INFORME DE PRÁCTICAS

Repositorio de proxecto: https://github.com/rubenvigo/vvsTest
Participantes no proxecto: Felipe Bello Santos f.bello@udc.es
Rubén Vigo Lamas ruben.vigo@udc.es

Validación e Verificación de Software

Índice

1. Descripción do proxecto

O software consiste nunha aplicación web de apostas deportivas. Permite crear usuarios, que poderan buscar eventos e crear apostas sobre unha opcion de aposta. Dispon dun usuario administrador que pode xestionar o portal: crear eventos, crear tipos de aposta, crear opcions de aposta e seleccionar a opcion de aposta ganadora unha vez finalice o evento.

2. Estado actual

2.1. Funcionalidades

- **Rexistro de usuarios:** Debe permitir rexistrar novos usuarios, así como permitir cambiar a información de rexistro posteriormente.
- **Autenticación y salida:** Un usuario autenticarase indicando o seu seudónimo e contraseña, coa posibilidade de recordar a contraseña para non ter que tecleala a seguinte vez. O usuario poderá salir explícitamente do sitio Web, o que provoca que xa non se recorde a súa contraseña, no caso de que seleccionara dita opción con anterioridade.
- **Búsqueda de eventos:** Calquera usuario (aínda que non esteña autenticado) poderá buscar eventos que non empezaran, indicando as palabras clave do nome e da categoría (sendo ambos opcionales). As palabras clave teñen que estar todas contidas no nome do evento como palabras ou parte de palabras, sen distinguir entre maiúsculas e minúsculas, e en calquera orde. Se o usuario especifica a categoría, a búsqueda restrinxirase aos eventos de dita categoría. Se o usuario especifica a categoría, pero non as palabras clave, a búsqueda mostrará todos os eventos (que non empezaran) de esa categoría. Se o usuario non especifica nin as palabras clave nin a categoría, mostraranse todos os eventos (que non empezaran). Os eventos que aparecen como resultado dunha búsqueda mostranse ordenados por data de celebración, en orde ascendente, é dicir, primeiro os máis cercanos no tempo. Por cada evento mostrarase o seu nome, a categoría á que pertence e a data de celebración. O nome do evento estará asociado a un enlace que permitirá visualizar os detalles do evento, incluíndo todos os seus tipos de aposta e as opcións de aposta de cada una de elas.
- **Realizar unha aposta:** Cada opción de un tipo de aposta levará asociado un enlace que permitirá a un usuario autenticado apostar por esa opción. Si o usuario está autenticado, o enlace levarao a páxina co formulario que permite apostar por esa opción. Si non está autenticado, redirixiraseo ao formulario de autenticación, e tras autenticarse, valorarase positivamente que se mostre automáticamente o formulario para apostar por esa opción.
- **Consultar el estado de las apuestas:** Un usuario autenticado poderá consultar as apostas feitas ao largo do tempo, ordenadas por data de realización da aposta, en orden descendente, é decir, primero as máis recientes. Por cada aposta, mostrarase a data na que se fixo, o nome e data do evento, a pregunta do tipo de aposta, a opción elexida, a cantidade de diñeiro que apostou, a opción (ou opcions) gañadoras, unha indicación do seu estado (pendiente, gañada ou perdida), e en caso de que resultase ganadora, a ganancia (o que apostou por esa opción * cuota de ganancia de esa opción).

Deberá existir outro rol de usuario dentro da aplicación, ademais do de usuario normal, o rol de administrador. Que debe dispoñer das seguintes funcionalidades.

- **Inserción de eventos:** O sitio Web ofrecerá un enlace para añadir un novo evento.
- **Búsqueda de eventos** O sitio Web permitirá buscar eventos da misma forma que a un usuario normal. Sen embargo, a diferencia do usuario final, o resultado da búsqueda incluirá tanto eventos que comenzaran como eventos pendientes de comenzar.
- **Xestión de tipos de aposta** Cando o administrador fai clic sobre o nome de un dos eventos que aparecen no resultado da búsqueda, mostraselle a misma páxina que ao usuario final, cas seguintes diferencias:
  1. Si o evento non empezou, mostrarase un enlace para añadir un novo tipo de aposta.
  2. Si o evento xa empezpu, para cada tipo de aposta á que aínda non se lle especificaran a opción ou opciones gañadoras, mostrarase un enlace que permite seleccionalas.
  3. As opcions dos tipos de apostas non dispoñen de enlace para apostar. Para os tipos de aposta que xa teñan fixadas as opcions gañadoras, estas resaltanse de algunha maneira.

2.2. Responsables do desenvolvemento

- **Felipe Bello Santos**
  - f.bello@udc.es
  - Desarrollador
- **Raquel López Cacho**
  - raquel.lcacho@udc.es
  - Xefe de Proxecto
- **Rubén Vigo Lamas**
  - ruben.vigo@udc.es
  - Desarrollador

2.3. Responsables do proceso de proba

- **Felipe Bello Santos**
  - f.bello@udc.es
  - Tester
- **Rubén Vigo Lamas**
  - ruben.vigo@udc.es
  - Tester

2.4. Compoñentes avaliados

2.4.1. Probas unitarias

- UserProfileDao
  - **Probas obxetivo:** 8
  - **Probas realizadas:** 2
  - **Probas superadas:** 2
  - **Porcentaxe executada:** 25 %
  - **Porcentaxe de éxito:** 100 %
- EventoDao
  - **Probas obxetivo:** 28
  - **Probas realizadas:** 28
  - **Probas superadas:** 28
  - **Porcentaxe executada:** 100 %
  - **Porcentaxe de éxito:** 100 %
- CategoriaDao
  - **Probas obxetivo:** 7
  - **Probas realizadas:** 1
  - **Probas superadas:** 1
  - **Porcentaxe executada:** 15 %
  - **Porcentaxe de éxito:** 100 %
- ApuestaRealizadaDao
  - **Probas obxetivo:** 10
  - **Probas realizadas:** 6
  - **Probas superadas:** 6
  - **Porcentaxe executada:** 60 %
  - **Porcentaxe de éxito:** 100 %
- UserService
  - **Probas obxetivo:** 50
  - **Probas realizadas:** 36
  - **Probas superadas:** 36
  - **Porcentaxe executada:** 72 %
  - **Porcentaxe de éxito:** 100 %

2.4.2. Probas de integración

- UserService
  - **Probas obxetivo:** 40
  - **Probas realizadas:** 28
  - **Probas superadas:** 28
  - **Porcentaxe executada:** 70 %
  - **Porcentaxe de éxito:** 100 %

3. Especificación de probas

O diseño das probas encontrase no directorio doc no raíz do proxecto.
As probas de unidades están divididas en varios archivos, que seguen a seguinte nomenclatura: **pu_NomeModulo.txt**
Mentras que as probas de integración encontranse no archivo **pruebas_integracion.txt**

4. Rexistro de probas

Apoiamonos nas seguintes ferramentas para a realización das probas sobre a aplicación:

- JUnit : Para a implementación das probas de unidade e integración diseñadas, enlazadas no apartado 3.
- Checkstyle : Introducimos esta ferramenta no site de maven, de maneira que podemos ver o resultado da análise de caixa branca de forma clara sólo xenerando o site.
- Pitest : Utilizamos esta ferramenta de mutación de código, engadindo o plugin de maven no pom do proxecto e executandoo.
- Quickcheck : Utilizamos a xeneración aleatoria de valores para introducir a cantidade a apostar, de maneira que comprobamos que se executa de maneira correcta con valores dinámicos.
- Cobertura : Para xenerar a cobertura do noso proxecto introducimos de novo un plugin no pom e xeneramos no site dito análise de cobertura.
- VisualVM : Utilizamos esta ferramenta para realizar unha monitorización do comportamento da aplicación. Para simplificar esta parte de probas de estrés pola falta de tempo, só monitorizamos a aplicación durante a execución das probas que implementamos.
- GraphWalker: Estudiamos os posibles estados dos obxectos da nosa aplicación e probamos o correcto funcionamento dos métodos que realizan a transición entre ditos estados. O grafo pódese atopar no paquete *testautomation* no paquete de recursos do proxecto. Os estados que existen son o estado inicial, Creado e Eliminado. Desde o estado inicial existe unha transición hacia o estado Creado, que representa a inicialización do obxeto. Unha vez en Creado pódense executar *saveOrUpdate* ou *find* para permanecer no mesmo estado, ou *remove* para pasar o estado Eliminado, onde existen *find* e *remove* que nos manteñen no mesmo.

5. Rexistro de erros

1. Bug en apostar, a cantidade debe ser maior que 0
   - **Ferramenta/Fase:** Pruebas unidad
   - **Ciclo de vida:** Solucionado
2. Codigo innecesario na clase persistente TipoApuesta
   - **Ferramenta/Fase:** pitest
   - **Ciclo de vida:** Solucionado
3. Bug en IncorrectPasswordException
   - **Ferramenta/Fase:** pitest
   - **Ciclo de vida:** Solucionado
4. Bug en método existsOpcionApuesta
   - **Ferramenta/Fase:** Checkstyle
   - **Ciclo de vida:** Solucionado
5. Bug en IncorrectPasswordException
   - **Ferramenta/Fase:** Checkstyle
   - **Ciclo de vida:** Solucionado
6. Uso de paréntesis innecesario
   - **Ferramenta/Fase:** Checkstyle
   - **Ciclo de vida:** Solucionado
7. Sentencia catch vacía
   - **Ferramenta/Fase:** Checkstyle
   - **Ciclo de vida:** Solucionado
8. Errores de espaciado
   - **Ferramenta/Fase:** Checkstyle
   - **Ciclo de vida:** Solucionado
9. Complexidade demasiado alta
   - **Ferramenta/Fase:** FindBugs
   - **Ciclo de vida:** Solucionado
10. Atributos definidos non utilizados
    - **Ferramenta/Fase:** FindBugs
    - **Ciclo de vida:** Solucionado
11. Modificadores de privacidad innecesarios
    - **Ferramenta/Fase:** FindBugs
    - **Ciclo de vida:** Solucionado

12. Documentación inexistente
    - **Ferramenta/Fase:** Checkstyle
    - **Ciclo de vida:** Solucionado
13. Distribuición incorrecta de chaves
    - **Ferramenta/Fase:** Checkstyle
    - **Ciclo de vida:** Solucionado
14. Liñas de codigo demasiado extensas
    - **Ferramenta/Fase:** Checkstyle
    - **Ciclo de vida:** Non Solucionado
    - **Motivo:** Correxironse pero o erro persiste.
15. Non existe línea en blanco ó final das clases
    - **Ferramenta/Fase:** Checkstyle
    - **Ciclo de vida:** Non solucionado
    - **Motivo:** Correxíronse pero o erro non desaparece.
16. Valores numéricos no código
    - **Ferramenta/Fase:** Checkstyle
    - **Ciclo de vida:** Non solucionado
    - **Motivo:** Dito erro encontrase na clase *jcrypt*, que como non foi implementada polo equipo de desenvolvemento, non entra no alcance das nosas probas.
17. Erro en UserServiceImpl, no método addTipoApuesta
    - **Ferramenta/Fase:** Pitest
    - **Ciclo de vida:** Non solucionado
    - **Motivo:** O erro aparece porque Hibernate, cando detecta que un obxeto persistente foi modificado, actualiza dito obxeto na base de datos automáticamente. Aínda asi é recomendable realizar as actualizacións dos obxetos implicitamente. De esta maneira, Pitest detecta que suprimindo a liña na que realizamos a actualización non surxe ningún problema. Decidimos non correxir este erro e manter esa liña porque consideramos que son boas prácticas con Hibernate.

```
286
287     List<Long> idsOpciones = new ArrayList<Long>();
288 1   for (OpcionApuesta opcion : opciones) {
289         idsOpciones.add(opcion.getIdOpcionApuesta());
290     }
291 1   for (Long id : ganadoras) {
292 1       if (!idsOpciones.contains(id)) {
293             throw new InvalidOptionException("La opción con id "
294                     + id.toString()
295                     + " no pertenece al tipo de apuesta con id "
296                     + tipoApuestaId.toString());
297         }
298     }
299 1   if (tipoApuesta.getOpcionesApuesta().iterator().next().getEstado() != null) {
300         throw new OpcionApuestaAlreadySolvedException(
301                 "No es posible establecer opciones "
302                         + "ganadoras sobre obciones ya resueltas.");
303     }
304 3   if (!tipoApuesta.isMultiplesGanadoras() && ganadoras.size() > 1) {
305         throw new SimpleWinnerException(
306                 "No es posible establecer varias opciones "
307                         + "ganadoras, la pregunta tiene una unica opcion ganadora.");
308     }
309
310 1   for (OpcionApuesta opcion : opciones) {
311 1       if (ganadoras.contains(opcion.getIdOpcionApuesta())) {
312 1           opcion.setEstado(true);
313         } else {
314 1           opcion.setEstado(false);
315         }
316 1       opcionApuestaDao.save(opcion);
317     }
318 }
319
```

Figura 1: *Bug encontrado en UserServiceImpl.*

18. Erro no constructor de Evento
    - **Ferramenta/Fase:** Pitest
    - **Ciclo de vida:** Non solucionado
    - **Motivo:** A mutación de código informanos dun erro, que se mostra a continuación. As liñas erróneas foron implementadas polas diferencias de almacenamento de datas entre Java Calendar e Sql Date. A primeira ten moita máis exactitude, polo que se decidiu poñer a 0 os valores que non soporte o Sql, xa que neste contexto a exactitude non era necesaria. Non correximos este erro porque son decisións do equipo de desenvolvemento, e está a tan baixo nivel que non creemos que poida influenciar no correcto funcionamento da aplicación.

```
62      public Evento(final String nombre, final Calendar fecha,
63              final Categoria categoria) {
64          super();
65          this.nombre = nombre;
66  1       if (fecha != null) {
67  1           fecha.set(Calendar.SECOND, 0);
68  1           fecha.set(Calendar.MILLISECOND, 0);
69          }
70          this.fecha = fecha;
71          this.categoria = categoria;
72      }
```

Figura 2: *Bug encontrado en Evento.*

19. Erro en set de Evento
    - **Ferramenta/Fase:** Pitest
    - **Ciclo de vida:** Non solucionado
    - **Motivo:** O erro ten a misma lóxica que o anterior, polo que o motivo serve para ambos

```
133    */
134    public final void setFecha(final Calendar fecha) {
135 1    if (fecha != null) {
136 1        fecha.set(Calendar.SECOND, 0);
137 1        fecha.set(Calendar.MILLISECOND, 0);
138        }
139    this.fecha = fecha;
140    }
141
142    /**
```

Figura 3: *Bug encontrado en Evento.*

6. Estatísticas

- **Número de erros encontrados semanalmente**
  Cada commit realizado por membros do equipo de probas está asociado con unha tarefa, polo que as gráficas do repositorio poden ser un reflexo de como se xestionaron as tarefas.

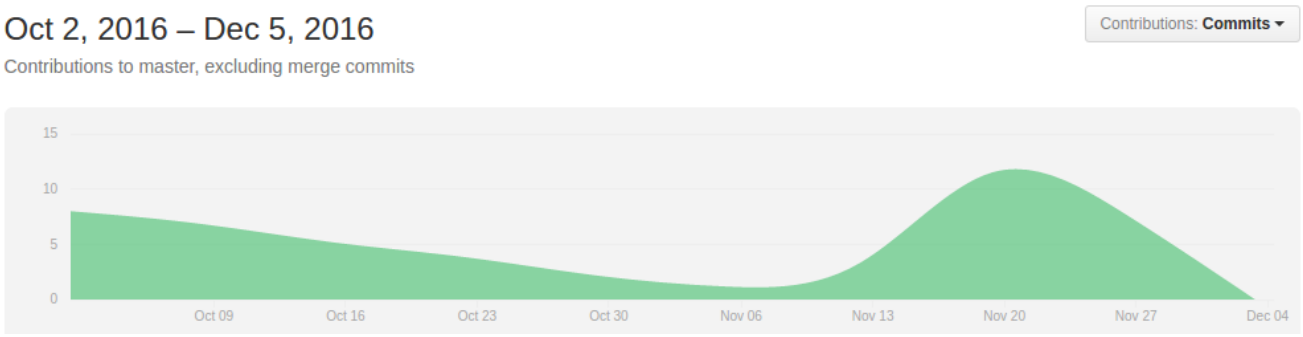


Figura 4: *Commits en el repositorio.*

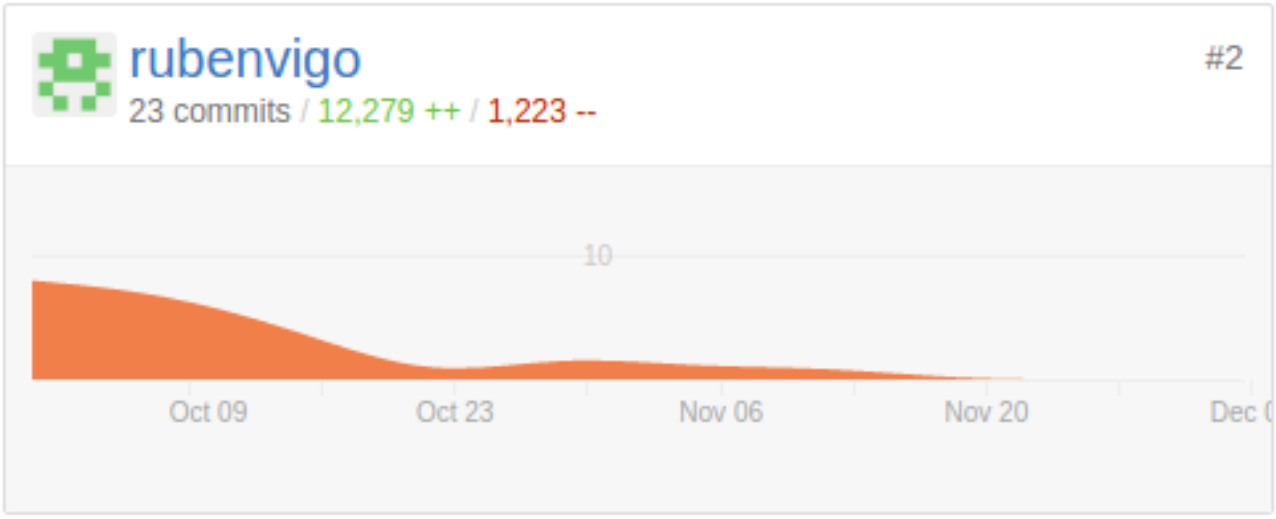


Figura 5: *Commits del usuario rubenvigo.*

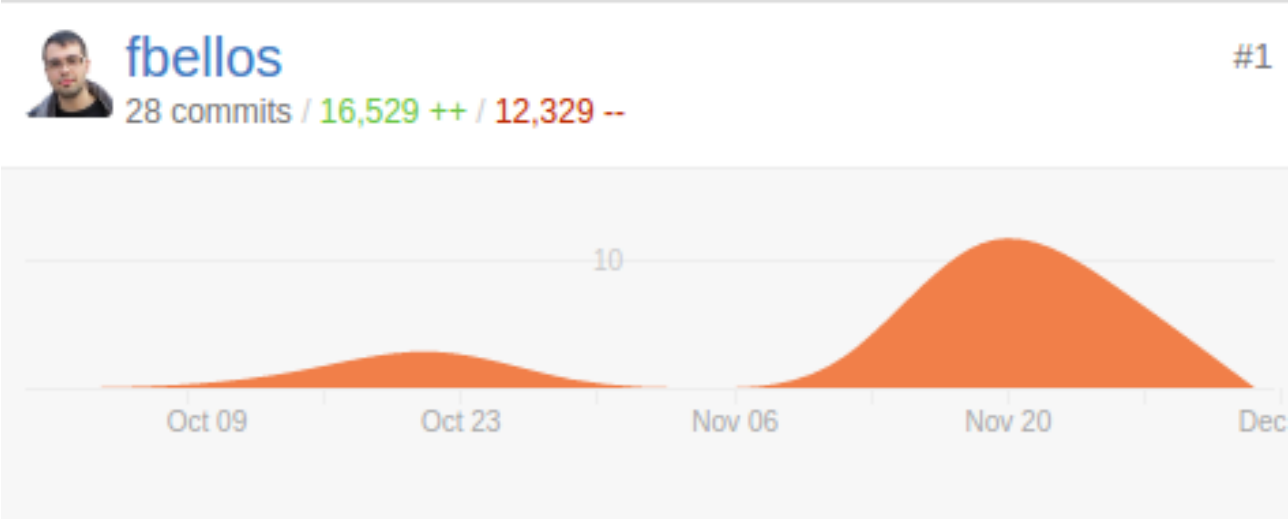


Figura 6: *Commits del usurio fbellos.*

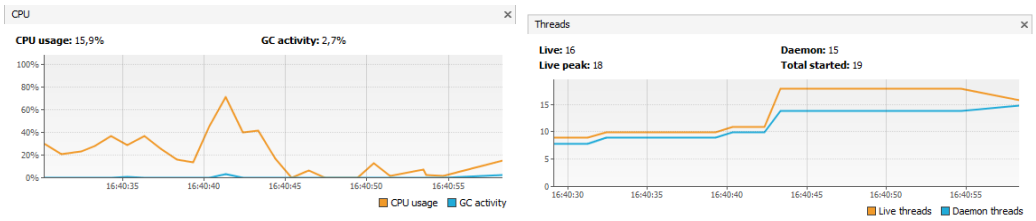
- **Nivel de progreso na execución das probas**

  - Número de probas diseñadas : 143
  - Porcentaxe implementado : 70,6 %
  - Porcentaxe executado : 70,6 %
  - Porcentaxe exitoso : 70,6 %
  - Porcentaxe con fallos : 0 %
  - Número de probas non deseñadas : 15
  - O análise de cobertura do proxecto excluíndo as clases e paquetes de axuda, dos que non realizamos probas. A media de cobertura de código e do 91 %, unha cobertura bastante alta, polo que creemos que o código está suficientemente probado.

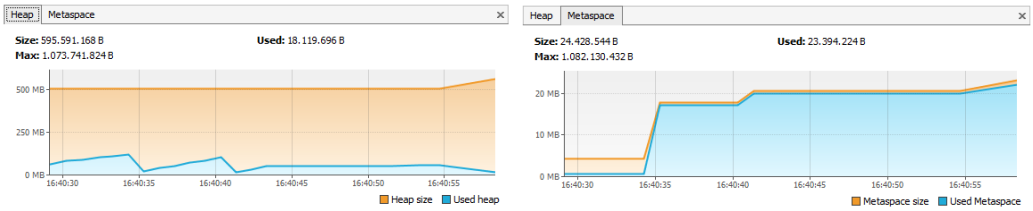
Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	34	91% 380/416	91% 143/156	1,662
es.udc.pa.pa002.practicapa.model.apuestarealizada	3	71% 20/28	N/A N/A	1
es.udc.pa.pa002.practicapa.model.categoria	3	84% 11/13	N/A N/A	1
es.udc.pa.pa002.practicapa.model.evento	3	99% 117/118	92% 87/94	3,45
es.udc.pa.pa002.practicapa.model.opcionapuesta	3	82% 19/23	N/A N/A	1
es.udc.pa.pa002.practicapa.model.tipoapuesta	3	84% 28/33	0% 0/4	1,214
es.udc.pa.pa002.practicapa.model.userprofile	3	91% 33/36	100% 2/2	1,118
es.udc.pa.pa002.practicapa.model.userservice	16	92% 152/165	96% 54/56	1,7

Figura 7: *Cobertura do proyecto.*

- Coa ferrammenta VisualVM monitorizamos o comportamento da nosa aplicación en execución. As probas execútanse nun PC con un procesador i5 de primeira xeración e con 8Gb de memoria RAM.



Como podemos observar nas graficas producese un pico de uso da CPU no intervalo de tempo [16:40:39-16:40:46] aproximadamente, este pico correspondese coa etapa de carga dos ficheiros de recursos,compilación das clases, creación das táboas e compilación das clases de test. No momento seguinte, correspondente a execución dos tests, prodúcese un aumento no número de threads e unha diminución importante no uso da CPU. Finalmente o número de threads diminúe según termina a fase de execución dos test.



Na primeira das gráficas observamos que os MB de heap usados diminúe concretamente no momento [16:40:40], esto é debido que se executou o recolector de basura. Este feito pódese observar na primeira das guas gráficas anteriores. Este feito volve a ocorrer o final da execución dos tests no momento [16:40:55].
Na ultima gráfica podemos observar que os MB de metaespace usados aumenta durante e execución das fases no momento de realizar mvn test. Aumenta de forma considerable no momento da carga dos recursos, compilación de clases,etc. E posteriormente volve a aumentar no momento da execución dos tests.

Análise do perfil de detección de erros

1. Bug en apostar, a cantidade debe ser maior que 0
   - **Localización:** AccountServiceImpl
   - **Tipoloxía:** Error en codificación, non fai algo que debería
   - **Criticidade:** Alta

2. Codigo innecesario na clase persistente TipoApuesta
    - **Localización:** TipoApuesta
    - **Tipoloxía:** Error en codificación, fai algo que non debería
    - **Criticidade:** Media
3. Bug en IncorrectPasswordException
    - **Localización:** IncorrectPasswordException
    - **Tipoloxía:** Error en codificación, non fai algo que debería.
    - **Criticidade:** Media
4. Bug en método existsOpcionApuesta
    - **Localización:** TipoApuesta
    - **Tipoloxía:** Código innecesario e complexo, ante futuros bugs pode chegar a ser un problema
    - **Criticidade:** Media
5. Bug en IncorrectPasswordException
    - **Localización:** IncorrectPasswordException
    - **Tipoloxía:**
    - **Criticidade:**
6. Uso de paréntesis innecesario
    - **Localización:**
        - EventoDaoHibernate
        - AnadirTipoApuesta
        - jcrypt
        - AnadirOpcionApuesta
        - UserServiceImpl
        - TipoApuestaCreada
        - EventoDetails
        - FindApuestasUser
        - TipoApuestaDetails
    - **Tipoloxía:** Código innecesario que pode xerar bugs futuros.
    - **Criticidade:** Baixa
7. Sentencia catch vacía
    - **Localización:**
        - AnadirTipoApuesta
        - AnadirOpcionApuesta
        - EspecificarGanadoras
        - TipoApuestaCreada
        - ApuestaCreada
        - EventoDetails

    - OpcionApuestaDetails
    - TipoApuestaDetails
    - OpcionApuestaEncoder
  - **Tipoloxía:** Fai algo que non debería, capturar unha excepción e non realizar ningunha acción.
  - **Criticidade:** Alta
8. Errores de espaciado
  - **Localización:** Todas las clases
  - **Tipoloxía:** Dificulta a comprensibilidade e a visión do código, o que pode ser fonte de bugs no futuro.
  - **Criticidade:** Baixa
9. Complexidade demasiado alta
  - **Localización:**
    - UserServiceImpl
    - FindApuestasUser
  - **Tipoloxía:** Fai algo máis complexo do que debería, en caso de bugs futuros complicaría a sua resolución
  - **Criticidade:** Alta
10. Atributos definidos non utilizados
  - **Localización:**
    - Layout
    - AnadirEvento
    - AnadirOpcionApuesta
    - EspecificarGanadoras
    - TipoApuestaCreada
    - FindEventos
    - FoundEventos
    - TipoApuestaDetails
  - **Tipoloxía:** Código innecesario, podería provocar bugs.
  - **Criticidade:** Media
11. Modificadores de privacidad innecesarios
  - **Localización:**
    - ApuestaRealizadaDao
    - EventoDao
    - UserProfileDao
    - UserService
  - **Tipoloxía:** Código innecesario, podería provocar bugs.
  - **Criticidade:** Media

12. Documentación inexistente
    - **Localización:** Todas las clases
    - **Tipoloxía:** Documentación inexistente, unha das causas máis importantes na aparición de bugs.
    - **Criticidade:** Media
13. Distribuición incorrecta de chaves
    - **Localización:**
        - Evento
        - jcrypt
        - EspecificarGanadoras
        - FindEventos
        - FoundEventos
        - AuthenticationPolicyWorker
        - AuthenticationValidator
        - OpcionApuestaEncoder
    - **Tipoloxía:** Dificulta a comprensibilidade e a visión do código, o que pode ser fonte de bugs no futuro.
    - **Criticidade:** Baixa
14. Liñas de codigo demasiado extensas
    - **Tipoloxía:** Dificulta a comprensibilidade e a visión do código, o que pode ser fonte de bugs no futuro.
    - **Criticidade:** Baixa
15. Non existe liña en blanco ó final das clases
    - **Localización:** Todas as clases
    - **Tipoloxía:** Dificulta a comprensibilidade e a visión do código, o que pode ser fonte de bugs no futuro.
    - **Criticidade:** Baixa
16. Valores numéricos no código
    - **Localización:** jcrypt
    - **Tipoloxía:** Dificulta a comprensibilidade e a visión do código, o que pode ser fonte de bugs no futuro.
    - **Criticidade:** Baixa
17. Erro actualización Hibernate
    - **Localización:** UserServiceImpl
    - **Tipoloxía:** O código é innecesario, xa que a función que realiza é automática.
    - **Criticidade:** Baixa

18. Erro no constructor de Evento
    - **Localización:** Evento
    - **Tipoloxía:** O código e difícilmente comprensible.
    - **Criticidade:** Baixa
19. Erro nun set de Evento
    - **Localización:** Evento
    - **Tipoloxía:** O código e difícilmente comprensible.
    - **Criticidade:** Baixa

Como se pode observar a clase que máis erros contén e en máis cantidade é a de jcrypt, unha clase na que nos apoiamos pero que non foi creada polo equipo de desenvolvemento. O resto de erros non están concentrados en ningún paquete ou clase especial, se non que están repartidos.

Segundo a súa tipoloxía, os erros de codificación céntranse no servicio do modelo, que é a clase con maior complexidade. Vemos que os erros que máis aparecen son de *código innecesario*, que correximos para facilitar a comprensibilidade do código e evitar bugs no caso de que haxa modificacións.

- **Informe de erros abertos e pechados por criticidade**
  Os erros con maior criticidade encóntranse no servicio da capa modelo. Trátanse de bugs de funcionalidades, que aparecen no servicio xa que é donde se implementa a lóxica de negocio da aplicación. Os erros con Media e Baixa criticidade trátanse de erros que actualmente non afectaban o funcionamento do sistema nin incumprían especificacións do mesmo pero que, co sistema en producción ou ante modificacións, poden levar a xerar bugs.
- **Avaliación global do estado de calidade e estabilidade actuais**

  Tras realizar o análisis da nosa práctica coas distintas ferramentas, detectar os problemas que estas indicaban, crear as respectivas issues e solucionalas podemos afirmar que o resultado é unha aplicación con máis calidade no código.

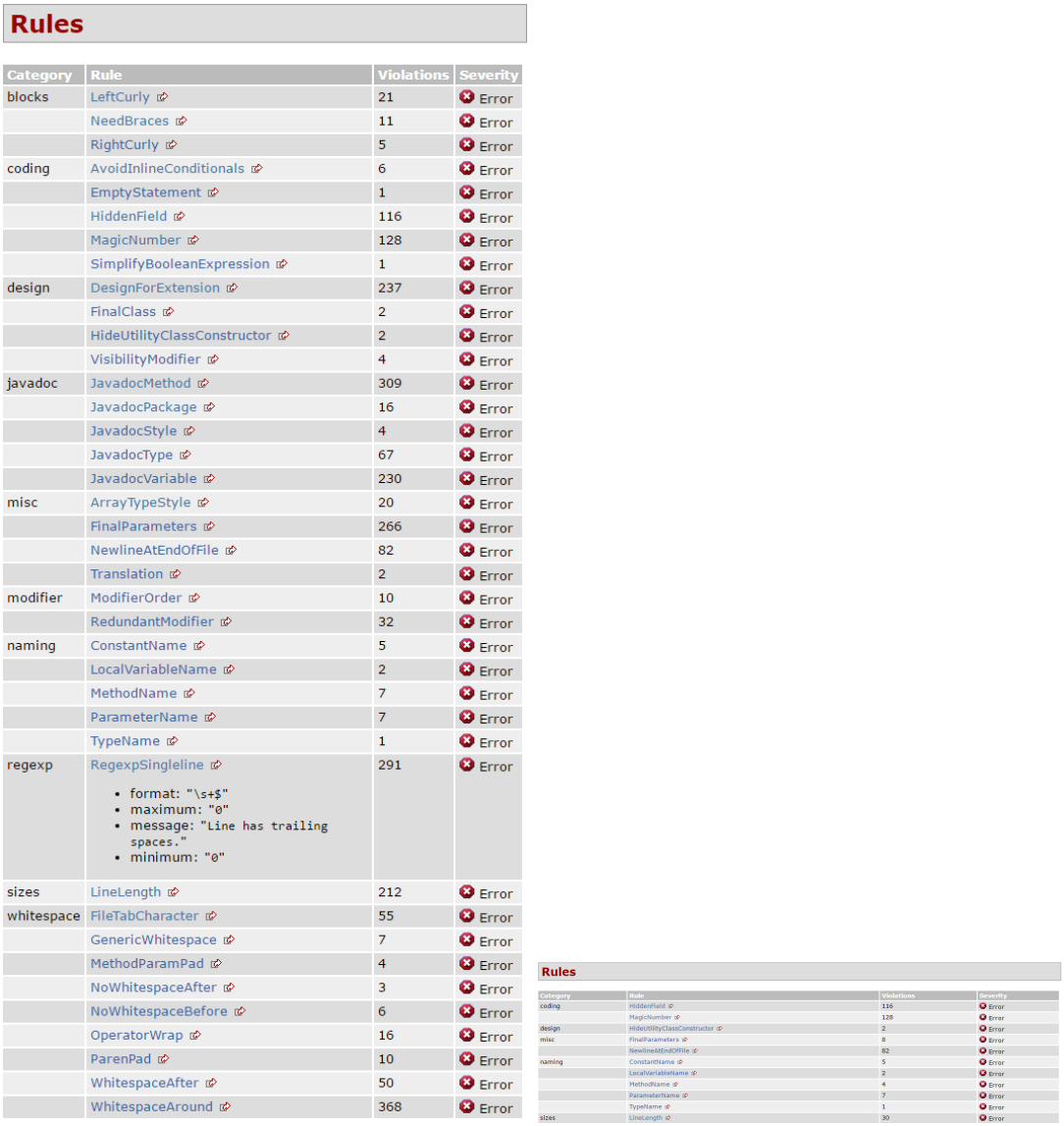
Summary

Files	Info	Warnings	Errors
105	0	0	2616

Summary

Files	Info	Warnings	Errors
121	0	0	385

Podemos ver que a ferramenta checkstyle pasa de detectar uns 2616 Erros a detectar uns 385 Erros.



Rules

Category	Rule	Violations	Severity
blocks	LeftCurly	21	Error
	NeedBraces	11	Error
	RightCurly	5	Error
coding	AvoidInlineConditionals	6	Error
	EmptyStatement	1	Error
	HiddenField	116	Error
	MagicNumber	128	Error
	SimplifyBooleanExpression	1	Error
design	DesignForExtension	237	Error
	FinalClass	2	Error
	HideUtilityClassConstructor	2	Error
	VisibilityModifier	4	Error
javadoc	JavadocMethod	309	Error
	JavadocPackage	16	Error
	JavadocStyle	4	Error
	JavadocType	67	Error
	JavadocVariable	230	Error
misc	ArrayTypeStyle	20	Error
	FinalParameters	266	Error
	NewlineAtEndOfFile	82	Error
	Translation	2	Error
modifier	ModifierOrder	10	Error
	RedundantModifier	32	Error
naming	ConstantName	5	Error
	LocalVariableName	2	Error
	MethodName	7	Error
	ParameterName	7	Error
	TypeName	1	Error
regexp	RegexpSingleline • format: "\s+$" • maximum: "0" • message: "Line has trailing spaces." • minimum: "0"	291	Error
sizes	LineLength	212	Error
whitespace	FileTabCharacter	55	Error
	GenericWhitespace	7	Error
	MethodParamPad	4	Error
	NoWhitespaceAfter	3	Error
	NoWhitespaceBefore	6	Error
	OperatorWrap	16	Error
	ParenPad	10	Error
	WhitespaceAfter	50	Error
	WhitespaceAround	368	Error

Rules

Figura 8: *Tipos de erros despois de solucionalos.*

O número de regras quebrantadas tamén disminue. Como explicamos anteriormente, a pesar de correxir os erros de coding e sizes con eclipse, Checkstyle non o entende así e marcaos como non solucionados.

Ademais, quedan sen solucionar os fallos de *finalParameters* y *naming*, que se encontran na clase *jcrypt*. Dita clase non foi implementada por nos e entendemos que non debemos correxir todos os errors detectados en ela.

7. Outros aspectos de interese

- Como se explicou anteriormente, utilizamos a ferramenta GraphWalker sobre os estados dos obxectos do proxecto. Sen embargo, non conseguimos executar a proba correctamente. Tras crear o grafo dirixido cos estados, neste caso do obxecto Categoria, xeramos mediante a ferramenta unha interfaz cos estados e transicións establecidos. Implementamos a interfaz, chamando os métodos implicados, no caso das transicións, e realizando as asercióǹs que indiquen o correcto funcionamento no caso dos estados. O problema xorde ca Inxección de Dependencias realizada por Spring. Para chamar os métodos necesitamos a clase que os implementa, e esta á sua vez necesita doutras clases, que deberían ser inxectadas por Spring. Pero non se realiza correctamente, a pesar de que a configuración da clase de probas é correcta. Ademais, executando os tests da aplicación non se permite executar dita proba, polo que a excluímos da execución e adxuntamos unha imaxe conforme en local funciona, executando *mvn graphwalker:test*

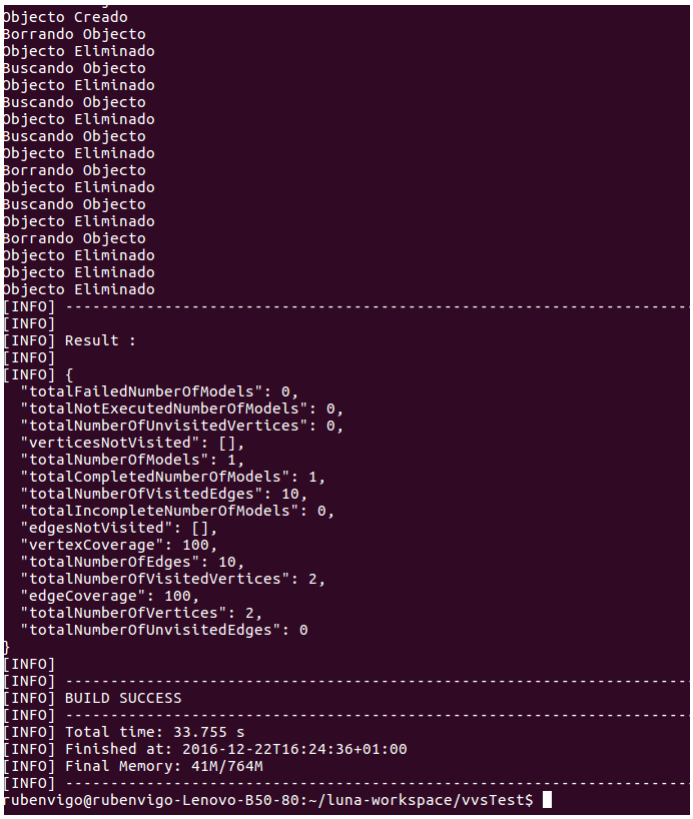


Figura 9: *Resultado execución test GraphWalker.*

- O uso da ferramenta VisualVM que facemos non é o óptimo para explotar as súas posibilidades. O ideal sería realizar un plan de probas con JMeter, simulando conexións de usuarios reais, realizando picos de usuarios, carga constante de peticións ou aumentos progresivos. E monitorizar como se comportan os aspectos dun ordenador no que se executa a aplicación.