# Sprint Review 7

Durante este sprint hemos conseguido cumplir todos los **objetivos** que habíamos planteado. En general, el trabajo de este sprint ha consistido en integrar los jugadores máquina, que se encontraban correctamente implementados en otra rama, en la rama principal y resolver algunos errores que no permitían jugar correctamente. Además, hemos ampliado la colección de tests de JUnit, lo que nos ha permitido encontrar algunos errores.

Durante este sprint uno de los cambios más significativos ha sido la **refactorización del sistema de guardado y carga de partidas.**

Anteriormente estábamos empleando el **patrón Memento** ya que nos resultaba bastante conveniente para resolver los problemas que se nos planteaban, esto también fue fruto de nuestro desconocimiento de las arquitecturas multicapa.

En este sprint, y como medida para desacoplar la clase AdminTurnos, hemos optado por refactorizar este sistema y adaptarlo a un **patrón DAO** gracias al cual hemos podido **separar el manejo de ficheros de la lógica de juego.**

Durante este sprint también se ha tenido que adaptar el sistema a la IA y todos los cambios que la inclusión de esta genera.

Por último, **contamos con un documento que explica toda la información relacionada con estas funcionalidades** y su evolución a lo largo del tiempo.

Por otro lado, durante este sprint se ha incluido la posibilidad de **iniciar una partida** en las diferentes modalidades que permite la aplicación (local, en red, cargar partida y unirse a una partida en red). Para ello hemos creado una ventana principal con varios botones que permiten las diferentes funcionalidades. Por otro lado, esta ventana principal será el único medio por el que se puede iniciar el juego, lo que facilita el uso y algunas pruebas del programa.

Además, gran parte del trabajo durante este sprint ha sido destinado a **arreglar errores**. Algunos de los que hemos arreglado provocaban que las máquinas no ejecutasen bien sus turnos, que la información pudiera estar bien representada tanto en el juego en red como en local, y que los jugadores se cargaran correctamente desde el diálogo que utilizamos para hacerlo, entre otros.

Hemos hecho un documento orientativo sobre **JUnit** en el que explicamos brevemente en qué consisten las pruebas y cómo se encuentran estructuradas, para facilitar el empleo de las mismas. Lo más importante es el suite ***AllTestsSuite.java***, el cual corre todos los tests realizados.

Los nuevos tests que se han hecho abarcan algunos comandos del juego que quedaban sin verificar, el generador del diccionario, el lobby, las casillas… También se han añadido nuevos casos de uso para tests ya existentes, como es el caso de TableroTest.java, y se ha probado que se cumplen las reglas del juego.

En este sprint también hemos hecho una refactorización del multijugador. Toda la funcionalidad estaba completada y funcionaba correctamente, pero habías varias clases de esta parte que habían solucionado con parches problemas que surgieron tras la refactorización del modelo. Así, decidimos que era un buen momento para realizarla y convertir estos parches en diseño.

Lo primero que se hizo, fue vaciar la clase traductorServidor, ya que aglutina demasiada funcionalidad, en otras más pequeñas a las que llamamos intérpretes. Llegado este punto, nos encontramos con un problema en el servidor. Encontrábamos distintos tipos de comandos que se ejecutaban de manera distinta según el estado del servidor, así, aplicando un patrón estado solucionamos el problema. La única cuestión que quedaba era la manera en que se construirían estos intérpretes, ya que tenía que cambiar el tipo dinámico en tiempo de ejecución. Este problema de diseño lo solucionamos creando una factoría de intérpretes que los crea dependiendo del estado del servidor.

En la parte del cliente pasaba lo mismo. Teníamos un traductor que tenía demasiada funcionalidad, así, creamos unos comandos que parsearan las órdenes que recibía por el socket y que se ejecutaban sobre unas clases nuevas que implementan cada una una interfaz del modelo. Para más, leer el documento “Multijugador en red”.

En este sprint hemos terminado de hacer las pruebas necesarias para dar el desarrollo del **funcionamiento de las** **IA** por terminado y las hemos incorporado al juego. Esta funcionalidad se desarrolló en otra rama en la que no se encontraban todos los cambios actuales, lo que ha dado problemas a la hora de incorporarlas a la rama principal. Como consecuencia, al intentar hacer un merge de toda la funcionalidad completa ha habido problemas con GitHub que han dejado la rama principal inutilizada. Por eso, nos hemos visto obligados a restaurar el estado de la rama antes del merge en otra rama que ahora utilizamos como rama principal. Después, se procedió a la incorporación por etapas de las máquinas en esta nueva rama y una vez estaba todo añadido nos hemos dedicado a adaptar y ajustar detalles para que funcionase correctamente con las diferencias presentes entre la rama en la que se desarrolló y la nueva rama principal. Para más información sobre la implementación de las IA véase el documento *RefactoriazciónIA*.