VANDAL Jade
SOUCHON Loïc
TRESTOUR Grégory
THOMAS Antoine
INFO5
2019-2020

Rapport final

I. Rappel du sujet

Pour rappel le guc voile c'est une association de planche à voile située au lac de Monteynard à environ 1h en voiture de Grenoble.
Là bas, se trouve un local qui permet de stocker le matériel de planche à voile. Il y a également à l'intérieur un ordinateur avec connexion internet. Cependant, à l'extérieur de ce local et sur l'ensemble du lac, il n'y a pas de connexion internet. C'était donc une problématique à prendre en compte lors de la phase de conception et la définition des besoins.

Différentes personnes seront amenées à utiliser les solutions que nous allons développer :

Anthony	**Le bureau**	**Les ouvreurs**	**Les pratiquants**
Le président du club	Composé d'une petite dizaine de membres, ce sont les personnes qui gèrent le club (trésorerie, secrétaire…)	Des personnes expérimentées en planche, connaissant le code d'accès au local et pouvant ouvrir des permanences (créneaux horaires où les pratiquants peuvent venir récupérer/rendre du matériel qu'ils empruntent)	Toutes les personnes inscrites au club et pratiquant ce sport

Bien que nous avions déjà réalisé notre projet d'ECOM sur ce sujet, nous avons décidé de repartir de zéro pour mieux prendre en compte les besoins réels du client (Anthony), nous avons donc organisé plusieurs réunions afin de discuter avec lui de ses besoins durant la phase de conception, qui a duré environ 2 semaines.

Au cours de ces réunions, voici les besoins que nous avons pu extraire :
- Pouvoir mieux s'organiser entre ouvreurs et pratiquants lors de l'ouverture et fermeture de permanence
- Pouvoir mieux s'organiser sur les trajets pour se rendre au lac depuis Grenoble
- Pouvoir obtenir le code du local facilement en tant qu'ouvreur (le code étant amené à changer régulièrement)
- Pouvoir gérer le matériel de planche à voile
- Pouvoir gérer la liste des pratiquants
- Pouvoir réserver / rendre du matériel lorsque l'on est sur place
- Pouvoir indiquer des dommages sur le matériel après sa sortie
- Pouvoir accéder à des informations spéciales, sous la forme de statistiques, qui pourront servir à mieux gérer le club dans sa globalité

Les solutions que nous avons donc décidé d'apporter sont les suivantes :

- **Une application web :**
  - avec affichage des permanences sur un calendrier facilement importable sur le propre calendrier personnel des pratiquants
  - une fonctionnalité de création de permanences (ouverture + fermeture) accessible seulement par les ouvreurs (donc besoin d'un système d'identification mais sans mot de passe)
  - un système de covoiturage accessible par tous afin de pouvoir s'organiser sur les trajets à destination du lac, tout en faisant attention à la sécurisation des informations de contact qui ne doivent pas pouvoir être récupérées facilement
  - une gestion d'envoi d'e-mail pour le système de covoiturage et de permanences, afin de récupérer des informations ou de recevoir des messages de confirmation lorsque les actions se sont bien déroulées
  - une page accessible que par les membres du bureau pour pouvoir indiquer le code du local, le mettre à jour, cela sert notamment pour l'envoyer aux ouvreurs lorsqu'ils créent des permanences
  - un système de gestion de matériel (voile, planche, harnais, combinaison) et de listes de pratiquants, ouvreurs, bureau, etc accessible bien sûr que par les membres du bureau

  Le but de cette application est qu'elle puisse être accessible sur ordinateur ET téléphone portable n'importe où, celle-ci est donc responsive.

- **Un logiciel :**

  - permettant aux pratiquants de s'identifier (sans mot de passe) en choisissant leur prénom et nom avec une méthode d'auto-complétion
  - permettant de réserver du matériel de planche à voile (planche + voile et harnais + combinaison optionnels) dans la liste du matériel disponible, avec accès à toutes les caractéristiques du matériel et la possibilité de rechercher dans cette liste
  - permettant de rendre le matériel une fois la sortie terminée
  - permettant d'indiquer si des problèmes ont été rencontrés avec le matériel emprunté (casses entre autres) au moment du rendu
  - dans ce dernier cas, un système d'envoi d'e-mail aux personnes concernées a été mis en place pour qu'ils puissent être prévenus qu'il faut réparer le matériel en question

  Ce logiciel sera installé localement sur l'ordinateur du local, l'idée étant que l'on puisse réserver/rendre du matériel uniquement au club et non pas depuis son téléphone n'importe où comme pour l'usage de l'application mobile.

II. L'équipe

Pour réaliser ce projet, nous étions donc 4 étudiants.

Grégory TRESTOUR qui était chef de projet.

Loïc SOUCHON qui était scrum master et qui a permis d'appliquer les règles d'Agilité au sein de l'équipe.

Jade VANDAL qui n'avait pas de rôle particulier.

Antoine THOMAS qui s'est chargé de la partie DevOPS et qui a agit en tant que Happiness Manager lorsque les situations le permettaient.

Nous étions tous développeurs fullstack mais avions cependant certaines préférences pour le front ou le back.

Au niveau de notre organisation, nous avions un rendez-vous hebdomadaire avec Anthony, notre client, afin de lui montrer notre avancement, de lui poser des questions, de répondre aux siennes et de revoir éventuellement la liste de nos tâches à réaliser.

Dans le cadre de notre cours de Management de Projets Innovants, nous avons réalisé des User Stories et Persona afin de mieux cibler les utilisateurs de notre application et logiciel. Anthony étant lui-même utilisateur, en discutant avec lui nous avons pu réaliser des études de terrain également. De même avec Antoine RIVOIRE, élève de notre classe, lui-même pratiquant au guc voile.

III. Technologies et outils

En ce qui concerne les technologies utilisées :

JHipster

Jhipster, un générateur de code. Nous l'avons choisi car c'est celui que nous connaissons désormais le mieux et qui correspondait parfaitement à nos besoins.

Electron, un framework qui permet de développer des applications multiplateformes de bureau à l'aide de technologies web : notamment Javascript, HTML et CSS. Cela fut formateur car nous n'avions jamais réellement développé en javascript. Il s'est avéré très facile, léger et rapide à utiliser. Vous nous avez parlé dans lors de notre soutenance intermédiaire d'un plugin JHipster Electron, nous nous sommes renseignés là dessus mais ayant déjà bien avancé avec Electron nous sommes restés sur notre solution de base. Le principe de notre logiciel est d'utiliser l'API associée à notre application JHipster en tant que backend.

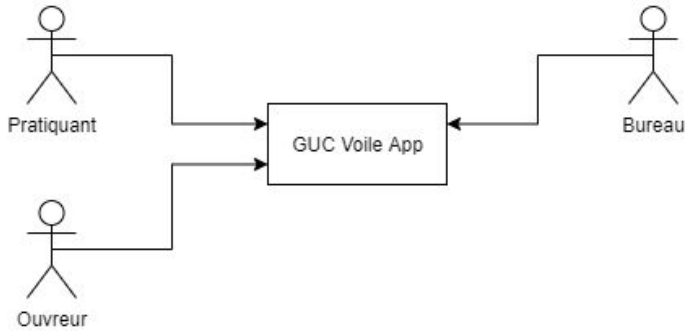
Gitlab, une plateforme de développement basée sur git qui fournit un gestionnaire d'issues, un Wiki, un système de suivi de bugs, intégration et livraison continue, etc. Nous avons essayé d'utiliser au maximum Gitlab, notamment pour le Wiki, les Snippets, le Board bien sûr, le versionning avec les branches et les merge request. Notre Gitlab est composé de 3 sous projets : Documentation, App et Logiciel. Documentation contient toute notre phase de conception, les prototypes, les screens, les tutoriels d'utilisation (réalisés en fin de projet), les documents réalisés en cours de Management de Projet Innovant etc. Nous utilisons également son Wiki pour de la documentation. App contient notre code pour l'application JHipster. Logiciel contient notre code pour le logiciel Electron. Toutes les instructions nécessaires pour la reprise ou l'installation du projet ont été écrites dans ces répertoires.

Visual Studio Code, un éditeur de code très utile, léger et flexible grâce à ses nombreuses extensions installables rapidement et facilement notamment pour l'auto-complétion, les indentations, la mise en forme du code etc.

Google API : qui nous a permis d'installer un calendrier sur notre application web, afin d'afficher les permanences d'ouverture et de fermeture. Nous avons rencontré de nombreuses difficultés avec cette API, notamment en ce qui concerne les clefs d'utilisation (tokens) mais nous sommes finalement parvenus à le faire fonctionner correctement et durablement.

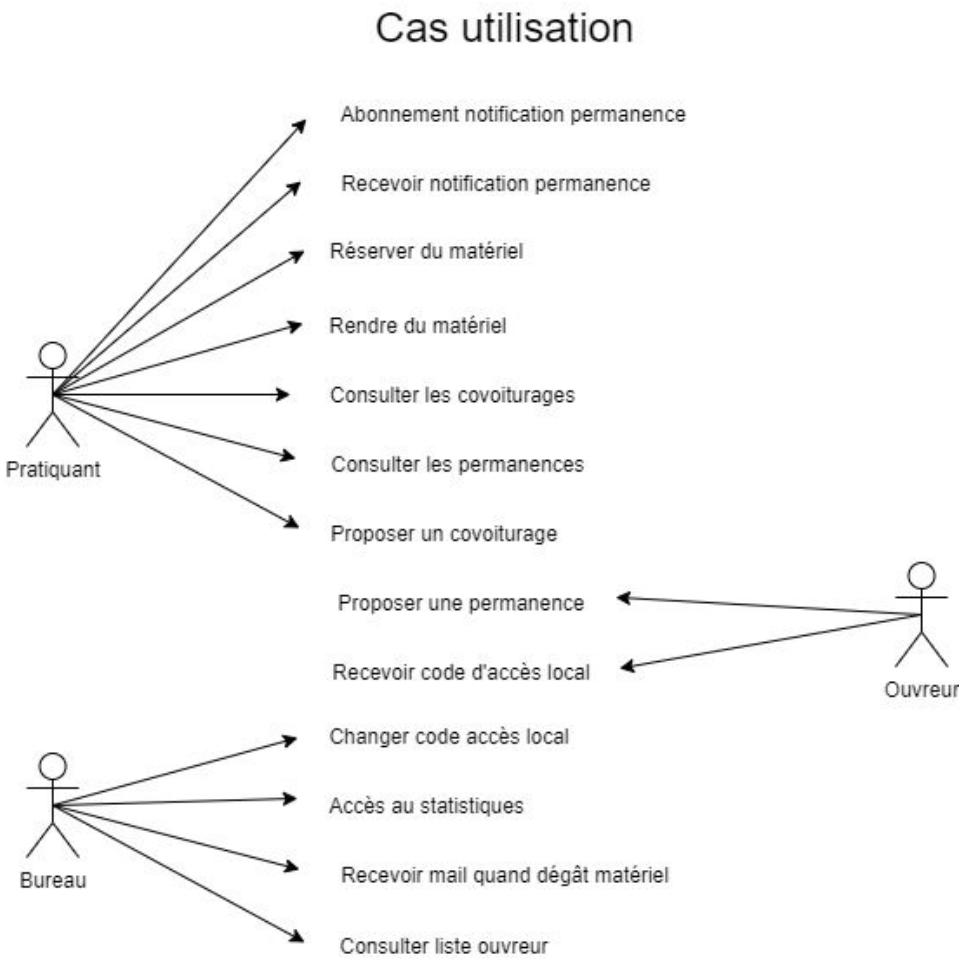
IV. Architecture

Dans un premier temps nous avons réalisé un **diagramme de contexte** du projet :



Le système que nous allons produire est l'application Guc Voile (qui comprend l'application ET également le logiciel). Les utilisateurs de notre système sont les pratiquants, incluant les ouvreurs et les membres du bureau, que nous avons choisi de dissocier afin que cela soit plus clair.

Nous avons également réalisé un diagramme de **cas d'utilisation** :



Un pratiquant peut :

- Accéder à un système d'abonnement / notifications et choisir les jours pour lesquels il souhaite être prévenu quand il y a de nouvelles permanences
- Recevoir une notification par email lorsque des permanences sont créées pour les jours choisis
- Réserver du matériel (planche + voile et optionnel : combinaison + harnais)
- Rendre du matériel qu'il a précédemment réservé et prévenir d'éventuels dommages
- Consulter les covoiturages dans les prochains jours pour se rendre au lac de monteynard et contacter les conducteurs en obtenant leurs coordonnées
- Proposer un covoiturage (indiquer son lieu de départ, heure de départ, heure de retour) et bien sûr modifier ce covoiturage par la suite pour indiquer lorsque celui-ci est complet (plus de place dans la voiture) ou encore le supprimer. Cette dernière fonctionnalité est accessible lorsque le conducteur reçoit le mail de confirmation d'ajout de covoiturage, dedans se trouve un lien unique menant au covoiturage et permettant les modifications.
- Consulter les permanences sur un calendrier

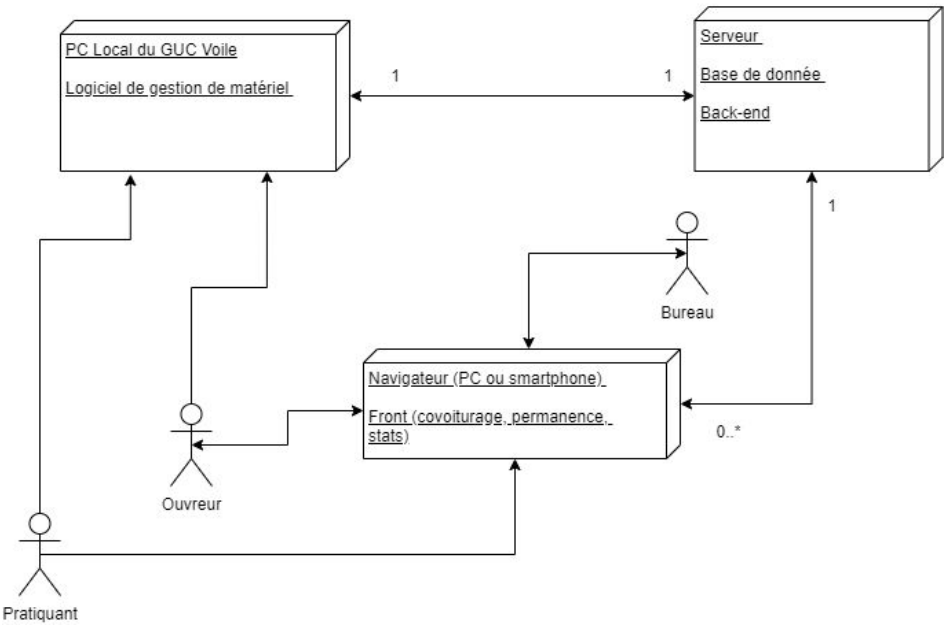
Un ouvreur peut :

- Proposer une permanence, c'est-à-dire indiquer un créneau d'ouverture et de fermeture pour lesquels les pratiquants pourront venir récupérer/rendre du matériel
- Recevoir le code d'accès au local par email, celui-ci étant amené à changer régulièrement

Un membre du bureau peut :

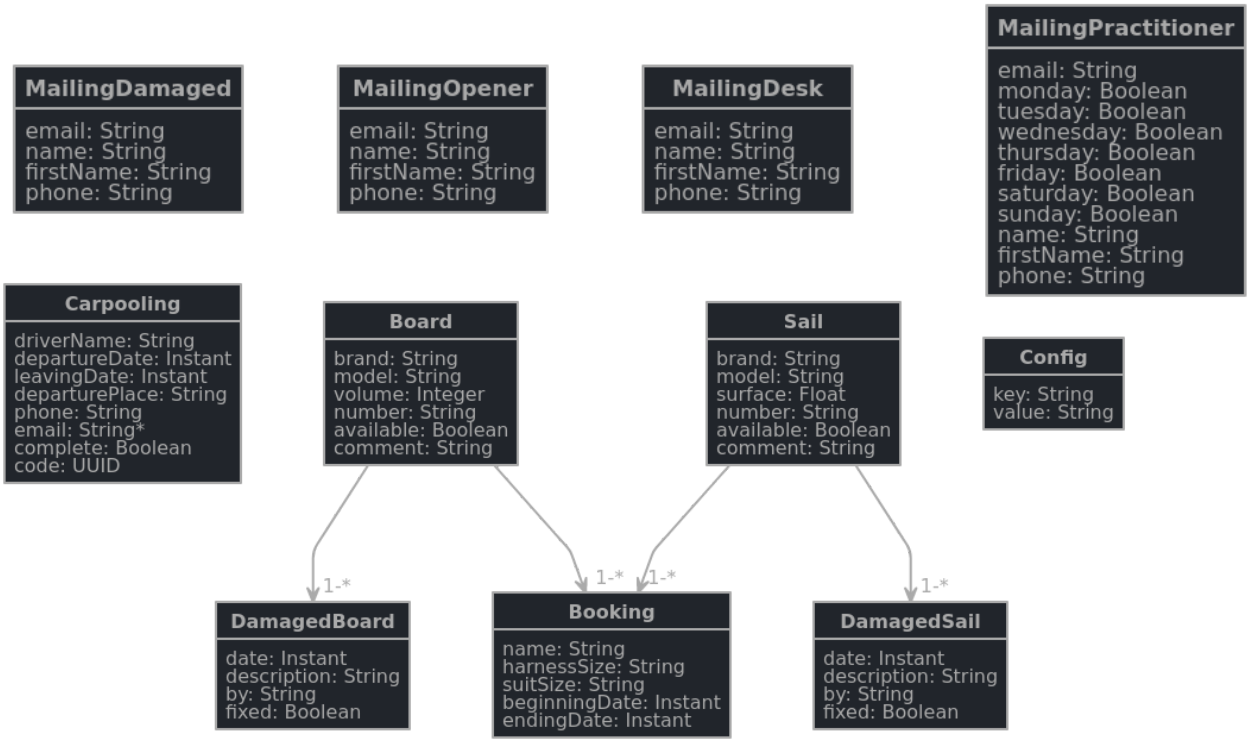
- Modifier le code d'accès au local
- Accéder à des statistiques concernant par exemple les permanences, le matériel, les pratiquants, cela lui permet de mieux gérer le club
- Recevoir un mail lorsque du dégât matériel a été déclaré et ainsi être prévenu qu'il faut le réparer
- Consulter la liste des ouvreurs et la modifier

Ensuite, la **vue physique** :



Nous avons un PC local installé dans le local du GUC Voile au lac de monteynard. Notre logiciel sera installé dessus et relié à internet par un câble ethernet. Avec cela, un serveur qui héberge notre base de données et notre back-end. Et enfin, un navigateur web (de type PC ou smartphone) qui permet l'affichage de notre front-end.

Notre **modèle de données** ressemble à ceci :



Nous avons :

- La table MailingDamaged qui permet de lister toutes les personnes recevant un e-mail lorsque du matériel est déclaré comme endommagé.
- La table MailingOpener qui permet de lister toutes les personnes de type "ouvreur", donc celles pouvant créer des permanences et recevoir le code du local.
- La table MailingDesk qui permet de lister tous les membres du bureau.
- La table MailingPractitioner qui permet de lister tous les pratiquants avec leurs préférences de notifications, par exemple si monday est à true, alors la personne recevra un email lorsqu'une permanence est ajoutée pour un lundi.
- La table Carpooling qui permet de gérer les covoiturages. Le champ code permet d'accéder à un covoiturage par lien unique (reçu par email) sans avoir besoin de gestion de crédentials.
- La table Board et Sail pour Planches et Voiles, qui sont reliées aux tables DamagedBoard et DamagedSail qui recensent les dégâts causés et également la tale Booking pour connaître les réservations.
- La table Config qui permet de gérer plusieurs choses telles que garder un décompte du nombre de permanences créées ou encore le code du local.

V. Réalisation technique

Concernant nos réalisations techniques :

Pour l'application :

- Nous avons développé une API dont la doc générée par Swagger est disponible et très facilement utilisable.
- Nous avons intégré un Google Calendar pour les permanences. Ce point nous a posé de nombreux problèmes notamment de gestion de tokens.
- Nous avons réalisé un système de notifications avec notamment l'envoi d'emails pour les fonctionnalités suivantes :
    - Confirmation de validation de permanence
    - Réception du code du local
    - Confirmation d'ajout de covoiturage
    - Réception des coordonnées de covoiturage
    - Notification lorsqu'il y a des dégâts matériel
- Nous avons réalisé un système de covoiturage avec notamment la création de liens uniques, c'est-à-dire de de type "xhysaASns21nS". Cela permet une plus grande sécurité et le fait de ne pas avoir besoin de credentials. Le principe est simple, lorsqu'un conducteur ajoute un covoiturage, il reçoit par email un lien unique qui lui permet de modifier son covoiturage (modifications, suppression, marquer comme complet). De base, nous aurions pu faire un lien de type "/covoit/modify/{id}" mais cela aurait induit que n'importe qui connaissant l'id du covoiturage aurait pu le modifier/supprimer. Ici, avec un lien unique, nous supprimons ce problème.
- Grâce à JHipster, la gestion de matériel et des différentes listes d'utilisateurs a été hautement simplifiée puisque auto-générée. Nous avons cependant apporté quelques améliorations comme :
    - La présence de search bar sur toutes les pages : pouvoir rechercher une planche (par exemple) parmi toute la liste est indispensable
    - Dans la page d'affichage des planches/voiles :
        - des boutons permettant de trier (décroissant/croissant) selon le volume/surface
        - des boutons permettant de trier (décroissant/croissant) selon la marque / le modèle
        - un bouton affichant les endommagés seulement
        - un bouton permettant d'indiquer le matériel comme réparé
    - Affichage d'images plus significatives que des booléens, notamment dans la page "liste des pratiquants" pour les jours de la semaine
- Et enfin, une page statistiques qui utilise le framework Highcharts pour afficher des graphiques de tous types : histogrammes, diagrammes circulaires, linéaires mais aussi des tables avec pagination et barre de recherche. Cette page a été un challenge important dans le cadre de l'UI/UX car il y avait beaucoup d'informations à afficher de façon efficace et visible, c'est pour cela que nous avons opté pour des onglets regroupant les catégories similaires d'informations. Nous avons aussi dû réaliser des requêtes SQL assez poussées pour obtenir des statistiques parlantes.
- En bonus, notre application est entièrement responsive, tout a été vérifié afin que l'affichage sur petits écrans mobiles soit optimisé au maximum.

Pour le logiciel :

- Le logiciel communique avec l'API de l'application par requêtes axios.
- Il est développé en html+javascript avec le framework Electron.
- Ses fonctionnalités sont basiques :
    - réservation de matériel (planche + voile) parmi la liste du matériel disponible et ajout en option d'un harnais ou d'une combinaison
    - rendu de matériel précédemment réservé
    - possibilité d'indiquer si le matériel a subi des dommages durant la sortie et de décrire le problème
    - système d'identification très simple avec seulement le prénom et le nom du pratiquant, le GUC Voile étant un petit club où tout le monde se connaît, lors de la phase de conception nous avons convenu qu'il n'était pas nécessaire d'avoir un système d'identification poussé
    - l'envoi d'e-mail automatiquement lorsque des dégâts sont déclarés

Nous avons aussi rajouté quelques fonctions rendant l'utilisation du logiciel plus efficace grâce à :

- une auto-complétion, c'est-à-dire que lorsque le pratiquant entre le début de son prénom, la suite de son nom est automatiquement suggéré.
- l'utilisation des touches du clavier, par exemple les touches "Entrer" et les flèches permettent de ne pas utiliser la souris.

VI. Gestion de projet

Pour rappel, nous avons travaillé 2 semaines sur la phase de conception étant donné que nous sommes reparti de zéro. Nous avons ainsi accordé entre 4 et 6 semaines aussi bien pour le développement de l'application que celui du logiciel.

Nous avons travaillé de façon incrémentale, nous avions toujours une version fonctionnelle sur la branche master, et de nouvelles fonctionnalitées étaient ajoutées à chaque fin de sprint.

nous avons intégré la méthode SCRUM avec toutes ses routines :

- Poker planning pendant la phase de conception avec Anthony, qui nous a gentiment apporté son jeu de cartes spécialement dédié et qui a participé avec nous à cette activité. Ce poker planning nous a permis de déterminer les tâches selon nous les plus difficiles et longues. Malheureusement nous n'avons pas assez respecté le poids des tâches pour la création des sprints.
- Chaque sprint durait une semaine
- Nous avons créé des issues avec nos user stories et toutes les tâches correspondantes : c'était notre backlog
- Les daily meetings chaque matin pour se tenir informer de l'avancement
- Les rétrospectives en fin de sprint pour faire le point sur ce qui se passe bien et ce qui est à revoir
- Les sprint review pour faire le point sur l'avancement du sprint

Notre utilisation de Git nous a aussi beaucoup aidé pour la gestion du projet grâce à toutes

les fonctionnalitées telles que :

- La création de branche pour chaque issue
- Une merge request sur master à la fin de l'issue
- Une revue de code au moment de la validation de la merge request
- L'affichage d'un board sur gitlab permettant visuellement de bien s'organiser entre les issues "To do", "In Progress" et "Done/Closed"
- Une fiche de suivi que nous remplissions à la fin de la journée

Concernant la répartition des tâches, chacun était libre de s'assigner les issues qu'il préférait. Quand les tâches étaient plus complexes nous faisions du pair-programming pour s'en sortir au mieux.

Enfin, nous nous sommes inspirés de notre cours de Management de projet innovant pour instaurer une routine un peu spéciale, en effet une fois par semaine, l'un de nous apportait le petit déjeuner, à tour de rôle. Cela était un moment important pour notre cohésion et ambiance générale.

VII. Métriques

Penchons nous dans un premier temps sur le nombre de lignes de code créées. Notre application ayant été générée avec JHipster, il y a beaucoup de code que nous n'avons pas écrit nous même. Nous avons donc réalisé un premier cloc à l'issue de cette génération et un deuxième cloc à la fin de notre projet. Le tableau ci-dessous montre la différence entre ces deux clocs.

Application

TypeScript	8 702
Java	5 436
HTML	3 065
Total	**17 203**

Logiciel

JavaScript	771
HTML	506
Total	**1 277**

Nous comptons donc un total de presque **20 000 lignes de code.**

De plus, grâce à Gitlab, nous savons que nous avons réalisé 70 merge requests, pour un total d'environ 110 issues.

Nous avons ajouté les métriques Sonarqube à nôtre projet afin d'évaluer la propreté du code, on voit que nous avons 70% du code couvert par des tests (surtout test généré par jhipster) et 3% de code dupliqué.

En terme de vulnérabilité, c'est à dire attaque malveillante nous sommes à 0, donc notre application web est sécurisé.

Nous sommes dans le tier A en terme de code smell, donc les normes de codage sont globalement respecté. Le tier D sur les bugs signifient qu'il faut faire attention lors de la création de nouvelles feature, il peut y avoir des soucis de maintenabilité.

Sur le graphe on peut voir que résoudre les problèmes de maintenabilités est rapide sur presque toutes les issues, uniquement 2 sont évalués à plus de 4h de travail.

VIII. Conclusion

En conclusion, ce projet nous a beaucoup plu.

Il était concret et nous nous sommes sentis utiles. De savoir qu'il serait utilisé à la fin par Anthony et les membres du club nous a beaucoup motivé à réaliser des solutions propres, de qualité et surtout, maintenables.

Le travail en groupe s'est très bien passé, nous avons réussi à nous répartir les tâches aisément et chacun a fait son travail. Nous avons compté sur la conscience professionnelle de chacun, ainsi nous nous sommes chacun organisé comme nous voulions, l'idée étant qu'à la fin le travail soit fait, ce qui était le cas.

Travailler avec les méthodes SCRUM et Agile nous a permis de se sentir plus à l'aise dessus. C'était une expérience de plus donc que du positif apporté.

Nous avons réalisé tous les objectifs qu'Anthony nous avait fixé. Toutes les fonctionnalités devant être réalisées le sont. Au début, nous ne pensions pas forcément arriver au bout, ainsi nous sommes très contents d'y être parvenus.

Nous sommes fiers d'avoir pu développer de nouvelles compétences sur ce projet, notamment l'utilisation d'Electron et de Google Calendar qui étaient des choses nouvelles à découvrir et utiliser.

Notre seul regret aura été que nous n'avons pas pu aller nous-même installer le logiciel au GUC Voile.

IX. Démonstration

- **Application**

Démonstration vidéo : https://www.youtube.com/watch?v=S_4OQx0hotI&feature=youtu.be
Démonstration slides : https://air.imag.fr/images/b/b6/Demo_application_gucvoile.pdf

- **Logiciel**

Démonstration vidéo : https://www.youtube.com/watch?v=ipfGFq5QVyo&feature=youtu.be
Démonstration slides : https://air.imag.fr/images/f/fe/Demo_logiciel_guc-voile.pdf

X. Glossaire

- **Un pratiquant :** une personne inscrite / adhérente au club GUC Voile
- **Un ouvreur :** un pratiquant ayant un niveau expérimenté en planche à voile et ayant accès au code du local pour pouvoir l'ouvrir et créer des permanences
- **Un membre du bureau :** un pratiquant aidant à la gestion du club
- **Le président :** un pratiquant qui gère le club sur tous ses aspects
- **Le lac de Monteynard :** lac situé dans l'Isère et alimenté par le Drac
- **Le local :** espace fermé permettant de stocker le matériel de planche à voile et où se trouve un ordinateur fixe avec connexion à internet
- **Une permanence :** un créneau horaire (début et fin) où le local est ouvert pour que les pratiquants puissent récupérer / rendre du matériel de planche à voile
- **Un covoiturage :** un conducteur et sa voiture qui se propose d'emmener avec lui une ou plusieurs personnes vers un lieu

XI. Bibliographie

- Google API Calendar
- Angular Bootstrap

- Angular ng2-search-filter module
- SonarQube explications
- RGPD explications
- Logiciel de MindMap
- Documentation Swagger
- Réalisation de prototypes Figma
- Poker Planning explications
- Draw.io : application de schémas
- Documentation Electron
- Electron tutoriel création d'exécutables
- Electron documentation exécutables
- Axios requests exemples