1. **Introduction :**

Dans ce premier chapitre, nous allons commencer par la mise en contexte de ce projet de fin d’étude qui est « Développement et mise en place du module maintenance d’une solution GMAO », puis nous allons présenter l’entreprise d’accueil « Diva Software » au sein duquel on a effectué notre stage de 4 mois .

1. **Présentation de l’organisme d’accueil:**

Diva Software est spécialisée dans le développement informatique de systèmes ERP, notamment pour les entreprises de confection. Elle s’appuie sur le potentiel de son équipe technique et commerciale pluridisciplinaire composée de talents issus d’écoles d’ingénierie tunisiennes. Diva Software a réussi à réaliser des produits innovants à haute valeur.

Elle s'est développée et a conquis de grands marchés internationaux, notamment en France et au Maroc

## **Informations de Contact de Diva Software :**

* **Adresse :**Boulevard Yasser Arafat Immeuble N° 38 - Bureau 44 Sahloul - 4054 Sousse
* **Propriétaire** : Mr Rochdi Jdidi
* **Administration :**(+216) 53 333 850  
   **Commercial :**(+216) 53 333 853
* **E-mail :** contact@diva-software.com
* **Site web :**[https://www.diva-software.com](https://www.diva-software.com/)



1. **Activités :**

Diva Software intervient dans la conception, le développement et le déploiement des systèmes d’information, de la fabrication à la livraison.

Elle gère l’ensemble des processus d’une entreprise en intégrant toutes ses fonctions, y compris la gestion des ressources humaines, l’aide à la décision, la vente et l’approvisionnement. Grâce à une compréhension approfondie et une interaction étroite avec le client, Diva Software s'engage à long terme. En effet, cela est dû à sa réputation en tant que partenaire de choix et à son engagement absolu pour la réussite et la commercialisation de l'excellente exécution de ses projets

1. **Service de Diva software :**

Diva Software fournit différents services pour optimiser le cycle de production et améliorer les performances de l’entreprise, en mettant à disposition divers modules :

* 1. **La gestion des stocks :**

Ce module résout les problématiques liées à la gestion des stocks, des magasins et des articles. Il permet de définir l'état des stocks et de conserver un historique des mouvements de stocks dans les différents magasins et entrepôts.

De plus, il autorise les entrées et sorties de stocks de manière manuelle ou automatique

* 1. **La gestion de la production :**

Ce module permet une gestion avancée des données techniques afin de préparer rapidement les dossiers de fabrication. Il offre une vision complète de la capacité de production (personnel, machines, matières premières, etc.). De plus, il permet de calculer les besoins nets (CBN) pour contrôler et optimiser les stocks en maintenant le lien entre les commandes clients et le lancement en fabrication associé aux commandes fournisseurs

* 1. **La gestion de la qualité :**

Ce module permet de collecter les informations de manière manuelle ou automatique, puis de les présenter sous forme d'indicateurs de performance de production. Cela garantit une amélioration continue du processus de fabrication.

* 1. **La gestion des expéditions :**

Ce module permet de gérer de manière simple les expéditions tout en générant les listes de colisage. Ces listes peuvent être automatiquement traduites en bons de livraison, qui serviront ensuite pour la facturation.

* 1. **La facturation :**

Ce module permet l’édition des factures.

* 1. **La gestion de la maintenance :**

Ce module est conçu pour gérer les différentes tâches de maintenance des équipements et assister quotidiennement les services de maintenance dans l'accomplissement de leurs missions.

1. **Organigramme de l'entreprise :**

L’entreprise est composée de 20 employés au total pour une gestion plus efficace et dynamique des principaux métiers de la société ; l'organigramme adopté est présenté dans la figure 2.



1. **Présentation du projet :**
   1. **Qu'est que c'est la GMAO:**

La GMAO, abréviation de Gestion de Maintenance Assistée par Ordinateur, est un système informatique conçu pour faciliter la gestion et l'optimisation des opérations de maintenance des équipements et des installations au sein d'une organisation.

* 1. **Contexte :**

La maintenance industrielle joue un rôle crucial dans la durabilité et la performance des entreprises. Dans ce contexte, la Gestion de Maintenance Assistée par Ordinateur (GMAO) émerge comme un outil stratégique permettant d'optimiser la gestion des équipements, de planifier les interventions, et d'améliorer la disponibilité des ressources. Ce projet de fin d'études vise à concevoir et développer un logiciel de GMAO sur mesure pour répondre aux besoins spécifiques d'entreprises opérant dans le secteur de textile

* 1. **Objectif :**

•Optimisation de la gestion des équipements

•Planification des interventions

•Amélioration de la disponibilité des ressources

•Personnalisation pour les besoins spécifiques du secteur textile

•Intégration de fonctionnalités stratégiques

•Interface utilisateur conviviale

•Sécurité et fiabilité

1. **Étude et analyse de l’existant :** 
   1. **Description de l’existant :**

Dans le secteur textile, la gestion de la maintenance des équipements repose principalement sur des processus manuels et des outils traditionnels tels que les feuilles de calcul. Ces méthodes présentent toutefois des limites en termes de traçabilité des interventions, de gestion des plannings de maintenance, et de coordination des ressources.

Les entreprises font face à des défis tels que des temps d'arrêt non planifiés, des difficultés à suivre les historiques de maintenance, et une gestion fragmentée des données. Malgré les efforts des équipes de maintenance, il existe des opportunités d'amélioration en termes de réduction des coûts, d'optimisation des plannings, et d'augmentation de la disponibilité des équipements.

L'adoption d'une solution moderne de GMAO sur mesure s'avère ainsi essentielle pour centraliser et structurer les données de maintenance, automatiser les processus, optimiser les interventions, et améliorer la performance globale de la gestion de maintenance dans le secteur textile.

* 1. **Critique de l’Existant :**

Le système de gestion de la maintenance actuel est plein de défauts et représente un grand obstacle au développement, concurrence et au gain des profits et il ne peut pas résister face aux challenges et défis. Parmi les défauts que nous trouvons dans ce système, nous citons :

* Utilisation des papiers pour gérer les activités de la maintenance.
* Absence de traçabilité et suivi des interventions.
* Mauvaise gestion des pièces de rechanges et manque de contrôle et suivie de la magasin PDR
* Absence d’un système d’informations qui centralise les données et facilite le partage et la collaboration entre les différents acteurs.
* Mauvaise communication interne.
* Absence et manque d’un inventaire complet et à jour des équipements à maintenir et des pièces de rechange.
* Les équipements et les actions correctives et préventives ne sont pas codifiés.
* Les documents techniques, administratives ou financiers liés aux équipements ne sont pas centralisés. — Absence d’un tableau de bord de suivi des KPI (fiabilité, maintenabilité, la disponibilité).
* Absence des rapports et statistiques et des outils d’aide de bonne prise de décision.
* Mauvaise gestion de la maintenance préventive accompagnée d’une mauvaise planification et ordonnancement.
* Inexistence d’un système de notification ou alerte lors de la gestion des activités de maintenance ou des pièces de rechanges

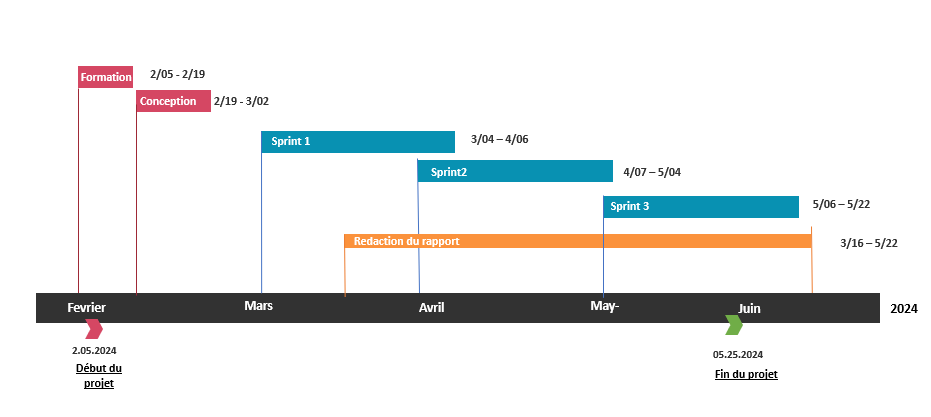
1. **Solution proposée :**

De plus, en développant sa propre solution de GMAO sur mesure, Diva Software pourra personnaliser un outil de gestion de maintenance en fonction de ses besoins spécifiques. Elle pourra ajouter des fonctionnalités sur mesure pour répondre aux exigences particulières de son secteur d'activité et des processus de suivi de maintenance spécifiques à la société. Cette flexibilité accrue permettra à Diva Software d'adapter l'outil à ses spécificités de processus de gestion de maintenance et d'améliorer l'expérience utilisateur de son équipe. L’affectation des tâches aux différents acteurs, admin, techniciens et responsables de maintenance, est assurée de manière centralisée par l’administrateur. Le suivi des états des interventions de maintenance est plus adapté aux besoins de la société. La classification des équipements est assurée pour chaque projet ainsi que leurs détails de réalisation comme la date de début, de fin, les membres de l’équipe, le budget, etc. Des recherches personnalisées sur un équipement donné peuvent fournir des informations détaillées sur ses interventions avec la société. Le suivi des tâches est assuré selon leurs états : Tâches en cours, terminées, à planifier et en retard.

1. **Planification et conduite du projet :**

La planification du projet est un outil indispensable pour bien gérer les différentes phases du projet en respectant les délais elle nous permet de :

* Fixer les objectives.
* Définir et organiser les ressources et moyennes.
* Avoir un suivi des travaux et leurs états d’avancement.
* Eliminer les risques de retard.
* Ponctualité, respect des dates limites et une bonne gestion du temps. Dans ce cadre, nous avons utilisé le fameux Diagrammes de GANTT pour établir le planning de mon stage.
* Ce diagramme est un outil qui permet de bien planifier le projet et avoir un suivi d’avancement en présentant l’enchainement et la durée des différentes tâches.



1. **Conclusion :**

Dans ce chapitre, nous avons présenté le cadre général de projet. Nous avons fait l’étude et l’analyse de l’existant pour pouvoir proposer une solution sous forme d’une application web qui couvre les limites des projets similaires. Dans le deuxième chapitre, nous allons faire la spécification des besoins fonctionnels et non fonctionnels ainsi que le Framework agile suivi.

**Chapitre2 :** **Spécification des besoins et pilotage du projet avec Scrum**

1. **Introduction :**
2. **Identification des acteurs :**

Un acteur dans une application est une entité externe qui interagit avec le système [1].

Les trois types d’acteurs de l’Application de gestion de maintenance assisté par un ordinateur que nous proposons sont les suivants :

1. **Administrateur :** C’est l’acteur principale dans notre application, il a le droit d’accéder à tous les menus de l’application.
2. **Technicien** : C’est l’acteur secondaire dans notre application, il a le droit d'accéder à tous les menus de l’application sauf celui de gestion des utilisateurs et consultation du profil utilisateur.
3. **Chef** **d’équipe** : C’est l’acteur secondaire dans notre application, il a le droit d'accéder à tous les menus de l’application sauf celui de gestion des utilisateurs et consultation du profil utilisateur, il a le droit de consulter les interventions, gérer l’utilisation de stock.
4. **Spécification des besoins :**
5. **Spécification des besoins fonctionnels :**

Il s'agit des fonctionnalités du système. Ce sont les besoins spécifiant un comportement d'entrée / sortie du Système. Le système à concevoir doit permettre à l’utilisateur d’effectuer les opérations suivantes :

* **Gestion des utilisateurs** :

il s'agit d'un outil permettant d'effectuer les opérations de gestion telles que l'ajout, la suppression, la modification et la consultation des informations caractérisant chacun des utilisateurs.

* **Gestion des travaux :**

l'application permet :

- La gestion des ordres d'intervention.

- La gestion de la maintenance préventive.

- La gestion des contrats de sous-traitance.

* **Consultation des statistiques des équipements** :

L'application fournit une vue permettant à un utilisateur de consulter les statistiques des coûts de maintenance.

* **Consultation de l'historique des agents :**

L'application permet au responsable de voir les activités réalisées dans le système, ordonnées par date.

* **Gestion de la documentation technique** :

Ce module est l'équivalent d'un gestionnaire bibliographique grâce auquel il est possible de faire la saisie et la recherche d’informations

* **Gestion des achats :**

L’application permet :

- la gestion du bon de commande

- la gestion du bon de sortie

- la gestion du bon de réception

- la gestion du bon de livraison

* **Gestion des stocks :**

Ce module doit contenir tous les outils nécessaires pour une gestion rigoureuse du magasin :

- La saisie des informations sur les produits.

- La vérification/ mise à jour de l'inventaire physique.

- La gestion des bons de commande.

- La gestion des fournisseurs et des sous-traitants.

- La gestion des entrées et des sorties.

1. **Spécification des besoins non fonctionnels :**

Les besoins non fonctionnels concernent les contraintes à prendre en considération pour mettre en place une solution adéquate aux attentes des concepteurs des architectures dynamiques. Notre application doit nécessairement assurer ces besoins :

* **L'extensibilité :** dans le cadre de ce travail, l'application devra être extensible, c'est-à-dire qu'il devra être possible d'ajouter ou de modifier de nouvelles fonctionnalités.
* **La sécurité :** l’application devra être hautement sécurisée, les informations ne devront pas être accessibles à tout le monde. Le site web sera accessible uniquement par un identifiant et un mot de passe attribué à une personne physique.
* **L'interface :** avoir une application qui respecte les principes des Interfaces Homme/Machine (IHM) tels que l'ergonomie et la fiabilité.
* **La performance :** l’application doit être performante, c'est-à-dire que le système doit réagir rapidement, quelle que soit l’action de l’utilisateur.
* **La convivialité :** l’application doit être simple et facile à manipuler, même pour des utilisateurs non experts.
* **L'ergonomie :** le thème adopté par l’application doit être inspiré des couleurs et du logo typique de l’entreprise d'accueil.
* En intégrant ces besoins non fonctionnels dans la conception et le développement de votre logiciel de GMAO, vous pouvez garantir sa qualité globale et son succès dans l'amélioration des processus de maintenance industrielle.

1. **Architecture :**

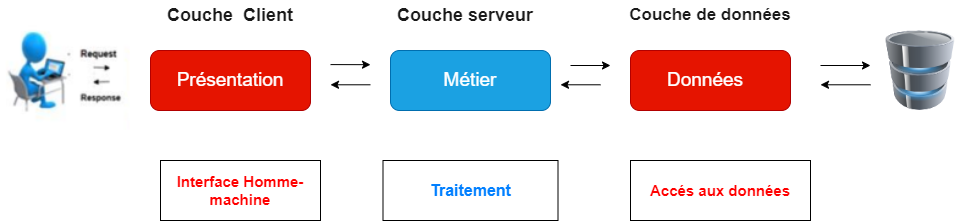
Avant de commencer à concevoir et à développer un système informatisé, il est essentiel de planifier son architecture.

* 1. **Architecture fonctionnelle :**



* 1. **Architecture physique :**

Notre application utilise une architecture en 3 tiers, une approche qui sépare clairement le système en trois couches distinctes. Ces trois couches sont représentées dans la figure 4 :



Il s’agit d’une architecture 3-tiers composé par 3 principaux couches :

**✓ Couche Client :** Elle représente l'interface utilisateur avec laquelle les utilisateurs interagissent directement.

**✓ Couche Web :** Elle fait référence à la partie de l'application qui traite les requêtes HTTP et génère des réponses pour les clients.

**✓ Couche de données :** Elle représente le serveur de base de données, fournisseur des données au serveur web.

* 1. **Architecture logicielle :**

L’architecture d’un logiciel décrit la manière dont seront agencés les différents éléments d’une application et comment ils interagissent entre eux. Cette étape est donc l’une des premières étapes du développement logiciel et intervient lors de la phase de conception[2].

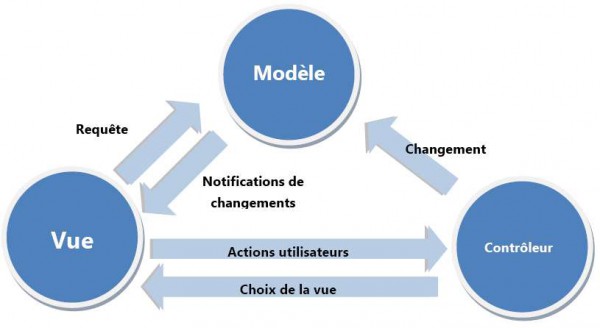
Notre système est conçu en utilisant l'architecture « MVC » pour la partie Back-end, « MVC » pour le Front-end, et « SOA » comme architecture globale de l'application.

* Choix de l’architecture côté back-end (côte serveur) :

Le choix de l'architecture côté serveur, ou back-end, dépend de plusieurs facteurs tels que les besoins fonctionnels de l'application, la scalabilité, la facilité de maintenance et la préférence des développeurs.

* **Architecture MVC (Modèle-Vue-Contrôleur)** :
* **Modèle :** cette partie gère ce qu'on appelle la **logique métier** de votre site. Elle comprend notamment la gestion des données qui sont stockées, mais aussi tout le code qui prend des décisions autour de ces données. Son objectif est de fournir une interface d'action la plus simple possible au contrôleur. On y trouve donc entre autres des algorithmes complexes et des requêtes SQL.
* **Vue** : cette partie se concentre sur l'**affichage**. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.
* **Contrôleur** : cette partie gère les **échanges** avec l'utilisateur. C'est en quelque sorte l'intermédiaire entre l'utilisateur, le modèle et la vue. Le contrôleur va recevoir des requêtes de l'utilisateur. Pour chacune, il va demander au modèle d'effectuer certaines actions (lire des articles de blog depuis une base de données, supprimer un commentaire) et de lui renvoyer les résultats (la liste des articles, si la suppression est réussie). Puis il va *adapter* ce résultat et le donner à la vue. Enfin, il va renvoyer la nouvelle page HTML, générée par la vue, à l’utilisateur. [3]

La figure suivante schématise le rôle de chacun de ces éléments :



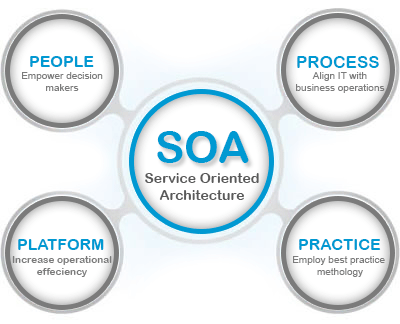
* 1. **L’architecture SOA :**

Notre application est basée sur l’architecture orientée services (SOA) qui est une approche utilisée pour créer une architecture basée sur l’utilisation d’un ensemble des services.

Ces services sont conçus pour effectuer des tâches spécifiques et peuvent être appelés par d'autres composants de l'application ou par des applications externes via des protocoles de communication standardisés tels que HTTP, SOAP (Simple Object Access Protocol), ou REST (Representational State Transfer).

**SOA visant à :**

* La décomposition d'une fonctionnalité en un ensemble de services.
* L'optimisation des performances des organisations.
* La diminution des coûts.
* L'amélioration de la souplesse des processus.



* 1. **Architecture de sécurité :**
* **L’intérêt de développer d’une API pour un progiciel :**

L'API peut être décrite comme une solution informatique essentielle qui permet à différentes applications de communiquer entre elles de manière fluide, en échangeant des services ou des données. En réalité, elle se compose d'un ensemble de fonctions accessibles via des requêtes HTTP, facilitant ainsi l'accès aux services d'une application à travers un langage de programmation défini.

En simplifiant la communication entre deux programmes informatiques, l'API offre un moyen efficace de favoriser leur évolution tout en renforçant la distinction entre le Back-end et le Front-end. Cela simplifie également le développement d'applications web en permettant aux développeurs de créer une "surcouche" personnalisée, par exemple pour un progiciel commercial, ou en laissant aux membres la possibilité de coder des fonctionnalités supplémentaires selon leurs besoins spécifiques.

En somme, l'API joue un rôle central dans l'architecture logicielle moderne, favorisant la flexibilité, la modularité et l'efficacité des systèmes informatiques interconnectés.

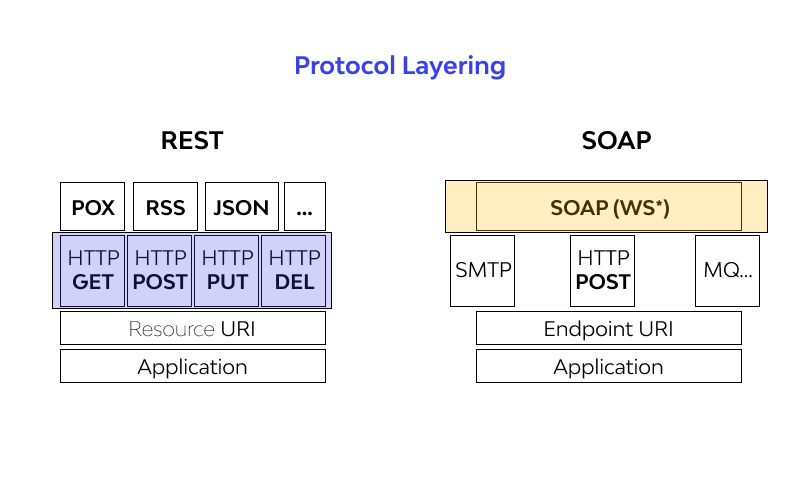
**▪ Les diverses structures d'API :**

Il existe deux principaux protocoles de communication sur lesquels reposent les API : le Simple Object Access Protocol (SOAP) et le Representational State Transfer (REST).

Le REST s'est progressivement imposé comme le choix privilégié par rapport au SOAP en raison de sa flexibilité accrue. Cette évolution a conduit à l'émergence des API REST ou RESTful.

La distinction fondamentale entre ces deux protocoles réside dans le niveau de couplage entre le client et le serveur.

La figure suivante montre la différence entre les 2 services :



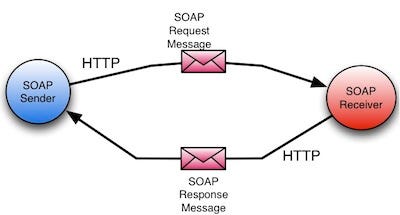
* **Modèle d'API basé sur SOAP :**

L'architecture d'API SOAP utilise le protocole SOAP (Simple Object Access Protocol) basé sur XML.

Les services sont décrits via des fichiers WSDL, assurant un contrôle élevé sur la sécurité et la fiabilité des communications, ce qui la rend adaptée aux environnements critiques.

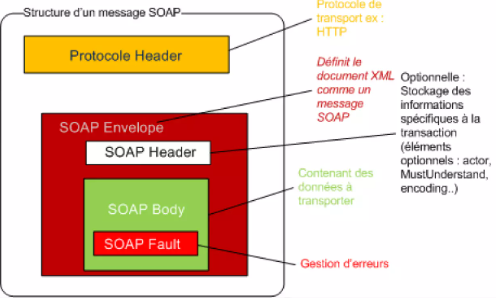
Cependant, cette architecture est souvent critiquée pour sa complexité et sa lourdeur par rapport à d'autres approches plus légères comme REST, ce qui limite son adoption dans certains cas d'utilisation.

La figure suivante présente la mode de communication :



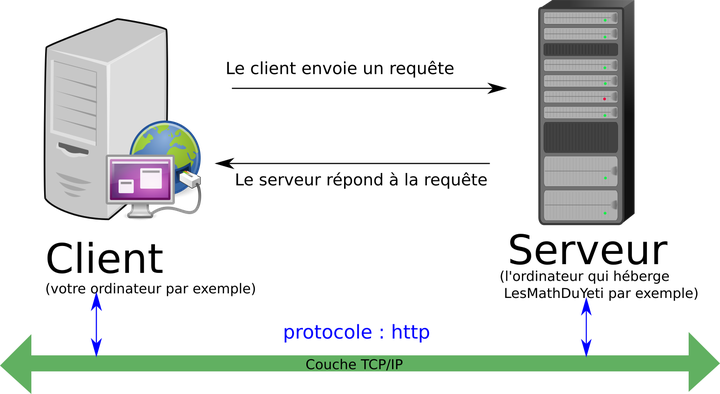
* **Structure d’un message SOAP :**

La figure suivante montre la structure d’un message SOAP :



* **Architecture d’API REST :**

L'architecture REST est un style d'architecture qui offre une approche plus légère que SOAP. Contrairement à SOAP qui utilise XML pour les requêtes, REST se base souvent sur des URLs simples. Bien qu'il puisse nécessiter des informations supplémentaires dans certains cas, la plupart des services web REST se concentrent sur la récupération des données nécessaires via des URLs spécifiques. Cette simplicité et cette approche centrée sur les ressources en font un choix attrayant pour de nombreux cas d'utilisation, offrant une communication efficace entre les clients et les serveurs avec une architecture flexible et évolutive.



* **L'architecture d'API utilisée :**

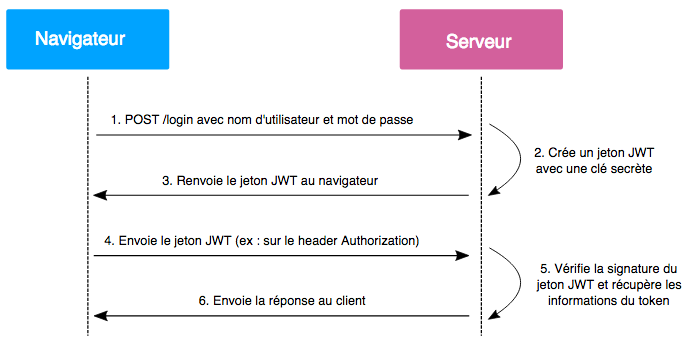
Nous avons choisi d'utiliser une architecture d'API REST avec Laravel pour notre backend pour plusieurs raisons clés. REST offre une approche légère et flexible, en accord avec la philosophie de développement rapide de Laravel. Elle est polyvalente pour de nombreux types d'échanges d'informations entre logiciels, utilisant uniquement le protocole HTTP. Cette API REST spéciale a été développée avec Laravel, permettant aux utilisateurs représentés par d'autres applications d'appeler notre API RESTful et d'obtenir des résultats. Cependant, étant donné la sensibilité des données échangées, cette API ne peut pas être publique sans sécurisation. Pour garantir la sécurité de notre API, nous avons mis en place une technique utilisant Laravel Passeport pour l'authentification et la gestion des jetons JWT (JSON Web Tokens). Cette approche assure un accès sécurisé à notre API REST de manière stateless, grâce à la gestion des autorisations par Laravel Passeport et l'utilisation des JWT pour l'authentification et l'autorisation des requêtes.

**▪ Principe de JWT :**

Le principe de JWT (JSON Web Token) repose sur l'utilisation de jetons numériques pour l'authentification et l'autorisation des requêtes dans les applications web et les services API.

JWT est couramment utilisé pour authentifier les utilisateurs après une connexion réussie, permettant ainsi à chaque requête d'être accompagnée du jeton pour prouver l'identité de l'utilisateur. De plus, il peut servir à l'autorisation en incluant des informations sur les rôles ou les autorisations spécifiques dans la charge utile du jeton.

* En résumé, JWT offre une solution sécurisée et efficace pour gérer l'authentification et l'autorisation dans les applications web modernes.

****

**▪ Structure de JWT :**

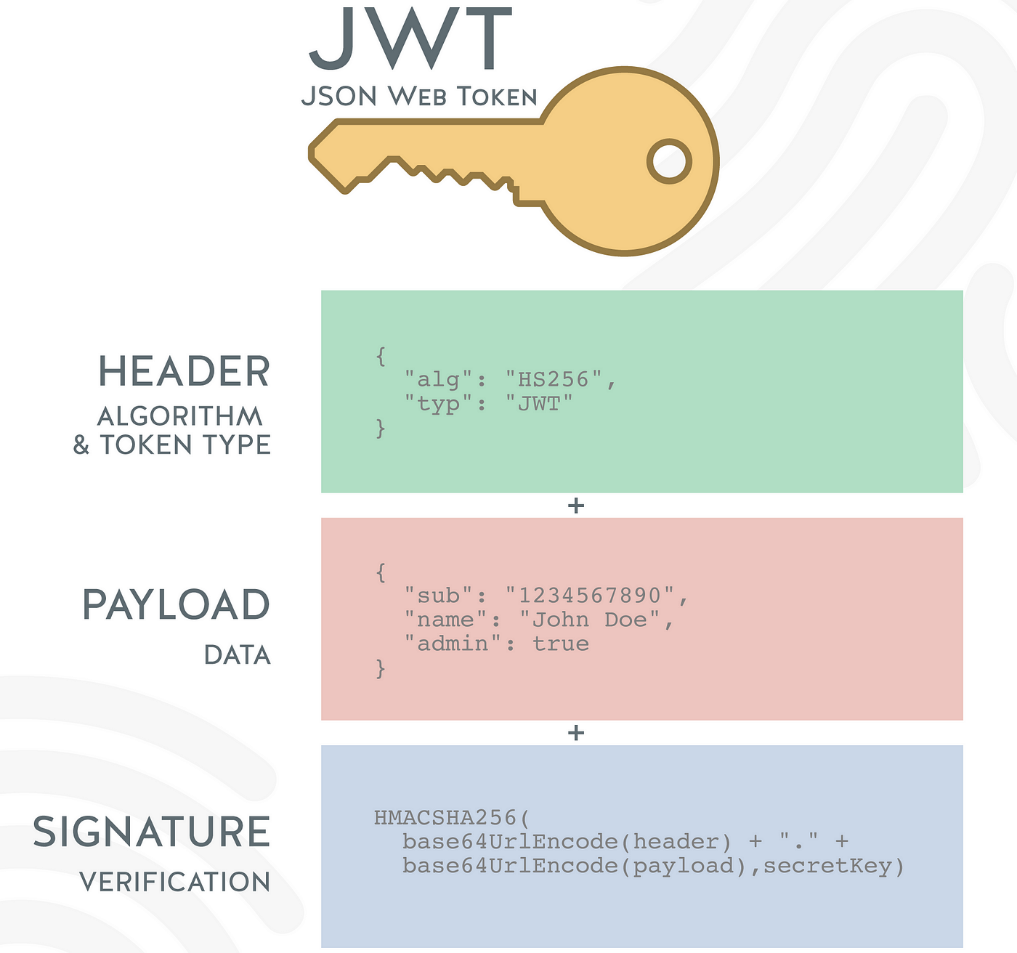
Un JSON Web Token (JWT) est composé de trois parties distinctes qui en définissent la structure.

Tout d'abord, l'en-tête (**header**) contient des métadonnées telles que le type de token (typ) et l'algorithme de signature utilisé (alg).

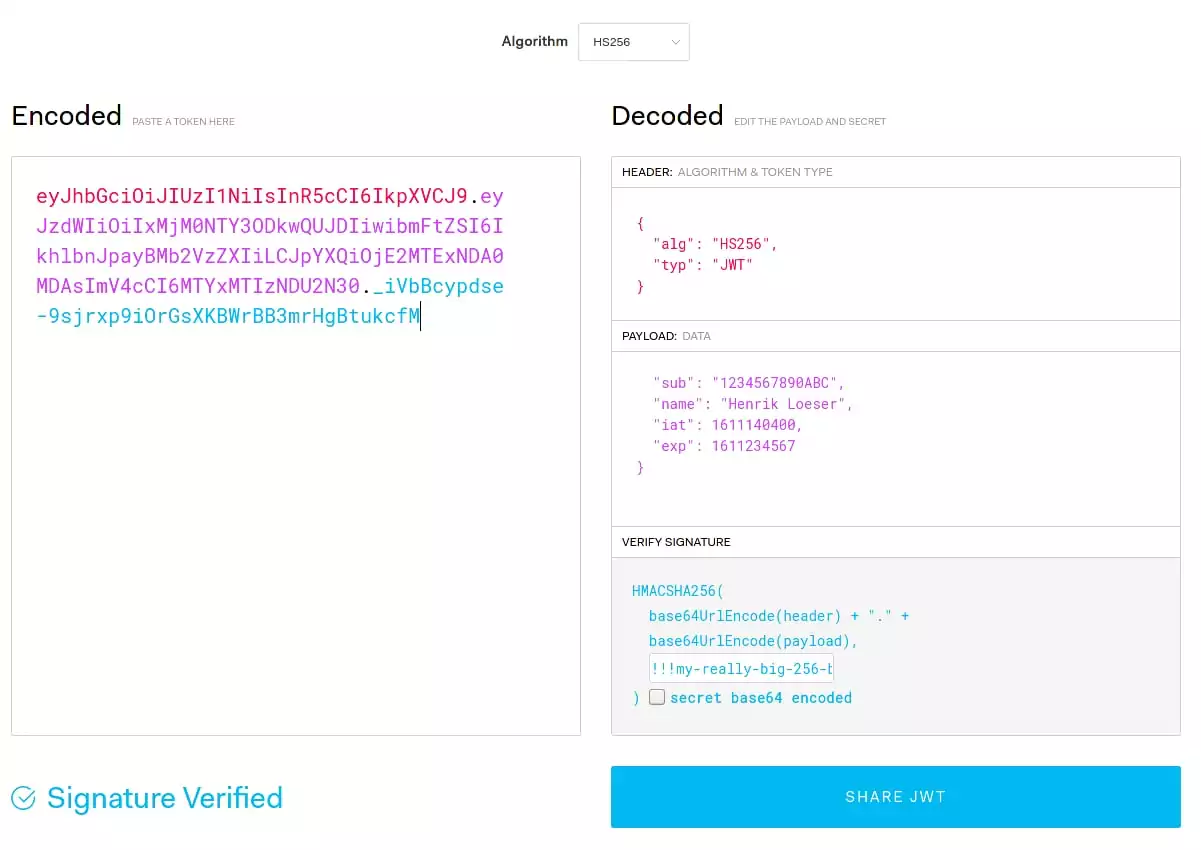
Ensuite, la charge utile (**payload**) transporte les données d'authentification ou d'autorisation, telles que l'identifiant de l'utilisateur, les rôles ou les autorisations.

Enfin, la **signature** garantit l'intégrité du token en combinant de manière sécurisée l'en-tête et la charge utile avec une clé secrète, générant ainsi un hash unique.

La structure complète du JWT se présente sous la forme suivante : **header.payload.signature.**



* La figure suivante montre un JWT qui a le Header et le Payload précédents et il est signé avec un secret :



1. **Choix de la méthodologie et formalisme adoptés :**

Pour assurer un développement performant en termes de productivité et de qualité, le choix de la méthodologie de développement est crucial. Avec la complexité croissante des systèmes, le génie logiciel propose des démarches structurées et des étapes précises. Ces méthodologies fournissent un cadre méthodique pour planifier, exécuter et évaluer les projets, améliorant ainsi l'efficacité des équipes et la qualité des produits logiciels livrés.

* 1. **Choix de la méthodologie :**

Le choix de la méthode de développement dépend du type de projet. Pour des projets avec des exigences claires dès le départ, comme ceux dans des domaines bien compris, la méthode en cascade peut suffire. En revanche, pour des projets où les besoins évoluent ou sont incomplets, comme souvent dans les projets actuels, les méthodes agiles, plus flexibles et itératives, sont recommandées. Cela permet d'ajuster le travail au fur et à mesure, en collaboration avec les parties prenantes, pour des résultats adaptés et de qualité.

Une méthode Agile garantit une meilleure qualité de communication avec l’utilisateur, une meilleure visibilité du client sur l’avancement des travaux, un meilleur contrôle de qualité par le fait que les tests sont effectués en continu, ce qui permet de détecter rapidement les problèmes. Elle intègre aussi la notion de travail en équipe [6].

Parmi les méthodes Agiles, nous utilisons « **SCRUM** » pour la réalisation de notre projet.

En effet, nous travaillons en équipe et le produit est livré à chaque terminaison d’une tâche. En plus le client est toujours intervenant dans le test de la solution.

* Pour la gestion de la maintenance assistée par ordinateur (GMAO), nous avons opté pour la méthodologie agile Scrum. Cette approche nous permet de travailler en équipe de manière collaborative, avec des livraisons fréquentes à chaque étape de la réalisation des tâches. De plus, nous impliquons activement l‘admin dans le processus de test de la solution, assurant ainsi une adaptation continue aux besoins et une qualité optimale du produit final.
  1. **Définition du Scrum :**

Son principe est d’accuser et de donner le choix d’organisation à

une équipe qui est livrée d’élaborer un produit utilisable lors d’itérations dans une durée bien déterminée. Scrum se base sur un petit jeu de pratiques et quelques rôles. Cette méthode peut être utilisée dans n’importe quelle condition ou un groupe de personnes ont besoin de travailler

ensemble pour atteindre un but commun.[7]

* 1. **Cycle de vie de la méthode Scrum :**

Elle est composée d’un Scrum Master, d’un Product Owner et d’équipe de développement.

* **Product Owner (PO) :** Il est l'un des acteurs clés du processus scrum.

Son rôle est de faire le pont entre la partie métier et la partie technique du projet. Il est donc le relai entre le client et l’équipe de développement.

Le Product Owner porte la vision du produit scrum.

Il est responsable de la rédaction des user stories et chargé de maintenir le Product Backlog à jour.[8]

* **Équipe Scrum (l’équipe de développement) :** cette équipe se compose de trois à dix développeurs. Elle est responsable de la mise en œuvre des solutions techniques et de la réalisation des tâches de développement [8].
* **Scrum master (SM) :** C’est un facilitateur et un coach plus qu’un superviseur.

Il a comme rôle d’aider l’équipe à mettre un cadre Scrum adapté au contexte de l’équipe.

Il protège l’équipe des éléments perturbateurs, ainsi que des

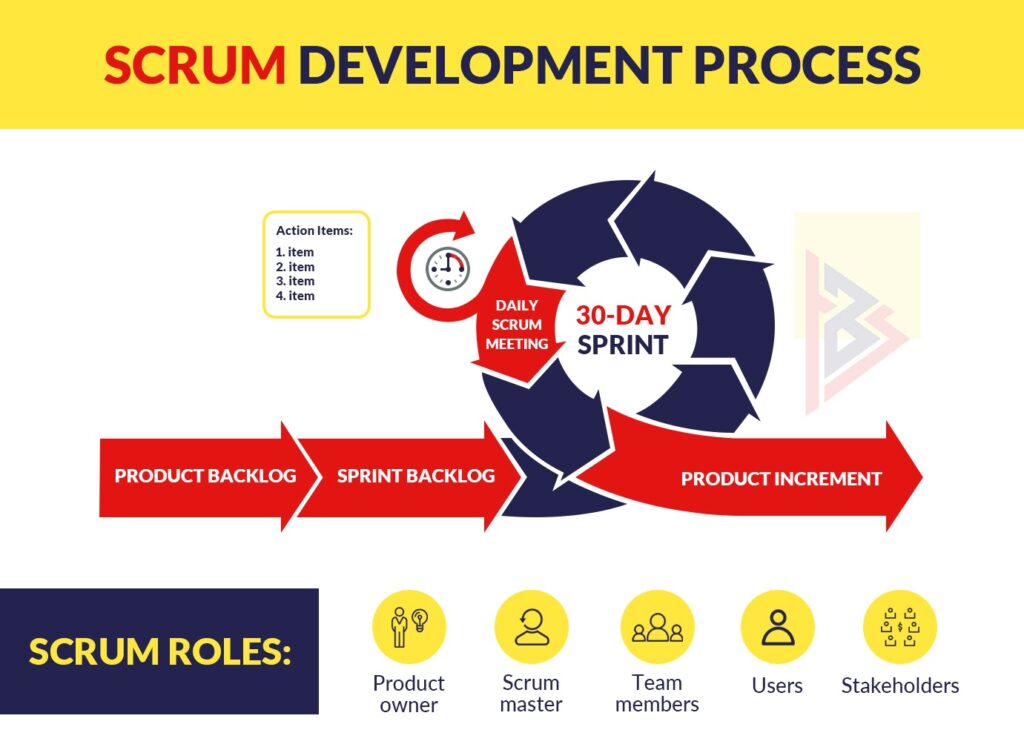
problèmes non techniques.

La figure suivante représente la zone d'intersection des responsabilités des divers acteurs impliqués dans un Framework Scrum :



Le tableau 1 met en perspective le Framework Scrum par rapport à notre projet. Il détaille les intervenants impliqués, leurs fonctions respectives, ainsi que les tâches qui leur incombent.

|  |  |  |
| --- | --- | --- |
| Rôle | Mission | Acteur(s) |
| Scrum team | Conception, développement, test et Validation, déploiement | Houimel Oumaima+  Mahjoub Ali |
| Product Owner | Définition des besoins du système | Mr Rochdi Jedidi |
| Scrum Master | Admission du projet | Mme Bouzid Emna |



➢ Démarrage du projet avec Scrum :

Le Product Owner est responsable de l’entretien du Product Backlog. Il s’agit de la liste des foncionnalités du produit à developper. Le Product Backlog est une liste ordonnée par priorité des besoins fonctionnels et exigences de l’utilisateur qui sont appelées des histoires utilisateurs (User Stories).

Ces dernières sont une description courte de l’exigence fonctionnelle formulée

en une phrase sous la forme : « En tant que "acteur", je veux "un service" ». Dans notre rapport, le Backlog du produit comprennent les champs suivants :

• Thème : représente le thème du user story.

• ID : représente l’identifiant de chaque story

• User story : représente le besoin fonctionnel désiré par l’utilisateur

• Importance : définit par la méthode "MoSCow" dans laquelle chaque user story

estpriorisée comme suivant :

✓ Must : la fonctionnalité doit être absolument réalisée

✓ Should : la fonctionnalité doit être réalisée s’il est possible

✓ Could : la fonctionnalité peut être réalisée

✓ Maybe : la fonctionnalité peut être réalisée si elle n’a pas d’impact

sur les autres tâches

✓ Won’t : la fonctionnalité sera réalisée plus tard

L’estimation du contenu du Product Backlog se fait par l’équipe pendant la réunion de

planification de la prochaine itération (Sprint Planning Meeting). Donc, le Product Owner identifie les tâches destinées à développer lors de la prochaine itération en identifiant la priorité des fonctionnalités. Le contour opérationnel réalisé lors d’itération est transformé par l’équipe en tâches estimées dans le Sprint Backlog.

Le sprint est une itération dans laquelle se réalise le développement des fonctionnalités défini par le Sprint Backlog petit à petit.

La mêlée quotidienne : est une courte réunion dans laquelle participent le Scrum Master et l’équipe dans le but de répondre à trois questions pour contrôler la réalisation des tâches du projet. Les questions sont : Qu’est-ce que vous avez fait hier ? Que vas-tu faire aujourd’hui ?Y’a-t-il des obstacles qui empêchent la réalisation d’une tâche ?

Le Sprint Review Meeting est une réunion de fin de sprint dans laquelle l’équipe présente au product owner une démonstration du produit et précise ce qui est fini et ce qui ne l’est pas.

Lors de Sprint Rétrospective Meeting, l’équipe est occupée de planifier les moyens pour améliorer sa qualité et son efficacité pour la prochaine itération.

▪ Backlog Produit Global

Le tableau 2 résume le backlog de produit global de notre application. Chaque thème de ce back- log est constitué d’un ou plusieurs user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thème** | | **ID** | **User story** | **Importance** |
| 1 | Authentification | 1 | En tant qu'utilisateur, je souhaite  m'authentifier en utilisant le nom d'utilisateur et le mot de passe fournis par l'administrateur  lors de la création de mon compte, afin d'accéder à mon compte. | Must |
| 2 | Consulter la liste des utilisateurs | 2 | Comme un administrateur, je veux avoir la Liste des utilisateurs. | Must |
| 3 | Modification du Compte utilisateur | 3 | En tant qu’administrateur, je souhaite  modifier les informations d’un utilisateur (nom, prénom, adresse e-mail, etc.) afin de les mettre à jour. | Could |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. **Diagramme de cas d’utilisation global :**

Cette section vise à décrire le comportement global attendu des acteurs de l'application. Pour cela, nous allons présenter les actions générales que peuvent effectuer les utilisateurs et les administrateurs de cette application à travers un diagramme de cas d'utilisation, utilisant le langage de modélisation UML.

Le diagramme de cas d’utilisation est un diagramme UML (Unified Modeling Language) qui présente le comportement fonctionnel d’un système logiciel. Il est utile pour des présentations auprès de la direction ou des acteurs d’un projet. Pour le développement, les cas d’utilisation sont plus appropriés. En effet, un cas d’utilisation (use case) représente une unité discrète d’interaction entre un utilisateur (humain ou machine) et un système [10].

Le diagramme 12 illustre les diverses activités et rôles sous la gestion et le contrôle de l'administrateur, ainsi que les diverses fonctionnalités proposées aux utilisateurs.

1. **Diagramme de classes :**

Pour cette section, nous illustrons l’aspect statique de notre application en présentant le diagramme de classes.

Les diagrammes de classes sont l'un des types de diagrammes UML les plus utiles, car ils décrivent clairement la structure d’un système particulier en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets [11].

1. **L'environnement de développement :**

L'environnement de développement fait référence à l'ensemble des outils, des ressources et des configurations utilisés pour concevoir, coder, tester et déployer des logiciels. Cela inclut l'environnement logiciel ainsi que les technologies et langages utilisés .

* 1. **Outils de conception :**

|  |  |
| --- | --- |
| Outils | Description |
| **Visual paradigm**  visual | Est un logiciel de modélisation et de conception visuelle pour le développement logiciel et la gestion de projets. |

* 1. **Outils de développement :**

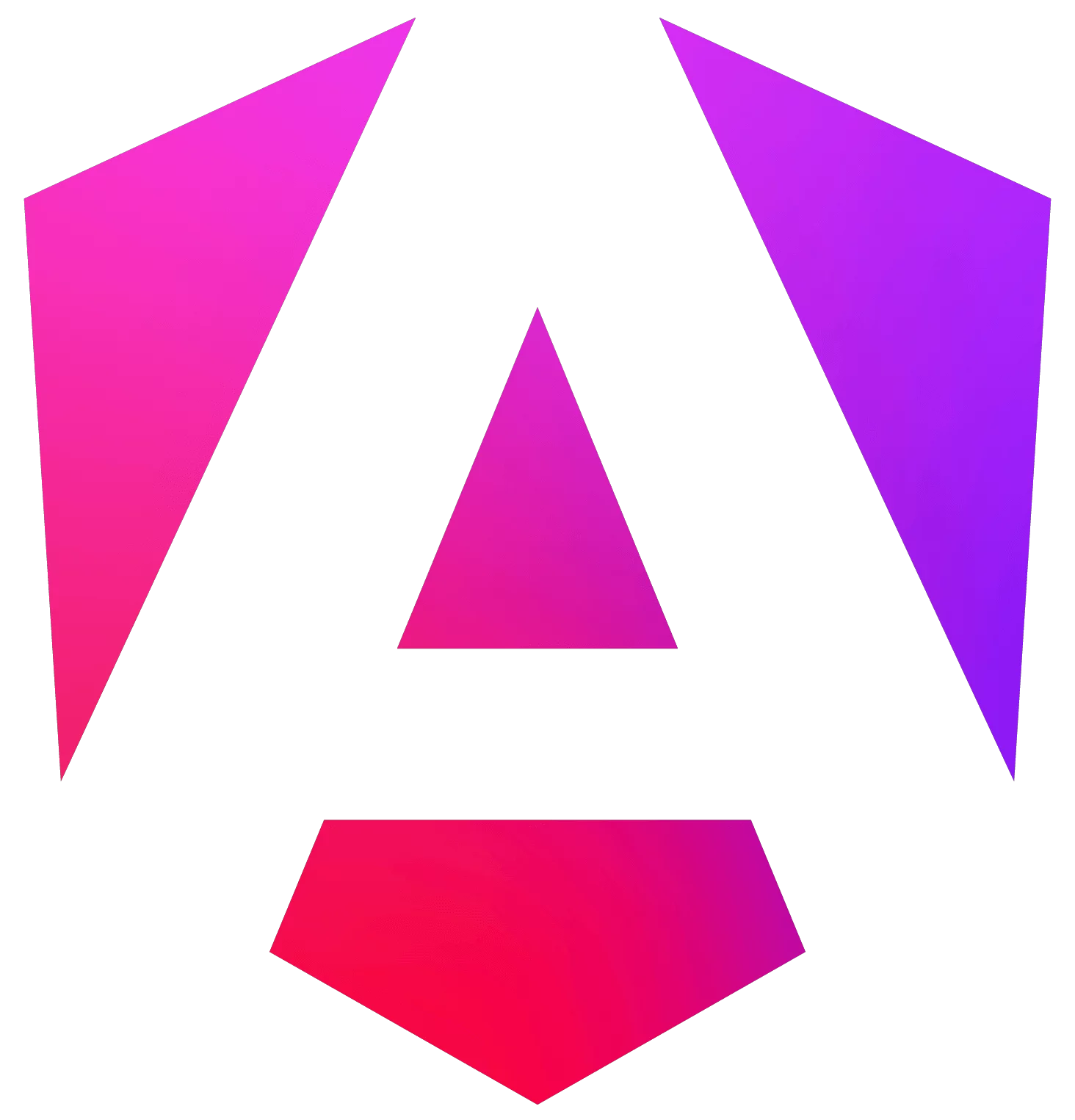
|  |  |
| --- | --- |
| Outils | Description |
| **Visual Studio Code** | Est un éditeur de code source léger, extensible et multiplateforme |
| **HTML 5** | C’est un langage qui permet de composer des pages web. |
| **CSS3** | Est un langage de style qui contrôle la manière dont les documents HTML sont présentés, incluant des aspects comme les polices, les couleurs, et les marges. |
| **Bootstrap** | Est un framework front-end pour le développement web, offrant des outils et des modèles prêts à l'emploi pour créer des sites réactifs et mobiles. |
| **Postman** | Les applications modernes reposent largement sur des APIs. Celle-ci est considérée comme l'API la plus complète, bénéficiant de l'adhésion de plus de 100 000 entreprises à travers le monde. |
| **PhpMyAdmin** | C'est une application Web de gestion pour les bases de données MySQL, développée en PHP et distribuée sous licence GPL. Elle est réputée comme l'une des interfaces les plus populaires pour gérer efficacement une base de données MySQL sur un serveur PHP. |
| **Xampp** | Est un logiciel libre et gratuit qui facilite la création et la gestion de serveurs web locaux. |
| **PowerBI** | Power BI est une suite d'outils d'analyse de données et de visualisation développée par Microsoft. |

Dans la suite, nous donnerons plus de détails sur les principales technologies utilisées, à savoir

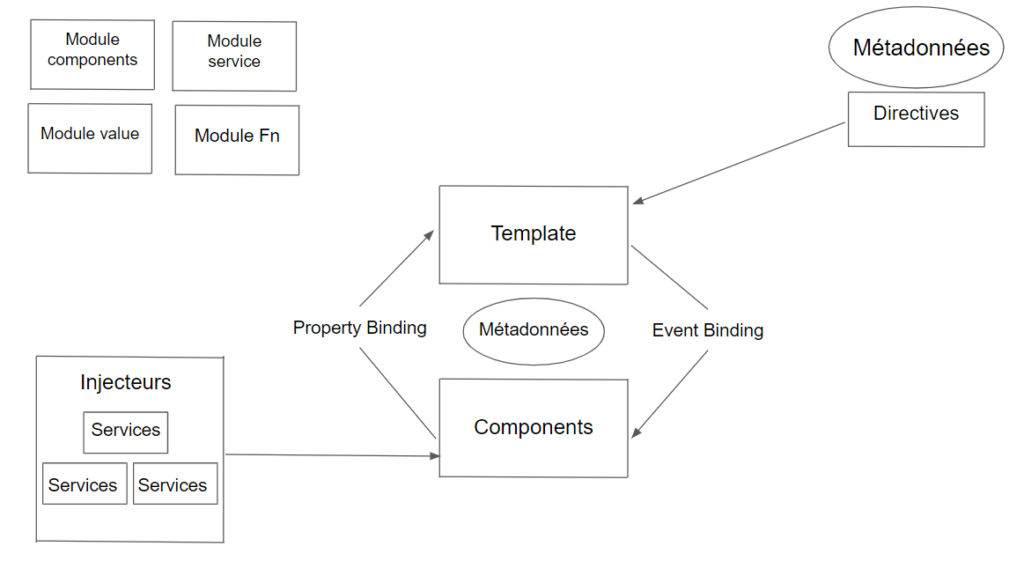
Angular 17, Laravel.

* 1. **Environnements de développement utilisés :**

Angular est un Framework JavaScript open source développé par Google. Est l’un des Framework les mieux réputés et les plus utilisés par les développeurs de front-end. Ce Framework est utilisé pour développement des applications web en javascript (TypeScript) [12]



La figure 19 représente l’architecture globale d’Angular :



La figure ci-dessus montre le Framework Angular qui est basé sur plusieurs concepts, nous détaillons dans la suite quelques-uns :

▪ **Le module** : c’est une classe type script avec le décorateur **@NgModule**, c’est la première structure de base dans Angular il est un peu comme un package Java. Une application Angular est composée d’au moins un module.

**▪ Le composant :** c’est une classe type script avec le décorateur **@Componant**. Il représente une vue de l’application. Une application Angular est un arbre de composants Angular.

**▪ Le service :** C’est une classe type script avec le décorateur **@Injectable**. Il sert à partager des données entre les composants, ou bien factoriser un traitement réutilisable.

Est un [Framework](https://fr.wikipedia.org/wiki/Framework) [web](https://fr.wikipedia.org/wiki/Web_application) [open-source](https://fr.wikipedia.org/wiki/Open-source) écrit en [PHP](https://fr.wikipedia.org/wiki/PHP) respectant le principe [modèle-vue-contrôleur](https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur) et entièrement développé en [programmation orientée objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet) [13].

Il offre une structure et des fonctionnalités puissantes pour la création d'applications web modernes et robustes.



1. **CONCLUSION :**

Dans le deuxième chapitre, nous avons précisé les besoins et adopté le cadre Scrum. Le Backlog de produit global et le diagramme de cas d'utilisation global ont été présentés, accompagnés de choix techniques pour l'application.

Le prochain chapitre détaillera le travail réalisé lors du premier sprint du projet.

Chapitre3 :

Bibliographie

[7] : https://fr.ryte.com/wiki/Product\_owner. [En ligne] [Citation : 17 Mars 2024].

[8] :https://www.tuleap.org/fr/agile/comprendre-methode-agile-scrum-10-minutes#grands-principes-scrum. [En ligne] [Citation : 17 Mars 2024].

[9] : <https://blog.myagilepartner.fr/index.php/2018/05/27/role-de-scrum-master/>. [En ligne] [Citation : 17 Mars 2024].

[10] : <https://fr.wikipedia.org/wiki/Diagramme_de_cas_d%27utilisation>. [En ligne] [Citation : 17 Mars 2024].

[11] : <https://www.lucidchart.com/pages/fr/diagramme-de-classes-uml>. [En ligne] [Citation : 17 Mars 2024].

[12] : https://en.wikipedia.org/wiki/JSON. [En ligne] [Citation : 18 Mars 2024].

[13] :https://fr.wikipedia.org/wiki/Laravel#:~:text=Laravel%20est%20un%20framework%20web,ses%20sources%20h%C3%A9berg%C3%A9es%20sur%20GitHub. [En ligne] [Citation : 18 Mars 2024].