RAPPORT
PROJET DJANGO

PSI L3 CL

KHAYEF-ALLAH Ayoub
40000476
BOUABDELLAH BOURSALI Amel
40000259
EL TEWIL Ahmed
40008003

TABLE DES MATIERES

I. INTRODUCTION *2*
I. ARCHITECTURE TECHNIQUE *4*
a. Description du Framework **4**
b. Mise en place de la base de données **5**
II. FONCTIONNALITES CLES *6*
a. Enregistrement de nouvelles données **6**
b. Suppression des informations **6**
c. Importation et exportation de données 7
d. Gestion des réservation **8**
III. DEFIS RENCONTRES ET SOLUTIONS APPORTEES *9*
a. Problèmes de sécurité des données **9**
b. Difficultés rencontrées lors de l'implémentation de fonctionnalités spécifiques **9**
c. Problème de base de données **10**
d. Problème de réservation **10**
IV. CONCLUSION *12*

I. INTRODUCTION

Le monde de l'éducation est en constante évolution et il est important de disposer de technologies modernes pour gérer les opérations quotidiennes. L'Université Paris Nanterre est un établissement prestigieux qui accueille des milliers d'étudiants chaque année. Pour gérer efficacement les informations sur les étudiants, les bâtiments et les professeurs, il est important d'avoir un système de gestion des informations à jour et facile à utiliser. L'utilisateur doit s'inscrire en fonction de son profil et il pourra par la suite avoir accès à ses informations personnelles, et mener plusieurs actions (réservation de salle...)

Fonctionnalités principales :

- Gestion des salles : permettre de consulter la liste des salles et d'en ajouter ou supprimer, et d'ajouter des détails sur leur capacité et d'autres informations pertinentes.
- Gestion des étudiants/professeurs : ils pourront réserver des salles et consulter leurs historique de réservation. Ils pourront également éditer leur profil.
- Gestion des réservations : les enseignants et les étudiants pourront consulter la disponibilité des salles et réserver des salles pour les cours ou des travaux de groupes. Et en fonction des salles, ils pourront choisir le matériel qu'il souhaite utiliser (micro, ordinateurs, chimie ...). Une fois la réservation faite, un mail de confirmation sera envoyé, et une modification/suppression de la réservation sera possible.
- Gestion des bâtiments : Action réalisés par les administrateurs . Ils pourront ajouter, supprimer des bâtiments et modifier des horaires d'ouverture du bâtiments

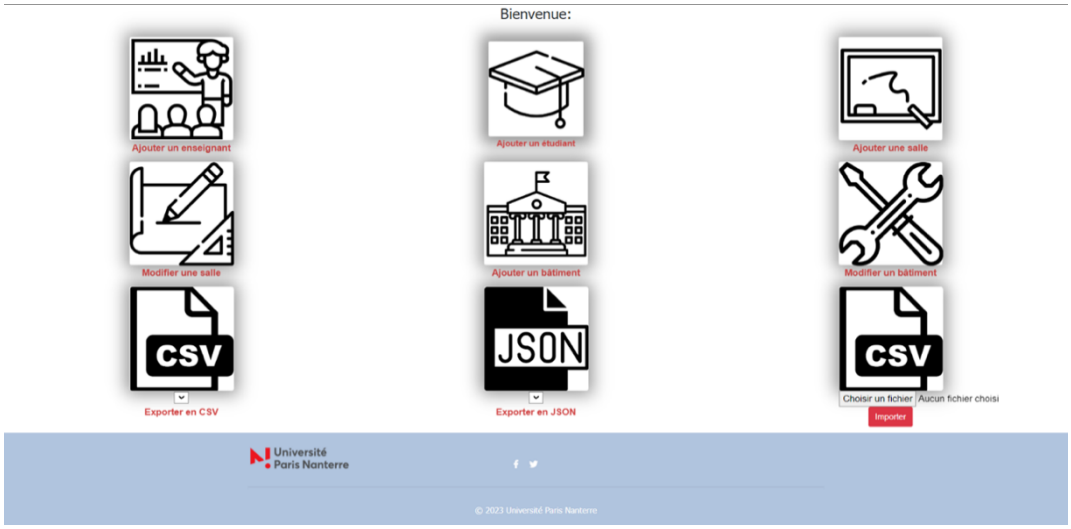
C'est pourquoi nous avons développé un site web pour permettre à l'Université Paris Nanterre de gérer les informations sur les étudiants, les bâtiments, les salles et les professeurs de manière efficace. Ce site web a été conçu en utilisant le Framework Django, un outil de développement de logiciels open source populaire, avec une base de données Sqlite. Il fournit une interface conviviale pour les utilisateurs de l'université pour enregistrer, afficher et mettre à jour les informations sur les étudiants, les bâtiments, les salles et les professeurs. De plus, il permet aux utilisateurs d'effacer des informations obsolètes ou incorrectes pour maintenir une base de données actualisée. Les utilisateurs peuvent également exporter et importer les données via CSV pour une utilisation ultérieure.

Toujours dans l'optique de garantir une base de données actualisés, nous avons créé une page contenant toutes les informations personnelles de chaque utilisateurs afin que ces derniers puissent eux-mêmes les modifier.

En fonction du type d'utilisateur (étudiants, professeurs ou administrateurs), ils pourront accéder directement à une pages qui leur est dédié, qui propose des fonctionnalités propre pour chaque profil d'utilisateurs.

Dans ce rapport, nous décrirons en détail l'architecture technique de notre site web, les fonctionnalités clés que nous avons implémentées, les difficultés que nous avons

rencontrées et les solutions que nous avons apportées pour livrer un produit fonctionnel et efficace. Nous discuterons également des perspectives d'avenir pour le projet et de l'évaluation de la performance.



I. ARCHITECTURE TECHNIQUE

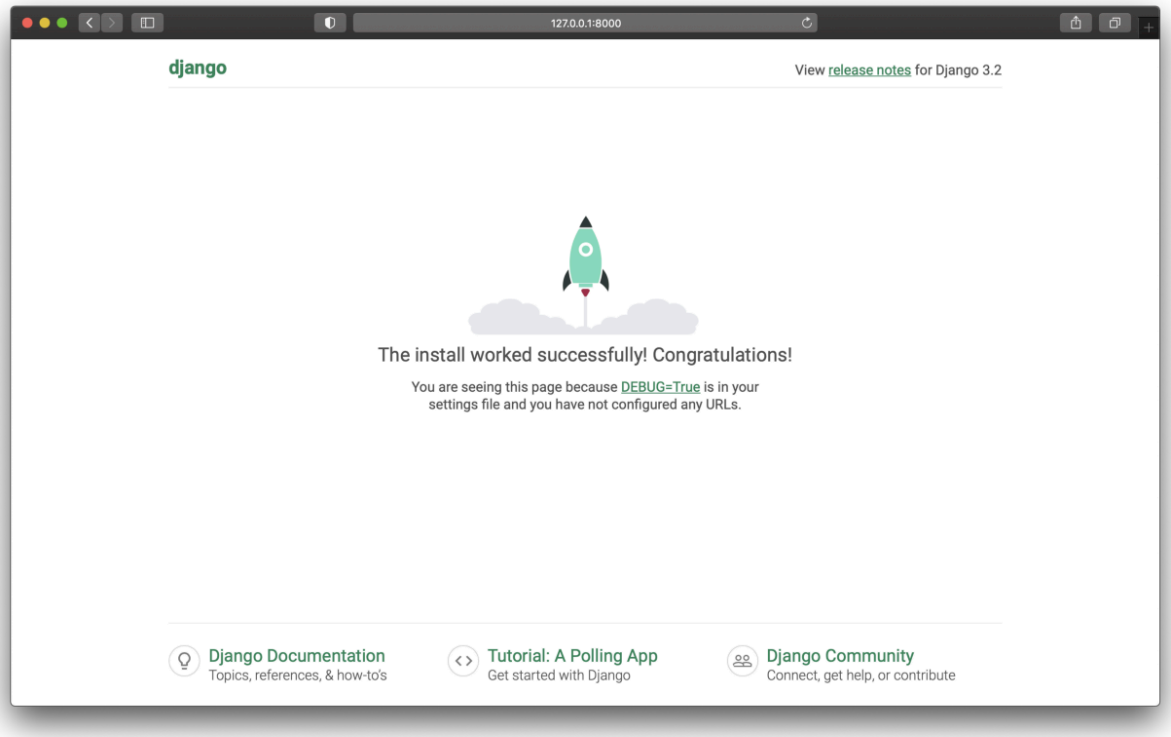
a. Description du Framework

Django est un Framework web open-source en Python conçu pour développer des applications web rapidement et facilement. Il a été créé pour gérer des sites web complexes tout en restant simple pour les développeurs.

Django inclut de nombreuses fonctionnalités prêtes à l'emploi telles que la gestion des utilisateurs, la protection contre les attaques par injection de code, la gestion de la base de données, la gestion des formulaires et la génération d'URLs.

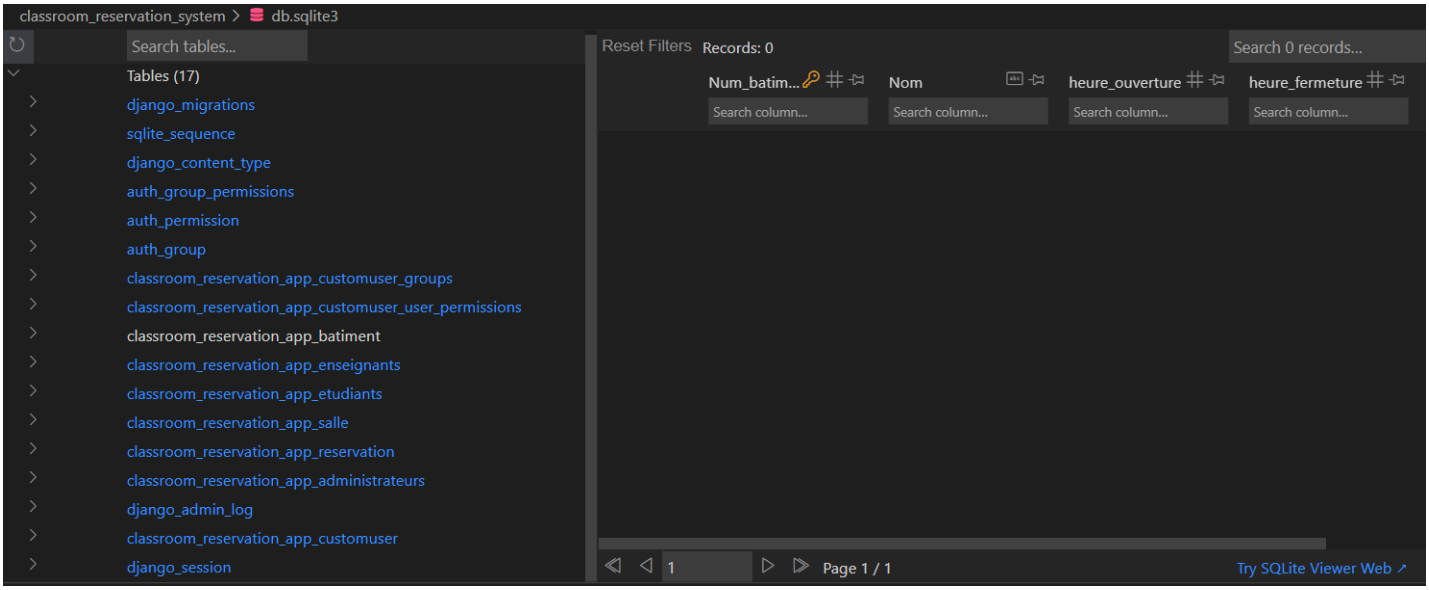
Il fournit également un système de modèles pour définir les données de votre application, ainsi qu'une interface d'administration pour gérer les données sans écrire de code. Le framework Django utilise également le pattern Model-View-Template (MVT), qui sépare les données, les fonctionnalités et les présentations de votre application.

Django est connu pour sa sécurité, sa rapidité et sa flexibilité, ce qui en fait un choix populaire pour les développeurs qui cherchent à développer des applications web à grande échelle. De plus, grâce à sa communauté active, il existe une large documentation et de nombreux paquets tierces disponibles pour étendre ses fonctionnalités.



b. Mise en place de la base de données

Nous avons créé une table spécifique pour chaque fonctionnalité du site web (bâtiment, enseignements ,étudiants, salle, réservation, administrateurs…)



II. FONCTIONNALITES CLES

a. Enregistrement de nouvelles données

Pour l'enregistrer de nouvelles données sur notre site web, nous avons procéder en plusieurs étapes en utilisant le terminal de commande et également des fonctions fournis par Django

Après avoir créé un modèle dans le fichier models.py de l'application, pour définir les champs de données que nous voulions enregistrer, nous avons effectués les migrations pour que Django puisse créer les tables correspondantes dans la base de données.

Par la suite, il a fallu créer une vue dans le fichier views.py pour gérer la logique de l'enregistrement de données. Et créer un formulaire dans le fichier forms.py pour gérer la validation des données soumises par l'utilisateur.

Après ça, nous avons créé une URL correspondant à la vue dans le fichier urls.py pour déterminer l'adresse web accessible par les utilisateurs pour enregistrer les données. Cela implique de faire correspondre une URL spécifique à notre vue, de sorte que lorsqu'un utilisateur accède à cette URL, la logique définie dans votre vue soit exécutée et les données puissent être enregistrées.

Et pour finir, nous avons créé une template HTML pour afficher le formulaire de saisie de données à l'utilisateur.

b. Suppression des informations

Nous avons suivi la même logique que pour l'enregistrent de nouvelles données.

Nous avons créé une vue dans le fichier views.py de notre application pour gérer la logique de suppression de données.

Et nous avons créé une URL correspondant à notre vue dans le fichier urls.py pour déterminer l'adresse web accessible par les utilisateurs pour supprimer les données.

Toujours comme pour la fonctionnalité précédente, nous avons créé un bouton dans la vue qui va permettre à l'utilisateur de pouvoir supprimer des informations. Cependant cette fonctionnalité ne s'adresse qu'aux administrateurs, qui gère la gestion des salles et des bâtiments.

c. Importation et exportation de données

Pour importer des fichiers CSV et exporter des fichiers CSV et JSON dans notre site web, nous avons utilisé la bibliothèque django-import-export.

Lors de l'importation, le fichier csv, qui comporte des colonnes pré-défini correspondant aux colonnes de nos tables de données, sera lu et intégré dans notre base de données.

Nous avons créé une vue pour importer et une vue pour exporter les données. Et nous avons ensuite ajouter des boutons pour que l'utilisateur puisse directement avoir accès à cette fonctionnalité.

```
246  def export(request,id_bat):
247      response = HttpResponse(content_type='text/csv')
248      writer  = csv.writer(response)
249      writer.writerow(['id salle','numero de la salle','capacite','type','id du batiment'])
250      for salle in Salle.objects.filter(id_batiment=id_bat).values_list('idsalle','num_salle','capacite','type','id_batiment'):
251          writer.writerow(salle)
252      response['Content-Disposition']='attachment; filename="salle.csv"'
253      return response
254
255  def export_json(request,id_bat):
256      data = serializers.serialize("json", Salle.objects.filter(id_batiment=id_bat))
257      out = open("Salle.json", "w")
258      out.write(data)
259      out.close()
260      return HttpResponseRedirect("/admin_home")
261
262  def importation(request):
263      csv_file = request.FILES['document']
264      decoded_file = csv_file.read().decode('utf-8')
265      io_string = io.StringIO(decoded_file)
266      for row in csv.DictReader(io_string):
267          num_salle=row['numero de la salle']
268          capacite=row['capacite']
269          type=row['type']
270          id_batiment=Batiment.objects.get(num_batiment=row['id du batiment'])
271          new_salles=Salle(num_salle=num_salle,capacite=capacite,type=type,id_batiment=id_batiment)
272          new_salles.save()
273      return HttpResponseRedirect("admin_home")
274
275
```

d. Gestion des réservation

Il est possible de réserver des salles avec du matériels à disposition. Un mail de confirmation est ensuite envoyé. Et il sera possible de modifier ou supprimer une réservation.

classroom_reservation_system > classroom_reservation_app > Student_views.py > add_reservation_save

```python
def add_reservation_save(request):
    if request.method != "POST":
        return HttpResponse("Method not allowed")

    else:
        date = request.POST.get("date")
        heure_debut = request.POST.get("heure")
        duree = request.POST.get("durée")
        nb_participants = request.POST.get("nb_pers")
        idsalle = request.POST.get("id")
        idetudiant = request.POST.get("id_user")
        batiment_id = request.POST.get("batiment_id")
        materiel = request.POST.get("Matériels")
        email = request.POST['email']

        salle = Salle.objects.get(idsalle=idsalle)
        etudiant = Etudiants.objects.get(idetudiant=idetudiant)
        try:
            heure_temp = heure_debut
            for x in range(int(duree)):
                reservation = Reservation(date=date, heure_debut=heure_debut, nb_participants=nb_participants,
                                          idsalle=salle, idetudiant=etudiant)
                reservation.save()
                heure_debut = int(heure_debut)+1;

            expediteur = 'settings.EMAIL_HOST_USER'
            send_mail('Votre réservation a bien été prise en compte',
                      'Détails de votre réservation:'+date,
                      expediteur,
                      [email],
                      fail_silently=False)
            return HttpResponseRedirect("manage_reservation_student/"+idetudiant)
        except:
            messages.error(request, "Erreur, veuillez ressaisir les informations !")
            return HttpResponseRedirect("/add_reservation_student/" + batiment_id+"/"+date+"/"+materiel+"/"+nb_participant

def manage_reservation(request, etudiant_id):
    reservation = Reservation.objects.filter(idetudiant=etudiant_id)
    return render(request, 'student_templates/manage_reservation_template.html', {"reservation": reservation})


def delete_reservation(request, reservation_id):
    reservation = Reservation.objects.get(idreservation=reservation_id)
    return render(request, 'student_templates/delete_reservation_template.html', {"reservation": reservation})
```

```python
def delete_reservation_save(request):
    if request.method != "POST":
        return HttpResponse("Method not allowed")

    else:
        reservation_id = request.POST.get("id")
        etudiant_id = request.POST.get("idetudiant")
        try:
            reservation = Reservation.objects.get(idreservation=reservation_id)
            reservation.delete()
            messages.success(request, "La réservation a été supprimée !")
            return HttpResponseRedirect("manage_reservation_student/"+etudiant_id)

        except:
            messages.error(request, "Erreur, la réservation n'est pas supprimée !")
            return HttpResponseRedirect("/delete_reservation_student/"+reservation_id)
```

III. DEFIS RENCONTRES ET SOLUTIONS APPORTEES

Tout d'abord l'un des principaux défis rencontrés était d'apprendre le Framework Django. En effet, c'était un tout nouveau langage pour toute l'équipe. De plus Django est vraiment perçu comme une Framework complexe, on a alors mis du temps à comprendre son fonctionnement. Il inclut de nombreuses fonctionnalités qui nous ont pris du temps à toutes les découvrir. Ce Framework nous a un peu mis des bâtons dans les roues en raison de sa complexité, de la quantité d'informations disponible mais aussi la nouveauté pour nous. Django utilise une approche modèle-vue-Template qui est un design pattern. Cette approche consiste à séparer les différentes parties d'une application en trois composantes distinctes. On a eu du mal à se répartir les taches vues que cette méthode de travail était nouvelle pour l'équipe.

a. Problèmes de sécurité des données

Django utilise un modèle ORM (Object Relational Mapping) pour mettre en relation les données de la base de données avec le code Python. Cependant, la complexité de ce modèle peut rendre difficile la gestion des relations entre les différentes tables de la base de données. De plus le lien entre la base de données et le code peut affecter les performances de l'application, en particulier si le modèle ORM n'est pas configuré correctement. Si la mise en relation entre la base de données et le code est incorrecte, il peut être difficile de déterminer la source des erreurs et de les corriger. Au fil du temps, la structure de la base de données peut évoluer et des modifications peuvent être nécessaires. La migration de la base de données peut être compliquée si le modèle ORM n'est pas configuré correctement.

Pour résoudre le problème de la complexité du modèle ORM, nous avons dû nous documenter sur le modèle ORM, nous avons aussi tester régulièrement les performances pour vérifier que cette mise en relation n'affecte pas les performances du site.
Il est important de disposer de mécanismes de débogage efficaces pour diagnostiquer rapidement les erreurs lors de la mise en relation entre la base de données et le code.

b. Difficultés rencontrées lors de l'implémentation de fonctionnalités spécifiques

Étant de jeune développeur naïf, la différence entre Template et Templates n'existait pas, pour nous. Ce projet nous as appris que si.
En informatique, la différence entre les termes "Template" et "Templates" dépend du contexte dans lequel ils sont utilisés. En général, le terme "Template" désigne un modèle générique utilisé pour créer d'autres objets en fonction de celui-ci. Par exemple, dans le développement de logiciels, un Template peut être utilisé pour créer des pages web ou des fichiers avec une structure et un format similaire. D'un autre côté, le terme "Templates" est utilisé pour désigner plusieurs modèles différents

Concernant la différence entre « Template » et « Templates », nous avons dû faire beaucoup de recherche pour comprendre l'origine du problème et pour pouvoir le résoudre.
Dans le contexte de Django, le Framework web open-source en Python, le terme "templates" est généralement utilisé avec un "s" pour désigner les fichiers de modèles de pages web. Les templates Django sont utilisés pour définir la structure et le contenu des pages web en utilisant un langage de modèle spécial appelé syntaxe de modèle Django. Cela sert à définir des variables et des boucles pour générer automatiquement le contenu de la page web à partir de données enregistrées dans la base de données. En utilisant les Templates Django, sa sépare la logique de l'application et de la présentation des pages web, ce qui permet une meilleure organisation du code et une plus grande flexibilité dans la mise en forme des pages web. Tandis qu'utilisez "template" sans "s" pour désigner les modèles de pages web dans Django cause des problèmes. Dans le contexte de Django, "template" sans "s" n'a pas de signification standard et pourrait être mal compris. Par conséquent, pour éviter toute confusion, utilisez "templates" avec un "s" pour désigner les modèles de pages web dans ce Framework.

c. Problème de base de données

Nous avons rencontré un problème de connexion entre notre site web Django et notre base de données en raison d'une erreur de liaison entre le nom d'hôte et le nom de la base de données.

Lorsque nous avons essayé de nous connecter à la base de données, Django a utilisé un nom d'hôte pour localiser la base de données et un nom de base de données pour identifier la base de données spécifique à laquelle nous voulions nous connecter. Malheureusement, ces informations étaient incorrectes ou ne correspondaient pas, ce qui a entraîné des erreurs de connexion.

Pour résoudre ce problème, nous avons dû vérifier que les informations de connexion que nous avons fournies à Django étaient correctes et correspondaient aux informations de notre base de données. Cela a impliqué de vérifier les informations de connexion telles que le nom d'hôte, le nom de la base de données, l'utilisateur et le mot de passe, et de les ajuster si nécessaire.

d. Problème de réservation

Nous avons rencontré un problème de connexion entre notre site web Django et notre base de données en raison d'une erreur de liaison entre le nom d'hôte et le nom de la base de données.

Lorsque nous avons essayé de nous connecter à la base de données, Django a utilisé un nom d'hôte pour localiser la base de données et un nom de base de données pour identifier la base de données spécifique à laquelle nous voulions nous connecter. Malheureusement, ces informations étaient incorrectes ou ne correspondaient pas, ce qui a entraîné des erreurs de connexion.

Pour résoudre ce problème, nous avons dû vérifier que les informations de connexion que nous avons fournies à Django étaient correctes et correspondaient aux informations de notre base de données. Cela a impliqué de vérifier les informations de connexion telles que le nom d'hôte, le nom de la base de données, l'utilisateur et le mot de passe, et de les ajuster si nécessaire.

IV. CONCLUSION

Nous arrivons à la fin de ce projet qui nous tient tant à cœur, nous avons décrit la création d'un site web permettant de gérer les informations des étudiants, des professeurs, des salles et des bâtiments de l'Université Paris Nanterre. Pour ce faire, nous avons utilisé le Framework Django, qui a permis de développer rapidement et efficacement ce site web en utilisant les Templates Django. Le site web offre une interface conviviale pour les utilisateurs, qui peuvent accéder et gérer facilement les informations enregistrées. La séparation de la logique de l'application de la présentation des pages web grâce aux templates Django a permis une meilleure organisation du code et une plus grande flexibilité dans la mise en forme des pages web.
En conclusion, nous avons réussi à créer un site web fonctionnel et efficace pour gérer les informations de l'Université Paris Nanterre en utilisant Django. Sa nous as pris du temps, de l'énergie, on a rencontré des problèmes mais nous y sommes arrivées et nous somme fière aujourd'hui de vous le présenter. Ce site web peut être facilement adapté pour répondre aux besoins futurs de l'université. Nous espérons que ce projet sera utile et apportera une valeur ajoutée à l'université.