**MINISTERE DE L’ENSEIGNEMENT SUPERIEUR**

**DE LA RECHERCHE ET DE LA TECHNOLOGIE**

**Université de la Manouba**

**Ecole Nationale des Sciences de l’Informatique**

****

**Rapport de projet de conception et de développement**

**Sujet :** Développement d’un site d’annonce

**Réalisé par :**

Amine Dridi Jihed Chokri Achref Chamekh

**Encadré par :**

M. Ben Rhouma Kamel

Année Universitaire 2015-2016

**REMERCIEMENTS**

Nous tenons à remercier en premier lieu, Monsieur Ben Rhouma Kamel, pour la qualité de son encadrement, son indulgence, ses remarques constructives et ses précieux conseils qu’il nous a prodigués tout le long de notre projet.

Nos vives considérations s’adressent à tous les professeurs de l’ENSI pour la formation d'excellence qui nous ont offerte, pour leurs conseils et leurs contributions pour que nous entamions notre carrière avec des bases solides.

Nous accordons également nos remerciements les plus distincts à tous les membres du jury pour avoir accepté d’évaluer ce travail.

Enfin, nos sincères remerciements à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce projet.

**Sommaire**

[Table des figures 4](#_Toc449896847)

[Tables des tableaux 5](#_Toc449896848)

[Liste des abréviations 6](#_Toc449896849)

[Introduction générale 7](#_Toc449896850)

[Chapitre1 : Contexte du projet 9](#_Toc449896851)

[1.1. Introduction 9](#_Toc449896852)

[1.2. Étude de l’existant et solutions proposées 9](#_Toc449896853)

[1.3. Travail demandé 10](#_Toc449896854)

[1.4. Conclusion 10](#_Toc449896855)

[ÉTAT DE L’ART 12](#_Toc449896856)

[Chapitre 2 : État de l’art 12](#_Toc449896857)

[2.1. Introduction 12](#_Toc449896858)

[2.2. La gestion de projet 13](#_Toc449896859)

[2.2.1. Le chef du projet 13](#_Toc449896860)

[2.3. Méthodologie de gestion de projet 13](#_Toc449896861)

[2.3.1. Présentation de la méthodologie SCRUM 13](#_Toc449896862)

[2.3.2. Les rôles de la méthodologie SCRUM 15](#_Toc449896863)

[2.4. Choix de C# et de Asp.net MVC 17](#_Toc449896864)

[2.5. Conclusion 18](#_Toc449896865)

[Chapitre 3 : Analyse et Spécifications des Besoins 19](#_Toc449896866)

[3.1. Introduction 19](#_Toc449896867)

[3.2. Présentation de l’équipe 20](#_Toc449896868)

[3.3. Backlog du produit 20](#_Toc449896869)

[3.4. Analyse des Besoins 23](#_Toc449896870)

[3.4.1. Besoins fonctionnels 23](#_Toc449896871)

[3.4.2. Besoins non fonctionnels 25](#_Toc449896872)

[3.5. Spécification des Besoins 25](#_Toc449896873)

[3.5.1. Unified Modeling Langage 26](#_Toc449896874)

[3.5.2. Identification des acteurs 26](#_Toc449896875)

[3.5.3. Diagramme des cas d’utilisation 26](#_Toc449896876)

[3.5.4. Scénario des cas d’utilisations 28](#_Toc449896877)

[3.6. Conclusion 33](#_Toc449896878)

[Chapitre 4 : Réalisation de la solution 35](#_Toc449896879)

[4.1. Introduction 35](#_Toc449896880)

[4.2. Architecture générale de la solution 35](#_Toc449896881)

[4.2.1. Introduction aux design patterns 37](#_Toc449896882)

[4.2.2 Le diagramme des paquetages 39](#_Toc449896883)

[4.3. Environnement de travail 40](#_Toc449896884)

[4.3.1. Environnement matériel 40](#_Toc449896885)

[4.3.2. Environnement logiciel 41](#_Toc449896886)

[4.5. Sprint 1 : Gestion des utilisateurs 45](#_Toc449896887)

[4.5.1. Conception 45](#_Toc449896888)

[4.5.2. Développement 48](#_Toc449896889)

[4.5.3. Test 49](#_Toc449896890)

[4.6. Sprint 2 : Gestion des annonces 49](#_Toc449896891)

[4.6.1. Conception 50](#_Toc449896892)

[4.6.2. Développement 53](#_Toc449896893)

[4.6.3. Test 54](#_Toc449896894)

[4.7. Sprint 3 : Enrichir l’application par d’autres services 55](#_Toc449896895)

[4.7.1. Conception 55](#_Toc449896896)

[4.7.2. Développement 57](#_Toc449896897)

[4.8. Conclusion 58](#_Toc449896898)

[Conclusion générale 60](#_Toc449896899)

[Bibliographie 63](#_Toc449896900)

[Webographie 65](#_Toc449896901)

# Table des figures

Fig.1 : Processus de gestion de projet

Fig.2 : Cycle de vie d’un projet avec la méthode SCRUM

Fig.3 : Equipe Scrum et déroulement de la méthodologie

Fig.4 : Division des histoires en sprints

Fig.5 : Diagramme Use Case

Fig.6 : Architecture générale de l’application

Fig.7 : Diagramme de paquetage global

Fig.8 : Fonctionnement d’un ORM

Fig.9 : sprint1 : diagramme des classes

Fig.10 : Diagramme de séquence : Opération d’enregistrement

Fig.11 : Procédure d’authentification via le Web Service

Fig.12 : Diagramme de séquence : Acceptation d’un nouvel utilisateur

Fig.13 : Page d’Enregistrement

Fig.14 : Page d’acceptation des nouveaux utilisateurs par l’administrateur

Fig.15 : Diagramme de séquence : ajout d’une annonce

Fig.16 : Page d’ajout d’une annonce

Fig.17 : Page d’ajout d’une annonce à un produit

Fig.18 : Diagramme des classes pour le sprint 3

Fig.19 : Page pour contacter le vendeur

Fig.20 : Information sur le produit

Fig.21 : Information sur le produit et ajout des commentaires

# Tables des tableaux

Tableau 1: Mot clés de la méthodologie SCRUM

Tableau 2 : Le backlog du produit

Tableau 3 : Les acteurs de l’application

# Liste des abréviations

**SOA** Services Oriented Architecture

**DAL** Data Accsess Layer

**BLL** Business Logic Layer

**MVVM** Model View ViewModel

**BO**  Business Objects

**WCF** Windows Communication Foundation

**ORM** Object Relationnal Mapping

**DTO**  Data Transfer Object

|  |
| --- |
| INTRODUCTION GENERALE |

# Introduction générale

Les consommateurs cherchent toujours à acheter des produits de qualité avec les meilleurs prix. Mais, trouver ce produit, nécessite un grand effort pour se déplacer entre les différents marchés et parfois beaucoup de temps libre et de l’argent pour se procurer le bien qui leur convient.

Nous proposons alors, une solution permettant de pallier à ces problèmes en offrant une application web pour relier les acheteurs des vendeurs afin d’assurant leur satisfaction.

Le présent rapport s’inscrit dans le cadre de développement d’un projet de conception et de développement au sein de l’Ecole Nationale Des Sciences de l’Informatique.

La méthodologie de gestion de notre projet se base sur la méthode agile SCRUM.

Le présent rapport s'articule autour de quatre chapitres à savoir :

* Le premier chapitre propose une description approfondie de l'étude préalable en ce qui concerne les modules que nous allons implémenter avec une spécification de notre cahier de charge.
* Le deuxième chapitre est une présentation de la notion de gestion de projet ainsi qu'une définition détaillée de la méthodologie SCRUM et le choix du langage de programmation.
* Le troisième chapitre sera consacré à l'analyse et à la spécification des besoins ainsi qu’à l’identification des acteurs et la conception scénarii des cas d'utilisation prioritaires
* Le quatrième chapitre, est consacré à la réalisation du projet.

Nous finirons le projet par illustrer quelques captures d'écran de notre application. Une conclusion et un ensemble de perspectives seront présentés à la fin pour clôturer notre travail.

|  |
| --- |
| CHAPITRE 1 :  CONTEXTE DU PROJET |

Plan du chapitre :

Introduction

Étude de l’existant et solution proposée

Travail demandé

Conclusion

# Chapitre1 : Contexte du projet

## 1.1. Introduction

Ce chapitre est dédié à mettre le projet dans son cadre général. Nous commençons par l’étude de l’existant ainsi que les différentes solutions proposées. Nous passons ensuite à un descriptif global du travail demandé et les outils utilisés durant le développement de l’application.

## 1.2. Étude de l’existant et solutions proposées

La majorité des solutions proposées dans ce cadre comme « Tayara, Tunisie annonce, Ballouchi,… » possèdent des fonctionnalités limitées au simple contact entre le client et le vendeur et ne permettent pas à l’acheteur de bien choisir ses produits de même l’accès aux différentes ressources s’avère parfois difficiles.

De même, la plus part des applications proposées n’offrent pas le service d’ajouter une annonce contenant plusieurs produits de différentes catégories.

## 1.3. Travail demandé

Le travail demandé consiste à réaliser une application Web offrant des multiples services afin faciliter la vente et l’achat avec le meilleurs prix qui satisfont l’acheteur et le vendeur.

L’application que nous envisagerons de réaliser fournit :

* Un espace utilisateur : permet aux membres authentifiés d’ajouter des annonces ainsi que d’autres fonctionnalités qu’on verra avec plus de détails dans les chapitres suivants.
* Un espace d’administration : permet aux membres qualifiés d’un rôle d’administrateur d’accepter ou de rejeter les nouveaux produits récemment ajoutés afin qu’elles apparaissent au public.
* Un espace commun : permet à tous les internautes de consulter les produits et de contacter par message son propriétaire.

Pour mettre en place cette solution, nous avons utilisé un ensemble d’outils, langage et méthodes à savoir :

* Méthode de gestion de projet : SRCUM ;
* Langage de programmation : C# ;
* Framework de développement : Asp.net MVC ;
* Langage de modélisation : UML.

## 1.4. Conclusion

Tout au long de ce chapitre, nous avons présenté le cadre général du projet. De ce fait, nous avons présenté le sujet de notre projet ainsi, une étude de l’existant afin de permettre de mieux comprendre le sujet.

Le chapitre suivant sera consacré à la présentation des concepts fondamentaux utilisés dans notre projet. De plus, nous allons entamer la partie technique de notre solution et présenter les différentes plateformes et en particulier le choix de « Asp.net MVC ».

|  |
| --- |
| CHAPITRE 2 : ÉTAT DE L’ART |

Plan du chapitre :

Introduction

La gestion de projet

Méthodologie de gestion de projet

Choix de C# et de «Asp.net MVC»

Conclusion

# Chapitre 2 : État de l’art

## 2.1. Introduction

Ce chapitre présente quelques notions de base nécessaires à la bonne compréhension du projet. Nous allons découvrir particulièrement le domaine « Gestion de Projet ».

Ce chapitre sera composé de quatre parties qui seront réparties comme suit : la première partie sera consacrée à la présentation du processus de gestion de projet. Ensuite, nous allons présenter la méthodologie de gestion de projet que nous avons utilisée par l’application afin d’atteindre nos objectifs. La troisième partie sera consacrée la présentation des plateformes de gestion de projet existante sur le marché. Enfin, nous terminerons ce chapitre par la justification du choix du C# et de «Asp.net MVC».

## 2.2. La gestion de projet

La gestion de projet ou conduite de projet est une démarche méthodologique servant à organiser, assurer et perfectionner le déroulement d’un projet. Au cours de sa réalisation, elle mobilise des ressources humaines et matérielles. Elle implique une action planifiée dans le temps et une budgétisation de moyens. L’objectif principal de la gestion de projet est de garantir la coordination des acteurs et des tâches en toute efficacité et rentabilité. La notion de management de projet intègre celle de la gestion de projet dans la mesure où elle ajoute d’autres paramètres dans la définition des objectifs stratégiques de la direction.



Fig.1 : Processus de gestion de projet

### 2.2.1. Le chef du projet

Gérer et animer un projet, être chef de projet c’est d’abord savoir en négocier l’objectif mais aussi prévoir. Pour cela, il faut savoir mettre en œuvre les outils de l’analyse fonctionnelle, de planification (WBS, OBS, matrice RACI, PERT, Gantt), gérer un budget, maîtriser les risques, motiver et animer une équipe-projet tout cela en conciliant les intérêts du maitre d’ouvrage et des parties prenantes. Les livrables doivent être clairement définis pour un résultat conforme à des normes de qualité, pour le moindre coût et dans le meilleur délai possible.

## 2.3. Méthodologie de gestion de projet

### 2.3.1. Présentation de la méthodologie SCRUM

Bien qu’il existe une multitude de méthodologies de développement logiciel, nous avons eu recours aux méthodes agiles. Ces derniers visent à réduire le cycle de développement du logiciel en réalisant une version minimale, puis en intégrant les fonctionnalités par un processus itératif basé sur une écoute client et des tests tout au long du cycle de développement. Donc Scrum est l’une des méthodes agiles pour la gestion des projets informatiques. C’est une méthode itérative basée sur des itérations de courte durée appelées « Sprint » et qui permettent de produire la plus grande valeur métier dans la durée la plus courte.

* Scrum vise avant tout à produire un logiciel fonctionnel en mettant en avant la rentabilité et surtout la satisfaction client. Ce dernier doit pouvoir donc juger sur le tas de l'avancement de son produit.
* Dans la méthodologie Scrum, l'équipe de développement s'organise elle-même pour déterminer la meilleure façon de produire les exigences les plus prioritaires.
* Scrum est une approche basée sur le principe d’une équipe soudée ou il y a plus de coopération et de communication.
* Finalement, à chaque fin de sprint, tout le monde a un pouvoir de décision. L'équipe devra décider de livrer le produit en l'état, ou de continuer à améliorer le produit sur une itération supplémentaire.

Afin de comprendre cette méthodologie on a établi le tableau I pour définir des mots clés qui vont nous servir tout au long du projet et de ce rapport.

|  |  |
| --- | --- |
| **Terme** | **Définition** |
| Backlog du Produit | Définition des besoins fonctionnels sous forme de « user story » |
| Backlog de Sprint | Liste des tâches à implémenter dans un sprint, classées par importance et état |
| Burn-Down Chart | Diagramme permettant à suivre l’état d’avancement du projet (Représentation du reste à faire) |
| Produit Partiel | Résultat du Sprint Testé et potentiellement livrable |
| Scrum Quotidien | Le SCRUM meeting est une réunion organisée tous les jours debout qui dure 5 minutes |

Tableau 1: Mot clés de la méthodologie SCRUM

La figure suivante donne une vue globale de Scrum en utilisant les mots clés qu’on vient de définir :



Fig.2 : Cycle de vie d’un projet avec la méthode SCRUM

### 2.3.2. Les rôles de la méthodologie SCRUM

L’équipe Scrum englobe plusieurs acteurs travaillant ensemble en but de réaliser des objectifs communs, ci-dessous on va détailler le rôle de chacun :

**Directeur de produit ou Product Owner** : représente à la fois les clients et les utilisateurs. Mais le terme de directeur est ici à prendre au sens de guide plus que de chef hiérarchique. En effet, ses responsabilités se bornent à l'établissement des limites du projet et de chaque itération.

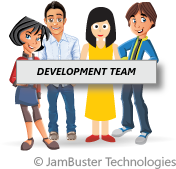
Le Product Owner a un ensemble de responsabilités qu’on peut résumer dans les points suivants :

* Choisir la date et le contenu de la release
* Responsable du retour sur investissement
* Définir les priorités dans le backlog en fonction de la valeur « métier »
* Ajuster les fonctionnalités et les priorités à chaque sprint si nécessaire

**Scrum Master** : représente le management du projet. On l'assigne bien souvent au rôle de manager de projet ou de Chef d'équipe. Son travail principal consiste remédier aux imprévus. C'est celui qui intervient dans le cas où une situation ou un événement peut empêcher ou retarder la progression du travail prévu au cours du sprint.

On peut résumer ses responsabilités dans les points suivants :

* Appliquer les pratiques de Scrum sur l’équipe
* Résoudre les problèmes
* S'assurer que l'équipe est complètement fonctionnelle et productive
* Faciliter une coopération poussée entre tous les rôles et les fonctions
* Protéger l'équipe des interférences extérieures



**Scrum Team** : Une bonne équipe SCRUM est assez réduite et comporte finalement 5 à 10 personnes. Une bonne équipe regroupe tous les rôles : Architecte, concepteur, développeur, spécialiste IHM, testeur … et elle est caractérisée par une forte coopération et haute communication entre les différents membres.

La figure ci-dessous met en évidence le déroulement de la méthodologie Scrum

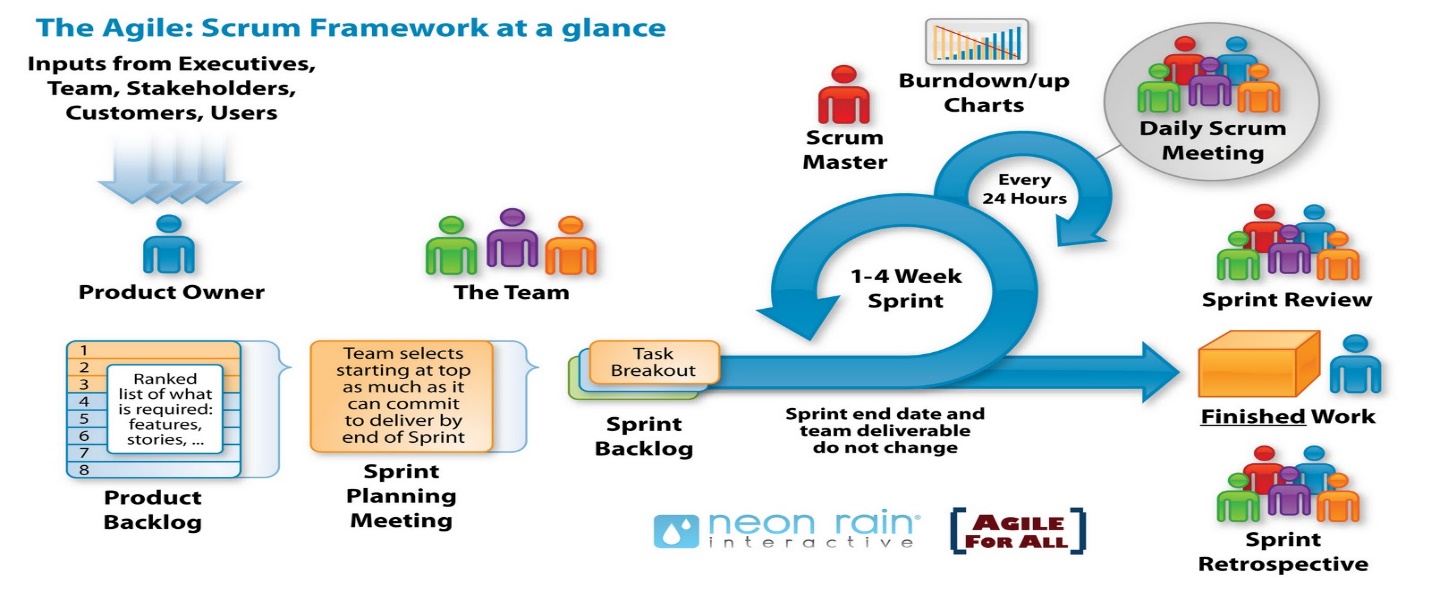


Fig.3 : Equipe Scrum et déroulement de la méthodologie

## 2.4. Choix de C# et de Asp.net MVC

Le géant de l’informatique « Microsoft » met à la disposition des développeurs du Web un Framework simple à utiliser et très puissant « Asp.net ».

Asp.net offre deux logiques de développement différents : WebForm et le MVC.

Nous avons choisi le MVC pour plusieurs raisons :

1. La simplicité.
2. Facilite la façon de développer une application web.
3. Facilite la maintenance.
4. Conseillé si l’application évolue dans le temps.
5. S'adapte bien à la pratique de développement Agile.

**Fonctionnalités**

* Sécurité : Les requêtes en base sont protégées des injections SQL, les fonctions de filtrage et de validation aident à la protection contre les attaques de types cross-sitescripting (XSS) et Cross-site request forgery(CSRF ou XSRF).
* Organisation du code : L'organisation des répertoires et des classes suit certaines normes. On peut ainsi construire son application par assemblages de blocs indépendants bien organisés entre eux.
* URL simples et claires : La forme des URL est entièrement paramétrable, ceci permet d'améliorer le référencement de ses sites.
* De nombreuses fonctions courantes sont facilitées : Accès en base de données, accès à des services externes, authentification et droits d'accès…

## 2.5. Conclusion

Tout au long de ce chapitre, nous avons présenté quelques notions théoriques qui sont indispensables à la compréhension de notre travail. Le prochain chapitre sera consacré à l’analyse et la spécification des besoins ainsi qu’à la définition de notre backlog de produit.

|  |
| --- |
| CHAPITRE 3 :  ANALYSE  ET SPECIFICATIONS  DES BESOINS |

Plan du chapitre :

Introduction

Présentation de l’équipe

Backlog du produit

Analyse des Besoins

Spécification des Besoins

Conclusion

# Chapitre 3 : Analyse et Spécifications des Besoins

## 3.1. Introduction

L’analyse et la spécification des besoins représentent une étape fondamentale dans un cycle de vie de développement d’une solution. De ce fait, nous consacrons ce chapitre à la présentation détaillée de cette notion.

Nous commencerons par définir notre backlog du produit. Ensuite, nous spécifions les besoins fonctionnels et non fonctionnels. Nous détaillons l’analyse des besoins en identifiant les fonctionnalités, les acteurs et en présentant le diagramme des cas d’utilisation avec la réalisation de quelques prototypes des interfaces.

## 3.2. Présentation de l’équipe

Dans notre projet, l’équipe est composée de :

* Product Owner : Ben Rhouma Kamel
* Scrum Master : Dridi Amine
* L’équipe :
* Dridi Amine
* Chokri Jihed
* Chemakh Achref

## 3.3. Backlog du produit

SCRUM utilise une approche fonctionnelle pour récolter les besoins des utilisateurs. L'objectif est d'établir une liste de fonctionnalités à réaliser, que l'on appelle backlog de produit.

Le backlog du produit est la liste des fonctionnalités attendues d'un produit. Plus exactement, au-delà de cet aspect fonctionnel, il contient tous les éléments qui vont nécessiter du travail. Les éléments sont classés par priorité ce qui permet de définir l'ordre de la réalisation.

Le tableau présente l’ensemble des histoires des utilisateurs.

Ces histoires sont décrites à l’aide de leur id, noms, complexités et leur priorité (importance).

Le backlog comprend les champs de base suivants:

* Le champ Importance, qui définit comme son nom l’indique l’importance contractuelle de la réalisation de cette partie du projet.
* Le champ ID qui détermine un identifiant unique pour l’histoire en question.
* Le champ User story qui décris de manière claire et succincte la fonctionnalité désirée par l’utilisateur.
* Le champ Estimation qui estime en « points » la charge de travail. Ce champ est fait suivant la suite de Fibonacci : Suite de Fibonacci : Pour tout n ≥ 1, un+1 = un + un−1 ; Les deux premiers termes étant 0 et 1. Le troisième terme est donc 1 (0 + 1 = 1), le quatrième terme 2 (1 + 1 = 2), le cinquième 3 (1 + 2 = 3), le sixième 5 (2 + 3 = 5), et ainsi de suite. Le début de la suite du célèbre mathématicien Fibonacci est donc : 0, 1, 1, 2, 3, 5, 8, 13...

**L’importance**

Le champ importance est déterminé selon une échelle totalement libre. En effet une histoire avec une importance à 80 ne sera pas forcément 4 fois plus importante qu’une histoire avec une importance de 20.

Il est important que chaque histoire ait une importance unique afin de créer une hiérarchie claire entre celles-ci et il est également important de laisser un écart suffisant entre chacune d’elle à la création du backlog. Il faudra être en mesure de pouvoir insérer de nouvelles histoires que l’on n’aurait pas forcément prévues.

Pour aider à déterminer l’échelle que l’on va utiliser, on peut se fixer 3 seuils avec un code couleur :

* 60 : Histoire de priorité élevée.
* 40 ≤X ≤60 : Histoire de priorité moyenne.
* < 40 : Histoire de priorité faible.

Le tableau suivant présente le backlog du produit relatif à notre application :

|  |  |  |  |
| --- | --- | --- | --- |
| id | User story | Estimation (point) | Importance |
| 1 | En tant qu’administrateur je veux m’authentifier | 13 | 80 |
| 2 | En tant qu’administrateur je veux accepter les nouveaux inscrits | 5 | 60 |
| 3 | En tant qu’administrateur je veux refuser les nouveaux inscrits | 5 | 60 |
| 4 | En tant qu’administrateur je veux ajouter des nouvelles catégories | 2 | 40 |
| 5 | En tant qu’administrateur je veux supprimer des catégories | 2 | 40 |
| 6 | En tant qu’administrateur je veux ajouter des nouvelles régions | 2 | 40 |
| 7 | En tant qu’administrateur je veux supprimer des régions | 2 | 40 |
| 8 | En tant qu’administrateur je veux supprimer des utilisateurs | 2 | 40 |
| 9 | En tant qu’administrateur je veux modifier les données d’un utilisateur | 2 | 40 |
| 10 | En tant qu’administrateur je veux accepter ou refuser un produit. | 3 | 60 |
| 11 | En tant qu’administrateur je veux ajouter, modifier et supprimer une annonce. | 3 | 60 |
| 12 | En tant qu’administrateur je veux consulter les annonces. | 1 | 30 |
| 13 | En tant qu’administrateur je veux ajouter, modifier et supprimer un produit d’une annonce. | 2 | 40 |
| 14 | En tant qu’administrateur je consulter les produits. | 2 | 40 |
| 15 | En tant qu’administrateur je veux ajouter, supprimer et modifier un commentaire. | 2 | 40 |
| 16 | En tant qu’administrateur je veux supprimer les commentaires non appropriés. | 3 | 60 |
| 17 | En tant que membre je veux m’authentifié. | 8 | 80 |
| 18 | En tant que membre je veux ajouter plusieurs produits dans une annonce. | 14 | 90 |
| 19 | En tant que membre je veux consulter, modifier et supprimer mes annonces. | 5 | 70 |
| 20 | En tant que membre je veux consulter, modifier et supprimer un produit appartenant à une annonce. | 2 | 40 |
| 21 | En tant que membre je veux être contacté par mail. | 4 | 50 |
| 22 | En tant que membre je veux consulter les produits en annonces. | 4 | 60 |
| 23 | Je veux ajouter, supprimer un commentaire sur un produit. | 13 | 80 |
| 24 | En tant que personne je veux m’inscrire au site. | 13 | 80 |
| 25 | En tant que personne je veux consulter les produits. | 2 | 40 |
| 26 | En tant que personne je veux contacter l’utilisateur par e-mail. | 3 | 50 |
| 27 | En tant que personne je veux lire les commentaires sur un produit. | 2 | 40 |
| 28 | En tant que personne je veux comparer des produits. | 6 | 50 |
| 29 | En tant qu’administrateur je veux avoir toutes les fonctionnalités d’un membre authentifié. | 5 | 80 |
| 30 | En tant que membre je veux avoir toutes les fonctionnalités qu’une personne. | 5 | 80 |

Tableau 2 : Le backlog du produit

Après avoir énuméré les différentes tâches avec le « Product Owner » et le reste des membres de l’équipe, nous avons organisé les tâches sous forme de sprint ou chaque sprint aura une durée de 20 jours ou plus avec l’objectif d’avoir un produit livrable et fonctionnel à la fin de chaque sprint.

Ce qui nous emmène à la figure 4 ou on a organisé nos fonctionnalités sur 3 sprints 



Fig.4 : Division des histoires en sprints

## 3.4. Analyse des Besoins

L'analyse des besoins est l’une des premières étapes du déroulement d'un projet. La complexité de l’analyse des besoins est aussi due à leurs nombreuses facettes. Il existe des besoins exprimés naturellement, des besoins implicites, des besoins inavoués et tout simplement des besoins dont les utilisateurs n’ont même pas conscience.

### 3.4.1. Besoins fonctionnels

Les besoins fonctionnels correspondent aux fonctionnalités du système. Ce sont les besoins spécifiant un comportement d'entrée / sortie du système.

Le système doit permettre :

A une personne :

* Consulter les produits.
* Contacter le vendeur par l’envoie d’un e-mail.
* Lui proposer une liste des produits en relations avec celles cherché par les internautes.
* Avoir un moteur de recherche.
* Disposer des outils de filtrages pour raffiner les recherches.

A un membre :

* Avoir les mêmes fonctionnalités qu’personne non registrer.
* S’authentifier.
* Lister les produits.
* Gérer ses annonces (CRUD).
* Gérer ses produits (CRUD sur un produit d’une annonce).
* Comparer les produits existants.

A l’administrateur de :

* Avoir les mêmes fonctionnalités qu’un membre.
* S’authentifier : Sécurisé l’administration par un identifiant et un mot de passe.
* Configurer l’application : mettre à jour la base de données : ajouter, modifier, supprimer, consulter certains enregistrements qui se trouvent dans la base de données de l’application à travers des interfaces appropriées.
* CRUD sur les catégories.
* CRUD sur les annonces.
* CRUD sur les produits.
* CRUD sur les utilisateurs.
* Affirmer/rejeter un produit d’une annonce.
* Contacter les utilisateurs.
* Affirmer/rejeter les inscriptions reçues.

### 3.4.2. Besoins non fonctionnels

Il s'agit des besoins qui caractérisent le système. Ce sont des besoins en matière de performance ou de type de matériel. Ces besoins peuvent concerner les contraintes d'implémentation (langage de programmation, type SGBD, de système d'Exploitation...).

Nous envisageons une solution qui sera :

Sécurité :

* Besoins d’authentification pour certains types d’utilisateur (niveau d’accès).

Ergonomie :

* Offrir une jolie interface aux utilisateurs.
* Facilité d’accès aux données.

Compatibilités :

* L’application de fonctionner sur des différents navigateurs.
* La taille de l’écran ne devra pas affecté la beauté de l’application.

Maintenance :

* L’architecture de l’application devra être ouverte à la modification.
* Les services offerts devraient être réutilisables pour d’autre application (exemple : application mobile, desktop,…)

Performance :

* Le temps de réponse au chargement de l’application et le temps de traitement des fonctions doivent être raisonnable.

## 3.5. Spécification des Besoins

Pour spécifier d’une façon formelle les besoins requis par l’application, nous avons utilisé les diagrammes de cas d’utilisation d’UML. Ceci est une manière pour assurer une meilleure compréhension des besoins et la réalisation des prototypes des interfaces.

### 3.5.1. Unified Modeling Langage

UML (Unified Modeling Language) est un langage de modélisation objet unifié très utilisé et très connu dans le monde, né de la fusion de plusieurs méthodes de modélisation objet. Il présente une démarche itérative et incrémentale guidée par les besoins des utilisateurs.

### 3.5.2. Identification des acteurs

Un acteur est une entité externe qui interagit avec le système. En réponse à l’action d’un acteur, le système fournit un service qui correspond à son besoin.

Dans notre l’application, on distingue plusieurs rôles représentés dans le tableau suivant :

|  |  |
| --- | --- |
| **Acteur** | **Rôle** |
| Administrateur | Responsable de l’administration, de l’exploitation du système et de la gestion des utilisateurs. Il intervient aussi pour approuver des produits et de supprimer des commentaires non appropriés. |
| Utilisateur inscrit (membre) | Il a le droit de :   * Ajouter des annonces. * Gérer ses produits. * Gérer son compte. |
| Utilisateur non inscrit (visiteur) | N’importe quel utilisateur de web. Il a la possibilité de consulter les produits ainsi que de contacter le propriétaire par e-mail. |

Tableau 3 : Les acteurs de l’application

### 3.5.3. Diagramme des cas d’utilisation

Pour avoir une vue globale sur les grandes lignes de notre solution, la figure ci-dessous présente le diagramme de cas d’utilisation général.

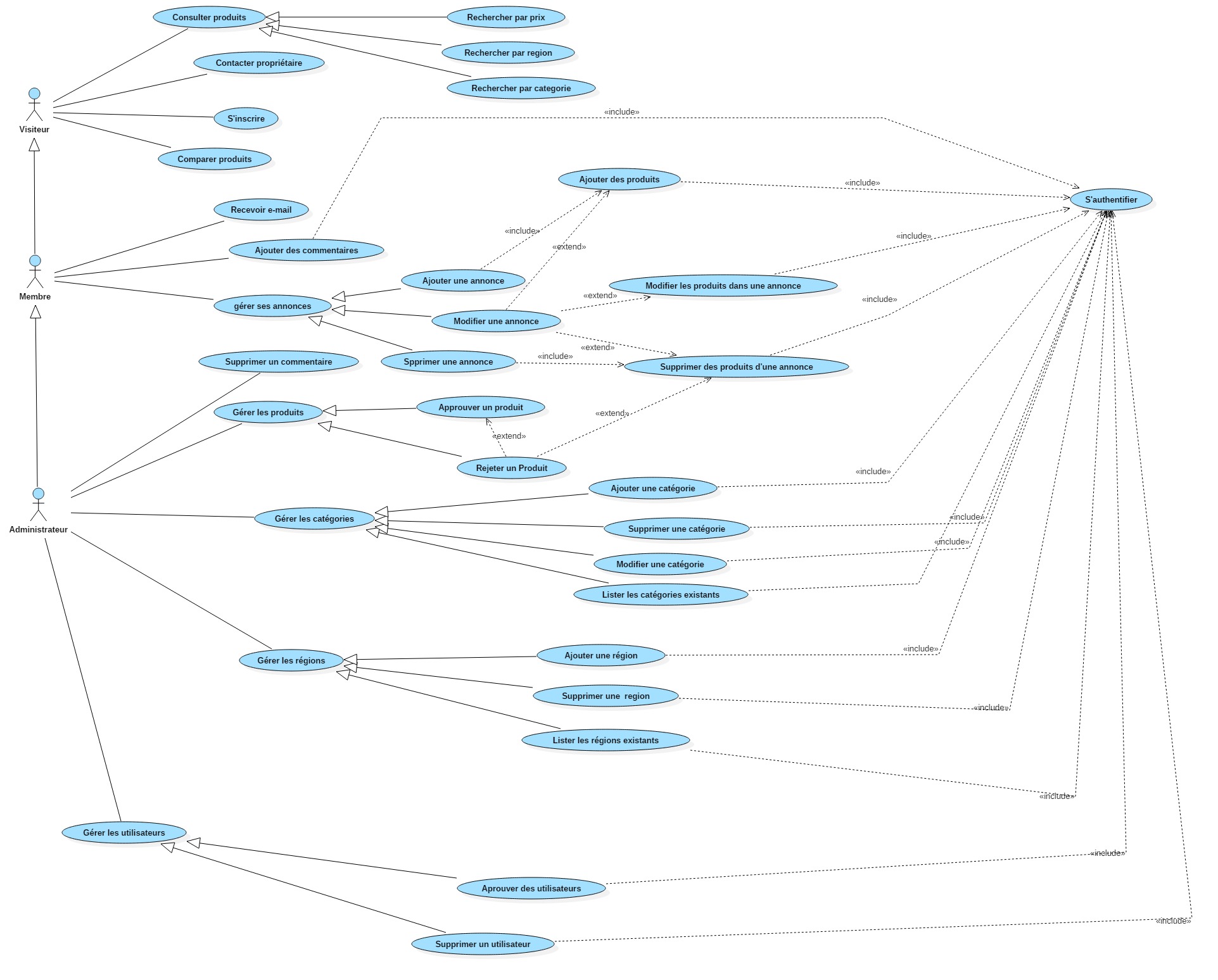


Fig.5 : Diagramme Use Case

### 3.5.4. Scénario des cas d’utilisations

#### 3.5.4.1. Description textuelle du cas d'utilisation « consulter un produit »

|  |
| --- |
| **Titre** : consulter un produit  **Acteur principal** : une personne  **Résumé** : chercher un produit selon des critères bien définis.  **Préconditions :**   1. utilisateur connecté sur une page lui permettant d’effectué une recherche sur un produit.   **Scénario Principal :**   1. L’utilisateur écrit le nom du produit recherché. 2. L’utilisateur choisit une ou plusieurs catégories. 3. L’utilisateur choisit une ou plusieurs régions. 4. L’utilisateur fixe un prix maximal et un prix minimal. 5. L’utilisateur clique sur le bouton Chercher. 6. Le système propose une liste des produits répondant aux critères imposés. 7. L’utilisateur choisit un produit parmi les produits proposés.   **Post condition :**   1. Le système affiche les informations propres au produit choisi.   **Scénario alternatif 1:** l’utilisateur passe directement de l’étape 1 du scénario principale à l’étape 5   1. Le système affiche une liste de tous les produits ayant un rapport avec le nom entré 2. On reprend le scénario principal à partir de l’étape 6.   **Scénario alternatif 2:** l’utilisateur passe directement à l’étape 5 du scénario.   1. Le système affiche une liste de tous les produits. 2. On reprend le scénario principal à partir de l’étape 6.   **Exception1 :** l’utilisateur entre un mot invalide et clique sur Rechercher.   1. Le système affiche aucun produit est disponible. |

#### 3.5.4.2. Description textuelle du cas d'utilisation «Comparer deux produits»

|  |
| --- |
| **Titre** : comparer deux produits  **Acteur principal** : une personne  **Résumé** : afficher sur une même page les informations sur les deux produits.  **Pré-conditions :**   1. utilisateur connecté sur la page Compare Produit.   **Scénario Principal :**   1. le système affiche un formulaire demande l’url de chacun des deux produits. 2. L’utilisateur ajoute l’url du premier produit. 3. L’utilisateur ajoute l’url du deuxième produit. 4. L’utilisateur clique sur le bouton Comparer.   **Post condition :**   1. Le système affiche les informations propres aux deux produits dans une seul page pour faciliter la comparaison.   **Exception1 :** l’utilisateur entre un Url invalide et clique sur Comparer.   1. Le système affiche sur la même page une erreur de type URL invalide ou produit inexistant. |

#### 3.5.4.3. Description textuelle du cas d'utilisation «Ajouter une annonces»

|  |
| --- |
| **Titre** : Ajouter une annonce  **Acteur principal** : membre  **Résumé** : ajouter une annonce disponible dans une ou plusieurs régions et contenant plusieurs produits de différentes catégories.  **Pré-conditions :**   1. Membre authentifié. 2. Membre connecté sur une page disposant le bouton Déposer votre Annonce.   **Scénario Principal :**   1. L’utilisateur clique sur le bouton Déposer votre Annonce. 2. Le système affiche un formulaire. 3. L’utilisateur doit remplir les champs  * Titre de l’annonce. * Nombre des produits à ajouter. * Le champ description est optionnel.  1. L’utilisateur clique sur le bouton suivant. 2. Le système affiche un formulaire concernant le produit à ajouter. 3. L’utilisateur doit remplir les champs :  * Nom * Prix * Description  1. L’utilisateur peut ajouter jusqu’à cinq photo de son produit. 2. L’utilisateur clique sur le bouton Suivant. 3. Le système affiche de nouveau le formulaire concernant l’ajout d’un produit. 4. Les étapes 5, 6, 7, 8 se reproduisent autant de fois que le nombre de produits précisé à l’étape 3. 5. Le système affiche un bouton Confirmer. 6. L’utilisateur clique sur le bouton Confirmer. 7. Le système affiche une page indiquant le succès de l’opération.   **Post condition :**   1. Le membre authentifié possède une nouvelle annonce. 2. Les produits sont ajoutés à l’annonce et ils sont à l’état en attente ; ils devraient d’abord être validés par l’administrateur pour qu’ils s’affichent aux publics. 3. Les images relatives à un produit sont ajoutées à un dossier portant le nom du produit. 4. Le dossier produit se trouve sous un répertoire contenant le nom de l’utilisateur.   **Exception1 :** l’utilisateur est à l’étape 7 du scénario principale.   1. L’utilisateur clique sur annuler au lieu de suivant. 2. Le système redirige l’utilisateur vers la page ajouter une annonce. 3. Aucune modification dans la base. |

#### 3.5.4.4. Description textuelle du cas d'utilisation « Ajouter un commentaire »

|  |
| --- |
| **Titre**: Ajouter un commentaire  **Acteur principal**: Membre  **Résumé** : Ajouter un commentaire sur un produit.  **Pré-conditions :**   1. Membre authentifié 2. Membre connecté sur la page Information sur le Produit   **Scénario Principal:**  1. Le membre possède la possibilité de commenté le produit.  **Post-conditions :**   1. Commentaire ajouté. |

**3.5.4.5. Description textuelle du cas d'utilisation « Approuver un produit »**

|  |
| --- |
| **Titre**: Approuverun produit  **Acteur principal**: Administrateur  **Résumé** : A travers ce cas, l’administrateur va accepter/rejeter les produits ajoutés par les membres.  **Pré-conditions :**  1. Administrateur authentifié  **Scénario Principal:**  1. L’administrateur demande de consulter la liste des produits en attente.   1. L’administrateur sélectionne plusieurs produits. 2. L’administrateur clique sur le bouton accepté 3. Tous les produits sélectionnés passent à l’état Accepté.   **Post-conditions :**   1. Un produit accepté devient consultable par tout le monde. 2. Une annonce contient des produits marqué par Accepté ou Rejeté ; lorsque le membre consulte ses produits. 3. Un produit rejeté n’est vu que par l’administrateur et le membre concerné jusqu’il soit supprimer ou passer à l’état accepté.   **Exception :** reprenons l’étape 1 du scénario principal   1. L’administrateur ne sélectionne aucun produit. 2. L’administrateur clique sur le bouton accepté ou rejeté 3. Le système affiche un message d’erreur : « aucun produit est sélectionné » |

#### 3.5.4.6. Description textuelle du cas d'utilisation « Supprimer un produit »

|  |
| --- |
| **Titre**: Supprimer un produit  **Acteur principal**: Administrateur  **Résumé** : l’administrateur peut supprimer n’importe quel produit existant dans la base de données.  **Pré-conditions :**   1. Administrateur authentifié   **Scénario Principal:**  1. L’administrateur demande de consulter la liste des produits.  2. L’administrateur clique sur le bouton Supprimer qui s’affiche à côté du produit.  3. le système affiche un message « Voulez-vous vraiment supprimer ce produit définitivement »  4. L’utilisateur clique sur le bouton Oui.  **Post-conditions :**   1. Produit supprimé définitivement. 2. Annonce modifiée ; ne contient plus ce produit. |

## 3.6. Conclusion

L’analyse des besoins nous a permis d’avoir une vision plus claire sur le sujet et une compréhension plus profonde des tâches à réaliser. La spécification fonctionnelle et non fonctionnelle des besoins va être la base sur laquelle nous allons réaliser la conception du système qui va nous faciliter l’étape d’implémentation que nous allons aborder dans le chapitre suivant « Réalisation ».

|  |
| --- |
| CHAPITRE 4 :  REALISATION  DE LA SOLUTION |

Plan du chapitre :

Introduction

Architecture générale de la solution

Le diagramme de paquetage

Environnement de travail

Sprint 0 : Installation et configuration de l’environnement du travail

Sprint 1

Sprint 2

Sprint 3

Conclusion

# Chapitre 4 : Réalisation de la solution

## 4.1. Introduction

La réalisation est une étape très importante dans le cycle de vie de toute application informatique. Comme nous avons utilisé SCRUM, nous avons divisé notre application en quatre livrables vu que le premier sprint est l’installation et la configuration de l’environnement de développement. Par de conséquent, nous avons quatre sprints. Chaque sprint a duré trois semaines et traite trois volets à savoir :

* La conception
* Le codage
* Les tests.

Nous commençons ce chapitre par présenter la conception architecturale de l’application. La partie suivante comprendra le diagramme de paquetages UML que nous avons utilisé. Ensuite, nous allons présenter l’environnement matériel et logiciel de la solution.

Nous entamerons ensuite le premier sprint. Où nous présenterons les différents tests techniques (Installation, configuration et intégration) des outils déjà présentés dans le deuxième chapitre. Pour le deuxième, troisième et quatrième sprint, nous allons présenter la conception, le développement, les tests et les rétrospectives.

## 4.2. Architecture générale de la solution

Afin d’éviter les problèmes de surcharge et de garantir la réutilisation du système, nous avons choisi de recourir à la séparation en couche logicielle du système en le divisant en sous-parties spécialisées. Ces couches logicielles sont structurées en niveau de telle sorte qu’une couche d’un niveau i ne peut utiliser que les couches de niveaux inférieurs.

* Couche présentation (partie client se repose sur le MVVM): correspond à la partie de l'application visible et interactive avec les utilisateurs.
* Couche métier (BLL) : représente les objets du métier et implémente leurs règles de gestion.
* Couche d’accès aux données (DAL): chargée de communiquer avec les bases de données, en exécutant sur celles-ci des requêtes de consultation, de suppression ou modification.

La communication entre les différentes couches dépendra de l’architecture physique de l’application. Etant donné que les couches, dans notre cas, se trouvent sur des machines physiquement distinctes, une architecture orientée services est mise en œuvre.

L’architecture SOA représente un mode d’architecture fondé sur le principe de séparation de l’activité métier en une série de services. Ces services peuvent être assemblés et liés entre eux selon le principe de couplage lâche pour exécuter l’application désirée.

Afin que notre système puisse appliquer l’architecture SOA, nous avons conçu une couche supplémentaire sous la forme de services. Ainsi, la couche présentation qu’on vient de concevoir ne manipule plus directement les objets métiers, mais passe forcément par des services. Les objets métiers se trouvant dans des bibliothèques de classes sont directement chargées dans le même processus que les services ainsi le coût des appels aux objets métiers est de plus en plus faible.

Les services agissent comme des boîtes noires. En effet, ils font abstraction de la complexité du modèle objet, présentent un ensemble de fonctionnalités restreint et permettent la réduction des échanges entre les couches et le nombre d'appels au serveur. Nous avons ajouté également le module « Common » qui contient les classes partagées entre les différentes couches. La communication entre la couche service et la couche présentation s’établit à travers des objets de transfert de données (DTO14). La Figure suivante explique l’architecture complète de notre projet :

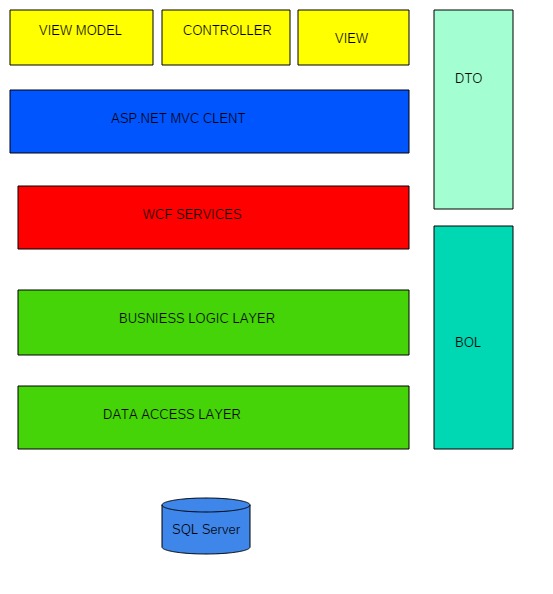


Figure 6 : Architecture générale de l’application

### 4.2.1. Introduction aux design patterns

« Un design pattern est une solution de conception commune à un problème récurrent dans un contexte donné »

Les designs patterns permettent de répondre à un problème de conception grâce à une solution éprouvée et validée par des experts. Ils facilitent l’évolution du code et sa réutilisation ainsi que l’indépendance vis à vis des plateformes matérielles et logicielles.

C’est pourquoi leur emploi apporte une évolutivité, une lisibilité et une efficacité aux développements.

**Le design pattern « DTO »**

Dans une application distribuée, nous sommes obligés d’émettre un nombre important d’appels vers le serveur afin de satisfaire une requête client mais ceci accroît le temps de réponse. La meilleure idée pour éviter ce problème est de créer un DTO qui contient toutes les données nécessaires à l’appel distant.

Un DTO est un simple conteneur pour un ensemble de données agrégées qui doivent être transférées dans un autre processus ou un autre réseau. Il ne doit pas contenir de logique métier et doit limiter son comportement à des activités de vérification de la cohérence interne et de validation de base.

**Le design pattern « DAL »**

Le pattern DAL permet d’établir le lien entre deux couches : la couche métier et la couche persistante. Le DAL centralise les mécanismes de mapping entre le système de stockage et les objets. Il permet également de prévenir un changement probable de système de stockage de données.

En effet, Il cache la connexion à la base de données ou tout autre outil d’accès aux données et fournit les méthodes d’ajout d’objet (sauvegarde en base de données), lecture d’objet (récupération de données depuis la base de données), modification et suppression d’objet.

**Le design pattern « MVVM »**

La méthodologie MVVM est une variation du patron de conception MVC, taillée sur mesure pour les technologies modernes d'interface utilisateur où la réalisation de la Vue est davantage confiée à un designer qu'à un développeur classique. Le terme designer désigne ici un artiste davantage concerné par les aspects graphiques et d'expérience utilisateur que par l'écriture de code. Le design en question est réalisé en langage déclaratif à l'aide d'outils spécialisés. Ce pattern d’architecture s’appuie très largement sur le principe model-vue-contrôleur. La signification de MVVM est « Model, View, View-Model » ce qui correspond aux 3 couches qui composent ce modèle :

* **La couche « Model » :** est basiquement une couche d’accès aux données. Elle contient les méthodes qui permettent de récupérer et traiter les données utilisées par l’interface graphique. Dans notre cas, il peut s’agir de la couche qui fait l’interface avec des web services par exemple. Le « Model » contient le métier de l’application.
* **La couche « View-Model » :** s’agit de la couche faisant l’interface entre la couche « View » et « Model ». On pourrait résumer la fonction de la couche View-Model à présenter la couche « Model » de façon à ce qu’elle soit directement consommable: Vue « programmatique » des données, repository d’actions (sous forme de triggers par exemple…). Le « View-Model » contient la logique IHM de l’application.
* **La couche « View » :** on retrouve dans la couche « View » ce qui compose l’IHM d’un point de vue graphique et statique. Parmi les préconisations du pattern MVVM, il est mentionné qu’il ne doit pas y avoir de code présent dans cette couche. On insiste lourdement sur cette mention qui selon moi est une des pierres fondatrices de ce pattern. Outre l’intérêt souvent évoqué de simplifier les tests unitaires, le respect de cette mention permet d’une part une réelle organisation du code et permet d’autre part une collaboration Développeur/Designer et Développeur/Développeur optimale. Le « View » contient le contenu graphique de l’application.

### 4.2.2 Le diagramme des paquetages

La figure 7 présente le diagramme de paquetage modélisant les différents modules de l’application à savoir :

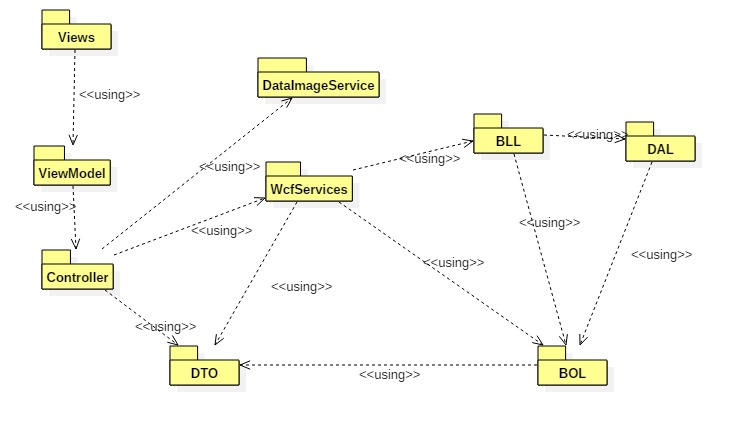


Fig.7 : Diagramme de paquetage global

## 4.3. Environnement de travail

### 4.3.1. Environnement matériel

Dans la réalisation de notre application, nous avons disposé d’un ordinateur portable caractérisé par :

* Fabriquant : Lenovo.
* Processeur : Intel CORE i7 (2.0 - 3.1 GHz).
* Disque Dur : 1 To.
* RAM : 6 Go.

### 4.3.2. Environnement logiciel

Cette solution a été développée sous Windows 10, 64 bit.



**Choix de langage de programmation**

Dans l’environnement .Net, plus d’une vingtaine de langages de programmation sont fournies. Les plus célèbres sont :

* C++ : Dans deux versions : native et modiﬁée pour supporter le .Net. Bien qu’elle soit orientée objet, elle a devenu vieille et freine la productivité de développeur.
* VB .Net : Le VB a été toujours un langage très supporté par Microsoft. Sa dernière version est compatible .Net, mais elle conserve quelques mauvaises habitudes de programmation héritées des versions précédentes.
* C# : C’est un langage récent Il a été disponible en versions beta successives depuis l’année 2000 avant d’être oﬃciellement disponible en février 2002 en même temps que la plate-forme .NET 1.0. Le choix s’est porté sur le C# pour deux grands privilèges :
* Il reﬂète le mieux l’architecture .Net. Il a été créé dans le but de fournir un langage capable d’exploiter toutes les capacités de l’infrastructure de framework.
* C# est proche syntaxiquement des langages populaires (C++ et Java).



**Choix de la technologie ASP.NET MVC 5**

ASP.NET est une technique pour générer à la demande des pages web, lancée par Microsoft en juillet 2000, et utilisée pour mettre en œuvre des applications web. Il s'agit d'une évolution majeure d'Active Server Pages (abréviation ASP), par laquelle cette technique a été incorporée dans la plateforme Microsoft .NET.

Il y a plusieurs raisons de choisir ASP.NET MVC pour le nouveau développement :

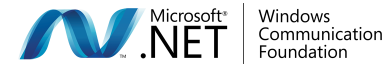
1. simplicité

2. Facilite la façon de développer une application web

3. Facilite la maintenance

4. Facilite l'implémentation de tests unitaires

5. S'adapte bien à la pratique de développement Agile

****

**Choix de la technologie de la communication distribué**

Notre application se base sur une architecture client/serveur, donc nous avons besoin d’une technologie qui implémente les services et la communication distribuée. WCF est la toute dernière des technologies de communication inter-applications proposée par Microsoft. Le modèle de programmation WCF est formé d’une couche d'abstraction unifiant et simplifiant la mécanique d'intégration des services web. Le développeur conçoit son service sans tenir compte de son implémentation à cible : il ne s'intéresse qu'aux caractéristiques structurantes du service pour son intégration au sein d'une architecture SOA. Les applications WCF peuvent être développées à l’aide de plusieurs langages de Microsoft .NET. Nous avons opté pour le choix de WCF pour ses diﬀérents avantages :

* WCF hérite d’une importante expérience de la part de Microsoft qui s’appuie sur les implémentations antérieures de la communication distribuée (COM, DCOM, .Net Remoting,..).
* Séparation claire des couches logicielles en trois éléments :
  + A (Addressing) : l’identiﬁcation des ressources physiques et support des plusieurs protocoles de communication (TCP, IP, HTTP, HTTPS, P2P,..).
  + B (Binding) : Le binding déﬁnit le comportement des couches de transports. WCF présente l’avantage de permettre à l’utilisateur de personnaliser cet élément par des appels de bas niveaux.
  + C (Contracts) : La matérialisation des clients se fait par la déﬁnition du contrat de service.

- Intégrité des données transférées et contrôle des facteurs d’optimisation (Throttling des appels, des sessions et des instances).

- Des mécanismes de sécurité très variés comme le cryptage, l’authentiﬁcation, les signatures numériques, etc...

**Choix de la base de données SQL Server 2014 Express LocalDB:**

Tout au long de notre projet, nous avons utilisé **SQL Server 2014 Express LocalDB** comme base de données puisqu’ elle est livrée gratuitement et vu sa simplicité d’utilisation dans le projet.

****

**Choix de l’Entity Framework 6 comme un ORM (object relational mapping)**

Entity Framework (EF) est un ORM développé pour L’ADO.NET et qui à partir de sa version devient une partie du Framework .Net.

Un ORM (**object relational mapping**) réalise la conversion de l'Objet en Relationnel et du Relationnel en Objet sans tenir compte du types de la base de données utilisée. Dans la figure ci-après explique le fonctionnement d’un ORM :

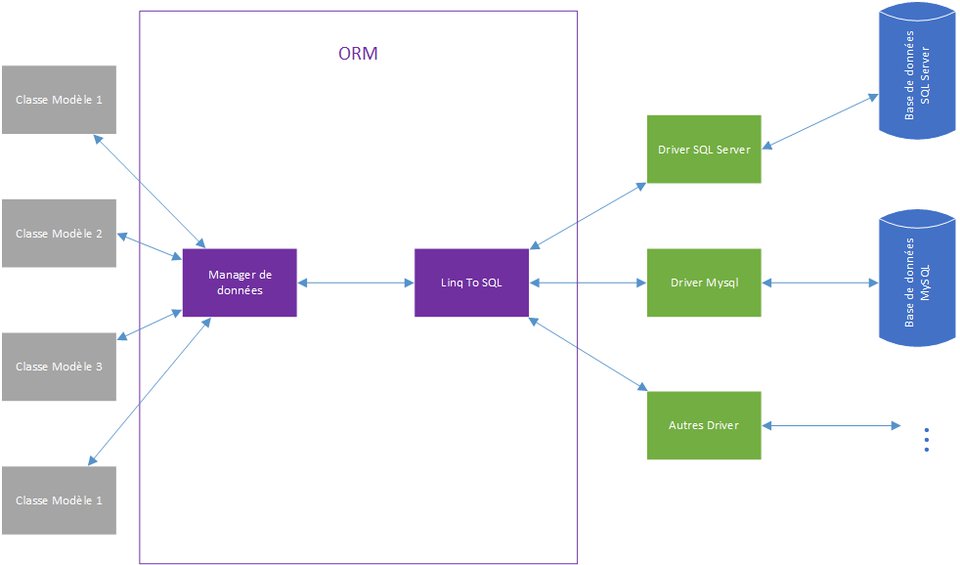
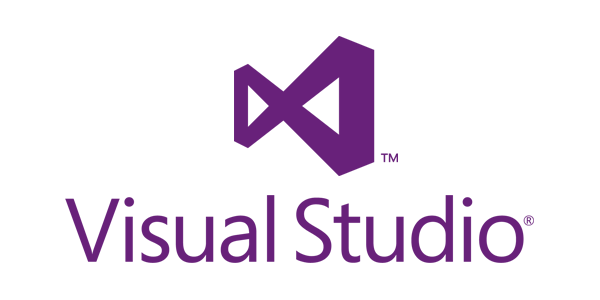


Fig.8 : Fonctionnement d’un ORM



**IDE (**Environnement de développement**) :**

Dans notre projet on utilise Visual studio 2015 professionnel

**4.4. Sprint 0 : Installation et configuration de l’environnement du travail**

Ce sprint présente la première étape dans le travail pratique qui consiste à faire les installations et les configurations nécessaires de l’environnement du travail. Nous allons voir comment préparer notre environnement de développement pour créer une application avec Asp.net MVC et WCF. L’installation et la configuration de l’environnement du travail a pris deux semaines de travail.

* Tout d'abord, nous téléchargeons Visual studio professionnel 2015 à partir de cette adresse: <https://www.microsoft.com/en-us/download/details.aspx?id=48147>
* Nous installons ensuite Microsoft SQL Server express 2014 à partir de l’adresse : <https://www.microsoft.com/en-us/download/details.aspx?id=42299>
* Installation du serveur IIS dans sa version 7.5.
* À l’aide de Visual Studio création d’une nouvelle solution contenant des projets de type:
* Asp.net MVC
* WCF
* DLL Library
* Ajout des DLL référençant Entity Framework pour la gestion de la base de données.
* Ajout des fichiers relatifs au Bootstrap via Nuget pour l’amélioration de la partie FrontEnd.

## 4.5. Sprint 1 : Gestion des utilisateurs

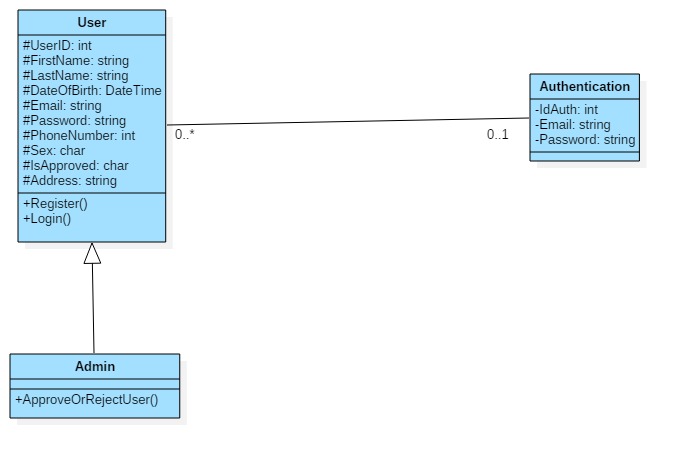
Le but de ce sprint consiste à développer le module nécessaire à la gestion des utilisateurs. Chaque internaute peut accéder au site et s’inscrire, puis l’administrateur se connecte et il accepte ou refuse les inscriptions.

### 4.5.1. Conception

Nous allons présenter la conception statique et dynamique du sprint 1 notamment le diagramme de classe et les diagrammes des séquences de différents scénario.

**Diagramme de classe**

Nous présentons le diagramme des classes du sprint « Gestion des utilisateurs ».

Fig.9 : sprint1 : diagramme des classes

**Diagramme des séquences**

Un diagramme de séquence montre chronologiquement (de haut en bas) les interactions entre un ensemble d'objets d’un cas d’utilisation spécifique. Ces objets sont en général des instances de classes mais ils peuvent également représenter des acteurs.

La figure 10 décrit le scénario du cas d’utilisation « Demande d’inscription » de Gestion des utilisateurs :

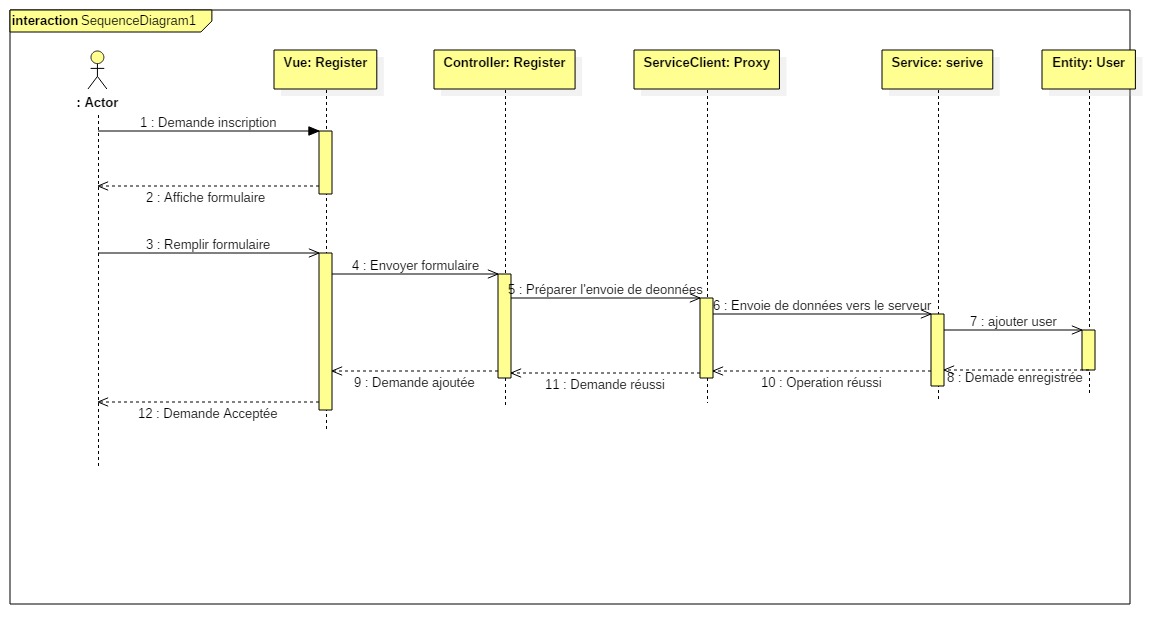


Fig.10 : Diagramme de séquence : Opération d’enregistrement

La figure suivante explique la phase d’authentification de point de vue outils utilisées.

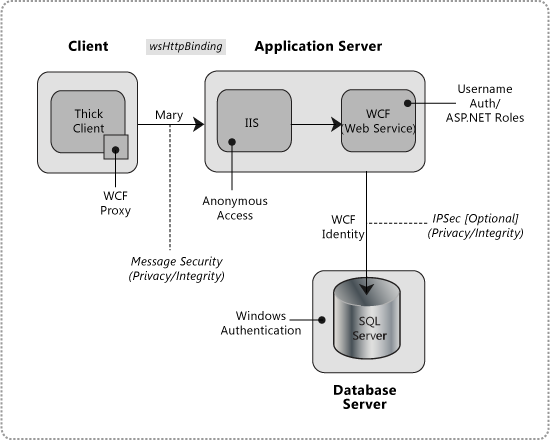


Fig.11 : Procédure d’authentification via Web Service

La figure 12 décrit le scénario du cas d’utilisation « Accepter demande » de Gestion des utilisateurs

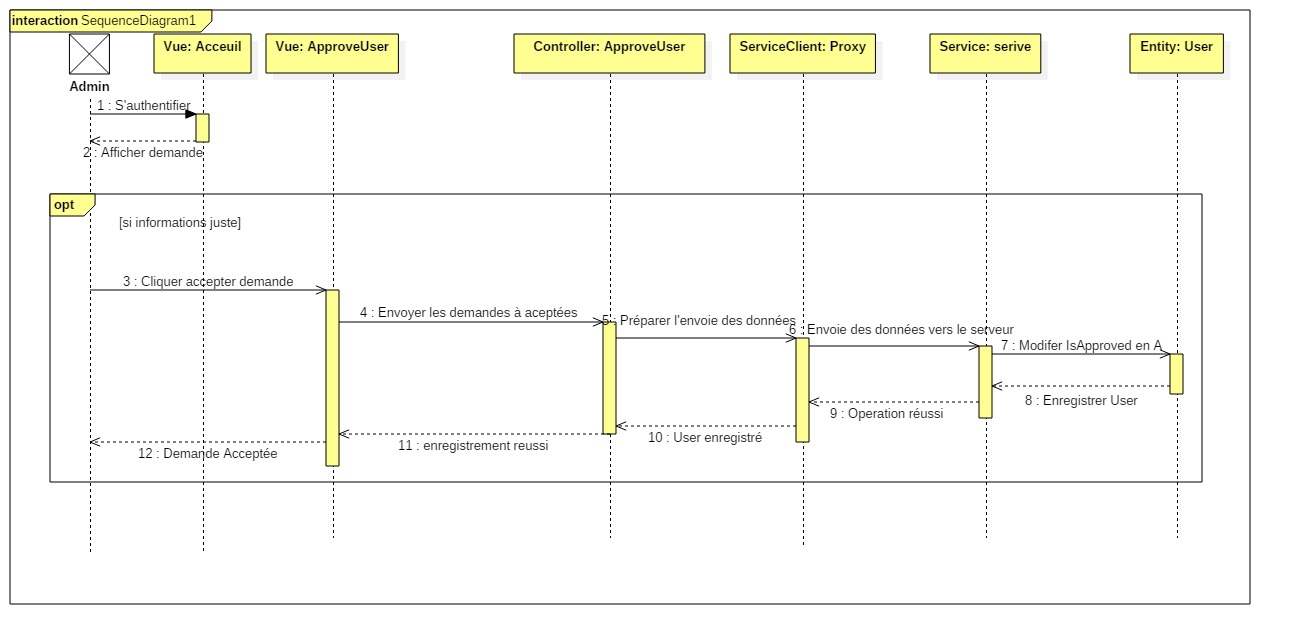


Fig.12 : Diagramme de séquence : Acceptation d’un nouveau utilisateur

### 4.5.2. Développement

Cette partie présente les étapes du développement des interfaces pour la gestion des utilisateurs. La figure 13 présente l’interface d’inscription.

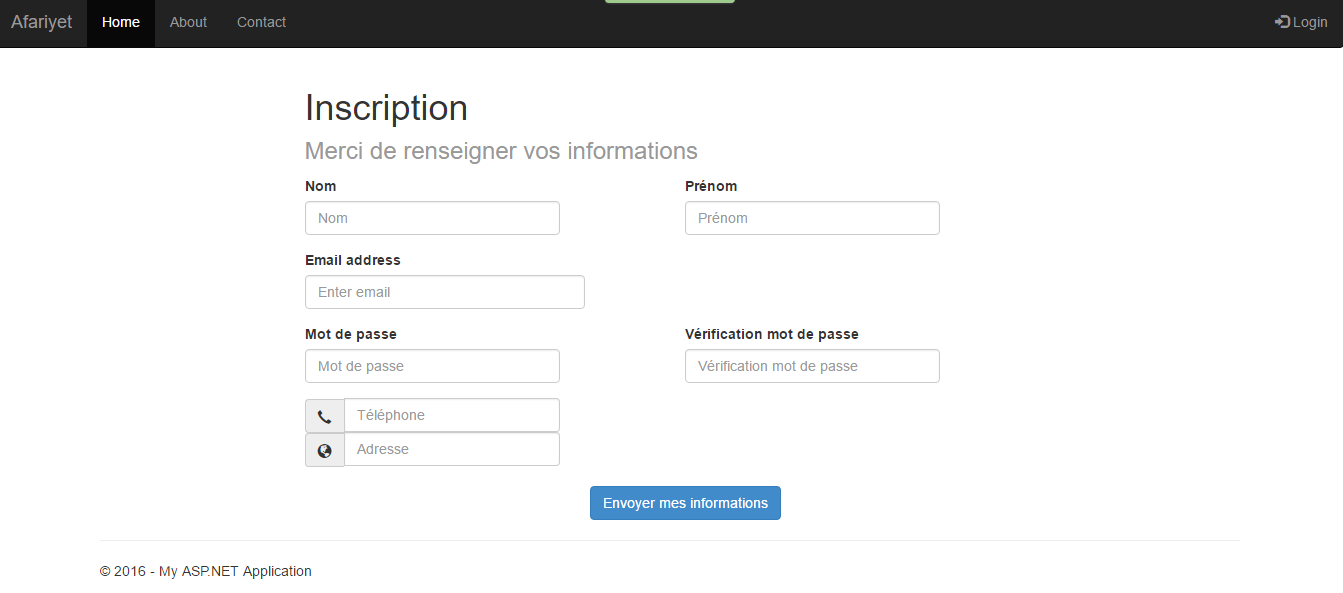


Fig.13 : Page d’Enregistrement

La figure 14 présente d’interface de la liste des demandes en attentes d’inscription. Les deux icones à droite, une pour l’acceptation et l’autre pour le refus.

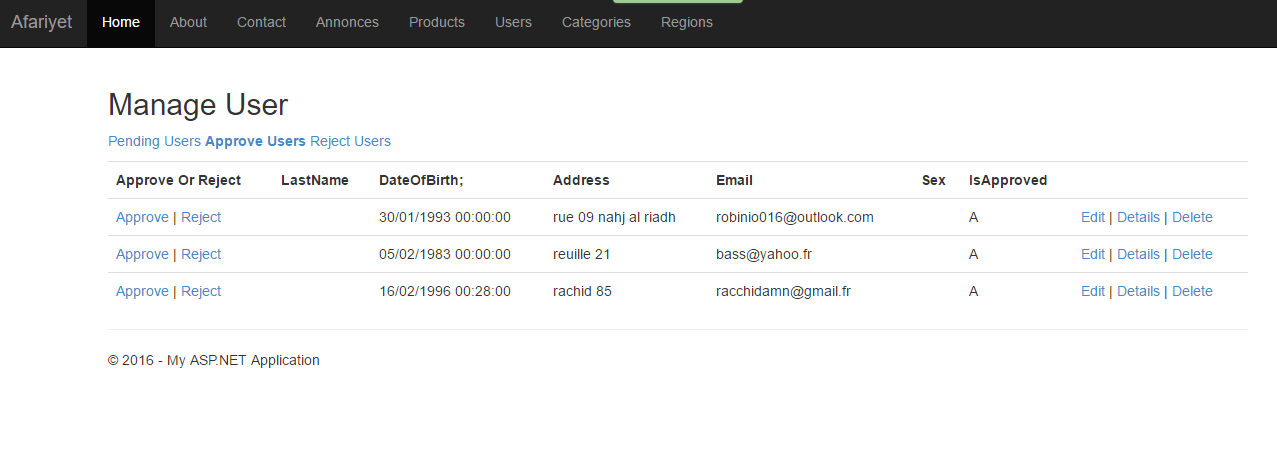


Fig.14 : Page d’acceptation des nouveaux utilisateurs par l’administrateur

### 4.5.3. Test

En fin de chaque sprint notre travail doit être intégré, testé et validé. Cette tâche revient en premier lieu par notre chef de projet ensuite par le Product Owner et le Client. Dans ce sprint on a testé l’Ergonomie : si le module respecte les aspects ergonomiques.

## 4.6. Sprint 2 : Gestion des annonces

Ce sprint consiste à développer le module de gestion des annonces des catégories et des régions.

### 4.­6.1. Conception

**Conception diagramme des classes de la partie client vers la partie serveur**

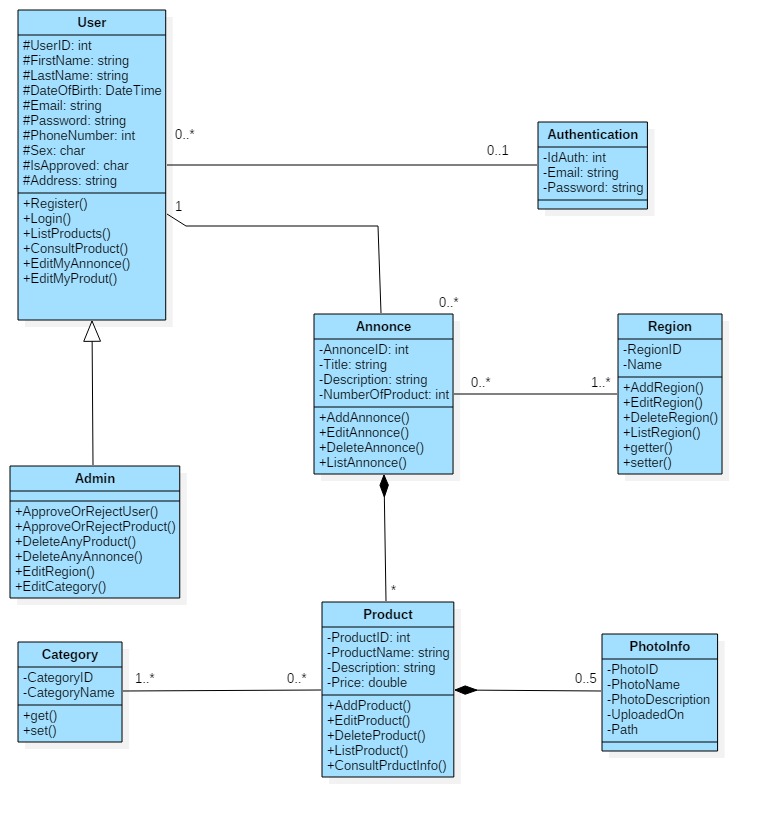
****

Fig.15  : Diagramme des classes de la sprint 2

**Conception dynamique diagramme de séquence : ajouter une annonce, ajouter une région, ajouter une catégorie.**

La figure suivante décrit le cas d’utilisation « ajouter une annonce ».

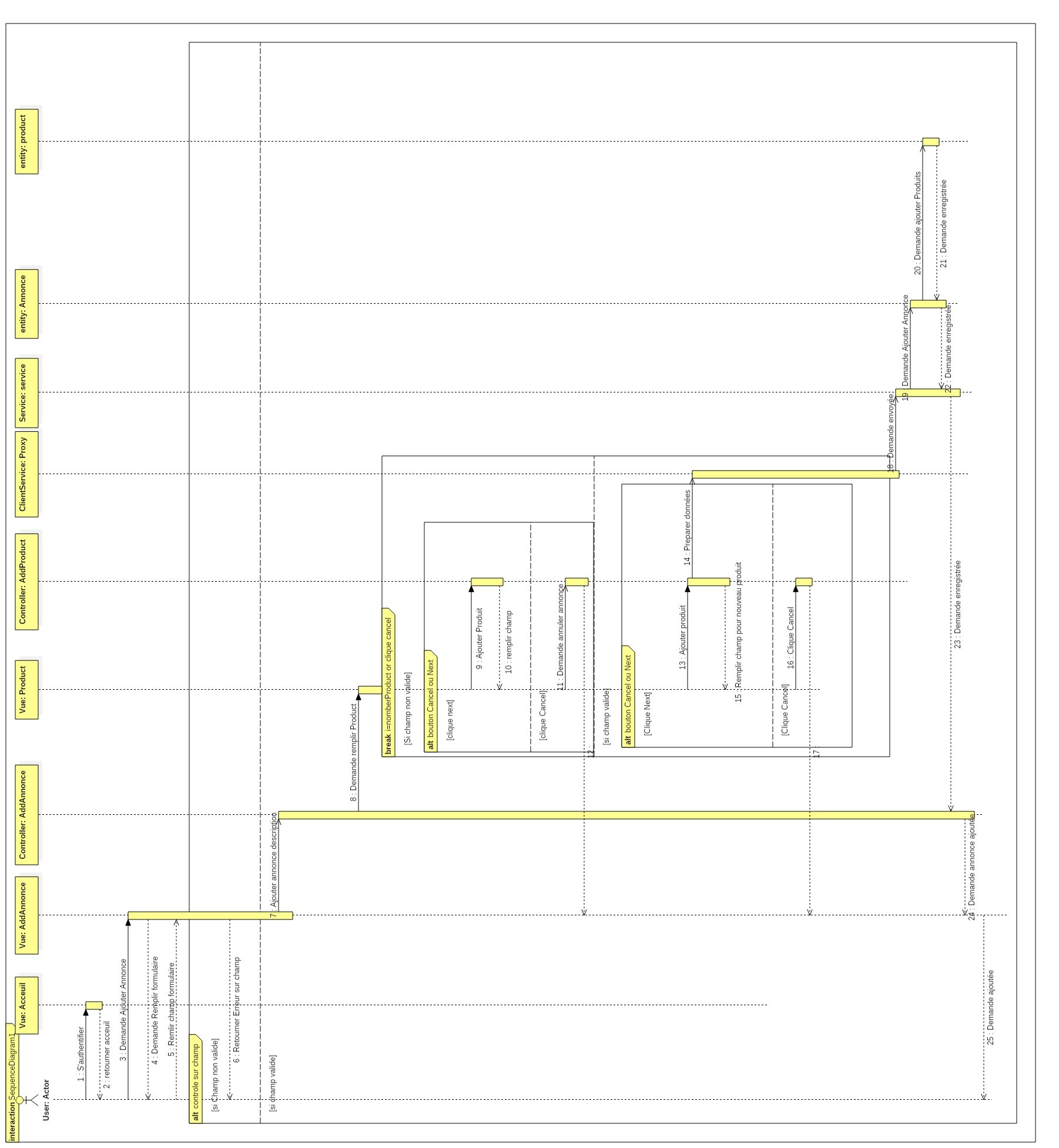


Fig.15 : Diagramme de séquence : ajout d’une annonce

### 4.­6.2. Développement

La figure suivante représente l’interface réelle du site de l’application développée concernant l’ajout d’une nouvelle annonce.

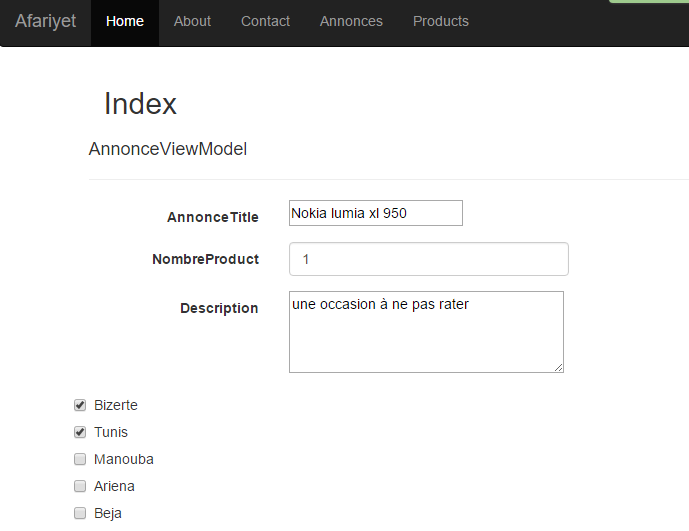
****

Fig.16 : Page d’ajout d’une annonce

Après la clique sur le bouton « Suivant », l’utilisateur peut ajouter autant de produits que le nombre précisé dans l’interface précédente.

La figure ci-après représente l’interface développée à ce stade.

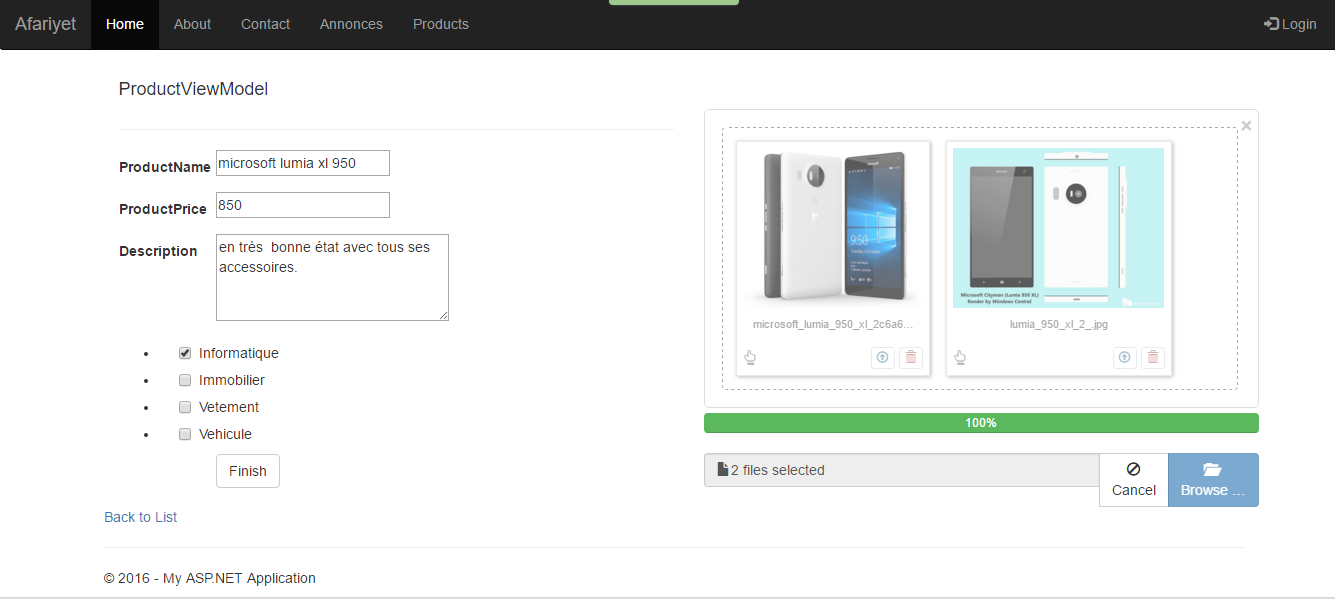
****

Fig.17 : Page d’ajout d’un produit à une annonce

### 4.­6.3. Test

A la fin de ce sprint on a fait deux types de teste :

* **Contrôle** : pour tester les contrôles sur les champs.
* **Affichage** : pour tester si l’affichage demandé lors de la spécification a été respecté

## 4.7. Sprint 3 : Enrichir l’application par d’autres services

Rendre l’application plus interactive avec l’utilisateur en lui fournissant un espace pour comparer les produits et un espace pour contacter l’utilisateur.

### 4.7.1. Conception

**Conception diagramme des classes de la partie client vers la partie serveur**

La figure suivante représente le diagramme des classes nécessaires pour l’ajout des fonctionnalités tels que : Ajout des commentaires, contact du vendeur etc.

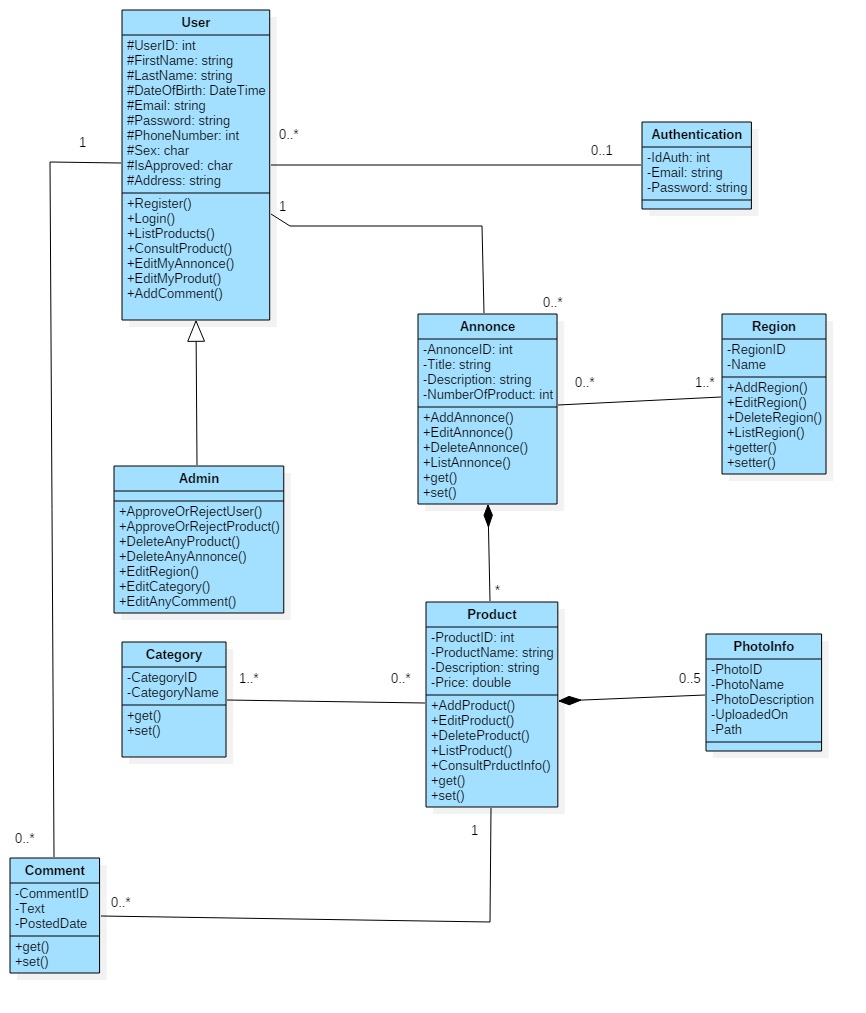
****

Fig.18 : Diagramme des classes pour le sprint 3

### 4.7.2. Développement

La figure ci-dessous montre l’interface développée pour contacter le vendeur

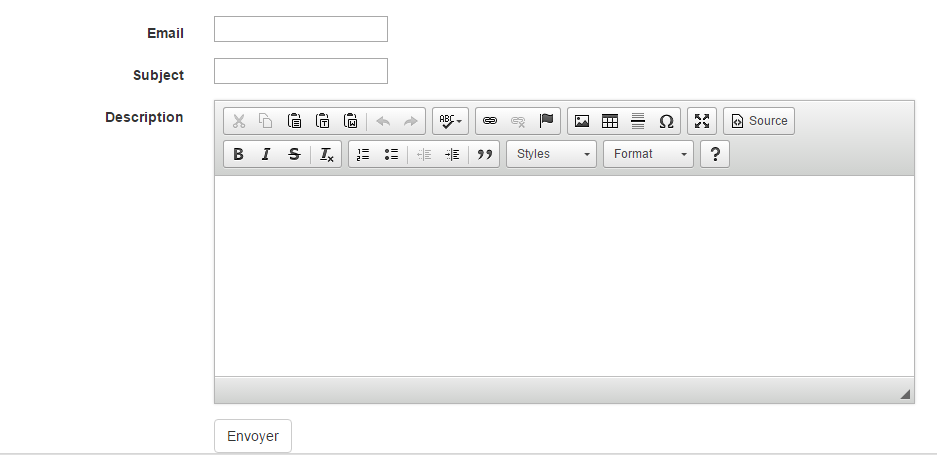
****

Fig. 19 : Page pour contacter le vendeur

Les deux captures d’écran suivantes illustrent la page reliée aux informations concernant un produit ajouté.

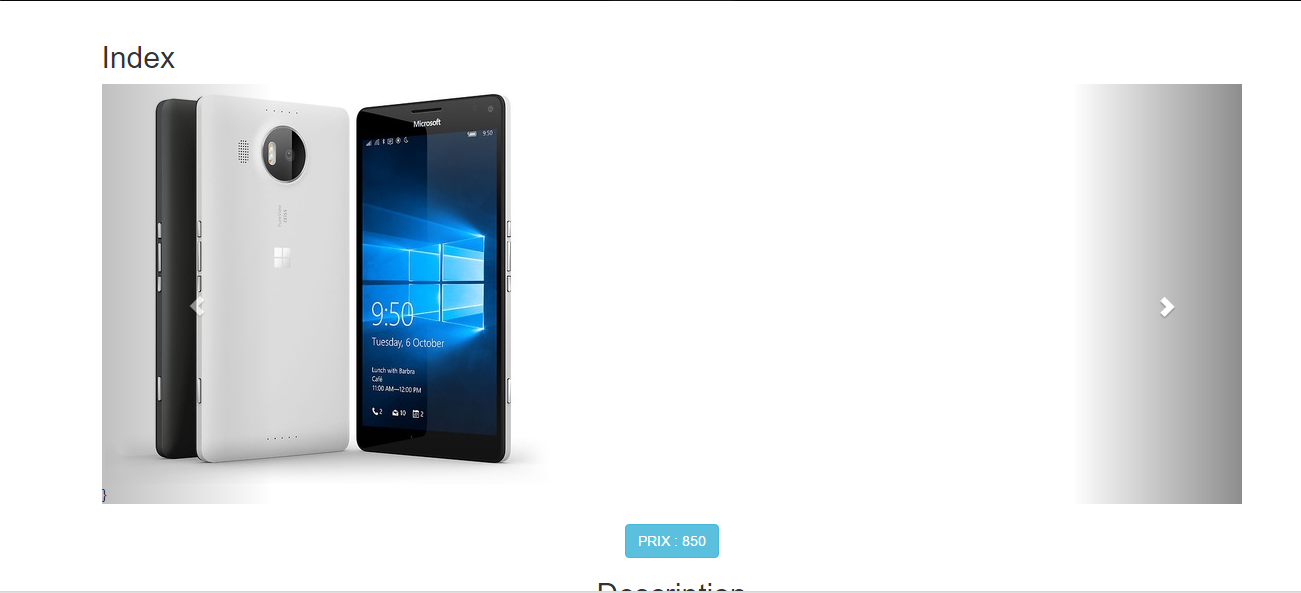
****

Fig. 20 : Informations sur le produit

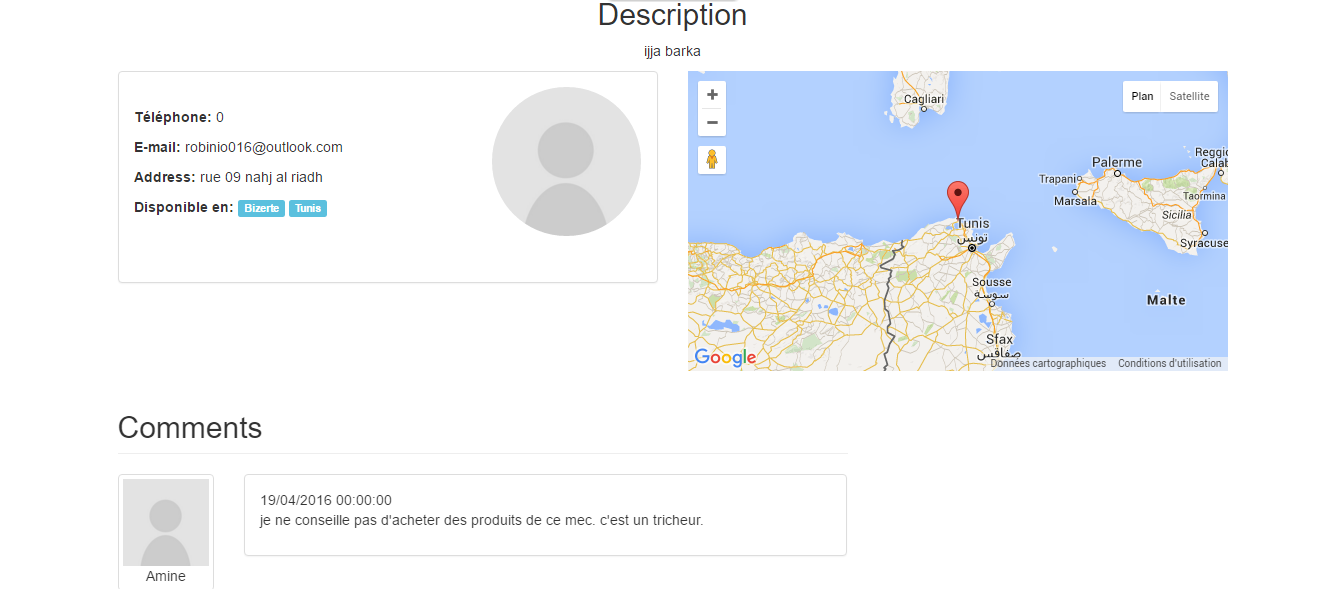


Fig. 21 : Information sur le produit et ajout des commentaires

## 4.8. Conclusion

Au cours de ce chapitre, nous avons présenté notre architecture générale de l’application. Ensuite, nous avons présenté le diagramme de paquetages et les plateformes logicielles sur lesquelles nous avons construit notre application.

Nous avons ensuite présenté nos sprints un par un selon la méthodologie de gestion de projet SCRUM, en décrivant la conception, le développement, les interfaces les plus importantes pour chaque sprint, ainsi que les tests.

|  |
| --- |
| **CONCLUSION GENERALE** |

# Conclusion générale

Avec la progression continue de l’informatique et plus spécifiquement le développement Web, les entreprises essayent toujours d’exploiter de nouvelles applications qui touchent plusieurs départements afin de les mettre à la disposition des clients des applications web selon leurs besoins.

Notre projet vise à concevoir et réaliser une solution complète qui met à la disposition des utilisateurs toutes les fonctionnalités nécessaires pour faciliter les achats et les ventes des objets personnels en garantissant la sécurité et fiabilité.

Pour parvenir à ce résultat, nous avons tout d’abord commencé par la présentation générale du projet en présentant les solutions existantes ayant la possibilité de répondre aux besoins des acheteurs et des vendeurs.

Nous avons ensuite, présenté les différents besoins et exigences relevés, la conception en commençant par l’architecture adoptée, pour obtenir ensuite une conception détaillée qui met l’accent sur l’aspect statique et dynamique du système décrivant minutieusement l’aspect métier de la solution.

Finalement, nous avons abordé l’étape de réalisation au cours de laquelle nous avons traduit notre modélisation conceptuelle en une implémentation physique moyennant les différentes technologies et techniques.

Sur le plan des nouvelles technologies, cette expérience nous a permis de découvrir et d'acquérir des connaissances nouvelles, de renforcer nos connaissances programmation, et d'autre part, nous avons appris à travailler dans l'environnement AGILE, particulièrement avec la méthode SCRUM.

Plusieurs perspectives peuvent être entrevues. En effet, on envisage de perfectionner certaines fonctionnalités que nous n’avons pu bien finaliser pour contraintes liées au temps, telles que :

* + - * + L’utilisation d’Ajax,
        + Traitement plus sophistiquée de la partie sécurité, etc.

En conclusion, ce projet nous a offert l’opportunité d’avoir un avant-gout sur la vie professionnelle et de nous familiariser avec la pratique et ses différentes contraintes.

|  |
| --- |
| **BIBLIOGRAPHIE** |

# Bibliographie

Adam Freeman, Pro ASP.NET MVC 4, Apress.

Jeffrey Palermo,Jimmy Bogard, Eric Hexter, ASP.NET MVC 4 IN ACTION, Manning

Tom Dykstra, Rick Anderson, Getting Started with Entity Framework 6 Code First using MVC 5, Microsoft

|  |
| --- |
| **WEBOGRAPHIE** |

# Webographie

<https://www.asp.net/mvc/overview/getting-started>

<https://msdn.microsoft.com>

<https://mva.microsoft.com/>

<http://www.tutorialspoint.com/>

<http://www.wcftutorial.net/>

<https://github.com/>

<http://www.entityframeworktutorial.net/>

<http://www.jambuster.in/ResourcesAgile.html#agilescrum>

<http://2.bp.blogspot.com/>

<http://www.bootply.com/fhQcmTInE4>