N° d'ordre :……../….

Université Cadi Ayyad
Ecole Nationale des Sciences Appliquées de Safi
Département Informatique, Réseaux et Télécommunications (IRT)

MEMOIRE
de PROJET DE FIN D'ETUDES

pour l'obtention du diplôme :

INGENIEUR D'ETAT en Génie Informatique

Refonte d'une application et proposition d'un nouveau modèle d'architecture d'application : SPA, SOA & Mini services

Réalisé Par
BENATHMANE Nohayla

Effectué à
Accenture - Casablanca

Encadré à l'ENSAS par :
Prof. Abdelghali AMMAR

Encadré à Accenture par :
Mr ABDERRAZAK Abdellah

Soutenu le …./06/2019 devant le jury composé de :

Mr. Abdelghali AMMAR, Professeur à l'ENSA de Safi
Mr. Hakim EL BOUSTANI, Professeur à l'ENSA de Safi
Mr. Redouane CHAHINE, Professeur à l'ENSA de Safi

Année Universitaire : 2018/2019

Appréciation et signature de l'encadrant

Dédicaces

A nos chers parents, qui n'ont pas cessé de nous encourager, soutenir et prier pour nous durant toutes les années de nos études, qui ont toujours été présents à nos cotés pour nous consoler quand il fallait. En ce jour mémorable, on vous dédie ce travail en signe de notre amour et vive reconnaissance.

A mon frère et ma sœur pour leur encouragement.

A tous ceux qui ont contribué de près ou de loin pour notre réussite.

Que ce travail soit l'accomplissement de vos vœux tant allégués et le fruit de votre soutien infaillible.

Merci d'être toujours là pour nous.

Remerciement

Je tiens à remercier toutes les personnes qui m'ont consacré un peu de leur temps ainsi que celles qui ont fait en sorte que mon stage se déroule dans de bonnes conditions.

Tout d'abord, j'adresse mes remerciements à mon tuteur en entreprise, M. ABDERRAZAK Abdallah, qui m'a accueilli et m'a bien intégré dans l'entreprise, il m'a rapidement formé à l'intégration continue grâce à ses qualités techniques et m'a aidé lorsque j'ai rencontré des difficultés. La confiance qu'il m'a accordée a été précieuse : j'ai été considérée comme un collaborateur à part entière contribuant à l'activité de l'équipe Génie Logiciel, enfin, il m'a soutenu jusqu'au bout de mon stage. J'ai pris un réel plaisir à travailler avec lui et je garderai volontiers le contact.

Je tiens à remercier vivement mon encadrant à l'ENSA de Safi, Mr AMMAR Abdelghali, qui n'a pas cessé ses encouragements pendant la durée de ce stage, ainsi pour sa générosité en matière de formation et d'encadrement et aussi pour l'aide et les conseils concernant les missions évoquées dans ce rapport.

Enfin, merci aussi à toute l'équipe pédagogique de l'Ecole Nationale des Sciences Appliquées de Safi pour son investissement et pour la qualité des cours qu'elle assure.

MERCI.

Tables de figures

Liste des acronymes

L'acronyme	Désignation
AGI	*AG Insurance*
DDD	*Domain Driven Design*
ASP	*Active Server Pages*
UML	*Unified Modeling Language*

Sommaire

Introduction générale

Le pouvoir d'une entreprise sur le marché peut être évalué en fonction du nombre de ses clients et partenaires et leurs niveaux de satisfaction par les produits et les solutions offertes par l'entreprise. Une entreprise opérante dans le secteur de développement logiciel devra concentrer ses recherches sur les astuces qui rendent ses solutions plus fiables et plus robustes tout en répondant aux besoins évolutifs de ses clients. Elle devra aussi bien fonder ses choix technologiques et logiciels en étudiant les avantages et les limites de chacun en particulier le choix architectural de ses applications. En effet, l'architecture d'une application présente la manière de création de l'application et le squelette de la solution finale.

L'architecture d'une application est l'infrastructure composée de modules actifs qui sont en interaction et d'un ensemble de règles qui gèrent cette interaction. À cet effet, l'équipe intégration doit suivre les évolutions technologiques qui ont apporté des changements dans les architectures logicielles et la gestion des cycles de vie des projets informatique. Ces changements ont pour but de développer des applications performantes, maintenables et extensibles.

À cet égard, devant un développement exponentiel et continu de ses processus métier, Accenture Maroc fait face à certains problèmes qui peuvent ralentir le fonctionnement de ses applications existantes tels que : l'augmentation du temps de réponse, la redondance des modules, la difficulté de maintenance et les besoins des clients qui ne cessent pas d'augmenter.

C'est dans ce cadre, que s'inscrit notre projet de fin d'études du cycle des ingénieurs à l'École Nationale des Sciences Appliquées de Safi réalisé à Accenture Maroc, société de conseil en management, technologies de l'information et externalisation. Notre tâche consiste à proposer à l'un des clients un nouveau modèle d'architecture d'application : SPA & Mini services en passant de l'architecture monolithique vers une architecture orientée services.

Le résultat souhaité est d'avoir une structure robuste, et fiable qui allège les complexités de l'architecture existante.

Le présent rapport décrit les différentes étapes de notre travail, et il s'articule autour de plusieurs chapitres :

Le premier chapitre comporte une brève présentation de l'organisme d'accueil et du cadre général de ce projet.

Le deuxième chapitre expose l'étude de l'existant et met l'accent sur la solution proposée. Il aborde à la fin la méthodologie de gestion de projet appliquée pour Assurer le bon déroulement de notre travail.

Le troisième chapitre présente les concepts théoriques clés, liés au style architectural mini services ainsi que les caractéristiques de base de notre architecture.

Le quatrième chapitre présente en premier lieu, une analyse détaillée des besoins fonctionnels et non fonctionnels globaux de l'ancienne application. En second lieu, il décrit le cas d'utilisation général, cas d'utilisation du produit, diagramme des classes selon l'approche DDD ainsi que l'architecture globale de notre solution.

Le cinquième et sixième chapitres détaillent le cycle de vie des sprints ayant pour objectif la réalisation des mini services du projet.

Le septième chapitre illustre l'intégration des mini services d'AG Online tout en exposant les choix technologiques utilisés pour la réalisation de notre solution, ainsi que les résultats obtenus selon ces technologies.

Nous clôturons par une conclusion générale qui présente une récapitulation du travail réalisé et ouvre quelques perspectives.

CHAPITRE I : Présentation de la société

Introduction

Pour comprendre un projet il est nécessaire d'en connaitre l'environnement. Ce chapitre a pour objectif de présenter Accenture Maroc qui est l'environnement où sera basé notre projet.

Présentation d'Accenture dans le monde

Accenture est une entreprise internationale de conseil en management, technologies et externalisation siégeant à Dublin, en Irlande, avec 350 000 employés intervenant dans plus de 120 pays. Combinant son expérience et ses capacités de recherche et d'innovation développées et mises en œuvre sur l'ensemble des métiers et secteurs d'activité, Accenture aide ses clients - entreprises et administrations - à renforcer leur performance.

Accenture est une entreprise présente partout dans le monde. Une décomposition géographique a été faite pour des besoins internes. Le plus haut niveau d'organisation géographique est composé de 3 zones : APAC, EALA et l'Amérique du nord. Une deuxième répartition a été effectuée et elle comporte 14 unités.

Voici les principaux pays dans lesquels elle est implantée :

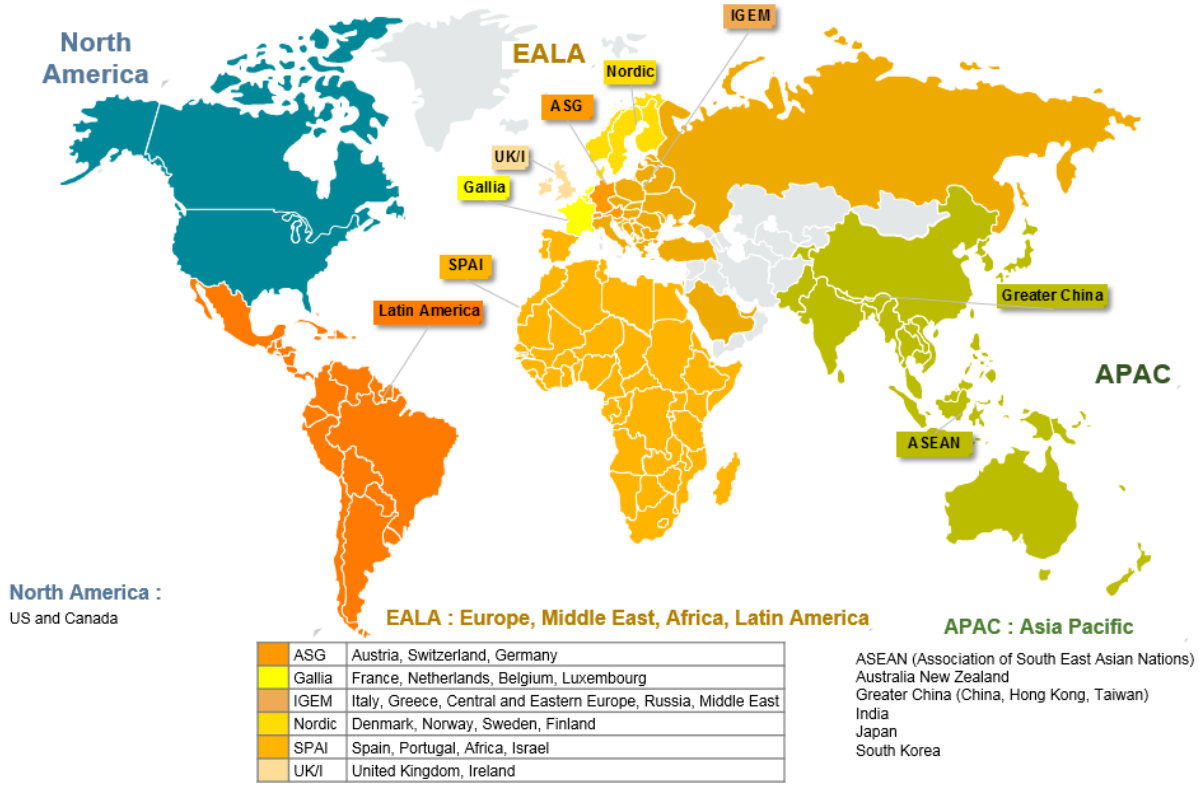

Figure 1 : Les unités géographiques d'Accenture

Sa stratégie axée sur « la haute performance réalisée » s'appuie sur sa capacité à créer de la valeur pour ses clients de manière durable. Grâce à son savoir-faire en matière d'offre de services et ses

capacités technologiques, elle identifie de nouvelles tendances économiques et technologiques, et développe des solutions pour permettre à ses clients, aux quatre coins du globe, de :

- Pénétrer de nouveaux marchés ;
- Accroître leurs chiffres d'affaires sur leurs marchés existants ;
- Améliorer leur performance opérationnelle ;
- Délivrer leurs produits et services de manière plus efficace.

Services d'Accenture Maroc

Accenture implantée au Maroc depuis 2004 par le biais « d'Accenture Services Morocco ». En 2008 le cabinet installe sa plateforme dans la zone de Casanearshore pour renforcer sa coopération avec le Maroc en matière de nouvelles technologies, surtout que le Royaume offre un climat propice à l'encouragement de tel investissement.

Le centre de Casablanca est devenu en 2008 le 50ème centre de services d'Accenture et le premier au Maghreb.

Ses équipes, présentes au Maroc depuis 2004, n'ont fait qu'augmenter : plus de 160 personnes à l'heure actuelle qui partagent les méthodologies, outils et les formations Accenture.

A son incorporation au réseau mondial de centres de services Accenture, le centre de Casablanca adopte une approche systématique de création et d'adoption de processus, méthodologies, outils et architectures éprouvés et reproductibles.

Les professionnels du centre de services travaillent en collaboration avec leurs homologues des centres Accenture dans le monde entier pour fournir à leurs clients des services d'intégration des systèmes.

Quelques chiffres d'Accenture

Voici quelques chiffres clés permettant de mieux comprendre ce qu'est devenu aujourd'hui Accenture :

- Avec environ 350 000 employés ;

- Une présence dans 120 pays et dans plus de 200 villes à travers le monde ;

- Au cours de ces cinq dernières années, ils ont travaillé sur plus de 20 000 missions auprès de 4 000 clients dont 84% des plus grandes entreprises mondiales ;

- Le chiffre d'affaire mondial Annuelle clôturée le 31 août 2015 a été de 31,8 milliards de dollars ;
- Plus de 319 000 collaborateurs dans le monde. Accenture travaille pour 98 des 100 plus grandes entreprises du monde.

Secteurs d'activités

Accenture intervient dans plusieurs secteurs d'activités :

Figure 2 : Secteurs d'activités d'Accenture

Pour une meilleure expertise au sein de chacun des secteurs, les professionnels interviennent par ligne de service.

Conseil Stratégique	• Stratégie d'entreprise, stratégie opérationnelle, fusions acquisitions et alliances.
Conduite du changement	• Assistance pour recruter, retenir les meilleurs talents pour capitaliser sur les compétences des collaborateurs, e-learning
Gestion de la relation client	• Conception et mise en place de bas de données, centre d'appels et outils marketing innovants pour trouver de nouveaux clients ou fidéliser les clients de l'entreprise, et leur proposer des offres ciblées.
Gestion de la performance financière	• Conception et mise en place d'outils de gestion financière et comptable, ainsi que de consolidation de contrôle de gestion.
Gestion de la chaine logistique	• Conception et mise en œuvre des modèles d'exploitation destinés à intégrer et optimiser les différents maillons de la chaîne, des fournisseurs aux consommateurs.
Solutions technologiques	• Définition des stratégies de technologies de l'information, développement d'applications, conception, installation et management de solution intégrées (SAP, Peoplesoft, Oracle, Siebel), architecture de réseaux et sécurité, insertion d'applications Internet dans les systèmes existants.

Tableau 1 : Lignes de services d'Accenture

Organigramme de la société

Le centre de Casablanca est piloté par Monsieur Arnaud Mas, Senior Manager, sous la supervision de Monsieur **John Malepa,** Senior Exécutive, qui a la responsabilité des centres de Maurice et du Maroc afin d'en assurer les synergies.

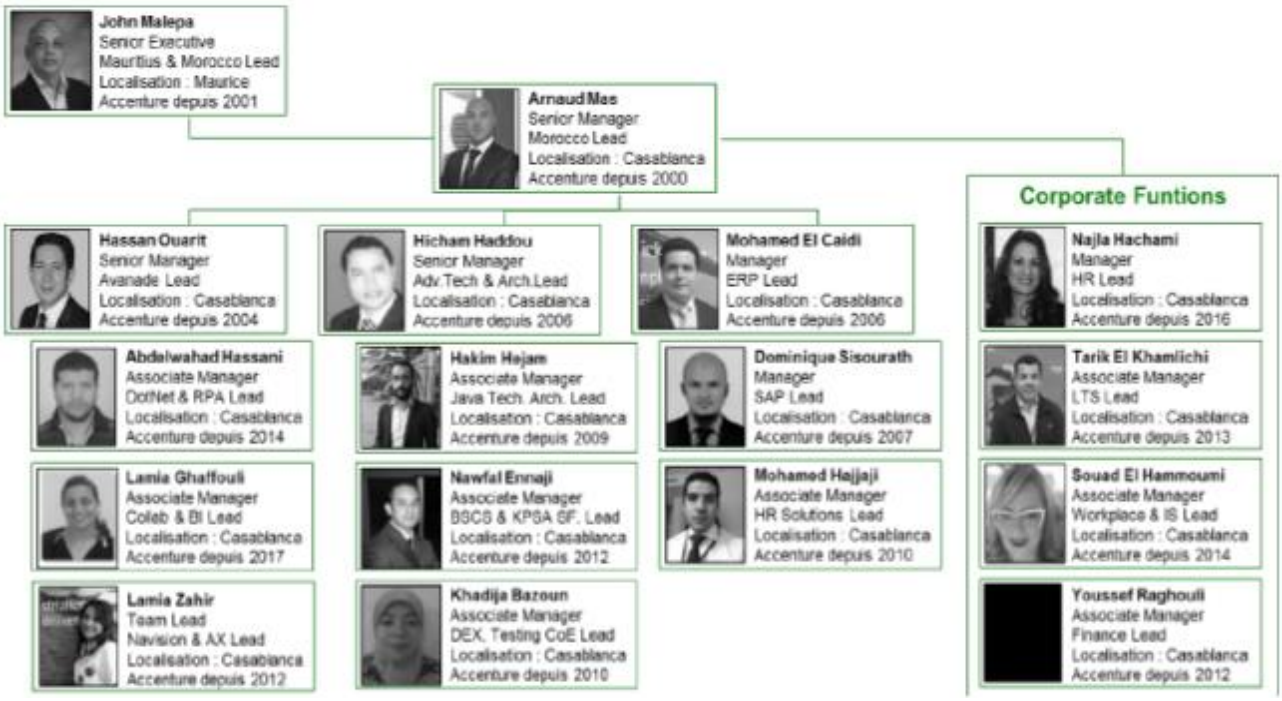

Figure 3 : Organigramme d'Accenture

Conclusion

Ce chapitre présente la structure d'Accenture qui est l'environnement où est basé notre projet durant lequel nous avons dévoilé le cadre général du travail. Le chapitre suivant sera consacré à présenter le contexte de notre projet.

CHAPITRE II : Etude générale du projet

Introduction

Dans le cadre de ce chapitre, nous allons commencer, dans un premier temps, par présenter AGI Online et ses limites. Nous allons exposer le travail demandé ainsi que la méthodologie de gestion de projet adoptée, dans un deuxième temps.

Présentation d'AGI Online

AGI Online est une application d'assurance qui gère le cycle de vie du contrat.

Il sert à obtenir des tarifs d'assurance, les offres et les contrats Primes couvrent plusieurs domaines : voiture, incendie, famille…

L'application gère maintenant plus de 10 produits et plus de 76 lois.

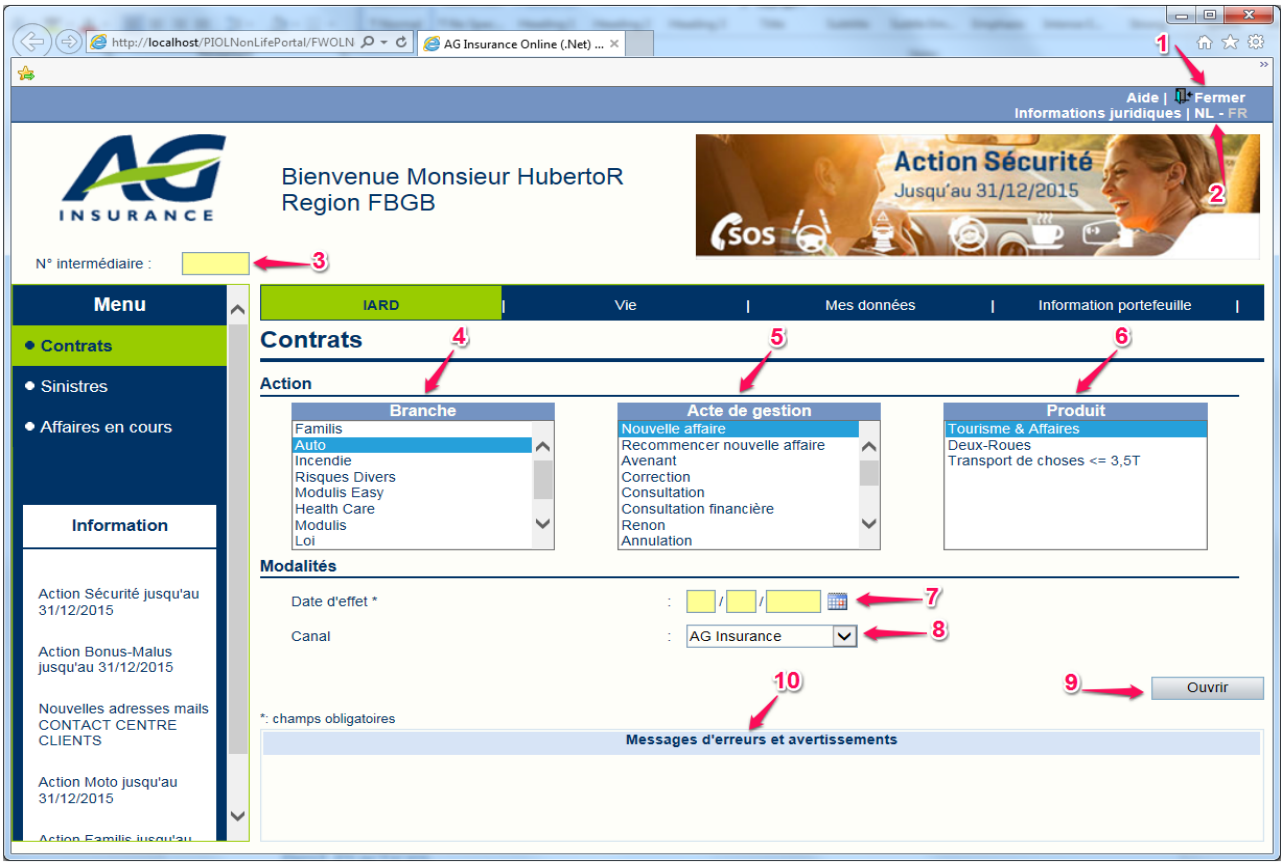

Figure 4 : AGI Online

AGI Online qui est une application monolithique à un niveau dans laquelle l'interface utilisateur et le code d'accès aux données sont combinés en un programme unique à partir d'une plate-forme unique.

WF est une technologie Microsoft pour définir, exécuter et gérer les workflows. Elle est l'un des composants de .NET permet d'invoquer des services web en utilisant des classes qui sont fortement couplées ce qui rend difficile sa maintenabilité et son évolution.

Le développement d'AGI Online est subdivisé par ordre de priorité : des fonctionnalités qui sont en production, d'autres en développement et d'autres en attente pour des itérations ultérieures. D'un autre côté, les cycles de changement sont répétitifs. Dans ces situations l'architecture monolithique a prouvé des limites en termes de flexibilité et dynamisme. Ce style de structure reste un frein face à la construction et la modification de cette application. Pour ceci Accenture Maroc prévoit de migrer cet outil afin de pallier ces problèmes.

Nous traitons dans le prochain paragraphe des inconvénients de l'architecture monolithique et ses impacts sur l'évolution d'AGI Online. Cette prochaine étape nous donnera l'élan suffisant pour définir nos axes d'amélioration et nous offrira une vue plus claire sur l'application existante.

Limites et critique de l'existant

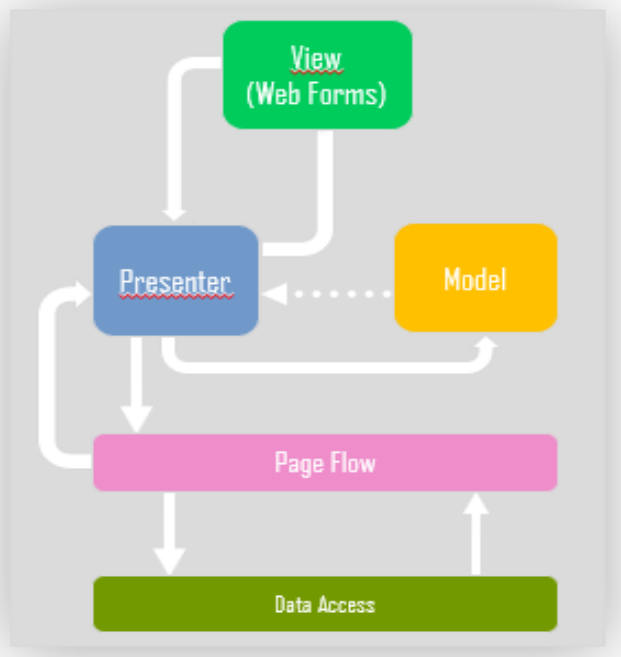

Figure 5 : L'architecture actuelle d'AGI Online

La couche modèle : contient les données "Métier" de l'application ainsi que la logique qui permet de les manipuler. Cette couche est constituée d'un ensemble d'objets basés sur des classes représentant les données à manipuler. La logique (donc les opérations) qui permet de manipuler ces données est représentée par des classes et des méthodes. Elle est développée en DTO (Data Transfer Object).

La couche Vue : Elle correspond à l'interface utilisateur. Une VUE peut être sous la forme d'une fenêtre, d'un état, d'une page Web ou d'une fenêtre mobile. Elle est développée en ASP .Net Web Forms, User Contrôle et JavaScript.

La couche Présentation : est une classe qui effectue la liaison entre la VUE et le MODELE. Elle organise et formate les données du MODELE à afficher dans la VUE. Elle regroupe les traitements concernant les actions de l'utilisateur.

Limites	Pourquoi ?	Domaine d'amélioration
L'architecture de l'application est complexe, inflexible.	-couplage inter couche élevé -Forte dépendance entre les composants	Dépendance
La connaissance des règles métier est limitée et les intégrations complexes multiples entraînent une complexité importante dans la mise à jour.	-Mauvaise documentation -Apprenabilité du code -La complexité des affaires	-La modélisation -Modulabilité par besoins fonctionnels
-Interfaces utilisateur multiples avec dépendances complexes -Le système frontal est étroitement lié au système central, ce qui nécessite des efforts considérables pour apporter des modifications	-Interface utilisateur générique pour plusieurs produits. -Le backend est responsable de toutes les règles commerciales	Dépendance
La duplication des fonctions et des processus existe, la raison principale étant que le flux ne permet pas de réutiliser certaines fonctionnalités	-Complexité technique : flux de page -Réutilisabilité	Modulabilité par besoins fonctionnels Codage standard

Les pages contenant plusieurs composants entraînent des problèmes de performances et d'expérience utilisateur	-Plusieurs composants sont rendus côté serveur	Efficacité

Après chaque itération, AGI Online, subit des améliorations. Des fonctionnalités s'ajoutent pour s'aligner plus au besoin du client. Le projet a commencé depuis des années et selon son plan d'évolution, il continuera à évoluer encore pour quelques années. Ceci a généré plusieurs défis.

En effet, la taille du projet n'a cessé d'augmenter pour devenir une application monolithique gigantesque, difficile à gérer et à comprendre. Même le respect des bonnes pratiques et les efforts fournis pour maintenir un code modulaire et évolutif n'a pas pu éliminer la complexité de ce projet. Avec une application qui comporte des milliers des lignes de code et un grand nombre de classes plusieurs problèmes se présentent.

En premier lieu, faire évoluer l'équipe est devenu de plus en plus coûteux. L'ajout d'un nouveau développeur au projet implique le sacrifice de plusieurs jours et semaines pour comprendre le code existant et afin qu'il soit capable de le modifier ou d'ajouter d'autres fonctionnalités vu l'architecture complexe et inflexible, la mauvaise documentation, dépendance complexe et le couplage excessif de l'application.

En second lieu, La connaissance des règles métier est limitée et les intégrations complexes multiples entraînent une complexité importante dans la mise à jour. La répétition manuelle de toute la chaîne de déploiement après chaque modification rend le travail de l'équipe de test plus coûteux en termes d'homme/jours.

En troisième lieu, la haute disponibilité, la rapidité de traitement et la fiabilité sont des priorités pour les fonctionnalités de base de l'application. Or, la réplication de toute l'application afin de garantir ces exigences pour quelques fonctionnalités est très coûteuse en termes de ressources matérielles.

Par ailleurs, un autre problème qui s'impose, depuis toujours, concerne la duplication des fonctions et des processus qui est dû au flux qui ne permet pas de réutiliser certaines fonctionnalités. Ainsi que les pages contenant plusieurs composants entrainent des problèmes de performances et d'expérience utilisateurs parce que plusieurs composants sont rendus coté client.

L'équipe opérationnelle exploite le code fourni par l'équipe de développement et elle est plutôt concentrée sur la stabilisation des services et de l'architecture ainsi que sur la performance des instances actuelles.

Tout cela joue un rôle important dans l'alourdissement de la mise en production. Ce qui nous a poussé à penser à un nouveau modèle d'architecture pour résoudre ces problèmes et répondre aux besoins du client.

Solution proposée et travail demandé

Accenture Maroc a proposé au client AG Insurance de faire une modernisation de l'application actuelle, et de faire une migration de l'architecture monolithique vers l'architecture Orientée services.

L'architecture orientée services ou SOA est une forme d'architecture de médiation qui est un modèle d'interaction applicative qui met en œuvre des services (composants logiciels) :

- Une forte cohérence interne (par l'utilisation d'un format d'échange pivot, le plus souvent XML ou JSON) ;
- des couplages externes (par l'utilisation d'une couche d'interface interopérable, le plus souvent un service web WS-*).

Domaine d'amélioration

Dans ce modèle d'architecture on doit améliorer plusieurs domaines et respecter les règles suivantes :

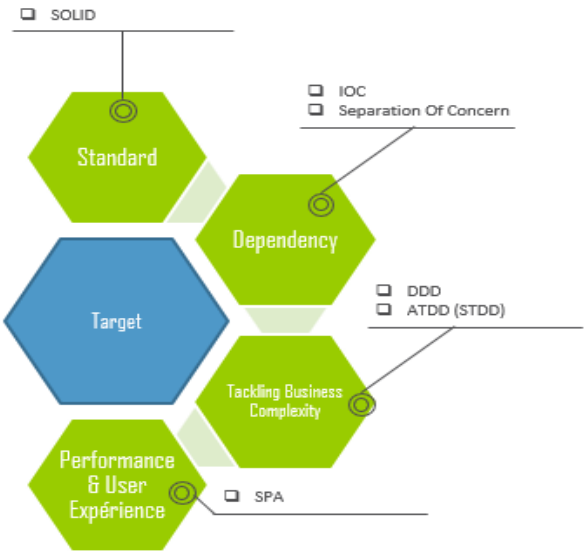

Figure 6 : domaines d'amélioration

On vise dans cette application à avoir un code SOLID qui est un acronyme représentant cinq principes de bases pour la programmation orientée objet.

SOLID signifie :

S - Single Responsibility Principle : ça veut dire qu'une classe ne doit avoir qu'une seule responsabilité.

O – Open/Close Principle : Une classe ou fonction doit être ouverte aux extensions et fermée aux modifications.

L – Liskov Substitution Principle : Une instance peut être remplacée par celle d'une sous classe sans impacter le code.

I – Interface Segregation Principle : Une classe ne doit jamais être forcée d'implémenter des méthodes qu'elle n'utilise pas.

D- Dependency Inversion Principle : Dépendre sur les abstractions pas sur les implémentations concrète

IOC : qui est un motif architectural qui permet de mieux découpler les couches. On l'utilise pour avoir une application modulaire ayant un couplage faible

Separation of concern (SoC) : est un principe de conception visant à séparer un programme informatique en parties, afin qu'il soit toujours centré sur un problème précis ça veut dire qu'on doit bien diviser le programme en petite

Pour la partie de la conception : On utilise la conception pilotée par le domaine (DDD/ Domain Driven Design) qui est une approche de la conception du logiciel basée sur deux principes :

- les conceptions complexes doivent être sur un modèle.

- l'accent doit être sur le domaine et la logique associée.

ATDD : signifie Acceptance Test Driven Développent (Développement piloté par les tests d'acceptation) : il s'agit d'une technique utilisée pour amener les clients dans le processus de conception des tests avant que le codage ne commence. C'est une pratique collaborative où les utilisateurs, les testeurs et les développeurs définissent des critères d'acceptation automatisés.

SPA : est une approche de conception de site Web dans laquelle le contenu de chaque nouvelle page est fourni non pas par le chargement de nouvelles pages HTML, mais généré dynamiquement par le biais de la capacité de JavaScript à manipuler les éléments DOM de la page existante elle-même. Elle est utilisée pour éviter les interruptions de l'expérience utilisateur entre les pages successives, pour que l'application se comporte comme une application bureau.

Méthodologie et gestion de projet

Le choix entre un modèle de processus de développement et un autre, dépend de la nature du projet et de sa taille. Lorsqu'il s'agit d'un projet où les données ne sont pas réunies dès le départ, où les besoins sont incomplets voire floues, il recommander de s'orienter vers une méthode itérative ou orientée prototypes.
Parmi les méthodes itératives, nous pouvons citer les méthodes AGILE largement utilisées de nos jours à travers le monde. Une méthode AGILE est menée dans un esprit collaboratif et s'adapte aux approches incrémentales. Elle engendre des produits de haute qualité tout en tenant compte de l'évolution des besoins du client. Elle permet aussi de gérer la qualité en continu et de détecter des problèmes le plus tôt au fur et à mesure, permettant ainsi d'entreprendre des actions correctrices sans trop de pénalités dans les coûts et les délais. Il y a plusieurs méthodes AGILE et il ne s'agit pas de choisir la meilleure méthode parmi celles existantes. Pour notre projet, nous nous sommes orientés vers une méthode de type AGILE et plus particulièrement SCRUM.

La méthode SCRUM

Le principe de la méthodologie SCRUM est de développer un logiciel de manière incrémentale en maintenant une liste totalement transparente des demandes d'évolutions ou de corrections à implémenter. Avec des livraisons très fréquentes, le client reçoit un logiciel fonctionnel à chaque itération. Plus le projet avance, plus le logiciel est complet et possède toujours de plus en plus de fonctionnalités.

Les rôles dans SCRUM

La méthodologie SCRUM fait intervenir trois rôles principaux qui sont :
• Product owner : dans la majorité des projets, le responsable produit (Product owner) est le responsable de l'équipe projet client. C'est lui qui va définir et prioriser la liste des fonctionnalités du produit et choisir la date et le contenu de chaque sprint sur la base des valeurs (charges) qui lui sont communiquées par l'équipe,
• Scrum Master : véritable facilitateur sur le projet, il veille à ce que chacun puisse travailler au maximum de ses capacités en éliminant les obstacles et en protégeant l'équipe des perturbations extérieures,
• Equipe : l'équipe s'organise elle-même et elle reste inchangée pendant toute la durée d'un sprint. Elle doit tout faire pour délivrer le produit.

Le sprint agile

Le sprint agile représente le cœur de la méthode scrum. Cette qualification lui correspond plutôt bien, puisque tous les développements incrémentaux menant petit à petit au produit final du projet sont réalisés au sein des sprints.
Un périmètre de développement est défini au début d'un sprint et doit être entièrement réalisé lorsqu'il se termine. Chaque sprint doit apporter des fonctionnalités supplémentaires à l'application en cours de développement qui doivent être livrées lorsqu'il se termine. Le Product owner est le responsable de définir les sprints et d'organiser le « backlog Product » afin de faciliter la construction du produit.

Conclusion

Ce premier chapitre constitue une étape primordiale pour fixer les repères de notre projet. Après avoir présenté l'organisme d'accueil et avoir reconnu ses attentes du projet, nous avons déterminé le cadre du travail ainsi que la méthodologie à emprunter lors de ce stage. Dans le prochain chapitre nous décrirons les caractéristiques de l'architecture mini services ainsi les concepts liés.

CHAPITRE III:
Concepts théoriques

Introduction

Afin d'atteindre les objectifs de notre projet, l'étude des concepts théoriques qui lui sont relatifs et des différents moyens mis à notre disposition est une étape primordiale.
Dans ce chapitre, nous nous intéresserons aux concepts de base liés à notre travail.
Nous décrirons, tout d'abord, l'architecture micro services. Puis, nous projetterons, les concepts liés à ce style architectural. A savoir, l'approche de la conception pilotée par le domaine, ou encore en anglais Domain Driven Design (DDD), le développement à base de composants et la polyglotte de persistances.

La nouvelle architecture proposée

Le problème qu'a causé l'ancienne architecture de l'application nous a poussé à penser à suivre un nouveau modèle d'architecture qui est l'architecture orientée services.

L'architecture orientée services (SOA):

C'est une architecture logicielle s'appuyant sur un ensemble de services simples. C'est un modèle de développement logiciel à base de composants applicatifs distribués et doté de fonctions de découvertes, de contrôle d'accès, de mappage de données et de sécurité. Il a deux grandes fonctions :

-Créer un ample modèle d'architecture qui définit les objectifs des applications et les approches pour les atteindre.

-Définir les caractéristiques de mise en œuvre précises, souvent liées à celles du langage de description de services WSDL et du protocole SOAP.

L'objectif de l'architecture orientée services :

-Structurer sous forme de services les procédures aux composants logiciels. Ces services sont conçus pour être faiblement couplés aux applications.

- Fournir un mécanisme de publication des services disponibles qui comprennent la fonctionnalité et les besoins d'entrée/sortie. La publication des services est de manière à faciliter leur intégration aux applications.

-Contrôler l'utilisation de ces services pour éviter tout problème de sécurité et de gouvernance.

L'architecture en Micro services :

L'architecture en micro services est utilisée par les entreprises afin de résoudre le problème de l'architecture orientée services monolithique qui est très lourde. Ce qui imposait pour chaque petite modification la reconstitution de tout l'édifice.

Les micro services ont été créés pour tenter de faciliter la tâche aux développeurs confrontés à de grandes applications exigeant l'assemblage de plusieurs cycles de modifications. Ils sont le fait de

diviser chaque service en plusieurs services plus petits, c'est le fait de morceler les services d'une SOA en autant micro services, donc c'est le développement permettant de concevoir une application de grande envergure sous la forme d'une série de services modulaires et chaque module prend en charge un objectif métier spécifique et utilise une interface simple et bien définie pour communiquer avec d'autres modules.

Dans l'architecture par micro services, chaque micro service s'exécute dans un processus unique et gère généralement sa propre base de données ainsi qu'il peut parallèlement être déployé, reconstitué et géré indépendamment.

Pourquoi on n'a pas opté pour l'architecture en micro services même si c'est un sujet de tendance :

Inconvénients de l'architecture en micro services :

Le terme « micro » est relatif : L'objectif de l'architecture micro services est de décomposer les applications en blocs gérables, mais les blocs doivent également être de taille raisonnable. Si les services sont trop grands ou trop petits, les applications créées à l'aide de ces derniers auront des problèmes. Un équilibre parfait peut être difficile à trouver, surtout pour des développeurs novices dans ce type l'architecture.

Une architecture fondée sur des bases de données cloisonnées pose des problèmes de mise à jour : Les transactions commerciales mettant à jour les données de plusieurs entreprises à la fois sont assez courantes. Ce type de transaction est simple à mettre en œuvre dans une application monolithique car il n'y a qu'une seule base de données. Toutefois, dans une application basée sur les micro services, différentes bases de données appartenant à différents services devront être mises à jour.

Une architecture distribuée nécessite un déploiement, un dimensionnement et des tests approfondis et ciblés : L'un des principaux avantages de l'architecture micro services, son caractère distribué, constitue également un défi considérable en matière de test, déploiement et dimensionnement des applications.

Les changements peuvent être compliqués : Les dépendances entre les services peuvent être source de problèmes lors de la mise en place de changements. Les développeurs doivent soigneusement planifier et coordonner le déploiement des changements pour chacun des services.

L'architecture en mini services :

Les Inconvénients de l'architecture en micro services nous ont poussé à repenser si l'architecture en micro services est le bon choix pour notre application ou pas. Donc on a opté pour choisir l'architecture en mini services puisqu'un mini service adopte une approche pragmatique du contenu de l'architecture de micro services. Les micro services représentent un modèle architectural très spécifique, comparé aux mini services.

Les mini-services ressemblent aux micro services en ce qui concerne l'agilité, l'évolutivité et les fonctionnalités de liaison entre services, sans que les conditions préalables relatives à la conception pilotée par les événements ne soient gênées.

Chaque micro service doit gérer ses propres données alors que les mini services partagent les données. Ce qui nous a poussé à opter pour cette architecture dans cette application.

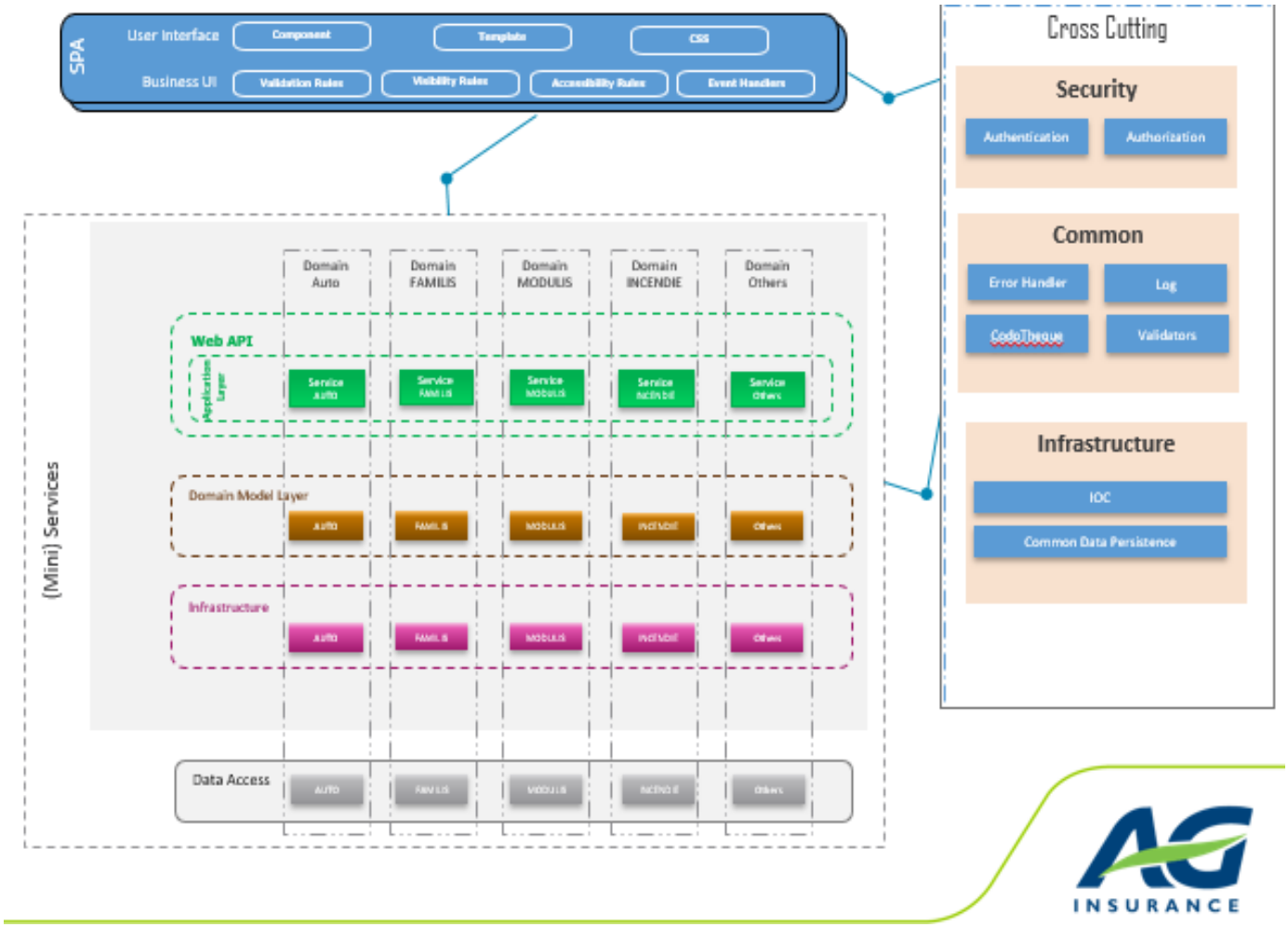

Figure 7 : L'architecture proposée AGI Online

Dans cette architecture on divise l'application en :

Couche présentation en SPA : Cette couche a pour responsabilité de présenter les informations observables du système à l'utilisateur et interpréter les actions de l'utilisateur pour interagir avec le système.

Elle est la responsable de la communication avec les autres micro services du système et elle délègue le traitement à la couche logique métier. Elle est développée en Angular **&** Type Script vu qu'il est :

- Rapide & réactif
- Une meilleure experience utilisateur
- Testabilité: E2e tests & Unit test: Jasmine, QUnit…
- Offre un contrôle pratique sur le modèle HTML
- Cadre multiplateforme : Web, mobile natif, bureau natif

Couche Application : Définit les taches que le logiciel est censé effectuer. Cette couche est mince. Elle ne contient pas de règles commerciales :

- Asp.net Web API
- Commandes et gestionnaire de commandes

Couche de modèle de domaine : Responsable de la présentation des concepts de l'entreprise, des informations sur la situation de l'entreprise et des règles commerciales :

- Entité de domaine avec données et comportement
- DD patterns:
    - Entité de domaine
    - Repository Interfaces

Couche Infrastructure : Représente la façon dont les données initialement conservées dans des entités de domaine (en mémoire) sont conservées dans des bases de données ou dans un autre magasin persistant :

- Persistence des données
- Mise en œuvre du repository (Data Access)

Cross cutting : Permet l'intégration de contrôle d'accès, d'audit et d'intégration d'identité (stratégies de sécurité) et de qualité de service. Inclut tous les services partagés.

Les concepts liés à cette architecture :

Dans cette section, nous exposerons les concepts liés à l'architecture mini services à savoir la conception pilotée par le domaine, le développement à base de composants ainsi que le polyglotte de persistance.

1- La conception pilotée par le domaine (DDD Domain Driven Design)

La conception pilotée par le domaine, dite en anglais Domain Driven Design (DDD), est une approche de développement qui favorise la création des systèmes informatiques autour des compétences métiers pour combler l'écart entre la réalité de l'entreprise et le code. En pratique, DDD encapsule la logique métier complexe en des modèles métiers.
C'est un ensemble de principes et modèles qui aident les développeurs à concevoir des systèmes d'objets élégants. Ayant pour objectif la définition d'une vision et d'un langage partagés par toutes les personnes impliquées dans la construction d'une application afin de mieux appréhender la complexité.

DDD a pour intérêt :

-L'amélioration de la stabilité et la généricité des objets du domaine métier.

-La facilité et la modularisation du modèle.

-La création d'un modèle et de le communiquer aux experts métier et aux acteurs de l'entreprise avec des spécifications fonctionnelles.

Etapes de la DDD :

Comprendre le domaine et le communiquer

La compréhension du domaine est la phase la plus importante dans une conception pilotée par le domaine, parce que ça nous permet de modéliser le domaine c'est à dire extraire le modèle. Il n'y a pas de méthode pour modéliser le domaine mais on peut utiliser des dessins, des diagrammes ou un texte écrit.
La communication du modèle se fait entre les experts du domaine, concepteurs logiciels et développeurs.
Dans cette étape l'équipe doit produire les User Stories qui représentent les besoins utilisateur à implémenter et donc définir le langage partagé.

Définition du langage partagé

Pour communiquer un domaine on doit utiliser un langage spécialisé ou une technique de la part des experts ou développeurs.
Le vocabulaire doit être commun et compréhensible pour le but d'avoir un domaine compréhensible et exploitable. Ce langage est le langage partagé qui permet de :

-Trouver les concepts clés qui définissent le domaine et ensuite la conception.
-Révéler les expressions utilisées dans le domaine.
-Chercher à résoudre les ambiguïtés et les inconnues.

Ce langage est le fruit de plusieurs discussions avec les experts du domaine.

Extraire les modèles du domaine à partir du domaine

Le modèle du domaine est une version simplifiée et implémentable du domaine qui ne contient que la partie du domaine pour laquelle le logiciel doit apporter une solution.

Qui modélise ? Quand ?

En début d'itération, l'équipe définit et échange sur les User Stories à traiter et s'accorde un temps de réflexion pour partager sa vision du modèle du domaine. Utilisateurs et experts métiers apportent leur connaissance du problème à résoudre ; développeurs et architectes fournissent leur point de vue d'experts techniques. L'équipe doit garder en tête que le modèle retenu a une limite de validité (modèle valide sous contraintes), puisqu'il est conçu pour :

- Modéliser une problématique bien précise et uniquement celle-ci : celle du *Bounded Context* en question
- Répondre au problème qui se pose aujourd'hui : demain, nous pourrons *raffiner et modifier* le modèle
- Être implémentable techniquement : si les composants et mécanismes purement techniques (relatifs à l'implémentation logicielle) ne font pas partie du domaine et ne doivent pas le *polluer*, la modélisation ne doit pas non plus ignorer ce qui effectivement réalisable et ce qui ne l'est pas !

Avec quels outils ?

On peut utiliser un tableau blanc et dessiner une représentation du modèle. Pour faire cela, UML peut aider, mais un AGL n'est pas nécessaire. Il n'est pas non plus nécessaire d'avoir une représentation exhaustive du modèle. La représentation que l'on donne n'est qu'un moyen d'échange, de partage de la compréhension par l'ensemble des acteurs.

Que mettre dans un modèle ?

La littérature abonde sur les éléments qui peuvent constituer un modèle du domaine à la sauce DDD. On peut notamment citer :

- Les Agrégats Roots : des objets ayant une *identité* et représentant chacun une *unité de cohérence* (ex : un agriculteur, une commande)
- Les *services du domaine*, représentant des opérations agissant sur plusieurs objets (ex : une opération bancaire)
- Les *repositories*, abstraction du moyen de stockage des *Agrégats Roots*
- Les spécifications, qui représentent des ensembles de critères et permettent d'exprimer des règles métier
- Les *évènements du domaine*, qui matérialisent des évènements importants survenus dans le domaine

Ces patterns peuvent aider à mieux structurer le code, à séparer les responsabilités et expliciter les concepts. Idéalement, ils sont connus par l'ensemble de l'équipe, y compris les profils non techniques : les échanges seront d'autant plus facilités que les concepts seront partagés.

Architecture et implémentation

L'architecture logicielle qui portera le domaine est celle proposée par Éric Evans (le créateur de l'ouvrage Domain Driven Design qui décrit cette approche).

L'interface Utilisateur :

Cette couche a pour responsabilité de :

- Présenter les informations observables du système à l'utilisateur
- Interpréter les actions de l'utilisateur pour interagir avec le système

L'interface Application :

Cette couche a pour responsabilité de coordonner les activités de l'application. Elle ne contient pas de logique métier et ne maintient aucun état des objets du domaine. Elle peut cependant maintenir un état de la session applicative. Elle porte la gestion des transactions et la sécurité. Sa responsabilité est également de récupérer les objets du domaine via le *repository* pour "injecter" la dynamique dans le domaine. Enfin, elle est également en charge de l'ajout ou la suppression d'objets dans le *repository*.

La couche Domaine :

Cette couche comporte toutes les classes correspondant aux éléments du modèle du domaine. Chaque agrégat correspond à une classe.

Couche Infrastructure :

Cette couche sert les autres couches, notamment pour :

- L'anti-corruption
- L'isolation de la complexité technique de l'application
- La persistance des données du modèle
- La communication entre les différentes couches

Finalement, cette architecture en couches n'est pas si éloignée de ce que l'on rencontre dans bon nombre d'applications. La différence fondamentale est que le domaine est un ensemble d'objets "intelligents" et que le code du domaine est suffisamment expressif pour que l'on comprenne ce que fait l'application.

CHAPITRE IV:
Analyse et spécification des besoins