|  |  |
| --- | --- |
| Atelier Django  AU 2022-2023 |  |

**Etude de cas**

L’objectif de cet atelier est de développer une application web de gestion des évènements en utilisant le Framework Django pour les deux parties Front-Office et Back-Office. Cette application est représentée par le diagramme de classe d’analyse suivant :

|  |
| --- |
| **Event**  1  Organizer  \* |
| Id  title  description  image  \*  \*  category  state  nbe\_participant  evt\_date  creation\_date  update\_date |

|  |
| --- |
| **Person** |
| cin  email |

|  |
| --- |
| **Participation** |
| participation\_date |

|  |
| --- |
| **User** |
|  |

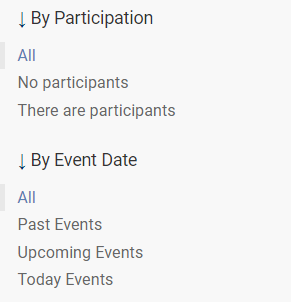
1. **Mise en place du projet**

* Django 4.1
* Python 3.11
* Créer un environnement virtuel nommé « **venv** »
* Créer un projet nommé «  **GestionEvenement** »
* Créer deux applications : **Event** et **Person**

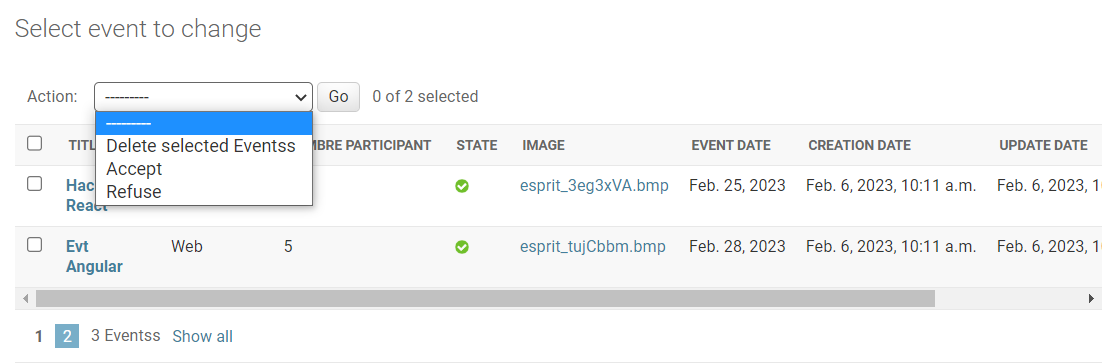
1. **Génération de la BD**
2. Développer les entités dans le fichier « **models.py** » en ce basant sur le diagramme de classe ci-dessus et en appliquant les contraintes suivantes :

* Le modèle « **Person** » hérite de l’entité « **User »** prédéfinit. Il faut changer la clé primaire « id » à « cin » qui doit avoir une longueur fixe égale à 8
* Le modèle « **Event** »doit respecter les contraintes suivantes :
* **Description** doit être de type « TextField ».
* **Image** doit être de type « ImageField », il faut installer la libraire **Pillow** via cette commande : «  pip install Pillow ».
* **State** (bool, default=False)
* **Email** doit se terminer par “@esprit.tn “.
* **Category** doit représenter uneliste de choix comme suit:
* Musique
* Cinema
* Sport
* Un événement est organisé par une seule personne et une personne peut organiser plusieurs événements.
* Une personne peut participer à plusieurs évènements. Un évènement peut avoir plusieurs participants.
* Déclarer la classe **Meta** dans la quelle on définit la contrainte sur l’attribut « **evt\_atetés dans le fichier models.pyuivantesi dessus e de on web date**> à date now ».
* Dans le modèle « **Participation** » est comme suit :
* L’attribut « participation\_date » doit avoir comme valeur par défaut la date système.
* Une personne ne peut pas participer plusieurs fois au même évènement.

1. Modifier le fichier **settings.py** pour que le nom de la BD porte le nom de la classe dont vous faites partie.
2. Générer les fichiers de migration via la commande « python manage.py makemigrations NomApp ».
3. Générer le schéma de la BD via la commande « python manage.py migrate ».
4. **Application Back-Office**
5. Créer un nouvel utilisateur de type admin pour accéder au Dashboard admin via cette commande : **python manage.py createsuperuser**.
6. Dans le fichier « **admin.py** » de l’application « **Person** » apportez les modifications suivantes :
   1. Inscrivez le modèle « Person » au site d’administration comme suit : **admin.site.register(Person).**
   2. Définissez la classe **ResearchPerson** qui permet de faire la recherche d’une personne selon son username : search\_fields=['username'].
7. Dans le fichier « **admin.py** » de l’application « **Event** » apportez les modifications suivantes :
   1. Inscrivez le modèle « Event » au site d’administration ;
   2. Créer la classe **EventAdmin**.
   3. Affichez la liste des évènements comme suit:
      1. Modifiez l’affichage de la liste des évènements via la fonction \_\_str\_\_( ) dans le fichier models.py. Le site d’administration affichera une seule colonne avec la représentation \_\_str\_\_() de l’objet Event.
      2. Définissez et personnaliser la variable **list\_display** à fin de contrôler quels champs seront affichés sur la page de la liste de l’interface d’administration.
      3. Ajouter une fonction qui permet d’afficher le nombre total des participants dans chaque évènement.
      4. Ajoutez la caractéristique **readonly\_fields** pour les deux attributs creation\_date et update\_date pour qu’on puisse les inclure dans la variable list\_diplay.
      5. Activez la pagination sur le tableau de la liste des évènements via **list\_per\_page**.
   4. Activez les filtres dans la barre latérale droite de la page d’affichage de la liste des évènements avec **list\_filter**:
      1. Un filtre selon le titre de l’évènement.
      2. Un filtre selon le nombre de participants avec deux catégories : « No participants » et «  There are participants ».
      3. Un filtre selon la date de l’évènement selon trois critères : « Past Events », «Upcoming Events » et « Today Events ».



* 1. Ajouter deux actions dans la liste des évènements qui permettent de rendre l’état d’un évènement « **accepté »** ou « **refusé »** comme le montre la figure ci-dessous :



* 1. Modifiez le formulaire de gestion d’Event :
     1. Définissez **fieldsets** qui est une liste de tuples binaires pour contrôler la mise en page des pages d’administration « ajouter » et « modifier ». chaque tuple représente une balise <fieldset> dans la page du formulaire d’administration.

fieldsets = (

        ('A propos', {"fields": ('title','description','image'),}),

        ('Date',{"fields":('event\_date','creation\_date','update\_date')

        }),

    ('Others',{"fields":('category','state','nbe\_participant') }),

    ('Personal',{"fields":('organizer',) }))

* + 1. Définissez **inlines** pour créer une instance de la classe « participation » qui permet d’ajouter les participants pour chaque évènement. Il faut donc créer la classe **ParticipationAdmin** qui sera de type InlineModelAdmin et tester les deux sous classes :
* [TabularInline](https://docs.djangoproject.com/fr/4.1/ref/contrib/admin/#django.contrib.admin.TabularInline)
* [StackedInline](https://docs.djangoproject.com/fr/4.1/ref/contrib/admin/#django.contrib.admin.StackedInline)

**NB :** inlines prendra comme valeur le nom de la classe **ParticipationAdmin.**

* + 1. Appliquez l’auto-complete lorsque vous cherchez une personne lors de la sélection de l’organisateur d’un évènement dans le champ « organizer » :

autocomplete\_fields=['organizer']

1. **Application Front Office**

* **CRUD évènement**

1. Créer une fonction qui permet d’afficher «  Bonjour 4TWIN » en utilisant la fonction HttpResponse().
2. Créer une fonction qui permet d’afficher «  Bonjour *classe* » en utilisant la fonction HttpResponse().*« classe »*  est une variable dynamique qui permet d’afficher dynamiquement la classe dont vous faites partie. Cette variable est récupèe via l’url.
3. Créer la fonction « ListEvt » qui permet d’afficher la liste des évènements avec 2 méthodes :
   1. En utilisant la queryset : Event.objects.all()
   2. En utilisant la classe générique : ListView
   3. La liste des évènements est triès selon la date de l’evènement.
4. Créer la fonction « DetailEvt » qui permet d’afficher les détails d’un évènement lors de la clique sur le bouton de l’évènement en question.
5. Développer la fonction qui permet d’ajouter un evènement « AddEvt ».
6. Dans le tableau d’évènement, ajouter un lien qui permet de modifier un évènement « UpdateEvt ».
7. Dans le tableau d’évènement, ajouter un lien qui permet de supprimer un évènement « UpdateEvt ».
8. Afficher la liste des évènements validés dans la fonction « ValidEvt ».
9. Dans la liste des évènements, ajouter le bouton « **participer** ». En cliquant dessous :
   * Le nombre de participant s’incrémente ;
   * Le bouton participer devient **disabled**.
   * Un bouton annuler participation sera affiché.

* **Intégration Template**

1. Créer un Template « base.html » qui contiendra le gabarit de base de la partie Front-Office.
   1. Utiliser "{% static 'ficher' %}" pour importer les fichiers bootsrup ainsi que les images à inclure dans la page web.
   2. Définir les block : Header, content et footer
2. Dans le fichier « header.html » ajouté la barre de menu avec des liens vers la page d’affiche, d’ajout, de register et de login.
3. Dans le fichier « footer.html » ajouté un copyright avec l’année en cours ainsi que votre classe.

* **Login/Logout Person**

1. Créer l’interface qui permet à une personne de créer un compte pour accéder à nos pages web.
2. Ajouter un lien dans la page d’accueil qui permet à une personne de se connecter. Une fois que la connexion est établie le username de la personne sera affiché.
3. Ajouter un lien dans la page d’accueil qui permet à une personne de se déconnecter.
4. Vérifier l’accès à l’interface d’affichage de la liste des évènements. Une personne ne peut y accéder sans qu’elle se connecte à l’application.