



République Tunisienne

\*\*\*\*\*

Ministère de l'Enseignement Supérieur et de la  
Recherche Scientifique

\*\*\*\*\*

Université de Monastir \*\*\*\*\*

Institut Supérieur d'Informatique et de Mathématiques  
de Monastir



# Projet de Fin d'Études

En vue de l'obtention du

## DIPLÔME NATIONAL DE LICENCE EN SCIENCES INFORMATIQUE

Spécialité : 1 Genie Logiciel

---

Conception et développement d'une application  
web d'évaluation des fournisseurs

---

*Réalisé par :*  
Tabbassi Ahlem  
Bensalem Aziz

Année universitaire 2023-2024

# Table des matières

<b>1</b>	<b>Contexte et objectifs</b>	<b>2</b>
	<b>Contexte et objectifs</b>	<b>2</b>
	Introduction . . . . .	2
	1. Positionnement et Problématique . . . . .	2
	1.1 Contexte général . . . . .	2
	1.2 Cadre académique . . . . .	2
	1.3 Présentation de l'organisme d'accueil . . . . .	2
	2. Problématique . . . . .	3
	3. Étude de l'existant . . . . .	3
	3.1 Solutions existantes . . . . .	3
	3.2 Critique de l'existant . . . . .	4
	4. Solution proposée . . . . .	5
	5. Méthodologie de travail et modélisation . . . . .	6
	5.1 La méthode agile : . . . . .	6
	5.2 Processus Unifié (UP) . . . . .	6
	5.3 Comparaison des méthodes et justification du choix . . . . .	7
	Conclusion . . . . .	9
<b>2</b>	<b>Analyse et capture des besoins</b>	<b>10</b>
	<b>Analyse et capture des besoins</b>	<b>10</b>
	Introduction . . . . .	10
	Analyse et identification des besoins . . . . .	10
	1 Identification des acteurs . . . . .	10
	2 Besoins fonctionnels . . . . .	10
	3 Besoins non fonctionnels . . . . .	11
	4 Besoins techniques . . . . .	11
	4.1 Choix technologique . . . . .	12
	4.1.1 Frontend . . . . .	12
	4.1.1.1 ReactJs . . . . .	12
	4.1.1.2 AngularJs . . . . .	13
	4.1.1.3 VueJs . . . . .	13
	4.1.2 Backend . . . . .	14
	4.1.2.1 Serveur . . . . .	14
	4.1.2.2 Étude comparative entre les Framework Backend . . . . .	14
	4.1.2.3 Base de données . . . . .	15
	Choix de MongoDB . . . . .	15
	4.1.2.4 Classement des bases de données . . . . .	16
	4.1.3 Choix technologique de notre application . . . . .	16
	4.2 Choix du modèle conceptuel . . . . .	17
	4.2.1 MVC . . . . .	17
	4.2.2 MVVM . . . . .	18
	4.3 Notre choix . . . . .	19
	Conclusion . . . . .	19

# Chapitre 1

## Contexte et objectifs

### Introduction

Dans ce chapitre, nous aborderons l'étude préalable du projet, en commençant par présenter le contexte général, le cadre académique, et l'organisme d'accueil. Nous exposerons ensuite la problématique, suivie d'une analyse critique de l'existant et des solutions proposées. Enfin, nous discuterons du choix de la méthodologie, en comparant différentes méthodes et en justifiant notre choix.

### 1. Positionnement et Problématique

#### 1.1 Contexte général

Dans toute entreprise, la gestion des risques liés aux fournisseurs est devenue l'une des principales missions des acheteurs d'aujourd'hui. Elle représente une priorité pour 60% d'entre eux. La qualité des produits ou services fournis, les coûts associés et la réputation de l'entreprise sont directement influencés par une gestion efficace de l'évaluation des fournisseurs. Cette évaluation permet d'identifier les partenaires les plus fiables et qualifiés, ce qui contribue à minimiser les risques associés à la chaîne d'approvisionnement et à optimiser les performances globales de l'entreprise. En somme, une approche rigoureuse dans la gestion des fournisseurs garantit non seulement la qualité des produits finaux, mais aussi la pérennité et la compétitivité de l'entreprise sur le marché.

#### 1.2 Cadre académique

Le présent travail s'inscrit dans le cadre du projet de fin d'études en vue de l'obtention du Diplôme de Licence en Sciences Informatique à « l'Institut Supérieur d'Informatique et de Mathématiques de Monastir ». Notre stage a été réalisé pendant une période de quatre mois au sein de la société "Vernicolor" dans le but d'atteindre un objectif final qui consiste à concevoir et à développer une application web d'évaluation des fournisseurs.

#### 1.3 Présentation de l'organisme d'accueil

Notre projet est réalisé au sein de l'entreprise Vernicolor, situé a Nabeul Z.I.Impasse de la physique Grombalia crée depuis 2008 en tunisie.Son domaine d'activité est la décoration plastique.Le Groupe VERNICOLOR est un leader dans la fabrication de pièces plastiques décoratives pour l'habitacle des voitures, peintes ou chromées



FIGURE 1.1 – Logo de l'entreprise Vernicolor

## 2. Problématique

Vernicolor Tunisie reconnaît l'importance cruciale d'établir un système efficace d'évaluation des fournisseurs. En mettant en place un tel système, Vernicolor Tunisie cherche à renforcer sa capacité à maintenir des normes élevées de qualité tout au long de sa chaîne d'approvisionnement, tout en optimisant ses processus opérationnels. La création d'une application web d'évaluation des fournisseurs sera alors un moyen pour la gestion de la chaîne d'approvisionnement, à assurer la cohérence et la fiabilité des fournitures, et à renforcer ses relations avec ses partenaires commerciaux. Cette application aura pour objectif d'intégrer des critères d'évaluation pertinents et alignés sur les objectifs stratégiques de l'entreprise. Elle permettra également de rationaliser les processus d'évaluation, de collecte de données et de communication avec les fournisseurs, tout en garantissant la confidentialité des informations sensibles. En adoptant cette approche axée sur la technologie, Vernicolor vise à améliorer sa compétitivité sur le marché en maintenant des normes élevées de qualité et d'efficacité dans sa chaîne d'approvisionnement.

## 3. Étude de l'existant

Dans le cadre de notre projet de plateforme d'évaluation des fournisseurs VERNICOLOR, il est crucial de comprendre les pratiques existantes dans le domaine de l'évaluation des fournisseurs. Nous allons examiner quelques plateformes déjà en place pour cette tâche, afin de mieux comprendre les fonctionnalités et les approches qui pourraient être pertinentes pour notre projet.

### 3.1 Solutions existantes

#### 3.1.1 ACESIA

ACESIA est une plateforme en ligne spécialisée dans l'évaluation des fournisseurs. Elle offre des outils avancés pour collecter et analyser les performances des fournisseurs, ainsi que pour gérer les contrats et les relations. Grâce à ACESIA, les entreprises peuvent évaluer la qualité et la fiabilité de leurs fournisseurs de manière transparente et objective, améliorant ainsi leurs décisions d'approvisionnement.

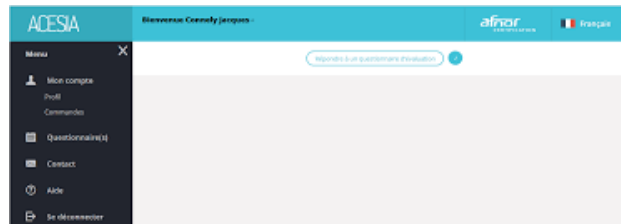


FIGURE 1.2 – Capture du site Acesia

#### 3.1.2 IsyBuy

IsyBuy est un logiciel Source-to-Pay conçu à la fois pour les directions achats, les acheteurs, ainsi que pour les opérationnels qui ne sont pas familiers avec les processus d'achat. Il comprend des modules allant de la gestion des fournisseurs jusqu'à la facturation, en passant par la sélection des articles sur les catalogues. Elle offre un module de Supplier Relationship Management (SRM), qui est un ensemble de pratiques visant à améliorer la performance des fournisseurs afin de favoriser une collaboration efficace et durable. Cette approche comprend l'analyse et l'évaluation de la performance des fournisseurs, le partage facile d'informations avec eux, ainsi que la collaboration en temps réel pour renforcer les relations et les transformer en véritables partenariats.

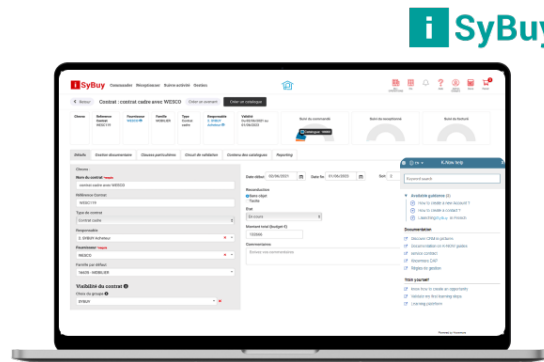


FIGURE 1.3 – Capture du site isybuy

### 3.1.3 Notion

Notion est une application de productivité tout-en-un qui offre une variété de fonctionnalités pour la gestion de projet, la prise de notes, la création de bases de données, la collaboration en équipe et bien plus encore.

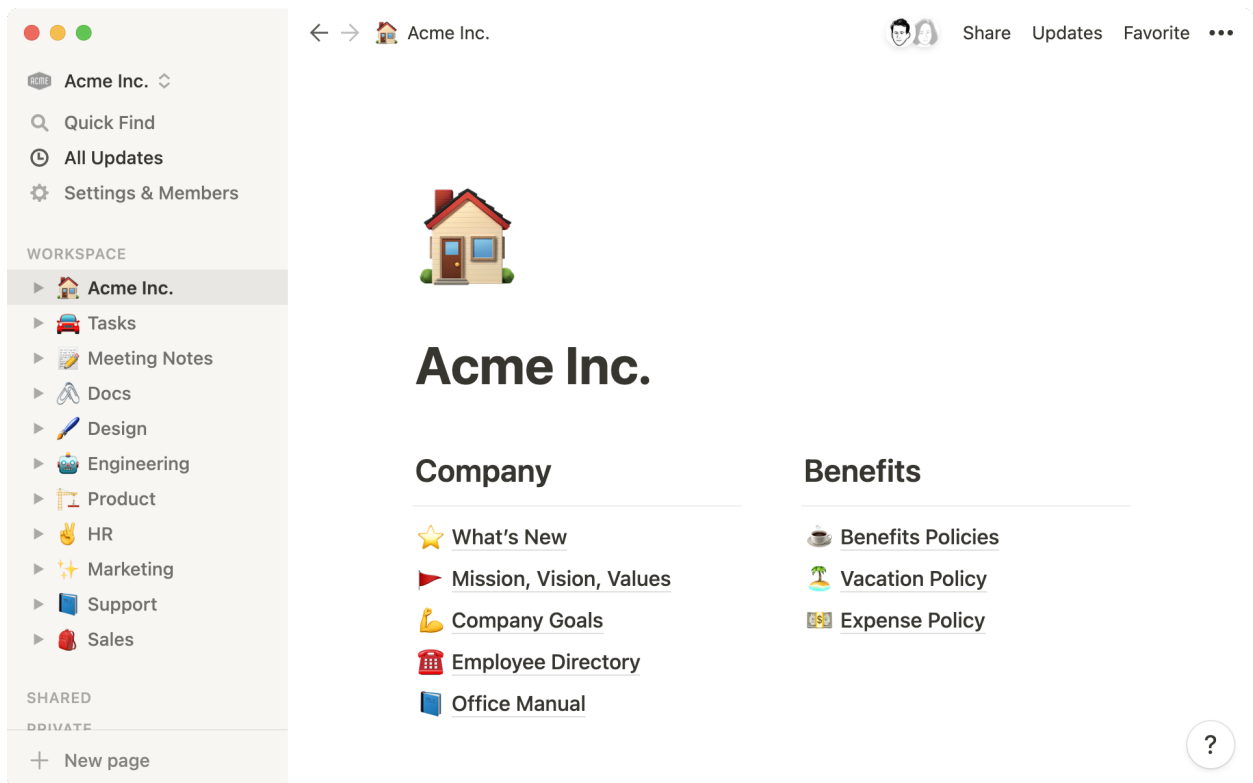


FIGURE 1.4 – Capture du site Notion

## 3.2 Critique de l'existant

Nous formulerons une critique constructive de l'existant, mettant en lumière ses limitations.

- **Gestion du profil** : L'utilisateur du site possède un profil contenant ses informations personnelles et il peut les modifier.
- **Messages** : Communication entre les utilisateurs à travers le service de messagerie.
- **Notifications** : L'utilisateur reçoit des notifications.

- **Frais de service** : Les applications ont des frais d'utilisation.
- **Gestion des certificats** : Fonctionnalité permettant de gérer les certificats des fournisseurs.
- **Gestion des évaluations** : Fonctionnalité permettant aux utilisateurs d'attribuer des évaluations aux fournisseurs selon des critères précis.
- **Gestion des utilisateurs** : Capacité d'une plateforme à gérer l'attribution des rôles permettant aux utilisateurs d'exercer des tâches parmi d'autres.

TABLE 1.1 – Analyse des plateformes existantes

Critères de l'étude	ACESIA	IsyBuy	Notion
Gestion du profil	Oui	Oui	Oui
Messages	Non	Non	Non
Notifications	Non	Oui	Oui
Frais de service	Oui	Oui	Oui (possède une version gratuite)
Gestion des certificats	Non	Non	Non
Gestion des évaluations	Oui	Oui	Oui
Gestion des utilisateurs	Oui	Oui	Non

## 4. Solution proposée

Tenant compte des critiques de l'existant et des résultats précédemment mentionnés, il est évident qu'aucune des solutions actuellement disponibles ne répond de manière adéquate à notre problématique. Cependant, notre objectif est de proposer une solution spécifique et parfaitement adaptée aux besoins de notre entreprise cliente. Pour ce faire, nous prévoyons de développer une application web sur mesure pour évaluer les fournisseurs. Cette application sera conçue pour répondre aux besoins spécifiques de Vernicolor Tunisie, en intégrant des critères d'évaluation alignés sur ses objectifs stratégiques. Elle permettra de maintenir des normes élevées de qualité dans la chaîne d'approvisionnement, d'optimiser les processus opérationnels et de renforcer les relations avec les partenaires commerciaux. En adoptant cette approche technologique, Vernicolor vise à améliorer sa compétitivité en garantissant la cohérence et la fiabilité des fournitures, tout en assurant la confidentialité des informations sensibles. Notre solution proposée est illustrée dans la gure suivante :

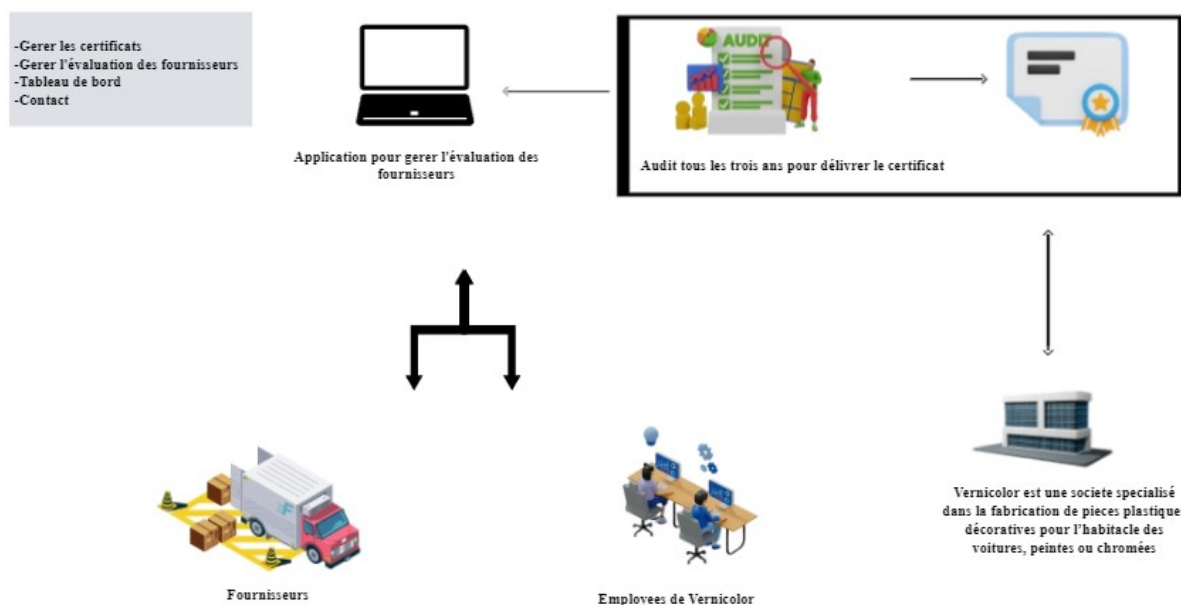


FIGURE 1.5 – Illustration de la solution

## 5. Méthodologie de travail et modélisation

Afin de mener à bien un projet et d'en garantir le succès, il est essentiel d'adopter une méthodologie adaptée qui guide sa réalisation de manière étape par étape, de la planification à la mise en œuvre, de manière à la fois simplifiée et structurée. Dans cette optique, pour choisir la méthode la plus appropriée, une étude approfondie de toutes les méthodes disponibles est nécessaire.

### 5.1 La méthode agile :

#### 5.1.1 Scrum :

La méthodologie Scrum repose sur un processus de développement de logiciels itératif et incrémental, caractérisé par une transparence totale des exigences d'évolution ou de correction à mettre en œuvre. Elle simplifie la gestion de projets complexes en structurant le travail en "sprints" de développement d'une durée généralement de deux à quatre semaines. À chaque itération, le client reçoit une version fonctionnelle du logiciel, favorisant ainsi une progression continue du produit grâce à l'ajout de nouvelles fonctionnalités à chaque phase du projet.

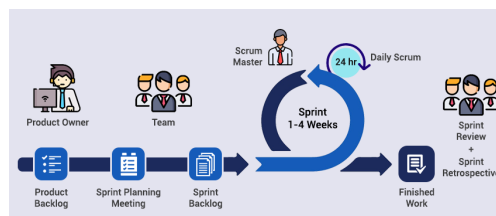


FIGURE 1.6 – Cycle de vie de Scrum

#### 5.1.2 eXtreme Programming (XP)

La méthodologie XP est une approche de gestion de projet qui pousse les principes du développement agile à leur maximum. Son focus se concentre sur les besoins du client en adoptant un processus de développement itératif et en promouvant l'intégration continue. Au cœur de XP se trouvent les relations étroites entre l'équipe projet et le client.

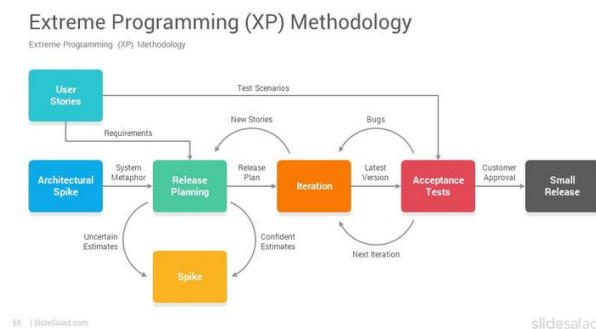


FIGURE 1.7 – Cycle de vie de XP

### 5.2 Processus Unifié (UP)

#### 5.2.1 RUP

Le Processus Unifié Rational est une méthodologie de développement de logiciels basée sur UML qui organise le développement de logiciels en quatre phases, chacune comprenant une ou plusieurs itérations exécutables du logiciel à cette étape du développement. Le RUP favorise la communication entre les différents acteurs du projet, mais il est considéré comme très complexe par rapport aux autres méthodologies disponibles.

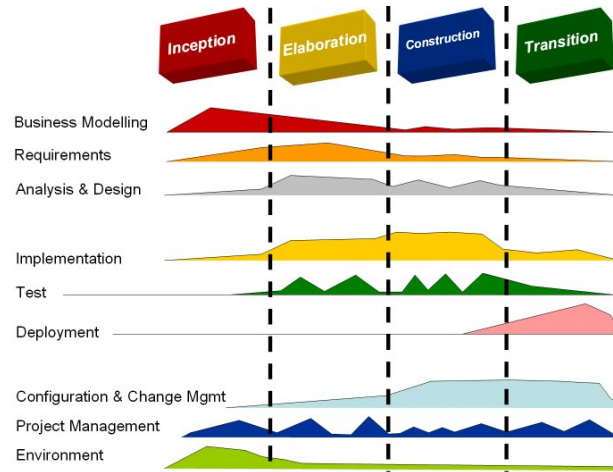


FIGURE 1.8 – Cycle de vie de RUP

### 5.2.2 2TUP

Le 2Track Unified Process est une méthode itérative de développement de logiciels qui s'appuie sur l'architecture logicielle et met en œuvre le processus unifié. Concrètement, elle se base sur le cycle de vie en Y. Cette approche du développement de logiciels permet de distinguer clairement les besoins fonctionnels et techniques tout en réduisant les risques associés au projet.

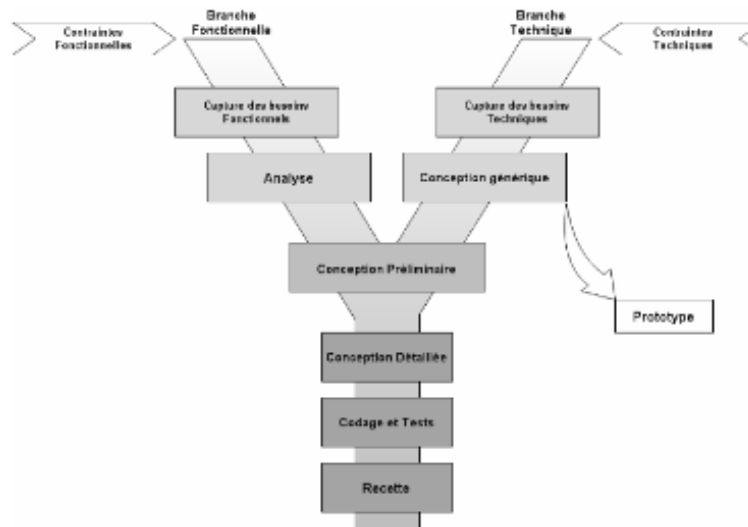


FIGURE 1.9 – Cycle de vie de 2TUP

## 5.3 Comparaison des méthodes et justification du choix

Après avoir examiné brièvement ces méthodologies, nous établissons le tableau comparatif ci-dessous afin de choisir la méthode la plus adéquate à notre projet.

Après une analyse approfondie des différentes méthodologies disponibles, nous avons choisi d'adopter la méthode de processus unifié à deux voies (2tup) pour la gestion de notre projet. Scrum est une approche agile qui se concentre sur la collaboration de l'équipe et l'adaptabilité aux changements. Il offre une structure flexible mais bien définie pour le développement de logiciels, ce qui correspond parfaitement à nos besoins pour ce projet. De plus, Scrum met l'accent sur la création de valeur pour le client à chaque itération, ce qui est essentiel pour assurer la satisfaction du client et la réussite du projet. En adoptant la méthode Scrum, nous sommes convaincus que nous pourrions gérer efficacement les défis de ce projet et fournir une solution de haute qualité à notre client.



Méthodologie	Points forts	Points faibles
<b>Scrum</b>	<ul style="list-style-type: none"> <li>— Basé sur des itérations</li> <li>— Axé sur la création d'un produit final offrant une valeur ajoutée significative</li> <li>— Favorise l'efficacité et la réduction des coûts</li> <li>— Favorise la collaboration au sein de l'équipe</li> <li>— Vise à améliorer la satisfaction des utilisateurs</li> </ul>	<ul style="list-style-type: none"> <li>— Dimension de l'équipe</li> <li>— Complexité de maîtrise</li> <li>— Réduction de la documentation</li> </ul>
<b>XP</b>	<ul style="list-style-type: none"> <li>— Basé sur des itérations progressives</li> <li>— Priorisation des risques</li> <li>— Engagement envers la qualité du code</li> <li>— Adaptabilité aux changements</li> <li>— Surveillance continue</li> <li>— Simplicité des processus</li> </ul>	<ul style="list-style-type: none"> <li>— Exige des spécialistes</li> <li>— Coûteux à adapter</li> <li>— Documentation limitée</li> </ul>
<b>RUP</b>	<ul style="list-style-type: none"> <li>— Structuré et formel</li> <li>— Définition claire des tâches et des rôles</li> <li>— Documentation exhaustive</li> <li>— Peut s'adapter à différents types de projets</li> <li>— Approche modulaire</li> </ul>	<ul style="list-style-type: none"> <li>— Lourd et complexe</li> <li>— Coûteux en ressources</li> <li>— Adaptation difficile</li> <li>— Nécessite une expertise</li> </ul>
<b>2TUP</b>	<ul style="list-style-type: none"> <li>— Flexibilité</li> <li>— Prise en compte des aspects fonctionnels et techniques</li> <li>— Gestion des risques</li> <li>— Itérations courtes</li> <li>— Architecture bien définie</li> </ul>	<ul style="list-style-type: none"> <li>— Peut être perçu comme moins rigoureux</li> <li>— Nécessite une solide architecture de départ</li> <li>— Dépendance vis-à-vis des fournisseurs de technologies</li> </ul>

TABLE 1.2 – Comparaison des méthodologies de développement de logiciels

## Conclusion

En conclusion, ce chapitre a permis d'établir le contexte et les objectifs de notre projet, en mettant en lumière l'importance de la gestion des fournisseurs pour les entreprises et en présentant notre client, l'entreprise Vernicolor. Nous avons également réalisé une étude de l'existant, en examinant quelques-unes des plateformes actuellement disponibles pour l'évaluation des fournisseurs, ainsi qu'une critique constructive de ces solutions. Enfin, nous avons proposé notre propre solution, une application web sur mesure pour évaluer les fournisseurs, et nous avons discuté de la méthodologie que nous allons utiliser pour gérer ce projet, en justifiant notre choix de la méthode Scrum. Dans les chapitres suivants, nous entrerons dans les détails de la mise en œuvre de notre solution, en commençant par la modélisation de notre application et la planification de notre projet en utilisant la méthode Scrum.

# Chapitre 2

## Analyse et capture des besoins

### Introduction

Après avoir présenté le contexte général de notre projet, nous nous concentrons dans ce chapitre sur la détermination des fonctionnalités attendues du système. Nous débutons par la présentation des acteurs concernés par notre système, puis nous entamons l'étude des besoins fonctionnels et non fonctionnels. Ces besoins seront ensuite exprimés sous la forme de diagrammes de cas d'utilisation, détaillant les scénarios réalisables par les différents acteurs.

### Analyse et identification des besoins

Dans cette partie, nous définirons les acteurs et leurs fonctions, puis nous examinerons les besoins fonctionnels et non fonctionnels que notre application doit répondre.

#### 1 Identification des acteurs

Tout système interactif doit garantir et simplifier l'interaction entre ses utilisateurs, qu'ils soient humains ou non. Un acteur représente le rôle d'une entité externe qui utilise le système à travers ses diverses interfaces. Dans notre système, nous avons identifié les acteurs suivants :

##### Acteurs principaux :

- **Admin** : Responsable du site, assure son bon fonctionnement et gère les comptes des fournisseurs et des employés.
- **Employé** : Peut gérer son compte, évaluer des fournisseurs, créer des comptes pour les fournisseurs, consulter les certificats, les évaluations et le tableau de bord.
- **Fournisseur** : Peut gérer son compte, gérer les certificats, consulter l'évaluation et le tableau de bord.

##### Acteurs secondaires :

- **Système de notifications** : Envoie des notifications aux fournisseurs, comme des alertes lorsque le délai des certificats est proche, ou aux employés pour effectuer les évaluations mensuelles.
- **Système de messagerie** : Permet la communication entre les employés et les fournisseurs.

#### 2 Besoins fonctionnels

Un besoin fonctionnel est une exigence spécifique décrivant une action que le système doit pouvoir accomplir. Notre solution vise à répondre à plusieurs de ces besoins fonctionnels.

Le plateforme doit permettre à l'**employé** de :

- **S'authentifier** : Permet à l'employé de se connecter à son espace avec ses identifiants.
- **Consulter les évaluations des fournisseurs** : Donne à l'employé la possibilité de consulter les évaluations passées des fournisseurs
- **Gérer son profil** : Permet à l'employé de consulter, mettre à jour ou supprimer ses informations personnelles.
- **Gérer les évaluations des fournisseurs** : Autorise l'employé à saisir de nouvelles évaluations pour les fournisseurs.

- **Consulter les certificats des fournisseurs** : Donne à l'employé la possibilité de consulter les certificats et les accréditations des fournisseurs.
- **Recevoir des notifications** : Permet à l'employé de recevoir des notifications concernant les évaluations, les certificats ou d'autres mises à jour pertinentes.
- **Communiquer via un système de messagerie** : Donne à l'employé la possibilité d'écrire, supprimer et recevoir des messages avec d'autres acteurs de la plateforme.
- **Gérer compte fournisseur** : Permet à l'employé de créer ou supprimer des comptes fournisseurs.

Le plateforme doit permettre à l'**admin** de :

- **S'authentifier** : Permet à l'admin de se connecter à son espace avec ses identifiants.
- **Gérer les comptes** : Autorise l'admin à gérer les comptes des fournisseurs et employés, y compris la création, la modification et la suppression des comptes.
- **Gérer les évaluations et les certificats** : Donne à l'admin la possibilité de superviser les évaluations des fournisseurs et les certificats, y compris l'approbation ou le rejet des évaluations soumises.
- **Gérer les paramètres de la plateforme** : Permet à l'admin de configurer les paramètres de la plateforme, tels que les notifications, les autorisations d'accès, etc.
- **Communiquer avec les utilisateurs** : Autorise l'admin à communiquer avec les employés et les fournisseurs via un système de messagerie intégré.

Le plateforme doit permettre au **fournisseur** de :

- **S'authentifier** : Permet au fournisseur de se connecter à son espace avec ses identifiants.
- **Consulter son profil** : Donne au fournisseur la possibilité de consulter et de mettre à jour ses informations personnelles.
- **Gérer des certifications** : Permet au fournisseur d'ajouter et de modifier des certifications.
- **Consulter les évaluations** : Donne au fournisseur la possibilité de consulter les évaluations qui lui sont attribuées.

Le plateforme doit permettre avec **le système de notification** de :

- **Recevoir des notifications** : Permet au fournisseur de recevoir des notifications concernant les évaluations, les certificats etc.

Le plateforme doit permettre avec **le système de messagerie** de :

- **Communiquer avec l'employé** : Donne au fournisseur la possibilité de communiquer avec l'admin via un système de messagerie intégré.

### 3 Besoins non fonctionnels

Les besoins non fonctionnels englobent toutes les exigences nécessaires pour garantir le bon fonctionnement du système et améliorer la qualité des services fournis aux utilisateurs. Notre système doit respecter les critères ci-dessous :

- **L'ergonomie des interfaces** : L'application doit respecter les contraintes ergonomiques suivantes : une interface simple, un accès rapide à l'information et une manipulation facile, ainsi que des interfaces graphiques conviviales et compréhensibles. De plus, elle doit utiliser des messages informatifs et d'erreur clairs pour améliorer l'expérience utilisateur.
- **La performance** : Nous cherchons à créer une application web mono-page (SPA) afin d'améliorer les performances. Cette approche permettra de réduire le temps de chargement et d'éviter le rechargement de la page à chaque action de l'utilisateur. Ainsi, nous assurerons une expérience utilisateur conviviale dès l'accès au site.
- **La sécurité** : Il est essentiel de prendre en compte la nécessité d'attribuer des rôles à chaque groupe d'utilisateurs. Cette mesure garantit la protection des différents espaces d'accès de notre application.

### 4 Besoins techniques

L'étape de capture des besoins techniques répertorie toutes les contraintes relatives à la conception du système, ainsi que les outils et le matériel sélectionnés. Dans le but de garantir une décision éclairée, une analyse des différents langages de développement et des frameworks sera réalisée dans le reste de cette section.

## 4.1 Choix technologique

### 4.1.1 Frontend

Un framework front-end est un ensemble de bibliothèques, d'outils et de conventions qui permettent aux développeurs de créer des interfaces utilisateur pour des applications web de manière plus rapide, plus efficace et plus cohérente. Le nombre des technologies de développement côté client ne cesse d'augmenter. Parmi les langages les plus populaires, on peut citer : ReactJs, AngularJs, VueJs.



FIGURE 2.1 – Les technologies Front-End les plus populaires

#### 4.1.1.1 ReactJs

React, créé par Facebook, simplifie la maintenance du code en permettant des mises à jour asynchrones de l'interface utilisateur sans rechargement de la page. Basé sur des composants réutilisables et écrit en JSX, il offre un DOM virtuel performant, idéal pour les applications à fort trafic.

**Avantages :**

- Facilité de création de composants
- Performance améliorée
- Gestion de l'état centralisée
- Grande communauté de développeurs
- Flexibilité
- Utilisation de React Hooks comme alternative à l'écriture de composants avec des classes, facilitant ainsi l'apprentissage de React

**Inconvénients :**

- Courbe d'apprentissage initiale
- Besoin de connaissances supplémentaires
- Complexité de configuration
- Risque de fragmentation de l'écosystème
- Lourdeur des mises à jour
- Limitation à des solutions front-end
- Documentation limitée

#### 4.1.1.2 AngularJs

Angular est un framework développé par Google, officiellement lancé en 2016. Il est basé sur TypeScript, le distinguant des autres frameworks de cette liste. Conçu pour répondre aux exigences croissantes de la technologie, Angular vise à combler le fossé entre ces exigences et les concepts conventionnels. Il est utilisé pour le développement d'applications web interactives et dynamiques. Angular adopte une approche basée sur les composants, permettant la construction d'interfaces utilisateur modulaires et réutilisables.

**Avantages :**

- Architecture basée sur les composants
- Liaison de données bidirectionnelle
- Gestion de l'état centralisée
- Performance
- Grande communauté de développeurs
- Compatibilité avec les moteurs de recherche

**Inconvénients :**

- Courbe d'apprentissage abrupte
- Syntaxe complexe
- Taille de l'application
- Gestion des mises à jour
- Dépendance à TypeScript
- Performance excessive

#### 4.1.1.3 VueJs

Vue.js est Créé en 2014 par Evan You, Vue.js est un framework front-end open-source reposant sur le modèle de vue. Il permet de diviser l'interface utilisateur en composants réutilisables, favorisant ainsi la maintenabilité et l'évolutivité des projets. Son utilisation d'une syntaxe simple et concise en fait un choix attrayant pour les nouveaux développeurs, facilitant ainsi leur apprentissage.

**Avantages :**

- Facilité d'apprentissage
- Réactivité
- Flexibilité
- Performance
- Composants réutilisables
- Écosystème en expansion

**Inconvénients :**

- Documentation moins complète
- Moins de ressources disponibles
- Limitations pour les applications à grande échelle
- Moins de soutien des grandes entreprises
- Courbe d'apprentissage plus élevée pour les fonctionnalités avancées

#### 4.1.1.4 Etude comparative et résumé

TABLE 2.1 – Tableau comparatif entre les différentes technologies Front-End

	AngularJs	ReactJs	VueJs
Performance	moyen	haute	haute
Scalabilité	haute	haute	faible
Apprentissage	difficile	moyen	facile
Disponibilité des développeurs	haute	haute	faible
Communauté des développeurs	grande	très grande	petite
Acceptation et confiance	haute	très haute	faible

### 4.1.2 Backend

Le backend (ou "côté serveur") est la partie du site Web que vous ne voyez pas. Il est responsable de stocker et d'organiser les données, et garantit le bon fonctionnement côté client. On peut décomposer le Backend en deux parties essentielles : le serveur et la base de données.

#### 4.1.2.1 Serveur

Nous présentons ici des exemples des différents langages de développement :

**Python** : Python est un langage polyvalent utilisé dans une variété de domaines, allant des applications web avancées aux simples scripts, en passant par l'intelligence artificielle. Grâce à sa syntaxe de haut niveau, Python est particulièrement apprécié pour sa facilité d'utilisation.

**Java** : Java est un langage multiplateforme, orienté objet et centré sur le réseau qui peut être utilisé comme une plateforme en soi. C'est un langage de programmation rapide, sécurisé et fiable pour coder tout, des applications mobiles et logiciels d'entreprise aux applications de big data et aux technologies côté serveur.

**Node.js** : Node.js est un environnement d'exécution JavaScript multiplateforme et open-source conçu pour faire fonctionner des applications web en dehors du navigateur du client. Il a été développé par Ryan Dahl en 2009.



FIGURE 2.2 – Les langages de developpement

#### 4.1.2.2 Étude comparative entre les Framework Backend

TABLE 2.2 – Tableau comparatif entre les différentes technologies Back-End

	Java	Python	NodeJs
Rapidité	le plus rapide	rapide	plus rapide
Scalabilité	élevé	moyenne	le plus élevé
Performance	élevé	élevé	faible
Simplicité	simple	plus simple	moyenne
Communauté	grande	grande	grande
Bibliothèque	bien	bien	excellente
Cout	payé	gratuit	gratuit
fonctionnalité transversale	élevé	élevé	élevé

### 4.1.2.3 Base de données

Lorsqu'on doit sélectionner une base de données, le choix crucial réside dans le fait de décider entre une structure de données relationnelle (SQL) ou non relationnelle (NoSQL).

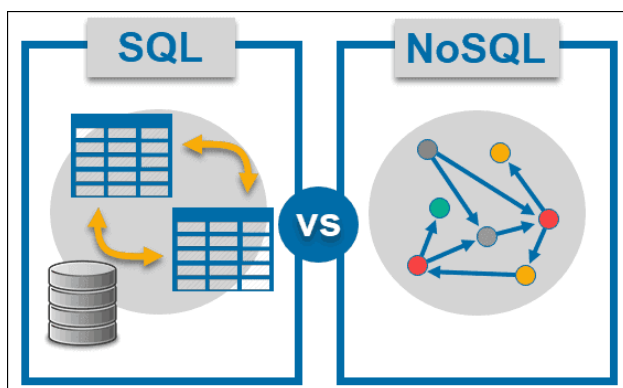


FIGURE 2.3 – Les bases de données SQL et NOSQL

Le tableau ci-dessous dresse une comparaison entre ces deux structures :

TABLE 2.3 – Tableau comparatif entre les bases de données

	SQL	NOSQL
Model	relationnelle	non-relationnelle
Données	structuré	semi structuré
Flexibilité	schéma strict	schéma dynamique
Transaction	ACID	mostly BASE, few ACID
cohérence	fort	éventuellement à fort
Disponibilité	Priorité à la cohérence	basique
échelle	verticale en améliorant le matériel	horizontale par partitionnement des données

-> Les bases de données NoSQL ont été créées en réponse aux limitations de la technologie traditionnelle des bases de données relationnelles. Comparées aux bases de données relationnelles, les bases de données NoSQL sont souvent plus évolutives et offrent des performances supérieures. De plus, la flexibilité et la facilité d'utilisation de leurs modèles de données peuvent accélérer le développement par rapport au modèle relationne

### Choix de MongoDB

MongoDB est une base de données NoSQL open-source, orientée document, conçue pour stocker, interroger et analyser de grandes quantités de données non structurées ou semi-structurées. Elle utilise des schémas dynamiques, ce qui signifie que nous pouvons créer des collections sans avoir à définir préalablement leur structure.

#### Avantages :

- Schémas de documents flexibles : Le modèle de document de MongoDB permet de manipuler facilement des structures de données variées. Son format de données BSON, inspiré de JSON, autorise la présence d'objets dans une collection avec des ensembles de champs différents. Cette souplesse est un avantage précieux pour gérer des données réelles et s'adapter aux changements de besoins ou d'environnement.
- Évolutivité : MongoDB propose des fonctionnalités de partitionnement automatique des données et de sharding, ce qui permet de mettre à l'échelle horizontalement et de gérer la charge supplémentaire. Il prend en charge la mise à l'échelle horizontale en permettant d'ajouter des nœuds pour augmenter la capacité et gérer la charge supplémentaire.
- Vitesse et performance : MongoDB est conçue pour gérer rapidement et efficacement de grandes quantités de données. Contrairement aux bases de données SQL traditionnelles, qui utilisent un modèle tabulaire, MongoDB stocke les données sous forme de documents JSON flexibles. Cette flexibilité permet des opérations de lecture et d'écriture plus rapides, offrant ainsi des performances exceptionnelles, particulièrement bénéfiques pour les projets axés sur l'expérience client. Avec MongoDB, nous avons



pu mettre en place des expériences orientées données complexes, basées sur l'agrégation, avec des performances qui surpassent celles d'une base SQL.



FIGURE 2.4 – MongoDB

#### 4.1.2.4 Classement des bases de données

La Figure 2.5 présente le classement de MongoDB par rapport à d'autres bases de données, tel que présenté par le site zucisystems.com.

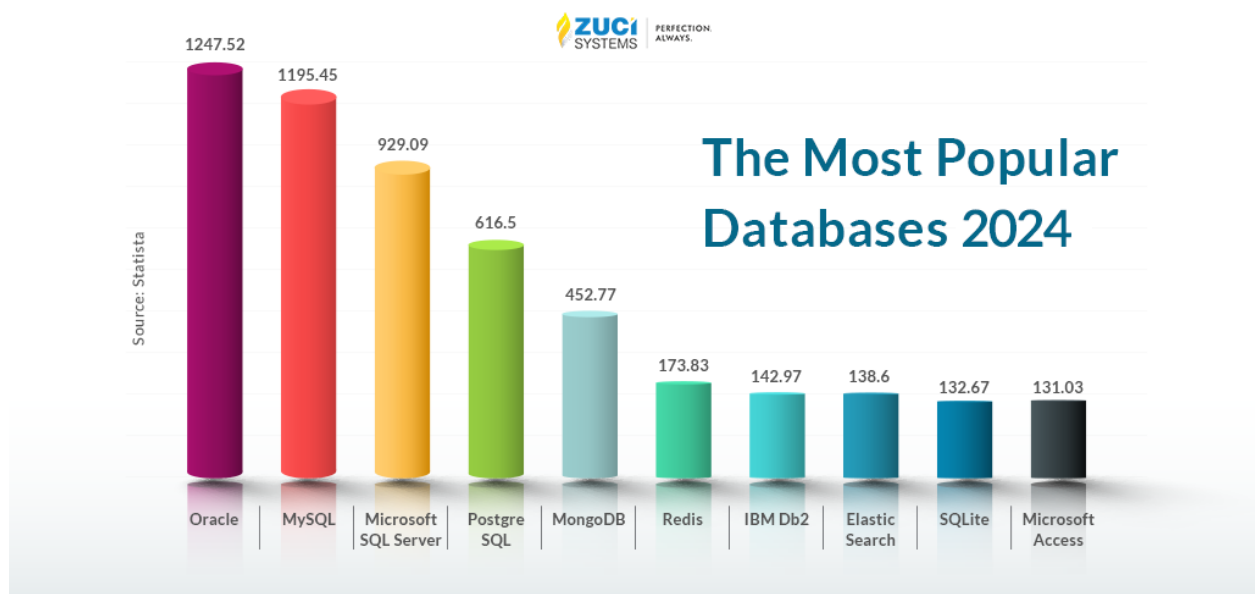


FIGURE 2.5 – Classement des bases de données

#### 4.1.3 Choix technologique de notre application

Nous avons décidé de développer notre application web avec la pile MERN (MongoDB, ExpressJs, ReactJs, NodeJs). Cette pile est largement utilisée pour le développement d'applications web à page unique (SPA) en raison de sa facilité de développement et de sa capacité à gérer les requêtes asynchrones.

# The MERN stack

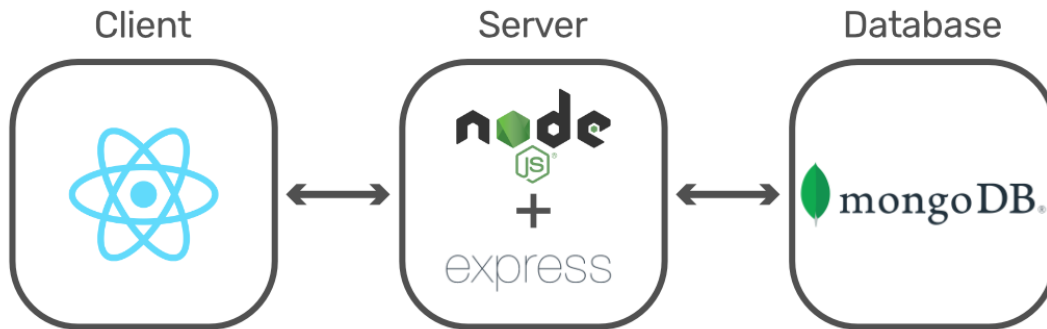


FIGURE 2.6 – Illustration du fonctionnement de la pile Mern

## 4.2 Choix du modèle conceptuel

Avant de commencer le développement de notre application, il est crucial de sélectionner un modèle de conception (pattern design) adapté. Nous avons choisi de considérer les modèles de conception les plus couramment utilisés, à savoir le Modèle-Vue-Contrôleur (MVC) et le Modèle-Vue-Vue-Modèle (MVVM).

### 4.2.1 MVC

Le Modèle-Vue-Contrôleur (MVC) est un patron de conception logicielle couramment utilisé pour le développement des interfaces utilisateur, qui divise la logique du programme en trois éléments interconnectés :

- **Modèle** : C'est l'élément central du patron, le cœur de l'application qui est indépendant de l'interface utilisateur. Il est responsable de la gestion de la logique, des données et des règles de l'application.
- **Vue** : Elle définit l'interface utilisateur telle qu'un graphique, un diagramme ou une table. Cette partie se concentre sur l'affichage des informations à l'utilisateur final.
- **Contrôleur** : Il orchestre le flux de données entre la vue et le modèle. C'est le point logique qui permet de gérer les événements, de gérer différentes conditions et de garantir la synchronisation.

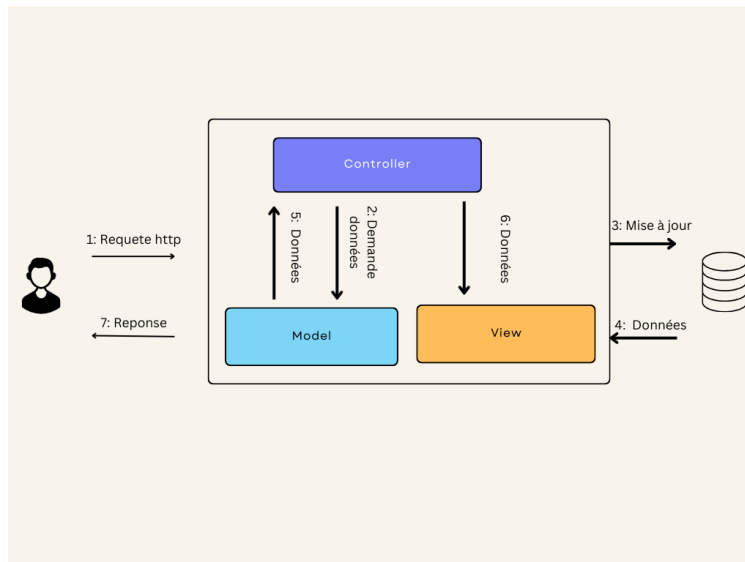


FIGURE 2.7 – Architecture MVC

#### 4.2.2 MVVM

En architecture logicielle, MVVM (Model View ViewModel) est un Design pattern (Modèle de conception) visant à séparer la logique de présentation d’une application en 3 couches :

- **Model** : Le modèle représente une collection de classes qui explique le modèle métier et le modèle de données. Il définit également les règles métier.
- **View (Vue)** : La vue est la description de l’interface graphique, elle fait le lien entre les actions de l’utilisateur et le modèle de vue. Elle définit où et comment sont placés les composants graphiques sur l’interface et décrit les liaisons de données (Data Bindings) entre les valeurs affichées et le modèle de vue.
- **ViewModel (Vue Modèle)** : Le modèle de vue est chargé de transformer et organiser les modèles métiers afin d’exposer les données à afficher par la vue.

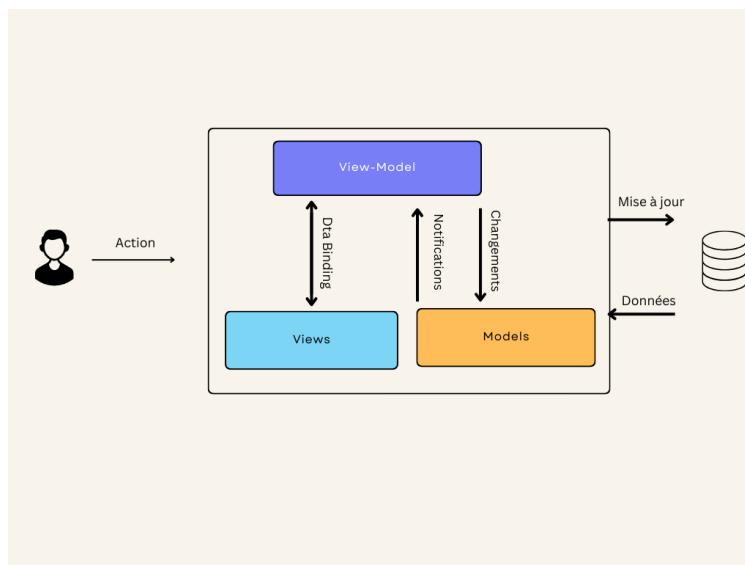


FIGURE 2.8 – Architecture MVVM

### 4.3 Notre choix

Après analyse, nous avons opté pour l'architecture MVC pour le développement de notre plateforme, car elle permet une séparation claire entre la logique métier et la présentation. Cette séparation facilite l'ajout et la modification du code sans perturber le reste de l'application, offrant ainsi une structure robuste et flexible.

## Conclusion

Dans ce chapitre, nous avons identifié les intervenants impliqués dans notre projet, ainsi que les besoins fonctionnels, non fonctionnels et techniques. Après avoir analysé ces exigences et choisi les technologies à utiliser, le chapitre suivant sera réservé à la conception.