ⵜⴰⵣⵍⵎⵉⵜ ⵏ ⵍⵎⵖⵔⵉⴱ
ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵜⵣⵔⴼⵜ ⴰⵏⴰⵎⵓⵔ ⴷ ⵜⵣⵡⵉⵔⵉ ⴰⵎⵙⵙⴰⵏ
ⴷ ⵜⵙⵡⵉⵍⴰ ⴰⵏⵉⵎⵓⵔⴰ ⴷ ⵜⵙⴰⵡⴰⵙ ⴰⵎⴰⵡⴰⵍ

المملكة المغربية
وزارة التربية الوطنية والتكوين المهني
والتعليم العالي والبحث العلمي

Académie régionale de Fès-Meknès

Direction provinciale de TAZA

Centre de préparation Brevet de Technicien Supérieur Lycée technique

Filière : **Multimédia et conception web**

Rapport de stage

Effectué au sein de : Université Sidi Mohamed Ben Abdellah
Faculté Polydisciplinaire de Taza Du 13/05/2025 au 13/06/2025

Sous le thème :

Développement d’une application web de gestion des parcours étudiants

Réalisé par l’étudiant : Abdelbadie Khoubiza

Encadrement pédagogique : équipe pédagogique

Encadrement de stage : responsable de stage

Édition publique du portfolio. Données personnelles minimisées ; objectifs et résultats distingués. Le rapport conserve sa structure académique.

Année professionnelle : 2024/2025

Table des matières

Remerciements **5**

Résumé **6**

Introduction Générale **7**

I Cadrage du Projet 9

1 Problématique et Objectifs **10**

1.1 Problématiques Identifiées 10

1.1.1 Suivi défaillant des parcours 10

1.1.2 Manque de centralisation des informations 10

1.1.3 Processus manuels chronophages 11

1.2 Objectifs du Projet (formulés de manière SMART) 11

1.2.1 Amélioration de l'efficacité administrative 11

1.2.2 Optimisation de l'expérience utilisateur 11

1.2.3 Centralisation et sécurisation des données 12

1.2.4 Modernisation des processus académiques 12

1.2.5 Conception et développement technique 12

2 Analyse et Spécification des Besoins **14**

2.1 Analyse de l'Existant 14

2.1.1 Rôle du Service 14

2.1.2 Processus Actuels 14

2.1.3 Interaction Étudiante avec le Système 15

2.2 Modélisation et Spécification des Besoins 15

2.2.1 Modélisation du Parcours Utilisateur 15

2.2.2 Spécifications Fonctionnelles 16

2.2.3 Spécifications Non-Fonctionnelles 19

2.2.4 Modélisation des Besoins Centrée Utilisateur 20

II Réalisation Technique **22**

3 Conception et Architecture Technique **23**
3.1 Choix Technologiques et Environnement de Développement 23
3.1.1 Stack Technique Backend 23
3.1.2 Stack Technique Frontend 23
3.1.3 Base de Données et Environnement 24
3.2 Architecture Logicielle 24
3.2.1 Implémentation du Pattern MVC 24
3.2.2 Description de la Couche Modèle 24
3.2.3 Description de la Couche Vue 25
3.2.4 Description de la Couche Contrôleur 25
3.2.5 Analyse des Avantages Architecturaux 25
3.3 Conception de la Base de Données 25
3.3.1 Structure des Entités Principales 25
3.3.2 Relations et Intégrité Référentielle 26
3.3.3 Stratégies d'Optimisation et de Performance 26
3.3.4 Considérations d'Évolutivité et de Maintenance 27

4 Mise en œuvre et Développement **28**
4.1 Développement Backend 28
4.1.1 Architecture des Modèles Eloquent 28
4.1.2 Logique Métier dans les Contrôleurs 29
4.1.3 Système d'Authentification et Contrôle d'Accès 29
4.1.4 Gestion de la Base de Données avec les Migrations 30
4.1.5 Services Métier et Architecture Modulaire 30
4.2 Développement Frontend 30
4.2.1 Conception de l'Expérience Utilisateur 30
4.2.2 Stratégie de Design et Justifications Techniques 31
4.2.3 Architecture des Composants Blade 31
4.2.4 Responsive Design et Accessibilité 31
4.2.5 Éléments d'Interface Clés 31
4.3 Implémentation des Fonctionnalités Clés 32
4.3.1 Gestion du Profil Étudiant 32
4.3.2 Sélection et Confirmation de Parcours 32
4.3.3 Génération d'Attestations Sécurisées 32
4.3.4 Historique et Traçabilité des Actions 33
4.3.5 Interface de Suivi des Demandes 33

5 Stratégie de Tests et Validation **34**
5.1 Tests Automatisés avec PHPUnit 34

5.1.1 Architecture de Tests . . . 34
5.1.2 Exemples Concrets d'Implémentation . . . 34
5.2 Données de Test avec les Seeders . . . 35
5.2.1 Stratégie de Peuplement des Données . . . 35
5.2.2 Conception des Scénarios de Test . . . 35
5.3 Validation des Données . . . 36
5.3.1 Sécurisation par FormRequest . . . 36
5.3.2 Implémentation Rigoureuse . . . 36
5.4 Gestion des Erreurs et Défis . . . 37
5.4.1 Architecture de Gestion d'Erreurs . . . 37
5.4.2 Stratégie de Journalisation . . . 37
5.4.3 Défis Techniques Rencontrés . . . 37
5.4.4 Solutions Adoptées . . . 38
5.5 Conclusion du Chapitre . . . 38

III Bilan et Perspectives 39

6 Évaluation du Projet et Perspectives d'Évolution 40
6.1 Évaluation du Projet . . . 40
6.1.1 Conformité aux Objectifs Initiaux . . . 40
6.1.2 Points Forts du Projet . . . 40
6.1.3 Limitations et Axes d'Amélioration . . . 41
6.1.4 Performance et Robustesse . . . 41
6.2 Perspectives d'Évolution . . . 42
6.2.1 Extensions Fonctionnelles Stratégiques . . . 42
6.2.2 Améliorations Techniques et Architecturales . . . 42
6.2.3 Innovation et Technologies Émergentes . . . 43
6.2.4 Scalabilité et Infrastructure . . . 43
6.3 Conclusion Stratégique . . . 44

7 Bilan Personnel et Compétences Acquises 45
7.1 Compétences Techniques Acquises . . . 45
7.1.1 Compréhension des Principes Architecturaux Avancés . . . 45
7.1.2 Application Systématique des Patterns de Conception . . . 45
7.1.3 Développement de Composants Réutilisables . . . 46
7.1.4 Vision Systémique . . . 46

Bibliographie 46

IV Annexes **48**

A Architecture Technique Détaillée **49**
A.1 Diagramme de l'Architecture Système 49
A.2 Structure des Dossiers du Projet 49
A.2.1 Structure Principale 49
A.2.2 Dossiers Clés et Leur Rôle 50
A.2.3 Fichiers de Configuration Importants 50

B Modèle de Base de Données **51**
B.1 Schéma Relationnel 52
B.2 Structure SQL des Tables Principales 53
B.3 Dictionnaire des Données 53

C Interfaces Utilisateur **54**
C.1 Diagramme de Flux Utilisateur 55
C.2 Captures d'Écran des Principales Interfaces 56
C.2.1 Interface de Connexion 56
C.2.2 Sécurité au Premier Accès 56
C.2.3 Tableau de Bord Étudiant 57
C.2.4 Consultation du Profil et des Résultats 57
C.2.5 Interface de Sélection de Parcours 59
C.2.6 Génération et Visualisation de l'Attestation 60
C.3 Responsive Design 62

D Sécurité et Authentification **63**
D.1 Mécanismes de sécurité 63
D.2 Configuration illustrative 63

E Considérations de Sécurité **64**
E.1 Mesures de Sécurité Natives de Laravel 64
E.2 Axes d'Amélioration de la Sécurité 64

F Stratégie de Tests et Qualité **66**
F.1 Tests Fonctionnels avec PHPUnit 66
F.2 Perspectives d'Amélioration de la Qualité 66

G Déploiement et Configuration **68**
G.1 Variables d'Environnement (.env) 68

H Manuel Utilisateur **69**
H.1 Guide de Connexion 69
H.2 Procédure de Sélection de Parcours 69

Remerciements

Ce rapport a été élaboré au terme de mon stage de fin de formation pour l'obtention du Brevet de Technicien Supérieur (BTS) en **Multimédia et Conception Web**. Ce stage, d'une durée d'un mois, s'est déroulé du 13 mai au 13 juin 2025 au sein de la **Faculté Polydisciplinaire de Taza**, rattachée à l'Université Sidi Mohamed Ben Abdellah. Cette expérience professionnelle a été une opportunité inestimable de mettre en pratique les connaissances acquises durant mon cursus et de m'immerger dans un projet de développement concret et à fort impact.

Avant d'entamer la présentation de mon travail, il me semble essentiel d'exprimer ma profonde gratitude envers ceux qui ont contribué au succès de cette expérience. Je tiens tout d'abord à remercier la direction de la Faculté Polydisciplinaire de Taza de m'avoir accueilli.

Mes remerciements les plus sincères s'adressent à mon encadrante de stage pour sa disponibilité, ses conseils avisés et la confiance qu'elle m'a accordée. Son expertise et son accompagnement constant ont été essentiels à la réalisation de ce projet.

Je souhaite également exprimer ma reconnaissance envers mon encadrante pédagogique pour son suivi rigoureux, sa bienveillance et son soutien tout au long de mon parcours. J'adresse aussi mes remerciements à l'ensemble du corps professoral du Centre de préparation au BTS du Lycée Technique de Taza, dont les enseignements m'ont fourni les bases nécessaires pour mener à bien cette mission.

Je n'oublie pas de remercier l'ensemble du personnel de la faculté pour leur accueil chaleureux et l'environnement de travail collaboratif qu'ils ont su créer.

Enfin, un grand merci à ma famille et à mes amis pour leur soutien moral et leurs encouragements indéfectibles durant toutes mes années d'études.

Ce document retrace les différentes étapes de mon projet, de l'analyse des besoins à la réalisation technique, et dresse le bilan des compétences que j'ai pu acquérir. J'espère qu'il témoignera de la richesse de cette expérience et de mon investissement personnel.

Résumé

Ce rapport de stage présente la conception et le développement d'une application web visant à moderniser la gestion des parcours étudiants à la Faculté Polydisciplinaire de Taza. Le projet répond à une problématique majeure de processus administratifs manuels chronophages et d'un manque de suivi centralisé des informations académiques.

Le framework Laravel 12 et l'architecture MVC structurent l'application développée. Les interfaces et extraits présentés documentent la consultation et la confirmation du parcours académique en ligne, ainsi que la génération d'attestations PDF.

Le système met en oeuvre les principaux parcours décrits dans ce rapport. Les gains administratifs, la satisfaction des utilisateurs et l'atteinte des objectifs chiffrés restent à mesurer. Les outils d'analyse avancée et de reporting détaillé figurent parmi les évolutions envisagées.

Abstract

This internship report presents the design and development of a comprehensive web application aimed at modernizing student pathway management at the Polydisciplinary Faculty of Taza. The project addresses critical issues of time-consuming manual administrative processes and the absence of centralized academic information tracking.

The application uses Laravel 12 and an MVC architecture. The interfaces and excerpts presented in this report document viewing and confirming academic pathways online and generating PDF certificates.

The system implements the main workflows described in this report. Administrative efficiency gains, user satisfaction and the numerical targets remain to be measured. Advanced analytics and detailed reporting are among the proposed future developments.

Introduction Générale

Contexte de la transformation numérique

L'enseignement supérieur marocain traverse une période de transformation numérique majeure, où les établissements universitaires repensent leurs processus administratifs pour répondre aux exigences croissantes d'efficacité et d'amélioration de l'expérience étudiante. Dans ce contexte d'évolution technologique, la Faculté Polydisciplinaire de Taza (FPT), composante de l'Université Sidi Mohamed Ben Abdellah (USMBA) de Fès, illustre parfaitement cette dynamique de modernisation institutionnelle.

Présentation de l'organisme d'accueil

Fondée en 2003 sous la direction de M. Hassan Tabyaoui, la FPT s'est imposée comme un acteur clé de la formation supérieure régionale, répondant aux besoins éducatifs spécifiques de la région de Taza. En tant qu'établissement public d'enseignement supérieur, elle dispense une formation pluridisciplinaire de premier cycle à travers 16 filières de licence et 13 mentions de master, couvrant un large spectre de domaines allant des sciences au droit, en passant par l'économie. Cette diversité académique positionne la faculté comme un pilier de la préparation des étudiants aux cycles supérieurs et à leur insertion professionnelle.

Positionnement du projet

L'engagement de la FPT dans la transformation numérique s'inscrit dans une démarche stratégique visant à optimiser l'efficacité administrative tout en enrichissant l'expérience étudiante. Cette orientation répond à un défi contemporain majeur de l'enseignement supérieur : concilier la massification de l'accès à l'université avec la nécessité de maintenir une gestion académique de qualité et personnalisée.

Au cœur de cette transformation digitale, l'USMBA a initié le développement d'un service de gestion des parcours étudiants, visant à centraliser et automatiser les processus académiques traditionnellement gérés de manière dispersée. Cette initiative représente une réponse concrète aux défis de modernisation que rencontrent les établissements d'enseignement supérieur marocains dans leur quête d'optimisation administrative et d'amélioration de l'expérience étudiante.

Problématique centrale

Le projet s'articule autour d'une problématique centrale : comment concevoir et déployer une solution numérique capable de fluidifier les interactions entre les différents acteurs de la communauté universitaire, tout en garantissant une gestion académique rigoureuse et transparente ? Cette interrogation soulève des enjeux tant techniques qu'organisationnels, nécessitant une approche méthodologique approfondie pour analyser les besoins existants et proposer des solutions adaptées au contexte spécifique de l'établissement.

Plan et objectifs

Ce rapport de stage se propose d'analyser en profondeur cette initiative de digitalisation, en examinant ses implications techniques, organisationnelles et pédagogiques. Il s'attachera à démontrer comment ce projet s'inscrit dans une vision plus large de modernisation de l'enseignement supérieur marocain, tout en répondant aux défis spécifiques de gestion des parcours étudiants dans un environnement académique en constante évolution.

Première partie

Cadrage du Projet

1. Problématique et Objectifs

1.1. Problématiques Identifiées

L'analyse préliminaire du système de gestion académique de l'USMBA a révélé plusieurs dysfonctionnements majeurs impactant directement l'efficacité opérationnelle et la satisfaction des usagers. Ces problématiques, comparables aux goulots d'étranglement dans un système de traitement distribué, nécessitent une approche systémique pour leur résolution.

1.1.1. Suivi défaillant des parcours

L'absence d'un système de traçabilité cohérent complique le suivi longitudinal des étudiants, générant des difficultés dans l'accompagnement personnalisé et la prise de décisions académiques éclairées. Cette situation peut être comparée à un système embarqué dépourvu de capteurs de monitoring : sans mécanisme de feedback continu, il devient impossible d'optimiser les performances ou de détecter les anomalies en temps réel.

Les conséquences observées incluent :

— Difficultés d'identification des étudiants en situation d'échec académique

— Impossibilité de fournir un accompagnement proactif personnalisé

— Perte de données historiques cruciales pour l'analyse des tendances

— Multiplication des démarches redondantes pour reconstituer l'historique académique

1.1.2. Manque de centralisation des informations

La dispersion des données académiques à travers différents systèmes et supports physiques crée des silos informationnels, entravant l'accès rapide aux informations et augmentant les risques d'incohérences. Cette architecture fragmentée rappelle un réseau de capteurs IoT mal intégré, où chaque nœud collecte des données sans coordination centrale, compromettant la vision globale du système.

Les impacts identifiés sont les suivants :

- Recherche d'informations ralentie par la dispersion des sources.

— Risques élevés d'incohérences entre les différentes sources de données

— Duplication des saisies et maintenance multiple des mêmes informations

— Difficultés de génération de rapports consolidés et fiables

1.1.3. Processus manuels chronophages

La persistance de procédures administratives manuelles engendre des délais de traitement importants, multiplie les risques d'erreurs humaines et mobilise excessivement les ressources humaines sur des tâches à faible valeur ajoutée. Cette situation est analogue à un système de contrôle industriel fonctionnant sans automatisation : chaque opération nécessite une intervention humaine, limitant la cadence et introduisant une variabilité de performance.

Les dysfonctionnements recensés comprennent :

- Délais liés au traitement manuel des dossiers d'inscription.
- Risque d'erreurs lors des saisies manuelles.
- Temps administratif consacré aux tâches répétitives.

— Indisponibilité fréquente des services pendant les pics d'activité

1.2. Objectifs du Projet (formulés de manière SMART)

Les objectifs SMART ci-dessous sont des cibles proposées, non des résultats observés. Le rapport ne présente aucune campagne de mesure démontrant leur atteinte. Les valeurs de départ, délais et pourcentages associés restent à établir.

1.2.1. Amélioration de l'efficacité administrative

Objectif spécifique : Automatiser les processus de gestion des parcours étudiants en visant une réduction de 30% des délais de traitement des inscriptions, une diminution significative des erreurs administratives et une optimisation de l'allocation des ressources humaines vers des missions à plus forte valeur ajoutée.

Indicateurs cibles à valider :

— Réduction du temps de traitement moyen de 10 jours à 7 jours maximum

— Diminution du taux d'erreur de 12% à moins de 3%

— Libération de 40% du temps administratif pour des missions stratégiques

— Traitement automatisé de 80% des demandes courantes

Cette approche s'inspire des principes de l'automatisation industrielle, où l'optimisation des flux de production permet d'augmenter simultanément la qualité et la productivité.

1.2.2. Optimisation de l'expérience utilisateur

Objectif spécifique : Développer une interface intuitive et accessible permettant aux étudiants et au personnel administratif d'accéder facilement aux informations et en fournissant au personnel administratif les données consolidées nécessaires à leurs missions, en visant un score de satisfaction utilisateur supérieur à 4/5 sur les enquêtes post-déploiement.

Indicateurs cibles à valider :

— Score de satisfaction utilisateur ≥ 4/5 dans les 6 mois suivant le déploiement

— Temps d'accès aux informations réduit de 75% (de 4 minutes à 1 minute)

— Taux d'adoption de la plateforme ≥ 85% après 3 mois

— Réduction de 60% des demandes de support liées à la navigation

Cette démarche suit les principes d'ergonomie des interfaces homme-machine (IHM), privilégiant la simplicité d'usage et l'intuitivité, à l'image des systèmes de contrôle des cockpits modernes où la clarté informationnelle est cruciale.

1.2.3. Centralisation et sécurisation des données

Objectif spécifique : Créer un référentiel unique et sécurisé des informations académiques, assurant une disponibilité des données de 99,5% et leur intégrité dans le temps.

Indicateurs cibles à valider :

— Disponibilité du système ≥ 99,5% (soit moins de 44 heures d'indisponibilité par an)

— Zéro perte de données critiques avec mécanisme de sauvegarde automatique

— Temps de récupération après incident < 2 heures (RTO)

- Protection des données sensibles à définir et à valider.

L'architecture proposée s'inspire des systèmes de bases de données critiques utilisés dans l'aéronautique, où la redondance et la tolérance aux pannes sont essentielles.

1.2.4. Modernisation des processus académiques

Objectif spécifique : Transformer les procédures traditionnelles en workflows numériques structurés, permettant une réduction de 50% du temps de traitement des dossiers étudiants et une prise de décision basée sur des données fiables et actualisées.

Indicateurs cibles à valider :

— Réduction du temps de traitement global de 15 jours à 7,5 jours maximum

— Automatisation de 75% des étapes de validation standard

— Génération automatique de 90% des documents administratifs

— Traçabilité complète des actions avec horodatage et identification des intervenants

Cette approche s'apparente aux systèmes de gestion de workflow industriels (BPMN), où chaque étape est optimisée et surveillée pour garantir l'efficacité globale du processus.

1.2.5. Conception et développement technique

Objectif spécifique : Concevoir, développer et déployer une plateforme web robuste et évolutive basée sur le framework Laravel, servant de support technique à l'ensemble des processus de gestion des parcours.

Indicateurs cibles à valider :

— Architecture modulaire permettant l'ajout de nouvelles fonctionnalités sans refactoring majeur

- Cible : réponse < 2 secondes pour 95 % des requêtes.
- Cible : 1000 utilisateurs simultanés, à valider par tests de charge.
- Cible future : couverture des tests automatisés de 80 % du code critique.

La conception répartit les responsabilités entre modèles, vues, contrôleurs et services métier. Cette organisation vise à faciliter la maintenance ; elle ne constitue pas une preuve de performance ou de capacité en production.

Ces objectifs accompagnent la modernisation des services de la Faculté Polydisciplinaire de Taza. Leur atteinte devra être évaluée par des mesures documentées de disponibilité, de charge, de couverture de tests, de satisfaction et de gains administratifs. Les fonctionnalités effectivement décrites sont présentées dans la partie Réalisation Technique.

2. Analyse et Spécification des Besoins

2.1. Analyse de l'Existant

L'analyse de l'existant constitue une étape fondamentale dans la compréhension du fonctionnement actuel du service de gestion des parcours étudiants à l'Université Sidi Mohamed Ben Abdellah (USMBA). Cette analyse permet d'identifier les processus en place, les mécanismes de gestion actuels et l'interaction des étudiants avec le système avant d'envisager toute évolution ou amélioration du portail étudiant.

2.1.1. Rôle du Service

Le service de gestion des parcours étudiants occupe une position centrale dans l'écosystème académique de l'USMBA. Sa mission principale consiste à centraliser l'ensemble des processus liés à la gestion académique des étudiants, depuis leur inscription jusqu'au suivi de leur progression universitaire.

Le service assure trois fonctions essentielles au sein de l'établissement. Premièrement, il facilite et automatise la sélection des parcours académiques, permettant aux étudiants de faire des choix éclairés en fonction de leurs profils et des critères d'admission. Deuxièmement, il gère l'ensemble des processus d'inscription administrative et pédagogique, garantissant la cohérence et la traçabilité des données étudiantes. Troisièmement, il assure le suivi académique continu des étudiants tout au long de leur cursus universitaire.

Le service joue également un rôle d'interface crucial entre les étudiants et l'administration universitaire. Il garantit une communication fluide, permettant aux étudiants d'accéder aux informations nécessaires tout en offrant à l'administration une vision consolidée de la population étudiante et de ses besoins.

2.1.2. Processus Actuels

L'analyse fonctionnelle révèle l'existence de processus automatisés qui remplacent progressivement les méthodes manuelles traditionnelles. Ces processus s'articulent autour de deux axes principaux : la gestion des parcours et la gestion des profils étudiants.

Le processus de sélection des parcours constitue l'un des mécanismes les plus sophistiqués du système actuel. Ce processus débute par une phase de vérification automatique des prérequis et des permissions, déterminant si un étudiant dispose de la capacité de choisir librement son parcours académique. Dans le cas où cette capacité est confirmée, l'étudiant accède à une interface de sélection lui permettant d'examiner les options disponibles et de formuler son choix. Inversement,

lorsque les conditions ne sont pas remplies, le système procède à une assignation automatique vers un parcours par défaut, respectant ainsi les règles institutionnelles tout en garantissant la continuité du parcours étudiant.

Le processus de mise à jour des informations personnelles constitue le second pilier fonctionnel du système. Ce processus intègre des mécanismes de validation automatiques, vérifiant la cohérence et la conformité des données saisies par les étudiants. Chaque modification fait l'objet d'un enregistrement automatique dans l'historique des actions, assurant la traçabilité des changements et permettant un contrôle institutionnel si nécessaire.

Ces processus automatisés présentent l'avantage significatif de réduire les risques d'erreurs tout en améliorant l'efficacité administrative globale. Ils permettent également une standardisation des procédures, garantissant un traitement équitable de tous les dossiers étudiants.

2.1.3. Interaction Étudiante avec le Système

L'étudiant constitue l'utilisateur principal du portail et interagit avec le système selon des permissions et des règles définies par l'administration. Son interaction avec le service se concentre essentiellement sur la gestion de son parcours académique et la maintenance de ses informations personnelles.

L'étudiant dispose de fonctionnalités lui permettant de consulter les options de parcours disponibles selon son profil académique et de formuler ses choix dans les limites autorisées par les règlements institutionnels. Il peut également mettre à jour ses données personnelles selon les règles établies par l'institution, avec une validation automatique garantissant la conformité des informations saisies.

Le système fournit à l'étudiant une interface personnalisée reflétant son statut académique, ses permissions spécifiques et l'historique de ses actions. Cette approche individualisée permet une expérience utilisateur adaptée tout en respectant les contraintes administratives et réglementaires de l'établissement.

L'autonomie accordée à l'étudiant s'inscrit dans un cadre contrôlé par des règles métier automatisées, assurant la cohérence des choix effectués avec les exigences académiques et administratives. Cette approche garantit un équilibre entre la liberté de choix étudiante et le respect des contraintes institutionnelles.

2.2. Modélisation et Spécification des Besoins

L'analyse de l'existant a permis d'identifier les besoins fonctionnels et non-fonctionnels de la future application. Pour synthétiser ces exigences et illustrer l'expérience utilisateur cible, le modèle graphique suivant est présenté.

2.2.1. Modélisation du Parcours Utilisateur

Le diagramme de flux ci-après (figure C.1) est l'outil central de modélisation pour ce projet. Il détaille de manière séquentielle le parcours complet de l'étudiant, depuis son authentification

jusqu'à la validation finale de son choix. Ce schéma sert de pont visuel entre les besoins fonctionnels identifiés et l'interface qui a été développée, en clarifiant chaque étape de l'interaction de l'utilisateur avec le système.

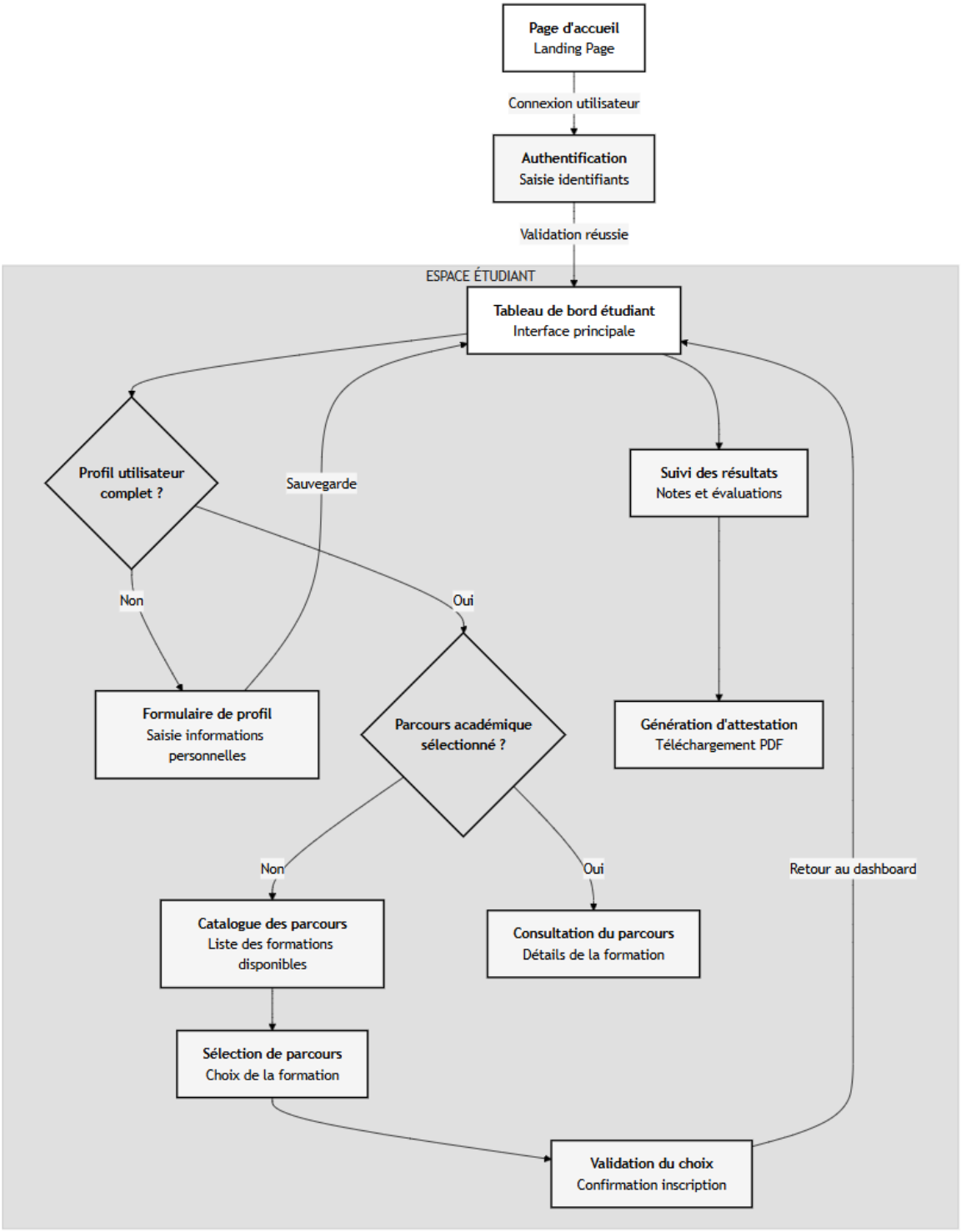


FIGURE 2.1 – Diagramme de flux détaillé du parcours utilisateur étudiant

2.2.2. Spécifications Fonctionnelles

Les spécifications suivantes sont des exigences cibles, distinctes du périmètre livré, pour répondre aux besoins quotidiens de gestion du parcours acadé-

mique.

SF1. Gestion du Profil Personnel

SF1.1 Consultation et Modification du Profil

— L'étudiant doit pouvoir consulter l'ensemble de ses informations personnelles et académiques

— L'étudiant doit pouvoir modifier ses données personnelles modifiables (coordonnées, informations de contact)

— L'application doit permettre à l'étudiant de télécharger ou mettre à jour sa photo de profil

— L'étudiant doit pouvoir consulter l'historique des modifications apportées à son profil

SF1.2 Validation et Sécurisation des Données

— L'application doit valider en temps réel les données saisies par l'étudiant

— L'étudiant doit recevoir des messages d'erreur clairs en cas de saisie incorrecte

— L'application doit sauvegarder automatiquement les modifications validées

— L'étudiant doit pouvoir annuler ses modifications avant validation définitive

SF2. Gestion du Parcours Académique

SF2.1 Exploration et Sélection des Parcours

— L'étudiant doit pouvoir consulter la liste des parcours disponibles selon son profil académique

— L'application doit présenter pour chaque parcours : objectifs, contenu pédagogique, prérequis et débouchés

— L'étudiant doit pouvoir comparer plusieurs parcours côte à côte

— L'application doit indiquer clairement si l'étudiant est éligible ou non à chaque parcours

SF2.2 Processus de Choix et Confirmation

— L'étudiant éligible au choix libre doit pouvoir sélectionner son parcours préféré

— L'application doit informer l'étudiant si un parcours par défaut lui est assigné automatiquement

— L'étudiant doit pouvoir confirmer définitivement son choix de parcours

— L'application doit permettre à l'étudiant de modifier son choix dans les délais réglementaires

SF2.3 Suivi du Statut du Parcours

— L'étudiant doit être informé en temps réel du statut de sa demande de parcours

— L'application doit notifier l'étudiant des étapes de validation (en cours, approuvée, refusée)

— L'étudiant doit pouvoir consulter les justifications en cas de refus de parcours

— L'application doit proposer des alternatives en cas d'impossibilité d'attribution du parcours souhaité

SF3. Consultation des Résultats Académiques

SF3.1 Visualisation des Performances

— L'étudiant doit pouvoir consulter ses résultats organisés par semestre et par matière
— L'application doit afficher les moyennes générales, par module et par unité d'enseignement
— L'étudiant doit pouvoir visualiser l'évolution de ses performances à travers des graphiques
— L'application doit indiquer le statut de validation de chaque semestre et année académique

SF3.2 Analyse et Projection Académique

— L'étudiant doit pouvoir consulter son classement relatif dans sa promotion
— L'application doit calculer et afficher les crédits ECTS acquis et restants
— L'étudiant doit pouvoir simuler l'impact de futures notes sur sa moyenne générale
— L'application doit fournir des recommandations personnalisées basées sur les performances

SF4. Documentation et Export

SF4.1 Génération de Documents Officiels

— L'étudiant doit pouvoir télécharger ses relevés de notes au format PDF
— L'application doit générer des attestations de scolarité personnalisées
— L'étudiant doit pouvoir exporter un récapitulatif de son parcours académique
— L'application doit permettre l'impression directe des documents depuis l'interface

SF4.2 Historique et Archivage Personnel

— L'étudiant doit pouvoir consulter l'historique complet de son parcours académique
— L'application doit permettre le téléchargement des documents des années précédentes
— L'étudiant doit pouvoir créer des collections personnalisées de ses documents
— L'application doit offrir un espace de stockage personnel pour les documents complémentaires

SF5. Communication et Notifications

SF5.1 Système de Notifications

— L'étudiant doit recevoir des notifications pour les échéances importantes de son parcours
— L'application doit alerter l'étudiant des nouveaux résultats disponibles
— L'étudiant doit être notifié des changements de statut de sa demande de parcours
— L'application doit permettre à l'étudiant de personnaliser ses préférences de notification

SF5.2 Support et Assistance

— L'étudiant doit pouvoir accéder à une aide contextuelle depuis chaque écran
— L'application doit proposer une FAQ interactive adaptée aux questions fréquentes
— L'étudiant doit pouvoir contacter le support technique directement depuis l'application
— L'application doit fournir des tutoriels vidéo pour les fonctionnalités principales

2.2.3. Spécifications Non-Fonctionnelles

Les spécifications non-fonctionnelles fixent des exigences cibles. La conformité, la disponibilité et les performances demandées ne sont pas démontrées par des mesures dans ce rapport.

SNF1. Expérience Utilisateur et Accessibilité

SNF1.1 Interface et Ergonomie

— L'application doit proposer une interface intuitive ne nécessitant aucune formation préalable

— L'application doit être entièrement responsive et s'adapter à tous les types d'écrans (mobile, tablette, desktop)

— L'interface doit respecter les standards d'accessibilité WCAG 2.1 niveau AA

— L'étudiant doit pouvoir personnaliser l'apparence de son interface (thème sombre/clair, taille de police)

SNF1.2 Facilité d'Utilisation

— L'étudiant doit pouvoir accomplir les tâches principales en moins de 3 clics

— L'application doit offrir des raccourcis clavier pour les utilisateurs avancés

— L'interface doit être cohérente dans tous les modules de l'application

— L'étudiant doit pouvoir revenir facilement à l'écran précédent à tout moment

SNF2. Performance et Disponibilité

SNF2.1 Temps de Réponse et Fluidité

— L'application doit afficher les pages principales en moins de 2 secondes

— Les interactions utilisateur doivent avoir un feedback visuel immédiat (< 0,1 seconde)

— L'application doit fonctionner fluidement même avec une connexion internet lente

— Le chargement des documents PDF doit s'effectuer en moins de 5 secondes

SNF2.2 Disponibilité et Fiabilité

— L'application doit être disponible 24h/24, 7j/7 avec un taux de disponibilité minimum de 99%

— L'application doit fonctionner de manière stable pendant les périodes de forte affluence (inscriptions, publication des résultats)

— L'étudiant doit pouvoir utiliser les fonctionnalités essentielles même en cas de panne partielle

— L'application doit effectuer des sauvegardes automatiques pour éviter la perte de données

SNF3. Sécurité et Confidentialité

SNF3.1 Protection des Données Personnelles

— L'application doit chiffrer toutes les données sensibles de l'étudiant

— L'étudiant doit être le seul à pouvoir accéder à ses informations personnelles et académiques

— L'application doit respecter rigoureusement le RGPD et les réglementations sur la protection des données

— L'étudiant doit pouvoir consulter et contrôler l'utilisation de ses données personnelles

SNF3.2 Sécurisation de l'Accès

— L'application doit proposer une authentification sécurisée (double facteur optionnel)

— L'étudiant doit être automatiquement déconnecté après une période d'inactivité

— L'application doit détecter et signaler les tentatives de connexion suspectes

— L'étudiant doit pouvoir consulter l'historique de ses connexions

SNF4. Compatibilité et Support Technique

SNF4.1 Compatibilité Multi-Plateforme

— L'application doit fonctionner sur tous les navigateurs modernes (Chrome, Firefox, Safari, Edge)

— L'application doit être compatible avec les systèmes d'exploitation mobiles (iOS, Android)

— L'application doit supporter les technologies d'assistance pour les étudiants en situation de handicap

— L'application doit fonctionner même avec JavaScript partiellement désactivé

SNF4.2 Maintenance et Évolution

— L'application doit pouvoir être mise à jour sans interruption de service pour l'étudiant

— L'étudiant doit être informé à l'avance des maintenances programmées

— L'application doit conserver la compatibilité avec les versions précédentes des documents

— L'étudiant doit pouvoir continuer à utiliser l'application pendant les mises à jour mineures

2.2.4. Modélisation des Besoins Centrée Utilisateur

Afin de synthétiser les interactions de l'utilisateur avec le système, le diagramme de flux suivant est présenté. Il modélise le parcours complet de l'étudiant, depuis son authentification jusqu'à la validation de ses choix, illustrant ainsi de manière concrète les spécifications fonctionnelles définies précédemment.

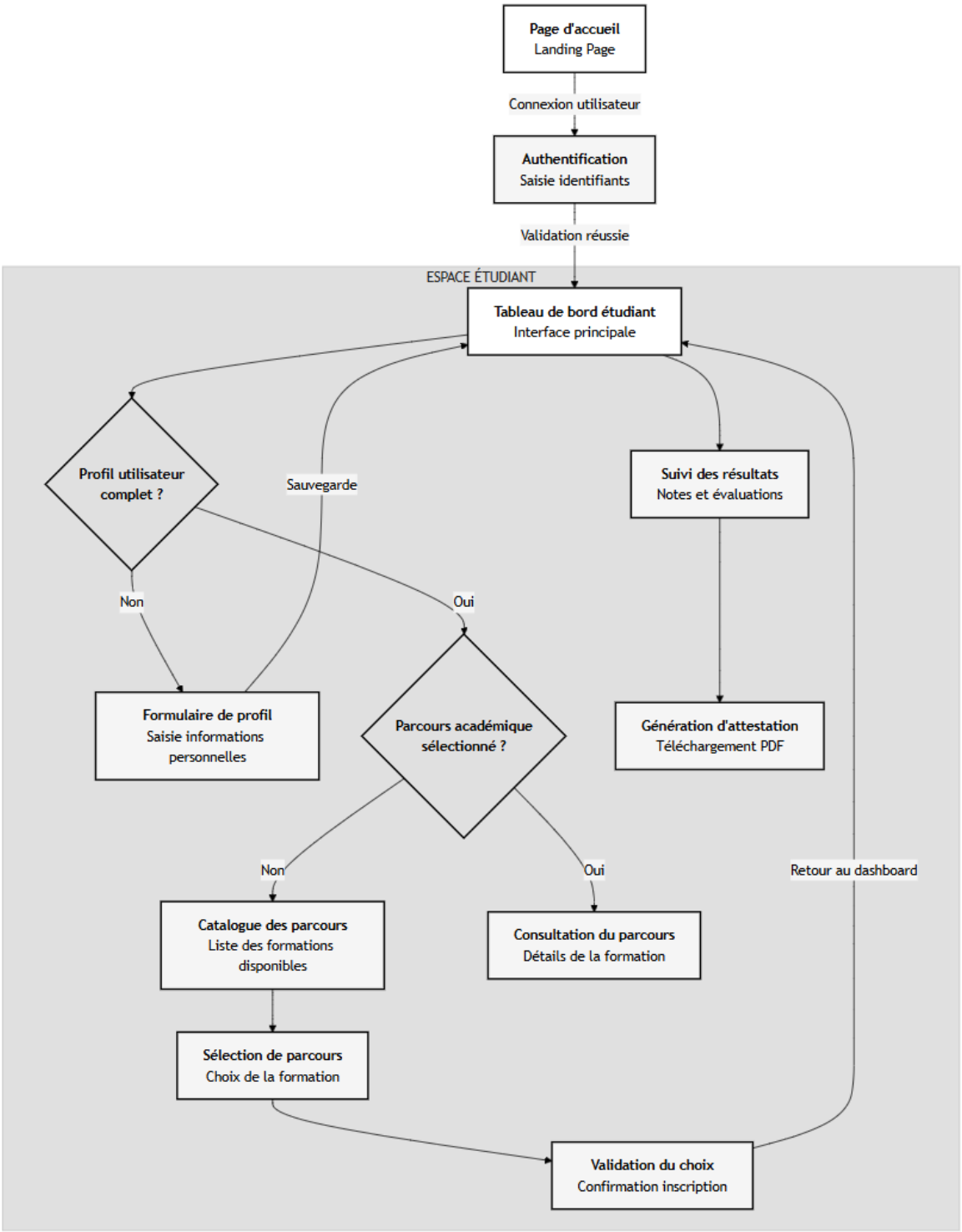


FIGURE 2.2 – Diagramme de flux du parcours utilisateur au sein de l'application

Deuxième partie

Réalisation Technique

3. Conception et Architecture Technique

3.1. Choix Technologiques et Environnement de Développement

3.1.1. Stack Technique Backend

Le développement backend repose sur une architecture PHP moderne et robuste. **PHP 8.2** a été sélectionné comme langage principal en raison de ses performances optimisées et de sa compatibilité native avec l'écosystème Laravel. Cette version apporte des améliorations significatives en termes de vitesse d'exécution et de gestion mémoire, essentielles pour une application de gestion académique traitant de volumes importants de données étudiantes.

Laravel 12.0 constitue le framework central de l'application. Ce choix s'appuie sur plusieurs facteurs déterminants : la structure MVC intégrée qui facilite la séparation des préoccupations, l'ORM Eloquent pour une gestion intuitive des données, et un écosystème riche en packages spécialisés. L'architecture Laravel permet un développement rapide tout en maintenant un code organisé et maintenable.

Plusieurs packages stratégiques complètent cette base technique. **Laravel Sanctum** assure la gestion des authentifications API avec un système de tokens sécurisés, crucial pour protéger les données sensibles des étudiants. **Barryvdh/laravel-dompdf** permet la génération automatisée de documents PDF pour les résultats académiques et les attestations. **Doctrine/dbal** facilite les migrations complexes de schéma de base de données, garantissant une évolution contrôlée de la structure des données.

3.1.2. Stack Technique Frontend

L'interface utilisateur s'appuie sur des technologies modernes privilégiant la performance et la réactivité. **Vite** a été choisi comme outil de build pour ses temps de compilation exceptionnellement rapides et son support natif du Hot Module Replacement, accélérant considérablement le cycle de développement.

Tailwind CSS structure l'approche de stylisation avec son paradigme utility-first. Ce framework CSS permet de créer des interfaces cohérentes et responsive sans écrire de CSS personnalisé, tout en maintenant un bundle final optimisé. La modularité de Tailwind s'aligne parfaitement avec les besoins d'une application institutionnelle nécessitant de nombreux formulaires et tableaux de données.

Alpine.js apporte l'interactivité nécessaire avec une empreinte minimale. Ce framework JavaScript léger permet d'ajouter des comportements dynamiques aux interfaces sans la complexité

d'un framework SPA, maintenant ainsi la simplicité architecturale tout en enrichissant l'expérience utilisateur.

3.1.3. Base de Données et Environnement

MySQL a été retenu comme système de gestion de base de données pour sa robustesse éprouvée et sa capacité à gérer efficacement de grandes quantités de données relationnelles. Cette solution offre les garanties ACID nécessaires pour la cohérence des données académiques et supporte nativement les transactions complexes requises par les opérations d'inscription et de notation.

L'environnement de développement intègre des outils standards de l'écosystème moderne. **Composer** gère les dépendances PHP avec un système de verrouillage de versions garantissant la reproductibilité des déploiements. **NPM** assure la gestion des packages frontend et l'exécution des scripts de build. **Git** structure le contrôle de version avec une approche collaborative permettant le travail en équipe.

La configuration de l'application utilise le système de variables d'environnement via le fichier `.env`, séparant les paramètres de configuration du code source. Cette approche garantit la flexibilité entre les environnements de développement, test et production, tout en renforçant la sécurité par l'isolation des informations sensibles.

3.2. Architecture Logicielle

3.2.1. Implémentation du Pattern MVC

L'architecture de l'application repose sur le pattern Model-View-Controller (MVC), un paradigme architectural éprouvé qui sépare clairement les responsabilités et favorise la maintenabilité du code. Cette approche structure l'application en trois couches distinctes, chacune ayant un rôle spécifique dans le traitement des données et la présentation des informations.

3.2.2. Description de la Couche Modèle

Les modèles constituent la couche de données de l'application et encapsulent toute la logique métier liée à la persistance et aux règles de gestion. Chaque modèle représente une entité métier spécifique du domaine académique : `Etudiant`, `Filiere`, `Parcour`, et leurs relations associées.

Ces modèles héritent de la classe `Eloquent` de Laravel, offrant une interface orientée objet pour interagir avec la base de données. Ils définissent les relations entre entités (one-to-many, many-to-many) et encapsulent les règles de validation des données. Par exemple, le modèle `Etudiant` gère non seulement les informations personnelles mais aussi les relations avec les parcours suivis et les résultats obtenus.

La logique métier complexe, comme le calcul des moyennes, la validation des prérequis de parcours, ou la génération des relevés de notes, est également encapsulée dans ces modèles. Cette approche garantit que les règles de gestion restent centralisées et cohérentes à travers toute l'application.

3.2.3. Description de la Couche Vue

Les vues utilisent le moteur de templates **Blade** de Laravel pour générer des interfaces utilisateur dynamiques et réactives. Blade offre une syntaxe épurée combinant HTML et directives PHP, permettant de créer des templates maintenables et réutilisables.

La couche vue se contente de présenter les données fournies par les contrôleurs, sans contenir de logique métier. Elle utilise les fonctionnalités avancées de Blade comme l'héritage de templates, les composants réutilisables, et les directives conditionnelles pour créer une interface cohérente.

L'intégration avec Tailwind CSS et Alpine.js s'effectue directement dans les templates Blade, créant des interfaces interactives tout en maintenant la séparation des préoccupations. Les composants Blade permettent de créer des éléments d'interface réutilisables (formulaires, tableaux, modales) qui standardisent l'expérience utilisateur.

3.2.4. Description de la Couche Contrôleur

Les contrôleurs orchestrent les interactions entre les modèles et les vues, gérant la logique applicative et les flux de données. Des contrôleurs spécialisés comme `StudentProfileController` et `AcademicResultsController` organisent les fonctionnalités par domaine métier étudiant, facilitant la maintenance et l'évolution du code.

Chaque contrôleur suit le principe de responsabilité unique, gérant un aspect spécifique de l'expérience étudiante. Le `StudentProfileController` s'occupe exclusivement de la gestion du profil étudiant : consultation et mise à jour des informations personnelles, gestion des paramètres de compte. L'`AcademicResultsController` centralise l'affichage des résultats académiques, des relevés de notes et du suivi du parcours.

Les contrôleurs implémentent principalement les opérations de lecture et de mise à jour limitée pour les données étudiantes, en utilisant les modèles pour récupérer les informations et en retournant les vues appropriées. Ils gèrent également la validation des données entrantes lors des modifications de profil, l'authentification des étudiants, et les autorisations d'accès aux informations personnelles.

3.2.5. Analyse des Avantages Architecturaux

Cette architecture MVC apporte plusieurs bénéfices substantiels au projet. La séparation des préoccupations facilite la maintenance en isolant les modifications dans leur couche respective. La testabilité est améliorée car chaque couche peut être testée indépendamment. La réutilisabilité du code est favorisée, particulièrement pour les modèles et les vues qui peuvent être partagés entre différentes fonctionnalités.

3.3. Conception de la Base de Données

3.3.1. Structure des Entités Principales

La conception de la base de données s'articule autour des entités fondamentales du domaine académique, structurées pour optimiser les performances tout en maintenant l'intégrité référentielle.

Cette architecture relationnelle reflète fidèlement les processus métier de l'institution éducative.

Table `etudiants`

La table `etudiants` constitue le cœur du système d'information, centralisant toutes les données relatives aux apprenants. Le champ `num_inscription` sert de clé primaire, garantissant l'unicité de chaque étudiant dans le système. Cette approche utilise un identifiant métier plutôt qu'un identifiant technique, facilitant les références externes et les imports de données.

La structure inclut les informations personnelles essentielles (nom, prénom, date de naissance, coordonnées), les données administratives (statut d'inscription, date d'entrée), et les informations académiques (niveau d'études, statut de redoublement). Des champs d'audit automatiques (created_at, updated_at) permettent le suivi des modifications pour la conformité réglementaire.

Tables `parcours` et `filieres`

La table `filieres` contient les grands domaines d'études disponibles dans l'institution, avec leurs caractéristiques spécifiques (durée, prérequis, modalités d'évaluation). Chaque filière possède un code unique permettant aux étudiants d'identifier clairement leur domaine d'études et facilitant les références dans le système.

La table `parcours` détaille les chemins académiques spécifiques au sein de chaque filière, accessibles en consultation par les étudiants pour comprendre leur progression académique. Elle établit une relation hiérarchique avec les filières via une clé étrangère, permettant aux étudiants de visualiser les parcours spécialisés ou les options disponibles dans leur filière. Cette structure préserve l'historique du parcours de chaque étudiant pour un suivi cohérent de sa progression.

3.3.2. Relations et Intégrité Référentielle

Le schéma relationnel implémente des contraintes d'intégrité référentielle strictes via des clés étrangères. La relation entre `etudiants` et `parcours` utilise une table de liaison `etudiant_parcours` pour supporter les inscriptions multiples ou les changements d'orientation, tout en permettant aux étudiants de consulter leur historique académique complet.

Les contraintes de clés étrangères sont configurées avec des actions de cascade appropriées pour préserver la cohérence des données. Cette architecture garantit que les étudiants peuvent toujours accéder à leur historique académique, même en cas d'évolution des structures de formation.

3.3.3. Stratégies d'Optimisation et de Performance

La conception décrit des index pour les recherches fréquentes. Le partitionnement par année académique est une piste d'optimisation ; sa mise en oeuvre et son intérêt nécessitent une validation sur des volumes représentatifs.

Les relations sont optimisées pour les opérations de lecture, particulièrement importantes pour l'affichage des informations étudiantes et les consultations de parcours. La dénormalisation contrô-

lée de certaines données calculées (moyennes, crédits obtenus) améliore les performances des requêtes de consultation tout en maintenant la cohérence via des mécanismes automatisés.

3.3.4. Considérations d'Évolutivité et de Maintenance

La structure de base de données est conçue pour supporter l'évolution des besoins métier sans modifications destructives. L'utilisation de champs JSON pour les données semi-structurées (configurations spécifiques de parcours, préférences utilisateur) offre la flexibilité nécessaire pour des extensions futures.

Les migrations Laravel versionnent les évolutions du schéma. Les procédures de sauvegarde et de restauration devront être adaptées à la criticité des données académiques et faire l'objet de tests documentés ; aucun résultat de restauration n'est présenté ici.

4. Mise en œuvre et Développement

Ce chapitre présente la mise en pratique des concepts théoriques abordés précédemment, en détaillant les aspects concrets du développement de l'application. Nous explorons l'architecture technique, les choix d'implémentation et les solutions apportées aux défis rencontrés lors du développement.

4.1. Développement Backend

4.1.1. Architecture des Modèles Eloquent

Le backend de l'application repose sur une architecture robuste utilisant les modèles Eloquent de Laravel. Ces modèles définissent non seulement la structure des données, mais également les relations complexes entre les différentes entités du système.

Le modèle `Etudiant` constitue le cœur de l'application, utilisant une approche particulière avec `num_inscription` comme clé primaire non auto-incrémentée. Cette décision technique permet de conserver la logique métier existante tout en offrant une identification unique et significative :

```
class Etudiant extends Authenticatable
{
    protected $primaryKey = 'num_inscription';
    public $incrementing = false;
    protected $keyType = 'string';

    public function filiere(): BelongsTo
    {
        return $this->belongsTo(Filiere::class);
    }

    public function actionsHistoriques(): HasMany
    {
        return $this->hasMany(ActionHistorique::class);
    }
}
```

Cette implémentation permet de maintenir la cohérence avec les systèmes existants de l'université tout en offrant la flexibilité nécessaire pour les fonctionnalités modernes.

4.1.2. Logique Métier dans les Contrôleurs

Les contrôleurs orchestrent la logique métier de l'application en gérant les interactions entre les utilisateurs et le système. Le `ParcourController` illustre parfaitement cette approche en combinant validation, logique conditionnelle et gestion d'état :

```
class ParcourController extends Controller
{
    public function index()
    {
        $etudiant = Auth::user();
        if ($etudiant->choix_confirme) {
            return redirect()->route('parcours.confirmation');
        }
        $filiere = Filiere::find($etudiant->filiere_id);
        if (!$filiere) {
            return redirect()->route('dashboard')
                ->with('error', 'Vous n\'avez pas de filière assignée.');
        }
    }
}
```

Cette implémentation démontre comment la logique métier est encapsulée dans les contrôleurs, assurant une séparation claire des responsabilités et facilitant la maintenance du code.

4.1.3. Système d'Authentification et Contrôle d'Accès

L'authentification utilise l'email académique comme identifiant principal. L'extrait suivant illustre une organisation possible du contrôle d'accès par middleware. La granularité des permissions et leur activation effective restent à vérifier, comme indiqué dans les annexes D et E.

```
class AccessControlMiddleware
{
    public function handle(Request $request, Closure $next): Response
    {
        if (!Auth::check()) {
            return redirect()->route('login');
        }

        // Vérification des privilèges basée sur le rôle utilisateur
        $user = Auth::user();
        if (!$this->hasRequiredPermissions($user, $request->route())) {
            return redirect()->route('dashboard')
                ->with('error', 'Accès refusé.');
        }
        return $next($request);
    }
```

```
    private function hasRequiredPermissions($user, $route): bool
    {
        // Logique de vérification des permissions
        return $this->evaluateUserPermissions($user, $route);
    }
}
```

Cette implémentation démontre une approche de développement évolutive, où l'architecture de sécurité est conçue pour s'adapter aux différents niveaux d'autorisation requis par l'organisation. Le système de contrôle d'accès permet une gestion flexible des privilèges sans compromettre la sécurité globale de l'application.

4.1.4. Gestion de la Base de Données avec les Migrations

Les migrations Laravel assurent l'évolution contrôlée de la structure de la base de données. La migration de création de la table `etudiants` illustre l'attention portée à l'intégrité référentielle :

```
Schema::create('etudiants', function (Blueprint $table) {
    $table->char('num_inscription', 10)->primary();
    $table->string('nom');
    $table->string('prenom');
    $table->string('email_academique')->unique();
    $table->timestamps();
    $table->foreign('filiere_id')->references('code_deug')->on('filieres');
});
```

Cette structure garantit la cohérence des données tout en permettant une évolution progressive du schéma de base de données selon les besoins fonctionnels.

4.1.5. Services Métier et Architecture Modulaire

L'architecture de l'application privilégie une approche modulaire où les services métier encapsulent la logique complexe. Le `AttestationPdfService` représente un exemple parfait de cette philosophie, gérant la génération sécurisée de documents PDF avec des éléments de sécurité intégrés comme les QR codes et les filigranes pour prévenir la falsification. Cette approche favorise la réutilisabilité du code et facilite la maintenance des fonctionnalités critiques.

4.2. Développement Frontend

4.2.1. Conception de l'Expérience Utilisateur

L'interface utilisateur a été conçue avec un focus sur l'intuitivité et la réactivité, répondant aux besoins spécifiques des étudiants dans un contexte académique. L'utilisation de Blade comme moteur de templates permet de créer des interfaces dynamiques qui s'adaptent au contexte de chaque utilisateur.

L'intégration d'Alpine.js apporte une couche d'interactivité moderne sans la complexité d'un framework JavaScript complet. Cette approche progressive permet d'enrichir l'expérience utilisateur tout en maintenant la simplicité de développement et la performance du côté serveur.

4.2.2. Stratégie de Design et Justifications Techniques

Le choix de Tailwind CSS comme framework de styles répond à plusieurs objectifs techniques et esthétiques. Cette approche utility-first permet une cohérence visuelle à travers l'application tout en offrant la flexibilité nécessaire pour s'adapter aux besoins spécifiques de chaque interface.

Les décisions de design privilégient la clarté et l'accessibilité, avec une hiérarchie visuelle claire qui guide naturellement l'utilisateur à travers les différentes étapes de son parcours. La palette de couleurs et la typographie ont été sélectionnées pour assurer une lisibilité optimale et renforcer l'identité institutionnelle.

4.2.3. Architecture des Composants Blade

La structure des composants Blade favorise la réutilisabilité et la maintenabilité du code frontend. Bien que les composants spécifiques ne soient pas visibles dans l'architecture actuelle, l'approche modulaire permet de créer des éléments d'interface cohérents comme les boutons, les cartes d'information et les formulaires.

Cette architecture componentisée facilite les mises à jour d'interface et assure une expérience utilisateur homogène à travers toutes les pages de l'application. Chaque composant encapsule sa logique de présentation et ses styles, permettant une maintenance simplifiée et une évolution contrôlée de l'interface.

4.2.4. Responsive Design et Accessibilité

L'utilisation de Tailwind CSS facilite considérablement l'implémentation d'un design responsive qui s'adapte naturellement aux différentes tailles d'écran. Les classes utilitaires permettent de définir des comportements spécifiques pour mobile, tablette et desktop sans complexité additionnelle.

La conception prend en compte les contrastes, la navigation au clavier et la structure sémantique HTML. La conformité WCAG et l'utilisation avec les technologies d'assistance restent à vérifier, comme indiqué dans les perspectives d'amélioration.

4.2.5. Éléments d'Interface Clés

L'interface de connexion privilégie la simplicité avec un formulaire épuré qui met l'accent sur les informations essentielles. Le design minimaliste réduit les distractions et facilite l'authentification rapide des utilisateurs.

Le tableau de bord adapte son contenu selon le rôle de l'utilisateur, présentant les informations et actions pertinentes de manière hiérarchisée. Cette personnalisation améliore l'efficacité de

navigation et réduit la charge cognitive.

Les formulaires de sélection de parcours intègrent une logique de validation en temps réel, guidant l'utilisateur dans ses choix et prévenant les erreurs avant la soumission. L'interface réactive fournit un feedback immédiat sur la validité des sélections.

Les pages de profil offrent une vue complète des informations académiques avec des options d'édition contextuelles, permettant aux utilisateurs de maintenir leurs données à jour facilement.

4.3. Implémentation des Fonctionnalités Clés

4.3.1. Gestion du Profil Étudiant

La fonctionnalité de gestion du profil constitue l'un des piliers de l'expérience utilisateur étudiant. L'implémentation technique repose sur une validation robuste des données et une interface intuitive pour la mise à jour des informations personnelles.

Implémentation Technique Le système utilise les modèles Eloquent pour gérer les interactions avec la base de données, assurant l'intégrité des données grâce à des règles de validation strictes. Les formulaires intègrent une validation côté client et serveur pour une expérience utilisateur fluide.

Défis Résolus La synchronisation entre les informations académiques officielles et les données modifiables par l'étudiant représentait un défi majeur. La solution implémentée distingue clairement les champs modifiables des données officielles, tout en maintenant la cohérence globale du profil.

4.3.2. Sélection et Confirmation de Parcours

Le processus de sélection de parcours constitue le cœur fonctionnel de l'application, nécessitant une orchestration complexe entre validation d'éligibilité, présentation des options et confirmation des choix.

Implémentation Technique Le `ParcourController` gère l'ensemble du processus, depuis la vérification des conditions d'éligibilité jusqu'à la confirmation finale des choix. Le système utilise des vues Blade dynamiques qui s'adaptent aux options disponibles pour chaque étudiant selon sa filière et son niveau.

Défis Résolus La gestion des cas particuliers comme les parcours personnalisés ou les situations d'exception a nécessité une logique conditionnelle sophistiquée. Le système gère également les choix par défaut et les modifications ultérieures tout en maintenant l'intégrité des données académiques.

4.3.3. Génération d'Attestations Sécurisées

La génération d'attestations représente l'aboutissement du parcours étudiant, nécessitant un niveau de sécurité élevé pour prévenir toute falsification.

Implémentation technique : AttestationPdfService utilise DomPDF pour générer les attestations. L'exemple en annexe montre un QR code et une empreinte de vérification. La présence de filigranes invisibles n'est pas établie par les éléments présentés.

Défis : la génération de PDF et leur vérification sont séparées dans la conception. Le QR code et l'empreinte participent au contrôle du document ; le rapport ne présente pas de mesure de rapidité ni de validation complète de ce mécanisme.

4.3.4. Historique et Traçabilité des Actions

L'historique conserve les actions enregistrées pour faciliter le suivi et le diagnostic. Son exhaustivité n'est pas mesurée dans ce rapport.

Implémentation Technique Basé sur le package `spatie/laravel-activitylog`, le système enregistre automatiquement toutes les actions significatives. Le modèle `ActionHistorique` utilise des castings pour gérer les données additionnelles sous forme de tableau, permettant un stockage flexible des métadonnées.

Défis : le rapport décrit des observers Eloquent et des index pour l'historique. Leur effet sur les temps de réponse n'a pas été mesuré ici. Le caractère asynchrone du traitement doit être vérifié dans l'implémentation avant toute affirmation de performance.

4.3.5. Interface de Suivi des Demandes

Pour compléter l'expérience utilisateur, une interface de suivi permet aux étudiants de visualiser l'état de leurs demandes et de comprendre les étapes restantes.

Implémentation Technique Cette fonctionnalité utilise une approche basée sur les états, où chaque demande progresse à travers des étapes définies. L'interface présente ces informations de manière visuelle avec des indicateurs de progression clairs.

Défis : la mise à jour du statut vise à informer l'étudiant tout en limitant les sollicitations du serveur. L'existence d'un cache ne suffit pas à démontrer un suivi en temps réel ; la fraîcheur des informations et les coûts restent à mesurer.

Ces éléments documentent les solutions retenues pour les parcours principaux. Les objectifs globaux de sécurité et de performance nécessitent des vérifications complémentaires.

5. Stratégie de Tests et Validation

La démarche qualité s'appuie sur les outils de test de Laravel, les données de test et la validation des entrées. Ce chapitre présente les premiers exemples et la stratégie envisagée. Il ne constitue pas un rapport de certification, de couverture mesurée ou de disponibilité en production.

5.1. Tests Automatisés avec PHPUnit

5.1.1. Architecture de Tests

PHPUnit a été retenu pour son intégration avec Laravel. Les exemples présentés constituent une première base de tests automatisés ; l'extension de la suite et la mesure de sa couverture restent à poursuivre.

La stratégie de tests s'articule autour de deux niveaux complémentaires :

Tests Unitaires : Ces tests se concentrent sur la validation des composants individuels de l'application, notamment les modèles, les services et les utilitaires. Ils permettent de vérifier que chaque unité de code fonctionne correctement de manière isolée, garantissant ainsi la fiabilité des briques de base du système.

Tests Fonctionnels : Ces tests valident le comportement de l'application dans son ensemble, en simulant les interactions utilisateur réelles. Ils vérifient que les différents composants fonctionnent correctement ensemble et que les cas d'usage métier sont respectés.

5.1.2. Exemples Concrets d'Implémentation

L'exemple suivant vérifie la réponse de la route principale dans l'environnement de test Laravel :

```
public function test_the_application_returns_a_successful_response(): void
{
    $response = $this->get('/');
    $response->assertStatus(200);
}
```

Ce test vérifie que la route principale renvoie HTTP 200 dans l'environnement de test Laravel. Il ne mesure ni la disponibilité en production ni la configuration d'un serveur web déployé.

Avantages de l'Approche Choisie

Cette stratégie de tests automatisés présente plusieurs avantages significatifs pour le projet :

- Détection des régressions : les tests aident lorsqu'ils sont exécutés. Leur automatisation en intégration continue reste une évolution envisagée.

— **Documentation Vivante** : Les tests servent de documentation technique, illustrant le comportement attendu de chaque fonctionnalité.

- Validation des versions : la suite existante constitue une première base à compléter avant de conclure à la fiabilité d'un déploiement.

5.2. Données de Test avec les Seeders

5.2.1. Stratégie de Peuplement des Données

Les seeders constituent un élément essentiel de la stratégie de test, permettant de créer un environnement de développement et de test cohérent et reproductible. Cette approche garantit que tous les développeurs travaillent avec des jeux de données identiques, facilitant ainsi la collaboration et la résolution des problèmes.

5.2.2. Conception des Scénarios de Test

L'extrait EtudiantSeeder ci-dessous utilise des données de démonstration. Il illustre un scénario de test, sans prétendre couvrir tous les cas possibles. Aucun identifiant de compte réel n'est nécessaire à cet exemple.

```
DB::table('etudiants')->insert([
    [
        'num_inscription' => 'DEMO-001',
        'nom_fr' => 'Dupont',
        'prenom_fr' => 'Marie',
        'email_academique' => 'marie.dupont@example.com',
        'password' => Hash::make('DEMO_PASSWORD_ONLY'),
        'filiere_id' => 'INF',
        'parcour_id' => 'INF01',
        'choix_confirme' => true,
        'date_choix' => now()->subDays(30),
    ],
    // Autres étudiants...
]);
```

Les scénarios de test représentent plusieurs états étudiants et doivent être complétés pour élargir la couverture des fonctionnalités :

— **Étudiants avec Choix Confirmés** : Données représentant des utilisateurs ayant finalisé leur sélection de parcours, permettant de tester les fonctionnalités de consultation et de génération d'attestations.

— **Étudiants sans Choix** : Profils d'utilisateurs n'ayant pas encore effectué de sélection, essentiels pour valider les workflows de sélection de parcours et les interfaces de choix.

— **Étudiants en Attente** : Données représentant des utilisateurs dont le statut nécessite une mise à jour automatique ou une synchronisation avec les systèmes externes, permettant de tester les processus de traitement différé et les états transitoires.

— **Étudiants avec Données Incomplètes** : Scénarios incluant des profils partiellement remplis pour tester la robustesse de l'application face aux données manquantes et les mécanismes de validation.

Ces scénarios représentent plusieurs états étudiants : choix confirmé, sélection à effectuer, attente et données incomplètes. Ils contribuent à la validation des parcours couverts, sans constituer une vérification exhaustive de tous les cas d'usage.

Diversité des Scénarios Couverts

Cette approche permet de tester diverses situations réelles :

— **Étudiants avec Choix Confirmés** : Pour valider les processus d'inscription définitive

— **Différentes Filières et Parcours** : Pour s'assurer de la flexibilité du système

— **Dates Variables** : Pour valider la gestion temporelle des inscriptions

Ces scénarios ne démontrent pas le comportement dans toutes les conditions de production.

5.3. Validation des Données

5.3.1. Sécurisation par FormRequest

La validation des données constitue un rempart essentiel contre les erreurs utilisateur et les tentatives malveillantes. L'utilisation des classes FormRequest de Laravel permet de centraliser et de standardiser cette validation, garantissant une approche cohérente à travers toute l'application.

5.3.2. Implémentation Rigoureuse

L'exemple du LoginRequest démontre cette approche méthodique :

```
public function rules(): array
{
    return [
        'email' => ['required', 'string', 'email'],
        'password' => ['required', 'string'],
    ];
}
```

Cette validation garantit plusieurs aspects de sécurité :

— **Présence Obligatoire** : Les champs critiques doivent être fournis

— **Format Correct** : L'email doit respecter la syntaxe standard

— **Type de Données** : Les valeurs doivent être des chaînes de caractères

Avantages de cette Approche

Cette stratégie de validation présente plusieurs bénéfices :

- Validation : rejet des données absentes ou mal formées selon les règles définies.

— **Expérience Utilisateur Améliorée** : Messages d'erreur clairs et contextuel

— **Maintenance Simplifiée** : Centralisation des règles de validation

— **Réutilisabilité** : Les règles peuvent être partagées entre différents contextes

5.4. Gestion des Erreurs et Défis

5.4.1. Architecture de Gestion d'Erreurs

Laravel fournit une gestion centralisée des exceptions. Cette organisation facilite le traitement des erreurs et leur journalisation ; l'exhaustivité des événements enregistrés dépend de la configuration et des chemins effectivement testés.

5.4.2. Stratégie de Journalisation

Bien que les logs spécifiques n'aient pas été détaillés dans la documentation technique, l'infrastructure Laravel permet une journalisation efficace des erreurs. Cette approche facilite :

— **Diagnostic Rapide** : Identification immédiate des problèmes en production

— **Traçabilité Complète** : Historique détaillé des erreurs et de leur contexte

— **Amélioration Continue** : Analyse des patterns d'erreurs pour optimiser l'application

5.4.3. Défis Techniques Rencontrés

Le développement de ce système a présenté plusieurs défis techniques significatifs :

Gestion des Relations Complexes : La modélisation des relations entre étudiants, filières, parcours et options a nécessité une réflexion approfondie pour éviter les incohérences de données et garantir l'intégrité référentielle.

Optimisation des Performances : Les requêtes impliquant de multiples jointures entre les tables ont requis une attention particulière pour maintenir des temps de réponse acceptables, notamment lors de l'affichage des listes d'étudiants avec leurs informations complètes.

Validation Métier Complexe : L'implémentation des règles métier spécifiques au domaine académique, comme la validation des prérequis ou la gestion des conflits d'horaires, a nécessité une approche méthodique et exhaustive.

5.4.4. Solutions Adoptées

Face à ces défis, plusieurs stratégies ont été mises en place :

— **Optimisation des Requêtes** : Utilisation de l'eager loading et de requêtes optimisées pour réduire le nombre d'accès à la base de données

— **Cache Intelligent** : Mise en cache des données fréquemment consultées pour améliorer les performances

— **Validation Multicouche** : Combinaison de validations côté client et serveur pour une expérience utilisateur fluide

5.5. Conclusion du Chapitre

Les tests PHPUnit, les seeders, la validation des entrées et la gestion des erreurs contribuent à la validation des scénarios couverts. Ils constituent une première démarche qualité ; la couverture mesurée, les performances et les conditions de production restent à approfondir.

Cette démarche qualité ne se limite pas à la phase de développement initial, mais constitue un investissement à long terme facilitant la maintenance évolutive et corrective de l'application. Elle assure également une meilleure collaboration au sein de l'équipe de développement et une confiance accrue lors des déploiements en production.

L'approche adoptée démontre une compréhension approfondie des enjeux de qualité logicielle et une volonté d'appliquer les meilleures pratiques de l'industrie, positionnant ce projet comme un exemple de développement professionnel et méthodique.

Troisième partie

Bilan et Perspectives

6. Évaluation du Projet et Perspectives d'Évolution

Au terme de cette phase de développement, il convient de dresser un bilan objectif du travail accompli et d'identifier les axes d'amélioration pour l'avenir. Ce chapitre présente une évaluation critique du projet, analysant sa conformité aux objectifs initiaux, ses réussites et ses limitations, avant de proposer une vision stratégique pour son évolution future.

6.1. Évaluation du Projet

6.1.1. Conformité aux Objectifs Initiaux

Les livrables documentent la mise en oeuvre des principaux parcours étudiants. Cette correspondance fonctionnelle avec les besoins ne démontre pas l'atteinte des objectifs chiffrés de disponibilité, de performance, d'adoption ou de satisfaction.

Fonctionnalités documentées : consultation du profil, sélection et confirmation d'un parcours selon les règles décrites, puis génération d'une attestation. Les captures illustrent ces étapes ; elles ne démontrent pas que l'ensemble de la gestion étudiante ou des exigences initiales a été livré.

La génération d'attestations PDF matérialise le choix du parcours. L'historique des actions enregistrées fournit des éléments de suivi. L'exhaustivité de cet historique et les conditions d'usage réel doivent être vérifiées séparément.

6.1.2. Points Forts du Projet

Excellence Technique : L'architecture technique repose sur des fondations solides qui témoignent d'une approche professionnelle du développement. L'utilisation du framework Laravel représente un choix stratégique judicieux, offrant robustesse, sécurité et maintenabilité. Cette plateforme éprouvée garantit une base stable pour les évolutions futures et facilite l'intégration de nouvelles fonctionnalités.

Les mécanismes décrits comprennent l'authentification, la gestion des sessions, le hachage des mots de passe et les protections CSRF/XSS de Laravel. Leur configuration effective en production et les durcissements listés en annexe E restent à vérifier.

Le schéma décrit les entités et leurs relations. Les contraintes référentielles et Eloquent structurent les accès aux données. L'effet des index et le comportement avec les volumes attendus restent à mesurer sur un jeu de données représentatif.

Interface réalisée : les captures présentent une navigation entre profil, résultats, sélection et attestation. Cette organisation vise à faciliter le parcours étudiant ; aucune étude utilisateur ni mesure de réduction des erreurs n'est présentée.

L'intégration harmonieuse des différentes fonctionnalités crée un ensemble cohérent où chaque module communique efficacement avec les autres. Cette approche holistique évite la fragmentation et assure une cohérence dans l'expérience globale d'utilisation du système.

6.1.3. Limitations et Axes d'Amélioration

Accessibilité Numérique : Bien que l'interface soit fonctionnelle et ergonomique, des améliorations significatives peuvent être apportées en matière d'accessibilité. L'implémentation des standards WCAG (Web Content Accessibility Guidelines) permettrait d'ouvrir la plateforme à un public plus large, incluant les utilisateurs en situation de handicap. Cette dimension inclusive représente un enjeu important dans le contexte de l'enseignement supérieur.

Documentation Technique : La documentation destinée aux développeurs mériterait d'être étoffée pour faciliter la maintenance et les évolutions futures. Une documentation complète incluant l'architecture système, les guides de déploiement et les procédures de développement réduirait significativement le temps d'appropriation pour de nouveaux contributeurs au projet.

Tests et Couverture : Bien que des tests aient été implémentés, l'extension de la couverture de tests, notamment pour les cas d'usage complexes et les scénarios d'erreur, renforcerait la robustesse du système. L'ajout de tests d'intégration et de tests de performance compléterait avantageusement la stratégie qualité actuelle.

6.1.4. Performance et Robustesse

État des mesures : le rapport présente le fonctionnement de l'interface, mais aucun protocole ni résultat chiffré ne permet de qualifier les performances, la satisfaction ou l'adoption. Des tests de charge documentés restent nécessaires.

Analyse des Limites : Cependant, des tests de charge sous contrainte forte n'ont pas encore été menés de manière exhaustive. Cette évaluation permettrait d'identifier les goulots d'étranglement potentiels et de dimensionner l'infrastructure nécessaire pour gérer les pics d'activité, notamment lors des périodes d'inscription intensive.

6.2. Perspectives d'Évolution

6.2.1. Extensions Fonctionnelles Stratégiques

Développement d'outils de pilotage et de reporting : L'amélioration des capacités de suivi et d'analyse représente la priorité évolutive majeure. Cette extension permettrait aux gestionnaires académiques d'obtenir une vision claire de l'ensemble du processus d'inscription grâce à des exports automatisés et des rapports personnalisés. Les fonctionnalités envisagées incluent la génération de fichiers Excel détaillés par filière, l'export PDF de synthèses statistiques, et l'envoi automatique de rapports périodiques par email.

Le système pourrait générer des tableaux de bord sous forme de documents PDF interactifs offrant une vision synthétique des inscriptions par filière, l'évolution temporelle des choix et l'identification des tendances. Ces rapports automatisés, programmables via des tâches cron, s'avéreraient précieux pour l'optimisation de l'offre pédagogique et la planification des ressources. Les gestionnaires recevraient quotidiennement ou hebdomadairement des fichiers de synthèse téléchargeables depuis un répertoire sécurisé.

Notifications et Communication : L'implémentation d'un système de notifications en temps réel transformerait l'expérience utilisateur en permettant un suivi actif des démarches. Les étudiants pourraient recevoir des alertes pour les échéances importantes, les confirmations de choix ou les modifications de planning. Cette fonctionnalité s'étendrait naturellement vers un système de messagerie intégré facilitant la communication bidirectionnelle et l'envoi d'informations personnalisées.

Analyses et Reporting Avancés : Le développement de capacités analytiques permettrait d'exploiter la richesse des données collectées. Des outils de génération de rapports automatisés produiraient des exports détaillés en formats multiples (CSV, Excel, PDF) contenant des insights précieux sur les tendances d'inscription, les préférences étudiantes et l'efficacité des parcours proposés. Ces analyses, disponibles sous forme de fichiers programmés et téléchargeables, supporteraient la prise de décision stratégique et l'amélioration continue de l'offre pédagogique.

L'ensemble de ces extensions privilégierait une approche basée sur l'automatisation et la génération de documents, minimisant les interventions manuelles tout en maximisant l'accès aux informations stratégiques pour les gestionnaires académiques.

6.2.2. Améliorations Techniques et Architecturales

Refactoring et Modularité : L'évolution naturelle du projet conduirait vers une architecture plus modulaire, facilitant la maintenance et l'extension des fonctionnalités. La restructuration du code selon les principes SOLID et l'implémentation de patterns architecturaux avancés amélioreraient la lisibilité et la maintenabilité du système.

Cette refactorisation inclurait la séparation claire des préoccupations, l'injection de dépendances et l'implémentation de services métier réutilisables. Ces améliorations techniques créeraient une base solide pour les développements futurs et réduiraient les coûts de maintenance à long terme.

Architecture Microservices : Pour répondre aux enjeux de scalabilité et de flexibilité, une transition progressive vers une architecture microservices pourrait être envisagée. Cette approche

permettrait de décomposer l'application monolithique actuelle en services indépendants, chacun responsable d'un domaine métier spécifique.

Les avantages de cette transformation incluent une meilleure scalabilité horizontale, la possibilité de déploiements indépendants et une résilience accrue du système global. Chaque service pourrait être développé, testé et déployé de manière autonome, accélérant les cycles de développement et réduisant les risques de régression.

Optimisation des Performances : L'implémentation de stratégies de cache avancées et l'optimisation des requêtes de base de données constituent des axes d'amélioration prioritaires. L'utilisation de Redis pour la mise en cache des données fréquemment consultées et l'implémentation de mécanismes de cache applicatif amélioreraient significativement les temps de réponse.

6.2.3. Innovation et Technologies Émergentes

Support Mobile et PWA : Le développement d'une application mobile native ou d'une Progressive Web App (PWA) répondrait aux attentes croissantes de mobilité des utilisateurs. Cette extension permettrait aux étudiants de gérer leurs inscriptions depuis n'importe quel appareil, améliorant l'accessibilité et la flexibilité d'usage.

Intelligence Artificielle et Personnalisation : L'intégration de fonctionnalités d'intelligence artificielle ouvrirait de nouvelles perspectives pour la personnalisation de l'expérience utilisateur. Des algorithmes de recommandation pourraient suggérer des parcours adaptés au profil de chaque étudiant, optimisant ainsi l'orientation académique et réduisant les risques d'échec.

Intégration Ecosystem : Le développement d'APIs robustes faciliterait l'intégration avec d'autres systèmes d'information de l'établissement (ENT, systèmes de gestion des emplois du temps, plateformes pédagogiques). Cette interopérabilité créerait un écosystème numérique cohérent et efficace.

6.2.4. Scalabilité et Infrastructure

Architecture Cloud-Native : La migration vers une infrastructure cloud native représente une évolution stratégique majeure. Cette transformation permettrait de bénéficier de l'élasticité du cloud, de la haute disponibilité et des services managés pour optimiser les coûts et améliorer la performance.

L'utilisation de conteneurs Docker et d'orchestrateurs comme Kubernetes faciliterait le déploiement et la gestion de l'application à grande échelle. Cette approche moderne garantirait une scalabilité automatique en fonction de la charge et une résilience élevée face aux pannes.

Monitoring et Observabilité : L'implémentation d'outils de monitoring avancés et de solutions d'observabilité permettrait un suivi proactif de la santé du système. Ces outils fourniraient des métriques détaillées sur les performances, la disponibilité et l'usage, supportant une approche DevOps mature.

6.3. Conclusion Stratégique

Le projet met en oeuvre les principaux parcours décrits dans le rapport, avec une architecture Laravel et des interfaces documentées. Les objectifs chiffrés et les conditions d'exploitation doivent encore être évalués ; ils ne sont pas assimilés à des résultats obtenus.

Les évolutions envisagées devront être accompagnées de tests et de mesures pour vérifier leur effet et limiter les régressions. La stabilité opérationnelle ne peut être garantie par la seule démarche itérative.

Ce projet illustre parfaitement la capacité à mener un développement logiciel professionnel, en combinant excellence technique, compréhension métier et vision stratégique. Il constitue un fondement robuste pour les développements futurs et témoigne d'une maîtrise des enjeux du développement logiciel moderne.

7. Bilan Personnel et Compétences Acquises

Ce stage au sein de la stratégie de transformation numérique de l'Université Sidi Mohamed Ben Abdellah a représenté une expérience formatrice exceptionnelle qui m'a permis de développer un ensemble de compétences techniques et transversales essentielles pour ma future carrière d'ingénieur logiciel. À travers le développement d'un portail étudiant robuste construit sur Laravel 12, j'ai pu non seulement consolider mes connaissances théoriques, mais également acquérir une expérience pratique significative dans la conception et l'implémentation de systèmes d'information complexes destinés à moderniser les processus académiques.

7.1. Compétences Techniques Acquises

7.1.1. Compréhension des Principes Architecturaux Avancés

Ce projet m'a permis de développer une **compréhension approfondie des principes d'architecture logicielle moderne**. J'ai appris à utiliser efficacement le pattern MVC de Laravel en respectant la séparation des responsabilités, où chaque couche (Modèle, Vue, Contrôleur) a un rôle clairement défini.

La conception de la base de données illustre ma maîtrise des **contraintes d'intégrité référentielle**, avec l'utilisation de clés étrangères sophistiquées (`onUpdate('cascade')`, `onDelete('restrict')`) et de clés primaires métiers non-incrémentales pour respecter la logique académique existante.

7.1.2. Application Systématique des Patterns de Conception

J'ai mis en pratique plusieurs **patterns de conception essentiels** dans un contexte professionnel :

- **Service Pattern** : pour encapsuler la logique métier complexe, notamment avec `AttestationPdfServic` qui centralise toute la logique de génération de documents officiels.
- **Repository Pattern** : implicite à travers les modèles Eloquent pour abstraire l'accès aux données et améliorer la testabilité.
- **Middlewares personnalisés** : pour gérer l'authentification et l'autorisation de manière transversale.
- **Composants Blade réutilisables** : pour standardiser l'interface utilisateur et réduire la duplication de code.

7.1.3. Développement de Composants Réutilisables

L'implémentation de composants Blade comme le système d'alertes contextuelle m'a sensibilisé à l'importance de la **réutilisabilité** et de la **maintenabilité** du code frontend. Ces composants acceptent des paramètres dynamiques et ajustent automatiquement leur comportement, garantissant une cohérence visuelle à travers l'application.

Ces patterns architecturaux m'ont permis de créer une architecture extensible et maintenable, facilitant les évolutions futures du système et la collaboration en équipe.

7.1.4. Vision Systémique

Au-delà des aspects techniques, j'ai développé une **vision systémique** qui me permet de concevoir des solutions en tenant compte de l'ensemble des contraintes techniques, fonctionnelles et de maintenance. Cette approche holistique est devenue un atout majeur dans ma démarche de développement.

Conclusion du Chapitre

Ce stage au cœur de la transformation numérique universitaire a représenté bien plus qu'une simple application pratique de connaissances théoriques. Il m'a permis de développer une **méthode de travail rigoureuse et professionnelle**, d'acquérir une **autonomie technique confirmée** face aux problèmes complexes de l'ingénierie logicielle, et de comprendre les **enjeux stratégiques du développement d'applications métier** dans un contexte institutionnel.

L'expérience de concevoir et développer un portail destiné aux étudiants m'a sensibilisé aux enjeux de performance, de sécurité et de scalabilité. L'architecture Laravel 12, la génération d'attestations et la traçabilité des actions m'ont donné une vision concrète des défis du développement d'une application métier. Le rapport ne fournit pas de mesure du nombre d'utilisateurs réels.

Les compétences acquises - allant de la maîtrise technique de l'écosystème Laravel/PHP 8.2 à la compréhension des patterns architecturaux avancés - constituent désormais un **socle technique solide** pour ma future carrière d'ingénieur logiciel. Cette expérience m'a également révélé l'importance cruciale de la **qualité du code**, de la **documentation technique**, et de la **collaboration** dans les projets informatiques d'envergure.

Au-delà des aspects purement techniques, ce projet m'a fait prendre conscience de l'**impact social** du développement logiciel. Contribuer à moderniser et fluidifier le parcours académique des étudiants m'a montré que l'ingénierie logicielle peut être un vecteur de transformation positive au service des usagers.

Je ressors de cette expérience avec une **confiance renforcée** dans ma capacité à concevoir, développer et maintenir des systèmes logiciels complexes, ainsi qu'une **vision stratégique** des défis et des opportunités qui caractérisent le métier d'ingénieur logiciel dans le contexte de la transformation numérique des organisations.

Bibliographie

[1] Robert C. Martin. *Clean Code : A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008.

[2] Martin Fowler. *Refactoring : Improving the Design of Existing Code*, 2e éd. Addison-Wesley Professional, 2018.

[3] Eric Evans. *Domain-Driven Design : Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2003.

[4] Roger S. Pressman. *Software Engineering : A Practitioner's Approach*, 9e éd. McGraw-Hill Education, 2019.

[5] Steve Krug. *Don't Make Me Think : A Common Sense Approach to Web Usability*, 3e éd. New Riders, 2014.

[6] Jakob Nielsen. *Designing Web Usability*. New Riders Publishing, 2000.

[7] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 7e éd. PMI Publications, 2021.

[8] Laravel LLC. *Laravel Framework Documentation Version 12.x*, 2024. URL : https://laravel.com/docs/12.x. (Consulté le 15 mai 2024).

[9] Tailwind Labs. *Tailwind CSS Documentation*, 2024. URL : https://tailwindcss.com/docs. (Consulté le 10 mai 2024).

[10] Alpine.js Team. *Alpine.js Documentation Version 3.x*, 2024. URL : https://alpinejs.dev/. (Consulté le 8 mai 2024).

[11] Oracle Corporation. *MySQL 8.0 Reference Manual*, 2024. URL : https://dev.mysql.com/doc/refman/8.0/en/. (Consulté le 12 mai 2024).

[12] OWASP Foundation. *OWASP Top 10 : 2021*, 2021. URL : https://owasp.org/Top10/. (Consulté le 25 avril 2024).

[13] Taylor Otwell. "Laravel 12 Release Notes". *Laravel News*, 2024. URL : https://laravel-news.com/laravel-12. (Consulté le 2 mai 2024).

Quatrième partie

Annexes

A. Architecture Technique Détaillée

A.1. Diagramme de l'Architecture Système

Schéma de conception ; il ne prouve pas le déploiement de chaque service représenté.

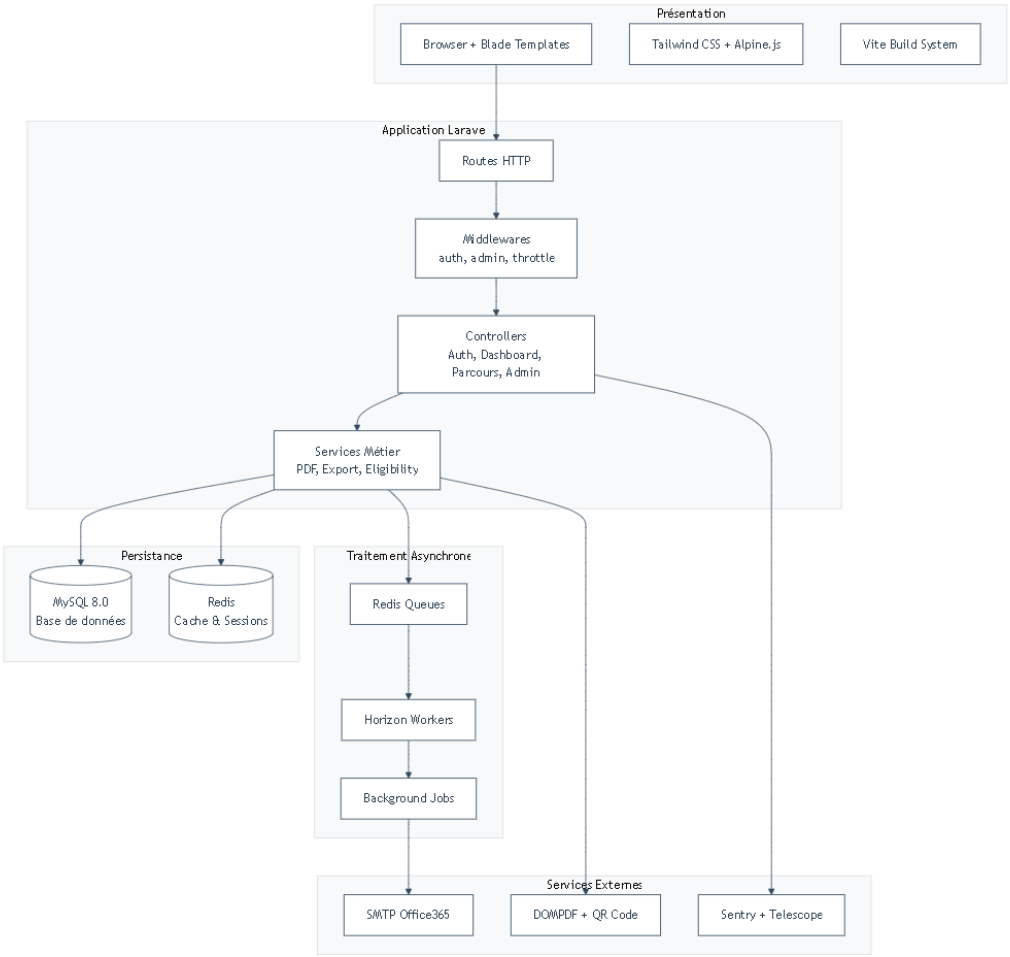


FIGURE A.1 – Diagramme de l'architecture technique globale

A.2. Structure des Dossiers du Projet

Cette section présente l'organisation du projet et les responsabilités des dossiers principaux.

A.2.1. Structure Principale

```
gestion-parcours-usmba/
 app/                    # Code source de l'application
```

```
    Console/          # Commandes Artisan personnalisées
    Http/             # Contrôleurs, Middleware, Requêtes de formulaire
        Controllers/
            Auth/
            ParcourController.php
            DashboardController.php
        Middleware/
        Requests/
    Models/           # Modèles Eloquent
        Etudiant.php
        Filiere.php
        Parcour.php
        ActionHistorique.php
    Providers/        # Fournisseurs de services
    Traits/           # Traits réutilisables
    View/             # Composants de vue
...
```

A.2.2. Dossiers Clés et Leur Rôle

app/Http/Controllers Gère la logique métier. Organisé par fonctionnalité (Auth, Etudiant, etc.).

app/Models Contient les modèles Eloquent et les relations entre eux.

database/migrations Définit la structure de la base de données et l'historique des modifications.

resources/views Contient les vues Blade, organisées par fonctionnalité.

routes/web.php Définit les routes de l'application et leur protection.

A.2.3. Fichiers de Configuration Importants

- `.env` : Variables d'environnement.
- `composer.json` : Dépendances PHP.
- `package.json` : Dépendances JavaScript.
- `vite.config.js` : Configuration de Vite pour les assets.

B. Modèle de Base de Données

B.1. Schéma Relationnel

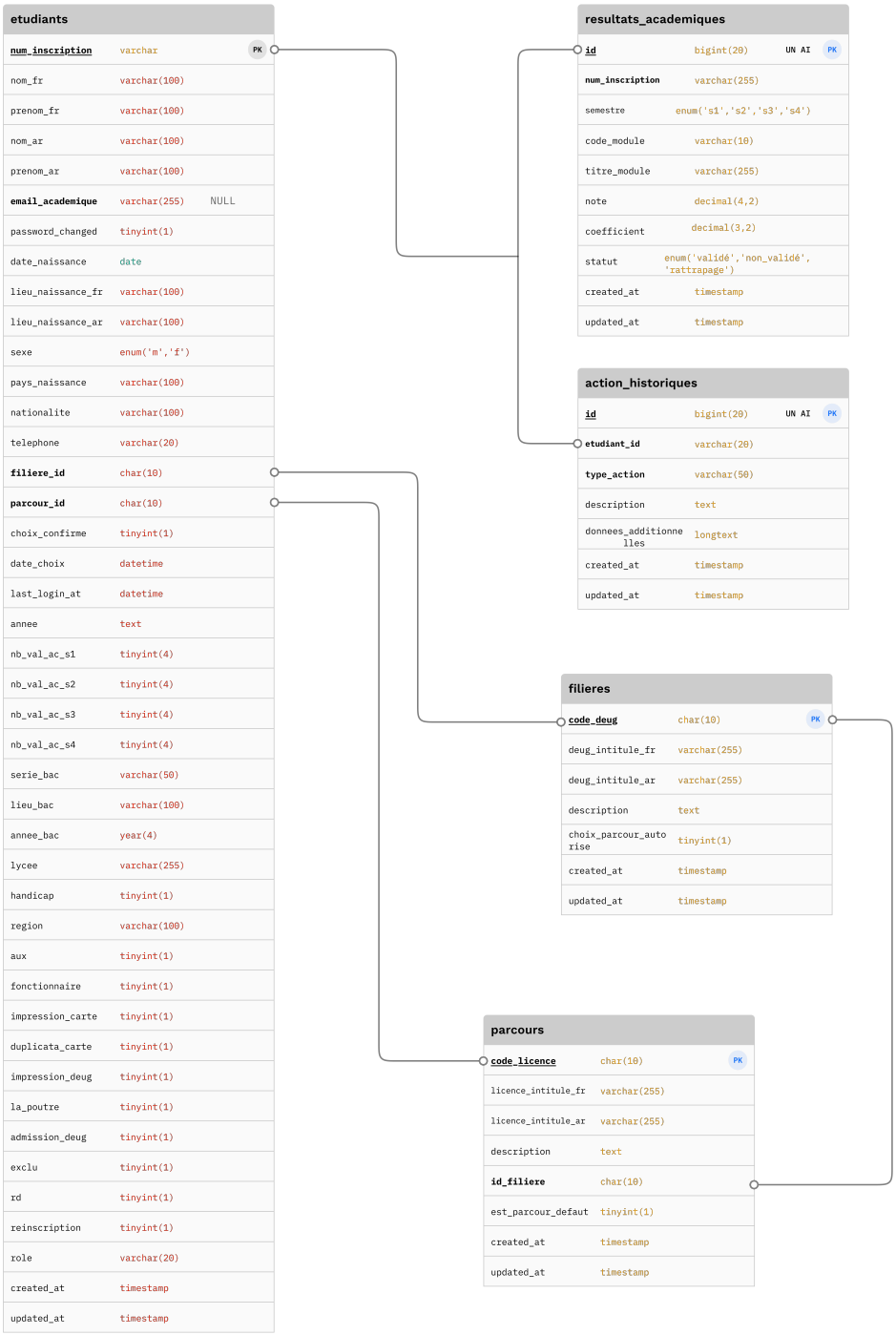


FIGURE B.1 – Modèle Logique de Données (MLD) de la base de données de l'application

B.2. Structure SQL des Tables Principales

```sql
-- Table des étudiants
CREATE TABLE etudiants (
    num_inscription VARCHAR(20) PRIMARY KEY,
    nom_fr VARCHAR(100) NOT NULL,
    prenom_fr VARCHAR(100) NOT NULL,
    email_academique VARCHAR(255) UNIQUE NOT NULL,
    -- ... autres colonnes
);

-- Table des filières
CREATE TABLE filieres (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    code_deug VARCHAR(10) UNIQUE NOT NULL,
    intitule_fr VARCHAR(255) NOT NULL,
    -- ... autres colonnes
);

-- Table des parcours
CREATE TABLE parcours (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nom_fr VARCHAR(255) NOT NULL,
    filiere_id VARCHAR(10) NOT NULL,
    prerequis JSON,
    -- ... autres colonnes
    FOREIGN KEY (filiere_id) REFERENCES filieres(code_deug)
);
```

B.3. Dictionnaire des Données

Table	Champ	Type	Description
etudiants	num_inscription	VARCHAR(20)	Numéro d'inscription unique
etudiants	email_academique	VARCHAR(255)	Email institutionnel
etudiants	choix_confirme	BOOLEAN	Statut de confirmation du parcours
filieres	code_deug	VARCHAR(10)	Code officiel de la filière DEUG
parcours	prerequis	JSON	Conditions d'accès au parcours

TABLE B.1 – Dictionnaire des données des tables principales

C. Interfaces Utilisateur

C.1. Diagramme de Flux Utilisateur

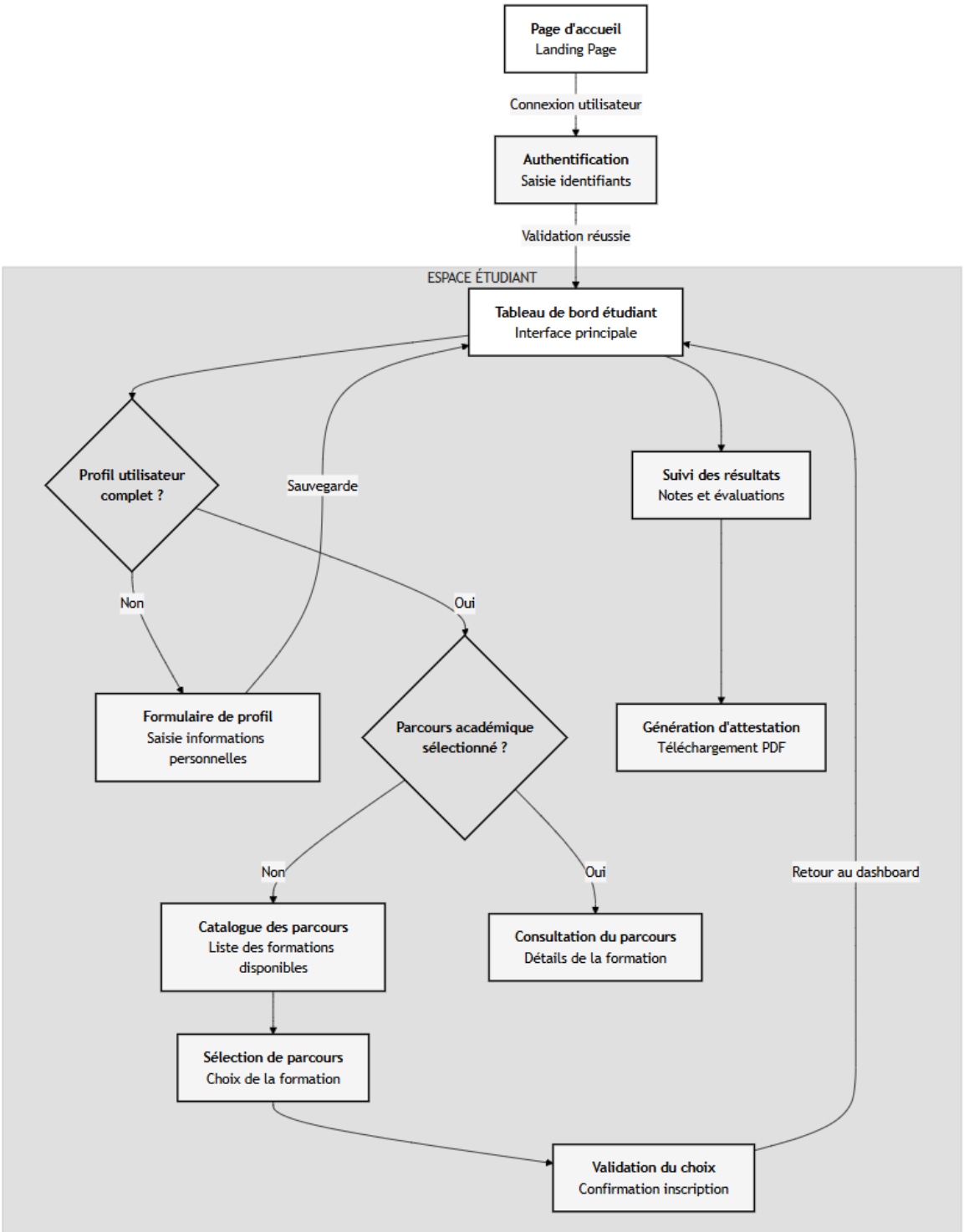


FIGURE C.1 – Diagramme de flux du parcours utilisateur au sein de l'application

C.2. Captures d'Écran des Principales Interfaces

Cette section présente les interfaces clés de l'application, illustrant le parcours complet de l'étudiant, de l'authentification à la génération de son attestation.

C.2.1. Interface de Connexion

L'écran de connexion est la porte d'entrée de l'application. Il est conçu pour être simple et sécurisé, demandant les identifiants institutionnels de l'étudiant.

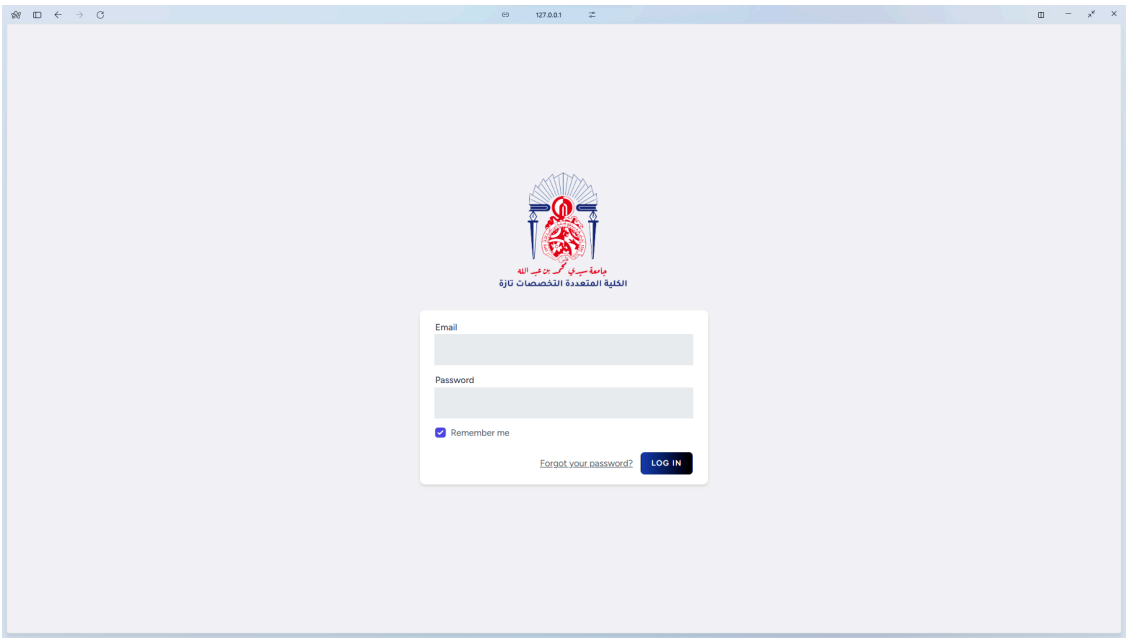


FIGURE C.2 – Écran d'authentification de l'application.

C.2.2. Sécurité au Premier Accès

Pour des raisons de sécurité, l'application peut forcer un changement de mot de passe lors de la première connexion.

Édition publique : les identifiants et données personnelles des captures sont retirés. Les interfaces illustrent le fonctionnement, sans attester un usage en production.

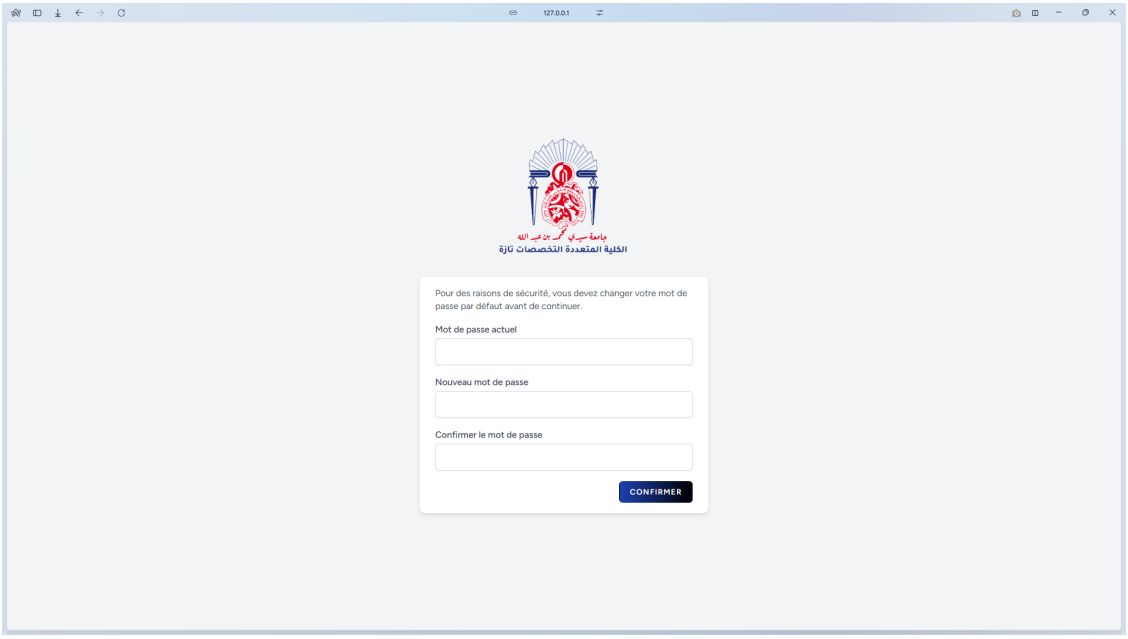


FIGURE C.3 – Interface de modification du mot de passe.

C.2.3. Tableau de Bord Étudiant

Une fois connecté, l'étudiant accède à son tableau de bord personnalisé. Cette interface centrale offre une vue d'ensemble des informations et des actions possibles, comme le choix d'un parcours.

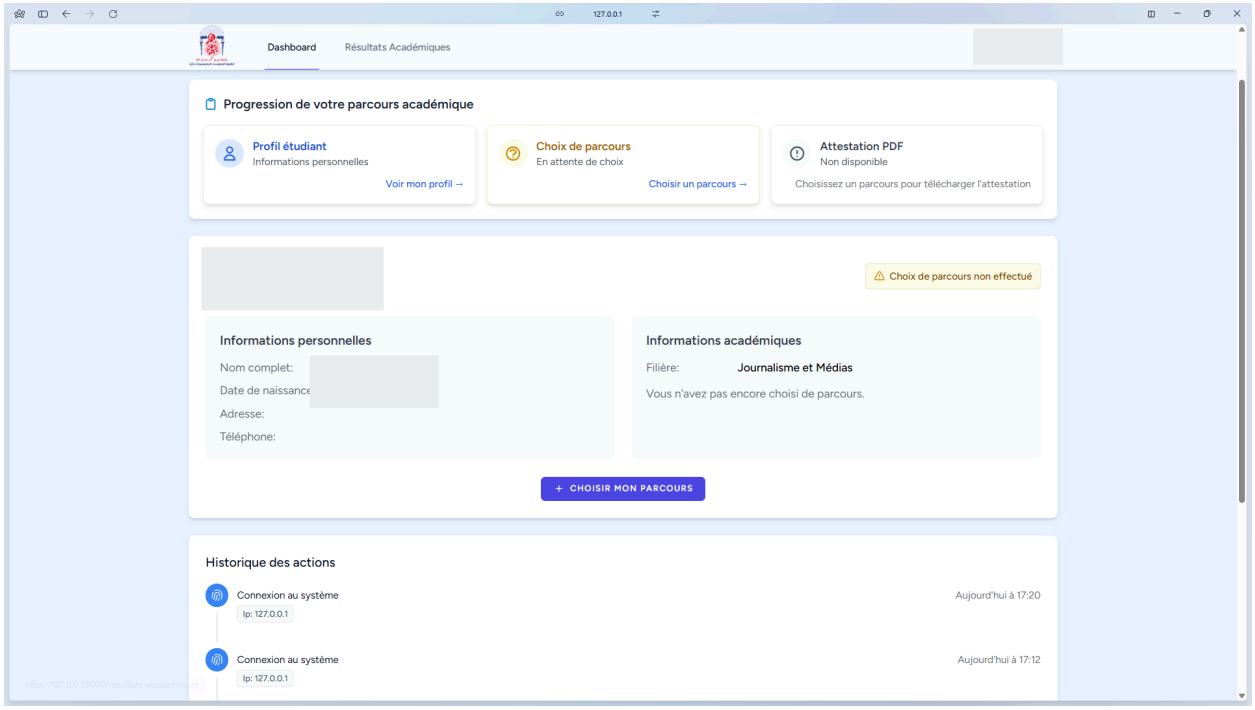


FIGURE C.4 – Vue principale du tableau de bord étudiant.

C.2.4. Consultation du Profil et des Résultats

Le tableau de bord permet d'accéder à des vues détaillées du profil personnel (figure C.5) et des résultats académiques avec des statistiques de performance (figure C.6).

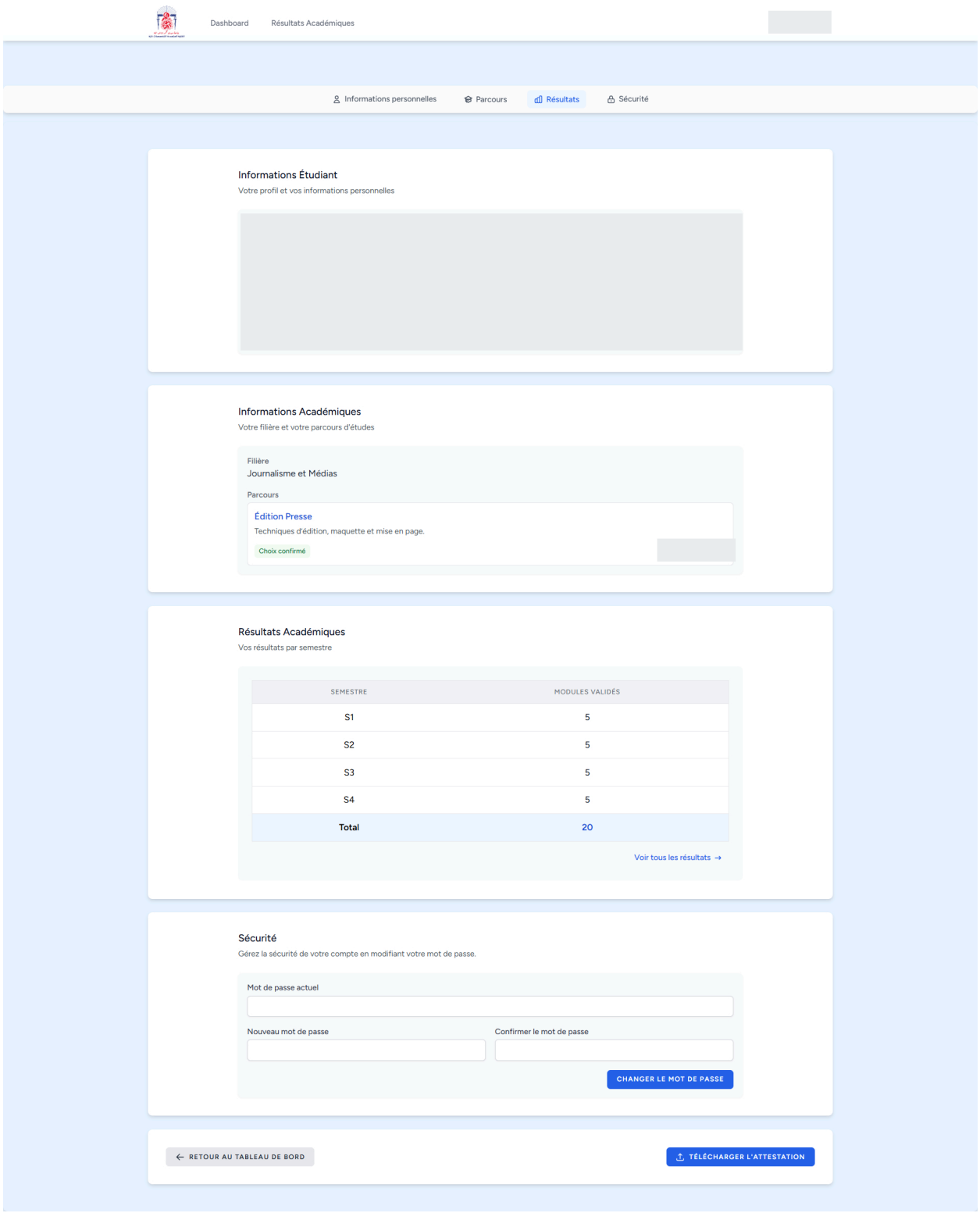


FIGURE C.5 – Interface de consultation et gestion du profil étudiant.

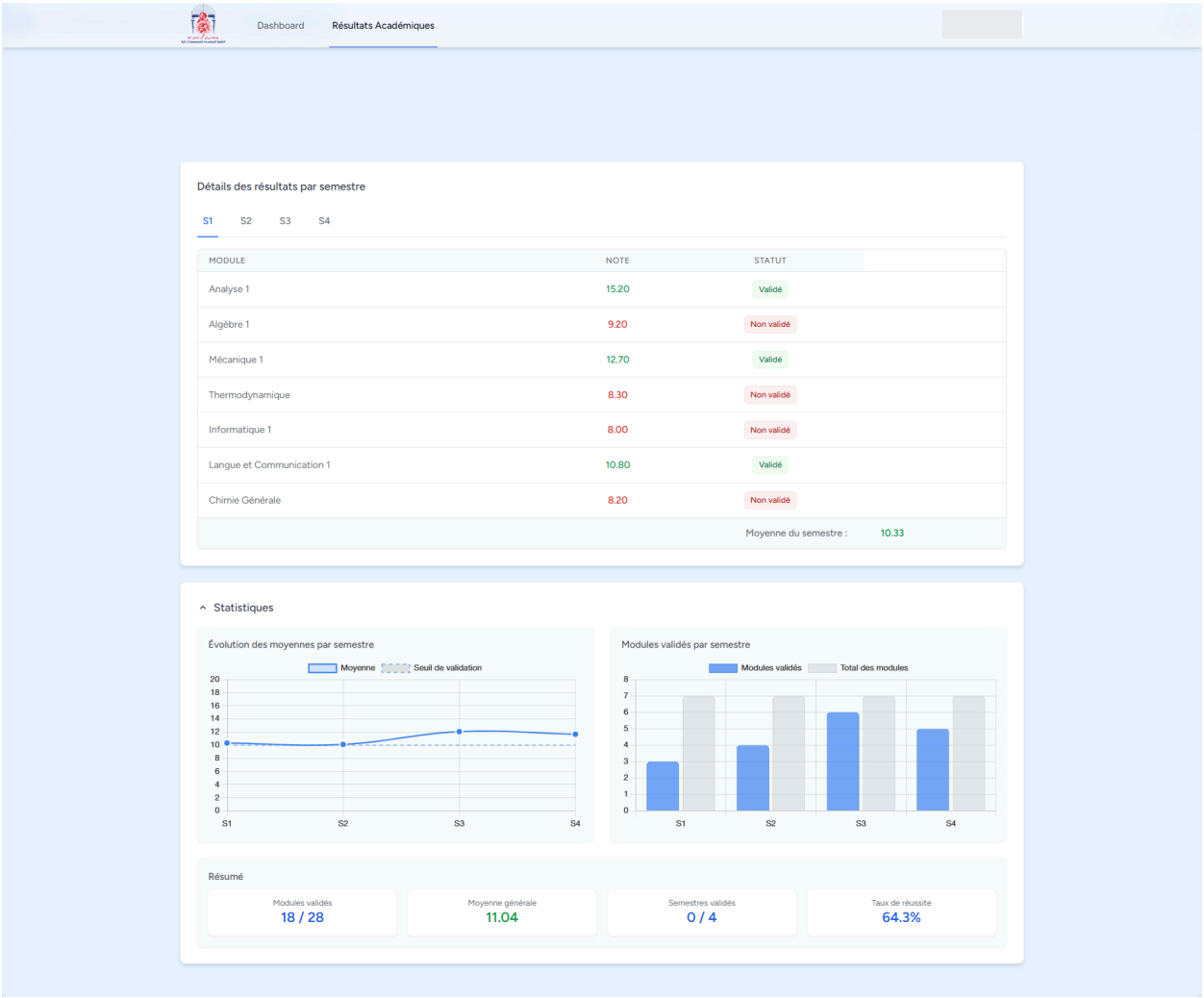


FIGURE C.6 – Vue détaillée des résultats académiques avec graphiques de suivi.

C.2.5. Interface de Sélection de Parcours

Cette interface guide l'étudiant dans le choix de son parcours, en présentant clairement les options disponibles et les informations relatives à chaque formation.

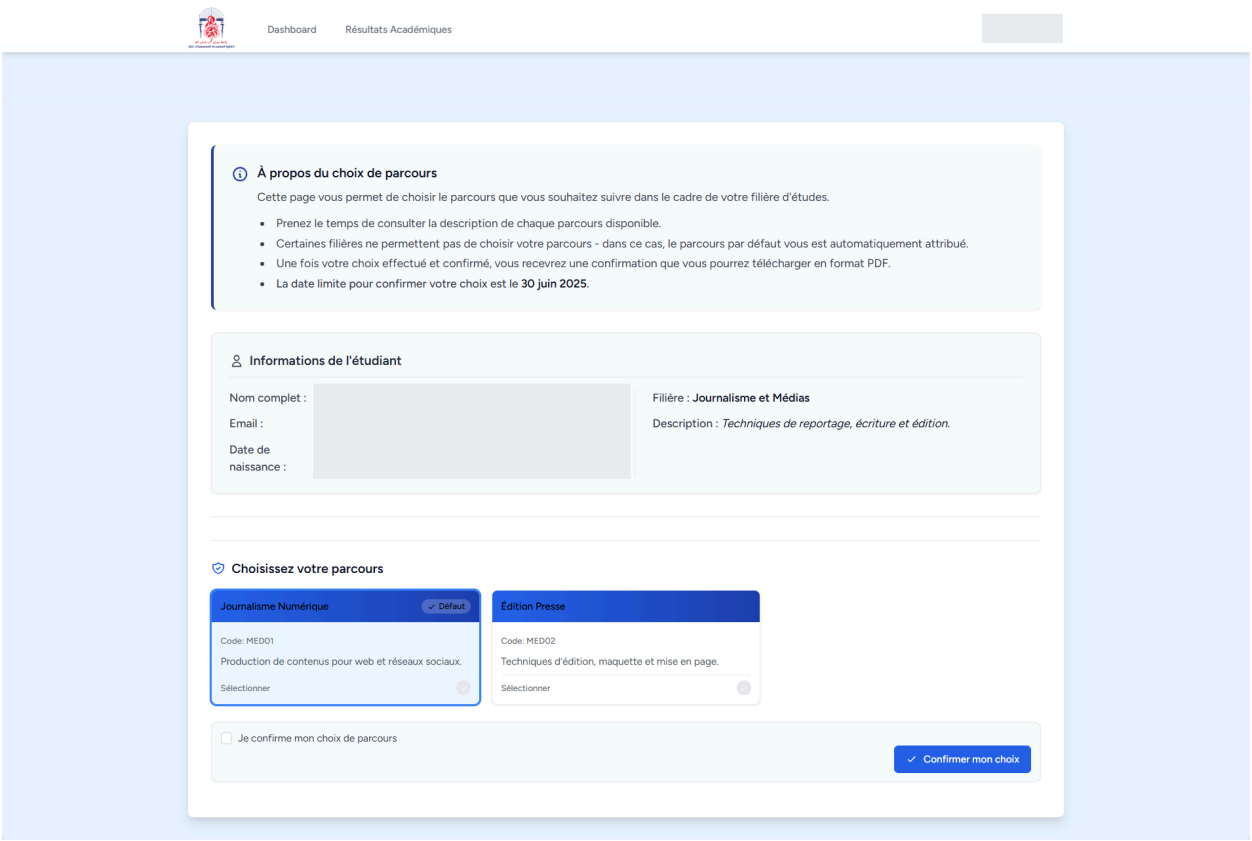


FIGURE C.7 – Écran de sélection et de confirmation du parcours.

C.2.6. Génération et Visualisation de l'Attestation

Après la confirmation d'un choix, une notification de succès apparaît (figure C.8), permettant de télécharger l'attestation officielle (figure C.9) qui confirme l'inscription.

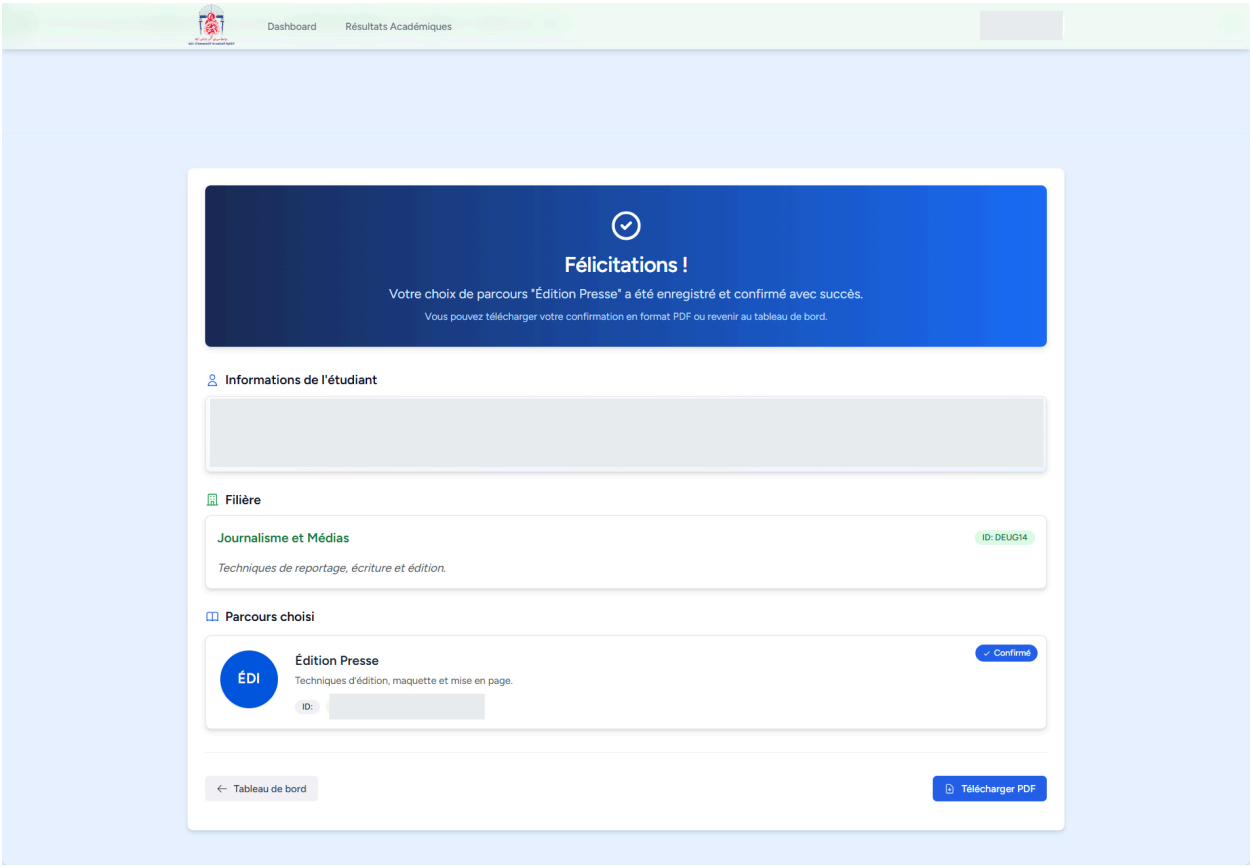
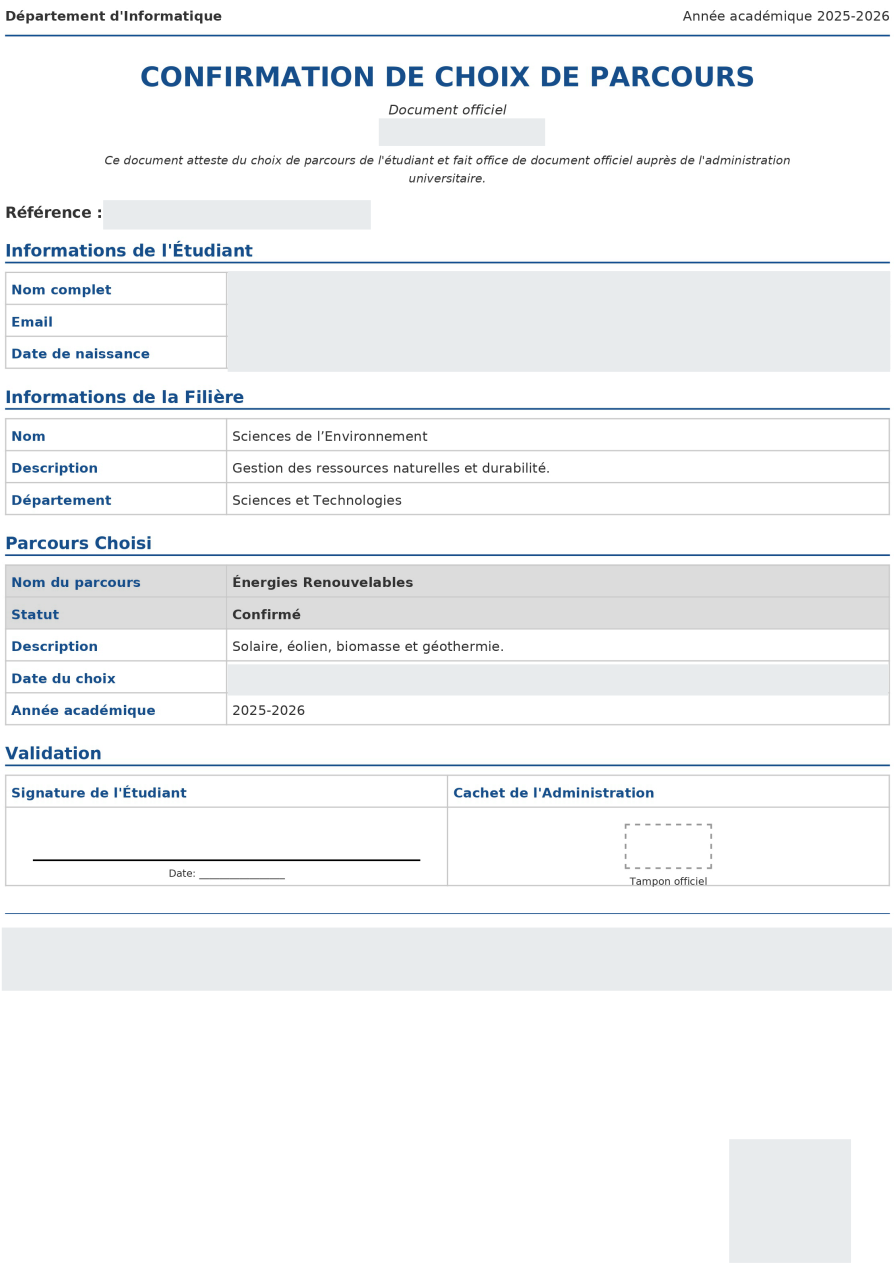


FIGURE C.8 – Écran de confirmation après le choix du parcours.



Département d'Informatique Année académique 2025-2026

CONFIRMATION DE CHOIX DE PARCOURS

Document officiel

Ce document atteste du choix de parcours de l'étudiant et fait office de document officiel auprès de l'administration universitaire.

Référence :

Informations de l'Étudiant

Nom complet	
Email	
Date de naissance	

Informations de la Filière

Nom	Sciences de l'Environnement
Description	Gestion des ressources naturelles et durabilité.
Département	Sciences et Technologies

Parcours Choisi

Nom du parcours	**Énergies Renouvelables**
Statut	**Confirmé**
Description	Solaire, éolien, biomasse et géothermie.
Date du choix	
Année académique	2025-2026

Validation

Signature de l'Étudiant	**Cachet de l'Administration**
Date: _____	Tampon officiel

FIGURE C.9 – Exemple de l'attestation de choix de parcours générée en PDF.

Exemple anonymisé : identité, référence, empreinte et QR code retirés.

C.3. Responsive Design

```
/* Configuration Tailwind CSS pour les breakpoints */
@media (min-width: 640px) { /* sm */ }
@media (min-width: 768px) { /* md */ }
@media (min-width: 1024px) { /* lg */ }
@media (min-width: 1280px) { /* xl */ }
```

D. Sécurité et Authentification

D.1. Mécanismes de sécurité décrits

1. **Authentification**
   - — Hachage des mots de passe avec bcrypt
   - — Sessions sécurisées avec Laravel Sanctum
   - - Limitation des tentatives : durcissement à vérifier.
2. **Autorisation**
   - — Middleware d'authentification sur toutes les routes sensibles
   - - Permissions par rôle : granularité à vérifier.
   - — Protection CSRF sur tous les formulaires
3. **Validation des Données**
   - — Validation côté serveur avec Laravel Validation
   - — Échappement XSS automatique
   - — Nettoyage des entrées utilisateur

D.2. Exemple de configuration de sécurité

```
// Configuration des sessions sécurisées
'secure' => env('SESSION_SECURE_COOKIE', true),
'http_only' => true,
'same_site' => 'strict',

// Protection CSRF
'csrf' => [
    'expire' => 120, // minutes
    'regenerate' => true,
],
```

Précision : cette annexe décrit des mécanismes et un exemple de configuration. Le rapport ne prouve pas leur activation en production. Le durcissement des cookies, la limitation des tentatives et les permissions fines restent à vérifier, conformément aux perspectives de l'annexe E.

E. Considérations de Sécurité

La sécurité a été une préoccupation centrale durant le développement. Le framework Laravel fournit nativement une base de sécurité robuste qui a été utilisée comme socle pour ce projet.

E.1. Mesures de Sécurité Natives de Laravel

L'application bénéficie des mécanismes de sécurité intégrés par défaut dans le framework, qui constituent une première ligne de défense efficace :

1. **Authentification Fondamentale :**
   - **Hachage des mots de passe :** Laravel utilise automatiquement l'algorithme **bcrypt** pour stocker les mots de passe de manière sécurisée, les rendant non réversibles.
   - **Protection des sessions :** Le système de session natif de Laravel protège contre le "session hijacking".
2. **Autorisation et Contrôle d'Accès :**
   - **Middleware d'authentification :** L'accès aux routes sensibles de l'application est protégé par le middleware `auth`, garantissant que seuls les utilisateurs connectés peuvent y accéder.
   - **Protection CSRF :** Tous les formulaires POST sont automatiquement protégés contre les attaques de type Cross-Site Request Forgery grâce au token CSRF de Laravel.
3. **Validation des Données :**
   - **Validation côté serveur :** Le moteur de validation de Laravel est utilisé pour s'assurer de l'intégrité et du format des données soumises par les formulaires.
   - **Protection XSS :** Le moteur de template Blade échappe par défaut toutes les variables affichées, ce qui prévient les attaques de type Cross-Site Scripting.

E.2. Axes d'Amélioration de la Sécurité

Bien que la base soit solide, plusieurs renforcements de sécurité sont identifiés comme des évolutions futures prioritaires :

- **Configuration des cookies sécurisés :** Forcer les options `secure` et `same_site='strict'` dans le fichier d'environnement pour renforcer la sécurité des cookies de session.
- **Limitation des tentatives de connexion (Rate Limiting) :** Mettre en place un middleware de "rate limiting" sur la route de connexion pour prévenir les attaques par force brute.

— **Gestion fine des permissions :** Implémenter un système de rôles et permissions plus granulaire si des profils administrateurs étaient ajoutés.

F. Stratégie de Tests et Qualité

Une démarche qualité a été initiée pour garantir la fiabilité de l'application, en s'appuyant sur les outils de test intégrés à Laravel.

F.1. Tests Fonctionnels avec PHPUnit

Des tests fonctionnels ont été rédigés pour valider les parcours utilisateurs les plus critiques de l'application. Ces tests simulent les actions d'un utilisateur et vérifient que le système répond correctement. L'exemple ci-dessous illustre un test qui valide le processus de sélection de parcours par un étudiant.

```
// Exemple de test fonctionnel pour la sélection de parcours
class ParcourSelectionTest extends TestCase
{
    public function test_student_can_select_parcour()
    {
        $user = Etudiant::factory()->create();
        $parcour = Parcour::factory()->create();

        $response = $this->actingAs($user)
            ->post('/parcours/select', [
                'parcour_id' => $parcour->id
            ]);

        $response->assertRedirect('/dashboard');
        $this->assertDatabaseHas('etudiants', [
            'num_inscription' => $user->num_inscription,
            'parcour_id' => $parcour->id
        ]);
    }
}
```

F.2. Perspectives d'Amélioration de la Qualité

La stratégie de test actuelle constitue une première étape. Pour atteindre un niveau de qualité industriel, les axes d'amélioration suivants sont identifiés :

— **Augmentation de la couverture de code :** Étendre la suite de tests (unitaires et fonctionnels) pour couvrir une plus grande partie du code source et viser une métrique de couverture (ex :

80%).

— **Analyse statique du code :** Intégrer des outils comme PHPStan ou PHP-CS-Fixer dans le processus de développement pour garantir le respect automatique des standards de codage (comme PSR-12) et détecter les erreurs potentielles avant l'exécution.

— **Automatisation des tests (CI/CD) :** Mettre en place un pipeline d'intégration continue qui lancerait automatiquement la suite de tests à chaque modification du code, assurant une détection immédiate des régressions.

G. Déploiement et Configuration

G.1. Variables d'Environnement (.env)

```
# Application
APP_NAME="Gestion Parcours USMBA"
APP_ENV=production
APP_KEY=base64:GENERATED_KEY_HERE
...
# Base de données
DB_CONNECTION=mysql
DB_HOST=localhost
...
```

Exemple de configuration uniquement : GENERATED_KEY_HERE et localhost sont des valeurs de démonstration. Cet extrait ne constitue pas une preuve de déploiement en production.

H. Manuel Utilisateur

H.1. Guide de Connexion

1. **Accès à la plateforme**
   - URL : https ://127.0.0.1 :8000/login (local)
   - Navigateurs supportés : Chrome 90+, Firefox 88+, Safari 14+
2. **Première connexion**
   - Identifiant : Numéro d'inscription
   - Mot de passe : Fourni par l'administration
   - Changement obligatoire du mot de passe

H.2. Procédure de Sélection de Parcours

1. Consultation des parcours disponibles
2. Vérification des prérequis
3. Sélection et confirmation
4. Génération de l'attestation
5. Consultation la scolarite