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

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]

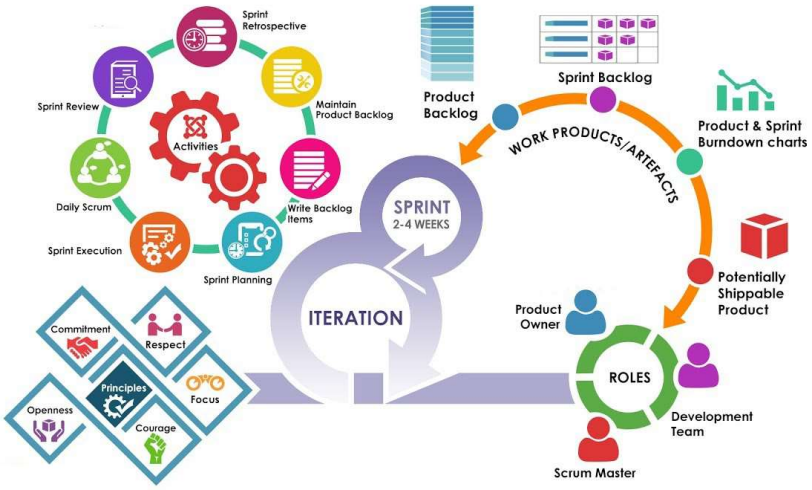


Figure 1.1 : SCRUM

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.

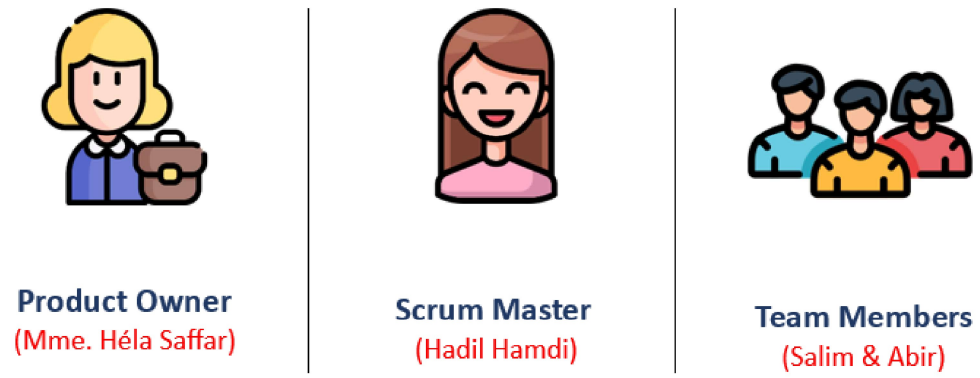


Figure 1.2 : Equipe SCRUM

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.

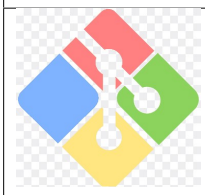
Outils	Description
 Visual Pardigm	**Visual Paradigm** est un outil de modélisation UML puissant et flexible, utilisé pour créer des diagrammes UML (cas d'utilisation, classes, etc.). [1] **Utilisation :** Nous avons utilisé Visual Pradigm pour concevoir les diagrammes de modélisation du projet, tels que les cas d'utilisation et les diagrammes de séquence.
LaTeX LaTeX	**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.

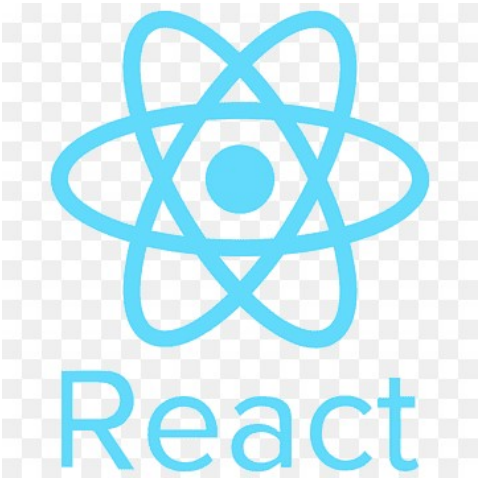


Figure 1.6 : React

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.

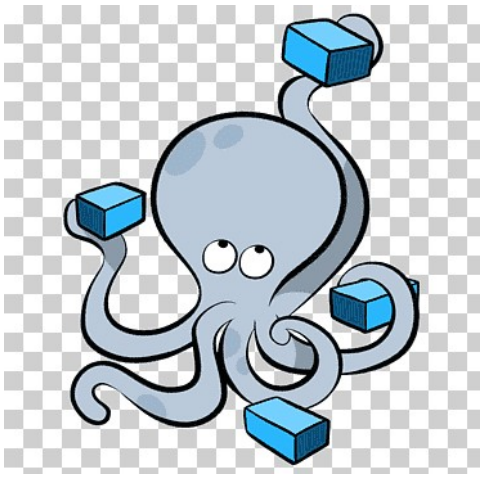


Figure 1.9 : Docker Compose

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.

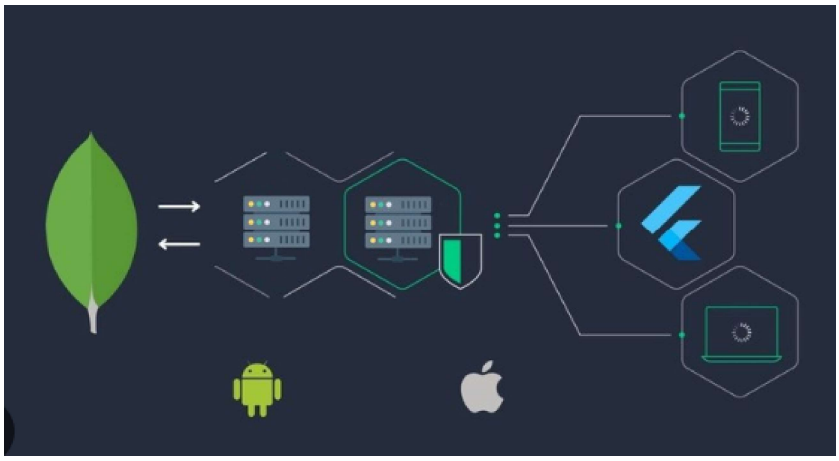


Figure 1.11 : Architecture 3-tiers de l'application

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.