# TUTORIALES

LOGIN Y REGISTRO

---

## OBJETIVO FINAL:

- Crear un modelo de usuario personalizado con campos como `email`, `nombre`, `rol`.
- Hacer el login y registro usando este modelo.
- Tener roles como `admin`, `cliente`, etc.
- Pisar (sobrescribir) el sistema de autenticación por defecto de Django.

## PASO 1: Crear el modelo de usuario personalizado

Edita `app/models.py` así:

```
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager,
PermissionsMixin
from django.db import models

class UsuarioManager(BaseUserManager):
    def create_user(self, email, nombre, rol, password=None):
        if not email:
            raise ValueError("El usuario debe tener un email")
        email = self.normalize_email(email)
        usuario = self.model(email=email, nombre=nombre, rol=rol)
        usuario.set_password(password)
        usuario.save(using=self._db)
        return usuario

    def create_superuser(self, email, nombre, rol='admin', password=None):
        usuario = self.create_user(email, nombre, rol, password)
        usuario.is_superuser = True
        usuario.is_staff = True
        usuario.save(using=self._db)
        return usuario
```

```python
class Usuario(AbstractBaseUser, PermissionsMixin):
    ROLES = (
        ('admin', 'Administrador'),
        ('cliente', 'Cliente'),
    )

    email = models.EmailField(unique=True)
    nombre = models.CharField(max_length=100)
    rol = models.CharField(max_length=20, choices=ROLES)
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)

    objects = UsuarioManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['nombre', 'rol']

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

## PASO 2: Configurar el nuevo modelo de usuario

En `settings.py`:

```python
AUTH_USER_MODEL = 'usuarios.Usuario'
```

Esto le dice a Django que tu modelo `Usuario` reemplaza al modelo `User` por defecto.

## PASO 3: Crear y aplicar migraciones

Ejecutamos en consola los comandos necesarios para aplicar nuestras migraciones a base de datos.

```
python manage.py makemigrations
python manage.py migrate
```

## PASO 4: Crear formularios de registro y login

Creamos los formularios pertinentes para el registro y el login con los campos que necesitemos, validaciones, estilo, etc.

### app/forms.py

```
from django import forms
from django.contrib.auth.forms import AuthenticationForm
from .models import Usuario

class RegistroForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)
    class Meta:
        model = Usuario
        fields = ['email', 'nombre', 'rol', 'password']

class LoginForm(AuthenticationForm):
    username = forms.EmailField(label="Correo")
```

## PASO 5: Crear vistas para login y registro

En nuestro views.py, creamos los métodos necesarios para hacer registrar un usuario y logearnos.

### app/views.py

```python
from django.contrib.auth import authenticate, login, logout
from django.shortcuts import render, redirect
from .forms import RegistroForm, LoginForm
from .models import Usuario

def registrar_usuario(request):
    if request.method == 'POST':
        form = RegistroForm(request.POST)
        if form.is_valid():
            usuario = form.save(commit=False)
            usuario.set_password(form.cleaned_data['password'])
            usuario.save()
            return redirect('login')
    else:
        form = RegistroForm()
    return render(request, 'usuarios/registro.html', {'form': form})


def login_usuario(request):
    if request.method == 'POST':
        form = LoginForm(request, data=request.POST)
        if form.is_valid():
            email = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            usuario = authenticate(request, email=email, password=password)
            if usuario is not None:
                login(request, usuario)
                return redirect('inicio')
    else:
        form = LoginForm()
    return render(request, 'usuarios/login.html', {'form': form})


def logout_usuario(request):
    logout(request)
    return redirect('login')
```

# TUTORIALES

LOGIN Y REGISTRO

---

**Explicación de métodos:**

## REGISTRO

| Línea | ¿Qué hace? |
|---|---|
| `if request.method == 'POST':` | Verifica si el formulario fue enviado. |
| `form = RegistroForm(request.POST)` | Crea una instancia del formulario con los datos enviados. |
| `if form.is_valid():` | Verifica que todos los campos cumplen validaciones (email único, rol válido, etc). |
| `usuario = form.save(commit=False)` | Crea el objeto usuario, pero aún no lo guarda en la BD. |
| `usuario.set_password(...)` | Encripta la contraseña antes de guardarla. Muy importante. |
| `usuario.save()` | Ahora sí, guarda el usuario en la BD. |
| `return redirect('login')` | Redirige al usuario al login. |
| `else: form = RegistroForm()` | Si no es POST, muestra el formulario vacío. |
| `render(...)` | Devuelve el HTML con el formulario cargado. |

## LOGIN

| Línea | ¿Qué hace? |
|---|---|
| `form = LoginForm(...)` | Usa `AuthenticationForm` (adaptado con email en vez de username). |
| `form.is_valid()` | Verifica si los campos están completos. |
| `email = form.cleaned_data.get('username')` | Aunque usamos `email`, Django lo llama "username". |
| `usuario = authenticate(...)` | Verifica si existe el usuario con ese email y contraseña. |
| `if usuario is not None:` | Si el usuario es válido... |
| `login(request, usuario)` | Django guarda la sesión. El usuario ahora está "logueado". |
| `redirect('inicio')` | Lo redirige a una vista de inicio (ajustá esto según tu app). |
| `render(...)` | Si GET o login falló, muestra de nuevo el formulario. |

## LOGOUT

| Línea | ¿Qué hace? |
|---|---|
| `logout(request)` | Elimina la sesión activa. El usuario ya no está autenticado. |
| `redirect('login')` | Lo lleva de nuevo al login. |

## PASO 6: Crear las URLs

Incluimos las urls enlazadas con nuestro métodos para cada una de las acciones que hemos configurado en views.py anteriormente.

### usuarios/urls.py

```
from django.urls import path
from views import *
urlpatterns = [
    path('registro/', registrar_usuario, name='registro'),
    path('login/', login_usuario, name='login'),
    path('logout/', logout_usuario, name='logout'),
]
```

## PASO 7: Crear las plantillas HTML

### usuarios/templates/usuarios/registro.html

```
<h2>Registro</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Registrarse</button>
</form>
```

### usuarios/templates/usuarios/login.html

```
<h2>Login</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Iniciar sesión</button>
</form>
```