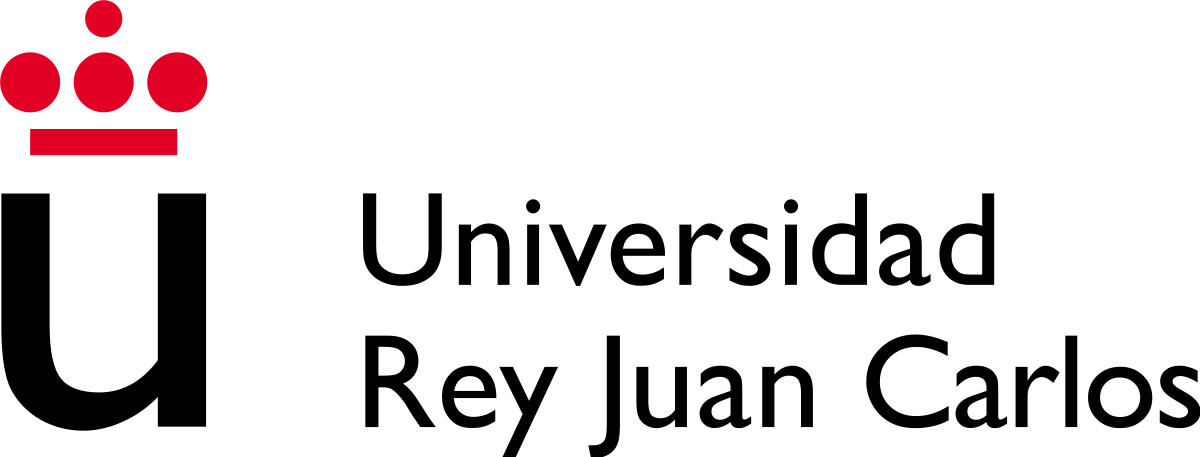
**Práctica 1**

Testing Tic Tac Toe

Introducción

Esta práctica tiene como objetivo la implementación de varios tests para testear la aplicación web TicTacToe. Estas pruebas van a consistir en:

* Pruebas unitarias de la clase Board
* Pruebas con dobles de la clase TicTacToeGame
* Pruebas de sistema

Aplicación Web

La aplicación TicTacToe implementa un tres en raya en formato multijugador vía web. Como se trata de un juego multijugador, para jugar una partida será necesario abrir dos navegadores (o dos pestañas del mismo navegador) apuntando a la dirección web [http://localhost:8082](http://localhost:8082/). Esta aplicación está implementada en Java con la tecnología Spring usando WebSockets.

Pruebas unitarias de la clase Board

La clase Board representa el tablero del juego. Consiste en una lista de celdas con métodos para activarlas o desactivarlas todas, getters para la celda, y métodos para comprobar si hay celdas ganadoras o si hay empate.

La clase BoardTest contiene los tests de la clase Board. Esta clase la componen los siguientes 3 casos de test:

* testGanadorJugadorUno
* testGanadorJugadorDos
* testEmpate

Se ha implementado un método auxiliar para cambiar el valor de las celdas. Se han realizado aserciones al final de cada test para comprobar que el valor devuelto por el método getCellsIfWinner() no es nulo, en el caso de los dos primeros test. Para el tercer test se comprueba que el método isDraw() devuelve ‘true’.

Pruebas unitarias con dobles de la clase TicTacToeGame

La clase TicTacToeGame gestiona el ciclo de vida del juego gestionando el tablero, el

turno del jugador, etc. Los métodos de esta clase se invocan cada vez que se recibe un mensaje de cualquiera de las conexiones con los navegadores web.

El objetivo es comprobar que de esta clase se envían eventos a los navegadores web usando los objetos Connection. En este caso usaremos mocks de la clase Connection en vez de los propios objetos.

La clase BoardTest contiene los tests de la clase Board. Esta clase la componen los siguientes 3 casos de test:

* testGanadorJugador1
* testGanadorJugador2
* testEmpate

En el método @setUp inicializamos los mocks como dobles de la clase Connection, y las añadiremos al objeto TicTacToeGame . Después creamos los objetos Player e inicializamos el contador de turnos a 1.

Se han implementado los siguientes métodos:

comprobarJoinGame() para comprobar que se ha unido el jugador a la partida: Cada vez que se llama a este método se resetea el contador de llamadas de los mocks Connection. Después añade los jugadores al objeto TicTacToeGame y verifica que los mocks han echo la llamada a sendEvents() con el evento JOIN\_GAME con un argumento de una colección que contenga al Jugador1. Luego añade al jugador 2 repitiendo el proceso y comprobando que es la segunda vez que se llama, pero comprobando además que en la segunda llamada a sendEvents() la colección que se pasa como argumento contiene a los dos jugadores.

ComprobarMark() para comprobar que se marca la casilla del juego: Este método marca la celda que recibe como argumento y comprueba que los mocks Connection envían tantas veces como el valga el contador de turnos, que los mocks hayan mandado el evento MARK a los navegadores, pasando como argumento. El subprograma finaliza con el aumento del contador de turnos en 1.

Los tests primero llaman al método comprobarJoinGame. Despúes se llama al método checkTurn del objeto TicTacToeGame comprobando el id del jugador que devuelve ‘true’ si es correcto, es decir, si va alternándose. Luego se llama al método comprobarMark() pasando como argumento la celda a pulsar simulando así una partida real y realizando las comprobaciones oportunas.

La clase WinnerResult hay que mencionarla ya que con ella se puede comprobar si el jugador 1 o el jugador 2 ha ganado ya que el método checkWinner comprueba que haya un ganador devolviendo un objeto WinnerResult. El atributo WinnerResult.win contiene un booleano que comprobando que es ‘true’ comprobaremos que hay ganador.

Para implementar el test de empate llamamos al método checkDraw que devuelve ‘true’ si todas las casillas están ocupadas. Esto es suficiente para comprobar si la funcionalidad del método de empate debido a que si todas las casillas están ocupadas puede significar o que no ha ganado ningún jugador o que ningún jugador puede ganar.

Pruebas de sistema

Las pruebas de sistema se encuentran en la clase WebAppTest

La clase WebAppTest contiene los tests de lsistema. Esta clase la componen los siguientes 3 casos de test:

* testJugador1Gana
* testJugador2Gana
* testEmpate

En el método @setUpClass se obtiene la instancia del driver de Chrome así como pone en marcha además, la aplicación web para que pueda ser utilizable y accesible. El método @setUp que se ejecuta antes de cada test inicia los navegadores y asocia las variables de espera a los mismos.

Se han implementado los siguientes métodos auxiliares:

* El método login() para logear a un jugador a una partida, que comprobará que el tablero se muestra a continuación de pulsar el botón Play.
* El método alertaValida() para verificar que la alerta mostrada es la correcta.

En el caso del primer test la alerta mostrada deberá ser la siguiente “Pepe wins! Juan looses.” y para el segundo test "Juan wins! Pepe looses.". Mientras que para el test de empate el mensaje mostrado será “Draw!”

Integración contínua

Para realizar la integración contínua de la aplicación TicTacToe se ha subido el proyecto a gitHub, en concreto al repositorio xxxx. Se ha implementado un job para el gestor software Jenkins. Se puede encontrar la configuración en la raíz del proyecto en el fichero JenkinsFile. Los resultados arrojados a la vista de Jenkins son los mostrados a continuación: