

Testing & Maintenance



Test and maintenance report of the
provided code

INTRODUCTION	3
TESTING PLAN	4
class GestionMesas	4
Testing values:	4
method definirServicio()	4
Condition/Decision coverage	5
class GestorReserva	7
TESTING CODE	7
class gestorMesa	7
method definirServicio()	7
method definirEstado()	9
method getStyleClass()	11
method isEnabled()	14
class GestorReservas	16
method realizarReserva()	16
COVERAGE REPORT	18
MAINTENANCE	19
Preventive maintenance	19
PMD	19
Checkstyle	19
Corrective Maintenance	21
CPD	21
FindBugs	21

1. INTRODUCTION

First of all, for each method we have created a table with the following columns:

- **CLASS:** the name of the class where we are currently. It indicates that the methods in the next field belongs to that specific class.
- **METHOD:** current method that is going to be evaluated in order to make the tests.
- **PARAMETER:** input parameters that has some interest for us in the evaluation of the method.
- **VALUES:** values that takes the designated parameter.
- **BOUNDARIES VALUES:** values that might have an impact on the program.
- **ERROR GUESSING:** values that can work inappropriately in the system.

We have omitted the columns Class and Methods in the images of the tables as they are especificated in each index but they remains in the Test Design file that can be accessed via the wiki in our github repository.

We have selected the **each use** procedure while making the tests for every method in order to have a good coverage.

About GestionMesas, getStyleClass and isEnabled both of them have conditions inside, so we have decided to make also a **condition/decision** coverage in order to test every behavior those conditions can have.

A few lines in some of the methods have been commented in order to be able for the test to be executed. Those lines belongs to the persistency package as they are calling to a ServicioDao object's method or a ReservaDao object's method:

- class GestionMesas:
 - line 75: servicioActivo = servicioDao.update(servicioActivo);
 - line 89: servicioActivo = servicioDao.update(servicioActivo);
- class GestionReservas:
 - line 71: reservaDao.persist(reserva);
 - line72: reservas = reservaDao.findAll();

We have also make some modification in the methods of the class GestionMesas so we can assign values easily and test it in a faster way.

What we have done here is define a new method called setUltimoEstadoMesa which allows us to set the vector of UltimoEstadoMesa with the values we need in each case so we can have one or other evaluation, from the specific condition that uses that vector, accomplished to make the correct test.

Finally, a new branch in github has been created under the name “Testing” in order to upload the Unitary test cases once we have finished them.

2. TESTING PLAN

a. class GestionMesas

i. Testing values:

method definirServicio()

PARAMETERS	EQUIVALENCE CLASSES	VALUES	BOUNDARIES	Error Guessing
fechaServicio	Date	"29/05/2019"		
comidaServicio	ComidaEnum.COMIDA	TRUE		
	ComidaEnum.CENA	FALSE		
turnoServicio	TurnoEnum.TURNO_1	0		
	TurnoEnum.TURNO_2	1		
	TurnoEnum.TURNO_3	2		
	Any other value	10		-5

method definirEstado()

mesald	(-inf, 0]	-15	0	
	values in MesaEnum [1,4]	4		
	[5, inf)	20	5	
estadold	(-inf, 0)	-10		
	values in EstadoEnum [0,8]	7		
	[9, inf)	10		

method getClassStyle()

mesald	(-inf, 0]	-20	0	
	values in MesaEnum [1,4]	2		
	[5, inf)	15	5	
estadold	(-inf, 0)	-5		
	[0]	0		
	[1, inf]	5		
ultimoEstadoMesa[mesald - 1]	(-inf, 0)	-10	0	
	[0,8)	5		
	[8, inf)	10	8	

method isEnabled()

mesald	(-inf, 0]	-2	1	
	values in MesaEnum [1,4]	1		
	[5, inf)	6	5	
estadold	(-inf, 0)	-5		
	[0]	0		
	[1, inf)	5		
ultimoEstadoMesa[mesald - 1]	(-inf, 0)	-10	0	
	[0, 8)	5		
	[8, inf)	10	8	
servicioActivo	Servicio	A valid Servicio object		
	not Servicio	null		

ii. Condition/Decision coverage

method getStyleClass()

<i>EstadoEnum.values().length - 1 == ultimoEstadoMesa[mesald - 1] && estadold==0</i>		
A	B	A and B
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE
<i>estadold > ultimoEstadoMesa[mesald - 1]</i>		
A		
VERDADERO		
FALSO		
<i>estadold < ultimoEstadoMesa[mesald - 1]</i>		
A		
VERDADERO		
FALSO		

Those are the values that has been taken for each of the selected combinations(the green ones in the table) in order to accomplish the condition/decision coverage for the getStyleClass() method.

Condition:

EstadoEnum.values().length - 1 == ultimoEstadoMesa[mesaId - 1] && estadoId == 0

mesald	estadold
A mesaId that has a value of 8 in ultimoEstadoMesa	0
A mesaId that has not a value of 8 in ultimoEstadoMesa	6

method: isEnabled()

<i>EstadoEnum.values().length - 1 == ultimoEstadoMesa[mesald - 1] && estadoId == 0</i>		
A	B	A and B
VERDADERO	VERDADERO	VERDADERO
VERDADERO	FALSO	FALSO
FALSO	VERDADERO	FALSO
FALSO	FALSO	FALSO
<i>return servicioActivo != null && estadoId >= ultimoEstadoMesa[mesald - 1]</i>		
A	B	A and B
VERDADERO	VERDADERO	VERDADERO
VERDADERO	FALSO	FALSO
FALSO	VERDADERO	FALSO
FALSO	FALSO	FALSO

Those are the values that has been taken for each of the selected combinations(the green ones in the table) in order to accomplish the condition/decision coverage for the isEnabled() method.

Condition:

EstadoEnum.values().length - 1 == ultimoEstadoMesa[mesald - 1] && estadoId == 0

mesald	estadoId	ultimoEstadoMesa[mesald - 1]
value that makes ultimoEstadoMesa[mesald - 1] = 8	0	8
value that makes ultimoEstadoMesa[mesald - 1] = 3	1	3

Condition:

return servicioActivo != null && estadoId >= ultimoEstadoMesa[mesald - 1]

mesald	servicioActivo	estadoId	ultimoEstadoMesa[mesald - 1]
value that makes ultimoEstadoMesa[mesald - 1] = 3	servicioActivo	5	3
value that makes ultimoEstadoMesa[mesald - 1] = 5	null	1	5

b. class GestorReserva

i. TESTING VALUES

method realizarReserva()

fechaReserva	A valid Date	29/05/19		null
comidaReserva	ComidaEnum.COMIDA	TRUE		
	ComidaEnum.CENA	FALSE		
turnoReserva	TurnoEnum.TURNO_1	0		
	TurnoEnum.TURNO_2	1		
	TurnoEnum.TURNO_3	2		
	other value	20		-5
mesaReserva	(-inf, 0)	-2	0	
	[0, 4]	2		
	(4, inf)	10	4	
nombreReserva	String	A valid String value	"Reserva"	null

3. TESTING CODE

a. class gestorMesa

i. method definirServicio()

Test cases

VALUES			TEST NAME
fechaServicio	comidaServicio	turnoServicio	
29-05-2019	TRUE	0	testDefinirServicio_1
29-05-2019	FALSE	1	testDefinirServicio_2
29-05-2019	TRUE	2	testDefinirServicio_3
null	FALSE	10	testDefinirServicio_4
29-05-2019	TRUE	-5	testDefinirServicio_5

```

public void testDefinirServicio_1() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(true);
    gm.setTurnoServicio(0);

    try {
        gm.definirServicio();
    } catch (Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testDefinirServicio_2() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(false);
    gm.setTurnoServicio(1);

    try {
        gm.definirServicio();
    } catch (Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testDefinirServicio_3() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(true);
    gm.setTurnoServicio(2);

    try {
        gm.definirServicio();
    } catch (Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testDefinirServicio_4() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(false);
    gm.setTurnoServicio(10);

    try {
        gm.definirServicio();
        fail("An error was expected");
    } catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```



```

public void testDefinirServicio_5() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(true);
    gm.setTurnoServicio(-5);

    try {
        gm.definirServicio();
        fail("An error was expected");
    } catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

ii. method definirEstado()

Test cases

mesald	estadold	
-15	-10	testDefinirEstado_1
4	7	testDefinirEstado_2
20	7	testDefinirEstado_3
0	10	testDefinirEstado_4
5	10	testDefinirEstado_5

Take into account:

In this case as we need a servicioActivo for each one, we will create an object which content will be as follows:
-FechaServicio: 29-05-2019
-ComidaServicio: true
-TurnoServicio: 1

```

public void testDefinirEstado_1() throws ParseException {
    int [] vectorEstados = {0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(true);
    gm.setTurnoServicio(1);

    try {
        gm.definirServicio();
        gm.definirEstado(-5,-10);
        fail("An error was expected");
    }catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testDefinirEstado_2() throws ParseException {
    int [] vectorEstados = {0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(true);
    gm.setTurnoServicio(1);

    try {
        gm.definirServicio();
        gm.definirEstado(4,7);
    }catch (Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testDefinirEstado_3() throws ParseException {
    int [] vectorEstados = {0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(true);
    gm.setTurnoServicio(1);

    try {
        gm.definirServicio();
        gm.definirEstado(20,7);
        fail("An error was expected");
    }catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testDefinirEstado_4() throws ParseException {
    int [] vectorEstados = {0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(true);
    gm.setTurnoServicio(1);

    try {
        gm.definirServicio();
        gm.definirEstado(0,10);
        fail("An error was expected");
    }catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testDefinirEstado_5() throws ParseException {
    int [] vectorEstados = {0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gm.setFechaServicio(date);
    gm.setComidaServicio(true);
    gm.setTurnoServicio(1);

    try {
        gm.definirServicio();
        gm.definirEstado(5,10);
        fail("An error was expected");
    }catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

iii. method getStyleClass()

Test cases

mesald	estadold	ultimoEstadoMesa[mesald - 1] *	
-20	-5	{-10,-10,-10,-10,-10,-10,-10,-10}	testGetStyleClass_1
2	0	{0,5,0,0,0,0,0,0}	testGetStyleClass_2
15	5	{10,10,10,10,10,10,10,10}	testGetStyleClass_3
1	-5	{0,0,0,0,0,0,0,0}	testGetStyleClass_4
5	0	{0,0,0,0,8,0,0,0}	testGetStyleClass_5

We have take into account:

In ultimoEstadoMesa if the mesald is not in the range of the ultimoEstadoMesa.length (which is 8) then we fill all the positions in the vector with the correspondent value, otherwise the value is only stored in the position mesald-1

```
public void testGetStyleClass_1() {
    int [] vectorEstados = {-10,-10,-10,-10,-10,-10,-10,-10};
    gm.setUltimoEstadoMesa(vectorEstados);

    try {
        gm.getStyleClass(-20, -5);
        fail("An error was expected");
    }catch(Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}
```

```
public void testGetStyleClass_2() {
    int [] vectorEstados = {0,5,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    try {
        gm.getStyleClass(2, 0);
    }catch(Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}
```

```
public void testGetStyleClass_3() {
    int [] vectorEstados = {10,10,10,10,10,10,10,10};
    gm.setUltimoEstadoMesa(vectorEstados);

    try {
        gm.getStyleClass(15, 5);
        fail("An error was expected");
    }catch(Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}
```



```

public void testGetStyleClass_4() {
    int [] vectorEstados = {0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    try {
        gm.getStyleClass(1, -5);
    } catch (Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testGetStyleClass_5() {
    int [] vectorEstados = {0,0,0,0,8,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    try {
        gm.getStyleClass(5, 0);
    } catch (Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

Those are the ones belonging to the Condition/Decision coverage:

```

public void testGetStyleClass_cd1()
{
    int[] vectorEstados = {0,0,8,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);
    String obtained = gm.getStyleClass(3,0);
    String expected = "estado-futuro";
    assertEquals(obtained, expected);
}

```

```

public void testGetStyleClass_cd2()
{
    int[] vectorEstados = {0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);
    String obtained = gm.getStyleClass(1,6);
    String expected = "estado-futuro";
    assertEquals(obtained, expected);
}

```

```

public void testGetStyleClass_cd3()
{
    int[] vectorEstados = {0,0,0,0,0,0,0,2};
    gm.setUltimoEstadoMesa(vectorEstados);
    String obtained = gm.getStyleClass(8,4);
    String expected = "estado-futuro";
    assertEquals(obtained, expected);
}

```

```

public void testGetStyleClass_cd4()
{
    int[] vectorEstados = {6,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);
    String obtained = gm.getStyleClass(1,3);
    String expected = "estado-pasado";
    assertEquals(obtained, expected);
}

```

```

public void testGetStyleClass_cd5()
{
    int[] vectorEstados = {0,0,0,0,0,0,7,0};
    gm.setUltimoEstadoMesa(vectorEstados);
    String obtained = gm.getStyleClass(6,3);
    String expected = "estado-pasado";
    assertEquals(obtained, expected);
}

```

```

public void testGetStyleClass_cd6()
{
    int[] vectorEstados = {0,0,0,0,1,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);
    String obtained = gm.getStyleClass(5,4);
    String expected = "estado-futuro";
    assertEquals(obtained, expected);
}

```

iv. method isEnabled()

Test cases

mesald	estadold	ultimoEstadoMesa[mesald - 1]	servicioActivo
-2	-5	{-10,-10,-10,-10,-10,-10,-10,-10}	A valid object
1	0	{0,5,0,0,0,0,0,0}	A valid object
6	5	{10,10,10,10,10,10,10,10}	A valid object
1	0	{0,0,0,0,0,0,0,0}	A valid object
5	-5	{0,0,0,0,8,0,0,0}	null

```
public void testIsEnabled_1() {
    int [] vectorEstados = {-10,-10,-10,-10,-10,-10,-10,-10};
    gm.setUltimoEstadoMesa(vectorEstados);

    Servicio servicio = new Servicio();
    gm.setServicioActivo(servicio);

    try {
        gm.isEnabled(-2,-5);
        fail("An error was expected");
    }catch(Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}
```

```
public void testIsEnabled_2() {
    int [] vectorEstados = {0,5,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    Servicio servicio = new Servicio();
    gm.setServicioActivo(servicio);

    try {
        gm.isEnabled(1,0);
    }catch(Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}
```

```

public void testIsEnabled_3() {
    int [] vectorEstados = {10,10,10,10,10,10,10,10};
    gm.setUltimoEstadoMesa(vectorEstados);

    Servicio servicio = new Servicio();
    gm.setServicioActivo(servicio);

    try {
        gm.isEnabled(6,5);
    }catch(Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testIsEnabled_4() {
    int [] vectorEstados = {0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    Servicio servicio = new Servicio();
    gm.setServicioActivo(servicio);

    try {
        gm.isEnabled(1,0);
    }catch(Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testIsEnabled_5() {
    int [] vectorEstados = {0,0,0,0,8,0,0,0};
    gm.setUltimoEstadoMesa(vectorEstados);

    Servicio servicio = new Servicio();
    gm.setServicioActivo(servicio);

    try {
        gm.isEnabled(5,-5);
    }catch(Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

Those are the ones belonging to the Condition/Decision coverage:


```

public void testIsEnabled_cd1() throws Exception {
    int[] ultimoEstadoMesa = {8,0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(ultimoEstadoMesa);

    Servicio servicio = new Servicio();
    gm.setServicioActivo(servicio);

    boolean esperado = true;
    boolean obtenido = gm.isEnabled(1, 0);
    assertEquals(esperado, obtenido);
}

```

```

public void testIsEnabled_cd3() throws Exception {
    int[] ultimoEstadoMesa = {3,0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(ultimoEstadoMesa);

    Servicio servicio = new Servicio();
    gm.setServicioActivo(servicio);

    boolean esperado = true;
    boolean obtenido = gm.isEnabled(1, 5);
    assertEquals(esperado, obtenido);
}

```

```

public void testIsEnabled_cd2() throws Exception {
    int[] ultimoEstadoMesa = {8,0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(ultimoEstadoMesa);

    Servicio servicio = new Servicio();
    gm.setServicioActivo(servicio);

    boolean esperado = false;
    boolean obtenido = gm.isEnabled(1, 1);
    assertEquals(esperado, obtenido);
}

```

```

public void testIsEnabled_cd4() throws Exception {
    int[] ultimoEstadoMesa = {6,0,0,0,0,0,0,0,0};
    gm.setUltimoEstadoMesa(ultimoEstadoMesa);

    boolean esperado = false;
    boolean obtenido = gm.isEnabled(1, 5);
    assertEquals(esperado, obtenido);
}

```

b. class GestorReservas

i. method realizarReserva()

Test cases

fechaReserva	comidaReserva	turnoReserva	mesaReserva	nombreReserva	
29/05/19	FALSE	0	-2	"Reserva de Daniel"	TestRealizarReserva_1
29/05/19	TRUE	1	2	"Reserva de Raquel"	TestRealizarReserva_2
29/05/19	FALSE	2	10	null	TestRealizarReserva_3
29/05/19	TRUE	20	0	"Reserva de Enrique"	TestRealizarReserva_4
29/05/19	FALSE	-5	4	"Reserva de Samuel"	TestRealizarReserva_5

```

public void testRealizarReserva_1() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gr.setFechaReserva(date);
    gr.setComidaReserva(false);
    gr.setTurnoReserva(0);
    gr.setMesaReserva(-2);
    gr.setNombreReserva("Reserva de Daniel");

    try {
        gr.realizarReserva();
        fail("An error was expected");
    } catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```



```

public void testRealizarReserva_2() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gr.setFechaReserva(date);
    gr.setComidaReserva(true);
    gr.setTurnoReserva(1);
    gr.setMesaReserva(2);
    gr.setNombreReserva("Reserva de Raquel");

    try {
        gr.realizarReserva();
    } catch (Exception e) {
        fail("The error was not expected");
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testRealizarReserva_3() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gr.setFechaReserva(date);
    gr.setComidaReserva(false);
    gr.setTurnoReserva(2);
    gr.setMesaReserva(10);
    gr.setNombreReserva(null);

    try {
        gr.realizarReserva();
        fail("An error was expected");
    } catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testRealizarReserva_4() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gr.setFechaReserva(date);
    gr.setComidaReserva(true);
    gr.setTurnoReserva(20);
    gr.setMesaReserva(0);
    gr.setNombreReserva("Reserva de Enrique");

    try {
        gr.realizarReserva();
        fail("An error was expected");
    } catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

```

public void testRealizarReserva_5() throws ParseException {
    SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy");
    Date date = format.parse("29-05-2019");

    gr.setFechaReserva(date);
    gr.setComidaReserva(false);
    gr.setTurnoReserva(-5);
    gr.setMesaReserva(4);
    gr.setNombreReserva("Reserva de Samuel");

    try {
        gr.realizarReserva();
        fail("An error was expected");
    } catch (Exception e) {
        System.out.println(new Throwable().getStackTrace()[0].getMethodName());
        e.printStackTrace();
    }
}

```

4. COVERAGE REPORT

EclEmma

As we have some troubles, which we were not able to find a final solution for them, with the JaCoCo report and also with the Surefire Report, we decided to use EclEmma.

EclEmma is a Eclipse plugin that brings code coverage analysis directly into the Eclipse workbench [\[1\]](#)

Gestion_Mesas (04-jun-2019 16:01:23)				
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▲ Gestion_Mesas	62,2 %	1.737	1.056	2.793
▷ src/test/java	80,5 %	1.187	288	1.475
▲ src/main/java	41,7 %	550	768	1.318
▷ es.esi.cr.iso.gestionmesas	0,0 %	0	45	45
▷ es.esi.cr.iso.gestionmesas.config	0,0 %	0	36	36
▷ es.esi.cr.iso.gestionmesas.controller	67,0 %	236	116	352
▷ es.esi.cr.iso.gestionmesas.model	40,7 %	70	102	172
▷ es.esi.cr.iso.gestionmesas.model.enums	54,0 %	244	208	452
▷ es.esi.cr.iso.gestionmesas.persistence	0,0 %	0	261	261

The coverage shown in Gestion_Mesas (the hole project) is not the real coverage. We have to look at the src/main/java folder, where the total coverage that we have reached is 41.7%, but the folder in which we have focused is controller, where we have a 67%.

We have made the test cases based in the classes of that folder, GestorMesas and GestorReservas as we thought they were interesting classes to take into account in the project.

▲ es.esi.cr.iso.gestionmesas.controller	67,0 %	236	116	352
▷ GestorMesas.java	72,3 %	170	65	235
▷ GestorReservas.java	56,4 %	66	51	117

5. MAINTENANCE

A new branch in the github repository has been created under the name of 'Maintenance' and every change or fix done will be uploaded there.

All the images shown in this report are from the branch Testing (which are all the reports before any maintenance changes is being applied) so in order to check differences between the changes done, it is advisable to check reports from Testing branch and reports from Maintenance.

We are going to explain for every task done which was the procedure applied.

a. Preventive maintenance

i. PMD

Some kind of violations that are shown in the PMD report are due to unused or not initialized variables, or having several return statements in a method instead of having only one, or change the name of the variables that have a short name on it.

Also, other typical errors are related with the Law of Demeter, but we are not going to cover those, as well as the ones related with final.

There are other errors in some classes related with the constructor of that class. In some cases where there is no constructor, it tells you to create it, but when you create it, it says that it is not necessary to do it, due to the compiler generates it, and vice versa. This error is not going to be solved too.

And finally, some errors occurred when a SimpleDateFormat object is declared because it needs a Locale. In order to solve this problem, we have added `java.util.Locale.getDefault()` when we instantiate the object.























```
SimpleDateFormat format = new SimpleDateFormat("dd-mm-yyyy", java.util.Locale.getDefault());
```

ii. Checkstyle

Summary

Files	 Info	 Warnings	 Errors
21	0	0	450

Rules

Category	Rule	Violations	Severity
blocks	LeftCurly 	7	 Error
imports	AvoidStarImport 	2	 Error
	RedundantImport 	2	 Error
	UnusedImports 	3	 Error
javadoc	JavadocMethod  <ul style="list-style-type: none">allowUndeclaredRTE: "true"scope: "public"	101	 Error
	JavadocPackage 	6	 Error
misc	NewlineAtEndOfFile  <ul style="list-style-type: none">lineSeparator: "lf"	6	 Error
	RedundantModifier 	5	 Error
regex	RegexSingleline  <ul style="list-style-type: none">format: "\s+\$"message: "Line has trailing spaces."	103	 Error
	FileTabCharacter  <ul style="list-style-type: none">fileExtensions: "java,xml"	20	 Error
	WhitespaceAfter 	195	 Error

In order to solve the problems related to WhitespaceAfter, LeftCurly and RegexSingleline as they are format errors.

FileTabCharacter is another format error errors appeared because Eclipse uses an auto format for tabs.

As we use other rules, in order to solve them we have created another format.

We have applied the following procedure to create the format profile in Eclipse:

In order to do that, in Eclipse, Preferenes > Java > Code Styles > Formatter, we have created another profile based on the one that we have already. Then, in the tab Indention, we change the Tab Policy form Mixed to Space only and then apply the changes. Then, in the project folder, right click Source > Format and save.

For the ones belonging to the NewlineAtEndOfFile, we just need to add to the specific classes that have that error, a new line at the end of the file.

AvoidStarImport appears when there is an import with a * at the end, for example, `java.util.*`; The way that we have solved this errors was to delete that import and use only the imports that we are going to use. In our case, that error appeared at `import static org.junit.Assert.*`; and we substituted it with `import static org.junit.fail` and `import static org.junit.assertEquals`, that they were the only asserts we were using at that time.

ModifierOrder in GestorMesas and GestorReservas as we put 'final static' instead of 'static final' while doing the PMD.


As its name indicates, RedundantImport refers to the redundant imports. In our case, it showed at the two test classes we have, because we imported the package where the class to test was. The way to solve this bug was to delete those imports.

RedundantModifier errors appeared in the Enum classes, because the constructors of those classes, except ComidaEnum and TurnoEnum, had the private indicator at the beginning of them. To solve those errors, we deleted the private from them.

b. Corrective Maintenance

i. CPD

CPD Results

The following document contains the results of PMD's CPD  5.6.1.

Duplications

CPD found no problems in your source code.

Regarding at the CPD report, no necessary changes has to be done in this section as no code is duplicated in the program.

ii. FindBugs

Once we generate the report, 20 bugs have been found in 8 of the 22 classes:

Summary			
Classes	Bugs	Errors	Missing Classes
22	23	0	0

Files	
Class	Bugs
es.esi.cr.iso.gestionmesas.controller.GestorMesas	5
es.esi.cr.iso.gestionmesas.controller.GestorReservas	2
es.esi.cr.iso.gestionmesas.controller.TestGestorMesas	3
es.esi.cr.iso.gestionmesas.model.AbstractEntity	2
es.esi.cr.iso.gestionmesas.model.enums.EstadoEnum	2
es.esi.cr.iso.gestionmesas.model.enums.MesaEnum	2
es.esi.cr.iso.gestionmesas.model.enums.SourceConcernEnum	2
es.esi.cr.iso.gestionmesas.model.enums.SourceEnum	4
es.esi.cr.iso.gestionmesas.persistence.AbstractDao	1

Checking the report we can find that 9 of them belongs to the MALICIOUS_CODE category, the rest 13 bugs belongs to the BAD_PRACTICE category.

All of them had a Medium priority.

Let's focus first on fixing those belonging to the MALICIOUS_CODE category.

Every single one of those bugs belongs to one of these two types:

- exposure of the internal representation while returning a class attribute (get method).
- exposure of the internal representation while storing an externally mutable object (set method).

So, in order to solve these problems, we just use copies of the values that are requested to be returned or stored so the internal representation is not exposed any more.

Class GestorMesas

Those were the changes made in the code in order to fix the malicious_code category's bugs in the class GestorMesas:

```
public EstadoEnum[] getEstados() {  
    return estados.clone();  
}  
  
public void setEstados(EstadoEnum[] estados) {  
    this.estados = estados.clone();  
}
```

```
public Date getFechaServicio() {  
    return new Date(fechaServicio.getTime());  
}  
  
public void setFechaServicio(Date fechaServicio) {  
    this.fechaServicio = new Date(fechaServicio.getTime());  
}
```

```
public void setUltimoEstadoMesa(int [] valor) {  
    this.ultimoEstadoMesa = valor.clone();  
}
```

Class GestorReservas

Again the same problem appears, so the Date is changed to be a copy of the values:

```
public Date getFechaReserva() {  
    return new Date(fechaReserva.getTime());  
}  
  
public void setFechaReserva(Date fechaReserva) {  
    this.fechaReserva = new Date(fechaReserva.getTime());  
}
```

Class AbstractEntity

In AbstractEntity we just have to change the data type.

We were using Date to fix the previous bugs (as their return type/calling parameter was Date) but now we have to use Timestamp. The result is as follows:

```
public Timestamp getFecha() {  
    return new Timestamp(fecha.getTime());  
}  
  
public void setFecha(Timestamp fecha) {  
    this.fecha = new Timestamp(fecha.getTime());  
}
```

For the rest of the bugs, the ones belonging to the Category BAD_PRACTICES, we can check that we have all of them except one which has the same description in the Details Column: ME_ENUM_FIELD_SETTER.

This means that those set methods could be changing their values by malicious code or by accident by another package. So the solution to fix those bugs is remove them or make them private for the package.

The solution we carried out to fix them was to remove the following methods:

- EstadoEnum.setDescripcion(String)
- EstadoEnum.setId(int)
- MesaEnum.setComensales(int)
- MesaEnum.setId(int)
- SourceConcernEnum.setId(int)
- SourceConcernEnum.setLabel(String)
- SourceEnum.setExtensions(list)
- SourceEnum.setId(int)
- SourceEnum.setLabel(String)
- SourceEnum.setSourceConcern(SourceConcernEnum)

AbstractDao has another description in the Details column: SE_BAD_FIELD; which we were not able to find out how to fix it.

And finally some bugs from the category STYLE referring to some returning values from calling methods that were not being used.

Those are the ones in the Test Class where we call a method that returns a String but we are not using that returning value so we have done no more changes.