

Programmation Web et framework Django

Prof . Samuel OUYA

Partie I : Initiation à la programmation web

Service Web

1- serveur Web apache2

Apache2 est le serveur web le plus utilisé dans le monde

Il est modulaire. On peut lui ajouter des modules pour lui faire des rôles supplémentaires autre que d'héberger des sites web ,

Il 3 paramètres fondamentaux à fournir à apache2 pour l'hébergement :

a- Le dossier dans lequel on doit copier le contenu du site web à travers le parametre

DocumentRoot

exemple :

DocumentRoot //var/www/html

b- **Le nom** du fichier page d'accueil à travers le parametre **DirectoryIndex**

Exemple :

DirectoryIndex index.html

c- Le nom à saisir sur un client HTTP pour accéder au site à travers le parametre

ServerName

Exemple :

ServerName toto.ec2lt.sn

Types d'hébergement de sites Web

Il ya plusieurs manières d'héberger de sites web sous apache2

1,1 Site hébergé par dossier

methode :

On cree un sous dossier dans le DocumentRoot du site principal qui est par défaut /var/www/html et on y copie le contenu de notre site

Pour accéder à un tel site à travers un navigateur :

<http://IPSERVEUR/Nomdossier/fichiervoulu>

exemple :

Si le contenu du site est copié dans /var/www/html/tp et contient le fichier index.html

<http://localhost/tp/index.html>

1,2 Site virtuel par nom

la mise en place d'un site virtuel par nom se fait en plusieurs etapes :

a- on cree le fichier de configuration du site dans le dossier /etc/apache2/sites-available par exemple site1.conf

```
<VirtualHost *:80>
ServerName toto.ec2lt.sn
DocumentRoot /var/www/html/tp
DirectoryIndex index.html
</VirtualHost>
```

b- on active le site par la commande

```
root@alain-VirtualBox:/var/www/html/tp# a2ensite site1.conf
Enabling site site1.
To activate the new configuration, you need to run:
systemctl reload apache2
```

c- on recharge le serveur apache2 par la commande

systemctl reload apache2

1.3 format general d'un fichier html

```
<html>
<head>
<title>
Mon premier Site
</title>
</head>
<body>
Voici le corps de mon premier site
</body>
</html>
```

NB : comme on n'a pas de serveur DNS, on peut faire une resolution locale du nom toto.ec2lt.sn dans le fichier `/etc/hosts`

127.0.0.1 localhost toto.ec2lt.sn

Test de notre Site



fig

2 Les balises html table et form

2.1 Balise table

La balise html table permet de créer un tableau
table a l'attribut border qui permet de donner la taille en pixel des bordures de notre tableau

la balise <tr> permet de definir une ligne de notre tableau

la balise <td> permet de definir une colonne

Exemple creeons un tables de presidents

tableau.html dans le dossier /var/www/html/tp

```
<html>
<head>
<title>
Mon deuxieme Site
</title>
</head>
<body>
<table border='1px'>
<th> Tableau des presidents </th>
<tr><td>Prenom</td><td>Nom</td><td>Email</td><td>Code</td><td>Solde</td><tr>
<tr><td>Macky</td><td>Sall</td><td>macky@gouv.sn</td><td>1111</td><td>250000</td></tr>
<tr><td>Abdoulaye</td><td>Wade</td><td>Abdoulaye@rtn.sn</td><td>2222</
td><td>4500000</td></tr>
<tr><td>Denis</td><td>Sassou</td><td>sassou@ec2lt.sn</td><td>3333</td><td>3050000</
td></tr>
</table>
</body>
</html>
```

Test



Tableau des presidents				
Prenom	Nom	Email	Code	Solde
Macky	Sall	macky@gouv.sn	1111	250000
Abdoulaye	Wade	Abdoulaye@rtn.sn	2222	4500000
Denis	Sassou	sassou@ec2lt.sn	3333	3050000

2.2 Foruulaire html

Un formulaire html permet de recueillir des données des utilisateurs et les faire traiter par un programme

On utilise pour cela la balise form

Exemple creons un formulaire d'enregistrement d'un president et les données seront traitées par le programme create.py

```
<html>
```

```

<head>
<title>
Mon troisieme Site
</title>
</head>
<body>
<form method ='POST' action = create.py >
Prenon:<input type ='text" name = 'nom'> <br>
Non:<input type ='text" name = 'prenom'> <br>
Email:<input type ='text" name = 'email'> <br>
Code:<input type ='text" name = 'code' > <br>
Solde:<input type ='text" name = 'solde'> <br>
<input type ="submit" value = "Valider"/>
</form>
</body>
</html>

```

resultat



The screenshot shows a web browser window with the address bar displaying 'localhost/tp/formulaire.html'. The page content is a form with five text input fields, each preceded by a label: 'Prenon:', 'Non:', 'Email:', 'Code:', and 'Solde:'. Below these fields is a button labeled 'Valider'.

Ameliorons notre formulaire en le mettant dans un tableau pour bien aligner les champs

```

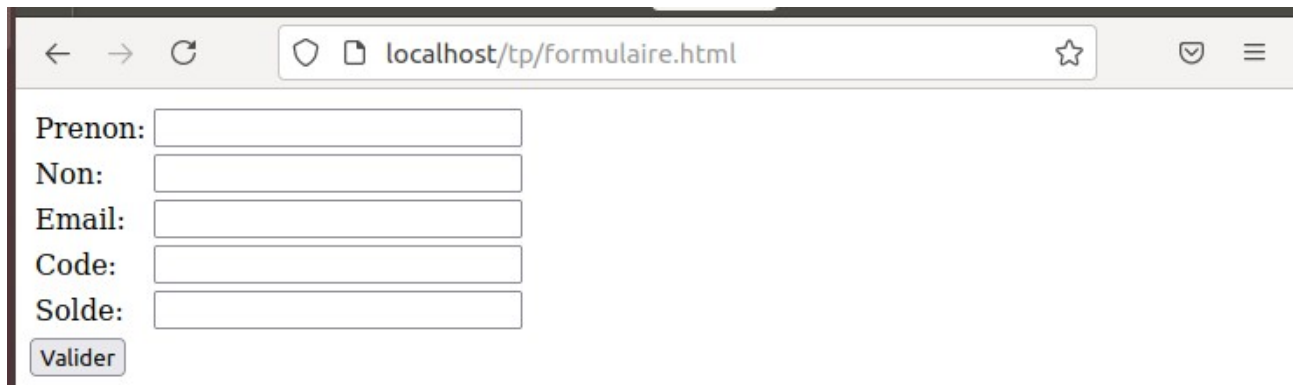
<html>
<head>
<title>
Mon troisieme Site
</title>

</head>
<body>
<form method ='POST' action = create.py >
<table>
<tr><td>Prenon:</td><td><input type ='text" name = 'nom'></td> </tr>
<tr><td>Non:</td><td><input type ='text" name = 'prenom'></td> </tr>
<tr><td>Email:</td><td><input type ='text" name = 'email'></td> </tr>
<tr><td>Code:</td><td><input type ='text" name = 'code' ></td> </tr>
<tr><td>Solde:</td><td><input type ='text" name = 'solde'></td> <tr>

```

```
</table>
<input type="submit" value = "Valider"/>
</form>
</body>
</html>
```

Resultat



The screenshot shows a web browser window with the address bar displaying 'localhost/tp/formulaire.html'. The page content consists of a form with five text input fields stacked vertically. The labels for these fields are 'Prenon:', 'Non:', 'Email:', 'Code:', and 'Solde:'. Below the last input field is a button labeled 'Valider'.

3 - les cgi

CGI (Common Gateway Interface) définit une méthode d'interaction entre un serveur web et des programmes générateurs de contenu externes, plus souvent appelés programmes CGI ou scripts CGI. Il s'agit d'une méthode simple pour ajouter du contenu dynamique à votre site web en utilisant votre langage de programmation préféré

3.1 installation de python3 et activation du module cgi d'apache 2

```
apt-get update
apt-get install python3 python3-pip
```

a2enmod cgi

puis on redemarre apache2

```
systemctl restart apache2
```

Par défaut, vous devez placer vos scripts cgi dans le dossier /usr/lib/cgi-bin

Exemple 1 de cgi

Allons dans ce dossier et creons notre premier script cgi

```
cd /usr/lib/cgi-bin
```

```
vim /usr/lib/cgi-bin/test.py
```

```
#!/usr/bin/python3
```

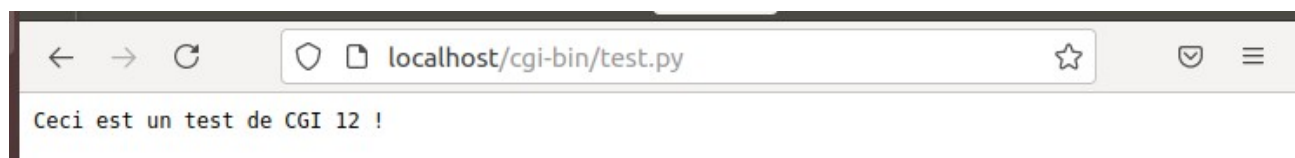
```
print('Content-Type: text/plain')
print("")
print('Ceci est un test de CGI 12 !')
```

Il faut votre programme executable

chmod 755 /usr/lib/cgi-bin/test.py

Pour acceder au script-cgi, on utilise un navigateur

<http://127,0,0,1/cgi-bin/test.py>



Exemple 2 cgi avec une base de données sans envoi de paramètres au CGI

/usr/lib/cgi-bin# apt-get install python3-mysqldb
Lecture des listes de paquets... Fait

```
mysql> select * from clients;
+----+-----+-----+-----+-----+
| id | prenom | nom  | email    | code | solde |
+----+-----+-----+-----+-----+
| 1  | Abdoulaye | Wade | wade@rtn.sn | 1111 | 2000000 |
| 2  | Macky    | Sall  | sall@rtn.sn | 2222 | 12000000 |
| 3  | Ali      | Bongo | bongo@rtn.sn | 3333 | 3000000 |
+----+-----+-----+-----+-----+
```

On cree dans le dossier /usr/lib/cgi-bin le fichier affich.py comme suit :

affich.py

```
#!/usr/bin/python3
```

```
import MySQLdb
```

```
print("Content-Type: text/html")
```

```
print()
```

```
print("<html><head><title>Comptes</title></head>")
```

```
print("<body>")
```

```
connection = MySQLdb.connect(user='bouki', passwd='passer123', db='banque')
```

```
cursor = connection.cursor()
```

```
cursor.execute("SELECT * from clients")
```

```
print('<table border="1px">')
```

```
print('<tr><td>Prenom</td><td>Nom</td><td>email</td><td>Code</td><td>Solde</td></tr>')
```

```
for row in cursor.fetchall():
```

```
    print("<tr><td>%s </td><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>" %  
(row[1],row[2],row[3],row[4],row[5]))
```

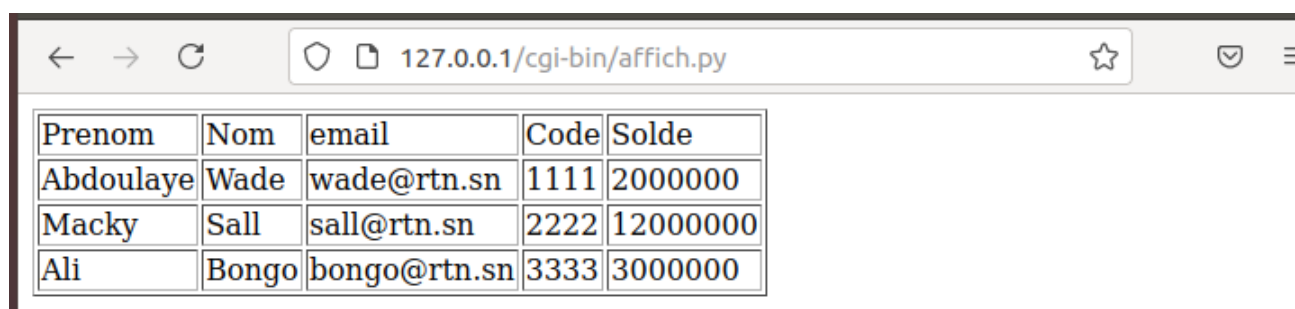
```
print("</table>")
```

```
print("</body></html>")
```

```
connection.close()
```

Testons avec un navigateur en saisissant :

<http://localhost/cgi-bin/affich.py>



Prenom	Nom	email	Code	Solde
Abdoulaye	Wade	wade@rtn.sn	1111	2000000
Macky	Sall	sall@rtn.sn	2222	12000000
Ali	Bongo	bongo@rtn.sn	3333	3000000

Donc on a crée de manière dynamique avec python un tableau contenant les présidents qui sont dans notre base de données banque plus précisément dans la table clients

Exemple 3 CGI recevant des données issues d'un formulaire

on cree un formulaire form1.html

form1.html

```

<!doctype html>
<html>
<head>
<title>demo cgi</title>
</head>
<body>
<form name="pyform" method="POST" action="/cgi-bin/crud.py">
Prenom: <input type="text" name="prenom" /><br>
Nom: <input type="text" name="nom" /><br>
Mail: <input type="text" name="email" /><br>
Code: <input type="text" name="code" /><br>
Solde:<input type="text" name="solde" /><br>
    <input type="submit" name="Submit" value="Valider" />
</form>
</body>
</html>

```

Creons maintenant notre script cgi

Mais pour pouvoir recuperer les les données du formulaire et gerer eventuellement des erreurs, il faut importer les modules cgi et cgitb

crud.py

```

#!/usr/bin/python3
# -*- coding: UTF-8 -*-
import cgi, cgitb
import mysql.connector
def database():
    global conn,cursor

conn=mysql.connector.connect(host="localhost",user="bouki",passwd="passer123",database="ban
que")
    cursor=conn.cursor()

def inserer(prenom,nom,email,code,solde):
    sql="insert into clients (prenom, nom,email,code,solde) values (%s, %s, %s, %s,%s)"
    database()
    val = (prenom, nom,email,code,solde)
    cursor.execute(sql, val)
    conn.commit()

print("Content-Type: text/html")
print()
print("<html><head><title>Comptes</title></head>")
print("<body>")

form = cgi.FieldStorage()
prenom = form.getvalue('prenom')
nom = form.getvalue('nom')
email = form.getvalue('email')

```

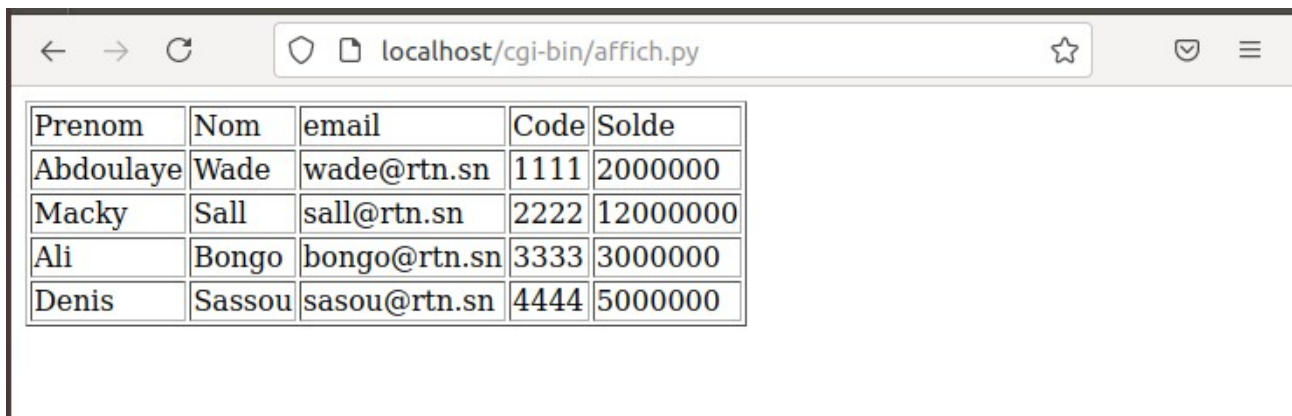


```

code = form.getvalue('code')
solde = form.getvalue('solde')
inserer(prenom,nom,email,code,solde)
print("Insertion reussie")
print("</body>")
print("</html>")

```

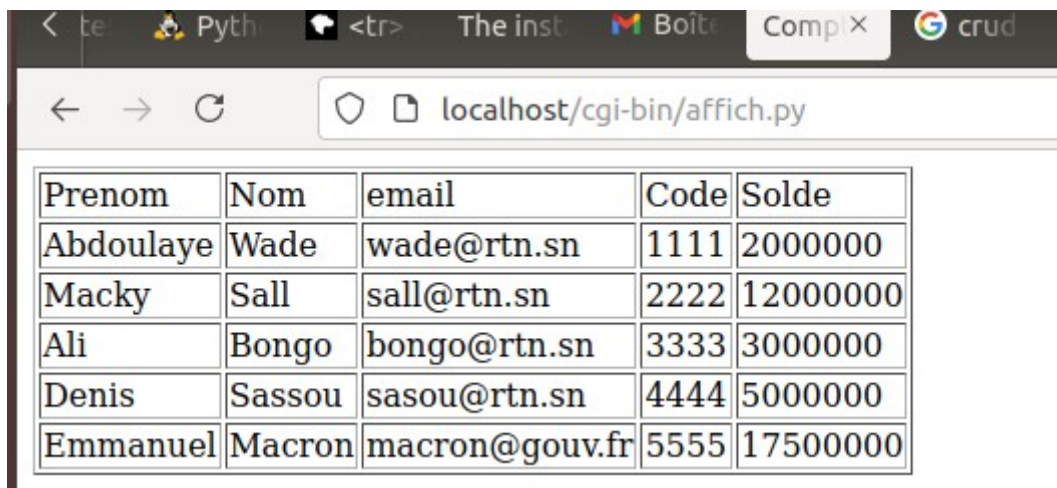
On arrive à travers le formulaire à insérer des données dans notre base banque par la methode POST



A screenshot of a web browser window displaying a table of bank account data. The browser's address bar shows 'localhost/cgi-bin/affich.py'. The table has five columns: 'Prenom', 'Nom', 'email', 'Code', and 'Solde'. It contains five rows of data.

Prenom	Nom	email	Code	Solde
Abdoulaye	Wade	wade@rtn.sn	1111	2000000
Macky	Sall	sall@rtn.sn	2222	12000000
Ali	Bongo	bongo@rtn.sn	3333	3000000
Denis	Sassou	sasou@rtn.sn	4444	5000000

Changeons dans le formulaire, la méthode POST par la methode GET et essayons d'insérer un autre président



A screenshot of a web browser window displaying a table of bank account data. The browser's address bar shows 'localhost/cgi-bin/affich.py'. The table has five columns: 'Prenom', 'Nom', 'email', 'Code', and 'Solde'. It contains six rows of data, including the new entry 'Emmanuel Macron'.

Prenom	Nom	email	Code	Solde
Abdoulaye	Wade	wade@rtn.sn	1111	2000000
Macky	Sall	sall@rtn.sn	2222	12000000
Ali	Bongo	bongo@rtn.sn	3333	3000000
Denis	Sassou	sasou@rtn.sn	4444	5000000
Emmanuel	Macron	macron@gouv.fr	5555	17500000

On constate que ça marche

Exercice

Écrire les scripts cgi permettant de :

- 1- Faire la mise à jour du solde d'un président en fournissant son id et son code et le montant à lui ajouter
- 2- supprimer un président en donnant son id

Correction Exercice

Exo 1

code du formulaire update

FormUpdate.html

```
<html>
<head>
<title>
Mon formulaire Update
</title>

</head>
<body>
<form method = "POST" action = "/cgi-bin/update.py" >
<table>
<tr><td>Id:</td><td><input type = "text" name = "id"></td> </tr>
<tr><td>Code:</td><td><input type = "text" name = "code"></td> </tr>
<tr><td>Montant:</td><td><input type = "text" name = "montant"></td> </tr>
</table>
<input type = "submit" value = "Valider"/>
</form>
</body>
</html>
```

le script cgi update.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
import cgi, cgitb
import mysql.connector
def database():
    global conn,cursor
```

```
conn=mysql.connector.connect(host="localhost",user="bouki",passwd="passer123",database="ban
que")
```

```
    cursor=conn.cursor()
```

```
def update(id,code,montant):
```

```
    sql="select code,solde from clients where id=%s and code=%s"
```

```

database()
val = (id,code)
cursor.execute(sql, val)
tab=cursor.fetchone()
if cursor.rowcount > 0:
    nouveaumontant = float(tab[1]) + montant
    print(nouveaumontant)
    sql1="update clients set solde=%s where id=%s"
    val1=(nouveaumontant,id)
    cursor.execute(sql1,val1)
    conn.commit()
    print ("Ajout effectue")
else:
    print("Info erronee")

```

```

print("Content-Type: text/html")
print()
print("<html><head><title>Comptes</title></head>")
print("<body>")

```

```

form = cgi.FieldStorage()
id = form.getvalue('id')
id = int(id)
code = form.getvalue('code')
montant = form.getvalue('montant')
montant=float(montant)
update(id,code,montant)
print("</body>")
print("</html>")

```

Exercice 2

code de formDelete.html

```

<html>
<head>
<title>
Mon Formulaire delete
</title>

</head>
<body>
<form method = "POST" action = "/cgi-bin/delete.py" >
<table>
<tr><td>Id:</td><td><input type = "text" name = "id"></td> </tr>
<tr><td>Code:</td><td><input type = "text" name = "code"></td> </tr>
</table>
<input type = "submit" value = "Valider"/>
</form>
</body>

```

</html>

Le code de script delete.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
import cgi, cgiib
import mysql.connector
def database():
    global conn,cursor

conn=mysql.connector.connect(host="localhost",user="bouki",passwd="passer123",database="ban
que")
    cursor=conn.cursor()

def delete(id,code):
    sql="delete from clients where id=%s and code=%s"
    database()
    val = (id,code)
    cursor.execute(sql, val)
    conn.commit()
    print("Suppresion effectuee")

print("Content-Type: text/html")
print()
print("<html><head><title>Comptes</title></head>")
print("<body>")

form = cgi.FieldStorage()
id = form.getvalue('id')
id = int(id)
code = form.getvalue('code')
delete(id,code)
print("</body>")
```

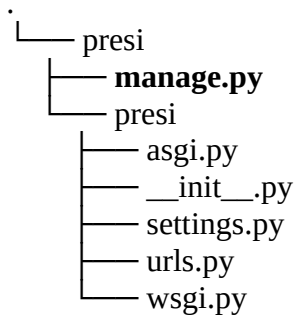
Partie II : Programmation web avec python et django

```
root@alain-VirtualBox:~# mkdir django
```

```
root@alain-VirtualBox:~# cd django/
```

```
root@alain-VirtualBox:~/django# django-admin startproject presi
```

```
root@alain-VirtualBox:~/django# tree
```



2 directories, 6 files

On constate qu'un repertoire presi est crée et comporte le fichier manager..py ainsi qu'un sous repertoire du meme nom est crée et contenant 5 fichiers

Ces fichiers sont :

- Le premier répertoire racine presi/ est un contenant pour votre projet. Son nom n'a pas d'importance pour Django ; vous pouvez le renommer comme vous voulez.
- `manage.py` : un utilitaire en ligne de commande qui vous permet d'interagir avec ce projet Django de différentes façons. Vous trouverez toutes les informations nécessaires sur `manage.py` dans [django-admin et manage.py](#).
- Le sous-répertoire presi/ correspond au paquet Python effectif de votre projet. C'est le nom du paquet Python que vous devrez utiliser pour importer ce qu'il contient (par ex. `presi.urls`).
- `presi/__init__.py` : un fichier vide qui indique à Python que ce répertoire doit être considéré comme un paquet. Si vous êtes débutant en Python,
- `presi/settings.py` : réglages et configuration de ce projet Django.
- `django/urls.py` : les déclarations des URL de ce projet Django, une sorte de « table des matières » de votre site Django.
- `presi/asgi.py` : un point d'entrée pour les serveurs Web compatibles aSGI pour déployer votre projet.
- `presi/wsgi.py` : un point d'entrée pour les serveurs Web compatibles WSGI pour déployer votre projet.

Le serveur de développement¶

Vérifions que votre projet Django fonctionne. Déplacez-vous dans le répertoire `mysite` si ce n'est pas déjà fait, et lancez les commandes suivantes :

Djuango

```
# pip3 install Django==3.2.5
```

on cree le projet banque

```
django-admin.py startproject banque
```

```
root@alain-VirtualBox:~/projDjango# tree
```

```
.
├── banque
│   ├── banque
│   │   ├── asgi.py
│   │   ├── __init__.py
│   │   ├── settings.py
│   │   ├── urls.py
│   │   └── wsgi.py
│   └── manage.py
```

Gérons notre application presi

```
root@alain-VirtualBox:~/projDjango# cd banque/
```

```
root@alain-VirtualBox:~/projDjango/banque# python3 manage.py startapp presi
```

```
tree
```

```
.
├── banque
│   ├── asgi.py
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-36.pyc
│   │   └── settings.cpython-36.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── manage.py
├── presi
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   └── models.py
```

```
├── tests.py
└── views.py
```

Lancement du serveur de projet

```
/projDjango/banque# python3 manage.py runserver
```

```
root@alain-VirtualBox:~/projDjango/banque# python3 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.

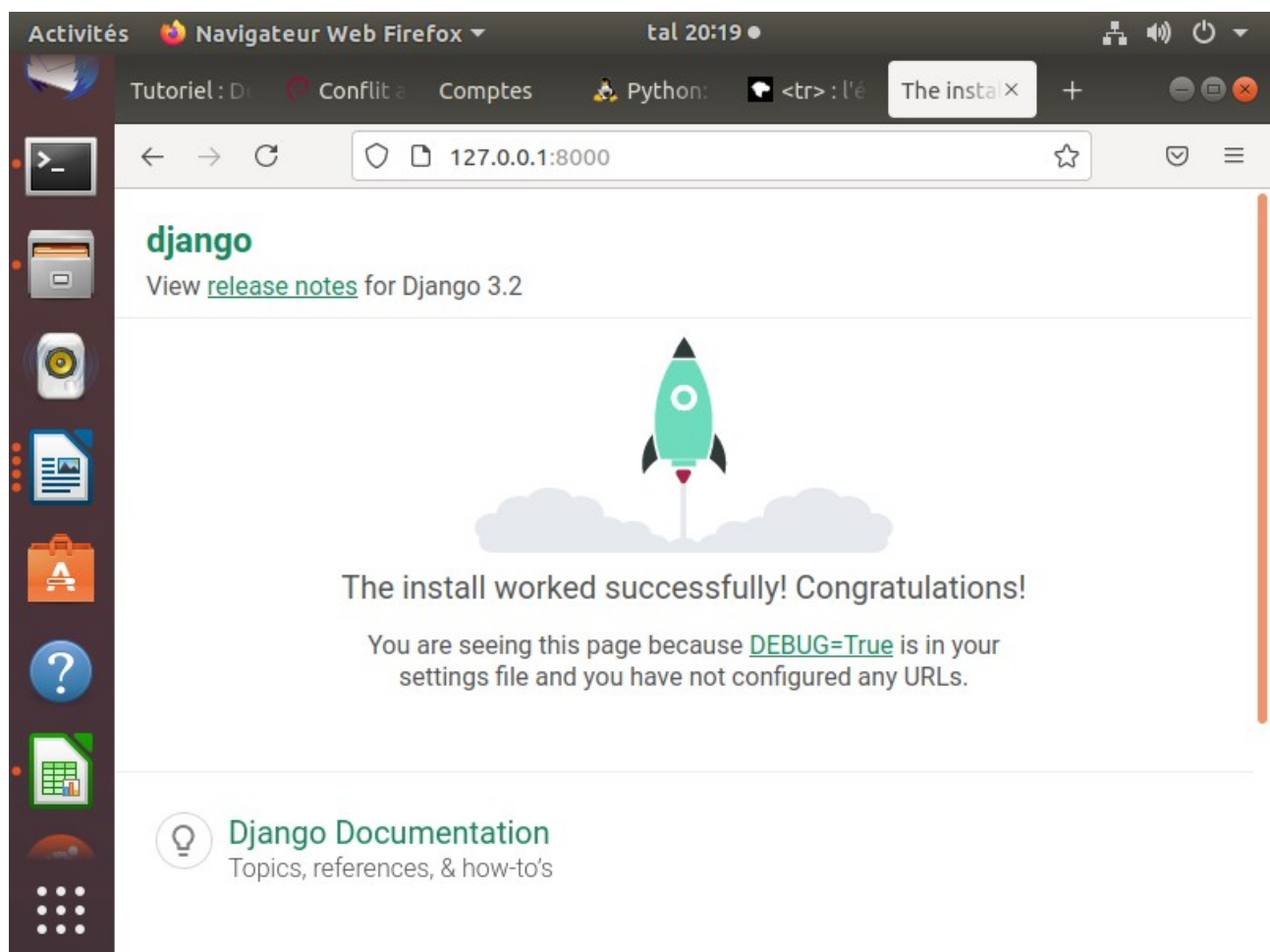
Run 'python manage.py migrate' to apply them.

July 06, 2021 - 20:18:52

Django version 3.2.5, using settings 'banque.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.



Fig

CRUD avec Django
Prof. Samuel Ouya

Objectif :

Développer une version Django de notre application comptepresident que nous avons déjà développée en :

- a- python desktop
- b- cgi apache2 pour le web

Projet presi

créons le dim11juillet devant contenir notre projet

```
root@post2-ESPRIMO-E500:/home/alain# mkdir dim11juillet
```

Allons dans le dossier dim11juillet

```
root@post2-ESPRIMO-E500:/home/alain# cd dim11juillet/
```

1- on crée le projet presi

```
root@post2-ESPRIMO-E500:/home/alain/dim11juillet# django-admin startproject presi
```

```
root@post2-ESPRIMO-E500:/home/alain/dim11juillet# cd presi/
```

```
(base) root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi#
```

2- on crée l'application comptepresi

```
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# python3 manage.py startapp comptepresi
```

3- la structure du projet

on tape la commande tree pour voir l'arborescence de projet presi

```
.
├── comptepresi
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
└── presi
    ├── asgi.py
    ├── __init__.py
    ├── __pycache__
    │   ├── __init__.cpython-38.pyc
    │   └── settings.cpython-38.pyc
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

4 directories, 15 files

4- paramètre de connexion au serveur de base de données

settings.py

Creation de la base djangodb

(base) root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# mysql -u root -p
Enter password:

```
mysql> create database djangodb;
```

dans le fichier presi/setting.py

```
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# vim presi/settings.py
```

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'djangodb',  
        'USER': 'root',  
        'PASSWORD': 'passer123',  
        'HOST': 'localhost',  
        'PORT': '3306'  
    }  
}
```

5- creation de modele pour la preparation de generation des tables **models.py**

```
from django.db import models  
# Create your models here.  
class President(models.Model):  
    prenom = models.CharField(max_length=30)  
    nom = models.CharField(max_length=30)  
    code = models.CharField(max_length=30)  
    solde = models.CharField(max_length=30)  
    class Meta:  
        db_table = "clients"
```

6- création de modèle de formulaire

forms.py

```
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# vim comptepresi/forms.py
```

```
from django import forms  
from comptepresi.models import President  
class PresidentForm(forms.ModelForm):  
    class Meta:  
        model = President  
        fields = "__all__"
```

7- création de fonctions de vue

views.py

```

from django.shortcuts import render, redirect

# Create your views here.
from comptepresi.forms import PresidentForm
from comptepresi.models import President
def emp(request):
    if request.method == "POST":
        form = PresidentForm(request.POST)
        if form.is_valid():
            try:
                form.save()
                return redirect('/show')
            except:
                pass
    else:
        form = PresidentForm()
        return render(request, 'index.html', {'form': form})
def show(request):
    presidents = President.objects.all()
    return render(request, 'show.html', {'presidents': presidents})

def edit(request, id):
    president = President.objects.get(id=id)
    return render(request, 'edit.html', {'president': president})
def update(request, id):
    president = President.objects.get(id=id)
    form = PresidentForm(request.POST, instance=president)
    if form.is_valid():
        form.save()
        return redirect("/show")
    return render(request, 'edit.html', {'president': president})

def destroy(request, id):
    president = President.objects.get(id=id)
    president.delete()
    return redirect("/show")

```

8- définition des routes du projet

urls.py

root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# vim presi/urls.py

```

from django.contrib import admin
from django.urls import path
from comptepresi import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('emp', views.emp),
    path('show', views.show),
    path('edit/<int:id>', views.edit),
    path('update/<int:id>', views.update),
    path('delete/<int:id>', views.destroy),

```

]

9- organisation des templates

Dans le repertoire comptepresi on crée le dossier templates

```
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi# mkdir templates
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi# cd templates/
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi/templates# vim
index.html
```

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<title>Index</title>  {{% load staticfiles %}}
<link rel="stylesheet" href="{% static 'css/style.css' %}" />
</head>

<body>
<form method="POST" class="post-form" action="/emp">
{% csrf_token %}
  <div class="container">
<br>

<div class="form-group row">
  <label class="col-sm-1 col-form-label"></label>
  <div class="col-sm-4">
    <h3>Enter Details</h3>
  </div>
</div>

<div class="form-group row">
  <label class="col-sm-2 col-form-label">President Code:</label>
  <div class="col-sm-4">
    {{ form.code }}
  </div>
</div>

<div class="form-group row">
  <label class="col-sm-2 col-form-label">President Prenom:</label>
  <div class="col-sm-4">
    {{ form.prenom }}
  </div>
</div>

<div class="form-group row">
  <label class="col-sm-2 col-form-label">President Nom:</label>
  <div class="col-sm-4">
    {{ form.nom }}
  </div>
</div>
```

```

<div class="form-group row">
  <label class="col-sm-2 col-form-label">President Solde:</label>
  <div class="col-sm-4">
    {{ form.solde }}
  </div>
</div>

<div class="form-group row">
  <label class="col-sm-1 col-form-label"></label>
  <div class="col-sm-4">
    <button type="submit" class="btn btn-primary">Submit</button>
  </div>
</div>
</div>
</form>
</body>
</html>

```

creons show.html

root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi/templates# vim show.html

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Liste Presidents</title>
{% load staticfiles %}
<link rel="stylesheet" href="{% static 'css/style.css' %}" />
</head>
dy>
<table class="table table-striped table-bordered table-sm">
<thead class="thead-dark">
<tr>
<th>Code President</th>
<th>Prenom President</th>
<th></th>
<th>Nom President</th>
<th>Solde President</th>
<th>Actions</th>
</tr>
</thead>

<tbody>

{% for president in presidents %}
<tr>
<td>{{ president.code }}</td>
<td>{{ president.prenom }}</td>
<td>{{ president.nom }}</td>
<td>{{ president.solde }}</td>
<td>
<a href="/edit/{{ president.id }}"><span
class="glyphicon glyphicon-pencil" >Edit</span></a>
<a href="/delete/{{ president.id }}">Delete</a>

```

```

                </td>
            </tr>
        {% endfor %}
    </tbody>
</table>

<br>
<br>
<center><a href="/emp" class="btn btn-primary">Add New Record</a></center>
</body>
</html>

```

edit.html

```

!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Index</title>
        {% load staticfiles %}
        <link rel="stylesheet" href="{% static 'css/style.css' %}" />
    </head>
    <body>
<form method="POST" class="post-form" action="/update/{{ employee.id }}">
    {% csrf_token %}
<div class="container">
<br>

    <div class="form-group row">
        <label class="col-sm-1 col-form-label"></label>
        <div class="col-sm-4">
            <h3>Update Details</h3>
        </div>
    </div>
</div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Employee Id:</label>
        <div class="col-sm-4">
            <input type="text" name="eid" id="id_eid" required maxlength="20"
value="{{ employee.eid }}" />
        </div>
    </div>

    <div class="form-group row">
<label class="col-sm-2 col-form-label">Employee Name:</label>
<div class="col-sm-4">
<input type="text" name="prenom" id="id_prenom" required maxlength="100"
value="{{ president.prenom }}" />
</div>
</div>

<div class="form-group row">

```

```

        <label class="col-sm-2 col-form-label">Employee Name:</label>
        <div class="col-sm-4">
            <input type="text" name="nom" id="id_nom" required maxlength="100"
value="{{ president.nom }}" />
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Employee Name:</label>
        <div class="col-sm-4">
            <input type="text" name="code" id="id_code" required maxlength="100"
value="{{ president.code }}" />
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Employee Name:</label>
        <div class="col-sm-4">
            <input type="text" name="solde" id="id_solde" required maxlength="100"
value="{{ president.solde }}" />
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-1 col-form-label"></label>
        <div class="col-sm-4">
            <button type="submit" class="btn btn-success">Update</button>
        </div>
    </div>
</div>
</body>
</html>

```

NB : il faut supprimer dans les 3 templates files en rouge dans stacticfiles

10- création de fichiers statiques css

```

root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi# mkdir static
(base) root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi# cd static/
(base) root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi/static# mkdir css
(base) root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi/static# cd css/
(base) root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi/comptepresi/static/css# vim
style.css
body {font:12px/1.4 Verdana,Arial; background:#eee; height:100%; margin:25px 0; padding:0}
h1 {font:24px Georgia,Verdana; margin:0}

```

```
h2 {font-size:12px; font-weight:normal; font-style:italic; margin:0 0 20px}
p {margin-top:0}
ul {margin:0; padding-left:20px}
```

```
#testdiv {width:600px; margin:0 auto; border:1px solid #ccc; padding:20px 25px; background:#fff}
```

```
#tinybox {position:absolute; display:none; padding:10px; background:#fff url(images/preload.gif)
no-repeat 50% 50%; border:10px solid #e3e3e3; z-index:2000}
#tinymask {position:absolute; display:none; top:0; left:0; height:100%; width:100%;
background:#000; z-index:1500}
#tinycontent {background:#fff}
```

```
.button {font:14px Georgia,Verdana; margin-bottom:10px; padding:8px 10px 9px; border:1px solid
#ccc; background:#eee; cursor:pointer}
.button:hover {border:1px solid #bbb; background:#e3e3e3}
```

11- structure finale du projet

root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# tree

```
.
├── comtepresi
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── static
│   │   └── css
│   │       └── style.css
│   ├── templates
│   │   ├── edit.html
│   │   ├── index.html
│   │   └── show.html
│   ├── tests.py
│   └── views.py
├── manage.py
└── presi
    ├── asgi.py
    ├── __init__.py
    ├── __pycache__
    │   ├── __init__.cpython-38.pyc
    │   └── settings.cpython-38.pyc
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

7 directories, 20 files

12- preparation de migration

on installe python-pymysql qui va fournir un client mysql pour python :
apt install python3-pymysql
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# apt install python3-pymysql

on ajoute le module de mysql dans setting .py

```
import pymysql
pymysql.install_as_MySQLdb()
```

root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# vim presi/settings.py
on ajoute l application comptepresi
dans le projet

settings.py

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

root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# python3 manage.py makemigrations
Migrations for 'comptepresi':
comptepresi/migrations/0001_initial.py
- Create model President

Generation de notre application avec toutes les tables

python3 manage.py migrate
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# python3 manage.py migrate

Operations to perform:

Apply all migrations: admin, auth, comptepresi, 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 comptepresi.0001_initial... OK
Applying sessions.0001_initial... OK

démarrage de notre application

```
python3 manage.py runserver  
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# python3 manage.py runserver  
Watching for file changes with StatReloader  
Performing system checks...
```

System check identified no issues (0 silenced).
July 11, 2021 - 18:36:20
Django version 3.2.5, using settings 'presi.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

on peut passer au test de notre application

Enter Details

President Code:
2222

President Prenom:
Macky

President Nom:
Sall

President Solde:
45000000

Submit

fig

Code	President Prenom	President Nom	President Solde	President Actions
1111	Abdoulaye	Wade	25000000	Edit Delete
2222	Macky	Sall	45000000	Edit Delete

[Add New Record](#)

fig

Verification dans la base de donnees

```
mysql> use djangodb
```

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

```
mysql> select * from client;
```

```
+----+-----+-----+-----+-----+
| id | prenom | nom | code | solde |
+----+-----+-----+-----+-----+
| 1 | Abdoulaye | Wade | 1111 | 25000000 |
+----+-----+-----+-----+-----+
1 row in set (0,00 sec)
```

```
mysql> select * from client;
```

```
+----+-----+-----+-----+-----+
| id | prenom | nom | code | solde |
+----+-----+-----+-----+-----+
| 1 | Abdoulaye | Wade | 1111 | 25000000 |
| 2 | Macky | Sall | 2222 | 45000000 |
+----+-----+-----+-----+-----+
2 rows in set (0,00 sec)
```

creation de compte admin

```
base) root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# python manage.py
```

```
createsuperuser
```

Username (leave blank to use 'root'): admin

Email address: admin@rtn.sn

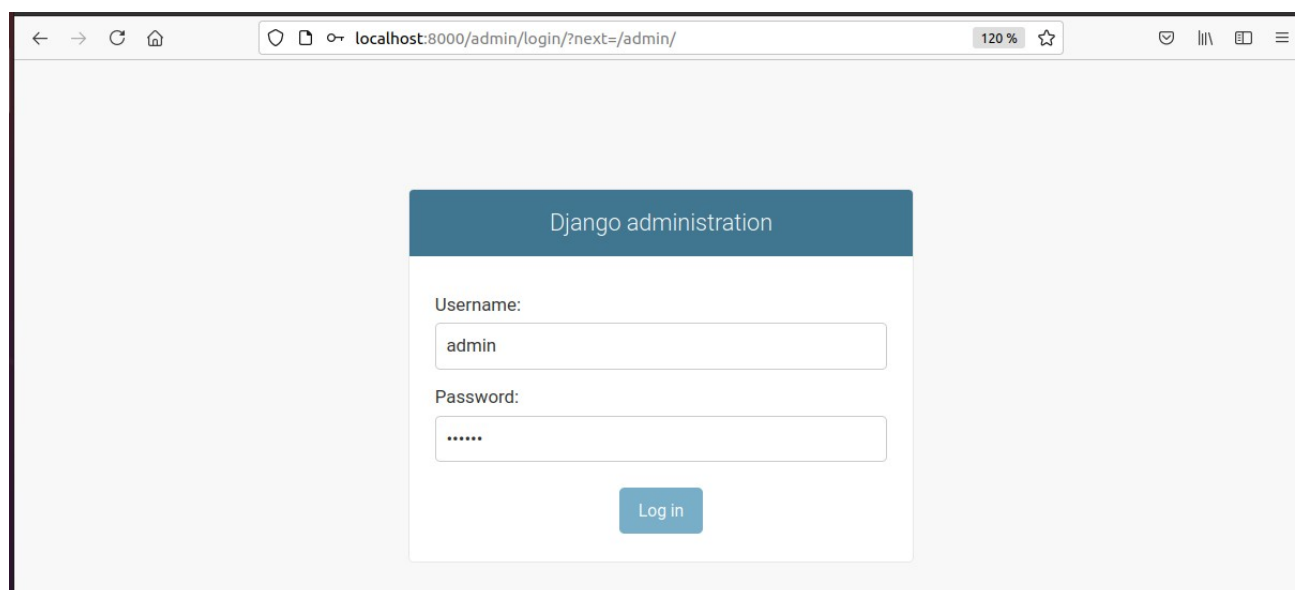
Password:

Password (again):

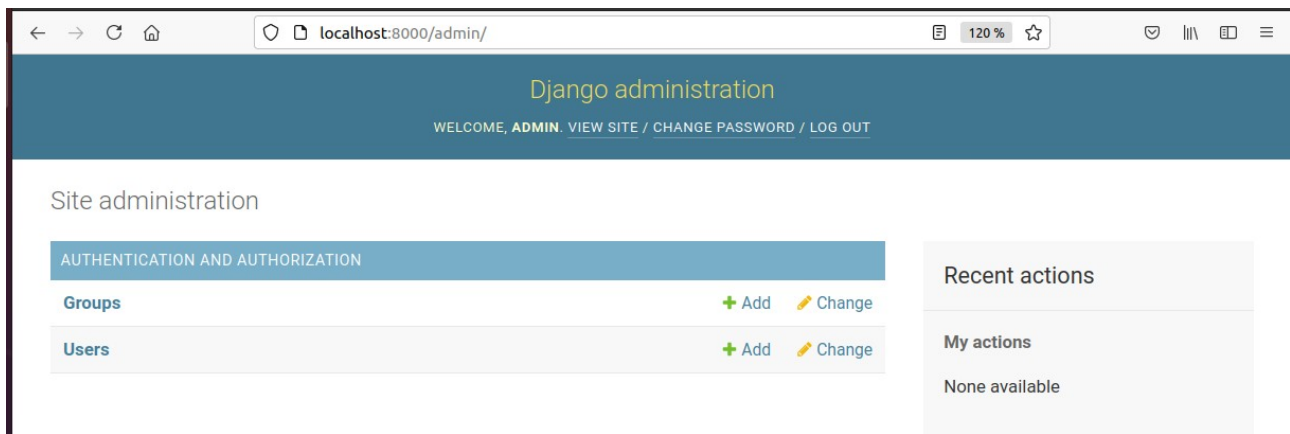
This password is too short. It must contain at least 8 characters.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.



Fig



fig

Rendons notre application visible sur l'interface
[admin](#)

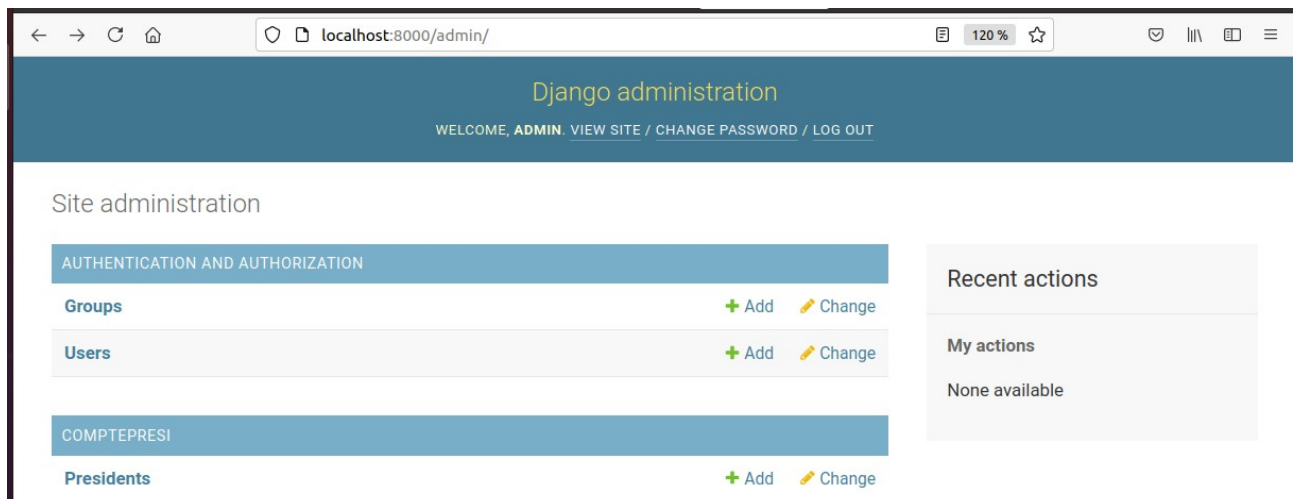
on edit le fichier admin.py

```
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# vim comptepresi/admin.py
```

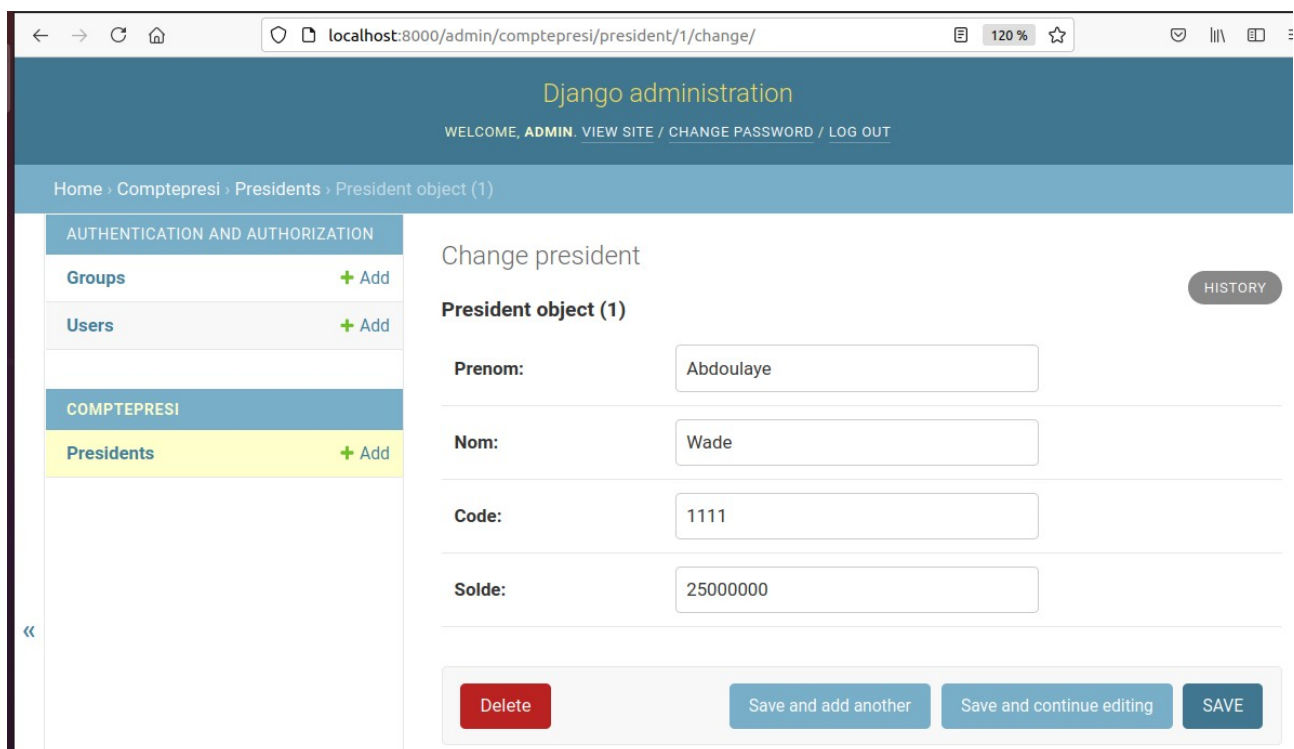
```
from django.contrib import admin
```

```
# Register your models here.
```

```
from .models import President
admin.site.register(President)
```



fig



fig

Modifions le solde du president Wade

Django administration

WELCOME, [ADMIN](#) / [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Comptepresi > Presidents > President object (1)

Change president

President object (1) HISTORY

Prenom:

Nom:

Code:

Solde:

Delete Save and add another Save and continue editing SAVE

puis on sauve

Django administration

WELCOME, [ADMIN](#) / [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Comptepresi > Presidents

Select president to change ADD PRESIDENT +

Action: Go 0 of 2 selected

☐ PRESIDENT

☒ President object (2)

☐ President object (1)

2 presidents

On verifie dans la base de donnees

```
root@post2-ESPRIMO-E500:/home/alain/dim11juillet/presi# mysql -u root -p
```

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 43

Server version: 8.0.25-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> use djangodb;
```

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

```
mysql> select * from clients;
```

ERROR 1146 (42S02): Table 'djangodb.clients' doesn't exist

```
mysql> select * from client;
```

```
+----+-----+-----+-----+-----+
| id | prenom | nom | code | solde |
+----+-----+-----+-----+-----+
| 1 | Abdoulaye | Wade | 1111 | 38000000 |
| 2 | Macky | Sall | 2222 | 45000000 |
+----+-----+-----+-----+-----+
2 rows in set (0,00 sec)
```

```
mysql>
```

On constate que le solde du president a bien change

Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home › Comptepresi › Presidents › Add president

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

COMPTEPRESI

Presidents [+ Add](#)

Add president

Prenom:

Nom:

Code:

Solde:

[Save and add another](#) [Save and continue editing](#) [SAVE](#)

En cliquant add

fig

Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home › Comptepresi › Presidents

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

COMPTEPRESI

Presidents [+ Add](#)

✓ The president "President object (3)" was added successfully.

Select president to change

[ADD PRESIDENT +](#)

Action: [Go](#) 0 of 3 selected

- ☐ PRESIDENT
- ☐ President object (3)
- ☐ President object (2)
- ☐ President object (1)

3 presidents

En verifiant dans la base de donnees
on constate que Idrissa Seck a ete cree

```
mysql> select * from client;
+----+-----+-----+-----+-----+
| id | prenom | nom | code | solde |
+----+-----+-----+-----+-----+
| 1 | Abdoulaye | Wade | 1111 | 38000000 |
| 2 | Macky | Sall | 2222 | 45000000 |
```



```
| 3 | Idrissa | Seck | 3333 | 17000000 |  
+---+-----+-----+-----+-----+  
3 rows in set (0,00 sec)
```

mysql>

Conclusion

Le framework Django simplifie la programmation web

La prochaine etape est de developper des API REST avec Django