



Université Chouaib DOUKKALI

Faculté des Sciences

Département d'Informatique



Projet de fin de module

Pour Clôturer le module

Master en Ingénierie Informatique et Analyse de Données

Promotion 2024/2025

Sous le thème

Application Web de Gestion des Professeurs

Encadré par :

Pr. Omar BOUTKHOUM

Réalisé par :

Nouhaila Chahmi

Abdeljalil Sersif

Yasmine Fkikih

Omaïma Laguribi

Table de figures

Figure 1:Diagramme de classes.....	12
Figure 2 : Interface d'authentification	32
Figure 3 : La page de réinitialisation de mot passe	32
Figure 4:La page de vérification de code envoyer sur mail	33
Figure 5 : La page pour saisir un nouveau mot de passe.....	33
Figure 6 : La page principale de admin	34
Figure 7 : Interface responsable de la gestion des fichiers Excel.....	34
Figure 8 : Interface responsable a l'ajout de Professeur	35
Figure 9 : Carte professionnelle au format PDF	35
Figure 10 : Interface du profile d'un professeur.....	36
Figure 11 : Interface de modification des informations d'un professeur	37
Figure 12 le nombre des commits de 1er collaborateur	38
Figure 13 : les détails des commits pour 1er collaborateur.....	38
Figure 14: le nombre des commit du 2ème collaborateur	38
Figure 15 les details des commits du 2ème collaborateur.....	39

Table de matières

TABLE DE FIGURES	2
INTRODUCTION GENERALE.....	6
CHAPITRE1 :	7
ANALYSE ET CONCEPTION DU SYSTEME	7
I. INTRODUCTION.....	8
1. Présentation Générale du Projet	8
2. Objectifs et Enjeux	8
3. Problématique	9
II. CONTEXTE ET JUSTIFICATION DU PROJET	9
1. Besoin d’une Gestion Efficace des Professeurs.....	10
2. Limites des Systèmes Existants	10
III. ÉTUDE DES BESOINS.....	10
1. Identification des utilisateurs	10
2. Fonctionnalités essentielles	11
IV. MODELISATION DU SYSTEME	11
1. Diagrammes UML	11
o Diagramme de classes	11
o Présentation Générale	12
o Description des Classes.....	13
o Relations entre les Classes	15
V. CONCLUSION	15
CHAPITRE2 :	16
DEVELOPPEMENT ET IMPLEMENTATION	16
I. INTRODUCTION.....	17
1. Backend : Node.js avec Express et MySQL.....	17
o Node.js	17
o Express.js	17
o MySQL	18
2. Frontend : React.js.....	18
o React.js.....	18

3.	Sécurité et gestion des utilisateurs.....	18
o	bcrypt	18
4.	Stockage et manipulation des fichiers	19
o	Multer (Upload de fichiers)	19
o	XLSX (Manipulation des fichiers Excel)	19
5.	Gestion du code source avec Git & GitHub	19
o	Git	19
o	GitHub.....	20
o	Organisation du projet avec Git	20
6.	Étapes pour cloner et lancer le projet	21
II.	ARCHITECTURE ET DEVELOPPEMENT	22
1.	API RESTful avec Express.js.....	22
2.	Communication entre les deux modules.....	22
III.	ARCHITECTURE GLOBALE	23
1.	Backend	23
2.	Routes et Fonctionnalités.....	23
3.	Frontend.....	26
IV.	PROBLEMES RENCONTRES ET SOLUTIONS.....	26
1.	Gestion des erreurs d'authentification	26
2.	Gestion des fichiers Excel (Importation et traitement).....	27
V.	CONCLUSION	28
CHAPITRE3 :		29
TESTS ET RESULTATS		29
I.	INTRODUCTION.....	30
II.	TESTS ET VALIDATION.....	30
1.	Tests unitaires	30
2.	Tests d'intégration	30
III.	RESULTATS OBTENUS	32
1.	Interface utilisateur avec React.js	32
o	Authentification	32
o	Récupération de mot passe.....	32
o	Admin	33
o	Importation fichier Excel	34

o Ajouter Professeurs	34
o Profile Professeurs	35
o Modifier Profile	36
2. La gestion du projet	37
IV. CONCLUSION	40
CONCLUSION GENERALE ET PERSPECTIVES	41

Introduction générale

Dans un monde où la transformation numérique joue un rôle central, la gestion efficace des informations devient un enjeu crucial pour de nombreuses organisations, notamment dans le domaine de l'enseignement. La gestion des professeurs, qui englobe l'enregistrement, le suivi et l'organisation de leurs données administratives et professionnelles, représente un défi majeur pour les établissements. Les méthodes traditionnelles de traitement manuel des informations montrent rapidement leurs limites, engendrant des risques d'erreurs, une perte de temps considérable et des difficultés liées à l'accessibilité et à la sécurité des données. Face à ces défis, l'automatisation des processus administratifs apparaît comme une solution incontournable pour améliorer la productivité et optimiser la gestion des ressources.

L'intégration des technologies web dans ce contexte permet de développer des outils performants, facilitant la centralisation et la structuration des données des professeurs. Une plateforme numérique adaptée offre non seulement une meilleure organisation, mais aussi une expérience utilisateur optimisée, rendant l'accès aux informations plus fluide et sécurisé. En combinant des bases de données robustes, des interfaces ergonomiques et des systèmes de gestion automatisés, il devient possible de simplifier considérablement les tâches administratives liées aux professeurs et de garantir une traçabilité efficace des informations.

Dans cette optique, ce travail s'inscrit dans une démarche d'innovation visant à concevoir et mettre en place une solution numérique capable de répondre aux exigences croissantes en matière de gestion des professeurs. Ce rapport explore ainsi les différentes étapes de conception et de développement d'une application web dédiée, en mettant en avant les choix technologiques, les défis rencontrés et les perspectives d'amélioration pour assurer une solution évolutive et performante.

Chapitre 1 : Analyse et Conception du Système

I. Introduction

Dans le cadre de ce projet, l'analyse et la conception du système jouent un rôle central pour répondre aux besoins identifiés et garantir la mise en place d'une solution adaptée et efficace. Ce chapitre explore les différentes étapes nécessaires pour définir les fondations du projet, en s'appuyant sur une approche structurée et méthodique.

Nous commençons par présenter le contexte et la justification du projet, mettant en évidence la nécessité d'une gestion efficace des professeurs et les limites des systèmes actuels, souvent manuels et source d'erreurs ou de perte de temps.

Ensuite, une étude approfondie des besoins permet d'identifier clairement les utilisateurs cibles, les administrateurs et les professeurs et les fonctionnalités clés attendues du système, telles que la gestion des professeurs, l'authentification sécurisée, la génération de cartes professionnelles avec QR code et l'importation de données via Excel.

Enfin, pour traduire ces besoins en un modèle technique cohérent, nous présentons une modélisation du système à l'aide de diagrammes UML, notamment un diagramme de classes, afin de poser les bases de l'architecture et de la conception logicielle.

Ce chapitre constitue ainsi une étape essentielle pour passer d'une idée à une solution concrète et fonctionnelle, en assurant une compréhension claire et partagée des objectifs du projet.

1. Présentation Générale du Projet

Dans un environnement académique en constante évolution, la gestion efficace des professeurs représente un enjeu majeur pour les établissements d'enseignement. Les méthodes traditionnelles de gestion, basées sur des documents papier ou des fichiers Excel, entraînent des risques d'erreurs, un manque de centralisation des informations et une complexité administrative accrue.

Ce projet vise à développer une application web intuitive et performante permettant aux établissements de gérer efficacement les professeurs en automatisant plusieurs processus clés, notamment la saisie des données, l'importation de fichiers Excel et la génération de cartes professionnelles avec QR code.

2. Objectifs et Enjeux

L'objectif principal de ce projet est de **simplifier et optimiser la gestion des professeurs** en proposant une plateforme moderne et sécurisée. Plus précisément, l'application permettra de :

- **Faciliter l'enregistrement et la mise à jour des informations** des professeurs.

- **Automatiser l'importation des données** via des fichiers Excel pour éviter la saisie manuelle.
- **Générer des cartes professionnelles** intégrant un QR code pour une identification rapide et fiable.
- **Garantir une interface utilisateur ergonomique**, facilitant l'interaction avec le système et réduisant le temps de gestion administrative.

Les enjeux majeurs de ce projet résident dans **l'amélioration de l'efficacité administrative** en automatisant les tâches répétitives, **la réduction des erreurs humaines** grâce à une gestion centralisée et sécurisée des informations, ainsi que **l'optimisation de l'accès aux données** pour une consultation rapide, fluide et structurée, facilitant ainsi la prise de décision et le suivi administratif.

3. Problématique

Les établissements d'enseignement rencontrent souvent des difficultés liées à la gestion des informations des professeurs, notamment en raison du manque d'outils adaptés. La gestion traditionnelle entraîne plusieurs contraintes :

- **Une surcharge administrative** due à la saisie et à la mise à jour manuelle des informations.
- **Un manque de centralisation des données**, rendant difficile la consultation rapide des informations des professeurs.
- **L'absence d'uniformisation et de sécurité** dans la gestion des identifications professionnelles.

Face à ces défis, la question centrale de ce projet est la suivante :

Comment concevoir une application web intuitive et performante permettant de centraliser, sécuriser et automatiser la gestion des professeurs au sein d'un établissement d'enseignement ?

II. Contexte et Justification du Projet

Dans le domaine de l'enseignement, la gestion des professeurs est une tâche essentielle qui nécessite une organisation rigoureuse et une mise à jour constante des informations administratives et professionnelles. Pourtant, de nombreux établissements s'appuient encore

sur des méthodes traditionnelles, telles que la gestion manuelle via des fichiers papier ou des tableurs, ce qui entraîne plusieurs limites et inefficacités.

1. Besoin d'une Gestion Efficace des Professeurs

Un système performant de gestion des professeurs permet de :

- **Centraliser et sécuriser les données** pour éviter les pertes ou les erreurs de saisie.
- **Faciliter l'accès aux informations** pour une meilleure organisation administrative.
- **Automatiser certaines tâches répétitives**, comme l'enregistrement des nouvelles données ou la génération de documents professionnels.
- **Optimiser le suivi des enseignants**, en assurant une meilleure visibilité sur leur statut et leur rôle au sein de l'établissement.

2. Limites des Systèmes Existants

Les approches classiques de gestion des professeurs présentent plusieurs inconvénients :

- **Risque élevé d'erreurs** dues à la saisie manuelle des informations.
- **Perte de temps importante**, notamment lors de la recherche ou de la mise à jour des données.
- **Difficulté à générer et partager des documents officiels**, comme les cartes professionnelles.
- **Manque de centralisation**, rendant l'accès aux informations difficile et limitant la collaboration entre les différentes parties prenantes.

Face à ces défis, il devient essentiel de mettre en place une **solution numérique efficace**, permettant de moderniser la gestion des professeurs tout en garantissant une utilisation simple, rapide et sécurisée des données.

III. Étude des besoins

1. Identification des utilisateurs

- **Administrateurs** : Utilisateurs ayant un accès complet au système, capables de gérer les professeurs, et les fonctionnalités de la plateforme. Ils peuvent également assurer la maintenance et la mise à jour des données.
- **Professeurs** : Utilisateurs ayant accès à leurs profils, pouvant mettre à jour leurs données personnelles et générer leurs cartes professionnelles.

2. Fonctionnalités essentielles

Gestion des professeurs :

- Ajout, modification et consultation des informations des professeurs.
- Consultation des profils des professeurs et gestion de leurs données de manière sécurisée et simple.

Authentification et sécurité :

- Mise en place d'un système d'authentification sécurisé pour les utilisateurs, incluant une gestion des rôles (administrateurs et professeurs).
- Systèmes de gestion des mots de passe (cryptage, récupération de mot de passe).

Génération de cartes professionnelles avec QR code :

- Génération automatique de cartes professionnelles pour les professeurs avec leurs informations clés (nom, fonction, photo, etc.).
- Intégration d'un QR code unique pour chaque professeur, permettant de vérifier ses informations et son identité de manière numérique.

Importation des données via Excel :

- Fonctionnalité d'importation de données des professeurs et étudiants via des fichiers Excel pour faciliter l'intégration massive des informations dans la plateforme.
- Validation et nettoyage des données avant l'importation pour garantir l'exactitude et la cohérence des informations.

IV. Modélisation du système

1. Diagrammes UML

○ Diagramme de classes

Le diagramme de classes représente l'architecture logique du système en modélisant les différentes entités, leurs attributs, leurs méthodes et leurs relations. Il constitue une étape essentielle dans la conception du projet, permettant d'assurer une structuration claire et modulaire des fonctionnalités.

○ Présentation Générale

Le système repose sur plusieurs classes interconnectées qui assurent la gestion des professeurs, l'importation des données, la génération de cartes professionnelles et la sécurisation des accès. Les principales entités sont **Admin**, **Professeur**, **FichierExcel**, **Importation**, **Système** et **CartePro**.

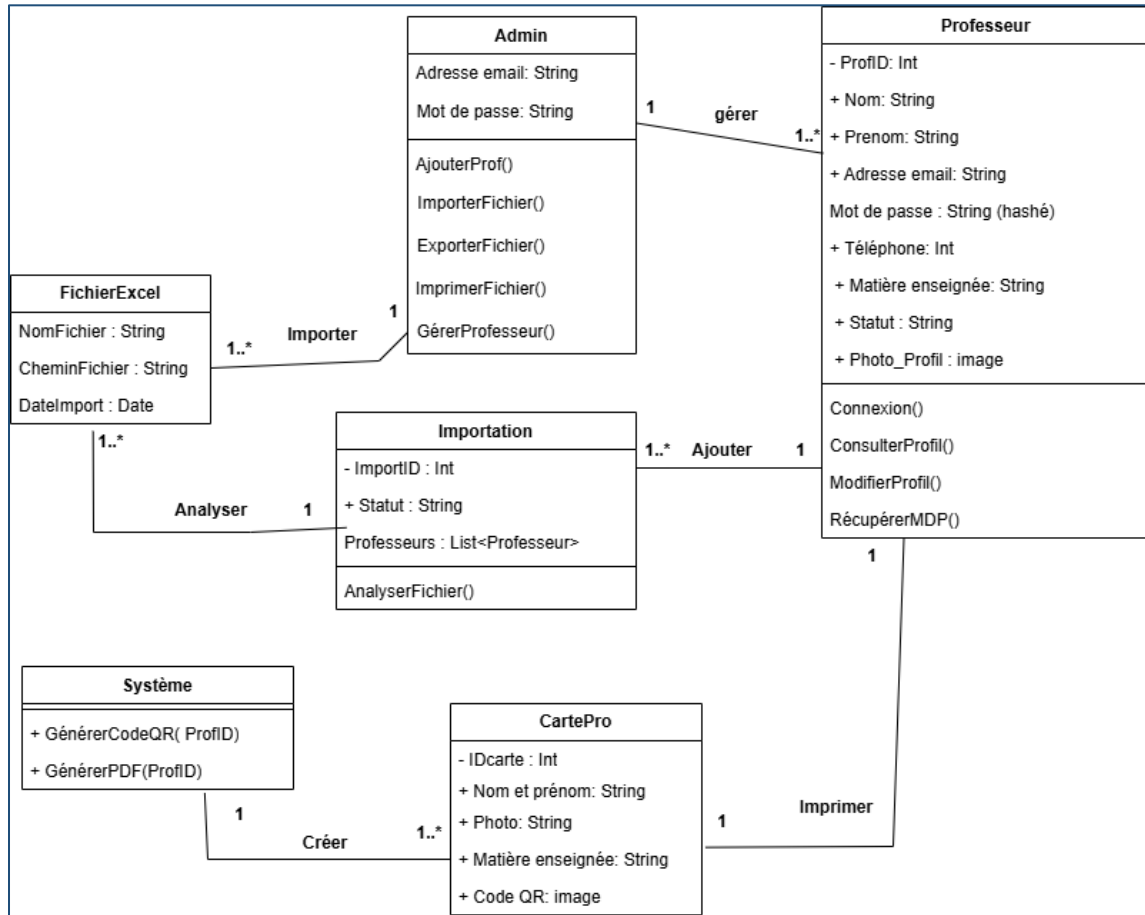


Figure 1:Diagramme de classes

○ Description des Classes

Admin

L'**Admin** est l'utilisateur principal du système, responsable de la gestion des professeurs et de l'importation des fichiers contenant leurs informations. Il dispose des fonctionnalités suivantes:

- **AjouterProf()** : Ajoute un nouveau professeur dans la base de données.
- **ImporterFichier()** : Permet d'importer un fichier Excel contenant des informations sur les professeurs.
- **ExporterFichier()** : Exporte les données des professeurs sous format Excel.
- **ImprimerFichier()** : Imprime les fichiers importés ou exportés.
- **GérerProfesseur()** : Gère les différentes opérations liées aux professeurs (modification, suppression, etc.).

Professeur

La classe **Professeur** représente un utilisateur qui peut se connecter au système et gérer son propre profil. Elle contient les attributs suivants :

- Identifiants personnels : ProfID, Nom, Prénom, Adresse email, Mot de passe (hashé), Téléphone.
- Informations académiques : Matière enseignée, Statut, Photo_Profil.
- Méthodes :
 - **Connexion()** : Permet au professeur d'accéder à son compte.
 - **ConsulterProfil()** : Affiche les informations du professeur.
 - **ModifierProfil()** : Met à jour les informations personnelles.
 - **RécupérerMDP()** : Permet de récupérer un mot de passe oublié.

FichierExcel

Cette classe stocke uniquement les métadonnées du fichier importé sans contenir directement les informations des professeurs, évitant ainsi toute redondance. Ses attributs sont :

- NomFichier : Nom du fichier importé.

- **CheminFichier** : Emplacement où est stocké le fichier.
- **DateImport** : Date d'importation du fichier.

Importation

La classe **Importation** assure l'analyse et l'intégration des données du fichier Excel dans la base. Elle dispose des attributs suivants :

- **ImportID** : Identifiant unique de l'importation.
- **Statut** : Indique si l'importation est en attente, validée ou en erreur.
- **Professeurs** : Liste des professeurs extraits du fichier.
- **AnalyserFichier()** : Analyse le contenu du fichier et valide les données avant leur insertion.

Système

La classe **Système** est responsable de la génération de documents et de QR codes pour identifier les professeurs :

- **GénérerCodeQR(ProfID)** : Crée un QR Code unique basé sur l'identifiant du professeur.
- **GénérerPDF(ProfID)** : Génère un document PDF contenant les informations du professeur.

CartePro

Chaque professeur peut générer une **CartePro**, contenant ses informations et un QR code pour vérification numérique. Elle comprend :

- **IDCarte** : Identifiant de la carte.
- **Nom et prénom** : Informations du professeur.
- **Photo** : Image du professeur.
- **Matière enseignée** : Discipline enseignée par le professeur.
- **Code QR** : QR code généré pour authentification rapide.

○ **Relations entre les Classes**

- Admin **gère** plusieurs Professeurs.
- Admin **importe** un FichierExcel.
- FichierExcel **est analysé par** Importation.
- Importation **ajoute des données dans** Professeur.
- Professeur **peut générer** CartePro.
- Système **crée** CartePro et **génère un QR Code**.
- CartePro **est imprimée par** Professeur.

V. Conclusion

Ce premier chapitre a permis d'établir les bases essentielles du projet en analysant les besoins et en concevant l'architecture du système. L'identification des utilisateurs et la définition des fonctionnalités clés, telles que la gestion des professeurs, l'authentification sécurisée, l'importation des données via Excel et la génération de cartes professionnelles avec QR code, ont permis de structurer efficacement le projet.

La modélisation à travers le diagramme de classes a apporté une vision claire et modulaire du système, garantissant ainsi une gestion optimale des données et une meilleure évolutivité. Cette phase d'analyse et de conception constitue un socle solide pour la mise en œuvre du projet, en assurant une organisation cohérente et adaptée aux besoins des utilisateurs.

Dans le chapitre suivant, nous aborderons la mise en place technique du système, en détaillant les technologies utilisées ainsi que l'implémentation des différentes fonctionnalités définies dans cette analyse.

Chapitre2 : **Développement et Implémentation**

I. Introduction

Ce chapitre détaille le développement et l'implémentation du projet, en mettant en avant les choix technologiques, l'architecture adoptée et les défis rencontrés. Le projet repose sur une stack moderne avec Node.js et Express.js pour le backend, associé à MySQL pour la gestion des données, tandis que React.js est utilisé pour le développement de l'interface utilisateur.

L'application permet notamment l'importation et la lecture de fichiers Excel, offrant ainsi une gestion efficace des données. Elle inclut également la génération de cartes professionnelles en PDF pour les professeurs.

Nous aborderons ensuite les défis techniques rencontrés, tels que la gestion des erreurs d'authentification, l'optimisation des requêtes SQL et l'amélioration de l'expérience utilisateur (UI/UX), en expliquant les solutions mises en place pour améliorer les performances et la fiabilité du système.

1. Backend : Node.js avec Express et MySQL

○ Node.js

Node.js est un environnement d'exécution JavaScript côté serveur, basé sur le moteur V8 de Google Chrome. Il est conçu pour être non-bloquant, événementiel et asynchrone, ce qui le rend particulièrement performant pour les applications nécessitant un traitement en temps réel et des opérations d'entrées/sorties rapides.



Pourquoi avons-nous choisi Node.js ?

- Il est léger et optimisé pour les applications web à grande échelle.
- Son architecture asynchrone permet de gérer plusieurs requêtes simultanément sans ralentissement.
- Il bénéficie d'une large communauté et de nombreux modules via npm.

○ Express.js

Express.js est un Framework minimaliste pour Node.js qui facilite la création d'API RESTful. Il offre une structure légère mais flexible, idéale pour construire des applications web robustes.

Pourquoi avons-nous choisi Express.js ?

- Il simplifie la gestion des routes et des requêtes HTTP.

- Il permet une intégration facile avec les bases de données et les middlewares.
- Il est compatible avec plusieurs bibliothèques de sécurité et de gestion des erreurs

○ **MySQL**

MySQL est un système de gestion de bases de données relationnelles (SGBDR) open-source largement utilisé. Il offre une excellente gestion des transactions et une forte intégrité des données.



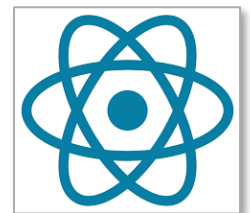
Pourquoi avons-nous choisi MySQL ?

- Il assure une gestion efficace des relations entre les différentes entités du projet.
- Il est bien supporté et compatible avec Node.js via le module mysql2.
- Il est optimisé pour les requêtes SQL complexes et les grandes bases de données.

2. Frontend : React.js

○ **React.js**

React.js est une bibliothèque JavaScript développée par Facebook, conçue pour créer des interfaces utilisateur dynamiques et performantes. Elle repose sur un système de composants réutilisables qui facilite la maintenance et l'évolution du projet.



Pourquoi avons-nous choisi React.js ?

- Son approche basée sur les composants rend l'interface plus modulaire et facile à gérer.
- Il offre un rendu rapide grâce au Virtual DOM, améliorant ainsi les performances de l'application.
- Il permet une intégration fluide avec les API RESTful fournies par le backend.

3. Sécurité et gestion des utilisateurs

○ **bcrypt**

La sécurité des mots de passe est essentielle pour protéger les comptes des utilisateurs. bcrypt est un algorithme de hachage qui permet de stocker les mots de passe de manière sécurisée.

Pourquoi avons-nous choisi bcrypt ?

- Il utilise un algorithme de hachage robuste avec un sel unique pour chaque mot de passe.
- Il protège les mots de passe contre les attaques par force brute et rainbow tables.
- Il est facile à implémenter avec Node.js via le module bcryptjs.

4. Stockage et manipulation des fichiers

○ Multer (Upload de fichiers)

Multer est un middleware utilisé pour gérer le téléchargement et le stockage de fichiers côté serveur.

Pourquoi avons-nous choisi Multer ?

- Il permet d'enregistrer les photos de profil des professeurs de manière efficace.
- Il gère les types de fichiers et limite leur taille pour des raisons de sécurité.

○ XLSX (Manipulation des fichiers Excel)

XLSX est une bibliothèque qui permet de lire et d'écrire des fichiers Excel en JavaScript.

Pourquoi avons-nous choisi XLSX ?

- Il facilite l'importation des données des professeurs depuis des fichiers Excel.
- Il est compatible avec de nombreux formats (xls, xlsx, csv).

5. Gestion du code source avec Git & GitHub

○ Git

Git est un outil de gestion de versions qui permet de suivre les modifications du code, de collaborer efficacement en équipe et de revenir à une version précédente en cas de problème.



Pourquoi avons-nous choisi Git ?

- Il facilite la gestion des différentes versions du projet.
- Il permet de travailler en équipe sans conflit de code.
- Il assure la sauvegarde du projet sur GitHub en cas de problème local.

○ GitHub

GitHub est une plateforme d'hébergement de code qui permet de stocker et de gérer les projets Git en ligne.



Pourquoi avons-nous choisi GitHub ?

- Il permet aux membres de l'équipe d'accéder au projet à tout moment.
- Il facilite la gestion des branches et la collaboration en équipe.
- Il offre des fonctionnalités avancées comme les pull requests, les issues et la CI/CD.

○ Organisation du projet avec Git

L'utilisation de Git dans un projet est essentielle pour la gestion du code source, la collaboration en équipe et le suivi des versions. Git permet de maintenir un historique précis des modifications, de faciliter les retours en arrière et de garantir une gestion efficace des branches et des conflits. Voici les étapes que nous avons suivies pour organiser notre projet avec Git et GitHub.

✚ Initialisation du projet avec Git : **git init**

- Cette commande crée un dépôt Git vide dans notre dossier de projet.
- Une fois cette commande exécutée, Git commence à suivre toutes les modifications qui seront apportées dans ce projet.
- Après l'initialisation, un fichier `.git` est créé dans le répertoire, ce qui permet à Git de gérer le projet.

✚ Ajouter les fichiers à l'index Git : **git add** .

- Cette commande permet d'ajouter tous les fichiers et dossiers du projet à l'index de Git.
- L'index est une zone intermédiaire où Git conserve les fichiers avant qu'ils ne soient définitivement ajoutés à l'historique avec un commit.
- Le caractère `.` représente tous les fichiers modifiés, ajoutés ou supprimés.

✚ Création du premier commit : **git commit -m "Initialisation du projet"**

- Cette commande crée un "instantané" du projet à un moment donné.
- Le message entre les guillemets, ici *"Initialisation du projet"*, doit décrire les modifications effectuées, par exemple *"Ajout des fichiers de base du projet"*.
- Il est important d'écrire des messages de commit clairs et descriptifs pour faciliter la compréhension des changements apportés.

✚ Lier le projet à un dépôt distant sur GitHub : **git remote add origin <URL_du_dépôt>**

- Cette commande associe notre projet local à un dépôt distant (sur GitHub ou une autre plateforme).
- L'URL du dépôt distant est l'adresse unique de notre projet sur GitHub, comme https://github.com/username/gestion_professeurs.
- Cette association nous permet d'envoyer (push) notre code vers GitHub et de récupérer (pull) les modifications effectuées par les autres membres de l'équipe.

Pousser le code vers GitHub (Premier Push) : **git push origin master**

- Cette commande envoie les commits locaux vers le dépôt distant.
- **origin** désigne le dépôt distant par défaut (le dépôt GitHub que vous avez configuré précédemment), et **master** fait référence à la branche par défaut.
- Cette opération est effectuée pour la première fois afin de synchroniser le projet local avec le dépôt distant.
- **Important** : GitHub a récemment renommé la branche par défaut **master** en **main**. Si vous utilisez cette nouvelle convention, remplacez **master** par **main**.

6. Étapes pour cloner et lancer le projet

Les membres de l'équipe doivent suivre ces étapes pour récupérer et exécuter le projet localement :

✚ Cloner le projet depuis GitHub

- git clone https://github.com/Abdo28102002/gestion_professeurs

✚ Installer les dépendances

- **cd frontend**
- **npm install**
- **npm start**

✚ Lancer le serveur backend

- **cd backend**
- Vérifier que le port 3001 est disponible et taper : **node server.js**

✚ Base de données et structure

- Dans votre base de données tu taper cette requête pour crée la base de données :

```
CREATE DATABASE IF NOT EXISTS gestion_professeurs;
```

```
CREATE TABLE IF NOT EXISTS professeurs (
```

```

id INT AUTO_INCREMENT PRIMARY KEY,
nom VARCHAR(100) NOT NULL,
prenom VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL UNIQUE,
telephone VARCHAR(15),
matieres TEXT,
statut ENUM('permanent', 'vacataire') NOT NULL,
photo_profil VARCHAR(255),
mot_de_passe VARCHAR(255) NOT NULL,
date_creation TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
reset_code INT NULL
);

```

II. Architecture et Développement

1. API RESTful avec Express.js

L'application que nous avons développée repose sur une architecture **client-serveur** basée sur des technologies modernes pour garantir performance, scalabilité et facilité de gestion. Elle se compose de deux principaux modules :

- **Frontend** : Développé avec **React.js**, il assure une interface utilisateur dynamique, réactive et intuitive, permettant aux administrateurs et aux professeurs de gérer facilement les informations.
- **Backend** : Basé sur **Node.js** avec le framework **Express.js**, il gère les opérations côté serveur, les interactions avec la base de données et assure la sécurité des échanges via l'authentification et la gestion des sessions.

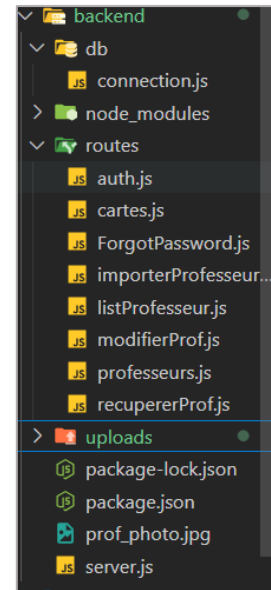
2. Communication entre les deux modules

Une **API RESTful** avec **Express.js** signifie une interface de programmation qui permet à des applications de communiquer avec des ressources via des requêtes HTTP standardisées, tout en suivant les principes de l'architecture REST (Representational State Transfer). Le frontend communique avec le backend via une **API RESTful**, utilisant des requêtes HTTP pour récupérer ou envoyer des données. Ces données sont échangées sous forme de **JSON**, facilitant l'intégration et le traitement côté client.

III. Architecture Globale

1. Backend

- **db/** : Contient la configuration de la connexion à la base de données. Le fichier `connection.js` est utilisé pour établir la connexion avec la base de données, probablement via un ORM comme Sequelize ou un client SQL natif comme MySQL ou PostgreSQL.
- **node_modules/** : Contient toutes les dépendances installées via npm ou yarn. Ce dossier est généré automatiquement après l'installation des packages et ne doit généralement pas être modifié manuellement.
- **routes/** : Contient toutes les routes de votre application. Chaque fichier représente une route spécifique pour gérer l'authentification (`auth.js`), la gestion des professeurs (`professeurs.js`, `modifierProf.js`, etc.), les cartes (`cartes.js`), et la récupération de mots de passe (`ForgotPassword.js`).
- **uploads/** : Probablement utilisé pour stocker les fichiers téléchargés par l'utilisateur, comme des images ou des documents. Le fichier `prof_photo.jpg` semble être un exemple d'une image stockée.



2. Routes et Fonctionnalités

auth.js (Authentification)

- Permet aux professeurs de se connecter via **email et mot de passe**.
- Vérifie les identifiants en **comparant le mot de passe haché** avec celui stocké en base de données.
 - express → Gestion des routes
 - bcrypt → Vérification et hachage des mots de passe
 - mysql2 (via db/connection.js) → Connexion à la base de données

cartes.js (Génération de cartes avec QR Code)

- Génère un fichier **PDF** contenant les informations du professeur.
- **Ajoute un QR Code** avec un lien vers le profil du professeur.
- **Télécharge la photo du professeur** et l'ajoute au PDF.
 - express → Gestion des routes
 - pdfkit → Génération de fichiers PDF
 - qrcode → Création du QR Code

- axios → Récupération d'images via une requête HTTP
- fs → Gestion des fichiers locaux

forgotPassword.js (Réinitialisation du mot de passe)

- Génère un **code de réinitialisation** et l'envoie par **email**.
 - Permet de **vérifier le code** et de modifier le mot de passe.
 - express → Gestion des routes
 - crypto → Génération aléatoire du code de réinitialisation
 - nodemailer → Envoi d'emails via Gmail
 - bcrypt → Hachage du nouveau mot de passe
 - mysql2 (via db/connection.js) → Connexion à la base de données

importerProfesseurs.js (Importation de professeurs depuis un fichier Excel)

- Permet d'importer une liste de professeurs en **format Excel**.
- **Ajoute automatiquement les professeurs** à la base de données.
- **Hache les mots de passe** des nouveaux professeurs.
 - express → Gestion des routes
 - multer → Gestion du téléchargement de fichiers
 - xlsx → Lecture et traitement des fichiers Excel
 - bcrypt → Hachage des mots de passe

modifierProf.js (Modification du profil d'un professeur)

- Permet de modifier les informations d'un professeur, y compris son nom, prénom, email, téléphone, matières, statut et photo de profil.
- Met à jour les données dans la base de données si toutes les informations sont valides.
- Gère l'upload d'une nouvelle photo de profil pour le professeur.
 - express → Gestion des routes
 - multer → Gestion de l'upload de fichiers (photo de profil)
 - mysql → Interaction avec la base de données pour effectuer les mises à jour

professeurs.js (Ajout d'un nouveau professeur)

- Permet d'ajouter un nouveau professeur à la base de données.
- Vérifie la validité des informations (email, mot de passe, etc.).

- Gère l'upload de la photo de profil du professeur.
- Hache les mots de passe des nouveaux professeurs avant de les enregistrer.
 - express → Gestion des routes
 - multer → Gestion de l'upload de fichiers (photo de profil)
 - bcrypt → Hachage des mots de passe des nouveaux utilisateurs
 - validator → Validation des emails

recupererProf.js (Récupération des informations d'un professeur)

- Permet de récupérer les informations d'un professeur à partir de son ID.
- Vérifie que l'ID est valide avant de faire une requête dans la base de données.
- Retourne les détails du professeur demandé sous forme de réponse JSON.
- Technologies utilisées :
 - express → Gestion des routes
 - validator → Validation de l'ID du professeur
 - mysql → Requête pour récupérer les informations du professeur depuis la base de données

server.js (Configuration et démarrage du serveur)

- Configure le serveur Express et les routes pour gérer les opérations liées aux professeurs, à l'authentification, etc.
- Gère les uploads de fichiers via multer et définit le dossier où les fichiers seront stockés.
- Connecte l'application à la base de données MySQL et gère les erreurs.
- Technologies utilisées :
 - express → Configuration du serveur et des routes
 - multer → Gestion de l'upload des fichiers
 - cors → Permet la communication entre différentes origines

- mysql → Connexion et gestion des requêtes à la base de données

3. Frontend

src/ (Code source de l'application)

- Contient tous les fichiers liés à l'application React.

components/ (Composants React)

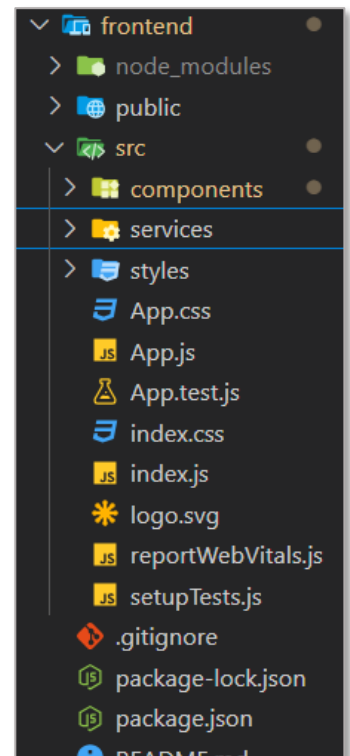
- Ce dossier est généralement utilisé pour stocker tous les composants réutilisables.
- Chaque composant peut être un fichier .js .

services/ (Gestion des appels API et logique métier)

- Contient les fichiers qui interagissent avec une API backend.

styles/ (Fichiers CSS)

Contient les fichiers .css pour styliser les composants



IV. Problèmes rencontrés et solutions

1. Gestion des erreurs d'authentification

Problème : Les erreurs d'authentification peuvent survenir dans plusieurs scénarios : mauvaise saisie des identifiants, mots de passe incorrects ou obsolètes, ou encore des tentatives répétées de connexion avec des informations erronées. Cela peut entraîner une mauvaise expérience utilisateur, et si les erreurs sont fréquentes, cela pourrait indiquer une vulnérabilité exploitée par des utilisateurs malintentionnés.

Solutions

Validation des identifiants : Lors de la connexion, il est crucial de vérifier les informations d'identification de l'utilisateur de manière robuste. Pour ce faire, tu peux utiliser des outils de hachage de mots de passe comme bcrypt pour comparer les mots de passe fournis avec ceux stockés dans la base de données. Le hachage des mots de passe ajoute une couche de sécurité, car les mots de passe ne sont jamais stockés en clair. Par exemple, avant de comparer le mot de passe saisi avec celui enregistré, il doit être haché avec bcrypt.compare() pour assurer une comparaison sécurisée.

Gestion des erreurs d'authentification : Il est important d'afficher des messages d'erreur clairs et précis pour aider les utilisateurs à comprendre pourquoi leur tentative a échoué (par exemple "Nom d'utilisateur incorrect" ou "Mot de passe incorrect"). Les messages doivent être spécifiques, mais pas trop détaillés, afin de ne pas fournir d'informations sensibles qui pourraient être utilisées pour de futures tentatives d'attaque.

2. Gestion des fichiers Excel (Importation et traitement)

Problème : L'importation de données depuis des fichiers Excel peut présenter plusieurs défis, surtout si le fichier contient des informations erronées ou si sa structure n'est pas conforme aux attentes du système. De plus, les fichiers Excel peuvent être volumineux, ce qui peut entraîner des problèmes de performance lors de leur traitement.

Solutions

Validation des données avant importation : Avant d'importer les données dans la base de données, il est important de valider leur format. Par exemple, vérifier que toutes les colonnes nécessaires sont présentes, que les types de données (texte, nombre, date) sont respectés, et que les données ne contiennent pas d'erreurs évidentes comme des champs vides ou des valeurs aberrantes. Si des erreurs sont détectées, il est possible de fournir un retour détaillé à l'utilisateur avec les lignes ou les colonnes problématiques à corriger.

Gestion des erreurs d'importation : Si un fichier est mal formaté ou contient des erreurs, il est essentiel de ne pas laisser l'utilisateur dans l'incertitude. Prévois un mécanisme pour gérer ces erreurs en affichant des messages clairs et utiles (par exemple : "Le fichier contient un email déjà utilisé dans la ligne 2 ."). Cela permettra à l'utilisateur de corriger rapidement les problèmes dans le fichier avant de le soumettre à nouveau.

V. Conclusion

Ce projet repose sur une architecture robuste et moderne combinant Node.js avec Express pour le backend et React.js pour le frontend. Grâce à MySQL, nous assurons une gestion efficace des données, tandis que des outils comme bcrypt et multer garantissent respectivement la sécurité des mots de passe et la gestion des fichiers.

L'intégration de Git et GitHub a facilité la collaboration et le suivi des versions du projet, assurant ainsi une meilleure organisation du code source. De plus, l'utilisation de bibliothèques comme XLSX et PDFKit a permis d'automatiser certaines tâches essentielles, telles que l'importation de données et la génération de documents.

En conclusion, l'application développée répond aux exigences de gestion des professeurs avec une interface intuitive et une API performante. Son architecture modulaire et évolutive permettra d'ajouter facilement de nouvelles fonctionnalités à l'avenir, garantissant ainsi une solution pérenne et efficace.

Chapitre3 :

Tests et Résultats

I. Introduction

Le présent chapitre aborde les différentes étapes liées aux tests, aux résultats obtenus et aux perspectives d'amélioration du projet de gestion des professeurs. Après avoir détaillé les méthodes de tests appliquées pour garantir la fiabilité et la stabilité du système, nous présentons les résultats obtenus, notamment en ce qui concerne l'ergonomie et l'efficacité des interfaces utilisateur. Enfin, ce chapitre propose des pistes d'évolution pour enrichir le projet et répondre aux besoins croissants des utilisateurs, en renforçant notamment la gestion des utilisateurs, la sécurité et les fonctionnalités proposées.

II. Tests et validation

1. Tests unitaires

Les tests unitaires ont pour objectif de vérifier le bon fonctionnement des fonctions individuelles du système de gestion des professeurs.

1. Test de récupération de la liste des professeurs

Vérifie que la fonction qui récupère la liste des professeurs depuis la base de données retourne bien un tableau contenant les informations attendues.

2. Test d'ajout d'un professeur

S'assure que la fonction permettant d'ajouter un professeur dans la base de données enregistre correctement les informations fournies.

3. Test de connexion d'un utilisateur

Vérifie que la fonction d'authentification renvoie un jeton valide en cas d'identifiants corrects et un message d'erreur en cas d'identifiants incorrects.

4. Test de modification des informations d'un professeur

Vérifie que la mise à jour des données d'un professeur est bien prise en charge par la fonction dédiée et que les nouvelles valeurs sont enregistrées correctement.

2. Tests d'intégration

Les tests d'intégration permettent de vérifier l'interaction entre les différentes parties du système, notamment entre le frontend et le backend.

1. Test de récupération des professeurs via l'API

Vérifie que la requête **GET** envoyée à `/api/list-professeurs` retourne bien la liste des professeurs en format JSON et que le frontend affiche correctement les données reçues.

2. Test d'ajout d'un professeur via l'interface utilisateur

Vérifie que lorsqu'un administrateur ajoute un professeur via le formulaire du frontend, les données sont bien envoyées au backend, insérées en base et affichées dans la liste des professeurs.

3. Test d'authentification des administrateurs et professeurs

Teste si la connexion via l'interface utilisateur fonctionne correctement en envoyant une requête au backend et en vérifiant que la réponse inclut un jeton valide.

4. Test d'exportation des données en Excel

S'assure que lorsqu'un administrateur clique sur le bouton d'exportation, un fichier Excel contenant les informations des professeurs est bien généré et téléchargeable.

5. Test d'importation de professeurs via un fichier Excel

Vérifie que l'importation d'un fichier Excel via l'interface utilisateur envoie correctement les données au backend, qui les enregistre ensuite dans la base.

6. Test de génération des cartes professionnelles avec QR code

S'assure que la fonctionnalité de génération de cartes professionnelles permet bien de créer un QR code unique associé à chaque professeur et qu'il redirige vers les bonnes informations lorsqu'il est scanné.

III. Résultats obtenus

1. Interface utilisateur avec React.js

○ Authentification

La page d'authentification est dédiée à la connexion d'un professeur pour accéder à son profil ou d'un administrateur pour accéder à son tableau de bord, en saisissant son adresse e-mail et son mot de passe.



Figure 2 : Interface d'authentification

○ Récupération de mot passe

En cas d'oubli du mot de passe, l'utilisateur peut cliquer sur le lien "Mot de passe oublié". Il sera redirigé vers une page de récupération où il pourra saisir son adresse e-mail. Après avoir reçu un code de vérification par e-mail, il devra l'entrer pour pouvoir définir un nouveau mot de passe.



Figure 3 : La page de réinitialisation de mot passe

Figure 4:La page de vérification de code envoyer sur mail

Figure 5 : La page pour saisir un nouveau mot de passe

○ Admin

Après la connexion en tant qu'administrateur, un menu latéral est disponible pour naviguer entre les différentes interfaces de l'administration et permettre la déconnexion. L'interface affiche la liste des professeurs en récupérant leurs informations depuis la base de données. De plus, une fonctionnalité d'exportation en Excel est intégrée, permettant de générer et de télécharger un fichier .xlsx contenant la liste des professeurs et aussi télécharger les cartes professionnelles de tous les professeurs en format pdf.



Nom	Prénom	Email	Statut	Actions
Kabajj	Laila	kabajj@gmail.com	vacataire	Carte
Ahmed	Jadd	ahmed@gmail.com	permanent	Carte
Benani	Manal	benani@gmail.com	permanent	Carte
Dupont	Jean	jean.dupont@example.com	permanent	Carte

Figure 6 : La page principale de admin

○ Importation fichier Excel

Cette interface permet d'ajouter des professeurs dans la base de données à partir d'un fichier Excel grâce au bouton "Importer". Lors de l'importation, plusieurs vérifications sont effectuées afin d'assurer l'intégrité des données. Tout d'abord, les adresses e-mail des nouvelles entrées sont extraites, puis comparées avec celles déjà enregistrées dans la base de données. Si des doublons sont détectés, le processus est automatiquement interrompu afin d'éviter les enregistrements en double.



Figure 7 : Interface responsable de la gestion des fichiers Excel

○ Ajouter Professeurs

L'interface permet également d'ajouter un professeur manuellement via un formulaire dédié à la création de compte. Ce formulaire collecte les informations essentielles du professeur, allant de son nom jusqu'à la création de son mot de passe. Une fois le formulaire soumis, les données sont vérifiées avant d'être enregistrées dans la base de données, garantissant ainsi l'exactitude et la cohérence des informations.

Figure 8 : Interface responsable a l'ajout de Professeur


○ Profile Professeurs

L'interface permet la récupération et l'affichage des informations du professeur à partir de la base de données. L'image de profil est également chargée depuis la base de données ; si aucune image n'est disponible, une image par défaut est affichée. Un bouton "Modifier Profil" est mis à disposition pour rediriger l'utilisateur vers un formulaire où il pourra mettre à jour ses informations personnelles. De plus, un lien est fourni pour permettre au professeur de générer et de télécharger sa carte professionnelle au format PDF.



Figure 9 : Carte professionnelle au format PDF

Bienvenue



Nom: Kabajj

Prénom: Laila

Téléphone: 0678943566

Email: kabajj@gmail.com

Matières enseignées: web

Statut: vacataire

Carte Professionnelle

Laila Kabajj
Professeur
web

Télécharger ma carte

Figure 10 : Interface du profile d'un professeur

○ Modifier Profile

Le formulaire de modification du profil professeur permet de récupérer les informations du professeur depuis la base de données afin de pré-remplir automatiquement les champs avec ses données actuelles. Ainsi, l'utilisateur peut facilement mettre à jour ses informations. Lors de la soumission du formulaire, les données modifiées sont envoyées au serveur pour être enregistrées et mises à jour dans la base de données.

The image displays two side-by-side screenshots of a web application interface. The left screenshot, titled 'le profil de Kabajj Laila', shows a form with several input fields: 'Kabajj', 'Laila', 'kabajj@gmail.com', '0678943566', 'web', and 'vacataire'. At the bottom, there is a file upload section with a button 'Choisir un fichier' and the text 'Aucun fichier choisi'. The right screenshot, titled 'Modifier le mot de passe', shows a form with three password input fields: 'Mot de passe actuel', 'Nouveau mot de passe', and 'Confirmer le nouveau mot c'. Each password field has an eye icon for toggling visibility. At the bottom of this form are two buttons: 'Annuler' and 'Mettre à jour'.

Figure 11 : Interface de modification des informations d'un professeur

2. La gestion du projet

Tant que nous avons travaillé en équipe de quatre personnes sur ce projet, notre équipe s'est occupée uniquement de la gestion des professeurs. Nous avons décidé d'attribuer cette tâche à un seul binôme afin de gagner du temps, tandis que l'autre binôme a commencé à travailler sur un projet plus long et plus complexe.

Pour simplifier le travail, chacun de nous a pris une partie du projet. Lorsqu'un membre avançait sur sa tâche, il effectuait un commit pour celle-ci. Comme mentionné précédemment, nous avons travaillé sur une seule branche, mais chacun faisait ses propres commits.

Voici les commits des membres de l'équipe pour ce projet :

M. Sersif Abdeljalil

Est le créateur du projet, il l'a également lancé sur GitHub et a effectué le premier commit.

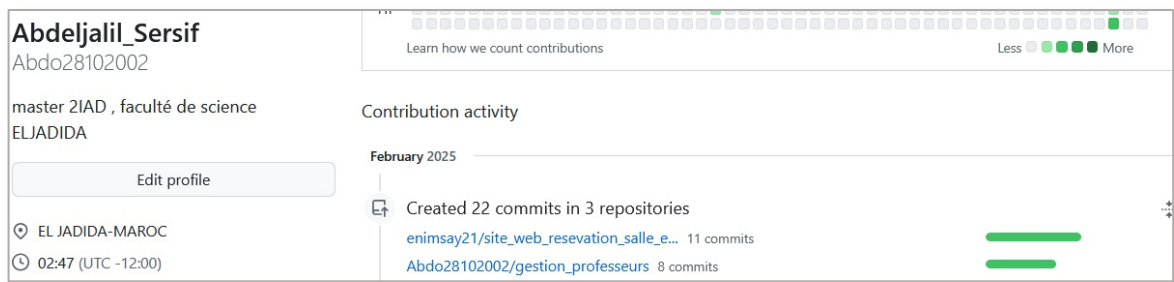


Figure 12 le nombre des commits de 1er collaborateur

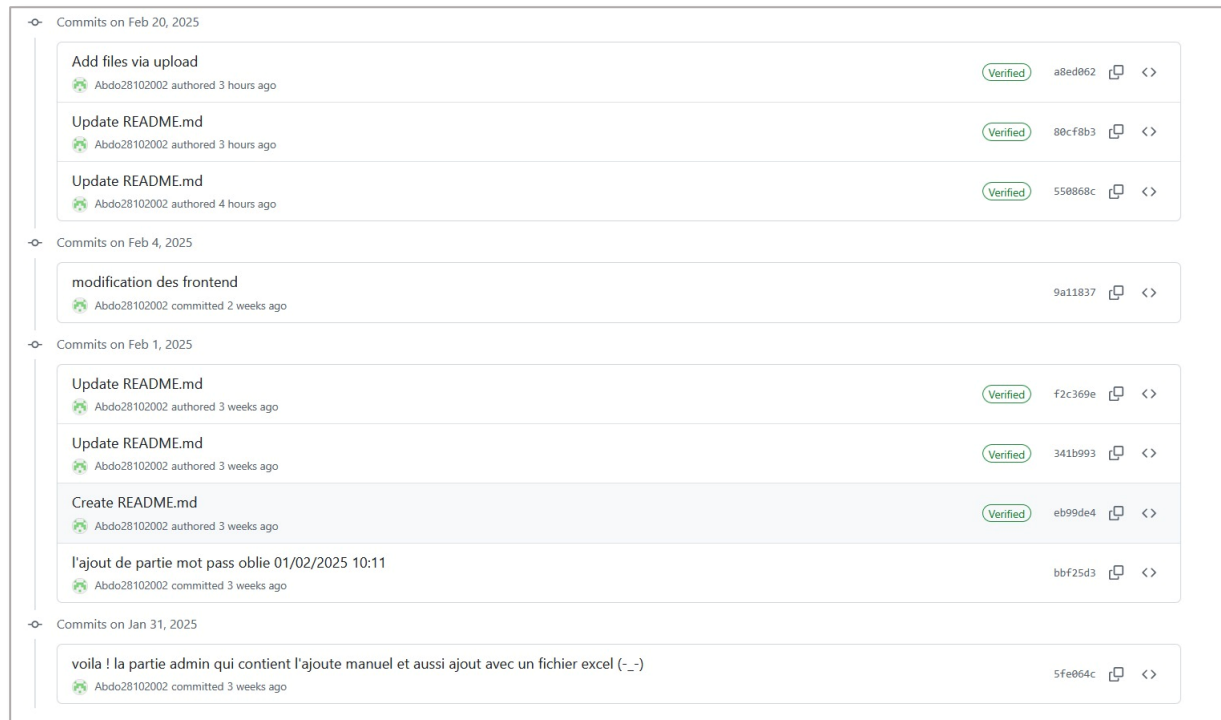


Figure 13 : les détails des commits pour 1er collaborateur

Mlle. Chahmi Nouhaila



Figure 14: le nombre des commit du 2ème collaborateur

Commits

master Nouna-N Jan 31 - Feb 21

Commits on Feb 20, 2025

- La version_Finale_3**
Nouna-N committed 2 hours ago e37d402
- La version_Finale_après les modification**
Nouna-N committed 2 hours ago f79b3e8
- La version_Finale_après les modification**
Nouna-N committed 3 hours ago cdede04

Commits on Feb 16, 2025

- modification carte pro**
Nouna-N committed 4 days ago 8120f5f

Commits on Feb 13, 2025

- version finale_1**
Nouna-N committed last week fec04d7

Commits on Feb 12, 2025

- version finale**
Nouna-N committed last week 3740898

Commits on Feb 10, 2025

- changement de style**
Nouna-N committed last week a956c95

Commits on Feb 4, 2025

- modification des cartes professionnelles**
Nouna-N committed 2 weeks ago e77d53d
- modification du style de profile**
Nouna-N committed 2 weeks ago 3623728

Commits on Feb 3, 2025

- modification du frontend**
Nouna-N committed 2 weeks ago 0288b40
- modification du frontend**
Nouna-N committed 2 weeks ago e9dc863

Commits on Jan 31, 2025

- les retouches de style**
Nouna-N committed 3 weeks ago 1ee4406
- les cartes professionnelles**
Nouna-N committed 3 weeks ago 1625f37

Figure 15 les details des commits du 2ème collaborateur

IV. Conclusion

En conclusion, ce chapitre a permis de mettre en évidence l'importance des tests pour assurer la qualité du projet, tout en montrant les résultats concrets obtenus en termes d'interfaces utilisateurs et de performance. Les perspectives d'amélioration soulignent la nécessité d'adapter le système aux nouvelles exigences des utilisateurs, avec des fonctionnalités avancées telles que la gestion des étudiants et un système d'authentification sécurisé. Ces améliorations contribueront à renforcer la pertinence et l'efficacité du projet dans le futur, tout en assurant une gestion optimale des informations académiques.

Conclusion Générale et perspectives

À travers ce travail, nous avons exploré les différentes étapes de conception et de développement d'une solution numérique dédiée à la gestion des professeurs. En réponse aux limites des méthodes traditionnelles, notre approche a permis de proposer un système automatisé et optimisé, facilitant l'organisation, l'accessibilité et la sécurisation des informations.

L'intégration des technologies web a joué un rôle central dans cette transformation, offrant une plateforme intuitive et performante qui simplifie les tâches administratives tout en garantissant une meilleure traçabilité des données. La mise en place de fonctionnalités essentielles, telles que l'authentification sécurisée, l'importation de données via Excel et la génération de cartes professionnelles avec QR code, a renforcé l'efficacité du système et son adaptabilité aux besoins des utilisateurs.

Bien que cette solution apporte une nette amélioration dans la gestion des professeurs, des perspectives d'évolution restent envisageables.

En conclusion, ce projet s'inscrit pleinement dans une démarche d'innovation numérique visant à moderniser la gestion administrative dans le domaine de l'enseignement. Il ouvre la voie à de futures améliorations et démontre l'importance des solutions numériques dans l'optimisation des processus organisationnels.

Dans le cadre de l'évolution du projet de gestion des professeurs, plusieurs améliorations peuvent être envisagées pour enrichir le système. Une des principales évolutions serait l'intégration de la gestion des étudiants, avec un module d'inscription permettant de renseigner les informations essentielles et un système d'attribution des modules selon le semestre. Un espace de consultation des données permettrait d'afficher la liste des étudiants inscrits, avec une fonctionnalité de recherche avancée. En parallèle, un système d'authentification pour enseignants et étudiants renforcerait la sécurité, tout en offrant un espace personnel sécurisé et une gestion des privilèges pour chaque utilisateur.