



Lyon 1

2016-2017

M1 INFORMATIQUE

PROJET TRANSVERSAL

RAPPORT - LIVRABLE FINAL

COLLABORATEURS

FIRAS ODEH
ARTHUR POIRET
ISABELLE FLORES
KASSIM HASSANI
MATHILDE BOLTENHAGEN
THOMAS FROMONT
STEVEN LEFEBVRE

RESPONSABLES

AURELIEN TABARD
LIONEL MEDINI
EMMANUEL COQUERY

EWIDE

25 novembre 2016

Introduction

Au cours de 4 semaines de conception et développement, notre équipe, constituée des collaborateurs: **FIRAS ODEH, ARTHUR POIRET, ISABELLE FLORES, KASSIM HASSANI, THOMAS FROMONT, MATHILDE BOLTENHAGEN, STEVEN LEFEBVRE (Groupe 5)** a mis au point un Environnement Web Intégré de Développement — ou Embedded Web Integrated Development Environment, *EWIDE* —, intégralement utilisable sur navigateur, et compatible avec tout système d'exploitation.

Objectifs

1. **Edition de projets:** Permettre la modification de fichiers inhérents à un projet de développement, directement depuis une page Web.
2. **Collaboration:** Intervention en simultané de plusieurs acteurs sur un même projet, avec gestion de droits, et outils collaboratifs divers (chat, wiki, tâches).
3. **Gestion de versions:** Les acteurs sont en mesure de consulter les versions antérieures de leurs fichiers depuis leur création, et de les restaurer au besoin.

Nous traiterons, dans la suite de ce rapport, des technologies et modes d'implémentations des fonctionnalités susnommées, employées au cours du projet.

SOMMAIRE

Introduction	2
Objectifs	2
I. Gestion de projet	5
1. Méthode Agile: SCRUM	5
a. Mêlées et réunions	5
b. Point de vue Client	6
c. Outils de collaboration	6
2. Suivi de développement	6
a. Diagramme de Gantt	6
b. Tests unitaires	7
c. SonarQube / SonarLint	7
II. Serveurs et technologies employées	8
1. Configuration des serveurs	8
a. Apache Tomcat	8
b. H2	8
2. Technologies employées	9
a. Spring	9
b. Hibernate	9
c. Bootstrap	9
d. JUnit	9
e. DBUnit	9
f. Git & JGit	10
g. CKEditor et Ace	10
III. Conception et développement	11
1. Architecture	11
a. Maven	11
b. Spring	11
c. Packages	12
d. Sécurité et authentification	12
2. Base de Données	13
a. Structure	13
b. Tests unitaires et d'intégration	14

IV. Fonctionnalités en images	15
1. Connexion et inscription	15
2. Changement de mot de passe	16
3. Création d'un projet	17
4. Gestion des utilisateurs d'un projet	18
5. Wiki	19
6. Gestion des tâches	20
7. Editeur de code	21
8. Conserver et gérer les versions du code	22
9. Compiler le code	23
10. Chatter	24
V. Améliorations	25
1. Tests	25
2. Sécurité	25
3. Description des projets	25
4. Sous-tâche	25
5. Dossiers personnels	26
6. Télécharger l'exécutable	26
VI. Conclusion	27

I. Gestion de projet

1. Méthode Agile: SCRUM

La méthode Scrum est une méthode agile qui suit un fonctionnement particulier. Le client explicite son besoin et les ressources allouables (temps/budget), traduits par le prestataire en un backlog produit, décrivant l'importance et la complexité des points-clés du projet. L'équipe de développement définit une durée de sprint, période de travaux au bout de laquelle les collaborateurs se réunissent pour en suivre l'avancement, et réviser le backlog en conséquence.

Un élément de l'équipe de développement est désigné "scrum master". Il anime des réunions appelées mêlées dans lesquelles chaque membre s'attribue une tâche pour le sprint. Dans notre cas, ce fût Mathilde Boltenhagen qui a encadré le projet, a planifié nos différentes réunions et suivi les différents modules afin de garder une bonne cohésion entre le code des différents membres.

a. Mêlées et réunions

Au cours de brèves réunions, Mathilde animait un tour de table où chaque membre de l'équipe rappelait ce qu'il devait produire, ce qu'il avait effectivement produit, les difficultés qu'il avait rencontrées et les conclusions de son travail. Ces concertations permettent de moduler les développements. Nous avons parfois associé des développeurs afin de fusionner des modules ou se répartir les tâches. Par exemple, Firas Odeh et Kassim Hassani se sont répartis l'éditeur, l'un gérant le front-end, l'autre le back-end (système de fichiers), quand cela n'était attribué qu'à Kassim en premier lieu.

Après chaque réunion, Mathilde Boltenhagen rédigeait un compte rendu, puis les autres membres relisaient pour correction. Une fois validés, les comptes-rendus étaient postés sur la forge (section Fichiers).

Outre cela, des mêlées en fin de semaine permettaient d'établir un nouveau backlog, dont les collaborateurs se répartissaient les tâches pour la semaine suivante.

b. Point de vue Client

A l'implémentation d'une fonctionnalité, nous la soumettions aux critiques d'utilisateurs externes, lesquels jouaient les clients improvisés. Nous avons, dans ce même sens, fait tester notre application à un développeur externe à l'UE, pour mieux en cerner les potentiels défauts.

SCRUM se voulant souple, nous définissions nos backlogs en fonction des éléments essentiels de l'application, lorsque nécessaire. Par exemple, nous avons mis de côté certaines options, telles la gestion du compte (avatar, thème), non critique pour le livrable requis par le client.

c. Outils de collaboration

Principe essentiel des méthodes agiles, nous nous sommes appliqués à collaborer un maximum. Nous avons travaillé quotidiennement dans la même salle, presque invariablement, et avons communiqué dès que le besoin se présentait. En cas de communication à distance, nous avons utilisé *Slack* comme plateforme de discussion et partage d'informations.

La forge proposée par l'université fût notre outil de collaboration principal.

- Chaque membre notait la ou les tâches *principales* qui lui étaient attribuées, et devait les tenir à jour. Cela nous permettait un suivi global du développement.
- Nous avons développé un projet versionné avec mercurial, en essayant de travailler sur des branches différentes en cas de conflits.
- Nous nous sommes servis du wiki et du gestionnaire de documents, afin de partager des recherches ou méthodes de développement communes à l'équipe.

2. Suivi de développement

a. Diagramme de Gantt

En supplément du diagramme de Gantt proposé par la forge, nous avons créé un diagramme de Gantt par semaine, et par fonctionnalité, afin d'étudier l'évolution de celles-ci. Ces documents sont disponibles sur la forge, onglet Fichiers.

b. Tests unitaires

Nous avons implémenté des tests unitaires sur les DAO. Nous aurions pu faire des tests sur les requêtes avec une technologie type Gatling, mais la limite de temps ne nous a pas permis de l'implémenter. Une description approfondie des tests est disponible dans la section **Conception et développement** du document.

c. SonarQube / SonarLint

SonarQube est une plateforme d'analyse de code de projets. Elle établit automatiquement des rapports de bugs, risques de sécurité, duplications de code, etc. Un temps humain précieux est ainsi sauvegardé pour le debugging. Les développeurs sont à même, outre résoudre les problèmes relevés, de générer de nouveaux rapports pour suivre l'évolution de la correction des bugs.

Soumis à évaluation, notre projet a passé avec succès la Quality Gate par défaut proposée par la plateforme.

Une déclinaison de SonarQube intégrée à Eclipse, SonarLint, aura permis de s'émanciper de l'utilisation d'un serveur SonarQube classique, et de la génération régulière de rapports. En effet, SonarLint interprète le code rédigé en temps réel au sein de l'IDE.

II. Serveurs et technologies employées

1. Configuration des serveurs

Nous avons trouvé sage de travailler avec deux serveurs, pour prévenir tout crash de l'une des machines virtuelles (VMs) les hébergeant:

- **192.168.77.24** serveur de production port **80**
- **192.168.77.126** serveur de sauvegarde port **8080**

Sur chacune des VMs, nous avons créé un utilisateur non-administrateur, lequel a la permission de lancer le serveur d'application (utilisateur *multimif*).

a. Apache Tomcat

Notre choix de serveur s'est porté sur *Apache TomCat*, révision 7.0.72. Tomcat 7 convient tout à fait à notre application Java, étant compatible avec les versions des différentes technologies que nous avons utilisées.

b. H2

H2 est le moteur SQL employé pour la persistance des données d'application. Cette décision a principalement été motivée par la légèreté de la base de données; les VMs mises à disposition des collaborateurs offrent effectivement des ressources limitées, nécessitant dès lors des optimisations.

La base de données est sauvegardée toutes les nuits, du serveur de production vers le serveur de sauvegarde. Les répliquions sont organisées en sept dossiers, nommés 01 à 07, représentant les sept jours de la semaine. Une semaine de sauvegardes est ainsi conservée.

2. Technologies employées

a. Spring

Spring est un framework Java, permettant le développement d'applications JEE. Celui-ci offre un très haut niveau d'abstraction, et prend en charge l'injection de dépendances. La gestion d'un projet en est alors simplifiée.

b. Hibernate

Hibernate est notre framework de persistance — ou ORM: il lie la base de donnée et les modèles du projet. L'établissement de relations entre nos objets Java est en outre transparent, grâce au mappings des classes requis par Hibernate.

c. Bootstrap

Bootstrap était la solution évidente côté front-end. Tout d'abord parce qu'il est sans aucun doute le framework le plus populaire, et donc le mieux maîtrisé par tous les collaborateurs à l'heure actuelle. D'autre part, il gère dynamiquement le squelette des pages, et le rendu qu'il était possible d'obtenir nous a convaincu. Enfin, nous comptons un développeur particulièrement expérimenté dans le groupe.

d. JUnit

Lors du développement d'un projet informatique, les tests sont indispensables. A fortiori les tests unitaires, dans le cadre d'Extreme Programming. Pour un projet en Java, le framework *JUnit* est majoritairement adopté.

e. DBUnit

Lors des tests avec la base de données, il est important de s'enquérir de la stabilité des données, et que les requêtes n'influent pas sur la base en production. *DBUnit* permet cela, effectuant des tests sur son propre jeu de données — environnement clos initialisé puis supprimé à chaque test.

f. Git & JGit

Un client *git* se cache derrière les fonctionnalités de gestion de versions d'EWIDE. Steven Lefebvre a, de plus, intégré au projet la librairie Java *JGit*: il s'agit d'une passerelle entre le gestionnaire de versions, et le monde des objets. Concrètement, développer avec JGit a l'avantage — passé le mur de la faible documentation disponible —, d'améliorer la lisibilité du code pour un développeur peu familiarisé au gestionnaire.

A l'origine toutes effectuées via des appels systèmes directs à *git*, les commandes les plus complexes ont ainsi pu être foncièrement simplifiées.

g. CKEditor et Ace

EWIDE contient deux éditeurs différents: celui du wiki, et l'éditeur de code. Leurs fonctionnalités diffèrent en ce que wiki requiert un outil de traitement de texte, et l'éditeur la complétion/coloration de code. *CKEditor* et *Ace*, deux librairies JavaScript, offrent respectivement ces options aux deux modules.

III. Conception et développement

1. Architecture

a. Maven

Ewide prend la forme d'une Application Web Maven. L'outil permet notamment la compilation, l'exécution de tests, et le déploiement de l'application sur un serveur; il gère par la même les dépendances du projet et le classpath.

Une commande — cible — nous permet, soit de lancer le projet en local pour tester notre développement, soit de déployer directement sur le serveur Tomcat de "production".

b. Spring

Notre application est réalisée avec Java Enterprise Edition, et emploie le conteneur Java Spring. A noter que l'application est structurée selon le design pattern MVC (cf. Figure 1).

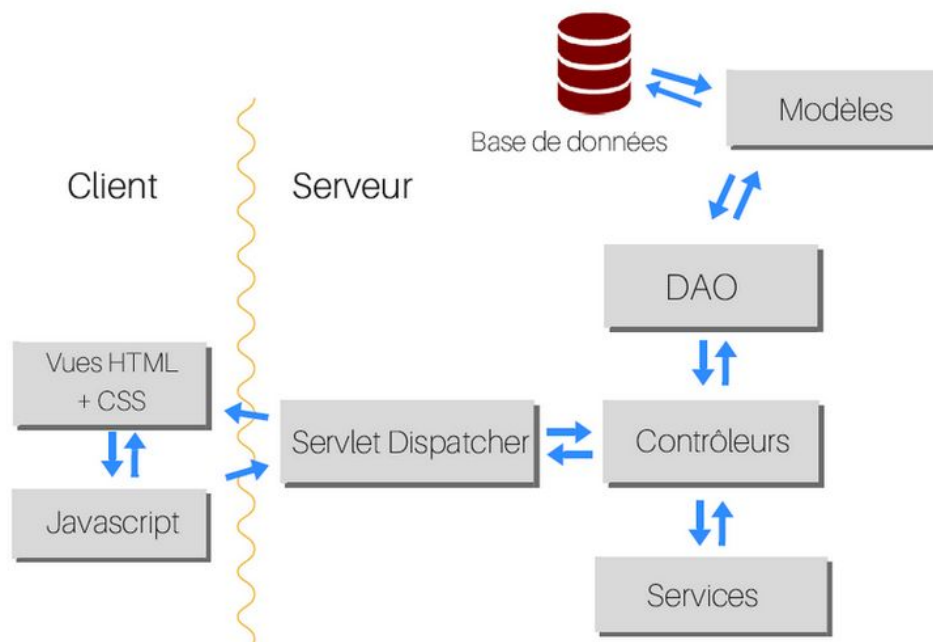


Figure 1 : Architecture du projet

Côté serveur, les requêtes HTTP sont redirigées sur différents contrôleurs en fonction de leurs URLs. Les contrôleurs font alors appel à des dépendances – DAO, Services, Modèles,... – pour traiter les requêtes, et construire les réponses renvoyées au client. Un mécanisme d'injection de Spring fournit ces dépendances aux contrôleurs.

c. Packages

- Les DAO fournissent aux contrôleurs des fonctionnalités de persistance. Chaque modèle dispose de son propre DAO. Ce package aura évolué au rythme des modifications du schéma de base de données, pour en refléter les mises à jours.
- Les Services contiennent des méthodes nécessaires aux contrôleurs. Voici les différents services qui ont été implémentés :
 - AuthenticationService : Informations relatives à l'utilisateur courant.
 - UserRoleService : Rôle de l'utilisateur pour un projet donné, gestion des permissions.
 - GitService : Méthodes pour la gestion de dépôts git, et versionning des fichiers d'un projet.
- Des outils ont été créés dans un package séparé afin de ne pas surcharger les contrôleurs. Le package utils contient notamment la création d'un fichier zip.

d. Sécurité et authentification

Différents mécanismes de sécurité ont été introduits, en particulier pour identifier les utilisateurs, et s'assurer qu'ils disposent des droits suffisants selon le contexte. Plusieurs niveaux de permissions sont ainsi définis, selon le rôle de l'utilisateur.

L'authentification des utilisateurs est assurée par SpringSecurity, un framework gérant la couche de sécurité sur une application Spring. L'ensemble des requêtes passent par un filtre, lequel détermine si l'utilisateur peut accéder hors authentification à une ressource spécifique. Dans notre application, seules les pages d'accueil et de connexion sont accessibles aux utilisateurs anonymes.

Les contrôleurs sont annotés avec des fonctions d'évaluation de permissions. Cela permet de s'assurer que l'utilisateur courant peut accéder aux entités qu'il requête. Par exemple, seuls les membres d'un projet peuvent accéder à l'éditeur de code de ce projet. Si la condition n'est pas satisfaite, l'utilisateur est redirigé vers la page /access-denied.

2. Base de Données

a. Structure

La base de données est un élément essentiel à toute application JEE. Elle a été globalement définie en début de projet, puis réformée au cours du développement, selon les besoins (cf. Figure 2).

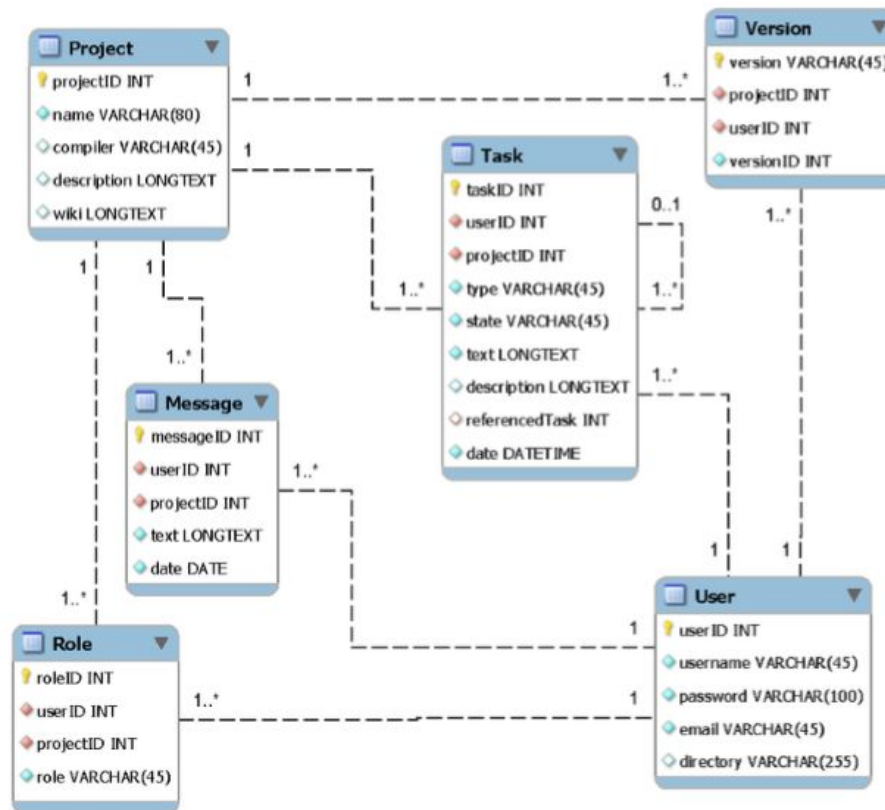


Figure 2 : Base de données finale d'EWIDE

Sur EWIDE, un utilisateur (User) peut créer un ou plusieurs projet(s) (Project), et en partager l'accès à des utilisateurs inscrits. Ce partage nécessite, pour un projet donné, l'affectation de rôles (Role) à d'autres utilisateurs de la plateforme.

Un projet EWIDE dispose d'une liste de tâches (Task) adjointes, et d'une liste de messages envoyés sur le Chat (Message), toutes deux liées à un utilisateur.

La table Version, quant-à-elle, fait le lien entre les dépôts git des projets, et la structure interne (Users, Projects) d'EWIDE. Elle permet d'associer à un ID de version unique, généré par *git*, un numéro de version intelligible par les utilisateurs, propre à chaque projet, ainsi que l'auteur de la version du projet.

b. Tests unitaires et d'intégration

Lors du développement agile, les tests unitaires et d'intégration prennent une place importante pour éviter la régression du code au fur à mesure du développement de nouvelles fonctionnalités.

Dans notre application, nous avons implémenté les tests unitaires via le framework JUnit, et les tests d'intégration de la BDD avec DBUnit. Tous les DAOs, et certains contrôleurs, ont une vérification sur la durée de leur fonctionnalité.

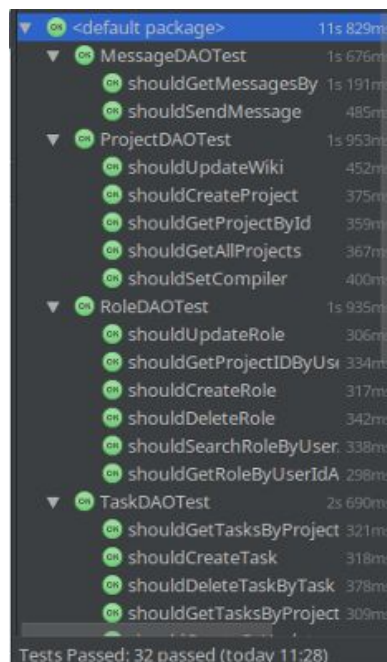
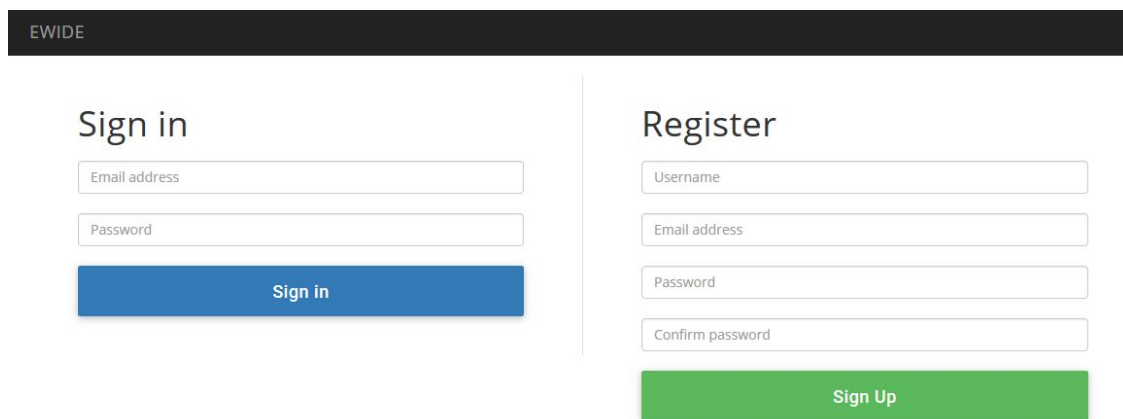


Figure 3 : Validation des Tests

L'architecture et les méthodes de développement présentées précédemment ont abouti en un livrable fonctionnel, offrant de nombreuses options.

IV. Fonctionnalités en images

1. Connexion et inscription



The image shows a web interface for EWIDE. At the top, there is a black header bar with the text "EWIDE" in white. Below the header, the page is divided into two columns. The left column is titled "Sign in" and contains two input fields: "Email address" and "Password". Below these fields is a blue button labeled "Sign in". The right column is titled "Register" and contains three input fields: "Username", "Email address", and "Password". Below these fields is a green button labeled "Sign Up".

Figure 4 : connexion/inscription

La connexion et l'inscription sont accessibles via la requête : GET /ewide/login. La page est composée de deux formulaires, dont les données sont évaluées côté serveur en Java. Lors d'une bonne authentification, l'utilisateur est renvoyé sur son tableau de bord : /ewide/dashboard (cf. Figure 5).

2. Changement de mot de passe

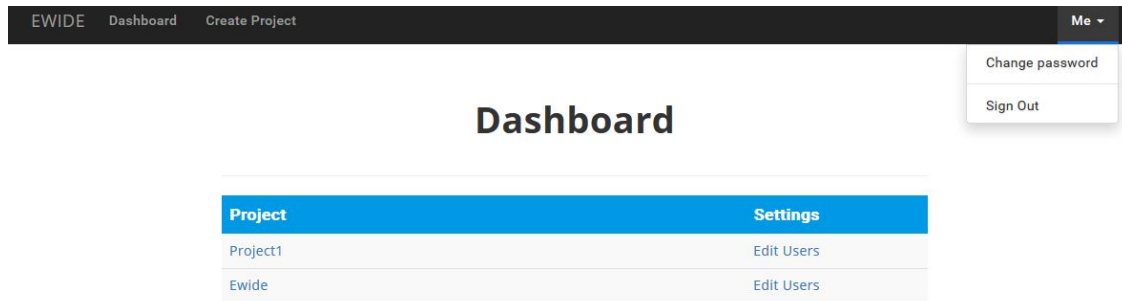


Figure 5 : Dashboard

Un utilisateur a la possibilité de changer son mot de passe grâce à un clic sur le menu déroulant “me” (cf. Figure 5).

Il accède ainsi à une nouvelle page, et peut renseigner un nouveau mot de passe. Celui-ci est vérifié par le contrôleur en charge, et une réponse indique à l'utilisateur si son changement est validé, ou comporte des champs erronés.

The screenshot shows the 'Change password' form. At the top, there is a navigation bar with 'EWIDE', 'Dashboard', and 'Me' links. The main heading 'Change password' is centered. Below the heading, there is a form with three input fields: 'Actual password:', 'New password:', and 'Confirm password:'. Each field contains five dots representing masked characters. Above the 'Actual password:' field, there is a red warning message: 'Minimum length of characters'. At the bottom of the form, there is a green 'Confirm' button.

Figure 6 : Changer de mot de passe

3. Création d'un projet

Un bouton "Create project" est disponible dans la barre supérieure, sur le Dashboard (cf. Figure 5). Un clic sur le bouton ouvre une page contenant un formulaire pour le nom, et une brève description du projet. L'utilisateur qui crée le projet en est le manager par défaut.

Le projet est alors ouvert et propose un menu avec différentes options (cf. Figure 7).

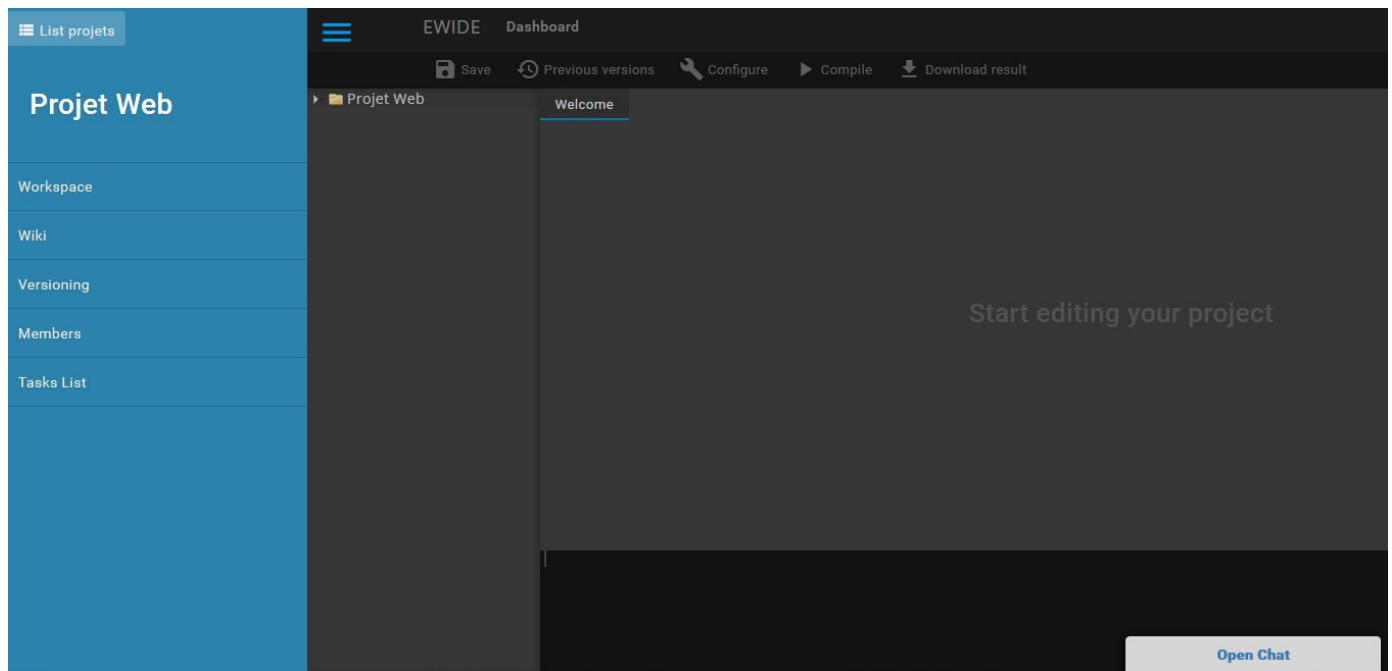


Figure 7 : Workspace et Menu d'un projet

4. Gestion des utilisateurs d'un projet

Dans la page Members, accessible via le volet glissant de gauche, ou directement dans la liste des projets, la liste des membres ayant un rôle dans le projet courant est disponible. Si l'utilisateur est manager, il peut alors modifier les rôles (cf. Figure 8). Sinon, il a accès à une liste d'utilisateurs et leur fonction.

Pour ajouter un membre, le manager utilise la barre de recherche et une fonction d'auto-complétion lui simplifie la recherche parmi les utilisateurs inscrits d'EWIDE.

The screenshot shows the 'Utilisateurs' (Users) management page. At the top, there is a dark navigation bar with 'EWIDE Dashboard' on the left and a user profile 'Me' with a dropdown arrow on the right. Below this, the title 'Utilisateurs' is centered. The main content area features a search bar with the placeholder 'Search a user...' and an 'Add' button. Below the search bar is a table with two columns: 'Pseudo' and 'Role'. The table contains two rows: one for 'fry' with the role 'MANAGER' and another for 'zoidberg' with the role 'DEVELOPER'. Each row has a red button with a trash icon to its right. At the bottom of the table is a large green 'Save' button.

Pseudo	Role	
fry	MANAGER	
zoidberg	DEVELOPER	

Figure 8 : gestion des utilisateurs

Un projet ayant plusieurs développeurs doit avoir un manager. Un manager peut ajouter et supprimer un autre manager.

5. Wiki

Le Wiki d'un projet est accessible via le volet glissant gauche.

CKEditor, une extension JavaScript, est employée comme outil de traitement de texte enrichi. Ainsi, nous pouvons appliquer différents effets de styles, ajouter des liens ou encore des tableaux (figure 9) au sein de zone d'édition.

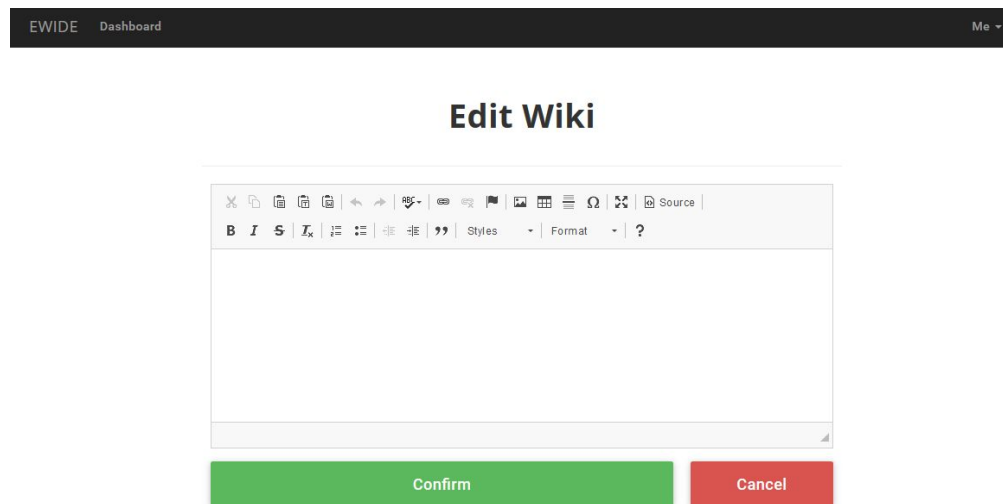


Figure 9 : Editeur du Wiki

6. Gestion des tâches

Les tâches d'un projet sont accessibles via le volet glissant gauche. Par défaut, les tâches actives du projet sont affichées. L'utilisateur peut ajouter une nouvelle tâche, modifier, ou fermer directement une tâche active.

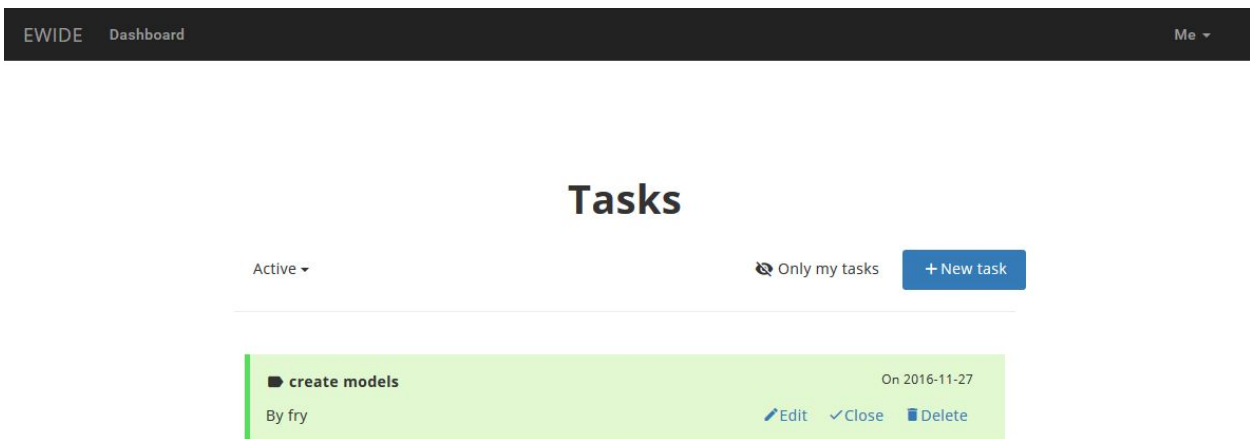


Figure 10 : Liste des tâches

Lors de la création d'une tâche, l'utilisateur doit saisir un titre, un type de tâche et un statut.

Figure 11 : Création d'une tâche

7. Editeur de code

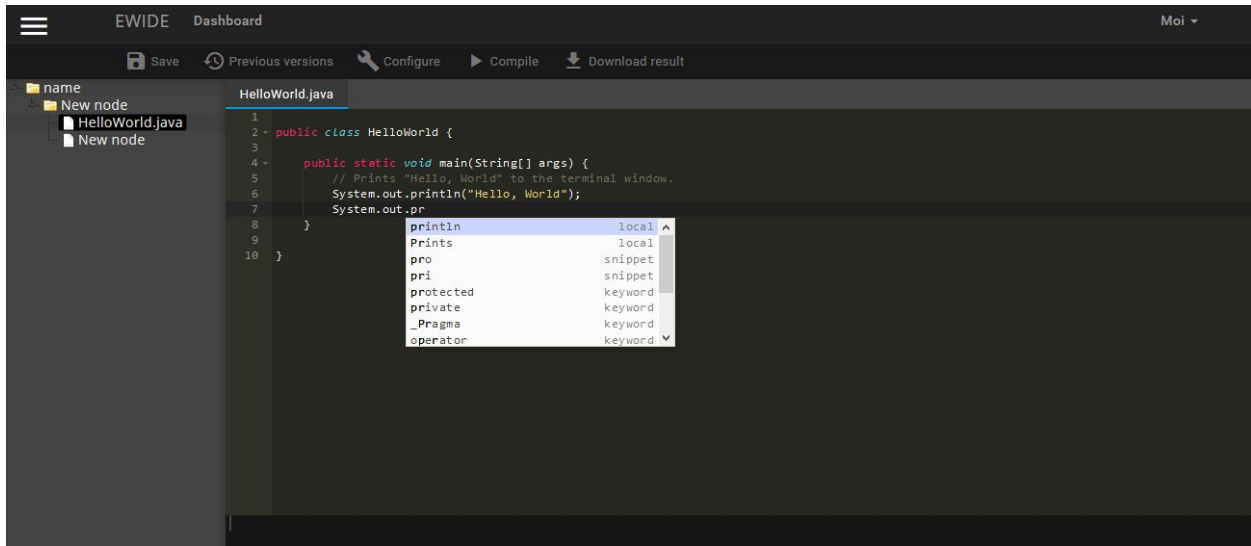


Figure 12 : Editeur de code

La page d'accueil d'un projet est son environnement de développement. Affiché dans des onglets séparés pour chaque fichier, l'éditeur lui-même fournit une prédiction de code sémantique selon le langage de programmation utilisé, en plus de la coloration syntaxique intégrée et l'affichage du nombre de lignes.

La page de l'éditeur comprend également la section de gestion des fichiers, avec la possibilité de modifier et de créer des fichiers et des dossiers.

Les principales fonctionnalités associées (compilateur, versionnage du fichier courant, téléchargement, ...) sont facilement accessibles depuis le menu principal.


Pour augmenter la productivité lors de l'écriture du code, nous avons ajouté des raccourcis clavier (Ctrl+S pour sauvegarder les fichiers par exemple).

8. Conserver et gérer les versions du code

Dès la création d'un projet sur le Dashboard, un nouveau dépôt git associé est automatiquement initialisé sur le serveur. Il s'agit de la "version 0" du projet, telle qu'identifiée sur EWIDE.


Puis, à chaque sauvegarde effectuée au sein de l'IDE, une nouvelle capture du projet et de ses fichiers est effectuée, permettant éventuellement une restauration de tout fichier en une version antérieure.

La page offrant cette fonctionnalité est accessible, dans l'éditeur de code, en ouvrant le fichier de son choix, puis en cliquant sur le bouton "Previous Versions" de la barre supérieure. Un résumé global des versions du projet est également accessible via le volet coulissant gauche.

 EWIDE Dashboard Me ▾

File versions

Number #	Date	User	
4	09-12-2016 00:10:10	zoidberg	+
3	09-12-2016 00:10:10	zoidberg	+
2	09-12-2016 00:09:49	zoidberg	+
1	09-12-2016 00:09:49	zoidberg	+
0	09-12-2016 00:09:23	zoidberg	+

Version #1 details: 

Comment:
New node has changed or has been changed

Content:
Bonjour !

Restore this version

Figure 13 : Outil de version

9. Compiler le code

L'éditeur propose d'effectuer une compilation du projet à l'aide de l'outil sélectionné et de ses arguments. Il supporte déjà Makefile et Maven en tant que moteur de production. L'utilisateur pourra aussi bien compiler directement avec des compilateurs tels que gcc, g++, javac et même Python. A la suite de la compilation, lancée sur la machine virtuelle, la sortie standard est affichée en bas de l'éditeur.

Il est possible pour les utilisateurs de télécharger le projet entier sous forme d'une archive au format zip. *Une amélioration possible aurait été le téléchargement de l'exécutable (si existant) ou de la version de déploiement de l'application écrite avec notre éditeur.*

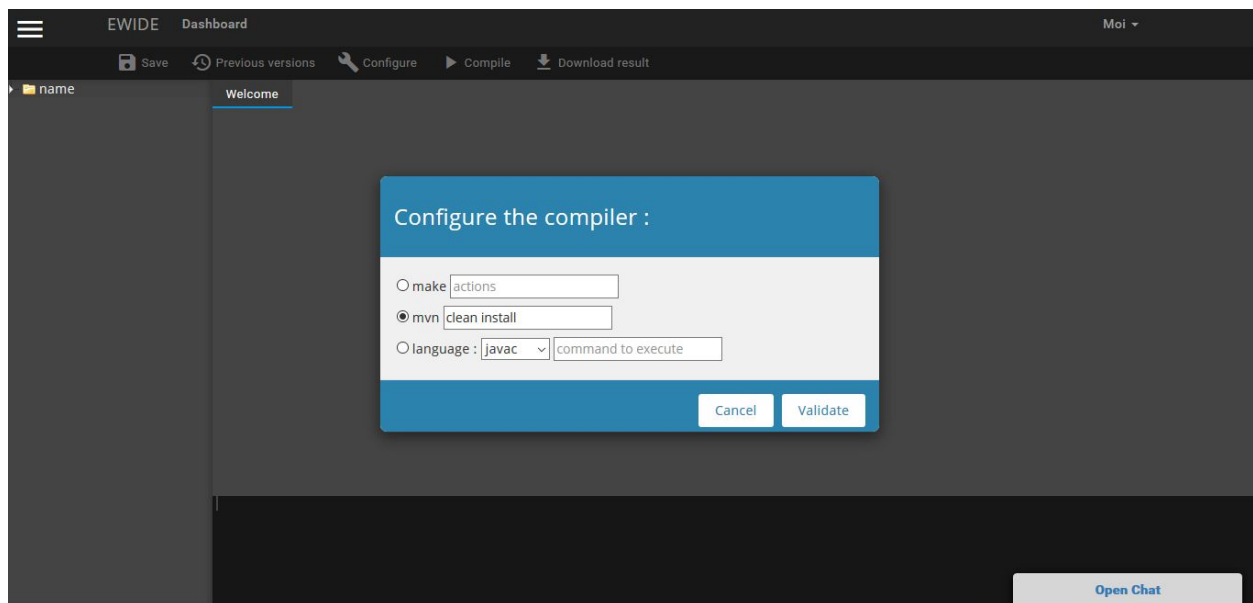


Figure 14 : Configuration du compilateur

10. Chatter

Afin de collaborer en direct, notre application EWIDE propose un chat permettant aux membres de communiquer entre eux. Le chat est disponible dans toutes les pages du projet.

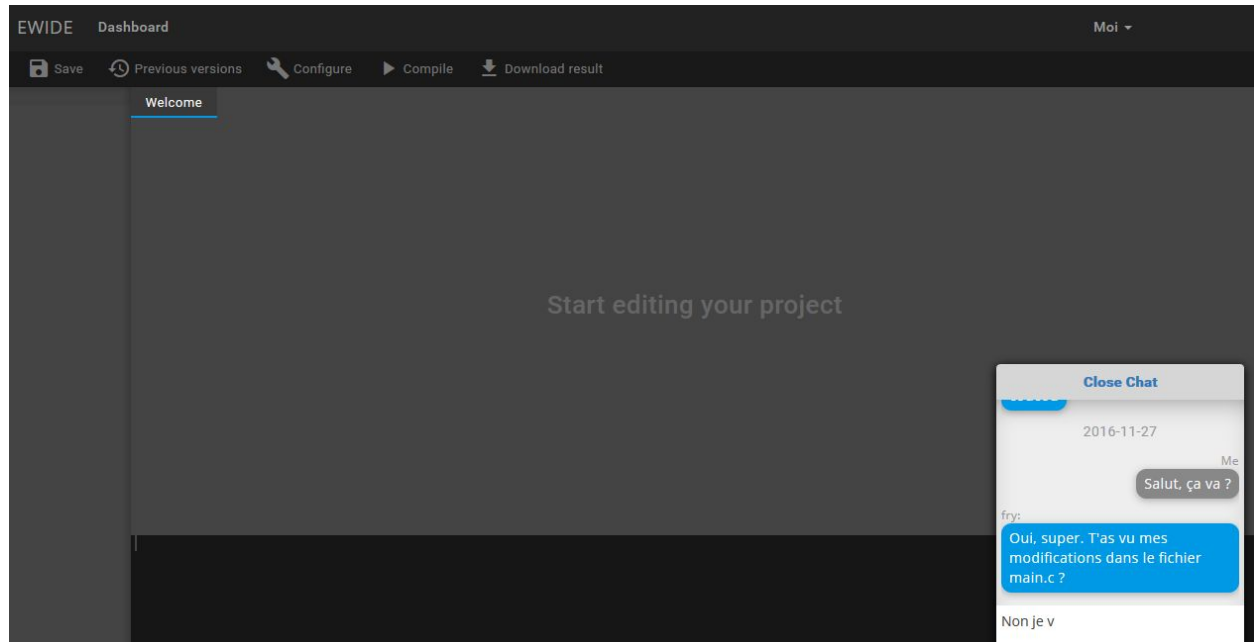


Figure 15 : Présentation du Chat

Une amélioration serait de recevoir une notification, mais cette option reste à négocier, car tout le monde le sait très bien, un développeur ne doit pas être dérangé lorsqu'il code ;) *

V. Améliorations

1. Tests

Malgré les tests unitaires sur la base de données, de nombreux tests doivent encore être faits pour assurer un suivi de qualité, notamment sur les contrôleurs et les services. Il faudrait pour cela utiliser le framework mockito, pour configurer des mocks, afin de simuler les appels à d'autres dépendances. Ces dépendances ainsi simulées permettraient de rendre unitaires les tests de ces classes. Effectivement, lors du développement d'EWIDE, les contrôleurs étaient la principale source d'introduction de bugs. Avec davantage de temps, la mise en place de tests fonctionnels aurait été particulièrement pertinente.

2. Sécurité

L'aspect sécurité de l'application doit aussi être retravaillé. Si la gestion des permissions permet de restreindre correctement l'accès à certaines ressources, une analyse complète de l'application doit être menée pour corriger d'éventuelles failles de sécurité. Sur ce point, nous avons identifié certaines vulnérabilités potentielles telles que l'Injection de commandes (CWE-78)¹ et le Stored/Reflected Cross Site Scripting (CWE-79)

3. Description des projets

Afficher la description des projets de manière esthétique dans la liste des projets.

4. Sous-tâche

La base de donnée prend déjà en compte la possibilité qu'une tâche dépende d'une autre tâche, mais ce n'est pas encore implémenté.

¹ Common Weakness Enumeration (<https://cwe.mitre.org/>)

5. Dossiers personnels

La base de données prend en compte un potentiel chemin vers un dossier personnel. Il est donc possible de faire en sorte qu'un utilisateur puisse garder un fichier sur son compte, indépendamment des projets.

6. Télécharger l'exécutable

Proposer l'exécutable nouvellement compilé au téléchargement selon le système d'exploitation de l'utilisateur.

VI. Conclusion

Nous désirons conclure en évoquant, plus avant, l'aspect humain de notre projet.

Au cours de sa formation, un étudiant n'est que trop peu amené à **travailler en groupe**, au-delà du simple binôme. Il n'est alors pas rare de se heurter au mur de l'assimilation en entreprise, une fois son diplôme obtenu.

Le projet transversal aura en ce sens été une initiation probante au métier d'ingénieur. La **pratique concrète de la gestion de projet** est effectivement plus que bienvenue. Composer avec les caractères, méthodes de travail de chacun, oser exposer son point de vue, convaincre autrui du bien-fondé de ses positions, avancer ensemble vers un objectif décidé unanimement; ce sont autant de challenges que nous avons relevés, et sommes fiers d'en présenter l'aboutissement.

EWIDE aura en outre nécessité l'**appréhension de nouvelles technologies**, lesquelles sortent évidemment du cadre strict du chaudron de la formation, et élargissent par là même notre champ de compétences.

La **gestion du temps**, enfin, aura été cruciale pour mener à bien notre labeur. Nombre sont les fonctionnalités qui auraient pu être implémentées, ne pas l'être, ou être non conformes aux demandes des mandataires, dépendamment de la pertinence des choix des collaborateurs au gré du développement. En cela, nous avons le sentiment d'avoir su tirer partie de ce que nous considérons à l'origine comme une contrainte: SCRUM. Les mêlées et réunions répétées auront servi à établir une stratégie commune, et à allouer notre ressource temps au mieux sur les points-clés de l'IDE.

Nous espérons qu'EWIDE vous aura donné entière satisfaction. :)*

* Le groupe ne saurait être tenu responsable de l'action délibérée du rédacteur d'insérer des smileys, qu'on le tienne pour su !