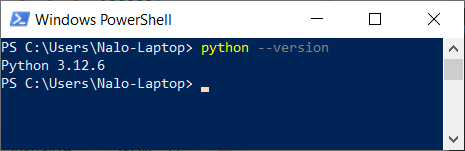
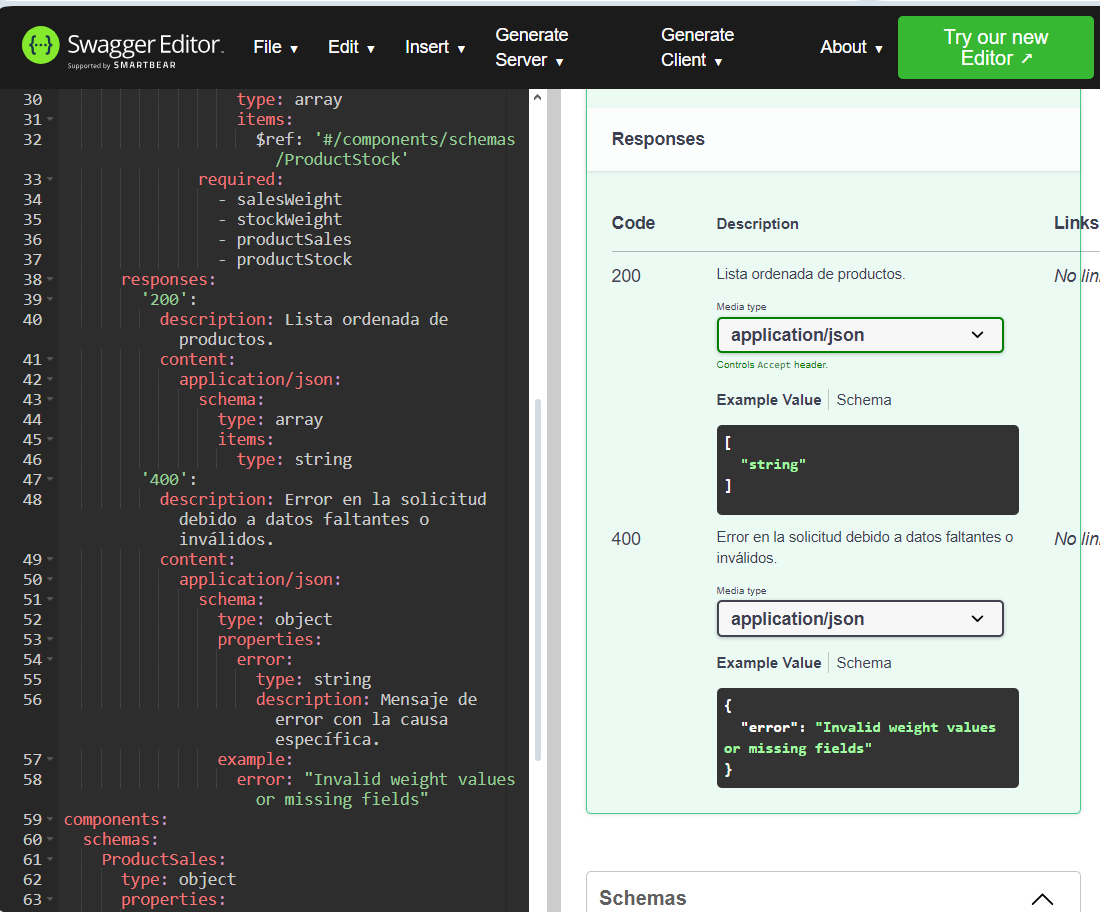
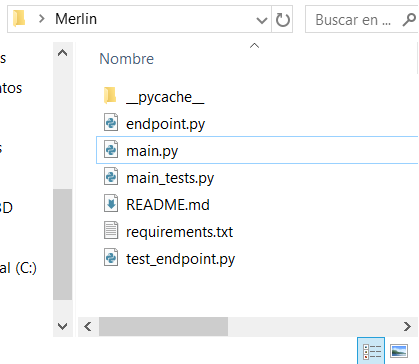
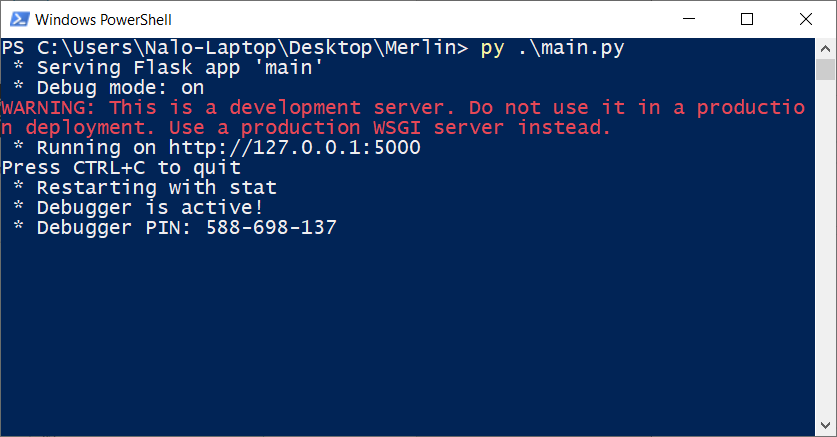
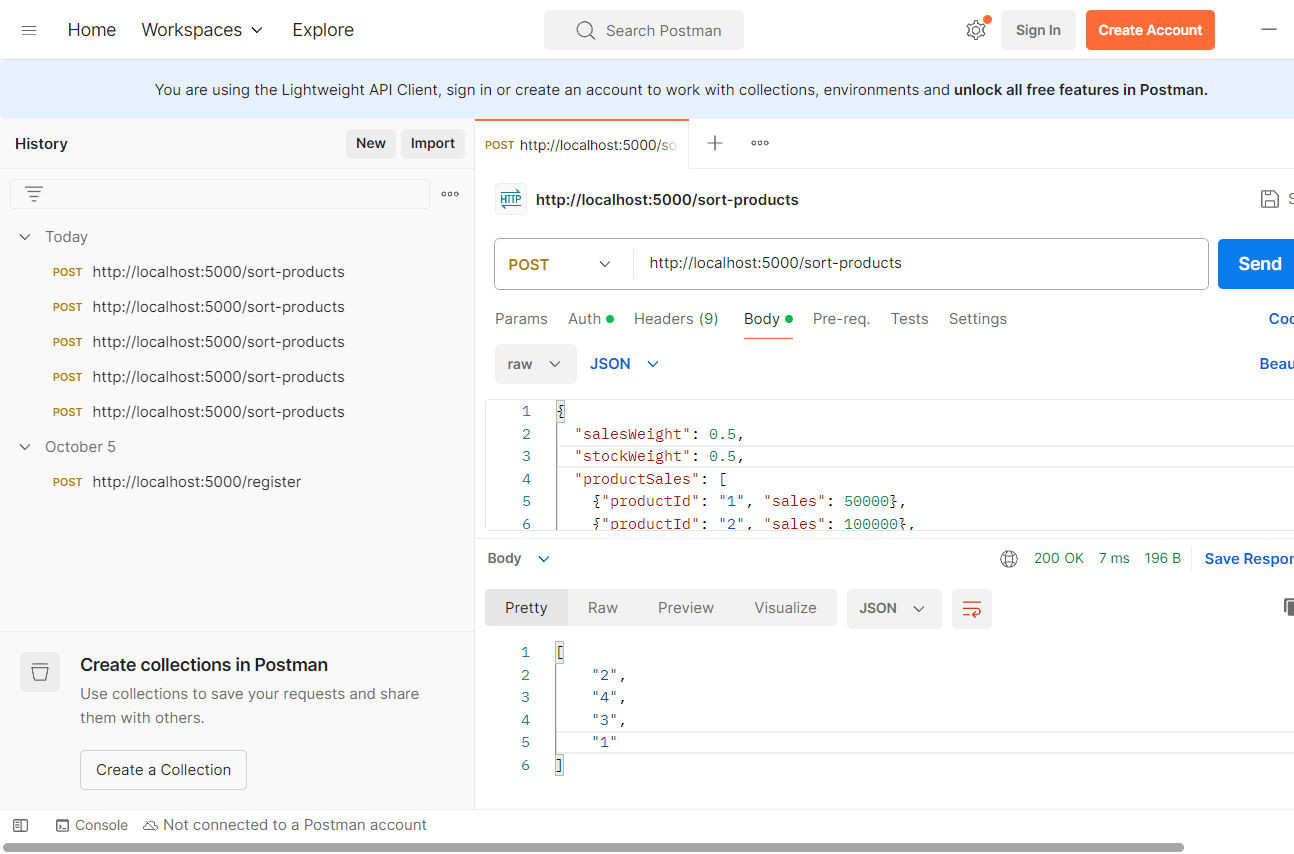
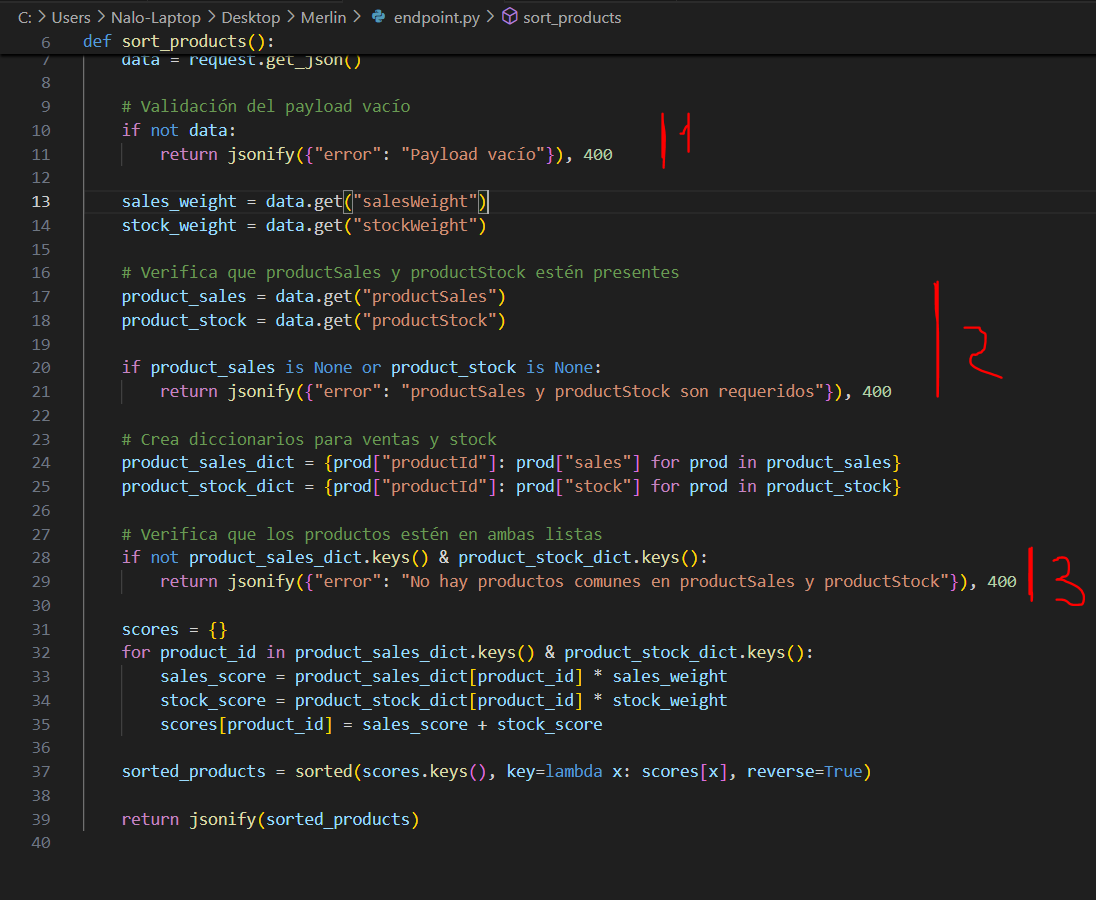
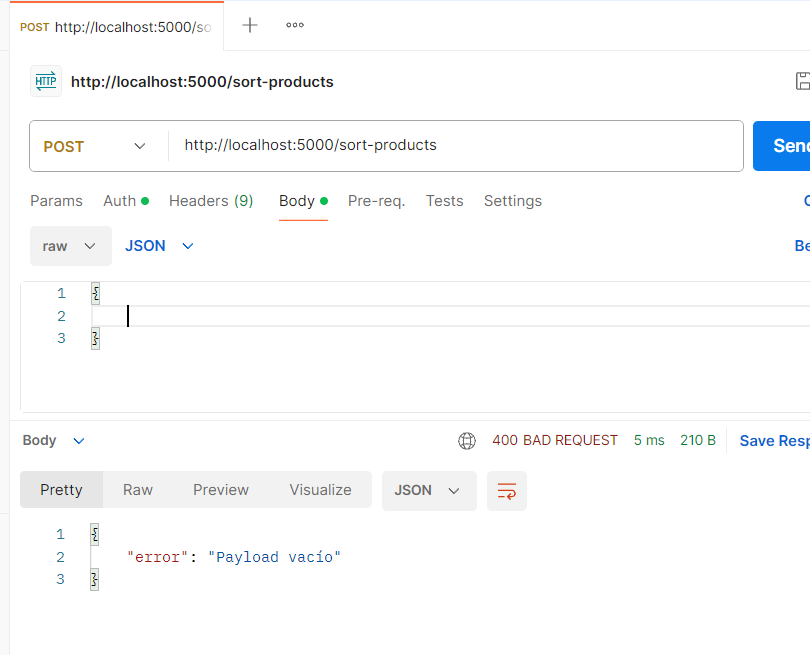
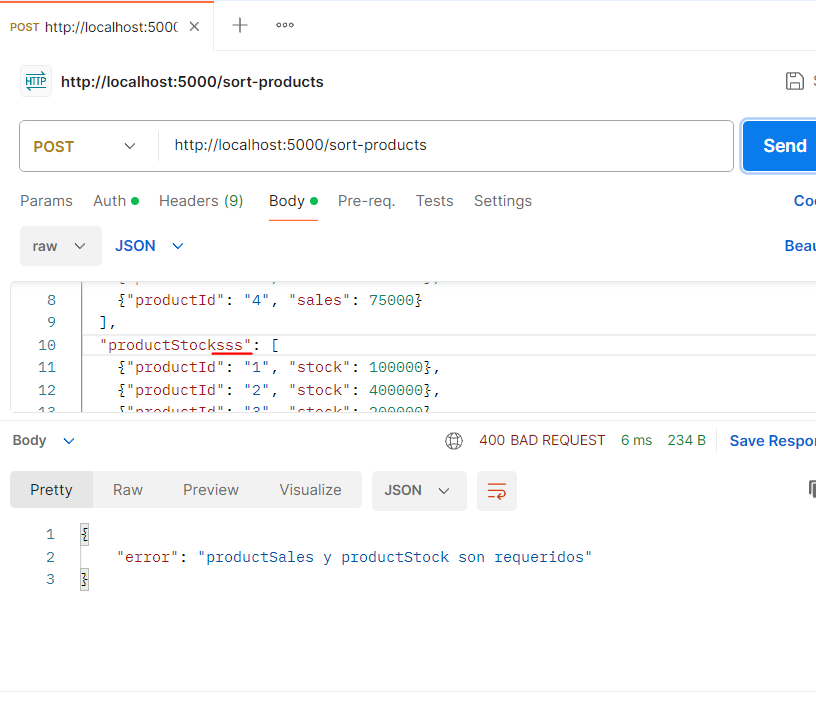
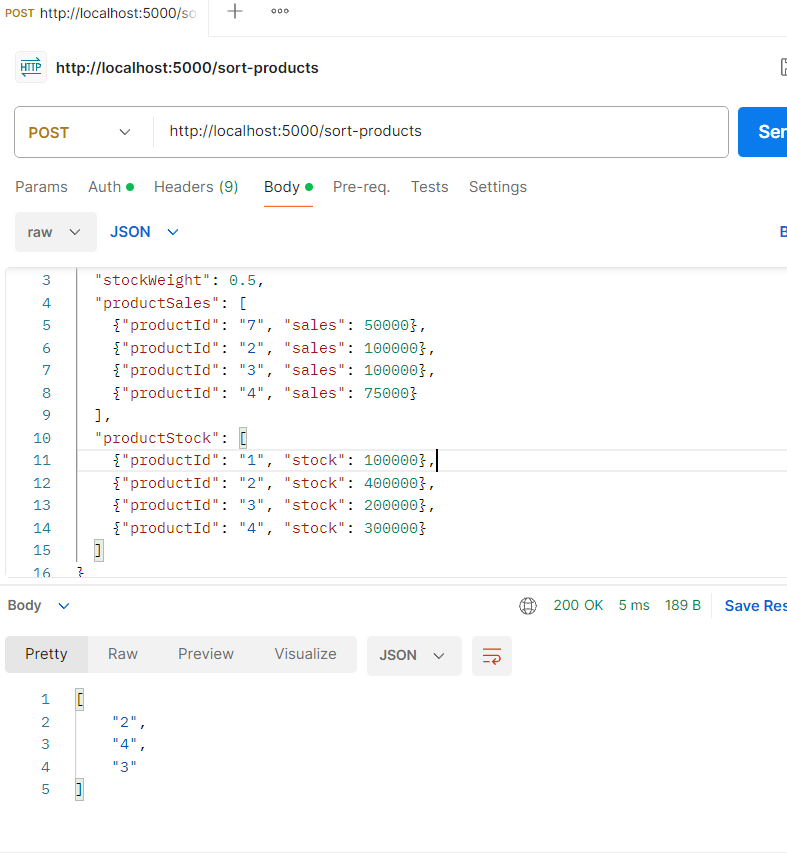
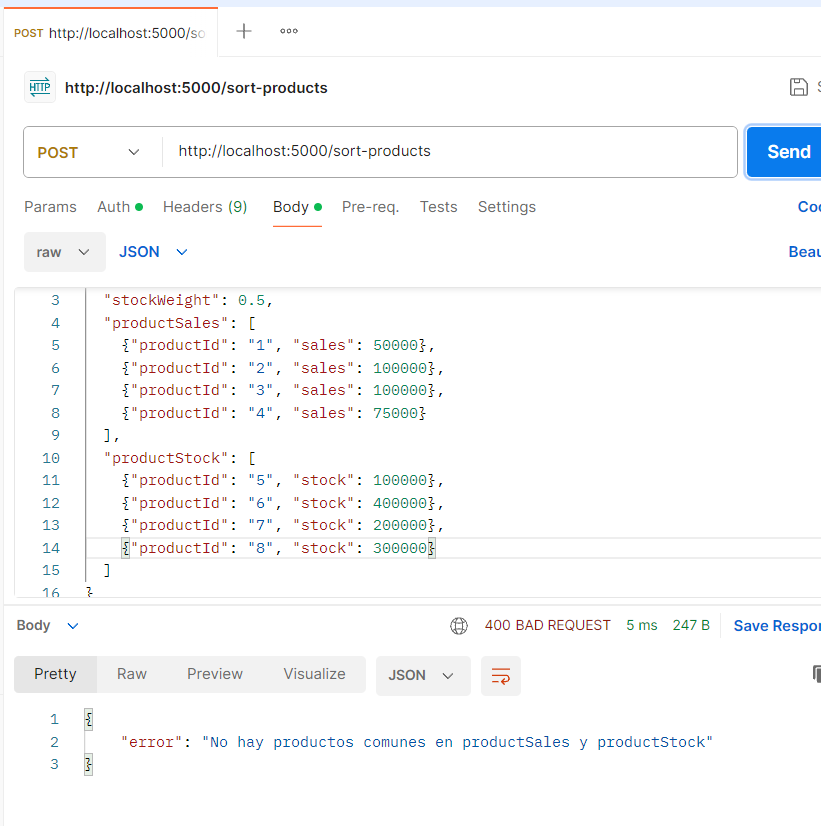
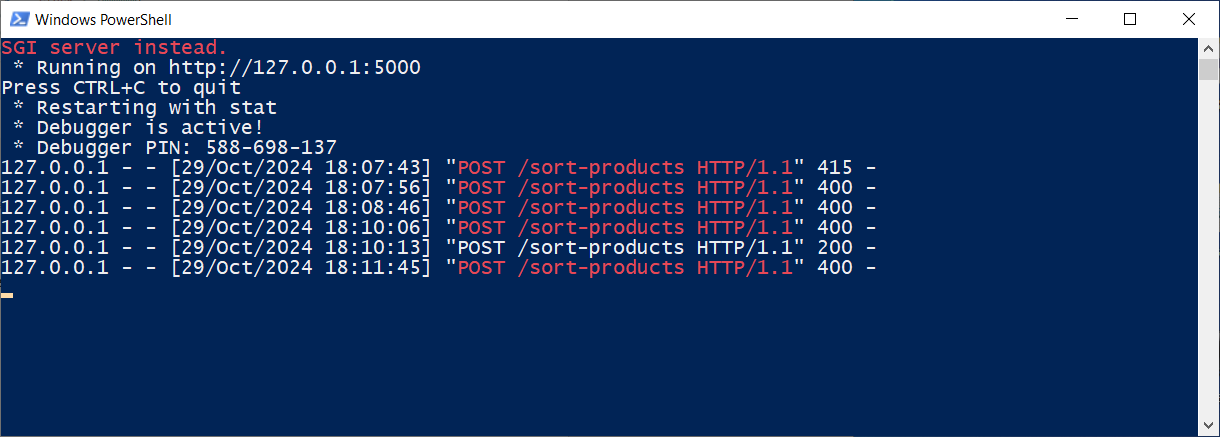
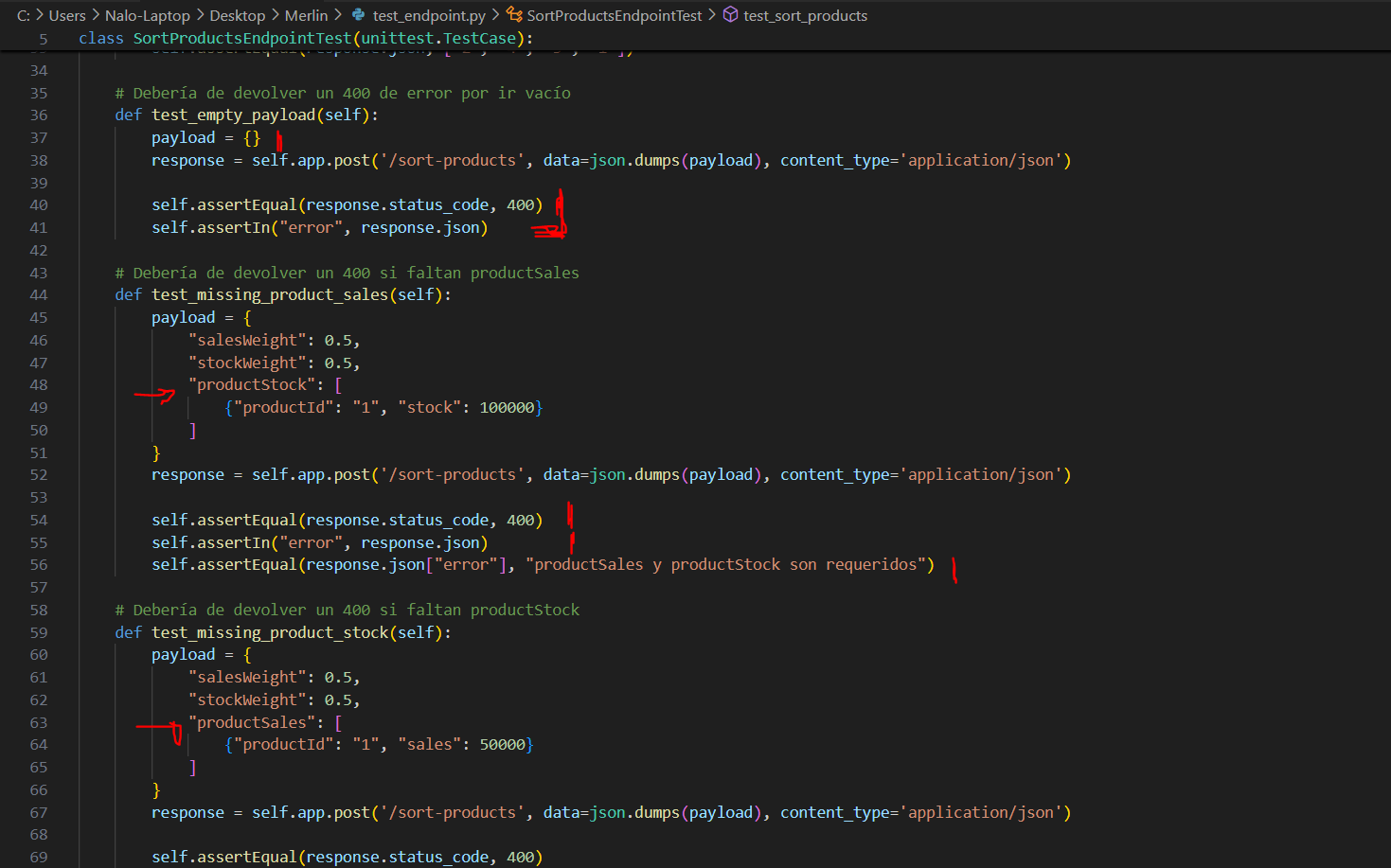
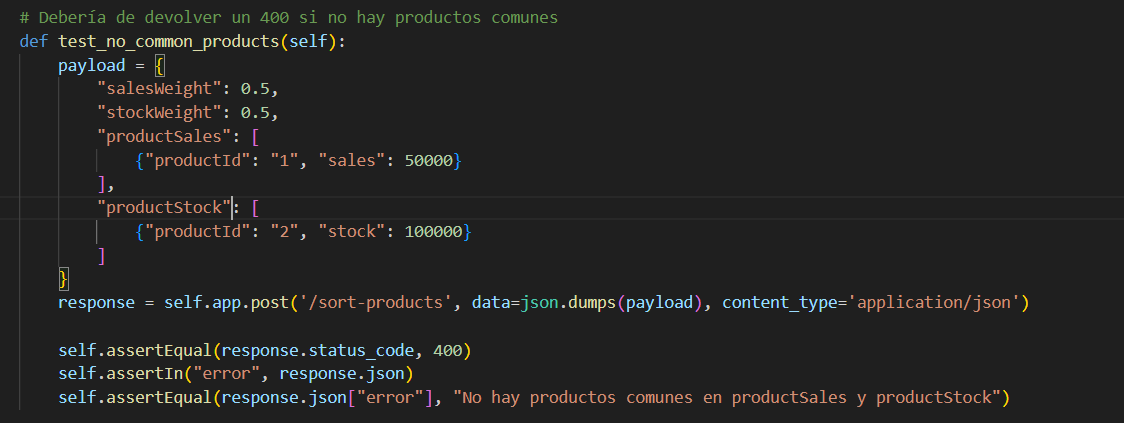
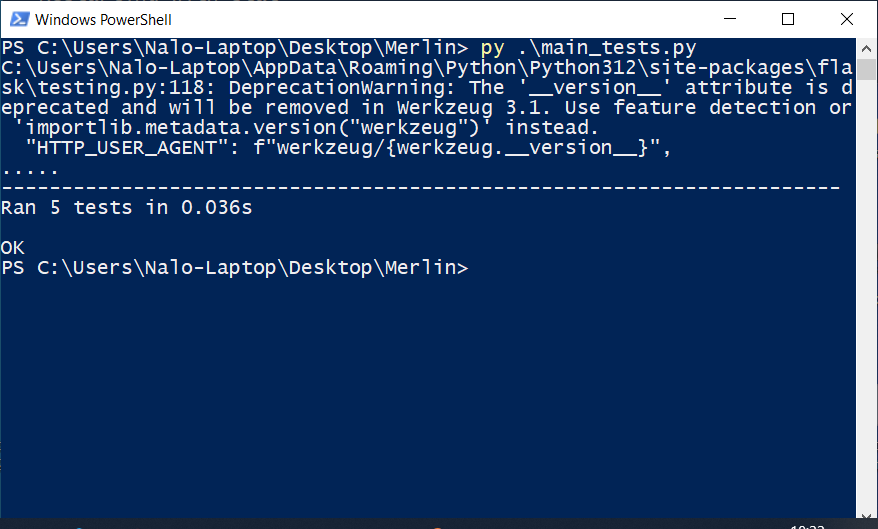
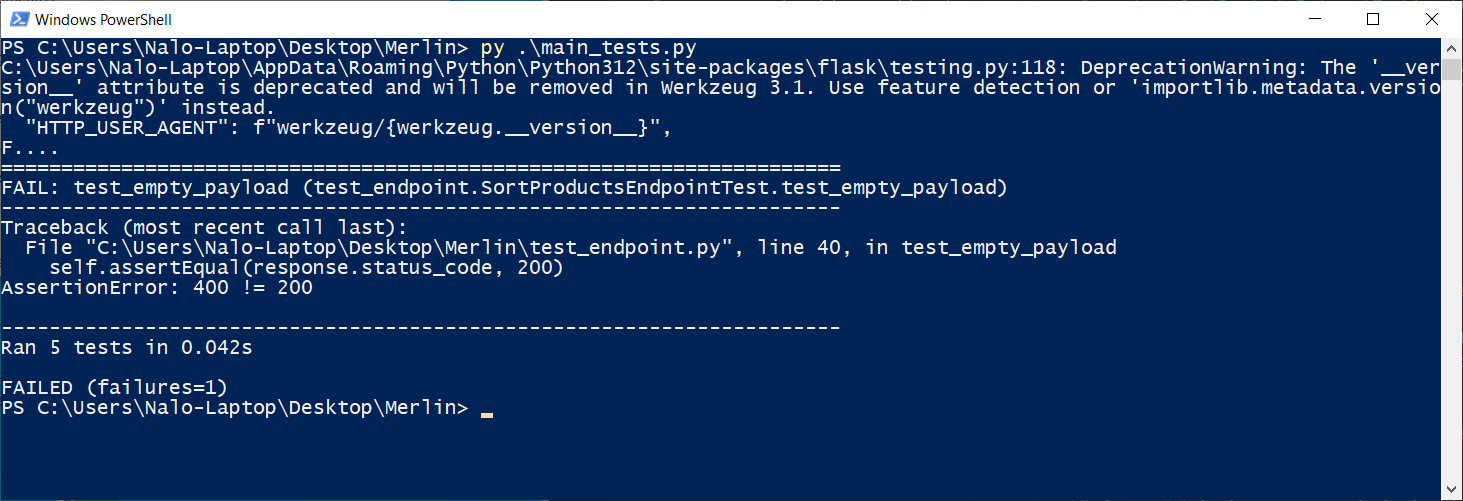
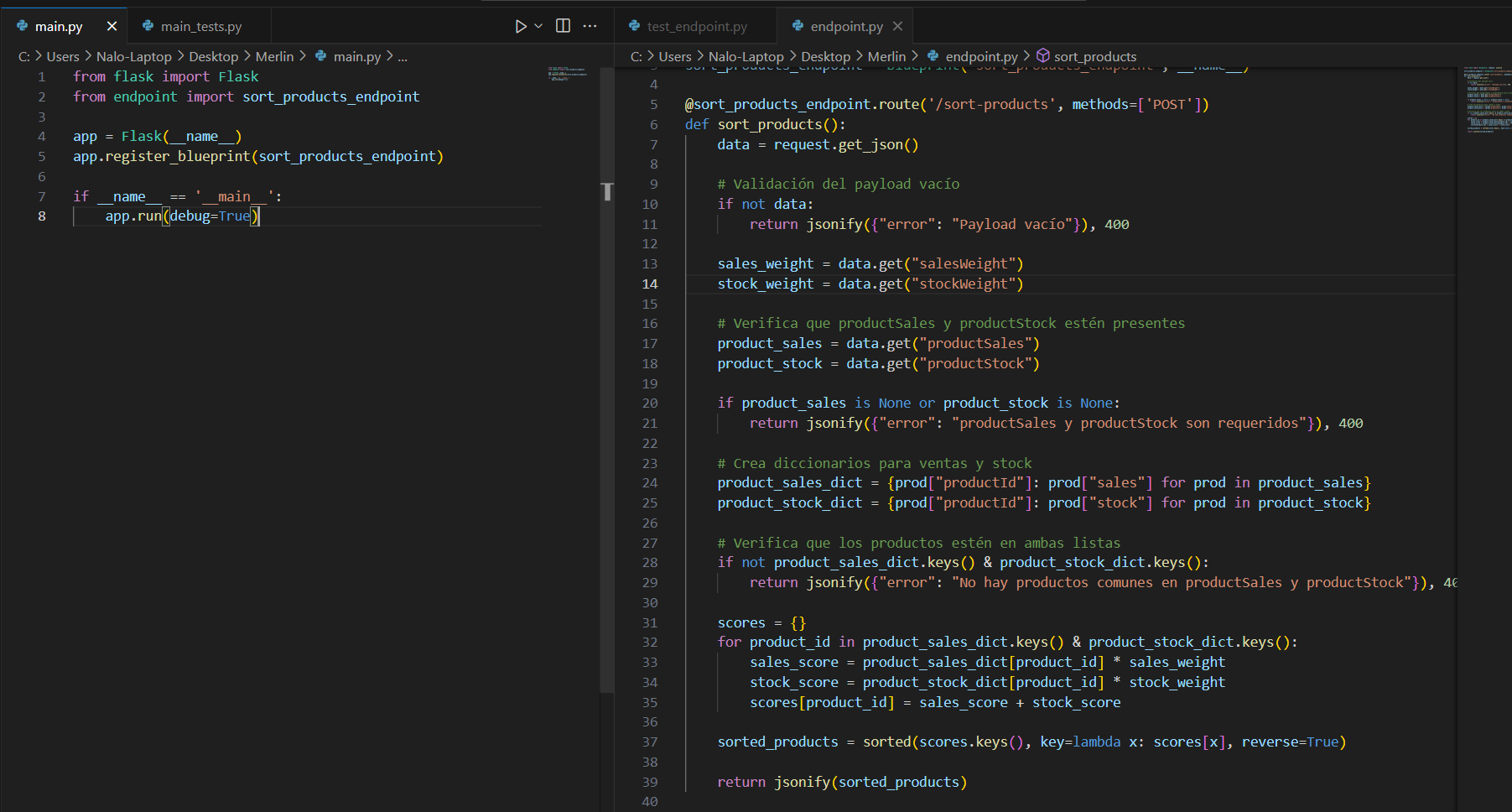
**PRUEBA TECNICA DAVID BEATO MEMORIA**  
  
En mi caso elegí Python, suelo estar más acostumbrado a manejar apirest con java pero por la robustez de los proyectos donde trabaje , por ir fuertemente tipado  
 Pero para la siguiente prueba lo realice en Python un poco por salir de la monotonía y por por un poco la velocidad que se supone que necesitaba otorgarle al proyecto ( según comentabais que era una prueba de concepto para mi empresa actual y querían ver algo medianamente funcional y rápido)  
Es por eso que elegi Python y Flask para poder también tener algo desacoplado rápido y fácil , es muy usado y está bien documentado - <https://flask.palletsprojects.com/en/stable/>   
  
para los test suelo estar también acostumbrado a usar JUNIT para java y aquí el framwork similar seria este:   
<https://docs.python.org/3/library/unittest.html> y fue el que use para las pruebas unitarias por lo mismo por que es de los mas usados está muy testeado y tiene buena documentación   
  
El IDE que use es Visual, porque actualmente no tengo otro insalado en el portátil pero suelo usar pycharm al estar siempre trabajando con intellij aunque maneje algunos otros IDE en todos estos años.  
  
Empezare mirando la versión que tengo de Python instalada  
  
En mi caso 3.12.6  
  
  
Realmente en el Swagger que me facilitasteis solo tiene una respuesta 200 ósea que todo puede salir bien entonces me tome un poco la libertad de añadí un 400 para errores , que luego use en el test realizo algún posible error, se que existen muchos mas errores que no controlo, pero bueno para la entrega creo que es suficiente.   
  
Inicio del servicio que lo tengo organizado de la siguiente manera:  
  
abrimos la consola en el proyecto y ejecutamos nuestro main ahora tenemos el servicio a la escucha en el puertyo 5000  
  
  
 Realizamos la primera prueba hagamos la prueba correcta pasando todo lo necesario:  
  
Vemos que nos lo ordena y nos devuelve un 200 por lo que esta correcto intentemos llegar a los casos supuesto que estamos controlando:  
  
Supuesto numero 1: mandamos un payload vacio  
  
Supuesto numero 2:  
  
terer supuesto basta con cambiar y añadir varias S para que salte el error de que no tenemos productStocks el 3 supuesto: Primero vamos a modificar 1 solo de ellas para ver que se ejcuta a pesar de no tener otro de ellos  
  
Aunque si ahora cambiamos todas las claves:  
  
Nos da el error de que no existen cosas que cotejar unas con las otras.  
Adjunto pantallazo de la consola tambien de ejecución:  
  
Vamos a ahora a ver los test:  
  
Explicacion de los test:  
  
  
En el primer apartado forzamos el payload para que este vacio, por lo que deberia de dar un 400 como assert en este caso es correcto tamein esperamos recibir una etiqueta "error"   
  
Segundo caso quitamos de nuestro payload productsale por lo que aceptamos un error 400  
y esperamos la misma respuest de error  
  
Tercer caso quitamos de nuestro payload producStock por lo que aceptamos un error 400  
y esperamos la misma respuesta de error  
  
Y el Cuarto caso quitamos forzamos el payload a no tener Ids en comun para que no salte de nuevo el 400 con la etiqueta error y en este caso deberia ser que no hay productos comunes para cotejar.



Ejecucion de los test  
  
Podemos cambiar la estructura de algún test para ver lo siguiente ( esto nos ayudara como bien sabéis a que si se modifica algo del código efectivamente sale el error de que algo se cambio y no se rehicieron los test o se cambio por error)  
Cambie el assert de unos de los test:  
  
para ver que efectivamente me devuelve el error al yo cambiarle que debería de devolver un 200 y me devuelve un 400  
  
  
  
  
  
  
  
  
  
  
  
  
  
Parte Aplicativo Main  
  
Parte Test  
