

A graphic on the left side of the slide. It features four overlapping horizontal bars in purple, orange, yellow, and blue. The text 'Agencia de Aprendizaje a lo largo de la vida' is written across these bars in white. The orange bar has a right-pointing arrow shape at its end.

Agencia de
Aprendizaje
a lo largo
de la vida

DJANGO

Reunión 28

Django: Auth - 1

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Reunión 27

Django: Admin - 2

- Modelos muchos a muchos
- Personalizando DjangoAdmin

Reunión 28

Django: Autenticación - 1

- Autenticación vs Autorización
- Instalación
- Usuarios y Grupos
- Is_authenticated
- Decorator login_required
- Limitar acceso en templates

Autenticación y Autorización

Que es autenticación? **SOY QUIEN DIGO SER**

- Es el acto de confirmar que algo (o alguien) **es quien dice ser**
- Tres factores básicos (sé, tengo, soy):
 - Sé: Usuario + clave, pin, etc.
 - Tengo: Token, App Mobile, etc.
 - Soy: Factores biométricos
- Autenticación integrada: LDAP.
- Autenticación federada: Delegar autentiación a entidad de confianza (google, facebook, etc)

Que es autorización? **QUE PUEDO HACER EN EL SISTEMA**

- La autorización implica otorgar o revocar un permiso a un usuario (ya autenticado) para hacer o ver algo en el sistema.
- seguridad basada en roles, implica definir un conjunto de permisos asociados a un grupo (rol) que luego un usuario puede tener asociado.

¿Qué es Django Authentication?

El sistema de autenticación de Django manejan tanto la **autenticación** como la **autorización**. En Django se utiliza autenticación para referirse a **ambas cosas**. Y consta de:

- Usuarios, Permisos, Grupos
- Sistema de hash de contraseña configurable.
- Formularios y herramientas de visualización para iniciar sesión o restringir contenido.
- Un sistema de backend configurable.

Configuración

settings.py

```
'django.contrib.auth'  
'django.contrib.contenttypes'
```

Framework de autenticación y sus modelos por defecto.

Sistema de tipo de contenidos de django, permite asociar permisos con modelos creados.

MIDDLEWARE

```
SessionMiddleware  
AuthenticationMiddleware
```

Gestiona sesiones a través de solicitudes (requests)

Asocia usuarios con solicitudes (requests) mediante sesiones (sesión)

Objeto User

- Solo existe una clase de usuario en el framework de autenticación, los superusuarios por ejemplo, solo son usuarios con un atributo en particular.

Atributos principales:

- username
- password
- email
- first_name
- last_name
- is_staff



Objeto User

```
>>> from django.contrib.auth.models import User
>>> user = User.objects.create_user('john', 'lennon@thebeatles.com', 'johnpassword')

# En este punto, el usuario es un objeto User ya fue grabado en la BD.
# Se puede continuar cambiando los atributos si se desea (es un model).
>>> user.last_name = 'Lennon'
>>> user.save()
```

superusuario -> is_staff = True

```
python manage.py createsuperuser --username=joe --email=joe@example.com
```

También pueden crearse desde el DjangoAdmin agregando la app auth que viene por defecto

Contraseñas

Django proporciona un sistema de almacenamiento de contraseñas flexible y usa PBKDF2 por defecto con un hash SHA256.

El atributo **password** de **User** es una cadena con el siguiente formato:

`<algorithm>$<iterations>$<salt>$<hash>`

settings.py predeterminado

```
PASSWORD_HASHERS = [  
    'django.contrib.auth.hashers.PBKDF2PasswordHasher',  
    'django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher',  
    'django.contrib.auth.hashers.Argon2PasswordHasher',  
    'django.contrib.auth.hashers.BCryptSHA256PasswordHasher',  
]
```

Esto significa que Django usará PBKDF2 por defecto para almacenar todas las contraseñas, pero admitirá verificar las contraseñas almacenadas con PBKDF2SHA1, argon2 y bcrypt.

Autenticando usuarios

```
user = authenticate(username='juan', password='secretor')
if user is not None:
    pass # Credenciales autenticadas por el backend
else:
    pass # Credenciales no autenticadas por el backend
```

A menos que esté escribiendo su propio sistema de autenticación, probablemente no lo use. Más bien, si está buscando una forma de iniciar sesión como usuario, use la vista basada en clases **LoginView**.

Permisos y Autorización

`django.contrib.auth`



Crea **permisos**
predeterminados por cada
modelo de Django definido en
cada aplicación en
INSTALLED_APPS



Crear
Modificar
Eliminar
ver

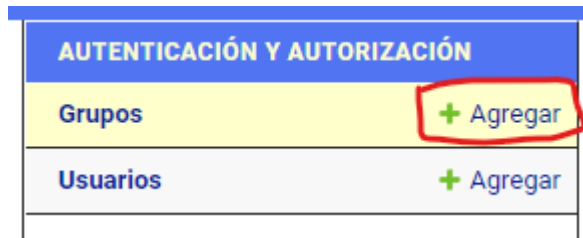


```
add: user.has_perm('cac.add_estudiante')
change:
user.has_perm('cac.change_estudiante')
delete:
user.has_perm('cac.delete_estudiante')
view: user.has_perm('cac.view_estudiante')
```

igualmente estos métodos rara vez son usados
explícitamente

Permisos y Autorización

Se recomienda la **creación** de **grupos** y **posibles permisos adicionales** necesarios desde el **admin**. **Previamente** habiendo hecho un **análisis** de los requerimientos de negocio y la **arquitectura de permisos** necesaria. (recordar crear siempre superusuario)



Agregar grupo

Nombre:

Permisos:

permisos disponibles ?

Q Filtro

- auth | usuario | Can change user
- auth | usuario | Can delete user
- auth | usuario | Can view user
- cac | comision | Can change comision**
- cac | comision | Can delete comision
- cac | curso | Can add curso
- cac | curso | Can change curso
- cac | curso | Can delete curso
- cac | docente | Can add docente
- cac | docente | Can change docente
- cac | docente | Can delete docente
- contenttypes | tipo de contenido | Can add content type
- contenttypes | tipo de contenido | Can change content type

Seleccionar todos/as ?

permisos seleccionados/as ?

- cac | comision | Can add comision
- cac | comision | Can view comision
- cac | curso | Can view curso
- cac | docente | Can view docente
- cac | estudiante | Can add estudiante
- cac | estudiante | Can change estudiante
- cac | estudiante | Can delete estudiante
- cac | estudiante | Can view estudiante
- cac | inscripcion | Can add inscripcion
- cac | inscripcion | Can change inscripcion
- cac | inscripcion | Can delete inscripcion
- cac | inscripcion | Can view inscripcion

Eliminar todos/as

Mantenga presionada "Control" ("Command" en una Mac) para seleccionar más de uno.

Permisos y Autorización

Administración de Codo a Codo

Administrador del Sitio

AUTENTICACIÓN Y AUTORIZACIÓN

Grupos

+ Agregar

✎ Modificar

Usuarios

+ Agregar

✎ Modificar

Permisos y Autorización

Agregar usuario

Primero introduzca un nombre de usuario y una contraseña. Luego podrá configurar opciones adicionales para el usuario.

Nombre de usuario:

Obligatorio. Longitud máxima de 150 caracteres. Solo puede estar formado por letras, números y los caracteres @/./+/-/_.

Contraseña:

Su contraseña no puede ser similar a otros componentes de su información personal.

Su contraseña debe contener por lo menos 8 caracteres.

Su contraseña no puede ser una contraseña usada muy comúnmente.

Su contraseña no puede estar formada exclusivamente por números.

Confirmación de contraseña:

Introduzca la misma contraseña nuevamente, para poder verificar la misma.

gestion

Algoritmo
“no
reversible”

Nombre de usuario:

gestion

Obligatorio. Longitud máxima de 150 caracteres. Solo puede estar formado por letras, números y los caracteres @/./+/-/_.

Contraseña:

algoritmo: pbkdf2_sha256 iteraciones: 260000 salt: IVmSfR***** hash: xf+sPG*****

El sistema no almacena las contraseñas originales por lo cual no es posible visualizar la contraseña de este usuario, pero puede modificarla usando [este formulario](#).

Información personal

Nombre:

Juan

Apellido:

Perez

Dirección de email:

juan.perez@gestion.com

Permisos

☒ Activo

Indica si el usuario debe ser tratado como un usuario activo. Desactive este campo en lugar de eliminar usuarios.

☐ Es staff

Indica si el usuario puede ingresar a este sitio de administración.

☐ Es superusuario

Indica que este usuario posee todos los permisos sin que sea necesario asignarle los mismos en forma explícita.

Grupos:

grupos disponibles ⓘ

Q

Filtro

grupos seleccionados/as ⓘ

inscripciones

No son lo mismo

También se
pueden asociar
permisos
individuales

Iniciar y cerrar sesión en un usuario

```
from django.contrib.auth import authenticate, login

def mi_Vista(request):
    username = request.POST['username']
    password = request.POST['password']
    user = authenticate(request, username=username, password=password)
    if user is not None:
        login(request, user)
        # Redireccionar a página de éxito.
        ...
    else:
        # Retornar mensaje de 'login inválido'.
        ...
```

Guarda el usuario en la sesión de Django

```
from django.contrib.auth import logout

def logout_view(request):
    logout(request)
    # Redireccionar a página de éxito
```

Hay vistas que hacen esto automáticamente

Limitar acceso a usuarios registrados en vistas

```
from django.conf import settings
from django.shortcuts import redirect

def mi_vista(request):
    if not request.user.is_authenticated:
        return redirect('%s?next=%s' % (settings.LOGIN_URL, request.path))
    # ...
```

Se suele redireccionar a una página de login o bien se puede mostrar un mensaje de error.

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

@login_required
def mi_vista(request):
    ...
```

Si el usuario no ha iniciado sesión redirige automáticamente a la página configurada en settings.LOGIN_URL (también hay parámetros opcionales)

Limitar acceso a usuarios registrados en vistas basadas en clases

Debe estar en la posición mas la izquierda
(reparar mixin de vistas basadas en clases)

```
class MiVista(LoginRequiredMixin, View):  
    login_url = '/login/'  
    redirect_field_name = 'redirect_to'  
    ...
```

Limitar acceso a vistas basadas según permisos

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

```
@permission_required('cac.add_estudiante')
```

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

```
...
```

Al igual que login_required, también
acepta parámetros opcionales

raise_exception lanza
PermissionDenied

```
from django.contrib.auth.mixins import PermissionRequiredMixin
```

```
class MiVista(PermissionRequiredMixin, View):
```

```
    permission_required = 'cac.add_estudiante'
```

```
    # 0 múltiples permisos
```

```
    permission_required = ('cac.add_estudiante', 'cac.eliminar_estudiante')
```

Limitar acceso a usuarios registrados en templates

Se encuentra dentro de RequestContext

```
{% if user.is_authenticated %}
    <p>Bienvenido\a, {{ user.username }}.</p>
{% else %}
    <button type="button" class="btn btn-outline-primary me-2">Iniciar Sesión</button>
    <button type="button" class="btn btn-dark">Registrarse</button>
{% endif %}
```

Limitar acceso a permisos en templates

Es un proxy booleano a User.has_module_perms()

```
{% if perms.cac %}  
    <p>Tenes permisos para hacer algo en la aplicación cac.</p>  
    {% if perms.cac.add_estudiante %}  
        <p>Podes agregar un estudiante!</p>  
    {% endif %}  
{% else %}  
    <p>No podes hacer nada en la aplicación cac.</p>  
{% endif %}
```

**No te olvides de completar la
asistencia y consultar dudas**

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

TODO EN EL AULA VIRTUAL