

CV

DEDICACE

J'ai le grand plaisir de dédier ce travail en témoignage d'affectation et de reconnaissance à tous ceux qui m'ont aidé à le réaliser :

- A mes chers parents qui m'ont tellement donné et à qui je dois tout, en signe de gratitude et de reconnaissance,
- A tous les membres de la famille, pour leurs encouragements, soutiens, affectations et confiances,
- A tous mes collègues qui m'ont donné du courage pour continuer les études,
- Aux enseignants et encadreurs de L'Ecole Nationale d'Informatique ainsi qu'à la société technologies et services qui m'ont donné confiance et espoir et qui m'ont soutenu durant tout le cursus universitaire de cette année.

Toutes ces valeurs m'ont donné confiance et espoir pour continuer les études.

REMERCIEMENT

Je tiens d'abord, avant tout, à remercier Dieu Tout Puissant et miséricordieux, qui m'a donné force, courage et santé pour que je puisse accomplir ces années d'études, et aussi ce mémoire.

Je tiens également remercier toutes les personnes qui m'ont aidé à réaliser ce travail et sans qui, ceci n'a pas pu être abouti. Mes plus vifs remerciements sont adressés à :

Monsieur RAMAMONJISOA Bertin Olivier, Professeur Titulaire, Directeur de l'Ecole Nationale d'Informatique qui nous a donné l'opportunité de faire des stages au sein des entreprises ;

Monsieur Pascal VILA, directeur de la société Netapsys MADAGASCAR, de m'avoir permis d'effectuer mon stage au sein de sa société.

Encadreur pedagogique

Chef de pole ra Dimby

Encdreur pro

Aux membres du jury d'avoir accepté d'examiner ce présent mémoire ;

A l'équipe de la société Netapsys Madagascar pour leur accueil chaleureux, leur sympathie, leurs conseils techniques précieux qui nous ont permis de bien travailler ensemble tout au long de ce stage ;

Tous les enseignants et au personnel de l'ENI, qui m'ont façonné et transmis leurs connaissances ;

A mes amis qui m'ont encouragé et apporté leur soutien moral.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

NOMENCLATURE

AJAX	: Asynchronous Javascript And XML
ASP	: Active Server Page
CRUD	: Create – Read – Update - Delete
CUR	: Centre Universitaire Régionale
CSS	: Cascading Style Sheet
CV	: Curriculum Vitae
DOC	: Word Document
DOCX	: XML Word Document
DSS	: Diagramme de Séquence System
DSC	: Diagramme de Séquence Conception
EE	: Entreprise Edition
ESPA	: Ecole Supérieure Polytechnique d’Antananarivo
ENI	: Ecole Nationale d’Informatique
GNU	: GNU’s Not Unix
GPL	: General Public Licence
HTML	: Hypertext Markup Language
HTTP	: Hypertext Transfer Protocol
IDE	: Integrated Development Environment
HTML	: HyperText Markup Language
JDBC	: Java Database Connectivity
JSON	: JavaScript Object Notation
JSP	: Java Server Page
LMD	: Licence – Master – Doctorat
MERISE	: Méthode d’Étude et de Réalisation Informatique par les Sous-ensembles
MVC	: Model View Controller
MVVM	: Model View View-Model
OMT	: Object Modeling Technique
ORDM	: Object Relational Mapping Diagrams
PC	: Portable Computer
PDF	: Portable Document File
PHP	: HypertextPreprocessor (acronyme récursif)
RAM	: Random Access Memory
RH	: Ressources Humaines
RG	: Règle de gestion
SGBD	: Système de Gestion de Base de Données
SGBDR	: Système de Gestion de Base de Données Relationnelles
SI	: Système d’Information

SR	: Systèmes et Réseaux
SQL	: Structured Query Language
SVN	: Subversion (tirée de l'application éponyme pour décrire les logiciels de gestion de version)
UL	: User Interface
UML	: Unified Modeling Language
VB	: Visual Basic
XHTML	: eXtensible Hypertext Markup Language
WWF	: World Wildlife Fund
XML	: eXtensible Markup Language

INTRODUCTION GENERALE

Actuellement, avec l'énorme avancement de la technologie, l'Informatique est désormais un domaine sur lequel tous les secteurs veulent se reposer pour trouver ses solutions en termes d'efficacité et de rendement. Le secteur de vente est notamment, celui qui est le plus concerné par cette évolution.

En effet, grâce au web, le commerce devient de plus en plus efficace en mettant facilement en relation le commerçant et ses consommateurs. Le marchand propose ses produits sur internet et d'un autre côté le consommateur aura une multitude de choix dans ses achats.

C'est dans ce contexte que s'inscrit notre projet de fin d'études qui s'intitule « Conception et réalisation d'une application web dynamique de site d'annonce ». Ce projet a été réalisé au sein de Netapsys Madagascar. Les technologies utilisées dans ce projet sont innovantes et surtout sont adaptées au mode de fonctionnement du site. Des Framework ont été nécessaires dans la réalisation dont les choix ont été faits en fonction des besoins des clients ainsi qu'à ce qui convient le plus au projet.

Ce mémoire propose, dans la première partie les présentations en faisant la présentation de l'ENI, la présentation de la société Netapsys Madagascar et la description du projet. Ensuite dans la deuxième partie, l'analyse et conception composé de l'analyse préalable, l'analyse conceptuelle et la conception détaillée. Et enfin dans la troisième partie, la réalisation tout en exposant la mise en place de l'environnement de développement et le développement de l'application.

PARTIE 1 : PRESENTATION

Chapitre 1 PRESENTATION DE L'ENI

Informations d'ordre général

L'Ecole Nationale d'Informatique, en abrégé ENI, est un établissement d'enseignement supérieur rattaché académiquement et administrativement à l'Université de Fianarantsoa.

Le siège de l'Ecole se trouve à Tanambao- Antaninarenina à Fianarantsoa.

L'adresse pour la prise de contact avec l'Ecole est la suivante :

- Ecole Nationale d'Informatique (ENI) Tanambao, Fianarantsoa.
- Le numéro de sa boîte postale est 1487 avec le code postal 301.
- Téléphone : 020 75 508 01.
- Son adresse électronique est la suivante : eni@univ-fianar.mg
- Site Web : [www. eni@univ-fianar.mg/eni](http://www.eni@univ-fianar.mg/eni)

Missions et historique

L'ENI se positionne sur l'échiquier socio-éducatif malgache comme étant le plus puissant secteur de diffusion et de vulgarisation des connaissances et des technologies informatiques.

Cette Ecole Supérieure peut être considérée aujourd'hui comme la vitrine et la pépinière des élites informaticiennes du pays.

L'Ecole s'est constituée de façon progressive au sein du Centre Universitaire Régional (CUR) de Fianarantsoa.

De façon formelle, l'ENI était constituée et créée au sein du (CUR) par le décret N° 83185 du 24 Mai 1983, comme étant le seul établissement Universitaire Professionnalisé au niveau national, destiné à former des techniciens et des Ingénieurs de haut niveau, aptes à répondre aux besoins et exigences d'Informatisation des entreprises, des sociétés et des organes implantés à Madagascar.

L'ENI a pour conséquent pour mission de former des spécialistes informaticiens compétents et opérationnels de différents niveaux notamment :

- En fournissant à des étudiants des connaissances de base en informatique ;
- En leur transmettant le savoir-faire requis, à travers la professionnalisation des formations dispensées et en essayant une meilleure adéquation des formations par rapport aux besoins évolutifs des sociétés et des entreprises.
- En initiant les étudiants aux activités de recherche dans les différents domaines des Technologies de l'information et de la communication (TIC).

L'implantation de cette Ecole Supérieure de technologie de pointe dans un pays en développement et dans une Province (ou Faritany) à tissu économique et industriel faiblement développé ne l'a pourtant pas défavorisée, ni empêchée de former des spécialistes informaticiens de bon niveau, qui sont recherchés par les entreprises, les sociétés et les organismes publics et privés sur le marché de l'emploi.

La filière de formation d'Analystes Programmeurs a été mise en place à l'Ecole en 1983, et a été gelée par la suite en 1996, tandis que la filière de formation d'ingénieurs a été ouverte à l'Ecole en 1986.

Dans le cadre du Programme de renforcement de l'Enseignement Supérieur (PRESUP), la filière de formation des Techniciens Supérieurs en Maintenance des Systèmes des Informatiques a été mise en place en 1986 grâce à l'appui matériel et financier de la Mission Française de coopération auprès de l'Ambassade de France à Madagascar.

Une formation pour l'obtention de la certification CCNA et / ou NETWORK + appelée « CISCO Networking Academy » a été créée à l'Ecole en 2002-2003 grâce au partenariat avec CISCO SYSTEM et l'Ecole Supérieure Polytechnique d'Antananarivo (ESPA). Cependant, cette formation n'avait pas duré longtemps.

Une formation de troisième cycle a été ouverte à l'Ecole depuis l'année 2003 – 2004 grâce à la coopération académique et scientifique entre l'Université de Fianarantsoa pour le compte de l'ENI et l'Université Paul Sabatier de Toulouse (UPST).

Cette filière avait pour objectif de former certains étudiants à la recherche dans les différents domaines de l'Informatique, et notamment pour préparer la relève des Enseignants-Chercheurs qui étaient en poste.

Pendant l'année 2007-2008, la formation en vue de l'obtention du diplôme de Licence Professionnelle en Informatique a été mise en place à l'ENI avec les deux options suivantes de formation :

Génie Logiciel et de Données.

Administration base des Systèmes et réseaux.

La mise en place à l'Ecole de ces deux options de formation devait répondre au besoin de basculement vers le système Licence – Master – Doctorat (LMD).

Mais la filière de formation des Techniciens Supérieurs en Maintenance des Systèmes Informatiques a été gelée en 2009.

En vue de surmonter les difficultés de limitation de l'effectif des étudiants accueillis à l'Ecole, notamment à cause du manque d'infrastructures, un système de « Formation Hybride » a été mis en place à partir de l'année 2010. Il s'agit en effet d'un système de formation semi-présentielle et à distance avec l'utilisation de la visioconférence pour la formation à distance.

Le système de formation hybride a été ainsi créé à Fianarantsoa ainsi qu'à l'Université de Toliara.

Organigramme institutionnel de l'ENI

Cet organigramme de l'Ecole est inspiré des dispositions du décret N° 83-185 du 23 Mai 1983.

L'ENI est administrée par un conseil d'Ecole, et dirigée par un directeur nommé par un décret adopté en conseil des Ministres.

Le Collège des enseignants regroupant tous les enseignants-chercheurs de l'Ecole est chargé de résoudre les problèmes liés à l'organisation pédagogique des enseignements ainsi qu'à l'élaboration des emplois du temps.

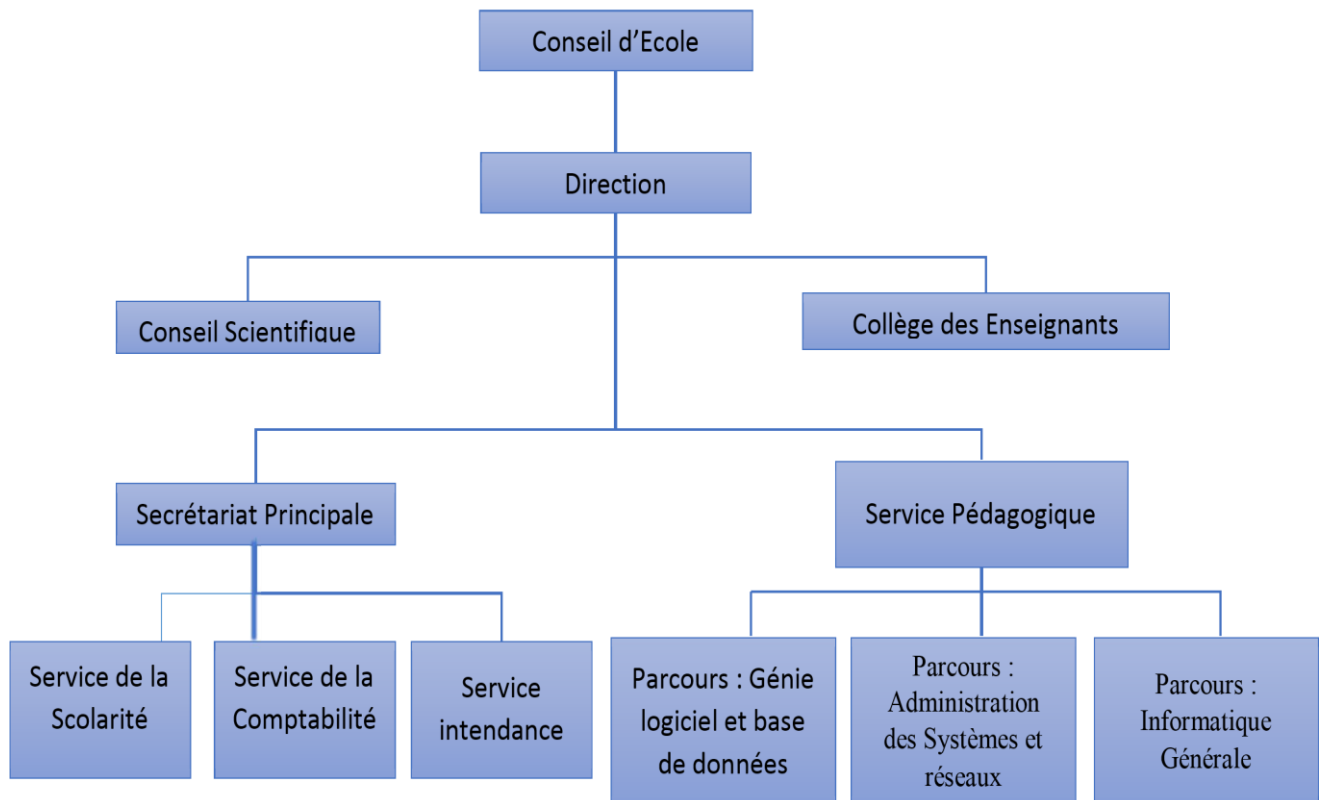
Le Conseil Scientifique propose les orientations pédagogiques et scientifiques de l'établissement, en tenant compte notamment de l'évolution du marché de travail et de l'adéquation des formations dispensées par rapport aux besoins des entreprises. Trois départements de formation caractérisent l'organigramme :

Le département de formation théorique à l'intérieur de l'Ecole ;

Le département de formation pratique pour la coordination et la supervision des stages en entreprise et des voyages d'études ;

Le département de formation doctorale pour l'organisation de la formation de 3ème cycle.

La figure 1 présente l'organigramme actuel de l'Ecole.



Sur cet organigramme, l'Ecole placée sous la tutelle académique et administrative de l'Université de Fianarantsoa, et dirigée par un Directeur élu par les Enseignants – Chercheurs permanents de l'Etablissement et nommé par un décret pris en Conseil des ministres pour un mandat de 3 ans.

Le Conseil de l'Ecole est l'organe délibérant de l'Ecole.

Le Collège des Enseignants propose et coordonne les programmes d'activités pédagogiques.

Le Conseil scientifique coordonne les programmes de recherche à mettre en œuvre à l'Ecole.

Le Secrétariat principal coordonne les activités des services administratifs (Scolarité, Comptabilité, et Intendance).

Conformément aux textes en vigueur régissant les Etablissements malgaches d'Enseignement Supérieur, qui sont barrés sur le système LMD, les Départements de Formation pédagogique ont été ainsi remplacés par des Mentions et des parcours. Et les chefs des Départements ont été ainsi remplacés par des responsables des mentions et les responsables des parcours.

Un administrateur des Réseaux et Systèmes gère le système d'information de l'Ecole et celui de l'Université.

Domaines de spécialisation

Les activités de formation et de recherche organisées à l'ENI portent sur les domaines suivants :

- Génie logiciel et Base de Données ;
- Administration des Systèmes et Réseaux ;
- Informatique Générale
- Modélisation informatique et mathématique des Systèmes complexes.

D'une manière plus générale, les programmes des formations sont basés sur l'informatique de gestion et sur l'informatique des Systèmes et Réseaux. Et les modules de formation intègrent aussi bien des éléments d'Informatique fondamentale que des éléments d'Informatique appliquée.

Le tableau 1 décrit l'organisation du système de formation pédagogique de l'Ecole.

Formation théorique	Formation pratique
Enseignement théorique Travaux dirigés Travaux pratiques	Etude de cas Travaux de réalisation Projets / Projets tutorés Voyage d'études Stages

Architecture des formations pédagogiques

Le recrutement des étudiants à l'ENI se fait uniquement par voie de concours d'envergure nationale en première année.

Les offres de formation organisées à l'Ecole ont été validées par la Commission Nationale d'Habilitation (CNH) auprès du Ministère de l'Enseignement Supérieur et de la Recherche Scientifique selon les dispositions de l'Arrêté N°31.174/2012-MENS en date du 05 Décembre 2012.

Au sein de l'ENI, il existe une seule mention (INFORMATIQUE) et trois parcours : Génie logiciel et Base de Données ; Administration des Systèmes et Réseaux ; Informatique Générale

L'architecture des études à trois niveaux conformément au système Licence-MasterDoctoral (LMD) permet les comparaisons et les équivalences académiques des diplômes au niveau international.

- L = Licence (Bac + 3) = L1, L2, L3 = 6 semestres S1 à S6

- M = Master (Bac + 5) = M1, M2 = 4 semestres S7 à S10

Le diplôme de licence est obtenu en 3 années des études après Baccalauréat. Et le diplôme de Master est obtenu en 2 ans après obtenu du diplôme de LICENCE.

Le MASTER PROFESSIONNEL est un diplôme destiné à la recherche emploi au terme des études.

Le MASTER RECHERCHE est un diplôme qui remplace l'ancien Diplôme d'Etudes Approfondies (DEA), et qui permet de s'inscrire directement dans une Ecole Doctorale.au terme des études.

- D = Doctorat (Bac +8)

Le Doctorat est un diplôme qu'on peut obtenir en 3 ans après l'obtention du diplôme de MASTER RECHERCHE.

Le tableau 2 présente l'architecture des études correspondant au système LMD.

+ 8	Doctorat		
+ 7			
+ 6		Master Recherche Master Professionnel	Marché du travail
+ 5	Master 2		
+ 4	Master 1		
+ 3	Licence 3	Licence Professionnelle	
+ 2	Licence 2	Diplôme de Technicien Supérieur	
+ 1	Licence 1	Brevet de Technicien Supérieur	

La licence peut avoir une vocation générale ou professionnelle.

Le master peut avoir une vocation professionnelle ou de recherche.

Le Tableau 3 représente la liste des formations existantes à l'ENI.

	FORMATION EN :
	LICENCE PROFESSIONNELLE ET MASTER HYBRIDE

Condition d'admission	Par voie de concours Formation professionnelle : 5 ans Formation hybride : 100 filières.	
Condition d'accès	Bac série C, D ou Technique	Licence Professionnelle.
Durée de formation	3 années	2 années
Diplôme délivrés	Diplôme de Licence Professionnelle en Informatique.	Diplôme de Master Professionnel ou de Master recherche.

L'accès en première année de MASTER se fait automatiquement pour les étudiants de l'Ecole qui ont obtenu le diplôme de Licence Professionnelle.

Le Master Recherche permet à son titulaire de poursuivre directement des études en doctorat et de s'inscrire directement dans une Ecole Doctorale.

Les Ecoles Doctorales jouissent d'une autonomie de gestion par rapport aux Etablissements de formation universitaire.

Il convient de signaler que par arrêté ministériel N° 21.626/2012 – MESupRES publié le 9 Août 2012 par la Commission National d'habilitation (CNH), l'Ecole Doctorale « Modélisation – Informatique » a été habilitée pour l'Université de Fianarantsoa.

Depuis l'année universitaire 2010-2011, l'ENI s'est mise à organiser des formations hybrides en informatique dans les différentes régions (Fianarantsoa, Toliara) en raison de l'insuffisance de la capacité d'accueil des infrastructures logistiques. En effet, le système de formation hybride semi - présentielle utilise la visioconférence pour la formation à distance.

Bien qu'il n'existe pas encore au niveau international de reconnaissance écrite et formelle des diplômes délivrés par l'ENI, les étudiants diplômés de l'Ecole sont plutôt bien accueillis dans les instituts universitaires étrangères (CANADA, Suisse, France...)

Relations de l'ENI avec les entreprises et les Organismes

Les stages effectués chaque année par les étudiants mettent l'Ecole en rapport permanent avec plus de 300 entreprises et organismes publics, semi-publics et privés, nationaux et internationaux.

L'Ecole dispose ainsi d'un réseau d'entreprises, de sociétés et d'organismes publics et privés qui sont des partenaires par l'accueil en stage de ses étudiants, et éventuellement pour le recrutement après l'obtention des diplômes par ces derniers.

Les compétences que l'Ecole cherche à développer chez ses étudiants sont l'adaptabilité, le sens de la responsabilité, du travail en équipe, le goût de l'expérimentation et l'innovation.

En effet, la vocation de l'ENI est de former des techniciens supérieurs de niveau LICENCE et des ingénieurs de type généraliste de niveau MASTER avec des qualités

scientifiques, techniques et humaines reconnues, capables d'évoluer professionnellement dans des secteurs d'activité variés intégrant l'informatique.

Les stages en milieu professionnel permettent de favoriser une meilleure adéquation entre les formations à l'Ecole et les besoins évolutifs du marché de l'emploi.

Les principaux débouchés professionnels des diplômés de l'Ecole concernent les domaines suivants :

- L'informatique de gestion d'entreprise
- Les technologies de l'information et de la communication (TIC)
- La sécurité informatique des réseaux
- L'administration des réseaux et des systèmes
- Les services bancaires et financiers, notamment le Mobile Banking
- Les télécommunications et la téléphonie mobile
- Les Big Data
- Le commerce, la vente et l'achat, le Marketing
- L'ingénierie informatique appliquée
- L'écologie et le développement durable

Parmi les sociétés, entreprises et organismes partenaires de l'Ecole, on peut citer :

ACCENTURE Mauritius, Air Madagascar, Ambre Associates, Airtel, Agence Universitaire de la Francophonie (AUF) , B2B, Banque Centrale, BFV-SG, BIANCO, BLUELINE, Bureau national de gestion des Risques et des catastrophes (BNGRC), CEDII-Fianarantsoa, Data Consulting, Central Test, Centre National Antiacridien, CNRE, CHU, CNRIT, COLAS, Direction Générale des Douanes, DLC, DTS/Moov, FID, FTM, GNOSYS, IBONIA, INGENOSIA, INSTAT, IOGA, JIRAMA, JOUVE, MADADEV, MAEP, MEF, MEN, MESupRES, MFB, MIC, MNINTER, Min des postes/Télécommunications et du Développement Numérique, NEOV MAD, Ny Havana, Madagascar National Parks, OMNITEC, ORANGE, OTME, PRACCESS, QMM Fort-Dauphin, SMMC, SNEDADRS Antsirabe, Sénat, Société d'Exploitation du Port de Toamasina (SEPT), SOFTWELL, Strategy Consulting, TELMA, VIVETEC, Société LAZAN'I BETSILEO, WWF ...

L'organisation de stage en entreprise continue non seulement à renforcer la professionnalisation des formations dispensées, mais elle continue surtout à accroître de façon exceptionnelle les opportunités d'embauche pour les diplômés de l'Ecole.

Partenariat au niveau International

Entre 1996 et 1999, l'ENI avait bénéficié de l'assistance technique et financière de la Mission Française de Coopération et d'action culturelle dans le cadre du Programme de Renforcement de l'Enseignement Supérieur (PRESUP) consacré à l'Ecole a notamment porté sur :

- Une dotation en logiciels, micro-ordinateurs, équipements de laboratoire de maintenance et de matériels didactiques
- La réactualisation des programmes de formation assortie du renouvellement du fonds de la bibliothèque
- L'appui à la formation des formateurs
- L'affectation à l'Ecole d'Assistants techniques français

De 2000 à 2004, l'ENI avait fait partie des membres du bureau de la Conférence Internationale des Ecoles de formation d'Ingénieurs et Technicien d'Expression Française (CITEF).

Les Enseignants-Chercheurs de l'Ecole participent régulièrement aux activités organisées dans le cadre du Colloque Africain sur la Recherche en Informatique (CARI).

L'ENI avait également signé un accord de coopération inter-universitaire avec l'Institut de Recherche en Mathématiques et Informatique Appliquées (IREMIA) de l'Université de la Réunion, l'Université de Rennes 1, l'INSA de Rennes, l'Institut National Polytechnique de Grenoble (INPG).

A partir du mois de Juillet 2001, l'ENI avait abrité le Centre de Réseau Opérationnel (Network Operating Center) du point d'accès à Internet de l'Ecole ainsi que de l'Université de Fianarantsoa. Grâce à ce projet américain qui a été financé par l'USAID Madagascar, l'ENI de l'Université de Fianarantsoa avait été dotées d'une ligne spécialisée d'accès permanent au réseau Internet.

L'ENI avait de même noué des relations de coopération avec l'Institut de Recherche pour le Développement (IRD).

L'objet du projet de coopération avait porté sur la modélisation environnementale du Corridor forestier de Fandriana jusqu'à Vondrozo (COFAV). Dans ce cadre, un atelier scientifique international avait été organisé à l'ENI en Septembre 2008. Cet atelier scientifique avait eu pour thème de modélisation des paysages.

Et dans le cadre du programme scientifique PARRUR, l'IRD avait financé depuis 2010 le projet intitulé « Forêts, Parcs et Pauvreté dans le Sud de Madagascar (FPPSM). Des étudiants en DEA et des Doctorants issus de l'ENI avaient participé à ce Programme.

Par ailleurs, depuis toujours la même année 2010, l'ENI de Fianarantsoa avait été sélectionnée pour faire partie des organismes partenaires de l'Université de Savoie dans le cadre du projet TICEVAL relatif à la certification des compétences en TIC ;

Le projet TICEVAL avait été financé par le Fonds Francophone des Inforoutes pour la période allant de 2010 à 2012, et il avait eu pour objectif de généraliser la certification des compétences en Informatique et Internet du type C2i2e et C2imi.

Dans le cadre du projet TICEVAL, une convention de coopération avec l'Université de Savoie avait été signée par les deux parties concernées. La mise en œuvre de la Convention de Coopération avait permis d'envoyer des étudiants de l'ENI à Chambéry pour poursuivre des études supérieures en Informatique.

Enfin et non des moindres, l'ENI avait signé en Septembre 2009 un protocole de collaboration scientifique avec l'ESIROI – STIM de l'Université de la Réunion.

Comme l'ENI constitue une pépinière incubatrice de technologie de pointe, d'emplois et d'entreprises, elle peut très bien servir d'instrument efficace pour renforcer la croissance économique du pays, et pour lutter contre la Pauvreté.

De même que le statut de l'Ecole devrait permettre de renforcer la position concurrentielle de la Grande Ile sur l'orbite de la modélisation grâce au développement des nouvelles technologies.

Débouchés professionnels des Diplômes

Le chômage des jeunes diplômés universitaires fait partie des maux qui gangrènent Madagascar. L'environnement socio-politique du pays depuis 2008 jusqu'à ce jour a fait que le chômage des diplômés est devenu massif par rapport aux établissements de formation supérieure existants.

Cependant, les formations proposées par l'Ecole permettent aux diplômés d'être immédiatement opérationnels sur le marché du travail avec la connaissance d'un métier complet lié à l'informatique aux TIC.

L'Ecole apporte à ses étudiants un savoir-faire et un savoir-être qui les accompagnent tout au long de leur vie professionnelle. Elle a une vocation professionnalisante.

Les diplômés en LICENCE et en MASTER issus de l'ENI peuvent faire carrière dans différents secteurs.

L'Ecole bénéficie aujourd'hui de 34 années d'expériences pédagogiques et de reconnaissance auprès des sociétés, des entreprises et des organismes. C'est une Ecole Supérieure de référence en matière informatique.

Par conséquent, en raison de fait que l'équipe pédagogique de l'Ecole est expérimentée, les enseignants-chercheurs et les autres formateurs de l'Ecole sont dotés d'une grande expérience dans l'enseignement et dans le milieu professionnel.

L'Ecole est fière de collaborer de façon régulière avec un nombre croissant d'entreprises, de sociétés et d'organismes publics et privés à travers les stages des étudiants. Les formations dispensées à l'Ecole sont ainsi orientées vers le besoin et les attentes des entreprises et des sociétés.

L'Ecole fournit à ses étudiants de niveau LICENCE et MASTER des compétences professionnelles et métiers indispensables pour les intégrer sur le marché du travail.

L'Ecole s'efforce de proposer à ses étudiants une double compétence à la fois technologique et managériale combinant l'informatique de gestion ainsi que l'administration des réseaux et systèmes.

D'une manière générale, les diplômés de l'ENI n'éprouvent pas de difficultés particulières à être recrutés au terme de leurs études. Cependant, l'ENI recommande à ses diplômés de promouvoir l'entrepreneuriat en TIC et de créer des cybercafés, des SSII ou des bureaux d'études.

Le tableau 4 représente les débouchés professionnels éventuels des diplômés que l'ENI délivre.

LICENCE	-	-	Analyste
	-	-	Programmeur
	-	-	Administrateur de site web/de portail web
	-	-	Assistant Informatique et internet
	-	-	Chef de projet web ou multimédia
	-	-	Développeur Informatique ou multimédia
	-	-	Intégrateur web ou web designer
	-	-	Hot liner/Hébergeur Internet
	-	-	Agent de référencement
	-	-	Technicien/Supérieur de help desk sur Informatique

	<ul style="list-style-type: none"> – Responsable de sécurité web – Administrateur de réseau – Administrateur de cybercafé 	
MASTER	<ul style="list-style-type: none"> – Administrateur de cybercafé – Administrateur de réseau et système – Architecture de système d'information – Développeur d'application /web /java/Python/ /Android – Ingénieur réseau – Webmaster /web designer – Concepteur Réalisateur d'applications – Directeur du système de formation – Directeur de projet informatique – Chef de projet informatique – Responsable de sécurité informatique – Consultant fonctionnel ou freelance – Administrateur de cybercafé 	IOS

Ressources humaines

Directeur de l'Ecole : Professeur RAMAMONJISOA Bertin Olivier
 Responsable de Mention : Docteur MAHATODY Thomas
 Responsable de Parcours « Génie Logiciel et Base de Données » Docteur RATIARSON Venot
 Responsable de Parcours « Administration Monsieur SIAKA
 Systèmes et Réseaux »
 Responsable de Parcours « Informatique Générale » Docteur RAKOTOASIMBAHOAKA Cyprien Robert

Nombre d'Enseignants permanents :	13 dont deux (02) Professeurs Titulaires, six (06) Maîtres de Conférences et cinq (05) Assistants d'Enseignement Supérieur et de Recherche
Nombre d'Enseignants vacataires :	10
Personnel Administratif :	23

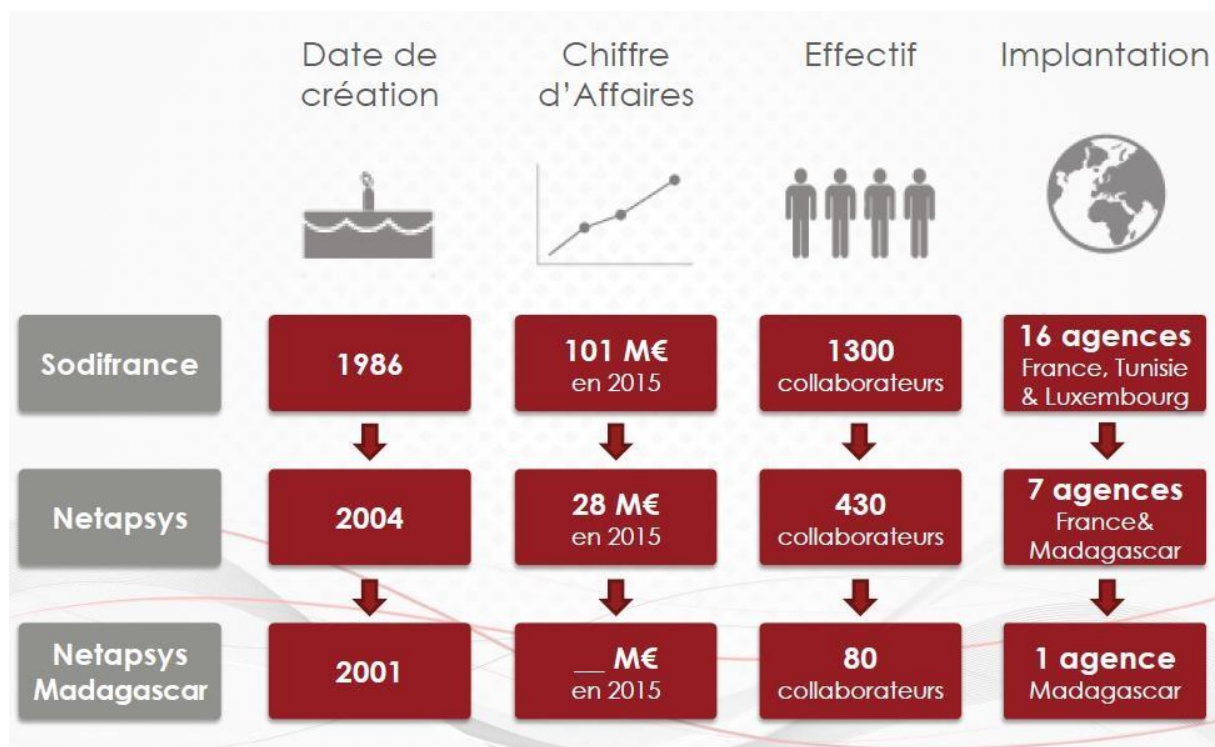
Chapitre 2 PRESENTATION DE LA SOCIETE NETAPSYS Madagascar

2.1 Présentation du Groupe Netapsys :

Netapsys est une société d'ingénierie informatique spécialiste des nouvelles technologies, des solutions de BI (Business Intelligence), de Gestion de contenu, Collaboratives, d'E-business, d'E-commerce et de Mobilité.

Netapsys conçoit, développe et maintient des applications informatiques et des systèmes d'information sur-mesure, basés sur des technologies de pointe.

Ses fondateurs, Yoann Hébert et Jérémy Rousselle, experts en technologies objet et en systèmes d'information, dirigent la société depuis sa création. Quelques chiffres de croissance sont représentés dans la figure 2



Netapsys qui est une société spécialiste de la transformation digitale (applications et SI sur-mesure) a rejoint le Groupe Sodifrance en 2015, ce dernier étant spécialiste de la transformation et modernisation des systèmes mainframe ; créant ainsi la complémentarité et la synergie des 2 marques. Netapsys est une Entreprise de Services en Numérique spécialisée dans :

- Les nouvelles technologies : .NET, Java JEE, LAMP, ... essentiellement issues du « monde internet »

- La conduite d'opérations en mode projet : engagement forfaitaire sur les délais, les livrables, les coûts, ...
- Le maintien en conditions opérationnelles : Tierce Maintenance Applicative(TMA), Infogérance et accompagnement technique.

2.2. **Netapsys Madagascar :**

Netapsys Madagascar, basée à Antananarivo, Porte B201 Immeuble ARO Ampefiloha, est spécialisée dans la conception, le développement et la maintenance de projets digitaux, applicatifs et mobiles en technologie Web Open Source LAMP et JAVA, en méthodologies agiles.

Dirigée par Pascal Vila et forte de 90 collaborateurs, Netapsys Madagascar occupe aujourd'hui une position de premier plan sur les métiers de l'ingénierie logicielle dans l'Océan Indien. Elle intervient pour des clients locaux, dans le domaine des télécommunications, de la finance, dans l'administration ou encore l'agroalimentaire mais également sur des projets français, en direct ou avec les agences françaises du groupe.

Elle est aujourd'hui constituée de 6 pôles à savoir : pôle PHP, pôle CMS, pôle Studio, Pôle java, pôle admin et le pôle Infrastructure ; basés sur :

- Formation

Netapsys a mis en place un programme de formation complet pour aider ses collaborateurs à s'adapter aux nouveaux défis technologiques. Les formations sont basées sur des workshops, MOOC, de la recherche, technical breakfast, ...

- Certification

Netapsys propose à ses ingénieurs de passer différents niveaux de certification. Ainsi ses ingénieurs peuvent évoluer et devenir des experts dans leur domaine.

- Autonomie et Responsabilités

Netapsys encourage chez ses collaborateurs les qualités d'autonomie, de prise d'initiatives afin d'augmenter l'agilité de l'entreprise.

- Carrière

Les collaborateurs ont des perspectives d'évolution géographique et professionnelle intéressantes, nombre d'entre eux ont pu évoluer avec le temps en passant de « Ingénieur d'Etudes et de Développement » à « Chef de Projet » et « Directeur de Pôle » par exemple.

- Innovation

Netapsys est à la pointe de la technologie. Ils se forment continuellement pour toujours rester au top, apprendre des nouvelles technologies pour satisfaire tous les besoins des clients.

Etre dynamiques : Devfest, DevoXx, Atlassian Tour, livres blancs, blog technique, séminaires...

- Multiculturel

Au sein de l'Agence Netapsys Madagascar, privilège d'une double culture francomalgache. Un échange dans les deux sens de visions professionnelles au départ différentes pour créer une vision commune où chacun apprend de l'autre.

- Solidarité & Convivialité

Mettre un point d'honneur à travailler dans une bonne ambiance, à s'aider les uns les autres. À mettre en place des activités pour souder les équipes.

- Sport

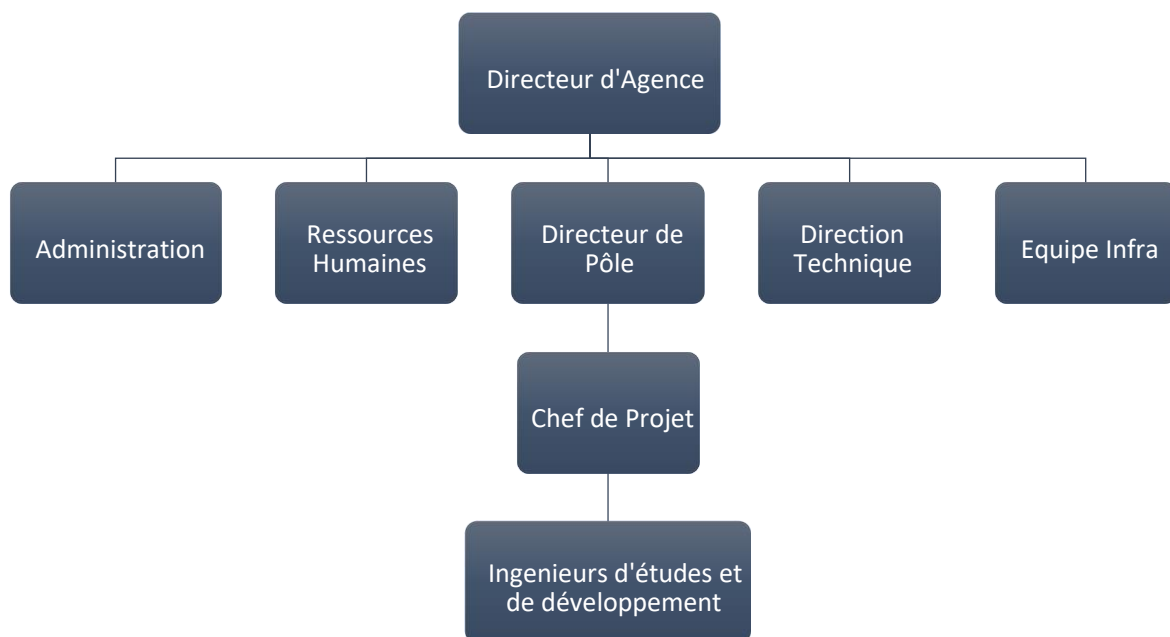
Netapsys participe à plusieurs tournois sportifs. Mais aussi, toujours dans cet esprit de convivialité, permet à ses collaborateurs de jouer au futsal tous les jeudis. Les mardis et vendredis c'est pétanque

- Bien vivre

Afin que les collaborateurs se sentent bien dans l'entreprise, ils ont droit bien évidemment à la CNAPS et l'OSTIE, mais aussi à une prime quotidienne de panier repas. Les horaires sont flexibles selon les préférences des collaborateurs.

2.3. Organigramme :

La figure 3 présente l'organigramme de Netapsys Madagascar.



2.4. Les valeurs du Groupe :

- Culture d'engagement

Bâtie sur deux principes fondamentaux, l'expertise technologique et la maîtrise de la conduite d'opérations en mode projet, Netapsys se démarque par sa capacité à s'engager auprès de ses clients.

- Engagement à proposer la solution (fonctionnelle, technique et organisationnelle) la plus adaptée au contexte du projet et à la stratégie du client.
- Engagement clair sur les résultats : exhaustivité des livrables ; qualité des travaux (en accord avec les normes du client, l'état de l'art du marché et la politique interne de Netapsys d'industrialisation des développements) et respect des délais.

- Démarche d'amélioration continue

Le partage des connaissances et le développement des compétences techniques sont au cœur de la culture Netapsys. L'ambition, portée par l'ensemble des équipes, est de devenir une référence française sur les nouvelles technologies. Pour atteindre cet objectif, Netapsys s'inscrit dans une démarche d'amélioration continue et s'appuie sur :

- Une montée en compétence continue de ses équipes (formation continue, technical breakfast, politique de certification ...)
- Un développement de son outillage logiciel (intégration continue, mesure permanente de la qualité des développements, gestion de projet, documentation...)

- Des méthodes de travail inspirées des méthodologies agiles
- Solidarité, convivialité, développement durable

Netapsys recherche une performance qui n'est pas uniquement financière, mais aussi humaine. Cela se traduit par la mise en place de pratiques fondées sur des valeurs éthiques et une politique RH visant à mettre en adéquation les envies personnelles et les projets de la société par :

- La dynamique collective, qui se traduit par le partage des résultats (accord d'intéressement ou participation), la mise en place de standup meeting permettant de favoriser l'échange et la communication ainsi que l'animation de technical breakfasts,
2 à 3 fois par semaine, durant lesquels un collaborateur présente à l'ensemble de la société un sujet technique.
- L'attention particulière accordée à la convivialité (budgets dédiés à la vie des pôles, séminaires, soirées, sponsoring d'évènements sportifs type marathons relais ou ludiques). L'équipe féminine Netapsys a participé en 2012 à La parisienne, une course pour soutenir la lutte contre le cancer du sein.

2.5. Technologies et outils maîtrisés

Pour réaliser des projets conformes aux attentes des clients, la société propose à ses clients les technologies et les outils de dernière génération.

2.5.1.. Industrialisation des savoir-faire

- Utilisation de Framework ;
- Mutualisation des développements ;
- Partage des méthodes et bonnes pratiques ;
- Normalisation des recettes ;
- Documents génériques de spécifications ;
- Méthodes : UML, Merise, AGILES.

2.5.2. Partage et diffusion des connaissances


- Intranet collaboratif de partage des méthodes, process et documentations techniques
- Échange et partage de liens et sources d'information ;
- Recensement exhaustif des modules et librairies utilisés ;
- Formations continues : 100 heures de formation dispensées chaque année.

2.5.3. Outils de pilotage projets




- TinyPM : une plateforme collaborative de gestion de projets qui permet de centraliser l'ensemble des informations relatives à un projet et permet un suivi en temps réel de l'avancement du projet.
- Une Plateforme de gestion des recettes projet : Netapsys a développé sous technologie Flex son propre outil de recettes, afin de coordonner au mieux cette phase cruciale avant livraison ;
- JIRA : un système de suivi de bugs, un système de gestion des incidents, et un système de gestion de projets, gestions des temps consacrés pour chaque sous tâche développé par Atlassian Software Systems ;
- Projector : un système de gestion de projet, permettant d'imputer le temps dans un projet
- Mantis Bug Tracker : plateforme de gestion d'événements Open Source.
- GIT : Netapsys utilise cet outil de gestionnaire des versions (ou gestionnaire des sources) pour assurer le développement en équipe.

2.6. Références clients :

La Société a une clientèle prestigieuse tant sur le marché local qu'à l'étranger. Le tableau 5 présente une liste (non exhaustive) de cette clientèle avec les projets réalisés et leurs domaines :

Clients	Projet	Domaine
 France	Refonte du réseau social de L'entreprise	TRANSPORT

 <p>France</p>  <p>La Réunion</p>	<p>Gestion demande de crédits Outil de contrôle des risques</p> <p>Suivi des commerciaux Aide à la vente de crédit/épargne App mobile classement des agences</p>	<p>BANQUE</p>
 <p>France</p>	<p>Mise en place du site team Europcar</p>	<p>SPORT</p>
  	<p>Site orange.mg Service d'actualités Vidéo streaming</p> <p>Interconnexion du SI de Telma et de Blueline pour vendre de l'Airtime</p> <p>Mvola Gestion de clients Application de covoiturage Création du SI Comores</p>	<p>TELE-OPERATEURS</p>
 <p>France</p>	<p>Site d'émissions (ONPQDC, nouvelles écritures)</p>	<p>MEDIA</p>

 <p>France</p>  <p>Madagascar</p>	<p>App de génération des notices de présentation pour tous les salons automobiles du monde</p> <p>App tablette fête de la bière Gestion documentaire pour la norme Iso 9001</p>	<p>INDUSTRIE</p>
 <p>France</p>	<p>Outils de production : Vente et publication de rapports de notation sur le marché de l'informatique</p>	<p>SERVICE</p>

Chapitre 3 DESCRIPTION DU PROJET

3.1. Formulation :

Le projet sur lequel on a travaillé consiste en un site d'annonce. C'est une vente en ligne inter-entreprises ou B2B puisqu'il s'agit d'un marché entre une société de fournisseur et d'un grossiste.

Ce site a été développé avec le langage de programmation javascript et utilise la technologie MEAN c'est à dire Angular 4 comme le front-end de l'application, au niveau du back office il y a le nodeJS comme serveur, ExpressJS qui est le framework de NodeJS pour gérer les routes et pour finir MongoDB pour le base de données.

3.2. Objectifs :

Comme on le sait déjà, le projet consiste en un site d'annonce. On a pour but d'implémenté d'autres fonctionnalités qui ont pour objectif d'améliorer le système de vente sur la plateforme, de rendre encore plus facile la relation vendeur et client et surtout, rendre l'utilisation du site plus efficace, aisé et attirant encore plus les utilisateurs.

3.3. Besoins des utilisateurs :

.3.1. Besoins fonctionnels :

Les besoins fonctionnels du site sont les suivantes :

- Permettre aux utilisateurs de se connecter.
- Permettre la création de compte client.
- Permettre aux clients la gestion du Blog (de poster, d'éditer, de supprimer).
- Permettre aux clients de commenter un blog et aussi de mettre un avis sur une blog (mettre un pouce vert ou rouge).
- Permettre aux clients la gestion d'annonce (de poster, d'éditer, de supprimer).
- Permettre aux clients de se communiquer entre eux (chat en groupe).
- Permettre aux clients de mettre en favoris une annonce publié.
- Permettre aux clients de choisir la langue du site (un site multilingue).
- Offrir aux administrateurs du site la possibilité de gérer les comptes clients et gérer les commandes.
- Permettre aux administrateurs de gérer les fonctionnalités du système, c'est-à-dire activer ou désactiver telle ou telle fonctionnalité, paramétrer le site.

.3.2. Besoins non fonctionnels :

Il s'agit des besoins qui caractérisent le système, concernant la performance mais surtout des contraintes d'implémentation.

En effet l'application doit être :

- Extensible, c'est-à-dire toujours réutilisable permettant l'évolution des fonctionnalités
- Sécurisé
- Rapide et fiable dans l'exécution
- Ergonomique et conviviale

3.4. Moyens nécessaires à la réalisation du projet :

Pour l'aboutissement du projet, des moyens en personnel sont nécessaire autant qu'en matériel et logiciel.

a. Moyens en personnel

Le chef de projet

Un développeur

b. Organisation actuelle

Pour la réalisation de ce projet, chaque entité tient un rôle très important et respecte une certaine organisation.

- **Chef de projet**

Cette entité assure de discuter avec le stagiaire des grandes lignes à entreprendre pour le projet, de le guider et conseiller, mais aussi de faire des vérifications et de mise en production des mises a jours.

- **Développeur**

Pendant la concrétisation de l'application, le développeur stagiaire doit réaliser les spécifications et assurer la mise en pré-production.

Leurs tâches sont réparties comme suit

Manier l'interface de l'application qui consiste à rendre ergonomique ses écrans et à réaliser les maquettes de design.

Développer le moteur de l'application, en d'autres termes : créer les fonctionnalités de tous les écrans,

Rendre dynamiques les pages de l'application qui consiste à minimiser les rechargements lors des traitements des formulaires,

Concevoir la structure de l'application ou établir l'architecture.

3.5. Matériels et logiciels

a. Ordinateur :

L'environnement de développement exige l'utilisation de plusieurs logiciels et ordinateur puissants pour pouvoir ensuite soutenir une production optimale. Il est aussi indispensable que la machine dispose d'un accès internet. Ainsi, un (01) ordinateur pour le développement.

	PC
Système d'exploitation	Windows 10
Processeur	Intel core i3 2,5Ghz
Ram	DDR3 8Gb
Disque Dur	500 Gb

b. Logiciel :

En ce qui concerne, les logiciels et outils, le projet a aussi besoin de :

- Visual Studio Code, un environnement de développement intégré ou IDE pour l'édition des codes.
- Un navigateur web pour l'accès au site.
- Visual paradigm, qui est un outil de modélisation utile pour la conception,
- MongoDB qui est un système de Gestion de Base de Données.
- Nodejs un serveur d'application web.
- Postman pour tester rapidement les requêtes Http.

- Robomongo qui est une interface utilisateur pour gérer les bases de données, collections et documents de mongoDB.
- Et Git qui est un système de gestion de version de l'application.

3.6. Résultats attendus :

Le résultat attendu dans l'élaboration de ce projet n'est autre que d'avoir une application qui répond au besoin du client. Et avoir un meilleur support de documentation pour l'application pour assister les collaborateurs dans leur tâche.

PARTIE 2 : ANALYSE ET CONCEPTION DU PROJET

Chapitre 4 ANALYSE PREALABLE

▪ Analyse de l'existant

Actuellement, il n'y a aucun précédent qui pourrait être recueilli et actuellement, il n'y a aucun précédent qui pourrait être analysé comme ressource existante du projet.

..1. Organisation actuelle

Avant la conception, Il n'y avait pas encore d'interface pour gérer la gestion d'annonce du site web dynamique

..2. Critique de l'existant

Si nous ne faisons recours qu'aux méthodes simples, c'est-à-dire en utilisant les outils de Microsoft Office Suite comme Excel et Access, le traitement se fera difficile. Nous avons une grande quantité de demandes à traiter et sa quantité augmente considérablement et rapidement.

Il sera plus difficile encore de gérer ces informations à main, c'est-à-dire avec des papiers comme à la méthode traditionnelle.

En tout, les solutions en main ne répondent pas totalement aux besoins cités précédemment.

▪ Conception avant-projet

..1. Scenarios et esquisses de solutions

Pour pallier à ces lacunes afin d'atteindre les objectifs et les besoins de l'utilisateur, nous avons faits une ébauche de solutions avec leurs avantages et inconvénients respectives.

- Scenario 1 : Développer une application web avec des frameworks.
- Scenario 2 : Acheter un logiciel CMS de vente en ligne, qui serait déjà complet en termes de fonctionnalités de vente en ligne.

a. Acheter un logiciel de gestion des licences

L'avantage d'acheter un logiciel c'est d'avoir déjà un logiciel prêt à être utilisé, offre plusieurs fonctionnalités, ne pas perdre du temps dans une conception ou réalisation d'un logiciel

Les logiciels achetés offrent beaucoup des fonctionnalités mais ne correspondent pas aux besoins de l'utilisateur.

b. Application web avec un Framework

Les Framework sont des ensembles de fonctionnalités organisées sous forme de bibliothèques. Ces bibliothèques appelées aussi librairies permettent aux développeurs de profiter de fonctions déjà créées afin de les imbriquer les unes aux autres et ainsi de former une application complexe. Les Framework apportent une grande flexibilité et permettent aussi de gagner du temps de développement.

Les Framework seront très utiles lors de création d'applications complexes nécessitant une vraie adaptation à une activité, et aussi de développer des fonctionnalités selon le besoin de l'utilisateur

..2. Solution retenue et justification :

Les solutions présentées ci-dessus ont ses avantages et ses inconvénients et chacun convient à des types de projets différents.

Néanmoins, on a jugé que le choix du *de création d'un logiciel* le plus adapté à notre problème et à l'avancement du projet.

▪ Choix des méthodes et technologies utilisées

..1. Méthode de modélisation

Nombreux sont les méthodes de modélisation actuellement. Mais on distingue deux méthodes qui sont plus utilisés et recommandé par les standards de développement de logiciel : Merise ou Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise, et la notation UML ou Unified Modeling Language (langage de modélisation unifiée).

❖ **Merise 2**

MERISE est élaborée en France en 1978, permet de concevoir un système d'information d'une façon standard et méthodique.

La méthode MERISE est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles (conceptuels, logiques & organisationnels et physiques). La séparation des données et des traitements assure une longévité au modèle. En effet, l'agencement des données n'a pas à être souvent remanié, tandis que les traitements le sont plus fréquemment.

La méthode Merise d'analyse et de conception propose une démarche articulée simultanément selon 3 axes pour hiérarchiser les préoccupations et les questions auxquelles répondre lors de la conduite d'un projet :

- Cycle de vie : phases de conception, de réalisation, de maintenance puis nouveau cycle de projet.
- Cycle de décision : des grands choix, la définition du projet (étude détaillée) jusqu'aux petites décisions des détails de la réalisation et de la mise en œuvre du système d'information. Chaque étape est documentée et marquée par une prise de décision.
- Cycle d'abstraction : niveaux conceptuels, organisationnel, logique et physique/opérationnel (du plus abstrait au plus concret) L'objectif du cycle d'abstraction est de prendre d'abord les grandes décisions métier, pour les principales activités (Conceptuel) sans rentrer dans le détail de questions d'ordre organisationnel ou technique.

La méthode Merise, très analytique, distingue nettement les données et les traitements, même si les interactions entre les deux sont profondes et s'enrichissent mutuellement (validation des données par les traitements et réciproquement). Certains auteurs (Merise/méga, puis Merise/2) ont également apporté la notion complémentaire de communications, vues au sens des messages échangés. Aujourd'hui, avec les SGBD-R, l'objet, les notions de données et de traitements sont de plus en plus imbriquées.

❖ **Notation UML**

Unified Modeling Language (UML) est un langage de modélisation graphique qui permet de mettre en évidence l'architecture ainsi que les détails de la conception d'un projet.

Notre choix s'est rivié sur UML puisqu'il est le plus célèbre et le plus satisfaisant outil de son type. On peut voir clairement chaque étape d'avancement et d'exécution du projet puisque UML nous offre différents points de vue du modèle.

UML est plus expressif, plus propre et plus uniforme que toutes les autres méthodes. Elle se rapproche des langages de programmation et peuvent bien définir l'architecture du projet en intègre le concept objet tandis que MERISE ne prend pas en charge la partie programmation.

..2. **Méthode de gestion de projet**

❖ **Méthode traditionnelle**

Depuis toujours, les projets sont gérés avec la méthode dite « classique » qui se caractérise par recueillir les besoins, définir le produit, le développer et le tester avant de le livrer. On parle alors ici d'une approche prédictive « cycle en cascade » comme illustré sur la figure 3.

Comme son nom l'indique, il s'agit ici de prévoir des phases séquentielles où il faut valider l'étape précédente pour passer à la suivante. Le chef de projet doit alors s'engager sur un planning précis de réalisation du projet en prévoyant des jalons de débuts et fins de phases ainsi que les tâches à effectuer.

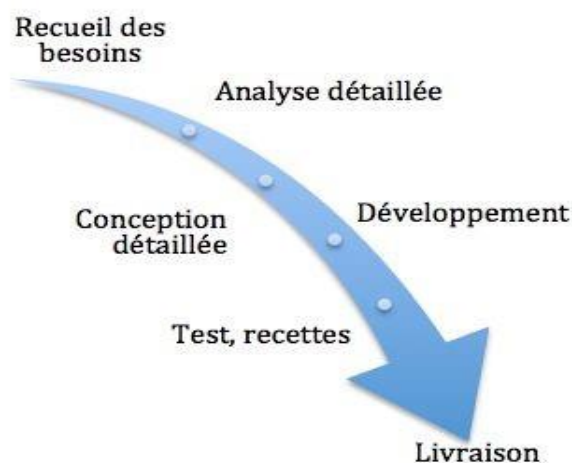


Figure 3. Cycle en cascade

Il faut tout faire bien du premier coup car elle ne peut pas permettre de retours en arrière. Une décision ou un problème rencontré dans une phase peuvent remettre en cause partiellement ou totalement les phases précédentes validées.

Dans un cycle « en cascade » les risques sont détectés tardivement puisqu'il faut attendre la fin du développement pour effectuer la phase de test. Plus le projet avance, plus l'impact des risques augmente : il sera toujours plus difficile et coûteux de revenir en arrière lorsqu'on découvre une anomalie tardivement.

Afin d'anticiper au mieux ces risques il est nécessaire de produire des documents très détaillés en amont (recueil des besoins, cahier des charges, zoning, wireframe etc...) qui seront validés par le client. Néanmoins, ces documents restent

théoriques et conceptuels jusqu'à ce que le dispositif soit testé dans des conditions réelles ; le client validera le contenu papier (conception, maquette, développement fonctionnalités etc...) mais sera toujours plus sensible à ce qu'il verra sur son écran.

□ **Méthodes agiles**

Le mouvement des méthodes agiles a commencé en 2001 aux Etats-Unis. Dix-sept experts en développement logiciel se sont réunis afin de mettre au point ces méthodes suite à un taux d'échec important des projets observés dans les années 90. Ce rassemblement a donné naissance à un Manifeste définissant quatre valeurs :

Les individus et leurs interactions avant les processus et les outils ;

Des fonctionnalités opérationnelles avant la documentation ;

Collaboration avec le client plutôt que contractualisation des relations ;

Acceptation du changement plutôt que conformité aux plans.

Les méthodes agiles utilisent un principe de développement itératif qui consiste à découper le projet en plusieurs étapes qu'on appelle « itérations ».

Ces itérations sont en fait des mini-projets définis avec le client en détaillant les différentes fonctionnalités qui seront développées en fonction de leur priorité. Le chef de projet établit alors une macro planning correspondant aux tâches nécessaires pour le développement de ces fonctionnalités.

Le but est d'assumer le fait que l'on ne peut pas tout connaître et anticiper quel que soit notre expérience. On découpe alors le projet en itérations plutôt que de tout prévoir et planifier en sachant que des imprévus arriveront en cours de route.

Voici les avantages du développement itératif :

- Meilleure qualité de la communication : L'utilisateur a la possibilité de clarifier ses exigences au fur et à mesure ;
- Meilleure visibilité : Le client a eu meilleure visibilité sur l'avancement des travaux ;
- Meilleur contrôle de la qualité : les tests sont effectués en continu ;
- Meilleure détection des risques : Les risques sont détectés plus tôt ;
- Motivation et confiance de l'équipe : satisfaction d'atteindre un objectif fixé ;
- Contrôle des coûts : le projet peut être arrêté s'il n'y a plus de budget.

- **Agile Manifesto [1]**

En février 2001, aux États-Unis, dix-sept spécialistes du développement logiciel se sont réunis pour débattre du thème unificateur de leurs méthodes respectives, dites méthodes agiles. Les plus connus d'entre eux étaient Ward Cunningham l'inventeur du Wiki via WikiWikiWeb, Kent Beck, père de l'extremeprogramming et cofondateur de JUnit, Ken Schwaber et Jeff Sutherland, fondateurs de Scrum, Jim Highsmith, prônant l'ASD, Alistair

Cockburn pour la méthode Crystal clear, Martin Fowler, et Dave Thomas ainsi qu'Arie van

Bennekum pour DSDM (Dynamic System Development Method) la version anglaise du RAD (développement rapide d'applications). Ces 17 experts venant tous d'horizons différents réussirent à extraire de leurs concepts respectifs des critères pour définir une nouvelle façon de développer des logiciels.

De cette réunion devait émerger le Manifeste agile, considéré comme la définition canonique du développement agile et de ses principes sous-jacents.

Le Manifeste agile est constitué de quatre valeurs et de 12 principes fondateurs :

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
5. Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
6. La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.
7. Un logiciel opérationnel est la principale mesure d'avancement.
8. Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
9. Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentiel.
11. Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

- SCRUM

Le Scrum ou « mêlée », créée par Ken Schwaber et Jeff Sutherland (signataires du Manifeste) en 1993, est un terme emprunté au rugby qui désigne la solidarité et la force qui lient les membres de l'équipe au succès de l'itération.

Le cycle de vie de Scrum est rythmé par des itérations de quatre semaines qu'on appelle sprints. On peut constater cela sur la figure 4.

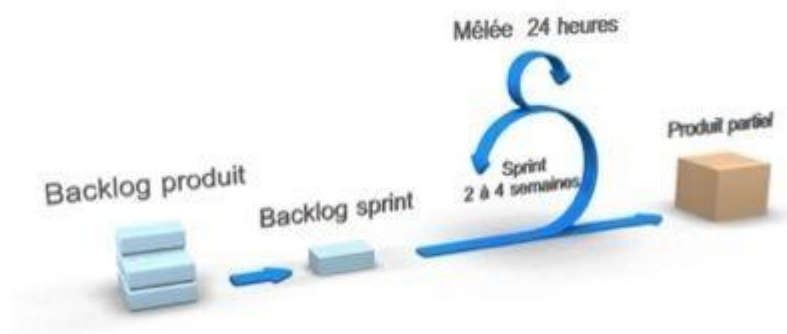


Figure 4. Cycle de vie de SCRUM

Avant chaque sprint, on effectue une réunion de planification appelée le sprint planning meeting qui consiste à sélectionner les exigences prioritaires pour le client dans le produit backlog qui seront développées, testées et livrées au client : le backlog sprint (sous-ensemble du produit backlog).

Des mêlées sont organisées quotidiennement (mêlée) durant le sprint afin de contrôler l'avancement pour s'assurer les objectifs sont tenus. A la fin du sprint, une démonstration des derniers développements est faite au client qui donnera lieu à un bilan qualitatif sur le fonctionnement de l'équipe.

Les valeurs mises en avant par cette méthode sont les suivantes :

- Visibilité : Avoir une vision réelle sur le résultat
 - Inspection : Vérifier l'écart par rapport à l'objectif initial.
 - Adaptation : S'adapter en fonction des écarts constatés afin de les ajuster.
- Scrum est favorable à des petits ajustements fréquents.

- XP (Extreme Programming) [2]

L'extreme programming a été inventée par Kent Beck, Ward Cunningham et Ron Jeffries pendant leur travail sur un projet « C3 » de calcul des rémunérations chez Chrysler. Kent Beck, chef de projet en mars 1996 commença à affiner la méthode de développement utilisée sur le projet. Celle-ci est née officiellement en octobre 1999 avec le livre Extreme Programming Explained de Kent Beck.

L'extreme programming repose sur des cycles rapides de développement (des itérations de quelques semaines) dont les étapes sont les suivantes :

- Une phase d'exploration détermine les scénarios "client" qui seront fournis pendant cette itération ;
- L'équipe transforme les scénarios en tâches à réaliser et en tests fonctionnels ;
- Chaque développeur s'attribue des tâches et les réalise avec un binôme ;
- Lorsque tous les tests fonctionnels passent, le produit est livré.

Le cycle se répète tant que le client peut fournir des scénarios à livrer. Généralement le cycle de la première livraison se caractérise par sa durée et le volume important de fonctionnalités embarquées. Après la première mise en production, les itérations peuvent devenir plus courtes (une semaine par exemple). La Figure 5 nous permet de voir les étapes de développement avec la méthode agile extreme programming.

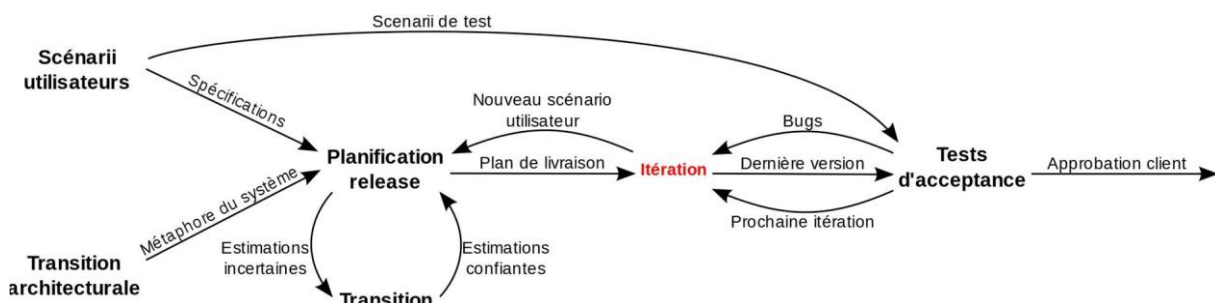


Figure 5. Les étapes de développement avec XP

L'extreme programming repose sur cinq valeurs fondamentales :

1. Communication

C'est le moyen fondamental pour éviter les problèmes. Les pratiques que préconise l'XP imposent une communication intense. Les tests, la programmation en binôme et le jeu du planning obligent les développeurs, les décideurs et les clients à communiquer. Si un manque apparaît malgré tout, un coach se charge de l'identifier et de remettre ces personnes en contact.

2. Simplicité

La façon la plus simple d'arriver au résultat est la meilleure. Anticiper les extensions futures est une perte de temps. Une application simple sera plus facile à faire évoluer.

3. Feedback

Le retour d'information est primordial pour le programmeur et le client. Les tests unitaires indiquent si le code fonctionne. Les tests fonctionnels donnent l'avancement du projet. Les livraisons fréquentes permettent de tester les fonctionnalités rapidement.

4. Courage

Certains changements demandent beaucoup de courage. Il faut parfois changer l'architecture d'un projet, jeter du code pour en produire un meilleur ou essayer une nouvelle technique. Le courage permet de sortir d'une situation inadaptée. C'est difficile, mais la simplicité, le feedback et la communication rendent ces tâches accessibles.

5. Respect

Cette valeur fut ajoutée dans la deuxième édition d'*Extreme Programming Explained* de K. Beck. Cette valeur inclut le respect pour les autres, ainsi que le respect de soi. Les programmeurs ne devraient jamais valider les modifications qui cassent la compilation, qui font échouer les tests unitaires existants ou qui retardent le travail de leurs pairs. Les membres respectent leur propre travail en cherchant toujours la qualité et la meilleure conception pour la solution et cela grâce au refactoring. Le tableau 9 montre un petit récapitulatif et la différence entre la méthode de développement traditionnelle ou classique et les approches agiles.

Tableau 9. Différence entre la méthode de développement traditionnelle et les approches agiles

Thème	Approche traditionnelle	Approche agile
Cycle de vie	En cascade ou en V, sans rétroaction possible, phases séquentielles.	Itératif et incrémental.
Planification	Prédictive, caractérisée par ces plans plus ou moins détaillés sur la base d'un périmètre et d'exigences définies et stables au début du projet.	Adaptative avec plusieurs niveaux de planification (macro et micro planification) avec ajustements si nécessaires au fil de l'eau en fonction des changements survenus.

Documentation	Produite en quantité importante comme support de communication, de validation et de contractualisation.	Réduite au strict nécessaire au profit d'incréments fonctionnels opérationnels pour obtenir le feedback du client.
Equipe	Une équipe avec des ressources spécialisées, dirigées par un chef de projet.	Une équipe responsabilisée où l'initiative et la communication sont privilégiées, soutenue par le chef de projet.
Qualité	Contrôle qualité à la fin du cycle de Développement. Le client découvre le produit fini.	Un contrôle qualité précoce et permanent, au niveau du produit et du processus. Le client visualise les résultats tôt et fréquemment.
Changement	Résistance voire opposition au changement. Processus lourds de gestion des changements acceptés.	Accueil favorable au changement inéluctable, intégré dans le processus.
Suivi de l'avancement	Mesure de la conformité aux plans initiaux. Analyse des écarts.	Un seul indicateur d'avancement : le nombre de fonctionnalités implémentées et le travail restant à faire.
Gestion des risques	Processus distinct, rigoureux, de gestion des risques.	Gestion des risques intégrée dans le processus global, avec responsabilisation de chacun dans l'identification et la résolution des risques. Pilotage par les risques.
Mesure du succès	Respect des engagements initiaux en termes de coûts, de budget et de niveau de qualité.	Satisfaction client par la livraison de valeur ajoutée

Sur cette analyse qu'on vient de faire donc, on peut en déduire des raisons pour lesquelles on doit choisir d'adopter le principe des méthodes agile pour ce projet :

Avec la méthode agile, les risques sont énormément réduits

- Elle permet d'accroître la productivité de l'équipe de développement
- Elle permet aussi de faire face à la complexité actuelle du développement logiciel
- Elle améliore la maintenance du logiciel
- Elle augmente la qualité des logiciels
- Elle augmente la visibilité du projet

Cependant, même si le choix est évident, dès fois on adapte la méthode selon les besoins et les contraintes que demande le projet tout en gardant la structure de

l'agilité : on se doit d'être souple pour avoir des résultats efficace et fiable, satisfaisant le client dans la réalisation du projet.

Cependant, même si le choix est évident, dès fois on adapte la méthode selon les besoins et les contraintes que demande le projet tout en gardant la structure de l'agilité : on se doit d'être souple pour avoir des résultats efficace et fiable, satisfaisant le client dans la réalisation du projet.

..3. Système de gestion de base de données

Un Système de Gestion de Base de Données (SGBD) est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer ces données stockées.

Parmi les logiciels les plus connus on cite : Mongodb, MySQL, PostgreSQL, SQLite, Oracle Database, Firebase, Microsoft SQL Server, Firebird ou Ingres.

Ces systèmes peuvent être catégorisés selon leur fonctionnement :

- **Système propriétaire** : Oracle Database, Microsoft SQL Server, DB2, MaxDB, 4D, dBase, Informix, Sybase
- **Système libre** : MySQL, PostgreSQL, MariaDB, Firebird, Ingres, HSQLDB, Derby
- **Orienté objet** : ZODB, db4o
- **Embarqué** : SQLite, Berkeley DB
- **NoSQL** : Cassandra, Redis, MongoDB, SimpleDB, BigTable, CouchDB, HBase, LevelDB, RethinkDB, Memcached
- Autre système** : Access, OpenOffice.org Base, FileMaker, HyperFileSQL, Paradox, Neo4j.

Chacun de ces SGBD sont adaptés à des projets spécifiques, et ils ont tous ces spécificité et avantages. Par exemple, les SGBD embarqués sont spécifiques au projet de petite taille, comme les applications mobiles ; NoSQL (Not only SQL) sont adaptés au projet dont les données ne sont pas relationnelles.

❖ **MySQL**

MySQL est un système de gestion de bases de données relationnelles distribué sous une double licence GPL et propriétaire. Toutefois, il est plus populaire en open source, et peut être téléchargé à partir de son site Web. Cette édition a toutes les fonctionnalités dont on a besoin pour les applications Web et l'utilisation sûre et sécurisées. Des sites Web comme Google, Wikipedia, Facebook, YouTube utilise MySQL.

MySQL fonctionne sur de nombreux systèmes d'exploitation différents, incluant Linux, Mac OS X, Solaris, Windows. Les bases de données sont accessibles en utilisant les langages de programmation C, C++, VB, VB .NET, C#, Delphi/Kylix, Eiffel, Java, Perl, PHP, Python, Windev, Ruby et Tcl ; une API spécifique est disponible pour chacun d'entre eux. Une interface ODBC appelée MyODBC est aussi disponible. En Java, MySQL peut être utilisé de façon transparente avec le standard JDO.

❖ **PostgreSQL**

PostgreSQL est un système de gestion de base de données sous licence libre, développé à l'origine par l'université de Berkeley. Il s'appuie sur les modèles relationnels et supporte SQL mais apporte des extensions objet (les classes, l'héritage, les fonctions)

; Cela permet de qualifier PostgreSQL de système de gestion de base de données "relationnel objet" (SGBDRO).

PostgreSQL est largement reconnu pour son comportement stable, et pour ses possibilités de programmation étendues, directement dans le moteur de la base de données, via PL/pgSQL. Le traitement interne des données peut aussi être couplé à d'autres modules externes compilés dans d'autres langages.

❖ NoSQL

Le NoSQL est plus souvent élargi que « Non seulement SQL », est une famille de modèles de gestion de base de données qui s'écarte des principes SGBD relationnel traditionnelles incarnées par bases de données comme MySQL. Les systèmes NoSQL, d'autre part, garantissent au plus des trois éléments suivants : Cohérence, Disponibilité, Tolérance de partitionnement de réseau.

Les systèmes NoSQL utilisent une architecture distribuée qui permet l'utilisation d'un grand nombre de serveurs pour atténuer les limitations de taille et de haute disponibilité ou de redondance. Cela rend la récupération rapide et fiable, ce qui réduit la dépendance sur une défaillance du système. Cela s'applique particulièrement aux systèmes modernes, où la performance en temps réel est préférable à la cohérence. En outre, l'architecture distribuée permet aux bases de données NoSQL évolutifs, dans lequel le système peut être étendu en ajoutant simplement plus de machines.

Comparatif des bases de données NoSQL				
Bases	Année de lancement	Schéma de données	Editeur prestataire de support	Positionnement
<u>Cassandra</u>	2008	Orienté colonnes	DataSax (Ex Riptano)	Adoptée par les géants du web et les start-up, Cassandra permet de gérer de gros volumes de données.
<u>Couchbase</u>	2010	Orienté documents	Couchbase	A la différence de MongoDB, Couchbase dispose d'un outil de requêtage normalisé SQL qui facilite sa prise en mains par des développeurs rompus aux bases SQL.
<u>Elasticsearch</u>	2004	Index inversé	Elasticsearch	Connu avant tout pour son moteur de recherche distribué, Elasticsearch tire sa force de l'indexation et de l'analyse des données.
<u>HBase</u>	2006	Orienté colonnes	Hortonworks	Souvent comparé à Cassandra, HBase joue la carte de la très forte volumétrie. Un produit complexe qui exige un gros travail de structuration.
<u>MongoDB</u>	2007	Orienté documents	MongoDB	Base NoSQL la plus populaire, MongoDB est saluée pour la souplesse de sa structure et sa capacité à répondre à un grand nombre de besoins.
<u>Redis</u>	2009	Clé-valeur	Redis Labs	Base de données en mémoire, Redis privilégie la vitesse d'exécution. En contrepartie, ses capacités de requêtage sont limitées.

<u>Riak</u>	2009	Clé-valeur	Basho Technologies	Riak se présente comme une sorte de Redis évoluer en étendant les capacités de requêtage via des index secondaires.
-------------	------	------------	--------------------	---

D'après la comparaison qu'on a vue précédemment, la solution retenue pour mener à bien notre travail est le SGBD MongoDB grâce à ses performances et sa simplicité d'utilisation et de compréhension.

..4. Langage de développement et technologies web

i. Plateforme de développement

Actuellement, il existe énormément de langages de programmation web parmi lesquels les développeurs doivent choisir avant de commencer un nouveau projet. Ce choix est relatif, il peut dépendre de beaucoup de chose :

- Besoins et contraintes du projet
- Stabilité du langage
- Le temps d'exécution du langage
- Nature et typage
- Portabilité
- La maintenance
- Ou encore sa pérennité

Aujourd'hui, les langages les plus utilisés pour le développement des sites et application web sont : Java, Python, C#, PHP, Javascript. Ce sont aussi les langages les plus demandé par les employeurs. Dans notre cas, le responsable a choisi d'utilisé Javascript.

Javascript :

Crée par Brendan Eich en 1995 et développé par association entre Netscape et SUN (créateur du fameux langage Java), il fait son apparition sous le nom de Livescript ,JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation (par exemple) de Node.js. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe. Le langage supporte le paradigme objet, impératif et fonctionnel. JavaScript est le langage possédant le plus large écosystème grâce à son gestionnaire de dépendances npm, avec environs 500 000 paquets en août 2017.

Javascript est un standard ECMA Script. En effet, ECMA Script est un langage de script coté client mais il sert de standard dont les spécifications sont respectées par les autres langages de script comme Javascript ou ActionScript.

Caractéristique du javascript

C'est un script (par exemple ouverture pop-up)

C'est un langage orienté objet

- Il est exécuté par le navigateur du visiteur (le *client*), et dépend donc de celui-ci.
- Il est déterminé par une norme, nommée *ECMA-262* ou *ECMAScript*.

Les avantages du JavaScript sont nombreux :

- Vitesse et gain de temps- Les fonctions du JavaScript ne doivent pas attendre pour des réponses de leur serveur pour agir, ce qui accélère l'ouverture des sites web.
- Simplicité - le JavaScript est relativement simple et facile à apprendre.
- Versatilité - Le JavaScript ne nécessite pas un programme spécial pour l'interpréter (Flash Player, "plug-ins"), ni pour l'écrire. De plus, JavaScript n'occupe pas un grand espace sur les sites web.
- Il est particulièrement utile pour concevoir des sites dynamiques
- Compatible avec tous les supports numériques car plus de 90% d'utilisateurs ont des navigateurs qui peuvent lire et interpréter le JavaScript

ii. Framework

a. Définitions

Un Framework ou structure logicielle est un ensemble cohérent de composants logiciels structurels, éprouvé et réutilisable, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie simple d'un logiciel (architecture). Un Framework se distingue d'une simple bibliothèque logicielle principalement par :

- Son caractère générique, faiblement spécialisé, contrairement à certaines bibliothèques ; un Framework peut à ce titre être constitué de plusieurs bibliothèques chacune spécialisée dans un domaine. Un Framework peut néanmoins être spécialisé, sur un langage particulier, une plateforme spécifique, un domaine particulier.
- Le cadre de travail qu'il impose de par sa construction même, guidant l'architecture logicielle voire conduisant le développeur à respecter certains patterns ; les bibliothèques constitutives sont alors organisées selon le même paradigme.

Les Framework sont donc conçus et utilisés pour modéliser l'architecture des logiciels applicatifs, des applications web, des middlewares et des composants logiciels.

Tous les Framework ne répondent pas aux mêmes besoins, et plusieurs peuvent être utilisés conjointement dans certaines situations. Ils peuvent répondre de manière spécifique à certaines problématiques de développement, comme les Framework de logging, les Framework de persistance et d'ORM, une technique permettant de transformer une table de la base de données en un objet d'une classe manipulable via ses attributs, ou encore les Framework de présentation de contenu web.

Le tableau 14 est un comparatif de quelques Framework

	AngularJS	Angular 2	ReactJS	Vue.js	Ember.js	Meteor.js
Definition	MVW framework	MVC framework	JavaScript library	MVC framework	MVC framework	JavaScript app platform
1st Release	2009	2016	2013	2014	2011	2012
Homepage	angularjs.org	angular.io	reactjs.net	vuejs.org	emberjs.com	www.meteor.com
# Contributors on GitHub	1,562	392	912	62	636	328
GitHub Star Rating	54,402	19,832	57,878	39,933	17,420	36,496

b. Intérêts d'un Framework

Dans le développement d'un logiciel, l'utilisation des Framework n'est pas obligatoire, certes. D'ailleurs, son utilisation sur un projet demande un temps d'apprentissage beaucoup plus long, et des connaissances préalables notamment sur les designs patterns et les bonnes pratiques d'ingénierie informatique. Cependant, les raisons pour laquelle l'utilisation d'un Framework sont indispensables sont :

→ Amélioration de la productivité des développeurs

L'objectif premier d'un Framework est d'améliorer la productivité des développeurs qui l'utilisent. Souvent organisé en différents composants, un Framework offre la possibilité au développeur final d'utiliser tel ou tel composant pour lui faciliter le développement, et lui permet ainsi de se concentrer sur le plus important.

Le développeur qui l'utilise se concentre sur l'essentiel dans son application : chaque formulaire effectue une action, et c'est cette action qui est importante, pas les formulaires.

→ Organisation du projet

Un Framework fournit en effet une architecture prédéfinie et une méthode de conception cohérente et rigoureuse, respectant les bonnes pratiques en matière de génie logiciel. L'utilisation d'un Framework MVC permet de séparer les concepts et peut faciliter le travail en équipe sur les gros projets.

Il permet le découpage logique du code source, la factorisation de composants communs, la séparation des logiques métiers et logiques de présentation et enfin il facilite la maintenance et l'évolution de l'application.

→ Réutilisation des bibliothèques

Les Framework viennent systématiquement avec leur lot de composants réutilisables. Du coup, son utilisation permet de réutiliser le code d'autres développeurs et fait ainsi gagner du temps.

En effet, plusieurs problématiques sont répandues sur les applications web et ont été traitées maintes et maintes fois. Par exemple : la gestion des comptes utilisateurs, de l'authentification, la connexion à différentes sources de données et encore d'innombrable fonctionnalité. L'utilisation des Framework permet de gérer ces cas d'utilisation en intégrant des outils simples, efficaces et éprouvés.

→ **Incitation aux bonnes pratiques**

Avec l'utilisation des Framework, les codes sources se standardisent. Ces bonnes pratiques sont le garant d'un code lisible et compréhensible par tous les développeurs.

→ **Base régulièrement mise à jour**

En choisissant un Framework, on choisit également une communauté active qui détectera et corrigera des failles ou des manques du Framework. En quelque sorte, on mutualise les moyens de développements. Ainsi on peut profiter des mises à jour du Framework avec le lot d'améliorations qu'ils comportent.

c. Choix du Framework

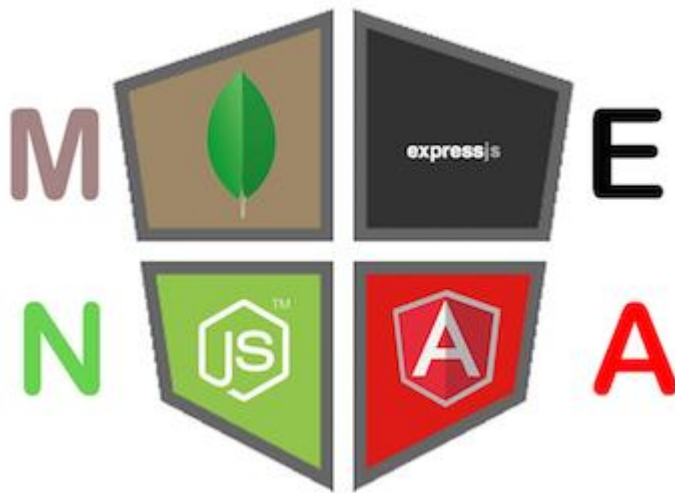
D'après ce qu'on a vu précédemment, il existe plusieurs Framework Javascript. Chaque Framework propose sa vision des choses et sa façon de coder. Par exemple : Le choix d'un Framework peut dépendre en fait de quelques critères dont : l'envergure du projet, la documentation ou encore choix imposé par le client.

Pour notre cas, le choix a été rivé sur la technologie MEAN Stack.

→ **MEAN Stack :**

MEAN est une plateforme fullstack JavaScript pour les applications web modernes. Il est très clair que MEAN vise toutes sortes de développeurs JavaScript (côté serveur et le client) et aussi que c'est une combinaison. Ces composants sont :

- MongoDB
- Express
- AngularJS/ Angular
- NodeJS



MEAN réunit quatre des technologies les plus utilisées et appréciées pour le développement JavaScript, établissant ainsi la base pour créer facilement des applications web complexes.

MongoDB :

C'est un système de gestion de base de données orientée documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. MongoDB permet de manipuler des objets structurés au format BSON (du JSON qui permet de gérer plus de types de valeur) il est de type NoSQL, sans schéma prédéterminé. En d'autres termes, des clés peuvent être ajoutées à tout moment « à la volée », sans reconfiguration de la base. Il est écrit en C++ et distribué sous licence AGPL.

Il faut savoir que MongoDB utilise de la RAM pour les documents les plus demandés, Il s'adapte parfaitement à MEAN en stockant tout en JSON. Bien que très simple à utiliser, il existe un excellent module souvent proposé avec ce stack nommé *Mongoose.js* (couche de persistance pour MongoDB).

Mongoose est un ODM (*Object Document Mapper*). Peut-être que vous connaissez le terme ORM (*Object Relational Mapper*) qui est utilisé en SQL. Et bien c'est la même chose pour le NoSQL (du moins pour les base de données NoSQL de type documents). *Mongoose* va vous simplifier diverses tâches de MongoDB, et va aussi apporter des fonctionnalités comme la population ou les Schémas.

Angular :

L'angular est framework JavaScript coté client de Google (un framework front-end.) L'Angular est souvent appelé un cadre MVW (Model-View-Whatever) mais aussi un framework MVC (Model- View- Controller) Ce Framework est construit avec TypeScript de Microsoft afin de rendre JavaScript plus agile et attrayant pour les grandes entreprises. Il dispose d'une architecture basée sur les composants, d'une DI améliorée (injection de dépendance), d'un service d'enregistrement efficace, de communications inter-composants et plus encore.

L'Angular est une meilleure option pour les applications d'entreprise ou pour des environnements de programmation stricts avec des normes élevées pour la lisibilité du code.

En effet, lorsque l'utilisateur souhaite charger une nouvelle page, Angular va seulement recharger les éléments qui changent en recevant du JSON depuis le serveur. Cela va donc accélérer le chargement des pages.

À la racine du projet, on retrouve un ensemble de fichiers de configuration :

package.json : fichier déclarant les dépendances NPM tirées lors de l'installation du projet et nécessaire à la compilation et les tests.

.editorconfig : ce fichier est issu du projet EditorConfig. Il a pour but de maintenir une cohérence dans le code entre l'ensemble des éditeurs et IDE du marché. Le fichier fonctionne nativement sur certains éditeurs alors qu'un plugin sera nécessaire pour d'autres. Très peu d'éditeurs/IDE ne connaissent pas ce fichier ; c'est donc un standard de fait.

README.md : fichier de présentation du projet au format Markdown utilisé notamment sur Github.

.gitignore : fichier permettant de déclarer les fichiers qui ne doivent pas être commités sur le repository Git.

karma.conf.js : fichier de paramétrage du Test runner Karma. Karma est un outil permettant de lancer des tests sur une série de browser/device automatiquement. Il est déjà configuré pour être lancé sur le navigateur Chrome avec le framework de test Jasmine.

protractor.conf.js : fichier de paramétrage de l'outil de e2e Protractor. E2E, ou end-to-end, est une discipline permettant de réaliser des tests d'intégration ; il est ainsi possible de réaliser des tests simulant un utilisateur final utilisant l'application dans un browser type Chrome.

tslint.json : fichier définissant les règles de codage TypeScript. Tout comme le fichier .editorconfig, il est reconnu par la majorité des éditeurs de code.

angular-cli.json : fichier de paramétrage central utilisé par Angular-cli. Ce fichier permet de définir où sont placés les sources de l'application, les différents fichiers de configuration, les scripts js et css tiers... Ce fichier est largement utilisé par la librairie webpack nouvellement ajoutée à Angular-Cli.

src : à la racine du répertoire src, on retrouve les fichiers classiques index.html, favicon.ico, styles.css, mais également le main.ts (bootstrap d'Angular), le fichier de configuration de la compilation TypeScript tsconfig.json, un fichier de définition TypeScript typings.d.ts, et un ensemble de polyfills utiles à Angular,

src/app : on retrouve les sources de notre premier projet, dont notre nouveau Component : AppComponent.

src/assets : cet espace permet d'y placer tous les assets tels que les images. Lors de la compilation de l'application via Angular-cli, un dossier dist va être créé. Le contenu de ce dossier sera placé à la racine de dist.

src/environments : les fichiers contenus dans ce dossier permettent de définir les variables spécifiques à chaque environnement d'exécution (prod, dev, integration). Par défaut, l'environnement de dev sera utilisé (fichier

environment.ts). Si l'on souhaite utiliser le fichier de production, il est nécessaire d'ajouter le paramètre -env=prod lors de l'appel de la commande ng build.

ExpressJS :

Express.js est un framework pour construire des applications web basées sur Node.js. C'est de fait le framework standard pour le développement de serveur en Node.js

C'est-à-dire qu'il est l'origine de la création d'un serveur http, les Middlewares. Qui sont des fonctions ayant accès à la requête et à la réponse et qui sont appelées avant le routeur par exemple.

NodeJS :

NodeJs sera un serveur web, NodeJS n'est pas un framework mais un environnement très bas niveau. Il va s'occuper de recevoir les requêtes HTTP, de les traiter et de renvoyer un résultat. Node.js repose sur le moteur Javascript V8 de Google. Node.js contient une bibliothèque de serveur HTTP intégrée, ce qui rend possible de faire tourner un serveur web sans avoir besoin d'un logiciel externe comme Apache, et permettant de mieux contrôler la façon dont le serveur web fonctionne.

Aussi, Node.js est non bloquant, basé sur des événements et singlethread (un seul coeur qui s'occupe de tout). L'avantage de Node réside dans sa capacité à répondre à un important nombre de requêtes par secondes. Toutes les requêtes sont traitées en même temps en parallèle. Donc même si la tâche lourde est placée en arrière-plan, le serveur mettra beaucoup de temps pour la terminer car il reste sur le même système *singlethread*.

iii. Outil de développement

➔ Outil de versionning

Un logiciel de gestion de versions ou VCS est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Il agit sur une arborescence de fichiers afin de conserver toutes les versions et les différences entre ces derniers.

Il existe aussi des logiciels et services de gestion de versions décentralisés (distribués) ou DVCS. Git et Mercurial sont deux exemples de logiciel de gestion de versions décentralisés et sont disponibles sur la plupart des systèmes Unix et Windows.

Les logiciels de gestion de versions sont utilisés notamment en développement logiciel pour conserver le code source relatif aux différentes versions d'un logiciel. En fait, ce système permet de mutualiser un développement : un groupe de développeurs autour d'un même développement se sert de l'outil pour stocker toute évolution du code source ; le système gère les mises à jour des sources pour chaque développeur, conserve une trace de chaque changement. Ceux-ci sont, en bonne utilisation, chaque fois accompagnés d'un commentaire. Le système travaille par fusion de copies locale et distante, et non par écrasement de la version distante par la version locale. Ainsi, lorsque deux développeurs, par exemple, travaillent de concert sur une même source, les changements du premier à soumettre son travail ne seront pas perdus lorsque le second, qui a donc travaillé sur une version non encore modifiée par le premier, renvoie ses modifications.

On utilise les VCS pour pouvoir bénéficier des avantages suivants :

- **Backup and Restore.** Les fichiers sont sauvegardés en même temps qu'ils sont modifiés, on peut se déplacer à n'importe quel moment dans le temps.
- **Synchronisation.** Permettre l'échange de fichiers entre les gens et rester à jour avec la dernière version.
- **Retour à court-terme.** Pouvoir annuler toutes les modifications au cas où il y aurait un problème et retourner à la version enregistrée dans la base de données.
- **Retour à long-terme.** On peut retourner à une version antérieure à une date précise pour voir ce qui a changé.
- **Trace des changements.** À chaque fois qu'un fichier est modifié, on peut mettre un message mentionnant le changement.
- **Traces de l'utilisateur.** Chaque changement est associé un utilisateur qui l'a effectué.
- **Sandboxing.** Tester une fonctionnalité dans un environnement indépendant et isolé du système.
- **Branching and merging.** Faire les modifications dans une branche séparée, puis faire un merge pour les combiner.

Nous allons présenter dans le tableau 15, les différents outils de versionning avec leurs avantages et inconvénients respectives.

Tableau 14. Comparaison entre outil de versionning : SVN, Git et Mercurial

	SVN	Git	Mercurial
+	<p>Système plus récent basé sur CVS</p> <p>Inclus des opérations atomiques</p> <p>Moins d'opérations sur les branches</p> <p>Large variété de plugins pour IDEs</p>	<p>Il existe de nombreux services sur internet qui permettent d'héberger des projets personnels</p> <p>Opérations rapides</p> <p>Moins d'opérations sur les branches</p> <p>Possibilité de travailler Offline</p> <p>Moins de conflits lors des merges</p> <p>La granularité des commits est plus fine</p> <p>Repository Décentraliser</p>	<p>Plus facile que git</p> <p>Meilleure documentation</p> <p>Repository</p> <p>Décentraliser</p>

-	Encore de bugs lorsqu'on renomme un fichier ou un répertoire		Pas de merge pour 2 parents
	Manque dans la gestion de repository		Basé sur des extensions plutôt que des scripts
	Lent	Pas optimal pour un seul programmeur	Moins de puissance dans la gestion du
	Repository centralisé	Plus de commande	« out of box »

(+ : avantages, - : inconvénients)

Ainsi, vu les plusieurs avantages qu'offre Git, nous l'avons choisi pour la gestion des versions du projet.

□ Environnement de développement Intégré (IDE)

Un IDE est une interface qui permet de développer, compiler et exécuter un programme dans un langage donné. Il comporte donc un éditeur pour le code source, un compilateur, un débogueur, ainsi que, souvent, un outil permettant de construire les interfaces graphiques des logiciels.

Un IDE rend plus pratique la programmation, peut disposer de fonctions de complétion automatique de textes pour accélérer la saisie.

Il dispose en général d'une documentation sur les outils et méthodes du langage concerné et facilite l'accès aux propriétés des bibliothèques communes.

L'utilisation d'un IDE dans le développement d'un logiciel n'est en fait pas obligatoire. Néanmoins, on a décidé d'utiliser Visual Studio Code, un IDE très complet et qui a plusieurs avantages rendant le développement du projet plus aisé.

- Facile à mettre en place
- L'auto-complétions
- Coloration de syntaxe
- Débogage rapide
- Vérification rapide
- Support de plusieurs langages de programmation et offre une facilité pour l'installation et configuration des Framework
- Multiplateforme
- Son intégration avec l'outil de gestion de version git
- Il a une communauté très active, et possède de nombreux plugins pour faciliter le développement.

Chapitre 5 ANALYSE CONCEPTUELLE

5.1. Présentation d'UML

5.1.1. Historique

Depuis 1974 jusqu'à 1990, de nombreuses approches objets ont apparues, ensuite en 1994, une cinquantaine de méthodes de conception objets ont existé, parmi elles, trois méthodes sont véritablement émergées :

- La méthode OMT de James Rumbaugh représentant graphiquement les aspects statique, dynamique et fonctionnel du système.
- La méthode BOOCH'93 de Grady Booch introduisant le concept paquetage (package).
- La méthode OOSE de Ivar Jacobson (Object Oriented Software Engineering) donnant une analyse sur la description des besoins des utilisateurs (cas d'utilisation).

Ces trois analystes ont décidé d'unir leur effort en constituant un langage commun, le fruit de ce travail est proposé en 1996 suite à une requête RFP (Request For Proposal), où la version 0.9 d'Unified Modeling Language a été présentée.

Le terme unified signifie que les analystes ont établi un couplage des concepts objets.

Le terme langage signifie qu'il s'agit de créer un langage de modélisation, et pas une méthode.

En janvier 1997, la version 1.0 d'UML est proposée initialement à l'OMG. Ce dernier ne l'a accepté qu'en Novembre 1997 dans sa version 1.1 où UML a été normalisé par l'OMG comme un standard international.

UML a évolué rapidement. En 1998, UML 1.2 a vu le jour avec des changements cosmétiques.

En Mars 2000, la version 1.3 d'UML a apparue avec des modifications dans le cas d'utilisation et les diagrammes d'activité.

En Septembre 2001, avec l'ajout des nouveaux composants, et profils, la version UML 1.4 est née.

En Avril 2004, une autre version d'UML est créée, c'est la version 1.5.19

Ainsi, en Juillet 2005, la version 2.0 est proposée, avec de nouveaux diagrammes et une bonne prise en compte de l'aspect dynamique.

En Avril 2006, la version 2.1 est créée. En Février 2009, UML s'est stabilisé à la version 2.2

5.1.2. Définitions

L'UML est un langage de modélisation graphique apparu dans le cadre de la conception orientée objet : c'est un langage de modélisation objet. En effet, il constitue un support d'analyse adéquat pour concevoir les objets.

D'après l'OMG, l'UML est un langage visuel dédié à la spécification, construction et la documentation des artefacts d'un système logiciel : il est centré sur l'architecture du logiciel guidée par les besoins d'utilisateur du système.

La modélisation graphique d'UML permet ainsi de :

- D'exprimer visuellement une solution objet, de faciliter la comparaison et évolution de solution, grâce à sa représentation graphique.
- Un gain de précision et de stabilité.
- La structuration cohérente des fonctionnalités et des données du logiciel.
- Bien définir les besoins clients.

- De vulgariser les aspects liés à la conception et à l'architecture, propres au logiciel, au client.
- D'apporter une compréhension rapide du programme à d'autres développeurs externes en cas de reprise du logiciel ou de maintenance.

5.1.3. Vue et diagrammes d'UML

Le langage UML est organisé en paquetages, chaque paquetage représente des concepts du système exprimé graphiquement par des diagrammes, chacun s'intéresse à un aspect particulier, ces diagrammes constituent des vues qui décrivent le système entier.

a) Vue externe du système par les utilisateurs finaux

- Diagramme de cas d'utilisation

Ce diagramme est constitué d'acteurs et de cas d'utilisations et il illustre les relations entre ces éléments. Un acteur est une entité extérieure qui déclenche l'un des cas d'utilisation.

Le diagramme de cas d'utilisation englobe les fonctionnalités offertes par le système et décrit son comportement, en identifiant les actions et les interactions entre les acteurs.

Ce diagramme est utilisé dans la phase d'analyse afin de définir les besoins des utilisateurs.

La figure 7 représente le formalisme d'un diagramme de cas d'utilisation

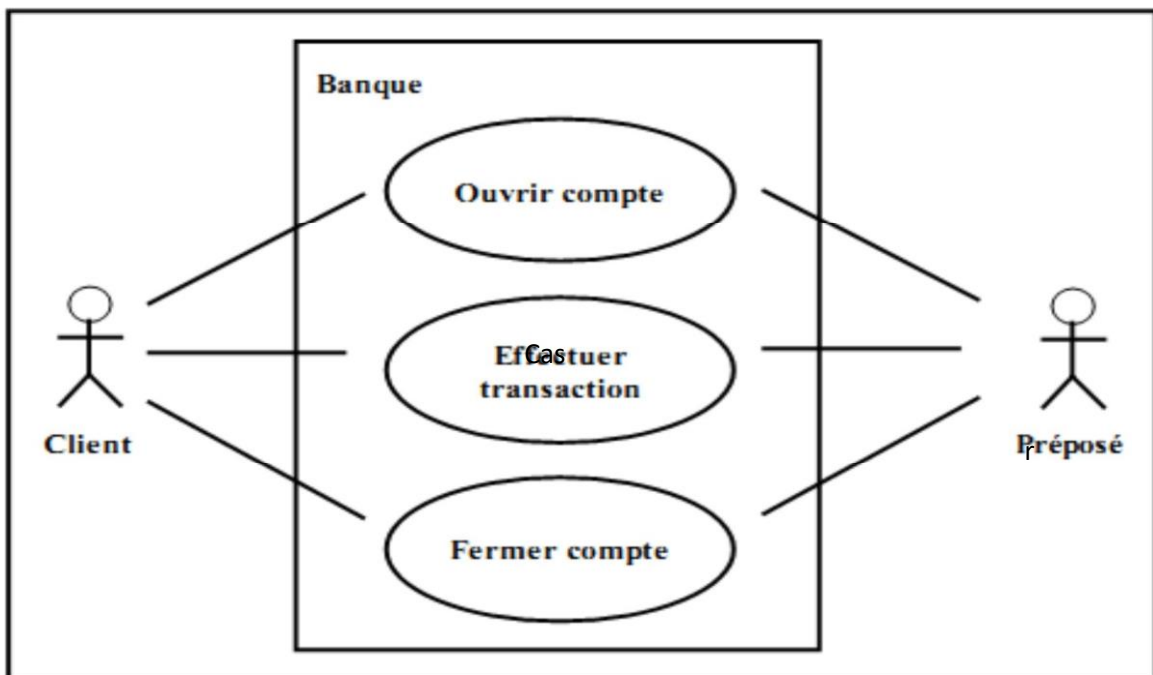


Figure 7. Formalisme d'un diagramme de cas d'utilisation

b) Vue logique statique représentant la structure des objets et leurs relations

Diagramme de classe

Il est constitué des classes, chacune avec sa composition interne (attributs, méthodes) et chaque classe correspond à un concept du système conçu, des associations, des interfaces et des modules (packages).

Le diagramme de classes couvre l'architecture d'un système, il est utilisé pour la description statique des données et des traitements.

La figure 8 représente le formalisme d'un diagramme de classe

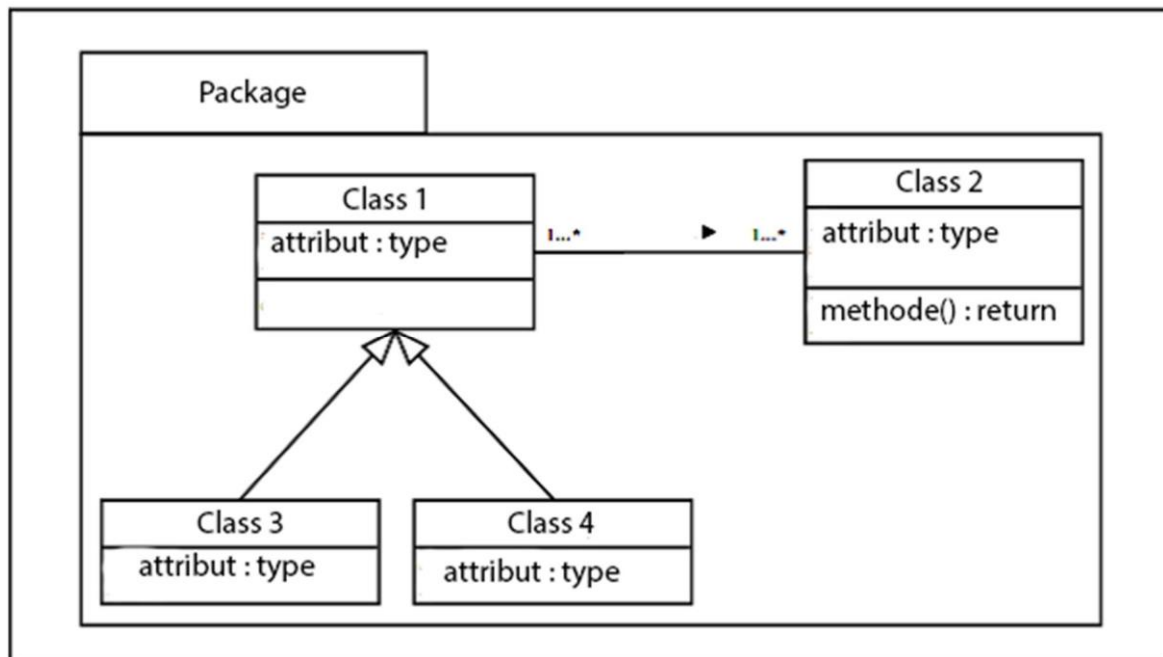


Figure 8. Formalisme d'un diagramme de classe

- Diagramme d'objet

Le diagramme d'objet modélise des exemples de classes. Il décrit le système à un instant particulier afin de vérifier le diagramme de classes.

Le diagramme d'objets utilise des éléments du diagramme de classes.

c) Vue logique dynamique représentant le comportement du système

- Diagramme de séquence

C'est un diagramme d'interaction entre les acteurs et le système qui se concentre sur l'ordre temporel des messages entre. Il présente le déroulement d'un cas d'utilisation sous forme d'un scénario entre les acteurs et leurs interactions séquentielles par le changement des messages entre les objets, en respectant l'ordre chronologique (en fonction de temps). La figure 9 montre le formalisme d'un diagramme de séquence

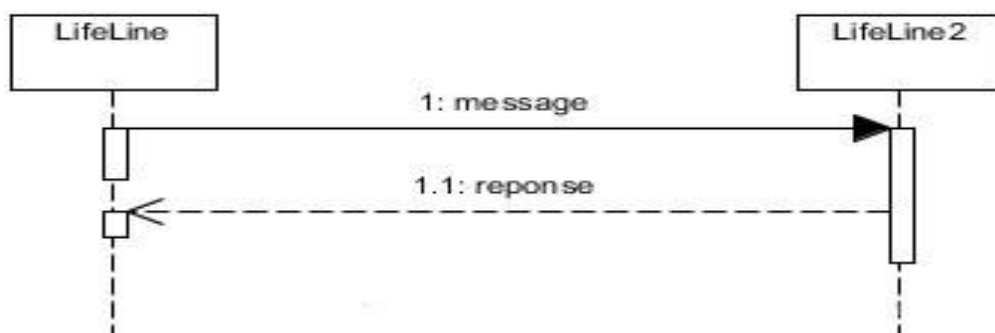


Figure 9. Formalisme d'un diagramme de séquence

- Diagramme de collaboration

C'est un diagramme d'interaction entre objets comme le diagramme de séquence mais il permet de représenter le contexte de l'interaction. On peut y préciser les états des objets qui interagissent.

Un diagramme d'interaction sert à visualiser l'organisation d'un ensemble d'entités qui réalisent un des comportements du système, et il présente le rôle des classes et des associations d'une collaboration, et les interactions entre ces classes.

- Diagramme d'état transition

Il est composé des états, des transitions et des événements.

Ce diagramme exprime le comportement d'une entité du système, il s'intéresse aux événements externes en couvrant les états possibles d'une classe ou d'un composant. Le changement d'un état d'un objet à un autre état (transition) est provoqué par un événement responsable à ce changement.

- Diagramme d'activités

Il est composé des actions et des transitions entre ces actions. Il permet d'exprimer le déroulement et le flot de contrôle interne d'un cas d'utilisation en décrivant le séquençement des activités et leur coordination.

La figure 10 représente le formalisme d'un diagramme d'activités

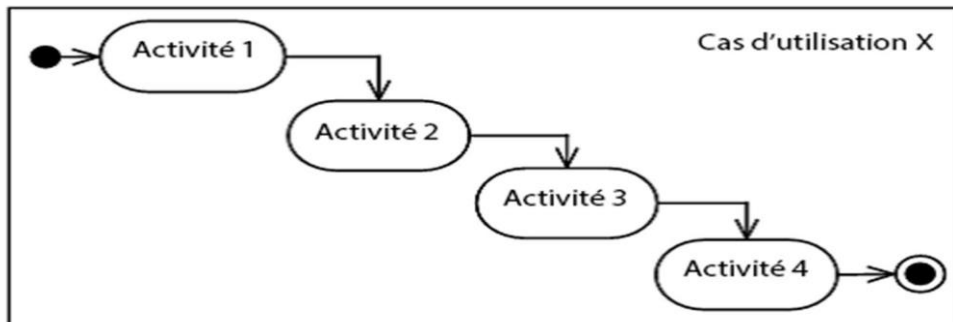


Figure 10. Formalisme d'un diagramme d'activité

d) Une vue d'implémentation représentant les composants logiciels

- Diagramme de composant

C'est un diagramme décrivant l'organisation du système du point de vue des éléments logiciels comme les modules (paquetages, fichiers sources, bibliothèques, exécutables), des données (fichiers, base de données) ou encore d'éléments de configuration (paramètres, scripts, fichiers de commandes). Il permet de mettre en évidence les dépendances entre les composants.

e) Vue de déploiement représentant la répartition des composants

- Diagramme de déploiement

Il représente l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux. La figure 11 illustre un exemple de diagramme de déploiement

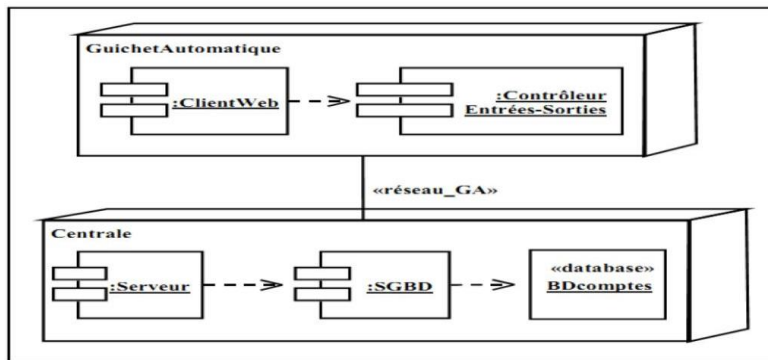


Figure 11. Exemple d'un diagramme de déploiement

5.2. Présentation de SCRUM

Pour mener à bien le projet, il est préférable d'établir une méthode pour bien gérer le développement. Mais être trop disciplinaire aussi ne permet pas aux développeurs de faire évoluer rapidement son travail. C'est pour cela qu'on a choisi d'adapter une méthode de développement correspondante à l'équipe, c'est-à-dire pas trop rigide et permettant d'optimiser au maximum la productivité de chacun.

Ainsi, les règles de base qu'on doit suivre sont mentionnées dans le « Manifesto Agile ». Et pour ne pas perdre l'agilité, on s'est inspiré de la méthode SCRUM et de XP mais on ne l'a pas implémenté entièrement. La méthode est alors adaptée en fonction du projet.

5.2.1. Les User Stories

La notion provient de XP. C'est une phrase simple dans le langage de tous les jours permettant de décrire avec suffisamment de précision le contenu d'une fonctionnalité à développer.

Mike Cohn, auteur du livre *User Stories Applied*, préconise l'emploi de la formulation suivante : As a <type of user>, I want <some goal> so that <some reason>

(En tant que <rôle d'utilisateur>, je veux <un but> afin de <une justification>)

Comme par exemple dans notre cas :

En tant que client, je veux filtrer la recherche des produits afin de voir tous les produits recherchés.

En tant que client, je veux une interface afin de voir le prix du produit.

L'User Story est ainsi une technique permettant de formaliser synthétiquement les besoins sans perdre de vue l'essentiel : le besoin concerne QUI, en QUOI il consiste et dans quel BUT.

5.2.2. Le carnet de produit ou backlog

Le backlog de produit est inspiré de SCRUM. Il sert à planifier les actions à entreprendre et surtout c'est une liste priorisée d'exigence. En d'autres termes, le Backlog est une liste ordonnancée (priorisée) des besoins (généralement formulés sous forme d'User story) du projet.

5.2.3. Les sprints

Sprint est le terme utilisé dans Scrum pour itération. C'est l'intervalle de temps court (1 mois maximum) pendant lequel l'équipe de développement va concevoir, réaliser et tester de nouvelles fonctionnalités.

. Les tests

Les différents tests se font en local avant la validation de l'encadreur. Chaque modification doit être testée pour que le site ne soit pas hors service au cas où il y aurait un bug. Puis on vérifie les pages du site et si tout marche bien, on envoie les modifications sur le serveur.

Les différents types de test qu'on a fait pour la gestion du projet sont :

- Smoketesting

C'est le test préliminaire qui révèle les simples erreurs assez sévères pour stopper le fonctionnement de l'application.

- Sanitytesting

C'est un test basique qui évalue rapidement si le calcul est pu être vrai. C'est là qu'on vérifie si le résultat rendu est rationnel ou non. On cherche seulement à trouver les valeurs qui pourraient être fausses mais pas à chercher les erreurs.

- Regressiontesting

Tests d'un programme préalablement testé, après une modification, pour s'assurer que des défauts n'ont pas été introduits ou découverts dans des parties non modifiées de l'application.

- Usabilitytesting

C'est une méthode permettant d'évaluer un produit en le faisant tester par des utilisateurs. Il permet d'observer directement la façon dont l'utilisateur se sert d'une application et ainsi identifier concrètement les véritables difficultés qu'il rencontre, aussi appelés problèmes d'utilisabilité.

5.3. Dictionnaire de données

Un dictionnaire de données est une collection de métadonnées ou de données de référence nécessaire à la conception d'une base de données.

Ce dictionnaire est un outil important car il constitue la référence de toutes les études effectuées ensuite. Il est représenté par un tableau à quatre colonnes contenant le nom du champ rangé par ordre alphabétique, la description, le type de donné et sa taille.

Dictionnaire de données

Nom de la rubrique	Description	Type	Taille
Id_annonce			
titreAnnonce			
DescriptionAnnonce			
imageAnnonce			
prixAnnonce			
createBy			
Id_blog			
Titreblog			
contenuBlog			
like			
dislike			
Likeby			
Dislikeby			
Createby			
Id_user			
Nom			
Prenom			

Email			
password			

A: Alphabétique

AN: Alpha-numérique

D: Date

5.4. Règles de gestion

La règle de gestion est une règle suivie ou une exigence formulée permettant de faciliter les cas et les cardinalités lors de la conception d'un projet. En d'autres termes, elle définit l'ensemble des règles à respecter pour les actions.

RG1 : Un utilisateur peut commander plusieurs annonces.

RG2 : Un utilisateur peut poster et proposer plusieurs annonces.

RG3 : Un produit peut être proposé par plusieurs utilisateurs avec des prix et des quantités différentes.

RG4 : Un utilisateur peut poster un ou plusieurs articles.

RG5 : Un utilisateur peut commenter un ou plusieurs articles.

RG6 : Un utilisateur peut mettre un pouce vert ou rouge sur un ou plusieurs blogs

RG7 : Un utilisateur peut contacter un ou plusieurs vendeurs

5.5. Modélisation du domaine

Le modèle de domaine nous représente les relations entre les classes persistantes dans la base de données du système

5.6. Représentation et spécification des besoins

5.6.1. Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation représente les interactions fonctionnelles entre les acteurs et le système étudié.

a. Définitions

Les acteurs : ils représentent un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement sur le système étudié. Un acteur participe à au moins un cas d'utilisation.

Les cas d'utilisation (use case) représentent un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier. Un cas d'utilisation modélise un service rendu par le système.

Association : utilisée dans ce type de diagramme pour relier les acteurs et les cas d'utilisation par une relation qui signifie simplement « participe à ».

Inclusion : Un cas A inclut un cas B si le comportement décrit par le cas A inclut le comportement du cas B : le cas A dépend de B. Lorsque A est sollicité, B l'est obligatoirement, comme une partie de A.

Extension : On dit qu'un cas d'utilisation A étend un cas d'utilisation B lorsque le cas d'utilisation A peut être appelé au cours de l'exécution du cas d'utilisation B. Exécuter B peut éventuellement entraîner l'exécution de A.

Généralisation : Un cas A est une généralisation d'un cas B si B est un cas particulier de A.

b. Identification des acteurs

Le tableau montre les acteurs du projet ainsi que leur description.

Acteurs	
Administrateurs	Ce sont les personnes qui s'occupent de mettre à jour les données sur le site web. Ils assurent aussi la gestion des bases de données, et le suivi de tout ce qui se passe et doit se passer sur le site.
Client	C'est la personne ayant un compte utilisateur sur le site, et pouvant suivre le processus d'achat et de vente et aussi la gestion de son propre blog
Visiteurs	C'est une personne n'ayant pas encore de compte mais qui visite le site anonymement.

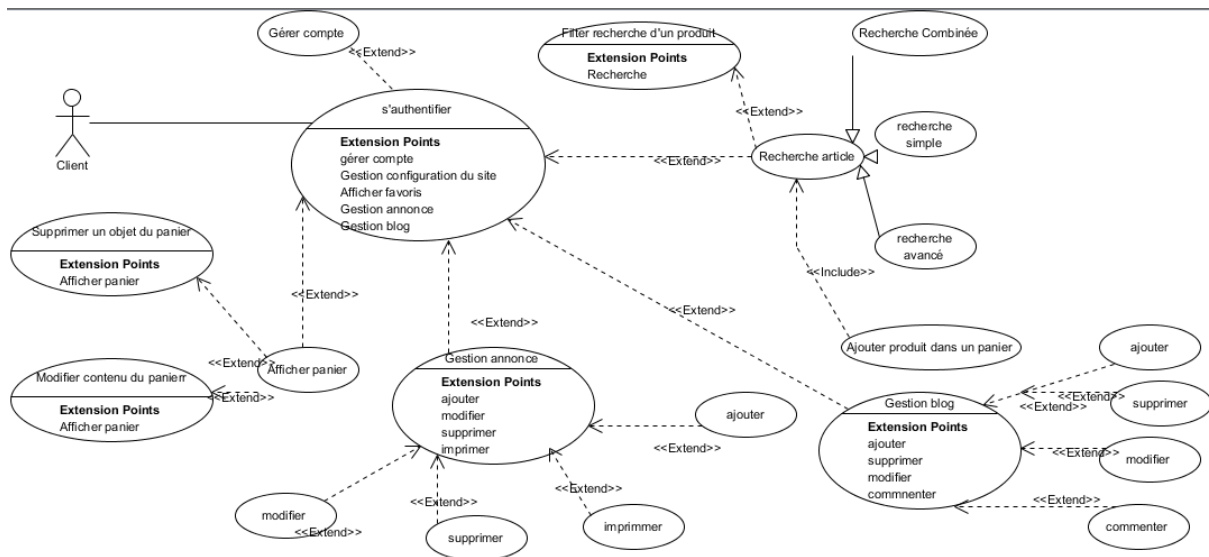
5.7. Identification des cas d'utilisation

Les cas d'utilisation sont donc toutes les actions que les acteurs peuvent effectuer sur le système, ainsi que les réponses du système face à cela.

Acteurs	Cas d'utilisation
Client	<ul style="list-style-type: none"> o S'authentifier o Recherche des produits o Mettre en favoris un produit o Gestion panier o Gérer Compte o Gestion annonce o Gestion du Blog o Commenter un Blog o Mettre un pouce vert ou pouce rouge un blog
Administrateurs	<ul style="list-style-type: none"> o S'authentifier o Gestion du Compte o Gestion de l'activation des fonctionnalités
Visiteurs	<ul style="list-style-type: none"> o Crée un Compte o Consulter les pages publics o Se connecter comme compte test

Diagramme de cas d'utilisation des clients

La figure 17 montre le diagramme de cas d'utilisation des clients



□ Diagramme de cas d'utilisation des administrateurs du système

La figure 15 représente le diagramme de cas d'utilisation des administrateurs du système

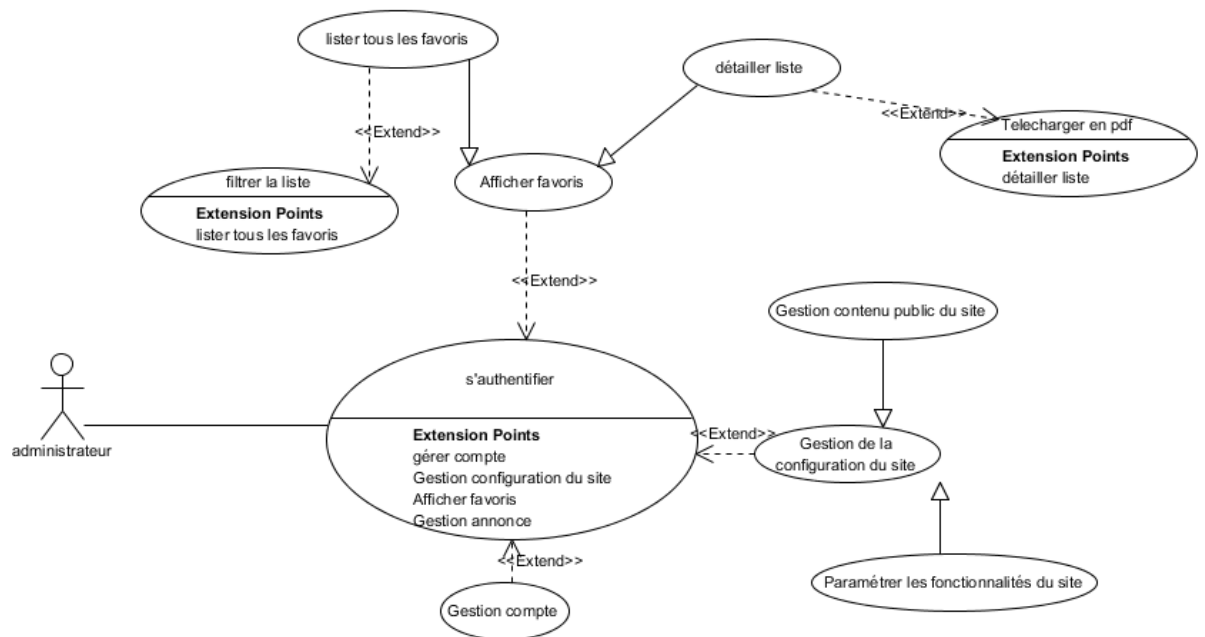
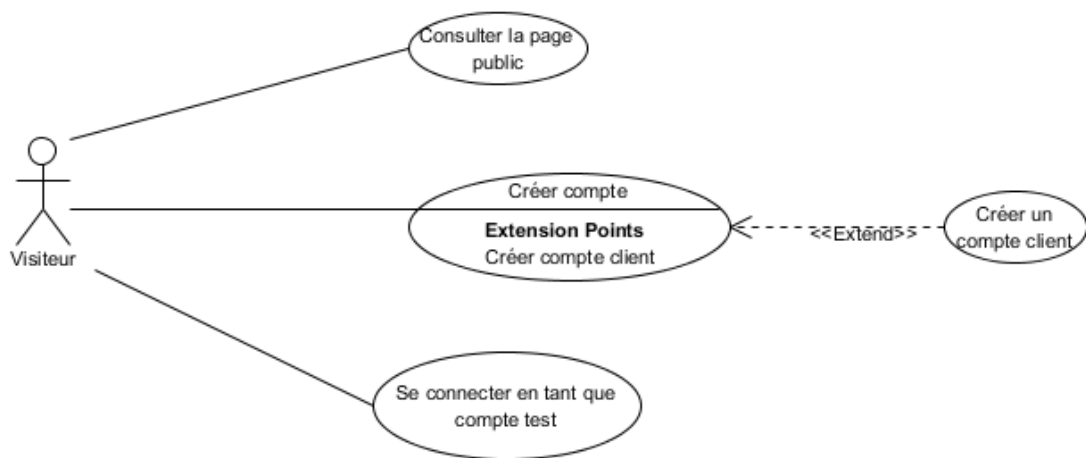


Diagramme de cas d'utilisation des visiteurs

La figure 17 montre le diagramme de cas d'utilisation des visiteurs



Description et détail des cas d'utilisations

Il consiste à décrire textuellement de façon précise et détaillée les scénarios qui définissent la suite logique des actions qui constituent le cas.

□ Description textuelle du cas d'utilisation « s'authentifier »

Objectifs : l'utilisateur pourra accéder à la page d'accueil du site

Acteur : Administrateur, Client

Pré condition : l'utilisateur n'est pas encore authentifié

Post Condition : utilisateur accède au page d'accueil

Scénario nominale :

1. L'utilisateur accède à l'URL
2. Le système affiche la page d'authentification
3. L'utilisateur saisi son login, son mot de passe et clique sur connecter
4. Le système vérifie les informations dans la base de données
5. La page d'accueil est affichée par le système

Scénario d'exception :

6. Le système affiche un message d'erreur et revient à l'étape 2

➔ Description textuelle du cas d'utilisation « filtrer la recherche »

Objectifs : l'utilisateur pourra voir les produits qu'il recherche

Acteur : Client

Pré condition : le client est authentifié

Post Condition : la liste des produits

Scénario nominale :

1. Le système affiche le formulaire de recherche
2. L'utilisateur saisit les critères de recherche
3. L'utilisateur clique sur le bouton recherche
4. Le système affiche la liste des articles

Scénario d'exception :

5. Le système affiche un résultat vide

➔ **Description textuelle du cas d'utilisation « ajouter un produit dans le panier »**

Objectifs : l'utilisateur ajoute des articles dans le panier afin de les commander après

Acteur : Client

Pré condition : l'utilisateur a fait une recherche et accède à la liste des articles

Post Condition : le nombre d'article dans le panier augmente

Scénario nominale :

1. L'utilisateur sélectionne un produit
2. L'utilisateur clique sur le bouton ajouter au panier
3. Le système enregistre les informations dans la base de données
4. Le système affiche la liste des produits dans le panier

➔ **Description textuelle du cas d'utilisation « supprimer un produit du panier »**

Objectifs : l'utilisateur pourra supprimer un article du panier

Acteur : Client

Pré condition : le client accède à la page panier et le panier n'est pas vide

Post Condition : affichage de la liste des articles dans le panier

Scénario nominale :

1. L'utilisateur supprime un produit
2. Le système supprime le produit du panier de la base de données
3. Le système revient sur la page gestion de panier

➔ **Description textuelle du cas d'utilisation « ajouter une nouvelle article »**

Objectifs : l'utilisateur ajoute des articles afin de les publier après

Acteur : Client

Pré condition : l'utilisateur accède à la page blog

Post Condition : le nombre d'article dans la page augmente

Scénario nominale :

1. L'utilisateur clique sur le bouton nouveau
2. L'utilisateur ajoute le titre et le contenu du blog
3. L'utilisateur clique sur le bouton ajouter
4. Le système enregistre les informations dans la base de données
5. Le système affiche la liste des articles dans la page blog

→ Description textuelle du cas d'utilisation « supprimer un article »

Objectifs : l'utilisateur pourra supprimer un article

Acteur : Client

Pré condition : le client accède à la page blog et qui n'est pas vide, dont l'auteur est l'utilisateur

Post Condition : affichage de la liste des articles dans la page blog

Scénario nominale :

1. L'utilisateur supprime un article
2. Le système supprime l'article de la base de données
3. Le système revient sur la page blog

→ Description textuelle du cas d'utilisation « modifier un article »

Objectifs : l'utilisateur pourra modifier un article

Acteur : Client

Pré condition : le client accède à la page blog et qui n'est pas vide, dont l'auteur est l'utilisateur

Post Condition : l'article sera modifié dans la liste

Scénario nominale :

1. L'utilisateur clique sur le bouton modifier
2. L'utilisateur entre le nouveau titre ou le contenu du blog
3. L'utilisateur clique sur le bouton sauver
4. Le système modifie l'article de la base de données

5. Le système revient sur la page blog

→ **Description textuelle du cas d'utilisation « commenter un article »**

Objectifs : l'utilisateur pourra commenter un article

Acteur : Client

Pré condition : le client accède à la page blog et qui n'est pas vide

Post Condition : affichage d'un commentaire dans la page blog

Scénario nominale :

1. L'utilisateur clique sur le bouton commenter
2. L'utilisateur entre son commentaire
3. L'utilisateur clique sur le bouton commenter
4. Le système ajoute un commentaire dans la base de données
5. Le système revient sur la page blog

→ **Description textuelle du cas d'utilisation « mettre un pouce vert ou rouge »**

Objectifs : l'utilisateur pourra aimer ou détester un article

Acteur : Client

Pré condition : le client accède à la page blog et qui n'est pas vide

Post Condition : le nombre de pouce vert ou rouge augmente

Scénario nominale :

1. L'utilisateur clique sur le pouce vert ou rouge
2. Le système incrémente le nombre de j'aime ou je n'aime pas dans la base de données
3. Le système revient sur la page blog

→ **Description textuelle du cas d'utilisation « ajouter une nouvelle annonce »**

Objectifs : l'utilisateur ajoute des annonces afin de les publier après

Acteur : Client

Pré condition : l'utilisateur accède à la page annonce

Post Condition : le nombre d'annonce dans la page augmente

Scénario nominale :

1. L'utilisateur clique sur le bouton nouveau
2. L'utilisateur ajoute le titre, la description, le prix et l'image de l'annonce

3. L'utilisateur clique sur le bouton ajouter
4. Le système enregistre les informations dans la base de données
5. Le système affiche la liste des annonces dans la page annonce

→ Description textuelle du cas d'utilisation « supprimer un annonce »

Objectifs : l'utilisateur pourra supprimer un article

Acteur : Client

Pré condition : le client accède à la page annonce et qui n'est pas vide, dont l'auteur est l'utilisateur

Post Condition : affichage de la liste des annonces dans la page annonce

Scénario nominale :

1. L'utilisateur supprime une annonce
2. Le système supprime l'annonce de la base de données
3. Le système revient sur la page annonce

→ Description textuelle du cas d'utilisation « modifier une annonce »

Objectifs : l'utilisateur pourra modifier une annonce

Acteur : Client

Pré condition : le client accède à la page annonce et qui n'est pas vide, dont l'auteur est l'utilisateur

Post Condition : l'annonce sera modifiée dans la liste

Scénario nominale :

1. L'utilisateur clique sur le bouton modifier
2. L'utilisateur entre le nouveau titre ou la description de l'annonce
3. L'utilisateur clique sur le bouton sauver
4. Le système modifie l'annonce de la base de données
5. Le système revient sur la page annonce

→ Description textuelle du cas d'utilisation « contacter un utilisateur »

Objectifs : l'utilisateur pourra contacter un autre utilisateur

Acteur : Client

Pré condition : le client accède à la page message

Post Condition : les utilisateurs peuvent envoyer ou recevoir des messages

Scénario nominale :

1. L'utilisateur clique sur le bouton contacter
2. L'utilisateur entre le message
3. L'utilisateur clique sur le bouton envoyer
4. Le système enregistre le message dans la base de données
5. Le système revient sur la page message

→ **Description textuelle du cas d'utilisation « Mettre en favoris une annonce »**

Objectifs : l'utilisateur pourra voir toute la liste de favoris

Acteur : Client

Pré condition : le client accède au page annonce

Post Condition : les utilisateurs peuvent voir toute la liste de favoris

Scénario nominale :

1. L'utilisateur clique sur l'icône favoris d'une annonce
2. Le système enregistre l'annonce choisit
3. Le système revient sur la page d'annonce
4. L'utilisateur clique sur le page favori
5. Le système affiche toutes les annonces

→ **Description textuelle du cas d'utilisation « Choisir la langue »**

Objectifs : l'utilisateur pourra choisir de quelle langue du site

Acteur : Client

Pré condition : le client accède au page menu

Post Condition : les utilisateurs peuvent choisir de quelle langue s'affichera la page

Scénario nominale :

1. L'utilisateur clique sur l'icône langue
2. L'utilisateur choisir entre la langue française ou anglais
3. Le système enregistre la langue choisit
4. Le système revient sur la page d'accueil

→ **Description textuelle du cas d'utilisation « Chat entre utilisateur »**

Objectifs : l'utilisateur pourra envoyer des messages entre eux

Acteur : Client

Pré condition : le client accède au page message

Post Condition : les utilisateurs peuvent discuter

Scénario nominale :

1. L'utilisateur clique sur message
2. Le système affiche de choisir quel type de discussion y accéder
3. L'utilisateur choisit son type
4. Le système enregistre et de rediriger l'utilisateur dans un groupe de chat

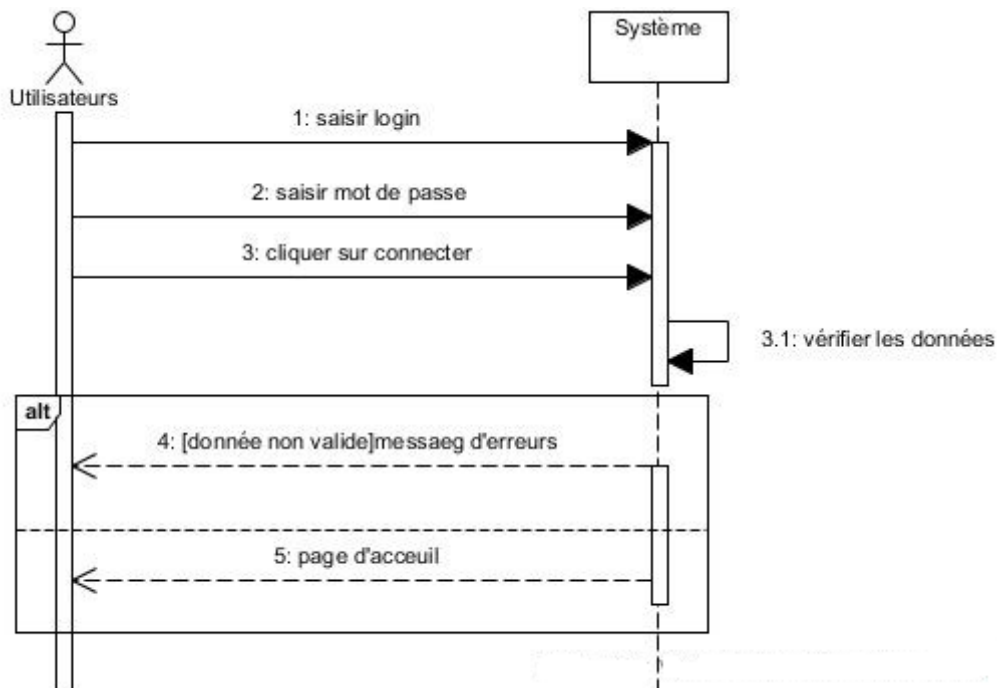
Diagramme de séquence système

Rappelons, un diagramme de séquence est un diagramme d'interaction entre les acteurs et le système qui se concentre sur l'ordre temporel des messages entre. Il présente le déroulement d'un cas d'utilisation sous forme d'un scénario entre les acteurs et leurs interactions séquentielles par le changement des messages entre les objets, en respectant l'ordre chronologique (en fonction de temps).

Et Les diagrammes de séquence système sont des diagrammes de séquence représentant les comportements du système vu de l'extérieure sans montrer comment il l'a réalisé.

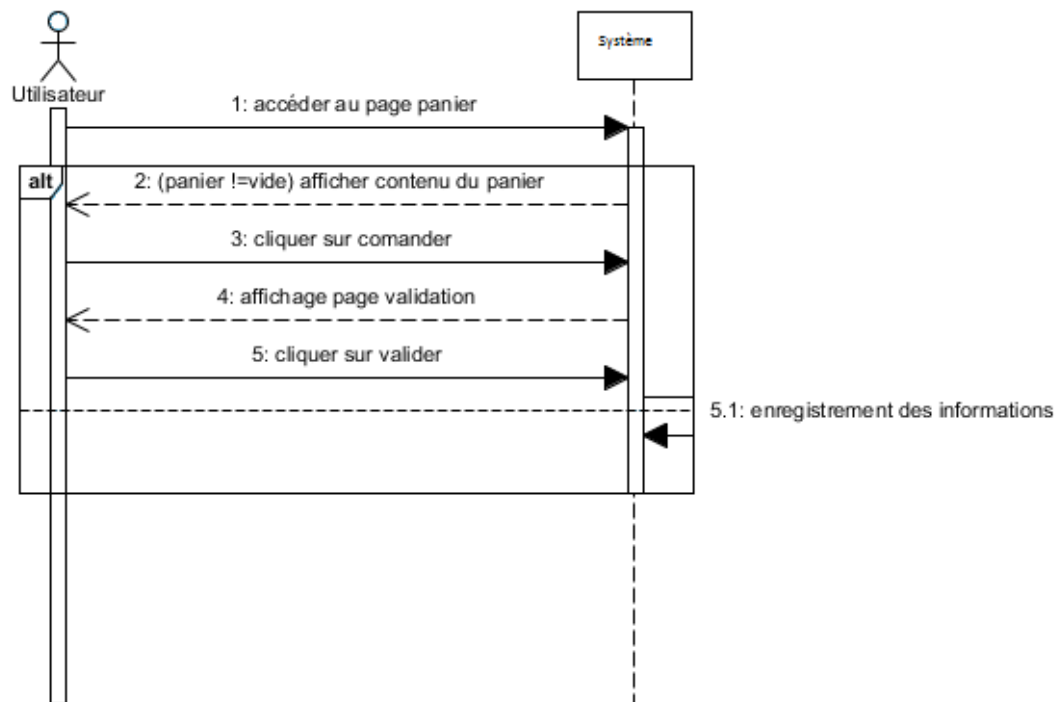
5.6.2.1. DSS du cas d'utilisation : « S'authentifier »

La Figure 18 nous montre le diagramme de séquence système pour ce cas d'utilisation.



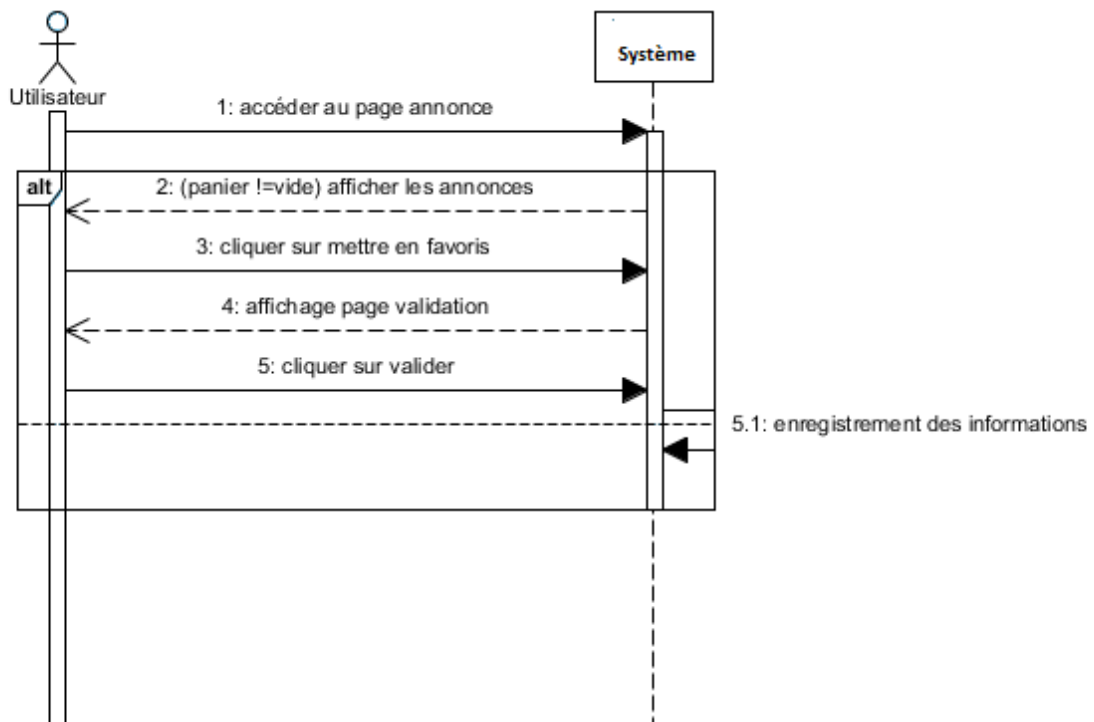
5.6.2.1. DSS du cas d'utilisation : « Commander un produit »

La Figure 18 nous montre le diagramme de séquence système pour ce cas d'utilisation.



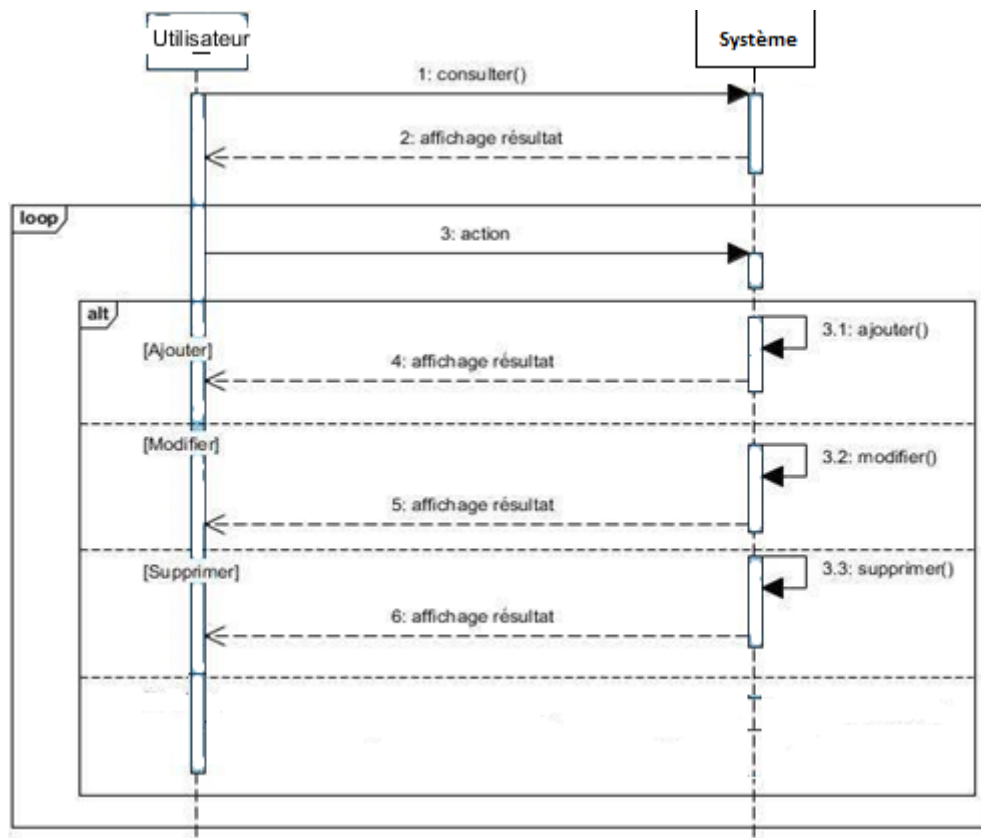
5.6.2.1. DSS du cas d'utilisation : « Mettre en favoris un produit »

La Figure 18 nous montre le diagramme de séquence système pour ce cas d'utilisation.



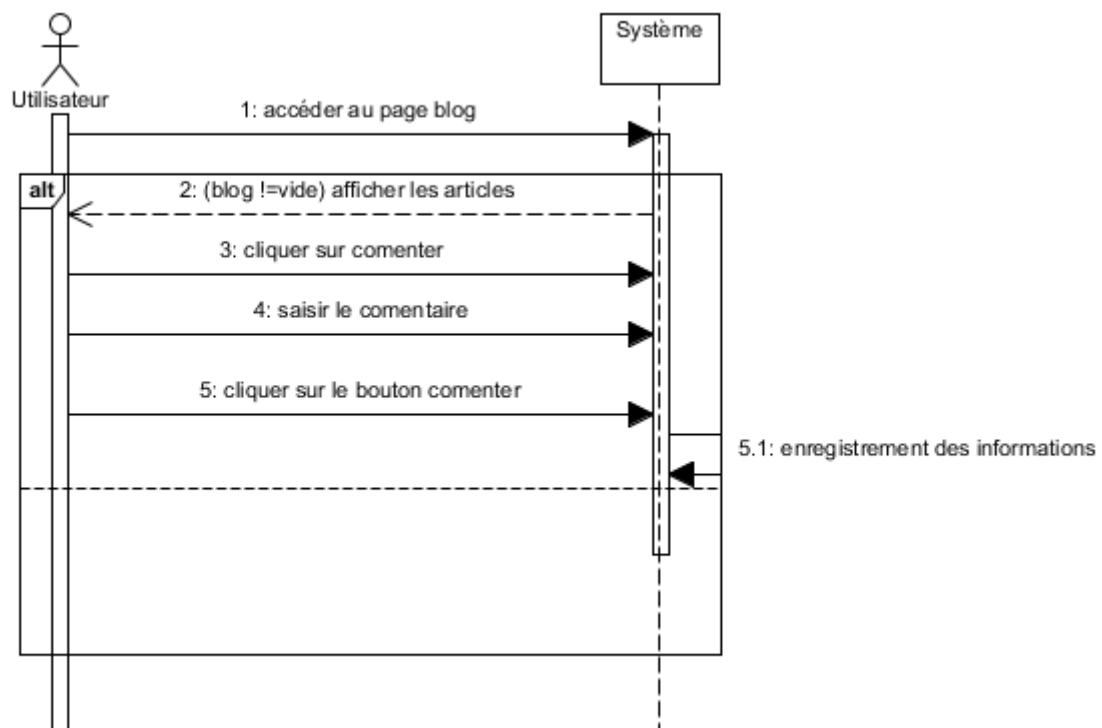
5.6.2.1. DSS du cas d'utilisation : « Gestion du blog »

La Figure 18 nous montre le diagramme de séquence système pour ce cas d'utilisation.



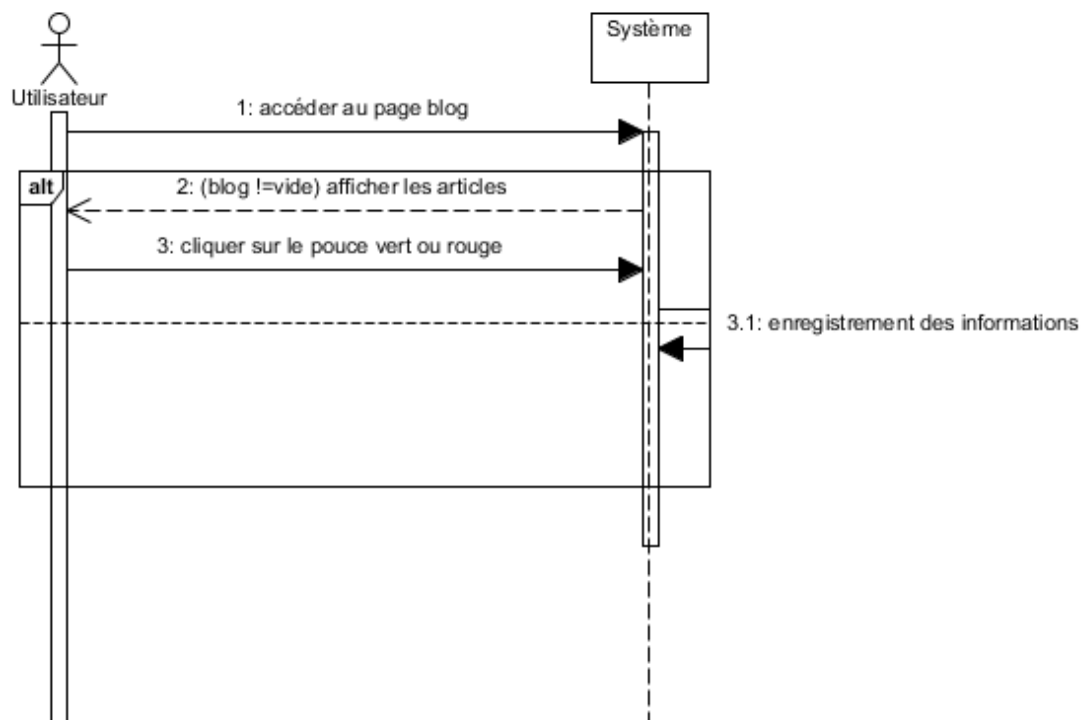
5.6.2.1. DSS du cas d'utilisation : « Commenter Blog »

La Figure 18 nous montre le diagramme de séquence système pour ce cas d'utilisation.



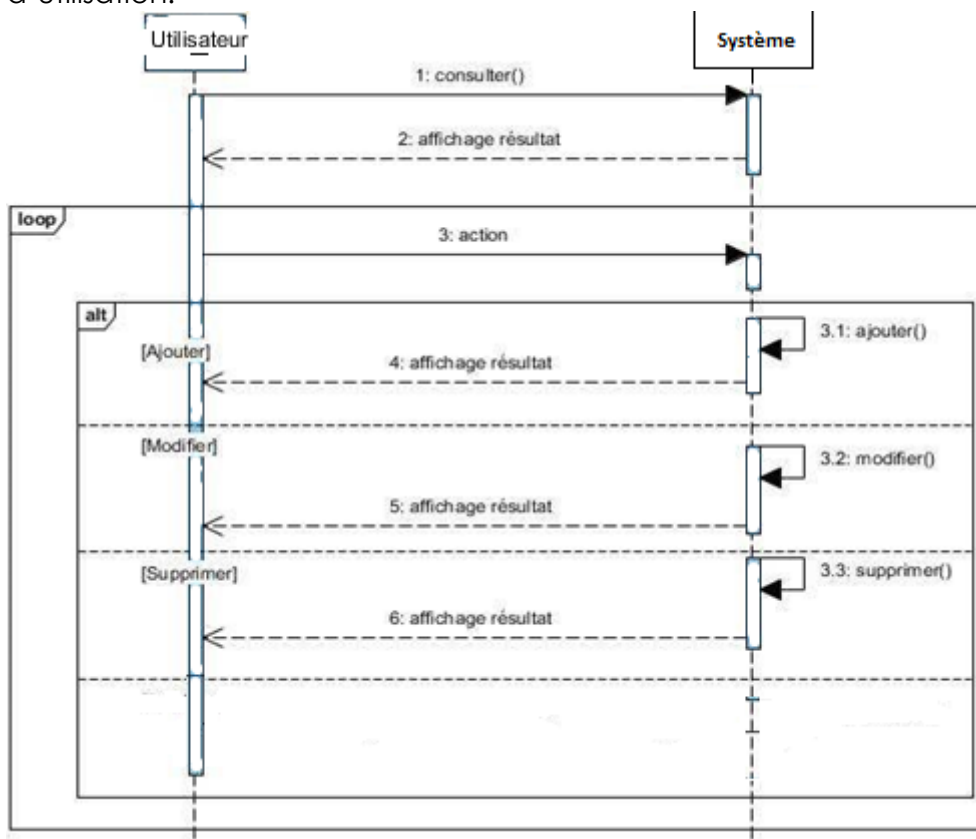
5.6.2.1. DSS du cas d'utilisation : « Mettre un pouce vert ou rouge une blog »

La Figure 18 nous montre le diagramme de séquence système pour ce cas d'utilisation.



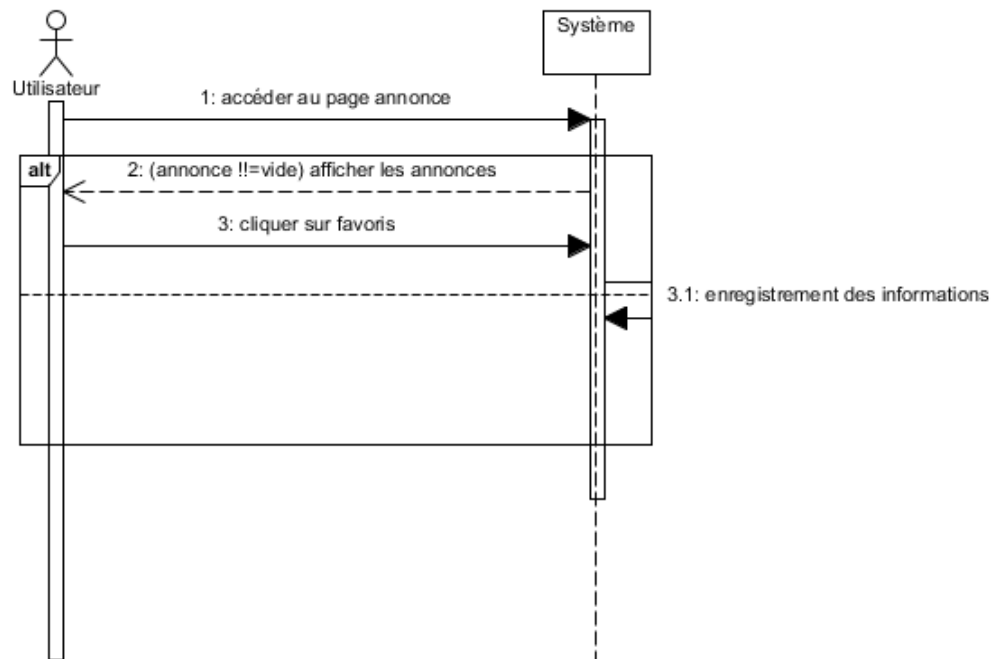
5.6.2.1. DSS du cas d'utilisation : « Gestion d'annonce »

La Figure 18 nous montre le diagramme de séquence système pour ce cas d'utilisation.



5.6.2.1. DSS du cas d'utilisation : « Mettre en favoris une annonce »

La Figure 18 nous montre le diagramme de séquence système pour ce cas d'utilisation.



Chapitre 6 CONCEPTION DETAILLEE

6.1. Architecture du système

Vue les nombreux avantages du MVC, il est dans l'intérêt du projet de suivre cette architecture. En plus, le Framework utilisé, Angular, adopte cette architecture.

C'est un modèle de conception de logiciel construit autour de l'interconnexion des trois principaux composants (modèles, vues et contrôleurs) et cible surtout la Programmation Orientée Objet

- Le modèle :

Il représente le cœur (algorithmique) de l'application : traitements des données, interactions avec la base de données, etc. Il décrit les données manipulées par l'application. Il regroupe la gestion de ces données et est responsable de leur intégrité. La base de données sera l'un de ses composants. Le modèle comporte des méthodes standards pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données

- La vue :

C'est sur quoi l'utilisateur agit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toute action de l'utilisateur. Ces différents événements sont envoyés au contrôleur. La vue n'effectue pas de traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.

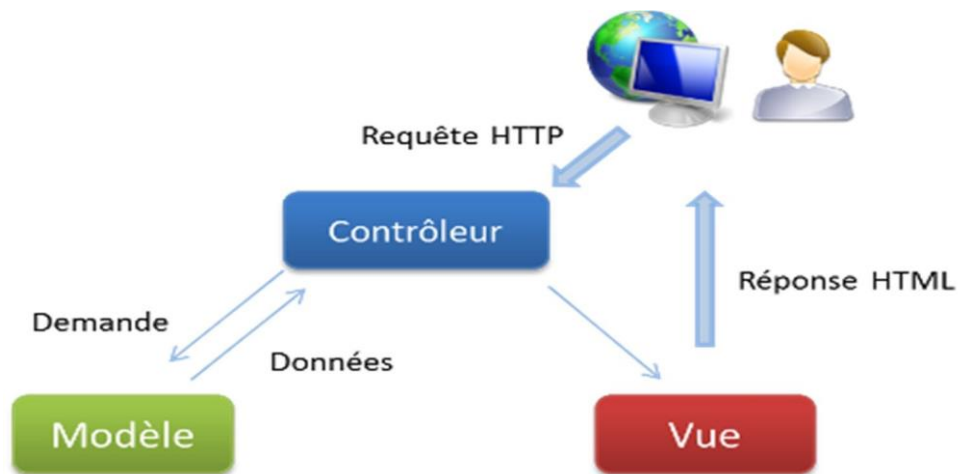
- Le contrôleur :

Il prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de la vue et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle afin que les données affichées se mettent à jour.

En résumé, lorsqu'un client envoie une requête à l'application :

- la requête envoyée depuis la vue est analysée par le contrôleur (par exemple un clic de souris pour lancer un traitement de données) ;
- le contrôleur demande au modèle approprié d'effectuer les traitements et notifie à la vue que la requête est traitée ;
- la vue notifiée fait une requête au modèle pour se mettre à jour (par exemple affiche le résultat du traitement via le modèle). [9]

La figure 28 illustre l'architecture MVC



6.2. Diagramme de séquence de conception pour chaque cas d'utilisation

Les diagrammes de séquence présentent la coopération entre différents objets. Les objets sont définis et leur coopération est représentée par une séquence de messages entre eux.

Les objets peuvent être connectés à des classes existantes ou bien être créés indépendamment de toute classe. Si les objets sont connectés à des classes, les messages peuvent être connectés à des opérations.

Les diagrammes de séquence sont créés dans les interactions.

Ils sont, en globalité, composés de lignes de vie qui interagissent entre elles par des messages.

6.2.1. Les lignes de vie

Une ligne de vie se représente par un rectangle, auquel est accrochée une ligne verticale pointillée, contenant une étiquette dont la syntaxe est :

[<nom_du_rôle>] :

[<Nom_du_type>]

Au moins un des deux noms doit être spécifié dans l'étiquette, les deux points (:) sont, quant à eux, obligatoires.

6.2.2. Les messages

Un message définit une communication particulière entre des lignes de vie. Plusieurs types de messages existent, les plus communs sont :

- L'envoi d'un signal ;
- L'invocation d'une opération ;
- La création ou la destruction d'une instance.

6.2.2.1. Messages asynchrones

Une interruption ou un événement sont de bons exemples de signaux. Ils n'attendent pas de réponse et ne bloquent pas l'émetteur qui ne sait pas si le message arrivera à destination, le cas échéant quand il arrivera et s'il sera traité par le destinataire. Un signal est, par définition, un message asynchrone.

Graphiquement, un message asynchrone se représente par une flèche en traits pleins et à l'extrémité ouverte partant de la ligne de vie d'un objet expéditeur et allant vers celle de l'objet cible comme le montre la figure 11.



Figure 11 : Exemple de messages asynchrones

6.2.2.2. Messages synchrones

L'invocation d'une opération est le type de message le plus utilisé en programmation objet. L'invocation peut être asynchrone ou synchrone. Dans la pratique, la plupart des invocations sont synchrones, l'émetteur reste alors bloqué le temps que dure l'invocation de l'opération.

Graphiquement, un message synchrone se représente par une flèche en traits pleins et à l'extrémité pleine partant de la ligne de vie d'un objet expéditeur et allant vers celle de l'objet cible. Ce message peut être suivi d'une réponse qui se représente par une flèche en pointillés.

Cette définition est expliquée par la figure 12.

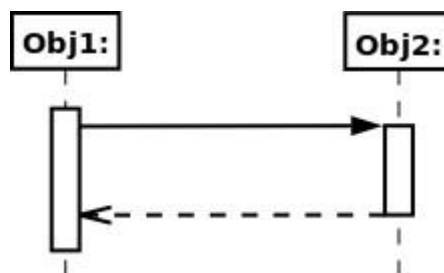


Figure 12 : Exemple de messages synchrones

6.2.2.3. Messages de création et de destruction d'instance

La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie

La destruction d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet. La destruction d'un objet n'est pas nécessairement consécutive à la réception d'un message.

Cette rubrique est accompagnée par la figure 13.

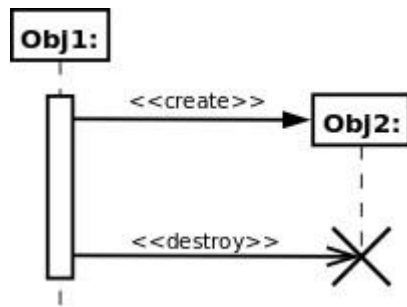


Figure 13 : Exemple de messages de création et de destruction d'instance

6.2.3. Les fragments d'interaction combinés

Un fragment combiné représente des articulations d'interactions. Il est défini par un opérateur et des opérandes. L'opérateur conditionne la signification du fragment combiné. Il existe 12 d'opérateurs définis dans la notation UML 2.0. Les fragments combinés permettent de décrire des diagrammes de séquence de manière compacte. Les fragments combinés peuvent faire intervenir l'ensemble des entités participant au scénario ou juste un sous-ensemble.

Un fragment combiné se représente de la même façon qu'une interaction. Il est représenté dans un rectangle dont le coin supérieur gauche contient un pentagone. Dans le pentagone figure le type de la combinaison, appelé opérateur d'interaction. Les opérandes d'un opérateur d'interaction sont séparés par une ligne pointillée. Les conditions de choix des opérandes sont données par des expressions booléennes entre crochets ([]).

La liste suivante regroupe les opérateurs d'interaction par fonctions :

- les opérateurs de choix et de boucle : alternative, option, break et loop ;
- les opérateurs contrôlant l'envoi en parallèle de messages : parallel et critical region ;
- les opérateurs contrôlant l'envoi de messages : ignore, consider, assertion et negative ;
- les opérateurs fixant l'ordre d'envoi des messages : weak sequencing, strict sequencing.

Par rapport aux diagrammes de séquence système, nous allons remplacer le système

vu comme une boîte noire par un ensemble d'objets en interaction. Pour cela, nous utiliserons encore dans ce chapitre les trois types de classes d'analyse, à

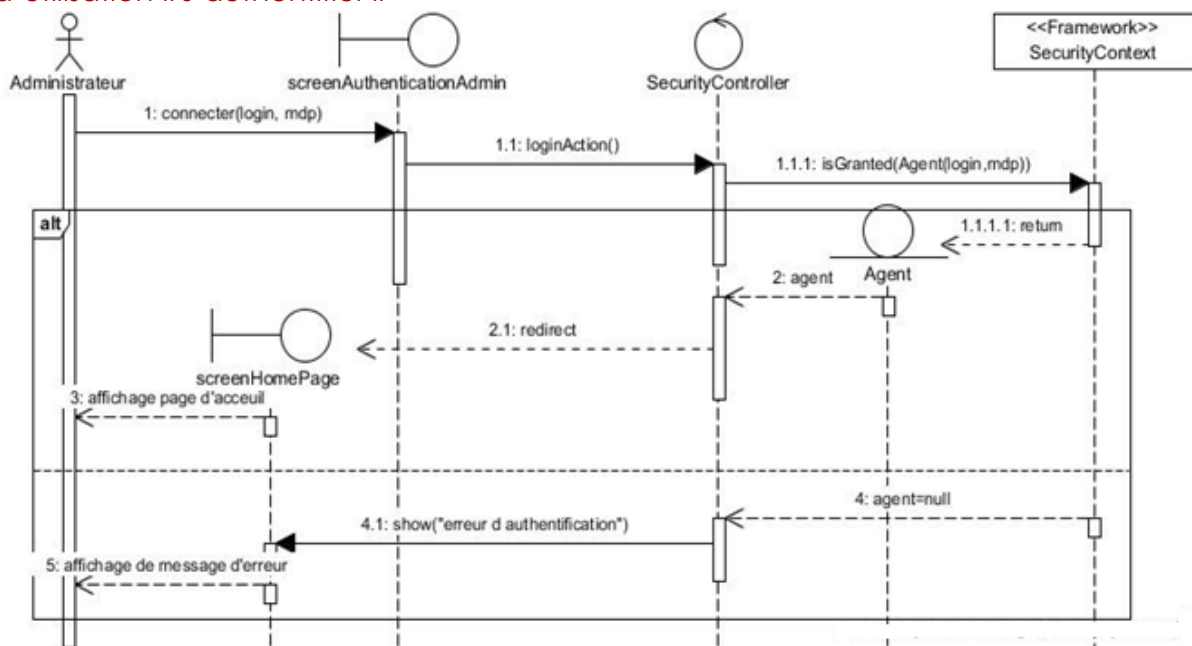
savoir les dialogues, les contrôles et les entités. Les règles d'interaction entre ces classes sont les suivantes :

- Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues.
- Les dialogues peuvent interagir avec les contrôles.
- Les contrôles peuvent interagir avec les dialogues, les entités, ou d'autres contrôles.
- Les entités ne peuvent interagir qu'entre elles.

Dans ce paragraphe, nous allons mettre en évidence les diagrammes de séquence de conception (**DSC**) pour chaque cas d'utilisation.

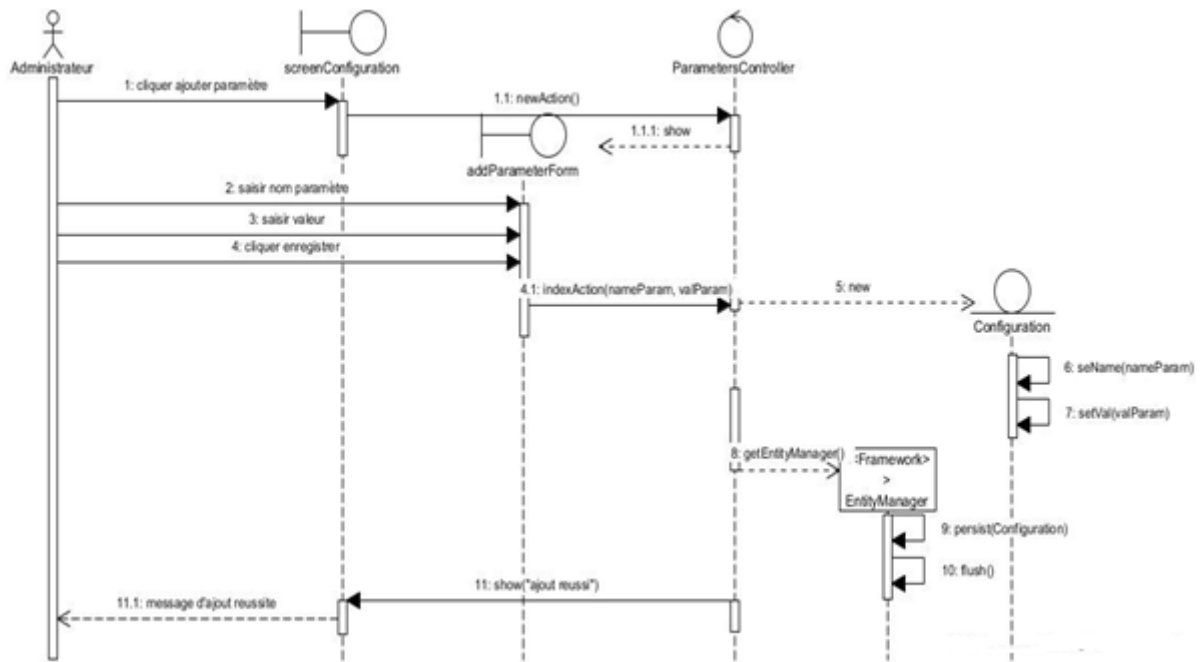
6.2.1. DSC du cas d'utilisation : « s'authentifier »

La figure 29 représente le diagramme de séquence de conception pour le cas d'utilisation « s'authentifier »



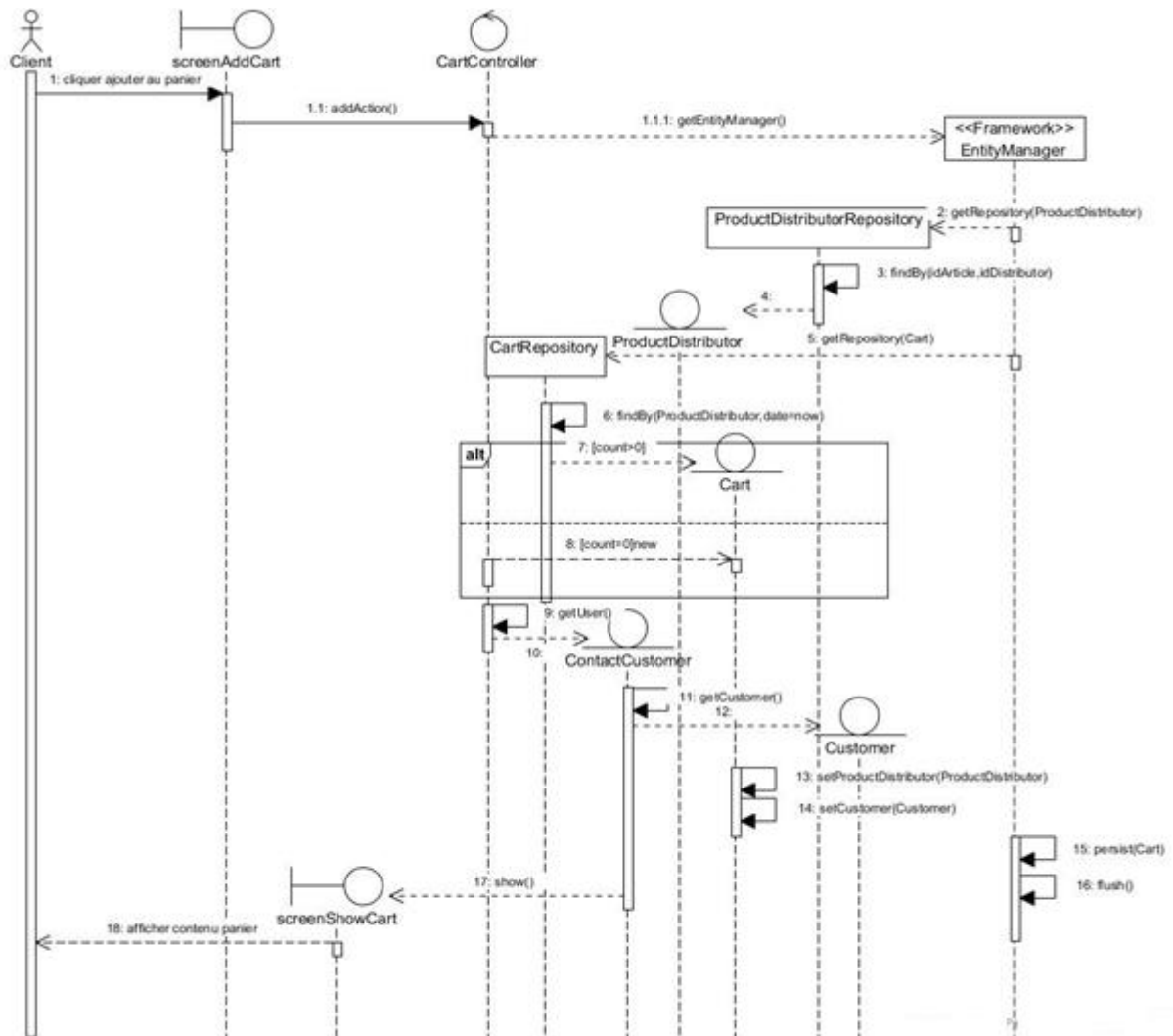
1.1.1. DSC du cas d'utilisation « paramétrage du site »

La figure 33 montre le diagramme de séquence de conception de ce cas d'utilisation

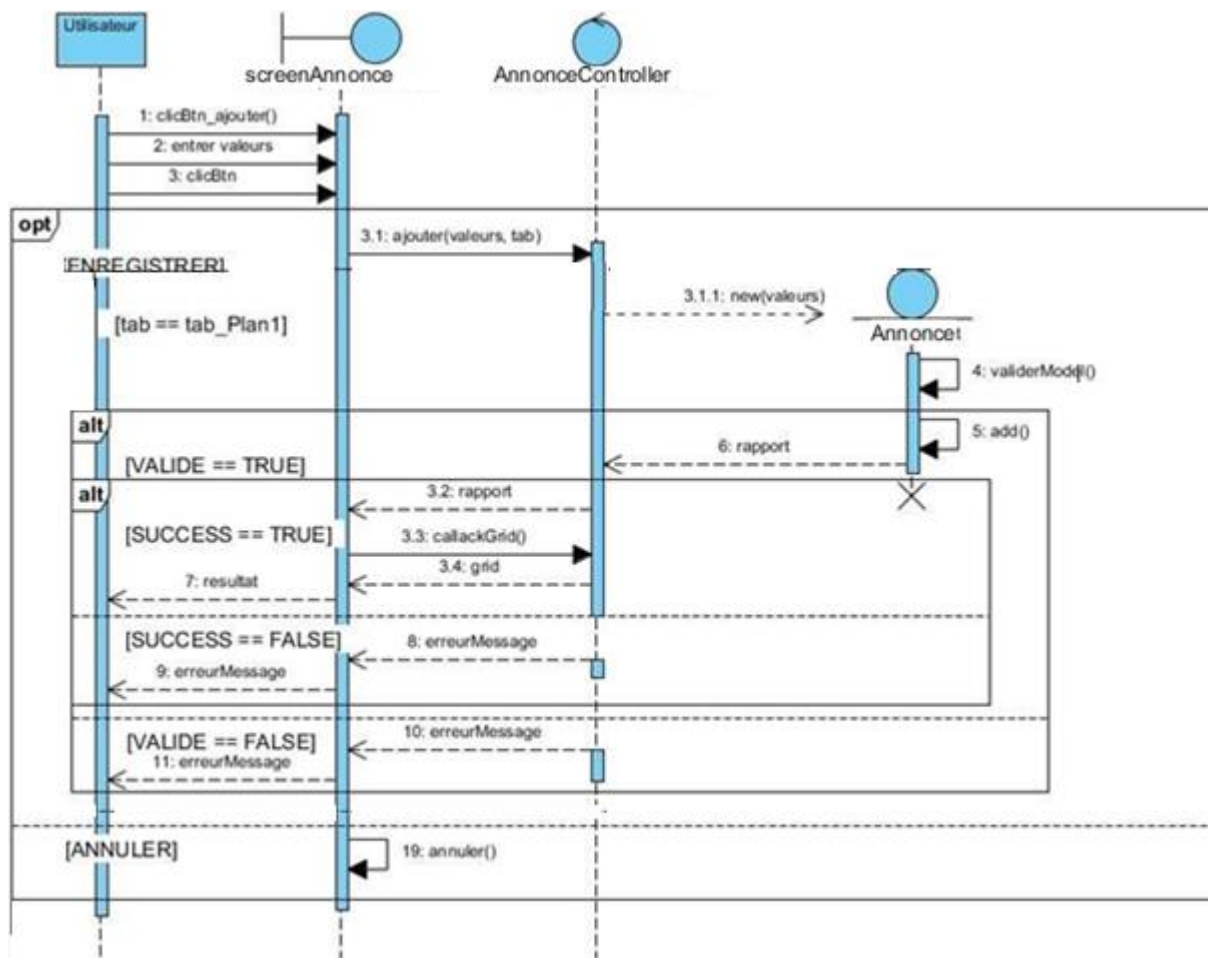


1.1.2. DSC du cas d'utilisation « ajouter un produit dans le favoris »

La figure 33 montre le diagramme de séquence de conception de ce cas d'utilisation



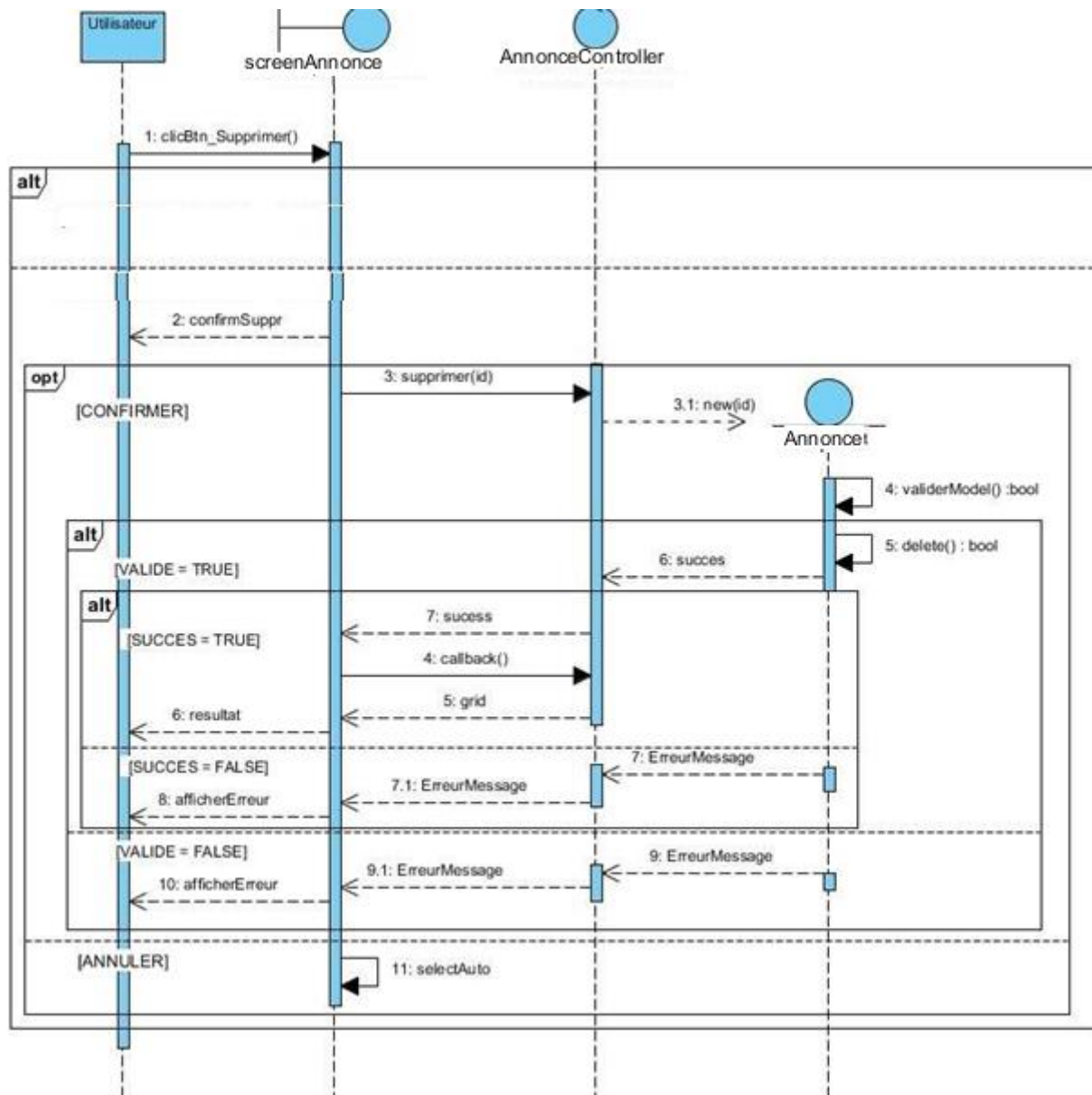
1.1.1. DSC du cas d'utilisation « ajouter une annonce »



1.1.1.

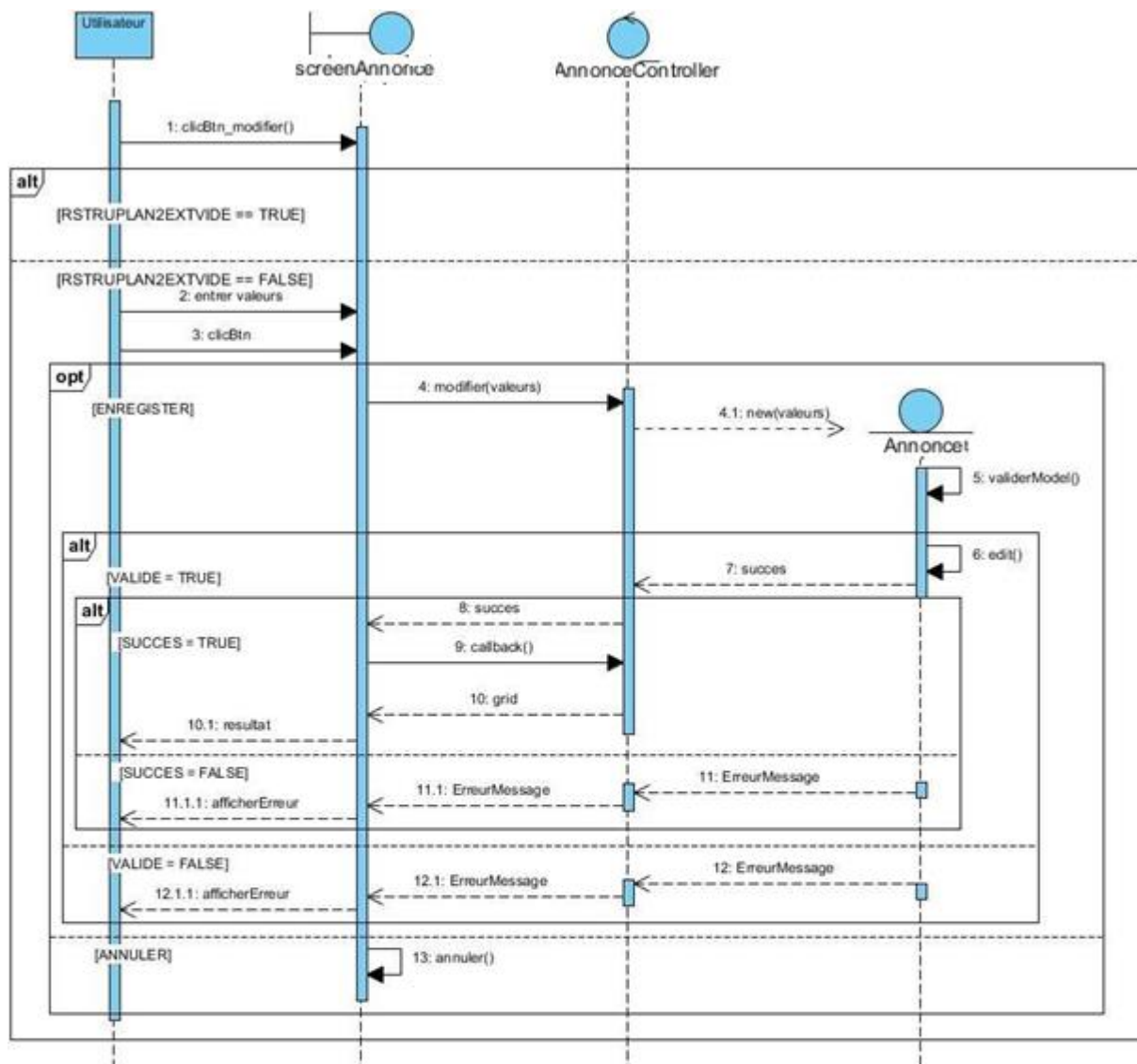
»

DSC du cas d'utilisation « supprimer une annonce



1.1.1.

DSC du cas d'utilisation « modifier une annonce »



1.2. Diagramme de classes de conception pour chaque cas d'utilisation

1.3. Généralité

Le diagramme des classes identifie la structure des classes d'un système, y compris les propriétés et les méthodes de chaque classe. Les

diverses relations, telles que la relation d'héritage par exemple, qui peuvent exister entre les classes y sont également représentées.

6.3.2. Représentation

Les éléments d'un diagramme des classes sont les classes et les relations qui les lient.

- **Les classes :**

Les classes sont les modules de base de la programmation orientée objet. Une classe est représentée en utilisant un rectangle divisé en trois sections. La section supérieure est le nom de la classe. La section centrale définit les propriétés de la classe. La section du bas énumère les méthodes de la classe comme illustrée dans la figure 16.

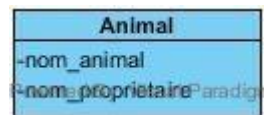


Figure 16 : Exemple d'une classe

- **Association :**

Une association est une relation générique entre deux classes. Elle est modélisée par une ligne reliant les deux classes. Cette ligne peut être qualifiée avec le type de relation, et peut également comporter des règles de multiplicité (par exemple un à un, un à plusieurs, plusieurs à plusieurs) pour la relation. Nous pouvons voir un exemple de cette association dans la figure 17.

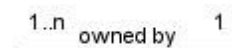


Figure 17 : Exemple d'une association

- **Composition :**

Si une classe ne peut pas exister par elle-même, mais doit être un membre d'une autre classe, alors elle possède une relation de composition avec la classe contenant. Une relation de composition est indiquée par une ligne avec un "diamant" rempli comme la montre la figure 18.



Figure 18 : Composition

- **Dépendance :**

Quand une classe utilise une autre classe, par exemple comme membre ou comme paramètre d'une de ces fonctions, elle "dépend"

ainsi de cette classe. Une relation de dépendance est représentée par une flèche pointillée. Cette représentation est accompagnée par la figure 19.



Figure 19 : Dépendance

- **Agrégation :**

Les agrégations indiquent une relation de contenant contenu.

Elle est décrite par une relation "possède". Une relation d'agrégation est représentée par une ligne avec un "diamant" creux comme avec la figure 20.



Figure 20 : Agrégation

- **Généralisation :**

Une relation de généralisation est l'équivalent d'une relation d'héritage en terme orientés objet (relation "est-un"). Une relation de généralisation est indiquée par une flèche creuse se dirigeant vers la classe "parent" comme le montre la figure 21.

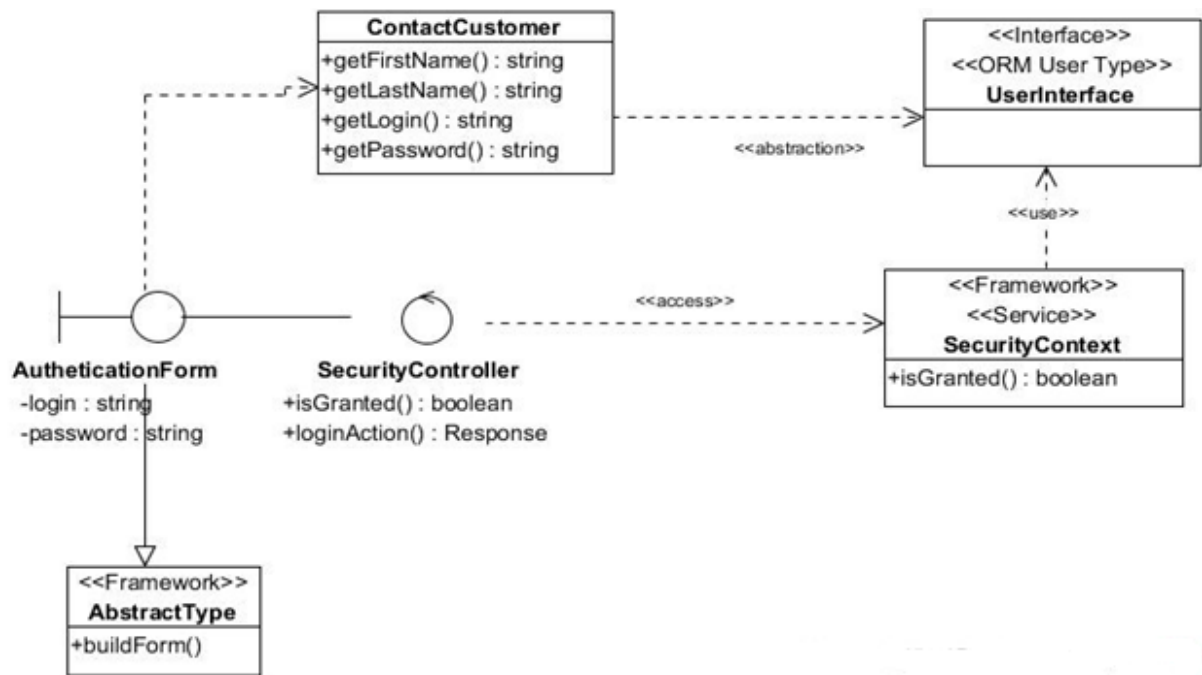


Figure 21 : Généralisation

Nous allons présenter les diagrammes de classe de conception correspondants à quelque cas d'utilisation.

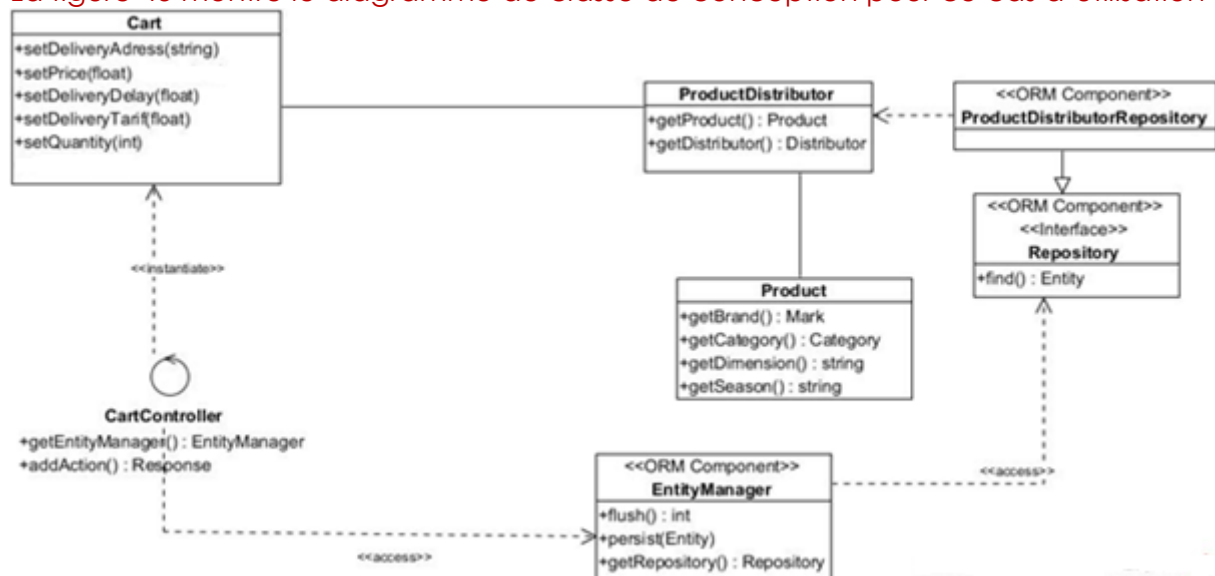
1.3.1. Diagramme de classe de conception pour le cas d'utilisation « s'authentifier »

La figure 40 montre le diagramme de classe de conception pour le cas d'utilisation « s'authentifier »



1.3.2. Diagramme de classe de conception pour le cas d'utilisation « ajouter un produit dans le favoris »

La figure 45 montre le diagramme de classe de conception pour ce cas d'utilisation



2. Diagramme de classe pour Gérer l'annonce

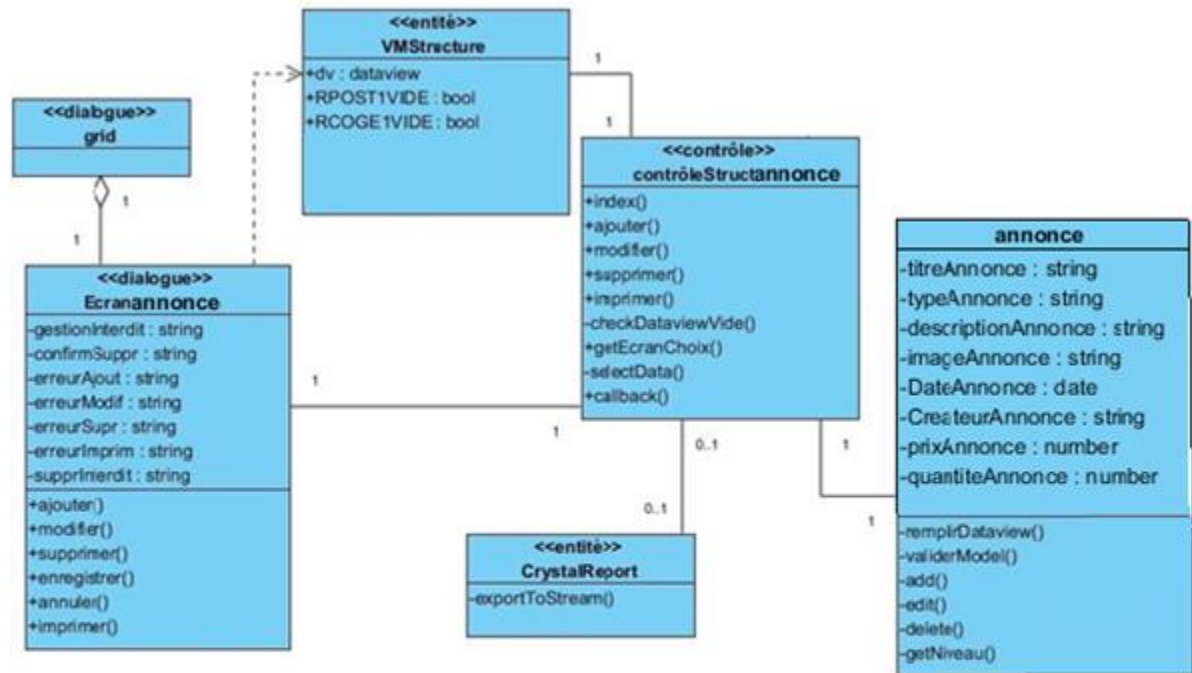


Diagramme Ger de l'application :

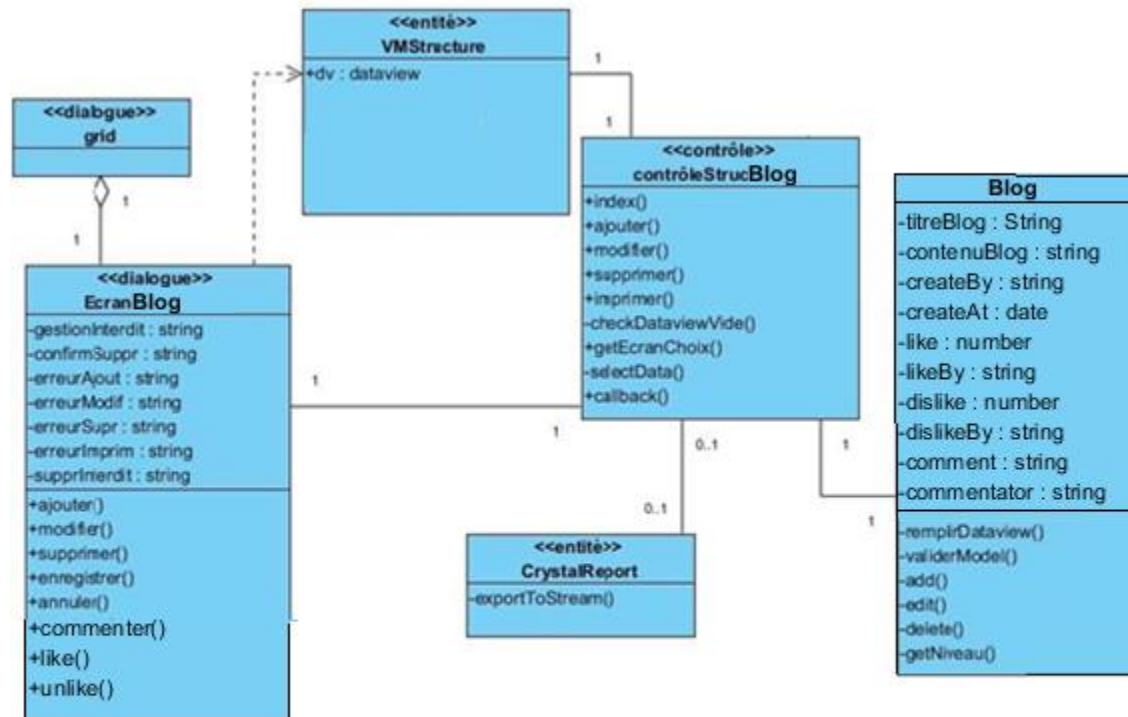
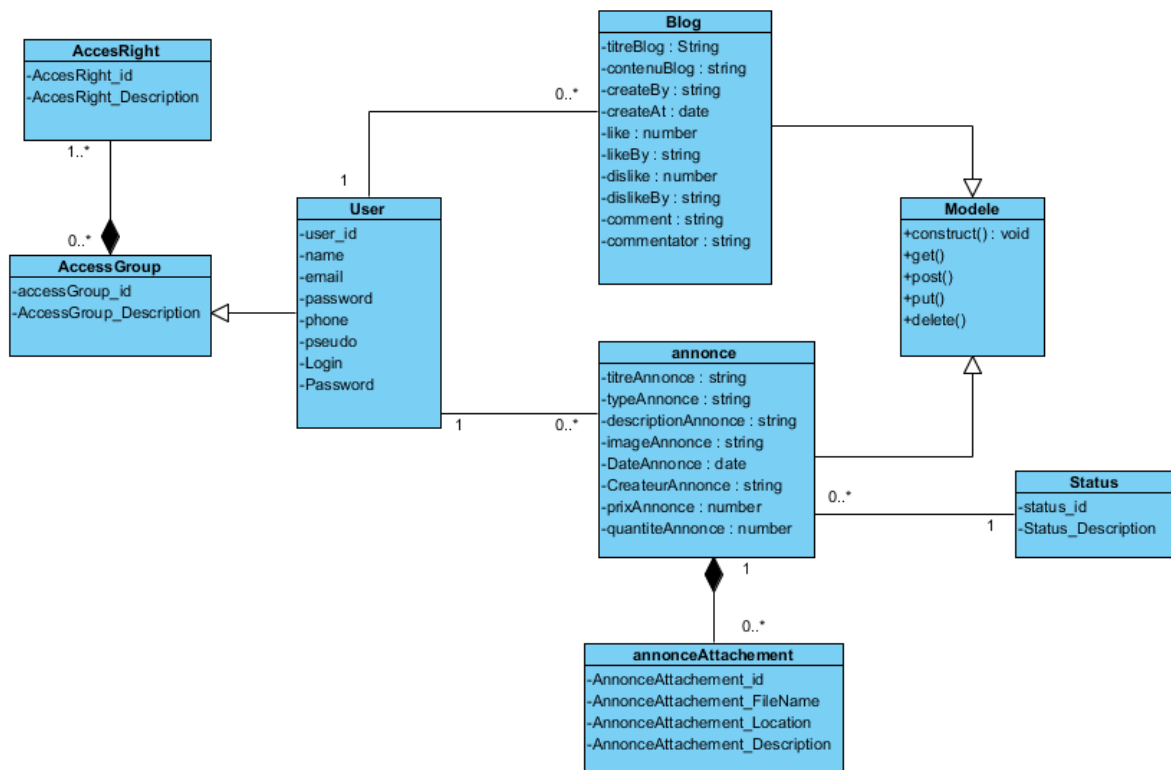


Diagramme global de l'application



2.1. Diagramme de paquetages

Le diagramme de paquetages montre l'organisation logique du modèle et les relations entre packages.

- Package (ou paquetage) : mécanisme général de regroupement d'éléments tels que classes, interfaces, mais aussi acteurs, cas d'utilisation, etc. Les packages peuvent être imbriqués dans d'autres packages.
- Importation : relation de dépendance entre packages qui rend visibles les éléments publics de l'un des packages au sein d'un autre.

Les packages sont des espaces de noms (namespace) : deux éléments ne peuvent pas porter le même nom au sein du même package. En revanche, deux éléments appartenant à des packages différents peuvent porter le même nom.

Ainsi, en prenant compte de la technologie utilisée et de l'architecture du projet, les package du système sont découpés comme suite :

- *src* : qui contient le code source du projet
- *app* : qui est le package contenant les codes sources gérant le côté « Front Office »
- *Server* : qui contient les codes sources gérant tous ceux qui est « Back Office »
- *View* : contenant les couches présentation de l'application
- *service* : qui contient les classes « services »
- *node_modules* : contenant toutes les plugings et les bibliothèques du back office et du Front office

- assets : qui contient les images de l'application
- models : contenant les entités de l'application
- uploads : contenant toutes les images uploadés

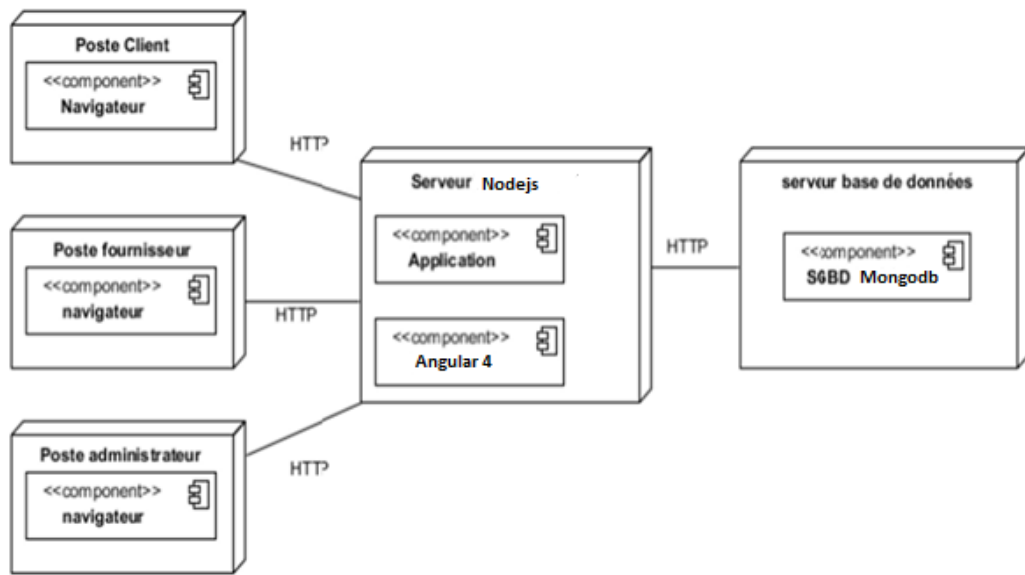
2.2. Diagramme de déploiement

En UML, un diagramme de déploiement est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux. Les éléments utilisés par un diagramme de déploiement sont principalement les nœuds, les composants, les associations et les artefacts.

Les caractéristiques des ressources matérielles physiques et des supports de communication peuvent être précisées par stéréotype.

- **Artefacts** : c'est la spécification d'une partie physique utilisée ou produite lors du processus du développement du logiciel (comme les fichiers .exe, .dll, .xml, etc.).
- **Nœuds** : c'est une ressource d'exécution sur laquelle les artefacts peuvent être déployés en vue d'être exécutés.
- **Manifestations** : c'est une relation qui montre qu'un élément du modèle est incorporé dans un artefact. Si un artefact est la représentation physique d'un composant, il constitue la manifestation du composant.
- **Chemin de communications** : c'est une association entre deux nœuds au travers de laquelle les nœuds peuvent communiquer par l'échange de messages et de signaux. On peut aussi faire figurer des chemins de communication entre des nœuds d'environnements d'exécution : on obtient ainsi des représentations plus précises qu'avec des liens entre les nœuds.
- **Spécifications de déploiement** : spécifie un ensemble de propriétés qui déterminent les paramètres d'exécution d'un artefact déployé sur un nœud.

La Figure 49 représente notre diagramme de déploiement pour le projet.



Chapitre 7 MISE EN PLACE DE L'ENVIRONNEMENT DE DEVELOPPEMENT

3.1. Installation et configuration des outils

3.1.1. Visual Paradigm

La phase de conception et de modélisation est une phase indispensable dans le processus de développement d'un logiciel ; toute la logique et les contraintes y sont décrites : c'est la base pour produire un logiciel de qualité.

Nous avons ainsi, utilisé Visual Paradigm for UML 2.0 pour la représentation de quelque modèle représentant le système.

Cet outil permet :

- La modélisation UML 2.0 qui inclut ses 13 diagrammes
- La génération de code de programmation dans une bonne partie des langages communs, comme Java, C#, VB.NET, PHP, ODL, ActionScript, IDL, C++, Delphi, Perl, XML Schema, Python, Objective-C, Objective-C 2.0, Ada95 et Ruby
- La modélisation de bases de données relationnelles
- La génération de code SQL et le déploiement dans les principaux SGBDR, à savoir MySQL, MS SQL Server, Oracle, HSQL, Sybase ASE, Sybase SQL Anywhere, PostgreSQL, Cloudscape-Derby, DB2, Ingres, OpenEdge, Informix, Firebird, FrontBase, Cache, SQLite et H2
- Le « reverse engineering », qui consiste en la création automatique de modèles depuis du code de programmation ou depuis une base de données existante
- La gestion des exigences
- L'analyse d'impacts, qui permet de connaître à l'avance les conséquences d'un changement
- La création de rapports automatisés

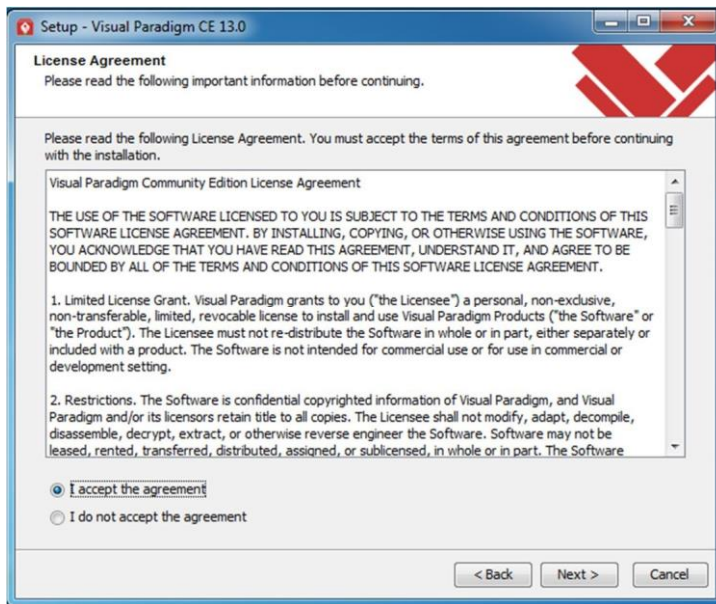
7.1.1.1. Etape d'installation de Visual Paradigm Community Edition

- ☐ Etape 1 : exécution du programme d'installation

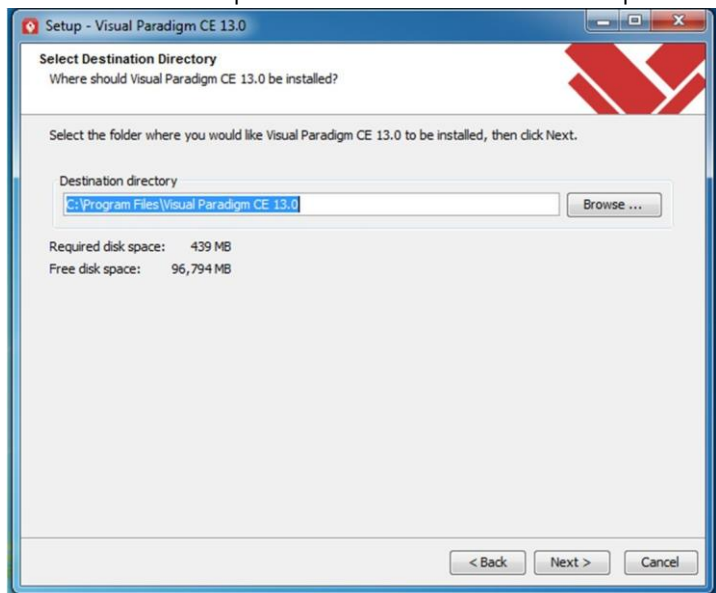


Visual_Paradigm_CE_13_0_20160304_Win64.exe

- ☐ Etape 2 : acceptation du terme de licence de Visual Paradigm et clique sur « Next »

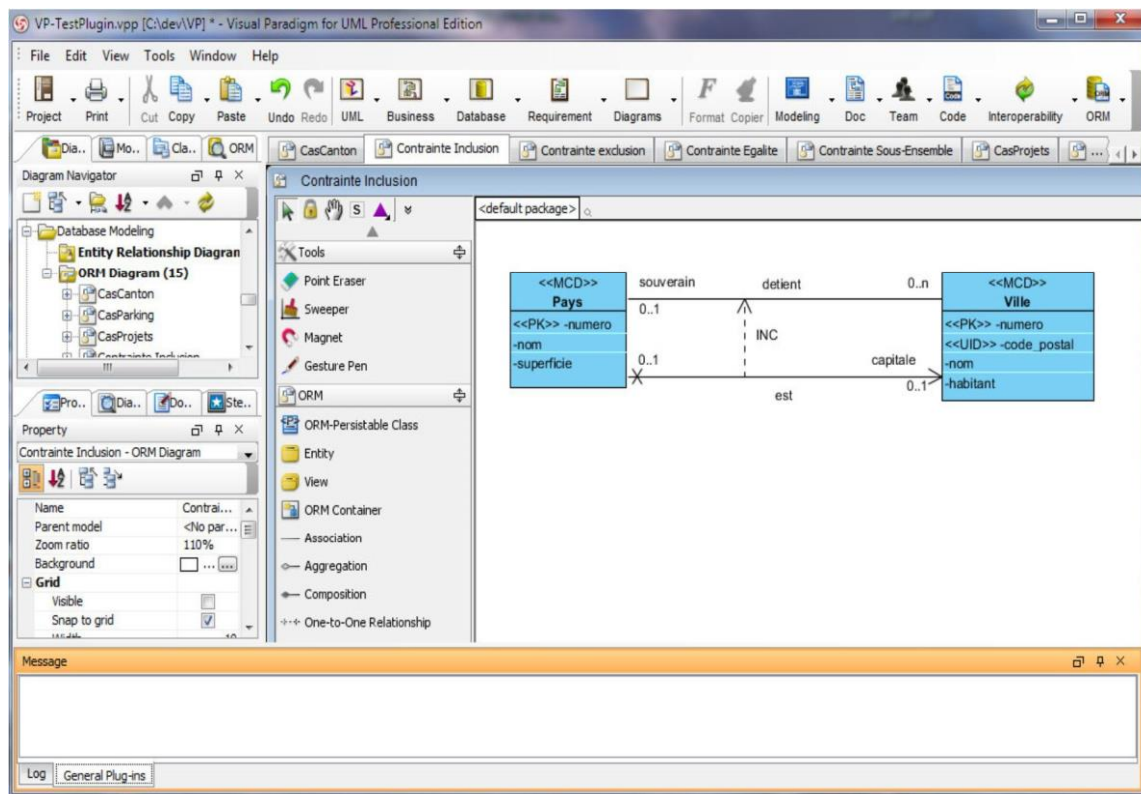


- Etape 3 : choix du répertoire d'installation et clique sur « Next »



7.1.1.2. Représentation de l'interface de Visual Paradigm

La figure 53 représente l'interface de Visual Paradigm avec ses grandes zones principales.



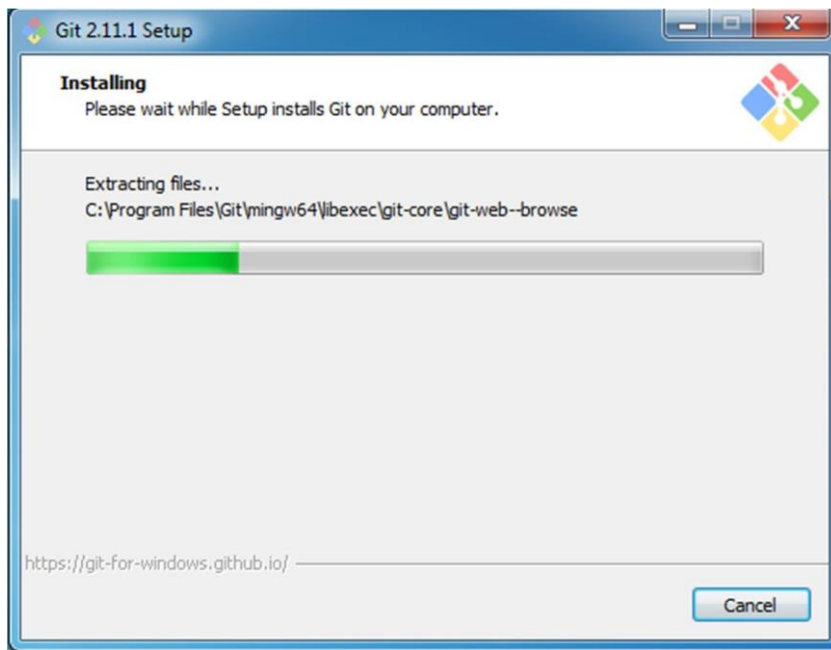
3.1.2. Git

Les dépôts ou « repository » pour la gestion des versions du projet sont hébergés sur Git. Git est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

Afin d'accéder aux dépôts du projet, il faut avoir un client Git installé sur la machine. Il est téléchargeable sur son site officiel, sur l'adresse <https://git-scm.com/>

Il suffit de télécharger la dernière version stable et lancer le programme d'installation en suivant les instructions. Durant l'installation, il peut être configuré d'intégrer Git à l'explorateur sous Windows. Cette option permet de lancer l'invite de commande Git depuis l'explorateur du Système d'exploitation de Microsoft ou d'effectuer certaines manipulations plus aisément.

La figure 54 montre l'installation de Git.

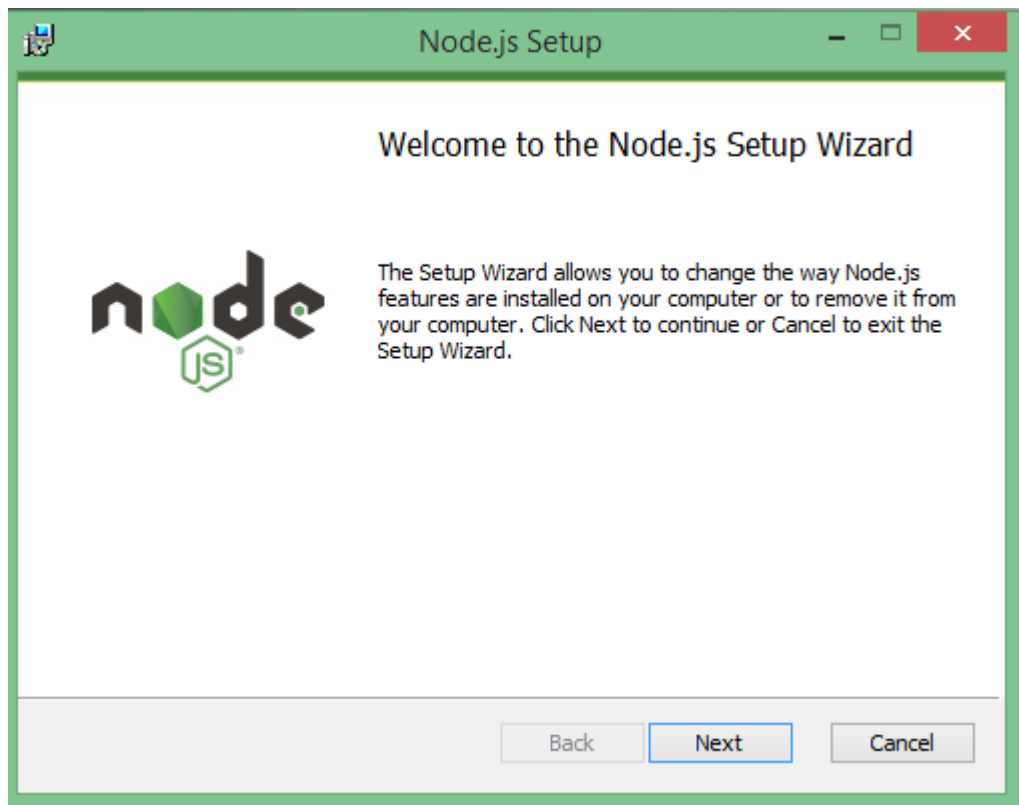


Comme Git suit le principe de « Convention over configuration », les conventions principales à retenir sont :

- **Repository** : la base qui stocke les fichiers
- **Server** : la machine qui contient le repo
- **Client** : la machine qui se connecte au repo
- Git dispose notamment des commandes suivantes (pour une liste complète, consultez la page de manuel Git) :
- **git init**, crée un nouveau dépôt ;
- **git clone**, clone un dépôt distant ;
- **git add**, ajoute de nouveaux objets blobs dans la base des objets pour chaque fichier modifié depuis le dernier commit. Les objets précédents restent inchangés ;
- **git commit**, intègre la somme de contrôle SHA-1 d'un objet tree et les sommes de contrôle des objets commits parents pour créer un nouvel objet commit ;
- **git branch**, crée une nouvelle branche de développement ;
- **git merge**, fusionne plusieurs branches de développement.

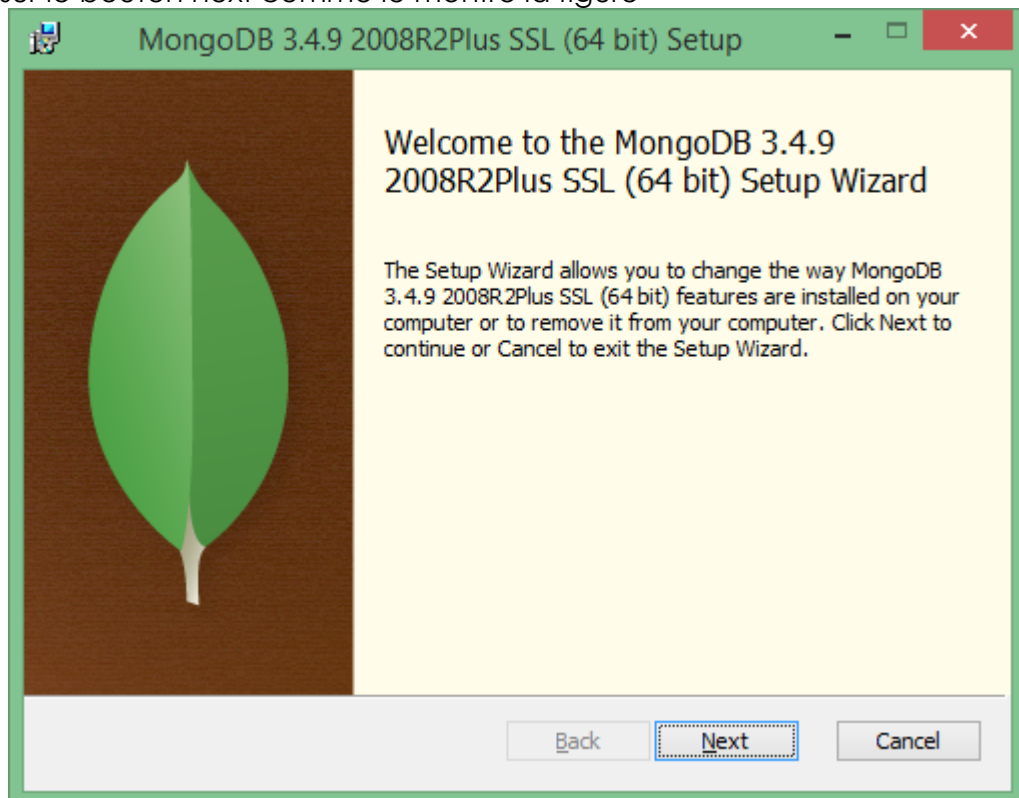
3.1.3. Installation nodejs

Pour pouvoir utiliser le produit node, il faut le télécharger sur son site officiel <https://www.nodejs.com/download> et d'exécuter le fichier téléchargé, l'installation débute et il nous accueille pour une fenêtre de bienvenue et on clique suivant sur le bouton next comme le montre la figure



3.1.4. Mongo

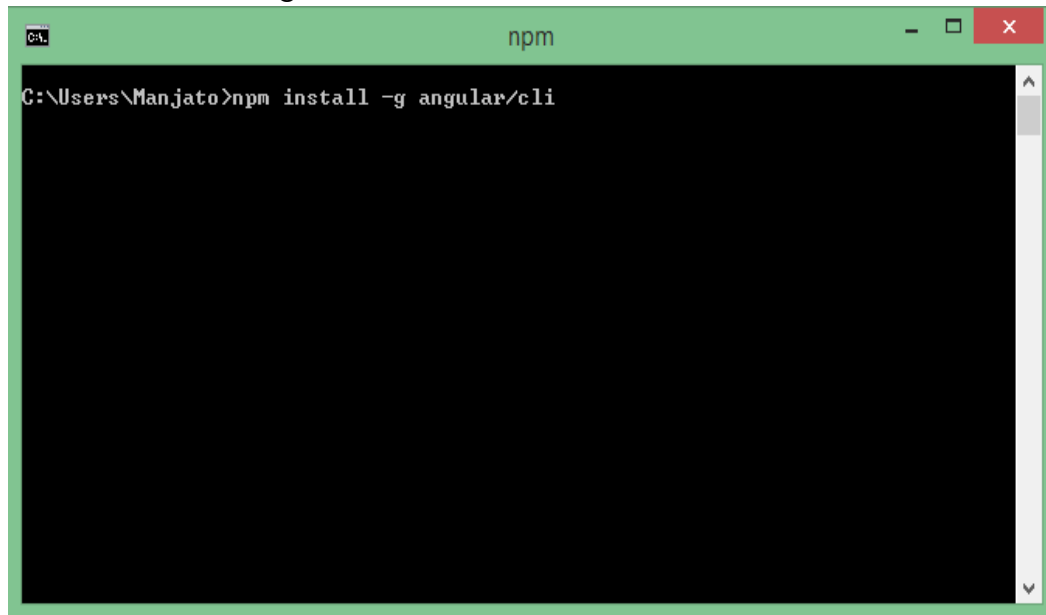
Pour pouvoir utiliser le produit node, il faut le télécharger sur son site officiel <http://www.mongodb.org/downloads> et d'exécuter le fichier téléchargé, l'installation débute et il nous accueille pour une fenêtre de bienvenue et on clique suivant sur le bouton next comme le montre la figure



Après l'installation, pour pouvoir utiliser mongodb il faut ouvrir la fenêtre de commande et d'écrire la commande : "C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe" --dbpath d:\mongo\db

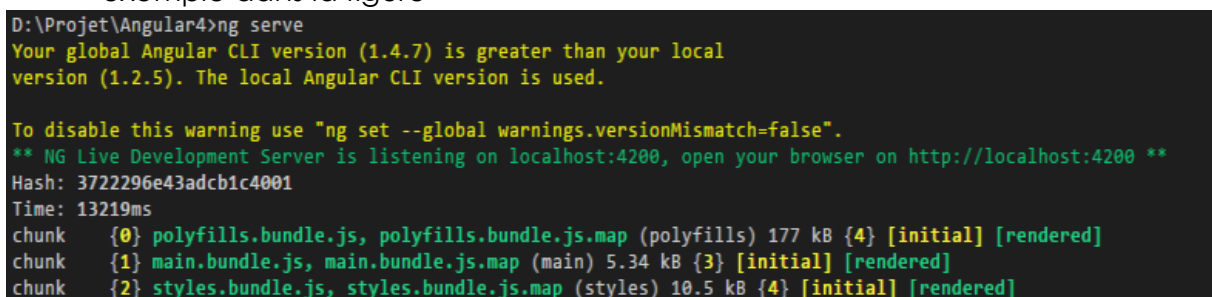
3.1.5. Installation angular

Avant de pouvoir installer Angular Cli, il faut, en prérequis, installer une version de [Node 4.x.x](#) minimum avec un NPM 3.x.x, il faut ouvrir la console et d'écrire la commande suivante : « npm install -g angular/cli » comme nous montre la figure



```
C:\Users\Manjato>npm install -g angular/cli
```

Angular-cli utilise l'exécutable nommé ng pour réaliser ses différentes fonctionnalités offertes. Le paramètre new indique que l'on va créer un nouveau projet qui est déclaré à la suite de : « ng g new monprojet », après l'installation et la configuration du projet « mon projet » il faut exécuter la commande suivante pour pouvoir lancer l'application : « ng serve » ou « ng serve port : {nombre} » parce que le port par défaut est : 4200. Comme par exemple dans la figure

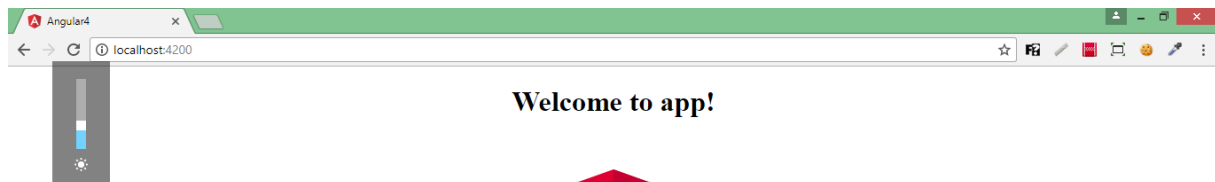


```
D:\Projet\Angular4>ng serve
Your global Angular CLI version (1.4.7) is greater than your local
version (1.2.5). The local Angular CLI version is used.

To disable this warning use "ng set --global warnings.versionMismatch=false".
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200 **
Hash: 3722296e43adcb1c4001
Time: 13219ms
chunk    {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 177 kB {4} [initial] [rendered]
chunk    {1} main.bundle.js, main.bundle.js.map (main) 5.34 kB {3} [initial] [rendered]
chunk    {2} styles.bundle.js, styles.bundle.js.map (styles) 10.5 kB {4} [initial] [rendered]
```

Angular-Cli va alors s'occuper de compiler l'ensemble du projet et de lancer un serveur web sur le port 4200. Vous pouvez maintenant lancer votre navigateur sur l'URL <http://localhost:4200> ; vous obtenez ainsi votre première application Angular :

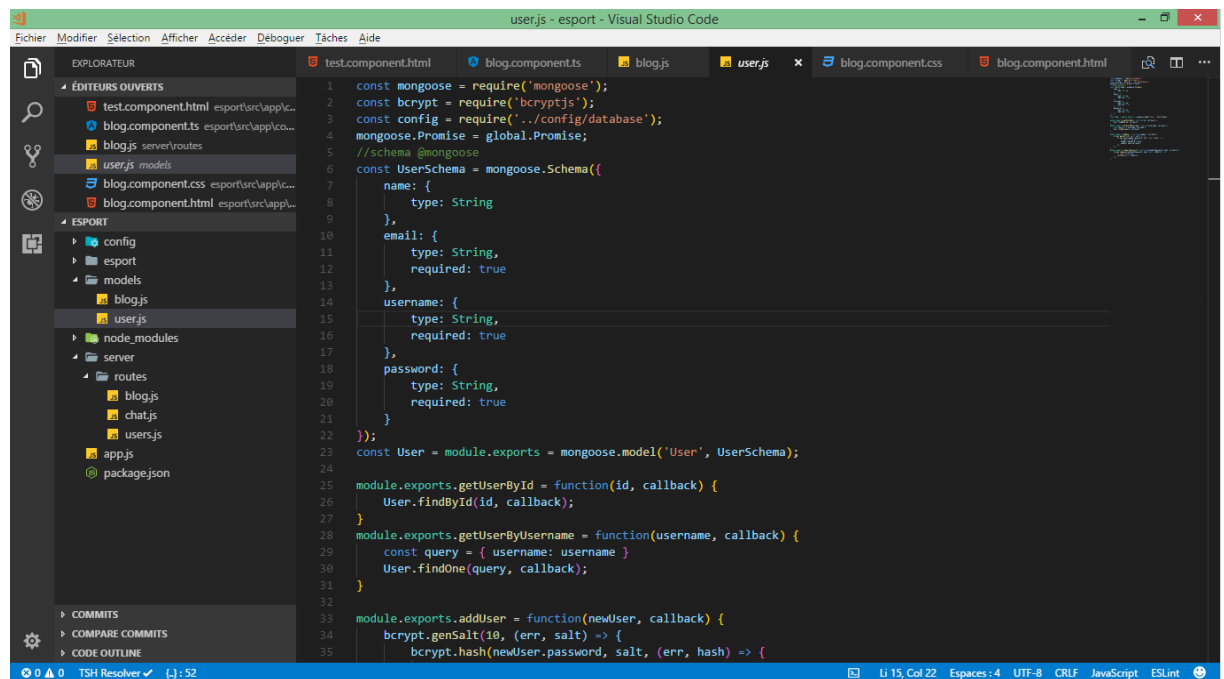
La figure nous montre la page d'accueil du projet angular 4



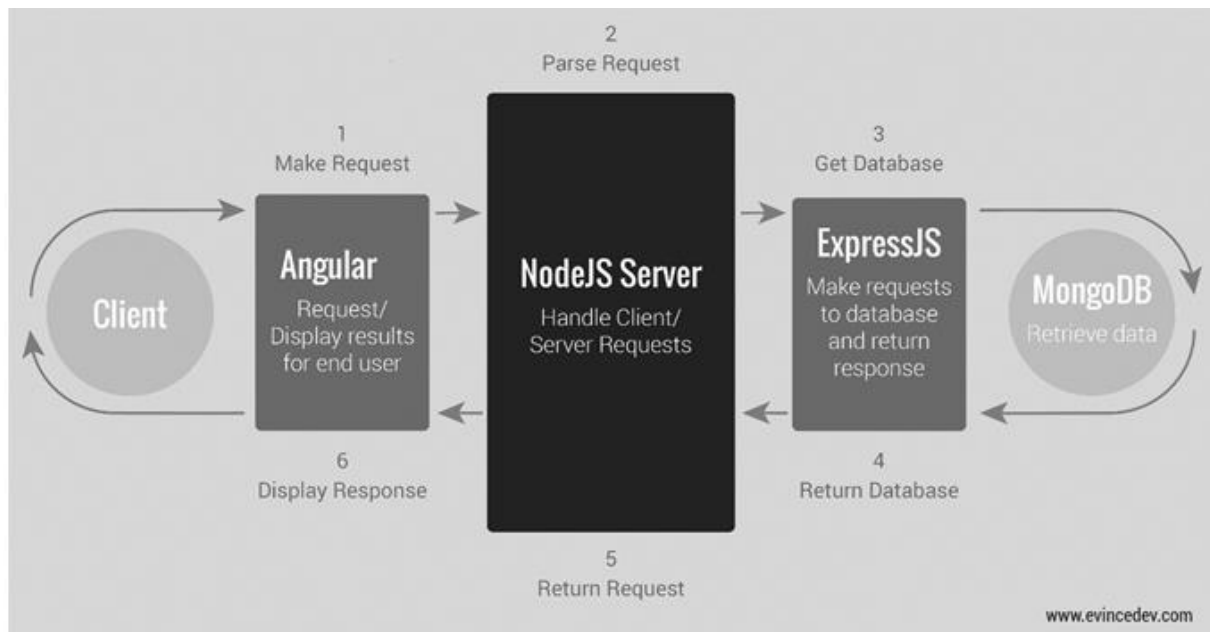
Here are some links to help you start:

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)

3.1.6. Visual Studio Code



3.2. Architecture de l'application



DEVELOPPEMENT DE L'APPLICATION

4.1. Configuration du projet

4.1.1. Création du serveur

On doit d'abord créer notre serveur avant de lancer dans le développement de l'application. Cette création de serveur se fait dans le fichier app.js.

On doit y renseigner toutes les informations comme connexion à la base de données, définir le port qu'on utilisera, la route.

La figure montre la partie de la création du serveur dans le fichier app.js

```
const bodyParser = require('body-parser');
const express = require('express');
const path = require('path');
const cors = require('cors');
const passport = require('passport');
const mongoose = require('mongoose');
const config = require('./config/database');

const app = express();
const port = 3000;

//connection mongodb 4.10.8
var MongoDB = mongoose.connect(config.nosql).connection;
//error
MongoDB.on('error', function(err) { console.log(err.message); });
MongoDB.once('open', function() {
  console.log('connection ' + config.nosql);
});
```

4.1.2. Génération des classes et entité

Voici l'entité User

```
const UserSchema = mongoose.Schema({
  name: {
    type: String
  },
  email: {
    type: String,
    required: true
  },
  username: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  }
});
```

4.2. Codage de l'application

Voir ci-dessous un fragment de code lors du codage de l'application. Puisque l'angular 4 qui est le front-end de l'application suit la norme MVC, alors le codage sera présenté ainsi.

Ajout d'un blog :

Vue :

```
<h1 class="page-header">Mon Blog</h1>
<div class="row show-hide-message">
  <div [ngClass]="messageClass">
    {{message}}
  </div>
</div>
<button type="button" name="button" class="btn btn-default" *ngIf="!newPost" (click)="newblog()">Nouveau</button>
<button [disabled]="loadingblog" type="button" name="button" class="btn btn-default" *ngIf="!newPost" (click)="
  <span class="glyphicon glyphicon-repeat"></span> Actualiser</button>
<br>
<br>
<!-- nouveau blog -->
<form name="blogForm" [formGroup]="form" (submit)="ajoutblog()" *ngIf="newPost">
  <!-- titre -->
  <div class="form-group">
    <label for="titre">Titre: </label>
    <div [ngClass]="{'has-success': form.controls.titre.valid, 'has-error': form.controls.titre.dirty && f
      <input type="text" name="titre" class="form-control" autocomplete="off" formControlName="titre" ([r
      <ul class="help-block">
        <li *ngIf="form.controls.titre.dirty && form.controls.titre.errors?.required">Ce Champ obligato
        <li *ngIf="(form.controls.titre.dirty && form.controls.titre.errors?.minlength) || (form.contr
          Max: 50 et min: 5</li>
        <li *ngIf="form.controls.titre.dirty && form.controls.titre.errors?.alphanumeric">
          Le Titre ne peut pas contenir que des textes ou nombres</li>
      </ul>
    </div>
  </div>
</div>
```

Contrôleur :

```

createblog() {
  this.form = this.fb.group({
    titre : ['', Validators.compose([
      Validators.required,
      Validators.maxLength(50),
      Validators.minLength(5),
      this.alphanum
    ])],
    contenu : ['', Validators.compose([
      Validators.required,
      Validators.maxLength(500),
      Validators.minLength(1)
    ])]
  });
}

alphanum(njato) {
  const re = /^[a-zA-Z0-9 ]+$/;
  if (re.test(njato.value)) {
    return null;
  }else {
    return{'alphanum' : true };
  }
}

newblog() {
  const titre = {
    titre : this.titre,
    contenu: this.contenu,
  }
}

```

4.3. Présentation de l'application

4.3.1. Les pages d'authentifications

Avant d'utiliser l'application, il est nécessaire de s'authentifier. La page d'authentification est présentée par la figure

Identifiez-vous

Pseudo

Mot de Passe

Se connecter

[Créer un compte](#)

Si jamais l'utilisateur n'a pas encore son compte, alors il peut la crée comme illustre la figure **XX**

CRÉEZ UN COMPTE

Nom

Pseudo

Email

Mot de passe

Inscription

[Vous avez déjà un compte ?](#)

Une fois connecté, l'utilisateur est redirigé vers la page d'accueil du site, qui est illustré par la figure

- 4.3.2. Les pages blog de l'utilisateur
- 4.3.3. Les pages d'annonces de l'utilisateur
- 4.3.4. Le page d'accueil

CONCLUSION GENERALE

Netapsys est une société de services en ingénierie informatique, spécialisée dans le domaine du développement d'applications informatiques notamment dans l'externalisation des services informatiques et de la sous-traitance offshore et constitue une véritable solution pour l'optimisation des coûts de développements informatiques. Elle possède de compétence en plusieurs technologies : PHP, Java, .Net, IHM, CMS. Ce projet s'est déroulé au sein du pôle PHP.

Le projet consiste au développement d'une plateforme de vente en ligne.

Durant la réalisation du projet, si la méthode de gestion utilisée a été basée sur le manifeste agile et ses approches, notamment SCRUM et XP, la méthode de conception qu'on a utilisée est le langage UML.

Pour la technologie utilisée, le site est développé avec le Framework Javascript Angular 4 utilisant la base de données Mongoddb et le serveur Nodejs.

Ce stage nous a permis de perfectionner nos connaissances acquises en programmation web et de nous faire intégrer dans la vie en société.

Cependant, les perspectives d'amélioration envisagées pour ce projet sont encore nombreuses. Comme l'optimisation des codes pour l'amélioration des temps de réponse, la refonte du design du site, l'intégration sur les plateformes mobiles, l'ajout de nouveau mode de paiement pour les clients.

BIBLIOGRAPHIE

WEBOGRAPHIE

GLOSSAIRE

Conception	Faculté de compréhension d'un domaine d'étude pour créer ou inventer le résultat tant attendu.
CRUD	Il désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données.
Framework	Ensemble cohérent de composants logiciels structurels, qui servent à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture).
Git	Logiciel de gestion de versions décentralisé ou distribué. Il est conçu pour être efficace tant avec les petits projets, que les plus importants. Il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé en 2016.
MVC	Architecture pour le développement d'applications logicielles qui sépare le modèle de données, l'interface utilisateur et la logique de contrôle. Ce modèle d'architecture impose la séparation entre les données, les traitements et la présentation, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.
Méthode	Manière de faire une chose en suivant certains principes et avec un certain ordre.
ORM	C'est une technique de programmation faisant le lien entre le monde de la base de données et le monde de la programmation objet. Elle permet de transformer une table en un objet facilement manipulable via ses attributs.
Système	Combinaison de parties qui se coordonnent pour concourir à un résultat ou de manière à former un ensemble.
Plateforme	C'est une base de travail à partir de laquelle on peut écrire, lire, développer et utiliser un ensemble de logiciels
SCRUM	Cadre méthodologique de gestion de projet agile.
SQL	Langage standard pour les traitements de base de données, signifiant "Structured Query Language".
UML	(U nified M odeling L anguage) Langage de modélisation graphique à base de pictogrammes, couramment utilisé en conception orientée objet et en développement informatique, conçu pour fournir une méthode normalisée pour visualiser la conception d'un système.

TABLE DES MATIERES

RESUME

Mon stage au sein de la société Netapsys MADAGASCAR est basé sur la réalisation d'un site web dynamique. Ce projet a pour but la gestion des annonces et du blog.

Cette application facilitera le traitement des besoins de clients, de fidéliser la relation avec ces derniers, de les satisfaire et d'améliorer l'image de l'entreprise.

Pour réaliser ce projet, le langage UML a été utilisé au niveau de la conception, ensuite l'application a été développée en Javascript avec le technologie MEAN en utilisant l'IDE Visual Studio Code.

Ce stage nous a permis d'acquérir une bonne expérience pratique dans le domaine de la conception et du développement logiciel que nous ne connaissions que de manière théorique à l'Ecole

Mots-clés : Clients, gestion, MEAN, Javascript, UML

ABSTRACT

My internship at Netapsys MADAGASCAR is focused of a dynamic website. Its objectives are the Ads and blog management.

This application will make the customer needs treatment easier, retain the customer relationship, satisfy them and optimize the company image.

During this project, UML language was used for the conception, then the application was developed on Javascript language using MEAN technology, on Visual Studio Code IDE.

This internship allowed us to acquire a good practical experience in the field of design and software development that we knew only theoretically at the School.

Keywords : Customers, management, MEAN, Javascript, UML

