

Punto 7

Linter: Elegimos usar **pylint** ya que es útil para asegurarnos de que el código siga buenas prácticas sin tener que revisar todo manualmente; nos ayuda a detectar errores antes de ejecutar el programa, señalar variables que no usamos, importaciones innecesarias y problemas de formato además de un puntaje de calidad, lo que hace más fácil ver si estamos mejorando o no.

Implementación:

Creamos un entorno virtual:


```
C:\Users\juand\Desktop\git hub\django-rest-framework>python -m venv venv

C:\Users\juand\Desktop\git hub\django-rest-framework>venv\Scripts\activate

(venv) C:\Users\juand\Desktop\git hub\django-rest-framework>
```

Ejecución:

Inicialmente el código se veía de la siguiente manera:



```
models.py | X
bookings > models.py > MedicalNote
1 from django.db import models
2 from doctors.models import Doctor
3 from patients.models import Patient
4
5
6 class Appointment(models.Model):
7     patient = models.ForeignKey(
8         Patient, related_name='appointments', on_delete=models.CASCADE
9     )
10    doctor = models.ForeignKey(
11        Doctor, related_name='appointments', on_delete=models.CASCADE
12    )
13    appointment_date = models.DateField()
14    appointment_time = models.TimeField()
15    notes = models.TextField()
16    status = models.CharField(max_length=10)
17
18
19 class MedicalNote(models.Model):
20     appointment = models.ForeignKey(
21         Appointment, related_name='medical_notes', on_delete=models.CASCADE
22     )
23     note = models.TextField()
24     date = models.DateField()
```

En una primera ejecución con el comando **pylint models.py** nos dio la siguiente salida:

```
(venv) C:\Users\juand\Desktop\git hub\django-rest-framework\bookings>pylint models.py
***** Module bookings.models
models.py:1:0: C0114: Missing module docstring (missing-module-docstring)
models.py:6:0: C0115: Missing class docstring (missing-class-docstring)
models.py:19:0: C0115: Missing class docstring (missing-class-docstring)

-----
Your code has been rated at 7.86/10 (previous run: 7.14/10, +0.71)
```

Donde se muestran los siguientes errores:

C0114: Missing module docstring: Básicamente, el archivo no tiene un comentario al inicio explicando qué hace. No es obligatorio para que funcione, pero ayuda a que otros entiendan más rápido de qué trata el código.

C0115: Missing class docstring: Las clases Appointment y MedicalNote no tienen una descripción. Pylint recomienda poner un comentario corto dentro de cada clase diciendo qué representa, para que sea más fácil de leer.

La calificación del código fue **7.86/10**, lo que no está mal, pero hay espacio para mejorar. Para arreglar estos avisos solo hay que agregar un par de comentarios explicando el propósito del archivo y de cada clase

Luego siendo corregido el código, paso a verse de la siguiente manera:

```
models.py 1 X
bookings > models.py > Appointment
1  """Módulo que define los modelos Appointment y MedicalNote."""
2
3  from django.db import models
4  from doctors.models import Doctor
5  from patients.models import Patient
6
7
8  class Appointment(models.Model):
9      """Modelo que representa una cita médica entre un paciente y un doctor."""
10     patient = models.ForeignKey(
11         Patient, related_name='appointments', on_delete=models.CASCADE
12     )
13     doctor = models.ForeignKey(
14         Doctor, related_name='appointments', on_delete=models.CASCADE
15     )
16     appointment_date = models.DateField()
17     appointment_time = models.TimeField()
18     notes = models.TextField()
19     status = models.CharField(max_length=10)
20
21
22     class MedicalNote(models.Model):
23         """Modelo que representa una nota médica asociada a una cita."""
24         appointment = models.ForeignKey(
25             Appointment, related_name='medical_notes', on_delete=models.CASCADE
26         )
27         note = models.TextField()
28         date = models.DateField()
```

Y luego en una segunda ejecución con el comando **pylint models.py** nos dio la siguiente salida:

```
(venv) C:\Users\juand\Desktop\git hub\django-rest-framework\bookings>pylint models.py
-----
Your code has been rated at 10.00/10 (previous run: 7.86/10, +2.14)
```

Después de hacer las correcciones, ejecutamos Pylint nuevamente y ahora el código obtuvo una calificación de 10.00/10, lo que significa que ya no hay advertencias ni errores. Antes, la calificación era 7.86/10, así que mejoró en +2.14 puntos.

Las mejoras que hicimos fueron agregar docstrings al módulo y a las clases, lo que ayudó a que el código sea más claro y cumpla con las buenas prácticas recomendadas

Opiniones del código

Juan David Suárez

"El código de mis compañeros está bien estructurado pero algunos archivos tienen pocas explicaciones. En general sí puedo modificarlo sin mucho problema pero creo que deberíamos mejorar los comentarios para que sea más fácil para todos."

John Jairo Paez

"Revisé varios archivos y en su mayoría son entendibles, pero noto que algunos nombres de variables podrían ser más descriptivos. En algunas vistas, hay funciones con nombres muy genéricos y toca leer toda la función para saber qué hace, si mantenemos un estándar de nombres claros será más fácil trabajar en equipo."

Nicolas Arturo Jaramillo

"Después de ejecutar Pylint, vi que hay muchos warnings por falta de docstrings en funciones y clases, también detectó algunas líneas demasiado largas y algunas importaciones que no se están usando, no es que el código esté mal, pero hay detalles que podemos mejorar para que sea más limpio y mantenible."

Juan Diego Velásquez

"El código en general estaba bien, pero había detalles que Pylint marcaba, sobre todo por falta de comentarios, no eran errores graves pero mejoraba la claridad. Hicimos unos ajustes y ahora todo está más ordenado."