

Réf : PFA2-2023- 14

Rapport de Projet de Fin d'Année de

Deuxième année en Génie Informatique

Présenté et soutenu publiquement le 11/05/2023

Par

Rania HAMDANI

Conception et développement d'une application de programmation et de résolution de problèmes

Composition du jury

Monsieur Ramzi FARHAT

Madame Sonda CHTOUROU

Président

Encadrant

Année universitaire : 2022-2023

Remerciement

Nous tenons tout d'abord à remercier grandement notre enseignant encadrant Madame Sonda Chtourou, pour toutes les connaissances qu'elle m'a communiquées tout au long de l'élaboration de ce projet, pour sa grande disponibilité, et ses précieux conseils. pour son soutien moral, ses encouragements et sa présence qui nous ont été d'une très grande utilité.

Nous remercions toutes les personnes qui ont participé de près ou de loin à l'élaboration de ce projet et nous ont aidés en fournissant des conseils pratiques.

Finalement, nous remercions chaleureusement les membres du jury pour m'avoir honoré en acceptant d'évaluer ce travail.

Table des matières

Table des figures	4
Liste des tableaux	6
1 Cadre général du projet	3
1.1 Introduction	3
1.2 Contexte du projet	3
1.3 Étude et critique de l'existant	4
1.3.1 Solutions existantes	4
1.3.2 Tableau comparatif des solutions existantes	6
1.4 La solutions proposée	6
1.5 Méthodologie de développement	8
1.6 Conclusion	8
2 Analyse et spécification des besoins	10
2.1 Introduction	10
2.2 Spécification des besoins	10
2.2.1 Identification des besoins fonctionnels	10
2.2.2 Identification des besoins non fonctionnels	11
2.3 Analyse	12
2.3.1 Identification des acteurs	12
2.3.2 Diagramme des cas d'utilisation global	12
2.3.3 Raffinement des cas d'utilisation	13
2.4 Conclusion	19
3 Etude Conceptuelle	20
3.1 Introduction	20
3.2 Modèle architectural	20
3.3 Diagramme de classe	21
3.4 Diagrammes de séquences	22

3.4.1	Diagramme de séquence du cas d'utilisation « Commencer l'exercice »	22
3.4.2	Diagramme de séquence du cas d'utilisation « Commencer l'interview »	23
3.5	Conclusion	24
4	Mise en œuvre	25
4.1	Introduction	25
4.2	Environnement matériel	25
4.3	Environnement logiciel	25
4.4	Choix des technologies de développement	26
4.4.1	Technologies back-end	26
4.4.2	Technologies front-end	26
4.5	Interfaces de l'application	27
4.5.1	Authentification	27
4.5.2	Accueil	27
4.5.3	Exercices	28
4.5.4	Problèmes	30
4.5.5	Interviews	31
4.5.6	Dashboards	34
4.5.7	Calendrier	36
4.5.8	Kanban	37
4.5.9	Différents thèmes dans l'application	37
4.6	Conclusion	38

Table des figures

1.1	Interface de la plateforme Codeforces [1]	4
1.2	Interface de la page préparation de Hackerrank [2]	5
1.3	Interface de cours de Topcoder [3]	5
1.4	Interface de cours de CodeChef [4]	6
2.1	Diagramme des cas d'utilisation global	13
2.2	Le raffinement du cas d'utilisation « Consulter les exercices »	14
2.3	Le diagramme de séquence du cas d'utilisation « Consulter les exercices »	15
2.4	Le raffinement du cas d'utilisation « Consulter les Interviews »	17
2.5	Le diagramme de séquence du cas d'utilisation « Consulter les interviews »	17
2.6	Le raffinement du cas d'utilisation « Consulter les dashboards »	19
3.1	Les différents composants du modèle MVC [5]	21
3.2	Le diagramme de classe	22
3.3	Le diagramme de séquence du cas d'utilisation « Commencer l'exercice »	23
3.4	Le diagramme de séquence du cas d'utilisation « Commencer l'interview »	24
4.1	Interfaces d'authentification	27
4.2	Interface d'accueil	28
4.3	Interface des exercices	29
4.4	Interface de compilation	30
4.5	Interface des problèmes	31
4.6	Interface des interviews	32
4.7	Interface des questions	33
4.8	Interface de fin de l'interview	33
4.9	Le dashboard des exercices	34
4.10	Le dashboard des problèmes	35

4.11 Le dashboard des interviews	36
4.12 Le calendrier	36
4.13 Le kanban	37
4.14 Les différents thèmes dans l'application	37

Liste des tableaux

1.1	Tableau comparatif des solutions existantes	9
2.1	Description textuelle du cas d'utilisation «Commencer l'exercice» .	16
2.2	Description textuelle du cas d'utilisation «Commencer l'interview» .	18

Introduction Générale

Les informaticiens sont constamment sollicités à améliorer leurs compétences en programmation, en résolution de problèmes et à se préparer aux entretiens techniques. Ceci leur permet de rester compétitifs sur le marché du travail et en constante évolution.

Dans ce cadre, plusieurs applications de programmation et de résolution de problèmes sont apparues. Ces applications offrent plusieurs fonctionnalités dont principalement : des exercices de programmation, des problèmes à résoudre et des simulations d'interviews. Les exercices de programmation permettent d'améliorer les connaissances de base (tableaux, chaînes de caractères, ...) et le niveau de programmation. La résolution des problèmes (mathématiques, recherche dynamique, ...) permet aux utilisateurs de s'entraîner à résoudre des défis complexes et améliorer leur pensée critique. La simulation des interviews offre au candidat l'opportunité de s'entraîner sur les tests techniques. Ce qui lui permet de mieux réussir ses entretiens et d'identifier ses lacunes. La simulation des interviews permet aussi aux recruteurs d'évaluer les compétences techniques des candidats.

Dans ce contexte et dans le cadre de notre projet, nous proposons de concevoir et développer une application de programmation et de résolution de problèmes. Notre application propose de nombreux exercices de programmation en langage Java. Elle offre aussi des problèmes à résoudre selon le contexte (mathématiques, recherche dynamique, arbre binaire, ...). Notre application permet également d'effectuer des simulations d'interviews. De plus, elle offre des dashboards pour visualiser les scores obtenus (exercices, problèmes et interviews). Finalement, elle propose un calendrier pour planifier les exercices, les problèmes et les interviews à faire et un kanban pour enregistrer leurs états (à faire, en cours ou terminé).

Ce projet est détaillé dans ce présent rapport organisé en quatre chapitres :

Le premier chapitre, intitulé «Cadre général du projet», présente le contexte général du projet et une critique des solutions existantes ainsi que la solution proposée.

Le deuxième chapitre, «Analyse et spécification des besoins», est consacré à l'identification des besoins fonctionnels et non fonctionnels attendus de l'application et à l'analyse des différents cas d'utilisation.

Le troisième chapitre, «Étude conceptuelle» est dédié à la présentation des différents modèles conceptuels qui préparent le projet pour son implémentation.

Le quatrième chapitre, «Réalisation et mise en oeuvre», présente l'environnement et les outils de développement utilisés et illustre les interfaces les plus pertinentes de notre application.

Finalement, ce rapport est clôturé par une «Conclusion générale» pour récapituler le travail réalisé et donner quelques perspectives.

Chapitre 1

Cadre général du projet

1.1 Introduction

D’abord, nous présenterons le cadre général du projet. Ensuite, nous proposerons une étude et critique de l’existant. Finalement, nous détaillerons la solution proposée.

1.2 Contexte du projet

Les applications de programmation et de résolution de problèmes offrent aux informaticiens des exercices (tableaux, chaînes de caractères, ...) pour améliorer leurs compétences en programmation. Elles offrent aussi des problèmes diversifiés (mathématiques, recherche dynamique...) pour améliorer leur capacité à résoudre des problèmes. Elles présentent également des interviews proposés par des entreprises pour se préparer aux entretiens techniques d’embauches.

Ces applications aident les informaticiens (étudiants, ingénieurs, techniciens, ...) à devenir meilleurs en termes de programmation et de qualité de code ce qui leur permet de mieux réussir leur carrière et leurs entretiens. D’un autre côté, ces applications offrent une solution complète aux recruteurs pour mieux évaluer les candidats lors des entretiens et aux entreprises pour améliorer le niveau et compétences de ses ingénieurs.

Dans ce contexte, nous proposons de concevoir et développer une application de programmation et de résolution de problèmes.

1.3 Étude et critique de l'existant

Dans cette partie, nous présentons les solutions existantes de programmation et de résolution de problèmes. Puis, nous les critiquons et nous proposons notre solution.

1.3.1 Solutions existantes

Parmi les plateformes de programmation et de résolution de problèmes existantes, nous pouvons citer :

Codeforces [1] - Une plateforme de programmation compétitive qui propose des problèmes de programmation à résoudre. Il existe des défis hebdomadaires, des concours en ligne et une communauté de programmeurs. La page d'accueil de cette plateforme est présentée dans la figure 1.1.

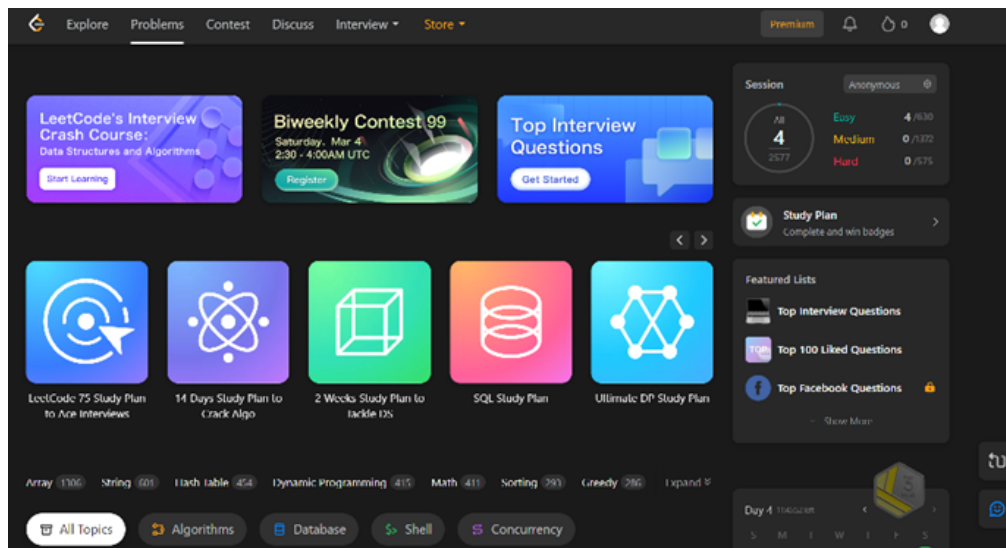


FIGURE 1.1 – Interface de la plateforme Codeforces [1]

HackerRank [2] - Cette plateforme propose des interviews et des problèmes de programmation de différentes difficultés. Il ya aussi des tutoriels et une documentation sur plusieurs sujets. La page de préparation de cette plateforme est présentée dans la figure 1.2.

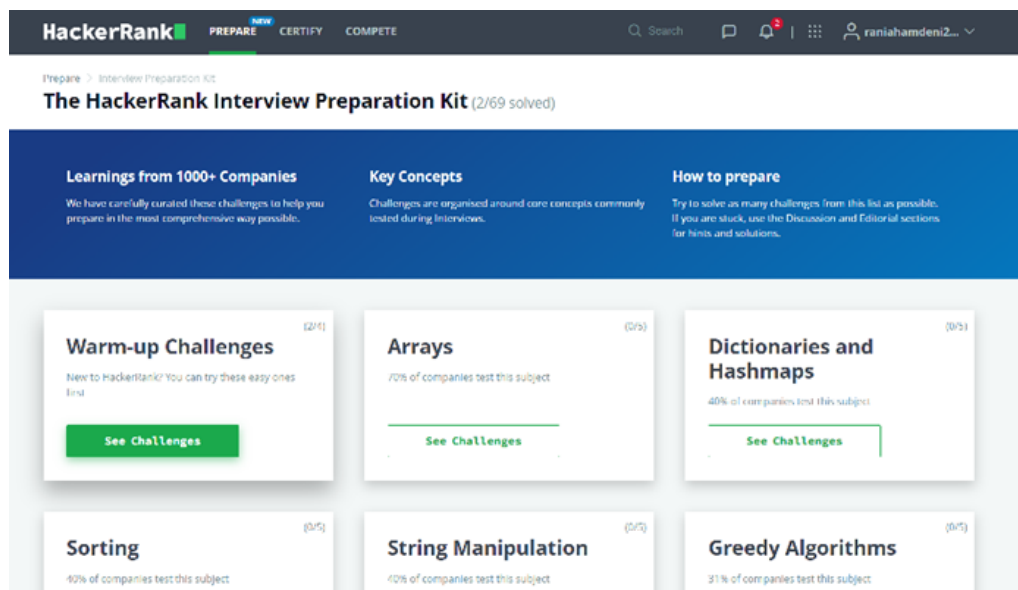


FIGURE 1.2 – Interface de la page préparation de Hackerrank [2]

TopCoder [3] - Une autre plateforme qui offre des problèmes de programmation. Elle présente aussi des défis de programmation et des tournois en ligne. La page des cours de cette plateforme est présentée dans la figure 1.3.

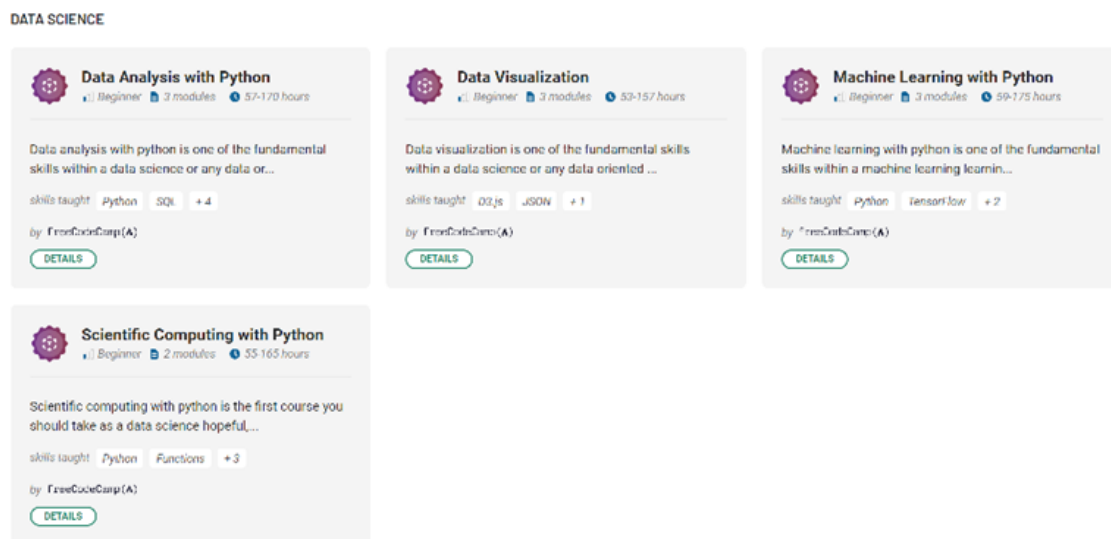
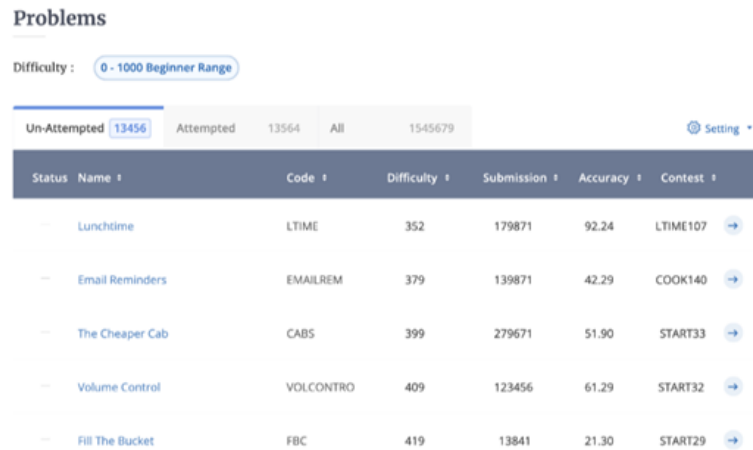


FIGURE 1.3 – Interface de cours de Topcoder [3]

CodeChef [4]- Cette plateforme propose des défis de programmation de différentes difficultés, ainsi que des concours et des défis pour les étudiants. La figure

1.4 présente l'interface des problèmes proposés par codechef.



The screenshot shows the 'Problems' page on CodeChef. At the top, there's a 'Difficulty' filter set to '0 - 1000 Beginner Range'. Below that, there are tabs for 'Un-Attempted' (13456), 'Attempted' (13564), and 'All' (1545679). A 'Setting' icon is visible on the right. The main table lists problems with the following columns: Status, Name, Code, Difficulty, Submission, Accuracy, and Contest. The first five rows of the table are as follows:

Status	Name	Code	Difficulty	Submission	Accuracy	Contest
---	Lunchtime	LTIME	352	179871	92.24	LTIME107
---	Email Reminders	EMAILREM	379	139871	42.29	COOK140
---	The Cheaper Cab	CABS	399	279671	51.90	START33
---	Volume Control	VOLCONTRO	409	123456	61.29	START32
---	Fill The Bucket	FBC	419	13841	21.30	START29

FIGURE 1.4 – Interface de cours de CodeChef [4]

1.3.2 Tableau comparatif des solutions existantes

Après avoir étudié l'existant, nous avons déterminé des critères à prendre en compte lors de la comparaison et de l'évaluation des plateformes. Ces critères ont été consignés dans le tableau T1.1.

Cette analyse nous a également permis de constater que chaque plateforme de résolution de problèmes possède ses propres caractéristiques et fonctionnalités uniques. LeetCode et HackerRank sont tous deux populaires pour leurs compétitions hebdomadaires et leurs défis de programmation, ainsi que pour leur communauté active. TopCoder se concentre davantage sur des problèmes de niveau intermédiaire à expert, tandis que CodeChef est une excellente plateforme pour les débutants qui souhaitent améliorer leurs compétences en programmation.

1.4 La solutions proposée

Nous proposons de développer une application de programmation et de résolution de problème qui contient la majorité des fonctionnalités trouvées dans les plateformes existantes et quelques nouvelles autres fonctionnalités.

Nous nous sommes inspirés des solutions existantes pour réaliser les fonctionnalités suivantes :

- Effectuer des exercices de programmation : l'utilisateur peut filtrer les exercices selon leurs niveaux de difficultés (facile, moyenne et difficile) et selon les catégories (chaînes de caractères, tableaux, polymorphisme, ...). Il lit la description, programme sa solution, la compile et consulte les erreurs de compilation.
- Effectuer des problèmes de programmation : l'utilisateur peut filtrer les problèmes selon leurs niveaux de difficultés (facile, moyenne et difficile) et selon les "tags" qui représente le contexte du problème (mathématiques, recherche dynamique, arbre binaire, ...). De même, il lit la description, programme sa solution, la compile et consulte les erreurs de compilation.
- Effectuer des interviews : l'utilisateur choisie un interview à effectuer en filtrant selon son rôle (développeur java, ingénieur logiciel, ingénieur en base de donnée, développeur javascript, ...), selon son expérience (s'il est senior ou junior) et selon l'entreprise qui a proposé cet interview (microsoft, google, ...).
- Niveaux de difficulté personnalisés : pour que les utilisateurs puissent choisir des problèmes adaptés à leur niveau de compétence.

Afin d'enrichir l'application, nous proposons de réaliser des nouvelles fonctionnalités :

- Dashboard : Offrir une visualisation personnalisé aux utilisateurs pour afficher les exercices et les problèmes résolues et les scores des interviews effectués avec leurs résultats selon ce score.
- Intégrer un chronomètre avec les interviews : Le chronomètre permet de donner une indication claire du temps qui reste à l'utilisateur pour terminer l'interview technique.
- La saisie semi-automatique du code : La saisie semi-automatique permet d'ajouter du code autour d'une expression que l'utilisateur vient de saisir se qui favorise la rapidité de la programmation.
- La présence d'un calendrier et d'un kanban : Le calendrier permet de planifier les exercices, les problèmes et les interviews que l'utilisateur compte faire et le kanban pour enregistrer leurs états (à faire, en cours ou terminé).
- Personnalisation des thèmes : L'utilisateur peut choisir un couleur parmi six pour personnaliser le thème de l'application et choisir le mode clair ou sombre

En ajoutant ces fonctionnalités, la plateforme pourrait aider les programmeurs à améliorer leurs compétences en résolution de problèmes de programmation tout en offrant une expérience utilisateur plus personnalisée et diversifiée.

1.5 Méthodologie de développement

Nous allons adapter la méthodologie Cycle en V. Cette méthodologie divise le développement de logiciels en phases distinctes, avec chaque phase de développement liée à une phase de test correspondante. Cette approche vise à créer des logiciels fiables et de haute qualité en détectant et corrigeant les erreurs dès les premiers stades du processus de développement.

1.6 Conclusion

Au cours de ce chapitre, nous avons exploré les plateformes de programmation et de résolution de problèmes. Nous avons fait également une étude comparatifs des solutions existantes et nous avons examiné les limites de ces plateformes. Finalement, nous avons détaillé notre solution. Le chapitre suivant sera consacré à l'analyse et spécification des besoins.

Fonctionnalités	LeetCode	HackerRank	TopCoder	CodeChef
Types de problèmes proposés	Algorithmes, bases de données, systèmes, etc.	Algorithmes, structures de données, IA, etc.	Algorithmes, mathématiques, conception de systèmes, etc.	Algorithmes, structures de données, mathématiques, etc.
Niveau de difficulté	Débutant à Expert	Débutant à Expert	Intermédiaire à Expert	Débutant à Expert
Présence d'un compilateur en temps réel	Oui	Oui	Oui	Oui
Fonctionnalités de saisie semi-automatique de code	Oui, disponible avec un abonnement premium	Oui, disponible avec un abonnement premium	Non	Oui, disponible avec un abonnement premium
Fonctionnalités des interviews pratiques	Non	Oui, disponible avec un abonnement premium	Non	Oui, disponible avec un abonnement premium
Présence d'un chronomètre pour se mettre en situation d'interview	Non	Non	Non	Non
Présence d'un Dashboard de visualisation	Oui	Non	Non,	Non
Présence d'un calendrier et d'un Kanban pour organiser les exercices	Non	Non	Non	Non
Présence des thèmes personnalisés	Non	Oui	Non	Oui

TABLE 1.1 – Tableau comparatif des solutions existantes

Chapitre 2

Analyse et spécification des besoins

2.1 Introduction

Dans ce chapitre, nous spécifierons les besoins fonctionnels et non fonctionnels, identifier les acteurs. Puis, nous détaillerons les raffinements de plusieurs cas d'utilisation.

2.2 Spécification des besoins

La spécification des besoins est la première étape dans le processus de développement de toute application. Cette phase est cruciale car elle permet une compréhension approfondie du projet, notamment l'identification des acteurs et la définition des fonctionnalités attendues.

2.2.1 Identification des besoins fonctionnels

Les besoins fonctionnels sont les fonctionnalités nécessaires à l'exécution de notre application. Elles incluent les actions que l'application doit effectuer, telles que :

Effectuer des exercices : L'utilisateur peut filtrer et choisir un exercice selon sa catégorie (chaînes de caractères, tableaux, récursivité, fonctions, ...) , sa description et son niveau de difficulté. Il lit l'énoncé, comprend l'exemple donnée, et écrit le code de la solution avec le langage de programmation Java. Une fois terminée, il soumet son travail pour la compilation et la vérification des erreurs de compilation. L'administrateur peut ajouter, supprimer ou modifier l'exercice et son

niveau de difficulté.

Résoudre des problèmes : L'utilisateur dispose de plusieurs options de filtrage pour les problèmes disponibles sur l'application. Il peut les trier selon leur niveau de difficulté, qui est classé en facile, moyen et difficile. De plus, il peut choisir les tags appropriés pour chaque problème, tels que les mathématiques, la recherche dynamique, ou les arbres binaires, pour s'exercer sur des problèmes qui correspondent à ses compétences et ses intérêts. En outre, il est possible de programmer une solution à un problème donné, de la compiler et de vérifier les erreurs de compilation.

Effectuer des Interviews : L'utilisateur peut choisir une simulation d'interview selon les exemples donnés par les grandes entreprises auparavant (microsoft, google, ...) et selon son rôle (développeur java, ingénieur machine learning, ...). Il peut filtrer selon l'entreprise, le rôle et le niveau de difficulté. Il commence l'interview et répond à des questions à choix multiples. L'utilisateur est contrôlé par un chronomètre dont la durée est déterminée selon l'interview. L'administrateur peut modifier tous les paramètres de l'interview, le supprimer ou en ajouter.

Consulter Dashboard : Après avoir effectué un ou plusieurs exercices, problèmes et/ou interview, l'utilisateur peut consulter un dashboard qui résume son avancement. Le Dashboard contient les exercices et les problèmes effectués correctement et son score pour les interviews qu'il a fait avec une allure générale de son niveau d'avancement.

Utiliser un calendrier et un kanban : Le calendrier et le kanban aident l'utilisateur à organiser ses exercices, problèmes et interviews en cours, à faire ou terminés.

Personnaliser le thème de l'application : l'utilisateur a la possibilité de personnaliser le thème de l'application en choisissant une couleur parmi six et en sélectionnant le mode clair ou sombre.

2.2.2 Identification des besoins non fonctionnels

Ce sont les exigences qui ne concernent pas spécifiquement le comportement du système mais qui assurent sa bonne qualité. Les principaux besoins non fonctionnels de notre plateforme sont :

- L'expérience utilisateur : cela consiste à concevoir la plateforme de manière à ce qu'elle soit facile à utiliser et à naviguer, en fournissant des éléments bien

organisés et lisibles, ainsi qu'un choix judicieux de couleurs.

- La maintenabilité du code : le code de la plateforme doit être bien écrit et compréhensible, ce qui facilite les tâches de maintenance.

2.3 Analyse

En analysant notre application, nous identifions les différents acteurs qui interagissent avec elle et créons une représentation fonctionnelle des acteurs et des fonctionnalités du système. Cette représentation est généralement présentée sous la forme d'un diagramme de cas d'utilisation.

2.3.1 Identification des acteurs

Dans ce projet, nous identifions deux acteurs principaux qui représentent des entités externes interagissant avec le système et un acteur secondaire

Les acteurs primaires :

- Administrateur : C'est la personne responsable de l'administration de l'application, il gère tous ses ressources.
- Utilisateur : Une personne qui possède un compte dans notre application.

L'acteur secondaire :

- JavaCompiler[14] : C'est une API qui sert à compiler un code java à partir d'une chaîne, il est nécessaire pour la compilation du code dans les exercices et les problèmes.

2.3.2 Diagramme des cas d'utilisation global

Le diagramme de cas d'utilisation est une représentation des fonctionnalités, en lien avec les acteurs impliqués. Le diagramme global de notre application illustré par la figure [2.1](#) permet d'avoir une vue d'ensemble du fonctionnement de l'application.

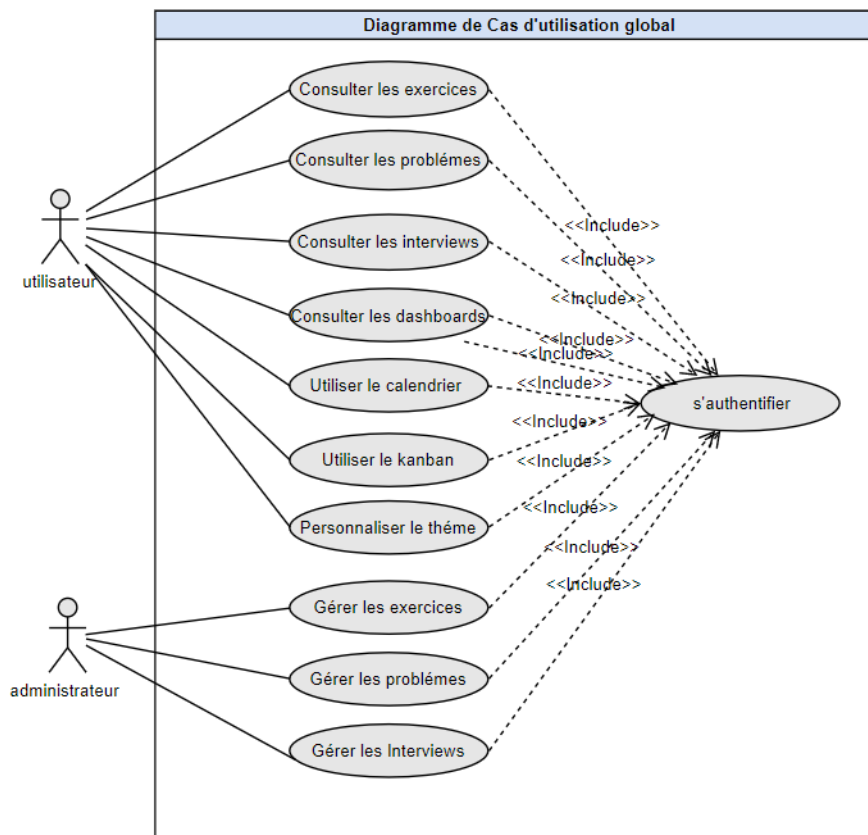


FIGURE 2.1 – Diagramme des cas d'utilisation global

2.3.3 Raffinement des cas d'utilisation

Nous présentons ici des cas d'utilisation sélectionnés pour une analyse plus approfondie à travers des diagrammes de cas d'utilisation détaillés, des descriptions textuelles et des diagrammes de séquence système d'analyse pour le scénario nominal.

2.3.3.1 Raffinement du cas d'utilisation « Consulter les exercices »

– Diagramme du cas d'utilisation « Consulter les exercices »

La figure 2.2 représente le cas d'utilisation « Consulter les exercices ». En choisissant cette fonctionnalité, l'utilisateur peut filtrer selon les différents attributs de l'exercice et en choisir un à effectuer.

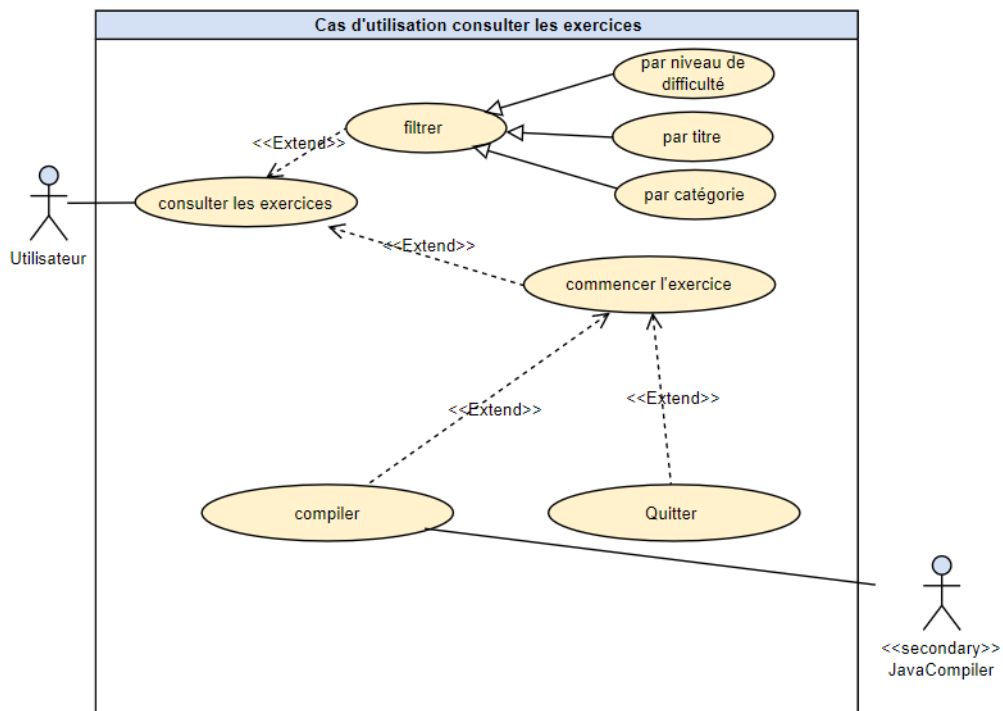


FIGURE 2.2 – Le raffinement du cas d'utilisation « Consulter les exercices »

La figure 2.3 présente le diagramme de séquence du cas d'utilisation « Consulter les exercices ».

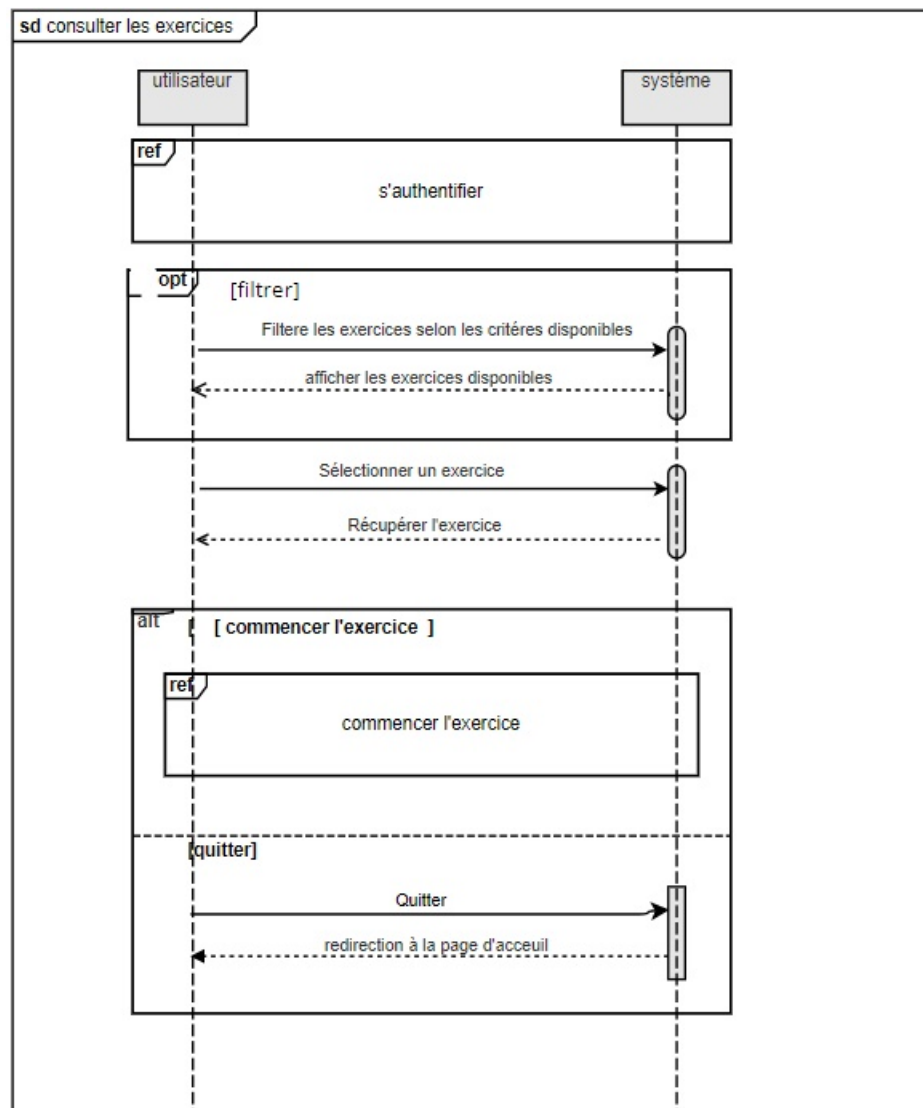


FIGURE 2.3 – Le diagramme de séquence du cas d'utilisation « Consulter les exercices »

– Description textuelle du cas d'utilisation «Commencer l'exercice»

Le tableau 2.1 présente la description textuelle du cas d'utilisation «Commencer l'exercice».

Titre	Commencer l'exercice
Acteurs	Utilisateur, JavaCompiler
Préconditions	Utilisateur authentifié et l'exercice est sélectionné
Postconditions	La solution de l'utilisateur est compilé avec succès et le score est enregistré dans la base de données
Scénario nominal	<p>1.Le système affiche l'interface contenant l'énoncé de l'exercice et un espace de programmation.</p> <p>2.L'utilisateur saisit son code en java puis clique sur le bouton compiler</p> <p>3.Le système envoie le code à l'API JavaCompiler pour compiler le code Java.</p>
Scénario d'erreurs	<p>E1 : Dans l'étape 2 du scénario nominal, L'utilisateur quitte l'exercice et retourne à la page d'accueil.</p> <p>E2 : Dans l'étape 4 du scénario nominal, si le code n'est pas correcte, le système affiche l'erreur de compilation.</p>

TABLE 2.1 – Description textuelle du cas d'utilisation «Commencer l'exercice»

2.3.3.2 Raffinement de cas d'utilisation «Consulter les interviews»

– Diagramme du cas d'utilisation «Consulter les interviews»

La figure [2.4](#) représente le cas d'utilisation «Consulter les interviews».

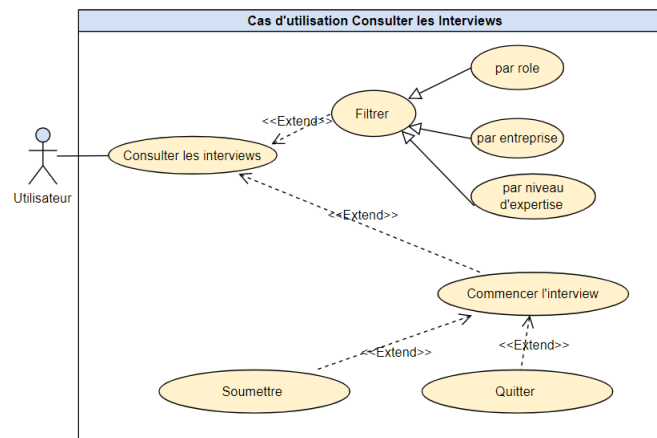


FIGURE 2.4 – Le raffinement du cas d'utilisation «Consulter les Interviews»

La figure 2.5 représente le diagramme de séquence du cas d'utilisation «Consulter les Interviews».

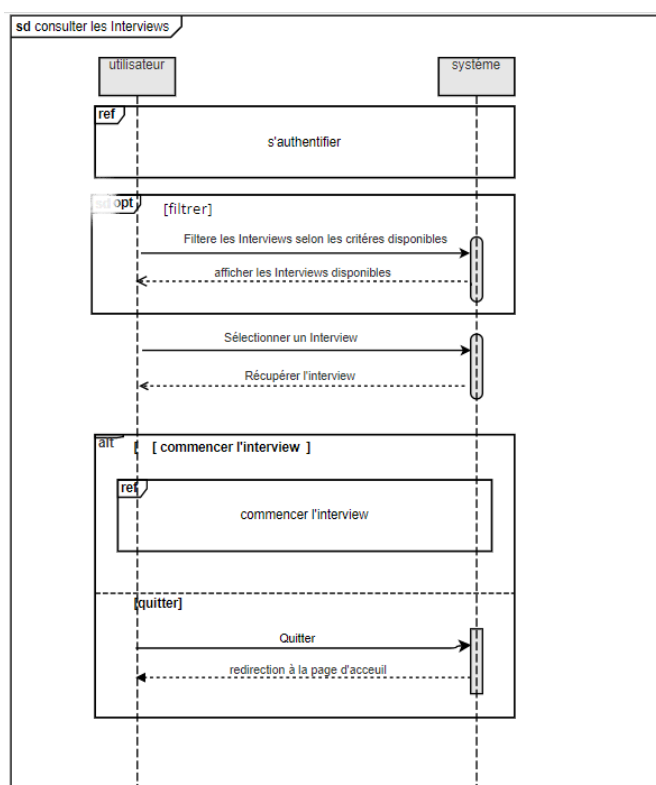


FIGURE 2.5 – Le diagramme de séquence du cas d'utilisation «Consulter les interviews»

2.3.3.3 Raffinement de cas d'utilisation «Commencer les Interviews»

- **Description textuelle du cas d'utilisation «Commencer les Interviews»**

Le tableau 2.4 présente la description textuelle du cas d'utilisation «Commencer l'interview».

Titre	Commencer l'interview
Acteur	Utilisateur
Préconditions	Utilisateur authentifié et l'interview est sélectionné
Postconditions	Interview est effectué et le score est affiché
Scénario nominal	<ol style="list-style-type: none">1. Le système affiche les questions et démarre le chronomètre.2. Pour toutes les questions, l'utilisateur répond aux questions et soumit ses réponses à chaque fois.3. L'utilisateur achève l'interview avant que la durée prédéfinie soit dépassée4. Le système calcule et affiche le score de l'interview.
Scénario d'erreur	Dans l'étape 2 du scénario nominale, l'utilisateur quitte l'interview et retourne à la page d'accueil.

TABLE 2.2 – Description textuelle du cas d'utilisation «Commencer l'interview»

2.3.3.4 Raffinement de cas d'utilisation «Consulter les Dashboards»

- **Diagramme du cas d'utilisation «Consulter les dashboards»**

La figure 2.6 représente le cas d'utilisation «Consulter les dashboards». L'utilisateur peut ainsi visualiser les scores des exercices et des interviews qu'il a effectué.

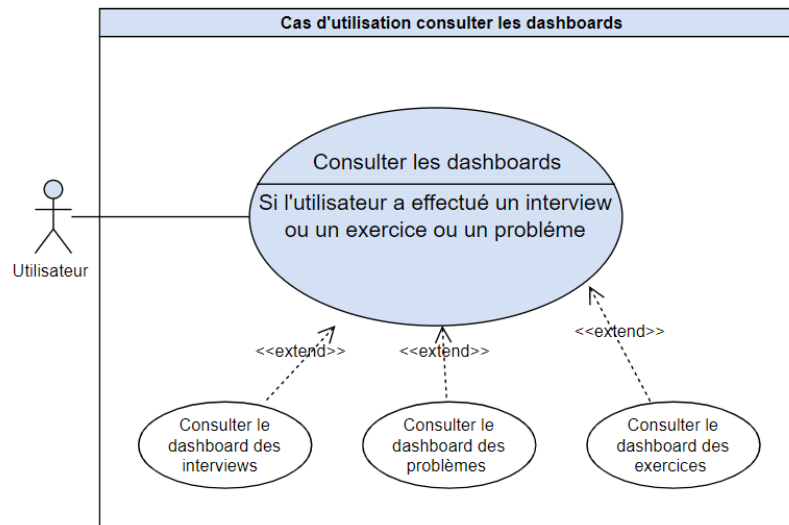


FIGURE 2.6 – Le raffinement du cas d'utilisation « Consulter les dashboards »

2.4 Conclusion

Le précédent chapitre a abordé l'analyse et spécification des besoins du projet, permettant de clarifier la problématique et d'identifier les fonctionnalités à mettre en œuvre. Le prochain chapitre se concentrera sur l'étude conceptuelle du projet.

Chapitre 3

Etude Conceptuelle

3.1 Introduction

Dans ce chapitre, nous exposerons notre étude conceptuelle. Nous présenterons le modèle architectural, puis nous détaillerons le diagramme de classe ainsi que les diagrammes de séquences.

3.2 Modèle architectural

Les logiciels de nos jours sont devenus de plus en plus complexes, ce qui a entraîné des codes tout aussi complexes et difficiles à comprendre et à maintenir. Afin de remédier à cela, il est devenu nécessaire d'utiliser une architecture qui permettrait d'organiser le code et de répartir les responsabilités entre plusieurs couches pour augmenter la cohésion et réduire le couplage. Les patrons d'architecture sont ainsi utilisés pour séparer la logique métier de la présentation d'un logiciel afin de garantir l'indépendance entre les différents composants du système. Cette approche permet une meilleure réutilisation du code, une meilleure maintenabilité et une meilleure résistance aux changements. Pour notre projet, nous avons choisi d'utiliser l'architecture Model View Controller (MVC) [5].

Le MVC un modèle de conception de logiciel conçu pour augmenter la lisibilité et la maintenabilité du code source, en répartissant le code dans les trois sections.

Modèle : Le modèle contient les données et la logique de l'application. Il est responsable de la gestion de l'état des données de l'application. Les interactions avec la base de données et les autres sources de données sont également gérées par le modèle.

Vue : La vue est responsable de la présentation des données à l'utilisateur. Elle affiche les données stockées dans le modèle et permet à l'utilisateur d'interagir avec l'application.

Contrôleur : Le contrôleur est responsable de la gestion des interactions entre l'utilisateur et l'application. Il reçoit les entrées de l'utilisateur, les traite et les transmet au modèle pour effectuer les opérations nécessaires. Il est également responsable de mettre à jour la vue en fonction des actions de l'utilisateur.

La figure 3.1 représente les différents composants du modèle MVC.

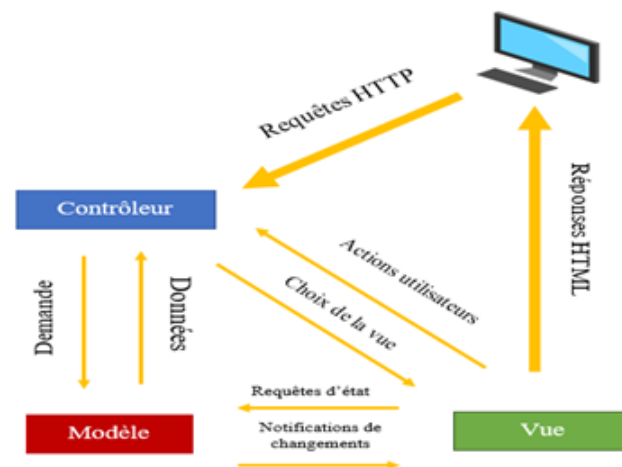


FIGURE 3.1 – Les différents composants du modèle MVC [5]

3.3 Diagramme de classe

Le diagramme de classe est essentiel en modélisation orientée objet. Il représente la structure statique du système en termes de classes et de relations entre classes.

La figure 3.2 représente le diagramme de classe de notre application. Chaque utilisateur est lié avec un exercice, un problème ou un interview par la classe score qui représente le score qu'il a obtenu après les avoir effectués. De même, chaque interview contient une liste de questions en QCM selon le rôle qu'il a choisi.

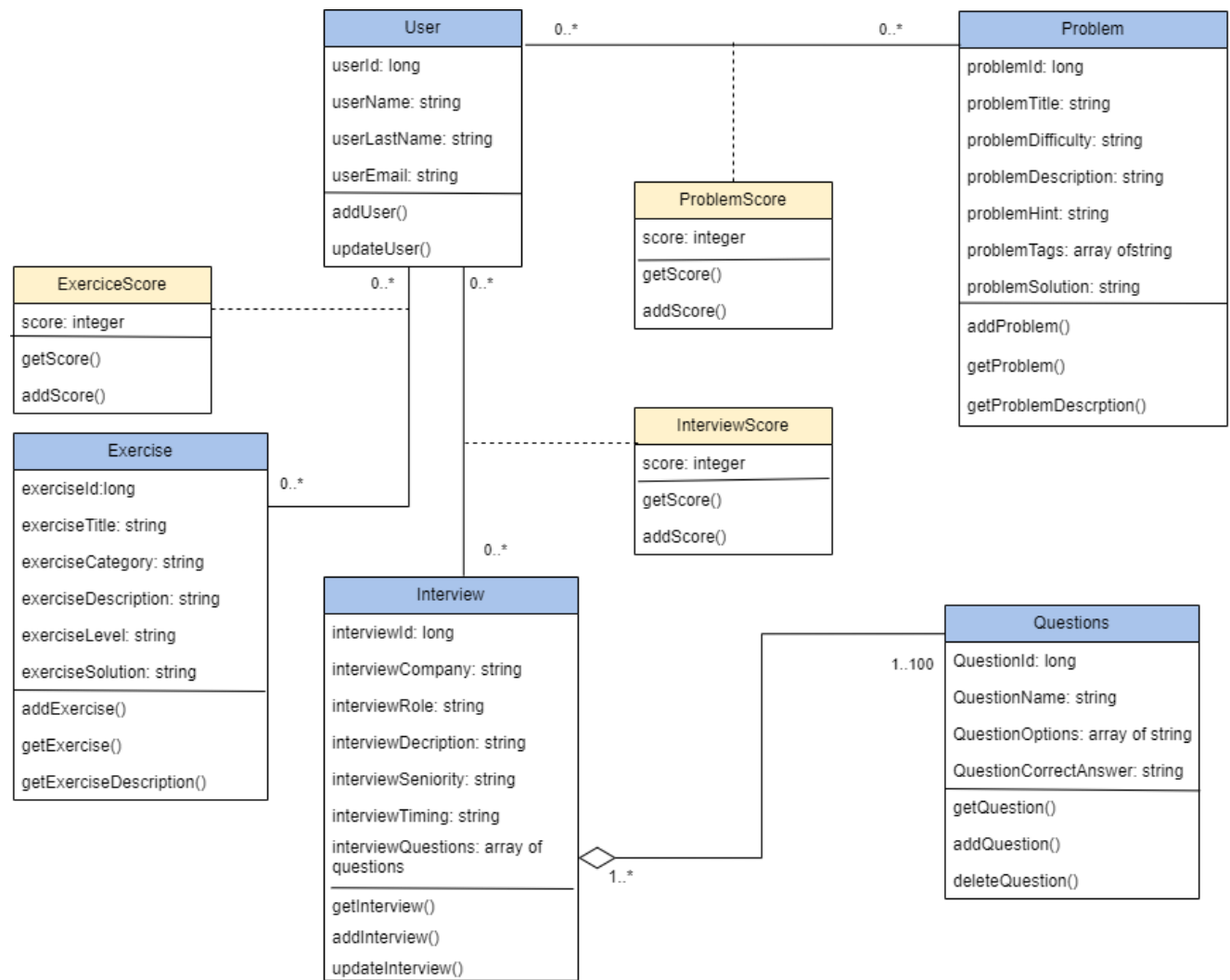


FIGURE 3.2 – Le diagramme de classe

3.4 Diagrammes de séquences

Les diagrammes de séquence sont utilisés pour modéliser les interactions entre les différents objets et les messages qu'ils échangent pour accomplir leur tâche.

3.4.1 Diagramme de séquence du cas d'utilisation « Commencer l'exercice »

Nous présentons par la figure 3.4 un diagramme de séquences détaillé du cas d'utilisation « Commencer l'exercice ».

Il s'agit de l'écriture de la solution et la vérifier avec le compilateur.

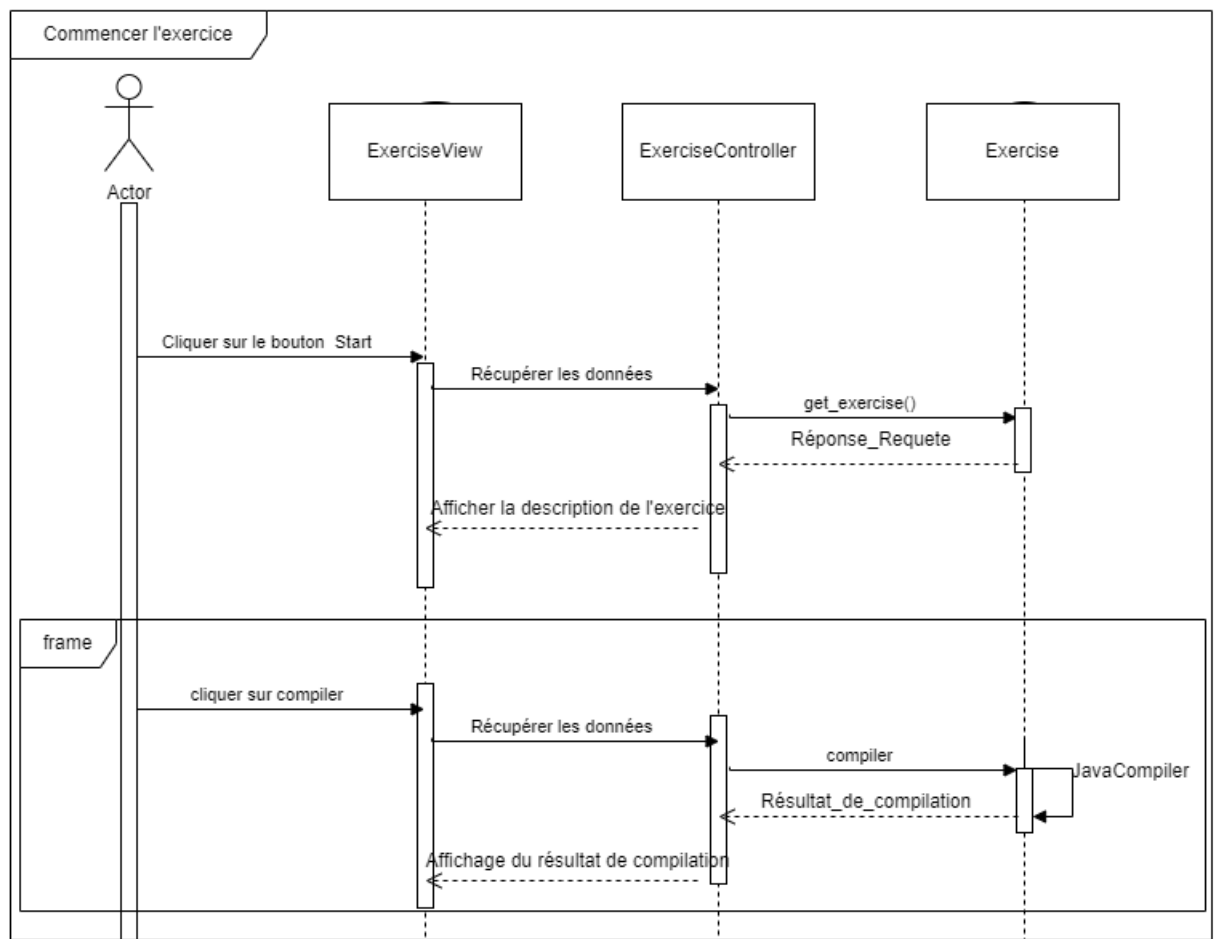


FIGURE 3.3 – Le diagramme de séquence du cas d'utilisation « Commencer l'exercice »

3.4.2 Diagramme de séquence du cas d'utilisation « Commencer l'interview »

Nous présentons par la figure 3.4 un diagramme de séquences du cas d'utilisation « Commencer l'interview ».

L'utilisateur répond aux questions le l'interview.

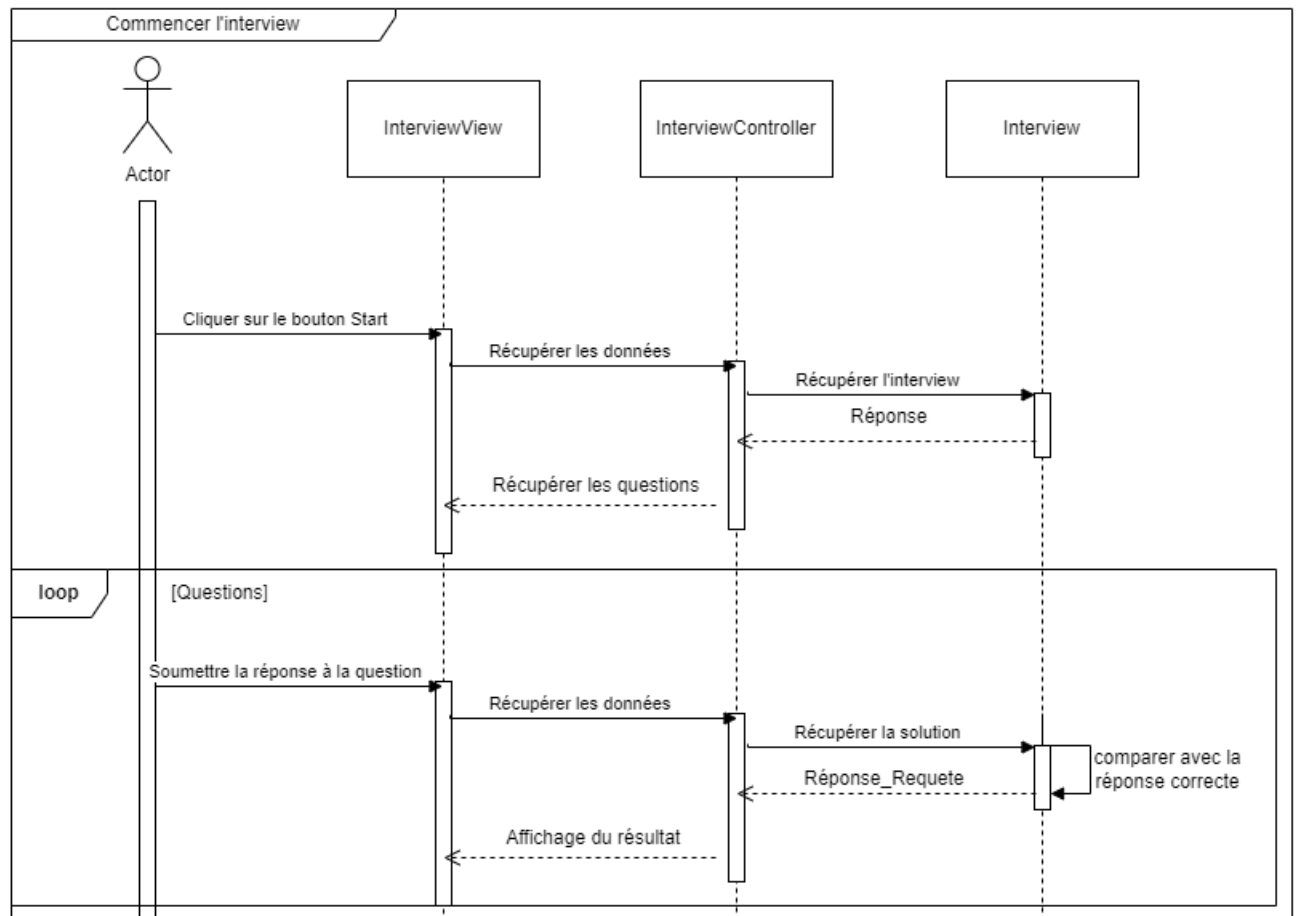


FIGURE 3.4 – Le diagramme de séquence du cas d'utilisation « Commencer l'interview »

3.5 Conclusion

Dans ce chapitre, nous avons détaillé l'étude conceptuelle du projet, qui représente une phase cruciale pour le fonctionnement de l'application. Dans le chapitre suivant, nous présenterons la mise en œuvre du projet.

Chapitre 4

Mise en œuvre

4.1 Introduction

Dans ce chapitre, nous détaillerons la phase de réalisation du projet. Nous présenterons les technologies que nous avons choisies pour implémenter notre application et nous exposerons les interfaces les plus pertinentes de notre projet.

4.2 Environnement matériel

Le projet est réalisé sur une machine possédant les caractéristiques décrites ci-dessous :

Modèle : HP

Processeur : Intel Core i7 CPU

Mémoire : 20 Gb «RAM»/500 Gb «SSD»

Système d'exploitation : Windows 10

4.3 Environnement logiciel

Cette section présente les différents outils logiciels que nous avons exploités pour réaliser notre application.

Visual Studio 2022 : C'est un éditeur de code optimisé pour la création et le débogage des applications web et Cloud modernes. Nous avons utilisé cet éditeur pour créer nos interfaces Web [6].

SQL server : C'est un serveur de base de données. Il est disponible sur la plupart des systèmes d'exploitation [7].

Postman : un outil qui permet de construire et de tester rapidement des requêtes HTTP [8].

IntelliJ IDEA : c'est un IDE pour le développement en Java. Il préserve la productivité grâce à une suite de fonctionnalités renforçant votre efficacité, avec notamment une assistance intelligente au codage, des refactorisations fiables, une navigation instantanée dans le code, des outils de développement intégrés, la prise en charge du développement web et d'entreprise, et bien plus encore[9].

4.4 Choix des technologies de développement

4.4.1 Technologies back-end

Java : Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy, présenté officiellement le 23 mai 1995 au SunWorld[10].

Spring boot : est un Framework libre et open source pour le développement des applications d'entreprise. Il est fondé pour structurer, améliorer et simplifier la configuration des applications grâce à l'injection de dépendances[11].

4.4.2 Technologies front-end

TypeScript : C'est un langage programmation open source qui s'appuient sur JavaScript. Il est développé par Microsoft en 2012[12].

ReactJS : React JS est une technologie open source conçue pour faciliter la création d'interfaces utilisateur pour les applications web. Elle permet de développer des applications web interactives et réactives en créant un DOM virtuel[13].

4.5 Interfaces de l'application

Nous présentons dans cette section quelques interfaces développées pour notre application de programmation et de résolution de problème.

4.5.1 Authentication

La Figure 4.1.a illustre l'interface d'authentification pour accéder à l'application. Cela est nécessaire pour garantir la confidentialité des données de l'utilisateur. La Figure 4.1.b illustre l'interface de création de compte.

The figure contains two side-by-side screenshots of web interfaces. The left screenshot, labeled (a), is titled 'Login' and features an 'Email' input field with the text 'rania.h.hamdani@gmail.c', a 'Password' input field with masked characters, a blue 'Login' button, and a link 'Don't have an account?Signup'. The right screenshot, labeled (b), is titled 'Signup' and features an 'Email' input field with the text 'rania.h.hamdani@gmail.c', a 'Password' input field with masked characters, a 'Confirm Password' input field with masked characters, a blue 'Signup' button, and a link 'Already have an account?Login'.

(a) Interface de Login

(b) Interface de SignUp

FIGURE 4.1 – Interfaces d'authentification

4.5.2 Accueil

La figure 4.2 expose la première interface à afficher lors de l'accès l'application. C'est la page d'accueil qui présente notre application pour introduire les grandes fonctionnalités : faire des exercices, résoudre des problèmes et effectuer des interviews.

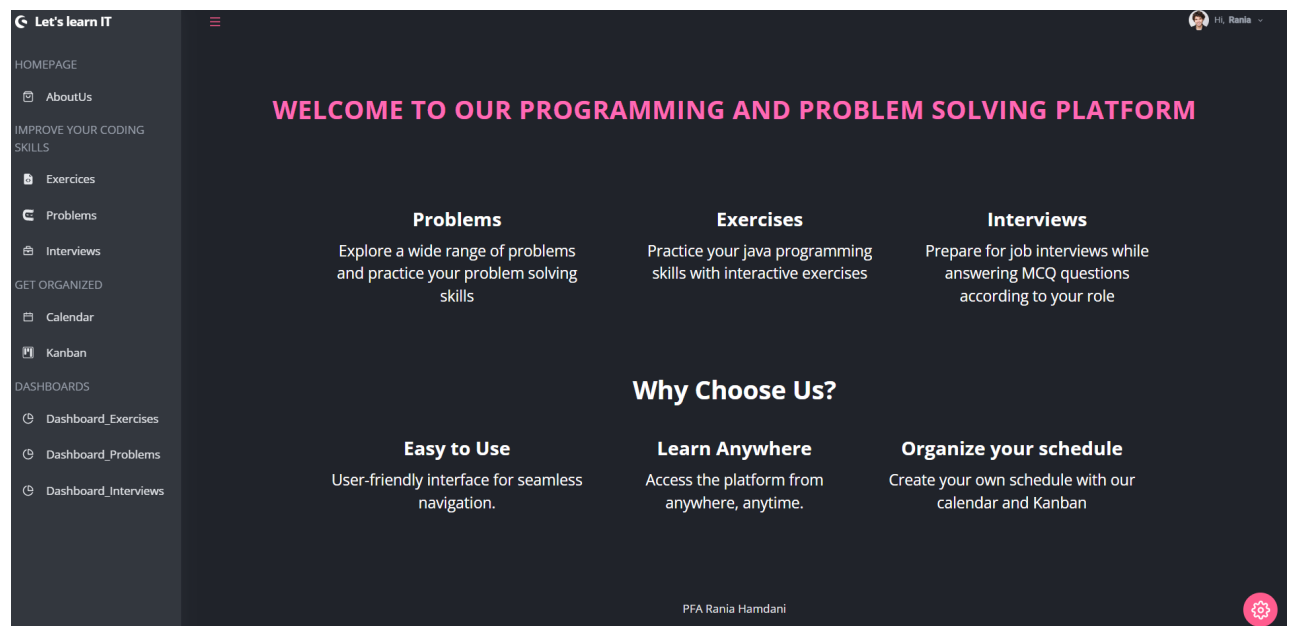


FIGURE 4.2 – Interface d'accueil

4.5.3 Exercices

La figure 4.2 l'interface des exercices présentant les différents paramètres par lesquelles l'utilisateur peut filtrer par titre, catégorie et par niveau. Une fois l'exercice est choisi, l'utilisateur clique sur le bouton "Start" pour commence l'exercice.

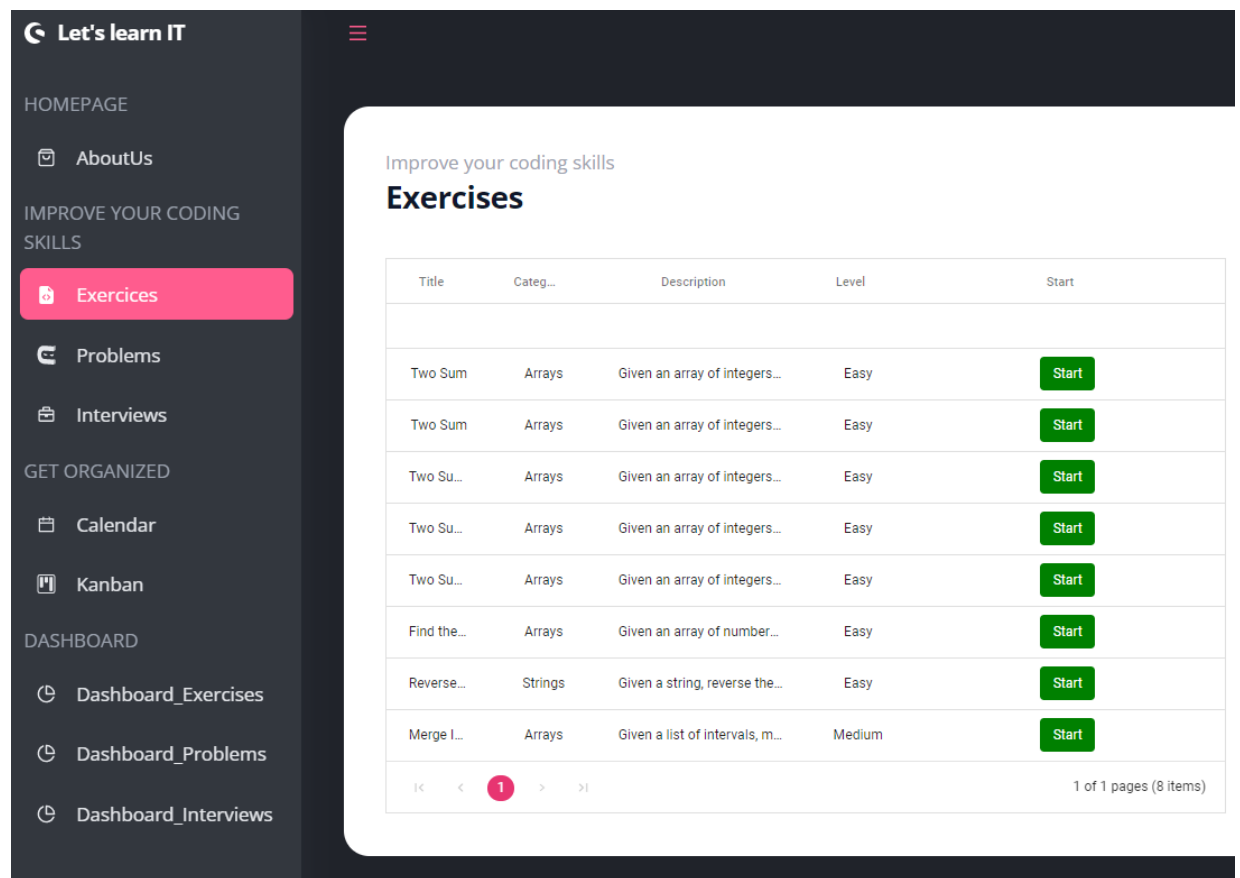


FIGURE 4.3 – Interface des exercices

Le système affiche l'interface contenant l'énoncé de l'exercice et un espace de programmation comme illustrée par la Figure 4.4. L'utilisateur saisit son code en java.

Puis, il clique sur le bouton "Compile". Le système envoie le code à l'API JavaCompiler pour compiler le code Java. Si le code n'est pas correcte, le système affiche l'erreur de compilation (voir Figure 4.4). Sinon, le contrôleur enregistre le score dans la base de données.

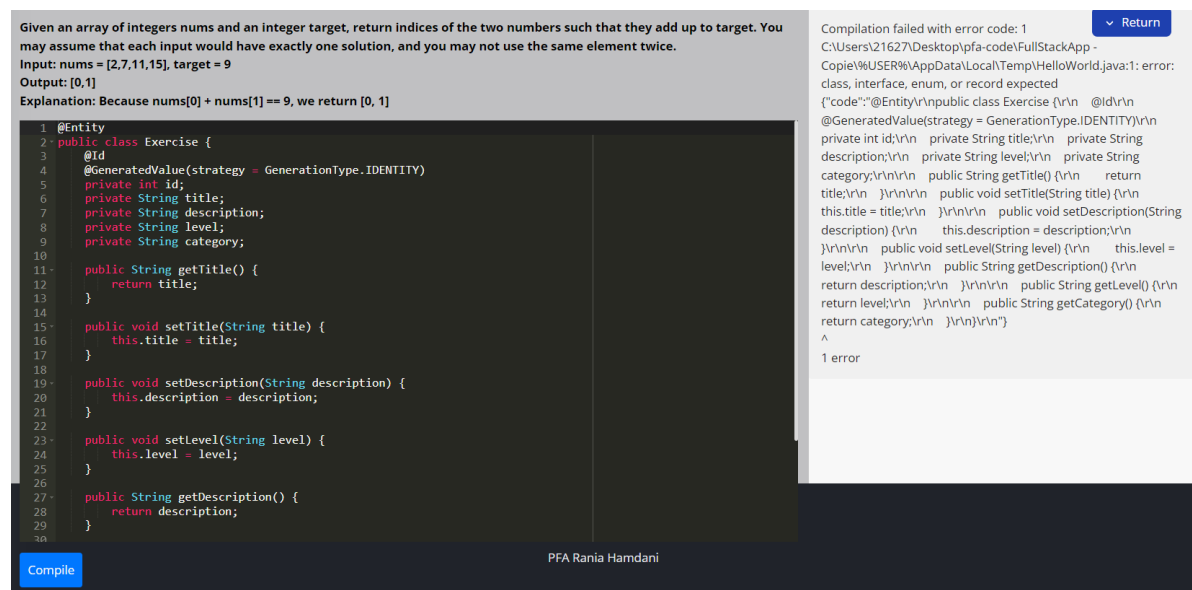


FIGURE 4.4 – Interface de compilation

4.5.4 Problèmes

La figure 4.5 l'interface des problèmes présentant les paramètres par lesquelles l'utilisateur peut filtrer par titre et par niveau de difficulté.

Les problèmes sont supportés par le paramètre "hint" qui est une indication sur la démarche de résolution du problème qui facilite sa résolution. Il y a de même le paramètre "tags" qui représente le thème du problème à résoudre (mathématique, recherche dynamique,..).

Une fois le problème est choisi, l'utilisateur clique sur le bouton "Start" pour commence le problème. Le système affiche une interface contenant l'énoncé du problème et un espace de programmation. Les étapes suivantes sont similaires à celles détaillées dans la partie exercice (voir section 4.5.3).

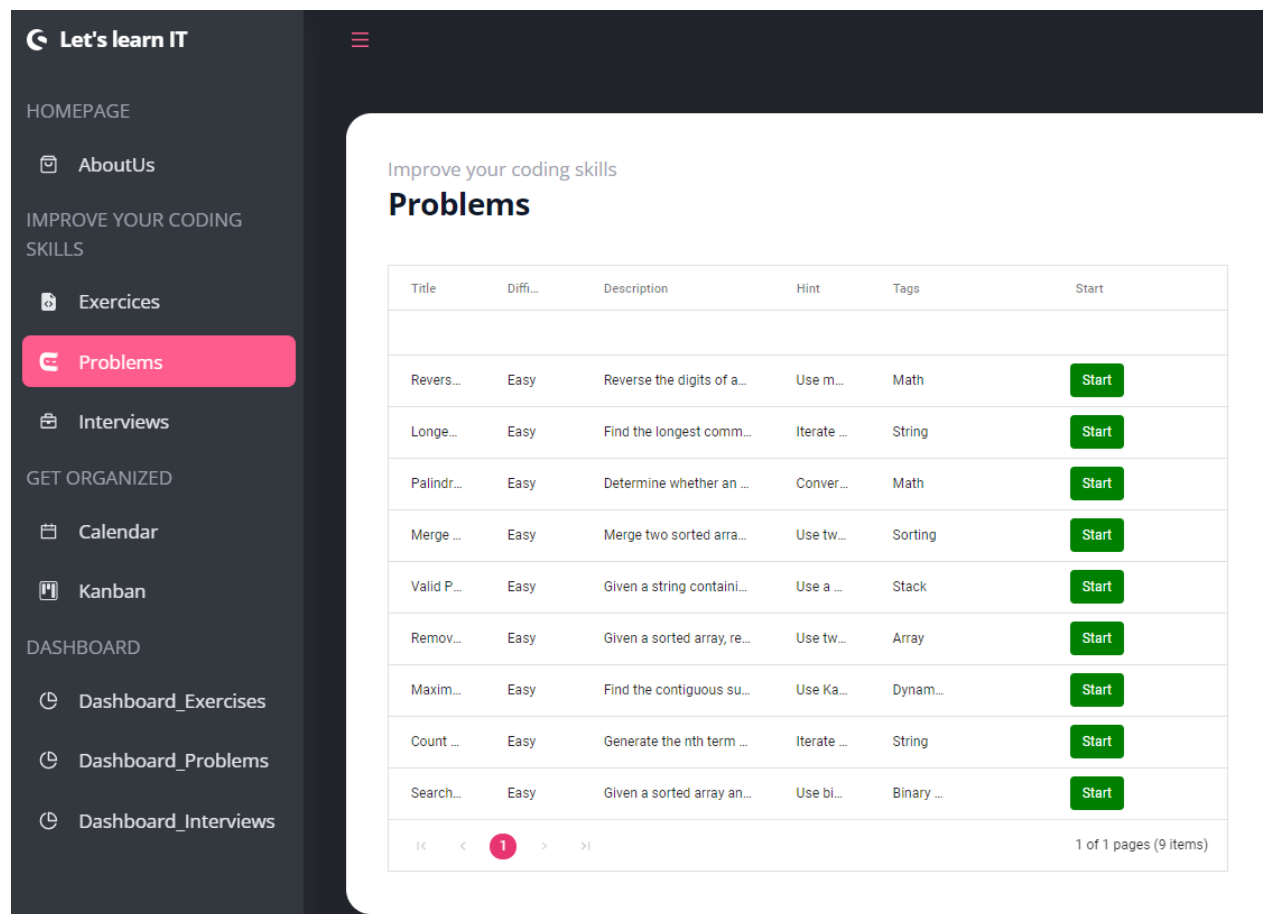


FIGURE 4.5 – Interface des problèmes

4.5.5 Interviews

La figure [4.6](#) l'interface des Interviews présentant les paramètres par lesquelles l'utilisateur peut filtrer. Il peut choisir le nom de l'entreprise qui a proposé cette interview, par son rôle (développeur java, ingénieur test, ..) et par sa ancienneté (junior ou senior).

Une fois l'interview est choisi, il clique sur le bouton "Start" qui affiche l'interface des questions à choix multiple qu'il doit effectuer et le chronomètre commence.

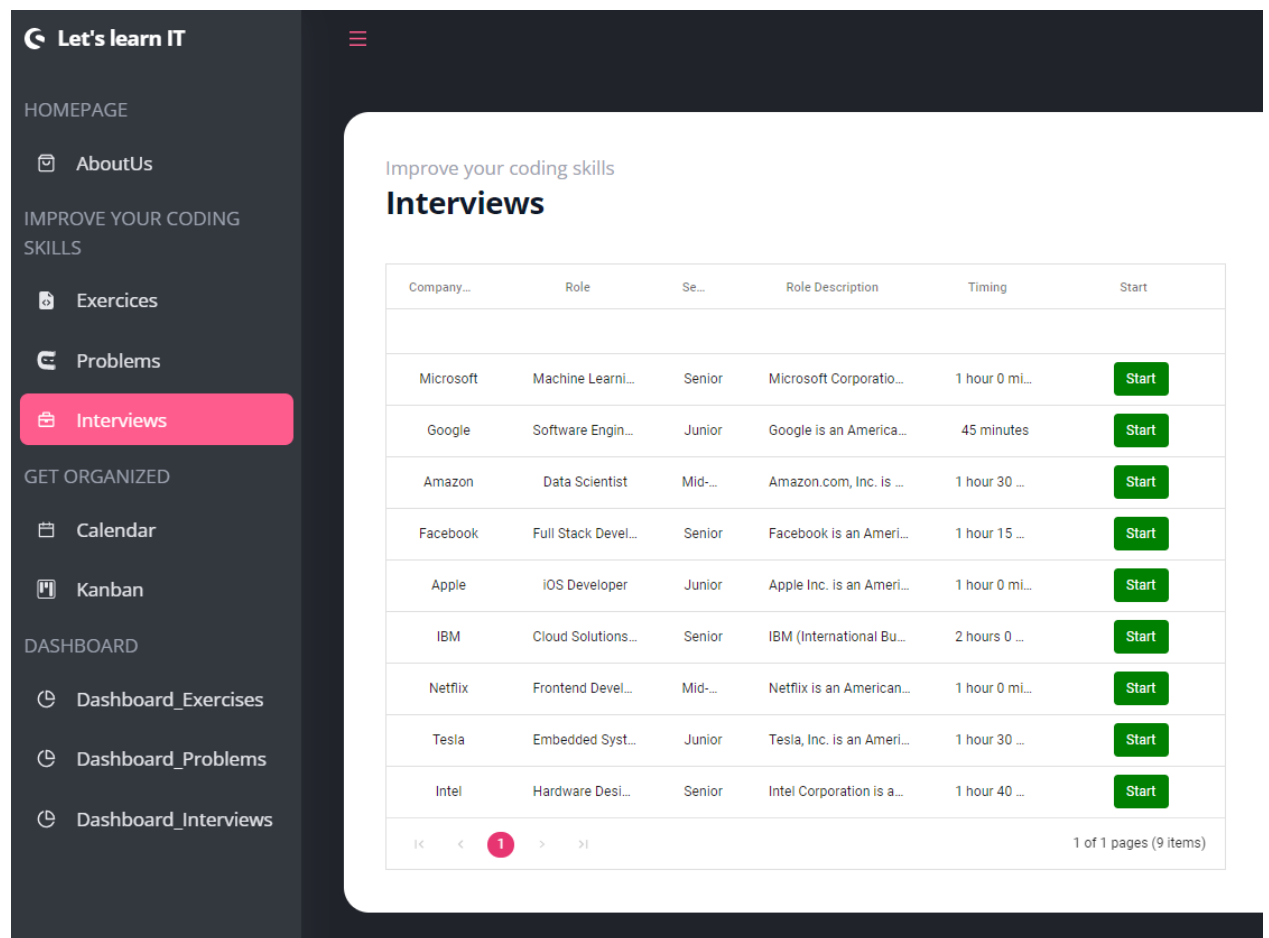


FIGURE 4.6 – Interface des interviews

La figure 4.7 l'interface des questions à quatre choix liées à l'interview choisi. L'utilisateur clique sur la réponse choisi et clique sur "Next" pour passer à la question suivante. En même temps, le chronomètre affiche le temps restant en secondes.

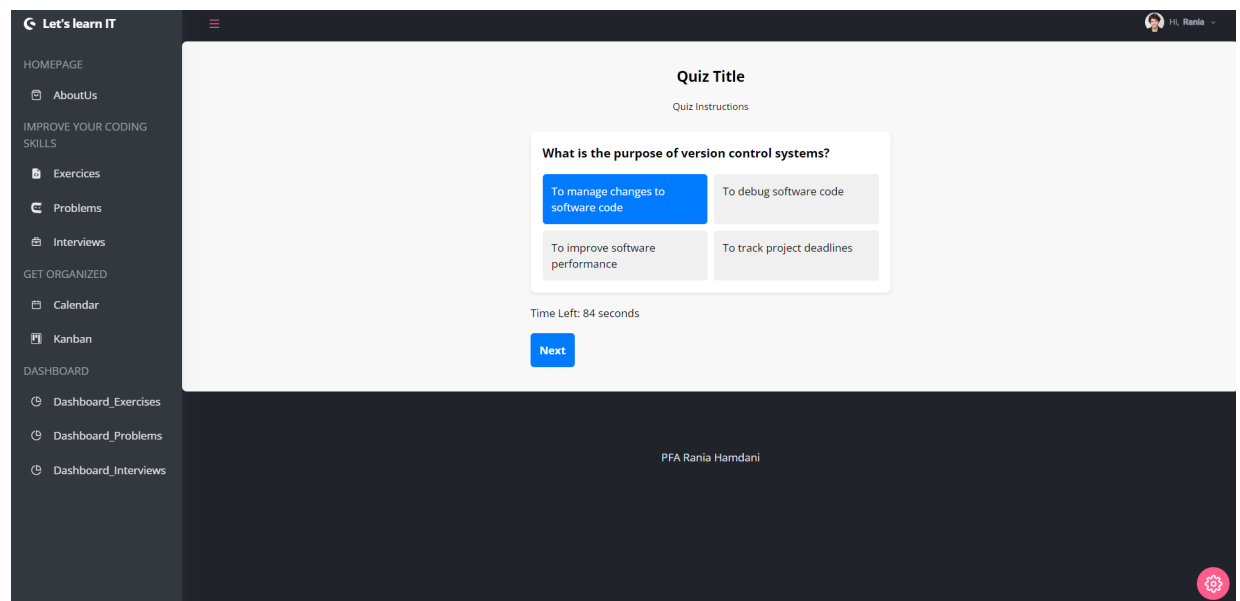


FIGURE 4.7 – Interface des questions

La figure 4.8 l'interface de fin de l'interview qui affiche son score après avoir répondu à tout les questions ou à la fin du temps alloué par le chronomètre.

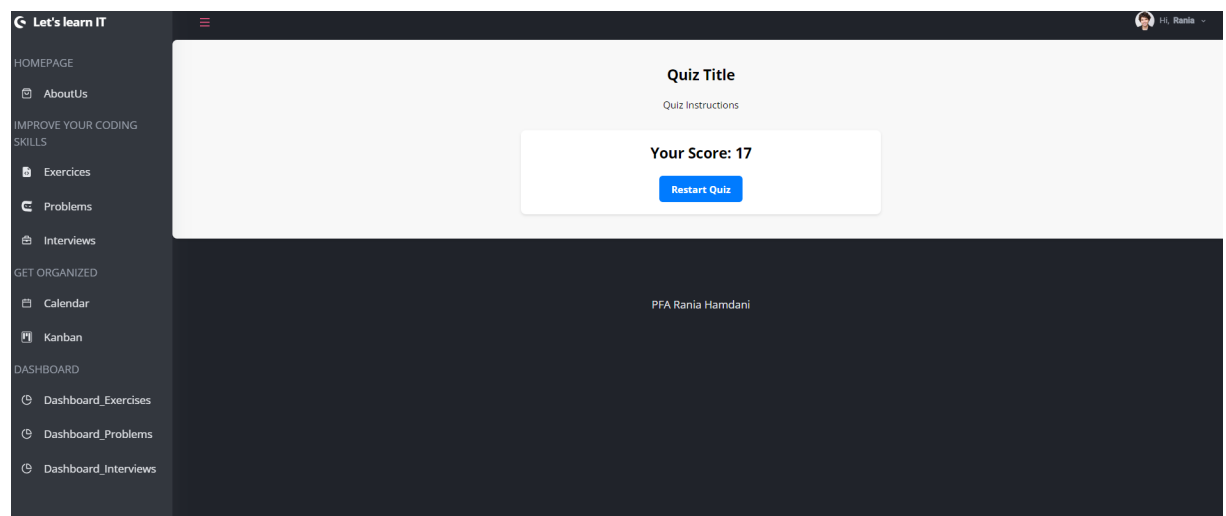


FIGURE 4.8 – Interface de fin de l'interview

4.5.6 Dashboards

Le dashboard des exercices :

L'utilisateur visualise les exercices qu'il a effectué avec succès, indiqués en vert dans le dashboard avec le message "Success" et les exercices qu'il n'a pas encore réussi à faire, indiqués en rouge avec le message "Not done yet!". Ceci est illustré par la figure 4.9.

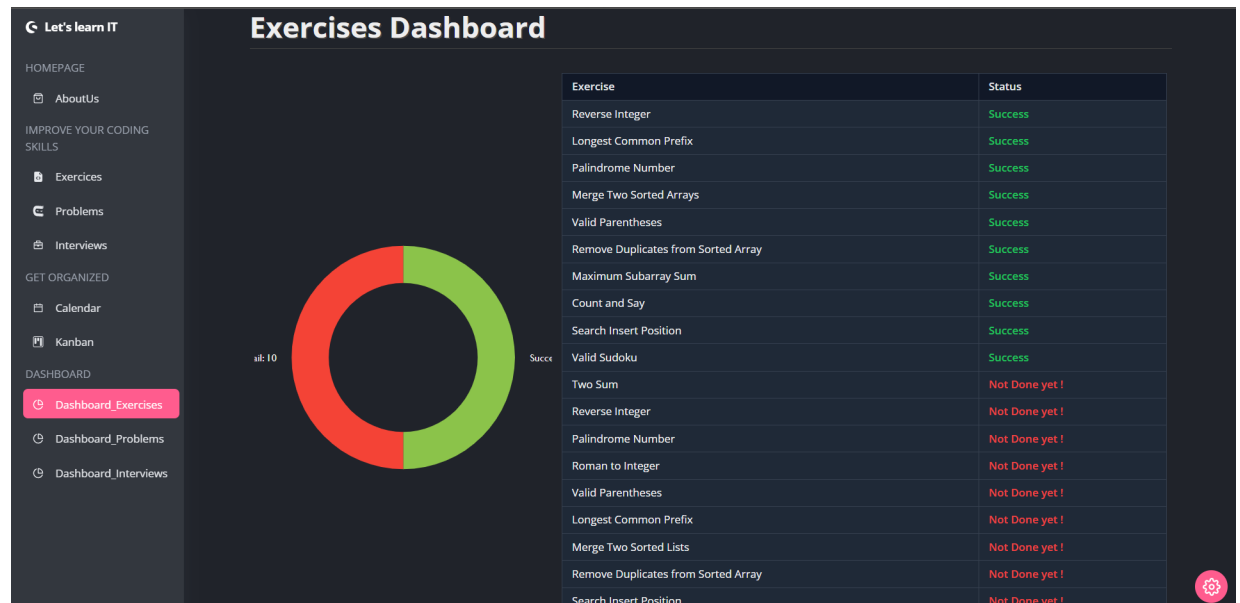


FIGURE 4.9 – Le dashboard des exercices

Le dashboard des problèmes :

L'utilisateur visualise les problèmes qu'il a effectué avec succès. Ils sont indiqués en vert dans le dashboard avec le message "Success". De meme, il visualise les problèmes qu'il n'a pas encore réussi à faire, indiqués en rouge avec le message "Not done yet !" comme indique la figure 4.10.

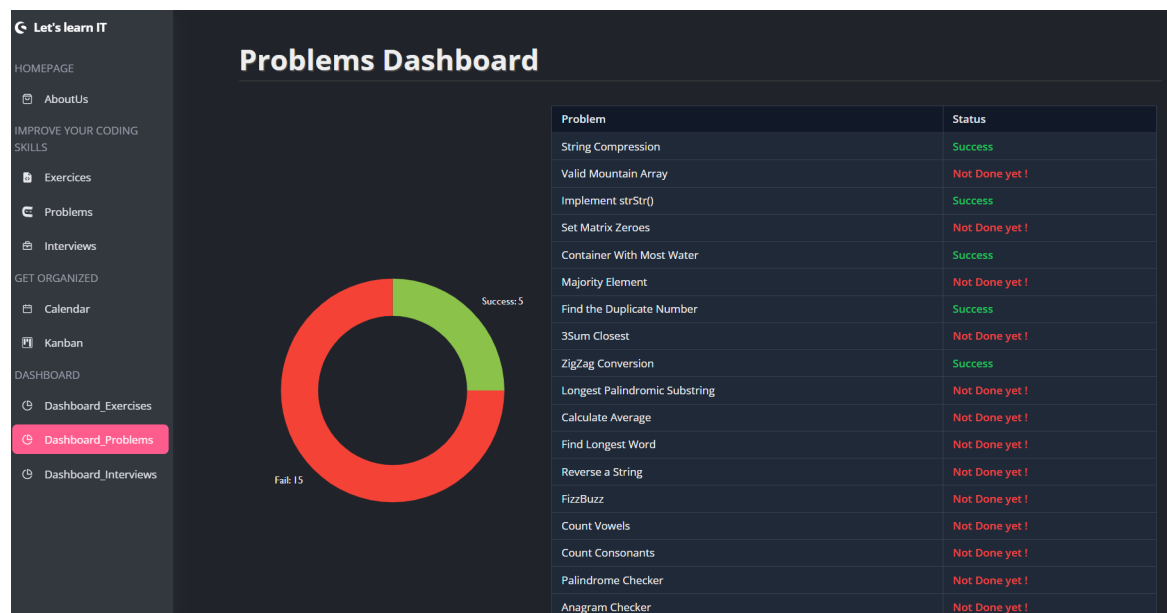


FIGURE 4.10 – Le dashboard des problèmes

Le dashboard des interviews :

L'utilisateur peut voir les interviews qu'il a effectué. Pour chaque interview, nous indiquons son rôle (développeur python, ingénieur en machine learning, ...), le score qu'il a obtenu et le résultat de son interview (Good fit, Able to improve, Not a fit).

- S'il a obtenu plus que 70%, on indique qu'il est "Good fit" qui est le statu de réussite (indiqué en vert).
- S'il a obtenu entre 40% et 70%, on indique qu'il est "Able to improve" qui indique qu'il peut faire mieux (indiqué en jaune).
- S'il a obtenu moins que 40%, on indique qu'il est "Not a fit" qui est le statu d'échec (indiqué en rouge).

Ceci est illustré par la figure 4.11.

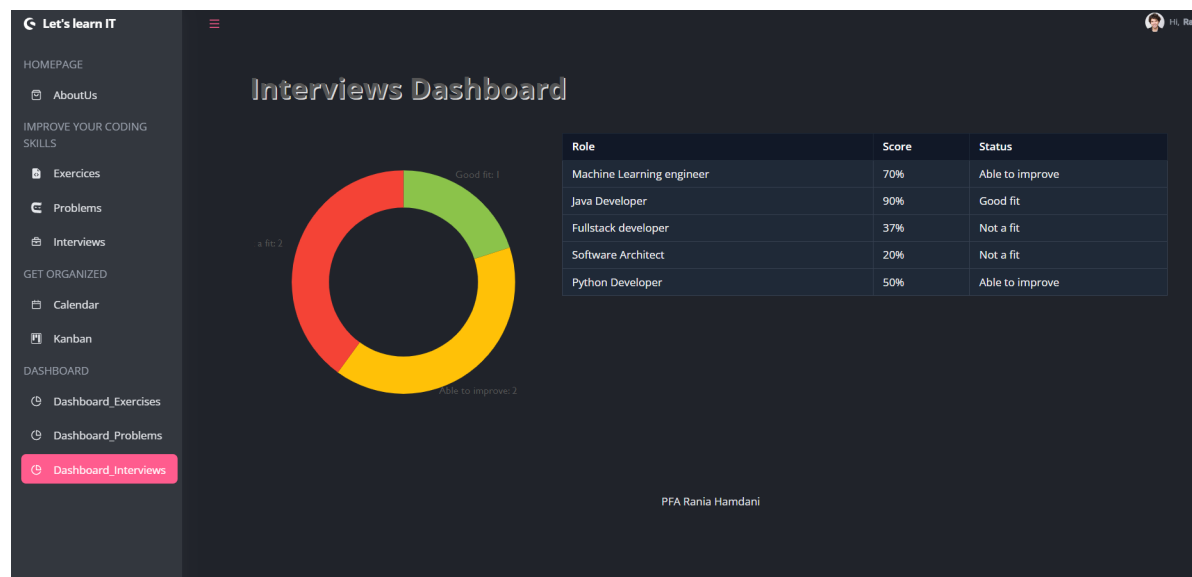


FIGURE 4.11 – Le dashboard des interviews

4.5.7 Calendrier

Le calendrier permet de planifier les exercices, les problèmes et les interviews que l'utilisateur compte faire. Ceci est illustré par la figure 4.12.

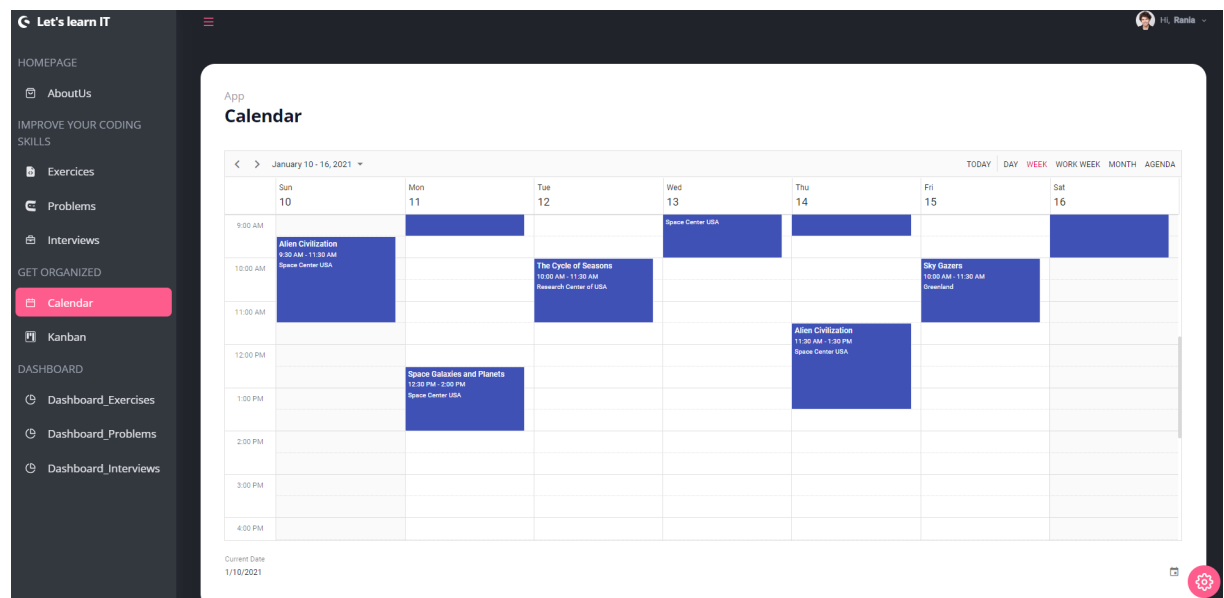


FIGURE 4.12 – Le calendrier

4.5.8 Kanban

Le kanban sert à enregistrer les tâches (à faire, en cours ou terminé). Ceci est illustré dans la figure 4.13.

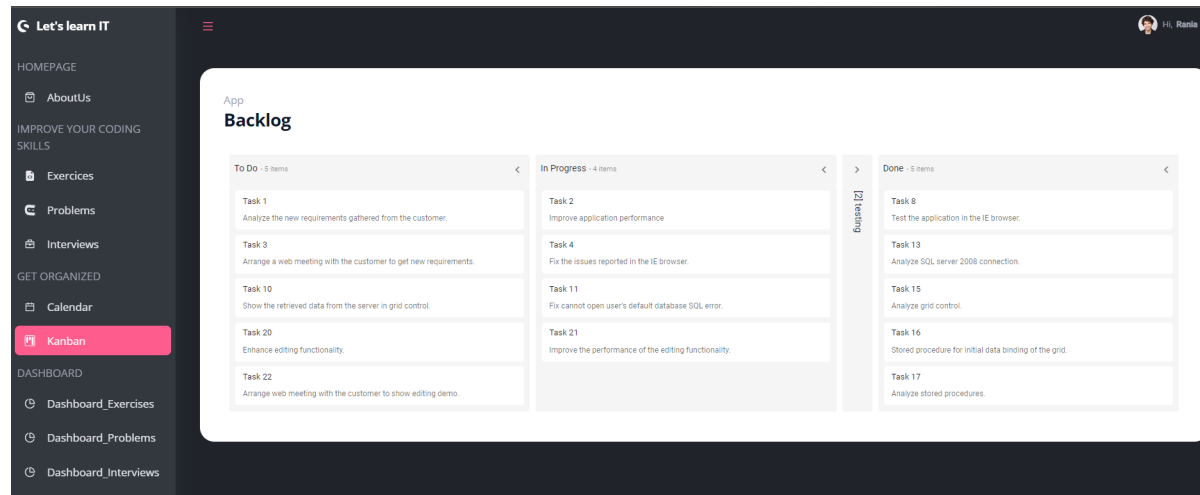


FIGURE 4.13 – Le kanban

4.5.9 Différents thèmes dans l'application

La figure 4.14 présente que l'utilisateur peut choisir un thème pour l'application qui varient entre 6 couleurs. Il peut aussi choisir le mode sombre ou le mode clair selon ses préférences.

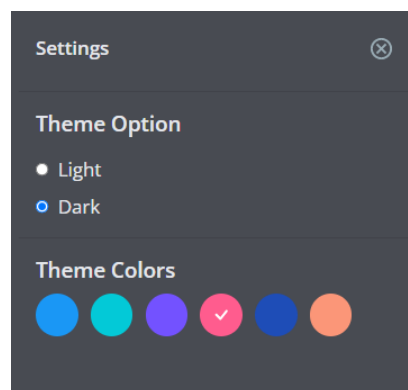


FIGURE 4.14 – Les différents thèmes dans l'application

4.6 Conclusion

Ce chapitre a été consacré à la présentation de l'environnement matériel et logiciel du travail, à la justification des choix technologiques et à l'exposition du projet réalisé.

Conclusion Générale

Nous avons conçu et développé une application web de programmation et de résolution de problèmes. En premier lieu, notre application propose de nombreux exercices de programmation qui ont pour objectif d'aider les utilisateurs à améliorer leurs connaissances de base et leur compétence en langage de programmation Java. L'utilisateur peut filtrer les exercices selon les catégories (tableaux, chaînes, ...) et selon les niveaux de difficultés.

En deuxième lieu, nous proposons un éventail de problèmes à résoudre afin de s'exercer à la résolution de problèmes et d'améliorer la capacité de raisonnement critique. Notre application permet aux utilisateurs de filtrer les problèmes selon le contexte (mathématiques, recherche dynamique, ...). Cette diversité de défis convient aussi bien aux programmeurs débutants qu'aux programmeurs chevronnés.

En troisième lieu, notre application permet aux candidats d'effectuer des simulations d'interviews avec la présence d'un chronomètre pour s'exercer à répondre aux questions de manière efficace et en respectant le temps alloué.

En quatrième lieu, notre application offre trois dashboards : un dashboard pour visualiser les exercices effectués correctement, un dashboard pour visualiser les problèmes résolus par l'utilisateur et un dashboard pour visualiser le score obtenu à la fin de l'interview avec son résultat selon son score. Nous offrons de même un calendrier pour planifier les exercices, les problèmes et les interviews à faire et un kanban pour enregistrer leurs états (à faire, en cours ou terminé).

Dans l'ensemble, ce projet a été une expérience extrêmement enrichissante et nous sommes convaincus que les connaissances acquises tout au long du processus seront précieuses dans notre avenir professionnel. En regardant vers l'avenir, il reste encore beaucoup à faire comme perspectives pour améliorer notre application. Par exemple, nous pouvons envisager d'ajouter des nouvelles fonctionnalités comme l'évaluation de la qualité de code et la proposition de corrections. Il est aussi intéressant d'ajouter d'autres langages de programmation.

Netographie

- [1] « **leetcode** », <https://leetcode.com/problemset/all/>, consulté le 04/03/2023
- [2] « **hackerrank** » <https://www.hackerrank.com/interview/interview-preparation-kit>, consulté le 04/03/2023
- [3] « **topcoder** » <https://www.topcoder.com/about-us/>, consulté le 04/03/2023
- [4] « **codeChef** » <https://www.codechef.com/>, consulté le 04/03/2023
- [5] « **Architecture MVC** », <https://learn.microsoft.com/fr-fr/aspnet/core/mvc/overview?view=aspnetcore-7.0>, consulté le 20/04/2023
- [6] « **VS code** » <https://visualstudio.microsoft.com/fr/vs/>, consulté le 04/03/2023
- [7] « **SQL server** » <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>, consulté le 04/03/2023
- [8] « **postman** » <https://www.postman.com/>, consulté le 04/03/2023
- [9] « **IntelliJ** » <https://www.jetbrains.com/fr-fr/idea/features/>, consulté le 04/03/2023
- [10] « **JAVA** » <https://fr.wikipedia.org/wiki/Java>, consulté le 04/03/2023
- [11] « **spring boot** » <https://spring.io/projects/spring-boot>, consulté le 04/03/2023
- [12] « **typescript** » <https://fr.wikipedia.org/wiki/TypeScript/>, consulté le 04/03/2023
- [13] « **ReactJS** » <https://ibracilinks.com/blog/quest-ce-que-reactjs-et-pourquoi-devrions-nous-utiliser-reactjs>, consulté le 04/03/2023
- [14] « **JavaCompiler** » <https://www.developer.com/design/an-introduction-to-the-java-compiler-api/>, consulté le 07/05/2023

Resumé

Résumé

Dans le cadre de notre projet, nous avons développé une application web de programmation et de résolution de problèmes. Notre application permet aux utilisateurs de s'entraîner avec les exercices de programmation, à apprendre à résoudre différents types de problèmes et à évaluer leurs connaissances avec des interviews. Elle offre une expérience utilisateur fluide et efficace.

Mots clés :exercices, résolution des problèmes, interview, programmation java

Summary

As part of our project, we have developed a web application for programming and problem-solving. Our application allows users to practice programming exercises, learn how to solve different types of problems, and assess their knowledge through interviews. It offers a smooth and efficient user experience.

Keywords : exercises, problem-solving, interview, Java programming.

ملخص :

في إطار مشروعنا، قمنا بتطوير تطبيق ويب للبرمجة وحل المشاكل. يسمح تطبيقنا للمستخدمين بممارسة تمارين البرمجة، وتعلم كيفية حل أنواع مختلفة من المشاكل، وتقييم معرفتهم من خلال المقابلات. يوفر التطبيق تجربة مستخدم سلسة وفعالة
كلمات مفتاحية: تمارين، حل المشاكل، مقابلة، Java