Rapport final

Projet Kiné Connecté 2.0

Rédigé par :

Eva Bardou - Xavier Devos

Adrian Houbron - Léo Jan

Antoine Pelisson

Projet de fin d'études INFO5 | 2019-2020

Table des matières

1. LE PROJET KINÉ CONNECTÉ 2.0

a. Le sujet

Ce projet a pour but de reproduire un équipement d'aide à la rééducation de l'oreille interne chez les patients souffrant de troubles (ex : équilibre) déjà présent au cabinet **Vestib+, notre client**. La solution mise en place doit être peu coûteuse, sans-fil et disponible à distance pour les patients et les kinésithérapeutes du cabinet Vestib+.

Nous avons repris un projet INFO5 de 2019 et notre solution se décompose en 4 parties :

- Une **interface Kiné (Vestib+ Kiné)** codée à l'aide du framework Angular (v.8) permettant à des kinésithérapeutes de programmer la rééducation de leurs patients. Elle permet également aux professionnels de santé de suivre l'évolution de leurs patients.
- Une **interface Patient (Vestib+)** codée à l'aide du framework Angular (v.8) permettant à des patients de réaliser les exercices programmés par leur kinésithérapeute à la maison via le casque connecté en BLE (Bluetooth Low Energy) sur leur application.
- Un **serveur** codé en Node.js et lié à une base de données MongoDB (via mongoose) qui sert essentiellement d'intermédiaire entre les front-ends et la base de données pour le passage des requêtes mais également pour le lancement de Socket.io qui est utilisé par le chat de nos applications.
- Un **casque connecté** codé en Arduino (proche du C++) composé d'une batterie, une carte ESP32, un module BLE (Bluetooth Low Energy), un gyroscope/accéléromètre, un boîtier et un bandeau de fixation frontale.

b. Cahier des charges

Le cabinet de kinésithérapie Vestib+ voulait tout simplement que nous mettions en production l'application qui avait été développée l'année dernière par le précédent groupe d'INFO5. Cela impliquait plusieurs développements :

- Ajout de fonctionnalités de base manquantes
- Établissement d'une communication bluetooth entre le casque connecté fait par les IESE et l'interface Patient
- Déploiement des deux interfaces web, Kiné et Patient
- Production de documentation

Étant donné que nous avons repris un projet existant, nous n'avons pas eu de vraie phase de conception comme nous avons pu en avoir sur d'autres projets lors de notre cursus à Polytech.

c. Notre équipe

Pour mener à bien ce projet, nous sommes cinq étudiants à travailler en étroite collaboration, dont trois suivent la spécialité Communication Multimédia et deux la spécialité Réseaux Informatiques.

- **Eva**, *chef de projet* et *développeuse full-stack sur les deux interfaces*, était principalement chargée de superviser le projet en s'assurant que le groupe avance bien et qu'il n'y avait de souci pour personne. Elle a veillé à ce que l'organisation du projet et du groupe suive le bon chemin du début à la fin du projet. Elle a également développé des fonctionnalités critiques sur l'interface Kiné comme sur l'interface Patient et activement participé à la résolution des bugs sur les deux interfaces en fin de projet.

- **Xavier**, *SCRUM Master*, *développeur full-stack sur l'interface Patient* et *responsable de la documentation de l'interface Patient*, s'est occupé de présider les Daily Meetings et les rétrospectives de fin de sprint. Il a réalisé des développements sur l'interface Patient principalement au niveau du front-end et quelques fois au niveau de la logique et a participé à la résolution de bugs sur cette même interface. Il a également rédigé la totalité de la documentation technique de l'interface Patient.

- **Adrian**, *développeur full-stack sur l'interface Patient* et *responsable de la documentation du serveur et de l'interface Kiné*, a réalisé des développements sur l'interface Patient principalement au niveau du front-end et quelques fois au niveau de la logique et a participé à la résolution de bugs sur cette même interface. Il a également rédigé la totalité de la documentation technique du serveur et de l'interface Kiné.

- **Léo**, *responsable du Git* et *développeur full-stack sur les deux interfaces*, avait pour rôle de s'assurer de la bonne utilisation du dépôt Git. Il a usé de son expérience pour réaliser les fonctionnalités restantes de l'interface Kiné et grandement participé à la production de code pour la partie Exercices de l'interface Patient. Il a également résolu de nombreux bugs sur les deux interfaces.

- **Antoine**, *développeur du code du casque BLE* et *responsable de la documentation de déploiement*, a été le seul à toucher au code du casque de rééducation sans lequel le projet ne pourrait pas fonctionner. Suite à ça, il s'est penché sur la production d'une feuille de route (documentation) permettant à notre client et ses patients d'effectivement pouvoir utiliser notre projet depuis n'importe quel ordinateur.

2. ORGANISATION DU PROJET

a. Méthode agile SCRUM

Pour l'organisation et la gestion de projet, nous avons choisi d'utiliser la méthode SCRUM.

Pour ce faire, à chaque séance, Xavier se chargeait de présider le Daily Meeting que nous avions programmé à 9h30 voire 14h si nous n'avions pas de séance de prévue en matinée. Tous les soirs, il fallait également que chacun reporte ce qu'il avait fait dans la journée sur la page AIR du projet (voir Fiche de suivi) et les heures effectuées sur un excel.

En fin de semaine, le vendredi à 15h30, Xavier organisait la rétrospective de fin de sprint où l'on pouvait tous échanger sur ce qui avait été fait durant la semaine, ce qui avait été abandonné, repoussé et les points qui pouvaient être améliorés au niveau de la gestion de groupe pour la semaine suivante. À la suite de la rétrospective, Xavier se chargeait également de reporter ce qui avait été dit sur notre page AIR, nous redéfinissions et réorganisions les issues pour le sprint suivant et Eva assurait le versionnage du projet en fusionnant la branche **dev** sur **master**. Le projet a été composé de 6 sprints au total.

b. Planification des sprints

Sprint	***Tâches planifiées***
0	- Mise en place des outils - Création des issues - Installation de l'ancien projet
1	- Prise en main de l'ancien projet - Réunion avec le client - Discussion autour des implémentations à faire sur les sprints suivants
2	- Recherche d'un moyen pour lier l'interface Patient et le casque - Début de la refonte du design de l'interface Kiné
3	- Recherche d'un moyen pour lier l'interface Patient et le casque - Suite de la refonte du design de l'interface Kiné - Début de la refonte du design de l'interface Patient
4	- Liaison effective du casque et de l'interface Patient - Fin de la refonte de l'interface Kiné - Suite de la refonte de l'interface Patient
5	- Fin de la refonte de l'interface Patient - Réunion avec le client - Bugfix - Préparation des démonstrations
6	- Dernières implémentations suite au retour du client la semaine précédente - Bugfix - Déploiement chez le client - Rédaction de la documentation - Rédaction des rapports - Préparation de la soutenance

c. Outils de gestion de projet

Pour acquérir une bonne organisation de groupe et que le projet se déroule pour le mieux pour tout le monde, nous avons mis en place plusieurs outils et conventions de communication, de partage de documents et d'uniformisation du code :

GitLab

GitLab est utilisé par les membres du groupe lors de l'implémentation et de l'écriture du code. Il nous sert notamment pour le versionnage du code et permet à chacun de travailler sur une tâche particulière en simultané. Afin que nous puissions nous y retrouver facilement, sans souci d'interprétation individuelle, nous avons mis en place des conventions assez strictes pour le nommage des commits, des branches, la manière dont nous changeons de version, etc.

Cet outil nous sert également lors de la gestion de projet notamment pour répartir et gérer les tâches générées par la méthode Agile grâce aux issues, aux boards, aux milestones, etc. Pour ce faire, nous avons également mis en place des conventions de nommage strictes pour que tout le groupe puisse s'y retrouver.

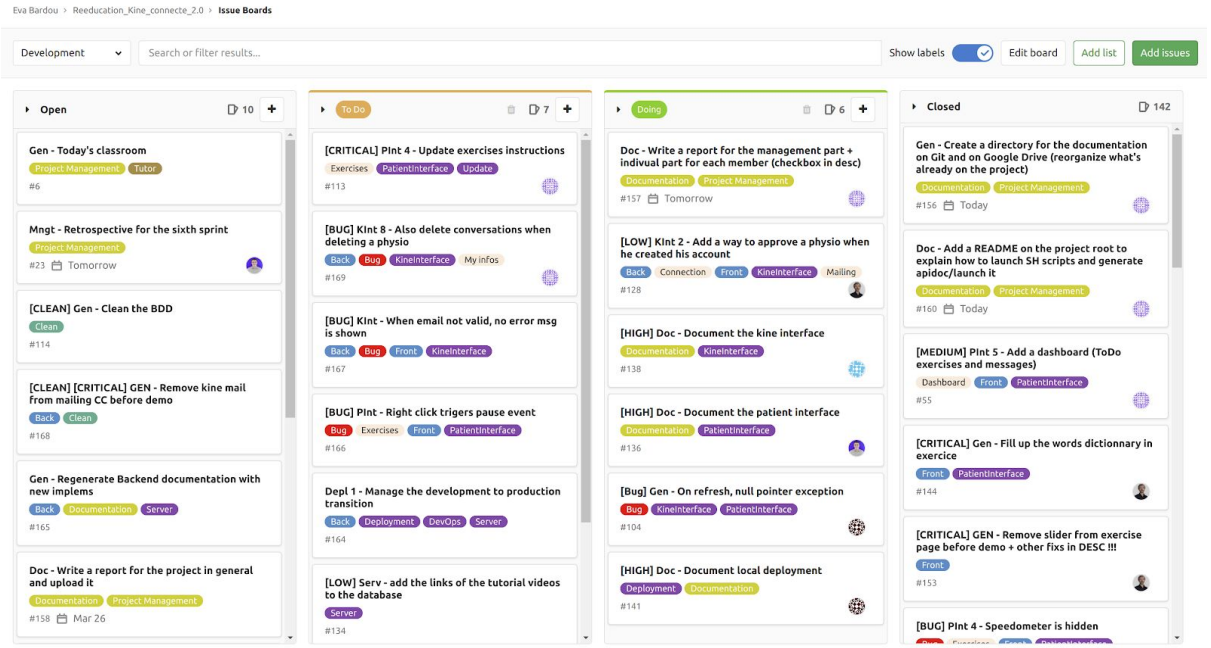


Tableau de bord pour le développement sur GitLab, issues organisées en catégorie : Ouvertes, A faire, En cours, Finies

Slack

Slack est un outil de communication grandement utilisé en entreprise, il propose un chat modulable suivant les besoins et des outils pour le partage de documents. Nous avons choisi de mettre en place un espace de travail Slack dans ce projet pour permettre à chacun d'être à jour rapidement en lisant les derniers messages importants qui ont été envoyés dans les différentes chaînes. L'outil d'épinglage des messages nous sert également énormément pour avoir accès aux informations critiques en un clic (ex : dates des soutenances, lien vers le journal de bord, etc).

Google Drive

Pour partager et collaborer lors de l'écriture des documents utiles et demandés dans le cadre de ce projet (comme ce rapport ou les documentations techniques), nous avons utilisé Google Drive. Il nous permet d'accéder rapidement à tous les documents qui n'ont pas leur place sur GitLab via un dossier propre au projet.

VSCode

VSCode est un éditeur de texte pratique qui propose de nombreux plugins et est donc très modulable suivant les envies de chacun. Il nous permet d'avoir un agencement du code commun à tous les membres du groupe (tabulation, etc) et est très facile à prendre en main.

Discord

Nous utilisons Discord dans le cadre du confinement lié à l'épidémie du virus COVID-19. C'est une application (disponible sur web également) similaire à Skype qui nous permet d'effectuer nos Daily Meetings en vocal, de partager notre écran pour demander de l'aide sur un problème en particulier et plus globalement de ne pas rester seul toute la journée.

3. TECHNOLOGIES

Afin de mener à bien ce projet, nous avons utilisé différentes technologies. Etant donné que nous avons repris un existant, elles étaient d'ores et déjà fixées et imposées lors que nous nous sommes lancés dans le développement.

a. Framework Angular

Pour la partie front-end, nous avons utilisé **Angular** (v8), un framework côté client basé sur TypeScript. La particularité de ce dernier est qu'il permet d'utiliser du HTML et du CSS afin d'écrire les modules de l'application, et d'écrire toute la logique de cette même application en TypeScript. Angular se base sur la notion de composants, un composant était formé d'une vue (HTML), d'un style (CSS) et d'une logique (TypeScript).

Certains membres du groupe avaient déjà eu l'opportunité de travailler avec cet outil, mais ce n'était pas le cas de tout le monde, il a donc été nécessaire pour ces personnes de se former à Angular.

b. Node.js, Express.js & Socket.io

Le Backend de notre application est, quant à lui, écrit en JavaScript à l'aide de **Node.js**. De plus, nous avons utilisé le framework **Express.js** afin de nous permettre de créer facilement notre API et de lancer rapidement notre serveur.

Nous avons aussi utilisé la bibliothèque JavaScript **Socket.io** lors de la création du système de messagerie alimentant les interfaces Kiné et Patient. En effet, cette bibliothèque nous permet, via l'utilisation des sockets, d'avoir une communication bidirectionnelle en temps réel entre plusieurs clients web.

c. MongoDB

En ce qui concerne la base de données, nous utilisons une base de données NoSql à l'aide de **MongoDB**. Cette base de données est donc orientée document : les données présentes dans la base sont stockées au format JSON dans des documents. De ce fait, cela permet de manipuler facilement ces objets. Nous avons utilisé le module **Mongoose** afin de permettre à notre serveur de communiquer avec cette base de données.

d. Arduino et BLE

Pour ce qui est du casque connecté, en 2019, les IESE avaient proposé une approche en Bluetooth normal mais cela ne convenait pas aux exigences du client. Nous avons re-paramétré et réécrit le code de ce dernier en **Arduino** (langage proche du C++), il communique désormais avec l'interface Patient via **Bluetooth Low Energy** (**BLE**).

4. ARCHITECTURE TECHNIQUE

a. Serveur et base de données

Les interfaces de notre solution ont besoin de régulièrement récupérer des informations qui sont présentes en base de données. Ainsi, elles communiquent via des requêtes **HTTP** avec l'**API** mise en place côté serveur. Ce serveur est commun aux deux interfaces et fait tourner une surcouche **Express.js**, un module d'envoi de mail, ainsi que le système de sockets utilisé par la messagerie en temps réel. Le serveur traite la requête envoyée et se charge de la communiquer à la base de données via le module **mongoose**. Cela nous permet de récupérer, de modifier ou de supprimer des entrées en base. Par la suite, il renvoie le résultat de ces changements au client.

b. Interface Kiné et interface Patient

Notre projet est composé de deux applications web distinctes :

- L'interface dédiée aux kinésithérapeutes, nommée ***Vestib+ Kiné*** ou *Interface Kiné*
- L'interface dédiée aux patients, nommée ***Vestib+*** ou *Interface Patient*

Depuis l'interface Kiné, l'utilisateur peut gérer les informations concernant sa patientèle, programmer des sessions d'exercices pour ses patients, analyser les résultats des exercices et communiquer avec ses patients et ses collègues grâce à une messagerie en temps réel.

Depuis l'interface Patient, l'utilisateur peut effectuer ses sessions programmées par son kinésithérapeute référent, observer le résultat de ses exercices déjà réalisés et éventuellement contacter son kinésithérapeute s'il rencontre un problème quelconque.

c. Casque connecté

Le casque connecté est utilisé par les patients pour réaliser leurs exercices. Afin d'assurer la communication bluetooth entre le casque connecté et l'interface Patient, nous avons utilisé

le module **AngularWebBluetooth**, qui est une surcouche du module **WebBluetooth**. Ce dernier fonctionne en **Bluetooth Low Energy** (BLE) sur les dernières versions des navigateurs Chrome, Microsoft Edge et Opéra. Le casque communique donc, avec l'interface Patient, via BLE, et envoie à cette dernière un flux de données dans lequel on retrouve la valeur de la vitesse angulaire sur les trois axes (***x***, ***y*** et ***z***).

d. Communication entre les différents éléments

Concernant l'architecture globale de cette solution, la partie front-end est régie par **Angular**. De ce fait, nous avons une structure en **composants** (components), mais aussi des **services**, qui peuvent être appelés par plusieurs composants. Prenons par exemple le ***patientService*** qui nous permet d'obtenir les informations du patient connecté, il est appelé par toutes les pages qui nécessitent l'affichage de ces informations.

Les deux interfaces, ***Patient*** et ***Kiné***, partagent cependant le même back-end. Ce dernier est écrit en **Javascript** à l'aide de **Node.js** et **Express.js**. Nous y retrouvons un fichier ***server.js*** dans lequel se retrouve la gestion des requêtes HTTP envoyées par les interfaces. Ainsi, si l'on souhaite afficher une information particulière sur une des interfaces, il est nécessaire de faire une requête à ce serveur, à l'aide d'une méthode que l'on retrouve dans les services cités précédemment. Ces méthodes nous renverront une souscription à laquelle nous pourrons nous abonner et qui nous notifiera de la réponse renvoyée par le serveur une fois le traitement fini. Tout ce mécanisme est exécuté de manière asynchrone.

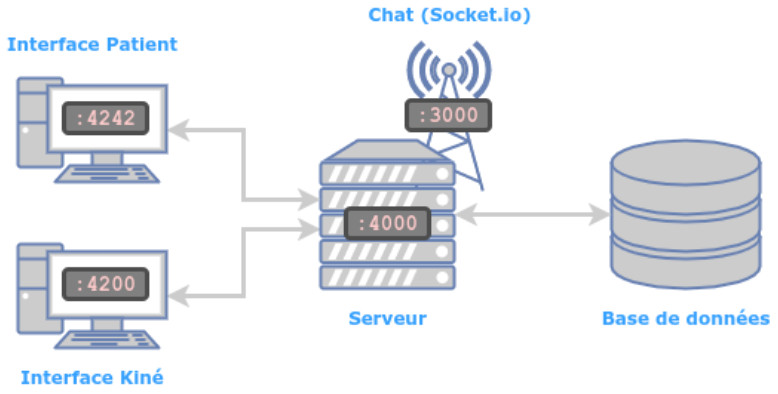


Schéma de la communication des différents éléments présents sur la solution Kiné Connecté 2.0

5. RÉALISATION TECHNIQUE

a. Refonte ergonomique et graphique

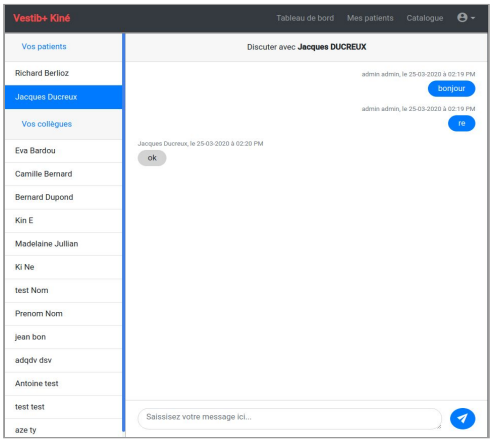
Ce n'était pas demandé, mais nous avons effectué une refonte graphique de toute l'application. Nous l'avons rendue plus esthétique, mais également plus ergonomique. Nous avons passé du temps sur la conception des interfaces, afin qu'elles soient le plus logique possible. Aussi, nous avons pensé aux utilisateurs de l'interface dédiée aux patients, qui sont pour la plupart des personnes âgées, en faisant en sorte que tout soit bien visible, écrit assez gros et que le déroulement des actions à effectuer soit assez logique.

Avant - Liste des patients

Après - Liste des patients

b. Ajout de fonctionnalités

De plus, nous avons ajouté des fonctionnalités importantes et/ou utiles, telle que l'affichage des tableaux de bord, afin de ne pas perdre de temps dans le parcours de sous-menus interminables. Au lieu de cela, un récapitulatif de tout ce que nous avons à faire nous est présenté sur une seule page. Nous proposons également une fonctionnalité d'envoi de mail, notamment pour la réinitialisation des mots de passe utilisateur, la validation d'un nouveau kinésithérapeute ou encore la notification auprès d'un patient resté trop longtemps inactif. De plus, une messagerie en temps réel a été mise en place. Elle fonctionne à l'aide de la communication entre deux sockets, chacune associée à un interlocuteur. Ainsi, deux utilisateurs, peu importe l'interface à laquelle ils sont connectés, peuvent communiquer ensemble à tout moment.



Messagerie

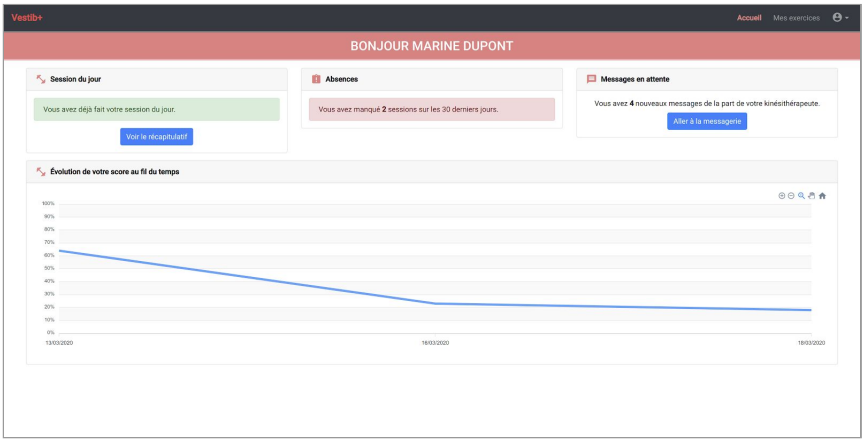


Tableau de bord - Interface patient

c. Communication entre le casque connecté et l'interface Patient

Nous avons également réussi à établir la communication en **BLE** entre le ***casque connecté*** et l'***interface Patient***. Ainsi, le patient n'a qu'à allumer son casque, se connecter sur son interface, appuyer sur un bouton pour s'appareiller avec le dispositif, pour ensuite être en mesure de réaliser ses exercices en toute simplicité. Cette communication est aujourd'hui possible car nous avons recodé la partie bluetooth du casque. En effet, à la reprise du projet, ce dernier devait être connecté en bluetooth simple, or nous avions besoin de BLE pour être en accord avec les technologies web les plus récentes qui ne fonctionnent qu'avec celui-ci.

d. Correction de bugs

Nous avons également intensivement signalé et corrigé tous les bugs pouvant apparaître en navigant sur notre solution. En effet, lorsque nous avons repris **Kiné Connecté 2.0**, le projet était fonctionnel, mais trop permissif, dans le sens où, par exemple, l'exactitude des informations saisies par l'utilisateur n'était jamais vérifiée. Cette permissivité pouvait engendrer des comportements inattendus. De plus, en ajoutant de nouvelles fonctionnalités, nous avons également favorisé l'apparition de certains bugs.

e. Restructuration du code

Nous avons aussi procédé à une restructuration du code de manière assez globale, notamment par rapport au routing et à la structure du projet en lui-même. Nous avons renommé et nettoyé bon nombre de composants, de variables, de routes, etc.

f. Mise en place du déploiement

Nous avons essayé d'assurer un déploiement dans les meilleures conditions possibles pour notre client. Le cabinet s'est chargé de louer un nom de domaine, sur lequel nous avons déployé les deux sites web, chacun sur deux sous-domaines différents. Pour simplifier la procédure de déploiement, nous avons mis en place deux environnements, un pour le développement, et un pour la mise en production qui permettent de facilement alterner entre une solution favorable aux développeurs ou aux utilisateurs.

g. Production de documentations

Pour finir, afin que notre projet soit facilement maintenable et utilisable, nous avons produit un bon nombre de documentations :

- Pour la documentation destinée à de futurs développeurs, on retrouve une documentation de l'API (disponible en ligne) qui détaille le fonctionnement de chaque *endpoint* ainsi que deux documentations techniques des interfaces.
- Nous avons également une documentation utilitaire, qui servira aux kinésithérapeutes du cabinet **Vestib+**, s'ils veulent modifier certains points sur le site, ils pourront ainsi le faire en toute autonomie (ex : changer les vidéos tutoriel ou changer l'adresse mail administrateur).
- Enfin, nous avons produit une documentation pour le déploiement, afin que les kinésithérapeutes du cabinet **Vestib+** soient en mesure, à terme et après la phase de test, de redéployer la solution sur une nouvelle VM qui conviendra un peu plus à leurs besoins.

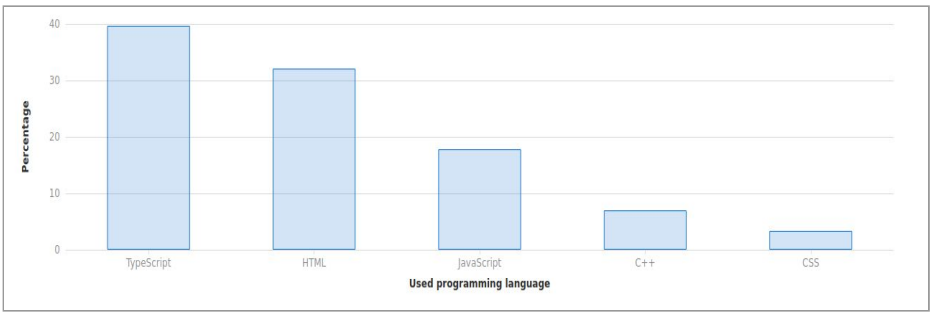
6. MÉTRIQUES LOGICIELLES

Nous avons énormément utilisé GitLab afin de nous aider au niveau de la gestion de projet, notamment avec le *Board* d'*Issues*. Ainsi, nous finissons le projet avec **193** commits, **79** merge requests, **190** issues, et **6** milestones (qui correspondent aux 6 sprints).

Lorsque nous avons repris le projet, il y avait un total de **35 572** lignes de code, à la soutenance de mi-parcours, nous étions à **39 552** lignes de code, et en fin de projet, nous sommes maintenant à **46 227** lignes de code.

Les langages que nous avons utilisé lors de ce projet sont les suivants :

- TypeScript **39,69%**,
- HTML **32,09%**,
- Javascript **17,81%**,
- C++ **6,99%**,
- et CSS **3,35%**.



En ce qui concerne le temps ingénieur, nous avons consacré un total de **989** heures sur ce projet, le détail en fonction des différents membres du projet est donné ci-dessous :

Eva - **276** heures Xavier - **175,5** heures

Adrian - **150,5** heures Léo - **167** heures

Antoine - **220** heures

On notera cependant, que Léo a été malade durant deux semaines, mais ce dernier a quand même continué de travailler depuis chez lui, ce qui explique que le nombre d'heures qu'il a accordé au projet soit légèrement inférieur à celui des autres membres du groupe.

Pour Adrian, c'est un cas différent : il effectuait le premier semestre de l'année en Erasmus en République Tchèque et ce dernier s'est terminé le 7 février. En comptant le temps de voyage retour en France, il n'a pu arriver sur Grenoble que le 13 février, soit deux semaines après le début du projet, ce qui explique un nombre d'heures de travail légèrement inférieur à celui des autres membres du groupe.

Toujours pour illustrer le travail fourni par chacun des membres de l'équipe, voici la répartition du nombre de commits, bien que cela ne soit pas extrêmement représentatif du travail de chacun car lors d'une merge request nous faisions un *squash des commits* :

Eva - **41** commits Xavier - **5** commits

Adrian - **9** commits Léo - **42** commits

Antoine - **12** commits

Enfin, à l'aide des connaissances acquises en cours de **Management de Projet Innovant** (MPI), nous avons calculé que notre projet coûterait **33 192,07€** à une entreprise qui aurait voulu le financer.

7. CONCLUSION

a. Ce qu'il aurait fallu améliorer

Dans une volonté d'être polyvalent à la fin de nos études, plusieurs membres du groupe auraient aimé avoir des tâches un peu plus variées tout au long du projet afin d'augmenter et de diversifier leurs connaissances personnelles. Cependant, il a été difficile de suivre cet état d'esprit tout en respectant nos délais. En effet, dès qu'une tâche demandait un certain niveau, on l'attribuait à la personne qui avait le plus de connaissances dans ce domaine.

D'autre part, avec le recul nous nous sommes rendu compte que certaines tâches auraient pu être découpées en plusieurs tâches courtes. Cela nous aurait permis de transformer une tâche longue effectuée par une seule personne en plusieurs tâches attribuées à plusieurs membres du groupe pour gagner du temps. Nous aurions également eu une meilleure visibilité sur l'avancement de notre travail.

b. Points positifs

Dans la filière Informatique à Polytech, nous avons déjà eu l'occasion de travailler sur de nombreux projets en groupe, ce n'était donc pas notre premier. Cependant, c'était la première fois pour nous que nous travaillions pour un vrai client (hors stage en entreprise).

Nous avons énormément donné pour ce projet car il nous intéressait particulièrement, que ce soit en terme de sujet ou d'apport personnel. Nous avons eu à travailler pour le client **Vestib+**, que nous avons d'ailleurs eu l'occasion de rencontrer plusieurs fois. Le fait de travailler pour un client qui prend en considération notre niveau, nos efforts et reste très cordial avec nous, nous a énormément motivé à travailler plus que de mesure sur la réalisation. Kiné Connecté 2.0 est également un projet concret qui sera utilisé à l'avenir par de vrais kinésithérapeutes et patients, cela nous a grandement incité à tous nous impliquer pour que les futurs utilisateurs soient satisfaits de notre travail.

8. DÉMONSTRATION

a. Déroulement

Vous pouvez retrouver le déroulement des opérations de la démonstration de l'utilisation des interfaces Kiné (**Vestib+ Kiné**) et Patient (**Vestib+**) à ce lien : KC2.0 - Déroulement démonstration.pdf

b. Enregistrement - Screencast

Vous pouvez retrouver un enregistrement de la démonstration, faite par l'un de nos développeurs, sur l'utilisation des interfaces Kiné (Vestib+ Kiné) et Patient (Vestib+) à ce lien : KC2.0 - Démonstration - Screencast.mpg

9. GLOSSAIRE

API Application Programming Interface.

Back-end Partie de l'application inaccessible directement par les utilisateurs, qui réalise le traitement des informations.

Base de données Ensemble d'informations structurées accessibles au moyen d'un logiciel.

Board Tableau virtuel permettant d'organiser nos tâches sur GitLab.

Commit Commande enregistrant la nouvelle version d'un fichier sur GitLab.

Composant Angular Section de l'interface de l'application.

Daily Meeting Courte réunion quotidienne permettant de faire un point sur l'avancement de chaque membre de l'équipe.

Endpoint Bout d'un canal de communication.

Environnement (dev, prod) Ensemble des matériels et des logiciels système, dont le système d'exploitation, sur lesquels sont exécutés les programmes de l'application.

Front-end Interface utilisateur.

Full-stack Un développeur full stack est un informaticien capable de réaliser des tâches à n'importe quel niveau technique de la pile des différentes couches qui constituent une application informatique.

Issue Tâche à réaliser sur GitLab.

Merge request Demande de fusion d'une branche dans une autre sur GitLab.

Milestone Jalon : étape marquant la fin d'une première étape juste avant le démarrage d'une seconde étape.

Module Logiciel prévu pour ajouter de nouvelles fonctionnalités à une application.

Nom de domaine Identifiant de domaine internet (un domaine est un ensemble d'ordinateurs reliés à Internet et possédant une caractéristique commune).

Oreille interne Partie du système auditif qui contient l'organe de l'ouïe, mais aussi le système vestibulaire, organe de l'équilibre, responsable de la perception de la position angulaire de la tête et de son accélération.

Plugin Paquet qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

Requête Moyen formel d'effectuer une recherche d'information dans un système d'information (par exemple une base de données).

Routing Mécanisme par lequel des chemins sont sélectionnés dans un réseau afin d'acheminer les donnés d'un expéditeur jusqu'à un ou plusieurs destinataires.

SCRUM Master Son rôle est de rendre l'équipe plus efficace grâce au Scrum.

Service Angular "Bout de code" ou classe contenant du code qui peut être réutilisable, ou des données que l'on veut partager entre plusieurs composants.

Socket Interface de connexion permettant une communication inter processus.

Sprint Rassemblement de personnes impliquées dans un projet afin de se concentrer sur le développement de ce dernier.

Versionnage Mécanisme grâce auquel on préserve la version d'une entité logicielle.

VM Virtual Machine : émulation d'un appareil informatique.

10. WEBOGRAPHIE

Domaine médical
- https://fr.wikipedia.org/wiki/Oreille_interne
- https://www.msdmanuals.com/fr/accueil/troubles-du-nez,-de-la-gorge-et-de-l%E2%80%99oreille/troubles-de-l%E2%80%99oreille-interne/pr%C3%A9sentation-de-l%E2%80%99oreille-interne

Déploiement
- https://angular.io/guide/deployment

Interfaces
- https://getbootstrap.com
- https://angular.io
- https://material.angular.io

Serveur
- https://www.mongodb.com
- https://expressjs.com/fr/
- https://nodejs.org/en/docs/

Capteur
- https://github.com/manekinekko/angular-web-bluetooth

Chat
- https://socket.io/docs/

Autre
- https://bitoduc.fr
- https://fr.wiktionary.org
- https://www.deepl.com
- https://drive.google.com
- https://gitlab.com
- https://air.imag.fr/index.php/Projet_Kine_2.0
- https://app.slack.com
- https://discordapp.com/
- https://stackoverflow.com
- https://trello.com
- https://openclassrooms.com