

# Rapport de projet – NutriWatch

<b>Introduction</b>	<b>2</b>
<b>I) Rappel du sujet / besoin et cahier des charges</b>	<b>3</b>
I.1) Situation actuelle	3
I.2) Les besoins	3
I.3) Le cahier des charges	3
<b>II) Technologies employées</b>	<b>4</b>
<b>III) Architecture technique</b>	<b>5</b>
<b>IV) Réalisations techniques</b>	<b>6</b>
<b>V) Gestion de projet (méthode, planning prévisionnel et effectif, gestion des risques, rôles des membres...)</b>	<b>7</b>
<b>VI) Outils (collaboration, CI/CD ...)</b>	<b>8</b>
<b>VII) Métriques logicielles : lignes de code, langages, performance, temps ingénieur (d'après vos journaux), la répartition des lignes de code et des commits en pourcentage entre les membres du projet...)</b>	<b>9</b>
<b>VIII) Conclusion (retour d'expérience)</b>	<b>10</b>

# Introduction

Le suivi de la dénutrition chez les patients hospitalisés est un enjeu important pour garantir leur santé et leur bien-être. Dans ce contexte, notre équipe a été sollicitée par une clinique pour concevoir une application web permettant de suivre l'état de dénutrition des patients de manière automatisée. Ce rapport a pour but de présenter le travail accompli par notre équipe pour répondre à cette demande. Nous commencerons par rappeler le sujet et les exigences techniques et fonctionnelles dans la première partie. Ensuite, nous décrirons les technologies utilisées et l'architecture technique de l'application dans les parties II et III. La partie IV sera consacrée aux réalisations techniques, notamment les fonctionnalités de l'application et les interfaces utilisateur. Nous aborderons ensuite la gestion de projet dans la partie V, ainsi que les outils de collaboration et l'intégration continue/déploiement continu dans la partie VI. Les métriques logicielles seront analysées dans la partie VII, avant de conclure avec un retour d'expérience sur la réalisation du projet et des suggestions d'amélioration pour le futur.

# I) Rappel du sujet

## I.1) Situation actuelle

Actuellement, le service chargé de surveiller l'état de dénutrition des patients n'a pas d'application de suivi spécifique calculant automatiquement les états de dénutrition. C'est-à-dire que chaque matin, les équipes doivent parcourir les autres applications de la clinique contenant les informations des patients entrés la veille dans les différents services de la clinique pour trouver leurs données et les copier/coller dans un Google Sheet. Cela demande beaucoup de temps de travail répétitif et ennuyeux car les applications actuelles de la clinique ne sont pas pensées pour cette tâche, il faut compter au minimum deux heures de copier/coller par jour pour l'équipe. Chaque patient a plusieurs données comme le poids actuel, le poids à 1 et 6 mois, l'EPA, le handgrip (mesure de la force de poigne), ou encore l'IMC. Une fois copié/collé dans leur Google Sheet, les calculs de dénutrition se font automatiquement et les diététiciens peuvent choisir d'aller voir certains patients ou de demander aux agents de nutrition d'aller collecter les données manquantes. Nous pouvons comprendre avec ce que nous avons décrit de la nécessité de proposer à la clinique une solution pour améliorer le processus actuel.

## I.2) Les besoins

Les besoins de la clinique sont multiples, nous pouvons les classer en 3 types distincts :

### 1. Les besoins vitaux :

Avoir une application fonctionnelle qui calcule les statuts de dénutrition automatiquement en récupérant les données des autres applications, et proposant plusieurs types de comptes différents ayant des droits différents pour les agents nutritionnels, les diététiciens et le chef de service.

### 2. Les besoins importants :

Proposer une page de statistiques recensant le nombre de diagnostics produits sur chaque service au fil du temps. Proposer aussi un système d'exportation de données en fichiers (tel que des CSV).

### 3. Les besoins bonus :

Initialement le projet était proposé avec des fonctionnalités techniquement difficile à faire et proposant peu de valeur ajoutée pour la clinique tel que l'estimation de calories par une photo de repas ou encore la connexion à des balances connectées. Ces fonctionnalités n'étant pas rentables en termes d'utilisation/temps de travail, nous les avons considérées comme bonus.

### I.3) Le cahier des charges

- La clinique commanditaire a besoin d'un outil de suivi de la dénutrition des patients.
- L'outil doit être une application web.
- L'application doit automatiser la mise à jour des données des patients.
- L'application doit calculer automatiquement l'état de dénutrition du patient à partir de ses données.
- L'application doit calculer automatiquement des statistiques journalières à partir de ses données.
- L'application doit envoyer des alertes aux médecins en cas de risque de dénutrition grave.
- L'application doit permettre la visualisation des données des patients et de leur état de dénutrition pour les médecins.
- L'application doit être développée en utilisant les langages et frameworks suivants : Java, JHipster et Angular.
- L'application doit être conçue selon une architecture MVC, avec des services web RESTful et une base de données.
- Les membres de l'équipe doivent avoir des rôles clairement définis et leur contribution au projet doit être mesurée.
- L'outil de collaboration Git doit être utilisé pour la gestion du code source.
- L'intégration continue/déploiement continu doit être mise en place pour faciliter le développement et le déploiement de l'application.

## II) Technologies employées

La conception de l'application web a nécessité l'utilisation de plusieurs langages et frameworks. Le choix des technologies utilisées a été basé sur la stack imposée par le projet. En ce qui concerne le back-end, nous avons opté pour le langage de programmation Java et le framework JHipster. Java profite d'une grande popularité et d'une bonne performance en matière de développement d'applications web. JHipster, quant à lui, est un générateur d'application web qui nous a permis de générer rapidement et facilement les couches de présentation, de service et de persistance. En ce qui concerne le front-end, nous avons utilisé le framework Angular pour sa capacité à fournir une expérience utilisateur interactive et dynamique. Il permet également de créer des applications web responsives qui s'adaptent à différents types d'appareils.

Les choix technologiques que nous avons faits nous ont permis de bénéficier de nombreux avantages. En utilisant Java, nous avons pu bénéficier d'une grande communauté de développeurs et d'une abondance de ressources en ligne pour résoudre des problèmes éventuels. L'utilisation de JHipster a également permis de générer rapidement une application web robuste et sécurisée, tout en respectant les meilleures pratiques de développement. En outre, Angular a été choisi pour ses performances de rendu rapides, sa structure de composants et sa capacité à simplifier la gestion des états de l'interface utilisateur. En somme, les technologies utilisées ont permis de développer une application web performante, sécurisée et facile à utiliser.

### III) Architecture technique

Pour notre projet nous avons choisi de repartir d'une nouvelle base code en régénérant l'application à partir d'un nouveau JDL. En effet, les données ainsi que l'architecture technique étant assez éloignées du projet eCom sur lequel ce projet fait suite, cela était nécessaire. Nous avons ensuite repris et adapté les fichiers concernant le front-end.

L'architecture technique est basée sur le modèle MVC (Modèle-Vue-Contrôleur) pour la partie front-end, et sur les services web RESTful pour la partie back-end. La base de données utilisée est PostgreSQL, qui a été choisie pour sa stabilité et sa sécurité.

Le modèle MVC a été choisi pour la partie front-end, car il permet de séparer la logique de présentation (Vue) de la logique métier (Modèle) et de la logique de contrôle (Contrôleur). Ainsi, il est plus facile de maintenir et de faire évoluer l'application. Angular, le framework front-end utilisé, implémente ce modèle de manière native et fournit des outils pour faciliter la mise en place de cette architecture.

Pour la partie back-end, les services web RESTful ont été choisis pour leur capacité à fournir des ressources web qui peuvent être facilement consommées par d'autres applications. Cette approche facilite l'intégration de l'application avec d'autres systèmes. JHipster, le générateur d'application web utilisé, fournit une infrastructure pour générer des services web RESTful en utilisant Spring Boot et Spring Data.

Enfin, PostgreSQL a été choisi comme système de gestion de base de données pour sa capacité à gérer les données structurées et non structurées. Il est également open source et offre une grande stabilité et sécurité pour les données sensibles de la clinique.

Le schéma de l'architecture technique de l'application web est le suivant :

- Front-end : Angular
- Modèle : PostgreSQL
- Back-end : Spring Boot, Spring Data, JHipster
- Services web : RESTful

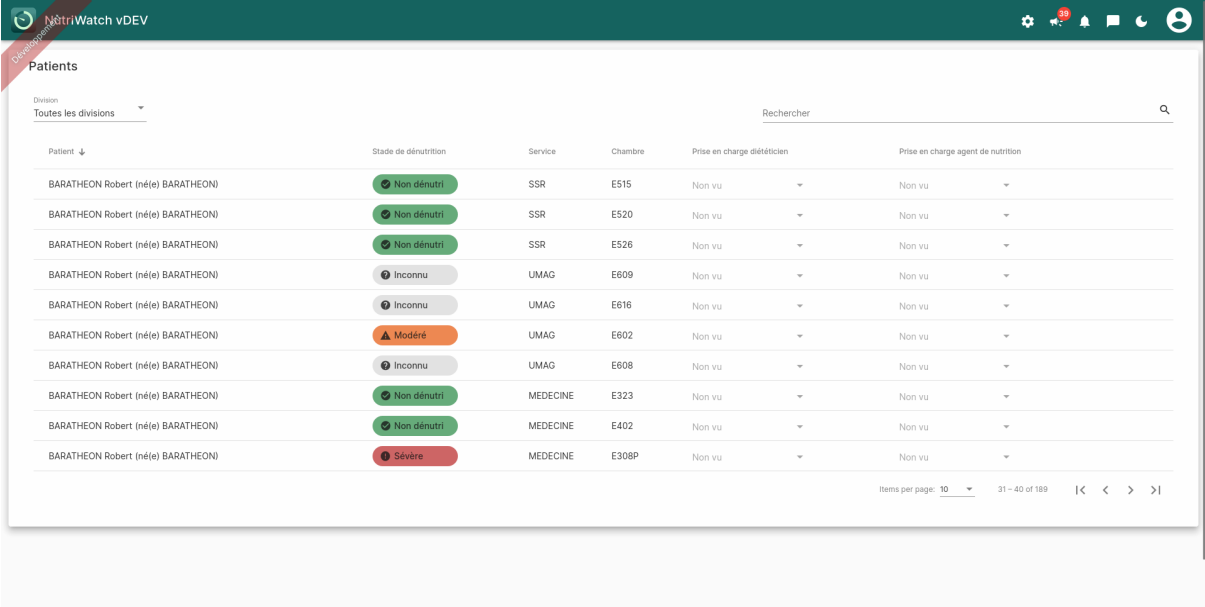
## IV) Réalisations techniques

La réalisation technique de l'application comprend plusieurs fonctionnalités destinées à répondre aux besoins de la clinique. Tout d'abord, l'automatisation de la mise à jour des patients permet de récupérer régulièrement les données des patients à partir d'un fichier CSV fourni par la clinique, sans nécessiter d'intervention manuelle. Cette partie a été longue à implémenter, notamment à cause des noms de colonnes ainsi que de l'encodage des valeurs contenues dans le CSV fourni par la clinique qui ont changé à plusieurs reprises, à chaque changement il était nécessaire de reprendre l'algorithme de parsing du CSV. Son lancement automatique a été cependant rapide à mettre en place grâce à Spring qui permet de configurer des tâches automatiques facilement à l'aide d'annotations.

A la suite du parsing, l'application calcule automatiquement l'état de dénutrition des patients à partir des données collectées, grâce à un algorithme de calcul de dénutrition à partir de l'IMC et de la perte de poids. A ce moment l'application en profite pour produire les statistiques journalières. L'application envoie également des alertes pour signaler aux médecins et infirmiers les cas de patients en état de dénutrition.

Les interfaces utilisateur ont été conçues pour être simples, intuitives et performantes, permettant aux utilisateurs de naviguer facilement à travers les différentes fonctionnalités de l'application. Les captures d'écran ci-dessous présentent les principales pages de l'application :

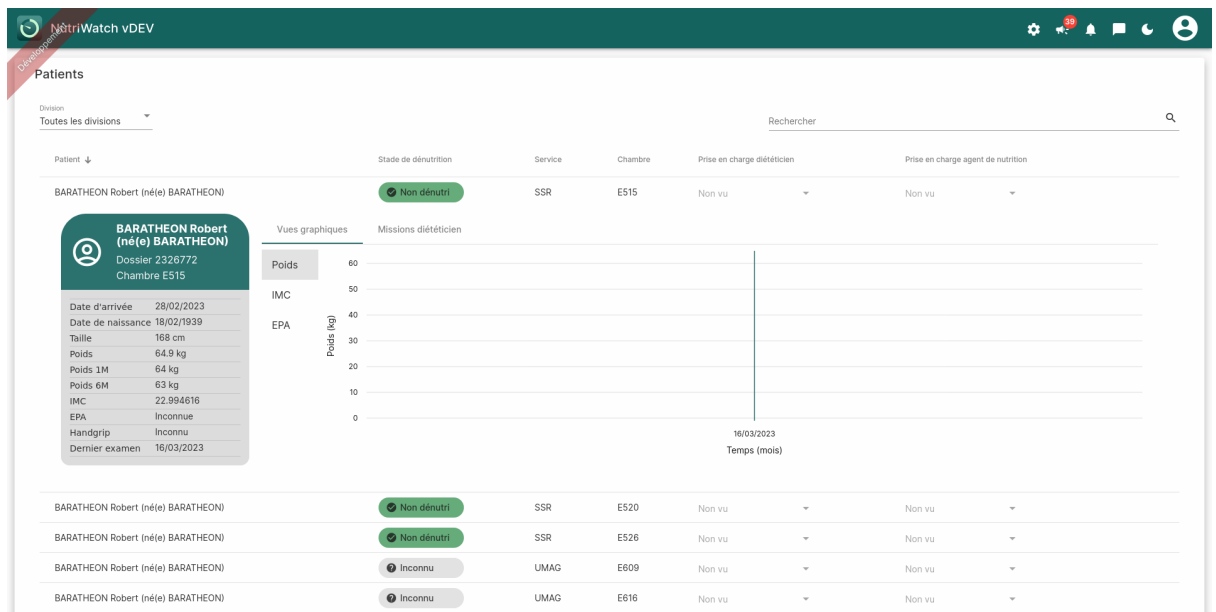
- La page d'accueil, qui affiche une vue d'ensemble de l'état de dénutrition des patients, avec la possibilité de trier et filtrer les résultats en fonction de différents critères (nom, état de dénutrition, service, etc...).



The screenshot displays the 'Patients' page of the NètriWatch vDEV application. The interface includes a header with the application name and a sidebar with a 'Patients' tab. The main content area shows a table of patients with columns for Patient, Stade de dénutrition, Service, Chambre, Prise en charge diététicien, and Prise en charge agent de nutrition. The table lists 10 patients, all with the name 'BARATHEON Robert (née) BARATHEON'. The nutritional status is indicated by colored circles: green for 'Non dénutri', orange for 'Modéré', and red for 'Sévère'. The table also includes columns for 'Service' and 'Chambre'. The bottom of the page shows pagination information: 'Items per page: 10' and '31 - 40 of 189'.

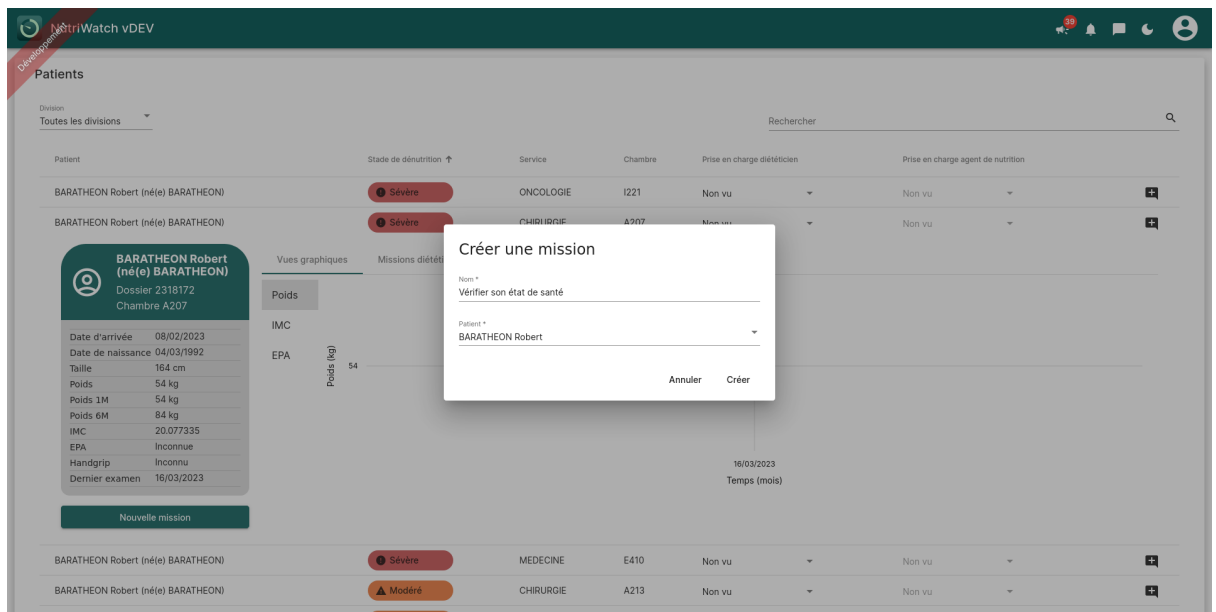
Patient	Stade de dénutrition	Service	Chambre	Prise en charge diététicien	Prise en charge agent de nutrition
BARATHEON Robert (née) BARATHEON	Non dénutri	SSR	E515	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Non dénutri	SSR	E520	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Non dénutri	SSR	E526	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Inconnu	UMAG	E609	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Inconnu	UMAG	E616	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Modéré	UMAG	E602	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Inconnu	UMAG	E608	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Non dénutri	MEDECINE	E323	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Non dénutri	MEDECINE	E402	Non vu	Non vu
BARATHEON Robert (née) BARATHEON	Sévère	MEDECINE	E308P	Non vu	Non vu

- La page de détail du patient, qui affiche toutes les informations relatives à un patient spécifique, y compris son état de dénutrition et l'historique de ses données.

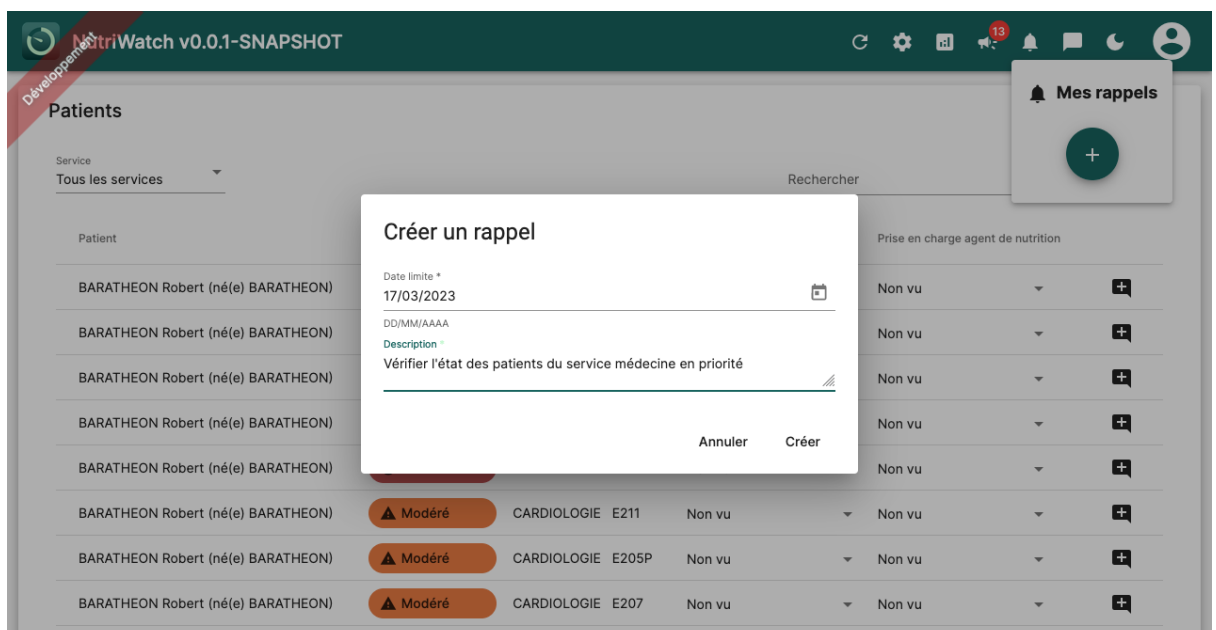


- La page de création de mission, qui permet aux diététiciens d'ajouter une nouvelle mission sur un patient.

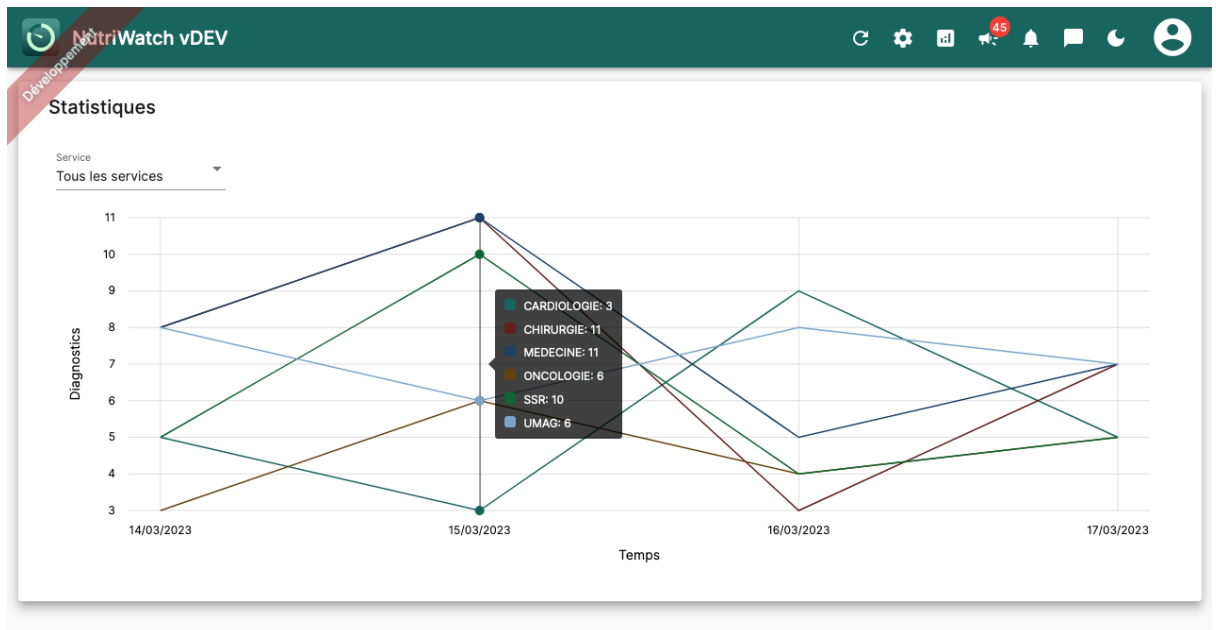




- La page de rappel qui permet à l'utilisateur de se programmer un rappel.



- La page de statistique qui permet de voir le nombre de diagnostique journalier par service



Ces pages ont été conçues avec un design simple et épuré, pour une utilisation intuitive et efficace par les utilisateurs de l'application.

Nous avons donc presque implémenté toutes les fonctionnalités correspondantes aux besoins vitaux et importants de l'hôpital, excepté l'export des données en fichier.

## V) Gestion de projet

En ce qui concerne les méthodes de gestion de projet, nous avons rapidement mis en place une méthodologie agile, sans toutefois réellement l'utiliser. Nous avons également effectué des réunions afin de mettre en commun ce qui a été réalisé par les membres du groupe. Ces réunions étaient fixées à l'avance, sans toutefois être récurrentes. Dans le cadre de notre projet nous avons utilisé plusieurs outils de gestion de projet pour faciliter la communication, la collaboration et la gestion des tâches. Ces outils comprennent Discord, Messenger, Trello et Git.

Pour ce qui est des rôles des membres de l'équipe, nous les avons répartis en fonction des connaissances de chacun.

Ainsi, Thomas s'est occupé du back-end et donc des routes d'API, du parsing du CSV des patients, du calcul de dénutrition, des statistiques journalières ou encore de la cohérence des données.

Théo a travaillé avec Loris sur le front-end, qu'il a fallu réadapter aux nouvelles contraintes de la clinique (qui diffèrent de celles d'un EHPAD, comme dans le projet eCOM).

Et Tom s'est chargé de toute la partie de mise en production, ce qui inclut la mise en place du Keycloak pour l'authentification, de la VM hébergée par la clinique et de l'assemblage de toutes les briques logicielles, afin d'assurer le déploiement de l'application web sur les serveurs du client.

Dans notre planning prévisionnel nous avons prévu de finir d'implémenter les fonctionnalités correspondantes aux besoins vitaux et importants une semaine avant la date de rendu, nous permettant de garder cette semaine pour une phase de test complet et de mise en production. Nous n'avons pas respecté ce planning, en effet nous nous sommes heurté à des difficultés imprévues ce qui nous a fait prendre du retard. Finalement à la date de rendu nous n'avons pas fini d'implémenter une fonctionnalité importante, nous n'avons donc pas pu tester l'application de manière aussi complète que nous le voulions au départ.

## VI) Outils

Pour la gestion de notre projet, nous avons utilisé Git comme outil de gestion de version et Github comme plateforme de collaboration en ligne. Cela nous a permis de travailler ensemble sur le même code source, de suivre les modifications et de résoudre les conflits éventuels. Git a également facilité la collaboration en équipe, car il nous a permis de gérer les conflits de code et de travailler de manière transparente et coordonnée.

Discord et Messenger ont également été utilisés pour la communication en temps réel entre les membres de l'équipe. Ces outils de messagerie instantanée nous ont permis de communiquer rapidement et efficacement, de partager des informations et des mises à jour sur le projet, et de discuter de toute question ou problème en cours. Nous avons également utilisé ces outils pour planifier des réunions (DM) et pour organiser des séances de travail en équipe.

Trello a été utilisé pour la gestion des tâches et la collaboration en équipe. Cette application a permis de suivre l'avancement de chaque tâche et de chaque étape du projet, de définir des échéances et des priorités, et de collaborer efficacement pour atteindre nos objectifs. Nous l'avons utilisé surtout au début du projet mais nous l'avons délaissé au fur et à mesure que nous avons avancé dans le développement.

En somme, ces outils de gestion de projet nous ont permis de travailler efficacement en équipe, de suivre l'avancement du projet et de gérer les tâches et les problèmes de manière coordonnée, tout ceci aussi bien en présentiel qu'en distanciel.

## VII) Métriques logicielles

Dans cette partie, nous allons présenter les métriques logicielles utilisées pour évaluer la qualité et les performances de l'application. Tout d'abord, nous avons analysé la répartition des lignes de code et des commits entre les membres du projet. Nous avons utilisé Git pour suivre les contributions de chaque membre du projet et établir un pourcentage de la participation de chacun, néanmoins nous n'avons pas trouvé cela représentatif du travail de chacun et nous allons argumenter pourquoi.

Il n'est pas intéressant pour notre projet de regarder le nombre de lignes de codes. En effet, Jhptser générant un très grand nombre de lignes de code, cela fausse totalement les chiffres. De plus le nombre de commits n'est pas aussi un bon indicateur, en effet les personnes travaillant sur le devOps et le déploiement vont peu apparaître sur ces derniers alors qu'ils travaillent tout autant que les autres. Pour finir, des membres du groupe devait s'occuper de la communication avec les équipes de l'hôpital ou de la gestion de projet, ce travail n'est pas quantifiable en lignes de code ou en nombre de commits, c'est pour cela que nous ne voulons pas nous baser sur ces métriques pour estimer la répartition de la quantité de travail produit entre les membres du groupe.

## VIII) Conclusion

La réalisation du projet de suivi de la dénutrition des patients de la clinique a été un succès qui reste toutefois améliorabile. Nous avons réalisé la majorité des tâches centrales de l'application, néanmoins, l'exportation des données en fichier, qui correspond à un besoin important de l'hôpital n'a pas été implémenté. La collaboration entre les membres de l'équipe a été productive, avec une répartition équilibrée des tâches et des responsabilités. Néanmoins, nous n'avons pas respecté le planning prévisionnel établi. Cela est dû à des tâches qui ont pris plus de temps que prévu ainsi que des difficultés que nous n'avons pas anticipé (format du CSV donné par la clinique et modifié plusieurs fois, modification multiple du JDL). Le choix des technologies employées (Java, JHipster, Angular) a été judicieux, avec des avantages clairs pour chaque technologie, notamment la facilité de développement et la performance de l'application ainsi que la connaissance de chacun des membres de l'équipe qui nous a permis de nous mettre au travail. Cependant, la rigidité de JHipster dû à la génération du code via JDL a entraîné des difficultés et des ralentissement dans la production de l'application. L'utilisation d'autres frameworks web comme Symfony ou Laravel aurait pu être intéressant. De son côté, l'architecture de l'application basée sur le modèle MVC et les services web RESTful a été très efficace, avec une base de données bien conçue pour stocker les données des patients.

Même si toutes les fonctionnalités ne sont pas implémentées, celles qui le sont, telles que l'automatisation de la mise à jour des patients, le calcul automatique de l'état de dénutrition, des statistiques et des alertes, ont été assez bien conçues et répondent aux exigences du cahier des charges. Les interfaces utilisateur sont claires et intuitives.

En conclusion, ce projet a été une réussite pour la clinique commanditaire, avec une application répondant à la majorité des exigences du cahier des charges. Des améliorations pourraient être apportées pour améliorer encore davantage l'application, notamment en ajoutant les fonctionnalités non-implémentées. Nous tenons à remercier la clinique commanditaire pour cette opportunité et tous les membres de l'équipe pour leur contribution à la réalisation de ce projet.