HINTS

CONSEJOS CONCEPTUALES

- Los permisos en Django se pueden establecer no solo por tipo de objeto, sino también por instancia de objeto específica. Al usar los métodos: `has_view_permission()`, `has_add_permission()`, `has_change_permission()` y `has_delete_permission()` proporcionados por la clase `ModelAdmin`, es posible personalizar los permisos para diferentes instancias de objetos del mismo tipo.

- El modelo User en Django tiene una relación Muchos a Muchos con el modelo Permissions y el modelo Groups.

- El método `get_all_permissions()` permite comprobar los permisos con los que cuenta un usuario: `user.get_all_permissions()`.

- Para comprobar cuáles son los permisos que tiene un usuario por los grupos a los que pertenece: `user.user.get_group_permissions()`.

- Los grupos son modelos Django que residen en el módulo `django.contrib.auth.models`, así que se puede hacer uso de la API de acceso a la base de datos para trabajar con los grupos a bajo nivel.

- Es posible escribir un mixin de verificación de roles y usarlo en las vistas, si se tiene una lógica de verificación de roles personalizada para restringir el acceso a una ruta.

- Un `mixin` es un tipo especial de herencia múltiple. Tienen dos principales usos:
    - Proporcionar muchas características opcionales para una clase.
    - Usar una función en particular en muchas clases diferentes.

- Los permisos son instancias de `django.contrib.auth.Permission`. Este modelo contiene tres campos: nombre, nombre en clave y `content_type`, donde `content_type` indica a qué modelo pertenece este permiso.

  El código para crear permisos con el modelo `Permission`, es el siguiente:

```
from django.contrib.auth.models import Permission,ContentType
from .models import Car
def add_permission(request):
  content_type = ContentType.objects.get_for_model(Car)
  permission = Permission.objects.create(name='Se pueden probar',
codena-me='test_car', content_type=content_type)
  return HttpResponse(¡El permiso se creó satisfactoriamente! '
```

- En caso de no utilizar la plantilla `registration/login.html`, se puede configurar el parámetro `template_name` como argumento extra en el método `as_view` en `URLconf`. Ejemplo:

```
path('accounts/login/', auth_views.LoginView.as_view
(template_name='myapp/login.html')),
```

- Si se habilita `redirect_authenticated_user`, otros sitios web podrán determinar si sus visitantes están autenticados en su sitio solicitando URL de redireccionamiento a archivos de imagen en su sitio web. Para evitar esta fuga de información de "huella digital de las redes sociales", aloje todas las imágenes y su favicon en un dominio separado.

- Si ninguna de las condiciones para el envío del email en la vista `passwordResetView` se cumple, el email no se enviará, pero el usuario no recibirá ningún error. En caso de requerir mostrar alguno, se puede crear una subclase de `PasswordResetForm` y hacer uso del atributo `form_class`.

- Si no desea utilizar las vistas integradas, pero quiere la comodidad de no tener que escribir formularios para esta funcionalidad, el sistema de autenticación proporciona varios formularios integrados ubicados en: `django.contrib.auth.forms`.