esprit Se former autrement HONORIS UNITED UNIVERSITIES	Workshop Python Framework for the web 2025-2026 **Partie Admin&Validators**

I. Description : Gestion d'une conférence scientifique

Dans le cadre de cet atelier, nous allons développer une application web de **gestion de conférences scientifiques**. L'objectif est de simuler le processus complet d'organisation et de participation à une conférence académique en intégrant plusieurs fonctionnalités essentielles.

L'étude de cas repose sur le diagramme de classe suivant :

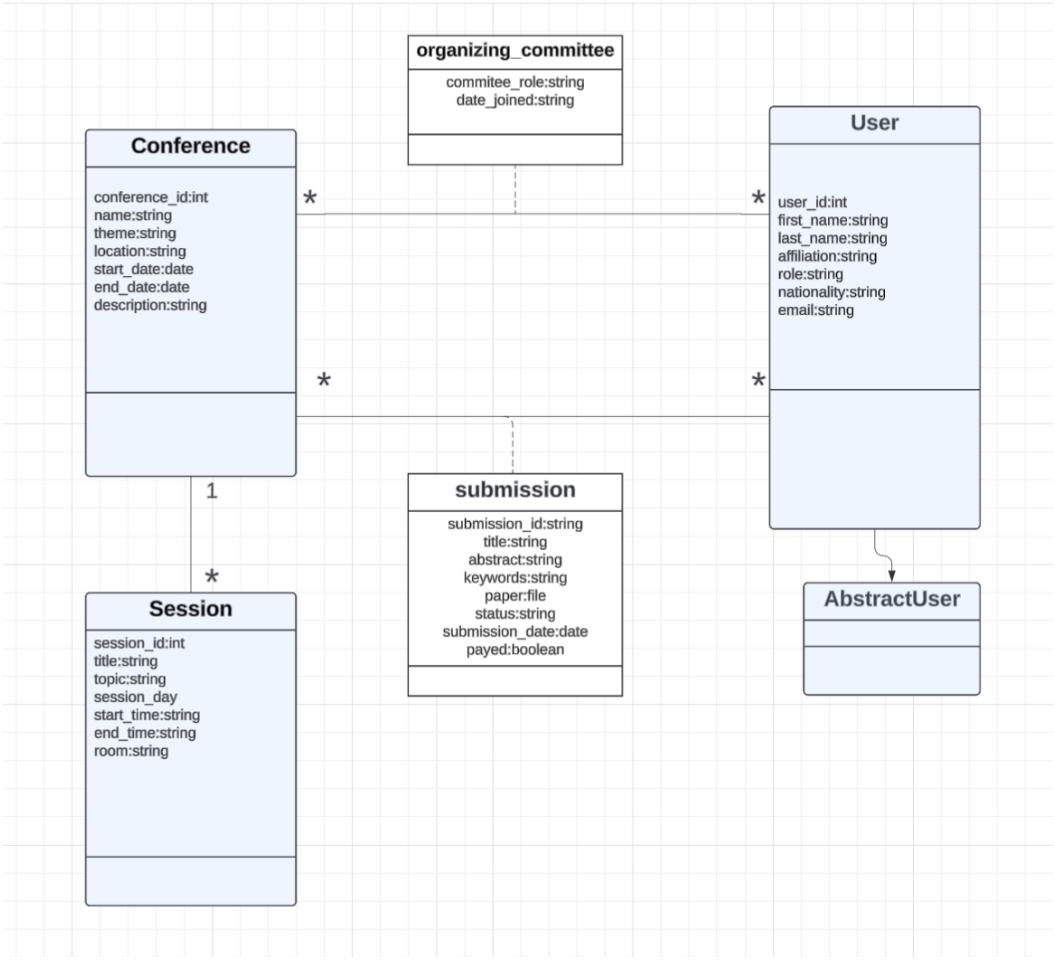

Figure1 : diagramme de classe

1. **Gestion des conférences**
   - Chaque conférence possède un titre, un thème, un lieu, une localisation, des dates de début et de fin, ainsi qu'une description générale.
   - Une conférence peut regrouper plusieurs sessions de présentations d'articles.

2. **Gestion des sessions**
   - Une session est associée à une seule conférence et comprend un titre, un sujet, une date, un horaire et une salle.
3. **Comité d'organisation**
   - Chaque conférence est gérée par un comité d'organisation composé d'un seul président, d'un seul co-président et de plusieurs membres.
   - Les rôles dans le comité sont définis et chaque membre est lié à une conférence.
4. **Gestion des utilisateurs**
   - Les utilisateurs peuvent avoir différents rôles : participant, organisateur ou membre du comité scientifique.
   - L'application gère l'inscription des utilisateurs avec leurs informations personnelles (nom, prénom, affiliation, email, etc.).
5. **Soumission et suivi des articles**
   - Les auteurs peuvent soumettre un article (titre, résumé, mots-clés, fichier PDF...).
   - Chaque soumission est liée à une conférence et peut passer par différents statuts : *soumis*, *en cours de révision*, *accepté*, *rejeté*.
6. **Inscriptions à la conférence**
   - L'inscription d'un utilisateur n'est **validée** que si deux conditions sont remplies :
     - **Son article est accepté**.
     - **Il paie les frais d'inscription**.
   - Sans article accepté, l'utilisateur ne peut pas être considéré comme participant officiel.

II. Mise en place du projet

Pour démarrer le projet, nous aurons besoin de :

- Installer **Python 3.11 ou une version ultérieure**.
- Créer un **environnement virtuel** dédié.
- Installer **Django 5.2.**
- Créer un **nouveau projet Django**.
- Créer trois applications : **SessionApp, ConferenceApp et UserApp.**

III. 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 :

   Tous les modèles possèdent des attributs created_at et updated_at.

=> Le modèle « User » :

- Hérite de l'entité « **AbstractUser** » prédéfinie. Il faut changer la clé primaire « **id** » à « **user_id**» qui doit avoir une longueur fixe égale à 8.
- L'attribut **email** doit être de type « **EmailField** » et unique.
- Si **l'administrateur** ajoute un utilisateur, le champ « **role** » doit être choisi parmi une liste prédéfinie : participant, organisateur ou membre du comité scientifique.
- Si **un utilisateur** s'inscrit pour participer à une conférence, son rôle est automatiquement défini comme participant.

=> Le modèle « Conference» :

- *L'attribut* « ***description*** » *doit être de type « **TextField** ».*
- L'attribut « ***Theme*** » est une liste de choix comme suit : ***Computer Science & Artificial Intelligence & Science & Engineering & Social Sciences & Education & Interdisciplinary Themes.***

=> Le modèle « Session » :

- L'attribut « ***title*** » est de type « **CharField** » et représente le titre de la session.
- L'attribut « ***topic*** » est de type « **CharField** » et décrit le sujet principal de la session.
- L'attribut « ***session_day*** » est de type « **DateField** ».
- Les attributs « ***start_time*** » et « ***end_time*** » sont de type « **TimeField** ».
- L'attribut « ***room*** » est de type « **CharField** » et indique la salle de la session.
- Chaque session est liée à **une seule conférence** (relation *Many-to-One* avec Conference).

=> Le modèle « Submission »:

- L'attribut « ***title*** » est de type « **CharField** ».
- L'attribut « ***abstract*** » est de type « **TextField** ».
- L'attribut « ***keywords*** » est de type « **CharField** » (ou « **TextField** » si plusieurs mots-clés séparés par des virgules).
- L'attribut « ***paper*** » est de type « **FileField** » (upload de l'article).
- L'attribut « ***status*** » est un champ à choix multiples parmi : **Submitted, Under Review, Accepted, Rejected.**
- L'attribut « ***submission_date*** » est de type « **DateField** » (auto_now_add=True).
- L'attribut « ***payed*** » est un « **BooleanField** » qui indique si les frais d'inscription ont été réglés.
- L'inscription d'un utilisateur n'est validée que si son article est *accepté* **et** si **payed=True.**

=> Le modèle *«* OrganizingCommittee »:

- L'attribut « ***committee_role*** » est de type « **CharField** » et doit être choisi parmi une liste prédéfinie (exemple : *chair*, *co-chair*, *membre*).
- L'attribut « ***date_joined*** » est de type « **DateField** » (date d'intégration au comité).

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
```

II. 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 « **UserApp** », inscriver le modèle « User » au site d'administration comme suit :

   admin.site.register(user)

3. De même pour les autres modèles.
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**).

III. Validateurs

1. Ajouter les validateurs suivants dans chaque modèle associé :
   - **user_id** : vérifier que l'identifiant a une longueur est exactement égale à 8 caractères et respecte un format comme "USERXXXX", il doit être généré automatiquement lors de l'ajout d'un utilisateur.
   - **email (User)** : s'assurer que l'adresse email appartient à un domaine universitaire (par ex. @esprit.tn ou autre). (Utiliser une fonction de validation personnalisée).
   - **firstname / lastname (User)** : les noms et prénoms ne doivent contenir que des lettres (et éventuellement des espaces ou tirets). (En utilisant RegexValidator).
   - **Conference title** : le titre de la conférence doit contenir uniquement des lettres et espaces (pas de chiffres). (Utiliser RegexValidator).
   - **conference description** : vérifier que la description a une longueur minimale (par exemple 30 caractères) pour éviter des descriptions trop courtes. (En utilisant MinLengthValidator).
   - **Conference day : la date début** de la conférence doit être **supérieur à la date fin**.

- **session_day (Session)** : la date de la session doit appartenir à l'intervalle de dates de la conférence associée (entre start_date et end_date). (En utilisant une fonction de validation personnalisée).
- **start_time / end_time (Session)** : vérifier que l'heure de fin est toujours **supérieure à l'heure de début**. (Utiliser une fonction de validation personnalisée).
- **room (Session)** : s'assurer que le nom de la salle ne contient que des lettres et chiffres (pas de caractères spéciaux). (Utiliser RegexValidator).
- **paper (Submission)** : limiter les fichiers aux extensions **.pdf** uniquement. (Utiliser FileExtensionValidator).
- **keywords (Submission)** : vérifier que la chaîne ne dépasse pas un nombre maximal de mots-clés (par ex. 10). (Utiliser une fonction personnalisée qui compte les mots séparés par virgules).
- **submission_date :** la Soumission ne peut être faite que pour des conférences à venir (En utilisant la fonction clean).
  - Limiter le nombre de conférences qu'un participant peut déposer une soumission par jour (ex : 3 conférences max par jour). (En utilisant la fonction clean)
- **Submission_id:** est automatiquement généré avec un format comme "SUB-ABCDEFGH".

IV. Personnalisation du Dashboard Admin

1. Modifier les en-têtes de l'interface d'administration

Dans le fichier admin.py, personnalisez les propriétés suivantes afin de remplacer les valeurs par défaut de Django :

- site_header : définit le titre affiché en haut de l'interface d'administration.
- site_title : définit le titre de la page dans l'onglet du navigateur.
- index_title : définit le titre affiché sur la page d'accueil du tableau de bord.

2. Personnalisation du modèle Conférence

Affichage dans la liste d'administration :

a. Montrer les colonnes : **name, theme, location, start_date, end_date.**
b. Ajouter une méthode personnalisée duration qui calcule la durée (en jours) entre **start_date** et **end_date.**
c. Afficher la durée dans la liste.

Filtres et recherche :

d. Ajouter des filtres sur theme, location et start_date.
e. Activer la recherche sur **name**, **description** et **location**.

Formulaire d'édition :

f. Organiser les champs en **sections (fieldsets)** :
   i. Informations générales : **name, theme, description.**
   ii. Logistique : **location, start_date, end_date.**

Améliorations visuelles :

g. Définir un **ordering par start_date**.
h. Définir **date_hierarchy** = "**start_date"** pour une navigation par calendrier.

Mise en place d'un Inline :

i. Créez une classe SubmissionStackedInline en héritant de admin.StackedInline.
ii. Configurez-la pour afficher les champs essentiels d'une soumission (title, abstract, status, payed, etc.).
iii. Limitez certains champs en lecture seule (submission_id, submission_date).
iv. Ajoutez l'inline dans l'admin de Conference pour que les soumissions apparaissent en **mode vertical (stacked)**.

Variante Tabulaire :

i. Créez également une classe SubmissionTabularInline en héritant de admin.TabularInline.
ii. Cette fois, n'affichez que les champs principaux dans un tableau (title, status, user, payed).
iii. Testez cette version en l'ajoutant à l'admin de Conference.

3. Personnalisation du modèle Submission

Affichage dans la liste d'administration :

a. Colonnes : **title, status, user, conference, submission_date, payed.**
b. Ajouter une méthode personnalisée **short_abstract** qui tronque l'abstract à 50 caractères pour l'affichage rapide.
c. Afficher la méthode dans la liste.

Filtres et recherche :

d. Filtres : **status, payed, conference, submission_date.**
e. Recherche : **title**, **keywords**, user__username.

Edition directe depuis la liste :

f. Rendre **status** et **payed** modifiables directement depuis la liste (list_editable).

Formulaire d'édition :

g. Organiser les champs par **sections** :
   i. **Infos générales : submission_id, title**, **abstract, keywords.**
   ii. **Fichier et conférence : paper, conference.**
   Iii. **Suivi: status, payed, submission_date, user.**

h. Rendre submission_id et submission_date **lecture seule**.

Actions personnalisées :

j. Ajouter une action pour **marquer plusieurs soumissions comme payées**.
k. Ajouter une action pour **accepter plusieurs soumissions en un clic**.