

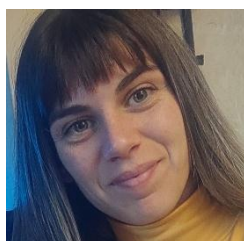
Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO 1

ALGORITMOS Y ESTRUCTURAS DE DATOS



Pablo Larnaudie – 340181



Natalia Rebella – 327283

Grupo: N3C

Docente: Hugo Cepeda

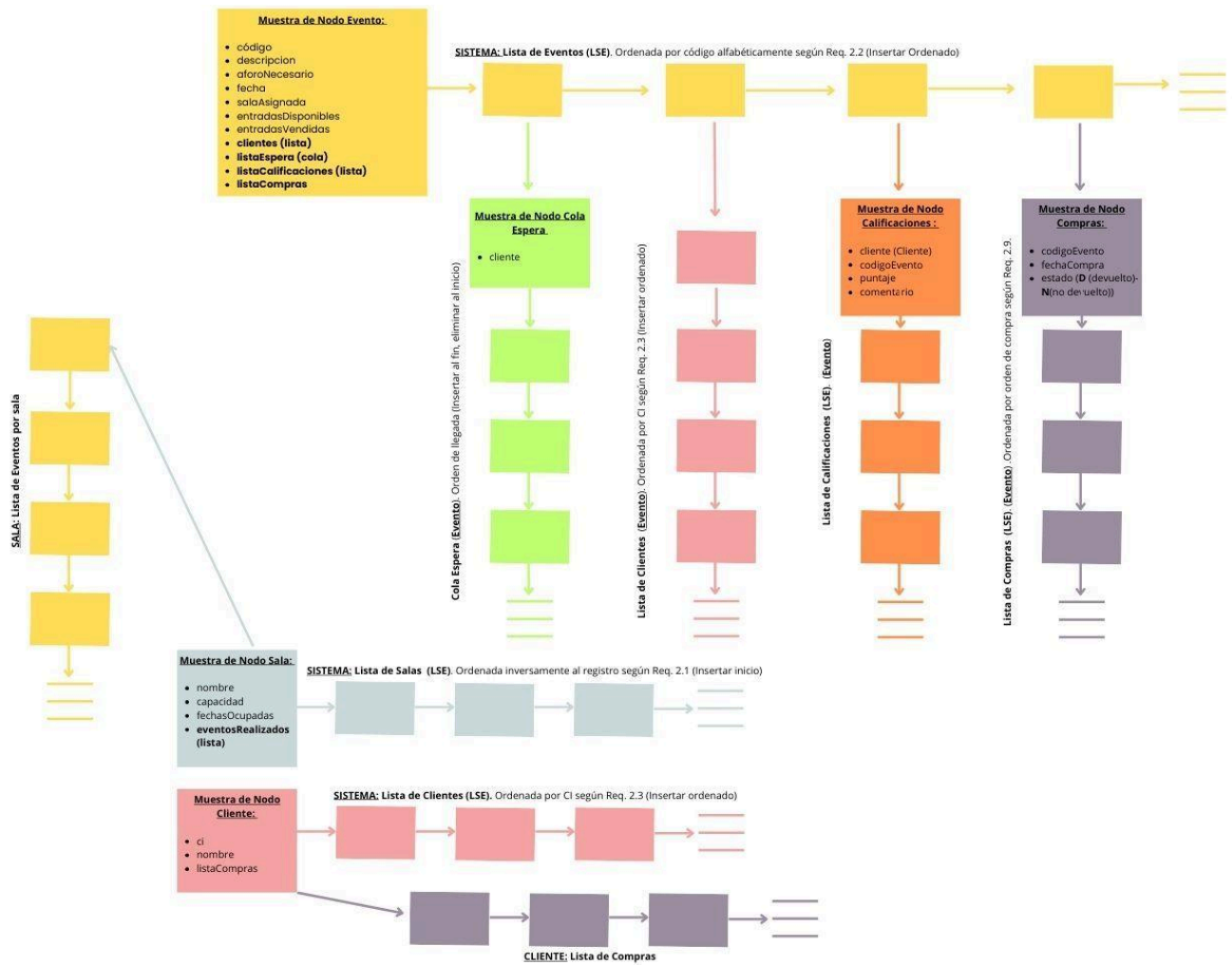
Analista en Tecnologías de la Información

Fecha de entrega del documento (12-05-2025)

Índice

1. Representación gráfica del sistema de gestión a resolver.....	3
2. Juego de pruebas, testeo y resultados.....	4
1.1. TESTEO REQUERIMIENTO 1.1.....	4
1.2. TESTEO REQUERIMIENTO 1.2 – OK.....	4
1.3. TESTEO REQUERIMIENTO 1.2 – ERROR 1.....	5
1.4. TESTEO REQUERIMIENTO 1.2 – ERROR 2.....	6
1.5. TESTEO REQUERIMIENTO 1.3 – OK.....	7
1.6. TESTEO REQUERIMIENTO 1.3 – ERROR 1.....	8
1.7. TESTEO REQUERIMIENTO 1.4 - OK.....	9
1.8. TESTEO REQUERIMIENTO 1.4 – ERROR 1.....	10
1.9. TESTEO REQUERIMIENTO 1.4 – ERROR 2.....	12
1.10. TESTEO REQUERIMIENTO 1.4 –ERROR 3.....	12
1.11. TESTEO REQUERIMIENTO 1.5 – OK.....	14
1.12. TESTEO REQUERIMIENTO 1.5 – ERROR 1.....	15
1.13. TESTEO REQUERIMIENTO 1.5 – ERROR 2.....	17
1.14. TESTEO REQUERIMIENTO 2.1.....	18
1.15. TESTEO REQUERIMIENTO 2.2.....	19
1.16. TESTEO REQUERIMIENTO 2.3.....	20
1.17. TESTEO REQUERIMIENTO 2.4.....	21

1. Representación gráfica del sistema de gestión a resolver

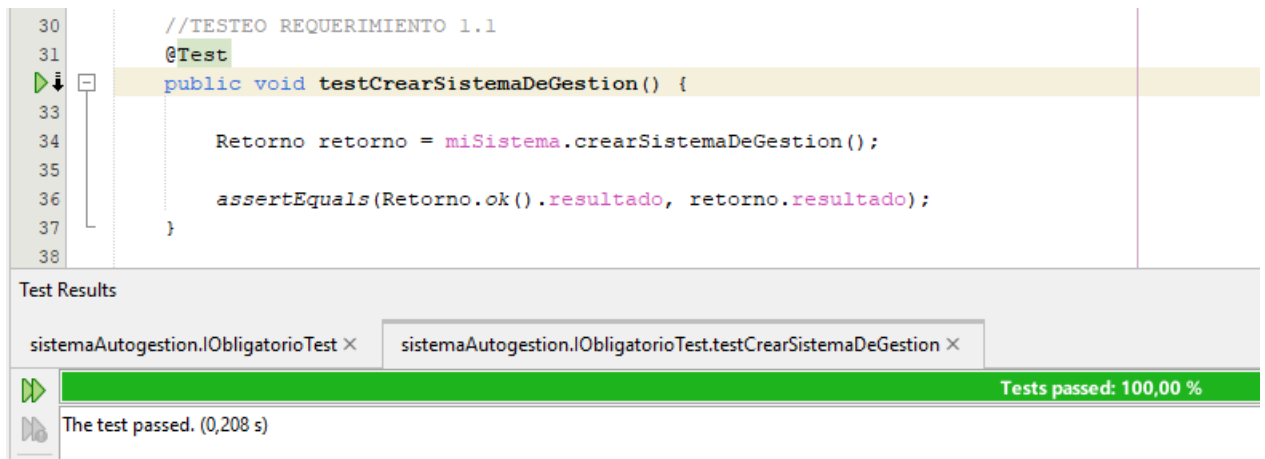


2. Juego de pruebas, testeo y resultados

1.1. TESTEO REQUERIMIENTO 1.1

@Test

```
public void testCrearSistemaDeGestion() {  
    Retorno retorno = miSistema.crearSistemaDeGestion();  
    assertEquals(Retorno.ok().resultado, retorno.resultado);  
}
```



1.2. TESTEO REQUERIMIENTO 1.2 – OK

@Test

```
public void testRegistrarSalaOK() {  
  
    Retorno retorno = miSistema.registrarSala("Sala Aire", 80);  
  
    assertEquals(Retorno.ok().resultado, retorno.resultado);  
  
    retorno = miSistema.registrarSala("Sala Fuego", 150);  
  
    assertEquals(Retorno.ok().resultado, retorno.resultado);  
}
```

```

    retorno = miSistema.registrarSala("Sala Agua", 200);

    assertEquals(Retorno.ok().resultado, retorno.resultado);

    retorno = miSistema.registrarSala("Sala Tierra", 100);

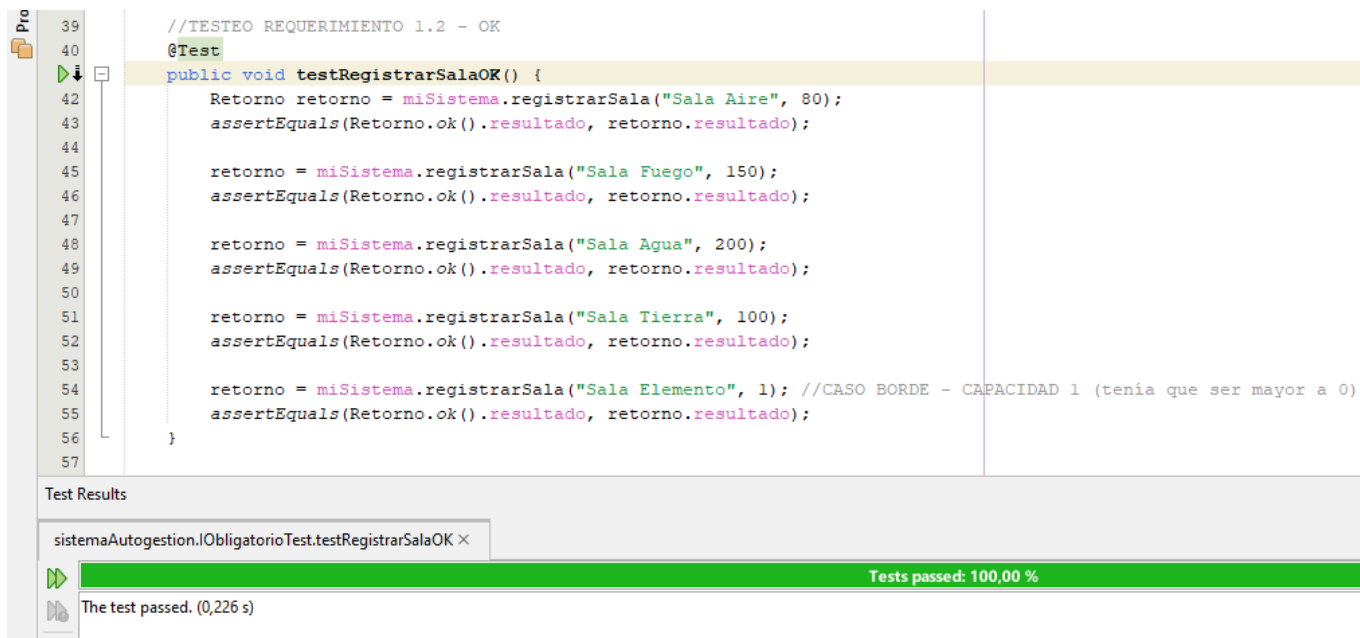
    assertEquals(Retorno.ok().resultado, retorno.resultado);

    retorno = miSistema.registrarSala("Sala Elemento", 1); //CASO BORDE -
CAPACIDAD 1 (tenía que ser mayor a 0)

    assertEquals(Retorno.ok().resultado, retorno.resultado);

}

```



1.3. TESTEO REQUERIMIENTO 1.2 – ERROR 1

@Test

```

public void testRegistrarSalaERROR1() {

    miSistema.registrarSala("Sala Aire", 100);

    Retorno retorno = miSistema.registrarSala("Sala Aire", 80); //ERROR POR
IGUAL NOMBRE

```

```

assertEquals(Retorno.error1().resultado, retorno.resultado);

retorno = miSistema.registrarSala("sala aire", 200);

assertEquals(Retorno.error1().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Aire ", 60);

assertEquals(Retorno.error1().resultado, retorno.resultado);

retorno = miSistema.registrarSala("sALA aIRE", 150);

assertEquals(Retorno.error1().resultado, retorno.resultado);

}

```

The screenshot shows an IDE window with a Java test method named `testRegistrarSalaERROR1()`. The code includes several calls to `miSistema.registrarSala()` with different parameters and `assertEquals()` assertions. Below the code, the 'Test Results' panel shows that the test passed successfully. The test runner window displays 'Tests passed: 100,00 %' and 'The test passed. (0,238 s)'.

```

81  @Test
82  public void testRegistrarSalaERROR1() {
83
84      miSistema.registrarSala("Sala Aire", 100);
85
86      Retorno retorno = miSistema.registrarSala("Sala Aire", 80); //ERROR POR IGUAL NOMBRE
87      assertEquals(Retorno.error1().resultado, retorno.resultado);
88
89      retorno = miSistema.registrarSala("sala aire", 200);
90      assertEquals(Retorno.error1().resultado, retorno.resultado);
91
92      retorno = miSistema.registrarSala("Sala Aire ", 60);
93      assertEquals(Retorno.error1().resultado, retorno.resultado);
94
95      retorno = miSistema.registrarSala("sALA aIRE", 150);
96      assertEquals(Retorno.error1().resultado, retorno.resultado);
97  }
98
99  //TESTEO REQUERIMIENTO 1.2 - ERROR2 - CAPACIDAD ES MENOR O IGUAL A 0

```

Test Results

sistemaAutogestion.IObligatorioTest.testRegistrarSalaERROR1 ×

Tests passed: 100,00 %

The test passed. (0,238 s)

1.4. TESTEO REQUERIMIENTO 1.2 – ERROR 2

@Test

```
public void testRegistrarSalaERROR2() {
```

```

Retorno retorno = miSistema.registrarSala("Sala Aire", 0);

assertEquals(Retorno.error2().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Fuego ", -1);

assertEquals(Retorno.error2().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Viento ", -2);

assertEquals(Retorno.error2().resultado, retorno.resultado);

}

```

The screenshot shows an IDE window with a Java test method named `testRegistrarSalaERROR2()`. The code is as follows:

```

@Test
public void testRegistrarSalaERROR2() {

    Retorno retorno = miSistema.registrarSala("Sala Aire", 0);
    assertEquals(Retorno.error2().resultado, retorno.resultado);

    retorno = miSistema.registrarSala("Sala Fuego ", -1);
    assertEquals(Retorno.error2().resultado, retorno.resultado);

    retorno = miSistema.registrarSala("Sala Viento ", -2);
    assertEquals(Retorno.error2().resultado, retorno.resultado);

}

```

Below the code editor, the 'Test Results' panel shows the test execution details:

- Test Name: `sistemaAutogestion.IObligatorioTest.testRegistrarSalaERROR2`
- Status: **Tests passed: 100,00 %**
- Message: **The test passed. (0,254 s)**

1.5. TESTEO REQUERIMIENTO 1.3 – OK

@Test

```
public void testEliminarSalaOK() {
```

```
//Completar para primera entrega
```

```
Retorno retorno = miSistema.registrarSala("Sala Espacio", 500);
```

```
assertEquals(Retorno.ok().resultado, retorno.resultado);
```

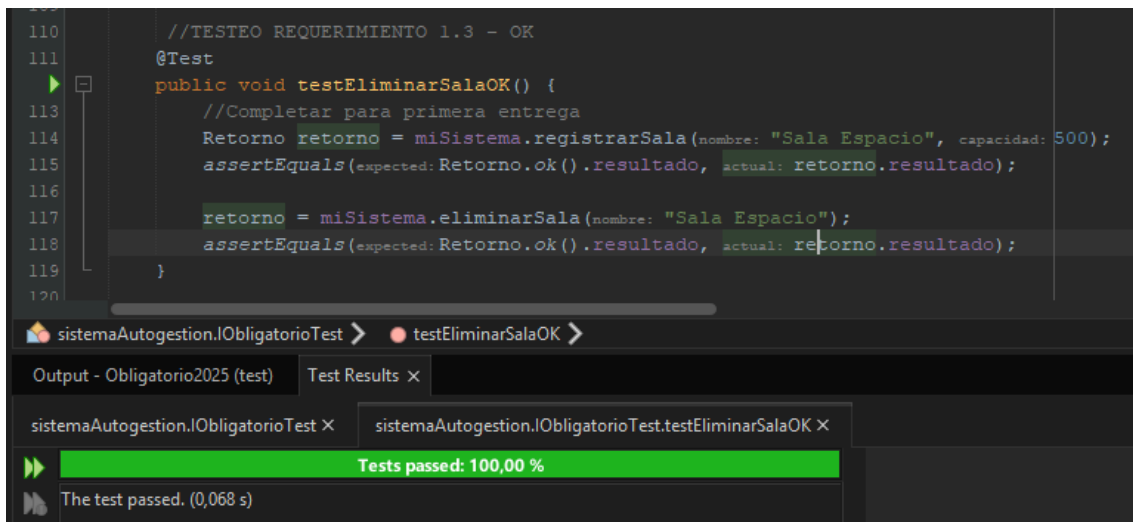
```

    retorno = miSistema.eliminarSala("Sala Espacio");

    assertEquals(Retorno.ok().resultado, retorno.resultado);

}

```



1.6. TESTEO REQUERIMIENTO 1.3 – ERROR 1

@Test

```

public void testEliminarSalaERROR1() {

    //Completar para primera entrega

    Retorno retorno = miSistema.registrarSala("Sala Zitarrosa", 500);

    assertEquals(Retorno.ok().resultado, retorno.resultado);

    retorno = miSistema.eliminarSala("Sala Espacio");

    assertEquals(Retorno.error1().resultado, retorno.resultado);

}

```


1.7. TESTEO REQUERIMIENTO 1.4 - OK

@Test

```
public void testRegistrarEventoOK() {

    miSistema.registrarSala("Sala Viento", 100);

    LocalDate fecha = LocalDate.of(2025, 5, 15);

    Retorno retorno = miSistema.registrarEvento("AAQ23", "Noche de gala", 50,
fecha);

    assertEquals(Retorno.ok().resultado, retorno.resultado);

    LocalDate fecha1 = LocalDate.of(2025, 4, 20);

    retorno = miSistema.registrarEvento("ABC12", "Baile gitano", 1, fecha1);

    assertEquals(Retorno.ok().resultado, retorno.resultado);

    LocalDate fecha2 = LocalDate.of(2025, 3, 13);

    retorno = miSistema.registrarEvento("XYZ34", "Bingo", 90, fecha2);

    assertEquals(Retorno.ok().resultado, retorno.resultado);

    LocalDate fecha3 = LocalDate.of(2025, 2, 3);

    retorno = miSistema.registrarEvento("LMN567", "Reunión empresarial", 30,
fecha3);

    assertEquals(Retorno.ok().resultado, retorno.resultado);

}
```

```
126 @Test
127 public void testRegistrarEventoOK() {
128
129     miSistema.registrarSala("Sala Viento", 100);
130
131     LocalDate fecha = LocalDate.of(2025, 5, 15);
132     Retorno retorno = miSistema.registrarEvento("AAQ23", "Noche de gala", 50, fecha);
133     assertEquals(Retorno.ok().resultado, retorno.resultado);
134
135     LocalDate fechal = LocalDate.of(2025, 4, 20);
136     retorno = miSistema.registrarEvento("ABC12", "Baile gitano", 1, fechal);
137     assertEquals(Retorno.ok().resultado, retorno.resultado);
138
139     LocalDate fecha2 = LocalDate.of(2025, 3, 13);
140     retorno = miSistema.registrarEvento("XYZ34", "Bingo", 90, fecha2);
141     assertEquals(Retorno.ok().resultado, retorno.resultado);
142
143     LocalDate fecha3 = LocalDate.of(2025, 2, 3);
144     retorno = miSistema.registrarEvento("LMN567", "Reunión empresarial", 30, fecha3);
145     assertEquals(Retorno.ok().resultado, retorno.resultado);
146 }
```

Test Results

sistemaAutogestion.I.ObligatorioTest.testRegistrarEventoOK x

Tests passed: 100,00 %

The test passed. (0,218 s)

1.8. TESTEO REQUERIMIENTO 1.4 – ERROR 1

@Test

```
public void testRegistrarEventoERROR1() {

    miSistema.registrarSala("Sala Baile", 200);

    LocalDate fecha1 = LocalDate.of(2025, 6, 15);

    LocalDate fecha2 = LocalDate.of(2026, 6, 15);

    LocalDate fecha5 = LocalDate.of(2025, 11, 15);

    LocalDate fecha6 = LocalDate.of(2026, 3, 15);

    LocalDate fecha7 = LocalDate.of(2025, 2, 13);

    miSistema.registrarEvento("AAQ23", "Bailamos", 90, fecha1);

    miSistema.registrarEvento("ZZZ12", "Lectura compartida", 100, fecha2);
```

```
Retorno retorno = miSistema.registrarEvento("AAQ23", "Noche de gala", 50,
fecha5);
```

```
assertEquals(Retorno.error1().resultado, retorno.resultado);
```

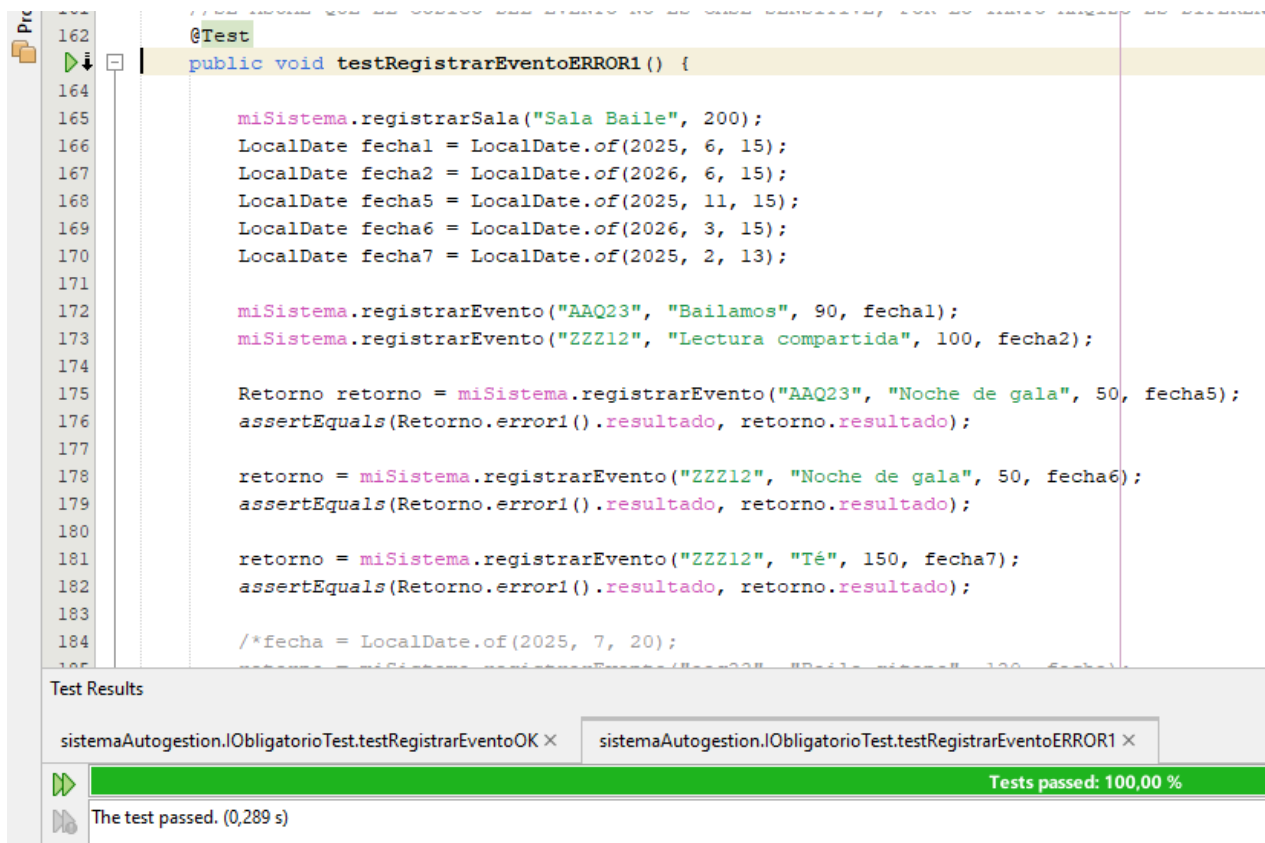
```
retorno = miSistema.registrarEvento("ZZZ12", "Noche de gala", 50, fecha6);
```

```
assertEquals(Retorno.error1().resultado, retorno.resultado);
```

```
retorno = miSistema.registrarEvento("ZZZ12", "Té", 150, fecha7);
```

```
assertEquals(Retorno.error1().resultado, retorno.resultado);
```

```
}
```



The screenshot displays a code editor with a Java test method named `testRegistrarEventoERROR1`. The method includes several date assignments and calls to `miSistema.registrarEvento` and `miSistema.registrarSala`. Below the code, the 'Test Results' panel shows that the test `sistemaAutogestion.IObligatorioTest.testRegistrarEventoERROR1` passed successfully. A green progress bar indicates 'Tests passed: 100,00 %' and a message states 'The test passed. (0,289 s)'.

```
162  @Test
163  public void testRegistrarEventoERROR1 () {
164
165      miSistema.registrarSala("Sala Baile", 200);
166      LocalDate fecha1 = LocalDate.of(2025, 6, 15);
167      LocalDate fecha2 = LocalDate.of(2026, 6, 15);
168      LocalDate fecha5 = LocalDate.of(2025, 11, 15);
169      LocalDate fecha6 = LocalDate.of(2026, 3, 15);
170      LocalDate fecha7 = LocalDate.of(2025, 2, 13);
171
172      miSistema.registrarEvento("AAQ23", "Bailamos", 90, fecha1);
173      miSistema.registrarEvento("ZZZ12", "Lectura compartida", 100, fecha2);
174
175      Retorno retorno = miSistema.registrarEvento("AAQ23", "Noche de gala", 50, fecha5);
176      assertEquals(Retorno.error1().resultado, retorno.resultado);
177
178      retorno = miSistema.registrarEvento("ZZZ12", "Noche de gala", 50, fecha6);
179      assertEquals(Retorno.error1().resultado, retorno.resultado);
180
181      retorno = miSistema.registrarEvento("ZZZ12", "Té", 150, fecha7);
182      assertEquals(Retorno.error1().resultado, retorno.resultado);
183
184      /*fecha = LocalDate.of(2025, 7, 20);
185      retorno = miSistema.registrarEvento("AAQ23", "Baile nico", 120, fecha1);
```

Test Results

sistemaAutogestion.IObligatorioTest.testRegistrarEventoOK × sistemaAutogestion.IObligatorioTest.testRegistrarEventoERROR1 ×

Tests passed: 100,00 %

The test passed. (0,289 s)

1.9. TESTEO REQUERIMIENTO 1.4 – ERROR 2

@Test

```
public void testRegistrarEventoERROR2() {

    miSistema.registrarSala("Sala Baile", 200);

    LocalDate fecha5 = LocalDate.of(2025, 11, 15);

    LocalDate fecha6 = LocalDate.of(2026, 3, 15);

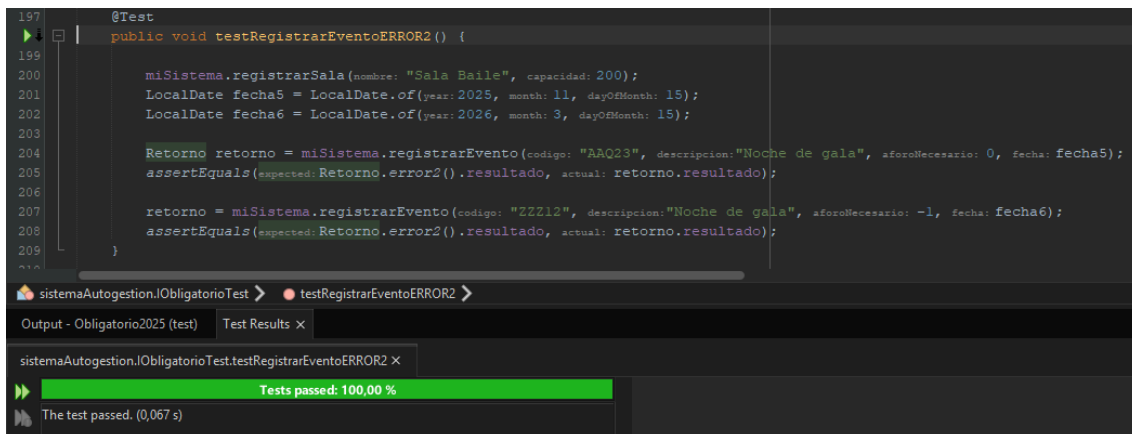
    Retorno retorno = miSistema.registrarEvento("AAQ23", "Noche de gala", 0,
fecha5);

    assertEquals(Retorno.error2().resultado, retorno.resultado);

    retorno = miSistema.registrarEvento("ZZZ12", "Noche de gala", -1, fecha6);

    assertEquals(Retorno.error2().resultado, retorno.resultado);

}
```



1.10. TESTEO REQUERIMIENTO 1.4 –ERROR 3

@Test

```
public void testRegistrarEventoERROR3() {

    miSistema.registrarSala("Sala Eventos", 100);
```

```

        LocalDate fecha1 = LocalDate.of(2025, 12, 15);

        LocalDate fecha2 = LocalDate.of(2026, 12, 15);

        Retorno retorno1 = miSistema.registrarEvento("AAQ23", "Noche de gala", 100,
fecha1);

        assertEquals(Retorno.ok().resultado, retorno1.resultado);

        //misma fecha

        Retorno retorno2 = miSistema.registrarEvento("AB7DG", "Evento", 90, fecha1);

        assertEquals(Retorno.error3().resultado, retorno2.resultado);

        //aforo insuficiente fecha disponible

        Retorno retorno3 = miSistema.registrarEvento("A456SD", "Fiesta", 101, fecha2);

        assertEquals(Retorno.error3().resultado, retorno3.resultado);

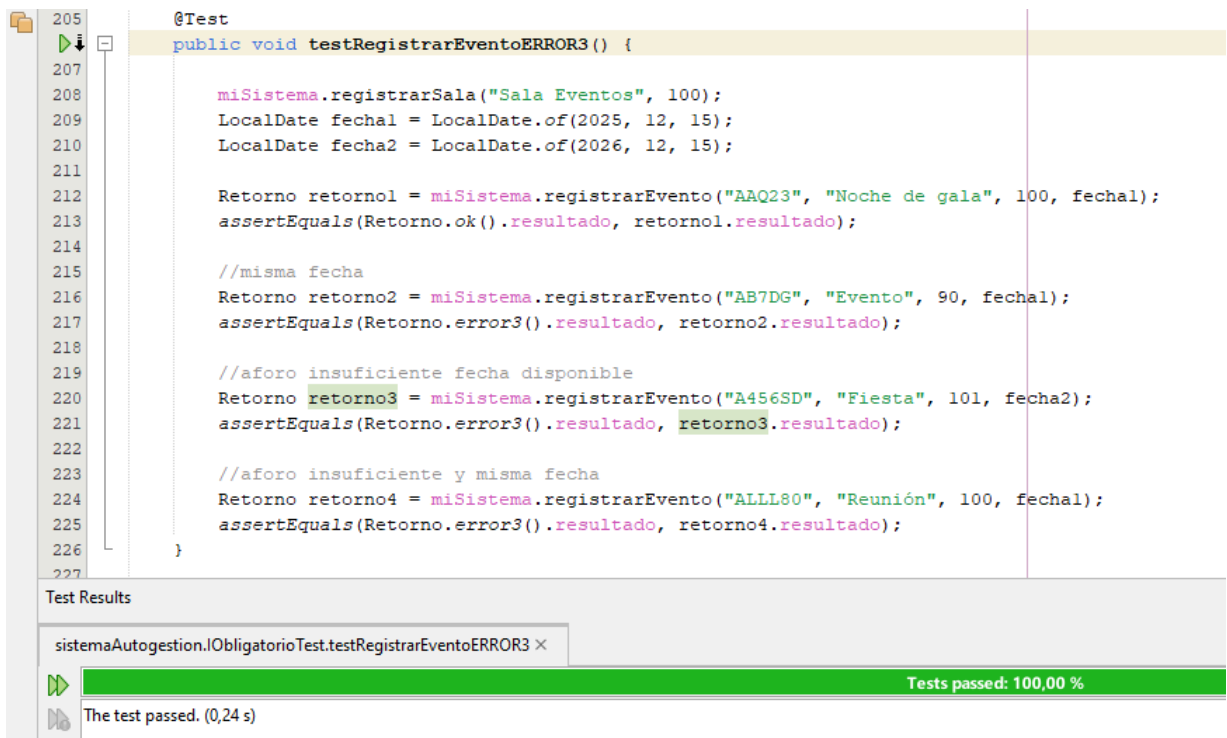
        //aforo insuficiente y misma fecha

        Retorno retorno4 = miSistema.registrarEvento("ALLL80", "Reunión", 100,
fecha1);

        assertEquals(Retorno.error3().resultado, retorno4.resultado);

    }

```



1.11. TESTEO REQUERIMIENTO 1.5 – OK

@Test

```
public void testRegistrarClienteOK() {
```

```
    //Completar para primera entrega
```

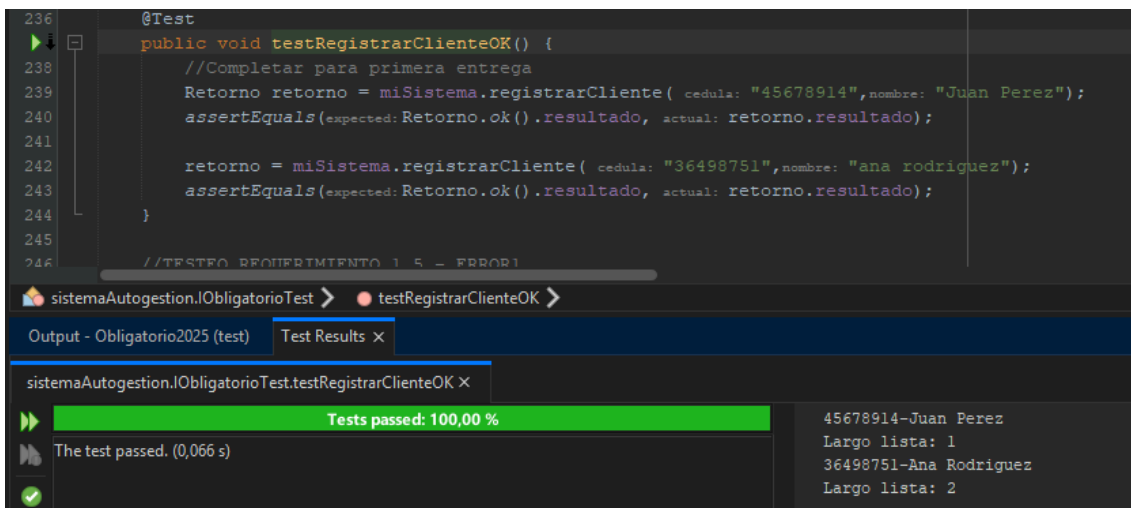
```
    Retorno retorno = miSistema.registrarCliente( "45678914","Juan Perez");
```

```
    assertEquals(Retorno.ok().resultado, retorno.resultado);
```

```
    retorno = miSistema.registrarCliente( "36498751","ana rodriguez");
```

```
    assertEquals(Retorno.ok().resultado, retorno.resultado);
```

```
}
```



1.12. TESTEO REQUERIMIENTO 1.5 – ERROR 1

@Test

```
public void testRegistrarClienteERROR1() {
```

```
    //Completar para primera entrega
```

```
    //Test con 9 digitos
```

```
    Retorno retorno = miSistema.registrarCliente( "456789147", "Julio Cesar");
```

```
    assertEquals(Retorno.error1().resultado, retorno.resultado);
```

```
    //Test con 7 digitos
```

```
    retorno = miSistema.registrarCliente( "4567891", "Marco Aurelio");
```

```
    assertEquals(Retorno.error1().resultado, retorno.resultado);
```

```
    //Test con 8 letras
```

```
    retorno = miSistema.registrarCliente( "asdfghjk", "Cleopatra Nilo");
```

```
    assertEquals(Retorno.error1().resultado, retorno.resultado);
```

```
    //Test con letras y numeros
```

```
    retorno = miSistema.registrarCliente( "1234ghjk", "Alejandro Magno");
```

```

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con simbolos

retorno = miSistema.registrarCliente( "!;?¿#@_-", "Pompeyo Magno");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con simbolos y letras

retorno = miSistema.registrarCliente( "!@#abcde", "Trajano Romano");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con simbolos y numeros

retorno = miSistema.registrarCliente( "!1234?86", "Marcus Ulpius Traianus");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con simbolos, numeros y letras

retorno = miSistema.registrarCliente( "!1234abc", "Escipion El Africano");

assertEquals(Retorno.error1().resultado, retorno.resultado);

}

```



```
248 @Test
249 public void testRegistrarClienteERROR1() {
250     //Completar para primera entrega
251     //Test con 9 digitos
252     Retorno retorno = miSistema.registrarCliente( cedula: "456789147",nombre: "Julio Cesar");
253     assertEquals(expected: Retorno.error1().resultado, actual: retorno.resultado);
254     //Test con 7 digitos
255     retorno = miSistema.registrarCliente( cedula: "4567891",nombre: "Marco Aurelio");
256     assertEquals(expected: Retorno.error1().resultado, actual: retorno.resultado);
257     //Test con 8 letras
258     retorno = miSistema.registrarCliente( cedula: "asdfghjk",nombre: "Cleopatra Nilo");
259     assertEquals(expected: Retorno.error1().resultado, actual: retorno.resultado);
260     //Test con letras y numeros
261     retorno = miSistema.registrarCliente( cedula: "1234ghjk",nombre: "Alejandro Magno");
262     assertEquals(expected: Retorno.error1().resultado, actual: retorno.resultado);
263     //Test con simbolos
264     retorno = miSistema.registrarCliente( cedula: "!;?_#@_-", nombre: "Pompeyo Magno");
265     assertEquals(expected: Retorno.error1().resultado, actual: retorno.resultado);
266     //Test con simbolos y letras
267     retorno = miSistema.registrarCliente( cedula: "!@#abcde", nombre: "Trajano Romano");
268     assertEquals(expected: Retorno.error1().resultado, actual: retorno.resultado);
269     //Test con simbolos y numeros
270     retorno = miSistema.registrarCliente( cedula: "!1234?86", nombre: "Marcus Ulpius Traianus");
271     assertEquals(expected: Retorno.error1().resultado, actual: retorno.resultado);
272     //Test con simbolos, numeros y letras
273     retorno = miSistema.registrarCliente( cedula: "!1234abc", nombre: "Escipion El Africano");
274     assertEquals(expected: Retorno.error1().resultado, actual: retorno.resultado);
275 }
276
```

sistemaAutogestion.IObligatorioTest > testRegistrarClienteERROR1 >

Output - Obligatorio2025 (test) Test Results x

sistemaAutogestion.IObligatorioTest.testRegistrarClienteERROR1 x

Tests passed: 100,00 %

The test passed. (0,065 s)

1.13. TESTEO REQUERIMIENTO 1.5 – ERROR 2

@Test

```
public void testRegistrarClienteERROR2() {
```

```
    //Completar para primera entrega
```

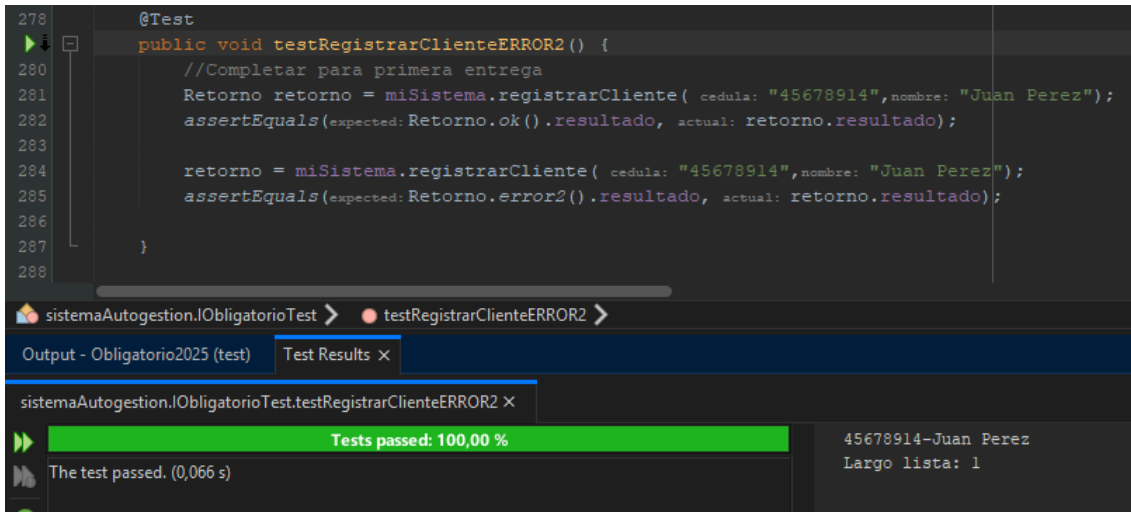
```
    Retorno retorno = miSistema.registrarCliente( "45678914","Juan Perez");
```

```
    assertEquals(Retorno.ok().resultado, retorno.resultado);
```

```
    retorno = miSistema.registrarCliente( "45678914","Juan Perez");
```

```
    assertEquals(Retorno.error2().resultado, retorno.resultado);
```

}



```
278     @Test
279     public void testRegistrarClienteERROR2() {
280         //Completar para primera entrega
281         Retorno retorno = miSistema.registrarCliente( cedula: "45678914", nombre: "Juan Perez");
282         assertEquals(expected: Retorno.ok().resultado, actual: retorno.resultado);
283
284         retorno = miSistema.registrarCliente( cedula: "45678914", nombre: "Juan Perez");
285         assertEquals(expected: Retorno.error2().resultado, actual: retorno.resultado);
286
287     }
288 }
```

sistemaAutogestion.IObligatorioTest > testRegistrarClienteERROR2 >

Output - Obligatorio2025 (test) Test Results x

sistemaAutogestion.IObligatorioTest.testRegistrarClienteERROR2 x

Tests passed: 100,00 %

The test passed. (0,066 s)

45678914-Juan Perez
Largo lista: 1

1.14. TESTEO REQUERIMIENTO 2.1

@Test

```
public void testListarSalas() {
```

```
    //Completar para primera entrega
```

```
    miSistema.registrarSala("Sala Aire", 80);
```

```
    miSistema.registrarSala("Sala Fuego", 150);
```

```
    miSistema.registrarSala("Sala Agua", 200);
```

```
    miSistema.registrarSala("Sala Tierra", 100);
```

```
    miSistema.registrarSala("Sala Elemento", 1); //CASO BORDE - CAPACIDAD 1
    (tenía que ser mayor a 0)
```

```
    Retorno retorno = miSistema.listarSalas();
```

```
    assertEquals(retorno.ok().resultado, retorno.resultado);
```

```
String esperado = "sala elemento-1#sala tierra-100#sala agua-200#sala
fuego-150#sala aire-80#";
```

```
assertEquals(esperado, retorno.valorString);
```

```
miSistema.listarSalas();
```

```
}
```

The screenshot shows an IDE with a Java test file. The test method `testListarSalas()` registers five rooms: Sala Aire (80), Sala Fuego (150), Sala Agua (200), Sala Tierra (100), and Sala Elemento (1). It then calls `miSistema.listarSalas()` and asserts that the returned string matches the expected string: "sala elemento-1#sala tierra-100#sala agua-200#sala fuego-150#sala aire-80#". Below the code, the test results are shown as passed.

```

290  @Test
291  public void testListarSalas() {
292      //Completar para primera entrega
293      miSistema.registrarSala(nombre: "Sala Aire", capacidad: 80);
294      miSistema.registrarSala(nombre: "Sala Fuego", capacidad: 150);
295      miSistema.registrarSala(nombre: "Sala Agua", capacidad: 200);
296      miSistema.registrarSala(nombre: "Sala Tierra", capacidad: 100);
297      miSistema.registrarSala(nombre: "Sala Elemento", capacidad: 1); //CASO BORDE - CAPACIDAD 1 (tenia que ser mayor a 0)
298
299      Retorno retorno = miSistema.listarSalas();
300      assertEquals(expected: retorno.ok().resultado, actual: retorno.resultado);
301
302      String esperado = "sala elemento-1#sala tierra-100#sala agua-200#sala fuego-150#sala aire-80#";
303      assertEquals(expected: esperado, actual: retorno.valorString);
304      miSistema.listarSalas();
305  }
306

```

Test Results

Tests passed: 100,00 %

The test passed. (0,066 s)

1.15. TESTEO REQUERIMIENTO 2.2

```
@Test
```

```
public void testListarEventos() {
```

```
miSistema.registrarSala("Sala Aire", 220);
```

```
LocalDate fecha1 = LocalDate.of(2025, 12, 15);
```

```
miSistema.registrarEvento("AAQ23", "Noche de gala", 100, fecha1);
```

```
LocalDate fecha2 = LocalDate.of(2024, 12, 15);
```

```
miSistema.registrarEvento("AZD456", "Tango feliz", 150, fecha2);
```

```
LocalDate fecha3 = LocalDate.of(2023, 12, 15);
```

```
miSistema.registrarEvento("AZK765", "Lectura", 200, fecha3);
```

```
Retorno retorno = miSistema.listarEventos();
```

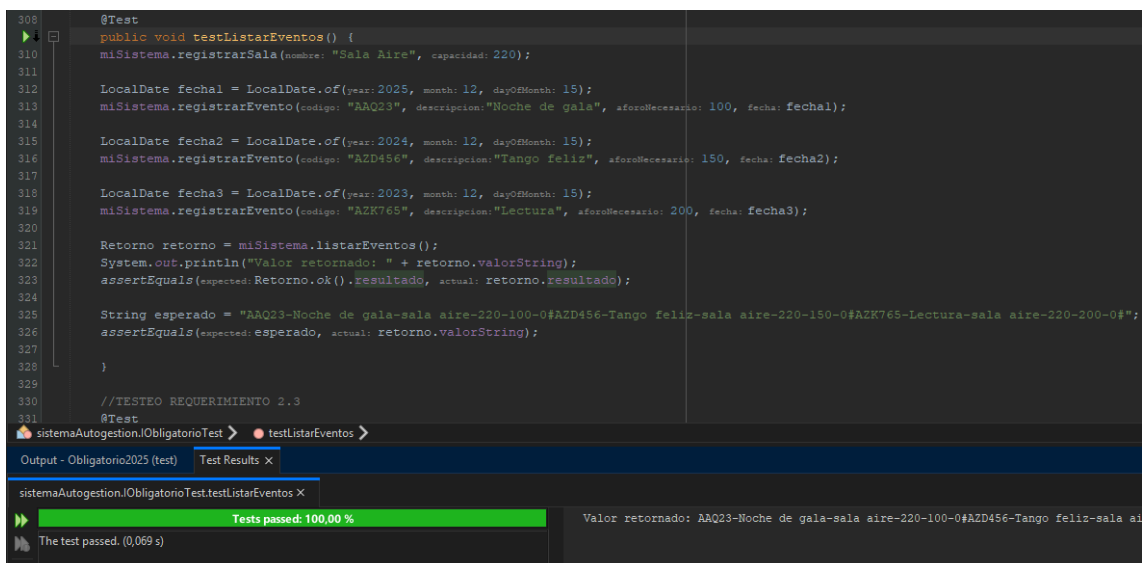
```
System.out.println("Valor retornado: " + retorno.valorString);
```

```
assertEquals(Retorno.ok().resultado, retorno.resultado);
```

```
String esperado = "AAQ23-Noche de gala-sala aire-220-100-0#AZD456-Tango  
feliz-sala aire-220-150-0#AZK765-Lectura-sala aire-220-200-0#";
```

```
assertEquals(esperado, retorno.valorString);
```

```
}
```



The screenshot shows an IDE with a Java test method `testListarEventos()` and its execution results. The test method is as follows:

```
309 @Test
310 public void testListarEventos() {
311     miSistema.registrarSala(nombre: "Sala Aire", capacidad: 220);
312
313     LocalDate fecha1 = LocalDate.of(year: 2023, month: 12, dayOfMonth: 15);
314     miSistema.registrarEvento(codigo: "AAQ23", descripcion: "Noche de gala", aforoNecesario: 100, fecha: fecha1);
315
316     LocalDate fecha2 = LocalDate.of(year: 2024, month: 12, dayOfMonth: 15);
317     miSistema.registrarEvento(codigo: "AZD456", descripcion: "Tango feliz", aforoNecesario: 150, fecha: fecha2);
318
319     LocalDate fecha3 = LocalDate.of(year: 2023, month: 12, dayOfMonth: 15);
320     miSistema.registrarEvento(codigo: "AZK765", descripcion: "Lectura", aforoNecesario: 200, fecha: fecha3);
321
322     Retorno retorno = miSistema.listarEventos();
323     System.out.println("Valor retornado: " + retorno.valorString);
324     assertEquals(expected: Retorno.ok().resultado, actual: retorno.resultado);
325
326     String esperado = "AAQ23-Noche de gala-sala aire-220-100-0#AZD456-Tango feliz-sala aire-220-150-0#AZK765-Lectura-sala aire-220-200-0#";
327     assertEquals(expected: esperado, actual: retorno.valorString);
328 }
329
330 //TESTEO REQUERIMIENTO 2.3
331 @Test
```

The execution results show that the test passed successfully. The output is:

```
Output - Obligatorio2025 (test) Test Results X
sistemaAutogestion.ObligatorioTest.testListarEventos X
Tests passed: 100.00 %
The test passed. (0.069 s)
Valor retornado: AAQ23-Noche de gala-sala aire-220-100-0#AZD456-Tango feliz-sala ai
```

1.16. TESTEO REQUERIMIENTO 2.3

@Test

```
public void testListarClientes() {
```

```
//Completar para primera entrega
```

```

miSistema.registrarCliente( "45678913","Julio Cesar");

miSistema.registrarCliente( "35678982","Marco Aurelio");

miSistema.registrarCliente( "15678965","Alejandro Magno");

miSistema.registrarCliente( "55678956", "Pompeyo Magno");

miSistema.registrarCliente( "49267892", "Escipion El Africano");

Retorno retorno = miSistema.listarClientes();

assertEquals(retorno.ok().resultado, retorno.resultado);

String esperado = "15678965-Alejandro Magno#35678982-Marco
Aurelio#45678913-Julio Cesar#49267892-Escipion El Africano#55678956-Pompeyo
Magno#";

assertEquals(esperado, retorno.valorString);

}

```

```

331  @Test
332  public void testListarClientes() {
333      //Completar para primera entrega
334
335      miSistema.registrarCliente( "45678913","Julio Cesar");
336      miSistema.registrarCliente( "35678982","Marco Aurelio");
337      miSistema.registrarCliente( "15678965","Alejandro Magno");
338      miSistema.registrarCliente( "55678956", "Pompeyo Magno");
339      miSistema.registrarCliente( "49267892", "Escipion El Africano");
340
341      Retorno retorno = miSistema.listarClientes();
342      assertEquals(esperado, retorno.ok().resultado, actual: retorno.resultado);
343      String esperado = "15678965-Alejandro Magno#35678982-Marco Aurelio#45678913-Julio Cesar#49267892-Escipion El Africano#55678956-Pompeyo Magno#";
344      assertEquals(esperado, retorno.valorString);
345  }
346

```

sistemaAutogestion.ObligatorioTest > testListarClientes >
 Output - Obligatorio2025 (test) Test Results X
 sistemaAutogestion.ObligatorioTest.testListarEventos X sistemaAutogestion.ObligatorioTest.testListarClientes X
 Tests passed: 100.00 %
 The test passed. (0.065 s)

1.17. TESTEO REQUERIMIENTO 2.4

@Override

```

public Retorno esSalaOptima(String[][] vistaSala) {

if (vistaSala == null || vistaSala.length == 0 || vistaSala[0].length == 0) {

Retorno ret = (Retorno.ok());

```

```

        ret.valorString = "No se ingresó por parametros valores validos";

        return ret;
    }

    int filas = vistaSala.length;

    int columnas = vistaSala[0].length;

    int columnasOptimas = 0;

    for (int col = 0; col < columnas; col++) {

        int maxOcupadosConsecutivos = 0;

        int actualesOcupados = 0;

        int asientosLibres = 0;

        for (int fila = 0; fila < filas; fila++) {

            String asiento = vistaSala[fila][col];

            if (asiento.equals("O")) {

                actualesOcupados++;

                maxOcupadosConsecutivos = Math.max(maxOcupadosConsecutivos,
actualesOcupados);

            } else {

                actualesOcupados = 0; // rompemos la secuencia
            }

            if (asiento.equals("X")) {

                asientosLibres++;

            }
        }
    }

```

```
    }

    if (maxOcupadosConsecutivos > asientosLibres) {

        columnasOptimas++;

    }

}

if (columnasOptimas >= 2) {

    Retorno ret = (Retorno.ok());

    ret.valorString = "Es optimo";

    return ret;

} else {

    Retorno ret = (Retorno.ok());

    ret.valorString = "No es optimo";

    return ret;

}

}
```

```
347 //TESTEO REQUERIMIENTO 2.4
348 @Test
349 public void testEsSalaOptima() {
350     String[][] vistaSala = {
351         {"#", "#", "#", "#", "#", "#", "#"},
352         {"#", "#", "X", "X", "X", "X", "#"},
353         {"#", "O", "O", "X", "X", "X", "#"},
354         {"#", "O", "O", "O", "O", "X", "#"},
355         {"#", "O", "O", "X", "O", "O", "#"},
356         {"#", "O", "O", "O", "O", "O", "#"},
357         {"#", "X", "X", "O", "O", "O", "O"},
358         {"#", "X", "X", "O", "O", "O", "X"},
359         {"#", "X", "X", "O", "X", "X", "#"},
360         {"#", "X", "X", "O", "X", "X", "#"},
361         {"#", "#", "#", "O", "#", "#", "#"},
362         {"#", "#", "#", "O", "#", "#", "#"}
363     };
364
365     Retorno ret = miSistema.esSalaOptima(vistaSala);
366     assertEquals(expected: Retorno.Resultado.OK, actual: ret.resultado);
367     assertEquals(expected: "Es optimo", actual: ret.valorString);
368 }
369
370 }
```

sistemaAutogestion.IObligatorioTest > testEsSalaOptima >

Output - Obligatorio2025 (test) Test Results ×

sistemaAutogestion.IObligatorioTest.testEsSalaOptima ×

Tests passed: 100,00 %

The test passed. (0,064 s)