

“Año del Bicentenario, de la consolidación de nuestra Independencia, y de la conmemoración de las heroicas batallas de Junín y Ayacucho”.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, DECANA DE AMÉRICA)
FACULTAD DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE



“Pruebas Funcionales”

Trabajo del curso de Verificación y Validación

Presenta los estudiantes:

Chuquispuma Merino, Fabricio Vidal
Fernandez Camacho Geomar Willy
Landeo Cuentas, Sebastian Alonso
Mirano Surquislla, Fiorella Patricia
Saavedra Monterrey Max Bruno
Neira Caquin Rogger Kevin
Ore Paredes Gianfranco

DOCENTE:

Edgar Sarmiento Calisaya

LIMA-PERÚ

a) Herramientas usadas

JUnit4, Mockito y Maven

b) Descripción

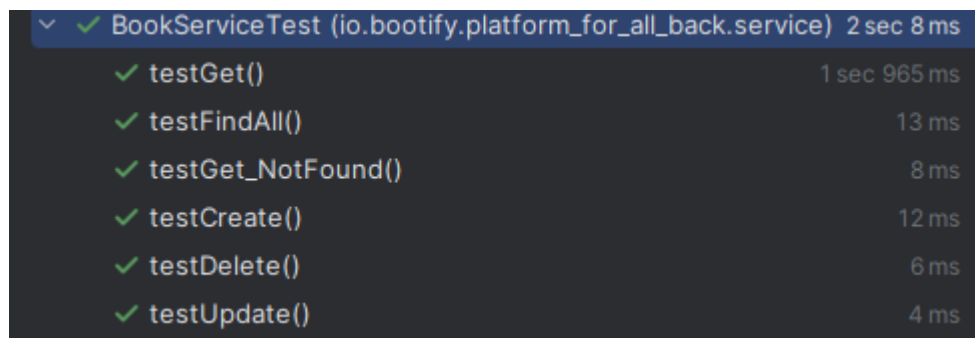
Usar JUnit y Mockito juntos permite crear pruebas unitarias sólidas y aisladas, lo que mejora la calidad del software al garantizar que cada componente funcione de manera correcta e independiente. JUnit proporciona la estructura y herramientas para definir, ejecutar y validar tests, mientras que Mockito simplifica la simulación de dependencias externas, como bases de datos, servicios web o cualquier objeto complejo. Esto facilita la detección temprana de errores, agiliza el desarrollo al ofrecer retroalimentación rápida y reduce el acoplamiento entre componentes, permitiendo que las pruebas se concentren en la lógica específica del código bajo prueba sin depender de implementaciones reales.

c) Análisis

1. Clase “BookService”

Nombre de la pruebas “BookServiceTest”:

Resultados:

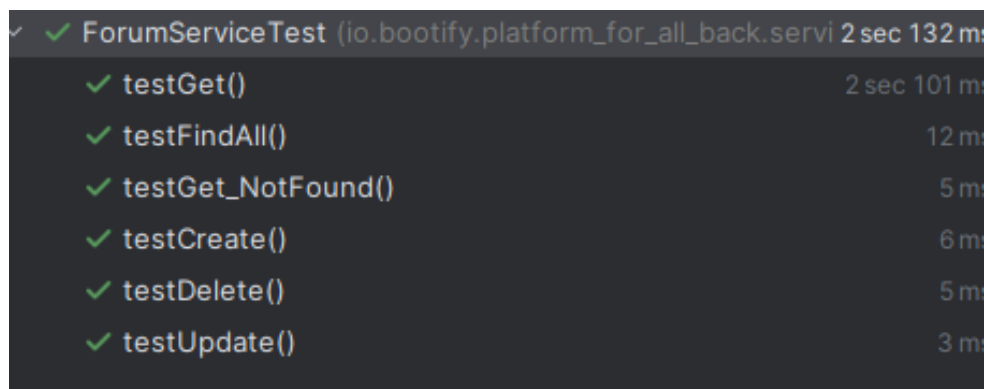
A screenshot of a JUnit test runner interface showing the results for the BookServiceTest class. The class name and package are displayed at the top, followed by the total execution time. Below this, a list of individual test methods is shown, each with a green checkmark indicating success and its corresponding execution time.

✓ BookServiceTest (io.bootify.platform_for_all_back.service)	2 sec 8 ms
✓ testGet()	1 sec 965 ms
✓ testFindAll()	13 ms
✓ testGet_NotFound()	8 ms
✓ testCreate()	12 ms
✓ testDelete()	6 ms
✓ testUpdate()	4 ms

2. Clase “ForumService”

Nombre de la pruebas “ForumServiceTest”:

Resultados:

A screenshot of a JUnit test runner interface showing the results for the ForumServiceTest class. The class name and package are displayed at the top, followed by the total execution time. Below this, a list of individual test methods is shown, each with a green checkmark indicating success and its corresponding execution time.

✓ ForumServiceTest (io.bootify.platform_for_all_back.servi	2 sec 132 ms
✓ testGet()	2 sec 101 ms
✓ testFindAll()	12 ms
✓ testGet_NotFound()	5 ms
✓ testCreate()	6 ms
✓ testDelete()	5 ms
✓ testUpdate()	3 ms

3. Clase “PrimarySequenceService”

Nombre de la pruebas “PrimarySequenceServiceTest”:

Resultados:

✓ ! PrimarySequenceServiceTest (io.bootify.platform_for_all_back.service)	1 sec 634 ms
✓ testGetNextValue_ExistingSequence()	1 sec 597 ms
! testGetNextValue_NewSequence()	27 ms
✓ testGetNextValue_ExceptionHandling()	6 ms
! testGetNextValue_InsertCalledWithCorrectObject_UsingArgumentCaptor()	4 ms

4. Clase “UserService”

Nombre de la pruebas “UserServiceTest”:

Resultados:

✓ UserServiceTest (io.bootify.platform_for_all_back.serv	1 sec 470 ms
✓ testGetUserNotFound()	1 sec 426 ms
✓ testUpdateUserNotFound()	4 ms
✓ testGet()	11 ms
✓ testFindAll()	10 ms
✓ testIdUserDoesNotExist()	3 ms
✓ testCreate()	5 ms
✓ testDelete()	4 ms
✓ testIdUserExists()	4 ms
✓ testUpdate()	3 ms

d) Cobertura

Element ^	Class, %	Method, %	Line, %	Branch, %
✓ io.bootify.platform_for_all_back	100% (4/4)	100% (27/27)	96% (114/118)	50% (1/2)
© BookService	100% (1/1)	100% (8/8)	100% (34/34)	100% (0/0)
© ForumService	100% (1/1)	100% (8/8)	100% (34/34)	100% (0/0)
© PrimarySequenceService	100% (1/1)	100% (2/2)	66% (8/12)	50% (1/2)
© UserService	100% (1/1)	100% (9/9)	100% (38/38)	100% (0/0)

e) Cosas a corregir

En PrimarySequenceServiceTest

Error 1:

org.mockito.exceptions.base.MockitoException:

Only void methods can doNothing()!

Example of correct use of doNothing():

```
doNothing().  
doThrow(new RuntimeException())  
.when(mock).someVoidMethod();
```

Above means:

someVoidMethod() does nothing the 1st time but throws an exception the 2nd time is called

```
at  
io.bootify.platform_for_all_back.service.PrimarySequenceServiceTest.testGetNextValue_  
NewSequence(PrimarySequenceServiceTest.java:60)  
at java.base/java.lang.reflect.Method.invoke(Method.java:568)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

Error 2:

org.mockito.exceptions.base.MockitoException:

Only void methods can doNothing()!

Example of correct use of doNothing():

```
doNothing().  
doThrow(new RuntimeException())  
.when(mock).someVoidMethod();
```

Above means:

someVoidMethod() does nothing the 1st time but throws an exception the 2nd time is called

```
at  
io.bootify.platform_for_all_back.service.PrimarySequenceServiceTest.testGetNextValue_  
insertCalledWithCorrectObject_UsingArgumentCaptor(PrimarySequenceServiceTest.java:  
81)  
at java.base/java.lang.reflect.Method.invoke(Method.java:568)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```