

AE6 – Administración de usuarios y permisos en Django

Módulo 6: Desarrollo de aplicaciones web con Django

Aprendizaje esperado: Implementar módulo de administración de usuarios y permisos con el admin de Django

Clase práctica + activa



Objetivos de la clase

1 Comprender el funcionamiento del sitio administrativo de Django

Exploraremos las capacidades y ventajas del panel administrativo integrado de Django

2 Crear un superusuario y gestionar accesos

Aprenderemos a configurar credenciales de administrador y controlar el acceso al sistema

3 Manejar usuarios, grupos y permisos

Implementaremos sistemas de roles y permisos granulares para diferentes tipos de usuarios

4 Usar el modelo auth en código y en el admin

Integraremos la autenticación tanto en las vistas como en el panel administrativo

5 Aplicar seguridad y buenas prácticas

Implementaremos medidas de seguridad robustas para proteger nuestras aplicaciones

El sitio administrativo de Django

El admin de Django es una herramienta poderosa y lista para usar que proporciona una interfaz completa de administración sin necesidad de programación adicional.

Operaciones CRUD

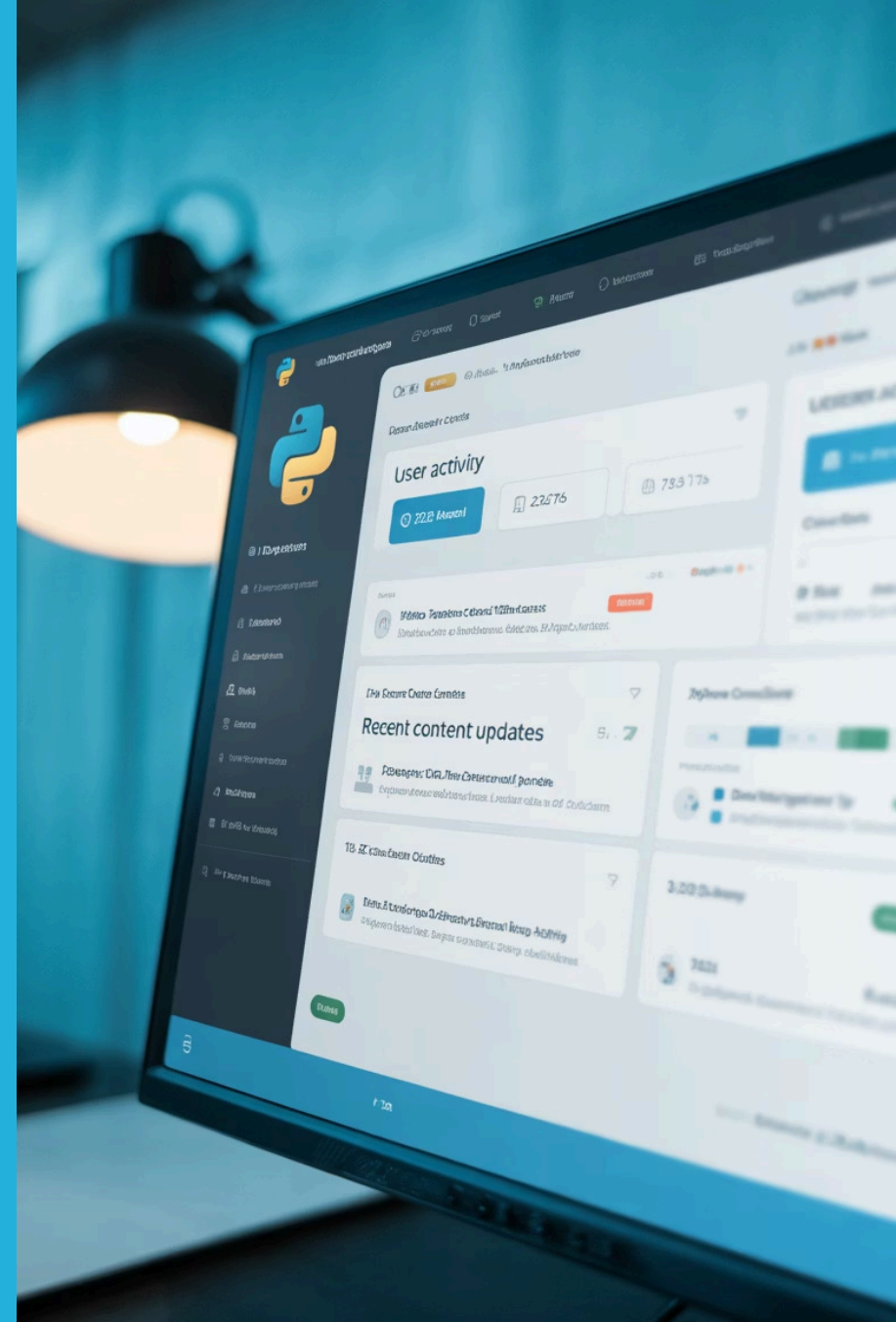
Crear, leer, actualizar y eliminar registros de forma intuitiva

Gestión de usuarios

Administrar usuarios, grupos y permisos desde una interfaz centralizada

Personalización avanzada

Customizar modelos y vistas sin programar interfaces desde cero





SECURITY

Ventajas del sitio administrativo



Rápido

No necesitas programar formularios de gestión desde cero. El admin se genera automáticamente basado en tus modelos.



Seguro

Autenticación integrada con sistemas de permisos robustos y protección contra ataques comunes.



Flexible

Personalizable con admin.py, permitiendo adaptar la interfaz a las necesidades específicas del proyecto.



Escalable

Permite gestionar muchos modelos en proyectos grandes sin perder rendimiento o usabilidad.


Primer paso: Iniciar servidor

Para acceder al panel administrativo, primero debemos iniciar el servidor de desarrollo de Django:

```
python manage.py runserver
```

Una vez iniciado el servidor, acceder a:

```
http://127.0.0.1:8000/admin/
```

  **Importante:** En producción se recomienda usar servidores como Apache o Nginx para mayor seguridad y rendimiento.

Actividad práctica (5 min): Cada estudiante debe ejecutar el servidor en su proyecto y verificar que el admin está funcionando correctamente.



Creando un superusuario

El superusuario tiene acceso completo al sistema administrativo. Para crearlo, ejecutamos:

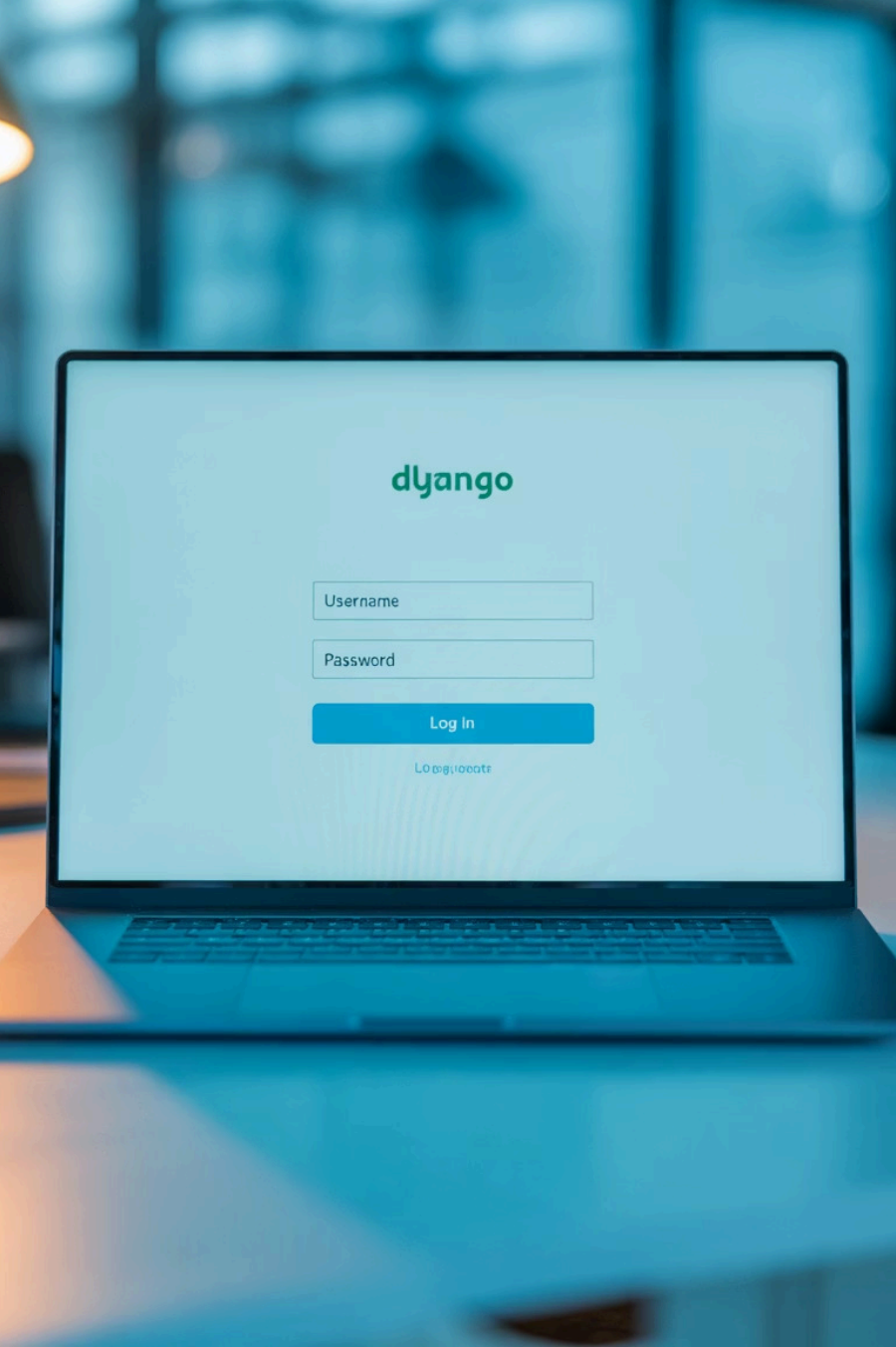
```
python manage.py createsuperuser
```

Django solicitará la siguiente información:

- Nombre de usuario (debe ser único)
- Correo electrónico (opcional pero recomendado)
- Contraseña segura (Django validará su fortaleza)

  **Seguridad crítica:** Las credenciales del superusuario deben guardarse en un lugar seguro y nunca compartirse. Son las llaves del reino.

Actividad práctica (5 min): Crear un superusuario y acceder exitosamente al panel admin.



Accediendo al admin

Una vez creado el superusuario, podemos acceder al panel administrativo:

01

Navegar a la URL del admin

URL por defecto:

`http://127.0.0.1:8000/admin/`

02

Iniciar sesión

Ingresar con las credenciales del superusuario creado

03

Explorar el panel inicial

Vista inicial muestra: "Usuarios", "Grupos" y todos los modelos registrados

El panel principal te dará acceso completo a la gestión de tu aplicación Django, desde usuarios hasta todos los modelos que hayas registrado.

Limitando el acceso

Para aumentar la seguridad del panel administrativo, podemos implementar varias capas de protección:

Middleware de autenticación

Configuración en `settings.py` para validar cada petición

```
MIDDLEWARE = [
```

```
'django.contrib.auth.middlew  
are.AuthenticationMiddlewar  
e',  
]
```

Restricción por IP

Limitando el acceso desde direcciones IP específicas usando Nginx, Apache o middleware personalizado

Roles y permisos

Implementando un sistema granular de permisos para acceso parcial según el rol del usuario



Registrando modelos en el admin

Para que un modelo sea visible en el panel administrativo, debe registrarse en el archivo admin.py:

```
from django.contrib import admin
from .models import Producto

admin.site.register(Producto)
```

Una vez registrado, el modelo aparecerá en el panel administrativo con funcionalidad CRUD completa automáticamente generada.



Crear registros

Agregar nuevos elementos al modelo



Visualizar datos

Ver listados y detalles de registros



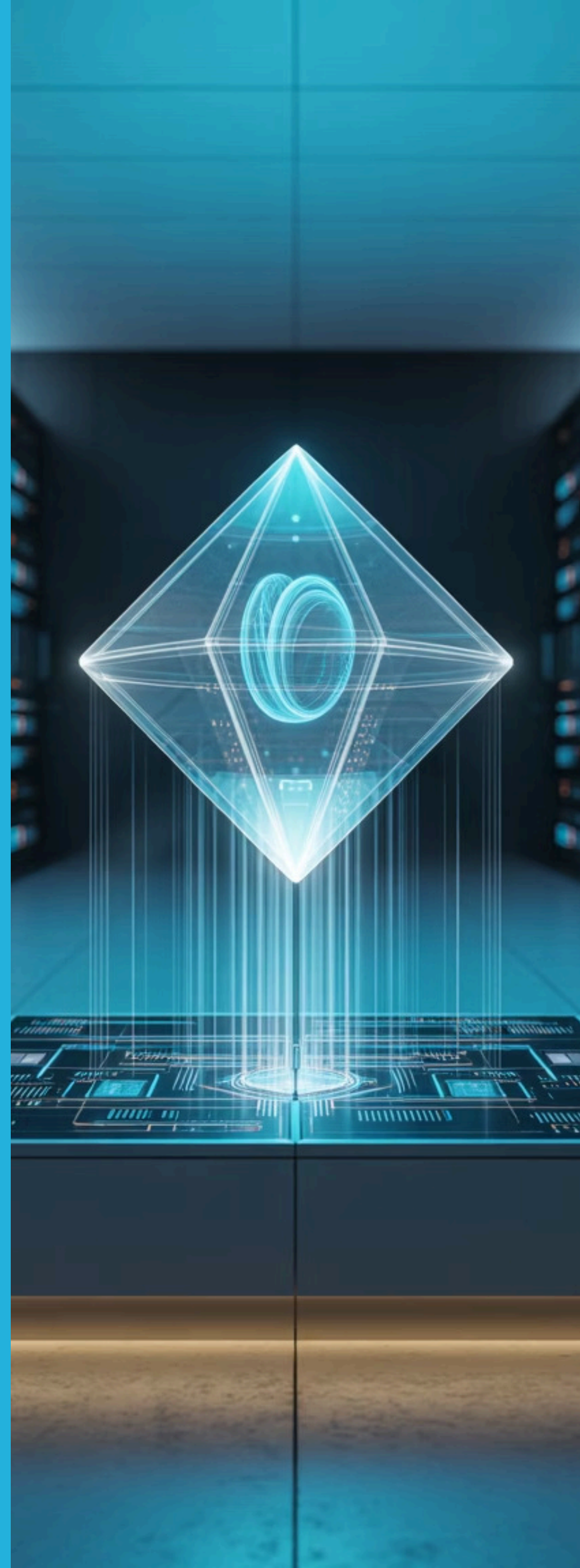
Editar información

Modificar registros existentes



Eliminar elementos

Borrar registros con confirmación



Personalizando el admin

Para mejorar la usabilidad del admin, podemos personalizarlo usando `ModelAdmin`:

```
class ProductoAdmin(admin.ModelAdmin):  
    list_display = ('nombre', 'precio')  
    search_fields = ('nombre',)  
    list_filter = ('categoria',)  
  
admin.site.register(Producto, ProductoAdmin)
```



`list_display`

Controla qué campos se muestran en la vista de lista, haciendo más fácil la navegación entre registros



`search_fields`

Habilita un cuadro de búsqueda que permite encontrar registros rápidamente por campos específicos



`list_filter`

Añade filtros laterales para segmentar los datos por categorías o valores específicos



Creando usuarios desde el admin

El panel administrativo permite crear y gestionar usuarios de forma intuitiva:

User Registron Form

Username

Password

Full Name

Submit

1

Acceder a la sección de usuarios

Navegar a Usuarios > Agregar usuario desde el panel principal

2

Completar información básica

Ingresar nombre de usuario, contraseña y datos opcionales como email y nombre completo

3

Configurar permisos

Asignar permisos específicos o agregar el usuario a grupos predefinidos

4

Guardar y activar

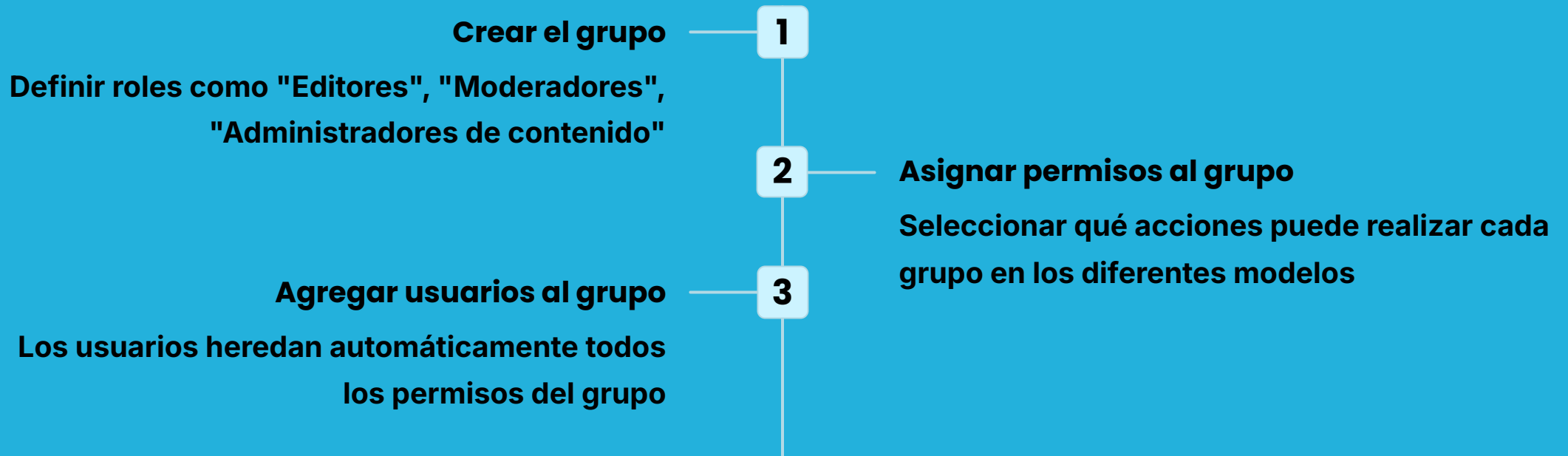
Finalizar la creación y verificar que el usuario puede acceder al sistema


Actividad práctica (10 min): Cada estudiante debe crear un usuario de prueba en el admin y verificar su funcionamiento.



Manejo de grupos

Los **grupos funcionan como roles** que simplifican la gestión de permisos. En lugar de asignar permisos individualmente, creamos grupos con permisos específicos.



 **Ventaja:** Si necesitas cambiar permisos, solo modificas el grupo y todos los usuarios se actualizan automáticamente.

Permisos por usuario

Django proporciona **cuatro permisos básicos** por cada modelo: **add, change, delete, view**. Estos pueden asignarse individualmente a usuarios específicos.

Ejemplo de asignación programática de permisos:

```
from django.contrib.auth.models import User, Permission

usuario = User.objects.get(username='alex')
permiso = Permission.objects.get(codename='add_producto')
usuario.user_permissions.add(permiso)
```

add_modelo

Crear nuevos registros del modelo

change_modelo

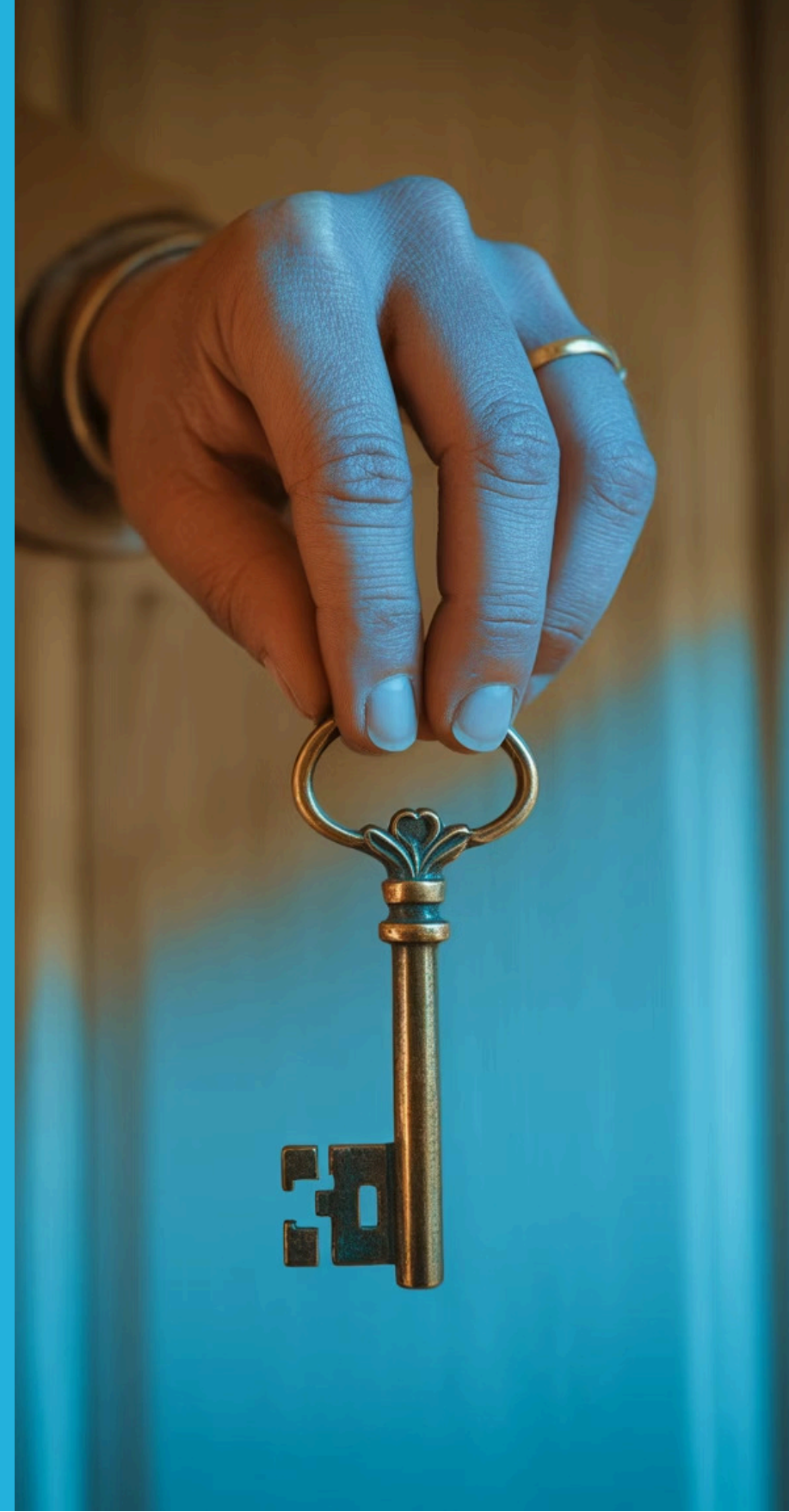
Modificar registros existentes

delete_modelo

Eliminar registros del modelo

view_modelo

Solo visualizar sin editar



Login

Email Address

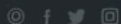
Password

Login

[Forgot password?](#)

[Create account](#)

Copyright password, a first session session



Uso del modelo Auth

El sistema de autenticación de Django integra perfectamente la gestión de usuarios con las vistas de la aplicación:

Gestión automática

Manejo de usuarios, contraseñas, sesiones y cookies de forma transparente

Grupos y permisos

Sistema granular para definir qué puede hacer cada tipo de usuario

Protección de vistas

Decoradores que restringen el acceso solo a usuarios autenticados

Ejemplo de vista protegida:

```
from django.contrib.auth.decorators import login_required
```

```
@login_required
```

```
def mi_vista(request):
```

```
    return render(request, 'mi_template.html')
```




Probando autenticación

Es fundamental probar que el sistema de permisos funcione correctamente antes de llevarlo a producción.

- 1 Crear usuario sin permisos**
Establecer un usuario básico sin acceso administrativo
- 2 Intentar acceder al admin**
Verificar que el sistema bloquea el acceso no autorizado
- 3 Observar mensajes de error**
Confirmar que los mensajes son claros y orientativos
- 4 Asignar permisos y probar**
Otorgar acceso gradual y verificar funcionamiento

Ejercicio práctico (10 min): Realizar este flujo completo para entender el comportamiento del sistema de autenticación.



Manejo de errores

Django maneja elegantemente los errores de autenticación, proporcionando una experiencia de usuario fluida y segura.

Errores comunes

- Contraseña incorrecta
- Usuario sin permisos suficientes
- Sesión expirada por inactividad
- Usuario desactivado

Respuesta de Django

- Redirección automática al login
- Mensajes de error claros y útiles
- Preservación de la URL destino
- Protección contra ataques de fuerza bruta

Seguridad en contraseñas

Django implementa **múltiples capas de seguridad** para proteger las contraseñas de los usuarios:

Hash seguro automático

Las contraseñas se almacenan usando algoritmos de hash robustos como PBKDF2, nunca en texto plano

Validación de fortaleza

Django valida automáticamente que las contraseñas cumplan criterios mínimos de seguridad

Salt único por usuario

Cada contraseña tiene un salt único que previene ataques de rainbow table

Recomendaciones de seguridad

- Usar contraseñas de al menos 12 caracteres
- Nunca compartir credenciales de administrador
- No otorgar permisos de admin innecesariamente
- Implementar rotación periódica de contraseñas



Caso práctico en equipo

Situación: Un sistema de biblioteca digital necesita implementar diferentes roles de usuario con permisos específicos.



Rol: Bibliotecario

- Agregar nuevos libros al catálogo
- Editar información de libros existentes
- Gestionar préstamos y devoluciones
- Generar reportes del sistema



Rol: Alumno

- Visualizar catálogo de libros disponibles
- Buscar libros por título, autor o categoría
- Ver historial personal de préstamos
- Sin permisos de edición del catálogo

Actividad grupal (20 min): Implementar el sistema completo con modelos, admin, grupos y usuarios de prueba.

Resumen de la clase

Hemos cubierto los aspectos fundamentales del sistema de administración y autenticación de Django:

1

Admin de Django

Herramienta potente, rápida y segura para gestión de datos sin programación adicional

2

Superusuario

Cuenta con control total del sistema, fundamental para la administración inicial

3

Usuarios y grupos

Sistema escalable para gestión granular de accesos y permisos por roles

4

Control de permisos

Capacidad de definir exactamente qué acciones puede realizar cada usuario

5

Buenas prácticas

Implementación de seguridad robusta con contraseñas seguras y roles apropiados

Cierre y reflexión

Para consolidar el aprendizaje, reflexionemos sobre lo que hemos construido hoy:

¿Qué ventajas da usar el admin en vez de programar paneles desde cero?

Considera el tiempo de desarrollo, la seguridad integrada y las funcionalidades automáticas

¿Cómo mejorarías la seguridad del admin en producción?

Piensa en autenticación de dos factores, restricciones de IP y monitoreo de accesos

¿Qué aprendiste creando usuarios y permisos hoy?

Reflexiona sobre la importancia de la planificación de roles y la gestión de accesos

👉 **Próxima clase:** Profundizaremos en autenticación personalizada con vistas de login y logout, sesiones avanzadas y middleware de seguridad.

