
Compte Rendu d'Activité

Victor Thompson 3A

Sujet :	1
Gestion des Vacataires	1
Introduction	2
Mes tâches attribués	3
2.1 Filtrer les vacataires	3
2.2 Affecter/Désaffecter un vacataire à un ou plusieurs modules	3
2.3 Modifier un Vacataire	5

Sujet

Gestion des Vacataires

La réforme du BUT a vu exploser le nombre de nos vacataires. Le département informatique souhaite affecter un responsable des vacataires. Il lui faut une interface de gestion de ses vacataires qui soit en lien avec l'embryon de base existant déjà dans [Flop!Edt](#).

Introduction

Le projet auquel notre groupe a contribué prend ses racines dans la réforme du BUT, qui a entraîné une augmentation significative du nombre de vacataires. Le département informatique, confronté à la gestion complexe de cette main-d'œuvre, a exprimé le besoin pressant d'un outil pour superviser et administrer ses vacataires.

À mon arrivée, l'application était partiellement opérationnelle, mais des fonctionnalités cruciales restaient à développer, notamment l'affectation d'un vacataire à un cours. De plus, la gestion du projet sur Github laissait place à l'amélioration, et la documentation était plutôt limitée. Cette situation rendait la compréhension du projet compliquée. Pour résoudre ces problèmes, nous nous sommes concertés avec les autres alternants, en particulier Marco, pour mettre en place un environnement de développement local complet.

La première étape consistait à créer un environnement de développement local, ce qui signifiait avoir une version du front-end, de la base de données et de l'API sur nos machines. Marco a assumé la responsabilité de mettre en place un Docker pour atteindre cet objectif. Cette amélioration était essentielle pour faciliter le développement, la compréhension du code et la résolution des problèmes. Malgré le fait que l'implémentation du Docker a causé quelques soucis, nous avons pu chacun développer plus efficacement nos tâches.

Mes tâches attribués

2.1 Filtrer les vacataires

Ma première tâche consistait à développer un système de filtrage des vacataires, une fonctionnalité importante pour la gestion efficace du nombre de vacataires en constante évolution. Pour ce faire, j'ai entrepris de nombreuses recherches pour comprendre à nouveau le fonctionnement d'Angular, ainsi que les meilleures méthodes de filtrage des données.

Au cours de ces recherches, j'ai exploré diverses approches, dont l'une impliquait la mise en place d'une barre de recherche dynamique pour les noms des vacataires. Cette barre de recherche devait fournir une liste de noms de vacataires en autocomplétion à mesure que l'utilisateur tapait. Pour ce faire, j'ai envisagé l'utilisation de différentes bibliothèques et plugins. Cependant, cette phase d'exploration a pris du temps, et j'ai finalement décidé de transférer cette tâche à mon collègue Michele.

Cette décision s'est avérée plus logique, car Michele travaillait déjà sur une fonctionnalité similaire concernant les filtres des modules. Transférer la tâche m'a permis de me concentrer sur d'autres aspects du projet, même si cela a représenté un détour dans mon travail.

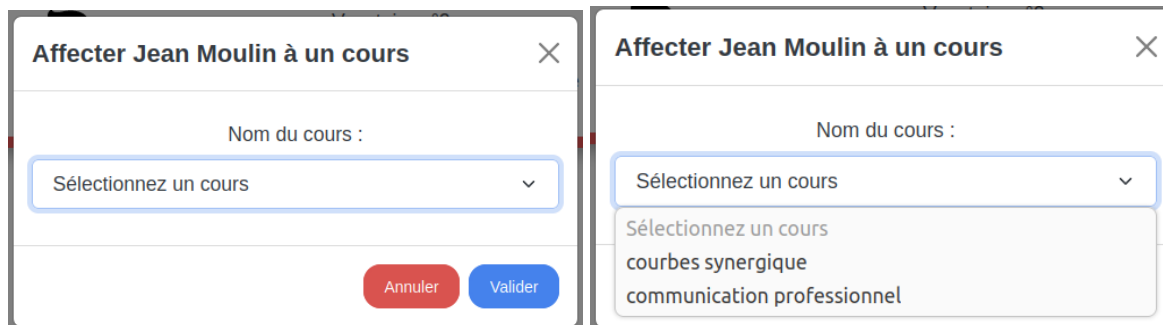
2.2 Affecter/Désaffecter un vacataire à un ou plusieurs modules

Issue : [Développer la fonctionnalité : Affecter/Désaffecter un Vacataires](#)

L'une de mes missions principales était de mettre en place les fonctionnalités essentielles permettant l'affectation et la désaffectation des vacataires à des cours, tant du côté front-end que du back-end. Voici un aperçu plus détaillé des étapes que j'ai suivies pour accomplir cette mission :

Front-End :

Modals : J'ai créé deux modals à l'aide de Bootstrap dans le fichier `liste-vacataires.component.html` pour gérer l'affectation et la désaffectation des vacataires. Le premier modal permet de sélectionner les cours auxquels un vacataire serait affecté, cours créés depuis la page de gestion des cours, tandis que dans le second modal, l'utilisateur avait la possibilité de désaffecter le vacataire du cours sélectionné. Ce second modal est accessible uniquement si le vacataire est déjà affecté à un cours.



Fonctions : Dans le fichier `liste-vacataires.component.ts`, j'ai implémenté les fonctions `affecterVacataire` et `desaffecterVacataire` pour gérer respectivement l'affectation et la désaffectation des vacataires. Ces fonctions sont appelées lorsque l'utilisateur interagit avec les modals. J'ai veillé à ce que ces fonctions communiquent correctement avec le service associé.

```

affecterVacataire(vacataireId: string, nomCours: string) {
  console.log(nomCours);
  const vacataire = this.vacataires.find(v => v._id === vacataireId);

  if (vacataire && nomCours && !vacataire.modules.includes(nomCours)) {
    this.vacatairesService.affecterVacataire(vacataireId, nomCours).subscribe(() => {
      vacataire.modules.push(nomCours);
      this.errorMessage = null;
      // Recharge la liste des vacataires après l'affectation
      this.vacatairesService.getVacataire().subscribe((data: unknown) => {
        this.vacataires = data as Vacataire[];
        // Ferme le modal ici, car il n'y a pas d'erreur
        const modal = document.getElementById('exampleModalToggle2-' + vacataireId);
        if (modal) {
          modal.querySelector('.btn-close')?.dispatchEvent(new Event('click'));
        }
      });
    });
  }, (error) => {
    console.error(error);
  });
} else {
  this.errorMessage = "Ce cours est déjà affecté.";
}
}

```

Back-End (Service) :

Service Angular : Dans le fichier `vacataires.service.ts`, j'ai créé les fonctions `affecterVacataire` et `desaffecterVacataire` pour gérer les requêtes vers l'API. Ces fonctions sont responsables de l'envoi des données au serveur et de la gestion des réponses.

```

affecterVacataire(id: string, nomCours: string): Observable<any> {
  const url = this.apiUrl + '/affecterVacataire/' + id
  return this.http.patch(url, { nomCours });
}

desaffecterVacataire(id: string, nomCours: string): Observable<any> {
  const url = `${this.apiUrl}/desaffecterVacataire/${id}`;
  return this.http.patch(url, { nomCours });
}

```

Back-End (API) :

Contrôleur API : Au niveau de l'API, précisément dans le fichier `vacataires.controllers.js`, j'ai mis en place les fonctions `affecterVacataire` et `desaffecterVacataire` pour répondre aux requêtes associées. Ces fonctions traitent les requêtes provenant du front-end, mettent à jour le statut du vacataire en conséquence, et gèrent l'affectation et la désaffectation des vacataires aux cours.

```

module.exports.affecterVacataire = async (req, res) => {
  try {
    const { id } = req.params;
    const { nomCours } = req.body;

    // Met à jour le statut du vacataire
    const updatedVacataire = await VacataireModel.findByIdAndUpdate(
      id,
      { status: "affecté" },
      { new: true }
    );

    if (!updatedVacataire) {
      return res.status(404).json({ message: 'Vacataire non trouvé' });
    }

    // Ajoute le cours à la liste des modules du vacataire
    updatedVacataire.modules.push(nomCours);
    // Sauvegarde les modifications
    const savedVacataire = await updatedVacataire.save();

    res.status(200).json(savedVacataire);
  } catch (err) {
    res.status(400).json(err);
  }
};

```

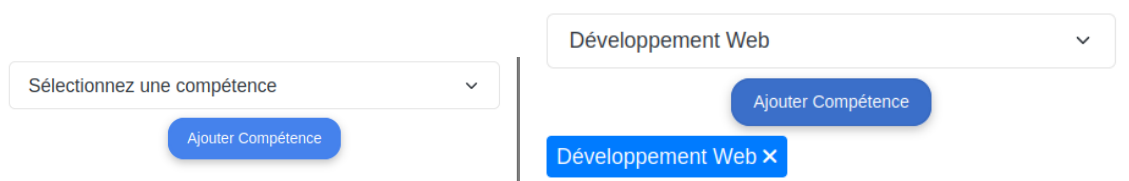
2.3 Modifier un Vacataire

Issue : [\[US\] : CRUD des Vacataires](#)

Dans le but de permettre la modification des informations des vacataires, j'ai apporté des améliorations à l'application, tant du côté du front-end que de l'API. Voici un aperçu des principales étapes que j'ai réalisés :

Front-End :

Gestion des Compétences : Afin d'améliorer la cohérence des données, j'ai remplacé la saisie de compétences sous forme de texte par un menu déroulant proposant une liste de compétences prédéfinies afin de d'éviter de saisir n'importe quoi, ou de saisir 2 fois la même compétence mais écrit différemment..



Ajout et Suppression de Compétences : J'ai introduit la possibilité d'ajouter et de supprimer des compétences sous forme de pastilles, offrant ainsi une manière conviviale de gérer les compétences.

```
addSkill(): void {
  if (!this.form.skills) {
    this.form.skills = [];
  }

  if (this.selectedSkill && !this.form.skills.includes(this.selectedSkill)) {
    this.form.skills.push(this.selectedSkill);
  }
}
```

```
removeSkill(skill: string) {
  // Supprime la compétence du tableau
  const index = this.form.skills.indexOf(skill);
  if (index !== -1) {
    this.form.skills.splice(index, 1);
  }
}
```

Fonctions de Modification : J'ai développé des fonctions de modification des vacataires au sein du fichier `liste-vacataires.component.ts`. Ces fonctions couvrent à la fois la mise à jour des informations personnelles et la gestion des compétences.

Back-End (Service) :

Service Angular : Dans le fichier `vacataires.service.ts`, j'ai créé des fonctions pour gérer les requêtes de modification des vacataires, permettant ainsi une interaction avec l'API.

Back-End (API) :

Contrôleur API : Au niveau de l'API, plus précisément dans le fichier `vacataires.controllers.js`, j'ai mis en place des fonctions pour traiter les demandes de modification des informations des vacataires.

J'ai principalement dirigé cette tâche en solo, bien que Christopher m'ait fourni certains codes qu'il avait déjà écrits en local pour m'assister. Il est important de noter que ces améliorations ont également été intégrées au formulaire de création de vacataire, car ce dernier a été modifié pour inclure la gestion des compétences/skills sous forme de listes.