École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
5e année
2014 - 2015

Rapport de Projet de Fin d'Études

Pilote, outil collaboratif de gestion de projets

Encadrant
Nicolas RAGOT
nicolas.ragot@univ-tours.fr

Étudiant
Rémi PATRIZIO
remi.patrizio@etu.univ-tours.fr

Université François-Rabelais, Tours

DI5 2014 - 2015

Version du 8 mai 2015

Table des matières

Table des figures

Remerciements

Je tiens à remercier les personnes qui m'ont aidé tout au long de cette année dans la réalisation de ce PFE.

Tout d'abord, merci à mon encadrant M. Nicolas Ragot, Maître de conférence à l'Université François-Rabelais, qui a été disponible, impliqué et dont les remarques constructives ont aidé à faire de Pilote le projet abouti qu'il est aujourd'hui.

Je remercie également les membres du Service Informatique de Polytech Tours, MM. Sébastien Beaufils, Pascal Meichel et Mickaël Rousseau pour leur aide technique et la mise à disposition des serveurs.

Je n'oublie pas non plus l'équipe des Beta-testeurs qui a testé Pilote, cherché et reporté les bugs et suggéré des améliorations : Manon Lasne, Alexandre Loubier, Valentin Pertuy, et tous les autres.

Je remercie particulièrement mon collègue de promotion Christophe Forycki qui m'a aidé pour la connexion au serveur LDAP et m'a fait gagner plusieurs jours de travail.

Merci aussi à M. Sébastien Aupetit, Maître de conférence à l'Université François-Rabelais, pour sa connaissance des licences Open Source et ses conseils avisés en la matière.

Enfin, je remercie toute l'équipe ayant participé l'an dernier à Pilote lors du PIL, MM. Hamza Ayoub, Valentin Chareyre, Sofian Hamou-Mamar, Alain Krok, Wenlong Li, Yamine Zaidou, ainsi que M. Sylvain Darcissac, l'initiateur du projet.

Introduction

Ce projet de développement d'un outil de gestion de projets s'inscrit dans le cadre des Projets de Fin d'Études (PFE) que les élèves de cinquième année sont amenés à réaliser à l'École Polytechnique Universitaire de Tours.

L'objectif principal est de mettre en pratique les connaissances théoriques et techniques acquises au cours des années d'études précédentes, au sein d'un projet s'étalant sur toute l'année d'étude.

Le travail à effectuer se déroule de septembre à mai et des livrables sont à fournir régulièrement aux encadrants. Un Cahier de Spécifications Système a été livré au mois de janvier, et le présent document représente le rapport final, la dernière étape de ce projet.

Ce PFE n'est pas relié à l'un des laboratoires de recherche de Polytech Tours. Il s'agit de la suite d'un Projet d'Ingénierie du Logiciel (PIL) réalisé l'an dernier. Le but est d'améliorer et de finaliser l'application web qui a été partiellement développée, afin de la rendre Open Source et de passer en phase de production.

Nous allons tout d'abord présenter le site web existant et sa concurrence. Dans un second temps nous verrons l'architecture qui a été choisie pour celui-ci, puis nous détaillerons les fonctionnalités qui ont été améliorées ou ajoutées durant ce PFE. Ensuite nous parlerons de la méthodologie de gestion de projet qui a été appliquée, et enfin nous ferons un bilan général du développement de cette application web.

1.1 Contexte

L'an dernier, une application web de gestion collaborative de projets a été partiellement développée, dans le cadre d'un Projet collectif d'Ingénierie du Logiciel à Polytech Tours. L'équipe, dont je faisais partie, était composée de sept étudiants en quatrième année. Cette application devait reprendre le mode de fonctionnement du site de gestion de tâches *Trello*, et rajouter plusieurs fonctionnalités, afin d'être utilisée à Polytech Tours pour le suivi des projets entre les étudiants et les professeurs. L'idée avait été proposée par M. Sylvain Darcissac, à l'époque professeur au Département Mécanique de Polytech Tours.

Actuellement le site web *Trello* est souvent utilisé par les étudiants, mais il manque certaines fonctionnalités. Par exemple, il est impossible de regrouper des listes de tâches par étape de développement, ou d'afficher les tâches dans un calendrier. De plus, comme toute solution propriétaire, rien n'assure que le site restera gratuit et en ligne à l'avenir. Il s'agissait donc de développer un outil équivalent afin de ne plus dépendre de celui-ci, en apportant des fonctions supplémentaires.

Le développement de ce site web, que nous avions baptisé Pilote, était assez avancé l'an dernier, mais il était loin d'être complet, opérationnel et utilisable. L'état d'avancement de Pilote à la fin de l'année dernière est précisé plus en détail dans la section 1.3.

Le but de ce Projet de Fin d'Études était de **reprendre ce développement**, de **rajouter un certain nombre de fonctionnalités** (définies par mon encadrant, M. Ragot) découlant pour la plupart des développements avortés de l'an dernier, et d'**en faire un projet Open Source**. Cela implique donc de fournir une documentation technique et utilisateur détaillée, afin que d'autres développeurs puissent reprendre le projet à l'avenir.

Les objectifs de ce PFE sont détaillés plus bas, à la page 10.

1.2 Étude de la concurrence

C'est un manque de fonctionnalités de Trello qui est à l'origine de Pilote. Faisons un tour d'horizon des solutions actuellement sur le marché.

Trello

Trello n'est pas Open Source, c'est un outil gratuit développé par une entreprise aux États-Unis. Il propose une interface très simple d'utilisation et très pratique, sous forme de cartes et de listes de cartes réagençables avec un glisser-déposer. Pour chaque tâche, on peut effectuer un grand nombre d'opérations, comme par exemple joindre un fichier, assigner des personnes, définir une *deadline*, etc. Un certain nombre de fonctionnalités sont payantes.

Il est très couramment utilisé par les étudiants des Départements Informatique et Mécanique de Polytech Tours. Comme tout site propriétaire, si l'entreprise venait à fermer ce service, plus personne ne pourrait l'utiliser.

Il a la particularité d'être assez généraliste et peut être utilisé pour un tout autre usage que les projets de développement.

Redmine

Redmine est un des outils open source de gestion de projets collaboratifs les plus aboutis mais aussi un des plus austères. Il permet de générer des diagrammes de Gantt, de créer un wiki, de gérer finement les droits des utilisateurs et il peut s'intégrer à Jenkins ou à Subversion.

Il est installé à Polytech mais la complexité de son interface et son manque de dynamisme rendent son utilisation peu populaire auprès des étudiants.

Kanboard

Kanboard est un site open source assez récent. Il se veut le pendant libre de Trello, néanmoins son interface est très simpliste, moins agréable que celle de Trello, et il lui manque des fonctionnalités comme l'upload de fichiers.

Néanmoins il propose des éléments intéressants comme le filtrage des tâches ou le principe de sous-tâches.

LibreBoard

Véritable clone open source de Trello, Libreboard est allé jusqu'à copier la typographie de son logo. On retrouve la même simplicité d'utilisation, mais on peut cette fois héberger le site sur son propre serveur. On retrouve également les principales fonctions basiques de l'original, mais la plupart des fonctions avancées (upload de fichier, ajout de *deadlines*, commentaires, etc) sont absentes.

De plus, à cause de sa forte ressemblance avec Trello, l'équipe en charge de LibreBoard a reçu en janvier 2014 une mise en demeure pour violation de droits d'auteur.

1.3 Présentation de l'existant

Au début de ce PFE, Pilote était une application web permettant de s'enregistrer et de se connecter. On pouvait créer des projets et, dans ces derniers, ajouter des listes de tâches, regroupés par domaines de métier et par étape du projet. On pouvait déplacer les tâches par glisser-déposer dans une liste ou entre deux listes, ou encore réordonner les listes.

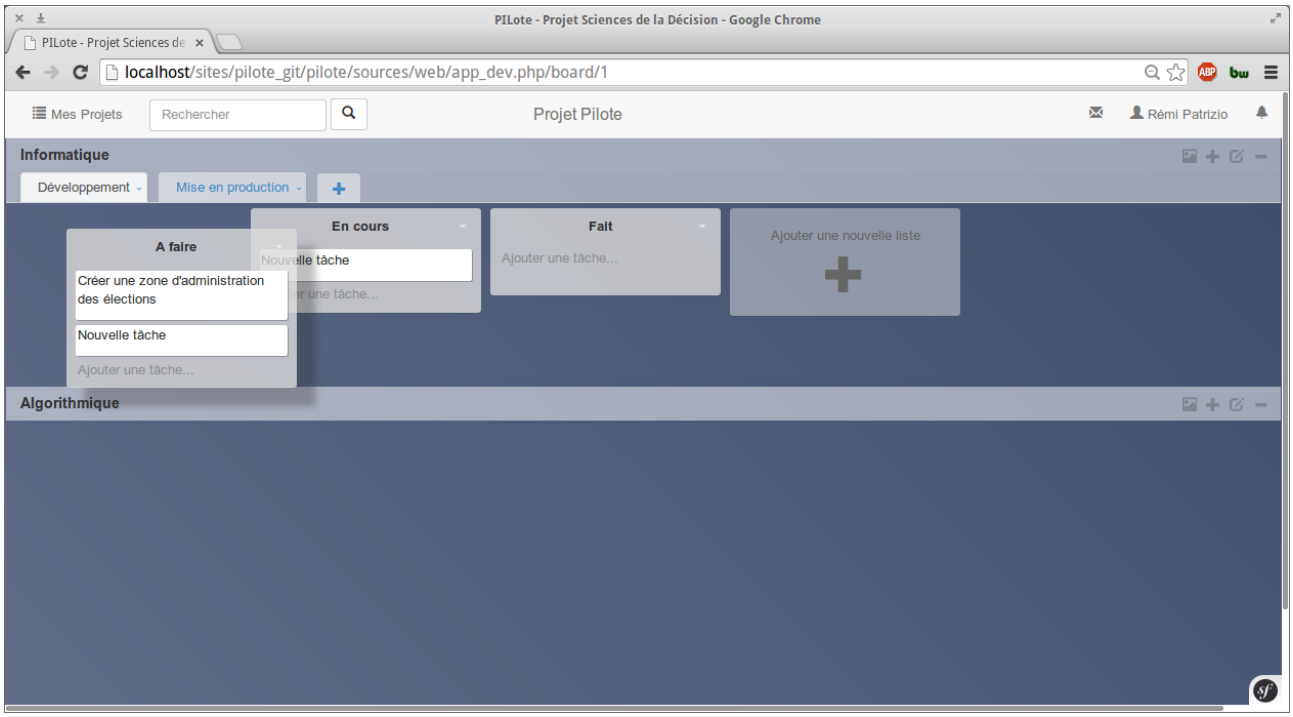


FIGURE 1.1 – Vue du Board avant le PFE

En cliquant sur une tâche, une popup s'ouvrait, affichant la description détaillée d'une tâche, ainsi que plusieurs boutons (voir la capture d'écran ci-dessous) pour l'instant inactifs.

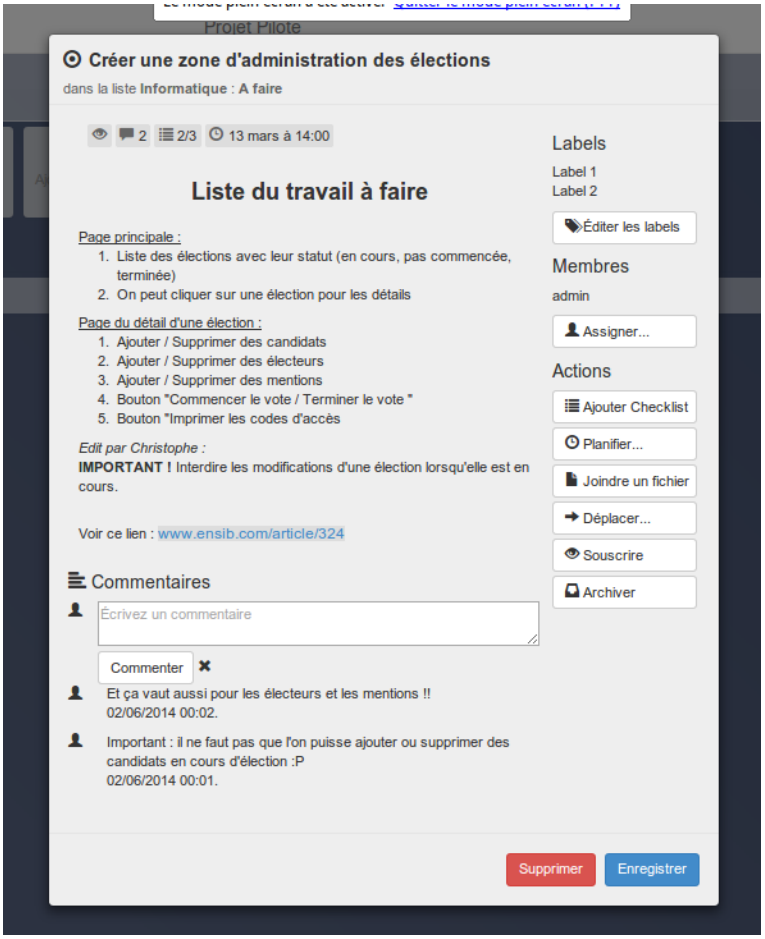


FIGURE 1.2 – Popup de détails d'une tâche avant le PFE

Tous les utilisateurs peuvent voir le détail de ces tâches, en modifier le titre ou le contenu et ajouter des commentaires. Ils ont aussi accès à un diagramme de Gantt partiellement fonctionnel, à un calendrier pour le moment vide et à une messagerie.

L'administrateur a accès à une zone d'administration, vide elle aussi.

1.4 Technologies utilisées

J'ai choisi de reprendre les technologies qui avaient été utilisées l'an dernier en PIL, d'une part car je les avais déjà utilisées, et d'autre part car certains développements auraient demandé plus de temps si j'étais reparti de zéro avec une autre technologie.

Néanmoins, je n'avais pas conscience au début de l'année de la quantité de Javascript qui serait nécessaire. Si je l'avais su, j'aurais pu utiliser un framework Javascript comme Angular.js ou React.js, mais il m'aurait fallu passer un certain temps à apprendre à les utiliser.

Symfony 2

Dans la continuité du projet de l'an dernier, le développement a été effectué dans le langage PHP, avec le framework Symfony. Respecter l'architecture bien définie par celui-ci permet de garder un code clair et compréhensible. De plus, il intègre un certain nombre de composants permettant de développer rapidement une application web complexe et sécurisée.

Symfony intègre un ORM (Object Relational Mapping) appelé Doctrine pour faire le lien avec la base de données. Ainsi, les tables SQL sont directement générées à partir des classes PHP de notre modèle de données.

Le projet est ainsi suffisamment léger pour tourner sur de petites configurations, peut importe le système d'exploitation (OS X, Linux, Windows, bien que l'on ait noté quelques lenteurs sur des PC sous Windows).

jQuery

Toujours dans la continuité de l'an dernier, jQuery est une librairie Javascript créée pour faciliter l'écriture de scripts côté client. Il est utilisé ici entre autres pour les animations dans les pages d'un Board.

Composer

Le gestionnaire de dépendances Composer installe automatiquement les différents composants externes indispensables au bon fonctionnement de notre application, comme par exemple la bibliothèque Symfony2 ou les bundles que nous utilisons (FOSUserBundle, FR3DLdapBundle, etc).

Twitter Bootstrap

Twitter Bootstrap est un framework CSS couramment utilisé pour structurer simplement ses pages et rendre son site web *adaptatif* (ou *responsive* en anglais, qui s'adapte aux différentes tailles d'écrans).

dhtmlxGantt

DHTMLX est une entreprise qui produit des composants ou des librairies complexes pour le web en Javascript sous double licence : une licence libre *GNU GPL* et une licence commerciale.

dhtmlxGantt est, comme son nom l'indique, un composant permettant d'intégrer un diagramme de Gantt personnalisable à une application web.

1.5 Objectifs

Le but principal de ce PFE était d'aboutir à un outil fonctionnel, utilisable et utile. Pilote devrait reprendre la facilité d'utilisation et les fonctionnalités de Trello, en ajoutant quelques éléments manquants, et en le rendant Open Source, à l'instar de Redmine.

Le premier objectif était tout d'abord de finir de développer les fonctionnalités prévues lors du PIL qui n'étaient pas complètement opérationnelles ou qui n'avaient pas été développées par manque de temps. Ensuite, quelques tâches supplémentaires définies conjointement par M. Ragot et moi-même ont été ajoutées à cette liste et sont détaillées plus bas. L'ensemble de ces objectifs figurent dans le Cahier de Spécifications Système.

Le tableau 1.1 ci-dessous répertorie les fonctions prévues lors du PIL ainsi que leur état d'avancement.

La figure 1.3 suivante représente les cas d'utilisation prévus pour ce PFE. Leur état d'avancement initial y est aussi représenté.

Fonctionnalités	État d'avancement à la fin du PIL	Repris pour le PFE
Afficher des listes de tâches sur un tableau type "Kanban", pouvoir déplacer et éditer ces tâches et ces listes	Fonctionnel	/
Regrouper les listes de tâches en Étapes et regrouper les Étapes en Domaines	Fonctionnel	/
Gérer précisément les droits d'accès, limiter les accès aux projets à ses seuls membres	Partiellement développé et partiellement fonctionnel	OUI
Ajouter un fichier en pièce jointe d'une tâche	Non développé, non fonctionnel	OUI
Assigner une personne à une tâche	Non développé, non fonctionnel	OUI
Ajouter des listes de cases à cocher à une tâche	Partiellement développé mais non fonctionnel	OUI
Ajouter un pourcentage de réalisation à une tâche	Non développé, non fonctionnel	OUI
Ajouter des dates de début et de fin à une tâche	Très partiellement développé mais non fonctionnel	OUI
Commenter une tâche	Partiellement développé, partiellement fonctionnel	OUI
Attribuer une priorité à une tâche	Non développé, non fonctionnel	OUI
Gérer les coûts et les risques liés à une tâche	Non développé, non fonctionnel	NON
Zone d'administration pour gérer les utilisateurs et les projets	Très partiellement développé mais non fonctionnel	OUI
Système de messagerie pour discuter entre utilisateurs	Très partiellement développé, non fonctionnel	OUI
Générer un diagramme de Gantt basé sur les dates associées aux tâches d'un projet	Partiellement développé et partiellement fonctionnel	OUI
Afficher les tâches d'un projet dans un calendrier	Partiellement développé mais non fonctionnel	OUI
Système de notifications pour être alerté des modifications et des nouveautés	Non développé, non fonctionnel	OUI

TABLE 1.1 – Liste des fonctionnalités prévues lors du PIL

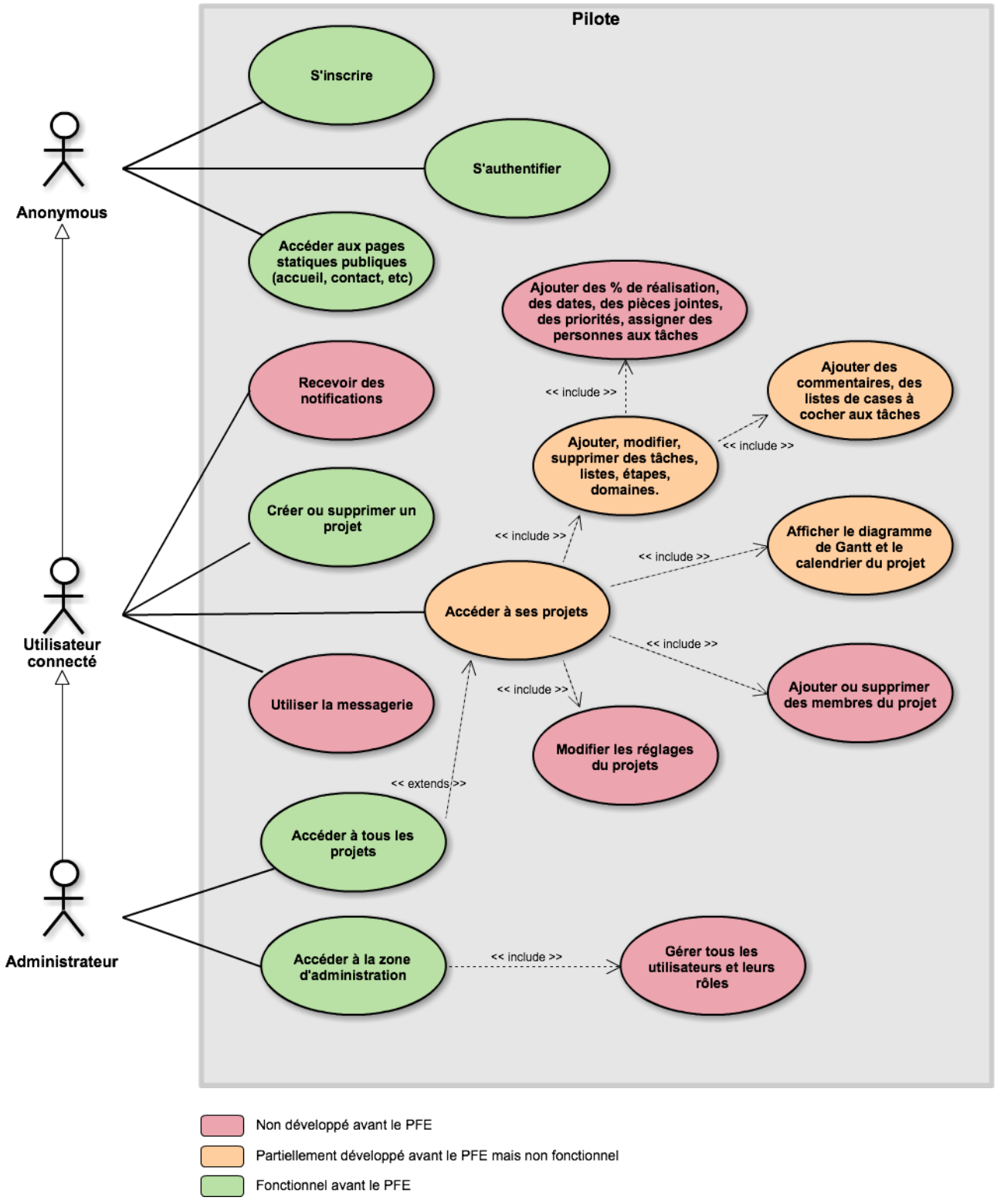


FIGURE 1.3 – Diagramme des cas d'utilisation, avec l'état d'avancement des cas au début du PFE

Voici donc la liste complète des tâches à réaliser lors de ce PFE :

Revue de code complète
Prendre en main le code des autres étudiants, supprimer les doublons, corriger les erreurs importantes, restructurer le projet.

Gestion complète des droits d'accès
Limiter les accès aux projets aux seuls membres de ce projet. Permettre à un membre d'un projet d'ajouter ou de supprimer un utilisateur d'un projet.

Assigner une personne à une tâche
Permettre à l'utilisateur d'assigner un membre d'un projet à une tâche.

Ajouter un pourcentage de réalisation à une tâche

Ajouter une priorité à une tâche

Joindre un fichier à une tâche

Commentaires sur les tâches

Listes de cases à cocher sur les tâches

Dates de début et de fin des tâches

Générer un diagramme de Gantt pour un projet

Afficher le calendrier du projet

Système de notifications
Permettre à l'utilisateur de recevoir en temps réel des notifications lorsque des éléments le concernant ont subi des modifications ou lorsqu'il a reçu un message.

Zone d'administration
Permettre aux administrateurs de gérer l'ensemble des utilisateurs et des projets.

Messagerie
Échanger des messages avec les autres utilisateurs, à deux ou à plusieurs.

Connexion avec les identifiants de Polytech
Permettre aux étudiants et aux professeurs de Polytech Tours de se connecter avec leurs identifiants, sans qu'ils aient besoin de s'inscrire au préalable.

Système d'extension
Utiliser le mécanisme d'héritage des *bundles* Symfony pour proposer aux développeurs de créer très simplement un bundle de gestion des tâches héritant de celui par défaut de Pilote et ajoutant des fonctionnalités plus spécifiques. Le but est de ne pas surcharger l'interface du site de fonctions peu usuelles et de la garder épurée, au contraire de Redmine.

Mise en Open Source
Rédiger une documentation utilisateur et technique, mettre à disposition le code sur une plateforme comme Github.

Architecture du système

Pilote est basé sur le framework PHP Symfony2, qui impose une structure bien définie. Il est indispensable d'avoir de bonnes connaissances du framework pour reprendre ce projet.

2.1 Structure d'un projet Symfony2

Un projet Symfony utilise souvent des dépendances externes, comme des *bundles* ou des librairies. Ceux-ci ne sont pas inclus directement dans les dépôts. Le fichier `composer.json` contient la liste de toutes ces dépendances, qu'il est possible d'installer ou de mettre à jour en une ligne de commande.

Symfony utilise une **architecture MVC** (Modèle-Vue-Contrôleur) adaptée pour le web : Les **vues** sont des pages HTML sans aucun traitement en PHP. Les données provenant du contrôleur (ex : une liste d'utilisateurs) sont incluses grâce à des balises spéciales. Ce langage s'appelle Twig, et ces fichiers de vues ont pour extension `.html.twig`. Les zones "dynamiques" sont codées en Javascript avec jQuery.

Le **modèle** est constitué d'un ensemble de classes PHP : Task, Step, Domain, Board, User etc. Le lien entre ces objets et les tables de la base de données est automatiquement effectué par Doctrine, un composant intégré à Symfony (c'est donc un ORM).

Des **fichiers de routage** font le lien entre une URL et le contrôleur qu'il faut appeler.

Ces **contrôleurs** appellent le modèle ou la session en cours pour récupérer des informations, effectuent un traitement puis appellent la vue concernée avec les données adéquates.

L'arborescence des dossiers et fichiers est la suivante :

```
sources
 - app
    - config            # Contient les fichiers de configuration
       - config.yml     # Paramètres de config. des dépendances de Pilote
       - parameters.yml # Paramètres spécifiques à une installation (non versionné)
       - route.yml      # Fichier de route principal
    - Resources
       - views          # Contient les vues (Twig) non spécifiques à un bundle

 - src                  # Contient tous nos bundles
    - Pilote
       - MainBundle     # Bundle principal pour les éléments communs à toutes les
                        # pages
          - Controller  # Contient les contrôleurs PHP du bundle
          - Entity      # Contient les classes PHP définissant le modèle
          - Form        # Contient les classes PHP générant des formulaires
                        # automatiquement
          - Resources
             - config   # Contient les routes des pages statiques
             - views    # Contient les vues des pages statiques et de la barre
                        # de menu
```

```
      - UserBundle      # Bundle gérant les sessions des utilisateurs
         - (idem)
      - ...

- vendor                # Toutes les dépendances sont stockées ici

- web
   - css                # Contient toutes les feuilles de style CSS
   - images             # Contient toutes les images
   - js                 # Contient tous les scripts JS

- composer.json         # Liste des dépendances externes
```

2.2 Évolution de l'architecture générale des bundles

Comme dans tout site web basé sur Symfony, Pilote est divisé en grands modules appelés *bundles*. Au début de ce PFE, nous avions quatre *bundles* :

- PILTaskerBundle : un "fourre-tout" dans lequel étaient les fichiers concernant la gestion des tâches et des projets, la page d'accueil, les pages statiques, etc.
- PILUserBundle : il héritait du bundle FOSUserBundle et contenait les fichiers concernant la gestion des utilisateurs, les pages de connexion, etc.
- PILMessageBundle : il héritait du bundle de messagerie FOSUserBundle et devait contenir les fichiers concernant la messagerie, mais il était pratiquement vide.
- PILAdminBundle : il devait contenir les fichiers concernant la zone d'administration mais il était lui aussi pratiquement vide.

On peut représenter cela par le schéma suivant :

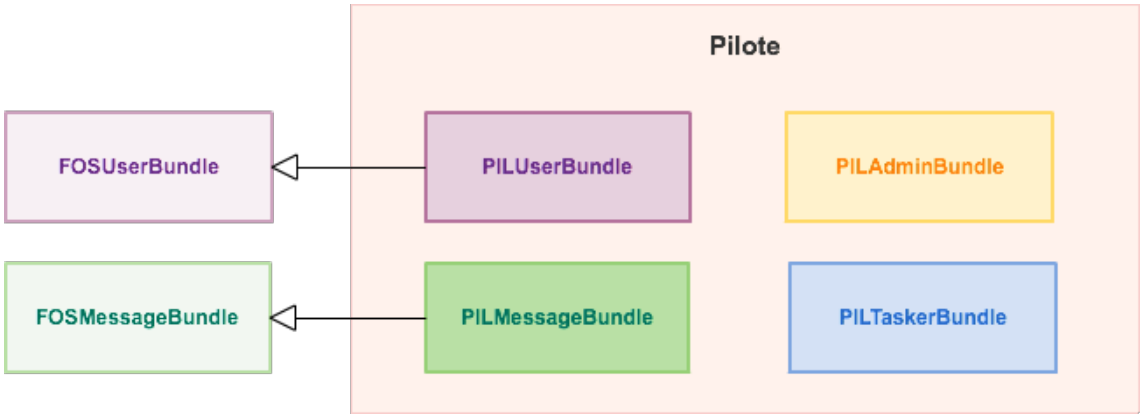


FIGURE 2.1 – Architecture des bundles avant le PFE

Pour le PFE, il était prévu dans le Cahier de Spécifications Système de créer un *bundle* spécifique pour les pages statiques et tous les éléments du site communs aux différentes zones, comme la barre de menu ou les notifications. Il s'appellerait PiloteMainBundle. De plus, les développeurs devaient avoir la possibilité de créer leurs propres *bundles* héritant du *bundle* de gestion des tâches de Pilote.

Le schéma suivant représente l'architecture prévue :

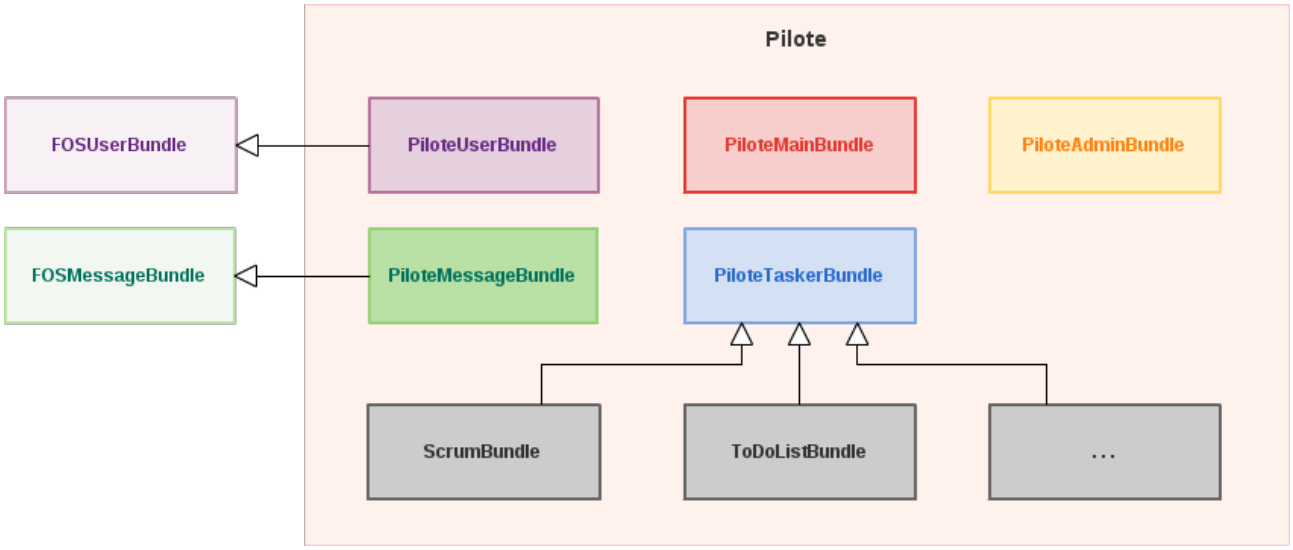


FIGURE 2.2 – Architecture des bundles prévue pour le PFE

Finalement FOSMessageBundle n'a pas été utilisé. De plus, le système d'extensions a finalement été abandonné. On arrive donc au schéma suivant :

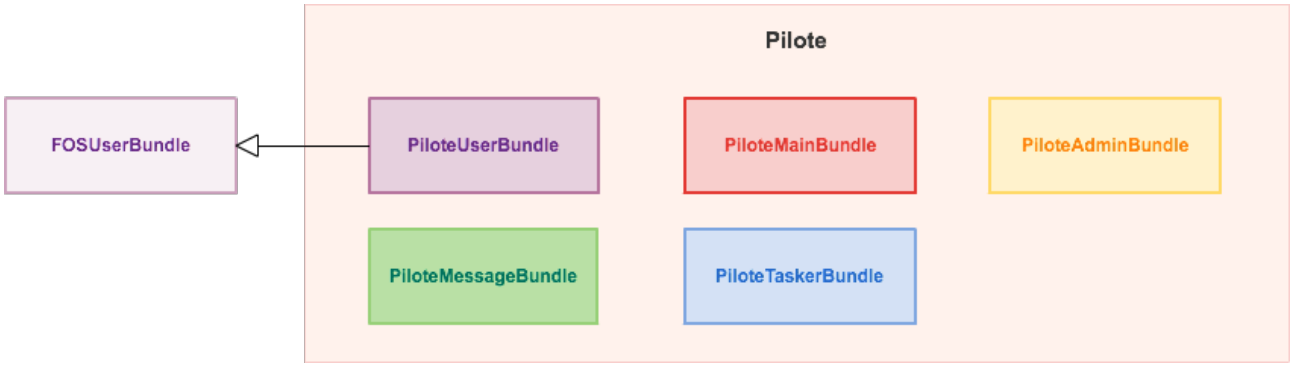


FIGURE 2.3 – Architecture finale des bundles

2.3 Fonctionnement des serveurs

Le serveur principal est un serveur Apache qui fait tourner le site web en Symfony. Dans la documentation technique, il est expliqué comment configurer Apache pour passer d'une URL comme `www.monsite.com/web/app_dev.php/accueil` telle qu'on en a pour tous les sites en Symfony à une URL plus élégante comme `www.monsite.com/accueil`.

En parallèle, un serveur basé sur Node.js fait tourner le système de notifications. Le fichier du serveur est situé à `web/js/notifications/app.js` et celui des clients est `web/js/notifications/notifs.js`. Des explications plus complètes sont disponibles à la section 3.6 et les détails techniques sont expliqués dans la documentation technique sur GitHub.

Le traitement d'une requête HTTP classique peut être représenté par le diagramme de séquence suivant : La requête d'un utilisateur est d'abord analysée par le routeur, qui regarde dans tous les fichiers de routage si l'URL demandée existe et à quelle action de quel contrôleur elle correspond. Ce dernier effectue le traitement associé à cette action. Pour cela, il appelle l'EntityManager pour accéder aux données stockées dans la base de données, génère la vue (c'est à dire la page web, ou le fragment de page) correspondante, et renvoie cette vue au client.

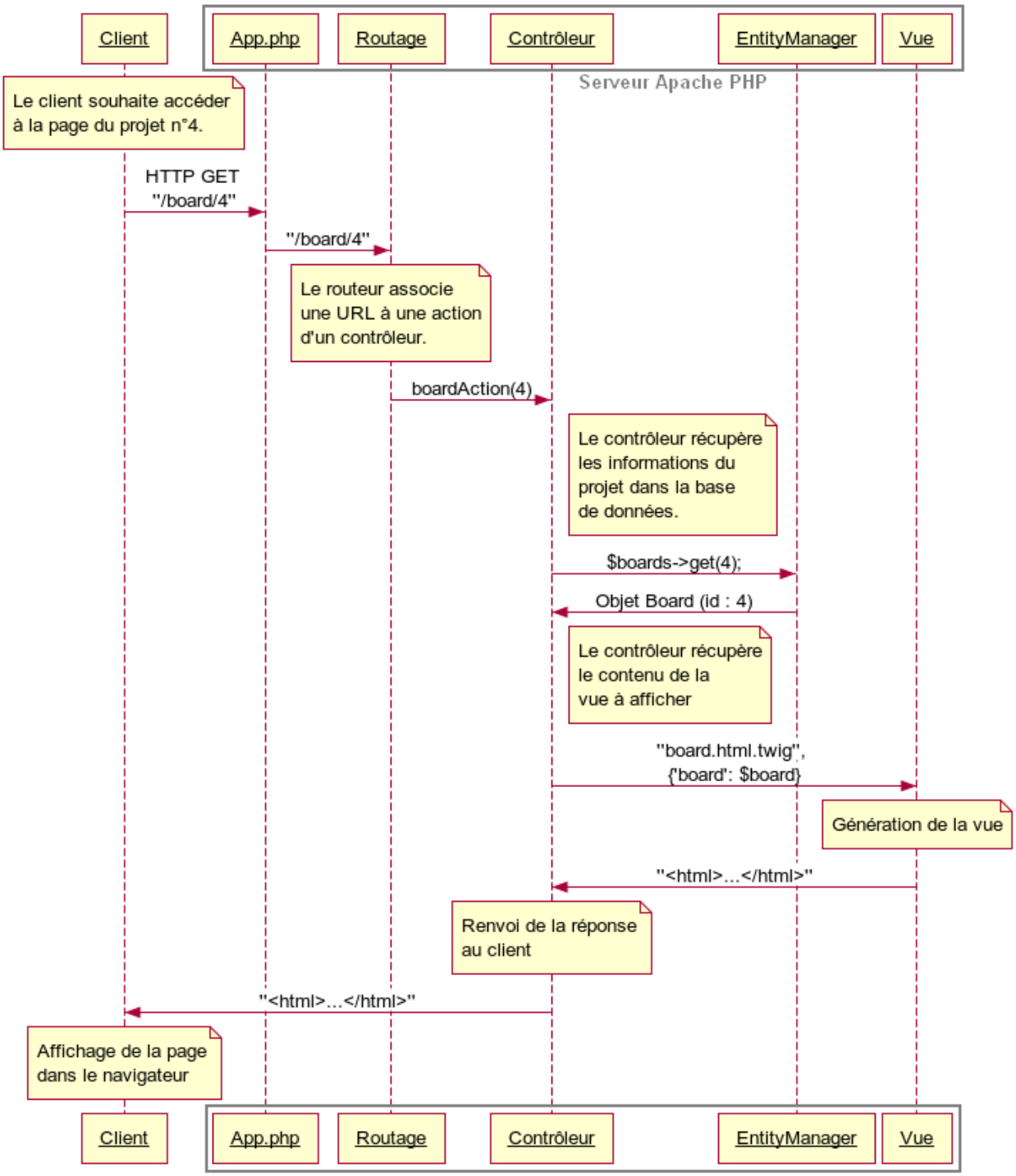


FIGURE 2.4 – Diagramme de séquence du traitement d'une requête HTTP dans Pilote

Si la requête entraîne l'envoi de notifications, le diagramme de séquence est plus complexe :

C'est dans le contrôleur que l'envoi est effectué. Celui-ci envoie une notification au serveur Node.js contenant plusieurs informations, dont la liste des utilisateurs concernés par la notification. Le serveur Node.js compare cette liste avec celle des clients connectés (c'est à dire l'ensemble des pages web de Pilote ouvertes chez les utilisateurs), et renvoie une notification via un WebSocket à chacun d'eux.

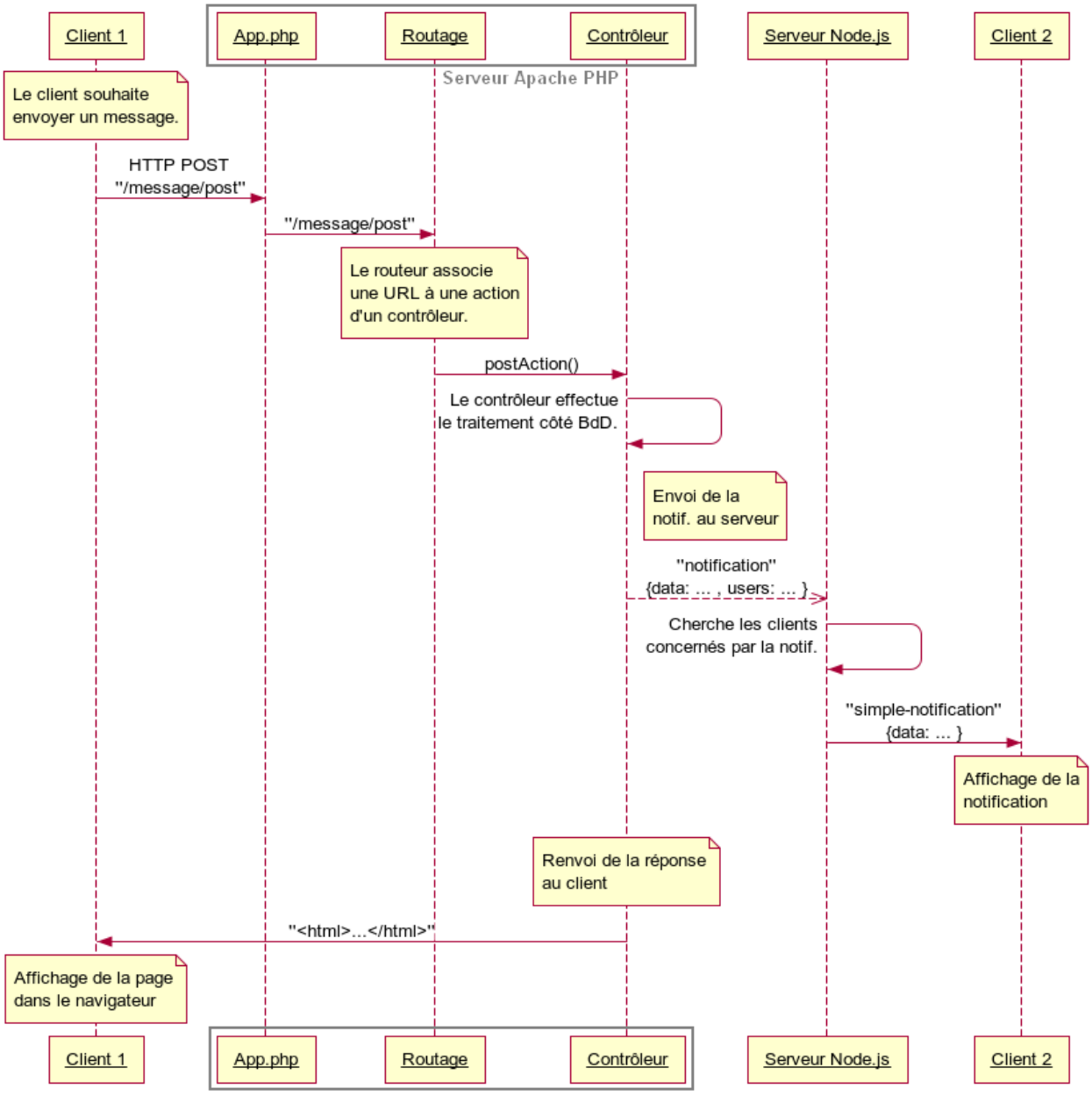


FIGURE 2.5 – Diagramme de séquence du traitement d'une requête HTTP avec notification

2.4 Modèle de données

Le diagramme de classes représenté ci-dessous résume le modèle (au sens MVC) de l'application. Les classes sont regroupées selon le bundle auxquelles elles appartiennent. Le schéma 2.6 représente le modèle tel qu'il était au début du PFE, et le schéma 2.7 suivant est le modèle actuel.

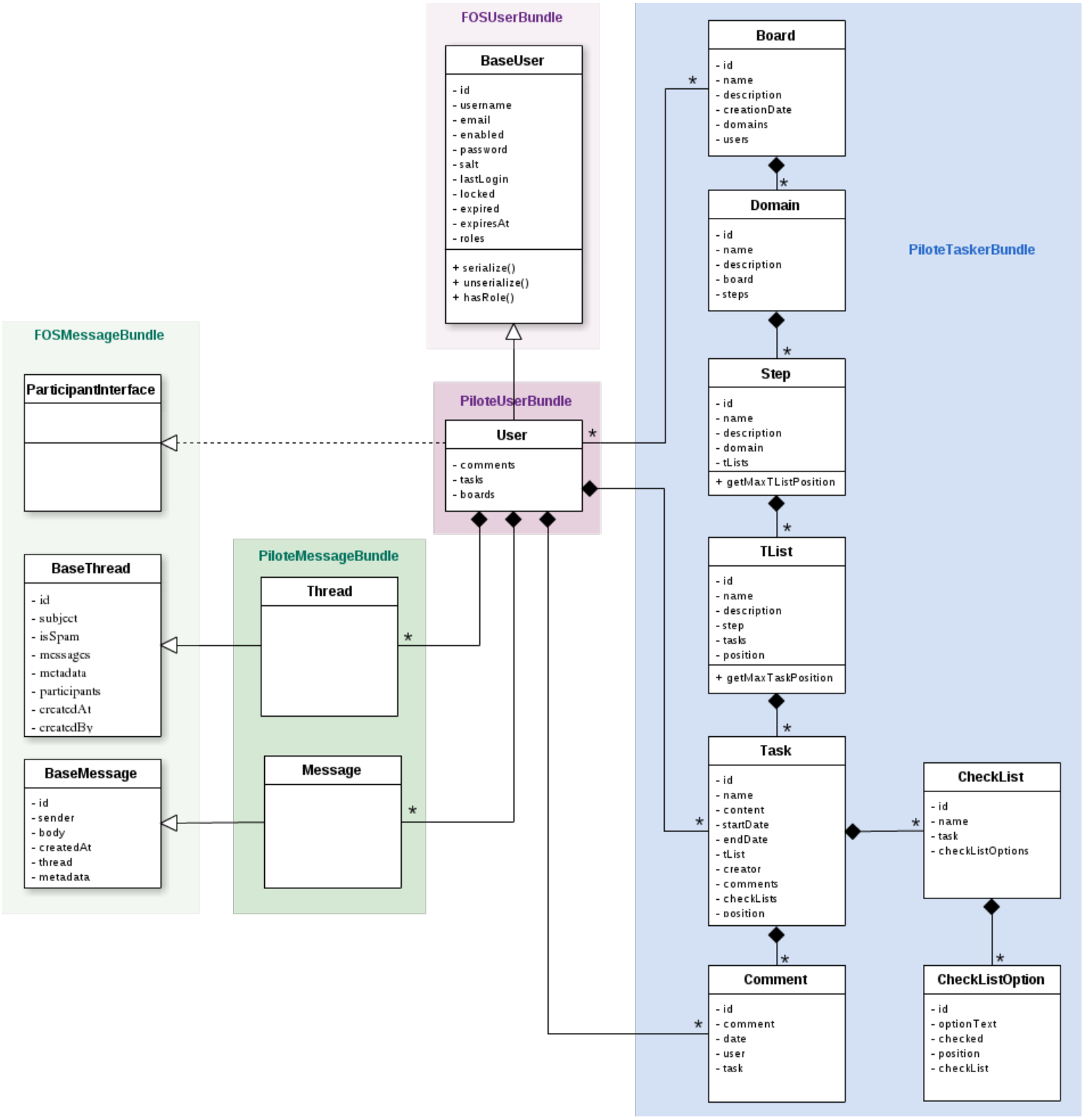


FIGURE 2.6 – Diagramme de classe au début du PFE

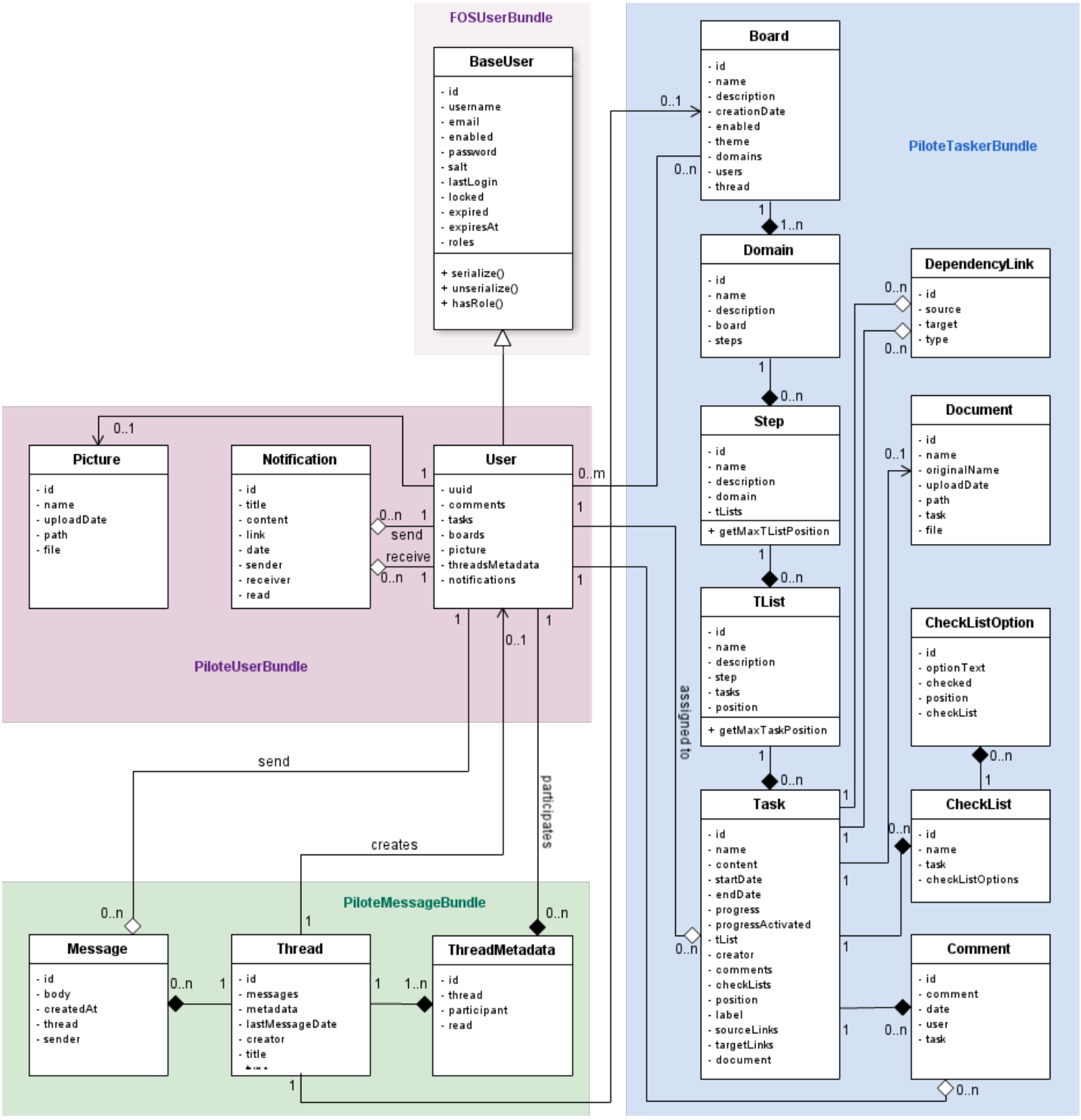


FIGURE 2.7 – Diagramme de classe actuel

Travail réalisé

3.1 Fonctions du Board

Réalisation des objectifs

Le "Board" est un terme utilisé à l'origine par Trello pour nommer la vue principale des listes de tâches, sous forme de panneau suivant la méthode *Kanban*. Nous l'avons repris pour nommer la même vue.

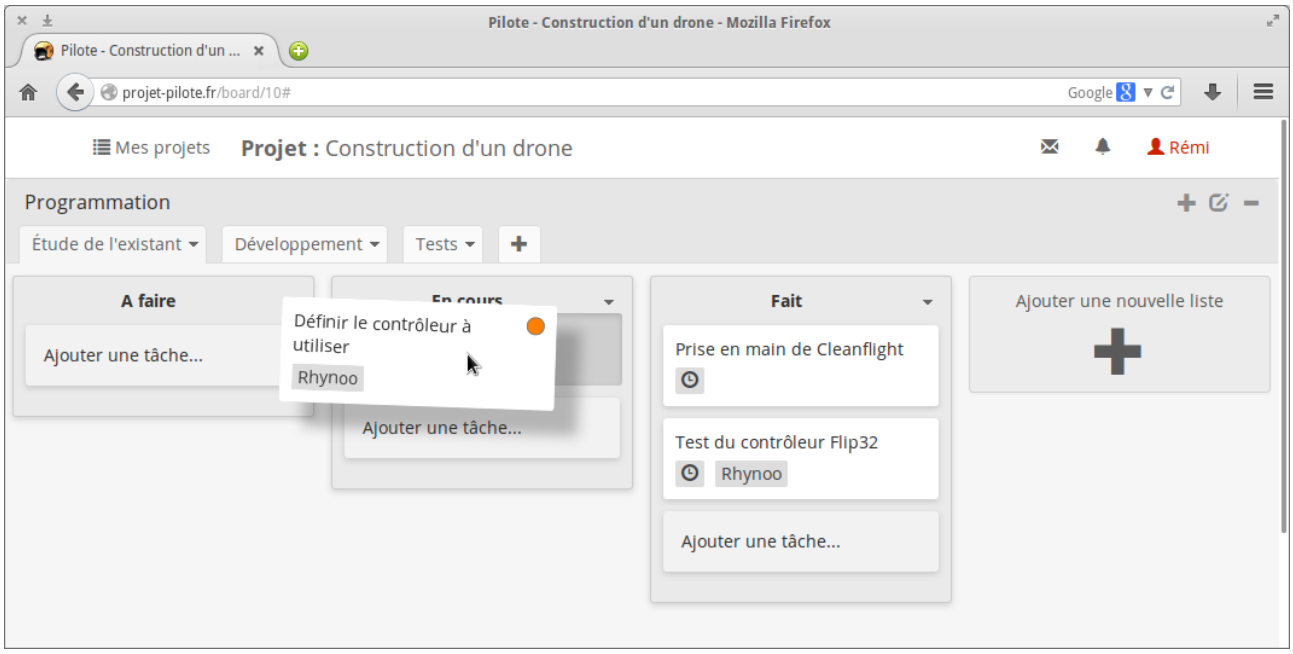


FIGURE 3.1 – Board d'un projet

Le Board donc, comportait un grand nombre de fonctionnalités non développées ou non finalisées : Dans la popup s'affichant au clic sur une tâche, on peut désormais ajouter un pourcentage de réalisation, une priorité, un fichier joint, des commentaires, des cases à cocher et des dates de début et de fin ou encore une personne assignée.

Pour chacune de ces fonctionnalités, le fonctionnement est globalement similaire :

— Dans les fichiers gérant la vue, on crée des boutons ou des listes déroulantes et on appelle le fichier Javascript `/web/js/board/taskDetails.js` ainsi que la librairie `jQuery.js`,
— Dans ce script, on écoute l'événement correspondant au clic sur le bouton (par exemple) et on appelle une fonction qui va exécuter une **requête AJAX**.
— Cette requête est reçue par le serveur et envoyée à l'un des `AjaxController`. Celui-ci effectue le traitement dans la base de données (par exemple, ajouter un commentaire), génère la vue correspondante (par exemple, le code HTML de notre nouveau commentaire), et la renvoie en réponse.
— Au retour de la requête, le script JS n'a plus qu'à insérer le code HTML au bon emplacement.

Par ailleurs, afin de limiter les accès des projets aux seuls membres de ce projet, une page de réglages d'un projet a été créée. Elle permet entre autres d'ajouter ou de supprimer des utilisateurs à ce projet. De plus, le contrôleur chargé d'afficher le board doit avant tout vérifier si l'utilisateur courant a accès ou non au projet avant d'afficher quoi que ce soit.

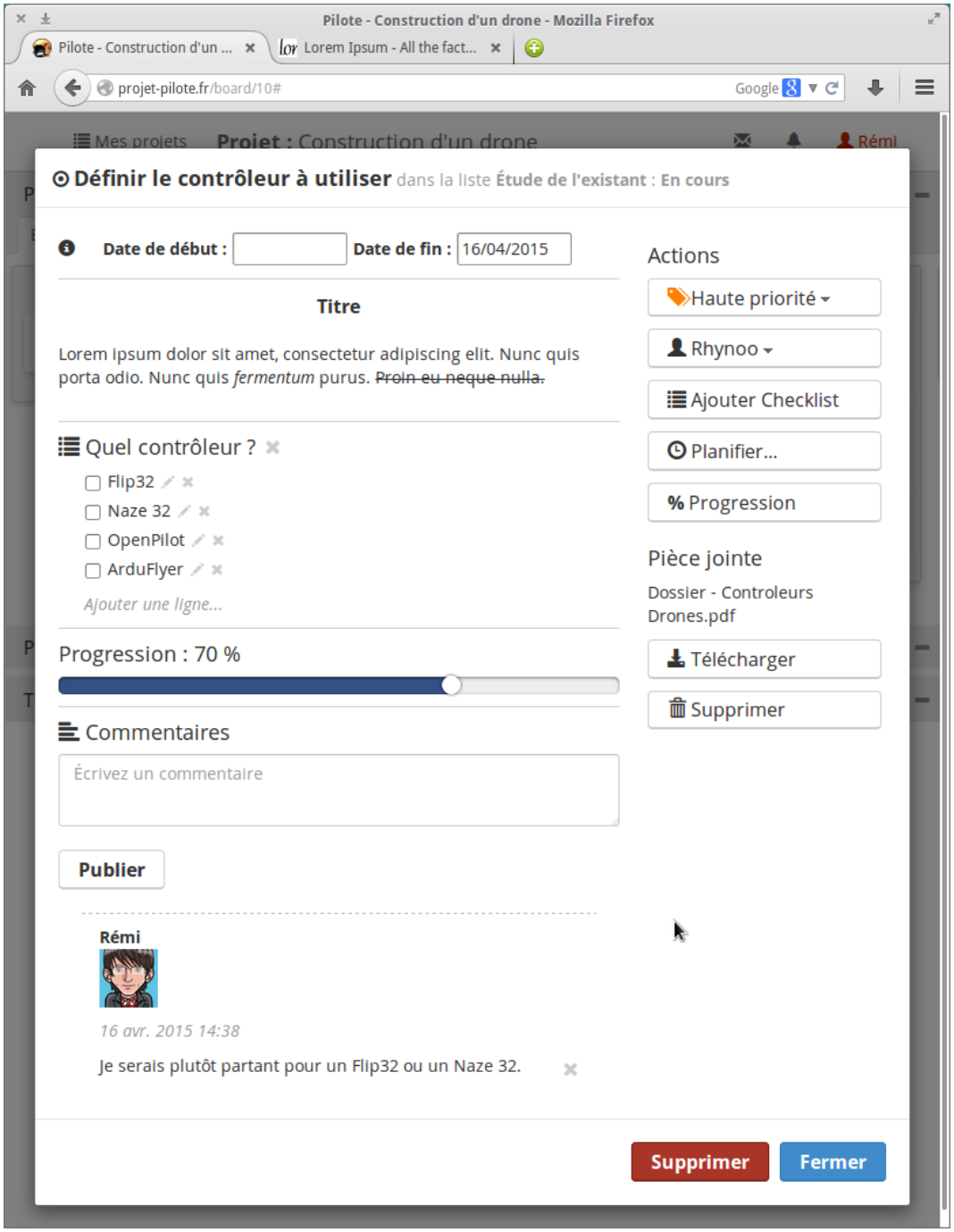


FIGURE 3.2 – Popup s'affichant au clic sur une tâche dans le Board, le diagramme de Gantt ou le calendrier

Problèmes rencontrés

Lorsque l'on manipule plusieurs fois des éléments du Board, il faut veiller à ce que nos fonctions en Javascript qui sont appelées lors d'événements (ex : clic sur un bouton) s'exécutent bien, mais ne s'exécutent qu'une seule fois. Cela a été un bug récurrent pendant le développement.

De plus, le debug de requêtes AJAX n'est pas aussi aisé que celui des requêtes HTTP classiques. Cela nécessite une bonne connaissance des outils de développement du navigateur utilisé.

Un autre problème se présente dès que l'on souhaite utiliser le système de routage de Symfony au sein d'un fichier Javascript, typiquement lorsque l'on code une requête AJAX. Il permet de générer des URLs à partir de nos fichiers de routes et il est utilisable dans n'importe quel fichier PHP ou TWIG, mais pas Javascript. Pour pallier à ce manque, la solution est d'utiliser **FOSJSRoutingBundle**. Ce *bundle* va générer un fichier Javascript qui contient toutes les routes exposées de nos fichiers de routes. Il suffit alors d'inclure ce fichier dans nos vues et d'utiliser nos routes normalement dans les scripts JS.

3.2 Diagramme de Gantt et calendrier

Réalisation des objectifs

Diagramme de Gantt

Le diagramme de Gantt est généré grâce à une librairie Javascript appelée **DHTMLXGantt** détaillée page 10.

Au début de ce projet, la génération du diagramme était aléatoire, plantait dans la plupart des cas et les diagrammes générés n'étaient pas très lisibles. De plus, au clic sur une tâche dans le diagramme, la popup de détail d'une tâche de la vue du Board s'affichait correctement, mais pour réaliser cela l'étudiant précédent avait édité directement le code obfusqué de la librairie JS, ce qui n'est pas du tout recommandé.

Désormais, on utilise les événements proposés par la librairie pour afficher cette popup ou effectuer d'autres actions. Il n'y a plus aucun bug d'affichage connu. Les tâches sont regroupées en étapes et en domaines. Seules les tâches possédant une date de début et une date de fin sont affichées. Celles possédant uniquement une date de fin sont considérées comme des jalons. Lorsque l'on modifie les dates d'une tâche dans la popup, cela est immédiatement répercutée dans le diagramme. On peut en outre placer des liens de dépendances entre deux tâches, ou encore filtrer les tâches selon plusieurs critères.

Par ailleurs, il est aussi possible pour l'utilisateur courant de visualiser un diagramme de Gantt personnel regroupant toutes les tâches auxquelles il est assigné, parmi tous les projets auxquels il participe.

La capture d'écran suivante montre un exemple de diagramme de Gantt avec des liens de dépendances entre les tâches. On peut voir les filtres au dessus du diagramme.

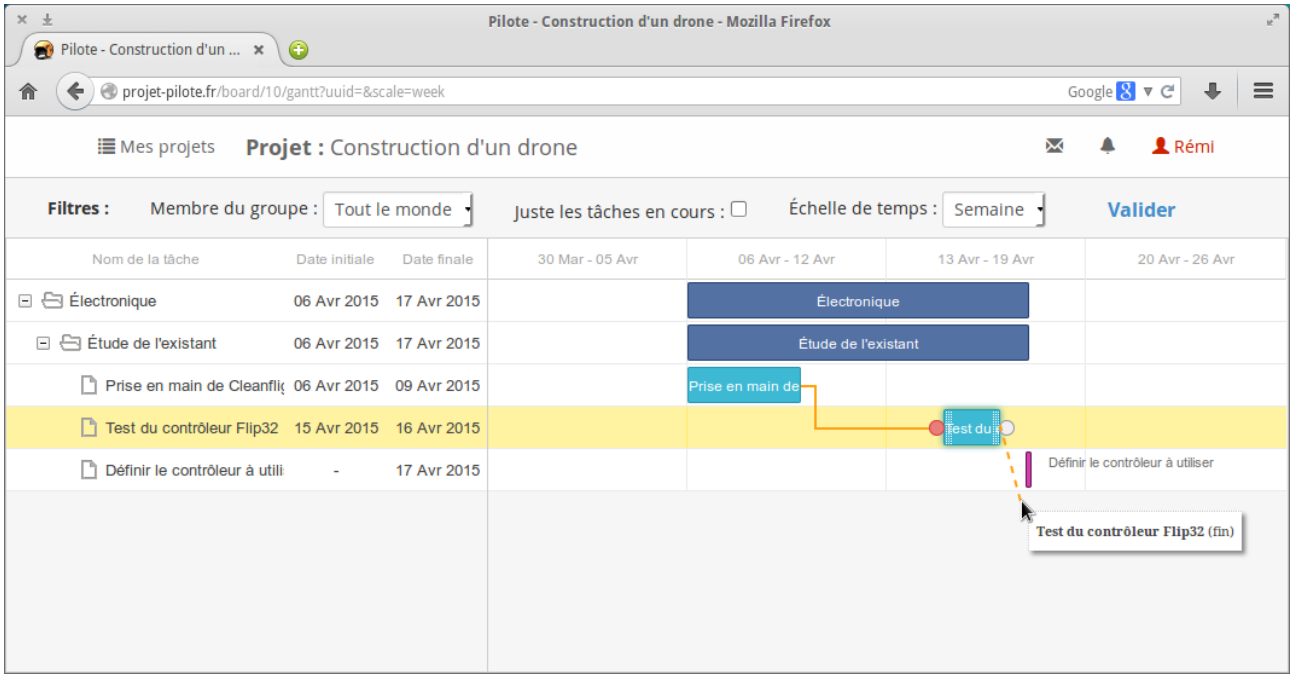


FIGURE 3.3 – Diagramme de Gantt d'un projet

Calendrier

Pour ce qui est du calendrier, nous avions choisi d'utiliser la librairie Javascript **FullCalendar** pour économiser du temps lors du développement de la vue car elle gère toutes les subtilités des dates (comme les mois de 28, 30 ou 31 jours ou les années bissextiles par exemple) et offre un rendu agréable et simple.

On peut désormais visualiser les tâches d'un projet dans un calendrier mensuel en grille, cliquer sur l'une d'elles pour afficher la popup de détail ou les déplacer directement dans le calendrier. La capture d'écran 3.4 présente le calendrier correspondant au diagramme de Gantt précédent.

Problèmes rencontrés

La librairie DHTMLXGantt étant bien documentée, je n'ai pas eu de problème particulier durant ces développements.

FullCalendar intègre une autre librairie, Moment.js pour manipuler les dates. Sa prise en main a demandé un petit temps d'apprentissage.

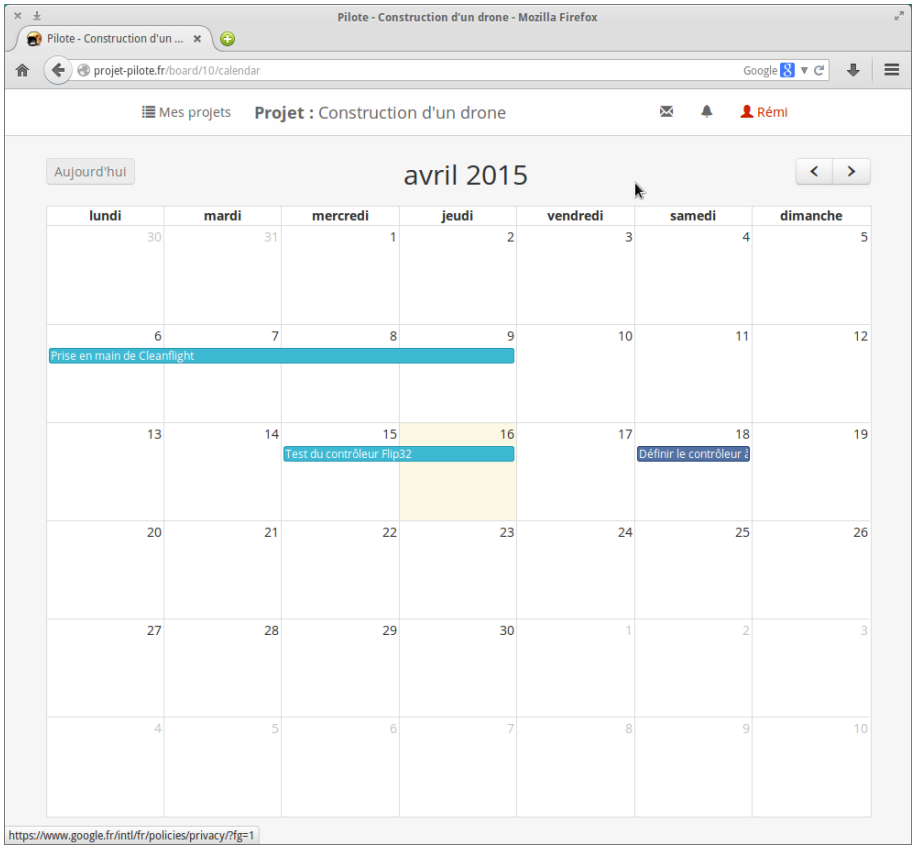


FIGURE 3.4 – Calendrier d'un projet

3.3 Zone d'administration

Réalisation des objectifs

Au début du PFE, cette zone consistait en trois pages statiques contenant des informations factices car nous n'avions pas eu le temps de la développer lors du PIL. Désormais, elle comporte quatre pages, comme on peut le constater sur la capture d'écran 3.5 :

- Un accueil affichant des informations générales comme la connexion avec le serveur de notifications ;
- Une liste des utilisateurs, avec des informations les concernant ainsi que la possibilité de les activer/désactiver, de les supprimer et de les promouvoir au rang d'administrateurs ;
- Une liste des projets similaire à celle des utilisateurs ;
- Une page permettant de créer à la volée des nouveaux utilisateurs.

Il est aussi possible de désactiver les inscriptions au site, ne laissant l'accès qu'aux personnes ayant déjà un compte (les administrateurs ayant toujours la possibilité de créer des comptes à la volée). Ce paramètre est très utile lorsque l'administrateur a configuré la connexion des utilisateurs via un serveur LDAP (voir la section 3.5).

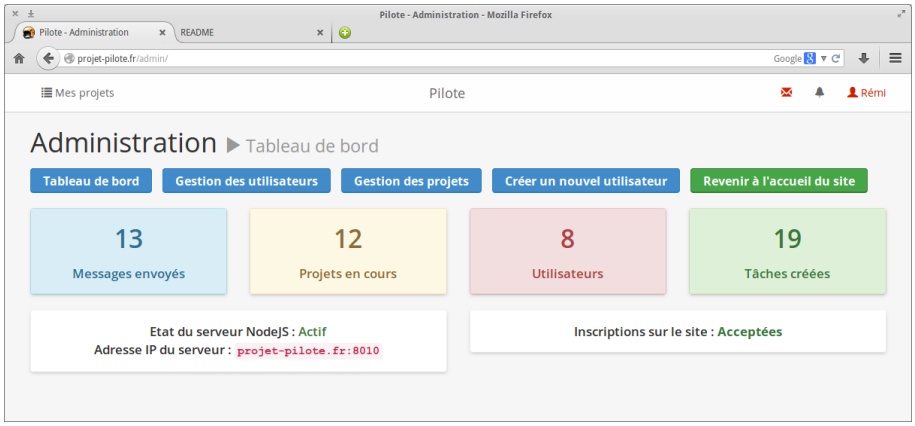


FIGURE 3.5 – Page d'accueil de la zone d'administration

3.4 Messagerie

Réalisation des objectifs

La messagerie était quasi-inexistante au début du PFE. Le *bundle* externe FOSMessageBundle était inclus dans le projet mais il n'était pas possible d'envoyer ou de recevoir de message, toujours par manque de temps à la fin du PIL.

Après avoir essayé de comprendre le fonctionnement et d'intégrer FOSMessageBundle à Pilote, j'ai décidé de l'abandonner et de développer moi-même le système de messagerie instantanée sans me baser sur un *bundle* existant.

PiloteMessageBundle propose ainsi de créer des discussions entre deux ou plusieurs personnes. Il existe trois types de discussions :

- Les **discussions privées**, qui peuvent être créées par n'importe quel utilisateur. Les membres de la conversation peuvent ajouter d'autres personnes à celle-ci.
- Les **discussions de projets**, créées automatiquement à la création d'un projet. Elle intègre évidemment tous les membres du projet.
- Les **discussions avec les administrateurs**. Chaque utilisateur simple peut contacter les administrateurs via la messagerie. Une discussion est alors créée avec cette personne et l'ensemble des administrateurs.

Il s'agit bien d'une messagerie instantanée : Chaque message posté est envoyé au serveur via une requête AJAX et est répercuté instantanément chez les autres clients ayant la même conversation affichée, grâce au système de notifications (voir la section 3.6).

PiloteMessageBundle contient trois entités, c'est à dire trois tables en base de données : Message, Thread et ThreadMetadata.

- Un **Message** contient un corps, une date de création, un expéditeur, et il est rattaché à un Thread.
- Un **Thread** représente une conversation. Il contient une liste de messages, un ensemble de métadonnées (une pour chaque membre de la conversation), un type, et peut éventuellement être rattaché à un Board (un projet).
- Un **ThreadMetadata** fait le lien entre une conversation et un utilisateur, et contient l'information de la lecture de la première par le second.

Problèmes rencontrés

Le *bundle* externe FOSMessageBundle choisi lors du PIL semblait être une bonne solution pour mettre en place rapidement une messagerie. Il souffrait néanmoins d'un manque de documentation qui a fait perdre quelques jours de développement. Au final il s'est avéré que ce *bundle* ne permettait pas de créer simplement les trois types de conversations prévues.

Le choix de développer moi-même et intégralement la messagerie, après avoir perdu plusieurs jours, a entraîné un petit retard sur le planning.

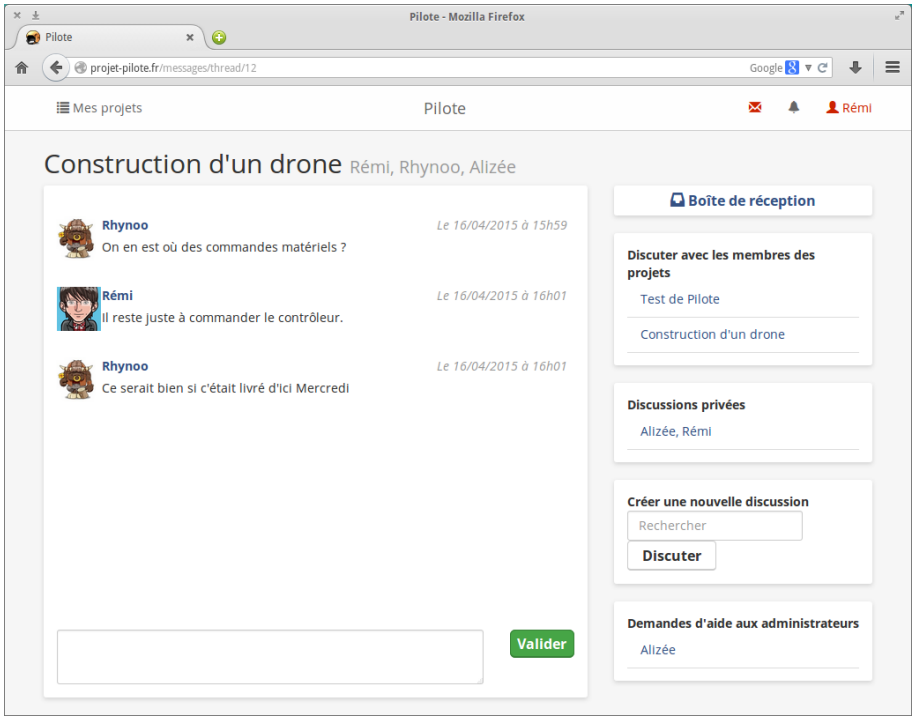


FIGURE 3.6 – Exemple de discussion lié à un projet

3.5 Authentification via un annuaire LDAP

Réalisation des objectifs

Dans l'idée de déployer Pilote sur le réseau de Polytech Tours, M. Ragot a proposé de permettre aux étudiants de se connecter en saisissant leurs identifiants de l'École plutôt que de se créer manuellement un compte. Polytech Tours, comme l'ensemble de l'Université François-Rabelais de Tours, utilise un annuaire LDAP référençant entre autres les étudiants et les professeurs. Il est nécessaire de posséder un compte spécial pour avoir le droit d'accéder à ces données. Ce compte a été fourni par le Service Informatique du D.I.

Pilote sera Open Source et ne sera pas réservé à Polytech, aussi la connexion via l'annuaire LDAP doit être une simple option. C'est pourquoi, sur le dépôt hébergeant le code source de Pilote, nous avons donc créé une **branche en parallèle** de la branche principale, qui contient la version de Pilote intégrant cette fonctionnalité. Les deux versions sont ainsi disponibles, avec ou sans la connexion au LDAP.

Le *bundle* externe **FR3DLdapBundle** a été utilisé pour réaliser cela. Pour l'intégrer à Pilote, il a fallu entre autres modifier notre classe `User` afin qu'elle implémente l'interface `LDAPUserInterface` de ce

bundle, et configurer les accès au serveur LDAP.

Cette configuration se passe dans le fichier app/config/config.yml.Le code concerné apparaît ci-dessous. On y retrouve les identifiants de connexion au serveur et le *Distinguished Name* (le *DN*) correspondant aux objets de l'annuaire qui représentent les étudiants et les professeurs.

```
fr3d_ldap:
    driver:
       host:     # IP du serveur
       port:     # Port du server
       username: # Identifiant de connexion au serveur
       password: # Idem
       useSsl:   false
    user:
        baseDn:     # Le DN correspondant aux étudiants et aux professeurs de l'École
        attributes: # Les attributs que l'on souhaite stocker dans notre base de données
                    # Ici, il s'agit des champs "username", "dn", "displayname" et "mail"
           - { ldap_attr: samaccountname,  user_method: setUsername }
           - { ldap_attr: DN, user_method: setDn }
           - { ldap_attr: displayname, user_method: setDisplayName }
           - { ldap_attr: mail, user_method: setEmail }
```

Le *DC* ressemble à peu près à ceci : OU=Etudiants,DC=univ-tours,DC=polytech.

On provoque ainsi un problème de sécurité. En effet, ce fichier est versionné et se trouve donc dans les dépôts. Pour éviter de diffuser les identifiants d'accès à l'annuaire de Polytech, il faut remplacer les informations "en dur" par des variables qui seront définies dans un fichier non versionné. C'est le rôle du fichier app/config/parameters.yml. Il faut donc ajouter quelques lignes à ce fichier et apporter certaines modifications à app/config/config.yml :

app/config/config.yml

```
fr3d_ldap:
    driver:
       host:     %ldap_host%
       port:     %ldap_port%
       username: %ldap_username%
       password: %ldap_password%
       useSsl:   false
    user:
        baseDn: %ldap_baseDn%
```

app/config/parameters.yml

```
parameters:
    ldap_host:     10.111.213.14
    ldap_port:     389
    ldap_username: ''
    ldap_password: ''
    ldap_baseDN:   OU=Etudiants,DC=univ-tours,DC=polytech
```

L'autre avantage de faire figurer ces variables dans ce fichier, est que le script d'installation de Pilote va chercher les variables situées à cet endroit pour en demander les valeurs à l'administrateur, lors de l'installation. Cette solution est donc à la fois sécurisée et pratique.

Problèmes rencontrés

Aucune documentation de la structure de l'annuaire LDAP de Polytech Tours n'est malheureusement disponible. On retrouve aussi plusieurs attributs similaires dans les objets représentants les étudiants. De plus, ceux représentant les professeurs et le personnel de l'École n'ont pas tout à fait la même structure. Nos nombreux tâtonnements nous ont fait perdre du temps et ont provoqué quelques bugs. Christophe Forycki avait déjà eu affaire à ces mêmes problèmes auparavant pour son PFE et son aide a été précieuse.

La manipulation des branches dans un dépôt SVN n'est pas aussi pratique qu'avec un système de gestion des versions moderne tel que Git. J'ai ainsi éprouvé quelques difficultés à transférer les commits d'une branche à l'autre. Après avoir réglé tous les conflits et corrigé toutes les erreurs, j'ai décidé d'abandonner le dépôt SVN de Polytech Tours pour le dépôt Git public sur le site GitHub, que nous utilisions l'année passée pour le PIL. Ma bonne connaissance de l'outil Git m'a ensuite évité d'autres problèmes de *versionning*.

3.6 Système de notifications

Réalisation des objectifs

Un outil collaboratif doit être le plus réactif possible lors des modifications. Au début du PFE, lorsque quelqu'un en effectuait une dans un projet, les autres personnes ayant affiché les mêmes informations sur leur écran devaient réactualiser manuellement la page pour la voir. Il était donc nécessaire de trouver une solution pour appliquer instantannément ces modifications chez les autres clients.

La solution choisie utilise un serveur **Node.js** et le module Socket.IO. Le premier permet de développer des applications côté serveur en Javascript. Il est beaucoup utilisé pour créer des serveurs *scalables* (réagissant bien à des fortes montées en charge) et il fonctionne principalement avec des événements, ce qui est tout indiqué pour des notifications en temps réel. **Socket.IO** facilite les communications en temps réel en créant des WebSockets entre les clients.

La question était ensuite de savoir comment faire communiquer notre serveur Node.js avec notre application en PHP. Pour cela, on utilise la librairie PHP **ElephantIO** qui permet de créer un WebSocket en PHP. Ainsi, notre serveur en PHP est un client du serveur Node.js. Il peut donc envoyer des notifications au serveur Node.js à chaque fois qu'il reçoit une requête AJAX par exemple.

Le serveur Node.js reçoit les notifications envoyées par l'application en PHP, les traite et en envoie de nouvelles aux clients concernés. Ceux-ci vont pouvoir appliquer les modifications dans la vue. Plus de détails sont disponibles dans la documentation technique [1]. Le diagramme de séquence 2.4 schématise les mécanismes aboutissant à l'affichage d'une notification chez un client à partir de l'action d'un autre client.

Afin de démarrer le serveur Node.js sur un serveur de production, nous utilisons **PM2**. C'est est un gestionnaire de processus pour monitorer les applications Node. Il les fait tourner en tâche de fond, les redémarre automatiquement après un crash, gère le *Load Balancing*, etc.

En plus de mettre à jour instantanément la vue des autres clients, il est aussi très pratique de notifier les utilisateurs, surtout s'ils ne sont pas sur la page concernée. Pour cela un menu des notifications a été

1. https://github.com/Artemis-Haven/pilote/blob/master/doc/contribution.md#envoi-des-notifications

intégré à la barre de menu en haut, et des notifications temporaires apparaissent quelques secondes dans le coin de l'écran, à la manières des principaux réseaux sociaux actuellement.

On peut voir un exemple de ces notifications sur la capture d'écran 3.7 ci-dessous : Le menu des messages (l'enveloppe) est devenu rouge, indiquant un message non lu. S'il s'agissait d'une notification concernant les projets, c'est le menu des notifications (la cloche) qui serait devenu rouge.

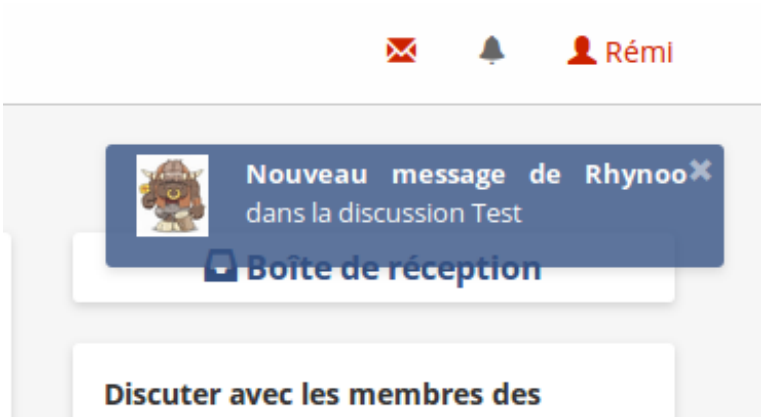


FIGURE 3.7 – Exemple de notification d'un nouveau message

Problèmes rencontrés

Les technologies utilisées ici sont très intéressantes et montrent un gros potentiel pour des applications web dynamiques. Mais elles nécessitent un certain temps d'apprentissage (surtout Node.js).

3.7 Refonte de l'interface

Ce travail ne figurait pas dans le Cahier de Spécification Système ou dans la liste des objectifs. L'an dernier lors du PIL nous avions discuté avec le Service Valorisation de l'Université qui nous avait conseillé de nous distinguer un peu plus de Trello en terme de design, pour éviter tout problème juridique de plagiat.

J'ai donc pris la liberté de m'y atteler sur mon temps libre. J'ai essayé d'appliquer un thème commun à l'ensemble des pages du site, avec des blocs blancs sur un fond gris pâle. Les boutons importants affichent une couleur vive pour se démarquer.

L'élément qui était visuellement le plus proche de Trello était la popup affichant le détail d'une tâche. En modifiant le style CSS et l'agencement de ses différents composants, j'ai atténué cette ressemblance. La vue du Board a aussi subi une refonte.

Lors de la mise en production de Pilote sur les serveurs de Polytech, l'une des demandes les plus fréquentes de mes premiers beta-testeurs était la possibilité d'appliquer différents thèmes aux projets. Toujours sur mon temps libre, j'ai ajouté un système de thèmes colorés qu'il est possible de choisir pour chaque projet. Cela offre un indice visuel assez pratique pour en trouver un rapidement sur la page d'accueil listant tous les projets par exemple.

En plus du thème blanc par défaut, il en existe donc quatre autres : Sable, Vert Sapin, Violet Polytech et Rouge Carmin.

Ces questions de design et d'ergonomie s'éloignent du cadre du PFE et sortent de mon domaine de compétences, bien que j'y accorde une grande importance.

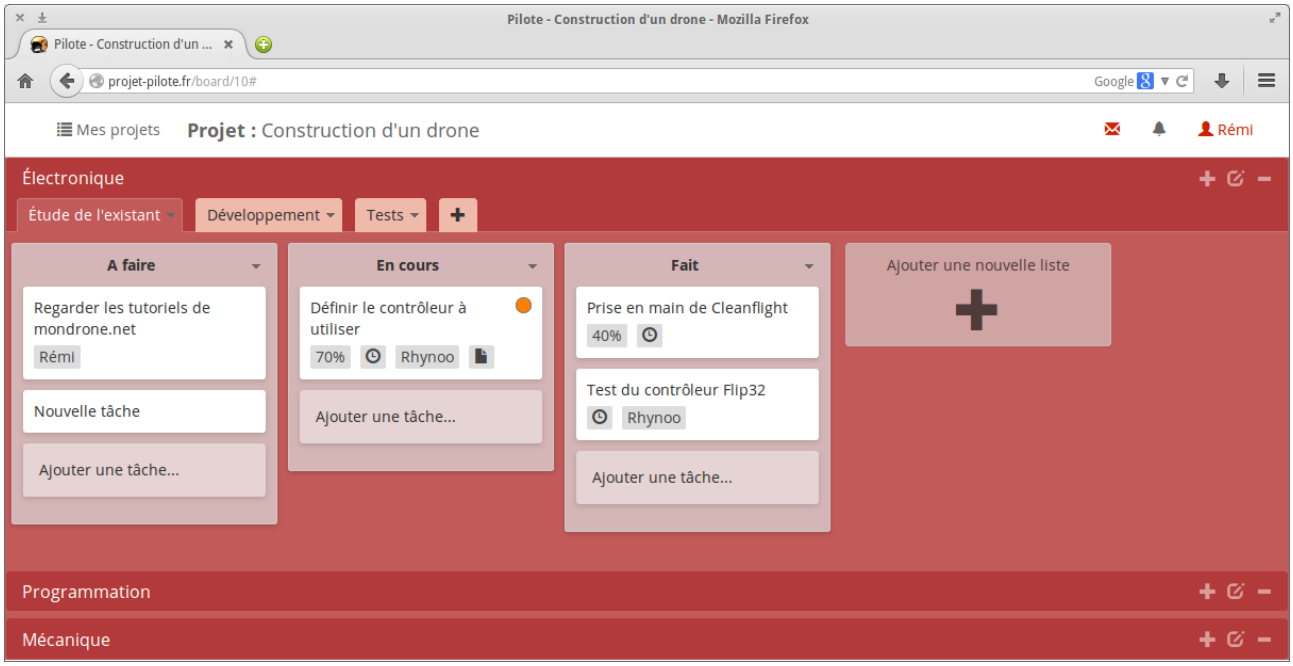


FIGURE 3.8 – Projet utilisant le thème Rouge Carmin

FIGURE 3.9 – Page listant les projets, reconnaissables à leur couleur

Gestion de projet

4.1 Méthodologie utilisée

Pour ce projet nous avons décidé de suivre la méthode **Scrum**. Pour un projet ne comportant aucun client et une équipe de développeur réduite à une seule personne, il n'est pas évident de respecter les rôles proposés par cette méthode, il a fallu l'adapter à la situation.

M. Ragot avait le rôle du **Product Owner** : il a défini les éléments du *Product Backlog* et leurs priorités au début du PFE, et a parfois pris la décision de les redéfinir pendant le développement, si besoin était.

Un Sprint durait six jours de travail. De septembre à décembre, nous avons donc eu trois Sprints. Puis de janvier à avril, chaque Sprint durait deux semaines. A la fin de chacun d'eux, nous organisions une **Revue de Sprint** avec M. Ragot, pour présenter et valider les tâches réalisées. Celui-ci apportait des remarques constructives et suggérait des améliorations. Ensuite nous discutions des tâches à effectuer dans le Sprint suivant, en se basant sur le *Product Backlog*.

4.2 Planification

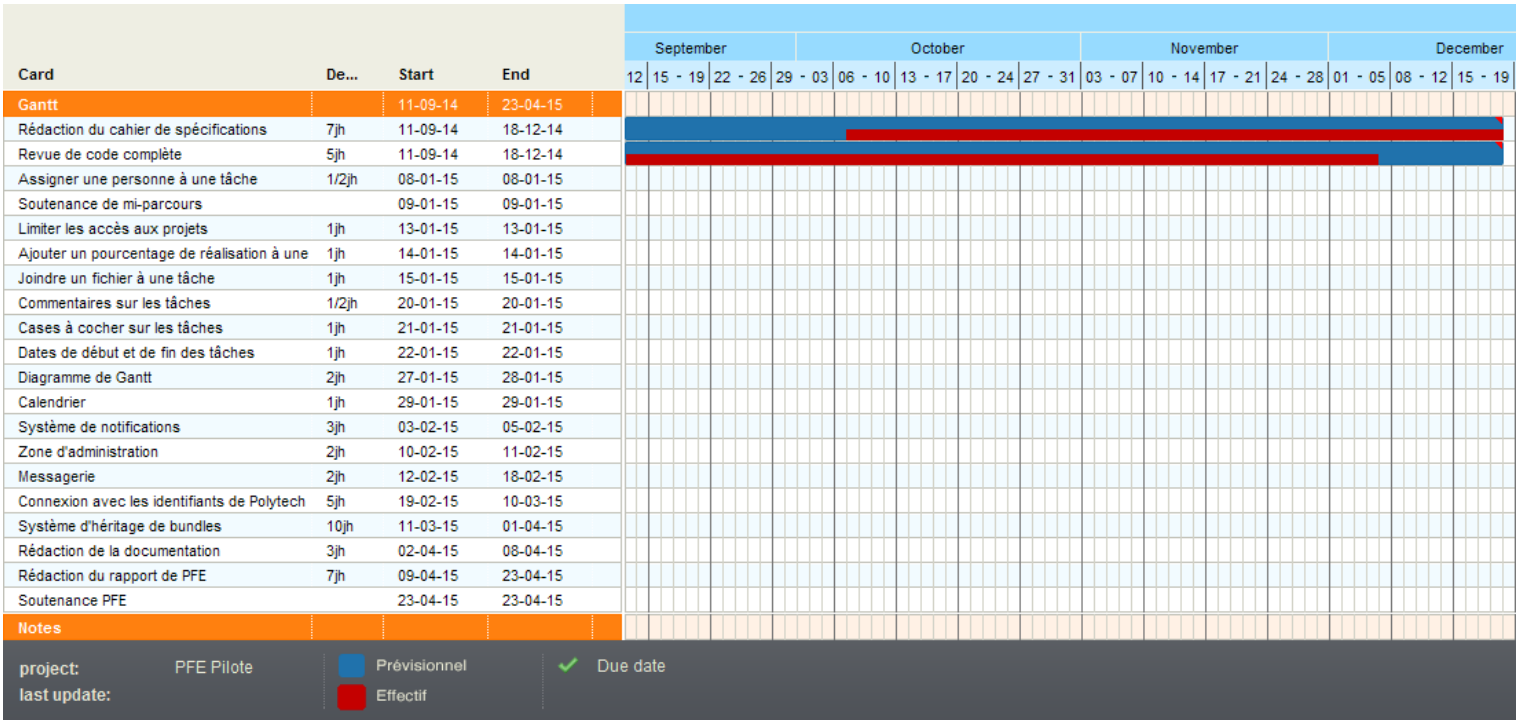
Au mois de décembre, nous avons défini un planning prévisionnel, sous la forme d'un diagramme de Gantt, à partir du *Product Backlog*. Sur la figure 4.1, on voit apparaître à la fois le planning prévisionnel (en bleu) et les dates effectives de réalisation des tâches (en rouge).

Les mois de septembre à décembre ont principalement été consacrés à reprendre le code existant, en faire une revue complète et le corriger, et à rédiger le Cahier des Spécifications Système.

De nombreuses tâches de développement de fonctionnalités de un à deux jours étaient prévues au mois de janvier. Ce planning a été bien respecté.

Au mois de février, le développement du système de notifications et de la messagerie ont pris plus de temps de prévu. Néanmoins j'ai profité de la disponibilité et des connaissances de Christophe Forycki sur la connexion à l'annuaire LDAP de Polytech Tours pour boucler en une journée et en avance sur le planning la tâche "Connexion avec les identifiants de Polytech", que j'avais estimée à cinq jours de travail sans son aide.

A partir du mois de mars, M. Ragot a décidé de mettre l'accent sur l'amélioration et la finition des fonctionnalités existantes. De plus, nous avons mis en ligne Pilote sur un serveur de l'École, afin qu'il puisse être testé par d'autres utilisateurs. Le planning a donc été grandement modifié : le système d'extensions qui devait être développé a été abandonné, au profit d'améliorations diverses du système de notifications et du diagramme de Gantt. Puis nous avons priorisé les bugs relevés par les Beta-testeurs et je les ai corrigés.



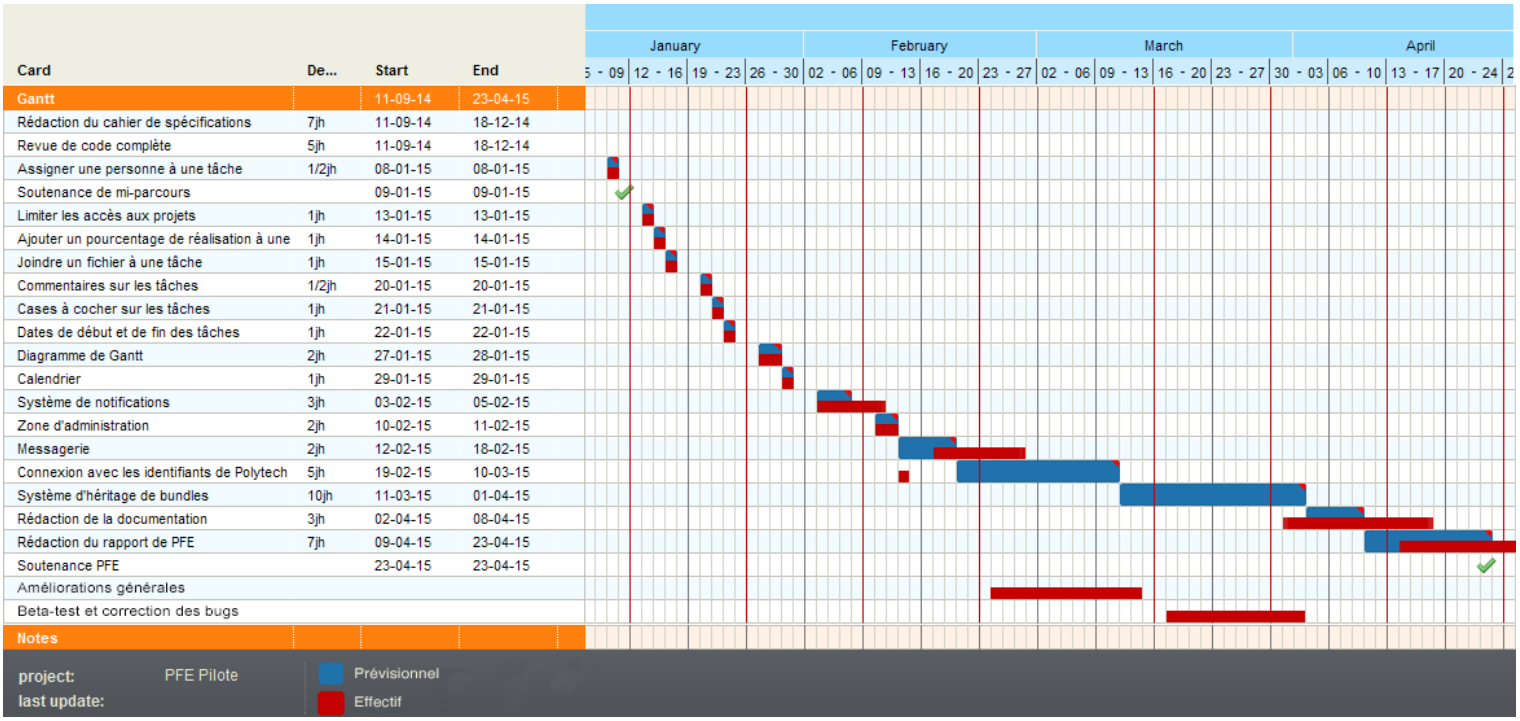


FIGURE 4.1 – Diagramme de Gantt du projet

4.3 Outils de gestion de projet

Trello, le "concurrent propriétaire" de Pilote, a été un outil très pratique pour m'organiser dans mon travail. Mais lorsque Pilote a été mis en ligne sur un serveur de l'École, j'ai migré vers ce nouvel outil. De plus, Pilote a été utilisé par les Beta-testeurs et moi-même comme *Bug Tracker* pour créer des tickets de

bugs, les prioriser et suivre l'avancement de leur correction.

Pour communiquer avec M. Ragot, en plus des mails et des Revues de Sprints, nous avons utilisé Redmine, notamment pour générer le diagramme de Gantt à partir des tâches et pour mettre par écrit le contenu du *Product Backlog* et des Revues de Sprints.

Bilan

5.1 Respect du Cahier de Spécifications

On pourrait résumer les critères du Cahier de Spécifications de la manière suivante : on souhaitait terminer le développement de l'application web Pilote, en ajoutant un certain nombre de fonctionnalités prévues mais non développées auparavant (plus deux nouvelles fonctions), afin de fournir une solution stable, fonctionnelle, utile et Open Source.

Les fonctionnalités prévues lors du PIL sont toutes développées et fonctionnelles. Grâce à la période de Beta-test, de corrections et d'améliorations, on peut dire que Pilote est désormais stable et utile. Il est aussi Open Source car l'ensemble des sources et de la documentation technique est disponible sur GitHub sous licence GNU GPL v3 (voir la section 5.3). La connexion via un annuaire LDAP fonctionne aussi, mais le système d'extensions a été abandonné.

Système d'extensions

Le système d'extensions devait permettre aux développeurs extérieurs de créer très simplement un bundle de gestion des tâches dérivé de celui par défaut de Pilote et ajoutant des fonctionnalités plus spécifique à leurs besoins. L'idée était de ne pas surcharger l'interface du site avec les fonctionnalités peu usuelles et de la garder épurée.

Il s'agissait dans le planning de la dernière tâche de programmation, avant la rédaction de la documentation et la fin du PFE. M. Ragot et moi-même avons estimé qu'il était plus important de se concentrer sur la correction des bugs et la finalisation des fonctions déjà développées afin de garantir la stabilité de l'application. De plus, l'interface actuelle du Board et de la popup de détail d'une tâche est suffisamment flexible pour être, au choix, concise ou plus complète selon les besoins de l'utilisateur. Le besoin d'un système d'extensions est donc assez limité.

Comme estimé dans le Cahier de Spécifications, cette tâche devrait prendre 7 à 10 jours de développement, de tests et de rédaction de documentation.

Tests de compatibilité des navigateurs

Le développement de Pilote s'est effectué sur des machines équipées d'OS X ou de distributions Linux, les tests approfondis de compatibilité des navigateurs lors du développement ont donc été fait avec Firefox, Chrome et Safari. Par ailleurs, aucun des Beta-testeurs n'utilisait Internet Explorer. Ce dernier n'a donc pas subi de tests approfondis, alors qu'il s'agit d'un des trois navigateurs les plus utilisés au monde.

Néanmoins, l'utilisation du framework CSS Bootstrap et de jQuery, qui sont tous deux réputés pour leur compatibilité avec IE, devrait limiter les éventuels problèmes.

Ces tests et les éventuelles corrections devraient occuper entre 1 et 3 jours.

5.2 Mise en production

Pour les besoins de tests de mise en production, le Service Informatique de l'École a mis à ma disposition une machine physique tournant sur Ubuntu 14.04 LTS sur laquelle j'ai pu déployer Pilote. Puis, après que celle-ci ait rendu l'âme, et afin de rendre Pilote accessible et testable par l'ensemble des étudiants de Polytech Tours, j'ai réinstallé une version plus avancée de Pilote sur une machine virtuelle hébergée sur les serveurs de l'École et tournant sur Ubuntu 14.10. Cela m'a permis de rédiger la documentation de mise en production et de mise à jour du serveur et de le faire tester par plusieurs étudiants de l'École. L'équipe de Beta-testeurs a fait remonter les bugs qu'ils ont découverts. C'était aussi le meilleur moyen de tester le bon fonctionnement de la branche LDAP : un serveur hébergé par Polytech Tours est indispensable pour permettre l'authentification des étudiants avec leurs identifiants, car l'annuaire LDAP n'est pas accessible aux machines extérieures.

Les applications basées sur Symfony sont connues pour avoir des performances moins bonnes sur des serveurs Windows, c'est pourquoi je me suis concentré sur l'installation sur des serveurs Linux. Néanmoins, il est tout à fait possible de l'installer sur Windows.

La documentation technique contient la démarche à suivre pour installer et configurer Pilote sous Linux (voir la section 5.3). En voici les principales étapes :

— Installer les paquets nécessaires à son bon fonctionnement : Apache, PHP, MySQL, Node.js, PM2, etc.
— Cloner le dépôt Git depuis GitHub.
— Lancer le script d'installation. Celui-ci demande des informations à l'administrateur (identifiants de connexion à la base de données ou à l'annuaire LDAP par exemple) pour compléter le fichier de paramètres.
— Démarrer le serveur Node.js (pour les notifications) avec le gestionnaire de processus PM2.

À ce stade, le site est opérationnel. On pourra ensuite :

— Configurer Apache pour avoir des URLs bien propres.
— Créer des comptes utilisateurs ou administrateurs directement depuis le terminal.

Comme pour l'installation, la documentation technique explique comment mettre à jour Pilote et que faire en cas de redémarrage.

Pour le mettre à jour, il faut d'abord récupérer les sources du dépôt (`git pull`), puis redémarrer le serveur Node.js et vider le cache de Symfony (en mode de production, aucune modification n'est effectuée réellement tant que le cache n'a pas été vidé).

En cas de redémarrage de la machine, il suffit de relancer le serveur Node.js avec PM2.

5.3 Livrables

Les livrables sont composés du Cahier de Spécifications, du présent rapport final de PFE, du code source de l'application web et de divers documents de documentation.

Comme de très nombreux projets Open Source, le code source et la documentation sont hébergés sur GitHub [1].

On retrouve :

— Une présentation générale de Pilote et de ses principales fonctionnalités ;
— Une documentation utilisateur [2], expliquant précisément comment naviguer dans le site et utiliser toutes ses fonctions ;
— Une documentation technique pour l'installation [3] et la mise à jour [4] de Pilote ;
— Une documentation technique pour d'éventuels contributeurs [5] décrivant la structure du projet et le fonctionnement détaillé des notifications.

Licence Open Source

Pilote utilise un grand nombre de composants, tous sous licence Open Source. La très grande majorité d'entre eux sont sous licence MIT, à quelques exceptions notables. DHTMLXGantt est sous double-licence : une version sous licence commerciale et une sous licence GNU GPL v2 qui est privée de certaines fonctionnalités. jQuery est aussi proposé sous double licence : MIT et GNU GPL v2.

Nous avons choisi de placer Pilote sous la licence **GNU GPL v3**. Elle permet d'accéder, de modifier et de diffuser le code source à n'importe qui, à condition que toutes les modifications d'une version distribuée soient retournées à la communauté sous la même licence. Cela empêche une éventuelle entreprise de reprendre Pilote, de le modifier et d'en faire un usage commercial. La licence MIT est compatible avec GNU GPL v3.

5.4 Possibilités d'évolution

M. Ragot, l'équipe des Beta-testeurs et moi-même fourmillons d'idées pour faire évoluer Pilote.

Dans l'optique d'en faire un vrai concurrent à l'usine à gaz Open Source qu'est Redmine, chaque projet pourrait intégrer son propre Wiki, ou bien une page permettant de consulter l'état et le contenu d'un dépôt GitHub ou Gitlab.

Il serait aussi intéressant d'offrir plus de flexibilité sur la visibilité des projets : ceux-ci pourraient être soit entièrement réservés à leurs membres, soit accessibles à tous en lecture seule, comme sur Trello.

Pilote pourrait à terme devenir un concurrent sérieux de ces deux acteurs aujourd'hui incontournables de la gestion collaborative de projets.

1. Dépôt public de Pilote : https://github.com/Artemis-Haven/pilote
2. Doc. utilisateurs https://github.com/Artemis-Haven/pilote/blob/master/doc/utilisateur.md
3. Doc. technique Installation : https://github.com/Artemis-Haven/pilote/blob/master/doc/installation.md
4. Doc. technique Maintenance : https://github.com/Artemis-Haven/pilote/blob/master/doc/maintenance.md
5. Doc. technique Contributeur : https://github.com/Artemis-Haven/pilote/blob/master/doc/contribution.md

5.5 Conclusion

Pilote est désormais opérationnel. Il propose une interface de gestion collaborative de projets avancée et dynamique, avec un grand nombre d'options liées aux tâches, une messagerie adaptée aux projets et une génération de diagrammes de Gantt. Il est installable et utilisable par tout un chacun.

La réalisation d'un tel projet a été particulièrement instructive car elle couvre toutes les phases de développement, de la conception à la mise en production, en passant par le développement, les tests et la rédaction d'une documentation. Elle m'a permis d'enrichir mes connaissances du framework Symfony, mais surtout de me faire découvrir plusieurs technologies modernes et intéressantes, comme Node.js par exemple.

Ce projet était d'autant plus intéressant qu'il ne s'agissait pas d'un simple travail d'exécutant où l'on se contente de réaliser les tâches confiées. Le système de notifications par exemple a requis une phase de recherche, puis il a fallu décider de la meilleure solution avant de la mettre en pratique. Il s'agissait donc plutôt d'un réel travail d'ingénieur nécessitant une certaine prise de responsabilité et d'initiative.

Bibliographie

[1] Fabien Potentier et SensioLabs. *The Symfony Book (FR)*. URL : http://symfony.com/fr/doc/current/book/index.html.

[2] Wisembly. *Elephant.io - Websocket client written in PHP*. URL : https://github.com/Wisembly/elephant.io.

[3] Nicolas Chaulet. *ElephantIOBundle - Elephant.io library integration in Symfony2*. URL : https://github.com/nchaulet/ElephantIOBundle.

[4] Maks3w. *FR3DLdapBundle*. URL : https://github.com/Maks3w/FR3DLdapBundle.

[5] Twitter. *Bootstrap*. URL : http://getbootstrap.com/components/.

[6] MindMup. *Bootstrap-Wysiwyg - Tiny WYSIWYG rich text editor for Bootstrap*. URL : http://mindmup.github.io/bootstrap-wysiwyg/.

[7] Dave Gandy. *FontAwesome, The iconic font and CSS toolkit*. URL : http://fortawesome.github.io/Font-Awesome/.

[8] jQuery. *jQuery API Documentation*. URL : http://api.jquery.com/.

[9] Mike Alsup. *jQuery Form Plugin*. URL : http://jquery.malsup.com/form/.

[10] DHTMLX. *Gantt Docs*. URL : http://docs.dhtmlx.com/gantt/.

[11] Adam Shaw. *FullCalendar - Javascript Event Calendar (jQuery Plugin)*. URL : http://fullcalendar.io/.

[12] MadMG (TAC GmbH). *Moment.js - Docs*. URL : http://momentjs.com/docs/.

[13] Joyent Inc. *Node.js Manual & Documentation*. URL : https://nodejs.org/api/.

[14] Socket.IO. *Socket.IO - Docs*. URL : http://socket.io/docs/#.

[15] Alexandre Bacco et OpenClassrooms. *Développez votre site web avec le framework Symfony2*. URL : http://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-symfony2.

[16] Sébastien Chopin. *Débuter avec Socket.io*. URL : http://atinux.developpez.com/tutoriels/javascript/debuter-avec-socket-io/.

[17] Nicolas Cabot. *Notifications via websocket avec Symfony2 et Node.js - Lexik Montpellier*. URL : http://devblog.lexik.fr/symfony2/notifications-via-websocket-avec-symfony2-et-node-js-2679.

[18] Alexandre Henriet. *Authentification LDAP avec Symfony 2.1 et FOSUserBundle*. URL : http://blog.henriet.eu/authentification-ldap-avec-symfony-2.1-et-fosuserbundle.html.

Pilote, outil collaboratif de gestion de projets

Département Informatique
5e année
2014 - 2015

Rapport de Projet de Fin d'Études

Résumé : Développement d'un outil collaboratif de gestion de projets commencé l'an dernier, en ajoutant plus de fonctionnalités : diagramme de Gantt, zone d'administration, messagerie, notifications, Authentification via un annuaire LDAP ...

Mots clefs : Outil de gestion de projets, Symfony, Application web, Kanban, Gantt, LDAP

Abstract: Development of a collaborative projects management tool started last year, adding more features: Gantt chart, administration zone, messaging, notifications using Node.js, authentication using a LDAP directory, ...

Keywords: Projects management tool, Symfony, Web app, Kanban, Gantt chart, LDAP

Encadrant
Nicolas RAGOT
nicolas.ragot@univ-tours.fr

Étudiant
Rémi PATRIZIO
remi.patrizio@etu.univ-tours.fr

Université François-Rabelais, Tours

DI5 2014 - 2015