

UNIVERSIDAD NACIONAL DE COLOMBIA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS E INDUSTRIAL

Entrega 06

Fecha: 14/11/2025

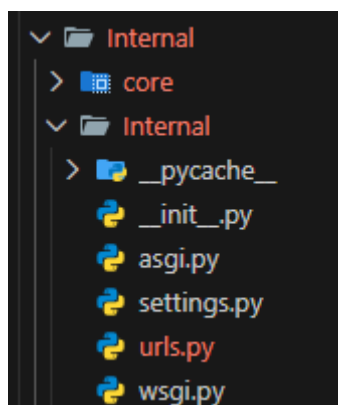
Joan Camilo Betancourt Gonzalez (jobetancourtg@unal.edu.co)
Nicolás Alejandro Diosa Benavides (ndiosab@unal.edu.co)
Jonathan Felipe López Nuñez (jolopezn@unal.edu.co)
Raúl Felipe Rodríguez Hernández (rrodriguezhe@unal.edu.co)

1. Informe de linter

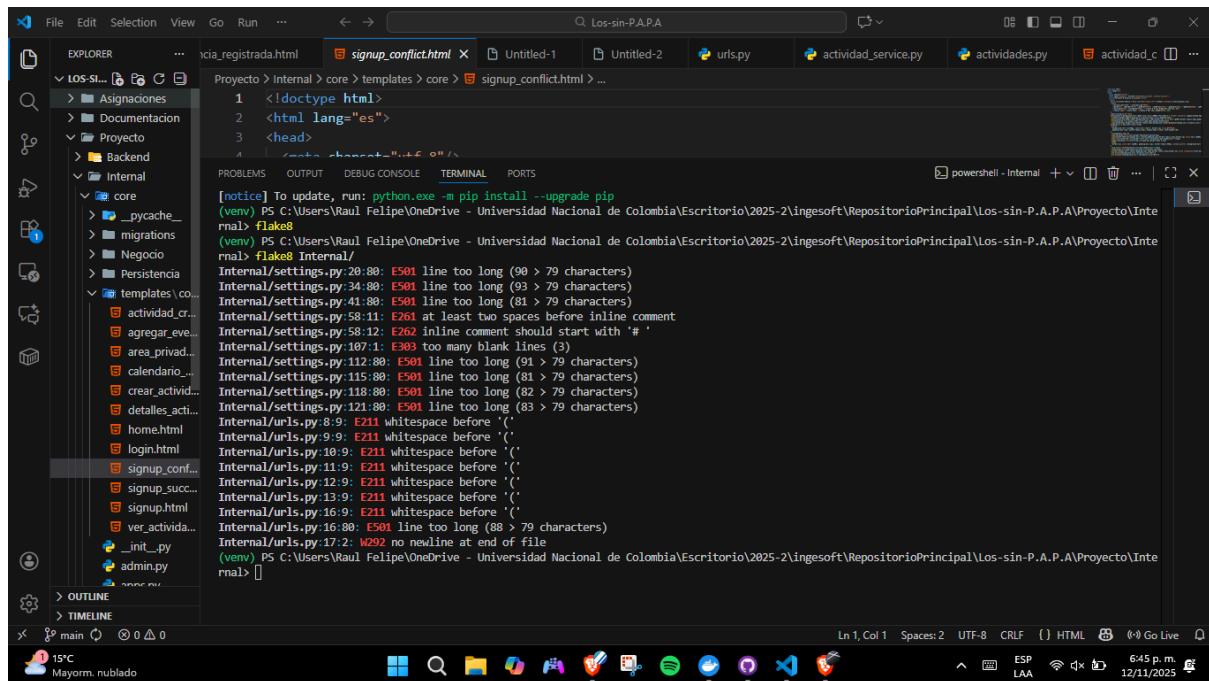
La herramienta que utilizamos es Flake 8



Es importante mencionar que utilizamos el linter para analizar dos carpetas específicas de nuestro repositorio. En primer lugar, ejecutamos la herramienta sobre la subcarpeta Internal, ubicada dentro de la carpeta principal del mismo nombre. Esta carpeta contiene los archivos `__pycache__`, `__init__.py`, `asgi.py`, `settings.py`, `urls.py` y `wsgi.py`, como se muestra en la siguiente imagen.



Luego de descargar Flake 8 en nuestro entorno virtual y ejecutarlo mediante el comando `flake8 Internal/` nos notificó que encontró los siguientes errores.



Captura de pantalla luego de ejecutar Flake8 en carpeta Internal

La captura muestra que Flake8 encontró un total de 19 errores, estos resultados eran errores de sintaxis, se puede ver que algunos son respecto a que las líneas de código superaban los 79 caracteres, el cual es la longitud de líneas que establece PEP 8, el cual es el estándar de estilo para códigos de python. Los otros errores eran respecto a espacios en blanco innecesarios.

Al momento de revisar las líneas que excedían la longitud recomendada nos dimos cuenta que todas estas líneas estaban en el archivo [settings.py](#), este archivo es generado automáticamente por Django al momento de crear el proyecto. Por esta razón, decidimos no modificar el archivo settings.py, ya que sus valores y estructura forman parte de la configuración interna del framework y cualquier cambio manual podría afectar el correcto funcionamiento del proyecto. Además este archivo evidentemente no fue creado por nosotros, el equipo desarrollador.

Debido a que no queríamos hacer modificaciones a [settings.py](#) o a diferentes archivos que no creamos nosotros mismos decidimos establecer unas reglas de negocio propias. Estas reglas se establecieron en un archivo .flake8. Se cambió la longitud máxima permitida de caracteres de cada línea de 79 a 100 y se indicó que se excluyan los errores presentados en `__pycache__`, `migrations`, `env` y `venv`.

The screenshot shows the VS Code interface with the Explorer pane on the left displaying the project structure. The file `.flake8` is selected. The main editor shows the content of `.flake8`:

```
1 [flake8]
2 max-line-length = 100
3 exclude = __pycache__, migrations, env, venv
4
```

The TERMINAL pane at the bottom shows the command `flake8 Internal/` being executed, resulting in several `E211` errors (whitespace before '(') and one `W292` error (no newline at end of file).

Reglas de negocio establecidas en el archivo flake8 y resultado de su ejecución.

En la imagen se puede ver que luego de establecer estas reglas la herramienta solo mostró los errores de los espacios innecesarios. Estos cambios se corrigieron y este fue el resultado.

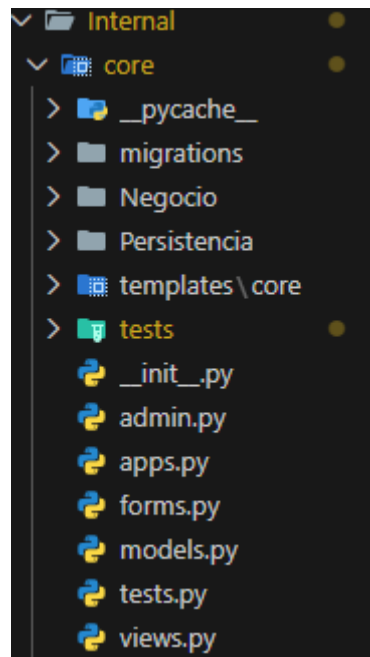
The screenshot shows the VS Code interface with the Explorer pane on the left displaying the project structure. The file `.flake8` is selected. The main editor shows the content of `.flake8`:

```
6 urlpatterns = [
7     path('admin/', admin.site.urls),
8     path('', views.home, name='home'),
9     path('actividades/', views.ver_actividades, name='actividades'),
10    path('area_privada/', views.ver_area_priv, name='area_privada'),
11    path('signup/', views.signup, name='signup'),
12    path('login/', views.login, name='login'),
13    path('logout/', views.logout, name='logout'),
14    path('crear_actividad/', views.crear_actividad, name='crear_actividad'),
15    path('actividad_creada/', views.actividad_creada, name='actividad_creada'),
16    path('actividades/cuid:id/', views.detalles_actividad, name='detalles_actividad')
17 ]
18
```

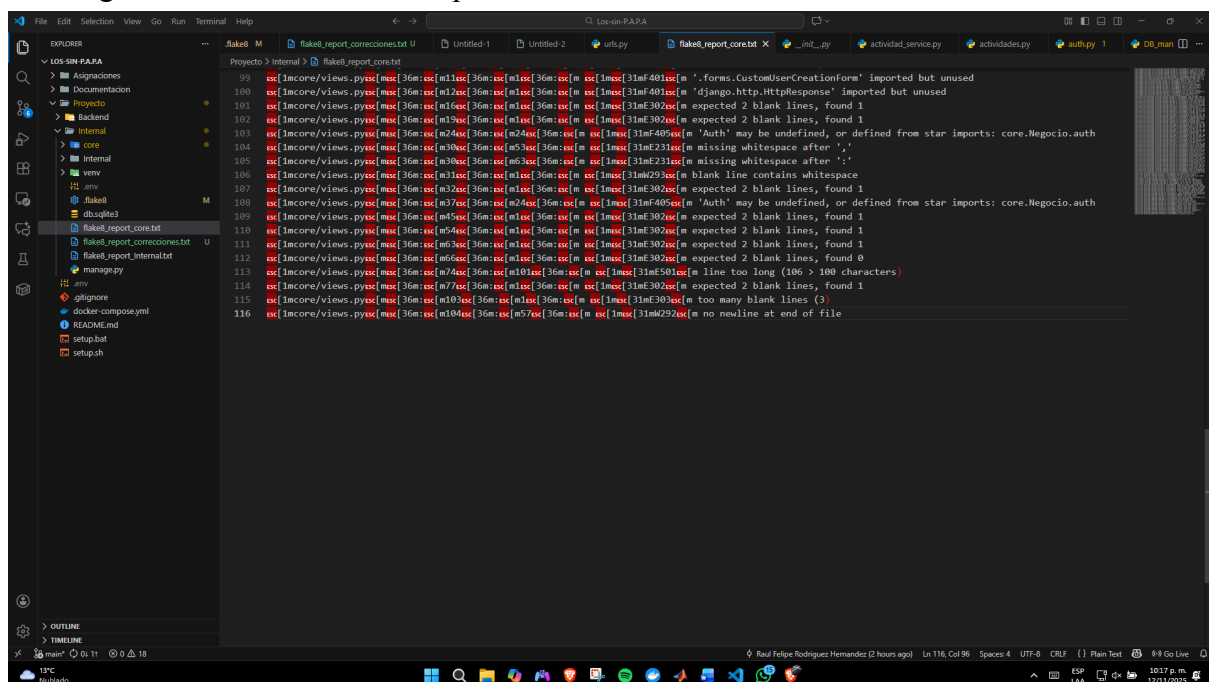
The TERMINAL pane at the bottom shows the command `flake8 Internal/` being executed, resulting in no errors.

Como se puede apreciar ya no se reportó ningún error. 😊

Después lo que hicimos fue correr Flake8 dentro de la carpeta core, en esta carpeta está la capa de Persistencia, negocio y presentación.



Utilizamos el comando `flake8 core/ --output-file=flake8_report.txt` además de ejecutar Flake8 generamos un archivo de reporte con los errores encontrados.



Archivo de reporte de errores encontrados en core

Se puede observar que este archivo tiene 116 líneas, lo que quiere decir que se encontraron 116 errores en total. De la misma manera que ocurría en la carpeta Internal, estos errores corresponden principalmente a problemas de estilo, organización del código y uso incorrecto de imports. Nos dimos cuenta que había varios imports que no eran utilizados, muchísimos errores con espacios en blanco que a veces a simple vista no son fáciles de detectar 🤖, problemas con saltos de línea ya que algunas veces faltaban las dos líneas en blanco requeridas entre definiciones de funciones o clases y a veces había un exceso de líneas en

blanco, también se observaron muchas líneas que excedían el límite de 100 caracteres que habíamos definido.

Corregimos la gran mayoría de errores, sin embargo en este proceso nos dimos cuenta que casi todos los errores de longitud de línea estaban en nuestro DB_manager, archivo que funciona como el controlador central de nuestra base de datos y contiene un número considerable de métodos create_* cuyos parámetros corresponden directamente a los campos de los modelos de Django. Debido a la cantidad de atributos necesarios en cada operación de creación, las llamadas a Model.objects.create() tienden naturalmente a ser extensas.

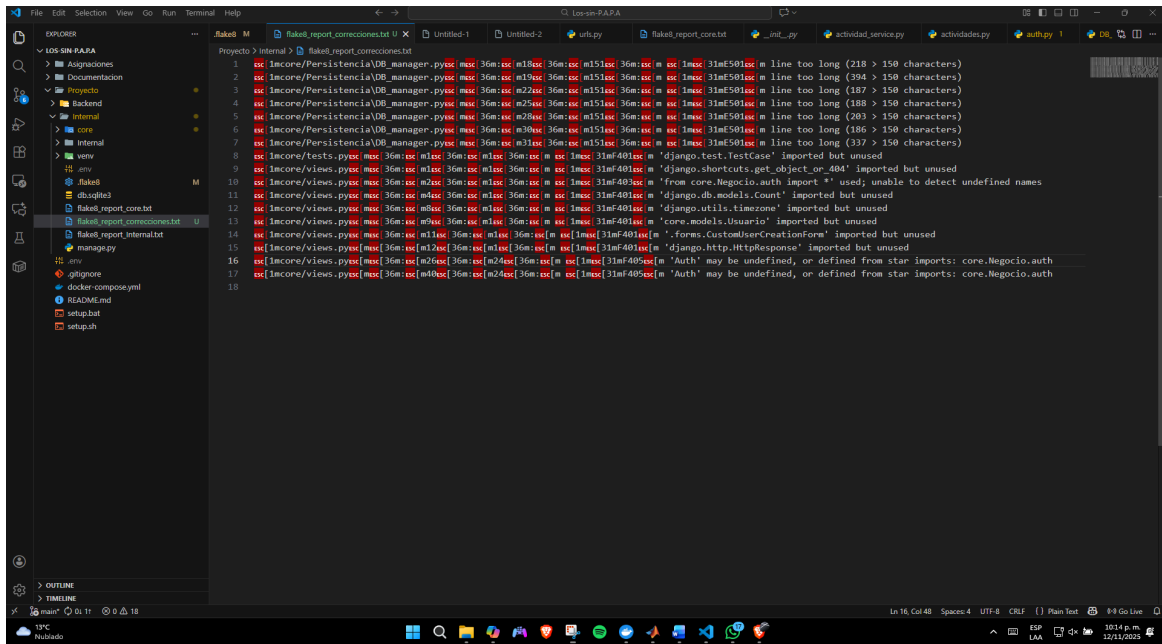
Es por esto que decidimos no modificar la longitud de líneas en este archivo y aumentar la longitud de línea recomendada a 150 caracteres. Aunque no queríamos modificar el DB_manager respecto a la longitud de las líneas, sí queríamos que Flake8 nos informara respecto a otros errores presentes en este archivo, por eso es que no le indicamos a la herramienta que ignorara el archivo DB_manager.py

Luego de corregir los errores podemos ver la salida en la terminal al ejecutar el comando `flake8 core/`

```
(venv) PS C:\Users\Raul_Felipe\OneDrive - Universidad Nacional de Colombia\Escritorio\2025-2\Ingesoft\RepositorioPrincipal\Los-sin-P.A.P.A\Proyecto\Internal> flake8 core/
core/Persistencia\DB_manager.py:18:151: E501 line too long (218 > 150 characters)
core/Persistencia\DB_manager.py:19:151: E501 line too long (394 > 150 characters)
core/Persistencia\DB_manager.py:22:151: E501 line too long (187 > 150 characters)
core/Persistencia\DB_manager.py:25:151: E501 line too long (188 > 150 characters)
core/Persistencia\DB_manager.py:28:151: E501 line too long (283 > 150 characters)
core/Persistencia\DB_manager.py:30:151: E501 line too long (186 > 150 characters)
core/Persistencia\DB_manager.py:31:151: E501 line too long (337 > 150 characters)
core/tests.py:1:1: F401 'django.test.TestCase' imported but unused
core/views.py:1:1: F401 'django.shortcuts.get_object_or_404' imported but unused
core/views.py:2:1: F403 'from core.Negocio.auth import *' used; unable to detect undefined names
core/views.py:4:1: F401 'django.db.models.Count' imported but unused
core/views.py:8:1: F401 'django.utils.timezone' imported but unused
core/views.py:9:1: F401 'core.models.Usuario' imported but unused
core/views.py:11:1: F401 'forms.CustomUserCreationForm' imported but unused
core/views.py:12:1: F401 'django.http.HttpResponse' imported but unused
core/views.py:26:24: F405 'Auth' may be undefined, or defined from star imports: core.Negocio.auth
core/views.py:40:24: F405 'Auth' may be undefined, or defined from star imports: core.Negocio.auth
```

Captura de consola luego de ejecutar Flake8 en core por última vez

Se puede apreciar que aún quedan errores por corregir, estos errores hacen referencia a DB_manager, imports que aún no se han utilizado pero que se van a utilizar en un futuro y a sintaxis y elementos del código que consideramos que son necesarios para el correcto funcionamiento y entendimiento del proyecto.



Reporte después de correcciones en core

Se generó un reporte con los errores presentados después de las correcciones y se puede apreciar que el número pasó de 116 a 17. 🎉