



## BACHELIER EN INFORMATIQUE DE GESTION

*Stage Medsoc 2021 – Projet eBoarding*

Rapport de stage effectué dans la firme

*Medsoc*

*Aberkan Yassin* Juin 2021



# Sommaire

## Table des matières

1. Introduction .....	2
2. Environnement .....	3
2.1 Système Informatique d'un Hôpital .....	3
2.2 Fondation Entreprise .....	4
2.3 Organisation Entreprise .....	4
2.4 Gestion de projet .....	5
2.5 Environnement de travail .....	6
3. Projet eBoarding .....	7
3.1 Introduction .....	7
3.2 Présentation du projet .....	9
3.2.1 Section du médecin .....	10
3.2.2 Agenda .....	11
3.2.3 Ajout d'une visite .....	12
3.2.4 Boarding .....	14
3.2.5 Envoie Prestation en facture .....	15
3.2.6 Prestation encodée pour la visite .....	16
4. Outil et Architecture .....	17
4.1 Outils utilisés .....	17
4.2 Architecture .....	22
4.2.1 Frontend .....	22
4.2.2 Backend .....	24
4.2.3 Architecture Hybrid .....	26
5. Sprint .....	27
5.1 Sprint 1 .....	27
5.2 Sprint 2 .....	31
5.3 Sprint 3 .....	34
6. Conclusion .....	36
4. Annexes .....	37
7.1 Bibliographie .....	37
7.2 Carnet de bord .....	38
7.3 Evaluation continue .....	47

## 1. Introduction

Du 15 février 2021 au 28 mai 2021, j'ai effectué un stage au sein de Medsoc (située à Wavre). Durant ce stage, j'ai pu approfondir mes connaissances au développement d'applications web et j'ai également pu apprendre davantage sur les conditions et attentes requises afin de produire une application web dédié au médecin.

Dans la suite de ce rapport, je présenterai plus en détail : le contexte, l'entreprise, mon projet, mes outils de travail et le travail fourni durant ces 3 mois.

### Remerciement

Je tiens tout particulièrement à remercier mon Maître de Stage, Noé Pottiez, pour son accueil, le temps passé ensemble et le partage de son expertise et de ses conseils au quotidien. Grâce aussi à son aide, j'ai pu totalement accomplir mes tâches au sein de l'entreprise. Il fut d'une aide précieuse dans les moments les plus délicats.

Je tiens également à remercier toute l'équipe Medsoc de m'avoir accueilli si chaleureusement dans leurs rangs, mais également de répondre à toute question de ma part si gentiment.

Enfin, je tiens à remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce rapport de stage : ma famille et mes amis Mathieu DEPRETER et Sidra DADA, camarades de promotion.

## 2. Environnement

Dans cette partie, nous allons passer en revue l'environnement dans lequel mon stage s'est déroulé. C'est-à-dire, l'entreprise, l'organisation et la présentation du matériel utilisé.

### 2.1 Système Informatique d'un Hôpital

Pour une meilleure compréhension de ce que fait l'entreprise, il est tout d'abord important de connaître quelques notions du système informatique d'un hôpital.

La gestion d'un hôpital est séparée en 2 grandes branches :

- **ADT** (trigramme anglo-saxon : Admission, Discharge, Transfert) : Les tâches administratives d'un hôpital qui concerne la gestion administrative des patients, de l'admission à la facturation.
- **DPI** (trigramme : Dossier Patient Informatisé) : Cette partie concerne la partie, médical, infirmier et paramédical du patient.

Ces deux parties bien distinctes s'échange les informations à travers des fichiers HL7. Le format HL7 est une norme internationale, qui garantit un niveau de détail élevé dans la transmission des informations médicales. Ainsi, il permet des échanges fiables concernant l'identité des patients.

#### 2.1.2 Système Informatique d'un Hôpital : Facturation ADT

La facturation d'un patient est un élément important de l'ADT.

Plusieurs notions sont à prendre en compte pour la facturation d'un patient, telle que la visite, les prestations, etc.

Une visite est un examen médical entre un patient et un ou plusieurs médecins. On y retrouve les visites polyclinique et les visites hospitalières.

- Une visite polyclinique ou ambulatoire est une consultation chez un prestataire des soins de santé (médecin, paramédical).
- Une visite hospitalière comprend au moins une nuit. Il y a une hospitalisation lorsqu'il n'est pas possible de manière ambulatoire de poser un diagnostic, d'appliquer une thérapie.

Par exemple : Un patient qui se rend chez le médecin pour un mal de tête est une visite polyclinique. Un patient qui se fait opérer pour une raison quelconque est une visite hospitalière.

Une prestation est un acte médical ou un bien fourni au patient lors d'une visite.

Par exemple : Une endoscopie médicale qui consiste à un examen spécifique pour un organe du patient, se dit une prestation endoscopique.

## 2.2 Fondation Entreprise

Medsoc est une association à but non lucratif créé en 1973 en Belgique. Cette entreprise a été fondée par le développement informatique au sein des mutualités chrétiennes afin de faciliter les mains d'œuvres hospitalière.

Par la suite, en 2000, le développement informatique s'est détaché des mutualités chrétiennes pour créer un nouveau statut d'ASBL qui est aujourd'hui Medsoc. Aujourd'hui, son siège social se trouve désormais à Wavre.



Figure 1 : Logo de Medsoc

## 2.3 Organisation Entreprise

Medsoc a donc pour ambition de fournir des solutions logicielles qui permettent aux hôpitaux belges d'optimiser leurs ressources et la gestion de données administratives. Ces solutions sont destinées aux hôpitaux, cliniques mais aussi aux centres de soins belges.

Ainsi, Medsoc est organisé en 2 équipes principales : Sofia et Apo.

Sofia fournit des logiciels dédiés à la tarification et facturation pour les hôpitaux mais également un suivi patient.

Deux logiciels sont d'application pour la branche Sofia :

- Sofia Invoice, solution de tarification et facturation.
- Sofia ADT, traite un grand nombre de tâches administratifs pour les patients.

Apo a pour but de faciliter le département achat et d'optimiser la gestion des pharmacies hospitalière. Un logiciel principal et un autre additionnel est d'application pour la branche Apo :

- Apo, logiciel pour une gestion optimale des pharmacies hospitalière.
- Echo, facilite les tâches de département achat qui est synchronisé avec Apo.

Les solutions Medsoc ont été élaborés sur base des besoins du secteur hospitalier en partenariat avec les hôpitaux.

## 2.4 Gestion de projet

L'équipe de développement utilise une méthodologie assez connue, l'Agile.

Scrum est un Framework qui est utilisé pour implémenter la méthode Agile de développement et de gestion de projet. C'est une implémentation vue en cours.

Jira est l'outil utilisé pour exploiter les fonctionnalités du Scrum.

Scrum consiste à découper un projet en sprints (boîte de temps).

Chaque sprint dure généralement 1 mois. Dans un Sprint, la quantité de travail prévue doit être effectuée par l'équipe et préparée pour la fin du sprint en cours.

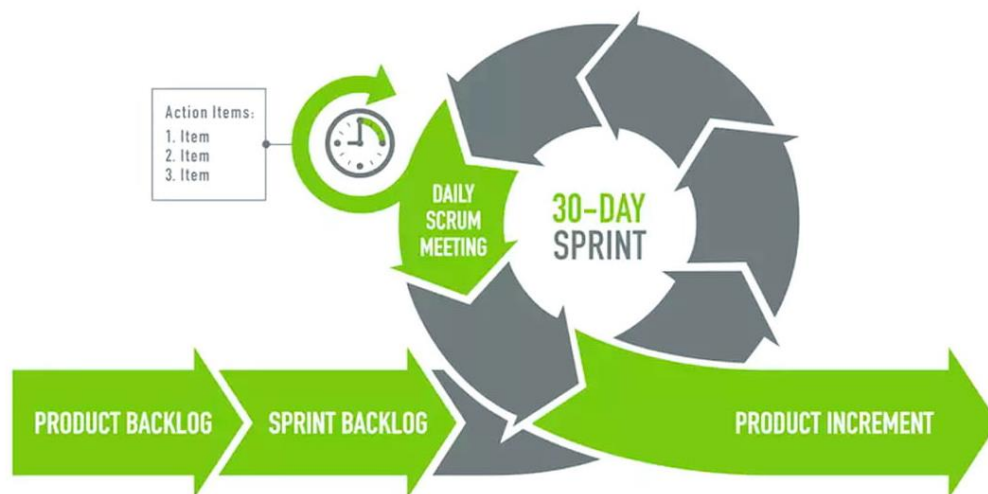


Figure 2 : Schéma Scrum

Chaque début de mois, toute l'équipe se réunit pour la planification du sprint et pour la rétrospective du Sprint précédent.

La rétrospective de Sprint est certainement celle qui peut être le principal vecteur d'amélioration continue. Chacun des développeurs qui ont participé au sprint discute librement des tâches effectuées au sprint précédent.

La réunion de planification du sprint est l'une des étapes les plus importantes d'un projet Scrum. Lors de cette réunion, l'équipe de développement sélectionne les éléments prioritaires du Product Backlog qu'elle pense pouvoir réaliser au cours du sprint.

Cette planification aboutit à la création d'un plan de sprint. C'est un travail collaboratif de toute l'équipe Scrum.

De plus, chaque jour a lieu le Daily Scrum. Le Daily Scrum est une réunion de synchronisation quotidienne. Elle permet de discuter de l'avancement des tâches, d'éventuels problèmes, de faire marcher l'entraide et résoudre les blocages possibles.

## 2.5 Environnement de travail

À mon arrivé, un bureau personnel m'a directement été attribué dans les locaux open-space de Medsoc.

L'équipe est composée de mon maitre de stage ainsi que 5 autres développeurs de la branche Sofia de l'entreprise.

Un ordinateur portable personnel m'a été fourni afin de travailler 4 jours semaine au bureau et 1 jours semaine en télétravail.

J'ai eu de la chance de travailler une grosse partie de mon stage en présentiel, cela me permettait d'avoir une réelle perspective sur l'environnement de travail.

Cela a également grandement facilité la communication pour d'éventuelle questions ou échanges. Ça m'a aussi permis d'améliorer l'interaction et en bien-être entre l'équipe de développement et moi.

Durant les 2 premiers mois, le lundi je travaillais en télétravail et le reste de la semaine en présentiel dans leur bureau.

Cependant, les mesures covid ont changé durant la période de mon stage nous obligeant à travailler beaucoup plus en distanciel. A partir de ce moment, j'ai travaillé 4 jours en distancielle et 1 jour en présentielle.

Pour ma part, j'ai eu de la chance de mettre en pratique mes savoirs pour le distanciel.

## 3. Projet eBoarding

### 3.1 Introduction

L'un des besoins demandés par plusieurs hôpitaux belges est de faciliter la tâche lors de l'encodage des prestations lors d'une consultation.

En effet, pour le moment, lorsqu'un patient termine une consultation avec son médecin. Le médecin coche sur papier toutes les prestations à encoder durant la visite pour la passer ensuite à une secrétaire médicale.

La secrétaire médicale l'encode sur une application Boarding compliqué d'utilisation.

Le but est d'offrir une solution simple, intelligente et facile d'utilisation pour les médecins. De plus, cela permettra avant tout de fournir moins de travail au secrétaire médicales mais surtout éviter le double encodage.

La solution la plus efficace pour Medsoc est de fournir une application web dédié au médecin ou au sein de l'application, il pourra directement encoder les prestations pour une visite sans procéder à un double encodage chez la secrétaire médicale.

Cette application doit être compatible à tous les écrans possibles de sorte qu'elle soit utilisable sur ordinateur, tablette, smartphone, etc.

En effet, l'application doit fournir de la manière la plus simple possible l'ensemble des prestations encodables sur la visite.

En effet, l'application de façon intelligente doit fournir toutes les prestations possibles à encoder lors de la visite.

Ainsi, le médecin devra juste cliquer sur les prestations qu'il souhaite encoder et le tout est automatiquement envoyé en pré-facturation.

A la page suivante, voici un exemple de prestations lié à une visite que le médecin donne à la secrétaire médicale sur papier.

Un boarding est une liste de prestation qui dépend du médecin.



BOARDING PASS		Clinique Saint Pierre	2021-01-29 16:01:00
Boarding : [REDACTED]		Patient : [REDACTED]	
Visite : [REDACTED]		Médecin : [REDACTED]	
Service : [REDACTED]			
Régime : 01 [REDACTED]			
Commentaire :			
Acompte : 0,00			
Médecin prescripteur : 12780937004 [REDACTED]			
		Qté.	Qté.
<input type="checkbox"/> A1	102.012 CONSULT.CHEZ UN SPECIALISTE	<input type="checkbox"/>	
<input type="checkbox"/> A2	475.075 EX. E.C.G. MIN.12 DERIV.+PROT.	<input type="checkbox"/>	
<input type="checkbox"/> A3	475.812 EPR.EFFORT +MONITOR.CONT. IDEI	<input type="checkbox"/>	
<input type="checkbox"/> A4	469.814 ECHO CARDIAQUE	<input type="checkbox"/>	
<input type="checkbox"/> A5	469.630 ECHO CARDIAQUE BIS	<input type="checkbox"/>	
<input type="checkbox"/> A6	476.210 HOLTER 24 H + PROT + CONSULT	<input type="checkbox"/>	
<input type="checkbox"/> A7	476.232 REPET. 476210 HOLTER DELAI 1AN	<input type="checkbox"/>	
<input type="checkbox"/> A8	476.254 R. TEST y compris CONSULTATION	<input type="checkbox"/>	
<input type="checkbox"/> A9	476.335 TILT-TEST A 60 DEGRES MIN. 45'	<input type="checkbox"/>	
<input type="checkbox"/> A10	475.856 REPROGRAM.STIM.CARD.CH.SIMPL.	<input type="checkbox"/>	
<input type="checkbox"/> A11	475.871 REPROGRAM.STIM.CARD.CH.DOUBI	<input type="checkbox"/>	
<input type="checkbox"/> A12	475.893 REPROGRAM.STIM.ANTITACHYCAR	<input type="checkbox"/>	
<input type="checkbox"/> A13	471.273 SPIROGR. + EPR.BRONCHODILATAT.	<input type="checkbox"/>	
<input type="checkbox"/> A14	471.391 ERGOSPIROMETRIE	<input type="checkbox"/>	
<input type="checkbox"/> A15	475.532 EPR.PHARMACODYN.+CONTR.E.C.G.	<input type="checkbox"/>	
<input type="checkbox"/> A16	469.836 ECHO TRANSOESOPHAGIEN	<input type="checkbox"/>	
<input type="checkbox"/> A17	469.954 Stress-test cardiaque par échographie au n°	<input type="checkbox"/>	
<input type="checkbox"/> A18	99911 COPIE DOSSIER(0.1 par feuille, Max 25	<input type="checkbox"/>	
<input type="checkbox"/> A19	99049 Supplément Expédition Facture	<input type="checkbox"/>	
<input type="checkbox"/> A20	99052 Pénalité Absence RV/No Show	<input type="checkbox"/>	
Autres codes :			
Remarques :			
<input type="checkbox"/> Nuit	<input type="checkbox"/> Week-end	<input type="checkbox"/> Jour férié	Cachet, date et signature du médecin :
Supplément d'honoraire :			

Figure 1 Boarding Papier Exemple

## 3.2 Présentation du projet

Mon projet est donc une application web dédiée aux médecins et aux secrétaires médicales. Une fois que l'utilisateur est connecté et validé en tant que médecin ou secrétaire médicale, la page web principale s'affiche comme ci-dessous.

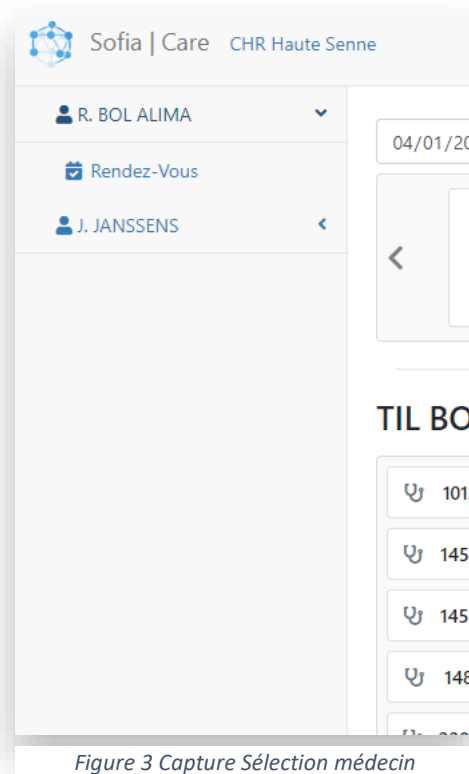
The screenshot displays the eBoarding Application interface. At the top, the header shows 'Sofia | Care' and 'CHR Haute Seine'. Below the header, there is a navigation bar with user profiles: 'R. BOL ALIMA' and 'J. JANSSENS'. The main content area is divided into two sections. The top section shows a consultation schedule for '04/01/2019'. It lists several doctors and their consultation times: YASSIN ABERKAN (09:10), ETIENNE RYCKMANS (09:48), TOM HARDY (11:07), TOM CRUISE (11:43), RAMI MALEK (13:41), WILL SMITH (14:43), and LEONARDO DICAP. (15:20). The bottom section, titled 'TIL BOL ALIMA', displays a list of medical procedures with their corresponding codes and descriptions. On the right side, there is a section for 'Prestations encodées' (Encoded Procedures) showing two entries: '144071' and '144071'.

Code	Description
101290	CONSULTATION AU CABINET PAR UN
145272	PANS.DERMATO COMPLI LES.ETEND
145574	INC.PHLEGMON SUPERF.OU ANTHRAX
148116	SUTURE FILS PAS FACE +/- 3 PLA
220253	CURE CHIRUR. PHLEGMON PROFOND
238195	REC.CROS.SAPH.INT.+ LIG.VELVAR
355692	PONCT.ORG.HEMATOP.SF FOIE-RATE
469232	EXAMEN DUPLEX COULEUR DES VAIS
469733	ECHO CARDIO ARTERES CAROTIDES+ VERTEBRALES
469770	ECHO CARDIO VAIS.SANG. MEMBRES
144071	INI.SCLEROS.VARICES /SEANCE
145515	EXTR. C.E. S/APON. SAUF OEIL
148094	SUTURE FILS PAS FACE 1 OU 2 PL
220231	EXTRACT. C.E.PROFOND.DS TISSUS
238114	LIG/REC.ETAG. +3 VEIN.VARIQU.
350070	CONSULTATION DE LONGUE DURÉE (
469210	MEN DUPLEX COULEUR DES VAISSEA
469711	ECHO CARDIO ARTERES CAROTIDES
469755	ECHO CARDIO VAIS.SANG.
30145	DOPPLER

Figure 2 eBoarding Application

### 3.2.1 Sélection du médecin

L'un des premiers éléments que je souhaite aborder est la sélection du médecin qui mènera par la suite aux fonctionnalités de l'application.



Sous la colonne de gauche, l'application affiche tous les médecins qui sont liés à l'utilisateur connecté.

C'est-à-dire que lorsqu'un médecin se connecte, il n'y aura que son propre médecin affiché.

Mais on peut supposer qu'il peut administrer un stagiaire. Dans ce cas : 2 médecins seront affichés (le médecin et le stagiaire).

Si une secrétaire médicale gère plusieurs médecins. La secrétaire médicale pourrait se connecter sur l'application et gérer l'encodage de prestations d'un ou de plusieurs médecins en particulier.

Lorsqu'on clique sur le médecin correspondant, une liste déroulante s'affiche avec comme entrée « Rendez-Vous ».

Cette entrée permet d'accéder aux visites du médecin pour procéder à l'encodage de prestation en état de facturation qui est l'élément principal de mon projet.

Pour l'instant il n'y a qu'une fonctionnalité disponible (mon projet eBoarding) mais on laisse les portes ouvertes pour d'éventuels autres fonctionnalités dédiées à un médecin qui sera disponible via cette application Web.

### 3.2.2 Agenda

L'agenda est l'un des 3 éléments principale du projet.

Celle-ci affiche sur une liste déroulante, des visites liés aux médecins en fonction de la date sélectionnée.

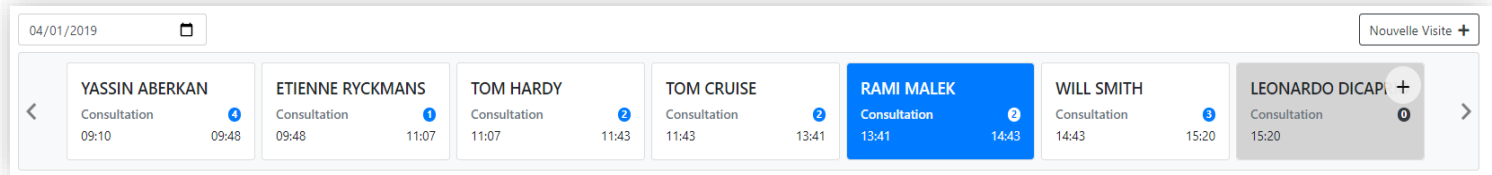


Figure 4 Agenda ensemble

L'application récupère toutes les visites existantes ou non du médecin correspondant.

Ainsi pour accéder à une visite particulière le médecin devra cliquer sur l'une des cartes qui représente une visite bien distincte.

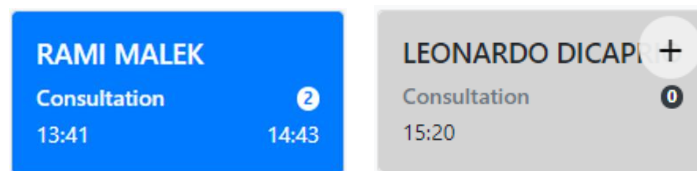


Figure 5 Agenda Carte

Une carte représente toutes les informations nécessaires a différencié une visite d'une autre.

On y retrouve :

- Le prénom et nom du patient
- Le type de visite
- L'heure du début et de fin du rendez vous
- Le nombre de prestation envoyé en facture pour la visite

Une carte s'affiche sous 3 couleurs différentes :

- ☐ Les visites ou le patient s'est déjà enregistré au guichet par exemple sera accessible par le médecin.
- ☐ La carte est grisée et inaccessible tant que le patient ne sera pas enregistré.
- ☒ La carte est bleue si la visite affichée est celle de la carte.

### 3.2.3 Ajout d'une visite

Il est également possible pour le médecin d'ajouter une visite, pour cela il y a deux façons qui s'offrent à lui :

- Création d'une visite qui est déjà planifié dans l'Agenda

Chaque carte grisée propose un bouton « + » qui permettra de créer une visite. Toutes les informations nécessaires sont déjà reconnues grâce aux informations fournit par l'agenda.

Ainsi le simple fait d'appuyer sur le bouton « + » permettra de crée la visite. Cette fonctionnalité a été demandé pour les mesures lié au covid-19.

Par exemple, le patient n'est plus forcément obligé de se rendre au guichet pour aller en consultation, mais peut directement aller chez le médecin.

- Création d'une nouvelle visite qui n'est pas planifié

Le bouton « Nouvelle Visite » permet de créer une nouvelle visite sans aucune information reçue au préalable.

Pour cela un premier écran apparait pour rechercher un patient qui sera lié à la nouvelle visite.

The screenshot shows a web application interface for creating a visit. At the top, there's a progress bar with two steps: 'PATIENT' (active) and 'PARAMETRES'. Below this, there are two search methods: 'Recherche par identifiant de patient' and 'Recherche personnalisée patient'. The 'Recherche personnalisée patient' section has input fields for 'yassin' and 'nom'. A 'Rechercher' button is present. Below the search section, a list of patients is displayed in a grid. Each patient card shows their ID, name, registration number, date of birth, and phone number. The patients are paginated, with a 'Suivant' button at the bottom right.

Patient 67177	Patient 72124	Patient 74977	Patient 139740
Registre	Registre	Registre	Registre
Date de Naissance : Oct 7, 1983	Date de Naissance : Aug 25, 1989	Date de Naissance : Dec 28, 1992	Date de Naissance : Feb 6, 1999
Téléph	Téléph	Téléph	Téléph

Patient 158441	Patient 163246	Patient 166040	Patient 210966
N	N	N	N
Registr	Registr	Registr	Registr
Date de Naissance : Sep 24, 1996	Date de Naissance : May 11, 1988	Date de Naissance : Apr 4, 1998	Date de Naissance : Nov 10, 1992

Figure 6 Capture recherche patient

Les patients sont affichés en pagination via des cartes ou soit sous forme de liste.

Une fois que le patient est sélectionné, un second écran apparaît pour savoir l'heure du rendez-vous ainsi que le boarding que le médecin va utiliser pour la visite.

Pour finir, quand toutes les informations sont complétées, la visite peut être créée par l'utilisateur. Ainsi, une nouvelle carte apparaît dans la liste des visites de l'Agenda.

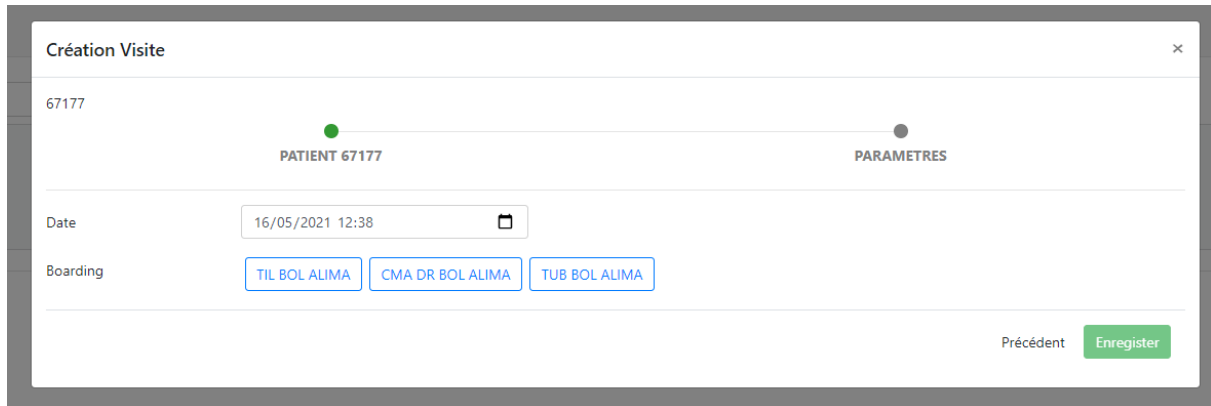


Figure 7 Capture paramètre Visite

La création d'une nouvelle visite a été mise en place pour consulter des patients via téléphone sans réel contact. En effet, il est impératif d'adapter l'application eBoarding en fonction des mesures sanitaires imposées.

### 3.2.4 Boarding

Un boarding représente l'ensemble des prestations du médecin qui peuvent être encodée lors d'une visite.

Par exemple pour un dentiste, le boarding représente l'ensemble des instruments dentaires ainsi que les services qui peuvent être effectués pour un patient.

Sur papier, le boarding est représenté sous forme de liste où le médecin coche chacune des prestations utilisées lors de la visite.

(Voir Figure 2 Boarding Papier Exemple)

Sur l'application Web le boarding est également représenté sous forme de liste où chacune des prestations affichées est cliquable.

#### TIL BOL ALIMA

101290	CONSULTATION AU CABINET PAR UN	0	144071	INJ.SCLEROS.VARICES /SEANCE	2
145272	PANS.DERMATO COMPLI. LES.ETEND	0	145515	EXTR. C.E. S/APON. SAUF OEIL	0
145574	INC.PHLEGMON SUPERF.OU ANTHRAX	0	148094	SUTURE FILS PAS FACE 1 OU 2 PL	0
148116	SUTURE FILS PAS FACE +/- 3 PLA	0	220231	EXTRACT. C.E.PROFOND.DS TISSUS	0
220253	CURE CHIRUR. PHLEGMON PROFOND	0	238114	LIG./REC.ETAG. +3 VEIN.VARIQU. ↔	0
238195	REC.CROS.SAPH.INT.+LIG.VELVAR ↔	0	350070	CONSULTATION DE LONGUE DURÉE (	0
355692	PONCT.ORG.HEMATOP.SF FOIE-RATE	0	469210	MEN DUPLEX COULEUR DES VAISSEA	0
469232	EXAMEN DUPLEX COULEUR DES VAIS	0	469711	ECHO CARDIO ARTERES CAROTIDES	0
469733	ECHO CARDIO ARTERES CAROTIDES+VERTEBRALES	0	469755	ECHO CARDIO VAIS.SANG.	0
469770	ECHO CARDIO VAIS.SANG. MEMBRES	0	+ 30145	DOPPLER	0

Figure 8 Capture Boarding

Une prestation représente l'ensemble des informations nécessaires pour différencier une prestation d'une autre.

238114	LIG./REC.ETAG. +3 VEIN.VARIQU. ↔	0
--------	----------------------------------	---

Figure 9 Capture Prestation

Une prestation a pour information :

- Le type de prestation
- Le code Inami de la prestation
- La description du code Inami
- Indicateur paramétrés nécessaires
- Nombre de fois que la prestation est encodée pour la visite

En effet, il existe plusieurs types de « facturable » :

- ProvidedCare : Tous les types de soins prestés au patient
- ProvidedMisc : Tous les choses divers qui peut être prescrit au patient, par exemple, une bouteille d'eau
- ProvidedBlood : Toute la quantité de sang à facturer
- ProvidedPharma : Tous les médicaments qui sont presté pour une visite

Les « facturable » qui peuvent être encodé dans un boarding sont les ProvidedCare et les ProvidedMisc.

Les prestations de type « ProvidedBlood » ne sont pas utilisé pour une visite mais ceux de la pharmacie peuvent être utilisé, cela rend la gestion des stocks complexe a traité.

Un indicateur paramétré est parfois nécessaire, cet indicateur est affiché via une icône. Par exemple, certaine prestation nécessite le numéro de dent exploité, une icône « ↔ » est affiché. Il est aussi demandé quel œil correspond.

### 2.2.5 Envoie Prestation en facture

Pour envoyer une prestation en facture pour la visite sélectionne, il suffit de cliquer sur la prestation correspondante.

Mais certaine prestation demande des informations supplémentaires qui sont indiqué via une icône.

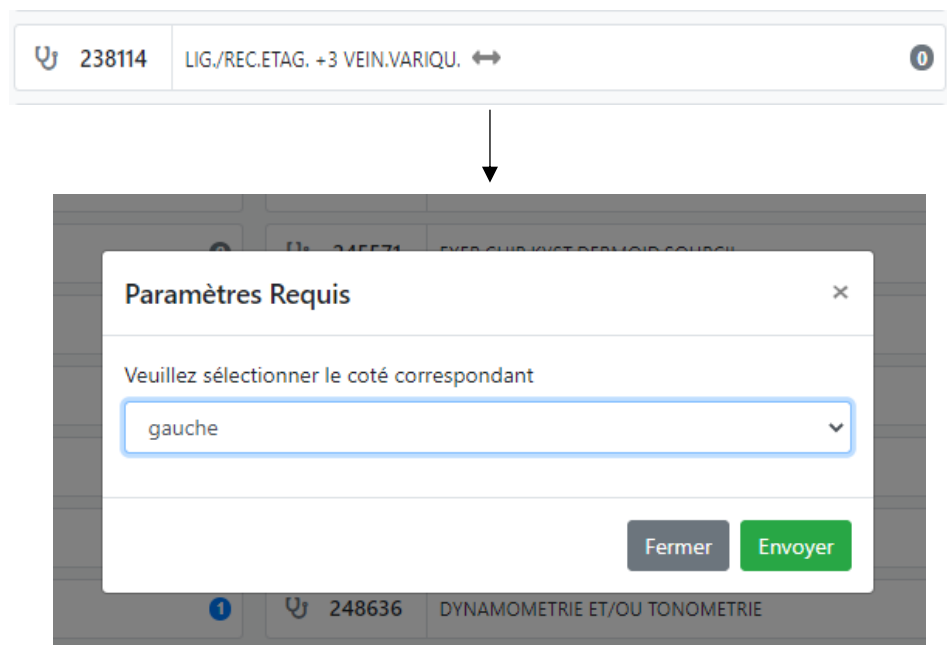


Figure 10 Capture Paramètres Requis



### 3.2.6 Prestation encodée pour la visite

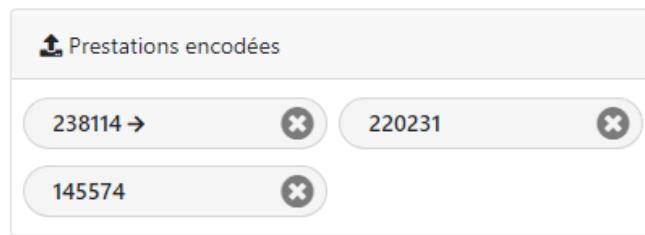


Figure 11 Capture Prestation encodées

Les prestations qui sont en état « à facturer » ou « déjà facturé » sont affichées ci-dessus.

Il est également possible de supprimer une prestation en attente de facturation, les prestations déjà facturées sont affichées en vert et ne peuvent être supprimées.

De plus, les prestations avec un paramètre nécessaire est identifiables via une icône : ici pour le code inami 238114, le côté droit a été sélectionné.

Pour les dents une icône dent est affichée avec le numéro qui lui correspond.

## 4. Outils et Architecture

### 4.1 Outils utilisés

Les outils utilisés pour les projets Sofia et Apo sont différents, étant donné que le projet qui m'a été attribué fait partie intégrante de la branche Sofia, je ne commence donc pas un projet à zéro.

Les outils sont donc imposés pour qu'elle puisse être rajoutée par la suite dans Sofia.

Les technologies utilisées sont :

- Java (langage backend)
- Java Spring (framework backend)
- Maven (gestion de projets Java)
- Angular (frontend)
- Bootstrap (frontend – css)
- HTML, CSS, JavaScript (frontend)
- TypeScript (application bureau, frontend)
- NPM (gestionnaire de paquets)
- Oracle (base de données)
- Swagger (Documentation API)
- Subversion (gestionnaire des versions)
- Jenkins (outil intégration continu)
- Nexus (gestion de dépôts)
- IntelliJ IDEA (éditeur de code backend)
- WebStorm (éditeur de code frontend)

#### 4.1.1 Java

« Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton.

Une particularité de Java est que les logiciels écrits dans ce langage sont compilés vers une représentation binaire intermédiaire qui peut être exécutée dans une machine virtuelle Java (JVM) en faisant abstraction du système d'exploitation. »



Java est utilisé pour le moteur de facturation de Sofia, également en lien avec Java Spring pour la programmation backend.

Toutes mes manipulations backend ont été fait en Java

### 4.1.2 Java Spring

La partie backend est orchestrée à l'aide du Framework Spring.

Spring Framework offre de nombreuses fonctionnalités de base pour développer des applications.



Plusieurs de ces modules sont utilisés tels que :

- Spring Core, c'est l'élément de base utilisé par tous les autres. Ce qu'il apporte de novateur et qui a fait son succès c'est l'injection de dépendance. Lorsqu'une classe a besoin d'une autre.
- Spring Web, propose des fonctionnalités de base pour les développements web (initialisation du conteneur, gestion des contextes, extraction des paramètres d'une requête http, ...) en respectant la conception MVC.

### 4.1.3 Maven

Maven est un outil permettant d'automatiser la gestion de projets Java.



Les fonctionnalités qui sont fournies et utilisées sont les suivantes :

- Compilation et déploiement des applications Java (JAR)
- Gestion des librairies requises par l'application. En effet, de nombreuses librairies sont produites et partagées en interne.
- Exécution des tests unitaires

### 4.1.4 Angular

Mon application web a été développée grâce au framework Angular (framework frontend open source développé par Google).



Il permet la création d'applications Web et plus particulièrement de ce qu'on appelle des « Single Page Applications », via des outils de routing ou de gestion du http, des composants qui facilite la communication père fils, etc.

### 4.1.5 TypeScript

TypeScript est utilisé pour la programmation frontend à l'aide du Framework Angular.

TypeScript s'inspire de langages de programmation orientée objet JavaScript et tend notamment à le sécuriser.



TypeScript est du JavaScript typé, ce qui veut dire qu'on peut rajouter des types à des variables, des types à des arguments de fonction ou encore des énumérations, des interfaces, etc... Ce qui permet de structurer notre javascript et de ne pas exécuter du code avec des erreurs de syntaxes.

Le code est écrit en TypeScript mais une fois transpilé (compiler un code d'un langage à un autre) ce sera du code javascript qui sera exécuté.

### 4.1.6 NPM

« NPM est l'abréviation de Node Package Manager, qui est un outil (programme) gérant les bibliothèques de programmation Javascript pour Node.js. Cet outil est réellement nécessaire pour le monde open source. »



NPM permet donc principalement de télécharger des librairies écrites par la communauté mais également les librairies partagées en interne. Exemple de librairies téléchargées pour l'application : Bootstrap, les api de Sofia, etc.

### 4.1.7 Subversion

Subversion est un logiciel de gestion de version qui est apparu en 2000 sous licence Apache.

Elle repose sur un système de versionning centralisé. Cela signifie qu'un seul répertoire général existe et tous les utilisateurs y ont accès.

Subversion permet également de charger et de modifier les sous chemins indépendamment du reste de l'arborescence.



Au sein du dépôt central, un répertoire m'a été attribué pour le développement de l'application web de mon projet.

Chaque module de Sofia est placé dans un répertoire distinct. Plusieurs sont utilisés pour le développement de mon projet.

### 4.1.8 Jenkins

Jenkins est un outil logiciel d'intégration continu. Il s'agit d'un logiciel open source, développé à l'aide du langage de programmation Java.



# Jenkins

Il permet de tester et de rapporter les changements effectués sur une large base de code en temps réel. En utilisant ce logiciel, les développeurs peuvent détecter et résoudre les problèmes dans une base de code et rapidement.

Ainsi les tests de nouveaux build peuvent être automatisés, ce qui permet d'intégrer plus facilement des changements à un projet, de façon continue.

Lorsqu'un changement est détecté via un commit par exemple, Jenkins prépare un nouveau build. Si le build rencontre une erreur, l'équipe concernée est notifiée. Dans le cas contraire, le build est déployé sur le serveur test.

### 4.1.9 Nexus

Nexus est une plateforme de gestion de dépôts, permettant d'héberger des artefacts. Ces artefacts sont des composants, générés par exemple au build d'un projet, et déposés ensuite sur Nexus grâce à l'outil Maven.



L'intérêt de Nexus est de pouvoir y partager des artefacts avec les autres développeurs du projet.

Ainsi cela facilite grandement l'utilisation des dépôts externe lié au projet. Ces éléments pouvant être récupérés lors du build d'un projet. Mais également d'avoir des versions différentes sur chacune des artefacts utilisés.

### 4.1.10 IntelliJ

« IntelliJ IDEA également appelé « IntelliJ », « IDEA » ou « IDJ » est un environnement de développement intégré (en anglais Integrated Development Environment - IDE) de technologie Java destiné au développement de logiciels informatiques. ».



IntelliJ IDEA est utilisé pour développer l'application java.

Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, la refactorisation du code et SVN intégré. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences.

#### 4.1.11 WebStorm

WebStorm est un IDE pour les langages Web (HTML, CSS et TypeScript), développé par l'entreprise JetBrains et basé sur la plateforme IntelliJ IDEA.



Il améliore la productivité et offre une expérience de développement agréable grâce un éditeur intelligent avec l'autocomplétions, la détection d'erreur à la volée, les refactorisations et le formatage de code, etc. Il offre un support avancé pour les Framework populaires comme Angular.

#### 4.1.12 Swagger

Swagger est un langage de description d'interface qui permet de décrire la structure des API sous format JSON.



Elle est utilisée pour documenter des API, mais aussi pour générer du code qui permettra d'utiliser ces API côté client et serveur.

## 4.2 Architecture

Nous allons parcourir 3 volets qui vont permettre le fonctionnement des applications Web de Sofia :

- Frontend
- Backend
- Architecture hybride

### 4.2.1 Frontend

En développement web, la notion de « front end » fait référence à l'ensemble des éléments visibles et accessibles directement sur une application web.

On utilise la SPA (Single Page Application), toute l'interface utilisateur est gérée par le browser et le serveur est accessible via des appels API (get, post, ...).

Cela signifie les données sont récupérées à l'aide d'interface REST, pour charger le code JavaScript. Il n'y a donc qu'une seule page. Le changement d'URL se fait directement depuis le code TypeScript, plus besoin d'allers-retours avec le serveur.

Grâce à cette architecture l'interface utilisateur est plus réactive, se rapprochant des applications natives.

Une bonne architecture doit être prise en compte pour garder le frontend maintenable.

Ainsi, avec le Framework Angular, elle va permettre la mise en place de pages dynamiques avec son architecture MVC orientée services à trois niveaux et l'intégration native du TypeScript.

### Angular

TypeScript nous permet d'avoir une surcouche JavaScript orientée objet et facilite grandement la communication avec une API orientée objet (Java).

Angular est très complet dans la création d'une application dynamique et orientée services, notamment grâce à l'injection de dépendances similaire aux beans que l'on peut voir dans Spring Boot.

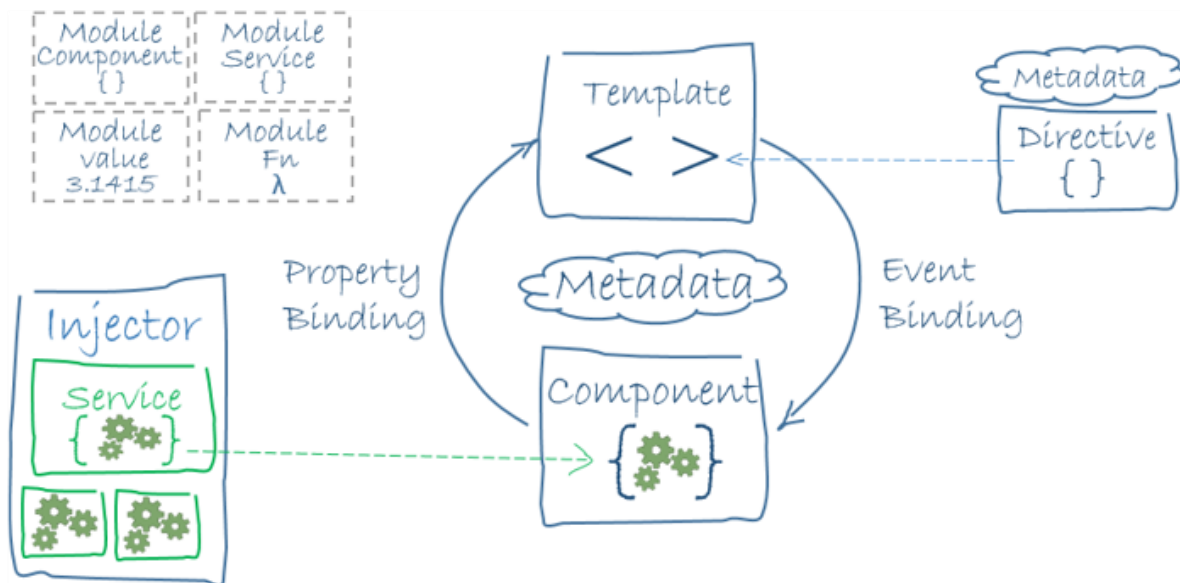


Figure 12 Schéma Angular Architecture

L'architecture d'une application Angular repose sur plusieurs modules. Le but d'un Module est de regrouper les composants et/ou services qui vont ensemble.

Une application comporte toujours au moins un module racine qui permet de démarrer et comporte généralement de nombreux autres modules fonctionnels.

Chaque application Angular a au moins un composant, le composant racine qui connecte une hiérarchie de composants.

Angular suit une architecture en 3 couches :

- La couche Vue : Une vue représente l'ensemble des éléments de l'écran. La vue consomme les données en data-binding et appelle les méthodes de la couche service.
- La couche Service : Elle fournit des fonctionnalités spécifiques qui ne sont pas directement liées aux vues. Cette couche est volontairement pauvre pour être gérée par le backend.
- La couche de persistance : Il s'agit ici de représenter toutes les données qui seront échangées entre le client et le serveur sous forme de DTO. Mais aussi toutes les méthodes qui communiquent avec le serveur via des GET, POST, ...



### 4.2.2 Architecture Backend

Avec l'architecture frontend en single page, le backend va accéder à la base de données non pas pour faire le rendu de l'interface utilisateur, mais pour renvoyer les données que le client a demandé via l'API.

#### REST

L'architecture choisie pour l'API est le REST. Le format de réponse est le JSON. C'est l'architecture la plus utilisée actuellement.

L'ensemble des données récupérées sont appelées « ressource ».

REST utilise la sémantique du World Wide Web, à savoir les URLs ainsi que les méthodes et les entêtes des requêtes HTTP.

Par exemple, récupérer la liste des médecins :

<https://monapplication.com/api/v1/care-providers>

Avec cette requête, on récupère l'ensemble des médecins en fonction des paramètres envoyés.

A noter qu'il y a également une notion de version via le chemin URL.

Cette façon permet de versionner une API REST en incluant le numéro de version dans le chemin de l'URL.

La première version sera en v1, les versions mineurs et correctives sont transparente, donc n'impacte pas l'URL. Un changement majeur qui dénote des modifications radicales de l'API implique une nouvelle version.

Par exemple, une api renvoie en JSON un objet, mais en fonction des paramètres envoyés, la requête pouvait renvoyer plusieurs objets.

Ici, le retour de l'API pose problème car il ne renvoie qu'un objet.

Ainsi j'ai dû ajouter une nouvelle version de l'API pour qu'il renvoie une liste d'objet passant de la version 1 à la version 2.

#### SPRING

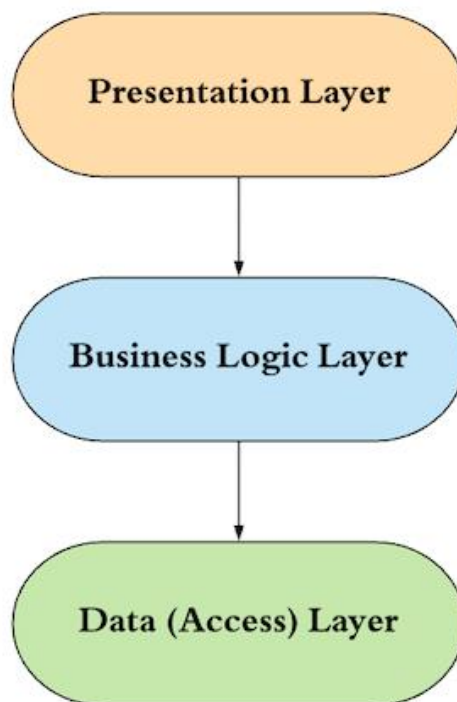
Pour gérer les requêtes http demandées par le frontend, l'architecture Spring MVC et Core sont employées.

Spring MVC suit une architecture en 3 couches dans laquelle chaque couche communique avec la couche directement en dessous ou au-dessus (structure hiérarchique).

Swagger est utilisé pour générer les interfaces serveurs et les clients pour s'y connecter. Ceci permet une meilleure organisation et un gain de temps de développement.

Les 3 couches sont :

- **Couche de présentation** : la couche de présentation gère les requêtes HTTP : elle authentifie la requête, désérialise et sérialise le JSON en objet et la transfère vers la couche métier.
- **Couche métier** : la couche métier gère toute la logique métier. Il se compose de classes de service puis communique avec les couches d'accès aux données. Il effectue également l'autorisation et la validation.
- **Couche de persistance** : la couche de persistance contient toute la logique de stockage et traduit les objets métier depuis et vers les lignes de la base de données. Mais aussi les opérations CRUD (création, lecture, mise à jour, suppression) sont effectuées en XML avec l'ORM myBatis.



*Figure 13 architecture 3 tier backend*

### 4.2.3 Architecture hybride

La séparation du frontend avec le backend permet donc à chacune des deux parties d'être mise en production indépendamment.

Le frontend n'étant plus que du HTML, CSS et TypeScript, il n'a plus besoin d'être sur le même serveur que le backend. Cela change l'infrastructure et offre plus de libertés au backend.

L'architecture exploitée par Sofia est de séparer chaque projet en service, toutes hébergées sur différents serveurs.

Par exemple mon projet eBoarding est un service qui fait appel à l'ATD backend (un autre service)

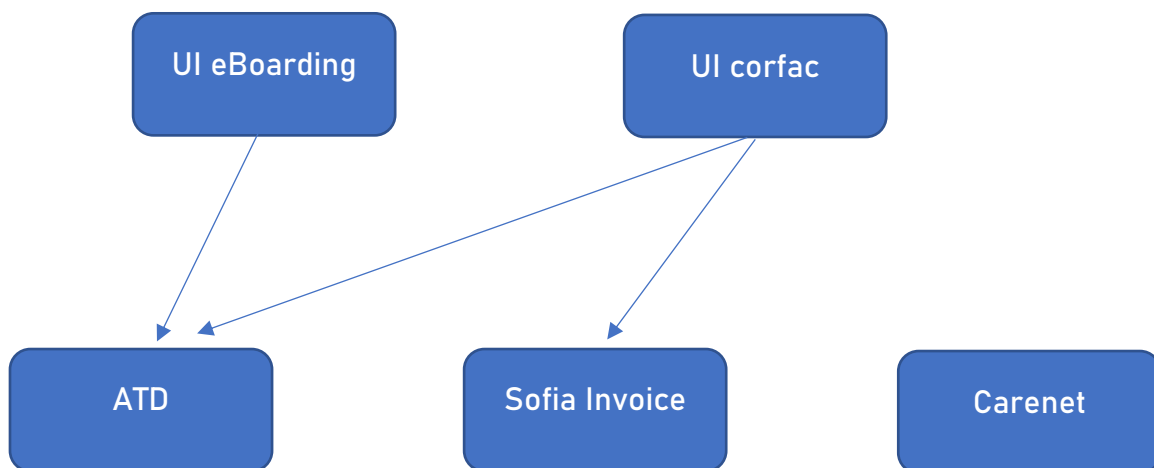


Figure 15 Schema architecture hybrid Sofia

## 5. Sprint

Dans ce chapitre, je vais parler plus en détail du déroulement des 3 sprints qui ont permis à l'aboutissement de mon projet attribué par Medsoc.

### 5.1 Sprint 1

#### 5.1.1 Introduction

Le premier sprint a d'abord nécessité une grande part d'adaptation, découverte de l'environnement de travail etc.

L'objectif de ce sprint était de présenter un premier affichage fonctionnel du projet eBoarding via une page Web.

#### 5.1.2 Sprint Backlog

Voici ci-dessous toutes les tâches qui ont été effectuées durant ce premier Sprint.

- Comprendre le projet eBoarding
- Familiariser avec le backend
- Fournir une analyse du projet et Mockup
- Encoder une prestation en état de facturation
- Création l'application web avec Angular
- Afficher le boarding d'un médecin

#### 5.1.3 Détail

##### Comprendre le projet eBoarding

Comme n'importe quel projet qui nous est imposé, il est nécessaire d'avoir bien compris ce qui nous est demandé avant de commencer son développement.

Pour cela, j'ai eu droit avec mon maître de stage à quelques heures d'explication sur le projet qui m'est attribué. De la documentation m'a été fournie concernant la demande des clients (médecin) sur le projet eBoarding pour mieux connaître les aspects techniques.

##### Familiariser avec le backend

Mon projet est un module qui s'intègre sur l'ADT de Sofia qui lui, est en production depuis plusieurs années.

Ainsi, pour fournir aux médecins la facturation des prestations pour un patient, il est nécessaire de bien connaître le flux de facturation. Mais également de se familiariser avec l'architecture orchestrée par Spring etc.

## Fournir une analyse du projet et Mockup

Les diagrammes de classe et de use case donnent une vue générale et globale du fonctionnement du eBoarding.

De plus, la création d'un mockup-écran a été réalisée pour avoir une idée de la présentation de l'application et également de comment respecter cet affichage.

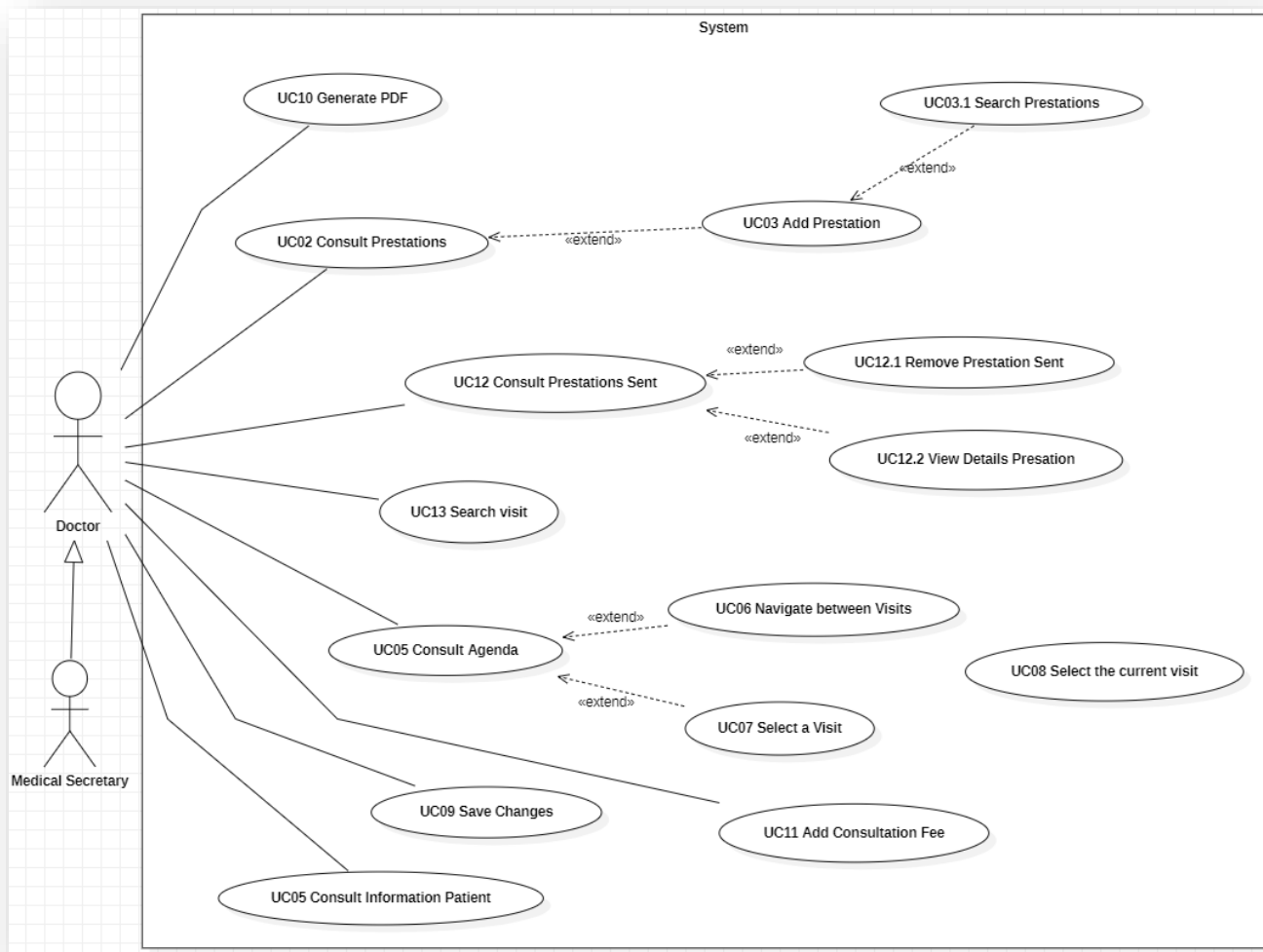


Figure 15 diagramme de use case eBoarding

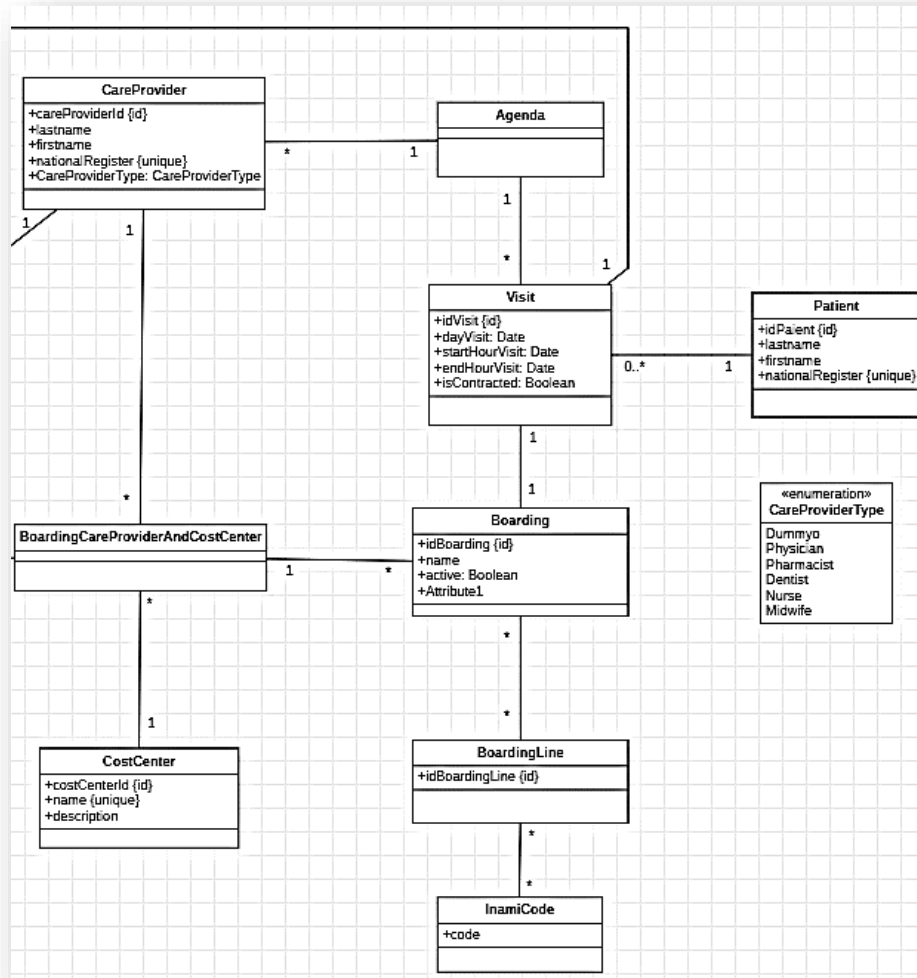


Figure 17 diagramme de classe eBoarding

Selection date

<

Yassin Aberkan

Description visite test 12

17:00

17:30

>

Cardio Dr Bettendorf

Qté 0

A1

102.012

Consult chez un spécialiste

Qté 1

Save

Prescription Enregistrée

A1 102.012

Figure 16 Mockup écran eBoarding

### Encoder une prestation en état de facturation

Après une inspection du code et une analyse du projet, il est déjà possible de récupérer le boarding d'un médecin. Cependant, pour avancer dans le projet il m'était nécessaire de pouvoir encoder une prestation en état : « à facturer ».

En ce qui concerne la création de service, il était nécessaire d'envoyer une sélection de prestation du boarding en facturation.

Il fallait également fournir des tests unitaires pour vérifier que le service appelé par le contrôleur fonctionne au préalable sans erreurs.

### Création de l'application Web Angular

Il fallait initialiser l'application web Angular avec toutes les dépendance et modules pour utiliser des templates de Sofia, accéder aux APIs du serveurs etc.

### Afficher le boarding d'un médecin

Une fois l'application initiale développé, il fallait créer les composants qui permettent d'afficher le boarding en appelant l'API avec les paramètres requis nécessaire.

## 5.2 Sprint 2

### 5.2.1 Introduction

Le but de ce sprint est de fournir une application fonctionnelle incluant la sélection de visite, de médecin et d'envoyer une prestation en facture.

Cela permet de présenter mon projet à l'hôpital de Soignies avec un médecin en fin du Sprint.

### 5.2.2 Sprint Backlog

Voici ci-dessous toutes les tâches qui ont été effectuées durant ce second Sprint.

- Encoder, supprimer une prestation d'une visite
- Ajouter les paramètres requis pour une prestation
- Implémenter l'agenda
- Mettre à jour l'url de la single page
- Afficher les médecins selon l'utilisateur connecté

### 5.2.3 Détail

#### Encoder, supprimer une prestation d'une visite

Grâce aux web services créés au sprint précédent, il fallait juste créer l'événement qui utilise le post pour l'encodage d'une prestation avec toutes les informations nécessaires.

De plus, il fallait afficher ces prestations déjà encodées dynamiquement sous forme de liste. Pour cela, un nouveau web service était nécessaire pour récupérer les 2 types de prestation (ProvidedCare et ProvidedMisc) liés au patient en fonction de la date de la visite.

Pour supprimer, il fallait également créer un web service en back end pour changer la prestation « A FACTURER » en état « DELETE ».

#### Ajouter les paramètres requis pour une prestation

Certaines prestations lors de l'encodage demandent des informations supplémentaires.

Cette information n'est pas détectable en récupérant la liste des prestations. Ainsi, il fallait fournir ces informations à travers une DTO de manière la plus générique possible pour y ajouter d'autres informations supplémentaires s'il le faut.

Une fois l'information d'un paramètre requis perçue pour une prestation, il fallait modifier le post de façon à transmettre les paramètres requis nécessaires.



## Implémenter l'agenda

L'agenda qui est sauvegardé dans la BDD nous permet de savoir tous les rendez-vous de chaque patient avec le médecin qui lui est attribué.

A noter qu'on peut encoder des prestations que si le patient s'est enregistré au guichet pour aller en visite.

Lorsque le patient s'enregistre, une visite se crée dans la BDD. Pour savoir si l'enregistrement a bien été effectué, il suffit de vérifier l'existence de la visite dans la BDD.

Ces informations sont envoyées à l'aide des ressources REST.

## Mettre à jour l'url de la single page

Etant donnée que mon application est une architecture SPA, il m'a été demandée contextualiser chaque page de l'application.

Lors du rechargement de la page, on perd toute manipulation effectuée avec l'application web.

Après un chargement de la page, il suffit d'écouter l'URL avec tous ces paramètres pour adapter la page.

## Afficher les médecins selon l'utilisateur connecter

(Voir 2.2.1 Sélection du médecin)

### 5.2.4 Conclusion

Pour finir, durant ce sprint j'ai pu avec mon maitre de stage présenter l'application eBoarding a l'hôpital de Soignies pour avoir un retour d'un médecin (client) qui utilisera l'application.

Le but de cette réunion est de faire le point sur ce qui a déjà été réalisé pour ensuite apporter un retour. Cela permet de mener à bien le développement et ne pas faire fausse route.

De plus, l'application ne traite que pour l'instant les visites polyclinique et non hospitalière, ainsi certaine notion pour traiter les visites hospitalières était encore flou. Cet échange en a également permis d'en apprendre plus sur le traitement des visites hospitalières.

Les retours était positif concernant le développement de l'application. Mais sans surprise, certaine fonctionnalité était encore manquante.

En effet, l'échange a pu conclure que le patient n'était plus obligé d'informer ça présence au guichet pour aller en visite. Des visites peuvent aussi s'effectuer à distance (par téléphone) sans forcément prendre rendez-vous.

Le retour apporté à Soignies a permis de mettre à jour et de rajouter des taches pour le dernier sprint.

## 5.3 Sprint 3

### 5.3.1 Introduction

La démo, ainsi que le workshop qui a suivi à Soignies m'a permis de finaliser les deux derniers points

Le but de ce dernier sprint, est de finaliser le projet eBoarding pour les visites polyclinique et de le déployer en tenant compte des retours de la réunion.

### 5.3.2 Sprint Backlog

Voici ci-dessous toutes les taches qui ont été planifié durant ce dernier Sprint.

- Créer une visite qui est déjà planifié dans l'Agenda
- Importer / modifier la recherche patient
- Créer une visite qui n'est pas planifié dans l'Agenda
- Tester l'application

### 5.3.3 Détail

#### Créer une visite qui est déjà planifié dans l'Agenda

Pour cette tâche la visite n'existe pas encore, elle doit être créée par un bouton que l'utilisateur cliquera.

Comme le rendez-vous est déjà planifié, nous avons toutes les informations à disposition pour créer la visite. Cependant le boarding qui sera fourni à sa création ne peut être forcément déduit. Un boarding est lié à un médecin mais il se peut que plusieurs soient liés à un même médecin.

Pour résoudre ce problème, il suffit de demander à l'utilisateur quelle boarding va être employée pour la visite.

#### Importer / modifier la recherche patient

Pour crée une visite sans aucune information au préalable, il faut rechercher un patient.

Pour se faire, la recherche patiente existe déjà dans le moteur de correction de facture. Pour éviter les doublons, il fallait rendre ce composant « exploitable » pour les 2 projets.

Des modifications devaient être apporter pour rendre le composant compatible pour les 2 projets et également fournir le retour des patients sous forme de liste.

### Créer une visite qui n'est pas planifié dans l'Agenda

Comme aucune information n'est reçue au préalable pour créer une visite, il est nécessaire de connaître le patient, la date, l'heure du rendez vous et le boarding employé pour la visite.

Pour obtenir le patient, il suffisait d'importer le composant « Recherche Patient ». Pour la date et le boarding, il y'a un formulaire à remplir.

Le problème étant que une fois la visite crée, on n'a pas accès à la visite car l'API qui se charge de retrouver les rendez-vous ne la prend pas en compte.

En effet, les rendez-vous affichez se fait à partir de l'agenda.

Pour résoudre ce problème, il fallait créer une nouvelle DTO qui récupère toutes les visites et tous les rendez-vous de l'agenda du médecin en fonction du jour.

### Tester l'application

Avant de pouvoir déployer l'application, il est nécessaire de la tester. Une réunion a été dédiée avec un développeur pour tester tous les cheminements possibles sur l'application. Les tests ont permis de parcourir plusieurs partie de l'application : la création de la visite, l'affichage du boarding, la connexion, etc.

#### 5.3.4 Conclusion

Pour finir, ce dernier sprint a permis de finaliser mon projet en prenant compte des retours apporté lors de ma visite à l'hôpital de Soignies.

Une fois mes tâches effectuées, j'ai pu tester l'application à l'aide d'un développeur pour valider mon projet.

La finalisation de l'application eBoarding et la validation de celui-ci à l'aide de tests permettra de la déployer.

## 6. Conclusion

Au bout de 15 semaines, l'objectif qui était de pouvoir réaliser une nouvelle application web eBoarding comme un développeur engagé à temps plein a donc été réalisé.

J'ai su faire des corrections, des améliorations et de rajouter de nouvelles fonctionnalités sur la partie backend. Cela m'a permis aussi de développer une toute nouvelle application frontend, ainsi que de la maintenir.

Mais également le workshop effectué à Soignies avec une démo a permis de mettre à jour et de rajouter des fonctionnalités au projet.

La majeure partie de mon stage s'est déroulé en présentiel, cela à apporter grandement en interaction et en bien-être entre l'équipe de développement et moi.

Ce stage fut une expérience tant enrichissante que positive au point de vue professionnel et humain. J'ai pu mettre en pratique et approfondir les acquis théoriques étudiés pendant mon parcours scolaire à l'École Supérieure en Informatique.

J'ai également pu acquérir de nombreuses connaissances durant ce stage. À savoir, une connaissance bien plus approfondie de java avec son Framework Spring, la découverte d'un nouveaux Framework Angular et de son langage TypeScript, etc.

J'ai donc été confronté à la réalité du monde professionnel et du travail en équipe. Ces apprentissages constituent, pour moi, une excellente expérience professionnelle.

## 7. Annexes

### 7.1 Bibliographie

- Backend. (2021). Wikipedia. <https://fr.wikipedia.org/wiki/Backend>
- Comment organiser le code d'application avec une architecture à 3 niveaux? (2019). WellDoneBy. <https://welldoneby.com/blog/how-organize-application-code-with-3-tier-architecture/>
- Comprendre ce que sont les microservices. (2020). Red Hat. <https://www.redhat.com/fr/topics/microservices>
- Développement front-end et back-end : Quelles différences ? (2021). LesJeudis. <https://blog.lesjeudis.com/developpement-front-end-et-back-end-queelles-differences>
- Introduction to Angular concepts. (2021). Angular. <https://angular.io/guide/architecture#:~:text=The%20architecture%20of%20an%20Angular,by%20a%20set%20of%20NgModules>.
- Jenkins : définition, fonctionnement, avantages du logiciel open source d'intégration continue. (2019). LeBigData. <https://www.lebigdata.fr/jenkins-definition-avantages#:~:text=Jenkins%20est%20un%20outil%20logiciel%20d'int%C3%A9gration%20continu.,de%20code%20en%20temps%20r%C3%A9el>.
- Maven. (2021). Maven. <https://maven.apache.org/>
- Medsoc. (2018). Medsoc. <https://medsoc.be/>
- Notions d'architecture 3-tier. (2018). MyScenari. <https://stph.scenari-community.org/bdd/lap2/co/webUC003archi.html>
- npm. (2021). Wikipedia. <https://fr.wikipedia.org/wiki/Npm>
- Qu'est-ce que la méthodologie Scrum ? (2017). PlanZone. <https://www.planzone.fr/blog/quest-ce-que-la-methodologie-scrum>
- Qu'est-ce que Typescript et pourquoi l'utiliser ? (2021). modern-javascript. <https://modern-javascript.fr/quest-ce-que-typescript-et-pourquoi-lutiliser/#:~:text=Typescript%20est%20une%20surcouche%20%C3%A0,de%20s%C3%A9r%C3%A9nit%C3%A9%20%C3%A0%20vos%20d%C3%A9veloppements>.
- Qu'est-ce qu'une application à page unique et pourquoi les gens l'aiment tellement? (2021). bloomreach. <https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html>
- Spring. (2014). jmdoudoux. <https://www.jmdoudoux.fr/java/dej/chap-spring.htm>

## 7.2 Carnet de Bord

### Semaine 1

#### Objectifs

1. Comprendre le projet qui m'est attribué
2. Me familiarisé avec leur code, leur environnement
3. Faire une Analyse du projet
4. Commencer les premiers tests unitaires

#### Compte rendu explicatif

1. J'ai eu droit avec mon maitre de stage a quelques heures de d'explication sur le projet qui m'est attribué et sur tout ce qu'ils utilisent comme langage, site de gestion de projet etc. Des documents m'ont également été fournis concernant la doc de leur code mais aussi concernant la demande des clients sur le projet qui m'est attribué.

2. Une analyse du code ma été demandé en me fournissant tous les accès nécessaires : codelabs, Oracle, ...

Pour mieux comprendre leur architecture : java spring, les web services, comment déployer une api sur les serveurs etc).

3. Une fois que tout est clair niveaux objectif demandé et comment je peux exploiter leur code pour leur projet, j'ai pu commencer à faire une analyse du projet.

Comme on a vu en ana3, j'ai fait un diagramme de classe, mais également de use case et des diagrammes d'activités.

Une fois que mon analyse a été terminé, mon maitre de stage ma fait un feedback pour voir ce qui pouvait être améliorer etc.

4. Ainsi, j'ai pu également commencer à coder, en faisant des tests unitaires sur mes premiers web services crée coté backend qui seront nécessaire pour le projet.

#### Notes et remarques personnelles

Très bonne accueil, a l'écoute quand j'ai des questions a poser etc.

## Weeks 2

### Objectives

1. Create a new web services and test it
2. Complete the analysis and create a mockup
3. Create a new api rest and test it
4. Begins to learn angular

### Explanatory Report

1. I had to create a new web services on the back end. Once done I had to show what I did to prove that I understood the project. when everything was good I was able to commit to the master.
2. I was also able to have a second explanation because some points were unclear for the creation of my project. Once I have asked all my questions and I have understood well I was able to complete my analysis and show a first mockup.
3. I was able to create a new api, new web service and make changes in the database.
4. As the front-end application is in angular with typescript, I was able to begin to understand its architecture and so on.

### Notes and personal comments

Nothing to say



## Semaine 3

### Objectifs

1. Apprentissage de l'architecture Angular
2. Création de mock up écran
3. Réfléchir à comment bien présenter l'application pour les users
4. Réalisation des premières tâches côté front end

### Détails

1. Comme l'application frontend est en typescript avec angular, j'ai du apprendre a me familiariser avec leur projet à eux en frontend et en apprendre plus sur les spécificités de cette architecture front end.
2. J'ai créé un mockup écran qui permettra de montrer toutes les applications affichées à l'écran pour avoir une idee plus clair
3. Certaines fonctionnalités restent encore floues, en effet pour certaines fonctionnalités il est intéressant de se poser les questions pour savoir comment l'utilisateur peut l'utiliser de manière simple et efficace.
4. J'ai commencer a coder et à réaliser ma première tâche en front end "Afficher les boarding lines"

### Remarques

Rien a signaler

## Semaine 4

### Objectifs demandés

1. Terminer l'affichage d'une fonctionnalité de l'application frontend
2. Présentation rétro sprint de mon projet
3. Réalisation de nouveaux web services pour apporter des modifications à la BDD

### Détails

1. J'ai du terminer l'affichage de ma tâche "afficher les Boarding Lines" côté frontend, j'ai dû également la mettre la plus Responsive possible pour qu'elle soit utilisable pour les tablettes etc.
2. Comme un nouveau mois commence, j'ai dû faire une présentation devant tous les développeurs pour montrer ce que j'ai fait durant le premier mois et ce que je compte faire le mois prochain. J'ai du également créer des nouvelles user stories pour ce mois et en discuter avec l'équipe
3. J'ai dû également rajouter des nouveaux web services et api rest, les tester avec des tests unitaires etc. pour permettre de faire des postes etc.

### Remarques

Rien a signaler

## Semaine 5

### Objectifs demandés

1. Terminer l'implémentation back end pour post ou delete une prestation
2. Afficher les prestations déjà enregistrées
3. Permettre de ajouter ou supprimer une prestations (frontend)
4. Adapter la page web pour qu'elle soit responsive.

### Détails

1. J'ai dû terminer l'implémentation du post et du delete d'une prestation. Il fallait rajouter des méthodes côté services et façades pour permettre des les liés au gets correspondant. Je devais aussi faire des tests unitaires pour bien vérifier si tout fonctionnait avant de commit et utiliser ça en frontend.
- 2.
3. Il fallait afficher les prestations déjà enregistrées sur la page web pour permettre de supprimer si le médecin le souhaite par après.
4. Lorsqu'on clique sur une prestation, j'ai fait en sorte qu'elle soit push directement puis l'afficher dans les 'prestations déjà enregistrées'.  
Pour cela j'ai dû appeler le post correspondant mais aussi update la quantité de la prestation cliqué car certaines ont une quantité maximum à ne pas dépasser. Une fois la quantité franchie j'ai fait en sorte que l'élément ne soit plus cliquable.  
Une fois cette étape faite, dans les prestations déjà sélectionnées, j'ai dû rajouter un bouton delete qui permet de mettre un flag delete dans la BDD pour qu'elle ne soit plus affiché.
5. Comme la page web pourra être utilisé via une tablette ou même téléphone dans les hôpitaux, j'ai du revoir certains éléments à l'affichage qui devront s'adapter en fonction de la résolution

### Remarques

Rien a signaler

## Semaine 6

### Objectifs demandés

1. Réflexion comment implanter des paramètres requis
2. Implémenter les paramètres requis backend
3. Implanter et afficher paramètre requis frontend

### Détails

1. Après avoir terminé la partie affichage des prestations, lorsqu'on veut ajouter une prestation pour certains, ils demandent des paramètres supplémentaires. J'ai dû réfléchir à comment implanter ça sur mon projet et aussi que ce soit générique pour permettre plus tard d'ajouter d'autres paramètres.
2. J'ai dû rajouter des classes, réécrire d'autres, créer un get et bien sûr tester tout ça avec des tests unitaires en créant une base de données vides. Lors du merge, il y avait un conflit avec d'autre partie du projet qui provoque des failed sur certains tests unitaires. j'ai dû également fixer tous ces problèmes.
3. Une fois l'implémentation backend finie, j'ai du rajouter lorsqu'on click sur une prestation, si un paramètre est demandé, créent une pop up qui va lister tous les paramètres supplémentaires demandés avant l'envoi dans la BDD.

### Remarques

Rien a signaler

## Semaine 7

### Objectifs demandés

1. Implémenter l'agenda backend
2. Implémenter l'agenda frontend
3. Présentation du projet au directeur
4. Lié l'agenda à mon projet

### Détails

1. Dès lors avoir terminé la partie consultation de mon projet, j'ai dû commencer à implémenter la partie agenda qui permet de consulter une visite d'un médecin. J'ai dû enrichir le modèle déjà proposé pour l'agenda, fournir des gets pour le frontend et les tester.
2. Une fois cette étape réalisée, j'ai dû réfléchir à comment bien incrémenter l'agenda sur ma web page. Puis créer les composants qui permettent de récupérer les informations via le get et l'afficher sur l'écran.
3. Mon maître de stage m'a convoqué pour présenter ce qui a déjà été fait au directeur de la boîte. Comme le projet a bien avancé, il a pu donner son avis positif et organiser un rendez-vous avec un ou plusieurs médecins dans les prochaines semaines pour donner leur avis etc.
4. Après avoir affiché l'agenda, j'ai dû créer un lien entre une visite de l'agenda et une prestation qui a déjà été faite auparavant. Dès lors qu'on clique, va charger la prestation demandée.

### Remarques

Avec les nouvelles mesure COVID, je fais 4 jours distanciels et 1 jour présentiel.

## Semaine 8

### Objectifs demandés

1. Apporter des url dynamiques sur la single page
2. Apprendre l'utilisation des websockets
3. Implémenter un websocket dans le projet
4. Préparation du retro spring mensuel

### Détails

1. Comme l'application web est une page unique, l'url ne change pas au fil des actions effectuées. J'ai dû trouver un moyen pour que l'url change au fil des actions pour par exemple quand on copie l'url on tombe directement sur la visite indiquée sur l'url avec la consultation du patient.
2. Il m'a été demandé d'intégrer un websocket qui permettra d'indiquer comme quoi un patient est enregistré chez la secrétaire médicale pour la consultation auprès du médecin. Je me suis donc informé sur comment utiliser les websockets, voir comment il a été implémenté dans d'autres projets, etc.
3. J'ai commencé à intégrer le websocket sur mon projet mais il est toujours impossible de savoir comment envoyer un message quand le patient s'est enregistré car il est utilisé sur une application externe.
4. Comme chaque mois, l'entreprise demande de présenter ce qui a été fait durant ce mois, j'ai donc mis à jour mes tâches et préparé une présentation.

### Remarques

Rien à signaler

## Semaine 9

### Objectifs demandés

1. Modifier certains éléments de la partie visite
3. Apporter des nouvelles tâches au projet
2. Établir un lien entre le user et le médecin

### Détails

1. Après avoir présenté l'évolution de mon projet lors du sprint planning mensuel, Il m'a été demandé de modifier certains éléments de l'interface.  
En effet l'affichage des visites manquait quelque information pour que le médecin puisse avoir toutes les informations nécessaires sur la page.
2. De plus dans le sprint planning mensuel, des nouvelles tâches ont été apportées pour avoir un maximum de fonctionnalité avant une démo qui sera testé par des médecins. Ainsi j'ai dû voir ce qui était possible à réaliser, organiser une réunion pour avoir plus d'informations.
3. Jusqu'à maintenant, quand le médecin se connecte je ne récupère pas les informations le concernant pour afficher par la suite son agenda.  
Ainsi, j'ai dû analyser la DB pour savoir quelles sont les tables utilisées par rapport au user. Par la suite, côté back end, j'ai dû faire un lien entre l'utilisateur connecté et le médecin pour fournir une API qui avec le login de l'utilisateur récupère les informations du médecin.

### Remarques

Rien à signaler

## Semaine 10

### Objectifs demandés

1. Création service user-médecin
3. Adapter frontend pour user-médecin

### Détails

1. Une fois que la modification de la BDD a été faite, j'ai dû mapper les tables liées aux users pour être utilisable en back-end.  
Par la suite, une fois la manipulation de la table fonctionnelle, j'ai dû la tester et commencer à créer le service qui permettra d'être appelé par ma nouvelle API. Elle va récupérer le login de l'utilisateur pour envoyer tous les médecins qui sont administrés / liés par cet utilisateur.
2. J'ai dû appeler ce service dès l'initialisation de mon application et permettre d'afficher en liste déroulante tous les médecins qui sont liés par l'utilisateur.  
Et adapter mon projet de base pour que chaque médecin sélectionné affiche un nouvel agenda lié au médecin avec ses propres rendez-vous.

### Remarques

Rien à signaler

## Semaine 11

### Objectifs demandés

1. Terminer Affichage des médecins
2. Reunion Soignies
3. Debrief Retour fonctionnalité Soignie

### Détails

1. J'ai dû régler quelques problèmes que quand on sélectionne un médecin, l'agenda se met à jour avec le médecin sélectionné. Pour cela, il faut détecter le changement de médecin et réinitialiser les visites mais aussi le boarding.
2. Je suis parti à l'hôpital de Soignies pour présenter mon projet avec un médecin en fin du Sprint. Le but de cette réunion est de faire le point sur ce qui a déjà été fait pour ensuite apporter un retour. Cela permet de mener à bien le développement et ne pas faire fausse route.  
De plus, l'application ne traite que pour l'instant les visites polyclinique et non hospitalière, ainsi certaine notion pour traiter les visites hospitalières était encore flou. Cet échange en a également permis d'en apprendre plus sur le traitement des visites hospitalières.
3. J'ai eu une réunion avec mon maître de stage pour conclure les retours apportés à Soignies qui ont conclu de mettre à jour et de rajouter des tâches pour le dernier sprint.

### Remarques

Rien a signaler

## Semaine 12

### Objectifs demandés

1. Création frontend visite avec agenda
2. Analyse creation visite

### Détails

1. Dans un premier temps, j'ai mis en place un bouton qui permet de créer une visite en frontend avec toutes les informations qui sont à disposition avec l'agenda. Un formulaire survient via un popup si plusieurs boardings sont liés à un médecin. Un service est déjà mis à disposition pour créer une visite. J'ai dû également mettre en place après le création de la visite, un refresh de la visite pour qu'il s'ajoute directement après sa création.
2. Après avoir montré la création de la visite a mon maitre de stage, il m'a conseillé de repenser à la création d'une visite pour la rendre la plus globale possible pour qu'elle soit également possible d'utilisation pour la création de visite vierge. Pour cela il faut penser à importer des composants d'autre application etc.

### Remarques

Rien a signaler

## Semaine 13

### Objectifs demandés

1. Adapter composant recherche Patient
2. Proposer recherche patient sous forme de liste
3. Importer recherche patient dans un projet de ressources partages

### Détails

1. Pour créer une visite vierge, il faut connaître le patient, pour cela il faut proposer au médecin une recherche de patient en fonction de son nom et prénom. Ce composant a déjà été développé mais il est uniquement exploité dans une autre application web frontend. Pour la réutiliser, tout d'abord il faut que le composant recherche patient soit totalement indépendant des autres.
2. Mon maître de stage m'a également demandé de proposer le résultat des patients sous forme de liste car ils sont proposés pour l'instant sous forme de carte.
3. Une fois le composant totalement indépendant, j'ai dû extraire le composant pour le placer un projet theme lib ou les composants peuvent être appelés dans n'importe quelle projet.

### Remarques

Rien a signaler

## Semaine 14

### Objectifs demandés

1. Création de visite vierges frontend
2. Création de visite avec agenda frontend
3. Adapter DTO

### Détails

1. Comme le rendez-vous est déjà planifié, nous avons toutes les informations à disposition pour créer la visite. Cependant le boarding qui sera fourni à sa création ne peut être forcément déduit. Un boarding est lié à un médecin mais il se peut que plusieurs soient liés à un même médecin.
2. Comme aucune information n'est reçue au préalable pour créer une visite, il est nécessaire de connaître le patient, la date, l'heure du rendez vous et le boarding employé pour la visite. Pour obtenir le patient, il suffisait d'importer le composant « Recherche Patient ». Pour la date et le boarding, il y'a un formulaire sur une autre page à remplir.
3. Le problème étant que une fois la visite crée, on n'a pas accès à la visite car l'API qui se charge de retrouver les rendez-vous ne la prend pas en compte. En effet, les rendez-vous affichez se fait à partir de l'agenda. Pour résoudre ce problème, il fallait créer une nouvelle DTO qui récupère toutes les visites et tous les rendez-vous de l'agenda du médecin en fonction du jour.

### Remarques

Rien a signaler

## Semaine 15

### Objectifs demandés

1. Finaliser Projet
2. Tester Application
3. Terminer rapport

### Détails

- Etant donné que c'est la dernière semaine, il est temps de finaliser mon projet pour permettre une version stable et prête d'utilisation. J'ai réglé le problème des dates qui après la création d'une visite, la date est affichée sans décalage horaire.
- Avant de pouvoir déployer l'application, il est nécessaire de la tester. Une réunion a été dédiée avec un développeur pour tester tous les cheminements possibles sur l'application. Les tests ont parcouru de la création de la visite, l'affichage du boarding, la connexion, etc.
- La remise du rapport est pour ce vendredi 28 mai, je finalise mon rapport en corrigeant les fautes, prenant en compte les retours de mon maître de stage etc.

### Remarques

Rien a signaler

## 7.3 Evaluation continue

**HE2B**  
ESI

### STAGE EN INFORMATIQUE : EVALUATION CONTINUE DU STAGE PAR LE RESPONSABLE AU SEIN DE L'ORGANISME.

#### Document du stagiaire

Nom de l'Organisme Nejma  
 Personne qui évalue le stage  
 Nom, Prénom Polig Joe Tél/GSM 0472 / 783055  
 Concerne l'étudiant: Nom Aberkan Prénom Yassin  
 Echelle de valeurs proposées  
 Insuffisant --- Faible -- Très Moyen - Moyen M Bon + Très bon ++ Excellent +++

Au terme du stage l'étudiant a atteint les seuils de compétences suivants :

	mi Mars	mi Avril	mi Mai
<b>FORMATION</b>			
Connaissances théoriques	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
Analyse	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
Pratique de la programmation	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
<b>TRAVAIL FOURNI</b>			
Quantité de travail fourni	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
Qualité du travail fourni	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
Adaptabilité au système et à l'environnement	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
Capacité d'auto-apprentissage	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
Capacité d'organisation	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
<b>COMPORTEMENT</b>			
Aptitudes à la communication	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
Sociabilité	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>
Réceptivité aux recommandations	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>	--- -- - M + ++ <u>+++</u>

#### Comment remplir la grille

Après une période de +/- 4 semaines, un entretien est réalisé entre le superviseur du stage et le/la stagiaire. Chacun des items de cette grille sont évalués et explicités par le superviseur. Le/la stagiaire est tenu de prendre note des remarques émises en les transcrivant dans son "Carnet de bord de Stage".

*Polig Joe*

Ce document ainsi que le "Carnet de bord de Stage" DOIVENT être présenté spontanément au professeur de référence, qui pourra en tenir compte pour sa part d'évaluation du stage. Le "Carnet de bord de Stage" fait partie du rapport final