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

Réalisation d'une application de gestion des imputations des collaborateurs au sein d'SQLI

Réalisé Par

Ghassan Guessous

Effectué à
SQLI - Rabat

Encadré à l'ENSAS par :
Prof. Hakim El Boustani

Encadré à SQLI par :
Mr Karim Raouine

Année Universitaire : 2018/2019

Table des figures

Table des matières

1 Contexte général

1.1 Présentation de l'organisme d'accueille

L'objectif ici est de présenter le groupe SQLI, et de décrire ses secteurs d'activités, ses chiffres clés, ainsi que quelques exemples de systèmes déployés sur son Intranet.

Figure 1: SQLI LOGO

1.1.1 SQLI Group

SQLI est une société française créée en 1990, pour accompagner les entreprises dans l'utilisation des nouvelles technologies.

Elle s'est spécialisée dans la réalisation des systèmes d'informations de la nouvelle génération. Elle est organisée en agences de proximité, afin de conserver le maximum de réactivité face aux besoins de ses clients. En prenant en compte, au sein même de son organisation, les préoccupations du tissu économique régional, SQLI offre une approche sur mesure aux enjeux spécifiques des entreprises.

SQLI comporte plus de 2000 collaborateurs répartis dans les 17 agences dont 10 en France (Paris, Lyon, Toulouse, Bordeaux, Rouen, Nantes et Lille), en Suisse (Lausanne et Genève), au Luxembourg, en Belgique (Bruxelles), aux Pays-Bas et au Maroc (Rabat et Oujda).

SQLI compte plus de 1200 clients, grands comptes et PME, issus de tous les secteurs d'activité (voir Figure 1).

- L'industrie
- Les services
- Les banques et assurances
- Les administrations et services publics

- La distribution
- L'immobilier
- Les transports
- Les télécoms

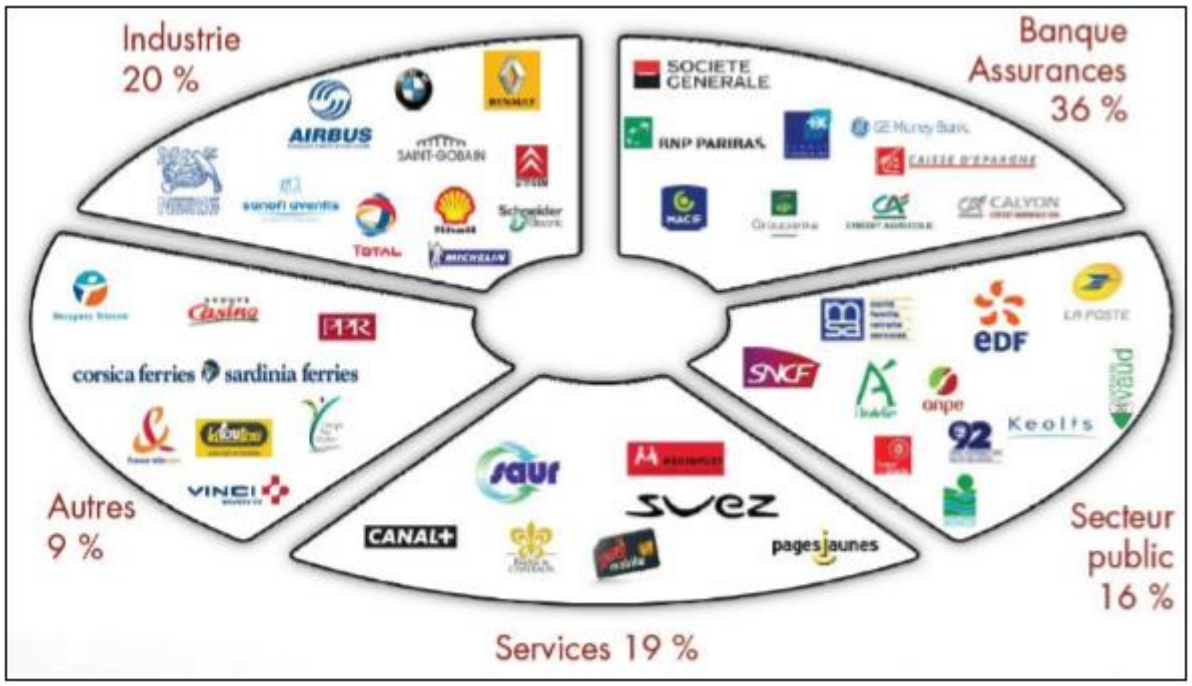

Figure 2 : Différents secteurs d'activité

1.1.2 Organigramme

Les différents centres de Rabat et Oujda étaient indépendants mais aujourd'hui avec la nouvelle organisation les deux sites ont mergé en donnant naissance à une seule entité « ISC Maroc ».

L'objectif de cette transformation est de gagner en synergie et donner une lecture plus simple de l'organisation.

La figure suivante montre l'organigramme de l'entreprise SQLI.

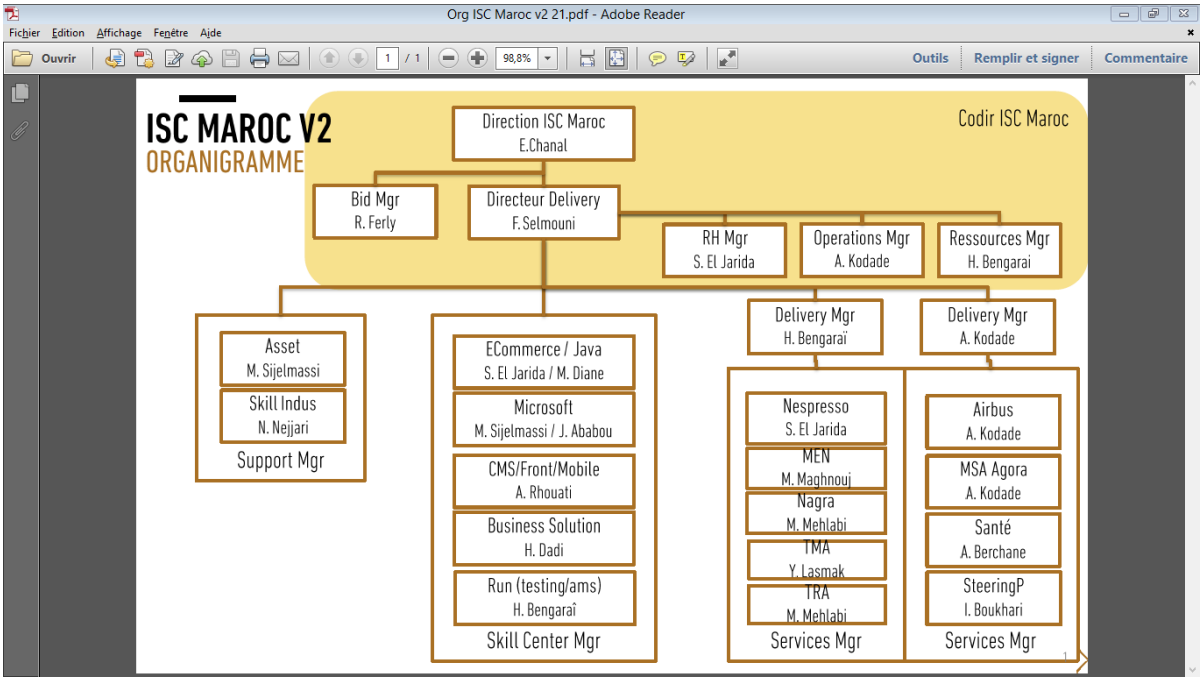

Figure 3 : Organigramme d'SQLI

Eric Chanal assure la Direction globale de l'ISC Maroc en s'appuyant sur Fouad pour les livraisons et Renaud Ferly pour la détection des futurs projets et la bonne coordination des avant-ventes.

Les deux sites passent sous la responsabilité de Fouad Selmouni. Fouad aura un focus Delivery global afin de s'assurer que les produits sont de haute qualité pour les clients avec des moyens de production adéquats.

Pour cela, il aura sous sa responsabilité :

- Saïd el Jarida, Responsable RH sur les deux sites. Il veillera au bon déroulement des processus RH et proposera des actions pour améliorer les conditions de travail et la montée en expertise des collaborateurs.
- Abderahmane Kodade, Operations Manager sur les deux sites. Son rôle est d'anticiper le résultat des activités pour les mois à venir afin de prendre au plus tôt les meilleures décisions et d'être prédictible dans les prévisions pour le groupe.
- Hicham Bengaraï devient Ressource Manager sur les deux sites. Il sera informé de tous besoins de recrutement et il optimisera les affectations sur les projets pour utiliser au mieux les ressources sur les 2 sites.

1.1.3 Métiers SQLI

SQLI s'est spécialisée dans les projets e-commerce, liés aux systèmes d'informations intégrant l'utilisation des technologies internet. Pour aider les entreprises à tirer parti des technologies internet, SQLI propose un accompagnement global sur tout le cycle du projet :

- Des prestations de conseil pour aider les clients à faire les bons choix.
- La mise en œuvre concrète de ces choix par la réalisation et l'intégration.

- Un accompagnement dans le déploiement des projets et le transfert de compétences.

Le groupe SQLI fédère toutes les compétences indispensables au bon déroulement des projets de ses clients, du conseil à la réalisation en passant par l'ergonomie, le design, l'interface utilisateur et la formation (voir Figure 4).

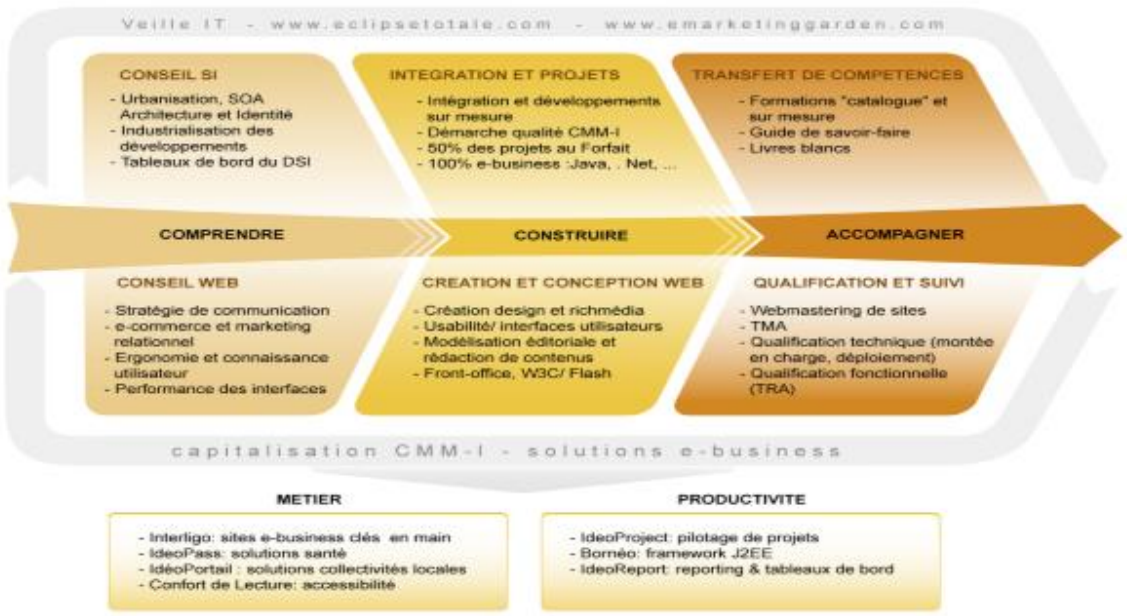

Figure 4 : Métiers SQLI

A l'instar de Nespresso ou d'Apple, les marques leaders conçoivent des expériences clients attractives, des parcours clients résolument cross-canal, et sont en mesure de générer de nouvelles sources de revenus.

Elles ont su adapter leur organisation et leur modèle opératoire : des processus de production transformés, la collaboration et l'innovation comme culture, des compétences digitales industrialisées, c'est la transformation digitale (Figure 5).

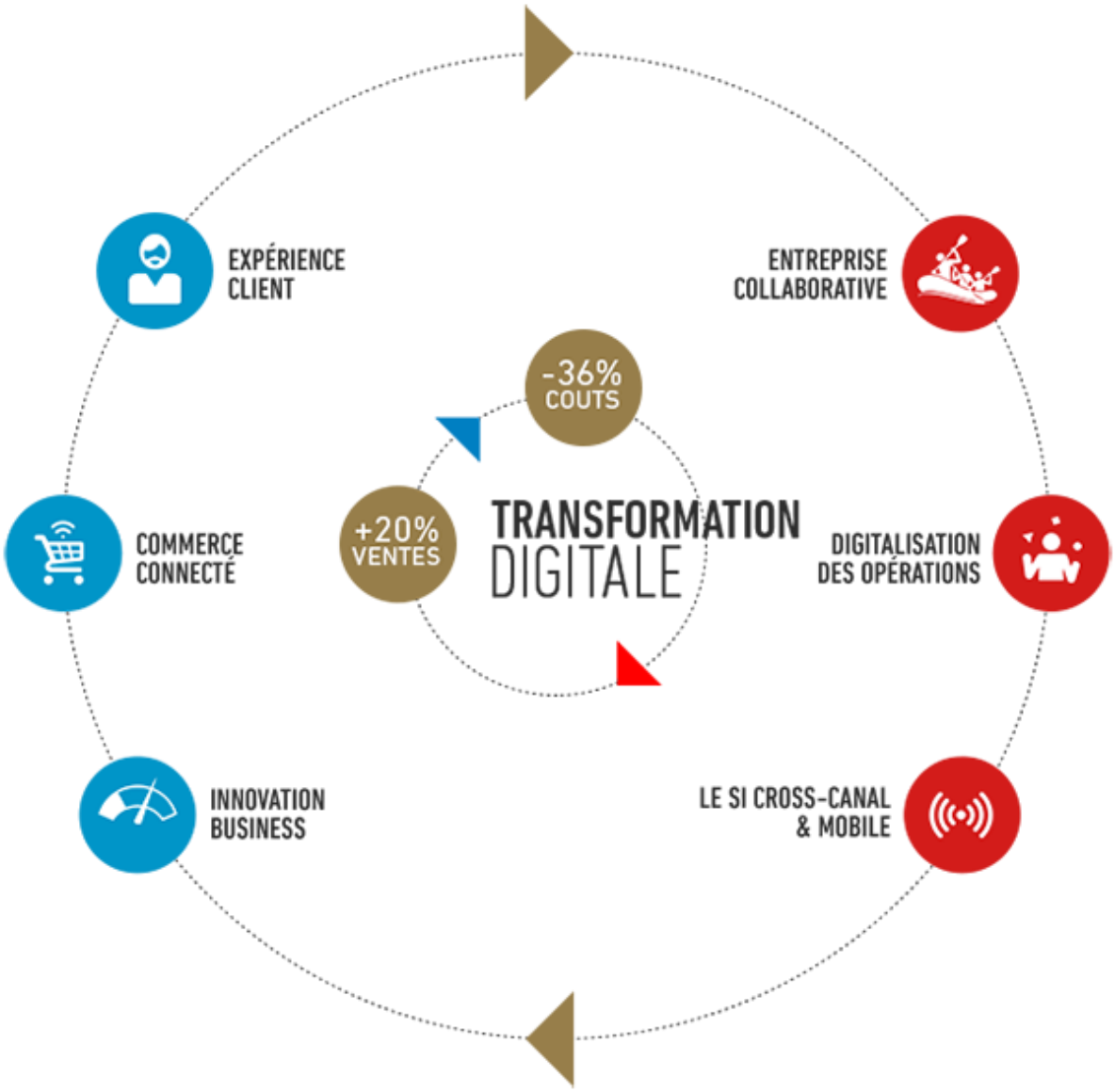

Figure 5 : Transformation digitale

1.1.4 Chiffres clés

Les chiffres clés suivants présentent la situation actuelle de SQLI :

- 29 ans d'expérience et d'innovation : SQLI place son développement sur une expertise technologique de pointe et sur sa politique intense de veille. En 2005, SQLI devient la première SSII française à obtenir la certification CMMI niveau 3.
- Plus de 2000 collaborateurs au groupe SQLI : SQLI est déjà présente au Benelux (Belgique Nederland Luxembourg), en Espagne, au Maroc et à Singapour en plus de sa présence en France.
- 232 M€ de chiffre d'affaires en 2018.

1.1.5 Structure du Groupe

Le groupe fédère toutes les compétences indispensables au bon déroulement des projets de ses clients, du conseil à la réalisation en passant par l'ergonomie, le design, l'interface utilisateur

et la formation.

Ainsi le métier du groupe SQLI est organisé en quatre pôles :

- Le pôle « Stratégie en Systèmes d'Informations »
- Le pôle « Ingénierie et intégration »
- Le pôle « Conception web : Studio SQLI »
- Le pôle « Formation et Transfert de compétences »

L'agence SQLI Rabat, dans laquelle j'ai effectué mon stage de fin d'études, est composée de plusieurs équipes travaillants sur des technologies différentes :

L'équipe Microsoft

- ASP.NET
- Windows Forms
- Silverlight
- SharePoint

L'équipe E-Commerce (Java)

- Front office
- Back office

L'équipe Open source

L'équipe FLEX

L'équipe SAP

L'équipe Agency

- Contient les projets internes à l'entreprise.

Chaque équipe est composée de plusieurs groupes de travail chacun travaillant sur un projet.

Un groupe de travail est composé d'habitude de :

- Directeur de projets : Gère tous les aspects du projet
- Un chef de projet (Scrum Master) : Gère un des aspects du projet qui est le cycle de développement logiciel.
- Architecte
- Un expert technique
- Un Business analyste
- Des développeurs (Team)
- Testeurs

1.2 Présentation du stage

Le stage de fin d'étude à SQLI est le fruit de presque trois mois de travail, après la participation au concours e-challenge j'ai intégré l'entreprise pour un stage d'une durée de six mois.

1.2.1 Compétition e-challenge

Depuis 2006 chaque année SQLI lance une édition de son concours national e-challenge visant à :

- Dresser une passerelle entre les écoles et SQLI
- Créer un espace d'échange mutuel
- Confronter les futurs lauréats à un contexte issus de la réalité du marché
- Proposer aux meilleurs candidats des stages de fin d'études pré-embauche
- Donner une chance de gagner des prix pour les finalistes

Le concours est une compétition de programmation qui regroupe des jeunes développeurs de toutes les écoles marocaines, le plan est divisé comme suit :

Etape 1 : présélection

Sur la base des Plans projets reçus par SQLI Rabat, une présélection sera faite pour choisir les candidats qui vont passer à la deuxième étape. Les candidats retenus à cette étape seront contactés par mail pour leurs préciser les modalités d'envoi du deuxième livrable.

Etape 2 : évaluation technique

Sur la base des livraisons d'applications reçues des candidats retenus à l'étape 1, une évaluation technique sera faite pour déterminer les candidats qui vont passer à l'étape 3.

Etape 3 : Soutenance et entretien RH

Les candidats choisis à l'issu de l'étape 2, seront amenés à :

D'abord, soutenir leur projet devant un jury. La durée de soutenance ne doit pas excéder 30 minutes.

Ensuite, passer un entretien RH individuel de 30 min max avec un jury de SQLI Rabat.

A l'issu de ces 3 étapes, les finalistes bénéficieront de prix décernés par SQLI. Les meilleurs candidats bénéficieront également de stage de fin d'études avec option de Pré-embauche.

1.3 Plan d'intégration

1.3.1 Contexte

L'entreprise SQLI comme la plupart des SSII est divisée en équipes projets, chaque équipe travail sur un projet de manière quasi indépendante, le chef de projet permet d'assurer les fonctions RH quotidiennes et le Scrum Master s'occupe de tout le cycle de vie logiciel.

Le plan d'intégration qui s'est étalé du Lundi 4 Février 2019 au Jeudi Février 2019, comportait principalement des formations sur différentes notions et technologies et était animé par les collaborateurs de l'entreprise.

Dans ce chapitre je présenterai le plan d'intégration auquel j'ai pris part, les objectifs et les points traités.

1.3.2 Objectifs

L'objectif principal de ce plan d'intégration est bien entendu de présenter le contexte des projets de l'entreprise et permettre aux collaborateurs d'intégrer les différentes équipes.

Les formations programmées consistaient tout d'abord en quelques rappels concernant des bases du langage JAVA et des principes de développement mais aussi des technologies utilisées dans le projet.

Le plan d'intégration devait surtout faire une montée en compétence des collaborateurs sur des points critiques, surtout en ce qui concerne la qualité de code puisque le code de base des projets est assez volumineux et nécessite un grand effort pour le rendre clair et lisible.

Et enfin, comme objectif il fallait initier les collaborateurs à une notion vitale, qui est le refactoring puisqu'une grande partie du code existant va devoir être améliorée.

1.3.3 Plan de formation

La formation s'est étalée du Lundi 4 Février 2019 au Jeudi 28 Février 2019, nous avons eu des exercices à rendre pendant toute la durée.

Thème	Responsable/Backup	Description	Durée (en j)	Commentaire	04/02/2019	05/02/2019	06/02/2019	07/02/2019	08/02/2019	09/02/2019	10/02/2019	11/02/2019	12/02/2019	13/02/2019	14/02/2019	15/02/2019	16/02/2019	17/02/2019	18/02/2019	19/02/2019	20/02/2019	21/02/2019	22/02/2019	23/02/2019	24/02/2019	25/02/2019	26/02/2019	27/02/2019	28/02/2019	01/03/2019	02/03/2019	03/03/2019
Séminaire d'intégration	Hajar Alaoui Ismaili	Séminaire d'intégration	0,25		0,25																										
Environnement technique de Dév	Younes El Haiti	Eclipse/IntelliJ (Checkstyle, Formattage, Racourcis...) Git Maven/Ant, Jenkins, Bitbucket...etc	0,75	Une application des différentes commandes de git est requise durant cette formation	0,75																										
Environnement technique de Dév (Suite)	Younes El Haiti	Eclipse/IntelliJ (Checkstyle, Formattage, Racourcis...) Git Maven/Ant, Jenkins, Bitbucket...etc	0,5	Une application des différentes commandes de git est requise durant cette formation		0,5																									
Agile	Younssef Bennouna	Jira Confluence Rituels scrum: SUM, spring planning....etc	0,5			0,5																									
Java / Clean Code /Design Pattern	Imani Abdelaziz	JAVA (principe OO, convention nommage, ...)	2	1/2 Théorie + 1/2 Pratique			1	1																							
	Rachid Amghari	Clean Code, Loi de Demeter Java efficace	2	1/2 Théorie + 1/2 Pratique					1			1																			
Spring	Bounaga Houcine	IOC MVC	1	1/2 Théorie + 1/2 Pratique									1																		
Hybris Core	El Mahdi Benzekri (3j) Hicham Jeffar (2j)	Hybris core + Hybris 123	5	1/2 Théorie + 1/2 Pratique										1	1	1			1	1											
Hybris commerce	Hicham Jeffar	Pres + commerce trails	5	1/2 Théorie + 1/2 Pratique																	1	1	1			1	1				
Skills	Abderrahmane Mrani	Formation officielle Skills	1																								1				

Figure 6 : Plan de la formation

1.4 Contenu de la formation

Dans ce qui suit, nous allons traiter quelques points qu'on juge important, l'objectif est de présenter les compétences qui étaient demandés par le projet et en même temps ça va résumer le déroulement du mois de formation.

1.4.1 Environnement technique

Le but de ce volet n'est pas bien entendu d'énumérer les technologies à utiliser, mais il est plutôt question d'efficacité et d'optimisation de l'utilisation.

En effet la plupart des développeurs savent utiliser des éditeurs comme Eclipse ou intelliJ pour coder, mais rare sont ceux qui ont tirent vraiment profit, ces éditeurs proposent un ensemble de fonctionnalités qui peuvent doubler le temps passé à coder.

C'est dans cette vision ou s'inscrit notre formation sur les éditeurs, nous avons pu étudier les principales fonctionnalités, tel que les plugins de gestionnaire de version, de qualité de code ou de tests unitaires.

Nous avons aussi appris les principaux raccourcis clavier qui sont vraiment très efficace pour

coder, éviter l'utilisation de la souris au maximum peut nous faire gagner pas mal de temps.

De la même manière des outils comme Git ou Maven sont indispensable pour des projets de grande envergure, sans Git par exemple travailler en groupe seras très compliqué, pendant les séances de pratique nous avons utilisé Bitbucket qui est une solution de gestion de code source et de collaboration basée sur Git et Mercurial dans le cloud.

1.4.2 Principes de base

1.4.2.1 Nommage

Les noms sont partout dans le logiciel. Nous nommons les variables, fonctions, les arguments, les classes, les paquets, les fichiers sources etc. Parce que nous faisons ça tellement, nous devons le faire bien.

Les noms doivent révéler l'intention de la classe, les fonctions et variables. Si les noms sont propres, alors il n'y a pas besoin de commentaires. Le contenu des commentaires peut être le nom de la fonction elle-même.

1.4.2.2 Les fonctions

Les fonctions doivent faire une seule chose, la faire bien, et ne faire vraiment que ça c'est la règle principale et à cela s'ajoute quelques points importants :

- Les fonctions doivent être brèves, si un bloc de code nécessite un commentaire alors ce bloc doit être dans une nouvelle fonction.
- Il faut avoir la déclaration, l'initialisation et l'appel des variables dans l'ordre.
- Eviter au maximum les effets de bords, il ne faut pas que la fonction ait un comportement caché.
- Eviter les paramètres de sortie et préférer les exceptions qu'aux codes de retour.

1.4.2.3 Les classes

Les classes sont les composants essentiels d'une application, si elles sont bien représentées alors l'application est bien structurée et surement vas être facilement extensible et maintenable.

Pour quelqu'un qui vas lire notre code, on doit s'assurer que notre classe est facilement compréhensible et qu'on révèle très bien nos intentions, en fait ça doit être comme une histoire qu'on lit de haut en bas et ce sont les noms des fonctions qui expliquent les différents scénarios.

Finalement les classes doivent avoir un nombre limité de variable et utiliser au maximum ces variables, la répartition des responsabilités, le couplage faible et la haute cohésion sont les facteurs essentiels d'un code propre et seront traité dans la suite.

1.4.2.4 Qualité de code

Compte tenu des exigences du projet, il n'est plus question de développer de manière traditionnelle et s'arrêter quand le code marche, il faut aller au de la et s'assurer que le code écrit est bien structuré et facilement compréhensible par les autres membres de l'équipe.

Cela permet notamment à un autre développeur de modifier le code existant ou au même développeur de s'y retrouver lorsqu'il reprend son propre code longtemps après.

Au début des points essentiels ont été traités, tel que les règles de nommage, le style d'écriture de code qui est un point important, dans l'équipe nous avons un style d'écriture commun qu'on a défini et qu'on affecte à nos éditeurs de texte et au logiciel de contrôle de la qualité de code pour qu'il soit respecté.

A vrai dire la qualité de code est le premier souci des grands projets, tant que le code s'agrandit la lisibilité diminue et l'effort de modification devient au fur et à mesure plus difficile, a un moment on peut trouver que le cout de refaire toute la solution est moins que le cout du changement.

Le code propre ou « Clean Code » est un point traité par des experts de la programmation tel que Robert C. Martin, notamment dans son fameux livre « Clean Code : A Handbook of Agile Software Craftsmanship ».

Coder proprement ne peut se faire qu'à travers un changement de la manière avec laquelle on voit le code, dorénavant on va coder pour les autres.

Pour se faire plusieurs règles et principes doivent être respectées, même si ces règles ne peuvent pas être quantifiées et dépendent beaucoup du contexte mais il est important de les connaitre.

1.4.2.5 Principes SOLID

Le mot SOLID est un acronyme pour les cinq premiers principes de conception orientée objet (OOD) de Robert C. Martin, populairement connu comme l'oncle Bob.

- S : Single-responsiblity principle
- O : Open-closed principle
- L : Liskov substitution principle
- I : Interface segregation principle
- D : Dependency Inversion Principle

Ces principes, lorsqu'ils sont combinés ensemble, rendent facile pour un programmeur à développer des logiciels qui sont faciles à maintenir et à étendre. Ils font aussi important pour éviter les erreurs de codage, faciliter l'optimisation du code, et sont aussi une partie du développement logiciel agile ou adaptative.

Par la suite nous allons présenter rapidement chacun des principes.

a. SRP : Principe de responsabilité unique

Ce principe indique qu'une classe ne doit avoir qu'une seule raison d'être modifiée.

Si une classe comporte plusieurs responsabilités alors ces responsabilités sont couplées. Le fait de changer une responsabilité va influencer les autres responsabilités de la classe.

Les classes comportant des responsabilités couplées entraîneront des difficultés de maintenance ou des dysfonctionnements inattendus.

La modification d'une responsabilité de la classe aura un impact sur les autres responsabilités.

b. OCP : Principe d'ouvert/fermé

Le principe s'énonce de la manière suivante : Une classe doit être extensible sans modifier son

code.

Le but de ce principe est de mettre en place des classes pour lesquelles il sera possible d'ajouter de nouveaux comportements sans modifier le code de la classe elle-même.

En effet, l'impact d'un changement dans le code est souvent risqué et coûteux.

c. LSP : Principe de Substitution de Liskov

A l'origine, ce principe a été énoncé par Barbara Liskov et Jeannette Wing.

LSP est une définition particulière concernant les sous-types d'une classe. Il a été exprimé de manière plus simple par Robert C. Martin de la manière suivante : « Tout type de base doit pouvoir être remplacé par ses sous-type ».

Ce principe peut aussi être expliqué grâce à la conception par contrat : il impose que les préconditions ne peuvent pas être renforcées dans une sous-classe et que les postconditions ne peuvent pas être affaiblies dans une sous-classe.

En utilisant ce principe, le code ne devra pas dépendre de la hiérarchie des classes. Par conséquent, il ne devra pas utiliser de cast ou d'opérateur « as » ou « is ».

d. ISP : Principe de Ségrégation des Interfaces

Ce principe est très simple à comprendre et s'exprime de la manière suivante :

Un client ne doit jamais être forcé de dépendre d'une interface qu'il n'utilise pas (Robert C. Martin).

Lorsqu'on y réfléchi, le fait d'avoir une interface « trop grosse » va vite entraîner des problèmes d'efficacité : chaque classe implémentant l'interface devra implémenter toutes les méthodes de l'interface. Il est donc intéressant de réduire cette interface au minimum nécessaire.

De plus, si deux classes A et B dépendent d'une interface I, le fait de modifier I pour la classe A va également demander de modifier la classe B. Nous remarquons donc déjà un problème de couplage fort entre A, B et I.

e. DIP : Principe de l'Inversion des dépendances

Le principe d'inversion des dépendances est celui que je préfère. En effet, il permet de mettre en place des applications plus souples.

DIP s'exprime de la manière suivante :

Les modules de haut niveau ne doivent pas dépendre des modules de bas niveau. Les deux doivent dépendre d'abstractions.

Les abstractions ne doivent pas dépendre des détails. Les détails doivent dépendre des abstractions.

Les modules de haut niveau sont ceux qui contiennent votre code métier et le fonctionnel de l'application. Les modules de bas niveau sont ceux qui contiennent les implémentations dépendantes de la machine, du stockage, de la communication ou des serveurs externes (exemple : base de données, serveur de logs).

f. Principes GRASP

GRASP pour « General Responsibility Assignment Software Patterns » sont des patrons de conception de haut niveau, à l'inverse des design patterns du GOF qui eux sont plus concret et peux être directement implémentés.

Le but principal de ces principes est de guider le développeur dans l'attribution des responsabilités à des objets qui collaborent.

On distingue 9 patrons de conception GRASP :

- Expert en information : C'est celui qui as l'information qui doit l'utiliser, cela s'applique si bien aux classes qu'aux méthodes
- Créateur : Déterminer quelle classe a la responsabilité de créer des instances d'une autre classe.
- Faible couplage : Diminuer le couplage des classes afin de réduire leurs interdépendances dans le but de faciliter la maintenance du code.
- Forte cohésion : Avoir des sous-classes terminales très spécialisées.
- Contrôleur : Affecter la responsabilité de réception et traitement de messages systèmes.
- Polymorphisme : Affecter un nouveau comportement à l'endroit de la hiérarchie de classes où il change.
- Fabrication pure : En vue d'avoir un système générique et évolutif on peut rapidement tomber dans le piège de la pure fabrication c'est-à-dire d créer des objets qui n'ont aucun intérêt pour l'applicatif.
- Indirection : C'est le principe de l'inversion de contrôle, la direction des dépendances est inversée.
- Protection : éviter l'impact des variations de certains éléments sur les autres éléments, principalement en cernant la portée de communication des classes.

1.4.3 Développement dirigé par les tests

1.4.3.1 Le cout du changement

La modification du code est une bonne chose, c'est ce que nous faisons tout le temps. Mais il y'a des façons de changer le code qui rendent la vie difficile, et il existe des moyens qui font en sorte qu'elle soit beaucoup plus facile.

Une courbe qui montre l'évolution du cout de projet en fonction de l'avancement du projet, on remarque tout de suite que l'évolution est exponentielle.

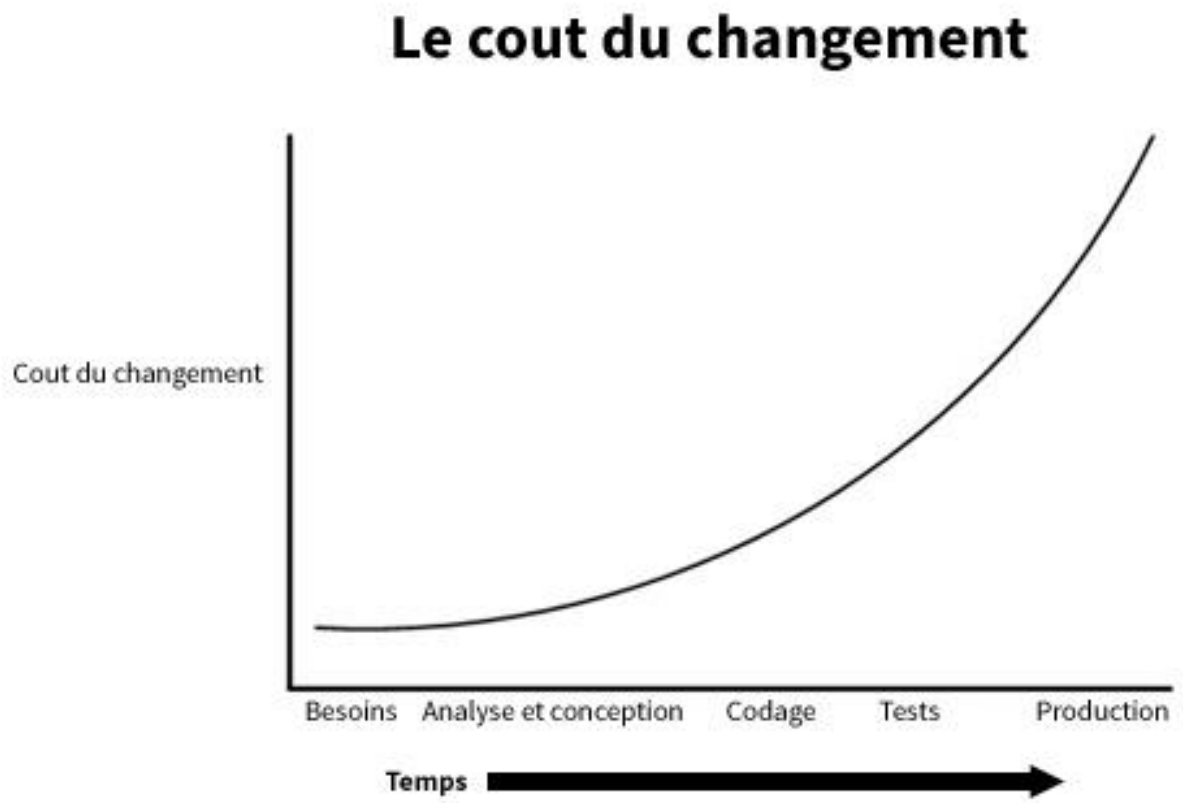

Figure 7 : Le cout du changement en fonction du temps

Dans l'industrie logicielle, la mécanique du changement a intéressé plusieurs experts, il n'est plus question de laisser le hasard jouer, pour cela une approche révolutionnaire de design et de développement a été introduite, c'est le développement conduit par les tests (Test Driven Developpement pour TDD).

1.4.3.2 Le TDD

Le « Test Driven Developpement » ou TDD encourage les programmeurs à écrire des tests avant d'écrire le code qui interagit avec ces tests, c'est un moyen de concentrer l'attention des programmeurs sur le problème qu'ils ont l'intention de résoudre plutôt que sur la solution. Cela a de nombreux avantages :

- Permet de bien exprimer le besoin avant le développement.
- On a rapidement le résultat d'un bout de code.
- Réduit le temps de révision de code.
- Alerte rapidement lorsqu'il y'a une régression.
- Permet d'avoir un design SOLID puisque le code écrit est obligatoirement structuré pour être testé unitairement.

La figure suivante schématise le développement d'une application en utilisant le TDD.

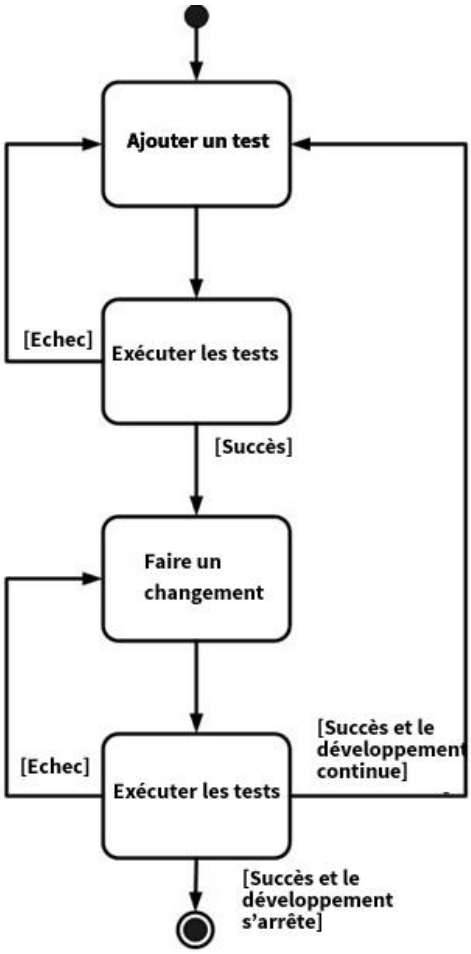

Figure 8 : Cycle de développement utilisant le TDD

- **Inconvénient du TDD**

Le développement conduit par les tests est difficile à apprendre, surtout si on essaie d'apprendre tout seul. On peut s'attendre à une baisse de productivité pendant 2-4 mois après la migration vers ce style de développement.

Un autre inconvénient est qu'il rend le développement plus lent, à chaque fois on est obligé d'écrire les tests avant de commencer à coder et à chaque modification d'un code il faut potentiellement mettre à jour les tests correspondants.

Bien que le TDD soit en lui-même suffisant pour assurer une bonne qualité de code, les développeurs vont petit à petit rencontrer certaines confusions comme que faut-il tester en premier, par lequel commencer ou encore combien faut-il de tests pour dire que c'est bon. Une réponse à ces questions nous la trouverons dans le développement dirigé par le comportement (Behavior Driven Development ou BDD).

1.4.3.3 Le BDD

Le développement dirigé par le comportement ou BDD est une évolution de la pensée derrière le TDD, c'est une méthode agile qui encourage la collaboration entre les développeurs en fournissant un langage standard.

Il rassemble des brins de TestDrivenDevelopment et DomainDrivenDesign en un ensemble

intégré, ce qui rend la relation entre ces deux approches évidentes pour le développement de logiciels plus puissants.

Au lieu de parler de "tests", une personne utilisant le BDD préfèrera le terme "spécifications". Il s'agit en fait de réunir dans un même document des exigences (User Stories) exprimés selon le canevas « given-when-then », ce qui signifie :

- given : étant donnée des conditions d'entrée
- when : quand l'action est executée
- then : on doit pouvoir constater tels résultats

1.4.4 Hybris

Figure 9 : Logo du Framework Hybris

Hybris suite est une solution e-commerce propriétaire basée sur un concept modulaire et flexible, permettant l'ajout d'un ensemble de fonctionnalités par extensions.

Après son rachat par la société SAP, hybris gagne en notoriété et avec l'ERP de SAP ils forment une combinaison puissante pour les grandes sociétés.

1.4.4.1 Les domaines d'Hybris

La plateforme hybris est constituée d'un ensemble d'extensions standard regroupés en modules, constituant la fonctionnalité principale d'hybris, un module est alors une suite d'extensions.

Les composants du produit de hybris, tels que des modules et extensions sont classés selon les domaines suivants :

- **Platform** : se compose d'un ensemble standard d'extensions qui fournissent la fonctionnalité principale d'une installation d'hybris.
- **Content area** : est un référentiel centralisé d'informations structurées, texte, contenu web, maquettes d'impression et plus encore.
- **Commerce** : Zone de commerce permet à la direction de la voie logique et les processus neutre et entreprise canal spécifique.
- **Channel** : Zone de canal assure la gestion de la visualisation de canal spécifique et l'interaction.

Hybris Multichannel Suite est un ensemble complet de fonctionnalités de tous les domaines.

Parmi les extensions une extension particulière permet de générer une structure d'une nouvelle extension, c'est ainsi qu'on définit nos propres extensions.

1.4.4.2 L'architecture générale d'Hybris

D'un point de vue commercial, l'hybris Multichannel Suite est divisée en paquets individuels, comme hybris PIM, hybris Commerce, hybris Print. Ces paquets sont des paquets de fonctionnalité assemblée pour une certaine gamme de fonctionnalités d'entreprise.

Tous ces paquets dépendent des fonctionnalités de base fournie par la plateforme hybris. Alors que la plateforme d'hybris peut fonctionner sans aucun paquet, aucun paquet ne peut fonctionner sans la plate-forme d'hybris.

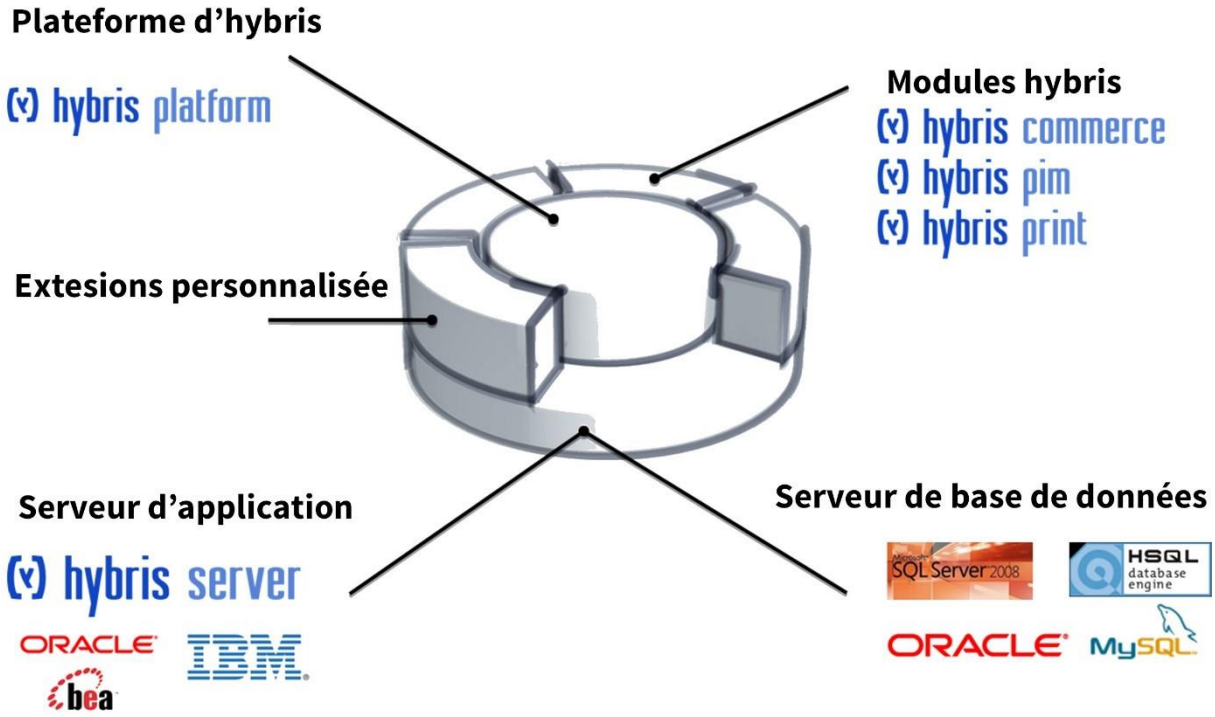

Figure 10 : Architecture d'hybris

1.5 Le projet Nespresso

Après avoir terminé la formation, j'ai été sélectionné en plus de 5 autres stagiaires, pour intégrer le projet phare de la société, le projet Nespresso.

Le projet Nespresso est l'un des projets clés de l'entreprise SQLI, il est certes son plus grand au Maroc, sa forte valeur ajoutée et sa grande rentabilité le rend un projet critique et impose que tous les membres de l'équipes soient dans un bon niveau de connaissances et d'engagement.

1.5.1 Le centre de service e-commerce

Le centre de service e-commerce est composé d'un ensemble de projets pour de grands comptes tel que Nespresso, Chanel ou Seb.

Chaque projet est organisé en équipe-projet, c'est une entité de taille limitée indépendante dans son organisation et sa gestion, on distingue entre autres les fonctions suivantes :

- Chef de projet : Manager de premier rang, personne responsable au niveau du projet
- Scrum Master : Responsable de tout le cycle de vie logiciel
- Solution Design : Responsable de la conception fonctionnelle et technique
- Expert technique : Développeur expérimenté, guide les développeurs dans leur travail
- Ingénieur concepteur développeur : spécialistes dans des domaines précis comme la création et l'intégration de l'interface utilisateur, la programmation et les bases de données.
- Testeur : exécute les différents tests de validation et de non-régression.

1.5.2 L'équipe Nespresso

L'équipe Nespresso est la plus grande équipe de SQLI Rabat, elle comporte 35 personnes réparties en cinq équipes, chaque équipe a des objectifs prés définis et travaille de manière quasi-indépendante.

Un coordinateur de chaque équipe assure le bon déroulement des procédures, un responsable de l'environnement s'occupe de la maintenance des environnements dédiés à chaque équipe et enfin les développeurs et les testeurs travaillent conjointement pour réaliser les différentes tâches.

Les équipes Nespresso situés à SQLI Rabat sont les suivantes :

- Equipe Atlas : Responsable des nouveaux développements.
- Equipe Abtal : Responsable des nouveaux développements.
- Equipe Menara : Responsable des nouveaux développements.
- Equipe Release Team : Responsable de la stabilisation des nouveaux développements avant du déploiement vers la production.
- Equipe Emergency Room : Responsable du suivie et de la correction des anomalies qui sont détectés en production.

1.5.3 Nespresso e-commerce

Le projet Nespresso e-commerce ou Nesclub2 comprend la création d'une plate-forme e-commerce internationale, basée sur la solution e-commerce Hybris, qui appuiera de nombreux pays avec des sites e-commerce spécifiques B2B et B2C.

Auparavant Nesclub1 était une solution unique et centralisée basée sur la technologie "Magic" et développée par l'équipe interne de Nespresso.

La solution actuelle comprend en plus de l'application web, un réseau d'intergiciels (NesOA) dédié à la communication entre la plate-forme e-commerce et l'ERP de Nespresso (Nessoft) qui fournit la majorité des données pour les sites Internet.

Les conceptions fonctionnelle et technique du projet ont été réalisées en collaboration entre des intervenants du Maroc et de la Suisse, tandis que le design a été conçu par le WebAgency de Paris et sa production s'est déroulée au Maroc.

Le projet est passé par plusieurs versions et plusieurs étapes et est en constante évolution.

2 Méthodologie de travail

La méthodologie de travail utilisée est une méthode agile basée sur SCRUM.

2.1 Principes d'une méthode agile

Les 12 principes d'une méthode agile :

- Satisfaire le client est la priorité
- Accueillir les demandes de changement « à bras ouverts »
- Livrer le plus souvent possible des versions opérationnelles de l'application
- Assurer une coopération permanente entre Client et Equipe projet
- Construire des projets autour d'individus motivés
- Privilégier la conversation en face à face
- Mesurer l'avancement du projet en termes de fonctionnalités de l'application
- Faire avancer le projet à un rythme soutenable et constant
- Porter une attention continue à l'excellence technique et à la conception
- Favoriser la simplicité
- Responsabiliser les équipes
- Ajuster, à intervalles réguliers, son comportement, ses processus pour être plus efficace

2.2 SCRUM

Une méthode agile est une approche itérative et incrémentale, qui est menée dans un esprit collaboratif avec juste ce qu'il faut de formalisme.

Elle génère un produit de haute qualité tout en prenant en compte l'évolution des besoins des clients.

- Valeurs caractéristiques d'un projet agile
- Les individus et les interactions plutôt que les processus et les outils
- Produit qui fonctionne plutôt qu'une documentation exhaustive
- La collaboration avec le client plutôt que la contractualisation des relations
- L'acceptation du changement plutôt que la conformité aux plans

2.2.1 Au quotidien

Chaque journée de travail commence par une réunion de 15 minutes maximum appelée mêlée quotidienne (Daily Scrum).

Seuls l'équipe, le directeur de produit et le ScrumMaster peuvent parler, tous les autres peuvent écouter mais pas intervenir (leur présence n'est pas obligatoire).

A tour de rôle, chaque membre répond à 3 questions :

- Qu'est-ce que j'ai fait hier ?
- Qu'est-ce que je compte faire aujourd'hui ?

- Quelles sont les difficultés que je rencontre ?

Le tour de parole doit être scrupuleusement respecté pour éviter que le Scrum dérive sur des discussions techniques et déborde des 15 minutes. Si le besoin s'en fait sentir, des discussions sont alors menées librement après le Scrum.

Cette réunion a un but de synchronisation pour l'équipe et ne doit pas être vécue comme un Reporting d'activité.

C'est le niveau quotidien du principe « Inspect And Adapt » de Scrum.

2.2.2 Les sprints

À la fin du sprint, tout le monde se réunit pour effectuer la revue de sprint, qui dure au maximum 4 heures.

L'objectif de la revue de sprint est de valider le logiciel qui a été produit pendant le sprint.

Une présentation puis une démonstration du produit réalisé est entamée. C'est sur la base de cette démonstration que le directeur de produit valide chaque fonctionnalité planifiée pour ce sprint.

Une fois le bilan du sprint réalisé, l'équipe et le directeur de produit proposent des aménagements sur les backlogs du produit et sur la planification provisoire de la release.

Il est probable qu'à ce moment, des items soient ajoutés, modifiés ou ré estimés, en conséquence de ce qui a été découvert.

2.2.3 Cycle de vie logiciel

Un cycle de vie d'un logiciel est un ordonnancement des différentes étapes du processus de développement. Un cycle de vie définit les étapes du processus, leur ordonnancement ainsi que les critères de passage d'une étape à une autre.

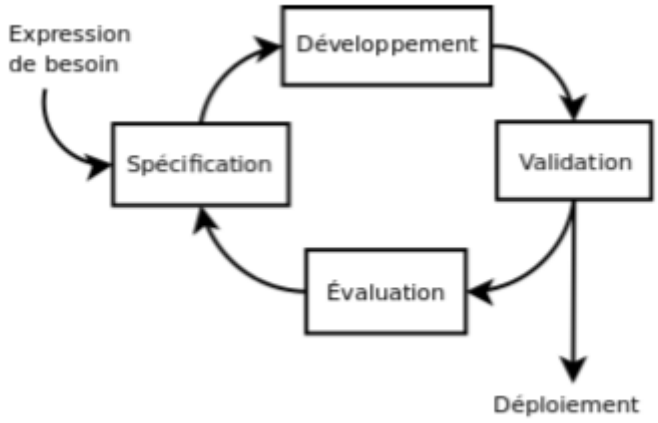

Figure 11 : Cycle de vie d'un logiciel

Puisque SCRUM exige l'organisation de la réalisation du projet en Sprints, chaque sprint sera représenté par une itération de 2 semaines concernera des charges à réaliser et aura comme produit une nouvelle version de l'application contenant de nouvelles fonctionnalités.

3 Etude de la problématique

3.1 Contexte du projet « Gestion des imputations des collaborateurs au sein du projet Nespresso »

Je me suis intégré au sein de l'équipe Menara « équipe de développement citée précédemment », et le chef de projet m'a confié d'abord un projet interne qui va être exploiter par tous les chefs de projets des équipes travaillants sur Nespresso, et après finalisation du projet je vais participer avec l'équipe au projet Nespresso.

Le projet concerne les imputations des collaborateurs (développeurs, testeurs, chefs de projets) sur lesquelles l'entreprise se base pour facturer au client (Nespresso-Nestlé) à la fin de chaque mois.

Pour expliquer le contexte du projet, ils existent trois types d'imputations concernant le projet Nespresso :

1. **Imputation APP :** les collaborateurs imputent chaque jour sur une plateforme interne qui s'appelle APP, la charge et l'effort journalier qu'ils ont travaillés.
2. **Imputation PPMC :** les collaborateurs imputent sur une plateforme hébergée chez le client pour que ce dernier suit et vérifie la disponibilité et l'engagement des collaborateurs travaillant sur son projet, en comparant avec APP.
3. **Imputation TBP :** les chefs de projet prévoient un chiffre d'affaire en le communiquant avec le service de management, en se basant sur la disponibilité de leurs équipes dans une période donnée.

Actuellement et depuis toujours les chefs de projets gèrent ce travail avec des fichiers Excel en comparant les imputations APP-PPMC et APP-TBP, et notifie tous les collaborateurs qui ont un écart, afin de justifier la non-compatibilité des charges imputés pour qu'au final régler ces écarts vis-à-vis du client et du service de management.

Le but du projet est de faciliter cette tâche. Les données de chaque type d'imputation proviennent de plusieurs services web, l'application que j'ai développé doit les consommer afin de traiter les données, les restructurés afin de produire le résultat voulu par les chefs de projets ; comparaison des différentes imputations journalières pour une période donnée pour chaque collaborateur, notifier ceux qui ont des écarts via des emails, corriger après vérification et justification, consulter l'historique et voir différents statistiques.

3.2 Situation existante