Année universitaire 2022-2023

SAÉ Semestre 3 :

Dossier de programmation

Wittmann Grégory

Rouillon Tom

UNIVERSITÉ DE LORRAINE

IUT Saint-Dié

Sommaire :

Introduction :

Ce dossier de programmation présentera 2 volets différents, l'un sur la partie technique et l'autre sur la partie gestion de projet.

Dans chacun de ces volets, nous présenterons notre point de vue sur l'application finale par rapport à nos attentes de départ, et aussi par rapport aux différents rendus sur la conception ou encore sur le planning.

Pour remettre en contexte, le projet était de réaliser une application web pour un club de sport, dans notre cas un club de karting, leur permettant de gérer un planning de compétitions, de gérer les adhérents, c'est-à-dire permettre aux utilisateurs de s'inscrire à des compétitions, de réserver des circuits, de devenir adhérent au club…

Ce projet était divisé en 3 modules :

- Un en java, permettant aux admins de créer et de gérer le planning
- Les 2 autres en html/css/php/js, permettant quant à eux de gérer les utilisateurs, donc de gérer ce qui a été évoqué ci-dessus

Dans le reste de ce dossier, le mot **"Utilisateur"** désignera toute personne connectée à Internet et donc ayant un accès au site web.

Le mot **"Adhérent"** désignera tout utilisateur ayant créé un compte sur le site web. Un adhérent est donc aussi un utilisateur.

Partie technique :

État d'avancement

Dans cette partie, nous verrons ce qui fonctionne, ce qui reste à faire et ce qui n'a pas été commencé sur notre application. Nous nous appuierons sur le Cahier des Charges pour comparer le rendu final et ce qui avait été pensé à la base, mais aussi sur le fichier « Éléments imposés ».

Tout d'abord, les principales fonctions de l'application fonctionnent. Nous incluons dans ce lot :

- L'inscription pour permettre aux utilisateurs de devenir adhérent
- La connexion sécurisée pour les adhérents à leur compte
- L'espace membre, permettant aux adhérents de consulter leurs informations, de changer leur photo de profil et d'ajouter un document administratif pour le club
- L'espace réservé aux adhérents leur permettant de réserver un circuit en indiquant le jour, l'heure de début et l'heure de fin, le circuit qu'ils veulent, ainsi que le nombre de participants
- La page d'accueil présentant rapidement le club, et présentant quelques photos
- La rubrique consacrée aux événements, montrant tous les matchs à venir, les matchs passés ainsi que le gagnant de ces matchs, un bouton pour s'inscrire si l'utilisateur est connecté (ou se désinscrire)
- La rubrique consacrée aux articles, permettant aux administrateurs d'ajouter des articles sur les matchs ou sur d'autres sujets, et permettant aux utilisateurs de les lire
- La rubrique mettant en avant les partenaires du club, avec leur site web ainsi que leurs réseaux sociaux s'ils en ont
- L'application java permettant aux administrateurs de créer un match, de modifier ou supprimer un match déjà existant, de gérer les demandes de réservations de circuit

Concernant la connexion sécurisée, plusieurs systèmes de sécurité ont été mis en place. Par exemple, les mots de passe sont hashés avant d'être insérés dans

la base de données, impliquant qu'il sera très difficile pour quiconque de récupérer le mot de passe en clair, avec uniquement le mot de passe hashé.

De plus, nous utilisons la technique du salage, permettant de rendre encore plus complexe le déchiffrement du mot de passe.

Maintenant, listons ce qu'il reste à faire et ce qui n'a pas été commencé :

- Lors de l'inscription, l'utilisateur est directement ajouté à la base de données, après vérification des informations. Il n'y a donc pas d'envoi de mail à un administrateur lui permettant d'accepter ou non les demandes
- Les articles n'ont pas d'images associées, il n'est donc pas possible de lier une image à un article
- Il n'est pas possible de télécharger le planning en pdf à partir du site web
- Il y a la possibilité pour chaque adhérent de choisir un niveau de pratique, mais cela ne change pas la gestion des inscriptions
- La réservation d'un circuit par un adhérent ne peut être faite qu'à partir du site
- L'ajout de match ne prend pas en compte les matchs sur plusieurs jours

Description de l'architecture

Dans cette partie, nous allons présenter les 3 couches de l'architecture : présentation, métier et données.

Présentation

Concernant le module java, c'est la bibliothèque Swing que nous avons utilisée pour créer l'IHM. Elle permet d'ajouter facilement des boutons, des zones de texte, etc… permettant une bonne ergonomie pour l'utilisateur du module.

Cependant, nous n'avons pas utilisé le modèle MVC, ce serait donc une piste d'amélioration pour avoir un code plus simple à gérer.

Concernant le module web, nous avons créé la presque totalité des fichiers css pour définir le style global de notre site, et nous avons utilisé Bootstrap pour l'affichage des formulaires d'inscription et de connexion.

Métier

Pour cette couche, nous allons présenter les classes permettant de gérer les processus "métier", c'est-à-dire dans notre cas la création du planning.

Pour pouvoir gérer entièrement un planning, il nous a fallu créer différentes classes java, comme la classe Match, représentant un match avec une date, une heure de début, une heure de fin, un circuit et le nombre de participants nécessaires.

Il y a donc aussi la classe Circuit, représentant un circuit avec son nom, son adresse et le nombre de places maximales.

Enfin, on peut évoquer la classe Planning regroupant tous les matchs. Cette classe gère tous les conflits qu'il peut y avoir quand un administrateur veut ajouter un match ou en modifier un. Par exemple, si on veut ajouter un match alors qu'un autre est déjà présent à la même date et aux mêmes horaires, un message d'erreur est affiché et le match n'est pas ajouté. Pareil si on veut modifier les horaires d'un match. Cependant, on peut évidemment ajouter un match aux même horaires qu'un autre sur un circuit différent.

De plus, il est impossible de créer un match si une réservation est déjà présente. Pareillement, il est impossible d'ajouter une réservation si un match est déjà présent.

Ensuite, il est impossible de modifier les horaires ou le lieu d'un match qui est déjà passé, mais on peut modifier le gagnant du match. Et au contraire, il est impossible de modifier le gagnant pour un match qui n'est pas encore passé.

Enfin, sur le site web, donc là où les adhérents peuvent s'inscrire aux matchs, s'il y a déjà assez de participants, plus personne ne peut s'inscrire à ce match, à moins qu'un adhérent se désiste.

Données

Finalement, pour cette dernière couche, nous allons parler de la base de données, quels choix ont été fait et les difficultés rencontrées.

Premièrement, étant donné que les différents modules devaient avoir des données communes, comme les matchs : les gérer avec le module java et les afficher et gérer les inscriptions des utilisateurs avec le module web, il a fallu créer des tables

contenant toutes les informations nécessaires aux 3 modules. Par exemple, la table match est commune aux 3 modules, mais la table personne n'est utilisée que par le module web. Il a donc fallu relier les 2 par le biais de la table match_has_adherent, permettant de gérer les inscriptions aux matchs, avec des clés étrangères : une sur l'id du match et l'autre sur l'id des adhérents.

Ensuite, pour éviter la redondance des données, nous avons créé une table personne, contenant les informations basiques sur une personne (son nom, prénom, mail…) puis nous avons créé une table adhérent, précisant quant à elle le niveau de pratique, la date de dernière visite…, et une table admin permettant au module web de savoir si un utilisateur est un administrateur ou non. Pour relier ces tables, il y a une clé étrangère dans les tables admin et adhérent, sur l'id contenu dans la table personne. Ainsi, nous évitons d'avoir des données répétées dans la base de données.

Finalement, dans le module web, nous avons créé un fichier à part permettant de se connecter à la base de données. Il est donc facile de changer à quelle base se connecter. Pour le module Java, il n'y a pas de classe dédiée, mais l'url de la base et les identifiants sont des attributs de classe, permettant également de changer aisément l'url ou les identifiants.

Retour à l'analyse

Dans cette partie, nous allons reprendre le dossier d'analyse et de conception pour comparer les idées originales, et le rendu final. Dans ce dossier, nous avions réalisé de multiples diagrammes UML : diagrammes de cas d'utilisation, d'activité, de scénarios, de séquence et de classe.

Ce sont ces derniers qui ont été les plus utiles, puisqu'ils nous ont permis de coder directement les classes java, et qu'il y avait donc juste à les traduire.

Cependant, il y a quand même eu des modifications de ces diagrammes au cours de la programmation. En effet, il y avait certains éléments qui nous avaient échappés lors de la conception, et il a donc fallu les prendre en compte. Par exemple, pour éviter d'avoir un match et une réservation qui se chevauchent, il a fallu ajouter une nouvelle fonction qui cherche dans la base de données toutes les réservations et vérifier s'il y avait conflit.

De plus, lors de la réalisation, on s'est rendu compte qu'il était plus simple et plus efficace de stocker l'id de l'adhérent directement dans la classe pour avoir accès à leurs informations sans avoir à repasser par la base de données.

Ensuite, lors de l'analyse, nous voulions implémenter une boutique avec un panier et une page de paiement. Or, lors de la réalisation, il s'est avéré que l'on n'avait pas le temps d'implémenter ces fonctionnalités.

On peut aussi ajouter le fait que dans les diagrammes d'activités, les administrateurs avaient beaucoup de tâches à accomplir, comme gérer manuellement toutes les inscriptions, les réservations, les inscriptions… Il s'est avéré qu'il était plus simple d'automatiser certaines tâches, comme les inscriptions qui sont directement ajoutées à la base de données.

À part ces quelques différences, nous avons réussi à respecter les exigences du cahier des charges, ainsi que les idées originales présentes dans le dossier d'analyse et de conception.

Finalement, ce dossier nous a permis de gagner du temps sur la réalisation, même s'il n'était pas complet et que nous avons dû faire quelques modifications.

« Si c'était à refaire »

Généralement, après avoir accompli un projet, il est important d'analyser ce qui a été fait et de se demander quelles sont les choses à changer si l'application devait être refaite. C'est ce que nous allons faire dans cette partie.

Concernant ce projet, les éléments à revoir seraient les suivants :

- Pour le module java, se renseigner davantage sur les bibliothèques préexistantes. Par exemple, pour la gestion du planning et des disponibilités, la première version de cette fonction utilisait le type String, et les vérifications étaient réalisées sur ces String. Cependant, au cours du développement, nous avons découvert l'existence de la bibliothèque LocalTime, permettant d'utiliser la fonction parse(String) permettant de transformer un String en LocalTime, ainsi que les fonctions isAfter(LocalTime) et isBefore(LocalTime) permettant de facilement déterminer si 2 horaires se chevauchent.

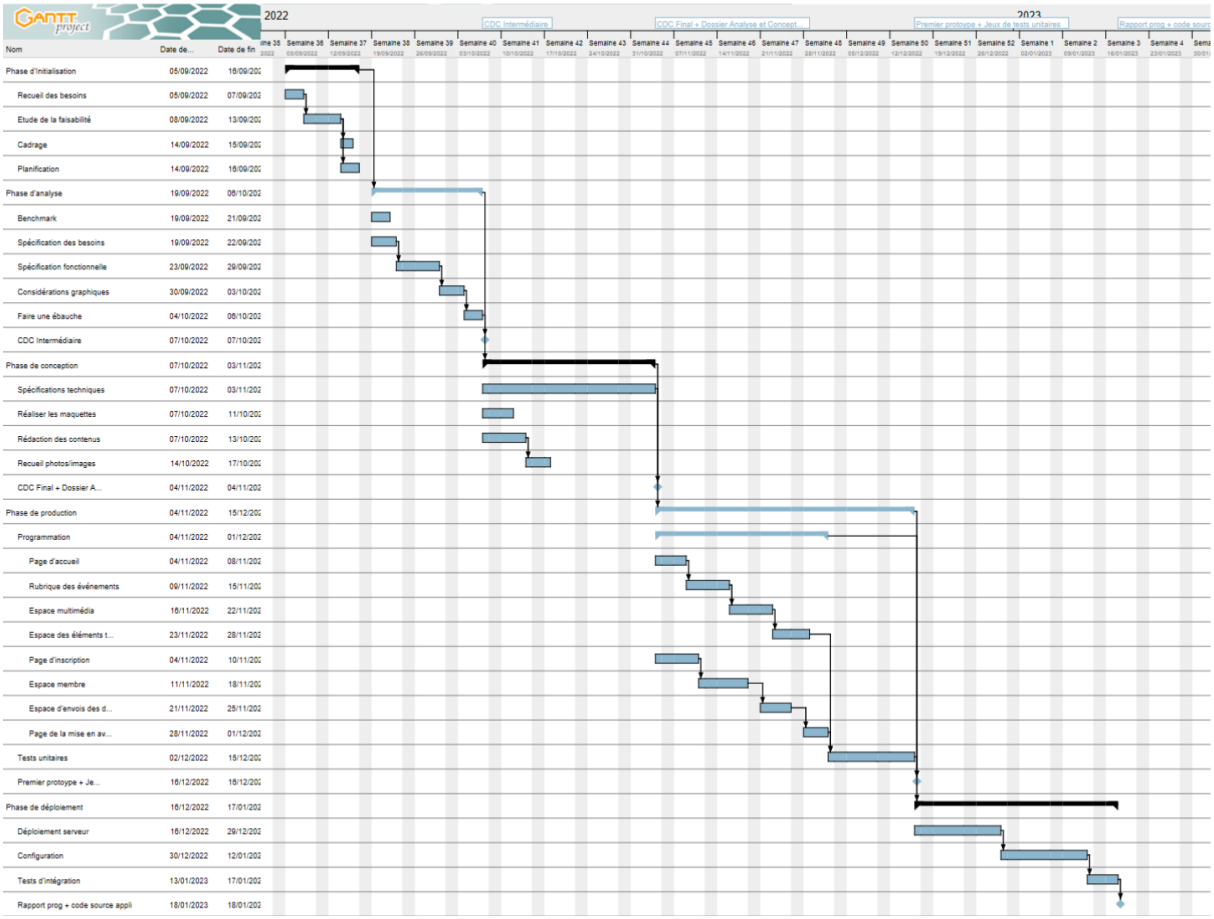
- Pour l'IHM, développer un modèle basé sur MVC (Modèle, vue, contrôleur). Cela aurait permis d'avoir un code plus simple à comprendre et à modifier.
- Pour les modules web, utiliser directement JQuery et ajax pour avoir un meilleur résultat lors de la gestion des erreurs. En effet, au début du développement, nous avions utilisé uniquement le php pour gérer le back-end. Cependant, il était difficile de gérer les messages d'erreurs. Par exemple, une des solutions que nous avions mise en place consistait à envoyer les erreurs via l'url et de les récupérer avec la variable $_GET. Le problème était qu'il était plus fastidieux d'appliquer cette solution lorsqu'il y avait beaucoup d'erreurs différentes. Ajax nous a permis de gérer plus élégamment ces messages. Il nous a donc fallu transformer l'ancien code pour l'adapter à JQuery et ajax, ce qui nous a fait perdre du temps.

En conclusion pour cette partie, même si certains points nous ont fait perdre du temps, nous sommes satisfaits de notre application et, à part les points cités ci-dessus, nous referions pareil si c'était à refaire.

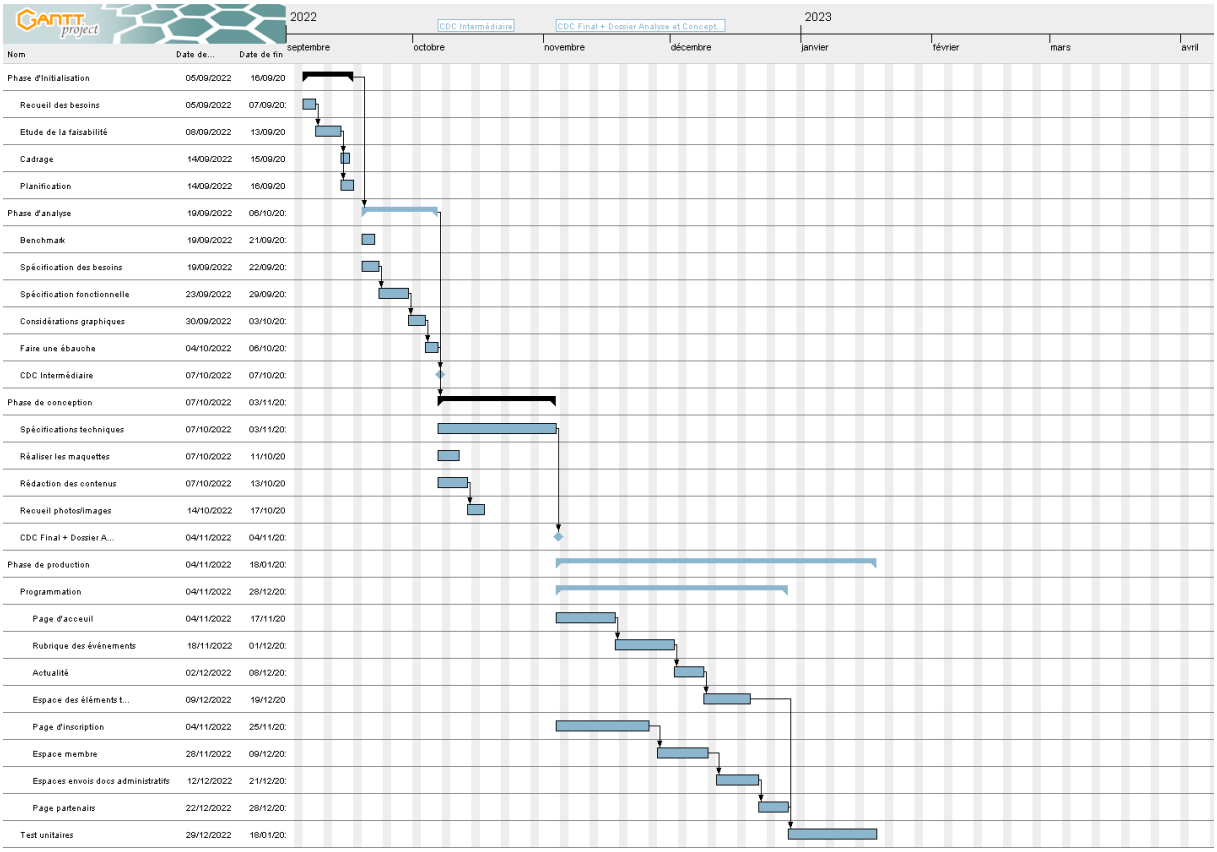
Partie gestion de projet

Planning des tâches

Au début du projet, nous avions rédigé un planning prévisionnel que voici :



Maintenant, voici le planning réel :



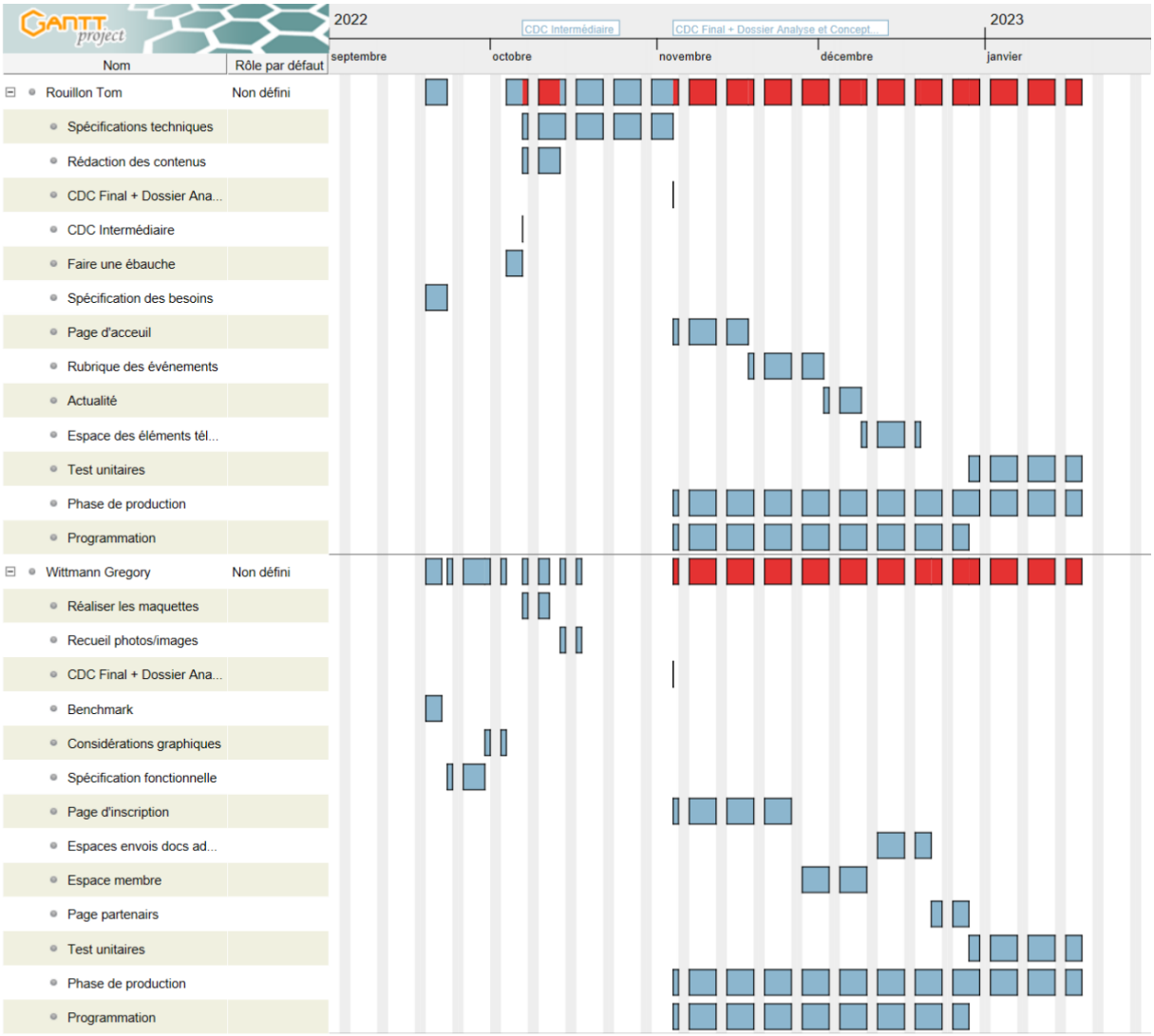
Les principales différences sont que la phase de programmation a pris plus de temps que prévu, mais surtout que la phase de déploiement a été ignoré. En effet, nous n'avons pas eu la possibilité de déployer notre serveur et notre base de données, nous n'avons pu que travailler en local.

C'est aussi pour cette raison que nous avons pu prendre plus de temps pour la phase de programmation.

Cela nous donne ainsi une meilleure idée du temps de chacune des tâches pour de futurs projets.

Partage des tâches

Voici ci-dessous le partage des tâches réel :



Nous remarquons donc que la plupart du temps, nous nous sommes réparti équitablement les tâches, pour que chacun ait environ la même charge de travail.

Gestion des versions

Comme tout projet se fait majoritairement en groupe, il est essentiel de partager le travail et il est essentiel que tout le monde travaille sur la même version. Il est donc important d'employer un système de partage en temps réel des fichiers.

Pour notre cas nous avons utilisé Git pour partager l'avancement à travers les commits. Ce qui est très pratique car nous pouvons retourner en arrière si le projet venait à être dans une impasse, en plus d'archiver toutes les modifications apportées au projet.

Protocole de tests

Afin que notre projet, plus particulièrement la partie programmation, soit qualitatif et fonctionnel dans tous les cas, nous avons donc établi plusieurs tests. Nous avons utilisé les tests unitaires, et dans notre cas, le framework JUnit5.

Ces tests sont surtout basés sur la classe « Planning.java » qui gère une importante partie des fonctionnalités de notre site internet.

Il y a au total 10 fonctions qui ont été testées :

- La fonction permettant d'ajouter un match avec un jour différent mais un même horaire afin de constater si le match prend bien en compte la hiérarchie jour puis heure.
- La fonction permettant d'éviter d'ajouter un match avec des heures incohérentes, par exemple un match qui commence à 12h00 et se finit à 11h00 afin de gérer les erreurs de frappe.
- La fonction permettant d'éviter d'ajouter un match en plein milieu d'un autre match, par exemple le match 1 qui est de 10h à 12h et le match 2 qui est de 11h à 13h afin d'éviter les collisions de match.
- La fonction permettant d'ajouter 2 matchs aux mêmes horaires mais sur un circuit différent afin de vérifier que le paramètre « circuit » soit pris en compte.
- La fonction permettant d'éviter d'ajouter un match sur un circuit qui n'existe pas afin de ne pas avoir de réservation sur un circuit inexistant.
- La fonction permettant de modifier la date et les horaires d'un match afin de vérifier que le match modifié prenne les valeurs voulues.
- La fonction permettant d'éviter de modifier les horaires d'un match sur les horaires d'un autre match déjà présent.
- La fonction permettant d'éviter de déplacer un match entre un autre match.

- La fonction permettant d'ajouter un match entre 2 matchs dans le cas où tout est bon, cela permet de voir si notre fonction fonctionne correctement.
- La fonction permettant d'éviter d'ajouter un match entre 2 matchs où un match y serait déjà présent.

En plus de ces tests, que l'on activait à chaque avancée majeure, nous nous sommes nous-mêmes mis dans la peau des utilisateurs potentiels pour juger l'interface, l'intuitivité des commandes et la robustesse des fonctions. Cela nous a permis de trouver quelques bugs qui n'étaient pas pris en compte dans les tests unitaires.

Bilan personnel

Grégory :

Premièrement j'ai trouvé ce projet très positif du fait que j'étais avec un bon partenaire. Ce projet était un défi à cause de sa longueur. Mais au fur et à mesure qu'on avance dans celui-ci, on se rend vite compte de la charge de travail qui peut faire peur.

Ce que je trouve de bien dans ce projet c'est qu'il n'y avait pas trop de contraintes et que le design était libre.

Mais le point négatif a été les heures prévues pour cette SAé, où certains enseignants faisaient cours sur ces heures, ce qui nous empêchait de travailler sur le projet.

Ce projet m'a aussi permis de me rendre compte de mes lacunes ainsi que mes atouts et c'était surtout le bon moment de revoir ce que l'on avait appris en cours et de l'appliquer.

Tom :

Pour ma part, j'ai trouvé ce projet satisfaisant, dans le sens où la majorité des connaissances accumulées au cours de ce BUT ont été mise à contribution, et cela

me procurait un sentiment d’accomplissement, au fur et à mesure que le projet se construisait.

Concernant les difficultés rencontrées, la principale était la charge de travail important comparée aux autres SAé. Nous avons donc dû partager le travail efficacement pour réussir à développer l’application dans le temps imparti. Cela a été réellement compliqué, notamment sur la fin où de plus en plus de rendu étaient demandés.

Finalement, ce projet a été bénéfique, puisque cela m’a apporté des connaissances supplémentaires sur les bonnes pratiques à adopter lors du développement d’une application web.