**UNIVERSITE JOSEPH KI-ZERBO**

**(UJKZ)**

**-=-=-=-=-**

**INSTITUT BURKINABE DES ARTS ET METIERS**

**(IBAM)**

****

**Master informatique**

**PROJET DE VIRTUALISATION AVEC VAGRANT**

**GROUPE 1**

**DEPLOIEMENT D’UNE APPLICATION WEB MULTI-TIERS**

**‘’Système de gestion d’une mutuelle de santé des travailleurs (SysGMS)’’**

|  |  |
| --- | --- |
| **Membres du groupe** | **Enseignant :** |
| KABORE Mahamady | Dr Flavien SOMDA |
| MILLOGO Souleymane |  |
| NIKIEMA W. Jessica |  |
| SAVADOGO Yves |  |
| TAPSOBA Chérif Ramzi Farès |  |

**Année Académique 2023-2024**

[INTRODUCTION GENERALE 4](#_Toc171980685)

[CHAPITRE I : ETUDE PREALABLE 4](#_Toc171980686)

[I. Présentation du Thème 4](#_Toc171980687)

[1. Contexte et justification 4](#_Toc171980688)

[2. Problématique 5](#_Toc171980689)

[3. Objectifs 5](#_Toc171980690)

[4. Résultats attendus 5](#_Toc171980691)

[5. Estimation du coût de réalisation 6](#_Toc171980692)

[II. Méthode d’analyse et conception 7](#_Toc171980693)

[1 Processus de développement 7](#_Toc171980694)

[2 Langage de modélisation 9](#_Toc171980695)

[III. Groupe de travail et distribution des rôles 10](#_Toc171980696)

[1. Groupe de pilotage 10](#_Toc171980697)

[2. Groupe de projet 10](#_Toc171980698)

[3. Groupe des utilisateurs 11](#_Toc171980699)

[IV. Planning des travaux 11](#_Toc171980700)

[CHAPITRE II : EXPRESSIONS DES BESOINS 12](#_Toc171980701)

[I. Description du fonctionnement attendu de l’application 12](#_Toc171980702)

[II. Spécifications fonctionnelles 12](#_Toc171980703)

[1 Identification des acteurs 14](#_Toc171980704)

[2 Identification des cas d’utilisation 14](#_Toc171980705)

[3 Diagramme de cas d’utilisation 15](#_Toc171980706)

[4 Diagramme de séquence 17](#_Toc171980707)

[III. Spécification technique 17](#_Toc171980708)

[1 Matériel nécessaire 17](#_Toc171980709)

[2 Architecture de développement 17](#_Toc171980710)

[CHAPITRE III : CONCEPTION GLOBALE 21](#_Toc171980711)

[I. Diagramme de classe 21](#_Toc171980712)

[1 Dictionnaire de données 21](#_Toc171980713)

[2 Diagramme de classe 23](#_Toc171980714)

[II. Diagramme de déploiement 25](#_Toc171980715)

[CHAPITRE IV : REALISATION 25](#_Toc171980716)

[I. Présentation des technologies et outils de réalisation 25](#_Toc171980717)

[1 Langages de programmation 25](#_Toc171980718)

[2 Plateforme de développement (Framework) 26](#_Toc171980719)

[3 Outils de conception 26](#_Toc171980720)

[4 Système de Gestion de Base de Données (SGBD) 27](#_Toc171980721)

[5 Serveur d’application 27](#_Toc171980722)

[6 Un serveur web 28](#_Toc171980723)

[II. Présentation de l’architecture MVC de l’application 28](#_Toc171980724)

[Figure x: Structure de l’architecture Modèle-Vue-Contrôleur 29](#_Toc171980725)

[III. Configuration des Machines Virtuelles 29](#_Toc171980726)

[0 Création des Vagrantfile : 30](#_Toc171980727)

[1 Provisionnement automatique : 30](#_Toc171980728)

[2 Mise en place du réseau : 30](#_Toc171980729)

[3 Partage de fichiers : 30](#_Toc171980730)

[IV. Développement de l’application 30](#_Toc171980731)

[1. Quelques maquettes IHM de notre application 30](#_Toc171980732)

[V. Déploiement de l’Application 30](#_Toc171980733)

[Scripts de provisionnement 30](#_Toc171980734)

[VI. Tests 30](#_Toc171980735)

[VII. Politique de sécurité 31](#_Toc171980736)

[1 Sensibilisation des utilisateurs 31](#_Toc171980737)

[2 Mesures de sécurité pour la phase le développement 31](#_Toc171980738)

[3 Le certificat SSL 32](#_Toc171980739)

[4 Politique de sauvegarde 32](#_Toc171980740)

[5 CONCLUSION GENERALE 33](#_Toc171980741)

INTRODUCTION GENERALE

Dans un contexte socio-économique en constante transformation, la protection sociale des travailleurs est cruciale pour garantir leur bien-être ainsi que celui de leurs familles. Les mutuelles de santé, en tant que piliers de la sécurité sociale, jouent un rôle essentiel en couvrant les risques liés à la santé. Gérées par leurs propres membres, elles visent à offrir des services de santé de qualité et accessibles, tout en favorisant la solidarité et l'entraide entre leurs adhérents.

Le système de gestion d'une mutuelle de santé des travailleurs est essentiel à son bon fonctionnement. Elle englobe divers mécanismes et procédures destinés à organiser, administrer et superviser les activités de la mutuelle afin de répondre efficacement aux besoins des adhérents.

Fort de ce constat, notre étude, intitulée **"Système de gestion d'une Mutuelle de Santé des Travailleurs"**, se propose d'examiner les différents aspects de la gestion des mutuelles de santé à travers une approche basée sur une application web multi-tiers. Cette analyse mettra en lumière les principes directeurs relatifs à la conception, au déploiement et à l’administrer d’une telle application en utilisant vagrant dans des environnements de machines virtuelles.

CHAPITRE I : ETUDE PREALABLE

1. Présentation du Thème

### **Contexte et justification**

Dans le cadre de l'évolution rapide des technologies informatiques et des besoins croissants en matière de sécurité sociale, la virtualisation et l'automatisation des infrastructures jouent un rôle crucial dans l'optimisation des systèmes de gestion des mutuelles de santé des travailleurs. Ce projet vise à répondre à ces défis en explorant l'utilisation de Vagrant pour le déploiement d'une application web multi-tiers dédiée au "Système de gestion d’une mutuelle de santé des travailleurs (SysGMS)". En virtualisant les composants clés tels que les bases de données, les serveurs d'application et les serveurs web, cette approche permettra une gestion agile et efficace des services de santé, tout en garantissant la sécurité et la disponibilité des données sensibles des adhérents.

1. Problématique

Comment concevoir et mettre en œuvre de manière efficace et sécurisée une infrastructure virtualisée pour le déploiement d'une application web multi-tiers dédiée au "Système de gestion d’une mutuelle de santé des travailleurs (SysGMS)" en utilisant Vagrant ? Quelles stratégies et quels outils sont nécessaires pour garantir la disponibilité, la sécurité et la performance de cette application tout en répondant aux exigences spécifiques des mutuelles de santé ?

1. Objectifs

L’objectif global est de mettre en place et gérer un Système de gestion d’une mutuelle de santé (SysGMS) en utilisant vagrant. Cette application Web Multi-Tiers permet de gérer les assurés (adhérents, personne à charge) et les structures partenaires, les paiements des factures des structures partenaires. Les objectifs spécifiques sont les suivants :

* Conception globale de l’application
* Réalisation de l’Application
* Configuration et déploiement de la composante bases de données MySQL avec Vagrant
* Configuration et déploiement de la composante serveur d’application Django avec Vagrant
* Configuration et déploiement de la composante serveur web Apache avec Vagrant

1. Résultats attendus

* L’application SysGMS est conçue ;
* L’application SysGMS est réalisée ;
* La Base de données MySQL est configurée et deploiyée avec Vagrant
* Le serveur d’application Django est configurée et deploiyée avec Vagrant
* Le serveur web est configurée et deploiyée avec Vagrant

### Estimation du coût de réalisation

L’estimation du coût de développement est une phase très importante lors de la conception d’un logiciel. En effet, cette opération permet de connaître les ressources humaines, financières et matérielles nécessaires pour mener à bien le projet. Il existe plusieurs méthodes pour faire cette estimation. Nous utiliserons la méthode COCOMO SIMPLIFIEE (Constructive Cost Model) pour la fiabilité de ces estimations.

* En plus, nous estimons le nombre de lignes de code de notre application à dix mille (10 000). En supposant que le salaire d’un Ingénieur-informaticien est égal à 300 000 F CFA, on peut alors estimer le coût de développement comme suit :
* - Effort = 2,4\*(10000/1000) 1,05 = 276,90 Homme/Mois
* - TDEV = 2,5\*Effort 0,38 = 2,5\*27 0,38 = 8,75 mois
* - CDEV = 27 \* 300 000 FCFA = 27 \* 30 000 FCFA = 8 100 000 FCFA
* En prenant en considération le coût du développement ainsi que celui du besoin en matériel informatique et logiciels, on obtient le coût total du projet tel que présenté comme suit :

**Tableau 2 : Devis de la réalisation du projet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Désignation** | **Caractéristiques** | **Quantité** | **Cout unitaire** | **Cout total** |
| Ordinateur | Mac Book Pro Core i7, 16 Go de RAM, 1 To de disque dur SSD | 5 | 1 200 000 | 6 000 000 |
| Serveur | Processeur : Intel Xeon Gold 16 cœurs 3,4 Ghz  Ram : 16 Go Extensible à 32 Go  Disque dur : 16 To extensible à 32 To | 1 | 12 415 000 | 12 415 000 |
| SYBASE PowerAMC | - | 1 | - | Version d’évaluation |
| GanttProject | - | 1 | - | Licence libre |
| Visual studio code | - | 1 | - | Licence libre |
| SpringBoot / Angular /Laravel | - | Ensemble |  | Licence libre |
| MySQL | - | 1 | - | Licence libre |
| Développement | - | 1 | - | 8 100 000 |
| **Cout total** | | | | **26 515  000 Fcfa** |

1. Méthode d’analyse et conception

Une méthode d'analyse et de conception est un procédé qui a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client. Pour ce faire, on part d'un énoncé informel (le besoin tel qu'il est exprimé par le client, complété par des recherches d'informations auprès des experts du domaine fonctionnel, comme les futurs utilisateurs d'un logiciel), ainsi que de l'analyse de l'existant éventuel (c'est-à-dire la manière dont les processus à traiter par le système se déroulent actuellement chez le client).

La phase d'analyse permet de lister les résultats attendus, en termes de fonctionnalités, de performance, de robustesse, de maintenance, de sécurité, d'extensibilité, d’interopérabilité, etc.

La phase de conception permet de décrire de manière non ambiguë, le plus souvent en utilisant un langage de modélisation, le fonctionnement futur du système, afin d'en faciliter la réalisation.

# **Processus de développement**

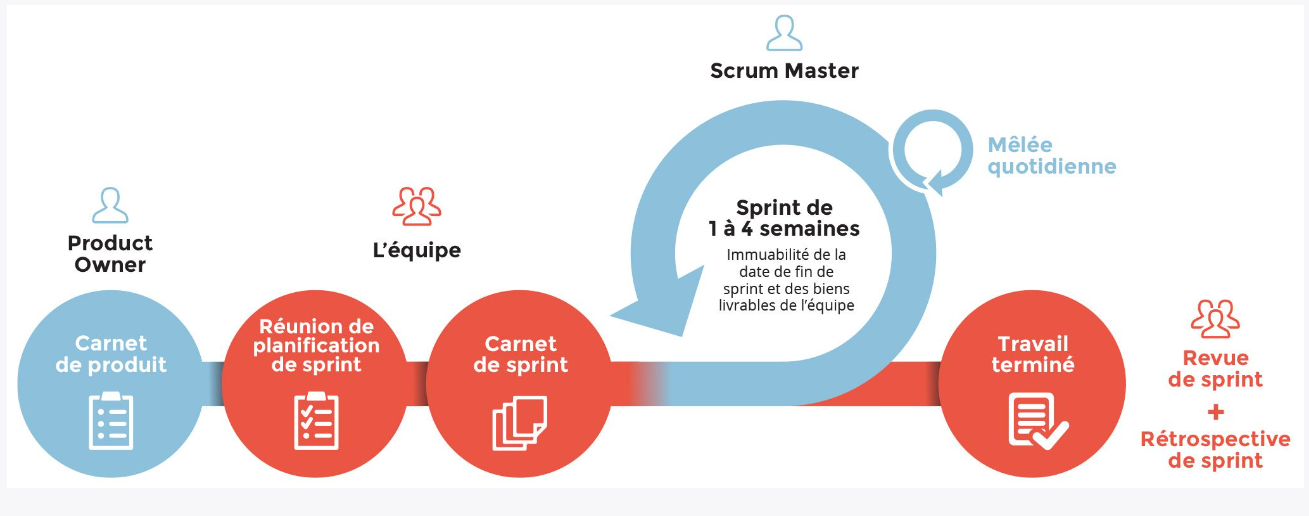
Le processus de développement constitue la démarche fondamentale permettant d’obtenir un produit logiciel dans un délai raisonnable tout en minimisant les ressources.

Dans le cadre de notre étude, nous avons opté pour la méthodologie Agile scrum en raison du temps qui nous ai imparti

La méthode agile est une approche itérative et incrémentale de gestion de projet, principalement utilisée dans le développement logiciel mais applicable à divers domaines. Elle se caractérise par une flexibilité et une adaptabilité élevées, permettant de répondre efficacement aux changements et aux évolutions des besoins des clients ou des utilisateurs finaux.

Voici les principes fondamentaux de la méthode agile :

* **Collaboration étroite avec le client** : Les équipes agiles travaillent en étroite collaboration avec les clients ou les parties prenantes pour comprendre et répondre rapidement à leurs besoins changeants.
* **Livraison itérative et incrémentale** : Le projet est découpé en petites itérations appelées "itérations" ou "sprints", généralement de 1 à 4 semaines. À la fin de chaque itération, une partie fonctionnelle du produit est livrée, ce qui permet aux clients d'avoir un retour rapide et d'ajuster les priorités.
* **Adaptabilité au changement** : L'agilité se caractérise par sa capacité à répondre rapidement aux changements de conditions, de besoins ou de technologies. Les équipes agile favorisent une attitude de flexibilité plutôt que de rigidité face aux modifications.
* **Auto-organisation des équipes** : Les équipes agile sont généralement auto-organisées et interdisciplinaires. Elles sont responsables de la planification, de l'exécution et de la gestion de leur travail, ce qui favorise un engagement plus fort et une meilleure responsabilité.
* **Communication constante** : Une communication régulière et transparente entre toutes les parties prenantes est essentielle pour garantir que tout le monde est aligné sur les objectifs du projet et sur les éventuels ajustements nécessaires.
* **Amélioration continue** : À travers des rétrospectives régulières, les équipes agiles réfléchissent sur leurs pratiques de travail et cherchent constamment à s'améliorer pour optimiser leur efficacité et la qualité du produit.



Source : <https://bubbleplan.net/blog/agile-scrum-gestion-projet/>

* Le [**Product Owner**](https://bubblemeeting.net/blog/product-owner-role-place-fiche-metier/) (Responsable produit) joue un **rôle clé** dans l’**équipe Scrum** en définissant les fonctionnalités du produit final lors de chaque itération. Il priorise les tâches et les spécificités du produit tout au long de la méthode.
* Le [**Scrum Master**](https://bubbleplan.net/blog/competences-scrum-master-methodologie-scrum/), quant à lui, occupe un [rôle de chef de projet](https://bubbleplan.net/blog/gp-qualites-chef-de-projet/)dans la [méthode Agile](https://bubbleplan.net/pedagogie-projet/methode-agile), chargé de dynamiser les interactions, le travail autonome et la communication au sein de l’équipe. De manière générale, il est responsable du **respect du processus Scrum**.
* Equipe : l’équipe s’organise elle-même et elle reste inchangée pendant toute la durée d’un sprint. Elle doit tout faire pour délivrer le produit

# **Langage de modélisation**

Un langage de modélisation est un langage artificiel qui peut être utilisé pour exprimer l’information ou la connaissance des systèmes dans une structure qui est définie par un ensemble cohérent de règles.

Pour développer une application, il faut d’abord organiser les idées, les documenter avant de commencer la réalisation tout en définissant les modules et les étapes. On appelle cette démarche « modélisation ». Pour réaliser cette modélisation, nous utiliserons le langage UML pour plusieurs raisons. Nous pouvons citer entre autres qu’il :

* présente l'avantage d'être le standard en matière de modélisation objet universellement reconnu ;
* est un langage visuel, car sa notation graphique permet d’exprimer visuellement des solutions objets facilitant ainsi la comparaison et l'évaluation de celles-ci ;
* est un langage formel et normalisé doté d'un gain de précisions et d'un gage de stabilité ;
* sert à formaliser tous les documents techniques d'un projet et permet d'affiner les détails de l'analyse au fur et à mesure de l'avancée du projet ;
* est capable d'utiliser le même atelier de génie logiciel, depuis l'expression des besoins des utilisateurs jusqu'à la génération de tout ou partie du code ;
* est un support de communication performant, car il cadre l'analyse tout en facilitant la compréhension des représentations abstraites complexes.

1. Groupe de travail et distribution des rôles

Le groupe de travail représente l’ensemble des personnes nécessaires à la réalisation du projet.

Ainsi, nous avons trois groupes de travail à savoir le groupe projet, le groupe de pilotage et le groupe utilisateur.

1. **Groupe de pilotage**

C’est le groupe dirigeant chargé de veiller au bon déroulement du projet. Il est chargé de la planification des dates clés du projet, de l’examen des propositions du groupe de projet et décide des orientations stratégiques.

1. **Groupe de projet**

Le groupe de projet est l’ensemble des personnes chargées de réaliser le projet. Il est l’intermédiaire entre le groupe de pilotage et le groupe des utilisateurs.

Chaque membre joue un rôle crucial dans le succès du projet de virtualisation et de déploiement de l'application SysGMS avec Vagrant, contribuant ainsi à la réalisation efficace de l'objectif global du projet. Les membres du groupe et leur rôles spécifiques se présentent comme suit :

| **N°** | **Membres** | **Rôle** | **Rôles spécifiques** |
| --- | --- | --- | --- |
| 01 | SAVADOGO Yves | Chef de projet |  Responsable de la supervision générale du projet.   Planification des tâches, gestion des délais.   Communication avec les parties prenantes et rapport régulier sur l'avancement du projet.   Prise de décisions stratégiques pour assurer la réussite du projet dans son ensemble. |
| 02 | MILLOGO Souleymane | Administrateur système |  Configuration et maintenance des serveurs et des environnements virtuels avec Vagrant.   Gestion des ressources système et optimisation des performances.   Support technique pour résoudre les problèmes liés à l'infrastructure. |
| 03 | NIKIEMA W. Jessica | Développeur Font End |  Responsable de la conception et du développement de l'interface utilisateur de l'application web.   Intégration des maquettes graphiques et des fonctionnalités interactives.   Optimisation de la performance et de l'accessibilité de l'interface utilisateur. |
| 04 | TAPSOBA Chérif Ramzi Farès | Développeur Back End |  Développement des fonctionnalités de l'application web côté serveur.   Gestion de la logique métier, des bases de données et des intégrations avec d'autres systèmes.   Implémentation des APIs et des services web nécessaires pour assurer le bon fonctionnement de l'application. |
| 05 | KABORE Mahamady | Testeur |  Responsable de la planification et de l'exécution des tests unitaires, d'intégration et de système.   Identification et rapport des bugs, suivi de leur résolution.   Validation de la conformité aux spécifications fonctionnelles et techniques. |

1. **Groupe des utilisateurs**

Ce groupe est constitué de l’ensemble de tous ceux qui vont utiliser le futur système. Il est composé d principalement des employés de la mutuelle de santé.

1. Planning des travaux

Pour mener à bien notre étude et respecter les délais, nous avons décomposé le projet en différentes tâches. À cette fin, nous avons employé le diagramme de Gantt, un outil d'ordonnancement et de gestion de projet. Il permet de visualiser, à l'aide d'un graphe, dans le temps, les différentes tâches liées à un projet. Le planning anticipé est synthétisé dans le tableau ci-dessous

CHAPITRE II : EXPRESSIONS DES BESOINS

1. Description du fonctionnement attendu de l’application

Cette description du fonctionnement attendu de l'application SysGMS définit les principales fonctionnalités et exigences pour répondre aux besoins spécifiques de gestion d'une mutuelle de santé des travailleurs, en mettant l'accent sur la virtualisation et l'automatisation des processus avec Vagrant

1. Spécifications fonctionnelles

La spécification fonctionnelle est la description des fonctions d’un logiciel en vue de sa réalisation. Dans cette partie, nous décrirons dans les détails les exigences du futur système à travers l’identification des acteurs, des cas d’utilisation et les différents diagrammes.

***Adhérent***

Dans la mutuelle de santé, il existe quatre (04) types d’adhérents :

* Les travailleurs permanents de la structure mère dont relève la mutuelle,
* Les travailleurs de la structure mère mises à la disposition dans d’autres structures
* Les retraités de la structure mère.,
* Les travailleurs propres de la mutuelle.

***Bénéficiaires des prestations***

Les bénéficiaires des prestations sont classés en deux (02) catégories (l’adhérent et les personnes à charge).

L’adhérent est la personne physique remplissant les conditions d’adhésion à la mutuelle et ayant effectivement adhéré.

Les personnes à charges sont les personnes ayant un lien d’affiliation avec un adhérent (conjoint(e), enfant biologique ou d’adoption). Pour ces cas, un document d’affiliation est exigé pour prouver le lien d’affiliation. L’acte de mariage pour le cas du conjoint ou de la conjointe, un acte judiciaire pour les enfants n’ayant pas de liens directs d’affiliation, un extrait d’acte de naissance pour les enfants biologiques ayant moins de 21 ans. Pour les enfants de plus de 21 ans, en plus de l’extrait d’acte de naissance, il faut joindre obligatoirement un certificat de scolarité.

***Partenaires***

Les partenaires sont de quatre (04) ordres :

* les centres de soins,
* les pharmacies,
* les lunetteries,
* les laboratoires d’analyse.

Pour être agréer, la structure partenaire doit signer une convention de partenariat avec la mutuelle en fournissant la fiche des prestations et la grille tarifaire.

***Processus de gestion***

Le bénéficiaire de la prestation (adhérent ou personne à charge) se présente dans la structure partenaire pour bénéficier d’une prestation. Le prestataire partenaire dispose d’une fiche de Bon où il enregistre les informations de l’adhérent (Nom, Prénom(s), numéro de carte).

Pour la personne à charge, en plus des informations personnelles, on ajoute le lien d’affiliation, et les informations de l’adhérent dont il a un lien d’affiliation.

En plus de ces informations, le bon est rempli avec la liste détaillées des prestations fournies au bénéficiaire, les tarifs de ces prestations ainsi que le montant à payer par la mutuelle par type de prestation et le montant total.

A la fin de chaque mois, au plus tard le 05 du mois suivant, chaque structure partenaire établie la facture de l’ensemble de ces bons et doit les faire parvenir à la mutuelle tout en précisant le montant global à payer.

La mutuelle réceptionne les pièces, les enregistre et passe à leur traitement.

Une fois les document réceptionnés, l’Assistant en assurance les enregistre dans le système. Il est chargé de la saisie des bons, des factures, les prestataires. Outre la fonction de saisi, il peut également consulter les différentes pièces.

# Identification des acteurs

Dans le Système de Gestion d’une Mutuelle de Santé, on a principalement trois (03) types d’utilisateurs que sont :

* L’Assistant en Assurance (Assistant) saisi les bons, les factures, les prestataires et les adhérents et leurs personnes à charge. En plus de la saisie, il peut également consulter les différentes informations.
* Le chef de Service des Prestations (Chef Prestation) : il est le responsable de l’Assistant en assurance. Il peut effectuer les tâches de l’Assistant en assurance, mais également il effectue les différentes modifications des informations saisies par l’Assistant en assurance. Il enregistre également la grille tarifaire des partenaires et édite les états de paiement des partenaires.
* L’Administrateur du Système (Administrateur) : il assure les tâches d’administration du système. Il gère les utilisateurs (création, modification, suppression).

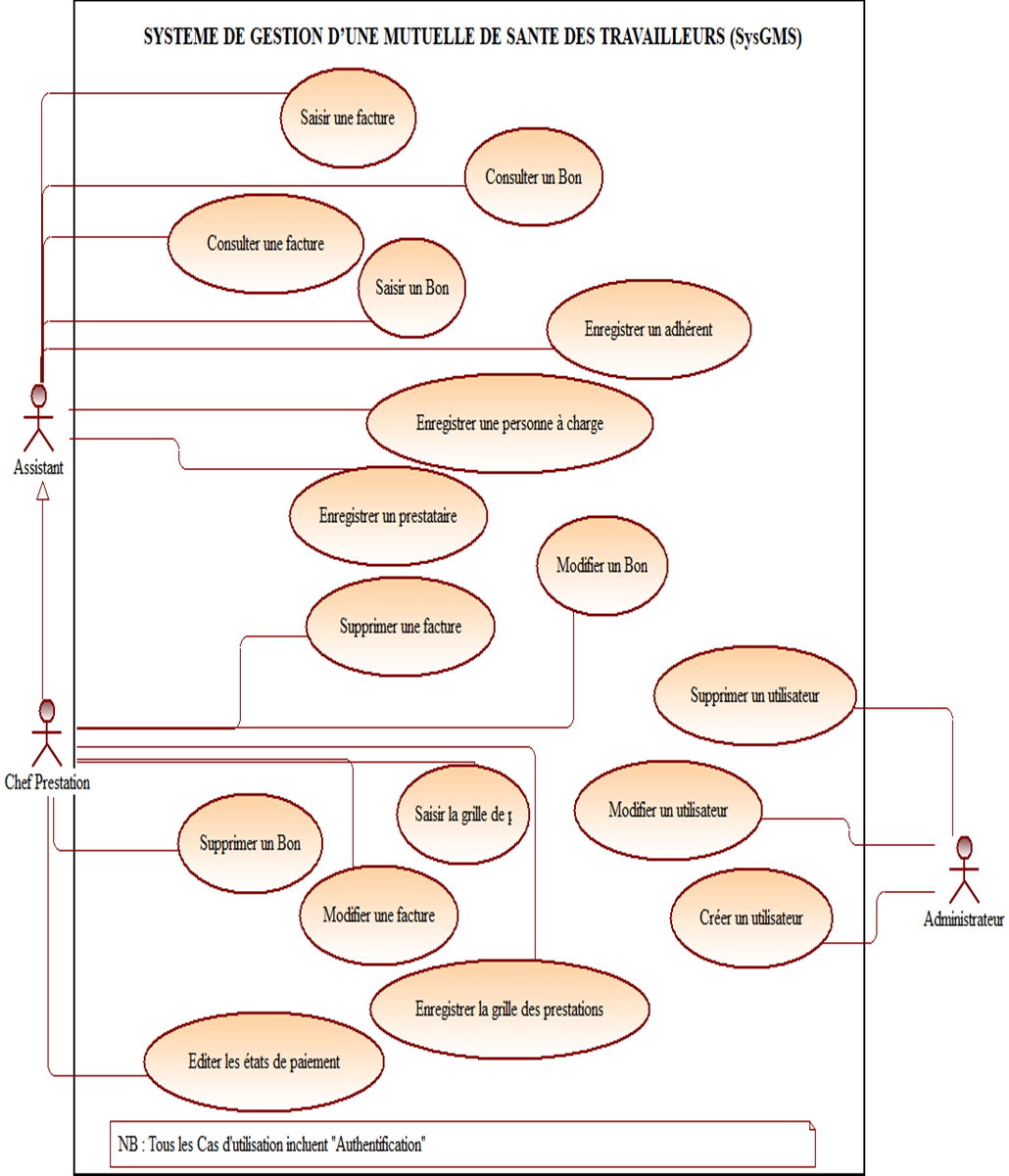
# Identification des cas d’utilisation

Pour réaliser le diagramme des cas d’utilisation, il faut d’abord identifier les différents cas d’utilisation. Le tableau ci-dessous énumère de façon non exhaustive, les principaux cas d’utilisation de notre système.

| **N°** | **Cas d’utilisation** | **Description** | **Acteur(s)** |
| --- | --- | --- | --- |
| CU01 | Créer un utilisateur | Permet la création d’un utilisateur du système | Administrateur |
| CU02 | Modifier un utilisateur | Permet de modifier le rôle ou les informations d’un utilisateur | Administrateur |
| CU03 | Supprimer un utilisateur | Permet de supprimer un utilisateur du système | Administrateur |
| CU04 | Enregistrer un adhérent | Permet d’enregistrer un adhérent | - Assistant  - Chef Prestation |
| CU05 | Enregistrer une personne à charge | Permet d’enregistrer une personne à charge d’un adhérent | - Assistant  - Chef Prestation |
| CU06 | Enregistrer un prestataire | Permet d’enregistrer une structure partenaire | - Assistant  - Chef Prestation |
| CU07 | Saisir un Bon | Permet de saisir le Bon de prise en charge dans le système | - Assistant  - Chef Prestation |
| CU08 | Modifier un Bon | Permet de modifier les informations d’un Bon | - Chef Prestation |
| CU09 | Supprimer un Bon | Permet de supprimer un Bon | - Chef Prestation |
| CU10 | Consulter un Bon | Permet de consulter un Bon | - Assistant  - Chef Prestation |
| CU11 | Saisir une facture | Permet de saisir une facture | - Assistant  - Chef Prestation |
| CU12 | Modifier une facture | Permet de modifier les informations d’une facture | - Chef Prestation |
| CU13 | Supprimer une facture | Permet de supprimer une facture | - Chef Prestation |
| CU14 | Consulter une facture | Permet de consulter une facture | - Assistant  - Chef Prestation |
| CU15 | Enregistrer la grille des prestations | Permet d’enregistre la grille tarifaire des prestations des structures partenaires | - Chef Prestation |
| CU16 | Saisir la grille de prise en charge de la mutuelle | Permet d’enregistrer la grille tarifaire de actes médicaux pris en charge par la mutuelle | - Chef Prestation |
| CU17 | Editer les états de paiement | Permet d’éditer les différents états de paiements des partenaires | - Chef Prestation |
| CU18 | Authentification | Permet de s’authentifier | Tous les utilisateurs |

# **Diagramme de cas d’utilisation**

Le diagramme de cas d'utilisation est un outil de modélisation UML qui vise à présenter une vue d'ensemble du comportement fonctionnel d'un système logiciel. Chaque cas d'utilisation représente une interaction spécifique entre un utilisateur (qu'il soit humain ou une machine) et le système. D'une part, il permet de structurer et clarifier les besoins des utilisateurs. D'autre part, il facilite l'identification des acteurs et des fonctionnalités du système, comme illustré dans la figure ci-dessous :



# **Diagramme de séquence**

Les diagrammes de séquences illustrent les échanges entre le système et les utilisateurs en exposant, sous forme de scénarios, la séquence temporelle des messages émis à partir d'un cas d'utilisation donné

1. Spécification technique
2. Matériel nécessaire

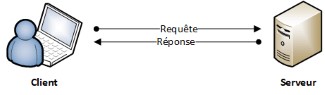
* Ordinateurs avec VirtualBox et Vagrant installés.
* Accès à Internet pour télécharger les boxes et les paquets nécessaires.
* Un dépôt Git pour chaque groupe.

1. Architecture de développement

Une architecture en informatique désigne un mode de communication à travers un réseau entre plusieurs éléments physiques et/ou logiques. Il en existe plusieurs types dont les plus répandus et performants sont du type deux (2) tiers et trois (3) tiers. Le tableau suivant est un tableau comparatif de ses deux architectures

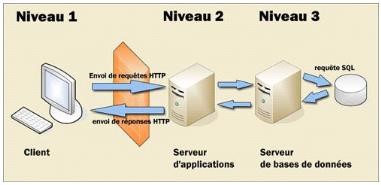
|  |  |  |
| --- | --- | --- |
| **Architecture deux (2) Tiers** | | |
| **Description** | **Avantages** | **Inconvénients** |
| Cette architecture, aussi appelée architecture client/serveur de première génération, est constituée de deux niveaux :   * **Un client** qui gère la présentation et la logique applicative ; * **Un serveur** qui stocke les données de façon cohérente et éventuellement une partie de la logique applicative.   La communication entre ces deux parties se fait en utilisant les protocoles http .  Dans ce type d’architecture, on constate une certaine indépendance du client par rapport au serveur.  Dans ce contexte, l’application devra être installée sur tous les postes clients c’est-à-dire sur toutes les machines des différents acteurs du système. Quant à la base de données, elle sera sur un serveur de données centralisé. La figure suivante représente cette architecture. | L’architecture deux (2) tiers présente les avantages suivants :   * Elle permet une exploitation multi-utilisateur de   l’application : plusieurs utilisateurs peuvent avoir accès à la même base de données ;   * Elle répartit les charges entre les machines : le   client s’occupe de l’interface graphique ainsi que les traitements sur les données tandis  que le serveur s’occupe de la recherche des données pour satisfaire aux requêtes des clients;  - Elle offre une bonne sécurité des données : la sécurité des données est assurée au niveau du système de gestion de la base de données. | * Le coût de déploiement élevé : il faut installer l’application sur tous les postes clients et les configurer * L’évolution difficile du système : à chaque mise à jour, de l’application correspond à un redéploiement sur tous les postes clients, ce qui entraîne de nouveaux coûts. * Il faut des postes clients assez performants |

|  |  |  |
| --- | --- | --- |
| **Architecture trois (3) Tiers** | | |
| **Description** | **Avantages** | **Inconvénients** |
| L’utilisateur final interagit avec l'application. Sa principale fonction est d'afficher des informations à l'attention de l'utilisateur et d'en collecter de ce dernier. Ce niveau de niveau supérieur peut s'exécuter sur un navigateur Web, en tant qu'application pour ordinateur de bureau, ou interface graphique utilisateur, par exemple.  **Le serveur d’application** ou niveau application, également appelé niveau logique ou niveau intermédiaire, est le cœur de l'application. Dans ce niveau, les informations collectées dans le niveau Présentation sont traitées, parfois par rapport à d'autres informations, dans le niveau Données, en utilisant la logique applicative,  **Le serveur de base de données ou** niveau données, parfois appelé système dorsal, est l'endroit où les informations traitées par l'application sont stockées et gérées. | Cette architecture se développe très vite grâce à ses multiples avantages que sont :  - Une bonne disponibilité et une évolution facile : les applications trois (3) tiers sont caractérisées par une facilité d’exploitation et une grande flexibilité pour l’évolution ; En effet, les mises à jour sont faites au niveau du serveur d’application, ce qui n’entraîne pas le redéploiement sur les postes clients ;  Un déploiement aisé : l’application n’est déployée que sur le serveur d’application. Les clients n’ont besoin que d’un navigateur web compatible avec l’application ;  Une sécurité accrue : ces types d’application possèdent une grande sécurité des données. En effet, l’accès à la base de données est effectué uniquement par le serveur d’application  Contrairement aux applications deux tiers où tous les utilisateurs sont connectés à la base. Il suffit donc de gérer la sécurité au niveau du serveur d’application. | Cette technologie est nouvelle et nécessite un personnel informatique initié pour sa mise en œuvre. De plus, il est recommandé d’exploiter ce type d’architecture dans un réseau haut débit.  Il faut des serveurs très performants. |



**Figure x : Schéma représentant l'architecture 2 tiers**

**Source** : <https://images.app.goo.gl/JctssrzMR6EVX4s38>



**Figure 3 : Schéma représentant l’architecture 3 tiers**

**Source** : <https://images.app.goo.gl/V7EJ1jG5Lze3wv2w8>

**Solution retenue :** A la suite de l’étude comparative que nous avons menée sur les deux (2) types d’architectures, notre choix se porte sur l’architecture trois (3) tiers. En effet, cette dernière répond mieux au besoin de la mutuelle et aux exigences de notre cahier de charge.

CHAPITRE III : CONCEPTION GLOBALE

1. Diagramme de classe
2. Dictionnaire de données

**Un dictionnaire de donnée se définit comme un référentiel de métadonnées** qui renseigne sur le contexte d’une base de données et qui fournit les informations nécessaires pour pouvoir l’interpréter. Cette documentation facilite la **gestion des bases de données** et permet aux administrateurs et utilisateurs de comprendre facilement la structure de leurs bases de données.

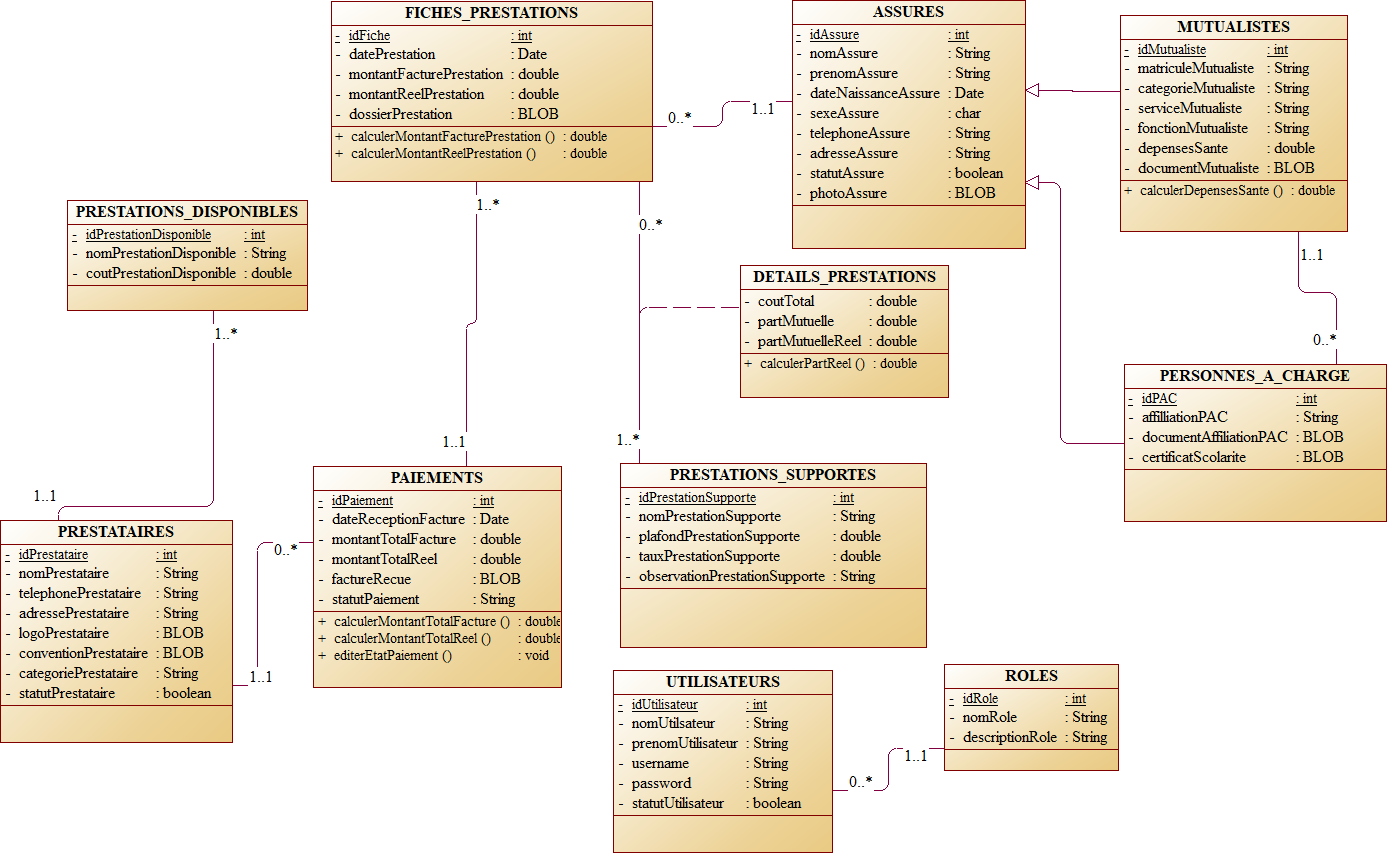
Le dictionnaire de données décrit les données qui se trouvent dans une base de données en concentrant **plusieurs types d’informations.**

**Le tableau ci-dessus représente le dictionnaire de données de la Base de Données de notre système.**

| **Nom** | **Description** | **Type** |
| --- | --- | --- |
| categoriePrestataire | Le nom de la catégorie du Prestataire | String |
| idPrestationDisponible | L’identifiant de la Prestation médicale disponible | int |
| nomPrestationDisponible | Le nom de la Prestation médicale disponible | String |
| couPrestationDisponible | Le cout de la Prestation disponible | double |
| idPrestataire | L’identifiant du prestataire | int |
| nomPrestataire | Le nom du prestataire | String |
| telephonePrestataire | Téléphone du prestataire | String |
| conventionPrestataire | Convention avec le prestataire | BLOB |
| adressePrestataire | Adresse du prestataire | String |
| logoPrestataire | Logo du prestataire | BLOB |
| statutPrestataire | Statut du prestataire | boolean |
| idPaiement | L’identifiant du payement | int |
| dateReceptionFacture | Date de réception de la facture | date |
| montantTotalFacture | Montant total de la facture | double |
| montantTotalReel | Montant total réel | double |
| factureRecus | Facture reçus | BLOG |
| statutPaiement | Statut du paiement | string |
| idFiche | L’identifiant de la fiche | int |
| datePrestation | Date de la prestation | Date |
| montantFacturePrestation | Mondant de prestation de la facture | double |
| montantReelPrestation | Montant réel de la prestation | double |
| dossierPrestation | Dossier de la prestation | BLOB |
| idPrestationsupporte | L’identifiant de l’Prestation supporte | int |
| nomPrestationSupporte | Nom de l’Prestation supporte | String |
| plafondPrestationSupporte | Le plafond de l’Prestation supporte | double |
| tauxPrestationSupporte | Taux de l’Prestation supporte | double |
| observationPrestationSupporte | Observation de l’Prestation supporte | String |
| idUtilisateur | L’identifiant de l’utilisateur | int |
| nomUtilisateur | Le nom de l’utilisateur | String |
| prenomUtilisateur | Prénom de l’utilisateur | String |
| username | Le nom d’utilisateur | String |
| passwd | Le mot de passe de l’utilisateur | String |
| statutUtilisateur | Le statut de l’utilisateur | boolean |
| idAssurance | L’identifiant de l’assurance | int |
| nomAssure | Nom de l’assure | String |
| prenomAssure | Prénom de l’assure | String |
| dateNaissanceAssure | Date de naissance de l’assuré | Date |
| sexeAssure | Sexe de l’assuré | carPrestationre |
| telephoneAssure | Le numéro de téléphone de l’assure | String |
| adresseAssure | L’adresse de l’assure | String |
| statutAssure | Statut de l’assure | boolean |
| photoAssure | Photo de l’assure | BLOB |
| coutTotal | Le cout total | double |
| categorieMutualiste | La catégorie du mutualiste | String |
| partMutuelle | La part mutuelle | double |
| partMutuelleReel | La part mutuel réel | double |
| idRole | L’identifiant du rôle | int |
| nomRole | Nom du rôle | String |
| descriptionRole | La description du rôle | String |
| idMutualiste | L’identifiant du mutualiste | int |
| matriculeMutualiste | Le matricule du mutualiste | String |
| documentMutialiste | Document justifiant la qualité du mutualiste | BLOB |
| serviceMutualiste | Le service du mutualiste | String |
| fonctionMutualiste | La fonction du mutualiste | String |
| depensesSante | La dépense santé | double |
| idPAC | L’identifiant de la personne à charge | int |
| affiliationPAC | Affiliation personne à charge | String |
| documentAffilialePAC | Document filiale personne à charge | BLOB |
| certificatScolarité | Le certificat de scolarité | BLOB |

1. Diagramme de classe

Le diagramme de classes exprime la structure statique du système en termes de classes et la relation entre ces classes. Pour satisfaire au besoin de notre client, nous utiliserons le diagramme de classes suivants :



**Figure x : Diagramme de classe**

1. Diagramme de déploiement

CHAPITRE IV : REALISATION

Plusieurs éléments importants entrent en compte dans la réalisation d’un logiciel. Ces éléments peuvent être le choix des outils de réalisation, les maquettes, les politiques de sécurité et le coût de réalisation de la solution. Nous les mettrons en lumière dans cette partie de l’étude.

1. Présentation des technologies et outils de réalisation

Les technologies et outils de réalisation sont des éléments qui aident un développeur dans le déroulement d’une activité de développement. Parmi ces éléments, nous étudierons notamment les langages de programmation utilisés et les outils de conception.

1. Langages de programmation

Un algorithme est un ensemble d’opérations à mener afin d’atteindre un résultat donné. Afin de formuler ces algorithmes, dans le domaine informatique, nous utilisons des langages de programmation.

Nous avons utilisé plusieurs langages pour le développement de notre application. Ce sont :

* **PHP**, pour Hypertext Preprocessor, est un langage de programmation sous licence libre qui peut être utilisé par n'importe qui de façon totalement gratuite. PHP s'utilise la plupart du temps côté serveur. Il génère du code HTML, CSS ou encore XHTML, des données (en PNG, [JPG,](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203259-jpeg-joint-photographic-experts-group-definition-traduction/) etc.) ou encore des fichiers PDF ;
* **TypeScript** : TypeScript est un langage de programmation open-source développé par Microsoft. Il s'agit d'un sur-ensemble de JavaScript. La principale caractéristique de TypeScript est l'ajout d'un système de typage statique en option à JavaScript, permettant aux développeurs de spécifier explicitement les types de variables, de paramètres de fonction et de retour de fonction.
* **HTML** 5 (HyperText Markup Language) : est un langage de marquage et de balisage servant à écrire des pages pour le World Wide Web. (Pour les pages web) ;
* **CSS** 3 (Cascading Style Sheet) : le CSS permet de faire la mise en forme des pages web.

1. Plateforme de développement (Framework)

Outre les langages de programmation, nous avons eu recours à des Frameworks dans le développement de notre solution. Un Framework est une infrastructure logicielle fournissant des composants structurels réutilisables facilitant la création d'applications. Les Frameworks utilisés sont :

* Angular : Angular est un framework open-source développé par Google, utilisé pour construire des applications web dynamiques et évolutives. Dans le cadre de notre projet, nous l’avons utilisé pour le Front End
* **Laravel :**nous avons utilisé Laravel en version 10, un Framework PHP open source basé sur le modèle MVC et la programmation orientée objet. Laravel permet de développer des APIs backend robustes, réutilisables par différents types d'interfaces. Dans le cadre de notre projet, nous l’avons utilisé pour le Backend ;
* **Bootstrap** 4.3.1 : nous avons utilisé Bootstrap en version 4.3.1. Bootstrap est un Framework frontend gratuit et open source pour développer des sites et applications web responsive, basé sur HTML et CSS. Il fournit des composants d'interface et des classes CSS réutilisables qui permettent de créer rapidement des maquettes et prototypes, en proposant une grille responsive, des styles prédéfinis pour boutons, tableaux, formulaires, etc. Bootstrap facilite et accélère le développement frontend.

1. Outils de conception

Les outils de conception désignent l’ensemble des outils utilisés depuis la conception jusqu’à la réalisation du projet. Ce sont :

* Spring Boot : Spring Boot est un outil open-source puissant pour développer des applications Java rapidement et efficacement en simplifiant la configuration, en intégrant un serveur embarqué, et en offrant une expérience de développement fluide avec une gestion des dépendances simplifiée.
* SYBASE PowerAMC : SYBASE PowerAMC est un outil de modélisation de données et de conception de bases de données, développé par Sybase, qui fait désormais partie de SAP
* **Git 2.39.0 :** Git est un logiciel de gestion de versions décentralisé. Il nous a permis de gérer et archiver le code source de la plateforme ;
* **Microsoft Office Word** et **PowerPoint 2019** : ce sont des outils efficaces pour le traitement des textes. Ils nous ont servi à rédiger la documentation nécessaire à la connaissance et à l’utilisation de l’application ;
* **Visual Studio Code :** nous avons utilisé Visual Studio Code comme éditeur de code.

Il s'agit d'un IDE développé par Microsoft, disponible sur Windows, Linux et MacOs. Cet éditeur open source et extensible nous a permis de développer efficacement les différents composants de notre solution.

* **Postman version 10.0.1** : nous avons utilisé Postman pour tester notre API backend. Postman est un outil très populaire pour tester et valider le fonctionnement des APIs et micro services. Cet outil nous a permis de vérifier facilement que notre API répondait comme attendu aux différentes requêtes envoyées.
* **GanttProject :** Il a servi à mettre en place notre diagramme de GANTT.

1. Système de Gestion de Base de Données (SGBD)

MySQL est un système de gestion de base de données relationnelle (SGBDR) fiable et performante pour les applications nécessitant une gestion efficace des données relationnelles.

Nous portons notre choix sur le **SGBD** **MySQL** version 5.7.24. En effet, MySQL est gratuit et open source ; il est facile à déployer et à prendre en main, contrairement aux autres SGBD.

1. Serveur d’application

Le serveur d’application contient le code métier, c'est-à-dire l’ensemble de la logique applicative (Règles de gestion, Sécurité, Couche d'accès aux données etc.). Il contient également des services qui exposent des API REST qui récupèrent les données issues du serveur de base de données, les transforment selon la logique et les exposent au client. Un serveur d’application met à la disposition des utilisateurs les différents services qu’il héberge. Ces applications peuvent être accédées via un navigateur web. Pour sa conception, nous avons utilisé Angular pour le Front End et Laravel pour le Back End.

1. Un serveur web

Notre choix se porte sur le serveur d’application PHP car PHP est un langage de script côté serveur largement utilisé et reconnu pour sa flexibilité, sa simplicité d'intégration avec les bases de données et les serveurs web, ainsi que pour sa capacité à gérer efficacement les interactions dynamiques avec les utilisateurs sur les sites web

## **Présentation de l’architecture MVC de l’application**

Notre système suit une architecture Modèle-Vue-Contrôleur (MVC) qui sépare l'application en trois composants : les données (Modèle), l'interface utilisateur (Vue) et la logique de contrôle (Contrôleur). Cette architecture divise l'interface homme-machine (IHM) en isolant la logique métier de la présentation. Le modèle gère les données, la vue s'occupe de l'affichage et le contrôleur fait le lien entre les deux et gère les actions de l'utilisateur. Cette séparation des préoccupations, formalisée en 1979 par Trygve Reenskaug, permet de construire une application robuste, maintenable et évolutive. L'architecture MVC est aujourd'hui un standard largement répandu.

* **Modèle**

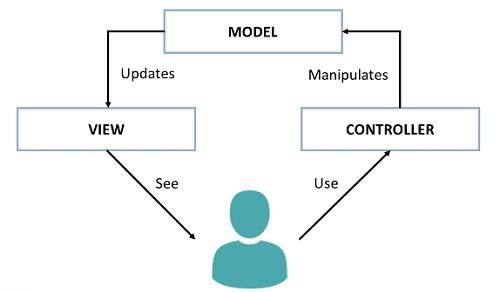
Il contient la représentation des données de l'application et les règles métiers associées. Il permet de récupérer, mettre à jour et garantir l'intégrité des informations stockées dans la base de données. Le modèle fournit ces données brutes au contrôleur, sans aucune présentation ou formatage. Son rôle est d'encapsuler la logique métier et l'accès aux données, indépendamment de l'interface utilisateur ;

* **Vue**

La vue gère l'interface utilisateur de l'application. Elle présente les données renvoyées par le modèle sous forme visuelle pour l'utilisateur, et transmet les actions de ce dernier au contrôleur. La vue n'effectue aucun traitement, elle affiche simplement le résultat des traitements du modèle. Elle est composée principalement de code HTML et JavaScript pour l'affichage, ainsi que de PHP pour les boucles et conditions d'affichage simples. Son rôle est de séparer la présentation de la logique métier, pour obtenir une interface utilisateur indépendante des données et traitements ;

* **Contrôleur**

Le contrôleur fait le lien entre le modèle et la vue. Il récupère les données auprès du modèle, les traite, prend les décisions nécessaires, et transmet les informations à afficher à la vue. Le contrôleur gère également les actions de l'utilisateur, en déclenchant les mises à jour du modèle et de la vue. Il est implémenté en PHP et contient la logique métier et de contrôle de l'application. Son rôle est de séparer la logique de l'application de la présentation et des données, pour obtenir un couplage faible entre ces différents composants.



# Figure x: Structure de l’architecture Modèle-Vue-Contrôleur

Source : <https://www.mshowto.org/mvc-yapisi-ve-ozellikleri.html>

## **Configuration des Machines Virtuelles**

# Création des Vagrantfile :

# Provisionnement automatique :

# Mise en place du réseau :

# Partage de fichiers :

## **Développement de l’application**

# Quelques maquettes IHM de notre application

## **Déploiement de l’Application**

# Scripts de provisionnement

La figure suivante représente la partie connexion de notre système. En effet, seul l’administrateur est autorisé à enregistrer un utilisateur. Tous les utilisateurs ayant un compte, peuvent se connecter à partir de ce formulaire de connexion.

## **Tests**

Les tests en développement logiciel évaluent la performance des composants et fonctionnalités pour garantir la qualité et la stabilité de notre application. Ils incluent des tests unitaires, d'intégration, bout en bout et de performance, chacun assurant le bon fonctionnement des différentes parties de l'application. Cette prise de conscience de leur importance nous motive à réaliser des tests approfondis pour garantir la fiabilité et les performances de notre application

* **Outils de test**

Nous avons choisi d'utiliser le Framework Pest pour tester notre application. Pest est un Framework de test PHP moderne, léger et expressif, spécialement conçu pour simplifier la création, l'organisation et l'exécution des tests. Sa syntaxe simple et lisible, associée à des fonctionnalités puissantes, facilite la vérification de chaque composant de notre application.

* **Mise en place des tests**

## **Politique de sécurité**

La politique de sécurité englobe toutes les mesures nécessaires visant à garantir la protection du logiciel. Assurer la sécurité de notre plateforme est une priorité essentielle. Cette politique repose sur la sensibilisation des utilisateurs, la mise en place de mesures de sécurité tout au long du processus de développement de la plateforme et la mise en œuvre de mesures spécifiques lors de sa mise en production.

# Sensibilisation des utilisateurs

Le volet humain jouant un rôle crucial dans la stratégie de sécurité, il est impératif d'offrir à nos utilisateurs des recommandations pratiques pour optimiser l'efficacité du service fourni par la plateforme. Ainsi, les clients peuvent prévenir divers incidents en adoptant les mesures suivantes :

* éviter de divulguer leurs identifiants à des tiers.
* ne pas laisser leur ordinateur sans surveillance, en veillant à l'usage qui en est fait.
* maintenir leurs antivirus à jour régulièrement.
* limiter autant que possible l'accès à des sites web non sécurisés ou dépourvus de certificat SSL.

Cela permettra de renforcer la sécurité globale de la plateforme en sensibilisant activement les utilisateurs aux bonnes pratiques de sécurité.

# Mesures de sécurité pour la phase le développement

L'accès au système est sécurisé par l'usage d'un nom d'utilisateur et d'un mot de passe. De plus, les fonctionnalités qu'un utilisateur peut exécuter sont déterminées par son profil. Pour garantir la sécurité de l'accès aux différentes ressources de la plateforme durant la phase de développement, un mécanisme d'authentification basé sur le protocole OAuth 2.0 a été instauré, utilisant openLDAP comme serveur d'authentification. Un système de filtrage détecte et signale automatiquement les données invalides lors de la soumission, invitant l'utilisateur à les corriger avant une nouvelle soumission, anticipant ainsi les erreurs de saisie.

# Le certificat SSL

Le SSL (Secure Socket Layer) est un protocole essentiel qui permet de chiffrer les échanges entre un navigateur et un serveur web. Un site web sécurisé par un certificat SSL se distingue par son URL commençant par HTTPS (HyperText Transfer Protocol Secure) et une icône de cadenas dans la barre d'adresse du navigateur. Son intégration dans notre système présente plusieurs avantages : la plateforme garde les données sensibles cryptées, ne pouvant être comprises que par le destinataire. Cela est particulièrement crucial lors de l'utilisation de points d'accès Wi-Fi publics, où des tiers malveillants ne peuvent pas intercepter les informations transmises. Dans notre système actuel, la sécurité SSL est mise en place aussi bien pour le serveur d'hébergement que pour l'accès à l'API. Ainsi, tout protocole HTTP est automatiquement redirigé vers le protocole HTTPS sécurisé.

# Politique de sauvegarde

La politique de sauvegarde vise à garantir l'intégrité des données en cas de dysfonctionnement du système. Nous préconisons les mesures suivantes pour une gestion efficace :

* des sauvegardes journalières conservées pendant une semaine ;
* des sauvegardes hebdomadaires conservées pendant un mois ;
* des sauvegardes mensuelles conservées sur une période de six (06) mois ;
* des sauvegardes semestrielles conservées pour une durée d'un an ;
* des sauvegardes annuelles archivées de manière permanente.

# CONCLUSION GENERALE

Ce projet de virtualisation avec Vagrant visant à déployer un système de gestion d’une mutuelle de santé des travailleurs (SysGMS) a permis d'atteindre avec succès les objectifs fixés. À travers l'utilisation de Vagrant, nous avons acquis une compréhension approfondie des technologies de virtualisation et de l'infrastructure. La conception, le déploiement et la gestion d'une application web multi-tiers ont été réalisés de manière efficace, en mettant en œuvre des composants essentiels tels que MySQL pour la base de données, Django pour le serveur d'application, et Apache pour le serveur web.

Ce projet a non seulement renforcé les compétences techniques des étudiants en matière de configuration et d'automatisation des déploiements, mais il a également souligné l'importance de la planification minutieuse, de la collaboration d'équipe et de la gestion des ressources. En intégrant les principes de sécurité et de performance, l'application SysGMS est prête à répondre aux besoins complexes de gestion des mutuelles de santé, tout en garantissant la confidentialité des données et la disponibilité des services pour les utilisateurs.

.