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

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 :

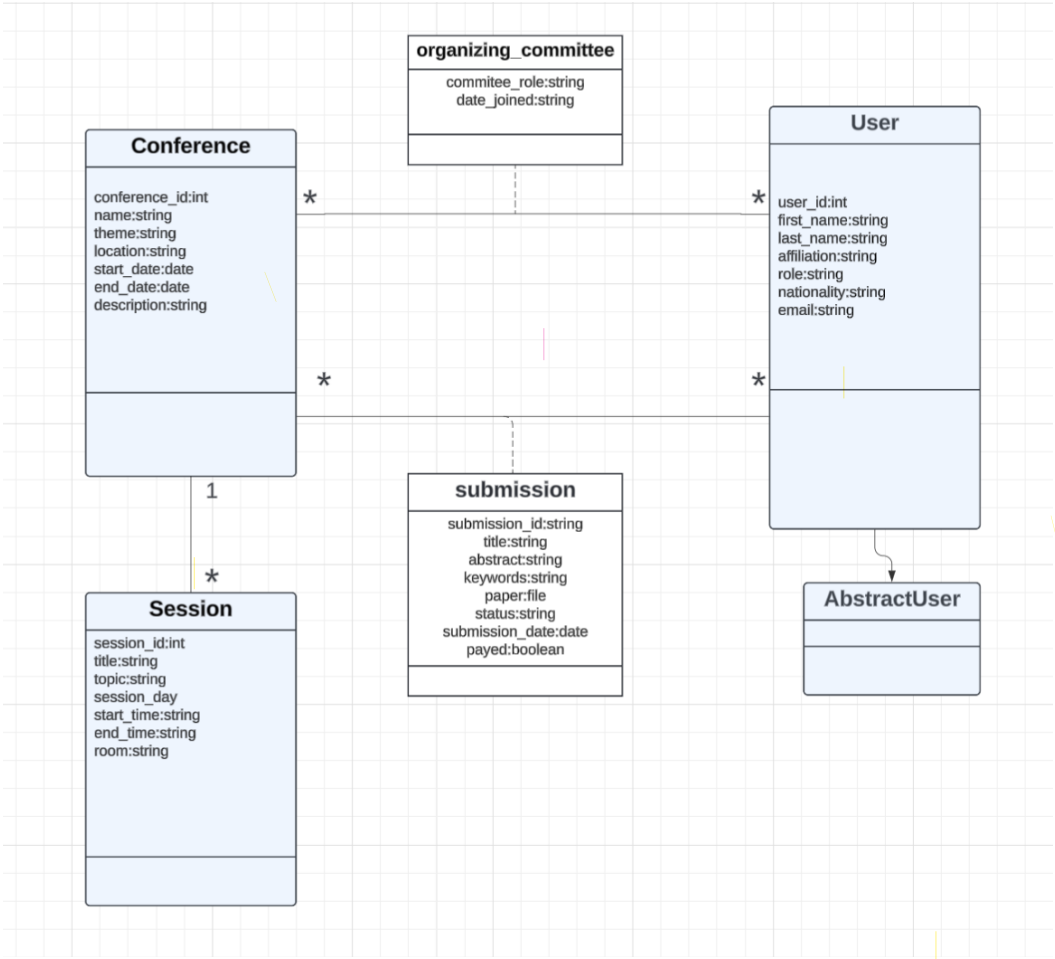

Figure1 : diagramme de classe

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*, *member*).
- 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
```