

SYSTÈME DE SUIVI DES DOCTORANTS

Projet Transverse II

Page à enlever et remplacer par le pdf “page de garde”.

Kerfalla Cissé
Bernard Huynh
Nikita Missiri
Paul Mulard
Ismet Cem Turhan

Table des matières

1. Introduction	3
1.2. Description du projet et de sa problématique	3
2. Analyses des besoins	5
2.1. Analyse des objectifs	5
2.2. Analyse des exigences	6
3. Conception	12
3.1 Modèle conceptuel	12
3.2. Modélisation logique de la base de données	12
3.3. Architecture du système	13
3.4. Déploiement	15
4. Interfaces et test utilisateurs	17
5. Implémentation des plugins	29
5.1. Plugin "Study tracker"	29
Création automatique des étapes et des tâches à partir d'un fichier XML	29
Algorithmes liés au calcul de semestres	31
Calcul du nombre de semestres entre deux dates	31
Algorithme d'ajout de semestres à une date	35
Vue du doctorant	36
Vue du superviseur	37
Vue du conseiller aux études et du directeur	37
5.2. Plugin Theme	38
6. Tests du système	39
6.1. Tests unitaires	39
Tests sur la création automatique des étapes et tâches de doctorat à partir d'un fichier XML	39
Tests sur les algorithmes de la timeline	39
6.2. Tests fonctionnels	41
Tests sur la vue du doctorant	41
Tests sur la vue superviseur	42
Tests sur la vue conseiller aux études	42
7. Gestion de projet	44
7.1. Scrum et gitlab	44
Le vocabulaire gitlab	45
7.2. Utilisation de l'outil de communication Discord	45
7.3. Difficultés d'implémentation	47

8. Conclusion

48

8.1. Travaux futurs

48

1. Introduction

Le projet sur lequel nous avons travaillé est un service numérique de suivi des doctorants. Ce projet est mandaté dans un contexte de développement des sciences et des services numériques à l'UNIGE. Il s'inscrit dans un plus grand projet, l'automatisation du règlement d'études, qui a pour but d'élaborer et d'instancier les différents règlements d'études pour l'étudiant et pour les différentes parties prenantes.

Suite à une profonde analyse de sa problématique et les besoins utilisateurs, nous sommes passés à la conception du système multi-vues. Enfin, nous avons réalisé plusieurs séries de tests, unitaires et fonctionnels.

1.2. Description du projet et de sa problématique

La gestion des étudiants durant leur formation est un ensemble de tâches administratives dont leur mise en œuvre demande un certain nombre de moyens humains et matériels. Cependant, ces derniers n'ont pas toujours été optimisés pour répondre aux réels besoins des différentes parties prenantes, qui incluent notamment les étudiants et la Direction.

Dans le cas du suivi des doctorants à l'Université de Genève, deux problèmes principaux sont à résoudre. Premièrement, il est difficile pour la Direction, les superviseurs, le secrétariat et le conseiller aux études de suivre chaque étudiant dans les différentes étapes du doctorat, particulièrement quand le nombre de doctorants est grand. Deuxièmement, il est compliqué pour les étudiants de rechercher les informations nécessaires à l'avancement du doctorat. De ce fait, organiser son emploi du temps est une tâche fastidieuse.

A l'heure actuelle, le suivi des doctorants se fait avec différents moyens mis à disposition par l'université. D'un côté, la gestion se fait avec Excel, donc la mise à jour des données se fait manuellement, et d'un autre côté, pour les doctorants, la recherche d'informations se fait par les différents sites web hébergés par l'unige, par mail ou oralement, ce qui n'est souvent pas très pratique pour retrouver certaines informations qui ont été dit ou écrit.

Notre solution, en prenant en compte des fonctionnalités qui ont été développées l'année passée, est de mettre à disposition un système centralisé avec plusieurs vues, une pour chaque acteur. En plus des doctorants, les parties prenantes sont les superviseurs, le conseiller aux études et la direction du programme doctoral (directeur-riche).

2. Analyses des besoins

2.1. Analyse des objectifs

Suite aux entretiens avec les parties prenantes du système, nous avons pu identifier les problèmes qu'elles rencontrent et donc les objectifs de notre projet. Une modélisation des objectifs découle de cette analyse, dont la figure est représentée ci-dessous :

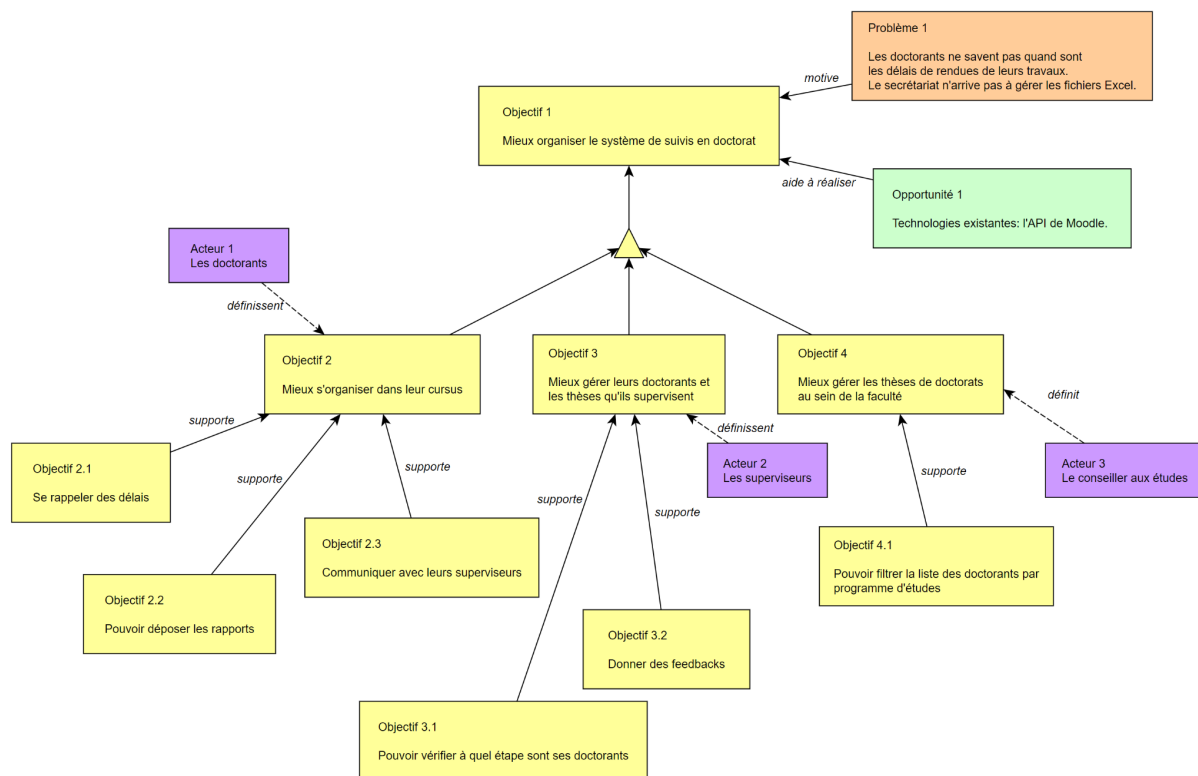


Fig. 1: Modèle des objectifs

En effet, le problème identifié est que le secrétariat, qui gère les délais, utilise un fichier tableur Excel pour stocker notamment les délais personnalisés de chacun des doctorants et doit envoyer des rappels manuellement par e-mail à chacun des doctorants. Par conséquent, il n'arrive pas à gérer le suivi de tous les doctorants inscrits au programme. Les doctorants, quant à eux, ne savent pas où récupérer les informations nécessaires, tels que le règlement d'études ou les délais de rendus pour leurs sujets de thèse ou leurs rapports annuels (*Problème 1*). Ce problème motive l'objectif principal qui est de mieux organiser le système des suivis des doctorants avec le développement d'un service (*Objectif 1*). Les technologies déjà existantes, notamment l'API de *Moodle*, un LMS qui fonctionne avec *Apache*, *PHP*, *MySQL* et *Mustache*, aide à réaliser cet objectif. En effet, l'aspect open-source de ce service donne la possibilité de créer des plugins qui répondent aux besoins de nos

clients. (*Opportunité 1*). Afin de réaliser cet objectif, il faut pouvoir réaliser trois sous-objectifs définis chacun par les différents acteurs du système.

Pour les doctorants, l'objectif est de mieux s'organiser dans leur cursus de doctorat (*Objectif 2*). Ce serait notamment de pouvoir se rappeler des délais de rendus de leur sujet de thèse et de leurs rapports annuels pour adapter leurs modes de travail (*Objectif 2.1*). Ils ont également exprimé le souhait de déposer les rapports sur une plateforme (*Objectif 2.2*) et dans la même occasion pouvoir recevoir des feedbacks sur leurs travaux de recherche et communiquer avec leurs superviseurs sur ces travaux (*Objectif 2.3*).

Les superviseurs, quant à eux, voudraient mieux gérer leurs doctorants associés et les thèses qu'ils supervisent (*Objectif 3*). Plus précisément, ils souhaitent dans la gestion de doctorants qu'ils suivent de pouvoir voir où ils en sont dans leur cursus de doctorat, à savoir s'ils doivent suivre des cours ou apporter une assistance à ces cours, s'ils doivent participer à des conférences, ou s'ils doivent bientôt soumettre leur sujets de thèse ou leurs rapports annuels (*Objectif 3.1*) et dans le cas des rendus de rapport, pouvoir donner un feedback à ses doctorants pour améliorer leurs travaux avant la soumission auprès de la communauté scientifique (*Objectif 3.2*).

Enfin, le conseiller aux études d'une faculté est également partie prenante du système et souhaite pouvoir mieux gérer les thèses de doctorats au sein de la faculté (*Objectif 4*). Plus précisément, le conseiller aux études souhaite pouvoir filtrer la liste des doctorants qu'il pourra visualiser par programme doctorales dans sa faculté (*Objectif 4.1*).

Tous ces sous-objectifs viennent supporter l'objectif principal de chacun des acteurs. Il peut y avoir d'autres fonctionnalités qui peuvent venir supporter les objectifs des acteurs, pour lesquelles les acteurs respectifs n'ont pas encore exprimé leurs besoins.

2.2. Analyse des exigences

Concernant le service de suivi des doctorants, le service est divisé en quatre principale exigence:

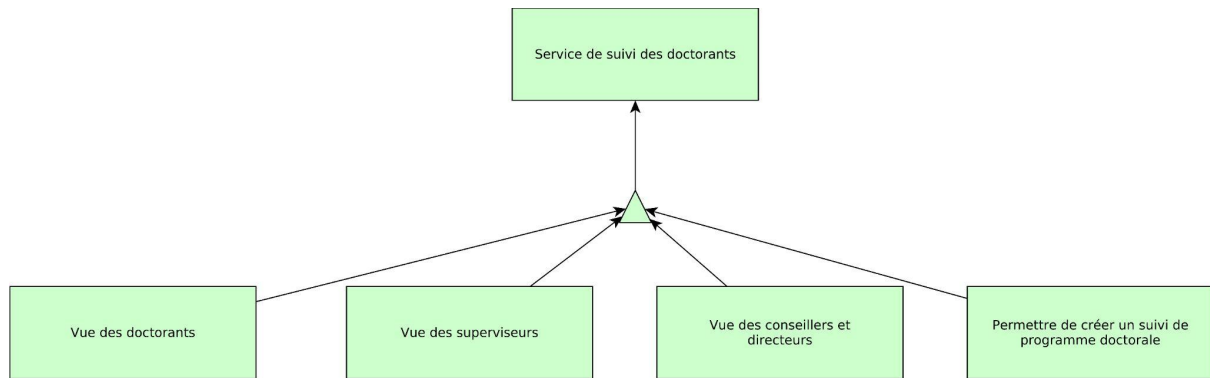


Figure 2.a: Modèle des exigences - les différentes exigences

Les trois premières correspondent aux fonctionnalités des vues des différents utilisateurs et la dernière à la création de suivi de programme doctorale. En effet, chaque type d'utilisateur pourra faire des actions différentes sur la plateforme concernant le suivi d'un programme doctorale.

La création de suivi de programme doctorale:

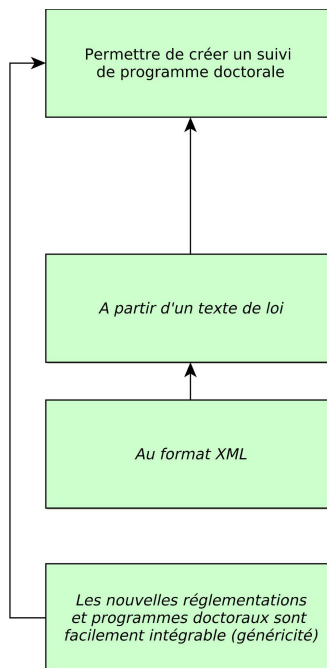


Figure 2.b: Modèle des exigences - création de suivi de programme doctorale

Les utilisateurs du service doivent pouvoir créer un suivi de programme doctorale. En prenant un texte de loi comme référence, le système crée automatiquement une séquence de tâches et d'étapes à suivre pour compléter un doctorat.

Afin d'avoir un système générique qui permet d'intégrer facilement de nouveaux programmes doctoraux et leur texte de loi, nous avons opté pour une représentation

des programmes au format XML. Ce choix est explicité dans le [chapitre plugin “study tracker”](#).

La première vue concerne les doctorants:

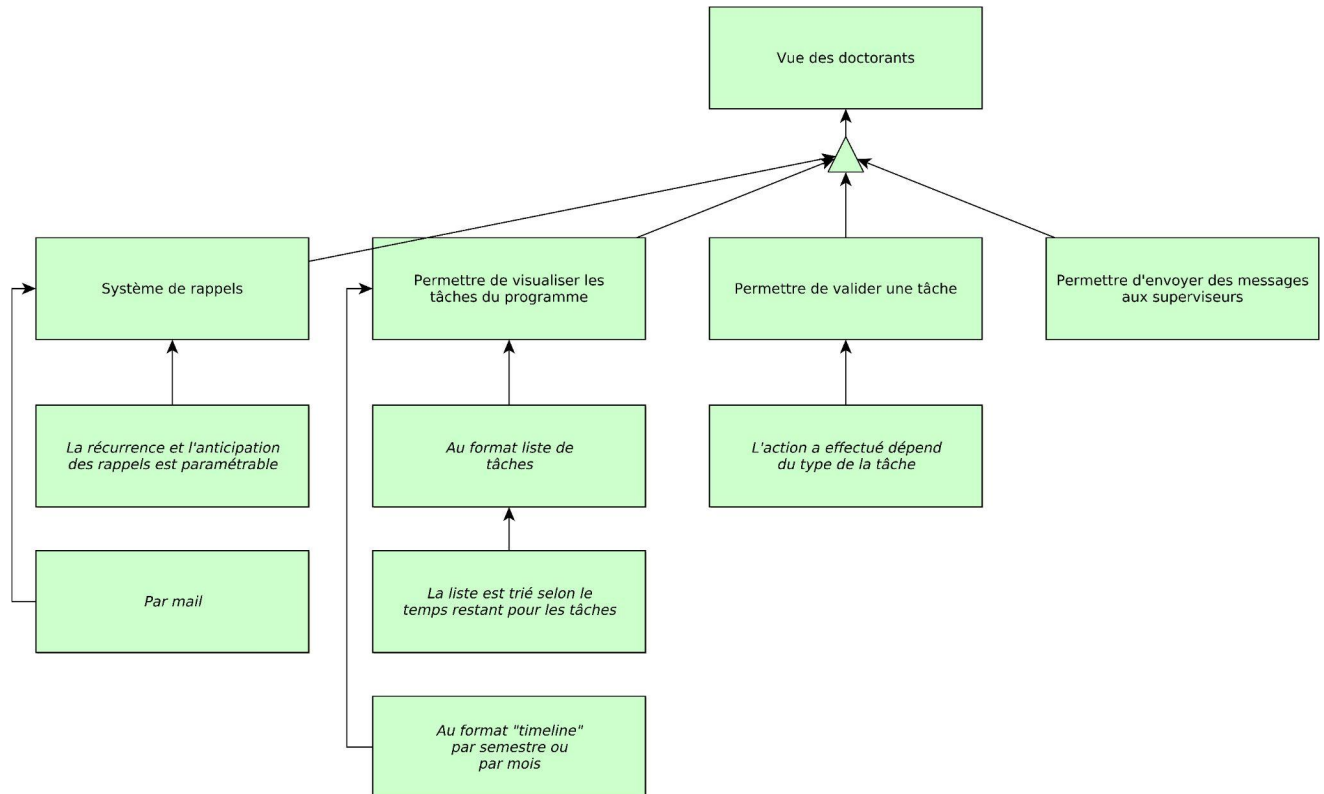


Figure 2.c: Modèle des exigences - vue des doctorants

Ce service a pour but d'aider les doctorants à organiser leur doctorat.

Les étudiants pourront visualiser les étapes de leurs doctorats: ce qu'ils ont accompli, ce qui leur reste à faire et les délais associés à chaque étape. De plus, ils pourront choisir de visualiser leurs étapes sous forme de liste ou alors sous forme de timeline.

En plus de cela, un système de rappel par mail aura pour rôle d'alerter les doctorants sur les deadlines proches. Pour avoir un service qui répond plus précisément aux besoins des doctorants, la temporalité et la récurrence des rappels sera paramétrable (un doctorant pourrait avoir besoin d'avoir un rappel un mois avant, tandis qu'un autre d'avoir en plus de cela un rappel trois mois avant et une semaine avant).

Les étudiants pourront valider/compléter les tâches qui constituent leur programme. Pour cela, ils devront faire une action comme par exemple soumettre un document ou simplement valider le fait qu'ils ont suivi un cours. Pour les documents, les doctorants n'ont qu'à les uploader sur la plateforme, ensuite leurs superviseurs et éventuellement les autres acteurs du service pourront corriger et/ou valider leurs rendus.

Finalement, les étudiants auront la possibilité de discuter avec leur responsable de thèse directement sur la plateforme afin de poser des questions mais aussi de répondre et collecter les retours de leur responsable. Les doctorants auront la possibilité de filtrer les conversations par tâches ou alors par personne.

La deuxième vue concerne les superviseurs :

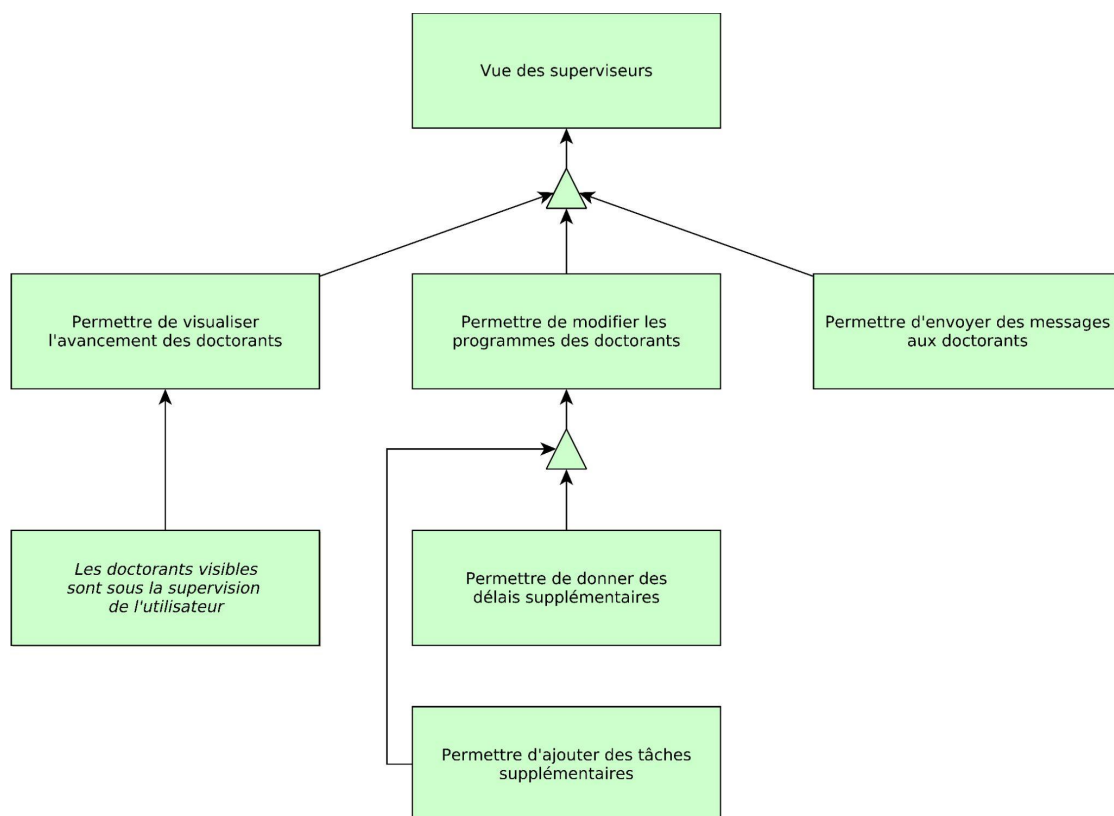


Figure 2.d: Modèle des exigences - vue des superviseurs

Cette vue sera accessible au superviseur/responsable de doctorants. Elle permet de visualiser les informations des doctorants dont l'utilisateur est responsable (et que ceux-ci). En effet, les données personnelles concernant l'avancement du doctorat d'un étudiant ne doivent être consultables que par ces derniers.

Au niveau des différents suivis qui ont été instanciés pour les doctorants, ils devront être modifiables. Il faut qu'un superviseur puisse modifier les délais de certaines

tâches, afin notamment de donner des délais supplémentaires (par exemple en cas de maladie). Aussi, le superviseur pourra ajouter de nouvelles tâches à un suivi d'un doctorant. Grâce à cela, les programmes seront personnalisables et un superviseur pourra ajouter par exemple un cours supplémentaire pour un doctorant.

Finalement, les responsables pourront consulter l'avancement des doctorants, c'est-à-dire les étapes qu'il/elle a achevé et ce qui lui reste à faire. Cela leur permettra de visualiser facilement l'avancement de tous leurs doctorants et s'il le faut de discuter avec eux. Pour cela, ils pourront consulter les documents remis par les doctorants et envoyer des messages. Ces fonctionnalités permettront, notamment, aux responsables de donner leur retour directement sur la plateforme.

La troisième vue concerne les directeurs et les conseillers d'études:

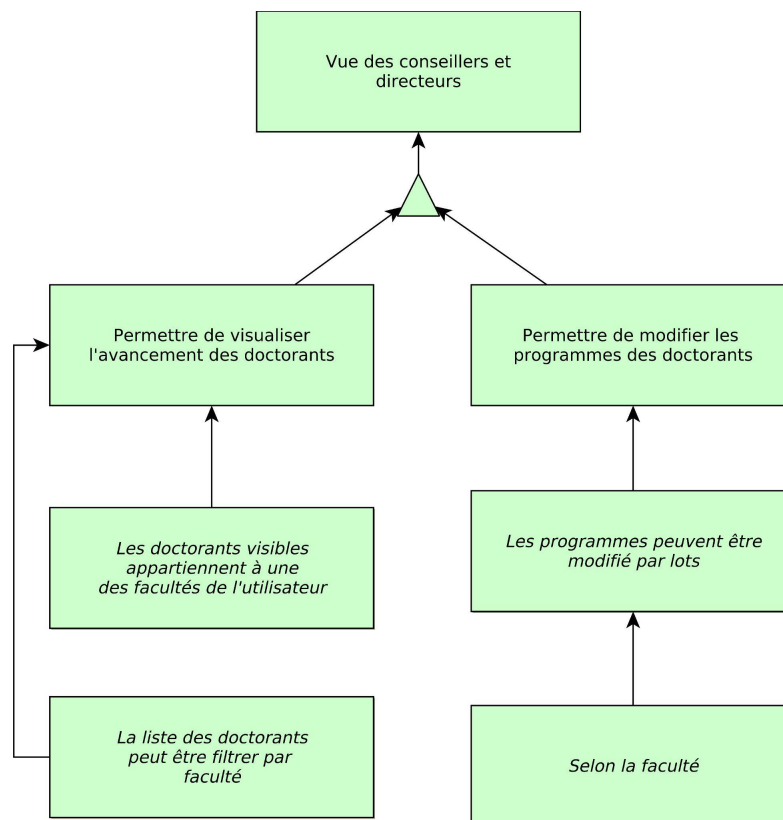


Figure 2.e: Modèle des exigences - vue des conseillers/directeurs

Pour les mêmes raisons que la vue superviseur, la vue des conseillers/directeurs ne sera accessible qu'aux utilisateurs authentifié comme tel et ceux-ci ne pourront voir que les doctorants inscrits dans leurs propre faculté. De plus et afin de faciliter la visualisation de l'avancement des doctorants, l'utilisateur pourra filtrer les doctorants selon une faculté.

Finalement, le conseiller d'études ou directeur de faculté pourra modifier des suivis de programme par lot. C'est-à-dire qu'il pourra modifier tous les suivis instanciés par

les doctorants d'une faculté en particulier pour par exemple ajouter des ressources ou modifier une tâche.

3. Conception

3.1 Modèle conceptuel

Afin de satisfaire les exigences, des informations sont identifiées dans le système. Tout d'abord, tout utilisateur du système peut être un doctorant, un superviseur, un conseiller aux études ou un directeur. Ils possèdent un nom et un prénom. Quant au doctorant, il possède aussi une faculté une date d'admission. C'est sur cette dernière qu'un programme doctoral s'appuie.

Un programme est composé de plusieurs tâches et étapes. Chaque tâche a une action et une date d'échéance. Par exemple "La manuscrit de thèse doit être rendu au plus tard durant le neuvième semestre". Les étapes sont en fait des groupes de tâches.

Chaque doctorant suit donc son programme. Pour que le superviseur suive l'avancée de ses doctorants, superviseurs et doctorants doivent pouvoir être associés.

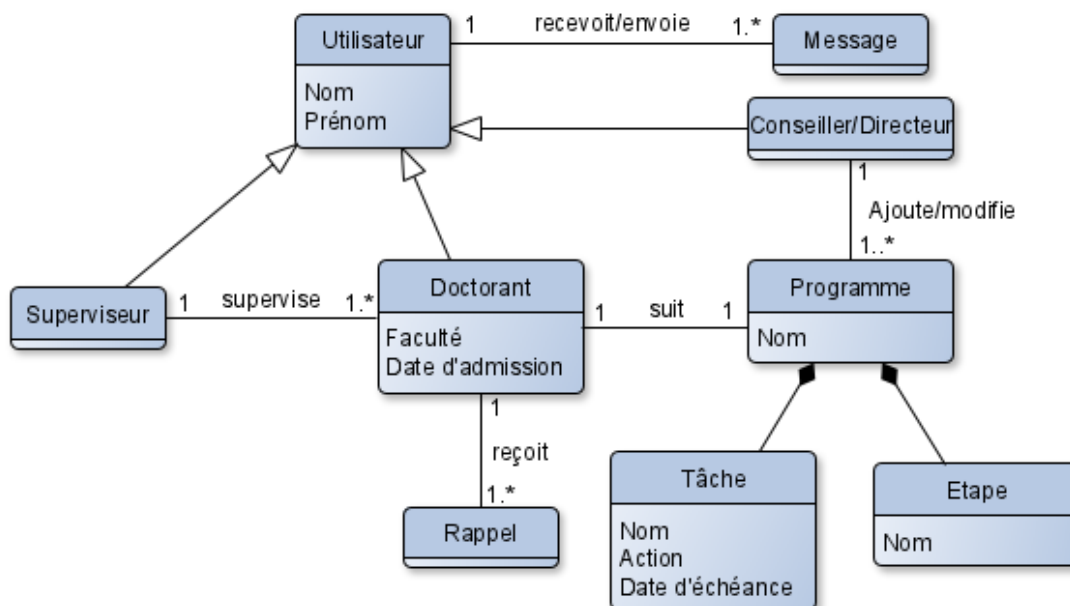


figure 3.a: modèle des concepts

3.2. Modélisation logique de la base de données

En analysant les technologies existantes, nous remarquons que les concepts requis par notre système coïncident avec la base de données de Moodle. Celle-ci

fonctionne par l'intermédiaire de l'API Moodle et contient plus de 250 tables qui doivent être installées dans un système de gestion de bases de données afin de pouvoir bénéficier des fonctionnalités offertes par l'API (p. ex.: consulter l'achèvement d'une activité).

Pour permettre aux doctorants de choisir leurs superviseurs, nous avons créé une table supplémentaire *local_select_supervisor* qui permet de lier les doctorants à leurs superviseurs, ce qui nous a éventuellement permis d'implémenter la vue superviseur (puisque cette vue permet à un superviseur de voir l'avancement de tous les doctorants qu'il supervise).

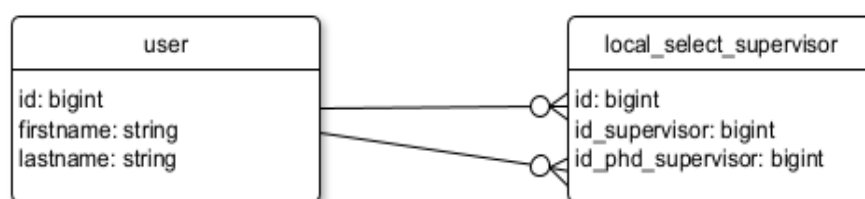


figure 3.b: schéma de la table "local_select_supervisor"

La table *user* comprend plusieurs champs et a été définie par l'API. Moodle offre la possibilité d'ajouter facilement des tables en définissant leur schéma au format XML. En effet, comme moodle supporte plusieurs moteurs de base de données qui ont un format légèrement différent pour certaines de leurs instructions de création de table. Moodle utilise XMLDB, un format standardisé pour définir la structure d'une table. L'éditeur XMLDB de moodle doit être utilisé pour définir la structure de toutes les tables de la base de données de moodle. Le schéma d'une table doit être spécifié dans un fichier *install.xml* situé à l'intérieur d'un dossier *db* afin que le noyau d'application de moodle puisse transformer le fichier *xml* en commande DDL¹ afin de l'ajouter dans la base de données de moodle.

¹ ensemble de commande SQL pour créer, supprimer et modifier la structure d'une table mais pas les données.

3.3. Architecture du système

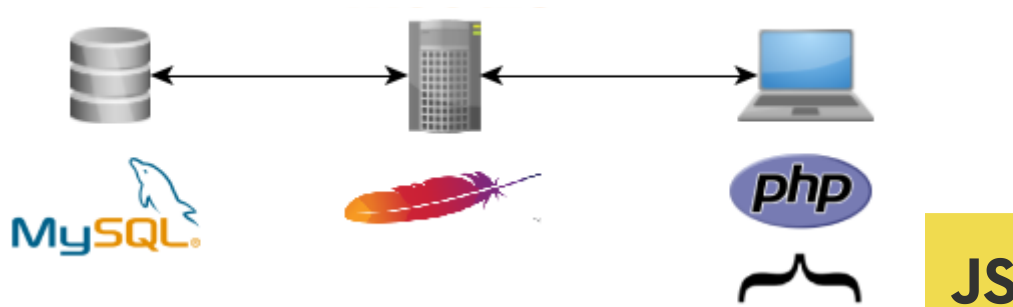


figure 3.c: architecture du système avec les technologies Mysql, Apache, Php, Javascript et Mustache

L'architecture de notre système repose sur l'architecture de Moodle, car nous avons utilisé l'API de Moodle pour développer le service de suivi des doctorants, du fait de son système modulaire.

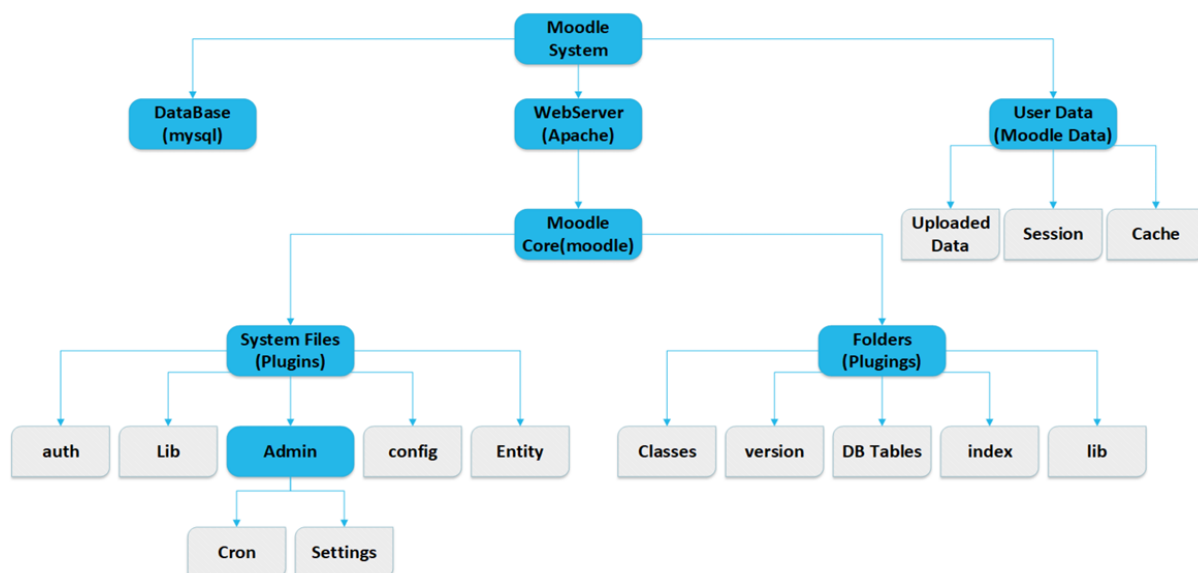


figure 3.d: architecture de moodle, tiré de MOODLE ARCHITECTURE - ABU NASER MD. NAFEW (10/24/2020)

La plupart des fonctionnalités de Moodle avec lesquelles les utilisateurs interagissent directement sont implémentés via des modules indépendants qui sont communément appelées "plugins". Moodle suit le pattern d'architecture en couche (voir la figure 3.d). Les requêtes effectuées par les utilisateurs traversent différentes couches qui interagissent entre elles pour répondre aux requêtes. Moodle est constitué comme un noyau d'application entouré de nombreux plugins qui fournissent des fonctionnalités spécifiques. Le dossier Plugin de moodle est un dossier qui contient tous les plugins qui ont une arborescence bien définies et sont généralement constitués de scripts PHP, CSS, JavaScript, Mustache etc. *Moodle core* (le noyau) fournit toute l'infrastructure nécessaire pour la conception d'un système

de gestion d'apprentissage (learning management system). Il met à disposition des plugins des fonctions et variables globaux qui permettent de mettre en œuvre des cours, des activités, etc. qui seront enregistrés dans la base de données de Moodle.

Un service utilisant l'api de moodle peut être développé sur plusieurs systèmes d'exploitation. Néanmoins, l'installation d'un serveur web tel que Apache est nécessaire pour exécuter des sites internet écrits avec le langage de programmation PHP. En effet, l'api fonctionne avec PHP et un système de base de données tels que MySQL, PostgreSQL, MongoDB, etc..

Pour implémenter notre service nous avons utilisé l'environnement linux, un serveur web Apache, une base de données MySQL et le langage de programmation PHP.

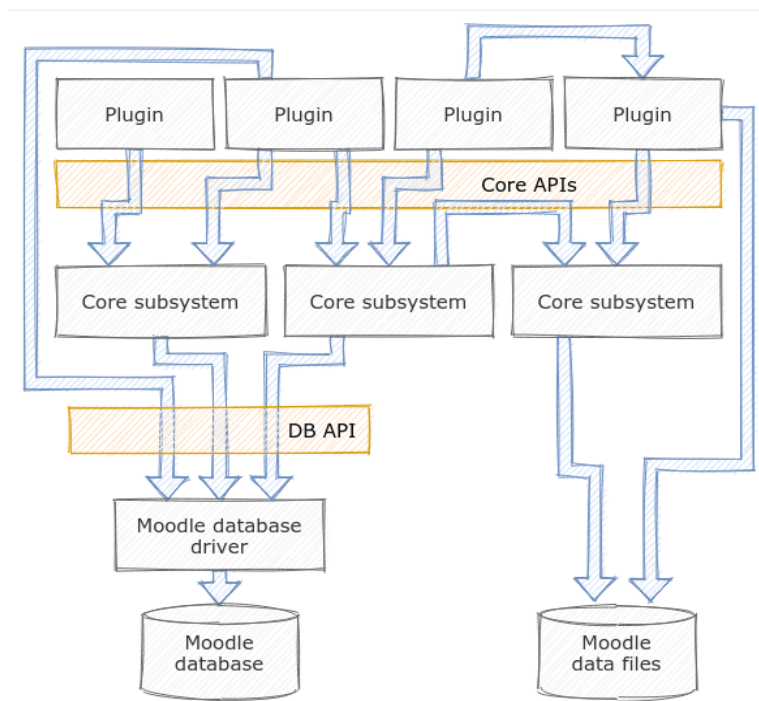


figure 3.e: architecture en couche de Moodle

(<https://moodle.academy/mod/lesson/view.php?id=850&pageid=170>)

3.4. Déploiement

Afin de déployer le service, nous avons créé une image docker. Cette image reprend l'architecture logiciel décrite plus haut. Elle est créée sur la base d'une image lamp dans laquelle nous chargeons le projet moodle. Enfin, nous spécifions des volumes pour pérenniser les données importantes.

Cependant, notre image est trop grande (1,6 Go) pour être déployé sur Portainer. Nous avons tout de même conservé notre image Docker mais avons opté pour une autre solution de déploiement.

Nous avons à disposition un serveur infomaniak auquel nous pouvons accéder par ssh. Nous avons donc utilisé Git afin de déployer l'entièreté de notre branche "main" sur le serveur. Nous avons dû adapter le fichier de configuration Moodle afin qu'il puisse correspondre aux données du serveur.

Notre plateforme est accessible sur le lien <https://unige.the-creator.ch>. Nous avons créé des utilisateurs test afin de représenter le fonctionnement de la plateforme :

Rôle	Nom d'utilisateur	Mot de passe
<i>Doctorant</i>	<i>ashley-caselli</i>	<i>ncC38nvCvRsWLZ&S</i>
<i>Superviseur</i>	<i>giovanna-dimarzo</i>	<i>ncC38nvCvRsWLZ&S</i>
<i>Conseiller/Directeur</i>	<i>marion-perrin</i>	<i>ncC38nvCvRsWLZ&S</i>

figure 3.f: utilisateurs de Moodle

4. Interfaces et test utilisateurs

Pour la partie interface du projet, nous nous sommes concentrés sur la vue des doctorants dans un premier temps puis par la suite sur la vue superviseur et conseiller/directeur.

Nous avons effectué une analyse des utilisateurs par le biais d'un sondage et d'un focus groupe. L'analyse des besoins des utilisations nous a permis d'esquisser une liste des fonctionnalités à implémenter dans l'interface.

Ces fonctionnalités concernent majoritairement les étapes du doctorat, c'est-à-dire soumettre des documents, consulter la liste des étapes, lire la description d'une étape, valider les tâches d'une étape. S'ajoute à cela la possibilité de maintenir des discussions avec le superviseur ou la direction à propos des différentes étapes. Enfin, l'envoi de rappels est également à implémenter.

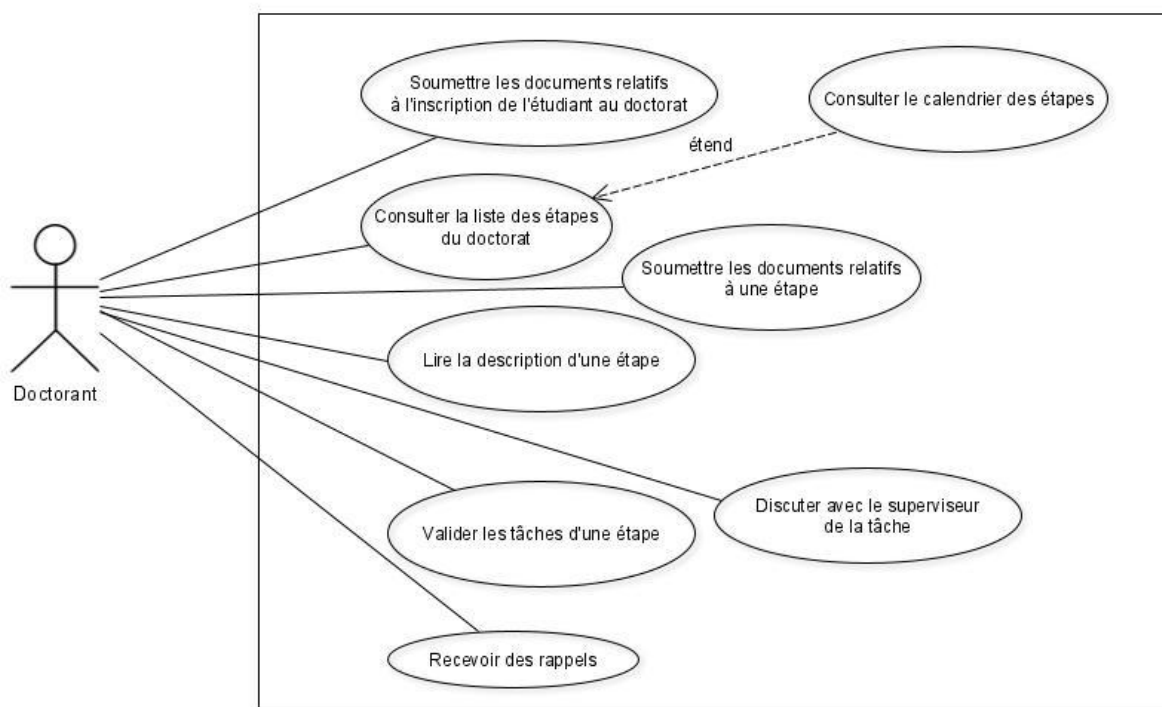


Fig. 4.a : Page de dépôt de documents pour l'inscription au doctorat.

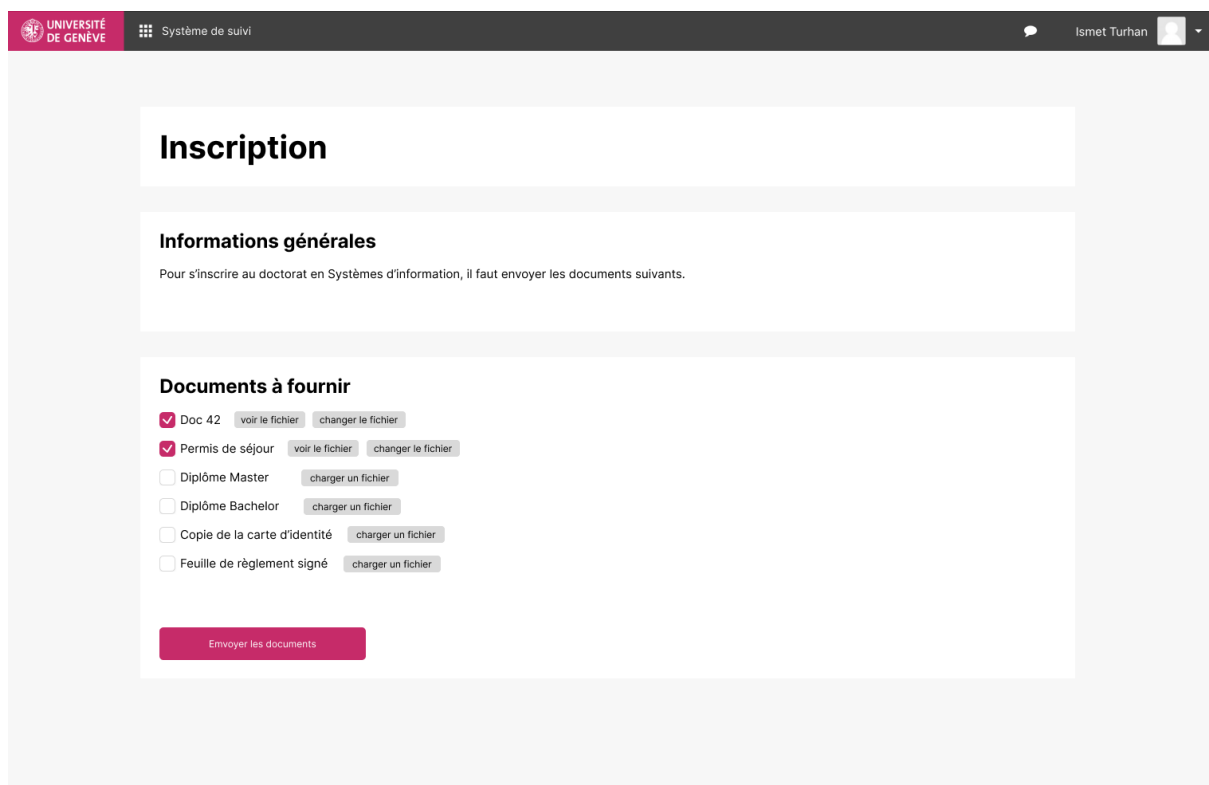
Nous avons pris la décision d'effectuer un prototype interactif avec l'outil Figma.

Afin de garder une cohérence dans le système de service de l'Unige, nous suggérons de reprendre le code couleur, la typographie et la structure général des services Unige tel que Moodle, mediaserver ou portfolio.

La palette de couleur utilisé est donc le gris foncé (#404040), le gris clair (#F7F7F7), le blanc (#FFFFFF) et le rose (#DD0860). La police utilisée est le "Segoe UI".

La structure Unige qui a été gardée sur notre plateforme est la barre d'entête avec sur la gauche le logo de l'Unige et un système de navigation entre les différents services unige. Sur la partie droite de l'en-tête nous retrouvons l'icône d'accès à la messagerie et l'onglet de profil avec l'accès aux informations personnelles.

Comme expliqué précédemment, l'authentification de l'utilisateur se fait avec Switch-edu. Lorsque l'utilisateur accède pour la première fois à la plateforme, celui-ci tombe sur la page de son inscription au doctorat. On lui demande alors de déposer les documents nécessaires (fig.4.b) puis d'appuyer sur "Envoyer les documents". Après avoir envoyé les documents, le doctorant doit attendre la validation de ses documents (fig. 4.c), il ne peut toujours pas accéder à la plateforme.



UNIVERSITÉ DE GENÈVE Système de suivi Ismet Turhan

Inscription

Informations générales

Pour s'inscrire au doctorat en Systèmes d'information, il faut envoyer les documents suivants.

Documents à fournir

- ☒ Doc 42 [voir le fichier](#) [changer le fichier](#)
- ☒ Permis de séjour [voir le fichier](#) [changer le fichier](#)
- ☐ Diplôme Master [charger un fichier](#)
- ☐ Diplôme Bachelor [charger un fichier](#)
- ☐ Copie de la carte d'identité [charger un fichier](#)
- ☐ Feuille de règlement signé [charger un fichier](#)

[Envoyer les documents](#)

Fig. 4.b : Page de dépôt de documents pour l'inscription au doctorat.

Inscription

Informations générales

Pour s'inscrire au doctorat en Systèmes d'information, il faut envoyer les documents suivants.

Documents à fournir

- ☒ Doc 42 [voir le fichier](#) [changer le fichier](#)
- ☒ Permis de séjour [voir le fichier](#) [changer le fichier](#)
- ☐ Diplôme Master [charger un fichier](#)
- ☐ Diplôme Bachelor [charger un fichier](#)
- ☐ Copie de la carte d'identité [charger un fichier](#)
- ☐ Feuille de règlement signé [charger un fichier](#)

[Envoyer les documents](#)

Votre inscription a bien été soumise

Fig. 4.c : Page de dépôt de documents lorsque l'utilisateur appuie sur le bouton "Envoyer les documents"..

Lorsque les documents ont été validés par la direction, le nouveau doctorant peut accéder à la page de ses préférences sur les fonctionnalités (fig. 4.d). Cette page vise à personnaliser le système afin que celui-ci soit adapté au besoin de chacun, comme il a été demandé au focus group. L'utilisateur peut ainsi choisir de visualiser les étapes en fonction d'une échelle de temps par défaut (vue semestriel ou vue sur toute la durée du doctorat), ou personnaliser la fréquence de rappels.

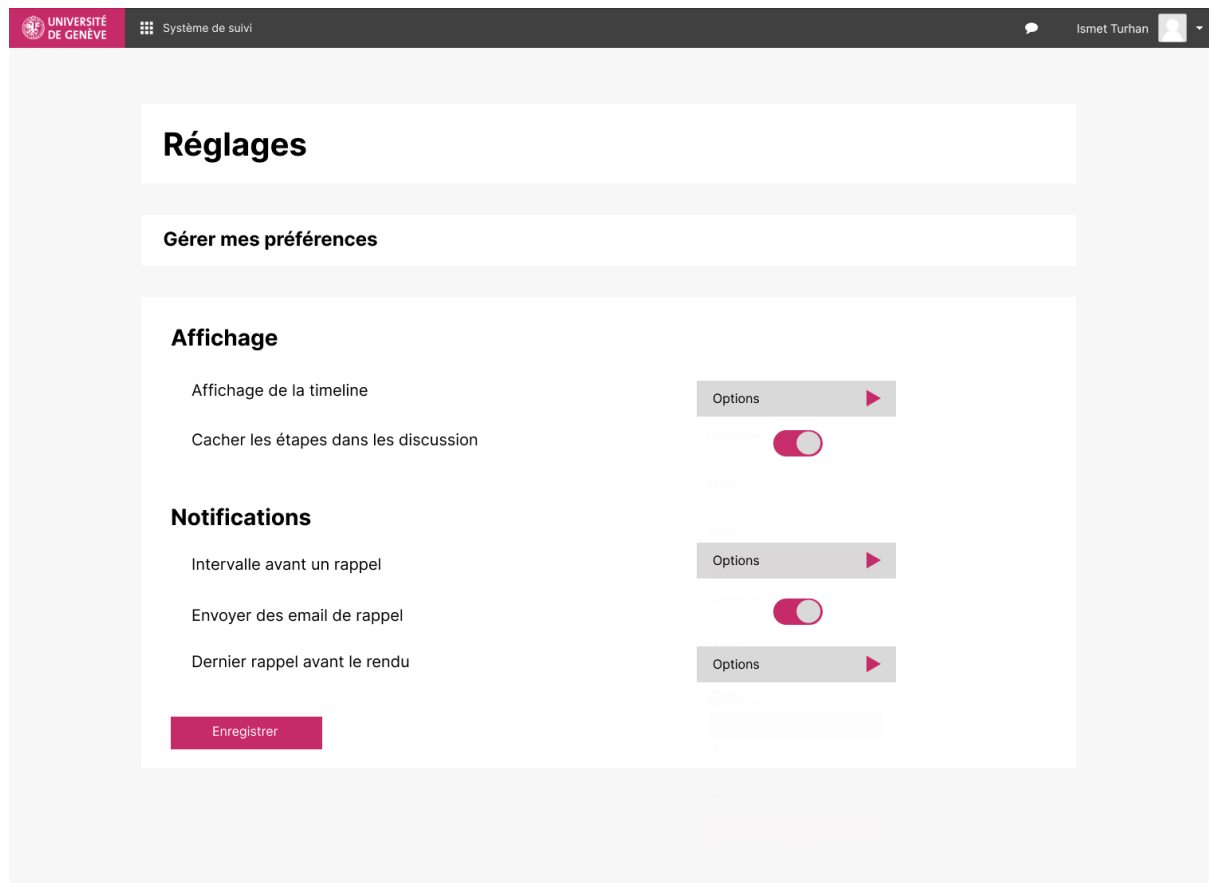


Fig. 4.d : Page de gestion des préférences de l'utilisateur à propos des fonctionnalités dans le système.

Une fois ses préférences enregistrés, l'utilisateur accède à la page de consultation des étapes, qui devient alors la page d'accueil de la plateforme (fig 4.e). La liste des tâches en cours est affichée au centre ainsi que le temps restant. S'ajoute à cela une ligne du temps afin de créer une vue plus globale des étapes restantes, tel qu'il a été souhaité lors du focus group.

UNIVERSITÉ DE GENÈVE

Système de suivi

Ismet Turhan

Système de suivi

[Gérer mes préférences...](#)

Tâche en cours

10 semestre restant
Séminaires scientifiques avancés
 Accomplir 5 journées complètes de séminaire scientifique avancé tel que CUSO.

4 semestre restant
Sujet de thèse
 Le sujet de thèse est présenté oralement et par écrit au comité scientifique du doctorat pour validation.

9 semestre restant
Discussion de thèse
 Les manuscrits de doctorat doivent être soumis au plus tard au cours du 9ème semestre. Dans les 3 mois, le comité de thèse et les doctorants discuteront de la thèse en privé. A l'issue de cette rencontre, le comité de thèse peut demander des modifications.

Semestrielle ▼

Nous sommes le 10 novembre

Semestre actuel

Prochain semestre

Semestre + 2

Semestre + 3

Smestre + 4

Etat de l'art
Choix des cours
Inscription

Progrès annuel

Etape 6

Progrès annuel

Etape 7

Cliquez sur les tâches pour voir le détail

Fig. 4.e : Page principale avec la liste des tâches en cours et la ligne du temps gradué en fonction de l'échelle choisie (ici semestre)

Lorsque l'utilisateur clique sur l'une des étapes situées sur la ligne du temps, la liste des tâches de celle-ci est déployée afin que l'utilisateur puisse valider les tâches (fig 4.f). A noter que certaines tâches ne peuvent pas être validées par le doctorant, comme par exemple "Confirmation par le superviseur".

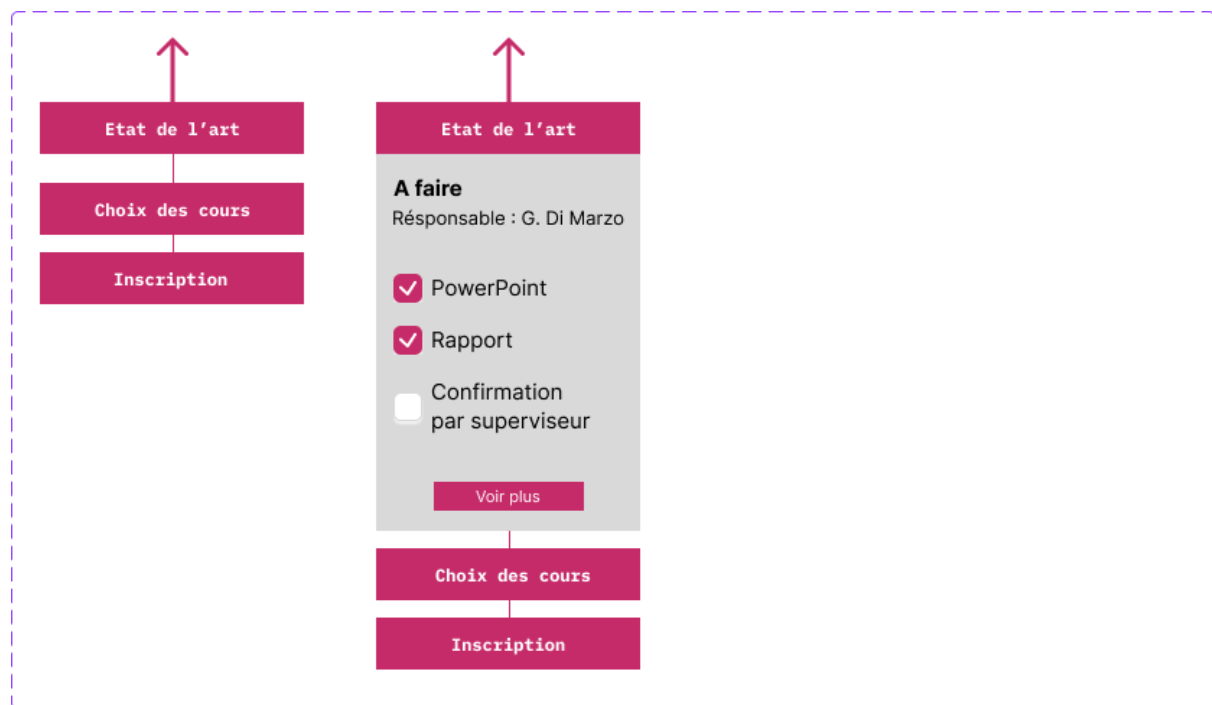


Fig. 4.f : Apparition de la liste des tâches lorsque l'utilisateur appuie sur "Etat de l'art".

Afin de consulter la page d'une étape, il faut cliquer sur "voir plus". Ainsi, l'utilisateur accède à la description de l'étape et à la possibilité de soumettre des documents (fig 4.g).

Etat de l'art

Description
Responsable : G. Di Marzo

L'état de l'art est l'état des connaissances dans tout domaine donné (scientifique, technique, artistique, médical, etc.) à un instant donné. L'état de l'art est parfois nommé "état de la question" et s'appuie sur la synthèse d'une bibliographie exhaustive d'un sujet, commentée par l'auteur.

A faire

- ☒ Revue littéraire
- ☒ Présentation scientifique de trois article scientifique
- ☐ Ecrire un article scientifique sur le sujet de travail
- ☐ Cliquer pour ajouter une nouvelle tâche

Taille maximale des fichiers : 250Mo ; nombre maximal de fichiers : 20

Fichiers

version 1

Plateforme...

Enregistrer Annuler

Fig. 4.g : Page de l'étape "Etat de l'art" avec possibilité de valider les tâches et rendre un document.

A tout moment, l'utilisateur peut accéder au système de chat (fig 4.h). Celui-ci est composé de plusieurs flux de communication dont le sujet dépend de l'étape en cours. Chaque flux est en fait une conversation entre le doctorant et le superviseur.

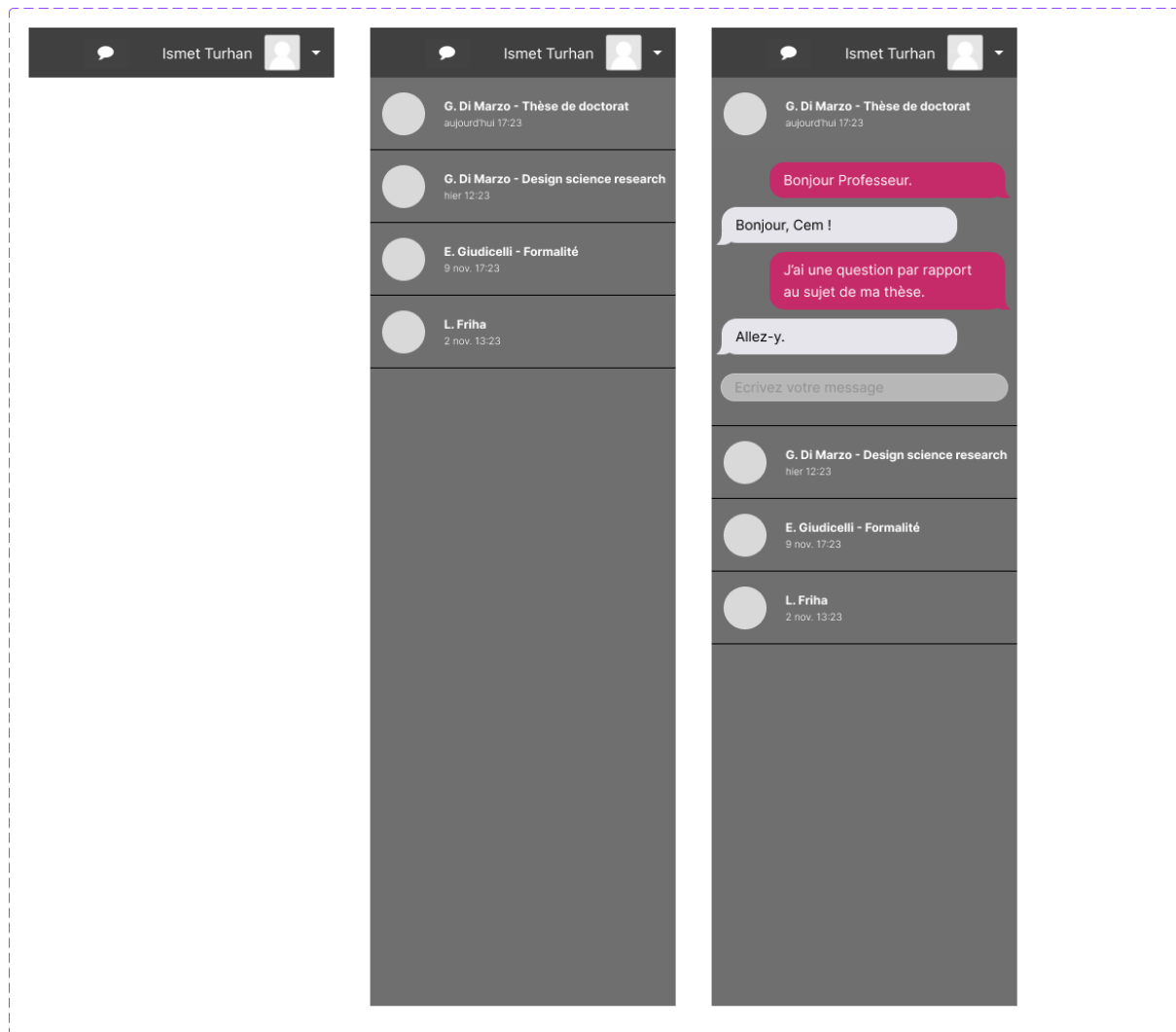


Fig. 4.h : Système de chat constamment atteignable pour l'utilisateur par l'icône de la bulle.

Afin de continuer nos analyses utilisateurs, nous avons, lors du hackathon Hackademia, travailler avec Mme. G. Di Marzo afin de déterminer les informations importantes qui devraient se trouver sur les vues du directeur/conseiller et superviseur. Nous avons déduit qu'il fallait avoir 2 sections distinctes. Une première qui se focalise sur le suivi des doctorants : leurs rendus, leurs délais ou leurs tâches. Une deuxième section qui permettra le suivi administratif de certains documents tel que les documents d'inscription.

Commençons par nous pencher par la partie sur le suivi des doctorants. L'utilisateur est accueilli sur une interface qui reprend toujours les mêmes codes couleurs de l'UniGe et la structure générale est identique à celle de la vue des étudiants/doctorants. Nous retrouvons la même en-tête en haut de la page.

Première section de la page est un rappel du nom du système à gauche et à droit, l'utilisateur à la possibilité de choix que faculté/programme. Ainsi, uniquement les

étudiants de cette faculté sont affichés sur la page. De ce fait, les conseillers pourront accéder rapidement aux étudiants dont ils veulent observer l'avancement. Ensuite, la seconde section de la page propose une vue d'ensemble rapide. Les étudiants sont listés et leur état (retard ou à jour) est indiqué.

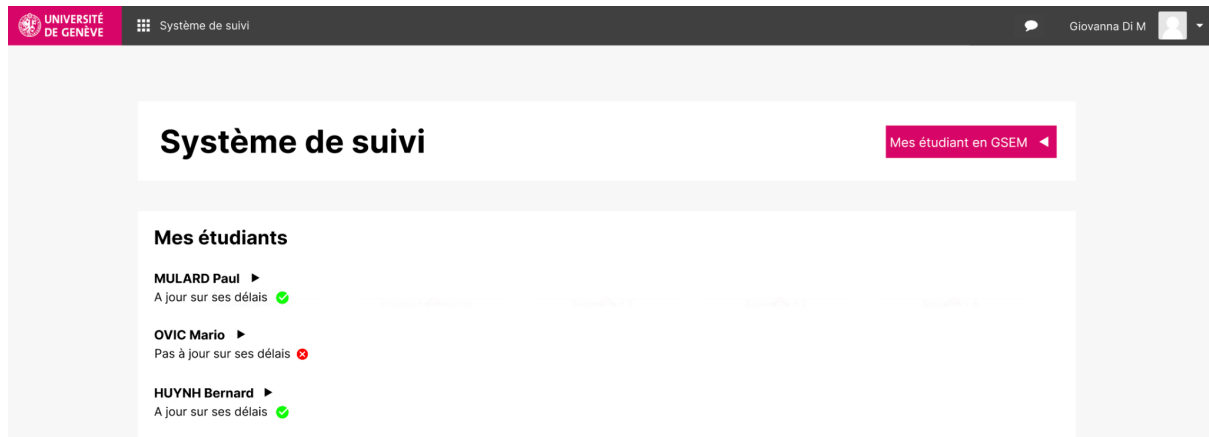


figure 4.i - affichage du suivi des doctorant

Il est possible de cliquer sur chaque étudiant afin d'afficher son avancement détaillée.

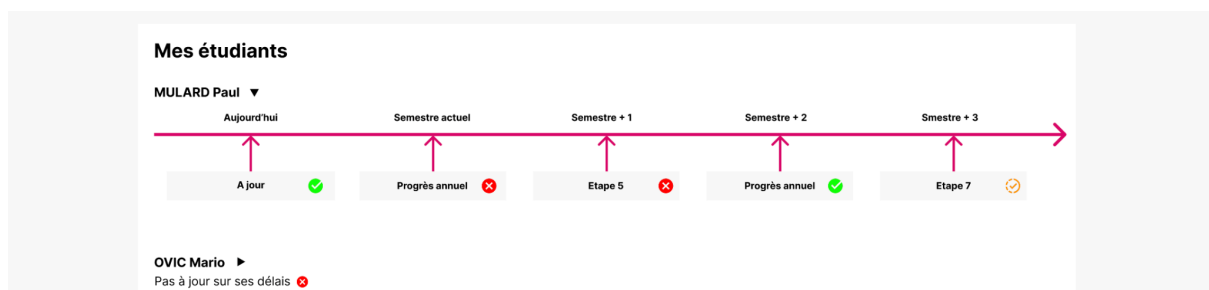


figure 4.j - affichage du suivi des doctorant déplié

Si davantage d'informations sont nécessaires, il est possible d'afficher la fiche détaillée de chaque étudiant en cliquant sur sa timeline.

Cette fiche détaillée informe sur l'avancement détaillé de l'étudiant et également de ses informations personnelles.

Cet affichage est partagé avec la vue superviseur.

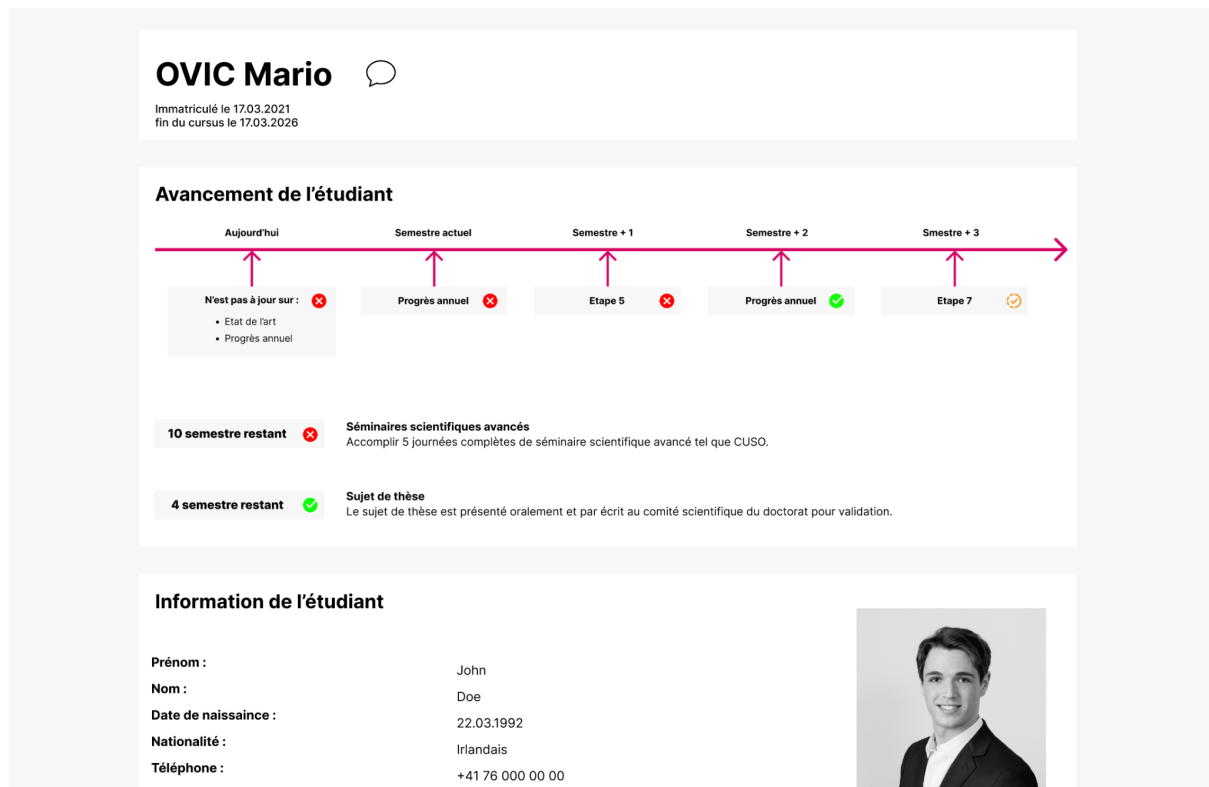


figure 4.k - affichage du suivi des doctorant détaillé

Continuons par la deuxième fonctionnalité importante évoquée : la gestion administrative. L'utilisateur est à nouveau accueilli sur une page reposant sur la même structure avec les mêmes couleurs.

La première section est le titre. La deuxième section présente l'état d'avancement des dossiers des étudiants voulant s'inscrire à un doctorat. Chaque étudiant possède une section à lui séparé par des lignes afin de distinguer visuellement chaque personne. L'avancement d'un dossier est représenté sous forme d'une ligne avec des points à chaque personne qui doit valider le dossier. La couleur verte apparaît comme une barre de chargement, si la ligne est entièrement verte, elle est validée sinon elle est cours de validation.

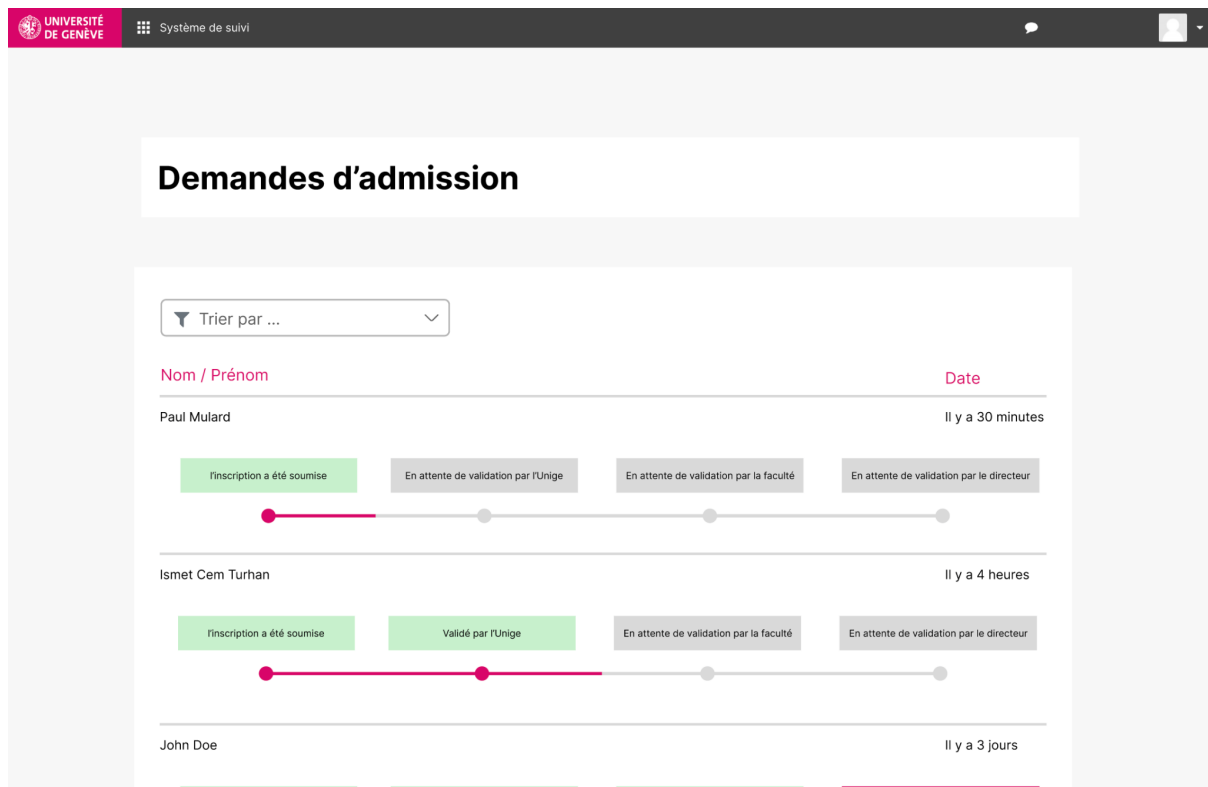


figure 4.l - affichage du suivi des demandes d'admission

Dès que la responsabilité de valider un dossier est chez l'utilisateur connecté, son point se transforme en couleur rose et ce point permet de valider un dossier.

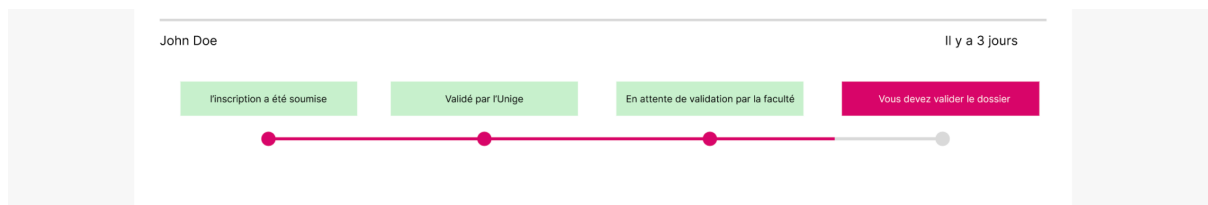





figure 4.m - affichage du suivi des demandes d'admission (une demande)

En cliquant sur chaque dossier, il est possible d'avoir une vue détaillée.

La barre de chargement est toujours affichée en haut de la page. En dessous, les informations personnelles de la personne sont affichées. Ensuite, un système de gestion des documents est présent. L'utilisateur a la possibilité de valider chaque document un par un ou tout valider en un clic. Ceci afin de refuser un document qui serait incomplet.


UNIVERSITÉ DE GENÈVE


Système de suivi


Giovanna Di M

Demande d'inscription pour le doctorat GSEM SI

l'inscription a été soumise

Validé par l'Unige

En attente de validation par la faculté

Vous devez valider le dossier

Valider le dossier

Information général

Prénom :

John

Nom :

Doe

Date de naissance :

22.03.1992

Nationalité :

Irlandais

Téléphone :


+41 76 000 00 00

Email :

john.doe@outlook.com

Address :

Bvd Carl-Vogt 35, 1205 Genève



Documents soumis

Nom de document	Date	Non lu/Lu	Status	Commentaire	Documents
CV	01.08.2022	Lu	Accepté		pdf2.3.pdf
Lettre de motivation	04.08.2022	Non lu	Valider le document		Test21.pdf

figure 4.n - affichage du suivi des demandes d'admission un doctorant détaillé

Le prototype interactif est accessible sur Figma avec le lien suivant : <https://www.figma.com/file/ZM5KETA9mV0lJuzLqnTzAI/PT2---doctorant?node-id=0%3A1>

Nous avons terminé notre phase de conception d'interface par effectuer des tests utilisateurs. Il est important de s'assurer que les interfaces sont conviviales et intuitives pour les utilisateurs, afin de garantir une expérience positive pour ceux qui utilisent le service. Pour évaluer l'efficacité de l'interface, nous avons effectué des tests utilisateurs de manière informelle sur les interfaces interactives, qui nous permettront de mesurer la satisfaction des utilisateurs avec le design et les fonctionnalités proposées. Nous avons montré et fait utiliser nos interfaces lors du Hackathon pour la vue conseiller/directeur et pour la vue doctorant à des utilisateurs type.

Les tests ont été très convaincants. La structure et la présentation des informations ont été appréciées par les utilisateurs.

5. Implémentation des plugins

5.1. Plugin “Study tracker”

La gestion des différentes vues se fait avec un plugin général.

Ce plugin implémente les fonctionnalités de création de programme personnalisable et de suivi des tâches pour les doctorants, les superviseurs, les conseillers d'études ainsi que les directeurs de faculté.

Il a été conçu avec l'API mise à disposition par moodle et selon les bonnes pratiques dictées par les développeurs moodle. C'est pourquoi le plugin est compatible avec un serveur quelconque dans lequel est installée une instance de moodle.

Création automatique des étapes et des tâches à partir d'un fichier XML

La première fonctionnalité du plugin est la création d'un programme personnalisable pour un doctorant et ses superviseurs.

Suite au hackathon, nous avons décidé de représenter les programmes doctorales avec le *Document type definition*, DTD, décrivant la structure d'un fichier XML, ses éléments et leurs attributs. En simplifié, la DTD spécifie qu'un programme est composé de tâches qui sont liées à des étapes (regroupement de tâches). Chaque tâche est composée, à son tour, d'une action et d'un délai. Les actions sont faites par une personne, elles ont un sujet d'action et un objet d'action. Par exemple, la tâche “Sujet de thèse”, peut appartenir à l'étape d'écriture de la thèse, et est composée d'un sujet d'action, *submit*, et d'un objet d'action, le sujet de thèse.

N'importe quel programme décrit selon les contraintes de la DTD est utilisable par le plugin. En effet, nous nous basons sur cette structure (définie dans la DTD) pour créer les programmes des étudiants.

Les différentes étapes de l'algorithme sont premièrement lire le fichier XML représentant le programme deuxièmement instancier un objet de la classe program puis finalement de sauvegarder le programme et le rendre utilisable par moodle en créant un “cours” moodle, composés de “modules” et de “sections”.

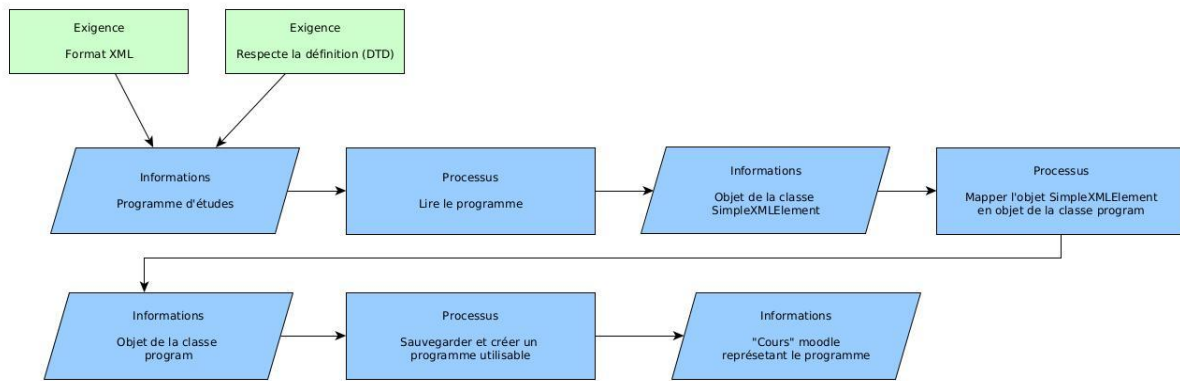


figure 5.a - processus de création d'un programme

Les transformations successives passent d'un objet *SimpleXMLElement* représentant un fichier xml de façon récursive à un objet intermédiaire *program*, composé d'objets *task* et *step*, pour finalement transformer le tout en un "cours" moodle composé de modules et de sections.

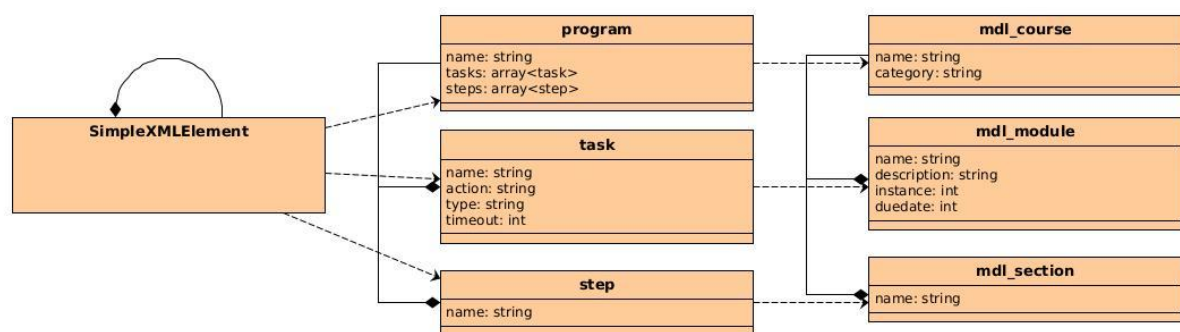


figure 5.b - séquence de transformations

PS: le "cours" moodle est le nom donné dans la documentation de Moodle pour désigner toute leçon, cours, ou séminaire dans lequel l'utilisateur peut s'inscrire. Dans notre cas, cet objet représente le programme du doctorant et son inscription est automatique.

Grâce à cette création automatique d'un programme à partir de sa représentation XML, nous avons un système générique dans lequel il est facile d'ajouter une nouvelle version d'un programme doctorale ou un nouveau programme. Pour cela, il suffit d'ajouter le texte de loi au format XML et selon la DTD. De plus, à partir du moment où le programme est créé pour un doctorant donné, le programme n'est plus dépendant du texte de loi. Les programmes peuvent donc être modifiés par exemple pour modifier des délais ou ajouter des ressources.

Ce programme ou suivi de programme sera l'élément central autour duquel les différents utilisateurs du système se concentrent. En effet, ils pourront tout au long du

cursus de l'étudiant s'y référer pour savoir ce qui a été fait et ce qu'il reste à faire par le doctorant.

Algorithmes liés au calcul de semestres

Après la lecture de plusieurs programmes d'étude, nous remarquons rapidement que la date d'échéance de la plupart des tâches à accomplir pour un étudiant dépend du nombre de semestres passés depuis sa date d'entrée au doctorat. Cependant, le système de semestres diffère selon les universités dans le monde et n'est pas intégré à Moodle. De plus, nous n'avons eu connaissance d'aucune API ou règlement fournissant des informations sur le système de semestres à l'Unige. Toutefois, il est nécessaire de définir exactement les dates usuelles de début et de fin d'un semestre, afin que le système puisse attribuer une date précise d'échéance, au jour près, à chaque tâche.

Nous nous sommes donc mis d'accord sur la conception d'un système de semestres qui coïncide approximativement avec les dates de rentrée à l'Unige. En ayant pris connaissance des calendriers académiques issus des années précédentes, nous supposons la règle suivante:

"Le semestre d'automne commence le deuxième lundi de septembre. Quant au semestre de printemps, il commence le troisième lundi de février. "

A l'aide de cette règle, nous pouvons déterminer avec précision le nombre de semestres entre une date A et une date B. Ce calcul est par exemple utilisé pour déterminer le temps restant à l'utilisateur pour accomplir une tâche.

En parallèle, nous pouvons également déterminer une date B, en fonction d'une date A et un nombre de semestres N. Ce calcul est par exemple utilisé pour déterminer, conformément au règlement, une date d'échéance précise à l'utilisateur pour accomplir une tâche.

Calcul du nombre de semestres entre deux dates

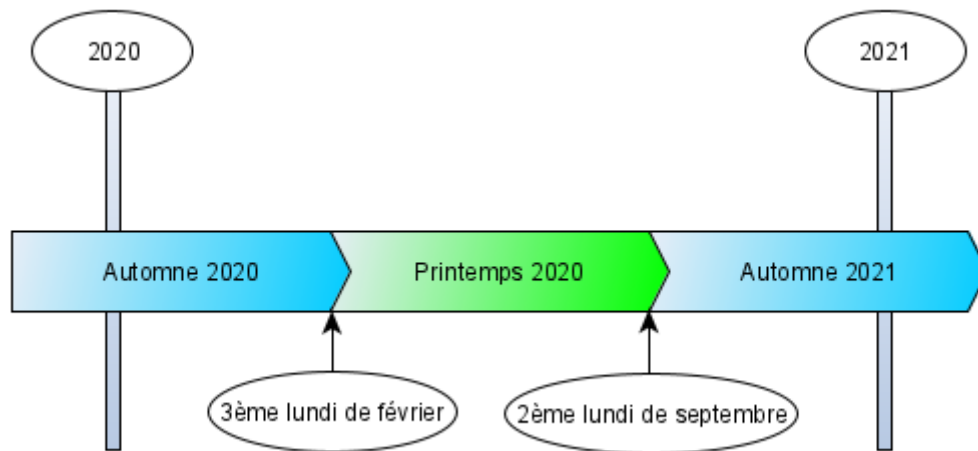


figure 5.c - semestres au cours de l'année 2020

Soit D , une date. Nous pouvons récupérer son année, noté Y , un nombre entier naturel, ainsi que sa période, noté S .

Au cours d'une année Y , il y a 3 périodes possibles, si l'on omet les vacances d'été (la règle des périodes):

- le semestre d'automne de l'année Y allant du 1er janvier jusqu'au second dimanche du mois de février,
- le semestre de printemps de l'année Y allant du premier lundi du mois de février jusqu'au premier dimanche du mois de septembre,
- et enfin le semestre d'automne de l'année $Y + 1$ allant du second lundi du mois de septembre jusqu'au 31 décembre.

Nous définissons une valeur à S , 0 si D se trouve dans la première période de l'année, 1 si D se trouve dans la deuxième période ou 2 si D se trouve dans la dernière période, comme suit:

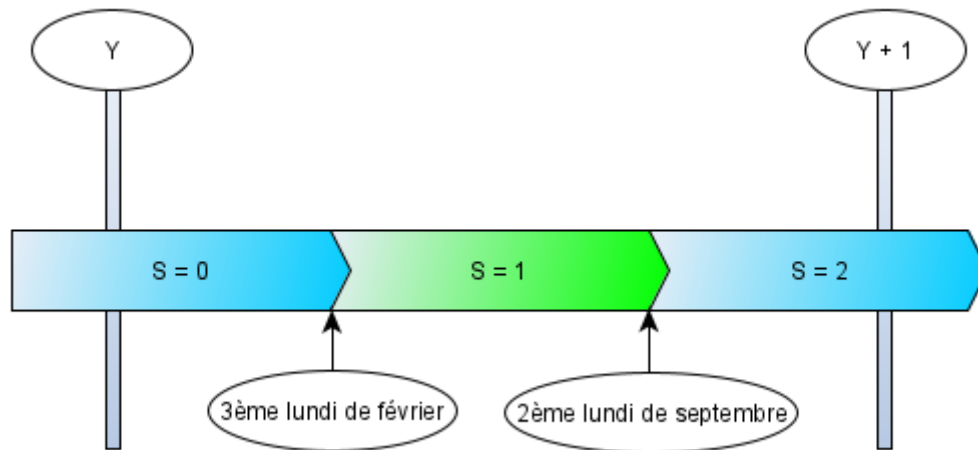


figure 5.d - semestres au cours d'une année Y

Une fois les valeurs Y et S connus pour une date A et une date B, nous pouvons déterminer l'équation suivante:

$$N = 2 * (Y_b - Y_a) + S_b - S_a$$

, où N est le nombre de semestres entre les deux dates.

Prenons par exemple une date A, le 14 janvier 2020. Selon la règle des périodes, S_a vaut 0, car la date se trouve entre le 1er janvier et le deuxième dimanche de février. Nous savons également que Y_a vaut 2020.

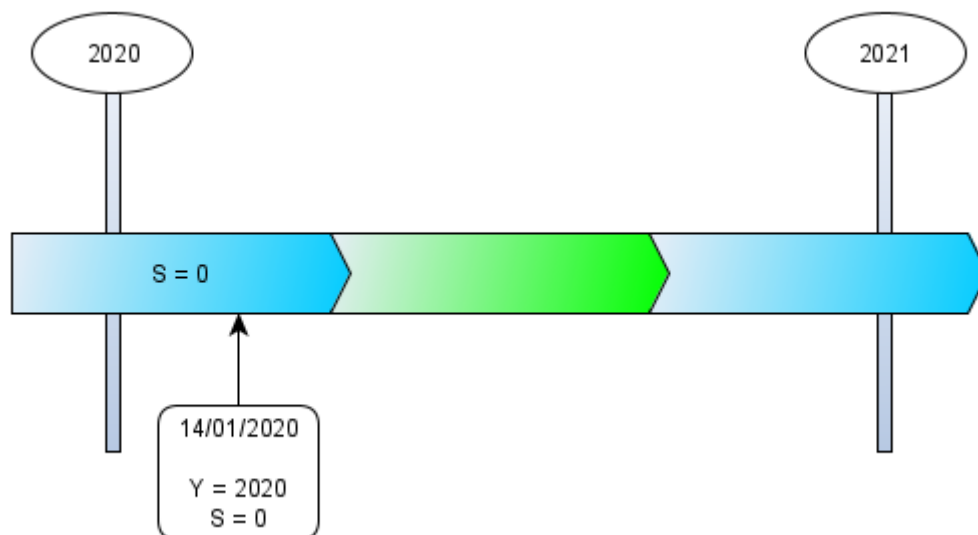


figure 5.e - exemple de date A

Prenons ensuite une date B, le 28 novembre 2021. Selon la règle des périodes, S_b vaut 2, car la date se trouve entre le 1er janvier et le deuxième dimanche de février. Nous savons également que Y_b vaut 2021.

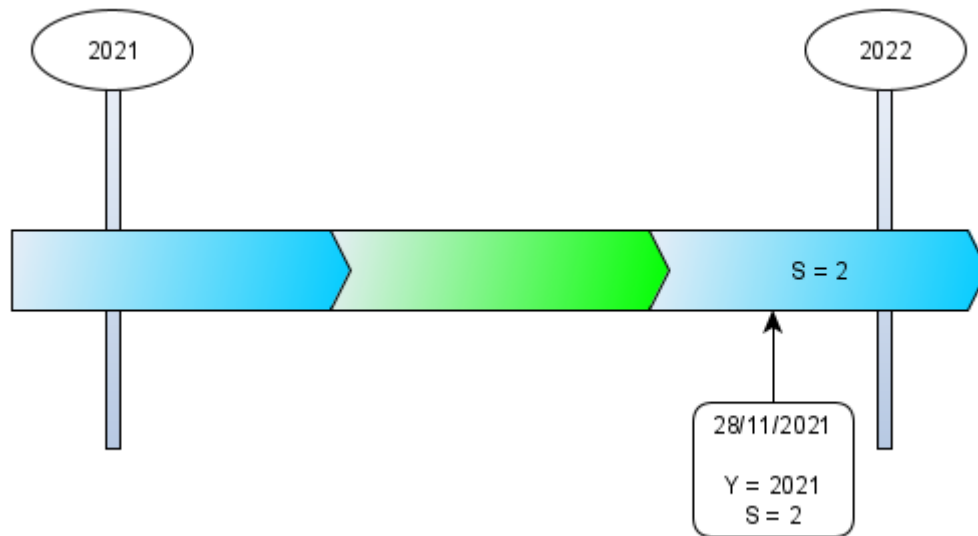
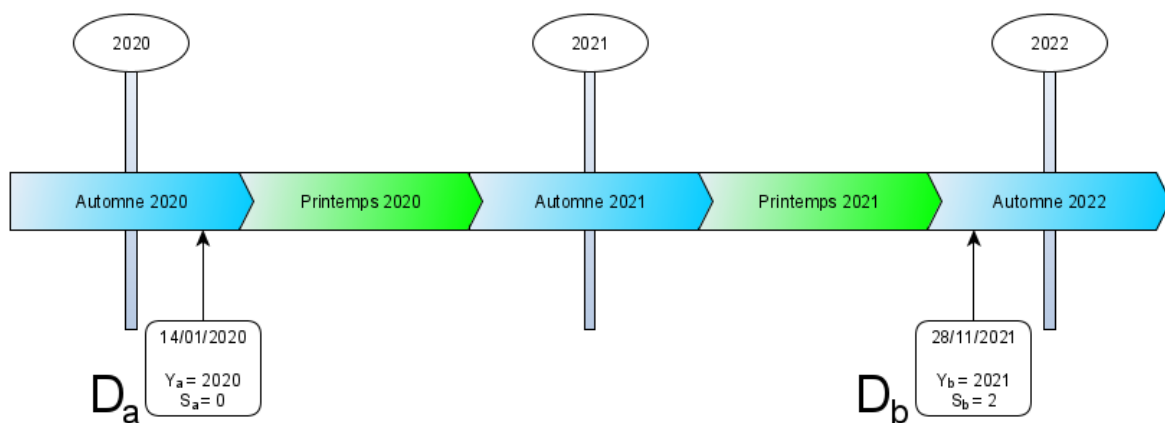


figure 5.f - exemple de date B

Nous pouvons donc calculer N :



$$\begin{aligned}
 N &= 2 * (Y_b - Y_a) + S_b - S_a \\
 &= 2 * (2021 - 2020) + 2 - 0 \\
 &= 4
 \end{aligned}$$

figure 5.g - calcul de N

Nous pouvons vérifier le calcul en comptant les semestres manuellement:

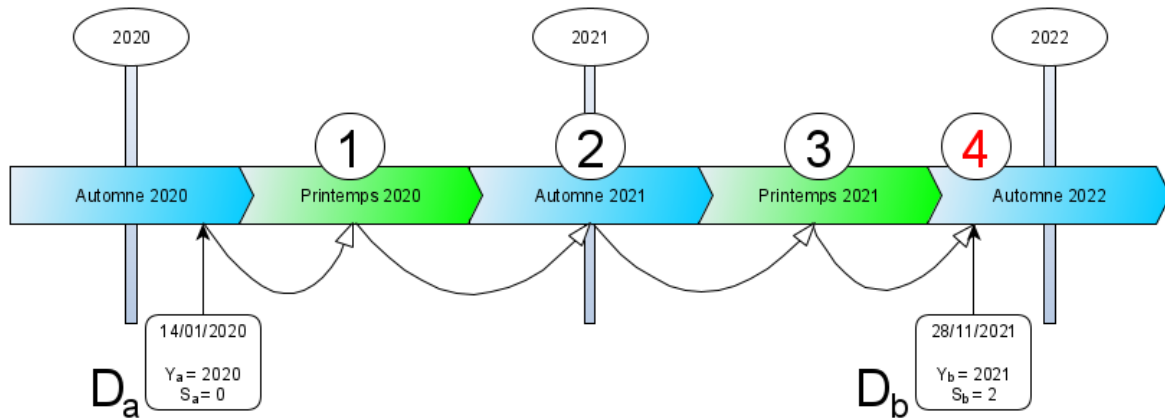


figure 5.h - calcul à la main

Nous avons donc pu calculer le nombre de semestres entre deux dates, en prenant en compte leur année, leur mois et leur jour. Dans notre cas, la date A pourrait être la date actuelle de l'utilisateur et la date B serait la date d'échéance d'une tâche. Le nombre de semestres ainsi calculé est utilisé pour être affiché dans la vue du doctorant afin que celui-ci puisse s'organiser.

Algorithme d'ajout de semestres à une date

Selon le règlement, certains délais pour accomplir une tâche sont définis en fonction d'un nombre de semestres. Par exemple,

"PhD manuscripts should be submitted at the latest during the 9th semester"

Il a fallu donc trouver un autre algorithme pour ajouter 9 semestres à la date d'admission de l'étudiant, afin de déterminer la date d'échéance de la tâche.

En reprenant le calcul du nombre de semestres entre deux dates, nous pouvons isoler Y_b :

$$N = 2 * (Y_b - Y_a) + S_b - S_a$$

$$Y_b = \frac{N + S_a - S_b}{2} + Y_a$$

Nous savons que S_b peut être soit un nombre pair, 0 ou 2, soit un nombre impair, 1. Si nous connaissons N , Y_a et S_a , nous pouvons déterminer l'algorithme suivant:

AjouterSemestre(Y_a , S_a , N):

Si $N + S_a$ est un nombre pair:

$S_b = 0$

Sinon:

$S_b = 1$

$Y_b = \frac{N + S_a - S_b}{2} + Y_a$

Retourner Y_b , S_b

Une fois avoir calculé Y_b et S_b , nous pouvons déterminer une date. Pour rappel, le calcul est utilisé pour retourner la date d'échéance de certaines tâches en fonction de la date d'admission. A l'aide de la règle des périodes, nous transformons Y_b et S_b en une date qui est le dernier jour du semestre en question. Par exemple, si l'on ajoute 2 semestres à la date du 4 mars 2020, le résultat est le dernier jour du semestre de printemps 2021.

La conception d'algorithmes de calcul du nombre de semestres entre deux dates et d'ajout d'un nombre de semestres à une date contribue à l'affichage des différentes vues du plugin.

Vue du doctorant

La vue du doctorant comprend, sous différentes formes, l'affichage des tâches créées par le plugin.

Tout d'abord, les informations tirées du programme lié au doctorant telles que le nom, la description et la date d'échéance de la tâche sont récupérées et stockées dans un objet *template*. Cet objet permet au moteur de template *Mustache* d'afficher correctement les données.

Dans une optique d'aider le doctorant à s'organiser dans le temps, une frise chronologique sur laquelle sont classées les tâches selon leur délai de rendu est d'abord représentée. En effet, des groupes de tâches ayant en commun soit le mois soit le semestre de leur échéance sont premièrement formés. Le choix de l'unité du temps pour ces groupes est à la préférence de l'utilisateur. Puis, ils sont affichés

dans l'ordre chronologique sur une ligne, dite *timeline*. Celle-ci affiche toutefois les semestres/mois sur lesquels ne figure aucune échéance, afin de représenter les périodes libres. En plus de cela, L'état de chacune des tâches, qui peut être non-validée, validée, ou arrivée à échéance, est également affiché, avec une couleur représentative. Lorsque l'utilisateur clique sur le nom d'une des tâches, il est redirigé à la page de la tâche.

Pour éviter un grand nombre de redirections, une forme plus informative des tâches est également figurée. Il s'agit d'une liste verticale de toutes les tâches, par ordre des priorités. Sont considérés comme prioritaires les tâches dont le délai est arrivé à échéance. Cette fois-ci, la description de chaque tâche est affichée sous son intitulé.

Grâce à cette vue, le doctorant peut rapidement et facilement visualiser son avancement dans le programme doctoral de manière à ce que celui-ci puisse s'organiser dans le temps.

Vue du superviseur

La vue du superviseur est un regroupement de plusieurs timelines concernant les doctorants qu'il supervise.

Pour cela, nous récupérons l'id des doctorants avec la table qui associe un superviseur à ses doctorants. Enfin, la création de la timeline est faite de manière semblable que pour celle de la vue du doctorant.

Dans la page de la tâche à laquelle on accède lorsque l'on clique sur un des éléments de la timeline, le superviseur a la possibilité de modifier la date d'échéance de celle-ci. Cette fonctionnalité est une des paramétrisations déjà prévues et intégrées par Moodle. Elle est rendue possible par le rôle du superviseur dans le cours Moodle représentant le programme, c'est-à-dire *professeur*, tandis que les doctorants ont le rôle d'*étudiant*.

Grâce à cette vue centralisée, le superviseur peut rapidement et facilement visualiser l'avancement de chacun de ses doctorants dans le programme doctoral.

Vue du conseiller aux études et du directeur

La vue du conseiller aux études et du directeur est semblable à la vue du superviseur, à la différence près que celle-ci possède une fonctionnalité supplémentaire.

Comme nous l'avons vu, l'idée est de permettre à ces parties prenantes de consulter les doctorants de plusieurs facultés différentes. Un système de filtre par faculté est

alors mis en place pour afficher les étudiants soit d'une faculté en particulier, soit de toutes les facultés. L'utilisation de Javascript permet d'avoir une page interactive sans besoin de rafraîchir la page pour utiliser le système de filtre.

Grâce à cette vue centralisée, le conseiller aux études et le directeur peuvent visualiser l'avancement de chacun des doctorants dans le programme doctoral pour chacune des facultés dont ceux-ci font partie.

5.2. Plugin Theme

Pour l'aspect visuel de notre système, nous avons dû créer un plugin afin de modifier le thème déjà en place sur Moodle. Nous avons appliqué les remarques des tests utilisateur sur le prototype Figma directement à notre interface final par souci de temps.

Le plugin réécrit sur les fichiers de Moodle afin de modifier l'aspect visuel déjà en place. Il est possible de s'appuyer sur un thème préexistant afin de créer son extension ou de créer son propre thème de zéro. Nous avons choisi l'option de s'appuyer sur un thème préexistant assez populaire nommé Boost afin d'éviter les problèmes liés à des manques de données. Il est important de préciser que notre plugin réécrit une très grande partie du thème déjà en place.

Le plugin reprend les couleurs Unige avec le rose(#DD0860), un gris foncé(#262626) et le blanc pale(#FEFEFE). Ce style est repris dans l'entièreté de la plateforme. La page de visualisation des tâches, la page administration du site et celle d'accueil reprennent ces couleurs.

6. Tests du système

6.1. Tests unitaires

Les tests unitaires ont été effectués à l'aide de *PHPUnit*, un framework PHP permettant de faire des tests unitaires avancées. Ce framework est installé avec la dépendance *Composer*, indépendamment de l'installation de Moodle.

Des tests ont été exécutés sur les aspects logiques de notre service. Concernant la création automatique des étapes et tâches de doctorat à partir d'un fichier XML, deux tests ont été faits. Puis, concernant les algorithmes de la timeline que nous avons créés, quatre tests différents ont été effectués.

Tests sur la création automatique des étapes et tâches de doctorat à partir d'un fichier XML

Un premier test consiste à vérifier si les informations d'un fichier XML sont correctement récupérées et stockées dans une classe intitulée *Programme*, comprenant des variables d'instances, tels que le nom, une liste de tâches et une liste des étapes, chacune des listes comprenant les bons paramètres. Afin que ce test fonctionne même lorsque les règlements d'études ont été modifiés avec le temps, un fichier test XML, ayant la même DTD (et donc la même structure) que les autres fichiers XML, a été créé.

Un second test consiste à vérifier si les dates d'échéance de chaque tâche du fichier XML sont correctement traitées. Les dates d'échéance sont traitées avec les dates relatives, c'est-à-dire que les dates d'échéance sont calculées en fonction de la date d'admission, ou de la date d'échéance de la tâche qui la précède si elle en a un. Dans ce test, on ne considère que les dates relatives au moment présent. Par exemple, si le fichier XML comprend dans l'élément de temps comme valeur 5 et l'attribut *timescale* "année", la valeur attendu du test est "+3 ans", i.e. la date relative à la date d'aujourd'hui qui vient dans 3 ans. Si le fichier XML ne comprend pas d'attributs, on suppose qu'on parle de semestres et on s'attend pour, par exemple, une valeur de l'élément égal à 8, la valeur "+48 mois".

Tests sur les algorithmes de la timeline

Dans un premier test des algorithmes de la timeline, nous vérifions si une date correspond à la bonne période. En effet, nous avons, pour rappel, défini trois périodes dans une année: le semestre d'automne allant du 1er janvier jusqu'au

second dimanche du mois de février, le semestre de printemps allant du premier lundi du mois de février jusqu'au premier dimanche du mois de septembre, et enfin le semestre d'automne allant du second lundi du mois de septembre jusqu'au 31 décembre. Trois assertions ont été exécutées pour trois dates dans les trois périodes différentes. Par exemple, la date du 22 janvier doit être dans la première période.

Dans un second test, nous testons le nombre de semestres qui sépare deux dates. Dans ce test, nous ne prenons pas en compte les années entre les deux dates, uniquement les périodes dans lesquelles se trouvent ces dates. 7 assertions ont été exécutées, avec des combinaisons ordonnées différentes de dates dans chaque période. Par exemple, la différence entre le 22 janvier et le 15 septembre doit être de 2, car 2 semestres séparent les deux dates.

Le troisième test est similaire au précédent, sauf que nous prenons cette fois-ci en compte les années qui séparent les deux dates. Nous avons effectué 48 assertions. Plus exactement, nous avons choisi 3 dates se trouvant dans trois périodes différentes comme date de départ pour lesquelles nous avons effectué 16 assertions, les dates d'arrivées correspondant aux périodes qui succèdent celle de la date de départ. Par exemple, la différence entre le 20 mars 2022 et le 14 février 2025 doit être de 5, car 5 semestres séparent les deux dates.

Finalement, le dernier test consiste à vérifier les dates d'échéance à partir de la date d'admission et le semestre d'étude calculé par rapport à cette date d'admission. Pour rappel, nous considérons que le dernier délai pour rendre un travail associé à un semestre correspond à la dernière date du semestre: pour un semestre d'automne, c'est le deuxième dimanche du mois de février et pour un semestre de printemps, c'est le premier dimanche du mois de septembre. Nous avons effectué 33 assertions. Tout comme pour le test précédent, nous avons choisi 3 dates se trouvant dans trois périodes différentes comme date de départ pour lesquelles nous avons effectué 11 assertions, correspondant aux semestres de 0 à 10, succédant celle de la date de départ. Par exemple, si un étudiant a été admis le 18 octobre 2018 et qu'il doit rendre un travail durant le 7ème semestre (la valeur étant de 6, car nous commençons depuis 0), le date d'échéance est fixé au deuxième dimanche du mois de février 2022, qui est la dernière date du semestre d'automne 2021.

6.2. Tests fonctionnels

Des tests fonctionnels ont également été effectués. Les tests ont été écrits en Python et font appel à l'API *Selenium*, un framework de tests informatiques. L'API fourni par *geckodriver* a également été utilisé pour communiquer avec les

navigateurs *Gecko*, en particulier, *Firefox*. Il existe, certes, une intégration de Selenium avec PHP, mais pour les tests fonctionnels, Python a été favorisé pour sa facilité d'intégration du framework Selenium. Les tests ont été effectués sur la version déployée du service.

En ce qui concerne les tests eux-mêmes, 3 séries de tests ont été faites pour chacune des trois vues: la vue doctorant, la vue superviseur et la vue conseiller aux études. Dans chacun des trois tests, des données de test ont été utilisées pour obtenir les privilèges de chaque type d'utilisateur. La première étape pour chaque test consiste donc à s'identifier avec le nom d'utilisateur et le mot de passe. Pour ces tests, nous reprenons les identifiants d'un étudiant fictif de la GSEM. Une fois connecté, une première assertion permet de vérifier si le nom de l'utilisateur apparaît sur la page principale, qui est la page des programmes du doctorant.

Tests sur la vue du doctorant

Pour la vue du doctorant, la page principale comprend logiquement un seul programme doctoral. Nous faisons donc une assertion pour vérifier qu'un doctorant n'a bien qu'un seul programme doctoral. On clique ensuite sur cet unique élément et nous faisons une autre assertion pour vérifier si le nom du doctorant apparaît dans l'en-tête de la page, et donc qu'on est bien sur sa page. On vérifie ensuite que, dans notre exemple, les étapes, et les tâches qui en sont associées, du programme doctoral en Systèmes d'Information de la GSEM apparaissent dans le bon ordre. On teste ensuite le fait de cliquer sur le bouton "Marquer comme terminé" associé à la tâche "Choix d'un directeur de thèse", puis on clique sur le lien "Sujet de thèse". Une autre assertion permet de vérifier que le dernier terme apparaît bien dans l'en-tête de la page et de s'assurer qu'on est arrivé sur la bonne page. On clique ensuite sur le bouton "Consulter les travaux remis", pour vérifier que le doctorant a bien remis son travail pour l'évaluation. On se dirige ensuite dans la rubrique "Accueil" en cliquant sur l'onglet correspondant dans la barre de navigateur, pour pouvoir y retrouver la timeline de l'étudiant. On vérifie dans ce cas, que les mêmes tâches du Programme doctorale réapparaissent dans la timeline. Finalement, sur ce timeline, on teste le fait que le lien vers une tâche soit bien la bonne, en faisant une assertion, que le nom de la tâche apparaisse dans l'en-tête de la page.

Au total, 36 assertions ont été exécutées pour la vue du doctorant.

Tests sur la vue superviseur

Pour la vue superviseur, la page principale comprend plusieurs programmes doctoraux associés au superviseur. On teste d'abord le triage des doctorants par dernier accès. Puis, on change l'affichage de Carte à Liste. On vérifie par une assertion qu'un étudiant qu'on recherche apparaît dans la liste. On clique ensuite sur son programme doctoral et on vérifie qu'on arrive bien à sa page, avec son nom dans l'en-tête. Tout comme pour le test sur la vue doctorant, on vérifie que toutes les tâches du programme doctoral en Système d'Information apparaissent dans les étapes correspondantes. On se dirige ensuite sur la page associée à la tâche "Sujet de thèse" et on vérifie que le nom de la tâche apparaît dans l'en-tête.

Nous procédons ensuite à l'évaluation du travail soumis. Pour cela, on clique sur "Évaluer" et on vérifie que le nom du doctorant apparaît sur la page, pour s'assurer qu'on est sur la bonne page. En cliquant sur "Non-évalué" puis sur "Activer le mode édition", nous pouvons insérer une note pour le travail fourni par l'étudiant. (Dans le cas présent, la note ne peut qu'être 0,00.) On clique sur "Enregistrer", puis sur "Quitter le mode édition". Pour s'assurer que les modifications ont été apportées, on vérifie que la note qu'on vient d'attribuer apparaît bien sur la page. Ensuite, on clique sur Accueil. On retrouve la liste de tous les doctorants qui sont associés à ce superviseur. Parmi eux, on doit pouvoir retrouver le doctorant qu'on vient de noter. En cliquant sur ce doctorant, on retrouve sa timeline. Tout comme pour la vue doctorant, on vérifie que les mêmes tâches figurant dans le programme doctoral en Systèmes d'Information apparaissent sur la timeline. On teste enfin que le lien associée à une tâche amène bien la bonne page, sur laquelle figure le nom de la tâche en en-tête.

Au total, 37 assertions ont été exécutées pour la vue du superviseur.

Tests sur la vue conseiller aux études

Finalement, pour la vue conseiller aux études, la page principale ne contient pas de programmes doctorales des doctorants associées au conseiller aux études. Nous nous redirigeons donc automatiquement à l'onglet Accueil, en cliquant sur le lien du même nom dans la barre de navigateur. On voit alors s'afficher une liste de doctorants. Le conseiller peut filtrer ces étudiants par faculté. On teste alors qu'un étudiant particulier appartenant à une faculté apparaît toujours lorsque la vue du conseiller passe de "Toutes les facultés" à "la faculté de la GSEM". Lorsqu'on clique ensuite sur ce doctorant, sa timeline apparaît. Comme pour les vues précédentes, on vérifie que la timeline comprend bien toutes les tâches du Programme doctorale de la GSEM. On vérifie ensuite qu'en cliquant sur une tâche, on arrive bien sur la bonne page, c'est-à-dire que le nom de la tâche est contenu dans l'en-tête de la page. Dans ce test, on choisit de cliquer sur "Sujet de thèse", et dans le cas présent, le doctorant

n'a pas encore terminé cette tâche. Nous pouvons donc, en cliquant sur "Consulter tous les travaux soumis", s'attendre à ce que l'étudiant n'ait pas encore soumis son travail pour l'évaluation.

Au total, 19 assertions ont été exécutées pour la vue du directeur.

7. Gestion de projet

7.1. Scrum et gitlab

Pour la méthode de développement, nous avons utilisé la méthode agile “scrum”. Le développement du service a été divisé en différents sprints, dont la liste des tâches (backlog) était définie avant chacun des sprints par une réunion de l’équipe.

Nous avons utilisé les éléments présents dans gitlab pour gérer nos sprints et nos backlogs. Les sprints ont été représentés par les “[milestone](#)” de gitlab qui définissent une période de temps.

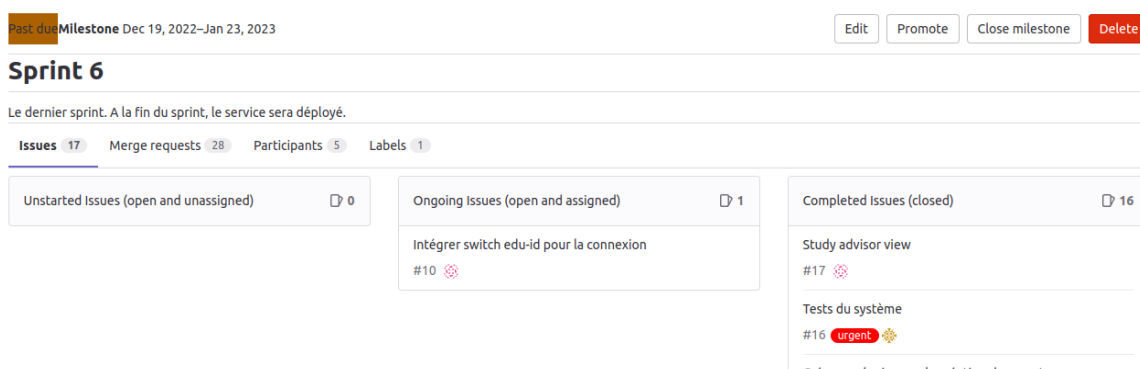


figure 7.a - milestone correspondant au sprint 6

Le backlog et les tâches ont été représentés par une liste d'[issues](#) qui étaient associés à un milestone. Chaque issue peut ensuite être assignée à quelqu'un.

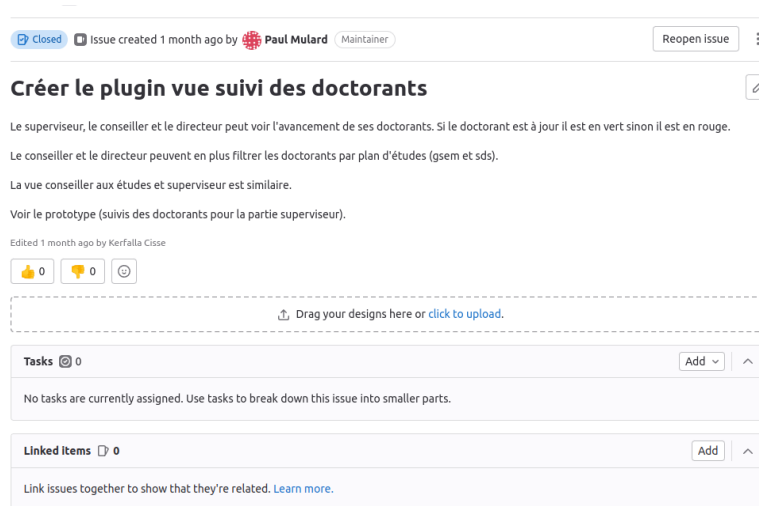


figure 7.b - une issue correspondant à une tâche

Finalement l'issue peut être liée à une [branche](#) de développement. Lorsque le développeur a fini d'implémenter la fonctionnalité, il peut faire une “[merge request](#)”

(demande de fusion) afin d'ajouter le code à la référence principale du projet (branche main). Cette décomposition a permis de simplifier le travail de chacun et de faciliter l'intégration des nouvelles fonctionnalités.

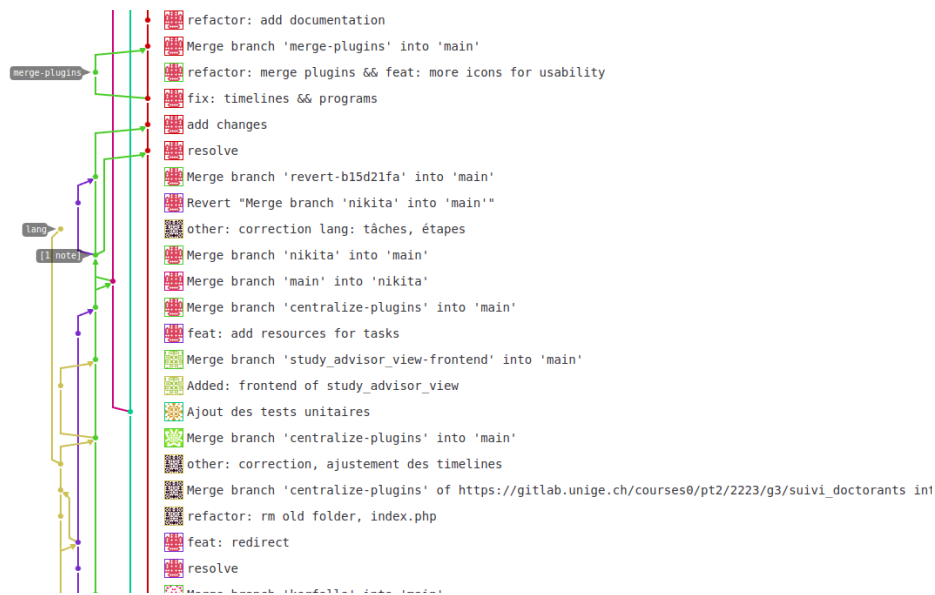


figure 7.c - graphe des fusions entre les différentes branches

Ces différents outils mis à disposition par gitlab nous ont permis de gérer efficacement notre travail et le développement agile de notre projet.

Le vocabulaire gitlab

- milestone: période de temps avec un nom à laquelle on peut attacher des issues.
- issue: tâche à faire (nom et description) à laquelle peut être attachée une branche de développement.
- branche de développement: version alternative du projet qui permet de mieux gérer les conflits entre le code de différents développeurs et de travailler plus facilement.
- merge request: demande de fusion d'une branche dans une autre, c'est-à-dire l'ajout du code créer dans la branche, le système de gitlab va gérer les conflits et les notifier au développeur.

7.2. Utilisation de l'outil de communication Discord

Concernant la communication entre les membres de l'équipe, l'outil qui a été utilisé est Discord. Pour cela, un serveur Discord a été créé. Ce serveur héberge plusieurs canaux différents, ayant chacun une intention précise.

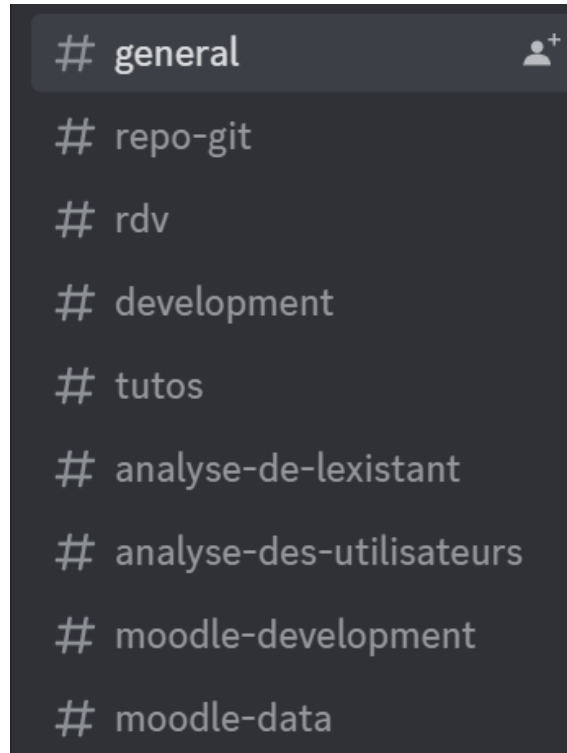


figure 7.d - canaux Discord

- Le canal **#general** permet d'envoyer des messages généraux, c'est-à-dire qui ne sont pas appropriés pour les autres canaux, notamment les rendez-vous de l'équipe et les informations concernant le hackathon qui a eu lieu en début novembre. Les questions concernant certaines implémentations et les commits sur GitLab sont également posées ici.
- Le canal **#repo-git** contient les liens GitHub pour les outils open-source, tels que Angular (que nous n'avons finalement pas utilisé) et Moodle.
- Le canal **#rdv** contient toutes les informations qui concernent le rendez-vous avec le promoteur de notre projet, notamment le lien Zoom, les questions à poser, et les réponses qui ont été fournies, mais également les informations que nous avons retenu du focus group avec les doctorants.
- Le canal **#development** contient notamment le projet Hackademia qui a été réalisé l'année précédente que nous avons utilisé comme inspiration pour le

développement de notre système. Le canal comprend également les informations concernant le déploiement de notre système.

- Le canal [#tutos](#) comprend tous les liens utiles pour apprendre une nouvelle technologie, que ce soit Docker, Moodle ou alors la gestion de projets agile avec GitLab.
- Le canal [#analyse-de-lexistant](#) concerne le projet qui a déjà été réalisé par des étudiants de l'année précédente, et notamment comment le faire marcher.
- Le canal [#analyse-des-utilisateurs](#) comprend les règlements d'études et les programmes d'études que nous avons utilisés pour développer notre système. Ce canal comprend également les informations sur les comités scientifiques associés à ces programmes d'études.
- Le canal [#moodle-development](#) concerne tout le développement avec l'API de Moodle, que ce soit un échange de scripts PHP, de plugins entiers, ou des aides à la paramétrisation du site Moodle.
- Le canal [#moodle-data](#) contient les données à implémenter dans les versions de Moodle de chaque membre de l'équipe, pour que nous travaillons tous avec les mêmes données. Cela peut être une instance des étapes du doctorat, ou alors des données d'utilisateurs test.

7.3. Difficultés d'implémentation

Lors du développement de moodle, nous avons rencontré quelques difficultés de programmation tel que la création des étapes du doctorant et la mise en place des dates relatives des tâches associés aux étapes en fonction de la date d'admission du doctorant.

En effet, bien que l'api de Moodle soit bien documenté, nous n'avons pas trouvé de ressource qui puisse nous aider à implémenter cette fonctionnalité. Nous avons donc dû essayer plusieurs méthodes afin de mettre en place cette fonctionnalité. De plus, certains traitements qui sont assez simples à mettre en œuvre avec PHP peuvent très vite être difficiles à mettre en place dans l'environnement de Moodle.

8. Conclusion

Pour conclure, nous avons pu développer un système qui permet aux superviseurs et conseillers d'études de suivre leurs doctorants, ainsi qu'aux doctorants de visualiser leur avancement dans le but de s'informer et de mieux s'organiser.

Les premières étapes du développement de notre service étaient une profonde analyse de sa problématique et les besoins utilisateurs. Puis nous sommes passés à la conception et au déploiement du système. Enfin, nous avons réalisé plusieurs séries de tests, unitaires et fonctionnels.

Ce système a été réalisé avec le projet open-source moodle. En effet, certaines fonctionnalités de moodle répondaient déjà à des exigences de notre service comme le système de rappel, le système de messagerie ou encore le système de gestion de documents. Nous avons donc pas ré-implémenter ces fonctionnalités

Pour le reste des fonctionnalités, c'est-à-dire la création de suivi de programme d'études et les différentes vues des utilisateurs, il nous a fallu créer un plugin moodle. Le plugin a été implémenté selon les contraintes de développement de moodle et utilise l'API mise à disposition, c'est pourquoi il est compatible avec n'importe quelle instance de moodle et installable en un clic.

Après l'installation du plugin, l'utilisateur peut créer un suivi d'un programme d'études (séquence de tâches à accomplir). Ce suivi est ensuite visible par l'étudiant, les superviseurs et les conseillers d'études avec comme aspect visuel la timeline qui indique à quelle étape se trouve l'étudiant et s'il est en retard ou non sur ses tâches. De plus, les différents utilisateurs peuvent interagir avec le suivi par exemple en validant les tâches (doctorant) ou en modifiant des délais (superviseur).

Ainsi, avec le système de suivi des doctorants, nous espérons aider les acteurs de la vie doctorale, c'est-à-dire les conseillers d'études, les directeurs, les superviseurs et les étudiants dans leurs tâches respectives et simplifier leur travail.

8.1. Travaux futurs

Des travaux futurs peuvent être conduits pour améliorer le service. Voici une liste non exhaustive de ce qui reste à développer/faire.

- Collaboration étroite avec unige,
- Intégration avec d'autre services unige (par exemple PGC)
- Intégration du plugin dans l'éco-système moodle.unige
- Permettre la modification par lots

- Simplifier la gestion des programmes comme l'ajout de délais ou de tâches
- Améliorer le système de deadline de moodle en rendant les dates relatives
- Tester le système de semestres (les algorithmes ont été créés par observation des calendriers issus des années précédentes), demander si l'UNIGE a une API/règlement pour les semestres
- Paramétrer les rappels par mails et le système de messagerie.