# RAPPORT DE PROJET

Centre Européen de Formation
Assurez votre succès

# Sommaire

# Création du projet

## 1. Créer d’un environnement virtuel

J’ai commencé par créer un dossier nommé « mediatheque », puis dans un second temps, j’ai entré les lignes de commande suivantes dans le Terminal :

- cd mediatheque
- cd Scripts
- ./activate *(pour activer l’environnement virtuel)*
- cd.. *(pour revenir dans le répertoire « mediatheque »)*

## 2. **Installer Django** (toujours dans le Terminal)

- python -m ensurepip –upgrade *(pour installer le package pip)*
- pip install Django
- python
- >>>import django

## 3. Créer une application Django

- django-admin startproject bibliothecaire

# Création de l’app « bibliothecaire »

## 1. Créer des « models »

Un model représentant une table dans la base de données, dans ce projet nous n’avons besoins que de deux classes (qui définissent des objets) : « Membre » et « Media ».

*Clic droit sur le dossier « bibliothecaire » ” , puis « New », puis « Python file »
On nomme ce fichier « models » et tous nos models seront stockés dans ce fichier « models.py ».

*En haut du fichier, on doit tout d’abord importer les modules déjà présents lors de la création du projet :

```
from django.db import models
```

** code pour la class Media*

```
class Media(models.Model):  Tang77
    TYPE_MEDIA = [
        ('livre', 'Livre'),
        ('CD', 'CD'),
        ('DVD', 'DVD'),
        ('Jeu_de_plateau', 'Jeu de plateau'),
    ]

    name = models.CharField(max_length=100)
    creator = models.CharField(max_length=100)
    type_media = models.CharField(max_length=20, choices=TYPE_MEDIA)
    disponible = models.BooleanField(default=True)
    dateEmprunt = models.DateTimeField(null=True, blank=True)
    emprunteur = models.ForeignKey(Member, null=True, blank=True, on_delete=models.SET_NULL)

    def est_disponible(self):  2 usages (2 dynamic)  Tang77
        return self.disponible and (self.type_media != 'Jeu_de_plateau') and (self.dateEmprunt is None)

    def __str__(self):  Tang77
        return f"{self.name} ({self.get_type_media_display()})"
```

## 2. Créer des « views »

Les views sont responsables de la logique de traitement des requêtes HTTP et de la génération des réponses. Les views interagissent avec les models. Pour le projet, nous allons créer des views afin de répondre aux besoins (fonctionnalités) du site.

- Home
- Liste des membres
- Ajouter un nouveau membre
- Page détails d'un membre
- Modifier un membre
- Supprimer un membre
- Ajouter un média
- Liste des médias
- Ajouter un emprunt
- Liste des emprunts
- Retourner un emprunt

*Clic droit sur le dossier « bibliothecaire » " , puis « New », puis « Python file »
On nomme ce fichier « views » et tous nos views seront stockés dans ce fichier « views.py ».

**code du view pour ajouter un membre*

```
16  def add_member_view(request):  1 usage  Tang77
17      if request.method == 'POST':
18          creation_member = CreationMember(request.POST)
19          if creation_member.is_valid():
20              member = Member()
21              member.first_name = creation_member.cleaned_data['first_name']
22              member.last_name = creation_member.cleaned_data['last_name']
23              member.email = creation_member.cleaned_data['email']
24              member.tel = creation_member.cleaned_data['tel']
25              member.save()
26              return redirect( to: 'member_details', id=member.id)
27      else:
28          creation_member = CreationMember()
29          return render(request,  template_name: 'bibliothecaire/add_member.html',  context: {'creationMember': cre
30
```

## 3. Créer des « forms »

Comme le nom l’indique, il s’agit des formulaires à remplir afin d’ajouter des membres, des médias ou des emprunts. Il suffit d’importer les formulaires déjà existants dans Django et de déterminer les caractéristiques.

```
from django import forms
```

On créé une nouvelle fois des classes puisqu’il s’agit aussi d’objets.

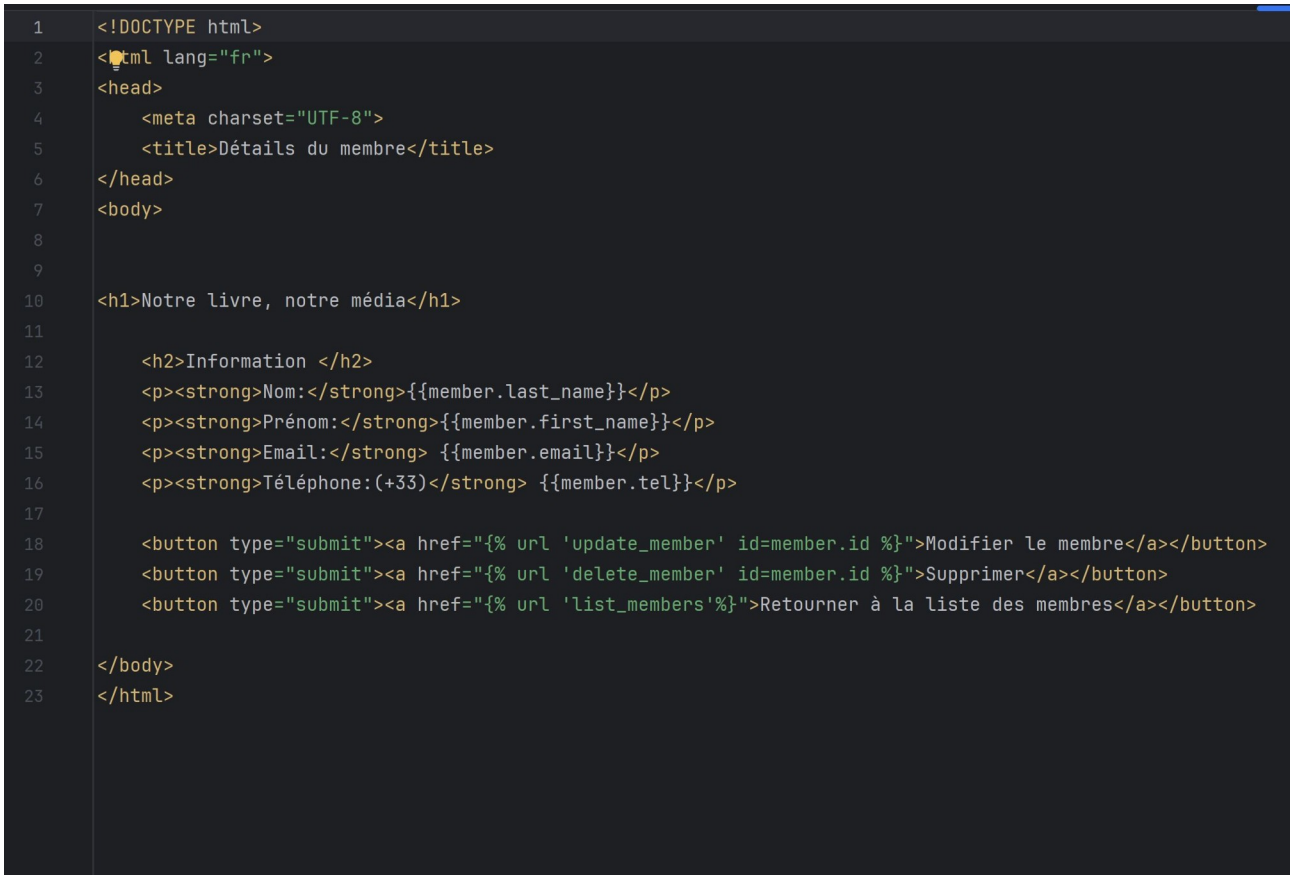
**code des formulaires*

```
class CreationMember(forms.Form):  5 usages  Tang77
    first_name = forms.CharField(required=True)
    last_name = forms.CharField(required=True)
    email = forms.EmailField(required=True)
    tel = forms.FloatField(required=True)

class CreationMedia(forms.Form):  3 usages  Tang77
    name = forms.CharField(label='Nom', max_length=100)
    creator = forms.CharField(label='Créé par', max_length=100)
    type_media = forms.ChoiceField(label='Type de Média', choices=Media.TYPE_MEDIA)


class EmpruntForm(forms.Form):  3 usages  Tang77
    member = forms.ModelChoiceField(queryset=Member.objects.all(), label="Membre")
    media = forms.ModelChoiceField(queryset=Media.objects.filter(disponible=True, type_media__in=['livre', '
```

## 4. Créer des « templates »

Les templates doivent être placés dans un dossier ayant le même nom que l'application « bibliothecaire » et celui-ci se situant dans un autre dossier nommé « templates ». Le code sera écrit dans des fichiers de type html.

**code du template de la page de détails d'un membre*



```
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <meta charset="UTF-8">
5      <title>Détails du membre</title>
6  </head>
7  <body>
8
9
10 <h1>Notre livre, notre média</h1>
11
12     <h2>Information </h2>
13     <p><strong>Nom:</strong>{{member.last_name}}</p>
14     <p><strong>Prénom:</strong>{{member.first_name}}</p>
15     <p><strong>Email:</strong> {{member.email}}</p>
16     <p><strong>Téléphone:(+33)</strong> {{member.tel}}</p>
17
18     <button type="submit"><a href="{% url 'update_member' id=member.id %}">Modifier le membre</a></button>
19     <button type="submit"><a href="{% url 'delete_member' id=member.id %}">Supprimer</a></button>
20     <button type="submit"><a href="{% url 'list_members'%}">Retourner à la liste des membres</a></button>
21
22 </body>
23 </html>
```

*ligne 13 : {{member.last_name}}

permet d'aller chercher le nom de famille du membre dans la base de données

*ligne 28 : <button type="submit"><a href="{% url 'list_members'%}">Retourner à la liste des membres</a></button>

bouton permettant d'aller à la page de la liste des membres, redirigeant vers l'url de cette page.

## 5. Lancer la migration

La migration sert à importer les données et à gérer les modifications de la base de données. La structure de la base de données est définie par les « models ». Django ne faisant pas la mise à jour automatiquement, nous devons donc le faire manuellement par des lignes de commandes dans le terminal.

Tout d'abord, dans le fichier « settings.py », il faut ajouter l'application « bibliothecaire » dans « INSTALLED_APPS = [] ».

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'bibliothecaire',
    'membres',
]
```

Puis s'assurer que le nom de la base de données est bien présent dans « DATABASES = [] »

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
```

Lignes de commandes pour effectuer la migration :

- python manage.py makemigrations *(importer la structure)*
- python manage.py migrate *(importer les données)*

Une base de données est créée au même niveau que le fichier manage.py et est nommée : « db.sqlite3 ».

## 6. Déterminer des urls

Les urls servent à définir les routes qui permettent de connecter les requêtes HTTP aux « views » correspondants.

- Première étape : créer un fichier python nommé « urls.py ».
- Dans ce fichier, importer les éléments suivants :

```
from django.urls import path, include
from . import views
```

- Puis définir les « paths » :

```python
from django.urls import path, include
from . import views

urlpatterns = [
    path('', views.home, name='bibliothecaire_home'),
    path('member/list', views.list_members_view, name='list_members'),
    path('member/add_member', views.add_member_view, name='add_member'),
    path('member/<int:id>', views.member_details_view, name='member_details'),
    path('member/<int:id>/update_member', views.update_member_view, name='update_member'),
    path('member/<int:id>/delete', views.delete_member_view, name='delete_member'),
    path('media/add_media', views.add_media_view, name='add_media'),
    path('media/list_media', views.list_media_view, name='list_media'),
    path('emprunt/add_emprunt', views.add_emprunt_view, name='add_emprunt'),
    path('emprunt/list_emprunt', views.list_emprunt_view, name='list_emprunt'),
    path('media/<int:id>/return/', views.return_emprunt_view, name='return_emprunt'),
    path('emprunt/error', views.error_view, name='error_emprunt'),
    path('membres/', include('membres.urls'))

    ]
```

# 7. Mettre en place des tests

- La première étape pour implémenter des test, on doit d'abord configurer l'environnement de test. Plusieurs méthodes existent, j'ai choisi d'utiliser « pytest ».

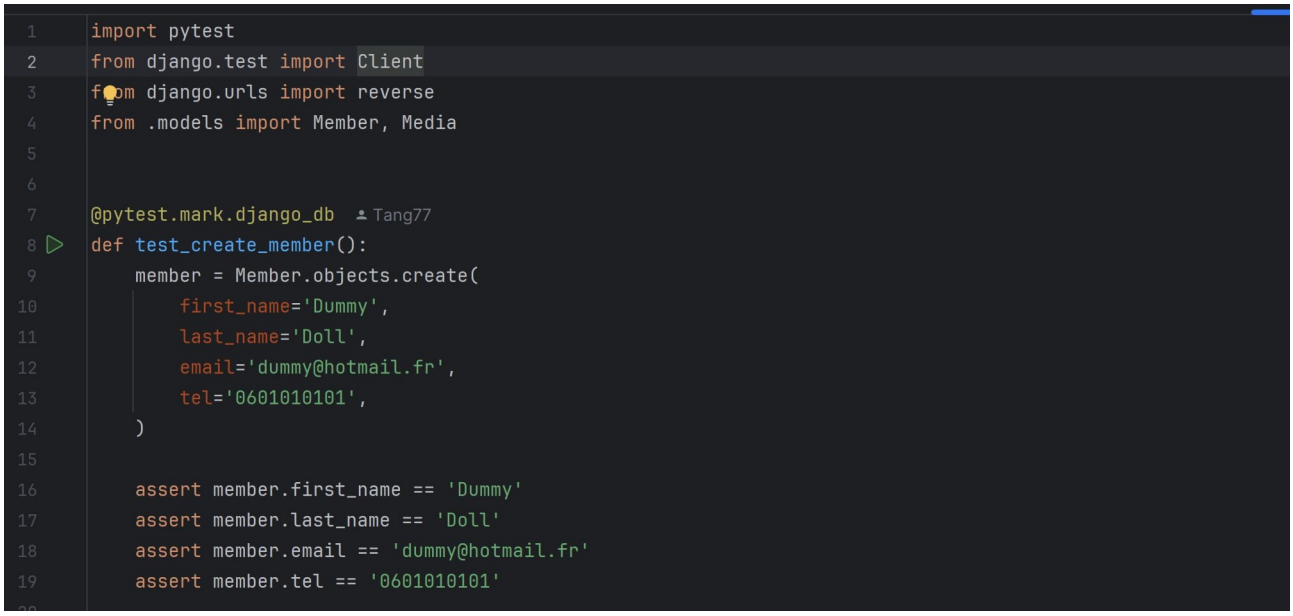
  Installation de « pytest » dans le Terminal :

  ```
  pip install pytest-django
  ```

- Dans PyCharm, au même niveau que « manage.py », créer un fichier text nommé « pytest.ini » et écrire le code suivant :

  ```
  [pytest]
  DJANGO_SETTINGS_MODULE = bibliothecaire.settings
  python_files = test.py test_*.py *_tests.py
  ```

- Les codes des tests seront écrits dans un fichier python nommé « test_bibliothecaire.py ».



```python
import pytest
from django.test import Client
from django.urls import reverse
from .models import Member, Media


@pytest.mark.django_db  Tang77
def test_create_member():
    member = Member.objects.create(
        first_name='Dummy',
        last_name='Doll',
        email='dummy@hotmail.fr',
        tel='0601010101',
    )

    assert member.first_name == 'Dummy'
    assert member.last_name == 'Doll'
    assert member.email == 'dummy@hotmail.fr'
    assert member.tel == '0601010101'
```

# Création de l’app « membres »

On procède de la même façon que pour le création de l’application « bibliothecaire ».

L’app « membres » est placé dans le dossier racine de l’app « bibliothecaire ». Elle partagera les models de « bibliothecaire » mais aura ses propres views, urls et templates.

```python
from django.shortcuts import render
from bibliothecaire.models import Media


def liste_medias(request):  1 usage  Tang77
    medias = Media.objects.all()
    return render(request, template_name: 'membres/liste_medias.html', context: {'medias': medias})
```

```python
from django.urls import path
from . import views

urlpatterns = [
    path('medias/', views.liste_medias, name='liste_medias'),
]
```

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Liste des médias</title>
</head>
<body>
    <h1>Liste des Médias</h1>
    <table>
        <thead>
            <tr>
                <th>Nom</th>
                <th>Créateur</th>
                <th>Type</th>
                <th>Disponibilité</th>
                <th>Emprunteur</th>
            </tr>
        </thead>
        <tbody>
            {% for media in medias %}
                <tr>
                    <td>{{ media.name }}</td>
                    <td>{{ media.creator }}</td>
                    <td>{{ media.get_type_media_display }}</td>
                    <td> {% if media.disponible %}
                            <span style="...">Disponible</span>
                        {% else %}
                            <span style="...">Indisponible</span>
                        {% endif %}
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
```