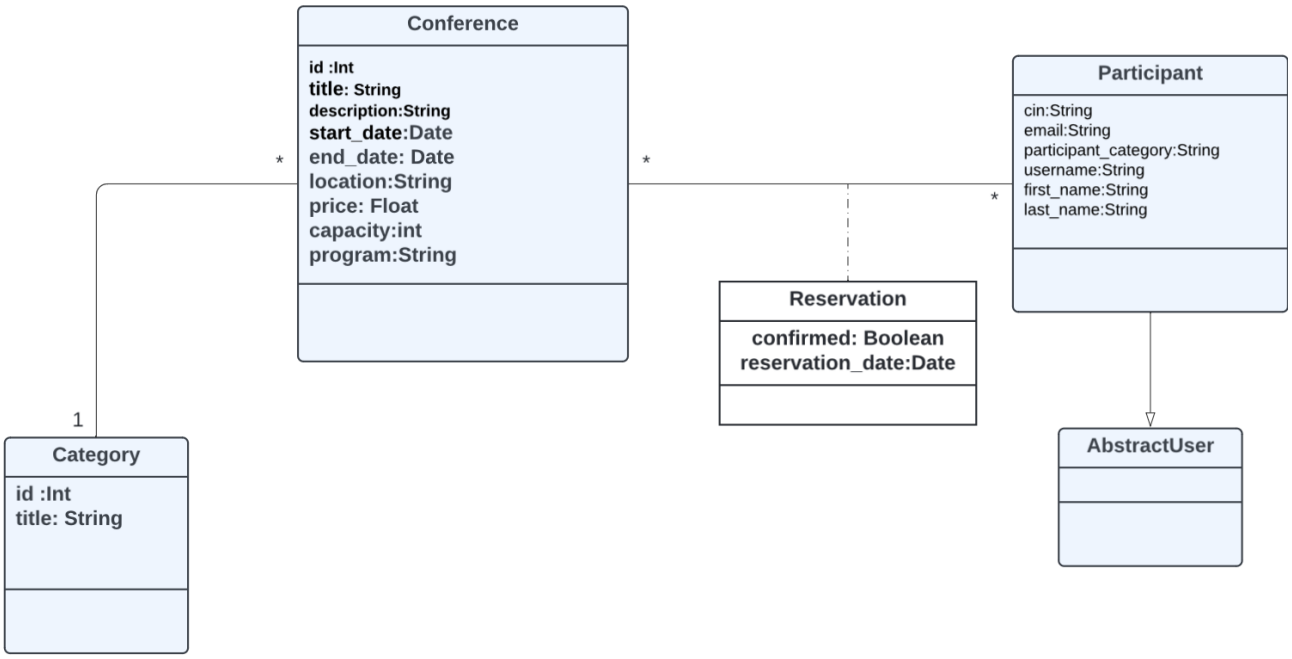
esprit Se former autrement HONORIS UNITED UNIVERSITIES	**Python Framework for the web** **2024-2025**

Etude de cas

I. Description : Système de gestion de conférences

Vous devez concevoir et développer une application web pour la gestion de conférences. Le système permet de créer et de gérer des conférences tout en intégrant les réservations des participants. Chaque conférence est associée à une catégorie ou à un type spécifique (par exemple, conférence académique, séminaire d'entreprise, conférence technique, etc.). Les participants peuvent s'inscrire à différentes conférences et confirmer leur participation après avoir effectué une réservation.

II. Objectifs du système :



1. **Gestion des conférences :**
   - Créer, modifier, afficher et supprimer des conférences.
   - Chaque conférence doit être liée à une **catégorie** (ou **type de conférence**) via une relation **Many-to-One**. Cela signifie qu'une catégorie peut regrouper plusieurs conférences, mais chaque conférence ne peut appartenir qu'à une seule catégorie.
2. **Gestion des participants :**
   - Créer et modifier des comptes.
   - S'authentifier.

Etude de cas

- Chaque participant peut réserver plusieurs conférences et confirmer sa participation après réservation.
- Cette relation entre les **participants** et les **conférences** est une relation **Many-to-Many**, car un participant peut assister à plusieurs conférences, et une conférence peut accueillir plusieurs participants.
- Lorsqu'un participant réserve une conférence, il doit pouvoir **confirmer** sa présence.

III. Mise en place du projet

Pour la mise en place du projet,on aura besoin:

- Installer Python 3.10 et plus.
- Créer un environnement virtuel.
- Installer Django 4.2
- Créer un projet Django.
- Créer trois applications : **Categorie**, **Conference** et **Participant.**

IV. Génération de la BD

1. Développer les entités dans le fichier « models.py » en se basant sur le diagramme de classe et en appliquant les contraintes suivantes :

=> Le modèle « **Participant** » :

- hérite de l'entité « **AbstractUser** » prédéfinie. Il faut changer la clé primaire « id » à « cin » qui doit avoir une longueur fixe égale à 8.
- l'attribut **participant_catgory** est une liste de choix qui contient ces valeurs **« étudiant, enseignant, doctorant, chercheur »**
- l'attribut **email** doit être de type **« EmailField »** et unique.
- l'attribut **username** doit être unique et configuré comme un USERNAME_FIELD.
- Ajouter dans à la fin du fichier **settings.py** la ligne :
  AUTH_USER_MODEL='Users.Participant'

-=> Le modèle « **Conference** » :

- l'attribut **description** doit être de type « TextField ».
- l'attribut **program** doit être de type « FileField », il faut installer la libraire **Pillow** via cette commande : pip install Pillow.
- l'attribut **capacity** doit être de type « IntegerField ».
- posséde les attribut **created_at** et **updated_at**.

=>Le modèle « **Reservation** » :

- Un participant peut réserver plusieurs conférences. Une conférence peut avoir plusieurs participants.
- L'attribut « **reservation_date** » doit avoir comme valeur par défaut la **date système**.
- Un participant ne peut pas réserver plusieurs fois la même conférence.

Etude de cas

=> Le modèle « **Category** »
- posséde les attribut **created_at** et **updated_at**.
- titre du category doit être unique.

2. Modifier le fichier **settings.py** pour que le nom de la BD porte le nom de la classe dont vous faites partie.

3. Générer les fichiers de migration via la commande :
- python manage.py makemigrations NomApp

4. Générer le schéma de la BD via la commande
- python manage.py migrate

V. Application Admin

1. Créer un nouvel utilisateur de type admin pour accéder au Dashboard admin via cette commande :

Python manage.py createsuperuser

2. Dans le fichier « **admin.py** » de l'application « **Users** », inscriver le modèle « **Participant** » au site d'administration comme suit :

admin.site.register(Participant)

3. De même pour les modèles **Category** , **Conference** et **Reservation**.
4. Consulter l'interface de l'administrateur en utilisant l'url : **http://localhost:8000/admin , Que remarquez-vous ?**
5. Faire les modifications nécessaires pour modifier les noms des modèles affichés dans l'interface administrateur. (Utiliser **verbose_name_plural**).

VI. Validateurs

1. Ajouter les validateurs suivant dans chaque modèle associé :
   - **category title :** le titre de la categorie doit être avoir seulement des caractères. (En utilisant une fonction ou **RegexValidator** directement)
   - **capacity :** la capacité de la conférence ne dépasse pas un certain nombre maximal. (En utilisant **MaxValueValidator**)
   - **program :** limiter les types de fichiers autorisés (En utilisant une fonction ou **FileExtensionValidator**).
   - **cin** : vérifier que le CIN ne contient que des caractères numériques et que la longueur de la chaîne est exactement égale à 8. (En utilisant **RegexValidator**)
   - **email :** forcer les adresses électroniques à appartenir à un certain domaine (par exemple, domaine universitaire : esprit.tn) en utilisant une fonction.
   - **start_date :** la date début de la conférence doit être supérieur à la date système. (En utilisant une fonction ou **la classe Q)**

Etude de cas

- **end_date :** la date de fin de la conférence doit être supérieur à la date de début. (En utilisant la fonction **clean**)
- **reservation_date :** la réservation ne peut être faite que pour des conférences à venir (En utilisant la fonction **clean**).
- Limiter le nombre de conférences qu'un participant peut réserver par jour (ex : 3 conférences max par jour). (En utilisant la fonction **clean**).

VII. Personnalisation du Dashboard Admin

1. Personnalisation du modèle Conférence

a. Modifiez l'affichage de la liste des conférences via la fonction **__str__( )** dans le fichier models.py.
b. Définissez et personnaliser la variable **list_display** afin de contrôler quels champs seront affichés sur la page de la liste de l'interface d'administration.
c. Ajoutez un champ de recherche via **search_fields** sur un attribut comme **title**.
d. Activez la pagination sur le tableau de la liste des conférences.
e. Définissez l'ordre des conférences dans la liste par date de début.
f. Organisez les champs du formulaire par sections via **fieldsets** (ex : Description, Horaires de la conférence, Catégorie de participant ....).
g. Afficher les champs **created_at** et **updated_at** mais les rendre **non modifiable**.
h. Afficher les entrées des réservations dans le formulaire d'ajout d'une conférence sous forme de tableau ou sous format empilé.
i. Modifier le champs **category** du formulaire d'ajout d'une conférence en lui affectant une fonction d'autocomplétion, ce qui facilitera la sélection des catégories lors de la création ou de la modification d'une conférence.
j. Activez les filtres dans la barre latérale droite de la page d'affichage de la liste des conférences avec **list_filter** :

- Un filtre selon le titre de la conférence.
- Un filtre selon le nombre de participants avec deux catégories : « No participants » et « There are participants ».
- Un filtre selon la date de la conférence selon trois critères : « Past Conferences », «Upcoming Conferences» et « Today Conferences»

2. Personnalisation du modèle Réservation :

a. Définissez et personnaliser la variable **list_display** afin de contrôler quels champs seront affichés sur la page de la liste de l'interface d'administration.
b. Ajouter deux actions dans la liste des reservations qui permettent de rendre l'état d'une reservation « confirmé » ou « non confirmé »

3. Personnalisation du modèle Participant :

Etude de cas

a. Faire les modifications nécessaires dans le fichier **admin.py** pour le modéle participant.