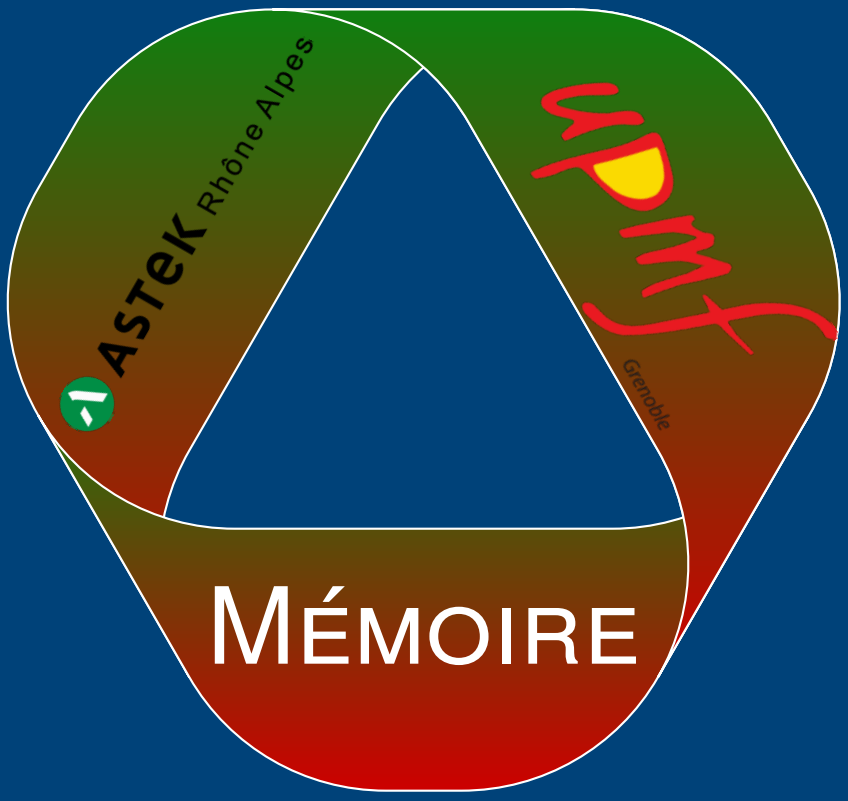
Master 2 Web Information et Connaissances

Université Pierre-Mendès-France

Création d'une application web de gestion de projets agiles

Mohamed Naji



Tuteur : Jérôme David

Encadrant : Kévin Delfour

Maître de stage : Xavier Spengler

Stage soutenu le 5 septembre 2013

Remerciements

Je tiens à remercier l'équipe AgileFactory et particulièrement Kévin Delfour qui a toujours été là pour répondre à mes nombreuses questions, qu'elles soient techniques ou méthodologiques. Je voudrais remercier également Oussama Abid qui a fait passer mon CV au sein d'Astek et mon maître de stage Xavier Spengler pour ses fortes connaissances agiles.

Table des matières

1 Introduction

En vue de la validation du master 2 IC2A (Ingénierie de la Cognition, de la Création et des Apprentissages) spécialité WIC (Web Informatique et Connaissances), j'ai effectué un stage de fin d'année en entreprise. J'ai intégré la société de service informatique Astek Rhône-Alpes, située à Échirolles, du 18 février au 30 août 2013 ; mon objectif était de gagner en compétences sur les technologies présentes dans le monde industriel et de bénéficier d'une première expérience conséquente dans un type de société où l'emploi reste dynamique malgré le contexte actuel, pour poursuivre ensuite en master recherche.

L'offre de stage consistait en la création d'une application web de gestion de projets agiles (dans les technologies Java/JEE, éventuellement GWT). Finalement les frameworks Grails et Bootstrap ont été retenus pour le développement. L'intérêt du stage reposait dans son aspect technique mais également méthodologique. Il a été l'occasion de découvrir, dans la pratique et la théorie, une méthode agile (Scrum notamment) en progression dans les projets en informatique et des outils relativement récents. Ces technologies sont des starters kits, c'est-à-dire qu'elles favorisent un développement rapide et une application robuste. Ce projet est issu du pôle open source de l'agence Astek de Grenoble. Il est effectué en tant que projet interne et non pour le compte d'un client. Ses ressources ont principalement étés deux stagiaires et trois développeurs en inter contrat sur des durées variables.

AgileFactory est le nom donné à l'application. Son objectif final est donc la création d'une application qui puisse permettre la gestion de projets mais aussi de formuler une réponse aux exigences de qualités de l'entreprise. Ces exigences sont les garanties pour les clients d'Astek d'avoir un interlocuteur conscient des questions entourant la qualité d'un produit numérique, questions d'autant plus importantes que les applications gagnent en complexité et en contraintes d'évolutivité. Les exigences de qualités d'Astek sont propres à l'entreprise et ne trouve pas de réponses dans les logiciels du marché, d'où la nécessité d'un produit sur mesure.

Ce rapport comprend une description de l'entreprise ainsi que du contexte du stage. Je poursuis ensuite avec une explication sur l'agilité en entreprise, essentielle à la compréhension de l'application AgileFactory, avant de m'attarder sur les questions techniques des outils utilisés. Enfin, j'aborde mes réalisations.

2 Contexte

2.1 Le secteur des sociétés de services en ingénierie

2.1.1 Description générale

Dans le secteur des activités informatiques, une SSII (nouvellement nommé ESN pour Entreprise du Service du Numérique) est une société de services spécialisée en génie informatique. Comme son nom l'indique elle se situe dans le secteur tertiaire et propose à ses clients divers services dans le but d'augmenter sa marge qui est calculée par la différence entre ses coûts fixes ajoutés des coûts salariales et le chiffre d'affaire de ces prestations.

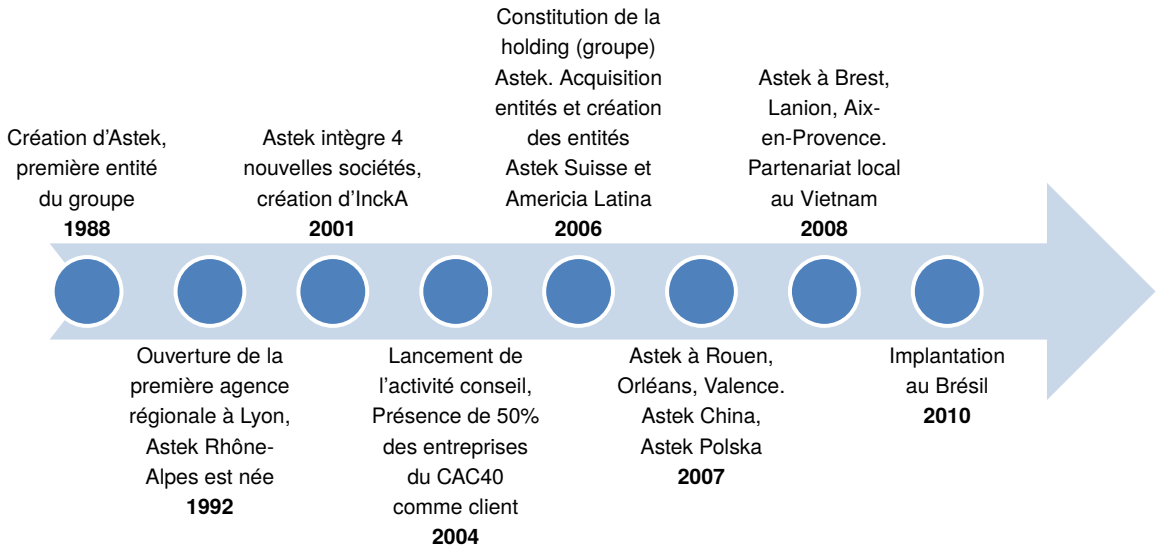
Les SSII ont pour objectif de répondre aux besoins des entreprises d'externaliser certains services. Ainsi, une entreprise fait appel à une SSII pour ses qualités de conseils, d'expertises et de réalisations de produits numériques. On peut généraliser les offres des SSII à deux grands types de contrats : contrats au forfait et assistance technique. Le plus souvent un contrat au forfait aboutit à la livraison d'un produit et une assistance technique au placement d'un employé dans l'entreprise cliente. Je reviendrai plus en détails sur ces points dans la section suivantes où je décris les prestations d'Astek.

2.1.2 Contexte de l'inter-contrat

Une SSII ne peut pas toujours placer ses employés sur un projet au forfait ou chez le client. L'employé peut à ce moment être intégré sur un projet interne qui lui est développé à l'agence. C'est l'occasion pour le développeur de monter en compétence sur des nouvelles technologies et de travailler sur des projets sans valeur ajoutée car ils ne rapportent pas d'argent à l'entreprise, mais qui peuvent lui être utile (par exemple un service de mails, de gestion des formations). Si l'entreprise joue la carte des formations, de veille technologique et de réflexions sur les besoins internes, les périodes d'inter-contrat peuvent être très bénéfiques et assurer une réactivité de l'expertise face au dynamisme des nouvelles technologies. Ce type de projets est aussi l'occasion d'intégrer de nouveaux stagiaires en les formant aux besoins présents et futures de l'expertise. Mon stage est le résultat de ce type de démarche et s'inscrit en plein dans l'actualité technologique et méthodologique de l'entreprise d'accueil. Technologique par l'utilisation de technologies web tel que les frameworks Grails et Bootstrap et méthodologique par la volonté de renforcer les méthodologies Agiles au sein de l'agence Astek de Grenoble.

2.2 Astek

2.2.1 Historique des dates importantes



2.2.2 Description d'Astek

Astek est une société de conseil, dingénierie en système dinformations de 3 200 collaborateurs. C'est une société indépendante créée en 1988. Elle regroupe 19 agences en France et de 15 implantations internationales à ce jour. Enfin, Astek propose des services variés : tierce maintenance applicative (TMA), conseil, forfaits, centre de services, gestion des infrastructures.

Forme juridique

Le groupe Astek est une Unité économique et sociale, c'est-à-dire un regroupement de plusieurs entreprises juridiquement distinctes ayant un CE commun. Les entités ainsi regroupées ont néanmoins une grande autonomie.

2.2.3 Astek Rhône-Alpes

Forme juridique et effectifs

Comme la grande majorité des filiales du groupe, Astek Rhône-Alpes est une SAS (société par actions simplifiées). Elle regroupe trois agences situées à Lyon, Grenoble et Valence. Elle emploie 590 consultants.

Logos

FIGURE 2.1. : Le logo du groupe Astek

FIGURE 2.2. : Le logo d'Astek Province

FIGURE 2.3. : Le logo d'Astek Rhône-Alpes

Chiffre d'affaire

Le chiffre d'affaire du groupe Astek s'élève à 205 millions d'euros en 2012. Cela place l'entité autour de la 20ème place des SSII en terme de chiffre d'affaire en France.

Chiffre d'affaire

Le chiffre d'affaire de d'Astek RA s'élève à 27 millions d'euros en 2012. Il a été en constance augmentation depuis 2008 où il s'élevait à 21 millions. Néanmoins, ces chiffres devraient être revus à la baisse pour l'année 2013.

Implantations

Astek Rhône-Alpes possède trois implantations : une à Grenoble, une à Valence et la dernière à Lyon.

Principaux clients

Les principaux clients de l'agence de Grenoble sont France Télécom, SFR, Total, le CHU de Grenoble, Ardis.

3 Projet AgileFactory

3.1 Introduction et contexte du projet

Le projet AgileFactory s'inscrit dans une dynamique de mise en place de la méthode Agile au sein de l'agence d'Astek Grenoble. A l'initiative de Kevin Delfour et Xavier Spengler, le projet visait à palier certains manques quant à la gestion des exigences de qualités entourant un projet Agile. En effet, si de nombreux outils de gestion existent sur le marché (nous avons par exemple utilisé tinyPM pour gérer le début des développements), ces outils ne gèrent pas toutes les contraintes que la charte de qualité que l'entreprise impose aux projets qui y sont développés ou alors d'une manière pas assez complète. Je décris certaines de ces exigences dans la suite du chapitre.

Il y a un réel intérêt pour un étudiant comme moi d'avoir à développer ce type d'application car elle permet d'approfondir dans la théorie les principes de l'agilité qui aujourd'hui opèrent une vraie poussée dans les entreprises. Dans un projet classique comme ceux dits au forfait, les cahiers des charges sont définis parfois pour plusieurs mois ; le problème est que les demandes du client peuvent évoluer et souvent le produit à la livraison ne correspond pas aux attentes. L'agilité apporte de la flexibilité dans la création du produit et renforce les interactions avec le client tout au long du développement. Ce stage m'a donc permis de gagner en compétence pour la gestion de projet agile.

Mon arrivée au sein d'Astek s'est effectuée le 18 février. Hao, le collègue stagiaire, travaillait à mi-temps sur l'application depuis janvier et à plein temps à partir de début mai. Après une rapide présentation à l'ensemble des employés de la boite, on m'a attribué un poste de travail. J'ai ensuite pris connaissance des spécifications de l'application et commencé l'installation des outils nécessaires :

TABLE 3.1. : Noms et versions des technologies utilisées

	Outils	Versions
IDE	Eclipse	JUNO
JAVA	JDK	7U11
BDD	PostgreSQL	9.2.2
FRAMEWORK	Grails	2.2.1
FRAMEWORK	Groovy compiler	2.0
VERSIONNING	Git	1.8.0

Une fois les installations effectuées, j'ai pris en main l'environnement de Grails et entamé la programmation d'un CRUD [1]. J'ai ensuite récupéré le projet sur le dépôt distant avec GIT

1. Create, Read, Update, Delete. C'est une application avec ces quatre méthodes et une interface utilisateur

pour prendre connaissance de l'avancement du projet. J'ai commencé le développement par la création d'une fenêtre modale pour la présentation d'un formulaire à l'utilisateur. Sous la tutelle de Kévin Delfour, j'ai progressé ainsi et pris des tâches de plus en plus complexes.

Je vais dans la suite du chapitre décrire en détails les besoins qui ont conduits à la volonté de produire AgileFactory avant de présenter le produit. J'y ajouterai la manière dont on a travaillé en équipe. Mais il est d'abord nécessaire de décrire ce qu'est l'agilité en entreprise et de présenter les termes agiles.

3.2 Méthode Agile

L'agilité opère depuis quelques années une véritable percée dans le monde des sociétés de services informatique car elle permet la flexibilité en impliquant le client tout au long de la création d'un projet. Les méthodes classiques font intervenir principalement le client au commencement et à la livraison du produit. Cela crée un grand espace de temps où il n'y a pas d'interaction avec le client ce qui pose un problème car devant la montée de la complexité entourant un projet informatique, il y a des facteurs non prévus dans les spécifications des besoins qui impactent la réalisation. Ce qui provoque un réel décalage entre les attentes du client et la livraison. L'agilité est un ensemble de méthodes avec un fondement itératif dans la mesure où un retour sur les besoins est effectué tout au long de la réalisation d'un produit. On y ajoute éventuellement le principe d'incrémentation, qui implique un découpage du projet en plusieurs fonctionnalités livrables.

Une définition commune des projets agiles est donnée dans "Le manifeste agile" [2]. L'un des obstacles à l'entrée de l'agilité dans les sociétés de services est justement le point fort de ces méthodes : l'itération. En effet les clients préfèrent définir tous les besoins dans un cahier des charges afin de gérer le temps et les coût du projet. Le problème est justement qu'il est très difficile de tout prévoir et la masse des documents produits se transforme souvent comme un boulet au pied alors que le temps passé à les produire n'est pas négligeable. J'ai pu observer un réel débat autour de ces questions durant le stage. Des données statistiques quant à la productivité des différentes méthodes de gestion commencent tout juste à apparaître et il semble bien que la réussite d'un projet est plus grande par une gestion agile.

Je vais dans la suite du rapport présenter en détails ce qu'est une méthode agile et plus particulièrement la méthode scrum qui se distingue des autres en ce qu'elle ne permet pas une planification temporelle des tâches à produire. Je vais également introduire le vocabulaire agile/scrum afin de faciliter la compréhension des éléments de l'application.

2. Document produit à la suite d'une réunion pour la factorisation des principes de la conduite de projet. Le document a du poids puisque la réunion est composée de grands noms du développement logiciel

3.2.1 Les principes de scrum

Un développement itératif et incrémental : En scrum, on définit des périodes (ce sont des sprint ou itérations) à la fin desquelles une livraison est effectuée avec une démonstration au client. Durant cette période, qui peut varier d'une à six semaines, aucune modification du client ou autre n'est prise en compte. C'est donc un développement itératif. Après la démonstration, une discussion avec le client et l'équipe est ouverte pour faire le point entre les attentes, les résultats et le contenu de la prochaine livraison. Avec cette manière de faire, on installe un développement incrémental. Dans la méthode scrum on ne définit pas le sprint suivant avant la livraison du sprint en cours.

La participation du client : Le client intègre l'équipe scrum dans le sens où il est présent tout au long du cycle de vie du projet. Les interactions entre lui et l'équipe sont régulières ce qui garantit de ne pas avoir de grands décalages entre ses attentes et les réalisations.

L'auto-organisation : En opposition avec un système hiérarchique (directeur de projet, chef de projet, expert, développeurs) scrum se base sur le principe d'auto-organisation. L'équipe prend les décisions ensemble, progresse ensemble. Par exemple un membre n'est pas assigné à un ensemble de tâches parce que c'est son domaine d'expertise, il choisit lui-même ses tâches (cela peut être pour progresser sur un point si la prise de la tâche n'entraîne pas de retard). Cette façon de faire peut créer un vrai dynamisme au sein de l'équipe mais à condition d'en accepter contraintes. Les limites de cette conception sont inhérentes aux individus : si un seul membre ne joue pas le jeu, c'est toute la productivité de l'équipe qui est remise en cause.

Les réunions quotidiennes : Tout les matins l'équipe se réunit, en générale quinze minutes, et chacun prend la paroles à tour de rôle pour répondre à trois questions :

1. Qu'est-ce que j'ai fait hier ?
2. Qu'est-ce que je vais faire aujourd'hui ?
3. Est-ce qu'il y a des points bloquants ?

Ces réunions quotidiennes permettent à l'ensemble de l'équipe de savoir ce que font les autres et où en est l'application. Elle se déroule debout pour éviter qu'elle dure trop longtemps.

Les réunions de fin de sprint : A la livraison d'une fonctionnalité, l'équipe et le scrum master se réunit pour faire le point. Les questions soulevées portent ce qui s'est bien passé et mal passé au cours du sprint afin de trouver des solutions pour gagner en productivité.

Complexité : Une story dans un projet scrum ne possède pas une planification temporelle mais on on estime une certaine complexité qui correspond à la difficulté de l'ensemble des tâches composants la story. Plus une story est complexe, plus sa complexité évaluée est haute. Les estimations sont effectuées en début de sprint pendant ce qu'on appelle un planning poker. La somme des complexité des story doivent être égales aux unités de complexité que l'équipe peut produire en un sprint.

3.2.2 Le vocabulaire scrum

Il existe un vocabulaire scrum qu'il faut connaître pour comprendre AgileFactory.

ScrumMaster : C'est celui qui fait en sorte que l'équipe puisse travailler dans de bonnes conditions. Il est le garant de la bonne pratique des méthodes scrum. Ce n'est pas un chef de projet, il n'a pas les mêmes responsabilités. Le voir comme un coach agile.

Product Owner : C'est celui qui représente le ou les clients. Il peut être le client. Son rôle est de fixer les exigences et d'accompagner le développement.

Team Member : C'est l'équipe développement. Elle est de préférence pluridisciplinaire mais où chacun touche à tout.

Daily standup : Réunion quotidienne du matin.

Itéarion/Sprint : C'est la période d'une à six semaines (souvent deux semaines) à la fin de laquelle l'équipe effectue une livraison avec démonstration au product owner.

Exigence/Requirement : ce sont les attentes hiérarchisées du clients présentées sous la forme : "en tant que...je veux...afin de..."

Scénario : Découpage des exigences en unités qui peuvent ensuite être traduite en tâches.

Release : Série de sprints successifs qui aboutissent à la livraison d'une version de l'application.

Tâche/Task : Unité de travail correspondant à une implémentation de code dans l'application.

Story/UserStory : Ensemble de tâches regroupées par thème qui peut correspondre à une exigence.

Rétrospective : réunion de fin de sprint.

3.3 Les besoins

3.3.1 Objectif de l'application

L'objectif d'AgileFactory est la création d'un logiciel de gestion de projet Agile. Avec la progression de l'agilité dans les entreprises, des outils de gestion visent à accompagner les acteurs d'un projet tout au long de sa réalisation. Les logiciels existants proposent déjà

de bonnes solutions pour la partie développement, mais ne prennent pas en charge l'industrialisation d'un projet ou des méthodes transverse au développement. AgileFactory fournira, en plus des fonctionnalités existantes des autres applications du marché, des possibilités de suivis des qualités, de gestions des exigences et des contraintes de traçabilité. AgileFactory s'accompagne également d'un cycle agile modifié, qui définit une chaîne, courte, visible à tous les étages par l'équipe de développement et du client. Le processus, type TDD [3], sont pilotés par les scénarios, fonctionnels, puis techniques, puis par les tests unitaires.

L'application est présentée en tableau de bord pour faciliter la navigation entre les différents modules. On y retrouve le Sandbox, le Backlog et le Taskboard. On y retrouvera également une section de référencement de tous les services annexes au projet, une historisation intelligente et complète, l'intégration de logiciels de gestion de versions, ou encore l'ajout de commentaires sur les itérations, stories ou tâches. L'application permettra également la visualisation graphique des informations pertinentes. L'objectif est donc de répondre à tous les besoins de gestions d'un projet agile, d'en faciliter la mise en oeuvre et d'en assurer la bonne pratique.

3.3.2 Pourquoi produire AgileFactory

Les autres applications du marché

D'autres logiciels de gestion propose des outils performant présentent des limites pour la gestion des projet d'Astek Rhône-Alpes :

1. La gestion des exigences dans les autres logiciels ne couvrent pas les besoins qualité d'Astek.
2. La limitation du nombre d'utilisateur et le prix de la licence des autres logiciels est une autre contrainte.
3. Voulant en faire trop, les applications sont parfois surchargées et les interfaces trop complexes
4. Les autres logiciels ne sont pas portables sur d'autres plate-formes (tablettes par exemple)

Les exigences de qualités

Les exigences entourant un produit Astek imposent certaines contraintes de traçabilité dans la gestion d'un projet. Ces exigences doivent maintenir une garantie de qualité dans le produit finale. Ces exigences sont par exemple la possibilité d'avoir un historique complet décrivant toute activité ou modification opérée dans le logiciel de gestion. TinnyPM, qui est un logiciel de gestion répandu, propose un historique mais AgilleFactory tente d'aller plus loin en y associant une recherche par mot clé. Une autres exigences de qualité

3. Test Driving Developement est une façon de programmer qui implique d'écrire les tests en premier lieu, puis d'écrire le code pour valider les tests

est l'association entre la demande d'un client (sous le mot clé "requirement") et le code répondant à cette demande. AgileFactory propose ainsi une matrice bidirectionnelle associant un commit à la tâche ou à la story. Un autre point important est la centralisation de tous les outils et service nécessaire à un projet. On retrouve ainsi dans AgileFactory une partie appelée dashboard dans laquelle l'équipe d'un projet peut entrer des liens internet ou des fichiers comme par exemple un document entourant la création du produit. Ces points seront développés dans le chapitre réservé à la description de l'application. Voici les trois contraintes principales auxquelles l'application doit répondre :

1. Le suivi de la demande d'un client jusqu'au code répondant à cette demande. Un client souhaite ajouter une fonctionnalité ; il propose ainsi une exigence (ou requirement) à l'équipe de développement. Cette exigence est ainsi découpée en tâches. Il s'agit donc de faire le lien entre un commit (ajout de code) et la demande du client.
2. Un historique répertoriant toutes les modifications opérées dans le logiciel de gestion. Ceci rentre également dans le cadre de lexigence de traçabilité. En effet l'on doit pouvoir retrouver toutes les modifications du client ou de l'équipe de développement car celles-ci impactent directement le projet.
3. La centralisation des services de l'équipe de développement. Les membres de l'équipe peuvent retrouver ainsi rapidement les documents externes quant aux outils de travail et par exemple éviter des conflits de versions.

La veille technologique

On a développé AgileFactory avec deux technologies encore jamais utilisées dans l'agence Astek de Grenoble : Grails et Bootstrap. Le but de ce choix est double : tester la pertinence de ces outils et faire monter en compétence les développeurs.

L'intégration de nouveaux stagiaires

AgileFactory a été également l'occasion d'intégrer deux stagiaires avec la perspective de les embaucher après le stage. C'est un processus répandu au sein des SSII. C'est également l'occasion pour l'entreprise daccueil de produire une application à moindre coût.

3.4 Description de l'application

AgileFactory est découpé en sept modules principaux. Certains de ces modules sont communs aux logiciels de gestion existants et d'autres visent à répondre aux différentes contraintes de traçabilité. Ces modules vont être décrits ici pour être ensuite résumé dans leurs fonctionnements sous la forme d'un schéma synthétique. L'interface utilisateur se découpe ainsi :

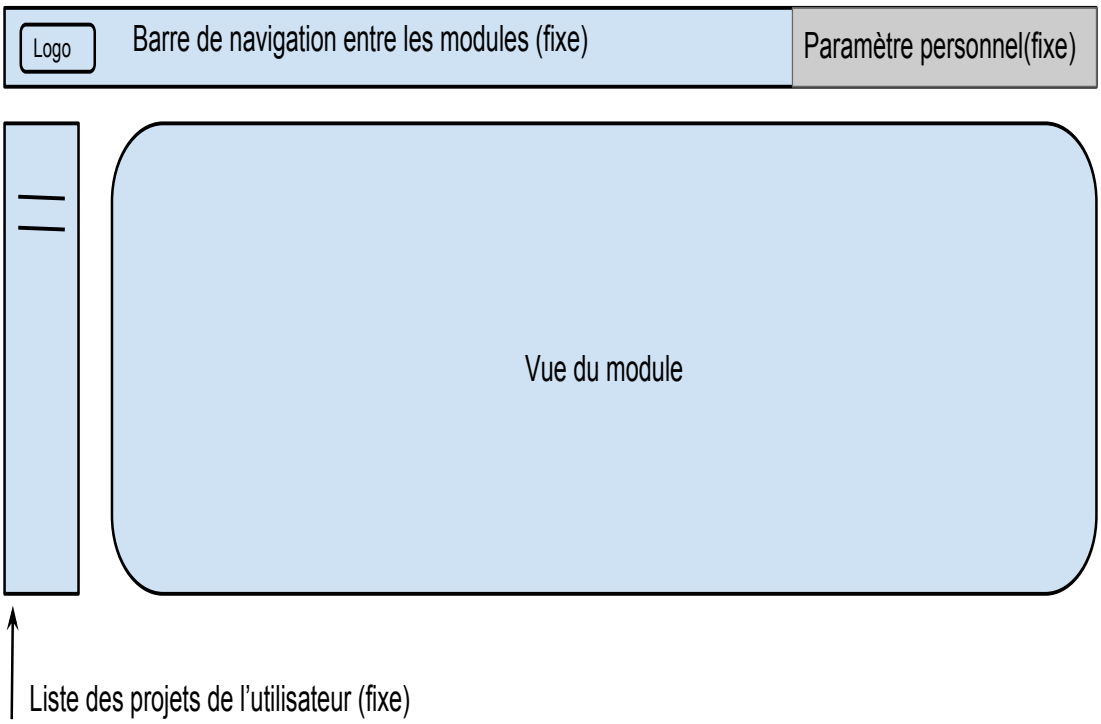


Figure 3.1. : Vue schématique de l'interface utilisateur AF

Les parties notées fixes ne changent pas d'une vue à une autre et d'un utilisateur à un autre. La barre de navigation permet de passer d'un module à l'autre et la liste d'un projet permet de changer le projet courant. En haut à droite l'on retrouve les paramètre utilisateur (déconnexion et paramètre personnel). La navigation entre les modules va impacter la partie centrale.

La compréhension des différents modules impose de connaître le vocabulaire agile. Se référer à la section précédente pour les explications. Il y a donc sept modules principaux plus celui du paramétrage personnel et général de l'application.

3.4.1 Les principaux modules

Dashboard : Les services disponibles pour chaque projet et les liens utiles sont visibles depuis le dashboard. Ces modules visent à centraliser tout les éléments extérieurs nécessaires à un projet.

Sandbox : On pose les demandes d'évolutions/idées ici, ainsi que la gestion des bugs. Sous forme d'exigences auxquelles sont associées des scénarios. Après validation, les exigences peuvent être directement converties vers le backlog en stories et les scénarios en tasks. Cette partie est accessible au product owner qui peut lui-même créer de nouvelles exigences. A la création, elles possèdent un état "NEW". Elles peuvent ensuite soit être passées en "VALIDATE" ou "REJECTED" suivant qu'elles sont acceptées ou rejetées. La

suppression d'une exigence n'est pas effective en base pour une question de traçabilité. Ainsi l'on peut toujours retrouver les éléments supprimés.

Backlog : La gestion des itérations, des releases et des paper stories s'effectue ici. Le drag&drop a été privilégié pour le déplacement de stories d'une itération à une autre. Il est possible de voir toutes les informations des composants agiles. Un sous module permet le traitement des stories et de ses tasks. Pour une story, on peut estimer sa complexité avec l'échelle classique, la splitter (en faire deux stories lorsqu'un nouveau découpage est souhaité), mettre à jour son état d'avancement. Les informations tirées des logiciels de gestion de versions sont visibles depuis les pages consacrées aux détails des stories.

Taskbord : Le suivi des tâches intègre la conversion des tâches en stories, la possibilité de déplacer des tâches entre différents stories ou entre différents états ("TODO", "CURRENT", "DONE"), le transfert d'une itération à une autre. La vue du module est découpée verticalement pour les itérations et horizontalement pour les états des tasks. Différents niveaux de détails sont possibles.

Historique : Une trace complète du projet depuis sa création peut être retrouvée ici. Les événements sont bien détaillés (qu'est-ce qui est changé ? Qu'est-ce qui est créé...). Le filtrage est également mis en place pour faciliter la recherche dans l'historique.

Quality : Ce module permet la visualisation sous forme d'un arbre de tous les composants agiles du projet. La racine est constituée par le nom du projet et les noeuds peuvent être des itérations, des releases, des exigences, des stories et des tasks suivant la hiérarchie agile. Les noeuds enfants des tasks sont les commits associés. Par clic, on retrouve le code d'avant et après commit.

Espace admin : Espace réservé à l'administrateur pour gérer les utilisateurs, les rôles (ce qu'il est possible ou non de faire pour un utilisateur), les projets, les services. Le scrum master se verra le plus souvent attaché à la partie admin.

Le fonctionnement des modules et leurs liens sont résumés par le schéma qui suit. Les modules sont regroupés par type et les différentes interactions avec les composants y sont résumées.

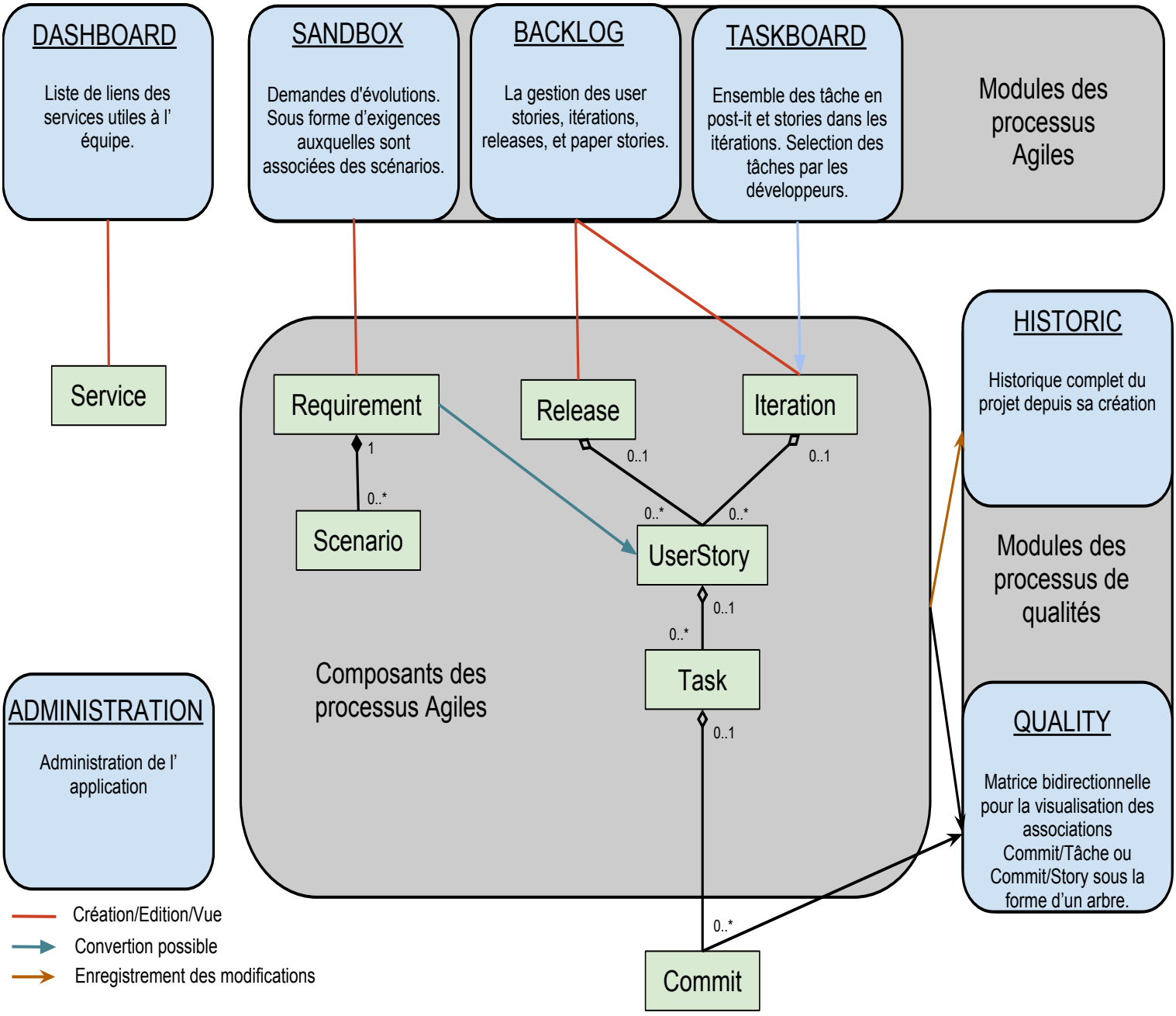


Figure 3.2. : Schéma d'ensemble de l'application AgileFactory

3.4.2 Points d'ergonomie

Les formulaires Tous les formulaires de saisies d'informations sont intégrés dans des fenêtres modales chargées en ajax. Ceci permet une homogénéité dans les interfaces et de ne pas surcharger les pages.

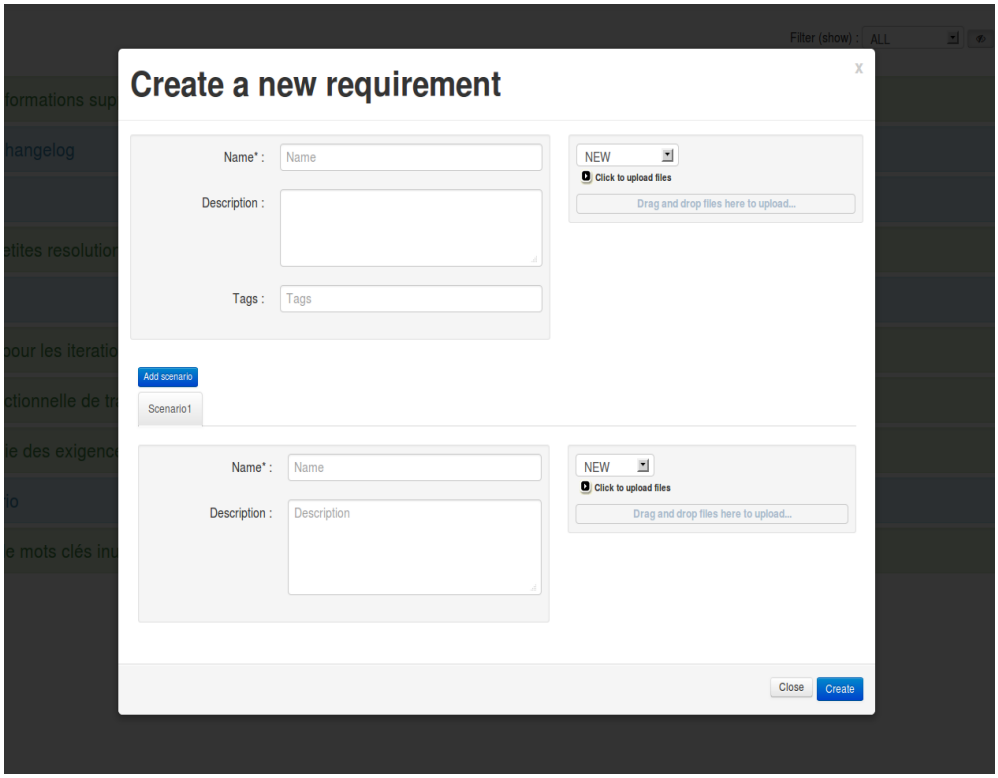


FIGURE 3.3. : Capture d'écran du formulaire de création de requirements

Le Drag&drop Le Drag&drop a été privilégié dans le baclog et le taskboard. Avant l'utilisation des logiciels de gestion de projets, les tâches sont représentées par des postits collés sur un tableau et les développeurs peuvent les déplacer pour les changer d'état dans différentes parties du tableau. C'est pour garder l'aspect de prendre une tâche et de la poser que le Drag&drop a été choisi.

Le responsive design Avoir une application multi plate-forme était un des besoins du projet. A l'aide de Bootstrap par l'utilisation des composants "span", un souci particulier a été apporté à cette contrainte. Néanmoins, certains points restent à revoir, notamment pour le dépassement de texte pour les plus petits supports. Les menus sont néanmoins correctement affiché suivant le support. Par exemple, la barre de navigation apparaît avec une liste déroulante pour un petit afficheur.

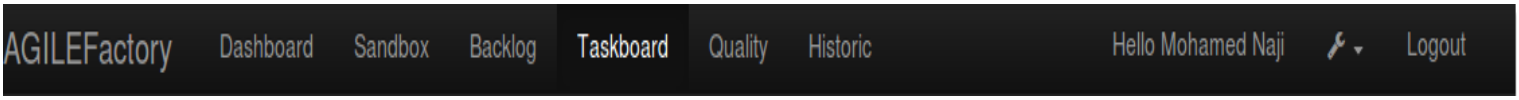


FIGURE 3.4. : Capture d'écran de la barre de navigation dans un grand écran

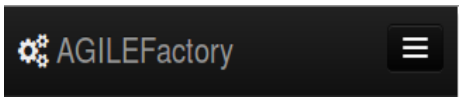


Figure 3.5. : Capture d'écran de la barre de navigation dans un petit écran

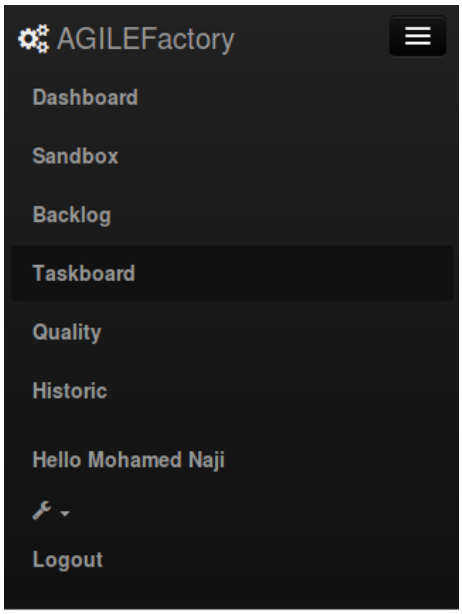


Figure 3.6. : Capture d'écran de la barre de navigation dans un petit écran après clic

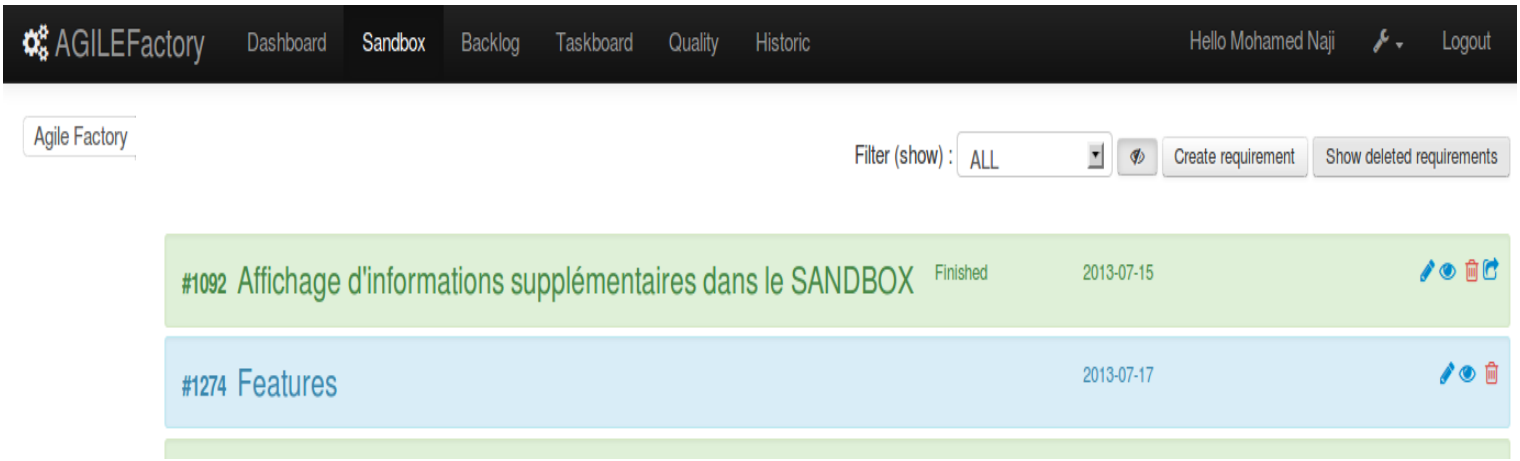


Figure 3.7. : Capture d'écran de l'application sur un support PC

grailsSchema.png

FIGURE 3.8. : Capture d'écran de l'application dans un support mobile (avec l'outil Web développeur de chrome)

On voit ici le texte coupé et certains boutons disparaître. Nous n'avons pas au cours du développement accordé d'importance à ce problème. C'est une évolution future dans le cas où une utilisation tablettes/mobiles s'effectue. Certaines configurations du framework bootstrap permettent de régler ce problème avec le principe du grid system (on choisit l'information affichée suivant le type de support)

Minimum de clic Les parties fixes de l'interface permettent d'atteindre n'importe quel module ou projet en un clic (sauf pour le sous module user story).

Icônes en police La plus part des boutons permettant à l'utilisateur d'intervenir avec l'application sont représentés avec des icônes. Les icônes utilisées sont vectorielles et donc adaptables à la résolution de l'utilisateur sans pertes. Un souci a été apporté à pertinence du choix des icônes mais un test utilisateur sera nécessaire.

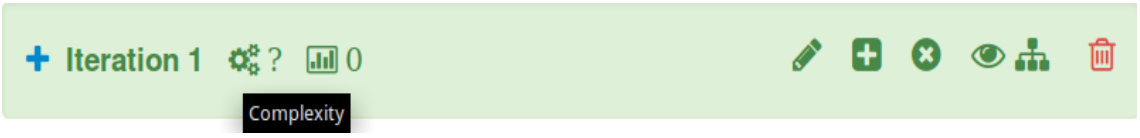


FIGURE 3.9. : Capture d'écran exemples d'icônes

3.5 Projet méthode Agile

AgileFactory a été développé avec une méthode Agile adaptée à nos contraintes. Un sprint a été mis en place avant mon arrivée qui a duré environ deux mois. Comme nous étions que deux développeurs à temps complet sur l'application, nous avons mis en place la méthode

dite Kamban qui est une méthode Agile. Elle ressemble à Scrum à ceci près que les sprint n'ont pas de durées fixes et peuvent être alimentés en cours de sprint. Ainsi les tâches étaient ajoutées au fur et à mesure. En avril nous avons été rejoint par un développeur en inter-contrat. Nous avons continué de travailler en Kamban jusqu'à l'arrivée de deux autres inter-contrats ce qui nous a permis de mettre en place Scrum. AgileFactory à ce moment-là, avait une version stable que nous avons utilisé pour la gestion de projet. Nous sommes entrés à ce moment véritablement dans l'agilité Scrum en mettant en place les standups et les itérations de deux semaines.

4 Techniques et méthodologies

4.1 Technologies

4.1.1 Grails/Groovy

Grails est framework open source développé par la société G2ONE compatible avec l'environnement JEE dont la maintenance est assurée par SpringSource depuis son rachat fin 2008. Le framework en est aujourd'hui à sa version 2.3.0.L'objectif de Grails est de fournir aux développeurs un stater kit, c'est-à-dire un outil permettant de produire des applications en un minimum de temps. Pour cela Grails à misé sur la philosophie "convention over configuration". Cela signifie que par des conventions définies (par exemple une certaines façon de nommer les contrôleurs et les vues) on évite des points de configuration (par exemple l'urlmapping). L'objectif dans l'utilisation de Grails pour cette application était de tester ses capacités de stater kit. Et en effet, Grails est dans ce sens un outil puissant.

Grails fournit une architecture de type MVC à laquelle est intégré un package services. L'architecture de notre projet est donc simple : il y a un ensemble de classes groovy formant le modèle. Les vue sont associées aux controlleurs (pour une vue il y a un controller). Ces derniers relèguent le traitement des données aux services qui gèrent également la persistance (pour un controller il y un service). Nous n'avons pas approfondi l'architecture de la base de données car celle-ci est construite par le Grails Object Relation Mapping.

Architecture de Grails

Les technologies qui gravitent autours de Grails sont connues de l'environnement JEE. On y retrouve par exemple Hibernate, Spring ou SiteMesh. Grails est basé sur le langage dynamique Groovy qui qui vient se placer au dessus du langage Java. Son principal intérêt est d'offrir une syntaxe simplifiée qui intègre également la syntaxe Java.

Ce que l'on retrouve dans Grails

SiteMesh : C'est framework Java/JEE qui facilite l'utilisation des layouts d'une application web. Il permet l'inclusion de templates dans les GSP (Groovy Server Page), d'un header ou autre composant réutilisable au travers de l'application. Par exemple, nous avons beaucoup utilisé dans AgileFactory le principe des templates avec une GSP principale et dans laquelle sont incluses d'autres GSP, ou encore pour avoir la même barre de navigation au sein de tous les modules.

Hibernate : Grails utilise le Object Relation Mapping de Hibernate mais en implémentant ses propres configurations pour donner une surcouche nommée GORM qui va permettre la persistance des objet du modèles de données. GORM permet entre autre d'intégrer les points forts de Grovvy comme le typage dynamique dans l'utilisation de Hibernate. Le mapping en base de données est grandement facilité par GORM qui inclut des méthodes déjà implémentées comme save() ou delete() et permet un accès rapide aux données avec la méthode withCriteria().

Spring : C'est un framework qui se charge de la création des objets et leurs mise en relation par un fichier de configuration.

C'est différents outils forment larchitecture de Grails. On pourrait ici citer également la Gant qui est une surcouche de Ant qui permet lautomatisation de certaines tâches récurrentes dans le développement d'un logiciel. Voici un schéma résumant l'architecture du framework. On y voit les différentes surcouches des grands logiciel opensource utilisés dans le monde industriel et leur intégration dans le modèle MVC de Grails.

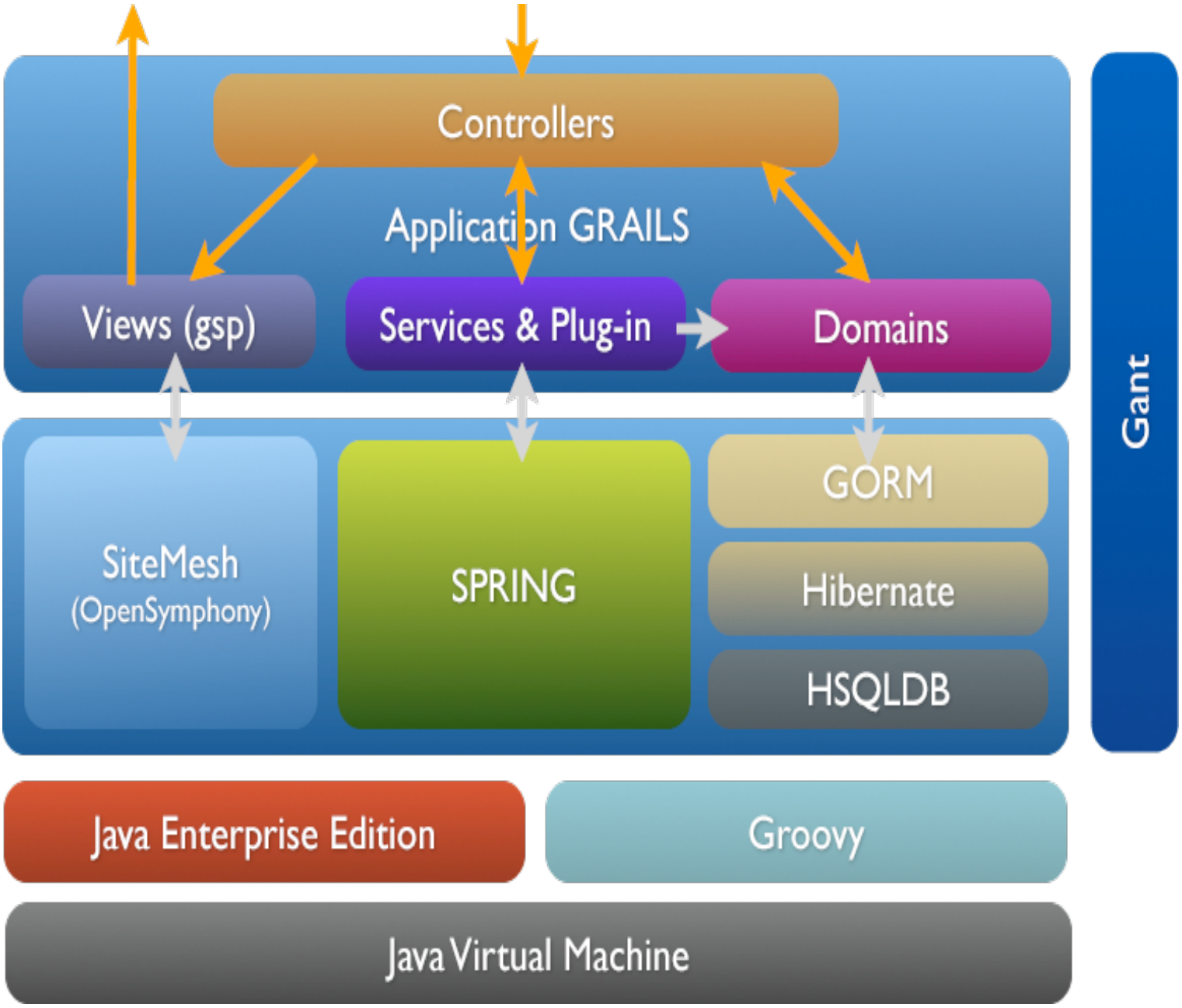


Figure 4.1. : Architecture de Grails

4.1.2 Bootstrap

Twiter Bootstrap est donc un framework open source facilitant la création d'interface utilisateur car il intègre des feuilles de styles CSS prédéfinies et des composants HTML/JS. Ainsi, il peut aider à la création d'une interface homogène, au design soigné. Outre ses caractéristiques il permet le responsive design et donc un développement multiplate-forme.

L'utilisation principale de Bootstrap consiste à donner à des balise HTML des noms de classes pour y effectuer un traitement CSS. Le code "spanX" par exemple(X étant un chiffre de 1 à 9) va donner à la balise des dimensions proportionnelles à la valeur du chiffre X, en tenant compte des éléments voisins. Le code "navs" lui va intégrer un menu par onglet dans la div.

4.2 L'intégration continue

Nous avons eu recours pour l'intégration aux tests unitaires et d'intégrations. Grails facilite l'intégration des tests. Par exemple il peut automatiser la création des tests lors de la création d'un controller ou d'un service. Grails intègre JUnit ce qui permet l'utilisation des méthodes assertFalse, assertEquals, etc.

4.2.1 Tests unitaires

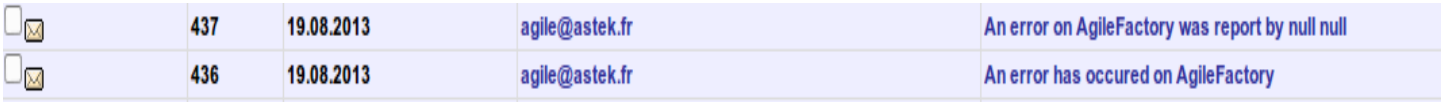
Les tests unitaires dans Grails permettent de cibler des éléments de code (le plus souvent des méthodes) afin de tester si le traitement des méthodes correspond bien aux attentes. Les test unitaires, contrairement aux tests d'intégration, ne nécessitent pas le recours à la base de données. La méthode MockDomain() permet de simuler la persistance des objets et les différentes méthodes associées (save(), validate()).

4.2.2 Tests d'intégrations

Contrairement aux test unitaires les test d'intégration ont accès à la base de données et nécessite un build du projet. Outre cette différence le principe est le même, testé des méthodes dans le but de vérifier si elles effectuent correctement le traitement spécifié.

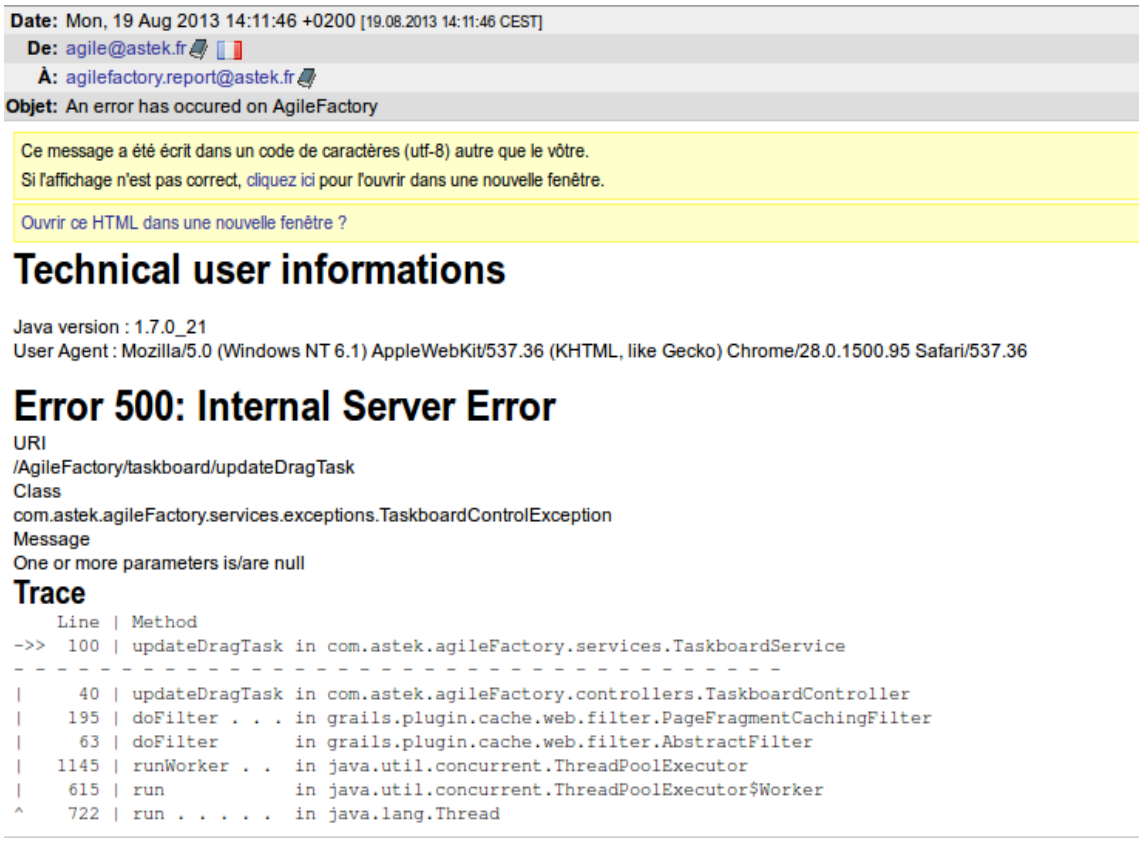
4.2.3 Hudson

Le serveur d'intégration Hudson a été mis en place pour effectuer une compilation et lancer les test durant la nuit. Cet outil permet d'assurer une robustesse tout le long du développement. En cas d'impossibilité de compilation ou d'échec des tests, un mail est envoyé à l'équipe pour l'informer de l'erreur. La pratique Agile veut que c'est toute l'équipe qui est responsable de l'intégration continue. Un test qui échoue par exemple n'est pas dévolu à celui qui a écrit le code et la maintenance peut être effectuée par un autre membre de l'équipe.



	437	19.08.2013	agile@astek.fr	An error on AgileFactory was report by null null
	436	19.08.2013	agile@astek.fr	An error has occured on AgileFactory

FIGURE 4.2. : Exemple de mail envoyé par Hudson



Date: Mon, 19 Aug 2013 14:11:46 +0200 [19.08.2013 14:11:46 CEST]
De: agile@astek.fr
À: agilefactory.report@astek.fr
Objet: An error has occured on AgileFactory

Ce message a été écrit dans un code de caractères (utf-8) autre que le vôtre.
Si l'affichage n'est pas correct, cliquez ici pour l'ouvrir dans une nouvelle fenêtre.

Ouvrir ce HTML dans une nouvelle fenêtre ?

Technical user informations

Java version : 1.7.0_21
User Agent : Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.95 Safari/537.36

Error 500: Internal Server Error

URI
/AgileFactory/taskboard/updateDragTask
Class
com.astek.agileFactory.services.exceptions.TaskboardControlException
Message
One or more parameters is/are null

Trace

```
    Line | Method
->>  100 | updateDragTask in com.astek.agileFactory.services.TaskboardService
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|     40 | updateDragTask in com.astek.agileFactory.controllers.TaskboardController
|    195 | doFilter . . . in grails.plugin.cache.web.filter.PageFragmentCachingFilter
|     63 | doFilter       in grails.plugin.cache.web.filter.AbstractFilter
|   1145 | runWorker . .  in java.util.concurrent.ThreadPoolExecutor
|    615 | run            in java.util.concurrent.ThreadPoolExecutor$Worker
^    722 | run . . . . .  in java.lang.Thread
```

FIGURE 4.3. : Détail d'un mail de Hudson

Le matin à la réception des mails l'équipe effectue un retour sur ces erreurs et les corrige. Une démonstration de l'application a été effectué à l'agence Astek de Lyon. Nous avons pris pour cela la version appelée AgileFactoryNightly qui correspond à la version du projet que Hudson met en place chaque nuit. Un retour a été fait par l'équipe de Lyon sur la qualité d'une telle démarche (robustesse en cours de développement de l'application)

5 Mes réalisations

5.1 Le sandbox

La sandbox est le module permettant de gérer les requirements. La vue principale est une liste de tous les requirements associés au projet courant, avec un code couleur informant de leur état de validation. La liste est paginée à l'aide du composant Bootstrap pagination et on ne récupère en base que les items affichés sur la page. Le module permet, via des fenêtres modales, la création et l'édition de requirements, avec les scénarios associés.

5.1.1 Difficultés

La difficulté consistait ici à l'utilisation de formulaires imbriqués pour la création des requirements et des scénarios en respectant la spécification graphique : un seul formulaire dans une fenêtre modale, liste des scénarios dans une barre d'onglets, appels ajax, possibilité d'uploader des fichiers pour chacun des items (requirement et scénario).

5.1.2 Solutions

La solution a consisté en l'adaptation du composant "navs" de Bootstrap qui permet l'affichage du barre d'onglet renvoyant à différentes balises de la page. Pour l'ajout d'un scénario, on effectue une requête ajax pour recevoir un template correspondant à un nouveau formulaire pour les scénarios qui va être intégré au formulaire du requirement.

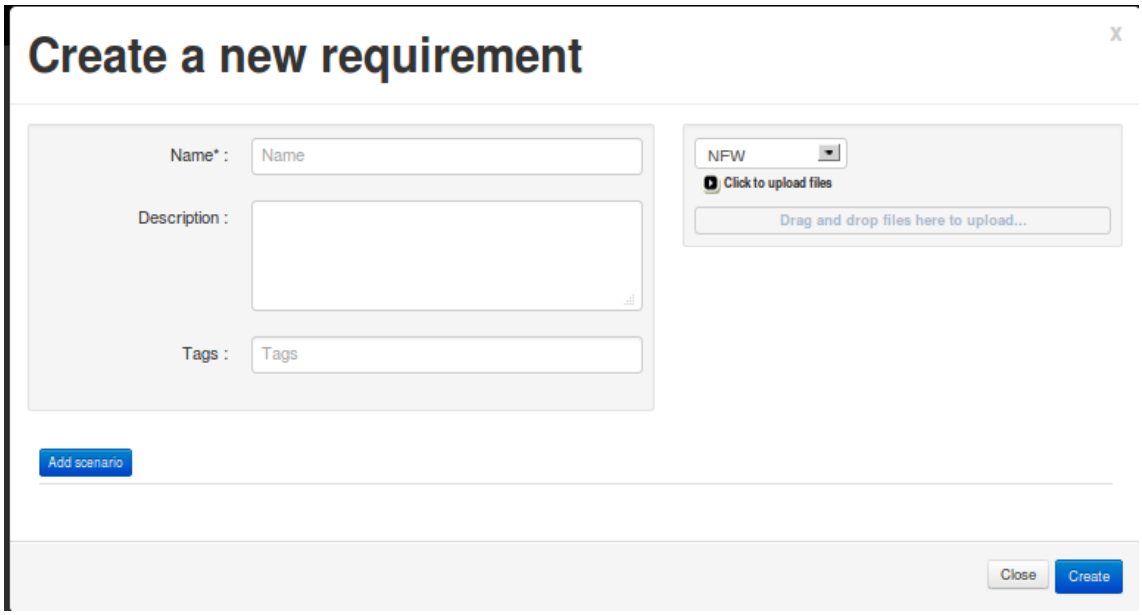


Figure 5.1. : Formulaire de création de requirements sans scénario

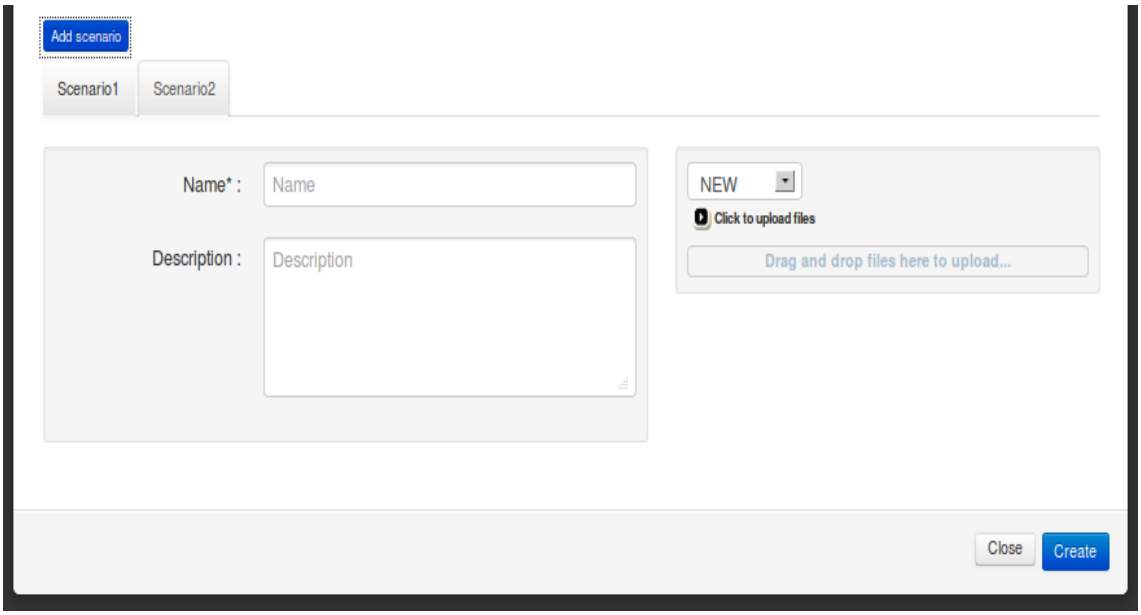


FIGURE 5.2. : Ajout de scénario dynamique après 2 clic sur "add scenario"

Pour un requirement et pour chaque scénario associé, il y a la possibilité d'uploader des fichier. Pour cela j'ai utilisé un plugin "uploadr" où il a été nécessaire de modifier certaines méthodes(en javascript et groovy) pour pouvoir l'adapter au mieux à cette situation. Ainsi les fichiers sont envoyés directement au serveur pour être stockés dans un dossier temporaire et être ensuite enregistré et associé au composant à la validation du formulaire.

5.2 Le backlog

Le backlog permet l'affichage des itérations et des releases avec leur stories associées, ainsi que les stories qui n'ont pas été rangées. J'ai développé le module hormis deux formulaires (création de paper work [1] et création de release) et la traitement du drag&drop qui permet le déplacement d'une story. Il n'y pas eu de grande difficulté, seulement un temps de développement relativement long car le module permet la création et la manipulation de plusieurs composants.

5.3 Le sous module userStory

Ce module s'est révélé plus complexe. Il permet la visualisation de toutes les informations d'une story et des tâches associées et leurs manipulations. Il est par exemple possible de splitter une storie afin d'obtenir les informations relative au logiciel de gestion (association commit/task). Les spécifications imposaient une fenêtre modale pour ce modules mais étant donné le grand nombre d'informations affichées, nous avons choisie de le placer sur toute une page. La difficulté du module reposait principalement dans la communication avec le logiciel de gestion (voir point suivant).

1. formulaire de création de template pour les user stories permettant leur réutilisation

5.4 Association avec les logiciels de versions

Un des avantages d'AgileFactory est son association bidirectionnelle qui permet de retrouver à partir d'un commit l'exigence du client et inversement. On garde ainsi une trace de ce qui a été fait et pourquoi cela a été fait. J'ai développé la communication avec GIT (SVN étant prévu plus tard). Git n'offre pas la possibilité de récupérer d'une façon automatique l'historique des commits se trouvant sur le dépôt du serveur distant. Il a donc fallu développer une solution de mise en relation de l'application avec le serveur de dépôt. A l'aide d'un plugin (Jsch), on établie une connexion ssh avec le serveur à l'aide de son adresse IP, d'un login et d'un mot de passe. Ces information sont au préalable rentrées par l'utilisateur qui indique également le chemin d'accès au répertoire GIT sur le serveur. Après la connexion, on exécute une commande, à l'aide de la classe ChannelExec de Jsch, pour atteindre le dépôt et effectuer une commande GIT pour récupérer les logs des commit en imposant un format qui permet de les parser et d'enregistrer les informations dans le base de données. Une autre commande est exécuté pour récupérer les diffs [2] associés au commit.

5.5 Plugins

Pour répondre au besoin de l'application il m'a fallut choisir, modifier et travailler sur des plugins. Grails permet une installation simples des plugins, notamment quand ils sont dédiés au framework. Il suffit de rentrer dans un des fichiers de configuration (BuildConfig.groovy) "runtime +'le nom du plugin suivi de son numéro de verison'".

5.5.1 Plugin uploader

Ce plugin permet l'upload de fichiers avec une fonctionnalité de drag&drop. Les limitations de sécurité de la balise imput (ici avec le type 'file') a posé un problème dans la mise en place de l'upload parce qu'il doit être multiple et pour plusieurs composants au sein d'un même formulaire. J'ai dans un premier temps développer une solution javascript (les fichiers sont tous envoyé au serveur après la validation du formulaire) avant d'installer le plugin, de le modifier pour l'adapter à nos besoins. La principales difficulté se trouvaient dans le fait que l'utilisateur charge des fichiers, pour un scénario par exemple, qui sont envoyés au serveur avant que le scénario ait été sauvegarder en base. Il a donc fallu modifier le plugin pour pour pouvoir retrouver l'association fichier/scénario sur le serveur.

5.5.2 Plugin présentation de l'arbre

Cet autre plugin permet la visualisation de la matrice bidirectionnelle sous la forme d'un arbre [3], dont les noeuds peuvent être réduits. Le plugin utilise la balise canvas pour dessiner les branche et les noeuds. L'outil répondait bien à nos besoins, et permettait une recherche par mots clef dans l'arbre mais le problème est qu'il n'était pas compatible avec

2. Il existe une commande GIT nommée diff qui permet d'avoir l'information sur le changement de code opéré par un commit dans un fichier.
3. voir imprime écran dans l'annexe

chrome. Il a donc fallu le rendre compatible. J'ai dû travailler aussi sur l'affichage qui posait un problème lorsque l'utilisateur effectue un scroll sur la page (le dessein disparaissait à la limite du début du scroll). La solution a été de rendre le dessin dynamique quand un événement scroll est détecté.

Conclusion et retour d'expérience

6

L'application AgileFactory sera toujours en développement après mon départ. Il y a encore de nombreuses évolutions prévues (comme l'association avec le logiciel de gestion SVN). Ce stage m'aura donc permis de progresser en termes de méthodologie agile et technique. J'ai donc eu la chance de suivre un projet interne pratiquement du début à la fin, de participer à sa conception et aux réflexions de l'équipe sur les points délicats. J'ai apprécié de travailler en mode agile, dont le concept d'auto-organisation est novateur et je pense qu'il tiendra une place de plus en plus importante dans la réalisation de projets numériques. Il permet au développeur de se sentir concerné par le produit.

L'utilisation de frameworks m'a permis d'appréhender le principe de productivité et de veille technologique. L'environnement JEE est vaste et l'utilisation d'outils tels que Grails est appréciée dans les entreprises. De plus la montée des supports tablettes et smartphones pousse les applications vers le multiplate-formes et Bootstrap aura été une bonne introduction au responsive design. La personne qui m'a encadré a été un atout pour mon stage car il a en lui la volonté de partager.

Le principal problème du stage aura été matériel : à mon arrivée on m'a attribué un poste qui n'était pas du tout à la hauteur de la puissance nécessaire pour les outils qu'on a utilisé (seulement 2 giga de RAM). Malgré mon instance pour changer de PC, rien n'a été fait. On a donc fini par installer xubuntu qui est moins vorace en puissance. Ce problème a conduit à perdre environ 2 heures de développement par jour pendant deux mois puis environ 1 heure par jour, après le passage à linux, ce qui a été une vraie source de frustration. Une entité telle qu'Astek cherche à attirer les développeurs et ce genre de problème d'organisation du matériel ne peut que nuire à la réputation de l'entreprise.

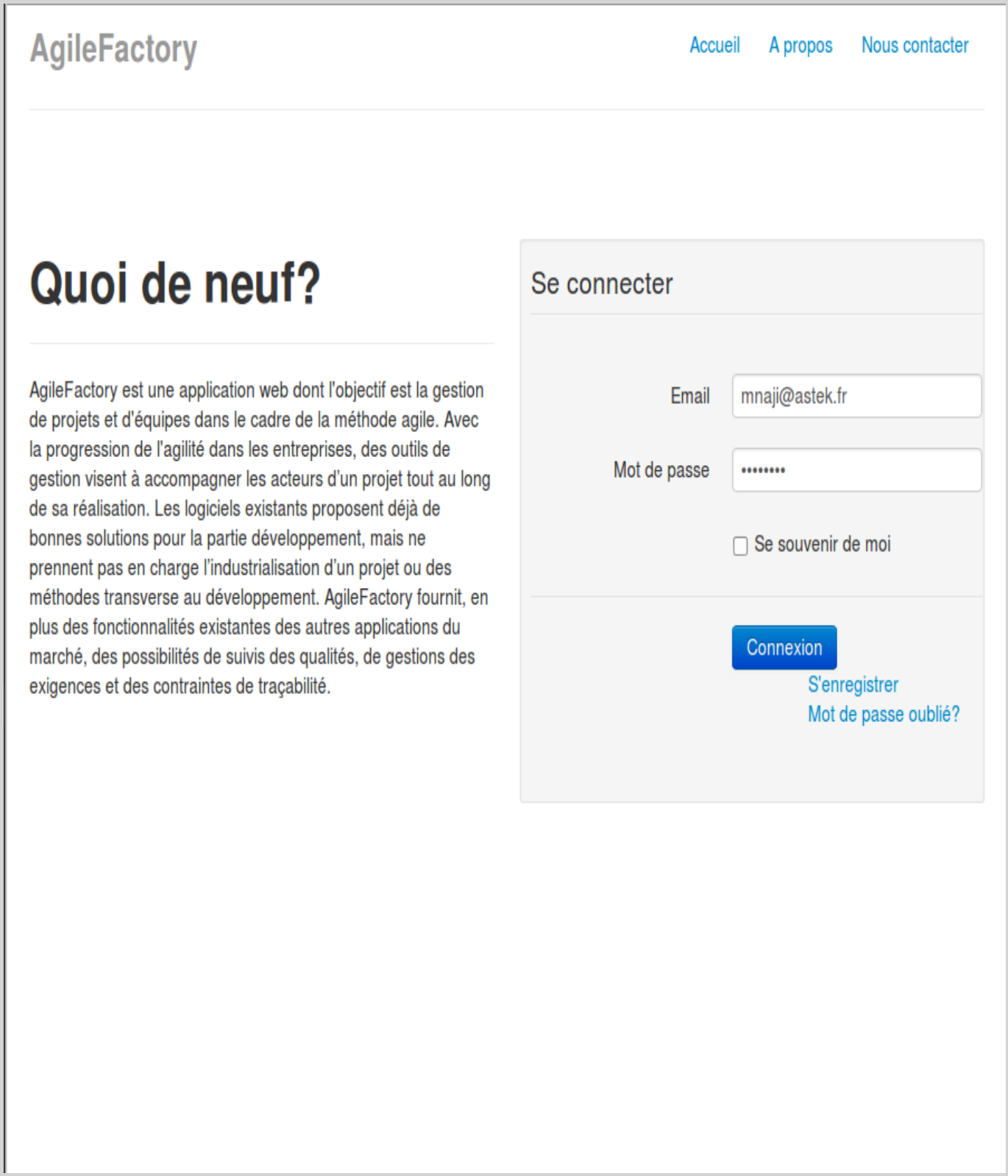
Le monde des SSII possède ses propres codes socioprofessionnels qu'il faut savoir intégrer pour y progresser. Je n'ai pas eu de problèmes particuliers à ce niveau là. L'entreprise n'a pas proposé de contrat aux stagiaires qui ont travailler dans les locaux de l'agence mais a proposer de reprendre contact courant octobre pour discuter de cette éventualité. Voulant continuer sur un master recherche j'ai décliné la proposition. Cet expérience professionnelle m'aura été vraiment bénéfique même si je ne souhaite pas faire carrière dans une société de service. J'ai néanmoins acquis des compétences que je vais essayer d'intégrer au mieux dans ma poursuite d'études en recherche. L'écriture de ce rapport aura été également l'occasion de travailler en latex.

Table des figures

Liste des tableaux

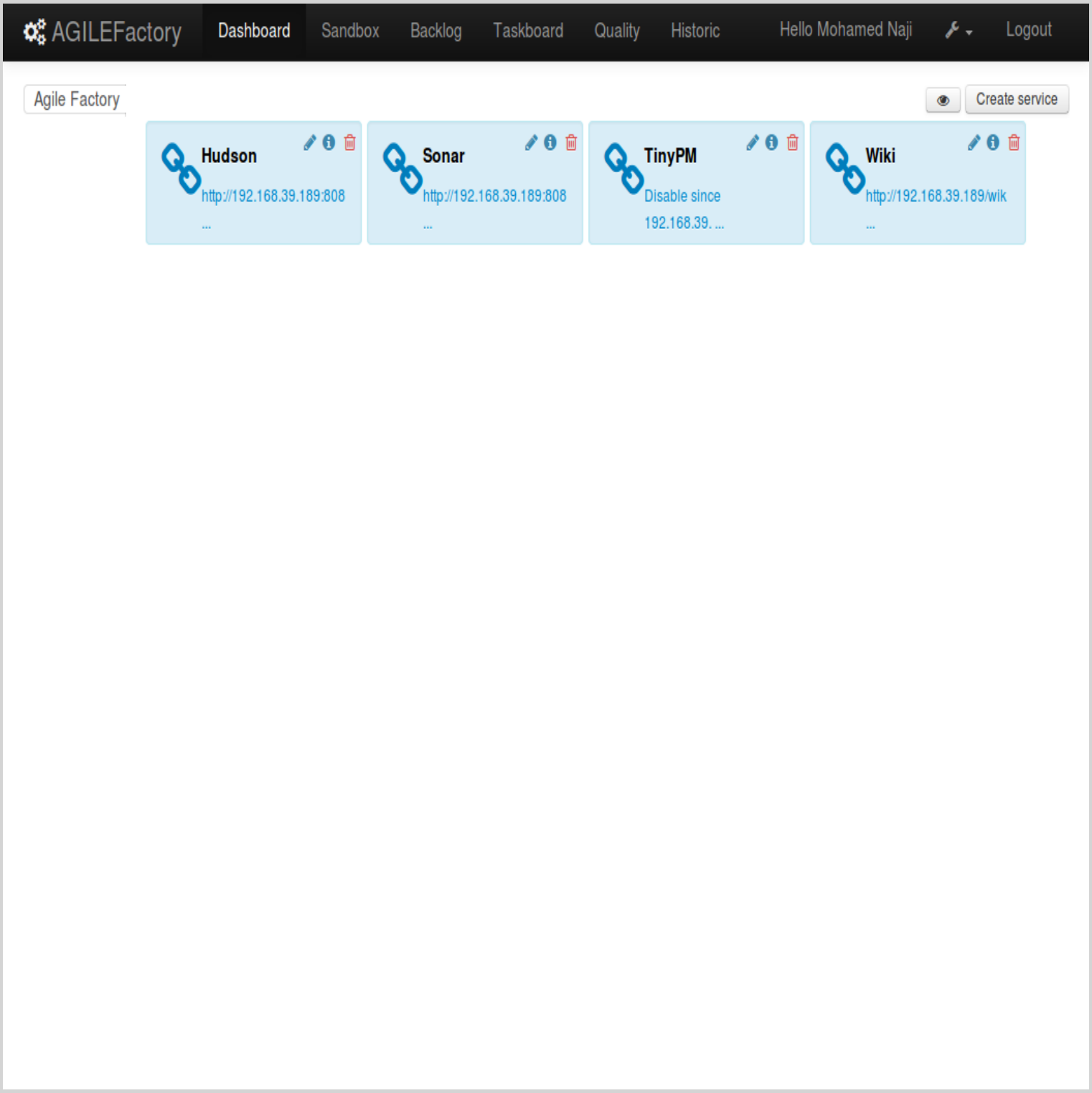
Vue de l'interface de connexion

A



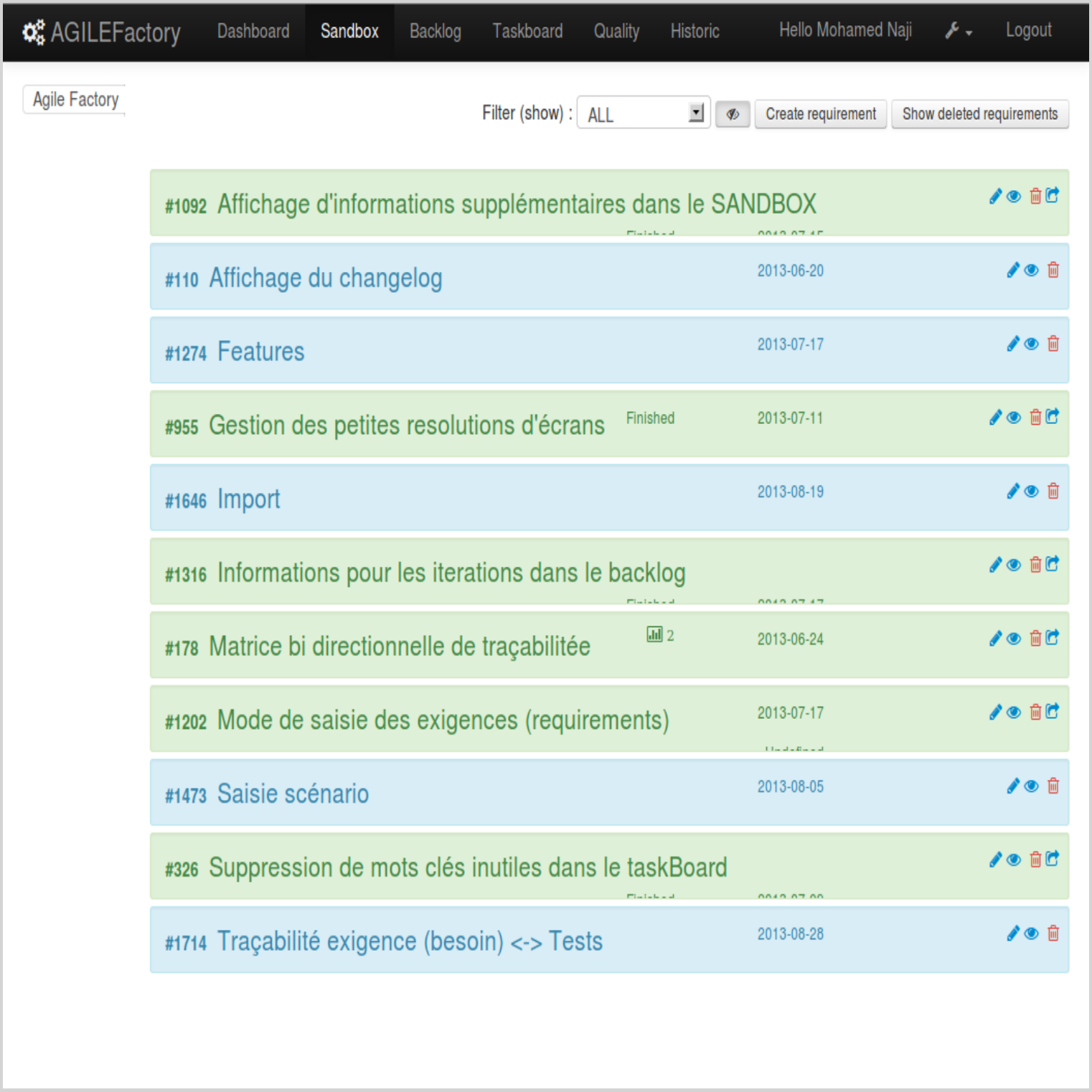
Vue de l'interface du module dashboard

B



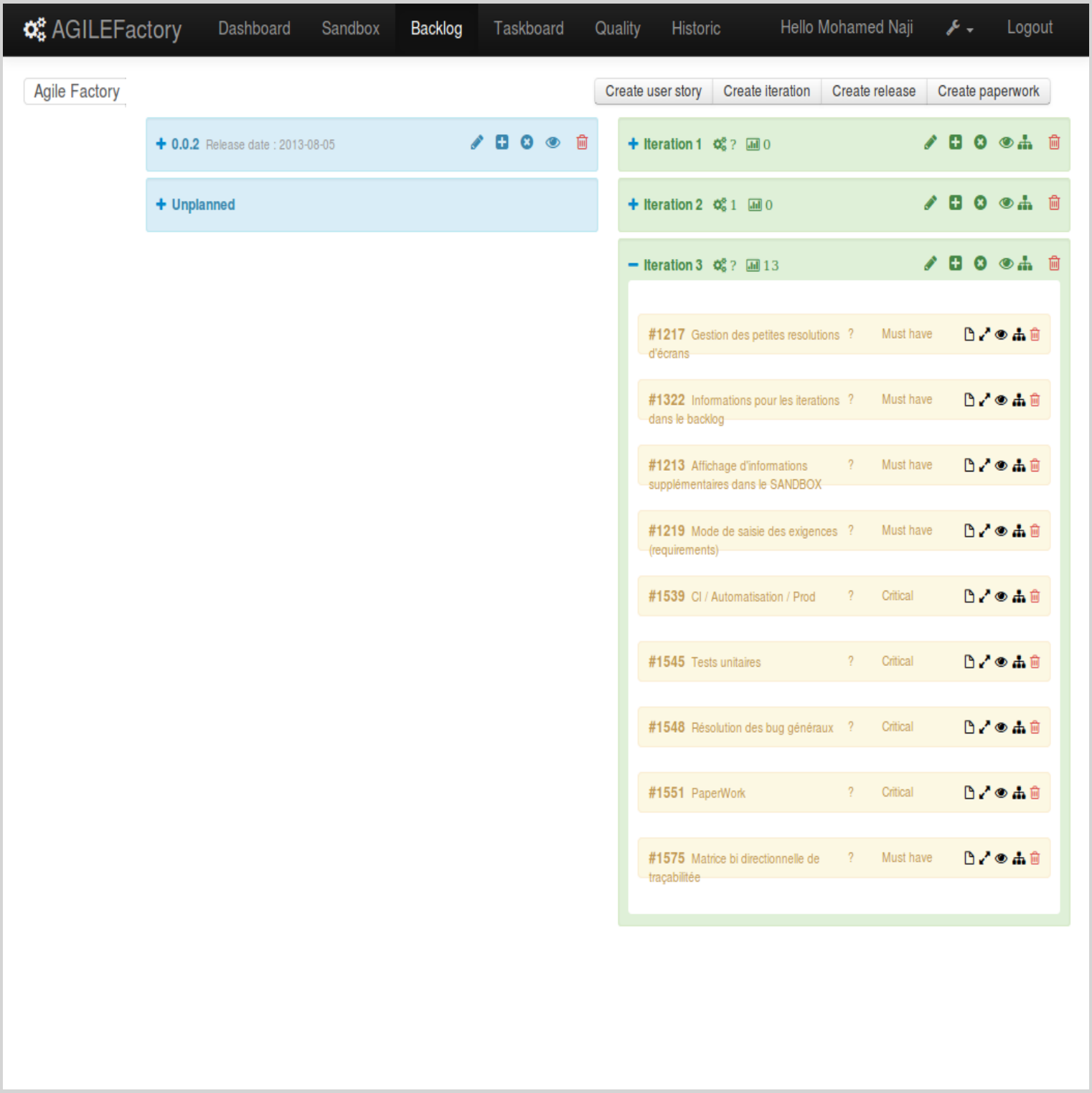
Vue de l'interface du module sandbox

C



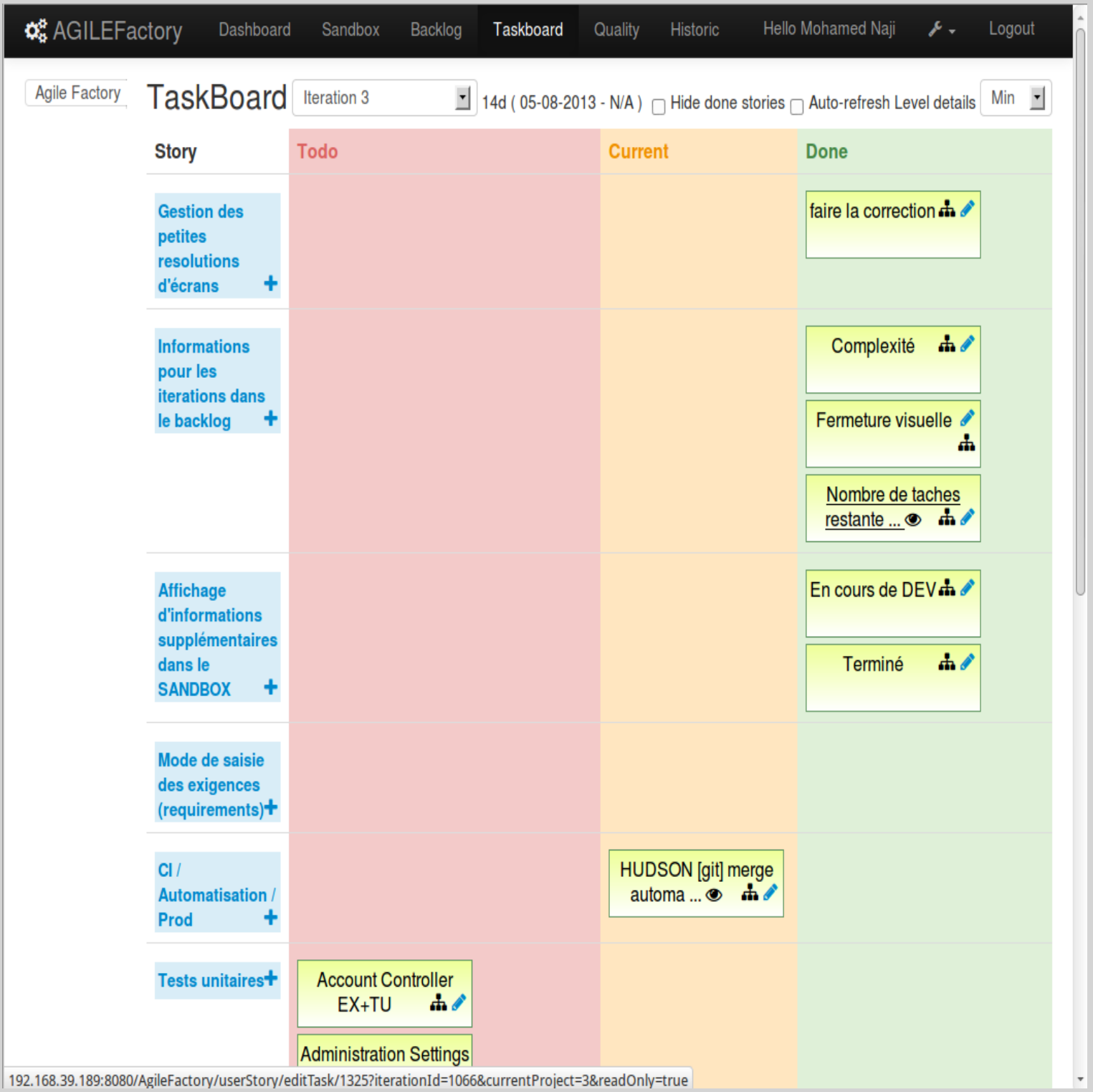
Vue de l'interface du module backlog

D



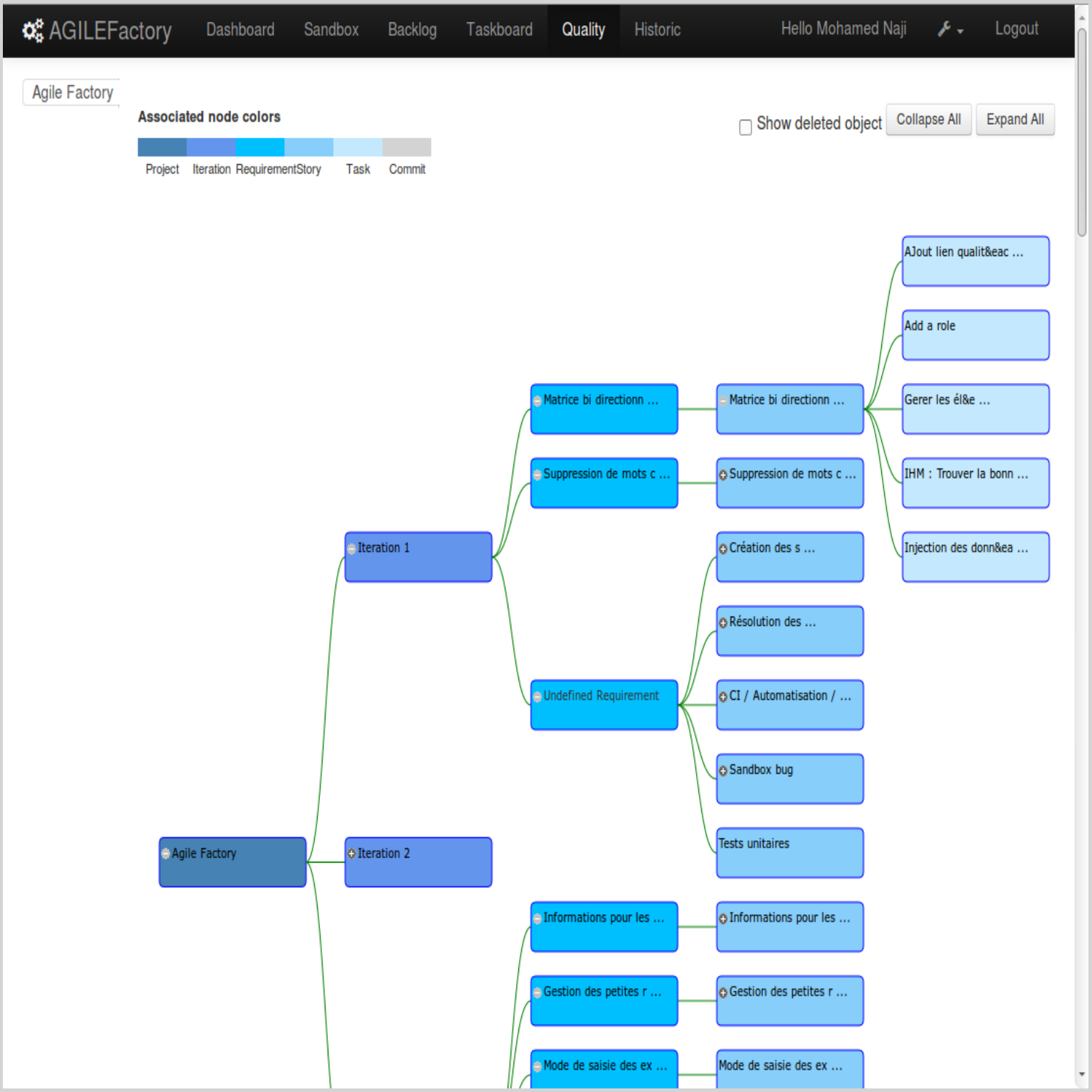
Vue de l'interface du module taskboard

E



Vue de l’interface du module quality

F



Vue de l'interface du module historic

G

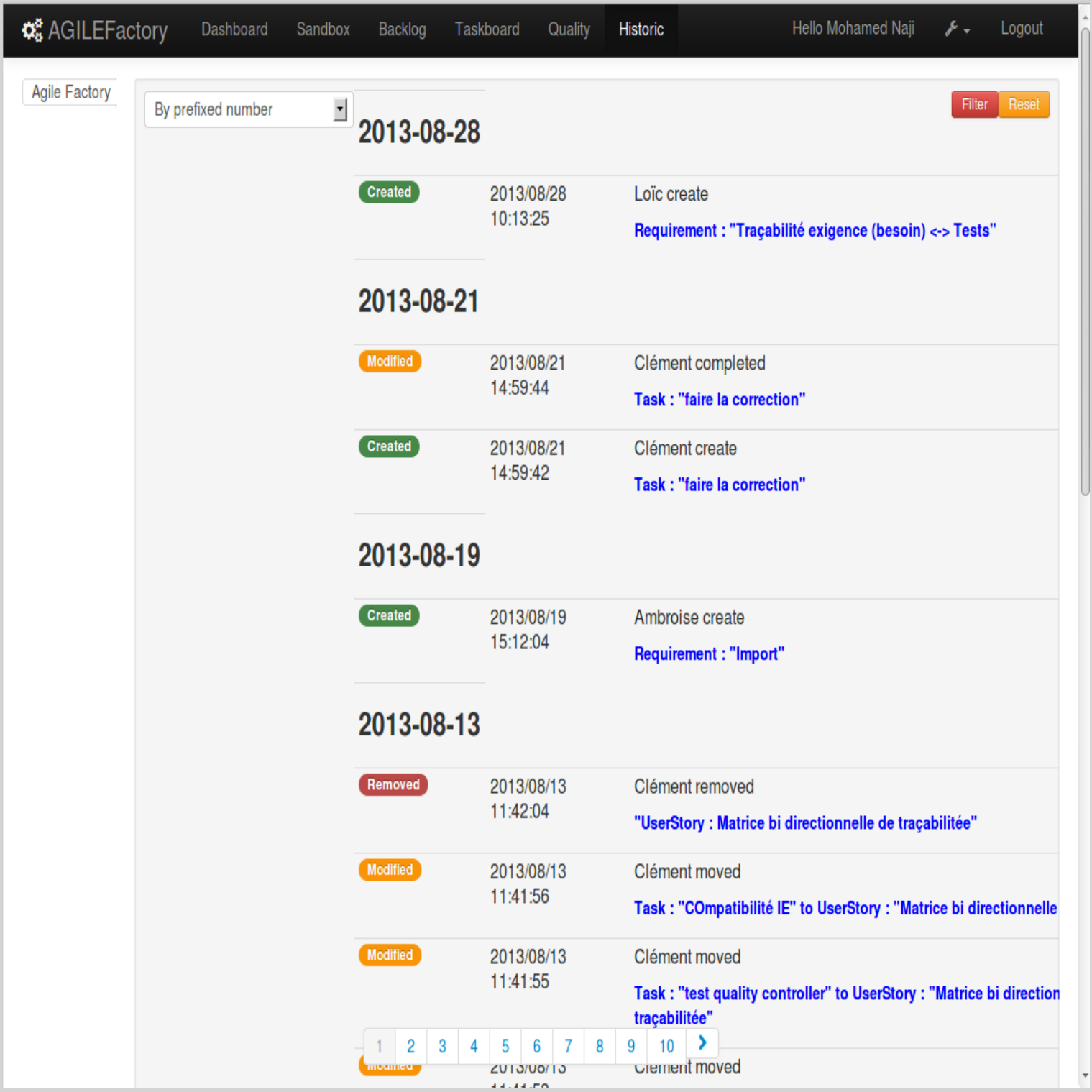
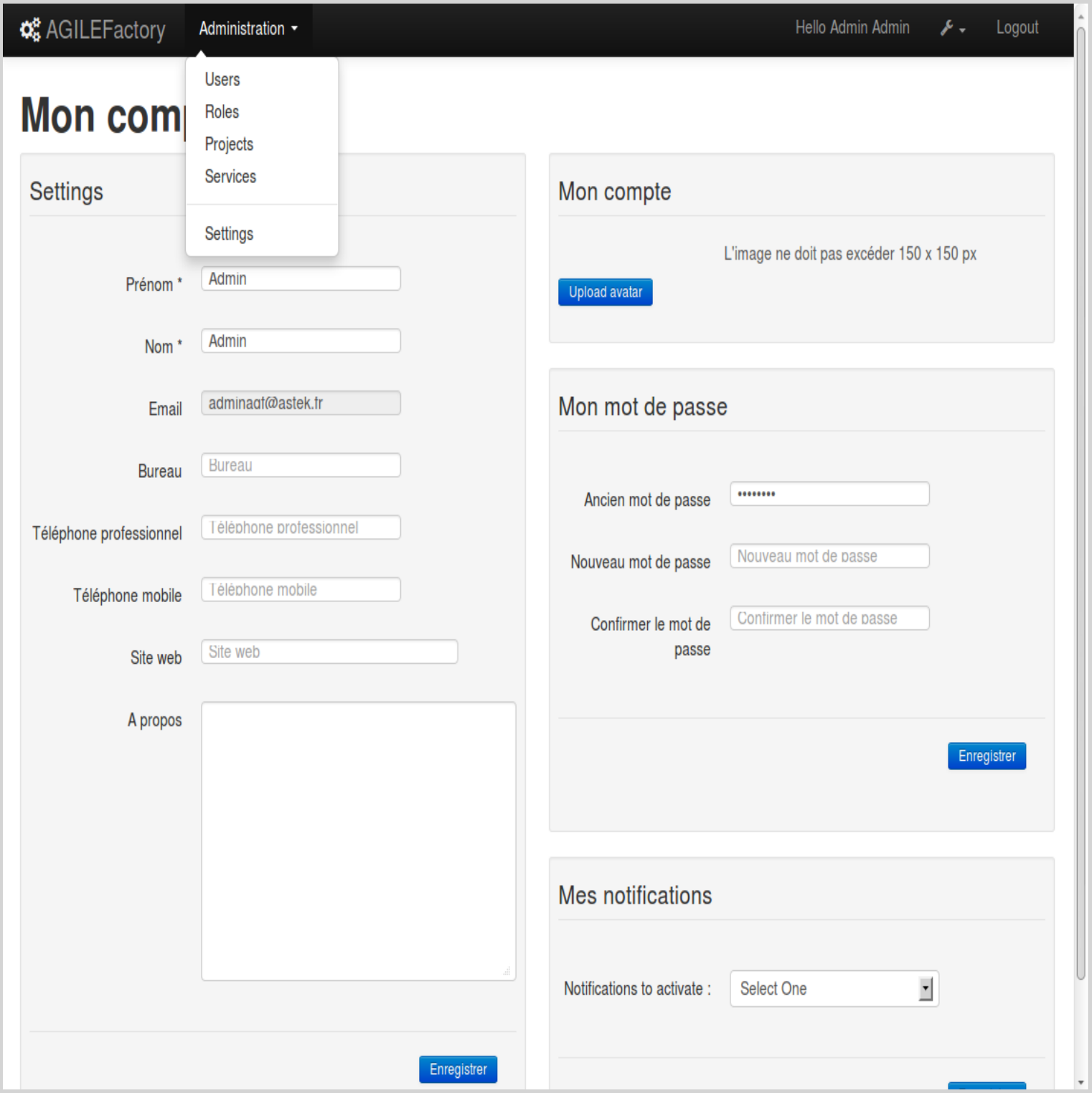


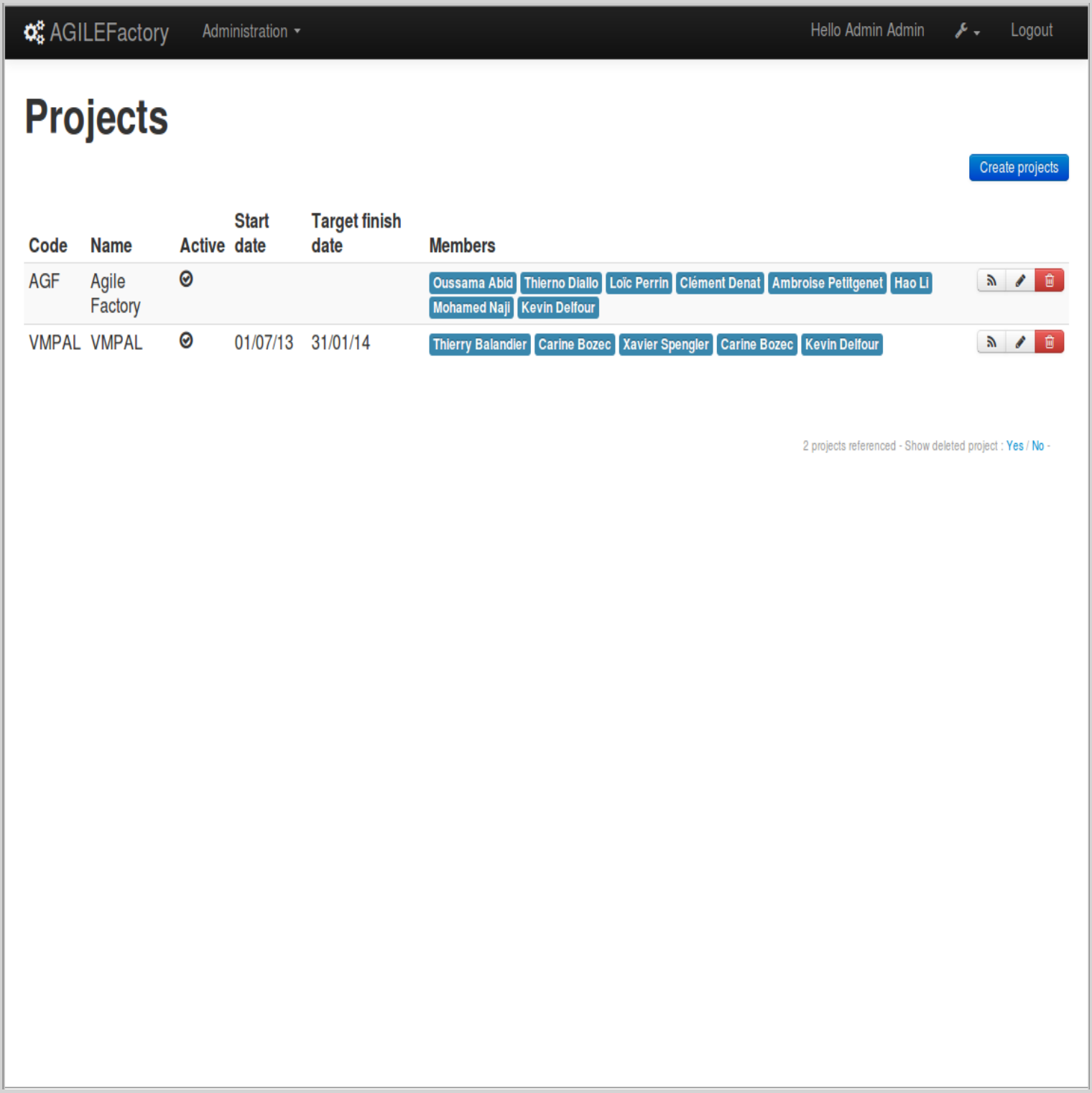
FIGURE G.1. : Formulaire de création de requirements sans scénario

Vue de l'interface de configuration

H

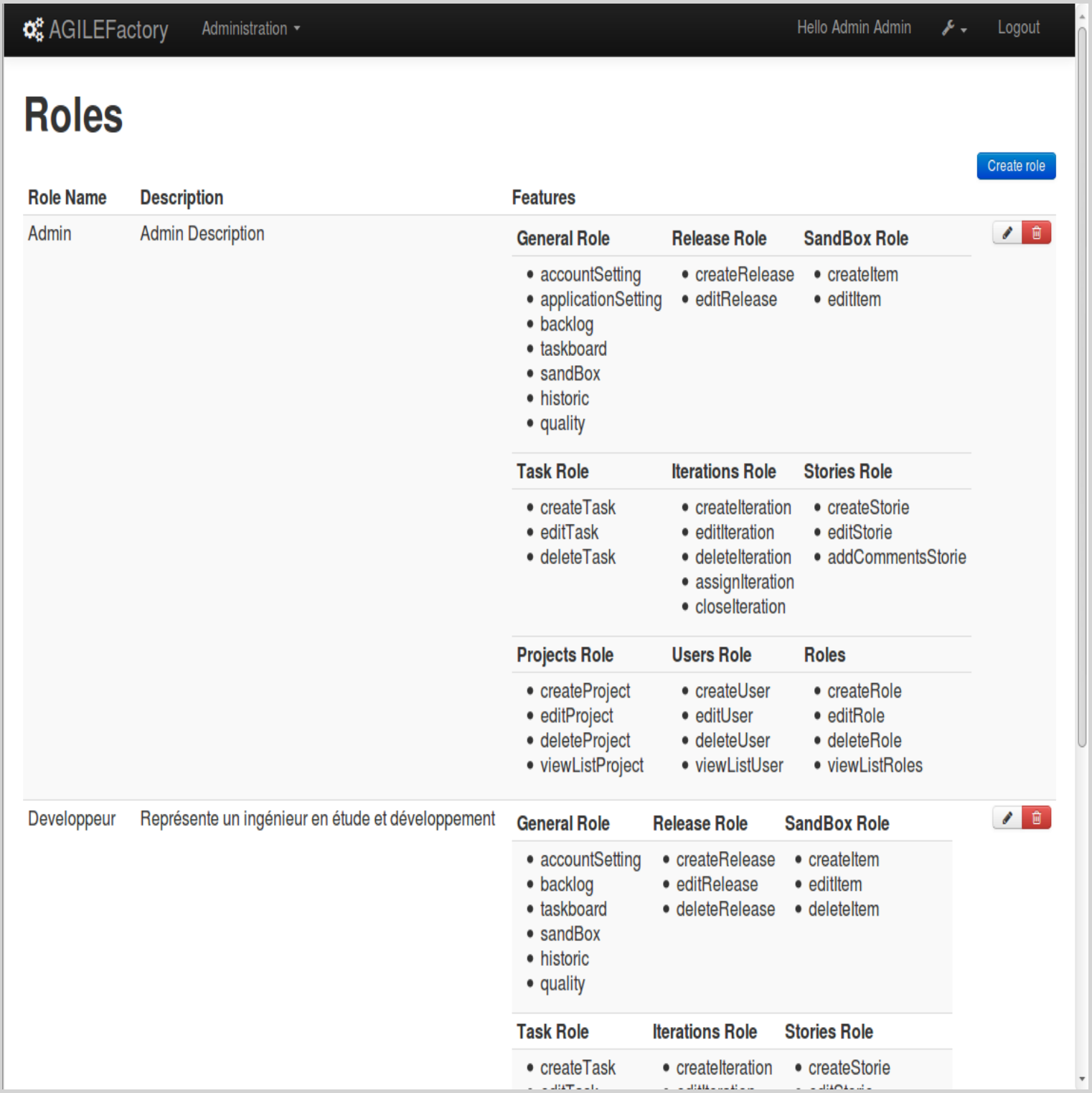


Vue de l'interface liste de projet



Vue de l'interface gestion des rôle

J



K

Vue de la matrice bidirectionnelle avec les commits