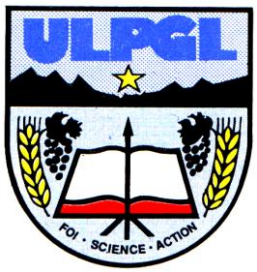
UNIVERSITE LIBRE DES PAYS DES GRANDS LACS

FACULTE DE SCIENCES ET TECHNOLOGIES APPLIQUEES

DEPARTEMENT DE GENIE ELECTRIQUE ET INFORMATIQUE



BP. 368 GOMA

www.ulpgl.net

CONCEPTION ET REALISATION D'UNE APPLICATION DE GESTION D'UNE PHARMACIE MEDICALE : cas de l'officine pharmaceutique Gracia

Par **MUSHAGALUSA BIRINGANINE Roger**

Travail présenté en vue de l'obtention du Diplôme de Gradué en Sciences et Technologies Appliquées

Option : Génie Electrique et Informatique

Directeur : Prof. **DI Dr Tech NKIEDIEL Alain AKWIR**

Encadreur : Ir. **Johnson KISAMBA**

ANNEE ACADEMIQUE 2021 - 2022

Dédicace

A mes très chers parents

Pour tout ce que vous m’avez donné, pour tout ce que vous avez fait pour moi.

Remerciements

Le présent travail étant le fruit des efforts consentis par certaines personnes, nous ne pourrions pas passer sans pour autant leurs rendre hommage ou leurs adressés nos sincères remerciements.

A Dieu tout puissant, le maitre du temps et des circonstances, pour la protection à notre égard au cours de notre recherche. Que puissance, honneur et la gloire lui soient rendus.

Nous remercions le DI. Dr. Tech NKIEDIEL Alain AKWIR et Ass. IR. Johnson KISAMBA pour avoir accepté de diriger et d'encadrer respectivement ce travail malgré leurs multiples occupations et d'autres tâches leurs encombrant pour le bien de la communauté.

Nous remercions notre chère université ULPGL/Goma pour nous avoir nourris des connaissances durant toutes ces années passées.

Nos sincères remerciements s'adressent à nos très chers parents BIRINGANINE NKUNZI Jonas et CIRHABWIRA CISHUGI Julienne qui par leurs efforts consentis nos sommes arrivés à ce stade.

Nous remercions également toute la famille MUYISA qui nous a fourni les soutiens de toutes natures sans lesquelles nous ne pourrions nullement arriver à ce niveau.

Nous remercions également nos camarades KAMBOYA KYALOMDAWA Héritier, KAMBALE KABUYAYA Promesse et MUGISHO CIRINDYE Frederick pour leurs participations respectives dans la finalisation du présent travail.

Et à toute autre personne dont nous n'avons pas citer le nom et nous a épaulée de loin ou de près dans la réalisation du présent travail, nous lui sommes infiniment reconnaissant.

Résumé

Un système informatique est un bon outil dans la gestion d'une pharmacie médicale pour permettre aux différents employés de suivre l'évolution du stock et en temps réel et faire un classement bien détaillé des activités liées à la vente et à l'approvisionnement de la pharmacie. Au cours de ce travail nous avons réalisé une application web dédié à la gestion de la pharmacie Gracia. Cette application web permet de faciliter le travail du pharmacien et du gérant en automatisant la gestion du stock, des ventes. Notre application web offre aussi la possibilité au gérant de lister les ventes et les produits en voie de péremption ou de rupture. Ce travail a été réalisé en utilisant le processus de développement "Processus Unifier" (UP) et le langage de modélisation UML afin de schématiser la solution. Nous avons choisi de programmer l'application avec le langage PHP avec le Framework Laravel et MYSQL comme SGBD.

Abstract

Computer systems are good tools in a management pharmacy to allow the different employees to follow the state of the stock in real time and make a real detailed classification of activities related to the sale and supply of the pharmacy Gracia. During the work we have realized a web application dedicated to the management of the pharmacy Gracia. This web application makes it possible to facilitate the work of the pharmacist and the manager by automating the management stock and sales. This application also offers the possibility to manager and pharmacist to list sales and products in process of expiry or discontinuation. This work was done using the Unified Process and the UML in order to schematize the solution. We also choose to program the application with PHP programming language and his framework Laravel and MySQL as SGBD

Sommaire

Liste des abréviations

CRUD	Create Read Update Delete
I/O	Input Output
LAMP	Linux Apache MySQL PHP, Perl, Python
PHP	PHP: Hypertext Preprocessor
RAM	Random Access Memory
SQL	Structured Query Language
UML	Unified Modeling Langage
UP	Unified Process

Liste des tableaux

Liste des figures

1. Introduction générale

1.1. Contexte

Actuellement dans notre société, nous assistons à une augmentation de l'utilisation des services numériques (application desktop, mobiles, réseaux sociaux, internet, …) et tout le monde est déjà à l'heure du numérique.

Les pharmacies médicales font partie intégrante des établissements que l'informatique pourra beaucoup aider. Jusqu'à ce jour, la gestion manuelle de l'information y est encore dominante ; d'où la nécessité d'y introduire l'informatique.

1.2. Identification et formulation du problème

Depuis toujours, avant l'invention de l'ordinateur, la gestion de l'organisation pose de sérieux problèmes de surveillance relative à la saisie, à la collecte, au traitement et à la diffusion des informations.

Avant on enregistrait toutes les informations manuellement sur des supports en papier ; ce qui engendrait beaucoup de problèmes tel que la perte considérable de temps dans la recherche de ces informations ou la dégradation de ces dernières, etc.

Considérant les principaux besoins à couvrir au sein des pharmacies, une tendance se dessine pour l'harmonisation des supports utilisés, ceci pour palier des problèmes fastidieux d'organisation et d'archivage des données. L'objectif du projet présenté dans ce travail est de concevoir et réaliser une application simple en vue d'informatiser la pharmacie et gérer la vente et la maintenance des médicaments dans une pharmacie.

Le système traditionnel ne peut, d'une manière automatique :

- Donner en temps réels l'état du stock de la pharmacie
- Classer l'état du stock par catégories

1.3. Questions de recherche

Ayant une formulation de recherche déjà établie, on peut se poser les questions suivantes :

- Comment faire pour avoir un système qui peux gérer les activités de la pharmacie dans un très bref délai tout en limitant le cout des dépenses de la main d'œuvre ?
- Comment classer l'état de stock de la pharmacie en tenant compte de la différence et de la ressemblance entre les médicaments ?
- Comment déterminer le temps de réapprovisionnement ?

1.4. Formulation des hypothèses

Étant donné que les hypothèses sont des réponses anticipées aux questions spécifiques d'une recherche, il sied d'évoquer les hypothèses ci-dessous, considérant que l'hypothèse relative à la question principale est que l'informatisation du système traditionnel actuel influencerait de manière considérable la pharmacie médicale dans le sens qu'elle lui ferait gagner en temps et en sécurité, lesquels sont par ailleurs parmi des éléments fondamentaux pour un bon fonctionnement d'une entreprise :

- Elle permettrait grâce à sa base de données numérique une classification, une organisation des données selon plusieurs critères. Mais aussi un suivi de l'état du stock selon une façon bien définie par l'utilisateur.
- Elle permettrait aussi l'organisation, la centralisation et la conservation de l'information ; ce qui faciliterait l'accès rapide aux données et un gain de temps considérable tout en diminuant le cout pour accéder aux données.

1.5. Justification du choix du sujet et motivations

Le temps est un des facteurs importants dans la productivité d'une entreprise. L'automatisation des tâches participe ainsi au gain du temps, et permet à l'entreprise d'être compétitive.
Théoriquement ce travail fera partie de la littérature et des travaux antérieurs qui serviront de base aux futurs chercheurs dans le domaine de la gestion informatique.

Etant un grand passionné par la programmation, ce travail me permettra d'approfondir mes connaissances en programmation surtout dans l'élaboration des applications de Gestion et aussi dans la conception des bases de données.

1.6. Énoncé des objectifs de recherche

1.6.1. L'objectif général

En nous basant sur les hypothèses formulées ci-haut, l'objectif générale de notre travail est la réalisation d'une application web permettant la gestion d'une pharmacie médicale.

1.6.2. Les objectifs opérationnels/spécifiques

En concevant notre système, nous nous baserons premièrement sur la partie pharmacien et la partie administrateur car dans toute pharmacie le pharmacien est le cœur de l'entreprise et notre travail lui permettra de bien mener son travail.

Nous allons réaliser une application qui permettra la gestion complète d'une pharmacie tout en respectant les normes de gestion d'une pharmacie médicale.

1.7. Méthodologie et délimitation du travail

Pour la méthodologie nous allons utiliser la comparaison et l'analyse du système de fonctionnement d'une pharmacie. La modélisation UML nous permettra de bien cerner les différentes parties de notre application et leurs utilités.

Notre travail sera mené en générale sur un système de gestion d'une pharmacie, et notre environnement d'étude est l'Officine Pharmaceutique Gracia

Voici la liste des techniques et méthodes utilisées dans ce travail :

- **Méthode analytique :** cette méthode nous a permis de décomposer le système d'information actuel pour bien faire la modélisation du nouveau système sans trop modifier sa structure.
- **Le processus Unifié (UP : Unified Process) :** cette méthode permet l'implémentation et le développement majoritairement utilisées dans le développement informatique. Il permet aux développeurs de transformer les besoins du client en une application (fonctionnalités). Ce processus utilise le langage UML qui est sa partie intégrante.

- **L'interview :** cette technique a consisté à interroger les différents acteurs intervenant dans la gestion de l' Officine Pharmaceutique Gracia.
- **La technique d'analyse documentaire :** cette technique nous a permis de récolter quelques informations dans les archives du système d'information actuel.

1.8. Subdivision du travail

Notre travail sera subdivisé en 3 chapitres en omettant l'introduction et la conclusion

- Chap. 1 : Cadre théorique et analyse de l'existant : qui consiste en la définition des termes clés que nous allons utiliser au cours de la réalisation de notre projet, il consiste aussi à présenter l'organisation qui fait l'objet de notre étude.

- Chap. 2 : modélisation de la solution retenue : elle consiste en la présentation des résultats de l'analyse du système.

- Chap. 3 : implémentation du nouveau système : consiste en la présentation des résultats obtenus après concrétisation du model conceptuel de notre système.

Chapitre 1 Cadre théorique et analyse de l'existant

1.1 Introduction

Dans ce chapitre, nous allons présenter quelques généralités concernant les concepts de base, les méthodes et les outils nécessaires pour la réalisation de notre application. En effet, il s'agit d'éclairer nos lecteurs sur la signification des concepts de base en vue de faciliter la compréhension du contenu de ce travail. C'est dans cette partie que nous présenterons le système actuel en apportant nos quelques critiques sur l'existant, nous présenterons aussi quelques propositions de solutions et une analyse du système future.

1.2 Concepts de base du sujet

Dans cette section, nous allons nous intéresser aux quelques concepts fondamentaux auxquels se focalise notre travail

1.2.1 Concepts de base

1.2.1.1 Information et système d'information [1]

Au sens étymologique, l'information est ce qui donne forme à l'esprit. Elle vient du verbe latin *informare,* qui signifie « donner forme à » ou « se former une idée de ». L'information désigne à la fois le message à communiquer et les symboles utilisés pour l'écrire ; elle utilise un code de signes porteurs de sens tels qu'un alphabet de lettres, une base de chiffres, des idéogrammes ou pictogrammes. Hors contexte, elle représente le véhicule des données comme dans la théorie de l'information ; et, hors support, elle représente un facteur d'organisation.

Une information est un élément qui permet de compléter la connaissance sur une personne, un objet, un événement. C'est aussi un ensemble d'éléments de connaissance susceptibles d'être codés pour être conservés, traités ou communiqués.

Elle est aussi l'émission ou la réception de signaux oraux, ou écrits, sonores, visuels ou multimédia dont le but est de déclencher les processus alimentant l'échange, base naturelle et indispensable de l'animation de l'organisation.

Dans certains cas, un ensemble d'informations peut se référer à un domaine précis ; cela constitue alors ce qu'on appelle un système d'informations. La plupart de systèmes actuels se basent sur la technologie de l'informatique. Cependant il existe encore de systèmes dans lesquels les informations sont stockées, manipulées et communiquées traditionnellement ; on peut citer parmi eux : les armoires, les fiches en papier ou les registres. Il existe une différence entre information et données comme cela peut être illustré par l'exemple suivant :

Dans une librairie, un client demande au vendeur si le livre " Apprendre à développer un site web avec PHP et MySQL (Livre de Olivier Rollet) est disponible en stock. Le vendeur consulte la base de données de la librairie à l'aide de son ordinateur et confirme au client que le livre est disponible. Le vendeur a donc donné au client l'information que le livre est en stock. Afin de pouvoir donner cette information, le vendeur a dû consulter les données qui représentent le stock de la librairie. Le fait de consulter le stock constitue un traitement sur les données du stock. [2]

Dans les technologies de l'information, une donnée est la représentation d'une information dans un programme : soit dans le texte du programme (*code source*), soit en mémoire durant l'exécution. Les données, souvent codées, décrivent les éléments du logiciel tels qu'une entité (chose), une interaction, une transaction, un événement, un sous-système, etc.

Les données peuvent être conservées et classées sous différentes formes : textuelles (chaîne), numériques, images, sons, etc.

1.2.1.2 Base des données [2]

Une base des données est un ensemble structuré organisé d'informations avec un objectif commun permettant le stockage de grandes quantités d'informations afin d'en faciliter les différentes manipulations ; c'est notamment le cas de l'ajout, de la mise à jour et éventuellement la recherche de données.

Une base de données informatique est un ensemble de données qui ont été stockées sur un support informatique, et organisées et structurées de manière à pouvoir facilement consulter et modifier leur contenu.

La gestion et l'accès à une base de données sont assurés par un ensemble de programmes qui constituent le Système de gestion de base de données (SGBD).

1.2.1.3 Système de gestion de base de données [3]

Un Système de Gestion de Base de Données (SGBD) est un logiciel (ou un ensemble de logiciels) permettant de manipuler les données d'une base de données. Manipuler, c'est-à-dire sélectionner et afficher des informations tirées de cette base, modifier des données, en ajouter ou en supprimer (ce groupe de quatre opérations étant souvent appelé "CRUD", pour Create, Read, Update, Delete).

Il existe de nombreux systèmes de gestion de bases de données ; quelques-uns de ces systèmes sont : PostgreSQL, MySQL, Oracle, IBM DB2, Microsoft SQL, Sybase, Informix, …

1.2.1.3.1 Le paradigme client – serveur [4]

L'environnement client/serveur désigne un mode de transaction entre plusieurs programmes ou processus : l'un qualifié de client envoie des requêtes, l'autre qualifié de serveur, attend les requêtes des clients et y réponds. Le serveur offre ici un **service** au client. Par extension, le client désigne souvent l'ordinateur sur lequel est exécuté le logiciel client et le serveur, l'ordinateur sur lequel est exécuté le logiciel serveur. Les machines serveurs sont généralement dotées de capacités supérieures à celles des ordinateurs personnels en ce qui concerne la puissance de calcul, les I/O et les connexions réseaux, afin de pouvoir répondre de manière efficace à un grand nombre de clients, les clients sont souvent des ordinateurs personnels ou terminaux individuels (téléphones, tablettes), mais pas systématiquement. Un serveur peut répondre aux requêtes de plusieurs clients, parfois le client et le serveur peuvent être sur la même machine.

1.2.1.3.2 MySQL [5]

MySQL est un SGBD Relationnelle open source qui s'appuie sur le langage SQL. Il est compatible avec presque toutes les plates formes notamment Linux, Mac et Windows. Utilisé pour toutes sortes d'applications, il est le plus souvent associé aux applications web et à la publication de contenus en ligne.

MySQL est un composant important de la pile d'entreprise open source dite LAMP. LAMP étant une plateforme de développement web qui utilise Linux comme IOS, Apache comme serveur web, MySQL comme SGBDR et PHP comme langage de script orienté objet (Perl ou Python sont parfois utilisés à la place de PHP).

1.2.1.3.3 SQL [1]

SQL est une abréviation anglaise de « Structured Query Language ». En français, il s'agit du langage de requête structurée informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles. Outre le langage de manipulation des données, la partie langage de définition des données permet de créer et de modifier l'organisation des données dans la base de données ; la partie langage de contrôle de transaction permet de commencer et de terminer des transactions, et la partie langage de contrôle des données permet d'autoriser ou d'interdire l'accès à certaines données à certaines personnes. Les instructions SQL s'écrivent de la manière qui ressemble à celle de phrases ordinaires en anglais. Cette ressemblance voulue vise à faciliter l'apprentissage et la lecture. On classe les ordres suivant 4 domaines :

- LDD (Langage de description des données),
- LMD (Langage de manipulation des données),
- LCD (Langage de contrôle des données),
- LCT (Langage de contrôle de Transaction).

Le langage SQL est un langage déclaratif, c'est-à-dire qui permet de décrire le résultat escompté, sans décrire la manière de l'obtenir.

Les SGBD sont équipés d'optimiseurs de requêtes, des mécanismes qui déterminent automatiquement la manière optimale d'effectuer les opérations. Les instructions de manipulation du contenu de la base de données commencent par les mots clés SELECT, UPDATE, INSERT ou DELETE qui correspondent respectivement aux opérations de recherche de contenu, modification, ajout et suppression.

1.2.1.4 Le langage UML [6]

Le langage de modélisation unifie, de l'anglais Unified Modeling Langage (en sigle UML), est un langage de modélisation graphique à base de pictogramme conçu comme une méthode normalisé de visualisation dans les domaines du développement logiciel et en conception orientée objet. Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sorte de systèmes ne se limitant pas au domaine informatique.

Il nous permet une meilleure conception de l'application avec ses notions d'objets et de classes, et nous donne une décomposition claire et simple afin de dégager les entités et les classes nécessaires.

1.3 Analyse du système existant

1.3.1 Présentation du milieu d'étude

L'officine pharmaceutique la Gracia a été ouverte en juillet 2015 et elle est localisée dans la ville de Goma, elle a été fondée par le docteur Hélène ; licenciée en Pharmacie.

1.3.1.1 Organisation de l'officine pharmaceutique Gracia

L'officine dans sa hiérarchie est composée de deux parties, la partie administration qui s'occupe de la vérification du travail et du bon déroulement des travaux au sein de l'officine

La partie personnelle qui est composée de deux pharmaciens dont les tâches principales sont la vente et la consultation des patients.

1.3.2 Cadre du projet et circuit de déroulement des activités

Dans le cadre de notre travail, réalisation d'une application de gestion d'une pharmacie médicale.

La procédure actuelle qui décrit la gestion de notre pharmacie est :

- A l'approvisionnement de la pharmacie le pharmacien complète une fiche pour renseigner les informations par rapports au stock qui vient d'arriver à la pharmacie. Ces informations sont :
    - Le nom du médicament

- La quantité qu'on vient d'acquérir
- Le type de médicament acquis : comprimé, flacon, etc.
- La date d'acquisition et la date d'expiration
- Le prix d'achat des médicaments, etc

- Apres acquisition le pharmacien procède à l'arrangement des nouveaux médicaments dans la pharmacie, il procède comme suit :
    - Il doit faire un tableau de compte total pour comparer le stock existant avant l'acquisition du dernier stock, il compare tous les éléments descriptifs des médicaments se trouvant dans la pharmacie avant l'approvisionnement à ceux qu'on vient d'acquérir.
    - Il fait les comptes et dresse une nouvelle table dans lequel il décrit l'état actuel du magasin.
    - Dans cette base de données on retrouve aussi toutes les informations sur les médicaments. C'est dans cette base ou on définit le prix unitaire de chaque médicament
- Apres avoir dresser la base de données on peut maintenant procéder à la vente des médicaments qui se fait en tenant compte des informations dans la base de données.

1.3.3 Critique de l'existant et proposition de la solution

Après avoir fait la présentation et la description du contexte du domaine d'étude, nous établissons présentement un diagnostic sur le plan informationnel et sur le plan de suivi avec comme défaillance :

- Risque de pertes des fiches contenant les renseignements sur l'approvisionnement ou sur la nouvelle base de données.
- Perte de temps dans la rédaction de la nouvelle base des données.

- Possibilité d'avoir de répétition quant aux dossiers à gérer.
- Risque de confusion dans la rédaction de la base de données ou dans la rédaction d'un rapport.

1.3.4 Solution proposée

Afin de pallier aux défaillances soulevées, nous proposons d'informatiser le procéder de gestion de pharmacie et pour cela nous optons pour le développement d'une application web qui sera exploiter par tous les membres de notre pharmacie dont l'objectif principale est la gestion de stock de notre pharmacie.

L'application doit faciliter la gestion et l'approvisionnement du stock, faciliter l'enregistrement des nouveaux médicaments dans notre base de données, la vente des médicaments et la surveillance de l'état du stock pour permettre une bonne suivie et une bonne gestion de la pharmacie.

1.4 Conclusion partielle

Dans ce chapitre nous avons présenté les grandes lignes théoriques sur lesquelles notre application reposera. Nous avons brièvement présenté les théories sur la base des données, les systèmes de gestion de base de données dont nous avons opté pour MySQL vu ses nombreux avantages. Nous avons aussi fait une présentation de notre milieu de travail et parlé brièvement des causes qui ont fait à ce que nous menions la réalisation de notre application. Nous avons fait l'analyse et le critique de l'existant et aussi nous avons proposé des solutions par rapport aux critiques.

Dans le chapitre suivant, nous allons présenter l'étude conceptuelle de la solution retenue.

Chapitre 2

Modélisation de la solution retenue

2.1 Introduction

Partant du contexte technologique actuel, il semble évident que la conception de toute application informatique n'est autre que d'efforts conjugués pour une bonne spécification. D'ailleurs, il est de plus en plus difficile de garantir la bonne qualité d'une bonne conception sans pour autant avoir une bonne spécification ni même négliger son importance puisqu'il s'agit d'une réponse aux questions posées.

Dans ce chapitre il sera question de présenter la conception vis du problème posé. Ce dernier n'est autre que la mise en place d'une application de gestion d'une pharmacie médicale.

2.2 Spécification du besoin

Comme mentionné dans les chapitres précédents, pour des raisons techniques et de sécurité il est préférable de mettre à jour la gestion de la pharmacie Gracia. Dans cette partie, nous nous intéresserons aux besoins utilisateurs à travers les spécifications fonctionnelles et non fonctionnelles pour aboutir à une application de bonne qualité selon les besoins.

2.2.1 Besoins fonctionnels

Les besoins fonctionnels se rapportent aux fonctionnalités que l'application doit offrir pour satisfaire les utilisateurs.
Le système qu'il nous faudra mettre en place devra implémenter les fonctionnalités suivantes :

- **Gestion des médicaments** : cette opération consiste à suivre l'état du stock à savoir les mouvements réalisés sur le stock (entrée /sortie de médicament, quantité des médicaments dans le stock, liste de médicaments en voie de rupture ou de péremption).

- **Gestion des commandes** : cette opération est établie lorsqu'il y a un besoin de renouveler le stock des médicaments. L'utilisateur peut créer un bon de commande correspondant à ses besoins ou se référer directement à la liste des produits en rupture dans le stock.
- **Gestion des ventes** : cette opération consiste à réaliser une vente sur l'application. L'utilisateur peut même consulter la liste des ventes (journalières, hebdomadaires et mensuelles).

2.2.2 Besoins non fonctionnels

Les besoins non fonctionnels sont indispensables et permettent l'amélioration de la qualité logicielle de notre système. Ils agissent comme des contraintes sur les solutions, mais leur prise en considération fait éviter plusieurs incohérences dans le système. Ce dernier doit répondre aux exigences suivantes :

- **Authentification** : le système doit permettre à l'utilisateur de saisir son login et son mot de passe pour accéder au système. Cette opération assure la sécurité du système et limite le nombre des utilisateurs.
- **Ergonomie** : le système devra offrir aux utilisateurs une interface qui soit la plus riche possible afin de limiter le nombre d'écrans. Par ailleurs, l'interactivité devra être adaptée (usage du clavier, menu, etc..).
- **Fiabilité** : le système doit être fiable (l'utilisateur doit avoir confiance en la qualité de son produit, pour mieux s'occuper du malade tant le domaine d'intervention est sensible).
- **Accessibilité** : l'application doit être mobile ; c'est à dire que le gérant ou le pharmacien peut accéder à cette dernière et avoir le même service en dehors de la pharmacie.

2.3 Les acteurs du système

Dans cette partie nous allons définir le rôle de chaque acteur qui interagit avec le système. Nous allons aussi modéliser leurs rôles sous forme de diagrammes de cas d'utilisation, puis nous définirons les cas d'utilisation et nous les modéliserons sous forme de diagramme de séquence. Nous finirons par le diagramme de classe et le modèle relationnel.

2.3.1 Définition [7]

Un acteur est une entité externe qui interagit directement avec le système (Utilisateur, dispositif matériel, ou autre système...). En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin (modification du système ou simple consultation). Les acteurs peuvent être classés hiérarchiquement.

2.3.2 Identification des acteurs

2.3.2.1 Les acteurs principaux

La pharmacie Gracia emploie actuellement un pharmacien et un gérant. Le pharmacien s'occupe des ordonnances, de la gestion des médicaments, d'achats et des statistiques tandis que le gérant se charge de voir les statistiques et gérer les pharmaciens.
En plus de ces deux, il y a un administrateur qui est le seul acteur ayant le droit d'accès au code source, la mise à jour de l'application et à sa maintenance en cas de panne.

2.3.2.2 Les acteurs secondaires

Dans notre application les acteurs secondaires sont :

- Le client : la personne venue faire une commande pour achat d'un médicament
- Le fournisseur : a pour rôle de fournir les différents produits dont la pharmacie a besoin.

2.4 Conception du système

2.4.1 Les diagrammes UML [8]

La modélisation permet de représenter et de simuler le fonctionnement d'un système informatique, avant de l'avoir développé. UML permet d'effectuer cette représentation sous forme des différents diagrammes.

UML propose aujourd'hui 13 diagrammes différentes :

- Les diagrammes structurels : diagrammes des classes, d'objets, de composants, de structure composite, de déploiement et de paquetages.
- Les diagrammes de comportement : diagrammes des cas d'utilisations, d'activités et d'états-transition.
- Les diagrammes d'interactions : diagrammes de séquence, de vue générale d'interaction et celui de temps

Ces diagrammes permettent de définir une application selon plusieurs points de vue :

- **Fonctionnel** (cas d'utilisation).
- **Statique** (classes, objets, structure composite).
- **Implémentation (**composant**,** déploiement, paquetage).
- **Dynamique** (séquence, états, activité, interaction, communication, temps).

Les diagrammes seuls ne permettent pas de définir toutes les contraintes de spécification requises L'utilisation du langage textuel de contraintes **OCL** en complément s'applique sur les éléments de la plupart des diagrammes.

2.4.2 Diagrammes UML utilisés

Pour modéliser notre système nous avons utilisé les trois types de diagrammes suivants :

Diagramme	Objectif	Type

Diagramme de classes	1. Point central de la modélisation du système pour décrire ce que le système doit faire (analyse) et avec quoi il va le faire (conception). 2. Représentation de la structure statique du système d'information. 3. Modélisation des classes et de leurs relations.	Statique
Diagramme de cas d'utilisation	1. Décrire la manière dont une organisation ou un système externe doivent interagir avec le système 2. Décrire ce que doit faire le système. 3. Mettre en évidence les services rendus par le système.	Fonctionnel
Diagramme de séquence	1. Validation des cas d'utilisation pour comprendre la logique de l'application. 2. Complète le diagramme des cas d'utilisation en mettant en évidence les objets et leurs interactions d'un point de vue temporel.	Dynamique

Tableau 1 description des diagrammes UML

2.4.3 Liens entre les diagrammes

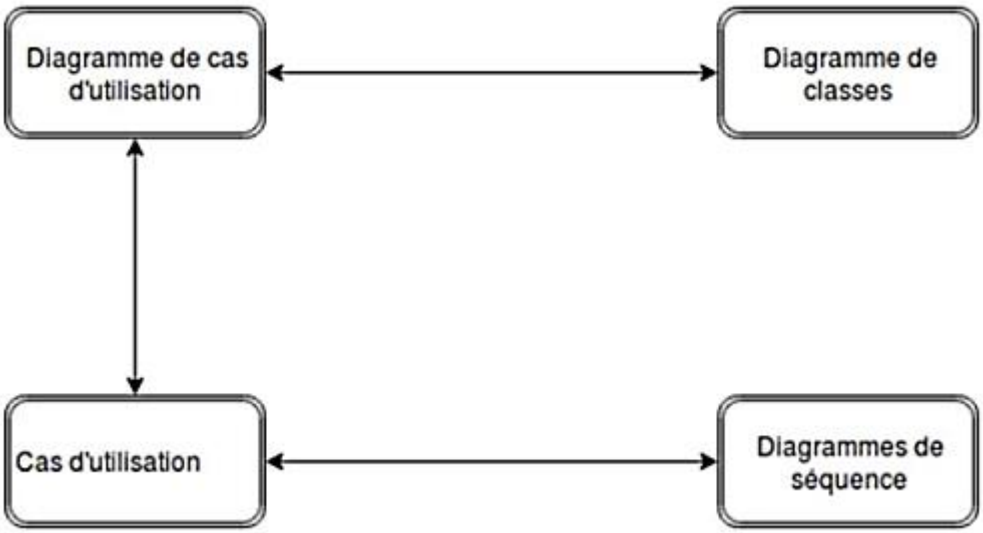


Figure 1 liens entre les 4 diagrammes UML.

2.4.4 Diagramme de cas d'utilisation

2.4.4.1 Définition [9]

Un diagramme de cas d'utilisation est un graphe d'acteurs, un ensemble de cas d'utilisation englobés par la limite du système, des relations (ou associations) de communication (participation) entre les acteurs et les cas d'utilisation, et des généralisations de ces cas d'utilisation.

Les relations de généralisation/spécialisation entre acteurs permettent de définir des profils d'acteurs. Les relations entre cas d'utilisation sont soit utilisé (uses) pour éviter de dupliquer des processus communs à plusieurs processus, soit étend (extends) pour décrire séparément des parties alternatives ou optionnelles ou exceptionnelles de processus. Les relations entre acteurs et cas d'utilisation indiquent les interactions.

2.4.4.2 Diagramme de cas d'utilisation du gérant

Le diagramme ci-dessous représente les cas d'utilisations identifiés pour le gérant.

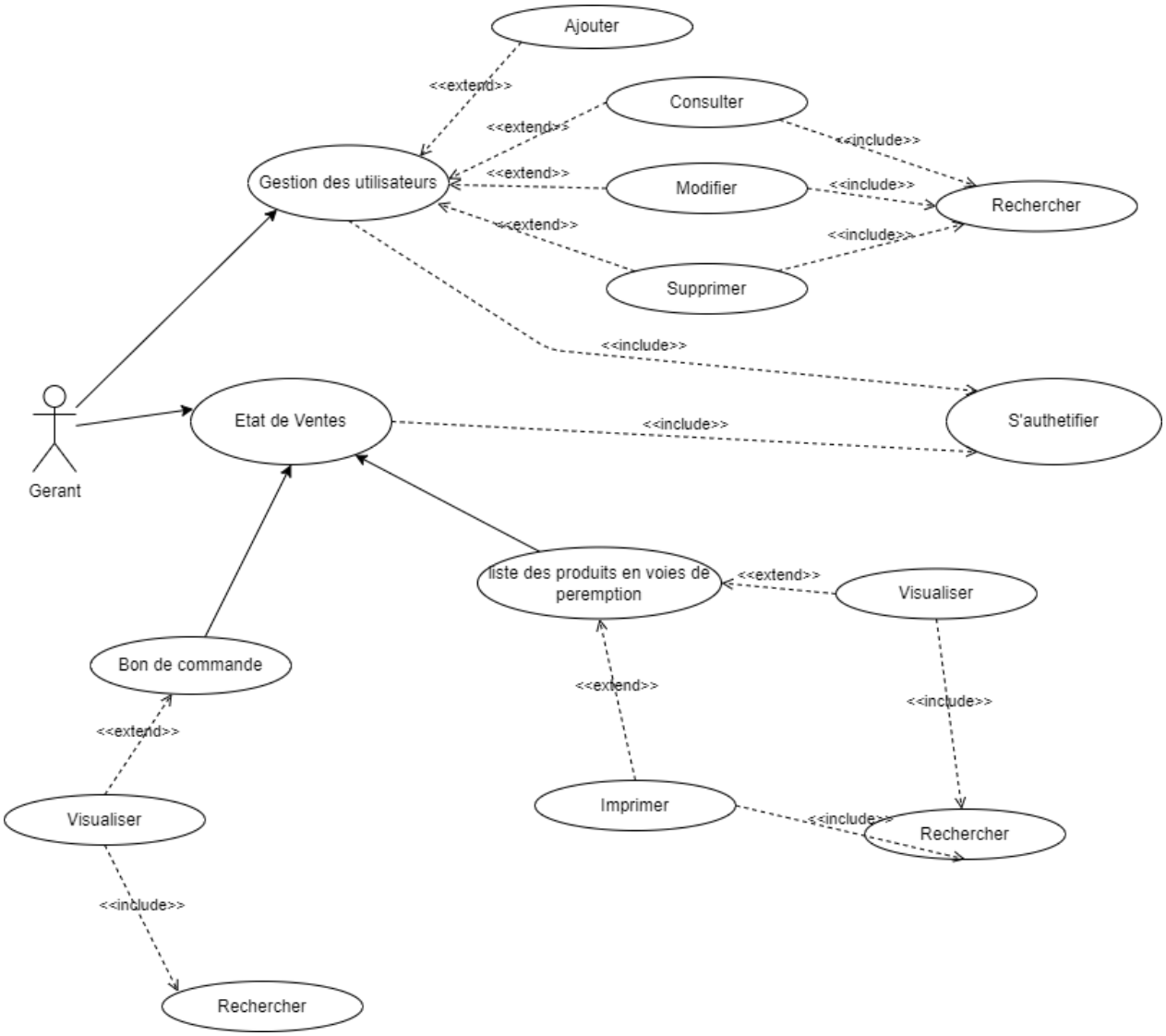


Figure 2 Diagramme de cas d'utilisation du gérant

2.4.4.3 Diagramme de cas d'utilisation du pharmacien

Le diagramme ci-dessous représente les cas d'utilisations identifiés pour le pharmacien.

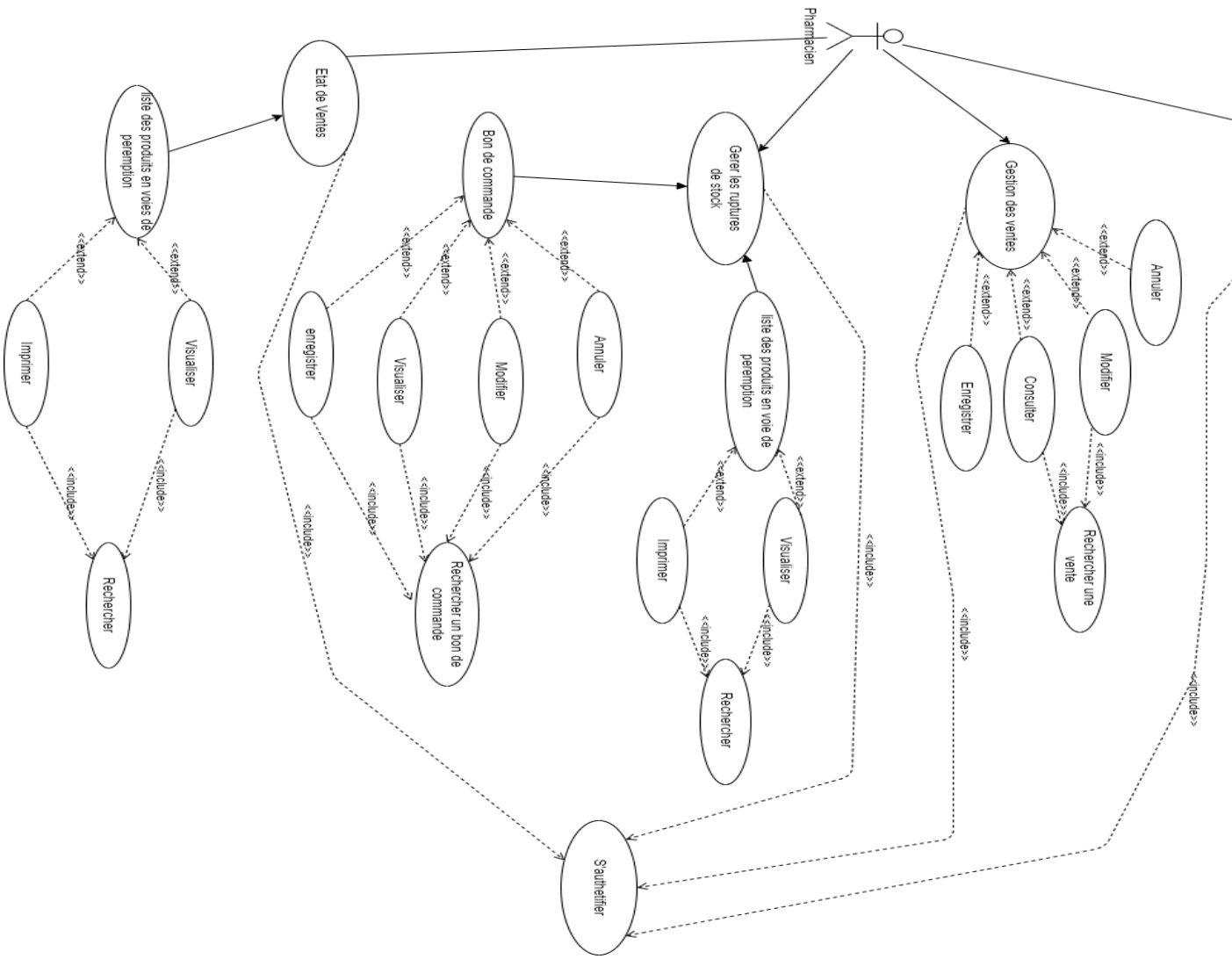


Figure 3 Diagramme de cas d'utilisation du pharmacien

2.4.4.4 Description textuelle des cas d'utilisations

Cas d'utilisation N°1 : Authentification

Acteurs Principaux	Gérant/Pharmacien
Objectif	S'authentifier avant d'accéder à la page d'accueil de l'application.
Précondition	Avoir une connexion internet et un navigateur.
Scénario	1. L'utilisateur se connecte à internet, lance l'application web via un navigateur web. 2. Le système demande à l'utilisateur de s'authentifier. 3. L'utilisateur saisit son nom et son mot de passe. 4. Le système vérifie la conformité des informations saisies en envoyant une requête aux serveurs. 5. La requête est vérifiée par le serveur et ce dernier envois une réponse favorable. 6. L'utilisateur accède au menu principal.
Alternative	En cas de réponse défavorable du serveur, le système affiche un message d'erreur en cas d'erreur de saisie ou bien d'un champ incomplet (retour à 2).

Tableau 2 Diagramme global de cas d'utilisations.

Cas d'utilisation N°2 : Gérer le stock

Acteur Principal	Pharmacien
Objectif	Enregistrer un produit dans le stock existant.
Précondition	Authentification.
Scénario	1. L'utilisateur saisie le nom du produit et tous ces caractéristiques (numéros de lot, date d'expiration, quantité…), puis clique sur enregistrer. 2. Le système envois la requête au serveur.

	3. Après le traitement des données par le serveur, il envoie un message au système. 4. Le système affiche à l'écran la réponse du serveur
Alternative	Le système affiche un message d'erreur en cas d'une erreur de saisie ou bien d'un champ incomplet (retour a -1-).

Tableau 3 Description textuelle du cas d'utilisation gérer le stock.

Cas d'utilisation N°3 : Gérer les ventes

Acteur Principal	Pharmacien
Objectif	Enregistrer une vente de produit/Imprimer la facture si besoin.
Précondition	Authentification.
Scénario	1. L'utilisateur saisie le nom du produit dans la barre de recherche de produit puis sélectionne le bon produit ajoute la quantité et clique sur enregistrer. 2. Une requête est envoyée au serveur pour traitement. 3. Le serveur envoie un message de succès à l'interface de l'utilisateur.
Alternative	Le système affiche un message d'erreur en cas d'une erreur de saisie ou bien d'un champ incomplet (retour a -1-).

Tableau 4 Description textuelle du cas d'utilisation gérer la vente.

Cas d'utilisation N°4 : Gérer les utilisateurs

Acteur Principal	Gérant.
Objectif	Enregistrer un utilisateur.
Précondition	Authentification.

Scénario	1. Le gérant accède à l'interface gérer les utilisateurs puis clique sur ajouter un utilisateur et enfin saisit le formulaire d'ajout et clique sur enregistrer. 2. Une requête est envoyée au serveur pour traitement. 3. Le serveur envoie un message de succès à l'interface de l'utilisateur.
Alternative	Le système affiche un message d'erreur en cas d'une erreur de saisie ou bien d'un champ incomplet (retour a -1-).

Tableau 5 Description textuelle du cas d'utilisation gérer la vente.

Cas d'utilisation N°5 : Gérer les ruptures

Acteur Principal	Pharmacien
Objectif	Enregistrer un produit dans le stock existant.
Précondition	Authentification.
Scénario	1. Le pharmacien accède à l'interface gérer les ruptures. 2. Le pharmacien demande a accédé à la liste des produits en rupture. 3. Une requête est envoyée au serveur pour traitement. 4. Le serveur envois la liste des produits en voit de rupture et l'interface l'affiche à l'écran. 5. Le pharmacien peut formuler le bon de commande. 6. A la fin de sa saisie il peut enregistrer 7. Une requête est envoyée au serveur pour traitement. 8. Le serveur repend par un message de succès.
Alternative	Le système affiche un message d'erreur en cas d'une erreur de saisie ou bien d'un champ incomplet (retour a -5-).

Tableau 6 Description textuelle du cas d'utilisation gérer les ruptures.

Cas d'utilisation N°6 : consulter l'état des ventes

Acteur Principal	Gérant/Pharmacien
Objectif	Accéder à la liste des produits vendue.
Précondition	Authentification.
Scénario	1. L' utilisateur demande le formulaire de consultation. 2. Le système affiche le formulaire. 3. Une requête est envoyée au serveur pour traitement. 4. Le serveur répond et le système affiche la liste demandée.
Alternative	Coupure internet/ou problème technique (réactualiser la page/ou vérifié la connexion internet.

Tableau 7 Description textuelle du cas d'utilisation consulter l'état des ventes.

2.4.5 Diagrammes de séquence

2.4.5.1 Cas d'utilisation Authentification

Lorsque l'utilisateur (Gérant, Pharmacien) veux accéder à notre application web, il sera obligé de s'authentifier avant d'y accéder en saisissant son identifiant et mot de passe, après la saisie le système envois une requête au serveur pour traiter les informations envoyées, si les informations sont correctes l'utilisateur accédera à sa session sinon un message d'erreur sera affiché et reconduira l'utilisateur à la page authentification.

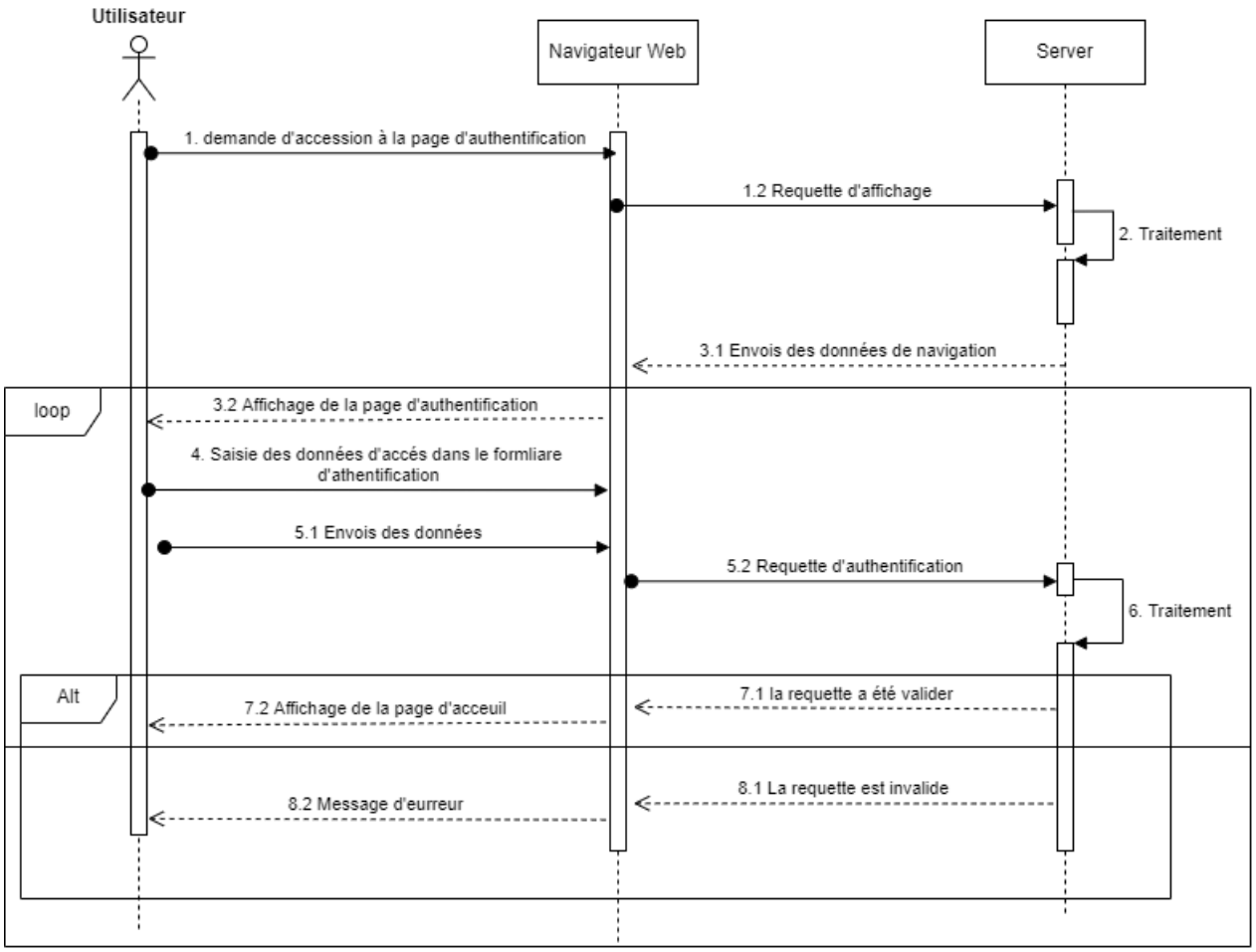


Figure 4 Diagramme de séquence du cas d'utilisation (Authentification).

2.4.5.2 Cas d'utilisation Gérer la vente

Lorsque le Pharmacien a accédé à l'application il lui sera possible d'effectuer une vente en cliquant sur gestion de vente, après le clique il pourra enregistrer une vente de produit en introduisant les champs requis et en cliquant sur enregistrer le système envois la requête au serveur qui lui enregistre les données et envoie un message de succès ou sinon un message d'erreur sera affiché si il manque un champ.

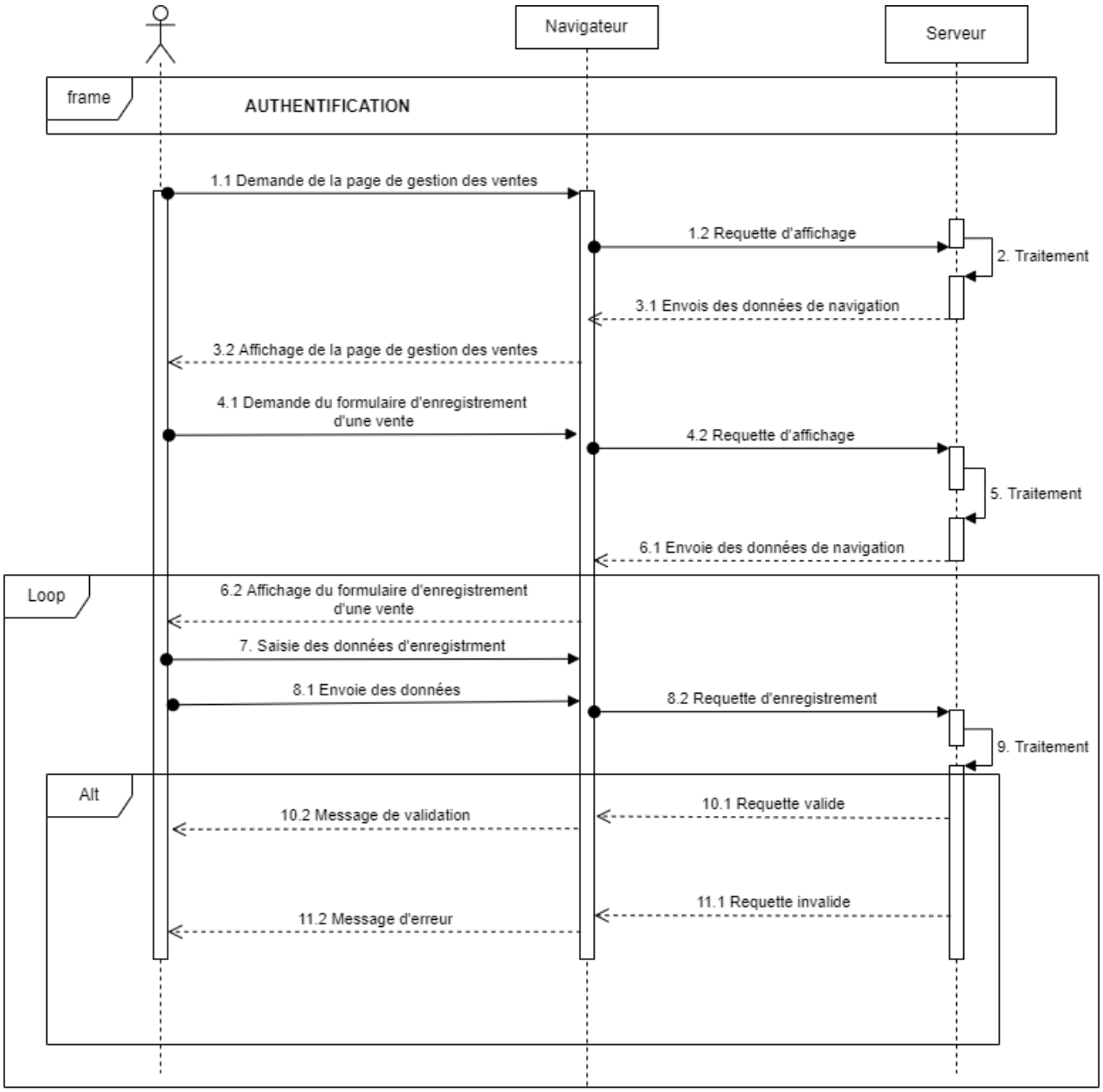


Figure 5 Diagramme de séquence du cas d'utilisation (Gérer la vente).

2.4.5.3 Cas d'utilisation Gérer le stock

Lorsque le Pharmacien veut gérer le stock c'est à dire enregistrer un arrivage de médicament, il doit tout d'abord cliquer sur gestion de stock puis ajouter un produit au stock, pour ensuite remplir les champ requis et enfin envoyer les données au serveur pour les enregistrer. Après traitement du serveur un message de succès ou d'erreur sera envoyé.

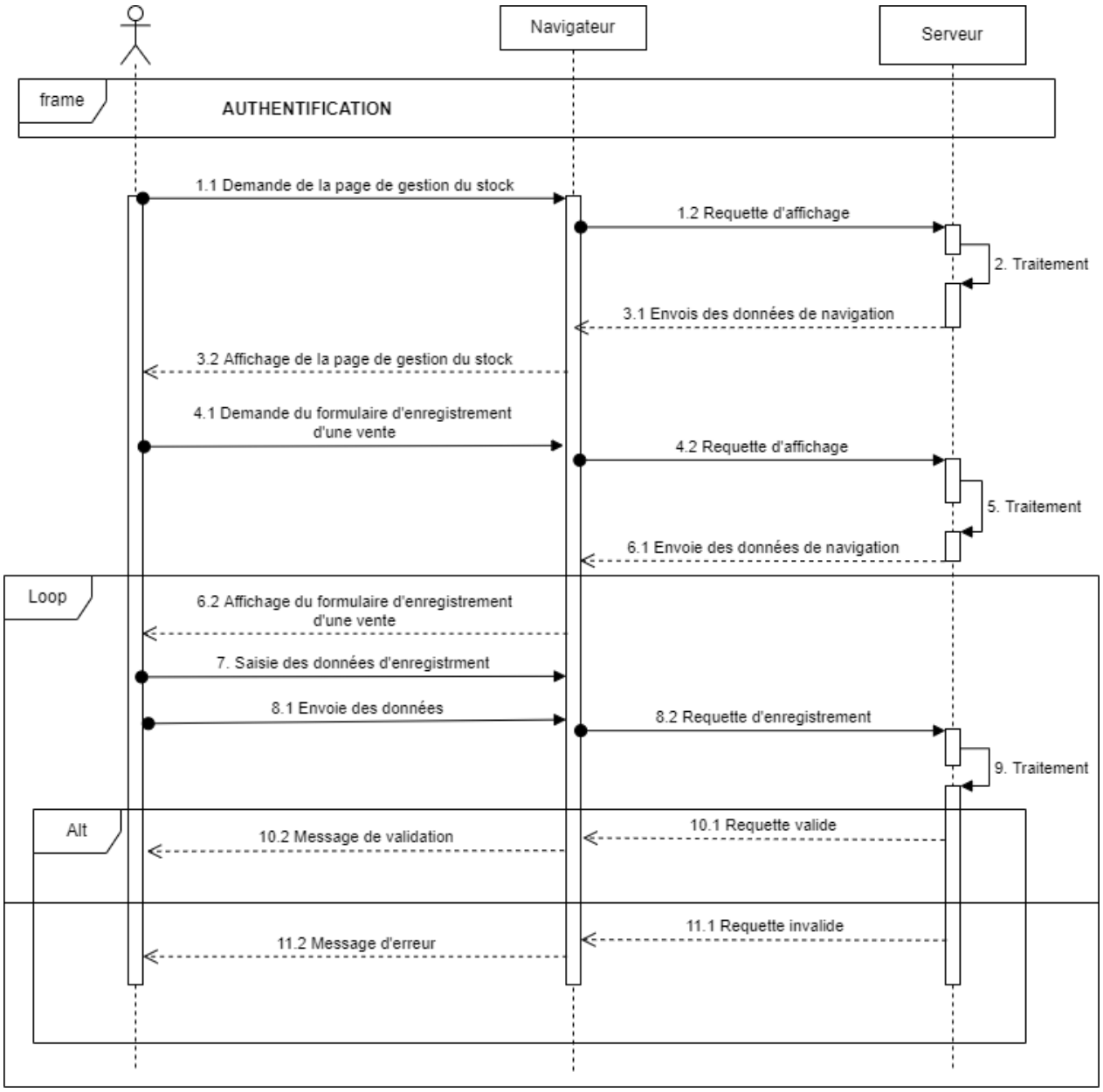


Figure 6 Diagramme de séquence du cas d'utilisation (Gérer la vente).

2.4.5.4 Cas d'utilisation Gérer les utilisateurs

Lorsque le gérant veut gérer les utilisateurs il doit tout d'abord accéder à la gestion des utilisateurs puis rechercher un utilisateur donné pour le modifier. Après ça recherche le gérant n'a qu'à cliquer sur modifier et remplir les champs requis et envoyer au serveur pour traitement et enregistrement. Un message d'erreur est envoyé en cas de champ incomplet sinon un message

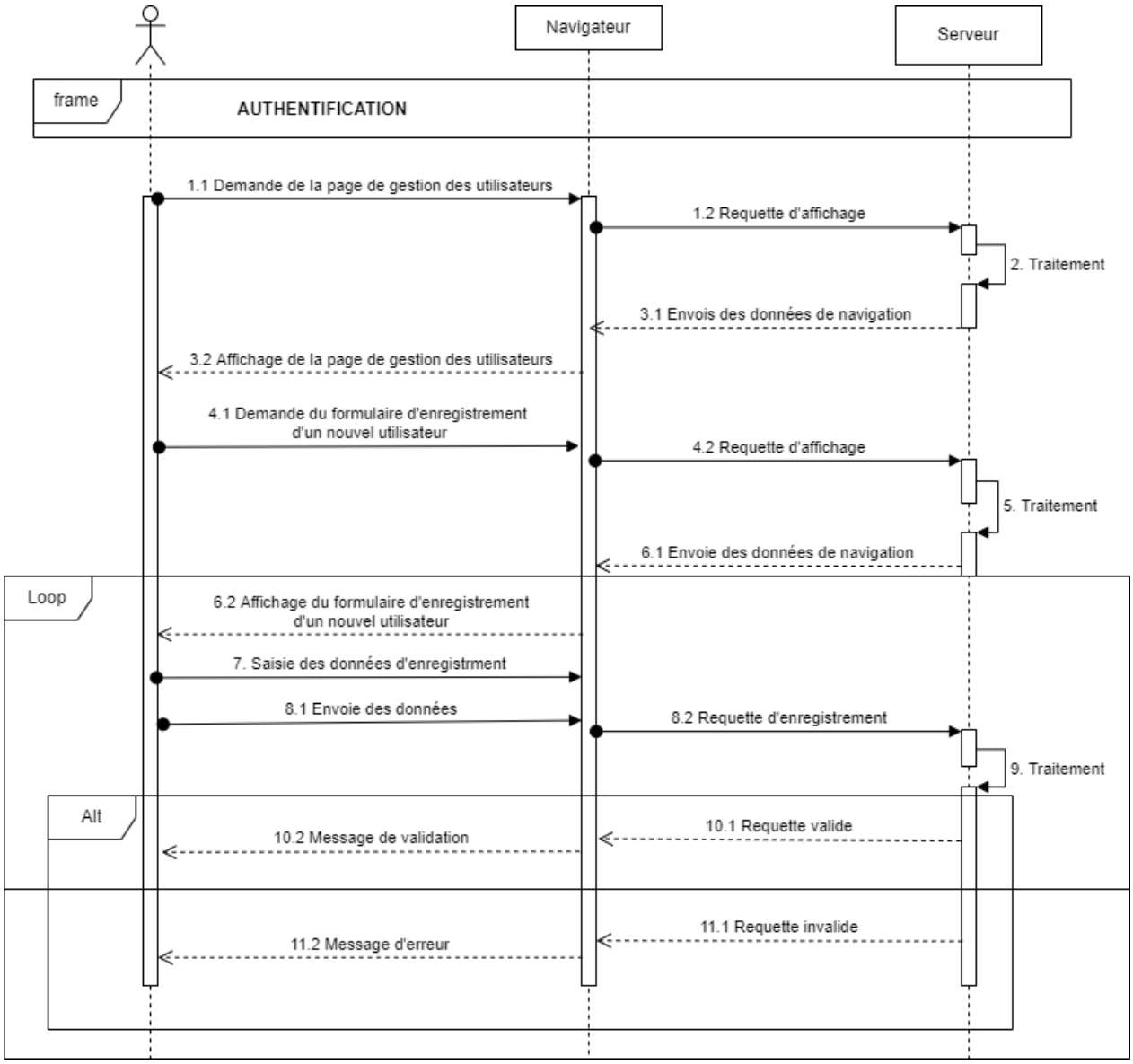


Figure 7 Diagramme de séquence du cas d'utilisation (Gestion des utilisateurs).

2.4.5.5 Cas d'utilisation Gérer les ruptures

Lorsque le pharmacien veut passer une commande au fournisseur il pourra formuler un bon de commande via notre application web. En effet, il doit accéder au formulaire gestion des ruptures et cliquer sur formuler le bon de commande puis remplir les champs requis et envoyé les données saisies au serveur pour traitement. Un message de validation serra afficher, l'employé pourra consulter le bon de commande et commander par téléphone sinon un message d'erreur serra envoyé par le serveur et le gérant devra corriger son erreur

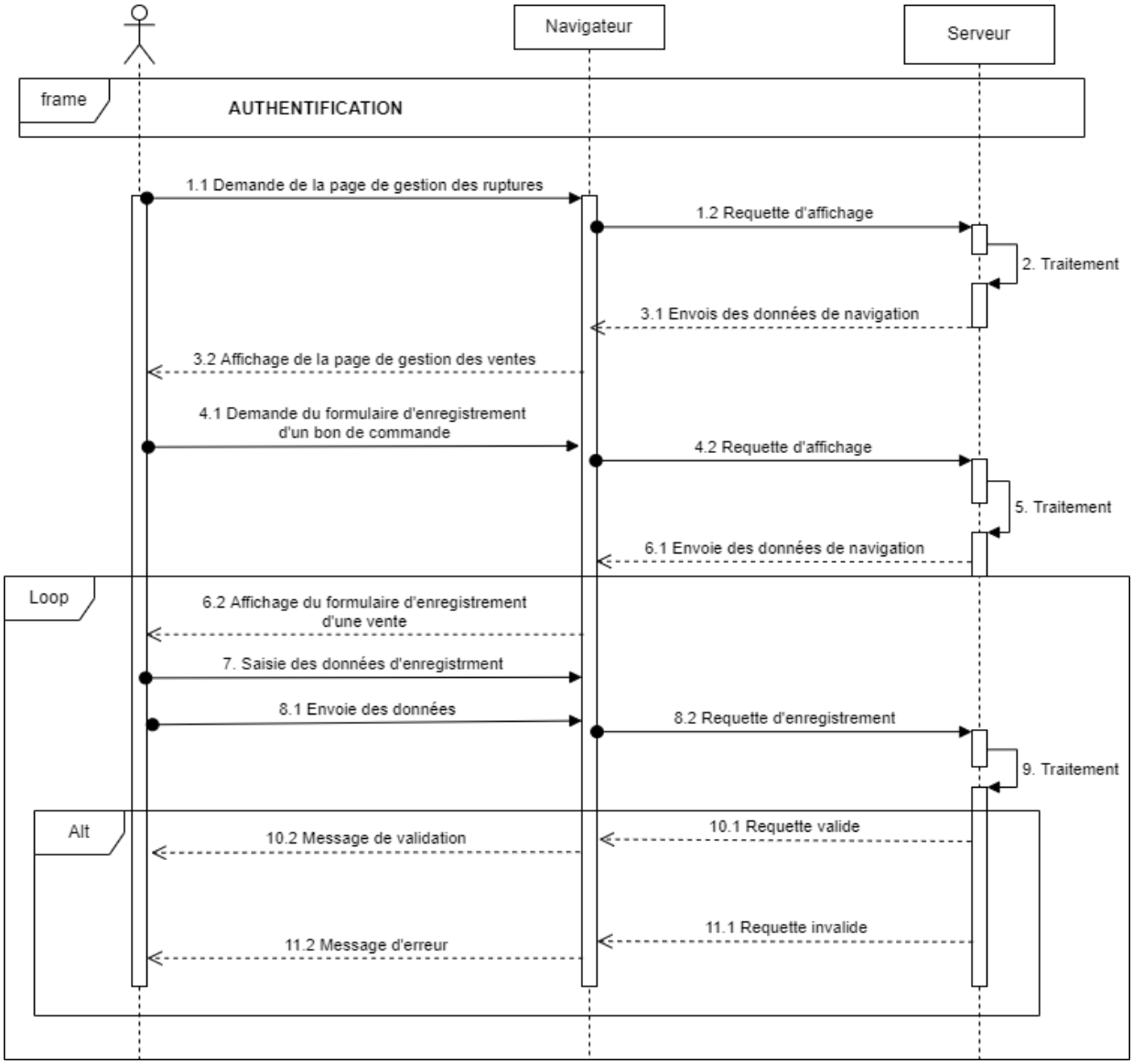


Figure 8 Diagramme de séquence du cas d'utilisation (Gestion des utilisateurs).

2.4.5.6 Cas d'utilisation consulter l'état des ventes

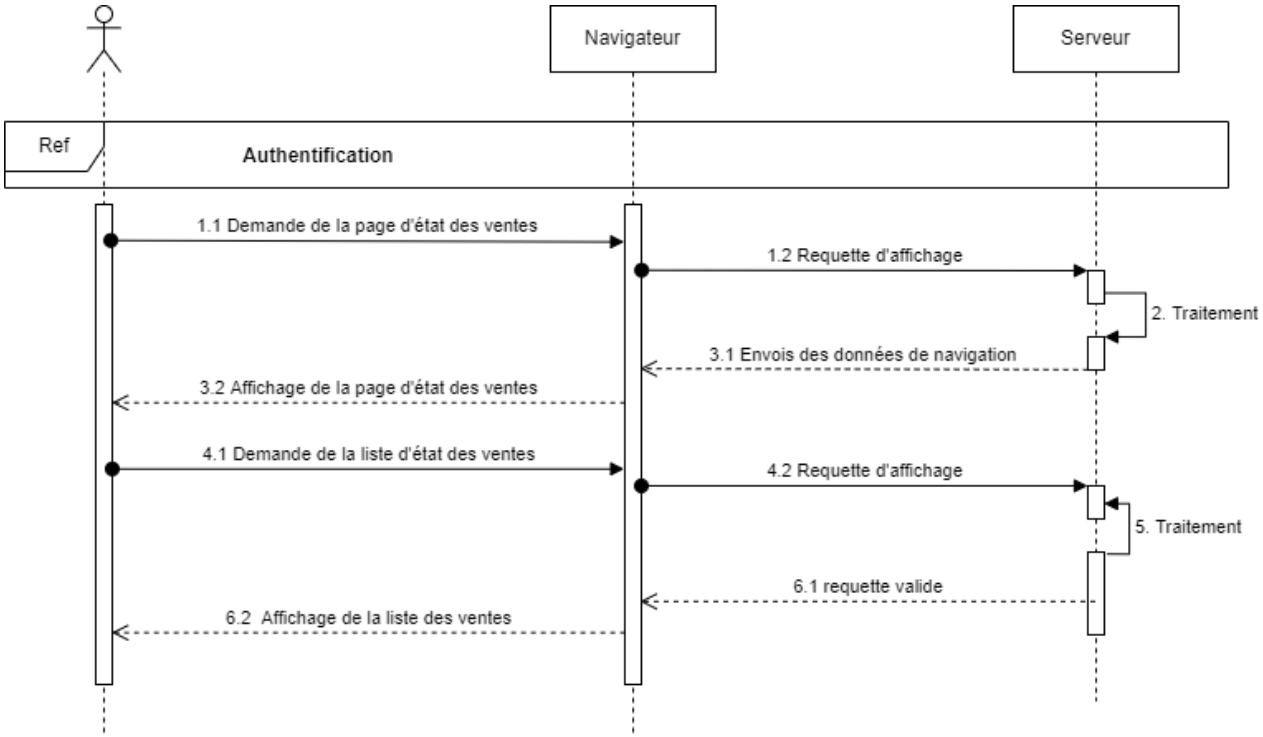


Figure 9 Diagramme de séquence du cas d'utilisation (État des ventes).

2.4.6 Diagramme de classe

2.4.6.1 Diagramme de classes

Le diagramme de classes est sans doute le diagramme le plus important à représenter pour les méthodes d'analyse orientées objet. En effet, il permet de spécifier QUI intervient à l'intérieur du système.

Un diagramme de classes fait abstraction des aspects dynamiques et temporels du système, il permet de représenter une vue statique du système d'information. Il s'agit plutôt des relations entre les classes, des services rendus et utilisés par chacune d'elles et de l'articulation de l'ensemble.

Notre diagramme de classe sera composé des éléments suivants :

- Classe : modélisation d'un objet d'intérêt pour l'utilisateur
- Association : modélisation d'une relation entre deux ou plusieurs classes.

- Cardinalités : modélisation des participations minimales et maximales d'une entité a une relation.
- Propriétés : modélisation des informations descriptives rattachées à une entité une relation.
- Identifiant : modélisation des propriétés contribuant à la détermination unique d'une occurrence d'une entité.
- Méthodes : éléments qui permettent de faire des actions sur une classe.

Présentons tout d' abord les données qui figureront dans le modèle conceptuel et cela dans un dictionnaire de données.

Classe	Attributs	Signification	Type
Utilisateurs	Id	Identifiant de l'utilisateur	Varchar
	Nom	Nom de L'utilisateur	Varchar
	Prenom	Prénom de l'utilisateur	Varchar
	Email	Email de l'utilisateur	Varchar
	MotDePasse	Mot de passe de l'utilisateur	Varchar
	Type	Type : Gérant/Pharmacien	Enum
Produit	N_Lot	Numéro Lot (Identifiant)	Integer
	Nom_C	Nom Commercial	Varchar
	Qte	Quantité	Integer
	PrixUn	Prix Unitaire	Float
	ExpDate	Date d'expiration	Date
	Dosage	Dosage	Varchar
	Conditionnement	Conditionnement	Varchar
Vente	Id_V	Identifiant de la vente	Varchar
	Qte	Quantité vendue	Integer
	Nom_C	Nom Commercial	Varchar
	prixUn	Prix Unitaire	Float

	total	Total d'une vente	Float
Bon de commande	Id_Bon Date_Bon	Identifiant du bon Date de création du bon	Varchar Date

Figure 10 Diagramme de séquence du cas d'utilisation (État des ventes).

Voici maintenant notre diagramme de classe relatif à notre système :

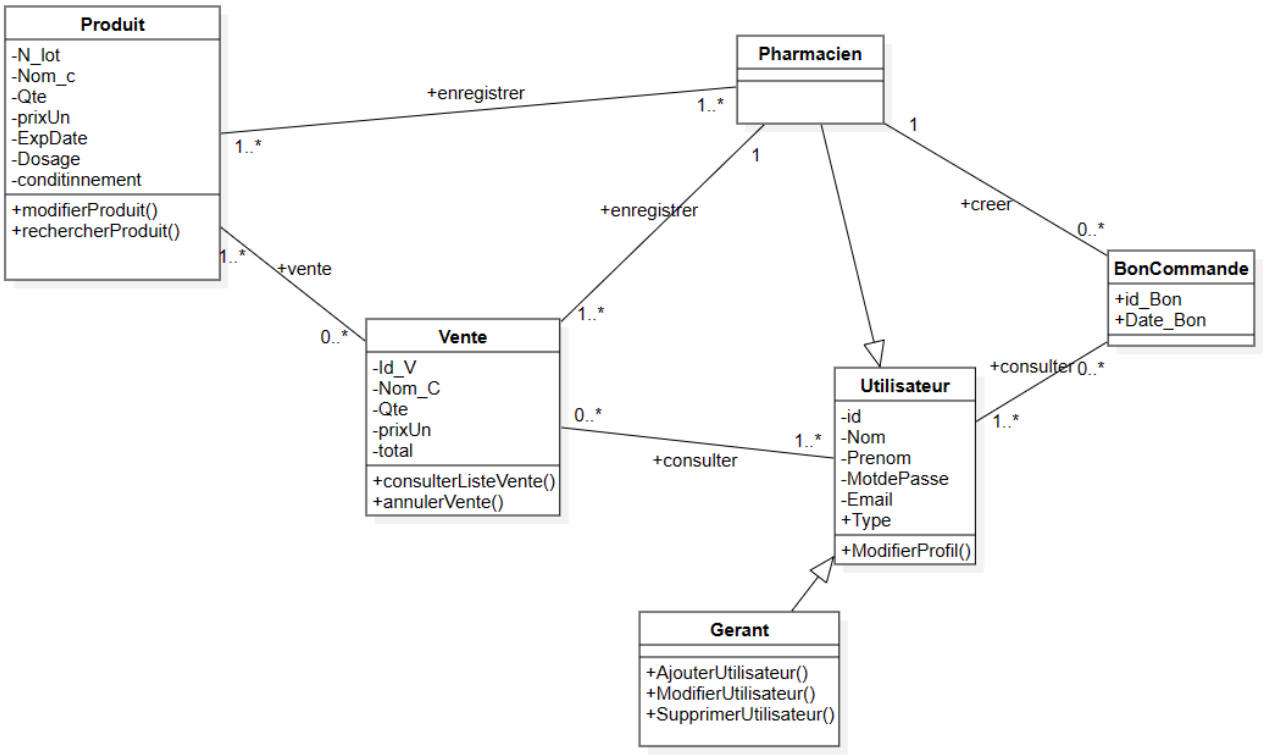


Figure 11 Diagramme de classes.

2.5 Conclusion partielle

Dans cette partie de notre travail, nous nous sommes focalisés sur la présentation de la solution conceptuelle adoptée. Nous avons présenté les diagrammes (diagrammes de cas d'utilisation, diagrammes de séquence et des classes) qui nous ont permis représenter les besoins.

Les résultats de cette partie seront exploités dans le prochain chapitre qui parle de la réalisation (implémentation) du système en nous basant sur des technologies et outils qui seront présentés dans le chapitre suivant.

Chapitre 3

Implémentation du nouveau système

3.1 Introduction

Apres l'analyse et la conception de l'application, viens ensuite l'implémentation de la solution retenue. Dans cette partie nous allons présenter les résultats du travail effectué durant ce projet.

Nous présenterons aussi l'environnement matériel et les outils de développement utilisés. Ce chapitre sera clôturé par quelques captures d'écran démontrant les fonctionnalités et les interfaces de notre application.

3.2 Environnement de travail

3.2.1 Environnement matériel

L'équipement utilisé pour la réalisation de notre travail se compose d'un ordinateur portable avec un système d'exploitation Windows 10 version 2019 et dont les propriétés ont les suivantes :

- Intel(R) Core (TM) i3-5010U CPU @ 2.10GHz 2.10 GHz
- Mémoire RAM: 8GB
- Espace de disque: 500GB

3.2.2 Environnement logiciel

1. Visual Studio Code : est un éditeur de code extensible développé par Microsoft pour Linux, Windows et MacOs.

2. Language utilisé : Laravel [10]

   Laravel est un Framework d'application web avec une syntaxe expressive et élégante.

Laravel s’efforce de fournir une expérience de développement incroyable tout en fournissant des fonctionnalités puissantes telles qu’une injection de dépendance approfondie, une couche d'abstraction de base de données expressive, des files d’attentes et des tâches planifiées, des tests unitaires et d’intégration, etc.

3.2.3 Présentation des résultats

Dans cette partie nous allons présenter les résultats après la réalisation de notre application. Nous ne pouvons pas présenter toutes les interfaces dans le travail car elles sont très nombreuses, nous nous limiterons à la présentation de quelques-unes qui sont pertinentes.

3.2.3.1 Interface d’accueil

L’interface ci-dessous représente l’accueil qui est le point d’entrer de notre application et qui est accessible par tous les utilisateurs.

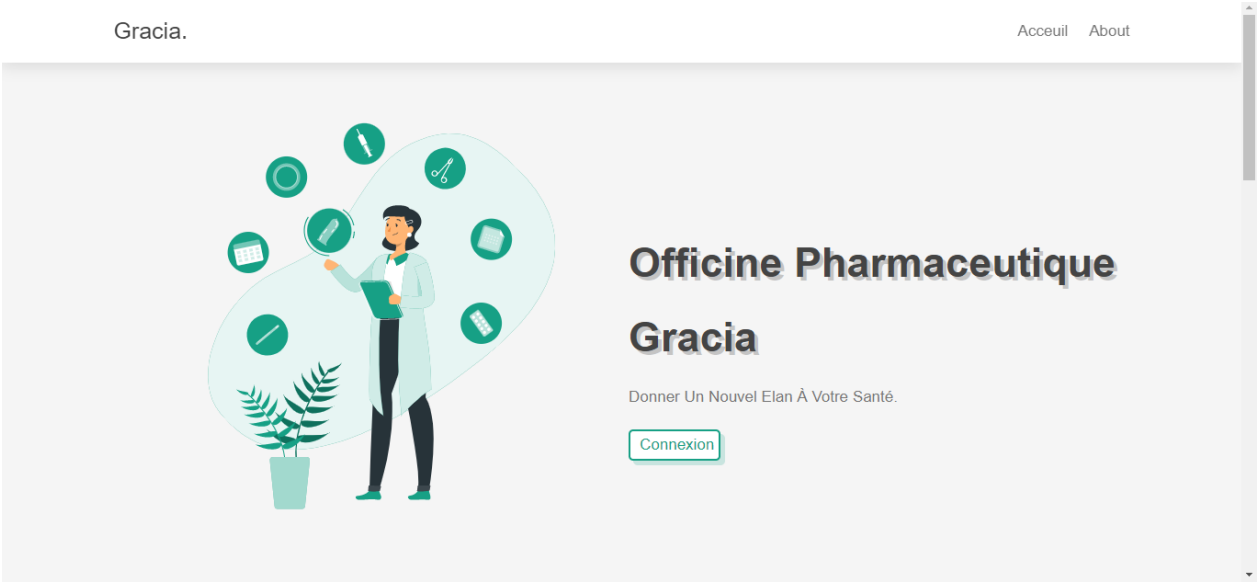


Figure 12 interface d’accueil.

3.2.3.2 Interface d’authentification

Tout utilisateur devra d’abord s’authentifier, il devra compléter le formulaire d’authentification comme présenté à la figure ci-dessous :

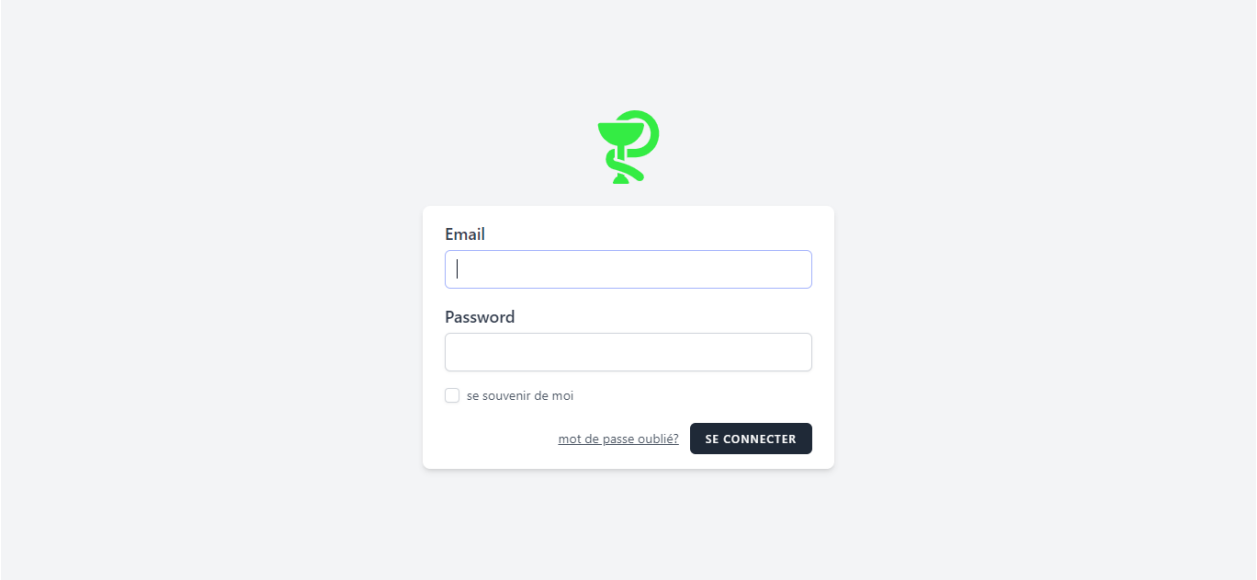


Figure 13 interface d'authentification.

3.2.3.3 Table de bord

Apres authentification, l'utilisateur (pharmacien ou gérant) accède à l'interface qui est représenté à la Figure 14. Grace à cette interface l'utilisateur peut voir ses identifiants, le nombre des ventes totales qu'il a déjà fait depuis son arrivé dans la pharmacie et les pourcentages que ça représente, mais aussi il peut voir le nombre des ventes qu'il a fait pendant la semaine en cours.

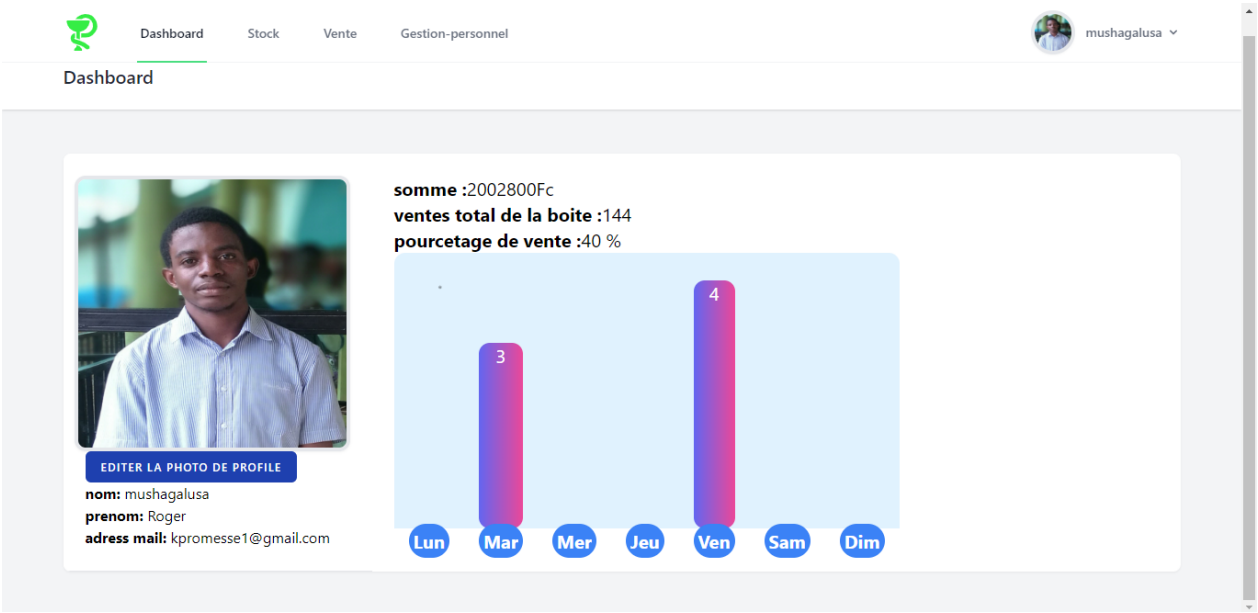
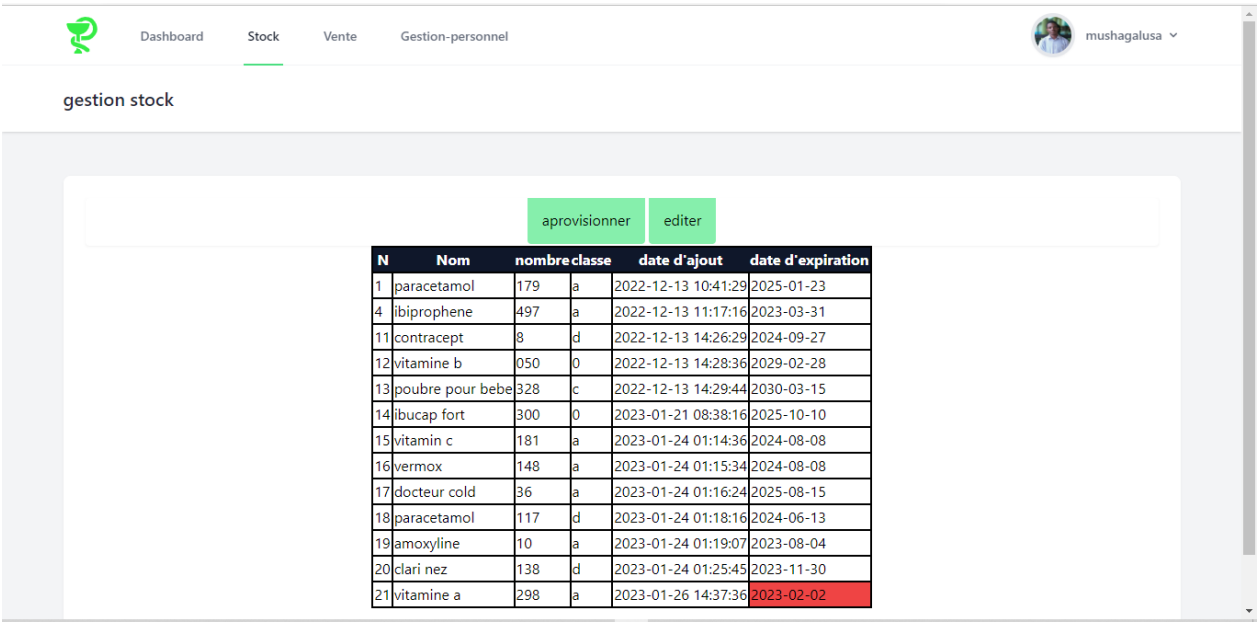


Figure 14 interface pour le dashboard

3.2.3.4 Gestion de stock

En cliquant sur l'option « stock » dans la barre de navigation, l'utilisateur accède à l'interface de gestion du stock, comme le montre la Figure 15 qui représente l'état du stock en générale :



N	Nom	nombre	classe	date d'ajout	date d'expiration
1	paracetamol	179	a	2022-12-13 10:41:29	2025-01-23
4	ibiprophene	497	a	2022-12-13 11:17:16	2023-03-31
11	contracept	8	d	2022-12-13 14:26:29	2024-09-27
12	vitamine b	050	0	2022-12-13 14:28:36	2029-02-28
13	poubre pour bebe	328	c	2022-12-13 14:29:44	2030-03-15
14	ibucap fort	300	0	2023-01-21 08:38:16	2025-10-10
15	vitamin c	181	a	2023-01-24 01:14:36	2024-08-08
16	vermox	148	a	2023-01-24 01:15:34	2024-08-08
17	docteur cold	36	a	2023-01-24 01:16:24	2025-08-15
18	paracetamol	117	d	2023-01-24 01:18:16	2024-06-13
19	amoxyline	10	a	2023-01-24 01:19:07	2023-08-04
20	clari nez	138	d	2023-01-24 01:25:45	2023-11-30
21	vitamine a	298	a	2023-01-26 14:37:36	2023-02-02

Figure 15 interface pour la gestion du stock

3.2.3.5 Approvisionner

En cliquant sur le bouton « approvisionner » de la Figure 15 on accède à l'interface d'ajout d'un médicament comme le montre la Figure 16.

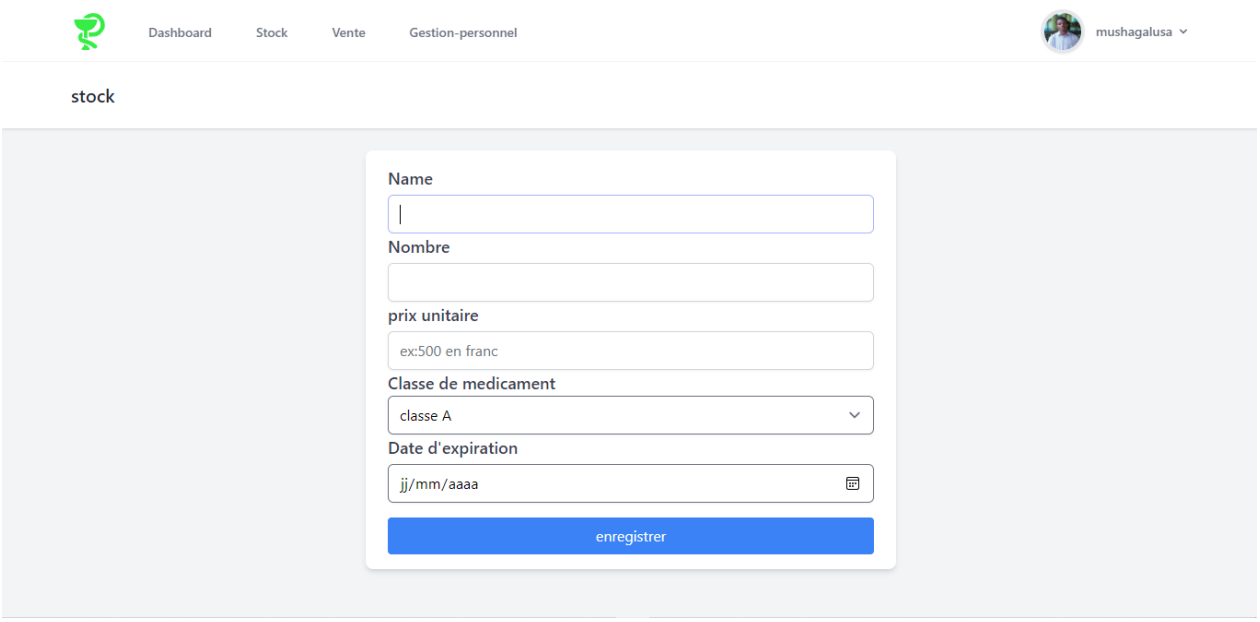


Figure 16 interface pour ajouter un médicament

3.2.3.6 Modifier le stock

En cliquant sur le bouton éditer de la Figure 15 on accède à l'interface de modification des médicaments, cette interface nous permet de modifier ou de supprimer un médicament déjà approvisionné dans le stock.

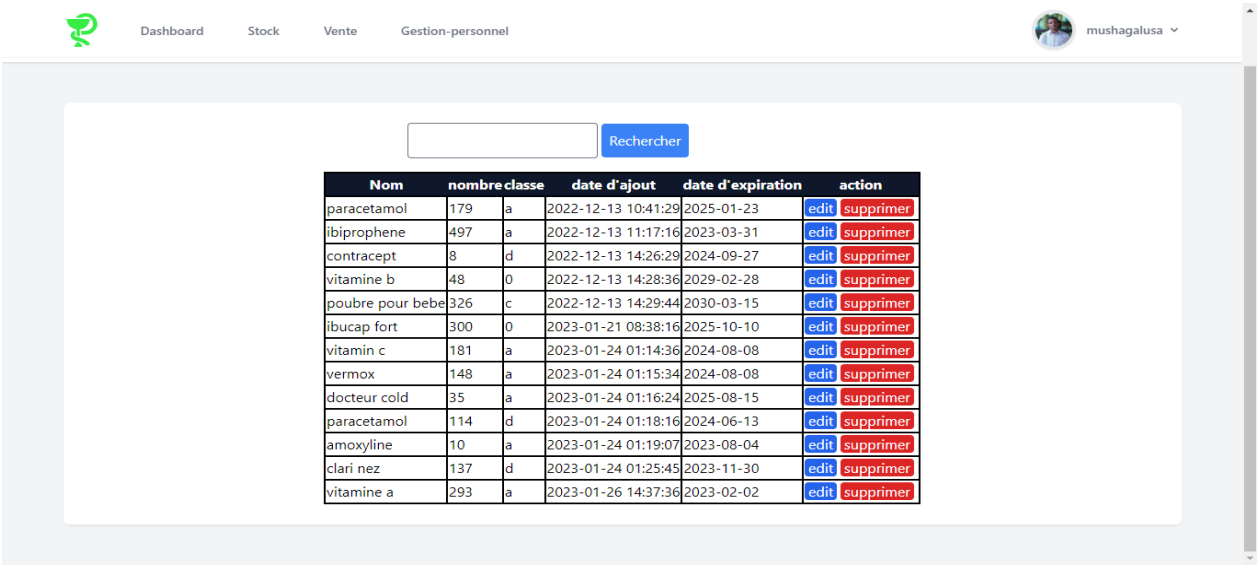


Figure 17 interface pour modifier les éléments du stock

3.2.3.7 Gestion des ventes

En cliquant sur l'option « vente » dans la barre de navigation, l'utilisateur accède à l'interface de gestion de vente comme le montre la Figure 18. Cette interface nous permet de faire le processus de vente dans notre pharmacie.

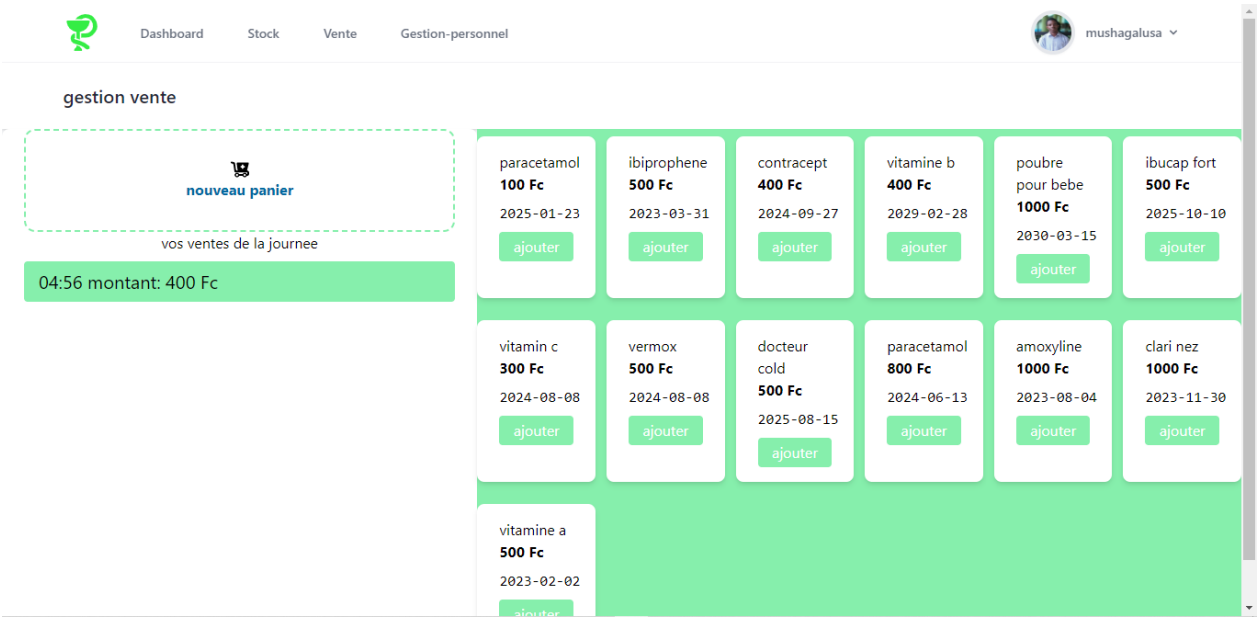


Figure 18 interface de gestion des ventes

3.2.3.8 Ajouter un panier

Pour procéder à une vente quelconque on doit d'abord ouvrir un panier, c'est dans ce panier où seront mises toutes les actions d'une vente quelconque. Pour créer un panier on clique sur le bouton « nouveau panier » de la Figure 18, après on a une interface comme montrer à la Figure 20.

NB : aucune vente ne peut être effectuée sans panier en cours.

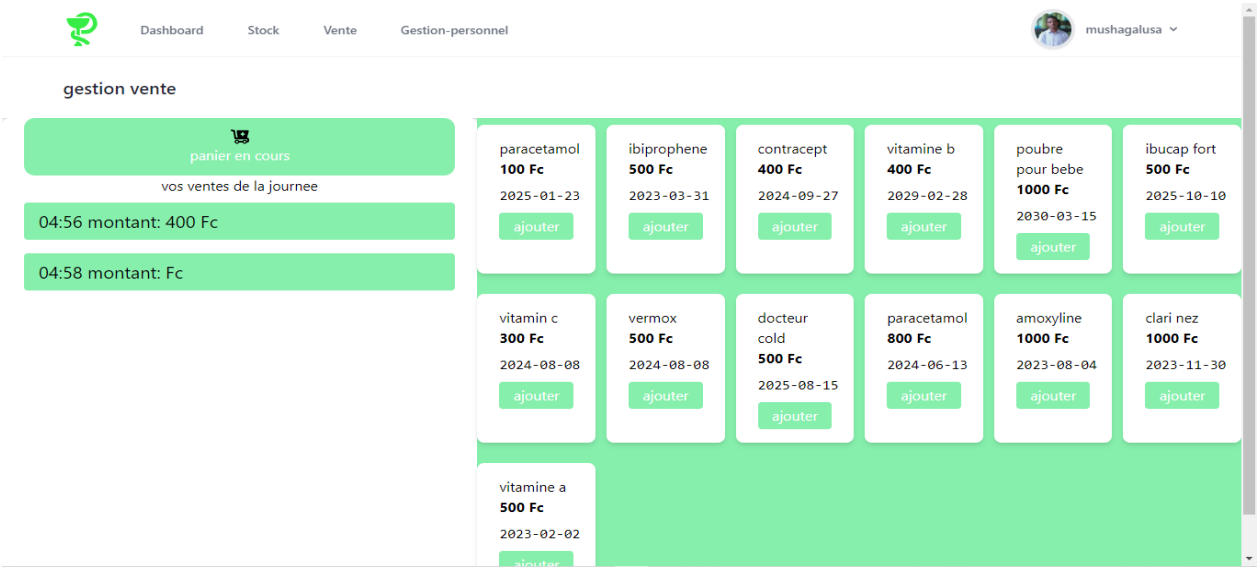


Figure 19 interface de gestion des ventes avec un panier en cours

3.2.3.9 Ajouter une vente

Apres la création d'un panier, on peut procède à une vente, pour y procéder on regarde le nom du médicament dont le client a besoin sur la Figure 19 et après on clique sur bouton « ajouter » en dessous du nom du médicament et on accède à l'interface d'ajout d'un médicament.

Cette interface nous permet d'entrer le nombre de médicament dont le client a besoin et après l'avoir ajouté au panier en cliquant sur le bouton « ajouter » ou annuler en cliquant sur le bouton « annuler » on revient sur l'interface de la Figure 19.

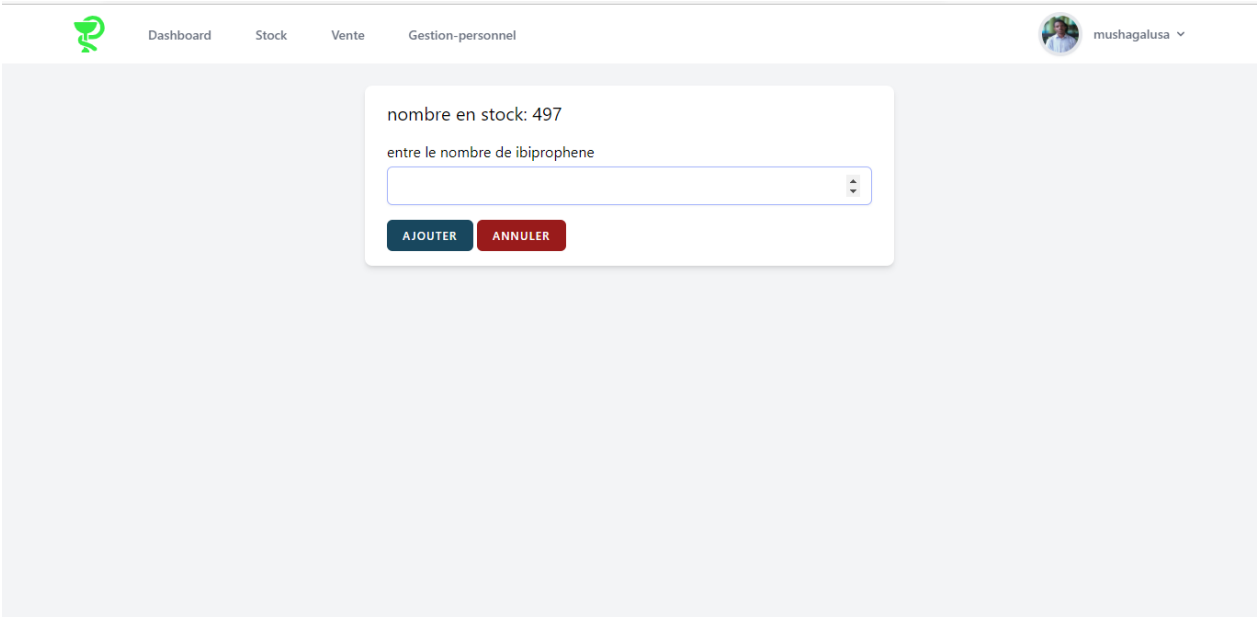


Figure 20 interface pour ajouter un médicament au panier

3.2.3.10 Modification d'un panier en cours

En cliquant sur le bouton « panier en cours » de la Figure 19 on accède à l'interface du panier en cours dans lequel on peut voir les médicaments qu'on va vendre et c'est dans cette interface où l'on peut procéder à la vente en cliquant sur bouton confirmer ou à l'annulation des commandes du client en cliquant sur le bouton annuler

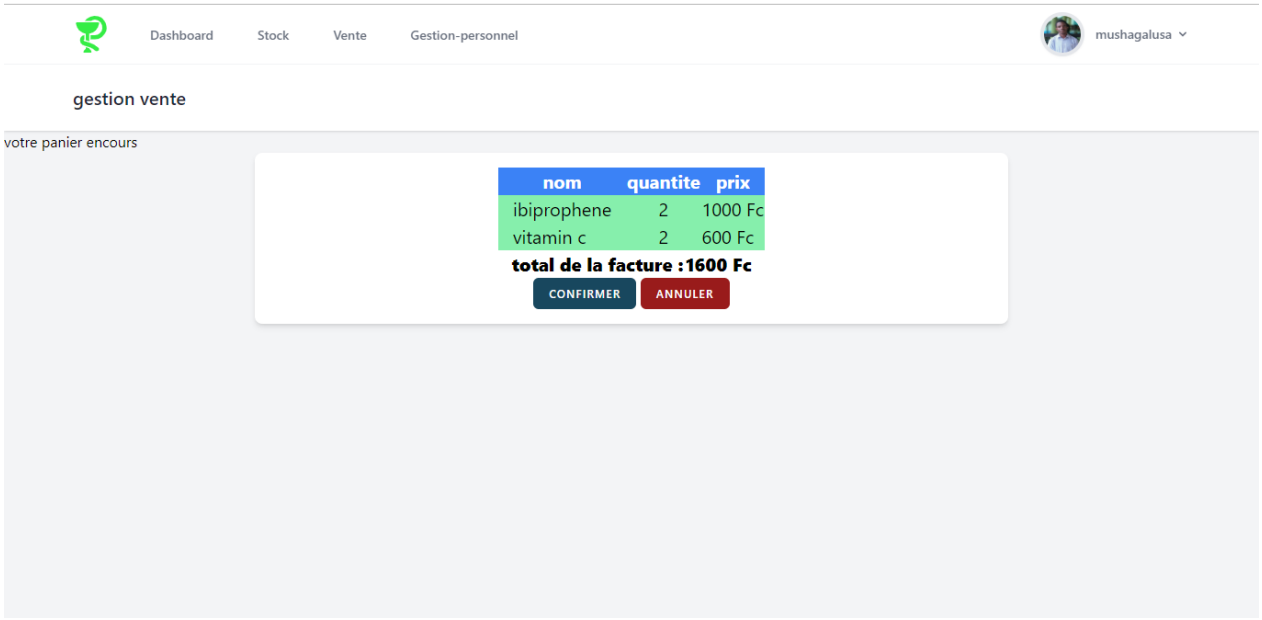


Figure 21 interface pour modifier le panier en cours

3.2.3.11 Gestion des utilisateurs

En cliquant sur l'option « vente » dans la barre de navigation, l'utilisateur accède à l'interface de gestion des utilisateurs, cette interface nous permet de faire les différentes actions par rapport à la gestion des utilisateurs.

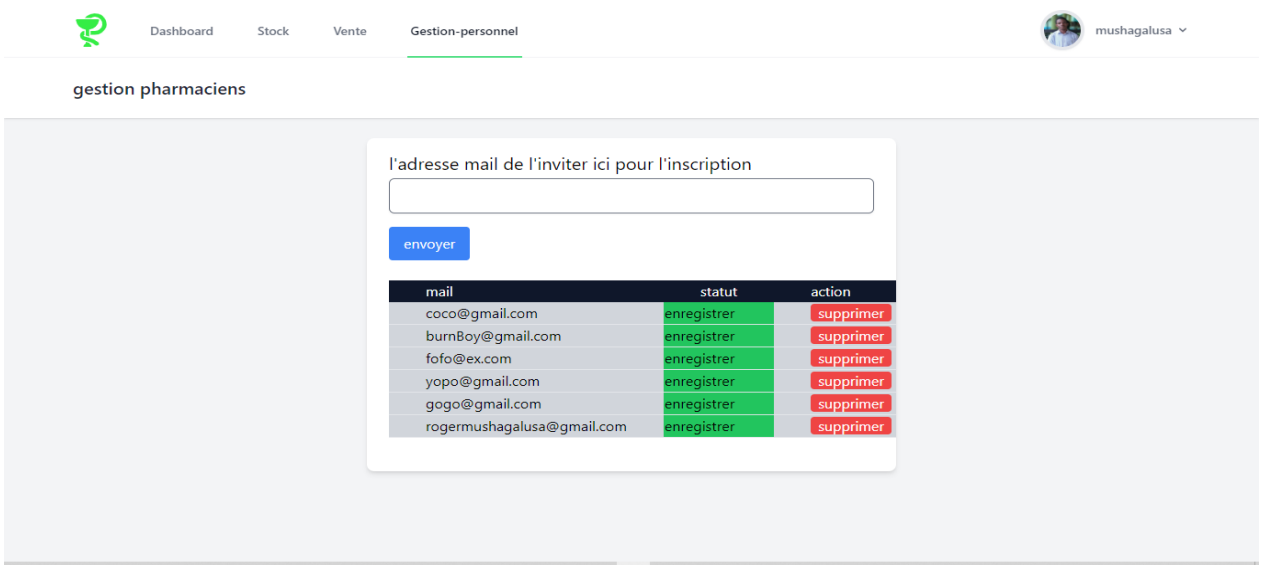


Figure 22interface de gestion des utilisateurs

3.2.3.12 Ajouter un pharmacien

Le champs vide de la Figure 23 nous permet d'insérer l'adresse mail d'un nouvel utilisateur pour lui envoyer une invitation en cliquant sur le bouton « envoyer ».

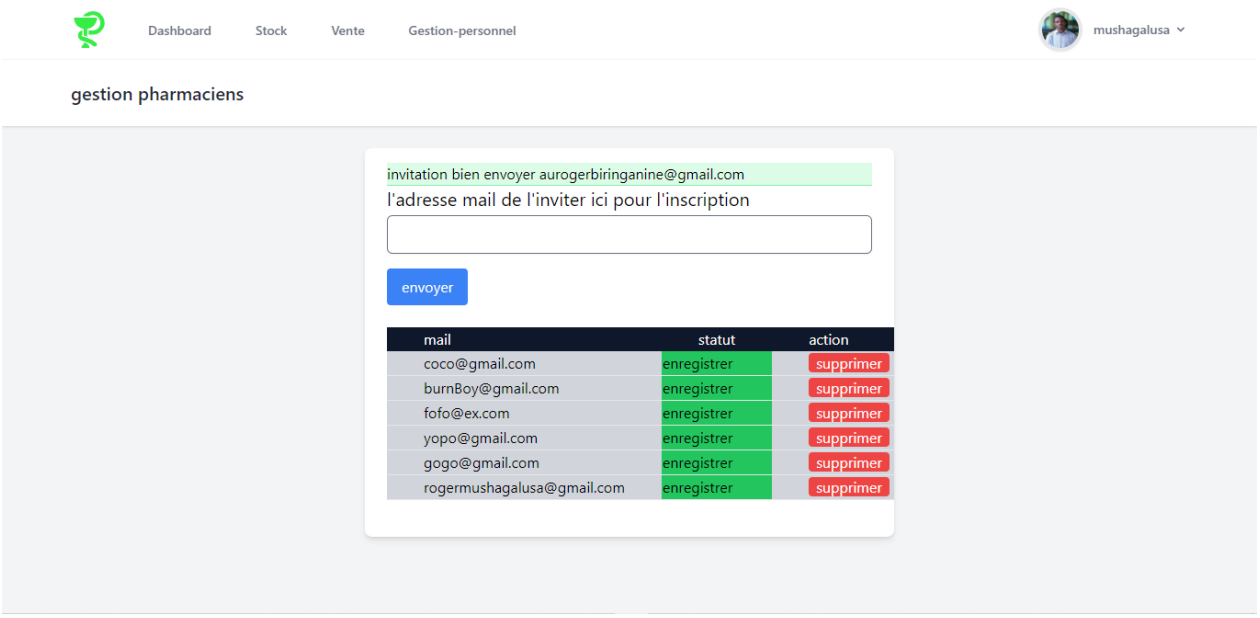


Figure 23 ajouter un pharmacien

3.2.3.13 MailDev

Etant donné l'utilisation de notre site en locale, nous avons utilisé MailDev qui est un serveur SMPT couplé à une interface web qui intercepte les emails émanant d'un serveur web afin de les visualiser et les tester. Cette interface nous a permis de créer des utilisateurs en les envoyant un mail pour la connexion comme le montre la Figure 24

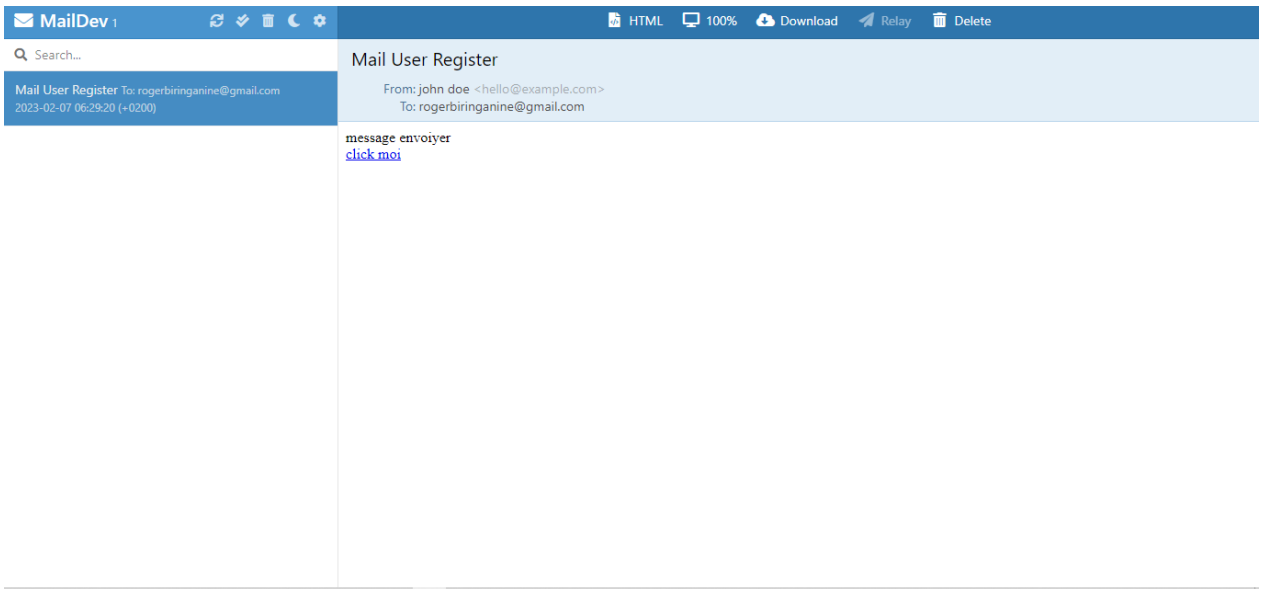


Figure 24 interface de Maildev

3.2.3.14 Inscription du pharmacien

Après l'envoi du mail au nouveau pharmacien celui-ci en ouvrant le mail clique sur le bouton « click me » pour accéder à l'interface d'inscription comme le montre la Figure 25.

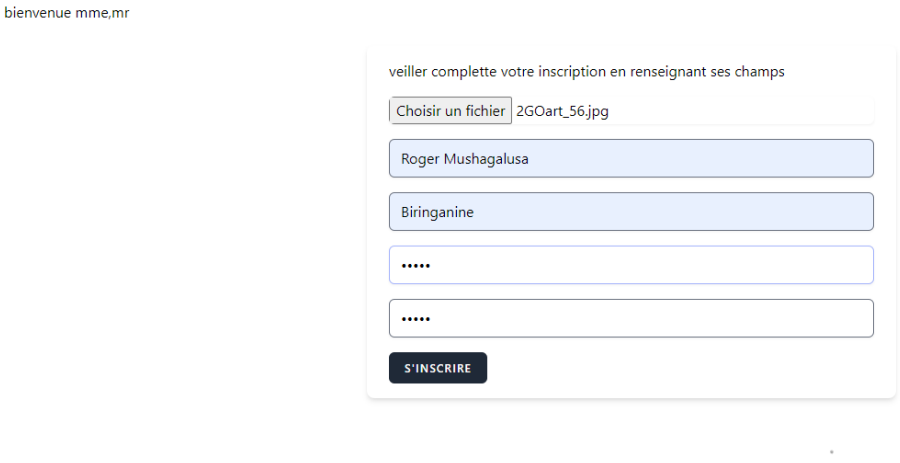


Figure 25 interface d'inscription d'un pharmacien

3.2.3.15 Dashboard du pharmacien

Après avoir compléter le formulaire d'inscription, le nouveau pharmacien accède à l'interface de dashboard (Figure 26) en cliquant sur le bouton « s'inscrire » de la Figure 25, et est donc habiliter à faire des actions sur le stock.

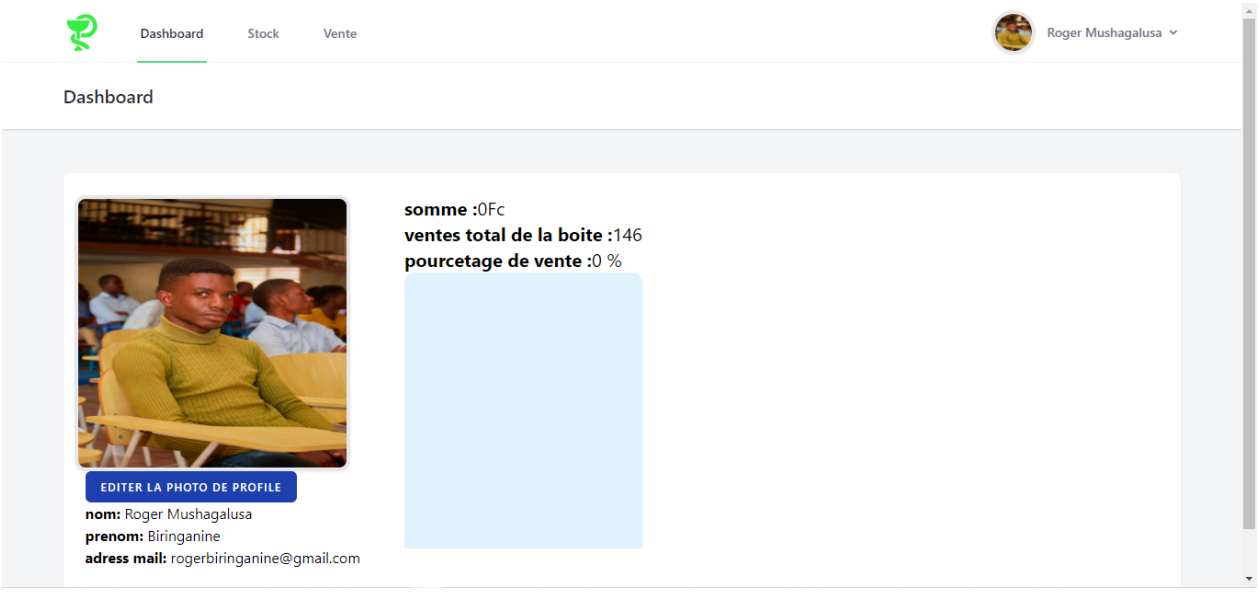


Figure 26 première connexion du nouveau pharmacien.

3.3 Conclusion partielle

Dans cette partie, il a été question de montrer ce que nous avons vu tout au long de notre travail en des interfaces qui résument notre plateforme de gestion de la pharmacie Gracia.

Conclusion générale

Notre travail a été réalisé dans le cadre d'un projet de fin de cycle et portant sur la gestion de l'officine pharmaceutique Gracia.

Au début de ce travail nous avons posé les questions suivantes :

- Comment faire pour avoir un système qui peux gérer les activités de la pharmacie dans un très bref délai tout en limitant le cout des dépenses de la main d'œuvre ?
- Comment classer l'état de stock de la pharmacie en tenant compte de la différence et de la ressemblance entre les médicaments ?
- Comment déterminer le temps de réapprovisionnement ?

De par les résultats obtenus dans ce travail, nous avons obtenu les réponses suivantes à ces questions :

- La conception d'un système informatique est une solution qui aiderais le suivis et le classement des états du stock de la pharmacie tout en tenant compte de la pertinence de chaque catégorie de médicament.
- Elle permettrait aussi l'organisation, la centralisation et la conservation de l'information ; ce qui faciliterait l'accès rapide aux données et un gain de temps considérable tout en diminuant le cout pour accéder aux données.

Le travail s'est déroulé en trois parties principales : l'analyse, la conception et l'implémentation du système. Pour la réalisation nous avons utilisé le framework Laravel du langage de programmation PHP et MySQL comme SGBD.

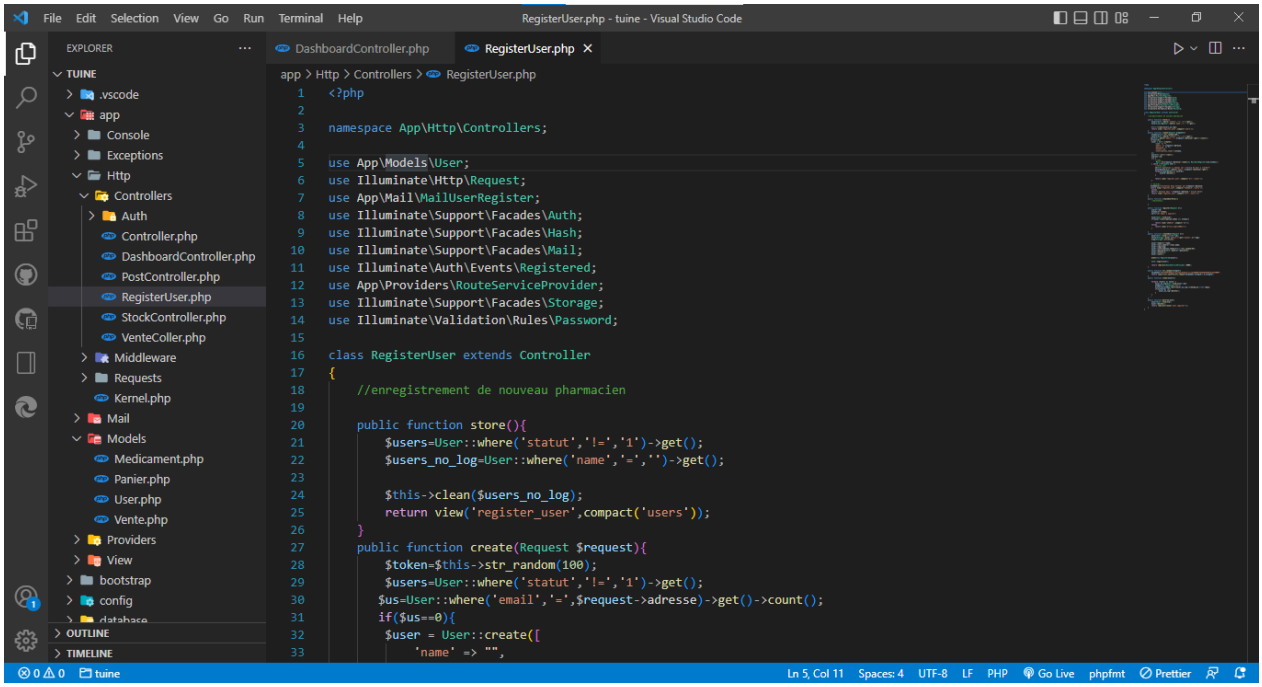
Ce projet à été pour nous un moyen efficace à l'initiation à la programmation web avec PHP.

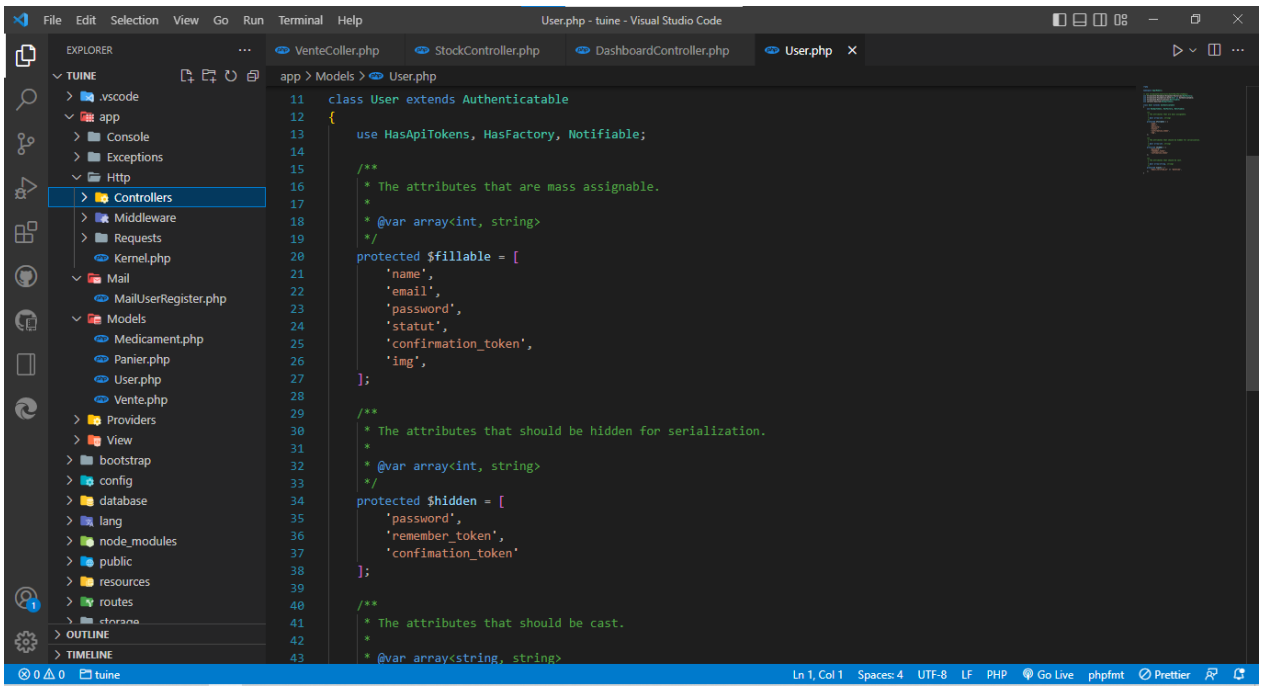
En perspective, notre application peut être améliorée en y intégrant d'autres fonctionnalité comme (gestion de livraison à domicile, commande en ligne des médicaments, etc.).

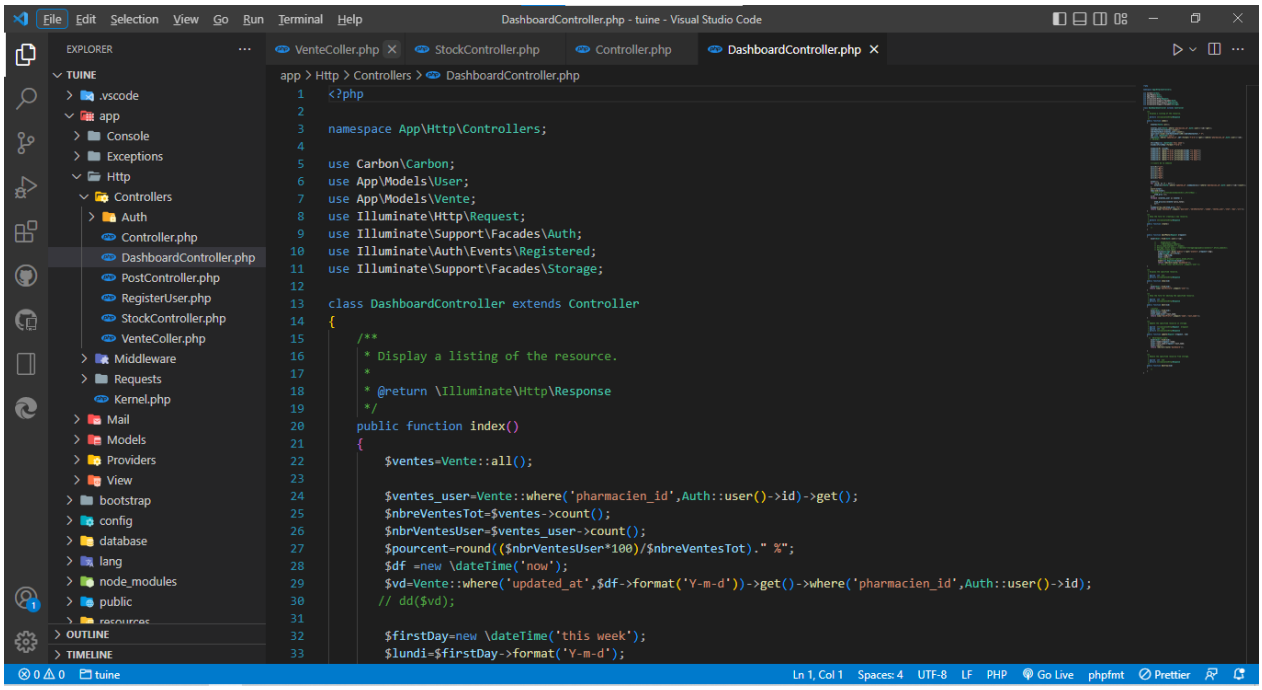
Annexe A

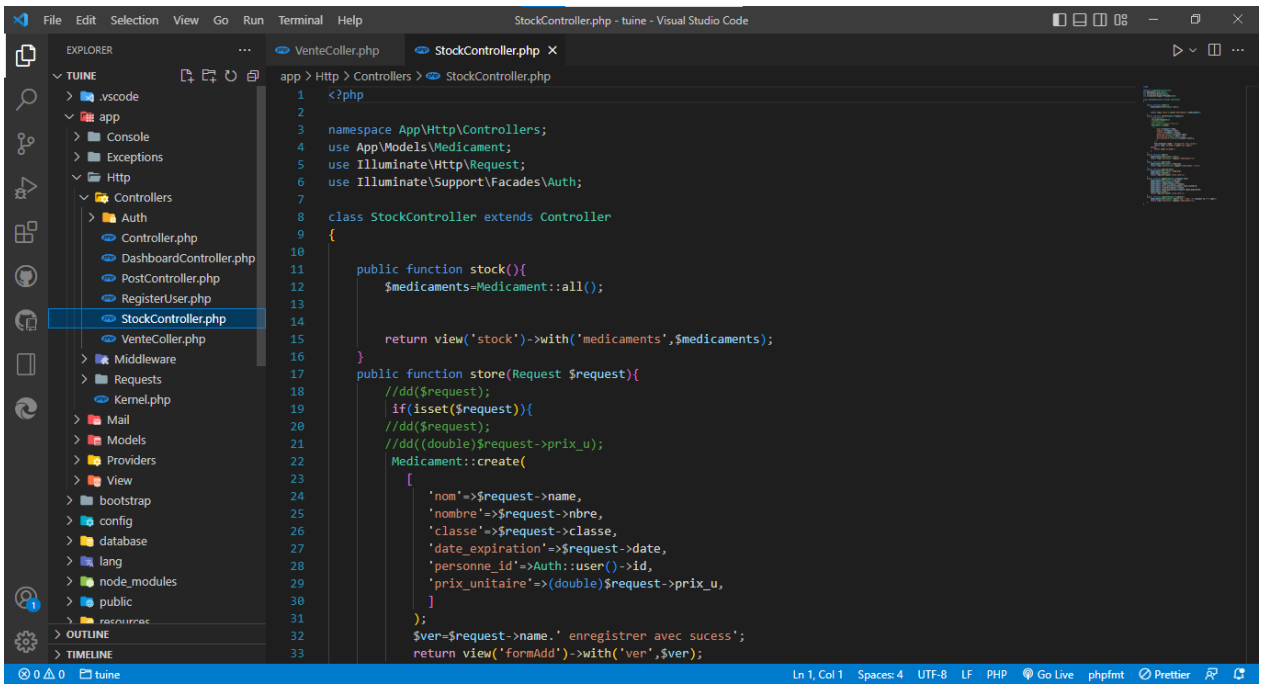
Démonstrations

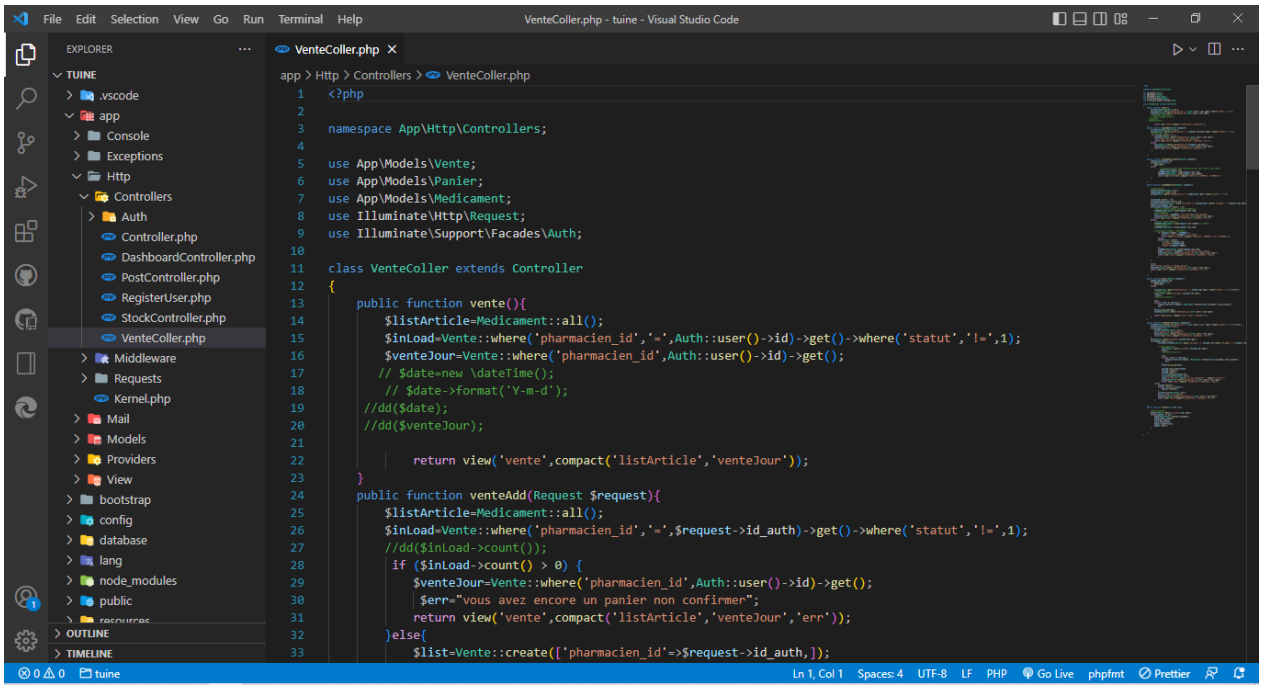
A.1 Quelques interfaces du Code source

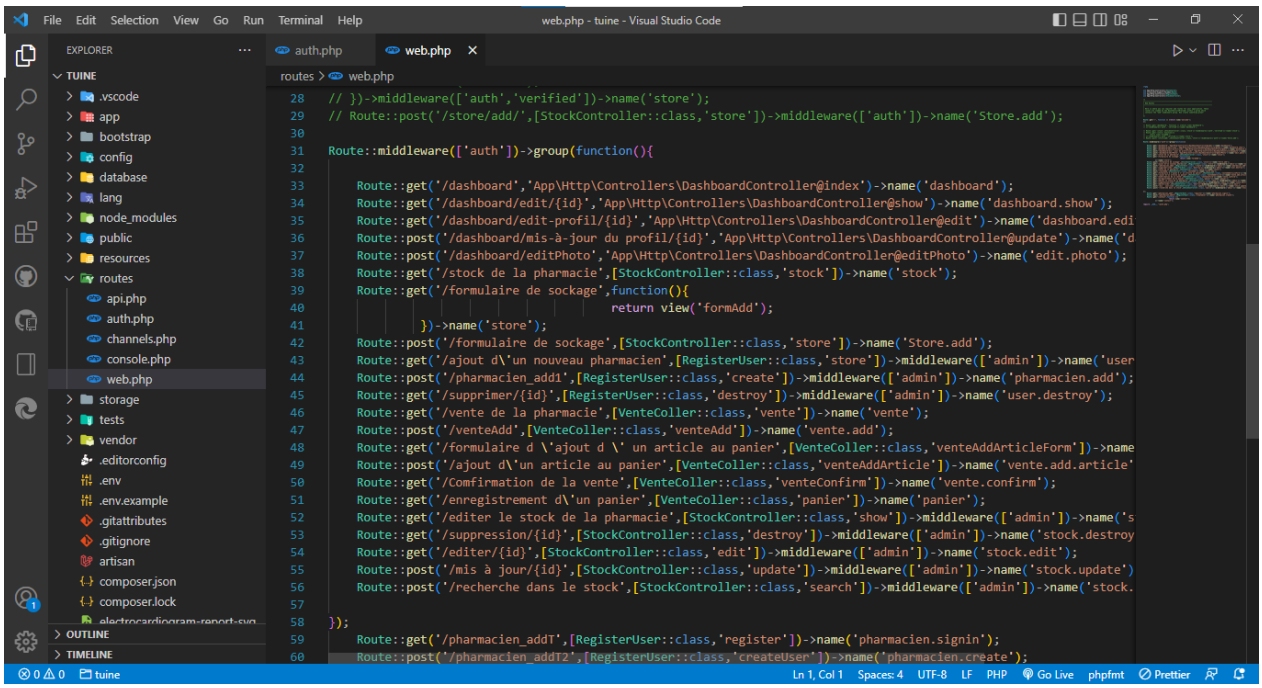


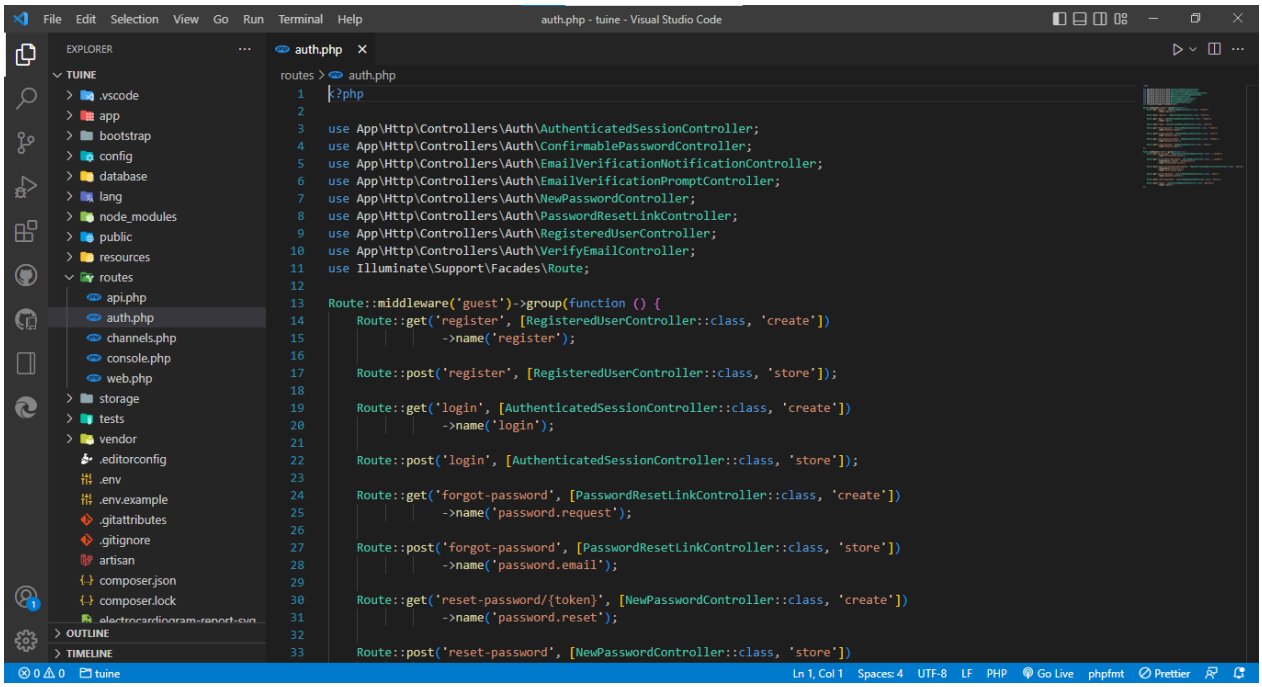

File Edit Selection View Go Run Terminal Help
User.php - tuine - Visual Studio Code
EXPLORER
TUINE
.vscode
app
Console
Exceptions
Http
Controllers
Middleware
Requests
Kernel.php
Mail
MailUserRegister.php
Models
Medicament.php
Panier.php
User.php
Vente.php
Providers
View
bootstrap
config
database
lang
node_modules
public
resources
routes
storage
OUTLINE
TIMELINE
VenteColler.php
StockController.php
DashboardController.php
User.php
app > Models > User.php
class User extends Authenticatable
{
use HasApiTokens, HasFactory, Notifiable;
/**
* The attributes that are mass assignable.
*
* @var array<int, string>
*/
protected $fillable = [
'name',
'email',
'password',
'statut',
'confirmation_token',
'img',
];
/**
* The attributes that should be hidden for serialization.
*
* @var array<int, string>
*/
protected $hidden = [
'password',
'remember_token',
'confimation_token'
];
/**
* The attributes that should be cast.
*
* @var array<string, string>
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier

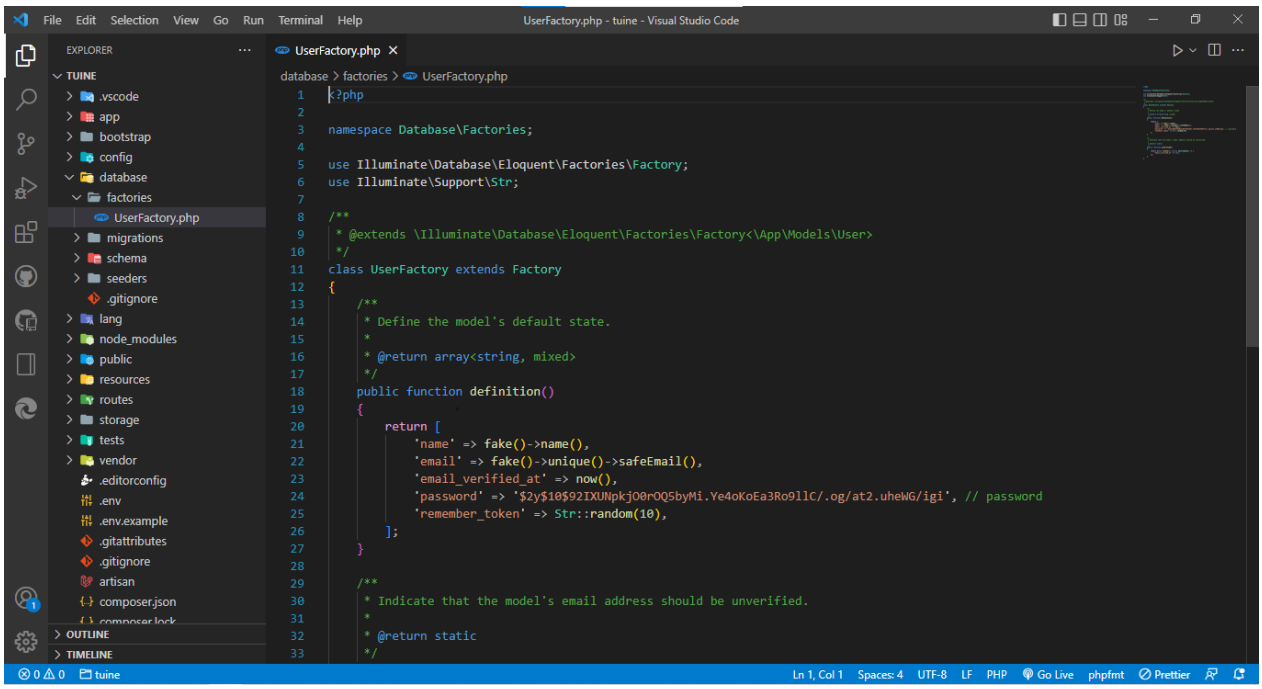

File Edit Selection View Go Run Terminal Help
DashboardController.php - tuine - Visual Studio Code
EXPLORER
TUINE
.vscode
app
Console
Exceptions
Http
Controllers
Auth
Controller.php
DashboardController.php
PostController.php
RegisterUser.php
StockController.php
VenteColler.php
Middleware
Requests
Kernel.php
Mail
Models
Providers
View
bootstrap
config
database
lang
node_modules
public
resources
OUTLINE
TIMELINE
VenteColler.php
StockController.php
Controller.php
DashboardController.php
app > Http > Controllers > DashboardController.php
<?php
namespace App\Http\Controllers;
use Carbon\Carbon;
use App\Models\User;
use App\Models\Vente;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Auth\Events\Registered;
use Illuminate\Support\Facades\Storage;
class DashboardController extends Controller
{
/**
* Display a listing of the resource.
*
* @return \Illuminate\Http\Response
*/
public function index()
{
$ventes=Vente::all();
$ventes_user=Vente::where('pharmacien_id',Auth::user()->id)->get();
$nbreVentesTot=$ventes->count();
$nbrVentesUser=$ventes_user->count();
$pourcent=round(($nbrVentesUser*100)/$nbreVentesTot)." %";
$df =new \dateTime('now');
$vd=Vente::where('updated_at',$df->format('Y-m-d'))->get()->where('pharmacien_id',Auth::user()->id);
// dd($vd);
$firstDay=new \dateTime('this week');
$lundi=$firstDay->format('Y-m-d');
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier

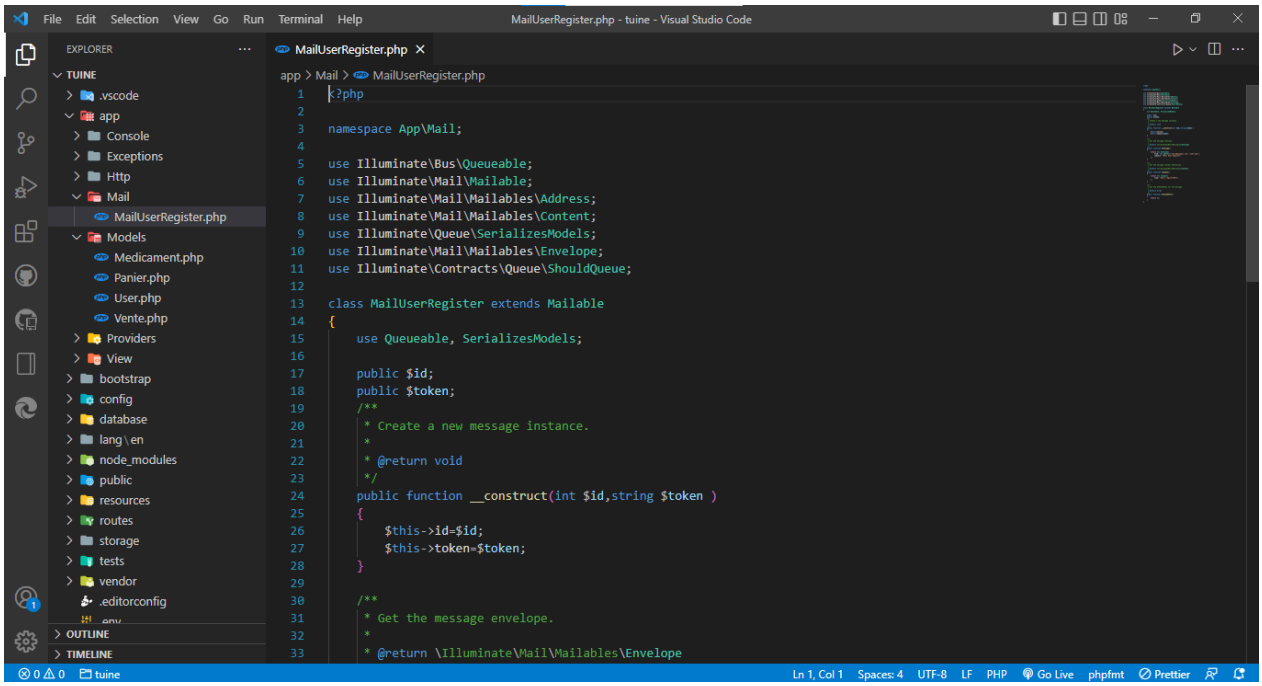

File Edit Selection View Go Run Terminal Help
StockController.php - tuine - Visual Studio Code
EXPLORER
TUINE
.vscode
app
Console
Exceptions
Http
Controllers
Auth
Controller.php
DashboardController.php
PostController.php
RegisterUser.php
StockController.php
VenteColler.php
Middleware
Requests
Kernel.php
Mail
Models
Providers
View
bootstrap
config
database
lang
node_modules
public
OUTLINE
TIMELINE
VenteColler.php
StockController.php
app > Http > Controllers > StockController.php
<?php
namespace App\Http\Controllers;
use App\Models\Medicament;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
class StockController extends Controller
{
public function stock(){
$medicaments=Medicament::all();
return view('stock')->with('medicaments',$medicaments);
}
public function store(Request $request){
//dd($request);
if(isset($request)){
//dd($request);
//dd((double)$request->prix_u);
Medicament::create(
[
'nom'=>$request->name,
'nombre'=>$request->nbre,
'classe'=>$request->classe,
'date_expiration'=>$request->date,
'personne_id'=>Auth::user()->id,
'prix_unitaire'=>(double)$request->prix_u,
]
);
$ver=$request->name.' enregistrer avec sucess';
return view('formAdd')->with('ver',$ver);
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier
tuine

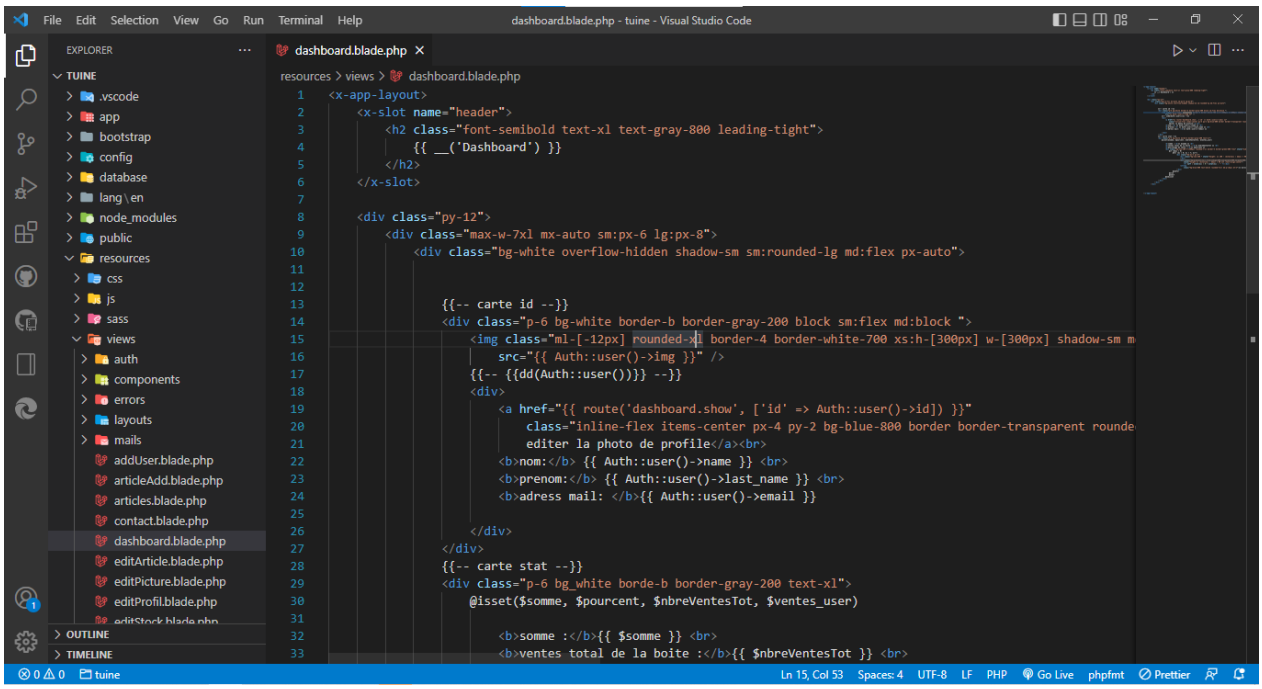

File Edit Selection View Go Run Terminal Help
VenteColler.php - tuine - Visual Studio Code
EXPLORER
TUINE
.vscode
app
Console
Exceptions
Http
Controllers
Auth
Controller.php
DashboardController.php
PostController.php
RegisterUser.php
StockController.php
VenteColler.php
Middleware
Requests
Kernel.php
Mail
Models
Providers
View
bootstrap
config
database
lang
node_modules
public
OUTLINE
TIMELINE
VenteColler.php
app > Http > Controllers > VenteColler.php
<?php
namespace App\Http\Controllers;
use App\Models\Vente;
use App\Models\Panier;
use App\Models\Medicament;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
class VenteColler extends Controller
{
public function vente(){
$listArticle=Medicament::all();
$inLoad=Vente::where('pharmacien_id','=',Auth::user()->id)->get()->where('statut','!=',1);
$venteJour=Vente::where('pharmacien_id',Auth::user()->id)->get();
// $date=new \dateTime();
// $date->format('Y-m-d');
//dd($date);
//dd($venteJour);
return view('vente',compact('listArticle','venteJour'));
}
public function venteAdd(Request $request){
$listArticle=Medicament::all();
$inLoad=Vente::where('pharmacien_id','=',$request->id_auth)->get()->where('statut','!=',1);
//dd($inLoad->count());
if ($inLoad->count() > 0) {
$venteJour=Vente::where('pharmacien_id',Auth::user()->id)->get();
$err="vous avez encore un panier non confirmer";
return view('vente',compact('listArticle','venteJour','err'));
}else{
$list=Vente::create(['pharmacien_id'=>$request->id_auth,]);
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier
tuine


File Edit Selection View Go Run Terminal Help
web.php - tuine - Visual Studio Code
EXPLORER
TUINE
auth.php
web.php
routes > web.php
// })->middleware(['auth','verified'])->name('store');
// Route::post('/store/add/',[StockController::class,'store'])->middleware(['auth'])->name('Store.add');
Route::middleware(['auth'])->group(function(){
Route::get('/dashboard','App\Http\Controllers\DashboardController@index')->name('dashboard');
Route::get('/dashboard/edit/{id}','App\Http\Controllers\DashboardController@show')->name('dashboard.show');
Route::get('/dashboard/edit-profil/{id}','App\Http\Controllers\DashboardController@edit')->name('dashboard.edi
Route::post('/dashboard/mis-à-jour du profil/{id}','App\Http\Controllers\DashboardController@update')->name('d
Route::post('/dashboard/editPhoto','App\Http\Controllers\DashboardController@editPhoto')->name('edit.photo');
Route::get('/stock de la pharmacie',[StockController::class,'stock'])->name('stock');
Route::get('/formulaire de sockage',function(){
return view('formAdd');
})->name('store');
Route::post('/formulaire de sockage',[StockController::class,'store'])->name('Store.add');
Route::get('/ajout d\'un nouveau pharmacien',[RegisterUser::class,'store'])->middleware(['admin'])->name('user
Route::post('/pharmacien_add1',[RegisterUser::class,'create'])->middleware(['admin'])->name('pharmacien.add');
Route::get('/supprimer/{id}',[RegisterUser::class,'destroy'])->middleware(['admin'])->name('user.destroy');
Route::get('/vente de la pharmacie',[VenteColler::class,'vente'])->name('vente');
Route::post('/venteAdd',[VenteColler::class,'venteAdd'])->name('vente.add');
Route::get('/formulaire d \'ajout d \' un article au panier',[VenteColler::class,'venteAddArticleForm'])->name
Route::post('/ajout d\'un article au panier',[VenteColler::class,'venteAddArticle'])->name('vente.add.article'
Route::get('/Comfirmation de la vente',[VenteColler::class,'venteConfirm'])->name('vente.confirm');
Route::get('/enregistrement d\'un panier',[VenteColler::class,'panier'])->name('panier');
Route::get('/editer le stock de la pharmacie',[StockController::class,'show'])->middleware(['admin'])->name('s
Route::get('/suppression/{id}',[StockController::class,'destroy'])->middleware(['admin'])->name('stock.destroy
Route::get('/editer/{id}',[StockController::class,'edit'])->middleware(['admin'])->name('stock.edit');
Route::post('/mis à jour/{id}',[StockController::class,'update'])->middleware(['admin'])->name('stock.update')
Route::post('/recherche dans le stock',[StockController::class,'search'])->middleware(['admin'])->name('stock.
});
Route::get('/pharmacien_addT',[RegisterUser::class,'register'])->name('pharmacien.signin');
Route::post('/pharmacien_addT2',[RegisterUser::class,'createUser'])->name('pharmacien.create');
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier


File Edit Selection View Go Run Terminal Help
auth.php - tuine - Visual Studio Code
EXPLORER
TUINE
auth.php
routes > auth.php
<?php
use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use Illuminate\Support\Facades\Route;
Route::middleware('guest')->group(function () {
Route::get('register', [RegisteredUserController::class, 'create'])
->name('register');
Route::post('register', [RegisteredUserController::class, 'store']);
Route::get('login', [AuthenticatedSessionController::class, 'create'])
->name('login');
Route::post('login', [AuthenticatedSessionController::class, 'store']);
Route::get('forgot-password', [PasswordResetLinkController::class, 'create'])
->name('password.request');
Route::post('forgot-password', [PasswordResetLinkController::class, 'store'])
->name('password.email');
Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])
->name('password.reset');
Route::post('reset-password', [NewPasswordController::class, 'store'])
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier


File Edit Selection View Go Run Terminal Help
UserFactory.php - tuine - Visual Studio Code
EXPLORER
TUINE
.vscode
app
bootstrap
config
database
factories
UserFactory.php
migrations
schema
seeders
.gitignore
lang
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
OUTLINE
TIMELINE
UserFactory.php
database > factories > UserFactory.php
<?php
namespace Database\Factories;
use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;
/**
* @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\User>
*/
class UserFactory extends Factory
{
/**
* Define the model's default state.
*
* @return array<string, mixed>
*/
public function definition()
{
return [
'name' => fake()->name(),
'email' => fake()->unique()->safeEmail(),
'email_verified_at' => now(),
'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', // password
'remember_token' => Str::random(10),
];
}
/**
* Indicate that the model's email address should be unverified.
*
* @return static
*/
0 0 tuine
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier


File Edit Selection View Go Run Terminal Help
MailUserRegister.php - tuine - Visual Studio Code
EXPLORER
TUINE
.vscode
app
Console
Exceptions
Http
Mail
MailUserRegister.php
Models
Medicament.php
Panier.php
User.php
Vente.php
Providers
View
bootstrap
config
database
lang \ en
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
OUTLINE
TIMELINE
MailUserRegister.php
app > Mail > MailUserRegister.php
<?php
namespace App\Mail;
use Illuminate\Bus\Queueable;
use Illuminate\Mail\Mailable;
use Illuminate\Mail\Mailables\Address;
use Illuminate\Mail\Mailables\Content;
use Illuminate\Queue\SerializesModels;
use Illuminate\Mail\Mailables\Envelope;
use Illuminate\Contracts\Queue\ShouldQueue;
class MailUserRegister extends Mailable
{
use Queueable, SerializesModels;
public $id;
public $token;
/**
* Create a new message instance.
*
* @return void
*/
public function __construct(int $id,string $token)
{
$this->id=$id;
$this->token=$token;
}
/**
* Get the message envelope.
*
* @return \Illuminate\Mail\Mailables\Envelope
0 0 tuine
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier


File Edit Selection View Go Run Terminal Help
dashboard.blade.php - tuine - Visual Studio Code
EXPLORER
TUINE
dashboard.blade.php
resources > views > dashboard.blade.php
<x-app-layout>
<x-slot name="header">
<h2 class="font-semibold text-xl text-gray-800 leading-tight">
{{ __('Dashboard') }}
</h2>
</x-slot>

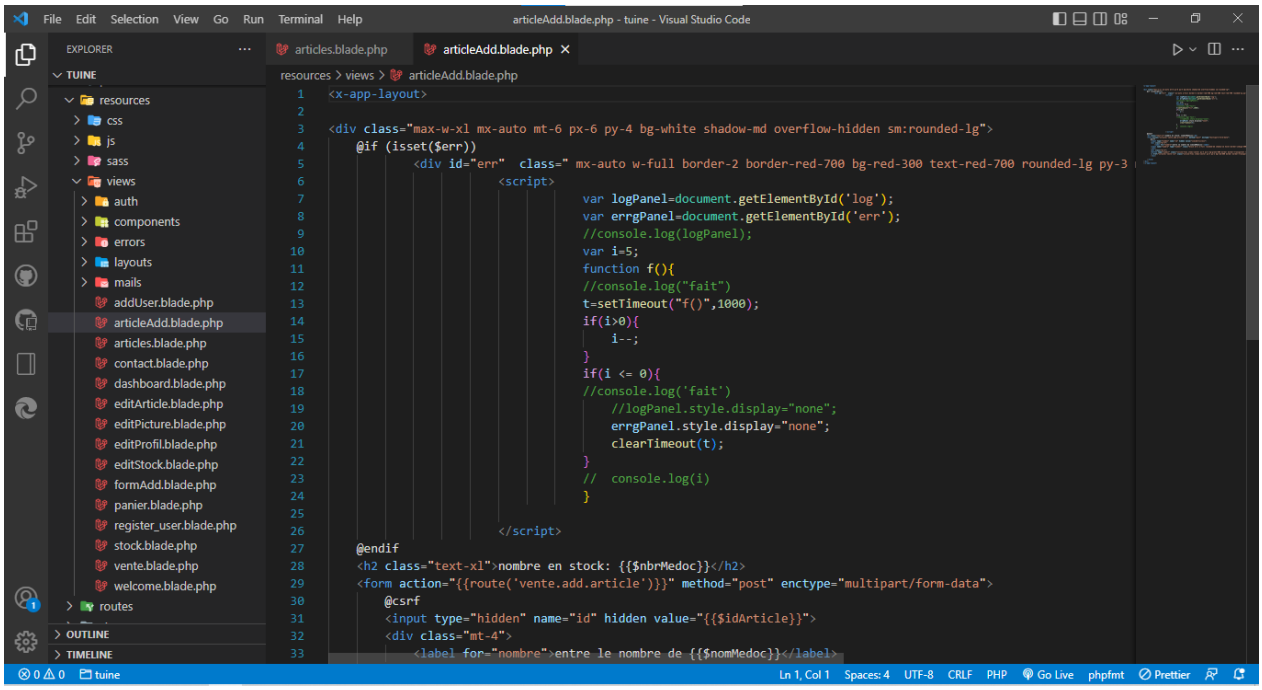
{{-- carte id --}}

<img class="ml-[-12px] rounded-xl border-4 border-white-700 xs:h-[300px] w-[300px] shadow-sm m
src="{{ Auth::user()->img }}" />
{{-- {{dd(Auth::user())}} --}}

 Auth::user()->id]) }}"
class="inline-flex items-center px-4 py-2 bg-blue-800 border border-transparent rounde
editer la photo de profile
nom: {{ Auth::user()->name }}
prenom: {{ Auth::user()->last_name }}
adress mail: {{ Auth::user()->email }}

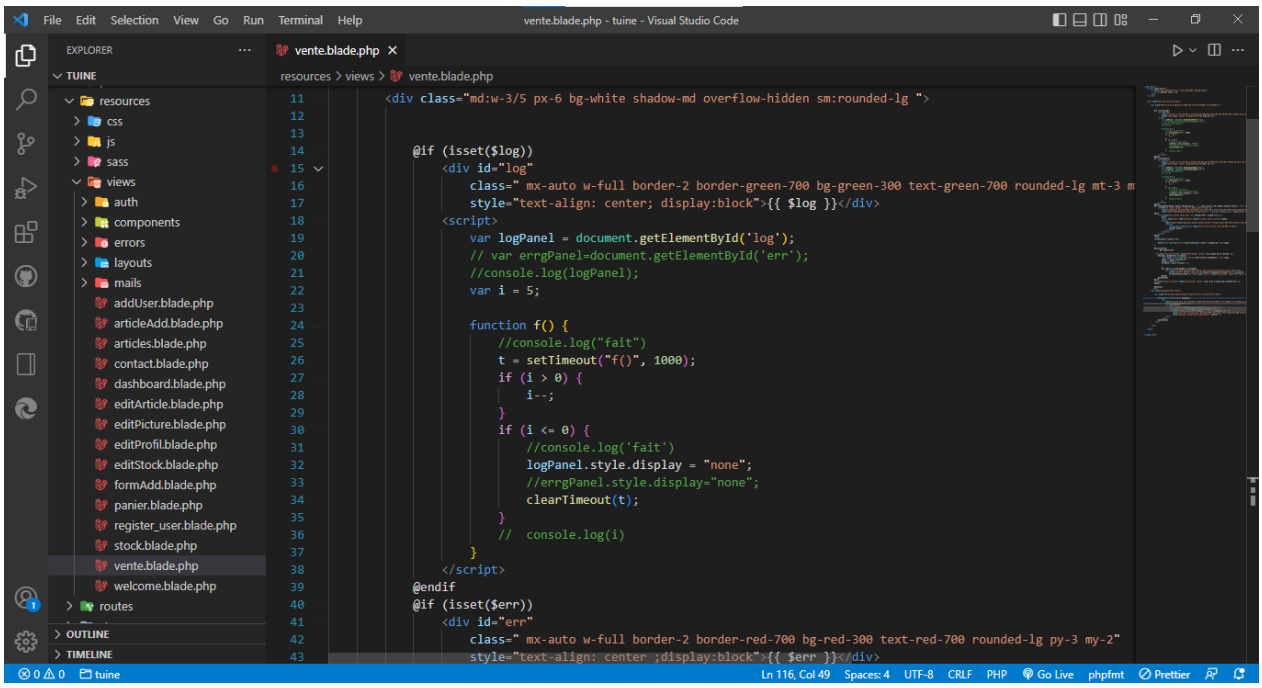
{{-- carte stat --}}

@isset($somme, $pourcent, $nbreVentesTot, $ventes_user)
somme :{{ $somme }}
ventes total de la boite :{{ $nbreVentesTot }}
OUTLINE
TIMELINE
Ln 15, Col 53 Spaces: 4 UTF-8 LF PHP Go Live phpfmt Prettier

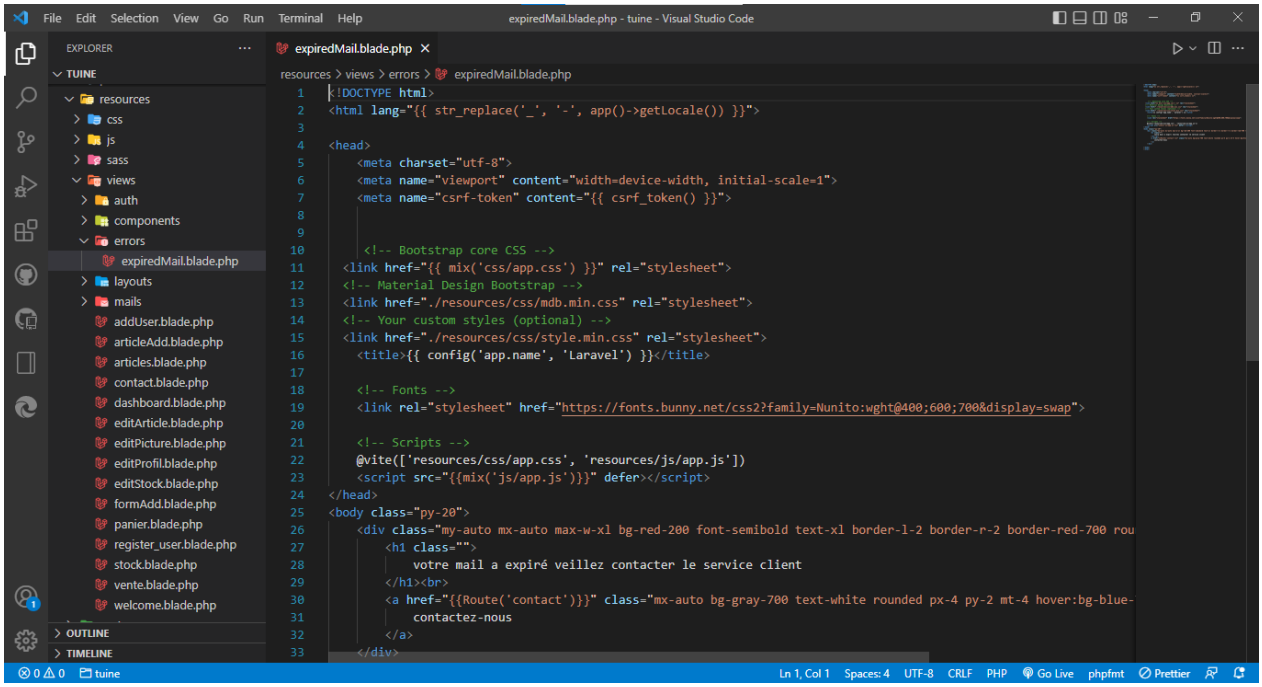

File Edit Selection View Go Run Terminal Help
articleAdd.blade.php - tuine - Visual Studio Code
EXPLORER
TUINE
articles.blade.php
articleAdd.blade.php
resources > views > articleAdd.blade.php
<x-app-layout>

@if (isset($err))
<div id="err" class=" mx-auto w-full border-2 border-red-700 bg-red-300 text-red-700 rounded-lg py-3
<script>
var logPanel=document.getElementById('log');
var errgPanel=document.getElementById('err');
//console.log(logPanel);
var i=5;
function f(){
//console.log("fait")
t=setTimeout("f()",1000);
if(i>0){
i--;
}
if(i <= 0){
//console.log('fait')
//logPanel.style.display="none";
errgPanel.style.display="none";
clearTimeout(t);
}
// console.log(i)
}
</script>
@endif
<h2 class="text-xl">nombre en stock: {{$nbrMedoc}}</h2>
<form action="{{route('vente.add.article')}}" method="post" enctype="multipart/form-data">
@csrf
<input type="hidden" name="id" hidden value="{{$idArticle}}">

<label for="nombre">entre le nombre de {{$nomMedoc}}</label>
addUser.blade.php
articleAdd.blade.php
articles.blade.php
contact.blade.php
dashboard.blade.php
editArticle.blade.php
editPicture.blade.php
editProfil.blade.php
editStock.blade.php
formAdd.blade.php
panier.blade.php
register_user.blade.php
stock.blade.php
vente.blade.php
welcome.blade.php
routes
OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF PHP Go Live phpfmt Prettier


File Edit Selection View Go Run Terminal Help
vente.blade.php - tuine - Visual Studio Code
EXPLORER
TUINE
vente.blade.php
resources > views > vente.blade.php
resources
css
js
sass
views
auth
components
errors
layouts
mails
addUser.blade.php
articleAdd.blade.php
articles.blade.php
contact.blade.php
dashboard.blade.php
editArticle.blade.php
editPicture.blade.php
editProfil.blade.php
editStock.blade.php
formAdd.blade.php
panier.blade.php
register_user.blade.php
stock.blade.php
vente.blade.php
welcome.blade.php
routes
OUTLINE
TIMELINE

@if (isset($log))
<div id="log"
class=" mx-auto w-full border-2 border-green-700 bg-green-300 text-green-700 rounded-lg mt-3 m
style="text-align: center; display:block">{{ $log }}
<script>
var logPanel = document.getElementById('log');
// var errgPanel=document.getElementById('err');
//console.log(logPanel);
var i = 5;
function f() {
//console.log("fait")
t = setTimeout("f()", 1000);
if (i > 0) {
i--;
}
if (i <= 0) {
//console.log('fait')
logPanel.style.display = "none";
//errgPanel.style.display="none";
clearTimeout(t);
}
// console.log(i)
}
</script>
@endif
@if (isset($err))
<div id="err"
class=" mx-auto w-full border-2 border-red-700 bg-red-300 text-red-700 rounded-lg py-3 my-2"
style="text-align: center ;display:block">{{ $err }}
0 0 tuine
Ln 116, Col 49 Spaces: 4 UTF-8 CRLF PHP Go Live phpfmt Prettier


File Edit Selection View Go Run Terminal Help
expiredMail.blade.php - tuine - Visual Studio Code
EXPLORER
TUINE
expiredMail.blade.php
resources > views > errors > expiredMail.blade.php
resources
css
js
sass
views
auth
components
errors
expiredMail.blade.php
layouts
mails
addUser.blade.php
articleAdd.blade.php
articles.blade.php
contact.blade.php
dashboard.blade.php
editArticle.blade.php
editPicture.blade.php
editProfil.blade.php
editStock.blade.php
formAdd.blade.php
panier.blade.php
register_user.blade.php
stock.blade.php
vente.blade.php
welcome.blade.php
OUTLINE
TIMELINE
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="csrf-token" content="{{ csrf_token() }}">
<!-- Bootstrap core CSS -->
<link href="{{ mix('css/app.css') }}" rel="stylesheet">
<!-- Material Design Bootstrap -->
<link href="./resources/css/mdb.min.css" rel="stylesheet">
<!-- Your custom styles (optional) -->
<link href="./resources/css/style.min.css" rel="stylesheet">
<title>{{ config('app.name', 'Laravel') }}</title>
<!-- Fonts -->
<link rel="stylesheet" href="https://fonts.bunny.net/css2?family=Nunito:wght@400;600;700&display=swap">
<!-- Scripts -->
@vite(['resources/css/app.css', 'resources/js/app.js'])
<script src="{{mix('js/app.js')}}" defer></script>
</head>
<body class="py-20">
<div class="my-auto mx-auto max-w-xl bg-red-200 font-semibold text-xl border-l-2 border-r-2 border-red-700 rou
<h1 class="">
votre mail a expiré veillez contacter le service client
</h1>
<a href="{{Route('contact')}}" class="mx-auto bg-gray-700 text-white rounded px-4 py-2 mt-4 hover:bg-blue-
contactez-nous

0 0 tuine
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF PHP Go Live phpfmt Prettier

Bibliographie

[1] K. K. Christian, CONCEPTION D'UN SYSTEME DE GESTION INFORMATISEE DES PATIENTS DANS UN HOPITAL, Goma, 2014-2015.

[2] P. D.-I. C. Takenga, Informatique de gestion, Goma, 2022.

[3] C. Gribaumont, Administrez vos bases de données avec MySQL, Site du Zéro, 2012.

[4] wikipedia, «Wikipedia,» 22 7 2018. [En ligne]. Available: https://fr.m.wikipediq.org/wiki/Client-serveur. [Accès le 20 12 2022].

[5] TechTarget, «lemagit,» TechTarget, [En ligne]. Available: lemagit.fr/definition/MySQL. [Accès le 8 Octobre 2022].

[6] N. Gaertner et P.-A. Muller, Modélisation objet avec UML, Eyrolles, 2003.

[7] M. N. François, CONCEPTION ET REALISATION D'UNE APPLICATION PHARMACEUTIQUE DANS LA VILLE DE GOMA : CAS DE LA PHARMACIE NGAMUPHAR, Goma: ULPGL\GOMA, 204-2015.

[8] A. M. Donfack, Méthodes d'analyse et de conception orientée objet et Modélisation UML, Agence Universitaire de la francophonie, 2021-2022.

[9] O. Guibert, Analyse et Conception des Systèmes d'Information – Méthodes Objet : Le langage de modélisation objet UML, Bordeaux : Institut Universitaire de Technologie de l'Université Bordeaux 1, 2010.

[10] «laravel,» 08 02 2022. [En ligne]. Available: laravel.com/docs/9.x. [Accès le 2023 02 2023].

[11] JUnit.org. [En ligne]. Available: http://www.junit.org. [Accès le 2 Janvier 2019].

[12] M. Nebra, Concevez votre site web avec PHP et MySQL, Paris, 2013.

[13] V. Thuillier, Programmez en orienté objet en PHP, Peais, 2013.

[14] B. M. Dieudonné, ANALYSE ET CONCEPTION D'UN SYSTEME D'INFORMATION PAR LA METHODE MERISE : CAS DE LA BIBLIOTHEQUE UNIVERSITAIRE, Goma: ULPGL\GOMA, 2014-2015.

[15] K. K. Benjamin, MISE EN PLACE D'UNE APPLICATION NUMERIQUE DE GESTION DE LA FACTURATIN DES ABONNES : CAS DE LA REGIDESO/GOMA, GOMA: ULPGL/GOMA, 2019-2020.

[16] P. Roques, Les cahiers du programmeur UML2 : modéliser une application web, Eyrolles, 2007.

[17] F. Brissonneau, Réalisateur, *Formation UML Maitriser La Modelisation.* [Film]. france: Alphorm , 2013.