Validación y Verificación de Software

INFORME DE PRÁCTICAS

Repositorio de proxecto:

https://github.com/elias-garcia/vvs

Participantes no proxecto:

-Elías García Mariño

-Lorena Reverendo Miguéns

-Santiago Villaviciosa Lodeiro

---

1. DESCRIPCIÓN DEL PROYECTO

La aplicación consiste en un servicio web de páginas de apuestas. Los usuarios visualizan eventos deportivos y pueden apostar a múltiples tipos de apuestas con sus respectivas opciones de apuesta y cuotas. Por otra parte existen administradores encargados de insertar y gestionar los eventos.
Para su implementación se ha seguido una arquitectura por capas sobre una plataforma Java EE. Se han usado un framework ORM(Hibernate) para gestionar la persistencia de las entidades, otro DI(Spring) para la inyección de dependencias y las gestión de las transacciones y Tapestry para desarrollar la capa Web. Utiliza MySQL como base de datos y Jetty como servidor de producción.
La aplicación consta de test implementados(JUnit), pero como el objetivo de la asignatura donde fue implementada no era el testing podemos decir que no cumplen una calidad aceptable y nuestra labor en esta práctica será alcanzarla.
Solo se han realizado pruebas de la capa modelo.

2. ESTADO ACTUAL

Listado de funcionalidades y su especificación:

Podemos diferenciar las funcionalidades en dos grupos según el tipo de usuario que puede usarlas. Los tipos de usuario son administrador y usuario. Se han numerado las funcionalidades para poder hacer referencias a ellas en todo este documento.

Funcionalidades de usuario:

-(F1) Registro de usuarios: Debe permitir registrar nuevos usuarios, así como permitir cambiar la información de registro posteriormente.

-(F2) Autenticación y salida: Un usuario se autenticará indicando su seudónimo y contraseña, con la posibilidad de recordar la contraseña para no tener que teclearla la siguiente vez. El usuario podrá salir explícitamente del sitio Web, lo que provoca que ya no se recuerde su contraseña, en caso de haber seleccionado dicha opción con anterioridad.

-(F3) Búsqueda de eventos: Cualquier usuario (aunque no esté autenticado) podrá buscar eventos que todavía no hayan comenzado, indicando las palabras clave del nombre y la categoría (siendo ambos opcionales). Así, por ejemplo, si el sitio Web contiene el evento “Deportivo - Real Madrid”, deberá aparecer cuando se busca “Deportivo - Real Madrid”, “Depor Madrid”, “madrid depor”, etc., es decir, las palabras clave tienen que estar todas contenidas en el nombre del evento como palabras o parte de palabras, sin distinguir entre mayúsculas y minúsculas, y en cualquier orden. Si el usuario especifica la categoría, la búsqueda se restringirá a los eventos de dicha categoría. Si el usuario especifica la categoría, pero no las palabras clave, la búsqueda mostrará todos los eventos (que no hayan empezado) de esa categoría. Si el usuario no

especifica ni las palabras clave ni la categoría, se mostrarán todos los eventos (que no hayan empezado). Los eventos que aparecen como resultado de una búsqueda se muestran ordenados por fecha de celebración, en orden ascendente, es decir, primero los más cercanos en el tiempo. Por cada evento se mostrará su nombre, la categoría a la que pertenece y la fecha de celebración. El nombre del evento estará asociado a un enlace que permitirá visualizar los detalles del evento, incluyendo todos sus tipos de apuesta y las opciones de apuesta de cada una de ellas.

-(F4) Realizar una apuesta: Cada opción de un tipo de apuesta llevará asociado un enlace que permitirá a un usuario autenticado apostar por esa opción. Si el usuario está autenticado, el enlace le llevará a la página con el formulario que permite apostar por esa opción. Si no está autenticado, se le redirigirá al formulario de autenticación, y tras autenticarse, se valorará positivamente que se le muestre automáticamente el formulario para apostar por esa opción (alternativamente, si no se logra implementar este grado de automatismo, es admisible que el usuario tenga que volver a buscar el evento, visualizar sus tipos de apuesta y pulsar sobre la opción elegida del tipo de apuesta por el que desea apostar).

-(F5) Consultar el estado de las apuestas: Un usuario autenticado podrá consultar las apuestas hechas a lo largo del tiempo, ordenadas por fecha de realización de la apuesta, en orden descendente, es decir, primero las más recientes. Por cada apuesta, se mostrará la fecha en la que se hizo, el nombre y fecha del evento, la pregunta del tipo de apuesta, la opción elegida, la cantidad de dinero que apostó, la opción (u opciones) ganadora, una indicación de su estado (pendiente, ganada o perdida), y en caso de que resultase ganadora, la ganancia (*lo que apostó por esa opción * cuota de ganancia de esa opción*).

Funcionalidades de administrador:

-**(F2) Autenticación y salida:** De la misma manera que un usuario normal.

-(F3) Búsqueda de eventos: El sitio Web le permitirá buscar eventos de la misma forma que al usuario normal. Sin embargo, a diferencia del usuario final, el resultado de la búsqueda incluirá tanto eventos que hayan comenzado como eventos pendientes de comenzar.

-(F6) Inserción de eventos: El sitio Web le ofrecerá un enlace para añadir un nuevo evento.

-(F7) Gestión de tipos de apuesta: Cuando el administrador hace clic sobre el nombre de uno de los eventos que aparecen en el resultado de la búsqueda, se le muestra la misma página que al usuario final, con las siguientes diferencias:

1. Si el evento todavía no comenzó, se mostrará un enlace para añadir un nuevo tipo de apuesta.
2. Si el evento ya comenzó, para cada tipo de apuesta a la que todavía no se le hayan especificado la opción u opciones ganadoras, se mostrará un enlace que permite seleccionarlas.

3. Las opciones de los tipos de apuestas no disponen de enlace para apostar. Para los tipos de apuesta que ya tengan fijadas las opciones ganadoras, éstas se resaltan de alguna manera.

Para no alargar innecesariamente la práctica, la información sobre categorías se introducirá mediante sentencias INSERT INTO en un script SQL (ejecutado desde Maven, al igual que el script de creación de tablas).

Personas responsables de su desarrollo

Elías García Mariño
Diego Tejeda Valcárcel

Personas responsables del proceso de prueba

Elías García Mariño
Lorena Reverendo Miguéns
Santiago Villaviciosa Lodeiro

2.1. COMPONENTES EVALUADOS

Componente	Funcionalidades en las que participa	Número pruebas objetivo	Número pruebas preparadas	Porcentaje ejecutado	Porcentaje superado
BetInfo	F4 y F5	3	3	100%	100%
BetService	F4 y F5	15	15	100%	100%
CategoryInfo	F3, F6 y F7	7	7	100%	100%
EventInfo	F3, F5, F6 y F7	6	6	100%	100%
EventService	F3, F6 y F7	44	44	100%	100%
TypeOption		1	1	100%	100%
UserProfile	F1, F2, F4 y F5	2	2	100%	100%
UserService	F1 y F2	26	26	100%	100%

3. ESPECIFICACIÓN DE PROBAS

- **Pruebas de unidad**

PR-UN-000

Unidad: CategoryInfoDao

Método: save y find

Motivación: Probar que el metodo save del GenericDao hace persistente un objeto CategoryInfo. GenericDaosave(E entity) no devuelve el objeto después de hacerlo persistente, por lo tanto la única manera de comprobar si el dato es persistente es usar el método GenericDaosave.find(PK id).

En conclusión, tambien estariamos probando que el metodo GenericDaosave.find(PK id) busca un objeto ya persistente correctamente.

Valores de entrada:

- save:
  - -CategoryInfo
- -find:
  - -CategoryInfoId

Valores de salida:

- NA

Inicialización:

- CategoryInfo

PR-UN-001

Unidad: CategoryInfoDao

Método: find

Motivación: Obtener la excepcion InstanceNotFoundException al pasarle un identificador de categoria no existente.

Valores de entrada:

- Identificador de categoria no existente

Valores de salida:

- InstanceNotFoundException

Inicialización:

- CategoryInfo

PR-UN-002

Unidad: CategoryInfoDao

Método: remove

Motivación: Comprobar que un objeto persistente es borrado correctamente.

Valores de entrada:
- save:
  CategoryInfo
- remove:
  CategoryInfoId
- find:
  CategoryInfoId

Valores de salida:
- NA

Inicialización:
- CategoryInfo

PR-UN-003

Unidad: CategoryInfoDao

Método: remove

Motivación: Probar que al intentar eliminar un CategoryInfo con un identificador no existente se genera la excepcion InstanceNotFoundException.

Valores de entrada:
- Identificador de categoria no existente

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo

PR-UN-004

Unidad: CategoryInfoDao

Método: save

Motivación: Probar que el metodo save actualiza con los datos de tenga la entidad en ese momento.

Valores de entrada:

- save:
  - CategoryInfo
- find:
  - CategoryInfoId

Valores de salida:

- Todas las categorias existentes hasta el momento.

Inicialización:

- CategoryInfo

PR-UN-005

Unidad: CategoryInfoDao

Método: findAllCategories

Motivación: Probar que el metodo findAllCategories devuelve todas las categorias persistentes hasta el momento.

Valores de entrada:

- NA

Valores de salida:

- Todas las categorias persistentes hasta el momento. Es decir, el mismo CategoryInfo que la inicializacion.

Inicialización:

- CategoryInfo

PR-UN-006

Unidad: CategoryInfoDao

Método: findAllCategories

Motivación: Probar que el metodo findAllCategories no devuelve nada si no hay objetos persistentes en ese momento.

Valores de entrada:

- NA

Valores de salida:

- NA

Inicialización:

- NA

PR-UN-007

Unidad: EventInfoDao

Método: findEvents

Para no tener que probarlos todos ya que segun los posibles valores de los parametros de entradas tendriamos demasiadas combinaciones por lo tanto demasiados casos de prueba y por tiempo decidimos que vamos a probar cada parametro por separado y un caso de prueba general que devuelva todas los eventos(sin filtro).

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento
con los filtros(parametros) de tal manera que se obtienen todas los eventos.

Valores de entrada:
- keywords: null
- categoryId: null
- eventsStarted: true
- startIndex: 0
- count: 4

Valores de salida:
- Todos los EventInfo insertados hasta el momento ordenados por la fecha ascendentemente.

Inicialización:
- CategoryInfo
- 4 EventInfo

PR-UN-008

Unidad: EventInfoDao

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento
filtrados por las keywords.

Valores de entrada:
- keywords: Cadena de caracteres coincidente/contenida con el nombre de dos eventos.
- categoryId: null
- eventsStarted: true
- startIndex: 0
- count: 4

Valores de salida:
- 2 EventInfo con el nombre coincidente/contenida de las keywords ordenados por la fecha ascendentemente.

Inicialización:

- CategoryInfo
- 4 EventInfo

PR-UN-010

Unidad: EventInfoDao

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento
filtrados por el identificador de una categoria existente.

Valores de entrada:
- keywords: null
- categoryId: Identificador de una categoria existente.
- eventsStarted: true
- startIndex: 0
- count: 4

Valores de salida:
- EventInfo pertenecientes a esa categoria ordenados por la fecha ascendentemente.

Inicialización:
- 2 CategoryInfo
- 4 EventInfo, 2 EventInfo asignados a cada CategoryInfo por separado.

PR-UN-011

Unidad: EventInfoDao

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento
filtrados por si son eventos activos(posteriores a la fecha actual).

Valores de entrada:
- keywords: null
- categoryId: null
- eventsStarted: false
- startIndex: 0
- count: 4

Valores de salida:
- 2 EventInfo activos ordenados por la fecha ascendentemente.

Inicialización:
- CategoryInfo
- 4 EventInfo, 2 EventInfo con la fecha posterior a la actual y otros

dos anterior a la actual.

PR-UN-012

Unidad: EventInfoDao

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento
filtrados por startIndex y count.

Valores de entrada:
- keywords: null
- categoryId: null
- eventsStarted: true
- startIndex: 0
- count: 2

Valores de salida:
- 2 EventInfo activos ordenados por la fecha ascendentemente.

Inicialización:
- CategoryInfo
- 4 EventInfo.

PR-UN-013

Unidad: UserProfileDao

Método: findByLoginName

Motivación: Probar que la función devuelve la excepción InstanceNotFoundException cuando se pasa un loginName que no coincide con ningún userProfile existente.

Valores de entrada:
- Username de un UserProfile no existente

Valores de salida:
- InstanceNotFoundException

Inicialización:
- UserProfile

PR-UN-014

Unidad: UserProfileDao

Método: findByLoginName

Motivación: Probar que la función devuelve el UserProfile correctamente en funcion del loginName

Valores de entrada:
- loginName de un UserProfile existente

Valores de salida:
- UserProfile correpondiente al loginName correspondiente

Inicialización:
- UserProfile con el loginName del que despues se busca

PR-UN-015

BetInfoDao - findBetsByUserId

Motivación: Comprobar que la función devuelve los bets de un user existente.
Se prueba, además, que estos sean tantos como se indica por count.

Entradas: userId = Id del user creado en la inicialización
startIndex = 0
count = 4

Salida: Los Bets del user creados en la inicialización.

Inicialización: Creamos dos user, un typeOption y un event asignado a typeOption.
Necesitamos crear también una category y betType que iran asignados al event, así
mismo el betType va a asignado también al TypeOption.
Además creamos 2 Bets y se los asignamos al user con el typeOption.

PR-UN-016

BetInfoDao - findBetsByUserId

Motivación: Comprobar que la función devuelve una lista vacía para un user que no tiene bets.

Entradas: userId = Id del user creado en la inicialización
startIndex = 0
count = 4

Salida: List vacía.

Inicialización: Creamos dos user.

Por falta de tiempo no comprobamos el caso de un usuario no existente.
Podemos obviarlo ya que esto se controlara en el BetService.

PR-UN-017

Unidad: EventServiceImpl

Método: createEvent

Motivación: Crear un evento correctamente con un nombre valido(not null y algun caracter sin ser espacio), una fecha valida(posterior al actual) y un id de una categoria existente.

Valores de entrada:
- eventName: Real Madrid - Barcelona
- eventDate: Fecha posterior a la actual
- categoryId: Identificador de un categoria existente

Valores de salida:
- EventInfo

Inicialización:
- CategoryInfo
- EventInfo

PR-UN-018

Unidad: EventServiceImpl

Método: createEvent

Motivación: Crear un evento con el nombre de evento con el valor null.
El nombre del evento no puede ser null(restriccion de diseño), por lo tanto se tendria que generar una excepcion NullEventNameException.

Valores de entrada:
- eventName: null
- eventDate: Fecha posterior a la actual
- categoryId: Identificador de un categoria existente

Valores de salida:
- NullEventNameException

Inicialización:
- CategoryInfo
- EventInfo

Se podria comprobar que el nombre tuviese carácteres alfanumericos y no una cadena de texto vacia o construida por espacios, pero por falta de tiempo hemos obviado ese caso de prueba.

Nueva issue: Inspeccionando el código se ha dectado que la implementacion no devuelve NullEventNameException de hecho ni siquiera esta creada. Crearla y modificar la implementacion

para que devuelva la excepcion.

PR-UN-019

Unidad: EventServiceImpl

Método: createEvent

Motivación: Crear un evento con la fecha de evento con el valor null.
Se deberia lanzar la excepcion EventDateException.
Valores de entrada:
- eventName: Real Madrid - Barcelona
- eventDate: null
- categoryId: Identificador de un categoria existente

Valores de salida:
- EventDateException

Inicialización:
- CategoryInfo
- EventInfo

Nueva issue: Inspeccionando el código se ha dectado que la implementacion no devuelve EventDateException cuando la fecha es null. Modificar la implementacion para que devuelva la excepcion en ese caso.

PR-UN-020

Unidad: EventServiceImpl

Método: createEvent

Motivación: Obtener excepción EventDateException
al intentar crear un evento con una fecha anterior a
la actual.

Valores de entrada:
- eventName: Real Madrid - Barcelona
- eventDate: Fecha anterior a la actual
- categoryId: Identificador de un categoria existente

Valores de salida:
- EventDateException

Inicialización:
- CategoryInfo
- EventInfo

PR-UN-021

Unidad: EventServiceImpl

Método: createEvent

Motivación: Obtener excepción InstanceNotFoundException
al intentar crear un evento con identificador de categoria
no existente.

Valores de entrada:
- eventName: Real Madrid - Barcelona
- eventDate: Fecha posterior a la actual
- categoryId: Identificador de un categoria no existente

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo

PR-UN-022

Unidad: EventServiceImpl

Método: findEvent

Motivación: Obtener excepción InstanceNotFoundException
al buscar un evento con identificador de evento
no existente.

Valores de entrada:
- Identificador de un evento no existente.

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo

PR-UN-023

Unidad: EventServiceImpl

Método: findEvent

Motivación: Comprobar que se busca correctamente un evento
por su identificador.

Valores de entrada:
- eventName: Identificador de un evento existente.

Inicialización:

- CategoryInfo
- EventInfo

PR-UN-024

Unidad: EventServiceImpl

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento segun
el startIndex y el account, devolviendo el correspondiente EventInfoBlock.

Valores de entrada:

- keywords: null
- categoryId: null
- eventsStarted: true
- startIndex: 0
- count: 4

Valores de salida:

- EventInfoBlock
  - events: Todos los EventInfo insertados hasta el momento ordenados por la fecha ascendentemente.
  - existsMoreBets: false.

Inicialización:

- CategoryInfo
- EventInfo

PR-UN-025

Unidad: EventServiceImpl

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento segun
el startIndex y el account, devolviendo el correspondiente EventInfoBlock.

Valores de entrada:

- keywords: null
- categoryId: null
- eventsStarted: true
- startIndex: 0
- count: 2

Valores de salida:

- EventInfoBlock

- events: Los dos primeros eventos insertados hasta el momento ordenados por la fecha ascendentemente.
- existsMoreBets: true.

Inicialización:
- CategoryInfo
- EventInfo

PR-UN-026

Unidad: EventServiceImpl

Método: findEvents

Motivación: Obtener un StartIndexOrCountException al pasarle al metodo los parametros startIndex o count negativos.

Valores de entrada:
- keywords: null
- categoryId: null
- eventsStarted: true
- startIndex: -1
- count: -1

Valores de salida:
- StartIndexOrCountException

Inicialización:
- CategoryInfo
- 4 EventInfo.

Nueva issue: Inspeccionando el código se ha dectado que la implementacion no devuelve StartIndexOrCountException es este metodo para este caso. Crearla y modificar la implementación para que devuelva la excepcion.

PR-UN-027

Unidad: EventServiceImpl

Método: addBetType

Motivación: Comprobar que se inserta adecuadamente una BetType en el sistema con sus BetTypeOption correspondientes asociada a un EventInfo.

Valores de entrada:
- eventId: Identificador de un evento existente
- type: BetType

- options: Lista de BetTypeOption

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOptions

PR-UN-028

Unidad: EventServiceImpl

Método: addBetType

Motivación: Obtener excepción InstanceNotFoundException
al intentar crear añadir un BetType con un identificador
de evento no existente.

Valores de entrada:
- eventId: Identificador de un evento no existente
- type: BetType
- options: Lista de BetTypeOption

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOptions

PR-UN-029

Unidad: EventServiceImpl

Método: addBetType

Motivación: Segun la especificación un BetType no puede
no tener la lista de BetTypeOption vacia. Por lo tanto
deberia devolver una excepcion NoAssignedTypeOptionsException

Valores de entrada:
- eventId: Identificador de un evento existente
- type: BetType
- options: Lista de BetTypeOption vacia o null

Valores de salida:

- CategoryInfo
- EventInfo

Nueva issue: Inspeccionando el código se ha dectado que la implementacion no devuelve NoAssignedTypeOptionsException es este metodo para este caso. Crearla y modificar la implementación para que devuelva la excepcion.

PR-UN-030

Unidad: EventServiceImpl

Método: addBetType

Motivación: Segun la especificación un BetType no puede
no tener la lista de BetTypeOption con el mismo nombre.
Por lo tanto deberia devolver una excepcion DuplicatedResultTypeOptionsException.

Valores de entrada:
- eventId: Identificador de un evento existente
- type: BetType
- options: Lista de BetTypeOption con al menos
dos TypeOptions con los result con el mismo
valor.

Valores de salida:
- DuplicatedResultTypeOptionsException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOptions con al menos 2 TypeOptions con el mismo nombre

Nueva issue: Inspeccionando el código se ha dectado que la implementacion no devuelve DuplicatedResultTypeOptionsException es este metodo para este caso. Crearla y modificar la implementación para que devuelva la excepcion.

PR-UN-031

Unidad: EventServiceImpl

Método: findAllCategories

Motivación: Obtener la lista de categorias insertadas
hasta el momento.

Valores de entrada:
- NA

Valores de salida:

- List<CategoryInfo>: Resultados obtenidos.

Inicialización:

- CategoryInfo

PR-UN-032

Unidad: BetServiceImpl

Método: createBet

Motivación: Crear una apuesta correctamente sobre la opción indicada y con la cantidad indicada por el usuario.

Valores de entrada:

- userId: identificador de un usuario existente
- typeOptionId: identificador de una opción existente
- amount: cantidad positivo de dinero

Valores de salida:

- BetInfo

Inicialización:

- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-UN-033

Unidad: BetServiceImpl

Método: createBet

Motivación: Obtener excepción InstanceNotFoundException al intentar crear una apuesta con un identificador de usuario no existente.

Valores de entrada:

- userId: identificador de un usuario no existente
- typeOptionId: identificador de una opción existente
- amount: cantidad positivo de dinero

Valores de salida:

- InstanceNotFoundException

Inicialización:

- N/A

Unidad: BetServiceImpl

Método: createBet

Motivación: Obtener excepción InstanceNotFoundException al intentar crear una apuesta con un identificador de opción no existente.

Valores de entrada:
- userId: identificador de un usuario existente
- typeOptionId: identificador de una opción no existente
- amount: cantidad positivo de dinero

Valores de salida:
- InstanceNotFoundException

Inicialización:
- UserProfile

PR-UN-035

Unidad: BetServiceImpl

Método: findBetsByUserId

Motivación: Obtener todas las apuestas realizadas por un usuario.

Valores de entrada:
- userId: identificador de un usuario existente
- startIndex: 0
- count: 4

Valores de salida:
- BetInfoBlock:
  - bets: todas las apuestas realizadas por el usuario
  - existsMoreBets: false

Inicialización:
- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption
- 4 Apuestas hechas por el usuario.

PR-UN-036

Unidad: BetServiceImpl

Método: findBetsByUserId

Motivación: Obtener una parte de todas las apuestas realizadas por un usuario.

Valores de entrada:

- userId: identificador de un usuario existente
- startIndex: 0
- count: 3

Valores de salida:

- BetInfoBlock:
  - bets: 3 apuestas realizadas por el usuario
  - existsMoreBets: true

Inicialización:

- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption
- 4 Apuestas hechas por el usuario.

PR-UN-037

Unidad: BetServiceImpl

Método: findBetsByUserId

Motivación: Obtener InstanceNotFoundException al intentar buscar las apuestas de un usuario no existente.

Valores de entrada:

- userId: identificador de un usuario no existente
- startIndex: 1
- count: entero

Valores de salida:

- InstanceNotFoundException

Inicialización:

- N/A

PR-UN-038

UserService- registerUser

Motivación: Comprobar que se registra correctamente un usuario.

Entradas: loginName = "user1"
clearPassword = "passwd"
userProfileDetails = UserProfileDetails

Salida: userProfile creado

Inicialización: -

PR-UN-039

UserService- registerUser

Motivación: Comprobar que al intentar registrar un usuario ya existente salta un excepción.

Entradas: loginName = "user1"
clearPassword = "passwd"
userProfileDetails = UserProfileDetails

Salida: DuplicateInstanceException

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-UN-040

UserService- login

Motivación: Comprobar que la contraseña pasada como entrada y la contraseña del usuario que corresponde
al loginName pasado, sean iguales.

Entradas: loginName = "user1"
password = "pass"
passwordEncrypted = false

Salida: userProfile creado en la inicialización.

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-UN-041

UserService- login

Motivación: Comprobar que la contraseña pasada como entrada y la contraseña del usuario que corresponde
al loginName pasado, no sean iguales y por tanto salte una excepción.

Entradas: loginName = "user1"
password = "pass2"
passwordEncrypted = false

Salida: IncorrectPasswordException

Inicialización: Creamos un user con loginName = "user1", password = "passwd" y UserProfileDetails.

PR-UN-042

UserService- login

Motivación: Comprobar que la función devuelve una excepción al pasarle un userName de un user no existente.

Entradas: loginName = "user2"
password = "pass"
passwordEncrypted = false

Salida: InstanceNotFoundException

Inicialización: Creamos un user con loginName = "user1", password = "passwd" y UserProfileDetails.

PR-UN-043

UserService- login

Motivación: Comprobar que la contraseña encriptada pasada como entrada y la contraseña encriptada del usuario que corresponde
al loginName pasado, sean iguales.

Entradas: loginName = "user1"
password = "pass"
passwordEncrypted = true

Salida: userProfile creado en la inicialización.

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-UN-044

UserService- findUserProfile

Motivación: Comprobar que la función devuelve una excepción al pasarle un id de un user no existente.

Entradas: userId = -1

Salida: InstanceNotFoundException

Inicialización: Creamos un user

PR-UN-045

UserService- findUserProfile

Motivación: Comprobar que la función devuelve un user al pasarle un id de un user existente.

Entradas: userId = userId del user creado en la inicialización

Salida: user creado en la inicialización

Inicialización: Creamos un user

PR-UN-046

UserService- changePassword

Motivación: Comprobar que la contraseña se cambia adecuadamente pasando un user existente.

Entradas: userProfileId = id del user creado en inicialización
oldClearPassword = password del user creado en inicialización
newClearPassword = "Xpass"

Salida: userProfile creado en la inicialización con la password nueva.

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-UN-047

UserService- changePassword

Motivación: Comprobar que la contraseña antigua pasada como entrada y la contraseña del usuario pasado,
no sean iguales y por tanto salte una excepción.

Entradas: userProfileId = id del user creado en inicialización
oldClearPassword = "Xpass"
newClearPassword = "Ypass"

Salida: IncorrectPasswordException

Inicialización: Creamos un user con loginName = "user1", password = "passwd" y UserProfileDetails.

PR-UN-048

UserService- changePassword

Motivación: Comprobar que la función devuelve una excepción al pasarle un userId de un user no existente.

Entradas: userProfileId = -1
oldClearPassword = "pass"
newClearPassword = "newPass"

Salida: InstanceNotFoundException

Inicialización: Creamos un user con loginName = "user1", password = "passwd" y UserProfileDetails.

PR-UN-049

UserService- updateUserProfileDetails

Motivación: Comprobar que el profileDetails se cambia adecuadamente pasando un user existente.

Entradas: userProfileId = id del user creado en inicialización
userProfileDetails = nuevo UserProfileDetails

Salida: userProfile creado en la inicialización con la password nueva.

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-UN-050

UserService- updateUserProfileDetails

Motivación: Comprobar que la función devuelve una excepción al pasarle un userId de un user no existente.

Entradas: userProfileId = -1
userProfileDetails = nuevo UserProfileDetails

Salida: InstanceNotFoundException

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-UN-051

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que en el caso de pasarle un betType con identificador no existente el metodo devuelve la excepcion correspondiente.

Valores de entrada:
- optionIds: Identificadores de las opciones ganadoras
- betTypeId: Identificador no existente del tipo de apuesta

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-UN-052

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que si no se pasaron identificadores de opciones a null se pusieron todas las opciones del tipo de apuesta a falso y el atributo pickedWinners a true.

Valores de entrada:
- optionIds: null
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-UN-053

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que si no se pasaron identificadores de opciones como una lista vacia se pusieron todas las opciones del tipo de apuesta a falso y el atributo pickedWinners a true.

Valores de entrada:
- optionIds: lista vacia
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-UN-054

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que si no se pasaron identificadores de opciones no existentes se devuelve la excepcion InstanceNotFoundException

Valores de entrada:
- optionIds: identificadores de apuestas(al menos uno) no existente
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-UN-055

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que el metodo devuelve TypeNotMultipleException si la lista de opciones ganadores contiene mas de una opcion y el tipo de apuesta no admite multiples.

Valores de entrada:
- optionIds: Lista con dos opciones existentes
- betTypeId: Identificador del tipo de apuesta que no admite multiples apuestas

Valores de salida:
- TypeNotMultipleException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-UN-056

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que se ponen las opciones ganadoras correctamente (atributo isWinner a true).

Valores de entrada:
- optionIds: Opcion ganadora
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-UN-057

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que se ponen las opciones no ganadoras correctamente (atributo isWinner a false).

Valores de entrada:
- optionIds: Opcion ganadora
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-UN-058

Unidad: EventInfoDao

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento
filtrados por unas keywords que contengan 2 palabras(strings separados por espacios), un identificador de una categoria existente y en cualquier momento.

Valores de entrada:

- keywords: String que corresponda con el nombre de un evento ya almacenado en base de datos
- categoryId: Identificador de una categoria existente de un evento almacenado en base de datos
- eventsStarted: false
- startIndex: 0
- count: 2

Valores de salida:
- 1 EventInfo con el nombre coincidente con las keywords y con la misma categoria que el identificador de la categoria pasada por parametro.

Inicialización:
- CategoryInfo
- 4 EventInfo.

PR-UN-059

BetInfoDao - testRemainingReturnedValues

Método - find

Motivación: Comprobar que los getters de amount y date no devuelven un valor negativo o nulo respectivamente, con el fin de matar dos mutantes.

Entradas: El id de la primera apuesta creada.

Salida: La primera apuesta creada.

Inicialización: Creamos dos user, un typeOption y un event asignado a typeOption.
Necesitamos crear también una category y betType que iran asignados al event, así mismo el betType va a asignado también al TypeOption.
Además creamos 2 Bets y se los asignamos al user con el typeOption.

PR-UN-060

Unidad: TypeOptionDao

Método: find

Motivación: Probar que la entidad TypeOption devuelve la odd correcta con el fin de matar un mutante.

Valores de entrada:
- typeOptionId: identificador de un typeOption existente.

Valores de salida:
- TypeOption

Inicialización:

- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption

- **Pruebas de Integración**

PR-IT-001

Unidad: BetServiceImpl

Método: createBet

Motivación: Crear una apuesta correctamente sobre la opción indicada y con la cantidad indicada por el usuario.

Valores de entrada:

- userId: identificador de un usuario existente
- typeOptionId: identificador de una opción existente
- amount: cantidad positivo de dinero

Valores de salida:

- BetInfo

Inicialización:

- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-IT-002

Unidad: BetServiceImpl

Método: createBet

Motivación: Obtener excepción NegativeAmountException al crear una apuesta con una cantidad negativa.

Valores de entrada:

- userId: identificador de un usuario existente
- typeOptionId: identificador de una opción existente
- amount: cantidad negativa de dinero

Valores de salida:

- NegativeAmountException

Inicialización:

- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-IT-003

Unidad: BetServiceImpl

Método: createBet

Motivación: Obtener excepción InstanceNotFoundException al intentar crear una apuesta con un identificador de usuario no existente.

Valores de entrada:
- userId: identificador de un usuario no existente
- typeOptionId: identificador de una opción existente
- amount: cantidad positivo de dinero

Valores de salida:
- InstanceNotFoundException

Inicialización:
- N/A

PR-IT-004

Unidad: BetServiceImpl

Método: createBet

Motivación: Obtener excepción InstanceNotFoundException al intentar crear una apuesta con un identificador de opción no existente.

Valores de entrada:
- userId: identificador de un usuario existente
- typeOptionId: identificador de una opción no existente
- amount: cantidad positivo de dinero

Valores de salida:
- InstanceNotFoundException

Inicialización:
- UserProfile

PR-IT-005

Unidad: BetServiceImpl

Método: findBetsByUserId

Motivación: Obtener todas las apuestas realizadas por un usuario.

Valores de entrada:
- userId: identificador de un usuario existente
- startIndex: 0
- count: 4

Valores de salida:
- BetInfoBlock:
- bets: todas las apuestas realizadas por el usuario
- existsMoreBets: false

Inicialización:
- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption
- 3 Apuestas hechas por el usuario.

PR-IT-006

Unidad: BetServiceImpl

Método: findBetsByUserId

Motivación: Obtener una parte de todas las apuestas realizadas por un usuario.

Valores de entrada:
- userId: identificador de un usuario existente
- startIndex: 0
- count: 3

Valores de salida:
- BetInfoBlock:
- bets: 3 apuestas realizadas por el usuario
- existsMoreBets: true

Inicialización:
- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption
- 3 Apuestas hechas por el usuario.

PR-IT-007

Unidad: BetServiceImpl

Método: findBetsByUserId

Motivación: Obtener InstanceNotFoundException al intentar buscar las apuestas de un usuario no existente.

Valores de entrada:
- userId: identificador de un usuario no existente
- startIndex: 1
- count: entero

Valores de salida:
- InstanceNotFoundException

Inicialización:
- N/A

PR-IT-008

UserService- registerUser

Motivación: Comprobar que se registra correctamente un usuario.

Entradas: loginName = "user1"
clearPassword = "passwd"
userProfileDetails = UserProfileDetails

Salida: userProfile creado

Inicialización: -

PR-IT-009

UserService- registerUser

Motivación: Comprobar que al intentar registrar un usuario ya existente salta un excepción.

Entradas:loginName = "user1"
clearPassword = "passwd"
userProfileDetails = UserProfileDetails

Salida: DuplicateInstanceException

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-IT-010

encriptada del usuario que corresponde
al loginName pasado, sean iguales.

Entradas: loginName = "user1"
password = "pass"
passwordEncrypted = true

Salida: userProfile creado en la inicialización.

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-IT-014

UserService- findUserProfile

Motivación: Comprobar que la función devuelve una excepción al pasarle un id de un user no existente.

Entradas: userId = -1

Salida: InstanceNotFoundException

Inicialización: Creamos un user

PR-IT-015

UserService- findUserProfile

Motivación: Comprobar que la función devuelve un user al pasarle un id de un user existente.

Entradas: userId = userId del user creado en la inicialización

Salida: user creado en la inicialización

Inicialización: Creamos un user

PR-IT-016

UserService- changePassword

Motivación: Comprobar que la contraseña se cambia adecuadamente pasando un user existente.

Entradas: userProfileId = id del user creado en inicialización
oldClearPassword = password del user creado en inicialización
newClearPassword = "Xpass"

Salida: userProfile creado en la inicialización con la password nueva.

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-IT-017

UserService- changePassword

Motivación: Comprobar que la contraseña antigua pasada como entrada y la contraseña del usuario pasado,
no sean iguales y por tanto salte una excepción.

Entradas: userProfileId = id del user creado en inicialización
oldClearPassword = "Xpass"
newClearPassword = "Ypass"

Salida: IncorrectPasswordException

Inicialización: Creamos un user con loginName = "user1", password = "passwd" y UserProfileDetails.

PR-IT-018

UserService- changePassword

Motivación: Comprobar que la función devuelve una excepción al pasarle un userId de un user no existente.

Entradas: userProfileId = -1
oldClearPassword = "pass"
newClearPassword = "newPass"

Salida: InstanceNotFoundException

Inicialización: Creamos un user con loginName = "user1", password = "passwd" y UserProfileDetails.

PR-IT-019

UserService- updateUserProfileDetails

Motivación: Comprobar que el profileDetails se cambia adecuadamente pasando un user existente.

Entradas: userProfileId = id del user creado en inicialización
userProfileDetails = nuevo UserProfileDetails

Salida: userProfile creado en la inicialización con la password nueva.

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-IT-020

UserService- updateUserProfileDetails

Motivación: Comprobar que la función devuelve una excepción al pasarle un userId de un user no existente.

Entradas: userProfileId = -1
userProfileDetails = nuevo UserProfileDetails

Salida: InstanceNotFoundException

Inicialización: Creamos un user con loginName = "user1", password = "pass" y UserProfileDetails.

PR-IT-021

Unidad: EventServiceImpl

Método: createEvent

Motivación: Crear un evento correctamente con un nombre valido(not null y algun caracter sin ser espacio), una fecha valida(posterior al actual) y un id de una categoria existente.

Valores de entrada:
- eventName: Real Madrid - Barcelona
- eventDate: Fecha posterior a la actual
- categoryId: Identificador de un categoria existente

Valores de salida:
- EventInfo

Inicialización:
- CategoryInfo
- EventInfo

PR-IT-022

Unidad: EventServiceImpl

Método: createEvent

Motivación: Crear un evento con el nombre de evento con el valor null.
El nombre del evento no puede ser null(restriccion de diseño), por lo tanto se tendria que generar una excepcion NullEventNameException.

Valores de entrada:
- eventName: null
- eventDate: Fecha posterior a la actual

- categoryId: Identificador de un categoria existente

Valores de salida:
- NullEventNameException

Inicialización:
- CategoryInfo
- EventInfo

PR-IT-023

Unidad: EventServiceImpl

Método: createEvent

Motivación: Crear un evento con la fecha de evento con el valor null.
Se deberia lanzar la excepcion EventDateException.
Valores de entrada:
- eventName: Real Madrid - Barcelona
- eventDate: null
- categoryId: Identificador de un categoria existente

Valores de salida:
- EventDateException

Inicialización:
- CategoryInfo
- EventInfo

PR-IT-024

Unidad: EventServiceImpl

Método: createEvent

Motivación: Obtener excepción EventDateException
al intentar crear un evento con una fecha anterior a
la actual.

Valores de entrada:
- eventName: Real Madrid - Barcelona
- eventDate: Fecha anterior a la actual
- categoryId: Identificador de un categoria existente

Valores de salida:
- EventDateException

Inicialización:
- CategoryInfo
- EventInfo

PR-IT-025

Unidad: EventServiceImpl

Método: createEvent

Motivación: Obtener excepción InstanceNotFoundException
al intentar crear un evento con identificador de categoria
no existente.

Valores de entrada:
- eventName: Real Madrid - Barcelona
- eventDate: Fecha posterior a la actual
- categoryId: Identificador de un categoria no existente

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo

PR-IT-026

Unidad: EventServiceImpl

Método: findEvent

Motivación: Obtener excepción InstanceNotFoundException
al buscar un evento con identificador de evento
no existente.

Valores de entrada:
- Identificador de un evento no existente.

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo

PR-IT-027

Unidad: EventServiceImpl

Método: findEvent

Motivación: Comprobar que se busca correctamente un evento
por su identificador.

Valores de entrada:

- eventName: Identificador de un evento existente.

Valores de salida:

- EventInfo

Inicialización:

- CategoryInfo
- EventInfo

PR-IT-028

Unidad: EventServiceImpl

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento segun
el startIndex y el account, devolviendo el correspondiente EventInfoBlock.

Valores de entrada:

- keywords: null
- categoryId: null
- eventsStarted: true
- startIndex: 0
- count: 4

Valores de salida:

- EventInfoBlock
  - events: Todos los EventInfo insertados hasta el momento ordenados por la fecha ascendentemente.
  - existsMoreBets: false.

Inicialización:

- CategoryInfo
- EventInfo

PR-IT-029

Unidad: EventServiceImpl

Método: findEvents

Motivación: Comprobar que se buscan adecuadamente los eventos insertados hasta el momento segun
el startIndex y el account, devolviendo el correspondiente EventInfoBlock.

Valores de entrada:

- keywords: null
- categoryId: null

- eventsStarted: true
- startIndex: 0
- count: 2

Valores de salida:
- EventInfoBlock
  - events: Los dos primeros eventos insertados hasta el momento ordenados por la fecha ascendentemente.
  - existsMoreBets: true.

Inicialización:
- CategoryInfo
- EventInfo

PR-IT-030

Unidad: EventServiceImpl

Método: findEvents

Motivación: Obtener un StartIndexOrCountException al pasarle al metodo los parametros startIndex o count negativos.

Valores de entrada:
- keywords: null
- categoryId: null
- eventsStarted: true
- startIndex: -1
- count: -1

Valores de salida:
- StartIndexOrCountException

Inicialización:
- CategoryInfo
- 4 EventInfo.

PR-IT-031

Unidad: EventServiceImpl

Método: addBetType

Motivación: Comprobar que se inserta adecuadamente una BetType en el sistema con sus BetTypeOption correspondientes asociada a un EventInfo.

Valores de entrada:
- eventId: Identificador de un evento existente
- type: BetType
- options: Lista de BetTypeOption

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOptions

PR-IT-032

Unidad: EventServiceImpl

Método: addBetType

Motivación: Obtener excepción InstanceNotFoundException
al intentar crear añadir un BetType con un identificador
de evento no existente.

Valores de entrada:
- eventId: Identificador de un evento no existente
- type: BetType
- options: Lista de BetTypeOption

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOptions

PR-IT-033

Unidad: EventServiceImpl

Método: addBetType

Motivación: Segun la especificación un BetType no puede
no tener la lista de BetTypeOption vacia. Por lo tanto
deberia devolver una excepcion NoAssignedTypeOptionsException

Valores de entrada:
- eventId: Identificador de un evento existente
- type: BetType
- options: Lista de BetTypeOption vacia o null

Valores de salida:
- NoAssignedTypeOptionsException

Inicialización:
- CategoryInfo
- EventInfo

PR-IT-034

Unidad: EventServiceImpl

Método: addBetType

Motivación: Segun la especificación un BetType no puede
no tener la lista de BetTypeOption con el mismo nombre.
Por lo tanto deberia devolver una excepcion DuplicatedResultTypeOptionsException.

Valores de entrada:
- eventId: Identificador de un evento existente
- type: BetType
- options: Lista de BetTypeOption con al menos
dos TypeOptions con los result con el mismo
valor.

Valores de salida:
- DuplicatedResultTypeOptionsException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOptions con al menos 2 TypeOptions con el mismo nombre

PR-IT-035

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que en el caso de pasarle un betType con identificador no existente
el metodo devuelve la excepcion correspondiente.

Valores de entrada:
- optionIds: Identificadores de las opciones ganadoras
- betTypeId: Identificador no existente del tipo de apuesta

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo
- BetType

- TypeOption

PR-IT-036

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que si no se pasaron identificadores de opciones a null se pusieron todas las opciones del tipo de apuesta a falso y el atributo pickedWinners a true.

Valores de entrada:
- optionIds: null
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-IT-037

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que si no se pasaron identificadores de opciones como una lista vacia se pusieron todas las opciones del tipo de apuesta a falso y el atributo pickedWinners a true.

Valores de entrada:
- optionIds: lista vacia
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-IT-038

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que si no se pasaron identificadores de opciones no existentes se devuelve la excepcion InstanceNotFoundException

Valores de entrada:
- optionIds: identificadores de apuestas(al menos uno) no existente
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- InstanceNotFoundException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-IT-039

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que el metodo devuelve TypeNotMultipleException si la lista de opciones ganadores contiene mas de una opcion y el tipo de apuesta no admite multiples.

Valores de entrada:
- optionIds: Lista con dos opciones existentes
- betTypeId: Identificador del tipo de apuesta que no admite multiples apuestas

Valores de salida:
- TypeNotMultipleException

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-IT-040

Unidad: EventServiceImpl

Método: pickWinners

Motivación: Comprobar que se ponen las opciones ganadoras correctamente.

Valores de entrada:

- optionIds: Opcion ganadora
- betTypeId: Identificador del tipo de apuesta

Valores de salida:
- NA

Inicialización:
- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-IT-041

Unidad: EventServiceImpl

Método: findAllCategories

Motivación: Obtener la lista de categorias insertadas hasta el momento.

Valores de entrada:
- NA

Valores de salida:
- List<CategoryInfo>: Resultados obtenidos.

Inicialización:
- CategoryInfo

<u>Pruebas de selección de datos aleatoria</u>

PR-RANDOM-IT-001

Unidad: BetServiceImpl

Método: createBet

Motivación: Comprobar que la apuesta se crea correctamente con cualquier cantidad positiva de dinero.

Valores de entrada:
- userId: identificador de un usuario existente
- typeOptionId: identificador de una opción existente
- amount: cantidad positiva aleatoria de dinero

Valores de salida:
- BetInfo

Inicialización:
- UserProfile

- CategoryInfo
- EventInfo
- BetType
- TypeOption

PR-RANDOM-IT-002

Unidad: BetServiceImpl

Método: createBet

Motivación: Comprobar que la apuesta lanza NegativeAmountException al crear una apuesta con cualquier cantidad negativa.

Valores de entrada:
- userId: identificador de un usuario existente
- typeOptionId: identificador de una opción existente
- amount: cantidad negativa aleatoria de dinero

Valores de salida:
- BetInfo

Inicialización:
- UserProfile
- CategoryInfo
- EventInfo
- BetType
- TypeOption

- **Pruebas de sistema:**

Consideramos las pruebas de sistemas las mismas que las de integración al solo testear el modelo.

- **Pruebas de aceptación:**

Las pruebas de aceptación las realizaron los profesores de Programación Avanzada evaluando que la aplicación cumple la especificación dada por ellos mismos. La implementación se ha modificado para solucionar errores, pero no se han añadido funcionalidades nuevas ni modificado las existentes, por lo que podemos decir que las pruebas de aceptación ya se habían realizado y con éxito antes de mejorar la calidad de los test.

Tipos de pruebas realizadas

- **Análisis estática de caja blanca**

Para el análisis de caja blanca utilizamos la herramienta checkstyle. Los componentes evaluados fueron todos excepto los de la capa web y los criterios que se utilizaron fueron los estándar de la herramienta.

- **Mutación de código**

Herramienta utilizada: PITEST

Para realizar las pruebas de mutación hemos utilizado la herramienta PITEST. Se han utilizado los operadores de mutación por defecto. En la siguiente imagen, se puede observar el estado de la aplicación tras ejecutar los tests de mutación por primera vez:

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
15	89%	262/293	80%	101/126

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
es.udc.pa.pa015.practicapa.model.betinfo	2	70%	19/27	67%	4/6
es.udc.pa.pa015.practicapa.model.betservice	2	100%	22/22	92%	11/12
es.udc.pa.pa015.practicapa.model.bettype	1	94%	30/32	75%	6/8
es.udc.pa.pa015.practicapa.model.categoryinfo	2	88%	14/16	75%	3/4
es.udc.pa.pa015.practicapa.model.eventService	3	90%	63/70	80%	37/46
es.udc.pa.pa015.practicapa.model.eventinfo	2	95%	56/59	79%	27/34
es.udc.pa.pa015.practicapa.model.typeoption	1	89%	25/28	71%	5/7
es.udc.pa.pa015.practicapa.model.userprofile	2	85%	33/39	89%	8/9

Report generated by PIT 1.1.11

Como se puede observar, aunque los porcentajes no están nada mal, hemos tratado de subirlos lo máximo posible. Tras eliminar los métodos toString() y crear varios tests, el resultado final ha sido:

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
15	94%	256/273	87%	103/119

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
es.udc.pa.pa015.practicapa.model.betinfo	2	70%	19/27	100%	6/6
es.udc.pa.pa015.practicapa.model.betservice	2	100%	18/18	100%	10/10
es.udc.pa.pa015.practicapa.model.bettype	1	100%	30/30	86%	6/7
es.udc.pa.pa015.practicapa.model.categoryinfo	2	100%	14/14	100%	3/3
es.udc.pa.pa015.practicapa.model.eventService	3	90%	64/71	83%	38/46
es.udc.pa.pa015.practicapa.model.eventinfo	2	100%	54/54	82%	27/33
es.udc.pa.pa015.practicapa.model.typeoption	1	100%	25/25	83%	5/6
es.udc.pa.pa015.practicapa.model.userprofile	2	94%	32/34	100%	8/8

Report generated by PIT 1.1.11

La cobertura de línea ha pasado de un 89% a un 94% y la cobertura de mutación de un 80% a un 87%, lo que es un cambio bastante significativo.

Los mutantes que quedan vivos en las entidades se debe a un método empleado por Hibernate únicamente para gestionar la bidireccionalidad de las relaciones, en el cual se ejecutan dos llamadas (cada una a un setter de dos entidades distintas), por lo cual no se han creado tests para probarla. El mutante sobrevive tras eliminar una de esas dos llamadas.

Los mutantes restantes (en el DAO de EventInfo y en eventService) se corresponden con mutaciones en condiciones, los cuales no nos ha dado tiempo a corregir.

- ## Pruebas de estrés

Como se explicará con más detalle en el apartado 4, sólo se han realizado pruebas contra el servidor para medir los tiempos de respuesta y el porcentaje de peticiones que son procesadas y devueltas al cliente y cuáles no.

Herramienta utilizada: JMeter
Servidor usado(producción): Jetty
Versión de la aplicación a la que aplicaron estas pruebas:
000a9687f44d3d31f83f4d9bd98dc979685deddb

Datos de la máquina donde se ejecuta el servidor:

Nombre del modelo: MacBook Pro
Identificador del modelo: MacBookPro11,1
Nombre del procesador: Intel Core i5
Velocidad del procesador: 2,6 GHz
Cantidad de procesadores: 1
Cantidad total de núcleos: 2
Caché de nivel 2 (por núcleo): 256 KB
Caché de nivel 3: 3 MB
Memoria: 8 GB

Criterios de aceptación:

Hemos asumido dadas las características de la máquina y el nivel de carga de la petición que un tiempo de respuesta razonable en este contexto es de **2000 ms(2 seg)**. Esto es una estimación completamente subjetiva debido a que una aplicación real de esta tipología se ejecuta en varios servidores y no con el mismo hardware.

Plan de diseño del test

Petición HTTP

http://127.0.0.1:9090//betting-app/search/events/$N/-1/0

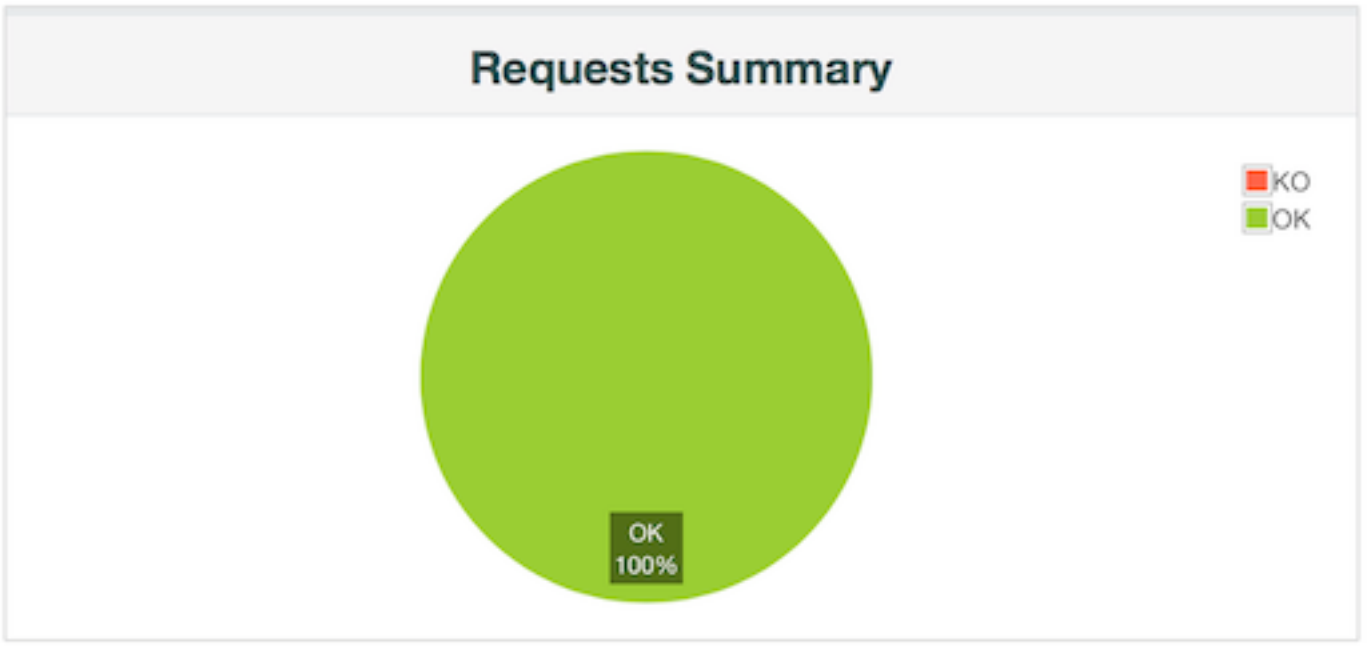
La petición devuelve entre 0 y 10(paginación web) eventos almacenados en base de datos. En nuestro caso debido a los datos ya insertados en base de datos todas las peticiones devolverán 10 eventos. La petición corresponde con

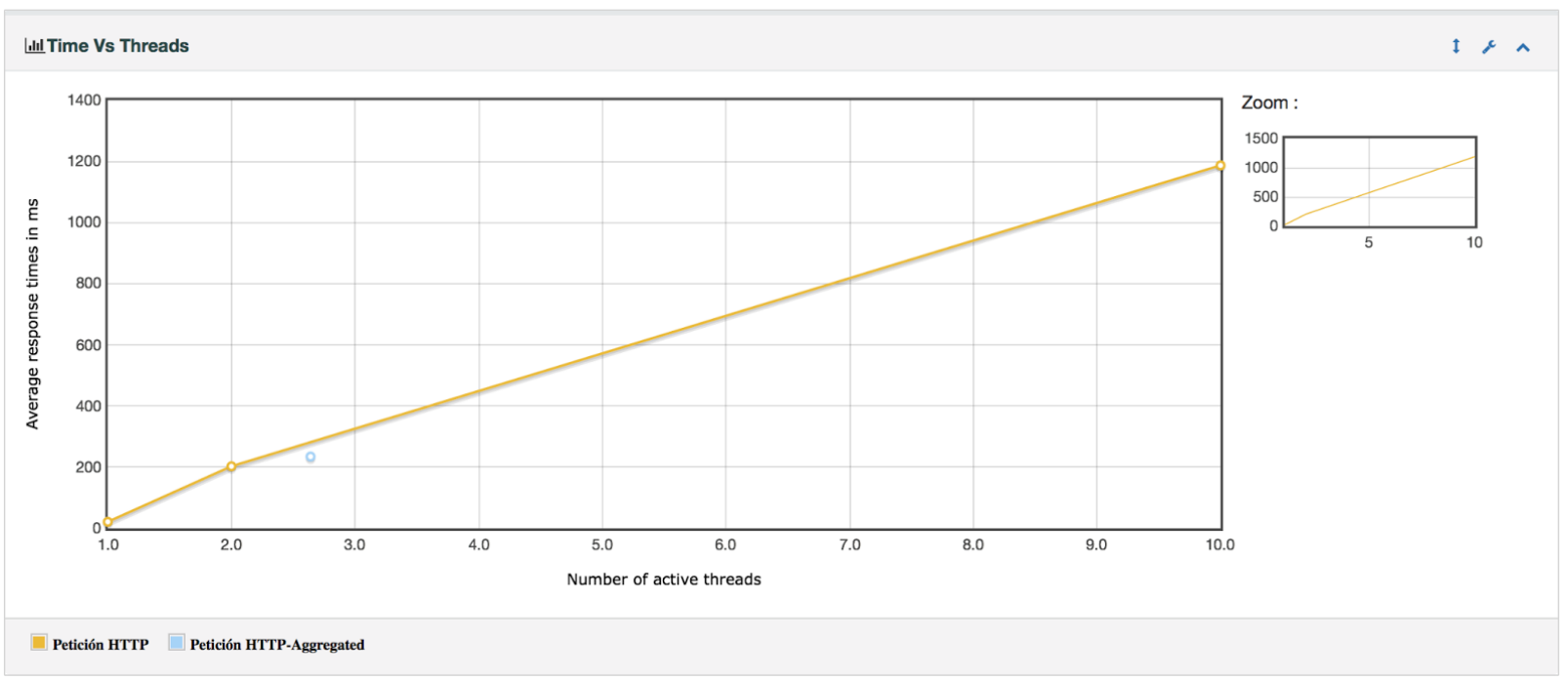
Propiedades de hilo

- **Prueba de** 50 **hilos**

  Número de hilos: 50

  Periodo de aceleración: 10 (segundos)



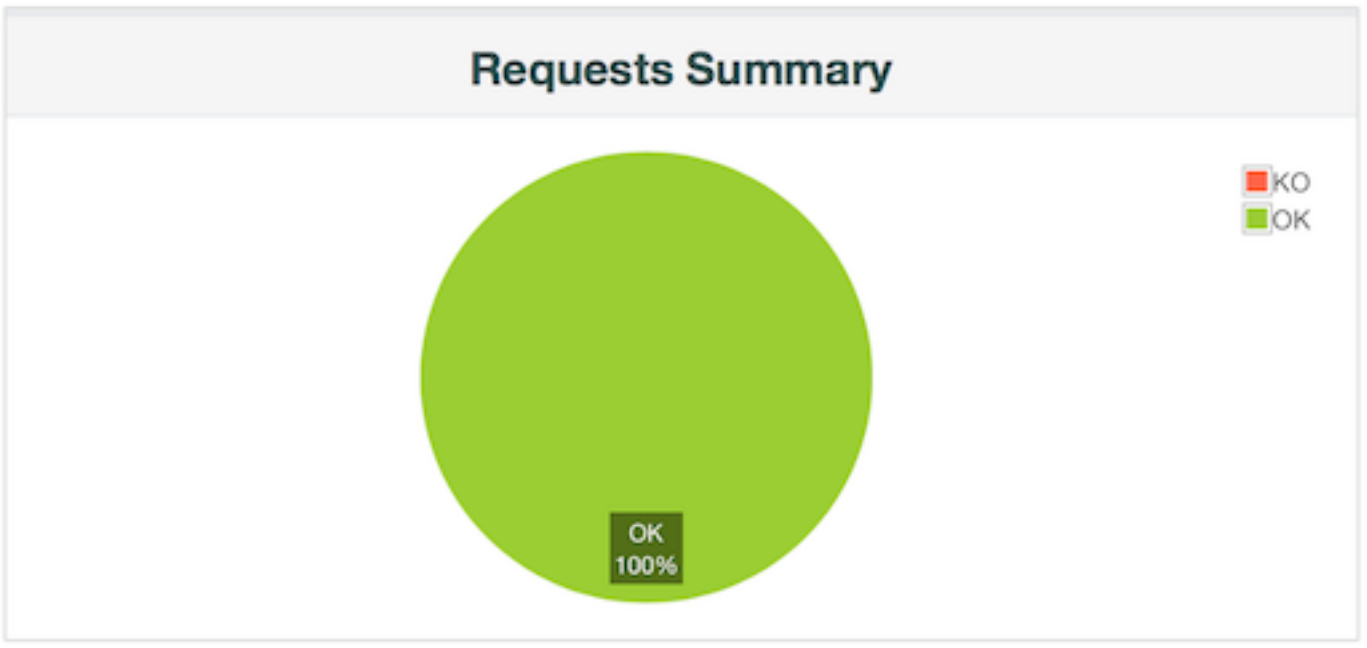


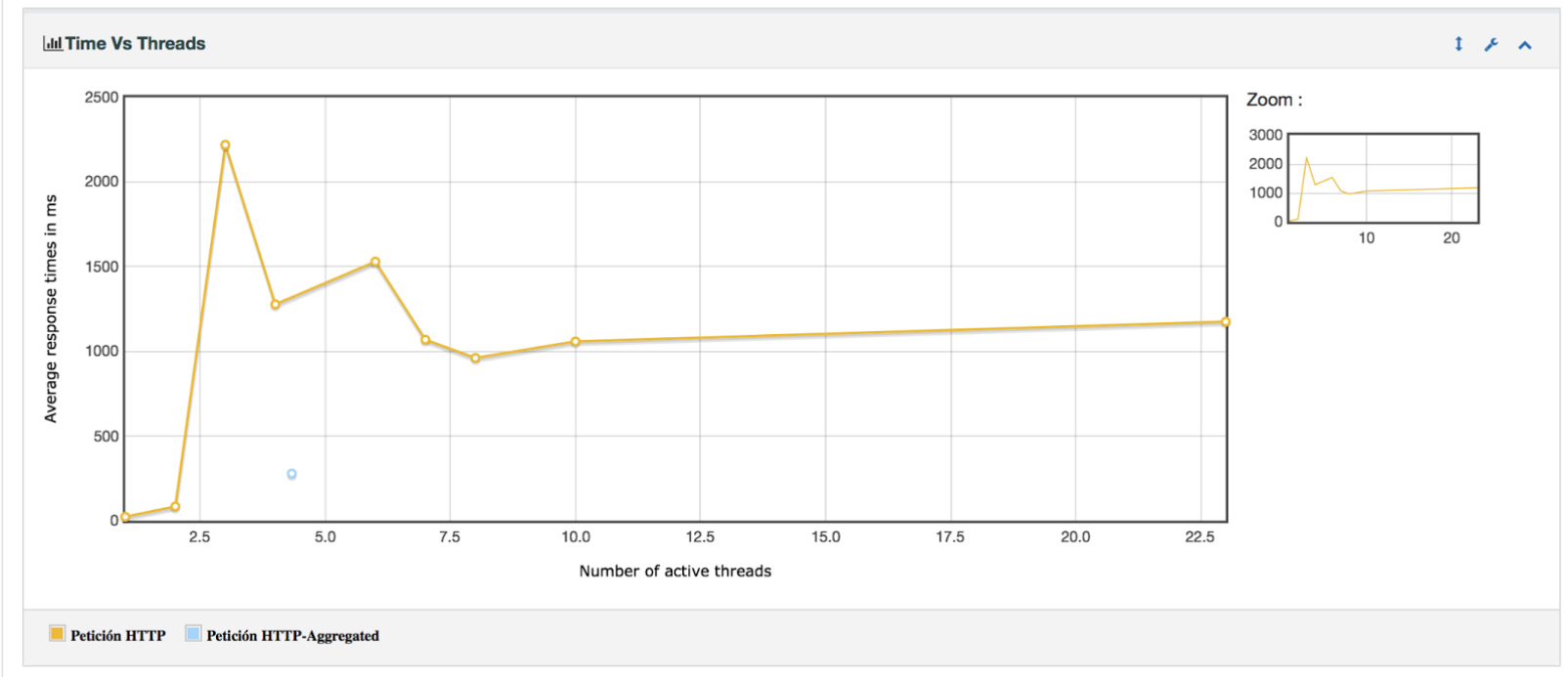
Como vemos, el servidor con 50 threads es capaz de atender el 100% de las peticiones. Ell tiempo de respuesta es estable y más o menos constante. Además el promedio de tiempo de respuesta está bastante lejos de nuestro criterio de aceptación.

- **Prueba de** 100 **hilos**

  Número de hilos: 100

  Periodo de aceleración: 10 (segundos)



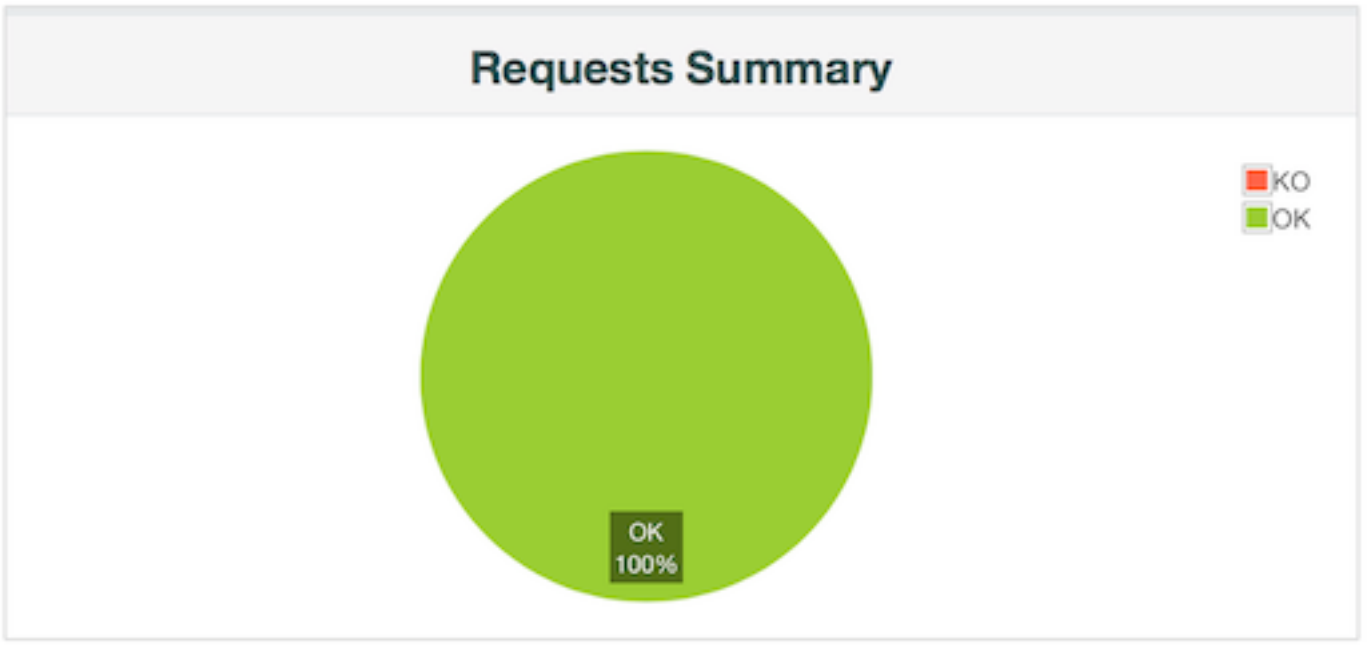


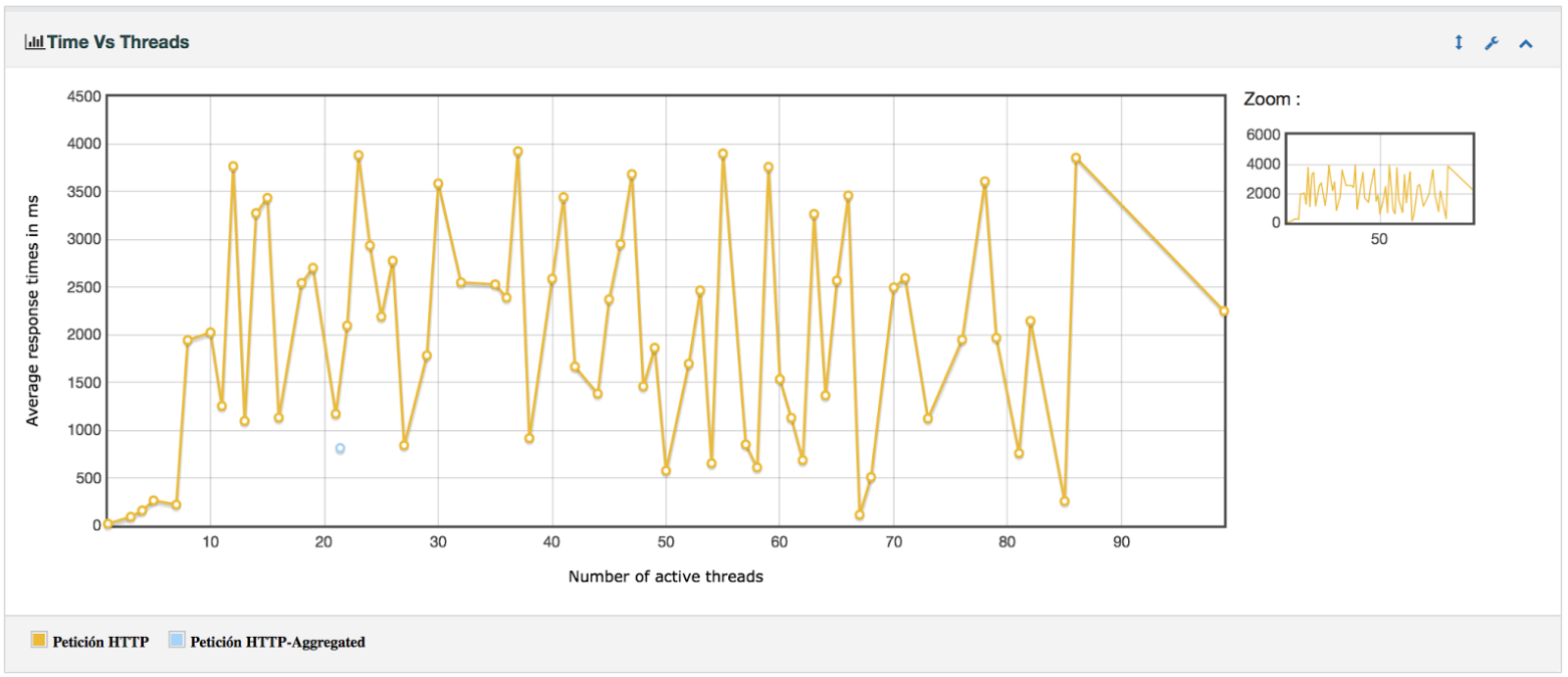
Con 100 threads y un periodo de aceleración de 10 segundos el servidor también es capaz de atender el 100% de las peticiones. El tiempo de respuesta sufre más picos, por lo que es menos estable. Aunque haya un pico que supere los 2 segundos, podemos decir que el servidor cumple los criterios de aceptación.

- **Prueba de** 250 **hilos**

  Número de hilos: 250

  Periodo de aceleración: 10 (segundos)



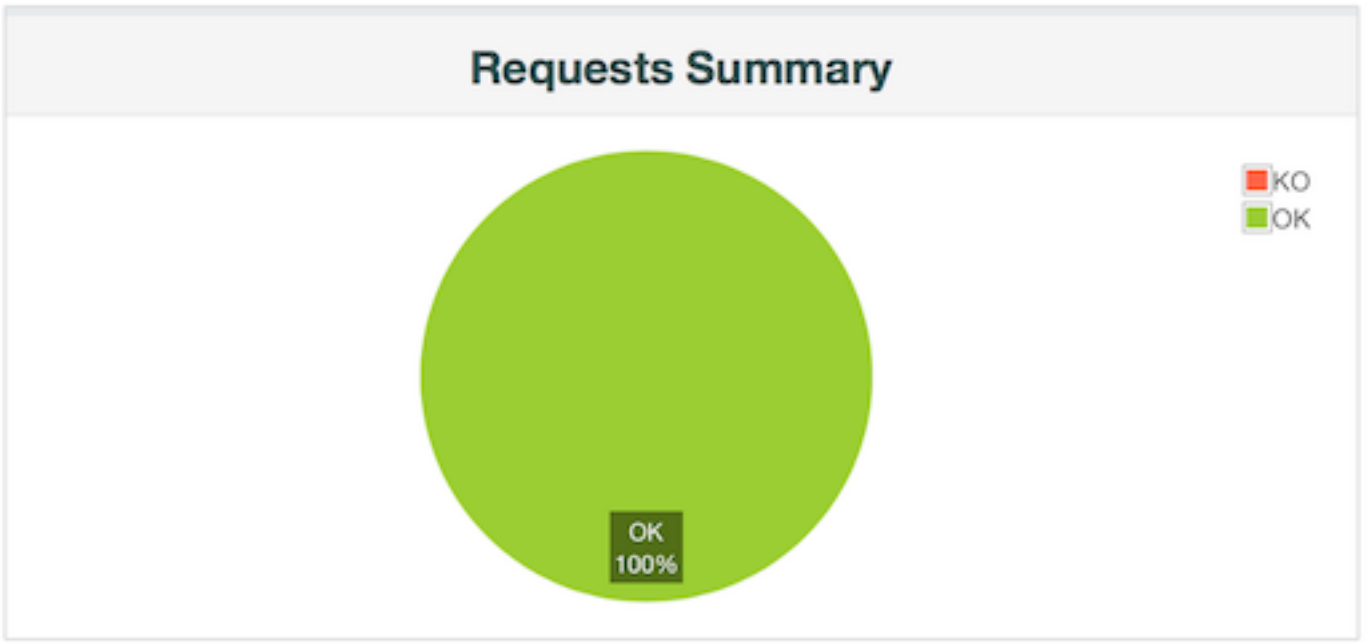


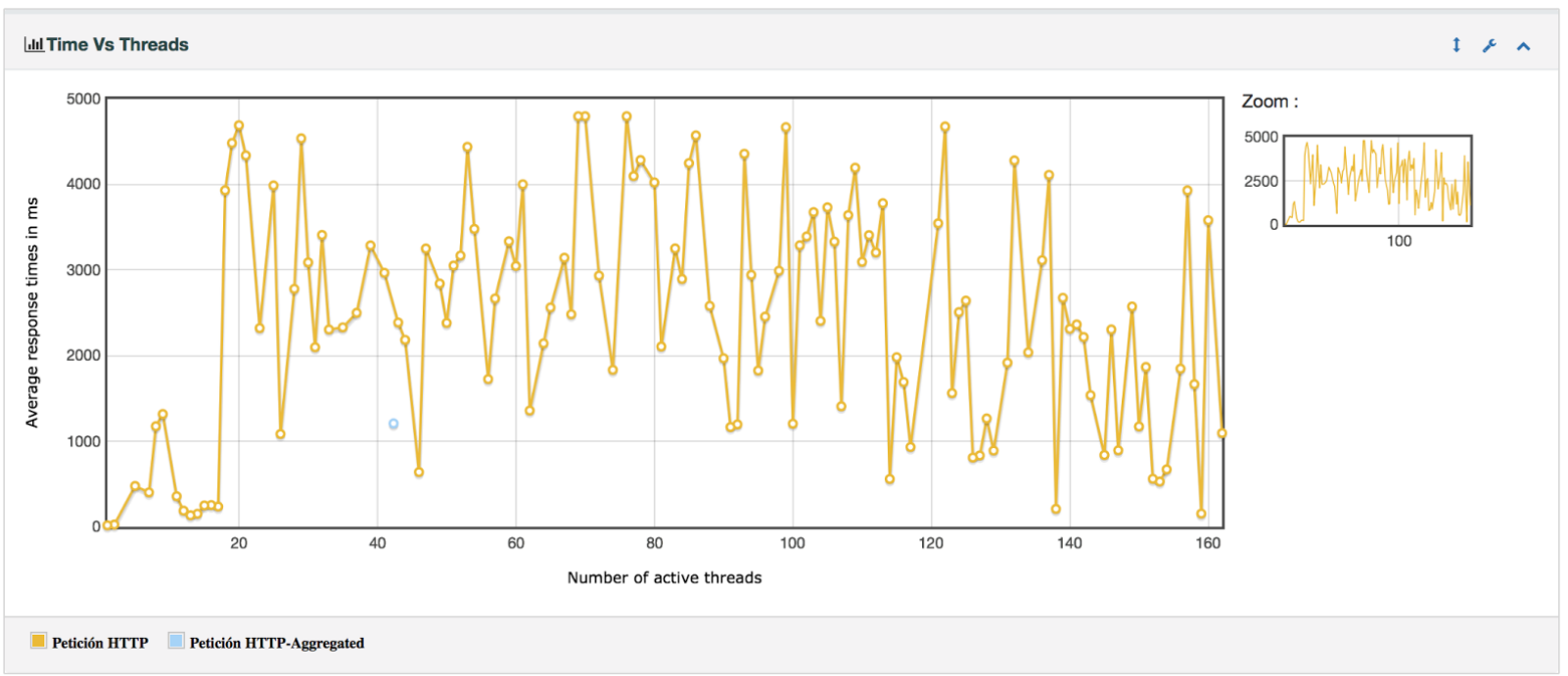
Con 250 threads y un periodo de aceleración de 10 segundos el servidor también es capaz de atender el 100% de las peticiones, pero en este caso, el los tiempos de respuesta no cumplen los criterios de aceptación en muchos picos. Aún así, el servidor atiende todas las peticiones, y los tiempos de respuestas más tardíos no superan el doble del criterio de aceptación.

- **Prueba de** 350 **hilos**

  Número de hilos: 350

  Periodo de aceleración: 10 (segundos)



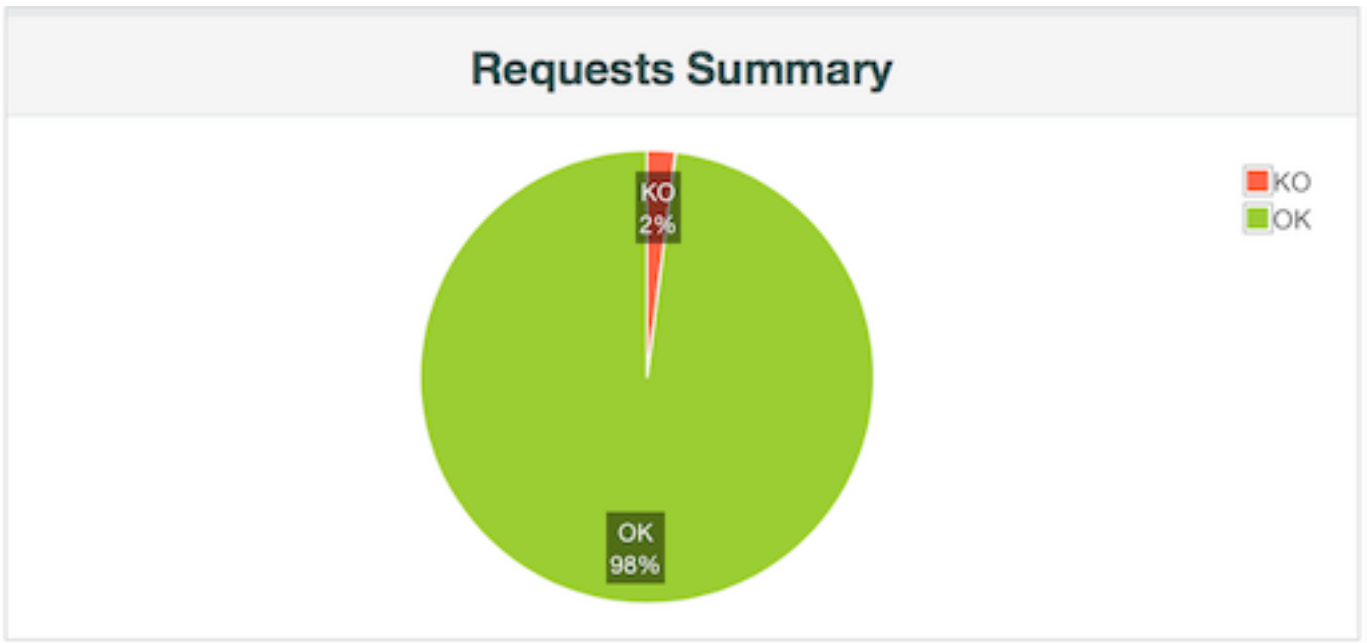


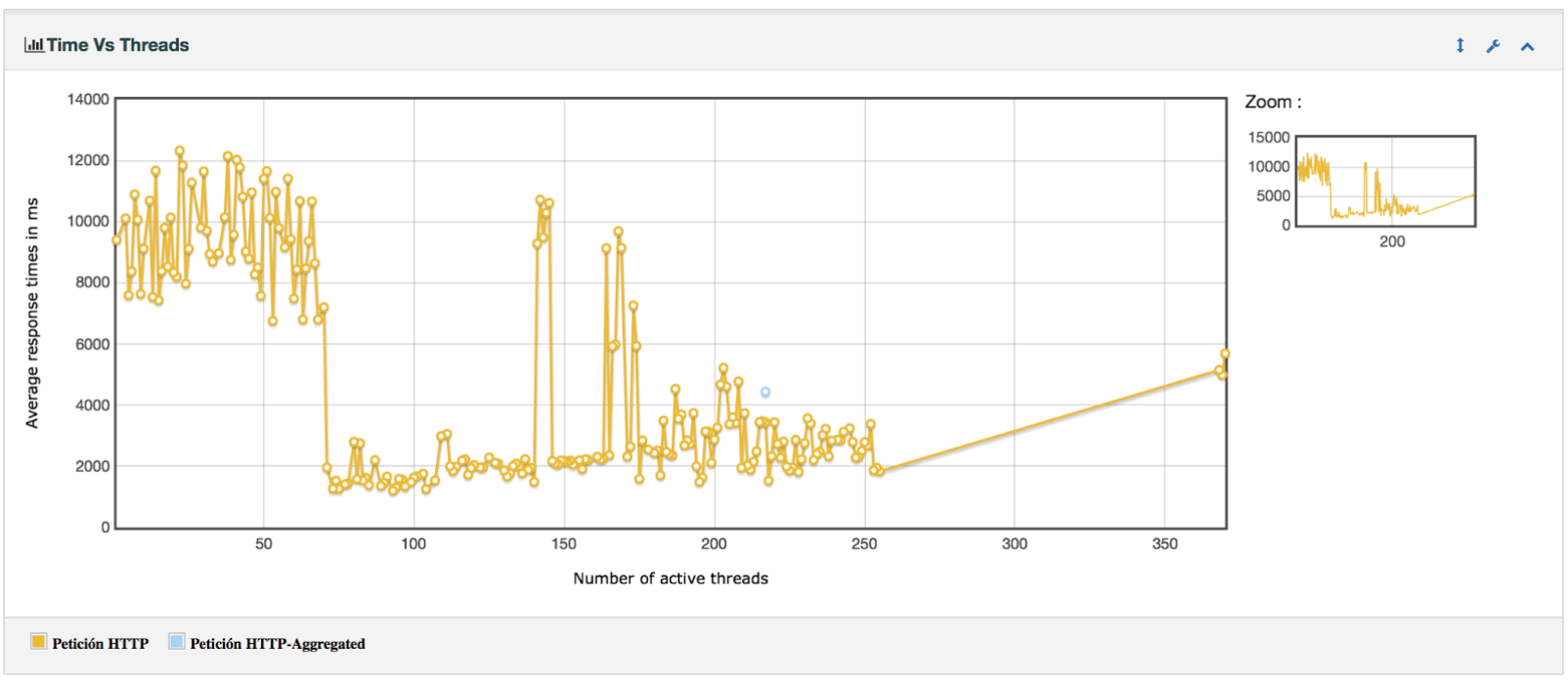
El servidor sigue atendiendo el 100% de las peticiones con 350 threadsy un periodo de aceleración de 10 segundos, pero la mayoría de los picos que se producen superan los criterios de aceptación.

- **Prueba de** 500 **hilos**

  Número de hilos: 500

  Periodo de aceleración: 1 (segundos)



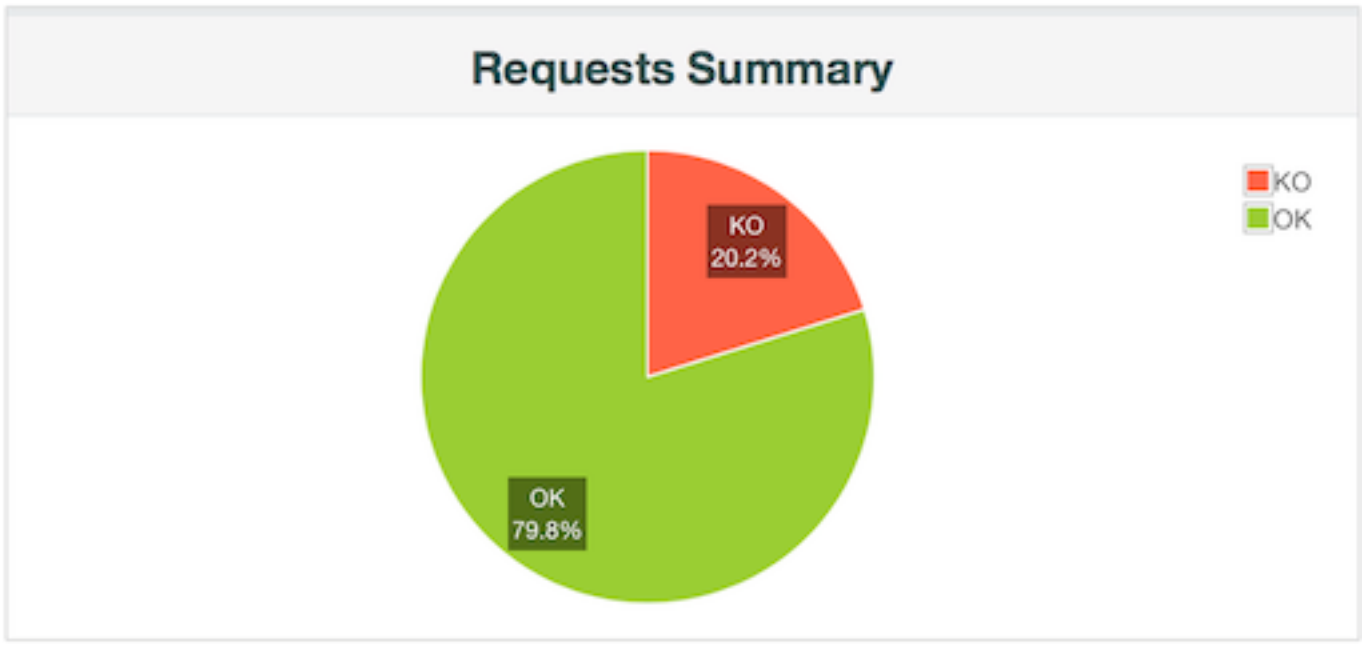


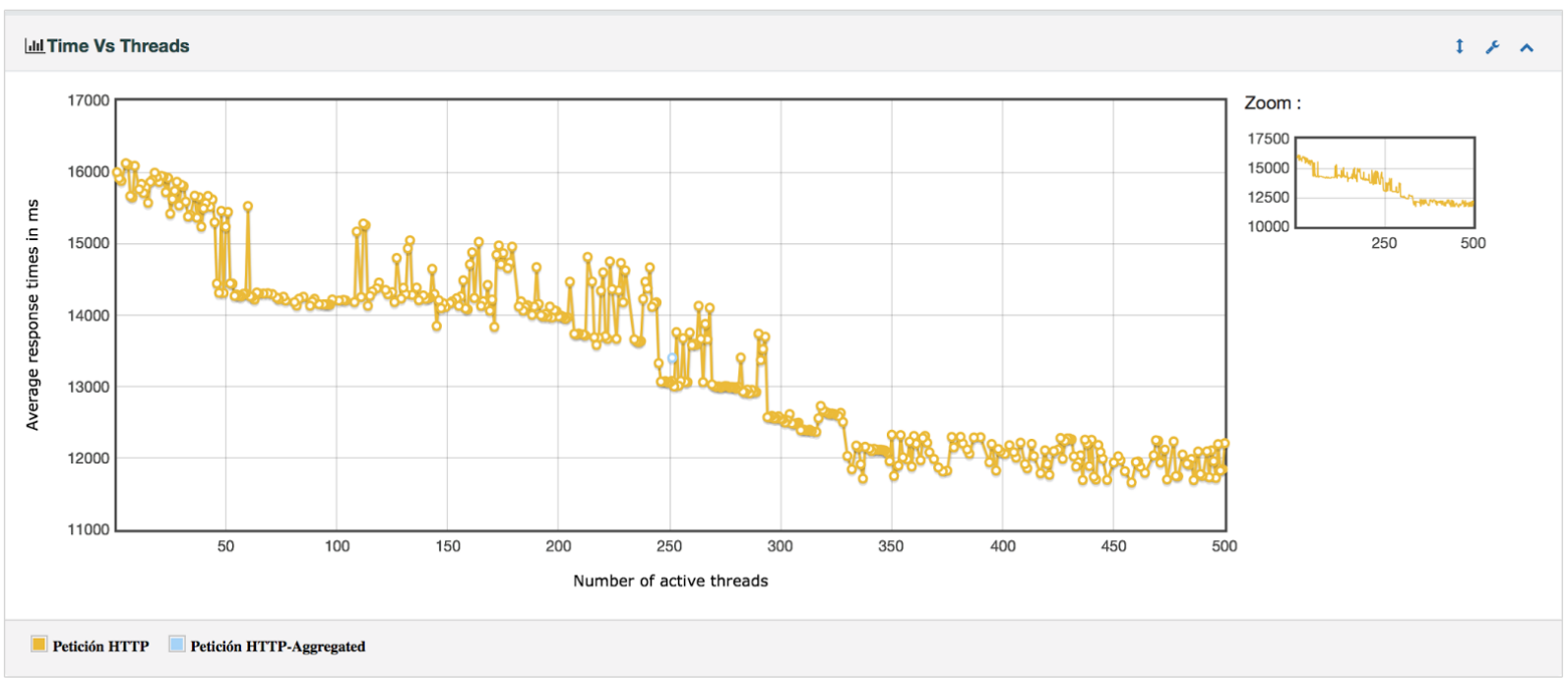
Ya que todas las peticiones siempre eran atendidas, decidimos reducir el periodo de aceleración a 1 segundos y efectivamente el servidor no da respuesta al 2% de las peticiones con 500 threads. Los tiempos de respuesta sufren picos muy distantes. Solo muy pocos bajos los criterios de aceptación.

- **Prueba de** 750 **hilos**

  Número de hilos: 750

  Periodo de aceleración: 10 (segundos)





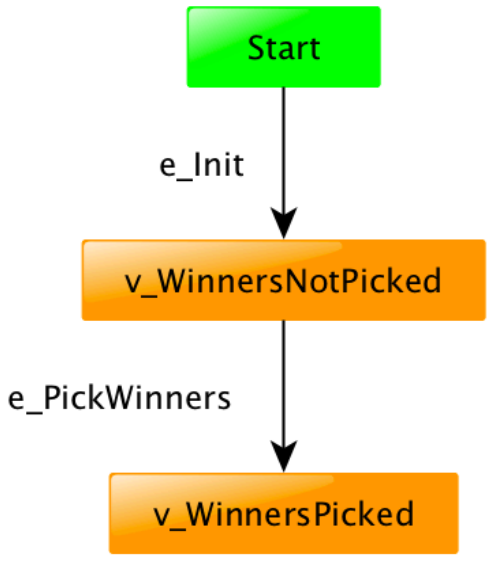
Con 750 threads dejamos de atender el 20% de las peticiones y los tiempos de respuestas se alejan demasiado a lo aceptable. No tiene sentido seguir haciendo pruebas con más threads.

Conclusión de las pruebas de estrés: Bajo los criterios de aceptación, podemos decir que corriendo el servidor en la máquina especificada seríamos capaces de garantizar unos tiempos de respuestas aceptables con las cotas de 250 peticiones repartidas(de manera más o menos uniforme) cada 10 segundos.

- **Pruebas basadas en modelos**

Herramienta utilizada: GRAPHWALKER

Para realizar las pruebas basadas en modelos, hemos intentado utilizar la herramienta GraphWalker. Hemos intentando modelar el estado de un tipo de opción (TypeOption), el cual puede tener los ganadores elegidos o no. Para ello, hemos diseñado el siguiente grafo en yEd:



Para diseñar el grafo, se han seguido las pautas indicadas en la página web de la herramienta. También se ha comprobado con la CLI de la herramienta que el grafo no contuviese errores.

Una vez acabado, se ha generado la interfaz Java correspondiente al grafo a través del goal generate-sources de Maven. Posteriormente, se ha procedido a la implementación de la interfaz según las pautas indicadas en la página web de la herramienta también y se han creado los tests de JUnit correspondientes. Se han modificado los métodos que aparecen en la web (deprecated) por los más recientes.

Finalmente, hemos tenido que abortar la ejecución de esta prueba comentando los tests creados, ya que el contexto de Spring parece no ser compatible con el contexto utilizado por esta herramienta.

- **Cobertura**

Herramienta utilizada: Cobertura

El uso de esta herramienta nos ha sido muy útil tanto para mejorar la calidad de los test como para ir viendo la progresión y evolución de los mismos a lo largo de toda la práctica.

Cobertura inicial:

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	89	53% 715/1346	33% 72/213	1,516
es.udc.pa.pa015.practicapa.model.betinfo	4	64% 22/34	N/A N/A	1
es.udc.pa.pa015.practicapa.model.betservice	6	61% 16/26	50% 2/4	1,091
es.udc.pa.pa015.practicapa.model.bettype	4	90% 27/30	N/A N/A	1
es.udc.pa.pa015.practicapa.model.categoryinfo	4	58% 10/17	N/A N/A	1
es.udc.pa.pa015.practicapa.model.eventService	10	66% 30/45	100% 8/8	1,267
es.udc.pa.pa015.practicapa.model.eventinfo	4	91% 51/56	82% 28/34	2,062
es.udc.pa.pa015.practicapa.model.typeoption	4	100% 29/29	N/A N/A	1
es.udc.pa.pa015.practicapa.model.userprofile	4	82% 33/40	100% 2/2	1,118
es.udc.pa.pa015.practicapa.model.userservice	5	95% 45/47	58% 7/12	1,562
es.udc.pa.pa015.practicapa.model.userservice.util	3	97% 452/462	89% 25/28	1,933
es.udc.pa.pa015.practicapa.model.util	2	0% 0/1	N/A N/A	1
es.udc.pa.pa015.practicapa.web.components	2	0% 0/7	0% 0/2	2
es.udc.pa.pa015.practicapa.web.pages	3	0% 0/2	N/A N/A	0
es.udc.pa.pa015.practicapa.web.pages.admin	7	0% 0/134	0% 0/36	2
es.udc.pa.pa015.practicapa.web.pages.search	6	0% 0/89	0% 0/18	1,452
es.udc.pa.pa015.practicapa.web.pages.user	8	0% 0/117	0% 0/24	2,75
es.udc.pa.pa015.practicapa.web.services	10	0% 0/189	0% 0/45	2,476
es.udc.pa.pa015.practicapa.web.util	3	0% 0/21	N/A N/A	1

Esta cobertura corresponde a los test que ya venían implementados. La cobertura es “aceptable” en algunos paquetes, los que contienen las entidades y los DAOs, pero los paquetes como *.betservice o *.eventservice que son los que contienen la implementación de la lógica de negocio o dicho de otra forma, el servicio de la aplicación, tienen una cobertura del 61% y 66% respectivamente.

Cobertura test de unidad + test de integración:

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	68	53% 704/1314	37% 97/261	1,637
es.udc.pa.pa015.practicapa.model.betinfo	3	63% 19/30	N/A N/A	1
es.udc.pa.pa015.practicapa.model.betservice	4	95% 23/24	100% 4/4	1,111
es.udc.pa.pa015.practicapa.model.bettype	3	93% 31/33	N/A N/A	1
es.udc.pa.pa015.practicapa.model.categoryinfo	3	87% 14/16	N/A N/A	1
es.udc.pa.pa015.practicapa.model.eventService	8	90% 70/77	92% 35/38	2,087
es.udc.pa.pa015.practicapa.model.eventinfo	3	89% 50/56	70% 24/34	2,062
es.udc.pa.pa015.practicapa.model.typeoption	3	89% 26/29	N/A N/A	1
es.udc.pa.pa015.practicapa.model.userprofile	3	84% 33/39	100% 2/2	1,118
es.udc.pa.pa015.practicapa.model.userservice	4	95% 42/44	87% 7/8	1,562
es.udc.pa.pa015.practicapa.model.userservice.util	2	97% 396/405	89% 25/28	1,933
es.udc.pa.pa015.practicapa.model.util	1	0% 0/2	N/A N/A	1
es.udc.pa.pa015.practicapa.web.components	1	0% 0/7	0% 0/2	2
es.udc.pa.pa015.practicapa.web.pages	2	0% 0/2	N/A N/A	0
es.udc.pa.pa015.practicapa.web.pages.admin	6	0% 0/140	0% 0/40	1,829
es.udc.pa.pa015.practicapa.web.pages.search	5	0% 0/87	0% 0/20	1,484
es.udc.pa.pa015.practicapa.web.pages.user	7	0% 0/109	0% 0/24	1,852
es.udc.pa.pa015.practicapa.web.services	8	0% 0/193	0% 0/61	2,905
es.udc.pa.pa015.practicapa.web.util	2	0% 0/21	N/A N/A	1

Report generated by Cobertura 2.1.1 on 18/12/16 19:25.

Esta cobertura corresponde con los test de unidad más los test de integración implementados, sin haber usado ninguna técnica específica ni herramienta, simplemente implementando lo test que habíamos diseñado. Como se puede apreciar, se ha mejorado bastante la cobertura en los paquetes de los servicios y en general. Las coberturas de los paquetes que han bajado se debe a que se ha añadido código y no se ha testeado.
Debilidades importantes:

- Line Coverage del paquete *.betinfo

- Branch Coverage del paquete *.eventinfo

Cobertura después de mejorar la cobertura:

Issue #15: Creada para mejorar la cobertura.
Inspeccionando con la herramienta qué código se nos ha olvidado testear, nos dimos cuenta que la cobertura baja en este paquete se debe a la clase BetInfo.java donde ocurren dos cosas:

-El método toString()([se ha sobrescrito] no se usa en ningún caso, por lo tanto se puede borrar.

-Los setter no se quitan para aumentar la cobertura porque Hibernate los usa para gestionar la sincronización de datos entre memoria y BBDD. Como crear métodos de testing para probar los setters de una entidad nos parece ridículo, hemos determinado esa cobertura como aceptable.

Otro paquete con cobertura baja era el branch coverage de *.eventinfo. Para aumentar su cobertura se ha implementado el caso de prueba PR-UN-058.

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	88	44% 796/1796	39% 200/504	1,639
es.udc.pa.pa015.practicapa.model.betinfo	4	66% 30/45	N/A N/A	1
es.udc.pa.pa015.practicapa.model.betservice	5	94% 33/35	100% 8/8	1,125
es.udc.pa.pa015.practicapa.model.bettype	4	95% 59/62	N/A N/A	1
es.udc.pa.pa015.practicapa.model.categoryinfo	4	92% 26/28	N/A N/A	1
es.udc.pa.pa015.practicapa.model.eventService	10	91% 132/145	92% 70/76	2,042
es.udc.pa.pa015.practicapa.model.eventinfo	4	95% 94/98	90% 54/60	2,062
es.udc.pa.pa015.practicapa.model.typeoption	4	94% 47/50	N/A N/A	1
es.udc.pa.pa015.practicapa.model.userprofile	4	92% 59/64	100% 4/4	1,118
es.udc.pa.pa015.practicapa.model.userservice	5	96% 77/80	87% 14/16	1,562
es.udc.pa.pa015.practicapa.model.userservice.util	3	93% 239/255	89% 50/56	1,933
es.udc.pa.pa015.practicapa.model.util	2	0% 0/4	N/A N/A	1
es.udc.pa.pa015.practicapa.web.components	2	0% 0/13	0% 0/4	2
es.udc.pa.pa015.practicapa.web.pages	3	0% 0/4	N/A N/A	0
es.udc.pa.pa015.practicapa.web.pages.admin	7	0% 0/253	0% 0/78	1,829
es.udc.pa.pa015.practicapa.web.pages.search	6	0% 0/144	0% 0/38	1,484
es.udc.pa.pa015.practicapa.web.pages.user	8	0% 0/196	0% 0/48	1,852
es.udc.pa.pa015.practicapa.web.services	10	0% 0/285	0% 0/116	2,905
es.udc.pa.pa015.practicapa.web.util	3	0% 0/35	N/A N/A	1

Los resultados después de esta issue son estos. Como se ve, BetInfo sigue teniendo una cobertura no dentro de los rangos de aceptabilidad por lo ya explicado. Y el paquete *.eventinfo ha pasado del 70% al 90%.

Cobertura final:

Después de usar todas las herramientas la cobertura resultante es esta.

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	89	46%	864/1874	39%	206/516	1,659
es.udc.pa.pa015.practicapa.model.betinfo	4	66%	30/45	N/A	N/A	1
es.udc.pa.pa015.practicapa.model.betservice	6	95%	42/44	100%	14/14	1,333
es.udc.pa.pa015.practicapa.model.bettype	4	96%	73/76	N/A	N/A	1
es.udc.pa.pa015.practicapa.model.categoryinfo	4	94%	32/34	N/A	N/A	1
es.udc.pa.pa015.practicapa.model.eventService	10	91%	133/145	92%	70/76	2,042
es.udc.pa.pa015.practicapa.model.eventinfo	4	97%	108/111	90%	54/60	2,133
es.udc.pa.pa015.practicapa.model.typeoption	4	95%	59/62	N/A	N/A	1
es.udc.pa.pa015.practicapa.model.userprofile	4	92%	71/77	100%	4/4	1,125
es.udc.pa.pa015.practicapa.model.userservice	5	96%	77/80	87%	14/16	1,562
es.udc.pa.pa015.practicapa.model.userservice.util	3	93%	239/255	89%	50/56	1,933
es.udc.pa.pa015.practicapa.model.util	2	0%	0/4	N/A	N/A	1
es.udc.pa.pa015.practicapa.web.components	2	0%	0/13	0%	0/4	2
es.udc.pa.pa015.practicapa.web.pages	3	0%	0/4	N/A	N/A	0
es.udc.pa.pa015.practicapa.web.pages.admin	7	0%	0/253	0%	0/78	1,829
es.udc.pa.pa015.practicapa.web.pages.search	6	0%	0/144	0%	0/38	1,484
es.udc.pa.pa015.practicapa.web.pages.user	8	0%	0/207	0%	0/54	1,889
es.udc.pa.pa015.practicapa.web.services	10	0%	0/285	0%	0/116	2,905
es.udc.pa.pa015.practicapa.web.util	3	0%	0/35	N/A	N/A	1

Explicacion de las coberturas por paquetes:

- *.betinfo: Ya explicado. La cobertura la reducen los setters ya que el codigo de los test no pasa directamente por ahí pero hacen falta para Hibernate, y testear unos setters no tiene sentido.
- *.betservice: No se testea el string que guarda la excepcion TypeNotMultipleException. Este mensaje sería visible en la capa web por eso no lo consideramos necesario testearlo. Lo mismo ocurre en el paquete *.userservice.
- *.bettype, *.categoryinfo, *.eventinfo, *.typeoption, *.userprofile: Lo mismo que en la clase BetInfo.java. El código de los test no pasa por algún setters de las entidades.
- *.eventservice: EventInfoBlock, el código no pasa por los setters. En cuanto a EventServiceImpl COBERTURA nos dice que el código no pasa por

```
20  8          */
21             @Transactional(readOnly = true)
22  8          public final EventInfoBlock findEvents(final String keywords, final Long categoryId, final boolean eventsStarted,
23  14                     final int startIndex, final int count) throws InstanceNotFoundException, StartIndexOrCountException {
24  2
25  6                  if (startIndex < 0 || count < 0)
26  26                         throw new StartIndexOrCountException();
27  16
28  16                 if (categoryId != null)
29  14                         categoryInfoDao.find(categoryId);
```

cuando está claro que si que pasa, ya que hay un test específico solo para ese caso de prueba. PR-UN-030

Ocurre lo mismo con

```
        if (!type.getIsMultiple() && optionIds.size() > 1) {
                throw new TypeNotMultipleException(type.getTypeId());
        }
```

PR-IT-039 implementado para pasar por ahí.

Esta es la justificación por la cual la cobertura no está al 100% en Cobertura, pero a efectos prácticos podemos decir que la cubrimos.

4. Registro de pruebas

El proceso de pruebas se ha ido desarrollando por iteraciones, por lo que el trabajo se concentró en la semana de la fecha de entrega de cada iteración y entre cada entrega el trabajo disminuyó. Las iteraciones se dividieron de la siguiente forma: la primera iteración para las pruebas de unidad, la segunda para las pruebas de iteración y la tercera para el uso de las herramientas en sus respectivas pruebas. Desde el comienzo de la práctica tenemos un fichero txt compartido en Dropbox donde fuimos documentando la división de las tareas, la numeración de las pruebas, el progreso de las mismas, etc … para llevar un control y registro del trabajo realizado. Cada integrante fue el responsable del diseño e implementación de los componentes asignados.
Este es el contenido actual del fichero:

PLANIFICACIÓN VVS
****
TEST DE UNIDAD:

CategoryInfoDaoUnitTest = 5(GenericDao) + 2 [Santi] = FINISH = [0-6]
EventInfoDaoUnitTest = 5 CPs [Santi] = FINISH [7-12]
UserProfileDaoUnitTest = 2 CPs [Santi] = FINISH [13-14]
BetInfoDaoUnitTest = 2 CPs [Lorena] = FINISH = [15-16]
BetTypeDaoUnitTest = 0 CPs
TypeOptionDaoUnitTest = 0 CPs
EventServiceUnitTest = 15 CPs [Santi] = FINISH = [17-31]
BetServiceUnitTest = 7 CPs [Elias] = FINISH = [32-37]
UserServiceUniTest = 4 CPs [Santi] = FINISH = [38-50]

EventServiceUnitTest - PickWinners = 7 CPS [Santi] = FINISH = [51-57]
Aumento de cobertura = 1 CPs = [Santi] = [58]
Mutacion = 2 CPs = [Elias] = [59-60]

****
TEST DE INTEGRACION:

BetServiceTest = IT[001-006] [Lorena]
UserServiceTest = IT[007-019] [Elias]
EventServiceTest IT[020-040] [Santi]
Elias - Mutacion = [41]

Problemas durante el proceso de prueba

-Problema con Tomcat y Jetty:

Al realizar las pruebas de estrés, y al ser Jetty nuestro servidor predeterminado para producción, nos dimos cuenta que herramientas de monitorización de procesos como VISUALVM o incluso el monitor de actividad del propio sistema operativo no diferenciaban a Jetty como un proceso. De esta manera, filtrar el consumo de memoria y CPU solo de Jetty resulta inviable. Como también disponíamos de Tomcat, se nos ocurrió ejecutar la aplicación desde el, ya que el SO sí que reconoce a Tomcat como un proceso diferenciado; el problema es que cuando se configuró la aplicación para que ejecutara sobre Tomcat, el “modelutil”(contiene el GenericDao) estaba importado directamente en el workspace como un proyecto maven independiente. Al comenzar esta práctica, tuvimos que integrar el modelutil en el proyecto, añadiendolo a sus librerías como un .jar. Creemos que este es el problema porque al ejecutarlo Tomcat da un error de configuración. Por falta de tiempo y hablando con la Laura decidimos testear solo los tiempos de respuesta con JMeter y la cantidad de respuestas procesadas por el servidor, pero no hicimos análisis de memoria y otros parecidos.

-Problema con la dependencia de graphwalker:

Se ha comentado en el pom.xml del proyecto la dependencia con graphwalker ya que daba error y no sabemos el motivo.

5. Registro de errores

Fuente(issues cerradas):
https://github.com/elias-garcia/vvs/issues?utf8=✓&q=is%3Aissue%20is%3Aclosed

Pruebas realizadas que revelaron incumprimentos de las especificacións:

(Los incumprimentos de las especificación que no tienen referenciados casos de prueba se detectaron por análisis de código)

- **Modificación de la función findEvent en EventInfoDao #4**
  - No se comprueba que el eventsStarted sea a true, por la tanto siempre que éste lo es lista todos los eventos.
  - Debe de listar solos los eventos empezados. e.eventDate < :eventDate
- **Fallo en la implementación en EventService - findEvents: No devuelve InstanceNotFoundException #5**
  - Caso de prueba relacionado: PR-UN-022
  - Inspeccionando el código se ha detectado que la implementación no devuelve InstanceNotFoundException es este método en el caso de que se le pase un identificador de categoría no existente. Modificar la implementación para que devuelva la excepción.
  - if (categoryId != null)
    categoryInfoDao.find(categoryId);
- **Falta la excepción NullEventNameException para EventServiceUnitTest - createEvent #6**
  - Caso de prueba relacionado: PR-UN-018
  - Inspeccionando el código se ha detectado que la implementación no devuelve NullEventNameException de hecho ni siquiera está creada. Crearla y modificar la implementación para que devuelva la excepción.
- **EventServiceImpl - findEvents no devuelve StartIndexOrCountException #8**
  - Caso de prueba relacionado: PR-UN-025
  - Inspeccionando el código se ha detectado que la implementación no devuelve StartIndexOrCountException es este método para este caso. Crearla y modificar la implementación para que devuelva la excepción.
- **EventServiceImpl - addBetType no devuelve NoAssignedTypeOptionsException si las TypeOptions están vacías. #9**
  - Caso de prueba relacionado: PR-UN-029

    - Inspeccionando el código se ha detectado que la implementacion no devuelveNoAssignedTypeOptionsException es este metodo para este caso. Crearla y modificar la implementación para que devuelva la excepción.

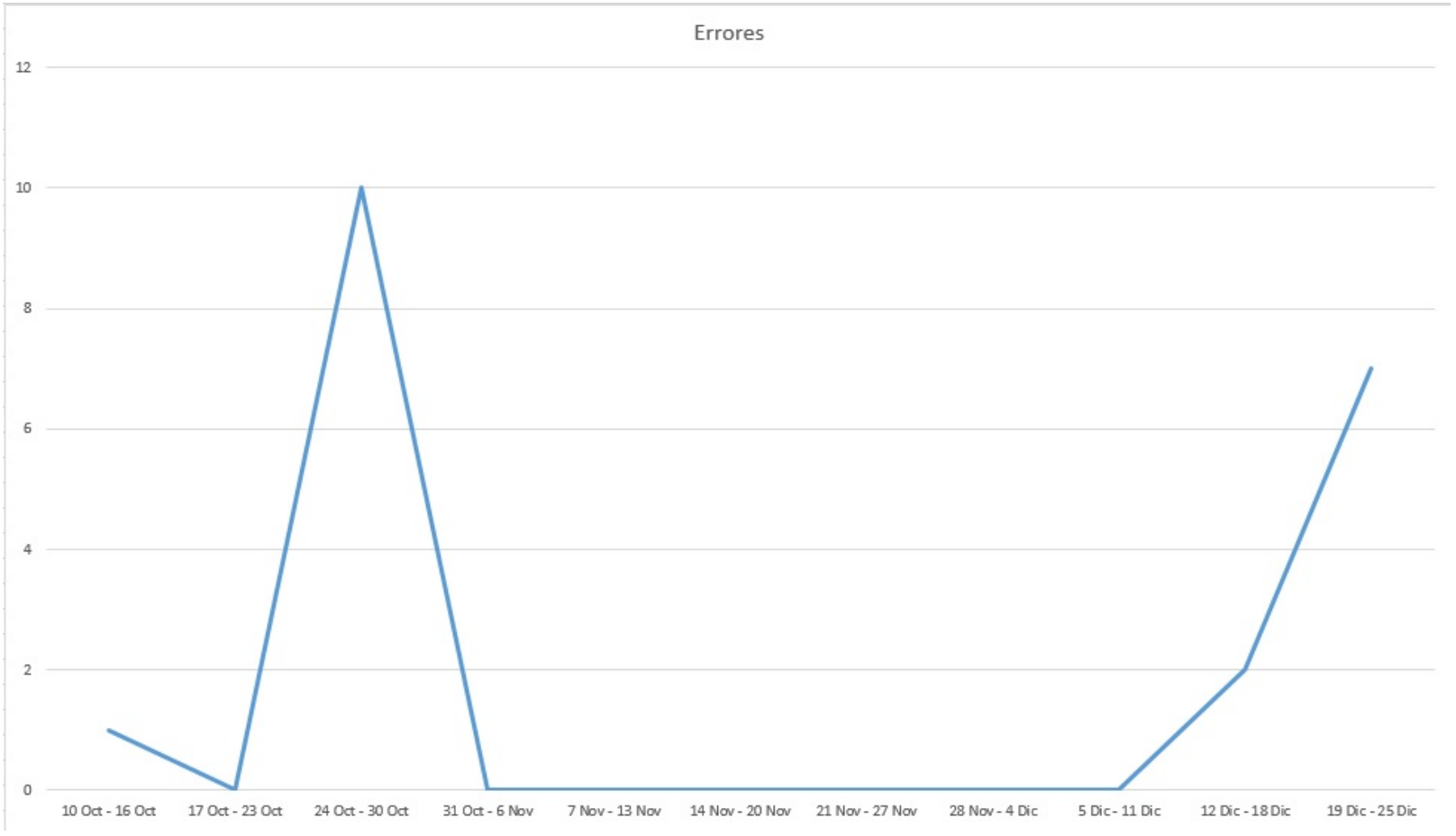
- **EventServiceImpl -addBetType no devuelve DuplicatedResultTypeOptionsException cuando las TypeOptions tienen nombres duplicados #10**
    - Caso de prueba relacionado: PR-UN-029

    - Inspeccionando el código se ha detectado que la implementación no devuelve DuplicatedResultTypeOptionsException es este método para este caso. Crearla y modificar la implementación para que devuelva la excepción.
- **EventInfoBlock no tiene setters #12**
    - EventInfoBlock no tiene setters, por lo que no se pueden inicializar los datos correctamente para ciertos casos de prueba como PR-UN-024 o PR-UN-025
- **Error en la implementación del método pickWinners en EventServiceImpl #13**
    - No hace falta llamar a los DAOs para hacer persistentes los cambios en los datos. Hibernate lo hace automáticamente al llamar al método SET
- **Falta de comprobación de una apuesta negativa #16**
    - El método createBet de BetServiceImpl no devuelve excepción si el amount que se le pasa es negativo. Se ha creado la excepción NegativeAmountException.

6. Estadísticas

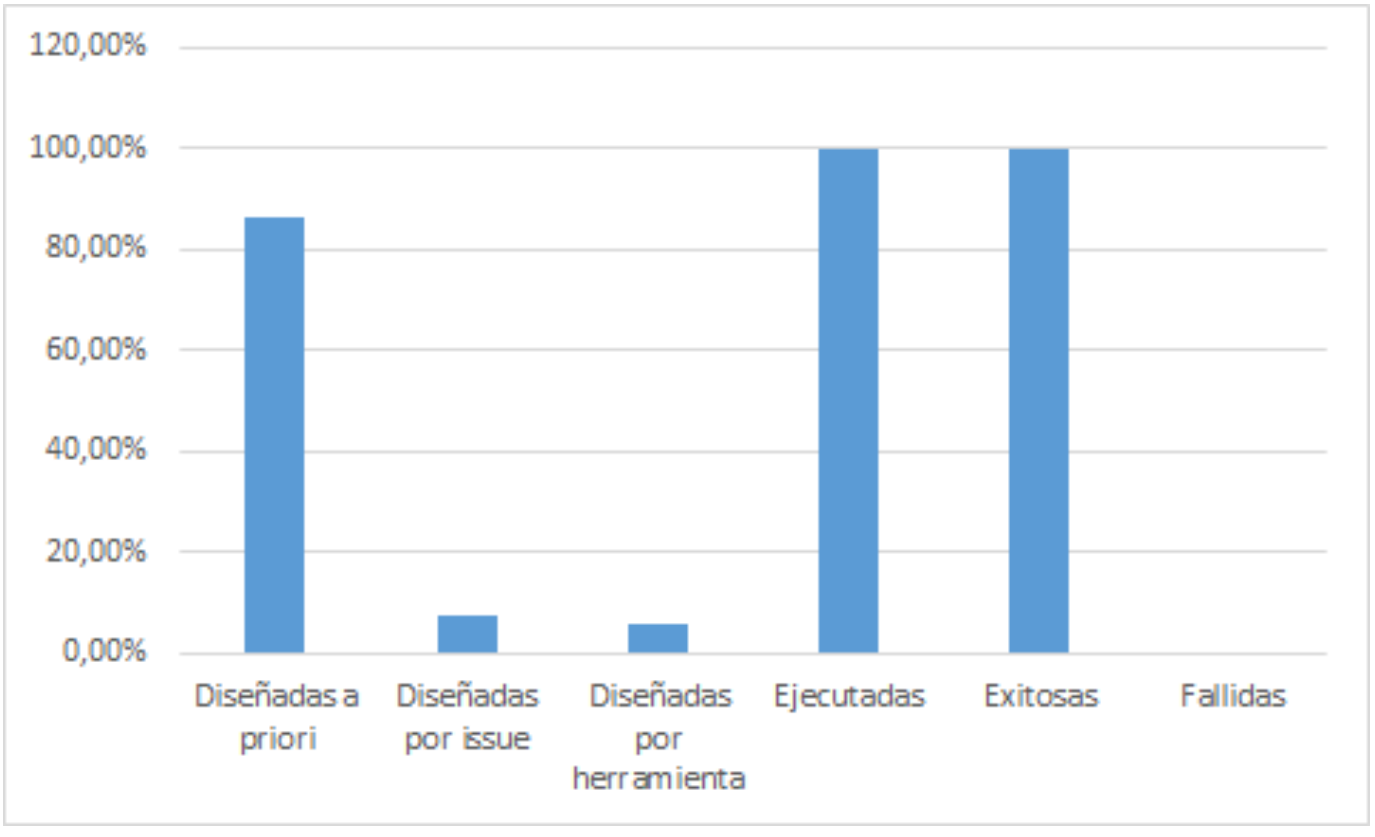
- Número de errores encontrados diariamente y semanalmente

Como ya se mencionó en el punto cuatro de esta memoria, la realización del proceso de pruebas se concentró en la semana de entregas de cada iteración. Por ello, la cantidad de errores se concentran a su vez en estas semanas de trabajo.



- Nivel de progreso en la ejecución de las pruebas

Diseñadas	Diseñadas a priori	Diseñadas por issue	Diseñadas por herramienta	Ejecutadas	Exitosas	Fallidas
103	89	8	6	103	103	0



- Análisis del perfil de detección de errores

Podríamos dividir los perfiles de detección de errores por el alcance del cambio en los componentes.

-Errores graves: Ha habido que cambiar la implementación de un componente de manera considerable. 0 errores de esta tipología encontrados.

-Errores medios: Ha habido que cambiar muy pocas lineas de codigo de un componente o añadir nuevas clases, como excepciones. Todos los listados en el apartado 5.

- Informe de errores abiertos y cerrados por nivel de criticidad

  Todas las issues abiertas han sido cerradas.

- Evaluación global del estado de calidad y estabilidad actuales

Podemos afirmar que, al haber hecho un diseño mucho mejor y más exhaustivo de las pruebas, haber utilizado técnicas y herramientas para testear no el código directamente sino parámetros de calidad como el tiempo de respuesta del servidor, la calidad de los test ha aumentado considerablemente y por lo tanto también la de la aplicación ya que se ha demostrado mayor cantidad de ausencia de errores.

7.Otros aspectos de interés

NA