

---

# INFORME DE PRÁCTICAS

*Repositorio de proxecto: <https://github.com/gemamb/VVS>*

*Participantes no proxecto: Raquel López Cacho e Gema Mariño Bodelón*

---

Validación e Verificación de Software

# Índice

<b>1. Descripción do proxecto</b>	<b>3</b>
<b>2. Estado actual</b>	<b>3</b>
2.1. Componentes avaliados . . . . .	4
<b>3. Especificación de probas</b>	<b>6</b>
3.1. Test de unidade . . . . .	6
3.1.1. EventDao . . . . .	7
3.1.2. CategoryDao . . . . .	19
3.1.3. BetTypeDao . . . . .	20
3.1.4. BetDao . . . . .	21
3.1.5. UserProfileDao . . . . .	23
3.1.6. BetServiceImpl . . . . .	24
3.2. Test de integración . . . . .	49
3.2.1. Escenarios . . . . .	50
3.2.2. Casos de proba simples . . . . .	51
3.2.3. Selección de datos aleatoria . . . . .	59
<b>4. Rexisto de probas</b>	<b>61</b>
<b>5. Rexistro de erros</b>	<b>61</b>
5.1. JUnit e Mocking . . . . .	61
5.2. CheckStyle . . . . .	61
5.3. PITest . . . . .	64
5.4. GraphWalker . . . . .	64
5.5. Cobertura . . . . .	65
<b>6. Estatísticas</b>	<b>67</b>
6.1. Número de erros atopados . . . . .	67
6.2. Nivel de progreso na execución das probas . . . . .	68
6.3. Análise do perfil de detección de erros . . . . .	71
6.4. Informe de erros abertos e pechados por nivel de criticidade . . . . .	71
6.5. Avaliación global do estado de calidade e estabilidade actuais . . . . .	72

## 1. Descrición do proxecto

Este software trátase dunha aplicación web de apostas deportivas. Permite visualizar eventos de distintas categorías, ver os tipos de apostas dispoñibles e as súas respectivas opcións de aposta.

Contémplanse dous tipos de usuarios:

**ADMIN** É o responsable de crear os eventos, os tipos de aposta e as súas opcións asociadas. Establece tras a finalización dos eventos, os resultados dos mesmos, marcando ganadas e perdas as opcións que estiveran dispoñibles. Ademais destas responsabilidades, é o único rol capaz de obter todos os eventos, incluídos os xa pasados.

**USER** Este é o rol da persoa que visita a nosa aplicación para buscar os eventos próximos e realizar apostas sobre os mesmos. Terá un apartado adicado ós resultados e estados das apostas feitas ó longo da súa actividade na aplicación.

## 2. Estado actual

A nosa aplicación componse de tres tipos de compoñentes:

### 1. Entidades e DAOs

- a) Bet
- b) BetOption
- c) BetType
- d) Category
- e) Event
- f) UserProfile

### 2. Servizos

- a) **UserService**: contén os casos de uso referentes á xestión de contas de usuarios do sistema:
  - Rexistro de usuarios
  - Logueo do usuario no sistema
  - Actualizar a información do usuario
  - Cambiar o contrasinal do usuario
  - Buscar usuarios polo seu identificador
- b) **BetService**: contén os casos de uso referentes ás funcionalidades propias do sistema, diferenciando as dispoñibles para o usuario e o administrador:
  - Insertar eventos
  - Buscar eventos (especificando opcionalmente palabras clave e categoría)
  - Insertar tipos de aposta e as súas respectivas opcións
  - Apostar por unha opción de aposta

- Actualizar o estado das opcións de aposta tras a finalización do evento
  - Visualizar as apostas realizadas do usuario
  - Buscar as entidades polo seu identificador
3. Capa web: esta capa foi implementada empregando o framework Tapestry, polo que consta de unha serie de compoñentes e páxinas coa súa correspondente clase Java asociada. Para axustarnos ao tempo dispoñible para a realización desta práctica decidimos non avaliar este módulo. Os elementos que o compoñen están agrupados en paquetes segundo a funcionalidade que implementan.
- a) **management**: contén as plantillas e páxinas que recollen as funcionalidades de insercións de evento, tipo de aposta e opción de aposta.
  - b) **search**: agrupa as funcionalidades de búsqueda de eventos e apostas, e tamén as de realizar aposta e marcar opcións gañadoras.
  - c) **user**: neste paquete atopamos as funcionalidades de rexistro e autenticación de usuarios, así como as que permiten modificar os datos dos mesmos.
  - d) **services**: aquí atópanse compoñentes necesarios nas demais páxinas, relacionados coa política de privacidade das mesmas.
  - e) **util**: este paquete é unha especie de "caixón desastre", no que se agrupan compoñentes empregados en varias páxinas, ou que serven de apoio para unha operación en concreto.

## 2.1. Compoñentes avaliados

Para esta práctica decidimos avaliar a capa de acceso a datos (DAOs) e a capa de servizos, centrándonos só nos módulos implementados por nós. Polo tanto, non se realizaron probas sobre os elementos que pertencían ao arquetipo inicial.

1. DAOs: todos os DAOs da nosa aplicación heredan dun DAO xenérico que non foi implementado por nós, no cal se atopan as operacións CRUD. Por este motivo, e por razóns de tempo, decidimos realizar probas sobre estas funcionalidades só na entidade **Event**, asumindo que se funcionan correctamente para un caso tamén o farán para os demais. Todos os tests contabilizados neste apartado son de unidade.
  - a) **Bet**: este compoñente recolle as operacións sobre a entidade *aposta*. Os métodos probados son **findBetsByUserId** e **findBetsByUserIdNumber** (cuxas funcionalidades se explican no apartado de deseño de probas).
    - Número de probas obxectivo: 16
    - Número de probas preparadas: 16
    - Porcentaxe de probas executadas: 100 %
    - Porcentaxe de probas superadas: 100 %
  - b) **BetType**: este compoñente recolle as operacións sobre a entidade *tipo de aposta*. O método probado é **findDuplicates** (cuxa funcionalidade se explica no apartado de deseño de probas).

- Número de probas obxectivo: 23
  - Número de probas preparadas: 23
  - Porcentaxe de probas executadas: 100 %
  - Porcentaxe de probas superadas: 100 %
- c) **BetOption**: este compoñente recolle as operacións sobre a entidade opción de aposta. Non implementa ningunha operación a maiores das CRUD, polo que non se deseñou ningún caso de proba específico.
- d) **Category**: este compoñente recolle as operacións sobre a entidade categoría. O método probado é **findCategories** (cuxa funcionalidade se explica no apartado de deseño de probas).
- Número de probas obxectivo: 27
  - Número de probas preparadas: 27
  - Porcentaxe de probas executadas: 100 %
  - Porcentaxe de probas superadas: 100 %
- e) **Event**: este compoñente recolle as operacións sobre a entidade evento. Aquí probamos os métodos CRUD (**save**, **remove**, **update** e **find**), e tamén os métodos específicos da entidade **getNumberOfEvents**, **findEvents** e **findDuplicates** (cuxas funcionalidades se explican no apartado de deseño de probas).
- Número de probas obxectivo: 28
  - Número de probas preparadas: 28
  - Porcentaxe de probas executadas: 100 %
  - Porcentaxe de probas superadas: 100 %
- f) **UserProfile**: este compoñente recolle as operacións sobre a entidade usuario. O método probado é **findByLoginName** (cuxa funcionalidade se explica no apartado de deseño de probas).
- Número de probas obxectivo: 11
  - Número de probas preparadas: 11
  - Porcentaxe de probas executadas: 100 %
  - Porcentaxe de probas superadas: 100 %
2. Servizos: dispoñemos de dous servizos na nosa aplicación, diferenciando entre **BetService**, que foi a nosa lóxica de negocio e polo tanto implementada por nós, e **UserService**, que pertence ó arquetipo proporcionado para a asignatura. Por razóns de tempo decidimos unicamente probar o primeiro módulo, asumindo que o servizo de usuarios compórtase de maneira correcta.
- a) **BetService**: este compoñente recolle os casos de uso relacionados coa creación, xestión e consulta de eventos, tipos de aposta, opcións de aposta e apostas. Os métodos probados son:

- findEventsGetNumber
- findEvents
- makeBet
- findBets
- insertEvent
- insertBetType
- findEvent
- findBetType
- findBetOption
- checkOptions
- findCategories
- findCategory
- findDuplicates
- findBetsByUserIdNumber

En canto ao número de probas executadas para este compoñente temos os seguintes datos:

- Número de probas obxectivo: 70
- Número de probas preparadas: 70
- Porcentaxe de probas executadas: 100 %
- Porcentaxe de probas superadas: 100 %

### 3. Especificación de probas

#### 3.1. Test de unidade

Para unha mellor organización, dividimos o deseño das probas a realizar segundo ó compoñente a avaliar. Á súa vez, para cada método a probar mostramos a súa especificación para un mellor entendemento do mesmo.

Como xa se comentou no apartado anterior, decidimos non realizar as probas dos métodos CRUD dos DAOs de cada entidade, senón só os da clase **EventDao**.

### 3.1.1. EventDao

#### CU-001 save

##### Parametros

- entity: objeto a persistir en la base de datos (sin id) o actualizar (con id)

##### Excepciones

- NA

##### Valor retorno

- void

```
public void save(E entity)
```

#### PR-UN-001 save

##### Motivación

- Crear un evento

##### Parámetros

- entity: evento sin id

##### Valor retorno

- void

##### Inicialización

- 

#### PR-UN-002 save

##### Motivación

- Actualizar campos de un evento

##### Parámetros

- entity: Evento ya almacenado en bd (con id)

##### Valor retorno

- void

##### Inicialización

- 

#### CU-002 remove

##### Parametros

- key: clave primaria del evento a eliminar

##### Excepciones

- `InstanceNotFoundException`: salta cuando se quiere eliminar un evento que no existe

#### **Valor retorno**

- `void`

```
public void remove(PK id) throws InstanceNotFoundException
```

#### **PR-UN-003** `remove`

##### **Motivación**

- Borrar un evento

##### **Parámetros**

- `key`: id de un evento almacenado en la base de datos

##### **Valor retorno**

- `void`

##### **Inicialización**

- 

#### **PR-UN-004** `remove`

##### **Motivación**

- La aplicación debe romper con `InstanceNotfoundException`

##### **Parámetros**

- `key`: identificador de un evento inexistente

##### **Valor retorno**

- `InstanceNotfoundException`

##### **Inicialización**

- 

#### **CU-003** `find`

##### **Parametros**

- `key`: clave primaria del evento buscado

##### **Excepciones**

- `InstanceNotFoundException`: salta cuando se quiere buscar un evento que no existe

##### **Valor retorno**

- Evento con la clave primaria indicada

```
public E find(PK id) throws InstanceNotFoundException
```



#### **PR-UN-005** find

##### **Motivación**

- Buscar un evento

##### **Parámetros**

- key: id de un evento almacenado en la base de datos

##### **Valor retorno**

- Evento con el id indicado

##### **Inicialización**

- 

#### **PR-UN-006** find

##### **Motivación**

- La aplicación debe romper con InstanceNotFoundException

##### **Parámetros**

- key: id de un evento inexistente

##### **Valor retorno**

- InstanceNotFoundException

##### **Inicialización**

- 

#### **CU-004** getNumberOfEvents

##### **Parametros**

- keyWords: palabras clave para filtrar la búsqueda de eventos
- categoryId: identificador de la categoría por la que filtrar la búsqueda de eventos
- admin: criterio para devolver todos los eventos o únicamente eventos posteriores a la fecha actual

##### **Excepciones**

- NA

##### **Valor retorno**

- Número de eventos que coinciden con los criterios de búsqueda

```
public int getNumberOfEvents(String keywords, Long categoryId, boolean admin)
```

#### **PR-UN-007** getNumberOfEvents

##### **Motivación**

- Obtener el número de todos los eventos existentes

**Parámetros**

- keyWords: null
- categoryId: null
- admin: true

**Valor retorno**

- Número de eventos existentes

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-008** getNumberOfEvents**Motivación**

- Obtener el número de todos los eventos con fecha posterior a la actual

**Parámetros**

- keyWords: null
- categoryId: null
- admin: false

**Valor retorno**

- Número de eventos existentes con fecha posterior a la actual

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-009** getNumberOfEvents**Motivación**

- Obtener el número de eventos de categoría Baloncesto

**Parámetros**

- keyWords: null
- categoryId: identificador de la categoría Baloncesto
- admin: true

**Valor retorno**

- Número de eventos de Baloncesto

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-010** getNumberOfEvents

**Motivación**

- Obtener número de eventos de una categoría no existente, siendo 0

**Parámetros**

- keyWords: null
- categoryId: categoryId: identificador de una categoría no existente
- admin: true

**Valor retorno**

- 0

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-011** getNumberOfEvents**Motivación**

- Obtener el número de eventos cuyo nombre tenga "Madrid", buscando con minúsculas

**Parámetros**

- keyWords: madrid
- categoryId: null
- admin: true

**Valor retorno**

- Número de eventos relacionados con "Madrid"

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-012** getNumberOfEvents**Motivación**

- Obtener el número de eventos cuyo nombre tenga "Madrid", buscando con parte del nombre

**Parámetros**

- keyWords: adri
- categoryId: null
- admin: true

**Valor retorno**

- Número de eventos relacionados con "Madrid"

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

#### **PR-UN-013** `getNumberOfEvents`

##### **Motivación**

- Obtener el número de eventos Real Madrid - Barcelona”, buscando en mayúsculas

##### **Parámetros**

- `keyWords`: MADRID - BARCELONA
- `categoryId`: null
- `admin`: true

##### **Valor retorno**

- Número de eventos Real Madrid - Barcelona”

##### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

#### **PR-UN-014** `getNumberOfEvents`

##### **Motivación**

- Obtener el número de eventos Real Madrid - Barcelona”, buscando con diferente orden

##### **Parámetros**

- `keyWords`: BARCELONA - MADRID
- `categoryId`: null
- `admin`: true

##### **Valor retorno**

- Número de eventos Real Madrid - Barcelona”

##### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

#### **PR-UN-015** `getNumberOfEvents`

##### **Motivación**

- Obtener número de eventos buscando por palabras clave no coincidentes, siendo 0

##### **Parámetros**

- `keyWords`: BarcMad

- categoryId: null
- admin: true

#### **Valor retorno**

- 0

#### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

### **PR-UN-016** getNumberOfEvents

#### **Motivación**

- Obtener el número de eventos Real Madrid”únicamente de baloncesto

#### **Parámetros**

- keyWords: Madrid
- categoryId: identificador de la categoría Baloncesto
- admin: true

#### **Valor retorno**

- Número de eventos del Real Madrid de baloncesto

#### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

### **CU-005** findEvents

#### **Parametros**

- keyWords: palabras clave para filtrar la búsqueda de eventos
- categoryId: identificador de la categoría por la que filtrar la búsqueda de eventos
- startIndex número que determina la paginación
- count número total de eventos que devuelve el método
- admin: criterio para devolver todos los eventos o únicamente eventos posteriores a la fecha actual

#### **Excepciones**

- NA

#### **Valor retorno**

- Lista de los eventos que coinciden con los criterios de búsqueda

```
public List<Event>findEvents(String keywords, Long categoryId, int startIndex,
int count, boolean admin)
```

### **PR-UN-017** findEvents

**Motivación**

- Obtener todos los eventos existentes

**Parámetros**

- keyWords: null
- categoryId: null
- startIndex: 0
- count: 10
- admin: true

**Valor retorno**

- Lista de los eventos existentes

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-018** findEvents**Motivación**

- Obtener todos los eventos con fecha posterior a la actual

**Parámetros**

- keyWords: null
- categoryId: null
- startIndex: 0
- count: 10
- admin: false

**Valor retorno**

- Lista de los eventos existentes con fecha posterior a la actual

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-019** findEvents**Motivación**

- Obtener todos los eventos de categoría Baloncesto

**Parámetros**

- keyWords: null
- categoryId: identificador de la categoría Baloncesto
- startIndex: 0
- count: 10

- admin: true

**Valor retorno**

- Lista de los eventos de baloncesto

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-020** findEvents**Motivación**

- Obtener todos los eventos de categoría no existente, ninguno

**Parámetros**

- keyWords: null
- categoryId: identificador de una categoría no existente
- startIndex: 0
- count: 10
- admin: true

**Valor retorno**

- Lista vacía

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

**PR-UN-021** findEvents**Motivación**

- Obtener todos los eventos cuyo nombre tenga "Madrid", buscando con minúsculas

**Parámetros**

- keyWords: madrid
- categoryId: null
- startIndex: 0
- count: 10
- admin: true

**Valor retorno**

- Lista de eventos relacionados con "Madrid"

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos

#### **PR-UN-022** findEvents

##### **Motivación**

- Obtener todos los eventos cuyo nombre tenga "Madrid", buscando con parte del nombre

##### **Parámetros**

- keyWords: adri
- categoryId: null
- startIndex: 0
- count: 10
- admin: true

##### **Valor retorno**

- Lista de eventos relacionados con "Madrid"

##### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

#### **PR-UN-023** findEvents

##### **Motivación**

- Obtener los eventos Real Madrid - Barcelona", buscando en mayúsculas

##### **Parámetros**

- keyWords: MADRID - BARCELONA
- categoryId: null
- startIndex: 0
- count: 10
- admin: true

##### **Valor retorno**

- Lista de eventos Real Madrid - Barcelona"

##### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

#### **PR-UN-024** findEvents

##### **Motivación**

- Obtener los eventos Real Madrid - Barcelona", buscando con diferente orden

##### **Parámetros**

- keyWords: BARCELONA - MADRID
- categoryId: null



- startIndex: 0
- count: 10
- admin: true

#### **Valor retorno**

- Lista de eventos Real Madrid - Barcelona”

#### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

### **PR-UN-025 findEvents**

#### **Motivación**

- No obtener resultados coincidentes, buscando por palabras clave no coincidentes

#### **Parámetros**

- keyWords: BarcMad
- categoryId: null
- startIndex: 0
- count: 10
- admin: true

#### **Valor retorno**

- Lista vacía

#### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

### **PR-UN-026 findEvents**

#### **Motivación**

- Obtener los eventos Real Madrid”únicamente de baloncesto

#### **Parámetros**

- keyWords: Madrid
- categoryId: null
- startIndex: 0
- count: 10
- admin: true

#### **Valor retorno**

- Lista de eventos del Real Madrid de baloncesto

#### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

#### **CU-006** findDuplicates

##### **Parametros**

- fullName: nombre del evento

##### **Excepciones**

- NA

##### **Valor retorno**

- Valor booleano según si existen más eventos con el mismo nombre

`public boolean findDuplicates(String fullName)`

#### **PR-UN-027** findDuplicates

##### **Motivación**

- Obtener valor true con nombre igual a evento existente

##### **Parámetros**

- fullName: "Real Madrid - Barcelona"

##### **Valor retorno**

- true

##### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

#### **PR-UN-028** findDuplicates

##### **Motivación**

- Obtener valor false porque no existe ningún evento con este nombre

##### **Parámetros**

- fullName: "Barcelona - Madrid"

##### **Valor retorno**

- false

##### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos

### 3.1.2. CategoryDao

#### CU-007 findCategories

##### Parametros

- NA

##### Excepciones

- NA

##### Valor retorno

- Lista de las categorías existentes en la base de datos

```
public List<Category>findCategories()
```

#### PR-UN-029 findCategories

##### Motivación

- Obtener las categorías existentes en la base de datos

##### Parámetros

- NA

##### Valor retorno

- Lista con dos categorías almacenadas

##### Inicialización

- Creación de dos categorías

#### PR-UN-031 findCategories

##### Motivación

- No obtener categorías

##### Parámetros

- NA

##### Valor retorno

- Lista vacía

##### Inicialización

- NA

### 3.1.3. BetTypeDao

#### CU-008 findDuplicates

##### Parametros

- eventId: identificador del evento del tipo de apuesta a comprobar
- fullName: pregunta del tipo de apuesta

##### Excepciones

- NA

##### Valor retorno

- Valor booleano según si existen más tipos de apuesta con la misma pregunta

```
public boolean findDuplicates(Long eventId, String fullName)
```

#### PR-UN-031 findDuplicates

##### Motivación

- Comprobar que no existe un tipo de apuesta con la misma pregunta

##### Parámetros

- eventId: identificador de un evento existente
- fullName: nombre de tipo de apuesta no existente

##### Valor retorno

- true

##### Inicialización

- 

#### PR-UN-032 findDuplicates

##### Motivación

- Comprobar que existe un tipo de apuesta con la misma pregunta

##### Parámetros

- eventId: identificador de un evento existente
- fullName: nombre de tipo de apuesta existente

##### Valor retorno

- false

##### Inicialización

-

### 3.1.4. BetDao

#### CU-009 findBetsByUserId

##### Parametros

- key: clave primaria del usuario propietario de las apuestas
- startIndex: número que determina la paginación
- count: número total de eventos que devuelve el método

##### Excepciones

- NA

##### Valor retorno

- Lista con las apuestas del usuario

```
public List<Bet>findBetsByUserId(Long userId, int startIndex, int count)
```

#### PR-UN-033 findBetsByUserId

##### Motivación

- Obtener la lista de apuestas de un usuario

##### Parámetros

- key: identificador de usuario con apuestas
- startIndex: 0
- count: 1

##### Valor retorno

- Lista de apuestas pertenecientes al usuario

##### Inicialización

- 

#### PR-UN-034 findBetsByUserId

##### Motivación

- Obtener lista vacía si un usuario no tiene apuestas

##### Parámetros

- key: identificador de usuario sin apuestas
- startIndex: 0
- count: 1

##### Valor retorno

- Lista vacía

##### Inicialización

-

#### **PR-UN-035** findBetsByUserId

##### **Motivación**

- Obtener lista vacía si el usuario no existe

##### **Parámetros**

- key: identificador de usuario no existente
- startIndex: 0
- count: 1

##### **Valor retorno**

- Lista vacía

##### **Inicialización**

- 

#### **CU-010** findBetsByUserIdNumber

##### **Parametros**

- key: clave primaria del usuario propietario de las apuestas

##### **Excepciones**

- NA

##### **Valor retorno**

- Número de apuestas del usuario

```
public int findBetsByUserIdNumber(Long userId)
```

#### **PR-UN-036** findBetsByUserIdNumber

##### **Motivación**

- Obtener el número de apuestas de un usuario sin apuestas

##### **Parámetros**

- key: identificador de usuario sin apuestas

##### **Valor retorno**

- 0

##### **Inicialización**

- Creación de un usuario sin apuestas

#### **PR-UN-037** findBetsByUserIdNumber

##### **Motivación**

- Obtener el número de apuestas si el usuario no existe, siendo 0

##### **Parámetros**

- key: identificador de usuario no existente

**Valor retorno**

- 0

**Inicialización**

- NA

**PR-UN-038** findBetsByIdNumber**Motivación**

- Obtener la lista de apuestas de un usuario

**Parámetros**

- key: identificador de usuario con apuestas

**Valor retorno**

- Número de apuestas pertenecientes al usuario

**Inicialización**

- Creación de dos categorías
- Creación de tres eventos
- Creación de un tipo de apuesta perteneciente a uno de los eventos
- Creación de dos opciones de apuesta pertenecientes al tipo de apuesta creado
- Creación de dos apuestas, una a cada opción almacenada
- Creación de un usuario, creador de las dos apuestas

**3.1.5. UserProfileDao****CU-011** findByLoginName**Parametros**

- loginName: login de un usuario

**Excepciones**

- InstanceNotFoundException: salta cuando no se encuentra el usuario

**Valor retorno**

- Usuario existente con la clave indicada

```
public UserProfile findByLoginName(String loginName) throws  
InstanceNotFoundException
```

**PR-UN-039** findByLoginName**Motivación**

- Obtener los datos del usuario con la clave indicada

**Parámetros**

- loginName: login de un usuario existente

**Valor retorno**

- Usuario coincidente con el login indicado

**Inicialización**

- 

**PR-UN-040** findByLoginName**Motivación**

- Obtener InstanceNotFoundException pues no se encuentra el usuario a partir del login

**Parámetros**

- loginName: login de un usuario no existente

**Valor retorno**

- InstanceNotFoundException

**Inicialización**

- 

**3.1.6. BetServiceImpl****CU-012** findEventsGetNumber

Método que devuelve el número de eventos coincidentes con los parámetros de búsqueda.

**Parametros**

- keyWords: palabras clave que tendrán que estar contenidas en el nombre de los eventos buscados
- categoryId: identificador de la categoría a la que pertenecen los eventos buscados
- admin: booleano que indica si la búsqueda se realiza para el usuario administrador o para un usuario normal. Si es admin se buscarán todos los eventos. En caso contrario sólo aquellos cuya fecha de inicio no haya pasado

**Excepciones**

- NA

**Valor retorno**

- Número de eventos que coinciden con los parámetros de búsqueda

```
public int findEventsGetNumber(String keywords, Long categoryId, boolean admin)
```

**PR-UN-040** findEventsGetNumber**Motivación**

- Comprobar el método devuelve 0 cuando no hay eventos coincidentes con los parámetros de búsqueda



**Parámetros**

- keyWords: null
- categoryId: null
- admin: true

**Valor retorno**

- 0

**Inicialización**

- 

**PR-UN-041** findEventsGetNumber**Motivación**

- Comprobar que se devuelve el número adecuado cuando hay eventos coincidentes con la búsqueda

**Parámetros**

- keyWords: deportivo
- categoryId: null
- admin: true

**Valor retorno**

- 0

**Inicialización**

- 

**CU-013** findEvents

Método que realiza una búsqueda de eventos según distintos parámetros. Los resultados se devuelven ordenados por fecha.

Con los parámetros startIndex y count se indica el número de apuesta a partir de la que se devuelven resultados y el número máximo de los mismos.

Con el parámetro admin se indica si el usuario que invoca el método es el administrador de la aplicación o no.

**Parametros**

- keyWords: palabras clave que tendrán que estar contenidas en el nombre de los eventos existentes
- categoryId: identificador de la categoría a la que pertenecen los eventos existentes
- startIndex posición a partir de la cual se empiezan a recuperar los eventos buscados
- count número máximo de eventos coincidentes con la búsqueda que se devuelve
- admin: booleano que indica si la búsqueda se realiza para el usuario administrador o para un usuario normal. Si es admin se buscarán todos los eventos. En caso contrario sólo aquellos cuya fecha de inicio no haya pasado

## Excepciones

- NA

## Valor retorno

- EventBlock objeto que contiene la lista de eventos ordenados y un booleano (existsMoreEvents) indicando si existen más eventos después del último incluido en dicha lista

```
public EventBlock findEvents(String keyWords, Long categoryId, int startIndex, int count, boolean admin)
```

### PR-UN-042 findEvents

#### Motivación

- Comprobar que EventBlock devuelve una lista vacía y false pues no hay eventos coincidentes con la búsqueda (con startIndex al inicio)

#### Parámetros

- keyWords: null
- categoryId: null
- startIndex: 0
- count: 1
- admin: true

#### Valor retorno

- Lista vacía de eventos
- ExistsMoreEvents valor false

#### Inicialización

- Creación de dos categorías
- Creación de tres eventos
- Inicialización de un eventDaoMock que devuelva una lista vacía de eventos

### PR-UN-043 findEvents

#### Motivación

- Comprobar que EventBlock devuelve una lista con un evento y false, no habiendo más eventos coincidentes con la búsqueda (con startIndex al inicio)

#### Parámetros

- keyWords: null
- categoryId: null
- startIndex: 0
- count: 1
- admin: true

#### Valor retorno

- Lista con un evento coincidente con la búsqueda
- ExitsMoreEvents valor false

#### **Inicialización**

- Creación de dos categorías
- Creación de un evento
- Inicialización de un eventDaoMock que devuelva una lista de un evento

### **PR-UN-044** findEvents

#### **Motivación**

- Comprobar que EventBlock devuelve una lista de dos eventos y true, habiendo más eventos coincidentes con la búsqueda (con startIndex al inicio)

#### **Parámetros**

- keyWords: null
- categoryId: null
- startIndex: 0
- count: 1
- admin: true

#### **Valor retorno**

- Lista con dos eventos coincidentes con la búsqueda
- ExitsMoreEvents valor true

#### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos
- Inicialización de un eventDaoMock que devuelva una lista de dos eventos

### **PR-UN-045** findEvents

#### **Motivación**

- Comprobar que EventBlock devuelve una lista de dos eventos y false, no habiendo más eventos coincidentes con la búsqueda (con startIndex al inicio)

#### **Parámetros**

- keyWords: null
- categoryId: null
- startIndex: 1
- count: 2
- admin: true

#### **Valor retorno**

- Lista con dos eventos coincidentes con la búsqueda

- ExitsMoreEvents valor false

#### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos
- Inicialización de un eventDaoMock que devuelva una lista de dos eventos

### **PR-UN-046 findEvents**

#### **Motivación**

- Comprobar que EventBlock devuelve una lista de tres eventos y true, no habiendo más eventos coincidentes con la búsqueda (con startIndex al inicio)

#### **Parámetros**

- keyWords: null
- categoryId: null
- startIndex: 1
- count: 2
- admin: true

#### **Valor retorno**

- Lista con tres eventos coincidentes con la búsqueda
- ExitsMoreEvents valor true

#### **Inicialización**

- Creación de dos categorías
- Creación de tres eventos
- Inicialización de un eventDaoMock que devuelva una lista de tres eventos

### **CU-014 findBets**

Método que realiza una búsqueda de apuestas por id de usuario. Los resultados se devuelven ordenados por fecha.

Con los parámetros startIndex y count se indica el número de apuesta a partir de la que se devuelven resultados y el número máximo de los mismos.

#### **Parametros**

- userId: identificador del usuario propietario de las apuestas buscadas
- startIndex: posición a partir de la cual se empiezan a recuperar las apuestas buscadas
- count: número máximo de apuestas coincidentes con la búsqueda que se devuelve

#### **Excepciones**

- NA

#### **Valor retorno**

- BetBlock: objeto que contiene la lista de apuestas ordenadas y un boolean (exitsMoreBets) indicando si existen más después de la última apuesta incluida en dicha lista

```
public BetBlock findBets(Long userId,int startIndex, int count)
```

#### **PR-UN-047** findBets

##### **Motivación**

- Comprobar que BetBlock devuelve lista vacía y false si no hay apuestas coincidentes con la búsqueda (con startIndex al inicio)

##### **Parámetros**

- userId: identificador de un usuario sin apuestas
- startIndex: 0
- count: 1

##### **Valor retorno**

- Lista vacía de apuestas
- ExitsMoreEvents valor false

##### **Inicialización**

- Configuración de un mock de BetDao para que devuelva una lista vacía al invocar BetDao.findBetsByUserId()

#### **PR-UN-048** findBets

##### **Motivación**

- Comprobar que BetBlock devuelve lista con las apuestas coincidentes y false (con startIndex al inicio)

##### **Parámetros**

- userId: identificador de un usuario con una apuesta
- startIndex: 0
- count: 1

##### **Valor retorno**

- Lista con la apuesta coincidente con la búsqueda
- ExitsMoreEvents valor false

##### **Inicialización**

- Configuración de un mock de BetDao para que devuelva una lista con un elemento al invocar BetDao.findBetsByUserId()

#### **PR-UN-049** findBets

##### **Motivación**

- Comprobar que BetBlock devuelve lista con las apuestas coincidentes y true (con startIndex al inicio)

### **Parámetros**

- `userId`: identificador de un usuario con más de una apuesta
- `startIndex`: 0
- `count`: 1

### **Valor retorno**

- Lista con las apuestas coincidente con la búsqueda
- `ExitsMoreEvents` valor `true`

### **Inicialización**

- onfiguración de un mock de `BetDao` para que devuelva una lista con más de un elemento al invocar `BetDao.findBetsById()`

## **PR-UN-050** `findBets`

### **Motivación**

- Comprobar que `BetBlock` devuelve lista con las apuestas coincidentes y `false` (con `startIndex` en una posición distinta de la primera)

### **Parámetros**

- `userId`: identificador de un usuario con una apuesta
- `startIndex`: 0
- `count`: 2

### **Valor retorno**

- Lista con la apuesta coincidente con la búsqueda
- `ExitsMoreEvents` valor `false`

### **Inicialización**

- Configuración de un mock de `BetDao` para que devuelva una lista con dos elementos al invocar `BetDao.findBetsById()`

## **PR-UN-051** `findBets`

### **Motivación**

- Comprobar que `BetBlock` devuelve lista con las apuestas coincidentes y `true` (con `startIndex` en una posición distinta de la primera)

### **Parámetros**

- `userId`: identificador de un usuario con una apuesta
- `startIndex`: 1
- `count`: 2

### **Valor retorno**

- Lista con la apuesta coincidente con la búsqueda
- `ExitsMoreEvents` valor `true`

### **Inicialización**

- Configuración de un mock de BetDao para que devuelva una lista con tres elementos al invocar BetDao.findBetsByUserId()

#### **CU-015** makeBet

##### **Parametros**

- userId: identificador del usuario que realiza la apuesta
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad que decide el usuario apostar

##### **Excepciones**

- InstanceNotFoundException: salta cuando no se encuentra el usuario o la opción escogida
- OutdatedBetException: salta cuando el estado de la opción ya está modificado (marcada como ganada o perdida)

##### **Valor retorno**

- Apuesta realizada con éxito

`public Bet makeBet(Long userId, Long betOptionId, Float betedMoney) throws InstanceNotFoundException, OutdatedBetException`

#### **PR-UN-052** makeBet

##### **Motivación**

- Obtener apuesta realizada correctamente

##### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad que decide el usuario apostar

##### **Valor retorno**

- Apuesta realizada correctamente

##### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de tres opciones de apuesta válidas
- Inicialización de un userProfileDaoMock que devuelva el usuario creado
- Inicialización de un betOptionDaoMock que devuelva la opción de apuesta creada
- Inicialización de un betDaoMock que devuelva la apuesta creada

#### **PR-UN-053** makeBet

##### **Motivación**

- Obtener excepción InstanceNotFoundException por usuario no existente

##### **Parámetros**

- userId: identificador de un usuario no existente
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad que decide el usuario apostar

##### **Valor retorno**

- InstanceNotFoundException

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de tres opciones de apuesta válidas
- Inicialización de un userProfileDaoMock que devuelva InstanceNotFoundException
- Inicialización de un betOptionDaoMock que devuelva la opción de apuesta creada

#### **PR-UN-054** makeBet

##### **Motivación**

- Obtener excepción InstanceNotFoundException por opción de tipo apuesta no existente

##### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de una opción de apuesta no existente
- betedMoney: cantidad que decide el usuario apostar

##### **Valor retorno**

- InstanceNotFoundException

##### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Inicialización de un userProfileDaoMock que devuelva el usuario creado
- Inicialización de un betOptionDaoMock que devuelva la opción de apuesta creada



## **PR-UN-055** makeBet

### **Motivación**

- Obtener excepción OutdatedBetException por la opción de apuesta ya marcada a ganadora

### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar, marcada como ganadora (true)
- betedMoney: cantidad que decide el usuario apostar

### **Valor retorno**

- OutdatedBetException

### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta válidas
- Creación de una opción de apuesta marcada a ganadora
- Inicialización de un userProfileDaoMock que devuelva el usuario creado
- Inicialización de un betOptionDaoMock que devuelva la opción de apuesta creada

## **PR-UN-056** makeBet

### **Motivación**

- Obtener excepción OutdatedBetException por la opción de apuesta ya marcada a perdida

### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar, marcada como perdida (falsa)
- betedMoney: cantidad que decide el usuario apostar

### **Valor retorno**

- OutdatedBetException

### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta

- Creación de dos opciones de apuesta válidas
- Creación de una opción de apuesta marcada a pérdida
- Inicialización de un userProfileDaoMock que devuelva el usuario creado
- Inicialización de un betOptionDaoMock que devuelva la opción de apuesta creada

#### **PR-UN-057** makeBet

##### **Motivación**

- Hacer que la aplicación rompa con WrongQuantityException si el usuario intenta realizar una apuesta con una cantidad de dinero negativa. Este test y el caso de uso al que aplica han sido modificados, ya que cuando se diseñó se detectó un bug en la aplicación (se creaban apuestas con cantidades menores que cero)

##### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad negativa de dinero

##### **Valor retorno**

- WrongQuantityException

##### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de tres opciones de apuesta válidas
- Inicialización de un userProfileDaoMock que devuelva el usuario creado
- Inicialización de un betOptionDaoMock que devuelva la opción de apuesta creada
- Inicialización de un betDaoMock que devuelva la apuesta creada

#### **CU-016** insertEvent

##### **Parametros**

- event: identificador del usuario que realiza la apuesta
- categoryId: identificador de la opción escogida para apostar

##### **Excepciones**

- AlreadyPastedDateException: salta cuando se intenta crear un evento con fecha pasada
- InstanceNotFoundException: salta cuando no se encuentra la categoría a partir de su identificador

- DuplicateEventNameException: salta cuando se intenta insertar un evento con un nombre que ya existe

#### **Valor retorno**

- Evento creado con éxito

```
public Event insertEvent(Event event, Long categoryId) throws
AlreadyPastedDateException, InstanceNotFoundException,
DuplicateEventNameException
```

#### **PR-UN-058 insertEvent**

##### **Motivación**

- Obtener evento creado correctamente

##### **Parámetros**

- event: evento inicializado
- categoryId: identificador de categoría existente

##### **Valor retorno**

- Evento creado correctamente

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Inicialización de un categoryDaoMock que devuelva la categoría creada
- Inicialización de un eventDaoMock que devuelva el evento creado

#### **PR-UN-059 insertEvent**

##### **Motivación**

- Obtener excepción AlreadyPastedDateException por intentar crear un evento con fecha pasada

##### **Parámetros**

- event: evento inicializado con fecha pasada
- categoryId: identificador de categoría existente

##### **Valor retorno**

- AlreadyPastedDateException

##### **Inicialización**

- Creación de una categoría
- Creación de un evento con fecha pasada
- Inicialización de un categoryDaoMock que devuelva la categoría creada

#### **PR-UN-060 insertEvent**

##### **Motivación**

- Obtener excepción DuplicateEventNameException por intentar crear un evento con el mismo nombre

#### **Parámetros**

- event: evento inicializado con nombre ya existente
- categoryId: identificador de categoría existente

#### **Valor retorno**

- DuplicateEventNameException

#### **Inicialización**

- Creación de una categoría
- Creación de un evento con nombre ya existente
- Inicialización de un categoryDaoMock que devuelva la categoría creada
- Inicialización de un eventDaoMock que devuelva con valor true si existe algún evento con el mismo nombre

### **PR-UN-061 insertEvent**

#### **Motivación**

- Obtener excepción InstanceNotFoundException por obtener una apuesta no existente

#### **Parámetros**

- event: evento inicializado
- categoryId: identificador de categoría no existente

#### **Valor retorno**

- Evento creado correctamente

#### **Inicialización**

- Creación de un evento válido
- Inicialización de un categoryDaoMock que devuelva InstanceNotFoundException

### **CU-017 insertBetType**

#### **Parametros**

- betType: tipo de apuesta inicializada

#### **Excepciones**

- DuplicateBetTypeQuestionException: salta cuando se intenta insertar un tipo de apuesta con una pregunta ya existente
- DuplicateBetOptionAnswerException: salta cuando se intenta insertar una opción de apuesta con una respuesta ya existente
- MinimumBetOptionException: salta cuando se intenta insertar un tipo de apuesta con menos de dos tipos de apuesta

#### **Valor retorno**

- Tipo de apuesta insertada con éxito

`public BetType insertBetType(BetType betType) throws`

`DuplicateBetTypeQuestionException, DuplicateBetOptionAnswerException, MinimunBetOptionException`

#### **PR-UN-062** insertBetType

##### **Motivación**

- Obtener un tipo de apuesta insertada correctamente

##### **Parámetros**

- betType: tipo de apuesta inicializada

##### **Valor retorno**

- Tipo de apuesta insertada correctamente

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de tres opciones de apuesta válidas
- Inicialización de un categoryDaoMock que devuelva la categoría creada
- Inicialización de un betTypeDaoMock que devuelva el tipo de apuesta creado

#### **PR-UN-063** insertBetType

##### **Motivación**

- Obtener excepción MinimunBetOptionException por intentar insertar un tipo de apuesta sin ninguna opción de apuesta

##### **Parámetros**

- betType: tipo de apuesta inicializada

##### **Valor retorno**

- MinimunBetOptionException

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Inicialización de un categoryDaoMock que devuelva la categoría creada

#### **PR-UN-064** insertBetType

##### **Motivación**

- Obtener excepción `MinimumBetOptionException` por intentar insertar un tipo de apuesta sin ninguna opción de apuesta (lista de apuesta nula)

#### **Parámetros**

- `betType`: tipo de apuesta inicializada

#### **Valor retorno**

- `MinimumBetOptionException`

#### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Inicialización de un `categoryDaoMock` que devuelva la categoría creada

### **PR-UN-065** `insertBetType`

#### **Motivación**

- Obtener excepción `MinimumBetOptionException` por intentar insertar un tipo de apuesta con una opción de apuesta

#### **Parámetros**

- `betType`: tipo de apuesta inicializada

#### **Valor retorno**

- `MinimumBetOptionException`

#### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de un tipo de apuesta
- Inicialización de un `categoryDaoMock` que devuelva la categoría creada

### **PR-UN-066** `insertBetType`

#### **Motivación**

- Obtener excepción `DuplicateBetTypeQuestionException` por intentar insertar un tipo de apuesta con un nombre ya existente

#### **Parámetros**

- `betType`: tipo de apuesta inicializada con nombre ya existente

#### **Valor retorno**

- `DuplicateBetTypeQuestionException`

#### **Inicialización**

- Creación de una categoría

- Creación de un evento
- Creación de un tipo de apuesta
- Creación de tres opciones de apuesta válidas
- Inicialización de un categoryDaoMock que devuelva la categoría creada
- Inicialización de un eventDaoMock que devuelva el evento creado
- Inicialización de un betTypeDaoMock que devuelva el tipo de apuesta creado

#### **CU-018** findEvent

##### **Parametros**

- eventId: identificador de un evento

##### **Excepciones**

- InstanceNotFoundException: indica que no existe ningún evento con el identificador dado

##### **Valor retorno**

- Evento existente

`public Event findEvent (Long Event) throws InstanceNotFoundException`

#### **PR-UN-067** findEvent

##### **Motivación**

- Obtener un evento a partir de su identificador

##### **Parámetros**

- eventId: identificador de un evento existente

##### **Valor retorno**

- Evento existente

##### **Inicialización**

- Creación de un evento
- Inicialización de un eventDaoMock que devuelva un evento

#### **PR-UN-068** findEvent

##### **Motivación**

- excepción InstanceNotFoundException por obtener un evento no existente

##### **Parámetros**

- eventId: identificador de un evento no existente

##### **Valor retorno**

- InstanceNotFoundException

### **Inicialización**

- Inicialización de un eventDaoMock que devuelva InstanceNotFoundException

## **CU-019** findBetType

### **Parametros**

- betTypeId: identificador de un tipo de apuesta

### **Excepciones**

- InstanceNotFoundException: salta cuando no se encuentra un tipo apuesta a partir de su identificador

### **Valor retorno**

- Tipo de apuesta existente

`public BetType findBetType (Long betTypeId) throws InstanceNotFoundException`

## **PR-UN-069** findBetType

### **Motivación**

- Obtener el tipo de apuesta existente

### **Parámetros**

- betTypeId: identificador de un tipo apuesta existente

### **Valor retorno**

- Tipo de apuesta existente

### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Inicialización de un betTypeDaoMock que devuelva el tipo apuesta creado

## **PR-UN-070** findBetType

### **Motivación**

- excepción InstanceNotFoundException por obtener un tipo apuesta no existente

### **Parámetros**

- betTypeId: identificador de una opción de tipo apuesta no existente

### **Valor retorno**

- InstanceNotFoundException

### **Inicialización**

- Creación de una categoría



- Creación de un evento
- Inicialización de un betTypeDaoMock que devuelva InstanceNotFoundException

#### **CU-020** findBetOption

##### **Parametros**

- betOptionId: identificador de una opción de un tipo de apuesta

##### **Excepciones**

- InstanceNotFoundException: salta cuando no se encuentra la opción de tipo apuesta a partir de su identificador

##### **Valor retorno**

- Opción de tipo de apuesta existente

`public BetOption findBetOption (Long betOptionId) throws InstanceNotFoundException`

#### **PR-UN-071** findBetOption

##### **Motivación**

- Obtener la opción de tipo de apuesta existente

##### **Parámetros**

- betOptionId: identificador de una opción de tipo apuesta existente

##### **Valor retorno**

- Opción de tipo de apuesta existente

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de una opción de tipo apuesta
- Inicialización de un betOptionDaoMock que devuelva la opción de tipo apuesta creada

#### **PR-UN-072** findBetOption

##### **Motivación**

- excepción InstanceNotFoundException por obtener una opción de tipo apuesta no existente

##### **Parámetros**

- betOptionId: identificador de una opción de tipo apuesta no existente

##### **Valor retorno**

- InstanceNotFoundException

### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Inicialización de un betOptionDaoMock que devuelva InstanceNotFoundException

### **CU-021** checkOptions

Caso de uso que recibe el id de un tipo de apuesta y una lista de ids de opciones de apuesta.

Las opciones de ese tipo de apuesta cuyo id está contenido en la lista se marcan como ganadoras (betState=true)

### **Parametros**

- betTypeId: identificador de un tipo de apuesta
- wonned: lista de ids de opciones de apuesta

### **Excepciones**

- InstanceNotFoundException: salta cuando no existe el tipo de apuesta indicado
- OnlyOneWonOptionException: salta cuando el tipo de apuesta es simple (sólo puede existir una opción ganadora) y la lista de ids de opciones tiene más de un elemento.
- NotAllOptionsExistsException: salta cuando alguno de los ids de las opciones de apuesta no pertenece al tipo de apuesta indicado.

### **Valor retorno**

- void

```
public void checkOptions (Long betTypeId, Set<Long>wonned) throws  
InstanceNotFoundException, OnlyOneWonOptionException,
```

```
NotAllOptionsExistsException
```

### **PR-UN-073** checkOptions

### **Motivación**

- Marcar como ganadora una opción de un tipo de apuesta simple

### **Parámetros**

- betTypeId: identificador de un tipo de apuesta simple
- wonned: lista con el id de una opción de apuesta perteneciente al tipo de apuesta indicado

### **Valor retorno**

- void

### **Inicialización**

- Creación de un tipo de apuesta
- Creación de dos opción de tipo apuesta
- Creación de un betTypeDaoMock que devuelva un tipo de apuesta simple con dos opciones al buscar por id.

#### **PR-UN-074** checkOptions

##### **Motivación**

- Marcar como ganadoras varias opciones de apuesta de un tipo de apuesta compuesto

##### **Parámetros**

- identificador de un tipo de apuesta compuesto
- wonned: lista con dos ids de opciones de apuesta

##### **Valor retorno**

- void

##### **Inicialización**

- Creación de un tipo de apuesta
- Creación de tres opción de tipo apuesta
- Creación de un betTypeDaoMock que devuelva un tipo de apuesta compuesto

#### **PR-UN-075** checkOptions

##### **Motivación**

- La aplicación rompe con NotAllOptionsExistsException al pasar un id de opción que no pertenece al tipo de apuesta dado

##### **Parámetros**

- identificador de un tipo de apuesta simple
- lista con un id de opción de apuesta que no pertenece al tipo de apuesta dado

##### **Valor retorno**

- NotAllOptionsExistsException

##### **Inicialización**

- Creación de un tipo de apuesta
- Creación de tres opción de tipo apuesta
- Creación de un betTypeDaoMock que devuelva un tipo de apuesta simple

#### **PR-UN-076** checkOptions

##### **Motivación**

- La aplicación rompe con OnlyOneWonOptionException al pasar más de un id de opción ganadora para un tipo de apuesta simple

**Parámetros**

- identificador de un tipo de apuesta simple
- wonned: lista con dos ids de opciones

**Valor retorno**

- OnlyOneWonOptionException

**Inicialización**

- Creación de un tipo de apuesta
- Creación de tres opción de tipo apuesta
- Creación de un betTypeDaoMock que devuelva un tipo de apuesta simple

**PR-UN-077** checkOptions**Motivación**

- La aplicación rompe con InstanceNotFoundException al pasar un id de un tipo de apuesta no existente

**Parámetros**

- identificador de un tipo de apuesta no existente
- wonned: -

**Valor retorno**

- InstanceNotFoundException

**Inicialización**

- Creación de un betTypeDaoMock que devuelva InstanceNotFoundException

**CU-022** findCategory**Parametros**

- categoryId: identificador de una categoría

**Excepciones**

- InstanceNotFoundException: salta cuando el método recibe un identificador de categoría que no existe

**Valor retorno**

- Categoría con el identificador dado

```
public Category findCategory(Long categoryId) throws InstanceNotFoundException
```

**PR-UN-078** findCategory**Motivación**

- Obtener una categoría existente

**Parámetros**

- categoryId: identificador de una categoría existente

**Valor retorno**

- Categoría con el id dado

**Inicialización**

- Creación de una categoría
- Inicialización de un categoryDaoMock que devuelva una categoría

**PR-UN-079** findCategory**Motivación**

- La aplicación rompe con InstanceNotFoundException

**Parámetros**

- categoryId: identificador de una categoría que no existe

**Valor retorno**

- InstanceNotFoundException

**Inicialización**

- Inicialización de un categoryDaoMock que devuelva InstanceNotFoundException

**CU-023** findCategories**Parametros**

- NA

**Excepciones**

- NA

**Valor retorno**

- Lista con las categorías existentes

```
public List<Category>findCategories()
```

**PR-UN-080** findCategories**Motivación**

- Obtener una lista de las categorías existentes

**Parámetros**

- NA

**Valor retorno**

- Lista de dos categorías

**Inicialización**

- Creación de dos categorías

- Inicialización de un categoryDaoMock que devuelva la lista de categorías creadas

#### **PR-UN-081** findCategories

##### **Motivación**

- Obtener una lista vacía

##### **Parámetros**

- NA

##### **Valor retorno**

- Lista vacía

##### **Inicialización**

- Inicialización de un categoryDaoMock que devuelva la lista de categorías creadas (lista vacía)

#### **CU-024** findDuplicates

##### **Parametros**

- eventId: identificador de un evento existente
- fullName: nombre de la pregunta del tipo de apuesta

##### **Excepciones**

- NA

##### **Valor retorno**

- Valor booleano que indica si existe otro tipo de apuesta con el mismo nombre

`public boolean findDuplicates(Long eventId, String fullName)`

#### **PR-UN-082** findDuplicates

##### **Motivación**

- Obtener valor true

##### **Parámetros**

- NA

##### **Valor retorno**

- true

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Inicialización de un eventDaoMock que devuelva el evento creado

- Inicialización de un betTypeDaoMock que devuelva el valor true conforme existe otro tipo de apuesta con la misma pregunta

#### **PR-UN-083** findDuplicates

##### **Motivación**

- Obtener valor false

##### **Parámetros**

- eventId: identificador de un evento creado
- fullName: nombre de un tipo de apuesta no existente

##### **Valor retorno**

- false

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Inicialización de un eventDaoMock que devuelva el evento creado
- Inicialización de un betTypeDaoMock que devuelva el valor false conforme no existe otro tipo de apuesta con la misma pregunta

#### **CU-025** findBetByUserIdNumber

##### **Parametros**

- userId: identificador de usuario

##### **Excepciones**

- NA

##### **Valor retorno**

- Número de apuestas del usuario dado

```
public int findBetsByUserIdNumber(Long userId)
```

#### **PR-UN-084** findBetByUserIdNumber

##### **Motivación**

- Comprobar el método devuelve 0 cuando el usuario no ha realizado apuestas

##### **Parámetros**

- userId: identificador de usuario sin apuestas

##### **Valor retorno**

- 0

##### **Inicialización**

- Creación de un usuario
- Inicialización de un betDaoMock que devuelva 0 al invocar findBetsByUserIdNumber

#### **PR-UN-085** findDuplicates

##### **Motivación**

- Comprobar el método devuelve el número de apuestas del usuario

##### **Parámetros**

- userId: identificador de usuario con una apuesta

##### **Valor retorno**

- 1

##### **Inicialización**

- Creación de un usuario
- Inicialización de un betDaoMock que devuelva 1 al invocar findBetsByUserIdNumber



### 3.2. Test de integración

Por falta de tempo abstivémonos de testear nesta fase de probas algúns casos de uso moi sinxelos, que se limitan a pasar a chamada aos métodos do módulo inferior (probados xa na fase de unidade), sen engadir lóxica de negocio.

Implementamos tres tipos de métodos de test:

1. Métodos que representan pequenos escenarios habituais no contexto da aplicación, e que inclúen varios casos de uso.
2. Métodos sinxelos que proban un único caso de uso.
3. Un método con selección de datos aleatoria

### 3.2.1. Escenarios

**PR-IN-001:** Escenario de creación de un evento y un tipo de apuesta con dos opciones asociadas, siendo administrador.

#### Descripción

- El administrador crea un evento, y posteriormente lo busca para comprobar si se ha añadido con éxito. A continuación crea un tipo de apuesta con dos opciones de apuesta.

#### Casos de uso implicados

- insertEvent
- findEvent
- insertBetType

#### Inicialización

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta

**PR-IN-002:** Escenario de marcar como ganadora una de las opciones de apuesta asociadas a un tipo de apuesta de un evento.

#### Descripción

- El administrador crea un evento, y posteriormente lo busca para comprobar si se ha añadido con éxito. A continuación crea dos opciones de apuesta para un tipo de apuesta nuevo. Por último, establece como ganadora una de las opciones de apuesta, quedando la restante como perdida. Tras esta operación, el administrador consulta el estado de las opciones para comprobar que se ha actualizado el estado.

#### Casos de uso implicados

- insertEvent
- findEvent
- insertBetType
- checkWinners

#### Inicialización

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta

**PR-IN-003:** Escenario de realización de una apuesta

#### Descripción

- Un usuario realiza una búsqueda de eventos, elige uno y realiza una apuesta sobre una de sus opciones de apuesta. A continuación realiza una búsqueda de sus apuestas para comprobar que existe la que acaba de realizar.

#### **Casos de uso implicados**

- findEvents
- makeBet
- findBets

#### **Inicialización**

- Creación de un usuario
- Creación de dos categorías
- Creación de tres eventos
- Creación de un tipo de apuesta con dos opciones (que se asigna al event1)

### **PR-IN-004: Escenario de comprobación del estado de una apuesta**

#### **Descripción**

- Un usuario realiza una apuesta. El administrador establece las opciones ganadoras para ese tipo de apuesta. El usuario busca su apuesta realizada y comprueba el estado que la opción por la que había apostado es ganadora.

#### **Casos de uso implicados**

- makeBet
- checkOptions
- findBets

#### **Inicialización**

- Creación de un usuario
- Creación de dos categorías
- Creación de tres eventos
- Creación de un tipo de apuesta con dos opciones (que se asigna al event1)

### **3.2.2. Casos de prueba simples**

#### **PR-IN-005: makeBet**

#### **Motivación**

- Obtener apuesta realizada correctamente

#### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad negativa de dinero

#### **Valor retorno**

- WrongQuantityException

#### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de tres opciones de apuesta válidas

#### **PR-IN-006: makeBet**

##### **Motivación**

- Obtener excepción InstanceNotFoundException por usuario no existente

##### **Parámetros**

- userId: identificador de un usuario no existente
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad negativa de dinero

##### **Valor retorno**

- InstanceNotFoundException

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de tres opciones de apuesta válidas

#### **PR-IN-007: makeBet**

##### **Motivación**

- Obtener excepción InstanceNotFoundException por tipo de apuesta no existente

##### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de una opción de apuesta no existente
- betedMoney: cantidad de dinero a apostar

##### **Valor retorno**

- InstanceNotFoundException

##### **Inicialización**

- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta

## **PR-IN-008: makeBet**

### **Motivación**

- Obtener excepción OutdatedBetException por la opción de apuesta ya marcada a ganadora

### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar, marcada como ganadora (true)
- betedMoney: cantidad de dinero a apostar

### **Valor retorno**

- OutdatedBetException

### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta válidas
- Creación de una opción de apuesta marcada a ganadora

## **PR-IN-009: makeBet**

### **Motivación**

- Obtener excepción OutdatedBetException por la opción de apuesta ya marcada a perdida

### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar, marcada como perdida (false)
- betedMoney: cantidad de dinero a apostar

### **Valor retorno**

- OutdatedBetException

### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta válidas
- Creación de una opción de apuesta marcada a perdida

#### **PR-IN-010: makeBet**

##### **Motivación**

- Ver como se comporta la aplicación al realizar una apuesta con una cantidad negativa de dinero

##### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad negativa de dinero

##### **Valor retorno**

- Apuesta realizada con cantidad negativa de dinero (ERROR)

##### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta válidas
- Creación de una opción de apuesta válidas

#### **PR-IN-011: insertEvent**

##### **Motivación**

- Obtener evento creado correctamente

##### **Parámetros**

- event: evento inicializado
- categoryId: identificador de categoría existente

##### **Valor retorno**

- Evento creado correctamente

##### **Inicialización**

- Creación de una categoría

#### **PR-IN-012: insertEvent**

##### **Motivación**

- Obtener excepción AlreadyPastedDateException por intentar crear un evento con fecha pasada

##### **Parámetros**

- event: evento inicializado con fecha pasada
- categoryId: identificador de categoría existente

**Valor retorno**

- AlreadyPastedDateException

**Inicialización**

- Creación de una categoría

**PR-IN-013: insertEvent****Motivación**

- Obtener excepción excepción DuplicateEventNameException por intentar crear un evento con el mismo nombre

**Parámetros**

- event: evento inicializado con el mismo nombre que un evento existente
- categoryId: identificador de categoría existente

**Valor retorno**

- DuplicateEventNameException

**Inicialización**

- Creación de una categoría
- Creación de un evento

**PR-IN-014: insertEvent****Motivación**

- Obtener excepción excepción InstanceNotFoundException por intentar crear una categoría no existente

**Parámetros**

- event: evento inicializado
- categoryId: identificador de categoría no existente

**Valor retorno**

- InstanceNotFoundException

**Inicialización**

- -

**PR-IN-015: insertBetType****Motivación**

- Insertar un tipo de apuesta correctamente

**Parámetros**

- betType: tipo de apuesta inicializada con dos opciones de apuesta

**Valor retorno**

- Tipo de apuesta creado correctamente

#### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta válidas

#### **PR-IN-016: insertBetType**

##### **Motivación**

- Obtener excepción `MinimumBetOptionException` por intentar insertar un tipo de apuesta sin ninguna opción de apuesta

##### **Parámetros**

- `betType`: tipo de apuesta inicializado sin opciones de apuesta

##### **Valor retorno**

- `MinimumBetOptionException`

##### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta

#### **PR-IN-017: insertBetType**

##### **Motivación**

- Obtener excepción `MinimumBetOptionException` por intentar insertar un tipo de apuesta con una opción de apuesta

##### **Parámetros**

- `betType`: tipo de apuesta inicializado con una opción de apuesta

##### **Valor retorno**

- `MinimumBetOptionException`

##### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta
- Creación de una opción de apuesta válida

#### **PR-IN-018: insertBetType**

##### **Motivación**



- Obtener excepción DuplicateBetOptionAnswerException por intentar insertar un tipo de apuesta con dos opciones iguales

#### **Parámetros**

- betType: tipo de apuesta inicializada con dos opciones de apuesta iguales

#### **Valor retorno**

- DuplicateBetOptionAnswerException

#### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta iguales

### **PR-IN-019: insertBetType**

#### **Motivación**

- Obtener excepción DuplicateBetTypeQuestionException por intentar insertar un tipo de apuesta con un nombre ya existente

#### **Parámetros**

- betType: tipo de apuesta inicializada con un nombre ya existente en otro tipo de apuesta del mismo evento

#### **Valor retorno**

- DuplicateBetTypeQuestionException

#### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta en bd
- Inicialización de un tipo de apuesta con el mismo nombre que el almacenado en bd

### **PR-IN-020: checkOptions**

#### **Motivación**

- Marcar como ganadora una opción de un tipo de apuesta simple

#### **Parámetros**

- betTypeId: identificador de un tipo de apuesta simple
- winned: lista con el id de una opción de apuesta perteneciente al tipo de apuesta indicado.

#### **Valor retorno**

- Void

### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta simple en bd con dos opciones de apuesta

#### **PR-IN-021: checkOptions**

### **Motivación**

- Marcar como ganadoras varias opciones de apuesta de un tipo de apuesta compuesto

### **Parámetros**

- betTypeId: identificador de un tipo de apuesta compuesto
- wonned: lista con dos ids de opciones de apuesta

### **Valor retorno**

- Void

### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta múltiple con tres opciones en bd.

#### **PR-IN-022: checkOptions**

### **Motivación**

- La aplicación rompe con NotAllOptionsExistsException al pasar un id de opción que no pertenece al tipo de apuesta dado

### **Parámetros**

- betTypeId: identificador de un tipo de apuesta simple
- wonned: lista con un id de opción de apuesta que no pertenece al tipo de apuesta dado

### **Valor retorno**

- NotAllOptionsExistsException

### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta simple con dos opciones en bd.

#### **PR-IN-023: checkOptions**

### **Motivación**

- La aplicación rompe con OnlyOneWonOptionException al pasar más de un id de opción ganadora para un tipo de apuesta simple

### **Parámetros**

- betTypeId: identificador de un tipo de apuesta simple
- wonned: lista con dos ids de opciones de apuesta

### **Valor retorno**

- OnlyOneWonOptionException

### **Inicialización**

- Creación de una categoría en bd
- Creación de un evento en bd
- Creación de un tipo de apuesta simple con dos opciones en bd.

## **PR-IN-024: checkOptions**

### **Motivación**

- La aplicación rompe con InstanceNotFoundException al pasar un id de un tipo de apuesta no existente

### **Parámetros**

- betTypeId: identificador de un tipo de apuesta no existente
- wonned: -

### **Valor retorno**

- InstanceNotFoundException

### **Inicialización**

- -

## **3.2.3. Selección de datos aleatoria**

## **PR-IN-024: makeBet**

### **Motivación**

- Utilizamos la herramienta de generación aleatoria QuickCheck para realizar apuestas con una cantidad aleatoria, comprendidas entre 0 y +100000.

### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad aleatoria generada con QuickCheck

### **Valor retorno**

- Apuesta realizada correctamente

### **Inicialización**

- Creación de un usuario
- Creación de una categoría

- Creación de un evento
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta válidas

#### **PR-IN-025: makeBet**

##### **Motivación**

- Utilizamos la herramienta de generación aleatoria QuickCheck para realizar apuestas con una cantidad negativa comprendida entre -100000 y 0.

##### **Parámetros**

- userId: identificador de un usuario existente
- betOptionId: identificador de la opción escogida para apostar
- betedMoney: cantidad aleatoria negativa, generada con QuickCheck

##### **Valor retorno**

- WrongQuantityException

##### **Inicialización**

- Creación de un usuario
- Creación de una categoría
- Creación de un evento
- Creación de un tipo de apuesta
- Creación de dos opciones de apuesta válidas

## 4. Rexisto de probas

Para a correcta e completa validación e verificación da nosa aplicación empregamos diferentes ferramentas de:

1. Implementación de test unitarios e de integración: as ferramentas empregadas foron **JUnit** e **Mocking**.
2. Análise estática de caixa branca: a ferramenta empregada foi **CheckStyle**.
3. Mutación de código: a ferramenta empregada foi **PITest**.
4. Estrés: a ferramenta empregada foi **VisualVM**.
5. Pruebas basadas en modelos: a ferramenta empregada foi **Graphwalker**.
6. Selección de datos aleatoria: a ferramenta empregada foi **QuickCheck**.
7. Cobertura: a ferramenta empregada foi **Cobertura**.

## 5. Rexistro de erros

Tras a execución do plan de probas e das ferramentas mencionadas anteriormente, foron aparecendo erros no noso código. Dividiremos esta sección segundo os resultados obtidos cas diferentes ferramentas:


### 5.1. JUnit e Mocking

Tras o deseño e a execución das probas, detectamos un erro no comportamento dun dos nosos casos de uso, `makebet` do servizo `BetServiceImpl`. Este método implementa o comportamento de realizar unha aposta, polo que un dos seus parámetros é a cantidade a apostar. Na nosa aplicación realizamos unha comprobación de que esta cantidade sexa positiva na capa vista, coas validacións propias de *Tapestry*, polo que no código do servizo non está incluído. É dicir, o servizo permite realizar apostas con cantidades negativas.

Tras detectar este bug, creamos unha tarefa para a súa resolución. O problema foi resolto mediante a comprobación no código de que a cantidade a apostar fose estritamente maior que 0. Ademáis, creouse unha nova excepción que é lanzada cando se intenta crear unha aposta cunha cantidade errónea: `WrongQuantityException`.

### 5.2. CheckStyle

O primeiro análise de CheckStyle rematou cos seguintes resultados:

Checkstyle Results			
The following document contains the results of Checkstyle  <a href="#">XML</a>			
Summary			
Files	Infos 	Warnings 	Errors 
71	0	0	9908

Para mellorar dito análise e dotar de calidade ó noso código, fomos abrendo tarefas e resolvendo os erros que estiveran relacionados coa mesma categoría:

**Spaces and Tabs** Recollimos nesta tarefa os erros do relativo ós espazos e tabulacións.

Regra	Número de erros
FileTabCharacter	4626 erros
RegexpSingleline	622 erros
NoWhitespaceAfter	4 erros
NoWhitespaceBefore	12 erros
WhitespaceAfter	86 erros
WhitespaceAround	746 erros
NewlineAtEndOfFile	71 erros

Tras a corrección, séguese mostrando erros do tipo `NewlineAtEndOfFile` a pesar de seguir as indicacións da ferramenta.

**JavaDoc** Recollimos nesta tarefa os erros do relativo ó Javadoc.

Regra	Número de erros
JavadocPackage	18 erros
JavadocMethod	618 erros
JavadocType	132 erros
JavadocVariable	530 erros
JavadocStyle	8 erros

**LineLength** Recollimos nesta tarefa os erros do relativo ó número de lonxitude de liñas do noso código.

Regra	Número de erros
LineLength	462 erros

Tras a corrección, permitimos 16 erros deste tipo, xa que non son da implementación do noso código, senón que están rexistrados nos `import` dalgunhas clases.

**DesignForExtension** Recollimos nesta tarefa os erros do relativo ás anotacións.

Regra	Número de erros
DesignForExtension	243 erros

Debido á gran cantidade de anotacións que temos no noso código polo uso de Hibernate e Spring, ademáis das propias de Java como `Override` e `Test`, decidimos eliminar a regra **DesignForExtension** da nosa análise da aplicación.

**Modifiers** Recollimos nesta tarefa os erros do relativo á declaración de clases e parámetros.

Regra	Número de erros
ModifierOrder	10 erros
RedundantModifier	34 erros
FinalClass	2 erros
FinalParameters	285 erros
HiddenField	140 erros
HideUtilityClassConstructor	2 erros

Tras a corrección dos erros relativos a esta tarefa, decidimos permitir a violación **FinalParameters** para 9 casos, pois para o correcto funcionamento do código é necesario que eses parámetros non sexan final.

Ademáis, os erros relativos á regra **HiddenField** decidimos non resolvelos pois cremos que supondría moitos cambios no noso código e non consideramos que proporcionarían calidade suficiente á nosa aplicación.

**Names** Recollimos nesta tarefa os erros do relativo ás regras de nomeado.

Regra	Número de erros
ConstantName	5 erros
LocalVariableName	2 erros
MemberName	1 erros
MethodName	4 erros
ParameterName	10 erros
TypeName	1 erros

Decidimos non resolver os erros relativos a estas regras que pertencen á clase **Jcrypt**, localizada en `es/udc/pa/pa001/apuestas/model/userservice/util/jcrypt.java`, pois non foi implementada por nós.

Polo tanto, o número de erros proporcionados por estas regras segue sendo o mesmo, exceptuando **ParameterName** que reduciu o seu número a 7 erros.

**Brackets and Braces** Recollimos nesta tarefa os erros do relativo ó uso de paréntesis e corchetes.

Regra	Número de erros
NeedBraces	21 erros
ArrayTypeStyle	20 erros

**MagicNumber** Recollimos nesta tarefa os erros do relativo ó uso de números sen ser empregados como variables.

Regra	Número de erros
MagicNumber	128 erros

Decidimos non resolver os erros relativos a esta regra pois cremos que supondría moitos cambios no noso código e non consideramos que proporcionarían calidade suficiente á nosa aplicación.

**Conditionals** Recollimos nesta tarefa os erros do relativo á maneira de realizar as sentencias condicionales.

Regra	Número de erros
AvoidInlineConditionals	10 erros
SimplifyBooleanExpression	6 erros

### 5.3. PITest

Tras a mutación do noso código coa ferramenta PITest, observamos que o obxecto `BetTypeBlock` non estaba sendo utilizado, polo que procedimos a eliminar dita clase da nosa aplicación.

Na entidade `Event` observamos como o método `finishedEvent` non estaba sendo probado, polo que engadimos un `assertEquals` nun dos métodos de test para comprobar se a propiedade `eventStart` é posterior ó momento actual.

```
public final boolean finishedEvent(final Long eventId) {  
    return eventStart.getTime().before(Calendar.getInstance().getTime());  
}  
}
```

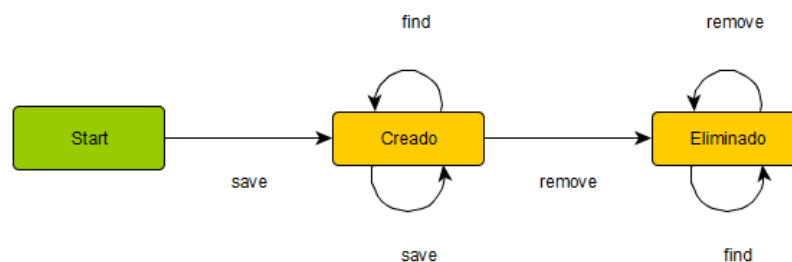
Na creación de eventos e apostas, un dos parámetros que se almacenan é a data de inicio dos mesmos. Debido a que os únicos datos da data que queremos gardar son o día, o mes, o ano, as horas e os minutos, inicializamos os segundos e os milisegundos a 0 nas chamadas ós constructores de `Event` e `Bet`. Por eso mesmo, os mutantes asociados a estes constructores sobreviven.

<pre>public Event(final String name, final Calendar eventStartDate,     final Category cat) {     super();     this.eventName = name;     this.eventStart = eventStartDate;     if (this.eventStart != null) {         this.eventStart.set(Calendar.SECOND, 0);         this.eventStart.set(Calendar.MILLISECOND, 0);     }     this.category = cat;     this.betTypes = new LinkedList&lt;BetType&gt;(); }</pre>	<pre>public Bet(final Float betedMoney, final UserProfile userProfile,     final Event eventId,     final BetOption betOption) {     super();     this.betedMoney = betedMoney;     this.date = Calendar.getInstance();     this.date.set(Calendar.MILLISECOND, 0);     this.userProfile = userProfile;     this.event = eventId;     this.betOption = betOption; }</pre>
---	---

### 5.4. GraphWalker

Con esta ferramenta queremos comprobar o correcto funcionamento das operacións básicas CRUD do compoñente DAO do que heredan as nosas implementacións.

Para isto, primeiro fixemos un grafo que representa os estados dos obxectos a través destas operacións de acordo coas regras da ferramenta.





Comprobamos a validez do noso grafo a través do comando: `java -jar graphwalker.jar offline -m src/main/resources/es/udc/pa/pa001/apuestas/testautomation random(edge_coverage(100))"`



Tras a reestructuración do noso proxecto, engadindo as carpetas correspondentes para o funcionamento da ferramenta, executamos `mvn graphwalker:generate-sources` para obter a interfaz a partir da cal engadir os nosos test. Realizamos as probas correspondentes, pero atopámonos co erro de que esa clase de test non é capaz de inicializar o DAO correspondente para a realización das operacións dos tests. Polo tanto, obtemos un resultado fallido tras a súa execución.

Sen embargo, para demostrar o funcionamento do grafo, adxuntamos unha execución do mesmo e como sería o curso do test a implementar:

```

{"currentElementName":"save"}
{"currentElementName":"Creado"}
{"currentElementName":"find"}
{"currentElementName":"Creado"}
{}
{"currentElementName":"Creado"}
{"currentElementName":"save"}
{"currentElementName":"Creado"}
{"currentElementName":"save"}
{"currentElementName":"Creado"}
{"currentElementName":"find"}
{"currentElementName":"Creado"}
{"currentElementName":"save"}
{"currentElementName":"Creado"}
{}
{"currentElementName":"Creado"}
{}
{"currentElementName":"Creado"}
{"currentElementName":"find"}
{"currentElementName":"Creado"}
{"currentElementName":"remove"}
{"currentElementName":"Eliminado"}
{"currentElementName":"remove"}
{"currentElementName":"Eliminado"}

```

Debido a que basámonos neste grafo para a implementación dos tests de unidade, non imos implementar dita execución.

## 5.5. Cobertura

Tras a análise de cobertura, decidimos ampliar as probas realizadas para a clase `BetServiceImpl` no caso de uso `insertBetType`.

Ademáis, observamos que os constructores co identificador nunca son usados, xa que

para crear novas instancias dos obxectos, úsanse sen eles. Por eso mesmo, eliminamos ditos constructores.

Decidimos realizar un novo test para aumentar a porcentaxe de cobertura no paquete `es.udc.pa.pa001.apuestas.model.event`, pois no método `getNumberOfEvents` acontecía o seguinte:

```

@Test
public final void testGetNumberEventsByKeyWordsCategoryUser() {

    /* SETUP */

    initializeCategories();
    initializeDates();
    initializeEvents();

    /* INVOCACION */

    int result = eventDao.getNumberOfEvents("Madrid", category1.getCategoryId(),
        false);

    /* ASERCIION */

    assertEquals(1, result);
}

final Calendar date = Calendar.getInstance();
if ((words == null) && (categoryId == null)) {
    if (!admin) {
        hqlQuery += " WHERE e.eventStart >= :date";
    }
} else {
    if (!admin) {
        hqlQuery += " AND e.eventStart >= :date";
    }
}

```

Na entidade `UserProfileDaoHibernate` temos varios casos de test que proban o método `findByLoginName`, e a cobertura de liña mostrada pola ferramenta *PITest* amosa unha porcentaxe do 100 %. Debido a que cremos que si que probamos esta clase, non imos realizar máis cambios relacionados con ela a pesar da porcentaxe mostrada pola ferramenta de *Cobertura*.

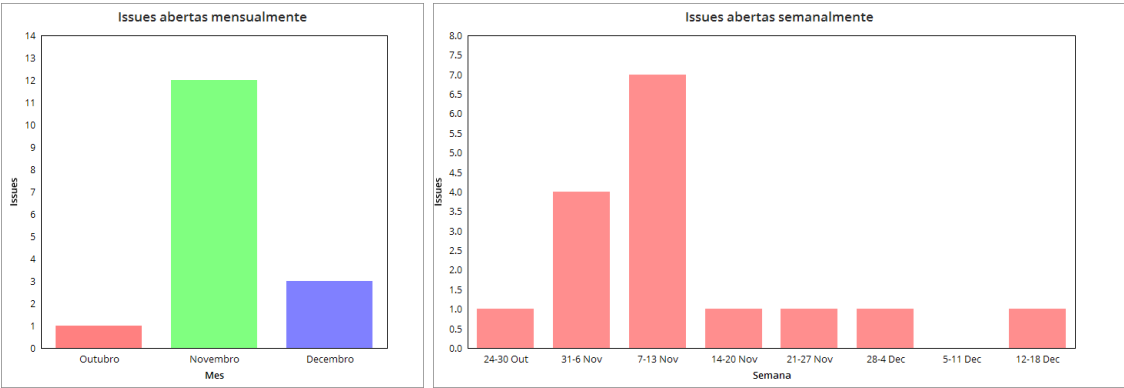
As porcentaxes de cobertura que non chegan ó 100 % son debidos a:

Paquete	Clase	Método
es.udc.pa.pa001.apuestas.model.bet	Bet	setBetedMoney
es.udc.pa.pa001.apuestas.model.bet	Bet	setDate
es.udc.pa.pa001.apuestas.model.bet	Bet	setUserProfile
es.udc.pa.pa001.apuestas.model.bet	Bet	setEvent
es.udc.pa.pa001.apuestas.model.bet	Bet	setBetOption
es.udc.pa.pa001.apuestas.model.betOption	BetOption	setAnswer
es.udc.pa.pa001.apuestas.model.betOption	BetOption	setRate
es.udc.pa.pa001.apuestas.model.category	Category	setName
es.udc.pa.pa001.apuestas.model.userprofile	Userprofile	setLoginName
es.udc.pa.pa001.apuestas.model.userprofile	Userprofile	setEncryptedPassword
es.udc.pa.pa001.apuestas.model.userprofile	Userprofile	setFirstName
es.udc.pa.pa001.apuestas.model.userprofile	Userprofile	setLastName
es.udc.pa.pa001.apuestas.model.userprofile	Userprofile	setEmail

## 6. Estatísticas

### 6.1. Número de erros atopados

Nas gráficas obsérvanse as *issues* abertas por mes e semana:



Na seguinte táboa recóllense as *issues* abertas durante a fase de probas según a data e a ferramenta que permitiu a detección dos erros:

Data	Número de <i>issues</i>	Ferramentas empregada para detectar o erro
24/10/16	1	Test de unidade ( <i>JUnit</i> e <i>Mockito</i> )
05/11/16	4	<i>Checkstyle</i>
08/11/16	6	<i>Checkstyle</i>
12/11/16	1	<i>Cobertura</i>
23/11/16	1	<i>PITest</i>
01/12/16	1	<i>Cobertura</i>
06/12/16	1	<i>Checkstyle</i>
16/12/16	1	<i>Cobertura</i> e <i>PITest</i>

## 6.2. Nivel de progreso na execución das probas

Finalmente o número de test executados na aplicación foi o seguinte:

Tests	Errors	Failures	Skipped	Success Rate	Time
110	1	0	0	99.091%	9.419

Figura 1: Número total de tests

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
es.udc.pa.pa001.apuestas.test.model.betservice	70	0	0	0	100%	0.652
es.udc.pa.pa001.apuestas.test.model.category	2	0	0	0	100%	0.023
es.udc.pa.pa001.apuestas.test.model.event	29	0	0	0	100%	0.466
es.udc.pa.pa001.apuestas.testautomation	1	1	0	0	0%	5.4
es.udc.pa.pa001.apuestas.test.model.bet	6	0	0	0	100%	2.854
es.udc.pa.pa001.apuestas.test.model.bettype	2	0	0	0	100%	0.024

Figura 2: Número de tests por paquete

Os resultados finais obtidos das execucións das ferramentas *Cobertura* e *PITest* son os seguintes:

### Coverage Report - All Packages

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	30	89% 312/347	94% 111/118	1,496
es.udc.pa.pa001.apuestas.model.bet	4	77% 34/44	N/A N/A	1
es.udc.pa.pa001.apuestas.model.betOption	3	83% 20/24	N/A N/A	1
es.udc.pa.pa001.apuestas.model.betType	3	100% 32/32	100% 2/2	1
es.udc.pa.pa001.apuestas.model.betservice	2	100% 75/75	100% 40/40	1,929
es.udc.pa.pa001.apuestas.model.betservice.util	8	100% 14/14	N/A N/A	1
es.udc.pa.pa001.apuestas.model.category	3	84% 11/13	N/A N/A	1
es.udc.pa.pa001.apuestas.model.event	4	98% 104/106	93% 69/74	2,565
es.udc.pa.pa001.apuestas.model.userprofile	3	51% 16/35	0% 0/2	1,125

Figura 3: Análise final de Cobertura

# Pit Test Coverage Report

## Project Summary

Number of Classes	Line Coverage	Mutation Coverage
14	92% <div><div></div></div> 308/334	99% <div><div></div></div> 155/156

## Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage
<a href="#">es.udc.pa.pa001.apuestas.model.bet</a>	3	77% <div><div></div></div> 34/44	100% <div><div></div></div> 11/11
<a href="#">es.udc.pa.pa001.apuestas.model.betOption</a>	1	83% <div><div></div></div> 19/23	100% <div><div></div></div> 5/5
<a href="#">es.udc.pa.pa001.apuestas.model.betType</a>	2	100% <div><div></div></div> 32/32	88% <div><div></div></div> 7/8
<a href="#">es.udc.pa.pa001.apuestas.model.betservice</a>	1	100% <div><div></div></div> 81/81	100% <div><div></div></div> 54/54
<a href="#">es.udc.pa.pa001.apuestas.model.category</a>	2	85% <div><div></div></div> 11/13	100% <div><div></div></div> 3/3
<a href="#">es.udc.pa.pa001.apuestas.model.event</a>	3	100% <div><div></div></div> 106/106	100% <div><div></div></div> 67/67
<a href="#">es.udc.pa.pa001.apuestas.model.userprofile</a>	2	71% <div><div></div></div> 25/35	100% <div><div></div></div> 8/8

Figura 4: Análise final de PITest

A última execução realizada da ferramenta *CheckStyle* é:

## Checkstyle Results

The following document contains the results of Checkstyle [📄](#). [XML](#)

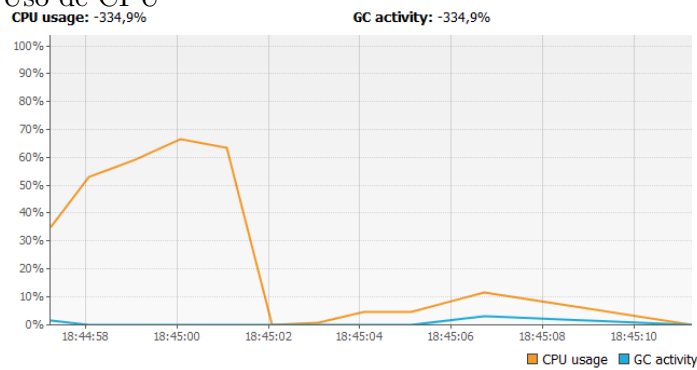
## Summary

Files	Infos ⓘ	Warnings ⚠	Errors ✖
88	0	0	341

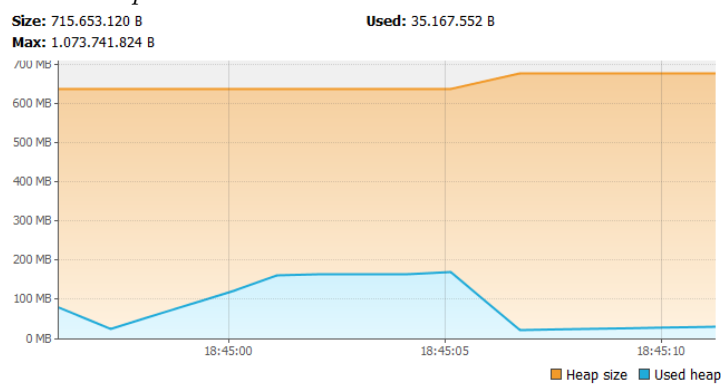
Figura 5: Análise final de CheckStyle

Por último, para analisar o rendimento da aplicação, empregamos a ferramenta *VisualVM* para monitorizar a execução dos tests, obtendo as seguintes gráficas como resultado:

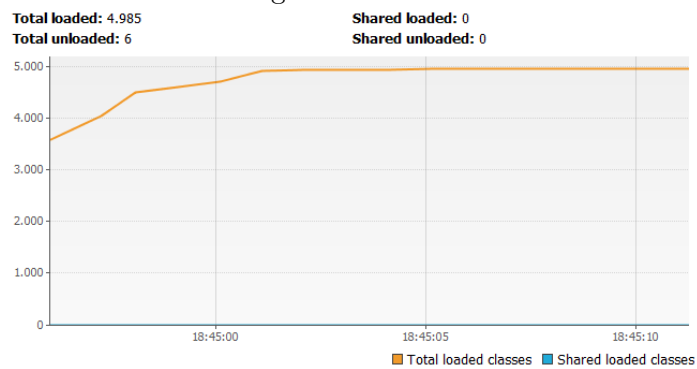
### ■ Uso de CPU



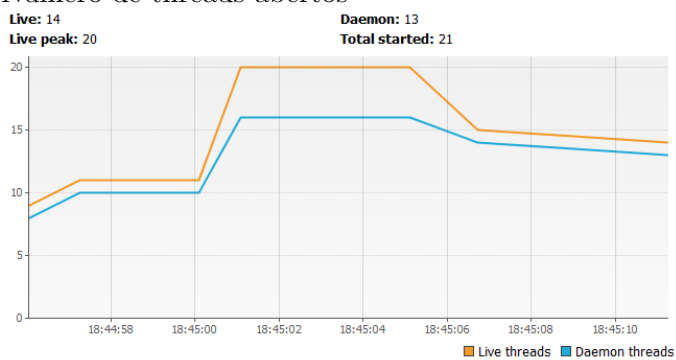
### ■ Uso de heap



### ■ Número de clases cargadas



### ■ Número de threads abiertos



### 6.3. Análise do perfil de detección de erros

O tipo de erros con maior presenza dentro da aplicación foi a violación de regras de *Checkstyle*, xa que o noso código non se axustaba a moitos dos estándares de codificación de Java. O número inicial de erros deste tipo era 9908, e estaban distribuídos uniformemente en todos os compoñentes.

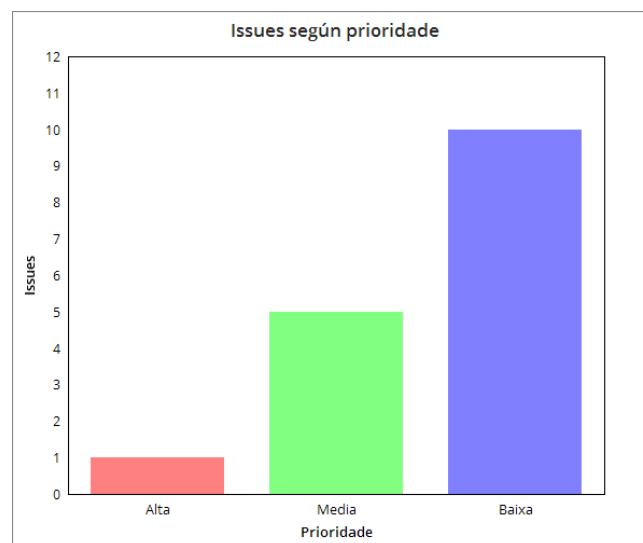
O seguinte tipo de erros máis habitual foi a falta de cobertura nalgúns partes do código, nalgúns casos provocada porque os métodos ou clases en cuestión non eran usados. Este tipo de erros concentrábanse principalmente nas entidades, xa que existían métodos *set* non probados.

Por outra banda detectamos un erro de comportamento no modelo, debido a que certas validacións eran delegadas na capa web. Este é o erro de maior prioridade que atopamos na fase de probas.

### 6.4. Informe de erros abertos e pechados por nivel de criticidade

Ao final da execución das probas, todas as *issues* foron pechadas. Todas elas foron resoltas agás dúas, que foron pechadas con *won't fix* debido a que eran de baixa prioridade e o tempo necesario para resolvelas sería demasiado elevado en comparación co valor que podían aportar ao noso código. As dúas *issues* non resoltas recollían erros relacionados con violacións de regras de *Checkstyle: Empty Block* e *Magic Number*.

Na seguinte gráfica podemos ver a distribución das *issues* según a súa prioridade:



- **Issues de prioridade alta:**

- Modificación do comportamento do modelo (<https://github.com/gemamb/VVS/issues/7>)

- **Issues de prioridade media:**

- Bug Checkstyle - JavaDoc (<https://github.com/gemamb/VVS/issues/9>)

- Cobertura Report - Añadir test makeBet (<https://github.com/gemamb/VVS/issues/18>)
- Bug Mutation Pitest *PITest* (<https://github.com/gemamb/VVS/issues/19>)
- Bug Cobertura - Constructores (<https://github.com/gemamb/VVS/issues/20>)
- Bug Cobertura y PITest - Event (<https://github.com/gemamb/VVS/issues/22>)

■ **Issues de prioridade baixa:**

- Bug Checkstyle - Spaces and Tabs (<https://github.com/gemamb/VVS/issues/8>)
- Bug Checkstyle - LineLength (<https://github.com/gemamb/VVS/issues/10>)
- Bug Checkstyle - DesignForExtension (<https://github.com/gemamb/VVS/issues/11>)
- Bug Checkstyle - LineLength and FileTabCharacter (<https://github.com/gemamb/VVS/issues/12>)
- Bug Checkstyle - Modifiers (<https://github.com/gemamb/VVS/issues/13>)
- Bug Checkstyle - Names (<https://github.com/gemamb/VVS/issues/14>)
- Bug Checkstyle - Brackets and Braces (<https://github.com/gemamb/VVS/issues/15>)
- Bug Checkstyle - MagicNumber (<https://github.com/gemamb/VVS/issues/16>)
- Bug Checkstyle - Conditionals (<https://github.com/gemamb/VVS/issues/17>)
- Bug Checkstyle - Block (<https://github.com/gemamb/VVS/issues/21>)

## 6.5. Avaliación global do estado de calidade e estabilidade actuais

Tras a execución da fase de probas podemos concluir que adaptamos o noso código aos estándares de Java, o que o fai máis lexible e fácil de manter. Ademais, o comportamento do modelo está amplamente probado (como amosan as análises de cobertura), o que nos aporta máis seguridade sobre o seu correcto funcionamento.

Cremos que o emprego das ferramentas *QuickCheck* e *GraphWalker* non foi demasiado significativo na nosa práctica. No primeiro caso, non tiñamos casos de uso para explotar esta ferramenta ao 100 %. Mentres que o *GraphWalker* foi unicamente empregado para as operacións CRUD dos DAOS, cun final fallido.

Ademais, a ferramenta *VisualVM* cremos que podería aportar máis valor combinada con algunha outra do estilo de *JMeter*, para poder simular carga de usuarios e así obter unha análise de rendemento máis realista. Debido a que a capa web queda fora desta fase de probas, decidimos non empregar dita ferramenta. Polo tanto, a análise de rendemento foi realizada unicamente sobre a execución dos tests implementados.

Sen embargo, as demais ferramentas permitíronnos probar amplamente todos os módulos a avaliar e corregir os erros e posibles vulnerabilidades da fase de desenvolvemento.

Con máis tempo poderíamos avaliar os demais módulos, así como o servizo de usuarios e a parte web da nosa aplicación.