EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: PERMISOS EN DJANGO, PERMISOS DEL MODELO BOARDS.
- EXERCISE 2: CREAR PERMISOS PERSONALIZADOS EN DJANGO.
- EXERCISE 3: ASIGNAR PERMISOS AL REGISTRAR UN USUARIO EN FORMULARIO DE REGISTRO.
- EXERCISE 4: USANDO EL LOGINREQUIREDMIXIN Y EL PERMISSIONREQUIREDMIXIN.

EXERCISE 1: PERMISOS EN DJANGO, PERMISOS DEL MODELO BOARDS.

El objetivo del presente ejercicio es plantear una guía paso a paso para entender cómo se asignan y manipulan los permisos y autorizaciones en las vistas a los usuarios en el Framework Django.

Requisitos previos:

- Tener conocimiento de una terminal o línea de comando e instalación de paquetes de software en el sistema operativo, tanto en Windows 10 como en Linux (en este caso Linux Ubuntu).
- En caso de Windows, se debe tener instalado el entorno virtual explicado en el Exercises del CUE03 y la configuración del proyecto en el CUE04.
- Tener previamente instalado la versión de Python 3 y el entorno virtual con (virtualenvwrapper).
- Hacer uso de la herramienta Visual Studio Code.

PERMISOS EN DJANGO, PERMISOS DEL MODELO BOARDS

Cada vez que creas un modelo y corres las migraciones se crean automáticamente 4 permisos (add, edit, delete y view) en `django.contrib.auth` para ese objeto. Por otro lado, el modelo usuario tiene una relación `ManyToMany` (Muchos a Muchos) con el modelo Permissions (que guarda los permisos anteriores) y el modelo Groups. Por lo que ya de primeras contamos con una relación entre usuarios, grupos y permisos que podemos aprovechar.

En nuestro caso, tenemos el modelo creado `BoardsModel` en el archivo `model.py`.

BOARDS/MODEL.PY

```
1  from django.db import models
2  from django.conf import settings
3
4  # Create your models here.
5
6  class BoardsModel(models.Model):
7      # Campos del modelo
8      titulo = models.CharField(max_length = 200)
9      descripcion = models.TextField()
10     modificado = models.DateTimeField(auto_now_add = True)
11
12     def __str__(self):
13         return self.titulo
```

Vamos a crear un nuevo usuario y verificar los permisos que se crean. Se trabajará con el interpretador de Django (shell). En la terminal:

```
1 $ python manage.py shell
2 Python 3.8.3 (default, Jul  2 2020, 16:21:59)
3 [GCC 7.3.0] on linux
4 Type "help", "copyright", "credits" or "license" for more information.
5 (InteractiveConsole)
6 >>>
```

Procedemos a importar las librerías para la creación de usuarios y gestión de permisos.

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

Creamos el usuario:

```
1 >>> usuario = User.objects.create_user('juanperez',
2 'juanperez@correo.com', '123456')
```

Consultamos los permisos que posee el usuario:

```
>>> usuario.get_all_permissions()
```

Y también los permisos creados por defecto del modelo BoardsModel, importamos el modelo y el `ContentType`:

```
>>> from .models import BoardsModel
>>> from django.contrib.contenttypes.models import ContentType
    >>> from django.contrib.auth.models import Permission
```

Obtenemos los objetos `Content_Type` del modelo:

```
>>> content_type = ContentType.objects.get_for_model(BoardsModel)
```

Obtenemos los permisos que podemos asignar al modelo:

```
>>> boards_permissions =
Permission.objects.filter(content_type=content_type)
```

Imprimimos por pantalla los permisos del modelo:

```
>>> [p.codename for p in boards_permissions]
['add_boardsmodel', 'change_boardsmodel', 'delete_boardsmodel',
'view_boardsmodel']
```

Procedemos a asignar el permiso `add_boardsmodel` al usuario creado previamente (juanperez), pero para ello debemos verificamos si lo tiene asignado.

```
>>> usuario.has_perm('boards.add_boardsmodel')
False
```

La forma de asignar un permiso a un usuario es: `usuario.user_permissions.add(permiso)`, pero el objeto permiso, se debe recuperar del modelo. Para ello colocamos:

```
>>> add_permission =
Permission.objects.get(content_type=content_type,
codename='crear_boardsmodel')
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "/home/luispc/.virtualenvs/projects_django/lib/python3.8/site-
packages/django/db/models/manager.py", line 85, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
  File "/home/luispc/.virtualenvs/projects_django/lib/python3.8/site-
packages/django/db/models/query.py", line 496, in get
    raise self.model.DoesNotExist(
django.contrib.auth.models.Permission.DoesNotExist: Permission
matching query does not exist.
```

Observamos que nos da un error, ya que el permiso `crear_boardsmodel` no se encuentra creado en el modelo `BoardsModel`. Solo tenemos los que se crean por defecto: add, view, edit y delete. Adecuamos con el permiso `add_boardsmodel` de la siguiente manera:

```
>>> add_permission = Permission.objects.get(content_type=content_type,
codename='add_boardsmodel')
```

Agregamos el permiso al usuario:

```
>>> usuario.user_permissions.add(add_permission)
```

Verificamos cuáles permisos tiene:

```
>>> usuario.get_all_permissions()
set()
```

Nos muestra que tiene permisos vacíos, debemos nuevamente consultar el objeto usuario.

```
>>> usuario = User.objects.get(username='juanperez')
>>> usuario.get_all_permissions()
{'boards.add_boardsmodel'}
```

Procedemos a asignar el permiso de ver del modelo `BoardsModel` al usuario juanperez.

```
>>> view_permission =
Permission.objects.get(content_type=content_type,
codename='view_boardsmode')
>>> usuario.user_permissions.add(view_permission)
>>> usuario = User.objects.get(username='juanperez')
>>> usuario.get_all_permissions()
{'boards.view_boardsmodel', 'boards.add_boardsmodel'}
```

Podemos consultar si un usuario posee determinado permiso; por ejemplo, el de eliminar y agregar:

```
>>> usuario.has_perm('boards.delete_boardsmodel')
False
>>> usuario.has_perm('boards.add_boardsmodel')
True
```

EXERCISE 2: CREAR PERMISOS PERSONALIZADOS EN DJANGO.

Existen dos maneras de crear permisos personalizados dentro de Django, directamente y dentro de la clase meta del modelo. Construyendo el permiso directamente:

Importamos las librerías:

```
>>> from boards.models import BoardsModel
>>> from django.contrib.auth.models import Permission
>>> from django.contrib.contenttypes.models import ContentType
>>> from django.contrib.auth.models import User
```

Creamos el nuevo permiso al modelo `BoardsModel`:

```
crear_permiso =Permission.objects.create(
...      codename='crear_boardsmodel',
...      name='Crear BoardsModel',
...      content_type=content_type,
... )
```

Consultamos el permiso creado:

```
>>> boards_permissions =
Permission.objects.filter(content_type=content_type)
```

```
>>> [p.codename for p in boards_permissions]
```

Asignamos el nuevo permiso al usuario:

```
>>> crear_permission =
Permission.objects.get(content_type=content_type,
codename='crear_boardsmodel')
```

Agregamos el permiso al usuario:

```
>>> usuario.user_permissions.add(crear_permission)
```

Consultamos los permisos:

```
>>> usuario.get_all_permissions()
{'boards.view_boardsmodel', 'boards.add_boardsmodel'}
>>> usuario = User.objects.get(username='juanperez')
>>> usuario.get_all_permissions()
{'boards.crear_boardsmodel', 'boards.view_boardsmodel',
'boards.add_boardsmodel'}
```

CREADO EL PERMISO POR MEDIO DE LA CLASE META EN EL MODELO

Editamos y creamos el permiso en el modelo, en la clase `Meta`:

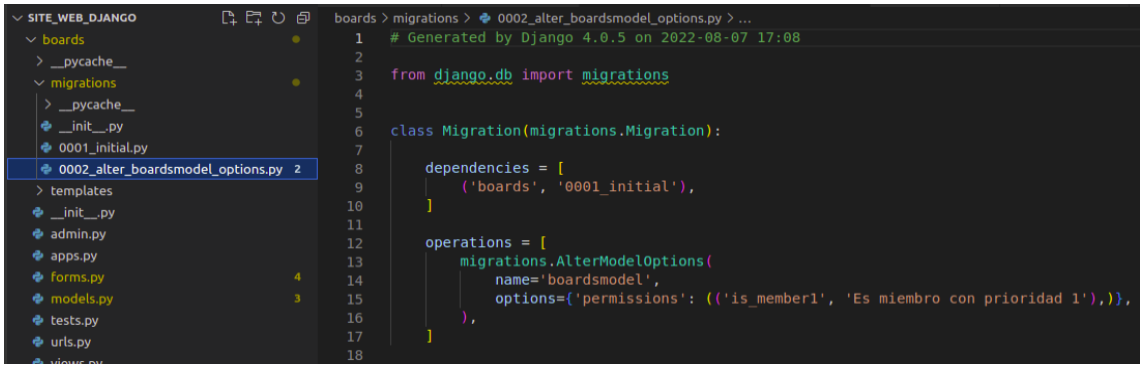
```
from django.db import models
from django.conf import settings

# Create your models here.

class BoardsModel(models.Model):
    # Campos del modelo
    titulo = models.CharField(max_length = 200)
    descripcion = models.TextField()
    modificado = models.DateTimeField(auto_now_add = True)

    class Meta:
        permissions = (
            ("es_miembro_1", "Es miembro con prioridad 1"),
```

```
        )

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

Realizamos las migraciones correspondientes, y observamos:

```
$ python manage.py makemigrations
Migrations for 'boards':
  boards/migrations/0002_alter_boardsmodel_options.py
    - Change Meta options on boardsmodel

$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, boards, contenttypes, sessions
Running migrations:
  Applying boards.0002_alter_boardsmodel_options... OK
```

Revisamos que se han creado las migraciones:



Revisamos en la consola de interprete de Django el permiso creado:

```
>>> boards_permissions =
Permission.objects.filter(content_type=content_type)
>>> [p.codename for p in boards_permissions]
['add_boardsmodel', 'change_boardsmodel', 'crear_boardsmodel',
'delete_boardsmodel', 'es_miembro_1', 'view_boardsmodel']
>>>
```

EXERCISE 3: ASIGNAR PERMISOS AL REGISTRAR UN USUARIO EN FORMULARIO DE REGISTRO.

Para proceder con esta práctica, debemos modificar la lógica de la vista que genera el formulario de registro de usuarios, para ello editamos la vista. Agregamos las librerías necesarias:

```
.....
# Para gestión de permisos
from django.contrib.auth.models import Permission
from django.contrib.contenttypes.models import ContentType
from .models import BoardsModel
```

Editamos el método `registro_view`:

```
def registro_view(request):
    if request.method == "POST":
        form = RegistroUsuarioForm(request.POST)

        if form.is_valid():
            # obtenemos el content type del modelo

            content_type =
ContentType.objects.get_for_model(BoardsModel)

            # obtenemos el permiso a asignar

            es_miembro_1 = Permission.objects.get(
            codename='es_miembro_1',
            content_type=content_type)
            user = form.save()

            # Agregamos el permiso al usuario el momento de
 registrarse
            user.user_permissions.add(es_miembro_1)
            login(request, user)
            messages.error(request, "Registrado Satisfactoriamente.")
            return HttpResponseRedirect ('/menu')
        messages.error(request, "Registro invalido. Algunos datos son
incorrectos.")

    form = RegistroUsuarioForm()
    return render (request= request,
template_name="registration/registro.html",
context={"register_form":form})
```

Registramos un nuevo usuario en: http://localhost:8000/registro/. Luego del registro, verificamos qué permisos tiene asignados:

```
>>> usuario = User.objects.get(username='usuariotest')
>>> usuario.get_all_permissions()
{'boards.es_miembro_1'}
>>>
```

EXERCISE 4: USANDO EL LOGINREQUIREDMIXIN Y EL PERMISSIONREQUIREDMIXIN.

Para esta práctica vamos a validar la página de inicio, que en este caso nos muestra el sencillo mensaje de: Saludos, Hola Mundo.

Modificamos la vista y agregamos:

BOARDS/VIEW.PY

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

Y en la clase `IndexPageView` modificamos del mismo archivo:

BOARDS/VIEW.PY

```
class IndexPageView(LoginRequiredMixin, TemplateView):
    login_url = '/login/'
    template_name = "index.html"
```

Para fines de prueba, verificamos el reinicio al hacer login redirigiéndolo a la raíz `return HttpResponseRedirect('/')`, en el método `login_view`, por ejemplo.

BOARDS/VIEW.PY

```
def login_view(request):
        if request.method == "POST":
```

```
                form = AuthenticationForm(request, data=request.POST)
  if form.is_valid():
    username = form.cleaned_data.get('username')
    password = form.cleaned_data.get('password')
    user = authenticate(username=username, password=password)
  if user is not None:
   login(request, user)
  messages.info(request, f"Iniciaste sesión como: {username}.")
  return HttpResponseRedirect('/')
  else:
  messages.error(request,"Invalido username o password.")
  else: messages.error(request,"Invalido username o password.")
  form = AuthenticationForm()
  return render(request=request,
template_name="registration/login.html", context={"login_form":form})
```

Verificamos que no redirige al login en http://localhost:8000, y nos muestra el mensaje de: Saludos, Hola Mundo. Procedemos a verificar los permisos con el `Mixis`, por medio de `PermissionRequiredMixin`. Para ello vamos a colocar un permiso que no hemos asignado; por ejemplo, el `boards.view_boardsmodel`. Adecuando la misma clase de index para efectos de la prueba:

Agregamos:

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

Y adecuamos:

```
class IndexPageView(LoginRequiredMixin, PermissionRequiredMixin,
TemplateView):
    login_url = '/login/'
    permission_required = 'boards.view_boardsmodel'
    template_name = "index.html"
```

Verificamos con el usuario creado, y nos refleja un error 403 Forbidden, que no tenemos acceso. Cambiamos el modelo por `permission_required = 'boards.es_miembro1'`, y observamos que nos permite el acceso, tanto nos valida que debemos haber iniciado sesión y valida un tipo de permiso particular.