

Rapport Individuelle SAE 3.01 PARTIE 3

1) Synthèse :

Durant cette dernière partie de SAE, nous devons récupérer le travail fait en amont (préparation de la base de données et de la partie algorithmique) et nous devons lier le tout a une application web fonctionnelle.

2) Analyse:

Durant cette deuxième partie de SAE, je me suis personnellement occupé de répartir les tâches et prendre certaines décisions en tant que chef de projet. De plus, j'ai également participé au développement de l'application en créant plusieurs pages html, et également en implémentant plusieurs liaisons front-end / back-end a travers le fichier views.py.

Pour les difficultés rencontrées, j'ai personnellement rencontré des difficultés au niveau du fichier views.py, c'est le fichier le plus conséquent de cette application et c'était aussi celui qui m'a pris le plus de temps. En effet l'implémentation de fonctionnalités n'était pas toujours évidente à cause de l'implémentation des pages et de la base de données qui avaient été faites juste avant.

Pour donner un exemple concret, une fonction qui ma prise plusieurs séance de SAE a finir a été celle de l'ajout de campagnes, en effet, en ajoutant une campagne, nous devons vérifier plusieurs choses telle que le budget, vérifier si la plateforme était disponible et si le personnel choisis était disponible à ces dates, vérifier les intervalles de maintenances et si il y avait déjà eu une maintenance de faite sur cette plateforme. Tout cela ajouté l'un après l'autre a fait que la fonction ajouter campagnes est assez conséquentes et est sûrement celle qui m'a posé le plus de problème.

Partie de la fonction ajoute campagne, qui vérifie le conflit de plateforme et personnel :

```
# 1. Vérifier le conflit de plateforme
campagnes_existantes_plateforme = Campagne.query.filter_by(
    idPl=plateforme_selectionnee.idPl).all()
for campagne_existante in campagnes_existantes_plateforme:
    date_debut_existante = campagne_existante.date_debut
    date_fin_existante = date_debut_existante + timedelta(
        days=campagne_existante.duree)
    if max(date_debut_nouvelle,
        date_debut_existante) < min(date_fin_nouvelle,
        date_fin_existante):
        flash(
            f"La plateforme '{plateforme_selectionnee.nom}' est déjà réservée pour la période du {date_debut_existante.strftime('%d/%m/%Y')} au {date_fin_existante.strftime('%d/%m/%Y')}.",
            "danger")
    return render_template('add_campagne.html', form=form)

# 2. Vérifier le conflit de personnel
personnel_selectionne_id = form.personnel_implique.data
for pers_id in personnel_selectionne_ids:
    personnel = Personnel.query.get(pers_id)
    for campagne_existante in personnel.campagnes:
        date_debut_existante = campagne_existante.date_debut
        date_fin_existante = date_debut_existante + timedelta(
            days=campagne_existante.duree)
        if max(date_debut_nouvelle,
            date_debut_existante) < min(date_fin_nouvelle,
            date_fin_existante):
            flash(
                f"Le membre du personnel '{personnel.nom}' est déjà occupé sur une autre campagne durant cette période du {date_debut_existante.strftime('%d/%m/%Y')} au {date_fin_existante.strftime('%d/%m/%Y')}.",
                "danger")
    return render_template('add_campagne.html', form=form)
```

3) Démonstration de compétences:

AC21.01 | Élaborer et implémenter les spécifications fonctionnelles et non fonctionnelles à partir des exigences

Chaque page est fonctionnelle, et chaque exigence a été remplie

AC21.03 | Adopter de bonnes pratiques de conception et de programmation

Séparation et rangement de code afin de rendre le tout plus clair

AC23.01 | Concevoir et développer des applications communicantes

Création d'une application utilisant des communications client-serveur

AC26.03 | Mobiliser les compétences interpersonnelles pour travailler dans une équipe informatique

Utilisation de github ainsi que différents moyens de communication tels que Discord