République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira-Béjaïa

Faculté des Sciences Exactes

Département Informatique

Mémoire de fin de cycle

En vue d'obtenir le diplôme de master professionnel en génie logiciel

Thème

Conception et réalisation d'une application Mobile dédié à la location « ValizaDz »

Réalisé par :

TADJINE Lounis

YAHIAOUI Ahcene

Devant le jury composé de :

Présidente : BOUKERRAM Samira

Examinatrice : GHANEM Souhila

Encadrante : KHALED Hayette

Promotion 2020/2021

Remerciements

Nous tenons tous d'abord à remercier vivement et à dédier ce travail à nos familles respectives et particulièrement à nos parents pour leurs soutiens qu'ils nous ont accordés tout au long de notre parcours.

Nous tenons à exprimer nos vifs remerciements envers notre encadreur ***Madame H. Khaled*** *pour sa disponibilité, son encadrement, sa confiance et les conseils qu'elle nous a généreusement prodigués.*

Nous remercions tout particulièrement les membres du jury qui ont accepté de juger notre travail ainsi que tous les enseignants qui ont contribué à notre formation.

Nous remercions aussi ***Mr D.Bouagache*** *Maître d'ouvrage et Directeur de l'entreprise* ***BDE*** *pour sa confiance, et de nous avoir permet de travailler sur l'application «* ***ValizaDz*** *» .*

Nous souhaitons également remercier l'entreprise ***VegaSoft*** *et plus particulièrement* ***Mr R.Badja*** *pour ses orientations et conseils.*

Enfin, nous tenons à remercier tous ceux qui ont contribué d'une façon ou d'une autre à la réalisation de ce mémoire.

Dédicaces

Ce modeste travail est dédié :

A nos chers parents qui nous ont soutenus et encouragés durant toute notre scolarité.

A nos frères et sœurs

A nos familles

A nos enseignants

A nos ami(e)s

Au personnel de l'école THE SOURCE.

A toutes les personnes qui nous ont apportés de l'aide.

Table des matières

Introduction

Générale.....10

CHAPITRE I : État de l'Art

I.1 Introduction 14

I.2 Critique de L'existant 14

I.2.1 ACR 14

I.2.1.1 Critiques (ACR) 15

I.2.2 seLoger 16

I.2.2.1 Critiques (seLoger) 17

I.3 Problématique 20

I.4 Les Besoins 20

I.5 Les Objectifs du Projet 21

I.6 Conclusion 21

II.1 Introduction 24

II.2 Spécification des Besoins 24

II.2.1 Les Besoins Fonctionnels 24

II.2.2 Les Besoins non Fonctionnels 26

II.3 Analyse des Besoins 27

II.3.1 Identification des Acteurs 27

II.3.2 Identification des Messages 28

II.3.2.1 Les Messages Entrants Echangés avec le Système 28

II.3.2.2 Les Messages Sortant Echanger avec le Système 29

II.3.3 Diagramme du Contexte Dynamique 30

II.3.4 Identification des Cas d'Utilisation 31

II.3.5 Diagramme de Cas d'Utilisation 32

II.4 Conclusion 33

III.1Introduction 36

III.2 Modélisation Dynamique 36

III.2.1Diagramme d'Etat-Transition 36

III.2.2 Les Diagrammes de Séquences 37
III.2.2.1 Cas d'Utilisation « S'authentifier » 37
III.2.2.2 Cas d'Utilisation « Ajouter une annonce » 40
III.2.2.3 Cas d'Utilisation « Consulter une annonce » 43
III.2.2.4 Cas d'Utilisation « Contrôle annonce » 44
III.2.2.5 Cas d'Utilisation « Contrôle client » 47
III.2.2.6 Cas d'Utilisation « S'inscrire » 49
III.3 Modélisation Statique 50
III.3.3 Diagramme de Classes 50
III.3.4 Modèle Logique de Données 51
III.3.5 Dictionnaire de Données 52
III.4 Conclusion 58
IV.1Introduction 61
IV.2 Environnement de Logiciels 61
IV.3 Environnement du Développement 61
IV.3.1Plateformes 61
IV.3.2 Logiciels 61
IV.3.2.1 Logiciels du Développement 61
IV.3.2.2 Logiciels de modélisation 62
IV.3.2.3 Logiciels d'édition de textes 62
IV.3.2.4 Logiciels de traitement d'images 62
IV.3.3 Technologies utilisées 62
IV.4 Caractéristiques 64
IV.4.1Caractéristiques Fonctionnelles 64
IV.4.2 Caractéristiques Non-Fonctionnelles 65
IV.5 Manuel de l'Application 65
IV.5.1IHM du Visiteur 65
IV.5.2 IHM du Client 68
IV.5.3 IHM de l'Administrateur 71
IV.6 Conclusion 74
Conclusion Générale 77

Table des Figures

Figure I.1 : Logo de Algerie Car Rental [1]. 14
Figure I.2 : Logo de seLoger [2]. 14
Figure I. 3 : Interface Recherche de l'application « ACR ». 15
Figure I. 4 : Interface Accueil de l'application « ACR ». 15
Figure I. 5 : Capture d'écran d'un avis de l'utilisateur 1 [1]. 15
Figure I. 6 : Capture d'écran d'un avis de l'utilisateur 2 [1]. 16
Figure I. 7 : Anglet Accueil « seLoger ». 16
Figure I. 8 : Anglet Recherche « seLoger ». 16
Figure I. 9 : anglet Mettre mon bien en location : problème rencontré. 17
Figure I. 10 : anglet Recherche de l'application « seLoger ». 18
Figure I. 11 : Capture d'écran d'un avis de l'utilisateur 3 [2]. 19
Figure I. 12 : Capture d'écran d'un avis de l'utilisateur 4 [2]. 19
Figure I. 13 : Capture d'écran d'un avis de l'utilisateur 5 [2]. 20

Figure II. 1 : Diagramme du contexte dynamique. 31
Figure II. 2 : Diagramme de cas d'utilisation. 33

Figure III. 1 : Diagramme d'état de transition d'une annonce. 36
Figure III. 2 : Diagramme de séquence pour le cas d'utilisation « S'authentifier ». 39
Figure III. 3 : Diagramme de séquence pour le cas d'utilisation « Ajouter une annonce ». 42
Figure III. 4 : Diagramme de séquence pour le cas d'utilisation « Consulter une annonce ». 44
Figure III. 5 : Diagramme de séquence pour le cas d'utilisation « Contrôle annonce ». 46
Figure III. 6 : Diagramme de séquence pour le cas d'utilisation « Contrôle client ». 48
Figure III. 7 : Diagramme de séquence pour le cas d'utilisation « S'inscrire ». 50
Figure III. 8 : Diagramme de classes. 51

Figure IV. 1 : Schéma explicative d'une application à une page. 64
Figure IV. 2 : L'interface « Formulaire d'inscription ». 66
Figure IV. 3 : L'interface « Annonces ». 67
Figure IV. 4 : L'interface « Consulter une annonce ». 68
Figure IV. 5 : L'interface « Connexion ». 69
Figure IV. 6 : L'interface « Ajouter une annonce ». 70
Figure IV. 7 : Interface « Réservation d'une annonce ». 71
Figure IV. 8 : Interface « Valider une annonce ». 72
Figure IV. 9 : Interface « Décliner une annonce ». 73
Figure IV. 10 : Interface « Annonce Admin ». 74

Liste des Tableaux

Tableau II. 1 : User stories « Inscription ». 24
Tableau II. 2 : User stories « Abonnement ». 25
Tableau II. 3 : User stories « Ajouter une annonce ». 25
Tableau II. 4 : User stories « Consultation ». 25
Tableau II. 5 : User stories « Réservation ». 26
Tableau II. 6 : User stories « Contrôler les clients ». 26
Tableau II. 7 : User stories « Contrôler les annonces ». 26
Tableau II. 8 : Les messages entrant échangés avec le système. 29
Tableau II. 9 : Les messages sortant échangés avec le système. 30
Tableau II. 10 : Identification des cas d'utilisation. 32

Tableau III. 1 : Description de cas d'utilisation « S'authentifier ». 38
Tableau III. 2 : Description de cas d'utilisation « Ajouter une Annonce ». 41
Tableau III. 3 : Description de cas d'utilisation « Consulter une annonce ». 43
Tableau III. 4 : Description de cas d'utilisation « Contrôle Annonce ». 45
Tableau III. 5 : Description de cas d'utilisation « Contrôle Client ». 48
Tableau III. 6 : Description de cas d'utilisation « S'inscrire ». 49
Tableau III. 7 : Le glossaire de la table « Connexion ». 53
Tableau III. 8 : Le glossaire de la table « Administrateur ». 53
Tableau III. 9 : Le glossaire de la table « Visiteur ». 53
Tableau III. 10 : Le glossaire de la table « Client ». 54
Tableau III. 11 : Le glossaire de la table « Annonce ». 55
Tableau III. 12 : Le glossaire de la table « Pays ». 55
Tableau III. 13 : Le glossaire de la table « Region ». 55
Tableau III. 14 : Le glossaire de la table « SousRegion ». 56
Tableau III. 15 : Le glossaire de la table « Catégorie ». 56
Tableau III. 16 : Le glossaire de la table « SousCatégorie ». 57
Tableau III. 17 : Le glossaire de la table « Reservation ». 57
Tableau III. 18 : Le glossaire de la table « Avis ». 57

Tableau IV. 1 : Recommandations minimales des smartphones. 61

Liste des Abréviations

ACR : Algerie Car Rental

AJAX : Asynchrones JavaScript And XML.

CSS : Cascading Style Sheets.

DOM : Document Object Model.

HTML : HyperText Markup Language.

IHM : Interface Homme-Machines.

MySQL : My Structured Query Language.

PHP : Hypertext PreProcessor.

SGBD : Système Gestion De Base Donnée.

SPA : Single page application.

SQL : Structured Query Language.

UML : Unified Modeling Language.

XP : eXtreme Programing.

Introduction Générale

De nos jours, les applications mobiles sont devenues un point central dans notre vie quotidienne, elles se multiplient chaque jour sur nos différents devices. On pourrait presque les appeler "raccourcis", présents la plupart du temps dans nos téléphones portables. Certains ne peuvent plus s'en passer. Les applications mobiles sont très récentes et pourtant leur apparition a très vite submergé et totalement changé le quotidien de la population.

La location existe depuis des années, location d'appartements, de voitures (1904), de vêtements (2009), et aussi de smartphones dernièrement. Cependant elle a connu plusieurs évolutions : des annonces sur les journaux, des boutiques, des sites web pour arriver aujourd'hui à des applications mobiles.

De nombreuses applications de location dans le monde entier voient le jour dans des différents domaines : 'Airbnb' pour la location de vacances disponible dans 190 pays, 'Kayak' pour la location de voiture, 'Loola app' la première application destinée à la location de vêtements en particulier…, donc pour chaque domaine il va nous falloir une application ce qui implique la surcharge des déférents terminaux, risque de se perdre entre toutes ces applications ainsi que d'avoir des difficultés à rechercher certains produits inexistants dans ses déférentes applications.

La réalité du terrain dans notre pays nous informe qu'il n'y a pas vraiment de site ou d'applications mobiles fiables auprès des utilisateurs qui offrent ce genre de service. C'est pour cela que nous venons avec notre projet afin de répondre aux besoins des clients dans les différents marchés en matière de location. Cela se fera dans une seule application mobile pour faciliter la tâche et augmenter les gains pour les entreprises de location.

L'entreprise « VegaSoft » dans laquelle nous avons effectué notre stage a décidé de faire en sorte d'avoir tout un réseau de tous types de location, pour faciliter la recherche, permettre une interaction fluide, contrôler les publications, évaluer les particuliers (les utilisateurs de notre application) et accéder un large choix en matière de location.

D'où la nécessité d'une solution informatique, où notre objectif est de développer une application mobile pour y remédier.

Le présent mémoire est divisé en quatre chapitres :

Dans le premier chapitre intitulé « Etat de l'art », nous allons étudier quelques applications déjà existantes.

Le deuxième chapitre, « Spécification et analyse des besoins » sera consacré à l'identification des besoins fonctionnels et non fonctionnels auxquels doit répondre notre application, ainsi qu'à l'identification des acteurs et leurs cas d'utilisation associés.

Le troisième chapitre, « Conception » portera sur le fonctionnement du futur système en répondant aux besoins identifiés dans le chapitre précédent. Il décrira les cas d'utilisations les plus significatifs sous forme de scénario et de diagramme séquence détaillé. Un diagramme de classe sera aussi présenté pour modéliser le côté statique de notre application. Ce diagramme sera transformé par la suite en modèle logique de données.

Dans le dernier chapitre qui porte comme titre « Réalisation », nous détaillerons les étapes à suivre pour déployer, tester et maintenir notre application. Nous allons également présenter l'architecture adoptée ainsi l'implémentation de l'application, la manière que nous allons choisir pour héberger notre futur système, et les pratiques que nous allons mettre en œuvre pour garantir une sécurité optimale.

Enfin, nous terminerons avec une conclusion générale, pour résumer l'essentiel du projet.

Chapitre I

État de l’Art

I.1 Introduction

Dans ce chapitre, nous allons commencer par analyser quelques applications similaires à notre projet, puis d'exposer les problèmes recensés en matière de location des différents produits : la recherche, la navigation, la réservation, … et les solutions proposées pour le client de VegaSoft.

I.2 Critique de L'existant

Nous avons pris deux applications mobiles une ACR [1] pour Algérie Car Rental et seLoger [2] pour analyser les points forts et les points faibles de chacune de ses applications :

Figure I.1 : Logo de Algerie Car Rental [1].

Figure I.2 : Logo de seLoger [2].

I.2.1 ACR

Représente une application mobile destinée à la location de véhicules les plus proches de chez vous en Algérie, elle est disponible sur play store et app store.

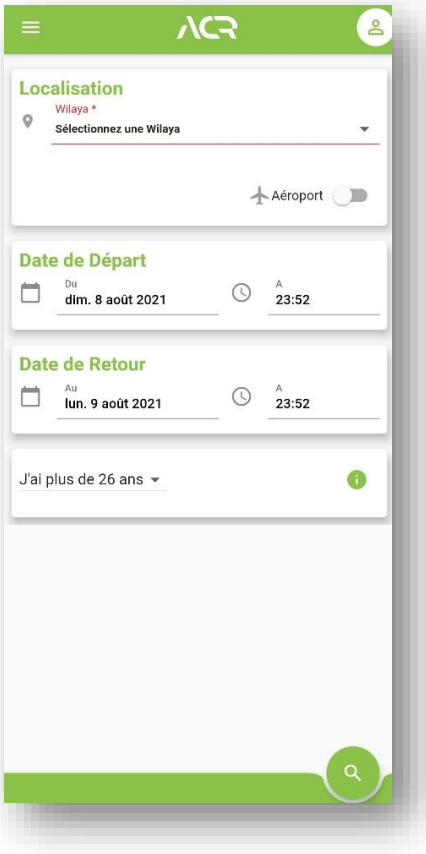

Figure I. 3 : Interface Recherche de l'application « ACR ».

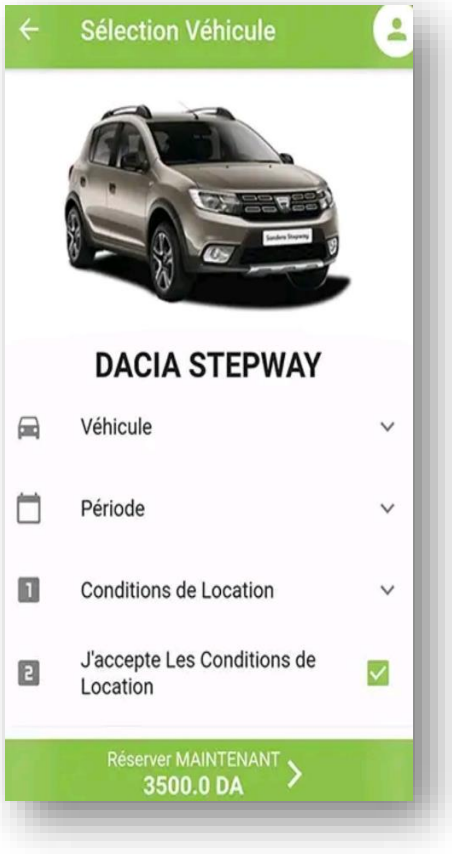

Figure I. 4 : Interface Accueil de l'application « ACR ».

I.2.1.1 Critiques (ACR)

Points faibles :

- Les utilisateurs trouvent que les prix sont très élevés par rapport aux agences de location de véhicules :

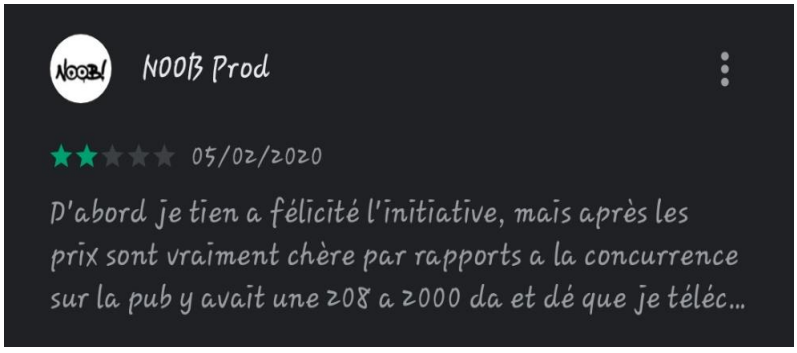

Figure I. 5 : Capture d'écran d'un avis de l'utilisateur 1 [1].

- Elle est disponible seulement dans quatre régions au niveau national.

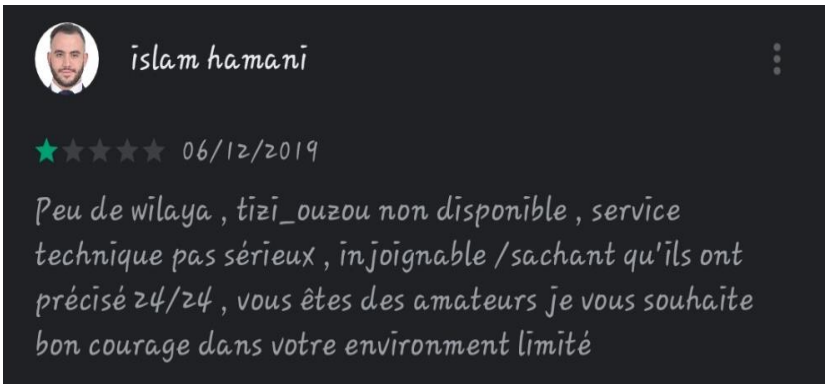

Figure I. 6 : Capture d'écran d'un avis de l'utilisateur 2 [1].

Points forts :

- Les données importantes sur les véhicules sont données de manière simple et précise.
- Le système de filtre pour la recherche est simple.

I.2.2 seLoger

Comme son nom l'indique, c'est une application qui permet de trouver des appartements, des maisons …, pour la location ou bien en vente.

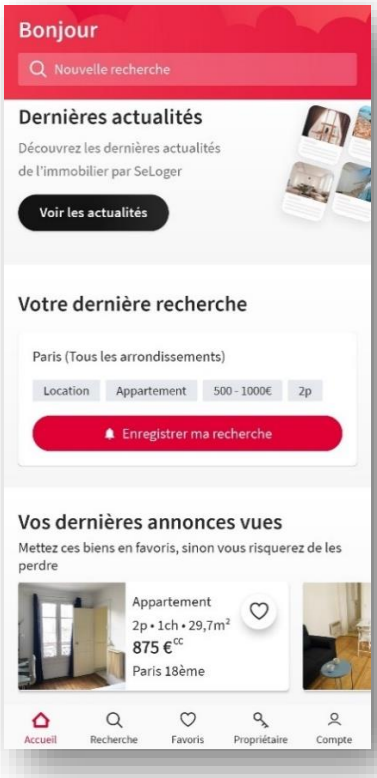

Figure I. 7 : Anglet Accueil « seLoger ».

Figure I. 8 : Anglet Recherche « seLoger ».

I.2.2.1 Critiques (seLoger)

Points faibles :

- Pour pouvoir mettre un bien en location, il faut d'abord passer par leur site seLoger.com, ce qui peut poser quelques problèmes de connexion et cela ne facilite pas la tâche :

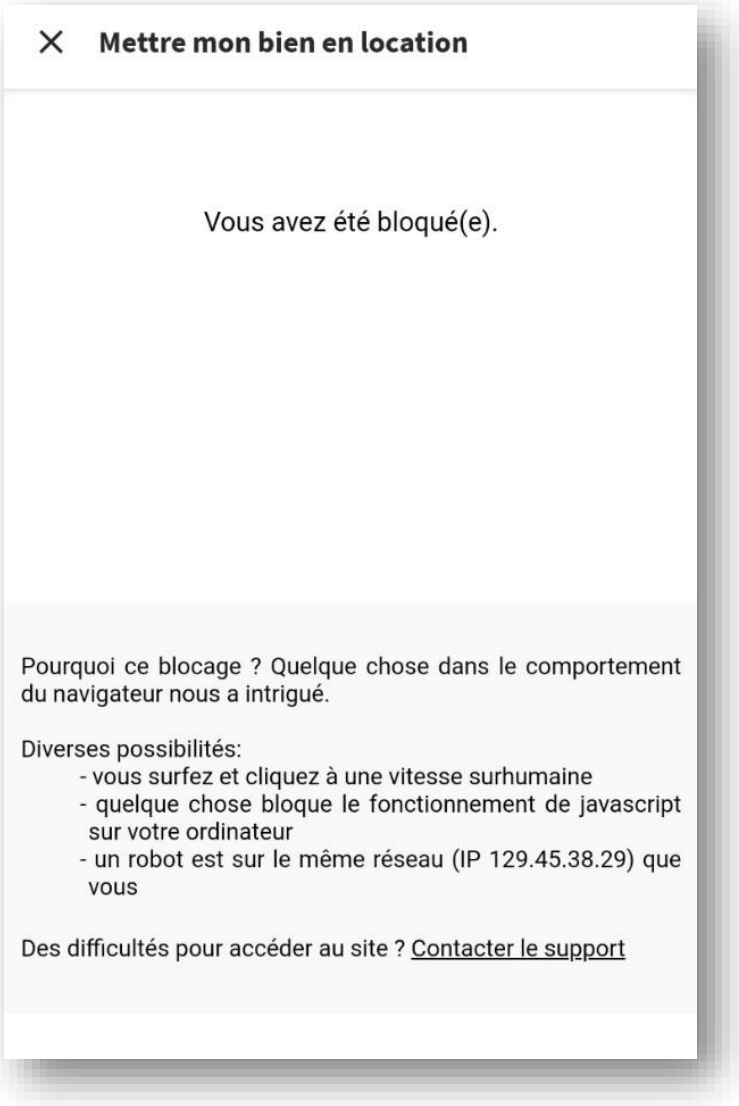

Figure I. 9 : anglet Mettre mon bien en location : problème rencontré.

- Recherche un peu complexe trop de filtres obligatoires (3 champs) :

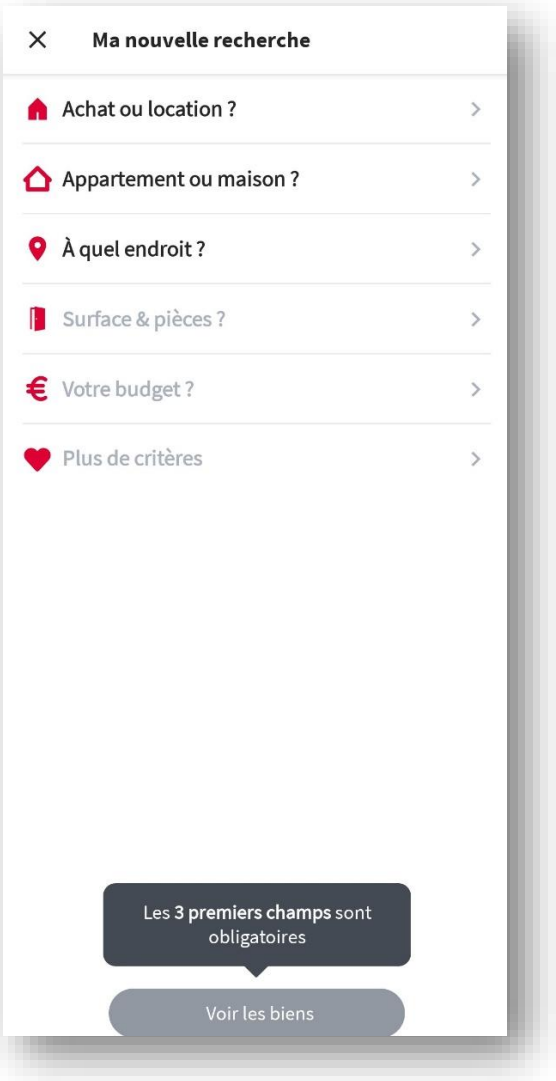

Figure I. 10 : anglet Recherche de l'application « seLoger ».

- Généralement les annonces ne sont pas crédibles et les pseudos agences ne répondent pas aux appels ni aux courriels, comme le témoigne ces deux commentaires sur play store :

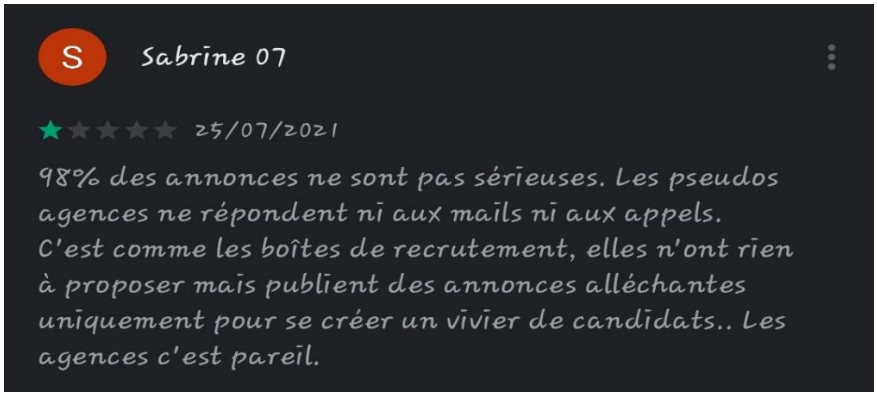

Figure I. 11 : Capture d'écran d'un avis de l'utilisateur 3 [2].

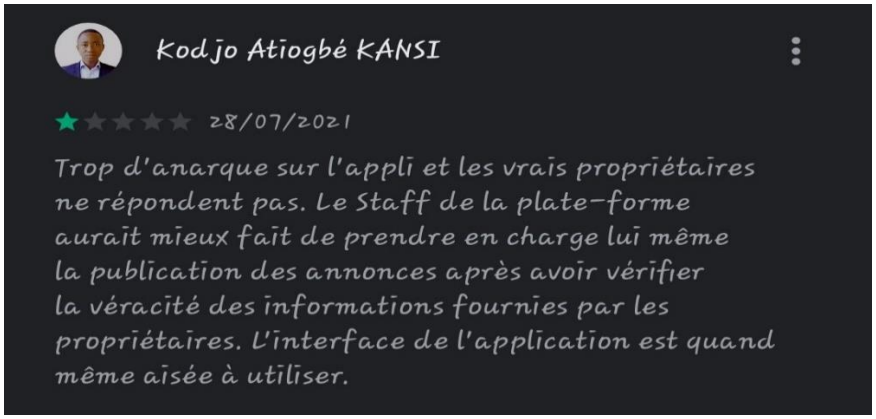

Figure I. 12 : Capture d'écran d'un avis de l'utilisateur 4 [2].

- Problème de la location sur la carte qui ne corresponde pas à la location réelle du bien mis en vente ou bien en location. Comme le témoigne Melodie Ripault :

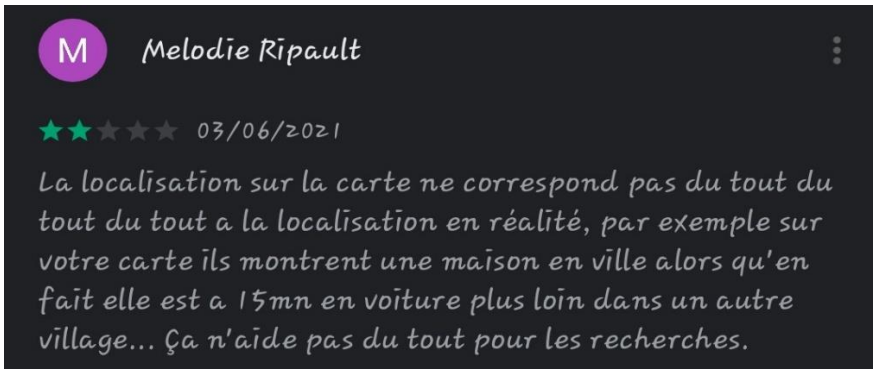

Figure I. 13 : Capture d'écran d'un avis de l'utilisateur 5 [2].

Points forts :

- Application rapide, intuitive et facile à utiliser avec de belles interfaces visuelles.
- Système de recherche précis et pertinent.

I.3 Problématique

En Algérie nous rencontrons un problème considérable en matière de location de différents produits en ligne, vu l'absence quasi totale des applications mobiles ou web qui fournissent ce genre de service. Nous trouvons par exemple des sites et des applications mobiles où nous pouvons louer quelques produits comme : Algerie Car Rental : spécialisé dans la location de véhicule, airbnb : pour la location de vacances, seLoger : pour louer ou vendre des bien tel que des appartements, des maisons (elle n'est pas disponible en Algérie). Mais on ne trouve pas certains produits en location tels que les vêtements, les engins de construction, des terrains agricoles… C'est pour cela que nous envisageons développer notre application mobile qui sera dédiée uniquement et explicitement à la location de tous produits existants, de faire en sorte de rassembler tous les domaines qui peuvent exister. Elle sera lancée et mise en œuvre pour l'été 2021.

I.4 Les Besoins

De nos jours se penchent de plus en plus vers la location dans des différents domaines, les personnes ne peuvent pas se permettre d'acheter certains produits ou bien ils ont besoin d'utiliser ce dernier qu'une seule fois ou occasionnellement, donc ils sont obligés d'opter pour cette option existante (la location) pour minimiser les coûts. Plusieurs personnes aussi possédant des objets de valeur ou d'utilité pour la société, veulent augmenter leurs profits et

revenus en les louant régulièrement. C'est pour cela que nous comptons développer cette application qui permettra de faciliter ces tâches.

I.5 Les Objectifs du Projet

Le client a exprimé ses besoins concernant l'application et son fonctionnement lors d'une réunion. Ce qui nous a permis d'arriver aux solutions suivantes :

1- Faciliter l'utilisation en générale.

2- Gestion des comptes des annonceurs et leurs publications (les annonces).

3- Rechercher et réserver facilement les produits mis en location, organiser les annonces par catégorie et sous-catégorie.

4- Donner un aspect professionnel pour l'application.

5- Fournir des informations riches et détaillées pour les visiteurs concernant les produits prêts à louer.

C'est pourquoi notre application doit être facile à utiliser et à manipuler par nos clients avec des interfaces intuitives et simples.

I.6 Conclusion

L'étude de l'existant nous a permis d'analyser des systèmes similaires à notre projet et d'ajouter des fonctionnalités que nous avons jugé nécessaires pour donner une meilleure expérience à l'utilisateur.

Dans le prochain chapitre, nous aborderons l'analyse et la spécification des besoins de notre projet. Nous allons précisément élaborer un cahier de charge qui va résumer les besoins du client et qui va définir les différents éléments comme : les cas d'utilisations qui vont servir de base à la conception de notre application mobile.

Chapitre II

Spécification et Analyse des Besoins

II.1 Introduction

Dans le cadre d'un projet informatique, le recours à la modélisation **UML** [3] (voir annexe 2) procure de nombreux avantages et nous permet de bien définir les besoins clients, c'est-à-dire, bien cerner notre projet. C'est pour cela que nous allons l'appliquer à notre projet, tout en nous appuyant sur la méthode agile **XP** [4] (voir annexe 1) pour « eXtreme programming ».

Après avoir bien défini la problématique, nous allons passer maintenant à l'analyse et spécification des besoins de notre projet en utilisant les outils précédemment cités. Tout d'abord, nous allons identifier les acteurs et les cas d'utilisations, puis nous passerons à la description de ces derniers.

II.2 Spécification des Besoins

II.2.1 Les Besoins Fonctionnels

Les besoins fonctionnels sont les fonctionnalités que notre client attend de notre système, sont ceux qui précisent ce que le système doit faire, en d'autres termes, ils spécifient une fonction, un comportement ou une action que le système doit exécuter. Nous les avons définis à l'aide des User story suivantes :

Inscription : un visiteur peut s'inscrire en remplissant un formulaire d'inscription avec ses informations personnelles : nom, prénom, email, mot de passe, adresse, numéro de téléphone, pour devenir un client et avoir le droit de publier des annonces, et de les partager avec l'ensemble des utilisateurs.

User stories :

Acteur	**Visiteur**
Action	Inscription
Objectif	Pour devenir un client.

Tableau II. 1 : User stories « Inscription ».

Abonnement : un visiteur ou bien un client peut s'abonner à une publication (un produit).

User stories :

Acteur	**Visiteur, Client**
Action	Abonnement
Objectif	Pouvoir suivre l'annonce pour les prochaines réservations.

Tableau II. 2 : User stories « Abonnement ».

Gestion des annonces : un client peut ajouter, modifier ou supprimer une annonce.

User stories :

Acteur	**Client**
Action	Ajouter une annonce.
Objectif	Publier dans le fil d'actualité.
Condition	Le client doit être validé par un administrateur.

Tableau II. 3 : User stories « Ajouter une annonce ».

Note : pour pouvoir consulter une annonce dans le fil d'actualité, elle doit d'abord être valider par un administrateur.

Consultation : un visiteur ou un client a le droit de consulter les publications dans le fil d'actualité.

User stories :

Acteur	**Visiteur, Client**
Action	Consulter
Objectif	Consulter les informations détaillées du produit mis en location.

Tableau II. 4 : User stories « Consultation ».

Réservation : un visiteur ou un client peut réserver (ou bien louer) un produit.

User stories :

Acteur	**Visiteur, Client**
Action	Réserver

Objectif	Réserver un produit pour une date définie.

Tableau II. 5 : User stories « Réservation ».

Contrôler les clients : un administrateur peut valider ou décliner une inscription d'un visiteur.

User stories :

Acteur	**Administrateur**
Action	Contrôle client
Objectif	Valider ou décliner une inscription.

Tableau II. 6 : User stories « Contrôler les clients ».

Contrôler les annonces (publications) : un administrateur a le pouvoir de valider ou de décliner la publication d'une annonce.

User stories :

Acteur	**Administrateur**
Action	Contrôle annonce
Objectif	Valider ou décliner la publication d'une annonce.

Tableau II. 7 : User stories « Contrôler les annonces ».

II.2.2 Les Besoins non Fonctionnels

Les besoins non fonctionnels décrivent **comment** le système doit fonctionner. Le comportement et la performance que le produit doit avoir. Dans notre cas ces besoins représentent : les contraintes de Sécurité, Esthétiques, Utilisabilité et les contraintes Techniques.

Sécurité :

- Fonctionnalité de chiffrement des données sensibles.
- Haché les mots de passe des utilisateurs.

- Gestion des accès à la donnée selon les droits attribués aux utilisateurs.

Esthétique :

- Avoir la même couleur de fond pour toutes les interfaces.
- Une page d'accueil spécifique à l'utilisateur.
- Les pages doivent être dans le même principe, c'est-à-dire garder le même aspect visuel dans toutes les pages.

Utilisabilité :

- Avoir des interfaces fluides et faciles à utiliser.
- Avoir une interaction homme/machine, entre l'utilisateur et l'interface grâce à des fenêtres et des messages explicatifs.
- Des interfaces intuitives.

Technique :

- L'utilisation du NodeJs [5].
- L'utilisation de java script pour manipuler les formulaires.
- L'utilisation de la base de données MySQL [6].
- La méthode agile **XP** [4] (Voir ANNEXE 1) basée sur le langage de modélisation **UML** [3] (Voir ANNEXE 2).
- L'utilisation de HTML5 [7], CCS3 [8] et Bootstrap [9] pour bien structurer notre application.
- L'utilisation d'Ajax [10] pour permettre de modifier partiellement la page affichée en évitant l'actualisation de la page, ce qui permet à notre application de gagner en fluidité et en ergonomie.
- L'utilisation du Framework Cordova [11] pour permettre de créer des applications "hybrides", principalement pour Android et iOS en HTML, CSS et JavaScript [12].

II.3 Analyse des Besoins

II.3.1 Identification des Acteurs

Un acteur c'est une entité externe (humain ou autre) qui interagit avec le système.

Après la spécification des besoins, nous avons identifié 3 acteurs :

Visiteur : représente une personne qui peut consulter et réserver des produits sans avoir besoin de s'authentifier, il pourra aussi s'inscrire et s'enregistrer.

Client : c'est la personne qui possède un compte au sein de notre système ayant le droit de publier des annonces (gestion des annonces), et ayant tous les droits que le visiteur.

Administrateur : c'est la personne qui a accès à tous les services de l'application pour lui permettre de contrôler les publications et les annonceurs.

II.3.2 Identification des Messages

Les acteurs peuvent communiquer avec le système : des messages entrants représentent la demande que l'acteur veut effectuer et des messages sortants représentent la réponse du système à la demande donnée par l'acteur.

II.3.2.1 Les Messages Entrants Echangés avec le Système

Acteur	Messages entrants
Visiteur	• Consultation de la liste des produits. • Demande d'inscription. • Demande de réservation d'un produit • Demande d'informations relatives à un produit
Client	• Consultation de la liste des produits. • Demande d'ajouter une publication (produit). • Demande de réservation d'un produit. • Demande d'informations relatives à un produit. • Demande de modifier les informations client.

Administrateur	• Consultation de la liste des produits. • Demande d'ajouter une publication (produit). • Demande d'informations sur un produit. • Demande de consulter les informations relatives à un client. • Demande de modifier les informations Administrateur.

Tableau II. 8 : Les messages entrant échangés avec le système.

II.3.2.2 Les Messages Sortant Echanger avec le Système

Acteur	Messages sortants
Visiteur	• La liste des produits. • Le formulaire approprié à l'inscription. • Le formulaire approprié à une réservation. • Affichage des informations relatives à un produit.
Client	• La liste des produits. • Le formulaire approprié à un ajout d'un produit. • Le formulaire approprié à une réservation. • Affichage des informations relatives à un produit • Espace compte.

Administrateur	• La liste des produits • Le formulaire approprié à un ajout d'un produit. • Affichage des informations relatives à un produit. • Les informations du client. • Espace compte.

Tableau II. 9 : Les messages sortant échangés avec le système.

II.3.3 Diagramme du Contexte Dynamique

Il s'agit d'un diagramme qui permet de délimiter le domaine d'étude, et mettre en évidence les messages échangés entre les acteurs et le système.

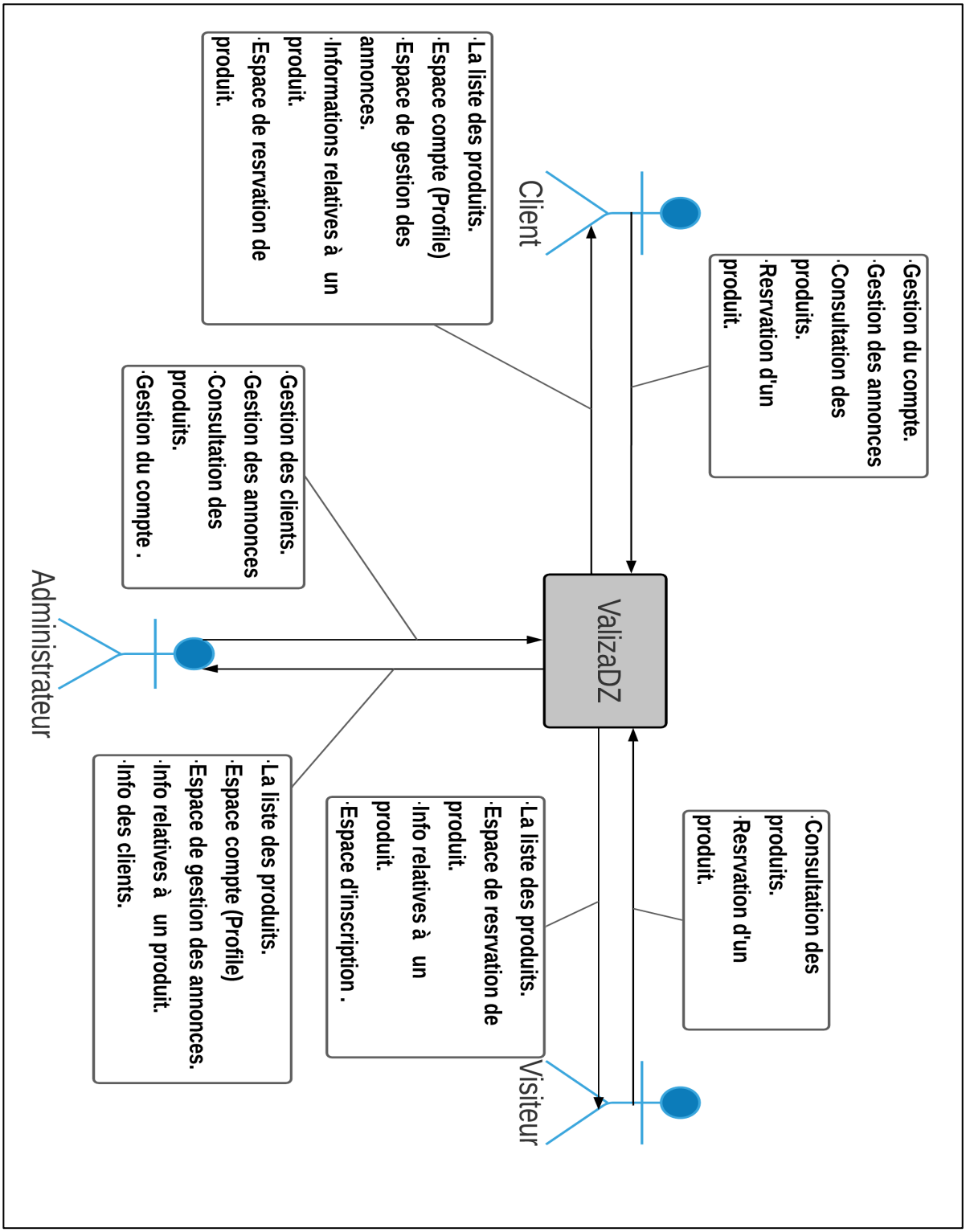

Figure II. 1 : Diagramme du contexte dynamique.

II.3.4 Identification des Cas d'Utilisation

Un cas d'utilisation est une fonctionnalité du système permettant de rendre un service à un acteur. Nous avons identifié les cas d'utilisation ci-dessus :

N°	Cas d'utilisation		Acteur
1	S'authentifier		Client Administrateur
2	S'inscrire		Visiteur
3	Consulter les annonces	Consulter les informations	Visiteur, Client, Administrateur
		Réserver	
4	Gérer les annonces	Ajouter	Client Administrateur
		Modifier	
		Supprimer	
5	Gérer le compte	Modifier les informations	Client Administrateur
6	S'abonner		Visiteur, Client
7	Contrôler les annonces	Valider	Administrateur
		Décliner	
8	Contrôler les annonceurs	Valider	Administrateur
		Décliner	

Tableau II. 10 : Identification des cas d'utilisation.

II.3.5 Diagramme de Cas d'Utilisation

C'est une représentation graphique permettant de capturer les fonctions d'un système. La figure suivante représente le diagramme de cas d'utilisation associé à notre système.

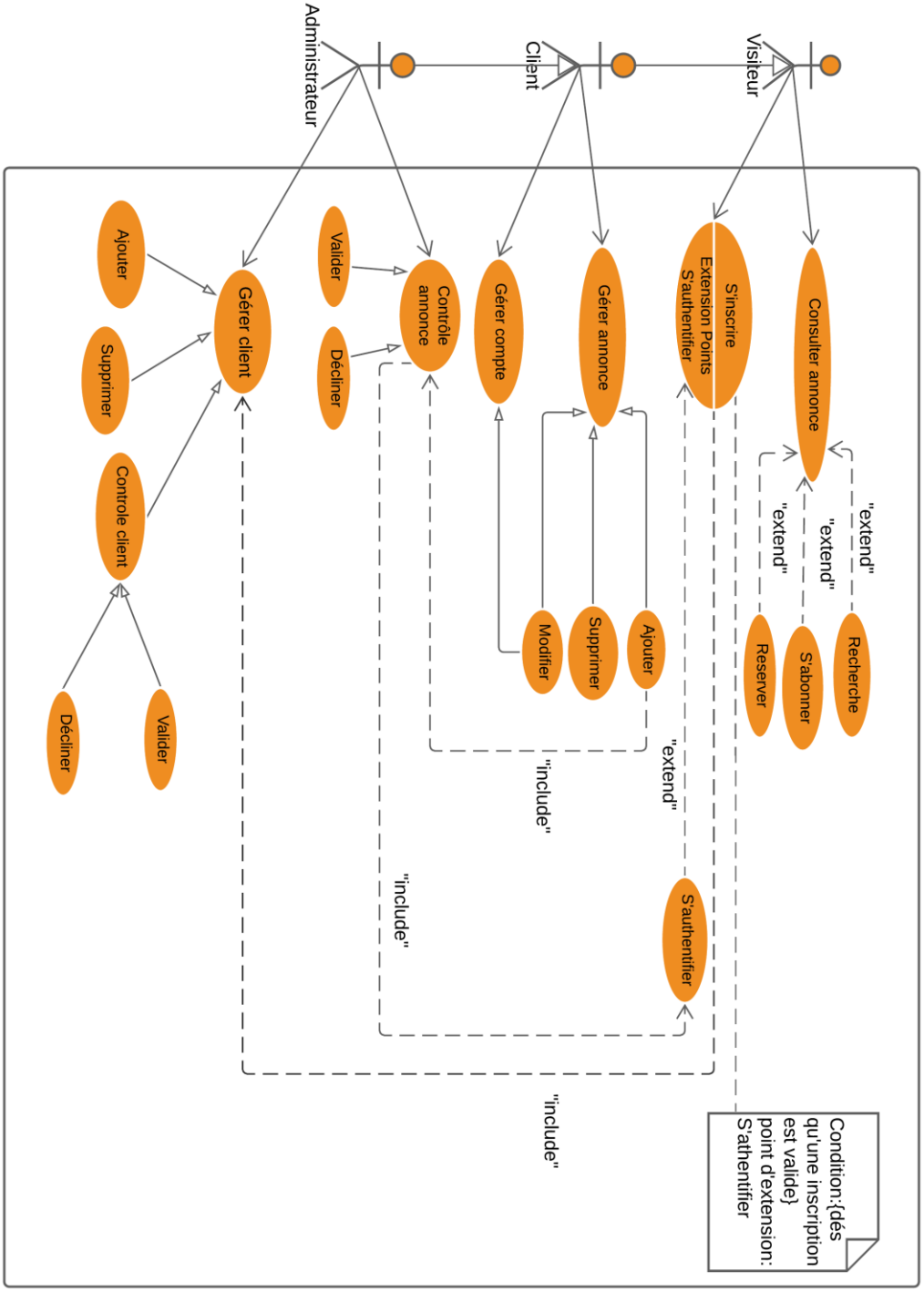

Figure II. 2 : Diagramme de cas d'utilisation.

II.4 Conclusion

Ce chapitre nous a permis de comprendre les exigences du client, les fonctionnalités que notre système doit accomplir et également d'avoir une vision globale sur notre système que nous allons détailler par la suite.

CHAPITRE III

Conception

III.1 Introduction

Dans cette phase, nous allons décrire chaque sous-ensemble de notre application mobile, grâce à la modélisation dynamique : diagramme d'état-transition et les diagrammes de séquences, puis nous détaillerons notre conception avec le diagramme de classe et de modèle logique de données.

III.2 Modélisation Dynamique

III.2.1 Diagramme d'Etat-Transition

Pour bien comprendre le fonctionnement de notre système, par rapport à la gestion des annonces par nos clients, nous avons défini des états de transition qui vont nous permettre par la suite d'avoir un contrôle des droits d'accès à la donnée.

La figure suivante représente une description des états des annonces lors d'un ajout de cette dernière.

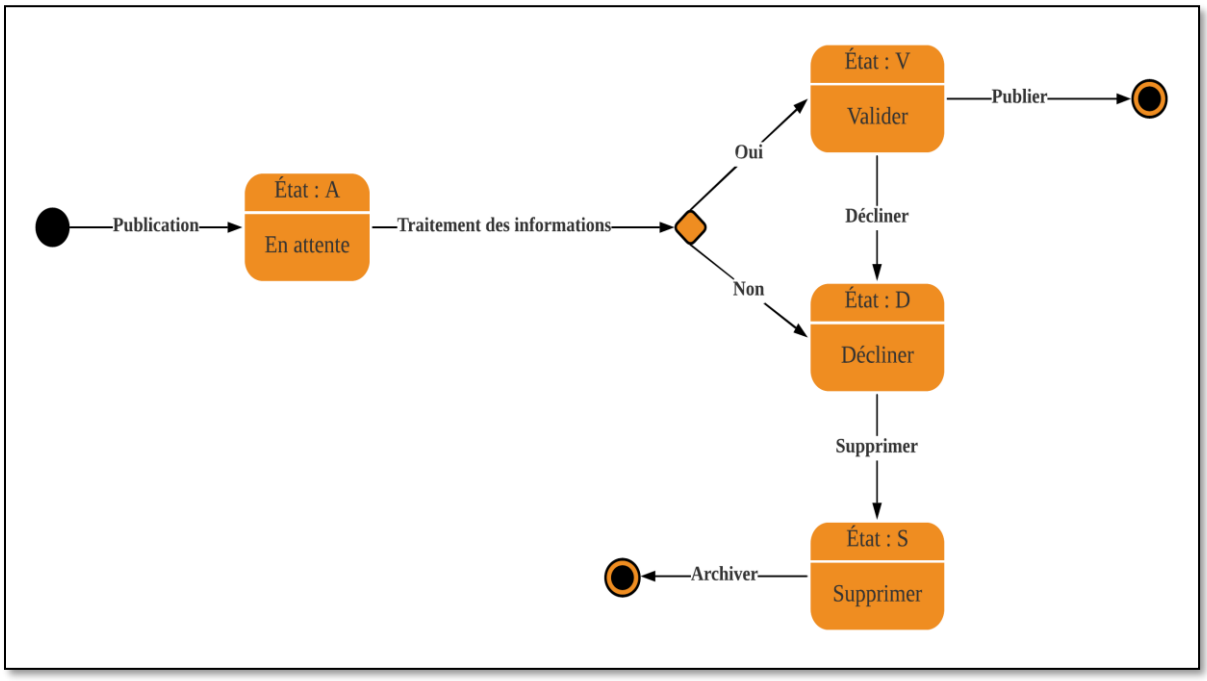

Figure III. 1 : Diagramme d'état de transition d'une annonce.

III.2.2 Les Diagrammes de Séquences

Les diagrammes de séquences permettent de décrire comment les éléments du système interagissent entre eux et avec les acteurs [22].[24].

III.2.2.1 Cas d'Utilisation « S'authentifier »

CAS D'UTILISATION N° 1	**« S'AUTHENTIFIER »**
RESUME	Vérification de l'identité des utilisateurs. Lorsque les informations sont valides (Email et mot de passe), le système permet aux utilisateurs d'accéder à des services protégés selon leur type (administrateur, client).
ACTEURS	Administrateur, Client.
SCENARIO NOMINAL	[Début] • L'utilisateur envoie une demande d'ouverture de session en spécifiant son login et mot de passe. • L'utilisateur envoie une demande de connexion. • L'interface connexion vérifie notre saisie si elle respecte les normes. • Les actions seront répétées jusqu'à ce que la saisie soit correcte. • La demande va passer de l'interface connexion vers le contrôleur du cas d'utilisation « se connecter ». • Grâce à la classe utilisateur, on peut déterminer si les données (Email et mot de passe) existent. [Si (donnée(s) n'existe pas)] • Afficher un message d'erreur. [Sinon]

	• Le contrôleur vérifie le type de l'utilisateur auquel correspond les informations saisies. [Si (l'utilisateur est de type client)] • Création d'une session et redirection vers l'espace client. [Si (l'utilisateur est de type admin)] • Création d'une session et redirection vers l'espace administrateur.

Tableau III. 1 : Description de cas d'utilisation « S'authentifier ».

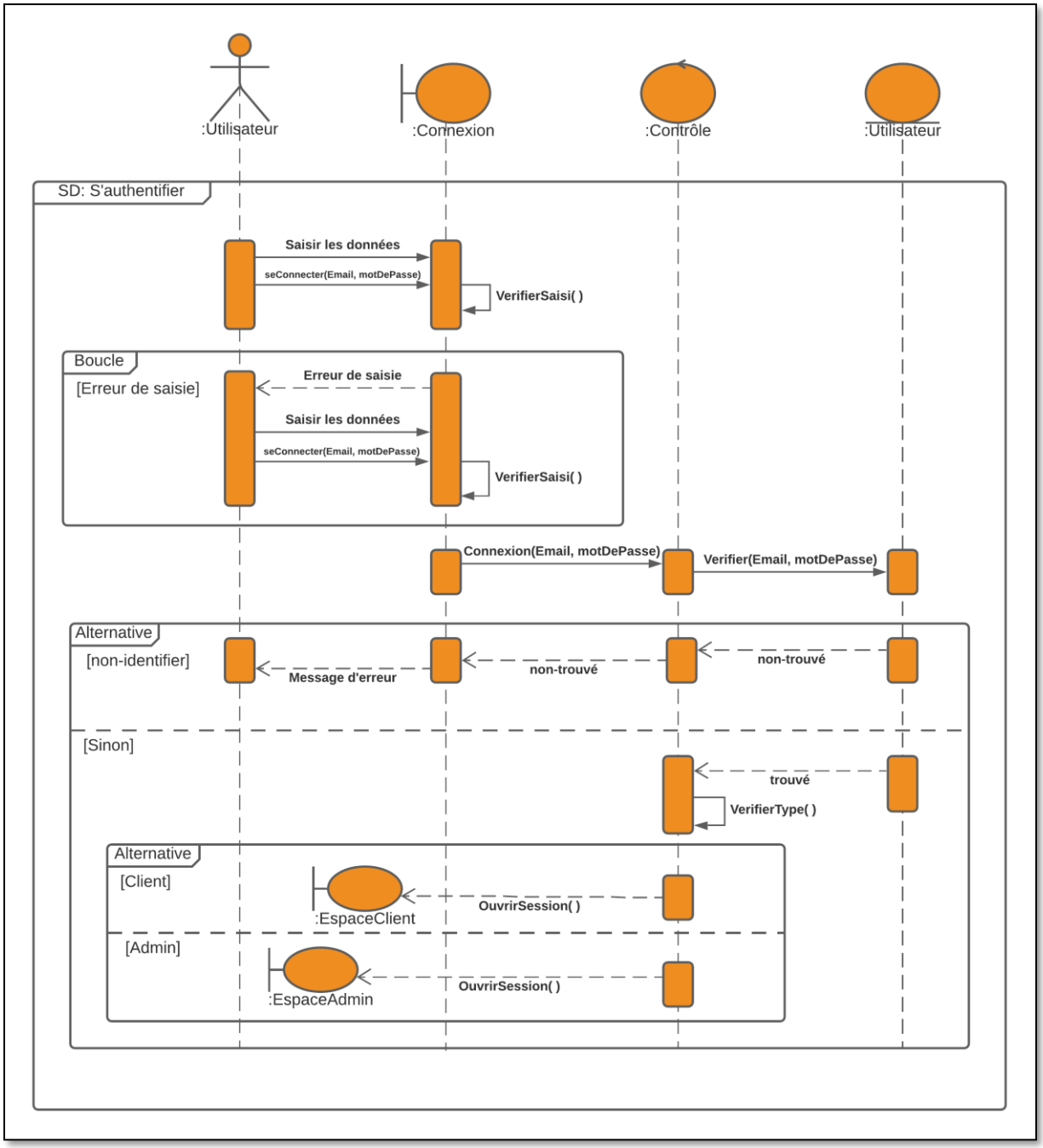

Figure III. 2 : Diagramme de séquence pour le cas d'utilisation « S'authentifier ».

III.2.2.2 Cas d'Utilisation « Ajouter une annonce »

CAS D'UTILISATION N° 2	« AJOUTER UNE ANNONCE »
RESUME	C'est le cas le plus important de notre application, il nous permet d'ajouter une nouvelle annonce, l'utilisateur remplit les informations de l'annonce dans un formulaire, puis elles doivent être contrôlées par un administrateur pour la publication.
ACTEURS	Administrateur, Client.
SCENARIO NOMINAL	[Début] • Le client envoie une demande d'ajout d'une annonce. [Si (le client n'est pas authentifié)] • Affichage de l'interface d'authentification. [Sinon] • Affichage du formulaire d'ajout d'une annonce. • Le client saisit les données. • Le client publie le formulaire pour l'interface classe. • Vérification de la saisie par la classe annonce. [Tant Que (erreur de saisie)] 1. Répéter le processus de remplissage du formulaire. • L'interface annonce demande de publier l'annonce.

	• Le contrôleur remet une demande de vérification (Contrôler l'annonce) à la classe admin. • La classe admin contrôle les informations de l'annonce. [Si (le client n'est pas authentifié)] • Affichage de l'annonce dans le fil d'actualité. [Si (le client n'est pas authentifié)] • Notification et le motif du refus.

Tableau III. 2 : Description de cas d'utilisation « Ajouter une Annonce ».

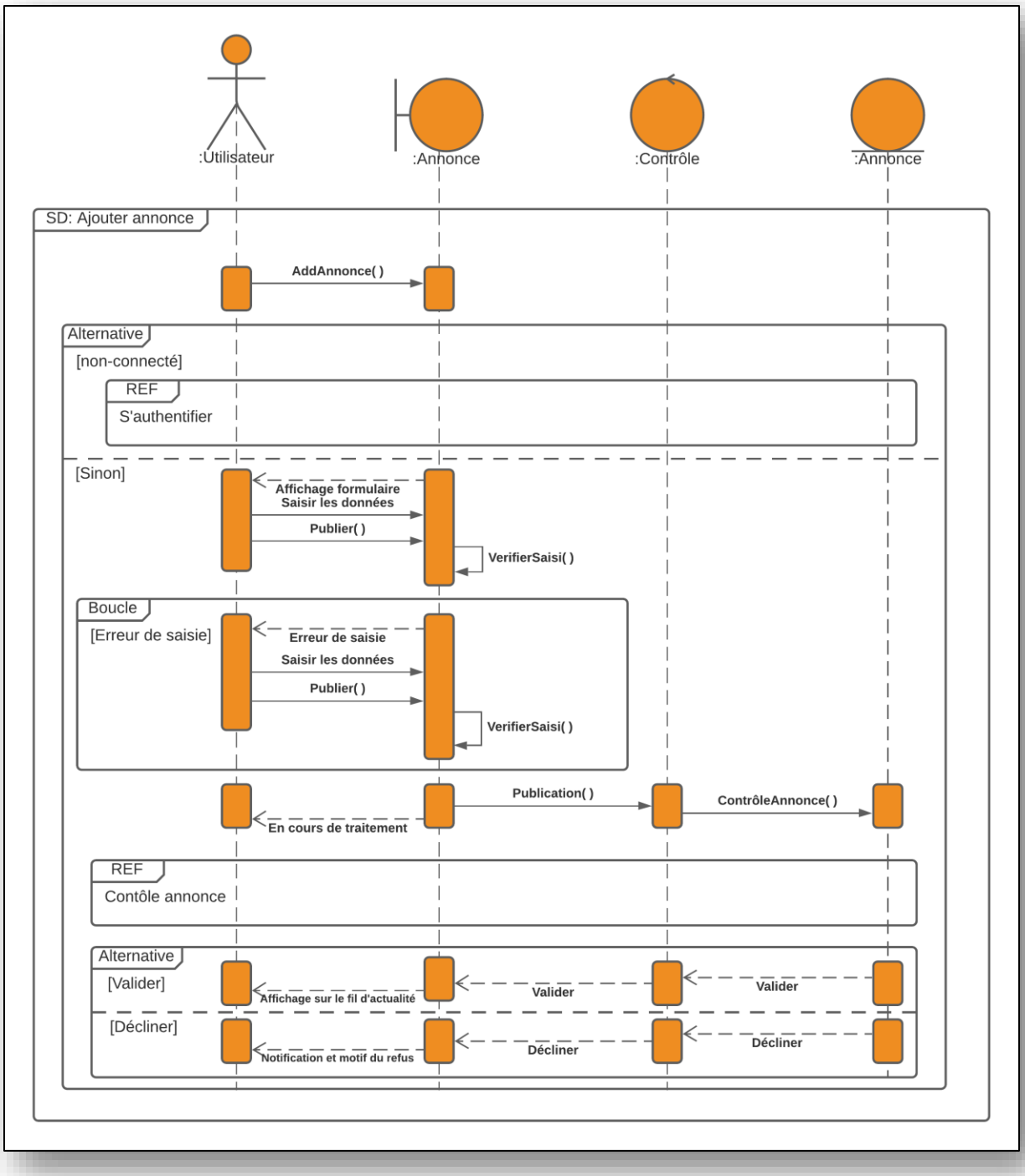

Figure III. 3 : Diagramme de séquence pour le cas d'utilisation « Ajouter une annonce ».

III.2.2.3 Cas d'Utilisation « Consulter une annonce »

CAS D'UTILISATION N° 3	« CONSULTER UNE ANNONCE »
RESUME	L'utilisateur consulte les informations de l'annonce.
ACTEURS	Visiteur, Client, Administrateur.
SCENARIO NOMINAL	[Début] • L'utilisateur formule une demande pour récupérer les annonces. • La liste des annonces est affichée pour l'utilisateur. • L'utilisateur choisit une annonce à consulter. • On récupère les informations concernant l'annonce à partir de la classe annonce. • Les informations concernant l'annonce sont affichées pour l'utilisateur.

Tableau III. 3 : Description de cas d'utilisation « Consulter une annonce ».

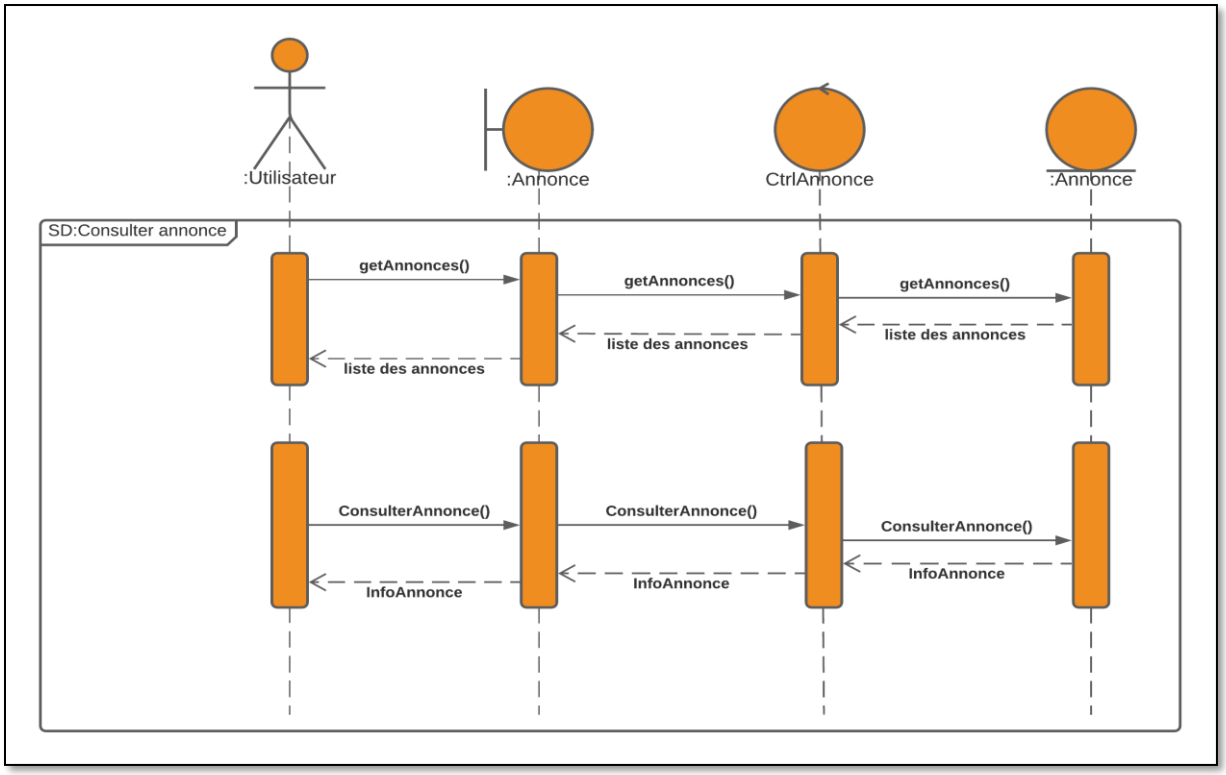

Figure III. 4 : Diagramme de séquence pour le cas d'utilisation « Consulter une annonce ».

III.2.2.4 Cas d'Utilisation « Contrôle annonce »

CAS D'UTILISATION N° 4	**« CONTROLE ANNONCE »**
RESUME	L'administrateur reçoit une notification du système pour vérifier la crédibilité des informations des annonces, puis décide de valider ou décliner la publication de ses dernières. Si l'annonce est validée elle sera publiée automatiquement dans le fil d'actualité, sinon, le client reçoit une notification et le motif du refus.
ACTEURS	Administrateur.
SENARIO NOMINAL	[Début] • L'utilisateur s'authentifie en tant qu'administrateur. • Envoi d'une requête pour récupérer la liste d'annonce.

	• Transfert de la requête de l'interface ListeAnnonce au Contrôle puis à la classe Annonce. • La classe annonce renvoie la liste des annonces à l'interface ListeAnnonce. • L'utilisateur sélectionne l'annonce qu'il souhaite afficher. • La classe annonce retourne les infos concernant l'annonce sélectionnée à l'interface annonce. [Si (valider)] • Affichage d'un message de confirmation. [Sinon] • Affichage d'un message de confirmation de la déclination de l'annonce.

Tableau III. 4 : Description de cas d'utilisation « Contrôle Annonce ».

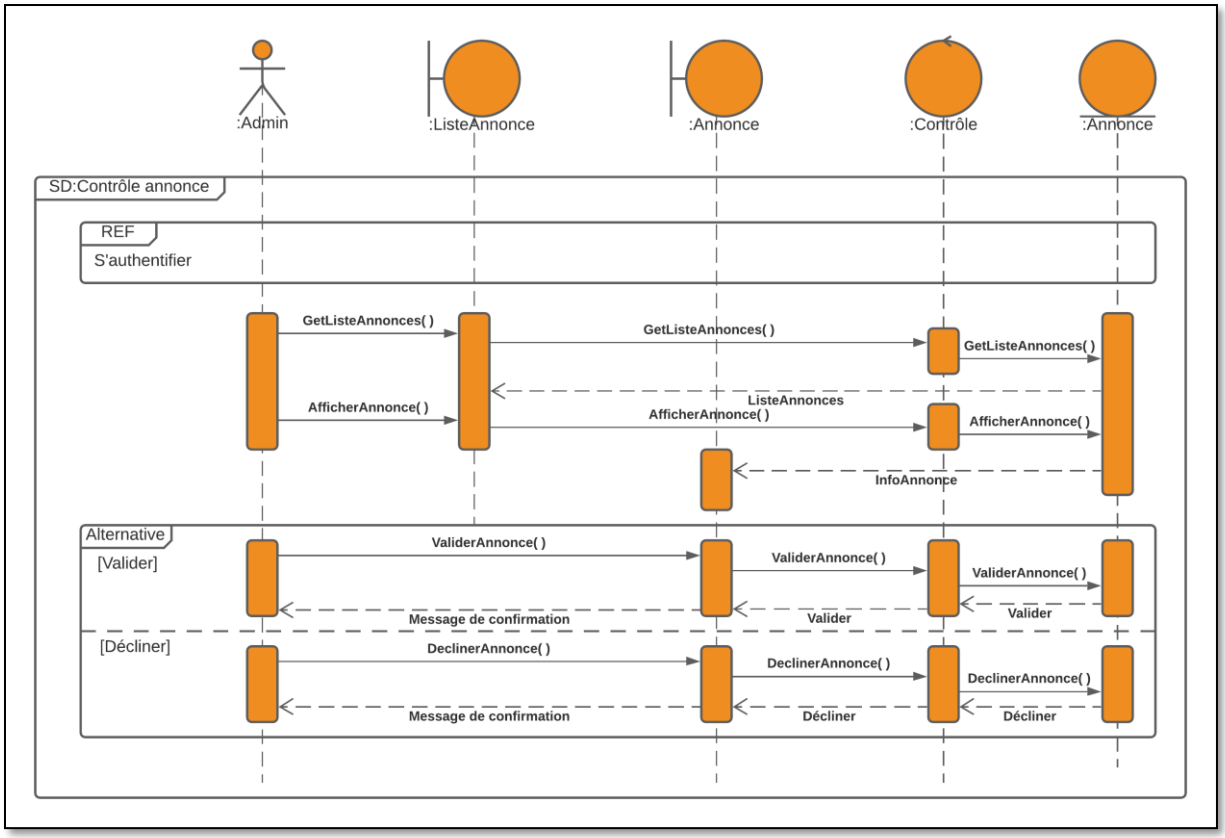

Figure III. 5 : Diagramme de séquence pour le cas d'utilisation « Contrôle annonce ».

III.2.2.5 Cas d'Utilisation « Contrôle client »

CAS D'UTILISATION N° 5	« CONTROLE CLIENT »
RESUME	L'administrateur reçoit une notification du système pour vérifier la crédibilité des informations des clients, puis décide de valider ou décliner leur inscription. Si c'est validé le compte sera créé, sinon le visiteur reçoit une notification et le motif du refus.
ACTEURS	Administrateur.
SCENARIO NOMINAL	[Début] • L'utilisateur s'authentifié en tant qu'administrateur. • Envoie d'une requête pour récupérer la liste des clients. • Transfert de la requête de l'interface ListeClient au Contrôle puis à la classe Client. • La classe Client renvoie la liste des Clients à l'interface ListeClient. • L'utilisateur sélectionne le client qu'il souhaite afficher. • La classe Client retourne les infos concernant le client sélectionné à l'interface Client. [Si (valider)] • Affichage d'un message de confirmation. [Sinon] • Affichage d'un message de confirmation du refus de l'inscription.

	• Le client reçoit une notification plus le motif du refus.

Tableau III. 5 : Description de cas d'utilisation « Contrôle Client ».

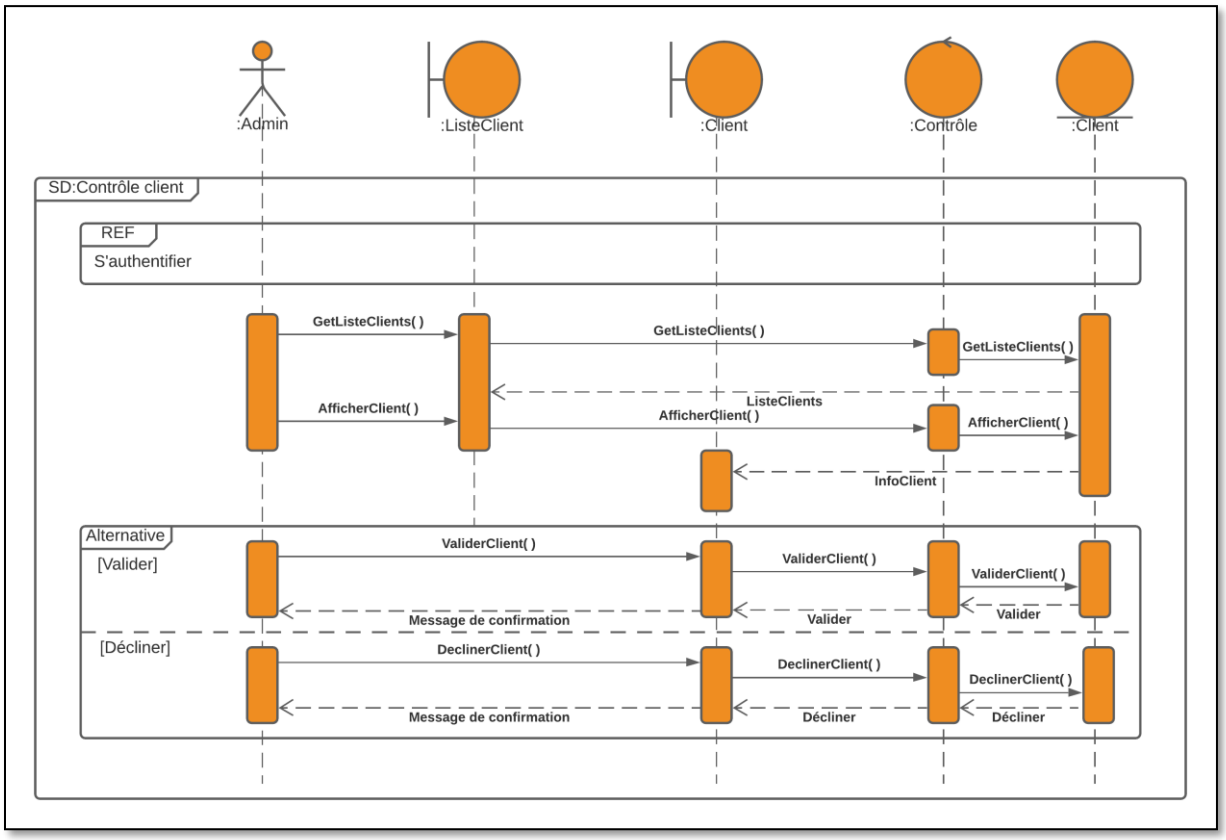

Figure III. 6 : Diagramme de séquence pour le cas d'utilisation « Contrôle client ».

III.2.2.6 Cas d'Utilisation « S'inscrire »

CAS D'UTILISATION N° 6	« S'INSCRIRE »
RESUME	Créer un nouveau compte pour devenir un client. L'utilisateur remplit le formulaire dédieé à l'inscription. L'administrateur contrôle les informations et valide ou bien décline la création de ce nouveau compte.
ACTEURS	Visiteur.
SCENARIO NOMINALE	[Début] • Le visiteur formule une demande d'inscription. • Un formulaire d'inscription sera affiché. • Le visiteur remplit le formulaire et saisit les informations. • Vérification des informations saisies par le visiteur. [Tant Que (erreur de saisie)] • Refaire le processus d'inscription. • Envoi d'une demande d'inscription au CtrlInscription. • Contrôle du client dans la base de la classe admin. • Affichage du message de confirmation en cours de traitement.

Tableau III. 6 : Description de cas d'utilisation « S'inscrire ».

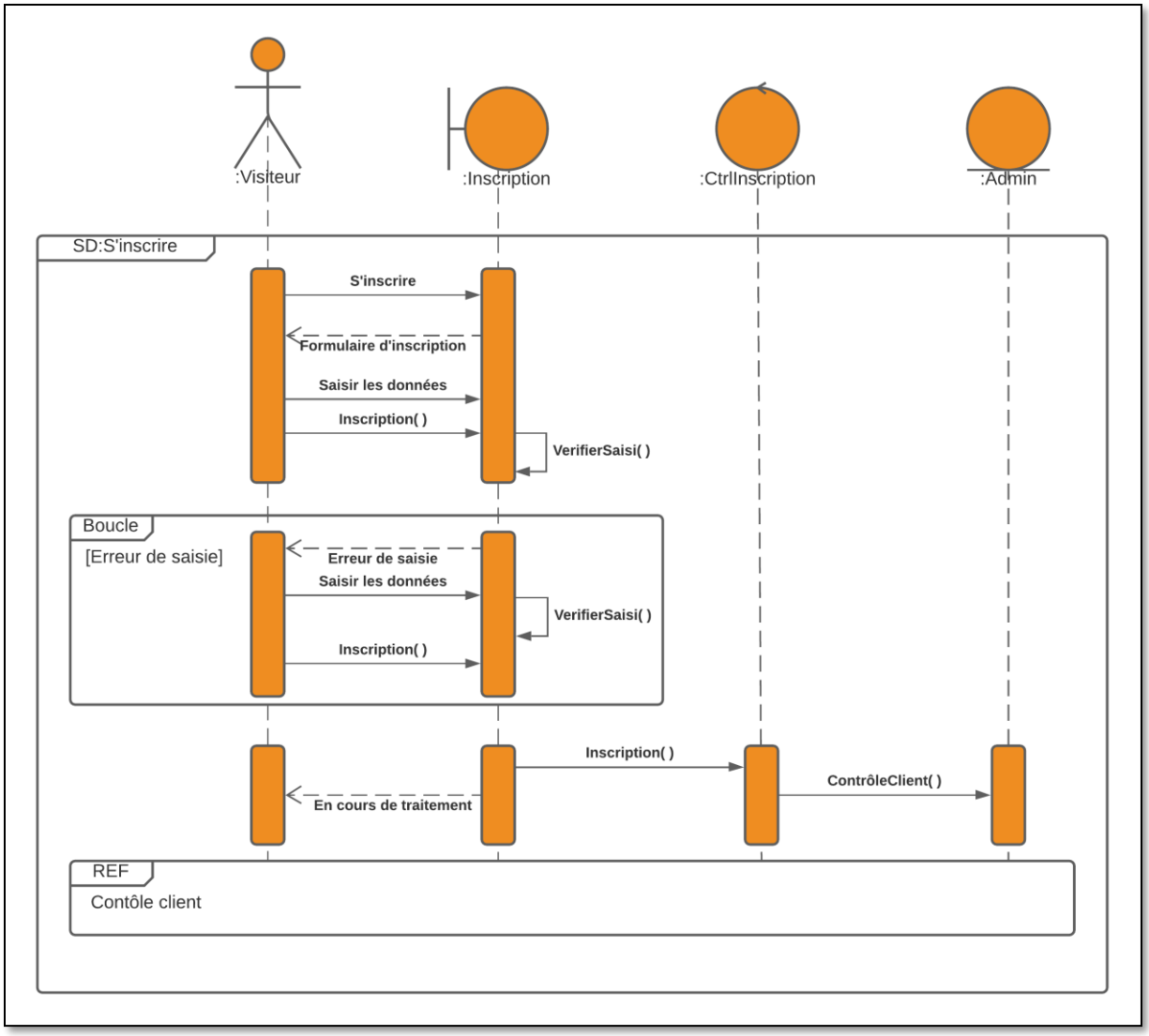

Figure III. 7 : Diagramme de séquence pour le cas d'utilisation « S'inscrire ».

III.3 Modélisation Statique

Pour modéliser la partie statique du système, la première étape est la réalisation du diagramme de classe représenté par la « figure III.7 », suivie par le dictionnaire de données et le modèle logique de données déduit à partir du diagramme de classes conçues.

III.3.3 Diagramme de Classes

Le diagramme de classe exprime d'une manière générale la structure statique d'un système, en termes de classes et de relations entre elles.

Le diagramme de classe de la « **figure III.7** » fournit une description bien détaillée des différentes classes de l'application ainsi que les différentes associations qui les relient.

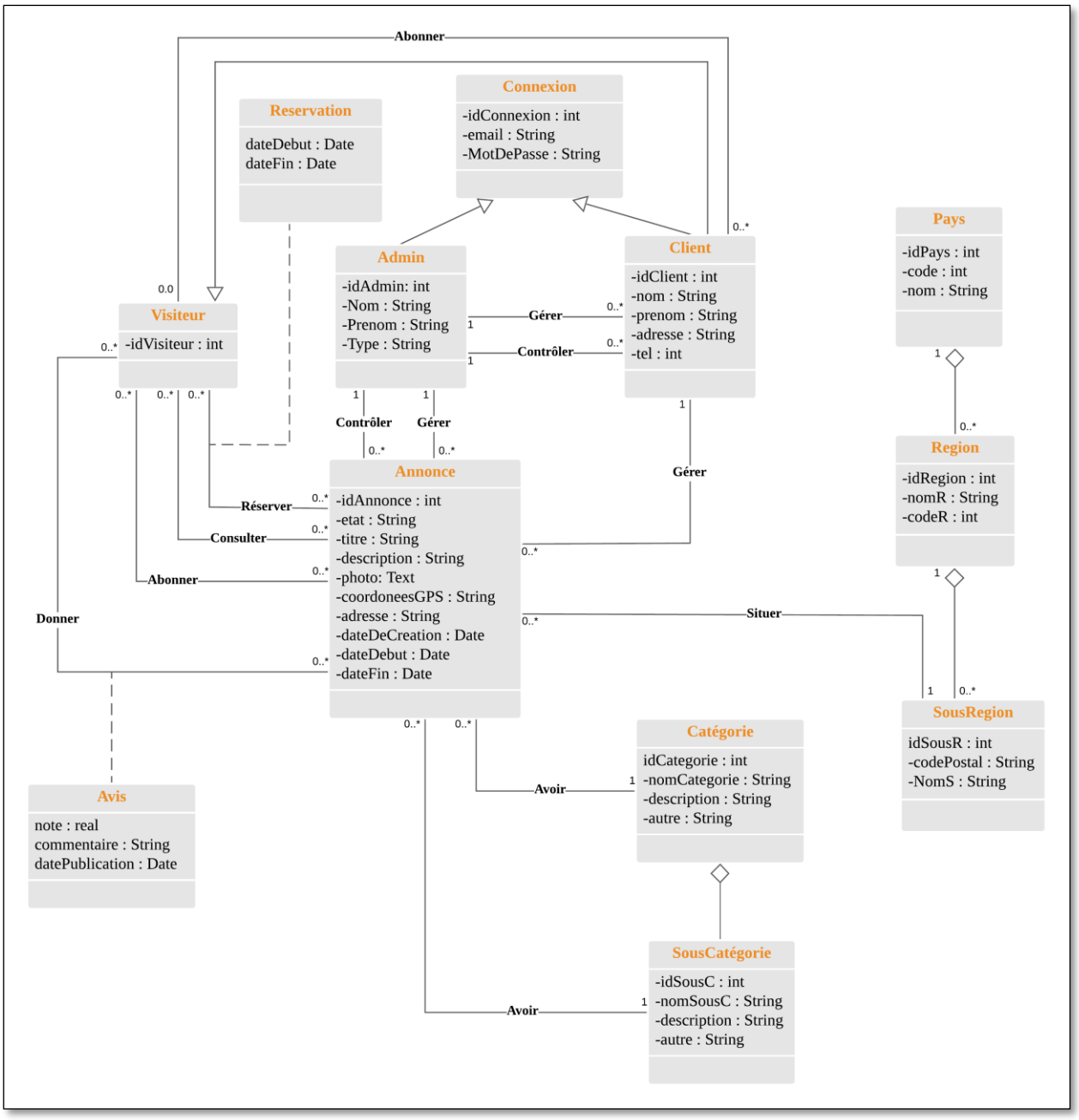

Figure III. 8 : Diagramme de classes.

III.3.4 Modèle Logique de Données

La description conceptuelle a permis de représenter le plus fidèlement possible les réalités de l'univers informatisé. Mais cette représentation ne peut pas être directement manipulée et acceptée par un système informatique, car le diagramme de classe est une

représentation des données dans un formalisme compris par les concepteurs et pas par la machine. Il est donc nécessaire de passer du niveau conceptuel à un second niveau plus proche des capacités des systèmes informatiques.

L'application des règles de passage (Voir ANNEXE 2), nous permet d'avoir le modèle logique de données associé au diagramme de classe de la « figure III.7 ».

Connexion (idConnexion, email, motDePasse, idAdmin#, idClient#).

Client (idClient, nom, prenom, adresse, tel, idAdmin#, idVisiteur#).

Admin (idAdmin, Nom, Prenom, Type).

Visiteur (idVisiteur, idClient#).

Annonce (idAnnonce, etat, titre, description, photo, coordoneesGPS, adresse, dateDeCreation, dateDebut, dateFin, idAdmin#, idClient#, idSousR#, idCategorie#, idSousC#).

Pays (idPays, code, nom).

Region (idRegion, nomR, codeR, idPays#).

SousRegion (idSousR, codePostal, NomS, idRegion#).

Catégorie (idCategorie, nomCategorie, description, autre).

SousCatégorie (idSousC, nomSousC, description, autre, idCategorie#).

Reservation (idVisiteur#, idAnnonce#, dateDebut, dateFin).

Avis (idVisiteur#, idAnnonce#, note, commentaire, datePublication).

Abonnement (idAnnonce#, idClient#).

III.3.5 Dictionnaire de Données

Le dictionnaire de données répertorie tous les termes et leurs définitions.

La table « Connexion », nous permet de stocker les informations techniques relatives à l'identification, ainsi que l'état du compte.

	Définition de l'attribut	*Attribut*	*Type*

Connexion	L'identifiant de la classe « Connexion »	idConnexion	Int(10)
	Email du client	email	Varchar (100)
	Le mot de passe haché avec un algorithme personnalisé.	motDePasse	Varchar (200)
	L'identifiant de la classe « Admin »	idAdmin	
	L'identifiant de la classe « Client »	idClient	Varchar (50)

Tableau III. 7 : Le glossaire de la table « Connexion ».

La table « Admin », nous permet de spécialiser l'administrateur de l'application et d'enregistrer ces informations

	Définition de l'attribut	*Attribut*	*Type*
Administrateur	L'identifiant de la classe « Administrateur »	IdAdmin	Int (10)
	Nom de l'Administrateur	Nom	Varchar (30)
	Prénom de l'Administrateur	Prenom	Varchar (30)
	Le type de l'Administrateur	Type	Varchar (30)

Tableau III. 8 : Le glossaire de la table « Administrateur ».

La table « Visiteur » nous permet d'enregistrer les informations du visiteur.

	Définition de l'attribut	*Attribut*	*Type*
Visiteur	L'identifiant de la classe « Utilisateur »	idVisiteur	Int (10)
	L'identifiant de la classe « Client »	idClient	Int (10)

Tableau III. 9 : Le glossaire de la table « Visiteur ».

La table « Client » nous permet d'enregistrer les informations du client.

	Définition de l'attribut	*Attribut*	*Type*
Client	L'identifiant de la classe « Client »	idClient	Int (10)
	Le nom du client	nom	Varchar (50)
	Le prénom du client	prenom	Varchar (50)
	L'adresse ou réside le client	adresse	Varchar (50)
	Le numéro de téléphone du client	tel	Int (10)
	L'identifiant de la classe « Admin »	idAdmin	Int(10)
	L'identifiant de la classe « Visiteur »	idVisiteur	Int(10)

Tableau III. 10 : Le glossaire de la table « Client ».

La table « Annonce » nous permet de spécialiser les annonces et d'enregistrer leurs informations.

	Définition de l'attribut	*Attribut*	*Type*
Annonce	L'identifiant de la classe « Annonce »	IdAnnonce	Int (10)
	L'état de l'annonce	etat	Varchar (50)
	Le titre de l'annonce	titre	Varchar (100)
	Des illustrations sur le produit	photo	Json
	La date de publication de l'annonce	coordoneesGPS	Varchar (100)
	Une description générale du produit	description	Varchar (300)
	L'adresse ou on peut trouver le produit	adresse	Varchar (100)
	La date de la création du produit	dateDeCreation	Date
	La date de début de l'offre	dateDebut	Date
	La date de fin de l'offre	dateFin	Date
	L'identifiant de la classe « Admin »	idAdmin	Int (10)
	L'identifiant de la classe « Client »	idClient	Int (10)

	L'identifiant de la classe « SousRegion »	idSousR	Int (10)
	L'identifiant de la classe « Catégorie »	idCategorie	Int (10)
	L'identifiant de la classe « SousCatégorie »	idSousC	Int (10)

Tableau III. 11 : Le glossaire de la table « Annonce ».

La table « Pays » nous permet d'avoir des informations sur le pays où se trouve l'annonce.

	Définition de l'attribut	*Attribut*	*Type*
Pays	L'identifiant de la classe « Pays »	idPays	Int (10)
	Le code correspondant au pays	code	Int (20)
	Le nom du pays	nom	Varchar (50)

Tableau III. 12 : Le glossaire de la table « Pays ».

La table « Region » nous permet d'avoir les informations de la région où se trouve l'annonce.

	Définition de l'attribut	*Attribut*	*Type*
Region	L'identifiant de la classe « Région »	idRegion	Int (10)
	Le nom de la région	nomR	Varchar (100)
	Le code de la région	codeR	Int (50)
	L'identifiant de la classe « Pays »	idPays	Int(10)

Tableau III. 13 : Le glossaire de la table « Region ».

La table « SousRegion » nous permet d'enregistrer les informations de la sous-région où se trouve l'annonce.

	Définition de l'attribut	*Attribut*	*Type*

SousRegion	L'identifiant de la classe « SousRegion »	idSousR	Int (10)
	Le code postal correspondant à la sous-région.	codePostal	Varchar (20)
	Nom de la sous-région.	NomS	Varchar (50)
	L'identifiant de la classe « Region ».	idRegion	Int (10)

Tableau III. 14 : Le glossaire de la table « SousRegion ».

La table « Catégorie » nous permet d'enregistrer les informations de la catégorie de l'annonce.

	Définition de l'attribut	*Attribut*	*Type*
Catégorie	L'identifiant de la classe « Catégorie ».	idCategorie	Int (10)
	Le nom de la catégorie.	nomCategorie	Varchar (100)
	Une description de la catégorie.	description	Varchar (300)
	Des informations sur la catégorie.	autre	Varchar (150)

Tableau III. 15 : Le glossaire de la table « Catégorie ».

La table « SousCatégorie » nous permet d'enregistrer les informations de la sous-catégorie de l'annonce.

	Définition de l'attribut	*Attribut*	*Type*
	L'identifiant de la classe « SousCatégorie »	idSousC	Int (10)

SousCatégorie	Le nom de la sous-catégorie	nomSousC	Varchar (100)
	Une description de la sous-catégorie	description	Varchar (300)
	Des informations diverses sur la « sous-catégorie »	autre	Varchar (100)
	L'identifiant de la classe « Catégorie »	idCategorie	Int (10)

Tableau III. 16 : Le glossaire de la table « SousCatégorie ».

La table « Reservation » nous permet d'avoir des informations sur l'association entre le Visiteur et l'Annonce.

Reservation	*Définition de l'attribut*	*L'attribut*	*Type*
	L'identifiant de la classe Visiteur	idVisiteur	Int (10)
	L'identifiant de la classe Annonce	idAnnonce	Int (10)
	La date de début de la location	dateDebut	Date
	La date de la fin de la location	dateFin	Date

Tableau III. 17 : Le glossaire de la table « Reservation ».

La table « avis » nous permet d'avoir des informations sur l'association entre l'utilisateur et l'annonce.

Avis	*Définition de l'attribut*	*Attribut*	*Type*
	L'identifiant de la classe « Visiteur »	IdVisiteur	Int (10)
	L'identifiant de la classe « Annonce »	IdAnnonce	Int (10)
	Commentaire sur l'association	commentaire	Varchar (100)
	Note de l'association	note	real
	La date de la publication de l'avis.	datePublication	Date

Tableau III. 18 : Le glossaire de la table « Avis ».

La table « Abonnement » nous permet d'accéder aux informations concernant les abonnements de notre client par rapport aux annonces.

	Définition de l'attribut	*Attribut*	*Type*
Abonnement	L'identifiant de la classe « Annonce »	IdAnnonce	Int (10)
	L'identifiant de la classe « Client »	idClient	Int(10)

Tableau III. 19 : Le glossaire de la table « Abonnement ».

III.4 Conclusion

Ce chapitre a été consacré à la conception de l'application web. En premier lieu, nous avons exposé les différentes interactions entre les objets du système grâce aux diagrammes de séquence détaillés. Sur ce, nous sommes arrivés à l'obtention du schéma relationnel en passant par la conception du modèle logique de données, déduit à partir du diagramme de classe.

Désormais, la phase conception est terminée, nous pouvons à présent enchaîner avec le dernier chapitre de notre projet qui sera consacré à la réalisation et la mise en œuvre de notre application.

Chapitre IV

Réalisation

IV.1 Introduction

Dans ce chapitre, nous allons définir des logiciels et les environnements du développement utilisés, et nous allons exposer quelques caractéristiques de notre application « ValizaDz ».

IV.2 Environnement de Logiciels

Ce tableau montre les configurations de téléphones où notre application peut être installée et utilisée.

Recommandations smartphones	
Os	Android
Version minimale	4.2.2
RAM minimum	512 Mo
Stockage libre	50 Mo
Connexion internet	Oui
GPS	Oui

Tableau IV. 1 : Recommandations minimales des smartphones.

IV.3 Environnement du Développement

Nous allons définir les différentes plateformes, logiciels et technologies que nous avions utilisés dans le développement de notre application mobile.

IV.3.1 Plateformes

Drive : permet de sauvegarder tous nos fichiers de manière sécurisée et d'y accéder depuis n'importe quel appareil. Elle permet également d'inviter facilement d'autres personnes à consulter, à modifier ou à commenter nos fichiers ou nos dossiers [23].

IV.3.2 Logiciels

IV.3.2.1 Logiciels du Développement

Visual Studio Code : Est un éditeur de code puissant, il prend en charge les opérations du développement tel que le débogage et l'exécution des taches, il fonctionne sur Windows, MacOs et linux [21].

MySQL : MySQL est un serveur de bases de données relationnelles Open Source. Un serveur de bases de données stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table [6].

IV.3.2.2 Logiciels de modélisation

Lucidchart : C'est une **solution évolutive de création de divers schémas** pou entreprises. Elle facilite la réalisation des diagrammes [20].

Modelio : Est un environnement de modélisation open source, il offre plusieurs fonctionnalités pour les développeurs de logiciels, les analystes et les concepteurs [19].

IV.3.2.3 Logiciels d'édition de textes

Adobe Acrobat XI Pro : Est un outil qui permet de créer, fusionner et modifier des documents de format PDF, il dispose d'une fonctionnalité spécifique dédiée à la création des formulaires web [18].

IV.3.2.4 Logiciels de traitement d'images

Photoshop : Logiciel de traitement et de retouche d'images et de photos, produit par la société Adobe. Photoshop est devenu le standard en matière de gestion des images matricielles (ou images "bitmap", constituées d'un "tapis de points") [17].

Adobe Illustrator : C'est un logiciel produit par Adobe Systems, il fonctionne souvent, mais pas nécessairement, avec d'autres logiciels de la gamme Adobe tels que Photoshop, InDesign ou After effect. Illustrator permet de créer et de modifier des dessins vectoriels, c'est-à-dire, des graphismes définis par des courbes mathématiques et non des pixels comme cela est le cas pour les images classiques (formats bitmap) [16].

IV.3.3 Technologies utilisées

Apache Cordova : C'est un Framework de développement mobile open-source. Il permet d'exploiter les technologies Web courantes telles que HTML5, CSS3 et JavaScript, pour développer des applications multi-plateformes, en évitant ainsi l'utilisation des langages natifs propres aux différentes plates-formes mobiles [11].

NodeJs : Est un langage interprété coté serveur écrit en JavaScript, il met en disposition des bibliothèques JavaScript selon le besoin, grâce au gestionnaire de baquet NPM [5].

SQL : Est l'acronyme de Structured Query Language. C'est un langage qui permet de communiquer des instructions à la base de données, pour stocker et lire les données [13].

AJAX : Désigne une architecture logicielle permettant de créer des pages et des applications web, capables d'interagir avec l'utilisateur et/ou d'autres applications sans la nécessité de recharger cette page dans le navigateur web du poste client. [10].

Javascript : JavaScript est un langage de programmation qui permet d'implémenter des mécanismes complexes sur une page web. C'est un langage de programmation qui permet de créer du contenu mis à jour de façon dynamique, contrôler le contenu multimédia, animer des images…[12].

CSS : CSS est l'acronyme de « Cascading Style Sheets » ce qui signifie « feuille de style en cascade ». Le CSS correspond à un langage informatique permettant de mettre en forme des pages web (HTML ou XML). Ce langage est donc composé des fameuses « feuilles de style en cascade » également appelées fichiers CSS (.css) et contient des éléments de codage [8].

HTML : L'HTML est un langage informatique utilisé sur l'internet. Ce langage est utilisé pour créer des pages web. L'acronyme signifie HyperText Markup Language, ce qui signifie en français "langage de balisage d'hypertexte". Cette signification porte bien son nom puisque ce langage permet de réaliser de l'hypertexte à base d'une structure de balisage [7].

Bootstrap : C'est un Framework développé par l'équipe du réseau social Twitter. Proposé en open source (sous licence MIT), ce Framework utilisant les langages HTML, CSS et JavaScript fournit aux développeurs des outils pour créer un site facilement [9].

Les applications progressives : Une progressive web app (PWA) est une application mobile accessible via le Web. Les PWA ont la même sensation et la même fonctionnalité que les applications natives. Cependant, les fichiers PWA n'ont pas besoin d'être téléchargés depuis l'App Store. Par contre, ils sont accessibles via un navigateur Web et charger instantanément grâce à l'assistance des travailleurs du service [14].

Les applications à une page (Single page application) : Une SPA est une application qui utilise le navigateur de l'utilisateur pour générer la page qu'il doit afficher. On dit que le code s'exécute côté client. Cela permet de naviguer sur toute l'application web sans rechargement de la page [15].

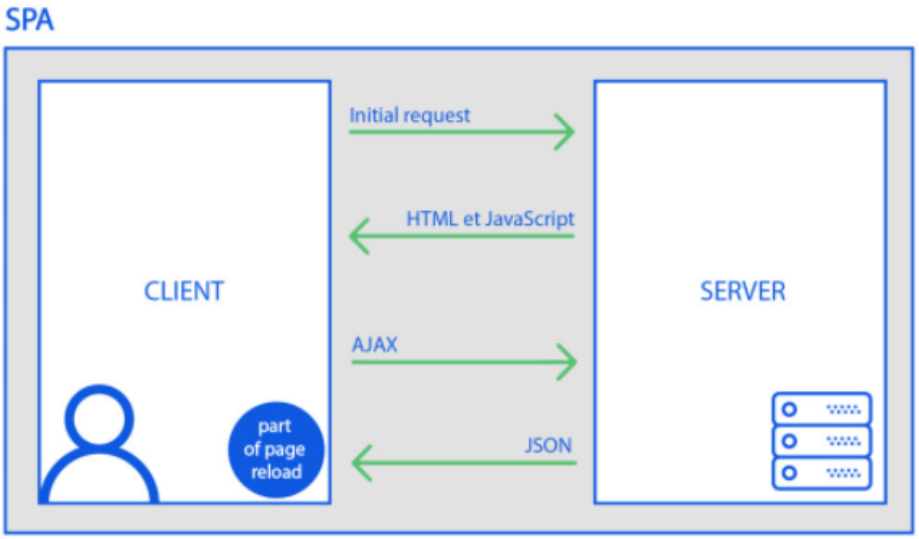

Figure IV. 1 : Schéma explicative d'une application à une page.

IV.4 Caractéristiques

IV.4.1 Caractéristiques Fonctionnelles

Nous allons spécifier les différentes caractéristiques fonctionnelles de notre application « ValizaDz ».

Côté Visiteur

L'inscription : Un compte est attribué pour chaque visiteur qui s'inscrit, ce qui permet d'avoir plus de visibilité sur le contenu de l'application, et d'avoir plus d'accès sur les ressources qu'il contient.

La consultation des articles : Le visiteur peut consulter une grande variété d'annonces, il a la possibilité de les lister par catégories et sous catégories.

Côté Client

La recherche personnalisée : le client peut effectuer une recherche d'annonce avec son nom ou une de ses caractéristiques.

La consultation des articles : Le visiteur peut consulter une grande variété d'annonces et il a la possibilité de les lister par catégories et sous catégories.

La réservation d'un produit : Le système permet au client de réserver un produit dans les catalogues et profiter de l'annonce qu'il trouve adéquate.

La publication d'une annonce : Le client pourra ajouter une annonce pour un produit qu'il souhaite louer.

Gestion des informations personnelles : Le client pourra modifier ces informations qui vont apparaitre sur son profil.

Côté Administrateur

La validation des annonces : l'administrateur valide la publication des annonces et peut les décliner.

La gestion des utilisateurs : valide la création d'un profil client et peut aussi ajouter un administrateur.

IV.4.2 Caractéristiques Non-Fonctionnelles

Nous allons spécifier l'ensemble des caractéristiques non-fonctionnelles de notre application :

La confidentialité : Il s'agit tout d'abord d'interdire l'accès en lecture et en écriture aux informations considérées comme privées à chaque utilisateur.

La sécurité : L'application doit la confidentialité des données personnelles des clients.

La fiabilité : L'application doit fonctionner de façon cohérente sans erreurs et doit être satisfaisante.

La réutilisation : L'application est conforme à une architecture standard permettant sa réutilisation.

Les droits d'accès : L'accès sera contrôlé selon le profil et le type d'utilisateur.

IV.5 Manuel de l'Application

Nous allons présenter, dans ce qui suit, quelques exemples représentatifs de l'IHM de notre application.

IV.5.1 IHM du Visiteur

La page du formulaire d'inscription : c'est l'interface qui s'affiche lors d'une demande d'inscription. Là dans l'image ci-dessous, nous demandons au visiteur de fournir ces informations personnelles pour devenir client de notre application.

Figure IV. 2 : L'interface « Formulaire d'inscription ».

L'interface annonces : Nous donne une liste des annonces disponibles avec une vue générale : titre et photo et d'autres infos.

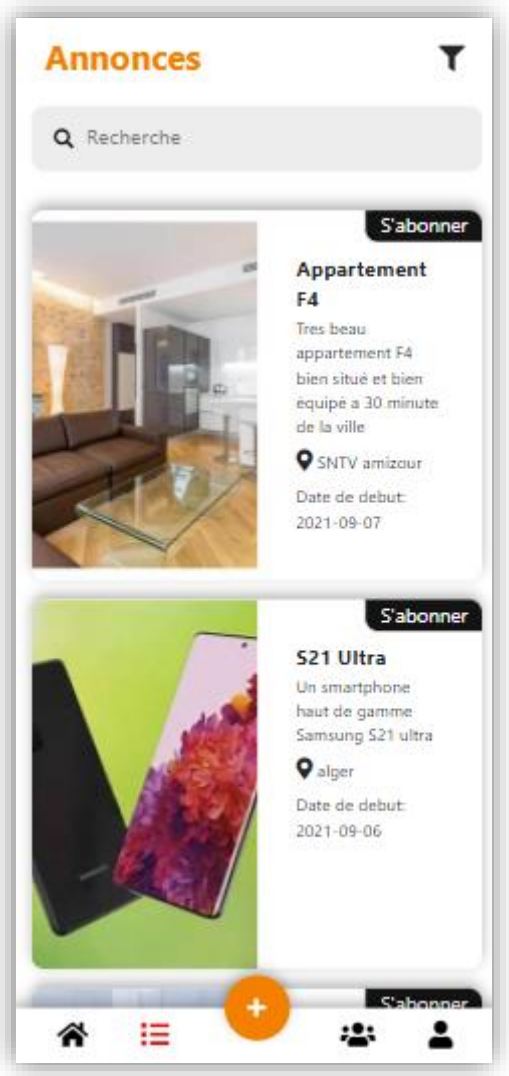

Figure IV. 3 : L'interface « Annonces ».

L'interface consulter une annonce : Elle contient toutes les informations d'une annonce existante sur la plateforme. Nous pouvons trouver des photos, une description, la localisation et d'autres informations.

Figure IV. 4 : L'interface « Consulter une annonce ».

IV.5.2 IHM du Client

L'interface Connexion : Elle contient deux champs à remplir pour s'authentifier en tant que client courriel et le mot de passe. Et deux boutons : « connexion » et « Mode visiteur ». Cela nous permet aussi de cliquer sur « S'inscrire » pour nous rediriger vers le formulaire d'inscription.

Figure IV. 5 : L'interface « Connexion ».

L'interface du formulaire « Ajouter une annonce » : C'est l'interface qui s'affiche lors d'une demande d'ajout d'une annonce. Dans la présente image, nous demandons au client de nos fournir des informations sur l'annonce qu'il veut publier.

Figure IV. 6 : L'interface « Ajouter une annonce ».

L'interface formulaire de « Réservation d'une annonce » : C'est le formulaire qui s'affiche lorsque nous prévoyons de réserver un produit proposé sur une annonce. Nous demandons au client de fournir ces informations personnelles ainsi que la période dans laquelle il souhaite louer le produit, comme nous l'indique l'image ci-dessous :

Figure IV. 7 : Interface « Réservation d'une annonce ».

IV.5.3 IHM de l'Administrateur

L'interface « Valider Annonce » : Est sous forme d'une boite de dialogues qui a deux boutons Ok et Annuler. Elle permet à l'administrateur de valider une annonce que le client souhaite publier, une fois validée, l'annonce apparait sur le fil d'actualité de l'application.

Figure IV. 8 : Interface « Valider une annonce ».

L'interface « Décliner Annonce » : Est sous forme d'une boite de dialogues qui a deux boutons Ok et Annuler. Elle permet à l'administrateur de décliner une annonce que le client souhaite publier considérant qu'elle ne respecte pas les standards de notre application.

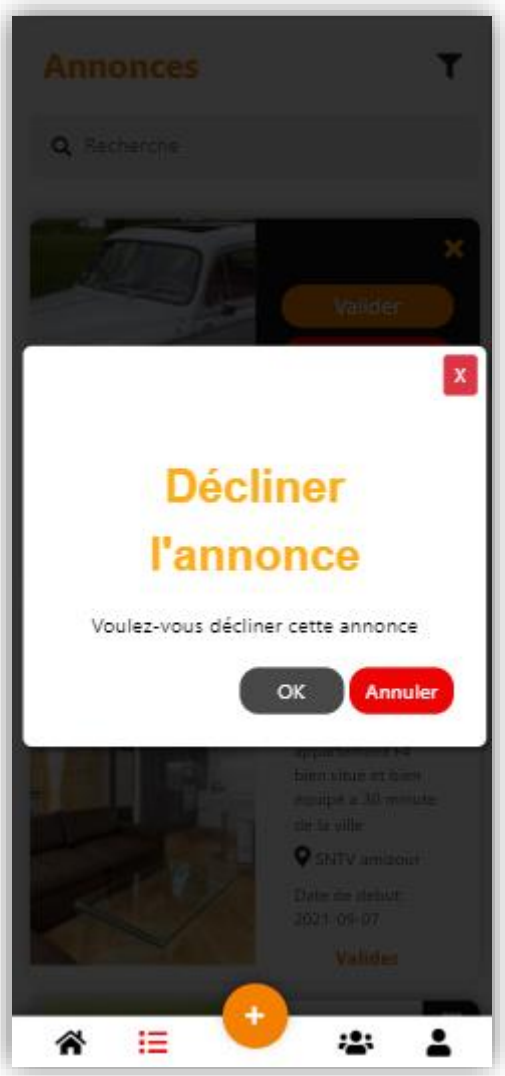

Figure IV. 9 : Interface « Décliner une annonce ».

L'interface « Annonce Admin » : Nous donne une liste des annonces publiées par les clients et qui sont en attente d'une validation, celles-ci sont suivies par des annonces déjà validées qui sont affichées dans le fil d'actualité de l'application.

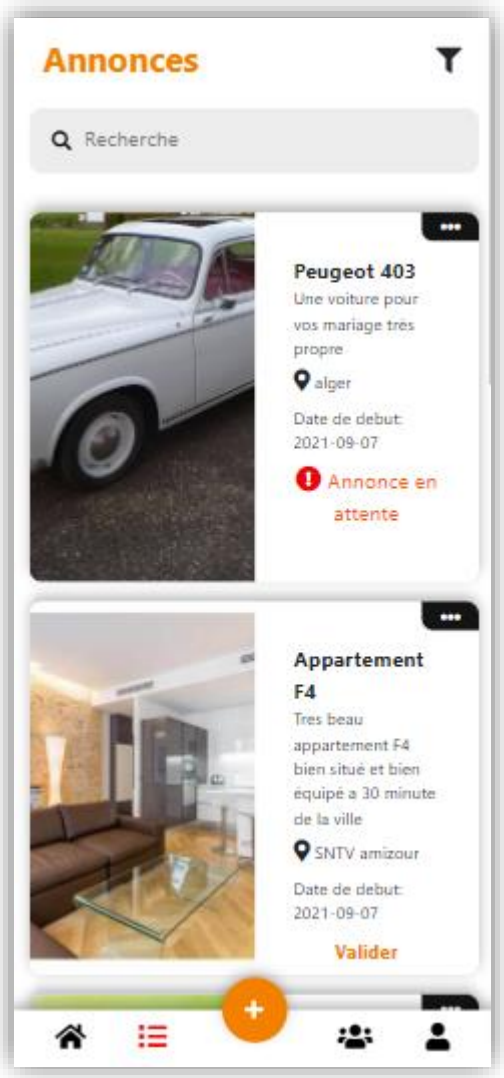

Figure IV. 10 : Interface « Annonce Admin ».

IV.6 Conclusion

Nous avons présenté au cours de ce chapitre la réalisation de notre application « ValizaDz » avec les différentes technologies, l'environnement logiciel et le développement, ainsi qu'une architecture globale de l'application, ainsi que les principales interfaces de notre application pour nos trois acteurs (Visiteur, Client ainsi que l'administrateur). Cela aidera à mieux comprendre le fonctionnement de l'application, son ergonomie et peut servir comme

manuel d'utilisation. Nous avons choisi d'exposer les interfaces les plus pertinentes et importantes.

Conclusion Générale

Notre travail était dans le but de concevoir une application mobile dédiée à la location de différents produits. Cette application mobile permet de louer des produits dont nous avons besoins ainsi que de proposer des produits pour la location. Pour ce faire, nous avons eu recours à un stage au sein de l'entreprise « VegaSoft » dans le but de nous impliquer et d'apprendre à développer d'une manière professionnelle et de bien comprendre les volontés du client.

Pour débuter notre rapport, nous avons commencé par recueillir des informations détaillées sur le projet du client de « VegaSoft », où cela nous a permis de comprendre notre objectif à réaliser, qui était de concevoir une application mobile pour tout type de location. Nous avons analysé quelques applications similaires. Ensuite, nous avons élaboré un cahier de charge qui illustre les besoins fonctionnels, et non fonctionnel du travail demandé.

Cela nous a aidé à comprendre comment mettre en pratique nos connaissances théoriques sur les démarches de développement. Nous avons appris aussi, à mieux utiliser UML et les méthodes agiles sur le terrain.

Enfin, nous avons pu réaliser notre travail en respectant le cahier de charge, et en ajoutant des fonctionnalités que nous avons jugées intéressantes et en développant une application mobile respectant les standards de nos jours (Temps de chargement réduit, interface utilisateur simple et fluide, …). Notre application est une innovation première de son genre sur le territoire national, nous apportons à nos utilisateurs l'opportunité de louer et de mettre en location tout type de produits avec de simple manipulation.

En guise de perspective, nous espérons élargir le domaine de notre application en rajoutant d'autres fonctionnalités comme une messagerie interne qui permettra à nos utilisateurs de communiquer directement sur notre application, nous comptons aussi gérer les réservations de nos clients (consulter ses réservations, consulter le planning des réservations d'un produit…). Nous essayerons aussi d'élargir l'espace géographique que ValizaDz couvre en dotant notre application d'une api de localisation pour trier les annonces qui s'affiche à l'utilisateur par rapport à sa position géographique et le pays dans lequel il se trouve.

Bibliographie

[1] : Google play store : ACR, consulté le 02/07/2021.

[2] : Google play store : seLoger, consulté le 02/07/2021.

[3] : AUDIBERT, Laurent, UML 2 De l'apprentissage à la pratique, Dévloppez.com,2éme édition, 2009, 142.

[4] : « Le cycle de vie de la méthode XP (eXtreme Programming) ». alecoledelavie.com. A L'ECOLE DE LA VIE. 19 novembre. 2015. Web. Consulté le 06 juin. 2021.

[5] : Site officiel de nodeJs,https://nodejs.org/en/, consulté le 05 juin 2021.

[6] : Site officiel de MySql, https://www.mysql.com/fr/, consulté le 07 juin 2021.

[7] : Glossaire infowebmaster, http://glossaire.infowebmaster.fr/html/, consulté le 07 juin 2021.

[8] : Glossaire atinternet, https://www.atinternet.com/glossaire/css/, consulté le 06 juin 2021.

[9] : Blog de hostinger, https://www.hostinger.fr/tutoriels/cest-quoi-bootstrap/, consulté le 8 juillet 2021.

[10] : Site dictionnaire du web, https://www.1min30.com/dictionnaire-du-web/ajax, consulté le 15 juillet 2021.

[11] : Site officiel de cordova https://cordova.apache.org/docs/fr/10.x/guide/overview/index.html, consulté le 17 juin 2021.

[12] : Site MDN Web Docs https://developer.mozilla.org/fr/docs/Learn/JavaScript/First_steps/What_is_JavaScript, consulté le 18 juin 2021.

[13] : Site officiel du SQL, https://sql.sh/, Consulté le 17 juin 2021.

[14] : Blog de stardust, https://www2.stardust-testing.com/blog-fr/pourquoi-lavenir-appartient-aux-applications-web-progressives, consulté le 04 Aout 2021.

[15] : Blog yoozly, https://www.yoozly.com/blog/single-page-application, Consulté le 04 Aout 2021.

[16] : Site Sculpteo, https://www.sculpteo.com/fr/glossaire/logiciel-illustrator-definition/, Consulté le 18 Aout 2021.

[17] : Glossaire web Référencement http://www.mosaique-info.fr/glossaire-web-referencement-infographie-multimedia-informatique/p-glossaire-informatique-et-multimedia/228-photoshop-definition.html, Consulté le 2 Aout 2021.

[18] : Blog 01net, https://www.01net.com/telecharger/windows/Bureautique/editeur_de_texte/fiches/36771.html , Consulté le 21 juillet 2021.

[19] : Site officiel modelio, https://www.modelio.org/, Consulté le 19 juillet 2021.

[20] : Site de appvizer, https://www.appvizer.fr/operations/business-process/lucidchart, Consulté le 15 Aout 2021.

[21] : Site officiel de Visual Studio, https://code.visualstudio.com/, Consulté le 02 juillet 2021.

[22] : Site officiel de l'université de Paris 13, https://lipn.univ-paris13.fr/~gerard/uml-s2/uml-cours05.html, Consulté le 3 juillet 2021.

[23] :https://play.google.com/store/apps/details?id=com.google.android.apps.docs&hl=fr&gl=US, Consulté le 20 Aout 2021.

[24] : Omar, Mawloud. « Cours de génie logiciel », université de Béjaia, Béjaia, 3 Mars 2019.

Annexe 1

1 Les méthodes agiles

Les méthodes de développement dites « méthodes agiles » (en anglais Agile Modeling, noté AG) visent à réduire le cycle de vie du logiciel (donc accélérer son développement) en développant 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. L'origine des méthodes agiles est liée à l'instabilité de l'environnement technologique et au fait que le client est souvent dans l'incapacité de définir ses besoins de manière exhaustive dès le début du projet. Le terme « agile » fait ainsi référence à la capacité d'adaptation aux changements de contexte et aux modifications de spécifications intervenant pendant le processus de développement.

L'agilité comprend plusieurs courants de pensée qui ont conduit à des méthodologies reposant sur les mêmes concepts mais présentant chacune des singularités. Présentant successivement eXtreme Programming, DSDM, ASD, CRYSTAL, SCRUM, FDD.

1.1 XP : eXtreme Programming

La méthode XP (pour eXtreme Programming) définit un certain nombre de bonnes pratiques permettant de développer un logiciel dans des conditions optimales en plaçant le client au cœur du processus de développement, en relation étroite avec le client.

1.1.1 Aux origines d'eXtreme Programming

En introduisant cette méthode, ses créateurs veulent en finir avec la dérive de délais, l'annulation des projets, la non-qualité, la mauvaise compréhension du métier du client et l'impossibilité de suivre les changements.

Son but principal est de réduire les coûts du changement. Dans les méthodes traditionnelles, les besoins sont définis, et souvent fixés, au départ du projet informatique, ce qui accroît les coûts ultérieurs de modifications. XP s´attache à rendre le projet plus flexible et ouvert au changement en introduisant des valeurs de base, des principes et des pratiques.

XP a été mis en œuvre pour la première fois en 1996 sur le projet 'C3', qui consistait à mettre en place un nouveau système de gestion de la paie des dix mille salariés du fabricant automobile. Sur ce projet, la méthode XP a été adoptée pour tenter de sauver le projet d'une dérive annoncée.

1.1.2 Les 5 valeurs fondamentales d'eXtreme Programming

XP met en avant quatre valeurs prenant en considération à la fois les enjeux commerciaux et les aspects humains des projets de développement d'applications : Feedback, Simplicité, Communication, Respect et le Courage.

Feedback (Retour de l'information)

Cette boucle essentielle de va-et-vient différencie les systèmes Agiles en général et la programmation extrême en particulier, des autres méthodologies de gestion de projet logiciel. La rétroaction continue peut fonctionner de différentes manières, mais elles contribuent toutes à rendre le système plus fort et plus fiable.

Les commentaires peuvent prendre différentes formes :

- Depuis le programme lui-même : le code est vigoureusement testé tout au long du cycle de développement du projet, afin que les modifications puissent être mises en œuvre par les développeurs.
- Du client : c'est une partie essentielle de la plupart des systèmes Agiles. Les clients écrivent les tests d'acceptation sur lesquels le développement est basé, ce qui constitue l'épine dorsale du processus de développement. Toutes les itérations sont également livrées au client, pour un retour périodique.
- De l'équipe : Une fois qu'un nouveau cas d'utilisation / histoire a été créé (user story), l'équipe revient immédiatement avec l'estimation des coûts et du calendrier, renforçant les exigences à mesure qu'elles surviennent. Dans eXtreme Programming, personne ne possède de code et, par conséquent, au sein des équipes de programmation extrême, les commentaires sur le code de l'autre sont encouragés.

Simplicité

XP recommande de traiter tous les problèmes par la solution la plus simple possible, partant du principe qu'un travail propre, simple et minimal aujourd'hui est facile à améliorer par la suite.

Communication

Il est essentiel que chacun puisse dire sans peur qu'une partie de code n'est pas optimale, qu'il a besoin d'aide, etc.

Respect

Le respect de chaque membre de l'équipe et de son travail sont primordiaux. Le management, l'équipe projet et le client se respectent mutuellement.

Courage

Il faut du courage pour effectuer certains changements comme essayer une nouvelle technique, recommencer une itération non validée ou revoir l'organisation du projet. Le courage permet de sortir d'une situation inadaptée.

1.2 Cycle de vie

Ce paragraphe donne une vision du cycle de vie idéal d'un projet XP. Plusieurs phases composent le cycle de vie d'une application.

Exploration

Au cours de cette phase, les développeurs se penchent sur des questions d'ordre technique destinées à explorer les différentes possibilités d'architecture pour le système et à étudier par exemple les limites au niveau des performances présentées par chacune des solutions possibles.

Le client de son côté s'habitue à exprimer ses besoins sous forme d'user stories que les développeurs devront estimer en termes de temps de développement.

User Story : Une user story est une demande fonctionnelle écrite de façon à mettre en avant les besoins utilisateurs. Elle est écrite dans un langage naturel compris par l'ensemble des acteurs du projet ou liés à celui-ci, avec :

Un acteur qui effectue **une action** dans **un objectif** donné. Une User story doit être développée entièrement pendant une itération.

Planning

Nous ne détaillerons pas ici les procédures entrant dans la phase de planning, Le planning de la première release (la première version de notre application) est fait de telle sorte qu'un système pourvu uniquement des fonctionnalités essentielles soit mis en production dans un temps minimum et soit enrichi par la suite.

Itérations jusqu'à la première release

C'est la phase de développement de la première version de l'application. Chaque itération produit un ensemble de fonctionnalités passant avec succès les tests fonctionnels associés. La première itération est dédiée à la mise en place de l'architecture du système.

Ces itérations courtes permettent de détecter rapidement toute déviation par rapport au planning qui a été fait jusqu'à la sortie de la release. En cas de déviation, quelque chose est à revoir, soit au niveau de la méthode, soit au niveau des spécifications, soit au niveau du planning.

Mise en production

Les itérations de cette phase sont encore plus courtes, ceci pour renforcer le feedback. En général, des tests parallèles sont conduits au cours de cette phase et les développeurs procèdent à des réglages affinés pour améliorer les performances (performance tuning). A la fin de cette phase, le système offre toutes les fonctionnalités indispensables et est parfaitement fonctionnel et peut être mis à disposition des utilisateurs.

Maintenance

Il s'agit dans cette phase de continuer à faire fonctionner le système désormais existant et de lui adjoindre les fonctionnalités secondaires qui avaient volontairement été laissées de côté jusque-là. Le développement de nouvelles fonctionnalités sur un système déjà mis en production requiert une prudence accrue de la part des développeurs et cette phase est donc cruciale. A chaque nouvelle release, une nouvelle phase d'exploration rapide doit avoir lieu.

Mort

La fin d'un projet intervient quand le client n'arrive plus à écrire d’user stories supplémentaires ce qui signifie que pour lui, tous ses besoins ont été satisfaits ou quand le système n'est plus capable de recevoir de modifications pour satisfaire de nouveaux besoins tout en restant rentable.

Ce cycle se répète tant que le client peut fournir des scénarios à livrer, comme l'explique la figure 1.

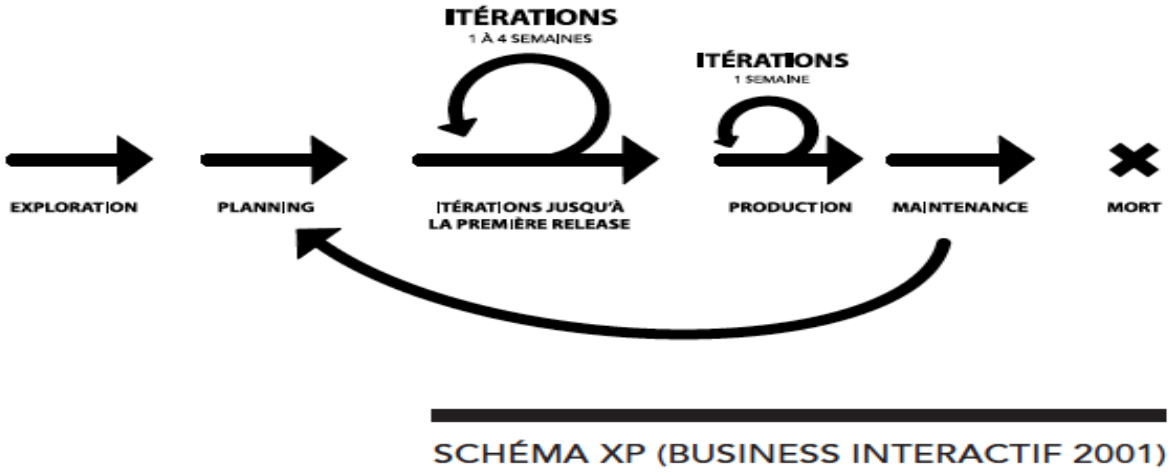

Figure 1 : Le cycle de vie de « XP ».

Annexe 2

1 Présentation générale d'UML

Le génie logiciel et la méthodologie s'efforcent de couvrir tous les aspects de la vie du logiciel. Issus de l'expérience des développeurs, concepteurs et chefs de projets, ils sont en constante évolution, parallèlement à l'évolution des techniques informatiques et du savoir-faire des équipes.

Comme toutes les tentatives de mise à plat d'une expérience et d'un savoir-faire, les méthodologies ont parfois souffert d'une formalisation excessive, imposant aux développeurs des contraintes parfois contre-productives sur leur façon de travailler. Avec la mise en commun de l'expérience et la maturation des savoir-faire, on voit se développer à présent des méthodes de travail à la fois plus proches de la pratique réelle des experts et moins contraignantes.

UML, qui se veut un instrument de capitalisation des savoir-faire puisqu'il propose un langage qui soit commun à tous les experts du logiciel, va dans le sens de cet assouplissement des contraintes méthodologiques.

UML (Unified Modeling Language) signifie « langage unifié de modélisation ». C'est un langage formel, normalisé et un support de communication performant qui permet grâce à sa représentation graphique, de concevoir des solutions, de faciliter la comparaison et l'évolution de celles-ci. Son caractère polyvalent et sa souplesse ont en fait un langage de modélisation universel. C'est un langage visuel dédié à la spécification, la construction et la documentation d'un système. L'OMG1 est actuellement responsable de documenter et de faire évoluer ce langage. UML intègre dans un même langage graphique des formalismes et des notations empruntés des méthodes de développement des systèmes qui ont fait leur preuve dont notamment les méthodes dites orientées objet.

1.1 Les diagrammes de l'UML

UML propose treize diagrammes complémentaires qui permettent la modélisation d'un projet tout au long de son cycle de vie, Chaque diagramme étant dédié à la représentation des concepts particuliers d'un système logiciel. Les treize types de diagrammes UML, sont répartis en deux catégories :

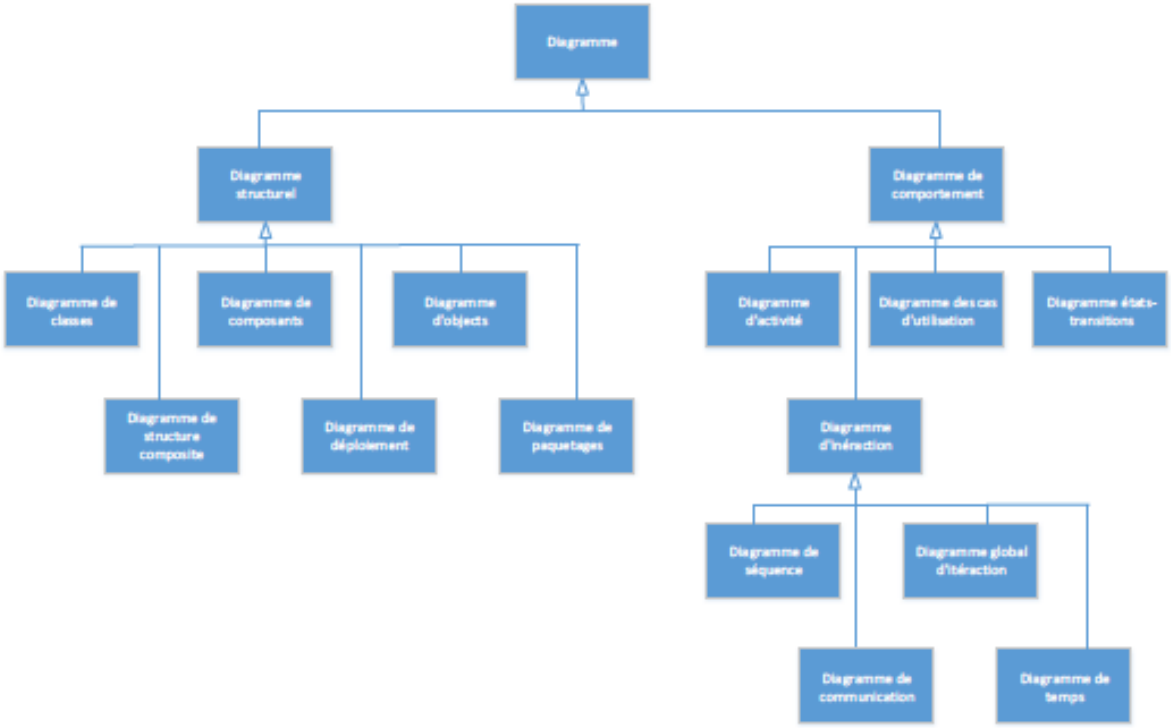

FIGURE 2 : Les 13 types de diagrammes de l'UML 2.0

Diagrammes structurels : Les six diagrammes structurels se présentent comme suit:

- Diagramme de classes : identifie la structure des classes d'un système y compris les propriétés et les méthodes de chaque classe.
- Diagramme d'objets : montre les instances des éléments structurels et leurs liens à l'exécution.
- Diagramme de packages : spécifie l'organisation logique du modèle et les relations entre packages.
- Diagramme de structure composite : désigne l'organisation interne d'un élément statique complexe.
- Diagramme de composants : indique des structures complexes avec leurs interfaces fournies et requises.
- Diagramme de déploiement : définit le déploiement physique des « artefacts » (Objet) sur les ressources matérielles.

Diagrammes comportementaux :

- Diagramme de cas d'utilisation : permet d'identifier les possibilités d'interactions entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système.
- Diagramme de vue d'ensemble des interactions : fusionne les diagrammes d'activité et de séquence pour combiner des fragments d'interaction avec des décisions et des flots.
- Diagramme de séquence : montre la séquence verticale des messages passés entre objets au sein d'une interaction.
- Diagramme de communication : désigne la communication entre objets dans le plan au sein d'une interaction.
- Diagramme de temps : fusionne les diagrammes d'états et de séquences pour montrer l'évolution de l'état d'un objet au cours du temps.
- Diagramme d'activité : montre l'enchaînement des actions et décisions au sein d'une activité.
- Diagramme d'états-transitions : permet de décrire sous forme de machines à états finis, le comportement du système ou de ses composants.

1.2 Règles de passage du diagramme de classes au modèle logique

Cette section présente les règles permettant de décrire un schéma logique dans les modèles relationnel et objet-relationnel à partir d'un schéma entité-association ou à partir d'un diagramme de classes UML.

- Transformation des entités/classes

Chaque entité devient une relation. L'identifiant de l'entité devient clé primaire pour la relation. Chaque classe du diagramme UML devient une relation. Il faut choisir un attribut de la classe pouvant jouer le rôle d'identifiant. Si aucun attribut ne convient en tant qu'identifiant, il faut en ajouter un de telle sorte que la relation dispose d'une clé primaire (les outils proposent l'ajout de tels attributs). On note cette règle par « R1 ».

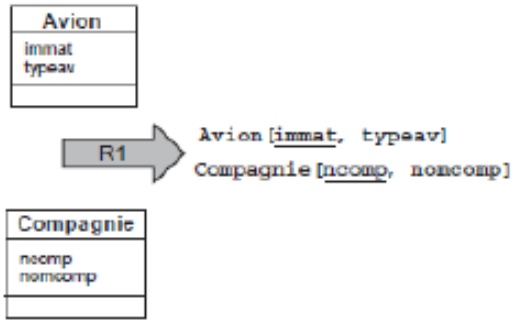

FIGURE 3 : Transformation d'entités/classes.

Remarque :

La règle R1 a été appliquée à l'exemple « **FIGURE 3** » de manière à dériver deux relations.

- Transformation des associations

Les règles de transformation que nous allons voire dépendent des cardinalités/multiplicités maximales des associations. Nous distinguons trois familles d'associations :

- Un-à-plusieurs.
- Plusieurs-à-plusieurs ou classes-associations, et n-aires.
- Un-à-un.

1. Associations un-à-plusieurs

Il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association.

L'attribut porte le nom de la clé primaire de la relation père de l'association. On note cette règle par « R2 ».

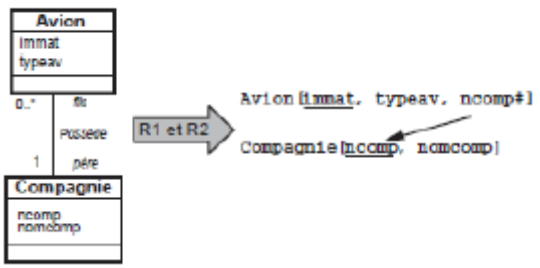

FIGURE 4 : Transformation d'une association un-à-plusieurs.

Remarque :

Les règles R1 et R2 ont été appliquées à l'exemple « **FIGURE 4** ». La règle R2 fait apparaître la clé étrangère ncomp dans la relation fils Avion qui a migré de la relation père Compagnie.

2. Associations plusieurs-à-plusieurs

L'association (classe-association) devient une relation dont la clé primaire est composée par la concaténation des identifiants des entités (classes) connectés à l'association. Chaque attribut devient clé étrangère.

Les attributs de l'association (classe-association) doivent être ajoutés à la nouvelle relation. Ces attributs ne sont ni clé primaire, ni clé étrangère. On note cette règle par « R3 ».

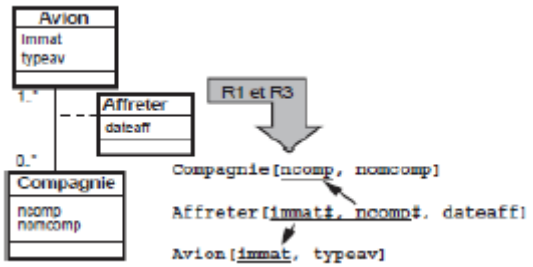

FIGURE 5 : Transformation d'une association plusieurs-à-plusieurs.

Remarque :

Les règles R1 et R3 ont été appliquées à l'exemple « **FIGURE 5** ». La règle R3 crée la relation Affreter dont la clé primaire est composée de deux clés étrangères. L'attribut dateaff de l'association est ajouté à la nouvelle relation.

3. Associations un-à-un

Il faut ajouter un attribut clé étrangère dans la relation dérivée de l'entité ayant la cardinalité minimale égale à zéro. Dans le cas de UML, il faut ajouter un attribut clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut portele nom de la clé primaire de la relation dérivée de l'entité (classe) connectée à l'association. Si les deux cardinalités (multiplicités) minimales sont à zéro, le choix est donné entre les deux

relations dérivées de la règle R1. Si les deux cardinalités minimales sont à un, il est sans doute préférable de fusionner les deux entités (classes) en une seule. On note cette règle par « R4 ».

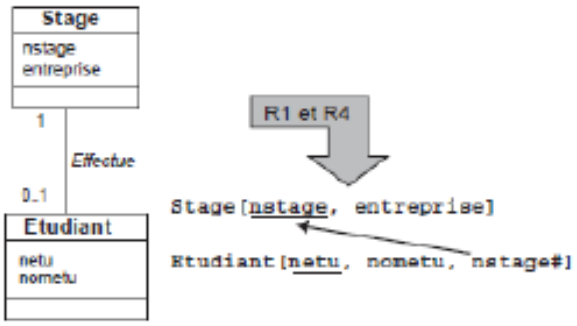

FIGURE 6 : Transformation d'une association un-à-un.

Remarque :

Les règles R1 et R4 sont appliquées à l'exemple « **FIGURE 6** ».

- Transformation de l'héritage

Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

- Décomposition par distinction.
- Décomposition descendante (push-down). S'il existe une contrainte de totalité ou de partition sur l'association d'héritage, il y aura deux cas possibles de décomposition.
- Décomposition ascendante (push-up).

Décomposition par distinction

Il faut transformer chaque sous-classe en une relation. La clé primaire de la sur-classe migre dans la (les) relation(s) issue(s) de la (des) sous-classe(s) et devient à la fois clé primaire et clé étrangère.

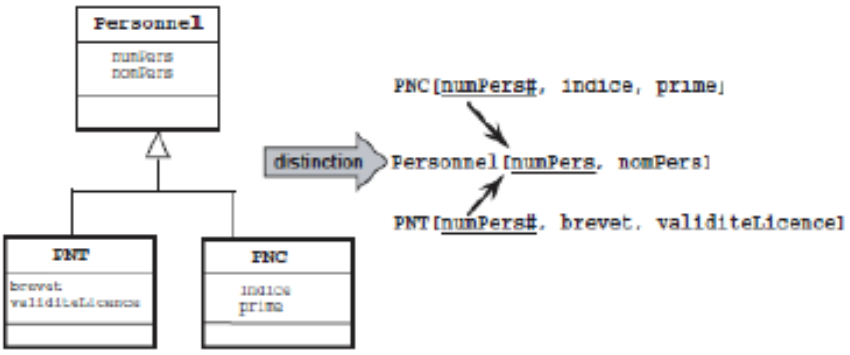

FIGURE 7 : Décomposition par distinction d'une association d'héritage.

Remarque :

L'association d'héritage est traduite en faisant migrer l'identifiant de la sur-classe (Personnel) dans les deux relations déduites des sous-classes. Cet attribut devient aussi clé étrangère.

Décomposition descendante (push-down)

S'il existe une contrainte de totalité ou de partition sur l'association, il est possible de ne pas traduire la relation issue de la sur-classe. Il faut alors faire migrer tous ses attributs dans la (les) relation(s) issue(s) de la (des) sous-classe(s).

Dans le cas contraire, il faut faire migrer tous ses attributs dans la ou les relation(s) issue(s) de la (des) sous-classe(s) dans la (les) relation(s) issue(s) de la (des) sous-classe(s).

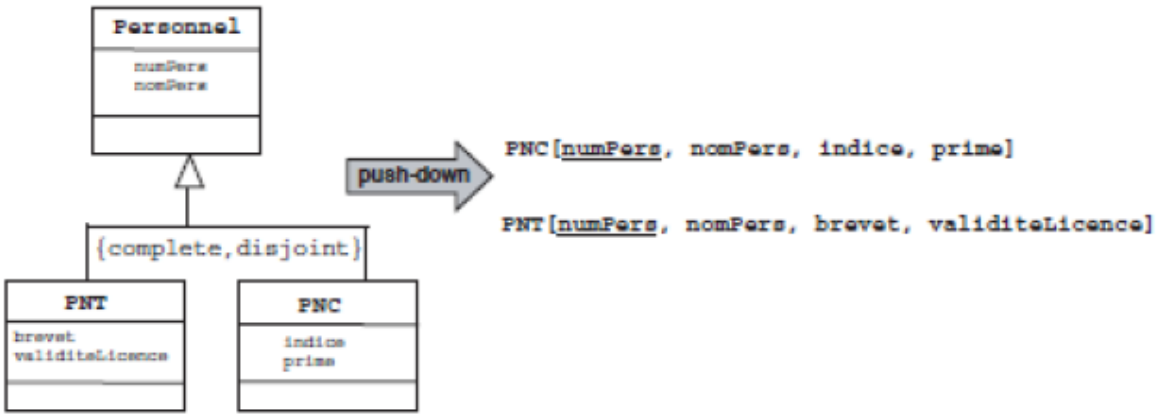

FIGURE 8 : Décomposition descendante (push-down) d'une association d'héritage.

Remarque :

L'exemple « **FIGURE 8** » décrit une contrainte de partition dans l'association d'héritage (aucun personnel ne peut être à la fois PNT et PNC et il n'existe pas un personnel n'étant ni PNT ni PNC). Les deux relations héritent du contenu intégral de la relation issue de la surclasse (Personnel). La relation Personnel n'apparaît plus au niveau logique et n'est pas nécessaire, car aucun personnel ne peut être ni PNT ni PNC. Ici, on peut dire que Personnel est une entité abstraite (il n'existera pas d'instance de cette entité).

Décomposition ascendante (push-up)

Il faut supprimer la (les) relation(s) issue(s) de la (des) sous-classe(s) et faire migrer les attributs dans la relation issue de la sur-classe.

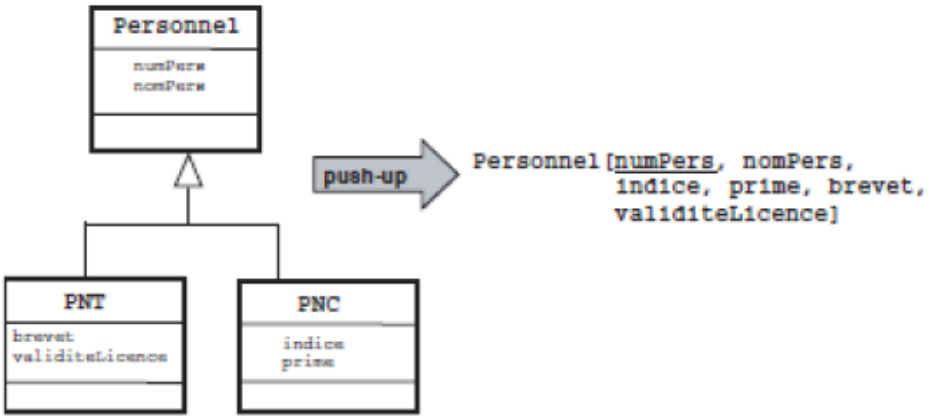

FIGURE 9 : Décomposition ascendante (push-up) d'une association d'héritage.

Remarque :

L'exemple « **FIGURE 9** » décrit une association d'héritage UML sans contrainte. En appliquant le principe de décomposition ascendante, nous obtenons une relation issue de la surclasse dans laquelle se trouve le contenu des relations issues des sous-classes.

Le langage UML nous apporte une aide à toutes les étapes de notre projet, comme il nous offert aussi de nombreux avantages pour l'analyse et la conception de notre système.

Résumé

Ce mémoire a pour principal objectif, le développement d'une application mobile pour permettre de louer tout type de produits existants. Pour cela, nous avons défini un ensemble de méthodes et de techniques permettant d'atteindre notre objectif. Pour concevoir notre application, nous avons utilisé UML comme langage de modélisation et « eXtreme programming » (méthodes agiles) comme démarche de développement.

Nous avons déployé et hébergé notre application, afin de la valider, et procéder à un ensemble de tests. Nous avons aussi évalué le fonctionnement de l'application pendant une longue période selon une étude statistique.

Mots clés : Location, Application mobile, UML2, démarche agile XP, NodeJs.

Abstract

The main objective of this thesis is the development of a mobile application to allow the rental of all types of existing products. Accordingly, we have defined a set of methods and techniques to achieve our goal. To design our application, we used UML as a modeling language and "eXtreme programming" (agile methods) as a developing process.

We deployed and hosted our application, in order to validate it, and carried out a set of tests. We, also, evaluated the operation of the application for a long time according to a statistical study.

Keywords: Rental, Mobile application, UML2, XP agile approach, NodeJs.