# CHAPITRE I : CONTEXTE ET OBJECTIFS DU PROJET

# 

# Introduction

Dans ce chapitre introductif, la première partie vise à présenter l'organisme d'accueil

« EDI-TUNISIE », son savoir-faire et ses qualités. Quant à la seconde partie, elle sera consacrée à l'idée générale de notre projet et aux problématiques qui motivent l'organisme à mener à bien cette candidature. Par la suite, nous examinons les solutions existantes à travers leurs critiques et présentons les solutions proposées. Enfin, nous terminons le chapitre par une analyse des différentes méthodes de développement et une comparaison pour choisir la méthode la plus adaptée à l'implémentation de notre plateforme.

# 1Contexte général

Cette partie présente le cadre général du projet ainsi que ses principaux objectifs.

1.1 Cadre du projet

Le présent travail s’inscrit dans le cadre du projet de fin d’étude en vue de l’obtention du Diplôme de Licence en Informatique délivré par l’Institut Supérieur d’Informatique et de Mathématiques de Monastir (ISIMM). Notre stage s'est déroulé au sein de la société " « EDI-TUNISIE " pour une durée de quatre mois du 01 février au 31 mai 2023.

1.2 Présentation de l’entreprise d’accueil

EDI-Tunisie éditrice de solutions logicielles pour le domaine des assurances. , voir Figure I.1, est basée à Tunis et comporte une équipe dynamique, des consultants et un large réseau de partenaires permettant d’offrir un service informatique global dans différents domaines en utilisant la majorité des technologies connues du marché. Elle a pour métier un service informatique de qualité auprès des PME/PMI, à très forte valeur ajoutée misant sur l’innovation et l’expertise. La société qui s’engage à garantir la satisfaction de ses clients.

|  |
| --- |
| Raison sociale EDI-Tunisie |
| Adresse Avenue de Carthage Tunis |
| Téléphone 71 256 237 /94 226 213 |
| Courriel hichem.boudegga @ edi-tunisie.net |
| Site web edi-tunisie.net |
| Compétences SQL Server 2019 Developper  Entity framework Core  Dotnet 8 , ASP.NET Core Blazor (VS 2022) |

Tableau I. 1 : Identité de la société

EDI est une société internationale, éditrice de solutions logicielles pour le domaine des assurances.



## 1.3 Idée maitresse

**Face à un environnement en constante évolution, marqué par une croissance et une évolution technique et scientifique rapides, l'entreprise doit impérativement établir une stratégie s'appuyant sur ses moyens techniques et humains. Parmi les activités clés figure la gestion des achats, et plus particulièrement celle par appel d'offres.**

**L'utilisation de solutions informatiques pour la gestion des appels d'offres gagne en popularité, notamment auprès des entreprises qui cherchent à réduire leurs coûts. En effet, les appels d'offres permettent d'effectuer des achats plus adaptés aux besoins spécifiques, de stimuler la concurrence et de négocier des prix plus avantageux.**

# 2 Problématique

# 3Etude de l'existant

Avant de réaliser un projet, il est essentiel de mener une étude préalable pour évaluer les forces et les faiblesses des systèmes existants ainsi que les besoins du client. Cette section présente une étude basée sur l'analyse de différentes applications web spécialisées dans le domaine gestion des demandes des appels d’offres en Tunisie et à l’étranger. Cette analyse nous permet d'identifier les avantages et les inconvénients de chaque système afin de déterminer les besoins et de les prendre en compte dans la conception et la réalisation de notre projet.

## 3.1 Solutions Existantes

### 3.1.1 Solutions en Tunisie

### 3.1.2 Solution à l’étranger

## 3.2 Critique de l’existant

# 4Solution proposée

# 5Méthodologie de développement

La méthodologie de développement est un cadre qui permet de structurer, planifier et contrôler le processus de développement d'une application. Elle consiste à modéliser le système avant sa réalisation pour en comprendre le fonctionnement et garantir sa cohérence. L'utilisation de modèles permet de réduire les coûts et les délais de développement, tout en assurant un niveau de qualité élevé pour le produit final. En effet, un modèle bien conçu est un élément clé pour assurer une maintenance efficace du produit dans le temps.

## 5.1 Étude de Méthodologie de développement

Avec la multitude de méthodes de développement disponibles, il est devenu difficile de faire un choix. Afin de sélectionner la méthode la plus adaptée, nous avons étudié plusieurs méthodologies de développement objet. Après analyse, nous avons opté pour celle qui répond le mieux à nos besoins.

## 5.2 Présentation de quelques méthodologies

### 5.2.1 Agile

La méthode Agile est une approche itérative et incrémentale de la gestion de projet qui permet de mieux s'adapter aux changements et aux évolutions du projet tout au long de son cycle de vie. Elle favorise la collaboration et la communication entre les membres de l'équipe de développement, ainsi que la réactivité aux feedbacks des utilisateurs. Cette

méthode est souvent utilisée pour des projets de développement de logiciels, car elle permet d'optimiser le processus de création de logiciels en fournissant des résultats plus rapides et plus pertinents, tout en limitant les risques et les coûts.

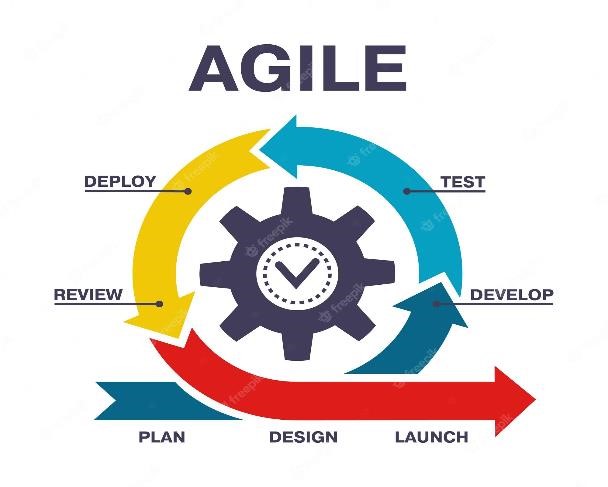


Figure I. 6 : Les étapes du processus Agile (5

#### 5.2.1.1 Scrum

Scrum est une méthodologie de gestion de projet Agile spécifiquement utilisée dans le domaine du développement logiciel. Elle se base sur un processus itératif et collaboratif pour organiser les tâches à réaliser par l'équipe de développement. Le processus de Scrum se déroule en cycles appelés sprints, qui durent généralement entre deux et quatre semaines.

Au début de chaque sprint, l'équipe de développement définit les tâches à réaliser et les ajoute à un tableau appelé tableau de bord Scrum (ou "Scrum board"). Pendant le sprint, l'équipe se réunit quotidiennement pour discuter de l'avancement du projet et des éventuels obstacles rencontrés, lors d'une réunion appelée "Daily scrum".

À la fin du sprint, l'équipe présente les résultats de leur travail, sous forme de fonctionnalités ou d'éléments de travail finalisés, lors d'une revue de sprint. Enfin, l'équipe se réunit pour une rétrospective, qui permet de discuter des aspects positifs et négatifs du sprint et d'identifier des pistes d'amélioration pour le prochain sprint.

La méthode Scrum permet d'optimiser le processus de développement en favorisant la communication et la collaboration entre les membres de l'équipe, tout en permettant une plus grande flexibilité et une réactivité accrue aux changements et aux feedbacks des utilisateurs.

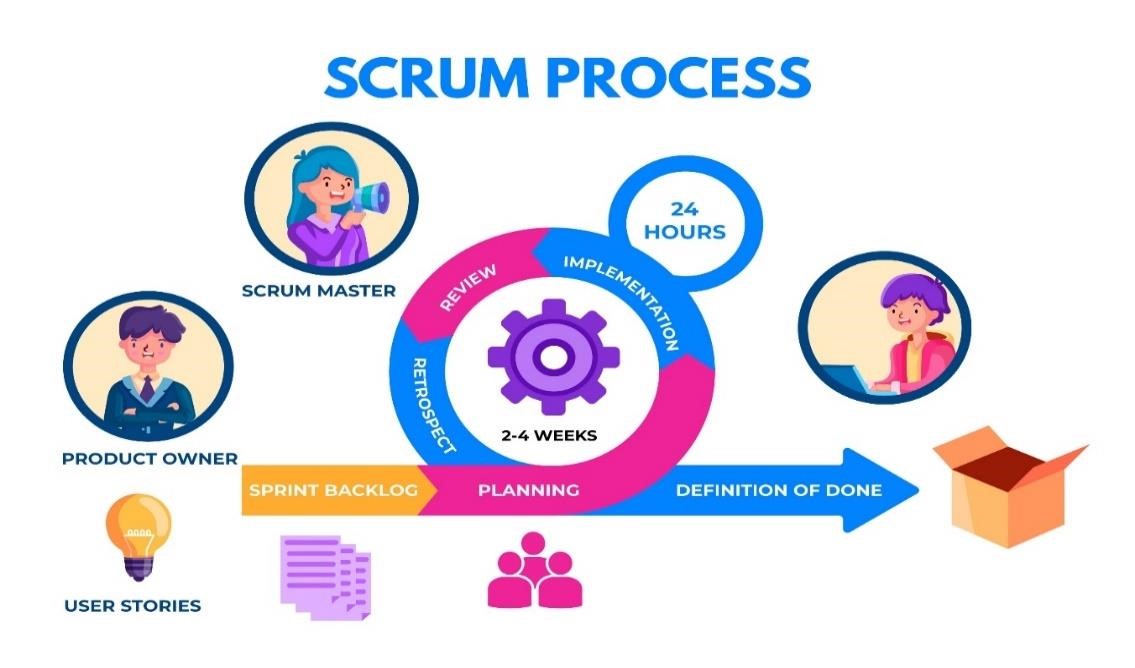


Figure I. 7 : Les étapes du processus Scrum (6)

#### 5.2.1.2 La méthode XP

La méthode XP (eXtreme Programming) est une approche de développement logiciel agile qui met l'accent sur la qualité, la flexibilité et la collaboration au sein de l'équipe de développement. XP a été créé dans les années 1990 par Kent Beck et est basé sur une série de principes et de pratiques. Voici certains aspects clés de la méthode XP :

1. Communication constante
2. Tests automatisés
3. Programmation en binôme
4. Intégration continue
5. Planification itérative

6.

Client impliqué

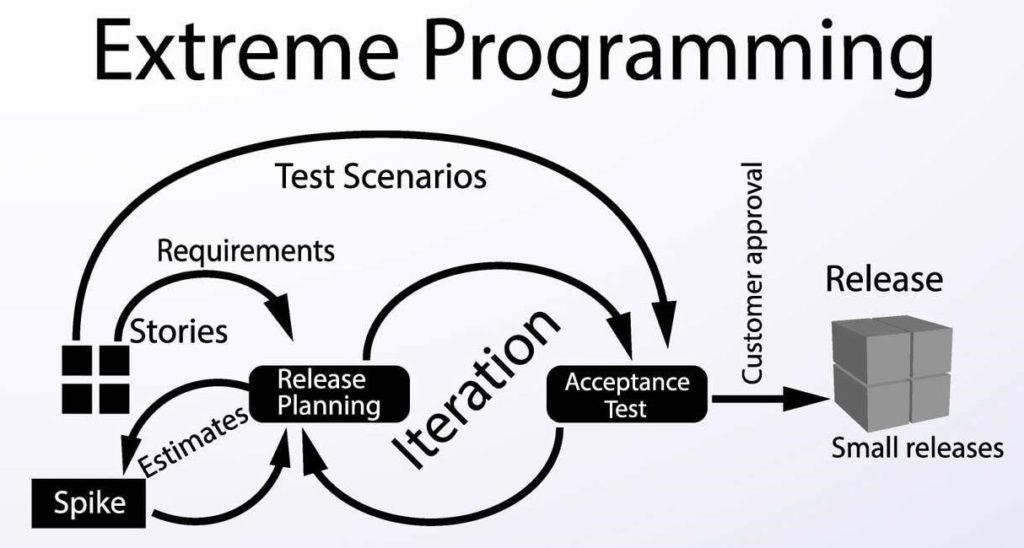


Figure I. 8 : principe et fonctionnement de la méthode XP

### 5.2.2 Processus unifié

Le Processus Unifié, voir Figure I.10, est une méthodologie de gestion de projet de développement logiciel qui se concentre sur la production d'un produit de qualité dans un délai et un budget défini. Elle se base sur un processus itératif et incrémental qui s'adapte aux besoins et aux changements du projet tout au long du cycle de vie.

Le Processus Unifié est divisé en quatre phases principales : l'Inception, l'Élaboration, la Construction et la Transition. Chaque phase est subdivisée en itérations ou sprints, qui sont des périodes courtes (2 à 6 semaines) de travail intensif sur un ensemble de fonctionnalités à développer.

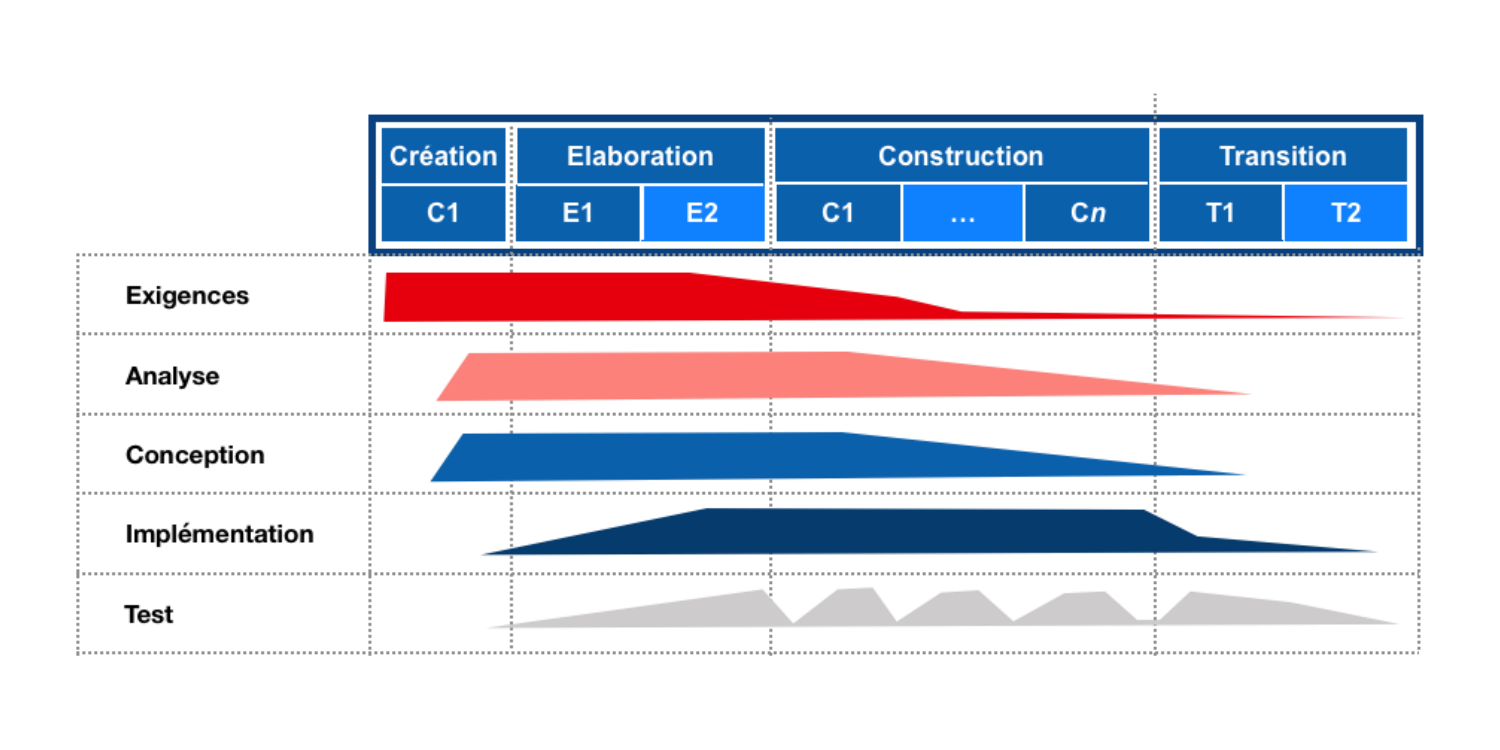


Figure I. 9 : Les étapes du processus unifié (7)

#### 5.2.2.1 Two Track Unified Process (2TUP)

Le processus unifié Two Track Unified Process (2TUP) se base sur le principe que toute évolution imposée à un logiciel peut être décomposée et traitée en parallèle, en suivant deux axes distincts : l'axe fonctionnel et l'axe technique. La réalisation du logiciel consiste ensuite à fusionner les résultats de ces deux branches du processus.

#### 5.2.2.2 Rational Unified Process (RUP)

Le Rational Unified Process (RUP) est un processus de développement de logiciel qui a été créé par Rational Software Corporation, qui a ensuite été acquis par IBM. Le RUP est un processus itératif et incrémental qui se concentre sur l'implémentation de fonctionnalités du logiciel à travers des cycles de développement courts et successifs. Il est basé sur une approche de développement centrée sur l'architecture et sur des principes de modélisation UML (Unified Modeling Language). Le RUP propose des phases, des activités et des rôles définis pour chaque membre de l'équipe de développement, et il met l'accent sur la communication et la collaboration entre les membres de l'équipe.

## 5.3 Étude comparative et évaluation

|  |  |  |
| --- | --- | --- |
| Méthodologies | Avantages | Inconvénients |
| Agile | Agile présente de nombreuses opportunités pour engager les  parties prenantes et les équipes. En impliquant le client à chaque étape du projet, cela favorise une forte collaboration entre le client et l'équipe de projet, permettant à l'équipe de mieux comprendre la vision du client. | * L'intégration des tests et des opérations n'a pas été aussi réussie que prévu dans ce mélange. * La technologie n'a pas été suffisamment mise en avant, ce qui peut rendre difficile la vente du concept aux cadres supérieurs qui ne comprennent pas le rôle que la culture joue dans le développement de logiciels. |
| Scrum | * Le projet est centré sur le client. * Esprit d'équipe favorisé. Communication ouverte et transparente. * Priorité donnée à la simplicité, l'efficacité et la qualité. * Capacité d'adaptation aux changements. * Avancement basé sur des résultats tangibles. | Le manque de pratique pour atteindre l'état "DONE" dans le contexte logiciel peut être un inconvénient de Scrum. |
| XP | -Communication accrue entre les membres de l'équipe.  - Adaptabilité aux changements fréquents. - Qualité du code améliorée grâce aux tests automatisés. | -Peut nécessiter une équipe hautement qualifiée et expérimentée.  - Peut être moins adapté aux projets avec des exigences strictes et prédictibles. |
| Unified  Process | * Il est basé sur une approche itérative et incrémentale. - Il met l'accent sur la gestion des risques. * Il repose sur des pratiques   éprouvées pour le développement et la maintenance de logiciels de qualité.   * Il permet d'obtenir un niveau de portabilité plus élevé. | * La mise en place de ce système   nécessite l'intervention d'experts.   * La personnalisation de ce système peut s'avérer coûteuse. * Très axé processus. |
| RUP | - Assure la traçabilité de l'ensemble du processus, depuis les cas d'utilisation jusqu'au déploiement. - Met l'accent sur l'architecture logicielle. - Gère efficacement les risques liés aux projets. - Favorise la réutilisation des composants logiciels. | - Coût de personnalisation souvent élevé. - Processus très axé. - Vision non évidente ni immédiate. |
| 2TUP | * Adaptable à différents types de projets. * Permet une réponse rapide aux besoins des utilisateurs. * Structuration et modélisation de toutes les étapes du développement de logiciel. * Génère un modèle centré sur les besoins métiers des utilisateurs finaux. | Pas de documents types. |

Tableau I. 3 : Tableau comparatif des méthodologies de développement

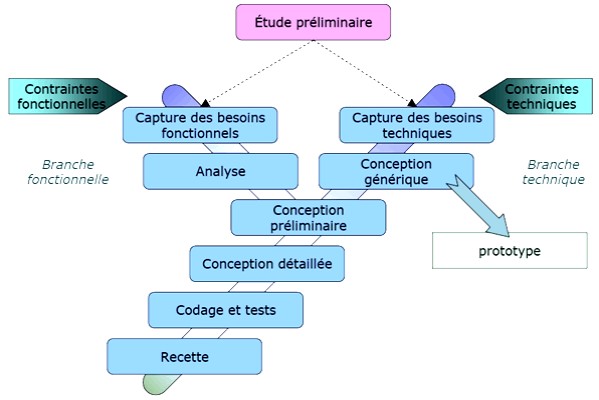
## 5.4 Choix de la méthodologie de travail

Notre projet repose sur un processus de développement rigoureux, qui commence par l'analyse des besoins fonctionnels du système et se poursuit jusqu'à la conception et la programmation finale. Pour cela, nous avons besoin d'un cycle de développement qui sépare les aspects techniques des aspects fonctionnels, en commençant par une étude préliminaire. Après une étude comparative approfondie, nous avons choisi la méthode 2TUP, car elle propose une approche innovante qui répond à nos besoins spécifiques tout en respectant le cadre de notre projet.

➢ Mise en pratique du processus 2TUP

2TUP est un processus unifié (c’est-à-dire construit sur UML, itératif, centré sur l’architecture conduit par les cas d’utilisation). Les étapes de ce processus sont clairement représentées dans la Figure I.11.

Figure I. 10 : Les étapes du processus 2TUP



Le principe fondateur du 2TUP est que toute évolution imposée à un logiciel peut se décomposer et se traiter parallèlement, suivant un axe fonctionnel et un axe technique.

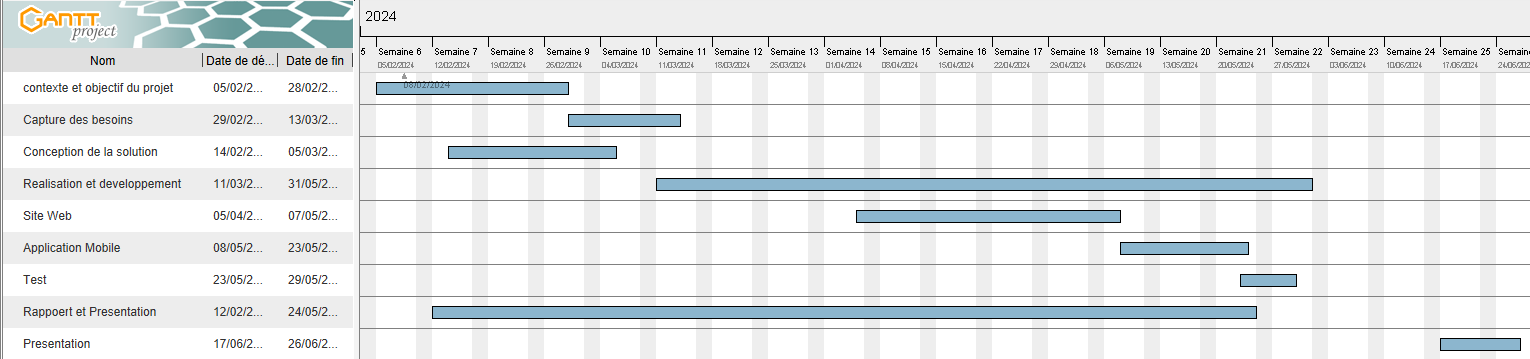
La réalisation du logiciel consiste à fusionner les résultats de ces deux branches du processus. Selon le processus 2TUP que nous suivons, nous passons obligatoirement par les phases suivantes à la suite de notre travail :

1. L’étude préliminaire : qui contient une description du service et les cas d’utilisation principaux, c’est une première version de la spécification générale.
2. La capture des besoins fonctionnels : qui définit le quoi faire à travers une spécification générale qui décrit le service à développer d’un point de vue fonctionnel et une spécification détaillée qui précise les traitements qui concernent chaque scénario des cas d’utilisation présent en spécification générale tout en respectant les contraintes fonctionnelles et non fonctionnelles.
3. L’analyse : où on effectue simultanément l’étude des données et l’étude des traitements à effectuer.
4. La capture des besoins techniques : qui permettent de satisfaire les contraintes techniques présentes dans le cahier des charges et donc répondre aux attentes de client.
5. La conception générique : qui définit le comment faire.
6. La conception détaillée : qui précise l’implémentation technique de l’application. Elle consiste en la fusion de la spécification détaillée et de la conception générique, pour déterminer comment faire le quoi faire dans le détail.

On y trouve le schéma de base de données, les diagrammes de classes et les diagrammes de séquence supplémentaires qui détaillent les interactions entre les composants du logiciel.

1. Le Codage et les Tests : qui décrivent comment est l’application est réalisée et installée sur un environnement d’exécution et comment la stratégie de validation est effectuée.

## 5.5 Diagramme de Gantt



# Conclusion

Au cours de ce premier chapitre, nous avons effectué une analyse préliminaire pour entamer notre projet avec une vision claire de la solution à développer. Nous avons examiné différentes approches de développement et nous avons finalement opté pour la méthodologie 2TUP pour la réalisation de notre projet. Dans le prochain chapitre, nous aborderons la phase de capture des besoins.

# CHAPITRE II : CAPTURE DES BESOINS

# Introduction

Le choix de la méthodologie 2TUP a été effectué en tenant compte de ses avantages en termes de développement de logiciels de qualité. Ce chapitre est donc consacré à la phase de capture des besoins, qui constitue une étape cruciale dans le processus de développement. Nous nous concentrerons sur l'identification des fonctionnalités de notre futur système en recensant les besoins fonctionnels, les exigences traduites par les besoins non fonctionnels et les besoins techniques. La collecte de ces informations est essentielle pour garantir une description sans ambiguïté du portail à développer et pour s'assurer que les attentes des utilisateurs finaux seront pleinement satisfaites.

# 1 Besoins fonctionnels

Fonctionnalités d'un système de gestion des appels d'offres (AO)

Saisir et gérer des dossiers d'appel d'offres (AO)

• Typologie (AO, demande de devis, ou consultation) : Le système permet de classer les

demandes en fonction de leur type, comme les appels d'offres (AO), les demandes de devis

ou les consultations.

• Plusieurs demandes de devis (différents branches, catégories) : Le système permet de

gérer plusieurs demandes de devis pour différentes branches ou catégories d'assurance.

• Calendrier des dates clefs (délais de réponse, délais interne à la compagnie) : Le

système permet de suivre les dates clés, telles que les délais de réponse des assureurs et

les délais internes de la compagnie pour la préparation des devis.

Rattachement des documents

• Fournis par le client : Le système permet de stocker et de gérer les documents fournis par

le client, tels que les cahiers des charges et les demandes de renseignements.

• Préparé par les gestionnaires : Le système permet de stocker et de gérer les documents

préparés par les gestionnaires, tels que les devis et les propositions.

Détail fiche client

• Le système fournit des informations détaillées sur les clients, telles que leurs coordonnées,

leur historique d'assurance et leurs besoins spécifiques.

Suivi des offres et avancement

• Les agents apporteurs et la Direction Commerciale peuvent suivre et visualiser les

offres et leurs avancements dans le temps : Le système permet aux agents apporteurs et

à la Direction Commerciale de suivre l'état des offres et leur avancement dans le temps.

• Chaque demande Appel Offre (AO) passe par un cycle de vie : Nouvelle (saisie), Devis

demandé, Devis en cours de préparation, Devis Validé, Devis envoyé au client,

Réception réponse (accepté / refusé) : Le système suit le cycle de vie des demandes

d'appel d'offres, de leur création à leur conclusion.

Gestion des délais et alertes

• Gérer les délais des dates clefs, alertes et relances : Le système permet de gérer les

délais clés, les alertes et les relances, garantissant que les demandes d'appel d'offres sont

traitées dans les délais.

Affectation des gestionnaires

• Chaque AO est affecté à un gestionnaire production (ou un service) afin de préparer

un devis (demandé par la direction commerciale) : Le système permet d'affecter les

demandes d'appel d'offres à des gestionnaires ou à des services spécifiques pour la préparation

des devis.

Préparation et validation des devis

• Le gestionnaire production prépare un devis, et l’envoi à la direction (saisie sur

système, GED, ...) : Les gestionnaires peuvent préparer des devis et les envoyer au

service commercial pour validation et approbation

• La direction commerciale peut accepter, refuser, demande des rectifications : La

direction commerciale peut examiner les devis, les accepter, les refuser ou demander des

rectifications.

• La direction commerciale valide le devis final, et l’envoi au client : La direction

commerciale valide le devis final et l'envoie au client.

Suivi des résultats

• Suivi du résultat client : Le système permet de suivre les résultats des demandes d'appel

d'offres, telles que les acceptations ou les refus.

Tableau de bord de suivi

• Dashboard de suivi des demandes AO : Le système fournit un tableau de bord pour

suivre l'état et l'avancement des demandes d'appel d'offres

# 2 Besoins non fonctionnels

Sécurité o Accès sécurisé aux données et aux fonctionnalités o Protection

contre les accès non autorisés et les cyberattaques o Protection des données

personnelles et respect de la vie privée

• Disponibilité o Disponibilité élevée du système pour garantir l'accès

aux données et aux fonctionnalités en tout temps o Temps de réponse

rapides pour les requêtes et les transactions

• Évolutivité o Capacité à gérer un nombre croissant de demandes d'appel

d'offres et d'utilisateurs o Possibilité d'ajouter de nouvelles fonctionnalités

et de s'intégrer à d'autres systèmes

• Fiabilité o Fonctionnement fiable du système avec un minimum de temps

d'arrêt ou d'erreurs o Traitement précis et cohérent des données

• Facilité d'utilisation o Interface utilisateur intuitive et conviviale o Navigation

et recherche faciles des informations

• Documentation o Documentation complète et à jour pour les utilisateurs et les administrateurs

.La portabilité : L'application doit être compatible avec tous les navigateurs courants, sans aucune restriction de version ou de plateforme. Il est également crucial qu'elle soit conçue pour s'exécuter sans problème sur les appareils Android et sur toutes les tailles d'écran, afin d'offrir une expérience utilisateur cohérente et optimale.

➢ Performance : L'application doit fournir tous les statuts et informations en temps réel et d'une manière optimale.

• Temps de réponse minimal : En effet, vu le nombre important des transactions quotidiennes, il est impérativement nécessaire que la durée d'exécution des traitements s'approche le plus possible du temps réel.

• Chargement rapide de l'application : le rafraîchissement des données doit être en temps réel, souple, et rapide

Conclusion

La phase de capture des besoins est essentielle pour garantir le développement d'un système qui répond aux attentes des utilisateurs finaux. En utilisant la méthodologie 2TUP, nous avons identifié les besoins fonctionnels, non fonctionnels et techniques du système de gestion des appels d'offres.

Les besoins fonctionnels décrivent les fonctionnalités spécifiques que le système doit fournir, tandis que les besoins non fonctionnels définissent les qualités globales du système, telles que la sécurité, la disponibilité et la facilité d'utilisation. Les besoins techniques spécifient les exigences technologiques nécessaires pour mettre en œuvre le système.

En capturant ces besoins de manière approfondie, nous avons établi une base solide pour la conception et le développement du système. Cela nous permettra de créer un système qui répond aux besoins de l'entreprise et offre une expérience utilisateur optimale.

CHAPITRE III :

# 1Besoins techniques

Nous nous concentrons sur la phase de capture des besoins techniques, qui vise à capitaliser le savoir-faire technique. Pour expliquer nos choix technologiques, nous avons mené une étude comparative des différentes technologies qui pourraient être utilisées dans notre projet.

## 1.1 Choix technologique

Dans la section suivante, nous allons présenter les Framework FrontEnd et Back-End que nous avons choisis pour le développement de notre site web.

### 1.1.1 Front-End

Parmi les choix de technologies pour le développement web, nous avons opté pour Blazor en tant que framework pour notre projet. Blazor est un framework open-source de Microsoft qui permet de construire des applications web interactives et riches en utilisant C# et .NET. Ce framework se distingue par sa capacité à exécuter du code C# côté client, offrant ainsi une expérience de développement familière aux développeurs .NET.

Avantages :

* Open Source & free
* use any library
* Utilization of C#
* Faster than JavaScript

Inconvénients :

* Initial Load Time
* Larger Download Size
* Limited Debugging Capability
* Server Connection Dependency

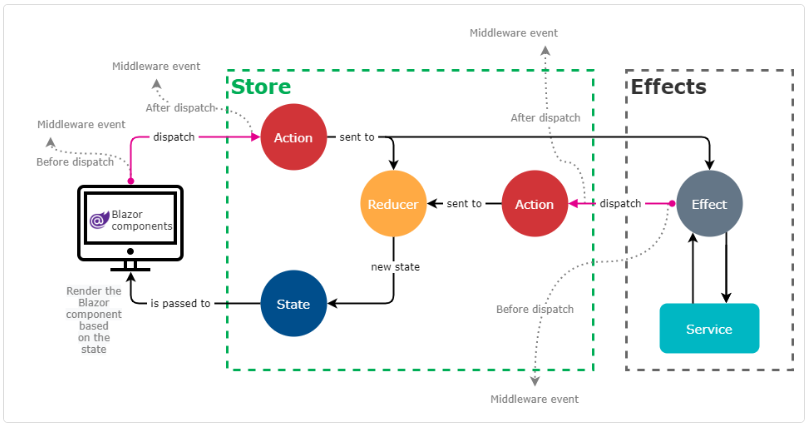


Figure II. Architecture d'une application utilisant Blazor

### 1.1.2 Back-End

Le Back-End, qui est invisible pour le client, représente une grande partie du développement d’un projet web. On peut décomposer le Back-End en deux parties essentielles :

* Serveur
* Base de données

#### 1.1.2.1 Serveur

Parmi les différents langages de programmation côté serveur, nous avons choisi SQL Server

Avec son Framework **Entity Framework (EF)** pour notre projet . **SQL Server** est une plate-forme de données d’entreprise permettant de gérer et stocker dans des bases de données tout type d’information :

* Données structurées : données relationnelles par exemple.
* Données non structurées : documents, images, …
* Données géospatiales

Bien que d'autres langages tels que Python, Java, java-script et PHP soient également largement utilisés pour les applications web côté serveur, Notre choix s'est porté sur SQL Server en raison de sa grande capacité de mise à l'échelle et de sa rapidité d'exécution.

#### 1.1.2.2 Base de données

Une base de données permet de stocker et de retrouver l'intégralité de [données brutes](https://fr.wikipedia.org/wiki/Donn%C3%A9es_brutes) ou d'[informations](https://fr.wikipedia.org/wiki/Information) en rapport avec un thème ou une activité. Lorsqu'il s'agit de choisir une base de données, la décision la plus importante est de choisir une structure de données relationnelle (SQL) ou non relationnelle (NoSQL). Le tableau ci-dessous dresse une comparaison entre ces deux structures :

|  |  |  |
| --- | --- | --- |
|  | BD Relationnelle | BD NoSQL |
| Type | Relationnelle | Non relationnelle |
| Données | Structurées, stockées dans des tables | Non structurées stockées dans des fichiers JSON |
| Schéma | Prédéfini, rigide | Non prédéfini, flexible |
| Évolutivité | Verticale (augmenter la charge sur un seul serveur) | Horizontale (partage ou l'ajout de serveurs) |
| Transaction | ACID | Théorème CAP |

Tableau II. 1 : Bases de données Relationnelles vs NoSQL

Selon cette comparaison, les systèmes de base de données Microsoft SQL Server ont l'avantage de fournir une grande évolutivité verticale et une capacité de stockage robuste sans sacrifier les performances. Ils permettent de stocker différents types de données, qu'elles soient structurées, non structurées ou géospatiales, ce qui les rend plus adaptés aux besoins actuels des entreprises en matière de gestion des données.

Nous avons donc choisi SQL Server comme base de données car il offre une grande robustesse pour le stockage de divers types de données et peut facilement être mis à l'échelle verticalement pour gérer de grands volumes de données

#### 

#### 3.1.2.3 Classement des bases de données

La Figure II.2 présente le classement de Microsoft SQL Server par rapport d’autres BDD présentée par le site DB-Engines.com.

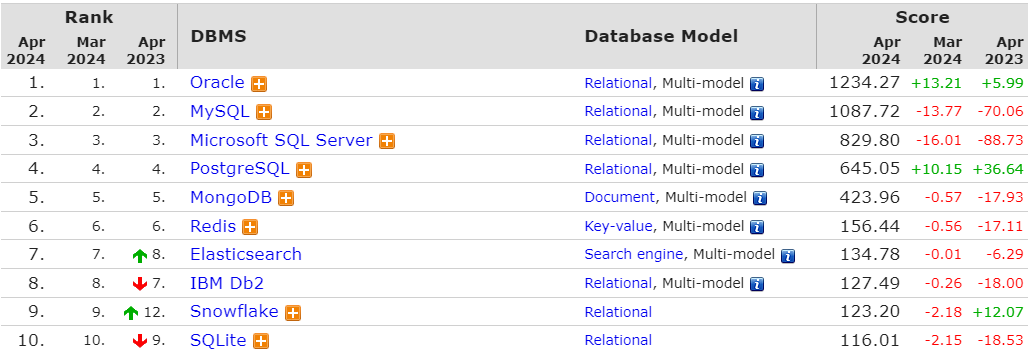


Figure II. 2 : Classement des bases de données (10)

## 3.2 Choix architectural

Avant de commencer le développement de notre projet, il est important de choisir un modèle architectural adapté à nos besoins. Parmi les modèles les plus couramment utilisés, nous avons choisi de nous concentrer sur deux d'entre eux : le modèle Modèle-Vue-Vue Modèle (MVVM) et le modèle Modèle-Vue-Contrôleur (MVC).

Le ViewModel communique avec le Model pour récupérer et enregistrer les données. Il expose ensuite ces données sous forme d'objets, qui sont liés aux contrôles de l'interface utilisateur de manière déclarative. Cela signifie que toute modification de l'état du ViewModel est immédiatement reflétée dans l'interface utilisateur, sans que celle-ci ait besoin de gérer les détails de la mise à jour des données.

La Figure II.4 suivante illustre la structure générique d’une architecture MVVM.

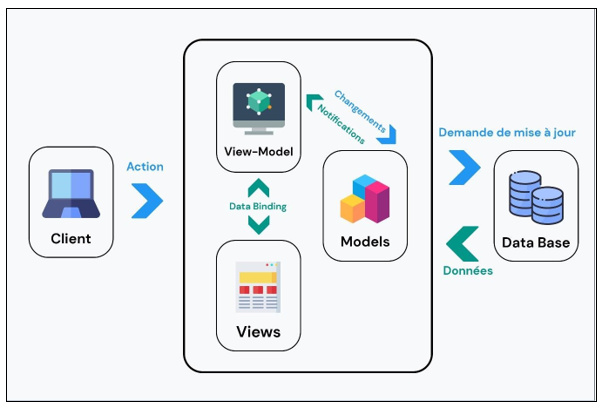


Figure II. Architecture MVVM

### 3.2.2 MVC

MVC (Modèle-Vue-Contrôleur) est un modèle de conception qui permet de séparer les responsabilités dans une application en trois parties distinctes :

* Le modèle : représente les données et leur traitement.
* La vue : représente l'interface utilisateur, c'est-à-dire ce que l'utilisateur voit à l'écran et avec quoi il interagit.
* Le contrôleur : gère les interactions entre la vue et le modèle, et est responsable de la coordination du flux de l'application.

L'idée derrière MVC est de séparer les différentes responsabilités de manière à rendre l'application plus facile à maintenir et à évoluer. En effet, si l'on change le modèle, cela n'affectera pas la vue ou le contrôleur, et vice versa.

Le modèle MVC est souvent utilisé dans le développement de sites web, mais il peut également être utilisé pour d'autres types d'applications. La Figure II.5 suivante illustre la structure générique d’une architecture MVC.

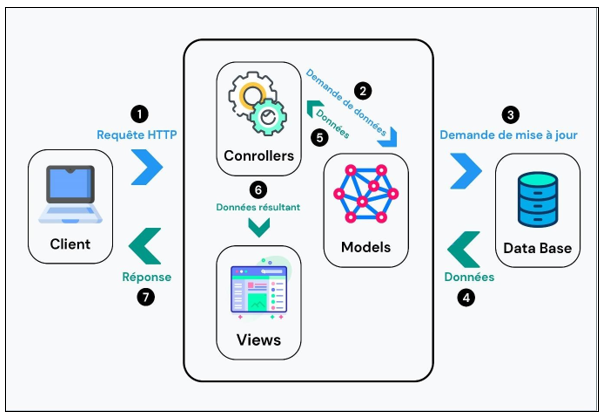


Figure II. 5 : Architecture MVC

### 3.2.3 Architecture retenue

D’après l'étude comparative réalisée, nous avons choisi de travailler avec l'architecture MVC pour notre projet. Cette architecture est adaptée à la nature et à la taille de notre application, car elle permet une séparation claire entre la logique de l'application et la partie présentation. Cette séparation facilite l'ajout et la modification du code d'une section sans affecter le reste de l'application. Le patron MVC convient particulièrement bien pour le développement des applications de taille moyennes et petites.

# Conclusion

Au cours de ce chapitre, nous avons appliqué le processus 2TUP pour la capture et la gestion des besoins de notre projet. Nous avons classé les différents besoins en trois catégories : fonctionnelles, non fonctionnelles et techniques. Après avoir analysé les exigences et choisi les technologies à utiliser, il est maintenant temps de passer à la conception de notre application dans le prochain chapitre.

# CHAPITRE III : CONCEPTION

Introduction

Dans ce chapitre, nous passons à la phase de conception de notre application web. Dans la première partie, nous allons aborder la conception graphique de notre application web. Nous allons également présenter la conception générique en utilisant le formalisme UML qui est basé sur les diagrammes dans la deuxième partie. Enfin, la troisième partie sera consacrée à la réalisation des différents diagrammes UML nécessaires à la conception de notre application.

# Conception préliminaire des interfaces – Prototypes

Une maquette est un prototype temporaire qui offre aux utilisateurs une représentation concrète mais non définitive de l'interface future de l'application. Elle est conçue rapidement pour améliorer la communication entre le développeur et le client. Les maquettes de notre application sont les suivantes :

# Conception générique

La phase de conception a pour objectif de décrire de manière précise et détaillée le fonctionnement futur du système en utilisant généralement un langage de modélisation. Cette étape est cruciale pour faciliter la réalisation du système en fournissant une représentation claire et structurée de toutes les fonctionnalités et interactions requises.

## 2.1 Le langage UML

Nous avons opté pour l'utilisation du langage de modélisation unifié «UML» (Unified Modeling Language) afin de faciliter notre tâche de conception. Ce langage standard permet de modéliser un problème de façon précise et est devenu une référence en termes de modélisation objet.

Les avantages de l'utilisation d'UML sont multiples :

* Il permet un gain de temps et encourage l'utilisation d'outils spécifiques, ce qui garantit la stabilité de notre application.
* Il permet de cadrer l'analyse et facilite la compréhension de représentations abstraites complexes.
* UML est normalisé et formel, ce qui le rend polyvalent et universellement reconnu.

## 2.2 Les diagrammes d'UML

UML est composé de treize types de diagrammes, chacun dédié à la représentation de concepts particuliers d'un système logiciel. Ces diagrammes sont regroupés en deux grandes catégories :

1. Sept diagrammes comportementaux :
   * + Diagramme de cas d'utilisation : Il montre les interactions fonctionnelles entre les acteurs et le système à l'étude.
     + Diagramme de vue d'ensemble des interactions : Il fusionne les diagrammes d'activité et de séquence pour combiner des fragments d'interaction avec des décisions et des flots.

Diagramme de séquence : Il montre la séquence verticale des messages passés entre objets au sein d'une interaction.

* + - Diagramme de communication : Il montre la communication entre objets dans le plan au sein d'une interaction.
    - Diagramme de temps : Il fusionne les diagrammes d'états et de séquence pour montrer l'évolution de l'état d'un objet au cours du temps.
    - Diagramme d'activité : Il montre l'enchaînement des actions et décisions au sein d'une activité.
    - Diagramme d'états : Il montre les différents états et transitions possibles des objets d'une classe.
* Six diagrammes structurels :
  + Diagramme de classes : Il montre les briques de base statiques : classes, associations, interfaces, attributs, opérations, généralisations, etc.
  + Diagramme d'objets : Il montre les instances des éléments structurels et leurs liens à l'exécution.
  + Diagramme de packages : Il montre l'organisation logique du modèle et les relations entre packages.
  + Diagramme de structure composite : Il montre l'organisation interne d'un élément statique complexe.
  + Diagramme de composants : Il montre des structures complexes, avec leurs interfaces fournies et requises.
  + Diagramme de déploiement : Il montre le déploiement physique des 'artefacts' sur les ressources matérielles.

# 3 Conception détaillée

## 3.1 Les vues de Kruchten

Kruchten offre des perspectives distinctes et complémentaires pour décrire la méthodologie 2TUP que nous avons choisie. Ainsi, en optant pour l'approche de Kruchten, nous pourrons passer à la phase de conception de notre projet. Le langage de modélisation UML sera utilisé pour cette étape et ses 13 types de diagrammes seront employés en fonction des besoins et répartis selon les vues de Kruchten, comme présenté dans la Figure III.5.

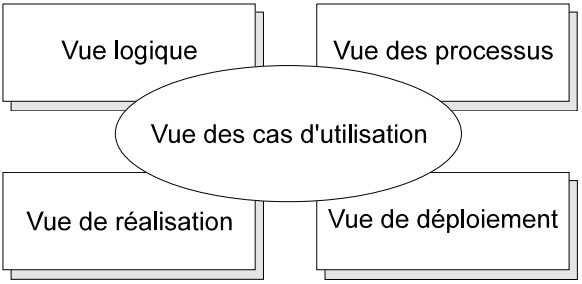


Figure III. 5 : Vue 4+1 (Kruchten)

* La vue des cas d'utilisation : est un guide pour toutes les autres vues. Elle définit les besoins des clients du système ainsi que l'architecture nécessaire pour répondre à ces besoins. Cette vue unifie les autres vues et motive les choix en identifiant les interfaces et en mettant l'accent sur les besoins des utilisateurs. Elle présente les diagrammes de cas d'utilisation et de package.
* La vue logique : décrit les aspects statiques et dynamiques d'un système en termes de classes, d'objets, de connexions et de communications. Elle se concentre sur l'abstraction et l'encapsulation.

Cette vue présente les diagrammes de classe et d'objet.

* La vue de processus : décrit les interactions entre les différents processus (threads, tâches) et exprime la synchronisation et l'allocation des objets. Elle vérifie également le respect des contraintes de fiabilité, d'efficacité et de performances des systèmes multitâches. Cette vue présente les diagrammes de séquence, d'activité, de collaboration, d'état de transition et de temps.
* La vue des composants : également appelée vue de réalisation, est de bas niveau et permet de visualiser l'organisation des composants (bibliothèque dynamique et statique, code source...) dans l'environnement de développement. Elle permet également de gérer la configuration (auteurs, versions...). Cette vue présente les diagrammes de composants et de structure composite.

La vue de déploiement : est utile pour la maintenance régulière du système pour la tolérance aux fautes et aux pannes. Elle permet également de traiter les contraintes

## 3.2 Modèles d'UML utilisés

Dans la partie précédente, nous avons présenté différents diagrammes UML qui permettent de décrire les objets et les liens qui les unissent. En particulier, pour répondre aux vues de Kruchten, nous avons choisi les diagrammes suivants :

* Le diagramme de cas d'utilisation pour la vue des cas d'utilisation.
* Le diagramme de séquence pour la vue des processus objet.
* Le diagramme de classe pour la vue logique.
* Diagrammes d’activites
* Diagrammes d’etat et transitions

Ces quatre diagrammes ont été soigneusement sélectionnés pour répondre aux besoins spécifiques de notre projet et couvrir les aspects essentiels de la modélisation.

## 3.3 Vue des cas d'utilisation : Diagramme de cas d’utilisation

Un cas d'utilisation correspond à une unité distincte d'interaction entre un utilisateur (qu'il soit humain ou machine) et un système. Il représente une tâche significative à réaliser pour l'utilisateur. Dans un diagramme de cas d'utilisation, les utilisateurs sont représentés en tant qu'acteurs, qui interagissent avec les cas d'utilisation. Ce diagramme permet d'identifier les différentes interactions possibles entre le système et les acteurs, ainsi que toutes les fonctionnalités que le système doit fournir.

