**UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE**

**FACULTÉ D’INFORMATIQUE**



Ingénieur Informatique 2ème année

Module : Programmation Orientée Objet

**Rapport du Projet Gestion Cabinet Médicale**

Réalisé par :

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Prénom | Matricule | Groupe |
| Miloudi | Yacine | 232331406303 | 2 |
| HATTAB | Hamza Riadh | 232331394516 | 1 |
| Hamroun | Sami abdelmalek | 232331406007 | 2 |

**1. Introduction :**

* 1. **Présentation du problème :**

La tenue des dossiers des patients, l’établissement de rendez-vous et la délivrance d’ordonnances dans les établissements de santé sont des processus manuels qui sont sujets à des inefficacités ainsi qu’à des erreurs et sont très fastidieux. Ces problèmes peuvent nuire à la qualité des soins prodigués au patient ainsi qu’à l’efficacité générale du fonctionnement de l’établissement de santé.

* 1. **Objectif principal :**

Le but premier de ce projet est de construire un système d’application puissant et facile à utiliser qui gérera la plupart des fonctions courantes d’une pratique médicale. Ces fonctions comprennent, sans s’y limiter, la gestion des dossiers des patients, l’accueil des patients et la gestion du profil des patients tout en encourageant la saisie rapide et facile de documents médicaux.

Un sondage réalisé en 2023 par l’Association Nationale des Médecins Généralistes (ANMG) a montré que 65 % des praticiens dans les cabinets médicaux trouvent que la gestion des dossiers patients et des rendez-vous représente une charge administrative excessive, réduisant le temps consacré aux consultations médicales.

De plus, 78 % des répondants ont signalé que l’absence d’un système centralisé pour gérer les informations médicales augmentait le risque d’erreurs administratives, entraînant parfois des retards dans les traitements.

**1.3 Avantages :**

• Amélioration de l’efficacité : diminue la saisie manuelle des données, réduit le temps consacré aux processus administratifs. Par exemple, une infirmière qui utilise ce système de gestion pour trier les dossiers des patients et qui a déjà passé des heures à faire ce travail peut maintenant le faire en quelques minutes, se concentrant ainsi sur d’autres tâches plus importantes. En d’autres termes, le nombre d’efforts que l’infirmière aurait dû déployer pour produire les dossiers des patients est considérablement réduit

• Amélioration de la précision et de l’intégrité des données : le stockage et la récupération précise et fiables des informations médicales du patient sont effectués, ce qui réduit les risques d’erreurs et d’incohérences.  Par exemple, les renseignements sur le patient sont utilisés pour remplir les ordonnances dont les calculs ont déjà été effectués ainsi que les médicaments que les patients ont été invités à prendre.

• Diminution de la charge de travail administratif : il est prévu que la charge de travail du personnel médical soit réduite au minimum afin qu’une plus grande attention puisse être accordée aux patients et à d’autres tâches pertinentes. Par exemple, un médecin ne serait plus dérangé par des processus de mise à jour des dossiers inutiles, mais pourrait plutôt se concentrer sur les consultations.

Il y a aussi une amélioration de l’efficacité dans la mise à jour des renseignements existants et dans l’entrée de nouveaux renseignements dans les dossiers médicaux des patients

•Amélioration de la qualité des soins : facilite la communication entre les professionnels de la santé et les patients, augmente l’efficacité des rendez-vous et assure l'accessibilité et l'organisation des dossiers médicaux des patients. Un exemple concret serait un patient ayant un historique médical complexe ; les médecins pourraient rapidement accéder à des données actualisées et complètes, assurant ainsi un diagnostic plus rapide et un traitement adapté.

**2. Exigences du système :**

**2.1 Exigences fonctionnelles :**

•  Gestion des dossiers médicaux :

o Création la mise à jour et la consultation des dossiers .

o Rechercher les dossiers de patients en fonction de plusieurs paramètres, notamment le nom, l’identifiant et les informations de diagnostic.

• Gestion des rendez-vous :

* Planifier, reporter et annuler les rendez-vous de manière efficace.
* Gérer la disponibilité des médecins et les rendez-vous des patients.

o Envoyer des rappels pour les rendez-vous automatiquement en utilisant l’ email ou SMS.

• Gestion du profil patient :

o Mettre à jour et stocker le profil complet du patient ainsi que les coordonnées personnelles et les détails de contact couvrant l’historique médical.

• Génération de documents :

o Générer auto les certificats médicaux et les prescriptions .

**2.2 Exigences non fonctionnelles :**

• Interface utilisateur : Une interface simple et facile à comprendre devrait être disponible pour tous les rôles d’utilisateurs (médecins, administrateurs, réceptionnistes).

• Indépendance de la plateforme : L’application doit être conçue pour s’étendre à plusieurs systèmes d’exploitation ( Java et ses multi-plateformes).

• Sécurité des données : Des mesures de sécurité supplémentaires doivent être prises pour empêcher l’accès sans autorisation, les abus ou les fuites de données sensibles sur le patient.

• Performance : Le système doit être suffisamment réactif et performant pour prendre facilement en charge de nombreux ensembles de données et utilisateurs simultanés.

•Veiller à ce que les mises à jour et les améliorations futures soient facilitées dans le but de structurer, documenter et maintenir facilement un code.

**3. Conception :**

Cas d’utilisation : Les tâches effectuées par différentes classes d’utilisateurs sur le système sont appelées cas d’utilisation.

•           Médecin :

o           Prendre rendez-vous

o           Vérifier l’historique du patient

o           Fournit des notes d’ajustement

o           Modifie les détails du patient

•           Administrateur :

o Superviser les comptes du médecin

o Contrôler les comptes des utilisateurs

o Configuration et administration du système

**3.1 Diagramme de Cas d'Utilisation:**

* **UML Diagrams:**
* **Use Case Diagram:**

**3.2 Diagramme de Classes:**

* **class:**

**1.Class Docteur :**

Cette classe représente le profil d’un docteur dans le système. Elle contient les informations personnelles telles que le **nom**, le **prénom** et la **spécialité** du docteur. Elle inclut également une méthode appelée **Créer Prescription**, permettant de générer des prescriptions médicales pour les patients.

**2. Classe Patient**

Cette classe gère les informations relatives aux patients du cabinet médical, telles que leur **âge**, leur **sexe**, et leur **adresse**. Elle hérite de la classe **Person**, qui contient des informations générales comme le **nom**, le **prénom**, et le **numéro de téléphone**.

**3. Classe Ordonnance**

Cette classe gère les informations liées à une ordonnance médicale. Elle inclut la **date** de création de l’ordonnance, le **docteur** qui l’a rédigée, et le **patient** concerné. Les **médications** et **instructions** sont représentées sous forme de listes de chaînes de caractères, car une ordonnance peut contenir plusieurs médicaments et instructions pour le patient.

**4. Classe rendez-vous**

Cette classe représente un rendez-vous médical. Elle inclut les informations principales telles que le **patient** concerné, le **docteur** responsable, la **date** et l’**heure** du rendez-vous , plus un string **summary**

**5. Classe Dossier Médical**

Cette classe gère le dossier médical d’un patient. Elle contient des informations telles que le **patient** et le **docteur** associés au dossier. De plus, un **ArrayList<SuiviMedicale>** est utilisé pour stocker les suivis médicaux liés au patient, permettant d’ajouter plusieurs entrées au fil du temps. La méthode **afficherDossierMedicale()** permet d’afficher les détails du dossier médical, incluant les suivis et les informations essentielles.

**6. Classe suivi Médical**

Cette classe contient les informations liées à un suivi médical : la **date** et l’**heure** du suivi, le **patient** concerné, l’**ordonnance** associée, ainsi que les **poids**, **taille** et **observations médicales**. Un résumé du traitement (**resumeDetraitement**) est également fourni.

**Class de Management et gestionment :**

**1. Classe Patient Management**

Cette classe gère une liste de **patients** sous forme d’un **ArrayList<Patient>**. Elle permet de créer, rechercher, afficher, ajouter, modifier et supprimer des patients. Les méthodes **creerPatient()**, **recherchePatient(String nom)**, **afficherPatient()**, **add()**, **edit()**, et **remove()** permettent de gérer l’ensemble des actions liées aux patients dans le système.

**2. Classe Gestion Patient**

Cette classe permet de gérer les actions liées aux **patients**, notamment la **modification** des informations, la **suppression** d’un patient, et la gestion globale de tous les patients. Les méthodes **ModifierPatient()**, **SupprimerPatient()**, et **Gestion\_Patients()** sont utilisées pour effectuer ces tâches.

**3. Classe Docteur Management**

Cette classe gère une liste de **docteurs** sous forme d’un **ArrayList<Docteur>**. Elle permet de créer, afficher, ajouter, modifier et supprimer des docteurs. Les méthodes **creerDocteur()**, **afficherDocteur()**, **add()**, **edit()**, et **remove()** sont utilisées pour gérer les informations des docteurs dans le système.

**4. Classe Gestion Docteur**

Cette classe permet de gérer les actions liées aux **docteurs**, notamment la **suppression** d'un médecin. La méthode **supprimer\_Medecin()** permet de supprimer un docteur de la base de données.

**5. Classe ordonnance Management**

Cette classe gère une liste d'**ordonnances** sous forme d’un **ArrayList<Ordonnance>**. Elle permet de créer, ajouter, modifier et supprimer des ordonnances. Les méthodes **creerOrdonnance()**, **add()**, **edit()**, et **remove()** sont utilisées pour gérer les ordonnances dans le système.

**6. Classe Gestion ordonnance**

Cette classe permet de gérer les actions liées aux **ordonnances**, notamment la **modification**, la **suppression**, et la gestion globale des ordonnances. Les méthodes **ModifierOrdonnance()**, **SupprimerOrdonnance()**, et **Gestion\_Ordonnance()** sont utilisées pour effectuer ces tâches.

**7. Classe Rendez-vous Management**

Cette classe gère une liste de **rendez-vous** sous forme d’un **ArrayList<Rendez\_vous>**. Elle permet de créer, ajouter, modifier et supprimer des rendez-vous. Les méthodes **creerRendezvous()**, **add()**, **edit()**, et **remove()** sont utilisées pour gérer les rendez-vous dans le système.

**8. Classe Gestion Rendez-vous**

Cette classe permet de gérer les actions liées aux **rendez-vous**, notamment la **modification**, la **suppression**, et la gestion globale des rendez-vous. Les méthodes **ModifierRendez\_vous()**, **SupprimerRendez\_vous()**, et **Gestion\_Rendez\_vous()** sont utilisées pour effectuer ces tâches.

**9. Classe Gestion Dossier Medical**

Cette classe est dédiée à la gestion des **dossiers médicaux**, en coordonnant les différentes entités telles que les **docteurs**, les **patients**, et les **ordonnances**. La méthode **Gestion\_Dossier()** centralise les opérations nécessaires pour maintenir et organiser les dossiers médicaux..

**10. Classe Dossier Management**

Cette classe gère une liste de **dossiers médicaux** sous forme d’un **ArrayList<DossierMedical>**. Elle permet de créer, mettre à jour, rechercher, ajouter, modifier et supprimer des dossiers. Les méthodes **creeNouveauDossier()**, **miseAjourSuivi()**, **rechercheDossier()**, **add()**, **edit()**, et **remove()** facilitent la gestion des dossiers médicaux.

**Main\_Function :**

**L**'application Main utilise des méthodes et des functionallitées cité dans ce driagram :

**4.Développement :**

**4.1 Technologies utilisées :**

• Langage de développement logiciel : Java

o Justification:

Indépendance de la plateforme : L’aspect le plus important du bricolage à travers Java est sa capacité à être universel, supporté par différents systèmes d’exploitation.

Un écosystème énorme : Il existe une riche source de bibliothèques de composants Java et beaucoup de développeurs Java, ce qui signifie que le produit final est rapidement développé et pris en charge à l’avenir.

**5. Caractéristiques :**

**5.1 Gestion des dossiers médicaux :**

• Création : Entrez les détails des nouveaux patients en plus de leurs données médicales historiques.

• Modification : modifier certaines informations qui peuvent avoir changé sur les patients dans leurs antécédents médicaux.

Consultation : le dossier des patients couvre l’histoire du médicament, le diagnostic fait et le traitement médical donné.

• Historique des consultations : Consulter et mettre à jour l’historique des consultations et des rendez-vous médicaux.

**5.2 Gestion des rendez-vous :**

• Planification et gestion : Planifier, modifier ou annuler des rendez-vous chaque fois que c’est nécessaire. Manipuler t les horaires des médecins et les rendez-vous.

• Notifications automatiques : authentifiez automatiquement les messages à envoyer aux patients ou aux médecins avant les rendez-vous, par exemple par EMAIL et les SMS.

**5.3 Gestion du profil patient :**

• Mettre à jour et gérer les dossiers des patients dans leur intégralité, qui comprennent :

Renseignements personnels : Nom complet, nationalité, numéro de téléphone, lieu de résidence.

- Une brève description des antécédents médicaux retrace les traitements antérieurs, les réactions médicamenteuses, les injections reçues, les médicaments prescrits précédemment et l’historique de vaccination.

**5.4 Génération de documents :**

• Prescriptions : Produire des ordonnances automatisées et indiquer les types de médicaments, leur dosage ou la dose appropriée comme une copie analogique.

• Certificats médicaux : Production de certificats médicaux automatiquement pour diverses raisons, notamment les congés de maladie et l’inactivité physique pour des raisons médicales.

**6. Analyse du Code**

**6.1 Package: gestion\_de\_person**

* **Classe: Person**
  + **Attributs:** nom, prenom, numeroT (nom, prénom, numéro de téléphone)
  + **Méthodes:** Méthodes getter et setter pour les attributs.

**6.2 Package: gestion\_de\_fiche\_patient**

* **Classe: Patient**
  + **Attributs:** Hérite de Person, ID, DateN (date de naissance), age, adresse, sexe
  + **Méthodes:** Méthodes getter et setter, toString(), afficherPatient() (afficher les informations du patient)
  + **Classe: Patient\_Management**
    - **Attributs:** patients (ArrayList d'objets Patient)
    - **Méthodes:**
      * createPatient(): Invite à fournir les informations du patient, valide les données et crée un nouvel objet Patient.
      * recherchePatient(String nom): Recherche les patients par nom.
      * afficherPatients(): Affiche les informations de tous les patients.
      * add(), edit(), remove() méthodes pour gérer la liste des patients.

**6.3 Package: gestion\_medecin**

* + **Classe: Docteur**
    - **Attributs:** Hérite de Person, ID, specialite, motDePasse
    - **Méthodes:** Méthodes getter et setter, toString(), afficherDocteur(), verifierMotDePasse() (vérification du mot de passe)
  + **Classe: Docteur\_Management**
    - **Attributs:** docteurs (ArrayList d'objets Docteur)
    - **Méthodes:** creerDocteur(), afficherDocteurs(), ajouter(), modifier(), supprimer() méthodes pour gérer les médecins.

**6.4 Package: gestion\_de\_rendez\_vous**

* + **Classe: Rendez\_vous**
    - **Attributs:** ID, docteur, patient, date, heure
    - **Méthodes:** Méthodes getter et setter, afficherDetails(), afficherRendezVous()
  + **Classe: Rendez\_vous\_Management**
    - **Attributs:** listeRendezVous (ArrayList d'objets Rendez\_vous)
    - **Méthodes:**
      * creerRendezVous(): Invite à fournir les détails du rendez-vous, crée un nouvel objet Rendez\_vous.
      * rechercherRendezVous(): Recherche les rendez-vous.
      * afficherRendezVous(): Affiche les rendez-vous.
      * ajouter(), modifier(), supprimer() méthodes pour gérer les rendez-vous.

**6.5 Package: gestion\_de\_suivi\_et\_dossier\_medicale**

* + **Classe: SuiviMedical**
    - **Attributs:** date, heure, patient, ordonnance, poids, taille, observationMedicale, resumeTraitement
    - **Méthodes:** Méthodes getter et setter, afficherSuiviMedical()
  + **Classe: DossierMedical**
    - **Attributs:** patient, medecin, listeSuiviMedical (ArrayList d'objets SuiviMedical)
    - **Méthodes:** Méthodes getter et setter, afficherDossierMedical()
  + **Classe: DossierManagement**
    - **Attributs:** listeDossiersMedicaux (ArrayList d'objets DossierMedical)
    - **Méthodes:**
      * creerNouveauDossier(Patient patient, Medecin medecin): Crée un nouveau DossierMedical.
      * mettreAJourSuivi(DossierMedical dossier, Ordonnance ordonnance): Met à jour le suivi.
      * rechercherDossier(String nom): Recherche les dossiers médicaux par nom du patient.
      * ajouter(), modifier(), supprimer() méthodes pour gérer les dossiers médicaux.

**6.6 Package: gestion\_ordonnance**

* + **Classe: Ordonnance**
    - **Attributs:** medecin, patient, medicaments, instructions, datePrescription
    - **Méthodes:** Méthodes getter et setter, toString(), afficherOrdonnance()
  + **Classe: Ordonnance\_Management**
    - **Attributs:** ordonnances (ArrayList d'objets Ordonnance)
    - **Méthodes:**
      * creerOrdonnance(Medecin medecin, Patient patient): Invite à fournir les détails de l'ordonnance, crée une nouvelle Ordonnance.
      * rechercherOrdonnance(String nom): Recherche les ordonnances par nom du patient.
      * afficherOrdonnances(): Affiche les ordonnances.
      * ajouter(), modifier(), supprimer() méthodes pour gérer les ordonnances.
  + **Classe: GestionOrdo**
    - **Méthodes:**
      * modifierOrdonnance(): Permet la modification des ordonnances existantes.
      * supprimerOrdonnance(): Permet la suppression des ordonnances existantes.
      * gestionOrdonnances(): Fournit un menu pour la gestion des ordonnances.

**7. Conclusion et perspectives d’avenir**

Avantages du logiciel : Le système de gestion médicale facilite la gestion des dossiers cliniques, la planification des rendez-vous ainsi que la gestion du profil du patient. Java garantit que l’application est sûre et robuste. Les diagrammes UML indiquent la structure des utilisations utilisées.

Recommandations :

Améliorations prévues pour l’avenir :

• Capacités de télémédecine : Inclure des fonctionnalités telles que les appels vidéo ou audio ou d’autres formes d’examens à distance.

• Améliorations de l’interface utilisateur : Faciliter l’utilisation et la navigation dans le programme.

**8.** **Approche Distincte de la Classe de Gestion et Sa Valeur :**

L’une des principales caractéristiques de conception de notre système est la classe de gestion(managment) qui est incluse dans chaque classe (par exemple, Patient\_Management dans le package patient), à l’exception de la classe d’application car elle contient la class Main .

Ce type d’approche présente plusieurs avantages :

* Code ordonné : Il devient plus facile de suivre le code puisque chaque paquet a une classe qui traite de ses éléments.
* Encapsulation of Logic : Ces opérations sont maintenues dans ces classes de gestion avec d’autres opérations ainsi la séparation est faite correctement.
* Amélioration de la maintenabilité : les changements qui se rapportent à la logique d’entreprise sont ainsi concentrés sur les classes de gestion, l’applicabilité et la gestion est donc plus facile.
* Modularité accrue : Chacun des paquets fonctionne indépendamment ce qui favorise la modularité de la conception et la réutilisabilité.

**9. Mesures de sécurité**

• **Politiques de mots de passe :** Imposer des mots de passe complexes (minimum de 12 caractères, incluant des chiffres et des symboles).

• Chiffrement des données : sécurité des informations sur les patients grâce à des techniques de chiffrement des données sensibles.

• Contrôles d’accès : créer un système de blocage pour les utilisateurs non autorisés en leur attribuant des rôles de rendu par rapport aux données patient.

**10.Bases de données en réseau**

Commencer par une première esquisse d’un schéma de base de données (qui serait adéquat pour les informations sur le patient, les renseignements sur le rendez-vous, l’historique médical, etc.

Intégration avec un SGBDR : Par exemple, MySQL, PostgreSQL, Oracle ou d’autres SGBDR.

**11. Utilisation plus efficace des ressources**

• Créer une procédure automatique : qui enverrait automatiquement deux rappels aux clients pour les rendez-vous qu’ils ont pris.

• Rendez-vous sur le Web : les clients peuvent utiliser l’Internet pour planifier leurs rendez-vous grâce à des calendriers intégrés dans l’application Web.

• Gestion des appels d’offres : La disponibilité et les demandes de rendez-vous peuvent être facilitées par l’intégration des affectations pour les appels d’offres.

**12. Caractéristiques de la télémédecine**

• Consultations vidéo prévues – étudier l’utilisation de la vidéoconférence et d’autres moyens qui permettront de tenir des consultations et de surveiller l’état de santé du patient à distance.

**13. Capacités de rapport**

• Statistiques sur les rendez-vous – ajouter des fonctionnalités pour automatiser la génération de rapports sur les statistiques sur les rendez-vous, le type de patients, l’endroit d’où ils viennent, les maladies qu’ils ont et d’autres variables.

En faisant face à ces améliorations, le système de gestion médicale peut être amélioré comme solution fiable et complète pour la gestion des ressources opérationnelles de l’établissement de santé contemporain.