République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique et de la Technologie de l'Information et de la Communication.

Ecole Supérieure des Sciences Appliquées et de la Technologie Privée de Gabès

Projet de fin d'études pour l'obtention du diplôme de licence

Parcours : Systèmes d'information d'entreprises (BIS).

Titre de projet :

Conception et Réalisation d'une plateforme E-learning

Réalise par :

Ahmed Guiza - Hachem moussa

Lieu de stage :

Société laghazala du désert formations et services

Encadrée par :

Mme. Nadia Ghaibi

DÉDICACE

REMERCIEMENTS

الخلاصة

RÉSUME

ABSTRACT

TABLE DES MATIERES

TABLE DES FIGURES

INTRODUCTION GÉNÉRALE

Dans un contexte marqué par la digitalisation accélérée des secteurs clés de la société, l'éducation connaît une transformation profonde, portée par l'émergence des technologies de l'information et de la communication (TIC), cette transformation s'est accélérée sous l'effet de crises mondiales, comme la pandémie de COVID-19, révélant autant les potentialités du numérique que les lacunes des systèmes éducatifs traditionnels.

Les plateformes d'apprentissage en ligne, ou *E-learning*, s'imposent aujourd'hui comme une réponse aux défis de l'accessibilité, de la flexibilité et de la personnalisation de l'éducation. Ce projet de fin d'études s'inscrit dans cette dynamique en proposant le développement d'une plateforme *E-learning* innovante, conçue pour répondre aux besoins des apprenants.

L'objectif principal de ce travail est de créer un environnement d'apprentissage interactif, intuitif et adaptatif, capable de transcender les limites géographiques et temporelles des méthodes pédagogiques traditionnelles. La plateforme vise à intégrer des fonctionnalités avancées telles que des cours modulaires, des évaluations automatisées, un suivi personnalisé des progrès. Elle s'adresse à un public large, incluant étudiants et professionnels en reconversion.

Notre solution s'appuie sur une méthodologie structurée qui combine deux approches essentielles. D'une part, nous plaçons l'utilisateur au cœur du processus avec une conception UX/UI soigneusement élaborée. D'autre part, nous exploitons des technologies contemporaines performantes : React.js pour l'interface utilisateur, Node.js pour la logique serveur, et MongoDB comme système de gestion de données.

Pour garantir l'efficacité de notre développement, nous avons adopté une méthode agile caractérisée par des cycles courts et répétés. Cette approche évolutive assure que notre solution reste pertinente et adaptée tout au long de son développement.

CHAPITRE I : ÉTUDE PRÉALABLE

1. INTRODUCTION

Le présent chapitre a pour objectif de poser les bases du projet et de présenter les éléments fondamentaux qui guideront la suite de la réalisation.

Dans un premier temps, il introduit l'organisme d'accueil, en mettant en avant son domaine d'activité. Ensuite, la mission du stage. Pour bien cerner le contexte, une analyse de l'existant est menée. Cette étude se décompose en une présentation et une critique de l'existant, permettant d'orienter la conception de la solution proposée**.**

Afin de structurer le développement de cette nouvelle solution, une méthodologie de développement est choisie. Après une brève introduction aux méthodes classiques et aux contraintes qui y sont liées, le chapitre se concentre sur les méthodes agiles et plus précisément sur **Scrum**.

Enfin, le chapitre se conclut par la présentation du langage de modélisation utilisé pour formaliser les différentes composantes du système à développer.

2. PRÉSENTATION DE L'ORGANISME D'ACCUEIL

Société Laghazala du désert formations et services est une société SARL crée en 2021 et à comme activité principale une école de formation professionnelle et une deuxième activité services informatiques basé à Kébili et à Gabes.

Figure 1 Logo de la société d'accueil

3. MISSION DU STAGE

Concevoir et développer une solution logicielle (Web) destinée à la gestion complète des apprenants dans un centre de formation.

4. ANALYSE D'EXISTANT

Avant de commencer un projet informatique, il est crucial d'analyser ce qui existe déjà. Cela implique de comprendre les fonctionnalités actuelles pour déterminer les lacunes et orienter le choix de la meilleure solution pour l'application à développer.

4.1. ÉTUDE DE L'EXISTANT

4.2. PRÉSENTATION

4.3. CRITIQUE DE L'EXISTANT

5. SOLUTION PROPOSÉE

6. MÉTHODOLOGIE DE DEVELOPPEMENT

6.1. INTRODUCTION

Selon des estimations, plus que 80% des projets qui utilisent des méthodologies classiques qui connaissent des retards et des dépassements budgétaires parce que ces méthodologies visent à prédire la façon dont les procédures devraient se passer selon un planning préétabli.

Notre projet suit le principe de développement itératif basé sur l'écoute du client, c'est pour cette raison que notre choix s'est orienté vers les méthodes agiles de développement et de gestion de projet.

6.2. METHODE AGILE

Les méthodes de développement dites « méthodes agiles » (en anglais Agile Modeling, noté AG) visent à réduire le cycle de vie du logiciel et par conséquent accélérer sa réalisation en développant une version minimale, puis en intégrant les fonctionnalités par un processus itératif basé sur une écoute client et des tests tout au long du cycle de développement.

Cette approche est née face à l'instabilité rapide des technologies et à la difficulté, pour le client, de définir exhaustivement ses besoins dès le lancement du projet. Le terme « agile » souligne précisément la souplesse nécessaire pour intégrer les changements de contexte et les ajustements de spécifications qui surviennent en cours de développement.

En adoptant les méthodes agiles, le client devient réellement acteur de son projet : il peut intervenir dès les premières phases, voir très vite une version opérationnelle de son logiciel et orienter son évolution en continu, en associant les utilisateurs finaux dès le début.

Méthode	Description	Principes clés	Avantages
SCRUM	• Cadre empirique et itératif basé sur des sprints de 1 à 4 semaines.	• transparence, inspection, adaptation. • Rôles : Scrum Master, Product Owner, équipe.	• Feedback régulier via revues de sprint. • Adaptation rapide aux changements de besoins.
KANBAN	• Méthode Lean de gestion de flux visuel avec limitation du WIP.	• Visualisation du flux via tableau Kanban. • Limitation du Work In Progress (WIP) .	•Amélioration continue par petits ajustements. • Flexibilité et transparence du processus.
XP	• Cadre agile axé sur l'excellence technique et les itérations courtes.	• Pair programming, Test-Driven Development (TDD), integration continue.	• Qualité de code élevée grâce aux tests et revues constantes. • Réactivité accrue aux changements des exigences.

Tableau 1Tableau comparatif des méthodes agiles

6.3. MÉTHODE ADOPTEE

Dans cette partie, nous présentons notre méthode de développement adoptée. Notre choix s'est porté sur la méthode **Scrum** car elle répond aux caractéristiques des méthodes agiles définies dans la section précédente.

Nous avons choisi la méthode Scrum pour les raisons suivantes :

- Elle est utilisée pour développer et tester les courtes itérations,

- Elle permet de produire la plus grande valeur métier dans la durée la plus courte,
- Elle permet l'augmentation de la productivité,
- Elle permet d'adapter le logiciel crée suivant l'évolution du projet

6.3.1. METHODE SCRUM

La méthode Scrum est un cadre de travail agile utilisé pour gérer des projets complexes, surtout dans le domaine du développement logiciel. Son objectif est de livrer un produit de qualité en respectant les besoins changeants des clients. Scrum fonctionne par itérations courtes appelées **sprints**.

Les principaux avantages de Scrum sont :

- Sa flexibilité (adaptation aux changements),
- Sa transparence (progrès visibles via un tableau de tâches ou **Scrum Board**),
- Son accent sur l'amélioration continue.

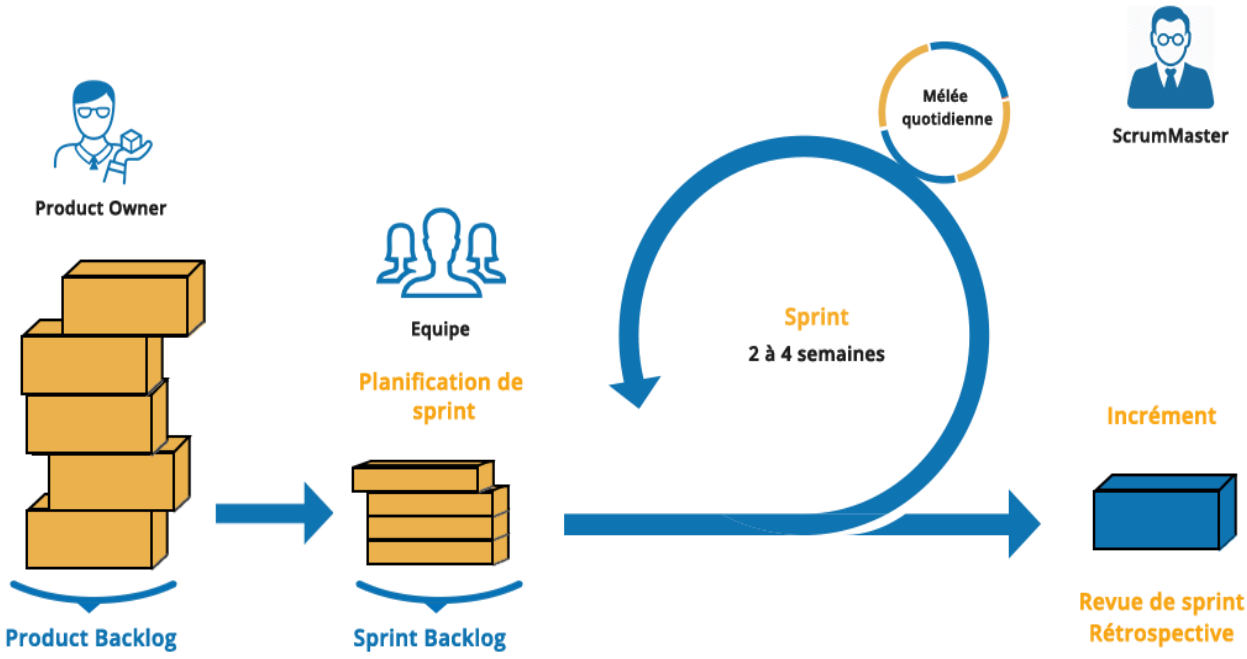


Figure 2 Cycle de vie Scrum

6.3.1.1. ÉQUIPE SCRUM

L'équipe Scrum se compose de trois rôles clés :

- **Product Owner**, qui représente le client du projet et priorise les fonctionnalités.
- **Scrum Master**, qui facilite l'application de la méthode, résout les obstacles et assure également l'organisation des réunions.
- **Équipe de développement**, qui réalise les tâches techniques.

6.3.1.2. BACKLOG DU PRODUIT

Chaque sprint commence par une planification où l'équipe définit les tâches à réaliser, sélectionnées dans une liste priorisée (le Product Backlog).

Les éléments du Backlog produit, appelé aussi des histoires utilisateurs, sont formulés en une ou deux phrases décrivant de manière claire et précise la fonctionnalité désirée par le client.

Généralement, écrit sous la forme suivante : En tant que **X**, je souhaite **Y**, afin de **Z**.

Le Backlog du produit comprend les champs suivants :

- ID : c'est un nombre unique et auto-incrémenté pour chaque histoire.
- Titre : c'est le résumé du user story,
- User Story : c'est une phrase décrivant la fonctionnalité désirée.
- Estimation : c'est l'effort nécessaire à la réalisation d'une histoire utilisateur. Il est estimé par l'équipe et il est fixe dans le temps

6.3.1.3. SPRINT

Le Sprint est une période qui varie entre 2 et 4 semaines au maximum. Au bout de cette période, l'équipe délivre un incrément du produit potentiellement livrable. Une fois la durée est fixée, elle reste constante pendant toute la durée du développement.

6.3.1.4. SCRUM MEETING

Le Scrum Meeting n'est pas une réunion pendant laquelle nous cherchons à résoudre les problèmes, mais uniquement à les identifier et les exprimer.

Le Scrum Master a pour rôle d'apporter des solutions ou de déléguer à un autre membre de l'équipe la résolution des problèmes soulevés durant le Scrum Meeting.

6.3.1.5. REVUE DE SPRINT

À la fin du sprint, tout le monde se réunit pour effectuer la revue du sprint. L'objectif de la revue du sprint est de valider le logiciel qui a été produit pendant le sprint. L'équipe commence par énoncer les items du Backlog de produit qu'elle a réalisés

7. LANGAGE DE MODÉLISATION UML

UML (Unified Modeling Language) est un langage graphique standardisé utilisé pour concevoir, visualiser et documenter des systèmes logiciels. Il propose des diagrammes normalisés (comme les diagrammes de classes, de cas d'utilisation, de séquence ou d'activité) pour représenter la structure, les interactions et les comportements d'un système.

UML facilite la communication entre les acteurs d'un projet et clarifie les exigences techniques dès les phases d'analyse et de conception.

Figure 3 Unified Modeling Language

8. CONCLUSION

Ce chapitre a permis de structurer les fondations du projet en clarifiant son contexte, ses objectifs et ses méthodes. Après avoir présenté l'organisme d'accueil et défini la mission du stage, une analyse critique de l'existant a identifié les lacunes des systèmes actuels, justifiant ainsi la nécessité d'une nouvelle solution.

L'adoption d'une méthodologie agile, spécifiquement Scrum, a été détaillée pour répondre aux défis des projets dynamiques. Cette approche garantit la flexibilité, transparence et livraison progressive de valeur.

Enfin, le choix du langage UML a été justifié pour modéliser visuellement les exigences et les architectures techniques, assurant une communication claire entre les parties prenantes.

En combinant rigueur analytique, agilité méthodologique et outils de modélisation standardisés, ce chapitre établit un cadre solide pour la phase de conception et de développement, qui sera approfondie dans les chapitres suivants.

CHAPITRE 2 : ANALYSE ET CONCEPTION GÉNERALE

1. INTRODUCTION

La planification joue un rôle essentiel dans la réalisation d'un projet et sert de fondement à chaque application. Ce chapitre se concentre sur la planification du backlog produit, qui vise à clarifier les besoins à satisfaire de manière précise.

2. SPECIFICATION DES BESOINS

2.1. IDENTIFICATIONS DES ACTEURS

Un acteur est une entité externe au système. Il représente une personne ou un autre système informatique qui attend un ou plusieurs services ouverts par une interface d'accès. Par ailleurs, notre application va intervenir les différents acteurs suivants :

- **Administrateur :**
  - Il possède le droit de gérer, archiver et désarchiver les formations.
  - Valider les demandes d'inscription aux formations.
  - Assigner un formateur d'après la liste des utilisateurs.
  - Il peut valider les utilisateurs.
- **Formateur :**
  - Il prend en charge de gérer les modules, ses vidéos, ses tests.
- **Internaute :**
  - Il peut inscrire à la plateforme.
  - Consulter les formations disponibles.
- **Apprenant :**
  - Il peut inscrire à une formation.
  - Consulter le cours passer des tests et obtient un certificat.

2.2. BESOINS FONCTIONNELS

L'analyse fonctionnelle est une approche qui permet d'identifier et de définir les fonctions qu'un produit doit offrir afin de répondre aux besoins des utilisateurs. Elle vise à prendre en compte les attentes spécifiques de chaque utilisateur vis-à-vis du système conçu.

Ainsi, nous présentons l'ensemble des besoins fonctionnels à implémenter dans notre projet, en les répartissant selon les différents modules de notre application :

- **Gestion des utilisateurs :**
  - Un internaute peut s'inscrire et consulter les formations disponibles.
  - Un administrateur valide les inscriptions des utilisateurs.
  - Un utilisateur validé peut s'inscrire à une formation.
  - L'inscription à une formation doit être validée par l'administrateur.
- **Gestion des formations :**
  - L'administrateur peut ajouter une formation.
  - Un formateur peut ajouter du contenu à une formation.
  - Une formation est composée de plusieurs modules.
  - Chaque module contient des vidéos et un test.
  - Un apprenant reçoit un résultat après avoir terminé un test.

2.3. BESOINS NON FONCTIONNELS

- **Sécurité et authentification :**
  - Utilisation de JWT pour l'authentification des utilisateurs.
  - Sécurisation des routes selon les rôles (admin, formateur, apprenant).
- **Performance :**
  - Optimisation des requêtes pour de meilleures performances.

- Mise en cache des données fréquemment consultées.

- **Expérience utilisateur (UX/UI) :**
  - Interface claire et intuitive pour la navigation.
  - Design responsive pour s'adapter aux différents écrans.
- **Disponibilité et fiabilité :**
  - Gestion des erreurs et affichage de messages clairs en cas de problème.
  - Sauvegarde et récupération des données en cas de panne.
- **Extensibilité :**
  - Possibilité d'ajouter de nouveaux types de contenus (PDF, quiz).
  - Système flexible permettant d'ajouter des fonctionnalités futures sans refonte complète.

3. PILOTAGE DU PROJET AVEC SCRUM

3.1. LES ROLES SCRUM

Pour mener à bien un projet en adoptant la méthodologie SCRUM, il est essentiel d'identifier en premier lieu l'équipe impliquée. La composition de notre équipe est illustrée dans la figure ci-dessous.

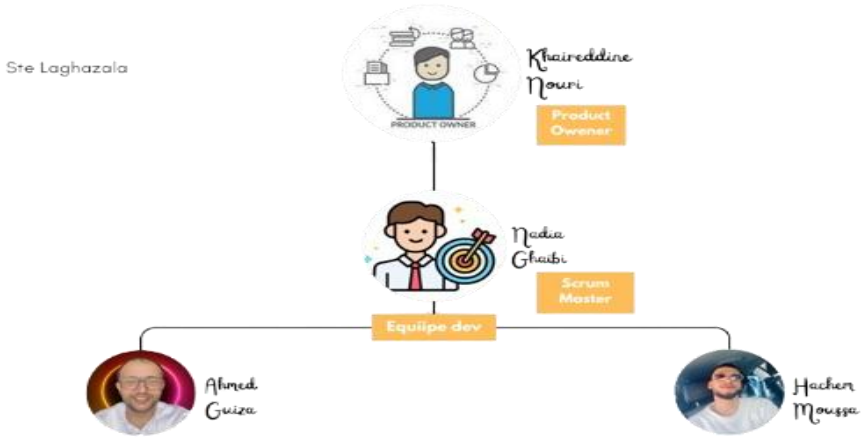


Figure 4 Equipe scrum

- ✓ Product owner (PO) : Mr. Nouri Khaireddine, représente le client du projet.
- ✓ Scrum Master (SM) : Mme. Ghaibi Nadia, notre encadrante.
- ✓ Equipe de développement :
  - Guiza Ahmed
  - Moussa Hachem

3.2. PLANIFICATION DU BACKLOG

La planification du backlog est une étape essentielle de la méthodologie Scrum. Cette section décrit en détail le processus de planification, incluant la création du diagramme de Gantt ainsi que l'organisation des sprints.

3.2.1. DIAGRAMME DE GANTT

Le diagramme de Gantt fournit une vision chronologique du projet, mettant en avant ses différentes phases, les jalons clés et les dépendances entre les tâches. Il constitue un outil visuel de référence pour la répartition du travail et l'optimisation des ressources tout au long du projet.

La figure ci-dessous présente le diagramme de Gantt de notre projet, offrant une représentation claire du calendrier prévu afin de faciliter la gestion du temps et des ressources.

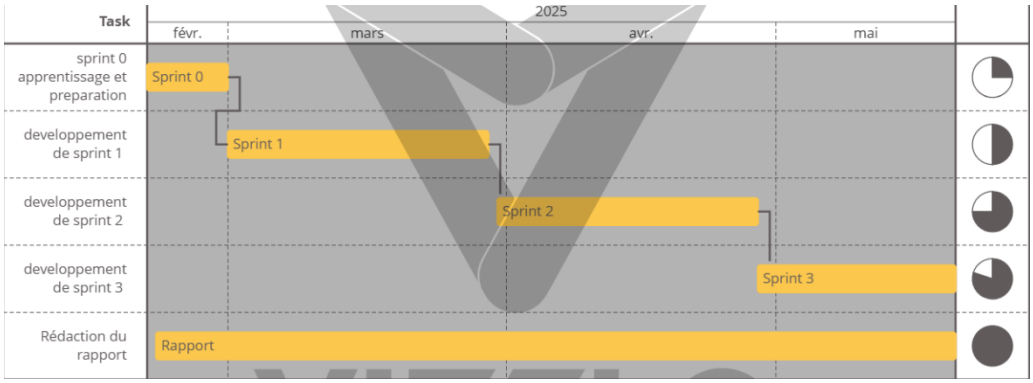


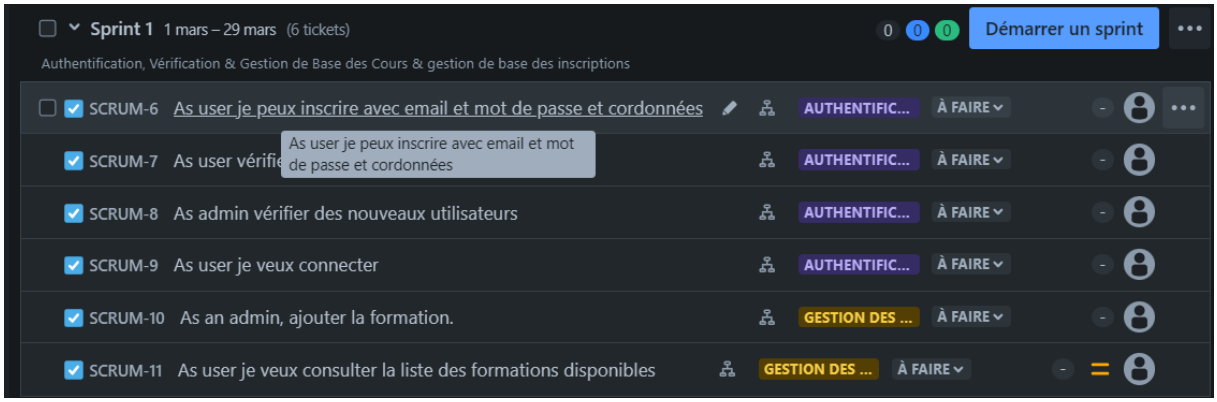
Figure 5 Diagramme de Gantt de notre projet

3.2.2. PLANIFICATION DES SPRINTS

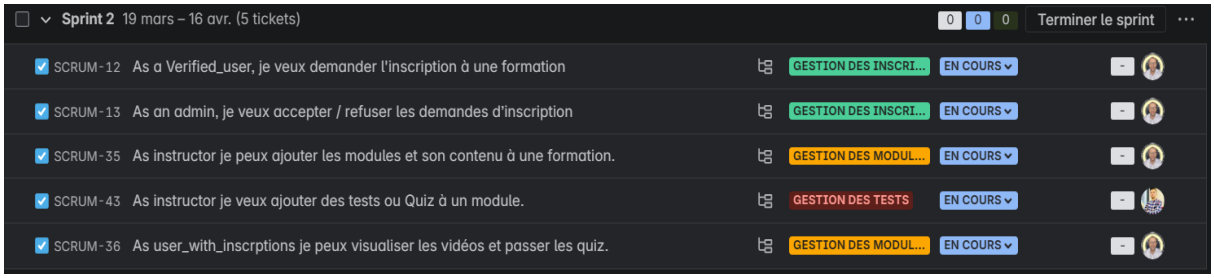
❖ **Sprint 0**

- Apprentissage des technologies (JavaScript, React, Jira)
- Préparation de l'environnement de développement.
- Conception du logo.
- Préparation des maquettes figma.

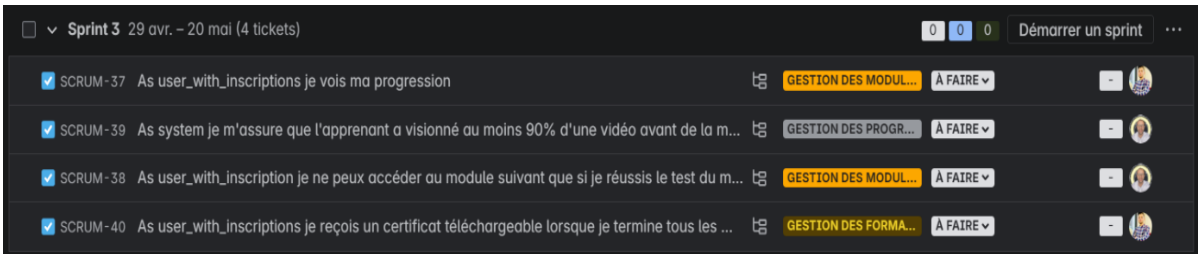
❖ **Sprint 1**



❖ **Sprint 2**



❖ **Sprint 3**



4. ANALYSE DE BESOINS

4.1. DIAGRAMME DE CAS D'UTILISATION GLOBALE

Nous donnons dans la figure ci-dessous une vue globale concernant le comportement fonctionnel du système. Ce diagramme permet de représenter les interactions entre les acteurs et les cas d'utilisation du système.

Figure 6 Diagramme de cas d'utilisation général

4.2 ARCHITECTURE LOGIQUE

L'architecture MERN suit généralement une approche RESTful API, où le backend (Node.js et Express.js) expose des endpoints permettant au frontend (React) d'interagir avec la base de données MongoDB.

Elle peut également être structurée selon le modèle MVC (Model-View-Controller), en séparant la logique métier (Modèle), la gestion des requêtes (Contrôleur) et l'affichage (Vue).

- MongoDB sert de base de données NoSQL pour stocker les données.
- Express.js gère les requêtes et les réponses côté serveur.
- React prend en charge l'interface.
- Node.js assure l'exécution du code backend en environnement JavaScript.

Enfin, cette organisation facilite la maintenabilité, la scalabilité et la communication entre le frontend et le backend.

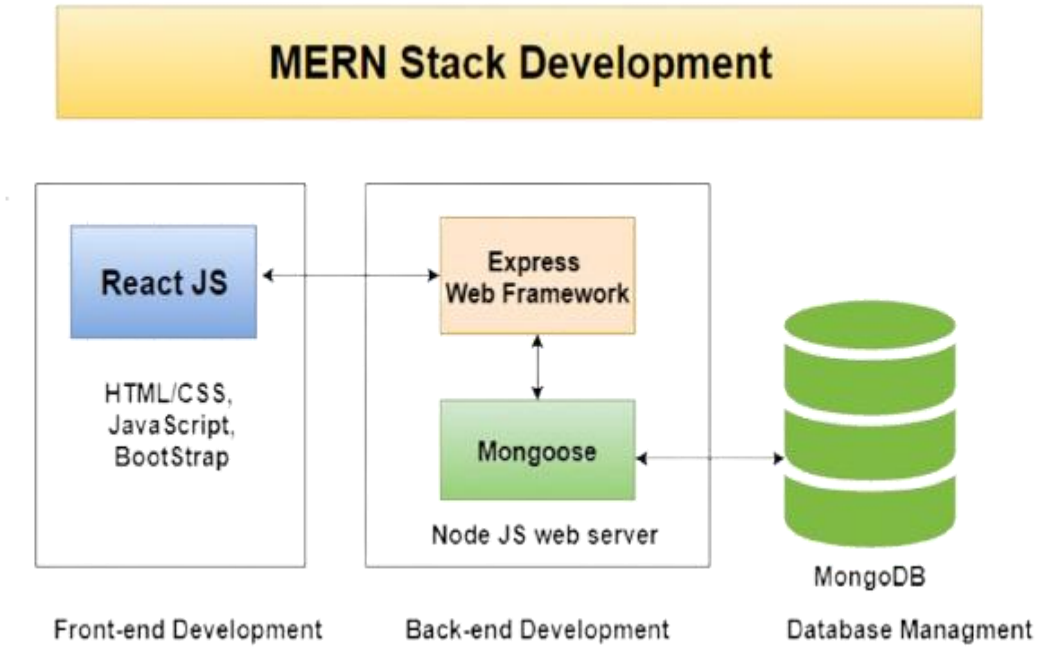


Figure 7 Architecture MERN

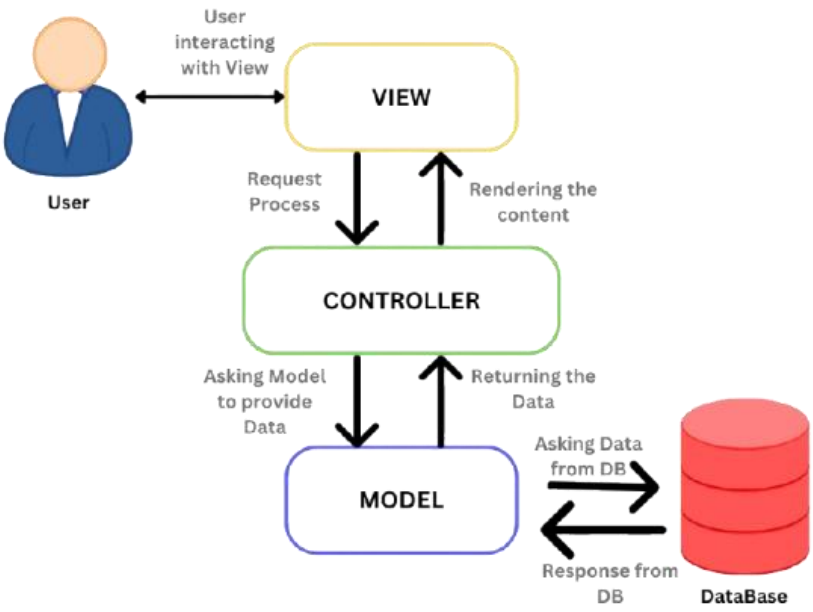


Figure 8 Modèle-Vue-Contrôleur REST API

4.3. DIAGRAMME DE CLASSES ENTITES

Passons maintenant à présenter le diagramme de classes entités de notre application, qui offre un aperçu détaillé de la structure de nos données et des interactions entre les entités

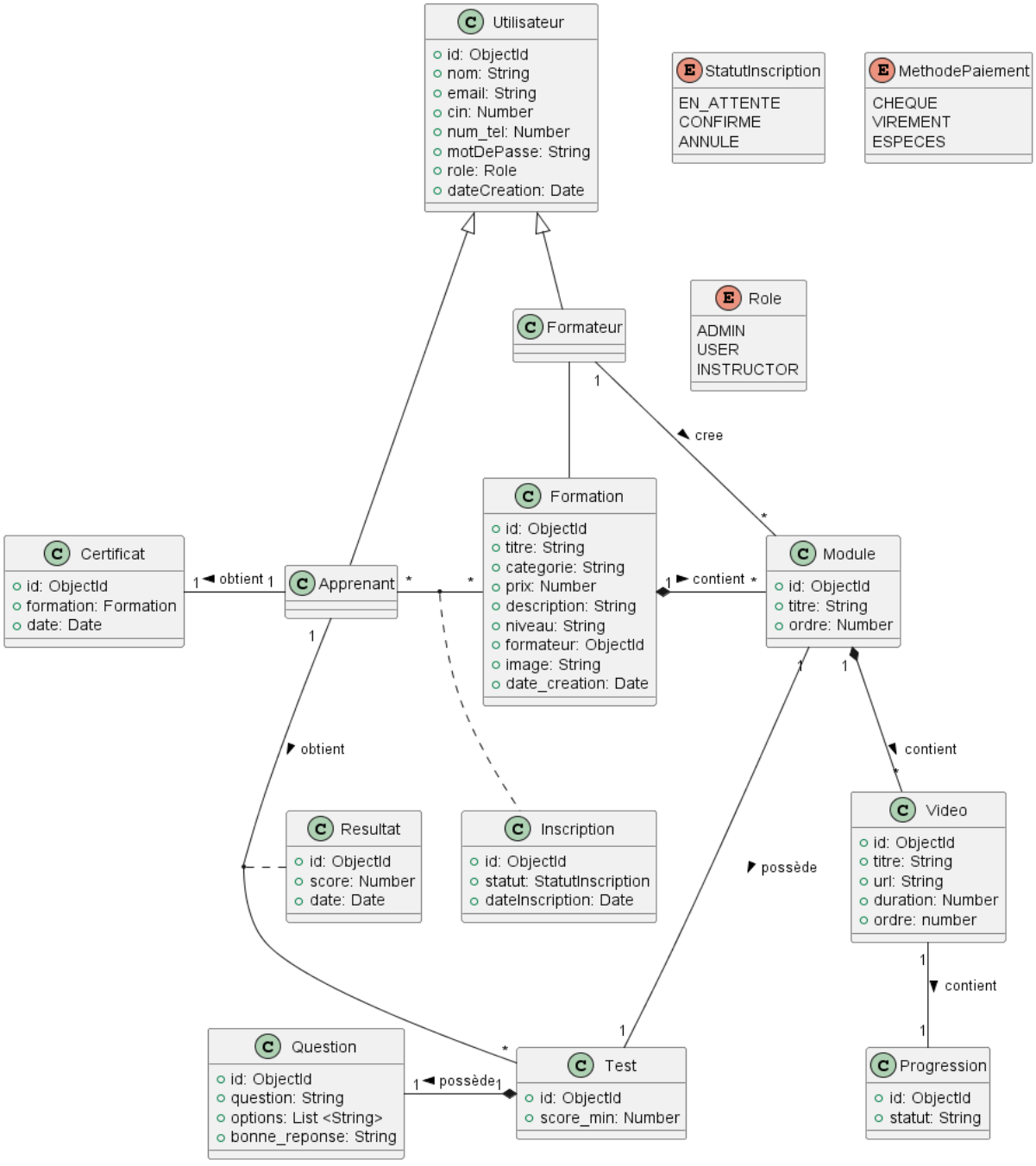


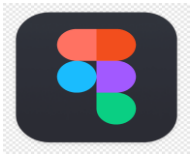
Figure 9 Diagramme de classe de notre projet

5. ENVIRONNEMENT DE DEVELOPPEMENT

Avant de commencer l'implémentation de notre projet, nous avons configuré notre environnement de développement en réunissant l'ensemble des outils logiciels, langages et frameworks nécessaires à sa réalisation.

Visual Studio Code : Éditeur de code polyvalent de Microsoft, avec débogage intégré, extensions personnalisables et prise en charge de multiples langages.

Postman : Outil de test et documentation d'API, permettant d'envoyer des requêtes HTTP, automatiser des tests et simuler des endpoints.



Figma : Outil de design collaboratif pour créer des interfaces utilisateur, des prototypes et des systèmes de design. Accessible via navigateur et en temps réel.

Git : Système de contrôle de version décentralisé pour suivre les modifications de code. Permet le travail en parallèle via des branches et fusion des changements

GitHub : Plateforme d'hébergement de projets Git avec gestion de collaboration (issues, pull requests). Utilisé pour l'open source et le CI/CD.

NodeJs : Environnement d'exécution JavaScript côté serveur basé sur le moteur V8. Permet de construire des applications réseau scalables et asynchrones.

ExpressJs : Framework web minimaliste pour Node.js, simplifiant la création d'APIs REST et d'applications avec routage et middleware.

PlantUml : Outil pour générer des diagrammes UML (classes, séquences) à partir de texte. Intégrable dans des docs, wikis ou IDE pour la modélisation visuelle.

JavaScript : Langage de script côté client et serveur (via Node.js) pour créer des sites interactifs. Essentiel pour le développement web moderne avec HTML/CSS.

Gimp : Logiciel libre et gratuit de retouche photo et de création graphique, offrant des outils avancés (calques, filtres). Alternative puissante à Photoshop pour l'édition d'images et la création artistique.

6. CONCLUSION

CHAPITRE 3 : CONCEPTION DETAILLEE ET REALISATION DES SPRINTS

1. INTRODUCTION

Dans ce chapitre nous présentons l'articulation entre la phase de conception technique et la mise en œuvre agile via des sprints. Ce chapitre détaille les itérations de développement, les décisions clés et les livrables intermédiaires validant les fonctionnalités prioritaires du projet.

2. SPRINT 0 PREPARATION

L'objectif de ce sprint est de de préparer l'environnement de travail et d'étudier la solution ainsi que les technologies à utiliser.

2.3. BACKLOG DU SPRINT

Objectif : Acquérir les compétences nécessaires et configurer l'environnement de travail pour démarrer le développement.

2.3.1. APPRENTISSAGE DES TECHNOLOGIES :

- **JavaScript :**
  - Maîtrise des concepts de base (ES6+, manipulation du DOM, Promesses, async/await).
  - Compréhension des frameworks frontend (React) et des outils de gestion de packages (npm).
- **React :**
  - Apprentissage des composants fonctionnels, hooks (useState, useEffect), routage (React Router).
  - Création d'interfaces dynamiques avec JSX et intégration d'API.

- Apprendre à utiliser redux pour la gestion de l'état.

- **Jira :**
  - Prise en main de la plateforme pour la gestion agile (création de tickets, suivi des sprints, tableaux, chronologie).
  - Rédaction des user stories et estimation des points d'effort.

2.3.2. PREPARATION DE L'ENVIRONNEMENT DE DEVELOPPEMENT.

- **Outils installés** :
  - VS Code : Configuration des extensions (ESLint, Prettier, React Snippets).
  - Node.js : Installation de la LTS (V22.x) et vérification avec node -v et npm -v.
  - Git : Initialisation du dépôt local et liaison avec GitHub.
  - Postman : Création d'un workspace pour tester les API.
  - MongoDB : Configuration d'un cluster Atlas et connexion à l'application via Mongoose.
- **Structure du projet** :
  - Initialisation du projet React.
  - Configuration du backend Node.js (dossiers server, models, routes, controller, middleware).

2.3.3. CONCEPTION DU LOGO.

- Gimp : Conception du logo.

Figure 10 Logo de notre plateforme

2.3.4. PREPARATION DES MAQUETTES FIGMA.

- Figma : Conception des maquettes d'interface.

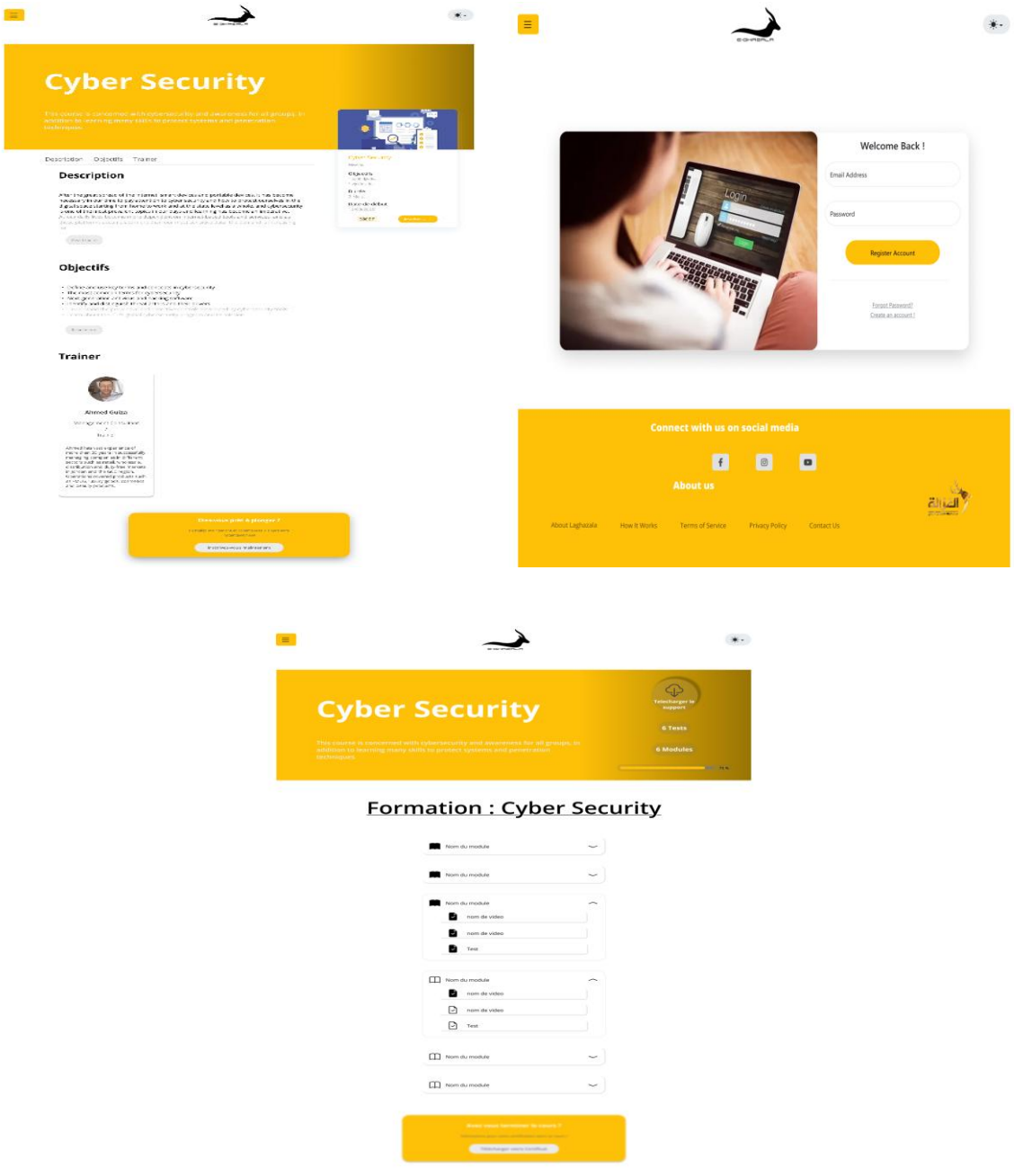


Figure 11Maquette Figma