République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D' **I**NFORMATIQUE
DEPARTEMENT DE GENIE BIOMEDICAL

Mémoire de Fin d'Etudes
De MASTER ACADEMIQUE

Domaine : **Sciences et Technologies**
Filière : **Génie Biomédical**
Spécialité : **Instrumentation Biomédical**

Présenté par
Mokrane FELLAG
Fatima BELHADI

Thème

Conception, réalisation et mise au point d'une application pour la gestion d'une pharmacie hospitalière

Mémoire soutenu publiquement le 29/06/ 2025 devant le jury composé de :

M. Mourad LAZRI
Pr, UMMTO, Président

M. Takfarinas CHELLI
MAA, UMMTO, Promoteur

M. M'hamed saadi BACHIR
MCB, UMMTO, Examinateur

Promotion : 2024/2025

Remerciements

Nous tenons tout d'abord à exprimer notre sincère gratitude à notre encadrant, Mr CHELLI Takfarinas, pour son soutien indéfectible, son accompagnement, sa disponibilité et ses conseils précieux tout au long de ce travail.

Nous souhaitons aussi remercier l'Université de Mouloud MAMMERI de Tizi Ouzou pour l'environnement académique et les ressources qu'elle a mises à notre disposition durant toutes nos années d'études.

Nous exprimons aussi notre reconnaissance à l'ensemble des membres du jury, pour le temps qu'ils ont consacré à l'évaluation de notre travail, ainsi que pour leurs remarques et suggestions constructives.

Nous tenons à exprimer notre profonde reconnaissance aux deux pharmaciennes, Dr OUALLOUCHE et Dr SADOUDI, pour leur accueil chaleureux, leur bienveillance et leur encadrement durant notre stage.

Un grand merci à toute l'équipe de la pharmacie hospitalière du CHU de Tizi Ouzou pour leur professionnalisme et les échanges enrichissants qui ont grandement contribué à la réalisation de ce projet.

Enfin, nous remercions toutes les personnes qui, de près ou de loin, ont apporté leur aide et leur soutien à l'accomplissement de ce mémoire.

Dédicaces

Je souhaite dédicacer se travail aux personnes les plus chère à mon cœur :

- A mon père, pour tous les efforts qu'il a faits, souvent dans l'ombre, afin que je puisse devenir la personne que je suis aujourd'hui. Pour son courage ses sacrifices silencieux, pour sa confiance inébranlable et les valeurs qu'il m'a transmises. De lui je tien le sens du devoir, le respect et la persévérance. Ce travail est le reflet de tous ses enseignements, et je lui serai éternellement reconnaissant.
- A ma mère, pour son amour inconditionnelle qu'elle me porte depuis mon arrivé dans ce monde, pour la patience et la bienveillance dont a elle fait preuve, surtout durant mon adolescence. Je ne te serai à jamais reconnaissent pour m'avoir épaulé et soutenue durant les étapes de ma vie. Merci de m'avoir élevé avec autant de tendresse et de douceur.
- A ma sœur adorée, tous les mots ne seront décrire l'amour que je te porte. Je me rappellerai toujours du jour ou ta courue pied nue afin de récupérer mon jouet (camion), des mains d'un autre enfant, une fois arrivé chez lui tu voulais pertinemment que sa mère le fasse sortir pour le frapper. Je tiens à te remercie pour tous les conseils avisés que tu m'as donnés et pour le soutien indéfectible que tu m'as apporté et que tu m'apporte toujours.
- A mon beau-frère, la dédicace devrait venir de toi car la lumière de la maison illumine ta vie à présent. Je te suis reconnaissant de prendre soin de la personne la plus chère à mon cœur. A présent tu fais partie de la famille, je me remémorerai toujours les excellents moments qu'on a passé ensemble, notamment à Bologhine en compagnie de Bachir, et les pizzas carrées qu'ont mangé chez Ghanou.
- A mon frère d'une autre mère ; Samir, tu as toujours fait mes louanges et les éloges en ma présence et en mon absence. Toutes mes meilleures anecdotes ont été avec toi.
- A mon autre frère ; Idir, ton humour et le faite que tu sois toujours occupé m'impressionneront toujours. J'ai toujours vue en toi de la bonté et de la sympathie.
- A mes amis, je vous remercie pour votre soutient et votre présence tout au long de ce mémoire et dans ma vie en général.
- A cette personne particulière, pour sa présence apaisante et nos discussion plaisante. Ta sue prendre une place précieuse à mes yeux. PTEAJ.
- Sans oublier tonton Ali, dada Djaffar ainsi que toute ma famille.

Je vous aime, Mokrane.

Dédicaces

Ça me tient vraiment à cœur de dédier ce merveilleux travail :

À mes chers parents, pour leur éducation, leur amour inconditionnel, leurs sacrifices silencieux, leurs prières, leur présence à tout moment que je sois triste ou heureuse, leur confiance en leur petite fille, leur soutien émotionnel et financier afin de subvenir à tous mes besoins et j'en passe. Quoi que je dise ça ne serai jamais assez pour décrire tout ce que vous avaient fait pour moi pour que je devienne cette jeune femme pleine de courage, d'amour, de bonté et de patience que j'ai appris et hérité de vous, et quoi que je fasse ça ne serai jamais assez pour que je vous rende l'appareil ne serai ce qu'un tier de ce que vous m'avaient donné. J'espère de tout mon cœur que je serai toujours la source de votre fierté et bonheur.

À ma petite sœur Alicia, pour la lumière que tu avais apporté à ma vie juste par ta présence, c'est vrai qu'on se chamaille peut-être un peu trop mais rien ne changera l'amour qu'on a l'une pour l'autre. J'espère que tu seras fière de ta grande sœur et que je serai un bon modèle pour toi, afin que tu te battes pour tes rêves et ambitions et tout faire pour les atteindre.

À la mémoire de mon oncle Ferhat, c'est vrai que t'es parti vraiment trop tôt, et que jusqu'à présent je ne réalise pas cela mais bon après tout c'est ça la vie et on doit accepter. Je tiens à te dédier ce travail car t'avait toujours eu foi en moi, même quand tout le monde en doutait j'espère que tu seras fière de ta nièce de là où t'est.

À ma meilleur amie Emily, ma sœur d'une autre mère, ma moitié et mon âme sœur version amitié, quoi que je dise je ne décrirai jamais assez l'amour que je porte pour toi et à quel point je suis reconnaissante d'avoir partagé avec toi la même classe, car c'était grâce à ça que j'avais gagné une amitié aussi forte et sincère. T'était toujours là pour moi pour mes bons et mauvais moments et tu l'es jusqu'à aujourd'hui même si la distance nous sépare mais rien à changer et ça ne changera jamais.

À mes copines Melissa, Dehbia et Sonia, mes petites protégées, qui ont toujours su transformer les moments difficiles en souvenirs joyeux. C'est vrai que ça ne faisait pas super longtemps qu'on se connaisse mais on avait très vite accroché car on portait les mêmes valeurs et le même amour entre nous, je ne vous remercierai jamais assez pour votre soutien et encouragement durant toute cette période je vous aime de tout mon cœur.

Et enfin, je tiens à remercier tous mes oncles et tantes, mes cousins et cousines, mes amis qui m'ont soutenue de prêt ou de loin afin que je finisse ce travail, j'espère que vous êtes tous fière de moi.

Je vous aime, FATIMA.

Résumé

Ce mémoire porte sur la conception d'une application web baptisée G-PHARMA, destinée à optimiser la gestion des dispositifs médicaux en pharmacie hospitalière. S'appuyant sur un stage au CHU de Tizi-Ouzou et des compétences en bases de données (SQL, modélisation) et développement web (HTML, PHP, JavaScript), le projet répond à des besoins concrets de traçabilité, d'automatisation et de sécurité. L'étude aborde aussi les architectures client-serveur, la gestion des stocks et les perspectives d'amélioration. G-PHARMA se veut un outil pratique, évolutif et adapté aux réalités du secteur hospitalier algérien.

Mots clés

Pharmacie hospitalière, Gestion, Base de données, Application web.

Abstract

This thesis focuses on the design of a web application named G-PHARMA, aimed at optimizing the management of medical devices in hospital pharmacies. Based on an internship at the CHU of Tizi-Ouzou and skills in database management (SQL, data modeling) and web development (HTML, PHP, JavaScript), the project addresses concrete needs in traceability, automation and potential improvements. G-PHARMA is intended to be a practical, scalable tool, well adapted to the realities of Algerian hospital environment.

Keywords

Hospital pharmacy, Management, Database, Web application.

Table des matières

Liste des abréviations

ACID : Atomicity, consistency, Isolation, Durability (propriétés fondamentales des transactions SQL).

ADRi : Annuaire de Droit de Remboursement intégré.

API : Application Programming Interface (interface de programmation d'application).

BDC : Bon de Commande.

CHU : Centre Hospitalier Universitaire.

CSS : Cascading Style Sheets (feuilles de style en cascade).

DM : Dispositif Médical.

EPH : Etablissement Publique Hospitalier.

FIFO : First In, First Out (premier entrée, premier sorti).

GEF : Gestion Economique et Financière.

HTML : HyperText Markup Language (langage de balisage hypertexte).

HTTP : HyperText Transfer Protocol (protocole de transfert hypertexte).

JSON : JavaScript Object Notation (format d'échange de données léger).

KPI : Key Performance Indicator (Indicateur clé de performance).

NoSQL : Not Only SQL (base de données non relationnelle).

OI : Ordonnances internes.

OTC : Over The Counter (médicament en vente libre).

PP : Produits Pharmaceutiques.

Prix TTC : prix Toutes Taxes Comprises.

PUI : Pharmacie à Usage Intérieur.

Pill ID : Pill Identification (identification des comprimés par caractéristiques visuelles).

REST : Representational State Transfer (architecture logicielle pour les services web).

RFID : Radio Frequency Identification (identification par radiofréquence).

SGBD : Système de Gestion de Base de Données.

SIH : Système d'Information Hospitalier.

SQL : Structured Query Language (langage de requête structuré).

SSL : Secure Sockets Layer (protocole de sécurisation des échanges).

TLS : Transport Layer Security (protocole de sécurité des communications).

TVA : Taxe sur la Valeur Ajoutée.

UML : Unified Modeling Language (langage de modélisation unifié).

WMS : Warehouse Management System (système de gestion d'entrepôt).

Liste des figures

Liste des tableaux

Introduction générale

Introduction générale

Dans le contexte hospitalier moderne, la gestion rigoureuse des produits pharmaceutiques représente un enjeu majeur pour assurer la qualité des soins dispensés aux patients mais aussi pour garantir une utilisation efficiente des ressources disponibles. Au cœur de cette dynamique, les pharmacies hospitalières jouent un rôle stratégique dans l'approvisionnement, le stockage, la distribution et la sécurisation des produits pharmaceutiques. Leur bon fonctionnement conditionne directement la fluidité des prises en charge et la sécurité des patients.

Cependant à force de constater que de nombreuses structures hospitalières, notamment dans les pays en développements continuent de recourir à des méthodes de gestion partiellement manuelles ou à des logiciels dont les fonctionnalités sont limitées. Cette situation engendre de multiples risques, tel que : les ruptures de stock, le surstockage inutile, les pertes financières, les erreurs de distribution ou l'absence d'une traçabilité précise. Ces dysfonctionnements peuvent impactés la qualité des soins et alourdir la charge de travail des professionnels de santé.

Face à ces constats, l'intégration des technologies de l'information et de la communication (TIC) s'impose comme une réponse incontournable. La digitalisation de la gestion pharmaceutique permet non seulement d'automatiser les processus, mais aussi d'assurer un meilleur suivi des stocks, une traçabilité fiable des produits et une réactivité accrue en cas de besoins. C'est dans ce cadre que s'inscrit notre travail, qui a pour objectif la conception et le développement d'une application web dédier à la gestion d'une pharmacie hospitalière avec un focus particulier sur les dispositifs médicaux et les consommables (Les éléments essentiels souvent négliger dans les solutions existantes).

Ce projet s'appuie sur une double approche : d'une part une immersion sur le terrain à travers un stage effectuer au sein de la pharmacie hospitalière du CHU Nedir Mohammed de Tizi-Ouzou, qui a permis d'identifier les besoins concrets , les limites du système en place et les attentes des professionnels ; d'autre part une mobilisation des connaissances techniques en développement web , architectures client-serveur , bases de donnée relationnelles et outils de programmation (HTML, CSS, PHP, JavaScript , MySQL), afin de proposer une solution adaptée , évolutive et pertinente.

Le mémoire est structuré de manière progressive pour accompagner le lecteur dans la compréhension du sujet. Il débute par une présentation générale des pharmacies hospitalières et des enjeux liés à leur gestion, puis se poursuit par un rappel des fondements théoriques des bases de données et des principes de conception d'applications web. Un chapitre est ensuite consacré

aux outils et technologies utilisés, avant d'aboutir à la description détaillée de l'application développée, illustrant l'ensemble des étapes de conception, de mise en œuvre et de test.

Ainsi, ce mémoire ambitionne de contribuer à l'amélioration des pratiques numériques en milieu hospitaliers en proposant une solution concrète, fondée à la fois sur l'observation du terrain et sur les apports des technologies modernes.

Chapitre I

Généralité sur les pharmacies hospitalières

Chapitre I : généralités sur les pharmacies hospitalières

I.1. Introduction

La gestion des médicaments est un enjeu important des établissements de santé. Elle consiste à organiser les flux d'achats, de stockage et de distribution en interne au niveau des pharmacies hospitalières. En effet, ces dernières sont cruciales pour garantir la sécurité et la disponibilité des produits nécessaires aux soins. Par ailleurs, les technologies de l'information et de la communication ont permis d'améliorer la gestion des stocks. Dans ce chapitre, nous détaillerons le fonctionnement des stocks et de la logistique des produits pharmaceutique au sein des établissements hospitaliers, en nous aidant de notre expérience de stage au sein de la pharmacie hospitalière du CHU Nedir Mohamed de Tizi Ouzou. Ensuite, nous présenterons et étudierons des solutions comme des applications et des logiciels hospitaliers déjà développés pour identifier leurs faiblesses et orienter notre démarche de conception d'une application dédiée au suivi et à la gestion des produits pharmaceutique.

I.2. Généralités sur la gestion des stocks

I.2.1 Définition et rôle de la gestion des stocks

La gestion des stocks représente une composante essentielle de la chaîne d'approvisionnement. Elle englobe l'ensemble des processus permettant de suivre, organiser et optimiser le flux des marchandises, depuis leur approvisionnement jusqu'à leur consommation ou distribution finale. L'objectif principal est d'assurer la disponibilité des bons produits, en quantité suffisante, au bon moment et au bon endroit [1].

Selon Pierre Zermati, la gestion des stocks comprend :

« L'ensemble des tâches, de la plus simple à la plus complexe, nécessaires à l'établissement et à la réalisation du programme d'approvisionnement, au stockage, et à l'orientation des ventes dans les meilleures conditions économiques, tout en évitant les ruptures de stocks et le surstockage. » [5].

I.2.2 Processus de gestion des stocks

La gestion des stocks fait l'objet d'une série d'opérations logistiques qui se succèdent avec rigueur. Elle débute par l'approvisionnement des produits, puis leur entreposage, jusqu'à la livraison des produits aux clients ou services utilisateurs. [1]

L'approvisionnement est une première phase au cours de laquelle des produits sont achetés pour être livrés à un entrepôt ou à un point de distribution. Cette opération vise à se constituer un stock de plusieurs produits afin de répondre en permanence aux commandes [1] [3].

L'entreposage qui est un composant essentiel de la gestion de stock, est l'étape de conservation temporaire des biens avant leur utilisation. A ce stade, deux circuits peuvent se dessiner pour les produits ; soit intégrés dans un circuit interne de traitement et préparés pour expédition, soit conservés en stock jusqu'à leur envoie [1].

L'utilisation des stocks est une phase qui permet de contrôler les niveaux disponibles, de prélever les produits nécessaires pour satisfaire une commande et les livrer aux clients ou aux services utilisateurs [1].

I.2.3 Méthodes modernes de gestion des stocks

Les organisations modernes recourent à des outils technologiques avancés afin d'optimiser la gestion de leurs stocks et d'améliorer la performance logistique.

La gestion périodique : consiste à effectuer des inventaires physiques (souvent mensuels ou trimestriels) et à comparer les données réelles avec les stocks théoriques du système [1]. Cette méthode reste manuelle mais nécessaire pour vérifier les écarts.

La gestion par code-barres : améliore la rapidité et la fiabilité du suivi des produits. Chaque unité est identifiée, scannée, ce qui réduit les erreurs de saisie et permet une traçabilité accrue [1].

La gestion par RFID (Radio Frequency Identification) : très utilisée pour les produits pharmaceutiques coûteux, cette méthode permet le suivi sans contact des produits, même à distance, et leur localisation en temps réel dans l'établissement [2].

La méthode FIFO (First In, First Out) : elle garantit que les produits les plus anciens soient utilisés en priorité, limitant ainsi les risques de péremption [4]. Cette méthode est souvent combinée à une signalisation visuelle dans les rayonnages (étiquettes de couleur, fléchage, etc.).

I.3. Les pharmacies hospitalières

I.3.1 Définition et rôle

La pharmacie hospitalière, également appelée Pharmacie à Usage Intérieur (PUI), est une entité stratégique au sein des établissements de santé. Elle a pour mission de gérer l'ensemble du circuit des produits pharmaceutiques et dispositifs médicaux, depuis leur sélection jusqu'à leur distribution et leur suivi dans les différents services [6].

Selon l'arrêté n° 79/MSP du 24 août 1996, elle est définie comme un service de soutien aux activités thérapeutiques et de prévention, chargé notamment de la sélection, l'approvisionnement, le stockage, la distribution et la dispensation des produits pharmaceutiques [7].

Elle joue également un rôle central dans la qualité des soins**,** en garantissant un usage sûr, efficace et rationnel des médicaments. Le pharmacien hospitalier y assure une fonction de conseil pharmaceutique, participe à la pharmacovigilance, et contribue à la formation continue du personnel médical [5].

I.3.2 Organisation et fonctionnement

Sur le plan administratif, la pharmacie hospitalière est placée sous la responsabilité d'un pharmacien chef. Ce dernier supervise l'ensemble de l'activité pharmaceutique, la gestion du personnel, le respect des règles de bonnes pratiques pharmaceutiques hospitalières**,** ainsi que la conformité aux obligations légales [8].

Le fonctionnement observé durant notre stage au CHU de Tizi-Ouzou repose principalement sur :

Une gestion rigoureuse des stocks pour limiter les ruptures ou le surstockage ;

Une traçabilité complète des médicaments et dispositifs médicaux ;

Une coordination constante avec les services cliniques pour répondre aux besoins en temps réel.

L'organisation interne comprend souvent :

Une zone de stockage organisée par classes thérapeutiques ou conditions de conservation ;

Un espace de réception des PP et de contrôle de la conformité de la livraison ;

Une unité de préparations magistrales ou stériles ;

Un bureau dédié à la gestion informatique des commandes et du stock.

I.3.3 Missions spécifiques

L'article 248 de la **loi sanitaire n°18-11 du 2 juillet 2018** précise que la pharmacie hospitalière est chargée de :

Gérer, stocker et dispenser les médicaments et dispositifs médicaux ;

Promouvoir le bon usage du médicament ;

Contribuer à la qualité, la sécurité et l'efficacité des traitements ;

Appliquer les bonnes pratiques pharmaceutiques hospitalières [8].

Ces activités sont généralement observées dans les pharmacies hospitalières, notamment à travers une organisation structurée du travail du personnel et la répartition claire des responsabilités au sein de l'équipe pharmaceutique.

I.3.4 Contribution à la qualité des soins

Le pharmacien hospitalier joue un rôle fondamental dans la sécurisation du circuit des produits pharmaceutiques**.** Il veille à ce que le bon produit soit délivré, au bon patient, dans

les bonnes conditions (dose, forme) au bon moment. Il participe aussi à l'analyse des prescriptions, à la prévention d'interactions médicamenteuses, et à la formation continue du personnel soignant [6].

Selon BONNABRY Pascal, la pharmacie hospitalière contribue à la qualité des soins selon six axes essentiels :

Approvisionnement,

Fabrication,

Information,

Vigilance,

Enseignement,

Recherche [9].

Cette approche globale vise à améliorer les pratiques pharmaceutiques et à garantir la sécurité des patients tout au long de leur prise en charge.

I.3.5 Personnel et compétences

Le personnel de la pharmacie hospitalière se compose de pharmaciens**,** préparateurs en pharmacie et agents spécialisés. Chacun doit recevoir une formation adaptée à ses fonctions, incluant des compétences en réglementation pharmaceutique**,** logistique hospitalière**,** systèmes d'information et pharmacie clinique [6][8].

I.3.6 Organigramme d'une pharmacie hospitalière

Dans le cadre de notre formation de stage au sein de la pharmacie hospitalière du CHU Nedir Mohamed de Tizi-Ouzou, on a eu le loisir d'apprécier l'organisation interne de ce service. Elle se décline selon une hiérarchie fonctionnelle bien définie, reliant le directeur de l'hôpital à la spécialisation des sections de la pharmacie.

L'organisation est faite de quatre grandes unités, Consommable et instrumentation, une section médicaments, une autre pour les pansements et la dernière section réactifs de laboratoire. Cette répartition vise à permettre une gestion plus efficace selon la nature du produit pris en charge.

On retrouve sur l'organigramme ci-dessous cette organisation :

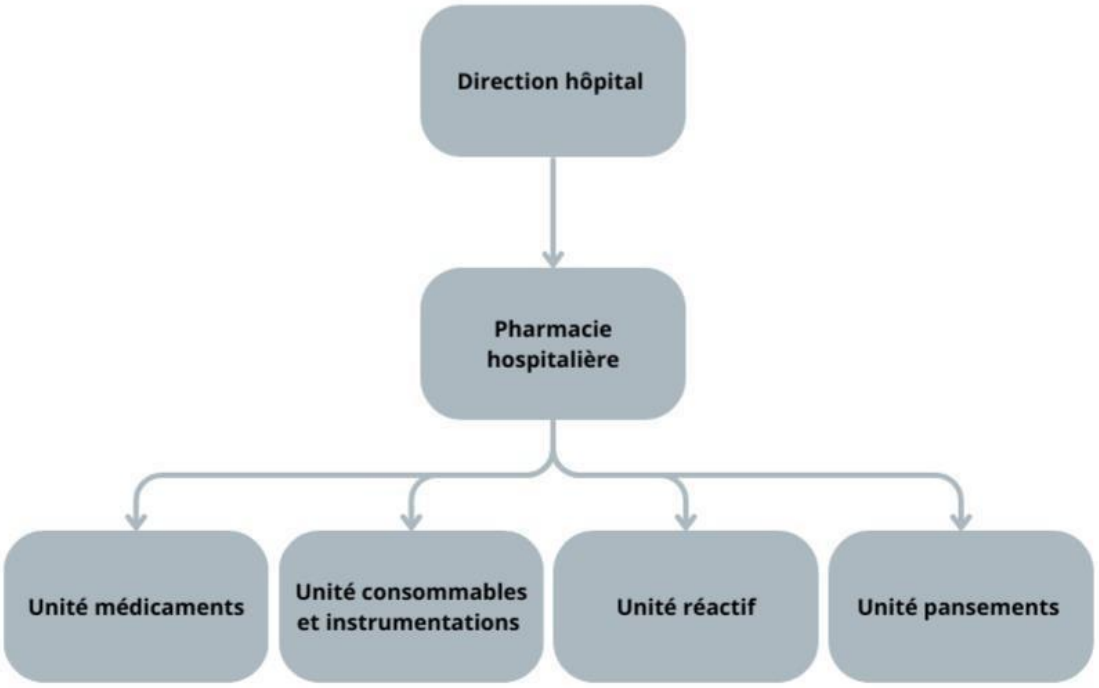

Figure I-1 : Organigramme d'une pharmacie hospitalière.

I.4. La gestion des stocks en milieu hospitalier

Au sein d'une pharmacie hospitalière, la gestion des stocks se révèle stratégique : elle conditionne la continuité des soins, la sécurité du patient et l'efficacité de la prise en charge thérapeutique. En effet, une rupture de stock peut ainsi hypothéquer un acte médical d'urgence ou engager le pronostic vital des patients alors qu'un surstockage peut engendrer non seulement des coûts inutiles, mais aussi la perte de produits par péremption.

I.4.1 Importance de la gestion des stocks

Dans un établissement hospitalier, les produits de santé représentent souvent une part importante du budget global. Leur mauvaise gestion entraîne des impacts majeurs :

Des **ruptures critiques**, affectant directement la prise en charge des patients, surtout en réanimation, urgences ou blocs opératoires,

Des **pertes financières** dues à la péremption, la surconsommation ou le stockage inadapté,

Une **désorganisation des services** médicaux, source de mécontentement pour les patients et le personnel soignant,

Une gestion performante permet :

D'éviter les interruptions de soins dues à une indisponibilité de produit pharmaceutique,

De réduire les pertes causées par la péremption, les erreurs humaines ou l'obsolescence des produits,

D'optimiser les ressources en adaptant les volumes de commande aux besoins réels (consommation moyenne mensuelle, urgences, prévisions saisonnières) [1].

I.4.2 Etapes de gestion des stocks d'une pharmacie hospitalière Circuit des PP

Dans le secteur hospitalier, le processus de gestion des stocks est substantiellement semblable à celui suivi dans d'autres types de chaînes logistiques que sont principalement celles ayant pour fonction d'obtenir des approvisionnements (l'approvisionnement), de conserver ces derniers (l'entreposage) et de les mettre à disposition des services utilisateurs (la distribution / la dispensation). Comme cela a été précisé, le fonctionnement d'une pharmacie hospitalière répond à des exigences de réalisation spécifiques à la nature des produits de santé selon leur traçabilité (DM implantables, médicaments et psychotropes), d'une part et/ou leur nécessité clinique en termes d'urgence, d'autre part.

L'approvisionnement vise à garantir la disponibilité permanente des produits pharmaceutique nécessaires aux soins. Cette étape inclut :

L'identification des besoins réels,

La planification des achats selon un programme annuel, trimestriel et mensuel,

L'établissement de bons de commande conformes aux procédures en vigueur (marchés publics, appel d'offres),

La réception des produits avec un contrôle quantitatif et qualitatif (conformité, date de péremption, numéro de lot),

L'enregistrement dans le registre et/ou système informatique de gestion [13].

Durant notre stage à la pharmacie hospitalière du CHU de Tizi-Ouzou, nous avons appris que l'approvisionnement des hôpitaux publics en produits pharmaceutiques se fait principalement par une livraison mensuelle de la Pharmacie Centrale des Hôpitaux (PCH). Pour les dispositifs médicaux et réactifs, des appels d'offres sont lancés annuellement à échelle nationale et les marchés sont attribués selon le principe du mieux-disant pour le meilleur rapport qualité/prix.

L'entreposage Une fois les produits réceptionnés, ils sont entreposés dans des conditions adaptées de sécurité et de conservation. L'entreposage implique :

Une organisation des zones de stockage selon les classes thérapeutiques et les types de produits,

Le respect des conditions environnementales (température, humidité, sécurité incendie),

Une gestion informatisée ou manuelle des mouvements pour assurer un inventaire permanent fiable [13].

La distribution consiste en une traçabilité systématique et minutieuse de toutes les sorties de stock vers les différents services hospitaliers, accompagnée de la gestion des situations urgentes et de la priorisation des produits selon la criticité des besoins cliniques.

Lors de notre stage en pharmacie hospitalière au CHU de Tizi-Ouzou, on a compris que les sorties de produits vers les services ont lieu chaque semaine généralement à partir de bons de commande internes émis par les chefs de service.

Chaque service est doté pour chaque produit pharmaceutique sur la base de ses besoins attendus et de son historique de consommation.

Nous avons également pu noter l'existence de sorties ponctuelles sur ordonnance interne : dans ce cas, un médecin prescrit pour un patient un produit particulier, que la pharmacie délivre alors de manière nominative.

Ces pratiques sont essentielles pour garantir la continuité des soins et limiter les risques liés aux ruptures de stock.

I.5. Applications et logiciels utilisés en biomédical

I.5.1G-stock-pharma

I.5.1.1 Présentation générale

G-stock-pharma est un logiciel de gestion d'entrepôt (WMS) développé par la société française **Electroclass** (spécialisée dans les systèmes automatisés de stockage et de distribution) destiné aux pharmacies hospitalières, cette application est principalement utilisée pour assurer l'optimisation du suivi des produits (entrées et sorties), la gestion des commandes et l'efficacité opérationnelle des pharmacies et distributeurs de médicaments [10].

I.5.1.2 Fonctionnalités principales

Gestion complète des stocks en temps réel avec une alerte pour les ruptures et l'approche des dates de péremption,

Suivi des lots et dates de péremption avec possibilité de retrait, transfert ou mise en quarantaine,

Automatisation des commandes et réapprovisionnements,

Traçabilité des mouvements de stock,

Analyses des consommation grâce aux rapport er statistiques fourni,

Interface utilisateur adaptée aux pharmaciens hospitaliers et totalement personnalisable [10][11].

I.5.1.3 Technologies utilisées

G-stock-pharma s'appuie sur des systèmes de traçabilité par code-barres intégrés dans ses armoires automatisées, associés à un logiciel de gestion de stocks hospitaliers [10][11]. Et concernant la technologie précise de leurs bases de données, elle n'est pas accessible aux publiques.

I.5.1.4 Avantages

Gain en temps : réduction de temps consacré à la gestion manuelle des stocks,

Traçabilité optimale et sécurisée : permet un suivi rigoureux des médicaments pour assurer la sécurité des patients,

Reduction des pertes : minimisation des pertes dues aux produits périmés ou en rupture ;

Gestion centralisée multi-sites : simplifie la gestion des stocks dans plusieurs entrepôts avec un seul outil,

Réapprovisionnement automatisé : réduit les risques des ruptures de stock et améliore la disponibilité des médicaments,

Accessibilité 24 /7 via self-service : cela augmente l'efficacité dans la distribution des médicaments,

Intégration facile avec les systèmes existants : facilite la synchronisation et l'adoption avec tout autres logiciels de gestion [10][11].

I.5.1.5 Limites

Concernant les limites de G-stock-pharma on n'en trouve pas vraiment de sources fiables sauf dans des revues utilisateurs de logiciels hospitaliers, comme sur Capterra.fr ou hospitalia.fr. Parmi ces limites on trouve :

Coût élevé : peut représenter un investissement significatif aux établissements hospitaliers,

L'interface utilisateur est parfois jugée peu intuitive ce qui nécessite une formation spécifique,

Parfois l'intégration est complexe avec certains logiciels hospitaliers plus anciens.

I.5.2 Epocrates

I.5.2.1 Présentation générale

Epocrates est une application mobile et un service web développée pour les professionnels de santé afin de pouvoir accéder aux informations cliniques des patients. Elle a été lancée en 1998 et elle est devenue une ressource médicale complète utilisée par plus d'un

million de cliniciens dans le monde, disponible sur diverses plateformes aidant les médecins, pharmaciens, assistants médicaux et autres à prendre des décisions éclairées sur : les médicaments et leurs interactions, les diagnostiques et traitements tout en améliorant la prise de décision clinique et la gestion des soins [12][14].

I.5.2.2 Fonctionnalités principales

Informations détaillées sur des milliers de médicaments avec posologie, contre-indications, effets secondaire…etc,

Calculateurs médicaux : plus de 600 calculateurs intégrés pour le diagnostic et la prise de décision clinique,

Identification de pilules (Pill ID) par caractéristiques physiques : forme, couleur et gravure,

La vérification des interactions médicamenteuses entre plusieurs traitements prescrits, avec la possibilité d'analyser jusqu'à 30 médicaments à la fois afin de garantir la sécurité des combinaisons,

Contenu orienté patient : informations destinées à améliorer l'éducation des patients sur leurs traitements,

Outils de décision clinique : avec des tables de références pour adultes et enfants, guide rapide sur diverses pathologies et protocoles de traitement [12].

I.5.2.3 Technologies utilisées

Les détails techniques spécifiques ne sont pas publiquement disponibles, mais Epocrates fonctionne sur les plateformes iOS et Android ainsi via une interface web [13]. Et concernant l'architecture technique exacte n'est pas publiée, mais l'usage d'une base de données relationnelle est très probable pour la gestion des données cliniques.

I.5.2.4 Avantages

Référence rapide et fiable : ça contient des informations claires, concises et facilement accessibles et surtout adaptés aux besoins des professionnels en situation de soin,

Large éventail d'outils cliniques : que ça soit de la pharmacologie, les interactions, identifications de pilules, guide clinique…etc,

Fiabilité des données : l'application propose un contenu vérifié par des experts médicaux,

Possibilité d'utilisation hors ligne : accès aux données même sans réseau ;

Mise à jour régulière : contenu médical constamment actualisé afin de garantir la fiabilité des informations [14].

I.5.2.5 Limites

Ces points sont fréquemment rapportés par les utilisateurs notamment sur les stores et forums professionnels, où on pourra citer :

Support client insuffisant : plusieurs utilisateurs rapportent un service client peu réactif et difficilement joignable lors de la connexion ou le transfert d'un compte,

Interface parfois surchargée : trop d'informations affichées simultanément peuvent rendre la navigation difficile,

Fonctionnalités premium payantes : certaines fonctions comme les guides complets de traitement sont réservées à la version payante [13].

I.5.3 Hospilog

I.5.3.1 Présentation générale

Hospilog est un logiciel de WMS (Warehouse Management System) spécialisé dans la gestion logistique hospitalière, développé par KLS Group un éditeur spécialisé dans les solutions de gestion d'entrepôt et d'optimisation des flux logistiques dans le secteur de santé. Il supervise l'ensemble des flux physiques et des caractéristiques associées à différents types de métiers et produits hospitaliers, notamment l'hôtellerie, la pharmacie, les blocs opératoires, la stérilisation…etc.

Il s'intègre avec la Gestion Economique et Financière (GEF) et d'autres logiciels pour assurer une gestion unifiée et une traçabilité fine des produits et prescripteurs [15][16].

I.5.3.2 Fonctionnalités principales

Gestion multi-sites et évolutive, adaptée aux Groupements Hospitaliers de Territoire (GTH) et platformes logistiques multi-sites,

Traçabilité fine des produits incluant la gestion des lots, dates de péremption et stérilisation,

Interface avec équipements automatisés (stockeurs, robots de dispentation) et outils d'identification (codes-barres, RFID),

Automatisation des réapprovisionnements avec alertes sur seuils critiques,

Module Web E-Order pour la dématérialisation des commandes et échanges internes/externes,

Suivi en temps réel les stocks et statistiques d'activité, pour faciliter l'analyse et la prise de décision [15][16][17].

I.5.3.3 Technologies utilisées

L'utilisation de technologies d'identification automatique : codes-barres, RIFD,

L'intégration avec les équipements automatisés de stockage et dispensation (les armoires sécurisées, automates de dispensation, classeurs rotatifs…etc),

Module web pour la gestion dématérialisée des commandes (Web E-Order) [15][16][17].

I.5.3.4 Avantages

Amélioration de la traçabilité des produits et des prescripteurs, tout en réduisant les erreurs et pertes,

Facilitation de la gestion multi-sites et des plateformes logistiques complexes,

Gain de temps grâce à l'automatisation et dématérialisation des échanges,

Conforme aux normes réglementaires [15][16].

Fiabilité éprouvée dans plusieurs établissements comme le CHU de Renne, avec un volume important de commandes et livraisons gérées [17].

I.5.3.5 Limites

Complexité de déploiement : exigeant en configuration SIH, formation et adaptation des processus ;

Coût élevé : l'installation, l'intégration et la maintenance sont coûteuses surtout les équipements automatisés,

Adaptation local nécessaire : il faut une personnalisation pour s'intégrer aux SIH existants [17][18].

I.5.4 Winpharma

I.5.4.1 Présentation générale

Winpharma est un logiciel de gestion actif depuis plus de 30ans, conçu spécifiquement pour les officines pharmaceutiques. Il permet de simplifier et automatiser l'ensemble de tâches quotidienne liées à la gestion des pharmacies, afin de maintenir un approvisionnement optimal de ces dernières [19].

I.5.4.2 Fonctionnalités principales

Gestion complète des officines : ça consiste en tout ce qui est traitement des ordonnances, gestion des ventes, facturation, suivi des rendez-vous et dossiers patients,

Gestion des stocks : c'est un suivi en temps réel avec alertes automatiques sur les ruptures, suggestion des commandes, et optimisation des approvisionnements,

Gestion des interactions médicamenteuses : c'est une consultation intégrée pour assurer la sécurité des patients,

Win Prescription et win Facture : ce sont des innovations récentes visant à accélérer et sécuriser la délivrance et la facturation,

Win Autopilote : ça consiste à l'automatisation des achats permettant de générer, d'envoyer et de réceptionner les commandes en seulement 5 minutes, tout en réduisant les erreurs ou stocks excessives et manquants,

Win Team : c'est un cahier de liaison dématérialisé intégré, qui facilite la communication et le partage d'informations au sein de l'équipe officinale,

Application mobile MOBIPHARM : C'est un outil nomade dédié aux pharmacies (officines ou PUI), qui permet de gérer et suivre l'activité de ces dernières à distance. Grace à la lecture de code-barres et data matrix (via l'appareil photo ou lecteur associé), l'application permet :

- La réception et rangement automatisés des médicaments ;
- La dispensation fluide en validant les plans de cueillette depuis le mobile ;
- La mise à jour en temps réel des stocks et inventaires, le tout via une interface ergonomique responsive et synchronisée avec le logiciel principal Winpharama [19].

I.5.4.3 Technologies utilisées

Winpharma fonctionne sous environnement Windows, ce qui garantit une large compatibilité avec les systèmes informatiques des officines. La version récente Winpharma 9, existe avec des interfaces modernes et ergonomiques, avec une architecture qui permet l'intégration automatique des données patients via ADRi et la connexion aux systèmes de santé.

Les innovations les plus récentes (winPrescription, winFactures) démontrent une orientation vers des processus automatisés et simplifiés, bien que les détails techniques précis reste peu accessible au public [19].

I.5.4.4 Avantages

Facile à utiliser et rapide à déployer : vente OTC en 3 secondes, interface utilisateur intuitive et ergonomique, facilité de navigation,

Automatisation avancée des achats : le temps d'auto gestion des commandes et fortement réduit et les achats sont optimisés en temps réel, avec un impact positif sur la trésorerie,

Propriété exclusive des données : les pharmaciens restent maîtres de leurs données sans interférence dans leur politique d'achat,

Faciliter la communication en équipe : grâce à winTeam, qui permet une collaboration améliorée même à distance,

Support client très réactif et à l'écoute : de très forts taux de satisfaction exprimée par les utilisateurs, la plupart félicitant l'équipe technique qui a su intervenir rapidement,

Logiciel co-construit avec les pharmaciens : qui favorise l'adéquation aux besoins réels du terrain avec une innovation continue,

Respect des normes : intégration automatique des droits patients et fiabilisation des remboursements SESAM-Vitale (système informatique de télétransmission mis en place par l'Assurance Maladie française, pour permettre aux professionnels de santé la transmission automatiques des feuilles de soins et l'obtention de remboursement via la carte vitale du patient),

Adapté à tous types de pharmacies [19][20].

I.5.4.5 Limites

Les retours négatifs des utilisateurs sont très peu visibles, ce qui reflète un manque de publications critiques.

I.5.5 Epipharm

I.5.5.1 Présentation générale

Epipharm est un logiciel médico-administratif développer en 1994, conçu pour gérer automatiquement les stocks et la distribution des médicaments au sein des pharmacies hospitalières [21]. Il est utilisé notamment dans des centres hospitaliers universitaires tel que **CHU de Tizi Ouzou** ou il a été fourni par le **ministère de la santé et de la réforme hospitalière**.

I.5.5.2 Fonctionnalités principales

Les fonctionnalités suivantes sont celles observées lors de notre stage au CHU Nedir Mohamed de Tizi-Ouzou en sain de leur pharmacie hospitalière, ce qui reflète l'usage réel du logiciel sur le terrain :

Représentation de chaque médicament ou dispositif et chaque laboratoire par un code spécifique,

Suivi des stocks en réalisant un compte de gestion,

Suivi des produits approvisionnés tout en surveillant leurs quantités et dates de péremption,

Distribution planifiée en respectant les dotations pour chaque service hospitalier,

Gestion des ordonnances internes pour les patients hospitalisés,

Suivi en temps réel du mouvement des stocks,

Effectuer les décharges et retour des produits entre établissements,

Effectuer les prévisions d'approvisionnement,

Alertes pour les dates de péremptions et ruptures de stock.

I.5.5.3 Technologies utilisées

L'Epipharm est basé sur une architecture client-serveur, compatible avec les normes locales de santé [21].

I.5.5.4 Avantages (observés durant le stage)

L'automatisation des taches ce qui permet de gagner du temps et traiter plusieurs commandes à la fois,

Répartition équitable des produits par quota entre services,

Prévention des pertes grâce aux alarmes qui avertissent l'approche des dates de péremption et des seuils critiques,

Réduction du surstockage.

I.5.5.5 Limites (observés durant le stage)

Absence de la possibilité de modifier en cas d'erreur de frappe ce qui pousse à tout refaire à zéro,

Absence de connexion intranet avec les services qui permettra de plus faciliter le travail,

Le logiciel ne permet pas le calcul exact des couts généraux,

Le non suivi des marchés privés.

*Comparaison et analyse critique des Applications/Logiciels biomédicaux

Dans le cadre de l'étude et du développement d'applications biomédicales orientées vers la gestion hospitalière, il est important de comparer les solutions existantes afin d'identifier leurs points forts, leurs limites et les opportunités d'amélioration. Voici une analyse critique comparative de plusieurs logiciels biomédicaux déjà utilisés dans le domaine hospitalier et pharmaceutique.

Tableau I-1 : Comparaison entre des Applications/Logiciels biomédicaux.

Application	Type	Avantage	Limites	Public cible	Technologie
G-Stock Pharma	Gestion de stock pharmaceutique	Traçabilité, automatisation des commandes, accessibilité 24/7	Coût élevé, interface peu intuitive, intégration complexe	Pharmacies hospitalières	Base de données relationnelle (SQL, Oracle), code-barres
Epocrates	Support à la décision	Fiabilité, large éventail d'outils,	Interface surchargée,	Médecins, pharmaciens,	Bases relationnelles,

Application	Type	Avantage	Limites	Public cible	Technologie
	clinique	usage hors ligne	support client faible, options payantes	cliniciens	plateformes iOS/Android
Hospilog	Gestion logistique hospitalière (WMS)	Traçabilité fine, gestion multi-sites, automatisation des flux, intégration équipements (robots, codes-barres)	Complexité de déploiement, coût élevé, adaptation SIH nécessaire	Hôpitaux, PUI, plateformes logistiques	Technologies RFID, codes-barres, module Web E-Order, intégration GEF
Winpharma	Gestion d'officine pharmaceutique	Automatisation des achats, support client, interface intuitive	Peu de détails techniques publiés, pas de version cloud, peu d'IA	Officines	Environnement Windows, intégration ADRi
Epipharm	Logiciel médico-administratif	Automatisation, alertes péremption, répartition équitable des dotations	Pas de modification d'erreurs, absence d'intranet, pas de calcul des coûts généraux	Pharmacies hospitalières algériennes	Architecture client-serveur compatible avec SI locaux

L'étude comparative exposée jusqu'à présent aide ainsi à mieux identifier les atouts spécifiques de chaque candidat applicatif au regard de son domaine d'utilisation. Elle permet également d'identifier les manques techniques, ergonomiques ou économiques auxquels les concepteurs seront amenés à apporter une réponse. Cette étude permettra en outre de réfléchir aux fonctionnalités à intégrer dans la conception de toute nouvelle solution proposée pour le milieu hospitalier, en particulier algérien, mais aussi à celles à éviter.

I.6 Conclusion

Ce premier chapitre nous a permis de cerner en profondeur les enjeux liés à la gestion des produits pharmaceutiques au sein d'une pharmacie hospitalière. Nous avons identifié les différents types de produits gérés, le rôle stratégique de la pharmacie dans le circuit du médicament, ainsi que les étapes critiques du processus de gestion, depuis l'approvisionnement jusqu'à la distribution.

L'étude des logiciels biomédicaux a mis en lumière les avantages indéniables que ces outils permettent d'offrir en matière de traçabilité, de gain de temps et de sécurité des approvisionnements. Toutefois, ces solutions présentes aussi des limites fonctionnelles, ergonomiques ou économiques, révélant un décalage persistant entre les fonctionnalités proposées et les besoins réels du terrain hospitalier, notamment dans le contexte algérien.

Ces constats justifient pleinement la nécessité de concevoir une solution adaptée, prenant en compte les spécificités locales, les contraintes de terrain et les besoins des professionnels de santé. Ce chapitre pose ainsi les fondations théoriques et pratiques nécessaire afin d'envisager la conception d'un outil numérique plus efficace, fiable et intégré.

Chapitre II

Les fondements des bases de données

Chapitre II : Les fondements des bases de données

II.1 Introduction

Les bases de données représentent l'un des piliers fondamentaux des systèmes d'informatique modernes. Elles permettent de stocker et de gérer de manière efficace des volumes importants de données structurées ou non structurées. Dans le cadre du développement d'applications informatiques, notamment les applications web, elles assurent la conservation des informations échangées entre les utilisateurs et les serveurs de façon cohérente et organisée.

Ce chapitre examine les diverses formes de bases de données, les modèles de conception tel que MERISE, ainsi que les technologies clés comme : MySQL, PostgreSQL ou ORACLE qui sont largement utilisées dans le domaine informatique actuel pour établir des bases solides nécessaires à la création d'applications dynamiques, sécurisées et performantes.

II.2 Concepts fondamentaux des bases de données

II.2.1 Définition et utilité

Une base de données est une structure électronique qui permet de stocker, de trier et d'organiser des informations (textes, nombres ou dates), elle est conçue pour être facilement accessible et constamment mise à jour tout en assurant une gestion sécurisée et rapide de ces données. Elle est essentielle dans plusieurs domaines tel que : l'informatique, la finance, la médecine…etc [22] [23].

Elle est principalement utilisée pour conserver, gérer et récupérer les données d'une manière efficace. Et pour cela on utilise un système de gestion de base de données (SGBD) qui se fait en langage SQL (Structured Query Language), où il existe plusieurs sur le marché mais parmi les plus connus on trouve : ORACLE, MySQL, Microsoft SQL Server et PostgreSQL [23]

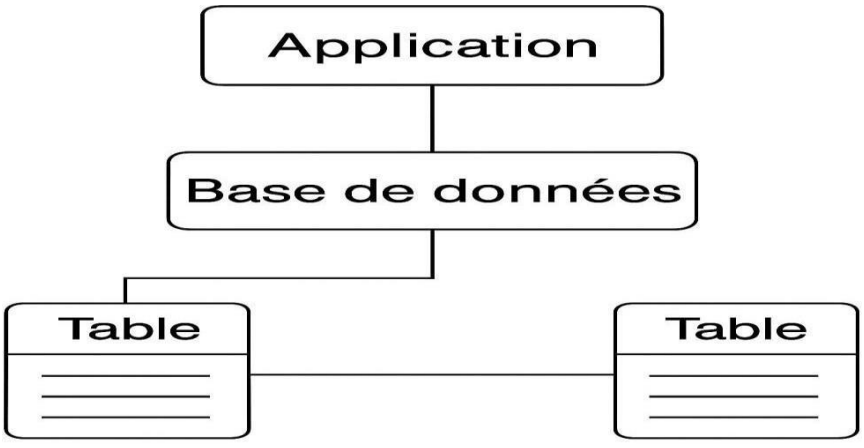

Figure II-1 : Illustration simplifiée du rôle d'une base de données dans un système informatique.

II.2.2 Données vs informations

Les données sont des faits bruts et non traités : des chiffres ou des textes recueillis par des observations d'expérience ou de mesures. Elles sont souvent sous forme binaire (0 ou 1), stockées dans un système informatique, et constituent la matière première utilisée pour traiter les informations. On trouve plusieurs formes de données tel que : les chiffres, les textes, les images, les audios et les vidéos [24][25].

Cependant, les informations sont la forme organisée et traitée des données brutes tel que : les rapports, les résumés et les graphiques...etc. elles sont utiles pour la prise de décision. En d'autres termes l'information est le traitement de n'importe quelle donnée avec des méthodes spécifiques, afin d'obtenir une nouvelle connaissance prête à être utilisée [24].

Figure II-2 : Transformation des données brutes en information utilisable via un traitement spécifique.

II.2.3 Types de bases de données

II.2.3.1 Bases relationnelles (SQL)

Une base de données relationnelle est composée de tableaux à 2 dimensions, elle stocke les données sous forme de tables où chaque table est composée de colonnes (champs) et de lignes (entrées) permettant des relations logiques entre différentes entités grâce à des clés primaires et étrangères. Elle utilise le langage SQL pour la gestion des données et elle est idéale pour les entreprises qui manipulent des données complexes [26].

II.2.3.2 Bases hiérarchiques

Une base de données hiérarchique organise les données sous forme d'un arbre, où chaque enregistrement n'a qu'un seul possesseur. Ce modèle est adapté aux classifications strictes comme les systèmes de fichiers, cependant à cause de ces limitations internes elle ne peut pas être utilisée pour décrire des relations complexes ou multiples entre données [27].

II.2.3.3 Bases orientées objets

Les bases de données orientées objets stockent les données sous forme d'objets, qui regroupent les informations et les méthodes associées. Elles modélisent les données selon les concepts de la programmation orientée objet (POO), ce type de base de données est adapté aux applications complexes qui manipulent des objets métiers et leurs relations [26] [28].

II.2.3.4 Bases non-relationnelles (NoSQL)

Les bases de données NoSQL sont conçues pour gérer des données non structurées ou semi-structurées. Elles n'utilisent ni clé primaire ni clé étrangère pour mettre en relation plusieurs tables, ce qui apporte une flexibilité très intéressante pour les applications modernes comme les réseaux sociaux [29].

Il existe 4 types de bases NoSQL :

Bases de données orientées documents : stockent les données sous forme de documents, souvent au format JSON ;

Bases de données clé-valeur : utilisent une paire clé-valeur pour le stockage des données ;

Bases de données orientées colonnes : stockent les données par colonnes tout en optimisant les requêtes analytiques ;

Bases de données orientées graphes : modélisent les données sous forme de graphes [30].

II.3 Modélisation des données

II.3.1 Cycle de vie d'une base de données

Le cycle de vie d'une base de données contient un ensemble d'étapes par lesquelles passent les données, depuis leur création jusqu'à leur suppression. Pour garantir la qualité, conformité et la sécurité des données il faut établir une bonne gestion de ce cycle [31].

Les étapes essentielles de ce cycle :

La collecte des données : la 1ère étape consiste à recueillir les données nécessaires, tout en respectant la réglementation notamment l'obtention du consentement des personnes concernées. Ces données peuvent être importées depuis une source externe (acquisition), saisies manuellement par l'utilisateur ou capturées automatiquement via les dispositifs numériques de l'entreprise [31].

Le stockage des données : une fois collectées les données doivent être stockées de manière sécurisée afin de garantir leur disponibilité et leur protection, et cela peut se faire sur différents supports comme les serveurs, le cloud ou le disque dur [31].

Le traitement des données : les données brutes ne peuvent pas être exploitables immédiatement, elles doivent d'abord être compressées, cryptées ou nettoyées [31].

L'analyse des données : une fois traitées les données sont analysées pour extraire des informations utiles [31].

La sauvegarde des données : des copies de sécurité sont créées et stockées ailleurs pour éviter les pertes, elles peuvent être totales ou partielles [31].

La réutilisation des données : les données peuvent être réutilisées dans de nouveaux contextes, et pour cela il faut une bonne organisation du système d'information afin d'assurer un accès fluide et rapide [31].

La suppression des données : à la fin du cycle lorsque les données deviennent obsolètes ou inutiles, elles doivent être détruites de manière sécurisée en fonction de leur sensibilité [31].

II.3.2 Méthodologie MERISE

La méthodologie de MERISE est une approche structurée de modélisation des systèmes d'information, largement utilisée en France [32]. Cette méthode permet d'assurer une conception méthodique et progressive des bases de données, elle repose sur une modélisation en trois niveaux :

II.3.2.1 Modèle Conceptuel de Données (MCD)

Le MCD représente les données de manière indépendante des contraintes techniques ou organisationnelles, sous forme d'un schéma entité-association compréhensible par tous (informaticiens et utilisateurs) [33].

II.3.2.2 Modèle Logique de Données (MLD)

Le MLD permet de traduire le MCD en un modèle adapté à la technologie choisie, tout en tenant compte des règles d'utilisation. Ce modèle prépare l'implémentation en définissant les tables, les clés primaires et étrangères et les contraintes d'intégrité nécessaires afin d'assurer la cohérence des données [33].

II.3.2.3 Modèle Physique de Données (MPD)

Le MPD concrétise le MLD dans un système de gestion de base de données spécifique, il détaille les aspects physiques tels que : les types de données, les ailles des champs, les index ainsi que les vues et les procédures stockées.

Ce découpage permet la séparation des préoccupations fonctionnelles, logiques et techniques, tout en facilitant la conception et la maintenance des bases de données [33].

II.3.3 Modélisation UML

La modélisation UML (Unified Modeling Language) ou (Langage de Modélisation Unifié) est une approche orientée objet qui peut être utilisée pour modéliser les données sous forme de diagrammes de classe. Ce type de diagrammes permet de spécifier les entités, leurs attributs et les associations afin de faciliter la transition vers une implémentation relationnelle [34] [35].

Elle est souvent employée en complément ou comme alternative à MERISE, faut juste retenir que l'UML n'est pas un langage de programmation, mais il existe des outils qui peuvent être utilisés pour générer du code en plusieurs langages à partir du diagramme UML [34].

II.4 Les systèmes de gestion de base de données (SGBD)

II.4.1 Définition et rôles d'un SGBD

Le SGBD est un logiciel qui permet de stocker, de récupérer, d'ajouter, de modifier et de supprimer les données dans une base de données de manière efficace et sécurisée tout en facilitant la manipulation et la gestion structurée des données. Il fonctionne comme un intermédiaire entre les utilisateurs, les applications et la base de données physique. Le SGBD permet de définir et de gérer le schéma des données dont elles sont stockées [22] [36].

Un SGBD doit essentiellement assurer ces fonctions indispensables à la gestion efficace des données :

La cohérence et l'intégrité des données : via des contraintes (clé primaire, clé étrangère…etc) ;

La sécurité d'accès aux données : en limitant l'accès selon le privilège des utilisateurs ;

L'organisation et la gestion des données : afin de garantir l'intégrité, la cohérence et la qualité des données stockées, tout en facilitant la manipulation et la récupération des informations ; La sauvegarde et la restauration en cas de panne [36].

II.4.2 Les principaux SGBD utilisés

Il existe plusieurs types de SGBD sur le marché, chacun ayant ses spécificités selon les besoins, et parmi les plus connus on trouve :

II.4.2.1 ORACLE

C'est un SGBD propriétaire développé par Oracle Corporation, connu pour ses fonctionnalités avancées, ses méthodes fiables de récupération des données et sa haute sécurité. Il est couramment utilisé dans les grandes entreprises et systèmes critiques, son rôle est de prendre en charge les bases relationnelles, les partitions et les fonction analytique…etc [22].

II.4.2.2 MySQL

C'est un SGBD relationnel open-source très utilisé, notamment pour les applications web. Il est apprécié pour sa simplicité, ses performances, sa rapidité et compatibilité avec le PHP. Il prend en charge le langage SQL standard et est maintenu par Oracle [37].

II.4.2.3 Microsoft SQL Server

C'est un SGBD relationnel robuste qui offre des outils d'analyse intégrés, des services de reporting, une intégration poussée avec les solutions Microsoft et une interface utilisateur conviviale [38].

II.4.2.4 PostgreSQL

C'est un SGBD relationnel et objet open-source, connu pour sa robustesse, sa conformité aux standards SQL et sa capacité de gérer des données complexes (comme JSON) [39].

II.4.3 Fonctionnalités principales d'un SGBD

II.4.3.1 Création et gestion des tables

Les SGBD permettent de définir la structure logique des données via la création de tables, chaque table contient des colonnes (champs) et des lignes (enregistrements). L'utilisateur peut définir la table par un schéma précisant les types de données, les contraintes (clé primaire, unique, non-nulle) et les relations avec d'autres tables via des clés étrangères [40].

La création et la modification des tables s'effectuent généralement avec le langage SQL, par exemple : avec la commande CREATE TABLE on crée une nouvelle table, et avec ALTER TABLE on modifie une table déjà existante [40].

II.4.3.2 Requêtes et manipulation des données

Les SGBD offrent des fonctionnalités complètes CRUD (Create, Read, Update, Delete) à travers des requêtes SQL, afin de manipuler les données :

Insertion (Create) : ajouter de nouvelles données dans les tables,

Lecture (Read) : interrogation des données selon des critères précis, avec tri, jointures…etc,

Mise à jour (Update) : faire des modifications dans des données existantes,

Suppression (Delete) : la suppression des données obsolètes ou incorrectes [40].

II.4.3.3 Gestion des utilisateurs et des droits d'accès

La sécurité des données est la priorité des SGBD permettent de gérer les utilisateurs et leurs privilèges, afin de contrôler qui peut accéder à quelles données et quelles opérations sont autorisées à effectuer (lecture, écriture, modification…etc). Les mécanismes d'authentification et d'autorisation garantissent la sécurité des données et la traçabilité des actions [22] [37].

II.5 Langage SQL (Structured Query Language)

Le langage SQL est un langage standard qui permet d'accéder à des bases de données et de les manipuler. Il permet d'effectuer plusieurs opérations de manipulation de données (exécution des requêtes, récupération des données, insertion, modification, suppression…etc) [41].

II.5.1 Requêtes de base

Les requêtes de base permettent de manipuler les données de manière efficace :

II.5.1.1. SELECT

Permet de récupérer des données depuis une ou plusieurs tables [41].

```sql
SELECT nom, prenom FROM clients;
```

Figure II-3 : Exemple de la requête SELECT en langage SQL.

II.5.1.2. INSERT

Permet d'ajouter de nouvelles lignes dans une table [41].

```sql
INSERT INTO clients (nom, prenom) VALUES ('Dupont', 'Jean');
```

Figure II-4 : Exemple de la requête INSERT en langage SQL.

II.5.1.3. UPDATE

Permet de modifier des données déjà existantes dans une table [41].

```sql
UPDATE clients SET prenom = 'Jean-Pierre' WHERE nom = 'Dupont';
```

Figure II-5 : Exemple de la requête UPDATE en langage SQL.

II.5.1.4. DELETE

Permet de supprimer des lignes d'une table [41].

```sql
DELETE FROM clients WHERE nom = 'Dupont';
```

Figure II-6 : Exemple de la requête DELETE en langage SQL.

II.5.2 Création et modification de structures

Le SQL permet aussi de définir et de modifier la structure des tables :

II.5.2.1. CREATE TABLE

Pour créer une nouvelle table avec des colonnes spécifiées [41]

```sql
CREATE TABLE clients (
    id INT PRIMARY KEY,
    nom VARCHAR(50),
    prenom VARCHAR(50)
);
```

Figure II-7 : Exemple de la création d'une table en langage SQL.

II.5.2.2. ALTER TABLE

Pour modifier la structure d'une table existante, exemple : ajouter une colonne [41].

```sql
ALTER TABLE clients ADD email VARCHAR(100);

```

Figure II-8 : Exemple de modification d'une table en langage SQL.

II.5.2.3. DROP TABLE

Pour supprimer une table existante et toutes ses données [41].

```sql
DROP TABLE clients;

```

Figure II-9 : Exemple de suppression d'une table en langage SQL.

II.5.3. Contraintes d'intégrité

Les contraintes d'intégrité permettent d'assurer la cohérence et la validité des données dans une base de données :

II.5.3.1. Clé primaire (PRIMARY KEY)

Elle permet d'identifier de manière unique chaque ligne d'une table, et elle ne peut pas contenir des valeurs nulles [41].

```sql
CREATE TABLE clients (
    id INT PRIMARY KEY,
    nom VARCHAR(50)
);

```

Figure II-10 : Exemple d'utilisation d'une clé primaire en langage SQL.

II.5.3.2. Clé étrangère (FOREIGN KEY)

Elle établit une relation entre deux tables en liant une colonne à la clé primaire d'une autre table, tout en empêchant la création de données orphelines [41].

```sql
CREATE TABLE commandes (
    id INT PRIMARY KEY,
    client_id INT,
    FOREIGN KEY (client_id) REFERENCES clients(id)
);

```

Figure II-11 : Exemple d'utilisation d'une clé étrangère en langage SQL.

II.5.3.3. Contraintes d'unicité (UNIQUE)

Garantissent que toutes les valeurs d'une colonne sont uniques, mais peuvent autoriser une valeur nulle [41].

```sql
ALTER TABLE clients ADD CONSTRAINT unique_email UNIQUE (email);
```

Figure II-12 : Exemple d'utilisation de la contrainte UNIQUE en langage SQL.

II.5.3.4. Contraintes de nullité (NOT NULL)

Elles empêchent l'insertion de valeurs nulles dans une colonne [41].

```sql
ALTER TABLE clients MODIFY nom VARCHAR(50) NOT NULL;
```

Figure II-13 : Exemple d'utilisation de la contrainte NOT NULL en langage SQL.

II.5.4 Optimisation des requêtes

II.5.4.1 Hachage

Le hachage est une technique d'indexation utilisée en interne par certains moteurs de bases de données. Il consiste à appliquer une fonction de hachage sur une clé pour déterminer l'emplacement où stocker ou bien pour retrouver une donnée, permettant ainsi d'accélérer la recherche dans les tables.

En SQL, le hachage est souvent utilisé dans les algorithmes de jointure, notamment la jointure de hachage comme : **hash join,** qui est efficace pour joindre de grandes tables en mémoire [42].

II.5.4.2 Index

Un index en SQL est une structure de données qui permet d'améliorer la vitesse des opérations de récupération de données sur une table. Il fonctionne en créant une structure séparée qui contient la valeur des colonnes indexées et des pointeurs vers les lignes correspondantes dans la table [43].

Il existe plusieurs types d'index :

Index B-tree: c'est l'index par défaut dans la plupart des SGBD relationnels (PostgreSQL, MySQL ou Oracle), utilisé pour des recherches rapides sur les colonnes triées ;

Index de hachage : moins courant et souvent spécifique à certains SGBD comme PostgreSQL. Il est utilisé pour des recherches rapides sur des valeurs exactes ;

Index cluster : dans SQL Server, il est utilisé pour déterminer l'ordre physique des données

Index non-cluster : utilisé pour créer des index supplémentaires sur une table, indépendants de l'ordre physique [43].

II.5.4.3 Les vues (VIEW)

Une vue est une table virtuelle créée à partir d'une requête SQL, elle permet de simplifier l'accès aux données complexes, de masquer certaines colonnes ou de filtrer les lignes selon des critères spécifiques. L'utilisation des vues permet d'améliorer les performances mais aussi elle facilite la réutilisation et la sécurité des données [44].

```sql
CREATE VIEW vue_ordonnances AS
SELECT p.nom, o.medicament, o.date_prescription
FROM patients p
JOIN ordonnances o ON p.id_patient = o.id_patient;
```

Figure II-14 : Exemple de la création d'une vue en langage SQL.

II.5.4.4 Les sélections (SELECT)

Les clauses de sélections avec WHERE, LIMIT ou DISTINCT permettent de restreindre le nombre de lignes traitées dans une requête. Cela réduit le volume de données à analyser et permet d'optimiser les performances [39] [41].

```sql
SELECT medicament, COUNT(*) AS total_prescriptions
FROM ordonnances
WHERE date_prescription >= '2025-01-01'
GROUP BY medicament
HAVING COUNT(*) > 5;
```

Figure II-15 : Exemple d'une sélection avec WHERE en langage SQL.

II.5.4.5 Les jointures (JOIN)

Les jointures en SQL permettent de combiner les lignes de deux ou plusieurs tables en fonction d'une condition commune, souvent une clé étrangère [41].

Les types de jointures les plus utilisées sont

INNER JOIN : renvoie les lignes qui ont des correspondances dans les deux tables [41].

```sql
SELECT p.nom, o.medicament
FROM patients p
INNER JOIN ordonnances o ON p.id_patient = o.id_patient;
```

Figure II-16 : Exemple de la jointure INNER JOIN en langage SQL.

LEFT JOIN : renvoie toutes les lignes de la table de gauche et les correspondances de la table de droite [41].

```sql
SELECT p.nom, o.medicament
FROM patients p
LEFT JOIN ordonnances o ON p.id_patient = o.id_patient;
```

Figure II-17 : Exemple de la jointure LEFT JOIN en langage SQL.

RIGHT JOIN : renvoie toutes les lignes de la table de droite et les correspondances de la table de gauche [41].

```sql
SELECT patients.nom, rendez_vous.date_consultation
FROM patients
RIGHT JOIN rendez_vous
ON patients.id_patient = rendez_vous.id_patient;
```

Figure II-18 : Exemple de la jointure RIGHT JOIN en langage SQL.

FULL JOIN : renvoie toutes les lignes quand il y'a une correspondance dans l'une des tables [41].

```sql
SELECT patients.nom, rendez_vous.date_consultation
FROM patients
FULL JOIN rendez_vous
ON patients.id_patient = rendez_vous.id_patient;
```

Figure II-19 : Exemple de la jointure FULL JOIN en langage SQL.

Remarque : Le FULL JOIN n'est pas disponible dans MySQL (jusqu'à la version 8), donc on le simule avec UNION de LEFT JOIN + RIGHT JOIN.

II.5.5 Les transactions

II.5.5.1. Définition

Une transaction dans une base de données est un ensemble d'opérations SQL, qui doivent être exécuter ensemble afin de garantir la cohérence des données dans un SGBD. Elle suit le principe de tout ou rien: si une des opérations échoue automatiquement toutes les autres doivent être annulées [45].

II.5.5.2. Propriétés ACID

Les transactions respectent les propriétés ACID, qui sont essentielles pour garantir la fiabilité et la cohérence des données :

Atomicité (Atomicity) : soit toutes les opérations de la transaction réussissent, ne soit aucune n'est appliquée (en cas d'erreur, un rollback annule toutes les modifications),

Cohérence (Consistency) : l'état de la base de données reste cohérent, avant et après la transaction,

Isolation (Isolation) : les modifications effectuées dans une transaction ne sont pas visibles par les autres transactions tant qu'elle n'est pas validée, évitant ainsi les interférences entre transactions concurrentes,

Durabilité (Durability) : une fois validée, la transaction est permanente même en cas de panne [45].

II.5.5.3. Syntaxe d'une transaction SQL

Voici un petit exemple classique en SQL :

```sql
START TRANSACTION;
UPDATE comptes SET solde = solde - 100 WHERE id = 1;
UPDATE comptes SET solde = solde + 100 WHERE id = 2;
COMMIT;

```

Figure II-20 : Exemple d'une transaction en langage SQL.

Dans cet exemple, on avait effectué un virement de 100 unités du compte 1 vers le compte 2. Si une des deux requêtes échoue, on peut annuler la transaction avec :

```sql
ROLLBACK;

```

Figure II-21 : Annulation d'une transaction en langage SQL.

II.5.5.4. Fonctionnement d'une transaction

Début de la transaction : on démarre la transaction avec la commande BEGIN ou BEGIN TRANSACTION,

Exécution des opérations : plusieurs requêtes SQL sont exécutées dans le cadre de cette transaction,

Validation ou annulation :

COMMIT : valide la transaction et rend permanentes toutes les modifications,

ROLLBACK : annule toutes les modifications effectuées depuis le début de la transaction, tout en restaurant la base à son état initial [45].

II.5.6 Les triggers (déclencheurs)

II.5.6.1. Définition

Un trigger ou déclencheur en français, est un programme associé à une table qui exécute automatiquement un ensemble d'instructions, lorsqu'un événement spécifique se produit comme : une insertion, une mise à jour ou une suppression de données. Son objectif principal, c'est d'automatiser certaines tâches dans un SGBD, tout en garantissant l'intégrité et la cohérence des données [26] [46].

Les triggers sont particulièrement utilisés pour :

Automatiser des tâches : comme par exemple, mettre à jour automatiquement le stock après une commande,

Audite les modifications : il enregistre chaque modification dans une table d'historique,

Valider des données : il empêche l'insertion de données invalide, comme un pris négatif,

Synchroniser des tables : il permet la mise à jour des tables secondaires après une modification,

Gérer des contraintes complexes : comme par exemple, interdire la baisse du prix d'un produit [26].

II.5.6.2. Syntaxe d'un trigger en SQL

MySQL est particulièrement adapté pour des tâches comme les logs ou validation des données, dans cet exemple on a un trigger où on va ajouter un log à chaque nouvelle insertion [26] :

```
1 CREATE TRIGGER LogInsert
2 AFTER INSERT ON utilisateurs
3 FOR EACH ROW
4 BEGIN
5 INSERT INTO logs (description, date_action)
6 VALUES ('Nouvel utilisateur ajouté : ' || NEW.nom, NOW());
7 END;
8
```

Figure II-22 : Exemple d'un trigger en langage SQL.

II.5.6.3Types de triggers SQL

BEFORE Trigger : ce trigger s'exécute avant qu'une action soit faite sur une table, il est utile pour valider ou modifier des données avant leurs insertions,

AFTER Trigger il s'exécute après que l'opération ait été réalisée, ce type de triggers est idéal pour effectuer des tâches comme : la mise à jour d'une autre table,

INSTEAD OF Trigger : celui la remplace l'action initiale par une autre, il est principalement utilisé avec des vues qui ne permettent pas de modifications directes [26].

II.6 Intégrité et sécurité des données

II.6.1 Intégrité référentielle

L'intégrité référentielle est un principe fondamental des bases de données relationnelles qui permet de garantir la cohérence des relations entre les tables. Elle s'appuie sur l'utilisation des clés primaires et étrangères chaque valeur de clé étrangère dans une table

doit correspondre à une valeur existante de clé primaire dans une autre table, ceci permettra d'assurer la validité des liens entre les données et d'éviter les incohérences [26] [45].

Les SGBD permettent de définir des contraintes d'intégrité référentielle lors de la création ou de la modification des tables. Ces contraintes empêchent alors les opérations qui violeraient la cohérence des données [26].

II.6.2 Sécurité et contrôle d'accès

La sécurité des bases de données est primordiale pour protéger les informations sensibles contre tout accès non autorisé ou modifications indésirables [46]. Elle comprend plusieurs aspects :

Authentification et autorisation : permet de vérifier l'identité des utilisateurs et de déterminer leurs droits d'accès ;

Chiffrement des données : cela protège les données en les rendant illisibles sans la clé de déchiffrement appropriée ;

Audit et surveillance : enregistrer les actions des utilisateurs afin de détecter les comportements suspects ou non autorisés ;

Contrôle d'accès : c'est le plier de la sécurité des données, il consiste à définir et à appliquer des règles permettant de déterminer quels utilisateurs ou systèmes peuvent y accéder, à quelles sources et avec quel niveau d'autorisation (lecture, écriture, modification, suppression, etc…) [46] [47].

Par exemple dans SQL Server, il est recommandé de :

Utiliser des mots de passes robustes et l'authentification à facteurs multiples ;

Mettre à jour régulièrement le logiciel pour corriger les vulnérabilités et les failles ;

Configurer des pares-feux pour restreindre l'accès au serveur de bases de données ;

Appliquer le principe du moindre privilège en accordant uniquement les permissions nécessaires aux utilisateurs [46] [48].

II.6.3 Sauvegarde et restauration des données

La sauvegarde et la restauration des données sont des processus essentiels pour assurer la disponibilité et la pérennité des informations en cas de panne, de corruption ou d'attaque. Elles permettent de récupérer les données à un état antérieur connu et stable.

La sauvegarde

Il existe plusieurs types de sauvegarde :

Sauvegarde complète : crée des copies de l'intégrité de la base de données ;

Sauvegarde différentielle : ne copie que les modifications apportées aux données depuis la dernière sauvegarde complète ;

Sauvegarde du journal des transactions : copie les transactions enregistrées depuis la dernière sauvegarde [49].

La restauration

Pour restaurer une base on commence d'abord par la dernière sauvegarde complète, puis les sauvegardes différentielles ou incrémentielles dans l'ordre afin de revenir à l'état le plus récent. Il est essentiel de tester régulièrement les procédures de restauration pour s'assurer de leur efficacité.

Une stratégie efficace combine ces méthodes afin d'optimiser la fréquence des sauvegardes et de minimiser la durée de restauration. Des outils comme SQL Server Management Studio (SSMS) facilitent ces opérations en proposant des interfaces graphiques pour planifier, exécuter et automatiser les sauvegardes et restaurations [38][49].

II.7 Application aux systèmes hospitaliers

II.7.1 Importance des bases de données dans les systèmes de santé

Les bases de données jouent un rôle crucial dans les systèmes de santé modernes, car elles permettent la collecte, la gestion et l'analyse efficace des informations médicales, ce qui facilite la prise en charge optimisée et personnalisée des patients. L'exploitation intelligente de ces données permet d'améliorer la coordination des soins, de réduire les erreurs médicales et d'optimiser la prise de décision clinique.

Elles contribuent à la sécurité du patient en permettant la prévention des situations indésirables grâce à la combinaison de données issues de différentes sources. Aussi elles optimisent la gestion opérationnelle des établissements de santé, en améliorant la gestion des flux de patients, la dotation en personnel et l'allocation des ressources.

De plus, la protection des données de santé est encadrée par des réglementations strictes, tel que celles du : Règlement Européen sur la Protection des Données (RGPD) ou la Loi Informatique et Libertés (LIL), qui garantissent la confidentialité et la sécurité des informations sensibles [38] [50].

II.7.2 Cas particulier de la gestion des pharmacies hospitalières

La gestion des pharmacies hospitalière est un domaine complexe qui nécessite une organisation rigoureuse et des systèmes d'informations performants. Elle est un cas spécifique où les bases de données sont essentielles pour garantir la disponibilité, la sécurité et la qualité des produits pharmaceutiques. De plus, ces bases facilitent également la coordination avec les autres services hospitaliers pour une prise en charge efficace des patients [51].

L'intégration de systèmes d'information spécialisés permet d'optimiser et de mieux maitriser le circuit des produits pharmaceutiques au sein d'un établissement hospitalier,

depuis leur réception jusqu'à leur utilisation ai lit du malade tout en facilitant le respect de la réglementation en vigueur et les bonnes pratiques de gestion de ces produits [50].

De plus, l'utilisation de tableaux de bord numériques au niveau des pharmacies hospitalières offre une visualisation en temps réel des indicateurs clés de performance (KPI) comme : les entrées, les niveaux de stock, les sorties, les consommations par services ou les alertes sur les produits critiques. Ces outils de pilotage permettent de faciliter la prise de décisions, tout en contribuant à une meilleure allocation des ressources et à une réduction des coûts liés aux péremptions ou au surstockage [38].

II.8 Conclusion

Ce deuxième chapitre nous a permis de poser les bases théoriques et techniques indispensables à la modélisation d'un système d'information efficace. En explorant les différents types de bases données, les méthodes de conception comme MERISE et UML et les langages de requêtes SQL, nous avons acquis une vision complète des outils permettant une gestion structurée, cohérente et sécurisée de l'information.

Nous avons également abordé les aspects essentiels liés à l'intégrité, la sécurité, la performance et l'optimisation des requêtes. Ces notions sont capitales pour garantir la fiabilité et la durabilité d'une application manipulant des données sensibles, comme celles d'une pharmacie hospitalière.

Enfin, la contextualisation des bases de données dans le domaine hospitalier a illustré leur rôle central dans la modernisation des systèmes de santé. Ce socle théorique constituera un appui précieux pour la conception technique de notre propre solution de gestion, tout en veillant à assurer une interopérabilité et une traçabilité optimales.

Chapitre III

Les applications web en général

Chapitre III : Les applications web en général

III.1. Introduction

L'évolution rapide des technologies numériques a profondément transformé les systèmes d'information modernes, notamment dans le domaine de la santé. L'une des avancées majeures repose sur les applications web**,** qui offrent un accès distant, centralisé et sécurisé aux données et services. Ce chapitre se propose d'explorer de manière détaillée ce qu'est une application web, ses caractéristiques techniques et fonctionnelles, ainsi que les architectures logicielles sur lesquelles elle repose. Nous aborderons également les différentes technologies utilisées pour leur conception, côté client et côté serveur, ainsi que les outils et frameworks les plus utilisés aujourd'hui. Enfin, une attention particulière sera portée à la sécurité des applications web, un enjeu majeur dans tout projet de développement, notamment lorsqu'il s'agit de manipuler des données sensibles comme celles du domaine hospitalier.

III.2 Définition et caractéristique des applications web

III.2.1 Définition d'une application web

Une application web désigne un logiciel applicatif hébergé sur un serveur et accessible via un navigateur web. Contrairement à un logiciel traditionnel, l'application web ne nécessite pas d'être installé, la connexion à l'application peut se faire par le biais d'un navigateur [52].

Elle repose généralement sur une architecture client-serveur, où le client envoie des requêtes au serveur via le protocole HTTP ou HTTPS, et reçoit des réponses contenant des contenus

HTML/CSS/JavaScript ou des données structurées. L'application peut intégrer une logique coté client et coté serveur, ainsi qu'une base de données pour stocker et gérer les informations [53].

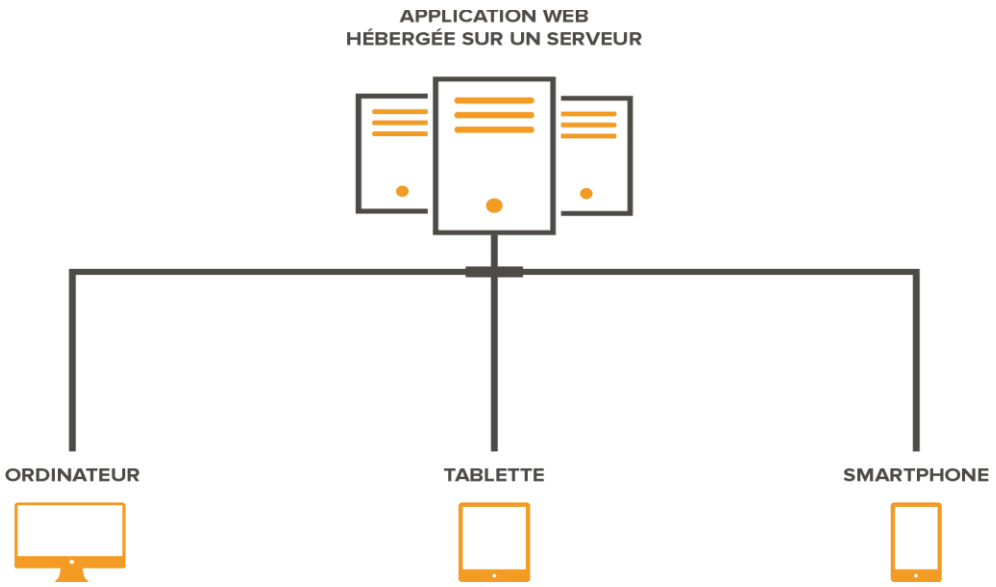

Figure III-1 : Schéma d'une application web accessible depuis différents terminaux via un serveur distant [52].

III.2.2 Différences entre site web, application web et application mobile

Application web, application mobile et site web sont des termes souvent employer de ma manière commune dans le langage courant, mais en réalité leur sens, sont bien distincts dans le domaine du développement numérique.

Un site web, également appelé site internet, se compose généralement de pages web statique ou dynamique reliées entre elles et accessibles à partir d'une même adresse appelée URL. Ces pages sont stockées sur un serveur et envoient des informations souvent à sens unique comme des articles, des images ou des vidéos, sans interactivité. Les technologies utilisées pour la conception d'un site web sont le HTML, CSS et parfois JavaScript, mais ne possède pas nécessairement de logique applicative complexe [53] [54].

Une application web, quant à elle est conçue pour fournir une interaction dynamique entre l'utilisateur et le système. A contrario d'un site web, elle permet le traitement côté serveur et côté client comme la gestion des comptes utilisateurs, la manipulation de bases de données ou le traitement de formulaires. Les applications web ont le même rôle qu'un logiciel classique, avec une logique métier intégrée et une gestion dynamique des données [52].

Enfin, une application mobile est programmée uniquement pour les systèmes d'exploitation mobiles comme Android et iOS. Elle est installée localement sur le smartphone ou la tablette de l'utilisateur et exploite les ressources matérielles de l'appareil comme l'appareil photo, le GPS, etc... Les applications mobiles peuvent être développées pour une plateforme spécifique ceci fait d'elle une application native ou elle peut être hybrides donc développées en technologies web mais encapsulées dans une application native [55].

Les trois types partage certains points communs, notamment l'interface utilisateurs, ils se distinguent par leur mode d'accès, leur niveau d'interactivité, leur environnement d'exécution, et les technologies mobilisées.

Tableau III-1 : Comparaison des principales caractéristiques entre un site web, une application web et une application mobile

Critère	Site web	Application web	Application mobile
Mode d'accès	Navigateur web	Navigateur web	Téléchargement via une boutique d'applications (App Store, Google Play)
Installation requise	Aucune	Aucune	Oui (installation locale sur l'appareil)
Interactivité	Faible à modérée (liens, formulaires simples)	Élevée (traitements dynamiques, authentification, interactions complexes)	Très élevée (gestes, notifications, accès matériel, etc.)
Dépendance à internet	Oui	Oui	Variable (fonctionnalités en ligne ou hors ligne selon le cas)
Technologies principales	HTML, CSS, JavaScript	HTML, CSS, JavaScript + backend (PHP, Node.js, etc.) + base de données	Java/Kotlin (Android), Swift (iOS), ou frameworks hybrides (Flutter, React Native…)
Accès aux fonctionnalités du matériel	Très limité	Limité (selon le navigateur et les permissions)	Élevé (GPS, caméra, capteurs, stockage local, etc.)
Portabilité multi-plateforme	Oui	Oui	Non native (une version spécifique par système ou utilisation de technologies hybrides)
	Automatique, côté serveur	Automatique, côté serveur	Nécessite une mise à jour via la boutique d'applications

III.2.3 Avantages et inconvénients des applications web

Les applications web ont une place importante dans les systèmes d'information modernes en raison de leur facilité d'utilisation, de leur accessibilité, et leur capité d'adaptation à diffèrent contextes technologiques. Cependant, elles présentent certaines limites techniques et fonctionnelles.

- **Avantage des applications web**

L'un des plus grands avantages d'une application web réside dans leur accessibilité multiplateforme ; elles peuvent être consultées depuis n'importe quel appareil équipé d'un navigateur, sans le besoin d'une installation local [52]. Cet atout permet une réduction des couts de déploiement et de maintenance, notamment dans un environnement professionnel où les mises à jour peuvent être centralisées cote serveur [53].

La maintenance d'une application web est significativement plus simple, car toute modification apportée au niveau du code serveur se répercute automatiquement pour l'ensemble des utilisateurs. De plus, leur l'utilisation du HTML, CSS et JavaScript leur donne une grande souplesse dans le développement et l'intégration de fonctionnalité nouvelles [53].

Pour finir, dans un contexte de modification numérique, les applications web sont plus rapidement déplorables que les applications mobiles, et permettent une meilleure compatibilité avec d'autres systèmes d'information [52].

- **Inconvénients des applications web**

Malgré ces nombreux atouts, les applications web présentent certains inconvénients. Elles dépendent fortement d'une connexion internet, ce qui peut freiner leur utilisation dans endroits à faible connectivité [55].

Sur ce qui concerne le fonctionnement, elles ont un accès limité aux fonctionnalités matérielles de l'appareil comme le GPS, stockage local, les capteurs, etc. à titre de comparaison les applications mobiles ont ces accès. Cette limite fonctionnelle peut restreindre leur champ d'application dans certains domaines spécialisés [55].

De plus, les performances peuvent être inférieures en termes de fluidité, de temps de réponse ou de graphisme comparé à une application native [53].

Enfin, l'inconvénient majeur reste la sécurité. Les applications web étant exposées sur internet, cella les rend vulnérable aux attaques courantes comme l'injection SQL, le cross-site scripting (XSS) ou encore les attaques CSRF [53].

III.2.4 Types d'applications web (monopage, multipage, progressive, etc.…)

Il y'a plusieurs types d'applications web elles ne se limitent pas à un seul modèle d'architecture ou d'usage. Elles peuvent avoir plusieurs catégories selon leur fonctionnement

leur niveau d'interactivité et leur structure de navigation. Les trois types les plus courant sont: les applications multipage (MPA), les applications monopage (SPA) et les applications web progressives (PWA) [52][53][55].

- **Les applications web multipage MPA (Multi Page Application)**

Une application multipage est basée sur une structure ou chaque clic ou interaction déclenche le chargement d'une page complète depuis le serveur [52].

Les MPA sont plus simples à sécuriser et permettent une organisation plus claire des contenue. Néanmoins, son utilisation peut être moins fluide, à cause de connexion lentes ou des rechargements fréquents [52].

- **Les applications web monopage SPA (Single Page Application)**

L'application web monopage n'a qu'un seul chargement ; toute la structure de la page est chargée une seule fois. Son fonctionnement reste simple, c'est en fonction des actions de l'utilisateur que les contenus dynamiques sont injectés via JavaScript, à l'aide d'AJAX ou d'API REST [53].

Le développement des SPA se fait souvent avec des bibliothèque ou frameworks comme React, Angular ou Vue.js. Elles offrent une expérience utilisateur plus fluide grâce à une navigation rapide, sans rechargement complet des pages [53].

- **Les Progressive Web Apps (PWA)**

Les PWA sont des applications web évoluées conçue pour fonctionner comme des applications mobiles, mais sans passer par une boutique d'applications. Elles peuvent fonctionner hors connexion grâce aux services workers [55]. Ce dernier est un script JavaScript qui permet de gérer de manière indépendante les requêtes réseau, la mise en cache, et certaines fonctionnalités avancées telles que les notifications push ou le fonctionnement hors-ligne [53].

III.3 Architecture d'une application web

III.3.1 Architecture client-serveur

Le modèle fondamental sur lequel repose la majorité des applications web modernes est l'architecture client-serveur. Elle définit la manière dont les différents composants d'une application (interface utilisateur, logique de traitement, données) sont répartis entre deux entités principales : le client et le serveur [53].

Le client est responsable de l'affichage et de l'interface graphique et, dans certains cas, d'une partie du traitement local grâce à JavaScript, par exemple [53].

Le serveur, reçoit les requêtes du client, exécute les traitements nécessaires comme l'accès à la base de données, puis renvoie une réponse par page HTML ou données JSON [53].

Ce modèle repose sur un protocole de communication standard : HTTP ou HTTPS, qui permet l'échange de données entre les deux entités. Ainsi, l'utilisateur interagit avec l'interface (client), qui envoie des requêtes au serveur, lequel répond en fournissant les informations ou les actions demandées [53].

Ce découplage entre client et serveur permet une meilleure répartition des ressources et une sécurité renforcée, puisque les données sensibles peuvent être centralisées et protégées côté serveur. C'est également ce modèle qui permet aux applications web d'être accessibles depuis n'importe quel appareil connecté à Internet, sans installation préalable [52][53].

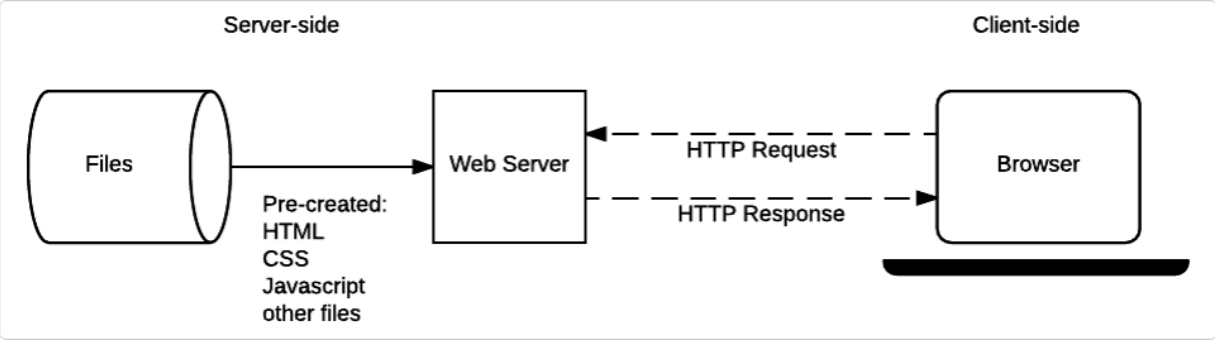

Figure III-2 : Schéma du modèle client-serveur dans une application web [53].

Cette figure si dessus représente un navigateur (client) qui envoie des requêtes http au serveur qui lui retourne les fichiers nécessaires (HTML, CSS, JavaScript). Ce modèle illustre la séparation des rôles entre la partie client et la partie serveur [53].

III.3.2 Modèle trois tiers : présentation / logique métier / données

Le modèle trois tiers est une forme évoluée du modèle client-serveur, largement utilisée dans les applications web modernes pour organiser l'architecture logicielle de manière modulaire, réutilisable et évolutive [22][54].

Ce modèle repose sur une séparation logique en trois couches indépendantes :

III.3.2.1 Couche de présentation

La couche de présentation est l'interface que l'utilisateur voit et manipule. Elle est développée avec des technologies telles que HTML, CSS, JavaScript et éventuellement des framworks frontend tel que React ou vue.js. Cette couche est responsable de l'affichage des données et de la navigation [53].

III.3.2.2 Couche de logique métier

La couche de logique métier aussi bien appelée couche applicative, est la partie qui contient les règles métier de l'application ; elle traite les données, exécute la logique fonctionnelle comme la vérification des identifiants et le traitement des commandes, et elle

fait le lien entre la présentation et la base de données. Elle est hébergée sur un serveur web, et est souvent développer en PHP, Python, Java ou NOde.js [53].

III.3.2.3 Couche de données

La couche de données est responsable du stockage, de la gestion et de la récupération des informations persistantes utilisées par l'application. Elle repose sur un système de gestion de base de données SGBD tel que MySQL, PostgreSQL ou MongoDB. Cette couche est accédée par la couche métier, ce qui renforce la sécurité des données [53].

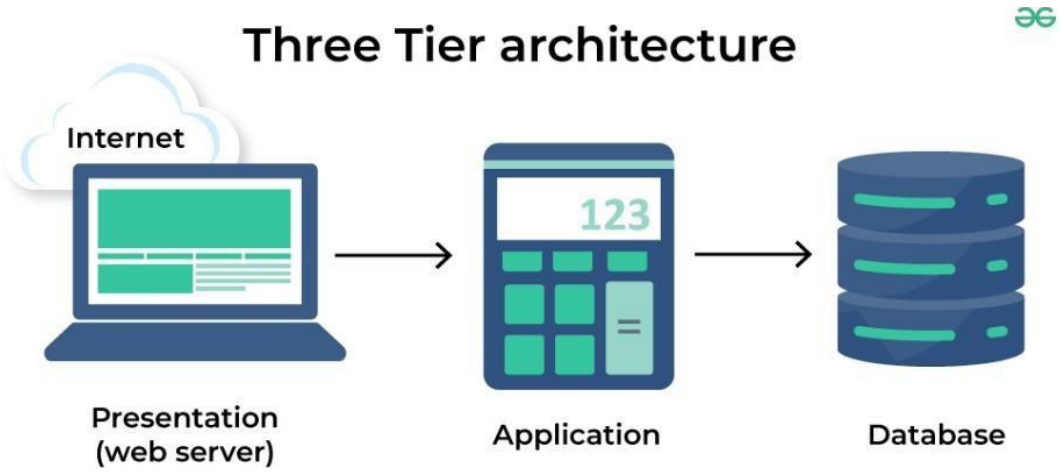

Figure III-3 : Schéma de l'architecture trois tiers d'une application web [54].

III.3.3 Technologies côté client (HTML, CSS, JavaScript…)

Les technologies cote client sont les langages et outils utilisé pour développer l'interface d'une application web. Elles sont exécutées directement dans le navigateur de l'utilisateur, et jouent un rôle central dans l'expérience utilisateur, la navigation, et la présentation des contenus.

III.3.3.1 HTML (Hyper Text Markup Language)

Le HTML est le langage de balise standard utilisé pour structurer le contenue d'une page web. Il définit les éléments tels que les titres, paragraphes, images, tableaux et formulaire. Chaque page web repose sur une base de HTML, qui sert de squelette a l'application [41].

En HTML, il y'a deux notions clés qui sont les balises et les éléments, elles sont souvent confondues mais techniquement différentes.

Une balise est un symbole ou une instruction entre deux chevrons (< >). Il en existe trois types : les balises ouvrantes, les balises fermantes et les balises orpheline [41].

La balise ouvrante indique le début d'un élément exemple (début du paragraphe).

La balise fermante indique la fin d'un élément exemple (fin du paragraphe).

La balise orpheline aussi appelé balise vide est une balise qui ne contient pas de contenu et ne nécessite pas de balise fermante. Elle est utilisée pour insérer un élément autonome dans le document exemple <img> cette balise permet d'insérer une image dans une page web [41].

Un élément HTML désigne l'ensemble constitué par la balise ouvrante, le contenue et la balise fermante exemple : Bienvenue sur mon site [41].

Figure III-4 : exemple de structure HTML d'une page simple.

Ce code représente la structure minimale d'un document HTML5. La déclaration <!DOCTYPE html> indique que le fichier suit le standard HTML5. L'élément <html> est l'élément racine de la page. La section <head> contient des métadonnées, dont <title>, qui définit le titre affiché dans l'onglet du navigateur. La section <body> regroupe tous les éléments visibles de la page. Ici, on y trouve un titre principal <h1>, un sous-titre <h2>, et un paragraphe , illustrant les balises de base utilisées pour structurer le contenu d'une page web.

III.3.3.2 CSS (Cascading Style Sheets)

Le CSS est utilisé pour définir l'apparence visuelle des pages HTML. Il permet de spécifier les couleurs, les polices, les espacements, les effets graphiques et les emplacements. Il permet aussi de créer dans interfaces responsive, adaptées aux différents types d'écrans [55]. Un code CSS peut d'être intégré de deux façons ; sois dans fichier externe, puis serra mis dans la section

<head> exemple <link rel="stylesheet" href="style.css"> ou dans une balise <style> dans la section <head> comme l'illustre la figure suivante

exemple.html > html > head > style > h2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>exemple</title>
    <style>
body {
    font-family: Arial, sans-serif;
    background-color: #f9f9f9;
    color: #333;
    margin: 20px;
    padding: 20px;
}
h1 {
    font-size: 32px;
    color: #005f73;
    margin-bottom: 10px;
}
h2 {
    font-size: 24px;
    color: #0a9396;
    margin-bottom: 8px;
}
p {
    font-size: 16px;
    line-height: 1.6;
    color: #555;
}

    </style>
</head>
<body>
    <h1>Titre principale</h1>
    <h2>Titre secondaire</h2>
    <p>Paragraphe</p>
</body>
</html>
```

Figure III-5 : Exemple d'intégration de styles CSS dans un document HTML.

Cette image illustre l'intégration directe de règles CSS dans une page HTML à l'aide d'une balise <style> placée dans l'élément **<head>**. Le code CSS permet ici de styliser les éléments

<body>, **<h1>**, **<h2>** et **<p>**, en définissant leurs couleurs, tailles de police, marges, et autres propriétés visuelles.

Le CSS permet de séparer la structure (HTML) de la présentation (CSS), ce qui améliore la lisibilité et la maintenabilité du code [35].

III.3.3.3 JavaScript/ AJAX

Le JavaScript est un langage de programmation à part entière, est écrit en texte pure et s'exécute dans le navigateur. Il permet de rendre une application web plus fonctionnelle, interactive et dynamique. Il est utilisé pour gérer les évènements, modifier le contenue (sans recharger la page), et manipuler le DOM (Document Object Model) pour ajuster l'affichage selon les actions de l'utilisateur [35] [57].

Le code JavaScript peut être intégré directement dans une page HTML entre les balises **<script></script>**, généralement situées à la fin de la section **<body>**, afin de s'exécuter une fois que le contenu HTML est entièrement chargé. Cette organisation permet d'éviter les erreurs liées à l'absence d'éléments HTML encore non affichés [35].

JavaScript est également la base des frameworks frontend modernes comme React, Vue.js ou Angular, qui facilitent le développement d'applications web complexes et réactives [56].

L'une des extensions les plus puissante de JavaScript est l'AJAX :

AJAX est l'acronyme pour Asynchronous JavaScript and XML, il apporte une technique qui permet d'échanger des données avec un serveur en arrière-plan et d'obtenir des informations complémenter, sans recharger la page entière. Cette technologie repose sur l'objet **XMLHttpRequest** ou **fech()** [57].

- **XMLHttpRequest :** Cet objet permet au code JavaScript d'accéder au serveur de manière asynchrone et en arrière-plan, sans recharger la page [57].
- **fech() :** Cette interface moderne permet d'effectuer des requêtes HTTP asynchrones plus simplement, en s'appuyant sur les Promesses pour améliorer la lisibilité du code [57].

Grace à AJAX, une application web peut :

- Afficher des suggestions de recherche en temps réel,
- Mettre à jour une section précise d'une page comme un tableau,
- Valider un formulaire sans rafraichissement,
- Communiquer avec API pour charger ou enregistrer des données [57].
- AJAX constitue un élément fondamental de la réactivité des applications web modernes, en apportant une expérience utilisateur plus fluide et interactive [57].

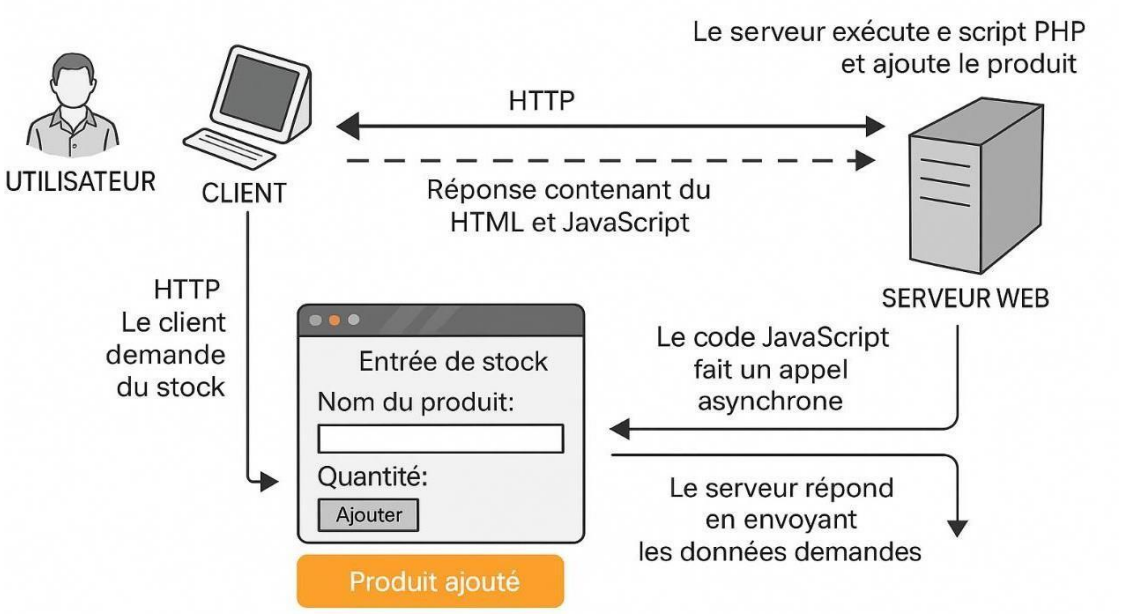

Figure III-6 : Comparaison entre un chargement classique d'une page web et un appel AJAX typique.

Cette figure illustre deux approches d'interaction entre le client et le serveur dans une application web :

Chargement classique (en haut) :

Le client (navigateur de l'utilisateur) envoie une requête HTTP au serveur web pour obtenir une page PHP. Le serveur exécute le script PHP, génère du contenu HTML (éventuellement enrichi en JavaScript), puis renvoie l'ensemble au navigateur, qui recharge entièrement la page. Ce fonctionnement est synchrone et implique une interruption complète de l'affichage à chaque action utilisateur.

Appel AJAX typique (en bas) :

Après un premier chargement HTML/JavaScript, le code JavaScript présent dans la page effectue des requêtes AJAX asynchrones, via XMLHttpRequest ou fetch(). Ces requêtes sont invisibles pour l'utilisateur : seul un fragment des données est demandé au serveur (souvent via une API), puis la réponse est traitée côté client pour mettre à jour dynamiquement l'interface sans rechargement complet. Cette approche améliore la réactivité, la fluidité, et l'expérience utilisateur.

III.3.4 Technologies côté serveur (PHP, Node.js, Python, etc.)

Les technologies côté serveur jouent un rôle central dans le fonctionnement d'une application web. Contrairement aux technologies côté client qui s'exécutent dans le navigateur de l'utilisateur, les technologies côté serveur s'exécutent sur une machine distante, appelée serveur, qui traite les requêtes, applique la logique métier, interagit avec la base de données, et

renvoie des réponses personnalisées au client. Ces technologies sont essentielles pour créer des applications web dynamiques, sécurisées et performantes [57] [58] [59].

III.3.4.1 PHP (Hypertext Preprocessor)

Le PHP est un langage de script cote serveur conçu spécialement pour le développement web. Il est interprété cote serveur, génère du HTML dynamique, et peut facilement interagir avec des bases de données comme MySQL. PHP est souvent utilisé dans des architectures classiques à base de LAMP (Linux, Apache, MySQL, PHP) [57].

Il est prisé pour sa courbe d'apprentissage accessible, sa large communauté, sa documentation riche, et son intégration directe avec HTML. Il reste aujourd'hui un pilier du développement web, notamment dans des structures publiques qui utilisent des solutions comme des ERP codés sur mesure [57].

« PHP permet de construire facilement un backend capable de répondre à des requêtes asynchrones provenant de l'interface utilisateur » [57].

- **Fonctionnement général**

Lorsqu'une requête est envoyée par le client vers le serveur, un script PHP peut être déclenché pour :

Recevoir les données envoyées par l'utilisateur (par exemple, via un formulaire),

Effectuer des traitements (validation, calcul, authentification, etc.),

Communiquer avec une base de données pour récupérer ou insérer des informations,

Générer une réponse structurée (HTML ou JSON) renvoyée au client [57].

- **Les variables superglobales en PHP**

PHP met à disposition un ensemble de superglobales, elles sont prédéfinies, ce qui signifie qu'elles sont toujours accessibles dans tous les contextes de script PHP sans avoir à les déclarer globalement [41].

Voici les principales, accompagnées d'un exemple :

$_GET : permet de récupérer les données transmises dans l'URL, généralement issues d'un formulaire utilisant la méthode GET [41].

$_POST : est utilisée pour accéder aux données envoyées par un formulaire en méthode post, souvent pour des traitements sécurisés**.** Contrairement à **$_GET,** ces informations ne sont pas visibles dans l'URL, elle est donc recommandée pour l'envoi de données sensibles comme les mots de passe [41].

$_REQUEST : regroupe les données provenant de **$_GET**, **$_POST** et **$_COOKIE**. Elle offre un accès unifié, quel que soit le moyen par lequel les données ont été transmises. Son utilisation doit cependant être prudente pour éviter les conflits ou effets indésirables [41].

$_SESSION : stocke des données spécifiques à l'utilisateur côté serveur pendant toute la durée de sa session. Elle est notamment utilisée pour conserver l'état de connexion, suivre un panier d'achat ou stocker des préférences temporaires entre plusieurs pages [41].

$_COCKIE : permet d'accéder aux données enregistrées sur le navigateur du client. Ces données sont stockées localement et renvoyées au serveur à chaque requête HTTP, ce qui permet de mémoriser des informations persistantes (langue, identifiant, préférences…) [41].

$_SERVER : contient un ensemble d'informations sur le serveur et la requête en cours, telles que le nom du serveur, l'adresse IP du client, le nom du fichier exécuté, la méthode HTTP utilisée, ou encore les en-têtes HTTP. Elle est essentielle pour adapter le comportement du script selon le contexte d'exécution [41].

$_FILES : gère les fichiers envoyés via un formulaire HTML. Elle permet de récupérer des informations sur le fichier (nom, taille) et de le déplacer vers un répertoire sécurisé sur le serveur [41].

$_ENV : permet d'accéder aux variables d'environnement[1] définies sur le serveur [41].

$_GLOBALS : permet d'accéder à toutes les variables globales déclarées dans le script, indépendamment de la portée dans laquelle on se trouve. Elle est utile pour manipuler des variables globales à l'intérieur de fonctions [41].

exemple.php > ...

```php
<?php
// $_GET : récupère les données passées dans l'URL (ex : page.php?nom=Mokrane)
echo $_GET['nom'];

// $_POST : récupère les données envoyées par un formulaire HTML
echo $_POST['email'];

// $_REQUEST : contient les données de $_GET, $_POST et $_COOKIE
echo $_REQUEST['id'];

// $_SESSION : stocke les données de session côté serveur
session_start();
$_SESSION['utilisateur'] = "admin";
echo $_SESSION['utilisateur'];

// $_COOKIE : accède aux cookies stockés sur le navigateur client
echo $_COOKIE['langue'];

// $_FILES : gère les fichiers téléchargés par formulaire
$nomFichier = $_FILES['document']['name'];
echo "Nom du fichier : " . $nomFichier;

// $_SERVER : fournit des infos sur le serveur et la requête en cours
echo $_SERVER['SERVER_NAME']; // Nom du serveur (ex : localhost)

// $_ENV : accède aux variables d'environnement du serveur
echo $_ENV['DB_HOST']; // Exemple : nom d'hôte de la base de données

// $_GLOBALS : permet d'accéder à toutes les variables globales du script
$a = 5;
$b = 10;
function somme() {
    $GLOBALS['resultat'] = $GLOBALS['a'] + $GLOBALS['b'];
}
somme();
echo $resultat;
?>
```

Figure III-7 : Illustration commentée des principales superglobales PHP.

Cette figure présente un exemple pratique d'utilisation des principales **superglobales en PHP**, accompagnées de commentaires explicatifs. Chaque variable met en évidence son rôle spécifique dans la gestion des données utilisateur, des sessions, des cookies, des fichiers, ou encore de l'environnement serveur. On y retrouve :

$_GET, **$_POST** et **$_REQUEST** pour la récupération des données soumises via HTTP **;**

$_SESSION pour le suivi de l'utilisateur connecté ;

$_COOKIE pour accéder aux informations stockées localement dans le navigateur ;

$_FILES pour le traitement de fichiers téléchargés ;

$_SERVER et **$_ENV** pour interagir avec le système serveur ;

$_GLOBALS pour manipuler des variables globales au sein d'une fonction.

Cette représentation permet de visualiser concrètement la puissance de ces outils intégrés à PHP pour développer des applications web dynamiques et sécurisées [41].

- **Traitement d'une requête POST en PHP**

L'une des utilisations les plus courantes de PHP consiste à traiter les données issues d'un formulaire HTML envoyé en POST. Cela permet, par exemple, d'insérer des informations dans une base de données MySQL, puis de renvoyer une réponse au format JSON à l'interface web [41].

exepmle.php > ...

```php
<?php
// Connexion à la base de données
$host = 'localhost';
$user = 'root';
$password = '';
$dbname = 'pharmacie_utilisateur_db';

$conn = new mysqli($host, $user, $password, $dbname);

if ($conn->connect_error) {
    die("Échec de la connexion : " . $conn->connect_error);
}

// Vérifie si la requête est une méthode POST
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Récupère les données du formulaire
    $nom = $_POST["nom"];
    $age = $_POST["age"];

    // Prépare une requête SQL (exemple simplifié)
    $sql = "INSERT INTO utilisateurs (nom, age) VALUES ('$nom', '$age')";
    $conn->query($sql);

    // Crée une réponse sous forme de tableau
    $reponse = array("message" => "Bonjour $nom, vos informations ont été enregistrées.");

    // Envoie la réponse encodée en JSON
    echo json_encode($reponse);
}
?>
```

Figure III-8 : Exemple de traitement d'une requête POST en PHP.

Ce code PHP illustre une interaction complète entre le serveur et la base de données MySQL avec récupération de données POST, insertion SQL, et réponse JSON.

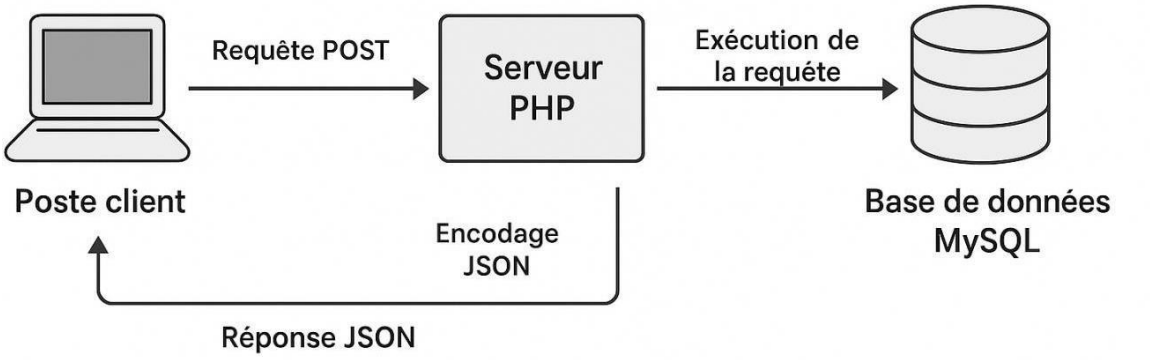

Figure III-9 : Schéma du fonctionnement de PHP avec une base de données.

Ce schéma illustre le processus typique d'une requête POST envoyée depuis un client (navigateur) vers un serveur PHP, qui traite les données, les enregistre dans une base de données MySQL, puis renvoie une réponse JSON.

- **Sécurité et usage de PDO**

Pour éviter les attaques par injection SQL, il est recommandé d'utiliser **PDO (PHP Data Objects)**. Ce mécanisme permet de préparer les requêtes SQL à l'avance avec des paramètres nommés ou positionnels, réduisant ainsi les risques de manipulation malveillante [41].

En somme, PHP reste une technologie puissante, souple et sécurisable pour la création d'applications web dynamiques, surtout lorsqu'il est couplé avec AJAX, MySQL et une bonne gestion des superglobales et des requêtes HTTP.

III.3.4.2 Node.js

Node.js est un environnement d'exécution JavaScript open source et multiplateforme. C'est un outil populaire pour presque tous les types de projets. Il est conçu pour permettre l'exécution de code JavaScript en dehors d'un navigateur, en utilisant le moteur V8 de Google Chrome [58].

Une application Node.js s'exécute dans un seul processus. Sans créer un thread[2] distinct pour chaque requête. A contrario des architectures classiques fondées sur le multi-threadings

Node.js repose sur un modèle événementiel non bloquant. C'est-à-dire que lorsqu'une opération d'entrée/sortie (lecture de fichier, accès réseau ou base de données) est déclenchée, Node.js ne bloque pas le thread principal, mais reprend son exécution [58].

Ce paradigme permet à un seul serveur Node.js de gérer des milliers de connexions simultanées, sans la surcharge liée à la gestion concurrente des threads, souvent source de bugs complexes [58].

Node.js facilite le travail des développeurs. Un développeur qui maitrise déjà JavaScript peut également développer la partie serveur sans avoir à apprendre un nouveau langage, assurant ainsi une unification du développement client-serveur [53].

Le code suivant représente un exemple classique d'un serveur HTTP minimal :

```
<script>
const { createServer } = require('node:http');

const hostname = '127.0.0.1';
const port = 3000;

const server = createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});

```

Figure III-10 : Cycle de fonctionnement d'un serveur HTTP en Node.js [58].

const { createServer } = require('node:http'); Cette ligne utilise la déstructuration pour importer uniquement la fonction createServer du module natif http fourni par Node.js.

Ce module permet de créer des serveurs web sans avoir besoin d'installer de bibliothèque externe [58].

const hostname = '127.0.0.1'; const port = 3000;

Ces deux constantes définissent le nom d'hôte (localhost) et le port sur lequel le serveur écoutera les requêtes [58].

127.0.0.1 désigne l'adresse locale du serveur [58].

3000 est un port souvent utilisé pour les tests en développement [58].

const server = createServer((req, res) => {

On crée le serveur avec la méthode createServer(), qui prend en argument une fonction de rappel (callback) avec deux paramètres :

req : représente la requête du client (ex. : ce qu'il envoie au serveur).

res : représente la réponse que le serveur va envoyer au client.

res.statusCode = 200;

On définit le code de statut HTTP à 200, ce qui signifie que la requête s'est bien déroulée.

```
res.setHeader('Content-Type',  'text/plain');
```

On précise que la réponse renverra du texte brut (text/plain) et non du HTML, JSON, etc.

```
res.end('Hello World');
```

Cette méthode termine la réponse en envoyant le texte "Hello World" au client. Cela ferme la connexion et affiche le message dans le navigateur ou le terminal.

```
});
```

Fin de la fonction de rappel associée au serveur.

```
server.listen(port, hostname, () => {
```

On demande au serveur de commencer à écouter sur l'hôte et le port définis plus haut. Quand le serveur est prêt, la fonction de rappel est exécutée.

```
console.log (`Server running at http://${hostname}:${port}/`);
});
```

Affiche dans la console un message de confirmation pour indiquer que le serveur est bien lancé et accessible à l'adresse donnée.

Ce petit serveur Node.js peut déjà répondre à une requête HTTP avec une réponse statique. C'est un excellent exemple de la simplicité et de la performance de Node.js pour démarrer une application serveur [58].

III.3.4.3 Python

Python est un langage de programmation polyvalent, réputé pour sa syntaxe simple, sa lisibilité naturelle, et sa large communauté d'utilisateurs dans les domaines de l'intelligence artificielle, de la science des données, et du développement web. Il est particulièrement adapté à des projets web côté serveur grâce à des frameworks puissants et modulaires [59].

Python est souvent utilisé avec des frameworks web comme Django et Flask, qui facilitent la création d'applications robustes, sécuriséeset bien structurées. Ces outils offrent un cadre de développement prédéfini, permettant de gagner du temps, de suivre des bonnes pratiques, et de mieux organiser le code source [59].

Points forts de Python pour le côté serveur

- Une courbe d'apprentissage douce et une syntaxe naturelle,
- Des performances suffisantes pour des applications critiques,
- Une communauté très active et une documentation riche [59].

```
<pre><code>
from flask import Flask

app = Flask(__name__)

@app.route('/')
def accueil():
    return "Bonjour depuis Flask !"

if __name__ == '__main__':
    app.run(debug=True)

</code></pre>
```

Figure III-11 : Exemple simple de serveur web avec Python (Flask) [59].

Explication du code

- from flask import Flask: importation de la classe Flask,
- app = Flask(__name__): instanciation de l'application,
- @app.route('/'): définition de la route racine,
- def accueil(): fonction retournant une réponse http,
- app.run(debug=True): exécution du serveur local avec le mode debug [59].

III.3.5 Protocole HTTP/HTTPS et API REST

Le protocole http (HyperText Transfer Protocol) constitue la base de la communication sur le web. Il permet aux navigateurs d'envoyer des requêtes aux serveurs et de recevoir des réponses, le plus souvent sous forme de pages HTML, de données JSON ou d'autres formats lisibles par les clients. La version sécurisée de ce protocole, HTTPS, chiffre les échanges entre le client et le serveur grâce à TLS/SSL, garantissant ainsi la confidentialité et l'intégrité des données échangées [53].

Chaque communication HTTP repose sur des **méthodes** comme :

GET : pour récupérer une ressource du serveur.

POST : pour envoyer des données (par exemple, depuis un formulaire).

PUT : pour mettre à jour une ressource.

DELETE : pour supprimer une ressource.

Ces méthodes sont fondamentales dans l'architecture des API REST (Representational State Transfer), un style d'architecture qui s'appuie sur HTTP pour permettre aux applications de s'échanger des données de manière simple, cohérente et sans état. Une API REST organise ses points d'accès autour de ressources (souvent représentées par des objets) accessibles via des URL et manipulables à l'aide des méthodes http [53].

Pour faciliter la lecture et le traitement des données côté client, les échanges s'effectuent le plus souvent en JSON (JavaScript Object Notation). Ce format léger et lisible permet de structurer les données de manière hiérarchique et d'assurer une compatibilité native avec JavaScript [59].

L'envoi des requêtes http de manière asynchrone coté client est rendu possible par deux mécanismes ; XMLHttpRequest et fetch() [57].

III.4. Outils et technologies de développement

III.4.1 Environnements de développement

Un environnement de développement intégré (IDE) est un outil fondamental dans la création d'applications web. Il permet d'écrire, organiser, tester et débloquer le code de manière efficace, tout en offrant une meilleure productivité au développeur. Les IDEs combinent plusieurs fonctionnalités dans une seule interface ; éditeur de texte, gestionnaire de projets et coloration syntaxique [53].

Parmi les outils le plus connu :

Visual Studio Code (VS Code) : un éditeur léger mais puissant développé par Microsoft. Il est multiplateforme, personnalisable grâce à ses extensions, et largement utilisé pour le développement web (HTML, CSS, JavaScript, PHP, Python, etc.) [53].

PhpStorm : un IDE complet pour le développement PHP, développé par Jet Brains. Il offre une intégration avancée avec les frameworks PHP, les bases de données et les outils de test [53].

PyCharm : également développé par JetBrains, cet IDE est spécialisé pour le langage Python. Il est utilisé dans le développement d'applications web via Django ou Flask [53].

Sublime Text : connu pour sa rapidité et sa simplicité, il propose une interface épurée et des fonctionnalités avancées comme le multi-curseur ou la recherche rapide [53].

Le choix d'un IDE dépend généralement de plusieurs facteurs : le langage de programmation utilisé, la nature du projet, les besoins en fonctionnalités, ainsi que les préférences personnelles du développeur. Il constitue l'interface directe entre l'humain et la machine, facilitant ainsi toutes les étapes de la conception, du débogage et de la maintenance des applications web [53].

III.4.2 Frameworks frontend (React, Angular, Vue.js…)

- **Les Frameworks**

Frontend sont des environnements de développement coté client, conçus pour simplifier la création d'interfaces utilisateur interactives. Ils permettent de structurer l'affichage, de manipuler le DOM de manière performante, et de gérer l'état des composants

dynamiquement. Ces outils sont basés sur des principes tels que la modularité, la réutilisabilité des composants et la synchronisation des données avec l'interface [53][60].

Voici les principaux frameworks les plus utilisés :

- **React**

React est une bibliothèque JavaScript orientée composants, développé par Facebook. Elle repose sur le concept de Virtual DOM, qui optimise le rendu en mettant à jour les éléments modifiés de la page. React favorise la création d'interfaces performantes et modulaires, avec des états gérés localement dans chaque composant [60].

- **Angular**

Angular, développé par Google, est un framework complet basé sur TypeScript. Il intègre de nombreuses fonctionnalités dès l'installation ; liaison bidirectionnelle des donnée, injection de dépendances, routage, formulaires réactifs, services et modules.

Angular suit une architecture basée sur le Modèle-Vue-Contrôleur (MVC), idéale pour de applications de grande envergure et bien structurées [60][61].

- **Vue.js**

Vue.js est un framework progressif créé par Evan You. Il combine la simplicité de React avec la structure d'Angular, tout en conservant une courbe d'apprentissage plus douce. Vue permet une intégration facile dans des projets existants ou le développement d'applications monopage (SPA) [60][62].

Il repose aussi sur un système de composants réactifs et de liaison de données, tout en restant léger et flexible [60][63].

Tableau III-2 : Tableau comparatif des principaux frameworks frontend.

Framework	Langage principal	Avantages clés	Inconvénients / Limites	Cas d'usage typique
React	JavaScript (JSX)	Modularité grâce aux composants DOM virtuel pour de meilleures performances Large communauté et écosystème riche (Redux, Next.js…)	- Nécessite une configuration initiale (webpack, Babel…) - Courbe d'apprentissage sur la gestion des états complexes	Applications web interactives, tableaux de bord dynamiques, SPAs
Angular	TypeScript (TS)	Framework complet : routage, formulaires, HTTP, tests intégrés Architecture MVC solide - Maintenu par Google	-Courbe d'apprentissage plus raide - Plus lourd à charger et à mettre en place	Applications métiers à grande échelle, ERP, plateformes complexes

Vue.js	JavaScript	-Facile à prendre en main pour les débutants -Intégration progressive dans des projets existants - Documentation claire et complète	Moins utilisé dans les grandes entreprises que React ou Angular Écosystème plus petit	Projets rapides, sites interactifs, applications SPA légères

III.4.3 Frameworks backend (Laravel, Express.js, Django)

Les frameworks backend sont des outils de développement qui permettent d'organiser, sécuriser et structurer efficacement la logique côté serveur d'une application web. Ils assurent le traitement des requêtes, la gestion des sessions, l'accès aux bases de données, la définition des routes, ainsi que le rendu des vues ou la fourniture de données via des API. Ces outils permettent un développement plus rapide, plus sûr et plus maintenable, tout en respectant des normes et architectures éprouvées comme MVC (Modèle-Vue-Contrôleur).

- **Laravel (PHP)**

Laravel est un framework backend open-source écrit en PHP. Il propose un écosystème riche comprenant un ORM (Eloquent), un système de migration pour la base de données, des routes intuitives, des middlewares pour la sécurité, ainsi qu'un moteur de templates (Blade) [63].

Les caractéristiques principales de Laravel sont ; sa gestion facile des bases de données avec Eloquent, son système de routage souple et clair et son support natif des taches planifiées, files d'attente et notification [63].

Eloquent est le composant le plus apprécié de Laravel il facilite la manipulation des données en base en utilisant des objets PHP orientés objet plutôt que du SQL classique [63].

Laravel connais plusieurs avantages il est idéal pour des objets web complexes ou évolutifs et possède une large communauté. Néanmoins, il possède aussi des inconvénients comme sa consommation de mémoire très élevées, et il requiert de bonnes connaissances en PHP [63].

- **Express.js (Node.js)**

Express.js est un frameworck rapide, léger et minimaliste pour Node.js. Il permet de développer des applications web et des API REST de manière efficace, tout en laissant une grande liberté au développeur. Express repose sur un modèle middleware qui permet de chaîner des fonctions pour gérer les requêtes et les réponses [64].

Les principales caractéristiques d'Express.js sont un routage simple mais puissant, une intégration facile avec des bases de données comme MongoDB ou MySQL, il est compatible avec toutes les bibliothèques Node.js [64].

Ce qui concerne ces avantages, Express.js est rapide, léger, flexible, avec une communauté importante. En revanche, il est parfois nécessaire de gérer de nombreux paramètres manuellement comme la sécurité et la structure du projet [64].

- **Django**

Django est un framework complet de développement web, écrit dans le langage de programmation Python, et fondé sur le concept "batteries incluses" : il fournit, entre autres, un système d'authentification, un système de gestion des utilisateurs, un espace d'administration, des mesures de sécurité contre le CSRF, le XSS et bien d'autres types d'injection, des outils de pagination et un outil de création de formulaires. Au-delà des aspects techniques, Django est largement utilisé dans des projets nécessitant robustesse, sécurité et scalabilité des systèmes [65].

Les principales caractéristiques de Djongo sont Un ORM puissant et fonctionnel permettant d'interagir avec les bases de données relationnelles, une interface d'administration générée automatiquement, un système de routing, des templates et la gestion des vues [65].

Ce qui concerne ces avantages, il est rapide à mettre en œuvre, très sécurisé, idéal pour des applications critique. Néanmoins, il a moins de flexibilité pour les petits projets [65].

III.4.4 CMS et solutions low-code/no-code (WordPress, Bubble, etc.)

Les CMS (Content Management Systems) et les solutions low-code/no-code commencent petit à petit à prendre de l'ampleur et une place croissante dans le milieu du développement web. Ces outils permettent de concevoir des sites et des applications sans avoir à écrire de grandes quantités de code, rendent ainsi le développement accessible à un public non technique ou permettant un gain de temps pour les professionnels [66] [67].

CMS – Systèmes de gestion de contenu

Un CMS est une plateforme qui facilite la création, la gestion et la publication de contenus numériques (textes, images, vidéos, etc.) via une interface graphique intuitive. Il repose généralement sur une architecture client-serveur avec une base de données en arrière-plan pour stocker les contenus [66].

Exemple principal : **WordPress**. Open-source et écrit en PHP, il alimente plus de 40 % des sites web dans le monde. Il offre des milliers de thèmes et d'extensions pour adapter l'apparence et les fonctionnalités sans coder [66].

L'utilisation d'un CMS a des avantages comme ; la mise en ligne rapide de contenue, la possibilité d'ajouter des fonctionnalités via des extensions, la gestion multi-utilisateurs et contrôle des accès et la mise à disposition de plusieurs interface utilisateur conviviale. Néanmoins, les CMS sont moins flexible que les codes personnalisés, il y'a de gros risques

sécuritaires si les extensions ne sont pas maintenues à jour et sur les gros projets il peut y'avoir un manque de performances [66].

Solutions low-code / no-code

Les solutions low-code permettent aux développeurs de créer une application avec peu de code, tandis que les solutions no-code permettent de créer une application sans utiliser de code, cette solution offre une interface graphique complète. Ces interfaces reposent sur une technique du glisser-déposer [67].

Exemple : avec Bubble, l'utilisateur glisse un bouton ou un champ de formulaire sur la zone d'édition, puis définit les workflows associés, ce qui permet de créer des applications entièrement fonctionnelles sans toucher au code [67].

III.4.5 Hébergement et déploiement (XAMPP, Apache, Nginx, Docker…)

L'hébergement et le déploiement représentent l'étape finale du cycle de développement web. Ils consistent à rendre une application accessible aux utilisateurs, que ce soit sur un serveur local pour des tests ou sur un serveur distant pour un usage réel. Plusieurs outils facilitent cette opération, selon les besoins du projet, son échelle et son environnement technique [68].

XAMPP - Environnement et développement local

XAMPP est un package tout-en-un qui regroupe Apache (serveur web), MySQL/MariaDB (base de données), PHP et Perl. Il permet de simuler un environnement serveur localement, idéal pour le développement et les tests avant déploiement [68].

- **Avantages :**

Facile à installer et configurer

Compatible Windows, macOS et Linux

Idéal pour tester des applications PHP/MySQL

- **Inconvénients :**

Ne convient pas pour le déploiement en production

Moins sécurisé qu'un environnement distant contrôlé

Apache - Serveur http classique

Apache HTTP Server est l'un des serveurs web les plus utilisés au monde. Il fonctionne sur un modèle de traitement multi-thread ou multi-processus, ce qui le rend robuste et modulaire. Il est particulièrement adapté pour héberger des applications PHP et s'intègre facilement avec des CMS comme WordPress [69].

Nginx - Serveur web performant et moderne

Nginx est un serveur web conçu pour la performance et la gestion des connexions simultanées. Contrairement à Apache, il adopte un modèle événementiel asynchrone, ce qui le rend plus léger et rapide dans certains contextes [70].

- **Cas d'usage :**

Serveur de fichiers statiques (HTML, images, JS),

Reverse proxy pour des applications Node.js ou Python,

Utilisé dans des architectures à haute disponibilité [70].

Docker - conteneurisation et déploiement flexible

Docker permet d'emballer une application et ses dépendances dans des conteneurs, garantissant ainsi que le code fonctionne toujours de la même façon, quelle que soit la machine [71].

Avantages :

- Isolation des environnements,
- Déploiement reproductible,
- Compatible CI/CD (intégration/déploiement continu) [71].
- Exemple d'usage :

Déployer une application web avec une base de données MySQL, un serveur Apache, et PHP dans des conteneurs séparés communiquant entre eux via Docker Compose [71].

III.5. Sécurité des applications web

La sécurité des applications web est un enjeu majeur dans le développement moderne. À mesure que les systèmes deviennent de plus en plus interconnectés et accessibles, ils s'exposent à des vulnérabilités croissantes qui risque de compromettre l'intégrité, la confidentialité et la disponibilité des données. Comprendre les menaces, adopter les bonnes pratiques de sécurité et mettre en place des mécanismes robustes de contrôle d'accès est aujourd'hui indispensable pour garantir la fiabilité des applications web [53][72].

III.5.1 Vulnérabilités courantes

Les applications web sont fréquemment exposées à diverses menaces de sécurité. Parmi les plus connues figurent :

- **Injection SQL (SQLi) :** attaque consistant à insérer du code SQL malveillant dans un champ d'entrée pour manipuler la base de données, comme accéder ou supprimer des données sensibles.
- **Cross-Site Scripting (XSS) :** injection de scripts malveillants dans une page web, pouvant être exécutés dans le navigateur de l'utilisateur et voler des cookies ou des sessions.
- **Cross-Site Request Forgery (CSRF) :** attaque où un utilisateur authentifié est piégé pour exécuter une action involontaire sur une application web, souvent via un lien malveillant.

- **Directory Traversal :** tentative d'accès à des fichiers sensibles du serveur à l'aide de chemins relatifs (../../).
- **Remote Code Execution (RCE) :** exécution de code arbitraire à distance sur le serveur vulnérable.

Ces vulnérabilités peuvent avoir des conséquences graves sur la confidentialité, l'intégrité et la disponibilité des données [72].

III.5.2 Bonnes pratiques de sécurité

Pour protéger les applications web, plusieurs bonnes pratiques doivent être appliquées dès les premières phases de développement :

- **Validation et filtrage des données** côté client et surtout côté serveur.
- **Utilisation de requêtes préparées (avec PDO, Mysqli...)** pour prévenir les injections SQL.
- **Mise en œuvre de Content Security Policy (CSP)** pour se prémunir contre les attaques XSS.
- **Vérification des jetons CSRF** pour chaque requête sensible.
- **Hachage des mots de passe** avec des algorithmes sécurisés comme bcrypt ou Argon2.
- **Gestion rigoureuse des erreurs** pour éviter les messages d'erreur détaillés exposant la structure de l'application.
- **Mises à jour régulières** des bibliothèques, frameworks et dépendances utilisées.

L'adoption de ces pratiques constitue une ligne de défense essentielle contre les attaques les plus répandues [53].

III.5.3 Gestion des utilisateurs et des accès

Un système sécurisé doit également bien gérer les identités et les autorisations :

- **Authentification :** vérifie l'identité d'un utilisateur (par login/mot de passe, jeton, etc.).
- **Autorisation :** définit ce qu'un utilisateur a le droit de faire (accès à certaines ressources ou actions).
- **Séparation des rôles :** administrateurs, utilisateurs, invités, etc. avec des droits différents.
- **Expiration automatique des sessions**, verrouillage après plusieurs échecs de connexion, journalisation des connexions.

Un contrôle d'accès strict est indispensable pour limiter les abus internes et protéger les données sensibles [73].

III.5.4 Certificats SSL et chiffrement des données

- **Le protocole HTTPS**, basé sur SSL/TLS, permet de sécuriser les échanges entre le client et le serveur en les chiffrant.
- **Les certificats SSL** assurent l'authenticité du serveur et empêchent les attaques de type "Man in the Middle".
- **Le chiffrement des données** (AES, RSA…) garantit que même si une donnée est interceptée, elle reste illisible.
- **Les jetons d'authentification JWT** peuvent aussi être signés/chiffrés pour renforcer la sécurité des API.

Le chiffrement et l'utilisation de certificats sécurisés sont aujourd'hui des standards incontournables pour toute application web [74].

III.6 Conclusion

Ce chapitre nous a permis de mieux comprendre le fonctionnement global des applications web, tant sur le plan de leur architecture que de leurs technologies. En clarifiant la distinction entre site web, application web et application mobile, nous avons acquis une grille de lecture claire des différentes solutions disponibles.

Nous avons étudié les modèles d'architecture logicielle, les langages de développement front-end et back-end, ainsi que les environnements d'hébergement et de sécurité. Ces connaissances sont fondamentales pour garantir une application robuste, fluide et sécurisée.

L'analyse des bonnes pratiques de sécurité (XSS, SQLi, CSRF…) nous a sensibilisés aux risques liés à la manipulation des données sensibles dans un contexte hospitalier. Ainsi, ce chapitre fournit donc les clés pour concevoir une solution web fiable et performante, tout en respectant les normes de sécurité les plus strictes.

Chapitre IV

Conception pratique de notre application

Chapitre IV : Conception pratique de notre application

IV.1 Introduction

Ce dernier chapitre présente la réalisation concrète de notre projet G-PHARMA, une application web dédiée à la gestion des dispositifs médicaux dans une pharmacie hospitalière. L'objectif est de décrire avec précision toutes les étapes de conception, les choix technologiques, les interactions entre composants, les bases de données impliquées, ainsi que les contraintes rencontrées.

IV.2 Analyse des besoins

IV.2.1 Etude du contexte hospitalier réel

La gestion des produits pharmaceutiques au sein de la pharmacie hospitalière du CHU Nedir Mohamed de Tizi-Ouzou observée lors du stage repose sur l'utilisation d'un logiciel déjà en place. Ce logiciel permet de réaliser plusieurs opérations essentielles telles que la gestion des entrées, la consultation des stocks, ou encore le suivi des produits. Toutefois, malgré son utilité, il présente quelques limites fonctionnelles majeures qui nuisent à une traçabilité complète :

Absence de suivi des marchés : aucune fonctionnalité ne permet d'enregistrer ou de suivre l'évolution des marchés passés avec les fournisseurs, ce qui empêche toute vérification contractuelle ou historique des approvisionnements.

Non-enregistrement des sorties de garde : les produits pharmaceutiques fournis en dehors des horaires standards (gardes de nuit, week-ends) ne sont pas saisis dans le logiciel, ce qui crée des ruptures de traçabilité.

Gestion manuelle des décharges : les documents de décharge ne sont ni générés ni archivés électroniquement ; tout se fait encore sur papier.

Ces opérations sont donc effectuées manuellement, soit sur registre papier, soit sur des fichiers Excel. Cela engendre un risque de perte d'informations, une duplication des tâches, et complique la supervision du flux de produits pharmaceutiques dans les services hospitaliers.

La conception d'une nouvelle application vise donc à combler ces lacunes précises, tout en intégrant les bonnes pratiques déjà établies par le logiciel existant.

IV.2.2 Définition des besoins fonctionnels

Les besoins fonctionnels ont été déterminés pour à la fois reprendre les fonctionnalités existantes du logiciel actuel et ajouter celles qui manquent :

- **Fonctionnalités à reprendre :**
  - Gestion des stocks (consultation, mise à jour)
  - Enregistrement des entrées (date, lot, quantité, fournisseur)
  - Attribution des dotations aux services
  - Enregistrement des sorties
  - Historique complet des mouvements
- **Fonctionnalités à ajouter ou améliorer :**
  - Enregistrement des sorties de garde
  - Création et gestion des décharges
  - Suivi des marchés et contrats fournisseurs
  - Filtrage dynamique et recherches intelligentes
  - Impression ou exportation PDF des données

IV.2.3 Spécifications techniques attendues

Les spécifications techniques visées pour l'application sont :

- **Technologies :**

Frontend : HTML5, CSS3, JavaScript, TomSelect

Backend : PHP (programmation modulaire), base de données MySQL

Environnement local : XAMPP (Apache, MySQL, PHP)

- **Structure modulaire avec bases séparées :**

Utilisateurs pour la gestion des accès

Dispositifs pour les consommables et dispositifs

Médicaments et réactifs pour les modules futurs

Bonne séparation du code (HTML/CSS, logique PHP, base de données)

Sécurité via PDO (protection contre les injections SQL)

Fonctions dynamiques AJAX

Support des opérations hors ligne par impression papier ou export PDF

IV.3 Choix initiaux et réorientation du projet

IV.3.1 Premier choix

Au début du projet, l'approche choisis consistait à concevoir une application reposant sur une base de données centralisée unique. Celle-ci devait regrouper toutes les informations nécessaires à la gestion de la pharmacie hospitalière **:** dispositifs médicaux, médicaments, réactifs, utilisateurs, services, fournisseurs, etc.

Par ailleurs, le projet devait être initialement développé sans utiliser JavaScript, en misant uniquement sur HTML, CSS et PHP.

IV.3.2. Raisons de ce choix initial

Ce choix initial reposait sur plusieurs arguments :

Simplicité de développement : une base unique signifiait des relations SQL simples à gérer dans un seul environnement.

Code PHP pur : éviter JavaScript permettait d'avoir un code serveur unique, plus facile à contrôler pour un développeur débutant.

Moins de dépendances techniques : pas besoin de bibliothèques JS externes, ni d'interactions AJAX complexes.

IV.3.3 Problèmes rencontrés

Au fil du développement, plusieurs limites sont apparues :

Quantité croissante d'informations : à mesure que le projet avançait, le volume des données à gérer augmentait.

Absence de séparation logique : toutes les fonctionnalités étant dans une seule base, le système devenait difficile à structurer.

Ralentissements : le volume des données augmentant, les performances se déradaient, surtout pour les historiques et les vues globales.

Rigidité sans JavaScript : l'absence de JavaScript rendait l'expérience utilisateur moins fluide (pas de listes dynamiques, pas de calculs automatiques dans les formulaires, pas de recherche instantanée).

Difficulté à implémenter des interfaces modernes : la navigation était limitée, les actions nécessitaient un rechargement complet de page, et l'ergonomie était réduite.

IV.3.4 Nouveaux : séparation en 4 bases de données et intégration de JavaScript

Face à ces difficultés, une réorientation majeure a été décidée. Le projet a été restructuré en adoptant :

Une architecture modulaire avec 4 bases distinctes :

pharmacie_utilisateur_db

pharmacie_dispositif_db

pharmacie_medicament_db

pharmacie_reactif_db

L'**intégration progressive de JavaScript**, notamment avec :

TomSelect pour les listes déroulantes dynamiques,

jsPDF pour l'exportation PDF,

AJAX pour les chargements sans rechargement de page.

IV.3.5 Justifications de cette restructuration

Les avantages obtenus avec cette nouvelle approche :

Séparation claire des informations : chaque base a son propre ensemble de tables.

Expérience utilisateurs améliorée : grâce à JavaScript, les formulaires deviennent plus dynamiques et interactifs.

Flexibilité du code : l'utilisation combinée de PHP et JavaScript permet d'ajouter plus facilement de nouvelles fonctionnalités sans tout recharge

IV.4 Modélisation de la base de données

La modélisation de la base de données n'a pas suivi une méthode formelle comme MERISE ou UML. Elle a été réalisée directement en SQL, en fonction des besoins concrets de l'application, à l'aide de l'outil phpMyAdmin fourni avec XAMPP.

L'ensemble des tables, vues, et relations ont été conçues manuellement, avec une attention portée à la cohérence des identifiants, la simplicité des jointures, et les performances de consultation.

IV.4.1 Création des bases et tables dans XAMPP

Deux bases sont détaillées ici :

phramacie_utilisateur_db :

Contient les informations générales sur les utilisateurs, les médecins, et les services hospitaliers :

utilisateur : comptes de connexion et rôles.

medecin : liste des médecins (utilisée à titre indicatif).

service : unités hospitalières (chirurgie, urgences, réanimation, etc.).

vue_service_medecin : vue SQL qui lie chaque médecin à son service.

pharmacie_dispositif_db :

Gère tout ce qui concerne les dispositifs médicaux et consommables :

Tables principales : dispositif, classe_dispositif, stock, entree, sortie_bdc, fournisseur, laboratoire_production, alarme, alerte_visuelle, etc.

Autres modules : marcher, detail_marcher, decharge, sortie_garde, sortie_oi.

IV.4.2 Vues SQL créées

Plusieurs **vues SQL** ont été créées pour faciliter l'affichage, les jointures complexes et l'exploitation dans l'interface PHP :

Tableau IV-1 : Vues SQL créées.

Nom de la vue	Objectif
vue_entree_dispositif	Liste les entrées avec désignation, numéro de lot, quantité, prix, date
vue_sortie_bdc_dispositif	Affiche les sorties avec nom du service, date, quantité livrée
vue_stock_dispositif	Permet de visualiser le stock actuel avec les quantités et dates de péremption
vue_decharges	Centralise les décharges enregistrées
vue_alarme_dispositif	Compare les seuils d'alarme et les niveaux réels
vue_suivi_marcher	Donne un aperçu de l'état d'exécution des marchés
vue_sortie_garde, vue_sortie_oi	Affichent les sorties hors BDC (garde, ordonnances internes)

Avantage

Elles évitent de répéter les jointures dans le code PHP.

Elles permettent de créer des interfaces propres avec tri, recherche et export PDF.

Elles améliorent les performances et la sécurité (lecture seule).

IV.4.3 Déclencheur automatique d'alerte (Trigger)

Un trigger SQL nommé trigger_surveille_stock a été créé afin de surveiller chaque nouvelle entrée ou modification du stock. Son rôle est de comprarer automatiquement la quantité et la date de péremption des produits avec les seuils définis dans la table alarme.

En cas de dépassement par exemple quantité < seuil minimal, une alerte est insérée dans la table alerte_visuelle avec un code couleur exemple orange pour seuil minimal, rouge pour seuil critique et bleu pour seuil maximal.

Un second trigger, trigger_surveille_stock_update, agit de la même manière lors des mises à jour de la table stock.

Cette logique permet une surveillance continue sans intervention humaine, garantissant une réactivité accrue en cas de rupture ou de surstock.

IV.4.4 Relations entre les bases de données

Même si chaque base est autonome, des **relations logiques** ont été établies à travers des identifiants :

Tableau IV-2 : Lien entre les tables.

Lien	Exemple
id_service	Utilisé dans la base utilisateur et référencé dans sortie_bdc, dotation, etc.
id_utilisateur	Utilisé pour tracer les actions ou contrôler les droits
id_fournisseur, id_laboratoire	Présents dans dispositif, entree, marcher

Important : aucun lien entre bases n'utilise de contrainte de clé étrangère, car MySQL ne permet pas les relations inter-bases par défaut. Les correspondances sont assurées uniquement par la logique des identifiants partagés et les vues.

IV.4.5 Utilisation des index

Les index sont automatiquement créés pour chaque clé primaire (id_*) et parfois ajoutés sur les colonnes utilisées pour les jointures fréquentes (ex : id_dispositif, id_service, numero_lot).

Ces index ont permis :

Une **accélération des requêtes SQL**, notamment dans les vues.

Une **meilleure réactivité** dans les filtres dynamiques et la recherche instantanée côté client.

IV.5 Architecture de l'application

L'application G-PHARMA a été développée selon une architecture client-serveur classique, adaptée à un déploiement local à l'aide de XAMPP. Elle repose sur une séparation claire entre la présentation, la logique de traitement, et la gestion des données. Cette structure garantit à la fois une bonne lisibilité du code, une modularité des composants, et une facilité d'évolution.

IV.5.1 Architecture client-serveur

L'application fonctionne selon un modèle client-serveur, dans lequel :

- **Le client** (navigateur web) est responsable de l'affichage et de l'interaction avec l'utilisateur. Il est construit avec :
- **HTML** pour la structure des pages,
- **CSS** pour la mise en forme,
- **JavaScript**, notamment via **jQuery** et **AJAX**, pour rendre les interfaces dynamiques (recherche en temps réel, chargement asynchrone de données, modals, etc.).

- **Le serveur** est exécuté localement à l'aide de XAMPP et gère les traitements backend:

PHP exécute la logique applicative, traite les formulaires, contrôle les autorisations et gère les sessions.

MySQL assure le stockage et la structuration des données dans plusieurs bases, selon une séparation modulaire (utilisateurs, dispositifs, médicaments, réactifs).

IV.5.2 Communication entre les fichiers

La communication dans l'application repose sur plusieurs mécanismes :

- **Formulaires HTML + méthode POST :**

Utilisés pour transmettre des données au serveur (ex. : saisie d'une entrée, ajout d'une alarme, modification de données).

- **Requêtes AJAX (GET ou POST) :**

Utilisées pour charger dynamiquement des données (ex.: liste des services, dispositifs disponibles, dotations par service).

Évitent le rechargement de la page et permettent une navigation fluide.

- **Sessions PHP :**

Utilisées pour identifier les utilisateurs connectés, stocker leurs informations (nom, autorisations) et contrôler l'accès aux pages.

Les sessions sont vérifiées en début de chaque page protégée.

IV.5.3 Authentification et gestion des rôles

L'accès à l'application est sécurisé par un système d'authentification basé sur une base de données dédiée. Lors de la connexion, les identifiants sont transmis via AJAX et validés côté serveur. Si la connexion est réussie :

Les informations de l'utilisateur sont stockées dans une session PHP.

Le champ statut_connexion est mis à jour pour refléter l'état de l'utilisateur.

Les droits d'accès sont déterminés par des champs spécifiques (acces_dispositif, acces_medicament, acces_reactif).

IV.6 Interfaces et navigation

L'interface utilisateur de l'application G-PHARMA a été conçue dans un souci d'ergonomie, de lisibilité et de simplicité d'utilisation. Chaque module est présenté sous forme de pages dynamiques accessibles depuis un tableau de bord centralisé. L'ensemble des interfaces repose sur des technologies web classiques (HTML, CSS) avec des composants interactifs renforcés par JavaScript et des bibliothèques comme **TomSelect** (pour les listes

déroulantes intelligentes) ou **jsPDF** (pour l'exportation PDF). Chaque interface mentionnée ci-dessous est illustrée par des captures dans l'annexe.

- **Page de connexion**

La première interaction avec l'application se fait via une page de connexion centrée, au design sobre, avec fond vert et formulaire d'authentification sécurisé. L'utilisateur saisit son identifiant et son mot de passe pour accéder à son tableau de bord. Un lien permet également la création d'un nouveau compte.

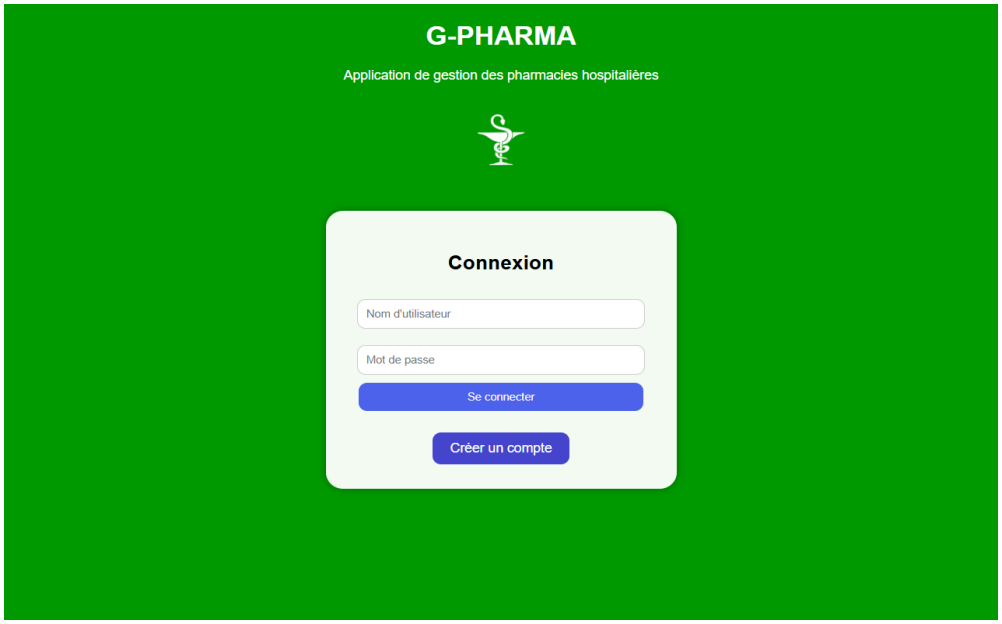

Figure IV-1 : Interface de connexion utilisateur.

- **Tableau de bord**

Après authentification, l'utilisateur est redirigé vers un tableau de bord principal qui affiche, selon ses droits, des boutons d'accès aux trois sections principales de l'application : **Médicaments**, **Réactifs** et **Dispositifs**. Aucun autre module n'est soumis à un contrôle d'accès par rôle.

Figure IV-2 : Page d'accueil de la section dispositifs.

- **Page de gestion des dispositifs**

Cette interface regroupe l'ensemble des fonctionnalités liées aux dispositifs médicaux. Elle est structurée autour d'un menu latéral contenant plusieurs modules :

Référentiel : Dispositifs, Classes, Laboratoires, Fournisseurs, Services, Médecins, Établissements.

Configuration : Alarmes de sécurité, Dotations par service.

Mouvements : Entrée, Sortie, Stock, Historique, Marchés, Décharges.

Chaque bouton de la zone principale permet d'accéder à un formulaire ou une liste liée au module sélectionné.

Un système d'alerte visuelle a été intégré à cette interface ; lorsqu'une alerte est détectée, un message s'affiche automatiquement en haut de la page (par exemple : *Attention :* stock critique détecté pour [nom du dispositif]). Ces alertes sont générées dynamiquement via une requête AJAX vers un fichier get_alertes_actives.php qui interroge la table alerte_visuelle.

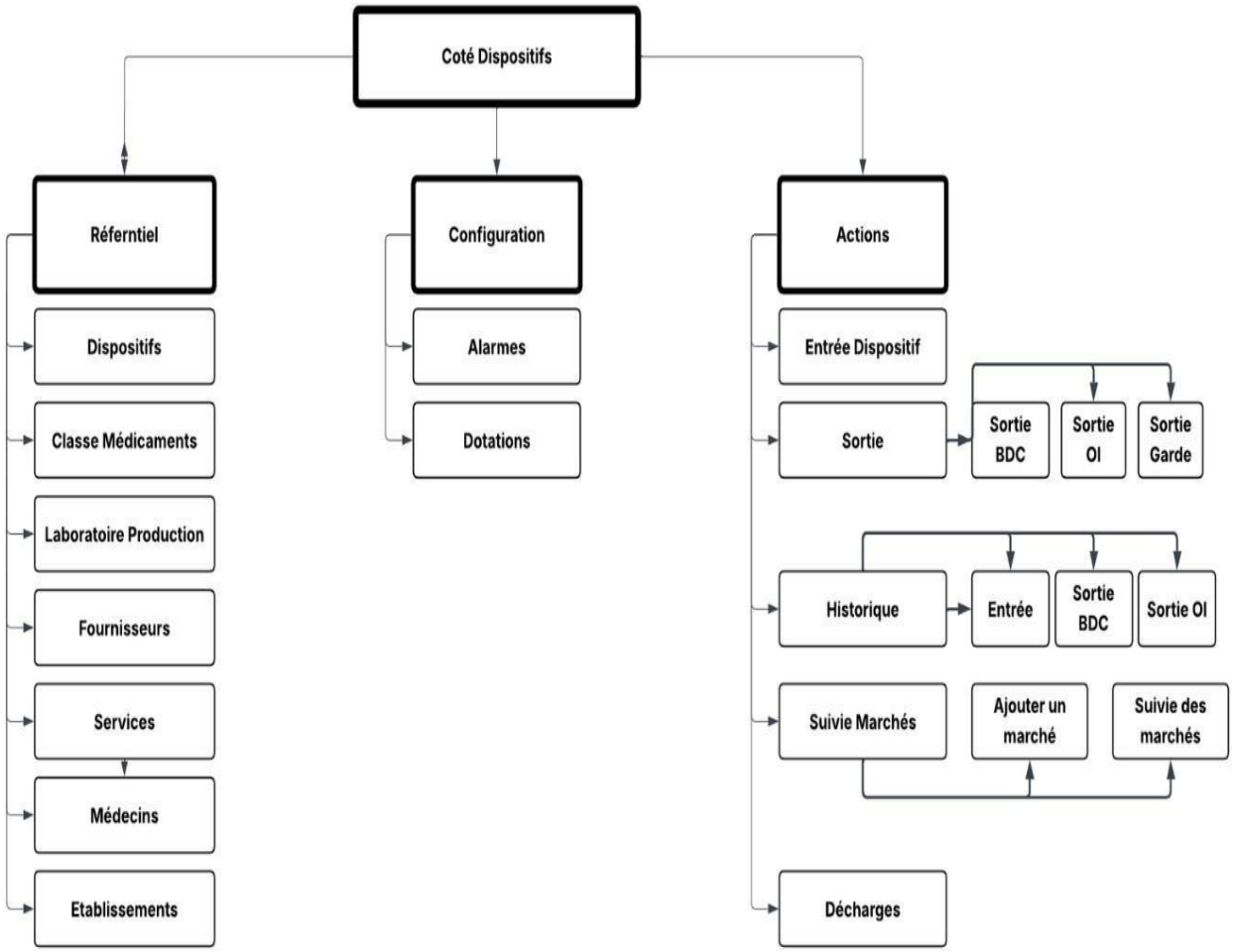

Figure IV-3 : Organisation fonctionnelle du module de gestion des dispositifs médicaux.

IV.6.1 Interfaces fonctionnelles principales

- **Entrée des dispositifs**

Cette interface permet l'enregistrement des entrées de dispositifs. Elle comporte un formulaire avec plusieurs champs : désignation, numéro de lot, dates (péremption, réception), quantité, prix unitaire, TVA, prix total. Trois boutons en haut de page permettent d'afficher les dernières entrées, d'imprimer une sélection ou d'exporter en PDF.

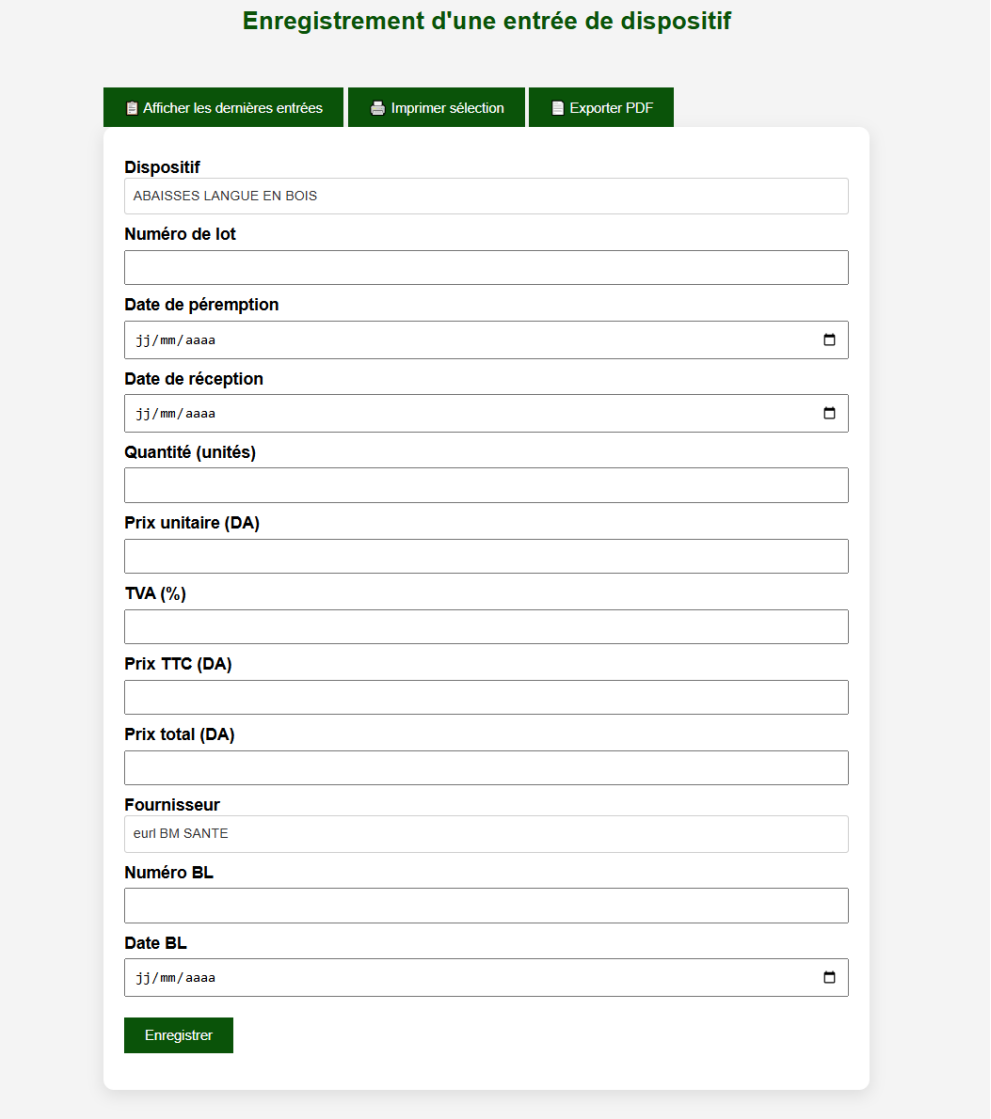

Figure IV-4 : Formulaire d'entrée des dispositifs.

- **Stock**

La section stock affiche l'état actuel des dispositifs disponibles, triés par désignation. Chaque ligne indique les informations essentielles : numéro de lot, date de péremption, quantité. Des boutons permettent également d'imprimer ou exporter la page.

Un système d'alerte a aussi été ajouter à cette interface. La ligne du tableau du dispositif en alerte est colorée avec la couleur représentant le seuil dépassé.

- **Sorties de dispositifs**

Trois types de sorties sont proposés :

Sortie par bon de commande (BDC) : formulaires dynamiques avec ajout de lignes.

Sortie par ordonnance interne : formulaire lié à un médecin et un service.

Sortie de garde : formulaire spécifique au pharmacien de garde.

Ces interfaces incluent la désignation, le numéro de lot, la quantité, le prix unitaire et le prix total.

- **Historique des mouvements**

Le module historique est divisé en trois sous-sections :

Entrées : filtrage avancé par fournisseur, lot, désignation, date de péremption, date de réception ou période.

Sorties BDC : tableau récapitulatif sans possibilité de filtrage.

Sorties OI : tableau récapitulatif sans possibilité de filtrage.

Chaque tableau affiche les informations essentielles : produit, quantité, prix, date, etc.

- **Alarmes**

L'interface "Alarme dispositif" permet de définir des seuils de gestion : critique, minimal, maximal et péremption (en jours). Ces seuils déclenchent des alertes visuelles dans les modules gestions des dispositifs.

- **Dotations par service**

La page affiche les dotations des services hospitaliers, avec la possibilité de filtrer par service ou par dispositif. Les dotations peuvent être ajoutées ou modifiées.

- **Listes des utilisateurs**

Cette interface liste les utilisateurs avec leurs rôles (admin, pharmacien) et leur statut de connexion (connecté/déconnecté).

IV.6.2. Interfaces administratives

- **Fournisseurs et laboratoires**

Chacune des deux interfaces permet de gérer une liste (consultation, ajout) avec formulaire d'ajout : nom, adresse, téléphone, email.

- **Services et médecins**

La page des services affiche leur nom et le chef de service (modifiable). La page des médecins permet de rechercher par service ou par nom, et d'ajouter/modifier/supprimer un médecin.

- **Etablissements de santé**

Cette interface permet l'enregistrement des hôpitaux avec leur type (CHU, EPH), adresse, téléphone, fax. Les établissements sont utilisés dans les transferts entre structures.

Marchés et suivi

Le module permet :

L'ajout de marchés (fournisseur, date, liste des dispositifs)

Le suivi de la livraison : quantités totales, livrées, restantes, taux de livraison, prix TTC, montant livré.

Décharges inter-établissements

Ce formulaire permet d'enregistrer des transferts de dispositifs d'un établissement à un autre, avec affichage de l'historique en bas de page.

IV.7 Difficultés rencontrées et solutions

La mise en place de l'application de gestion des dispositifs médicaux a été matérialisée par de nombreuses étapes techniques et prises de décisions, accompagnée de plusieurs difficultés qui ont nécessité des ajustements progressifs.

Et voici les principales difficultés rencontrées au cours du développement, suivi de chaque solution apportée :

Organisation initiale de la base de données

- **Problème:** au départ, l'idée était de centraliser l'ensemble des données (utilisateurs, médicaments, dispositifs et réactifs) dans une seule base de données. Cela s'est révélé complexe à maintenir et ont avait très vite rencontrer de nombreux problèmes, notamment en termes de lisibilité, de performance et de gestion des autorisations spécifiques à chaque domaine.
- **Solution:** une restructuration complète a été réalisée, tel qu'on avait divisé notre projet en quatre bases de données distinctes: pharmacie_utilisateur_db, pharmacie_dispositif_db, pharmacie_medicament_db et pharmacie_reactif_db. Ce découpage nous a permis d'avoir une meilleure séparation des responsabilités, une organisation plus claire des données et une gestion plus fluide des accès.

Définition des accès utilisateurs

- **Problème** : initialement, on avait envisagé que chaque utilisateur ait des autorisations spécifiques par module fonctionnel (stock, entrée, sortie, etc…). Cependant, cela aurait alourdi la gestion des rôles et complexé la logique d'accès.

- **Solution** : un compromis a été adopté, on a décidé de limiter les autorisations d'accès global par domaine (médicaments, dispositifs et réactifs), ce qui a permis de simplifier la gestion et couvrir la majorité des besoins rencontrés dans le service hospitalier.

Gestion des connexions et sessions

- **Problème** : lors de l'implémentation de la page index.php avec la méthode AJAX, ont rencontré quelques difficultés pour authentifier correctement les utilisateurs et maintenir leurs sessions active, tout en assurant une redirection fluide selon leurs privilèges.
- **Solution :** l'utilisation de connecter.php avec une réponse JSON nous a permis de valider les identifiants via PDO, de stocker les autorisations (acces_dispositif, acces_medicament et acces_reactif) dans la session PHP, puis de rediriger dynamiquement vers la page login.php après authentification. Mais aussi, la mise à jour du champ statut_connexion nous permet de suivre les connexions actives.

Problèmes liés à la méthode GET et au chargement dynamique

- **Problème:** le chargement dynamique des listes (dispositifs, laboratoires, services, etc…) via la méthode GET posait parfois problème, notamment en cas d'incohérence dans les réponses JSON ou de lenteurs lors de rafraîchissement de certains composants.
- **Solution :** l'utilisation de bibliothèques comme **TomSelect** pour les listes déroulantes, a permis d'améliorer considérablement l'expérience utilisateur, avec des fonctionnalités de recherches et de sélection dynamique. De plus, la séparation claire entre les fichiers HTML et les fichiers get_*.php a permis un meilleur contrôle du chargement des données.

Implémentation de l'impression et de l'export en PDF

- **Problème :** il nous a été difficile d'intégrer une solution d'impression sélective et d'export en PDF pour les entrées et sorties, surtout en conservant une mise en page lisible et professionnelle.
- **Solution :** l'intégration de la bibliothèque **jsPDF** avec **autotable** a permis de générer de manière dynamique les tableaux en PDF. Et puis, des boutons spécifiques «Imprimer» et «Exporter PDF» ont été ajoutés dans les interfaces

comme: entree_dispositif.php, stock_dispositif.php et sortie_bdc_dispositif. php.

Gestion des redirections et comportement du bouton « Retour »

- **Problème :** lors de l'ajout d'un nouvel enregistrement, le bouton « Retour » du navigateur renvoyait l'utilisateur vers le formulaire rempli, provoquant parfois des doublons ou une confusion sur la validation.
- **Solution :** le fichier intermédiaire (insert_success.php) a été mis en place afin d'afficher un message de confirmation et éviter de retourner vers le formulaire initial, tout en assurant un meilleur contrôle de navigation.

Interface de sortie BDC (bon de commande)

- **Problème :** il était nécessaire de créer une interface qui permet d'effectuer plusieurs sorties en une seule soumission. Et les premières versions n'étaient pas suffisamment dynamiques ni adaptables aux besoins réels du terrain.
- **Solution :** un tableau interactif a été mis en place avec la possibilité d'ajout de lignes dynamiquement, de calcule automatique des prix totaux et de vérification des stocks, les dotations et les dates de péremption pour chaque dispositif avant toute soumission**.**

Surveillance automatique du stock et déclenchement d'alertes

- **Problème** : Il était nécessaire de détecter automatiquement les situations de stock critique ou de produits proches de la péremption, sans intervention manuelle.
- **Solution** : Deux triggers SQL (trigger_surveille_stock et trigger_surveille_ stock_update) ont été créés pour insérer dans la table alerte toute anomalie détectée en temps réel. Un script AJAX (get_alertes_actives.php) permet ensuite d'afficher un message sur l'interface Page de gestion des dispositifs, informant l'utilisateur des alertes en cours.

IV.8 Conclusion

Ce dernier chapitre représente l'aboutissement de notre démarche, tel qu'il retrace l'origine du projet, les étapes de réflexion, les choix technologiques ainsi que les difficultés rencontrées tout au long de la conception et du développement de notre application G-PHARMA.

En nous appuyant sur les connaissances acquise et observations lors de notre stage au sein de la pharmacie hospitalière de CHU de Tizi-Ouzou, nous avons conçu une solution qui répond aux besoins réels du terrain notamment pour le suivi des dispositifs médicaux. Le développement de l'application s'est articulé autour d'une architecture client-serveur, avec l'utilisation des technologies telles que PHP, MySQL, JavaScript et HTML/CSS, assurant une interface fonctionnelle et une gestion efficace des bases de données.

Nous avons structuré l'application autour de modules essentiels comme l'enregistrement des entrées et sorties, la dotation, l'alerte automatique…etc, tout en utilisant le langage SQL tel que : les vues, les déclencheurs et index afin d'assurer la cohérence des données.

Les choix de modélisation des bases de données et de la logique fonctionnelle de l'application, ont été guidés par une volonté de simplicité d'utilisation, de performance et d'adaptabilité à d'éventuelles extensions futures. Et malgré les contraintes techniques et les défis rencontrés notamment liés à l'intégration de plusieurs modules ou aux limitations de l'environnement de test local, nous avons su faire preuve d'adaptation et de persévérance afin d'aboutir à un prototype opérationnel conforme aux exigences initiales du projet.

En somme, ce chapitre illustre la capacité de passer de la théorie à la pratique en matérialisant une solution applicative pertinente, directement inspirée du terrain et techniquement maitrisée.

Conclusion générale

Conclusion générale

À l'issue de ce travail, nous avons pu démontrer l'importance de la digitalisation dans la gestion des pharmacies hospitalières, en particulier pour les dispositifs médicaux et les consommables, dont la traçabilité et la disponibilité sont cruciales pour la continuité des soins. L'étude des solutions existantes a permis de mettre en évidence certaines limites récurrentes telles que le manque d'intégration, l'absence de suivi en temps réel ou la difficulté d'adaptation aux besoins spécifiques du terrain.

Le développement de notre propre application G-PHARMA a concrétisé cette réflexion, mobilisant des compétences transversales en informatique (bases de données, développement web, architecture client-serveur) et en organisation hospitalière. Cette solution propose une pragmatique aux limites constatées sur le terrain, tout en s'alignant sur les standards de sécurité et de performance actuels.

Cependant, cette première version de l'application ne représente qu'une étape initiale. Plusieurs perspectives d'amélioration ont été identifiées pour étendre les fonctionnalités de la plateforme et renforcer son intégration dans l'environnement hospitalier :

- **Finalisation des sections médicaments et réactifs :** seule la partie «dispositif médicaux» a été entièrement développée. Les sections «médicaments» et «réactifs» ont été préparées en base de données, mais nécessite encore une intégration complète avec des fonctionnalités adaptées à chaque catégorie (suivie de lots, classes thérapeutiques…etc).
- **Connexion inter-services :** un objectif majeur est d'instaurer une communication directe entre les services hospitaliers et la pharmacie, en permettant aux chefs de service de soumettre leurs besoins via des BDC numérique. Cela supprimera les échanges papier et améliorera la traçabilité et la réactivité.
- **Suivi des stocks intra-services :** chaque service pourrait suivre en temps réel les produits qui lui sont attribués, favorisant une meilleure anticipation des ruptures, une responsabilisation accrue et une optimisation des réapprovisionnements.
- **Ajout d'un module de statistiques et de rapports :** l'intégration d'un générateur de rapports mensuels/annuels (PDF, Excel) sur les consommations, les ruptures et les entrées/sorties faciliterait la prise de décision stratégique par les responsables de la pharmacie.
- **Renforcement de la sécurité :** actuellement en local, l'application devra être migrée vers un environnement HTTPS avec une authentification renforcée, voir une vérification en deux étapes (2FA), surtout pour les profils sensibles.

- **Adaptation mobile et développement Android :** une version mobile ou Android permettrai, un accès rapide à certaines fonctions clés (vérification de stock, envoi d'alerte, validation de commande) directement sur le terrain via un smartphone ou tablette.

Ainsi, ce projet ouvre de nombreuses perspectives futures concrètes. Il constitue une base solide sur laquelle s'appuyer pour proposer des solutions innovantes et adaptées aux besoins réels des professionnels hospitaliers. Nous espérons que cette initiative contribuera à l'amélioration des pratiques numériques en milieu hospitalier et à la qualité de la prise en charge des patients.

Références bibliographiques

Références bibliographiques

[1] IBM, Gestion des stocks. [En ligne]. Disponible sur : https://www.ibm.com/fr-fr/topics/inventory-management. Consulter le : 25 Mars 2025.

[2] IBM Watson, Visibilité des stocks en temps réel. [En ligne]. Disponible sur: https://www.ibm.com/watson/supply-chain/resources/real-time-inventory-visibility. Consulter le : 25 Mars 2025.

[3] SER Group, Processus d'approvisionnement. [En ligne]. Disponible sur: https://www.sergroup.com/fr/blog/article/processus-d-approvisionnement.html. Consulter le: 25 Mars 2025.

[4] SER Group, Ibid., chapitre sur le circuit des produits pharmaceutiques, pp. 86Ŕ88.

[5] P. Zermati, Pratique de la gestion de stock, 5e éd., Paris, France : Dunod, 1996, p. 7.

[6] CHU Joffre-Dupuytren, La pharmacie hospitalière. [Document PDF].

[7] Ministère de la Santé, Arrêté n° 79/MSP du 24 août 1996 portant création de la pharmacie principale au sein des CHU.

[8] République Algérienne, Loi n°18-11 du 2 juillet 2018 relative à la santé, art. 247Ŕ 248.

[9] P. Bonnabry, Contribution de la pharmacie hospitalière à la qualité des soins, Université de Genève, 2001.

[10] ELECTROCLASS, G-Stock-Pharma Ŕ Logiciel de gestion de stock de médicaments. [En ligne]. Disponible sur : https://www.electroclass.com/web/sante/produits/logiciel-de-gestion-de-stock-de-medicaments/g-stock-pharma.html. Consulter le : 03 Avril 2025.

[11] ELECTROCLASS, G-Stock-Pharma Ŕ Solution-santé. [En ligne]. Disponible sur : https://www.electroclass.com/fr/solutions/solutions-sante/g-stock-pharma/. Consulter le : 03 Avril 2025.

[12] Epocrates, Clinical Decision Support. [En ligne]. Disponible sur: https://www.epocrates.com. Consulter le : 03 Avril 2025.

[13] Aptoide, Epocrates Ŕ Aptoide. [En ligne]. Disponible sur: https://epocrates.fr.aptoide.com/app. Consulter le : 03 Avril 2025.

[14] A. Pandey et al., "Epocrates as a decision support tool in pharmacotherapy," Journal of Pharmacology & Pharmacotherapeutics, vol. 8, n° 3, pp. 144Ŕ147, 2017. [En ligne]. Disponible : https://pmc.ncbi.nlm.nih.gov/articles/PMC5603444/. Consulter le : 03 Avril 2025.

[15] KLS Group, Logiciel WMS santé Hospilog – Solution logistique pour les établissements de santé, [En ligne]. Disponible sur : https://www.kls-group.fr/kls-medical/logiciel-wms-sante-hospilog/. Consulter le : 04 Avril 2025.

[16] KLS Group, Hospilog – Logiciel de gestion pour pharmacie hospitalière, [En ligne]. Disponible sur : https://www.kls-group.fr/kls-medical/hospilog-logiciel-pharmacie/. Consulter le : 04 Avril 2025.

[17] HOSPITALIA, Le CHU de Rennes choisit le WMS Gildas Hospilog, [En ligne], 2014. Disponible sur : https://www.hospitalia.fr/Le-CHU-de-Rennes-choisit-le-WMS-Gildas-Hospilog_a758.html. Consulter le : 04 Avril 2025.

[18] FAQ Logistiques, Le WMS Gildas Hospilog optimise la gestion logistique hospitalière, [En ligne], 12 Mars 2015. Disponible sur : https://www.faq-logistique.com/CP20150312-KLS-WMS-Gildas-Hospilog.htm. Consulter le 04 Avril 2025.

[19] Winpharma, Logiciel de gestion d'officine Winpharma. [En ligne]. Disponible sur : https://www.winpharma.com/. Consulter le : 06 Avril 2025.

[20] Foxeet, Logiciel de gestion d'officine Ŕ Winpharma. [En ligne]. Disponible sur : https://foxeet.fr/contenu/logiciel-gestion-officines-winpharma. Consulter le : 06 Avril 2025.

[21] I. Tazi-Bouziane, Intelligence artificielle appliquée aux systèmes d'information hospitaliers en Algérie, Oran, Algérie : CRASC, 2023, p. 10. [En ligne]. Disponible sur : https://www.ouvrages.crasc.dz/pdfs/intelligence-artificielle-appliquee-systmes-information-hospitaliers-algerie.pdf. Consulter le : 06 Avril 2025.

[22] ORACLE. [En ligne]. Disponible sur : https://www.oracle.com/. Consulter le : 11 Avril 2025.

[23] DIGORA. Introduction aux bases de données relationnelles Ŕ YouTube. [Vidéo en ligne]. YouTube, 2023. Disponible sur : https://www.youtube.com/watch?v=GGs3JYw9Eco. Consulter le : 11 Avril 2025.

[24] TEXTCORTEX. Donnée vs information : comprendre la différence. [En ligne]. TextCortex, 2024. Disponible sur : https://textcortex.com/fr/post/data-vs-information. Consulter le : 11 Avril 2025.

[25] ZEENEA. Comment bien comprendre la différence entre une donnée et une information ? [En ligne]. Zeenea, 2024. Disponible sur : https://zeenea.com/fr/comment-bien-comprendre-la-difference- entre-une-donnee-et-une-information/. Consulter le: 12 Avril 2025.

[26] DATA BIRD. [En ligne]. Data Bird, 2024. Disponible sur: https://www.data-bird.co/blog. Consulter le : 12 Avril 2025.

[27] AUDIBERT, Laurent. Cours sur les bases de données Ŕ Introduction aux bases de données. [En ligne]. Developpez.com, [s. d.]. Disponible sur: https://laurent-audibert.developpez.com/Cours- BD/?page=introduction-bases-de-donnees. Consulter le : 12 Avril 2025.

[28] DIGORA. Les différents types de bases de données en 2024 : un guide complet. [En ligne]. Digora, 2024. Disponible sur : https://www.digora.com/fr/les-differents-types-de-bases-de-donnees-en-2024- un-guide-complet. Consulter le : 15 Avril 2025.

[29] INTELLIGENCE ARTIFICIELLE SCHOOL. NoSQL : tout comprendre sur cette database non relationnelle. [En ligne]. IAS, 2024. Disponible sur : https://www.intelligence-artificielle- school.com/ecole/technologies/nosql-tout-comprendre-sur-cette-database-non-relationnelle/. Consulter le : 15 Avril 2025.

[30] MONGODB. NoSQL explained. [En ligne]. MongoDB, 2024. Disponible sur: https://www.mongodb.com/fr-fr/resources/basics/databases/nosql-explained. Consulter le : 16 Avril 2025.

[31] TALEND. Cycle de vie des données. [En ligne]. Talend, 2024. Disponible sur: https://www.talend.com/fr/resources/cycle-vie-donnees/. Consulter le : 16 Avril 2025.

[32] Inconnu. Conception d'une base de données Ŕ MCD, MLD, MPD. [En ligne]. Scribd, 2017. Disponible sur : https://fr.scribd.com/document/373465262/Conception-d-Une-Bdd-Mcd-Mld-Mpd. Consulter le : 16 Avril 2025.

[33] VANDEVELDE, Louis. Méthodologie de conception MERISE. [En ligne]. [s. l.] : Louis Vandevelde, [s. d.]. Disponible sur: https://louisvandevelde.be/index.php?dos=my&fic=meris. Consulter le : 19 Avril 2025.

[34] LUCIDCHART. Langage UML : Définition et usage. [En ligne]. Lucidchart, [s. d.]. Disponible sur : https://www.lucidchart.com/pages/fr/langage-uml. Consulter le : 19 Avril 2025.

[35] OPENCLASSROOMS. [En ligne]. Disponible sur: https://openclassrooms.com/fr/. Consulter le : 19 Avril 2025.

[36] ESLSCA. Qu'est-ce qu'un système de gestion de base de données (SGBD) ? [En ligne]. ESLSCA Business School, [s . d.]. Disponible sur : https://www.eslsca.ma/blog/quest-ce-quun-systeme-de- gestion-de-base-de-donnees-sgbd. Consulter le: 21 Avril 2025.

[37] MYSQL. MySQL 8.0 Reference Manual Ŕ Introduction. [En ligne]. Oracle, 2024. Disponible sur : https://dev.mysql.com/doc/refman/8.0/en/introduction.html. Consulter le : 21 Avril 2025.

[38] MICROSOFT. Microsoft Learn, 2024. [En ligne]. Disponible sur : https://learn.microsoft.com/fr-fr/. Consulter le : 21 Avril 2025.

[39] POSTGRESQL. About PostgreSQL. [En ligne]. PostgreSQL Global Development Group, 2024. Disponible sur : https://www.postgresql.org/about/. Consulter le : 21 Avril 2025.

[40] THOME, Nicolas. Cours SGBD 1 Ŕ Introduction aux bases de données. [PDF en ligne]. Université Côte d'Azur, [s.d.]. Disponible sur: https://webusers.i3s.unice. fr/~nlt/cours /licence/sgbd1/sgbd1_cours.pdf. Consulter le: 22 Avril 2025.

[41] MINNICK, Chris, ABRAHAM, Nat et HOLLAND, Eva. Programmation Web pour les Nuls. Traduit de l'anglais par Damien MANIEZ. Paris : First Interactive, 2023. (Œuvre originale publiée en 2023).

[42] USE THE INDEX, LUKE ! Jointure de hachage. [En ligne]. [s. d.]. Disponible sur: https://use- the-index-luke.com/fr/sql/l-operation-de-jointure/jointure-de-hachage. Consulter le : 22 Avril 2025.

[43] DATASCIENTEST. Index SQL : tout savoir. [En ligne]. [s.d.]. Disponible sur : https://datascientest.com/index-sql-tout-savoir. Consulter le: 22 Avril 2025.

[44] SQLSHACK. SQL Server indexed views, [En ligne]. [s. d.]. Disponible sur : Consulter le: 25 Avril 2025.

[45] DATACAMP. Comprendre les transaction ACID en SQL. [En ligne]. DataCamp Blog, 2024. Disponible sur : https://www.datacamp.com/fr/blog/acid-transactions. Consulter le : 25 Avril 2025.

[46] SQL.SH. CREATE TRIGGER – Déclencheur SQL. [En ligne]. SQL.sh, 2024. Disponible sur : https://sql.sh/cours/create-trigger. Consulter le : 26 Avril 2025

[47] BASES DE DONNEES. Intégrité référentielle. [En ligne]. [s. d.]. Disponible sur : https://www.base-de-donnees.com/integrite-referentielle/. Consulter le: 26 Avril 2025.

[48] IBM. Contraintes référentielles. [En ligne]. [s. d.]. Disponible sur: https:// www.ibm.com/docs/fr/i/7.5.0?topic=constraints-referential. Consulter le: 27 Avril 2025.

[49] DATASUNRISE, [En ligne]. Disponible sur : https://www.datasunrise.com/fr/. Consulter le : 27 Avril 2025.

[50] NINJAONE. Comment sauvegarder et restaurer une base de données de SQL Server ? [En ligne]. 2023. Disponible sur : https://www.ninjaone.com/fr/blog/comment-sauvegarder-et-restaurer-une-base- de-donnees-de-sql-server/. Consulter le: 29 Avril 2025.

[51] IONOS. Les types de sauvegardes : quelles différences ? [En ligne]. Disponible sur:https://www.ionos.fr/digitalguide/serveur/know-how/les-types-de-sauvegardes/. Consulter le : 29 Avril 2025.

[52] IDEEMATIC, [En ligne]. Disponible sur : https://www.ideematic.com/. Consulter le : 31 mai 2025.

[53] MDN Web Docs, [En ligne]. Disponible sur : https://developer.mozilla.org/fr/. Consulter le : 31 mai 2025.

[54] MY LITTLE BIG WEB, Définition d'un site web : qu'est-ce qu'un site web ?, [En ligne] : https://mylittlebigweb.com/blogue/definition-site-web/, consulter le : 31 mai 2025.

[55] GOOGLE DEVELOPERS, Mobile App vs. Web App, [En ligne] : https://developer.android.com/guide, consulter le : 31 mai 2025.

[56] FREECODECAMP, What is JavaScript? [En ligne]: https://www.freecodecamp.org/news/what-is-javascript/, Consulter le : 31 mai 2025.

[57] Audra Hendrix, Bogdan Brinzarea, Cristian Darie, AJAX et PHP Ŕ Comment construire des applications web réactives, Pearson Education France, 2007.

[58] Node.js, About Node.js. [En ligne] : https://nodejs.org/en/about, Consulter le : 2 juin 2025.

[59] Python Software Foundation, About Python, [En ligne]: https://www.python.org/about, Consulter : le 2 juin 2025.

[60] MACÉ, Noël. Développement et architecture des applications Web modernes : retrouver les fondamentaux. Saint-Herblain : Éditions ENI, 2021. 422p. ISBN 978-2-409-02790-0.

[61] GOOGLE DEVELOPERS, Angular Ŕ Introduction, [En ligne] : https://angular.io, consulter le : 6 juin 2025.

[62] Vue.js, Vue Documentation, [En ligne] : https://vuejs.org, consulter le : 6 juin 2025.

[63] Laravel, About Laravel Framework, [En ligne] : https://laravel.com/, Consulter le : 8 juin 2025.

[64] Express.js, Introduction to Express, [En ligne] : https://expressjs.com/, Consulter le : 8 juin 2025.

[65] Django Software Foundation, About Django, [En ligne] : https://www.djangoproject.com/, Consulter le : 8 juin 2025.

[66] WordPress.org, About WordPress, [En ligne] : https://wordpress.org/about/, Consulter le : 8 juin 2025.

[67] Bubble.io, What is Bubble?, [En ligne] : https://bubble.io/, Consulter le : 8 juin 2025.

[68] Apache Friends, XAMPPŕIntroduction, [En ligne]: https://www.apachefriends.org/index.html, consulter le : 9 juin 2025.

[69] Apache Software Foundation, About Apache HTTP Server, [En ligne] : https://httpd.apache.org/, consulter le : 9 juin 2025.

[70] Nginx, Overview of Nginx Architecture, [En ligne]: https://www.nginx.com/resources/glossary/nginx/, Consulter le : 9 juin 2025.

[71] Docker, What is a Container?, [En ligne] : https://www.docker.com/resources/what-container/, Consulter le : 9 juin 2025.

[72] OWASP Foundation, OWASP Top Ten Web Application Security Risks, [En ligne]: https://owasp.org/www-project-top-ten/, Consulter le : 8 juin 2025.

[73] Auth0, Authentication and Authorization, [En ligne]: https://auth0.com/docs/authenticate, Consulter le : 8 juin 2025.

[74] Cloudflare, What is HTTPS?, [En ligne]: https://www.cloudflare.com/learning/ssl/what-is-https/, Consulter le : 8 juin 2025.

Annexe

ANNEXE

Annexe A – Installation et fonctionnement de XAMPP

A.1 Téléchargement de XAMPP

Figure A.1 : Page de téléchargement officielle de XAMPP.

A.2. Installation de XAMPP (étapes)

1. **Lancer l'exécutable `xampp-windows-x64-xx-installer.exe`**
2. **Choisir les composants à installer (laisser tous cochés par défaut)**

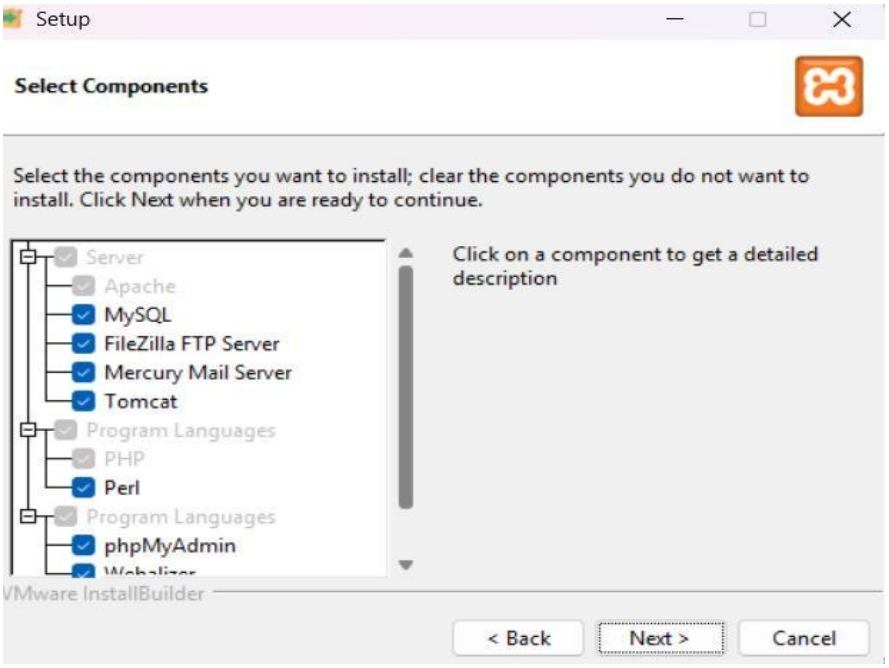

Figure A.2 : Écran de sélection des composants à installer

3. Choisir le dossier d'installation (par défaut : `C:\xampp`)

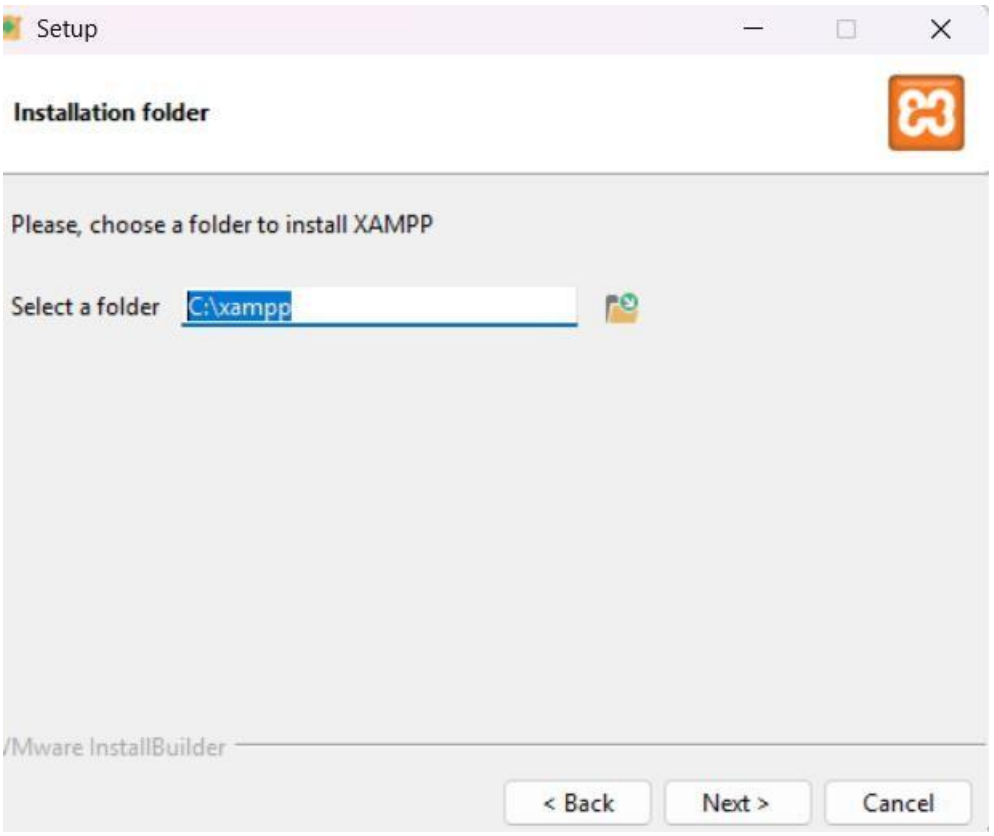

Figure A.3 : Choix du dossier d'installation de XAMPP

4. Terminer l'installation et lancer le panneau de contrôle XAMPP

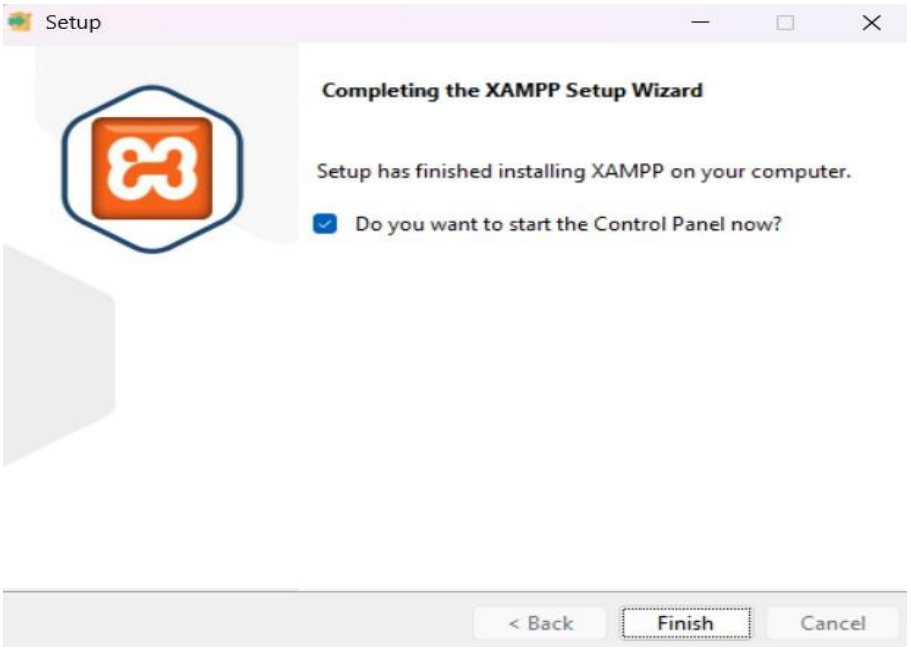

Figure A.4 : Écran de fin d'installation de XAMPP

A.3. Utilisation du panneau de contrôle XAMPP

Le panneau de contrôle de XAMPP permet de démarrer ou arrêter les services **Apache** (serveur web) et **MySQL** (base de données).

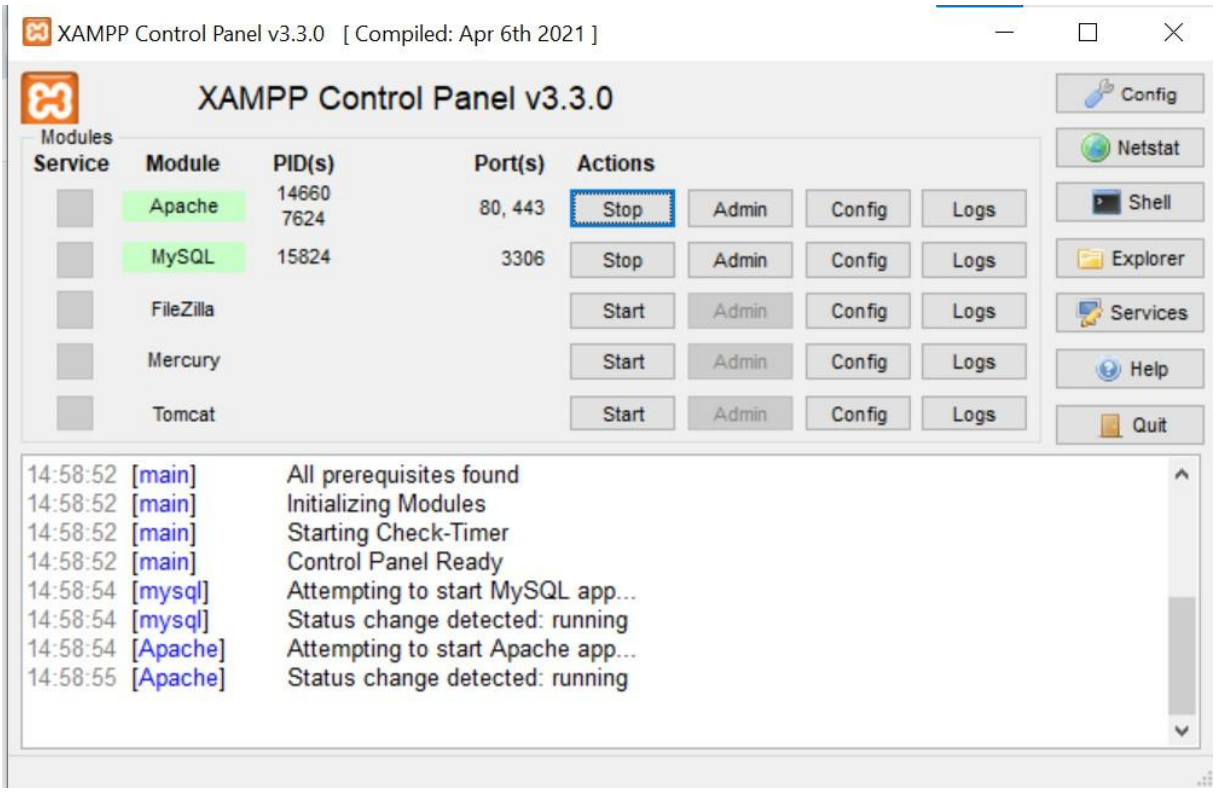

Figure A.5 : Panneau de contrôle XAMPP avec Apache et MySQL activés

A.4. Accès à phpMyAdmin

phpMyAdmin est un outil graphique permettant de gérer les bases de données MySQL/MariaDB via un navigateur. Il est accessible en local à l'adresse http://localhost/phpmyadmin.

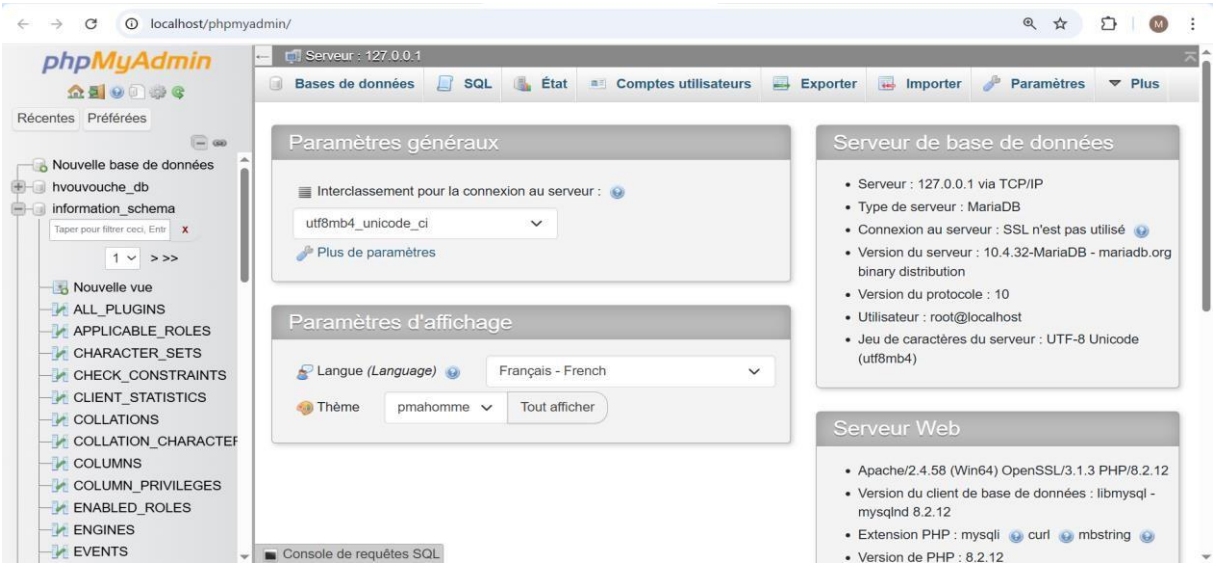

Figure A.6 : Page d'accueil de phpMyAdmin sur `localhost/phpmyadmin`

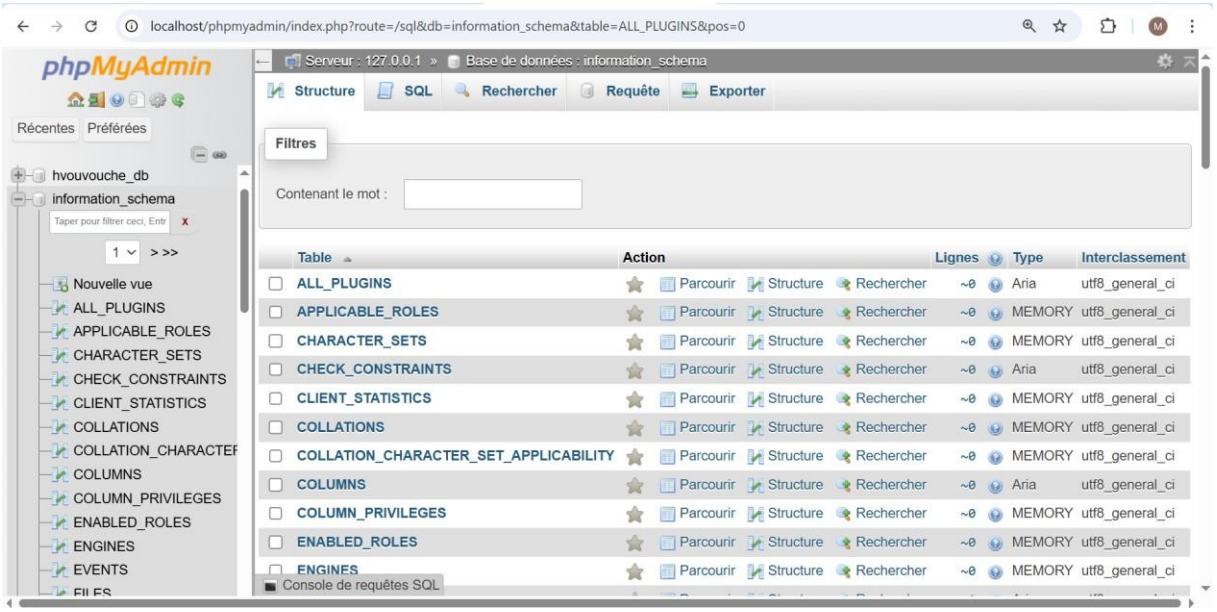

Figure A.7 : Interface de navigation dans les bases de données via phpMyAdmin (`information_schema`)

Annexe B – Installation et utilisation de Visual Studio Code (VS Code)

B.1. Téléchargement de VS Code

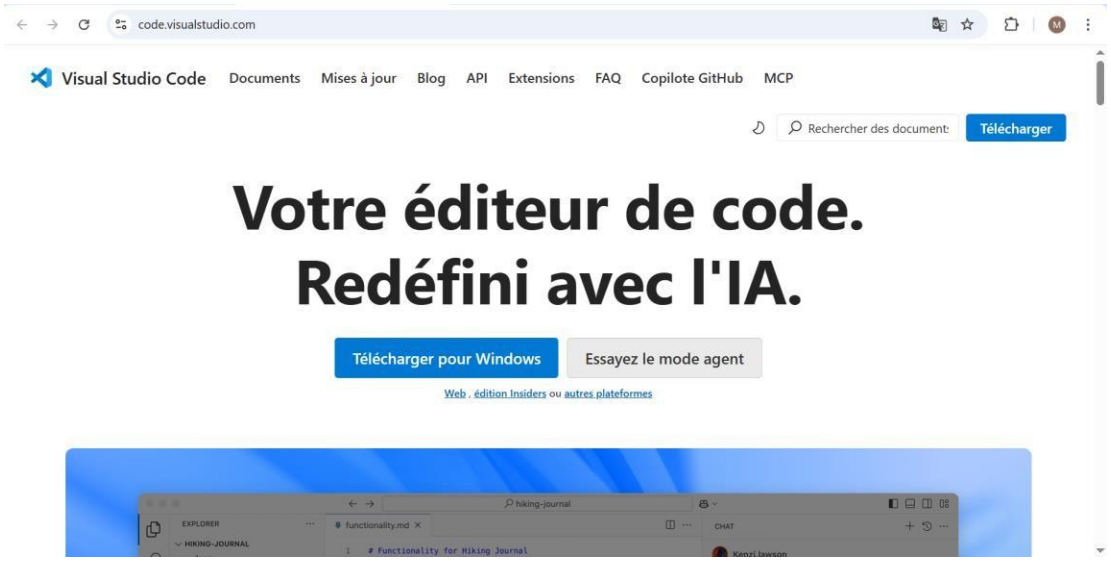

Figure B.1 : Page officielle de téléchargement de Visual Studio Code sur le site `code.visualstudio.com`

B.2. Installation de VS Code

1. **Lancer l'exécutable** `VSCodeUserSetup-x64.exe`
2. **Accepter les termes du contrat de licence**
3. **Choisir l'emplacement d'installation**
4. **Laisser les options par défaut (y compris "Ajouter à la variable PATH")**
5. **Terminer l'installation**

B.4. Lancement de VS Code et découverte de l'interface

Lors du premier lancement, VS Code propose une page d'accueil avec des raccourcis pour ouvrir un dossier, créer un fichier, ou installer des extensions.

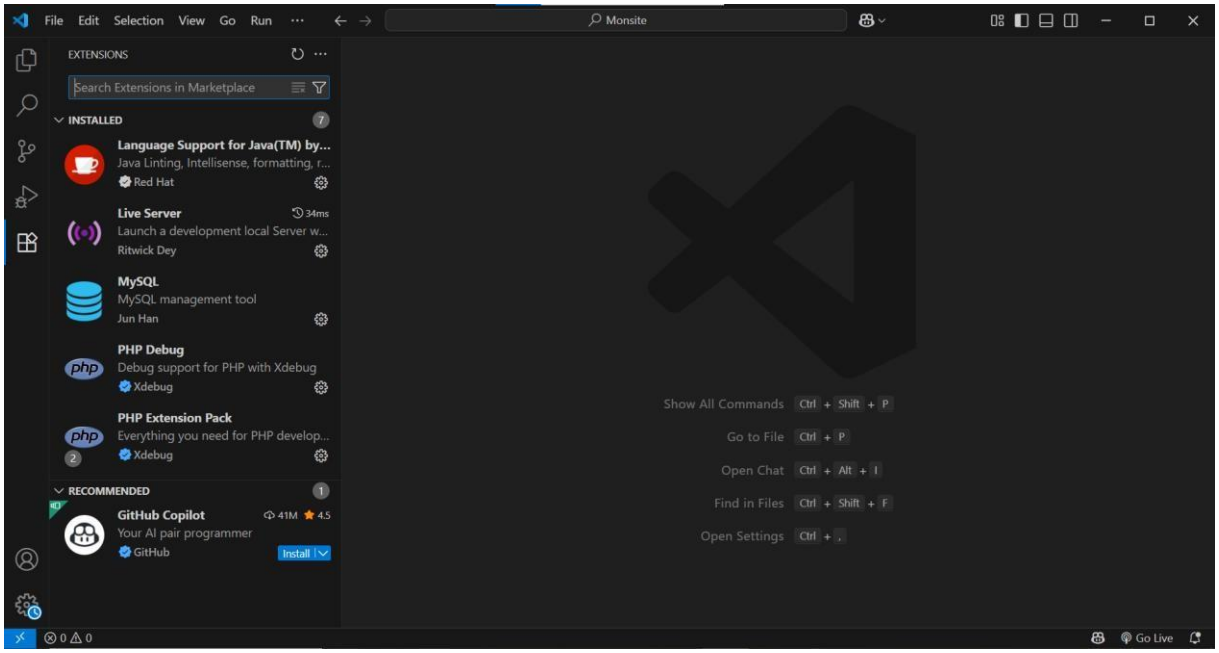

Figure B.2 : Interface de Visual Studio Code avec les extensions installées (MySQL, PHP Debug, Live Server…)

B.5. Ouverture du dossier de projet

Pour travailler sur un projet local, il suffit d'ouvrir le dossier contenant les fichiers (dans C:\xampp\htdocs\mon_projet) via **Fichier > Ouvrir un dossier**.

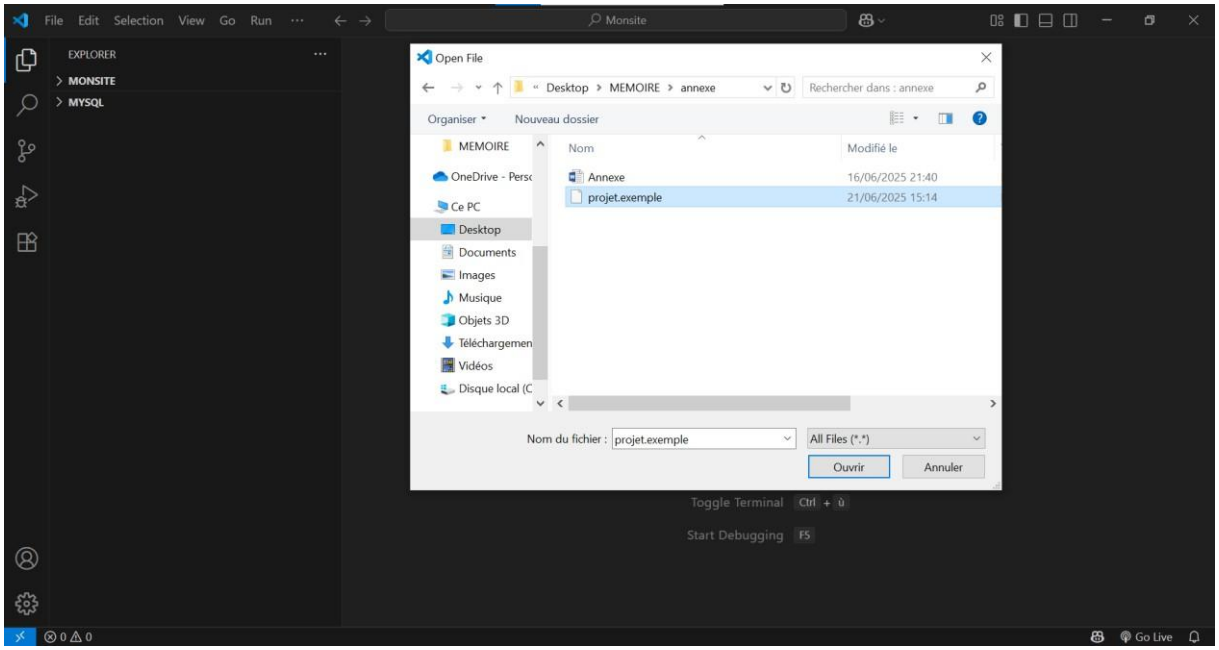

Figure B.3 : Ouverture d'un dossier de projet local dans Visual Studio Code via la boîte de dialogue "Open File"

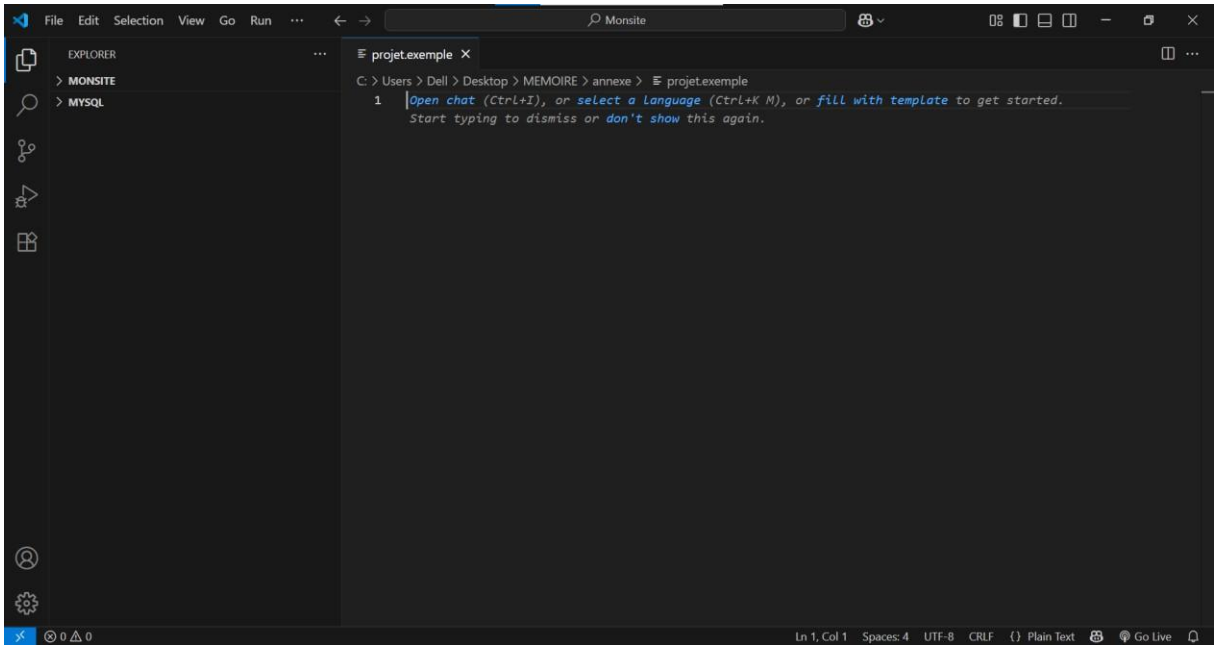

Figure B.4 : Visual Studio Code affichant un fichier ouvert dans un projet local (projet.exemple)

B.7. Avantages de VS Code dans ce projet

- **Intégration facile avec XAMPP** (via l'ouverture du dossier `htdocs`)
- **Aide au codage** avec autocomplétion et coloration syntaxique
- **Possibilité de travailler avec plusieurs fichiers et langages en même temps**
- **Extensions nombreuses et gratuites,** adaptées à chaque étape du projet

Annexe C – Glossaire des termes techniques utilisés dans l'application

Cette annexe présente une explication simple et claire de certains termes techniques, bibliothèques ou outils utilisés dans la conception et le développement de l'application de gestion de la pharmacie hospitalière.

TomSelect :

TomeSelect est une bibliothèque JavaScript qui est faite pour améliorer les éléments <select> HTML standards. Parmi ses fonctionnalités on peut citer :

- Recherche dynamique intégrée, avec surlignage automatique des correspondances,
- Support de la sélection multiple avec suppression intuitive,
- Création d'options à la volée (utile pour les champs libres),
- Tri**,** filtrage et affichage personnalisés des résultats [1].

```
< input  id =" tom-select-it " />
< link  rel =" stylesheet " href =" /css/tom-select.default.css " >
< script  src =" /js/tom-select.complete.js " > </ script >
< script >
var  options  =  { } ;
tomSelect ( '#tom-select-it' , { } ) ;
</ script >
```

Figure C.1 : Exemple minimal d'intégration de la bibliothèque Tom Select dans une page HTML [1].

jsPDF : jsPDF est une bibliothèque JavaScript qui génère des fichier PDF directement depuis le navigateur. Elle permet d'exporté l'interface ou les données affichées coté client sous forme de document PDF, pour enregistrement [2].

API (Application Programming Interface)

Une **API** (*Application Programming Interface*) est un ensemble de règles et de protocoles qui permettent à différentes applications logicielles de communiquer entre elles. Elle agit comme une interface intermédiaire entre deux systèmes, en facilitant l'échange de données ou de fonctionnalités sans que les développeurs aient besoin de comprendre ou de modifier le code source sous-jacent [3].

Selon Red Hat, *« une API permet aux développeurs de créer des applications plus facilement en fournissant les briques logicielles déjà prêtes à l'emploi, comme l'accès à une base de données, l'envoi de messages ou l'authentification d'un utilisateur »* [3].

JSON (JavaScript Object Notation)

JSON est un format d'échange de données léger et facile à lire, utilisé pour structurer des données sous forme de texte. JSON est à la fois lisible par l'humain et facilement analysable par les machines, ce qui en fait un format très utilisé dans les échanges entre applications web, en particulier lors des communications avec des API [4].

[1] npmjs, *Tom Select v1.0.0-b.1*, [en ligne] : https://www.npmjs.com/package/tom-select/v/1.0.0-b.1, consulter le 22 juin 2025.

[2] NPM, *jsPDF - Generate PDF files in client-side JavaScript*, [en ligne] : https://www.npmjs.com/package/jspdf, consulter le 22 juin 2025.

[3] Red Hat, *Qu'est-ce qu'une interface de programmation d'applications (API)*, [en ligne] : https://www.redhat.com/fr/topics/api/what-are-application-programming-interfaces, consulter le 22 juin 2025.

[4] JSON.org, *Introducing JSON (JavaScript Object Notation)*, [en ligne] : https://www.json.org/json-en.html, consulter le 22 juin 2025.