

UE-L204 MINI-PROJET

Rapport intermédiaire

*Sylvain Chambon,
Jade Faroux,
Jeanne Salvadori,
Zoé Van De Moortele*

4 décembre 2025

1 La méthode de travail

1.1 Outils utilisés

- Teams pour la communication et le partage de fichiers;
- GitHub pour la mise en commun du code sur un dépôt et pouvoir mieux gérer les modifications via le système de branche;
- Forum de groupe de l'UE : communication et retour sur l'avancement du projet;
- Word pour une rédaction commune et L^AT_EX pour la finalisation des rapports;
- un système d'intelligence artificielle générative (Gemini) pour générer les entrées dans les tables.

1.2 Étude préliminaire

1.3 Scénario

Créer un site universitaire avec différents niveaux d'accès : étudiant et professeur, éventuellement un administrateur.

Les professeurs peuvent :

- Créer, modifier, supprimer des cours.
- Voir la liste d'étudiant inscrit.
- Gérer les inscriptions (accepter/refuser les étudiants dans leurs cours).
- Modifier leur profil (informations personnelles).

Les étudiants peuvent :

- Consulter l'ensemble des cours disponibles.
- Rechercher des cours (nom, professeur, domaine).
- S'inscrire à des cours.
- Modifier leur profil (informations personnelles).

Si assez de temps, mettre en place un système de note (prof donne des notes et les étudiants peuvent y accéder), et d'emploi du temps.

1.4 Objectifs adaptés à notre scénario

1.4.1 Page de connexion

- Formulaire d'entrée où l'utilisateur entre son login et mot de passe.
- Système vérifie si cet utilisateur existe dans la BDD.
- Le mot de passe doit être crypté.

- Si correct, accès au portail étudiant ou professeur.
- Si incorrect, message d'erreur.
- Changement de mot de passe lors de la première connexion (sécurité).

1.4.2 Page de recherche

- Champ de recherche de cours (par nom, professeur ou domaine).
- Doit interroger la BDD avec les critères de recherche.
- Afficher les résultats sous forme de tableau, ou message d'erreur si aucun résultat.

1.4.3 Page d'ajout de contenu

- Les professeurs ont une page dédiée pour ajouter, modifier ou supprimer leur cours.
- Les professeurs et étudiants peuvent modifier leurs informations personnelles (mail, tel, mot de passe...).
- Toutes ces modifications doivent être enregistrées dans la BDD.

1.4.4 Vérification des données

Avant d'enregistrer quoi que ce soit dans la BDD, tests de sécurité à effectuer systématiquement :

- Avant d'enregistrer quoi que ce soit dans la BDD, tests de sécurité à effectuer systématiquement :
- Vérifier que les champs obligatoires soient remplis
- Les formats des entrées doivent être corrects
- Pas de caractères dangereux (balise html)
- Messages d'erreur si une de ces vérifications est non conforme

2 Avancement du projet

2.1 Base de données

Nous avons modélisé notre base de données dans [diagrams.net](#). Nous avons pensé à définir deux utilisateurs pour notre base de données : l'enseignant et l'étudiant.

En structurant notre base, nous nous sommes aperçus que ces deux entités partageaient beaucoup d'attributs donc nous avons préféré définir l'entité plus générale d'utilisateur et définir les entités enseignant et étudiant comme des héritages de l'entité utilisateur avec certains attributs spécifiques.

Dans cette base de données, nous aurons évidemment aussi une entité cours.

Nous aurons enfin des tables de liaison :

- une table permettant d'associer des enseignants avec des cours (entité enseigne);
- un autre permettant d'associer des étudiants avec des cours (entité étudiant);
- une dernière permettant de définir des contraintes sur les inscriptions dans certains cours (entité prérequis).

2.1.1 Entité utilisateur

L'entité utilisateur aura les attributs (voir requête n° ??) :

- login : identifiant de connexion;
- mot_de_passe : mot de passe pour la connexion : il sera encrypté par l'application (ce n'est pas de la responsabilité de la BBD);
- mot_de_passe_provisoire : booléen servant de drapeau pour savoir si le mot de passe initial a été changé ou pas;
- nom, prenom, email : pour renseigner des éléments liés à la personne et son contact;
- role : trois rôles sont définis ici pour pouvoir tester les droits relatifs à chaque utilisateur de la BDD;
- date_creation pour stocker la date à laquelle l'utilisateur a été saisi dans la BDD;
- actif drapeau pour connaître si la personne est encore en activité dans la faculté.

Requête SQL n° 1 : Création de la table utilisateur

```
1 CREATE TABLE utilisateur (
2     id INT PRIMARY KEY AUTO_INCREMENT,
3     login VARCHAR(255) UNIQUE NOT NULL,
4     mot_de_passe VARCHAR(255) NOT NULL,
5     mot_de_passe_provisoire BOOLEAN DEFAULT TRUE,
6     nom VARCHAR(100) NOT NULL,
7     prenom VARCHAR(100) NOT NULL,
8     email VARCHAR(255) UNIQUE NOT NULL,
9     role ENUM('enseignant', 'etudiant', 'admin') NOT NULL,
10    date_creation TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
11    actif BOOLEAN DEFAULT TRUE
12 );
```

On se rend compte que tous ces attributs seront partagés à la fois par les enseignants et par les étudiants.

Un ajout d'un utilisateur dans la BDD peut donc se déclarer ainsi (voir exemple de requête n° ??)

Requête SQL n° 2 : Création d'un utilisateur dans la table

```
1 (1, 'turing', '<mot-de-passe>', 'Turing', 'Alan', 'alan.turing@univ.fr', '  
enseignant'),
```

2.1.2 Entité *enseignant*

L'entité enseignant hérite de l'entité utilisateur avec quelques spécificités :

- bureau : localisation de la salle de travail de l'enseignant;
- telephone : numéro;
- specialite : domaine d'expertise;
- statut : sous quel titre l'enseignant a-t-il été embauché.

On constate la présence d'une clé étrangère afin de lier l'entité enseignant avec une entité utilisateur existante (héritage).

Requête SQL n° 3 : Création de la table *enseignant*

```
1 CREATE TABLE enseignant (  
2     id INT PRIMARY KEY AUTO_INCREMENT,  
3     utilisateur_id INT UNIQUE NOT NULL,  
4     bureau VARCHAR(50),  
5     telephone VARCHAR(20),  
6     specialite VARCHAR(255),  
7     statut ENUM('titulaire', 'vacataire', 'contractuel') DEFAULT 'titulaire',  
8     FOREIGN KEY (utilisateur_id) REFERENCES utilisateur(id) ON DELETE CASCADE  
9 );
```

On ajoutera un enseignant ainsi dans la base (voir exemple de requête n° ??)

Requête SQL n° 4 : Création d'un enseignant dans la table

```
1 INSERT INTO enseignant (utilisateur_id, bureau, telephone, specialite, statut)  
2     VALUES  
3     (1, 'B101', '0102030401', 'Intelligence Artificielle', 'titulaire');
```

2.1.3 Entité *etudiant*

L'entité etudiant hérite de l'entité utilisateur avec quelques spécificités :

- le classique numero_etudiant, comme référence nationale;
- niveau qui référence si l'étudiant est en licence ou master et en quelle année;
- date_inscription qui peut-être de la date_creation si l'étudiant n'a pas validé ses frais de scolarité par exemple.

On constate ici aussi la présence d'une clé étrangère afin de lier l'entité étudiant avec une entité utilisateur existante (héritage, comme pour l'entité enseignant).

Requête SQL n° 5 : Création de la table etudiant

```
1 CREATE TABLE etudiant (
2     id INT PRIMARY KEY AUTO_INCREMENT,
3     utilisateur_id INT UNIQUE NOT NULL,
4     numero_etudiant VARCHAR(20) UNIQUE NOT NULL,
5     niveau ENUM('L1', 'L2', 'L3', 'M1', 'M2') NOT NULL,
6     date_inscription DATE NOT NULL,
7     FOREIGN KEY (utilisateur_id) REFERENCES utilisateur(id) ON DELETE CASCADE
8 );
```

Pour ajouter un étudiant dans la base, on pourra procéder ainsi (voir exemple de requête n° ??)

Requête SQL n° 6 : Création d'un étudiant dans la table

```
1 INSERT INTO etudiant (utilisateur_id, numero_etudiant, niveau, date_inscription)
2   VALUES
3     (12, '20250001', 'L3', '2024-09-01');
```

2.1.4 Entité cours

Requête SQL n° 7 : Création de la table etudiant

```
1 CREATE TABLE etudiant (
2     id INT PRIMARY KEY AUTO_INCREMENT,
3     utilisateur_id INT UNIQUE NOT NULL,
4     numero_etudiant VARCHAR(20) UNIQUE NOT NULL,
5     niveau ENUM('L1', 'L2', 'L3', 'M1', 'M2') NOT NULL,
6     date_inscription DATE NOT NULL,
7     FOREIGN KEY (utilisateur_id) REFERENCES utilisateur(id) ON DELETE CASCADE
8 );
```

3 Difficultés rencontrées / pas encore résolues

4 Projections pour la 2^e semaine