Avant-propos

Nom et prénom de l'étudiant :

TALAGHZI Hamza

Intitulé du travail :

Refonte d'un portail de gestion électronique des documents (Arkevia Refonte)

Établissement d'accueil :

- **Nom de l'entreprise** : CEGEDIM MAROC
- **Adresse** : Arribat Center Immeuble D et E 2ème étage Avenue Omar Ibn Khattab 10090 Agdal Rabat.
- **Site web** : `https://www.cegedim.fr`

Encadrant de l'établissement d'accueil :

M. GHAICH Mourad

Tuteur académique de la Faculté des Sciences de Rabat :

Mme. ZITI Soumia

Date de début et de fin du stage :

Du **15 Février 2021** au **15 Août 2021** inclus

Remerciement

Résumé

Le présent rapport synthétise le travail effectué sur une période de six mois au sein de l'entreprise Cegedim SRH, et qui s'inscrit dans le cadre de la validation du projet de fin d'études du Master Ingénierie de Données et Développement Logiciel à la Faculté des Sciences de Rabat.

La première phase du travail a consisté à auditer et analyser l'architecture d'un portail de gestion électronique de documents appelé **Arkevia**, puis à identifier et corriger les problèmes et anomalies détectés, ainsi qu'à proposer des axes d'amélioration suite à l'audit architectural réalisé. Ensuite, dans une seconde phase, les travaux ont porté sur le mécanisme de notification chargé d'informer les utilisateurs des documents récemment déposés dans leurs coffres-forts Arkevia. En effet, étant donné l'importance que porte Cegedim à la qualité de ses solutions, il a été convenu de refondre ce mécanisme et de le rendre indépendant de l'application mère Arkevia. L'application produite est principalement basée sur une approche de traitement par lots multithread via les technologies Spring Boot et Spring Batch.

Mot clés :
spring, batch, processing, sirh, ged.

Abstract

This report summarizes the work carried out over a period of six months at Cegedim SRH, in the context of the approval of the end-of-study project for the Master's degree in Data Engineering and Software Development at the Faculty of Science in Rabat.

The first stage of the work consisted in auditing and analyzing the architecture of an electronic document management portal called **Arkevia**, then to identify and correct the problems and anomalies detected, as well as to propose lines of improvement according to the architectural audit. Then, in a second phase, the work focused on the notification mechanism responsible for informing users of documents recently deposited in their Arkevia safes. Given the importance Cegedim places on the quality of its solutions, it was agreed to overhaul this mechanism and make it independent of the Arkevia main application. The resulting application is primarily based on a multi-threaded batch approach using Spring Boot and Spring Batch technologies.

Sommaire

Liste des figures

Liste des tables

Glossaire des acronymes

Acronyme	Désignation
API	**A**pplication **P**rogramming **I**nterface
B2B	**B**usiness **to** **B**usiness
BPO	**B**usiness **P**rocess **O**utsourcing
BU	**B**usiness **U**nit
CA	**C**hiffre d'**a**ffaires
CI/CD	**C**ontinuous **I**ntegration / **C**ontinuous **D**elivery
CSS	**C**ascading **S**tyle **S**heets
EDI	**E**lectronic **D**ata **I**nterchange
GED	**G**estion **É**lectronique des **D**ocuments
GTA	**G**estion des **T**emps et **A**ctivités
JVM	**J**ava **V**irtual **M**achine
ORM	**O**bject-**R**elational **M**apping
POJO	**P**lain **O**ld **J**ava **O**bject
QA	**Q**uality **A**ssurance
R&D	**R**esearch and **D**evelopment
RDP	**R**emote **D**esktop **P**rotocol
RH	**R**essources **H**umaines

Suite à la page suivante

SaaS	**S**oftware **a**s **a** **S**ervice
SASS	**S**yntactically **A**wesome **S**tyle **S**heets
SGBDR	**S**ystème de **G**estion de **B**ases de **D**onnées **R**elationnelles
SIRH	**S**ystème d'**I**nformation **R**essources **H**umaines
SRH	**S**ervice des **R**essources **H**umaines
UML	**U**nified **M**odeling **L**anguage

Chapitre 1 Méthodologie de gestion de projets

Introduction

Dans ce chapitre, nous aborderons les aspects de la gestion de projet, de la composition des équipes, des outils et des méthodes de développement de logiciels adoptés par Cegedim SRH.

1.1 Étude préliminaire

1.1.1 Origine du génie logiciel

> **Définition :** Le génie logiciel est la science de l'ingénieur qui s'intéresse aux procédés scientifiques de construction et d'entretien des logiciels, et bien évidemment à la « matière » même de cette construction : d'abord les programmes eux-mêmes, les fichiers et bases de données, les scripts de paramétrage nécessaires à l'exécution du programme, puis tout ce qui gravite autour (spécification de besoins et exigences des futurs utilisateurs, conception, tests, documentation pour les mainteneurs, le support technique et les usagers)[4].

À la fin des années 50, l'informatique est devenue de plus en plus populaire et s'est étendue à d'autres disciplines. Ceci en raison de l'impulsion des grands projets spatiaux et militaires, grands consommateurs de logiciels et imposant des exigences de qualité et de sûreté de fonctionnement et du fait que les informaticiens du génie logiciel étaient plus enclins et plus à même de développer des outils logiciels d'aide à la conduite des activités.

En octobre 1968, l'OTAN a organisé une première conférence, à Partenkirchen en Allemagne, sur l'industrialisation de l'élaboration du logiciel. C'est à cette occasion qu'est forgée l'expression « Software Engineering » pour donner une tournure délibérément industrielle au propos[5].

À la fin des années 60, les grands systèmes commerciaux ont démontré qu'il était difficile d'adapter à grande échelle les principes qui avaient été adéquats jusque là. Les grands projets dépassaient les budgets et les délais, ce fût alors « the software crisis » la crise du logiciel[1].

La conférence de l'OTAN a été immédiatement suivie par de nombreuses autres

conférences portant sur des thèmes liés au génie logiciel, tels que la fiabilité des logiciels, le test du logiciel, la spécification du logiciel, la maintenance du logiciel, etc.

En 1975 s'est tenue la première conférence véritablement dédiée à l'ensemble du génie logiciel, accompagnée d'une exposition de premiers outils, notamment pour l'aide au test. Depuis, l'IEEE tient une rencontre annuelle internationale. Cette première conférence a correspondu à la création, au sein de l'IEEE, d'un comité technique dédié au génie logiciel, animateur d'une série de conférences et ateliers récurrents et éditeur de deux revues dédiées au génie logiciel[5].

1.1.2 Comparaison des différentes méthodes de développement

1.1.2.1 Les approches traditionnelles

Dans une approche traditionnelle, le cycle de vie d'un logiciel passe par quatre phases principales : **initiation**, **développement**, **déploiement** et **exploitation**.
L'initiation vise à déterminer la mission du système et à effectuer des travaux exploratoires permettant de vérifier la pertinence du projet. **Le développement** est la phase qui prend en charge la réalisation du logiciel. Elle précise les besoins, réalise le produit et valide son fonctionnement. **Le déploiement** rend le produit accessible à ses utilisateurs, par la mise en service du logiciel dans son environnement de production. **L'exploitation** est la période de vie utile du produit au cours de laquelle des améliorations peuvent être apportées au produit.

1.1.2.2 Les méthodes agiles

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[6].

Le Manifeste Agile déclare quatre valeurs dans toute approche agile. Chaque méthode adopte ensuite sa propre terminologie et recommande un certain nombre de pratiques. Les quatre valeurs du Manifeste sont[2, 6] :

- 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.

1.1.2.3 Synthèse

Bien que les approches traditionnelles présentent divers avantages, elles ont aussi certaines limites, en particulier dans les environnements évolutifs. En effet, les

environnements évolutifs nécessitent des adaptations constantes. Avec une planification rigoureuse, il est difficile d'apporter des changements à cause de leurs impacts, et plus un changement est fait tard, plus il est coûteux. Cette situation tardive exige de modifier l'enchaînement des étapes antérieures et seuls les changements les plus importants seront opérés.

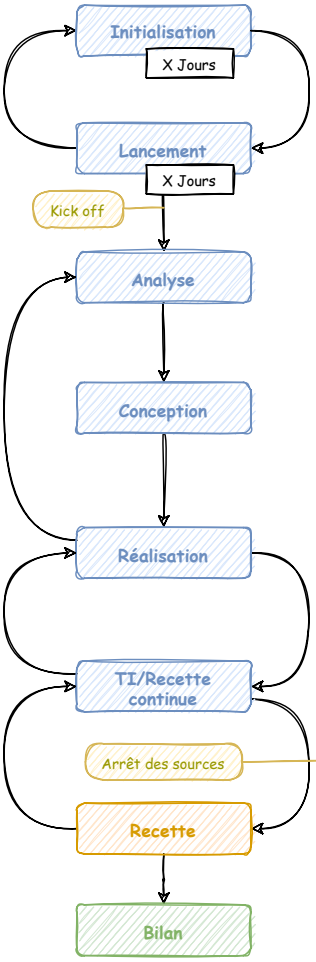
Selon une nouvelle étude menée par Organize Agile[3] auprès de professionnels de 19 pays, près de la moitié des organisations utilisent des méthodes agiles depuis trois ans ou plus. Ces entreprises utilisent principalement Agile comme méthodologie pour leurs programmes de réforme - connus sous le nom de transformations agiles. Il s'agit de grands changements organisationnels qui adoptent le travail agile au sein de petites équipes multidisciplinaires qui s'attachent à fournir des résultats de manière rapide, expérimentale et itérative, ce qui révèle l'impact positif de ces méthodes sur les flux de travail des projets informatiques à l'échelle mondiale. Au Maroc, selon un rapport d'étude publié en 2011 impliquant 48 organisations participantes représentant les secteurs privé et public a révélé que le taux de satisfaction de la maîtrise d'ouvrage est plus élevé pour les DSI adoptant une des méthodes agiles ce qui démontre leur pragmatisme et leur efficacité par rapport aux méthodes traditionnelles en cascade[7].

Les approches agiles et traditionnelles convergent et divergent sur différents aspects. L'expérience de l'équipe peut faire diminuer les risques et contribuer à minimiser les exigences de suivi. Les approbations et le manque d'autonomie des équipes peuvent ralentir le projet. Une organisation peu hiérarchisée, ayant une culture collégiale, intégrera difficilement une structure traditionnelle.

1.2 Conduite de projet

1.2.1 Modèle de livraison

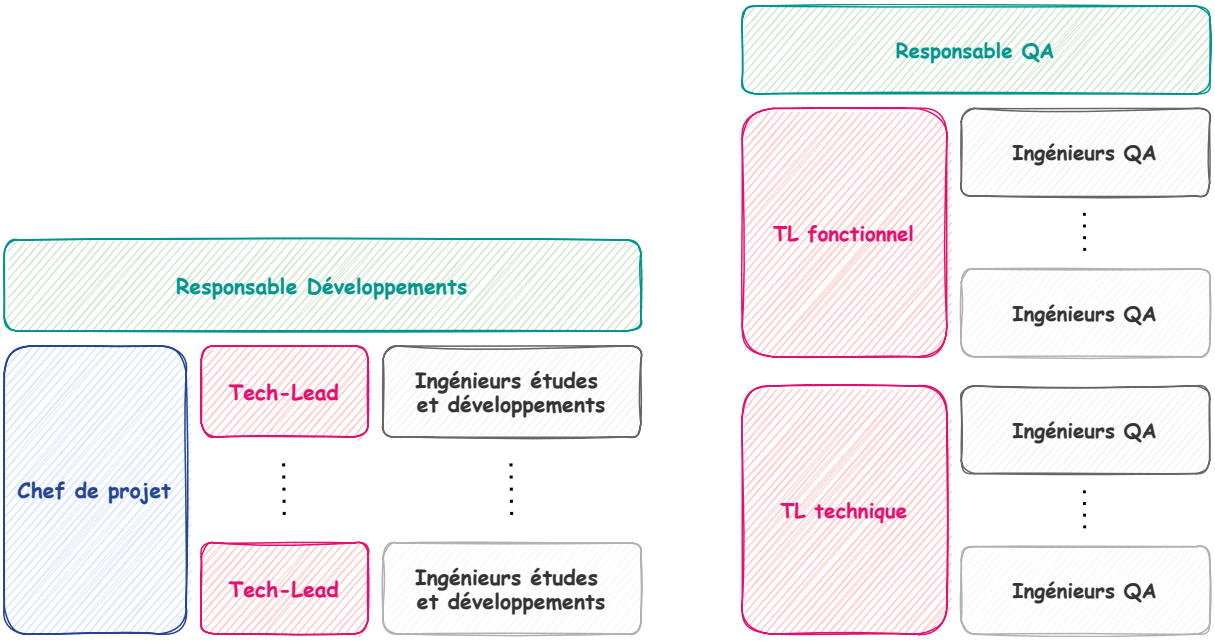
Aujourd'hui, Cegedim adopte une approche de livraison agile que tout projet est censé suivre. Le schéma ci-dessous montre les différentes phases de ce modèle (voir figure 1.1) :



Acteurs	Objectifs	Livrables
Responsables : PO **Contributeurs** : CP DEV	• Présentation du périmètre fonctionnel • Questions / Réponses • Identification des rôles	• SPEC • JIRA Projet initialisé • Mail de notification de lancement du projet
Responsables : CP DEV, Team Lead QA **Contributeurs** : Tech-Lead, Ingénieur QA	• Validation des spécifications fonctionnelles • Macro chiffrage	• Support de Kick off • SPEC détaillée • Planning prévisionnel • Initialisation des documents projets • Planning prévisionnel QA
Responsables : CP DEV, Team Lead QA **Contributeurs** : Tech-Lead, Ingénieur QA	• Identification des risques • Découpage détaillé des tâches • Chiffrage détaillé	• MAJ du projet JIRA • Spécifications techniques • Planification détaillée • MAJ de la Matrice de Risque
Coordinateurs : CP DEV, CP QA **Responsables** : Développeur, Ingénieur QA **Contributeurs** : Tech-Lead, Team Lead QA	• Réalisation de la conception technique • Cadrage des cas de test	• MAJ Spécifications techniques • Plan de TI • MAJ de la Matrice de Risque • Plan de test QA
Coordinateurs : CP DEV, CP QA **Responsables** : Développeur, Ingénieur QA **Contributeurs** : PO, Tech-Lead, Team Lead QA	• Réalisation des développements • Réalisation des Tests unitaires • Audit de code • Rédaction des scénarios de test QA	• Code source • MAJ Spécifications techniques • Preuves de test • Rapport d'audit du code • Démo au PO & QA • Référentiel de test
Responsables : CP DEV, Team Lead QA **Contributeurs** : Tech-Lead, Ingénieur QA	• Réalisation des TI • Report du code sur Patch • Livraison des paramétrages • Validation des exigences fonctionnelles	• CR de livraison • MAJ des documents projets • MAJ des jiras avec des preuves de test de RC
Responsables : Team Lead QA **Contributeurs** : Ingénieur QA	• Déroulement du plan de test • Validation des exigences fonctionnelles	• Rapport de recette
Responsables : CP DEV **Contributeurs** : PO, QA, TL, Équipe DEV	• Identifier les axes d'amélioration • Définir les actions correctifs	• Indicateurs clés • MAJ du plan d'amélioration continue

Figure 1.1. Modèle de livraison

Les acteurs impliqués dans le processus de livraison peuvent être répartis en deux grandes divisions (voir figure 1.2, tableaux A.1 et 1.2) :



(a) Organisation cible dév (b) Organisation cible QA

Figure 1.2. Modèle organisationnel visé pour les équipes de développement et de QA

Le tableau suivant (A.1) décrit les différentes activités et tâches confiées aux membres de l'équipe de développement impliqués dans le processus de livraison :

Table 1.1 : Responsabilités et missions des différents acteurs de l'équipe de dév

Équipe de développement
Responsable Développements • Accompagner le chef de projet dans la gestion des projets. • Produire des indicateurs sur l'activité de développement. • Suivi des risques et gestion des alertes. • Travailler en collaboration avec les autres responsables de division.
Chef de projet • Valider les spécifications fonctionnelles. • Assurer le suivi du processus de livraison et des indicateurs clés. • Élaboration du plan de charge et planification des équipes. • Garant des engagements (Qualité, Délai, Coût).

Suite à la page suivante

Table 1.1 : Responsabilités et missions des différents acteurs de l'équipe de dév (Suite)

Tech-Lead
• Responsable de la qualité technique (Audit, conception, etc.). • Responsable des bonnes pratiques de développement. • Encadrement et assistance technique. • Participer aux développements. • Assister le chef de projet pour les estimations de charge.
Ingénieur étude et développement
• Corriger les anomalies et développer de nouveaux modules. • Respecter les bonnes pratiques de développement. • Participer à la définition de la couverture de tests techniques. • Rédiger des documents techniques.

Le tableau suivant (1.2) décrit les différentes activités et tâches assignées aux membres de l'équipe d'assurance qualité (QA) participant au processus de livraison :

Table 1.2 : Responsabilités et missions des différents acteurs de l'équipe QA

Équipe QA
Responsable QA
• Établir et piloter une stratégie de test. • Accompagner le Team Lead dans la gestion des projets. • Contribuer à l'amélioration des processus de test. • Produire des indicateurs sur l'activité de testing. • Suivi des risques et gestion des alertes. • Travailler en collaboration avec les autres responsables de division.

Suite à la page suivante

Table 1.2 : Responsabilités et missions des différents acteurs de l'équipe QA (Suite)

TL fonctionnel/technique • Responsable du suivi du processus de livraison et des indicateurs clés. • Élaboration du plan de charge et la planification des équipes. • Accompagner et suivre les testeurs dans la mise en place des bonnes pratiques et des outils. • Accompagner les ingénieurs QA dans l'élaboration des plans de test. • Réaliser le bilan des activités. • Accompagner l'intégration des nouveaux arrivants et veiller à la montée en compétence des équipes. • Participer aux activités de test.
Ingénieur QA • Formaliser des scénarios de test fonctionnels et automatisés. • Valider et vérifier le développement de l'application. • Respecter les bonnes pratiques de testing. • Rédiger des documents fonctionnels.

1.2.2 Déroulement de projets - Méthode SCRUM

Depuis 2019, les projets de R&D ont progressivement passés en mode Agile. Ils utilisent la partie Agile de JIRA (catégorie Software). Le mode Agile permet de gérer les projets en mode Scrum ou Kanban. Pour l'instant, Cegedim SRH utilise le mode Scrum.

1.2.2.1 L'équipe Agile

L'équipe Agile est une équipe qui entend être totalement autonome. Elle rassemble toutes les compétences nécessaires pour faire évoluer le produit. Le maître mot d'une équipe agile est la coopération. En effet, les membres de l'équipe ne travaillent pas séparément sur les fonctionnalités qui leur sont assignées, mais ensemble, y compris pour l'identification, la définition, la réalisation et les tests des fonctionnalités.

Les capacités d'écoute et d'entraide de l'équipe facilitent la montée en compétence. Chaque membre devient petit à petit plus ou moins polyvalent et en capacité d'aider les autres dans la réalisation des différentes fonctionnalités. L'équipe devient donc de plus en plus performante et efficace, tout en améliorant également le ressenti et les conditions de travail de chacun.

> Une équipe Agile est en perpétuelle progression et évolution.

Il n'y a pas de composition type pour une équipe Agile. Celle-ci dépend totalement du produit à réaliser. Certains rôles clefs sont néanmoins nécessaires au bon fonctionnement de l'équipe. La composition d'une équipe Agile au sein de la R&D SRH suivra à minima le modèle suivant :

- **Equipe Agile R&D**
  - 1 Product Owner ;
  - 1 Scrum Master (optionnel mais conseillé) ;
  - X développeurs (il est recommandé d'avoir +2) ;
  - 1 QA (optionnel) ;
  - X testeur(s)

Bien entendu, selon les produits, cette structure sera susceptible d'évoluer.

1.2.2.2 Le Sprint

Le Sprint agile est le cœur de la méthode SCRUM. Tous les développements sont réalisés de manière incrémentale au sein des Sprints. Un périmètre de développement - l'objectif du Sprint - est défini au début du Sprint lors du **Sprint Planning** avec la liste des **User Stories** à traiter pendant le Sprint.

À la conclusion d'un Sprint, le bilan est réalisé lors du **Sprint Review**, puis un nouveau cycle démarre avec un nouveau Sprint. La périodicité d'un Sprint dépendra de l'équipe Agile, mais sera généralement entre 2 et 4 semaines.

1.2.2.3 Les Rituels

Les rituels (ou cérémonies) sont des réunions de travail et de suivi qui viennent rythmer un Sprint. Chaque réunion joue un rôle précis dans le Sprint et correspond à une temporalité particulière.

- **Sprint Planning** : Le Sprint Planning est la réunion de lancement d'un Sprint. Au début de chaque Sprint, le Product Owner présente les User Stories qu'il souhaiterait intégrer au Sprint. Les User Stories sont examinées une par une dans l'ordre de priorité du Backlog jusqu'à ce que le Sprint soit complet en termes de vélocité. L'examen d'une User Story suit les étapes suivantes :
  - Lecture de la User Story par le Product Owner.
  - Questions des développeurs (optionnel).
  - Poker Planning : les développeurs estiment le temps nécessaire pour le développement de la User Story en votant chacun pour une durée. Pour passer à l'étape suivante, il doit y avoir un consensus sur l'estimation. Si tel n'est

pas le cas, les développeurs ayant donné les estimations extrêmes doivent s'expliquer et un nouveau vote est réalisé.
- Intégration de la User Story au Sprint : passage en "A faire" dans le Board de l'équipe.

> **Note :** Le Sprint Planning a lieu impérativement le premier jour du Sprint, de préférence le matin afin d'optimiser un maximum le temps de travail de l'équipe sur le Sprint.

> **Remarque :** A l'étape "Questions", si les développeurs ne comprennent pas ce qu'il faut faire, la User Story est replacée dans le Backlog en vue d'être détaillée plus avant par le Product Owner. De même, lors du "Poker Planning", si ces derniers n'arrivent pas à se mettre d'accord sur une estimation de temps commune.

- **Daily Stand-Up Meeting** : Le Daily Stand-up meeting est une réunion quotidienne très courte (entre 5 et 15 minutes) à heure et lieu fixes qui rassemble l'ensemble des membres de l'équipe Agile où chaque membre de l'équipe prend la parole à tour de rôle. Chacun doit expliquer ce qu'il a fait depuis l'itération précédente du Stand-up meeting ou le Sprint Planning et ce qu'il prévoit faire jusqu'au prochain Stand-up meeting.

  > **Remarque :** Le temps de parole de chacun ne doit généralement pas dépasser deux minutes.

  Le Stand-up meeting a pour but de favoriser la circulation des informations au sein de l'équipe. Elle permet à l'ensemble de l'équipe d'avoir une vue complète de l'avancée des tâches et d'identifier d'éventuels points de blocages. Ces échanges dynamiques contribuent également à la cohésion et l'implication de l'équipe.

  > **Attention :**
  > - La tenue du Stand-up meeting n'est pas facultative.
  > - Si un sujet est susceptible de déborder, il doit être traité en dehors du cadre du Stand-up meeting avec les membres concernés.

- **Sprint Retrospective** : La Sprint Retrospective est la réunion d'analyse du déroulement du Sprint par l'équipe. Elle est positionnée à la fin de chaque sprint, après le Sprint Review. L'idée pour l'équipe est de capitaliser sur le vécu du Sprint écoulé pour adapter son organisation dans le but de renforcer son efficacité. C'est un élément clef dans le processus de progression et d'apprentissage de l'équipe.

L'équipe doit définir un plan d'action et d'amélioration à partir des constats et idées remontés pendant la réunion.

- **Sprint Review** : Le Sprint Review est une réunion qui se situe en toute fin d'un Sprint juste avant le Sprint Retrospective. Elle a pour but de présenter le travail réalisé durant le Sprint courant. Le but est d'obtenir un maximum de retours sur le réalisé afin d'assurer qu'il est bien en accord avec les attentes.
  Si l'avancée le permet, elle s'agrémente généralement en fin de Sprint Review d'une démonstration des nouveautés. Si tel n'est pas le cas, la démonstration peut être effectuer séparément du Sprint Review.

> **Note :** Cette réunion inclut non seulement l'équipe Agile mais aussi les parties prenantes et les décideurs.

1.2.3 Processus de développement

Compte tenu de la diversité des applications développées par Cegedim SRH et afin de mener à bien la mise en place des solutions, cette dernière adopte un processus de développement où tout projet est amené à suivre. Je ne ferai référence dans cette partie que sur un périmètre limité de ce processus, et sur lequel je suis intervenu à collaborer et échanger.

- **Analyse fonctionnelle et définition des objectifs** : Lors de cette phase préalable au démarrage du projet, les parties prenantes définissent ensemble :
  - les objectifs et la portée du projet,
  - les livrables attendus,
  - les délais souhaités,
  - le degré de souplesse qui pourra être accordé.
- **Étude de faisabilité et formalisation des spécifications techniques** : Une étude de faisabilité peut être menée afin de cerner les contraintes susceptibles de peser sur la mise en œuvre du projet. Ensuite, les spécifications techniques sont formalisées, faisant état des méthodes, processus et technologies qui seront utilisés pour répondre aux contraintes du projet.
- **Découpage et chiffrage** : Il s'agit d'établir la liste des tâches en associant les besoins et les coûts correspondants, tout en incluant les sous-tâches et les tâches induites par la réalisation afin de chiffrer au plus juste le projet.
- **Planification** : La planification vise à ordonner les tâches et à indiquer leur enchaînement logique en tenant compte des ressources disponibles et de leur charge de travail maximale.
- **Codage** : La phase de codage, également appelée programmation, consiste à traduire en code les fonctionnalités et les exigences techniques préalablement

définies.

- **Tests unitaires** : Le concept de test unitaire n'est pas un élément nouveau. Depuis les prémices de l'informatique, les tests font partie de l'activité quotidienne d'un développeur. Ce qui est nouveau aujourd'hui, c'est que cette activité, notamment les tests unitaires, est placée au cœur du processus de conception. En effet, un test unitaire permet de valider la conformité de chaque composant logiciel pris comme une unité par rapport à sa spécification détaillée. Autrement dit, un scénario de test unitaire ressemble à une expérience scientifique dans laquelle une hypothèse est examinée en fonction de trois éléments clés :
  1. Les données en entrée.
  2. L'objet à tester.
  3. Les observations attendues.
- **Audit de code** : L'étape d'audit de code permet de s'assurer de la qualité du codage en vérifiant que chaque module ou sous-ensemble de la solution informatique est conforme aux règles et bonnes pratiques de développement logiciel. Dans le cadre de ce sujet, nous nous référerons plus particulièrement aux règles établies par Cegedim SRH (voir la checklist **??** en annexe) dans le but de systématiser les bonnes pratiques et d'éviter les erreurs classiques de développement au sein des équipes de dév.
- **Recette** La phase de recette est le processus de validation par l'équipe de validation et acceptance (QA) de la conformité des livrables avec les spécifications initiales.
- **Documentation** : À l'issue de la recette, une documentation de projet est produite afin de rassembler les informations nécessaires à l'utilisation de la solution informatique et en vue de ses développements ultérieurs.
- **Déploiement** : Une fois le projet qualifié, la solution informatique peut être déployée : il s'agit de la livraison du produit final et de sa mise en service.

1.2.4 Gestion du workflow git

Le workflow de travail pour les branches GIT choisi au niveau de Cegedim SRH est Gitflow. Git Flow est un modèle de dépôt git permettant d'améliorer les processus de développement et de mise en production d'un projet.
Voici un schéma présentant l'organisation du dépôt git ainsi que les différentes interactions qu'il peut y avoir entre chaque branche :

1.3 Planification et suivi du projet

La planification est parmi les phases d'avant-projet. Elle consiste non seulement à délimiter le périmètre temporel du projet, mais aussi à prévoir le déroulement des activités tout au long de la période allouée au stage.

1.3.1 Diagramme de Gantt

La figure suivante détaille la planification prévisionnelle du projet (voir figure 1.3) :

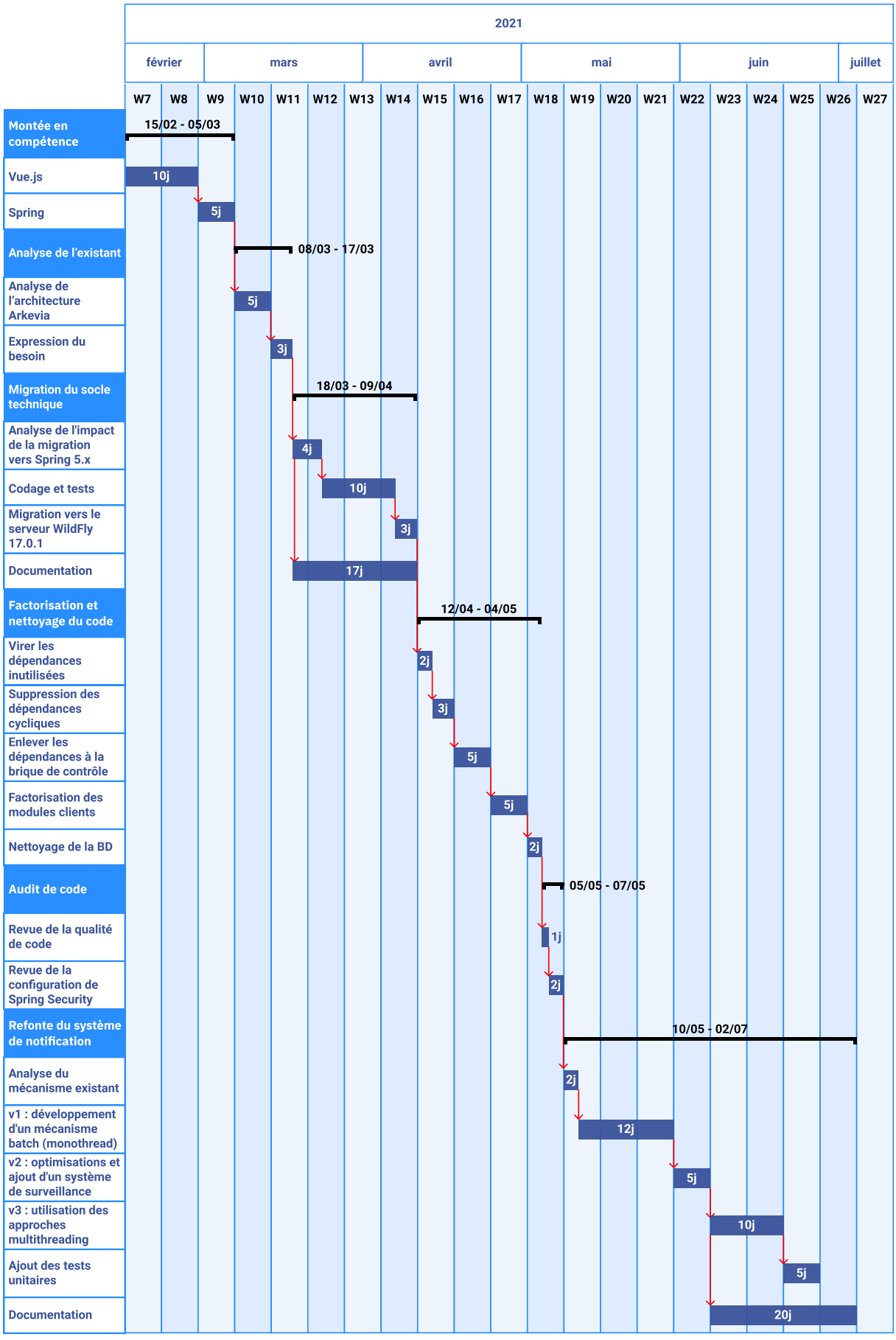


Figure 1.3. Diagramme de Gantt

Conclusion

La gestion de projets comme l'on peut le constater s'avère être d'une grande nécessité. Il est nécessaire pour mener à bien un projet de prendre en compte plusieurs dimensions dont toutes ne sont pas forcément quantifiables.

Cegedim SRH adopte désormais la méthode Scrum, l'une des méthodes Agile les plus répandues, pour le développement de ses propres solutions. L'Agilité, résolument basée sur l'efficacité et la création de valeur ajoutée, place véritablement clients et éditeurs dans un mode collaboratif avec comme vision un objectif commun.

En conséquence, grâce à son expertise avérée de cette méthodologie. Cegedim SRH a pu réduire les délais de conception et de production sans pour autant compromettre la qualité du produit final.

ANNEXES

A.1 Audit de code : checklist élaborée par Cegedim SRH

Table A.1 : Extrait des règles et pratiques de développement logiciel instaurées par Cegedim SRH

Réf.	**Description de la règle**
RG_DEV_GN_001	Single responsibility principle : Une fonction doit être responsable d'une et d'une seule tâche. Lorsqu'une fonction fait plus d'une tâche, elle est plus difficile à composer, à tester et à comprendre.
RG_DEV_GN_002	Don't repeat yourself : Souvent, nous avons du code en double parce que nous avons deux ou plusieurs choses légèrement différentes, qui partagent beaucoup en commun, mais leurs différences nous obligent à avoir deux ou plusieurs fonctions distinctes qui font en grande partie les mêmes choses. Supprimer le code en double signifie créer une abstraction qui peut gérer cet ensemble de choses différentes avec une seule fonction / module / classe. Factoriser le code en double.
RG_DEV_GN_003	Open/Closed principle : Les nouvelles fonctionnalités ne doivent pas modifier le code existant.
RG_DEV_GN_004	Test unitaire : Chaque fonction doit être testée avec au minimum un cas passant et un cas non passant.
RG_DEV_GN_005	Respect des warnings eslint sonarLint : Veiller au maximum à ce qu'il n'y a pas de warnings eslint et/ou sonarlint dans la classes/ou parties de classes impactées dans le cadre d'une évolution, l'objectif est d'atteindre 0 Warnings dans le possible.
RG_DEV_GN_006	Correction des NullPointerException : Éviter de contourner le problème de nullité en ajoutant des tests, et aller plus en profondeur dans le diagnostic pour détecter l'origine du problème et le résoudre d'une manière radicale.

Suite à la page suivante

Table A.1 : Extrait des règles et pratiques de développement logiciel instaurées par Cegedim SRH (Suite)

RG_DEV_GN_007	Arguments de fonctions : Limiter le nombre de paramètres d'une fonction est extrêmement important, un ou deux arguments est le cas idéal, et trois devraient être évités si possible. Rien de plus que cela devrait être utilisé. Utiliser des objets si nous avons besoin de beaucoup d'arguments.
RG_DEV_GN_008	Cas des dépassements de capacité : Éviter de contourner le problème en tronquant des valeurs, et aller plus en profondeur dans le diagnostic pour détecter l'origine du problème et le résoudre d'une manière radicale.

A.2 Outils utilisés pour la réalisation de ce rapport

Adobe Illustrator Draw.io LaTeX Figma Vs code

Références

[1] Dino Mandrioli **Carlo Ghezzi** Mehdi Jazayeri. *Fundamentals of Software Engineering*. 2e éd. Pearson, Prentice Hall, 2003, p. 1-5. **isbn** : 0133056996 , 8120322428.

[2] L'equipe produit **CMMI**. *CMMI® pour le développement, Version 1.3 - Amélioration des processus pour le développement de meilleurs produits et services*. 2011. **url** : https://resources.sei.cmu.edu/asset_files/WhitePaper/2010_019_001_28803.pdf.

[3] **Consultancy.eu**. *Half of companies applying Agile methodologies & practices*. consulté le 26 septembre 2021. Organize Agile, Consultancy.eu analysis. **url** : https://www.consultancy.eu/news/4153/half-of-companies-applying-agile-methodologies-practices.

[4] Jacques **PRINTZ**. *LOGICIELS*. consulté le 26 septembre 2021. Encyclopædia Universalis [en ligne]. **url** : https://www.universalis.fr/encyclopedie/logiciels/.

[5] Jean-Claude **Rault**. *La Lettre d'ADELI n°76 – Été 2009*. consulté le 26 septembre 2021. ADELI [en ligne]. **url** : https://espaces-numeriques.org/wp-content/uploads/2019/01/l76p08.pdf.

[6] Véronique MESSAGER **ROTA**. *Gestion de projet - Vers les méthodes agiles*. Eyrolles, 2008. **isbn** : 978-2-212-12165-0.

[7] Badr **TALAGHZI**. « Processus du développement logiciel - Paradigmes & Perspectives ». In : *Omn.univ.europ.* Editions universitaires européennes, 2016. **isbn** : 9783639543124, 978-3639543124.