ISIS INGÉNIEURS
CASTRES

Rapport de projet :

SYSTÈME D'AIDE A LA DÉCISION DEVANT UN GOITRE

PRESENTÉ PAR :

Tina NANTENAINA
Agathe SUTARIK
Julie LEROY
Kawtar HAMDI

TABLE DES MATIÈRES

LISTE DES FIGURES

INTRODUCTION

Au cours notre deuxième année en cycle d'ingénieur à l'école d'ingénieur ISIS, nous avons l'opportunité de réaliser un projet tutoré afin de favoriser le développement de nos compétences d'élèves ingénieur en concevant et en développant une solution informatique et/ou une solution de systèmes d'information pour la santé. Nous avons choisi de créer un module logiciel d'aide au diagnostic devant une suspicion de goître. Celui-ci s'appuierait sur un logiciel de gestion de dossier patient déjà existant.

Pour mener à bien ce projet, le présent rapport doit être bien structuré pour pouvoir être exploité après la mise en place de l'application. Nous l'avons organisé de la manière suivante :

Un premier chapitre intitulé "étude préalable" vise à présenter le sujet et à faire l'analyse des besoins.

Une deuxième partie "élément d'ingénierie" qui consiste à présenter les outils et technologies qu'on a utilisés pour réaliser ce projet.

Le dernier chapitre portera sur "la gestion de projet"

I. ETUDE PREALABLE

I.1. Présentation du sujet

La politique d'informatisation des cabinets médicaux et la performance des logiciels métiers ont dynamisé un secteur d'activité en pleine mutation depuis quelques années. Les logiciels métiers sont devenus des outils de travail indispensables au bon fonctionnement d'un cabinet médical. Les éditeurs de logiciels destinés aux médecins proposent des fonctionnalités communes à d'autres professions, mais surtout des outils adaptés à la pratique médicale.

Notre projet intitulé "Système d'aide à la décision et diagnostic devant un goitre est de développer un système d'aide pour que chaque médecin puisse établir facilement et rapidement un diagnostic à partir des symptômes observés, les examens cliniques et paracliniques réalisés chez le patient. L'idée de notre application est de faciliter aux médecins la rédaction d'un bon bilan en se basant sur toutes les données renseignées et en répondant à des questions précises, qui après, grâce à des règles établies, sortira divers diagnostics qui seront classés du plus probable au moins probable. Notre objectif est aussi de réduire les taux d'errance de diagnostic ainsi que de gagner du temps lors de la prise en charge d'un patient, notamment dans le cas où le patient présenterait un goître.

I.2. Rappel des parties prenantes

Dans le cadre de ce projet, notre équipe représente à la fois la partie MOA (maîtrise d'ouvrage) et la partie MOE (maître d'œuvre). En effet, nous avons le rôle de client et de maître d'œuvre présentant le cahier des charges de notre projet. La personne supervisant notre travail, notre tuteur-école est BASTIDE Rémi. Dans notre équipe, nous avons donc réparti les rôles comme ci-dessous :

Chef de projet : Nantenaina Tina

Développeur Back-end: Sutarik Agathe et Nantenaina Tina

Développeur Front-end : Julie Leroy et Kawtar Hamdi

I.3. Analyse des besoins

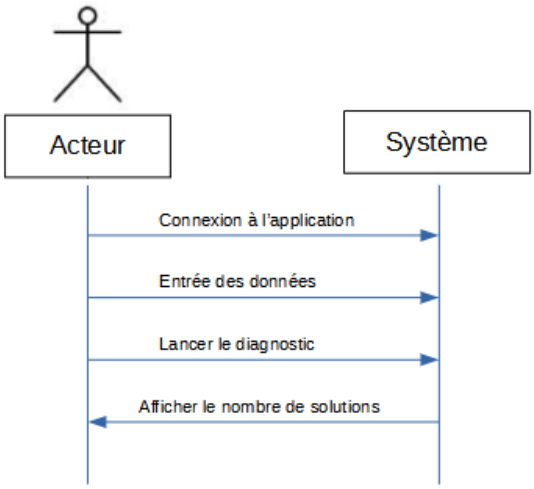


Figure 1. Analyse des besoins

Nous souhaitons élaborer plusieurs méthodes de résolution, qui permettra de disposer d'un outil permettant de comparer facilement les différents diagnostics. En particulier, nous avons développé une interface utilisateur graphique permettant de proposer et de choisir simplement le diagnostic le plus adapté au patient. Notre logiciel n'étant pas indépendant puisque c'est un module d'extension d'un autre logiciel de gestion des patients, le médecin n'aura pas besoin de se connecter, car il l'aura déjà fait sur l'autre logiciel. Le RPPS, le nom et le prénom du médecin seront transmis à notre logiciel. Un médecin souhaitant déterminer la cause d'un goître devra donc juste aller sur le module depuis son logiciel de gestion de patient.

Il pourra importer des anciens patients déjà existants dans la base de données du logiciel, ou bien ajouter des nouveaux patients. Ensuite il commencera à remplir les différentes grilles d'examens en fonction des informations données par le patient et des informations obtenues lors des examens cliniques réalisés sur le patient.

Notre outil permet également de stocker les informations et les données antérieures par exemple des clichés radiographiques ou d'autres examens déjà réalisés.

Une fois toutes ces informations réunies, le médecin pourra ainsi être aiguillé vers différents diagnostics avec chacun leur probabilité d'exactitude.

II. ELEMENTS D'INGENIERIE

II.1. Modèle conceptuel de données et déploiement des objets

Après avoir rencontré quelques problèmes lors de l'initialisation de notre base de données, nous avons décidé de modifier notre modèle. Nous avons illustré ce nouveau modèle dans l'annexe 1.

Les classes ajoutées par rapport au modèle précédent sont :

- **Valeur_examen** : classe créée pour avoir une liste de valeur dans la classe Examen, elle contient toutes les informations sur une valeur (exemple : dans un examen des signes généraux, il y a plusieurs valeurs, comme la tachycardie et la fièvre, on les renseigne donc individuellement dans cette classe). Cette classe contient les attributs suivants :
    - **nom_valeur** : le nom de la valeur
    - **est_valeur** : est-ce que la valeur est présente (exemple : si pour la valeur fièvre, est_valeur=true, alors cela signifie que le patient a de la fièvre)
    - **valeur** : la valeur, cela peut être une observation, un nombre, ou bien être laissé vide

- **Symptome** : classe permettant d'avoir une liste de symptômes dans l'anamnèse du patient. Représente un symptôme du patient et a comme attribut :
    - **nom** : représente le nom du symptôme.

- **Valeur_signe_compression** : c'est une classe qui nous renseigne si l'affection du patient est au stade de complication, qui se traduit par des signes de compression. On aura alors une liste de signes de compression, avec les attributs suivants :
    - **nom_valeur_signe compression** : le nom du signe de compression

- **est_valeur_sc** ; renvoie un boolean si le signe de compression est présent ou pas.

Les classes modifiées sont:

- **Examen** : les valeurs de l'examen sont maintenant stockées en dehors dans la classe Valeur_Examen.

- **SigneFonctionnel** : auparavant, la classe comprenait les symptômes, mais nous les avons séparés. En effet, les signes fonctionnels représentent la principale plainte du patient, tandis que les symptômes sont les signes accompagnateurs, ils sont donc différents. Par conséquent, l'anamnèse comprend maintenant une liste de signes fonctionnels et une liste de symptômes.

- AtcdPersonnelMedical : c'est une classe qui comprend les antécédents personnels médicaux utilisés dans le diagnostic. On lui a rajouté un attribut valeur_atcd_perso qui permet de préciser la valeur de l'antécédent, si nécessaire.

Après avoir effectué l'étude et la conception de notre application, nous passons à la phase d'implémentation. La partie suivante va présenter le résultat du travail effectué durant ce projet.

II.2. Environnement matériel

Pour développer notre application, nous avons utilisé les matériels suivants :

- Un ordinateur portable ASUS (système d'exploitation Windows 10, processus Intel core i5, 8 Go de RAM)
- Un ordinateur portable MSI (système d'exploitation Windows 10, processus Intel core i7, 8 Go de RAM).

Après avoir présenté les moyens matériels mis à notre disposition que nous avons utilisés dans le cadre de ce projet, nous abordons par la suite, les logiciels utilisés pour la réalisation de ce projet.

II.3. Présentation des outils utilisés

Magic draw :

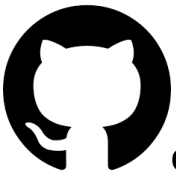
MagicDraw est un outil de modélisation visuel UML , SysML , BPMN et UPDM avec prise en charge de la collaboration en équipe. Conçu pour les analystes métier, les analystes logiciels, les programmeurs et les ingénieurs QA, cet outil de développement dynamique et polyvalent facilite l'analyse et la conception de systèmes et de bases de données orientés objet. Nous avons donc utilisé cet outil afin d'établir nos modèles de données sur lesquels nous avons construit notre application (1).

NetBeans :

NetBeans est un environnement de développement intégré (EDI). En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur avec coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web), c'est donc le choix qui nous a semblé être le plus approprié et le plus efficace pour travailler en groupe sur plusieurs aspects de notre projet en même temps grâce à sa diversité de langage. En effet, nous avons pu y développer le front-end comme le back-end. Pour développer, nous avons utilisé la version 12.1 (2).

GanttProject :

GanttProject est un logiciel de gestion de projet dont la principale fonctionnalité est la création de plannings à l'aide de diagrammes de Gantt. Il permet d'améliorer l'organisation d'une entreprise et d'assigner des tâches spécifiques à chaque personne participant à un projet. C'est à partir de celui-ci que nous avons établi notre trame de travail et que nous avons pu suivre l'évolution de notre projet (3).



GitHub :

GitHub est un service web d'hebergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de version Git. Dans le cadre de notre projet, nous avons utilisé cet outil tout au long du développement afin que chaque membre puisse avoir accès au code pour le modifier et partager ses modifications. En effet, de par son aspect collaboratif et dynamique, il est très efficace pour la gestion de projet informatique (4).

II.4. Les technologies utilisées

❖ BACK-END

Git :

C'est un logiciel de gestion de versions décentralisé. Le code informatique développé est stocké non seulement sur l'ordinateur de chaque contributeur du projet, mais il peut également l'être sur un serveur dédié (5).

Java Enterprise Edition J2EE:

La technologie Java Enterprise Edition J2EE, avec le *framework* Spring MVC qui permet de s'affranchir de la gestion des servlets. Nous avons également effectué de la gestion directe des bases de données (couche DAO, requêtes SQL brutes) en utilisant un *framework* de persistance des objets : Hibernate (6).

Spring Boot :

SpringBoot est un module récent de l'écosystème Spring. Il permet de simplifier le développement en Spring (configuration maven, programme principal, etc.) Nous l'avons utilisé car il permet de s'affranchir des problèmes de dépendances et d'automatiser la création des fichiers .jar (7).

Thymeleaf

Thymeleaf est un *template engine* (moteur de rendu de document) écrit en Java. Principalement conçu pour produire des vues Web, il fournit un support pour la génération de documents HTML, XHTML, JavaScript et CSS pour gérer la communication entre le *front* et le *back*. Celui-ci va permettre d'afficher à l'utilisateur des informations venant de la partie *back office*, comme des valeurs persistées en base de données par exemple (8).

Drools :

C'est une solution de système de gestion des règles métier (BRMS). Grâce à ce cadre, les utilisateurs définissent des règles qui spécifient l'action à entreprendre lorsqu'une condition particulière est remplie. Drools étend et met en œuvre l'algorithme de correspondance des motifs Rete. Dans les projets, il est généralement

utilisé pour définir des règles métier. Les règles métier sont composées de faits et de déclarations conditionnelles (9).

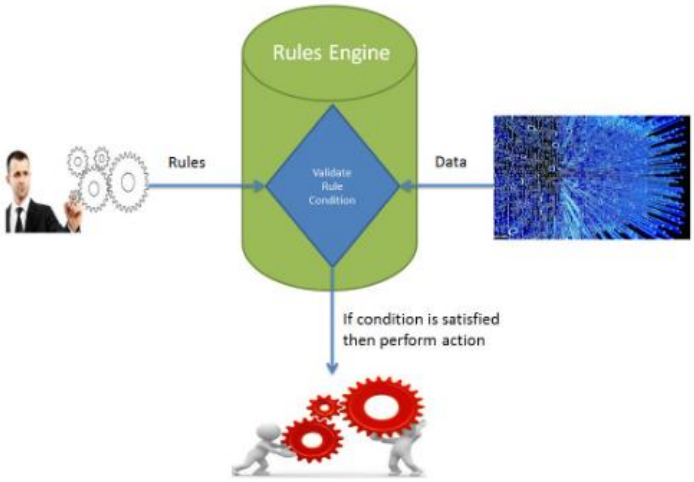


Figure 2. Présentation du drools

Un moteur de règles peut être décrit comme un système expert de domaine qui applique les règles commerciales aux données d'entrée données et si la condition de la règle est satisfaite, l'action correspondante est exécutée.

Les avantages de l'utilisation de Jboss Drools est que c'est un langage déclaratif. C'est-à-dire que toutes les tâches à réaliser sont décrites en petites étapes. Les règles sont écrites séparément dans le fichier **drl**. Il n'y a donc pas de couplage étroit entre les règles métier et le code/données.

Son implémentation dans notre projet :

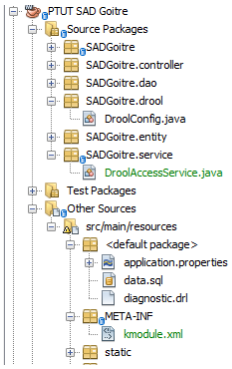


Figure 3. Implémentation du drools

❖ FRONT-END

Pour réaliser les différentes interfaces, un ensemble de pages dédiées aux patients et aux examens cliniques et paracliniques ont été réalisées en HTML5/CSS3 et contiennent des scripts JavaScript, intégrés directement dans les pages ou au sein de scripts externes. Les 2 langages principaux que nous avons utilisés sont complémentaires, ils sont décrit ci-dessous

HTML (Hypertext Markup Language)

C'est un langage conçu pour représenter les pages web. Ce langage de balisage permet de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images et des formulaires de saisie.HTML5 est la dernière évolution de la norme définissant le langage HTML (10).

CSS (Cascading Style Sheets)

Le CSS est un langage informatique permettant d'ajouter des contraintes de mise en forme graphique dans des documents web, dont les normes sont établies par le *World Wide Web Consortium* (W3C) (11).

II.5. Architecture logiciel

a) Interface graphique

Notre application permet de poser un diagnostic global à la fin en s'appuyant sur les différentes pages à remplir :

- **La page d'accueil** présente une interface simple permettant de lancer la consultation en appuyant sur le bouton Lancer La Consultation. On a essayé de se baser sur la simplicité pour notre page d'accueil afin de ne pas avoir beaucoup d'informations sur la première page.

- **Page patient** : La page patient contient un tableau regroupant tous les patients avec leur nom, prénom, âge, genre, date de naissance, symptômes (si le patient est déjà venu, un résumé apparaît ?) et le médecin qui lui est attribué. Sur cette page, il est également possible d'ajouter un patient, voir les examens déjà réalisés, et ajouter des nouveaux examens.

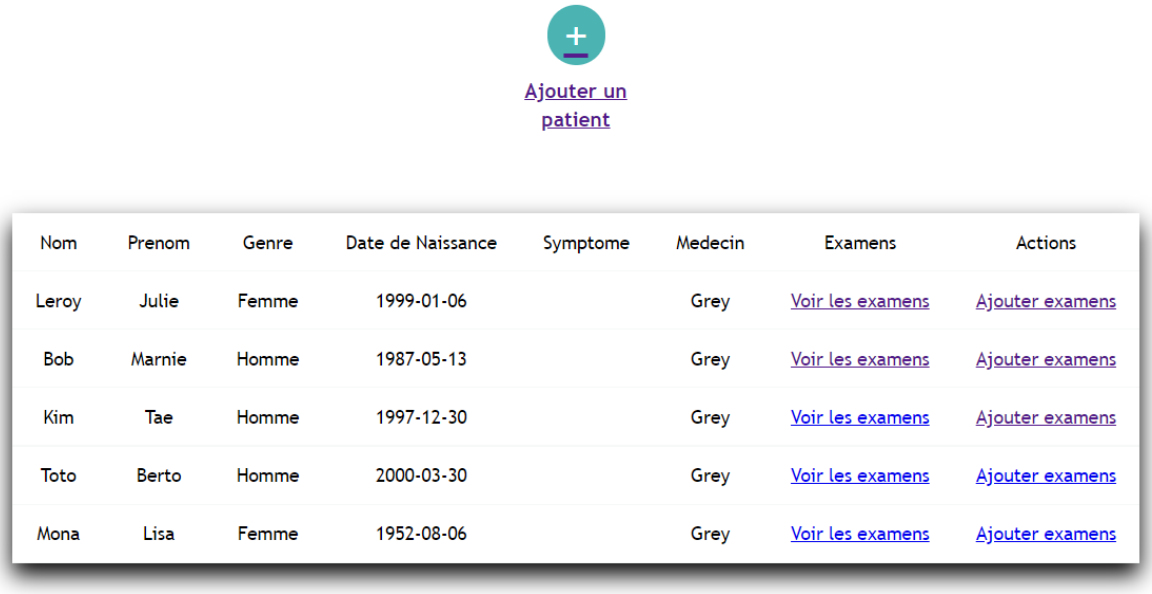


Figure 4. Image de la page patient

- **La page Examen** se divise en 2 catégories, les examens cliniques, composés des formulaires pour renseigner les signes généraux, l'examen région cervicale, l'examen ophtalmologiques, l'examen neurologique, l'examen aire ganglionnaire ; puis les examens paracliniques composés de plusieurs pages de formulaire sur les examens morphologiques, biologiques et fonctionnels. Ces formules sont composées de case à cocher et de zone de texte où le médecin pourra sélectionner les paramètres qu'il trouve pendant son examen et il pourra aussi entrer des commentaires. Les deux images suivantes montrent la navigation pour accéder à ces examens. Cette navigation a été conçu pour faciliter au médecin de retrouver quel examen il veut remplir.

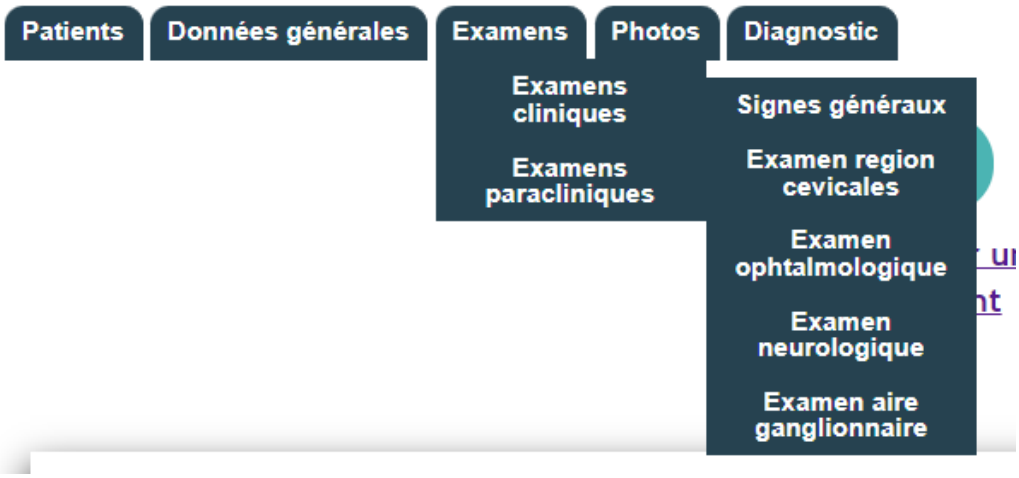


Figure 5. Onglet examens cliniques

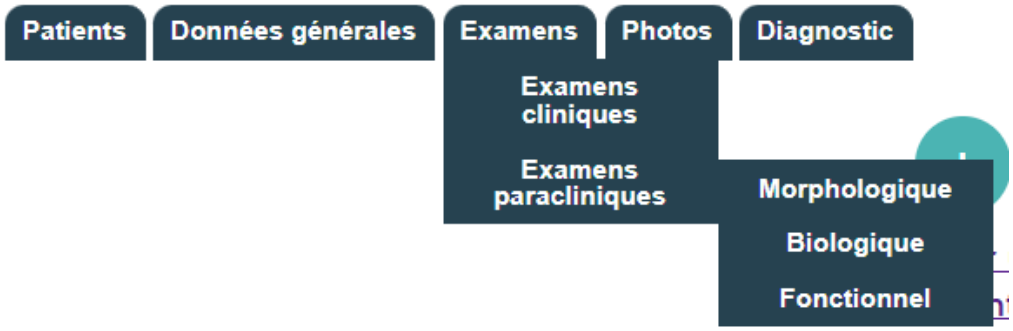


Figure 6. Onglet examens paracliniques

- **La page données générales** permet de renseigner les circonstances de découverte qui est composée par les antécédents du patient, qu'ils soient médicaux, familiaux, chirurgicaux, mais également les traitements en cours ainsi que les allergies. Cette page renseigne donc l'anamnèse du patient.

- **La page Photos** : Sur cette page, le médecin pourra retrouver tous les clichés radiographiques ou photos de lésion ajouter dans le dossier du patient. Il pourra y renseigner ses observations et y faire un compte rendu pour chaque photo. L'idée d'avoir cette page est de pouvoir tous recueillir dans un seul endroit les informations sur le patient, ainsi de pouvoir les exporter plus tard sous format pdf un compte rendu complet avec les examens cliniques et les imageries médicales. Cependant, au stade de notre projet, cette page n'est pas encore fonctionnelle.

- **La page Diagnostic** : c'est la page finale de notre application, elle propose les différents diagnostics orientant du plus probable au moins probable la cause du goitre. Ces résultats seront associés à des pourcentages afin de guider au mieux le médecin, mais aussi une description détaillée en fonction des observations entrées par rapport au patient.

b) Interface console

➢ Racine du projet

Dossier Files : c'est dans ce dossier que sont enregistrés les fichiers téléversés sur le serveur

application.properties : fichier de configuration de l'application. On peut y trouver par exemple le nom de l'utilisateur et le mot de passe par défaut

pom.xml : c'est le fichier utilisé par Maven dans lequel se trouvent les dépendances du projet.

➢ Dossier source

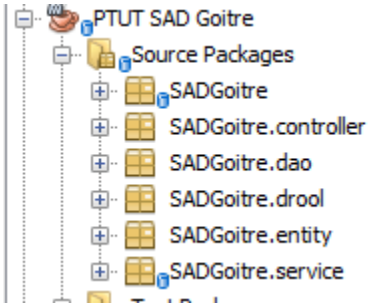


Figure 7. Les dossiers sources

- **WebApp** : classe contenant une méthode d'initialisation de l'application. Contient aussi le programme principal (main) démarrant l'application Spring Boot

- **SADGoitre.controller** : dossier contenant les classes qui viennent faire correspondre toutes requêtes de l'utilisateur à des méthodes

- **SADGoitre.dao** : dossier contenant les repository de la classe correspondante

- **SADGoitre.entity** : dossier contenant toutes les entités.

- **SADGoitre.service :** dossier contenant les services.

➢ Les ressources

- **static** : les fichiers qui ne sont pas concernés par du traitement opéré par un *framework* (Thymeleaf par exemple)

- **templates** : dossier utilisé par Thymeleaf pour générer les fichiers HTML à envoyer au client après le traitement du modèle (*template*) concerné

- **data.sql** : permet d'exécuter des instructions PSQL au démarrage de l'application

- La base de données

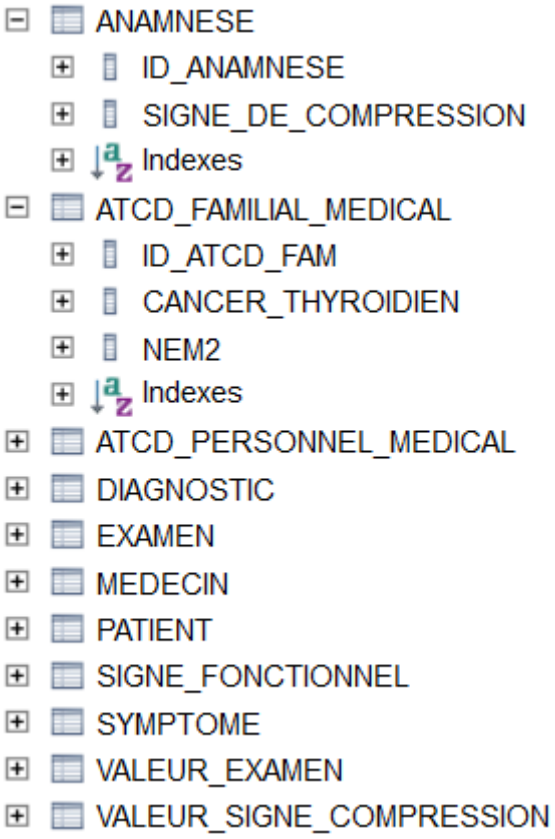


Figure 8. Vision de la base de données H2 avec Spring Boot

On voit dans notre base de données toutes les classes figurant dans le modèle de données. La seule différence au niveau des données étant les identifiants ajoutés à chaque classe par souci de respect des contraintes de clé primaire. Nous avons représenté chaque table par des entity dans le code.

Cette base de données contient quelques valeurs renseignées dans le fichier data.sql vu plus haut.

III. GESTION DE PROJET

III.1. L'équipe

Notre équipe de projet est composé de personnalités complémentaires avec des antécédents différents mais nous sommes dans le même cursus du cycle ingénieur en informatique spécialisé en e-santé.

Notre chef de projet qui est à la fois médecin nous a beaucoup été d'une grande aide dans l'accomplissement du projet avec ses connaissances du métier médical.

Pour mieux illustrer les différents membres de l'équipe ainsi que leurs taches attribués et effectués durant le projet, nous avons fait le diagramme suivant :

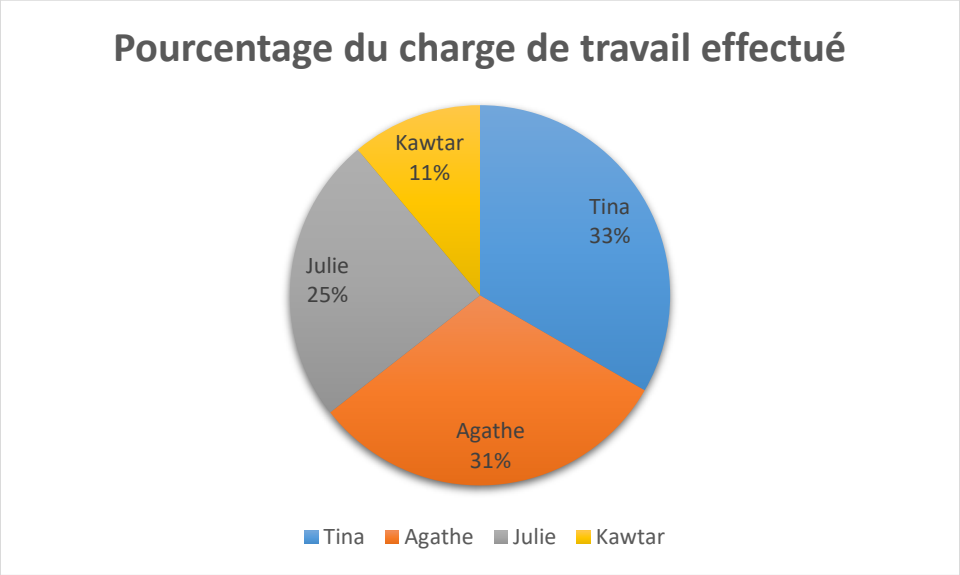


Figure 9. Diagramme charge de travail effectué

III.2. Comparaison du Prévu et du réaliser

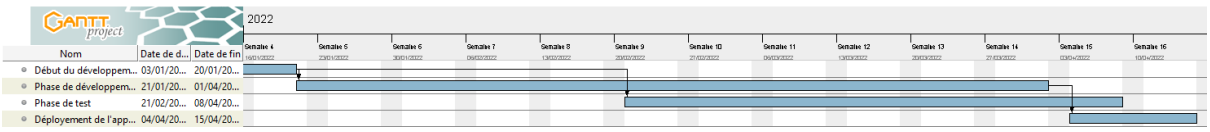


Figure 10. Diagramme de Gantt

L'image 2 présente les différentes activités à faire durant le deuxième semestre en spécifiant les références de conduite du projet pour ne pas oublier les étapes les plus importantes à réaliser dans les bons délais.

Ce diagramme de Gantt est un outil pratique afin de planifier nos différentes phases à accomplir, grâce à une vue d'ensemble des tâches planifiées, chaque personne concernée sait quelle activité doit être effectuée et à quelle date précise.

Sur l'image ci-dessus on a défini 4 principaux aspects qu'on devrait terminer, notre premier échelon étant de débuter le semestre par la partie développement sur le logiciel Netbeans, car c'est la tâche la plus importante ; son succès est lié au prochain aspect de notre projet : le test. Pour finir, on conclut sur le déploiement de notre application finale.

III.3. Problèmes rencontrés

Concernant la partie FrontEnd on n'a pas eu des problèmes que ce soit par rapport à la phase HTML et ou bien la phase CSS.

Concernant le Backend la majorité des problèmes ont été rencontrés au niveau de la modélisation. En effet nous avons travaillé avec une base de données déjà existante, or certaines données ne peuvent être présentées que sous la forme d'une liste, ce qui entre en contradiction avec notre implémentation. Pour résoudre ce problème, nous avons dû changer la modélisation de notre projet, en ajoutant plusieurs classes : Valeur_examen, Symptome et Valeur_signe_compression.

De plus, faire des formulaires contenant des valeurs multiples s'est avéré compliqué en Java Spring Boot et Thymeleaf : d'une part nous devions faire des formulaires pour chaque examen, mais chaque examen contenait sa propre liste de valeurs d'examens avec des noms définis. Nous les avons donc définis dans le contrôleur associé. Ensuite, les valeurs remplies devaient être envoyés (toujours avec le même nom mais dont les autres propriétés auront changé. Il a donc fallu d'abord sauvegarder dans le contrôleur chaque valeur renseignée par l'utilisateur, afin d'enfin pouvoir enregistrer l'examen avec les valeurs mises à jour.

```java
/**
 * Montre le formulaire permettant d'ajouter un examen de signes generau
 *
 * @param model pour transmettre les informations à la vue
 * @param idPatient L'id du patient
 * @return le nom de la vue à afficher ('formulaireExamen.html')
 */
@GetMapping(path = "addSignesGeneraux")
public String ajoutSignesGeneraux(Model model, int idPatient) {
    List<Valeur_examen> val_examens;
    val_examens = Arrays.asList(
            new Valeur_examen("Fièvre"),
            new Valeur_examen("Prise de poids"),
            new Valeur_examen("Perte de poids"),
            new Valeur_examen("Tachycardie"),
            new Valeur_examen("Bradychardie"),
            new Valeur_examen("Hypertension artérielle")
    );
    /*for (Valeur_examen v : val_examens) {
        System.out.println(" here" + v.getId_valeur_examen());
    }*/
    Examen exam = new Examen();
    exam.setValeur_examen(val_examens);
    exam.setPatient_examen(daoPatient.getOne(idPatient));
    exam.setEst_examen_clinique(true);
    exam.setDate_examen(LocalDate.now());
    exam.setEst_examen_clinique(true);
    exam.setNom_examen("Signes généraux");
    model.addAttribute("examen", exam);
    model.addAttribute("patient", daoPatient.getOne(idPatient));
    return "formulaireExamenSignesGeneraux";
}
```

Figure 11. Exemple de contrôleur avec un ajout de valeurs en brut (le nom des valeurs)

III.4. Point à améliorer

CONCLUSION ET PERSPECTIVE

L'objectif de notre projet tutoré était de créer une application web d'aide à la décision et diagnostic devant un goitre pour minimiser les errances diagnostiques et le temps de prise en charge. Cependant, cette application ne remplacera pas l'intelligence et le savoir du médecin, mais ce sera juste un outil pour l'aider dans son travail. Ce projet nous a alors permis de faire un premier pas dans les systèmes d'aide à la décision. Nous avons pu définir toutes les informations nécessaires pour établir les règles de décision.

Plusieurs technologies ont été nécessaires pour la réalisation de notre projet, on citera le langage java qui permet de développer des applications client-serveur, le Drools qui nous a permis d'établir les règles pour faire sortir les diagnostics.

Après le passage par les différentes étapes de développement, l'application a abouti à un logiciel quasi fonctionnel qui répond globalement aux critères imposés.

Le présent travail nous a permis d'acquérir des connaissances, de maîtriser et de se perfectionner dans le domaine de programmation, ainsi que de consolider nos connaissances en conception logicielle.

Comme perspective à ce travail, nous proposons d'approfondir le développement de l'application et de réfléchir à l'éventualité de créer des fonctions supplémentaires (traitement de texte, agenda, connexion avec une application sur le téléphone portable) ou de créer différentes extensions pour d'autre maladie par exemple

REFERENCES WEBOGRAPHIQUES

(1). Récupéré sur https://en.wikipedia.org/wiki/MagicDraw

(2). Récupéré sur https://www.techno-science.net/glossaire-definition/NetBeans.html

(3). Récupéré sur https://ganttproject.fr.softonic.com/

(4). Récupéré sur https://github.com/

(5). Récupéré sur https://git-scm.com/

(6). Récupéré sur https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm

(7). Récupéré sur https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started.html#getting-started.first-application

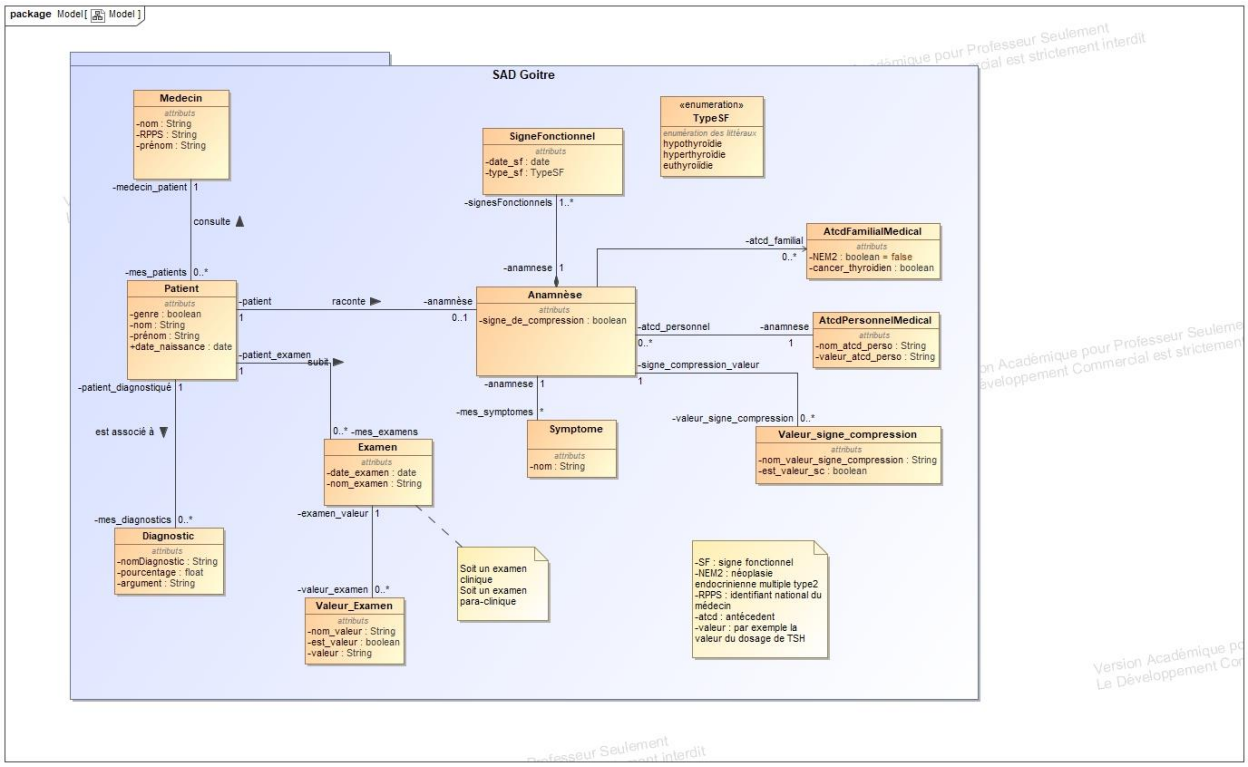
(8). Récupéré sur https://gayerie.dev/docs/spring/spring/thymeleaf.html

(9). Récupéré sur https://www.drools.org/

(10). Récupéré sur https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5,

(11). Récupéré sur https://yesyouweb.com/css3-guide-reference/

ANNEXES


package Model[Model]
SAD Goitre
Medecin
attributs
-nom : String
-RPPS : String
-prénom : String
-medecin_patient 1
consulte
-mes_patients 0..*
Patient
attributs
-genre : boolean
-nom : String
-prénom : String
+date_naissance : date
-patient
raconte
-anamnèse
1
0..1
-patient_examen
subit
1
-patient_diagnostiqué 1
est associé à
-mes_diagnostics 0..*
Diagnostic
attributs
-nomDiagnostic : String
-pourcentage : float
-argument : String
0..* -mes_examens
Examen
attributs
-date_examen : date
-nom_examen : String
-examen_valeur 1
-valeur_examen 0..*
Valeur_Examen
attributs
-nom_valeur : String
-est_valeur : boolean
-valeur : String
Soit un examen clinique
Soit un examen para-clinique
SigneFonctionnel
attributs
-date_sf : date
-type_sf : TypeSF
-signesFonctionnels 1..*
«enumeration»
TypeSF
énumération des littéraux
hypothyroïdie
hyperthyroïdie
euthyroïdie
-anamnese 1
Anamnèse
attributs
-signe_de_compression : boolean
-atcd_familial
0..*
AtcdFamilialMedical
attributs
-NEM2 : boolean = false
-cancer_thyroidien : boolean
-atcd_personnel
0..*
-anamnese
1
AtcdPersonnelMedical
attributs
-nom_atcd_perso : String
-valeur_atcd_perso : String
-anamnese 1
-mes_symptomes *
Symptome
attributs
-nom : String
-signe_compression_valeur
1
-valeur_signe_compression 0..*
Valeur_signe_compression
attributs
-nom_valeur_signe_compression : String
-est_valeur_sc : boolean
-SF : signe fonctionnel
-NEM2 : néoplasie endocrinienne multiple type2
-RPPS : identifiant national du médecin
-atcd : antécedent
-valeur : par exemple la valeur du dosage de TSH