PLAN DE PROBAS

Título proxecto: practica-vvs
Ref. proxecto:
https://github.com/Xokage/practica-vvs-4malladores.git

Validación e Verificación de Software

Data de aprobación	Control de versións	Observacións
01/12/2015	1.0	Realizamos o plan de probas

1. Introdución

1.1. Propósito

O plan de probas realizarase sobre a primeira iteración do proxecto. Os obxetivos globais das probas da iteracion serán encontrar erros con respecto a funcionalidade do software, con respecto á especificación e detectar posibles melloras con respecto aos requerimentos non funcionais (no caso de requerírense).

2. Compoñentes avaliados

Listaxe de compoñentes que se van someter a proba, cos seus numeros de version:

- Anuncio - 1.0
- Canción - 1.0
- Emisora - 1.0
- ServidorSimple - 1.0
- ServidorConRespaldo - 1.0

3. Funcionalidades

https://moodle.udc.es/mod/page/view.php?id=343367

- Relativas a Contido:
  - Pódense procurar contidos usando o título para comparar.
  - Posúe os atributos Título, Duración e Lista de Reprodución, e métodos para obter estes atributos.
  - Relativas a Anuncio:
    - Agregar e eliminar non teñen efecto.
    - Duran sempre 5 segundos.
    - Sempre teñen por título: 'PUBLICIDAD'
  - Relativas a Canción:
    - Agregar e eliminar non teñen efecto.
  - Relativas a Emisora:

    - Agregar e Eliminar agregan e eliminan contidos da lista de reproducción.

- Relativas a Servidor:
  - O token caduca tras devolver 10 contidos.
  - Só un token especial pode agregar ou eliminar contidos.
  - Se se usa un token baleiro nas procuras a lista de contidos comeza con e ten un anuncio cada tres contidos.
  - Dous tipos de servidor. O servidor simple sen respaldo, e o servidor con respaldo, este último terá un servidor de respaldo ao que lle preguntará no caso de que a procura en local resulte baleira.

4. Interfaces

O noso software non se relaciona con outros sistemas ou compoñentes externos ó propio sistema, polo que non cremos necesario engadir nada ó respecto.

5. Procesos de negocio

O noso sistema non afecta a ningún proceso de negocio.

6. Parámetros de calidade

A especificación non fai referencia ao cumplimento de ningún tipo de requerimento non funcional, polo cal non podemos probalo.

7. Especificación de probas

Probas dinámicas de unidade

- **Mockito**
  - Descrición do escenario

    Probas con Mockito para Anuncio,Cancion,Emisora,ServidorSimpleImp e ServidorSimpleConRespaldoImp, para o cal se crean os mocks correspondentes que simularán a implementación de ditos compoñentes: MockAnuncio, MockCancion, MockEmisora, MockServidorSimpleImp, MockServidorSimpleConRespaldoImp.

Toda a documentación relacionada co uso da ferramenta Mockito pódese consultar aquí.

Un mock é un obxecto que simula o comportamento dun compoñente que ainda non está construído. Son útiles para simular dito comportamento inexistente, pero requieren dunha certa configuración que se leva a cabo en tempo de execución. Neste caso concreto, para simular o comportamento dos compoñentes citados anteriormente indicamoslle ao mock utilizado en cada caso concreto o valor que ten que devolver tras invocar a un método con uns parámetros de entrada concretos. Isto podese facer indicandolle ao mock como se ten que comportar tras a invocación a certas funcionalidades. Para que sirva de exemplo, indicarei o que fixen para configurar o mock nun determinado test:

```
@Test public void obtenerTituloConAnuncioMockTest()
    when(anuncioMock.obtenerTitulo()) then Return("PUBLICIDAD");
    assertTrue(anuncioMock.obtenerTitulo().equals("PUBLICIDAD"));
```

Como Mockito é unha ferramenta moi intuitiva, pódese interpretar case literalmente o funcionamento da mesma sen necesidade de ter un coñecemento moi amplo sobre ela. Da segunda liña podese deducir facilmente que cando se invoca a funcionalidade obtenerTitulo() de anuncioMock, devolve a cadea "PUBLICIDAD". Con isto, conseguimos configurar o mock para que cando reciba esa chamada na liña 3, devolva a cadea "PUBLICIDAD", que é o título que teñen todos os anuncios por defecto. A finalidade disto é poder probar compoñentes que aínda non existen e nun futuro poder reemprazar os mock polos compoñentes reais sen necesidade de modificar a lóxica dos tests. Se os tests están ben feitos, o resultado dos mesmos deberán ser iguais antes e despois da sustitucion dos mesmos polos seus respectivos compoñentes software aos que imitan.

- Criterios das probas:
  - Éxito: As probas pasan sen erros.
  - Abandono: Modificouse unha funcionalidade no software e as probas non son aplicables.

- **GraphWalker**

- Descrición do escenario

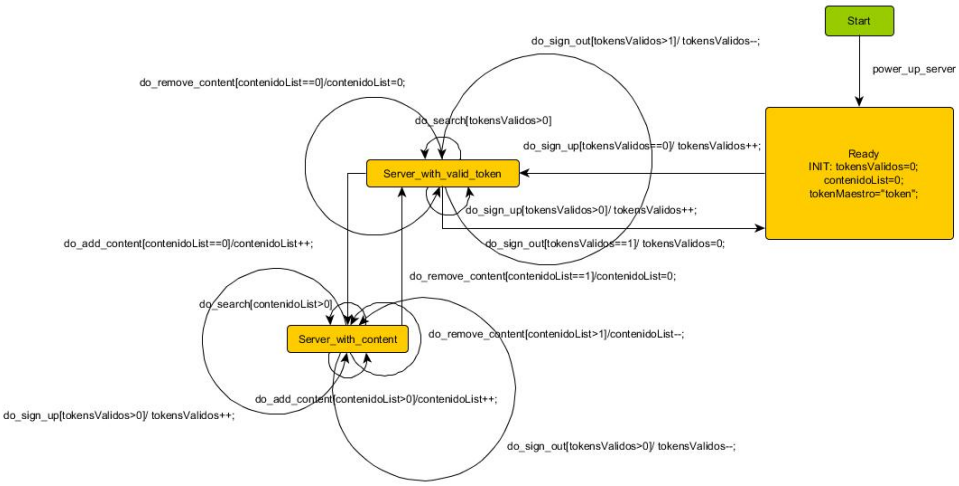
  GraphWalker consiste nunha ferramenta para automatizar a xeración e execución de probas dinámicas. A través desta ferramenta, vamos a probar a clase servidor ServidorSimpleImp coas suas correspondentes operacións.

  Toda a información necesaria para entender mellor o funcionamento de GraphWalker pódese atopar no seguinte enlace.

  Para poder executar estas probas, o primeiro paso foi crear un diagrama de transición de estados mediante a ferramenta YedGraph. Neste diagrama

  (/practica-vvs/src/main/resources/testautomation/VVS.graphml)

  é necesario ter en conta todas as posibles transicións entre os diferentes estados polos que pode pasar o servidor. Tamén é moi importante destacar a importancia que ten a nomenclatura utilizada no diagrama xa que se non se sigue o modelo tal e como se indica na documentación de GraphWalker non se xerarán correctamente as probas. O diagrama queda, polo tanto, da seguinte forma:



  Este diagrama será necesario comprobalo mediante o ficheiro .jar que se aporta na paxina de GraphWalker. Unha vez comprobado que o diagrama cumpre o estándar proposto, engadimos as dependencias necesarias ó proxecto e executaremos **mvn graphwalker:generate-sources**. Isto

xeraranos o código necesario no proxecto, o cal teremos que implementar para completar as probas. Unha vez implementadas, estas probas executaranse coma se fose un test JUnit máis. O ideal neste caso sería que, da execución das probas. obtivesemos que a totalidad do código funciona correctamente, pero non foi así, xa que houbo transicións entre distintos estados que non eran correctas. Para elo, foi necesarios modificar o diagrama varias veces durante a execución das probas e asegurarnos de que as transicións entre os distintos estados era consistente. Debemos indicar, que no test no que se comproba o 100 % da cobertura de rama de graphwalker, tivemos que engadir unha condición a maiores de parada despois de 15 segundos xa que os tests encontraban unha secuencia de transicións infinita na que nunca paraba polo que non se podían completar os demais tests JUnit con éxito.

- Criterios das probas
  - Éxito: As probas pasan sen erros.
  - Suspensión: As probas atopan erros ou fallan os assert. Habería que revisar o código e volver a realizar todas as probas, incluídas as que non son de unidade.
  - Abandono: Modificouse unha funcionalidade no software e as probas non son aplicables.

- **JUnit**
  - Descrición do escenario

    As probas de unidade realizaranse usando **JUnit** como base. Probarán cada uso habitual de cada método de cada clase, é decir, aqueles casos nos que o comportamento sexa normal, sen provocar excepcións non esperadas. Os datos de entrada ás probas xeraranse automáticamente con **QuickCheck**.
  - Criterios das probas
    - Éxito: As probas pasan sen erros.
    - Suspensión: As probas atopan erros ou fallan os assert. Habería que revisar o código e volver a realizar todas as probas, incluídas as que non son de unidade.

- Abandono: Modificouse unha funcionalidade no software e as probas non son aplicables.

Probas non funcionais - Rendemento

- **JETM**
  - Descrición do escenario

    Para as probas non funcionais utilizamos a ferramenta **JETM** que permite realizar probas de rendemento e comprobar a velocidade de execución dos métodos.

    Estas probas atópanse no paquete test.performance e execútanse co método main da clase GenerateReport. Ao inicialo executa cada método da clase un número de veces configurable (por defecto 10000) e comproba o tempo que se tarda en realizar as iteracións. A táboa que se obtén é similar á seguinte:

```
[INFO ] [EtmMonitor] JETM 1.2.3 started.
|-------------------------------------------------|---|---------|--------|--------|--------|
|                Measurement Point                | # | Average |   Min  |   Max  |  Total |
|-------------------------------------------------|---|---------|--------|--------|--------|
| AnuncioPerformance:buscar                       | 2 |   2,081 |  2,030 |  2,132 |  4,162 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| AnuncioPerformance:obtenerDuracion              | 1 |   1,295 |  1,295 |  1,295 |  1,295 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| AnuncioPerformance:obtenerListaReproduccion     | 1 |   1,194 |  1,194 |  1,194 |  1,194 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| AnuncioPerformance:obtenerTitulo                | 1 |   1,654 |  1,654 |  1,654 |  1,654 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| CancionPerformance:buscar                       | 2 |   2,516 |  2,165 |  2,867 |  5,033 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| CancionPerformance:obtenerDuracion              | 1 |   1,504 |  1,504 |  1,504 |  1,504 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| CancionPerformance:obtenerListaReproduccion     | 1 |   1,247 |  1,247 |  1,247 |  1,247 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| CancionPerformance:obtenerTitulo                | 1 |   1,486 |  1,486 |  1,486 |  1,486 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| EmisoraPerformance:agregar                      | 1 |   2,687 |  2,687 |  2,687 |  2,687 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| EmisoraPerformance:buscar                       | 2 |   2,623 |  2,075 |  3,172 |  5,247 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| EmisoraPerformance:eliminar                     | 1 |   4,158 |  4,158 |  4,158 |  4,158 |
|-------------------------------------------------|---|---------|--------|--------|--------|
| EmisoraPerformance:obtenerDuracion              | 1 |   1,290 |  1,290 |  1,290 |  1,290 |
|-------------------------------------------------|---|---------|--------|--------|--------|
```

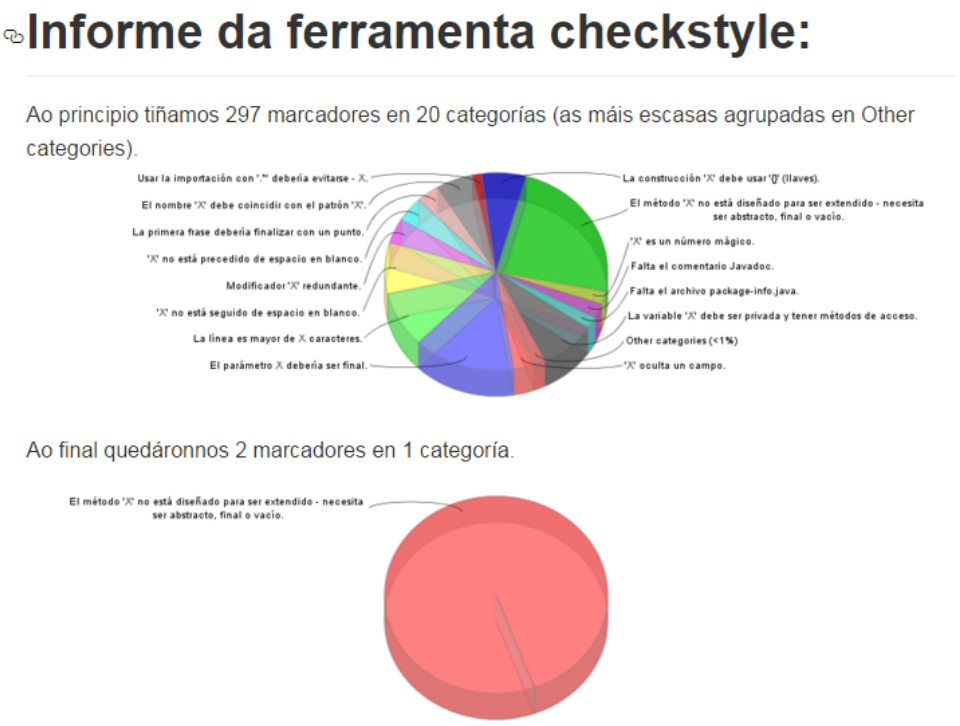
  - Criterios da proba
    - Éxito: O tempo Average e Max é razoable (entendemos por razoable unha medida non superior ao minuto para os métodos onde se manipule unha lista e non superior aos 5 segundos no resto de métodos, xa que supoñemos que se tardan máis que iso hai fallos de rendemento).
    - Suspensión: As probas tardan máis do que deberían. Habería que corrixir o código e volver a realizar as probas de performance.

- Abandono: Modificouse unha funcionalidade no software e as probas non son aplicables.

Probas estruturais/estáticas - Estilo de código

- Descrición do escenario

  Para realizar estas probas utilizamos a ferramenta **CheckStyle**.
  Na wiki pódese atopar información sobre o uso desta ferramenta:



- Criterios das probas
  - Éxito: As incidencias que queden por solucionar do checkstyle son mínimas e/ou de pouca importancia.
  - Suspensión: Hai un número moi elevado de incidencias. Habería que corrixir o código e volver a comprobar co checkstyle.
  - Abandono: Trócase o estándar de codificación.

8. Necesidades

Un pc con soporte para as ferramentas que se van a utilizar para a construcción da suite de probas (Mockito, JUnit, GraphWalker, Cobertura, ...). En canto as necesidades do persoal e de formación é necesario que éstes se familiaricen coas distintas ferramentas de proba que se van a empregar para a correcta construcción da suite de probas.

9. Responsabilidades

A reponsabilidade da preparación, execución, notificación, xestión, deseño e resolución de conflitos recae sobre o equipo de desenvolvemento formado polos seguintes individuos:

- Xoán Antelo Castro
- Cristian Canosa Pérez
- Francisco Pais Fondo
- Martín Quinteiro Santos

10. Planificación

Non dispoñemos do tempo suficiente e necesario para poder levar a cabo unha correcta planificación do proxecto.

11. Outros aspectos de interese

Non existen outros aspectos de interese aos xa mencionados.