Chapitre 2 : Analyse et Conception

**Introduction :**

La conception est une étape essentielle pour assurer le développement efficace de notre logiciel de gestion hospitalière. Après avoir clarifié l’idée de notre projet en précisant ses fonctionnalités, nous décrivons ici de manière détaillée la conception du logiciel puis son architecture.

2. Analyse et Conception

* 1. Cycle de vie

Le cycle de vie d’un logiciel (ou software life cycle) englobe toutes les phases de développement d’un logiciel, de sa conception initiale à sa mise hors service. Son découpage en étapes permet de vérifier la progression du développement du logiciel en validant qu’il répond aux besoins exprimés tout en assurant la conformité du processus de développement aux méthodes définies. Il existe plusieurs modèles de cycle de vie, chacun adapté au type de logiciel et aux exigences spécifiques du client.

* 1. Choix du Modèle de cycle de vie

Le choix du modèle de cycle de vie est crucial car il détermine l’organisation et la gestion du développement logiciel. Pour notre projet de gestion hospitalière, le modèle en V a été retenu pour sa capacité à structurer le processus de développement de manière méthodique. Ce modèle met l’accent sur la validation et la vérification à chaque étape, assurant ainsi la conformité du logiciel aux exigences et spécification définies.

* 1. Méthodes de conception

La conception logicielle nécessite l’utilisation de méthodes et techniques appropriées pour élaborer une architecture cohérente et fonctionnelle. Nous avons recours à des pratiques telles que l’analyse des besoins, la modélisation des données et l’utilisation de diagrammes UML pour représenter visuellement les interactions entre les composants du système. Ces méthodes nous aident à clarifier les exigences, à identifier les fonctionnalités essentielles et à concevoir une solution robuste et évolutive.

3. Modélisation avec le langage UML

3.1 Langage UML

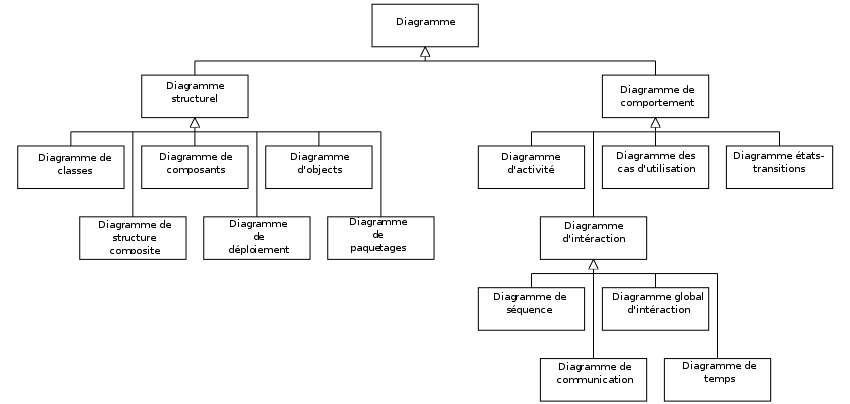
* Signification : UML signifie « Unified Modeling Language » (langage de modélisation objet unifié).
* Origines : UML est le résultat de la fusion de plusieurs méthodes de modélisation :
* Booch: Grady Booch (1991),
* OMT (Object Modelling Technique): James Rumbaugh (1991),
* OOSE (Object Oriented Software Engineering): Ivar Jacobson (1992).
* Évolution

Naissance de UML : UML a émergé dans les années 1990.

Normalisation : UML 1.0 a été normalisé en janvier 1997 par l'Object Management Group (OMG).

Versions successives : UML 2.0 a été adopté en juillet 2005, suivi de la version 2.5.1 en 2017, validée par l'OMG.

* Objectifs
* Faciliter la conception : UML est conçu pour simplifier la création des documents nécessaires au développement logiciel en tant que standard de modélisation d’architecture logicielle.
* Types de diagrammes : UML 2.0 propose 14 types de diagrammes, comprenant six diagrammes structurels et huit diagrammes comportementaux.
* Complémentarité des diagrammes : Les diagrammes UML sont hiérarchiquement dépendants et se complètent pour permettre la modélisation complète d’un projet tout au long de son cycle de vie.



3.2 Identification des acteurs :

L’acteur est une entité qui interagit avec le système, qu’il s’agisse d’un utilisateur humain, d’une organisation, d’une machine ou d’un autre système externe.

Dans notre système, les acteurs sont :

Administrateur : L’administrateur joue un rôle crucial en assurant la création de nouveaux comptes pour le personnel hospitalier, ainsi que la distribution des rôles et des permissions associées. Il est responsable de la gestion des services disponibles à l’hôpital, la gestion des traitements. De plus, l’administrateur à la possibilité d’ajouter de nouvelles chambres au système.

Agent de la Finance : il est responsable de l’établissement des factures pour des services médicaux fournis aux patients, le suivi des paiements effectués par les patients.

Responsable des Stocks: Le responsable des stocks joue un rôle essentiel dans la gestion des stocks de fournitures médicales et non médicales. Il est chargé de veiller à ce que les ressources nécessaires aux soins soient toujours disponibles en quantité suffisante. Ses responsabilités incluent la réception, le stockage et la distribution des fournitures médicales, ainsi que la tenue à jour des inventaires.

Médecin : Les médecins ont la responsabilité d’intégrer les patients dans leur liste de suivi et d’accéder à leurs dossiers médicaux complets, proposer des calendriers pour les rendez-vous, de recevoir des notifications concernant les rendez-vous à venir, ainsi d’affecter les patients aux patients aux chambres appropriées en fonction de leur état de santé et de leurs besoins.

Agent de Réception :

Le rôle de l'agent de réception est crucial pour assurer une gestion efficace de l'hôpital. Il est chargé de l'enregistrement des patients, de la mise à jour de leurs informations dans le système, ainsi que de l'attribution des chambres libres aux patients lors de leur hospitalisation. De plus, il gère la libération des chambres lorsque les patients sont prêts à partir. En outre, si les patients souhaitent prendre rendez-vous avec un prestataire de soins, l'agent de réception organise la planification et le système envoie des notifications aux professionnels de santé concernés.

3.3 Diagramme de cas d’utilisation

Le diagramme de cas d’utilisation représente la structure des principales fonctionnalités nécessaires aux utilisateurs du système. Il offre une vue globale des interactions entre les acteurs et le système, mettant en évidence les différents scénarios d’utilisation.

L’objectif principal du diagramme de cas d’utilisation est de clarifier les exigences fonctionnelles du système et de fournir une base pour la conception détaillée de celui-ci. Il sert également de point de départ pour l’identification des besoins des utilisateurs et des interactions entre les différents composants du système.

Diagramme de cas d’utilisation

Figure : diagramme de cas d’utilisation du système

Figure : cas d’utilisation détails pour l’administrateur

Figure : Cas d’utilisation détails pour l’agent de la finance

Figure : cas d’utilisation pour le responsable de stock

Figure : cas d’utilisation pour le Médecin

Figure : cas d’utilisation pour le Réceptionniste

3.4 Diagramme de séquence :

Un diagramme de séquence comprend un ensemble d’objets, représentés par des lignes de vie, et les messages échangés entre ces objets lors de l’interaction. Il illustre les interactions entre les objets dans le cadre d’un scénario de cas d’utilisation. Les messages peuvent être de différents types : synchrones, asynchrones et réflexifs.

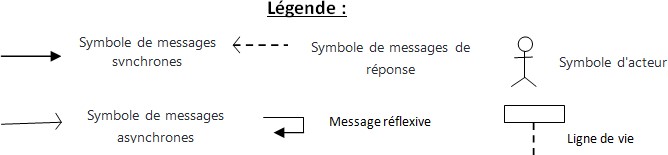
Définitions :

* Message : définit une communication particulière entre des lignes de vie (objets ou acteurs)
* Message synchrone : bloque l’expéditeur du message jusqu’à la réponse du destinataire.
* Message asynchrone : l’expéditeur du message ne reste pas bloqué en attendant la réponse du destinataire.
* Message réflexive : c’est un message d’un objet vers lui-même

Objet : c’est une instance d’une classe.

Ligne de vie : représente un participant à une interaction (objet ou acteur)

Scénario : c’est une instance d’un cas d’utilisation



*Ci-dessous quelques diagrammes de séquences relatifs à notre système :*

Cas d’utilisation : Authentification

Acteurs : Utilisateur et système

**Description :** L’authentification est une étape clé de notre système. Elle permet d'ouvrir l'interface appropriée pour chaque utilisateur du système, garantissant ainsi que seuls les utilisateurs autorisés peuvent accéder aux fonctionnalités correspondantes à leurs rôles.

Figure : diagramme de séquence de l’authentification

Explication Détaillée:

Après avoir lancé l'application, l'utilisateur est dirigé vers la fenêtre de connexion. L'utilisateur doit alors fournir ses identifiants (matricule et mot de passe). Le système vérifie ensuite ces identifiants. Si les identifiants sont corrects, l'utilisateur est dirigé vers l'interface qui lui est destinée. En cas d'identifiants incorrects, un message d'erreur est affiché.

Cas d’utilisation : Gestion du planning

Acteurs : Médecin et Système

Description : La gestion du planning consiste à établir des plannings de rendez-vous qui seront consultés lors de la prise de rendez-vous par les patients et autres acteurs. Cette fonctionnalité permet aux médecins de gérer efficacement leur emploi du temps.

Figure :

Explication Détaillée:

Lorsque le médecin demande de gérer son calendrier, le système recherche les calendriers existants dans la base de données et les affiche à l'écran. Le médecin a alors la possibilité d'ajouter un nouveau calendrier ou de modifier les calendriers existants.

Cas d’utilisation : Générer facture

Acteurs : Agent de la Finance, Contrôleur, Système, Base de Données et imprimante.

Description : La facturation qui consiste à générer des factures pour les patients, est très importante car elle permet d’avoir une vision claire des traitements offerts par l’hôpital et gérer la finance efficacement.

Figure :

Explication Détaillée :

L'agent de la finance initie le processus de génération de facture en demandant au système de créer une facture pour un patient donné. Pour cela, il doit fournir des informations essentielles concernant le patient et le médecin traitant, ainsi que les traitements effectués.

Le système propose une liste des traitements disponibles, qui sont fournis par le système lui-même. L'agent de la finance sélectionne les traitements appropriés parmi cette liste et les inclut dans la facture.

Une fois toutes les informations requises fournies, le système enregistre la facture dans la base de données. À ce stade, l'impression de la facture est automatiquement déclenchée, générant ainsi un document prêt à être remis au patient ou à d'autres parties concernées.

Cas d’utilisation : Gestion des patients

Acteurs : Réceptionniste, Système.

Description : Le réceptionniste utilise le système pour gérer les informations des patients. Le système récupère les données des patients depuis la base de données et les affiche au réceptionniste. Le réceptionniste peut chercher des patients, mettre à jour ses informations ou d'enregistrer un nouveau patient. En fonction de ces actions, le système effectue les opérations nécessaires et envoie une confirmation au réceptionniste.

Figure :

Explication Détaillée :

Lorsque le réceptionniste lance une recherche, il fournit un mot-clé tel que le nom ou le prénom du patient. Le système utilise ce mot-clé pour interroger la base de données et récupérer une liste de patients correspondant aux critères de recherche. Cette liste est ensuite affichée à l'écran pour que le réceptionniste puisse consulter les informations pertinentes sur les patients.

Si le réceptionniste souhaite ajouter un nouveau patient ou mettre à jour les informations d'un patient existant, il doit remplir le formulaire correspondant dans le système. Le formulaire contient des champs à remplir avec les informations du patient, tels que le nom, le prénom, la date de naissance, etc. En fonction de l'action sélectionnée (ajout ou mise à jour), le système effectuera le traitement approprié : l'enregistrement d'un nouveau patient dans la base de données ou la mise à jour des informations du patient existant. Une fois que le réceptionniste a rempli et validé le formulaire, le système traitera les données fournies et enverra une confirmation de l'action effectuée, que ce soit l'enregistrement d'un nouveau patient ou la mise à jour des informations d'un patient existant.

Cas d’utilisation : Gestion des Utilisateurs

Acteurs : Système, Administrateur, Base de données, Contrôleur.

Description : La gestion des utilisateurs est une fonctionnalité essentielle du système, car elle permet à l'administrateur de créer, mettre à jour et supprimer des comptes utilisateurs dans le système. Cette fonctionnalité est cruciale car elle garantit que seules les personnes autorisées ont accès au système et aux fonctionnalités appropriées en fonction de leur rôle attribué.

Figure :

Explication Détaillée :

L'administrateur accède à la fonctionnalité de gestion des utilisateurs dans le système, le système recherche les utilisateurs existants dans la base de données et afficher sur la fenêtre de l’administrateur.

Lorsque l'administrateur souhaite rechercher un utilisateur existant, il fournit un mot-clé ou une description du compte recherché. Le système effectue une recherche dans la base de données pour trouver les comptes correspondants à la description fournie puis afficher le résultat de la rechercher.

L'administrateur a la possibilité d'ajouter, de modifier ou de supprimer des comptes utilisateurs.

Pour chaque opération, l'administrateur doit remplir un formulaire avec les informations nécessaires concernant le compte utilisateur. Le système effectue alors les opérations demandées, que ce soit l'ajout, la modification ou la suppression du compte utilisateur.

Après avoir effectué le traitement, le système envoie un message de confirmation à l'administrateur pour indiquer que l'opération a été réalisée avec succès.

Pendant toutes ces opérations, un contrôleur vérifie les champs saisis par l'administrateur pour s'assurer de leur validité. Si les champs sont corrects, le traitement se poursuit normalement.

En cas de champs incorrects ou incomplets, le contrôleur envoie un message à l'administrateur pour lui demander de remplir ou de corriger les champs concernés.

Cas d’utilisation : Gestion des rendez-vous

Acteurs : Agent de réception, système et prestataire de soins.

Description : La gestion des rendez-vous permet de planifier, modifier et annuler les rendez-vous des patients avec les prestataires de soins. Cela facilite la coordination des horaires et des disponibilités entre les patients et les professionnels de la santé.

Figure :

Explication Détaillée :

Lorsque l’agent de réception demande l'accès à la fonctionnalité de gestion des rendez-vous, le système recherche les rendez-vous en cours et les affiche à l’écran.

Si l’agent souhaite planifier un rendez-vous, elle doit fournir les détails du rendez-vous tels que ceux du patient et du prestataire de soins concerné. En fonction des informations relatives au prestataire de soins, le système recherche le créneau horaire disponible pour ce dernier. L’agent sélectionne ensuite l’horaire souhaité par le patient et valide la prise de rendez-vous.

Lors de la modification ou de la suppression d'un rendez-vous existant, l'agent de réception fournit les informations nécessaires pour identifier le rendez-vous à modifier ou à supprimer. Une fois les informations fournies, le système effectue les modifications ou la suppression correspondante dans le système de gestion des rendez-vous. Après avoir effectué ces actions, le système renvoie une confirmation à l'agent de réception pour indiquer que la modification ou la suppression a été effectuée avec succès.

3.5 Diagramme de Classe :

Qu’est-ce qu’une classe ?

Une classe représente un objet ou un ensemble d'objets qui partagent des attributs et des comportements (méthodes) communs. C’est une abstraction des choses du monde réel.

Une classe possède :

* **Un nom** : le nom est placé dans la première section de la figure d’une classe.
* **Des attributs** : les attributs sont placés dans la deuxième section de la figure d’une classe.
* **Des méthodes** : elles sont placées dans la troisième section de la figure d’une classe.

Un diagramme de classe permet de modéliser les objets qui constituent le système, pour afficher les relations entre les objets, pour décrire ce que ces objets font et les services qu'ils fournissent. C’est le diagramme le plus importance et indispensable dans la modélisation UML d’un système réalisé avec le concept Orienté Objet.

Il est représenté par un rectangle divisé en trois (3) parties la première contient le nom de la classe, la seconde la liste des attribues avec leurs types et visibilité et la troisième partie contient les méthodes (ou Opérations).

Types de Visibilité:

* **Public (+)** : Accessible depuis n'importe où.
* **Privé (-)** : Accessible uniquement à l'intérieur de la classe.
* **Protégé (#)** : Accessible à l'intérieur de la classe et des sous-classes.
* **Package** : Accessible aux classes du même package (utilisé souvent en Java).

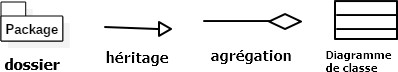
**Relation entre les Classes**:

* Association : Une relation simple où une classe utilise une autre.
* Agrégation : Une relation "a un" où une classe est composée d'une ou plusieurs instances d'une autre classe, mais sans dépendance forte.
* Composition : Une forme forte d'agrégation où la classe composante ne peut pas exister indépendamment de la classe composite.
* Héritage : Une relation où une classe (sous-classe) hérite des attributs et méthodes d'une autre classe (super-classe).

1.1 signifie Un et Un seul

1.. \* signifie Un ou plusieurs

0.. \* signifie zéro, Un ou plus



***Légende***

Figure :

4. Modélisation avec la méthode Merise

4.1 Introduction

La méthode Merise, née dans les années 1970 et largement utilisée jusqu'aux années 1980, est un ensemble de techniques d'analyse, de conception et de réalisation de systèmes d'informations. Son utilisation était répandue pour l'informatisation des organisations à grande échelle à cette époque.

Elle repose sur une approche méthodique et structurée, visant à modéliser les différents aspects d'un système d'information, depuis sa conception jusqu'à sa mise en œuvre opérationnelle. Cette méthode offre un cadre formel pour la compréhension des besoins des utilisateurs, la spécification des fonctionnalités du système, la représentation des données et des traitements, ainsi que la conception de l'architecture technique.

La méthode Merise se compose de plusieurs étapes clés, allant de l'analyse des besoins initiaux à la définition des spécifications détaillées du système. Parmi ces étapes, on retrouve notamment la conception du Schéma Directeur des Données (SDD), l'Analyse des Flux (ADF), la création du Modèle Conceptuel des Données (MCD), du Modèle Organisationnel des Données (MOD), du Modèle Logique des Données (MLD), et enfin du Modèle Physique des Données (MPD).

L'approche Merise permet également de prendre en compte l'évolution des systèmes d'informations au fil du temps, en facilitant la maintenance et les évolutions futures. En outre, elle favorise la collaboration entre les différents acteurs impliqués dans le projet, tels que les analystes, les concepteurs, les développeurs et les utilisateurs finaux.

4.2 Dictionnaire de Données :

Le dictionnaire de données est un outil fondamental dans la méthode Merise. Il documente de manière détaillée chaque élément de données du système d'information, en fournissant des informations telles que le nom de la donnée, son type, sa description, son domaine de valeurs, les contraintes qui lui sont associées, sa source, son utilisation dans le système et ses relations éventuelles avec d'autres données. Ce document constitue la première étape de l'élaboration du Modèle Conceptuel des Données (MCD), servant de référence pour la modélisation précise et cohérente des données du système.

|  |  |  |
| --- | --- | --- |
| Identifiant | Type de données | Description |
| ID\_Patient | Entier | Identifiant unique du patient. |
| Matricule | Chaîne de caractères | Identifiant unique de l'utilisateur. |
| num\_chambre | Entier | Numéro unique de la chambre. |
| id\_patient | Entier | Identifiant unique du patient. |
| num\_facture | Chaîne de caractères | Numéro unique de la facture. |
| codeTraitement | Chaîne de caractères | Code unique du traitement. |
| code\_service | Entier | Code unique du service. |
| ID\_Article | Entier | Identifiant unique de l'article. |

(Note : Seules les colonnes pertinentes pour notre analyse ont été incluses dans ce dictionnaire de données.)

4.3 Modèle Conceptuel de données (MCD)

Le MCD représente les entités du système, ainsi que les relations entre ces entités et leurs attributs clés. En visualisant le MCD, les utilisateurs peuvent comprendre la structure des données et les interactions entre elles dans le système d'information. Cela facilite la conception et la modélisation précise du système, en fournissant une représentation graphique claire des entités et de leurs relations.

4.4 Modèle Logique de données (MLD)

Le modèle logique de données (MLD) représente une étape essentielle dans la méthodologie Merise, traduisant le Modèle Conceptuel de Données (MCD) en une structure de données indépendante de la technologie utilisée. Cette traduction se fait en définissant les tables, les clés primaires et étrangères, ainsi que les contraintes d'intégrité nécessaires à la cohérence des données dans la base de données.

Pour passer du MCD au MLD, nous suivons les règles suivantes :

* **Transformation des entités en relations :** Chaque entité conceptuelle du MCD se transforme en une relation dans le MLD. Par exemple, une entité "Client" dans le MCD pourrait devenir une relation "Client" dans le MLD, avec des attributs tels que nom, prénom et adresse.
* **Création de relations pour les associations :** Les associations de type (*1, n) ou (*, n) dans le MCD deviennent des relations distinctes dans le MLD. Par exemple, une association "Commande" entre les entités "Client" et "Produit" pourrait devenir une relation "Commande" dans le MLD, avec des attributs tels que date et quantité.
* **Gestion des associations de type (*, n) (*,1) :** Les associations de type (*, n) (*,1) disparaissent dans le MLD. Par exemple, si une association "Affectation" entre les entités "Employé" et "Projet" a été définie dans le MCD, elle pourrait être gérée dans le MLD en dupliquant la clé de l'entité "Employé" comme clé étrangère dans l'entité "Projet".
* **Héritage des entités :** Dans le cas d'une association de type héritage, la clé de l'entité mère devient une clé étrangère dans les entités filles. Par exemple, si une entité "Véhicule" est la mère de "Voiture" et "Moto", alors la clé de "Véhicule" deviendra une clé étrangère dans "Voiture" et "Moto".

Représentation du MLD :

Les clés primaires sont soulignées et les clés étrangères sont précédées par le symbole ‘ # ’ :

1. Patient (id\_patient, dateNais, nom, prenom, sexe, etatcivil, telephone, sanguin, taille, poids, lien, assureur, profession, adresse)
2. dossier (#ID\_Patient, #Matricule\_Med, détails, prescription, resultats\_test)
3. rendez\_vous (#ID\_Patient, #Matricule, type, date\_rdv, heure, details\_rdv, statut)
4. chambre (num\_chambre, type, disponibilite, categorie)
5. Hospitalisation (#id\_patient, #num\_chambre, date\_admission, date\_sortie, status)
6. notification (#Matricule, #id\_rdv, message, date\_Envoie)
7. Facture (num\_facture, #ID\_Patient, #codeTraitement, #Matricule, détails, montant, date, status)
8. Traitement (codeTraitement, type, nom, description, prix)
9. Service (code\_service, #codeTraitement, nom, description)
10. stock (id\_article, article, type, quantite, date\_entre)
11. stock\_livre (#code\_service, #id\_article, quantite, date\_livraison)
12. Utilisateurs (Matricule, Nom, Prenom, DateNaissance, Password, Telephone, Specialite, Email, Role, Sexe)
13. Liste\_User\_Service (#Matricule, #code\_service)
14. Personnel\_Administrateur (#Matricule, Fonction)
15. Personnel\_Médicale (#Matricule, Spécialité)
16. Calendar (jour, heure, #Matricule)

Conclusion :

La phase d'analyse et de conception est une étape cruciale dans le développement de notre logiciel de gestion hospitalière. En détaillant les différents aspects de la conception, nous avons pu établir une base solide pour la réalisation du système.

Nous avons commencé par choisir le modèle de cycle de vie le plus adapté à notre projet, optant pour le modèle en V pour sa rigueur méthodologique. Ensuite, nous avons utilisé des méthodes de conception telles que l'analyse des besoins et la modélisation avec le langage UML pour définir les fonctionnalités du système et son architecture.

À travers les diagrammes de cas d'utilisation, de séquence et de classe, nous avons pu visualiser les interactions entre les utilisateurs et le système, ainsi que la structure des données et des traitements. De plus, en utilisant la méthode Merise, nous avons pu élaborer un modèle conceptuel et logique des données, définissant ainsi la structure de la base de données du système.

En résumé, la phase d'analyse et de conception nous a permis de clarifier les besoins et les exigences du projet, de concevoir une solution adaptée et de définir une architecture solide pour notre logiciel de gestion hospitalière. Cela nous met sur la voie d'un développement efficace et réussi du système, en garantissant qu'il répondra aux attentes des utilisateurs et aux exigences du domaine médical.