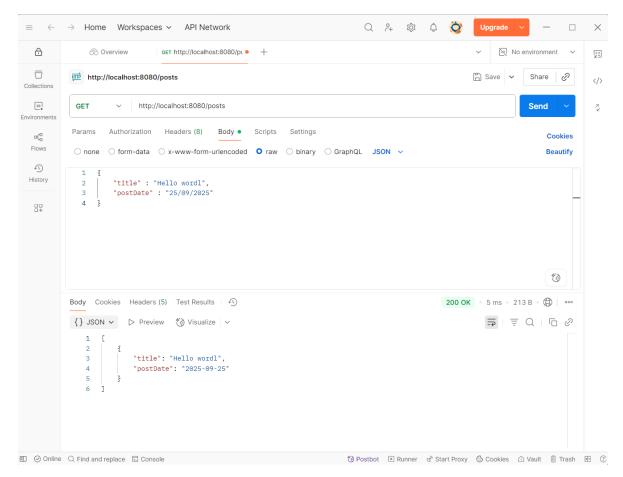
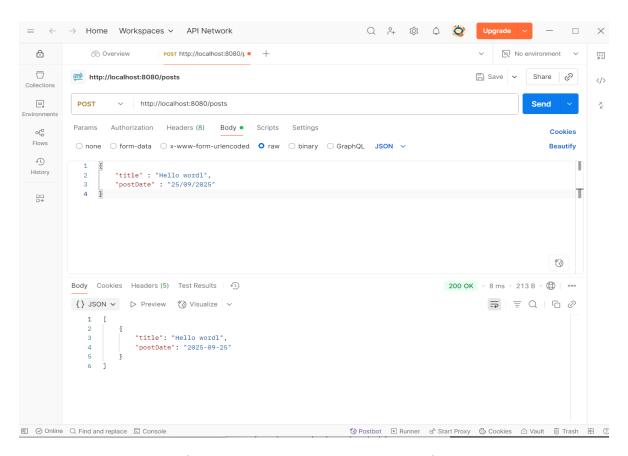
Norman Enmanuel Quijano Amaya QA240640

Guia07

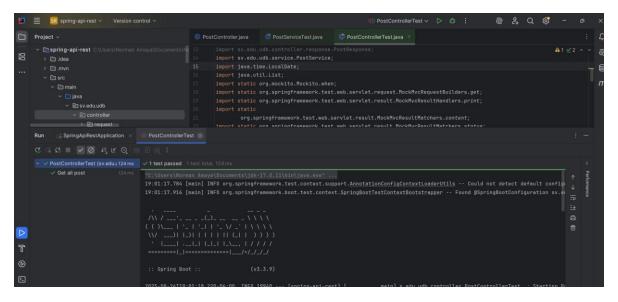
Desarrollo de Aplicaciones con Web Frameworks DWF404 G03L



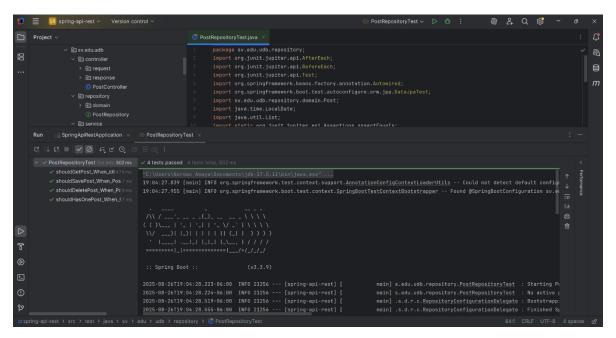
Ya al correr el SpringAplication nos vamos al postman a probar nuestra api rets donde con el método get podemos visualizar lo que registramos con el método post.



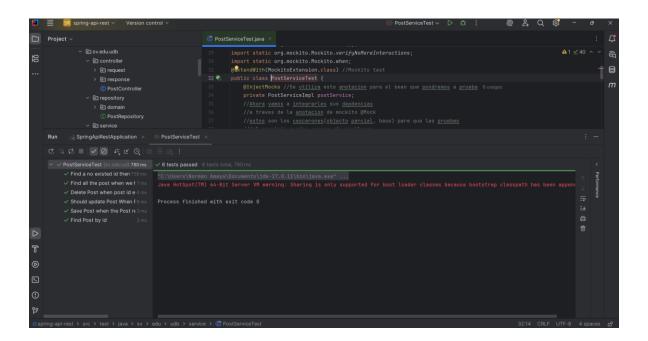
En esta captura es el registro de los datos que queremos registrar y de tal manera se va a visualizar cuando pongamos el método get como se mostraba en la captura anterior.



Las pruebas del controller verifican la capa de exposición, es decir los endpoints REST de la aplicación. Para esto se utiliza la anotación @WebMvcTest, que levanta un contexto mínimo del framework web de Spring y permite simular peticiones HTTP con la clase MockMvc. En estas pruebas no se prueba la lógica de negocio ni la base de datos, sino que se mockea el servicio (@MockBean PostService) y se valida que el controlador responda correctamente a las solicitudes. Por ejemplo, al hacer un GET /posts, se espera recibir un JSON con los datos y un código de estado 200 OK. Este nivel garantiza que los endpoints estén bien configurados y devuelvan lo que el cliente espera.



Las pruebas del **repository** se enfocan en validar la capa de persistencia, es decir, la parte que se comunica directamente con la base de datos. Para esto se utiliza la anotación @DataJpaTest, que permite probar las operaciones de JPA en un contexto aislado con una base en memoria H2. En este nivel se verifica que los métodos de PostRepository (como findAll, findByld, save y deleteByld) funcionen correctamente. Un ejemplo típico sería guardar un objeto Post y comprobar que puede recuperarse con findByld, o bien, eliminar un registro y confirmar que ya no existe en la base.



Las pruebas del **service** validan la lógica de negocio de la aplicación, que normalmente está en la clase PostServiceImpl. Aquí no se utiliza una base de datos real, sino que se trabaja con **mocks** y también con **Mockito**. Esto permite simular el comportamiento de dependencias como el PostRepository y el PostMapper. Con estas pruebas se comprueba que los métodos del servicio ejecuten la lógica esperada: por ejemplo, que al buscar por un ID válido se devuelva un PostResponse, que al buscar un ID inexistente se lance una excepción, o que un PostRequest se convierta, guarde y retorne correctamente. El enfoque principal está en probar reglas de negocio sin depender de la infraestructura.