



Ecole Polytechnique de l'Université de Tours  
Département Informatique  
64 avenue Jean Portalis  
37200 Tours, France  
Tél. +33 (0)2 47 36 14 14  
[polytech.univ-tours.fr](http://polytech.univ-tours.fr)

## Projet Recherche & Développement 2021-2022

# Visualisation et gestion du parcours patient



**POLYTECH<sup>®</sup>**  
**TOURS**

**Entreprise**

**Polytech**



**Tuteur entreprise**

**Simon MOULARD**

**Étudiant**

**François BRISON (DI5)**

**Tuteur académique**

**Yannick KERGOSIEN**

# Liste des intervenants

## Entreprise

Polytech  
64 avenue Jean Portalis  
37200 Tours, France  
[polytech.univ-tours.fr](mailto:polytech.univ-tours.fr)



Nom	Email	Qualité
François BRISON	<a href="mailto:francois.brison@univ-tours.fr">francois.brison@univ-tours.fr</a>	Étudiant DI5
Yannick KERGOSIEN	<a href="mailto:yannick.kergosien@univ-tours.fr">yannick.kergosien@univ-tours.fr</a>	Tuteur académique, Département Informatique
Simon MOULARD	<a href="mailto:simon.moulard@univ-tours.fr">simon.moulard@univ-tours.fr</a>	Tuteur entreprise



# Avertissement

Ce document a été rédigé par François BRISON surnommé l'auteur.

L'entreprise Polytech est représentée par Simon MOULARD surnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Yannick KERGOSIEN surnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



## Pour citer ce document

François BRISON, *Visualisation et gestion du parcours patient:* , Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
  author={BRISON, François},
  title={Visualisation et gestion du parcours patient: },
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

# Table des matières

Liste des intervenants	<b>a</b>
Avertissement	<b>b</b>
Pour citer ce document	<b>c</b>
Table des matières	<b>i</b>
Table des figures	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1 Acteurs, enjeux et contexte .....	1
2 Objectifs .....	1
3 Hypothèses .....	1
4 Bases méthodologiques.....	2
<b>2 Description générale</b>	<b>3</b>
1 Environnement du projet .....	3
2 Caractéristiques des utilisateurs.....	3
3 Fonctionnalités du système .....	3
3.1 Diagramme de Gantt (Vue) .....	3
3.2 Filtrage (Vue).....	4
3.3 Base de données (Modèle).....	4
3.4 Import/Export d'instances/solutions (Modèle).....	4
3.5 Contraintes (Contrôleur).....	4
3.6 Ordonnancement (Contrôleur) .....	4
4 Structure générale du système.....	4

<b>3</b>	<b>État de l'art / Veille technologique</b>	<b>6</b>
1	Logiciels de gestion de parcours patients .....	6
2	Librairies de diagramme de Gantt.....	6
2.1	Critères .....	6
2.2	C++ .....	7
2.3	Java .....	7
2.4	Python .....	7
2.5	Javascript .....	7
3	Librairies pour réaliser une Interface Utilisateur Graphique.....	8
4	Modélisation proposée.....	8
<b>4</b>	<b>Mise en oeuvre</b>	<b>9</b>
1	Outils et librairies utilisés .....	9
2	Éléments d'implémentation, choix techniques .....	9
3	Analyse des résultats, évaluation, qualité .....	9
<b>5</b>	<b>Bilan et conclusion</b>	<b>11</b>
1	Bilan du semestre 9 .....	11
1.1	Tâches faites.....	11
	Présentation des sujets et choix du PRD .....	11
	Cahier des charges / spécifications.....	11
	Modélisation .....	11
	Recherche de technologies.....	11
	Tests de technologies .....	11
	Apprentissage de librairies.....	11
1.2	Tâches en cours .....	12
	RDV Client .....	12
	Rédaction du rapport .....	12
	Interface - Gantt .....	12
	Interface - BDD .....	12
	Modèles - Gantt .....	12
	Modèles - BDD .....	12
	Modèles - Import/Export .....	12
	Contrôleur - Lien IHM/Modèle.....	12
1.3	Tâches à effectuer .....	12
	Interface - Modification .....	12
	Contrôleur - Lien BDD.....	12
	Contrôleur - Contraintes.....	12
	Contrôleur - Ordonnancement .....	12
	Validation .....	13

2	Bilan du semestre 10.....	13
3	Bilan sur la qualité .....	13
4	Bilan auto-critique.....	13
<b>Annexes</b>		<b>14</b>
<b>A Planification, gestion de projet</b>		<b>15</b>
1	Evolution du projet .....	15
2	Description des tâches.....	15
	Présentation des sujets et choix du PRD .....	15
	Cahier des charges / spécifications.....	16
	Modélisation .....	16
	Recherche de technologies.....	16
	Tests de technologies .....	16
	Apprentissage de bibliothèques.....	16
	RDV Client .....	16
	Rédaction du rapport .....	17
	Interface - Gantt .....	17
	Interface - BDD .....	17
	Interface - Modification .....	17
	Modèles - Gantt .....	17
	Modèles - BDD .....	17
	Modèles - Import/Export .....	17
	Contrôleur - Lien IHM/Modèle.....	18
	Contrôleur - Lien BDD.....	18
	Contrôleur - Contraintes.....	18
	Contrôleur - Ordonnancement .....	18
	Validation .....	18
<b>B Description des interfaces</b>		<b>19</b>
1	Interfaces matérielles/logicielles .....	19
2	Interfaces homme/machine.....	19
<b>C Cahier de Spécifications</b>		<b>23</b>
1	Spécifications Fonctionnelles .....	23
1.1	Fonctionnalités à développer .....	23
	Diagramme de Gantt (Vue) .....	23
	Base de données (Modèle) .....	24
	Import/Export d'instances/solutions (Modèle) .....	24
	Contraintes (Contrôleur) .....	24

	Ordonnancement (Contrôleur) .....	25
2	Spécifications non fonctionnelles .....	25
2.1	Contraintes de développement et conception .....	25
2.2	Contraintes de fonctionnement et d'exploitation.....	25
2.2.1	Performances .....	25
2.2.2	Capacités .....	25
2.2.3	Contrôlabilité .....	26
2.2.4	Sécurité .....	26
<b>D</b>	<b>Cahier du développeur</b> .....	<b>27</b>
1	Introduction .....	27
2	Diagrammes architecturaux et UML .....	27
3	Descriptions détaillées de données exploitées .....	29
	Fichier instance .....	29
	Fichier solution .....	29
	Fichier dictionnaire .....	29
<b>E</b>	<b>Document d'installation</b> .....	<b>30</b>





# Table des figures

<b>2 Description générale</b>	
2.1 Le modèle de l'application .....	5
<b>A Planification, gestion de projet</b>	
A.1 Le diagramme de Gantt Initial.....	15
A.2 Le diagramme de Gantt Final .....	18
<b>B Description des interfaces</b>	
B.1 Maquette de l'onglet 1 de l'application, vue Patient .....	19
B.2 Maquette de l'onglet 1 de l'application, vue Ressource .....	20
B.3 Maquette de l'onglet 2 de l'application.....	20
B.4 Maquette de la fenêtre type de modification des données.....	21
B.5 Maquette de la fenêtre d'import/export .....	21
<b>D Cahier du développeur</b>	
D.1 Diagramme des classes du modèle et de leurs interactions .....	28
D.2 Diagramme du modèle de la base de données du projet .....	28

# 1

## Introduction

### 1 Acteurs, enjeux et contexte

Ce rapport et le cahier de spécifications associé sont les documents servant de fondation au Projet Recherche & Développement 2021-2022 qui sera réalisé par François Brison. Ce projet aura deux encadrants, Simon Moulard et Yannick Kergosien. Sa raison d'existence est de venir compléter la thèse du doctorant Simon Moulard, qui prendra donc également le rôle de client.

Cette thèse a pour sujet l'ordonnancement de rendez-vous de patients dans un hôpital de jour en fonction des ressources disponibles (salles, machines, personnel et leurs compétences). Ainsi, le PRD Visualisation et Gestion du parcours patients a pour but de visualiser les solutions générées par cet ordonnanceur, voire intégrer ce dernier directement.

### 2 Objectifs

L'objectif principal du projet est de permettre via une interface graphique type diagramme de Gantt de visualiser, déplacer, et modifier des rendez-vous assignés à un ou plusieurs patients sur une journée. Cette interface, de manière générale, devra rester aussi simple d'usage que possible.

De plus, l'application utilisera une base de données afin de stocker les informations nécessaires. Il faut noter que l'application devra être capable de charger différentes bases. En effet, d'un point de vue ordonnancement, les données (patients, rendez-vous, ...) représentent une instance du problème, et la planification (heures des rendez-vous d'une journée) représente la solution à cette instance du problème. Ainsi, il est nécessaire de pouvoir changer d'instance et les sauvegarder.

Les contraintes feront donc également partie des objectifs de ce projet. L'ordonnancement décrit des problèmes basés notamment sur des contraintes, et celles-ci devront être vérifiées lors de placement ou déplacement de rendez-vous. On peut ainsi prendre pour exemple le fait qu'un docteur ne puisse être à deux RDV en même temps : par conséquent, il ne peut y avoir plus de ressources utilisées à chaque instant qu'il n'y en a de disponibles.

### 3 Hypothèses

Premièrement, ce projet dans le cadre d'un hôpital de jour. Ainsi, les patients verront tous les rendez-vous de leur parcours patient être placés dans la journée, et ne resteront pas pour la nuit.

Le diagramme de Gantt qui constitue la partie la plus importante de l'IHM commencera donc à une heure de début d'une journée, et finira plus tard à une heure de fin dans la même journée.

Ensuite, hypothèse sera faite que tous les rendez-vous auront une durée multiple de 5 minutes. Cette hypothèse a pour origine le fait que, dans tous les parcours patients décrits dans un document fourni par le client, les rendez-vous suivent ce principe de multiple de 5.

## 4 Bases méthodologiques

Premièrement, les différentes fonctionnalités ont été découpées afin de mieux les cerner et les exprimer. Cela a permis, dans un premier temps, la réalisation d'un diagramme de Gantt afin d'essayer de prévoir les avancements nécessaires du projet sur toute sa durée.

Ensuite, des diagrammes représentant les modèles de données et leurs équivalents en Base de Données ont été réalisés, dans une première étape de modélisation de l'application dans son intégralité.

Ainsi, un autre diagramme, représentant les différentes classes et leurs rôles a pu être fait.

Pour tout ce qui touche au développement, les règles du Génie Logiciel seront respectées. Une convention de nommage sera appliquée, les classes et méthodes seront commentées au minimum par un en-tête, un gestionnaire de version sera utilisé et le logiciel sera découpé, selon le modèle précédent, en classes bien distinctes.

De manière plus générale, des réunions avec le client régulières sont prévues, de manière à faire part de la manière dont l'application sera développée, puis pour rendre compte de l'avancement une fois le développement démarré.

# 2

## Description générale

### 1 Environnement du projet

Ce projet s'effectue en parallèle de la thèse de Simon MOULARD, thèse ayant pour sujet l'ordonnancement de rendez-vous de parcours patients dans un hôpital de jour. Ainsi, ce projet a pour but de permettre la visualisation et la modification des résultats donnés par son ordonnanceur, soit via lecture de fichiers instance et solution, soit via intégration directe de l'ordonnanceur dans le code du PRD.

C'est pourquoi l'un des attendus du projet est l'ajout d'un ordonnanceur basique : ainsi, il sera plus facile de simplement remplacer cette brique logicielle par une autre le cas échéant.

### 2 Caractéristiques des utilisateurs

Il n'y a qu'un seul type d'utilisateur. Ainsi, il n'y a pas d'administrateurs, ou quoi que ce soit de la sorte. Chaque utilisateur a donc un accès immédiat, sans identification préalable, à toute l'application.

L'utilisateur ne doit pas posséder un niveau en informatique particulièrement élevé pour pouvoir utiliser ce logiciel. Son but sera de rester relativement simple d'usage et ergonomique, même si, vu son usage dédié à un secteur très spécifique, la lecture d'une page d'instructions expliquant le fonctionnement global de l'application sera probablement nécessaire pour une bonne prise en main de l'application par un utilisateur lambda.

### 3 Fonctionnalités du système

#### 3.1 Diagramme de Gantt (Vue)

Cette fonction a pour but de représenter des rendez-vous d'un parcours, pour un patient, sur une journée. Ces rendez-vous seront, depuis le diagramme de Gantt, déplaçables dans le temps (via drag-and-drop), sur différentes ressources, et modifiables.

### 3.2 Filtrage (Vue)

Filtrage des données affichées dans le diagramme de Gantt par patient, ressource, ou parcours-patient.

### 3.3 Base de données (Modèle)

Sauvegarde dans une base de données des données patient, ressource, RDV, parcours et précédence au sein du parcours.

### 3.4 Import/Export d'instances/solutions (Modèle)

Le client possède un ordonnanceur externe à l'application. Celui-ci prend en entrée des instances (patients/ressources/parcours) et propose des solutions (horaires des rendez-vous). Ainsi, il faut que l'application puisse charger les instances et les solutions du client afin de pouvoir les visualiser/modifier graphiquement.

### 3.5 Contraintes (Contrôleur)

Les contraintes devront permettre de s'assurer que les manipulations réalisées par l'utilisateur soient possibles, c'est-à-dire ne pas assigner à un patient deux rendez-vous en même temps, et ne pas utiliser une ressource à chaque instant par plus de rendez-vous qu'il n'y a de cette ressource de disponible.

### 3.6 Ordonnancement (Contrôleur)

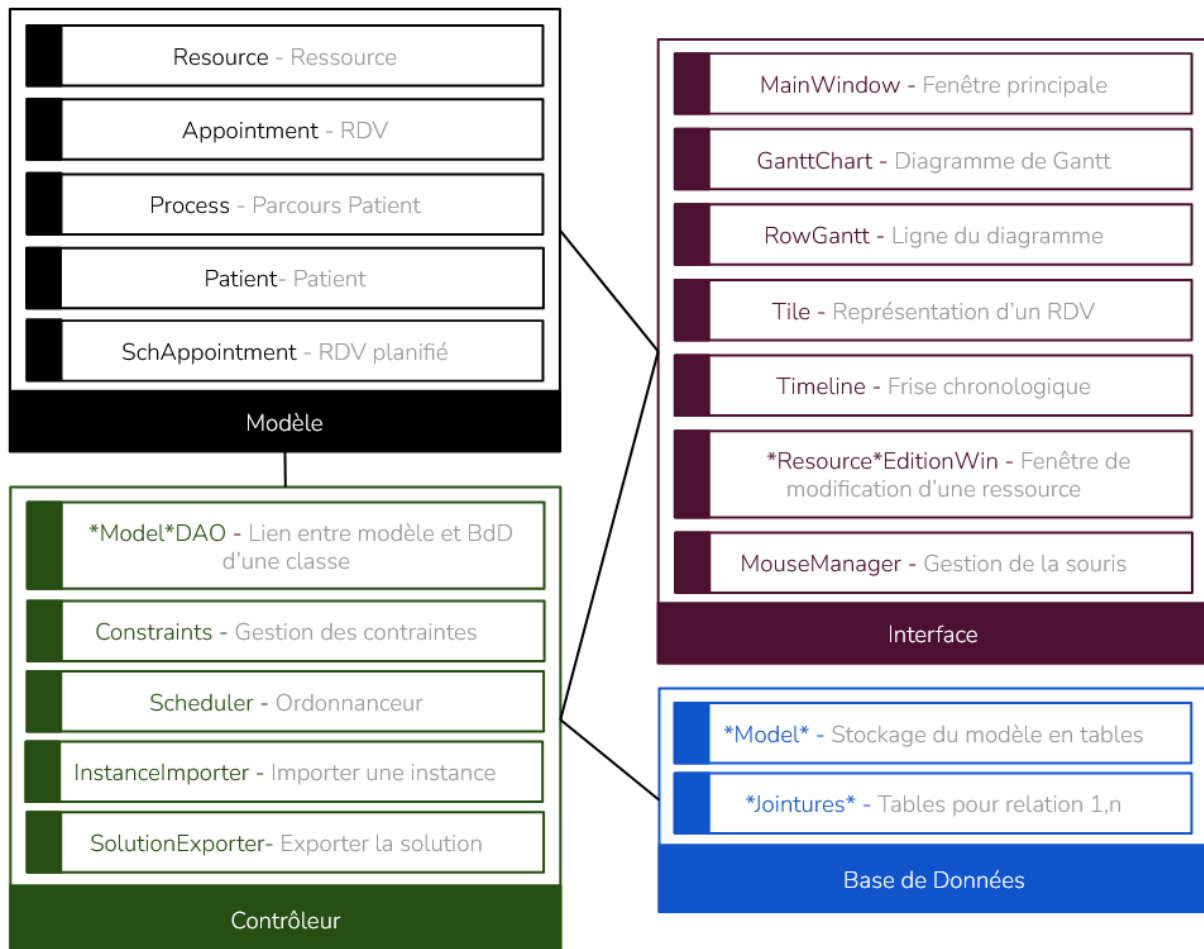
Placer les différents RDV de manière automatique. L'algorithme qui sera implémenté sera simple et sera destiné à être remplacé.

## 4 Structure générale du système

Voici un diagramme représentant les classes importantes de cette application. Elles sont ici découpées en quatre blocs en fonction de leur type. Ce schéma représente les différentes classes qui apparaîtront dans l'application durant le développement de celle-ci.

Ces classes ont été découpées suivant le modèle MVC. Ainsi, les modifications par l'Interface du Modèle passent par le Contrôleur, et celui-ci gère également la répercussion des changements du modèle Java sur la base de données. Toutefois, la récupération d'informations du Modèle par l'Interface est permise.

À noter que les éléments listés ci-dessous sont bien des classes, à l'exception de ceux de la Base de Données.



**Figure 2.1** – *Le modèle de l'application*

# 3

## État de l'art / Veille technologique

### 1 Logiciels de gestion de parcours patients

J'ai dans un premier temps tenté de trouver des logiciels proposant des fonctionnalités similaires à ce que le client recherchait. De ces recherches, deux familles de logiciels sont apparues.

- Les logiciels de gestion de parcours patient comprenant le terme "parcours patient" comme "l'ensemble des rendez-vous qu'un patient a eu durant sa vie". Ces logiciels sont donc plus axés sur la représentation suivant une frise temporelle de documents associés à la vie médicale d'une personne : ainsi, ils sont très éloignés de ce que l'on recherche ici.
- Les logiciels de planification au sens général ; peu importe ce qu'il s'agit faut planifier, de nombreux logiciels proposent des interfaces type Gantt, et proposent de déplacer ces rendez-vous afin de créer un planning.

Ainsi, ce qui émerge de ces recherches est qu'il n'existe pas à ma connaissance de logiciel faisant ce qui est recherché par le client. Néanmoins, l'interface pourra s'inspirer des nombreux exemples de logiciels de planification qui existent déjà.

### 2 Bibliothèques de diagramme de Gantt

Le langage dans lequel l'application devrait être développée n'étant pas spécifié, j'ai pu chercher des bibliothèques dans plusieurs langages. Premièrement, dans ceux que je connaissais bien, comme le C++, le Java, et le Python, mais aussi en Javascript, comme suggéré par mon encadrant.

Une liste de critères à remplir pour la sélection de la bibliothèque a été dressée. J'ai donc, dans un premier temps, grossièrement cherché les bibliothèques semblant remplir au moins une partie de ces critères, puis j'ai testé ou approfondi mes recherches sur celles qui restaient afin de savoir laquelle conviendrait le mieux. Voilà un résumé des recherches et conclusions auxquelles j'ai abouti.

#### 2.1 Critères

Premièrement, voici la liste de critères sur le diagramme de Gantt que la bibliothèque doit permettre d'implémenter :

- Possibilité d'avoir plusieurs rendez-vous sur une seule ligne.

- Pouvoir déplacer les rendez-vous via un glisser-déposer.
- Avoir une granularité temporelle de l'ordre de 5 minutes.
- Librairie libre.

Ces critères représentaient le minimum qu'une librairie devait posséder. Par la suite, il faudrait se renseigner sur comment l'utiliser pour récupérer les rendez-vous déplacés afin de répercuter les changements effectués sur la GUI directement sur le modèle.

## 2.2 C++

La librairie **CChart** est une librairie extrêmement dense, avec pour seule documentation un manuel en chinois. La quasi-totale absence de commentaires dans le code rend son utilisation très complexe sinon impossible.

## 2.3 Java

En Java, la première librairie que j'ai trouvée est **JFreeChart**. Elle a le défaut de ne pas avoir de glisser-déposer implémenté, et de contenir beaucoup plus qu'un diagramme de Gantt, ce qui augmente aussi logiquement sa complexité. Elle doit néanmoins pouvoir permettre son ajout via les méthodes de Mouse Event des librairies Java de base. Toutefois, elle a été conçue pour être utilisée en tant que servlet, domaine avec lequel je suis très loin d'être familier, ce qui rend son usage encore plus complexe.

La seconde librairie à avoir attiré mon attention est **DGantt**. La documentation relative à cette librairie est très faible, et les descriptions ou exemples de ses fonctionnalités sont plus que rares. Elle a de plus été développée par ce qui semble être une personne seule, et abandonnée depuis 8 ans. Cela rend son usage potentiellement risqué. Toutefois, après l'avoir testée, elle s'est révélée être étonnamment bien organisée, développée et commentée ; mais elle a aussi montré son incapacité à implémenter la présence de multiples rendez-vous par ligne. Cela signifie donc qu'une partie importante de cette librairie loin d'être petite devrait être réécrite pour être utilisée.

## 2.4 Python

La première librairie Python que j'ai pu évaluer est la librairie **Gantttrify**. Les défauts immédiats de la librairie sont son impossibilité à obtenir l'échelle désirée, son absence de glisser-déposer, et sa qualité inconnue étant donné qu'il s'agit de nouveau d'une librairie développée par un utilisateur seul.

Ensuite, je me suis tourné vers **Dash Plotly**. Cette librairie ne propose pas de réelle interactivité, ce qui demanderait des modifications assez importantes. Toutefois, il s'agit d'une librairie importante, donc probablement sûre.

## 2.5 Javascript

Enfin, la librairie **GSTC (Gantt Schedule Timeline Calendar)** a, après quelques tests, remplis tous les critères de la liste. Il s'agit néanmoins d'une librairie Javascript, langage avec lequel je ne suis pas familier. À première vue, cette librairie ne possède aucun défaut particulier. Néanmoins, après mise en place, il s'est avéré que celle-ci ne pouvait s'exécuter sans connexion Internet au démarrage : en effet, cette librairie contacte un serveur pour s'assurer que l'utilisateur possède bien une clé de licence.



Ces clés de licence sont fournies en quantité illimitée, et sont directement intégrées au code de la librairie. Ainsi, lors du téléchargement, je n'avais eu qu'à l'exécuter sur un serveur local pour qu'elle fonctionne normalement, mais tout usage sans serveur provoquait son arrêt sans avertissement. C'est en fouillant le code de la librairie que j'ai compris que cela était dû à l'impossibilité de contacter le serveur des possesseurs de la librairie comme dit précédemment.

C'est un problème qui ne m'avait pas sauté aux yeux dès le début, car la librairie est sous une licence qui permet son usage pour un cadre non-commercial. Néanmoins, dans le cadre d'un usage commercial, l'achat d'une licence serait obligatoire. Cette nouvelle restriction a fait que le client et le tuteur académique ont décidé que cette librairie ne devrait pas être utilisée. Ainsi, étant donné le reste des librairies de la liste, j'ai décidé de partir sur une librairie graphique autre et de créer la partie Interface/Gantt moi-même, par volonté de gain de temps.

### 3 Librairies pour réaliser une Interface Utilisateur Graphique

Comme indiqué précédemment, la seule librairie offrant directement un diagramme de Gantt répondant à nos critères ayant été écartée, il a fallu que je me tourne vers des librairies de plus bas niveau en C++ ou Java afin d'implémenter l'interface moi-même. La première librairie que j'ai trouvée est [Java2D](#).

Il s'agit d'une librairie graphique basique en Java, qui est intégrée par défaut dans les JDK, et qui possède des méthodes à surcharger permettant de réaliser précisément les tâches de glisser-déposer que je voulais. J'ai donc pris une journée pour l'implémenter dans un petit projet de diagramme de Gantt basique, afin de voir si toutes les fonctionnalités que je voulais étaient bien présentes. Ce test a été un succès, et il a été facile de déplacer des blocs, de déclencher des événements en lien à leur position et changements de position, de gérer les collisions entre blocs, etc. J'ai donc fait le choix d'adopter cette librairie.

### 4 Modélisation proposée

# 4

## Mise en oeuvre

### 1 Outils et librairies utilisés

Le JRE pour tester et exécuter le .jar de l'application est à la version 13 (JRE 13).

Les librairies utilisées dans le projet sont les suivantes :

- JUnit 4.13.2, pour effectuer des tests unitaires.
- Hamcrest core 1.3, qui accompagne JUnit en implémentant des méthodes permettant les tests unitaires simplement.
- SQLite JDBC 3.36.0.3 pour gérer la base de donnée intégrée.
- JGoodies Forms 1.8.0 est une librairie aidant Swing, qui est elle-même une librairie graphique incluse dans le JRE.

De plus, les outils utilisés en parallèle sont :

- L'IDE Eclipse pour le développement en Java.
- WindowBuilder 1.9.7, add-on d'Eclipse pour la création d'interfaces graphiques.
- DB Browser for SQLite version 3.12.1, logiciel permettant de consulter et éditer une base de données au format SQL.

### 2 Éléments d'implémentation, choix techniques

Dans les choix techniques effectués, il est à noter que les DAO ont été réalisés à la main. Cela a permis d'interagir avec la base de données exactement comme voulu, mais a pour limite que les modifications de l'application seront plus complexes, quoi que l'installation du projet sera simple en contrepartie.

### 3 Analyse des résultats, évaluation, qualité

Je n'ai pas disposé d'assez de temps pour pouvoir implémenter des tests unitaires et de non régression autant que désiré. Ainsi, certains tests en très petite quantité en JUnit ont pu être ajoutés seulement. Le bon fonctionnement de l'application a sinon été testé dans des tests unitaires informels qui n'ont pas été gardés. Idem pour les tests fonctionnels.

La qualité a été garantie autant que possible via un certains nombres de mesures. Ainsi, parmi les plus triviales, on peut retrouver les commentaires en tant qu'en tête des méthodes et classes, mais aussi au sein des méthodes, en tant qu'explications, avec des conventions de nommages respectées. Des efforts ont également été faits pour respecter le plus possible le modèle MVC, et rendre cette découpe évidente via la séparation en packages (un package "model", un "controler", et un "gui").

# 5

## Bilan et conclusion

### 1 Bilan du semestre 9

#### 1.1 Tâches faites

##### **Présentation des sujets et choix du PRD**

La première semaine a été dédiée à la présentation de tous les sujets de PRD parmi lesquels choisir, puis au choix.

##### **Cahier des charges / spécifications**

Temps accordés à la rédaction, dans un premier temps, d'un cahier des charges, puis, dans un second, d'un cahier des spécifications.

##### **Modélisation**

Créneaux accordés à la découpe du projet en plus petites tâches, du logiciel en classes, et des données en objets ou tables de la base de données.

##### **Recherche de technologies**

Recherche afin de savoir quelles librairies utiliser concernant l'affichage d'un diagramme de Gantt ; soit des librairies permettant directement son affichage, soit une donnant des outils permettant d'atteindre ce but.

##### **Tests de technologies**

Implémentation des librairies susmentionnées dans des projets de test afin de savoir si elles convenaient bel et bien aux attentes du client.

##### **Apprentissage de librairies**

Création d'un projet test plus dense afin de découvrir l'usage de manière plus exhaustive d'une librairie.

### 1.2 Tâches en cours

#### **RDV Client**

Tous les rendez-vous planifiés avec le client permettant de vérifier que le projet avance dans la bonne direction, que la modélisation ou que le cahier des spécifications correspond à ses attentes.

#### **Rédaction du rapport**

Temps passé à la rédaction du rapport, à recherche d'outils compilant du Latex, et à leur utilisation.

#### **Interface - Gantt**

Temps sur lesquels j'ai travaillé à implémenter la partie Gantt de l'onglet 1 de l'interface.

#### **Interface - BDD**

Créneaux attribués à la modélisation de l'onglet 2 de l'interface.

#### **Modèles - Gantt**

Écriture des classes représentant les données affichées dans le diagramme de Gantt (Rendez-vous planifiés, Cases, Lignes, ...).

#### **Modèles - BDD**

Classes Patients, Ressources, Parcours patients, Rendez-vous.

#### **Modèles - Import/Export**

Temps accordé à l'écriture des méthodes permettant l'import d'une base de données à partir de fichiers.

#### **Contrôleur - Lien IHM/Modèle**

Créneaux dédiés à l'écriture des méthodes permettant l'export de la base de données vers des fichiers.

### 1.3 Tâches à effectuer

#### **Interface - Modification**

Créneaux attribués à la modélisation des fenêtres permettant la modification des champs d'un objet.

#### **Contrôleur - Lien BDD**

Partie permettant de faire le lien entre les modifications appliquées aux objets Java et leurs équivalents dans la base de données.

#### **Contrôleur - Contraintes**

Écriture des fonctions autorisant la vérification que les contraintes sont respectées.

#### **Contrôleur - Ordonnancement**

Temps accordé à l'ajout d'un ordonnanceur basique.

### Validation

Partie des rendez-vous avec le client permettant d'établir que le logiciel fonctionne comme il l'entend, tout au long de son développement.

## 2 Bilan du semestre 10

Le bilan global est mitigé. En effet, la quantité de travail à faire a été mal évaluée, ce qui a conduit à de nombreux retards sur beaucoup de fonctionnalités prévues au S9, qui ne seront finalement pas livrés.

## 3 Bilan sur la qualité

La qualité du code est un bon qui n'est pas sans faute. En effet, bien que la majorité du code soit correcte et commentée, il a été relevé des écarts du modèle MVC, notamment dans la partie Gantt. De plus, l'absence de tests de non-régression ne permet pas de garantir la qualité du code.

## 4 Bilan auto-critique

Ma principale erreur a été de sous-estimer une tâche que je pensais connaître. En effet, j'avais déjà réalisé des connexions à des bases de données et des interfaces auparavant, mais jamais à des échelles de projet "réel". Cela m'a conduit à ne pas assez me concentrer sur l'estimation de ces tâches, et à des retards suite à leur sous-estimation.

## Annexes

# A

## Planification, gestion de projet

### 1 Evolution du projet

Le diagramme de Gantt Initial pour la planification de ce projet.

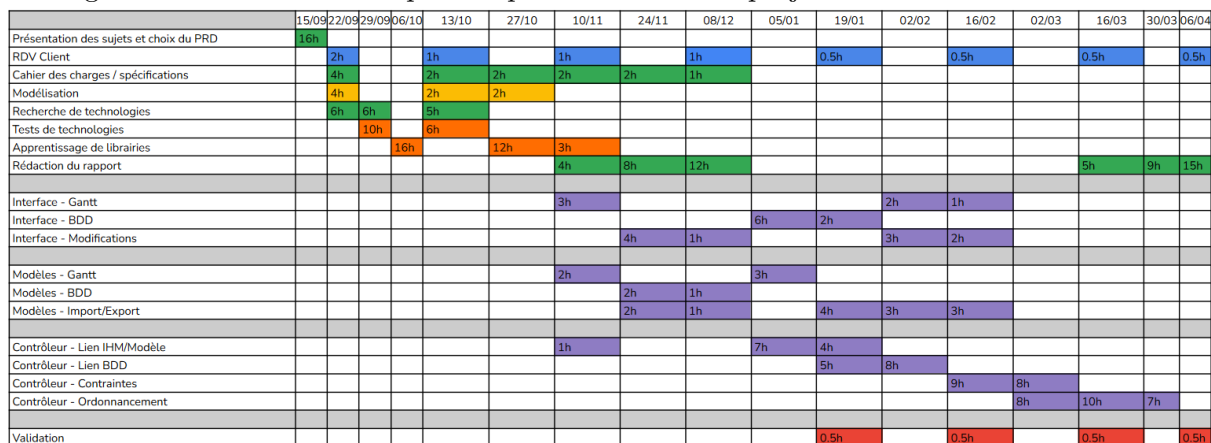


Figure A.1 – Le diagramme de Gantt Initial

A noter que ce diagramme n'a pas été réalisé dès le début du projet ; il a été réalisé début novembre. Ainsi, la raison pour laquelle la tâche "Test de technologies" est interrompue puis reprend correspond au fait que, après une réunion avec le client, la librairie que j'avais sélectionné ne correspondait pas pleinement aux attentes.

En effet, il s'agissait d'une librairie propriétaire : ce critère a donc été ajouté, et cette librairie éliminée des options. Il a donc fallu reprendre la recherche de librairies.

### 2 Description des tâches

#### Présentation des sujets et choix du PRD

- Date de début : 15/09/2019
- Date de fin : 21/09/2019
- Durée : 16 heures



- Description : La première semaine a été dédiée à la présentation de tous les sujets de PRD parmi lesquels choisir, puis au choix.

### Cahier des charges / spécifications

- Date de début : 22/09/2021
- Date de fin : 08/12/2021
- Durée : 13 heures
- Description : Temps accordés à la rédaction, dans un premier temps, d'un cahier des charges, puis, dans un second, d'un cahier des spécifications.

### Modélisation

- Date de début : 22/09/2021
- Date de fin : 27/10/2021
- Durée : 8 heures
- Description : Créneaux accordés à la découpe du projet en plus petites tâches, du logiciel en classes, et des données en objets ou tables de la base de données.

### Recherche de technologies

- Date de début : 22/09/2021
- Date de fin : 13/10/2021
- Durée : 16 heures
- Description : Recherche afin de savoir quelles librairies utiliser concernant l'affichage d'un diagramme de Gantt ; soit des librairies permettant directement son affichage, soit une donnant des outils permettant d'atteindre ce but.

### Tests de technologies

- Date de début : 29/09/2021
- Date de fin : 13/10/2021
- Durée : 16 heures
- Description : Implémentation des librairies susmentionnées dans des projets de test afin de savoir si elles convenaient bel et bien aux attentes du client.

### Apprentissage de librairies

- Date de début : 06/10/2021
- Date de fin : 10/11/2021
- Durée : 31 heures
- Description : Création d'un projet test plus dense afin de découvrir l'usage de manière plus exhaustive d'une librairie.

### RDV Client

- Date de début : 22/09/2021
- Date de fin : 06/04/2022
- Durée : 7 heures
- Description : Tous les rendez-vous planifiés avec le client permettant de vérifier que le projet avance dans la bonne direction, que la modélisation ou que le cahier des spécifications correspond à ses attentes.

### Rédaction du rapport

- Date de début : 10/10/2021
- Date de fin : 06/04/2022
- Durée : 53 heures
- Description : Temps passé à la rédaction du rapport, à recherche d'outils compilant du Latex, et à leur utilisation.

### Interface - Gantt

- Date de début : 10/10/2021
- Date de fin : 16/02/2022
- Durée : 6 heures
- Description : Temps sur lesquels je travaille à implémenter la partie Gantt de l'onglet 1 de l'interface.

### Interface - BDD

- Date de début : 05/01/2022
- Date de fin : 19/01/2022
- Durée : 8 heures
- Description : Créneaux attribués à la modélisation de l'onglet 2 de l'interface.

### Interface - Modification

- Date de début : 24/12/2021
- Date de fin : 16/02/2022
- Durée : 10 heures
- Description : Créneaux attribués à la modélisation des fenêtres permettant la modification des champs d'un objet.

### Modèles - Gantt

- Date de début : 10/10/2021
- Date de fin : 05/01/2022
- Durée : 5 heures
- Description : Écriture des classes représentant les données affichées dans le diagramme de Gantt (Rendez-vous planifiés, Cases, Lignes, ...).

### Modèles - BDD

- Date de début : 24/11/2021
- Date de fin : 08/12/2021
- Durée : 3 heures
- Description : Classes Patients, Ressources, Parcours patients, Rendez-vous.

### Modèles - Import/Export

- Date de début : 24/11/2021
- Date de fin : 16/02/2022
- Durée : 13 heures
- Description : Temps accordé à l'écriture des méthodes permettant l'import d'une base de données à partir de fichiers.

## Contrôleur - Lien IHM/Modèle

- Date de début : 10/11/2021
- Date de fin : 19/01/2022
- Durée : 12 heures
- Description : Créneaux dédiés à l'écriture des méthodes permettant l'export de la base de données vers des fichiers.

## Contrôleur - Lien BDD

- Date de début : 19/01/2021
- Date de fin : 02/02/2022
- Durée : 13 heures
- Description : Partie permettant de faire le lien entre les modifications appliquées aux objets Java et leurs équivalents dans la base de données.

## Contrôleur - Contraintes

- Date de début : 16/02/2022
- Date de fin : 02/03/2022
- Durée : 17 heures
- Description : Écriture des fonctions autorisant la vérification que les contraintes sont respectées.

## Contrôleur - Ordonnancement

- Date de début : 02/03/2022
- Date de fin : 30/03/2022
- Durée : 25 heures
- Description : Temps accordé à l'ajout d'un ordonnanceur basique.

## Validation

- Date de début : 19/01/2022
- Date de fin : 06/04/2022
- Durée : 2 heures
- Description : Partie des rendez-vous avec le client permettant d'établir que le logiciel fonctionne comme il l'entend, tout au long de son développement.

Diagramme de Gantt final :

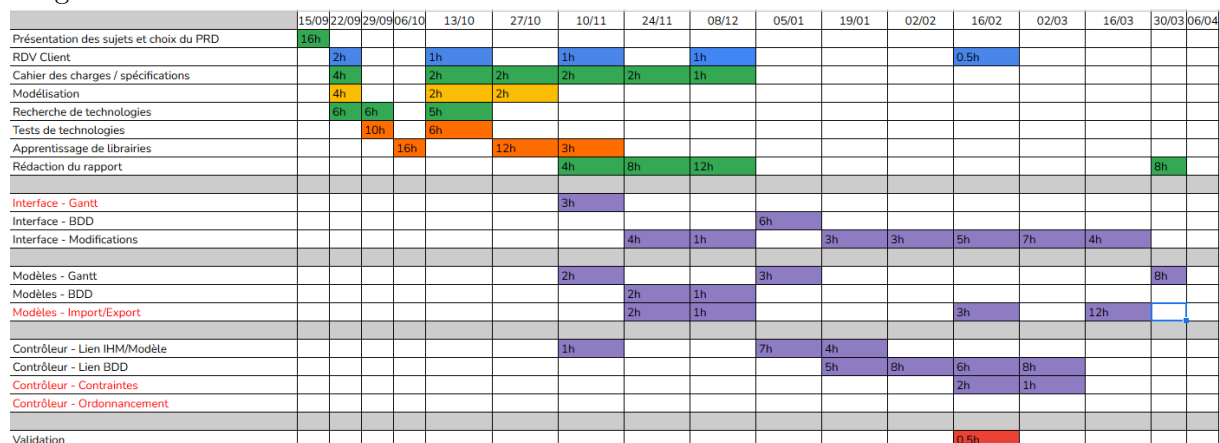


Figure A.2 – Le diagramme de Gantt Final

# B

## Description des interfaces

### 1 Interfaces matérielles/logicielles

Le logiciel devra tourner sur un ordinateur personnel typique et sous le système d'exploitation Windows 10.

### 2 Interfaces homme/machine

L'application comportera deux onglets : un représentant le diagramme de Gantt, et un autre permettant de gérer toutes les données (Ressources, Patient, ...) disponibles. Voici donc des maquettes du premier onglet :

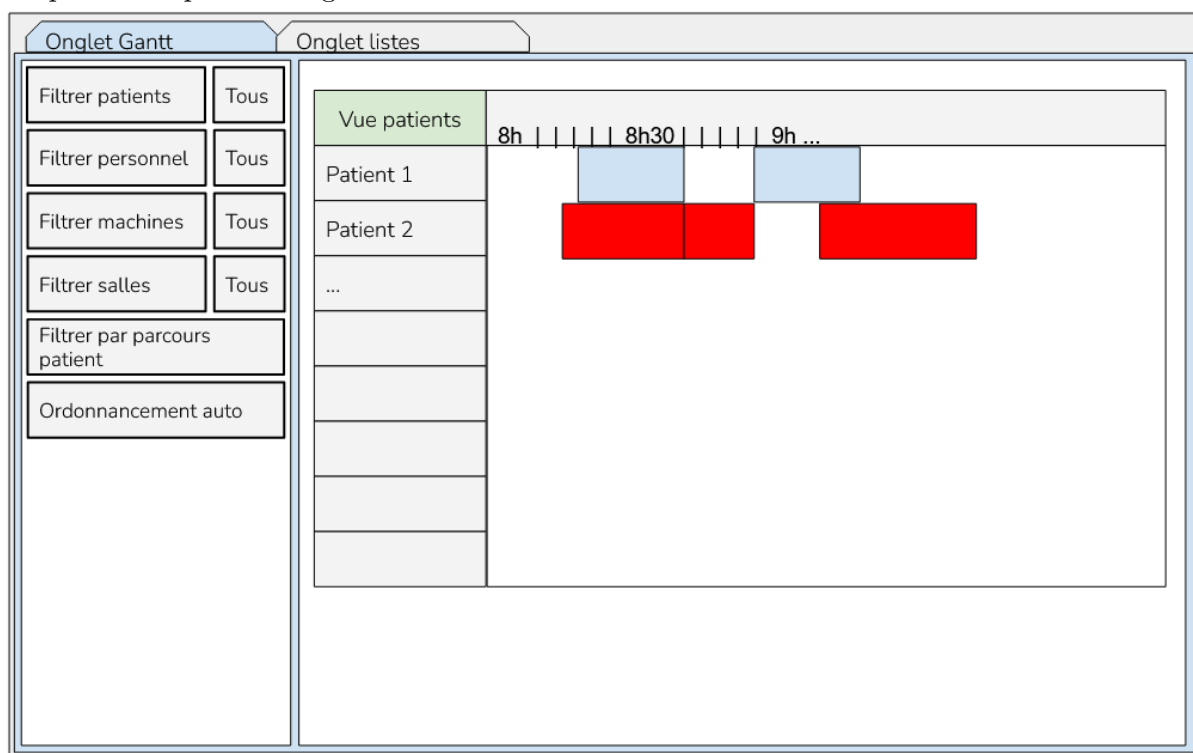
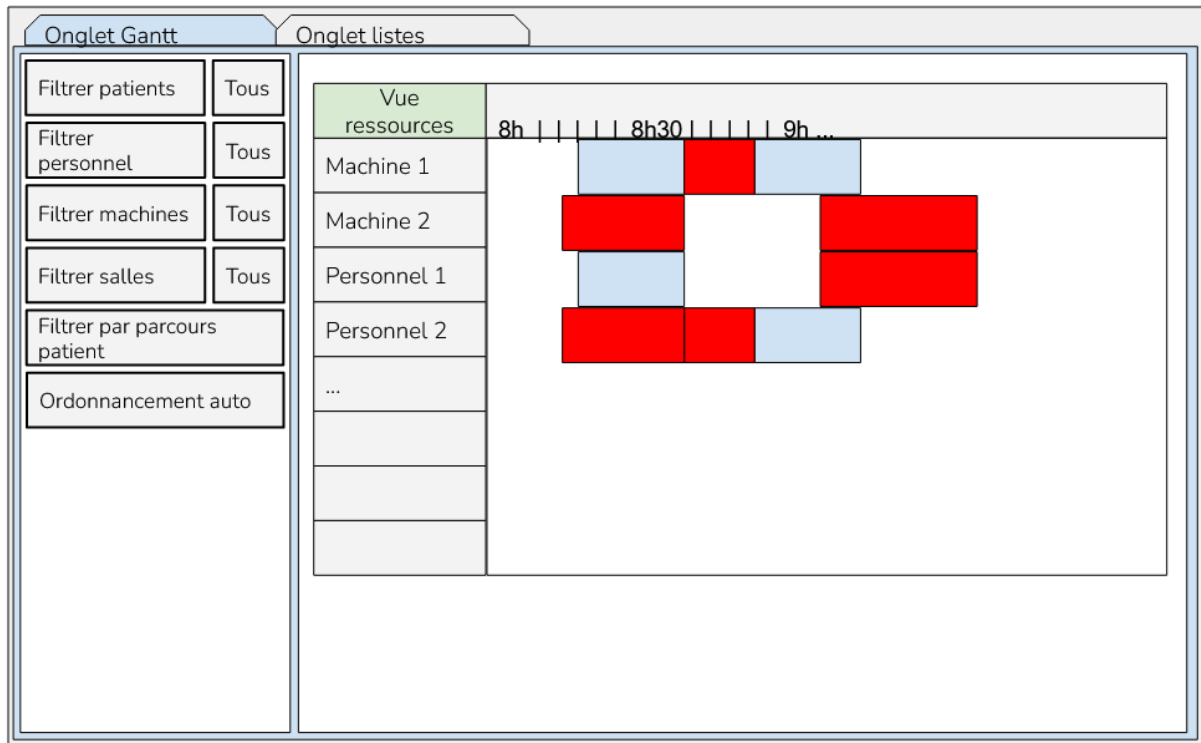
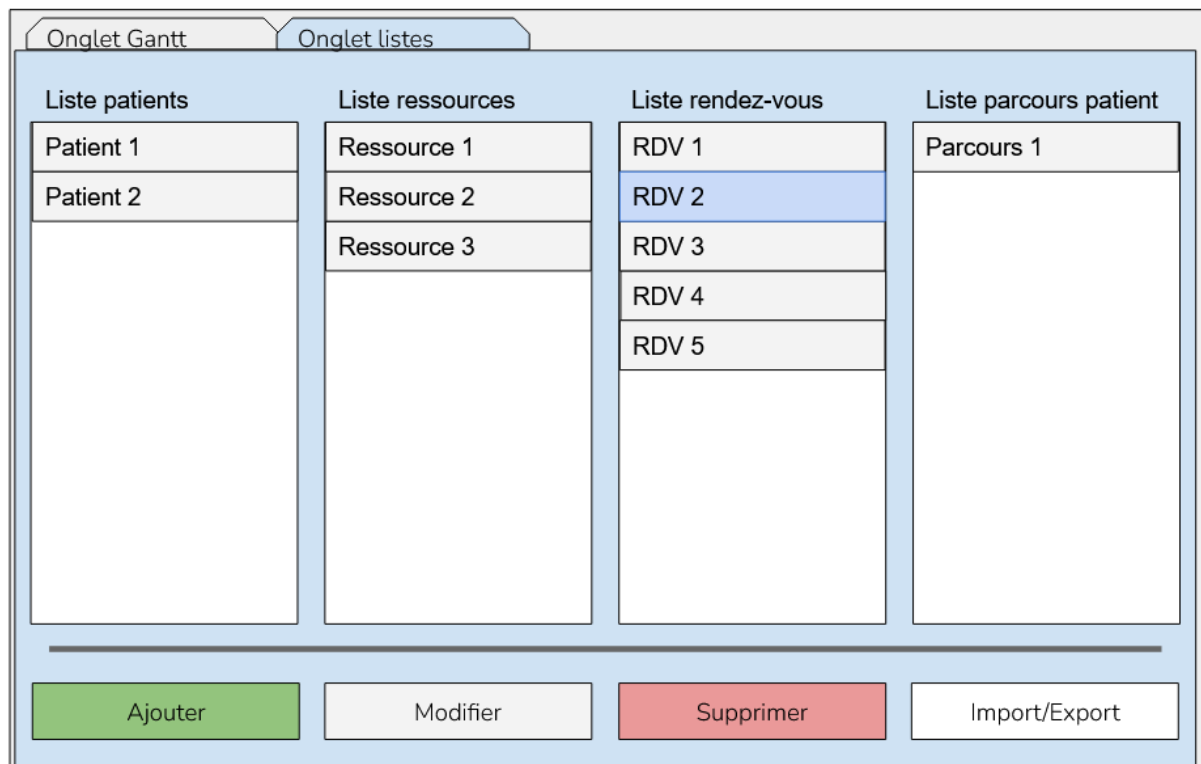


Figure B.1 – Maquette de l'onglet 1 de l'application, vue Patient



**Figure B.2** – Maquette de l'onglet 1 de l'application, vue Ressource

Ensuite, voici la maquette du second onglet, responsable de la modification des données :



**Figure B.3** – Maquette de l'onglet 2 de l'application

Une fenêtre pop-up se chargera de permettre la modification des champs d'un objet quelconque lors d'un clic sur les boutons "Ajouter" ou "Modifier" :

Pop-up de modification d'un élément

Champ 1 (texte non modifiable) :

Champ 2 (texte modifiable) :

Champ 3 (choix d'une liste) :

**Figure B.4** – Maquette de la fenêtre type de modification des données

Enfin, le bouton Import/Export doit renvoyer vers une fenêtre permettant d'importer une solution à partir de certains fichiers :

Pop-up d'import de données

Fichier dictionnaire (.dict) :

Fichier instance (.msrcmp) :

Fichier solution (.sol) :

**Figure B.5** – *Maquette de la fenêtre d'import/export*



# Cahier de Spécifications

## 1 Spécifications Fonctionnelles

### 1.1 Fonctionnalités à développer

#### Diagramme de Gantt (Vue)

Cette fonction a pour but de représenter des rendez-vous d'un parcours, pour un patient, sur une journée. Ces rendez-vous seront, depuis le diagramme de Gantt, déplaçables dans le temps (via glisser-déposer), sur différentes ressources, et modifiables. Elle interagira avec le modèle directement et la partie du contrôleur chargée de modifier le modèle et la base de données.

#### Représentation d'un rendez-vous comme case d'un diagramme de Gantt

Entrée : Rendez-vous planifiés

Sortie : Graphique

Préconditions : /

Postconditions : /

Priorité : Haute

#### Déplacement par glisser-déposer d'un rendez-vous

Entrée : Case / rendez-vous du diagramme de Gantt et capture du curseur de la souris

Sortie : Graphique

Préconditions : /

Postconditions : Le rendez-vous du modèle est déplacé en accordance au mouvement.

Priorité : Haute

#### Vue Patient et vue Ressources (les lignes du Gantt représentent des Patients ou des Ressources) ■

Entrée : Rendez-vous planifiés, Patients, Ressources

Sortie : Graphique

Préconditions : /

Postconditions : /

Priorité : Haute

#### Filtres afin de réduire le nombre de Patients/Ressources/Rendez-vous affichés

Entrée : Rendez-vous planifiés, Patients, Ressources

Sortie : Graphique



Préconditions : /  
 Postconditions : /  
 Priorité : Moyenne

**Voir par un clic tous les rendez-vous sélectionnés en vue Ressource**

Entrée : Rendez-vous planifiés  
 Sortie : Graphique  
 Préconditions : /  
 Postconditions : /  
 Priorité : Moyenne

**Modifier un rendez-vous par un double-clic sur sa case**

Entrée : Rendez-vous planifiés  
 Sortie : Graphique, Rendez-vous modifié  
 Préconditions : /  
 Postconditions : /  
 Priorité : Moyenne

**Base de données (Modèle)**

Sauvegarde dans une base de données des données patient, ressource, RDV, parcours et précédence au sein du parcours. Elle recevra des ordres du contrôleur uniquement.

**Interaction avec la base de données**

Entrée : Modèle de données  
 Sortie : Données  
 Préconditions : /  
 Postconditions : /  
 Priorité : Moyenne

**Import/Export d'instances/solutions (Modèle)**

Le client possède un ordonnanceur externe à l'application. Celui-ci prend en entrée des instances (patients/ressources/parcours) et propose des solutions (horaires des rendez-vous). Ainsi, il faut que l'application puisse charger les instances et les solutions du client afin de pouvoir les visualiser/modifier graphiquement. Elle interagira avec le contrôleur de la base de données.

**Import via un ensemble de fichiers**

Entrée : Fichier .dict (nom des ressources/rendez-vous/patients), fichier .msrcmp (description des interactions patients/parcours, rendez-vous/ressources, ...) = fichier instance, fichier .sol (placement des rendez-vous sur une journée) = fichier solution.  
 Sortie : Objets décrivant un ensemble de rendez-vous, ressources, patients, parcours.  
 Préconditions : /  
 Postconditions : /  
 Priorité : Haute

**Export en un ensemble de fichiers**

Entrée : Objets décrivant un ensemble de rendez-vous, ressources, patients, parcours.  
 Sortie : Fichier .dict (nom des ressources/rendez-vous/patients), fichier .msrcmp (description des interactions patients/parcours, rendez-vous/ressources, ...) = fichier instance, fichier .sol (placement des rendez-vous sur une journée) = fichier solution.  
 Préconditions : /  
 Postconditions : /  
 Priorité : Moyenne

**Contraintes (Contrôleur)**

Les contraintes devront permettre de s'assurer que les manipulations réalisées par l'utilisateur soient possibles, c'est-à-dire ne pas assigner à un patient deux rendez-vous en même temps, et ne

pas utiliser une Ressource plus qu'elle n'est disponible (4 machines d'un type donné implique que quatre RDV maximum peuvent utiliser ce type de machine simultanément). Elle interagira avec le modèle Java et l'interface.

#### Vérification des contraintes

Entrée : Rendez-vous planifiés, Patients, Ressources

Sortie : Liste des Rendez-vous planifiés ne respectant pas les contraintes

Préconditions : /

Postconditions : /

Priorité : Haute

#### Ordonnancement (Contrôleur)

Placer les différents RDV de manière automatique. L'algorithme qui sera implémenté sera simple et destiné à être remplacé. Cette fonctionnalité interagira avec l'interface et le modèle de données.

#### Ordonnanceur Premier Arrivé Premier Servi

Entrée : Rendez-vous, Patients, Ressources

Sortie : Liste de rendez-vous planifiés

Préconditions : /

Postconditions : Les rendez-vous planifiés ne briseront pas les contraintes, même si cela signifie qu'ils ne seront pas tous placés

Priorité : Haute

## 2 Spécifications non fonctionnelles

### 2.1 Contraintes de développement et conception

Le client n'a pas exprimé de contrainte relative au langage ou aux outils utilisés pour le développement. Néanmoins, l'utilisation de bibliothèques libres, et non propriétaires est une contrainte qui a été exprimée par le client. L'application devra être fonctionnelle fin Mars, et toutes les fonctionnalités listées précédemment seront attendues.

### 2.2 Contraintes de fonctionnement et d'exploitation

#### 2.2.1 Performances

Le logiciel n'a pas de contrainte de performance particulière. Il doit donc tourner de manière fluide, avec des ralentissements de quelques secondes au temps de chargement permis.

#### 2.2.2 Capacités

Le logiciel fonctionnera avec une base de données intégrée, ce qui signifie que seule l'application sera cliente de la base de données. La taille maximale d'une instance avec laquelle le logiciel devrait rester complètement fluide est de 50 Patients, 500 Rendez-vous, et 30 Ressources.

Par conséquent, chaque fichier de base de données ne devrait jamais atteindre trop conséquente (<50 Mo). De plus, les algorithmes en Java ne devraient pas rencontrer trop de difficultés à gérer un tel nombre d'objets.

### 2.2.3 Contrôlabilité

Le client n'a pas souhaité ajouter de contrôlabilité particulière pour l'application.

### 2.2.4 Sécurité

Le logiciel n'a pas de composante sécurité importante : il est destiné à tourner en local, et n'a pas pour but d'être distribué dans un cas réel, où les données des patients devraient être protégées.

# D

# Cahier du développeur

## 1 Introduction

Ce chapitre va décrire plus en détail, et d'un point de vue orienté programmation, la structure globale de l'application et de ses composants.

## 2 Diagrammes architecturaux et UML

Les données de ce projet seront stockées sous deux formes : celle d'objets, et celles d'entrées dans les tables d'une base de données. Les deux graphiques suivants permettent de les décrire.

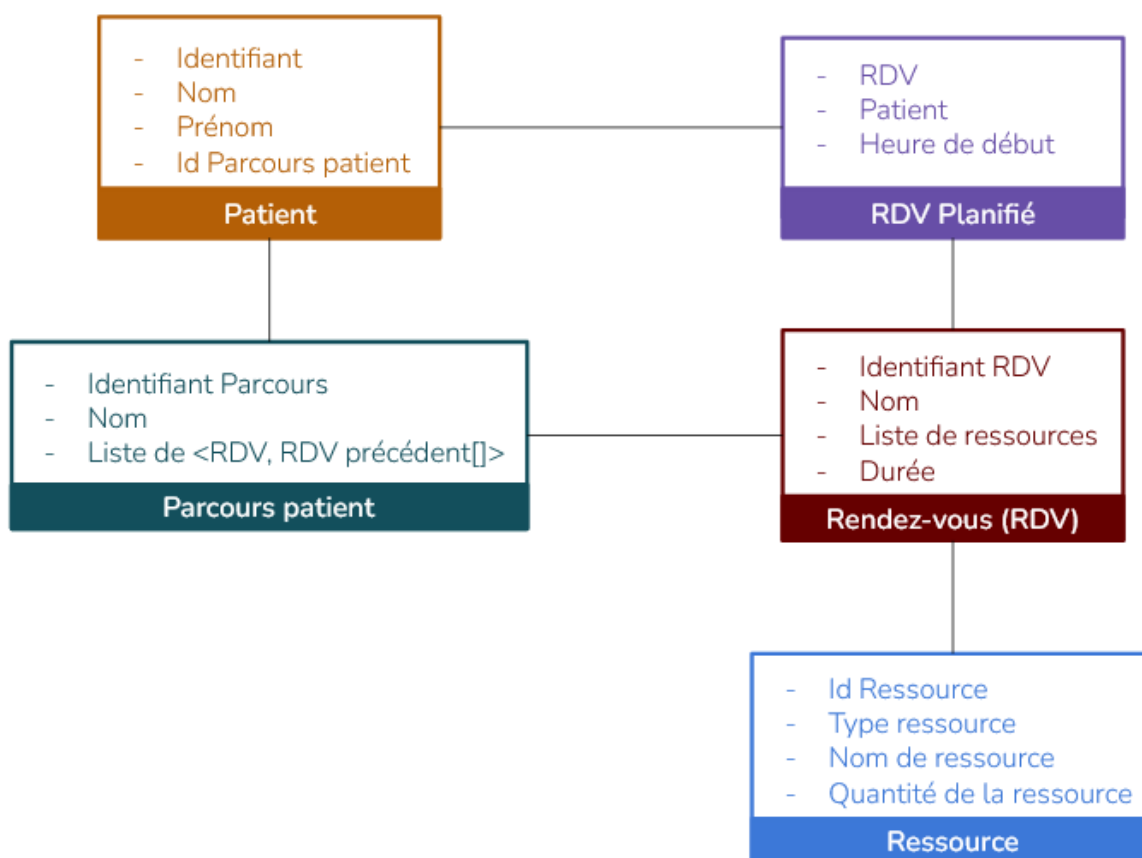


Figure D.1 – Diagramme des classes du modèle et de leurs interactions

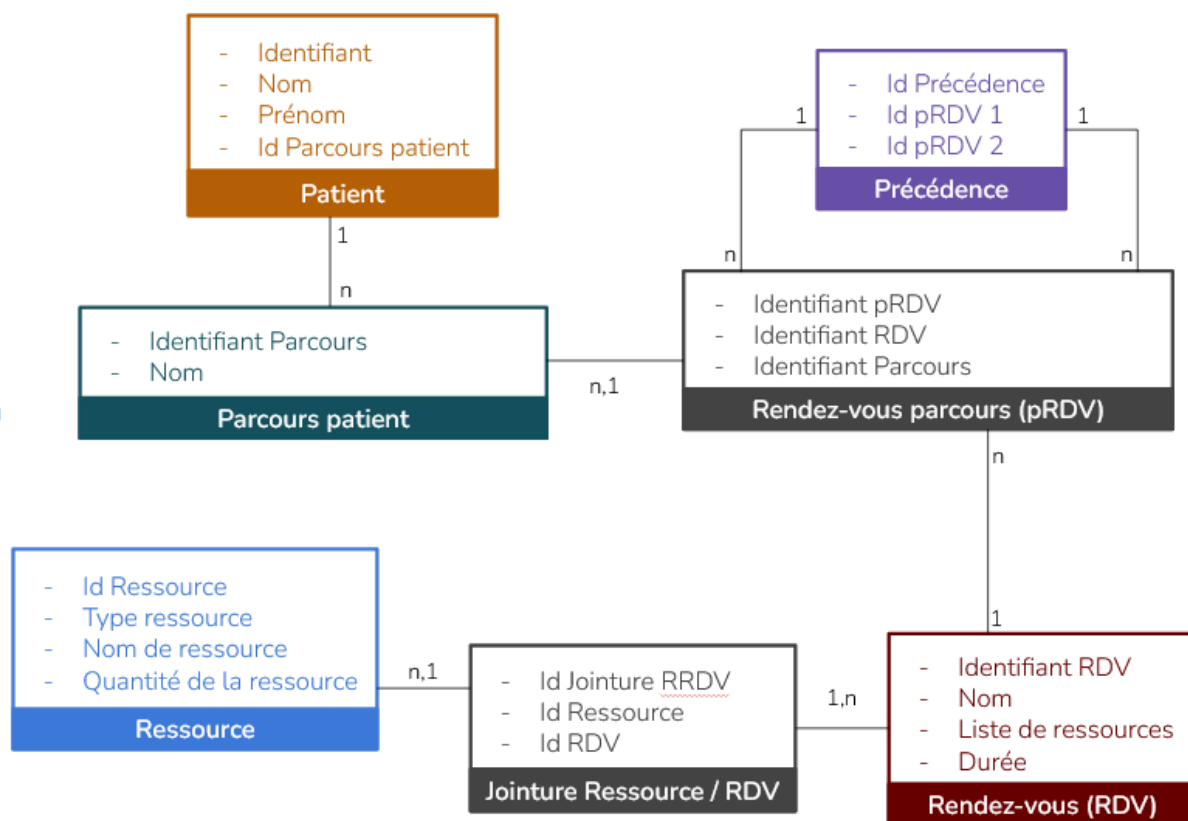


Figure D.2 – Diagramme du modèle de la base de données du projet

### 3 Descriptions détaillées de données exploitées

L'application interagira peu avec des fichiers extérieurs. Néanmoins, il y a tout de même trois types de fichiers, impliqués dans la partie Import/Export, qu'il faut expliquer.

Tout d'abord, il convient de rappeler que ce projet s'effectue autour d'une thèse sur l'ordonnement de rendez-vous dans un hôpital de jour. Dans ce cadre d'ordonnement, deux notions nous intéressent ici : les instances et les solutions.

Les instances décrivent un problème, c'est-à-dire l'intégralité des Ressources, Patients, Rendez-vous, et Parcours. Les solutions décrivent ce problème résolu, donc le placement des rendez-vous sur une journée dans le respect des contraintes.

Ainsi, dans le cadre d'un Import/Export, il va falloir charger ces données. Néanmoins, les fichiers contenant ces données ne leur donnent pas de nom, uniquement des identifiants qui sont leur position dans le fichier. Ainsi, un troisième type de fichier est impliqué : le fichier dictionnaire, faisant le lien entre les données et leurs noms.

#### Fichier instance

L'extension d'un fichier instance est `.msremp`. Son contenu est le suivant :

- Table de correspondance entre les ressources et les compétences
- Activités de chaque parcours patient
- Durée de chaque activité
- Compétences requises par chaque activité
- Relations de précédence entre activités

Dans ce projet, les compétences ne sont pas traitées : ainsi, on considère que chaque Ressource ne possède qu'une seule et unique Compétence, et inversement. Par conséquent, demander pour un rendez-vous une Compétence revient à demander la Ressource qui lui est associée.

#### Fichier solution

L'extension d'un fichier solution est `.sol`. Il ne contient que des listes des dates de fin des activités, par bloc (un bloc étant un parcours).

#### Fichier dictionnaire

L'extension d'un fichier solution est `.dict`. Il contient :

- Le nombre de compétences suivi de leurs noms.
- Le nombre de ressources suivi de leurs noms.
- Le nombre de parcours patients suivi de leurs noms.
- Le nombre de rendez-vous suivi de leurs noms.

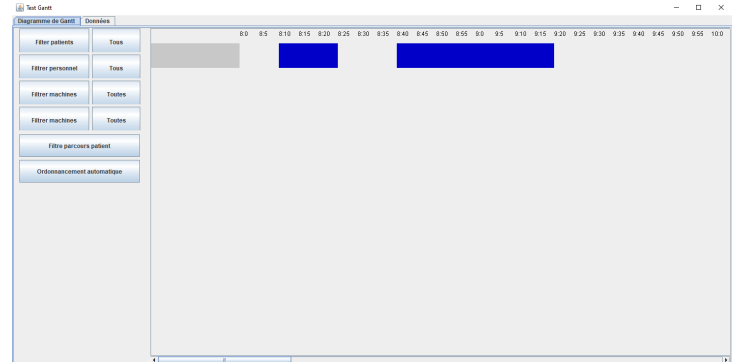


## Document d'installation

Le projet est exécutable sur l'IDE Eclipse sur une version supérieure à celle de Décembre 2019. Les librairies sont comprises dans le projet et le code source qui sera rendu. Le .jar exécutable est généré via Eclipse. Les librairies étant packagées avec le code, seul un JRE de version préférablement supérieure à 13 est nécessaire pour exécuter l'application.

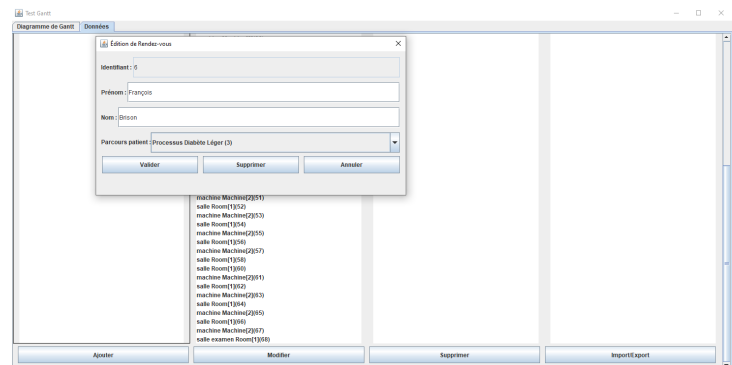
## Objectifs

- Visualisation de RDV de patients sur une journée
- Déplacement de ces RDV
- Modification des données



## Mise en œuvre

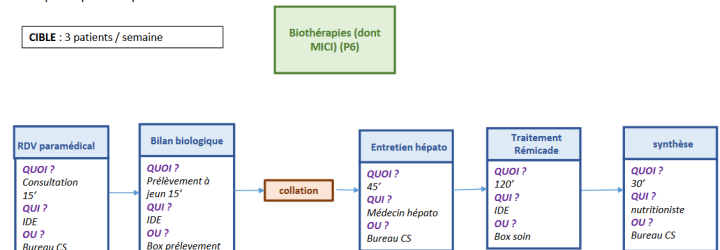
1. Application Java, développée avec Eclipse et Window Builder.
2. Ajout d'une base de données intégrée SQL via SQLite.
3. Mise en place d'une architecture MVC.



## Résultats attendus

Une application permettant de visualiser des Rendez-vous de patients sur une journée, de les déplacer et de les modifier via une interface graphique. Possibilité également de charger des calendriers, rendez-vous, patients, ... via des fichiers au format texte standardisés.

Exemple de parcours patient :





# Visualisation et gestion du parcours patient :

François BRISON

Encadrement : Yannick KERGOSIEN

## Objectifs

- Visualisation de RDV de patients sur une journée
- Déplacement de ces RDV
- Modification des données

## Mise en œuvre

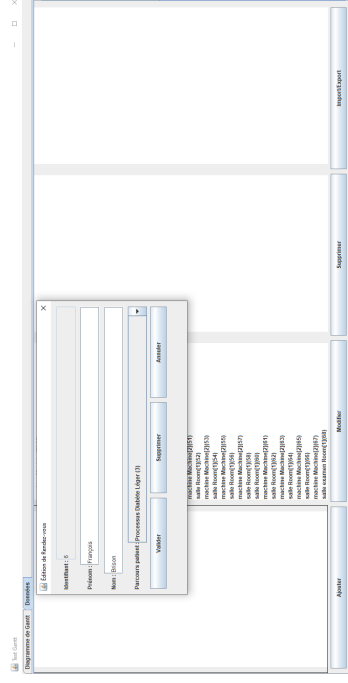
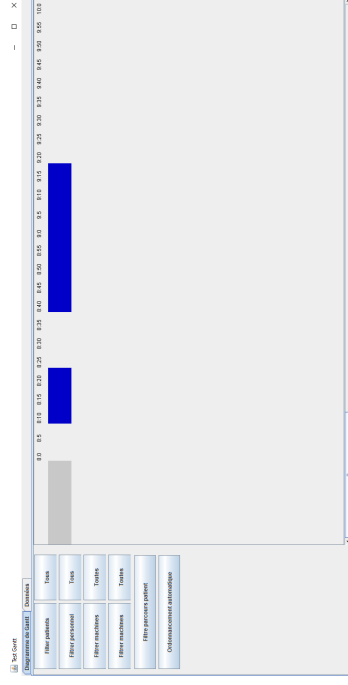
1. Application Java, développée avec Eclipse et Window Builder.
2. Ajout d'une base de données intégrée SQL via SQLite.
3. Mise en place d'une architecture MVC.



En collaboration avec Polytech

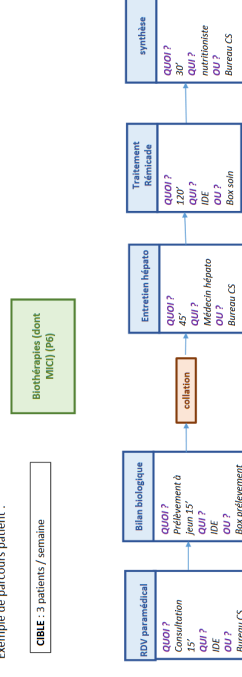
## Résultats attendus

Une application permettant de visualiser des Rendez-vous de patients sur une journée, de les déplacer et de les modifier via une interface graphique. Possibilité également de charger des calendriers, rendez-vous, patients, ... via des fichiers au format texte standardisés.



Exemple de parcours patient :

CIBLE : 3 patients / semaine



# Visualisation et gestion du parcours patient

## Résumé

Ce PRD a pour objectif la réalisation d'une Interface Homme-Machine permettant de visualiser et gérer des parcours patients sous la forme d'un diagramme de Gantt. De plus, il s'inscrit dans le cadre d'une thèse axée sur l'ordonnancement de parcours patients. Ainsi, cette application a notamment pour but de servir à la visualisation de ces résultats, ainsi qu'à l'implémentation potentielle en son sein d'un algorithme d'ordonnancement réalisé par le doctorant encadrant ce projet, Mr.Moulard Simon.

## Mots-clés

gantt, parcours patient, ordonnancement

## Abstract

The goal of this Project is to realize a Human-Machine Interface allowing the management and the visualisation of a Patient Process as a Gantt chart. Furthermore, this project is made in parallel of a thesis about the scheduling of appointments from patient processes. Thus, this application will help visualize its results, and may as well be used to directly implement the scheduling algorithm made by the PhD student supervizing this project, Mr.Moulard Simon.

## Keywords

gantt, patient process, scheduling

**Entreprise**

Polytech



**Tuteur entreprise**

Simon MOULARD

**Étudiant**

François BRISON (DI5)

**Tuteur académique**

Yannick KERGOSIEN