FACULTAD DE TECNOLOGIA Y CS. APLICADAS – UNCa
Ingeniería en Informática

PROGRAMACION III

UNIDAD IV: ADMINISTRADOR DE DJANGO

DOCENTES DE CATEDRA:

- Mgtr. Cecilia E. Gallardo
- Esp. Marta Miranda

ADMINISTRADOR DE DJANGO

Django. Administración

Administrador de Django https://docs.djangoproject.com/en/4.0/ref/contrib/admin/

- El paquete de Django trae incorporado una interfaz de administración por defecto que permite crear objetos de nuestros modelos.
- Esto puede ser útil para el desarrollador para crear registros con el fin de probar la aplicación. También se puede usar como una interfaz para el Administrador del Sistema en producción.
- Para utilizar el Admin de Django, primero se debe crear un "Usuario administrador", mediante el comando:

```
python manage.py createsuperuser
```

  - *Se debe proporcionar un nombre de usuario, un email y contraseña.*
  - *Este comando creará en la B.D. un usuario con todos los permisos.*

Django. Administración

Administrador de Django

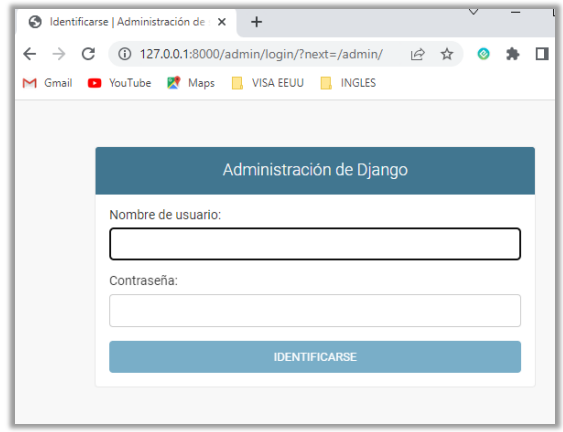
- Luego de crear un usuario Administrador, debemos indicar qué modelos deseamos que se puedan manipular mediante el Admin de Django.
- Para ello, dentro del archivo **"admin.py"** de la aplicación, importar y registrar los modelos correspondientes:

```
from django.contrib import admin

# Register your models here.
from apps.persona.models import Persona

admin.site.register(Persona)
            ó
@admin.register(Persona)
class PersonaAdmin(admin.ModelAdmin):
   list_display = ("id", "dni", "nombre_completo")
```

Django. Administración

Administrador de Django

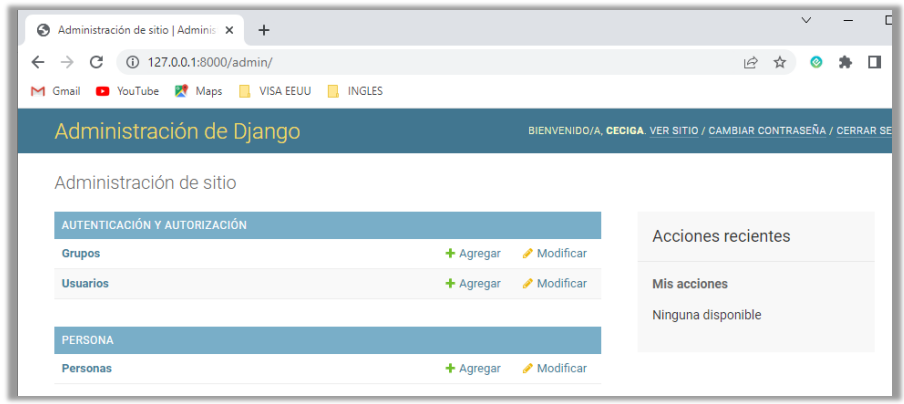
- Para acceder a la interfaz de administración de Django, debemos dirigirnos a la url: "/admin", para autenticarnos con el Usuario Administrador creado anteriormente:



Django. Administración

Administrador de Django

- Después de iniciar sesión, se podrá acceder a una página como la siguiente, donde puede crear, editar y eliminar objetos de los modelos registrados oportunamente en la aplicación de "Administración de Django" :



Django. Administración

Administrador de Django

- Ejemplo de Lista de objetos de un modelo, en este caso, de Personas:

LOGIN Y LOGOUT DE USUARIOS

Django. Usuarios

Login y Logout de Usuarios en la Aplicación Web

- Django tiene incorporadas funciones para realizar autenticación e inicio de sesion de usuarios que podemos usar para nuestras aplicaciones.
- Ejemplo de implementación:
  1. En nuestro proyecto Asistencias, crear una aplicación llamada “usuarios”
  2. En el archivo “urls.py” de la **app** agregar las rutas (y luego incluirlas en el archivo urls.py del **proyecto**):

```
app_name = 'usuarios'
urlpatterns = [
   path('', views.index, name="index"),
   path("login", views.login_view, name="login"),
   path("logout", views.logout_view, name="logout")
 ]
```

https://docs.djangoproject.com/en/4.1/topics/auth/default/#how-to-log-a-user-in

Django. Usuarios

Login y Logout de Usuarios en la Aplicación Web

- *Ejemplo de implementación:*
  3. Crear el template “templates/usuarios/login.html”:

```
{% extends “base/base.html" %}
{% block titulo %}Inicio Sesión{% endblock %}
{% block contenido %}
   {% if msj %}
      <div>{{ msj }}</div>
   {% endif %}
   <form action="{% url ‘usuarios:login' %}" method="post">
      {% csrf_token %}
      <input type="text" name="username" placeholder="Username">
      <input type="password" name="password" placeholder="Password">
      <input type="submit" value="Login">
   </form>
{% endblock %}
```

Django. Usuarios

Login y Logout de Usuarios en la Aplicación Web

- *Ejemplo de implementación:*
  4. Y crear el template “templates/usuarios/usuario.html”:

```
{% extends “base/base.html" %}
{% block titulo %}Inicio Sesión{% endblock %}
{% block contenido %}
    <h1>Bienvenido, {{ request.user.first_name }}</h1>
   <ul>
      <li>Username: {{ request.user.username }}</li>
      <li>Email: {{ request.user.email }}</li>
   </ul>

   <a href="{% url ‘usuarios:logout' %}">Log Out</a>
{% endblock %}
```


Django. Usuarios

Login y Logout de Usuarios en la Aplicación Web

- *Ejemplo de implementación:*
  5. En el archivo “usuarios/views.py” generar la vista “index” que controla si el usuario está autenticado:

```
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.contrib.auth import authenticate, login, logout
from django.urls import reverse


def index(request):
   if not request.user.is_authenticated:
      return HttpResponseRedirect(reverse(“usuarios:login"))
   return render(request, "usuarios/usuario.html")
```

Django. Usuarios

Login y Logout de Usuarios en la Aplicación Web

- *Ejemplo de implementación:*

6. En el mismo archivo “views.py” generar la vista de login:

```
def login_view(request):
    if request.method == "POST":
        username = request.POST["username"]
        password = request.POST["password"]
        user = authenticate(request, username=username, password=password)
        if user:
            login(request, user)
            return HttpResponseRedirect(reverse("usuarios:index"))
        else:
            return render(request, "usuarios/login.html", { “msj": “Credenciales
                    incorrectas" })
    return render(request, "usuarios/login.html")
```


Django. Usuarios

Login y Logout de Usuarios en la Aplicación Web

- *Ejemplo de implementación:*

7. También en el archivo “usuarios/views.py” generar la vista para desloguearse del sistema:

```
def logout_view(request):
    logout(request)
    return render(request, "usuarios/login.html", { "msj": “Deslogueado" })
```

Django. Usuarios

Login y Logout de Usuarios en la Aplicación Web

- *Ejemplo de implementación:*

Django. Restricción de acceso a Usuarios Autenticados

Control en Vistas:

- Para restringir el acceso a una vista en particular, solo para usuarios que hayan iniciado sesión en la aplicación, utilizar el siguiente decorador:

```
@login_required(login_url='usuarios:login')
def programa_create(request):
    …
```

https://docs.djangoproject.com/en/4.1/topics/auth/default/#the-login-required-decorator

Control en Templates:

```
{% if user.is_authenticated %}
  <p>Bienvenido, {{ user.username }}.</p>
{% else %}
  <p>Bienvenido, nuevo usuario. Por favor, inicie sesión.</p>
{% endif %}
```

https://docs.djangoproject.com/en/4.1/topics/auth/default/#users

Django. Restricción de acceso a Usuarios con Permisos

Permisos:

- Al tener instalada la aplicacion "django.contrib.auth" en la configuracion INSTALLED_APPS del archivo settings.py, esto asegura la creación de cuatro permisos por defecto para cada Modelo Django: **add, change, delete,** y **view**.
- Por ejemplo, suponiendo que se tiene una aplicacion llamada "foo" y un modelo llamado "Bar", para probar los permisos básicos se debe usar:

1. **AGREGAR REGISTRO**: user.has_perm('foo.add_bar')
2. **MODIFICAR REGISTRO**: user.has_perm('foo.change_bar')
3. **ELIMINAR REGISTRO**: user.has_perm('foo.delete_bar')
4. **VER REGISTROS**: user.has_perm('foo.view_bar')

Django. Restricción de acceso a Usuarios con Permisos

Control en Vistas:

- Para restringir el acceso a una vista en particular, no solo para usuarios autenticados, sino a aquellos que tengan un permiso específico:

```
@login_required(login_url='usuarios:login')
@permission_required('programa.add_programa',
raise_exception=True)
def programa_create(request):
    …
```

https://docs.djangoproject.com/en/4.1/topics/auth/default/#the-permission-required-decorator

Control en Templates:

```
# La variable "perms" contiene los permisos del usuario autenticado
{% if perms.programa.add_programa %}
   <a href='…'>Crear Programa</a>
{% endif %}
```

https://docs.djangoproject.com/en/4.1/topics/auth/default/#permissions

FACULTAD DE TECNOLOGIA Y CS. APLICADAS – UNCa
Ingeniería en Informática

GRACIAS POR TU ATENCIÓN!!

DOCENTES DE CATEDRA:

- **Mgtr. Cecilia E. Gallardo**
- **Esp. Marta Miranda**