

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de La Manouba
École Nationale des Sciences de l'Informatique



Rapport de Stage d'Immersion en Entreprise

Sujet

AMÉLIORATION DE LA PLATEFORME DE RECRUTEMENT DE LINEDATA

Réalisé par
Donia HAMMAMI



Organisme d'accueil : Linedata Tunisie

Nom du responsable : Mondher Merai

Supervisé par : Aymen Barrak

Adresse : Imm Cleopatre Center, Centre Urbain Nord-1082 TUNIS

Tél : +216 71 185 800

Année Universitaire 2017/2018

Signature du Superviseur

Dédicaces

A mes parents,
Aucun hommage ne pourrait être à la hauteur de l'amour
et de l'affection dont ils ne cessent de me combler.
Qu'ils trouvent dans ce travail un témoignage de mon profond amour et éternelle
reconnaissance.

Que dieu leur procure bonne santé et longue vie.

A mes chères soeurs Abir et Salma pour leur soutien durant toutes ces années,

Une spéciale dédicace à cette personne qui compte
déjà énormément pour moi, et pour qui je porte beaucoup
de tendresse et d'amour.

A toi Mohamed Ghorbel

A tous mes amis, Je dédie ce travail ...

Donia

Remerciements

Je tiens, avant de présenter mon travail, à exprimer ma grande reconnaissance envers les personnes qui m'ont, de près ou de loin, apporter leur soutien.

Je tiens à remercier vivement mon superviseur Mr Aymen Barrak pour son accueil, le temps passé ensemble et le partage de son expertise au quotidien. Grâce aussi à sa confiance j'ai pu m'accomplir totalement dans mes missions. Il fut d'une aide précieuse dans les moments les plus délicats.

Je remercie également l'équipe Icon et Optima pour leur accueil, leur esprit d'équipe qui m'a beaucoup aidé dans mes missions.

Que les membres de jury trouvent, ici, l'expression de mes remerciements pour l'honneur qu'ils nous font en acceptant de juger ce travail.

Table des matières

Introduction générale	1
1 Cadre du projet	3
I Présentation de l'organisme d'accueil	3
I.1 Présentation générale de Linedata	3
I.2 Services offerts par Linedata	4
I.3 Les produits Linedata	6
II Problématique	7
III Critique de la solution de l'existant	7
IV Travail demandé	7
V Méthodologie de gestion de projet	8
V.1 Présentation de la méthodologie SCRUM	8
V.2 Organisation	9
Conclusion	10
2 Analyse et spécification des besoins	11
I Capture des Besoins	11
I.1 Définition des acteurs	11
I.2 Analyse des besoins	12
I.2.1 Besoins fonctionnels	12
I.2.2 Besoins non fonctionnels	14
II Spécification des besoins	14
II.1 Diagrammes de cas d'utilisation	14
II.1.1 Diagramme de cas d'utilisation relatif à l'administrateur	15
II.1.2 Diagramme de cas d'utilisation relatif au test taker . .	16

II.1.3	Diagramme de cas d'utilisation relatif au test maker .	16
II.1.4	Diagramme de cas d'utilisation relatif au test checker .	18
II.2	Description de quelques scénarii	18
II.2.1	Scénario du cas d'utilisation "'S'authentifier"'	18
II.2.2	Scénario du cas d'utilisation "'Créer un test"'	19
Conclusion		20
3	Conception	21
I	Conception architecturale de l'application	21
I.1	Architecture physique	21
I.1.1	Modélisation de l'architecture physique	23
I.2	Architecture logique	24
II	Conception détaillée	25
II.1	Diagramme de paquet	25
II.2	Conception détaillée du Controleur	26
II.3	Conception détaillée du paquet DBContext	27
II.4	Conception détaillée du paquet ViewModels	29
III	Description de quelques scénarii	31
III.1	Scénario "'Créer un test"'	31
III.2	Scénario "'Passer un test"'	32
Conclusion		32
4	Réalisation	33
I	Environnements de travail	33
I.1	Langages de programmation et framework utilisés	33
I.1.1	ASP.NET.MVC framework	33
I.1.2	Entity framework	34
I.1.3	C sharp	34
I.1.4	HTML5, Bootstrap CSS3, jQuery, JavaScript et AJAX pour la construction des interfaces riches	34
I.2	Environnements de développement logiciels	35
I.2.1	Visual studio	35
I.2.2	SQL server	36

	I.2.3	SQL server management studio	36
II		Le Backlog du produit	36
III		Déroulement des Sprints	37
	III.1	Interfaces Homme-machine	38
	III.2	Tests effectués	53
	III.2.1	Tests du sprint 1	53
	III.2.2	Tests du sprint 2	53
	III.2.3	Tests du sprint 3	54
IV		Chronogramme	54
		Conclusion	55
		Conclusion générale	56
		Netographie	58

Table des figures

1.1	Bureaux de Linedata dans le monde [N1]	4
1.2	Cycle de vie de la méthode SCRUM [N2]	9
2.1	Cas d'utilisation relatif à l'administrateur	15
2.2	Cas d'utilisation relatif au test taker	16
2.3	Cas d'utilisation relatif au test maker	17
2.4	Cas d'utilisation relatif au test checker	18
2.5	Diagramme de séquence système d'authentification	19
2.6	Diagramme de séquence système de création de test	20
3.1	Architecture trois tiers [N5]	22
3.2	Diagramme de déploiement du système	23
3.3	Architecture MVC [N7]	24
3.4	Diagramme de paquet de iRecruit	26
3.5	Conception détaillée du paquet Controleur	27
3.6	Modèle relationnel de la base de données	28
3.7	Diagramme de classe du paquet ViewModels	30
3.8	Diagramme de séquence détaillé du scénario créer un test	31
3.9	Diagramme d'activités du scénario Passer un test	32
4.1	Backlog du produit	37
4.2	Décomposition en sprints	38
4.3	Page d'accueil de iRecruit	39
4.4	Page d'authentification	39
4.5	Dashboard d'un test maker	40
4.6	Ajouter une nouvelle catégorie	40

4.7	Ajouter un nouveau sujet	41
4.8	Créer un test	41
4.9	Créer question	42
4.10	Question à choix multiples	42
4.11	Question de type paragraphe	43
4.12	Librairie des questions	43
4.13	Ajouter question	44
4.14	Liste des tests créés	44
4.15	Test créé	45
4.16	Liste de tous les tests	45
4.17	Publier un test	46
4.18	Approuver demandes d'inscription des test takers	46
4.19	Approuver demandes d'inscription des test makers	46
4.20	Liste des test checkers	47
4.21	Assigner un test checker	47
4.22	Choisir un test checker	48
4.23	Liste des resultats obtenus de tous les tests	48
4.24	Test corrigé d'un candidat	49
4.25	Page d'inscription des test takers	50
4.26	Page d'inscription des test takers	50
4.27	Passer test	51
4.28	Ajouter un commentaire	51
4.29	Vérifier état du test	52
4.30	Liste des tests à corriger	52
4.31	Correction d'un test	53
4.32	Diagramme de gantt réel	54

Introduction générale

Atteindre le succès à long terme pour toute entreprise est directement lié à la hiérarchisation des stratégies de croissance et à saisir toutes les opportunités. Outre le type d'entreprise et les contraintes externes auxquelles elle fait face, un niveau de performance élevé est particulièrement important pour garantir tout développement possible.

À cet égard, la main-d'oeuvre est considérée comme l'atout le plus important car elle est essentielle non seulement sur une base opérationnelle quotidienne, mais aussi pour obtenir un grand avantage concurrentiel. Il ne fait aucun doute alors qu'une forte productivité de la main-d'oeuvre assurera une rentabilité à long terme pour toute entreprise.

Trouver des employés qualifiés et fiables est le résultat de l'analyse des compétences techniques, relationnelles et professionnelles dans un processus de recrutement efficace. La quête de l'optimisation de ce processus reste récurrente et tourne autour l'organisation des tests de connaissances pour le recrutement des candidats et nécessitera un haut niveau de planification, de délégation de rôle et de communication entre les différents participants.

L'équipe de recrutement de Linedtata reconnaît l'importance de cette phase et fait actuellement la plupart de celle-ci manuellement. Cela a causé des retards et une perte d'efforts et de temps. Ce qui pose également des difficultés lors de la communication des différentes tâches à tous les intervenants. Sans oublier les problèmes liés à la gestion des grandes archives qui dépendent constamment de la consultation et de la sélection des tests.

C'est pourquoi, nous avons constaté que l'automatisation au moins du processus de test de recrutement est une nécessité urgente et primordiale.

Dans ce contexte s'inscrit notre stage d'immersion en entreprise du cycle de formation des ingénieurs à l'École Nationale des Sciences de l'Informatique, réalisé à Linedata, fournisseur global de solutions dans le domaine de l'investissement et de la gestion du crédit.

Notre tâche consiste à faire des améliorations sur la plateforme web et intranet, afin de gérer les tests de connaissances lors du processus de recrutement du département des ressources humaines de Linedata. Nous devons effectuer aussi les tests appropriés pour vérifier si le travail atteint répond aux exigences initialement indiquées et nous fournirons le déploiement de la solution sur les serveurs de Linedata.

Le présent rapport décrit les différentes étapes de notre travail, et il s'articule autour de quatre chapitres :

Le premier chapitre comporte une brève présentation de l'organisme d'accueil Linedata Tunisie et du cadre général de ce projet. Il expose en effet, l'étude de l'existant et met l'accent sur la solution proposée. Et il aborde à la fin la méthodologie de gestion de projet appliquée pour assurer le bon déroulement de notre travail.

Le deuxième chapitre présente en premier lieu une analyse détaillée des besoins fonctionnels et non fonctionnels de notre plateforme. En second lieu, il décrit les différents cas d'utilisation.

Le troisième chapitre détaille l'approche conceptuelle adoptée pour mettre en place notre plateforme iRecruit.

Le quatrième chapitre illustre les choix technologiques et expose les résultats obtenus à partir de quelques interfaces homme-machine.

Nous clôturons par une conclusion générale qui présente une récapitulation du travail réalisé et présente un nombre de perspectives.

Chapitre 1

Cadre du projet

Dans ce premier chapitre, nous allons nous intéresser tout d'abord à la présentation du cadre de notre projet. Il s'agit en effet d'une présentation de l'organisme d'accueil.

Après l'exposition de la problématique, nous abordons l'étude de l'existant et nous présentons ensuite la solution proposée. Enfin, nous entamons ce chapitre par une présentation de la méthodologie de gestion de projet adoptée.

I Présentation de l'organisme d'accueil

Nous allons présenter l'organisme d'accueil dans laquelle s'est déroulé notre stage d'été, ses services et ses produits.

I.1 Présentation générale de Linedata

Linedata [N1] est un éditeur de logiciels mondial dédié aux professionnels de l'asset management, de l'assurance et du crédit. Linedata a été au service de l'industrie financière dès le premier jour et fournit des logiciels et des services critiques et novateurs qui aident ses clients à se développer dans plus de 50 pays.

En effet, pionnier il y a plus de 10 ans avec la mise en place d'infrastructures ASP pour le monde financier, Linedata a évangélisé et répandu ce modèle à travers toute la communauté financière dans le monde et sur toute sa gamme de produits. Ses clients peuvent ainsi croître et se déployer très vite en s'appuyant sur ses infrastructures. Depuis sa création Linedata est plus qu'un simple éditeur de logiciels. Elle soutient

ses clients dans toutes les étapes de leurs projets. Ainsi Linedata prend en charge la conception et le développement de ses solutions logicielles, réalise leur intégration et offre toutes les solutions de support et d'hébergement dont les établissements financiers ont besoin.

Linedata a su acquérir et intégrer de nombreuses sociétés en quelques années. Cette politique pro-active lui permet d'accompagner ses clients à travers des solutions Front to Back intégrées et modulaires pour les professionnels de la gestion d'actifs, de l'assurance et du crédit. Basé en France, Linedata a réalisé un chiffre d'affaires de 160,3 millions d'euros en 2013.

Linedata a une organisation régionale avec des bureaux à travers le monde et avec plus de 900 employés tel que présentée par la figure 1.1 ci-dessous. Linedata est une entreprise multiculturelle, ouverte à toutes les nationalités du monde.



FIGURE 1.1 – Bureaux de Linedata dans le monde [N1]

I.2 Services offerts par Linedata

Linedata [N1] offre 6 types de services différents :

- **SERVICES INSTALLATION ET DÉPLOIEMENT** : Linedata propose une offre de services complète pour l'installation des plate-formes technologique. Elle travaille de façon étroite avec les clients afin de comprendre leurs besoins, qu'ils

soient standards ou spécifiques, et d'y répondre de façon personnalisée. Une phase d'étude préalable permet de leur offrir une installation optimisée en termes de calendrier, de coûts et de ressources.

- **SERVICES HÉBERGEMENT ET SAAS** : Depuis plusieurs années la communauté informatique étudie le déploiement d'architectures SOA et la mise en place de solutions SaaS. Pionnier en Europe puis aux Etats Unis avec le déploiement d'offres ASP pour les professionnels Buy Side, Linedata a étendu son savoir-faire afin de proposer divers types d'approches dans la gestion de projet technologique.
- **SERVICES FORMATION** : Les programmes de formation que Linedata propose à ses clients leur apportent l'expertise nécessaire afin de maximiser leur utilisation de sa gamme de solutions logicielles. Quel que soit le produit qu'ils utilisent, Linedata apporte la formation adéquate en fonction des besoins des utilisateurs.
- **SERVICES SUPPORT CLIENT** : Le service qu'apporte Linedata à ses clients en plus de ses solutions constitue le ciment des relations de confiance qu'elle tisse avec eux depuis des années. L'accompagnement de ses clients fait partie intégrante de son offre. Au cours des années, elle a étendu le périmètre de services offerts, afin de répondre au mieux aux besoins opérationnels et technologiques de ces clients, avec pour objectif essentiel de leur apporter une plus grande compétitivité.
- **SERVICES ACCOMPAGNEMENT SUR MESURE** : Linedata offre une gamme de services d'accompagnement de ses solutions à travers des prestations de consulting et des développements spécifiques afin que ses clients tirent le meilleur parti de la technologie et définissent leur stratégie opérationnelle.
- **SERVICES BUY SIDE** : Linedata travaille en partenariat étroit avec les acteurs "buy side". A leurs côtés, elle a pu développer une connaissance très approfondie des services requis par ce type d'organisation. Les organisations étant à la recherche de solutions plus efficaces pour répondre aux besoins de leurs clients, Linedata travaille avec eux pour offrir de nouveaux services en phase avec ces demandes croissantes.

I.3 Les produits Linedata

Linedata offre à ses clients 27 produits à savoir :

- Linedata Admin Edge.
- Linedata Bdb.
- Linedata Capitalstream.
- Linedata Chorus.
- Linedata Compliance.
- Linedata Disclosure Manager.
- Linedata Ekip 360.
- Linedata Global Hedge.
- Linedata I-BOR.
- Linedata I-CIPS.
- Linedata Icon.
- Linedata Icon Retail.
- Linedata Longview.
- Linedata Longview for Wealth Managers.
- Linedata Lynx.
- Linedata Master I.
- Linedata Mfact.
- Linedata Mshare.
- Linedata Navquest.
- Linedata Optima.
- Linedata Noee.
- Linedata Profinance.
- Linedata Reporting.
- Linedata Star.
- Linedata Trader+.
- Linedata Uniloan.
- Linedata Webpass.

II Problématique

La procédure de test suivie par l'équipe de recrutement de Linedata devrait viser à fournir une sélection dynamique, automatisée et efficace à partir d'une liste de candidats. Le processus de recrutement devrait être optimisé et devrait être également assez rapide pour pouvoir être facilement compris et réalisé. Cependant, le processus de recrutement actuel de Linedata est une utilisation inefficace du temps et des ressources de l'entreprise. En procédant de manière manuelle pour vérifier et mener les tests, le processus de recrutement est ralenti et se heurte constamment à des problèmes de communication. L'ancienne procédure de recrutement entraîne également la nécessité de la présence de l'équipe RH lors des tests afin de surveiller en permanence les candidats. Ajouter à cela, il y a une difficulté remarquée de gestion des archives de tests vue la quantité énorme de papiers. Pour résoudre les problèmes énumérés ci-dessus, Linedata prévoit de mettre en place une plateforme intranet pour mener à bien le processus de recrutement. C'est pour quoi, Linedata voit qu'il est temps d'automatiser ce processus et d'abandonner les méthodes manuelles.

III Critique de la solution de l'existant

La première tentative de développement de la plateforme de recrutement, faite par un autre stagiaire, a été ratée puisqu'elle ne répond pas tout à fait aux exigences décrites par Linedata. Plusieurs bugs et erreurs sont trouvés par les testeurs doivent être corrigées et beaucoup de fonctionnalités manquantes doivent être développées.

C'est pourquoi, il est primordial d'apporter des améliorations sur la solution existante et corriger les éventuelles anomalies trouvées.

IV Travail demandé

Après une étude approfondie de la solution existante et discussion avec les futurs utilisateurs de la plateforme de recrutement, il est donc primordial au regard des limites de iRecruit d'améliorer cette solution afin qu'elle pourra répondre aux besoins des recruteurs de Linedata et en particulier l'équipe RH. Notre stage d'été consiste à améliorer iRecruit, corriger les erreurs dégagées par l'équipe de test et à développer les

fonctionnalités manquantes.

V Méthodologie de gestion de projet

Le choix entre une méthode et une autre, dépend de la nature du projet et de sa taille. Pour des projets de petite taille et dont le domaine est maîtrisé, par exemple, un cycle de vie en cascade s'avère largement suffisant. Lorsqu'il s'agit d'un projet où les données ne sont pas réunies dès le départ, où les besoins sont incomplets voire flous, il faut s'orienter vers une méthode itérative ou orientées prototypes.

Parmi les méthodes itératives, nous pouvons distinguer les méthodes AGILE largement utilisées de nos jours à travers le monde. Une méthode AGILE est menée dans un esprit collaboratif et s'adapte aux approches incrémentales. Elle engendre des produits de haute qualité tout en tenant compte de l'évolution des besoins du client. Une méthode AGILE assure une meilleure communication avec le client et une meilleure visibilité du produit livrable. Elle permet aussi de gérer la qualité en continu et de détecter des problèmes le plus tôt au fur et à mesure, permettant ainsi d'entreprendre des actions correctrices sans trop de pénalités dans les coûts et les délais. Il y a plusieurs méthodes AGILE et il ne s'agit pas de choisir la meilleure méthode parmi celles existantes. Il s'agit plutôt de sélectionner la méthode la plus adaptée au projet. La nature de projet qui doit être évolutif et dont tous les besoins n'ont pas encore été totalement identifiés, nous a orientées vers une méthode de type AGILE et plus particulièrement SCRUM.

V.1 Présentation de la méthodologie SCRUM

Le principe de la méthodologie SCRUM [N2] est de développer un logiciel de manière incrémentale en maintenant une liste totalement transparente des demandes d'évolutions ou de corrections à implémenter. Avec des livraisons très fréquentes, le client reçoit un logiciel fonctionnel à chaque itération. Plus nous avançons dans le projet, plus le logiciel est complet et possède toujours de plus en plus de fonctionnalités. Pour cela, la méthode s'appuie sur des développements itératifs à un rythme constant. La figure 1.2 ci-dessous illustre le cycle de vie de la méthode SCRUM :



FIGURE 1.2 – Cycle de vie de la méthode SCRUM [N2]

V.2 Organisation

La méthodologie SCRUM [N3] fait intervenir 3 rôles principaux qui sont :

- **Product owner** : dans la majorité des projets, le responsable produit (Product owner) est le responsable de l'équipe projet client. C'est lui qui va définir et prioriser la liste des fonctionnalités du produit et choisir la date et le contenu de chaque sprint sur la base des valeurs (charges) qui lui sont communiquées par l'équipe,
- **Scrum Master** : véritable facilitateur sur le projet, il veille à ce que chacun puisse travailler au maximum de ses capacités en éliminant les obstacles et en protégeant l'équipe des perturbations extérieures. Il porte également une attention particulière au respect des différentes phases de SCRUM,
- **Equipe** : l'équipe s'organise elle-même et elle reste inchangée pendant toute la durée d'un sprint. Elle doit tout faire pour délivrer le produit.

Dans notre projet, nous pouvons distinguer les rôles suivants :

- Product owner : M. Abdallah Hajji,
- Scrum Master : M. Aymen Barrak,
- Testeur : Mlle. Marwa Ben Aissia,
- Développeur : Mlle. Donia Hammami.

Conclusion

Dans ce chapitre, nous avons présenté le contexte général du projet. Nous avons commencé tout d'abord par une présentation de l'organisme d'accueil qui a été suivie par une étude de l'existant. Ceci nous a permis de comprendre les besoins et d'envisager la solution la plus adéquate aux attentes du client.

Le prochain chapitre est consacré à la présentation des besoins fonctionnels et non fonctionnels. Nous terminons par une spécification de ces besoins en nous basant sur les diagrammes d'UML.

Chapitre 2

Analyse et spécification des besoins

L'application qu'on se propose d'améliorer doit tenir en compte des exigences des différents utilisateurs. Une étude des besoins de ces acteurs est alors nécessaire. Le présent chapitre s'articule autour de deux principaux volets : un premier où nous exposerons les différents besoins fonctionnels et non fonctionnels des différents acteurs. Un deuxième volet concerne la spécification de ces besoins via les diagrammes de cas d'utilisation.

I Capture des Besoins

L'étape de l'analyse des besoins est très importante puisque la réussite de toute application dépend de la qualité de son étude. Il faut donc bien déterminer les fonctions attendues par le système.

I.1 Définition des acteurs

Avant d'analyser les besoins, nous avons identifié trois acteurs : Le test taker, le test maker, le test checker et l'administrateur.

- L'administrateur : Un administrateur d'iRecruit est un collaborateur Linedata dont le rôle est de gérer le fonctionnement de la plateforme et de superviser le processus de test en général.
- Le Test Maker : Un Test Maker est un collaborateur Linedata dont le rôle est de créer des questions et des tests.

- Le Test Checker : Un Test Checker est un collaborateur Linedata dont le rôle est de corriger les tests.
- Le Test Taker : Un Test Taker est un candidat qui a terminé avec succès la phase de sélection préliminaire et doit passer par le processus de recrutement de Linedata.

I.2 Analyse des besoins

Les besoins sont divisés en deux catégories, à savoir les besoins fonctionnels et les besoins non fonctionnels.

I.2.1 Besoins fonctionnels

Ce sont les actions et les réactions que le système doit faire suite à une demande d'un acteur principal. Tenant compte de la nature de l'application, on distingue les besoins par acteurs :

- **Administrateur :**

L'application doit permettre à l'administrateur de :

- Approuver et/ou refuser les demandes d'inscriptions des utilisateurs.
- Consulter la liste des utilisateurs de la plateforme.
- Accéder aux informations relatives aux utilisateurs de la plateforme.
- Supprimer des utilisateurs approuvés.
- Publier ou refuser un test.
- Supprimer un test.
- Assigner les tests effectués aux test checkers pour la vérification.
- Consulter les résultats des tests.
- Accéder aux commentaires des test takers.
- Connaître l'avancement du processus de vérification pour un test pris.
- Avoir une classification en fonction de la prise ou du refus d'un test.
- Modifier ses informations.

- **Test Maker :**

L'application doit permettre au test Maker de :

- S'inscrire sur la plateforme.
- Créer des tests selon une catégorie et un sujet bien déterminés.
- Créer des questions pour un test donné.
- Ajouter les réponses aux questions et la note pour chaque question créée.
- Modifier les détails généraux d'un test.
- Modifier le contenu des questions et des réponses dans un test.
- Parcourir la bibliothèque des questions et faire des requêtes de recherche selon des critères spécifiques.
- Ajouter des questions aux tests depuis la bibliothèque.
- Générer dynamiquement des questions pour un test à partir de la bibliothèque.
- Supprimer des questions de la bibliothèque.
- Supprimer des questions dans un test.
- Supprimer des tests.
- Accéder à la liste des tests créés.
- Modifier ses paramètres.

- **Test Checker :**

L'application doit permettre au test checker de :

- S'inscrire sur la plateforme.
- Consulter la liste des tests à corriger.
- Attribuer une note pour chaque question d'un test.
- Modifier ses paramètres.

- **Test Taker :**

L'application doit permettre au test taker de :

- S’inscrire sur la plateforme.
- Consulter la liste des tests assignés.
- Accepter ou refuser de passer un test.
- Consulter l’avancement de la vérification des tests passés.
- Consulter le score obtenu après la correction du test.
- Accéder à l’espace des commentaires.

I.2.2 Besoins non fonctionnels

Les besoins non fonctionnels correspondent à la manipulation de l’application et précisent l’environnement de l’application.

- **Compréhensibilité** : le système doit permettre aux utilisateurs d’avoir une idée du contenu de la plateforme.
- **L’extensibilité** : L’architecture de l’application permettra l’évolution et la maintenance (ajout ou suppression ou mise à jour) au niveau de ses différents modules d’une manière flexible.
- **La sécurité** : L’accès aux informations n’est possible qu’après vérification des privilèges et des droits d’accès. Ainsi tout utilisateur passera par une phase d’authentification pour pouvoir consulter les services offerts par l’application.
- **L’ergonomie et la convivialité** : L’application fournira une interface conviviale et simple à utiliser et qui ne requiert aucun prérequis, donc elle pourra être exploitable par tout type d’utilisateurs (même les non informaticiens).

II Spécification des besoins

Pour la spécification des besoins, nous nous référerons aux diagrammes d’UML : les diagrammes de cas d’utilisation et les diagrammes de séquence.

II.1 Diagrammes de cas d’utilisation

Dans cette partie, nous présentons les diagrammes de cas d’utilisation principaux par acteur. Cette phase représente la vue fonctionnelle de notre application.

II.1.1 Diagramme de cas d'utilisation relatif à l'administrateur

La figure 2.1 ci-dessous représente le diagramme de cas d'utilisation relatif à l'administrateur de notre application.

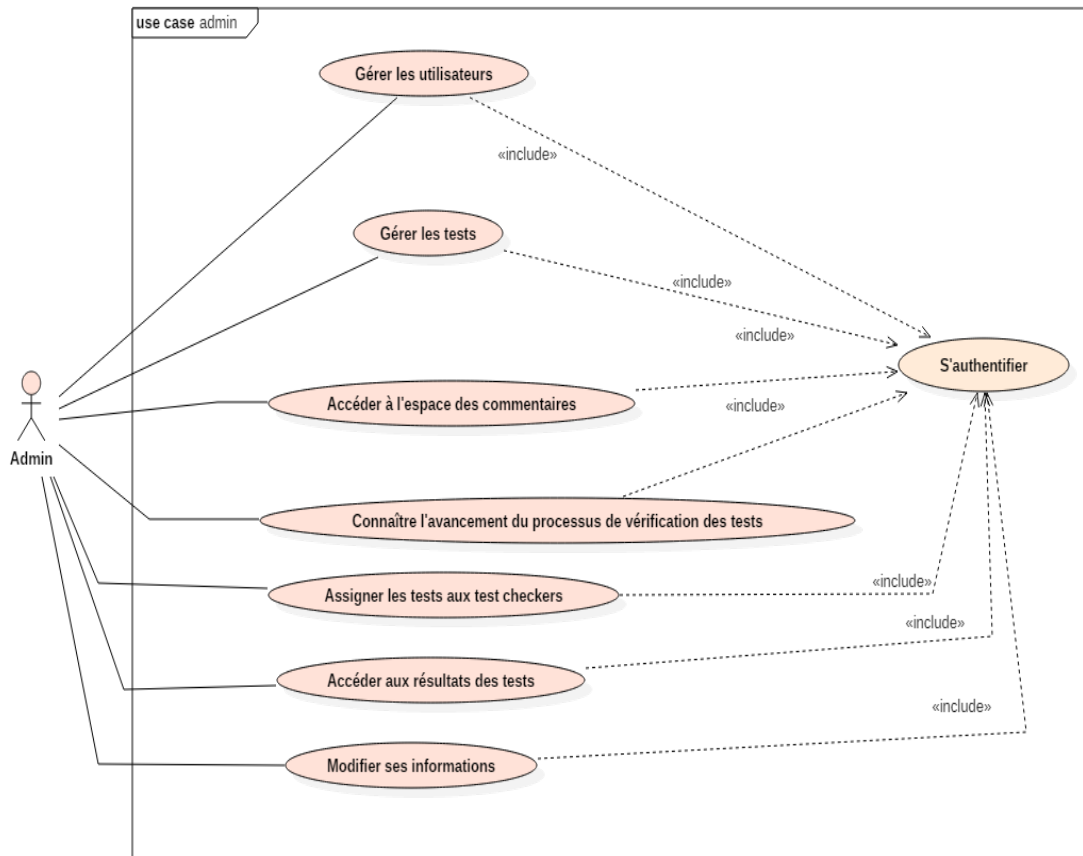


FIGURE 2.1 – Cas d'utilisation relatif à l'administrateur

Ce diagramme de cas d'utilisation présente les différentes fonctionnalités que l'administrateur de notre application peut faire. L'administrateur doit s'authentifier pour accéder à son espace sur la plateforme. Il est le responsable des comptes utilisateurs, en effet, il peut accepter ou refuser les demandes d'enregistrement des utilisateurs sur la plateforme comme il peut supprimer un compte utilisateur existant. Il est le responsable de la gestion des tests. L'administrateur est celui qui publie ou non les tests créés par les test makers, il a aussi la possibilité de supprimer les tests existants. L'administrateur peut consulter les commentaires des test takers sur les tests.

II.1.2 Diagramme de cas d'utilisation relatif au test taker

La figure 2.2 ci-dessous représente le diagramme de cas d'utilisation relatif à un test taker.

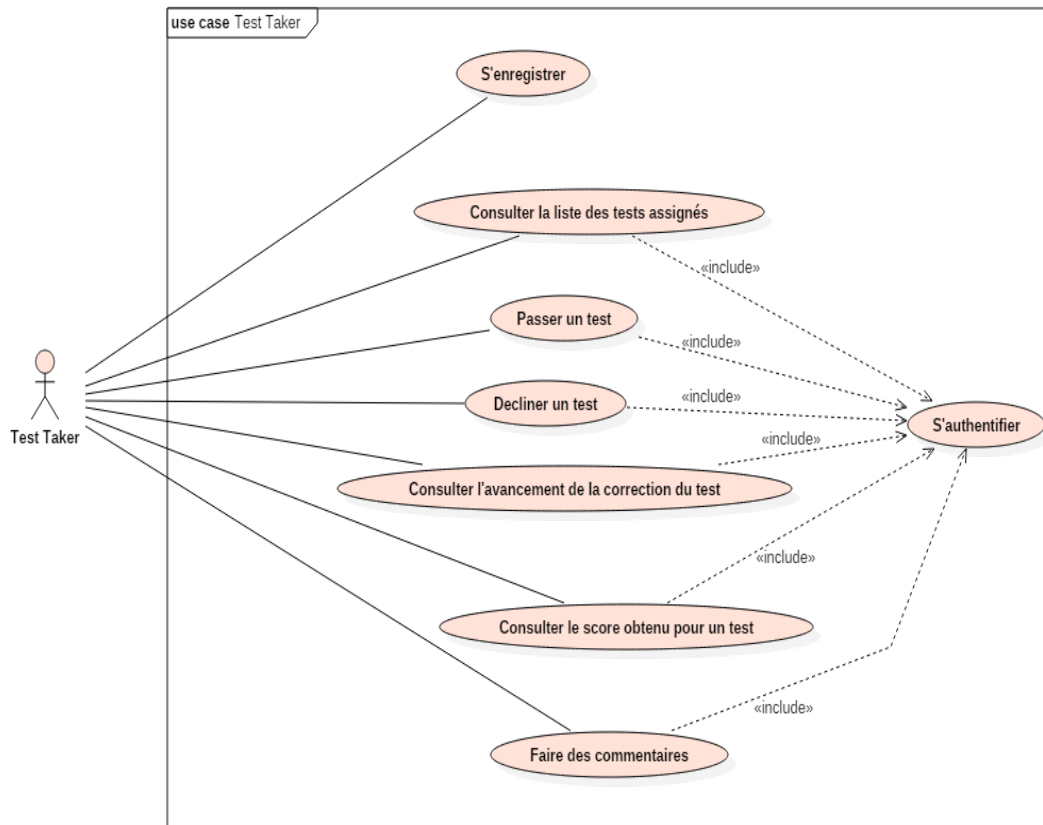


FIGURE 2.2 – Cas d'utilisation relatif au test taker

Ce diagramme de cas d'utilisation présente les différentes fonctionnalités qu'un test taker peut faire. Un test taker doit tout d'abord s'enregistrer sur la plateforme. Après que l'administrateur approuve sa demande, le candidat doit s'authentifier pour accéder à son espace. Il peut consulter la liste des tests qui lui sont assignés. Il a la possibilité d'accepter ou de refuser de passer un test. Il peut faire le suivi des tests passés et consulter le score qu'il a obtenu. Un test taker peut aussi laisser des commentaires sur les tests qu'il a passé comme il peut supprimer ses commentaires.

II.1.3 Diagramme de cas d'utilisation relatif au test maker

La figure 2.3 ci-dessous représente le diagramme de cas d'utilisation relatif à un test maker.

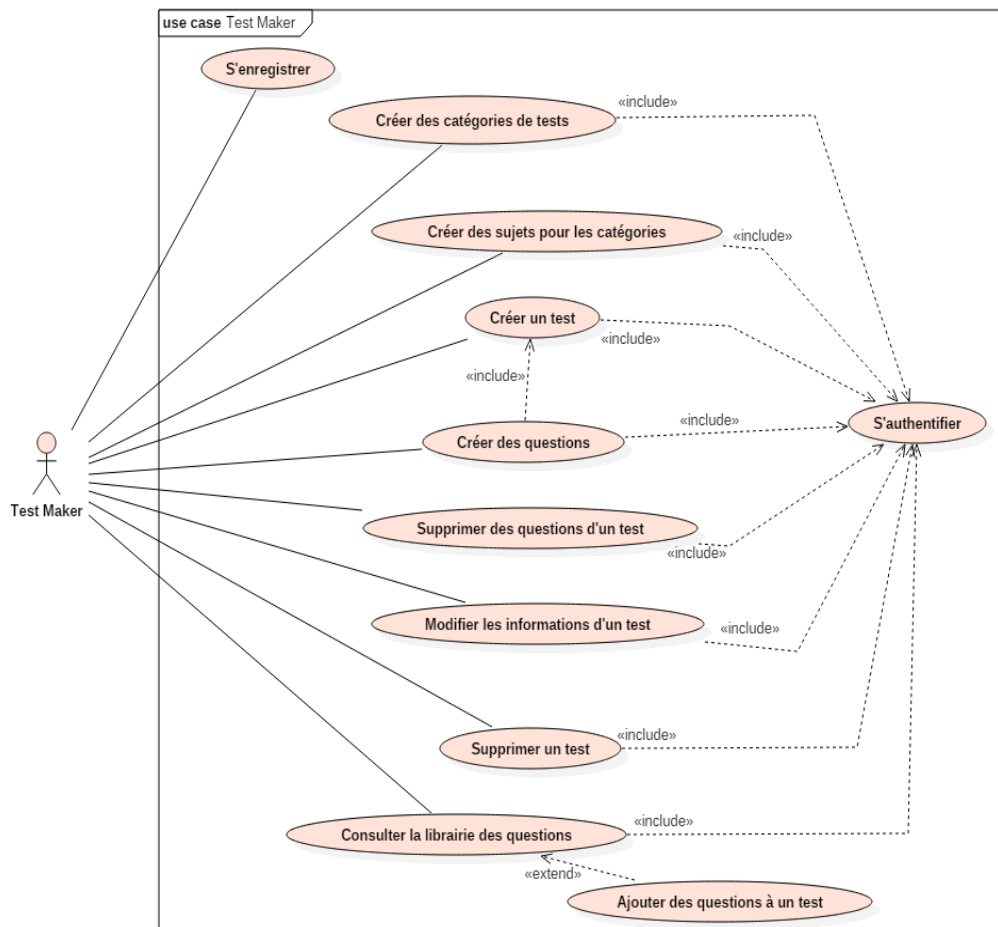


FIGURE 2.3 – Cas d'utilisation relatif au test maker

Ce diagramme de cas d'utilisation présente les différentes fonctionnalités qu'un test maker peut faire. Un test maker doit tout d'abord s'enregistrer sur la plateforme. Après que l'administrateur approuve sa demande, il peut accéder à son espace afin de créer des tests. Un test maker crée des catégories de test et pour chaque catégorie il peut créer plusieurs sujets. Ensuite, il crée un test en remplissant les différents champs et en choisissant une catégorie et un sujet. Un test maker peut aussi créer des questions pour un test donné. Pour chaque question il doit ajouter le score, le type de la question et la réponse correcte. Il peut consulter la librairie des questions et ajouter une ou plusieurs questions à un test. Un test maker peut modifier ou supprimer des tests ou des questions.

II.1.4 Diagramme de cas d'utilisation relatif au test checker

La figure 2.4 ci-dessous représente le diagramme de cas d'utilisation relatif à un test checker.

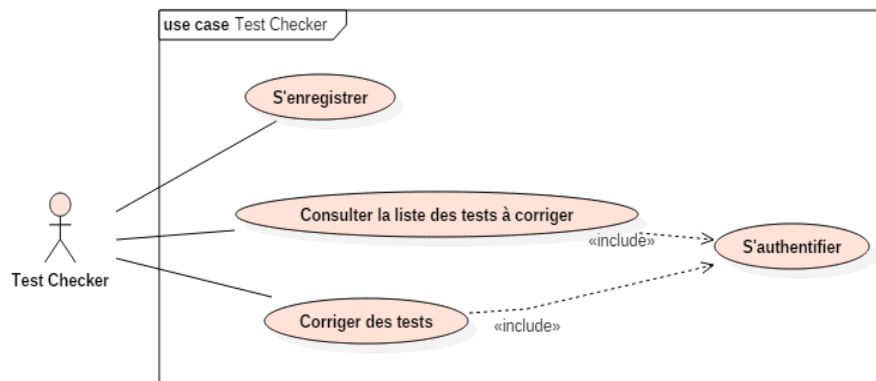


FIGURE 2.4 – Cas d'utilisation relatif au test checker

Ce diagramme de cas d'utilisation présente les différentes fonctionnalités qu'un test checker peut faire. Un test checker doit tout d'abord s'enregistrer sur la plateforme. Après que l'administrateur approuve sa demande, il peut accéder à son espace et consulter la liste des tests qu'il doit corriger.

II.2 Description de quelques scénarii

II.2.1 Scénario du cas d'utilisation "S'authentifier"

La figure 2.5 ci-dessous représente le diagramme de séquence système du scénario d'authentification pour un test taker :

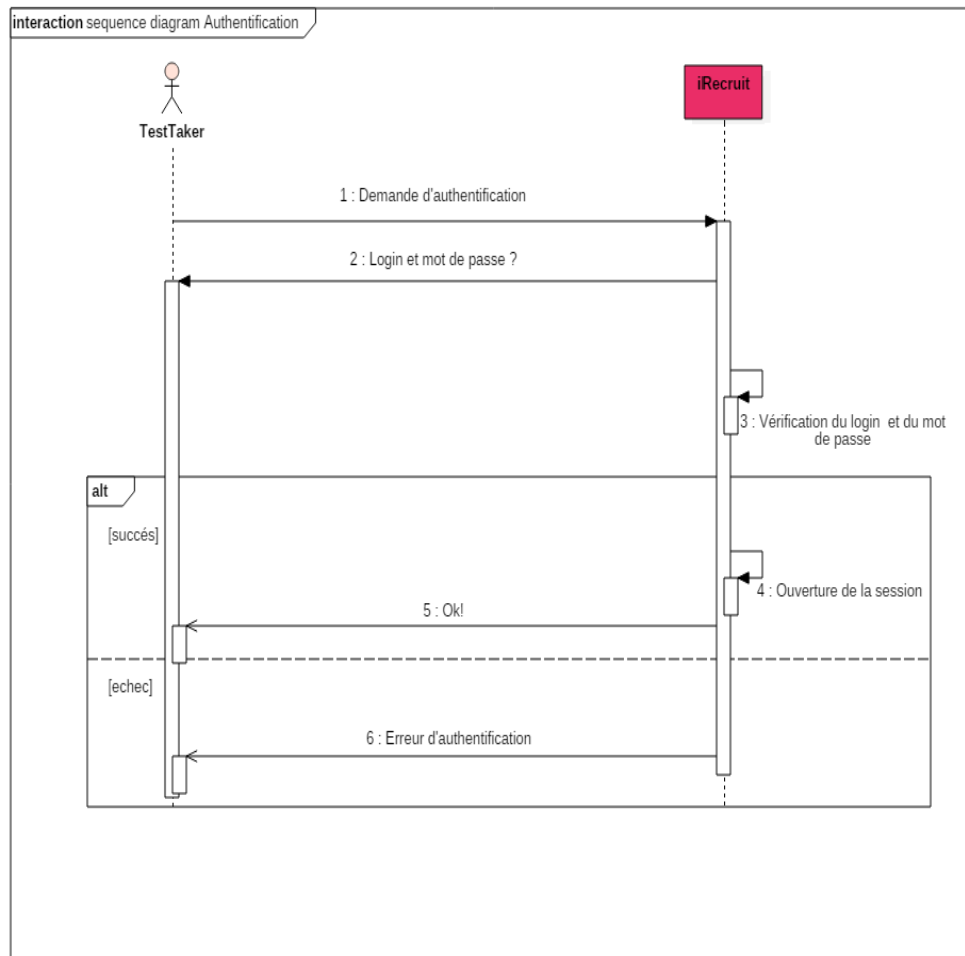


FIGURE 2.5 – Diagramme de séquence système d'authentification

Pour s'authentifier, un test taker doit saisir son login et son mot de passe, si les données saisies sont correctes alors une session sera ouverte pour lui et il sera redirigé automatiquement à son espace. Si les données sont erronées alors un message d'erreur apparaîtra demandant au test taker de saisir de nouveau le login et le mot de passe corrects.

II.2.2 Scénario du cas d'utilisation "Créer un test"

La figure 2.6 ci-dessous représente le diagramme de séquence système du scénario de création d'un nouveau test pour un test maker :

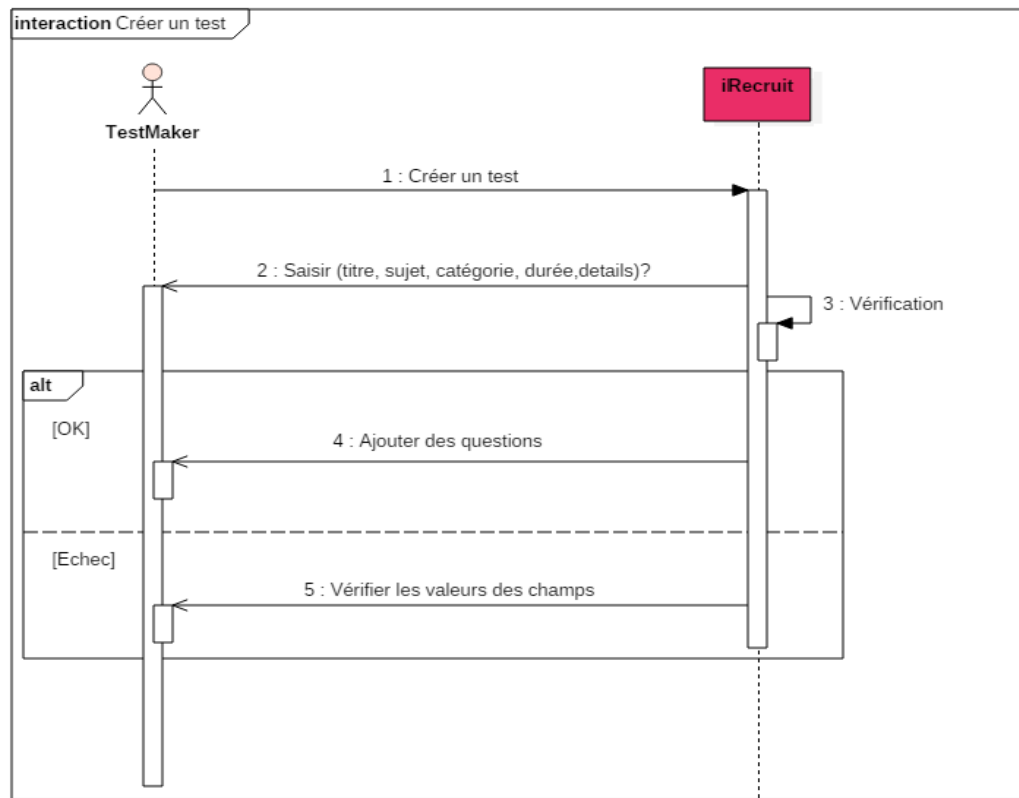


FIGURE 2.6 – Diagramme de séquence système de création de test

Après avoir être authentifier, un test maker peut aller dans son espace. Dans la section Créer un test, il doit saisir les différents champs nécessaires : le titre du test, choisir une catégorie et un sujet pour le test et enfin préciser sa durée et quelques informations. Si les données saisies sont correctes alors il peut commencer à ajouter les questions. Si les données sont erronées ou vides alors un message d’erreur apparaîtra demandant au test maker de vérifier les valeurs des champs.

Conclusion

Dans ce chapitre nous avons procédé à l’identification et la spécification des besoins de notre système. Dans le chapitre suivant, nous entamerons la partie de conception de notre système.

Chapitre 3

Conception

Dans ce chapitre, nous allons aborder la tâche la plus importante dans l'élaboration de ce travail, à savoir la tâche de conception. En effet, nous présentons, en premier lieu, l'architecture générale de notre application afin d'en extraire les différents modules qui la composent. Puis, nous détaillons chacun de ces modules conformément à la notation UML par la description des différents diagrammes de séquences et d'activités relatifs aux cas d'utilisation qui ont été exprimés dans le chapitre précédent.

I Conception architecturale de l'application

I.1 Architecture physique

Pour la réalisation de notre application, l'architecture trois tiers [N4] est retenue. Ce choix est justifié par le fait que ce type d'architecture :

- Offre une plus grande flexibilité/souplesse : cette flexibilité permet d'envisager une grande souplesse pour l'introduction de toutes nouvelles technologies.
- Garantit une sécurité accrue : avec une architecture trois tiers l'accès à la base n'est effectué que par le serveur applicatif. Ce serveur est le seul à connaître la façon de se connecter à cette base. Il ne partage aucune des informations permettant l'accès aux données, en particulier le login et le password de la base. Il est alors possible de gérer la sécurité au niveau de ce serveur applicatif, par exemple en maintenant la liste des utilisateurs avec leurs mots de passe ainsi que leurs droits d'accès aux fonctions du système.

- Procure de meilleures performances, étant donné le partage des tâches entre les différents serveurs.
- Réduit fortement les coûts de déploiement et d'administration : en effet, l'avantage principal d'une architecture trois tiers est la facilité de déploiement. L'application en elle-même n'est déployée que sur la partie serveur (serveur applicatif et serveur de base de données). Le client ne nécessite qu'une installation et une configuration minime. En effet il suffit d'installer un navigateur web compatible avec l'application pour que le client puisse accéder à l'application, ce navigateur étant par ailleurs souvent installé par défaut sur toutes les machines. Cette facilité de déploiement aura pour conséquence non seulement de réduire le coût de déploiement mais aussi de permettre une évolution régulière du système.

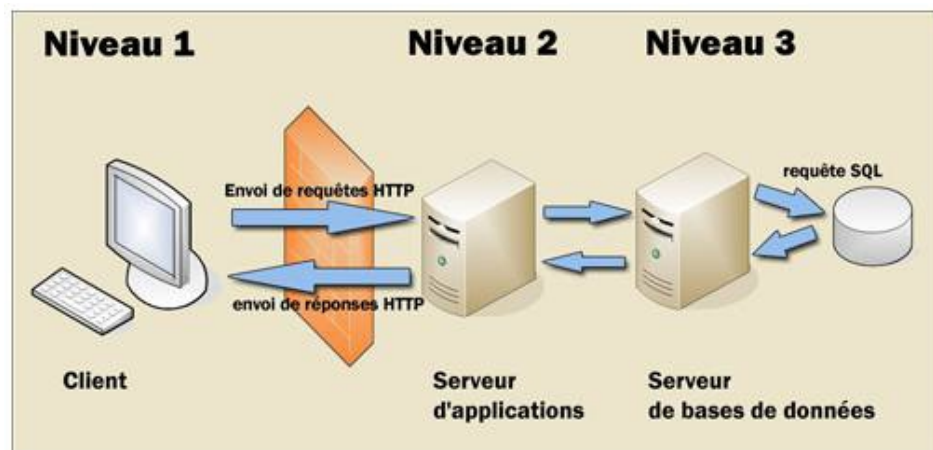


FIGURE 3.1 – Architecture trois tiers [N5]

L'architecture trois tiers comme le montre la figure 3.1, vise à séparer trois couches logicielles au sein d'une même application, à modéliser et à présenter cette application comme un empilement de trois couches dont le rôle est clairement défini :

- La présentation des données : correspondant à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur ;
- Le traitement métier des données : correspondant à la mise en oeuvre de l'ensemble des règles de gestion et de la logique applicative ;
- L'accès aux données persistantes : correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive.

I.1.1 Modélisation de l'architecture physique

Pour illustrer le déploiement de notre application, nous avons utilisé le diagramme de déploiement, comme le montre la figure 3.5, qui illustre la disposition physique des différents matériels(ou noeuds) et la répartition des composants au sein des noeuds :

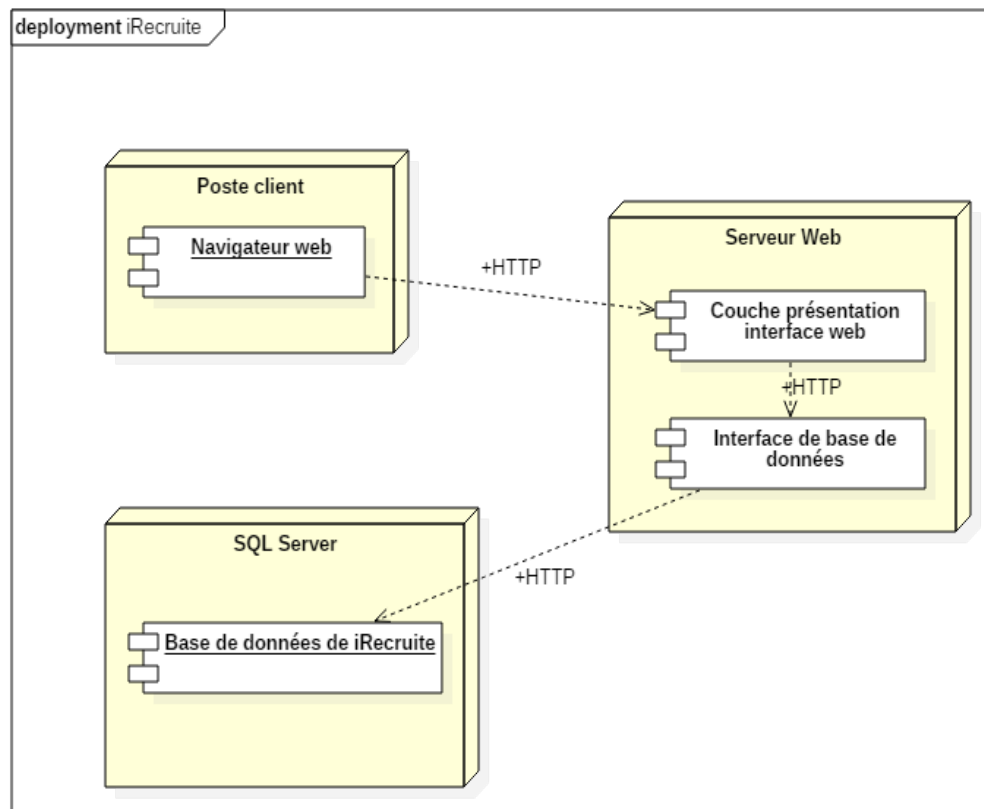


FIGURE 3.2 – Diagramme de déploiement du système

Notre diagramme de déploiement est constitué de trois nœuds principaux :

- Le poste client qui est composé d'un navigateur web, qui sert d'outil de communication entre les utilisateurs de notre système et le reste des nœuds. Les utilisateurs lancent leurs demandes sous forme de http request et en reçoivent des http response.
- Le serveur web qui regroupe deux composants :
 - La couche présentation qui regroupe l'ensemble des formulaires et interfaces à communiquer au navigateur par flux HTTP suite à une demande de l'utilisateur.

- Interface de la base données, qui est l'interface reliant notre application avec la base de données.
- Le serveur de base de données qui contient la base de données de notre application.

I.2 Architecture logique

Pour concevoir l'architecture logique de notre système nous optons pour l'architecture MVC[N6] (Modèle - Vue - Contrôleur) qui est une façon d'organiser une interface graphique d'un programme. Elle consiste à distinguer trois entités distinctes qui sont, le modèle, la vue et le contrôleur ayant chacun un rôle précis dans l'interface. MVC est un patron de conception très répandu pour réaliser des sites web. Ce patron de conception est une solution éprouvée et reconnue permettant de séparer l'affichage des informations, les actions de l'utilisateur et l'accès aux données. C'est un modèle qui a été conçu au départ pour des applications dites "client lourd", c'est-à-dire dont la majorité des données sont traitées sur le poste client (par exemple : un traitement de texte comme Word). MVC était tellement puissant pour ces applications "client lourd", qu'il a été massivement adopté comme modèle pour la création d'application web (dites "client léger"). Dans l'architecture MVC, les rôles des trois entités sont les suivants :

- modèle : données (accès et mise à jour),
- vue : interface utilisateur (entrées et sorties),
- contrôleur : gestion des événements et synchronisation.

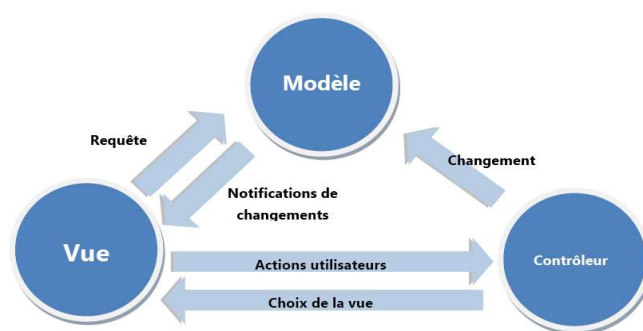


FIGURE 3.3 – Architecture MVC [N7]

II Conception détaillée

Dans cette partie nous allons entamer la description des détails conceptuels relatifs à notre plateforme iRecruit. Nous commençons par présenter les composants de notre solution via le diagramme package. Ensuite, nous détaillons chaque paquet par le diagramme approprié.

II.1 Diagramme de paquet

Le diagramme de paquetage est la représentation graphique des relations existant entre les paquets composant un système, dans le langage UML. Comme précédemment établi, nous avons choisi le modèle de conception MVC, donc nous avons trois paquets :

- Le paquet Modèles : ce paquet contient le package contextuel de la base de données, le package ViewModels et le package EntityManager. Dans le package entity-Manager, nous avons les classes suivantes :

- UserManager : cette classe contient des opérations liées aux administrateurs, test makers et test checkers.
- TakerManager : cette classe contient des opérations liées aux test takers.
- TestManager : cette classe contient des opérations portant sur la création des tests.

- Le paquet Vues : dans ce paquet, nous identifions les interfaces utilisateur (paquet CShtml) ainsi que les composants graphiques (packages CSS et Javascript).

-Le paquet Controleurs : dans ce paquet, nous avons plusieurs classes qui contiennent chacune un certain nombre d'opérations portant sur les modules de l'application.

La figure 3.4 présente les différents paquets pour le modèle, la vue et le contrôleur, décrit précédemment.

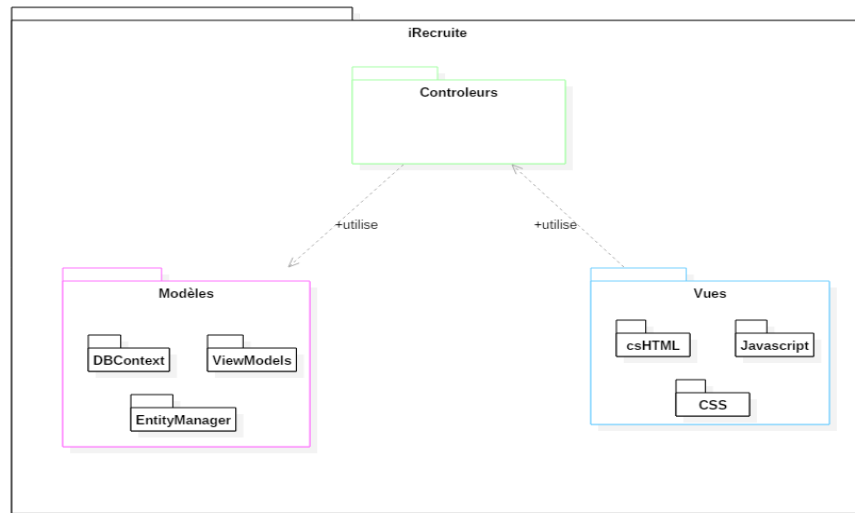


FIGURE 3.4 – Diagramme de paquet de iRecruit

II.2 Conception détaillée du Contrôleur

Le diagramme de classe de la figure 3.5 illustre les différents contrôleurs de notre application :

- **Account** : cette classe encapsule les opérations de gestion des comptes utilisateur.
- **Home** : cette classe contient des opérations liées à la direction des utilisateurs aux pages d'erreur et à la réception.
- **Library** : cette classe contient des opérations en relation avec l'espace bibliothèque des questions.
- **AdminOperations** : cette classe contient toutes les opérations que l'administrateur peut exécuter.
- **TestMaking** : comme son nom l'indique, cette classe contient les opérations que le test maker peut effectuer.
- **TestCheking** : cette classe contient des opérations liées au test checker.
- **TestTaking** : cette classe contient des opérations qui traitent la prise des tests par les test takers.

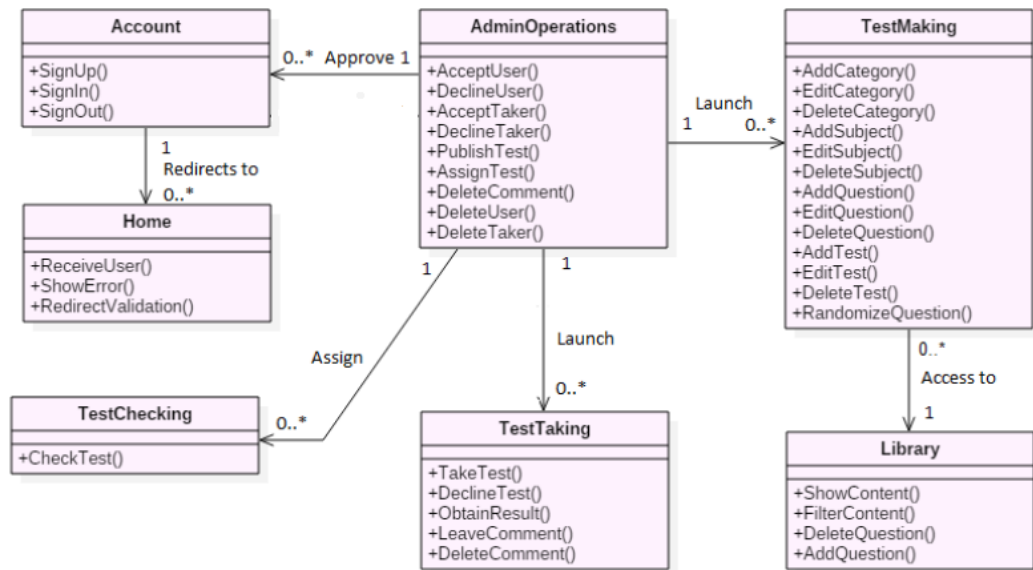


FIGURE 3.5 – Conception détaillée du paquet Contrôleur

II.3 Conception détaillée du paquet DBContext

Le package DbContext implémente les tables de base de données de notre plateforme comme indique dans la figure 3.6.

- **La table user** : cette table fait référence aux utilisateurs internes de Linedata. Un utilisateur a au moins un rôle et peut créer plusieurs catégories de test en comme étant un test maker et peut publier plusieurs tests en tant qu'administrateur.
- **Les tables category, subject et test** : chaque catégorie peut avoir plusieurs sujets et chaque sujet peut avoir plusieurs tests.
- **Les tables question, library, answer et Item** : les questions sont créées dans le cadre de la création des tests et seront ajoutées automatiquement dans la bibliothèque. Ainsi, un test peut contenir plusieurs questions et la bibliothèque contient une ou plusieurs questions. Chaque question lui correspond une réponse correcte.
- **Les tables taker, takerInterest et comment** : la table test taker se réfère aux candidats lors du recrutement. Chaque test taker doit avoir au moins un intérêt indiquant le type de sa candidature et peut également laisser un ou plusieurs commentaires.

- **Les tables publishing et assignChecking** : la table publishing est destinée à contenir des informations sur les tests publiés. La table AssignChecking est destinée à contenir des informations sur la correction des tests publiés. Par conséquent, un test publié pourrait être effectué par plusieurs candidats et être vérifié par un seul correcteur.
- **Les tables TestTaking, TakerAnswer et TakerItem** : la table TestTaking se réfère spécifiquement à la prise des tests, car elle contient des données sur chaque question prise et chaque test passé. Dans le cas où la question est un paragraphe ou une courte question, la table TakerAnswer contient la réponse et si la question est un bouton radio ou une case à cocher, la table TakerItem contient les réponses du test taker.

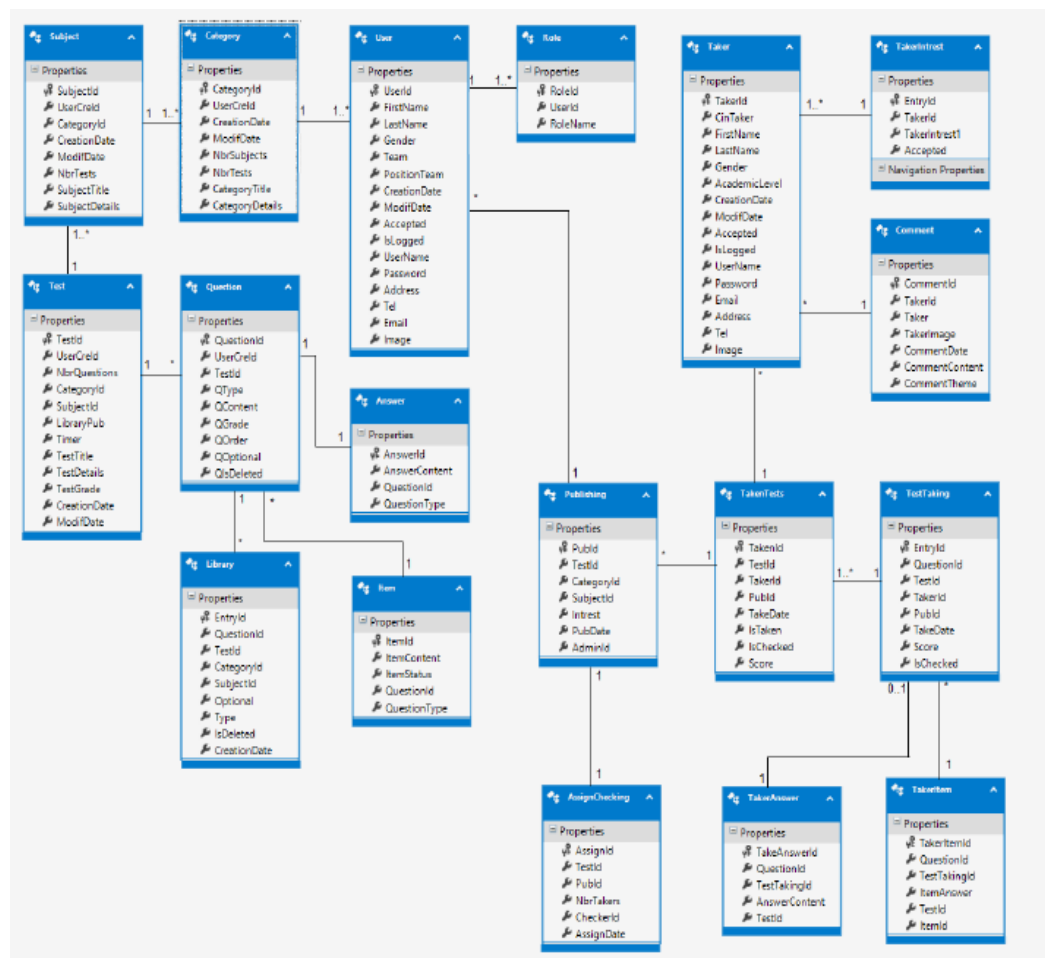


FIGURE 3.6 – Modèle relationnel de la base de données

II.4 Conception détaillée du paquet ViewModels

La figure 3.7 ci-dessous représente la conception détaillée du paquet ViewModels.

- **La classe Role** : cette classe se réfère aux différents rôles (administration, test maker et test checker) qui peuvent être attribués avec les applications aux utilisateurs internes.
- **La classe InternalUser** : cette classe générique représente les collaborateurs Linedata qui doivent avoir chacun un rôle spécifique.
- **La classe Administrator** : Cette classe représente chaque utilisateur interne dont le rôle est administrateur. Cet utilisateur peut accepter et refuser les demandes d'enregistrement des autres utilisateurs. Il peut également publier des tests et assigner des tests pour la vérification.
- **La classe Test maker** : Le créateur de test est un utilisateur interne dont le rôle est test maker. Cet utilisateur peut gérer des questions, des tests, des sujets et des catégories de test. Il peut également accéder l'espace bibliothèque des questions existantes.
- **La classe Test Checker** : Cet utilisateur est un utilisateur interne qui s'occupe de la vérification des tests. Le test checker peut vérifier plusieurs tests qui lui sont assignés.
- **La classe Test Taker** : C'est un candidat qui peut prendre ou refuser plusieurs tests et laisser ou supprimer des commentaires.
- **La classe Category** : une catégorie de test se compose de sujets multiples.
- **La classe Subject** : Cette classe est liée à la classe de Category, donc si la catégorie est supprimée, tous les sujets qui lui sont associés seront supprimés.
- **La classe Test** : cette classe concerne les tests de recrutement. Chaque test appartient à un sujet et à une catégorie spécifique et contient un certain nombre de questions. Si la catégorie associée ou le sujet est supprimé, le test sera supprimé automatiquement.

- **La classe Question** : Cette classe représente les questions des tests et ceux dans la bibliothèque. En supprimant le test, les questions du test ne seront pas supprimées.
- **La classe Library** : Cette classe représente l'archive des questions et si on supprime des questions de la bibliothèque, elles seront complètement supprimées.
- **La classe Comment** : Cette classe représente les commentaires laissés par les test takers qui peuvent aussi supprimer un ou plusieurs commentaires. L'administrateur a les privilèges de visualiser ces commentaires et de les supprimer.

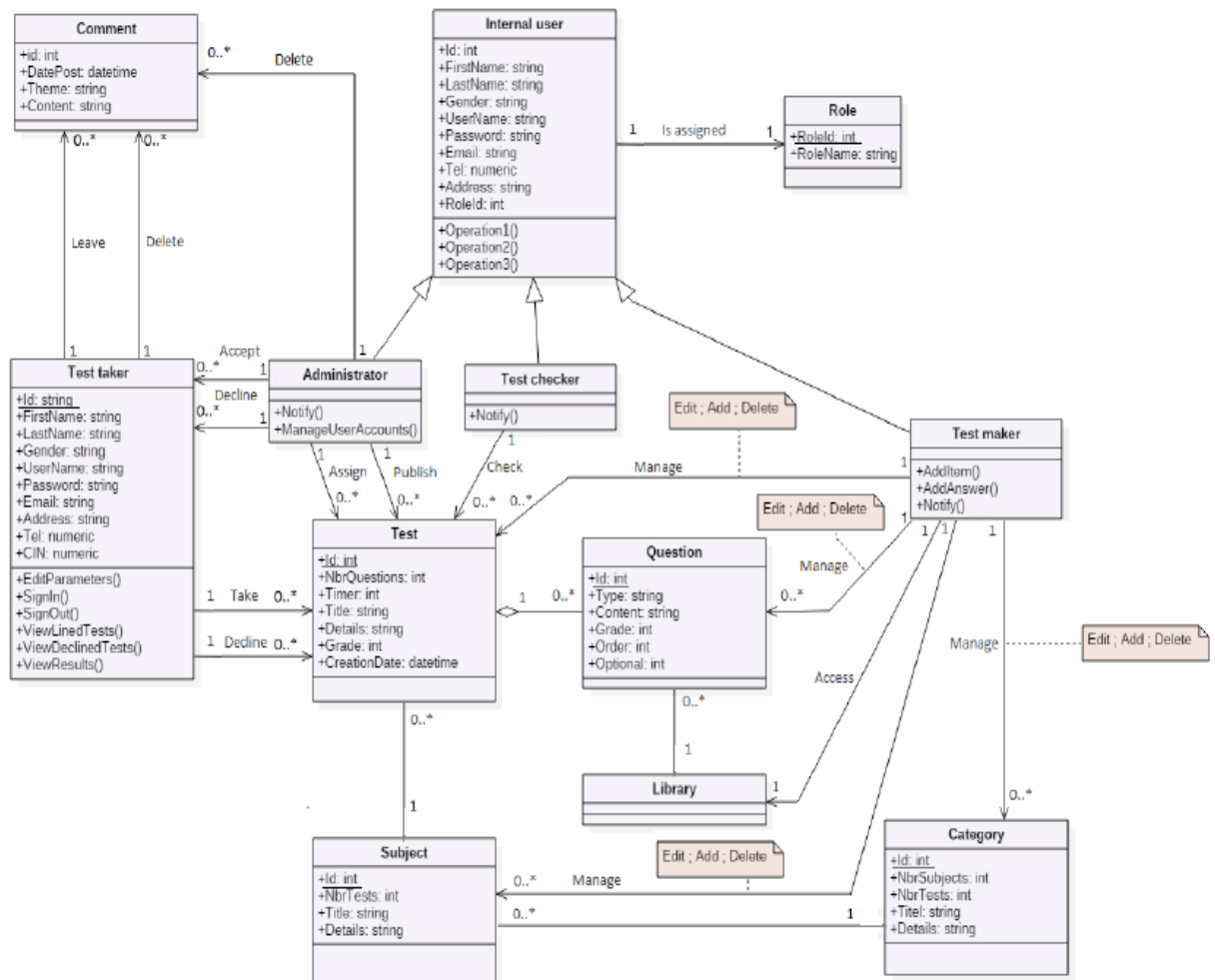


FIGURE 3.7 – Diagramme de classe du paquet ViewModels

III Description de quelques scénarii

III.1 Scénario "Créer un test"

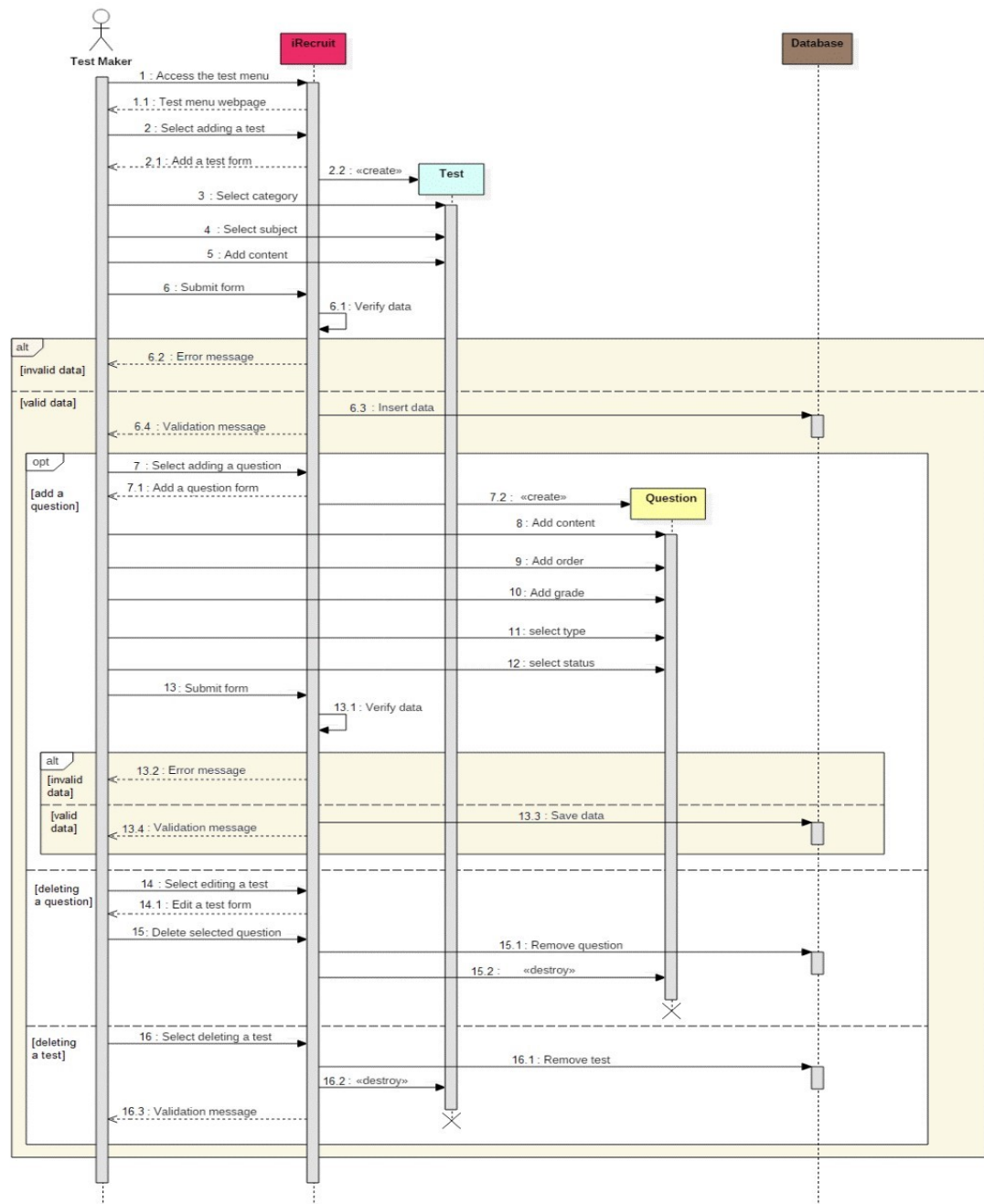


FIGURE 3.8 – Diagramme de séquence détaillé du scénario créer un test

La figure 3.8 ci-dessus représente le diagramme de séquence détaillé du scénario de création d'un nouveau test. Une fois authentifié, un test maker peut créer un nouveau test, y ajouter ou supprimer des questions. Il a la possibilité de supprimer ce test.

III.2 Scénario "Passer un test"

La figure 3.10 représente le diagramme d'activité du scénario passer un test. Ce diagramme illustre les actions et les décisions prises du test taker lorsqu'il passe un test. Il a la possibilité aussi de décliner le test.

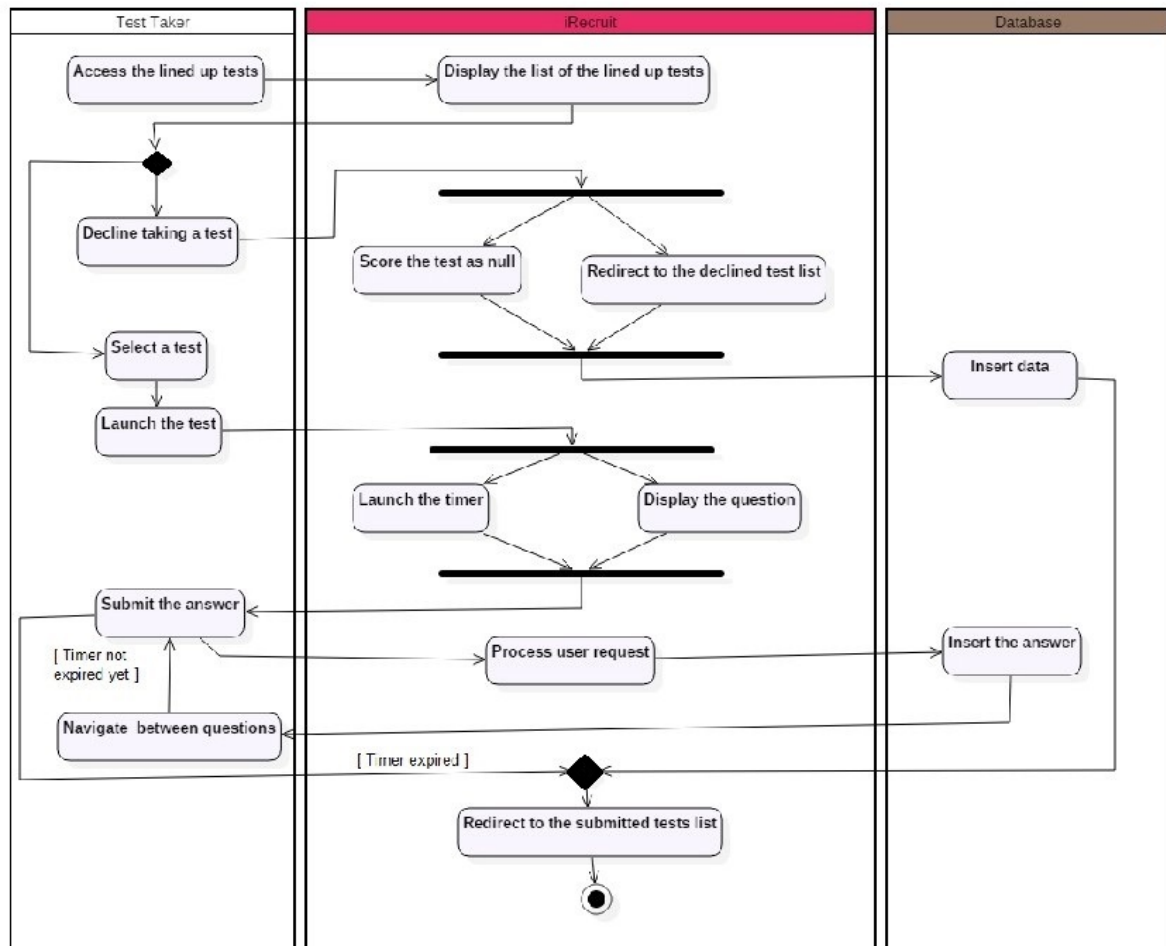


FIGURE 3.9 – Diagramme d'activités du scénario Passer un test

Conclusion

A travers ce chapitre, nous avons présenté notre conception de l'application. Nous avons fourni, dans un premier lieu, une conception globale de l'organisation de notre système. Ensuite, nous avons présenté la conception détaillée de l'application à travers les diagrammes UML. A présent, nous sommes capables d'entamer la partie réalisation.

Chapitre 4

Réalisation

C^E chapitre constitue le dernier volet du rapport ayant pour objectif d'exposer le travail achevé. Pour ce faire, nous allons présenter dans un premier temps l'environnement matériel et logiciel supportant notre application. Par la suite, nous présentons la plateforme de développement et les choix technologiques. Ensuite, nous allons passer en revue les différentes tâches réalisées à travers quelques interfaces homme-machine et un chronogramme récapitulatif qui décrit toutes étapes de mise en oeuvre de notre système.

I Environnements de travail

Tout au long de la réalisation de notre application, nous avons utilisé des logiciels et des langages de programmation bien particuliers.

I.1 Langages de programmation et framework utilisés

Dans cette partie, nous nous intéressons aux langages, aux bibliothèques et aux techniques de programmation utilisées tout au long de la réalisation de notre application en justifiant notre choix.

I.1.1 ASP.NET.MVC framework

ASP.NET MVC [N8] fournit une alternative au modèle ASP.NET Web Forms pour créer des applications Web. Le framework est un cadre de présentation léger et

hautement vérifiable et intégré aux fonctionnalités Asp.Net. La version Asp.Net.MVC 5 a été utilisée pour la mise en oeuvre de cette solution.

I.1.2 Entity framework

Entity Framework [N9], également appelé EF, est un mappeur objet-relationnel qui permet aux développeurs .NET de travailler avec des données relationnelles à l'aide d'objets spécifiques au domaine. Il élimine la nécessité d'accès aux données et des requêtes SQL du côté de l'application que les développeurs doivent généralement écrire. Nous avons utilisé, tout au long de ce travail, la version Entity framework 6.

I.1.3 C sharp

Le langage de programmation C# [N10] est un langage de programmation multi-paradigme englobant des disciplines de programmation forte, impérative, déclarative, fonctionnelle, générique, axée sur l'objet (basée sur la classe) et orientée vers les composants. Il est également livré avec une bibliothèque de classe riche qui facilite la mise en oeuvre de nombreuses fonctionnalités et qui est multi-plateforme et distribuée. La version C# 6.0 a été utilisée pour la mise en oeuvre de cette solution.

I.1.4 HTML5, Bootstrap CSS3, jQuery, JavaScript et AJAX pour la construction des interfaces riches

HTML5 [N11] a simplifié certaines balises afin d'alléger le code. Il introduit également un ensemble de nouvelles balises afin de donner plus de sémantique à nos pages. HTML 5 nous a permis une utilisation plus propre, code plus propre et nous pouvons, ainsi, éliminer la plupart des balises div et les remplacer par des éléments HTML 5 sémantiques.

Bootstrap CSS3 [N12] nous a permis d'automatiser certains effets visuels qui nécessitaient jusqu'à présent l'utilisation d'images, de scripts ou de modifications du code HTML : ombres portées, coins arrondis, opacité, arrières plans multiples, dégradés complexes, multi-colonage, effets textuels... L'avantage majeur et bien entendu une optimisation sur le temps de chargement des pages, car moins d'images, moins de code, moins de fichiers flash ect.

JavaScript [N13] est un langage de programmation Web distinct du HTML qui nous a permis d'ajouter de l'interactivité à nos pages Web en accédant directement aux éléments de la page HTML et en les manipulant. Il est, sans aucun doute, un des langages les plus populaires et les plus utilisés sur Internet, surtout qu'il fonctionne sur bon nombre de navigateurs différents.

jQuery [N14] est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web. Il nous a permis de mieux développer nos pages web puisqu'il est :

Stable et robuste, facile à apprendre et à utiliser, syntaxe claire et concise réduisant le nombre de lignes de code à écrire.

AJAX [N15] (acronyme d'asynchronous Javascript and XML) est une méthode de création d'applications interactives pour le Web qui traitent immédiatement les demandes des utilisateurs. Ajax combine plusieurs outils de programmation, y compris JavaScript, HTML dynamique (DHTML), langage de balisage extensible (XML), feuilles de style en cascade (CSS) et bien d'autres encore. AJAX est utilisé dans l'application pour assurer une communication supplémentaire avec les contrôleurs au besoin.

I.2 Environnements de développement logiciel

Dans cette section, nous décrivons les outils logiciels utilisés pour le développement et l'achèvement des travaux pour ce projet.

I.2.1 Visual studio

Visual Studio [N16] est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du framework .NET, qui fournit un accès à des technologies clés simplifiant le

développement d'applications web ASP et de services web XML grâce à Visual Web Developer.

I.2.2 SQL server

Microsoft SQL Server [N17] est un système de gestion de base de données relationnelle basé sur SQL conçu pour être utilisé dans des applications d'entreprise. Nous avons utilisé Microsoft SQL server express 2016 avec une taille de base de 10 GB maximum pour le développement de cette application.

I.2.3 SQL server management studio

SQL Server Management Studio [N18], également connu sous le nom de SSMS, est un environnement intégré pour la gestion de toute infrastructure SQL, de SQL Server à la base de données SQL. Le logiciel SSMS a été utilisé pour interagir avec le serveur lors de la mise en oeuvre de cette solution.

II Le Backlog du produit

Nous commençons par définir le backlog qui consiste en une liste de tâches à réaliser durant ce stage. Ces tâches, exprimées sous forme de d'un tableur excel, sont priorisées par le Product Owner ce qui permet d'établir un ordre à respecter lors de l'amélioration de l'application. La figure 4.1 montre le backlog du produit :

ID	Status	Version	Tests Status	Priority	Replication Details	Observed behaviour	Expected behaviour
12	Done	Version 1.4	Passed	Must	1-log in with the same checker in different browsers 2-open the tests section 3-open the unchecked sub-section 4-press on the button "Check these tests" 5-press on the button "correct this test" 6-For the same test, each checker in a different session add a	the sum of the two grades affected	we shouldn't allow the same checker to open different sessions and correct
16	In progress		failed	Must	1-log in as a test taker 2-open the tests section 3-open the lined up sub-section 4-pick up a test 5-press on "take the test" button 6-while taking the test, open a second session with the same link 7-close the first session and continue taking the test in the second one	a second session is opened with another timer and the candidate can continue taking the test normally	the taker shouldn't be able to open two sessions of t
18	Done	Version 1.4	Passed	Should	1-log in with a maker account 2-press on the tests section 3-pick a test 4-press on the edit button 5-press on the questions sub-section	message is displayed: the order is al	the question should be added in the deleted Orc

FIGURE 4.1 – Backlog du produit

III Déroulement des Sprints

Avant de présenter les différents sprints qui composent notre backlog du produit, nous évoquons le planning de notre travail qui s'est étalé sur une période de six semaines. Tout au long de ce chapitre, nous présentons une vue dynamique à travers les interfaces hommes machine et pour terminer nous présentons les tests effectués à la fin de chaque sprint. Durant cette période, le travail a été réparti comme le montre la figure 4.2 :

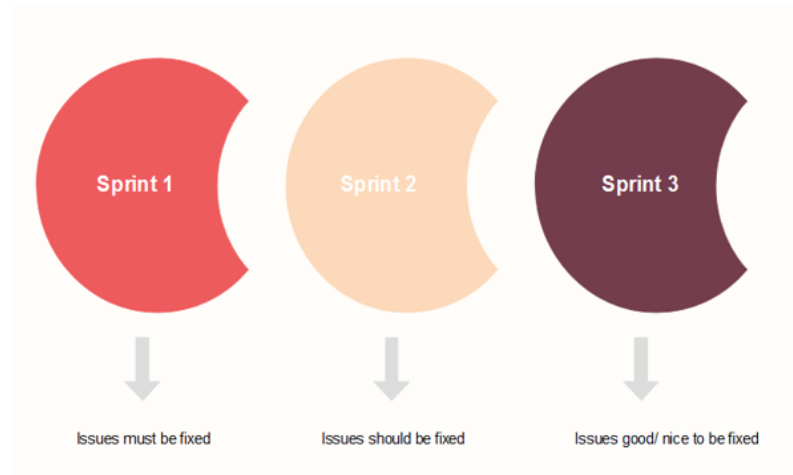


FIGURE 4.2 – Décomposition en sprints

La décomposition du projet en sprints a été décrite par le backlog du produit. Il s'agit à chaque étape, de faire une réunion avec les collaborateurs pour :

- Enoncer les objectifs d'un sprint (le produit du sprint),
- Proposer des estimations temporelles en nombre de jour et décider de l'ordre de réalisation des tâches de chaque sprint,
- Valider la réalisation à la fin du sprint en effectuant une démonstration du travail,
- Modifier éventuellement certains objectifs définis au départ en fonction de nouvelles requêtes.

III.1 Interfaces Homme-machine

Tout utilisateur de notre plateforme aura accès à la page d'accueil, qui lui permet soit de s'enregistrer soit de s'authentifier. La figure 4.3 ci-dessous représente la page d'accueil de iRecruit.

La figure 4.4 ci-dessous représente la page d'authentification via laquelle les utilisateurs de iRecruit doivent s'authentifier afin d'accéder à leurs espaces.

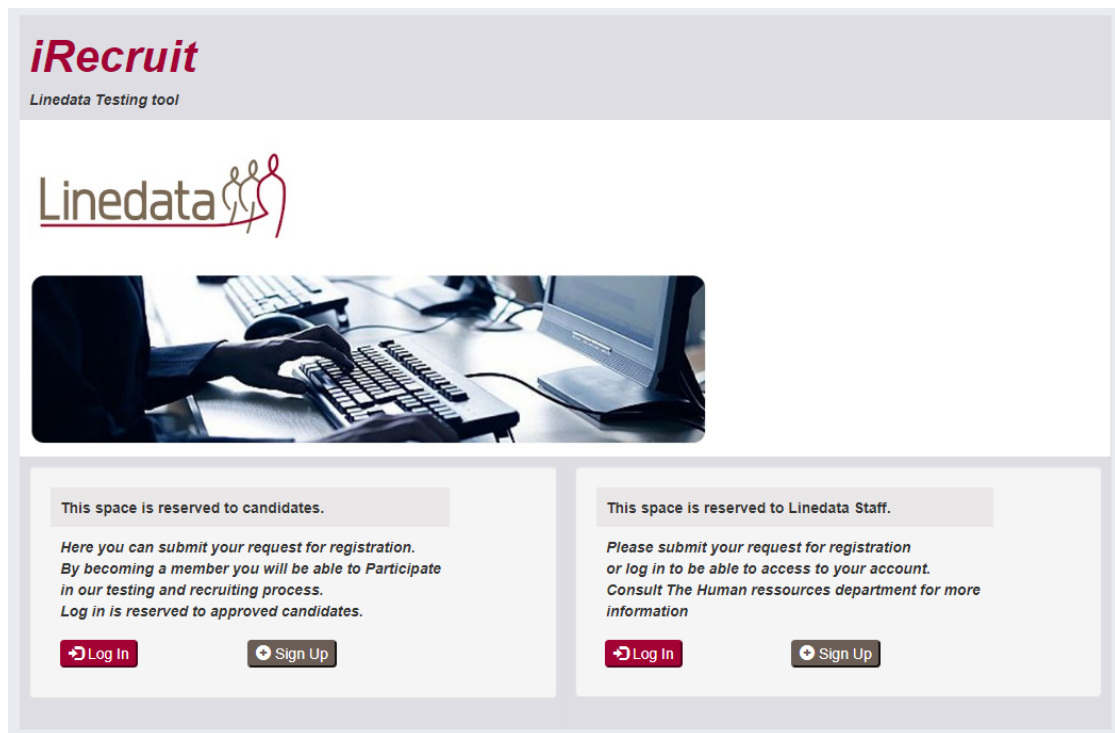


FIGURE 4.3 – Page d'accueil de iRecruit

The image shows a 'Sign In' page. At the top, the text 'Sign In' is centered. Below it, there are two input fields: 'User' and 'Password'. At the bottom, there are two buttons: a grey 'Reset' button and a red 'Sign In' button. To the right of the buttons, the text 'First Page' is displayed.

FIGURE 4.4 – Page d'authentification

La figure 4.5 ci-dessous représente le dashboard d'un test maker.

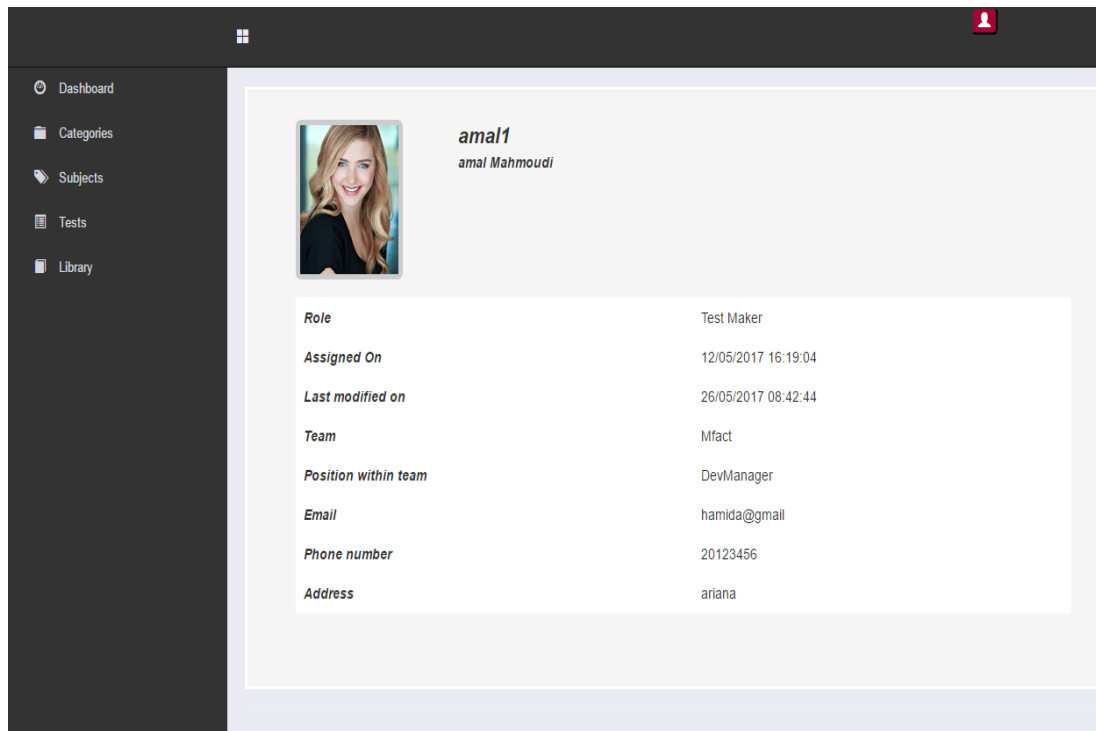


FIGURE 4.5 – Dashboard d'un test maker

La figure 4.6 ci-dessous représente l'interface via laquelle le test maker peut ajouter des nouvelles catégories.

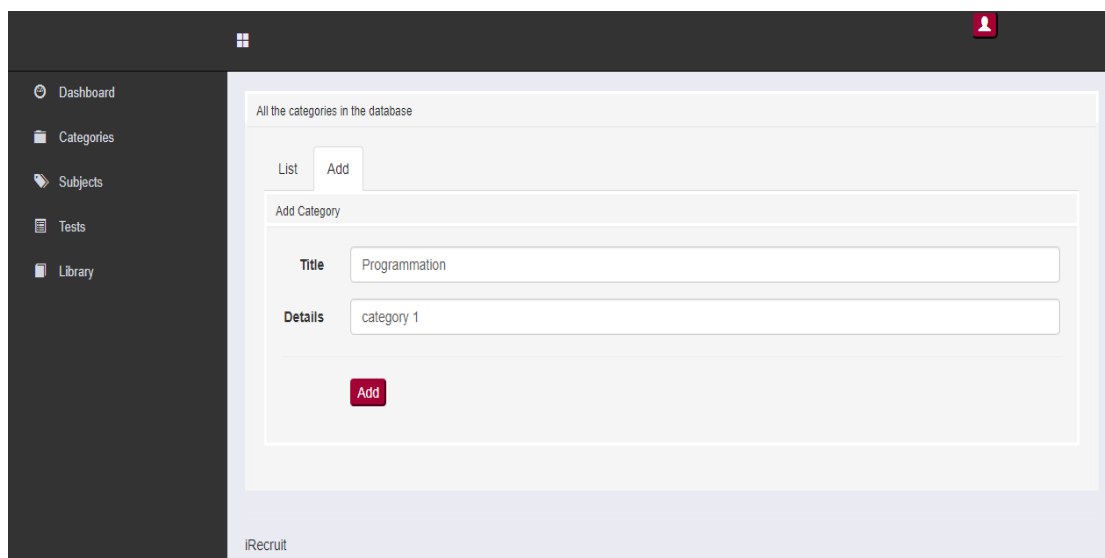
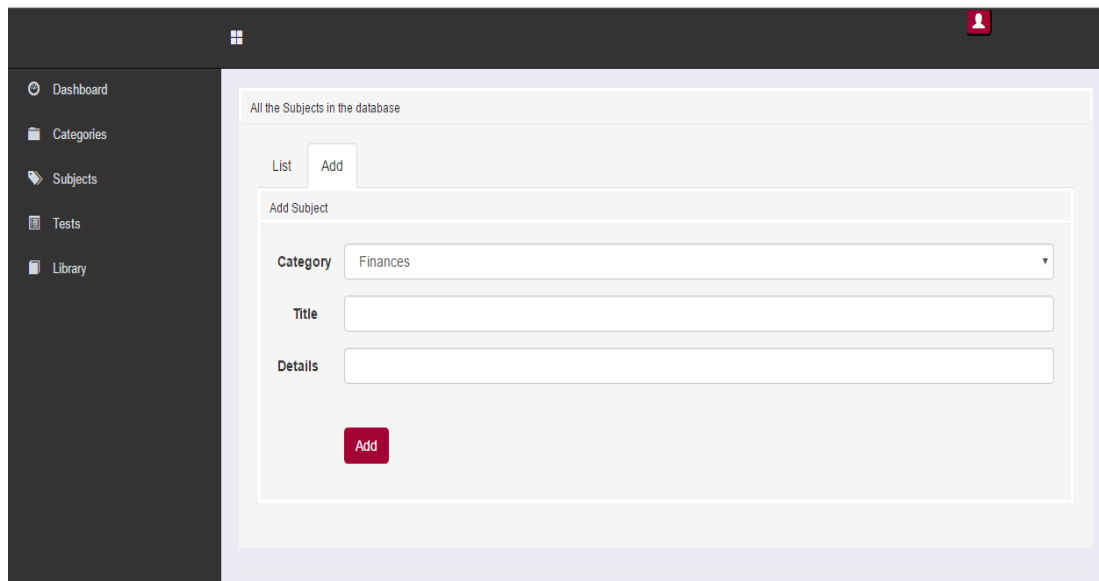


FIGURE 4.6 – Ajouter une nouvelle catégorie

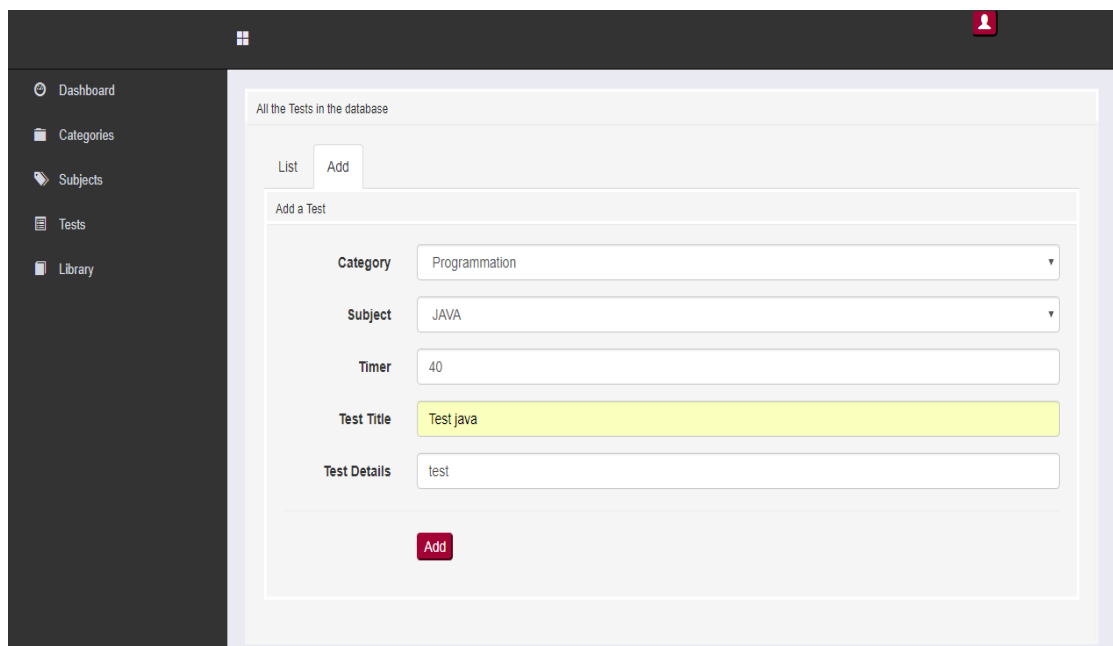
La figure 4.7 ci-dessous représente l'interface via laquelle le test maker peut ajouter des nouveaux sujets.



The screenshot shows a web application interface with a dark sidebar on the left containing navigation links: Dashboard, Categories, Subjects, Tests, and Library. The main content area is titled 'All the Subjects in the database' and features a 'List' button and an 'Add' button. Below this is a form titled 'Add Subject' with the following fields: a 'Category' dropdown menu set to 'Finances', a 'Title' text input field, and a 'Details' text input field. A red 'Add' button is positioned at the bottom of the form.

FIGURE 4.7 – Ajouter un nouveau sujet

La figure 4.8 ci-dessous représente l'interface via laquelle le test maker créer un nouveau test.



The screenshot shows the same web application interface as Figure 4.7, but with the 'Add Test' form displayed. The sidebar remains the same. The main content area is titled 'All the Tests in the database' and has 'List' and 'Add' buttons. The 'Add a Test' form includes: a 'Category' dropdown menu set to 'Programmation', a 'Subject' dropdown menu set to 'JAVA', a 'Timer' text input field with the value '40', a 'Test Title' text input field with the value 'Test java' (highlighted in yellow), and a 'Test Details' text input field with the value 'test'. A red 'Add' button is at the bottom of the form.

FIGURE 4.8 – Créer un test

La figure 4.9 ci-dessous représente l'interface via laquelle le test maker peut créer des questions.

The screenshot shows the 'Test content Management' interface. On the left is a dark sidebar with navigation links: Dashboard, Categories, Subjects, Tests, and Library. The main area has two tabs: 'New Question' (active) and 'Generate Questions'. Below the tabs is a 'Test Content' section. It contains a 'Test' dropdown set to 'Java advanced', an 'Order' input field with '1', a 'Question' text area containing 'What is a JVM?', a 'Grade' input field with '3', a 'Type' dropdown set to 'Check boxes', and a 'Status' dropdown set to 'Must have'. At the bottom are three buttons: 'Reset' (red), 'Confirm' (green), and 'Back to list' (dark grey).

FIGURE 4.9 – Créer question

La figure 4.10 ci-dessous représente l'interface via laquelle le test maker peut ajouter les différentes réponses pour une question à choix multiples.

The screenshot shows the interface for adding multiple-choice answers. The question title is '1 . What is a JVM ?'. There is an 'Item' input field with a green '+' icon and a red 'X' icon to its right. Below the input field are 'Correct' and 'Wrong' buttons. A list of three items is shown: '1 . it is a java code side component', '2 . enables machines to run java programs' (which is checked with a green checkmark), and '3 . short for java virtual modal'. To the right of each item is a red 'X' icon. At the bottom right is a 'Finish' button with a green checkmark.

FIGURE 4.10 – Question à choix multiples

La figure 4.11 ci-dessous représente l'interface via laquelle le test maker peut ajouter la réponses à une question de type paragraphe.

La figure 4.12 ci-dessous représente la librairie des questions.

FIGURE 4.11 – Question de type paragraphe

FIGURE 4.12 – Librairie des questions

La figure 4.13 ci-dessous représente l'interface permettant au test maker d'ajouter une question depuis la librairie à un test.

La figure 4.14 ci-dessous représente l'interface permettant au test maker de consulter la liste de ses tests.

FIGURE 4.13 – Ajouter question

Test Id	Test Title	Details	Creation Date
52	Java Basics	Testing java concepts.	26/05/2017 14:20:42
53	Bonds : Basics	Questions about bonds	26/05/2017 14:30:19
54	test1	testing basics	26/05/2017 15:43:23
55	Java advanced	Testing advanced java programming skills	29/05/2017 09:02:48

FIGURE 4.14 – Liste des tests créés

La figure 4.15 ci-dessous représente l'interface permettant au test maker de visualiser le contenu d'un test créé et/ou de le modifier.

La figure 4.16 ci-dessous représente l'interface permettant à l'administrateur de iRecruit de visualiser les tests créés par les test makers afin décider s'ils sont prêts à publier ou non, ou de les supprimer.

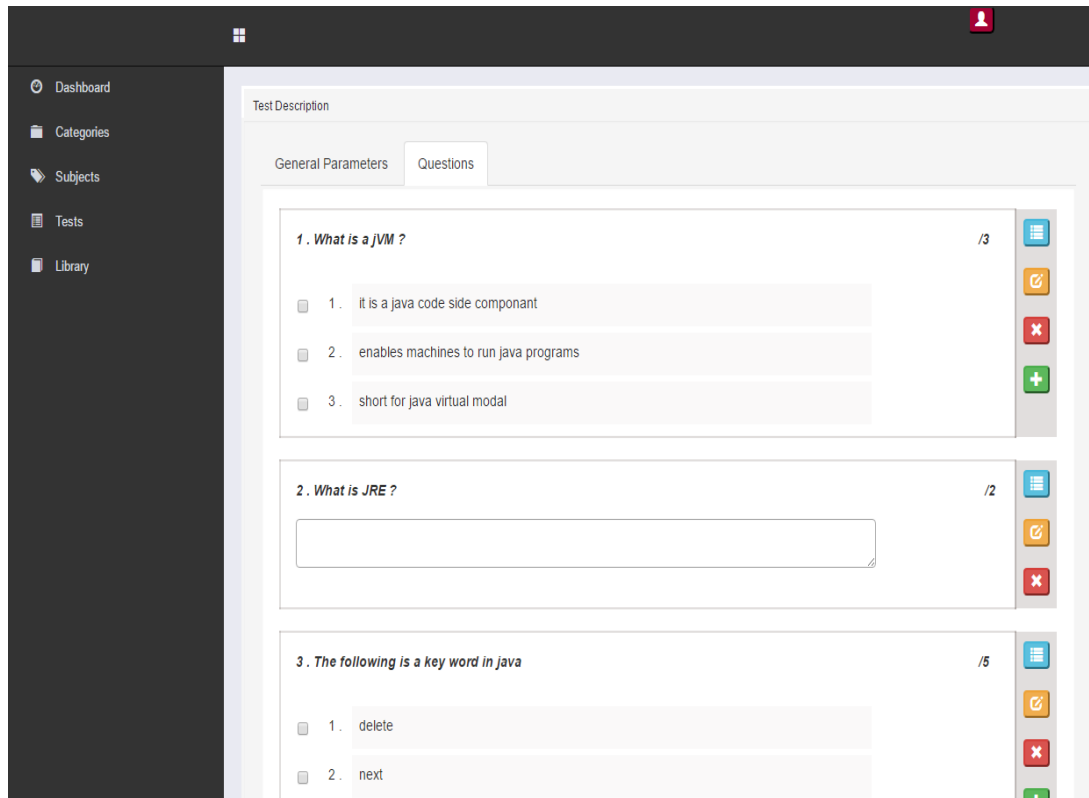


FIGURE 4.15 – Test créé

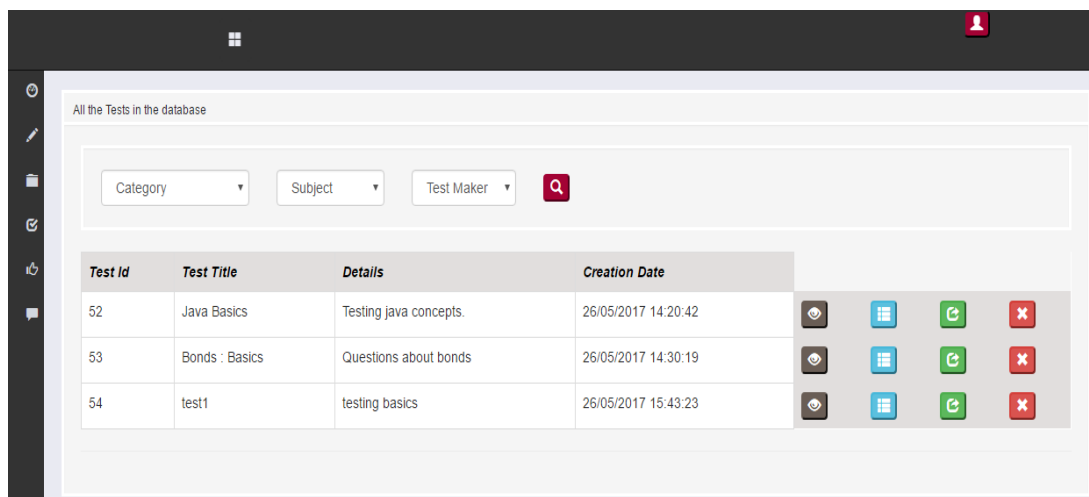


FIGURE 4.16 – Liste de tous les tests

La figure 4.17 ci-dessous représente l'interface permettant à l'administrateur de publier un test et choisir pour quelle catégorie de candidats il est destiné.

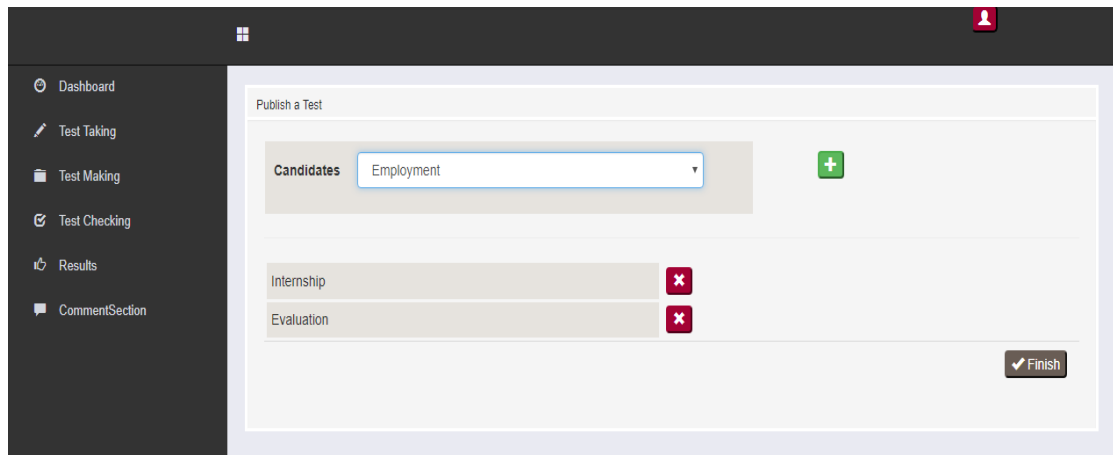


FIGURE 4.17 – Publier un test

Les figures 4.18 et 4.19 ci-dessous représentent les interfaces via lesquelles l'administrateur approuve les demandes d'inscription des autres utilisateurs des iRecruit.

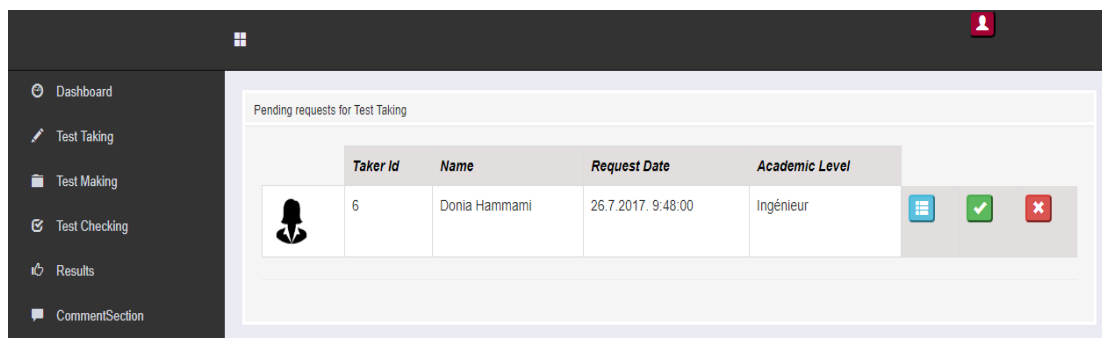


FIGURE 4.18 – Approuver demandes d'inscription des test takers

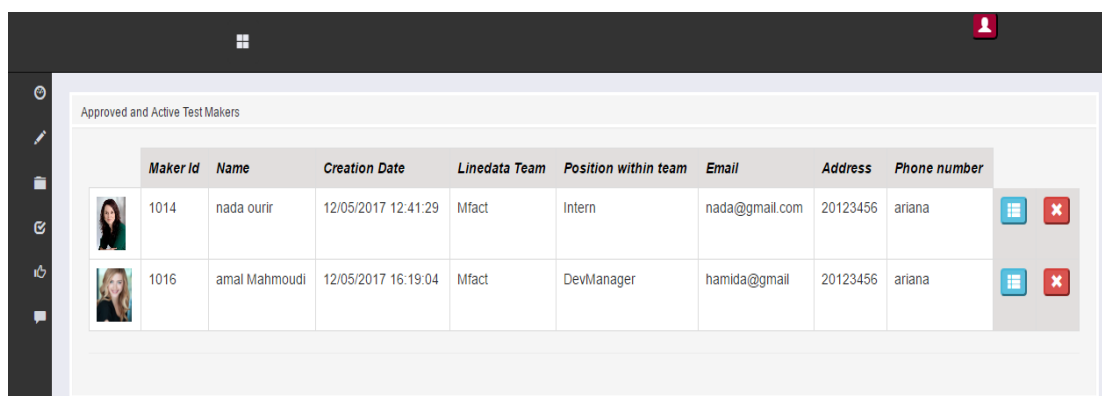
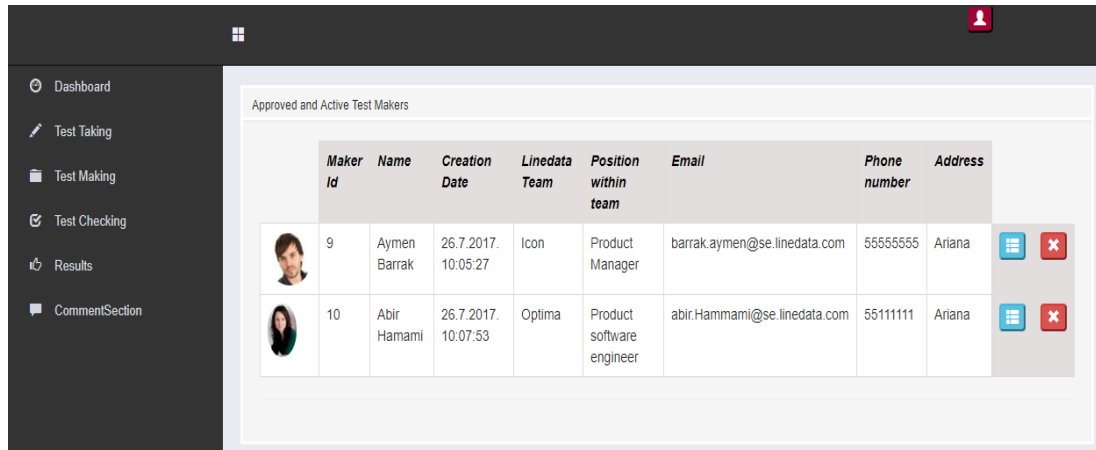


FIGURE 4.19 – Approuver demandes d'inscription des test makers

La figure 4.20 ci-dessous représente l'interface via laquelle l'administrateur consulte la liste des test checkers approuvés.









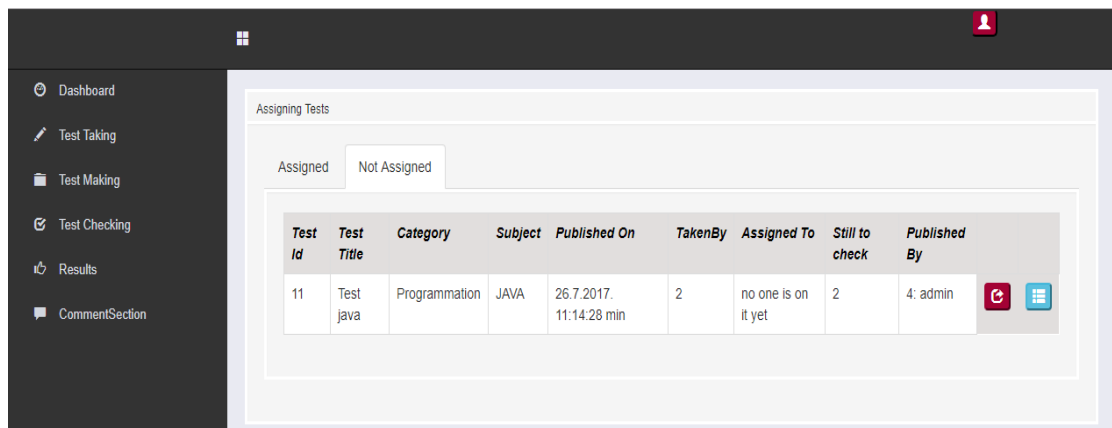
	Maker Id	Name	Creation Date	Linedata Team	Position within team	Email	Phone number	Address	
	9	Aymen Barrak	26.7.2017. 10:05:27	Icon	Product Manager	barrak.aymen@se.linedata.com	55555555	Ariana	 
	10	Abir Hamami	26.7.2017. 10:07:53	Optima	Product software engineer	abir.Hammami@se.linedata.com	55111111	Ariana	 

FIGURE 4.20 – Liste des test checkers

Les figures 4.21 et 4.22 ci-dessous représentent les interfaces via lesquelles l'administrateur assigne un test à corriger à un test checker.





Test Id	Test Title	Category	Subject	Published On	TakenBy	Assigned To	Still to check	Published By	
11	Test java	Programmation	JAVA	26.7.2017. 11:14:28 min	2	no one is on it yet	2	4: admin	 

FIGURE 4.21 – Assigner un test checker

Assign

Checker	User Name
	manwa

Test Id: 11

Number of candidates: 2

Checker Id: 11

Add

FIGURE 4.22 – Choisir un test checker

Les figures 4.23 et 4.24 ci-dessous représentent les interfaces via lesquelles l'administrateur peut consulter les résultats des test takers.

All The Approved TestCheckers

Test Id	Test Title	Category	Subject	Number of questions	Taker Id	Taken On	Result
11	Test java	Programmation	JAVA	3	8	27.7.2017. 8:50:51	6,00/6,50

FIGURE 4.23 – Liste des resultats obtenus de tous les tests

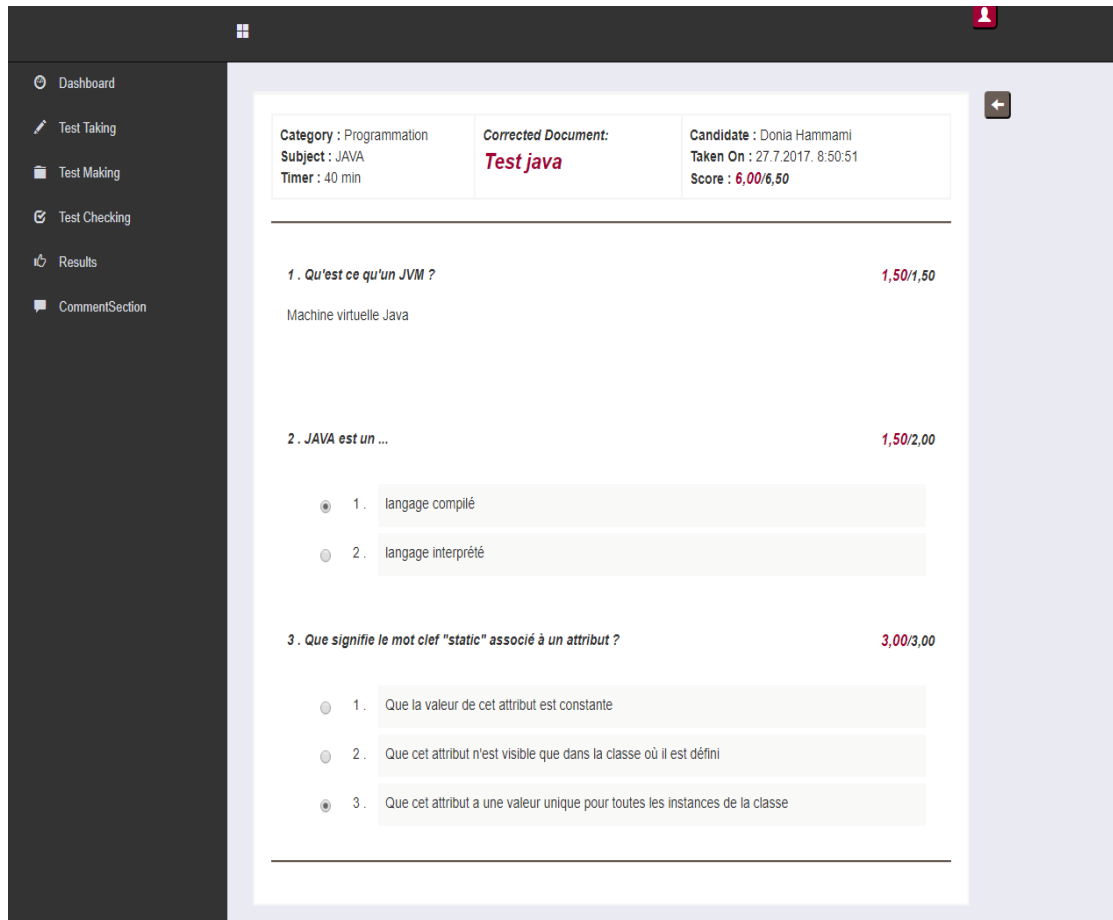
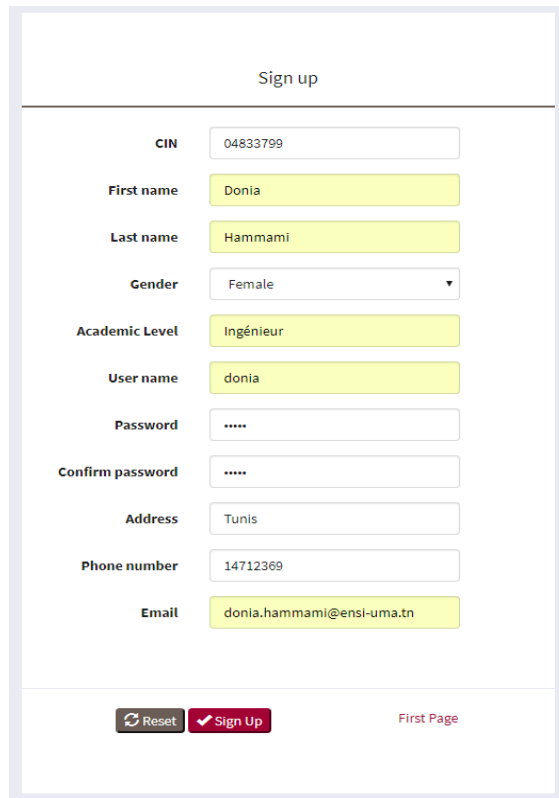


FIGURE 4.24 – Test corrigé d'un candidat

La figure 4.25 ci-dessous représente la page d'inscription des candidats sur iRecruit.

La figure 4.26 ci-dessous représente la liste des tests qu'un test taker peut les passer ou bien les décliner.



Sign up

CIN: 04833799

First name: Donia

Last name: Hammami

Gender: Female

Academic Level: Ingénieur

User name: donia

Password: ****

Confirm password: ****

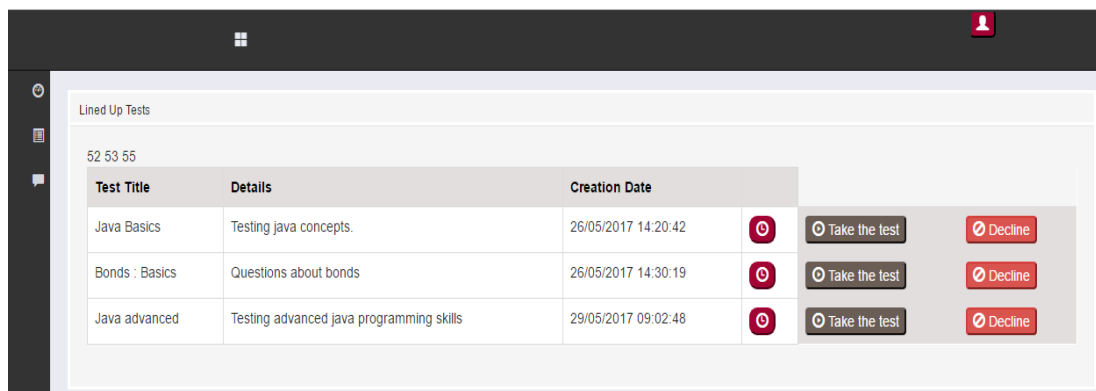
Address: Tunis

Phone number: 14712369

Email: donia.hammami@ensi-uma.tn

Reset Sign Up First Page

FIGURE 4.25 – Page d'inscription des test takers



Lined Up Tests

52 53 55

Test Title	Details	Creation Date		
Java Basics	Testing java concepts.	26/05/2017 14:20:42		Take the test Decline
Bonds : Basics	Questions about bonds	26/05/2017 14:30:19		Take the test Decline
Java advanced	Testing advanced java programming skills	29/05/2017 09:02:48		Take the test Decline

FIGURE 4.26 – Page d'inscription des test takers

La figure 4.27 ci-dessous représente l'interface permettant à un test taker de répondre aux questions du test.

La figure 4.28 ci-dessous représente l'interface permettant à un test taker laisser un commentaire pour l'administrateur de iRecuite.

Candidate :
Donia
Hammami

Category : Programmation
Subject : JAVA
Test : *Test java*

Duration : 40 min
Marked : 6,50 pts

0 33

1 2 3

3. /3,00

Que signifie le mot clef "static" associé à un attribut ?
(Optional Question)

- 1. Que la valeur de cet attribut est constante
- 2. Que cet attribut n'est visible que dans la classe où il est défini
- 3. Que cet attribut a une valeur unique pour toutes les instances de la classe

Finish

FIGURE 4.27 – Passer test

Dashboard
Tests
Comment section

Comments

My Comments Add a Comment

nada ourir

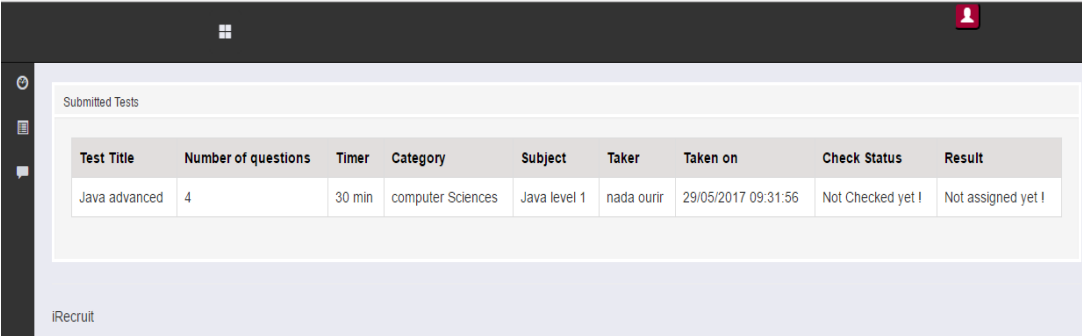
Title : About sql tests

Content : I think adding the PL/SQL questions on the basic tests is too much

Add Comment

FIGURE 4.28 – Ajouter un commentaire

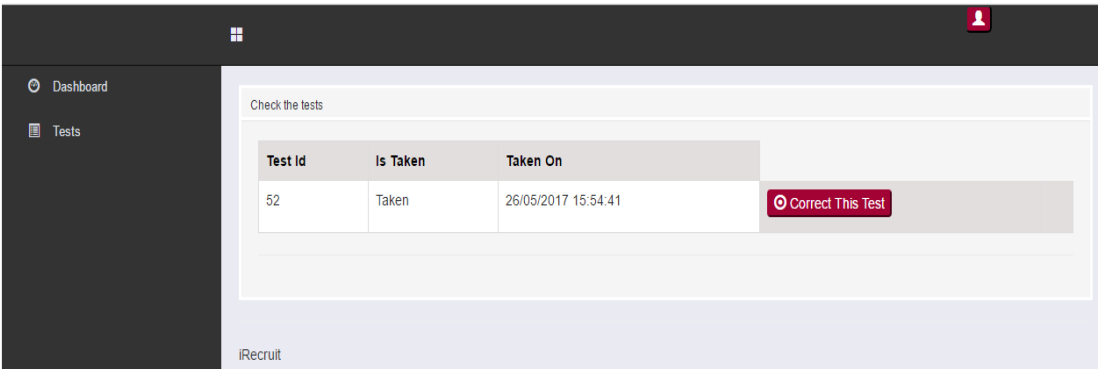
La figure 4.29 ci-dessous représente l'interface permettant un test taker de vérifier si ses tests sont corrigés ou non.



Test Title	Number of questions	Timer	Category	Subject	Taker	Taken on	Check Status	Result
Java advanced	4	30 min	computer Sciences	Java level 1	nada ourir	29/05/2017 09:31:56	Not Checked yet !	Not assigned yet !

FIGURE 4.29 – Vérifier état du test

La figure 4.30 ci-dessous représente l'interface permettant un test checker de consulter la liste des tests qui lui sont assignés afin de les corriger.



Test Id	Is Taken	Taken On
52	Taken	26/05/2017 15:54:41

FIGURE 4.30 – Liste des tests à corriger

La figure 4.31 ci-dessous représente l'interface permettant un test checker de corriger un test.

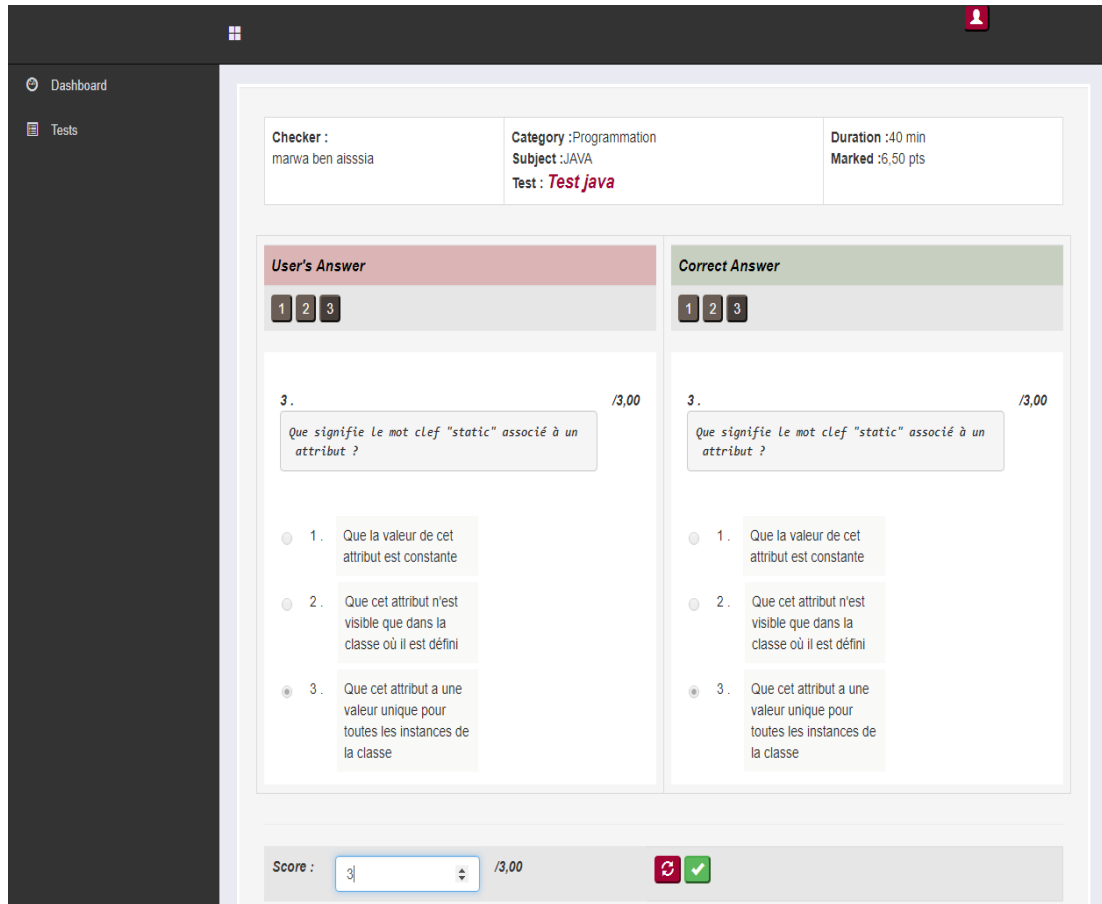


FIGURE 4.31 – Correction d'un test

III.2 Tests effectués

III.2.1 Tests du sprint 1

Les défauts de haute priorité, "issues must be fixed", sont tous corrigés. Nous avons effectués les jeux de test nécessaires sur les améliorations apportées sur iRecruit afin de s'assurer de son bon fonctionnement.

III.2.2 Tests du sprint 2

Les défauts de priorité importante, "issues should be fixed", sont corrigés. Nous avons effectués des jeux de test sur les modifications apportées à l'application. Nous avons effectué aussi des tests de régression sur iRecruit, pour s'assurer que des nouveaux défauts n'ont pas été introduits ou découverts dans des parties non modifiées de

l'application.

III.2.3 Tests du sprint 3

Les défauts de priorité basse, "issues good/nice to be fixed", sont tous corrigés. Nous avons effectués des jeux de test sur les nouvelles modifications apportées à l'application. Nous avons effectué de nouveau des tests sur toutes les améliorations ou les modifications faites ainsi que des tests de régression.

Après le déploiement de l'application sur le serveur Linedata, nous avons retesté encore une fois iRecrute, cela nous a permis de découvrir de nouveaux défauts. Nous avons apporté les modifications nécessaires afin de résoudre ces anomalies et nous avons effectués des tests de regression afin de s'assurer du bon fonctionnement de iRecruit.

IV Chronogramme

Ce travail a été réalisé durant une période de six semaines. La répartition des tâches durant notre stage est illustrée par le diagramme de Gantt réel de la figure 4.32 ci-dessous :

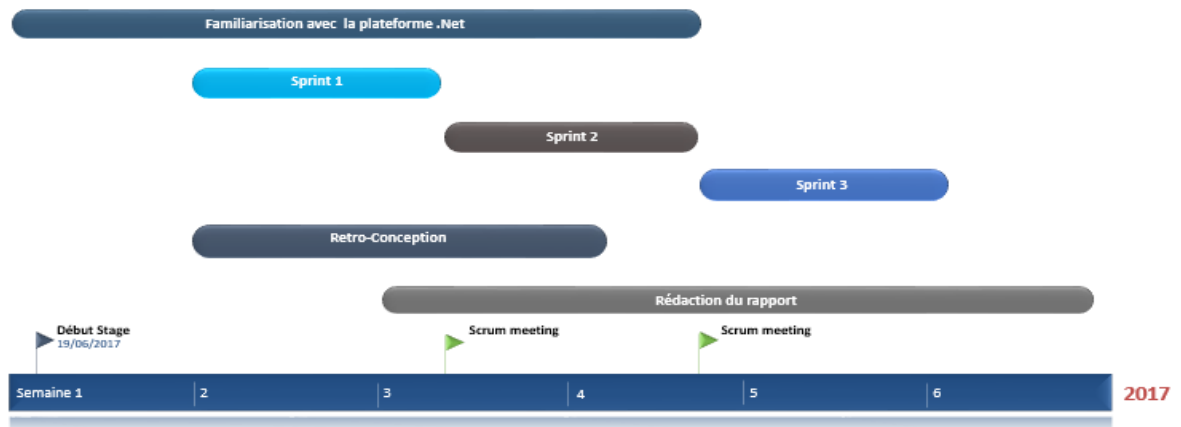


FIGURE 4.32 – Diagramme de gantt réel

Conclusion

Ce chapitre représente une récapitulation de tout le travail élaboré pendant le stage d'été ainsi qu'une présentation des résultats atteints. En effet, nous avons décrit les environnements matériels et logiciels sur lesquels nous avons travaillé. Nous avons ensuite passé à la présentation de quelques interfaces de notre application et enfin nous avons clôturé ce chapitre par la présentation du chronogramme des tâches.

Conclusion générale et perspectives

LE recrutement est une tâche stratégique délicate qui conditionne la performance des organisations. Bien mené, ses effets agissent directement sur la productivité, l'efficacité, l'image et le climat de travail. Donc l'automatisation de ce processus permet d'accroître la productivité administrative du service RH. Cependant cette productivité ne s'accompagne pas nécessairement par une réduction des effectifs mais, elle permet plutôt de libérer du temps pour des tâches autrefois négligées.

C'est dans ce cadre que s'inscrit notre stage d'été qui consiste à améliorer la plateforme intranet iRecruit de Linedata ayant pour objectif de faciliter le processus de recrutement. Cela permet notamment d'optimiser la gestion des talents avec une solution globale de gestion du recrutement, de la formation, des compétences, des entretiens et des carrières en entreprise. Une couverture fonctionnelle complète dans une solution flexible et intuitive.

Dans le présent rapport, nous avons détaillé les étapes par lesquelles nous sommes passés pour améliorer notre solution. Pour aboutir à ce résultat, nous avons tout d'abord commencé par présenter le cadre général de notre travail. Puis, nous avons présenté les différents besoins et les exigences relevées. Ensuite, nous avons abordé la phase de conception qui nous a expliqué l'architecture de l'application. Finalement, l'étape de réalisation, au cours de laquelle nous avons présenté iRecruit.

Grâce à notre expérience au sein de Linedata, nous avons appris, d'un point de vue personnel, à gérer notre projet de façon méthodique et organisée. Ce travail nous a été bénéfique dans la mesure où il nous a permis de mettre en pratique nos connaissances théoriques acquises tout au long de notre formation. Il nous a permis également

d'approfondir nos connaissances et d'apprécier l'importance d'une méthodologie de gestion de projet.

Durant ce projet, nous avons été confrontés à plusieurs problèmes et obstacles au niveau développement. En effet, nous n'avons pu se familiariser facilement avec le code de la plateforme déjà écrit par un autre stagiaire vue qu'il n'était pas bien commenté. Nous avons aussi eu des problèmes avec la base de données puisque les contraintes garantissant l'intégrité référentielle entre les tables ne sont pas bien établies. Nous n'avons pas eu assez de temps pour recréer une nouvelle base de données en respectant les contraintes d'intégrité car cela va falloir réécrire la majorité du code existant.

Comme perspectives de travaux futurs, nous proposons d'enrichir cette application en s'intéressant à certains points. Pour étendre notre solution, nous pouvons l'enrichir par d'autres fonctionnalités afin qu'elle soit utilisée non seulement par Line-data Tunisie mais aussi par les autres bureaux.

Netographie

- [N1] <http://fr.linedata.com>, dernière visite le 22/08/2017
- [N2] <http://www.agiliste.fr>, dernière visite le 22/08/2017
- [N3] <https://www.pentalog.fr/notre-demarche/methode-agile-scrum.htm>, dernière visite le 22/08/2017
- [N4] <http://cedric.babault.free.fr/rapport/node4.html>, dernière visite le 03/09/2017
- [N5] <http://www.memoireonline.com/mConceptionetrealisationdunrobotvirtuel.html>, dernière visite le 03/09/2017
- [N6] <https://openclassrooms.com/courses/apprendre-asp-net-mvc/le-pattern-mvc>, dernière visite le 03/09/2017
- [N7] <http://slideplayer.fr/slide/1290926>, dernière visite le 03/09/2017
- [N8] <http://nicolasesprit.developpez.com/tutoriels/dotnet/nouveautes-asp-net-mvc-3>, dernière visite le 19/09/2017
- [N9] <http://www.entityframeworktutorial.net/what-is-entityframework.aspx>, dernière visite le 19/09/2017
- [N10] <https://fr.wikibooks.org/Paradigmesdeprogrammation>, dernière visite le 19/09/2017
- [N11] <https://www.html5rocks.com/en/>, dernière visite le 29/09/2017
- [N12] <https://www.w3schools.com/css/css3intro.asp>; dernière visite le 29/09/2017
- [N13] <http://www.toutjavascript.com/main/index.php3>, dernière visite le 29/09/2017
- [N14] <https://www.w3schools.com/jquery/>, dernière visite le 29/09/2017
- [N15] <http://www.chiny.me/ajax-c-est-quoi-10-1.php>, dernière visite le 29/09/2017
- [N16] <https://docs.microsoft.com/visualstudio>, dernière visite le 29/09/2017
- [N17] <http://www.orsys.com/formations-sql-server.asp>, dernière visite le 29/09/2017
- [N18] <https://docs.microsoft.com/sql-server-management-studio-ssms>, dernière visite le 29/09/2017