MONITORIZACIÓN DE PROBAS

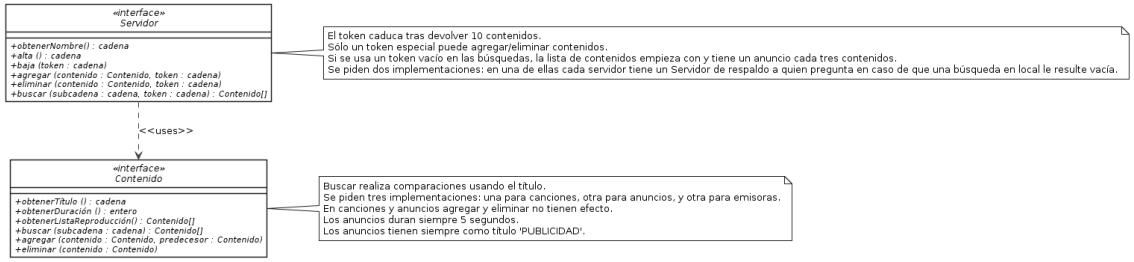
Título proxecto: vvsTesting
Ref. proxecto: https://www.github.com/Kaizo88/prac3-vvs1516

Validación e Verificación de Software

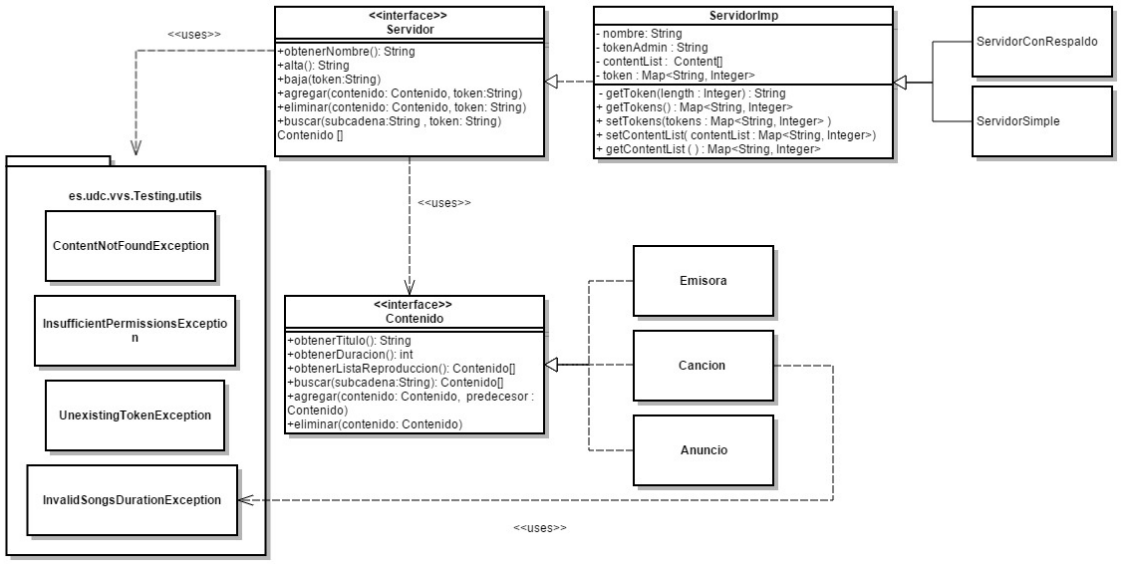
Data de aprobación	Control de versións	Observacións

1. Contexto

Partiendo del siguiente especificación:



Se paso a la implementación en Java de dicha especificación. El siguiente diagrama UML muestra las clases y las relaciones entre ellas:



En el paquete es.udc.vvs.Testing.utils están contenidas las distintas excepciones, y se optó por implementar la clase ServidorImp que implementaba las funcionalidades de la interfaz Servidor, comunes para todos los servidores, y que implementanciones concretas de Servidor pueden redefinir. Los clases ServidorConRespaldo y ServidorSimple heredan de ServidorImp. En el caso de ServidorConRespaldo, redefine los métodos buscar.

2. Estado actual

2.1. Funcionalidades

2.1.1. Servidor

- obtenerNombre: Devuelve el nombre del servidor.
- agregar: Agrega un contenido al servidor. Si el token no es el token de administrador devolvera una excepcion.
- alta: Da de alta un nuevo token en el servidor, y lo devuelve.
- baja: Da de baja un token especificado del servidor.
- buscar: Busca contenidos que contengan una subcadena, y utilizando un token. Devuelve los contenidos encontrados. Si el token pasado para buscar no existe lanza una excepcion.
- eliminar: Elimina un contenido del servidor.

2.1.2. Contenido

- agregar: Agrega elementos de tipo Contenido a una emisora. Si le pasas un predecesor null, el contenido se inserta al principio de la lista de reproduccion de la emisora. En otro caso, después del predecesor indicado (si existen varias veces el predecesor en la lista, se inserta después de cada uno).
- obtenerDuracion: Devuelve la duración del contenido. O la duración total de los contenidos incluídos dentro de una emisora.
- obtenerListaReproduccion: Metodo que obtiene la lista de reproduccion con el contenido. O la lista de todos los contenidos contenidos en el caso de una emisora.
- buscar: Metodo que obtiene la lista de contenidos que coinciden con la busqueda por subcadena.
- eliminar: Metodo que elimina todas las apariciones de un contenido.
- obtenerTitulo: Devuelve el título del contenido.

2.2. Utils

- ContentNotFoundException: Excepción lanzada cuando no se cuentra el contenido.
- InsufficientPermissionsException: Excepción lanzada cuando no se disponen de los permisos necesarios para realizar ciertas tareas sobre el servidor.
- InvalidSongsDurationException : Excepción lanzada cuando se intenta crear una canción con duración igual, o menor a 0.
- UnexistingTokenException: Excepción lanzada cuando se pasa un token en la búsqueda de contenidos en el servidor, y el token no es válido.

2.3. Pruebas dinámicas realizadas

- Servidor Para el servidor se han realizado :
  - JETM : 6
  - QuickCheck: 6
  - Mock : 4
  - JUnit : 10
  - GraphWalker : 2
- Para el contenido se han realizado :
  - JETM : 18
  - QuickCheck: 4
  - JUnit : 12

Para un porcentaje del 100 % de tests pasados.

2.4. Personal encargado del desarrollo y pruebas

Para el desarrollo de las pruebas, se dividieron los dos paquetes a probar. Se asignó el paquete Contenido a Pablo y Jose. Se asignó el paquete Servidor a Diego y Jorge.

2.5. Estado actual de las pruebas

Se realizaron todas las pruebas previstas, consiguiendo una cobertura de 100 % y con todos los errores, bugs y fallos corregidos.

3. Registro de pruebas

3.1. Herramienta para automatizar la ejecución de pruebas dinámicas

Utilizando el framework JUnit, para la realización de pruebas unitarias y de integración, por su conocimiento previo y aprovechando que otras herramientas de prueba utilizadas dependían de este framework para ejecutarse. Para mantener la integridad de las pruebas de unidad, utilizamos el framework Mockito para realizar mocks de los contenidos y utilizarlos en las pruebas de los servidores.

3.2. Pruebas funcionales de caja negra

3.2.1. Valores-Frontera

En la implementación de los test de unidad, se escogieron valores frontera significativos para intentar encontrar errores, como agregar contenido a una emisora vacia y no vacía.

3.2.2. Tablas de decisión

Utilizamos tablas de decisión para planificar ciertas pruebas.

Tabla de decisión del método agregar de la clase emisora

Cadro 1: *tabla de decisión*

	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6
Precedente nulo	si	si				
Precedente en lista			si			
Precedente varias veces en lista				si		
Precedente no está en lista					si	si
Lista vacía	si	no	no	no	si	no

Tabla de decisión del método eliminar de la clase emisora:

Cadro 2: *tabla de decisión*

	caso 1	caso 2	caso 3
ausente	si		
presente una vez		si	
presente varias veces			si

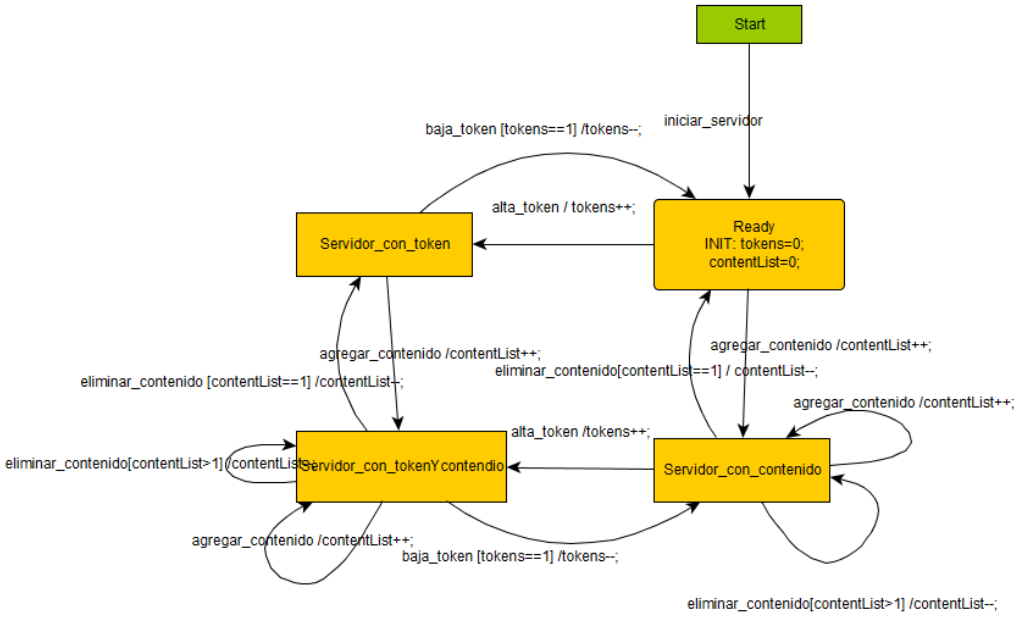
3.2.3. Generación de datos aleatorios

Herramienta utilizada: Quickcheck. Mediante el framework mencionado, se crearon test que generaban instancias de Anuncio o Canciones de forma aleatoria, con titulos y duración variable. Además a partir de las tablas de decisión anteriormente mencionadas se crearon test generando las emisoras con listas de canciones aleatorias, para probar a eliminar y agregar canciones.

3.2.4. Basadas en modelos y propiedades

Utilización de JUnit para la implementación y ejecución de test con aserciones.

Utilización de GraphWalker para la ejecución y generación de test a partir de una máquina de estado finito.



3.3. Pruebas funcionales de caja blanca

3.3.1. Cobertura de instrucciones

Empleo de Coveralls.io para monitorizar la cobertura de las lineas del código recorridas por los test.

3.3.2. Revisión de código

Empleo de checkstyle para la revisión de errores de codificación y estilo. Se genera automaticamente un reporte en el site automatizado de maven.

3.3.3. Mutación de código

Empleo de PIT para la automatización de los tests de mutación, y la cobertura de las mutaciones de código que nuestras pruebas eliminan. Se genera automaticamente un reporte en el site automatizado de maven.

3.4. Pruebas no funcionales de caja blanca

3.4.1. Análisis del tiempo de ejecución

Empleo del framework JETM para el análisis del tiempo de ejecución de los diferentes métodos de nuestras clases. Se genera automaticamente un reporte con los tiempos calculados por el framework, en el site automatizado de maven.

4. Rexistro de erros

1. Utilizacion de equals(null) en ServidorConRespado bug. cerrado 2/12
2. Comprobar estilo en paquete Servidor enhancement. cerrado 3/12
3. Fallo en implementación emisora: eliminar bug. cerrado 1/12
4. Fallo en implementación emisora: agregar bug. cerrado 1/12
5. Se pueden crear canciones con duracion negativa bug cerrado. 8/12
6. Búsqueda de contenido incorrecta bug cerrado. 23/12
7. Error metodo agregar emisora : Predecesor que no existe bug. cerrado 18/11
8. Mostrar en UML excepciones. cerrado. 8/12
9. Participación desequilibrada. cerrado 13/12
10. Uso de constantes na implementación do servidor. cerrado 13/11
11. Tipos de variables en UML cerrado. 12/11
12. variable sin usar cerrado. 12/11
13. Uso del equals en el metodo agregar bug help wanted. cerrado 17/11
14. Operaciones no permitidas agregar y eliminar. cerrado 12/11
15. Nomes de interfaces incorrecto. cerrado 12/11
16. Aínda falta documentación. cerrado 15/11
17. Falta documentación. cerrado 11/11

5. Estatísticas

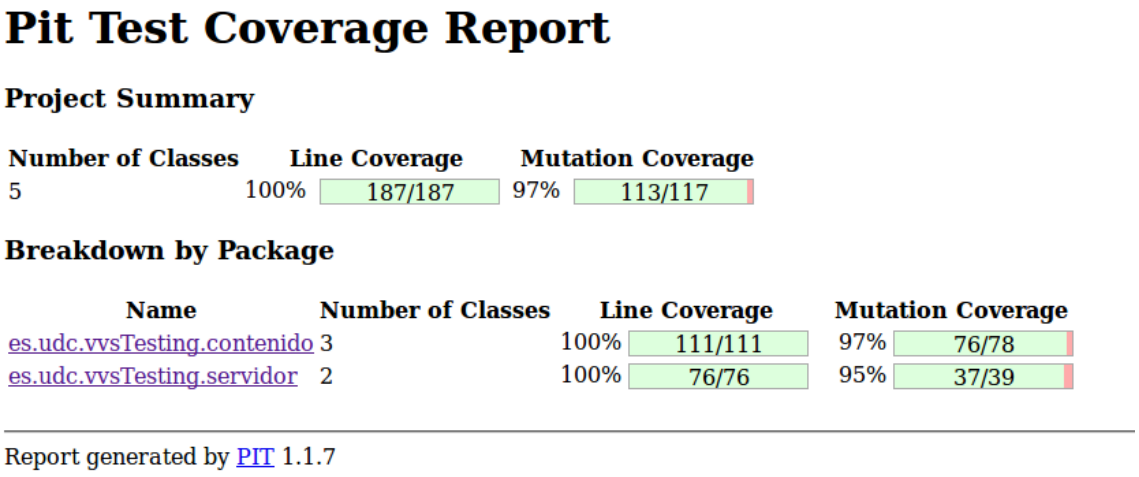
5.1. Errores semanales

- **Semana 1 11/11-17/11** Errores encontrados 9
- **Semana 2 18/11 -24/11** Errores encontrados 4
- **Semana 3 25/11 -1/12** Errores encontrados 4
- **Semana 4 2/12 - 8/12** Errores encontrados 4

El resto de los 107 commits fueron para crear funcionalidad ,añadir tests y borrar fallos propios.

5.2. Porcentajes

- Porcentaje de PIT global del proyecto: 100 % cobertura de lineas, 97 % Cobertura de mutaciones



Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
5	100%	187/187	97%	113/117

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
es.udc.vvsTesting.contenido	3	100%	111/111	97%	76/78
es.udc.vvsTesting.servidor	2	100%	76/76	95%	37/39

Report generated by PIT 1.1.7

- Porcentajes de PIT del paquete contenido: 100 % cobertura de lineas, 97 % Cobertura de mutaciones

Pit Test Coverage Report

Package Summary

es.udc.vvsTesting.contenido

Number of Classes	Line Coverage		Mutation Coverage	
3	100%	111/111	97%	76/78

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Anuncio.java	100%	27/27	94%	17/18
Cancion.java	100%	40/40	97%	29/30
Emisora.java	100%	44/44	100%	30/30

Report generated by PIT 1.1.7

- Porcentajes de PIT del paquete servidor: 100 % cobertura de lineas, 95 % Cobertura de mutaciones

Pit Test Coverage Report

Package Summary

es.udc.vvsTesting.servidor

Number of Classes	Line Coverage		Mutation Coverage	
2	100%	76/76	95%	37/39

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
ServidorConRespaldo.java	100%	7/7	100%	4/4
ServidorImp.java	100%	69/69	94%	33/35

Report generated by PIT 1.1.7

- Resumen de mediciones de JETM:

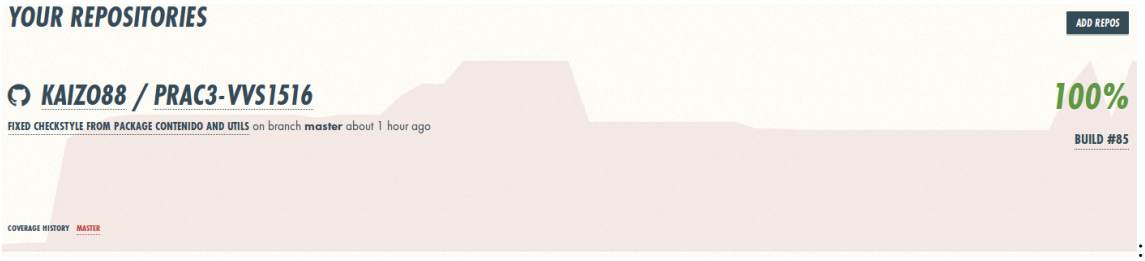
File Breakdown

This is a list of, per XML file, the measurements taken.

Timing.xml

Name	Average (sec)	Measurements	Minimum (sec)	Maximum (sec)	Total
AnuncioPerfTest:obtenerListaReproduccion	0,00	1	0,00	0,00	0,00
EmisoraPerformance:eliminar	0,00	1	0,00	0,00	0,00
ServidorConRespaldo:buscar	0,06	1	0,06	0,06	0,06
ServidorSimple:agregar	0,00	1	0,00	0,00	0,00
AnuncioPerfTest:obtenerTitulo	0,00	1	0,00	0,00	0,00
ServidorSimple:eliminar	0,00	1	0,00	0,00	0,00
AnuncioPerfTest:obtenerDuracion	0,00	1	0,00	0,00	0,00
EmisoraPerformance:buscar	0,05	1	0,05	0,05	0,05
EmisoraPerformance:agregar	0,00	1	0,00	0,00	0,00
CancionPerformance:buscar	0,01	1	0,01	0,01	0,01
CancionPerformance:obtenerTitulo	0,00	1	0,00	0,00	0,00
CancionPerformance:obtenerListaReproduccion	0,00	1	0,00	0,00	0,00
EmisoraPerformance:obtenerListaReproduccion	0,00	1	0,00	0,00	0,00
ServidorSimple:baja	0,00	1	0,00	0,00	0,00
CancionPerformance:obtenerDuracion	0,00	1	0,00	0,00	0,00
AnuncioPerfTest:buscar	0,00	1	0,00	0,00	0,00
ServidorSimple:obtenerTitulo	0,00	1	0,00	0,00	0,00
ServidorSimple:alta	0,02	1	0,02	0,02	0,02
EmisoraPerformance:obtenerDuracion	0,00	1	0,00	0,00	0,00

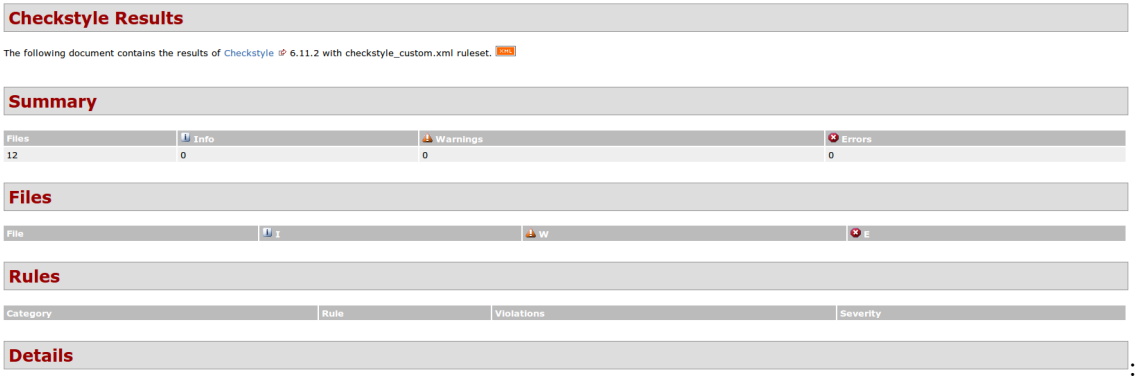
- Gráfica de la cobertura del código en Coveralls.io



:

100 % cobertura de líneas

- Captura del reporte de Checkstyle



:

100 % Corregido

- Captura del reporte de findBugs: 100 % Corregido

5.3. Análisis del perfil de detección de errores

Bugs

- Utilizacion de equals(null) en ServidorConRespado. Paquete servidor. Encontrado con checkstyle

- Se pueden crear canciones con duracion negativa. Paquete contenido.Encontrado con Quickcheck
- Mostrar en UML excepciones. Documentación. Revisión de código
- Comprobar estilo en paquete Servidor. Paquete servidor. Encontrado con checkstyle.
- Variable servidorRespaldo sin usar Paquete servidor.Revisión de código.
- Falta documentación. Documentación. Encontrado con revisión de código(otro grupo)
- Errores checkstyle. Paquete servidor.Encontrado con checkstyle.
- Errores encontrados con findbugs. Paquete servidor y contenido. Encontrado con findbugs
- Eliminada la excepcion searchlimitreachedexception porque no se usaba.Paquete utils. Revisión de código.
- Sobreescrito el equals y el hashcode para anuncio y cancion, ahora compara los contenidos por equlas y no por titulo. Paquete contenido.

Mejoras

- Participación desequilibrada.
- Uso de constantes na implementación do servidor. Paquete servidor. Encontrado con revisión de código(otro grupo)

Bugs de especificación

- Fallo en implementación emisora: eliminar. Paquete contenido. Encontrado con quickcheck.
- Fallo en implementación emisora: agregar. Paquete contenido. Encontrado con quickcheck.
- Búsqueda de contenido incorrecta . Paquete servidor. Encontrado con junit.

- Error metodo agregar emisora : Predecesor que no existe bug. Paquete contenido. Encontrado con junit.
- Tipos de variables en UML cerrado. Documentación. Encontrado con revisión de código(otro grupo)
- Operaciones no permitidas agregar y eliminar. Paquete contenido. Encontrado con revisión de código(otro grupo)
- Nomes de interfaces incorrecto. Paquete contenido y servidor. Encontrado con revisión de código(otro grupo)

Issues que no resultaron ser errores

- Uso del equals en el metodo agregar

5.4. Informe de erros abertos e pechados por nivel de criticidade

1. Fallo en implementación emisora: eliminar.
2. Fallo en implementación emisora: agregar.
3. Búsqueda de contenido incorrecta .
4. Error metodo agregar emisora : Predecesor que no existe bug.
5. Tipos de variables en UML cerrado.
6. Operaciones no permitidas agregar y eliminar.
7. Nomes de interfaces incorrecto.
8. Sobreescrito el equals y el hashcode para anuncio y cancion, ahora compara los contenidos por equlas y no por titulo.
9. Se pueden crear canciones con duracion negativa.
10. Utilizacion de equals(null) en ServidorConRespado.
11. Mostrar en UML excepciones.
12. Comprobar estilo en paquete Servidor.
13. Variable servidorRespaldo sin usar Paquete servidor.

14. Falta documentación.
15. Errores checkstyle.
16. Errores encontrados con findbugs.
17. Eliminada la excepcion searchlimitreachedexception porque no se usaba.
18. Participación desequilibrada.
19. Uso de constantes na implementación do servidor.

5.5. Evaluación global del estado de calidad y estabilidad actuales.

Consideramos que ha alcanzado un nivel de estabilidad bastante alto , pues hemos realizado varios tipos de prueba posible.

6. Outros aspectos de interese

- Para mayor información acerca de los reportes de los frameworks, ejecutar la directiva de maven encontrada en el readme, y navegar a la carpeta target/site/index.html
- Algunos de los frameworks, principalmente JETM, carecen de documentación suficiente para su ejecución. Especialmente para el apartado de reporting en html.