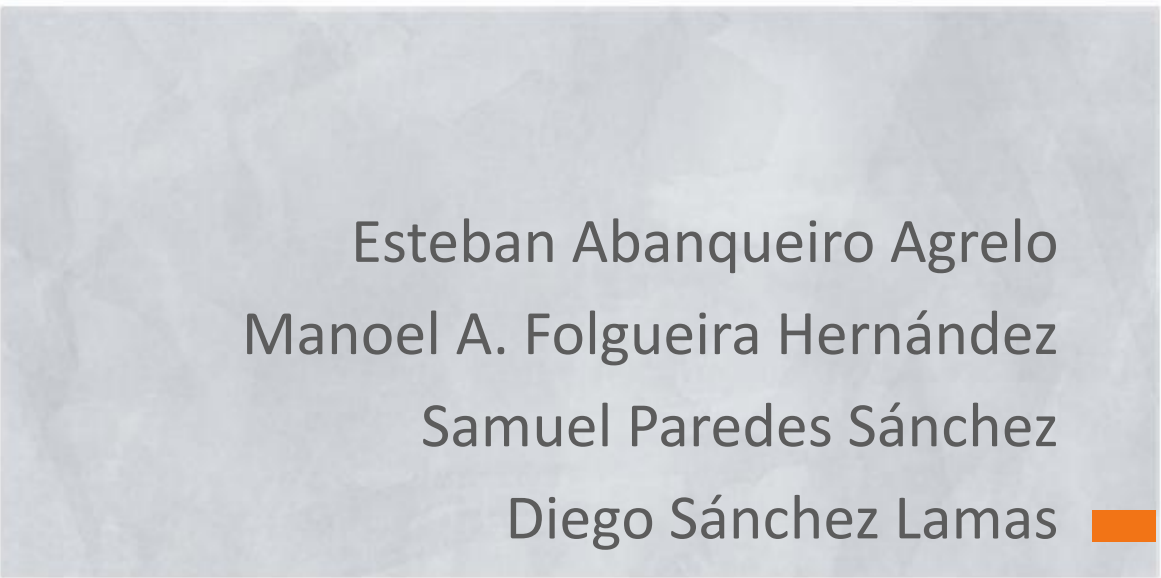





**EQUIPO-10**



Esteban Abanqueiro Agrelo  
Manoel A. Folgueira Hernández  
Samuel Paredes Sánchez  
Diego Sánchez Lamas 



presentación de

# Validación Automática Servidor de Contenidos



# Índice de contenidos (I)

---

- **Diseño - diagrama de clases UML**
- **Cronología de Trabajo**
  - **Semana 1**
    - **Git Branching**
    - **Travis**
    - **JUnit**
  - **Semana 2**
    - **Issues**



# Índice de contenidos (II)

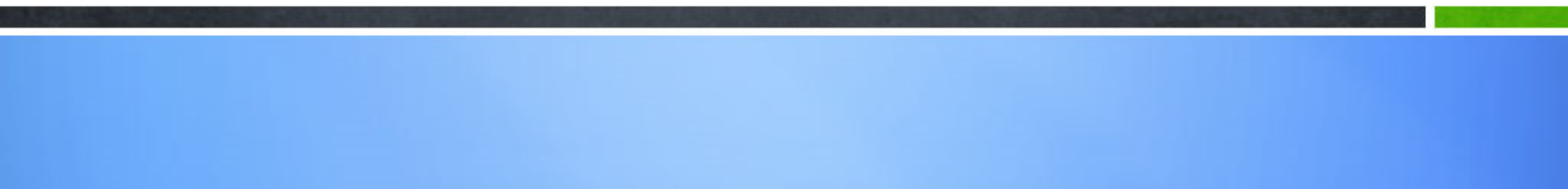
---

- **Cronología de Trabajo**
  - **Semana 3**
    - Junit
    - Cobertura
    - QuickCheck
    - GraphWalker
    - Mockito
    - FindBugs
    - PMD
    - CheckStyle

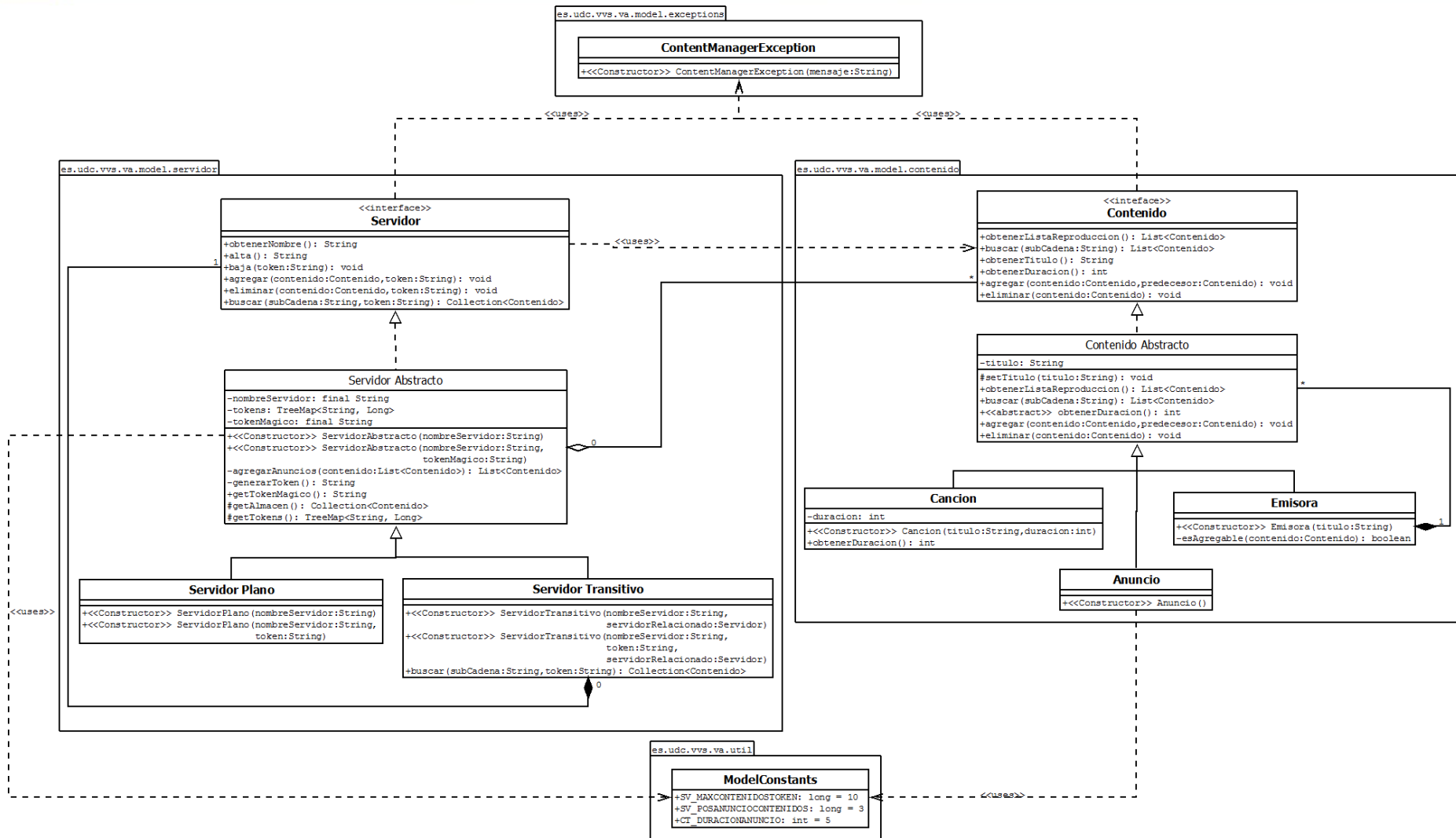




# Diseño



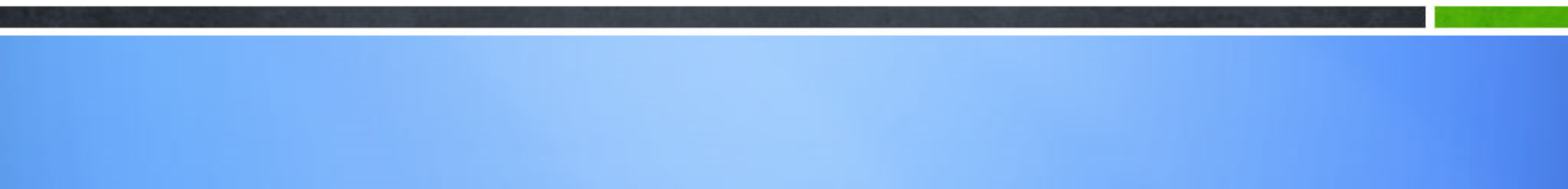
# Diseño: Diagrama de clases UML





# Cronología de trabajo

## Semana 1



## Git Branching

- Para facilitar el trabajo en paralelo ramificamos el repositorio usando la “nomenclatura” de git workflow.





## Travis-CI

“ Test and Deploy your Code with Confidence “



# Travis CI



## JUnit Semana 1: Happy Testing

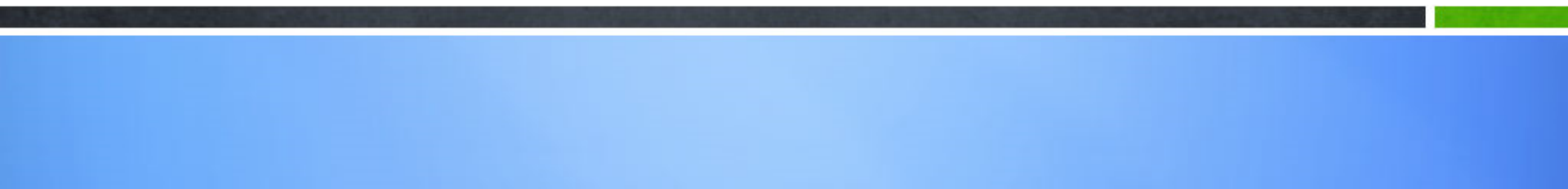
JUnit sirve  
para hacer  
tests  
repetibles  
a las  
aplicacion  
es de Java.

```
263
264     @Test
265     public void testObtenerDuracionEmisoraConEmisora()
266         throws ContentManagerException {
267
268         // Obtener duracion de una emisora con otra emisora dentro
269         Contenido e1 = crearEmisoraConContenido("e1");
270         Contenido e2 = crearEmisoraConContenido("e2");
271         e1.agregar(e2, null); // emisora2 en emisora1
272         assertEquals(1510, e1.obtenerDuracion());
273
274         // Obtener duracion de e1 una vez modificado el contenido de e2
275         Contenido anuncio = new Anuncio(); // nuevo anuncio (duracion = 5)
276         e2.agregar(anuncio, null); // anuncio en emisora2 (y en emisora1)
277         assertEquals(1515, e1.obtenerDuracion()); // se esperaba 5 pero fue 0
278
279     }
280
```








# Cronología de trabajo

## Semana 2



# Cronología de trabajo – Semana 2

## Issues más relevantes (1): issue recibida inválida

- ☐  **Excepciones en el diagrama de clases** 1  
#9 opened 28 days ago by eabanqueiro
- ☐  **Participación desequilibrada** invalid  1  
#8 opened 29 days ago by lauramcastro
- ☐  **Posible problema de eficiencia al obtener la duración de la emisora** invalid  2  
#7 opened on 12 Nov by adrianleira
- ☐  **Problema co "badge" de Travis-CI**  1  
#4 opened on 11 Nov by lauramcastro
- ☐  **Interface Contenido**  1  
#3 opened on 11 Nov by lauramcastro

## Issues más relevantes (2)

manoelfolgueira commented 27 days ago

Collaborator



A la hora de buscar en servidores de respaldo, usando un token de dar de alta en un servidor, éste sólo se encuentra en ese servidor y no es reconocido por los demás servidores de la cadena.

El test que lo demuestra:

### Compartición de tokens (bug)

```
@Test
public void testBusquedaTransitiva() throws ContentManagerException {
    ServidorPlano servidorPlano = new ServidorPlano("Servidor plano test");
    String tokenMagicoPlano = servidorPlano.getTokenMagico();
    servidorPlano.agregar(new Cancion("Read my mind", 5), tokenMagicoPlano);

    ServidorTransitivo servidorTransitivo =
        new ServidorTransitivo(
            "Servidor transitivo test", servidorPlano);
    String tokenTransitivo = servidorTransitivo.alta();

    // Busca en el Servidor Plano a traves del Servidor Transitivo
    // es.udc.vvs.va.model.exceptions.ContentManagerException:
    // Token invalido.
    // Los tokens "no se comparten".
    Collection<Contenido> contenidosEncontrados
        = servidorTransitivo.buscar("Read", tokenTransitivo);
}
```

### Issues más relevantes (3): cobertura el revelador

manoelfolgueira commented 11 days ago

Collaborator



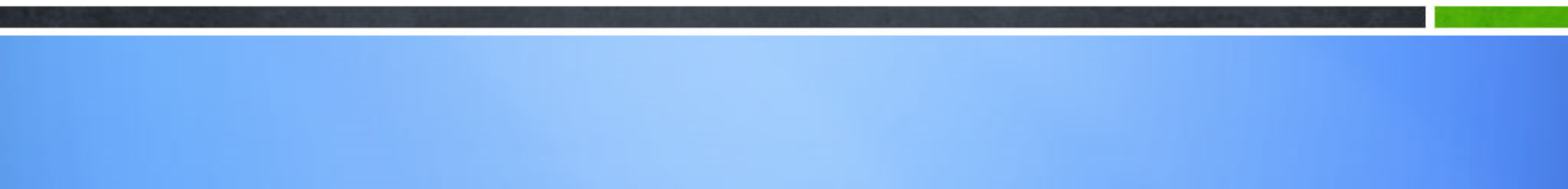
Viendo los últimos resultados de la cobertura, jamás se realiza el test de la baja de token delegada.





# Cronología de trabajo

## Semanas 3-5



## Cobertura (1): (98% de línea y 95% de rama)

Cobertura es una herramienta para validación de la calidad de las pruebas. Calcula el porcentaje de código visitada por las pruebas. Se puede utilizar para identificar qué partes del programa carece de cobertura de la prueba.

Inicialmente el valor de la cobertura era de 70-75%

### Packages

[All](#)

[es.udc.vvs.va.model.contenido](#)

[es.udc.vvs.va.model.exceptions](#)

[es.udc.vvs.va.model.servidor](#)

[es.udc.vvs.va.test.model.testautomation](#)

[es.udc.vvs.va.util](#)

**es.udc.vvs.va.model.servidor**

### Classes

[Servidor](#) (N/A)

[ServidorAbstracto](#) (96%)

[ServidorPlano](#) (100%)

[ServidorTransitivo](#) (100%)

### Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	12	98%	168/171	95%	80/84	1,967
<a href="#">es.udc.vvs.va.model.contenido</a>	5	100%	81/81	100%	44/44	2,231
<a href="#">es.udc.vvs.va.model.exceptions</a>	1	100%	2/2	N/A	N/A	1
<a href="#">es.udc.vvs.va.model.servidor</a>	4	97%	85/87	90%	36/40	1,963
<a href="#">es.udc.vvs.va.test.model.testautomation</a>	1	N/A	N/A	N/A	N/A	1
<a href="#">es.udc.vvs.va.util</a>	1	0%	0/1	N/A	N/A	0

Report generated by [Cobertura](#) 2.1.1 on 17/12/15 11:22.



## Cobertura (2)

### Packages

[All](#)

[es.udc.vvs.va.model.contenido](#)

[es.udc.vvs.va.model.exceptions](#)

[es.udc.vvs.va.model.servidor](#)

[es.udc.vvs.va.test.model.testauton](#)

[es.udc.vvs.va.util](#)

es.udc.vvs.va.model.servidor

### Classes

[Servidor](#) (N/A)

[ServidorAbstracto](#) (96%)

[ServidorPlano](#) (100%)

[ServidorTransitivo](#) (100%)

```
64 27
65 27
66
67
68
69
70
71
72 2
73
74
75
76
77
78
79
80
81
82
83
84 17
85 17
86
87 17
88 17
89
90

/* Para mas informacion
 * @see es.udc.vvs.vas.Servidor#obtenerNombre()
 */
@Override
public String obtenerNombre() {
    return nombreServidor;
}

/* Para mas informacion
 * @see es.udc.vvs.vas.Servidor#alta()
 */
@Override
public String alta() {
    String tokenNuevo;

    // para asegurarnos de que el nuevo token generado es distinto
    // de tokenMagico y no se encuentra en nuestra estructura de tokens
    while ((tokenNuevo = this.generarToken()).equals(tokenMagico)
        || tokens.containsKey(tokenNuevo));

    tokens.put(tokenNuevo, ModelConstants.SV_MAXCONTENIDOSTOKEN);
    return tokenNuevo;
}
```

## QuickCheck (1): Generador

```
class ContenidoListGenerator implements Generator<List<Contenido>> {
    Generator<List<Integer>> lGen = lists(PrimitiveGenerators
        .positiveIntegers());

    @Override
    public List<Contenido> next() {
        List<Integer> l = lGen.next();
        List<Contenido> lC = new ArrayList<Contenido>();
        boolean rand;

        for (Integer dur : l) {
            try {
                rand = new Random().nextBoolean();
                if (rand) {
                    lC.add(new Cancion(dur.toString(), dur));
                } else {
                    Emisora e = new Emisora(dur.toString());
                    e.agregar(new Cancion(dur.toString(), dur), null);
                }
            } catch (ContentManagerException e) {
                e.printStackTrace();
            }
        }

        return lC;
    }
}
```

## QuickCheck (2): Test

```
@Test
public void testDuracionEmisoraContenidosGenerados()
    throws ContentManagerException {

    int sumaDuraciones;

    for (List<Contenido> lc : Iterables
        .toIterable(new ContenidoListGenerator())) {

        Emisora e = new Emisora(lc.toString());
        sumaDuraciones = 0;

        for (Contenido contenido : lc) {
            e.agregar(contenido, null);
            sumaDuraciones += contenido.obtenerDuracion();
        }

        assertEquals(sumaDuraciones, e.obtenerDuracion());
    }
}
```

## JetM (1): Código de ejemplo

Es una herramienta para realizar monitorización de rendimiento (prueba no funcional).

```
public void buscarEnEmisoraConContenidos() throws ContentManagerException {  
    List<Emisora> emisoras = new ArrayList<Emisora>();  
    for (int i = 0; i < 10; i++) {  
        for (List<Contenido> lc : Iterables  
            .toIterable(new ContenidoListGenerator())) {  
            Emisora e = new Emisora(lc.toString());  
            for (Contenido contenido : lc) {  
                e.agregar(contenido, null);  
            }  
            emisoras.add(e);  
        }  
    }  
    for (Emisora e : emisoras) {  
        EtmPoint point = etmMonitor  
            .createPoint("Emisora:buscarContenidos");  
        e.buscar("1");  
        point.collect();  
    }  
}
```



## JetM (2): Resultados

[INFO] [EtmMonitor] JETM 1.2.3 started.

Measurement Point	#	Average	Min	Max	Total
Cancion:buscar	2000	0,001	0,000	0,037	1,131
Cancion:obtenerDuracion	2000	0,000	0,000	0,012	0,483
Emisora:buscarCanciones	2000	0,002	0,000	0,054	3,852
Emisora:buscarContenidos	2000	0,001	0,000	0,033	1,207
Emisora:obtenerDuracionCanciones	2000	0,001	0,000	0,038	1,170
Emisora:obtenerDuracionContenidos	2000	0,000	0,000	0,015	0,651
ServidorPlano:buscarCanciones	2000	0,019	0,000	33,713	37,275
ServidorPlano:buscarContenidos	2000	0,001	0,000	0,013	1,355
ServidorTransitivo:buscarCanciones	2000	0,005	0,002	0,538	10,590
ServidorTransitivo:buscarContenidos	2000	0,001	0,000	0,039	1,447

[INFO] [EtmMonitor] Shutting down JETM.

## GraphWalker (1)

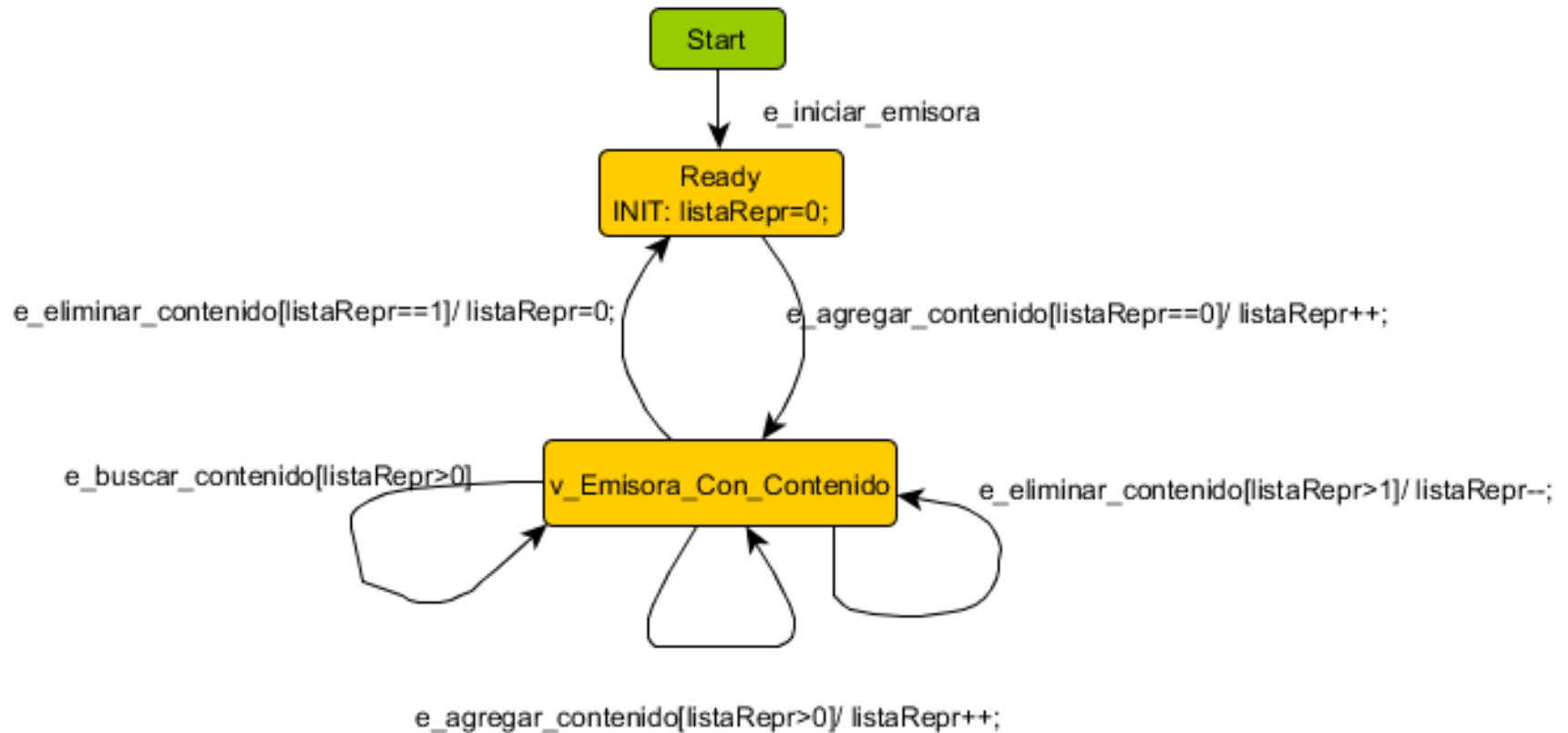
Es una herramienta para automatizar la generación y ejecución de pruebas dinámicas. Partiendo de un diagrama de estados (que debe seguir unas reglas concretas) genera una interfaz.

**\$ mvn graphwalker:generate-sources**

Dicha interfaz debe implementarse, generando para cada estado y cada transición las comprobaciones correspondientes.



## GraphWalker (2)





## GraphWalker (3)

### Smoke test

Verifica el flujo básico del modelo. Usando el algoritmo A\*, crea un camino recto desde el punto de partida, e\_Init, en el gráfico, al vértice v\_Browse.

```
@Test
public void runSmokeTest() {
    new TestBuilder()
        .setModel(MODEL_PATH)
        .setContext(new ContenidoTestImp())
        .setPathGenerator(new AStarPath(
            new ReachedVertex("v_Emisoras_Con_Contenido")))
        .setStart("e_iniciar_emisora")
        .execute();
}
```

### GraphWalker (4)

#### Functional test example

Cubre el gráfico completo. Se inicia a partir e\_Init, y termina tan pronto como la condición de parada se ha cumplido. Le indicamos que pase al menos una vez por cada transición.

```
@Test
public void runFunctionalTest() {
    new TestBuilder()
        .setModel(MODEL_PATH)
        .setContext(new ContenidoTestImp())
        .setPathGenerator(new RandomPath(
            new EdgeCoverage(100)))
        .setStart("e_iniciar_emisora")
        .execute();
}
```

## GraphWalker (5)

### Stability test example

Pedimos a GraphWalker que recorra el modelo al azar, hasta que se cumpla la condición de parada. Eso sucederá cuando haya pasado el tiempo definido. En una prueba real, podría ser de 30 minutos, o incluso horas.

```
@Test
public void runStabilityTest() {
    new TestBuilder()
        .setModel(MODEL_PATH)
        .setContext(new ContenidoTestImp())
        .setPathGenerator(new RandomPath(
            new TimeDuration(5, TimeUnit.SECONDS)))
        .setStart("e_iniciar_emisora")
        .execute();
}
```

## Mockito

En la imagen de la derecha podemos ver un ejemplo de cómo se crean objetos mock y como definimos lo que devuelve cada método de los *mock objects*.

Hemos detectado que el método buscar de emisora era case sensitive.

```
c1 = mock(Cancion.class);
c2 = mock(Cancion.class);
c3 = mock(Cancion.class);
a = mock(Anuncio.class);

when(c1.obtenerDuracion()).thenReturn(6);
when(c2.obtenerDuracion()).thenReturn(4);
when(c3.obtenerDuracion()).thenReturn(5);
when(a.obtenerDuracion()).thenReturn(5);

when(c1.obtenerTitulo()).thenReturn("Cancion 1");
when(c2.obtenerTitulo()).thenReturn("Cancion 2");
when(c3.obtenerTitulo()).thenReturn("Cancion 3");
when(a.obtenerTitulo()).thenReturn("PUBLICIDAD");

e = new Emisora("Emisora 1");
e.agregar(c1, null);
e.agregar(c2, c1);
e.agregar(c3, c2);
e.agregar(a, c3);
```

## CheckStyle (1)

Una vez checkeado el proyecto, hemos observado que nos aparecen numerosos errores de convencion, entre ellos están:

- uso de tabuladores en vez de espacios
- sangrías incorrectas
- orden incorrecto en la organización de los imports
- no uso de llaves para sentencias y bucles de una línea
- javadoc mal generado
- nombres de métodos con más de 1 mayúscula
- uso del mismo nombre para variables locales





## CheckStyle (2)

```
1
2
3 import es.udc.vvs.va.model.contenido.Contenido;
4 import es.udc.vvs.va.model.exceptions.ContentManagerException;
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31 */
32 public ServidorTransitivo(String nombreServidor, String token,
33     Servidor servidorRelacionado)
34     throws ContentManagerException {
35     super(nombreServidor, token);
36
37     if (servidorRelacionado == null) {
38         throw new ContentManagerException("Los servidores transitivos "
39             + "necesitan un servidor de apoyo.");
40     }
41
42     this.servidorRelacionado = servidorRelacionado;
43 }
```

## FindBugs

### Validacion Project

Last Published: 2015-12-17 | Version: 0.0.1-SNAPSHOT

**Project Documentation**

- Project Information
- Project Reports
  - Cobertura Test
  - Coverage
  - FindBugs**
  - PMD

Built by:  **maven**

## FindBugs Bug Detector Report

The following document contains the results of [FindBugs](#) 

FindBugs Version is *3.0.1*

Threshold is *medium*

Effort is *min*

## Summary

Classes	Bugs	Errors	Missing Classes
12	0	0	0

## Files

Class	Bugs
-------	------



## PMD (1): Report



### PMD Results

The following document contains the results of [PMD](#) 5.3.2.

### Files

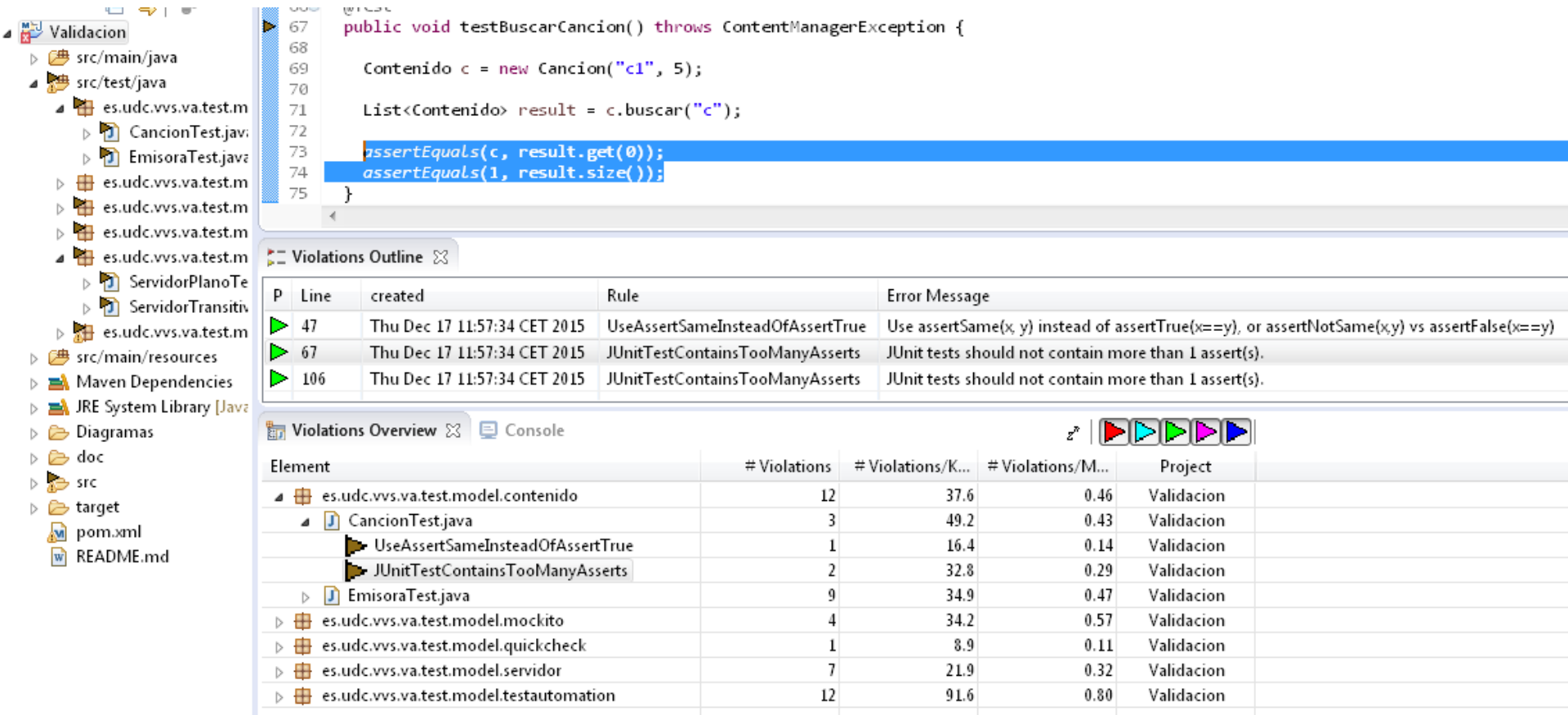
[es/udc/vvs/va/model/contenido/ContenidoAbstracto.java](#)

Violation	Line
Abstract classes should be named AbstractXXX	10-108
Avoid using if statements without curly braces	92-93
Avoid using if statements without curly braces	94-95
Avoid using if statements without curly braces	96-97
Avoid using if statements without curly braces	100-101
Avoid using if statements without curly braces	102-103

[es/udc/vvs/va/model/contenido/Emisora.java](#)

Violation	Line
Avoid variables with short names like i	92

## PMD (2): Too Many Asserts



```
67 public void testBuscarCancion() throws ContentManagerException {
68
69     Contenido c = new Cancion("c1", 5);
70
71     List<Contenido> result = c.buscar("c");
72
73     assertEquals(c, result.get(0));
74     assertEquals(1, result.size());
75 }
```

P	Line	created	Rule	Error Message
▶	47	Thu Dec 17 11:57:34 CET 2015	UseAssertSameInsteadOfAssertTrue	Use assertSame(x, y) instead of assertTrue(x==y), or assertNotSame(x,y) vs assertFalse(x==y)
▶	67	Thu Dec 17 11:57:34 CET 2015	JUnitTestContainsTooManyAsserts	JUnit tests should not contain more than 1 assert(s).
▶	106	Thu Dec 17 11:57:34 CET 2015	JUnitTestContainsTooManyAsserts	JUnit tests should not contain more than 1 assert(s).

Element	# Violations	# Violations/K...	# Violations/M...	Project
es.udc.vvs.va.test.model.contenido	12	37.6	0.46	Validacion
CancionTest.java	3	49.2	0.43	Validacion
UseAssertSameInsteadOfAssertTrue	1	16.4	0.14	Validacion
JUnitTestContainsTooManyAsserts	2	32.8	0.29	Validacion
EmisoraTest.java	9	34.9	0.47	Validacion
es.udc.vvs.va.test.model.mockito	4	34.2	0.57	Validacion
es.udc.vvs.va.test.model.quickcheck	1	8.9	0.11	Validacion
es.udc.vvs.va.test.model.servidor	7	21.9	0.32	Validacion
es.udc.vvs.va.test.model.testautomation	12	91.6	0.80	Validacion

## PMD (3): Unnecessary Assert

```
357  */
358  @Test(expected = ContentManagerException.class)
359  public void testDarDeBajaToken() throws ContentManagerException {
360
361      // Damos de baja el token existente, no debería dar excepcion
362      try {
363          // Primero buscamos con el token para comprobar que es un token
364          // valido
365          servidorPlano.buscar("cl", token);
366          servidorPlano.baja(token);
367      } catch (ContentManagerException e) {
368          assertTrue(false);
369      }
```

### Violations Outline

P	Line	created	Rule	Error Message
▶	175	Thu Dec 17 11:57:34 CET 2015	JUnitTestContainsTooManyAsserts	JUnit tests should not contain more than 1 assert(s).
▶	147	Thu Dec 17 11:57:34 CET 2015	JUnitTestContainsTooManyAsserts	JUnit tests should not contain more than 1 assert(s).
▶	302	Thu Dec 17 11:57:34 CET 2015	JUnitTestContainsTooManyAsserts	JUnit tests should not contain more than 1 assert(s).
▶	368	Thu Dec 17 11:57:34 CET 2015	UnnecessaryBooleanAssertion	assertTrue(true) or similar statements are unnecessary
▶	266	Thu Dec 17 11:57:34 CET 2015	JUnitTestContainsTooManyAsserts	JUnit tests should not contain more than 1 assert(s).

### Violations Overview

Element	# Violations	# Violations/K...	# Violations/M...	Project
▶ es.udc.vvs.va.test.model.contenido	12	37.6	0.46	Validacion
▶ es.udc.vvs.va.test.model.mockito	4	34.2	0.57	Validacion
▶ es.udc.vvs.va.test.model.quickcheck	1	8.9	0.11	Validacion
▶ es.udc.vvs.va.test.model.servidor	7	21.9	0.32	Validacion
▶ ServidorPlanoTest.java	6	33.5	0.46	Validacion
▶ JUnitTestContainsTooManyAsserts	5	27.9	0.38	Validacion
▶ UnnecessaryBooleanAssertion	1	5.6	0.08	Validacion
▶ ServidorTransitivoTest.java	1	7.1	0.11	Validacion
▶ es.udc.vvs.va.test.model.testautomation	12	91.6	0.80	Validacion

### Debemos recordar que...

Aunque **FindBugs**, **PMD** y **CheckStyle** son todas ellas herramientas de pruebas estructurales...

- **FindBugs** sirve para detectar errores potenciales.
- **PMD** sirve para evitar malas prácticas.
- **CheckStyle** se centra en la detección de violaciones de convenciones de codificación.
- **Conclusión**: el uso combinado de las tres herramientas cubre el rango necesario de aspectos a analizar en nuestro código.





FIN DE LA PRESENTACIÓN  
¿PREGUNTAS?