



Etablissement d'enseignement
supérieur privé
Université privée de Monastir
POLYTECH MONASTIR

Rapport Projet de Fin d'année

Approche Agile SCRUM

**Déployer et orchestrer un projet
e-commerce en MERN Stack**

Réalisé par :

Saad Raja

Encadrante pédagogique : **M Hatem Jarboua**

Année Universitaire : 2024 - 2025

Remerciements

Au terme de ce travail nous tentons à exprimer notre sincère gratitude à tous ceux qui ont contribué dans la réalisation de ce projet .

En premier lieu, nous tenons à exprimer notre profonde gratitude et nos sincères remerciements à notre professeur M. Hatem Jarboua pour sa qualité d'encadrement, sa motivation professionnelle, sa gentillesse et sa patience.

En second lieu, nous remercions également notre famille pour leur soutien.

Finalement, Un grand merci aux membres du jury pour leur intérêt. à ce travail en acceptant de nous faire l'honneur de juger ce travail avec autant de gentillesse, de disponibilité et de rigueur.

Table des matières

Introduction générale	1
1 Cadre général du projet	2
1.1 Étude de l'existant	3
1.2 Description de l'existant	3
1.2.1 Critique de l'existant	4
1.2.2 Solution proposée	4
1.3 Méthodologie et approche de développement	5
1.3.1 Choix de la méthodologie	5
1.3.2 Méthodologie de gestion de projet	5
1.3.3 Méthodologie de modélisation et de conception	7
1.3.4 Environnement Logiciel (langages et technologies)	8
1.4 Conception Architecturale	13
1.4.1 Architecture Logique	13
2 Étude préalable	15
2.1 Spécification des besoins	16
2.1.1 Identification des acteurs	16
2.2 Identification des besoins fonctionnels	16
2.2.1 Identification des besoins non fonctionnels	18
2.3 Backlog Produit Priorisé	19
2.4 Planification des Sprints	20
3 Sprint 1 : Développement de l'application	23
3.1 Mise en place de l'environnement de développement	24
3.2 Planification et objectifs du Sprint 1	24
3.3 Backlog du Sprint 1	25
3.4 Analyse	26

3.4.1	Représentation du diagramme de cas d'utilisation	26
3.4.2	Conception	26
3.5	Réalisation	29
3.5.1	Revue de Sprint	29
3.6	Rétrospective du Sprint	33
4	SPRINT 2 – Conteneurisation de l'application avec Docker	36
4.1	Nom de Sprint 2	37
4.2	Planification et Objectif du Sprint 2	37
4.3	Le Backlog du Sprint 2	38
4.4	Analyse	38
4.4.1	Représentation du diagramme de cas d'utilisation	38
4.5	Conception	42
4.5.1	Diagramme de classe	42
4.5.2	Diagramme d'activité	43
4.6	Réalisation	46
4.6.1	Revue de Sprint	46
4.7	Rétrospective du Sprint 2	50
5	Gestion et interaction communautaire	53
5.1	Nom de Sprint 3	54
5.2	Planification et Objectif du Sprint 3	54
5.3	Le Backlog du Sprint 3	55
5.4	Analyse	55
5.4.1	Représentation du diagramme de cas d'utilisation	55
5.4.2	Description textuelle des cas d'utilisation	57
5.5	Conception	59
5.5.1	Diagramme d'activité	59
5.5.2	Diagramme de classe du sprint 3	60
5.6	Réalisation	60

5.6.1	Revue de Sprint	60
5.7	Rétrospective du Sprint 3	66
	Conclusion générale	68

Table des figures

1.1	SCRUM	6
1.2	Equipe SCRUM	7
1.3	MERN Stack	9
1.4	Mongodb	10
1.5	Express	10
1.6	React	11
1.7	Node	11
1.8	Docker	12
1.9	Docker Compose	12
1.10	GitHub Action	12
1.11	Architecture 3-tiers de l'application	14
3.1	Diagramme de cas d'utilisation	26
3.2	Diagramme d'activité : processus de commande	27
3.3	Diagramme de classe avec l'entité User	28
3.4	Diagramme de classe avec l'entité Admin	29
3.5	Interface de connexion	30
3.6	Interface D'inscription	31
3.7	Interface Forgot Password	31
3.8	Interface Mongo Express : consultation de la base admin	32
3.9	Mongo Express : affichage de la base db-ecommerce et de ses collections .	33
4.1	backlog sprint 2	38
4.2	Diagramme de cas d'utilisation du sprint 2	40
4.3	Description textuelle du cas d'utilisation	42
4.4	Diagramme de classe	43
4.5	Diagramme d'activité	44
4.6	Burnup report du sprint 2	46

4.7	Réunion quotidien de l'équipe	47
4.8	interface de vote	48
4.9	interface de commentaire	49
4.10	Interface " Reset password "	49
4.11	Interface " Email de réinitialisation "	50
5.1	Backlog sprint3	55
5.2	Diagramme de cas d'utilisation du sprint 3	56
5.3	Description textuelle	57
5.4	Diagramme d'activité de sprint 3	59
5.5	Diagramme de classe de sprint 3	60
5.6	Burnup report	61
5.7	Réunion quotidienne de l'équipe	61
5.8	interface de rejoindre une communauté	62
5.9	interface de signaler	63
5.10	interface de recherche	64
5.11	interface de modifier profile	65

Liste des tableaux

1.1	Environnement logiciel	8
2.1	Besoins fonctionnels du projet e-commerce MERN	18
2.2	Backlog Produit Priorisé de la plateforme e-commerce	20
2.3	Planification des Sprints selon Scrum	21

Introduction générale

Dans un environnement technologique en constante évolution, le commerce en ligne se positionne comme un pilier stratégique pour les entreprises qui cherchent à rester compétitives et à répondre rapidement aux attentes de leurs clients. La clé du succès réside dans la capacité à développer des solutions e-commerce non seulement performantes et évolutives, mais également agiles et réactives face aux changements rapides du marché. Dans ce contexte, l'adoption de pratiques **DevOps** devient essentielle pour assurer une gestion optimale du développement, du déploiement et de l'infrastructure.

Le projet *Déployer et orchestrer un projet e-commerce en MERN Stack* s'inscrit dans cette dynamique, avec pour objectif de créer une plateforme e-commerce moderne et scalable, capable de supporter des volumes de transactions importants tout en offrant une expérience utilisateur fluide et sécurisée. En s'appuyant sur la stack MERN (MongoDB, Express.js, React, Node.js), ce projet met en œuvre une architecture robuste et flexible, tout en intégrant des pratiques **DevOps** pour automatiser les pipelines de développement, garantir une intégration continue (CI) et un déploiement continu (CD), et optimiser la gestion des versions.

L'utilisation d'outils comme **Jenkins** pour l'orchestration des pipelines CI/CD, **Docker** pour la conteneurisation de l'application, et **GitHub Actions** pour l'automatisation des déploiements, permet de répondre aux enjeux modernes de scalabilité et de maintenance continue. Grâce à ces outils, ce projet vise à offrir une solution e-commerce agile, où les mises à jour sont rapides et sûres, et où la collaboration entre les équipes de développement et d'opérations devient fluide et efficace.

Ce rapport détaille les étapes clés de ce projet, les choix technologiques adoptés, les méthodologies **DevOps** implémentées, ainsi que les défis surmontés pour garantir la réussite de cette solution e-commerce performante et évolutive.

CHAPITRE 1

CADRE GÉNÉRAL DU PROJET

Plan

1	Étude de l'existant	3
2	Description de l'existant	3
3	Méthodologie et approche de développement	5
4	Conception Architecturale	13

Introduction

Le projet de fin d'année que je développe se concentre sur la mise en place d'un système d'intégration continue et de déploiement continu (CI/CD) pour une plateforme de e-commerce utilisant la stack MERN (MongoDB, Express, React, Node.js). L'objectif principal est de créer une solution e-commerce entièrement scalable et flexible, en intégrant des pratiques DevOps avancées afin de garantir des déploiements rapides et fiables.

La stack MERN a été choisie pour sa capacité à offrir une intégration fluide entre le frontend et le backend, tout en répondant aux exigences de performance et de modularité des applications modernes. En parallèle, l'utilisation de Docker permet de conteneuriser l'application, assurant ainsi une portabilité maximale et une gestion simplifiée des environnements de développement, de test et de production. Le processus CI/CD, orchestré par GitHub Actions, automatise non seulement les déploiements, mais aussi les tests unitaires, assurant ainsi une qualité constante du code tout au long du cycle de vie du projet.

Ce projet illustre la puissance des pratiques DevOps pour améliorer la collaboration entre les équipes de développement et d'exploitation, tout en réduisant les risques associés au déploiement, à la maintenance et à l'évolutivité des applications web modernes.

1.1 Étude de l'existant

1.2 Description de l'existant

Les solutions de développement et de déploiement des plateformes e-commerce traditionnelles présentent souvent des limitations en termes de scalabilité, de gestion des environnements et d'automatisation des déploiements. Ces systèmes monolithiques manquent de flexibilité, sont difficiles à maintenir et ralentissent les cycles de développement. En l'absence de pratiques DevOps, les environnements de développement, de test et de production sont souvent incohérents, ce qui engendre des erreurs et allonge les délais de mise en production. Ce projet propose une approche moderne utilisant la stack MERN,

Docker et GitHub Actions pour surmonter ces défis, en offrant une solution plus agile, scalable et automatisée.

1.2.1 Critique de l'existant

Les solutions traditionnelles de développement e-commerce, souvent basées sur des architectures monolithiques, manquent de flexibilité et sont difficiles à maintenir, rendant les mises à jour et évolutions complexes. L'absence de pratiques DevOps et de pipelines CI/CD entraîne des déploiements manuels sujets à des erreurs humaines, augmentant les coûts et les délais. De plus, la gestion fragmentée des environnements de développement, de test et de production génère des incohérences, rendant difficile un déploiement fluide. Ces limitations appellent une transformation vers des solutions modernes, comme la stack MERN, Docker et GitHub Actions, qui permettent de surmonter ces défis et d'améliorer l'agilité et l'efficacité des processus.

1.2.2 Solution proposée

Afin de surmonter les limitations des solutions existantes, ce projet propose la mise en place d'une plateforme e-commerce moderne basée sur la stack MERN (MongoDB, Express, React, Node.js), associée à des pratiques DevOps avancées pour garantir une scalabilité, une flexibilité et une automatisation optimales. L'utilisation de Docker permet de conteneuriser l'application, assurant ainsi une portabilité maximale entre les différents environnements de développement, de test et de production. De plus, l'intégration de GitHub Actions pour l'automatisation des pipelines CI/CD garantit un déploiement rapide et fiable, tout en intégrant des tests unitaires pour assurer la qualité du code à chaque étape.

Cette solution vise à améliorer l'agilité du processus de développement et à réduire les risques liés aux déploiements grâce à l'automatisation et à la gestion cohérente des environnements. En combinant ces technologies modernes, le projet permet non seulement de répondre aux besoins de performance et de scalabilité des plateformes e-commerce actuelles, mais aussi d'optimiser la collaboration entre les équipes de développement et d'opérations.

1.3 Méthodologie et approche de développement

Cette méthodologie, associée à une approche DevOps et l'utilisation des technologies adaptées, assure une gestion agile et efficace du projet, tout en garantissant une livraison continue et de qualité des fonctionnalités.

1.3.1 Choix de la méthodologie

Le choix de la méthodologie agile pour ce projet a permis d'assurer une gestion flexible, réactive et collaborative. En complément de l'approche DevOps, cette méthodologie a permis de gérer efficacement les itérations de développement, les tests, et les déploiements continus, tout en garantissant la qualité du produit. Ce cadre a été particulièrement adapté aux exigences de flexibilité et d'adaptation rapide du projet e-commerce.

1.3.2 Méthodologie de gestion de projet

"L'agilité est une attitude, pas une technique avec des limites. Une attitude n'a pas de limites"

Dans un grand projet, l'adoption d'une méthodologie devient une nécessité pour garantir un bon niveau de qualité et satisfaire les besoins du client dans des délais courts. Pour cela, notre choix s'est porté sur la méthodologie agile SCRUM.

SCRUM est de nos jours l'approche agile la plus plébiscitée par les équipes de développeurs car elle promeut les valeurs du fameux Agile : la collaboration avec le client, l'acceptation du changement, l'interaction avec les personnes et des logiciels opérationnels[1]. Sa force est de se reposer sur des cycles de développements courts, adaptés en permanence, sans jamais perdre de vue l'expérience utilisateur.

Parmi ses bénéfices :

- Une gestion plus souple, plus intelligente du travail, améliorant l'efficacité des équipes,

- Une meilleure visibilité du projet et de son évolution,
- Une communication interne renforcée, et donc une meilleure cohésion d'équipe,
- le partage des savoirs et la favorisation de l'entraide,
- un gain de temps et une meilleure réactivité grâce aux réunions fréquentes et aux insights du client.
- La satisfaction de client. [2]

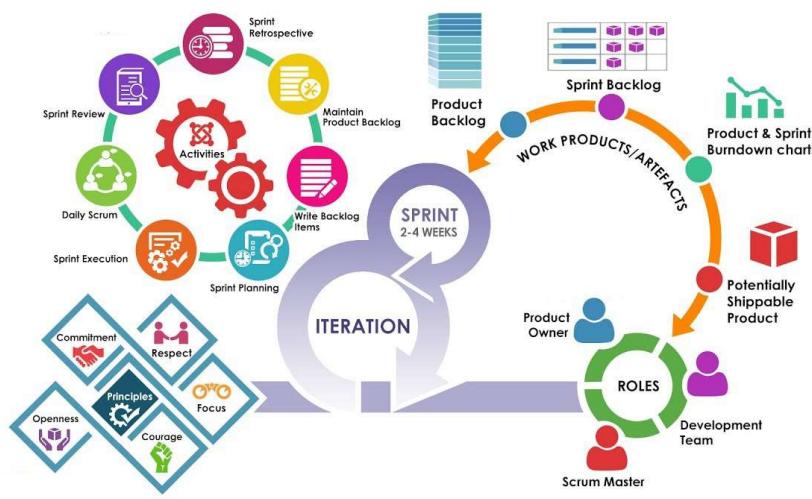


FIGURE 1.1 : SCRUM

1.3.2.1 Les rôles de SCRUM

L'équipe SCRUM se compose de :

- **le product Owner** : Il cherche à comprendre les attentes des clients (le représentant du client) et il est le responsable du Product Backlog.
- **Le scrum master** : C'est le coach de l'équipe.
- **Les développeurs** : Ils doivent être entre 2-9 personnes.

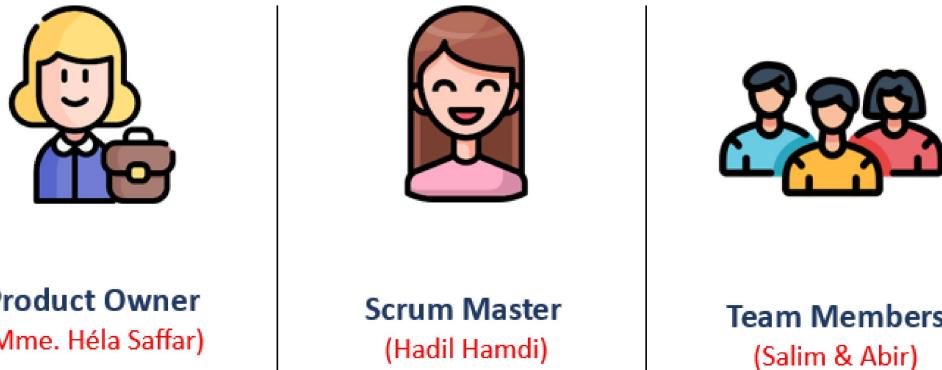


FIGURE 1.2 : Equipe SCRUM

1.3.2.2 Les cérémonies SCRUM

Les cérémonies Scrum sont des réunions qui constituent des rituels clés, rythmant les sprints lors d'un projet agile[5] :

- **Sprint review** : C'est une réunion où l'équipe projet présente aux parties prenantes les différents livrables terminés. Plus qu'une simple présentation, c'est l'occasion de faire une démonstration en conditions réelles afin de s'assurer que le produit réponde aux besoins exprimés par le client.
- **Sprint retrospective** : L'équipe Scrum se réunit pour faire le point sur le sprint qui vient de s'achever, afin d'en tirer des enseignements et devenir encore plus efficace lors du prochain. L'idée est de suivre une démarche d'amélioration continue.[3]

1.3.3 Méthodologie de modélisation et de conception

Pour mettre en œuvre notre projet, nous avons choisi d'utiliser UML (Unified Modeling Language) comme langage de modélisation.

Le langage UML (Unified Modeling Language, ou langage de modélisation unifié) a été pensé pour être un langage de modélisation visuelle commun, et riche sémantiquement et syntaxiquement. Il est destiné à l'architecture, la conception et la mise en œuvre de

systèmes logiciels complexes par leur structure aussi bien que leur comportement.[5]

1.3.4 Environnement Logiciel (langages et technologies)

Dans cette partie, nous allons lister les différents outils, les librairies et les frameworks utilisés dans notre application.

Outils	Description
 Visual Pardigm	Visual Paradigm est un outil de modélisation UML puissant et flexible, utilisé pour créer des diagrammes UML (cas d'utilisation, classes, etc.). [1] Utilisation : Nous avons utilisé Visual Pradigm pour concevoir les diagrammes de modélisation du projet, tels que les cas d'utilisation et les diagrammes de séquence.
 LaTeX	LaTeX est un système de composition de documents de haute qualité utilisé principalement pour les documents scientifiques et techniques.[3] Utilisation : Nous avons utilisé LaTeX pour rédiger la documentation du projet avec des fonctionnalités avancées de mise en page.
 Visual Studio Code	Visual Studio Code est un éditeur de code source léger et extensible, supportant une large gamme de langages de programmation et d'outils de développement.[2] Utilisation : Nous avons utilisé VSCode pour l'écriture et le débogage du code, ainsi que pour l'intégration avec des extensions comme ESLint et Prettier.
 Git	Git est un système de contrôle de version distribué qui permet de suivre les modifications du code source et de collaborer efficacement entre développeurs.[4] Utilisation : Nous avons utilisé Git pour gérer le code source du projet, en assurant un suivi des versions et une collaboration fluide grâce à des plateformes comme GitHub.

TABLEAU 1.1 : Environnement logiciel

1.3.4.1 Langages de Programmation

Le choix des langages de programmation a été fait en fonction des besoins spécifiques du projet, notamment pour garantir une performance optimale, une maintenabilité élevée, et une compatibilité avec les frameworks adoptés. Les langages utilisés sont :

1.3.4.2 Frameworks et Technologies Utilisés

Pour le développement du projet e-commerce, plusieurs frameworks et technologies ont été choisis pour répondre aux exigences de performance, de scalabilité et de facilité de gestion. Ces choix ont été faits en tenant compte de la robustesse des outils, de leur communauté active, ainsi que de leur compatibilité avec les besoins spécifiques du projet.

Voici les principaux frameworks et technologies utilisés :

- **MERN Stack (MongoDB, Express.js, React, Node.js)** La stack MERN a été choisie pour le développement de la plateforme en raison de sa cohérence et de ses nombreux avantages pour la création d'applications web modernes. Elle permet un développement full-stack JavaScript, ce qui facilite la gestion du projet et permet une meilleure collaboration entre les équipes frontend et backend :



FIGURE 1.3 : MERN Stack

- **MongoDB** : MongoDB est une base de données NoSQL qui stocke les données sous forme de documents JSON, offrant une flexibilité accrue pour gérer des données non structurées ou semi-structurées. Sa scalabilité horizontale et sa capacité à gérer des volumes importants de données font de MongoDB un choix idéal pour des applications à fort trafic comme une plateforme e-commerce.



FIGURE 1.4 : Mongodb

Express.js : Express.js est un framework minimaliste pour Node.js qui facilite la gestion des requêtes HTTP, des routages et des middlewares. Il permet de créer rapidement des API robustes pour interagir avec la base de données MongoDB et le frontend React.



FIGURE 1.5 : Express

React : React est un framework JavaScript largement utilisé pour la création d’interfaces utilisateur dynamiques et performantes. Il permet de développer des composants réutilisables et d’améliorer l’expérience utilisateur grâce à des mises à jour d’interface rapides et efficaces via son système de gestion du DOM virtuel.

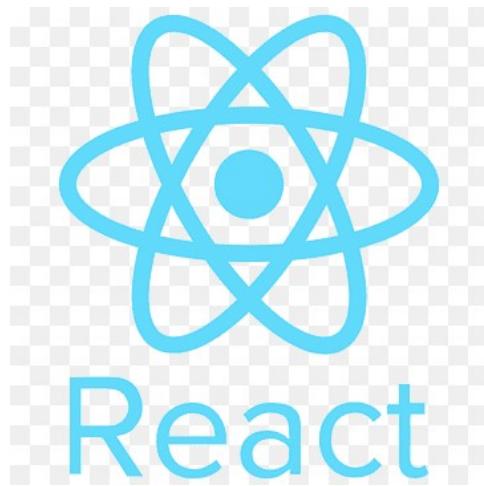


FIGURE 1.6 : React

Node.js : Node.js est un environnement d'exécution JavaScript côté serveur, utilisé pour la gestion des requêtes et le traitement des données en temps réel. Sa capacité à gérer de manière asynchrone un grand nombre de connexions simultanées le rend particulièrement adapté aux applications web interactives et à fort trafic.



FIGURE 1.7 : Node

Docker : Docker a été utilisé pour containeriser les services du backend et du frontend, permettant une gestion simplifiée des environnements de développement, de test et de production. L'utilisation de Docker garantit la portabilité de l'application entre les différentes machines, ainsi qu'une gestion centralisée des dépendances pour chaque

service.



FIGURE 1.8 : Docker

Docker Compose : Docker Compose est un outil permettant de définir et de gérer des applications multi-conteneurs. Il a été utilisé pour orchestrer les services backend (Node.js + MongoDB) et frontend (React), simplifiant ainsi le déploiement de l'application sur différentes plateformes tout en garantissant une gestion cohérente des environnements.

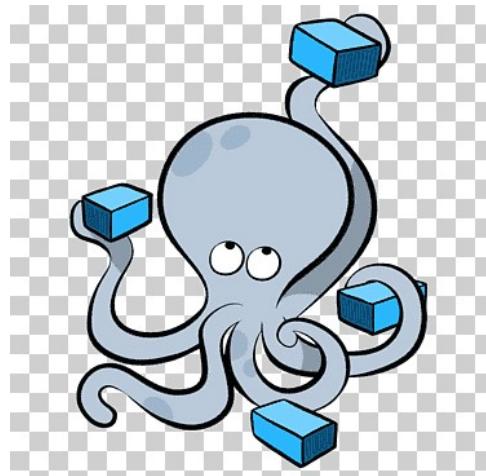


FIGURE 1.9 : Docker Compose

GitHub Actions : GitHub Actions a été utilisé pour mettre en place un pipeline d'intégration et de déploiement continu (CI/CD). Ce service permet d'automatiser les tests, les builds et les déploiements à chaque modification du code, garantissant ainsi une livraison continue et la détection rapide des erreurs.



FIGURE 1.10 : GitHub Action

1.3.4.3 Conclusion

À travers ce projet, l'intégration des technologies modernes telles que la stack MERN, Docker, Jenkins et GitHub Actions illustre une approche DevOps aboutie, centrée sur l'agilité, la performance et l'automatisation. Cette solution technique permet non seulement de construire une application e-commerce performante et évolutive, mais aussi d'assurer une chaîne de déploiement continue, stable et fiable. En combinant développement et opérations dans un flux unifié, le projet démontre comment les pratiques DevOps transforment profondément la manière dont les applications web sont conçues, testées, déployées et maintenues. Cette démarche place la plateforme dans une logique de croissance durable et d'adaptabilité face aux évolutions rapides du marché numérique.

1.4 Conception Architecturale

1.4.1 Architecture Logique

Pour répondre aux besoins du projet et garantir une séparation claire des responsabilités, nous avons opté pour une architecture en trois couches (3-tiers). Cette architecture permet de structurer l'application de manière modulaire, assurant ainsi une meilleure maintenabilité, extensibilité, et gestion des données.

- **Couche de présentation des données :** Responsable de l'interface utilisateur et de la communication avec l'utilisateur final, cette couche prend en charge l'affichage des données et les interactions via des composants graphiques.
- **Couche métier :** Elle contient la logique métier de l'application, gère le traitement des données, et assure la coordination entre la couche de présentation et celle des données.
- **Couche d'accès aux données :** Cette couche est dédiée à la gestion des données. Elle communique directement avec la base de données pour effectuer des opérations comme l'insertion, la modification, ou la récupération des données.



FIGURE 1.11 : Architecture 3-tiers de l'application

Conclusion

La présentation du cadre du projet nous a fourni une base solide sur laquelle nous avons construit le plan d'action et avancé avec confiance vers la réalisation des objectifs.

CHAPITRE 2

ÉTUDE PRÉALABLE

Plan

1	Spécification des besoins	16
2	Identification des besoins fonctionnels	16
3	Backlog Produit Priorisé	19
4	Planification des Sprints	20

Introduction

Avant de procéder à la réalisation du projet, une étude préalable a été réalisée pour analyser les besoins fonctionnels et techniques, choisir les technologies adaptées et évaluer les outils DevOps nécessaires. Cette étape est essentielle pour garantir que la solution proposée soit cohérente, performante et durable.

2.1 Spécification des besoins

2.1.1 Identification des acteurs

Un acteur est une entité externe qui interagit sur le système de façon directe afin de réaliser certains tâches .

Dans notre projet, nous avons deux principaux acteurs :

- Utilisateur (Client)
- Administrateur
- Développeur/Opérateur DevOps

Chaque acteur interagit avec la plateforme selon des rôles bien définis, influençant le comportement et les exigences fonctionnelles du système.

2.2 Identification des besoins fonctionnels

Acteurs	Fonctionnalités

Administrateur	<ul style="list-style-type: none"> — Gérer les utilisateurs (création, modification, suppression) — Gérer les produits (ajout, modification, suppression) — Gérer les catégories de produits — Gérer les commandes (suivi et mise à jour du statut) — Accéder aux tableaux de bord et aux statistiques de ventes — Modérer les avis des clients
Client	<ul style="list-style-type: none"> — Créer un compte client — Se connecter à la plateforme — Rechercher des produits — Consulter les fiches produits — Ajouter des produits au panier — Passer une commande et effectuer le paiement — Suivre ses commandes et consulter l'historique — Modifier ses informations personnelles — Laisser des avis sur les produits
Système (Automatisation DevOps)	<ul style="list-style-type: none"> — Déploiement automatique de l'application via GitHub Actions — Construction et orchestration des conteneurs avec Docker — Déploiement continu automatisé avec Jenkins — Surveillance de l'état des services (containers, APIs)

TABLEAU 2.1 : Besoins fonctionnels du projet e-commerce MERN

2.2.1 Identification des besoins non fonctionnels

En complément des besoins fonctionnels, il est essentiel d'identifier les besoins non fonctionnels du projet afin de garantir la performance, la sécurité et la fiabilité de la plateforme e-commerce. Ces besoins visent à assurer la qualité globale du système au-delà des simples fonctionnalités offertes.

- **Scalabilité** : Le système doit être capable de gérer une augmentation du nombre d'utilisateurs, de produits et de transactions sans perte de performance, grâce notamment à l'utilisation de conteneurs Docker et de mécanismes d'orchestration.
- **Performance** : L'application doit offrir des temps de réponse rapides pour garantir une expérience utilisateur fluide, tant au niveau de la navigation que lors des transactions.
- **Sécurité** : La plateforme doit protéger les données des utilisateurs (authentification sécurisée, protection des données personnelles, transactions sécurisées).
- **Fiabilité** : Les services doivent être disponibles en permanence, avec des mécanismes de monitoring et d'alerte pour détecter rapidement toute anomalie (via Jenkins, GitHub Actions et autres outils de surveillance).
- **Portabilité** : Grâce à l'utilisation de Docker, le projet doit être facilement déployable sur différents environnements (local, cloud, serveur dédié).
- **Automatisation** : Le processus de déploiement doit être automatisé pour limiter les erreurs humaines et accélérer les mises à jour, en utilisant des pipelines CI/CD avec GitHub Actions et Jenkins.
- **Maintenabilité** : Le code et l'infrastructure doivent être conçus de manière à faciliter les évolutions, les corrections de bugs et l'ajout de nouvelles fonctionnalités.

2.3 Backlog Produit Priorisé

Le Backlog Produit priorisé représente la liste structurée et classée par ordre d'importance des fonctionnalités à développer pour la plateforme e-commerce. Chaque élément (ou "user story") décrit une fonctionnalité attendue par un acteur, associée à un niveau de priorité et à une estimation de l'effort requis.

ID	User Story	Priorité	Effort (pts)
US01	En tant que client, je veux créer un compte pour accéder à la boutique.	Haute	3
US02	En tant que client, je veux me connecter pour passer une commande.	Haute	2
US03	En tant que client, je veux consulter les produits disponibles.	Haute	2
US04	En tant que client, je veux ajouter des produits à mon panier.	Moyenne	3
US05	En tant que client, je veux effectuer un paiement sécurisé pour finaliser ma commande.	Haute	5
US06	En tant qu'administrateur, je veux ajouter et modifier les produits du catalogue.	Haute	4
US07	En tant qu'administrateur, je veux supprimer des produits obsolètes.	Moyenne	2
US08	En tant que client, je veux consulter l'historique de mes commandes.	Moyenne	3
US09	En tant que système, je veux contenir l'application avec Docker pour faciliter la portabilité.	Haute	5

US10	En tant que système, je veux automatiser le déploiement via GitHub Actions.	Haute	5
US11	En tant que développeur, je veux configurer un pipeline CI/CD avec Jenkins pour automatiser les processus.	Moyenne	4
US12	En tant qu'administrateur, je veux consulter les statistiques de vente.	Basse	3

TABLEAU 2.2 : Backlog Produit Priorisé de la plateforme e-commerce

2.4 Planification des Sprints

Dans le cadre de ce projet, nous avons appliqué les principes de la méthodologie Scrum en divisant le développement en trois Sprints. Chaque Sprint correspond à une étape clé du projet, avec des objectifs précis, une durée définie et des livrables concrets.

Sprint	Objectifs	Durée
Sprint 1 – Développement de l'application	<ul style="list-style-type: none">— Création de l'interface utilisateur (React)— Mise en place de l'API backend avec Node.js et Express— Intégration de la base de données MongoDB— Authentification utilisateur (connexion / inscription)— Gestion des produits (CRUD)	4 semaines

Sprint 2 – Conteneurisation avec Docker	<ul style="list-style-type: none"> — Création des Dockerfiles pour le frontend, le backend et la base de données — Construction des images Docker — Utilisation de Docker Compose pour le développement multi-conteneurs — Tests de déploiement local avec Docker 	2 semaine
Sprint 3 – Mise en place CI/CD	<ul style="list-style-type: none"> — Création d'un pipeline CI/CD avec GitHub Actions — Intégration de Jenkins pour la construction et le déploiement automatisé — Automatisation des tests unitaires à chaque push — Déploiement de l'application sur un serveur distant 	2 semaine

TABLEAU 2.3 : Planification des Sprints selon Scrum

2.4.0.1 Conclusion

Ce chapitre nous a permis de définir clairement le périmètre de ce projet et d'en affiner la vision globale. Nous avons présenté l'architecture de la nouvelle infrastructure, identifié les besoins fonctionnels et non fonctionnels, ainsi que les différents acteurs du système. Le backlog produit a également été élaboré et priorisé. Enfin, nous avons planifié l'exécution du projet en le découplant en trois sprints principaux. Les chapitres suivants détailleront la mise en œuvre de cette nouvelle infrastructure.

SPRINT 1 : DÉVELOPPEMENT DE L'APPLICATION

Plan

1	Mise en place de l'environnement de développement	24
2	Planification et objectifs du Sprint 1	24
3	Backlog du Sprint 1	25
4	Analyse	26
5	Réalisation	29
6	Rétrospective du Sprint	33

Introduction

Ce premier sprint a pour objectif principal de poser les bases fonctionnelles de l'application e-commerce. Il s'agit d'implémenter les principales fonctionnalités côté frontend et backend en utilisant la stack MERN (MongoDB, Express.js, React, Node.js).

3.1 Mise en place de l'environnement de développement

Sprint 1 : Développement de l'application

Pour garantir un développement fluide et cohérent, nous avons commencé par la configuration des environnements de développement :

- Initialisation d'un dépôt Git pour la gestion de version.
- Création du projet backend avec Node.js et Express.
- Création du projet frontend avec React.
- Connexion de l'API à MongoDB via Mongoose.

3.2 Planification et objectifs du Sprint 1

La planification du Sprint 1, intitulé « *Fondation fonctionnelle* », a été orientée vers la mise en œuvre des fonctionnalités de base permettant aux utilisateurs d'interagir avec l'application. En s'appuyant sur les priorités définies dans le Product Backlog, ce sprint visait à poser les fondations fonctionnelles nécessaires au bon fonctionnement de la plateforme e-commerce.

Les objectifs principaux fixés étaient les suivants :

- Permettre la création de comptes utilisateurs.
- Implémenter le mécanisme d'authentification (inscription et connexion).
- Offrir aux utilisateurs un accès aux fonctionnalités principales de la plateforme, telles que la navigation des produits, l'ajout au panier et la gestion du profil.

- Poser les bases pour une structure modulaire et évolutive de l'application.

Ce sprint marque une étape clé dans la réalisation du projet, en assurant les premières interactions entre l'utilisateur et le système, tout en garantissant une architecture solide pour les développements à venir.

3.3 Backlog du Sprint 1

Le backlog du Sprint 1 regroupe l'ensemble des *user stories* sélectionnées comme prioritaires pour construire la première version fonctionnelle de l'application. Ces tâches couvrent les fonctionnalités essentielles permettant aux utilisateurs de s'enregistrer, de se connecter et d'interagir avec l'interface principale de la plateforme e-commerce.

Ce backlog comprend :

- **US01 – Enregistrement utilisateur** : En tant qu'utilisateur, je veux pouvoir créer un compte afin d'accéder à la plateforme.
- **US02 – Authentification** : En tant qu'utilisateur, je veux pouvoir me connecter à mon compte afin de gérer mes activités sur la plateforme.
- **US03 – Navigation produit** : En tant qu'utilisateur, je veux parcourir les produits disponibles afin d'explorer les articles proposés.
- **US04 – Ajout au panier** : En tant qu'utilisateur, je veux ajouter des produits à mon panier afin de les acheter plus tard.
- **US05 – Gestion du profil** : En tant qu'utilisateur, je veux consulter et modifier mes informations personnelles.

Ces user stories ont été choisies afin de garantir une base fonctionnelle cohérente et utilisable dès la fin du premier sprint, tout en facilitant l'extension de nouvelles fonctionnalités dans les sprints suivants.

3.4 Analyse

3.4.1 Représentation du diagramme de cas d'utilisation

Le diagramme de cas d'utilisation pour le **Sprint 1** illustre les interactions entre les différents acteurs du système, notamment l'utilisateur (client) de la plateforme e-commerce. Il met en évidence les principales fonctionnalités mises en œuvre au cours de ce sprint, comme la création de compte, la connexion, la consultation des produits, ou encore la gestion du panier.

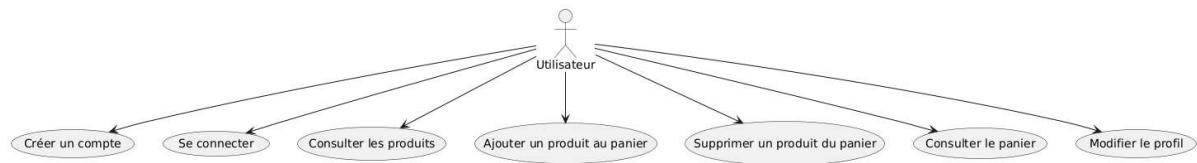


FIGURE 3.1 : Diagramme de cas d'utilisation

3.4.2 Conception

La phase de conception constitue une étape essentielle dans le cycle de développement logiciel. Elle permet de traduire les besoins exprimés en une architecture technique claire et cohérente, facilitant ainsi le passage à l'implémentation. Elle permet également de garantir la maintenabilité, l'évolutivité et la robustesse du système.

3.4.2.1 Diagramme d'activité

Le diagramme d'activité permet de visualiser le déroulement logique d'un processus métier ou fonctionnel. Il décrit les différentes étapes d'une activité ainsi que les transitions entre elles, en tenant compte des conditions, des choix possibles, et des actions qui peuvent être exécutées en parallèle. Dans le cadre de ce projet, ce diagramme est utilisé pour représenter les étapes de navigation et d'interaction d'un utilisateur sur la plateforme e-commerce.

