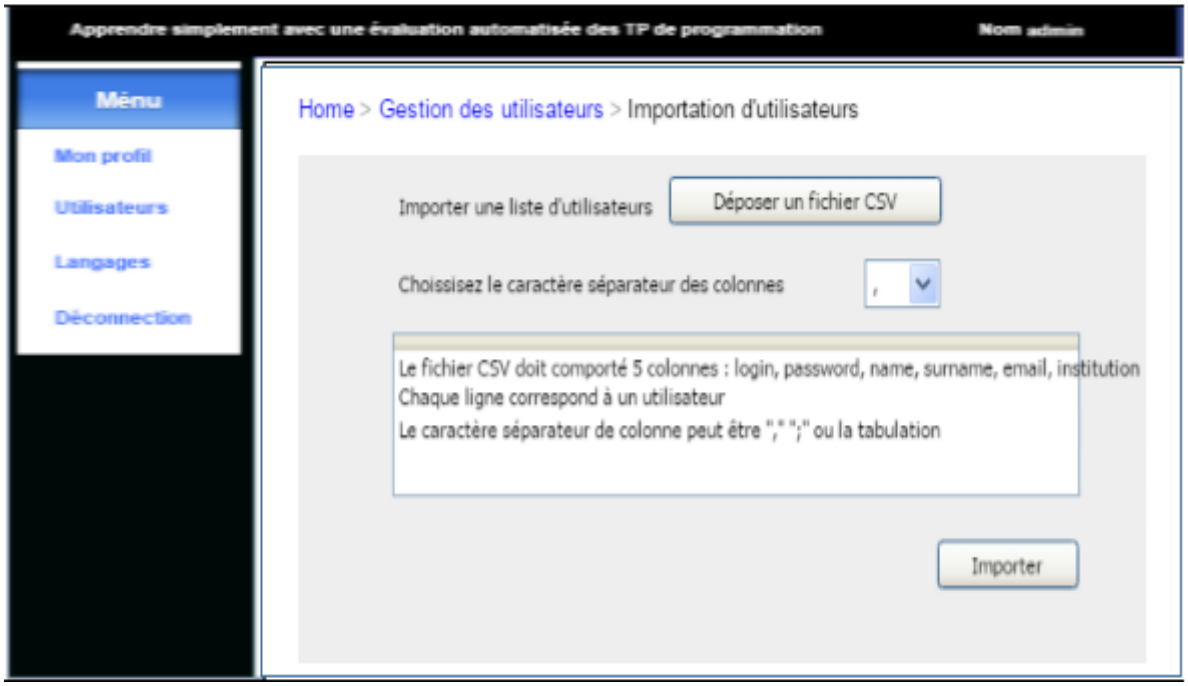


Figure 3 : IHM « Importation des utilisateurs »

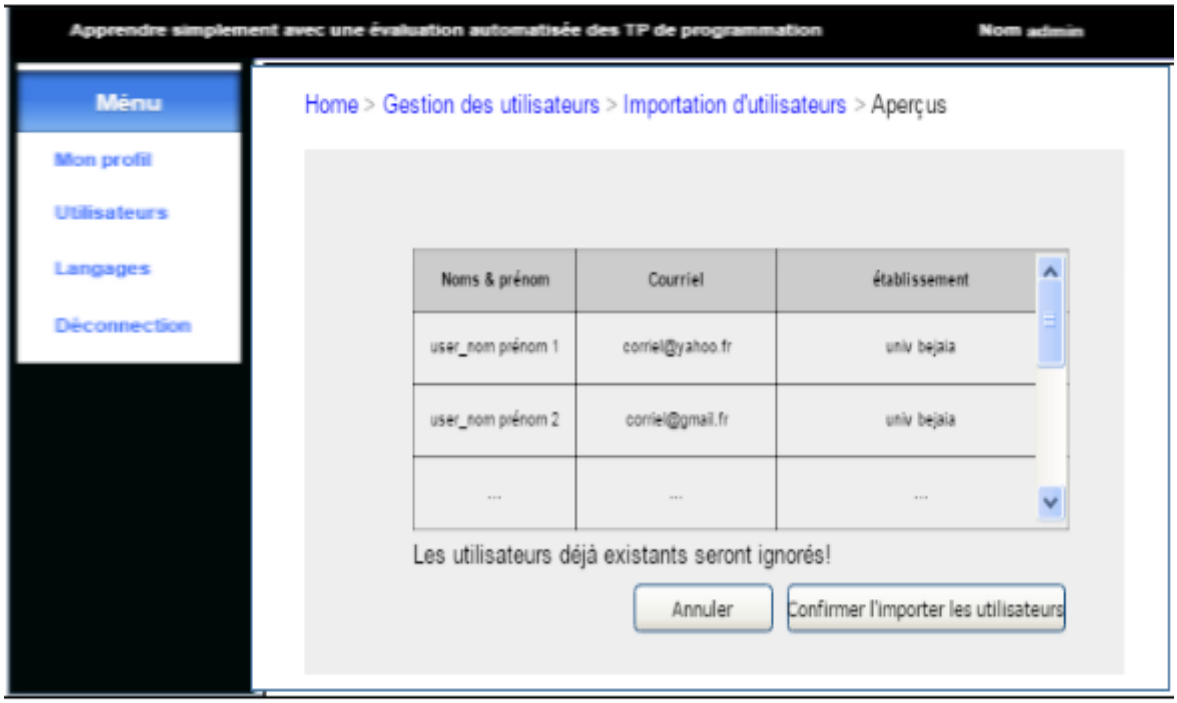


Figure 4 : IHM : Aperçus sur l'importation des utilisateurs

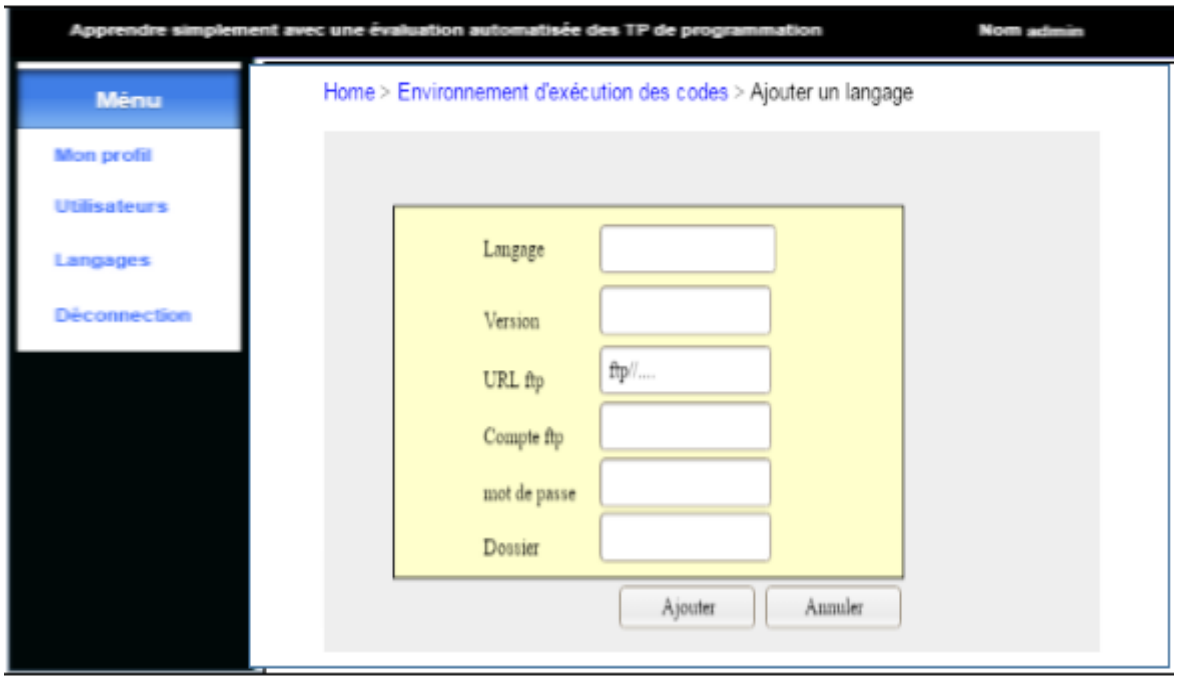


Figure 5 : IHM : Ajouter un langage de programmation

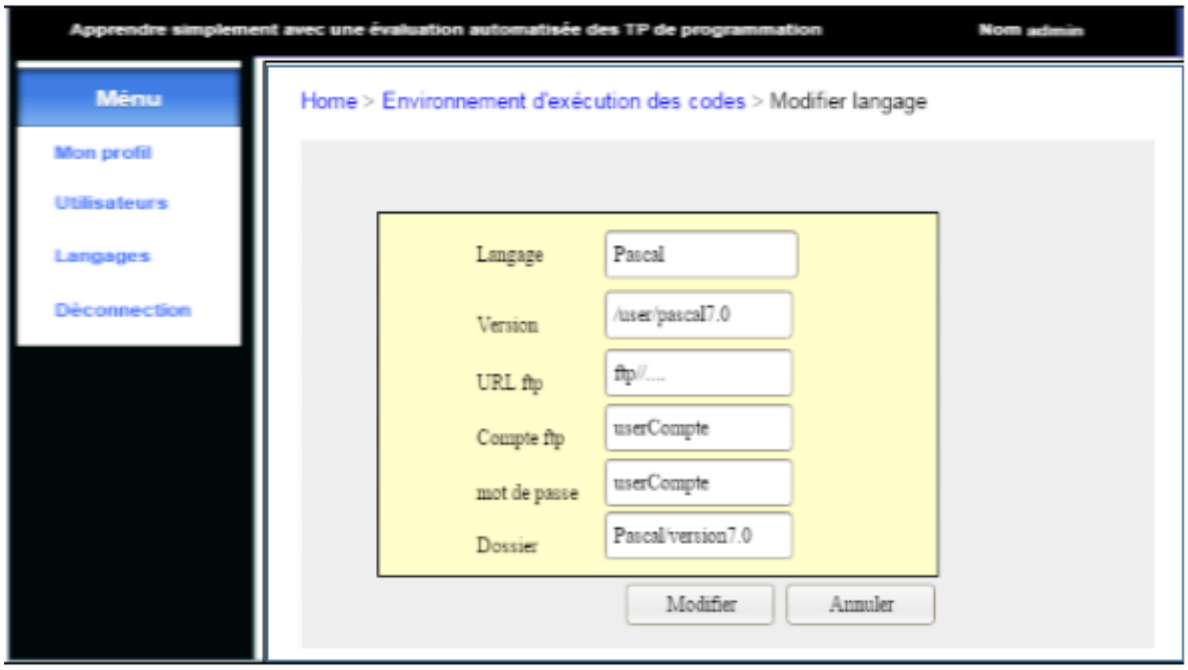


Figure 6 : IHM : Modifier un langage de programmation

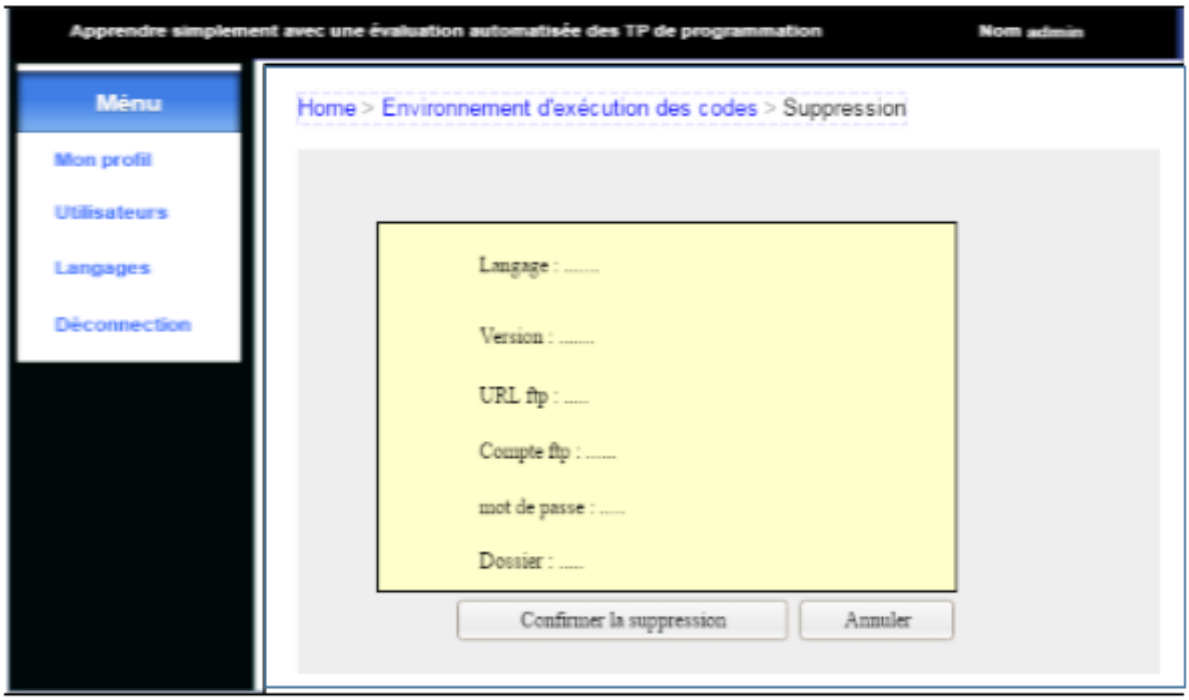


Figure 7 : IHM : Supprimer un langage de programmation

✓ Coté Enseignant :

Figure 8 : IHM Superviser une session

-Cet écran représente la page supervision selon l'étudiant.

-L'enseignant accèdera à cette page avec un clic sur superviser apres le choix de <selon les étudiants> de la figure précidante.

-L'enseignant peut consulté les différents information d'un étudiant.

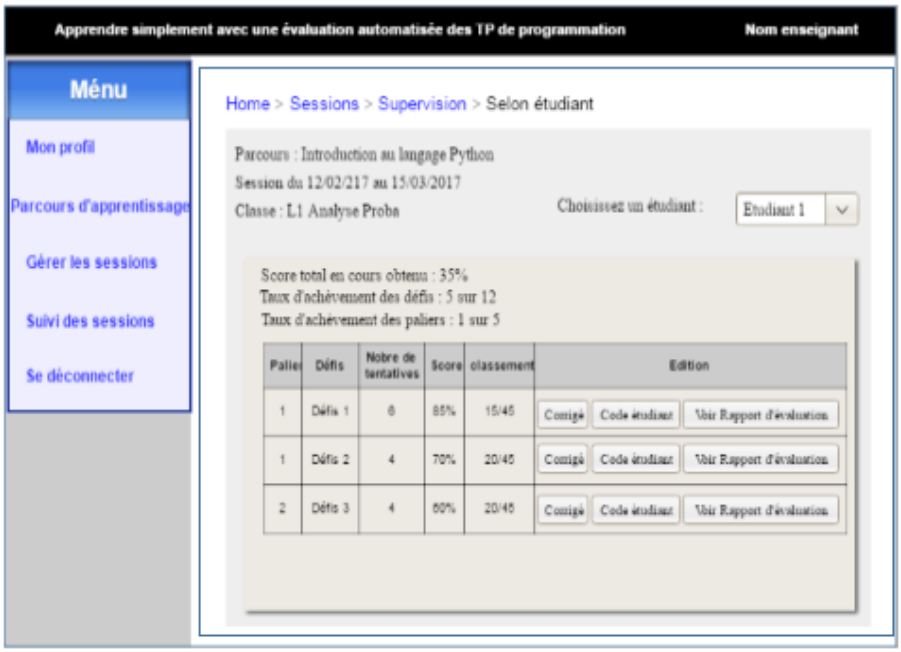


Figure 9 : IHM : Superviser selon l'étudiant

-L'enseignant accèdera à cette page avec un simple clic sur le boutton <voir rapport d'évaluation> de la figure 15.

-L'enseignant peut constaté la méilleur tentative sur un défi et un rapport généré d'une facon automatiser sur un étudiants précise.

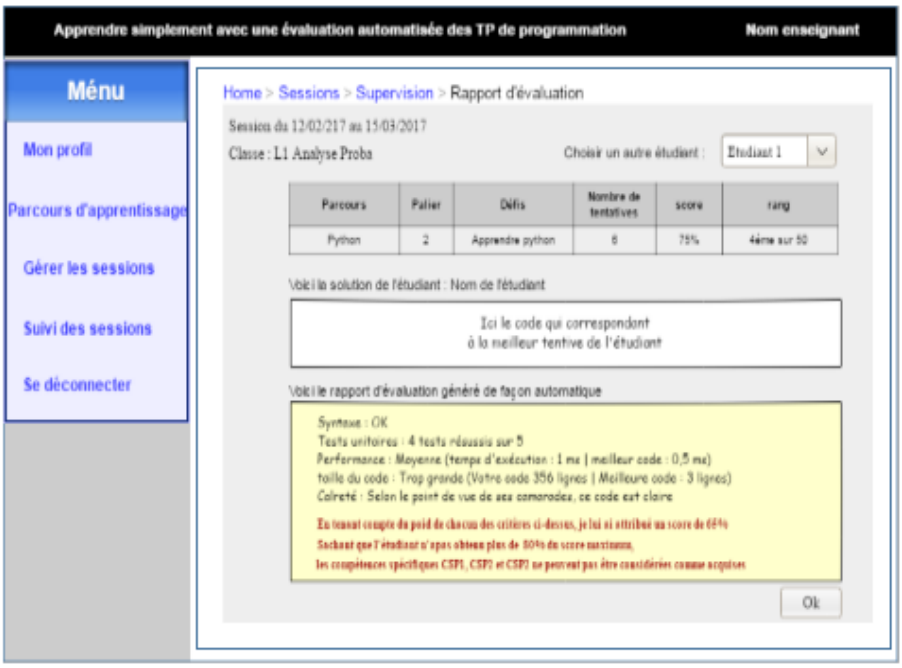


Figure 10 : IHM : Superviser selon le rapport d'évaluation

-L'enseignant accèdera à cette page avec un clic sur superviser apres le choix de <selon un défis>

-L'enseignant constate en détaile un tableau de données pour chaque défi choisi.

Figure 11 : IHM : Superviser selon un défi

-L'enseignant accèdera à cette page avec un clic sur superviser apres le choix de <selon un défis>

-L'enseignant constate en détaile un tableau de données pour chaque défi choisi.

Figure 12 : IHM : Superviser selon un défi

-L'enseignant accèdera à cette page avec un clic sur superviser apres le choix de <selon un paliers>

- L'enseignant constate en détaile un tableau de données pour chaque paliers choisi.

Figure 13 : UHM : Superviser selon un palier

-Cet écran représente la page de profil d'un étudiant avec tous les informations.

-L'étudiant a la possibilité de modifier son compte avec le bouton modifier.

Figure 14 : IHM : Mon profil

-Cet écran représente la page de modifier le profil d'un étudiant.

-L'étudiant peut modifier ces cordonné et de charger une nouvelle photo et ensuite de valider la modification ou de l'annuler.

Figure 15 : IHM : Modifier profil

✓ Coté étudiant :

-L'étudiant doit consulter les compétences attendu et aussi la possibilité de résoudre, consulté les ressources ou demander un indice, comme il peut constater les différents compétences à mobiliser.

Figure 16 : IHM : Découvrir le code secret

-Cet écran représente la page résoudre un défi.

-L'étudiant doit écrire son code dans la zone spécifique

-Le nombre de tentative incrémente pour chaque échec.

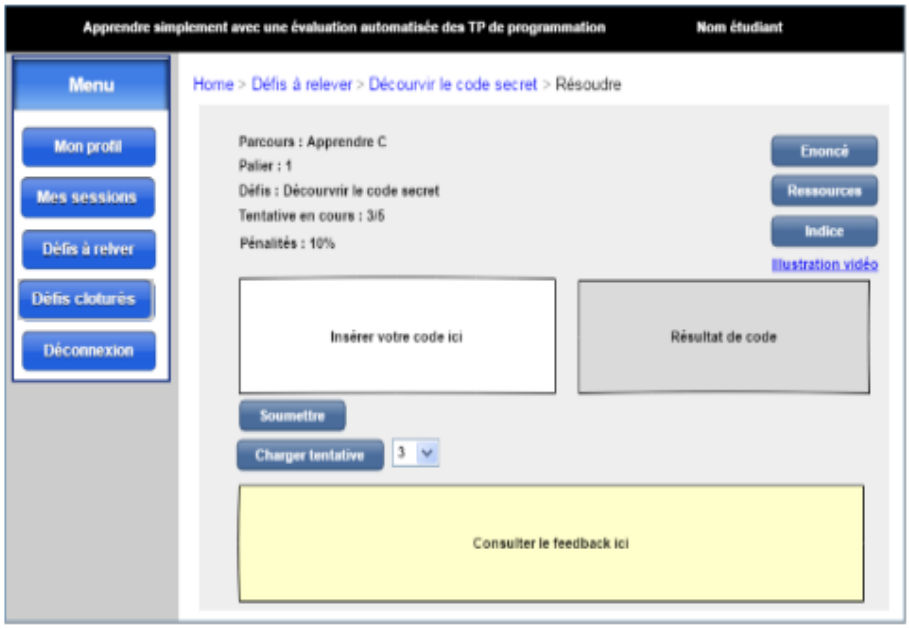


Figure 17 : IHM : Résoudre un défi

-Cet écran représente la page d'illustration d'une vidéo spécifique sur le code secret.

-L'étudiant peut regarder cette vidéo avec un simple clic sur le lien illustration vidéo.

Figure 18 : IHM : Illustration vidéo

-Cet écran représente la page d'accès à des ressources mise par un enseignant.

-Avec un simple clic sur le boutton ressource, l'étudiant peut télécharger les resources suivants : tuto, article,livre ou bien un document.

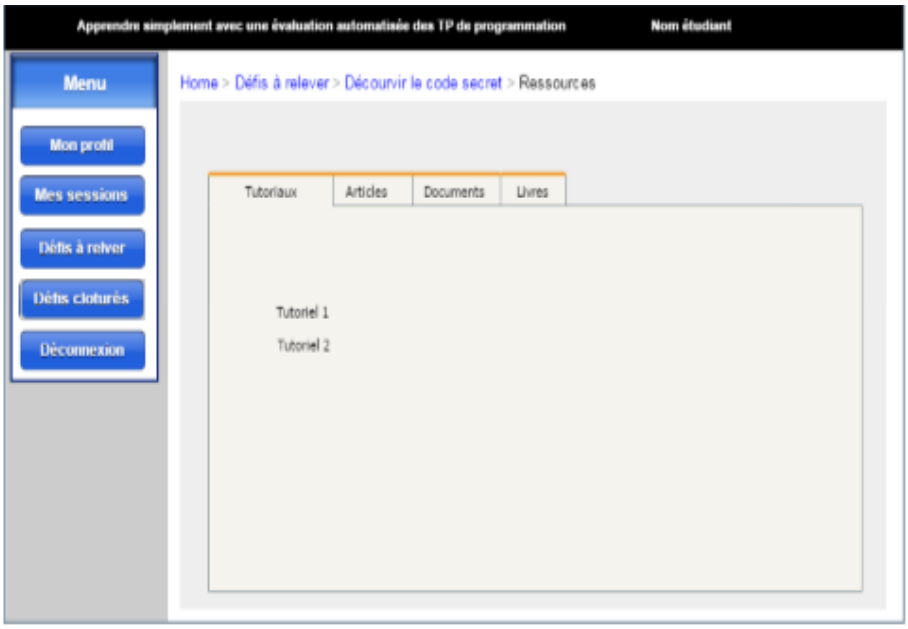


Figure 19 : IHM : Voir Ressource

-Cet écran représente la page sur les codes des autres étudiants qui ont déjà réussi leur défi.

-Une vue sur le code et le feedback d'un défi est apparu à chaque tentative choisi.

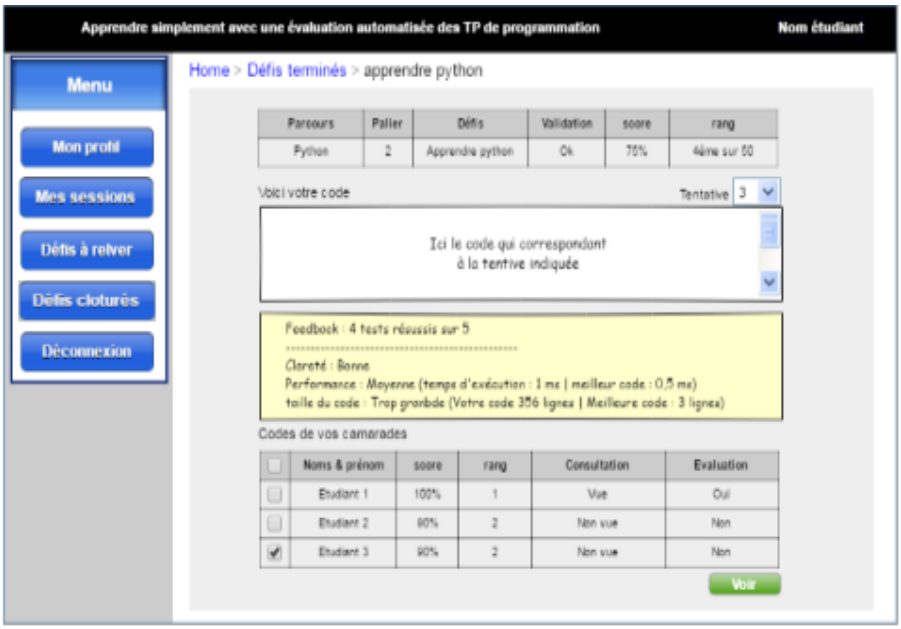


Figure 20 : IHM : Voir les codes

-Cet écran représente la page sur une erreur détecté par un étudiant.

-L'étudiant doit signaler cette erreur afin de la corriger à travers d'un enseignant qui s'occupe de ce parcours.

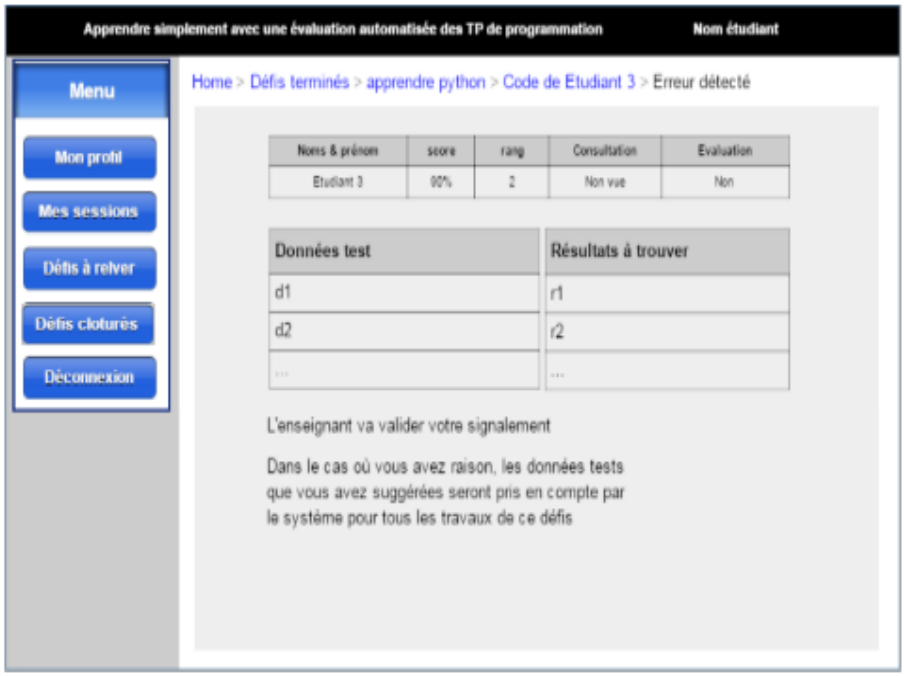


Figure 21 : IHM : Erreur détecter

Annexe 2
Diagrammes de séquence système

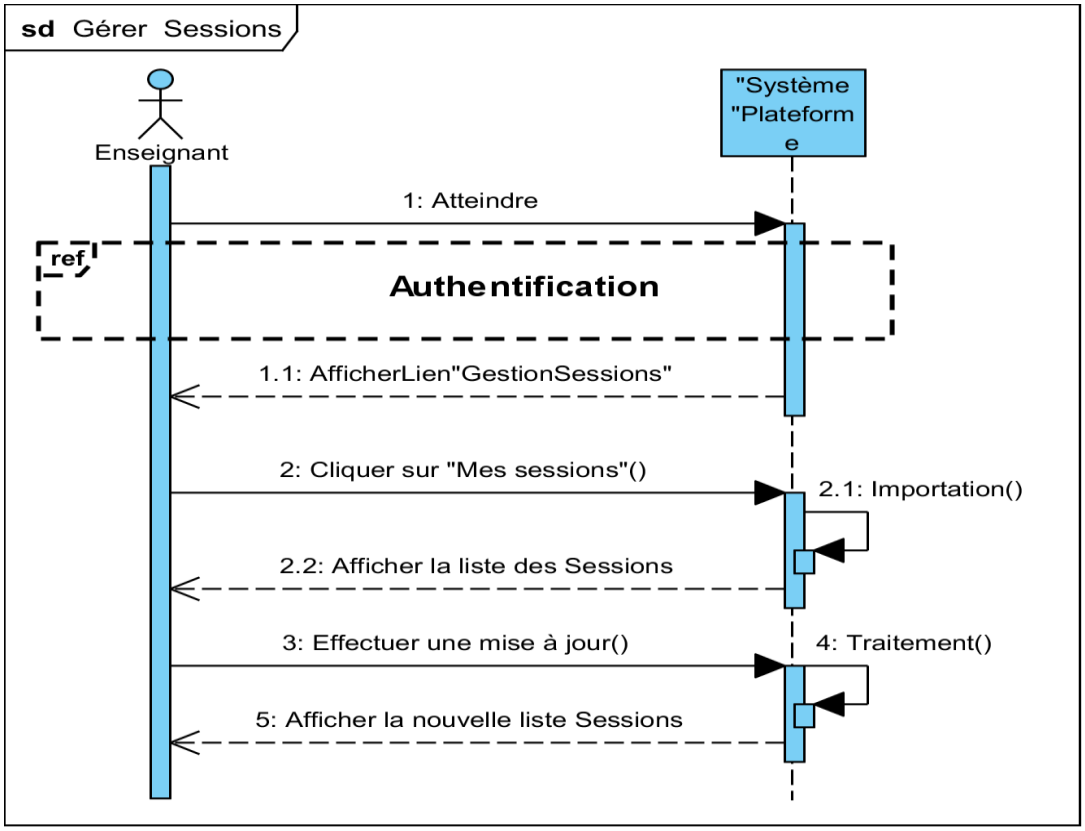


Figure 22 : DSS : gérer session

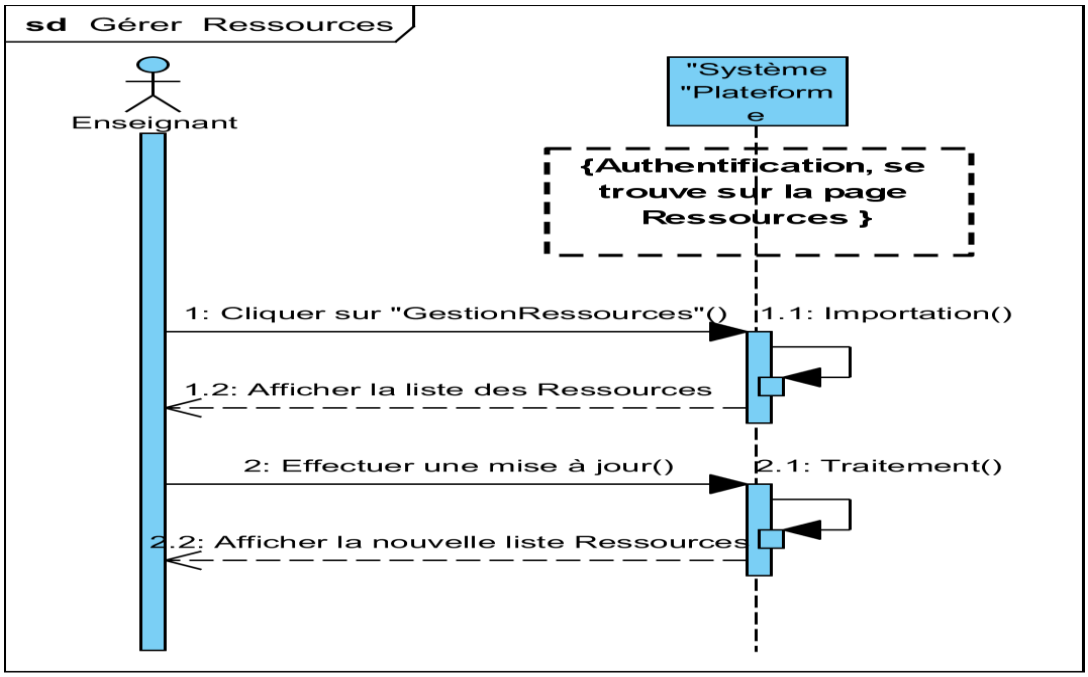


Figure 23 : DSS : gérer les ressources

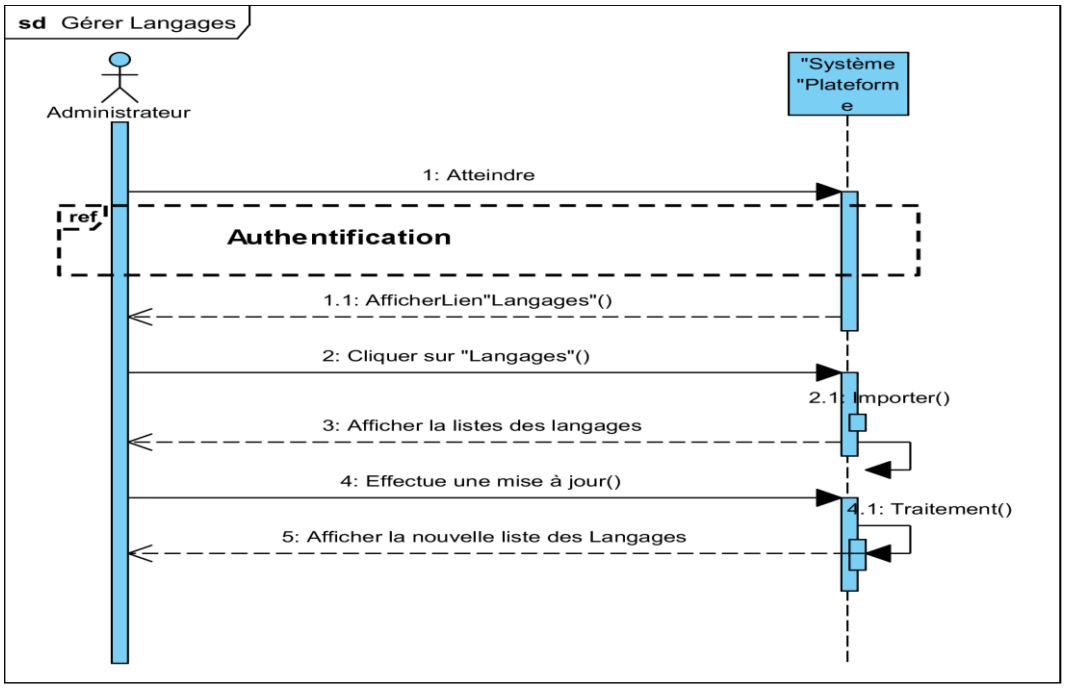


Figure 24 : DSS : gérer langages

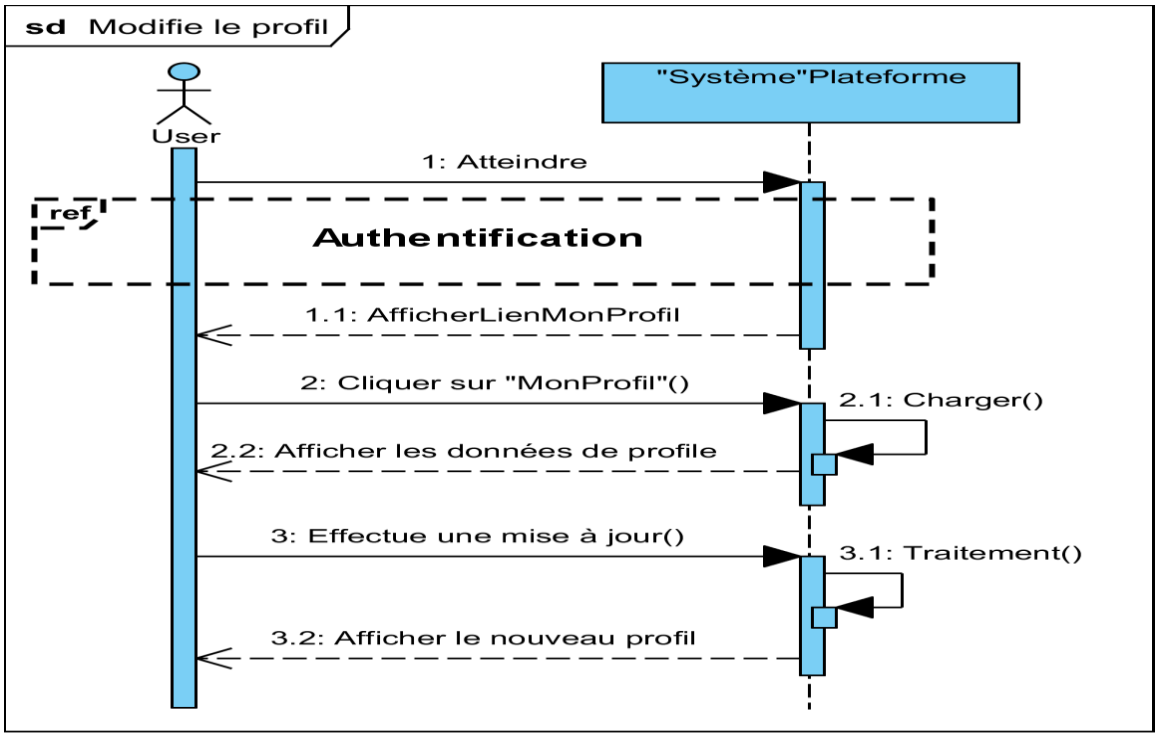


Figure 25 : DSS : modifier le profil

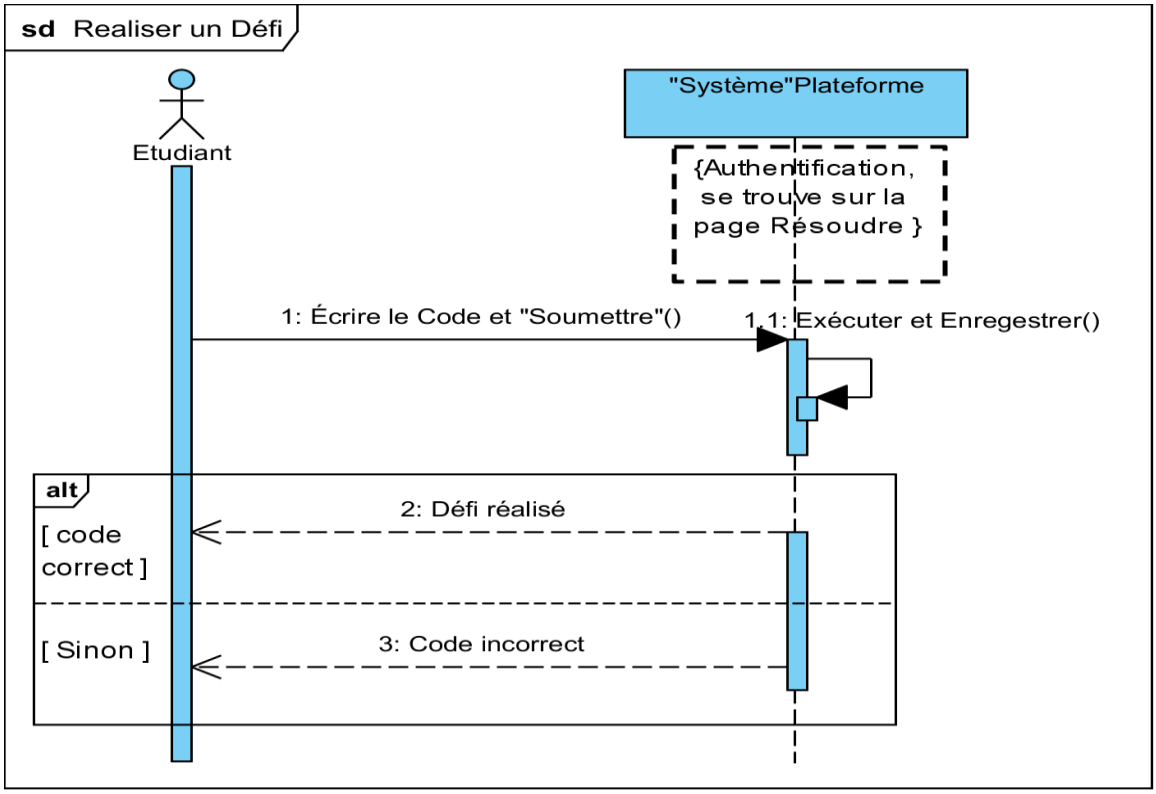


Figure 26 : DSS : réaliser un défi

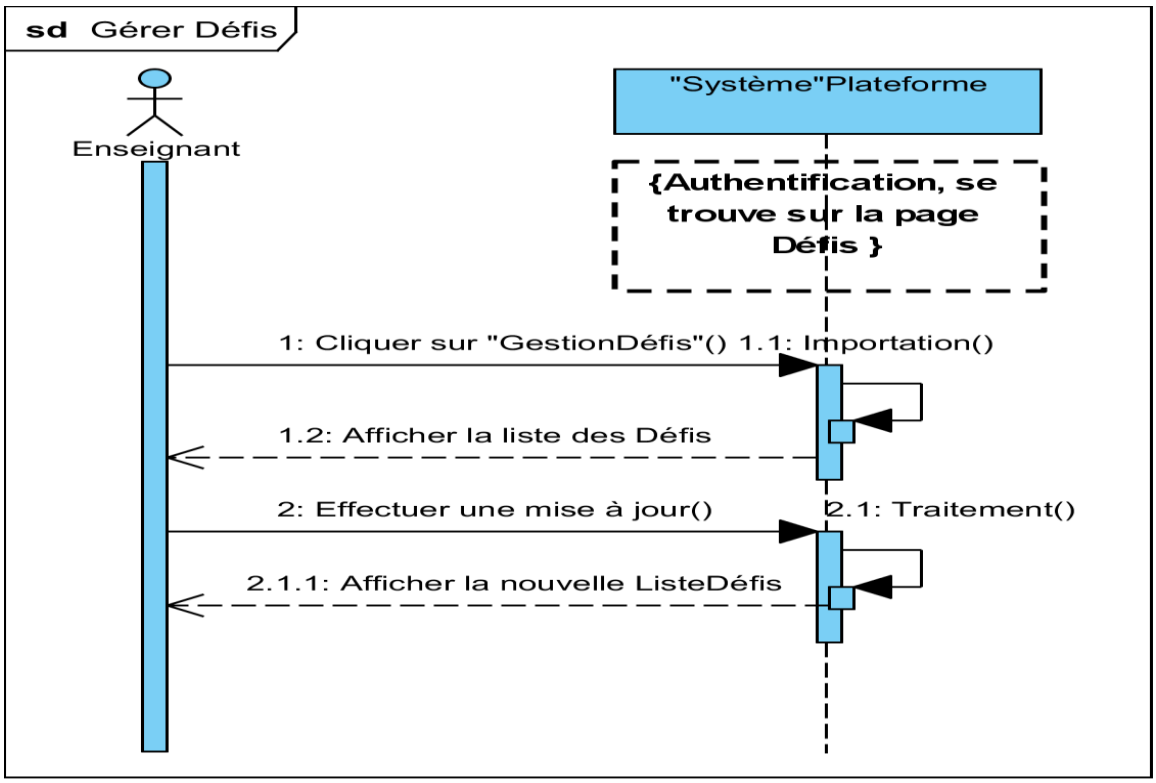


Figure 27 : DSS : gérer les défis

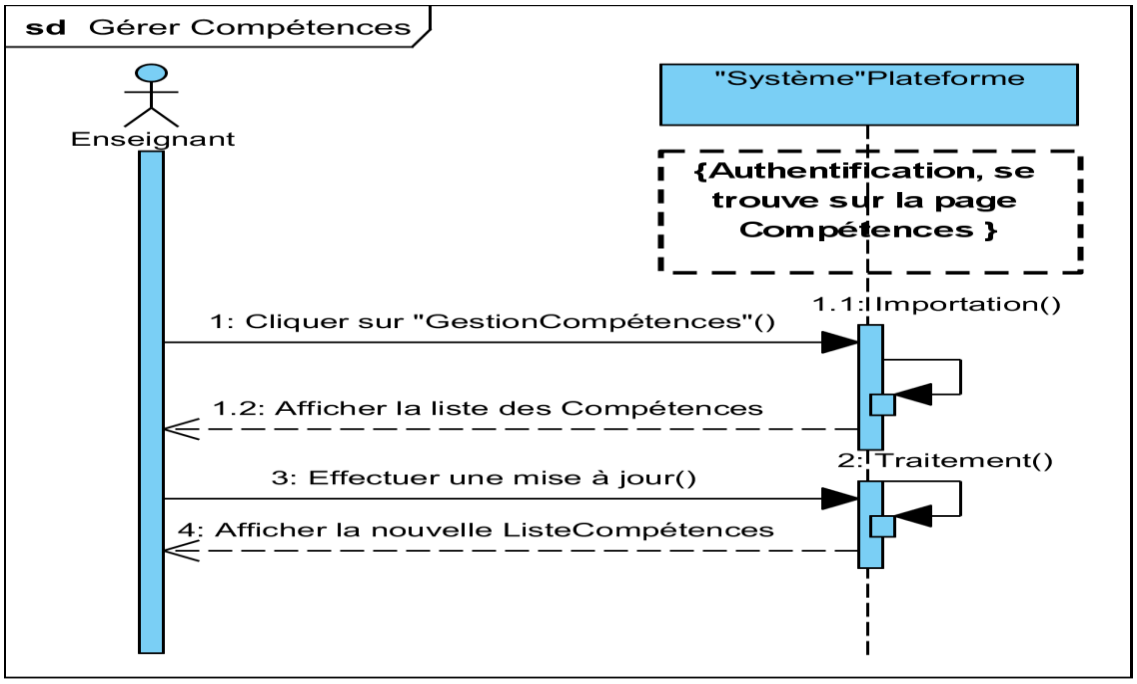


Figure 28 : DSS : gérer compétence générale

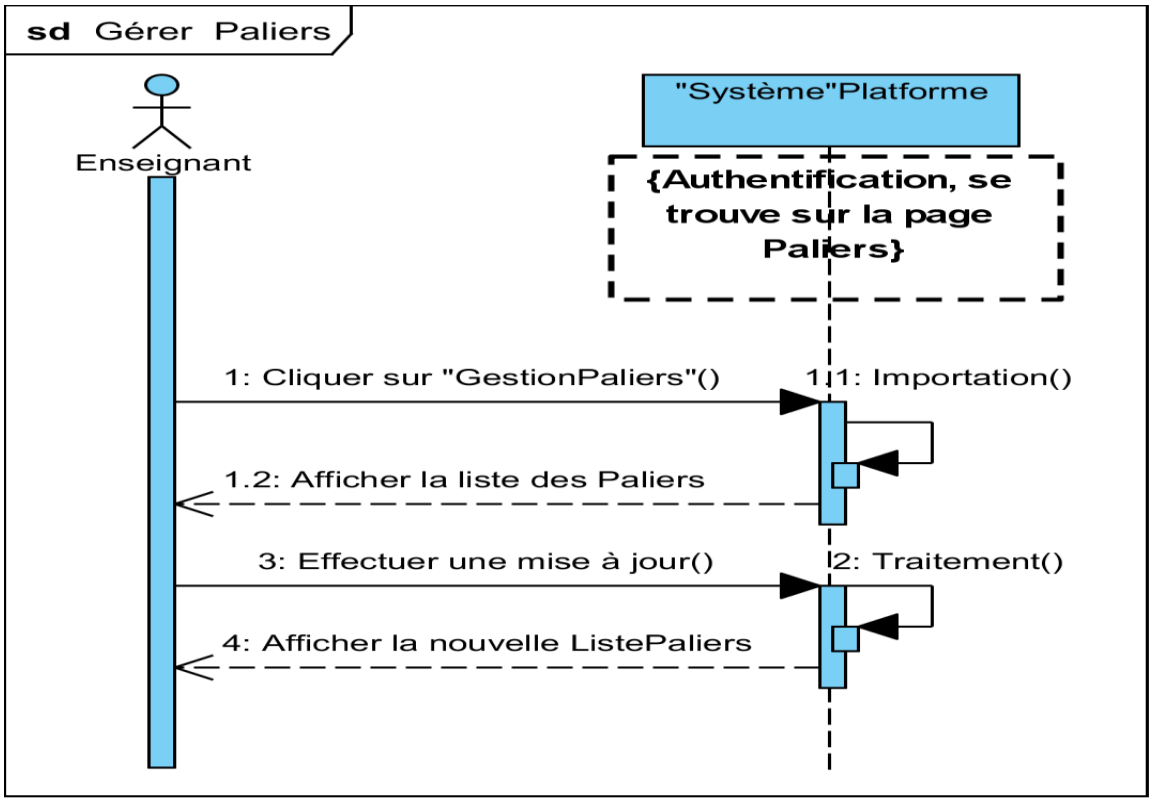


Figure 29 DSS : gérer paliers

Annexe 3
Diagrammes de classe participantes

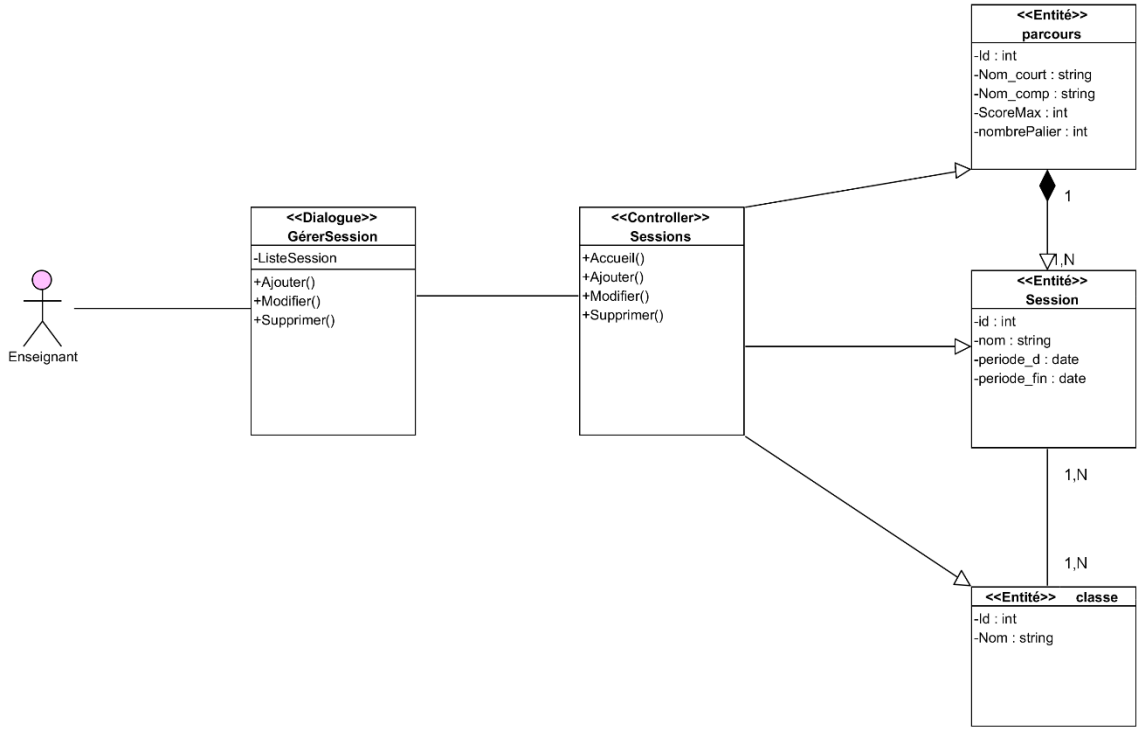


Figure 30 : DCP : Gérer session

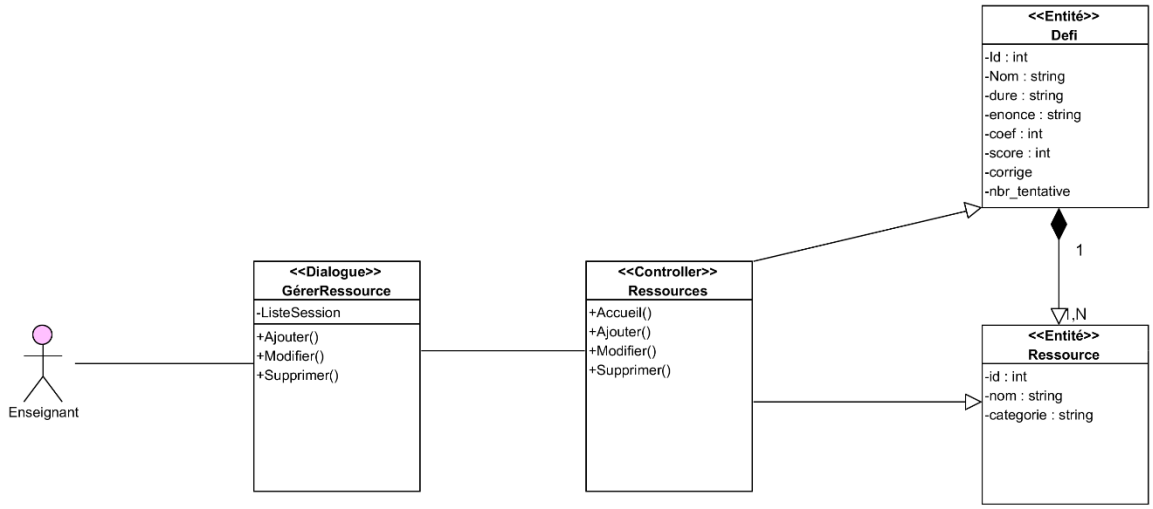


Figure 31 : DCP : Gérer ressource

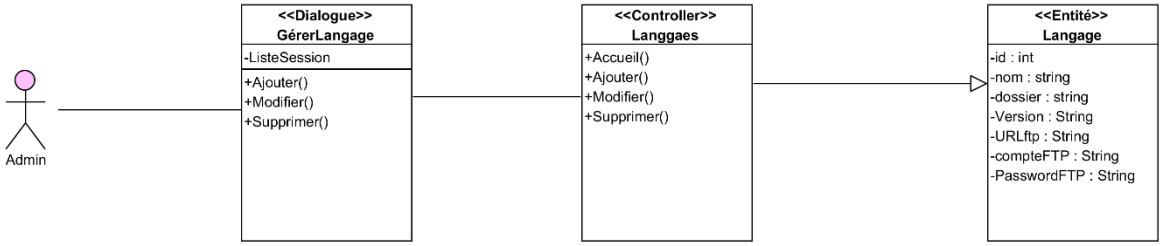


Figure 32 : DCP : Gérer langage

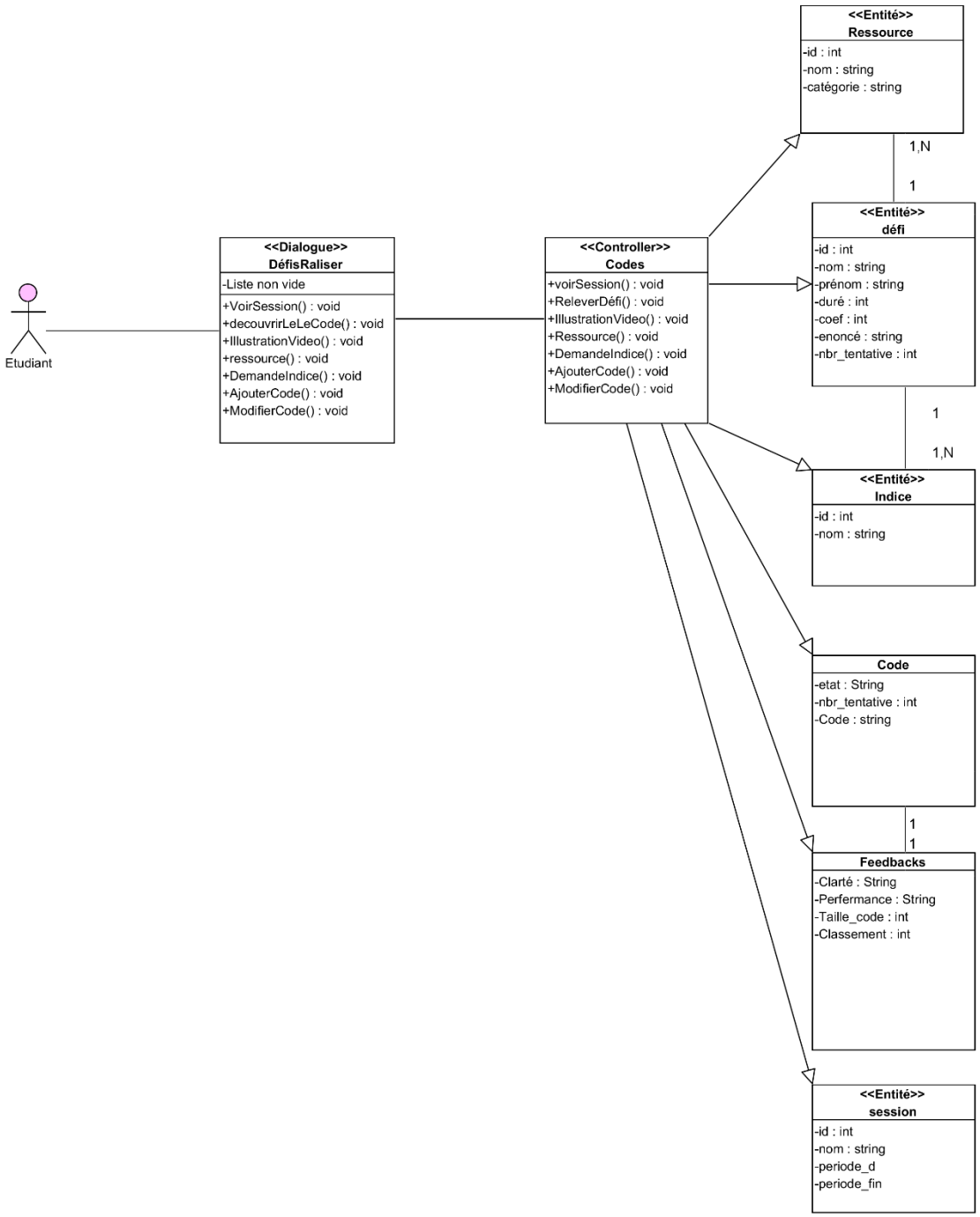


Figure 33 : DCP : réaliser un défi

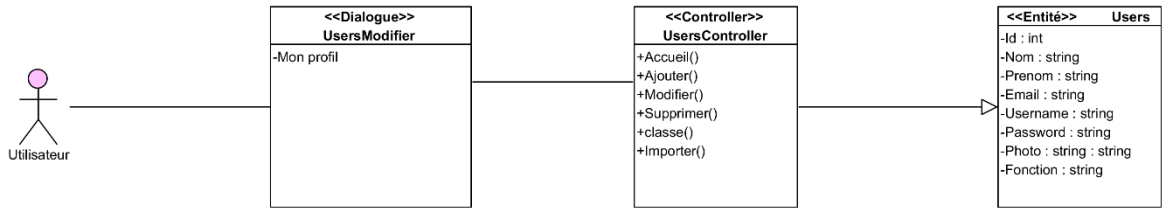


Figure 34 : DCP : Modifier le profil

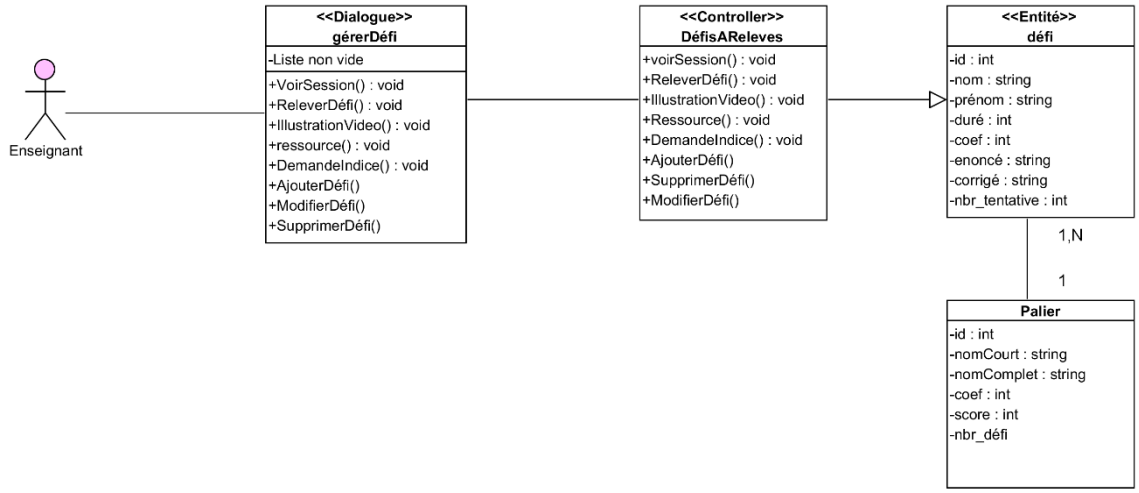


Figure 35 : DCP : Gérer défi

Annexe 4
Diagrammes d'interaction

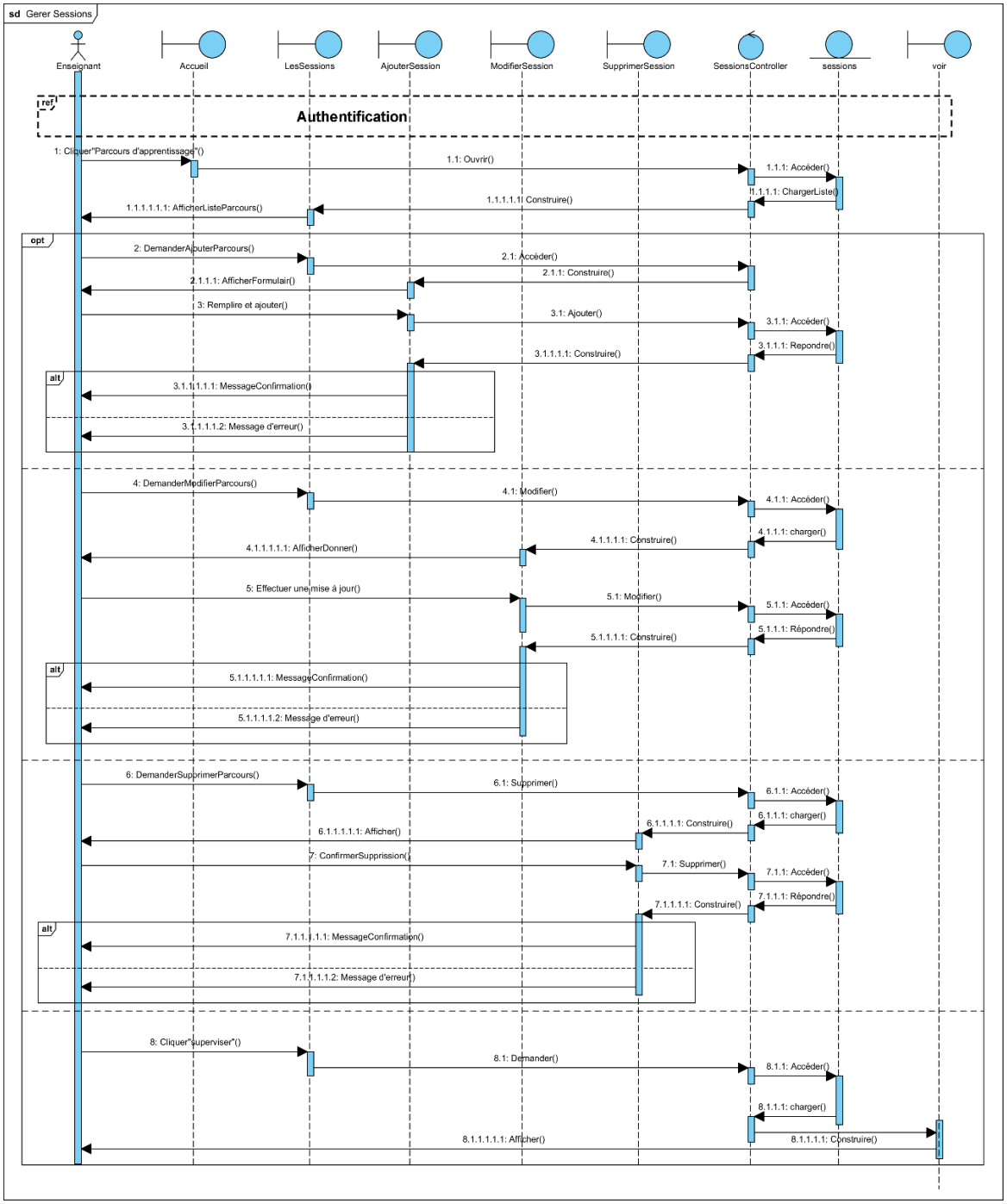
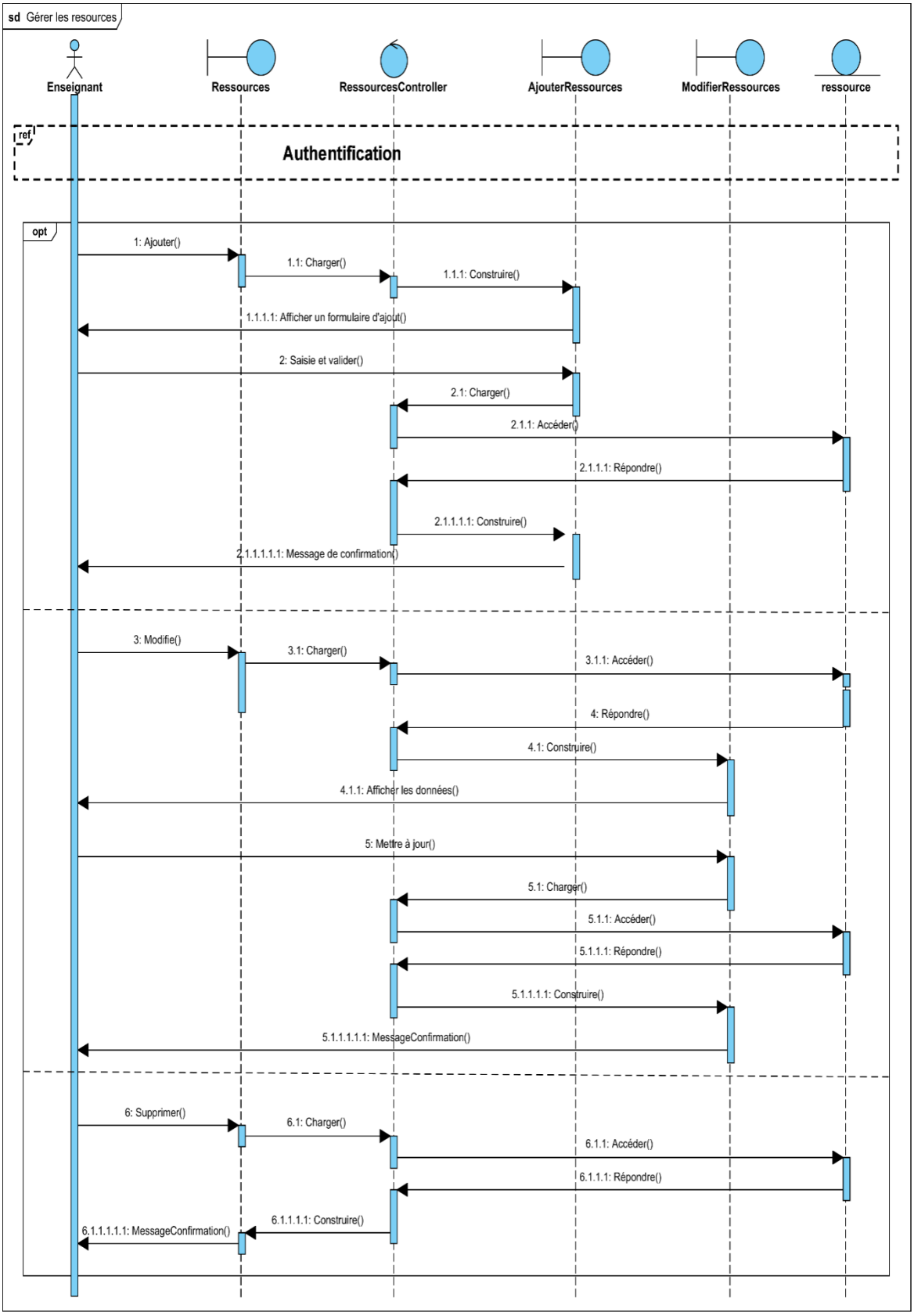


Figure 36 : DSD : gérer session

Figure 37 : DSD : gérer ressource



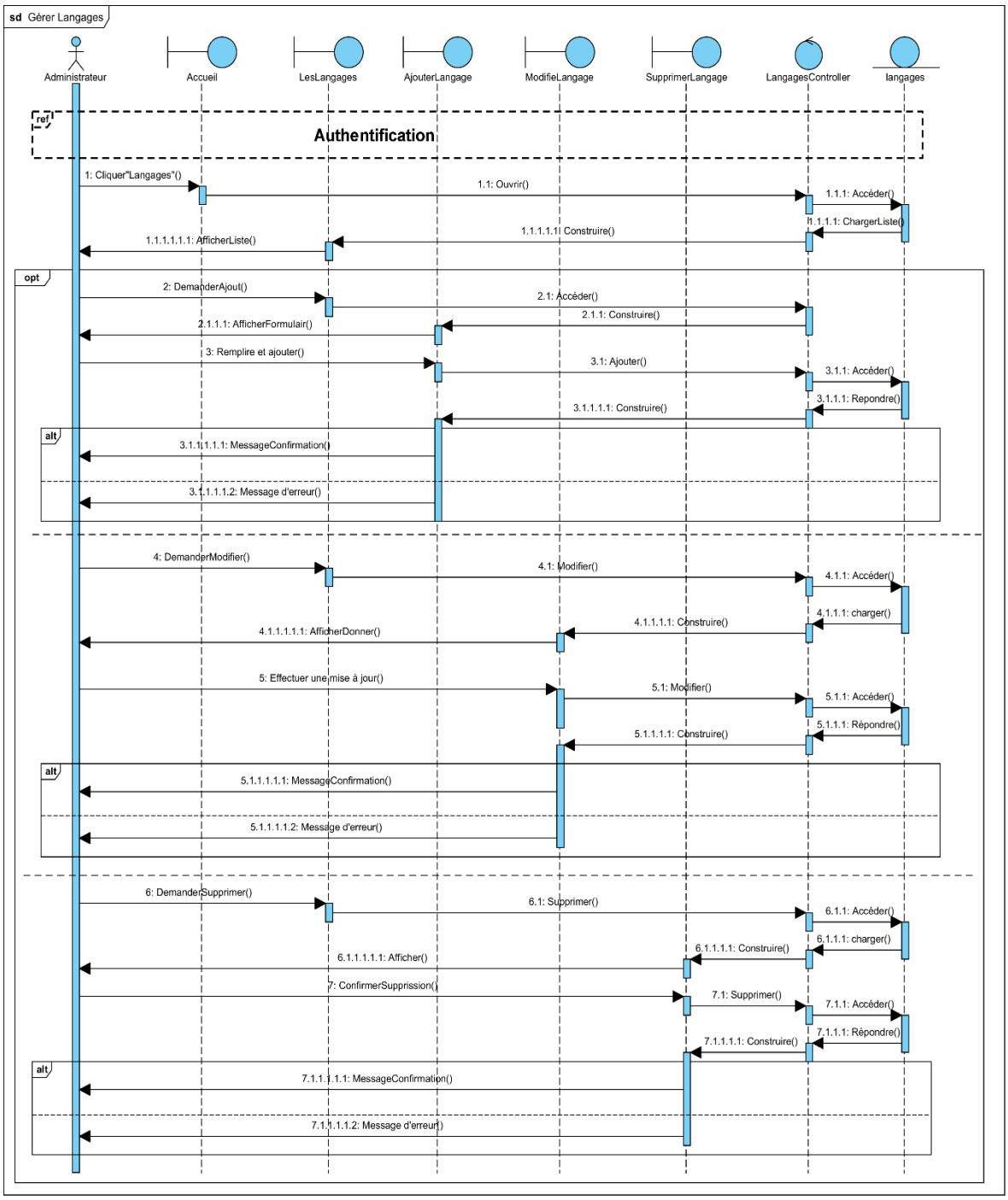


Figure 38: DSD : gérer langages

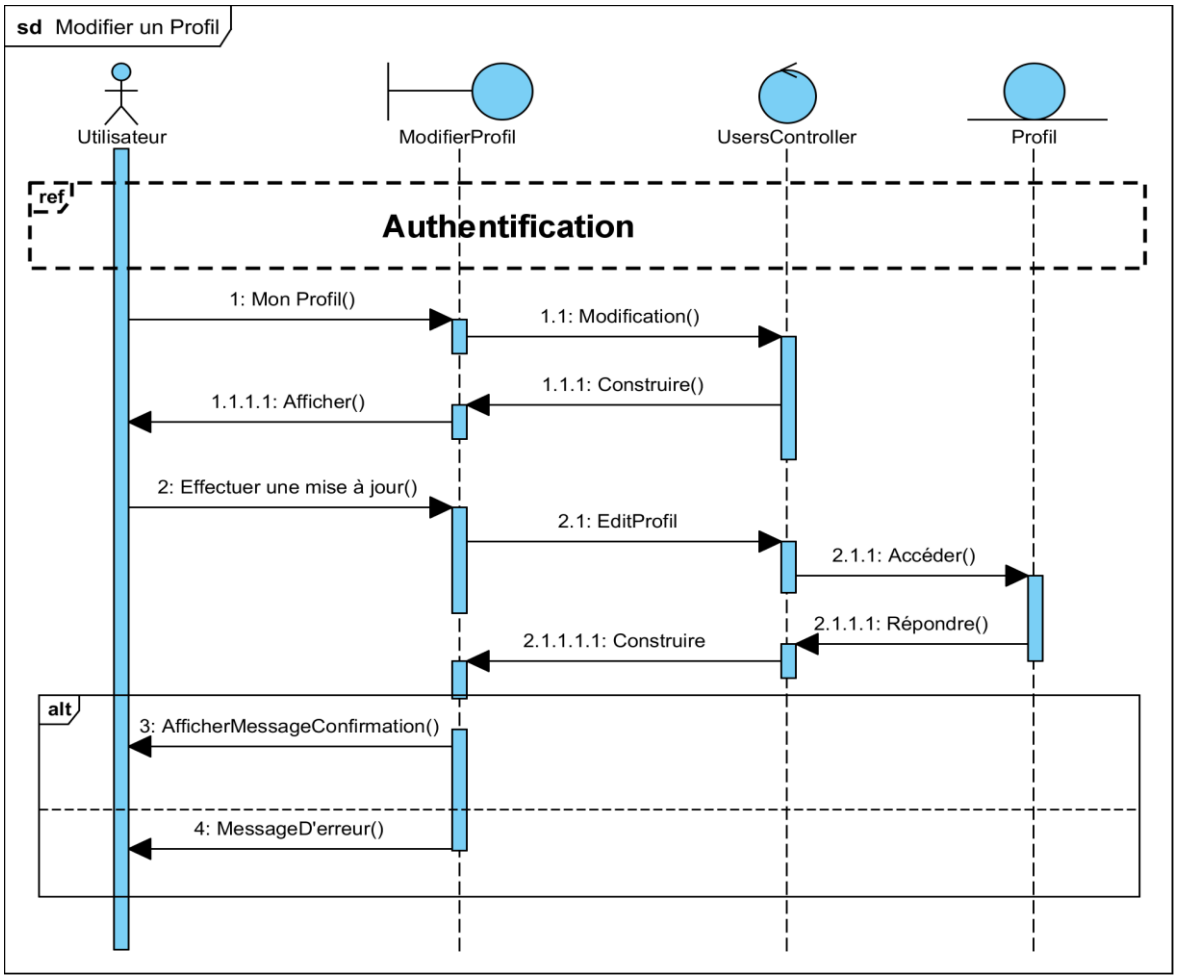


Figure 39: DSD : modifier le profil

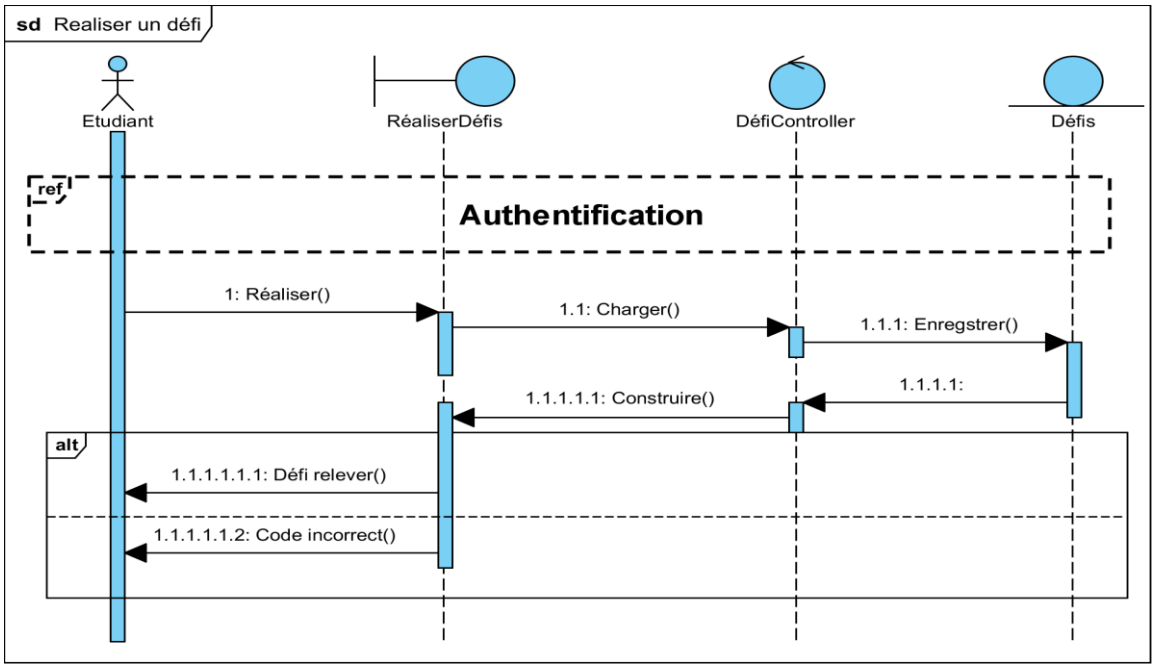


Figure 40: DSD : Réaliser un défi

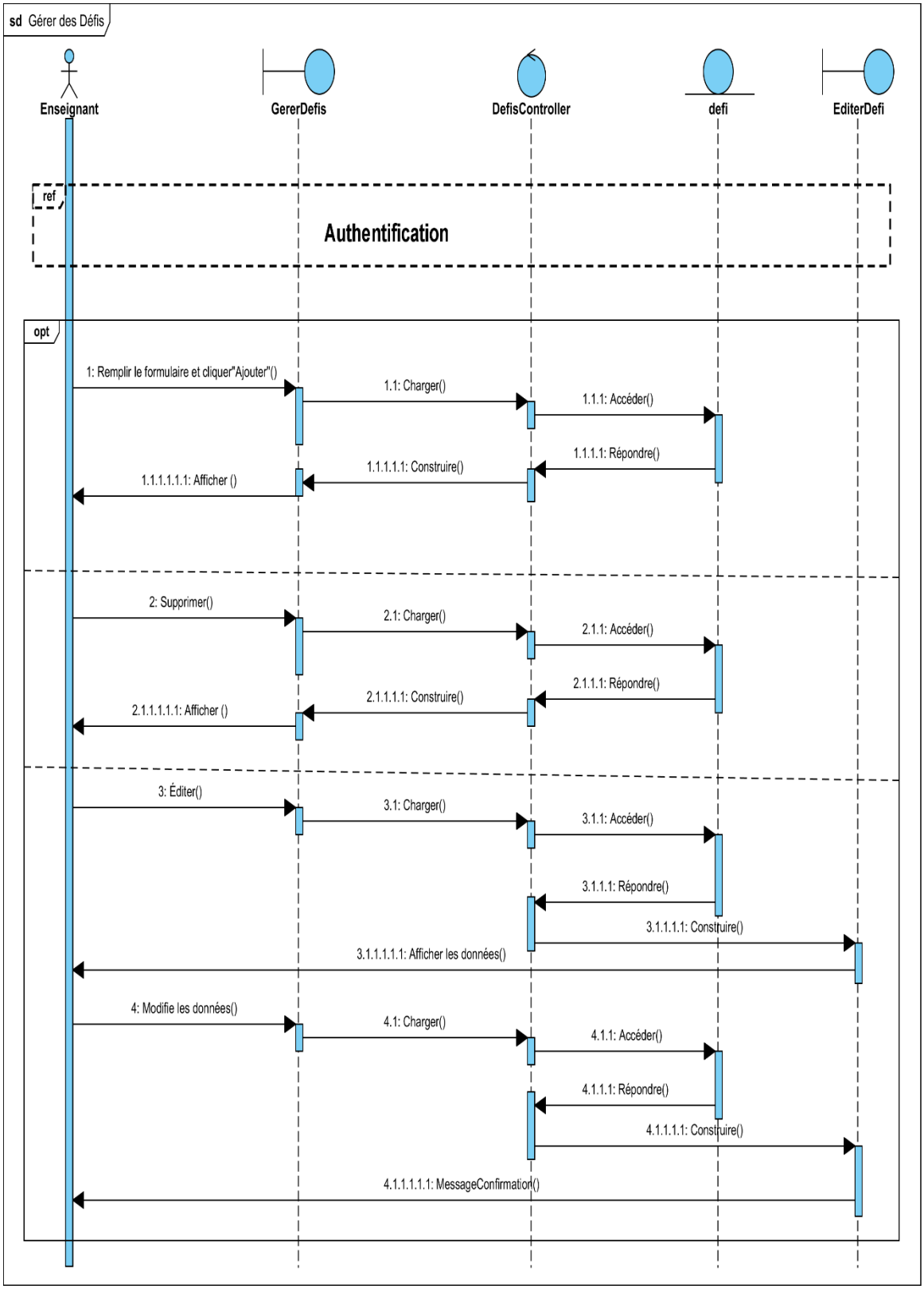


Figure 41 : DSD : gérer des défis

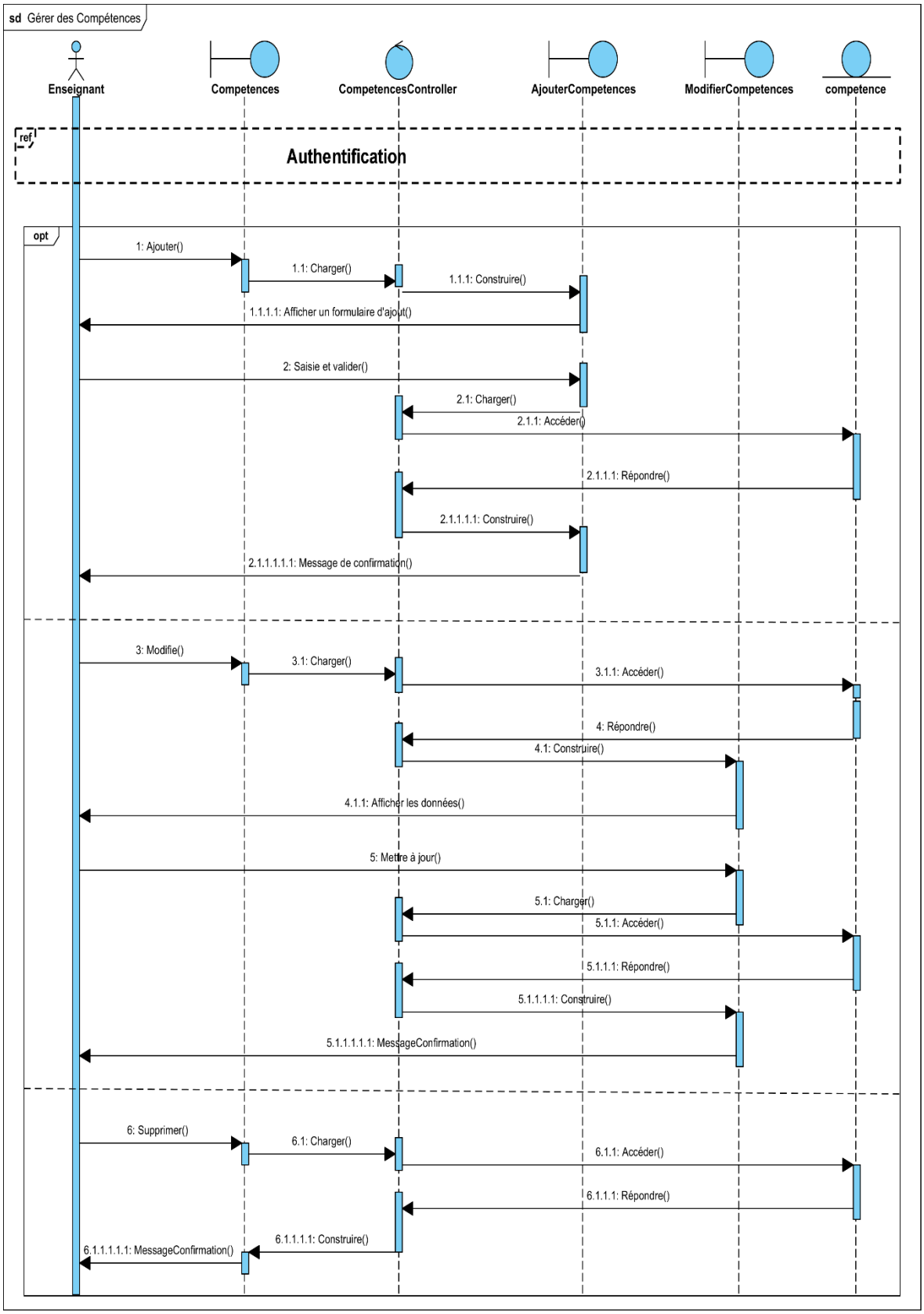


Figure 42 : DSD : gérer les compétences générales

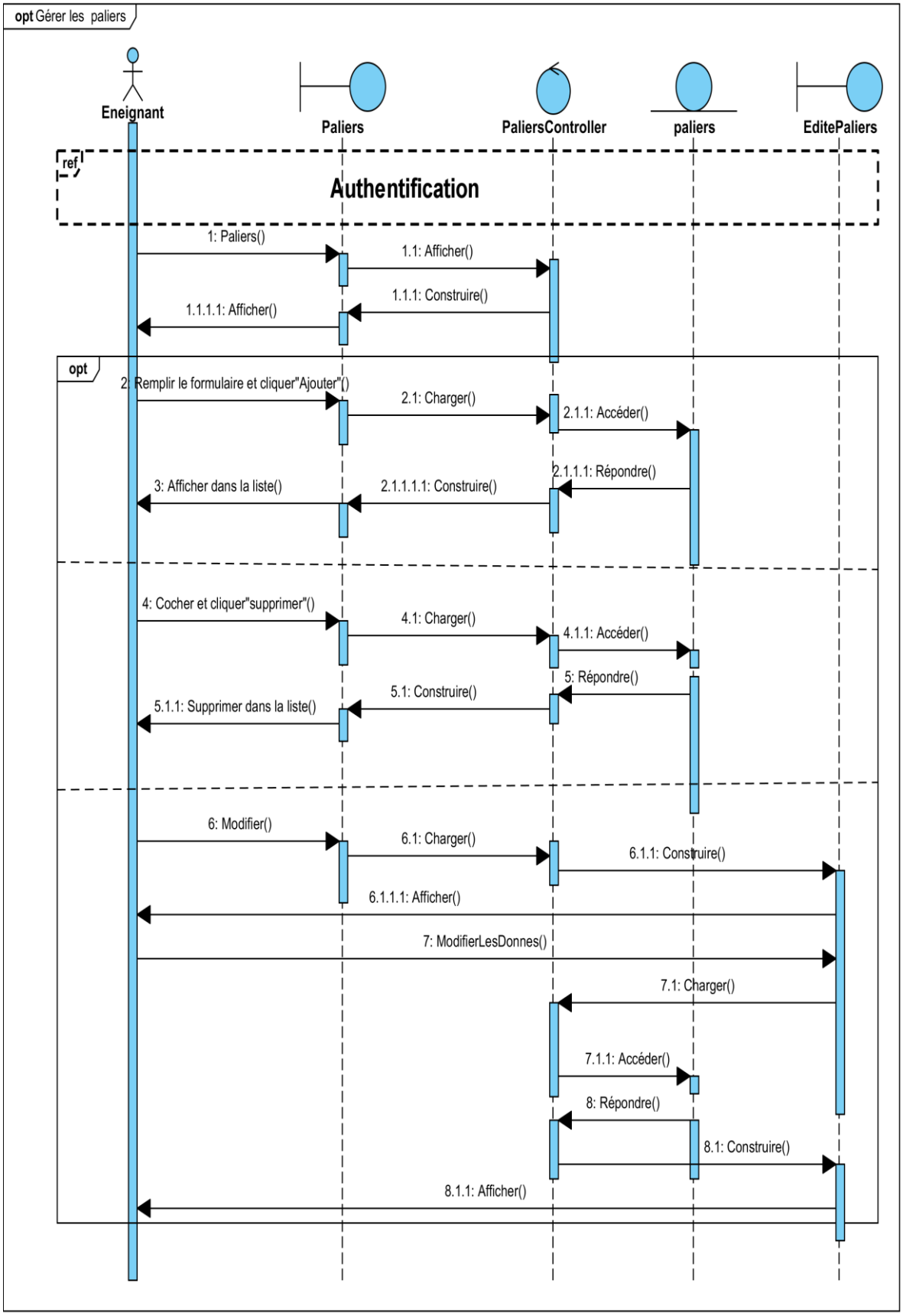


Figure 43 : Gérer les paliers

RÉSUMÉ

La problématique de l'évaluation des TP de programmation est cruciale à la fois pour les étudiants qui subissent une forte pression surtout lors des examens, mais aussi pour les enseignants qui doivent corriger un grand nombre de travaux. Pour pallier à cette situation, des solutions ont été proposées depuis un peu plus d'une décennie. Leur but est d'automatiser l'évaluation des codes soumis par les étudiants. Notre projet, s'inscrit dans ce domaine de l'évaluation automatisée. Sa particularité, en plus de rendre des rapports d'évaluation automatisée, permet de gérer des parcours d'apprentissage offrant ainsi un cadre plus complet pour la gestion et le suivi des TP de programmation.

MOTS CLÉS : Evaluation automatisée, TP de programmation, Apprentissage en ligne, parcours d'apprentissage

SUMMARY SINTESI RESUMEN ZUSAMMENFASSUNG

The assessment of programming codes is a crucial issue both for students who are under heavy pressure especially during exams but also for teachers who have to correct a large number of works. To overcome this situation, solutions have been proposed for a little more than a decade. Their purpose is to automate the assessment of codes submitted by students. Our project is part of this area of automated assessment. In addition to providing automated assessment reports, its unique feature is the management of learning paths, thus providing a more complete framework for the management and monitoring of programming tasks.

KEYWORDS: Automated assessment, Programming codes, Learning Online, Learning paths