Résumé du rapport du TPI

Situation de départ

Le but du projet est de développer une application web avec Laravel de publication de podcasts. Pour les auteurs, il doit être possible de créer et modifier leurs podcasts, et créer, éditer et supprimer des épisodes dans leurs podcasts. Les épisodes doivent pouvoir être publiés dans le futur et caché par l'auteur si besoin. Le projet est parti de rien (il ne s'appuie pas sur un autre projet). J'ai choisi d'appeler l'application Podz. Les critères spécifiques demandaient de faire une modélisation des données pertinentes, de respecter les principes du modèle MVC, d'avoir une interface utilisateur propre et utilisable. Il était aussi demandé de suivre les normes d'écriture de code, d'utiliser un système de versionning en faisant des petits commits atomiques et fréquents. Les épisodes devaient aussi être correctement écoutables dans les navigateurs (Firefox, Edge et Chrome). Le temps à disposition est de 90h réparties sur 4 semaines.

Mise en oeuvre

En plus de l'utilisation du framework Laravel, j'y ai ajouté Livewire, AlpineJS et TailwindCSS. Ces 4 frameworks que j'avais utilisé en stage et pour des projets personnels forment la stack TALL et sont régulièrement utilisé dans l'écosystème Laravel. Pour ne pas avoir à développer la connexion et la création de compte, j'ai utilisé le starter kit Jetstream qui mettait déjà tout en place. J'ai fait le MCD et MLD de ma base de données. J'ai réfléchi aux différentes pages nécessaires pour utiliser les fonctionnalités requises et j'ai fait des maquettes pour chacune des pages. La page de détails d'un podcast a en fait plusieurs vues, selon si l'on est visiteur ou auteur, et en tant qu'auteur on peut ouvrir ou fermer les formulaires pour modifier des épisodes ou les informations du podcast. Une fois cette analyse terminée, j'ai développé une après l'autre toutes les fonctionnalités demandées, tout en suivant ma planification. J'ai eu un peu d'avance au départ sur le premier sprint (j'ai avancé une tâche du sprint 2 au sprint 1) puis comme la création d'épisode avec l'upload de fichiers était plus complexe que je l'imaginais, j'ai eu un peu de retard sur mon planning, mais j'ai réussi à rattraper le retard à la fin et tout finir dans les temps. La particularité de mon TPI par rapport à d'autres TPI en développement, c'est que j'ai écrit de nombreux tests automatisés pour m'assurer que la majeure partie du comportement de mon application était correct et restait fonctionnel tout le long du projet. J'ai écrit des tests fonctionnels et parfois unitaires et j'ai utilisé PHPUnit pour écrire ces tests. L'écriture a pris un peu de temps tout au long du projet, mais ils ont permis d'accélerer la validation du fonctionnement, j'ai ainsi pu éviter beaucoup d'essais à la main puisque j'avais confiance sur le fait que mon backend fonctionne. Il restait bien sûr à s'assurer que tout fonctionne comme prévu dans mon navigateur mais cela était plus rapide à déterminer.

Résultats

Toutes les fonctionnalités demandées ont pu être implémentées et testées. La création d'un podcast se fait sur une page dédiée, tandis que l'édition d'un podcast, la création, modification et suppression d'épisodes se font toutes dans la même page Détails d'un podcast. Je n'ai pas eu de difficultés particulières à designer mon application, je n'ai pas eu besoin d'utiliser de template. Les points spécifiques demandés ont été respectés. 44 tests ont été écrits et ils passent tous à la fin du projet.