Rapport final de SAÉ

Site pour la gestion des avis de poursuites d'études

Réalisé par

BARNIER Dorian, BROUSSARD Tim, SIMONGIOVANNI Alexandre, SOUQUET Gauthier

Projet encadré par

M. TROMBETTONI Gilles

Pour l'obtention du Bachelor Universitaire de Technologie Informatique

Année universitaire 2024-2025

Remerciements

Nous tenons à exprimer notre gratitude envers toutes les personnes qui ont contribué à la réussite de ce projet.

Tout d'abord, nous remercions chaleureusement M. Gilles Trombettoni, notre client, pour sa disponibilité, et la confiance qu'il nous a accordés tout au long de ce projet. Ses retours constructifs ont grandement enrichi notre travail.

Nous souhaitons également adresser nos sincères remerciements à Mme Nathalie Palleja, notre professeur et Super Scrum Master, pour son accompagnement exemplaire, son engagement, et son soutien tout au long de ce projet. Sa pédagogie et sa rigueur ont été des atouts inestimables pour l'atteinte de nos objectifs.

Enfin, nous remercions l'ensemble des membres de l'équipe et toutes les personnes ayant directement ou indirectement participé à ce projet pour leur collaboration et leur investissement.

Sommaire

Table des figures

Glossaire

Les termes définis dans ce glossaire sont identifiables dans le corps du texte au moyen d'un astérisque (*) lors de la première occurrence.

Les mots apparaissent dans l'ordre alphabétique.

Mot en gras : définition du terme ou développement de l'acronyme.

Product Owner : Responsable de la gestion du backlog et des priorités pour maximiser la valeur du produit.

User Stories : Responsable de la gestion du backlog et des priorités pour maximiser la valeur du produit.

Planning Poker : Méthode d'estimation collaborative des efforts/complexités en Agile, avec des cartes.

Daily Scrum Meeting : Réunion quotidienne de 15 minutes pour aligner l'équipe et identifier les obstacles.

Sprint : Période de travail, généralement de 2 à 4 semaines, au cours de laquelle une équipe agile travaille sur un ensemble spécifique de tâches ou de fonctionnalités.

Backlog : Liste priorisée des tâches, fonctionnalités ou améliorations à réaliser dans un projet agile.

Diagrammes UML (Unified Modeling Language) : Les diagrammes UML aident à représenter graphiquement des concepts comme les classes, les cas d'utilisation, les séquences, et plus.

MVC (Modèle-Vue-Contrôleur) : Architecture logicielle qui sépare une application en trois parties : le modèle (logique métier), la vue (interface utilisateur) et le contrôleur (gère les interactions entre le modèle et la vue).

phpMyAdmin : Outil libre de gestion de bases de données MySQL, utilisé pour administrer et gérer des bases de données via une interface web.

RGPD : Règlement général sur la protection des données, il s'agit de la réglementation de l'Union européenne concernant la collecte, le traitement et le stockage des données personnelles.

Merge Requests : Demandes de fusion dans un système de gestion de version (comme Git), permettant de proposer les modifications d'une branche pour les

intégrer dans une autre.

Gitflow : Modèle de gestion de branches dans Git qui définit des processus de travail pour le développement de logiciels, incluant des branches pour les fonctionnalités, les versions, et les corrections de bugs.

Burnup chart métier : Graphique représentant l'avancement d'un projet en termes de tâches réalisées, souvent utilisé pour suivre la progression du travail au fur et à mesure des itérations ou des sprints.

Revues de sprint : Réunions à la fin d'un sprint où l'équipe présente le travail accompli, recueille des retours et ajuste la planification pour le sprint suivant.

Burndown charts : Un graphique utilisé en gestion de projet Agile pour visualiser la quantité de travail restante par rapport au temps.

Vélocité : Une mesure en Agile qui représente la quantité de travail (souvent en points ou en tâches) qu'une équipe peut accomplir pendant un sprint.

SAÉ : Situation d'Apprentissage et d'Évaluation

LDAP : Lightweight Directory Access Protocol, Un protocole utilisé pour accéder et gérer des services d'annuaires, comme des bases de données centralisées contenant des informations sur les utilisateurs, groupes ou permissions dans un réseau.

Introduction

Dans un contexte où les parcours académiques deviennent de plus en plus variés et compétitifs, les établissements d'enseignement supérieur doivent s'assurer de gérer efficacement le processus d'évaluation des candidatures. Face à des méthodes souvent manuelles et peu ergonomiques, le besoin d'une solution numérique centralisée s'impose.

Ce rapport détaille le développement d'un site web conçu pour gérer les avis de poursuites d'études des étudiants du département Informatique de l'IUT Montpellier-Sète. Ce projet a été réalisé dans le cadre d'une **SAÉ*** (Situation d'Apprentissage et d'Évaluation) en collaboration avec M. Gilles Trombettoni, notre client, et Mme Nathalie Palleja, notre encadrante.

L'objectif principal était de concevoir une application intuitive permettant aux jurys d'attribuer des mentions (très favorable, favorable, réservé) et d'ajouter des commentaires pour chaque étudiant, tout en respectant les contraintes techniques, légales et ergonomiques. Ce rapport présente successivement l'analyse des besoins, la conception et la réalisation technique, ainsi que les méthodologies de gestion de projet employées. Enfin, il évalue les résultats obtenus et les enseignements tirés de cette expérience collaborative.

1. Analyse

Cette analyse vise à explorer les besoins fonctionnels et non fonctionnels du projet, en mettant en lumière les attentes du client, le contexte du projet, et les spécifications nécessaires pour répondre aux objectifs définis. Nous utiliserons des outils de modélisation comme les diagrammes **UML*** pour clarifier les relations entre les entités, ainsi que les processus et scénarios d'utilisation. Enfin, nous identifierons les contraintes techniques, ergonomiques et légales qui influenceront la conception et le développement du système.

1. Présentation du sujet et analyse du contexte

Le projet consiste à développer une plateforme destinée à faciliter le jury de poursuite d'études pour les étudiants en troisième année de BUT informatique. Ce système permettra de visualiser les notes d'un étudiant ainsi que les avis donnés pour sa poursuite d'études. Les principales fonctionnalités incluent la consultation des données, l'attribution d'avis, et l'agrégation des notes à partir de plusieurs ressources et notes agrégées.

Analyse du sujet

Le besoin principal exprimé par le client est d'offrir un outil centralisé pour :

- Consulter les notes et les avis des étudiants.
- Fournir des avis de poursuite d'études directement à partir d'un tableau interactif.
- Importer des données étudiantes via des fichiers CSV.
- Créer des moyennes pondérées à partir de ressources spécifiques.

Ce projet cible trois types d'utilisateurs :

1. **Responsable de poursuite d'études** : Gestion des utilisateurs et des avis.
2. **Professeurs** : Consultation des notes et attribution d'avis.
3. **Étudiants** : Accès à leurs propres notes.

Analyse de l'existant

Actuellement, des outils comme ScoDoc ou des tableurs sont utilisés pour répondre à ces besoins. Cependant, ces solutions ne permettent pas une gestion centralisée et intuitive des données ni l'automatisation des avis et des moyennes. Le nouveau système cherche à pallier ces lacunes.

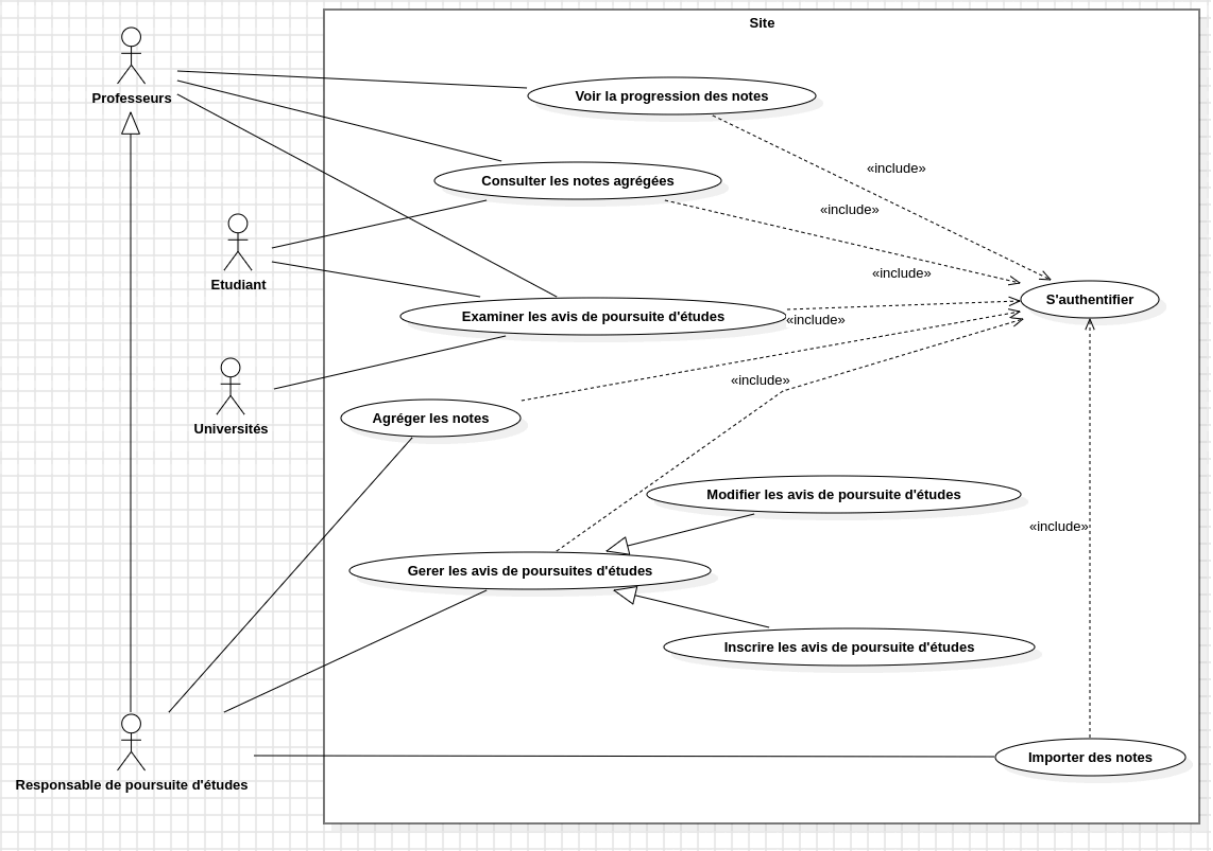
Contexte d'intégration

Le système sera développé en HTML, CSS, PHP et JavaScript avec une base

de données gérée via **phpMyAdmin***. L'application sera déployée dans un environnement sans framework, respectant ainsi les exigences techniques du client.

1.1. Présentation des besoins fonctionnels (backlog/users stories)

Les besoins fonctionnels représentent les fonctionnalités principales que le système doit offrir pour répondre aux attentes des utilisateurs. Ces besoins ont été modélisés sous forme de diagrammes UML, notamment le diagramme de cas d'utilisation ci-dessous :



Description des fonctionnalités principales

1. **S'authentifier** : Les utilisateurs doivent s'identifier avec un login et un mot de passe pour accéder au système.
2. **Consulter les notes et les avis** : Les professeurs peuvent consulter les notes et avis de tous les étudiants. Les étudiants peuvent consulter leurs propres notes.
3. **Donner un avis** : Le responsable de poursuite d'étude peut attribuer un avis à partir d'un tableau interactif.

4. **Créer une note agrégée** : Calcul des moyennes pondérées à partir de différentes ressources avec coefficients.
5. **Importer des données** : Le responsable de poursuite d'études peut importer des fichiers CSV contenant les informations des étudiants.
6. **Gérer les utilisateurs** : Création et gestion des comptes utilisateurs par le responsable de poursuite d'études.

1.2. Analyse des besoins non-fonctionnels

Les besoins non fonctionnels incluent les contraintes et spécifications techniques, légales et ergonomiques qui doivent être respectées pour garantir la qualité et la conformité du système.

1.2.1. Spécifications techniques

Les spécifications techniques prévues pour le projet sont les suivantes :

- **Environnement de développement** : Le projet sera développé en HTML, CSS, PHP et JavaScript, assurant une compatibilité avec les technologies web modernes.
- **Base de données** : La gestion des données sera assurée via phpMyAdmin, garantissant une administration simplifiée.
- **Compatibilité** : Le système sera conçu pour fonctionner sur les navigateurs récents et ne nécessitera pas l'utilisation d'un framework.

1.2.2. Contraintes légales

Pour respecter le **RGPD***, les mesures suivantes seront mises en place :

- **Stockage sécurisé des données** : Les données des utilisateurs seront stockées de manière sécurisée afin d'éviter tout accès non autorisé.
- **Collecte limitée des données** : Seules les informations nécessaires, comme les noms et prénoms des utilisateurs, seront collectées. Aucune donnée sensible ne sera enregistrée.

1.2.3. Contraintes ergonomiques

L'ergonomie du système repose sur les principes suivants :

- **Interface utilisateur intuitive** : Une attention particulière sera portée à la création d'un tableau interactif facile à utiliser par tous les profils d'utilisateurs.
- **Accessibilité** : Bien qu'aucune norme spécifique ne soit requise, l'interface sera conçue pour garantir une utilisation fluide et claire.

2. Rapport technique

Dans cette section, nous présenterons les aspects techniques du projet.

Nous détaillerons les étapes clés de la conception et de la réalisation, en illustrant notre démarche avec des extraits de code pertinents et commentés.

L'objectif est d'expliquer non seulement ce que nous avons fait, mais aussi pourquoi nous l'avons fait, tout en offrant une perspective pédagogique sur notre démarche.

Les exemples de code sont soigneusement choisis pour mettre en avant des points importants, tels que la structure de notre application, la gestion des données ou la mise en œuvre de fonctionnalités spécifiques. Chaque extrait est accompagné d'explications et de commentaires détaillés pour garantir une compréhension claire et approfondie.

2.1. Conception

Cette section présente une vue logique du logiciel, en identifiant les principales entités du domaine, les processus qui interagissent avec elles, ainsi que les règles métier. Nous détaillons également les choix de conception effectués pour structurer l'application et organiser le code.

L'application repose sur une architecture MVC (Modèle-Vue-Contrôleur) et utilise une base de données relationnelle gérée avec phpMyAdmin. Ces choix permettent de séparer les responsabilités, de garantir une bonne organisation du code, et d'assurer une gestion efficace des données.

2.1.1 Base de données

La base de données, structurée en SQL via phpMyAdmin, est utilisée pour stocker les principales entités du domaine (utilisateurs, étudiants, parcours, notes, etc.). Elle permet de gérer les relations entre ces entités et garantit l'intégrité des données grâce à des clés primaires et étrangères.

2.1.2 Structure et organisation

L'architecture MVC a été choisie pour séparer les responsabilités :

Modèle : Gère les données et leur interaction avec la base SQL.
Vue : Présente les données à l'utilisateur via une interface en HTML/CSS.
Contrôleur : Intermédiaire entre les deux, il orchestre les actions en fonction des requêtes utilisateur.

2.1.3 Fonctionnalités clés

Les principales fonctionnalités (authentification, gestion des avis de poursuites d'étude, gestion des agrégations) reposent sur des algorithmes optimisés pour interagir efficacement avec la base de données et maintenir une expérience utilisateur fluide.

2.2. Réalisations

Pour l'implémentation du code, nous avons opté pour des conventions de nommage simples et compréhensibles, tout en restant concis et en utilisant des termes en français. Par exemple, une liste d'étudiants était nommée « étudiants », ou des fonctions étaient désignées par des phrases courtes telles que « getNoteParEleve ». Une fois chaque composant du site développé, il était crucial de les tester. Pour ce faire, nous avons rédigé des tests sur Trello représentant une liste de fonctionnalités à valider. Après l'implémentation de chaque fonctionnalité, nous l'avons testée dans une vue séparée afin de vérifier qu'elle répondait bien à nos exigences.

2.3. Test/Validation

Des tests ont été réalisés sur Trello et étaient rédigés par le Product Owner. Chaque tâche comportait environ 5 à 6 tests, qui couvraient différentes situations, y compris des cas extrêmes, afin de garantir la réussite du projet.

Par la suite, nous avons également créé des vues de test afin de vérifier le bon fonctionnement des différentes fonctions ou méthodes que nous avons développées. Ces vues de test nous permettaient d'exécuter et d'observer le ce que la fonctionnalité retournait ou faisait. Cela nous permet de détecter et de corriger d'éventuelles erreurs ou anomalies avant d'intégrer ces fonctionnalités dans la vue principale ou l'application finale. Cette approche garantit une meilleure stabilité et une qualité accrue du produit final en limitant les risques liés à l'introduction de bugs.

2.4. Qualité de développement

Premièrement, un bon exemple issu de notre SAE serait la partie agrégation, celle-ci permet au jury de pouvoir gérer des agrégations et de s'en servir dans la partie liste des étudiants afin de les filtrer par celles-ci afin de mettre avant les points souhaités.

Cette partie (voir commit init) se décompose en plusieurs sous partie, tout d'abord, la partie gestion des agrégations (voir annexe) celle-ci devait nous permettre de créer, supprimer et mettre à jour une agrégation, pour cela nous avons décidé de séparer notre page en 2 partie, la partie gauche qui nous permettrai de créer nos agrégations composé de une ou plusieurs ressources et/ou de une ou plusieurs agrégations (pouvant elles mêmes être composé comme décrit précédemment), une des contraintes de cette création était d'interdire la création d'agrégation n'étant composé ni de ressource ni d'agrégation car cela nuirait au calcul de celles-ci. Une fois l'agrégation créée nous pouvons la retrouver en rafraîchissant la page à droite cette fois-ci où nous pouvons mettre à jour et supprimer les agrégations.

Enfin, ces agrégations se retrouvent dans notre liste des étudiants (voir annexe) où nous pouvons sélectionner dans le filtre une ou plusieurs agrégations souhaitées afin de mettre à jour le tableau pour y afficher chaque étudiants avec ses notes calculées pour chaque agrégations.

Pour ce qui est de la solution mise en place (voir diagramme de classes) pour pouvoir créer une agrégation composée d'éléments et d'agrégations, nous avons décidé d'utiliser un pattern "liste de pièces", ainsi nous manipulons 3 objets : les ressources, les agrégations et les éléments qui sont une super classe regroupant les ressources et les agrégations.

De ce fait, lors de la création d'une agrégation (et d'une ressource mais nous ne traitons pas de celà dans cette partie) nous créons d'abord un élément ayant pour idElement, l'id de l'agrégation (son nom dans notre conception), puis l'agrégation associé (voir méthode). Pour stocker sa composition nous avons créons une classe (et une table dans notre base de données) qui associe un idAgregation avec un idElement ainsi nous pourrions ajouter et supprimer des catégorie "d'éléments" et notre solution resterait inchangé, par exemple nous pourrions décider de séparer les notes des SAE et ainsi créer une classe (et donc une table en base de données) SAE et s'en servir dans le calcul de certaines agrégations.

Ensuite, du point de vue de la modification et de la suppression qui se situe dans la partie droite de notre page, nous avions mis en place une généralisation du code pour éviter la duplication mais nous avons été amené à utiliser celle-ci d'une manière différente, nous avons créé d'autres méthodes (voir méthode mettreAJour et supprimer) qui nous permettent de supprimer tous les éléments associés à cette agrégation ainsi que son association avec les agrégations mère dans lesquels il était contenu. Enfin, pour ce qui est du calcul des notes en rapport avec les agrégations de chaque élève, nous vérifions chaque élément et traitons son comportement en fonction de son type que nous pondérons avec les coefficient donnés par l'utilisateur à la création.

De plus, l'utilisation de l'architecture **MVC*** nous permet le respect d'un principe SOLID, le Single Responsability, en effet, nous séparons les modèles en deux avec d'un côté les data object qui définissent les objets et de l'autre les repository qui définissent leur comportement. Nous avons également les vues qui représentent la partie front-end de notre site ou encore les controllers qui eux font le lien entre le front-end (donc les vues) et le back-end (donc les modèles), chaque classe a donc une seule responsabilité.

Par la suite, il nous a été demandé de réaliser une connexion entre le réseau de l'ENT de Montpellier nommé **LDAP*** ainsi que notre site web afin que les étudiants, les professeurs et le directeur de poursuite d'étude puissent y accéder directement via leurs identifiants.

Pour chacune des utilisations de LDAP il faut instancier une connexion et récupérer tous les utilisateurs pour les filtrer par la suite.

Il a donc été décidé de créer une classe spécifique pour la connexion avec LDAP et une fonction permettant de récupérer directement tous les résultats, de cette manière nous évitons la duplication de code ainsi que la trop forte responsabilité d'une certaine classe.
Ici (annexe 1), la classe ConnexionLDAP permet aux classes qui utilisent les résultats LDAP d'éviter une duplication du code récurrente (annexe 2) lors de la récupération des utilisateurs.

Nous avons également intégré un pattern Singleton pour cette classe afin d'éviter que plusieurs connexions coûteuses soient créées en même temps, minimisant donc les risques de saturer la base de données LDAP.

Commit fait lors de l'ajout de la classe "ConnexionLDAP" :
https://gitlabinfo.iutmontp.univ-montp2.fr/sae3a/projets/simongiovannia-souquetg-broussardt-barnierd/sae3a-base/-/commit/01c89b78fa584178049b03185f66cba9fb2494cc

3. Gestion de projet

Dans cette section, nous explorerons la gestion de notre projet en nous appuyant sur la méthode agile Scrum. Nous analyserons de manière critique les retours issus des revues de sprint et ferons le bilan de notre travail en équipe, en mettant en évidence les forces, les axes d'amélioration et les enseignements tirés au fil des sprints.

3.1. Méthode agile

Pour mettre en œuvre la méthode agile, nous avons commencé par attribuer les rôles de **Scrum Master*** et de **Product Owner*** aux membres de l'équipe par un vote. Cette organisation vise à assurer une meilleure gestion tout au long de chaque sprint. Ensuite, nous avons organisé des sessions de ***planning poker**** pour collaborer et rédiger ensemble les ***user stories****. Cela nous a permis d'attribuer une valeur métier à chaque fonctionnalité, ainsi qu'une estimation de l'effort nécessaire pour sa réalisation, favorisant ainsi une priorisation efficace et partagée.

Afin de maintenir la bonne avancée du projet, nous avons mis en place des **daily scrum meetings*** ainsi que divers diagrammes (**Burnup chart métier***, **Burnup Chart effort***, **Burndown Chart***, **Vélocité***)

Alex:

Pour la gestion de ce projet, nous avons adopté la méthode agile Scrum. Cela nous a permis de structurer notre travail en sprints, avec des objectifs clairs et des itérations régulières pour améliorer et affiner notre produit. Nous avons commencé par définir les rôles clés dans l'équipe :

- **Product Owner*** : Ce rôle a été attribué à Alexandre SIMONGIOVANNI. Il était responsable de la gestion des priorités et de la définition des user stories, en assurant que le produit répondait aux attentes du client.
- **Scrum Master*** : Gauthier SOUQUET a pris ce rôle afin de faciliter la communication et de garantir que les bonnes pratiques Scrum étaient respectées. Il a également veillé à éliminer les obstacles pouvant nuire à l'avancement du projet.

Nous avons organisé des sessions de **planning poker*** pour estimer l'effort nécessaire pour chaque fonctionnalité. Cette approche a permis de clarifier nos objectifs et de nous assurer que l'ensemble des membres de l'équipe avait une vision partagée du travail à réaliser.

Afin de suivre l'avancée du projet, nous avons régulièrement utilisé des **daily scrum**

meetings pour discuter des tâches accomplies, des problèmes rencontrés et des prochaines étapes. Ces réunions ont favorisé une communication fluide et une réactivité face aux éventuels obstacles.

Organisation des Sprints

Le projet a été divisé en plusieurs sprints, chacun d'une durée de deux semaines. Pour chaque sprint, nous avons planifié les tâches à accomplir en fonction des priorités définies dans le **backlog*** du produit. À la fin de chaque sprint, nous avons effectué une **revue de sprint*** pour évaluer les progrès réalisés et ajuster nos objectifs si nécessaire. Les **Burnup charts** et **Burndown charts** ont été utilisés pour suivre la progression du travail et les efforts nécessaires pour terminer chaque sprint.

Planning global des sprints :

- **Sprint 0 :** Analyse des besoins, maquettage et réalisation de diagrammes.
- **Sprint 1** : Développement des premières fonctionnalités, tableau principal, ajout d'avis, importation des CSV, création de la base de donnée
- **Sprint 2** : Ajout et modifications des fonctionnalités de base, modification du contenu du tableau, visualisation des notes, calcul des moyennes
- **Sprint 3** : Modification des fonctionnalités existantes et ajout de features comme, pouvoir s'authentifier, créer des notes agrégées, gestion des utilisateurs par l'administrateur, génération de PDF.
- **Sprint 4** : Amélioration et finalisation du site, ce sprint nous permet de finir toutes les fonctionnalités qui avaient un problème lors des sprint précédents.

3.2. Bilan critique par rapport aux revues de sprint

Lors des revues de sprint, nous avons appris à mieux estimer les efforts nécessaires pour certaines tâches. Les premiers sprints ont été marqués par des délais plus longs que prévus en raison de la complexité des tâches et des ajustements nécessaires dans l'architecture du projet. Cependant, à chaque sprint, nous avons affiné nos estimations, ce qui a permis d'améliorer la planification et de réduire les imprévus dans les sprints suivants.

Les **Burnup charts** ont été particulièrement utiles pour visualiser l'avancement du travail et ajuster les priorités en fonction des retards ou des blocages identifiés. À la fin de chaque sprint, nous avons évalué la vitesse de travail de l'équipe (vélocité) et adapté nos objectifs en fonction des capacités réelles de l'équipe.

3.3. Bilan du travail en équipe

Le travail en équipe a été globalement harmonieux, mais quelques points d'amélioration ont été identifiés. Le système de gestion des **branches Git** a facilité la collaboration entre les membres, permettant à chacun de travailler sur des fonctionnalités spécifiques tout en évitant les conflits de code. Nous avons mis en place un **gitflow*** structuré avec des branches dédiées pour chaque fonctionnalité et des **Merge Requests*** pour l'intégration de ces fonctionnalités dans la branche principale.

L'une des **Merge Requests*** les plus significatives a concerné l'intégration de la gestion des utilisateurs. Cette fonctionnalité a impliqué des ajustements sur la base de données, ainsi que la mise en place de processus de validation des comptes et d'attribution de rôles. Le processus a été largement testé avant son intégration, et l'équipe a veillé à ce que les spécifications initiales soient respectées.

En ce qui concerne les **réunions**, elles se sont bien déroulées, mais nous avons constaté qu'une meilleure préparation des sujets à discuter aurait permis de gagner en efficacité. De plus, les décisions prises lors des réunions ont été documentées dans des **comptes-rendus** partagés, permettant de s'assurer que tout le monde était aligné sur les actions à entreprendre.

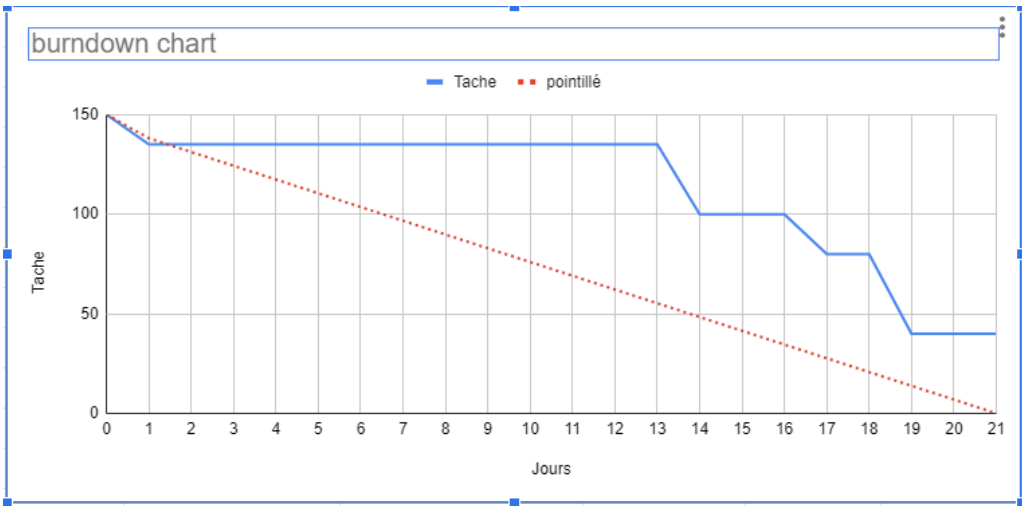
Évaluation de la qualité du travail en équipe

Le travail en équipe a été globalement très efficace, avec une répartition claire des tâches et une collaboration constante. Les **revues de sprint*** ont permis d'identifier rapidement les points à améliorer, et chaque membre de l'équipe a contribué de manière significative au projet. Le suivi des **commit** sur GitLab a permis de garder une trace claire des contributions de chacun, facilitant ainsi le processus de validation et de mise en production.

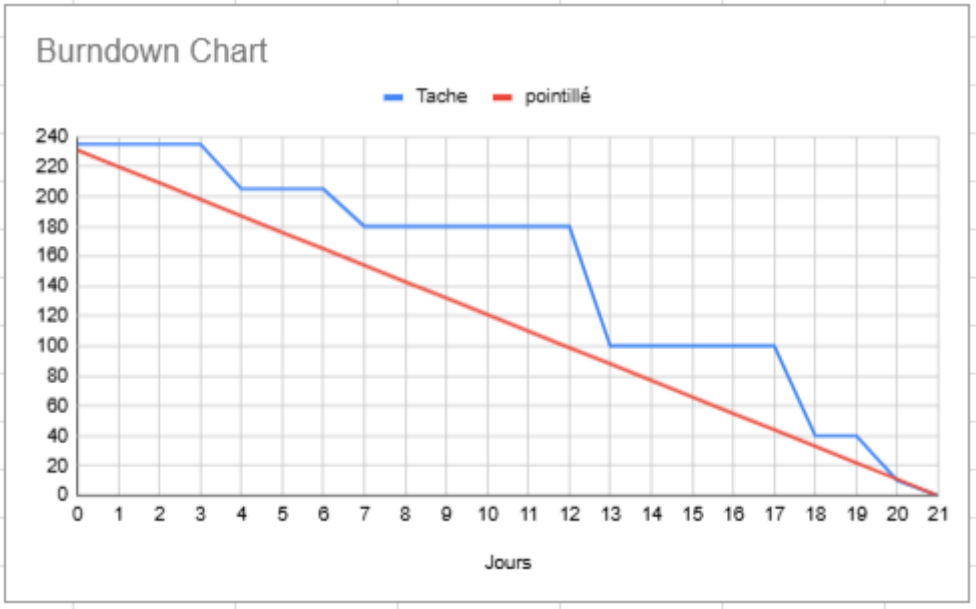
La qualité du travail en équipe peut être attribuée à notre capacité à nous adapter aux défis du projet, à utiliser des outils collaboratifs de manière optimale et à maintenir une communication constante. Toutefois, une meilleure gestion des délais et des prévisions de ressources pourrait encore améliorer l'efficacité future.

3.4. Analyse des diagrammes

Burndown chart du sprint 1:



Burndown chart du sprint 4:



Dans le cadre de la SAE, nous avons utilisé plusieurs outils pour suivre notre progression et améliorer notre productivité. Parmi ces outils, les diagrammes de burndown chart ont été essentiels pour visualiser l'évolution de notre travail au cours des sprints. Nous avons décidé de présenter celui du premier sprint et celui du dernier pour mettre en évidence l'amélioration de notre efficacité.

Le premier burndown chart montrait que nous avons mis beaucoup de temps à commencer et que nous avons pas eu le temps de tout finir, ce qui reflétait une phase de prise en main et d'adaptation aux exigences du projet. En revanche, lors du dernier sprint, le burndown chart montre une courbe plus régulière et une meilleure gestion du temps, ce qui témoigne d'une amélioration notable de notre capacité à respecter les délais et à finaliser les tâches.

Une des raisons principales de cette amélioration réside dans l'utilisation des diagrammes de vélocité. En analysant notre capacité de production à chaque sprint,

nous avons pu ajuster nos estimations d'effort et mieux répartir les tâches entre les membres de l'équipe. Les valeurs d'effort que nous avons attribuées à chaque tâche ont été réévaluées à chaque itération, ce qui nous a permis d'être plus réalistes dans nos prévisions et d'ajuster notre charge de travail en fonction de notre vélocité.

Ainsi, cette démarche d'analyse continue des burndown charts et des diagrammes de vélocité nous a permis d'identifier nos points forts et nos axes d'amélioration. Grâce à une meilleure estimation des efforts et une gestion plus rigoureuse de notre capacité de production, nous avons pu améliorer notre efficacité d'un sprint à l'autre et livrer des résultats de plus en plus fiables et dans les délais impartis.

4. Bilan des apprentissages

Ce projet nous a permis de renforcer nos compétences en gestion de projet, en particulier dans l'application de la méthode agile Scrum. Nous avons acquis de l'expérience dans la gestion des sprints, la priorisation des tâches et la collaboration en équipe. Les différents **rôles Scrum** nous ont permis de mieux comprendre la répartition des responsabilités et l’importance d’une communication fluide. L'utilisation d'outils comme GitLab, Trello et les diagrammes de suivi de projet a été essentielle pour la gestion du travail collaboratif.

Enfin, cette expérience nous a permis de mieux comprendre les enjeux de la gestion de projet dans un environnement de développement logiciel, tout en nous offrant un cadre d’apprentissage pour l’application de méthodologies agiles dans un contexte réel.

Conclusion

La réalisation de ce projet a permis de répondre aux attentes exprimées par notre client en fournissant un outil performant, ergonomique et conforme aux contraintes imposées. Le site de gestion des avis de poursuites d'études centralise et simplifie les processus d'évaluation, tout en garantissant la sécurité et la confidentialité des données conformément au RGPD.

Ce projet a également représenté une opportunité pour l'équipe de mettre en pratique les compétences acquises tout au long de notre formation en Bachelor Universitaire de Technologie Informatique, notamment dans les domaines de l'analyse des besoins, de la conception logicielle, de la programmation, et de la gestion de projet en méthode agile.

Enfin, ce travail a mis en évidence l'importance de la collaboration et de la communication au sein d'une équipe, ainsi que les défis liés au respect des délais et à l'adaptation aux retours du client. Ces enseignements seront précieux pour nos projets futurs, tant académiques que professionnels.

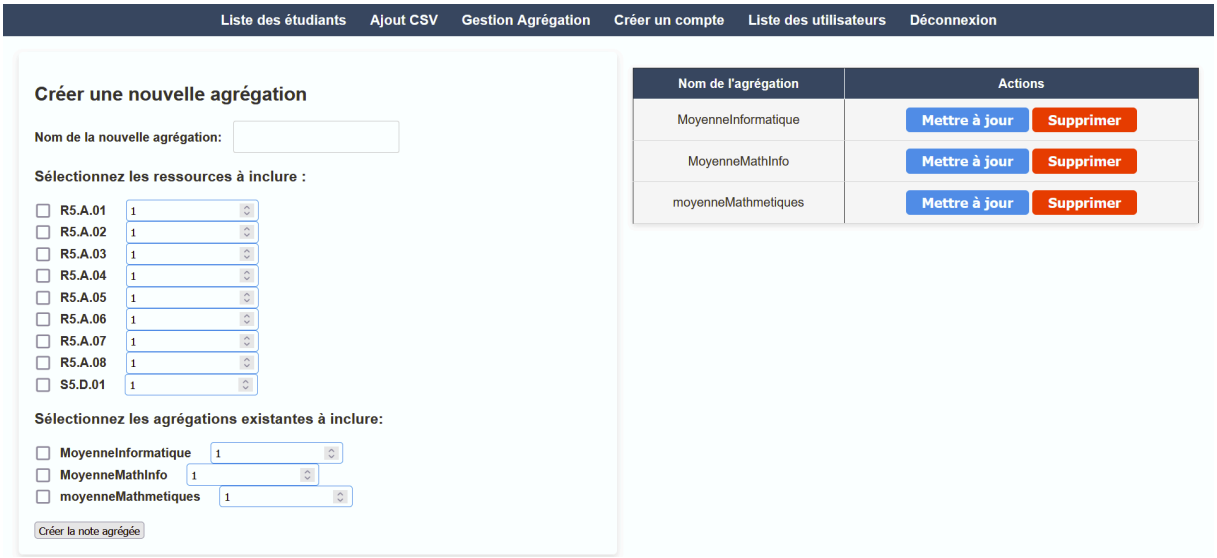
Bibliographie

[1] FPDF, "FPDF: Free PDF Generator for PHP,". Lien: http://www.fpdf.org/. Jan. 8, 2025.

Annexes techniques

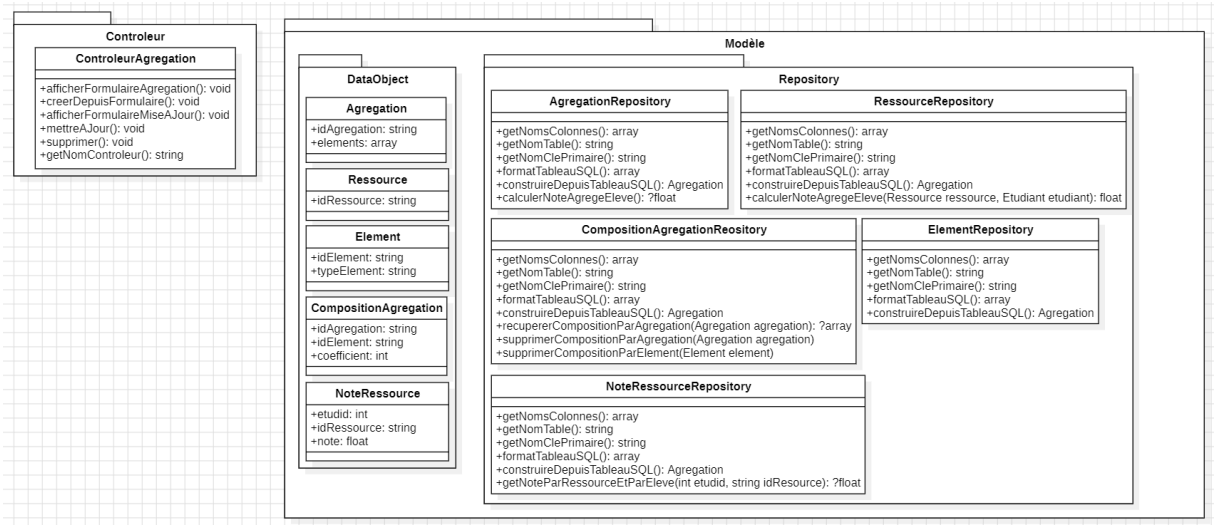
Annexes premier exemple :

Annexe 1 : Page gestion des agrégations



Annexe 2 : Page liste des étudiants

Annexe 3 : Diagramme de classe



Annexe 4 : classe ControleurAgregation - méthode creerDepuisFormulaire

```php
no usages  Tim Broussard
public static function creerDepuisFormulaire(): void
{
    $elementFormatTableau = array(
        "idElement" => $_POST["idAgregation"],
        "typeElement" => "agregation");
    $element = (new ElementRepository())->construireDepuisTableauSQL($elementFormatTableau);
    (new ElementRepository())->ajouter($element);

    $agregationFormatTableau = array(
        "idAgregation" => $_POST["idAgregation"]);
    $agregation = (new AgregationRepository())->construireDepuisTableauSQL($agregationFormatTableau);
    (new AgregationRepository())->ajouter($agregation);

    if (!isset($_POST["composition"])) {
        (new ControleurAgregation())->afficherErreur( messageErreur: "agregation vide");
        (new AgregationRepository())->supprimer($agregation->getIdAgregation());
        (new ElementRepository())->supprimer($element->getIdElement());
    } else {
        $composition = $_POST["composition"];
        foreach ($composition as $element) {
            $compositionAgregationFormatTableau = array(
                "idAgregation" => $_POST["idAgregation"],
                "idElement" => $element,
                "coefficient" => $_POST["coefficients"][$element]
            );
            $compo = (new CompositionAgregationRepository())->construireDepuisTableauSQL($compositionAgregationFormatTableau);
            (new CompositionAgregationRepository())->ajouter($compo);
        }
        ControleurAgregation::afficherVue( cheminVue: "vueGenerale.php", [
            "titre" => "Agrégation Crée",
            "cheminCorpsVue" => "agregation/agregationCree.php",
        ]);
    }
}
```

Annexe 5 : classe ControleurAgregation - méthode mettreAJour

```php
public static function mettreAJour() : void
{
    if (!ConnexionUtilisateur::estAdministrateur() && !ConnexionUtilisateur::getTypeUtilisateur() == "Professeur" && !ConnexionUtilisateur::getTypeUtilisateur() == "Université") {
        (new ControleurAgregation())->afficherErreur( messageErreur: "vous n'avez pas l'autorisation de modifier cette agrégation");
    }
    elseif (!isset($_REQUEST["composition"])) {
        (new ControleurAgregation())->afficherErreur( messageErreur: "agregation vide");
    }
    else {
        $agregation = (new AgregationRepository())->recupererParClePrimaire($_REQUEST["idAgregation"]);
        CompositionAgregationRepository::supprimerCompositionParAgregation($agregation);
        $composition = $_REQUEST["composition"];
        foreach ($composition as $element) {
            $compositionAgregationFormatTableau = array(
                "idAgregation" => $_REQUEST["idAgregation"],
                "idElement" => $element,
                "coefficient" => $_REQUEST["coefficients"][$element]
            );
            $compo = (new CompositionAgregationRepository())->construireDepuisTableauSQL($compositionAgregationFormatTableau);
            (new CompositionAgregationRepository())->ajouter($compo);
        }
        ControleurAgregation::afficherVue( cheminVue: "vueGenerale.php", [
            "titre" => "Agrégation mise à jour",
            "cheminCorpsVue" => "agregation/agregationMiseAJour.php",
        ]);
    }
}
```

Annexe 6 : classe ControleurAgregation - méthode supprimer

```php
public static function supprimer() : void
{
    if (!ConnexionUtilisateur::estAdministrateur() && !ConnexionUtilisateur::getTypeUtilisateur() == "Professeur" && !ConnexionUtilisateur::getTypeUtilisateur() == "Université") {
        (new ControleurAgregation())->afficherErreur( messageErreur: "vous n'avez pas l'autorisation de supprimer cette agrégation");
    }
    else {
        $agregationASupprimer = (new AgregationRepository())->recupererParClePrimaire($_REQUEST["idAgregation"]);
        $elementASupprimer = (new ElementRepository())->recupererParClePrimaire($_REQUEST["idAgregation"]);
        CompositionAgregationRepository::supprimerCompositionParAgregation($agregationASupprimer);
        CompositionAgregationRepository::supprimerCompositionParElement($elementASupprimer);
        (new AgregationRepository())->supprimer($agregationASupprimer->getIdAgregation());
        (new ElementRepository())->supprimer($agregationASupprimer->getIdAgregation());
        ControleurAgregation::afficherVue( cheminVue: "vueGenerale.php", [
            "titre" => "Agrégation supprimé",
            "cheminCorpsVue" => "agregation/agregationSupprime.php",
        ]);
    }
}
```

Annexe 7 : classe AgregationRepository - méthode calculerNoteAgregeEleve

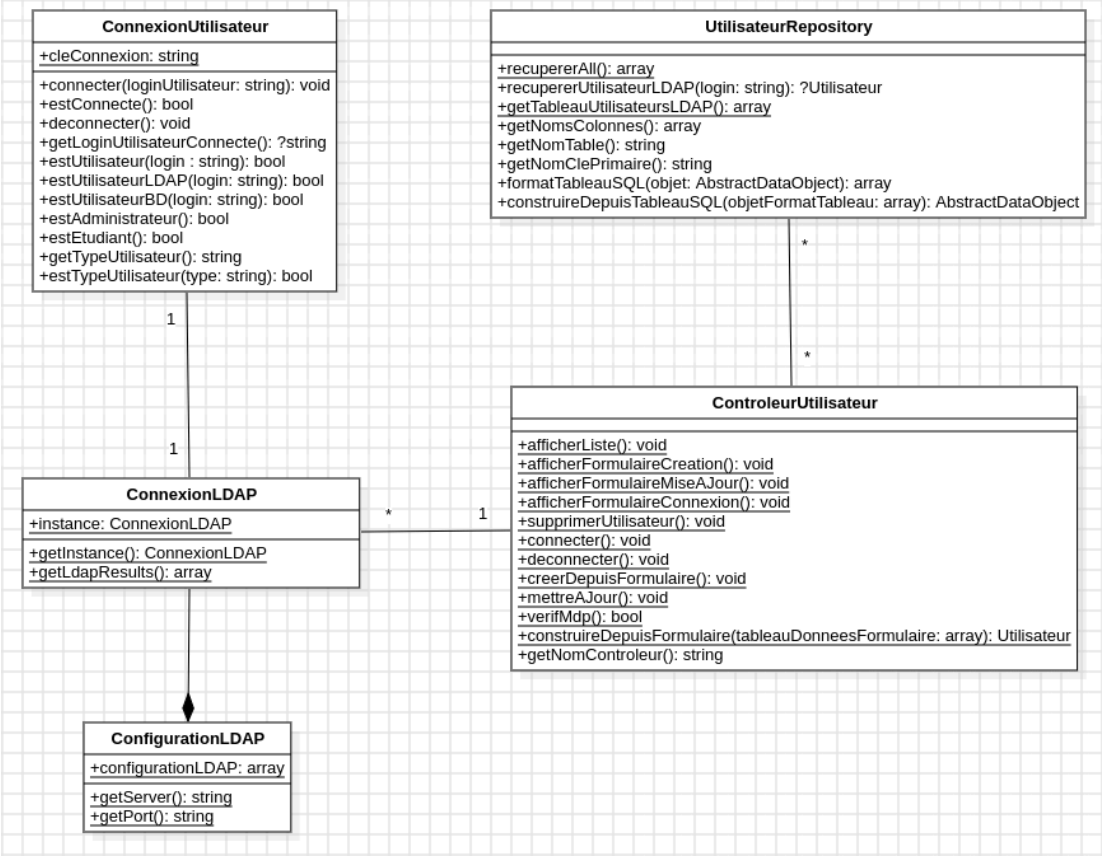
```php
public static function calculerNoteAgregeEleve(Agregation $agregation, Etudiant $etudiant) : ?float
{
    $composition = CompositionAgregationRepository::recupererCompositionParAgregation($agregation);
    $coefficientTotal = 0;
    $noteTotal = 0;
    foreach ($composition as $compositionAgregation) {
        $coefficient = $compositionAgregation->getCoefficient();
        if ((new AgregationRepository())->verifExistant($compositionAgregation->getIdElement())) {
            $elem = (new AgregationRepository())->recupererParClePrimaire($compositionAgregation->getIdElement());
            $note = AgregationRepository::calculerNoteAgregeEleve($elem, $etudiant) * $coefficient;
        }
        elseif ((new RessourceRepository())->verifExistant($compositionAgregation->getIdElement())) {
            $elem = (new RessourceRepository())->recupererParClePrimaire($compositionAgregation->getIdElement());
            $note = RessourceRepository::calculerNoteAgregeEleve($elem, $etudiant) * $coefficient;
        }
        else {
            return null;
        }
        $coefficientTotal += $coefficient;
        $noteTotal += $note;
    }
    if ($coefficientTotal > 0) {
        return round( num: $noteTotal/$coefficientTotal,  precision: 2);
    }
    return null;
}
```

Annexe 8 : classe RessourceRepository - méthode calculerNoteAgregeEleve

```php
public static function calculerNoteAgregeEleve(Ressource $ressource, Etudiant $etudiant) : float
{
    $note = (new NoteRessourceRepository())->getNoteParResssourceEtParEleve($etudiant->getEtudid(), $ressource->getIdRessource());
    return $note;
}
```

Annexes deuxième exemple :

Annexe 1 : Diagramme de classes



Annexe 2 : Code de la fonction getLdapResults()

```php
$ldapServer = ConfigurationLDAP::getServer();
$ldapPort = ConfigurationLDAP::getPort();

$ldapConnection = ldap_connect( uri: "ldap://{$ldapServer}:{$ldapPort}");

ldap_set_option($ldapConnection, option: LDAP_OPT_PROTOCOL_VERSION, value: 3);
$ldapBaseDN = "dc=info,dc=iutmontp,dc=univ-montp2,dc=fr";
$ldapSearch = ldap_search($ldapConnection, $ldapBaseDN, filter: "(objectClass=person)");

return ldap_get_entries($ldapConnection, $ldapSearch);
```