Giulia Bonaspetti Martins
Marceau Decamps
Antoine Rivoire
Maxence Vincent

Rapport final

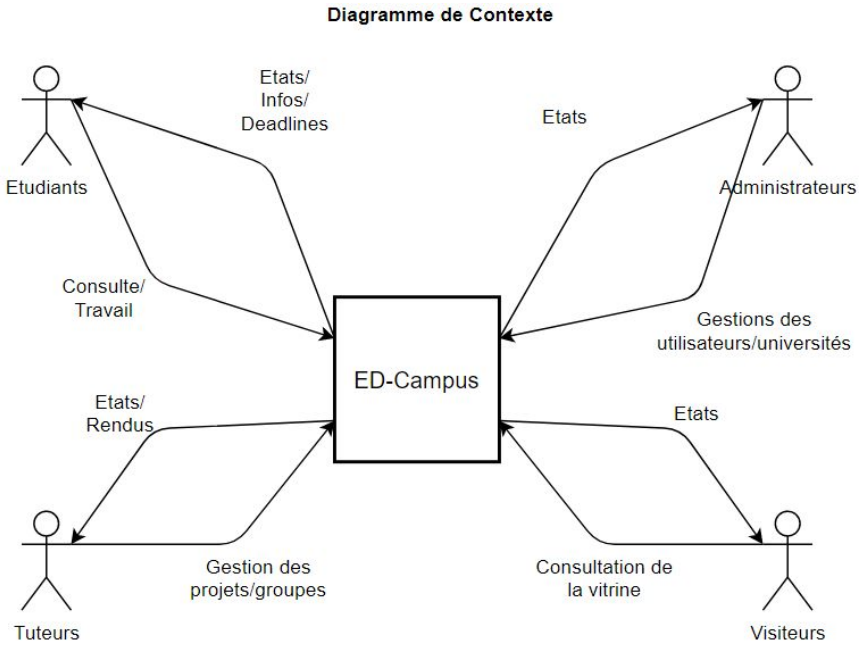
EDCampus

Rappel du sujet/besoin et cahier des charges

La plateforme EDCampus est une solution permettant la gestion de projets pluridisciplinaires complète, pouvant regrouper des entreprises et/ou des étudiants de différentes universités.

Cette plateforme est donc accès principalement sur les projets universitaires et intègre une multitude d'outils afin de gérer des projets et de travailler en groupe. Entre autres, la plateforme propose un outil de gestion des tâches (Kanban, liste, Gantt), une vue calendrier regroupant les différentes *deadlines*, un serveur permettant le dépôt de fichier, un *chat*, un module de gestion de finance et bien d'autre.

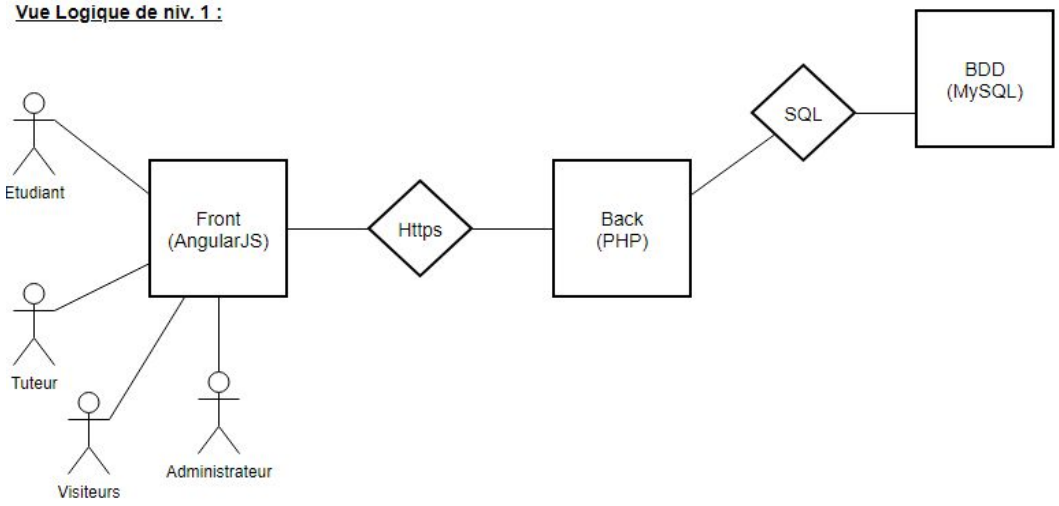
En plus de cette partie principale qui est la gestion et la réalisation de projets, EDCampus intègre aussi un aspect un peu plus "commercial" qui permet aux étudiants/universités de promouvoir leurs travaux auprès des différentes entreprises au travers d'une vitrine de projets. Cet aspect est donc un point essentiel puisqu'il permet aux entreprises de se rendre compte très facilement du travail qui à été réalisé, et donc de savoir qui contacter si des postes sont ouverts.



Cette plateforme est un projet *open-source* de Disrupt Campus. Les principaux contributeurs étant des étudiants, notre travail s'est accès sur la maintenance du site avec principalement des corrections de *bugs* et des améliorations techniques. Notre rôle était donc d'intégrer le projet comme si nous étions en entreprise et de réaliser les différentes tâches qui nous étaient données pour chaque *sprint* (application de la méthode agile, voir plus bas).

Architecture techniques et technologies employées

L'architecture utilisée pour ce projet est une architecture 3-tiers, un classique pour les applications web. Ainsi, nous retrouvons de l'AngularJS sur la partie *front*, du php en *back* et finalement du MySQL pour la base de donnée.



Réalisations techniques

Nous avons participé à un projet collaboratif, qui était déjà en cours. Les tâches réalisés ont été définis à chaque sprint par Anthony Geourjon (ingénieur à temps plein qui travail dans le projet et notre *Scrum Master*). Elles étaient surtout choisis par priorité. Priorité définis par les demandes utilisateurs et des clients potentielles. Ainsi, nous nous occupions d'abord des tâches critiques. De nombreuses tâches ont été réalisés, et elles étaient globalement très indépendantes. Cela a permis de facilement se répartir le travail. Parmi les tâches accomplies, celles qui sont les plus significatives :

Correctif de bug

C'était des bugs plus ou moins critique. Ils pouvaient toucher à la fois au *back* ou au *front*, voire les deux. La durée de la tâches dépendait également de la difficultée. Globalement tout le monde a eu ce genre de tâche. Mais c'est surtout Giulia qui s'est occupé des bugs liées au *front*.

Amélioration et correction de la vitrine

C'était une tâches assez importante car la vitrine n'était pas encore achevée et quelques points ne fonctionnaient pas. L'amélioration de cette partie était particulièrement demandée par un

client. C'est notamment Marceau qui s'est occupé de cette partie en apportant des correctif *front* et *back*.

Amélioration et correction du *chat*

Lorsque nous sommes arrivés dans le projet, cette tâches était quasiment fonctionnel et utilisable mais de gros *bugs* empêchaient de l'utiliser. C'était une tâche qui était très demandée par de nombreux utilisateurs car le chat permet d'utiliser pleinement EDCampus, et donc d'utiliser un seul et unique outil centralisateur. C'est Maxence qui s'est occupé de cette tâche et a permis l'utilisabilité de celui-ci.

Téléchargement d'un zip des livrables

Il s'agit d'une nouvelle fonctionnalité. C'est notamment la seule fonctionnalité qui a été réalisé entièrement, de l'élaboration du cahier des charges au test. Le but de cette fonctionnalité est assez simple, il permet de télécharger pour un enseignant tous les livrables rendus par des élèves dans un même projet en un seul clic, et donc d'avoir un gain de temps. C'était une demande par les enseignants utilisateurs de EDCampus. C'était une tâches assez complexe car elle demandait des notions dans beaucoup de domaines :

- Le *back* et le *front*.
- Sur la gestion de base de donnée.
- La création d'archive zip.
- La gestion de fichier.

Vérification et de formatage automatique de code (CI)

Tâche réalisée à la fin du projet par Giulia. Cette tâche a été demandée pour avoir un code de meilleure qualité pour les prochaines contributions et une diminution de la dette technique de la plate-forme. C'était l'ajout d'un script lancer automatiquement à chaque commit pour la vérification et la correction du formatage du code.

Gestion de projet

Planning

Nous avons fonctionné par *sprint* de une semaine avec la méthode *Scrum*. Notre *Scrum Master* était Anthony, nous avions une réunion avec lui à chaque début de semaine. Le but de celle-ci était de discuter des tâches du *sprint* précédent avec un tour de table, les difficultés rencontrées ainsi que les tâches réalisées. Chaque *sprint* était évalué et la quantité de travail était bien estimée. Cependant, deux tâches ont été sous estimées et ont pris bien plus de temps que prévu. Cette erreur comprenait la tâche d'Antoine sur les téléchargement de livrable en zip ainsi que celle de Maxence sur le *chat* (expliquées dans les réalisations techniques), mais globalement, le planning était bien respecté. Selon les sprints, nous avions souvent quelques tâches en plus, prévues initialement pour le *sprint* suivant, au cas ou nous étions plus efficace que prévu. Pour finir, le *backlog* et la gestion d'*issue* étaient réalisés sur le board de GitLab, ce qui permettait d'avoir une gestion des *issues* en temps réel.

Méthode de travail

Afin de faciliter la gestion de projet, l'architecture 3-tiers est divisée en 2 *repo* Git, avec le *back* et la base de données d'un côté, et la partie *front* de l'autre côté. Ainsi, il est plus facile de dissocier/diviser les *issues*, et cela facilite grandement l'organisation, notamment pour le *Scrum Master*, mais aussi pour les développeurs qui ont besoin d'ouvrir qu'une partie de l'application pour faire une tâche.

À chaque *sprint*, chaque personne choisissait une tâche et se l'assignait sur le board via GitLab. Il apportait sa modification en s'assurant du bon fonctionnement et en vérifiant qu'il n'y avait pas de régression sur les autres fonctionnalités. Ensuite il faisait une demande de merge request pour intégrer son code, et c'était Anthony qui vérifiait et validait, ou non, la merge request.

Il est assez difficile de résumer nos méthodes de travail car cela était très différent d'une tâche à l'autre :

- Certaine tâches en *front* faisait par exemple plusieurs aller-retour entre développement et validation afin d'être le plus intuitif possible et avoir un bon retour-utilisateur.
- Pour la recherche de bug, nous utilisions beaucoup le debugger côté back de PhpStorm afin de localiser le problème.
- Certaine tâche ont nécessité de la programmation en binôme (pair programming) avec Anthony car nous étions bloqué, et qu'un point de vue extérieur de sa part avec l'apport de son expérience était très bénéfique et pouvait rapidement débloquer la situation.

D'autres habitudes plus générales ont été prises notamment de créer une nouvelle issue à chaque fois qu'un *bug* ou problème était trouvé et qu'il n'avait pas de lien avec la tâche courante.

Rôles des membres

Globalement, chaque personne a eu un rôle de développeur *full stack* (*back* et *front*). Mais au final, chacun d'entre nous a eu un rôle assez différent.

Giulia a corrigé des *bugs* et a apporté des améliorations à divers endroits du site. Elle a aussi réalisé une tâche de DevOps.

Maxence a réalisé toutes les issues liées au *chat*.

Marceau a majoritairement corrigé des *bugs*, mais a aussi ajouté des améliorations, notamment liées à la vitrine.

Antoine a ajouté la fonctionnalité de téléchargement des livrables sous le format zip.

Communication

Un aspect très important dans la gestion de groupe reste la communication. Mais c'est quelque chose qui a été un peu mis de côté vu le contexte de notre projet. En effet, chaque tâche était trop indépendante, il y avait donc peu d'entraide et de travail de groupe. Nous avons tout de même choisi de mettre une réunion journalière à chaque début de journée afin de savoir

l'avancement des autres et donner un autre point de vue en cas de problème.

Cependant nous avons eu une bonne communication avec notre tuteur Anthony et nous le remercions énormément pour cela. Il était bien disponible et à notre écoute. Cela a été une plus-value dans notre projet.

Outils

Les outils que nous utilisons peuvent être divisés en outils de développement et en outils de communication et de gestion de groupe.

- Outils de développement :
    - Visual Studio Code - L'éditeur de code utilisé principalement pour le développement du *frontend*.
    - PhpStorm - L'éditeur de code utilisé principalement pour le développement du *backend*.
    - phpMyAdmin - L'application Web utilisée pour la gestion de base de données MySQL.
    - pre-commit - Le *framework* utilisé pour ajouter un contrôle de la qualité du code. Il fait la gestion des scripts que Git exécute avant un commit, alias pre-commit hooks. Dans le *framework* pre-commit, les hooks utilisés étaient prettier (formateur de code des fichiers JS, CSS, JSON, Markdown et YAML), htmllint (vérificateur de code des fichiers HTML), pre-commit-hooks (formateur de attributs généraux).
- Outils de communication et de gestion de groupe :
    - Slack - Le logiciel utilisé pour la communication entre les membres du groupe ainsi qu'avec le responsable du projet.
    - GitLab Gricad - La plate-forme utilisée pour le travail collaboratif et la gestion de tâches.

Métriques logiciels

EDCampus compte actuellement environ 1.200 utilisateurs et plus de 300 projets. Son utilisation est en majorité via les ordinateurs, mais il y a aussi des utilisations via les appareils mobiles. Depuis 14 mois, 40 étudiants et un ingénieur à temps plein ont contribué à la plateforme.

Nous pouvons calculer, par exemple, le coût que notre groupe de 4 développeurs aurait eu pour le projet, en tenant compte du fait que nous travaillons officiellement 21 jours. Nous ne compterons pas les locaux car l'université nous offre la place et le calcul deviendrait trop compliqué. Le salaire moyen d'un développeur est de 2.200€ net/mois, ce qui devient 3.900€/mois en comptant les charges patronales, soit 185€/jour/personne. Nous avons ajouter au calcul l'essence que Maxence a dépenser pour rejoindre notre tuteur dans ses bureaux, ainsi que l'amortissement du matériel d'Antoine qui à moins d'un an. Tout cela nous amène à un total de 15.572,94€ sur notre période de projet.

Dès le début de ce projet, 2 jours de dette technique ont été ajoutés. Cette dette est principalement due à deux facteurs :

- Nous travaillons sur un grand projet qui n'a pas de tests d'intégration, de tests unitaires, ou d'autres formes de contrôle en dehors de la révision du code et des tests manuels effectués par notre Anthony ou par nous-mêmes.
- Comme nous avons travaillé avec un vrai client et en utilisant des méthodes agiles, il était parfois plus important pour le client que le service soit fait rapidement et non que le code soit fait avec rigueur.

Notre groupe a participé à 35 *issues* au total, dont 28 ont déjà été fusionnées dans le code principal et 7 sont en cours de révision.

Étant donné que chaque *issue* présente un niveau de difficulté différent, le pourcentage des *issues* que chacun a fait n'est pas aussi significatif qu'un échantillon du travail effectué. Souvent, une *issue* complexe prend autant de temps que d'innombrables *issues* simples. Cela dit, la répartition des *issues* était la suivante :

- Antoine Rivoire - 8,8%
- Giulia Bonaspetti Martins - 38,2%
- Marceau Decamps - 41,2%
- Maxence Vincent - 11,8%

Nous comptons également le nombre de lignes de code modifiées par chaque développeur selon les chiffres donnés comme étant différents du code original par GitLab au moment du merge.

Ce nombre n'est pas non plus une indication de la quantité de travail effectuée, car il ne tient pas compte de la complexité du contenu et, comme le projet n'avait pas de formateur de code ou de règles de codage à suivre, la façon dont la personne écrit le code influence également sur ce nombre, parfois l'éditeur de code lui-même peut contenir des paramètres pour "ajouter une nouvelle ligne à la fin du fichier" ou "supprimer les espaces restants à la fin de la ligne" automatiquement si le fichier ne l'a pas :

- Antoine Rivoire - 271 lignes de code ajoutées et 61 supprimées parmi les *issues* terminées et actuellement en attentes de révision.
- Giulia Bonaspetti Martins - 1137 lignes de code ajoutées et 1.882 supprimées parmi les *issues* terminées et actuellement en attentes de révision. Pour l'*issue* de vérification et de formatage automatique de code nous ne pouvons pas prendre en compte les valeurs données par GitLab, car cette issue modifie automatiquement presque tous les fichiers du projet.
- Marceau Decamps - 979 lignes de code ajoutées et 631 supprimées parmi les *issues* terminées et actuellement en attentes de révision. D'autres lignes de code ont été ajoutées dans un fichier CSS mais ne sont pas cohérente dû à l'organisation de ce dernier.
- Maxence Vincent - 75 lignes de code ajoutées et 32 supprimées parmi les *issues* terminées et actuellement en attentes de révision.

Conclusion

Pour conclure le projet, nous avons organisés une **rétrospective** au sein de notre équipe. Cette rétrospective avait pour but de définir ce qui a bien marché et ce qui a moins bien marché pendant ce projet.

- Points positifs :

  Les points positifs sont **très nombreux**.

  - Le projet était en lui même très intéressant. C'est un projet complet, qui nous a fait découvrir de nouveaux langages et outils (php, JavaScript, PhpStorm, phpMyAdmin, …) et nous a permis de mettre en application nos connaissances. Le projet est également concret, notre travail a été utile. Il y a de vrais utilisateurs et donc une demande en temps réel.
  - De plus, le projet étant en *open-source*, cela peut nous servir de portfolio.
  - Notre encadrant et *Scrum Master* (Anthony Geourjon) était très investi et disponible. Les tâches était bien défini ce qui rendait les rôles clairs et précis. On ne perdait donc pas de temps pour l'organisation. Le suivi qu'il nous assurait était de qualité et nous avons même travaillé en programmation en binôme (pair programming) avec lui dans les cas les plus extrêmes.
  - Nous avions une grande liberté d'horaires : peu importe les horaires de travail, tant que le travail est fait dans les temps.
  - Pour terminer, nous jugeons que le travail que nous avons accompli a été de qualité.

- Points négatifs :

  Ils sont **bien moins nombreux** que les points positifs et en sont leur **conséquences directes**.

  - Les langages utilisés dans ce projet nous était inconnu au début du projet, ce qui nous a fait perdre du temps à la prise en main.
  - Nos tâches était si bien découpées que nous avions peu de travail en commun à proprement parler. Nous avions remédier à ce soucis en organisant un daily-meeting entre nous chaque jour.
  - Comme le projet est collaboratif, il y a un manque de documentation sur l'existant et le code n'est pas très propre à certains endroits.

Glossaire

Back/Backend : Partie de traitement.
Backlog : Liste des tâches.
Bug : Problème de fonctionnement.
Chat : Fenêtre de discussion.
Deadline : Date de rendu maximale.
Framework : Abstraction qui unit des codes communs entre divers projets de logiciels fournissant des fonctionnalités génériques.

Front/Frontend : Partie de présentation.
Full stack : Développeur travaillait sur le *front* et le *back*.
Issue : Tâche à réaliser lors d'un sprint.
Open-Source : Le code source est public et partagé pour permettre à tout le monde de contribuer.
Repo : Du terme "repository", un répertoire/fichier.
Scrum : Méthodologie agile pour la gestion et la planification de projets.
Scrum Master : Personne qui organise les sprint notamment.
Sprint : Période entre deux réunion avec un certain nombre de tâche à réaliser.

Bibliographie

Disrupt Campus Université Grenoble Alpes : https://disrupt-campus.univ-grenoble-alpes.fr/
EDCampus : https://gricad-gitlab.univ-grenoble-alpes.fr/edcampus
GitLab Gricad : https://gricad-gitlab.univ-grenoble-alpes.fr/
htmllint : https://github.com/Lucas-C/pre-commit-hooks-nodejs
PhpStorm : https://www.jetbrains.com/phpstorm/
phpMyAdmin : https://www.phpmyadmin.net/
pre-commit : https://pre-commit.com/
prettier : https://prettier.io/
pre-commit-hooks : https://github.com/pre-commit/pre-commit-hooks
Slack : https://slack.com/
Visual Studio Code : https://code.visualstudio.com/