

1. Installation d'un environnement virtuel

- Télécharger et installer virtualenvwrapper-win
- Créer un environnement virtuel avec la commande **mkvirtualenv <name>**, <name> est le nom que vous donnez à l'environnement virtuel créé
Nous choisirons **name=django5**
mkvirtualenv django5
- Votre nouveau environnement virtuel est activé après création, si non faites **workon django5**

```
(django5) C:\Users\leoni>
```

- Installer django avec **pip**
pip install django

```
(django5) C:\Users\leoni\CoursDjango>pip install django
Collecting django
  Using cached Django-5.0.3-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.7.0 (from django)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse<=0.3.1 (from django)
  Using cached sqlparse-0.4.4-py3-none-any.whl.metadata (4.0 kB)
Collecting tzdata (from django)
  Using cached tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached Django-5.0.3-py3-none-any.whl (8.2 MB)
Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.0.3 sqlparse-0.4.4 tzdata-2024.1
```

2. Création d'un projet django

- Créer un dossier **<CoursDjango>** dans lequel vous allez mettre tous vos projets :
- Placez-vous dans ce dossier
- Créer votre premier projet : **django-admin startproject HelloWorld**
<HelloWorld> est le nom de votre nouveau projet.
- Avec la commande « Tree /f », visualisez l'arborescence de votre projet :

```
(django5) C:\Users\leoni\CoursDjango>tree /f
Structure du dossier pour le volume OS
Le numéro de série du volume est 26FB-2034
C:
├── HelloWorld
│   ├── manage.py
│   └── HelloWorld
│       ├── asgi.py
│       ├── settings.py
│       ├── urls.py
│       ├── wsgi.py
│       └── __init__.py
```

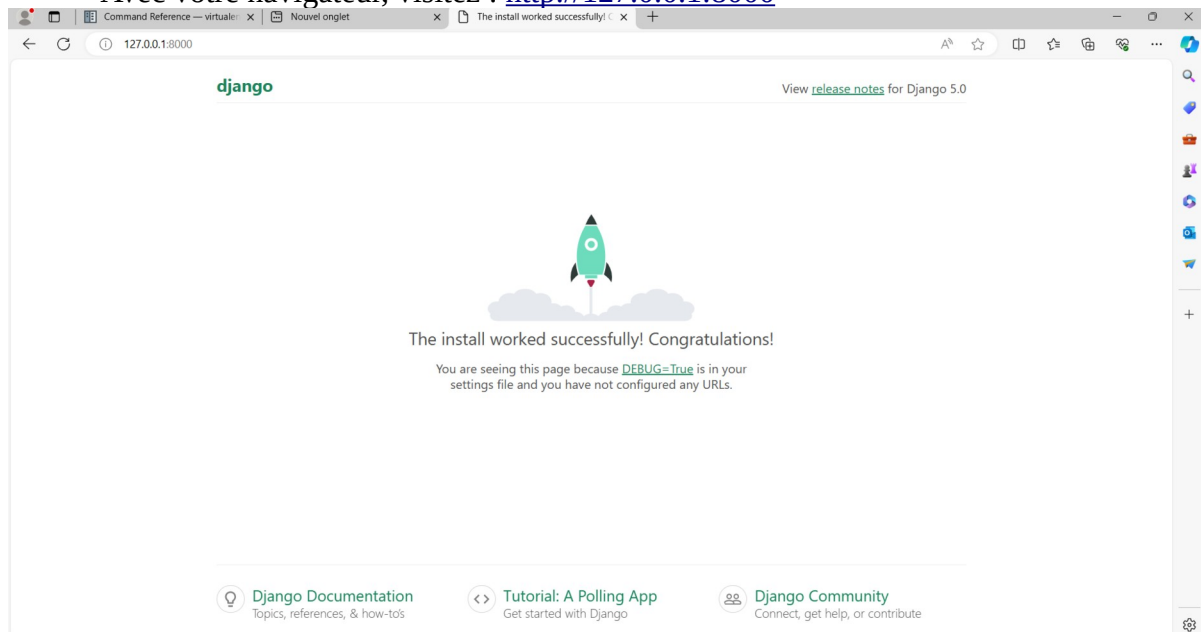
- Placez-vous dans le dossier HelloWorld
- Lancer la commande **py manage.py runserver**

```
(django5) C:\Users\leoni\CoursDjango\HelloWorld>py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
March 30, 2024 - 08:57:17
Django version 5.0.3, using settings 'HelloWorld.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- Avec votre navigateur, visitez : <http://127.0.0.1:8000>



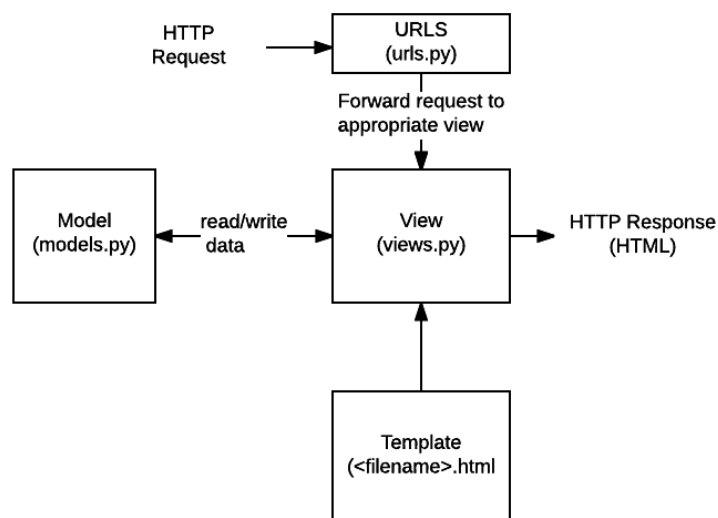
3. Création d'une application au sein du projet

- tapez : **py manage.py startapp appHelloWorld**
Ici le nom de l'application est appHelloWorld

- Avec la commande `tree /f`, visualisez l'arborescence de votre projet :

```
(django5) C:\Users\leoni\CoursDjango\HelloWorld>tree /f
Structure du dossier pour le volume OS
Le numéro de série du volume est 26FB-2034
C:.\
|-- db.sqlite3
|-- manage.py
|-- appHelloWorld
|   |-- admin.py
|   |-- apps.py
|   |-- models.py
|   |-- tests.py
|   |-- views.py
|   |-- __init__.py
|   |-- migrations
|   |   |-- __init__.py
|-- HelloWorld
|   |-- asgi.py
|   |-- settings.py
|   |-- urls.py
|   |-- wsgi.py
|   |-- __init__.py
|-- __pycache__
|   |-- settings.cpython-312.pyc
|   |-- urls.cpython-312.pyc
|   |-- wsgi.cpython-312.pyc
|   |-- __init__.cpython-312.pyc
```

4. Fonctionnement de Django



- C'est la requête HTTP qui initie la navigation. Cette requête contient les informations qui seront traitées par le fichier **urls.py**. Ce fichier fait une correspondance entre les URLs de la requête et les fonctions du fichier **views.py**
- C'est à la fonction de **views.py** sélectionnée de déterminer quelle sera la réponse de la requête. Pour cela la fonction de **views.py** en question peut interagir avec les bases de données, les templates etc.

Exemple sur notre projet HelloWorld

Dans le fichier settings de HelloWorld, nous devons renseigner(ajouter) l'application appHelloWorld,

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

Supposons que nous voulons que l'utilisateur tape <http://127.0.0.1:8000/welcome> pour recevoir un texte de bienvenu.

Nous créons dans l'application 'appHelloWorld', le fichier 'urls.py'.

```
appHelloWorld > 🐘 urls.py > ...  
1  from django.urls import path  
2  from . import views  
3  
4  urlpatterns=[path('welcome', views.hello)]
```

c'est urlpatterns qui fait la correspondance entre les URLs et les fonctions à sélectionner dans views.

Modifions le fichier urls.py du repertoire HelloWorld

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('appHelloWorld.urls'))
]
```

Et modifions le fichier views.py comme suit :

```
1  ∨ from django.shortcuts import render
2    from django.http import HttpResponse
3
4    # Create your views here.
5
6  ∨ def hello(request):
7      return HttpResponse("Bonjour, Soyez le bienvenu!!!")
```

Faisons la migration avec la commande : **py manage.py makemigrations**
et **py manage.py migrate**

```
(django5) C:\Users\leoni\CoursDjango\HelloWorld>py manage.py makemigrations
No changes detected
```

```
(django5) C:\Users\leoni\CoursDjango\HelloWorld>py manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

relançons le serveur avec la commande : **py manage.py runserver**

```
(django5) C:\Users\leoni\CoursDjango\HelloWorld>py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 30, 2024 - 10:57:11
Django version 5.0.3, using settings 'HelloWorld.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Tapez <http://127.0.0.1:8000/welcome>



Si vous voyez l'écran ci-dessus, tout est bon, félicitations :)
Dans le cas contraire, vous devez faire troubleshooting.

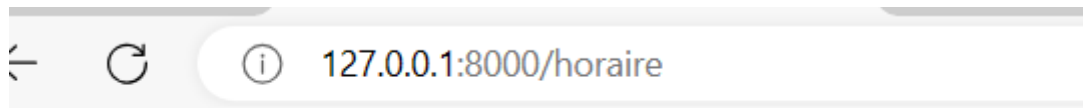
Supposons que nous voulons que l'utilisateur tape 127.0.0.1/horaire pour voir l'heure :

Ajoutons la fonction `horaire()` dans **views.py** :

```
def horaire(request):
    return HttpResponse("<html> <ul> <li> manger </li><li>Dormir </li> </ul></html> ")
```

Créons l'entrée correspondante dans **urls.py** de **appHelloWorld**

```
urlpatterns=[path('welcome', views.hello),
             path('horaire', views.horaire)]
```



- manger
- Dormir

Les templates

Les templates font la liaison entre python et les pages html. Les valeurs calculées ou retournées par python sont insérées dans les pages html à l'aide des templates.

Exemple 1 :

Créons un nouveau projet **Programming** et une nouvelle application **appProgramming**. N'oublions pas d'ajouter **appProgramming** dans les applications installées dans le projet de Programming, et de faire la migration.

```
appProgramming > views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponse
3  from django.template import loader
4  # Create your views here.
5
6  def code(request):
7
8      template=loader.get_template("languages.html")
9      D={"title":"Langes", "py":"python","li":"lisp"}
10     context={"D":D}
11
12     return HttpResponse(template.render(context,request))
13
```

Modifions (ou créons si le fichier n'existe pas) les fichiers views.py et les 2 fichiers urls.py (l'un dans le projet principal et l'autre dans l'application appProgramming).

```
appProgramming > urls.py > ...
1  from django.urls import path
2  from . import views
3
4
5  urlpatterns=[path('code',views.code)]
```

```

Programming > 🗑️ urls.py > ...
17  from django.contrib import admin
18  from django.urls import path,include
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('',include("appProgramming.urls"))
23  ]
24

```

Dans l'application appProgramming, créons le répertoire **templates**, dans lequel nous allons mettre les templates de notre applicaion. Dans ce template, ajoutons le fichier suivant :

```

appProgramming > templates > <> languages.html > 📁 html > 📁 head > 📁 title
1  <html>
2      <head>
3          <title>
4              {{D.title}}
5          </title>
6      </head>
7      <body>
8          Hello      {{D.py}} {{D.li}}
9
10     </body>
11 </html>

```

Explication :

Le fichier **views.py** contient un dictionnaire D. C'est ce dictionnaire qui inséré dans le template « **langages.html** ». D.py donne la valeur de la clé « py », etc.

La fonction « loader » permet de charger la template « langages.html ».

Exemple 2 :

Reppprenons l'exemple des ménages que nous avons fait durant le test dernier. Il nous faut donner les résultats comme résultat des requêtes HTTP :

Nous devons répondre à deux requêtes : 1. revenu total par ménage
2. revenu moyen par ménage


```

appMenages > views.py > ...
1  from django.shortcuts import render
2  from django.template import loader
3  from django.http import HttpResponse
4  from statistics import mean
5
6  # Create your views here.
7
8  Menages={
9      "menage1":{"papa":1800, "mama":2500,"enfant11":800,"enfant12":1600},
10     "menage2":{"papa":0, "mama":2800,"enfant21":1100,"enfant22":1500},
11     "menage3":{"papa":1800, "mama":2500,"enfant31":800, "enfant32":1600,"enfant33":1200},
12     "menage4":{"papa":1800, "mama":2500},
13     "menage5":{"mama":2500,"enfant51":1000,"enfant52":1500},
14     "menage6":{"papa":1300, "mama":2500,"enfant51":800,"enfant52":1600},
15     "menage7":{"papa":1800, "mama":0,"enfant71":800,"enfant72":1600,"enfant73":1200},
16     "menage8":{"papa":1800,"enfant81":800,"enfant82":1600,"enfant83":1200},
17 }
18 #Revenu total de chaque menage(en utilisant comprehension dictionary)
19 #Revenu_total_par_menage={key:sum(value.values()) for key,value in Menages.items()}
20
21 def listeMenages(request):
22     Revenu_total_par_menage={key:sum(value.values()) for key,value in Menages.items()}
23     context={"Revenu_par_menage":Revenu_total_par_menage}
24
25     template=loader.get_template('revenu_par_menage.html')
26     return HttpResponse(template.render(context,request))
27
28 def revenuMoyen(request):
29     revenu_moyen_par_menage={key:round(mean(value.values()),2) for key,value in Menages.items()}
30     context={'revenuMoyen':revenu_moyen_par_menage}
31     template=loader.get_template('revenuMoyen.html')
32     return HttpResponse(template.render(context,request))

```

```

appMenages > urls.py > ...
1  from django.urls import path
2  from . import views
3
4
5
6
7  urlpatterns=[path('listeMenages',views.listeMenages),
8               path('revenuMoyen',views.revenuMoyen)]

```

Nous devons créer 2 fichiers html dans le dossier **templates**.

```

v MENAGES
v appMenages
> __pycache__
> migrations
v templates
  <> revenu_par_menage.htm
  <> revenuMoyen.html

```

```

appMenages > templates > < revenu_par_menage.html > html
1  <html>
2    <head>
3      <title>
4        Revenu par menage
5      </title>
6    </head>
7    <body>
8      Revenu par menage: <br>
9      {%for k,v in Revenu_par_menage.items %}
10     | {{k}}:{{v}} <br>
11     | {%endfor%}
12   </body>
13 </html>

```

```

appMenages > templates > < revenuMoyen.html > html > body
1  <html>
2    <head>
3      <title>
4        Revenu  moyen par menage
5      </title>
6    </head>
7    <body>
8      Revenu par menage: <br>
9      {%for k,v in revenuMoyen.items %}
10     | {{k}}:{{v}} <br>
11     | {%endfor%}
12   </body>
13 </html>

```

{ % **instruction** %} : pour écrire des instructions
 {{ **variable** }} : pour évaluer les variables

```
← ↻ ⓘ 127.0.0.1:8000/revenuParMenage

Revenu par menage:
menage1:6700
menage2:5400
menage3:7900
menage4:4300
menage5:5000
menage6:6200
menage7:5400
menage8:5400
```

```
← ↻ ⓘ 127.0.0.1:8000/revenuMoyen

Revenu par menage:
menage1:1675
menage2:1350
menage3:1580
menage4:2150
menage5:1666.67
menage6:1550
menage7:1080
menage8:1350
```

Les modèles en Django

Les modèles en django sont des fichiers .py qui créent dans la base de données les tables. Par défaut, django utilise **sqlite** comme base de données. Les models sont créés dans le fichier « models » de django

Exemple

Créons un nouveau projet **Personnel** et une applicaion **appPersonnel**.

N'oublions pas d'insérer 'appPersonnel' dans les applications du projet « Personnel ».

Nous créons le fichier **models.py** dans appPersonnel :

```
settings.py  models.py X  admin.py

appPersonnel > models.py > Personne
1  from django.db import models
2
3
4
5  # Create your models here.
6
7
8  class Personne(models.Model):
9      name=models.CharField(max_length=50)
10     age=models.IntegerField()
11     salaire=models.IntegerField()
12
13
```

Nous créons le fichier **admin.py** dans l'**appPersonnel**.

```
settings.py  models.py  admin.py X

appPersonnel > admin.py
1  from django.contrib import admin
2  from .models import Personne
3  # Register your models here.
4
5  admin.site.register(Personne)
```

Le fichier **admin.py** permet d'avoir le modèle dans l'interface d'administration.
Mais nous devons créer d'abord l'utilisateur « admin » pour administrer la base de données.

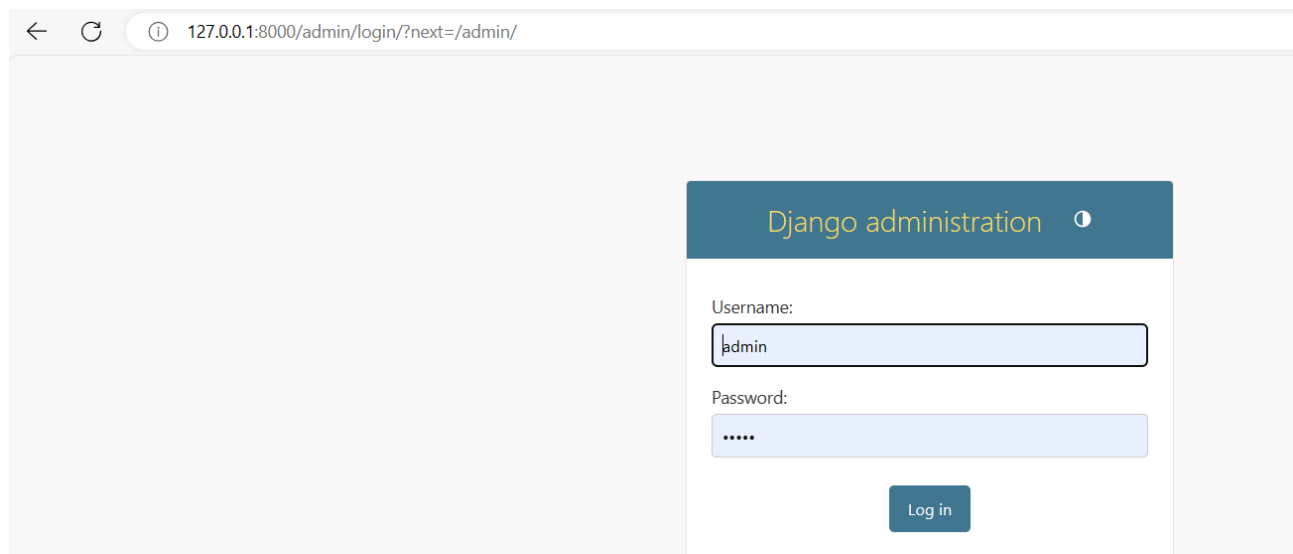
```
(django5) C:\Users\leoni\CoursDjango\Personnel>py manage.py createsuperuser
Username (leave blank to use 'leoni'): admin
Email address: admin
Error: Enter a valid email address.
Email address: admin@admin.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Nous devons faire la migration créer le modèle dans la base de données :

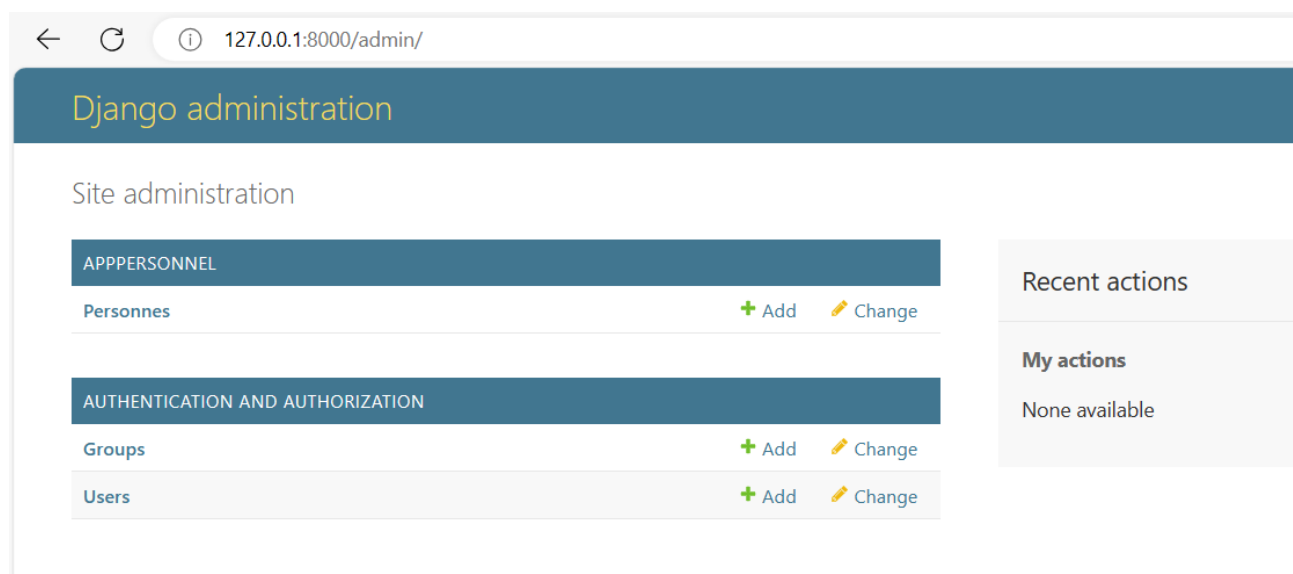
```
(django5) C:\Users\leoni\CoursDjango\Personnel>py manage.py makemigrations
Migrations for 'appPersonnel':
  appPersonnel\migrations\0001_initial.py
    - Create model Personne

(django5) C:\Users\leoni\CoursDjango\Personnel>py manage.py migrate
Operations to perform:
  Apply all migrations: admin, appPersonnel, auth, contenttypes, sessions
Running migrations:
  Applying appPersonnel.0001_initial... OK
```

Et on peut atteindre l'interface d'administration :



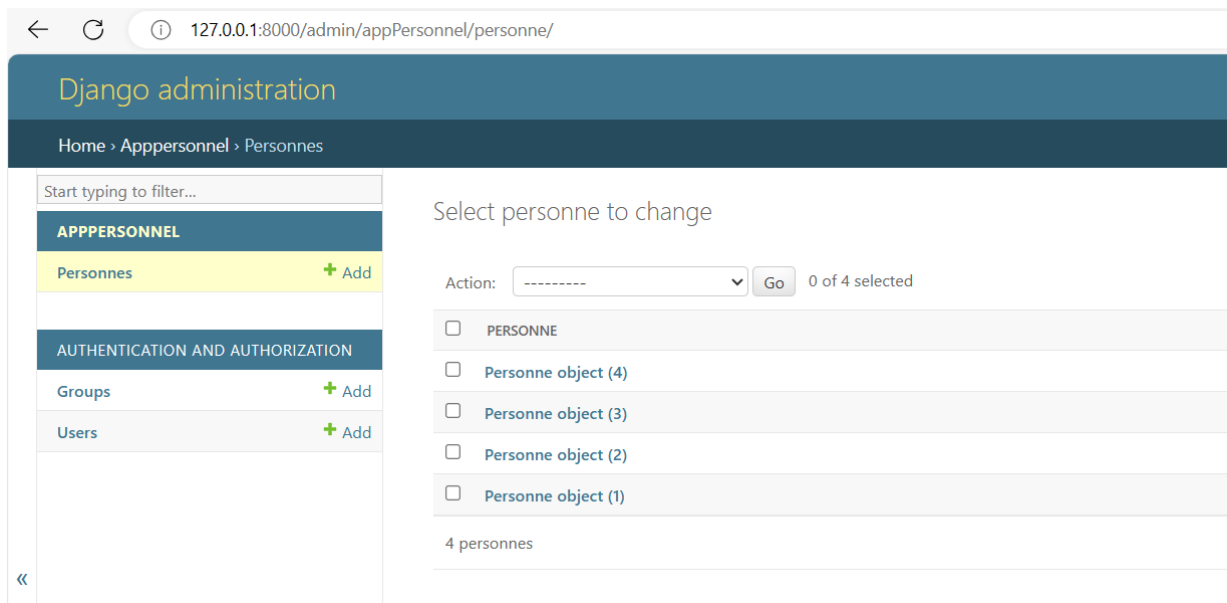
The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/admin/login/?next=/admin/'. The page features a login form titled 'Django administration'. The form includes a 'Username:' field with the text 'admin' entered, a 'Password:' field with masked characters '.....', and a 'Log in' button.



The screenshot shows the Django administration site home page. The address bar displays '127.0.0.1:8000/admin/'. The page has a dark blue header with the text 'Django administration'. Below the header, the 'Site administration' section is visible, containing two main categories: 'APPPERSONNEL' and 'AUTHENTICATION AND AUTHORIZATION'. Under 'APPPERSONNEL', there is a link for 'Personnes' with '+ Add' and 'Change' buttons. Under 'AUTHENTICATION AND AUTHORIZATION', there are links for 'Groups' and 'Users', each with '+ Add' and 'Change' buttons. On the right side, there is a 'Recent actions' section with a 'My actions' subsection, which currently shows 'None available'.

Remarquons qu'il y a un « s » s'il la classe « **Personne** ».

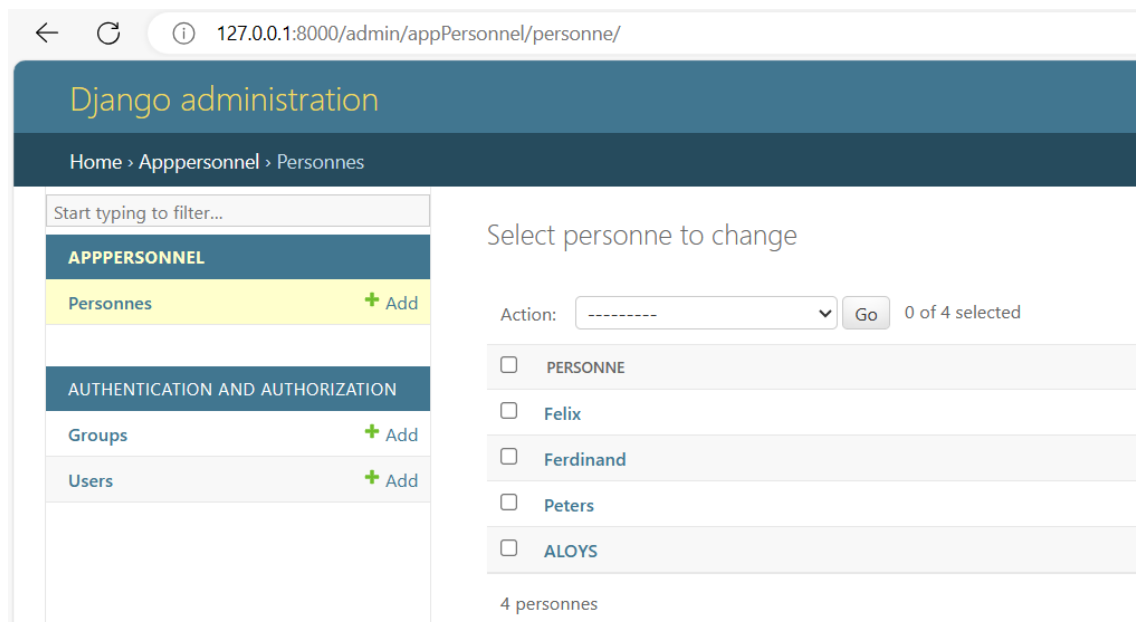
Nous pouvons créer les différentes personnes suivant notre modèle en cliquant sur « **Personnes** »



Les personnes créées peuvent lister en cliquant sur « **Personnes** ». Elles sont « **Personne object** ».

En ajoutant le code ci-dessous dans le modèle **Personne**, on aura les « **name** » pour désigner les personnes:

```
def __str__(self):  
    return self.name
```



Nous allons afficher la liste des personnes qui sont dans la base de données en utilisant les templates

```
settings.py  models.py  views.py  X  urls.py appPersonnel  urls.py Personne

appPersonnel > views.py > listerPersonnes
1  from django.shortcuts import render
2  from .models import Personne
3  from django.template import loader
4  from django.http import HttpResponse
5
6  # Create your views here.
7
8
9  def listerPersonnes(request):
10     personnes=Personne.objects.all().values()
11     template=loader.get_template('personnes.html')
12     context={"personnes":personnes}
13     return HttpResponse(template.render(context,request))
14
```

```
settings.py  models.py  views.py  urls.py appPersonnel  X  urls.py Personne  <> personnes.ht

appPersonnel > urls.py > ...
1
2  from django.urls import path
3  from . import views
4
5
6  urlpatterns=[path('listerPersonnes',views.listerPersonnes)]
```

```
settings.py  models.py  views.py  urls.py appPersonnel  urls.py Personne  X

Personnel > urls.py > ...
10
17  from django.contrib import admin
18  from django.urls import path,include
19
20  urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('',include('appPersonnel.urls'))
23 ]
24
```

```
settings.py  models.py  views.py  urls.py appPersonnel  urls.py Personnel  personnes.html X  admin.py

appPersonnel > templates > personnes.html > html > body
1  <html>
2      <head>
3          <title>
4              Liste des Personnes
5          </title>
6      </head>
7      <body>
8          La liste des personnes:
9          <ul>
10             {% for p in personnes%}
11             <li>{{p.name}}</li> <br>
12             {%endfor%}
13          </ul>
14      </body>
15 </html>
```



Les formulaires

Django nous permet la création des formulaires à partir des modèles. (Don't repeat yourself).

Créons un fichier formPersonne.py

```
from django import forms
from .models import Personne

class formulairePersonne(forms.ModelForm):
    class Meta:
        model=Personne
        fields=['name','age','salaire']
```

Ce fichier, dans sa classe « Meta » indique à django d'utiliser le modèle Personne, les champs « name », « age » et « salaire »

On change le fichier « views », en ajoutant la fonction nous permettant d'enregistrer les personnes :

```
def savePersonne(request):
    if request.method=="POST":
        form=formulairePersonne(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect("success, well saved")
        else:
            form=formulairePersonne()

    template=loader.get_template("formulaire.html")
    context={"form":form}
    return HttpResponseRedirect(template.render(context,request))
```

On crée dans template le fichier « formulaire.html »

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Formulaire </title>
</head>
<body>
  <span> Formulaire </span>: <br>
  <form method="Post" action="savePersonne">
    {{form.as_p}}
    {% csrf_token %}
    <input type="submit" value="enregistrer">
  </form>
</body>
</html>

```

« form » est une instance de la classe formPersonne.

« csrf_token » est pour la sécurité des formulaires.

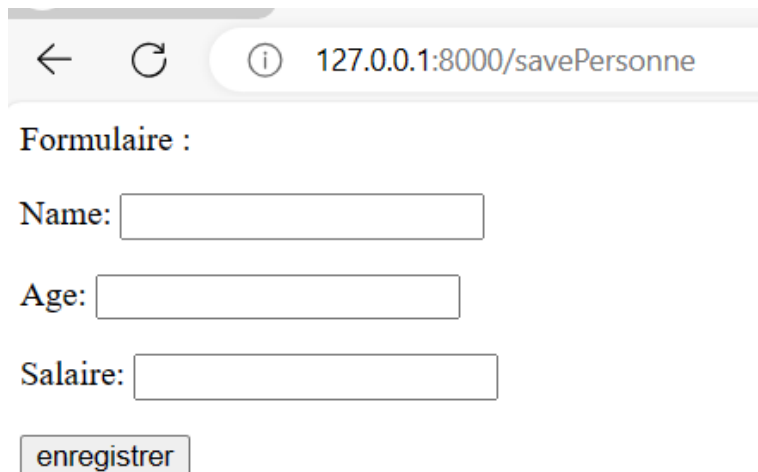
On modifie ensuite « urls.py » de « appPersonne » :

```

urlpatterns=[path('listerPersonnes',views.listerPersonnes),
             path('savePersonne',views.savePersonne)]

```

Et enfin on visite :



← ↻ ⓘ 127.0.0.1:8000/savePersonne

Formulaire :

Name:

Age:

Salaire:

On enregistre les personnes et on peut afficher les personnes se trouvant dans la base de données :



127.0.0.1:8000/listerPersonnes

La liste des personnes:

- ALOYS
- Peters
- Ferdinand
- Felix
- kilo
- Muka