Ecole Nationale Supérieure des Sciences Appliquées Agadir - ENSAA

Université Ibn Zohr - UIZ

Rapport du projet de fin du module JEE

Plateforme de Supervision de la Consommation d'Eau

Réalisé par :

Abderrahman BOUANANI
Abou Kekeli Efrayim

Encadré par :

Pr. BENIDER

Filière : DLA2 - 2ème Année Développement Logiciel et Applicatif

Année Universitaire : 2025–2026

Déclaration

Je soussigné(e), **Abderrahman Bouanani**, de l'ENSA Agadir, déclare que le présent rapport, y compris l'ensemble des textes, figures, tableaux, extraits de code et illustrations, constitue le fruit de mon travail personnel. Les sources externes utilisées ont fait l'objet de citations et références explicites. Je reconnais que tout manquement à cet engagement relèverait du plagiat, considéré comme une infraction académique passible de sanctions.

J'autorise la mise à disposition d'un exemplaire de ce rapport à des fins pédagogiques, au bénéfice des futurs étudiants et chercheurs, et j'accepte qu'il soit consulté dans l'intérêt général de l'Enseignement Supérieur et de la Recherche.

Abderrahman Bouanani
7 novembre 2025

Remerciements

Il est d'usage d'exprimer sa gratitude envers toutes les personnes et organismes ayant contribué à la bonne réalisation du projet. Les remerciements peuvent inclure, entre autres, l'encadrant ou les encadrants pour leur accompagnement, les amis et collègues pour leurs soutiens, ainsi que toute instance (département, laboratoire, institut, etc.) ayant mis des ressources ou des installations à disposition. Cette section reste facultative, mais elle constitue un espace pour reconnaître publiquement toute aide ou appui dont le projet a bénéficié.

Abstract

This document provides a template for a project report along with guidelines on how to write a report. It also includes several useful examples to help you become accustomed to LaTeX. The number and titles of the chapters may vary depending on the type of project and individual preferences. The section titles presented here are only illustrative and should be adapted as necessary. Likewise, the number of sections within each chapter remains flexible. This template may or may not suit your project, so it is advisable to discuss the structure of your report with your supervisor.

Guidelines for writing an abstract : The abstract is a summary of the report, presented in a single paragraph and not exceeding 250 words. It must be *self-contained* and should not refer to other sections, figures, tables, equations, or references. In general, an abstract comprises four key elements :

1. **Introduction** (background and purpose of the project) ;
2. **Methods** (experiments, techniques, or implementation) ;
3. **Results** (main conclusions obtained, along with their significance) ;
4. **Conclusions** (implications for the field of study).

The distribution of these four parts should reflect their relative importance within the body of the report. The abstract begins with a few sentences describing the general theme and main objective of the project, then presents the targeted problem. This is followed by a brief description of the methodology employed, before summarizing the results obtained and their meaning. Finally, the conclusion highlights the project's major contributions and its impact on the field.

Keywords : Up to five keywords or key phrases, separated by commas

Report's Total Word Count : The word count must be stated after the abstract. The report should contain at least 10,000 words and at most 15,000 words (counting from Chapter 1 through the end of the Conclusions chapter, but excluding the list of references, appendices, abstract, and text contained in figures, tables, listings, and captions), roughly 40 to 50 pages.

The project's source code must be uploaded to GitLab. The GitLab link should be included here, following the word count.

The report (preferably in PDF format) must be submitted via the "Ecampus" platform before the deadline. If a student has resit examinations for any taught modules, the submission deadline for the dissertation will be postponed by two weeks from the original deadline. donner en anglais :

Résumé

Ce document propose un modèle de rapport de projet ainsi que des indications sur la manière de rédiger un rapport. Il inclut également plusieurs exemples utiles permettant de s'habituer à LaTeX. Le nombre et l'intitulé des chapitres peuvent varier selon le type de projet et les préférences de chacun. Les titres de section présentés ici ne sont qu'illustratifs et doivent être adaptés. De même, le nombre de sections dans chaque chapitre reste flexible. Il est possible que ce modèle convienne plus ou moins à votre projet ; il est donc conseillé de discuter de la structure du rapport avec votre encadrant.

Conseils sur la rédaction d'un résumé : Le résumé est une synthèse du rapport, présentée en un seul paragraphe, d'une longueur maximale de 250 mots. Il doit être *autonome* et ne doit pas faire référence à d'autres sections, figures, tableaux, équations ou références. En général, un résumé comprend quatre éléments clés :

1. **Introduction** (contexte et objectif du projet) ;
2. **Méthodes** (expérimentations, techniques ou implémentation) ;
3. **Résultats** (principales conclusions obtenues, ainsi que leur portée) ;
4. **Conclusions** (implications pour le domaine d'étude).

La répartition de ces quatre parties doit refléter l'importance relative de chacune dans le corps du rapport. Le résumé débute par quelques phrases décrivant la thématique générale et l'objectif principal du projet, puis présente la problématique ciblée. Ensuite, une courte description de la méthodologie employée est donnée, avant de rappeler les résultats obtenus et leur signification. Enfin, la conclusion souligne la contribution et les retombées majeures pour le domaine concerné.

Mots-clés : maximum de cinq mots-clés ou expressions-clés, séparés par des virgules

Compte de mots du rapport : Le nombre de mots doit être indiqué après le résumé. Le rapport doit comporter au moins 10 000 mots et au maximum 15 000 mots (en comptant du premier chapitre jusqu'à la fin du chapitre de conclusion, mais en excluant la liste des références, les annexes, le résumé ainsi que les textes contenus dans les figures, tableaux, listings et légendes), soit environ 40 à 50 pages.

Le code source du projet doit être déposé sur GitLab. Le lien GitLab doit figurer ici, après le décompte de mots.

Le rapport (idéalement en format PDF) doit être remis via la plateforme « Ecampus » avant la date limite. Si un(e) étudiant(e) a des examens de rattrapage pour les modules enseignés, la date limite de remise du mémoire sera repoussée de deux semaines par rapport à la date initiale.

Liste des abréviations

Note : Trier en ordre alphabétique

DL	Deep Learning
IA	Intelligence Artificielle
JEE	Java Enterprise Edition
ML	Machine Learning

Table des figures

Liste des tableaux

Table des matières

Introduction Générale

Cette section introductive propose une vision globale du projet et des travaux qui y sont associés. De manière générale, plusieurs éléments y sont abordés :

- **Contexte et champ d'application** : un aperçu du cadre dans lequel s'inscrit le projet (théories, systèmes, algorithmes, applications concrètes, etc.), permettant de bien cerner les enjeux et la pertinence du travail à réaliser.
- **Description du problème** : une présentation succincte de la thématique ou de la question étudiée, en soulignant l'importance du sujet et les difficultés potentielles.
- **Objectifs du projet** : la définition des buts visés — ce qui doit être accompli ou démontré à la fin de l'étude ou du développement. Il peut s'agir de résoudre une problématique, de concevoir un prototype, de réaliser une expérience, etc.
- **Approche et méthodologie** : la façon dont le problème sera traité ; il peut s'agir de techniques d'implémentation, de protocoles expérimentaux ou de méthodes théoriques.
- **Résultats et interprétations attendus** : un bref résumé des conclusions majeures ou des retombées espérées (efficacité, performance, validation, etc.), sans toutefois rentrer dans les détails.
- **Organisation du rapport** : un aperçu du contenu des chapitres à venir, afin de guider le lecteur dans la structure globale du document.

Il est fortement recommandé de **consulter votre encadrant** pour valider le contenu et l'étendue de cette introduction, car la forme et la longueur de celle-ci peuvent varier selon le type de projet (développement d'application, analyse algorithmique, travaux expérimentaux, approche théorique, etc.) et selon les préférences individuelles.

Dans tous les cas, l'introduction doit fournir au lecteur **un cadre clair** sur la problématique, la démarche et les objectifs poursuivis, tout en donnant envie de découvrir la suite du rapport.

Chapitre 1

Contexte général du projet

1.1 Introduction

Ce chapitre a pour vocation de situer le projet dans son ensemble et de préciser les différents paramètres qui en encadrent la réalisation. Il permet de comprendre le lien avec les développements antérieurs (présentés dans le chapitre précédent, le cas échéant) ainsi que la façon dont il s'inscrit dans la continuité du rapport. Plus particulièrement :

- **Rappel du contexte :** Il est essentiel de rappeler les grandes lignes du domaine ou de la problématique déjà abordées auparavant. Cela permet de maintenir une cohérence globale et de souligner la progression logique du rapport.
- **Objectif du chapitre :** Ce chapitre vise à décrire la nature du projet, sa finalité ainsi que le champ de recherches ou d'applications qu'il couvre. Les informations présentées orientent et justifient les choix à venir. Elles sont également nécessaires pour comprendre les motivations et la portée du travail réalisé.

1.2 Présentation du projet

1.2.1 Sujet du projet

Le **choix du sujet** résulte souvent d'une commande (entreprise, institution, cahier des charges) ou d'une problématique identifiée (besoin d'automatisation, d'amélioration de processus, etc.). Les points suivants méritent d'être clarifiés :

- **Origine du projet :** Le contexte (académique, professionnel, industriel) dans lequel l'idée a émergé, ainsi que les motivations (ex. résolution d'un problème concret, innovation technique...).
- **Domaine d'application et utilisateurs cibles :** Identifier qui seront les principaux bénéficiaires ou acteurs (étudiants, professeurs, clients, opérateurs, etc.). Il est important de préciser les besoins et contraintes propres à ce public (environnement technique, niveau d'expertise, etc.).

1.2.2 Intérêt du projet

Cette partie met en avant la **plus-value attendue** du projet :

- **Améliorations apportées :** Souligner en quoi le projet se démarque de l'existant (par exemple, automatisation d'une tâche auparavant manuelle, optimisation d'une application, gain de temps...).
- **Impact et pertinence :** Expliquer pourquoi il est important ou nécessaire de résoudre cette problématique. Dans quel cas d'usage ou contexte (universitaire, industriel, etc.) cette solution présente-t-elle un véritable atout ?

1.3 Problématique et état de l'existant

Afin de **justifier** la mise en place du projet, il est souvent crucial de procéder à une analyse de la littérature ou des solutions déjà disponibles :

- **Solutions ou méthodes proposées :** Cette étape peut consister en une revue de l'existant (logiciels, algorithmes, démarches) ou en l'observation d'un processus manuel devenu obsolète (par exemple, saisie manuelle fastidieuse). Les principales forces et faiblesses de ces approches doivent être soulignées.
- **Défauts et limites :** Souligner clairement les points qui ne sont pas (ou mal) couverts par les solutions actuelles : complexité excessive, coûts trop élevés, manque d'ergonomie, etc. Ces lacunes mettent en évidence la **nécessité** d'une nouvelle approche ou d'une amélioration.

1.4 Objectifs du projet

Les objectifs offrent un cadre structurant pour l'ensemble du travail. Ils peuvent être **classés en différentes catégories** :

- **Objectifs fonctionnels :** Il s'agit de décrire les fonctionnalités attendues du système ou de l'application. Par exemple : gérer des comptes utilisateurs, effectuer un calcul d'angles, permettre la visualisation de données en temps réel, etc.
- **Objectifs techniques :** On aborde ici les aspects liés à la performance (temps de réponse, robustesse du système), à la maintenabilité (structure modulaire, documentation), ou encore aux technologies privilégiées (langages, frameworks, bases de données...). Ces objectifs peuvent inclure des considérations de sécurité ou de conformité à des standards.
- **Contraintes :** Les contraintes peuvent être multiples :
  - **Temporelles :** date butoir pour la livraison, temps limité pour la phase de développement ou de test.
  - **Budgétaires :** moyens financiers alloués ou ressources matérielles disponibles.
  - **Organisationnelles :** disponibilité des intervenants, politiques internes de l'institution, etc.

1.5 Démarche de gestion de projet

1.5.1 Méthodologie (Scrum, Cycle en V, etc.)

Cette partie explicite la **méthode de gestion de projet** choisie et la façon dont elle s'adapte au contexte :

- **Présenter la méthode :** Par exemple, Scrum (méthode agile) favorise l'adaptabilité et la communication fréquente avec le client, tandis que le Cycle en V implique des étapes plus linéaires (spécification, conception, validation).
- **Rôle des intervenants :** Définir qui est le client (ou commanditaire), le product owner, l'équipe de développement, etc., et expliquer brièvement les interactions (séances de feedback, revues intermédiaires...).

La figure **??** présnete le schéma du cycle Scrum et de ses principales étapes.

1.5.2 Planification du projet

La planification vient concrétiser la méthode sélectionnée :

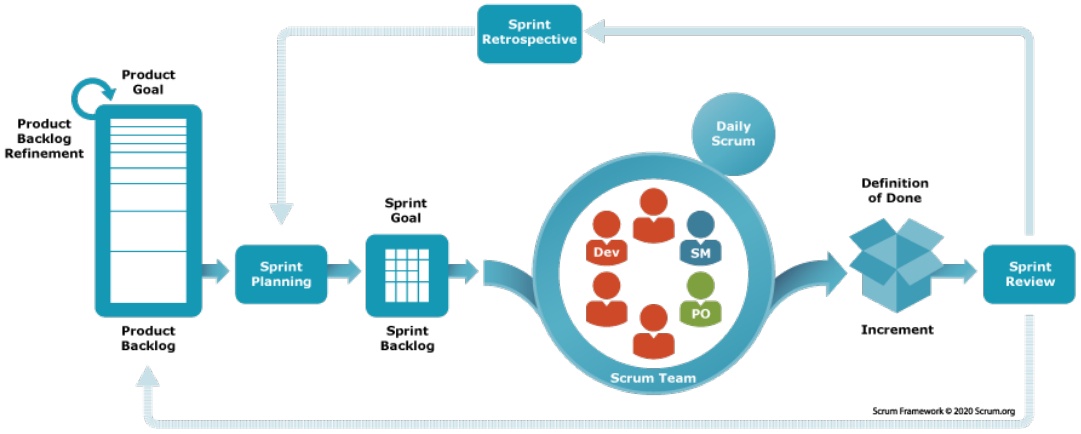

Figure 1.1 : Schéma du cycle Scrum et de ses principales étapes.

- **Phases ou sprints :** Décrire chaque phase de manière synthétique : objectifs, durée, livrables à fournir. Si vous utilisez Scrum, mentionnez le nombre de sprints et leur contenu. Dans le cas d'un Cycle en V, mettez l'accent sur les étapes (spécification, conception, implémentation, tests).
- **Outils de planification :** Un diagramme de Gantt peut illustrer le calendrier global, tandis qu'un backlog Scrum répertorie l'ensemble des tâches à accomplir. L'idéal est de décrire comment vous assurez le suivi (réunions quotidiennes, rétrospectives, logiciel de gestion de tâches, etc.).

Le tableau **??** présente le planning détaillé des sprints du projet, incluant les objectifs spécifiques, la durée de chaque sprint ainsi que les livrables attendus à l'issue de chaque phase.

Table 1.1 : Planification des sprints.

Sprint	Objectifs	Durée	Livrables
1	Mise en place de l'environnement de développement, configuration des outils et préparation du dépôt de code.	2 semaines	Environnement configuré, dépôt initial
2	Développement du module d'authentification et gestion des utilisateurs.	3 semaines	Module d'authentification opérationnel, tests unitaires
3	Conception de l'interface utilisateur et début de la gestion des données.	2 semaines	Maquettes validées, premiers écrans fonctionnels
4	Intégration des fonctionnalités de base et réalisation de tests d'intégration.	3 semaines	Application intégrée, rapport de tests
5	Optimisation des performances et finalisation des modules, préparation de la version finale.	2 semaines	Version finale livrée, documentation complète

1.6 Conclusion

En guise de synthèse, il est recommandé de récapituler les éléments clés présentés, afin de poser des bases solides pour la suite :

- **Synthèse du contenu :** Le chapitre a permis de dresser un portrait global du projet : on connaît désormais ses motivations, les solutions existantes, les objectifs à atteindre et les contraintes à respecter. Ces informations permettent de comprendre la pertinence et la finalité du travail.
- **Limites et difficultés rencontrées :** Si certaines incertitudes ou obstacles ont été identifiés (par exemple, un manque de clarté dans le cahier des charges, des variables non maîtrisées, etc.), il convient de les mentionner. On peut esquisser des pistes de solutions ou décider de les aborder dans un chapitre ultérieur.
- **Transition vers le chapitre suivant :** Les informations collectées et analysées ici constitueront la base de l'étude fonctionnelle et/ou de la conception à venir. Le prochain chapitre pourra, par exemple, détailler l'architecture envisagée, le modèle de données ou encore les diagrammes UML. Cette transition assure la cohérence de la démarche globale et guide le lecteur dans la progression du rapport.

Chapitre 2

Étude préliminaire et fonctionnelle

2.1 Introduction

Ce chapitre a pour objectif de **cadrer les besoins** auxquels le projet doit répondre, qu'ils soient liés aux fonctionnalités principales, aux contraintes techniques ou encore à l'expérience utilisateur. Il permet également de clarifier **l'identité** des différents acteurs, ainsi que leur rôle dans le système. Enfin, il détaille la logique de fonctionnement au travers de cas d'utilisation et de processus métier éventuels.

2.2 Étude des besoins

Dans cette section, on distingue généralement deux grandes catégories de besoins : **fonctionnels** (qui décrivent ce que le système doit faire) et **non fonctionnels** (liés aux performances, à la fiabilité, à la sécurité, etc.).

2.2.1 Besoins fonctionnels

- **Lister les fonctionnalités principales :** Il s'agit de décrire toutes les actions ou services que l'application devra proposer (ex. création de comptes, calcul d'un paramètre spécifique, édition de rapports, etc.).
- **Illustrer par des scénarios d'utilisation concrets :** Décrire, par exemple, les étapes qu'un utilisateur va suivre pour atteindre un objectif (inscription, consultation, validation, etc.). Cela aide à mieux cerner la séquence d'actions nécessaires et l'interface ou les interfaces prévues.

2.2.2 Besoins non fonctionnels

- **Contraintes de performance, de sécurité et de fiabilité :** Par exemple, temps de réponse maximal acceptable, taux de disponibilité, modes d'authentification ou de chiffrement des données, etc.
- **Implications sur le choix de l'architecture ou de la technologie :** Mentionner les répercussions sur les choix de framework (Spring Boot, Django, etc.), de base de données (SQL ou NoSQL), ou encore sur le design (microservices, monolithique).

2.3 Identification des acteurs

L'identification des acteurs consiste à **définir les différents profils** qui interagiront avec le système, ainsi que leurs droits et responsabilités.

- **Description des profils :** Par exemple, un administrateur, un utilisateur classique, un superviseur, un étudiant, un enseignant, etc.
- **Fonctionnalités associées :** Chaque acteur se voit attribuer des permissions spécifiques (créer un compte, valider une saisie, accéder à certaines données sensibles, etc.). Lister en détail ces habilitations permet d'anticiper la gestion des rôles et des autorisations.

2.4 Cas d'utilisation et diagrammes

Les **use cases** (cas d'utilisation) sont très utiles pour formaliser comment chaque acteur utilise le système. Ils peuvent être représentés sous forme de diagramme UML, accompagné d'une description textuelle plus précise.

- **Présentation des use cases :** Un diagramme UML permet de visualiser rapidement quelles actions sont possibles pour chaque acteur. Chaque use case peut ensuite être décrit textuellement (préconditions, flux principal, flux alternatif, postconditions).
- **Couverture des besoins fonctionnels :** Vérifier que l'ensemble des fonctionnalités clés identifiées dans la section précédente se retrouve bien dans les cas d'utilisation. Cela contribue à la cohérence du projet.

2.5 Processus métier (si nécessaire)

Dans certains projets, il peut être pertinent de détailler le **workflow global** (suite d'étapes) sous forme de **diagramme BPMN** ou de **diagramme d'activités UML**.

- **Décrire le flux d'informations :** Illustrer l'enchaînement des opérations (par exemple, traitement d'une demande, validation par un chef d'équipe, notification à un utilisateur, etc.).
- **Validations et étapes-clés :** S'il y a lieu de valider un document, de signer un contrat, de changer l'état d'une commande, etc., il faut expliciter ces jalons importants (qui valide ? comment ?).

La figure **??** présente un exemple d'un diagramme BPMN.

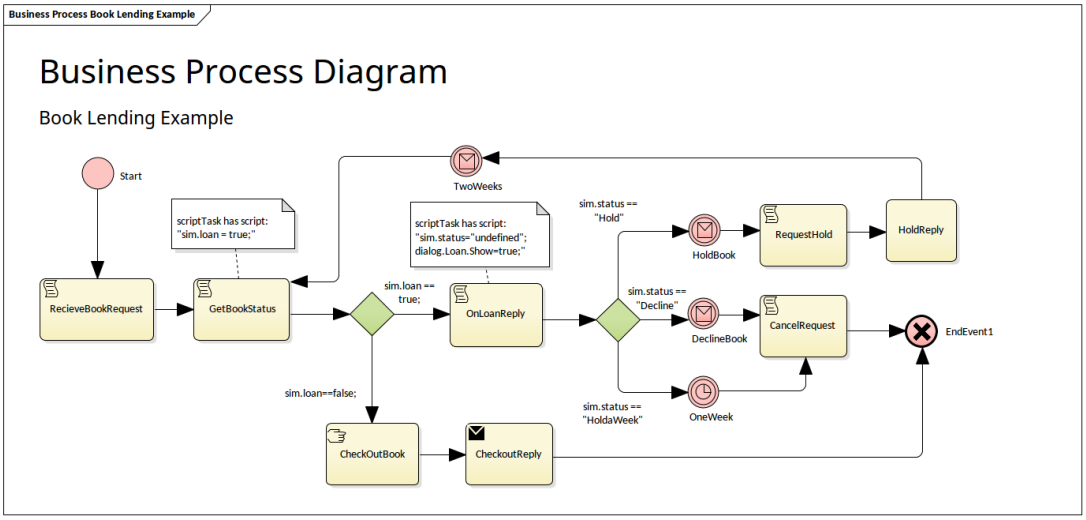

Figure 2.1 : BPMN Diagramme Processus Métier - Exemple Prêt de Livres.

2.6 Conclusion

Pour conclure, il est important de dresser un bilan synthétique de l'étude fonctionnelle et préliminaire, afin de clarifier les points suivants :

- **Résumé de la vision fonctionnelle :** Rappeler brièvement les grandes fonctionnalités prévues, les rôles et responsabilités des acteurs, et les contraintes majeures (performance, sécurité, etc.).
- **Transition vers l'état de l'art ou travaux préexistants :** Le chapitre suivant, souvent consacré à la *revue de littérature* ou à l'étude de l'existant, permettra de confronter ces besoins avec les solutions ou approches déjà disponibles, et d'affiner la pertinence du choix technique qui sera opéré.

Chapitre 3

État de l'art

3.1 Introduction

Le présent chapitre vise à positionner le projet dans son contexte scientifique et technique, en s'appuyant sur un examen des approches et des travaux déjà réalisés dans le domaine. Il a pour rôle de **situer votre démarche** par rapport aux solutions existantes et d'aider à **justifier la pertinence** de votre proposition. Les éléments abordés peuvent inclure une recherche bibliographique, une comparaison d'outils ou de produits déjà disponibles, et une mise en exergue des besoins encore non satisfaits.

3.2 Travaux connexes et solutions existantes

Dans cette section, il convient de **recenser et décrire** les principaux acteurs ou contributions pertinentes pour le projet. Ces travaux connexes peuvent être :

- **Outils ou méthodologies :** logiciels ou techniques permettant de résoudre des problèmes similaires, de gérer des données comparables, ou de mettre en place des processus d'automatisation.
- **Articles ou études notables :** même sans citer de manière formelle, on peut évoquer des recherches marquantes ayant posé des bases conceptuelles (théorie, algorithmes, etc.).
- **Solutions industrielles ou open source :** produits ou frameworks déjà disponibles sur le marché, avec leurs fonctionnalités et leur public cible.

En exposant ces différentes solutions, il est important de pointer leurs **points forts** et leurs **faiblesses**, par exemple :

- Atouts : rapidité, simplicité d'utilisation, maintenabilité, large communauté d'utilisateurs, etc.
- Limites : coûts de licence, lacunes en sécurité, manque d'extensibilité, dépendances technologiques, etc.

Cette analyse permettra de préparer une **comparaison plus poussée** dans la section suivante.

3.3 Analyse comparative

Cette partie consiste à **comparer** les solutions ou approches précédemment citées en les évaluant selon divers critères, par exemple :

- **Aspects techniques :** langage utilisé, architecture, compatibilité avec d'autres systèmes, performance, évolutivité.
- **Aspects économiques :** coûts de mise en place, de maintenance ou de licence.

- **Aspects fonctionnels :** ergonomie, richesse des fonctionnalités, capacité d'adaptation à différents scénarios.

Il peut être utile de synthétiser ces éléments dans un **tableau récapitulatif**, mettant en regard chaque solution et les critères retenus. Cette démarche met en évidence :

- **Les atouts existants** à exploiter ou à intégrer dans votre propre solution.
- **Les lacunes ou opportunités** de conception que vous envisagez de combler (ex. proposer une interface plus intuitive, une meilleure performance, etc.).

La conclusion de cette analyse comparative consiste à **dégager la spécificité** ou l'originalité de votre projet. Vous pouvez y souligner si votre proposition apporte une innovation (méthode inédite), une performance supérieure, ou répond à un cas d'utilisation jusqu'ici négligé.

3.4 Conclusion

Pour conclure ce chapitre, il est essentiel de faire une **mise en perspective** :

- **Bilan de l'analyse :** rappeler brièvement les solutions déjà rencontrées et les raisons pour lesquelles elles peuvent être jugées incomplètes ou perfectibles dans le cadre du projet.
- **Orientation :** expliquer en quoi cette étude de l'existant oriente ou confirme vos choix stratégiques (technologiques, méthodologiques, etc.).
- **Transition vers le chapitre suivant (Conception) :** annoncer que les conclusions tirées ici serviront de base pour *élaborer une architecture adaptée*, définir une modélisation adéquate (UML, diagrammes de classes, etc.), ou sélectionner les outils de développement appropriés (framework web, base de données, etc.).

Ainsi, le chapitre «État de l'art» fournit un **cadrage clair** des approches préexistantes et met en relief la **valeur ajoutée** que le futur système compte proposer. Il pave la voie au prochain chapitre, qui se concentrera sur la **conception** et l'architecture de la solution.

Chapitre 4

Analyse et conception

4.1 Introduction

Cette section présente l'objectif du chapitre, qui consiste à traduire les besoins fonctionnels et non fonctionnels du projet en une solution technique détaillée. Elle explique comment les choix de modélisation, d'architecture et de conception ont été déterminés et justifiés, et comment ils s'inscrivent dans la logique globale du projet.

4.2 Formalisme de modélisation (UML, etc.)

Motivation et choix

Décrivez ici pourquoi vous avez choisi d'utiliser un langage de modélisation tel que l'UML. Précisez en quoi ce formalisme aide à :

- Décrire la structure et le comportement du système.
- Communiquer efficacement entre les membres de l'équipe.
- Garantir la cohérence et la maintenabilité du projet.

Diagrammes retenus

Listez et décrivez les différents diagrammes que vous avez retenus pour modéliser le système, par exemple :

- **Diagramme de classes :** pour représenter la structure statique (entités, attributs, méthodes, relations).
- **Diagrammes de séquence :** pour illustrer l'enchaînement des interactions lors de scénarios clés.
- **Autres diagrammes (activités, composants, déploiement) :** selon les besoins du projet.

4.3 Architecture logicielle

Cette section décrit l'**architecture globale** du système, en distinguant clairement l'approche adoptée et en expliquant la répartition fonctionnelle et technique du code.

4.3.1 Approche architecturale

Présentez ici le choix entre une architecture **monolithique** et une architecture **microservices** en détaillant :

- **Architecture monolithique :**

  - *Avantages* : simplicité de développement initial, déploiement unifié.
  - *Inconvénients* : difficulté de maintenance et de scalabilité sur le long terme.
- **Architecture microservices :**
  - *Avantages* : scalabilité fine, isolation des pannes, flexibilité technologique.
  - *Inconvénients* : complexité de déploiement et de gestion des communications inter-services.

Justifiez le choix fait pour votre projet en fonction des exigences et contraintes identifiées.

4.3.2 Schéma de l'architecture et organisation en couches

Expliquez la division du système en différentes couches, par exemple :

- **Couche de présentation :** gère l'interface utilisateur.
- **Couche de logique métier :** contient les règles de traitement et la gestion des opérations.
- **Couche de données :** responsable de la persistance et de l'accès aux informations.

Vous pouvez également présenter un schéma illustrant les interactions entre ces couches.

4.3.3 Discussion

Précisez en quoi cette architecture répond aux besoins fonctionnels et non fonctionnels du projet (maintenabilité, scalabilité, performance, sécurité, etc.) et comment elle prépare la voie pour l'implémentation.

4.4 Diagrammes de conception

Cette section détaille les différents diagrammes qui ont servi à formaliser la conception du système.

4.4.1 Diagramme de classes

- **Présentation des entités :** listez les classes principales, leurs attributs et leurs méthodes.
- **Relations et cardinalités :** décrivez les associations, agrégations, compositions et héritages, en précisant les cardinalités.

4.4.2 Diagrammes de séquence

- **Scénarios clés :** illustrez l'enchaînement des interactions pour des processus importants (exemple : authentification, traitement d'une requête, etc.).
- **Flux d'interaction :** montrez comment les messages circulent entre les objets ou composants pour répondre à une action utilisateur.

4.4.3 Autres diagrammes utiles

Selon les spécificités du projet, vous pouvez également inclure :

- **Diagramme de composants :** pour visualiser la répartition modulaire du système et les dépendances entre les modules.
- **Diagramme d'activités :** pour représenter le flux de travail global ou des processus métier complexes.
- **Diagramme de déploiement :** pour illustrer la répartition des composants sur l'infrastructure matérielle (serveurs, conteneurs, cloud, etc.).

4.5 Conclusion

Pour conclure ce chapitre, il convient de récapituler les éléments essentiels présentés :

- **Synthèse des choix d'architecture et de modélisation :** rappeler brièvement le modèle choisi (architecture et formalisme de modélisation) et les diagrammes qui en résultent.
- **Justification des décisions :** expliquer en quoi ces choix répondent aux besoins identifiés et aux contraintes du projet (modularité, évolutivité, sécurité, performance, etc.).
- **Transition vers la phase de réalisation :** indiquer que les modèles présentés serviront de base pour l'implémentation effective du système, qui sera détaillée dans le chapitre suivant.

Ce chapitre d'analyse et de conception constitue ainsi le socle technique sur lequel reposera la réalisation du projet.

Chapitre 5

Technologies et réalisation

Ce chapitre décrit l'ensemble des technologies, outils et méthodes qui ont été utilisés pour la mise en œuvre du projet. Il présente d'abord les choix techniques, puis détaille l'environnement de déploiement, l'implémentation du système (structure du code et interfaces utilisateur), ainsi que la stratégie de tests appliquée. Enfin, une conclusion synthétique fait le point sur les réalisations et les éventuelles difficultés rencontrées.

5.1 Introduction

Dans cette section introductive, l'objectif est de donner une vue d'ensemble des aspects techniques et pratiques abordés dans ce chapitre. On y explique notamment :

- Les choix technologiques réalisés en fonction des besoins fonctionnels et des contraintes techniques.
- La manière dont l'environnement de déploiement est configuré pour supporter le système.
- Les détails de l'implémentation du code et de l'interface utilisateur.
- La stratégie adoptée pour valider et tester le système.

Cette partie prépare le lecteur à comprendre comment les décisions prises au niveau de la conception se traduisent concrètement lors de la réalisation.

5.2 Choix techniques et outils

Cette section détaille les principales technologies et outils qui ont été sélectionnés pour le développement du projet.

5.2.1 Langages, Frameworks, Bases de données...

- **Langages de programmation :** Justifier le choix du ou des langages (par exemple, Java pour la robustesse et la scalabilité, Python pour la rapidité de développement ou JavaScript/React pour le front-end interactif).
- **Frameworks :** Expliquer la sélection des frameworks, tels que Spring Boot pour le back-end, React ou Angular pour le front-end, ou encore d'autres frameworks spécifiques à la tâche.
- **Bases de données :** Préciser le type de base de données utilisée (relationnelle comme MySQL/PostgreSQL ou NoSQL comme MongoDB) en fonction des critères de performance, de flexibilité et de volume de données.
- **Critères de sélection :** Indiquer les critères qui ont guidé ces choix, par exemple la taille de la communauté, la robustesse, la simplicité d'intégration et la performance.

Exemple d'utilisation des références : Dans ce rapport, diverses références ont été utilisées pour étayer la démarche adoptée. Par exemple, la conception de sites web modernes est détaillée dans **?**, tandis que **?** offre une vue d'ensemble de l'architecture Java EE. Pour une approche approfondie de l'ORM avec Hibernate, on peut se référer à **?**. Enfin, **?** fournit un guide complet sur le développement de JavaServer Pages.

5.2.2 Outils de développement (IDE, gestion de versions...)

- **Environnement de développement intégré (IDE) :** Mentionner les outils utilisés, par exemple VS Code, Eclipse ou IntelliJ IDEA, et expliquer pourquoi ces outils ont été privilégiés.
- **Système de gestion de versions :** Indiquer l'utilisation d'un outil comme Git (hébergé sur GitHub ou GitLab) pour assurer le suivi des modifications, faciliter la collaboration et la gestion du code source.
- **Autres outils de collaboration et de tests :** Par exemple, l'utilisation de Maven ou Gradle pour la gestion de projet, ainsi que des outils de CI/CD pour automatiser les builds et les tests.

5.3 Architecture de déploiement

Cette section décrit l'infrastructure sur laquelle le système sera déployé et les outils utilisés pour assurer son fonctionnement continu.

- **Environnement cible :** Décrire le ou les serveurs, l'utilisation éventuelle de conteneurs (Docker) ou du cloud (AWS, Azure, Google Cloud), ainsi que les systèmes d'exploitation concernés.
- **Pipeline d'intégration continue (CI/CD) :** Indiquer si un pipeline CI/CD a été mis en place pour automatiser les phases de build, de tests et de déploiement, et présenter les outils utilisés (Jenkins, GitLab CI, Travis CI, etc.).

5.4 Implémentation et interfaces

Cette section est consacrée à la description de l'implémentation du système ainsi qu'à la présentation des interfaces utilisateur.

- **Structure du code :** Exposer l'organisation du code en packages ou modules. Décrire brièvement la répartition entre le front-end, la logique métier et la couche de données.
- **Interfaces utilisateur :** Présenter l'interface graphique à travers des captures d'écran ou des maquettes. Décrire les principales fonctionnalités disponibles pour l'utilisateur final.
- **Fonctionnalités principales :** Détaillez les fonctionnalités clés implémentées, tant côté front-end (navigation, formulaires, affichage des données) que côté back-end (gestion des requêtes, traitement des données, sécurité).

5.5 Stratégie de tests

Pour garantir la qualité et la robustesse du système, une stratégie de tests complète a été mise en place.

- **Tests unitaires :** Décrire la couverture des tests unitaires qui vérifient le fonctionnement de chaque composant individuel (par exemple, avec JUnit, pytest, etc.).
- **Tests d'intégration :** Expliquer comment les différents modules du système ont été testés ensemble pour s'assurer de la bonne communication entre eux.
- **Tests système :** Préciser si des tests globaux ont été effectués pour valider le comportement complet du système, incluant l'interface utilisateur.

- **Outils de test et plan de validation :** Mentionner les outils utilisés (Selenium pour les tests d'interface, Postman pour les API, etc.) et résumer le plan de validation ainsi que les résultats obtenus de manière synthétique.

5.6 Conclusion

Pour clore ce chapitre, il est important de faire un bilan sur l'ensemble des décisions techniques et de réalisation qui ont été prises :

- **Synthèse des choix techniques :** Récapituler les technologies, outils et méthodologies adoptés, en insistant sur leur adéquation avec les besoins du projet (simplicité d'utilisation, performance, évolutivité, etc.).
- **Retour sur l'architecture de déploiement et l'implémentation :** Expliquer brièvement comment l'environnement de déploiement et la structure du code assurent la stabilité et la pérennité du système.
- **Perspectives et défis :** Mentionner les difficultés éventuelles rencontrées lors de l'implémentation (limitations techniques, problèmes d'intégration, etc.) et indiquer les pistes d'amélioration à envisager pour les phases ultérieures.
- **Transition vers le chapitre suivant :** Annoncer que les éléments techniques présentés ici constitueront la base pour l'implémentation concrète du système, qui sera détaillée dans le chapitre de réalisation pratique.

Ce chapitre constitue ainsi un socle technique solide, garantissant que les choix de développement sont cohérents avec les objectifs du projet et prêts à être mis en œuvre lors de la phase de réalisation.

Conclusion générale

La conclusion générale met en lumière les principaux enseignements et aboutissements du projet. Elle récapitule les objectifs initiaux, synthétise les contributions et résultats obtenus, identifie les limites du travail réalisé, et propose des pistes pour des développements futurs.

- **Récapitulation des objectifs initiaux :**
  - Les problématiques et questions de recherche abordées dès le départ sont rappelées afin de souligner l'enjeu initial du projet.
  - Une évaluation est présentée pour vérifier dans quelle mesure les objectifs techniques et fonctionnels ont été atteints.
- **Contributions et résultats obtenus :**
  - Les réalisations concrètes du projet (fonctionnalités développées, prototypes validés, analyses réalisées, etc.) sont synthétisées.
  - Les apports spécifiques du projet sont mis en avant, qu'il s'agisse d'innovations techniques, de gains de temps ou d'améliorations de performance.
- **Limites du travail :**
  - Les points de blocage et contraintes rencontrés (budgétaires, délais, aspects techniques) sont identifiés.
  - Une analyse critique montre en quoi ces limites peuvent influencer l'utilisation ou l'évolution du système développé.
- **Perspectives et améliorations futures :**
  - Des pistes d'évolution sont proposées, telles que l'ajout de nouvelles fonctionnalités, l'optimisation des performances, ou la migration vers des technologies plus avancées.
  - Pour les projets à dimension scientifique, des axes de recherche complémentaires sont suggérés pour approfondir les résultats obtenus.

Apport du projet

Cette section offre l'occasion de prendre du recul sur l'expérience acquise au cours de ce projet. Plutôt que de simplement lister les compétences techniques (maîtrise de divers langages de programmation, rédaction de rapports en LaTeX, utilisation d'outils techniques divers), il s'agit ici de réfléchir sur l'ensemble du processus de recherche et de développement, sur la démarche de résolution de problèmes et sur l'impact de cette expérience sur votre parcours professionnel et personnel.

Au début du projet, les objectifs étaient clairement définis et orientés vers la résolution d'une problématique spécifique. Au fil de la réalisation, il est apparu que la démarche de recherche et d'expérimentation permettait non seulement de valider des choix techniques, mais également de développer une approche structurée de la résolution de problèmes. Les différentes étapes, de l'identification des besoins à la mise en œuvre des solutions, ont constitué un véritable apprentissage en gestion de projet et en innovation.

Parmi les connaissances et compétences développées, on peut citer :

- La maîtrise de plusieurs langages et frameworks, permettant d'appréhender des technologies variées et de choisir la solution la plus adaptée aux exigences du projet.
- L'utilisation de LaTeX pour la rédaction de rapports professionnels, ce qui a permis d'acquérir une rigueur dans la mise en forme et la structuration des documents techniques.
- L'adoption de méthodes de travail agiles, notamment l'approche Scrum, qui a favorisé une meilleure organisation, une communication efficace au sein de l'équipe et une capacité à s'adapter aux imprévus.

Toutefois, ce projet a également présenté son lot de défis. Certaines contraintes, telles que des délais stricts ou des limitations techniques, ont parfois rendu difficile la mise en œuvre complète de toutes les fonctionnalités envisagées. Des difficultés ont notamment été rencontrées dans l'intégration de modules complexes et dans l'optimisation des performances du système. Ces obstacles ont été autant d'occasions de repenser la stratégie initiale et d'adapter les choix techniques en fonction des réalités du terrain.

Si ce projet devait être réalisé à nouveau, plusieurs axes pourraient être améliorés. Par exemple, une phase de planification plus approfondie permettrait d'anticiper davantage les risques et d'allouer les ressources de manière plus efficace. De même, une collaboration renforcée avec des experts techniques pourrait faciliter la résolution de problèmes complexes et accélérer la mise en œuvre de solutions innovantes. Enfin, une meilleure documentation interne dès le départ aurait permis de simplifier la maintenance et l'évolution du système.

Annexe A

Annexes (Facultatif)

Les annexes regroupent des documents complémentaires — tels que des tableaux volumineux, des extraits de code, des données brutes, des démonstrations détaillées ou d'autres preuves — qui, bien qu'importants pour une compréhension approfondie du projet, ne constituent pas l'essence même du rapport. Une annexe est destinée aux lecteurs souhaitant obtenir des informations supplémentaires ou une explication plus complète sur certains aspects du projet. Il est recommandé d'utiliser une annexe par idée ou par thème afin de maintenir une organisation claire et structurée.

L'inclusion d'annexes est facultative. Si vous estimez que les informations supplémentaires ne sont pas indispensables à la compréhension du projet, il est préférable de ne pas les ajouter. En effet, inclure des documents non pertinents ou superflus pourrait augmenter indûment le nombre total de pages du rapport et distraire le lecteur.

Annexe B

Annexes (Facultatif)

...

Abderrahman Bouanani

Résumé du Projet

Le présent projet vise à développer une solution innovante destinée à [décrire brièvement l'objectif principal du projet]. Pour atteindre cet objectif, une approche méthodologique rigoureuse a été adoptée, combinant une analyse fonctionnelle détaillée, une conception modulaire et l'implémentation de technologies modernes.

Les contributions majeures du projet incluent la mise en œuvre d'une interface utilisateur intuitive, l'intégration d'un système performant de gestion des données et la validation de la solution via des tests exhaustifs. Ces résultats traduisent une amélioration significative en termes de [performance, réactivité, sécurité, etc.] par rapport aux solutions existantes.

Ce projet a permis d'acquérir des compétences avancées en [développement web, modélisation UML, architecture logicielle, etc.], tout en consolidant la compréhension de la gestion de projet dans un environnement technique exigeant. En outre, il ouvre des perspectives prometteuses pour de futurs développements, tels que l'ajout de nouvelles fonctionnalités, l'optimisation des performances et l'amélioration de l'ergonomie de l'interface utilisateur.

L'ensemble de ces réalisations témoigne de l'engagement à proposer une solution à la fois innovante et efficace, répondant aux problématiques identifiées dès le début du projet.

Mots clés : solution innovante, gestion des données, interface utilisateur, tests exhaustifs, optimisation des performances