FACULTÉ DES SCIENCES
UNIVERSITÉ D'ANGERS

LERIA

Université d'Angers

Master 2 Informatique

Rapport de Projet

Application pour conception de maquettes de formation

Étudiants :
Reda Lasri
Ibrahima Sory Diallo
Habib Khteira

Encadrants :
David Lesaint
Corentin Behuet
Aurélien Simon

19 décembre 2025

Table des matières

1 Introduction Générale **2**
1.1 Contexte du Projet . . . 2
1.2 Problématique et Objectifs . . . 2

2 Concepts Fondamentaux et Modélisation **3**
2.1 Le Modèle de Données (Maquette) . . . 3

3 Base de Données **3**
3.1 Base de Données Externe . . . 3
3.1.1 Tables Exploitées . . . 3
3.1.2 Interface de Programmation (API) . . . 3
3.2 Base de Données Locale . . . 4

4 Réalisation **7**
4.1 Introduction . . . 7
4.2 Fonctionnement du framework Django . . . 7
4.2.1 Gestion d'API REST externes avec Django . . . 7
4.2.2 Base de données et ORM Django . . . 8
4.3 Fonctionnement du framework Vue.js . . . 8
4.4 Bibliothèque de modélisation graphique : Vue Flow . . . 8
4.5 Conteneurisation par Docker . . . 9
4.6 Gestion du projet : GitHub . . . 9

5 Spécifications Fonctionnelles Détaillées **9**
5.1 Visualisation des maquettes . . . 9
5.1.1 Vue Standard (Graph View) . . . 10
5.1.2 Vue Organisée . . . 11
5.1.3 Vue Colonnes . . . 11
5.1.4 Vue Tabulaire . . . 12
5.2 Modification des maquettes . . . 12
5.3 Découpage . . . 13
5.4 Gestion de la Mutualisation . . . 13
5.5 Fonctionnalités avancées de visualisation . . . 14
5.6 Espace étudiants . . . 14
5.7 Espace des Inscrits . . . 16

6 Conclusion **17**

1 Introduction Générale

1.1 Contexte du Projet

Ce projet a été réalisé au sein du Laboratoire d'Étude et de Recherche en Informatique d'Angers (LERIA), sous la supervision de David Lesaint, Corentin Behuet et Aurélien Simon. Il s'intègre dans une initiative de recherche plus large visant à moderniser et automatiser la gestion des emplois du temps universitaires.

Le laboratoire dispose déjà d'un écosystème logiciel, incluant un solveur de contraintes performant exposé via des services web. Cependant, il manquait une interface utilisateur intuitive permettant de définir les données d'entrée de ce système, c'est-à-dire la structure même des formations.

1.2 Problématique et Objectifs

La maquette d'une formation universitaire est un objet complexe. Elle ne se limite pas à une liste de cours, mais constitue une arborescence structurée incluant des semestres, des unités d'enseignement (UE), des cours (obligatoire ou optionnelles et des modalités. Chaque élément possède ses propres caractéristiques temporelles et pédagogiques.

L'objectif principal de ce stage était de concevoir une application web capable de :

1. **Modéliser visuellement** cette structure arborescente, en permettant aux responsables de formation de manipuler graphiquement les blocs pédagogiques.
2. **Gérer le découpage des modalités** : en permettant aux responsables des unités de découper le volume horaire de leurs modalités en un ou plusieurs blocs de séance.
3. **Identifier les mutualisations**, c'est-à-dire repérer et gérer efficacement les cours partagés entre plusieurs filières, un enjeu clé pour l'optimisation des ressources.
4. **Guider l'étudiant**, via un module d'inscription interactif qui s'assure que les choix d'options respectent les règles de validité pédagogique définies en amont.

2 Concepts Fondamentaux et Modélisation

2.1 Le Modèle de Données (Maquette)

Chaque maquette est composée des entités principales :

1. **Formation** : La racine de l'arbre.
2. **Step (Étape)** : Une subdivision temporelle ou logique.
3. **Conteneur (UE / Groupe)** :
   — *Unité d'Enseignement (UE)* : Regroupe des cours cohérents.
   — *Groupe d'Options* : Définit une règle de choix pour l'étudiant (ex : "Choisir 1 cours parmi la liste suivante").
4. **Cours** : L'unité atomique d'enseignement.
5. **Modalité (CM, TD, TP)** : Une modalité est liée à un cours et a un volume horaire.

3 Base de Données

L'architecture des données du projet est hybride, répartie entre une base de données externe et une base de données **locale**.

3.1 Base de Données Externe

Le système central repose sur une base de données PostgreSQL gérée par un serveur Flask.

3.1.1 Tables Exploitées

— `formations` : Table racine définissant les diplômes.
— `steps` : Représente les étapes temporelles (semestres/périodes) liées à une formation et une période.
— `courses` : Contient le catalogue des enseignements (Modules).
— `step_course_junction` : Table d'association gérant la relation "Many-to-Many" entre les étapes et les cours.
— `modalities` : Définit les types d'heures (CM, TD, TP) associés à un cours précis.

3.1.2 Interface de Programmation (API)

Notre application communique avec cette base via une liste stricte d'API REST, classés par nature d'opération :

1. Endpoints de Lecture (GET) Ces points d'entrée permettent de synchroniser l'affichage local avec l'état du serveur :
— `/api/formations/` : Liste toutes les formations disponibles.
— `/api/formations/{id}/content` : Récupère la structure hiérarchique complète (Arbre) d'une formation.
— `/api/steps/`, `/api/courses/`, `/api/modalities/` : Listent les entités atomiques.
2. Endpoints de Modification (POST, PUT, DELETE) Toute action d'édition dans l'interface graphique déclenche l'un de ces appels pour persister la modification sur le serveur distant :

- **Gestion des Étapes** :
  - `POST /api/steps/` (Création)
  - `PUT /api/steps/{id}` (Mise à jour et liaison à une formation)
  - `DELETE /api/steps/{id}` (Suppression)
- **Gestion des Cours** :
  - `POST /api/courses/` (Création)
  - `PUT /api/courses/{id}` (Modification libellé)
  - `DELETE /api/courses/{id}` (Suppression)
  - `POST /api/courses/assign-to-steps/{id}` (Association à un semestre via table de jonction)
- **Gestion des Modalités** :
  - `POST /api/modalities/` (Ajout d'un type d'heure)
  - `PUT /api/modalities/{id}` (Modification du volume horaire)
  - `DELETE /api/modalities/{id}` (Retrait)

3.2 Base de Données Locale

Localement nous maintenons une base de données locale (SQLite en développement, PostgreSQL en production), gérée via l'ORM de Django.

Le schéma se décompose en deux modules principaux :

1. Module Scolarité (Inscription) Ce premier ensemble de tables gère l'état civil et le parcours de l'étudiant :

- `Student` : Stocke l'identité de l'étudiant (Nom, Prénom, Email, Numéro Étudiant).
- `Registration` : Table de liaison représentant l'inscription d'un étudiant à une formation donnée pour une année. Elle contient également l'affectation aux groupes de TD/TP.
- `StudentChoice` : Enregistre les décisions pédagogiques de l'étudiant.

2. Module Enrichissement Pédagogique Ce second module permet d'ajouter des métadonnées qui ne sont pas présentes dans le système d'information central :

- `TeachingUnit (UE)` : Permet de définir localement des conteneurs logiques et d'y associer des crédits ECTS.
- `CourseGroup` & `CourseOption` : Ces tables sont essentielles pour modéliser les règles de choix complexes. Elles structurent l'offre d'options présentée dans le module d'inscription.

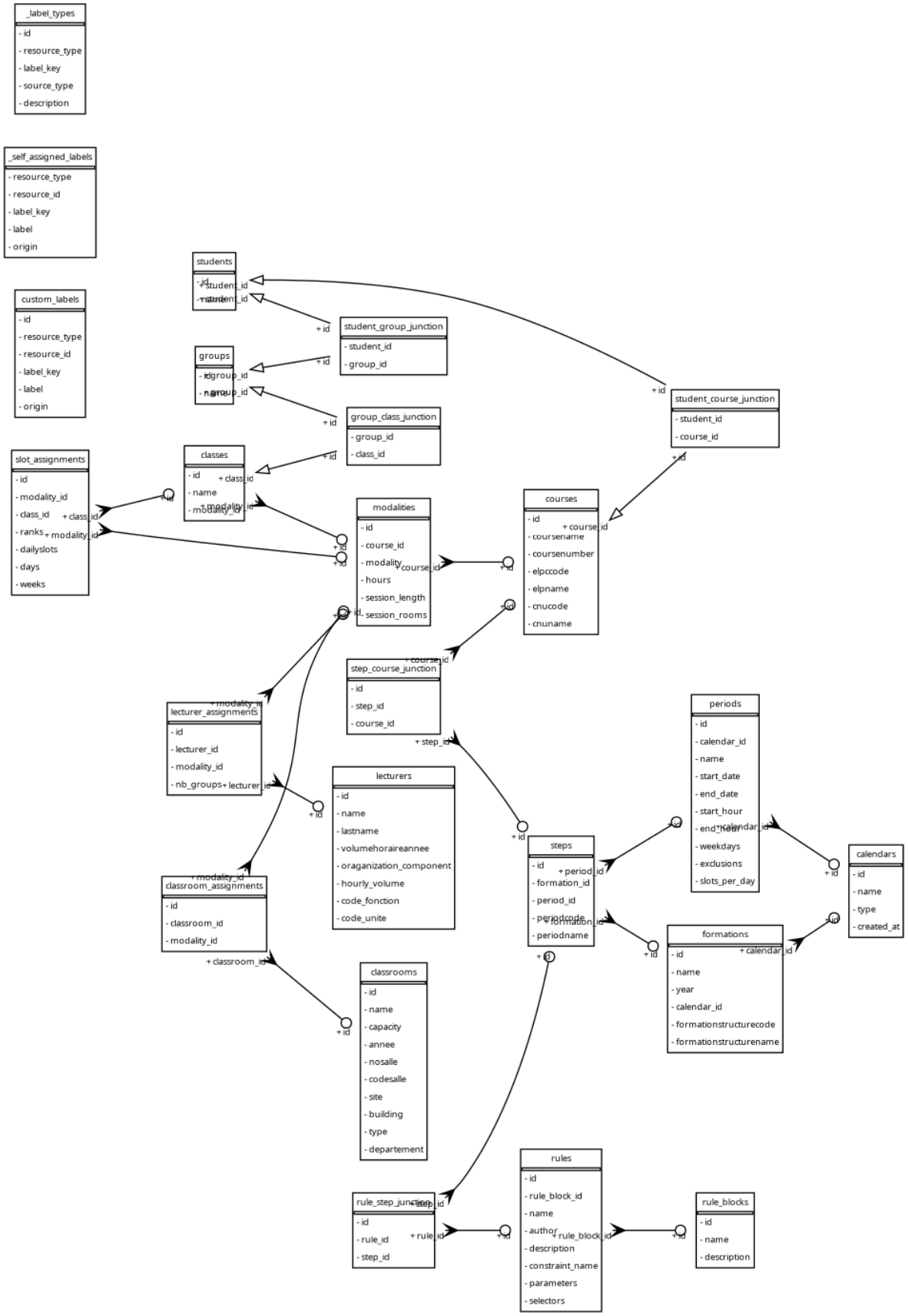

FIGURE 1 – Schema du base de donnée externe

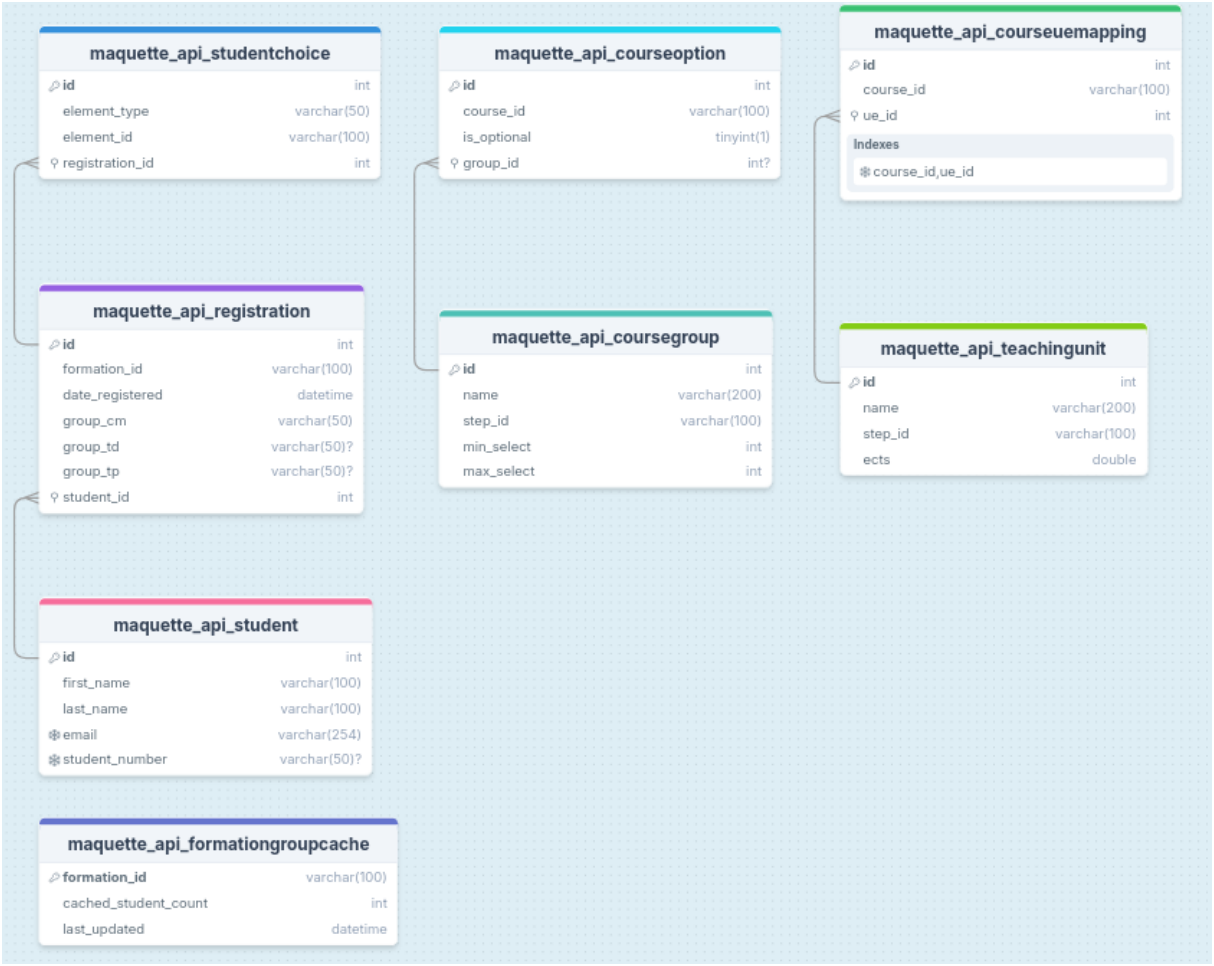

FIGURE 2 – Schema du base de donnée locale

4 Réalisation

4.1 Introduction

Afin de mener à bien les différentes phases du développement de l'application, nous avons eu recours à plusieurs outils et technologies qui nous ont permis d'atteindre nos objectifs tout en améliorant notre productivité. Cet ensemble d'outils comprend des frameworks logiciels ainsi que des services de collaboration, utilisés tout au long du projet pour maintenir un bon rythme de travail et résoudre efficacement les problèmes rencontrés.

Parmi les principaux outils utilisés, nous pouvons citer :

- Django comme framework pour le développement du backend ;
- Vue.js comme framework pour le développement du frontend ;
- **Vue Flow** pour la modélisation graphique et interactive des maquettes ;
- Docker pour assurer la conteneurisation et la gestion efficace des différents services de l'application ;
- GitHub comme outil de gestion de versions et de collaboration entre les membres de l'équipe.

Dans la suite, nous décrivons le fonctionnement de chacun de ces outils, leur intégration dans notre projet, ainsi que leur impact sur la productivité et la qualité du résultat final.

4.2 Fonctionnement du framework Django

Django est un framework web backend écrit en Python, conçu pour faciliter le développement rapide et sécurisé d'applications web. Il repose sur une architecture appelée MTV (Model–Template–View), qui est une variante du modèle MVC.

- **Modèle (Model)** : Le modèle représente la couche de données de l'application. Dans Django, les modèles sont définis sous forme de classes Python et sont directement liés à la base de données via l'ORM (Object-Relational Mapping). Chaque modèle correspond à une table de la base de données, et chaque attribut du modèle représente une colonne.
- **Vue (View)** : Les vues contiennent la logique applicative. Elles traitent les requêtes HTTP reçues, interagissent avec les modèles et retournent des réponses HTTP. Dans notre projet, les vues sont principalement utilisées pour exposer des services web sous forme d'API REST.
- **Template (Template)** : Cette couche gère la présentation. Bien que Django dispose d'un moteur de templates, celui-ci n'a pas été utilisé directement dans notre projet, puisque l'interface utilisateur est entièrement gérée par le frontend Vue.js.

Django intègre nativement de nombreux mécanismes essentiels tels que la gestion des routes, la sécurité et l'administration, ce qui réduit considérablement le temps de développement.

4.2.1 Gestion d'API REST externes avec Django

Dans notre projet, le framework Django n'a pas uniquement été utilisé pour gérer la logique métier et la base de données locale, mais également pour interagir avec des services externes via des API REST. Ces API, hébergées sur un autre serveur , permettent de

récupérer, mettre à jour et synchroniser des données provenant d'un système d'information externe.

Django agit ainsi comme un client REST, en envoyant des requêtes HTTP vers des endpoints externes. Ces requêtes peuvent être de différents types, notamment `GET` pour la récupération de données, `POST` pour l'ajout de nouvelles données, `PUT` pour la mise à jour, et `DELETE` pour la suppression.

Les données reçues depuis le serveur externe sont retournées au format JSON.

4.2.2 Base de données et ORM Django

Dans le cadre de notre backend, nous avons utilisé une base de données **SQLite**. Ce choix s'explique par sa simplicité de configuration et son adéquation avec ce projet.

SQLite est une base de données légère, intégrée directement au projet, ne nécessitant pas de serveur dédié. Django interagit avec cette base de données à travers son ORM, ce qui permet de manipuler les données sous forme d'objets Python sans écrire directement de requêtes SQL.

L'ORM Django facilite les opérations CRUD (Create, Read, Update, Delete) et assure la cohérence entre les modèles définis dans le code et la structure de la base de données.

4.3 Fonctionnement du framework Vue.js

Vue.js est un framework JavaScript frontend utilisé pour développer des interfaces utilisateur dynamiques et réactives. Il repose sur une architecture basée sur des composants, où chaque composant représente une partie indépendante de l'interface.

Chaque composant Vue.js est constitué de trois parties principales :
- **Template** : Définit la structure HTML du composant ;
- **Script** : Contient la logique JavaScript, incluant la gestion des données et des événements ;
- **Style** : Définit l'apparence graphique du composant.

Dans notre projet, Vue.js communique avec le serveur Django afin d'afficher dynamiquement les données, gérer les interactions utilisateur et assurer une expérience fluide sans rechargement de page.

4.4 Bibliothèque de modélisation graphique : Vue Flow

Pour la fonctionnalité centrale de notre application, à savoir la modélisation visuelle des maquettes, nous avons utilisé la bibliothèque **Vue Flow**. C'est une bibliothèque performante et modulaire pour Vue.js qui permet de créer des graphes interactifs basés sur des nœuds et des liens.

Nous avons choisi Vue Flow pour :
- Sa réactivité et son intégration native avec l'écosystème Vue 3.
- Sa flexibilité qui nous a permis de créer des **nœuds personnalisés** (Formations, Steps, Cours, Modalités) adaptés à notre modèle de données spécifique.
- Ses fonctionnalités intégrées de zoom et de mini-map.

4.5 Conteneurisation par Docker

Docker est une plateforme de conteneurisation qui permet d'encapsuler une application et ses dépendances dans des conteneurs isolés. Cette approche garantit que l'application fonctionne de manière identique quel que soit l'environnement d'exécution.

Dans notre projet, Docker a été utilisé pour conteneuriser le backend Django et le frontend Vue.js. Chaque service dispose de son propre conteneur, ce qui simplifie le déploiement, la configuration et la gestion des dépendances.

L'utilisation de Docker permet également d'améliorer la collaboration entre nous, en assurant un environnement de développement homogène pour tous.

4.6 Gestion du projet : GitHub

GitHub est une plateforme de gestion de versions basée sur Git, utilisée pour le suivi du code source et la collaboration en équipe. Elle permet de conserver l'historique des modifications, de gérer les branches et de résoudre les conflits de manière efficace.

Dans notre projet, GitHub a été utilisé pour :
- Le versionnement du code source ;
- Le travail collaboratif via des branches dédiées ;
- Le suivi des évolutions et des corrections.

Cette organisation nous a permis une meilleure coordination et une progression structurée du développement.

5 Spécifications Fonctionnelles Détaillées

Dans cette section, nous allons présenter les différentes fonctionnalités de l'application qui permettent de réaliser les objectifs du projet. Cette application offre une interface conviviale permettant aux enseignants et aux administrateurs de réaliser diverses opérations sur les maquettes et de gérer les groupes des étudiants inscrits, et aux étudiants de réaliser leur inscription et de faire les choix pédagogiques des cours optionnels.

5.1 Visualisation des maquettes

L'innovation majeure du projet réside dans son système de "Vues Synchronisées". L'utilisateur peut basculer instantanément entre quatre représentations des mêmes données, sans perte d'information.

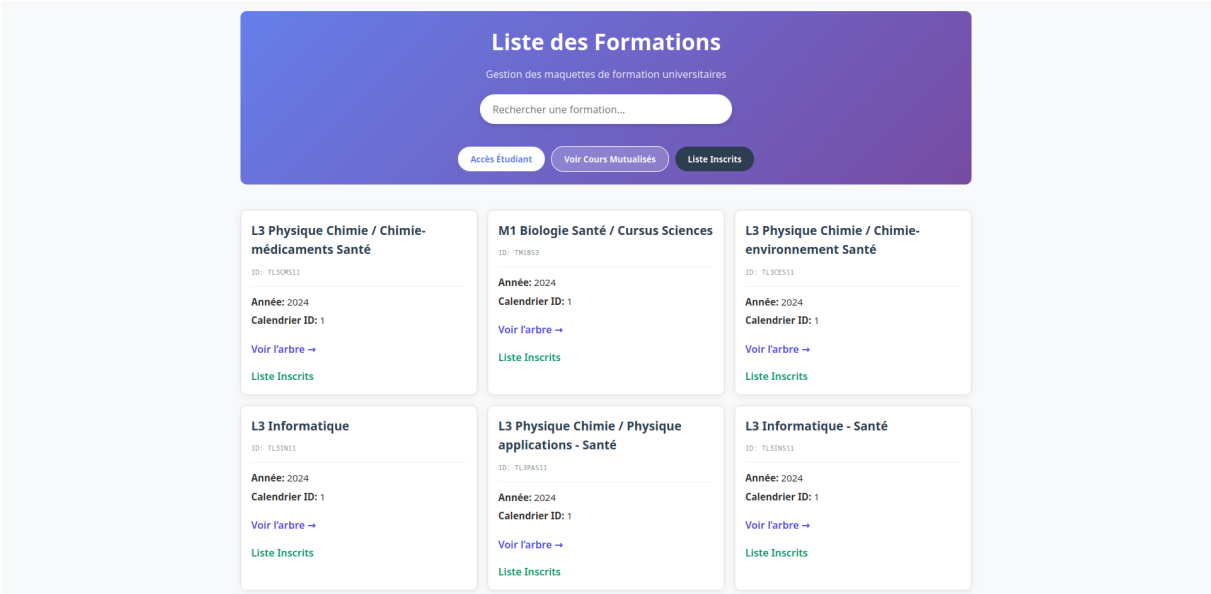

FIGURE 3 – Liste des formations

5.1.1 Vue Standard (Graph View)

C'est la vue par défaut destinée à la conception libre.

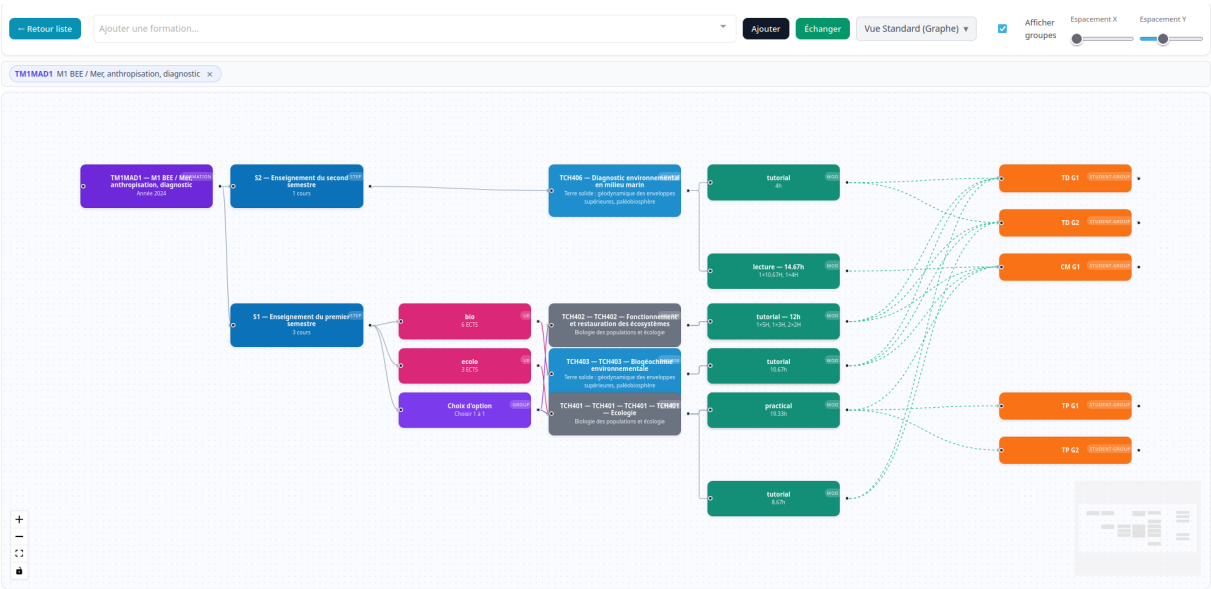

FIGURE 4 – Vue Standard

— **Fonctionnalités** :
  — Création de nœuds via le panneau latéral.
  — Création de liaisons (arêtes) par glisser-déposer entre les poignées des nœuds.
  — Déplacement libre des nœuds sur le canevas infini.
  — Visualisation des groupes.

5.1.2 Vue Organisée

Cette vue permet de représenter les formations avec une liaison directe entre les modalités et les étapes, sans passage obligatoire par l'affichage hiérarchique des cours et en minimisant les informations affichées, ce qui permet une meilleure compréhension des modalités par étape.

— **Visualisation standard** : Formation -> Étape -> UE/Options -> Cours -> Modalité -> Groupes
— **Visualisation organisée** : Formation -> UE/Options -> Cours -> Modalité -> Étapes

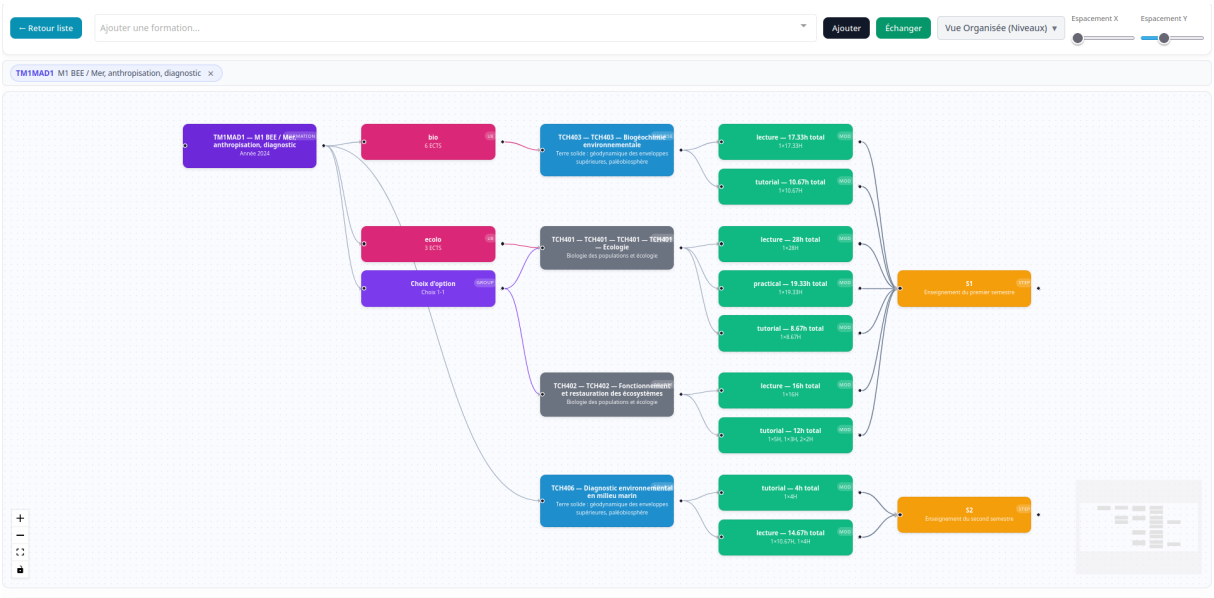

FIGURE 5 – Vue Organisée

5.1.3 Vue Colonnes

Il s'agit d'une version améliorée de la vue organisée. Au lieu de représenter les liens entre modalités et étapes par des traits, les modalités sont directement placées dans des colonnes, chaque colonne représentant l'étape (semestre) correspondante.

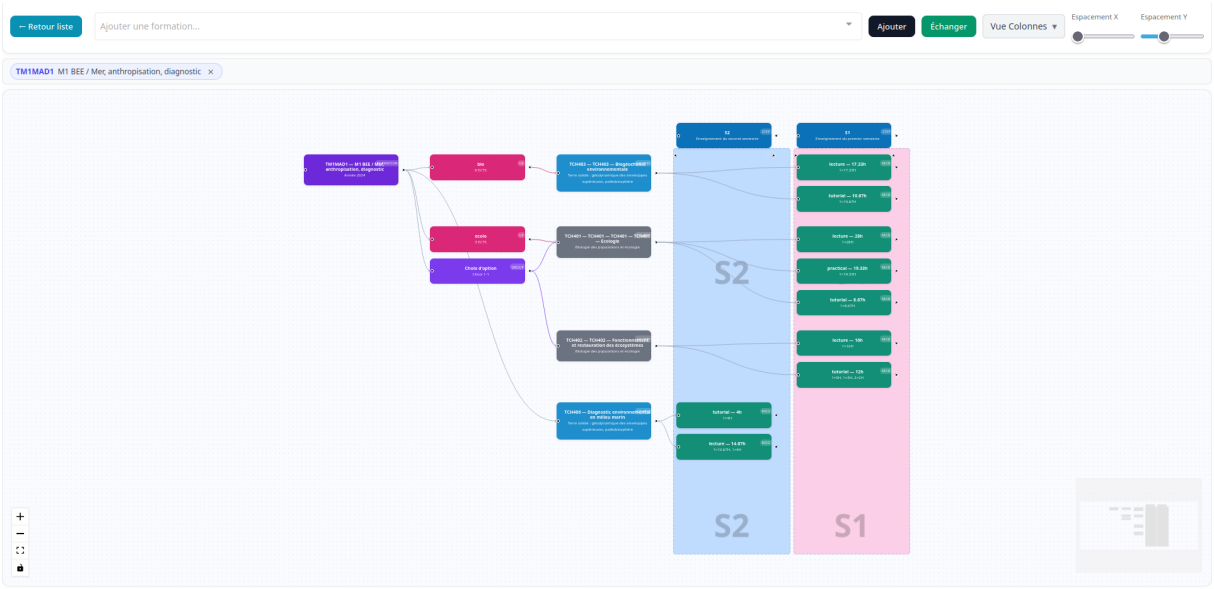

FIGURE 6 – Vue Colonne

5.1.4 Vue Tabulaire

Une vue basée totalement sur une représentation tabulaire classique, sans utilisation de graphes, pour une gestion rapide des données alphanumériques.

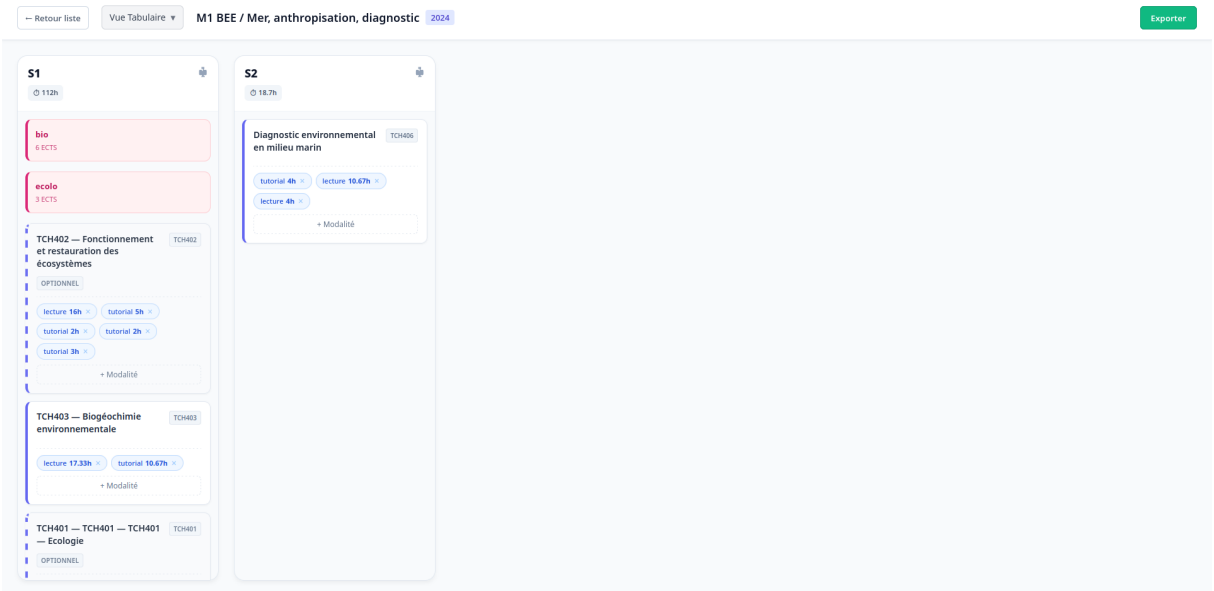

FIGURE 7 – Vue Tabulaire

5.2 Modification des maquettes

Dans toutes les pages de visualisation, l'utilisateur possède des fonctionnalités pour faire plusieurs modifications sur les maquettes.

— Modification des informations des éléments (Étape, UE, Groupes Options, Cours, Modalité)
— Création des UE et des groupes Options
— Créer/Supprimer les liens entre les différents éléments

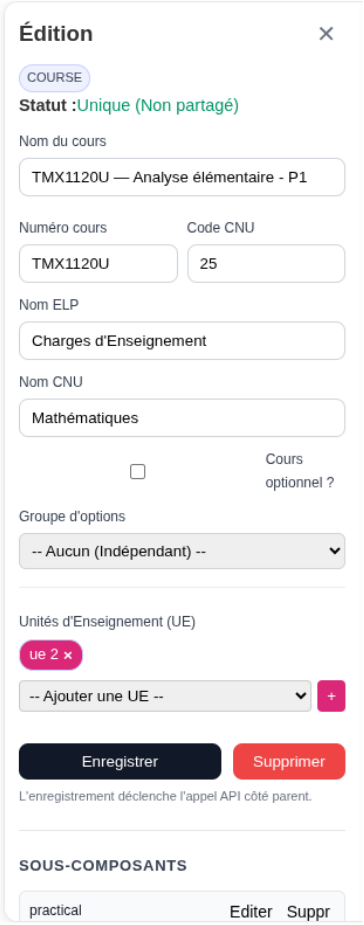

FIGURE 8 – Modification des maquettes

5.3 Découpage

L'une des fonctionnalités du projet est le découpage des modalités, qui permet au responsable d'une unité de découper les horaires d'une modalité en un ou plusieurs blocs. Dans la base de données, chaque bloc est enregistré comme une modalité distincte. L'interface graphique se charge ensuite de les regrouper visuellement si elles portent le même nom, pour une meilleure lisibilité.

5.4 Gestion de la Mutualisation

La mutualisation est gérée par référence. Dans la base de données, une modalité peut être liée à des cours situés dans des étapes de formations différentes ; elle est alors considérée comme mutualisée. Visuellement, l'application détecte ces occurrences multiples et applique un style spécifique (couleur différente) pour alerter l'utilisateur : "Attention, modifier ce cours impactera d'autres formations".

5.5 Fonctionnalités avancées de visualisation

Au-delà des vues principales, l'application propose des outils pour affiner l'affichage et travailler sur plusieurs contextes :

- **Filtrage par étape** : Permet de se concentrer sur une période spécifique (ex : Semestre 1) en grisant les éléments non concernés, ce qui améliore la lisibilité des graphes complexes.
- **Contrôle de l'espacement** : Des curseurs permettent de modifier dynamiquement l'écart vertical et horizontal entre les nœuds pour adapter la densité du diagramme aux besoins de l'utilisateur.
- **Multi-Formations** : Il est possible d'ajouter une ou plusieurs formations supplémentaires sur le même canevas pour les comparer visuellement au sein d'un espace de travail unifié.

5.6 Espace étudiants

Notre application permet aux étudiants à accéder à un espace pour réaliser leur inscription et faire le choix des cours optionnels.

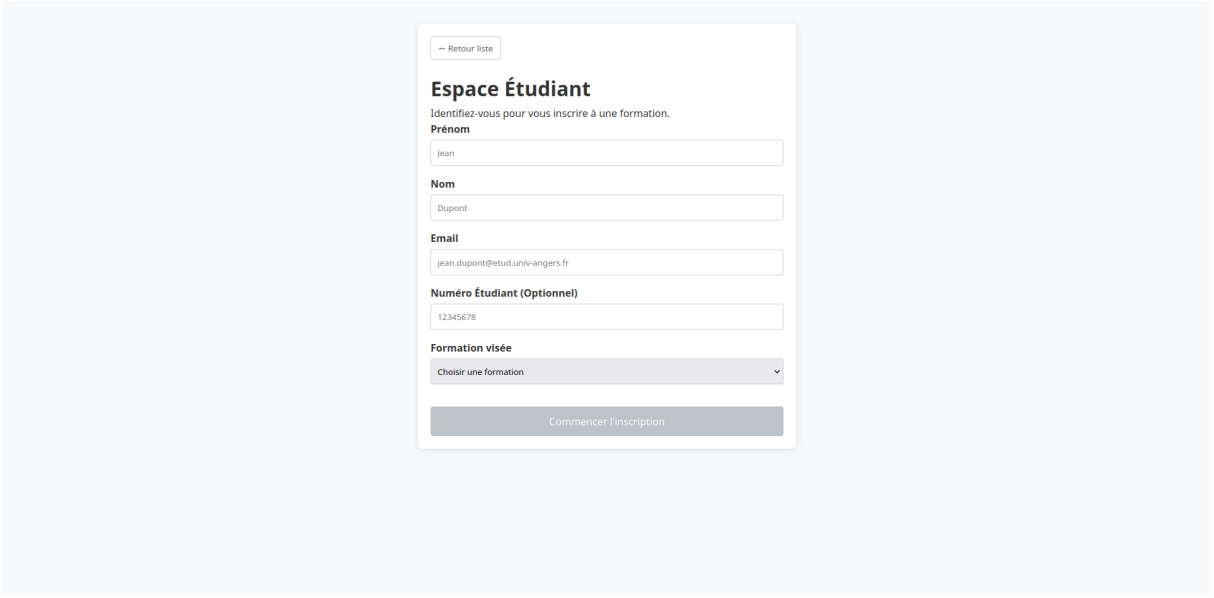

FIGURE 9 – Formulaire d'inscription

FIGURE 10 – Choix pédagogiques 1

FIGURE 11 – Choix pédagogiques 2

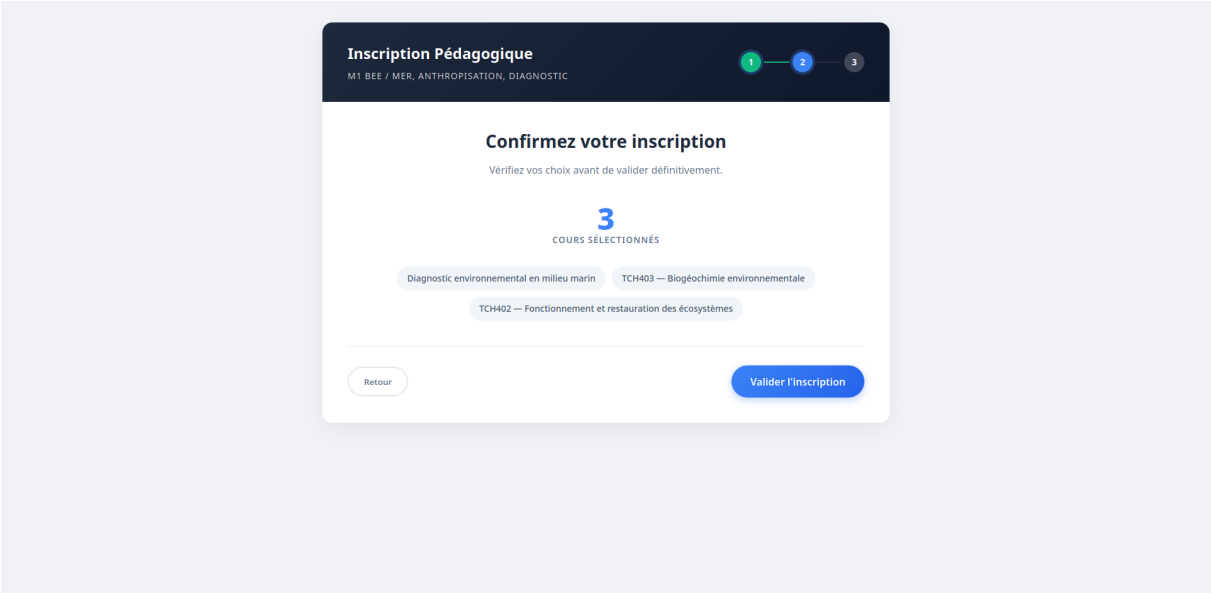

FIGURE 12 – Choix pédagogiques 3

5.7 Espace des Inscrits

Pour gérer les étudiants inscrits dans les différentes formations et gérer les groupes de ces étudiants, notre application propose un espace avec les fonctionnalités suivantes :

- **Affichage** : Liste des étudiants avec leurs groupes d'affectation.
- **Gestion des Groupes** : L'interface permet de gérer les groupes de TD et TP.
- **Automatisation** : Le découpage et la répartition initiale des étudiants dans les groupes sont effectués automatiquement par le programme, facilitant la création des classes équilibrées.

FIGURE 13 – Espace des inscrits

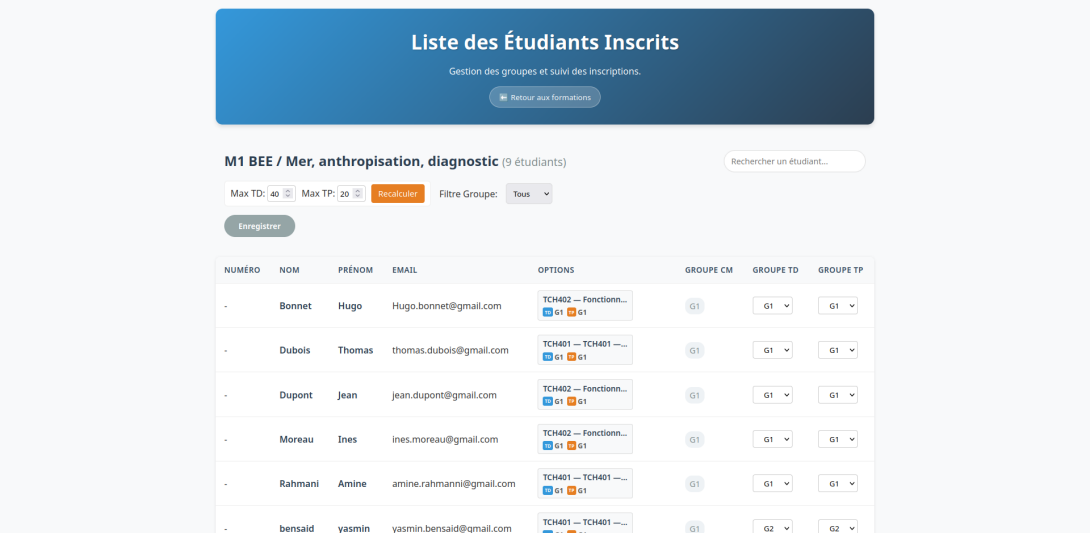

FIGURE 14 – Espace des inscrits d'une formation

6 Conclusion

Ce projet nous a permis de créer une application web fonctionnelle proposant plusieurs fonctionnalités innovantes. Nous avons implémenté des fonctionnalités essentielles qui facilitent les tâches des différents utilisateurs en termes de gestion des maquettes, découpage, inscription et gestion des groupes. Bien que nous ayons rencontré quelques défis en cours de route, nous avons réussi à surmonter les obstacles techniques et à développer des compétences précieuses en développement d'applications tout au long du processus.

En travaillant sur ce projet, nous avons consolidé nos compétences en conception de base de données et développement front et backend, avec la maîtrise du framework Django et Vue.js, les outils Docker et GitHub et l'utilisation des API REST. En fin de compte, ce projet a constitué une expérience d'apprentissage inestimable et nous avons pris beaucoup de plaisir à le réaliser. Nous sommes conscients que l'application pourrait encore être largement améliorée avec plus de temps. Par exemple :

- **Évolution de la base de données** : Actuellement, une étape dépend d'un cours. Il serait pertinent de modifier le schéma PostgreSQL pour établir un lien direct entre une modalité et une étape. Cela permettrait de répartir les modalités d'un même cours sur des étapes (semestres) différentes, ce qui n'est pas possible aujourd'hui.
- **Amélioration de l'UX** : L'expérience utilisateur pourrait être enrichie avec des raccourcis clavier pour les actions fréquentes et, surtout, un système de "Drag-and-Drop" dans la vue Colonnes pour déplacer facilement les modalités d'une étape à l'autre.
- **Correction de bugs** : Il reste sûrement quelques dysfonctionnements mineurs à corriger pour garantir une stabilité parfaite.

Nous sommes fiers de vous offrir cet outil qui vous aidera dans le futur. En conclusion, nous tenons à remercier tous ceux qui ont contribué à la réalisation de ce projet, notamment nos encadrants qui nous ont orientés et aidés à garder un bon rythme tout au long du projet. Ce projet a été une expérience enrichissante qui a renforcé nos compétences et notre passion pour le développement d'applications web.