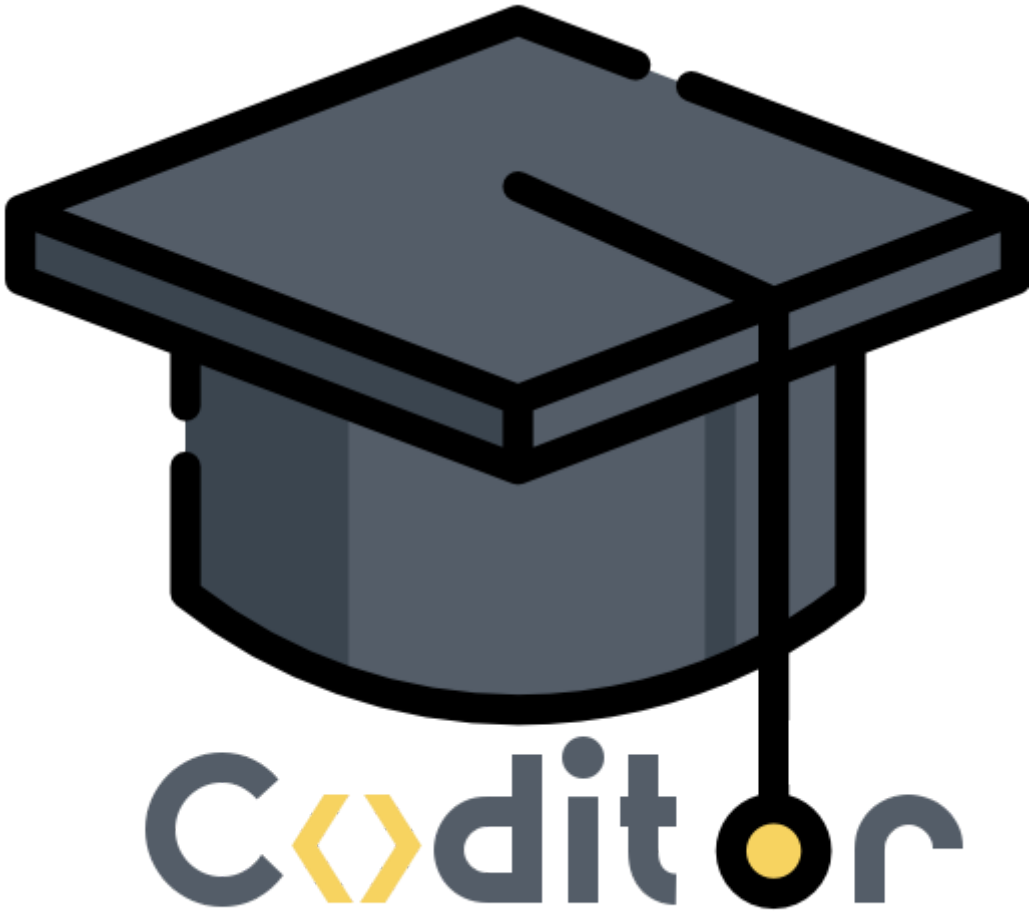
Analyse des besoins

Coditor

Tuteur : BOUMAZA Amine

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



Cordier - Ralli - Rohrbacher - Zink

Sommaire

Introduction du sujet **3**
- Les problématiques 3
- Notre réponse 3

Cas d'utilisation **4**

Périmètre du projet **5**

Etude de l'existant **8**

Solutions envisagées **9**
- Exécution du code 9
- Choix du langage 9
- Base de données 9

Acteurs **10**

Cas d'usages **11**

Exemple **12**

1. Introduction du sujet

a. Les problématiques

L'origine de notre sujet de projet tutoré résulte d'un problème soulevé par notre tuteur, M. Boumaza. En effet, en tant que professeur en charge des étudiants de DUT informatiques ainsi que des étudiants de CIASIE depuis quelques années, il a remarqué la chose suivante : ***les exercices proposés aux étudiants sont souvent identiques***.

De plus, il a également pour soucis de se ***détacher du système actuel de notation***. Seul le fait de savoir si l'étudiant a bien compris la notion abordée est important, tandis que la note classique sur 20 peut être perçue comme une sanction.

Cela est d'autant plus vrai pour les exercices d'initiation à un langage pour lesquels il devient redondant de recréer un cours et fournir à nouveau les ressources pour la résolution des exercices de base …

Il faudrait donc rendre ces exercices disponibles en permanence pour tous les étudiants.

b. Notre réponse

La finalité de notre projet tutoré est donc de proposer une application web mettant à disposition ce type d'exercices pour les étudiants.

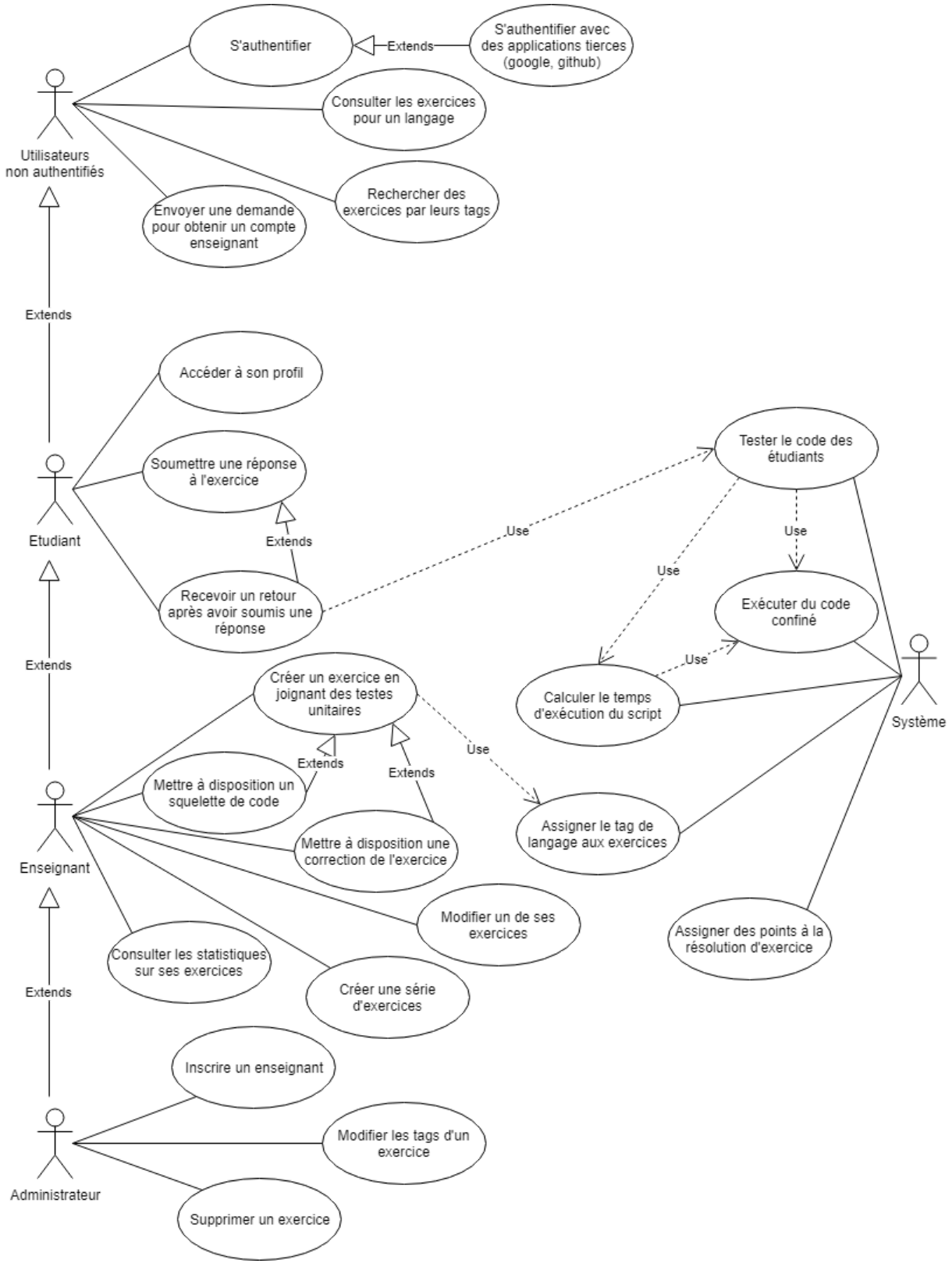
Ainsi, il suffirait aux étudiants de s'inscrire sur la plateforme pour y avoir accès.

Cette application permettra à des étudiants de réaliser des exercices que des enseignants vont proposés. Ces exercices permettront d'abords aux étudiants de s'entraîner et de situer leurs lacunes, de façon à s'améliorer en programmation.

Un système d'auto-évaluation est mis en place via un système de retour immédiat, basé sur des testes unitaires fournis par le professeur, après chaque exercice. Cela permettra à l'étudiant de savoir ce qu'il n'as pas réussi à faire ou ce qu'il devrait améliorer.

D'une façon générale, les étudiants se trouveront plus autonomes puisqu'ils auront un retour sur le code produit, ce qui permettra au professeur de disposer de plus de temps durant la séance.

2. Cas d'utilisation



3. Périmètre du projet

1. Le système exécute le code dans un environnement confiné

Dès que l'étudiant envoie son code, le serveur va l'exécuter dans un environnement sécurisé afin de se prémunir de scripts malfaisants

2. Le système test le code des étudiants

Grâce au tests unitaires, le serveur va pouvoir tester le code de l'étudiant afin de s'assurer que son code est conforme aux attentes du professeurs

3. Le système assigne le tag de langage à l'exercice créé

Lorsque le professeur crée un exercice dans un langage, le système attribue automatiquement le langage à l'exercice

4. Le système calcule le temps d'exécution du code envoyé par l'étudiant

Lorsqu'un étudiant envoie son code, le système vas l'exécuter et retourner le temps qu'il à mis à être exécuté.

5. Le système assigne des points à la résolution de l'exercice

A chaque exercice résolu, l'étudiant obtient 1 point supplémentaire pour le langage dans lequel il vient d'effectuer un exercice : le but étant de créer un esprit compétitif chez l'étudiant pour le pousser à aller plus loin.

6. Un utilisateur peut s'identifier/se déconnecter/s'inscrire

Dès lors qu'un utilisateur se rends sur l'application, il à la possibilité de s'inscrire. Après cette inscription, il pourra se connecter au site et se déconnecter quand il le souhaite.

7. Un utilisateur peut s'identifier avec des applications tierces (google, github)

Lors de l'inscription, un utilisateur peut choisir de se connecter à l'application via un compte d'une autre application, comme Google ou GitHub. Cela permet aux utilisateurs de se créer un compte et de se connecter facilement et rapidement.

8. Un utilisateur peut demander à obtenir un compte enseignant

Dès qu'un utilisateur est inscrit, il peut alors effectuer une demande pour devenir enseignant et pouvoir créer des exercices.

9. Un utilisateur peut consulter les exercices pour un langage

Lorsqu'un utilisateur se rends sur l'application, celui-ci à la possibilité de voir les exercices disponibles spécifiques à un langage.

10. Un utilisateur peut rechercher des exercices par leurs tags

Lorsqu'un utilisateur se rends sur l'application, celui-ci à la possibilité de voir les exercices disponibles selon un tag (ex: Boucles, Fonctions, …) que l'enseignant vas définir lors de la création d'un exercice.

11. Un étudiant peut accéder à son profil

A tout moment, un étudiant peut, lorsqu'il est connecté, accéder à son profil et ainsi accéder à ses informations (mail, mdp, …)

12. Un étudiant peut soumettre une réponse à un exercice choisi

Lorsqu'un étudiant se rends sur un exercice et le réalise, il à la possibilité de soumettre sa réponse si il pense que son code est correct.

13. Un étudiant peut voir la résolution d'un exercice lorsqu'il soumet son code

Lorsqu'un étudiant soumet sa réponse à un exercice, il peut alors voir la résolution de cet exercice que l'enseignant a mis.

14. Un enseignant peut créer un exercice en joignant des tests unitaires

Lorsqu'un enseignant créer un exercice, il peut ajouter ses propres tests unitaires afin que le système qui teste le code des étudiants réponde aux mieux à la problématique.

15. Un enseignant peut mettre à disposition un squelette de départ pour l'exercice

Lorsqu'un enseignant créer un exercice, il peut choisir de fournir un squelette qui permettra aux étudiants de démarrer l'exercice plus facilement.

16. Un enseignant peut mettre à disposition une correction de l'exercice

Lorsqu'un enseignant créer un exercice, il peut lui joindre la correction de cet exercice qu'il a lui même créer.

17. Un enseignant peut créer une série d'exercices

Si un enseignant le veux, il peut créer une suite d'exercices qui seront réalisés dans l'ordre qu'il définit.

18. Un enseignant peut consulter les statistiques de ses exercices

Quand un enseignant à créer un exercice et que des étudiants l'ont réalisés, il peut décider de voir les statistiques de celui-ci, comme le taux de réussite au premier essai, etc...

19. Un enseignant peut modifier un de ses exercices

Si il le souhaite, un enseignant peut modifier un exercice qu'il a créer, ou y ajouter des choses comme un squelette, des tests, etc...

20. Un administrateur peut accepter la demande de création de compte d'un enseignant

Lorsqu'un utilisateur fait la demande pour devenir enseignant, c'est à un administrateur de vérifier cette demande et de la valider ou non.

21. Un administrateur peut changer les tags d’un exercice

Si il le juge utile, un administrateur pourra changer les tags mis par un enseignant sur un exercice, afin d’en améliorer la visibilité dans l’application.

22. Un administrateur peut supprimer un exercice

Si il juge qu’un exercice n’est pas conforme, un administrateur pourra supprimer un exercice déjà fait.

4. 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.

5. Solutions envisagées

a. Exécution du code

Pour l'exécution du code dans un environnement confiné nous avons pensé à deux solutions afin de répondre à ce problème : Docker et AWS Lambda. Ces deux services répondent à nos attentent car ils nous permettent d'exécuter du code à l'extérieur de notre serveur.

En utilisant Docker, nous avons accès à une machine virtuelle afin de tester le code. Cet environnement peut être modifiable à souhait et donc avons le contrôle sur le traitement fait en interne.

Pour AWS Lambda, nous devons envoyer le script de l'étudiant sur un serveur pour qu'il soit traiter. L'inconvénient par rapport à Docker est que nous avons peu de contrôle du traitement qu'il fera de la fonction envoyée et que le service est payant par requêtes.

b. Choix du langage

Au niveau du langage de programmation côté serveur, nous avons le choix entre PHP et Javascript. Nous avons décidé de nous orienter vers Javascript et son framework NodeJs.

Premièrement, il nous paraît plus simple d'avoir un seul langage pour toute l'application que ce soit du côté client ou serveur. Ensuite, l'aspect asynchrone nous permet de gérer l'envoie de plusieurs requêtes vers le serveurs sans problèmes. Cela nous aidera lors de la résolution des exercices par les étudiants. Enfin NodeJs est hautement performant, configurable facilement et possède un gestionnaire de paquet (NPM) qui comprend des librairies dont nous auront l'utilité dans notre projet.

c. Base de données

Lors de la conception du projet, nous avions le choix entre utiliser Mysql ou MongoDB en tant que base de données. Étant donné que nous devrons certainement traité une grande quantité d'information, nous avons décidé de nous orienter vers MongoDB. De plus, MongoDB est un technologie montante ces dernières années et nous voulons la mettre en pratique au sein d'un projet concret.

6. 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.

7. Cas d’usages

Un enseignant en langage de programmation JavaScript, Léon, 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, Léon 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, Anthony, souhaite alors faire l’exercice que cet enseignant à créer. Il va alors sur l’application et s’inscrit.
Après quoi, Anthony va sur la liste des langages qui existent et choisi le langage JavaScript.
Plusieurs solution s’offrent à lui :

- Il peut rechercher un exercice par mot-clés
- Il peut consulter tous les exercices d’un langage précis
- Il peut éventuellement consulter les exercices les plus populaire d’un langage

Afin de retrouver l’exercice qu’il souhaite, il va écrire “Fonction”, qui est un tag décrivant l’exercice qu’il recherche. De là, il peut voir tous les exercices sur les fonctions et en sélectionne un. Il lis la description de cet exercice et décide de le commencer.
L’étudiant peut alors lire le sujet complet.
Il code ensuite son programme, après quoi il valide sa résolution.
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. Anthony 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.

8. Exemple

Voici un schéma qui montre le cheminement lors de la création d'un exercice par un professeur.