



Ministère de l'Enseignement Supérieur et de la recherche scientifique
Direction Générale des Etudes Technologiques
Institut Supérieur des Etudes Technologiques de Sfax
Département Technologies de l'Informatique



Codestage :
Soutenu le :

STAGE DE FIN D'ETUDES

Application Web « La gestion et le suivi de la consommation papiers »

Organisme d'accueil : ISET Sfax

Elaboré par : Abd Mouleh Lotfi & Aouissaoui Fahmi

JURY

Président :

Rapporteur :

Encadreur1 : Zayeni Mohamed

Encadreur2 : Fourati Hounaida

Encadreur3 : Halouani Naima

Année Universitaire : 2018/2019

TABLE DES MATIERES

INTRODUCTION GENERALE	6
CHAPITRE 1 : ETUDE PREALABLE.....	8
INTRODUCTION.....	9
1. PRESENTATION DU PROJET	9
2. PRESENTATION DE L'ORGANISME D'ACCUEIL.....	9
3. ETUDE DE L'EXISTANT	10
3.1 Analyse de l'existant.....	10
3.2 Critique de l'existant	11
3.3 Objectifs du projet	11
3.4 Solution proposée	12
CONCLUSION.....	13
CHAPITRE 2 : ETAT DE L'ART	14
INTRODUCTION.....	15
1. AXE DE RECHERCHE	15
2. METHODOLOGIE ET APPROCHES ADOPTEES	16
3. LES TECHNIQUES DE PRODUCTION LOGICIELLE.....	18
3.1 La programmation modulaire et orientée composants.....	18
3.2 Outils d'automatisation de la construction Java.....	22
3.3 Framework back-end.....	25
3.4 Framework front-end.....	27
3.5 La programmation orientée aspects	28
3.6 La gestion de version.....	31
4. L'ARCHITECTURE APPLICATIVE FUTURE.....	34
CONCLUSION.....	35
CHAPITRE 3 : ANALYSES ET SPECIFICATIONS	36
INTRODUCTION.....	37
1. SPECIFICATION DES BESOINS	37
1.1 Besoins fonctionnels	37
1.2 Besoins non fonctionnels	37
2. BACKLOG GENERAL DU PRODUIT	38

3.	CHRONOGRAMME DE TRAVAIL	39
	CONCLUSION.....	39
CHAPITRE 4 : ETUDE CONCEPTUELLE (SPRINT 0).....		40
	INTRODUCTION.....	41
1.	CONCEPTION GENERALE.....	41
1.1	<i>Architecture logique</i>	41
1.2	<i>Architecture physique</i>	44
2.	CONCEPTION DETAILLE	45
2.1	<i>Diagramme de cas d'utilisation</i>	45
2.2	<i>Diagramme de classes</i>	46
	CONCLUSION.....	48
CHAPITRE 5 : MISE EN PLACE D'UN SOCLE LOGICIELLE (SPRINT 1)		49
	INTRODUCTION.....	50
1.	PLANIFICATION PREVISIONNELLE	50
2.	ANALYSE ET SPECIFICATION DES BESOINS	50
2.1	<i>Diagramme de cas d'utilisation « Authentification »</i>	50
2.2	<i>Diagramme de cas d'utilisation « Gestion des utilisateurs »</i>	51
3.	CONCEPTION	53
3.1	<i>Diagramme de classes</i>	53
3.2	<i>Diagramme d'activité</i>	53
3.3	<i>Diagramme de séquences</i>	55

Liste des figures

Figure 1: organigramme de l'organisme	10
Figure 2: Cycle de vie DevOps	15
Figure 3: processus de travail avec Scrum	16
Figure 4: Equipe Scrum	17
Figure 5: Programmation modulaire	18
Figure 6: cycle d'exécution d'un Filter	28
Figure 7: cycle d'exécution d'un Interceptor	29
Figure 8: architecture des aspects	30
Figure 9: Architecture de Git et SVN	33
Figure 10: schéma d'interaction des différents outils et frameworks de développement	34
Figure 11: Architecture logique	42
Figure 12: Diagramme de paquetage	43
Figure 13: Architecture physique	44
Figure 14: Diagramme de déploiement	45
Figure 15: Diagramme de cas d'utilisation générale	46
Figure 16: Diagramme des classes	47
Figure 17: Diagramme de cas d'utilisation « Authentification »	51
Figure 18: Diagramme de cas d'utilisation « Gestion des utilisateurs »	52
Figure 19: Diagramme de classes « gestion des utilisateurs »	53
Figure 20: Diagramme d'activité « Authentification »	54
Figure 21: Diagramme d'activité « Ajouter utilisateur »	54
Figure 22: Diagramme de séquences « Authentification »	55
Figure 23: Diagramme de séquences « Affecter rôle »	56

Liste des tableaux

Tableau 1 : L'équipe Scrum	18
Tableau 2:Tableau comparatif sur la journalisation	20
Tableau 3 : Tableau comparatif sur la persistance.....	21
Tableau 4 : tableau comparatif sur le Reporting.....	22
Tableau 5 : tableau comparatif entre les outils d'automatisation	25
Tableau 6:Tableau comparatif sur les Framework back-end.....	26
Tableau 7:Tableau comparatif entre les Framework front-end	28
Tableau 8: Tableau comparatif entre les outils de La programmation orientée aspects .	31
Tableau 9: Tableau comparatif entre les outils de gestion de version	33
Tableau 10:Architecture logique cible.....	34
Tableau 11:BACKLOG général du produit.....	39
Tableau 12:Dictionnaire de données.....	48
Tableau 13:Planification de sprint	50
Tableau 14:Description textuelle « Authentification».....	51
Tableau 15: Description textuelle « Gestion des utilisateurs».....	52
Tableau 16: Description des attributs des tables user et rôle	53

Introduction générale

Le projet de fin d'études est une initiation à la vie professionnelle vu que c'est une opportunité pour valoriser nos compétences et nos connaissances dans le domaine informatique grâce à nos cursus universitaires en particulier à l'Institut Supérieure des Etudes Technologiques de Sfax. C'est le fruit de nos travaux en vue de l'obtention du diplôme de Mastère Professionnelle en Génie Logiciel et Nouvelles Technologie.

Aujourd'hui, les technologies de l'information changent le monde entier autour de nous, devenant désormais un facteur principal du développement qui touche tous les secteurs de la société.

L'éducation est parmi les domaines qui profitent de l'utilisation de ces nouvelles technologies et ce afin de faciliter tous les processus des établissements éducatifs.

Nous constatons que la situation des systèmes éducatifs actuels des instituts et des écoles présenter plusieurs problèmes qui ne devraient pas exister dans une époque d'émergence de la technologie.

Cependant, la diversification et la croissance des activités au sein des établissements universitaires ont fait grandir le besoin d'intégrer plusieurs technologies. La tâche de gérer efficacement toutes ces activités s'avère ainsi de plus en plus complexe et difficile. Pour surmonter ses propres difficultés, une entreprise a besoin d'utiliser des outils optimisés qui s'adaptent à la réalisation de toutes les activités en offrant des fonctionnalités riches et utiles.

Dans ce sens, l'Institut Supérieur des Études Technologiques de Sfax, tend à s'aligner sur cette nouvelle stratégie d'automatisation à travers le projet de mise en place d'un ERP dans le but d'homogénéiser le Système d'Information de l'ISSET avec un outil unique qui est capable de couvrir un large périmètre de gestion. Pour cela, il a été convenu de proposer un lot de sujets complémentaires pour la mise en place de cet ERP.

Dans ce contexte, il nous a été proposé d'élaborer une application web composée :

- D'un support logiciel (socle) sur lequel seront greffées des applications métiers.
- D'une application permettant de gérer le service tirage et assurer le suivi de la consommation de papier au sein de l'ISSET de Sfax dans le but de rationaliser les dépenses.

La première partie a été conçue et développée en mode transversal et collaboratif avec les autres sujets proposés. La deuxième partie est propre à un cadre métier spécifique à notre sujet.

Il est à noter que tous les modules développés (transversaux et spécifiques) ont été mis à disposition en open source sous un Docker propre de l'ISET.

Le présent rapport s'est évertué à rapporter la progression du travail élaboré et à détailler ses différentes phases. Il est structuré en (N) chapitres. Dans le premier chapitre nous commencerons par la présentation de l'organisme d'accueil « ISET Sfax », puis par un survol des différents problèmes dégagés et les solutions suggérées pour achever le travail demandé. Ensuite, nous aborderons dans le deuxième chapitre l'état de l'art qui illustre, à travers un travail de recherche, une étude des différentes technologies utilisées dans le domaine d'automatisation du développement de logiciel complétée par une étude comparative des outils disponibles sur laquelle nous nous sommes basés pour choisir une solution qui répond aux objectifs du projet.

Chapitre 1 : Etude préalable

Introduction

Ce chapitre est réservé à la présentation du cadre général du projet de fin d'études. Nous commençons par présenter l'organisme d'accueil. Nous présentons par la suite une description globale du projet. Et nous terminons par l'étude de l'existant.

1. Présentation du projet

Notre projet comprend deux grandes composantes :

- Une première composante technique imposée par un environnement de développement Web JEE qui consiste à concevoir et mettre en œuvre une plateforme de production et d'intégration de logiciels dans le but de mettre en place un ERP pour l'ISSET de Sfax. Il s'agit de produire un socle logiciel sur lequel seront greffées des applications métiers à travers l'utilisation de techniques spécifiques de programmation.
- Une deuxième composante se rapporte aux fonctionnalités métiers qui concernent un domaine de gestion particulier de l'ISSET à savoir la gestion et le suivi de la consommation papiers. Notre projet tout au long de son cycle de développement passe aussi par l'utilisation du modèle DevOps.

2. Présentation de l'organisme d'accueil

L'institut supérieur des études technologiques de Sfax (ISSET Sfax) est un établissement public doté de la personnalité civile et de l'autonomie financière. Relevant au ministère de l'Enseignement Supérieur et de la Recherche Scientifique, sa création a été décrétée par la loi n°92-51 du 18 mai 1992.

L'ISSET Sfax contient cinq départements :

- Sciences Économiques et Gestion
- Technologie de l'Informatique
- Génie des Procédés
- Génie Civil
- Génie Mécanique

Dans la figure ci-dessous nous avons présenté l'organigramme de l'organisme.

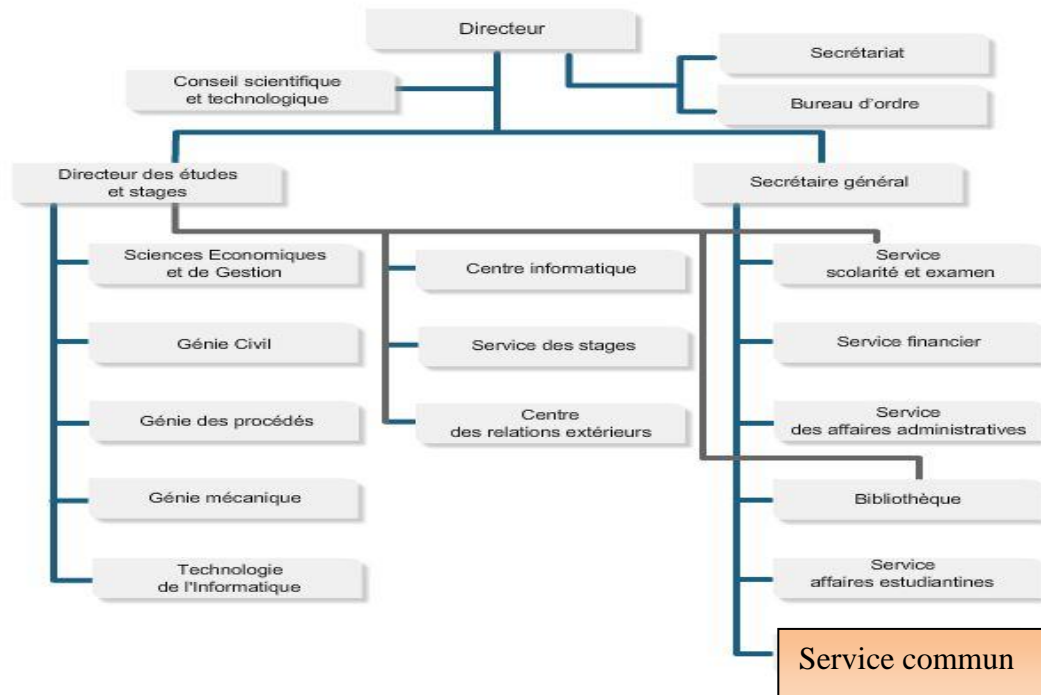


Figure 1: organigramme de l'organisme

3. Etude de l'existant

L'étude de l'existant est une phase importante pour bien comprendre le système actuel et définir ses objectifs.

3.1 Analyse de l'existant

La gestion du service de tirage au sein de l'ISET de Sfax se fait manuellement jusqu'à lors.

En effet :

- Les demandes de tirage se font directement au service tirage. Les enseignants déposent leurs documents en leur associant une fiche qui précise entre autres le nombre de copies demandées et la date voulue pour reprendre les copies.
- Les ouvriers du service tirage procèdent à l'opération de photocopie sans aucun ordonnancement préalable des différentes demandes de tirage.
- Les agents de tirage peuvent parfois procéder à certaines opérations d'entretien sur les machines tel que la recharge des cartouches d'encre ou le nettoyage des machines.
- L'approvisionnement en produits consommables tel que papiers, trombones, agrafe se fait par un carnet de fourniture adressé au magasin.

- En cas de panne des photocopieurs, le secrétaire général de l'institut contacte un prestataire de service pour faire une réparation des pannes.

3.2 Critique de l'existant

Durant ces dernières années plusieurs applications élémentaires ont été développées comme la gestion des emplois de temps, la gestion des charges, ...etc. l'inconvénient majeur c'est que ces applications ne disposent pas d'un référentiel commun ce qui engendre une redondance et incohérent de données et une faible communication d'information.

Bâties sur des environnements de développement diverses ce qui rend impossible leur intégration dans un outil logiciel unique pour l'établissement.

Coté métier, l'absence d'un outil informatique qui assure la gestion du service tirage, a engendré un travail anarchique qui pose beaucoup de problèmes dont nous citons les principaux :

- Des retards dans la livraison des copies ce qui peut nuire au bon déroulement des cours.
- Des pertes ou des oublis des document provoquant des retards.
- Des erreurs de classement des documents peut entraver et retarder le déroulement des cours.
- Une perte de temps due aux opérations manuelles de traitement, de tri et de distribution des papiers.
- La désorganisation des documents qui en résulte une difficulté de recherche de ces dernières.
- Le risque de perte de la traçabilité de documents.
- La consommation de papier extravagante.

3.3 Objectifs du projet

Nous avons en-déduit que le système actuel de gestion de tirage de l'ISSET de Sfax n'est pas performant. Il ralentit le travail, ainsi que le risque de la perte des documents est toujours présent. Cela cause une perte de temps et de ressources humaines. D'où la nécessité de concevoir et développer une solution logicielle permettant de dépasser ces problèmes et lacunes.

Le premier objectif de notre projet est de construire en collaboration avec les autres équipes de développement un socle logiciel commun qui servira de base pour y intégrer des modules relatifs à divers domaines métiers. Le socle de l'application comprend toutes les fonctionnalités standards et communes entre les équipes de développement entre autres une base de données globale et commune et aussi des composants transversaux tel que l'authentification, la persistance, la journalisation, ...etc.

Le deuxième objectif de notre projet vise à mettre en place un système performant pour assurer la gestion et le suivi du service tirage. La solution métier permet de :

- Déminuer les retards des cours.
- Éliminer le gaspillage du papier.
- Informatiser la gestion de tirage de documents.
- Assurer la traçabilité d'actions faites par les enseignants, les agents de tirage, ou l'administrateur de l'application.
- Suivre l'état des documents (en attentes, en cours de tirage, photocopiés).
- Réduire le taux de consommation de papiers.
- Assurer la préservation des documents.

3.4 Solution proposée

3.4.1 Solution technique

Permet de construire un socle logiciel commun en collaboration avec les autres équipes de développement qui servira de base pour y intégrer des modules relatifs à divers domaines métiers, cette solution technique assure d'implémenter une base de données centralisée et des composants transversaux, grâce à ce socle n'importe quel domaine métier sera utilisé dans chaque département d'ISSET de Sfax.

3.4.2 Solution métier

L'application que nous envisageons de développer permet de gérer le service tirage, le suivi de consommation de papier et de centraliser les données dans une seule base de données, qui sera le noyau mais extensible d'une future application riche avec des modules complémentaires plus conviviale.

Avec notre application permettra d'assurer :

- La bonne organisation grâce à une plateforme qui facilite la gestion de tirage.
- La réduction des dépenses par l'ISSET pour l'achat du papier.

- Une infrastructure adaptée pour ajouter d'autres modules.

Conclusion

Dans ce chapitre, nous avons présenté le contexte du projet. Puis nous avons présenté l'organisme d'accueil où nous avons effectué notre stage. Nous avons fait une étude et critique de l'existant et proposé notre solution adéquate.

Dans le chapitre suivant, nous allons présenter l'état d'art sur les différentes technologies et Framework de développement.

Chapitre 2 : Etat de l'art

Introduction

Dans ce chapitre, nous allons illustrer le résultat d'une recherche sur les différentes technologies utilisées dans l'automatisation de production de logiciel complété par une étude comparative entre les différents Framework et outils de développement. Le chapitre sera finalisé par une présentation de la solution adoptée.

1. Axe de recherche

Notre solution logicielle se comporte comme un projet modulaire élaboré par deux équipes de développement (Dev) et deux équipes d'exploitation (Ops) pour cela notre projet suit un cycle de vie DevOps qui est un ensemble de pratiques qui automatisent les processus entre les équipes Dev et Ops afin de leur permettre de développer, tester et livrer des logiciels plus rapidement et avec plus de fiabilité.

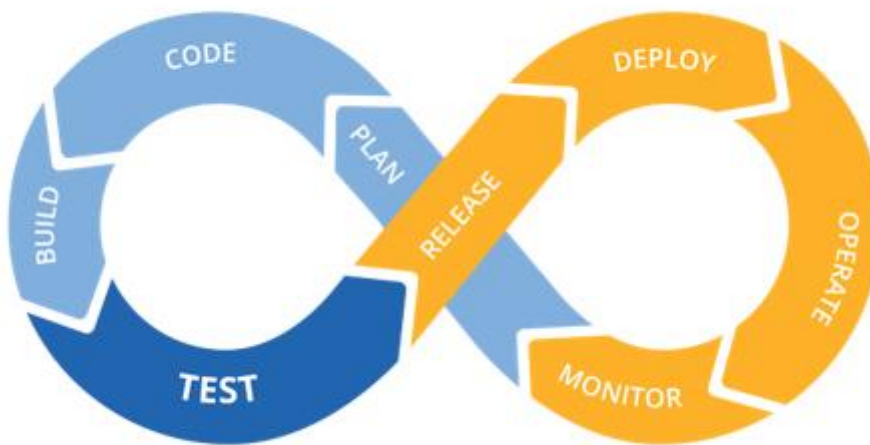


Figure 2: Cycle de vie DevOps

Nous avons représenté le DevOps comme un cycle de vie où l'on passe d'une phase de développement à une phase opérationnelle, dès que cette phase est terminée, un nouveau cycle démarre.

Nous allons focaliser notre recherche sur les outils et les Framework de la partie développement.

2. Méthodologie et approches adoptées

❖ Méthode Scrum

Les méthodes dites « agiles » ont trouvé leurs places au sein des entreprises, et principalement des startups. Cela permet une gestion réactive, itérative et incrémentale des différents projets de l'entreprise. La plus connue de ces méthodes est très certainement la méthode SCRUM. Pour bien conduire notre projet et nous assurer du bon déroulement des différentes phases, nous avons opté pour SCRUM comme une méthodologie de conception et de développement.

- SCRUM s'adapte bien avec le modèle DevOps en intégrant l'ensemble des principes agiles.

❖ Planification d'un projet avec Scrum

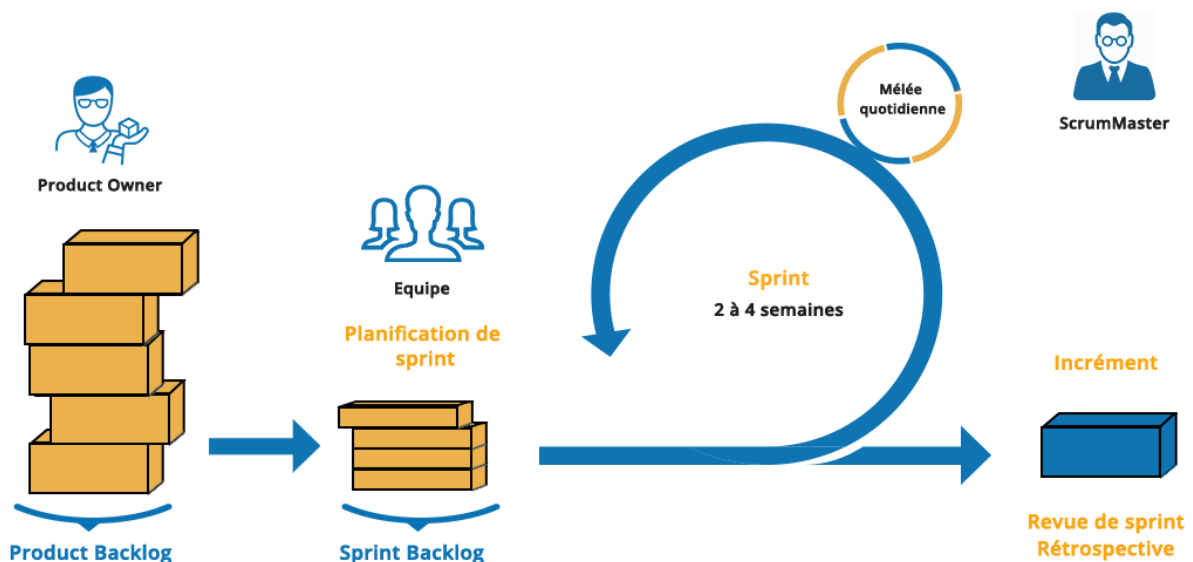


Figure 3: processus de travail avec Scrum

✚ **Planification du sprint** : S'appuie sur la planification de la « release » réalisée en pente. La première réunion du sprint ne se limite pas à la planification, on y trouve également les activités suivantes :

- Valider les « stories » du BACKLOG pris en compte dans le sprint et concevoir les solutions.
- Identifier et estimer les tâches.

✚ **Revue du sprint** : Permet de montrer les résultats du développement effectués au cours du sprint, seule une version opérationnelle est montrée.

✚ **Rétrospective** : Faite en interne en équipe (avec la présence du SCRUM Master), l'objectif est de comprendre ce qui n'a pas bien fonctionné dans le sprint, les erreurs commises et de prendre des décisions pour procéder aux améliorations.

✚ **SCRUM quotidien** : S'agit d'une réunion de synchronisation de l'équipe de développement qui se fait debout en 15 minutes maximum au cours de laquelle chacun répond principalement à 3 questions :

- Qu'est-ce que j'ai fait hier ?
- Qu'est-ce que je ferai aujourd'hui ?
- Quels obstacles me retardent ?

❖ Équipe et rôles

L'équipe a un rôle capital dans SCRUM : elle est constituée dans le but d'optimiser la flexibilité et la productivité. Pour cela, elle s'organise elle-même, et doit avoir toutes les compétences nécessaires au développement du produit. Elle est investie avec le pouvoir et l'autorité pour faire ce qu'elle a à faire.

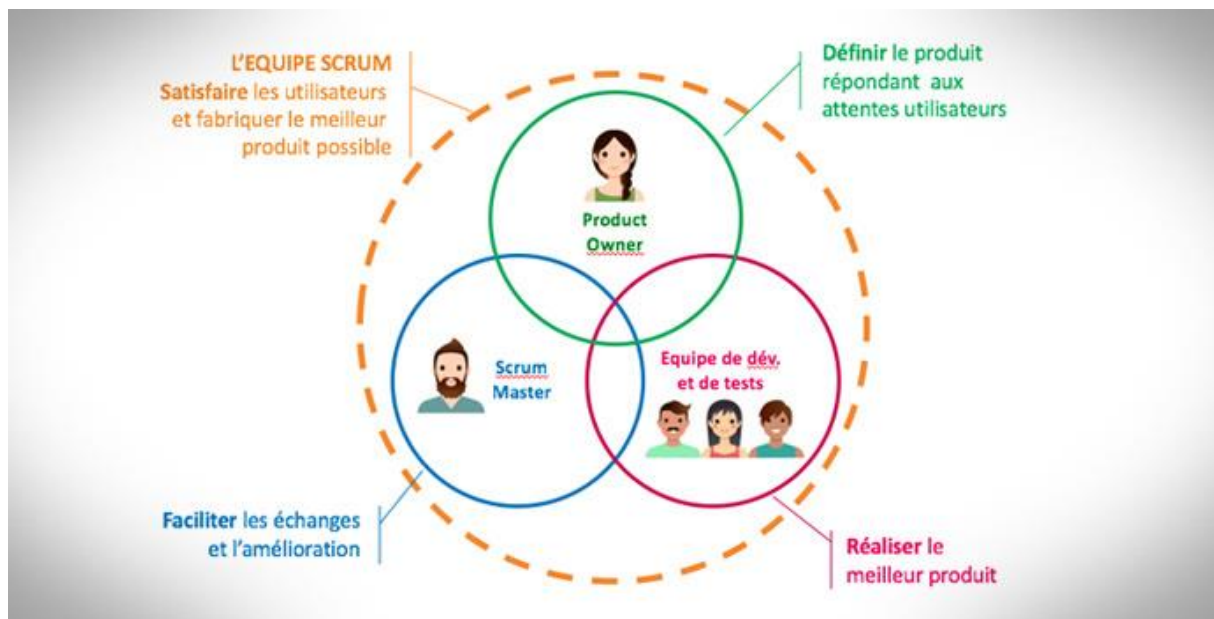


Figure 4: Equipe Scrum

Bref, SCRUM définit trois rôles qui sont :

- **Le Product Owner** (le propriétaire du produit) : c'est une personne qui porte la vision du produit à réaliser, généralement c'est un expert dans le domaine.
- **Le SCRUM Master** (le directeur de produit) : c'est la personne qui doit assurer le bon déroulement des différents sprints du release, et qui doit impérativement maîtriser SCRUM.

- **Le SCRUM Team** (l'équipe de SCRUM) : constitué des personnes qui seront chargées d'implémenter les différents besoins du client. Bien évidemment, cette équipe sera constituée des développeurs, des testeurs, ...etc.

Dans notre projet l'équipe SCRUM présentée comme le tableau suivant.

Product Owner	Mr. Jmal Ahmad
Scrum Master	Mr. Zayeni Mohamed, Mme. Fourati Hounaida, Mme. Halwani Naiima
Scrum Team	Aouissaoui Fahmi, Nasr Doula, AbdMouleh Lotfi, Mesbehi Hiba, Bouthaina, Bouali Tasnim, Nabila

Tableau 1 : L'équipe Scrum

3. Les techniques de production logicielle

3.1 La programmation modulaire et orientée composants

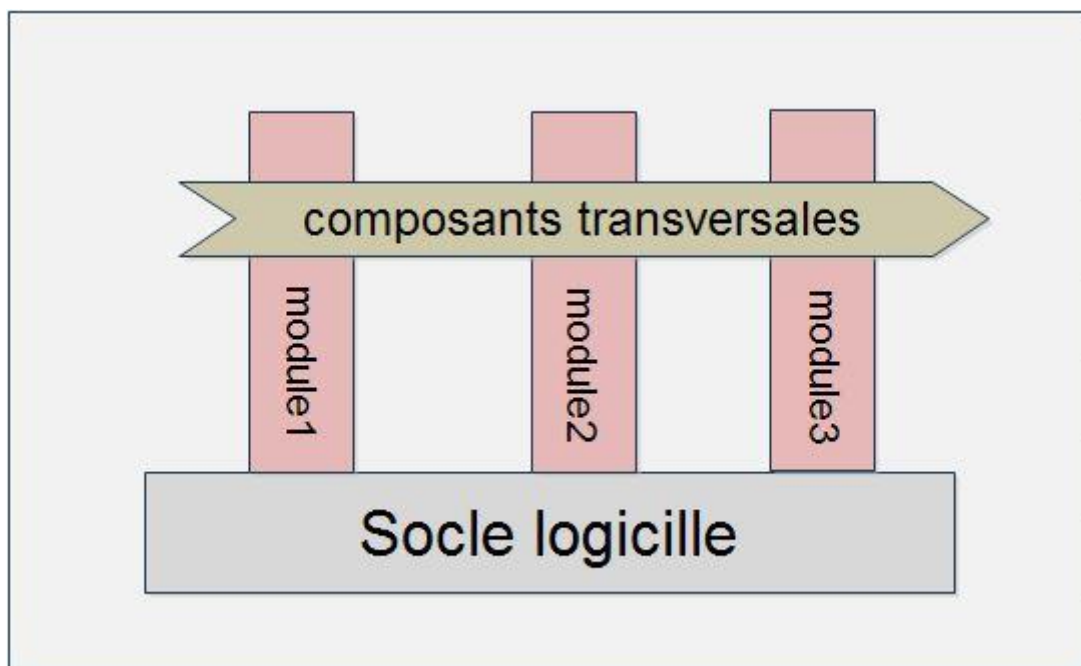


Figure 5: Programmation modulaire

3.1.1 Définition de la programmation modulaire

La programmation modulaire reprend l'idée de fabriquer un produit (programme) à partir de composants (modules). Cette technique décompose une grande application en modules, groupes de fonctions, de méthodes et de traitements, pour pouvoir les développer et les améliorer indépendamment, puis les réutiliser dans d'autres applications.

Ce mode de programmation présente plusieurs avantages, nous citons par exemple :

- Diviser le travail sur plusieurs équipes.

- Créer des morceaux de programme indépendants de la problématique globale, donc réutilisables pour d'autres logiciels.
- Supprimer les risques d'erreurs qu'on avait en reprogrammant ces morceaux à chaque fois.
- Réduire la taille des programmes sources améliorant la lisibilité et réduisant les temps de compilation.
- Développer des modules paramétrables et/ou polymorphes.

Du coup où le domaine métier est complexe, sa modélisation peut contenir des centaines de Services, Entités et Objets-valeurs. Pour en simplifier l'organisation, le concept de «modules» permet de structurer les fonctionnalités en modules logiques, et ainsi de réduire la complexité globale. Cet agencement en modules doit également observer un couplage faible pour conserver une architecture flexible.

La présence d'une même portion de code à plusieurs endroits du programme est un obstacle à d'éventuelles modifications. Le but est alors d'éviter les duplications de code. Pour cela, la notion de composants transverses pourra résoudre ce type de problème. Il s'agit de mettre en facteur un ensemble de sous-modules à exploiter dans plusieurs modules principaux.

3.1.2 Composants transversaux

Dans le contexte de développement d'un socle logiciel, plusieurs fonctionnalités transversales seront conçues sous formes de composants logiciels tel que la journalisation, la persistance, l'authentification, ...etc.

a. Journalisation

La journalisation consiste à garder les traces, sur un support, des événements survenus dans un système ou dans une application.

Un ou plusieurs fichiers de log au format prédéfini sont générés en cours d'exécution et conservent des messages informant sur la date et l'heure de l'évènement, la nature de l'évènement et sa gravité par un code ou une description sémantique, éventuellement d'autres informations (utilisateur, classe, ...etc.).

Nous donnons par la suite un tableau comparatif de deux outils logiciel (Log4j2 et Logback) utilisés dans le processus de journalisation :



Caractéristiques	Log4j2 	Logback 
Fonctionnalités	<ul style="list-style-type: none"> - Implémentation distinct de l'API. - Async logging. - Rechargement automatique de la configuration. - Support d'autres APIs. - options de configuration (XML, properties, JSON, YAML). 	<ul style="list-style-type: none"> - Capacités de filtrage avancées. - compression des fichiers journaux archivés. - Journalisation d'accès HTTP.
Langage	Java	Java
Source	Ouvert	Ouvert
Licence	Apache	LGPLv2
S. E	Windows/MacOs/ Linux	Windows/Android
Prix	Gratuit	Gratuit

Tableau 2: Tableau comparatif sur la journalisation

- Toutes ces paramètres comparatifs nous mènent à en-déduire que Log4j2 est le plus avancé et le plus rapide dans la journalisation.



b. Persistance

Permet de sauvegarder rapidement un objet java dans une base de données. Il permet de s'affranchir des requêtes SQL ¹écrite à la main et le plus souvent difficile à maintenir dans le temps.

Avec JDBC ²Classique, nous aurions probablement écrit des requêtes SQL dans une classe proposant un ensemble de méthode pour gérer les objets et les relier à une base de données.

Par contre JPA ³ et Hibernate vont nous permettre de nous affranchir de l'écriture de ces longues requêtes SQL. Il s'agit de ce qu'on appelle des ORM ⁴. C'est un Framework qui va se charger de faire la correspondance entre notre objet et la table correspondante dans la base de données.

Dans le tableau suivant nous présentons les différences entre le JDBC et Hibernate :

Caractéristiques	JDBC 	Hibernate 
Mapping	Manuel	Fichier XML/Annotation
Connexion à base de données	Avec une requête	Connexion automatique
Langage	SQL	SQL et HQL ⁵

¹StructuredQueryLanguage

²Java Data Base Connectivity

³Java Persistence API

⁴Object Relational Mapping

⁵Hibernate QueryLanguage

Mapping entre les tables	Nécessite d'écrire beaucoup de codes	À travers le XML ou les annotations
Cache	Gérée via un code écrit explicitement	Fixé à l'espace de travail de l'application

Tableau 3 : Tableau comparatif sur la persistance

- JPA-Hibernate facilite la persistance grâce à l'utilisation des annotations.

c. Authentification

L'authentification pour un système informatique est un processus permettant au système de s'assurer de la légitimité de la demande d'accès faite par une entité afin d'autoriser l'accès de cette entité à des ressources du système conformément au paramétrage du contrôle d'accès. Parmi les Framework d'authentification nous citons SpringSecurity.

Spring Security est un Framework d'authentification puissant et hautement personnalisable et d'un cadre de contrôle d'accès. Il est le standard pour la sécurisation des applications web. C'est l'un des projets Spring les plus matures et largement utilisé. Fondée en 2003 et activement maintenu par Spring Source, il est aujourd'hui utilisé pour sécuriser de nombreux environnements les plus exigeants, y compris les agences gouvernementales, les applications militaires et les banques centrales. Il est distribué sous la licence Apache 2.0

d. Reporting

Le reporting est une technique informatique de préparation de rapports, consistant à extraire des données pour les présenter dans un rapport plus facilement lisible, voir pédagogique ou de vulgarisation pour les données complexes (affichable ou imprimable). On parle alors d'informatique décisionnelle, du fait que ces rapports constituent une aide à la décision par la visualisation de la situation présente (sous forme de statistiques, évolutions temporelles, cumuls et sous-totaux, ...etc.) qu'ils dévoilent. Parmi les Framework de reporting nous citons Jasper Report, Crystal Report.

❖ Jasper Report

C'est une librairie open source de reporting, qui permet de fournir des données sur l'écran, à l'imprimante, ou en différents formats de fichier. Le Framework jasper Report est entièrement écrit en Java. Son objectif principal est d'aider à créer les rapports et des documents prêts à être imprimés de manière simple et flexible. Il permet la représentation de données sous forme textuelle, mais aussi la génération de graphiques divers (sous forme de barre, courbe, nuage de point).

❖ Crystal Report

C'est un Logiciel de reporting et de Business Intelligence permettant de concevoir et diffuser des rapports pertinents afin simplifier et accélérer la prise de décision. Les fonctionnalités offertes par Crystal Report sont :

- Générer des rapports au rendu parfait.
- Créer rapidement des rapports parfaitement mis en forme avec une interface de conception intuitive et des workflows efficaces.
- Diffuser un contenu sous format standard, tel que PDF, feuille de calcul et HTML, même au sein de très grandes entreprises.

❖ Etude comparative

Caractéristiques	Jasper Report 	Crystal Report 
Prix de la licence	Gratuit	Gratuit
.NET	Oui (difficile de configurer)	Oui
Java	Oui	Oui (Configuration difficile)
PDF	Oui	Oui
Tables imbriquées	Oui	Oui

Tableau 4 : tableau comparatif sur le Reporting

- Jasper Report est la meilleure librairie fonctionnelle avec un projet java.

3.2 Outils d'automatisation de la construction Java

3.2.1 Ant

Apache Ant ("AnotherNeatTool") est une bibliothèque Java utilisée pour l'automatisation des processus de construction des applications Java. De plus, Ant peut être utilisé pour créer des applications non Java. Il faisait initialement partie de la base de code Apache Tomcat et a été publié en tant que projet autonome en 2000.

Avantage :

- Intégré dans la quasi-totalité des IDE.
- Supporté nativement par les outils d'Intégration Continue.
- Connu pratiquement par tout le monde, il est aussi enseigné dans les écoles.
- On peut tout faire avec Ant.
- Plutôt facile à prendre en main, relativement intuitif.
- Grande flexibilité.

Inconvénient :

- Pas de conventions dans l'écriture du XML, ce qui amène de la complexité dans le script.
- Aucun mécanisme de gestion des dépendances.
- Création complexe et répétitive d'un processus complet de création d'un artefact JAR ou WAR.

3.2.2 Maven

Apache Maven est un outil de gestion des dépendances et d'automatisation de la construction, principalement utilisé pour les applications Java. Maven continue à utiliser des fichiers XML, tout comme Ant, mais de manière beaucoup plus simple. Le nom du jeu ici est convention sur la configuration.

Alors qu'Ant donne la flexibilité et nécessite que tout soit écrit à partir de rien, Maven s'appuie sur des conventions et fournit des commandes prédéfinies (objectifs).

Avantage :

- Gestion des versions des librairies.
- Configuration et interception de toutes les phases de construction de votre projet à l'aide de plugins.
- Gestion de dépendances.
- Gestion de la distribution.
- Disponibilité des plugins sur les plateformes : eclipse, netbeans, intellij idea.

Inconvénient :

- Trop grande utilisation de plugins, y compris pour réaliser certaines tâches simples.
- Manque de souplesse, de rigueur sur certains principes (difficile de sortir du cycle de vie par exemple).
- Utilisation de repositories pas forcément clean : il est possible de définir un ou plusieurs repositories externes pour gérer les dépendances. Cependant, si la gestion d'une des librairies est mal faite, cela conduit à des erreurs ou à des différences dans les jars du même projet.

3.2.3 Etude comparative



Caractéristiques	Ant 	Maven 
Mieux pour contrôler le processus de construction	Oui	Non
Gestion de la dépendance	Non	Oui
Inconsistant build	Non	Oui
Facile à écrire avec ses propres crochets java-build	Oui	Non
Complexe	Non	Oui

Tableau 5 : tableau comparatif entre les outils d'automatisation

Maven Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java. Plus efficacement que Ant.

3.3 Framework back-end

Les Framework back-end sont évalués par les outils de programmation, les langages et les interfaces qu'ils offrent. En termes simples, un Framework back-end avancé accélère la vitesse de développement, réduisant ainsi la perte de temps des tâches pour les développeurs. Il existe plusieurs Framework back-end nous citons Spring boot, Ejb.

3.3.1 Spring boot

Spring Boot est un module récent de l'écosystème Spring (*Spring est l'infrastructure de développement d'applications la plus répandue pour les entreprises Java. Des millions de développeurs du monde entier utilisent Spring Framework pour créer du code hautement performant, facilement testable et réutilisable*). Il permet de simplifier le développement en Spring (configuration Maven, programme principal, etc.). Les éléments nécessaires pour une application autonome en Spring Boot sont :

- Un fichier pom.xml récupérant les éléments SpringBoot (pom parent).
- Un fichier Main.java qui initialise le conteneur Spring.
- Des annotations dans les classes.

SpringBoot une plate-forme qui se base sur la notion de conteneur léger par opposition au conteneur EJB considéré lourd technologiquement et surtout peu adaptatif par rapport aux différents types de problématiques courantes. Le conteneur léger, fournit un support simple et

puissant pour gérer une application via un ensemble de composants, c'est-à-dire des objets présentant une interface dont le contenu interne n'est pas connu par les autres composants.

3.3.2 EJB

Les EJB (Entreprise Java Bean) sont un des éléments très importants de la plate-forme Java EE pour le développement d'applications distribuées. La plate-forme Java EE propose de mettre en œuvre les couches métiers et persistance avec les EJB. Particulièrement intéressants dans des environnements fortement distribués, jusqu'à la version 3, leur mise en œuvre est assez lourde sans l'utilisation d'outils tels que certains IDE.

3.3.3 Etude comparative



Concept/service	Ejb 3.0 	
Injection de dépendance	Les EJB, les sources de données, les ressources JMS et JPA	Les listes, les maps, les propriétés et les ressources JNDI
Gestion des transactions	Fonctionne immédiatement, mais seul JTA ⁶ est pris en charge	Il faut le configurer pour le faire fonctionner, avec JTA, JDBC et Hibernate.
Persistance	Bien intégré via JPA	JPA, Hibernate, JDBC...
Gestion d'état	Prise en charge robuste via les State Beans de session et le contexte de persistance étendue	Prise en charge indirecte dépendante de la gestion de session de conteneur Web
AOP ⁷	Prise en charge simple mais limitée par des intercepteurs.	Support solide via AspectJ et Spring AOP.
Sécurité	Prise en charge intégrée de la sécurité déclarative et programmatique via JAAS.	Prise en charge intégrée via SpringSecurity

Tableau 6: Tableau comparatif sur les Framework back-end

- Spring boot assure la prise en charge indirecte dépendante de la gestion de session de conteneur Web et notre projet sera construit sur cette vertu.

⁶Java Transaction Api

⁷Aspect Oriented Programming

3.4 Framework front-end

Aujourd'hui, le nombre des Framework de développement et des bibliothèques basés sur java script est en croissance continue. En effet, les technologies client les plus reconnues sont Angular le Framework de Google et son concurrent fondé par Facebook qui s'appelle ReactJs.

a. Angular



Angular est un Framework côté client open source basé sur TypeScript dirigée par l'équipe du projet Angular à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète de AngularJS.

b. ReactJs

ReactJs est une bibliothèque JavaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état.

c. Etude comparative

Une comparaison entre les technologies Angular et ReactJs est résumé par le tableau ci-dessous selon divers critères à savoir la facilité d'apprentissage, la consommation de mémoire, le degré d'abstraction et autres.

Caractéristiques	ReactJs 	Angular 
Langage	Se repose sur plusieurs librairies, principalement le JSX ⁸ syntaxe pré de html compilé en java script	Se repose sur le type script.
Complexité	Elevé	Réduite
Design du code	JavaScript	JavaScript en HTML
Complétion du code	Faible	Elevée
Taille en production	144 KO	117 KO
Taille en développement	1 Mo	559 KO
Courbe d'apprentissage	Fiable	Moyenne
Abstraction	Fort	Fort
Débogage	JavaScript fort/ HTML faible	JavaScript fort/ HTML faible
Liaison des données	Unidirectionnel	Bidirectionnel
Réutilisation des composants	Fort	Passable
Support du design pattern	Oui	ViewElement

⁸JavaScript XML

MVC		
-----	--	--

Tableau 7:Tableau comparatif entre les Framework front-end

- Angular 7 est plus intéressant grâce à son taille réduit de développement et de production, aussi la bidirectionnalité de liaisons des données représente notre orientation du projet.

3.5 La programmation orientée aspects

La programmation orientée aspect est un paradigme de programmation qui permet de traiter séparément :

- Les préoccupations transversales, qui relèvent souvent des aspects techniques (Journalisation, Sécurité, Transaction, ...)
- Les préoccupations métiers, qui constituent le cœur d'une application permet de séparer le code métier du code technique.

La programmation orientée aspect est bien une technique transversale et n'est pas liée à un langage de programmation particulier et peut être mise en œuvre aussi bien avec un langage orienté objet comme Java qu'avec un langage procédural comme le C.

Parmi les moyens de séparation nous citons :

a. Filtre

Un filtre, comme son nom l'indique, est une classe Java exécutée par le conteneur de servlets pour chaque demande http entrante et pour chaque réponse http. De cette façon, il est possible de gérer les demandes entrantes HTTP avant qu'elles n'atteignent la ressource, telles qu'une page JSP, un contrôleur ou une simple page statique ; De la même manière, il est possible de gérer la réponse sortante HTTP après l'exécution de la ressource.

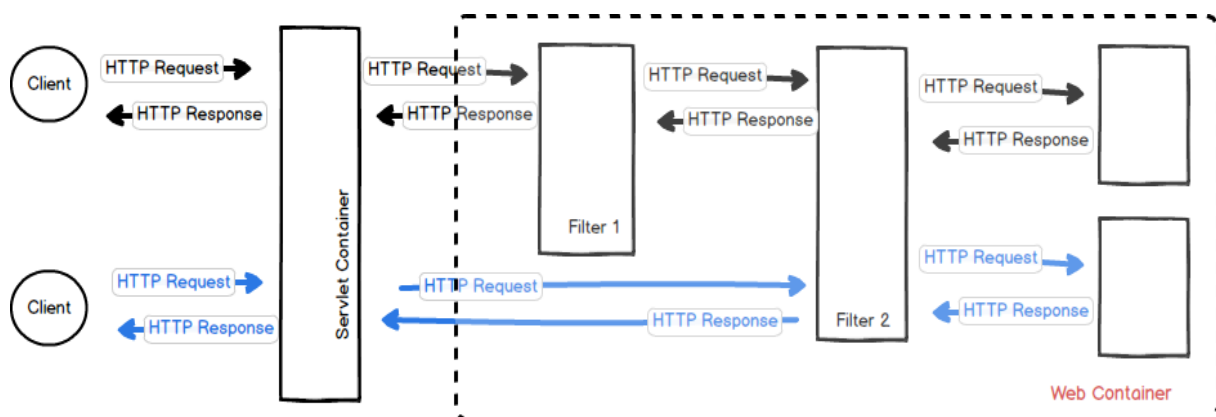


Figure 6: cycle d'exécution d'un Filter

- Filter s'exécute une seule fois lors de l'exécution du projet et l'appel des méthodes non configurable.

b. Intercepteur

Les intercepteurs peuvent être définis dans une classe cible en tant que méthode d'intercepteur ou dans une classe associée appelée classe d'intercepteur. Les classes d'intercepteur contiennent des méthodes appelées conjointement avec les méthodes ou les événements de cycle de vie de la classe cible.

Les intercepteurs Spring sont similaires aux filtres, mais ils agissent dans le contexte Spring. Ils sont donc très puissants pour gérer les requêtes et les réponses HTTP, mais ils peuvent implémenter un comportement plus sophistiqué car ils peuvent accéder à tous les contextes Spring.

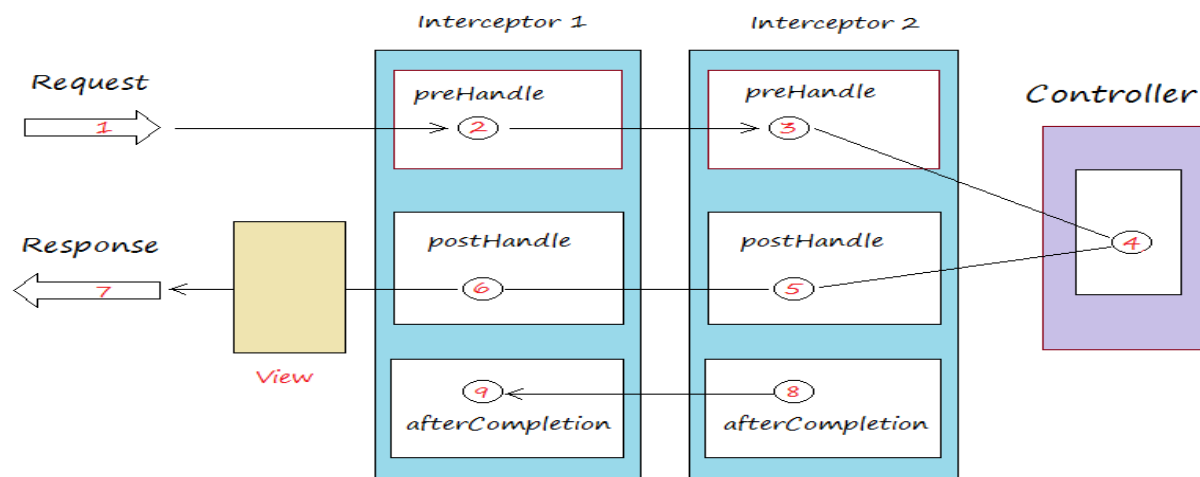


Figure 7: cycle d'exécution d'un Intercepteur

Les intercepteurs affectent seulement 3 méthodes PreHandle, PostHandle, AfterCompletion.

c. Aspects

Un aspect est la modularisation d'une préoccupation qui concerne plusieurs classes. La journalisation unifiée peut être un exemple d'une telle préoccupation transversale. Il s'agit essentiellement d'un moyen d'ajouter un comportement au code existant sans modifier ce code.

Chaque aspect se concentre sur une fonctionnalité transversale spécifique

- **Joinpoint** : C'est un point particulier lors de l'exécution de programmes tels que l'exécution de méthodes, les appels de constructeurs ou les affectations de champs.
- **Advice** : l'action prise par l'aspect dans un point de jonction spécifique.

- **Pointcut** : une expression régulière qui correspond à un point de jonction. Chaque fois qu'un point de jointure correspond à un point coupé, un avis spécifié associé à ce pointtout est exécuté.
- **Tissage** : le processus de liaison des aspects avec des objets ciblés pour créer un objet conseillé.

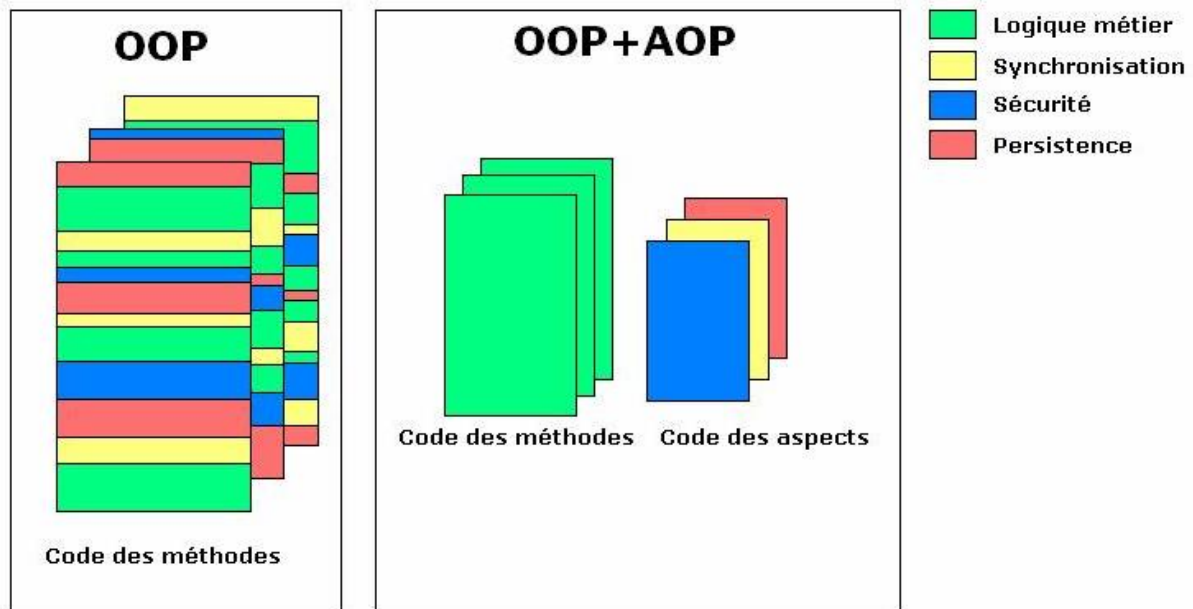


Figure 8: architecture des aspects

- Les aspects sont plus riches que les intercepteurs car il existe plusieurs advices qui nous aide dans notre projet.

Parmi les aspects nous citons AspectJ et Spring AOP.

❖ AspectJ

AspectJ propose un support très complet des possibilités offertes par l'AOP.

AspectJ utilise sa propre syntaxe pour la création d'un aspect mais surtout les aspects peuvent être tissés au runtime avec un agent dédié (classloader qui enrichit le bytecode lors de son chargement) ou à la compilation avec le compilateur dédié d'AspectJ.

AspectJ 5 permet la création d'un aspect sous la forme d'une simple classe annotée avec des annotations dédiées comme **@Aspect**.

❖ Spring AOP

Spring AOP est un module du framework Spring qui permet une mise en œuvre d'une partie des fonctionnalités de l'AOP. Il propose un tisseur d'aspects sous la forme de proxys qui sont créés dynamiquement au runtime.

❖ Spring AOP et AspectJ

Il est possible dans une même application d'utiliser des aspects tissés par Spring AOP et par AspectJ.

Lorsque les aspects sont définis en utilisant les annotations d'AspectJ, il est nécessaire de préciser les aspects qui seront tissés par Spring AOP et ceux qui le seront par AspectJ.

Les aspects pris en charge par Spring AOP doivent être précisés en utilisant le tag `<aop:include>` dans le fichier de configuration du contexte de Spring.

Les aspects pris en charge par AspectJ doivent être précisés dans le fichier `aop.xml` pour un tissage dynamique.

Ainsi chaque tisseur prendra en charge les aspects qui le concernent.

Le tableau suivant résume les différences entre les deux outils :


Caractéristiques	Spring AOP 	ASPECTJ
Langage	Implémenté en Java pur	Implémenté à l'aide d'extensions du langage de programmation Java
Compilation	Pas besoin de processus de compilation séparé	Nécessite compilateur AspectJ sauf si LTW ⁹ est configuré
Implémentation	Ne peut être implémenté que sur les beans gérés par le conteneur Spring	Peut-être implémenté sur tous les objets de domaine
Exécution des Pointcut	Prend en charge uniquement les pointcuts d'exécution de la méthode	Soutenir tous les pointcuts
Performance	Beaucoup plus lent qu'AspectJ	Meilleure performance

Tableau 8: Tableau comparatif entre les outils de La programmation orientée aspects

AspectJ est plus intéressant que Spring AOP dans notre projet parce que nous recherchons la performance, l'implémentation doit être sur tous les objets domaine et surtout AspectJ permet de soutenir tous les pointcuts et c'est exactement ce qui nous intéresse.

3.6 La gestion de version

Un logiciel de gestion de versions s'articule autour d'un concept très simple : la sauvegarde de l'entièreté des modifications faites sur tous les fichiers du projet et le maintien du code source tout au long du processus de développement. En d'autres termes, il s'agit d'un type d'archivage moderne muni de nombreuses fonctionnalités. Parmi les outils de gestion de versions nous citons Git, Svn.

⁹Load-time Weaving

3.6.1 *Git*

C'est un logiciel de gestion de versions décentralisé. Il est open source, créé aux alentours de 2005 par Linus Torvalds, développeur du noyau Linux.

Git ne repose pas sur un serveur centralisé, mais il utilise un système de connexion pair à pair. Le code informatique développé est stocké non seulement sur l'ordinateur de chaque contributeur du projet, mais il peut également l'être sur un serveur dédié. C'est un outil de bas niveau qui se veut simple et performant, dont la principale tâche est de gérer l'évolution du contenu d'une arborescence.

Git est considéré comme performant, au point que certains autres logiciels de gestion de version, qui n'utilisent pas de base de données, se sont montrés intéressés par le système de stockage des fichiers de Git pour leur propre fonctionnement. Ils continuent toutefois à proposer des fonctionnalités plus évoluées.

La décentralisation de Git a aussi beaucoup apporté au développement des logiciels libres, puisque le besoin de demander un compte sur un dépôt SVN ou CVS centralisé devient obsolète.

Avantage :

- Très rapide.
- Sait travailler par branches (versions parallèles même projet) de façon très flexible.
- Assez complexe, il faut un certain temps d'adaptation pour bien le comprendre et le manipuler.

3.6.2 *SVN*

Un autre grand parmi les grands, c'est Apache Subversion, ou SVN. Subversion est un logiciel libre développé par l'Apache Software Foundation depuis 2000. Remplaçant de l'illustre logiciel Concurrent Versions System ou CVS, Subversion propose en substance les mêmes fonctionnalités que Git pour le travail en équipe.

Avantage :

- Subversion permet de tracer les versions de répertoires, de fichiers et de droits sur les fichiers.
- Subversion permet de renommer un fichier ou un répertoire tout en conservant son historique.

- Les propagations de version (commit) sont atomiques. Une propagation réussit uniquement si tous les fichiers de la version sont correctement propagés.
- Les numéros de versions concernent une propagation et non les fichiers eux-mêmes.

3.6.3 Etude comparative

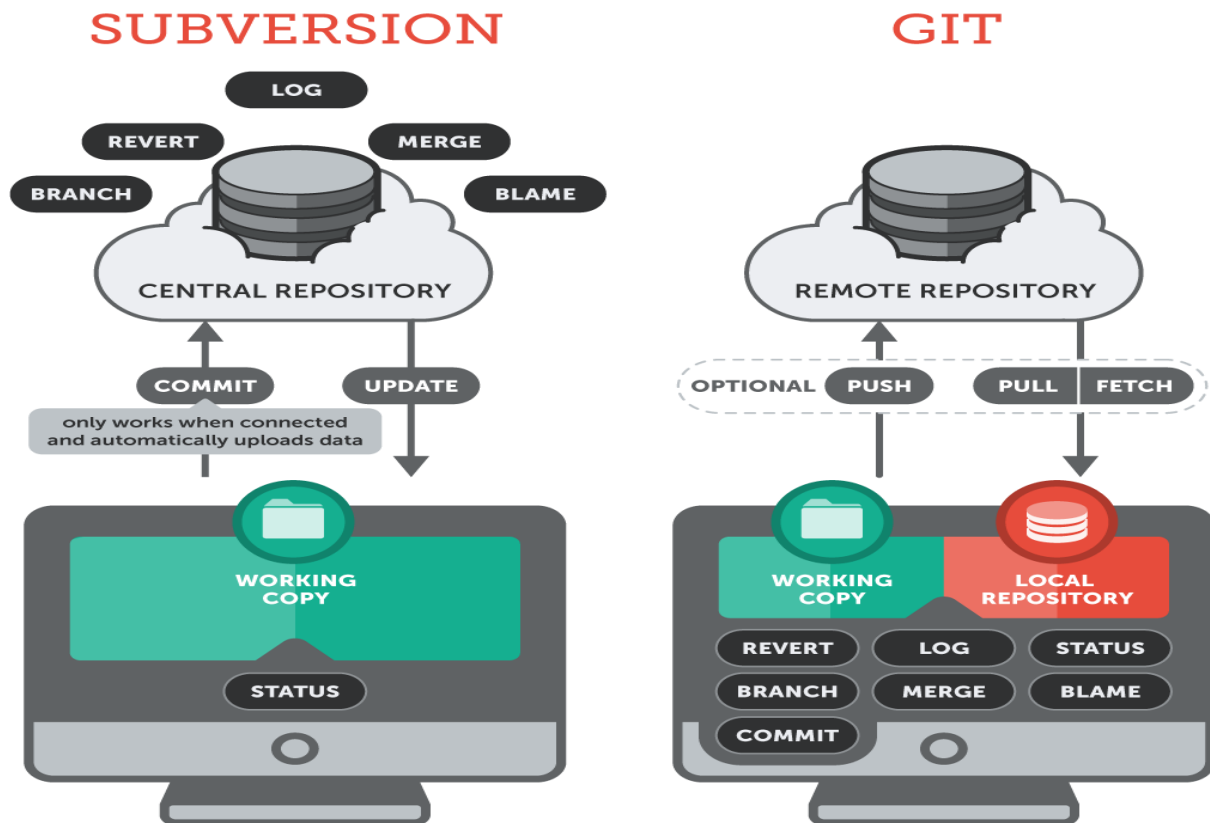


Figure 9: Architecture de Git et SVN

Caractéristiques	SVN	GIT
Gestion des versions	Central	Distribué
Dépôt	Un référentiel central dans lequel les copies de travail sont générées	Copies de référentiel disponibles localement dans lesquelles des fichiers et des documents peuvent être traités
Autorisation d'accès	Basé sur le chemin	Pour tout l'annuaire
Suivi des modifications	Registres de fichiers	Enregistre le contenu
Changer l'historique	Compléter uniquement dans le référentiel. Les copies de travail ne contiennent que la dernière version	Le référentiel et les copies de travail contiennent l'historique complet
Connexion réseau	Nécessaire pour l'accès	Seulement nécessaire pour la synchronisation

Tableau 9: Tableau comparatif entre les outils de gestion de version

- Suite à cette comparaison, nous avons opté pour le serveur Git, qui grâce à son architecture décentralisée et le gain en espace qu'il fournit (dépôt 30 fois inférieur à celui de SVN).

4. L'architecture applicative future

Acteur	Equipe DevOps			
	Equipe de développement 1		Equipe de Développement 2	
Socle	Journalisation/Reporting/Sécurité/AOP			
Services	Gestion d'impression	Gestion d'intervention	Intégration continue	Gestion du parc automobile
Solution métier	ERP			

Tableau 10: Architecture logique cible

Après une étude exhaustive des différents outils et Framework de développement, nous avons opté pour une solution logicielle modulaire basée sur Spring Boot comme Framework back-end et Angular 7 comme Framework front-end, l'outil Maven pour l'automatisation de production logiciel, AspectJ pour la gestion des composants transversaux, Log4j2 pour la gestion de journalisation, SpringSecurity pour la gestion d'authentification et les droits d'accès, Jasper Report pour la génération des rapports et en fin, Git pour la gestion des versions.

Dans la figure ci-dessous nous allons schématiser l'interaction des différents outils et frameworks de développement.

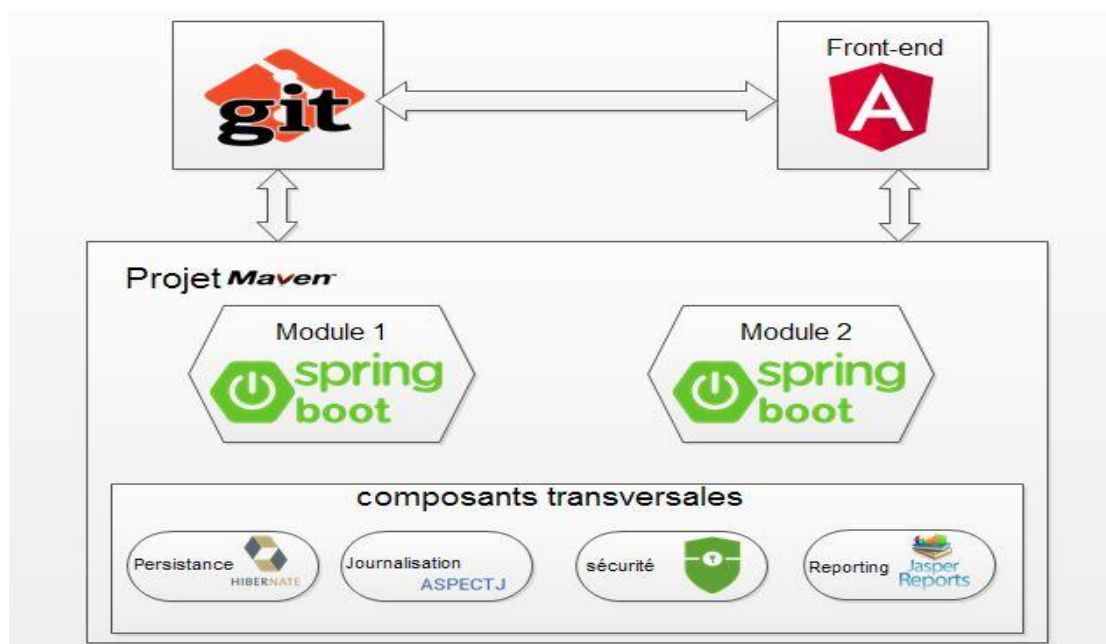


Figure 10: schéma d'interaction des différents outils et frameworks de développement

Conclusion

Ce chapitre présente une phase indispensable pour notre application. Nous avons élaboré une étude comparative entre les différents outils et environnements logiciel sur les quels notre application est basée afin de prouver notre choix des différents outils.

Dans le chapitre suivant, nous allons présenter une étude des exigences techniques et fonctionnelles ainsi que la planification du projet.

Chapitre 3 : Analyses et spécifications

Introduction

Le travail qui nous a été confié pendant notre stage consiste à développer une application web gérant le suivi de la consommation papiers dans l'ISSET de Sfax.

Pour ce faire, il est indisponible de réaliser une analyse générale. En premier temps, nous allons dégager tous les besoins fonctionnels et non fonctionnels. Ensuite, nous entamons la phase de spécification du projet en suivant une méthodologie de travail précise.

1. Spécification des besoins

1.1 *Besoins fonctionnels*

Ils décrivent les fonctionnalités du système à développer. Ces sont les besoins qui spécifient le comportement d'entrées/sorties du système à concevoir, qui doit permettre à l'utilisateur d'effectuer les opérations suivantes :

- Prise en charge des différentes données de base (enseignants, groupes d'étudiants, matières).
- Gestion d'approvisionnement en papier (appels d'offres, commandes, réception des commandes).
- Prise en charge des opérations d'entretien des machines de tirages.
- Prise en charges des opérations de demande de tirage par un enseignant pour une matière et un groupe toute en faisant le téléchargement du fichier à tirer.
- Consultation de la liste des demandes ordonnées par date et possibilité de validation en cas d'exécution de l'opération de tirage correspondante.

1.2 *Besoins non fonctionnels*

Les besoins non fonctionnels sont des exigences qui ne concernent pas spécifiquement le comportement du système mais identifient plutôt les contraintes internes et externes du système. Parmi ces besoins nous citons :

- **Performance** : l'application doit être fiable, rapide et pratique.

- **Ergonomie et souplesse** : L'application doit offrir une interface qui respecte les critères d'ergonomie. Cette interface doit également être exploitable et interactive avec les actions de l'utilisateur.
- **Extensibilité** : L'application doit être extensible, c'est-à-dire qu'il pourra y avoir une possibilité d'ajouter ou de supprimer des nouvelles fonctionnalités.
- **Sécurité** : L'application doit être sécurisée, les informations ne devront pas être accessibles sans authentification.

2. BACKLOG général du produit

Le BACKLOG du produit est la liste des fonctionnalités attendues à réaliser. Plus exactement, il contient tous les éléments nécessaires pour le travail en équipe. Ces éléments sont classés selon leurs priorités qui permettent de définir l'ordre de réalisation des tâches. Dans le tableau suivant, nous définissons la description, l'acteur et la priorité adéquate à chaque sprint.

Sprint	Description	Acteur	Priorité	Date début	Date fin
Sprint 0 : Architecture logiciel et conception	-Choix de l'architecture du système. -La conception générale de notre système.	Fahmi et Lotfi	1	01/02/2019	
Sprint 1 : Authentification et gestion des rôles	-Gérer l'authentification. - Gérer l'affectation des droits. - Gérer les rôles. - Gérer les utilisateurs.	Fahmi et Lotfi	2		
Sprint 2 : gestion demande de tirage	- téléverser les fichiers sur la plateforme. - Gérer la file d'attente d'impression.	Fahmi et Lotfi	3		
Sprint 3 : gestion d'entretien des photocopieurs et imprimantes	- Gérer les entretiens. - Gérer la recharge des cartouches d'encre.	Fahmi et Lotfi	4		
Sprint 4 : gestion d'approvisionnement	- Gérer le stock papiers - Gérer les demandes d'approvisionnement	Fahmi et Lotfi	5		

ement en papier	papiers.				
--------------------	----------	--	--	--	--

Tableau 11:BACKLOG général du produit

3. Chronogramme de travail

Dans cette section on se propose d'établir le diagramme de Gantt qui décrit la répartition des tâches du projet, qui a duré du 01/02/2019 au 30/05/2019, afin de donner une vue globale de la répartition du temps par rapport au travail demandé.

Ce diagramme est illustré par la figure suivante :

[[Diagramme de Gantt à ajouter]]

Conclusion

Ce chapitre présente une phase indispensable pour l'étude et l'analyse de notre application. Nous avons défini les principaux besoins fonctionnels et non fonctionnels du projet. Nous avons aussi prouvé le choix de la méthodologie « SCRUM » afin de livrer un produit livrable dans le délai spécifié. Puis, nous avons finis par présenter le « BACKLOG » du produit en spécifiant les différentes fonctionnalités qui le composent. En fin, nous avons présenté le chronogramme de travail à travers le diagramme de GANTT.

Dans ce qui suit nous allons présenter l'architecture logicielle de notre application et une conception générale.

Chapitre 4 : Etude conceptuelle (Sprint 0)

Introduction

La phase de conception est une phase très importante dans le cycle de développement d'une application. Elle a pour objectif de modéliser et concevoir l'architecture générale et détaillée de notre futur système.

1. Conception générale

Le choix de l'architecture globale est primordial à la conception d'une application. Il permet d'assurer de meilleures performances et un bon fonctionnement.

Dans cette partie, nous allons présenter l'architecture logique et l'architecture physique de notre application.

1.1 *Architecture logique*

Dans le but de répondre aux besoins de notre application, nous avons choisi une architecture applicative qui vise à séparer quatre couches logicielles au sein d'une même application. La séparation en couche, permet de réaliser une certaine indépendance entre elles afin de faciliter une maintenance future. Chaque couche a un rôle bien précis dans le fonctionnement de notre application.

- **Couche présentation** : c'est la partie web qui est développée en Angular7. Elle contient les interfaces Homme/Machine et les services pour assurer la communication avec la partie back-end.
- **Couche services** : c'est la couche qui contient les contrôleurs REST¹⁰. Ces contrôleurs ont pour but d'assurer l'interconnexion entre la couche présentation et la couche métier. Cette couche permet de sérialiser les objets en format JSON¹¹ pour qu'ils soient compréhensibles par la couche présentation.
- **Couche métier** : cette couche contient les traitements métiers de l'application à savoir les services des différents modules fonctionnels. Ces services sont implémentés sous la forme des services Spring.

¹⁰ REpresentational State Transfer

¹¹ Java Script Object Notation

- **Couche domaine** : c'est la couche qui contient toutes les entités .Elle est responsable à la manipulation des données, elle permet de sauvegarder et faire appel aux données qui existent dans une base de données MySql grâce à Spring Data.

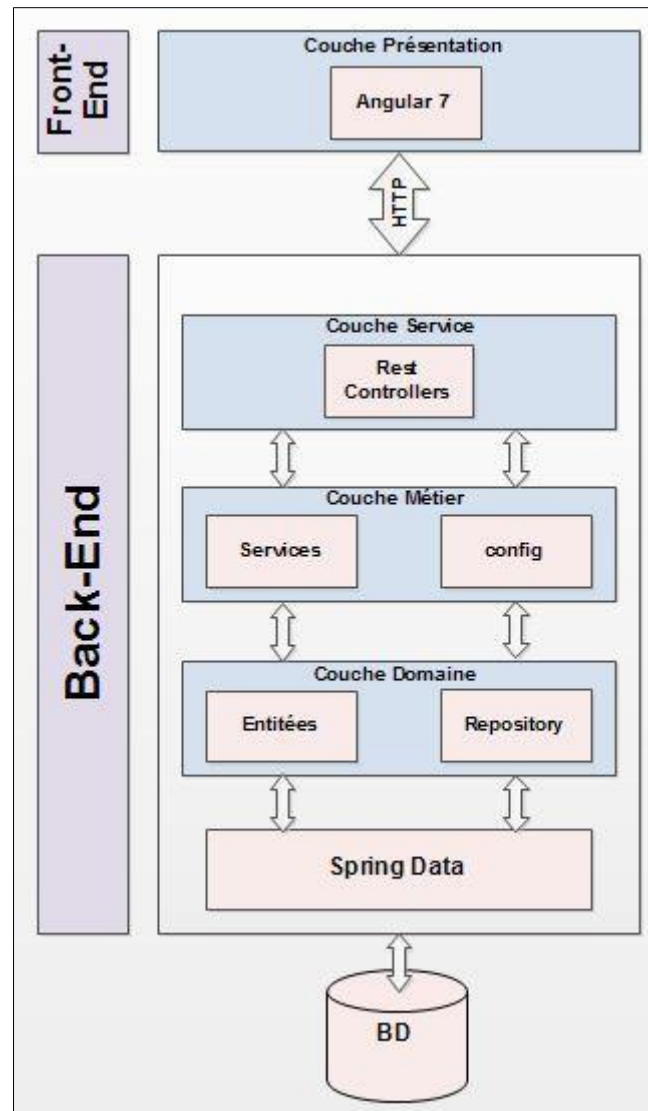


Figure 11:Architecture logique

❖ Diagramme de paquetage

Un paquetage étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML, le diagramme de paquetage sert à représenter les dépendances entre ensembles de définitions. Cette représentation nous permettra d'améliorer les facteurs de couplage et cohésion de notre système c'est-à-dire la forte dépendance intra package et le faible échange inter package.

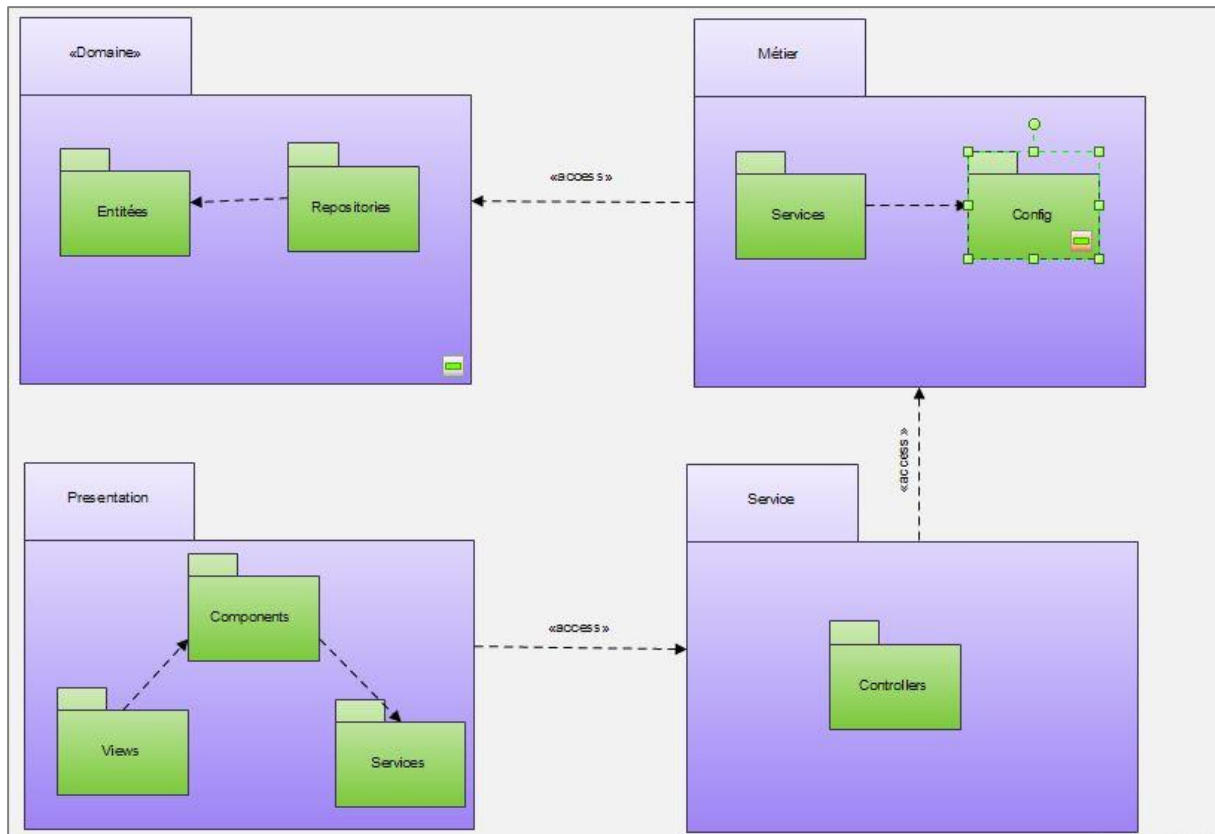


Figure 12:Diagramme de paquetage

Nous présentons ici les différents paquetages existants dans la solution :

- **Views** : contient les interfaces développées dans notre application.
- **Components** : représente le cœur de la couche présentation, utilisés pour manipuler les logiques métiers en utilisant les services.
- **Services** : ce sont les services de la partie web qui permettent l'appel des web services REST en utilisant le protocole HTTP.
- **Contrôleurs** : ce paquetage contient les contrôleurs Restful. Il assure l'interconnexion entre la présentation et le métier grâce aux web services REST.
- **Services** : ce sont les services de la partie métier. Ce paquetage regroupe les classes responsables des traitements métiers de l'application.
- **Entités** : Contient les entités ayant comme rôle de faire le lien entre les objets java et la base de données. Chaque classe de ce paquetage contient un ensemble d'attributs similaires aux champs existants au niveau d'une table de la base.
- **Repositories** : Contient les interfaces qui décrivent les méthodes de manipulation des entités.
- **Config** : contient les classes de configuration.

1.2 Architecture physique

Dans cette partie, nous allons présenter l'architecture physique de notre application. Cette architecture se compose principalement d'un client web, un serveur d'application où notre application sera déployée et un serveur de base de données. Notre système suit donc l'architecture 3-tiers.

Cette architecture est constituée de :

- **Client** : C'est la partie web développée en Angular7.
- **Serveur d'application** : Il regroupe toutes les couches de la partie Back-end à savoir la couche service web, la couche service métier et la couche accès aux données.
- **Serveur de base de données (MySQL)** : C'est un serveur de base de données utilisé pour la partie persistance des données de notre application.

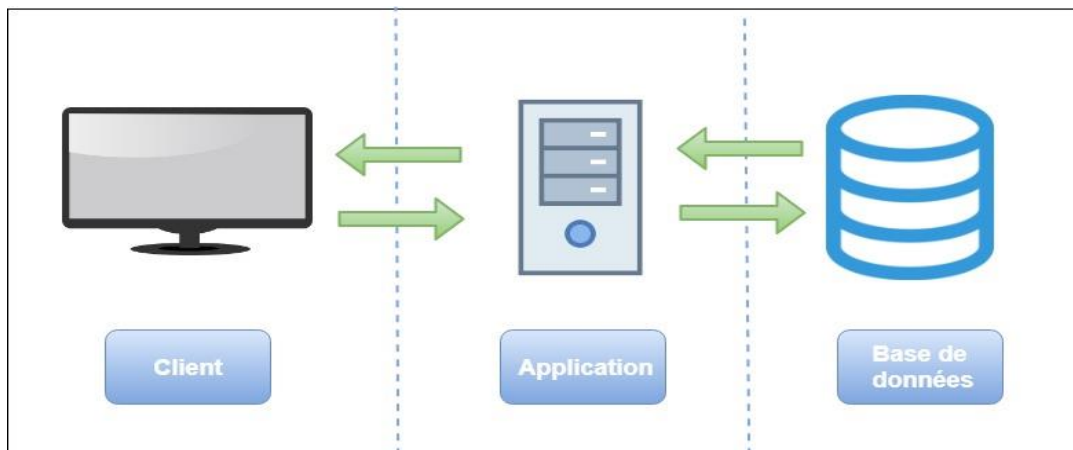


Figure 13:Architecture physique

❖ Diagramme de déploiement

Le diagramme de déploiement nous permet d'avoir une vue statique qui représente l'utilisation de l'architecture physique, et de la manière à travers laquelle les composants du système sont répartis.

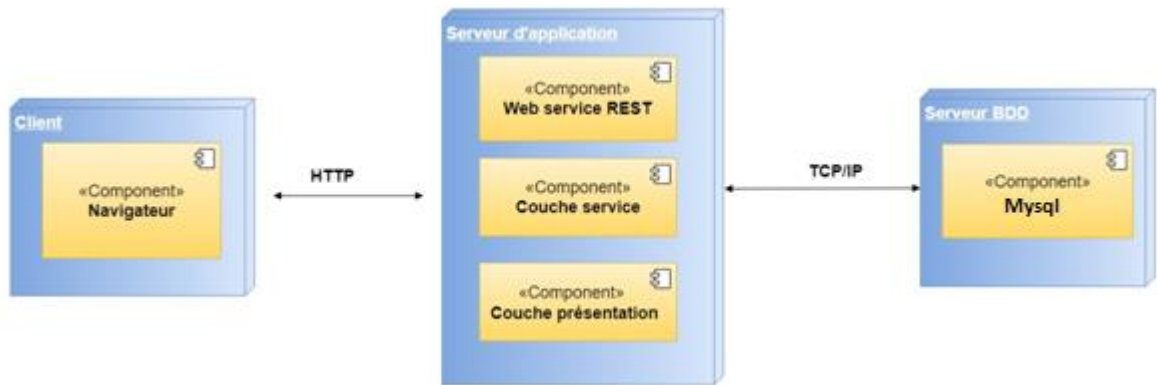


Figure 14:Diagramme de déploiement

Comme illustré dans la figure, notre solution est composée de 3 grandes parties :

- La partie client qui est composée d'un navigateur.
- La partie serveur qui est développée en Spring Boot et Angular7.
- La partie base de données MySql.

2. Conception détaillé

La phase de conception détaillée joue un rôle très important pour faciliter le développement d'une application. Elle nous aide à transformer le modèle d'analyse en un modèle concret à partir duquel nous pouvons directement implémenter le système. Nous allons commencer par présenter le diagramme de cas d'utilisation. Ensuite, nous allons présenter le diagramme de classe.

2.1 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation décrit le comportement de notre système du point de vue d'un utilisateur, sous forme des actions et des réactions. Ils permettent ainsi de définir les limites du système et les relations entre le système et l'environnement.

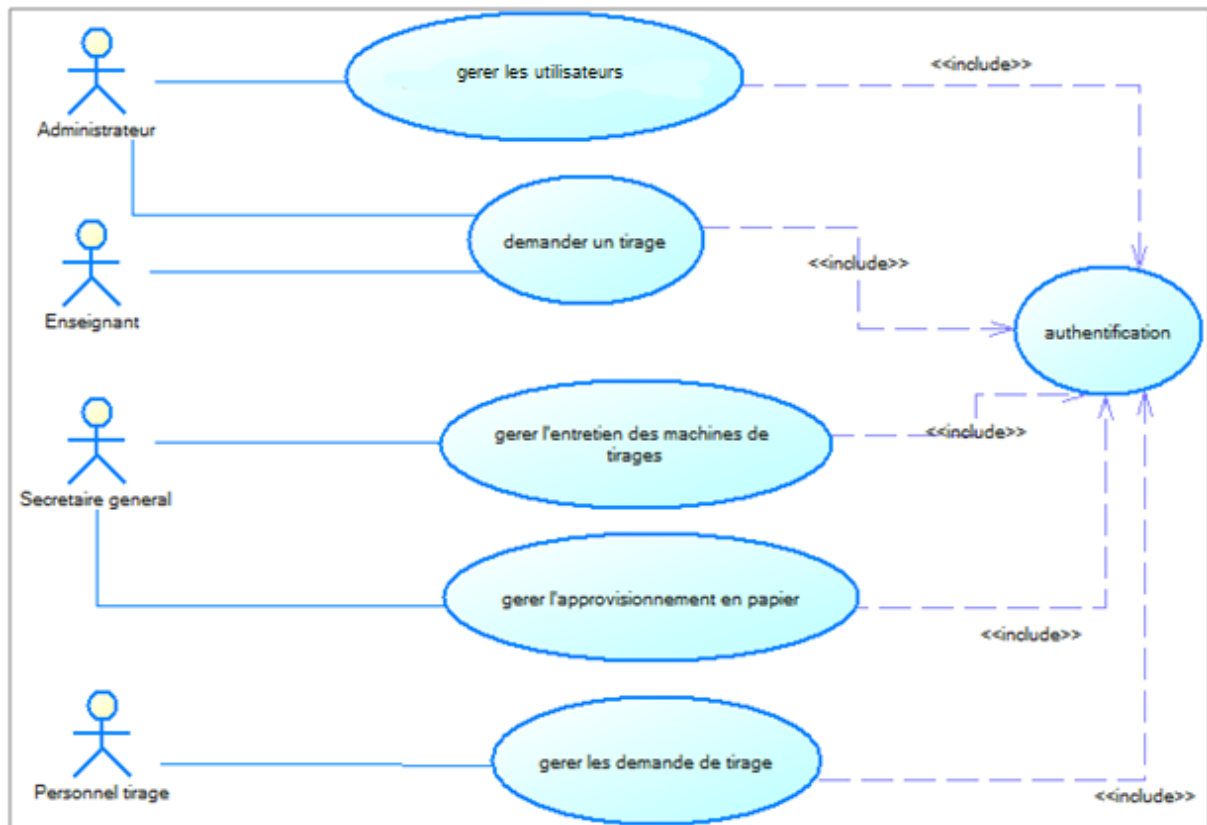


Figure 15:Diagramme de cas d'utilisation générale

2.2 Diagramme de classes

Un élément central de conception d'un logiciel est le diagramme de classes, c'est un modèle représentant la structure du système à concevoir. Le diagramme de classes est un schéma utilisé pour présenter les classes et les interfaces d'un système ainsi que les différentes relations.

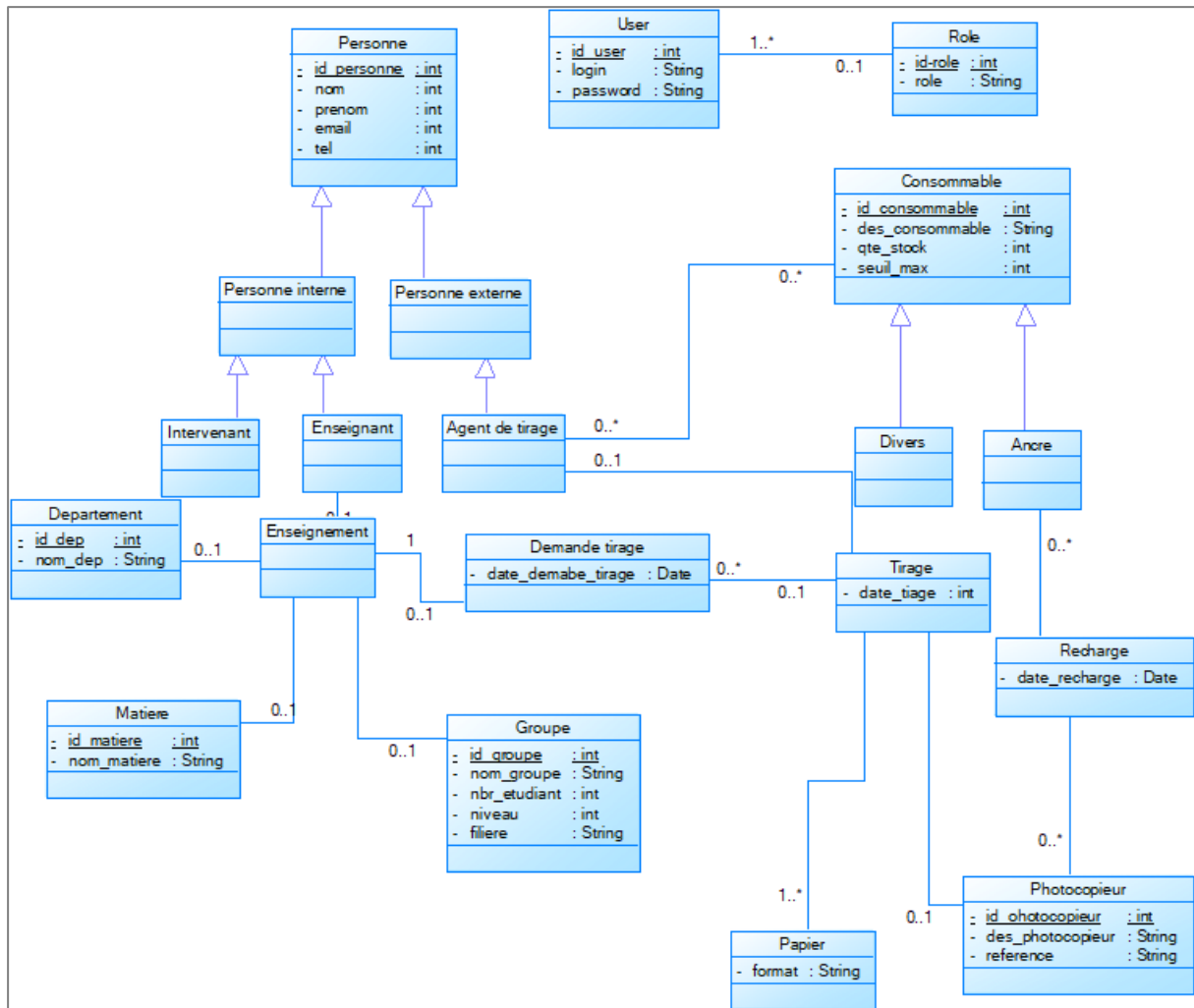


Figure 16:Diagramme des classes

❖ Dictionnaire de données

Le tableau ci-dessous représente la liste des attributs composants toutes les classes formants notre système ainsi que leur description et leur type

Nom	Type de données	Description	Classificateur
id_user	int	Identifiant d'utilisateur	User
login	String	Login d'utilisateur	User
password	String	Mot de passe d'utilisateur	User
id-role	int	Identifiant du rôle	Role
role	String	Rôle spécifique à chaque utilisateur	Role
id_photocopieur	int	Identifiant du photocopieur	Photocopieur
des_photocopieur	String	Désignation du photocopieur	Photocopieur
reference	String	La référence de photocopieur	Photocopieur
id_groupe	int	Identifiant du groupe	Groupe
nom_groupe	String	Nom du groupe	Groupe
nbr_etudiant	int	Nombre des étudiants	Groupe
niveau	int	Niveau	Groupe

Filiere	String	Filière	Groupe
id_consommable	int	Identifiant des consommables	Consommable
des_consommable	String	Désignation de consommable	Consommable
qte_stock	int	Quantité en stock de consommable	Consommable
seuil_max	int	Seuil maximal	Consommable
date_tirage	int	Date de tirage	Tirage
id_matiere	int	Identifiant du matière	Matiere
nom_matiere	String	Nom de matière	Matiere
date_recharge	Date	Date de recharge d'ancre	Recharge
format	String	Format de papier	Papier
date_demande-tirage	String	Date de demande de tirage par l'enseignant	Demande de Tirage
id_personne	int	Identifiant du personne	Personne
nom	int	Nom du personne	Personne
prenom	int	Prénom du personne	Personne
email	int	Email du personne	Personne
tel	int	Téléphone du personne	Personne
id_dep	int	Identifiant du département	Departement
nom_dep	String	Nom du département	Departement

Tableau 12:Dictionnaire de données

Conclusion

Dans ce chapitre, nous avons présente une conception architecturale de notre application. Ensuite, nous avons fourni une conception détaillée à l'aide d'un diagramme de cas d'utilisation et d'un diagramme de classe. Cette phase permet de faciliter l'étape de réalisation de notre plateforme que nous allons la présenter dans les chapitres suivants.

Chapitre 5 : Mise en place d'un socle logicielle (Sprint 1)

Introduction

Dans notre application l'authentification et les gestions des rôles et utilisateurs avec l'affectation des droits d'accès consistent à assurer la sécurité. Le but de ce sprint est de gérer l'authentification, utilisateurs, rôles et les droits d'accès pour qu'ils puissent accéder à l'application.

1. Planification prévisionnelle

La planification des tâches est une phase très importante pour la définition du nombre des tâches nécessaires pour réaliser un sprint et pour la spécification des durées de chaque sprint.

Comme l'indique le tableau suivant :

Tâches	Acteurs	Priorité	Période/Jour
Authentification	Fahmi et Lotfi	Elevée	3jrs
Gestion des utilisateurs	Fahmi et Lotfi	Elevée	3 jrs
Gestion des roles	Fahmi et Lotfi	Elevée	3 jr
Teste et validation	Fahmi et Lotfi	Elevée	1jr

Tableau 13:Planification de sprint

2. Analyse et spécification des besoins

Les besoins à réaliser dans le sprint 1, étant spécifiés et afin de mieux expliquer nous allons présenter les diagrammes des cas d'utilisation de l'authentification, gestion des rôles, gestion des utilisateurs et l'affectation des droits.

2.1 Diagramme de cas d'utilisation « Authentification »

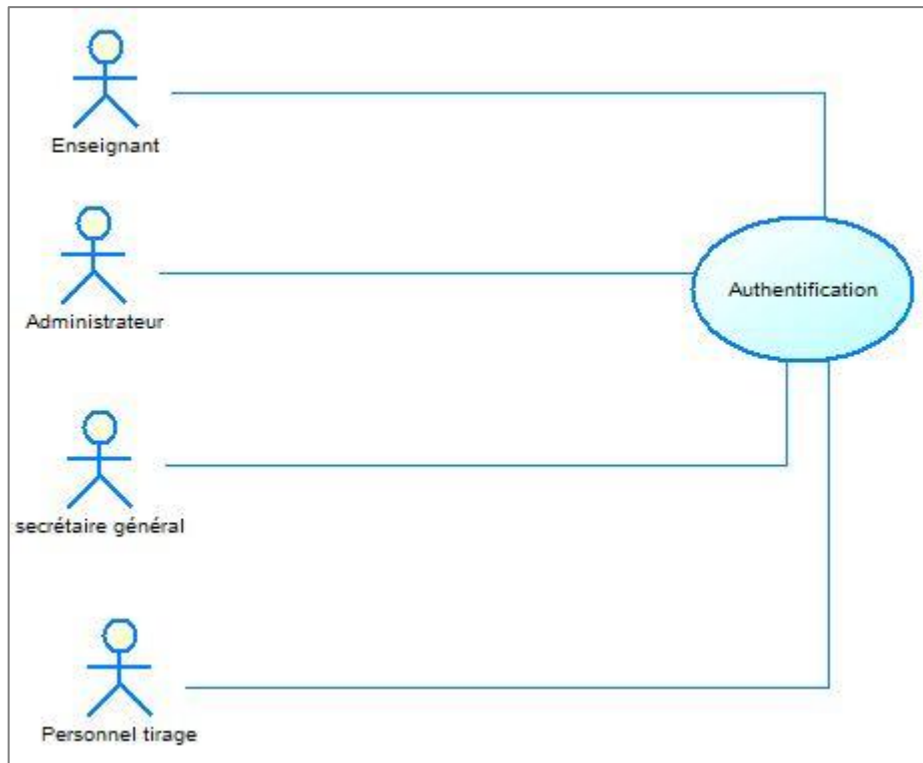


Figure 17:Diagramme de cas d'utilisation« Authentification »

Le tableau ci-contre présente les descriptions textuelles du diagramme des cas d'utilisations de l'authentification.

Cas d'utilisation	Authentification
Acteur	-Administrateur -Secrétaire général -Enseignant -personnel de tirage
Pré condition	-Besoin d'accès.
Post condition	Authentification validée et succès d'accès.
Scénario nominal	<ol style="list-style-type: none"> 1. Le système affiche la page de connexion. 2. L'utilisateur saisit son identifiant et son mot de passe. 3. Le système vérifie la validité des informations fournies. 4. Le système donne l'accès à l'utilisateur, le menu principal s'affiche.
Exception	-E1 : Coordonnées d'accès sont incorrectes. -Retour à 1 : l'interface de connexion affiche un message d'erreur.

Tableau 14:Description textuelle « Authentification »

2.2 Diagramme de cas d'utilisation « Gestion des utilisateurs »

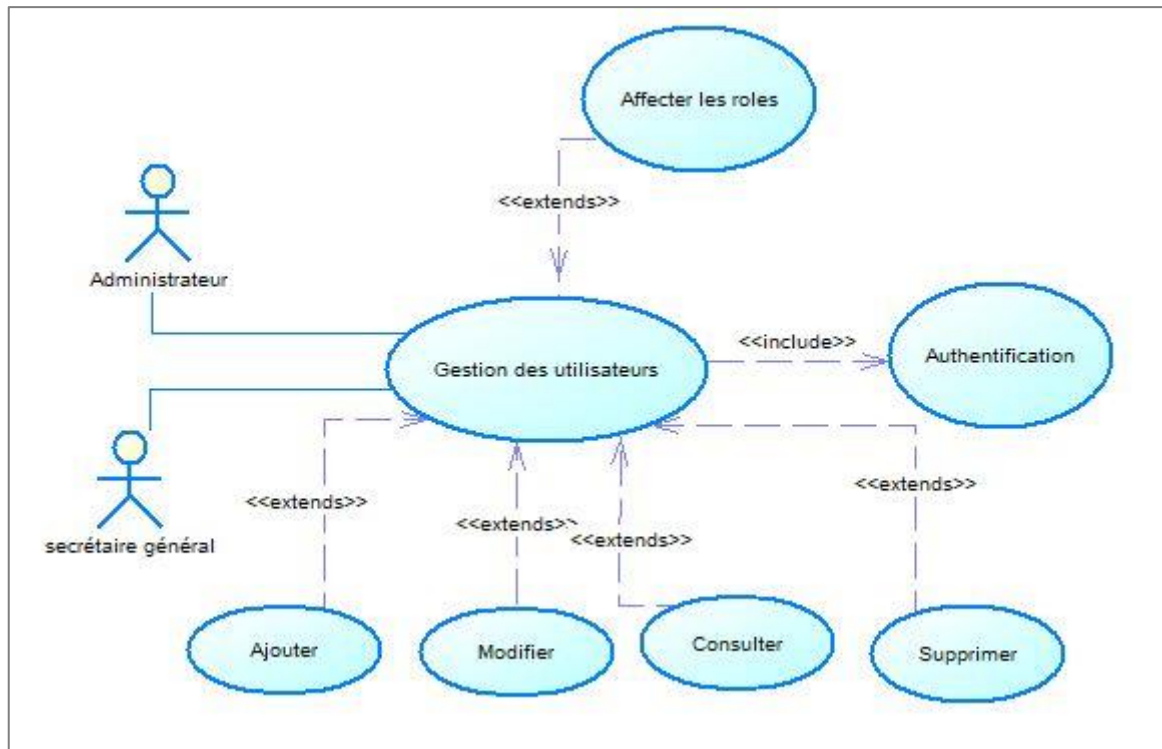


Figure 18: Diagramme de cas d'utilisation « Gestion des utilisateurs »

Le tableau ci-contre présente les descriptions textuelles du diagramme des cas d'utilisations de l'authentification.

Cas d'utilisation	Gestion des utilisateurs
Acteur	-Administrateur -Secrétaire général
Pré condition	-Besoins d'ajout, modification, suppression, consultation affectation des rôles pour les utilisateurs.
Post condition	-Ajout, modification, suppression avec succès. - consultation.
Scénario nominal	<ol style="list-style-type: none"> 1. Le système affiche un menu. 2. L'utilisateur choisi la rubrique correspondante. 3. Le système affiche le formulaire correspondant. 4. L'utilisateur saisie les informations (ajout ou modification). 5. Le système vérifie la validité des informations fournies. 6. L'utilisateur consulte les informations stockées avec la possibilité de suppression. 7. L'utilisateur sélectionne l'ensemble des utilisateurs pour affecter le rôle correspondant.
Exception	<ul style="list-style-type: none"> - E1 : Champ vide ou type d'information est incorrecte. le système affiche un message d'erreur. L'enchainement E1 reprend à l'étape 3 du scénario. - E2 : Utilisateur déjà le système affiche un message d'erreur. L'enchainement E1 reprend à l'étape 3 du scénario.

Tableau 15: Description textuelle « Gestion des utilisateurs »

3. Conception

La phase de conception est la phase initiale de création et de la mise en œuvre de notre projet. En fait, elle représente une étape importante de réflexion dans le cycle de développement logiciel. Dans ce sprint nous allons présenter notre conception à travers les diagrammes de classes, les diagrammes de séquences et les diagrammes d'activités.

3.1 Diagramme de classes

Un élément central de conception d'un logiciel est le diagramme de classes, c'est un modèle représentant la structure du système à concevoir. Le diagramme de classes est un schéma utilisé pour présenter les classes et les interfaces d'un système ainsi que les différentes relations. Dans ce sprint nous sommes concentrés sur la définition des classes nécessaires pour la gestion d'authentification, gestion des utilisateurs et l'affectation des rôles.

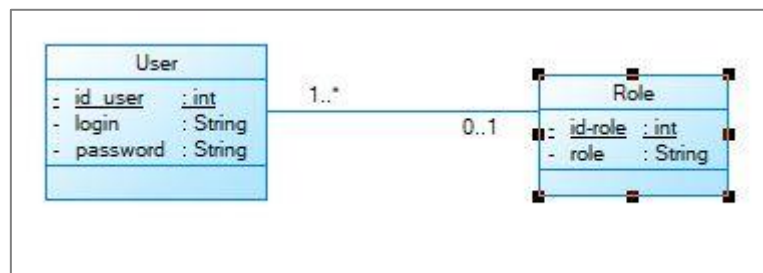


Figure 19: Diagramme de classes «gestion des utilisateurs »

Le tableau ci-contre présente les attributs du diagramme des classes relatif à « Authentification ».

Nom	Type de données	Description	Classificateur
id_user	int	Identifiant d'utilisateur	User
login	String	Login d'utilisateur	User
password	String	Mot de passe d'utilisateur	User
id-role	int	Identifiant du rôle	Role
role	String	Rôle spécifique à chaque utilisateur	Role

Tableau 16: Description des attributs des tables user et rôle

3.2 Diagramme d'activité

3.2.1 Diagramme d'activité « Authentification »

Pour que l'utilisateur puisse accéder au système, il doit s'authentifier en saisissant son login et son mot de passe. Une fois connu, l'utilisateur accède au menu des fonctionnalités qui sont lui offertes. Le processus d'authentification peut être résumé dans le diagramme d'activité suivant.

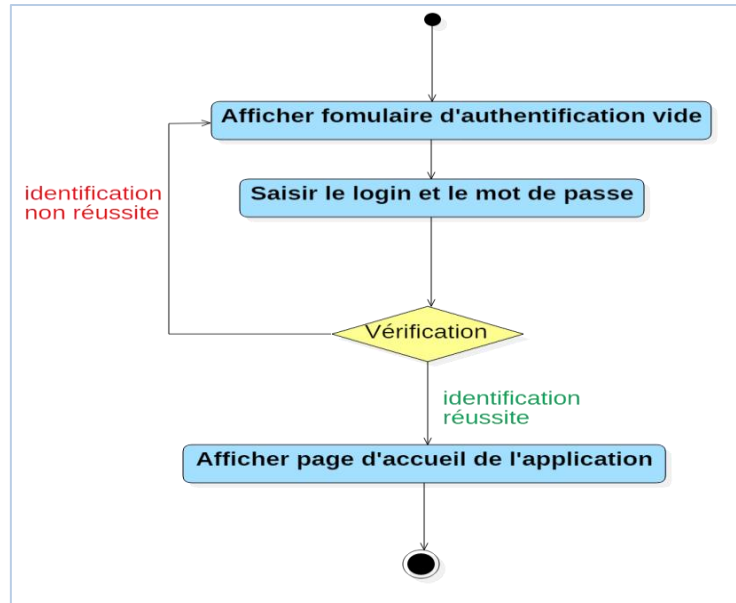


Figure 20:Diagramme d'activité « Authentification »

3.2.2 Diagramme d'activité « Gestion des utilisateurs »

Dans la figure suivante, nous présentons le processus d'ajout d'un utilisateur par le diagramme d'activité.

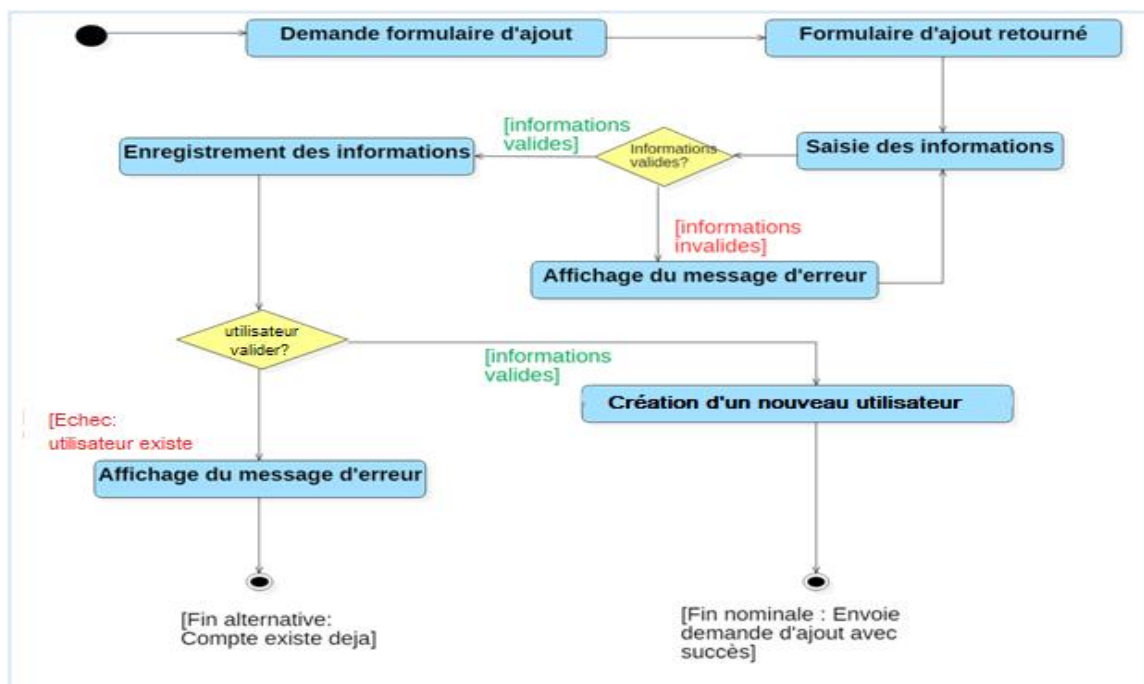


Figure 21: Diagramme d'activité « Ajouter utilisateur »

3.3 Diagramme de séquences

Dans cette partie, nous allons présenter quelques diagrammes de séquences pour modéliser l'aspect dynamique de notre système. Un diagramme de séquences a pour but de montrer les relations entre objets d'un point de vue temporel en mettant l'accent sur la chronologie des envois de messages.

3.3.1 Diagramme de séquences « Authentification »

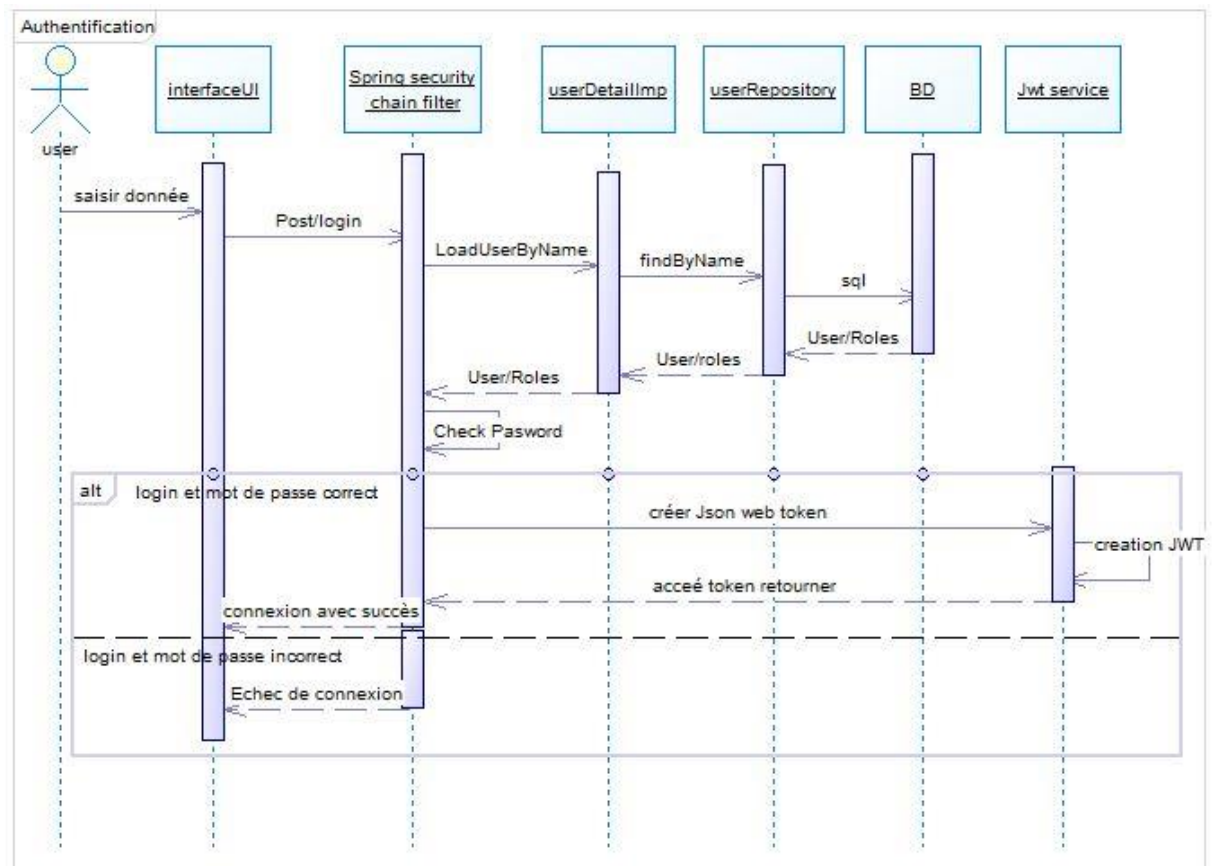


Figure 22:Diagramme de séquences « Authentification »

3.3.2 Diagramme de séquences « Affecter rôle »

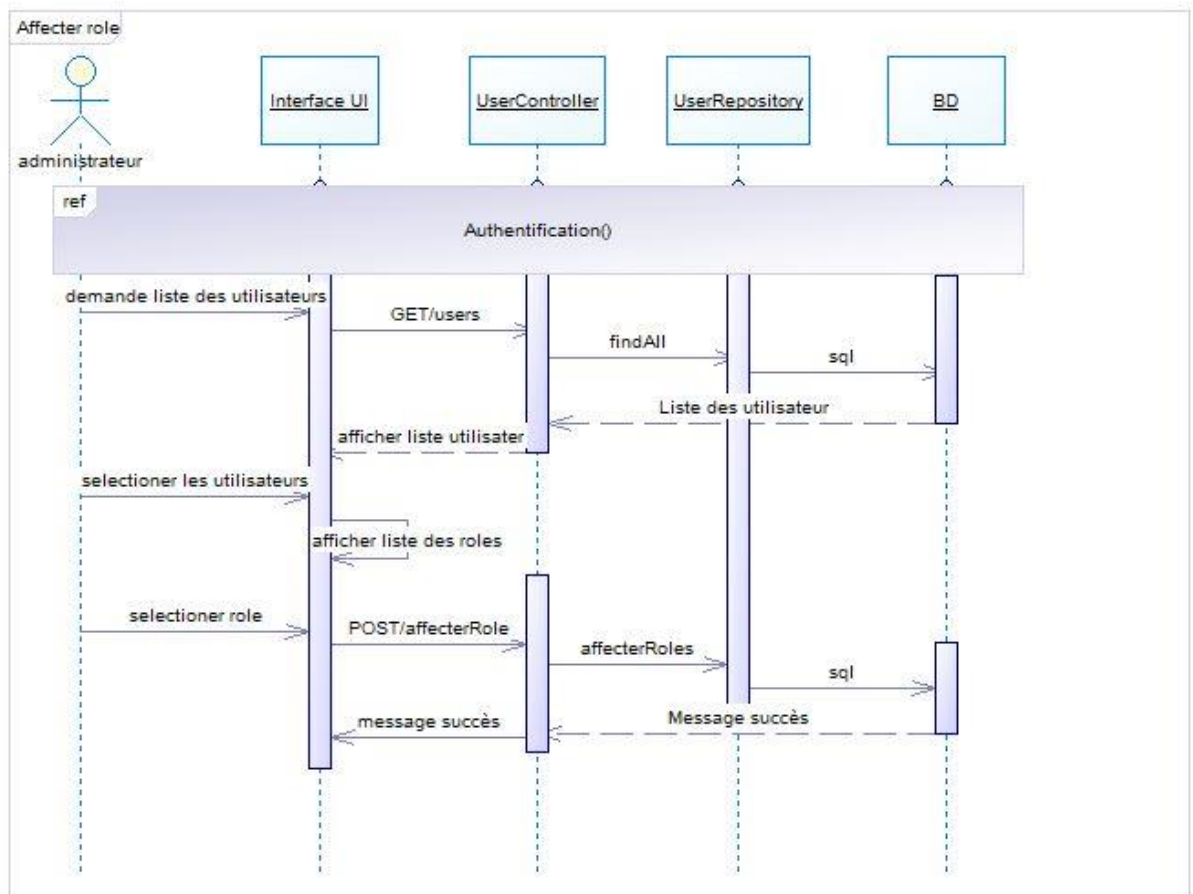


Figure 23:Diagramme de séquences « Affecter rôle »