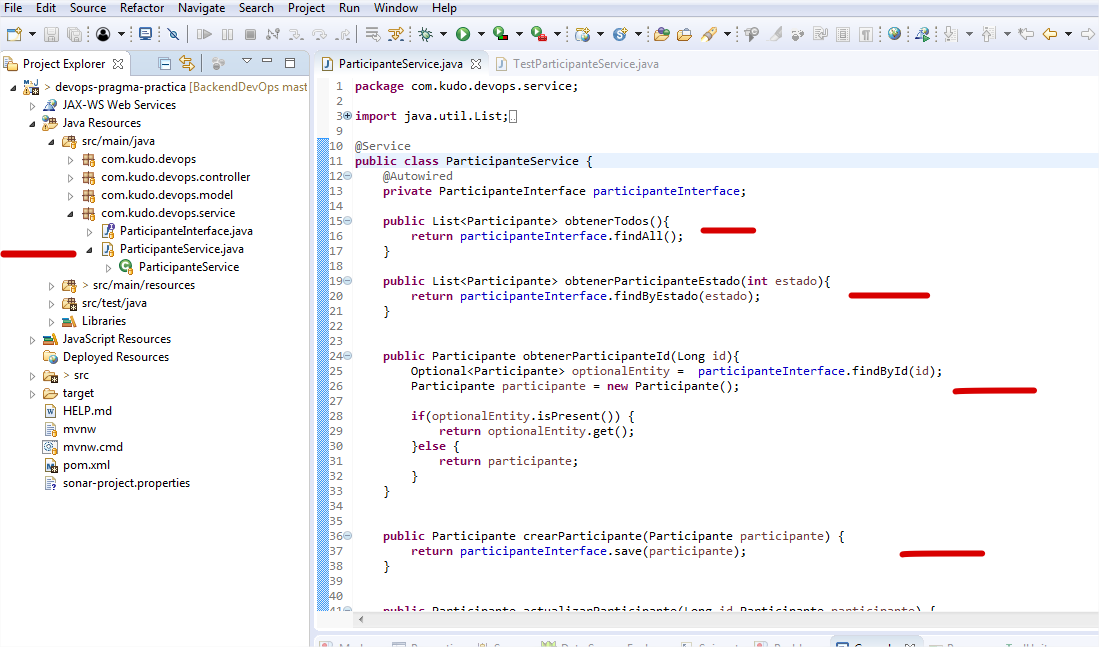
**Pruebas Unitarias Usando Mocks**

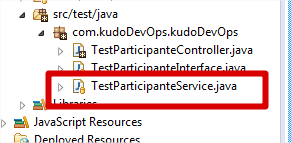
A lo largo de la práctica en el grupo de DevOps hemos visto diferentes procedimientos que se llevan a cabo dentro de la metodología ágil uno de estos procedimientos es el UnitTesting o Pruebas Unitarias. Que son pequeños trozos de código diseñados para comprobar que las funcionalidades que desarrollamos se comporten como esperamos.

A continuación se muestra un ejemplo de la implementación del Unit Testing usando como ejemplo nuestra aplicación de práctica realizada con Spring Boot

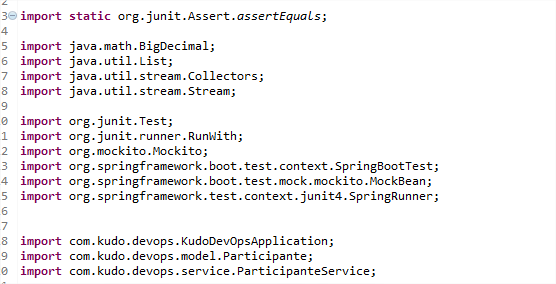
1. Probaremos las operaciones de nuestra clase ***ParticipanteService***



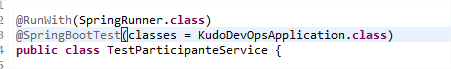
1. Para ello creamos un clase en el paquete Test desde la cual crearemos los métodos que usaremos pára probar la clase ***ParticipanteService***



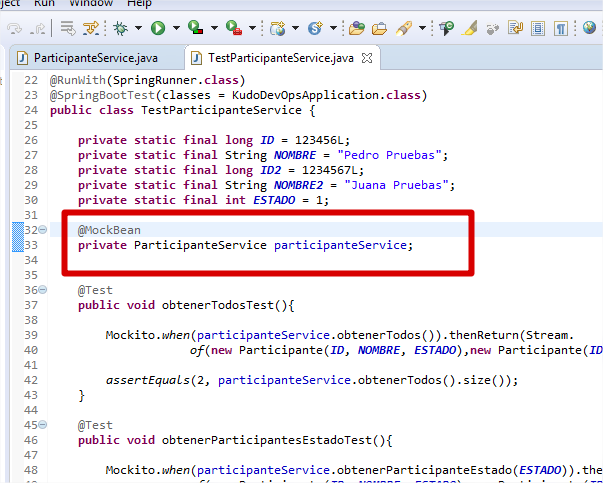
1. Importamos las librerías necesarias



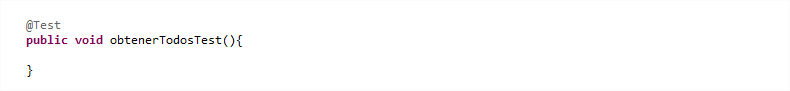
1. Se le asignan las anotaciones pertinentes a la clase que va lanzar los Tests (@*RunWith*) y la clase por la cual va iniciar la aplicación (@*SpringBootTest*)



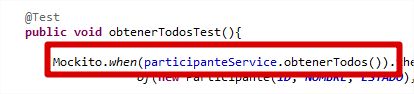
1. Un **Mock** es la imitación del comportamiento de un objeto, es decir un objeto de prueba con el cual podemos simular una respuesta esperada, a continuación crearemos el mock de la clase ***ParticipanteService*** para probar sus operaciones usando la anotación (@*MockBean*)



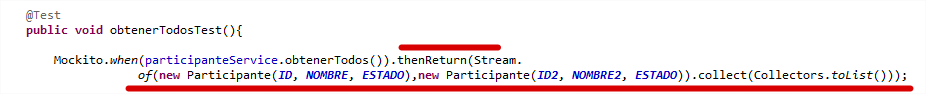
1. Creamos el método que usaremos para probar y se le pondrá la anotación (@*Test*) para que al momento de ejecutar la aplicación esta sepa que ese es un método que prueba alguna funcionalidad



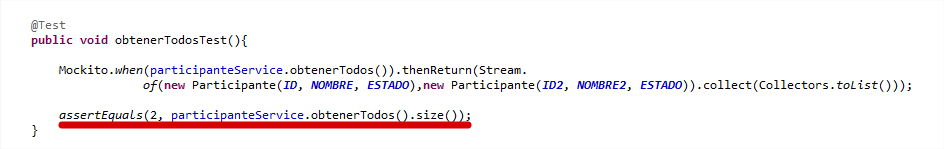
1. Crearemos el Mock indicándole que funcionalidad vamos a probar, en este caso sería la operación **obtenerTodos**() de la clase ***ParticipanteService*** que habíamos definido como MockBean previamente.



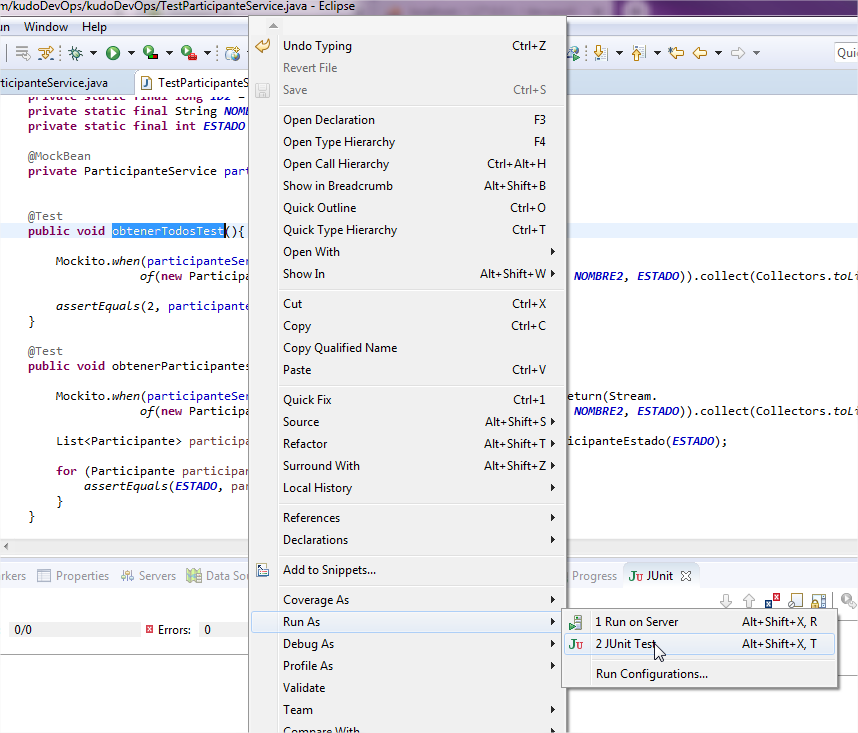
1. Le añadiremos al Mock la respuesta que queremos obtener cuando sea llamada la operación **obtenerTodos**() de la clase ***ParticipanteService***, esta respuesta se agrega despues del método ***thenReturn*** del mockl. que para este caso es una lista de dos participantes.

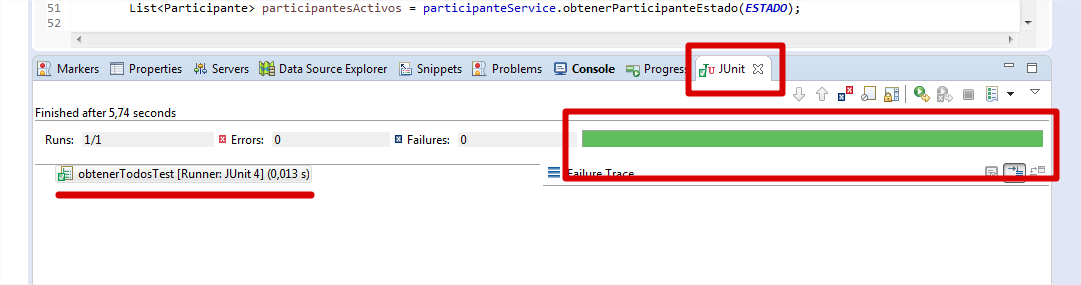


1. Usamos el método ***assertEquals*** para hacer el llamado a la operación **obtenerTodos**() de la clase ***ParticipanteService*** y validar el tamaño de la lista retornada en base a la respuesta que esperamos obtener que en este caso es “2” ya que en el paso anterior le dijimos al Mock que debería retornar una lista de 2 participantes

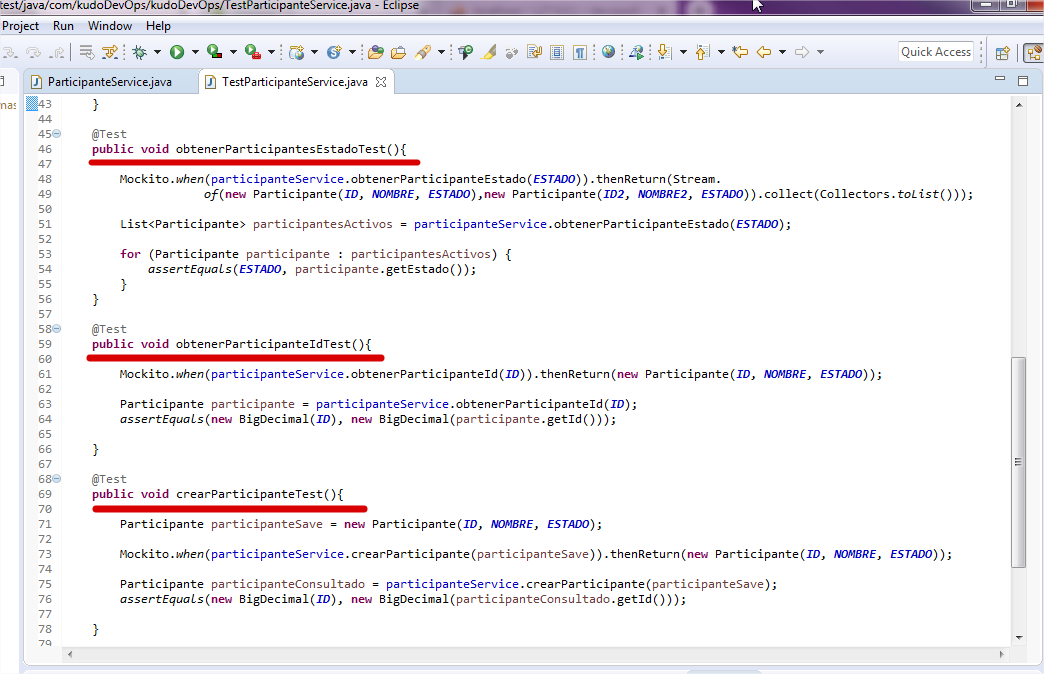


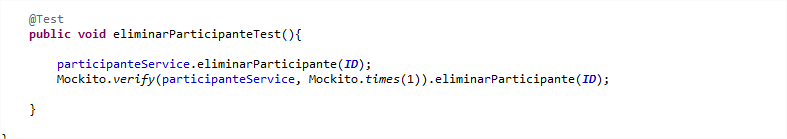
1. Seleccionamos el método que creamos, le damos clic derecho y vamos a la opción “Run As >> JUnit Test”



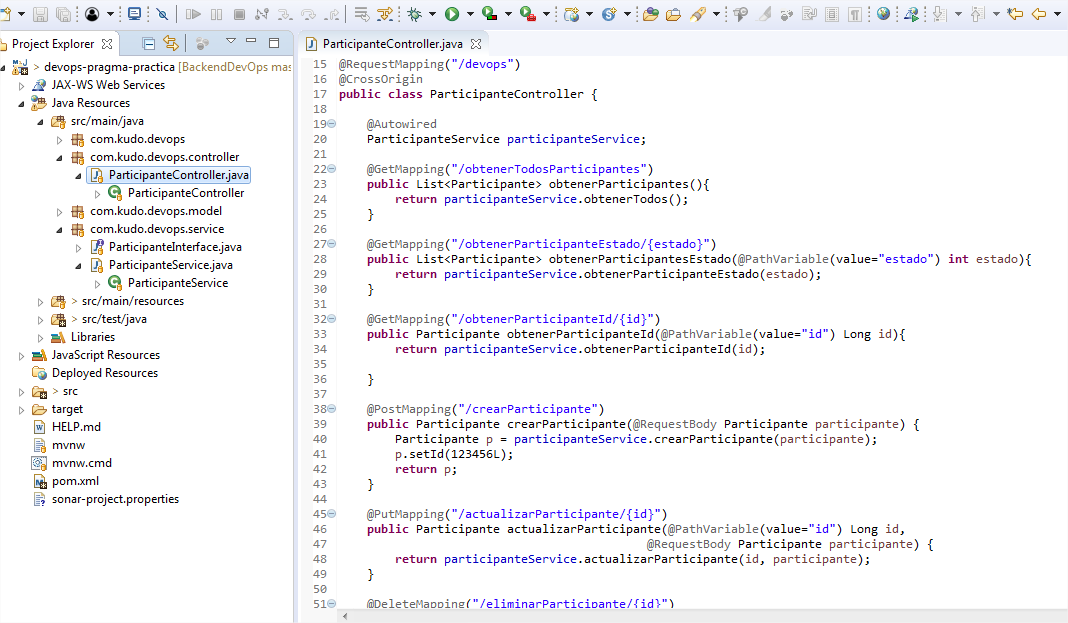
1. En el tab de JUnit veremos el resultado del test que para este caso es exitoso según vemos la barra verde y el check de la consola

1. Realizamos los mismos pasos para el resto de las operaciones

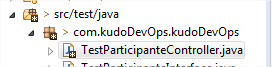




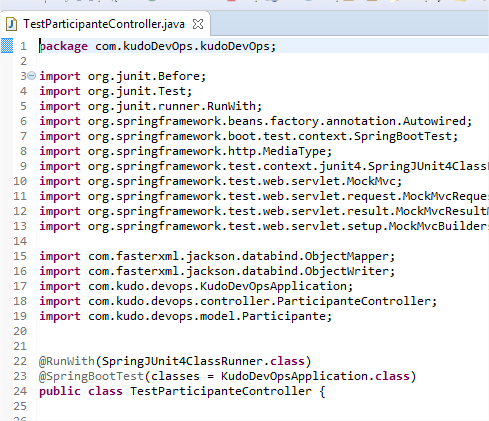
1. **Ahora usaremos Mocks para probar nuestro Controller**



1. Creamos la clase ***TestParticipanteController***



1. Importamos las librerías necesarias y se definen las anotaciones iniciales (@*RunWith*) (@*SpringBootTest*)



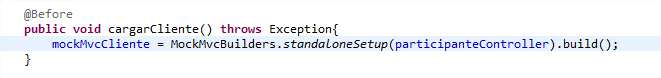
1. Declaramos el objeto que va emular el cliente con el cual vamos probar el consumo del servicio usando la clase **MockMvc** que nos proporciona Spring

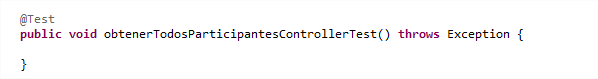
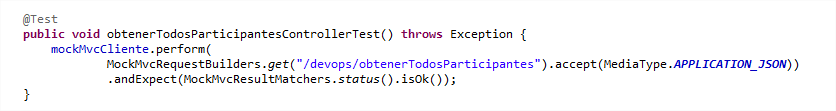


1. Inyectamos la dependencia de nuestra clase ***ParticipanteController***



1. Antes de probar las operaciones del controlador se hace necesario que el cliente se haya construido para lo cual hacemos uso de la anotación (@*Before*) y de la clase ***MockMvcBuilders***



1. Creamos el método Test que usaremos para probar la operación del API Rest desde el controller
2. Usando el objeto cliente (*mockMvcCliente*) que habíamos declarado. construimos el llamado a la operación de nuestro controller, indicando el método por el cual vamos a consumir la operación ***MockMvcRequestBuilders.get***, el tipo de contenido ***.accept (MediaType APPLICATION\_JSON)*** que vamos a enviar en el consumo y la respuesta que esperamos obtener del consumo ***.andExpect(MockMvcResultMatchers.status().isOk()***
3. Realizamos el mismo procedimiento para el resto de las operaciones

