Analyse des besoins

Coditor

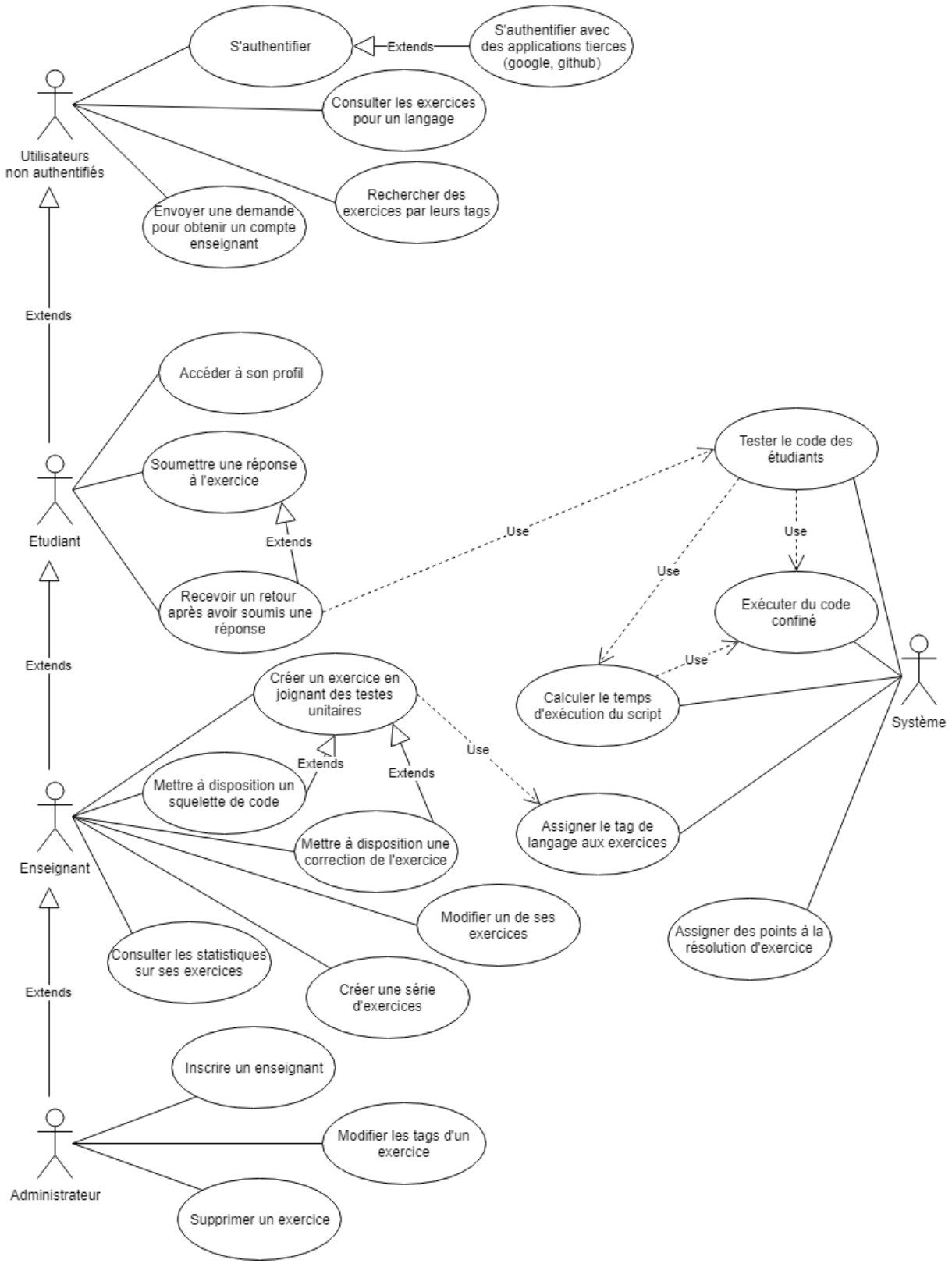
Environnement d'apprentissage et d'auto évaluation de la programmation

Cette application permettra à des étudiants de réaliser des exercices proposés par des enseignants, de façon à s'améliorer en programmation mais également à s'auto-évaluer grâce à un retour automatique et rapide de leur travail.

Cordier - Ralli - Rohrbacher - Zink

1. Cas d’utilisation

Pour une meilleure qualité, veuillez vous référer au fichier pdf joint dans l’archive.



2. Périmètre du projet

1. Le système exécute le code dans un environnement confiné
2. Le système test le code des étudiants
3. Le système assigne des tags aux exercices
4. Le système calcule le temps d'exécution du code envoyé par l'étudiant
5. Le système assigne des points à la résolution de l'exercice

6. Un utilisateur peut s'identifier/se déconnecter/s'inscrire
7. Un utilisateur peut s'identifier avec des applications tierces (google, github)
8. Un utilisateur peut demander à obtenir un compte enseignant
9. Un utilisateur peut consulter les exercices pour un langage
10. Un utilisateur peut rechercher des exercices pas leurs tags

11. Un étudiant peut accéder à son profil
12. Un étudiant peut soumettre une réponse à un exercice choisi
13. Un étudiant peut recevoir la résolution d'un exercice

14. Un enseignant peut créer un exercice en joignant des tests unitaires
15. Un enseignant peut mettre à disposition un squelette de départ pour l'exercice
16. Un enseignant peut mettre à disposition une correction de l'exercice
17. Un enseignant peut créer une série d'exercices
18. Un enseignant peut consulter les statistiques de ses exercices
19. Un enseignant peut modifier un de ses exercices

20. Un administrateur peut accéder la demande de création de compte d'un enseignant
21. Un administrateur peut changer les tags d'un exercice
22. Un administrateur peut supprimer un exercice

OPTIONNEL

23. Un enseignant peut créer une série d'exercice privée
24. Un enseignant peut créer des groupes d'étudiant
    → Une url sera généré afin que les étudiants puissent rejoindre le groupe

3. Etude de l’existant

CodinGame -

Apprentissage de la programmation par le jeu : Chaque étudiant vas avoir un objectif sous forme d’un jeux vidéo (faire tourner un véhicule, attaquer un personnage, etc …).
L'utilisateur code ce qui est attendu, l’application lui donnes un message personnalisé si ce code contient une erreur. Après quoi, l’utilisateur voit comment réagis ce qu’il a créer via le jeu, et devra (ou non) réfléchir et adapter son travail de façon à battre le score des autres joueurs et de répondre à la problématique posée. Chaque “niveau” apprends au joueur une fonctionnalité de la programmation (Variables, puis boucles, etc …).

CodeCademy -

CodeCademy reprends en partie le style d’OpenClassrooms, sauf que les cours sont créer par des personnes travaillant pour ce site. Chaque étudiant vas suivre le cours et fait en même temps les exercices à réaliser. Ces exercices seront testé automatiquement et il aura un retour simple (réussite, une erreur, etc …).
- Cette fonctionnalité est notre fonctionnalité principale : Gérer un code écrit par l’étudiant, le tester, retourner un message personnalisé.

SoloLearn -

SoloLearn est une application mobile permettant aux étudiants d’apprendre la programmation via un système “cours - exercice “. Chaque page d’un cours explique un concept, une fois cette page suivi, l’utilisateur à un exercice sous forme de “texte à trou” à remplir avec un objectif. A la moindre erreur, l’application propose à l’utilisateur de relire le cours ou de réessayer.
- Le programme ne test pas le code mais attends que l’utilisateur donnes le code exact attendu afin de répondre au problème.

Udemy -

Udemy est un site principalement anglais mais très utilisé également. Il se base sur des cours (payants afin de rémunérer les enseignants) principalement en vidéo. N’importe qui peut créer un cours de la façon qu’il souhaite. Il ne se base pas sur des exercices mais sur des projets concrets à réaliser.
Les grandes différences sont que les projets sont corrigés par le créateur du cours qui lui rédige le retour, mais également le prix des cours et des projets, nos exercices seront gratuits.

4. Acteurs

Les utilisateurs non authentifiés

Ils ont accès à de nombreuses fonctionnalités comme consulter la liste des exercices en la parcourant en fonction d'un langage de programmation ou en effectuant une recherche par tags.

Naturellement, ils peuvent également s'inscrire ou se connecter en créant leur compte sur le site ou en s'identifiant avec une application tierce (comme google par exemple). Si l'on souhaite obtenir un compte de niveau enseignant, il faut passer par un formulaire spécifique pour y joindre des pièces justificatives que les administrateurs traiteront.

Les étudiants

Ceux ci possèdent des droits supplémentaires. Ils peuvent accéder à leur profil et y modifier leurs informations.

Ils peuvent aussi, évidemment, tenter de répondre à un exercice que le système traitera. Ce dernier enverra un retour sur le travail produit par l'étudiant.

Les enseignants

Ils peuvent créer des exercices en joignant plusieurs éléments :

- un fichier de testes unitaires (obligatoire)
- une liste de tags(obligatoire)
- un squelette de code (optionnel)
- un script de correction (optionnel)

Naturellement, ils peuvent modifier leurs exercices à tout moment. Ils peuvent aussi créer une série d'exercices en y ajoutant des exercices qu'ils n'ont pas forcément créé eux mêmes.

Finalement, en se rendant sur le page de profil, ils peuvent accéder à des informations statistiques sur les exercices qu'ils ont créer.

Les administrateurs

Les administrateurs peuvent examiner les demandes des utilisateurs pour obtenir des comptes enseignant : si elles sont conformes ils peuvent alors créer le compte et le transmettre à la personne concernée par mail ou un tier moyen.

Enfin, ils ont la possibilité de modifier les tags d'un exercice ou de le supprimer.

5. Cas d’usages

Un enseignant en langage de programmation JavaScript, Léono, 64 ans, souhaite créer un exercice réalisable par tous.
Pour ce faire, il se rend sur l’application et envoie une demande d’inscription.
Après acceptation l’enseignant va ensuite sur son profil et décide de créer un nouvel exercice.
Arrivé sur la page de création d’un exercice, Léono doit alors renseigner : un titre, une description, le sujet complet, des tags, puis il télécharge ses testes unitaires, un squelette et une correction. Il vérifie son exercice et le valide.

Un étudiant de l’Université Nancy-Charlemagne, Anthonio, souhaite alors faire l’exercice que cet enseignant à créer. Il va alors sur l’application et s’inscrit.
Après quoi, Anthonio va sur la liste des langages, choisi le langage en question, puis recherche un tag que l’exercice contient. Il lit alors la description et démarre l’exercice.
L’étudiant peut alors lire le sujet complet.
Il code ensuite son programme, après quoi il valide son code.
L’application informe alors Anthonio que son code comporte une erreur, il recommence, re-valide son code et l’application lui dis que son code remplit les conditions et lui donne le temps d'exécution de celui-ci. Anthonio se déconnecte de l’application.

Un administrateur de l’application, Jean-Jacques, décide de supprimer un exercice jugé non-conforme car enregistrer sans teste unitaire. Pour ce faire, Jean-Jacques se connecte avec son compte administrateur, se rend sur la liste des exercices et recherche celui souhaité. De là, il va sur la page d’administration de l’exercice et supprime l’exercice.