

Activité 3.1 : Consulter la liste des sujets

Objectif :

Permettre aux enseignants de voir une liste complète des PFE avec tous les détails nécessaires.

Étapes Logiques :

1. Connexion de l'enseignant :

- L'enseignant doit être authentifié. Le middleware `loggedMiddleware` assure que le token d'authentification est présent et valide.
- Le middleware `isTeacher` garantit que seul un enseignant peut accéder à cette route.

2. Récupération des données depuis la base :

- Utilisation du modèle PFE pour récupérer tous les sujets déposés par les étudiants.
- La méthode `populate` est utilisée pour inclure les détails de l'étudiant lié au PFE (comme son nom et son email). Cela permet d'avoir une vue complète des PFEs.

3. Structure des données retournées :

- Pour chaque sujet :
 - **Titre et description** du sujet sont inclus.
 - **Technologies utilisées** sont ajoutées pour permettre aux enseignants de filtrer ou de sélectionner selon leurs préférences.
 - **Informations sur l'étudiant** (nom, prénom, email) sont ajoutées pour identifier l'auteur du sujet.
 - **Nom et email de l'entreprise** sont fournis pour indiquer où le projet sera réalisé.
 - **Informations sur l'affectation** :
 - Si un enseignant est déjà affecté au sujet (`assignedTeacher`), son ID sera inclus.
 - Si aucune affectation n'est encore faite, la valeur est `null`.
 - **Informations sur la soutenance** :
 - Si une soutenance a été planifiée, elle sera incluse. Sinon, un message "Non planifié" est retourné.

4. Réponse formatée :

- Les données sont renvoyées sous forme d'un tableau JSON, avec un message confirmant que la liste a été récupérée avec succès.

Activité 3.2 : Choisir un ou plusieurs sujets

Objectif :

Permettre aux enseignants de choisir un ou plusieurs sujets disponibles pour encadrer les étudiants.

Étapes Logiques :

1. **Connexion et autorisation :**
 - Comme pour 3.1, seuls les enseignants connectés et autorisés peuvent effectuer cette action.
 2. **Récupération du PFE à affecter :**
 - L'identifiant du sujet est passé en paramètre (`id`) dans la route.
 - La base de données est interrogée pour trouver le sujet correspondant avec `PFE.findById`.
 3. **Vérification des conditions :**
 - Si le sujet n'existe pas dans la base, une réponse d'erreur (404) est renvoyée : "PFE non trouvé".
 - Si le sujet est déjà affecté à un autre enseignant, la requête est bloquée avec une réponse (400) : "Ce PFE est déjà choisi par un autre enseignant".
 - Pour cette vérification, on compare l'enseignant affecté au sujet (`assignedTeacher`) avec l'enseignant connecté (`req.user.id`).
 4. **Affectation du sujet :**
 - Si les vérifications passent, le champ `assignedTeacher` du sujet est mis à jour avec l'ID de l'enseignant connecté.
 - Le sujet est ensuite sauvegardé dans la base de données.
 5. **Réponse confirmant l'affectation :**
 - Une réponse JSON est renvoyée avec un message "PFE choisi avec succès", incluant les informations du sujet mis à jour.
-

Résumé des Relations Entre les Deux Activités

- **3.1 (Consulter la liste des sujets)** fournit à l'enseignant une vue d'ensemble des sujets disponibles, incluant des informations sur les sujets déjà affectés ou non.
- **3.2 (Choisir un sujet)** permet à l'enseignant de sélectionner un sujet parmi ceux affichés dans l'activité précédente.

- Ces deux activités forment un flux logique complet où l'enseignant peut explorer les sujets (3.1) et en choisir (3.2) en fonction de ses préférences et des conditions d'affectation.

code:

Modèle PFE ([models/Pfe.js](#))

```
const PfeSchema = mongoose.Schema({
  title: { type: String, required: true },
  description: { type: String, required: true },
  technologies: [String],
  nameCompany: { type: String, required: true },
  emailCompany: { type: String, required: true },
  student: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Student",
    required: true,
  },
  assignedTeacher: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Teacher",
    default: null,
  },
  defenseInfo: { type: String, default: "Non planifié" }, // Informations de soutenance
});
```

Fonctionnalité 3.1 : Consulter la liste des sujets

```
import PFE from "../models/Pfe.js";
import Student from "../models/Student.js";

export const getPFEDetailsForTeachers = async (req, res) => {
  try {
    const pfes = await PFE.find()
      .populate("student", "firstName lastName email") // Inclure les détails de l'étudiant
      .exec();
```

```

const formattedPFES = pfes.map((pfe) => ({
  id: pfe._id,
  title: pfe.title,
  description: pfe.description,
  technologies: pfe.technologies,
  nameCompany: pfe.nameCompany,
  emailCompany: pfe.emailCompany,
  student: pfe.student ? {
    name: `${pfe.student.firstName} ${pfe.student.lastName}`,
    email: pfe.student.email,
  } : null,
  assignedTeacher: pfe.assignedTeacher || null, // Ajouter l'enseignant affecté, si disponible
  defenseInfo: pfe.defenseInfo || "Non planifié", // Information sur la soutenance
}));

res.status(200).json({
  data: formattedPFES,
  message: "Liste des PFES récupérée avec succès.",
});
} catch (error) {
  console.error("Erreur lors de la récupération des PFES:", error);
  res.status(500).json({
    error: "Une erreur s'est produite lors de la récupération des PFES.",
    details: error.message,
  });
}
};

```

Route (PfeRoutes.js)

```

import express from "express";
import { getPFEDetailsForTeachers } from "../controllers/PfeController.js";
import { loggedMiddleware } from "../middlewares/authMiddlewares.js";
import { isTeacher } from "../middlewares/roleMiddlewares.js";

const router = express.Router();

router.get("/", loggedMiddleware, isTeacher, getPFEDetailsForTeachers);

export default router;

```

Fonctionnalité 3.2 : Choisir un ou plusieurs sujets

```
export const choosePFE = async (req, res) => {
  try {
    const { id } = req.params; // ID du PFE
    const teacherId = req.user.id; // ID de l'enseignant connecté (extrait du token)

    const pfe = await PFE.findById(id);

    if (!pfe) {
      return res.status(404).json({
        error: "PFE non trouvé.",
      });
    }

    if (pfe.assignedTeacher && pfe.assignedTeacher.toString() !== teacherId) {
      return res.status(400).json({
        error: "Ce PFE est déjà choisi par un autre enseignant.",
      });
    }

    pfe.assignedTeacher = teacherId; // Affecter l'enseignant au PFE
    await pfe.save();

    res.status(200).json({
      message: "PFE choisi avec succès.",
      data: pfe,
    });
  } catch (error) {
    console.error("Erreur lors de la sélection du PFE:", error);
    res.status(500).json({
      error: "Une erreur s'est produite lors de la sélection du PFE.",
      details: error.message,
    });
  }
};
```

Route

```
import { choosePFE } from "../controllers/PfeController.js";
```

```
router.patch("/:id/choice", loggedMiddleware, isTeacher, choosePFE);
```

GET /PFE (3.1 : Consulter la liste des sujets)

- Méthode : GET
- Headers :
 - Authorization : Bearer <token>
- Body : Aucun
- Réponse :

json

Copy code


```
{
  "data": [
    {
      "id": "63f1c6a98fb4c85f2c123456",
      "title": "Développement d'une application mobile",
      "description": "Application de gestion des événements.",
      "technologies": ["React Native", "Node.js"],
      "nameCompany": "XYZ Tech",
      "emailCompany": "contact@xyztech.com",
      "student": {
        "name": "John Doe",
        "email": "john.doe@student.com"
      },
      "assignedTeacher": null,
      "defenseInfo": "Non planifié"
    }
  ],
  "message": "Liste des PFEs récupérée avec succès."
}
```



PATCH `/PFE/:id/choice` **(3.2 : Choisir un sujet)**

- **Méthode :** `PATCH`
- **Headers :**
 - `Authorization` : `Bearer <token>`
- **Body :** Aucun
- **Exemple de réponse :**

json

 Copy code

```
{
  "message": "PFE choisi avec succès.",
  "data": {
    "id": "63f1c6a98fb4c85f2c123456",
    "title": "Développement d'une application mobile",
    "assignedTeacher": "63f1c6a98fb4c85f2c654321"
  }
}
```

Ce code implémente clairement les fonctionnalités pour l'activité 3.1 et 3.2.