SCRUM

INGENIERÍA DEL SOFTWARE II

Grupo 05

Guillermo García Patiño Lenza
Tania Romero Segura
María Cristina Alameda Salas
Alejandro Rivera León
Gema Blanco Núñez
David Cruza Sesmero

ÍNDICE

INTRODUCCIÓN **6**

SCRUM **7**

ESTRUCTURA Y FUNCIONAMIENTO DEL EQUIPO **7**

3.1. PRODUCT OWNER 8

3.2. DEVELOPMENT TEAM 8

3.3. SCRUM MASTER 9

3.4. ORGANIZACIÓN DEL TRABAJO Y EVOLUCIÓN 10

3.5. MÉTODOS DE COORDINACIÓN Y HERRAMIENTAS 12

3.5. CORONAVIRUS 13

HISTORIAS DE USUARIO **14**

4.1 Generar mazo de fichas 14

4.2 Ver el tablero 15

4.3 Colocar fichas en el tablero 16

4.4 Quitar fichas del tablero 18

4.5 Saber el número de fichas que quedan por robar 19

4.6 Robar fichas 19

4.7 Salir de la partida 20

4.8 Saber mi conjunto de fichas actual 21

4.9 Saber cuántos puntos vale cada ficha 22

4.10 Saber las puntuaciones actuales 23

4.11 Cambiar fichas para usar en el próximo turno 25

4.12 Verificar una palabra 25

4.13 Introducir mi nick 27

4.14 Conocer turnos de la partida 28

4.15 Pasar de turno 29

4.16 Poner en duda una verificación 30

4.17 Verificar si una palabra está en el diccionario 31

4.18 Restar puntuación por verificación incorrecta 31

4.19 Restar puntuación por duda incorrecta 32

4.20 Finalizar partida 33

4.21 Almacenar mi puntuación 34

4.22 Utilizar casilla de letra con premio 34

4.23 Colocar más de una ficha por turno 35
4.24 Ganar monedas del juego 36
4.25 Comprar ventajas 36
4.26 Rellenar mano 37
4.27 Colocar más de una ficha por turno 38
4.28 Jugar con otro jugador 38
4.29 Saber quién empieza la partida 39
4.30 Tener en la pila de fichas dos comodines 40
4.31 Canjear comodines de la mano por letras 41
4.32 Obtener puntos por mis jugadas 41
4.33 Poner la primera ficha de la partida en el centro 42
4.34 Guardar partida a medias 42
4.35 Cargar una partida a medias 43
4.36 Pedir si quieres New Game / Load Game 45
4.37 Verificar las palabras automáticamente al finalizar el turno 45
4.38 No pasar de turno si he colocado alguna ficha que no forme palabra 46
4.39 Solo fichas al lado de fichas 47
4.40 Obtener puntos por las monedas sobrantes 47
4.41 Cambiar una ficha 48
4.42 Cambiar una ficha por un comodín a cambio de monedas 49
4.43 Saltar al jugador siguiente a cambio de monedas 50
4.44 Invertir el orden de jugadores a cambio de monedas 52
4.45 Disponer de casillas especiales 53
4.46 Poder jugar con jugadores controlados por la IA 53
4.47 Poder ver la mano en la GUI 55
3.48 Poder ver el tablero en la GUI 55
3.49 Poder ver los turnos en la GUI 56
4.50 Poder ver los puntos de los Jugadores en la GUI 56
4.51 Poder ver las monedas de los jugadores en la GUI 57
4.52 Poder cargar una partida desde la GUI 58
4.53 Poder guardar una partida desde la GUI 58
4.54 Colocar una ficha en el tablero desde la GUI 59
4.55 Poder cambiar una ficha de mi mano desde la GUI 59

4.56 Iniciar una nueva partida desde la GUI 60
4.57 Introducir jugadores y eliminar desde la GUI 60
4.59 Poder pasar turno desde la GUI 61
4.60 Poder saltar jugador en la GUI 62
4.61 Poder invertir el sentido en la GUI 63
4.62 Poder comprar un comodín en la GUI 64
4.63 Poder acceder a las instrucciones de la GUI 65
4.64 Poder jugar en red con otros jugadores 65
4.65 Poder usar comodines 66
4.66 Poder jugar de acuerdo a unas reglas 66
4.67 Iniciar una nueva partida 67
SPRINT REVIEWS **68**
Sprint 1: **68**
Sprint 2: **69**
Sprint 3: **73**
Sprint 4: **74**
Sprint 5: **75**
Sprint 6: **76**
Sprint 7: **78**
Evolución de las Sprint Reviews **80**
SPRINT RETROSPECTIVES **82**
Sprint 1: **82**
Puntos buenos: 82
Puntos a mejorar: 83
Puntos malos: 83
Sprint 2: **84**
Puntos buenos: 84
Puntos a mejorar: 84
Puntos malos: 85
Sprint 3: **85**
Puntos Buenos: 85
Puntos a mejorar: 86

Puntos malos: 86
Sprint 4: 86
Puntos buenos: 86
Puntos a mejorar: 87
Puntos malos: 87
Sprint 5: 88
Puntos buenos: 88
Puntos a mejorar: 88
Puntos malos: 89
Sprint 6: 89
Puntos buenos: 89
Puntos a mejorar: 90
Puntos malos: 91
Sprint 7: 91
Puntos buenos: 91
Puntos a mejorar: 92
Puntos malos: 92
Evolución de las Sprint Retrospectives 93
SPRINT PLANNINGS 95
Sprint 1 95
Sprint 2 95
Sprint 3 95
Sprint 4 95
1. Objetivos: 95
2. Historias de Usuario: 96
Sprint 5 100
1. Objetivos: 100
2. Historias de Usuario: 101
3. Asignación de tareas adicionales 103
Sprint 6 104
1. Objetivos: 104
2. Historias de Usuario: 104

Sprint 7 **106**

1. Objetivos: 106

Evolución de las Sprint Plannings **106**

PRODUCT BACKLOG **107**

SPRINT BACKLOG **108**

DESCRIPCIÓN DEL TRABAJO REALIZADO POR CADA MIEMBRO **108**

1. INTRODUCCIÓN

A lo largo de este documento vamos a exponer los conceptos básicos que hemos aprendido de Scrum y nuestra labor durante todo el cuatrimestre respecto a ello.

Para cada apartado, primero presentaremos una breve teoría sobre este punto. A continuación, presentaremos cómo nos hemos estructurado y recogeremos

los documentos correspondientes a ese punto. Acabaremos reflexionando sobre la evolución que hemos tenido respecto a ese apartado.

2. SCRUM

Scrum es un marco de trabajo ágil para el desarrollo y mantenimiento de productos complejos. Emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. Se basa en la teoría empírica, que toma decisiones basándose en la experiencia.

Características:

- **Transparencia:** Todos los aspectos significativos del proyecto deben ser visibles para todos los responsables del resultado.
- **Inspección:** Se deben inspeccionar los artefactos de Scrum y el proceso hacia un objetivo, detectar variaciones.
- **Adaptación:** Si al inspeccionar se detectan variaciones fuera de lo aceptable, el producto no será aceptable y por tanto deberá ser ajustado.

Durante todo el proyecto hemos tratado de hacer cumplir estas “tres reglas”.

La transparencia la mantenemos gracias al Sprint Planning, que nos proporciona visibilidad acerca de aquello que íbamos a hacer en el sprint; el Daily Scrum, que aporta visibilidad sobre las tareas que realizamos diariamente, los impedimentos y cómo marchaba el trabajo; y el Sprint Review, que nos ofrece visibilidad sobre los logros, resultados y el progreso. Por último, la Sprint Retrospective, que contribuye con la inspección y la adaptación del proceso.

La inspección la llevamos a cabo especialmente durante el Sprint Review y el Sprint Retrospective.

La adaptación la realizamos después de haber concluido la inspección, plasmándola en el Sprint Retrospective.

3. ESTRUCTURA Y FUNCIONAMIENTO DEL EQUIPO

Los equipos Scrum son autoorganizados y multifuncionales. Deben elegir la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Cómo son multifuncionales, tienen todas las competencias necesarias para llevar a cabo el proyecto sin depender de otras personas que no son parte del equipo.

Los equipos scrum entregan productos de forma iterativa e incremental. Las entregas incrementales de producto terminado aseguran que siempre estará disponible una versión potencialmente útil y funcional del producto. Al ser iterativas nos ayudan a encontrar posibles riesgos o variaciones.

3.1. PRODUCT OWNER

Es el responsable de maximizar el valor del producto y del trabajo del equipo de desarrollo. Es la persona responsable de gestionar el Product Backlog. Esta gestión incluye ordenar sus elementos de la mejor manera posible, asegurar que esta lista es transparente y clara para todos, asegurarse de que contiene todos los requisitos del proyecto y que el equipo entienda los elementos de la lista.

El primer laboratorio del cuatrimestre realizamos una votación para elegir al Product Owner. Así, Tania Romero Segura resultó elegida Product Owner

3.2. DEVELOPMENT TEAM

El equipo de desarrollo está formado por todas las personas que desempeñan el trabajo de entregar un incremento del producto terminado al final de cada Sprint. Nuestro equipo de desarrollo está conformado por los siguientes miembros.

- Guillermo García Patiño Lenza
- Tania Romero Segura
- María Cristina Alameda Salas
- Alejandro Rivera León
- Gema Blanco Núñez
- David Cruza Sesmero

Durante el desarrollo del equipo hemos tratado de mantener los principios propios de un equipo Scrum.

Hemos sido autoorganizados. Nadie externo al equipo nos ha dicho cómo debíamos organizado. Hemos tratado de hacerlo por nosotros mismo.

Hemos tratado de ser un equipo multifuncional. Aunque a veces hemos pedido ayuda al profesor para contrastar ideas, se supone que hemos tenido las capacidades necesarias para llevar a cabo el proyecto.

Aunque al principio algunas historias de usuario las realizábamos por parejas, cabe decir que no lo hemos hecho desde el 2º sprint, no ha habido sub-equipos dentro del equipo de desarrollo.

Además, hemos sido un equipo pequeño. Lo suficientemente grande para como para poder completar el proyecto y lo suficientemente pequeño como para permanecer ágil. Ya que el número de personas de un equipo venía ya dado, no hemos aportado demasiado a este punto, aunque hay que destacar que prácticamente todos los miembros del equipo hemos llegado al final aportando gran cantidad de trabajo en cada sprint, de manera que lo hemos mantenido suficientemente grande para poder llevar a cabo el proyecto.

3.3. SCRUM MASTER

El scrum Master es responsable de asegurar que les Scrum es entendido y utilizado por todas las personas del proyecto.

Al igual que el Product Owner, elegimos a Guillermo García Patiño Lenza para este puesto en la primera reunión de laboratorio.

El scrum Master debe estar al servicio del Product Owner, le ayuda a encontrar técnicas para gestionar la lista de productos y para que conozca cómo debe ordenar está lista para maximizar el valor.

Otras de las responsabilidades ha sido estar al servicio del equipo de desarrollo. Nos ha guiado a lo largo de todo el proyecto para que fuera guiando al equipo a que sea autoorganizado y multifuncional.

También se ha asegurado en todo momento de que el equipo Scrum trabajara ajustándose a las prácticas y reglas de Scrum, recordándonoslas en aquellos momentos en los que nos desviábamos. Uno de estos momentos se ha producido cuando no teníamos las facilidades que solíamos tener gracias a las clases presenciales para realizar los daily scrum, reuniones de planning de sprint, retrospective… en todo momento nos ha recordado la importancia de realizar estas reuniones.

Además, también ha estado a disposición del equipo organizando estas reuniones, estableciendo un día y hora en la que pudiéramos todos los miembros del equipo.

3.4. ORGANIZACIÓN DEL TRABAJO Y EVOLUCIÓN

- ***Organización del trabajo***

Con respecto a la organización del trabajo, prácticamente siempre hemos seguido la misma línea, aunque naturalmente ha habido una evolución con el paso del tiempo.

Comenzamos el sprint realizando el Sprint Planning. En esta reunión acordamos el trabajo que se va a llevar a cabo durante el nuevo Sprint. Estudiamos que podemos entregar en el nuevo incremento del sprint y cómo podemos conseguir realizar el trabajo para entregarlo en ese sprint. Entonces, se elabora el Sprint Backlog a partir del Product Backlog y el último incremento del proyecto. En este momento, le asignamos a cada una de las historias de usuario un programador responsable, siendo la carga de trabajo lo más parecida posible entre miembros o adaptada a la cantidad de trabajo que puede realizar en este tiempo.

Una vez elaborado el Sprint Backlog, pasábamos las tareas a la pestaña de “Projects” de GitHub. De esta manera, todos los miembros del equipo tenía conocimiento del estado actual del proyecto, sabiendo cuales eran las tareas completadas, en progreso, o sin empezar.

Entonces comenzaban oficialmente las tareas del sprint. Durante un período de dos semanas, cada miembro del equipo realizaba las tareas de las que era responsable. Seguíamos manteniendo el contacto a través de las herramientas nombradas en el apartado siguiente. De esta manera, conocíamos el estado del proyecto por otra fuente distinta a la pestaña de “Projects” de GitHub. Gracias a esta comunicación, si algún miembro del equipo tenía algún problema para completar su tarea podía pedir ayuda para completarla. O si se encontraban bugs de alguna parte ya completada en otro sprint, estábamos al tanto de ello para que lo arreglara un programador disponible. Además, intentábamos realizar Daily Scrums cada día después o antes de clase pero no siempre resultó así.

Si teníamos algún problema muy concreto, abríamos una “Issue” en GitHub. Cuando llegábamos a un acuerdo sobre su solución, se cerraba.

Con respecto a la realización de una tarea, nuestro objetivo ha sido siempre mantener la concordancia entre el código y los diagramas. Así, la terminar una tarea no comprendía solamente programar código sino también mantener la concordancia de este con el diseño. Para ello, solíamos primero diseñar la historia en diagramas de secuencia y de clases y a partir de ese momento crear el código. No siempre hemos seguido este orden, pero casi siempre para final de sprint teníamos ambas tareas realizadas.

Una vez que terminaba este período de dos semanas, cerrábamos sprint.

El cierre del sprint requería de la celebración de dos reuniones de una hora aproximadamente cada una.

Primero, el Sprint Retrospective. En esta reunión, el Equipo Scrum se inspecciona a sí mismo. El propósito es inspeccionar cómo fue el último sprint en cuanto a personas, relaciones, procesos y herramientas; identificar y ordenar los elementos que salieron bien, las posibles mejoras que se podrían realizar y

crear un plan para implementar estas mejoras de la mejor manera posible. Es la reunión básica de inspección de la organización del equipo.

Después, el Sprint Review. Se inspecciona el incremento y se adapta el Product Backlog si fuese necesario. En esta reunión el equipo presenta lo que se hizo durante el sprint. Cada miembro del equipo exponía los cambios que había realizado en el proyecto y se establecía si su tarea había sido completada.

- ***Evolución de la organización***

La organización del trabajo y del proyecto ha variado a lo largo del cuatrimestre.

Con respecto a la asignación de un programador responsable, no la realizamos el primer sprint, sprint que resultó ser bastante caos con respecto a la organización. Muchas de las tareas se dejaron para el final del sprint puesto que nadie era responsable de ellas. Sin embargo, tras aprender de nuestros errores, decidimos asignar a un programador a cada una de las historias, de esta manera un miembro se preocupa solo de las tareas que se le han asignado y por sentirse responsable, en la mayoría de los sprints hemos cumplidos con los objetivos propuestos. A partir de aquel sprint siempre se ha realizado.

Durante los primeros sprints tampoco utilizábamos las pestañas “Projects” ni “issues” de GitHub, perdiéndonos las facilidades que aportan a la hora de controlar el estado del proyecto y organizar las discusiones sobre distintos temas.

Como se puede apreciar, la evolución ha sido siempre a mejor. Aquellas cosas en las que encontramos un problema, buscábamos una solución que incorporábamos en el siguiente sprint.

3.5. MÉTODOS DE COORDINACIÓN Y HERRAMIENTAS

Durante el desarrollo del proyecto, hemos comenzado a usar varias herramientas que nos facilitan la coordinación como las pestañas de proyectos e issues de GitHub, Skype, Google Docs,y el grupo de WhatsApp en el que participamos todos los miembros del grupo.

- WhatsApp:

  Esta herramienta fue la primera que empezamos a usar para comunicarnos. Al principio del desarrollo del proyecto, creamos un grupo e incluimos a todos en él. A partir de ese punto, el grupo ha servido como punto de encuentro para que cada integrante del grupo se mantenga

informado, pida ayuda y consulte dudas con el resto cuando lo necesite, o para que entre todos tomemos decisiones sobre asuntos que incumben al grupo entero como por ejemplo, horas de tutorías y reuniones, entre otros asuntos.

Además, esta plataforma ha servido para que los diferentes integrantes del grupo comuniquen al resto las actualizaciones o comunicados que se publicaban en las demás plataformas.

- Google Docs:

  Google Docs ha sido una herramienta fundamental para el desarrollo del proyecto desde el inicio de éste. Nos hemos valido de este recurso para elaborar de forma colaborativa la mayoría de documentos de Scrum, el documento de historias de usuario, y para poner en común algunos documentos de diseño.

- Skype:

  Durante el confinamiento que ha provocado la situación, la comunicación a través de sistemas de mensajería ha sido útil, pero no ha podido sustituir la comunicación presencial de la que disponíamos durante el periodo de clases en la facultad. Es aquí cuando hemos hecho uso de Skype para mantener las reuniones que eran necesarias para cerrar los sprints, o para acordar ciertos aspectos del diseño.

- Issues en GitHub:

  La función de issues de la que dispone GitHub nos ha sido útil para tratar aspectos de diseño que son demasiado extensos para tratarlos por otros medios. Además, nos ha sido útil para exponer nuestros diseños mediante diagramas al resto del grupo y debatir acerca de ellos.

  Por otro lado, nos ha permitido recibir algunas correcciones del profesor, que nos han ayudado a diseñar el juego más limpiamente, y en consecuencia, a implementar ciertas funcionalidades más limpia y fácilmente.

- Projects en GitHub:

  A pesar de no haber empleado esta herramienta hasta pasada la fase inicial del desarrollo, en cuanto comenzamos a hacerlo, nos dimos cuenta

de lo útil que es la pestaña de projects de GitHub para tener una visión panorámica del estado de las tareas que planificamos para cada sprint.

Además, nos ha servido para estimar la dificultad de las diferentes tareas que planificamos para cada sprint, y de esta manera, ajustar mejor la carga de trabajo para cada sprint.

3.5. CORONAVIRUS

Debido a la situación excepcional en la que nos hemos encontrado, tuvimos que suspender por completo las Daily Scrum presenciales.

Aunque no hemos continuado con ellas a una hora establecida como se realizaba antes de esta situación, sí que aumentamos consideradamente el tiempo empleado a la comunicación a través de las herramientas destinadas a ello. Puesto que el objetivo de los Daily Scrum se ha mantenido, podemos afirmar que no ha afectado significativamente al proyecto.

Las reuniones antes presenciales que se realizaban durante los laboratorios se han seguido realizando sin problemas el mismo día que correspondía.

El único apartado en el que sí ha influido directamente ha sido con la carga de trabajo que podíamos realizar durante un sprint, ya que el confinamiento ha supuesto un incremento de las horas que podíamos dedicar a la asignatura. De esta manera, podemos decir que ha influido positivamente a la hora de terminar el proyecto o en añadir más funcionalidad a la aplicación.

4. HISTORIAS DE USUARIO

Representación de un requisito del proyecto. Se le asigna un número y un título, una prioridad, una complejidad, un programador y una pequeña descripción de la forma: Como …. Quiero… Para….

Es un artefacto de Scrum vivo, a medida que avanza al proyecto, surgen nuevos requisitos y se deben añadir a esta lista. Por tanto, es un artefacto ampliable.

4.1 Generar mazo de fichas

ID: H_1	**Usuario:** Jugador
Nombre historia: Generar mazo de fichas	
Prioridad: Alta	**Estado:** Finalizado - Eliminada
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: María Cristina Alameda Salas	
Descripción: Como Jugador quiero que se genere el mazo de fichas para que pueda utilizarlas en mi mano.	

Descripción detallada

Según se organizó el Product Backlog, esta historia de usuario corresponde con una de las funcionalidades básicas del juego.

El objetivo era claro, poder generar un conjunto aleatorio de fichas para que pudieran jugar los jugadores.

Esta Historia de Usuario fue eliminada más tarde debido a que consideramos que esto no era realmente una Historia de Usuario sino una funcionalidad interna del juego necesaria para la implementación de otras Historias de Usuario.

4.2 Ver el tablero

ID: H_2	**Usuario:** Jugador
Nombre historia: Ver el tablero	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable:	
Descripción:	

Como jugador quiero poder ver el tablero para seguir la colocación de las fichas.

Descripción detallada

Una de las primeras funcionalidades que decidimos implementar en el proyecto fue el tablero donde los jugadores colocan fichas. Al principio se mostraba por consola, pero finalmente esta vista fue reemplazada por otra en una GUI.

Posteriormente se incluyó como evolución de esta Historia de Usuario el contenido de la Historia de Usuario Poder ver el tablero en la GUI. Consideramos que esta segunda historia realmente es la misma que la primera cambiando la forma en la que se presenta la información y por lo tanto consideramos que debería haber sido tratada como una evolución de la primera.

ID: H_1	**Usuario:** Jugador
Nombre historia: Ver el tablero	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable:	
Descripción: Como jugador quiero poder ver el tablero para seguir la colocación de las fichas.	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable: Guillermo García Patiño Lenza	
Estado: Finalizado	**Iteración asignada:** 4

4.3 Colocar fichas en el tablero

ID: H_3	**Usuario:** Jugador
Nombre historia: Colocar fichas en el tablero	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: María Cristina Alameda Salas, Alejandro Rivera León, Guillermo García Patiño Lenza.	
Descripción: Como jugador quiero poder colocar fichas al tablero para poder formar palabras.	

Descripción detallada

Con esta funcionalidad el objetivo principal era dotar al usuario de la posibilidad de poner fichas en el tablero usando como valores de entrada las coordenadas y la ficha que quisiera poner.

Al igual que con otras historia de esta índole, posteriormente si incluyó como parte de su evolución a la historia Colocar fichas en la GUI ya que supone una modificación del funcionamiento de ésta primera y no es realmente una historia nueva. La historia resultante es la siguiente:

ID: H_2	**Usuario:** Jugador
Nombre historia: Colocar fichas en el tablero	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Alejandro Rivera León	
Descripción:	

Como jugador quiero poder colocar fichas al tablero para poder formar palabras.
Evolución
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.
Programador responsable:
Estado: Finalizado **Iteración asignada:** 1

4.4 Quitar fichas del tablero

ID: H_4	**Usuario:** Jugador
Nombre historia: Quitar fichas en el tablero	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable:	
Descripción: Como jugador quiero quitar las fichas que he colocado este turno del tablero para poder rectificar y hacer una mejor jugada.	

Descripción detallada:

Al haber desarrollado ya la funcionalidad de colocar ficha, otra de las funcionalidades sencillas de hacer fue la de quitar una ficha del tablero. Fue sencilla de implementar gracias a la aplicación del patrón comando.

Tal y como ocurre con el resto de historias que fueron ampliadas con nuevas historias tras la inclusión de la GUI, la historia creada a posteriori ha sido incluida en esta como parte de su evolución. La historia resultante es la siguiente:

ID: H_3	**Usuario:** Jugador
Nombre historia: Quitar fichas del tablero	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como jugador quiero quitar las fichas que he colocado este turno del tablero para poder rectificar y hacer una mejor jugada.	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable:	
Estado: Finalizado	**Iteración asignada:** 1

4.5 Saber el número de fichas que quedan por robar

ID: H_5	**Usuario:** Jugador
Nombre historia: Saber el número de fichas que quedan por robar.	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Maria Cristina Alameda Salas	
Descripción: Como jugador quiero saber cuántas fichas quedan por robar para poder definirme estrategias y hacer mejores jugadas.	

Descripción detallada

El objetivo de esta historia era que los jugadores pudieran saber en todo momento el número de fichas que quedan por repartir ya que una de las maneras de finalizar una partida es cuando no queden fichas restante en el mazo.

4.6 Robar fichas

ID: H_6	**Usuario:** Jugador
Nombre historia: Robar fichas	
Prioridad: Alta	**Estado:** Sin empezar
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: - - -	
Descripción: Como jugador quiero robar fichas para aumentar mi número de fichas	

Descripción detallada

Al comienzo del proyecto, planteamos esta historia para solventar la necesidad de que los jugadores dispusieran de 7 fichas al comienzo de su turno tal y como ocurre en el juego original. Esta solución tenía como objetivo permitirle al jugador robar fichas hasta completar su mano del mazo de fichas del juego.

Esta historia no llegó a desarrollarse y fue sustituida posteriormente por la Historia de Usuario Rellenar mano del jugador.

4.7 Salir de la partida

ID: H_7	**Usuario:** Jugador
Nombre historia: Salir de la partida	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: David Cruza Sesmero	

Descripción: Como jugador quiero salir de la partida para acabar la partida.

Descripción detallada

El objetivo de esta historia era que un jugador pudiera salir de una partida a medias. Esto se conseguía añadiendo una nueva clase a la jerarquía de comandos.

Posteriormente esta historia se modificó para incluir dentro de su evolución la reevaluación de esta funcionalidad debido a la incorporación de de la GUI. La Historia de Usuario resultante es la siguiente:

ID: H_5	**Usuario:** Jugador
Nombre historia: Salir de la partida	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable:	
Descripción: Como jugador quiero salir de la partida para acabar la partida.	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable: Guillermo García Patiño Lenza	
Estado: Finalizado	**Iteración asignada:** 1

4.8 Saber mi conjunto de fichas actual

ID: H_8	**Usuario:** Jugador

Nombre historia: Saber mi conjunto de fichas actual	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Jorge Rodríguez	
Descripción: Como jugador quiero salir de la partida para acabar la partida.	

Descripción detallada

Esta historia tiene como objetivo dar a conocer al usuario uno de los elementos fundamentales del juego, que es el atril del jugador en el cual se encuentran las fichas de las que dispone para colocar en el tablero.

Posteriormente se incluyó la Historia de Usuario Poder ver mi conjunto de fichas en la GUI como evolución de esta primera. El resultado es el siguiente:

ID: H_6	**Usuario:** Jugador
Nombre historia: Saber mi conjunto de fichas actual	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Jorge Rodríguez (como desapareció el resto suplimos esta funcionalidad)	
Descripción: Como jugador quiero saber mi conjunto de fichas actual para poder tomar decisiones sobre mi jugada respecto a las fichas de las que dispongo	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable: María Cristina Alameda Salas	

Estado: Finalizado	**Iteración asignada:** 4

4.9 Saber cuántos puntos vale cada ficha

ID: H_9	**Usuario:** Jugador
Nombre historia: Saber cuantos puntos vale cada ficha	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: María Cristina Alameda Salas	
Descripción: Como jugador quiero saber cuántos puntos vale cada ficha de las que dispongo para poder estimar los puntos que gano con una palabra.	

Descripción detallada

Esta historia suponía cumplir con unos de los requisitos del juego que consistía en que los jugadores pudieran conocer la puntuación de cada una de las fichas que aparecieran en el juego.

Posteriormente se incluyó en esta historia como parte de su evolución las modificaciones necesarias tras la incorporación de la GUI:

ID: H_7	**Usuario:** Jugador
Nombre historia: Saber cuantos puntos vale cada ficha	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: María Cristina Alameda Salas	
Descripción:	

Como jugador quiero saber cuántos puntos vale cada ficha de las que dispongo para poder estimar los puntos que gano con una palabra.
Evolución
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.
Programador responsable: María Cristina Alameda Salas
Estado: Finalizado **Iteración asignada:** 4

4.10 Saber las puntuaciones actuales

ID: H_10	**Usuario:** Jugador
Nombre historia: Saber las puntuaciones actuales	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: - - -	
Descripción: Como jugador quiero saber las puntuaciones actuales para poder hacer estimaciones sobre la puntuación final	

Descripción detallada

Conocer las puntuaciones actuales de los jugadores era clave para poder finalizar una partida, por ello fue una de las historias de usuario que antes se realizaron.

Al igual con el resto de historias que se reevaluaron tras la incorporación de la GUI a esta historia se incorporó su sucesora como parte de su evolución:

ID: H_8	**Usuario:** Jugador

Nombre historia: Saber las puntuaciones actuales	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable:	
Descripción: Como jugador quiero saber las puntuaciones actuales para poder hacer estimaciones sobre la puntuación final	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable: Alejandro Rivera León	
Estado: Finalizado	**Iteración asignada:** 4

4.11 Cambiar fichas para usar en el próximo turno

ID: H_11	**Usuario:** Jugador
Nombre historia: Cambiar fichas para usar en el próximo turno	
Prioridad: Alta	**Estado:** Eliminada
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: - - -	
Descripción: Como jugador quiero cambiar fichas para poder usarlas el próximo turno y tener más jugadas posibles	

Descripción detallada

Esta Historia de Usuario fue eliminada antes de poder ser implementada. Llegamos a la conclusión de que no tenía sentido cambiar fichas para el próximo turno. Por ello fue eliminada y más adelante se incluyó otra Historia de Usuario para crear una funcionalidad que permitiese cambiar una ficha por otra y poder usarla en ese mismo turno.

4.12 Verificar una palabra

ID: H_12	**Usuario:** Jugador
Nombre historia: Verificar una palabra	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Tania Romero Segura y María Cristina Alameda Salas	
Descripción: Como jugador quiero verificar una palabra para aumentar mi puntuación	

Descripción detallada

Uno de las claves de este juego era poder verificar una palabra. Este fue un primer intento sobre el que se evolucionó en una historia posterior. En esta primera aproximación, el jugador disponía de un comando para verificar una palabra al que tenía que pasarle las coordenadas de la primera letra y las coordenadas de la última. Después, se creaba una palabra a partir de cada letra del tablero contenida entre esas posiciones y se comprobaba si esa letra estaba en el diccionario.

Posteriormente se creó una historia nueva para reimplementar esta funcionalidad de manera que se hiciese de manera automática al pasar el turno. Después, se decidió que en vez de tener dos historias se podía tener una sola en la que figurase la segunda como evolución de la primera. El resultado es el siguiente:

ID: H_20	**Usuario:** Jugador
Nombre historia: Verificar una palabra	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: María Cristina Alameda Salas y Tania Romero Segura.	
Descripción: Como jugador quiero que se verifiquen las palabras que pongo	
Evolución	
Motivación: Esta historia se reabrió posteriormente para que esta funcionalidad se hiciera de forma automática al pasar de turno. En un principio se hacía con un comando del jugador.	
Programador responsable: María Cristina Alameda Salas	
Estado: Finalizado	**Iteración asignada:** 3

4.13 Introducir mi nick

ID: H_13	**Usuario:** Jugador
Nombre historia: Introducir mi nick	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: David Cruza Sesmero	
Descripción: Como jugador quiero introducir mi nick para identificarme en mis partidas.	

Descripción detallada

Se tenía que introducir un nick para cada jugador puesto que en este sprint se añadiría la opción de jugar contra otro jugador y debíamos diferenciarlo de alguna manera además de que en un principio teníamos la idea de crear un ranking pero dicha idea se acabó desechando.

Posteriormente fue necesario permitirle al juagdor introducir su nick a través de la GUI. Como resultado hemos creado una Historia de Usuario en la que figura este cambio como parte de su evolución:

ID: H_9	**Usuario:** Jugador
Nombre historia: Introducir mi nick	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: David Cruza Sesmero	
Descripción: Como jugador quiero introducir mi nick para identificarme en mis partidas.	
Evolución	
Motivación: Esta historia se reabrió posteriormente para que se pudiera introducir los nicks de los jugadores desde la nueva interfaz gráfica	
Programador responsable: María Cristina Alameda Salas y Tania Romero Segura se encargaron de la primera funcionalidad	
Estado: Finalizado	**Iteración asignada:** 2

4.14 Conocer turnos de la partida

ID: H_14	**Usuario:** Jugador
Nombre historia: Conocer turnos de la partida	

Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Alejandro Rivera León	
Descripción: Como jugador quiero conocer los turnos de la partida para saber cuál es el orden de jugada de los jugadores.	

Descripción detallada

Uno de los puntos fundamentales para mantener la coherencia en una partida con varios jugadores es conocer quién está jugando actualmente y el orden que siguen los turnos para saber quién será el siguiente en jugar.

Tal y como ocurre con otras historias que fueron reevaluadas en nuevas historias tras la incorporación de la GUI, finalmente se incluyeron esas nuevas historias en las originales como parte de su evolución:

ID: H_10	**Usuario:** Jugador
Nombre historia: Conocer turnos de la partida	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Alejandro Rivera León	
Descripción: Como jugador quiero conocer los turnos de la partida para saber cuál es el orden de jugada de los jugadores.	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar esta funcionalidad a la GUI	

Programador responsable: Gema Blanco Núñez	
Estado: Finalizado	**Iteración asignada:** 4

4.15 Pasar de turno

ID: H_15	**Usuario:** Jugador
Nombre historia: Pasar de turno	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Gema Blanco Núñez	
Descripción: Como jugador quiero pasar de turno para terminar mi jugada.	

Descripción detallada

Para la jugabilidad entre dos o más jugadores es fundamental poder ceder el turno al siguiente contrincante. Al principio, esta funcionalidad fue implementada por medio de comandos que el usuario debía introducir. Posteriormente, fue implementada mediante un acceso por botón en la GUI.

4.16 Poner en duda una verificación

ID: H_16	**Usuario:** Jugador
Nombre historia: Poner en duda una verificación	

Prioridad: Alta	**Estado:** Sin empezar
Puntos estimados: 1	**Iteración asignada:**
Programador responsable: - - -	
Descripción: Como jugador quiero poner en duda una verificación de una palabra de otro jugador para evitar que aumente su puntuación.	

Descripción detallada

Esta Historia de Usuario se incluyó en la primera versión del Product Backlog. En ese momento, la idea que se tenía para implementar el sistema de verificación de palabras era que un jugador pudiera verificar una palabra y que esto consistiera en que se le sumasen los puntos independientemente de si está en el diccionario o no. Por eso, se pretendía darle la oportunidad a los contrincantes de poner en duda la verificación de otro jugador para que este no sumase puntuación por palabras que no existen.

Posteriormente, el Product Owner decidió eliminar esta historia porque prefería que solo se pudieran verificar palabras correctas ya que, técnicamente, así es como funciona el Scrabble y si se colocan palabras incorrectas es precisamente por errores humanos que no deberían existir en la versión que se está realizando para ordenador.

4.17 Verificar si una palabra está en el diccionario

ID: H_17	**Usuario:** Verificador
Nombre historia: Verificar si una palabra está en el diccionario	
Prioridad: Alta	**Estado:** Sin empezar
Puntos estimados: 5	**Iteración asignada:**
Programador responsable: - - -	
Descripción: Como verificador quiero verificar si una palabra está en el diccionario para saber si una palabra es correcta.	

Descripción detallada

Esta Historia de Usuario se incluyó en la primera versión del Product Backlog. En ese momento, la idea que se tenía para implementar el sistema de verificación de palabras era que un jugador pudiera verificar una palabra independientemente de si está en el diccionario o no. Para que los contrincantes pudieran poner en duda una verificación tal y como se plantea en la historia anterior, era necesario poder comprobar que una palabra está en el diccionario.

Dado que es una de las primeras historias que creamos, tiene un error de concepto en lo que se refiere al usuario. Aquí asignamos a una entidad verificador como usuario. Esto en realidad pretendía ser una clase o similar que se llamase Verificador que se encargase de realizar estas tareas.

Posteriormente, el Product Owner decidió eliminar esta historia porque prefería que solo se pudieran verificar palabras correctas tal y como se menciona previamente.

4.18 Restar puntuación por verificación incorrecta

ID: H_18	**Usuario:** Verificador
Nombre historia: Restar puntuación por verificación incorrecta.	
Prioridad: Alta	**Estado:** Sin empezar
Puntos estimados: 1	**Iteración asignada:**
Programador responsable: - - -	
Descripción: Como verificador quiero restar puntuación por verificación incorrecta para disminuir la puntuación del jugador que quiso verificar la palabra.	

Descripción detallada

Esta Historia de Usuario se incluyó en la primera versión del Product Backlog. En ese momento, la idea que se tenía para implementar el sistema de verificación de palabras era que un jugador pudiera verificar una palabra independientemente de si está en el diccionario o no. Si un contrincante ponía en duda una verificación y ésta resultaba incorrecta, se pretendía incluir una penalización por la que el jugador que había cometido la infracción perdería los puntos que había ganado más unos pocos más como castigo.

Al igual que la historia anterior, tiene un error de concepto en la asignación del usuario. Además de que esto, si se hubiera implementado no debería ser una Historia de Usuario individual sino parte de las reglas del juego.

Posteriormente, el Product Owner decidió eliminar esta historia porque prefería que solo se pudieran verificar palabras correctas tal y como se menciona previamente.

4.19 Restar puntuación por duda incorrecta

ID: H_19	**Usuario:** Verificador
Nombre historia: Restar puntuación por por duda incorrecta.	
Prioridad: Alta	**Estado:** Sin empezar
Puntos estimados: 1	**Iteración asignada:**
Programador responsable: - - -	
Descripción: Como verificador quiero restar puntuación por duda incorrecta para disminuir la puntuación del jugador que dudó de una palabra.	

Descripción detallada

Esta Historia de Usuario se incluyó en la primera versión del Product Backlog. En ese momento, la idea que se tenía para implementar el sistema de verificación de palabras era que un jugador pudiera verificar una palabra independientemente de si está en el diccionario o no. Si un contrincante ponía en duda una verificación y ésta resultaba correcta, se pretendía penalizar la reclamación incorrecta con una sustracción de puntos.

Al igual que la historia anterior, tiene un error de concepto en la asignación del usuario. Además de que esto, si se hubiera implementado no debería ser una Historia de Usuario individual sino parte de las reglas del juego.

Posteriormente, el Product Owner decidió eliminar esta historia porque prefería que solo se pudieran verificar palabras correctas tal y como se menciona previamente.

4.20 Finalizar partida

ID: H_20	**Usuario:** Jugador
Nombre historia: Finalizar partida	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: - - -	
Descripción: Como jugador quiero finalizar la partida para comprobar quién de los jugadores ha ganado.	

Descripción detallada

Era importante para el juego poder finalizar una partida. De esta manera al acabarse, se pueden conocer los ganadores y perdedores de la misma.

4.21 Almacenar mi puntuación

ID: H_21	**Usuario:** Jugador
Nombre historia: Almacenar mi puntuación	
Prioridad: Media	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: David Cruza Sesmero	
Descripción: Como jugador quiero almacenar mi puntuación para poder establecer un ranking respecto de otras partidas.	

Descripción detallada

Esta historia de usuario se acabó eliminando en los posteriores sprints por problemas con la carga de los jugadores puesto que todavía no teníamos muy claro cómo iban a puntuar los jugadores. En un primer momento se añadió una "leaderboard" que guardaba a los jugadores con la puntuación obtenida en la partida y la guardaba pero no comprobaba si había nicks iguales en dicho documento.

4.22 Utilizar casilla de letra con premio

ID: H_22	**Usuario:** Jugador
Nombre historia: Utilizar casilla de letra con premio	
Prioridad: Baja	**Estado:** Sustituida
Puntos estimados:	**Iteración asignada:**
Programador responsable: - - -	
Descripción: Como jugador quiero utilizar casillas de letra con premio para doblar la puntuación que me otorga la letra colocada en esa casilla.	

Descripción detallada

Tanto esta Historia de Usuario como la siguiente tenían como objetivo permitir al usuario jugar con un tablero igual al original de Scrabble y para ello era necesario incluir las casillas con premio. En esta historia en concreto se pretendía proporcionar al jugador las casillas que daban doble de puntuación y triple de puntuación por letra.

Posteriormente, estas dos historias fueron sustituidas por la Historia de Usuario Disponer de casillas especiales. Además, como puede verse, hay un error en la descripción dado que solo se habla de duplicar la puntuación.

4.23 Colocar más de una ficha por turno

ID: H_23	**Usuario:** Jugador
Nombre historia: Utilizar casilla de palabra con premio	
Prioridad: Baja	**Estado:** Sustituida

Puntos estimados:	**Iteración asignada:**
Programador responsable: - - -	
Descripción: Como jugador quiero utilizar casillas de palabra con premio para doblar la puntuación que me otorga la palabra si esta pasa por este tipo de casilla.	

Descripción detallada

Tanto esta Historia de Usuario como la anterior tenían como objetivo permitir al usuario jugar con un tablero igual al original de Scrabble y para ello era necesario incluir las casillas con premio. En esta historia en concreto se pretendía proporcionar al jugador las casillas que daban doble de puntuación y triple de puntuación por la puntuación total de la palabra.

Posteriormente, estas dos historias fueron sustituidas por la Historia de Usuario Disponer de casillas especiales.Además, como puede verse, hay un error en la descripción dado que solo se habla de duplicar la puntuación.

4.24 Ganar monedas del juego

ID: H_24	**Usuario:** Jugador
Nombre historia: Ganar monedas del juego	
Prioridad: Baja	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Tania Romero Segura	
Descripción: Como jjugador quiero ganar monedas del juego para realizar compras durante las jugadas.	

Descripción detallada

Desde el principio del desarrollo del proyecto hicimos varios planteamientos sobre cómo podíamos ampliar el juego. Al final optamos por darle al jugador la posibilidad de comprar ventajas que le permitieran mejorar sus jugadas y entorpecer el avance de otros jugadores. Para ello, era necesario crear un

sistema de pago por la compra de esas ventajas. Como resultado surgió esta Historia de Usuario por la que pretendíamos darle al jugador la posibilidad de ganar x monedas cada ciertos puntos para que luego pudiera comprar ventajas.

4.25 Comprar ventajas

ID: H_25	**Usuario:** Jugador
Nombre historia: Comprar ventajas	
Prioridad: Baja	**Estado:** Sustituida
Puntos estimados:	**Iteración asignada:**
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero comprar 'ventajas' para mejorar mis jugadas	

Descripción detallada

Como ampliación del juego decidimos otorgarle al jugador la posibilidad de comprar ventajas para mejorar sus jugadas o incluso entorpecer las de sus contrincantes. Para ello se incluyó esta Historia de Usuario cuando aún no estaban definidas las ventajas que íbamos a incluir.

Posteriormente, una vez se decidió qué ventajas se iban a incluir, se optó por dividir esta historia en tres Historias de Usuario más pequeñas con el fin de aportar claridad sobre este asunto. Así se crearon las Historias de Usuario: Comprar comodín a cambio de monedas, Saltar a un jugador e Invertir sentido de jugadores.

4.26 Rellenar mano

ID: H_6	**Usuario:** Jugador
Nombre historia: Rellenar mano.	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 1

Programador responsable: María Cristina Alameda Salas y Tania Romero Segura
Descripción: Como jugador quiero que mi mano se rellene con fichas nuevas del mazo.

Descripción detallada

El objetivo de esta historia de usuario es que una vez se viera decrementado el número de fichas de su mano después de que ejecutara determinados comandos durante su turno, pudiera volver a tener en su mano siete fichas para continuar sus jugadas.

4.27 Colocar más de una ficha por turno

ID: H_24	**Usuario:** Jugador
Nombre historia: Colocar más de una ficha por turno	
Prioridad: Media	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Alejandro Rivera León	
Descripción: Como jugador quiero poder colocar más de una ficha en el tablero cada turno para formar palabras y acumular puntuación.	

Descripción detallada

Anteriormente, debido a que no podían jugar varios jugadores de forma simultánea ni había una lógica de turnos, un jugador solamente podía poner una ficha por turno, con la inclusión de varios jugadores y diversos turnos esto tenía que ser modificado permitiendo así que el jugador pudiese colocar más de una ficha en el tablero antes de pasar al siguiente jugador

4.28 Jugar con otro jugador

ID: H_19	**Usuario:** Jugador
Nombre historia: Jugar con otro jugador	

Prioridad: Media	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como jugador quiero poder jugar otra partida con otra persona.	

Descripción detallada

Con esta historia de usuario se pretendía sentar las bases del juego multijugador. Para implementar esta funcionalidad se creó la clase AdminTurnos, cuya finalidad inicial era manejar el orden en el que los jugadores realizan acciones sobre el juego.

Posteriormente fue necesario reevaluar esta funcionalidad tanto para aplicarla a la GUI como para adaptarla a las máquinas y al multijugador en red:

ID: H_14	**Usuario:** Jugador
Nombre historia: Jugar con otro jugador	
Prioridad: Media	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable:	
Descripción: Como jugador quiero poder jugar una partida multijugador.	
Evolución	
Motivación: Esta historia se reabrió posteriormente para que esta funcionalidad se pasara a la nueva interfaz gráfica. Entra dentro de toda la funcionalidad de poder iniciar una partida desde la GUI y demás.	
Programador responsable: David Cruza Sesmero, María Cristina Salas, Guillermo García Patiño Lenza, Tania Romero Segura (estos tres últimos ayudaron con el desarrollo y a adaptar la funcionalidad a las máquinas y cliente/servidor)	

Estado: Finalizado	**Iteración asignada:** 4, 5, 6, 7

4.29 Saber quién empieza la partida

ID: H_26	**Usuario:** Jugador
Nombre historia: Saber quién empieza la partida	
Prioridad: Media	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero poder saber quién tiene el primer turno de la partida.	

Descripción detallada

Esta Historia de Usuario tiene como objetivo conseguir que el orden de turnos no sea siempre el mismo, es decir, que si un jugador se apunta primero a la partida no sea necesariamente el que empieza. Para ello se creó un sistema que elegía de manera aleatoria qué jugador va a comenzar el orden de turnos.

4.30 Tener en la pila de fichas dos comodines

ID: H_27	**Usuario:** Jugador
Nombre historia: Tener en la pila de fichas dos comodines	
Prioridad: Media	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: María Cristina Alameda Salas	

Descripción: Como jugador quiero tener dos comodines disponibles para jugarlos.

Descripción detallada

Consideramos durante el segundo sprint añadir un nuevo tipo de ficha que fuera un comodín de manera que tuviera funciones especiales.

4.31 Canjear comodines de la mano por letras

ID: H_28	**Usuario:** Jugador
Nombre historia: Canjear comodines de la mano por letras	
Prioridad: Media	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: María Cristina Alameda Salas	
Descripción: Como jugador quiero poder canjear los comodines de mi mano por letras para poder formar palabras.	

Descripción detallada

Gracias a esta función, un jugador puede utilizar los comodines que forman parte del mazo. El uso de un comodín consiste en que supla a la letra que el jugador quiera con puntuación 0. Ahora, al colocar este tipo de ficha, se le pide al usuario la letra a colocar.

4.32 Obtener puntos por mis jugadas

ID: H_23	**Usuario:** Jugador
Nombre historia: Obtener puntos por mis jugadas	
Prioridad: Media	**Estado:** Finalizado

Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero poder ganar puntos al poner una palabra en el tablero.	

Descripción detallada

Como es lógico, para todo juego es necesario tener un sistema para decidir quién gana la partida. En este caso la puntuación es ese sistema. Por lo tanto, se ha creado un sistema que decide cuántos puntos gana un jugador por cada palabra que pone basado en el juego original de Scrabble utilizando tanto la puntuación de las letras como las casillas especiales.

4.33 Poner la primera ficha de la partida en el centro

ID: H_24	**Usuario:** Jugador
Nombre historia: Poner la primera ficha de la partida en el centro	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como jugador quiero que la primera ficha de la partida sea colocada en el centro del tablero	

Descripción detallada

Con esta historia de usuario pretendíamos implementar en el juego ciertas reglas de negocio relacionadas con la colocación de fichas en el tablero. Más en concreto, buscábamos restringir las casillas del tablero donde se puede colocar fichas según un atributo llamado ‘disponible’. Al principio, la única casilla disponible será la central, y cuando se coloque una ficha en esa casilla, sus adyacentes se convertirán en disponibles también.

4.34 Guardar partida a medias

ID: H_25	**Usuario:** Jugador
Nombre historia: Guardar partida a medias	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Alejandro Rivera León	
Descripción: Como Jugador quiero poder guardar la partida en mitad de una partida	

Descripción detallada

Durante el sprint 3, antes de que se anunciase que teníamos que introducir la posibilidad de guardar y cargar partidas, se nos ocurrió la idea de implementar un sistema que permitiera guardar datos relevantes del usuario (su nombre, puntuación, monedas…) para posteriormente cargarlas y añadir una tabla de puntuaciones.

Posteriormente, se tomó la decisión de eliminar la Historia de Usuario que hacía referencia a guardar una partida desde la GUI para que pasar a formar parte de esta historia como evolución de la misma. El resultado fue el siguiente:

ID: H_18	**Usuario:** Jugador
Nombre historia: Guardar una partida	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Alejandro Rivera León	
Descripción: Como Jugador quiero poder guardar una partida para poder retomarla más adelante	

Evolución
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la GUI.
Programador responsable: Alejandro Rivera León
Estado: Finalizado **Iteración asignada:** 4

4.35 Cargar una partida a medias

ID: H_26	**Usuario:** Jugador
Nombre historia: Cargar una partida a medias	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Gema Blanco Núñez	
Descripción: Como jugador quiero que la primera ficha de la partida sea colocada en el centro del tablero	

Descripción detallada

El objetivo de esta historia de usuario nunca fue concretado. Inicialmente apareció de la mano con guardar una partida y se mantuvo hasta actualmente por si en algún momento quisiéramos recuperar el estado de una partida guardada.

Posteriormente, se tomó la decisión de eliminar la Historia de Usuario que hacía referencia a guardar una partida desde la GUI para que pasar a formar parte de esta historia como evolución de la misma. El resultado fue el siguiente:

ID: H_19	**Usuario:** Jugador
Nombre historia: Cargar una partida	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3

Programador responsable: Gema Blanco Núñez
Descripción: Como jugador quiero poder cargar una partida guardada con anterioridad
Evolución
Motivación: Esta historia se reabrió posteriormente para pasar su funcionalidad a la GUI
Programador responsable: Gema Blanco Núñez
Estado: Finalizado **Iteración asignada:** 4

4.36 Pedir si quieres New Game / Load Game

ID: H_27	**Usuario:** Jugador
Nombre historia: Pedir si quieres New Game / Load Game	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: David Cruza Sesmero	
Descripción: Como jugador puedo elegir cargar una partida a medias o bien crear una nueva la cual me mostrará un menú secundario en el que tendré como opciones añadir jugadores, eliminar jugadores, mostrar información del juego y comenzar la partida.	

Descripción detallada

Se añadió por primera vez un menú principal que te daba a elegir entre cargar una partida o iniciar una, de entre otras cosas, también se podían añadir y eliminar a jugadores en la partida. El trasiego de jugadores se realizaba mediante la manipulación de una lista de jugadores. En cuanto a la selección entre partida nueva o cargar una se realizaba pidiendo por consola cual se debía de ejecutar.

Posteriormente se decidió eliminar esta historia dado que podía incluirse en una nueva historia referente a inciar una partida como evolución de la misma. De esta manera tanto esta historia como la historia Iniciar una nueva partida desde la GUI podían verse dentro de un mismo entorno. Para ver la historia resultante véase la Historia de Usuario Iniciar una partida.

4.37 Verificar las palabras automáticamente al finalizar el turno

ID: H_28	**Usuario:** Jugador
Nombre historia: Verificar las palabras correctas automáticamente al finalizar el turno	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: María Cristina Alameda Salas	
Descripción: Como jugador quiero que se verifiquen las palabras correctas automáticamente al finalizar mi turno para que se sumen a mis puntos todas las correctas.	

Descripción detallada

Dentro de las funcionalidades básicas del juego estaba el verificar una palabra para poder aumentar los puntos de los jugadores. Fue un reto ya que había que comprobar que esta acción se realizaba perfectamente y sin errores. En el Sprint 2 se realizó una versión preliminar que verificaba una palabra dada las coordenadas de la palabra a verificar y además el jugador elegía cuando verificar una palabra. En esta versión, se realizó un cambio profundo sobre la función anterior y ya no elegía el jugador el momento ni la palabra, si no que se realizaba automáticamente al terminar el turno.

Posteriormente se decidió incluir esta historia dentro de la historia anterior Verificar una palabra como evolución de la misma.

4.38 No pasar de turno si he colocado alguna ficha que no forme palabra

ID: H_29	**Usuario:** Jugador
Nombre historia: No pasar de turno si he colocado alguna ficha que no forme parte de una palabra	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: María Cristina Alameda Salas	
Descripción: Como Jugador quiero que no pueda pasar de turno si he colocado alguna ficha que no forme palabra para que sean las partidas de acuerdo a las normas.	

Descripción detallada

Esta historia se realizó como una evolución de la primera, una vez que se hubiera implementado la anterior. No supuso gran dificultad.

4.39 Solo fichas al lado de fichas

ID: H_31	**Usuario:** Jugador
Nombre historia: Solo fichas al lado de fichas	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Guillermo García Patiño	
Descripción: Como jugador quiero que solamente se permita colocar una ficha en una casilla si la casilla tiene otra adyacente con una ficha	

Descripción detallada

Esta es otra historia de usuario que se encuentra destinada, más que a implementar un incremento en la funcionalidad, a implementar otra regla de negocio más acerca del modo en el que se colocan las fichas. Se trata de

implementar el manejo adecuado del atributo ‘disponible’ de las casillas para solamente permitir colocar una ficha en una casilla que tenga otras casillas adyacentes que tengan una ficha colocada.

Posteriormente se decidió eliminar esta historia dado que consideramos que no es realmente una Historia de Usuario sino una regla del juego. Como vimos que había muchas historias que hacían referencia a reglas del juego consideramos oportuno crear una Historia de Usuario nueva a la que llamamos Poder jugar de acuerdo a unas reglas.

4.40 Obtener puntos por las monedas sobrantes

ID: H_32	**Usuario:** Jugador
Nombre historia: Si finaliza la partida obtener puntos por las monedas sobrantes	
Prioridad: Baja	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero poder obtener puntos como recompensa por no haber gastado todas mis monedas al finalizar la partida.	

Descripción detallada

Pensamos que podría ser interesante premiar a los jugadores que no gastasen todas sus monedas al final del juego dado que su puntuación no se ha visto mejorada por usar ventajas. Al final, se decidió que para ello ototgaríamos dos puntos adicionales por cada moneda no gastada al final del juego.

4.41 Cambiar una ficha

ID: H_33	**Usuario:** Jugador
Nombre historia: Cambiar ficha	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3

Programador responsable: Guillermo García Patiño Lenza
Descripción: Como jugador quiero poder cambiar una de las fichas de mi mano por la siguiente ficha del mazo para poder hacer una mejor jugada

Descripción detallada

Durante el sprint 3, estábamos terminando de implementar las funcionalidades básicas del juego. Una de ellas era dar la posibilidad al jugador de cambiar una de las fichas de su mano por otra perteneciente al mazo. Para ello, tuvimos que cambiar la implementación del mazo, ya que antes no permitía volver a meter fichas en él.

Posteriormente se tomó la decisión de incluir como evolución de esta historia la Historia de Usuario Cambiar una ficha desde la GUI. El resultado fue el siguiente:

ID: H_23	**Usuario:** Jugador
Nombre historia: Cambiar ficha	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como jugador quiero poder cambiar una de las fichas de mi mano por la siguiente ficha del mazo para poder hacer una mejor jugada	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable: Guillermo García Patiño Lenza	
Estado: Finalizado	**Iteración asignada:** 4

4.42 Cambiar una ficha por un comodín a cambio de monedas

ID: H_34	**Usuario:** Jugador
Nombre historia: Cambiar una ficha por un comodín a cambio de monedas	
Prioridad: Baja	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero poder convertir una de mis fichas en un comodín a cambio de monedas	

Descripción detallada

Para este sprint se decidió incluir la posibilidad de comprar ventajas. Una de ellas es la de permitir al jugador cambiar una ficha de su mano a cambio de un comodín por el precio de 5 monedas del juego.

Posteriormente se incluyó dentro de esta Historia de Usuario la creada posteriormente para pasar su funcionalidad a la GUI (Cambiar una ficha por un comodín en la GUI). El resultado es el siguiente:

ID: H_24	**Usuario:** Jugador
Nombre historia: Cambiar una ficha por un comodín a cambio de monedas	
Prioridad: Baja	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Tania Romero Segura	

Descripción: Como jugador quiero poder convertir una de mis fichas en un comodín a cambio de monedas
Evolución
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.
Programador responsable: Tania Romero Segura
Estado: Finalizado **Iteración asignada:** 4

4.43 Saltar al jugador siguiente a cambio de monedas

ID: H_35	**Usuario:** Jugador
Nombre historia: Saltar al jugador siguiente a cambio de monedas	
Prioridad: Baja	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero poder saltar al jugador siguiente a cambio de monedas.	

Descripción detallada

Para este sprint se decidió incluir la posibilidad de comprar ventajas. Una de ellas es la de permitir al jugador saltar el turno del jugador siguiente por el precio de 3 monedas del juego.

Posteriormente se incluyó dentro de esta historia como parte de su evolución la Historia de Usuario Saltar a un jugador desde la GUI. El resultado fue el siguiente:

ID: H_25	**Usuario:** Jugador
Nombre historia: Saltar al jugador siguiente a cambio de monedas	
Prioridad: Baja	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero poder saltar al jugador siguiente a cambio de monedas.	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable: Tania Romero Segura	
Estado: Finalizado	**Iteración asignada:** 4

4.44 Invertir el orden de jugadores a cambio de monedas

ID: H_36	**Usuario:** Jugador
Nombre historia: Revertir el orden de jugadores a cambio de monedas	
Prioridad: Baja	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero poder cambiar el orden de jugadores a cambio de monedas	

Descripción detallada

Para este sprint se decidió incluir la posibilidad de comprar ventajas. Una de ellas es la de permitir al jugador invertir el orden de jugadores por el precio de 5 monedas del juego.

Posteriormente se incluyó dentro de esta historia la Historia de Usuario Invertir sentido en la GUI. De esta manera los cambios realizados en la segunda quedan reflejados como evolución de la primera.

ID: H_26	**Usuario:** Jugador
Nombre historia: Invertir el orden de jugadores a cambio de monedas	
Prioridad: Baja	**Estado:** Finalizado
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Tania Romero Segura	
Descripción: Como jugador quiero poder cambiar el orden de jugadores a cambio de monedas	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable: Tania Romero Segura	
Estado: Finalizado	**Iteración asignada:** 4

4.45 Disponer de casillas especiales

ID: H_23	**Usuario:** Jugador
Nombre historia: Disponer de casillas especiales	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 5	**Iteración asignada:** 2

Programador responsable: Tania Romero Segura
Descripción: Como jugador quiero poder aprovechar las casillas especiales para obtener más puntos por mis jugadas

Descripción detallada

El juego de Scrabble se caracteriza precisamente en gran medida por el tablero y las casillas especiales que tiene. Esta historia tiene como objetivo introducir estas casillas en nuestro juego. Se decidió incluir todas las casillas originales del juego (doble por letra, doble por palabra, triple por letra, triple por palabra).

4.46 Poder jugar con jugadores controlados por la IA

ID: H_37	**Usuario:** Jugador
Nombre historia: Poder jugar con jugadores controlados por IA	
Prioridad: Alta	**Estado:**
Puntos estimados: 4	**Iteración asignada:** 4
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder jugar contra un jugador controlado por la máquina en tres niveles de dificultad (fácil, medio, difícil)	

Descripción detallada

El profesor nos pidió que incluyéramos jugadores automáticos. Como respuesta creamos otro tipo de jugador al que llamamos máquina que posee distintas estrategias para jugar su turno dependiendo del nivel de dificultad. Estas máquinas evalúan qué palabras pueden formar con sus letras y luego coloca una en el tablero siguiendo un orden de prioridad acorde al nivel de dificultad.

Posteriormente se incluyó en esta Historia de Usuario la evolución sufrida a causa de una refactorización el sprint siguiente. El resultado es el siguiente:

ID: H_27	**Usuario:** Jugador
Nombre historia: Poder jugar contra la máquina	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 4	**Iteración asignada:** 5
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder jugar contra un jugador controlado por la máquina en tres niveles de dificultad (fácil, medio, difícil)	
Evolución	
Motivación: Esta historia se reabrió posteriormente para cambiar la implementación	
Programador responsable: Tania Romero Segura	
Estado: Finalizado	**Iteración asignada:** 6, 7

4.47 Poder ver la mano en la GUI

ID: H_38	**Usuario:** Jugador
Nombre historia: Poder ver mi conjunto de fichas en la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: María Cristina Alameda Salas	
Descripción: Como Jugador quiero poder ver las fichas que tengo en mi mano desde la GUI	

Descripción detallada

Realmente supone una evolución de la historia de Saber mi conjunto de fichas actual. La diferencia con la anterior es la vista en que se muestra, antes por consola y ahora en GUI. Gracias a esta historia, se mostraban las fichas en la mano del jugador en la GUI

3.48 Poder ver el tablero en la GUI

ID: H_39	**Usuario:** Jugador
Nombre historia: Poder ver el tablero en la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 5	**Iteración asignada:** 4
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como Jugador quiero poder ver el tablero, y las fichas que contiene, en la GUI	

Descripción detallada

Cuando empezamos a programar la GUI del juego, una de las principales cuestiones a la que nos enfrentamos fue la representación del tablero y las fichas en él. Realmente, esta historia de usuario representa una evolución de la historia de usuario anterior que se refiere a la representación del tablero por consola.

Posteriormente esta Historia de Usuario pasó a formar parte de la Historia de Usuario Ver el tablero como evolución de la misma.

3.49 Poder ver los turnos en la GUI

ID: H_40	**Usuario:** Jugador
Nombre historia: Poder ver los turnos en la GUI	
Prioridad: Media	**Estado:**
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Alejandro Rivera León	

Descripción: Como Jugador quiero saber cual es el estado de los turnos en la GUI

Descripción detallada

Debido a la inclusión de la GUI para facilitar la interacción del usuario con el juego se tuvo que adaptar la forma en la que se le mostrarían los turnos para permitir que esta información se viese reflejada también en la GUI.

Al igual que el resto de Historias incorporadas en el momento en que se creó la GUI esta historia pasó a formar parte de la historia Conocer turnos de la partida como evolución de la misma.

4.50 Poder ver los puntos de los Jugadores en la GUI

ID: H_41	**Usuario:** Jugador
Nombre historia: Poder ver los puntos de los Jugadores en la GUI	
Prioridad: Media	**Estado:**
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Gema Blanco Núñez	
Descripción: Como Jugador quiero saber los puntos que tengo en la GUI	

Descripción detallada

Esta historia de usuario tiene como objetivo mostrar por la GUI los puntos que posee el jugador. Se trata de una evolución de la historia de usuario por la cual se obtienen puntos adaptada a la interfaz gráfica.

4.51 Poder ver las monedas de los jugadores en la GUI

ID: H_42	**Usuario:** Jugador
Nombre historia: Poder ver las monedas de los jugadores en la GUI	
Prioridad: Media	**Estado:**
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Gema Blanco Núñez	
Descripción: Como Jugador quiero saber las monedas que tengo en cualquier momento desde la GUI	

Descripción detallada

Esta historia de usuario tiene como objetivo mostrar por la GUI las monedas que posee el jugador. Se trata de una evolución de una historia de usuario antigua la cual se llamaba “Poder almacenar mi puntuación” adaptada a la interfaz gráfica.

4.52 Poder cargar una partida desde la GUI

ID: H_43	**Usuario:** Jugador
Nombre historia: Poder cargar una partida desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 4	**Iteración asignada:** 4
Programador responsable: Gema Blanco Núñez	
Descripción: Como Jugador quiero poder cargar una partida desde la GUI	

Descripción detallada

Esta historia de usuario tiene como finalidad permitir cargar una partida desde un botón de la GUI. Se trata de una evolución de una historia anterior la cual permite cargar una partida previamente guardada.

4.53 Poder guardar una partida desde la GUI

ID: H_44	**Usuario:** Jugador
Nombre historia: Poder guardar una partida desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Alejandro Rivera León	
Descripción: Como Jugador quiero poder guardar partida desde la GUI.	

Descripción detallada

Durante este sprint se nos comunicó que debíamos incluir en nuestro juego un modo de poder guardar y cargar partidas de forma que estas pudieran ser retomadas en otro momento.

4.54 Colocar una ficha en el tablero desde la GUI

ID: H_45	**Usuario:** Jugador
Nombre historia: Poder colocar una ficha en el tablero desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: María Cristina Alameda Salas	

Descripción: Como Jugador quiero poder colocar una ficha en el tablero desde la GUI

Descripción detallada

Esta historia corresponde a una evolución de interfaz del juego. En este caso su predecesora es la historia colocar ficha en el tablero, una de las historias de máxima prioridad.

4.55 Poder cambiar una ficha de mi mano desde la GUI

ID: H_56	**Usuario:** Jugador
Nombre historia: Poder cambiar una ficha de mi mano desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como Jugador quiero poder cambiar una ficha de mi mano desde la GUI	

Descripción detallada

Esta historia de usuario se presenta como una evolución de la anterior historia llamada “Poder Cambiar Ficha”. Esta historia anterior estaba destinada a incluir la funcionalidad en la vista por consola, mientras que la actual pretende implementar la funcionalidad en la vista de la GUI a través de un botón

4.56 Iniciar una nueva partida desde la GUI

ID: H_47	**Usuario:** Jugador
Nombre historia: Iniciar una nueva partida desde la GUI	
Prioridad: Alta	**Estado: Finalizado**

Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder iniciar una partida nueva desde la GUI	

Descripción detallada

En este sprint comenzamos a desarrollar la interfaz gráfica de nuestra aplicación así que se creó un menú el cual mostraba las opciones de crear una partida y cargarla. Crear una partida te llevaba a un submenú donde podías iniciar la partida, añadir jugadores o eliminarlos, mientras que con cargar partida directamente cargaba la partida sin opción a eliminar jugadores o añadirlos a la misma.

4.57 Introducir jugadores y eliminar desde la GUI

ID: H_48	**Usuario:** Jugador
Nombre historia: Introducir jugadores y eliminar desde la GUI	
Prioridad: Alta	**Estado: Finalizado**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder introducir los jugadores y sus nicks desde la GUI	

Descripción detallada

Se introducían en un principio mediante un campo de texto y una label donde se iban mostrando la lista de jugadores en partida. Tras introducirlos todos, te daba la posibilidad de eliminar a jugadores, al ser tan cerrado en cuanto a funcionalidad y no poder eliminar a los jugadores a la vez que podías añadirlos se acabó modificando este código.

4.59 Poder pasar turno desde la GUI

ID: H_49	**Usuario:** Jugador
Nombre historia: Poder pasar el turno en la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como Jugador quiero poder pasar mi turno desde la GUI	

Descripción detallada

De nuevo, esta historia de usuario aparece como una evolución de su anterior correspondiente a la vista por consola. Presentó algunas dificultades, y su implementación trajo consigo un refactor al modelo.

Más adelante se incluyó como evolución de esta historia poder pasar de turno desde la GUI:

ID: H_11	**Usuario:** Jugador
Nombre historia: Pasar de turno	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Gema Blanco Núñez	
Descripción: Como jugador quiero pasar de turno para terminar mi jugada.	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar esta funcionalidad a la GUI	
Programador responsable: Guillermo García Patiño Lenza	

Estado: Finalizado	**Iteración asignada:** 5

4.60 Poder saltar jugador en la GUI

ID: H_50	**Usuario:** Jugador
Nombre historia: Poder saltar jugador en la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder comprar una ventaja para saltar el turno a un jugador	

Descripción detallada

Durante este sprint se decidió incluir una nueva interfaz gráfica, por lo tanto era necesario revisar la funcionalidad para que estuviera presente en la GUI. Se creó una historia de usuario separada aunque en el último backlog se decidió que debería tratarse como una evolución de la previa. Como cuando se trabajó sobre esto estaba como una historia separada la tratamos aquí de igual manera.

Para pasar esta funcionalidad no era necesario tocar la funcionalidad de saltar a un pero sí era necesario proporcionarle al usuario una forma de pedirle al sistema que se ejecutara esta funcionalidad. Por eso, se creó una ventana de compra a la que se podía acceder a través de un botón. Desde esa ventana de compra se puede seleccionar qué ventaja se desea comprar y una vez confirmada la compra el sistema ejecuta la funcionalidad igual que hacía previamente.

Posteriormente esta Historia de Usuario pasó a formar parte de la evolución de su predecesora: Saltar al jugador siguiente a cambio de monedas.

4.61 Poder invertir el sentido en la GUI

ID: H_51	**Usuario:** Jugador
Nombre historia: Poder invertir sentido en la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder comprar una ventaja para poder invertir el sentido desde la GUI	

Descripción detallada

Durante este sprint se decidió incluir una nueva interfaz gráfica, por lo tanto era necesario revisar la funcionalidad para que estuviera presente en la GUI. Se creó una historia de usuario separada aunque en el último backlog se decidió que debería tratarse como una evolución de la previa. Como cuando se trabajó sobre esto estaba como una historia separada la tratamos aquí de igual manera.

Para pasar esta funcionalidad no era necesario tocar la funcionalidad de invertir sentido pero sí era necesario proporcionarle al usuario una forma de pedirle al sistema que se ejecutara esta funcionalidad. Por eso, se creó una ventana de compra a la que se podía acceder a través de un botón. Desde esa ventana de compra se puede seleccionar qué ventaja se desea comprar y una vez confirmada la compra el sistema ejecuta la funcionalidad igual que hacía previamente.

Posteriormente esta Historia de Usuario pasó a formar parte de la evolución de su predecesora: Invertir el orden de jugadores a cambio de monedas.

4.62 Poder comprar un comodín en la GUI

ID: H_52	**Usuario:** Jugador
Nombre historia: Poder comprar un comodín en la GUI	

Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder ver las fichas que tengo en mi mano desde la GUI	

Descripción detallada

Durante este sprint se decidió incluir una nueva interfaz gráfica, por lo tanto era necesario revisar la funcionalidad para que estuviera presente en la GUI. Se creó una historia de usuario separada aunque en el último backlog se decidió que debería tratarse como una evolución de la previa. Como cuando se trabajó sobre esto estaba como una historia separada la tratamos aquí de igual manera.

Para pasar esta funcionalidad no era necesario tocar la funcionalidad de comprar un comodín pero sí era necesario proporcionarle al usuario una forma de pedirle al sistema que se ejecutara esta funcionalidad. Por eso, se creó una ventana de compra a la que se podía acceder a través de un botón. Desde esa ventana de compra se puede seleccionar qué ventaja se desea comprar y una vez confirmada la compra el sistema ejecuta la funcionalidad igual que hacía previamente.

Posteriormente esta Historia de Usuario pasó a formar parte de la evolución de su predecesora: Cambiar una ficha por un comodín a cambio de monedas

4.63 Poder acceder a las instrucciones de la GUI

ID: H_53	**Usuario:** Jugador
Nombre historia: Poder acceder a las instrucciones de la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: David Cruza Sesmero	

Descripción: Como Jugador quiero poder ver las instrucciones del juego.

Descripción detallada

Se añadió la opción de ver las instrucciones del juego desde el menú principal y desde la propia partida mediante un JMenuBar. En el siguiente sprint también se añadió otra opción que explicaba cómo jugar (colocar ficha, robar ficha, pasar turno ,etc ...)

4.64 Poder jugar en red con otros jugadores

ID: H_54	**Usuario:** Jugador
Nombre historia: Poder jugar en red con otros jugadores	
Prioridad: Alta	**Estado:**
Puntos estimados: 5	**Iteración asignada:** 6
Programador responsable: María Cristina Alameda Salas	
Descripción: Como Jugador quiero poder jugar una partida con otros jugadore en red.	

Descripción detallada

Durante el cuarto sprint se nos comunicó que los jugadores deberían poder jugar con otros jugadores en red.

4.65 Poder usar comodines

ID: H_16	**Usuario:** Jugador
Nombre historia: Poder usar comodines	
Prioridad: Media	**Estado:** Finalizado
Puntos estimados:	**Iteración asignada:** 2
Programador responsable: María Cristina Alameda Salas	

Descripción: Como jugador quiero poder usar los comodines en mis jugadas

Descripción detallada

Esta Historia de Usuario surge como sustituta de las dos historias creadas previamente: Tener en la pila de fichas dos comodines y Canjear comodines de la mano por letras. Esto es porque la primera no puede considerarse como Historia de Usuario ya que en realidad es parte de lo que consideramos como reglas del juego.

4.66 Poder jugar de acuerdo a unas reglas

ID: H_21	**Usuario:** Jugador
Nombre historia: Poder jugar de acuerdo a unas reglas	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 4	**Iteración asignada:** Todas
Programador responsable: María Cristina Alameda Salas, Guillermo García Patiño Lenza, Tania Romero Segura, Alejandro Rivera León, Gema Blanco Núñez	
Descripción: Como Jugador quiero poder jugar con una serie de reglas que me impidan a mi y a mis oponentes hacer trampas.	

Descripción detallada

Esta Historia de Usuario fue creada tras notar que teníamos muchas “historias” que realmente no eran historias sino reglas del juego. Por ello decidimos que sería mejor eliminar esas historias y sustituirlas por esta nueva historia. Las funcionalidad que se aplicaron son las mismas pero englobadas en una Historia de Usuario.

4.67 Iniciar una nueva partida

ID: H_28	**Usuario:** Jugador
Nombre historia: Iniciar una nueva partida	
Prioridad: Alta	**Estado:** Terminado
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder iniciar una partida nueva	
Evolución	
Motivación: Esta historia se reabrió posteriormente para pasar la funcionalidad a la interfaz gráfica.	
Programador responsable: David Cruza Sesmero	
Estado: Finalizado	**Iteración asignada:** 4, 5, 6

Descripción detallada

Durante el final del proyecto se decidió que dado que ha habido varias historias referentes al inicio de una partida se decidió que sería mejor sustituirlas por una nueva Historia de Usuario para dejar más clara la funcionalidad y poner los cambios como parte de su evolución.

5. SPRINT **REVIEWS**

Las Sprint Reviews se llevan a cabo mediante una reunión de una hora en la que se inspecciona el incremento y se adapta el Product Backlog si fuese necesario. En esta reunión el equipo presenta lo que se hizo durante el sprint. El scrum Master debe asegurarse de que este evento se lleve a cabo y que los asistentes entiendan su finalidad.

ELEMENTOS:

- Asiste todo el **Equipo Scrum** (el equipo de desarrollo, el Scrum Master y el Product Owner).
- El Product Owner explica que elementos del Product Backlog se han terminado y cuáles no.

- El equipo de desarrollo habla acerca de los problemas que aparecieron en el sprint, cómo se resolvieron y las cosas que fueron bien.

A continuación se presentan todos los Sprint Reviews que hemos llevado a cabo durante el desarrollo del proyecto.

➔ Sprint 1:

- **¿Qué hemos hecho?**

Hemos conseguido implementar todas las historias de usuario que habíamos asignado al primer sprint:

- **Generar Mazo de Fichas:**

Se ha creado una clase GeneradorMazo que se encarga de generar un vector de fichas desde un archivo JSON que contiene para cada letra: las veces que aparece y los puntos que vale una ficha con esa letra.

Después las introduce en un vector que mezcla aleatoriamente y las pasa al mazo del juego en forma de pila.

- **Ver el tablero:**

La clase tablero tiene un toString que cogimos de la práctica 1 de TP porque saca un tablero por pantalla similar al que necesitamos en este proyecto. Ese toString genera una matriz de los toStrings de cada ficha que hay en el tablero y les da el formato de una casilla cuadrada que muestra por pantalla.

- **Colocar Fichas en el Tablero:**

Se ha implementado mediante un comando que, al ejecutarse, busca la letra introducida como argumento en un array de fichas que simula el del jugador, y si la encuentra, comprueba si hay una ficha puesta ya en la casilla indicada en el tablero, y si no la hay, la coloca.

- **Quitar Fichas del Tablero:**

Se ha implementado mediante un comando cuya ejecución consiste, de momento, en acceder a la casilla que se indica en el comando, y si encuentra una ficha en esa casilla, la quita del tablero. En un futuro este comando debería añadir la ficha que se ha quitado a la mano del jugador que ejecuta el tablero.

- **Saber el número de fichas que quedan por robar:**

  Se añade al toString del Game el número de fichas que quedan en el mazo, valor que se pide a la clase Mazo.

- **Rellenar Mano:**

  Se ha implementado mediante una función en el Game que al final de cada turno (ahora mismo el turno acaba al poner una ficha), comprueba cuántas fichas tienes en tu mano, y añade nuevas fichas a la mano procedentes del mazo hasta que tengas 7 fichas.

- **Salir de la Partida:**

  El controller en cada ciclo del juego, comprueba una variable booleana que indica si la partida ha acabado o no. Esa variable booleana se modifica mediante la ejecución de un comando “Salir de la partida”.

➔ Sprint 2:

- **Saber mi conjunto de fichas actual**

  Para saber la mano que tiene el jugador que tiene el turno, se ha introducido en el toString de la clase jugador, que muestra por pantalla la mano que en ese momento tiene dicho jugador.

  Ese toString es llamado por la clase Game cada vez que se imprime el tablero. (El atributo Player de la clase Game cambia cada turno).

- **Saber cuántos puntos vale cada ficha**

  Para saber cuántos puntos vale cada ficha, primero se asignan los puntos al mazo.json. Cuando se crean las fichas se guarda el carácter y los puntos correspondientes a la respectiva letra. Solo se altera el método toString de Ficha para poder conocer los puntos, que ahora además de mostrarme la letra, también muestra su puntuación.

- **Introducir nicks**

  Se ha implementado la función de introducir jugadores dentro de la clase Controller, creando una lista de jugadores a la cual vamos añadiendo participantes introduciendo sus nicks.

- **Conocer turnos de la partida**

  Para conocer los turnos de la partida se ha implementado el método toString en la clase AdminTurnos, este método es llamado en la clase Turno, concretamente en el método Run. El método toString se ha implementado usando un bucle y una lista circular que es recorrida dentro de este, al finalizar el bucle se muestra el orden en el que los jugadores realizarán sus turnos por pantalla.

- **Pasar de turno**

  Se ha creado un comando nuevo para pasar de turno en la clase ComandoPasarTurno que implementa el método parse para verificar el comando y el método execute, que devuelve un booleano para indicar a la clase Turno que el jugador pasará su turno al siguiente.

- **Colocar más de una ficha por turno**

  Aprovechando que la la introducción de comandos se encuentra en la clase Turno, lo que hacemos es darle libertad al usuario para poner el número de fichas que quiera hasta que seleccione el comando pasar turno.

- **Jugar con otro jugador**

  Se ha creado una nueva clase llamada AdminTurnos que se encarga de ir asignando turnos a cada jugador en el orden establecido. Además, se ha creado una clase turno que encapsula la información de lo que sucede en el turno de un jugador y le dice al Game sobre qué jugador tiene que efectuar los cambios En esta clase turno es donde se encuentra el bucle de introducir comandos que antes se encontraba en la clase Controller.

- **Saber quien empieza la partida**

  Se ha creado un método en la clase AdminTurnos que utiliza la función de math que te da un valor aleatorio entre dos números. En este caso te da un valor entre 0 y el número de jugadores menos uno, este valor se corresponde con la posición en la lista de jugadores que ocupa el primer jugador.

- **Verificar si una palabra es válida**

  Se ha creado un comando verificar al que se le pasa la posición de la primera ficha de la palabra y la posición de la última. El execute de este comando comprueba que las posiciones son válidas y obtiene la palabra contenida a partir de las letras de cada una de las fichas entre ambas

posiciones. Además, se ha creado un diccionario con una lista de string y un método iniDiccionario que lee las palabras de un documento de texto y las almacena en la lista. De esta manera, en el execute también se comprueba que el diccionario contiene la palabra obtenida. Si la palabra está en el diccionario se confirma la palabra y pasa el turno. Si no, continúa el turno del jugador.

- **Obtener puntos por mis jugadas**

  Se ha modificado el execute del comando verificar de forma que al mismo tiempo que se obtienen las letras de la palabra se van sumando los puntos. Al final, si la palabra está en el diccionario se suman los puntos al jugador.

- **Obtener bonificaciones en función de las casillas que ocupa la palabra que pongo**

  Se han creado las clases correspondientes a las casillas especiales (casillas roja, naranja, morada y azul) que extienden a casilla y la clase casilla normal que también extiende a casilla. Estas casillas tienen un atributo multiplicador que determina qué tipo de bonificación se obtiene con las puntuaciones de cada una (1 = casilla normal | 2 = doble por letra | 3 = triple por letra | 4 = doble por palabra | 5 = triple por palabra).

  Se ha modificado la clase Tablero de manera que cuando se inicializa el tablero se crea con la disposición de casillas correspondiente al juego del Scrabble.

  Se ha modificado el método execute del comando verificar de manera que al sumar los puntos para las palabras se tenga en cuenta la casilla en la que está cada ficha. Si las casillas son de doble por letra o triple por letra se multiplican los puntos obtenidos por la ficha. Si son de doble por palabra o triple por palabra se acumula el multiplicador en una variable y luego se multiplican los puntos totales al final.

- **Tener en la pila de fichas dos comodines**

  Se ha añadido en el archivo mazo.json, un nuevo elemento, el comodín. Este tiene como letra el carácter ‘*’ y como puntos 0, según las normas oficiales del Scrabble.

- **Canjear comodines de la mano por letras**

  Para canjear los comodines se jugador solo tiene que colocar la ficha como cualquier otra. Se han realizado unos cambios en el método colocarFicha. Se han añadido, que cuando un jugador quiere colocar un

comodín (carácter ‘*’), se le solicita la letra y se crea una nueva ficha que es la que se añade al tablero.

- **Guardar partida**

  Para guardar el estado de la partida se ha creado un método save en la clase AdminTurnos el cual es llamado desde la clase Controller una vez finaliza el bucle del juego. El método save mediante un bucle recorre la lista de jugadores guardando sus nicks, puntuaciones y monedas en el fichero que previamente el usuario elija.

- **Quitar una ficha del tablero**

  Para quitar una ficha del tablero, se ha incluido en el game una lista de pares que indican las coordenadas de las fichas puestas por un jugador durante su turno. Cuando se ejecuta el comando quitar ficha sobre una de las casillas del tablero, se comprueba, mirando en esa lista que la ficha ha sido colocada durante ese turno y si es así, se devuelve a la mano del jugador.

  Esta lista de coordenadas se vacía al acabar cada turno.

- **Eliminar Jugadores:**

  Se ha implementado la función de eliminar jugadores dentro de la clase Controller. Esta función muestra una lista numerada de los jugadores, el usuario que quiera eliminar a un jugador solo debe introducir la posición en la que se encuentra dicho jugador pudiendo repetir esta operación las veces que vea necesario.

- **Mostrar información de los jugadores en cada turno:**

  Este método está implementado dentro de la clase AdminTurnos, concretamente dentro de siguiente turno, esta función muestra los nicks de los jugadores de la partida y su puntuación en cada momento.

➔ Sprint 3:

Consideramos que todo lo que planeamos hacer para este sprint se encuentra implementado y funciona en la mayoría de casos. Como caso a destacar, el único fallo visible que es realmente significativo es que hay un error en la carga de una partida a medias, además de un bug que sucede al quitar un comodín del tablero.

Para el siguiente Sprint deberíamos considerar arreglar estos dos errores.

Tanto la productividad como el esfuerzo del equipo no ha disminuido respecto a sprints anteriores a pesar de las circunstancias en las que nos encontramos y la nueva adaptación a la docencia online.

Además, la carga de trabajo para este sprint ha sido adecuada, pues hemos logrado completar todo el trabajo que habíamos planificado, a excepción de un par de diagramas que quedan por incluir.

Consideramos que todas las historias de usuario se encuentran implementadas en el proyecto en un grado suficiente como para incluirlas en el product backlog.

En cuanto a decisiones de diseño, para mantener encapsulada la información sobre los turnos y las acciones que han sucedido durante ellos y que condicionan lo que puede o no suceder después, hemos cambiado la forma en la que se ejecutan los comandos. Ahora también tienen un método abstracto *posible(Turno t)* y un método *perform(Turno t, Game game)*. El segundo método es común para todos los comandos y consiste en calcular un booleano ejecutando el método "posible" del comando, que comprueba a través del turno si el comando puede o no ejecutarse conforme a lo que ha sucedido anteriormente, y en función de ese booleano llamar al método execute de los comandos tal y como estaba en iteraciones anteriores, para finalmente registrar los cambios que ha efectuado la ejecución del comando en el Turno al que pertenece por medio de la ejecución del método *addCambios(Turno t)*.

Por otro lado, se ha decidido implementar el patrón Memento para facilitar el guardado y el cargado de las partidas. Se ha decidido que un Memento es un JSONObject que almacena información del estado del programa, y que estos se crean por medio de una cadena de llamadas que comienza en el *AdminTurnos* por medio de la llamada al método *addCambios* del comando Guardar.

➔ Sprint 4:

Durante este sprint no hemos completado todo lo que habíamos planificado. Se ha quedado pendiente la creación del menú principal que permite comenzar una nueva partida o cargar una partida anterior.

Por otro lado, siguen presentes uno de los fallos que destacamos en el sprint review anterior que se da al cargar una partida. Deberíamos solucionarlos para poder hacer pruebas correctamente con la GUI.

En cuanto a la productividad y el esfuerzo del equipo de desarrollo, consideramos que hemos hecho un esfuerzo adecuado durante el sprint, lo que nos ha permitido cumplir la mayoría de los objetivos planificados.

En lo que a organización se refiere, hemos podido dividir todas las tareas adecuadamente excepto la correspondiente a la creación de las IA, que se podía haber dividido en varias subtareas dentro del sprint para facilitar el seguimiento de su desarrollo.

➔ Sprint 5:

Durante este sprint no hemos conseguido cumplir todos los objetivos que habíamos planteado. Esto ha sucedido porque, en mitad del sprint nos hemos dado cuenta de que era necesaria una refactorización del código perteneciente a los jugadores controlados por la máquina para hacer el diseño más uniforme y correcto. Debido a esto, no hemos podido avanzar en ciertos aspectos que teníamos planteados ni en ciertas correcciones al modelo que requerían un correcto funcionamiento por parte de todos los jugadores.

Por otro lado, no hemos sido capaces de hacer la refactorización mencionada debido a la complejidad del problema, nuestra incapacidad técnica y al poco tiempo del que disponíamos para hacerla.

Además, se han refactorizado los observadores tal y como se había planteado en el sprint anterior para reducir las implementaciones vacías de las clases observadoras del modelo. (Más explicaciones en el documento Nuevos Observadores del Sprint 5).

También hemos cambiado la forma en la que están hechas las casillas pasando todas las clases que antes heredaban de ‘Casilla’ a ser una sola que calcula los puntos gracias a un atributo que se le pasa en el constructor.

Se ha añadido una ventana que posibilita a los jugadores humanos la compra de ventajas dentro del juego.

Se ha resuelto un problema relativo al uso del MVC que provocaba que las fichas desaparecieran de la mano cuando había varias fichas con

la misma letra en la mano y se colocaba una de ellas pasando el id de la ficha que se quiere colocar al comando correspondiente si se ejecuta desde la GUI.

En cuanto a la productividad del equipo, varios miembros han conseguido completar sus tareas, mientras que otros, debido a los problemas ya mencionados, no.

En general, durante este sprint no hemos conseguido añadir casi ningún incremento significativo, ya que hemos necesitado dedicarlo a recuperar la deuda técnica adquirida durante los anteriores.

➔ Sprint 6:

Durante este sprint no hemos conseguido cumplir todos los **objetivos** que habíamos planteado. Esto se debe a varias razones. En lo que a iniciar una nueva partida desde la GUI se refiere, no se ha podido finalizar dado que se ha presentado un error que no se ha sabido resolver antes de la finalización del sprint. Por otra parte, durante este sprint una parte del trabajo se ha centrado en la refactorización del diseño y algunos de estos elementos no se han podido incluir en la rama principal del proyecto antes de la finalización del sprint, dado que es necesario tanto terminar algunas como someter a otras a pruebas antes de su inclusión.

Otra parte del trabajo durante este sprint, ha sido la inclusión de **multijugador en red**. Con estas modificaciones, cada uno de los jugadores que participan en la partida puede tener su propia vista. Para incorporar a nuestro proyecto esta parte, hemos seguido la arquitectura Cliente-Servidor. En nuestro caso, hemos considerado el sistema de la siguiente forma:

El servidor es un proveedor de servicios y atiende a las demandas de los clientes, por tanto en él reside el modelo de nuestro proyecto (la lógica del juego).El cliente es un demandante de servicios, en él reside la vista (la interfaz del juego).La comunicación entre el cliente y el servidor la hemos conseguido gracias al uso de sockets.

Además, se han añadido una serie de clases a la vista que nos permiten poder hacer login con nuestro nick y otra para poder ver los jugadores conectados al juego en tiempo real. (para más información sobre diseño ver documento “Multijugador en red”).

Por otro lado, durante este sprint también se ha **refactorizado el Modelo**, puesto que al incluir la GUI era necesario cambiar el

comportamiento del Modelo para que dejara de estar guiado por bucles, y pasará a estar dirigido por los eventos que se producían en la GUI. Como consecuencia, se han eliminado los bucles que pedían comandos por la consola, y los que hacían avanzar el juego asignando turnos a los jugadores uno por uno. Toda la información referida a esta refactorización se encuentra en el documento 'RefactorModelo'.

La **refactorización de la IA** ha permitido que el comportamiento de la máquina ahora sea más similar al de un jugador humano. Para facilitar este comportamiento, se ha decidido crear una nueva clase Diccionario con una estructura de datos trie (para más información leer el documento "refactorizaciónIA" almacenado en la carpeta del Sprint 6). Como consecuencia, se ha modificado el sistema por el que se diferencian los niveles de dificultad y, para mejorar el diseño original, se han modificado también tanto las estrategias como las clases Integrante, Jugador y Maquina (de nuevo, para más información leer el documento "refactoriaciónIA"). Queda pendiente la incorporación de este nuevo sistema a la rama principal del proyecto.

En cuanto a la **parte visual de la aplicación**, se ha mejorado la claridad del código y se ha dado un gran salto en cuanto a la calidad de la GUI.

Respecto a la ventana principal (el tablero de juego y los diversos paneles que lo componen) se han cambiado tanto iconos, colores y formatos para mejorar la visibilidad de la información y se ha mejorado el reescalado de dichos paneles permitiendo así una mayor adaptabilidad.

Dicho tratamiento se ha extendido también a los menús con los que cuenta nuestra aplicación (Menú principal, Lobby.. etc) donde se ha mejorado notablemente la presentación, se ha ajustado el reescalado y se ha adecuado la paleta de colores respecto a la que usa la ventana principal (el tablero de juego).

En cuanto a **JUnit,** aunque estamos a unas alturas ya bastante avanzadas del proyecto, las pruebas unitarias nos han sido de gran utilidad. La razón por la que no empezamos a generar pruebas desde un principio ha sido la falta de conocimiento sobre JUnit. Estas pruebas abarcan las clases Tablero, ComandoPasarTurno, ComandoColocarFicha, ComandoInvertirSentido y GeneradorMazo.

La idea de cara al próximo sprint es seguir generando pruebas unitarias hasta tener probadas por lo menos las partes más críticas de la aplicación y la funcionalidad básica del juego. También intentaremos crear

nuevos casos de uso para las pruebas que sean superadas sin errores con el objetivo de encontrar nuevos fallos en el juego.

En cuanto a la **creación de jugadores y máquinas**, se pueden añadir y eliminar a los mismos hasta llegar a un máximo de cuatro componentes por partida, obligando a la partida a que al menos haya un jugador, por otra parte, al crear una máquina podremos seleccionar su dificultad mediante un diálogo el cual nos mostrará las tres dificultades disponibles.También se ha añadido la posibilidad de ver los comandos que se pueden utilizar así como las instrucciones del juego mediante un JMenuBar al que se puede acceder en todo momento.

➔ Sprint 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 *RefactorizaciónIA*.

Evolución de las Sprint Reviews

Uno de los errores más importantes que tuvimos respecto a las Sprint Reviews tiene que ver con la confusión sobre el objetivo de las mismas. No fue hasta el Sprint 3 que empezamos a analizar la productividad y el esfuerzo invertido en el sprint correspondiente. Hasta ese momento nos limitábamos a describir las historias de usuario desarrolladas durante el sprint. A partir del Sprint 3 corregimos nuestros errores gracias al feedback recibido por parte del profesor de la asignatura de ISII Gonzalo Méndez.

Respecto a las historias de usuario desarrolladas en cada sprint, todas ellas se corresponden con las historias que conformaban el Sprint Backlog. Sin embargo, las Sprint Reviews deben dar un Product Backlog actualizado. En nuestro caso no fuimos refinando el Product Backlog a medida que se completaban los sprint debido a que no lo consideramos necesario en el momento o no teníamos una perspectiva lo suficientemente clara. A alturas más avanzadas del proyecto, con una visión más general del mismo, nos hemos dado cuenta de que algunas historias de usuario del Product Backlog no tenían la suficiente relevancia, mientras

que otras debían ser actualizadas debido a la introducción de la GUI, entre otros motivos.

A pesar de no estar incluido el análisis de esfuerzo y productividad en el Sprint 1 y 2, podemos extraer de ellos las conclusiones de que tanto el esfuerzo como la productividad fueron satisfactorios, especialmente para el Sprint 2. En el Sprint 1 se introdujo el primer patrón usado en el proyecto, el patrón Comando.

En el Sprint 3 se observa que comienzan a aparecer fallos, aún así decidimos que podemos dar todas las historias de usuario del Sprint Backlog por concluidas para dicho sprint, quedando pendiente de corrección los fallos. Durante este sprint también se incluye el patrón Memento y se hace una primera refactorización sobre los comandos.

Al final del Sprint 4 y 5 nos quedamos con objetivos sin sumplir. Analizando ambos sprints, al final del 4 concluimos que la refactorización de las IA se podría haber dividido en subtareas para facilitar el seguimiento de la misma. En el sprint 5 nos dimos cuenta de que era necesaria otra refactorización de las IA y, debido a la complejidad de la misma, no se pudo finalizar. Quizá si hubiéramos repartido de otra manera la refactorización esta podría haber sido concluida en el Sprint 5. También cabe mencionar que al final del Sprint 4 siguen sin corregirse los fallos del Sprint 3 respecto a cargar una partida y esto supuso una mayor deuda técnica que dificultó la finalización del Sprint 5. A pesar de esto, refactorizamos tanto las casillas como los observadores.

Durante el Sprint 6 llevamos a cabo más refactorizaciones respecto al modelo (consultar documento “refactorModelo”) para que este fuese dirigido por eventos en consonancia con la GUI, en vez de por bucles como se hacía antes. Cabe destacar que en este sprint incluímos por primera vez las pruebas unitarias con JUnit y se mejoró la legibilidad del código.

Respecto a las pruebas unitarias, estas nos han sido de gran utilidad para depurar el código y si las hubiéramos ido implementando desde el primer sprint nos hubiésemos ahorrado parte de la deuda técnica con la que nos encontramos al final del Sprint 5.

Finalmente, concluimos el Sprint 7 con gran satisfacción debido a la cantidad de trabajo realizado que nos permitió incluir la refactorización de las IA en la rama principal del proyecto en GitHub, realizar una nueva refactorización de sistema de guardado y carga de partidas mediante el patrón DAO y otra nueva refactorización sobre el multijugador. Asimismo, en este sprint mudamos nuestro juego definitivamente a la GUI y abandonamos la consola, arreglamos fallos, ampliamos la colección de JUnit, organizamos código...

6. SPRINT RETROSPECTIVES

Reunión de 1 hora en la que el Equipo Scrum se inspecciona a sí mismo. Tiene lugar después de la Sprint Review y antes del Sprint Planning. El Scrum Master debe asegurarse de que se lleve a cabo. El propósito es inspeccionar cómo fue el último sprint en cuanto a personas, relaciones, procesos y herramientas; identificar y ordenar los elementos que salieron bien, las posibles mejoras que se podrían realizar y crear un plan para implementar estas mejoras de la mejor manera posible.

Sprint 1:

Puntos buenos:

- Hemos cumplido con los objetivos del sprint.
- Tal y como habíamos planteado el sprint, la carga de trabajo ha sido adecuada.
- La organización del Repositorio en GitHub está bien.
- Hemos aprendido a usar herramientas de modelado en UML como Modelio / Umodel.

- Tenemos una versión del juego que funciona.

Puntos a mejorar:

- Repartiremos las tareas entre los miembros del grupo.

- Haremos los diagramas UML antes de escribir el código correspondiente para que el diseño se corresponda más adecuadamente al código.

- Estableceremos fechas para tener diseños hechos y poder comentarlos entre todos. Así controlamos que el código y los diseños que generamos son mejores.

- Comenzaremos a comentar más claramente el código para que el resto del equipo lo pueda entender.

- Trataremos de reunirnos para comentar los avances del día anterior, o al menos comentarlos por un grupo de WhatsApp.

- Emplear herramientas como Trello, (en vez del Sprint Backlog) para conocer el estado de las distintas tareas a lo largo del sprint.

Puntos malos:

- No hemos repartido las historias de usuario.

- No hemos aplicado Scrum correctamente (no hemos hecho reuniones todos los días, se han hecho tareas que no estaban planificadas para este sprint…).

- El código ha ido por delante de los diagramas UML.

- Faltan los diagramas de una historia de usuario.

Sprint 2:

Puntos buenos:

- Hemos cumplido con los objetivos del sprint marcados al principio de este con margen y hemos podido expandir el mismo.

- La organización del Repositorio en GitHub está bien, se han añadido comentarios a cada commit que hacemos para poder tener una idea superficial de los cambios que se han hecho.

- La comunicación en este segundo sprint ha sido constante y hemos podido mantener una buena organización como grupo, llevando un trabajo parejo puesto que hemos organizado mejor las tareas poniendo unas fechas límites para las mismas.

- Tenemos una versión del juego que permite que varias personas puedan jugar y puedan guardar los resultados de la partida.

- Esta vez el diagrama UML ha ido por delante del código.

Puntos a mejorar:

- Tal y como habíamos planteado el sprint, la carga de trabajo no ha sido adecuada. Los objetivos propuestos se terminaron a la mitad del sprint. En el próximo tendremos que adecuar la carga conforme a lo que hemos podido observar en este sprint.

- Podríamos detallar un poco mejor la utilidad del código que escribimos tanto con comentarios dentro del código como con mejores resúmenes de lo incluido en cada comit

Puntos malos:

- A pesar de que en el anterior sprint lo propusimos como mejora, no hemos usado la pestaña proyectos de GitHub ni nada similar (Trello)

- Un miembro del equipo no está realizando las tareas que se le asignan, ni dando explicaciones de su falta de trabajo.

Sprint 3:

Puntos Buenos:

- Comunicación y organización entre los miembros del grupo

- Se han cumplido los objetivos del Sprint

- Vamos cumpliendo con Scrum en mayor medida.

- Documentamos mejor los commits que hacemos y el código que subimos al repositorio

- El diseño ha ido por delante del código.

- Nos hemos adaptado bien a la situación del confinamiento, adaptando tanto carga de trabajo como medios para comunicarnos.

Puntos a mejorar:

- Aún no hemos integrado en nuestra forma de trabajar ni la pestaña del proyecto ni las Issues de GitHub.

- Deberíamos dedicarnos a probar más el código y hacer tests con JUnit

- Deberíamos incluir condiciones de aceptación para las historias de usuario que escribamos.

- Deberíamos generar un documento para el Sprint planning también.

- Deberíamos mantener más ordenado el repositorio de GitHub.

- Para mejorar la calidad del código, podríamos empezar a usar el sistema de Pull Request que tiene GitHub

Puntos malos:

- Seguimos teniendo un miembro ausente, con todo lo que ello implica.

Sprint 4:

Puntos buenos:

- Hemos empezado a usar las Issues de GitHub y la pestaña de proyectos.

- Hemos creado un documento específico para el sprint planning.

- A pesar de no haber cumplido con lo planificado para esta sprint (en concreto para la parte del menú principal), hemos conseguido crear la GUI necesaria para las funciones básicas del juego.

- Aunque haya una parte que no se ha terminado a tiempo esto no se ha debido a que la carga de trabajo no sea adecuada sino a problemas surgidos a raíz de un problema técnico que no se ha sabido resolver a tiempo. Por lo tanto, concluimos que la carga de trabajo ha sido adecuada.

- Hemos corregido todos los errores encontrados menos uno.

Puntos a mejorar:

- Deberíamos comenzar a crear condiciones de aceptación para las historias de usuario

- Deberíamos crear test con JUnit.

- Deberíamos incluir diagramas UML generales del juego.

Puntos malos:

- No hemos cumplido con algunos de los objetivos propuestos en el apartado de puntos a mejorar del sprint retrospective anterior.

- No se han cumplido con los objetivos marcados para este sprint en lo que respecta al desarrollo del menú de la GUI.

- No ha habido una buena comunicación entre los miembros del equipo que tenían tareas relacionadas desde un principio. Como consecuencia se han tenido que hacer varios arreglos posteriormente para poder unir las partes desarrolladas por cada uno antes de que se estableciera el sistema que se iba a seguir.

- No se ha corregido uno de los fallos planteados en el review anterior (cargar partida).

Sprint 5:

Puntos buenos:

- Hemos empezado a usar las Issues de GitHub y la pestaña de proyectos.

- Hemos conseguido arreglar los errores del Load.

- Hemos comenzado a hacer un refactor de los jugadores máquina

- Hemos creado diagramas UML generales

- El repositorio está mejor organizado

Puntos a mejorar:

- Incluir comentarios útiles en el código con JavaDoc.

- Comenzar a hacer pruebas con JUnit

- Usar de manera más eficiente la pestaña de proyectos de GitHub

- Organizar mejor el repositorio, separando la parte del código de lo demás

- Elaborar documentos explicativos sobre el diseño a medida que se va actualizando el código.

Puntos malos:

- No hemos cumplido con algunos de los objetivos propuestos en el apartado de puntos a mejorar del sprint retrospective anterior.

- No se han cumplido con los objetivos marcados para este sprint en lo que respecta al desarrollo del menú de la GUI.

Sprint 6:

Puntos buenos:

- Hemos empezado a usar JUnit para crear pruebas unitarias. Estas pruebas abarcan las clases Tablero, ComandoPasarTurno, ComandoColocarFicha, ComandoInvertirSentido y GeneradorMazo. Con esto conseguimos encontrar fallos en el juego. Además, en el proceso de depuración también hemos encontrado métodos y atributos obsoletos que se han eliminado.

- Hemos usado JavaDocs. Esto hace que el código sea más limpio y legible.

- Hemos usado la pestaña de Issues de github. Esto nos ha ayudado ya que hemos recibido feedback tanto de los compañeros como del profesor y conseguimos aclarar cuestiones respecto al diseño.

- Hemos usado la pestaña de projects de github. Esto nos ha ayudado a que todos los integrantes del equipo conociéramos los trabajos que quedan por hacer y cuales han sido terminados, para poder organizarnos mejor internamente.

- Hemos cumplido con los objetivos planteados para este sprint salvo iniciar una nueva partida desde la GUI. Gracias a la comunicación entre los miembros del equipo mediante diversos medios hemos podido corregir una serie de problemas e incorporar nuevas tareas de primera necesidad llevándolas a cabo con éxito antes de la finalización del sprint.

Puntos a mejorar:

- Tener la documentación actualizada. En este Sprint se ha revisado y actualizado la documentación sobre las reglas del juego.

- No crear métodos nuevos en las clases para hacer pruebas unitarias a no ser que sea estrictamente necesario, en cuyo caso se comentará dicho método.

- Separar las clases en paquetes para facilitar la visión general sobre el código.

Puntos malos:

- No hemos cumplido con todos los objetivos del sprint.

- Durante la refactorización que hemos hecho en este sprint, no nos ha dado tiempo a adaptar al nuevo modelo algunas funcionalidades como la de guardar/cargar. El proyecto tiene errores, pero se puede jugar.

Sprint 7:

Puntos buenos:

- Tenemos pruebas unitarias que cubren más partes del código. Esto nos ayuda a depurar el proyecto y a comprender qué fallos tiene.

- Hemos generado la documentación con JavaDoc, así que podemos acceder a esa información sin acceder al código de la aplicación.

- Hemos usado la pestaña de projects de github. Esto nos ha ayudado a que todos los integrantes del equipo conociéramos los trabajos que quedan por hacer y cuales han sido terminados, para poder organizarnos mejor internamente.

- Hemos cumplido con todos los objetivos del sprint.

- Hemos sido lo suficientemente organizados y eficientes para poder resolver muchos de los errores que tenía el código.

- Hemos sabido resolver una mini-crisis provocada por un merge mal hecho que dejó la rama master completamente inutilizada.

- Hemos mantenido un buen nivel de comunicación durante todo el sprint. Esto nos ha permitido solucionar problemas con mayor eficiencia y mantener a todos los integrantes informados de la situación actual en cada momento.

- Hemos mejorado la estructura del código agrupando de forma más óptima las clases.

Puntos a mejorar:

- Somos conscientes de que aún podemos pulir más el diseño del juego, pero no lo hemos hecho por falta de tiempo.

- Durante este sprint podríamos haber hecho mejor uso de las issues de GitHub para documentar e informar a todos los miembros del equipo de los bugs que existían y se iban resolviendo a lo largo de todo el sprint. Aún así, hemos sido capaces de mantener un nivel de comunicación suficiente por otras vías.

Puntos malos:

- No hemos conseguido adelantar tanto trabajo como queríamos de cara a la entrega final.

- Hacer merge entre ramas nos ha retrasado mucho durante este sprint, y hemos tenido que continuar el desarrollo empleando otra rama.

Evolución de las Sprint Retrospectives

En general, hemos cumplido el objetivo de las Sprint Retrospectives siguiendo una estructura en la que análizabamos los puntos buenos, los malos y los que nos proponíamos como mejora de cara al siguiente sprint.

Durante los tres primeros sprints se cumplieron los objetivos de dicho sprint sin ningún problema, a pesar de que durante el segundo sprint un miembro del equipo de desarrollo dejó de realizar su trabajo asignado. En los dos siguientes sprints (4 y 5) los objetivos no se llegaron a cumplir, pero para el Sprint 6 cumplimos con casi todos los objetivos y, finalmente, en el Sprint 7 conseguimos recuperarnos de la deuda técnica que llevábamos arrastrando desde el Sprint 3 y cumplimos todos los objetivos.

Respecto al reparto de tareas, hemos sabido adecuar la carga de trabajo a lo largo de los sprints. Esto se puede observar comparando el primer sprint, en el cual no hubo reparto de tareas, hasta el sprint 7, en el que cada miembro del equipo tiene claro su trabajo. Este reparto de tareas se empezó a implementar desde el segundo Sprint y, a pesar de que a partir de ese punto no se vuelve a hacer mención en las retrospectives respecto a este aspecto, se puede observar que el reparto ha sido adecuado en todo momento ya que el incumplimiento de objetivos se debió siempre a fallos técnicos y no a un exceso de carga de trabajo.

En lo que concierne a la comunicación del equipo, esta ha ido mejorando a lo largo de los sprints, notando dicha mejoría a partir del segundo sprint. La mejora en la comunicación nos permitió tener una mejor organización en el proyecto. La base de nuestra comunicación era a través de WhatsApp y presencialmente durante las reuniones. Debido a la situación de cuarentena en la que nos vimos envueltos, las reuniones se vieron sustituidas por el uso de Skype. También mantuvimos comunicación a través de las pestañas de Issues de GitHub, aunque en menor medida y en el último sprint concluimos que deberíamos haber hecho más uso de ellas para informarnos entre nosotros, a pesar de que continuábamos comunicandonos por otros medios.

Sobre el uso de herramientas, a lo largo de proyecto hemos aprendido a manejar varias:

- En el primer sprint comenzamos a usar Modelio y UModel para elaborar los diagramas, aunque finalmente nos quedamos con Modelio.
- A partir del segundo sprint nos dimos cuenta de la utilidad de los comentarios en los commits de GitHub y empezamos a documentar mejor tanto los commits como los comentarios en el código.
- En el cuarto sprint incorporamos las pestañas de projects en nuestro modo de trabajo, lo cual nos permitió seguir el estado de las tareas durante el sprint de una forma más cómoda y visual, ya que inicialmente el estado de las tareas lo indicábamos en el sprint backlog.
- También es durante el cuarto sprint cuando incorporamos la funcionalidad que ofrecen las issues de GitHub, que nos permitieron recibir feedback por parte de miembros del equipo y del profesor sobre las cuestiones que nos iban surgiendo, especialmente en lo que concierne al diseño.
- Durante el Sprint 6 empezamos a usar JavaDocs para comentar el código, tal y como nos habíamos propuesto en el sprint anterior, lo cual nos benefició ya que obtuvimos un código más limpio y legible.

- El uso de JUnit también lo incorporamos en el Sprint 6, a pesar de proponer su uso al final del Sprint 4. Este atraso en JUnit se debía a nuestra falta de conocimiento sobre el mismo y sobre su utilidad, que nos resultó de gran ayuda para encontrar errores y depurar el juego más fácilmente.

A lo largo de los sprints hemos ido mejorando aquellos puntos que nos habíamos propuesto mejorar en el sprint anterior, aunque fuimos arrastrando algunos, como es el caso del uso de JUnit o el del uso de la pestaña de Projects en GitHub, que fue propuesto en el segundo sprint y se consolidó en el cuarto sprint. Un aspecto que nos ha ayudado a organizarnos como equipo y poder cumplir con los plazos de duración del sprint para cumplir con los objetivos fue el establecimiento de fechas límite internas, que fueron propuestas a partir del segundo sprint debido a lo desastroso que había sido el primero en cuanto a reparto de tareas.

En cuanto a fallos que se podrían mencionar en las retrospectives uno de ellos es la ausencia de análisis del funcionamiento del equipo aplicando Scrum y otro es la falta de conclusiones/objetivos sobre algunos aspectos que comentábamos, como puede ser en el sprint 4 cuando mencionamos que usamos las pestañas de issues y projects en GitHub sin hacer referencia a los beneficios que estas nos aportaban, o en el sprint 5 cuando comentamos que hemos mejorado el repositorio de GitHub sin hacer referencia al objetivo de dicha acción, el cual era facilitar la visión general del proyecto y mejorar la organización del mismo.

7. SPRINT PLANNINGS

Reuniones en las que se organiza el trabajo que se va a llevar a cabo en el nuevo Sprint. En esta reunión participa todo el equipo (el equipo de desarrollo, el Product Owner,y el Scrum Master). En esta reunión se debe responder a las siguientes preguntas:

- **¿Qué puedo entregar en este incremento del sprint que comienza?**

Para poder responder esta pregunta se necesita el Product Backlog y el último incremento del producto. Se elabora un objetivo del sprint, este objetivo debería lograrse durante el Sprint a través de completar el Sprint Backlog. También se seleccionan los elementos de la lista del producto.

- **¿Cómo conseguir hacer el trabajo necesario para entregar el incremento?**

Se elabora el Sprint Backlog a partir del Product Backlog y de los objetivos para este sprint.

➔ Sprint 1

No realizado

➔ Sprint 2

No realizado

➔ Sprint 3

No realizado

➔ Sprint 4

1. Objetivos:

a) Crear jugadores automáticos con niveles de dificultad
b) Tener una GUI básica.
c) Arreglar los pequeños fallos ocasionales.

2. Historias de Usuario:

ID: H_39	**Usuario:** Jugador
Nombre historia: Poder ver el tablero en la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 5	**Iteración asignada:** 4
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como Jugador quiero poder ver el tablero, y las fichas que contiene, en la GUI	

ID: H_56	**Usuario:** Jugador
Nombre historia: Poder cambiar una ficha de mi mano desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como Jugador quiero poder cambiar una ficha de mi mano desde la GUI	

ID: H_40	**Usuario:** Jugador
Nombre historia: Poder ver los turnos en la GUI	
Prioridad: Media	**Estado:**
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Alejandro Rivera León	
Descripción: Como Jugador quiero saber cual es el estado de los turnos en la GUI	

ID: H_41	**Usuario:** Jugador
Nombre historia: Poder ver los puntos de los Jugadores en la GUI	
Prioridad: Media	**Estado:**
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Gema Blanco Núñez	

Descripción: Como Jugador quiero saber los puntos que tengo en la GUI

ID: H_42	**Usuario:** Jugador
Nombre historia: Poder ver las monedas de los jugadores en la GUI	
Prioridad: Media	**Estado:**
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Gema Blanco Núñez	
Descripción: Como Jugador quiero saber las monedas que tengo en cualquier momento desde la GUI	

ID: H_43	**Usuario:** Jugador
Nombre historia: Poder cargar una partida desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 4	**Iteración asignada:** 4
Programador responsable: Gema Blanco Núñez	
Descripción: Como Jugador quiero poder cargar una partida desde la GUI	

ID: H_44	**Usuario:** Jugador

Nombre historia: Poder guardar una partida desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Alejandro Rivera León	
Descripción: Como Jugador quiero poder guardar partida desde la GUI.	

ID: H_45	**Usuario:** Jugador
Nombre historia: Poder colocar una ficha en el tablero desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: María Cristina Alameda Salas	
Descripción: Como Jugador quiero poder colocar una ficha en el tablero desde la GUI	

ID: H_56	**Usuario:** Jugador
Nombre historia: Poder cambiar una ficha de mi mano desde la GUI	
Prioridad: Alta	**Estado:**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: Guillermo García Patiño Lenza	

ID: H_47	**Usuario:** Jugador
Nombre historia: Iniciar una nueva partida desde la GUI	
Prioridad: Alta	**Estado: Finalizado**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder iniciar una partida nueva desde la GUI	

ID: H_48	**Usuario:** Jugador
Nombre historia: Introducir jugadores y eliminar desde la GUI	
Prioridad: Alta	**Estado: Finalizado**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder introducir los jugadores y sus nicks desde la GUI	

ID: H_37	**Usuario:** Jugador
Nombre historia: Poder jugar con jugadores controlados por IA	
Prioridad: Alta	**Estado:**
Puntos estimados: 4	**Iteración asignada:** 4
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder jugar contra un jugador controlado por la máquina en tres niveles de dificultad (fácil, medio, difícil)	

➔ Sprint 5

1. Objetivos:

- Pasar el turno
- Usar ventajas
  - Invertir sentido
  - Saltar un jugador
  - Comprar un comodín
- Menú principal
- Mostrar instrucciones en el juego
- Arreglar los fallos del load
- Reestructurar el código de la máquina
- Reestructurar las clases de las casillas
- Reestructurar observers
- Hacer diagramas de clases y secuencia generales
- Hacer una limpieza generalizada del juego para eliminar las partes que se han dejado de utilizar con el avance del proyecto.
- Cambiar la lógica para finalizar la partida.
- Planificar cómo podemos hacer el juego en red.

2. Historias de Usuario:

ID: H_47	**Usuario:** Jugador
Nombre historia: Iniciar una nueva partida desde la GUI	
Prioridad: Alta	**Estado: Finalizado**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder iniciar una partida nueva desde la GUI	

ID: H_48	**Usuario:** Jugador
Nombre historia: Introducir jugadores y eliminar desde la GUI	

Prioridad: Alta	**Estado: Finalizado**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder introducir los jugadores y sus nicks desde la GUI	

ID: H_49	**Usuario:** Jugador
Nombre historia: Poder pasar el turno en la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como Jugador quiero poder pasar mi turno desde la GUI	

ID: H_50	**Usuario:** Jugador
Nombre historia: Poder saltar jugador en la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder comprar una ventaja para saltar el turno a un jugador	

ID: H_51	**Usuario:** Jugador
Nombre historia: Poder invertir sentido en la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder comprar una ventaja para poder invertir el sentido desde la GUI	

ID: H_52	**Usuario:** Jugador
Nombre historia: Poder comprar un comodín en la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Tania Romero Segura	
Descripción: Como Jugador quiero poder ver las fichas que tengo en mi mano desde la GUI	

ID: H_53	**Usuario:** Jugador
Nombre historia: Poder acceder a las instrucciones de la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: David Cruza Sesmero	
Descripción:	

Como Jugador quiero poder ver las fichas que tengo en mi mano desde la GUI

3. Asignación de tareas adicionales

A continuación repartimos los objetivos planteados en el sprint planning que no han sido descritos como historias de usuario entre los desarrolladores:

Tarea	Desarrollador asignado
Arreglar los fallos del load	Gema Blanco Núñez
Reestructurar el código de la máquina	Tania Romero Segura
Reestructurar las clases de las casillas	María Cristina Alameda Salas
Reestructurar observers	Guillermo García Patiño Lenza
Hacer diagramas de clases y secuencia generales.	Alejandro Rivera León
Hacer una limpieza generalizada del juego para eliminar las partes que se han dejado de utilizar con el avance del proyecto	María Cristina Alameda Salas

Cambiar la lógica para finalizar la partida	Guillermo García Patiño Lenza
Planificar cómo podemos hacer el juego en red	María Cristina Alameda Salas

➔ Sprint 6

1. Objetivos:

- Acabar la refactorización de los jugadores controlados por la máquina
- Aplicar las refactorizaciones necesarias para adaptar el producto al juego en red
- Crear pruebas unitarias para el producto con JUnit
- Incluir comentarios útiles en el código con JavaDocs
- Incluir el menú principal dentro de la aplicación
- Embellecer la GUI
- Actualizar diagramas generales

2. Historias de Usuario:

ID: H_47	**Usuario:** Jugador
Nombre historia: Iniciar una nueva partida desde la GUI	
Prioridad: Alta	**Estado: Finalizado**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder iniciar una partida nueva desde la GUI	

ID: H_48	**Usuario:** Jugador
Nombre historia: Introducir jugadores y eliminar desde la GUI	
Prioridad: Alta	**Estado: Finalizado**
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: David Cruza Sesmero	
Descripción: Como Jugador quiero poder introducir los jugadores y sus nicks desde la GUI	

ID: H_49	**Usuario:** Jugador
Nombre historia: Poder pasar el turno en la GUI	
Prioridad: Alta	**Estado:** Finalizado
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Guillermo García Patiño Lenza	
Descripción: Como Jugador quiero poder pasar mi turno desde la GUI	

ID: H_30	**Usuario:** Jugador
Nombre historia: Poder jugar en red con otros jugadores	
Prioridad: Alta	**Estado:**
Puntos estimados: 5	**Iteración asignada:** 6
Programador responsable: María Cristina Alameda Salas	

Descripción:
Como Jugador quiero poder jugar una partida con otros jugadore en red.

➔ Sprint 7

1. Objetivos:

- Testear las máquinas, arreglar posibles errores e incluirlas en la rama principal.
- Modificar el método verificar para que aproveche el nuevo diccionario.
- Iniciar una partida desde la GUI, tanto en red como en local.
- Reestructurar guardado y carga de partidas.
- Hacer más pruebas unitarias y añadir casos de uso para las pruebas superadas.
- Segundo sustituto en el servidor.
- Ir generando documentación necesaria para la entrega final.

Evolución de las Sprint Plannings

La realización del documento de Sprint Planning ha ido evolucionando favorablemente a lo largo de desarrollo del proyecto.

Durante los primeros sprints a pesar de que si que organizabamos las tareas y las repartiamos entre los miembros del equipo de desarrollo, estas no eran reflejadas en el documento, esto sucedió hasta el Sprint 4 que fue el primero que tuvimos que realizar desde casa.

A partir del Sprint 4 comenzamos a incluir la realización del sprint planning en la reunión de cierre de sprint aunque sin mucho éxito ya que no conocíamos realmente las diferencias entre este documento y el sprint backlog.

Esto fue solucionado gracias a la ayuda del profesor Gonzalo Méndez durante una tutoría que mantuvo con el grupo.

Del sprint 5 en adelante los sprint plannings se realizaron correctamente siguiendo el formato establecido.

La carga de trabajo establecida en todos los sprint plannings fue coherente con el tiempo que teníamos y en ningún momento se dio la situación de no llegar a cumplir plazos por problemas derivados de una excesiva carga de trabajo.

8. PRODUCT BACKLOG

De él se encarga el Product Owner. Este documento incluye una lista ordenada de los objetivos del proyecto.

Estos objetivos se concretan en **historias de usuario**: Representación de un requisito del proyecto. Se le asigna un número y un título, una prioridad, una complejidad, un programador y una pequeña descripción de la forma: Como…. Quiero… Para….

Es un artefacto vivo, a medida que avanza al proyecto, surgen nuevos requisitos y se deben añadir a esta lista. Por tanto, es un artefacto ampliable.

Para poder ver un análisis más profundo de la evolución de este documento a lo largo del proyecto puede verse el documento que ha creado el Product Owner para este propósito (Product Backlog).

9. SPRINT BACKLOG

Conjunto de elementos seleccionados que se deben llevar a cabo durante este Sprint para cumplir con el objetivo. Estos elementos son extraídos del Product Backlog durante el Sprint Planning.Está elaborado por el equipo de desarrollo y debe ser actualizado por él. Estos elementos deben ser tareas muy cortas.

10. DESCRIPCIÓN DEL TRABAJO REALIZADO POR CADA MIEMBRO

Alejandro Rivera León

Durante el desarrollo del proyecto he estado encargado (a veces junto a otros miembros del equipo) de diversas historias de usuario y funcionalidades.

Adapte el Tablero cuando creamos la clase Casilla permitiendo que este trabajase con dicho tipo de datos en lugar de con números como hacía anteriormente.

He participado en la funcionalidad **colocar ficha en el tablero**, concretamente encargándome de todo el código referido a la ejecución de el comando colocar ficha, para ello tuve que hacer modificaciones en Colocar ficha, Game y Tablero.

Cosa que me fue de gran utilidad para posteriormente llevar a cabo la funcionalidad de **poner más de una ficha por turno** ya que lo único que tuve que hacer fue añadir pequeñas modificaciones a lo que había realizado anteriormente.

He participado en la funcionalidad **quitar ficha en el tablero**, concretamente en lo relativo a la realización de los diagramas y el diseño de esta.

Fui encargado de llevar a cabo la funcionalidad de **conocer turnos de la partida,** para ello tuve que modificar y agregar código en la clase AdminTurnos.

A dicha funcionalidad **la tuve que adaptar una vez que creamos la GUI** haciendo uso del Patrón Observador implementado por otro integrante.

Participé en todas las versiones de la funcionalidad guardar / cargar partida.

En nuestra primera versión (**guardar partida a medias**) estuve encargado solamente de la parte de guardado y el diseño general de ambas funcionalidades.

Posteriormente en las funcionalidades **guardar partida desde la GUI** y **cargar partida desde la GUI** tuve varios papeles importantes:

En la **primera versión** (Patrón Memento) **estuve a cargo del diseño de ambas funcionalidades y la implementación de guardar partida** aunque la implementación de cargar partida la realizó otro miembro del equipo.

Y en la **segunda versión** (Patrón DAO) **estuve a cargo tanto del diseño como de la implementación de ambas funcionalidades**

También tengo que añadir que **documenté todo el proceso** de evolución de dicha funcionalidad de manera clara y extendida **en un documento de diseño** para ayudar a mis compañeros a entender mejor la motivación detrás de los cambios y el diseño.

Realicé un **rework visual a todos los menús y paneles de la GUI** (quitando el menú principal y el menú de selección de jugadores) **y en ocasiones** tuve que modificar **también la funcionalidad** ya que algunos paneles de la GUI no funcionaban correctamente u otros podían ser mejorados en cuanto a código se refiere.

Para tener un contexto adecuado del calibre de los cambios visuales es recomendable mirar este (Evolución de la GUI).

Poder ver los puntos de los jugadores de la GUI y **poder ver las monedas de los jugadores de la GUI** eran funcionalidades que originalmente estaban a cargo de otro desarrollador pero que me vi obligado a modificar funcionalmente casi en su totalidad a la hora de realizar el rework visual ya que estaba demasiado acoplado a otros paneles cuando este debía ser uno independiente.

Otras aportaciones

Aparte de lo mencionado anteriormente también considero que he sido de gran utilidad al equipo de desarrollo en otros ámbitos.

En ciertas ocasiones **he apoyado a otros miembros del equipo de desarrollo** en lo relativo a **problemas técnicos o** de **diseño** de sus funcionalidades.

He tenido una **gran participación en lo referente a identificación y solución de Bugs** a lo largo de todo el desarrollo por lo que considero que tengo una **gran visión general del proyecto**.

Esto es **debido a que la mayoría de bugs que he identificado y solucionado no sé encontraban entre las funcionalidades que yo había realizado** lo que me obligó a comprender el funcionamiento del resto del trabajo en mayor o menor medida.

He sido capaz de llevar mis **diagramas al día y he aportado diagramas de carácter general** ayudando a mejorar la explicación de diversas partes del proyecto (diagrama general del modelo, diagramas específicos de implementación de interfaces…etc.).

A raíz del anterior punto considero que es oportuno mencionar que **he tenido muy presente a lo largo de todo el desarrollo los principios del diseño** y he procurado siempre pensar en el diseño más conveniente para cada situación aunque he de reconocer que a veces esto no ha sido posible llevarlo a cabo por limitaciones temporales.

Considero que he realizado un **correcto uso de Github** sacándole partido a las diversas facilidades que este ofrece (como por ejemplo la pestañas de **issues** y **project**) y aprendiendo a **manejar varias ramas de desarrollo** simultáneas.

Considero que he mantenido una **gran comunicación con todos los miembros del equipo de desarrollo** en todo momento cosa que debo agradecer también al resto de mis compañeros ya que puedo afirmar que realmente hemos trabajado como un verdadero equipo (no como lo que ocurre a veces de que un equipo de desarrollo se separa en varios independientes).

A raíz de esto creo que debo mencionar también que considero que **he sido de utilidad en las reuniones de cierre de sprint** ya que he sido una de las personas más activas en dichas reuniones.

Otro punto que creo que es importante destacar en que me he preocupado a lo largo de todo el desarrollo de subir **código funcionalmente correcto** y **que se acoplase perfectamente al resto de funcionalidades** que estaban desarrollando otros miembros del equipo **logrando minimizar las situaciones de crisis** producidas por los bugs que no realizar esto correctamente puede desencadenar.

David Cruza Sesmero

A lo largo del proyecto he diseñado varias funcionalidades, las cuales en su mayoría se han centrado en aspectos relacionados con el menú, poder iniciar partidas o aportar información al jugador.

En un primer momento, me encargué de hacer que se pudiese **salir de la partida** y de **crear un menú** que nos sirviese de base en el arranque del proyecto además de hacer una aportación sobre las casillas especiales del tablero, la cual fue modificada posteriormente.

Durante el segundo sprint comenzamos a ampliar la funcionalidad de nuestro juego pudiendo así jugar dos personas o más, lo cual nos obligaba a **poder identificar a cada jugador**, por tanto se pedía el número de jugadores que iba a haber en la partida así como su nombre. Este código en un principio no daba opción a añadir más jugadores de los establecidos en un principio ni rectificar sobre el nick introducido ni tampoco eliminarlos. Con ello también aporté una idea para que el juego tuviera un carácter más competitivo que era la implementación de un **ranking de jugadores**, el cual realicé pero que más tarde se acabó desechando porque necesitábamos centrarnos en otras partes del proyecto. En un primer momento, se guardaba el nombre de los jugadores de la partida y su puntuación final y se volcaban dichos nombres y puntuación en un documento (He de puntualizar que no se revisaba si dicho jugador estaba en el ranking ni se sustituía su puntuación si la que conseguía era superior a la que ya había).

A medida que el proyecto iba avanzando con posibilidades como cargar una partida a medias fue necesario cambiar el menú que había por entonces, ampliando dicho menú en el cuál se mostraban si se quería **empezar una partida o cargar una ya existente** (Esto en un principio era algo complicado puesto que no se mostraba ninguna lista de partidas que podías cargar, como tal podías cargarlas pero el jugador no iba a ser conscientes de que partidas estaban disponibles). Puesto que se empezó a modificar el menú principal y en anteriores sprints la “manipulación de jugadores” ya sea añadir o eliminarlos de la partida se encontraba en un punto muy pobre se añadió un submenú dentro de crear nueva partida que te daba la posibilidad de **añadir jugadores, eliminarlos y de iniciar la partida** cuando el jugador lo decida dándole posibilidad de moverse, con facilidad, entre estos. Además de poder acceder a la información del juego (De qué trata el Scrabble) desde el menú.

En posteriores sprints comenzamos a desarrollar la interfaz gráfica de nuestra aplicación por tanto **todo lo desarrollado en el anterior sprint se pasó a la GUI**. En un principio este menú era muy simple, solo se centraba en botones sin ningún tipo de temática puesto que priorizamos en un momento que el producto pudiera ser jugable y luego nos encargaríamos del carácter visual del proyecto.

Por último se modificó el **menú principal**, mejorándolo en el **carácter visual** además de **cambiar la manera en la que se introducían los jugadores**, estos ahora se añadían mediante una tabla en la cual se podían añadir y eliminar a los jugadores de una manera mucho más fácil y rápida. Tras añadirse las máquinas a nuestro juego se debían de incluir estas a la hora de crear una partida y en un primer momento existía un botón con el cual se **añadían máquinas y se podían seleccionar el nivel de dificultad de cada una**. Más tarde esto fue modificado por Guillermo puesto que generar las máquinas con un

nivel de dificultad distinto presentaba problemas (se pedían las máquinas después de guardar la lista de jugadores) y ahora todas las máquinas que se añadían presentaban la misma dificultad. También se añadió un JMenuBar tanto en el menú principal como en la partida en el cual se podía ver **que acciones podía realizar el jugador** y de **que trataba nuestro juego**.

En cuanto a aspectos que difieren del código, estos han sido mencionados en la valoración grupal e individual comentando aspectos como la comunicación del grupo, aportaciones de ideas, uso de GitHub, buena adaptación a la metodología Scrum, trabajo extra, interés mostrado sobre la materia no quedándote en las tareas que se te asignaba. En general hemos sido un buen grupo que hemos sabido adaptar la carga de trabajo en un proyecto con cierta dificultad y además hemos reaccionado bien a la situación que hemos vivido adaptándola lo mejor posible.

Gema Blanco Núñez

Respecto a las **historias de usuario**, este miembro del equipo ha desarrollado las siguientes:

- Quitar ficha (H_4): parte del método *execute()* del respectivo comando, que fue completado por otros miembros del equipo durante el primer sprint.
- Pasar de turno (H_15), con sus respectivos diagramas de secuencia, de clases y una explicación. Estos diagramas se corresponden con la versión que teníamos del patrón Comando antes de hacer la refactorización.
- Cargar una partida a medias (H_31), con sus respectivos diagramas de secuencia, de clases y una explicación. Estos diagramas fueron realizados con posterioridad a la primera implementación para cargar una partida, con lo que se corresponden con el patrón Memento incorporado en el Sprint 3 y que es usado actualmente.
- Poder ver los puntos de los jugadores en la GUI (H_40), con sus respectivos diagramas de secuencia, de clases y una explicación.
- Poder ver las monedas de los jugadores en la GUI (H_41), con sus respectivos diagramas de secuencia, de clases y una explicación.
- Poder cargar una partida desde la GUI(H_42), implementación de una clase que contenía un botón para cargar una partida, que finalmente fue eliminado ya que los miembros del equipo encargados de realizar el Menú Principal de la GUI no hicieron uso de dicha clase.

Adicionalmente, ha llevado a cabo las siguientes **tareas**:

- Corrección del fallo al cargar una partida que apareció al final del Sprint 3.
- Comentarios con JavaDoc relativos a comandos, la clase *Integrante.java*, la clase *AdminTurnos.java,* los tests, la clase *Casilla.java*...
- Corrección de bugs encontrados durante las pruebas unitarias
- Traslado de la creación de la ventana principal del juego (la que contenía el tablero, las fichas...) a la MainWindow. Anteriormente se encontraba en un "main" creado provisionalmente en la clase *PanelMano.java* desde la que hacíamos las pruebas sobre la GUI. De esta forma el programa pasó a ejecutarse desde el "main" que teníamos como launcher principal (todavía desde consola).
- Cambios en aspectos visuales de la GUI.
- Actualización del documento con las reglas del juego reflejando partes del código que no estaban establecidas en las reglas (por ejemplo, el coste de comprar cada ventaja, o la cantidad de monedas recibidas cada ciertos puntos). Para más información sobre las reglas del juego consultar documento "Reglas.docx".
- Tras la refactorización de los comandos implementó el método *execute()* del comando *ComandoInvertirSentido.java* y del *ComandoSaltarJugador.java*.
- Elaboración de un documento orientativo sobre la aplicación de JUnit en el proyecto. Para más información consultar "*JUnit.docx*".
- Acerca de JUnit, ha llevado a cabo las siguientes pruebas:
    - GeneradorTestSuite.java
    - GeneradorMazo.java
    - GeneradorDiccionario.java
    - CommandTestSuite.java
    - ComandoCambiarFichaTest.java
    - ComandoColocarFichaTest.java
    - ComandoComprarComodinTest.java
    - ComandoInvertirSentidoTest.java
    - ComandoPasarTurnoTest.java
    - ComandoQuitarFichaTest.java
    - ComandoSaltarJugadorTest.java
    - TableroTest.java (pruebas *testAnadirFicha(), testQuitarFicha1()* y *testTableroPrimerTurno()*)
    - CasillaTest.java
    - AllTestsSuite.java

En lo que concierne a **otros aspectos** sobre el trabajo en equipo, comunicación, uso de herramientas y aplicación de Scrum, consultar la valoración grupal que ha sido realizada por los miembros del equipo.

Guillermo García Patiño Lenza:

Durante el desarrollo me he encargado de **varias tareas**, desde las primeras **implementaciones de varias clases** como el AdminTurnos, el Tablero o el Controller, pasando por **varias refactorizaciones** que afectaban a los **Observadores** del Modelo y al propio **Modelo** en sí para adaptarlo al funcionamiento por eventos, y creando por último la **ventana principal** que permite iniciar diferentes tipos de partidas.

Además, he **intervenido** indirectamente en **gran parte del diseño** del juego, **debatiendo** sobre el diseño del propio modelo, **refactorizándolo varias veces** para obtener un diseño más limpio, **ayudado un poco a Cristina en** el diseño de la parte de **cliente-servidor**, **debatiendo y consensuando con Alejandro** el diseño de la primera versión del **guardado del juego**,de los **jugadores automáticos** (en menos medida) colaborando **con Tania**, y creando el **menú principal de la aplicación**.
Además, también he **creado un par de tes**t tanto para asuntos que veía complejos de probar, como para aquellos que creía esenciales para el funcionamiento del juego.

Al **principio del desarrollo**, me encargué de **implementar el tablero**, tarea más o menos **sencilla** ya que en la **práctica de TP1** habíamos implementado un tablero que se mostraba por consola, y tenía una **referencia fiable** que tener en cuenta.

Después, en el **sprint 2**, me enfoqué en **permitir el juego multijugador**. Para ello, **creé la clase AdminTurnos** que se ha mantenido hasta la release final del proyecto. Para esta clase no tenía referencia alguna, y dado que **no teníamos muchos conocimientos de diseño**, la clase acabó agrupando **demasiadas funcionalidades** que deberían quedar relegadas a la clase Scrabble.

Además, con la introducción de esta funcionalidad, **creé también la clase Turno que en ese momento tenía el bucle que permitía a los jugadores introducir comandos a través de la consola**.
Por otro lado, en esta clase es dónde acabó **guardándose la información de las acciones que ha realizado el jugador que tiene el turno** durante el mismo.
Para poder guardar la información, **consensuando con Cristina**, acabé **extendiendo el patrón comando** añadiendo los **métodos 'posible' y 'addCambios'** que determinaban la posibilidad de un comando de ejecutarse dentro de un turno (por ejemplo, se podrá cambiar una ficha solo si se han hecho menos de 7 cambios antes) y registraban en la clase Turno la ejecución del comando, respectivamente.

Más adelante, **colaboré con Alejandro** al **plantear y consensuar** con él la forma en la que **él implementó el guardado del juego** por primera vez. Entre los dos, **concluimos** que la mejor manera de hacerlo en ese momento era **aplicando el patrón memento** con las interfaces Memento y Originator.

Seguidamente, **comenzamos a implementar una vista del juego** empleando una GUI. En concreto, durante esta fase **creé la interfaz ModelObserver** y **añadí algunos métodos** que veía necesarios para comunicar la GUI y el Modelo. Más tarde, **otros miembros del grupo añadieron métodos** a la interfaz para completar las funcionalidades que les fueron asignadas.

Por mi parte, **implementé el tablero de juego** que se ve en la GUI, creando la **clase PanelTablero y la clase CeldaTablero**, que cumplían la función de **representar el tablero en la GUI** y permitir al jugador **quitar fichas del tablero**.

Por otro lado, en este sprint también implementé y **diseñé el medio** a través del cual los **comandos viajan** por el programa **desde la GUI hasta el Modelo** y son ejecutados en él, el **método 'runCommand'**. Este medio ha sufrido **modificaciones** desde su diseño **hasta la release final**, pero la esencia se ha mantenido: viajar a través del Controller hasta el Modelo.

Seguidamente, al tener los elementos de la vista demasiados métodos vacíos, **refactoricé** los observadores, **dividiendo ModelObserver en TManagerObserver y GameObserver,** obteniendo así un código más limpio y entendible. (Se explicaba en el documento 'Nuevos Observadores', ahora obsoleto).

En el siguiente sprint, me ocupé de **seguir implementando alguna funcionalidad básica más**, como por ejemplo la de **cambiar una ficha** de la mano por otra del mazo y la de **pasar turno**. Es con esta última funcionalidad con la que **me di cuenta de que el estado del modelo no era el adecuado** para **soportar una entrada de comandos por GUI**. A pesar de eso, **tardé** un sprint **más en dar con la solución** que precisaba el problema.

Tratando de hallar la solución, **contacté con el profesor para consultar varias dudas, y abriendo varias issues en GitHub** para comentar el asunto con el resto de compañeros. Tras debatir con el profesor y con los compañeros usando GitHub, **seguí pensando en una solución factible**.

Mientras tanto, **Cristina empezó a implementar el juego en red**, y debatí con ella acerca de ciertos aspectos de la creación de jugadores. **Aclaramos que la lista de jugadores debería pasarse al constructor del modelo**, por lo que tendríamos que alterar la forma de crearlos.

Durante la segunda mitad del sprint 6**,** abrí una **issue** relativa a la **creación de los objetos del modelo**, ya que había llegado a una **solución relativamente buena** que requería un refactor bastante grande en la estructura del modelo. Tras **recibir opiniones de todos** los miembros del grupo, e incluso un par de **respuestas del profesor**, procedí a **refactorizar el modelo** según **expongo en** el documento explicativo que creé para ello **'Refactor Modelo'**

Una vez acabé el refactor, me **preocupé de adaptarlo para** que tanto el **juego de cliente-servidor** que estaba implementando Cristina, como los **jugadores máquina** que estaba implementando Tania pudieran **integrarse más o menos fácilmente** en el nuevo modelo. Para ello miré qué llevaban desarrollado y hablé con las dos acerca del funcionamiento de cada parte.

Con Tania concluí que los **jugadores automáticos deberían generar comandos** para ejecutarlos sobre el modelo al igual que hacen los jugadores humanos.

Por otro lado, con **Cristina llegamos a la conclusión** de que era más sencillo y limpio crear una **clase** que **agrupara los elementos de la vista del cliente** que sufrieran actualizaciones. Así, cuando el cliente recibiera una actualización por parte del servidor, se esta clase se encargará de **avisar únicamente** a los **elementos** de la vista que estuvieran **afectados por el cambio**. Esta clase se llamaba **'sustituto'**. Más adelante, ella ha desarrollado esa clase y ha delegado sus responsabilidades en otras clases.
Además, ella **detectó un fallo** que a mí se me había pasado, en el juego en red la vista de la mano no se actualizaba correctamente. Para solucionarlo, **cambié la implementación del Modelo** de modo que se pudieran r**egistrar observadores** en **todos los jugadores**, no solo en el que tenía el turno, tal y como Cristina sugirió.

Además, durante esta fase final del desarrollo, viendo que **aparecían algunos errores**, **programé los test 'TurnosTest' y 'TableroTest'** para depurar el código más fácilmente y que el grupo fuera capaz de localizar futuros errores.

En el **último sprint**, me dediqué a **crear una ventana** desde donde el usuario pudiera iniciar una partida **eligiendo entre todas las posibilidades** de partida que ofrece el juego.

Por último, dentro del grupo se me eligió como **Scrum Master**. Como Scrum Master, me he ocupado de **convocar los daily scrum** (y de que si no era posible hacerlos, se comentaran las tareas de cada uno en algún momento del día mientras había clases presenciales), de **convocar reuniones de emergencia** cuando han sido necesarias, de actuar como **representante del grupo frente a agentes externos** y de **proporcionar medios y herramientas** para que cada uno pudiera desarrollar su trabajo de manera adecuada. Además, creo que he mantenido un **buen nivel de comunicación** con los miembros del equipo, y hemos podido mantener **reuniones cada dos semanas** para cerrar sprint y abrir el siguiente, **además de alguna otra más**, necesaria para comentar objetivos y reorganizar tareas.

María Cristina Alameda Salas

Durante el proyecto he realizado varias tareas, tanto individual como con otros compañeros.

Para describir las tareas realizadas a lo largo del proyecto, he realizado una recopilación de los **commits subidos a github** y para facilitar la explicación lo **dividiré** por **labores** y a su vez alguna de ellas por **sprints**.

- **Labores de diseño y desarrollo**

  Durante el primer sprint, diseñé la jerarquía inicial de **comandos** con el **Patrón Comando,** se realizó ya que tendríamos de estos una gran cantidad y con este patrón se nos solucionarían muchos de los problemas que surgirían al parsearlos, esto lo habíamos aprendido en TP y me pareció bien hacerlo así. Para ello creé la clase **Command y CommandGenerator** y **tres de los comandos** que se iban a usar a lo largo del juego, el comando colocar ficha, quitar ficha y salir de la partida.

  Otras de las aportaciones fueron las clases **GeneradorMazo y Ficha** en prácticamente su versión final, ya que apenas han cambiado, solo se les añadieron algunos métodos. Primero se creó la clase Ficha, y después el generador del mazo a partir de un archivo json obtenía la información correspondiente a las fichas del juego y creaba las instancias correspondientes de ellas y luego las mezclaba aleatoriamente.

  Ya que se tenían que implementar las funcionalidades de los comandos, también añadí a la aplicación las clases **Main** para poder probar el funcionamiento del juego, **Controller** (un controlador básico que gracias a un bucle pedía comandos y una vez generados (por el método estático de la clase CommandGenerator a modo de creador de comandos), los ejecutaba sobre el diseño), y la clase **Game**, sobre la que se ejecutaban los comandos. De esta manera aplicábamos el **patrón modelo-vista-controlador**.

  Durante el segundo laboratorio, en colaboración con Tania, creamos la clase **Jugador**. Se realizó en vista a poder ampliar la ejecución de algunos comandos almacenando información en el jugador (su lista de fichas), como rellenar las manos para poder colocar fichas o quitarlas. De esta manera abstraíamos la funcionalidad de un jugador en una clase.

  En el primer laboratorio del segundo sprint, junto con Tania creamos el **diccionario** e implementamos el **verificar una palabra**. A lo largo del este, me dediqué a solucionar **bugs del tablero** y del **colocarficha**. También creé la clase **instrucciones** con su contenido y se llevó a cabo la historia de usuario de utilizar los **comodines**.

Durante el tercer sprint surgieron problemas con los **turnos** y de ello se derivaron muchos bugs en el modelo, es por ello por lo que gran parte del sprint la dediqué a resolver bugs. Además, había algunas funcionalidades (quitar ficha) que no se ejecutaban correctamente (se podían quitar fichas que habías puesto en otro turno, fichas de otros jugadores…). Es por ello por lo que rediseñé la implementación de esta historia añadiendo **la lista de fichas puesta durante un turno,** primero al Game ya que se me informó que debía ser ahí para no alterar otras clases. Además, se me encargó durante este sprint el **verificar palabras automáticamente**, para ello cree las versiones finales de la clase Coordenadas, la ScoredWord y la versión final del método verificar.

Entonces Guillermo y yo pensamos que el lugar de esta lista de fichas no debía ser el Game, y ya que era algo relacionado con el turno, debía estar en esta clase. Para ello consensué con Guillermo que la mejor manera de solucionar que el quitar ficha pudiera acceder al turno para poder añadir o no una ficha que se colocaba correcta o incorrectamente, sería añadir a los **comandos** los métodos '**addCambios y 'posible**'. De esta manera, al ejecutar un comando primero se chequea si es posible (al quitar una ficha, si la ficha está en esta lista de fichas colocadas durante el turno), después de ejecuta el comando y a continuación se añaden los cambios pertinentes al turno (si coloco una ficha correctamente se añade a esta lista, o si la quito se elimina). Añadí los diagramas correspondientes de diseño.

Durante el cuarto sprint nos dedicamos a integrar la **GUI** en la aplicación. Íbamos bastante a ciegas pues solo habíamos visto la introducción en TP. Aún así, conseguí implementar la mano y el colocar ficha en la GUI. Creé la clase **FichaView** es su versión final y el panel Mano. También tuve que cambiar el **PanelTablero** por varios bugs que hubo.

Como había que probar el **colocar ficha para la GUI** se probó esta funcionalidad numerosas veces por lo que se descubrieron varios errores en dicho método del Game. Al final, tuve que rehacer completamente el método para que funcionara correctamente.

Debido a cambios realizados al introducir los turnos y el administrador, se tuvo que rehacer de nuevo el comando **Salida**. También se arreglaron bugs de **cambiar ficha.**

Durante el quinto sprint lo primero que se realizó fue un cambio en las **casillas** ya que cuando encontrábamos un bug en ellas perdíamos mucho tiempo cambiando los errores de todas. Se redujeron las clases de casillas de 6 a 1.

Entonces encontramos problemas cuando se colocaba una ficha desde la GUI cuando había varias fichas con la misma letra y no se escogía la primera. Pensé entonces que la solución más fácil de implementar sería que el comando colocar ficha funcionara con el id de la ficha que ya tenían y no se usaba, lo planteé al equipo pero parte no quiso que se hiciera así ya que suponía introducir por consola el id (mucho más largo que una letra). Finalmente, se llevó a cabo el cambio del colocar ficha. Teniendo en cuenta lo molesto que resultaría por consola introducir el id, se realizaría introduciendo el **id de la ficha** o la letra, de esta manera el execute del comando al comprobar si la mano del jugador tiene esa ficha comprueba primero si es un id y después si es una letra. Ahora ya funciona correctamente.

Se arreglaron bugs a la hora de crear las máquinas.

Después de realizar los diagramas se llevó a cabo una **reorganización** del repositorio para separar código y documentos y se borraron clases y métodos obsoletos.

Durante el sexto sprint se quiso añadir el **multijugador en red**. Estaba demasiado perdida ya que no sabía nada de este tema y después de hacer un gran trabajo de investigación de enteder cómo funcionaba empecé a diseñarlo pero me surgieron problemillas. Por ello contacté con el profesor de la asignatura que me orientó bastante para poder llevarlo a cabo.

Así, diseñé la primera versión del **traductor, el servidor, JugadorConectado, el lobby, el LobbyObserver, el protocolo, el cliente, el traductor del cliente, estado del cliente, los paneles del lobby y del login y un panel start** que luego se suprimió. También se cambió completamente la **MainWindow** a la versión final y se introdujo en este el cambio entre paneles.

Entonces me di cuenta de que con un pequeño cambio en los comandos, podrían reutilizarse en el servidor para interpretar varios mensajes que llegaban por el socket. Se llevaron a cabo y me dieron la idea que implementé en el siguiente sprint.

También se rediseñó el controlador para que los datos de los jugadores no se pidieran en su interior sino desde fuera, esto se realizó por que el servidor una vez pedidos los datos debía pasarlos al modelo de alguna manera.

Entonces cuando implementé el cliente, me di cuenta de que la vista no podía recibir el controlador porque no había esa parte, por ello se creó la interfaz **Registrador** (actual Controller) en vista a que hubiera varios controladores y el sustituto la implementara como controlador que era.

Se cambió la vista para que ahora recibiera el registrador y se cambiaron y modificaron varias clases de la GUI para adaptar parches realizados al diseño. Se modificaron los observadores de TManagerObserver, ModelObserver y GameObserver.

Durante este sprint tuve que subir varias funcionalidades mías y de otros compañeros de nuevo tras perderse cuando alguien hizo merge de manera incorrecta.

También se generó un **documento** que agrupara toda esta parte de multijugador con multitud de diagramas (Multijugador en red.pdf) y se comentó con **JavaDoc** toda esto y parte del modelo.

Aquí se llevó a cabo una refactorización del modelo, y se tuvieron que arreglar bastantes errores que surgieron de esta antes de finalizar el sprint.

Durante este último sprint, se siguieron solucionando errores derivados de la refactorización.

Una vez que se hubieron solucionado, llevé a cabo una **refactorización** del multijugador. Se crearon los **intérpretes** y su **factoría** y reforme el cliente. También se crearon dos nuevas jerarquías de comando, los **comandos del cliente** y los **comandos del lobby**. También rehíce el documento de Multijugador.

- **Labores de Scrum**

En la celebración de la primera reunión se eligió al Product Owner, pero debido a enfermedad no pudo asistir, así que me encargué temporalmente de las tareas a realizar por el Product Owner el primer Sprint. Creé la primera versión del Product Backlog y di formato a las historias de usuario, formato que se mantuvo a lo largo de todo el proyecto.

- **Otras aportaciones**

Realicé todos los **diagramas de secuencia y de clases** correspondientes a la funcionalidad que me tocó implementar y ayudé a otros integrantes a realizar algunos (como el ver tablero).

He **ayudado** a otros compañeros siempre que he podido, como a Alejandro a embellecer la GUI, poniendo las transparencias a los botones y haciendo que se movieran o que al poner una ficha en el tablero de la GUI se mantuviera de fondo el color correspondiente (para ello cambié los observadores para que informaran de la multiplicidad de la casilla).

He participado todo lo que he podido en las **reuniones** tratando de aportar ideas para mejorar el diseño, las funcionalidades implementadas y la manera de organizarnos. Siendo pesadita para que estuvieran los documentos de la entrega final completos y que todos realizaran sus tareas.

Tania Romero Segura

Para evaluar mi trabajo durante este proyecto, con el fin de aportar más claridad, voy a dividir la evaluación en dos partes: las aportaciones referentes al desarrollo del juego que se me habían asignado y otras aportaciones que harán referencia a colaboraciones, solución de problemas y producción de documentación.

- Aportaciones referentes al desarrollo del juego:

  Para todos los sprints creé los **diagramas UML** antes de escribir el código correspondiente y ésto permitió que mis aportaciones fueran limpias y siguieran un camino concreto y bien marcado desde el principio lo que minimizaba los conflictos entre mi trabajo y el trabajo de mis compañeros. La única excepción fue la elaboración de los jugadores automáticos. Para esa parte los diseños en ocasiones se elaboraron antes del código y en otras ocasiones a posteriores, esto se debe a la gran cantidad de cambios que ha sido necesario hacer sobre la marcha.

  Durante el primer sprint ha sido imposible recuperar cuál fue mi trabajo dado que tuve algunos problemas con GitHub y no puede publicar mi trabajo.

  Durante el segundo sprint se me encargó la incorporación de **casillas especiales** al tablero. Además, elaboré junto con Cristina la funcionalidad que permitía al jugador **verificar una palabra** a través de un comando. Esta última funcionalidad fue sustituida posteriormente por un sistema automático creado por Cristina. A esto también hice el sistema que permite al jugador **obtener una puntuación** por las palabras que coloca una vez terminadas las casillas especiales ya que tenía tiempo suficiente para incluir esta funcionalidad extra.

  En el tercer sprint, me encargué principalmente de crear la funcionalidad que permite al jugador **ganar monedas** y gastarlas para **comprar los tres tipos de ventajas** que hemos incluido.

  En el cuarto sprint, me encargué de **pasar la compra de ventajas a la nueva interfaz gráfica**. Para ello diseñé la estructura tal y cómo quería que quedase visualmente para el jugador la ventana de compras. A partir de este diseño estético y respetando el diseño UML creado para la funcionalidad por consola desde la ejecución del comando procedí a la

implementación de esta parte. Para ello busqué las imágenes en distintas fuentes, reorganicé varias veces la ventana de compras y hice pruebas para comprobar que todo seguía funcionando correctamente.

A partir del sprint 5 mi función ha sido principalmente **crear los jugadores automáticos** (para una explicación a fondo del diseño véase el documento refactorizaciónIA). Para la creación de la primera versión de las IA y para su refactorización fue necasario invertir mucho tiempo de **documentación por internet** complementario. Leí trabajos de master, artículos al respecto y busqué implementaciones avanzadas de jugadores automáticos para diversos juegos para encontrar una forma accesible de implementar las IA. Todo lo que encontraba tenía aspectos muy avanzados así que tuve que dedicar mucho tiempo a desgranar cada parte para obtener un plan medianamente razonable que pudiera implementar y fuera lo más eficiente posible. Aunque el resultado no es tan bueno como yo quisiera lo he hecho lo mejor que he podido con la información de la que disponía.

En el sprint 5 desarrollé un **primer sistema** en el que las máquinas no robaban fichas y buscaban cómo colocarlas. En cambio, las **máquinas buscaban una palabra en el diccionario con cierto grado de aleatoriedad que pudieran colocar y robaban las fichas necesarias para poder colocarlas**. Guillermo me ayudó a probar el funcionamiento de la máquina (que se desarrollaba en una rama a parte) dado que no me daba tiempo a probarlo todo yo sola. Y le pregunté su opinión sobre un conflicto que se daba porque solo se podían colocar fichas al lado de fichas y como él había hecho esa funcionalidad necesitaba saber su opinión para que la máquina colocase sus fichas.

En el sprint 6 me dediqué a una **refactorización completa de las IA** con el objetivo de hacer que tuvieran un **comportamiento más similar al jugador humano**. Para ello se creó un sistema que permitía a la máquina robar fichas y buscar combinaciones de esas fichas existentes en el diccionario para colocarlas en el tablero.

El sprint 7 lo dediqué íntegramente a **añadir las IA a la rama principal**. Esto resultó especialmente difícil porque había una serie de errores debido a que el modelo ya no estaba preparado para incluir las máquinas después de la refactorización del modelo y de la inclusión del menú principal. Fueron necesarios **varios días de depuración para conseguir encontrar el origen de los problemas** dado que los errores principales se daban fuera de los métodos de las máquinas. Las modificaciones que tuvieron que hacerse los comento en el siguiente apartado. También elaboré un **documento** más extenso explicando cómo se habían implementado las IA llamado **refactorizaciónIA**.

- Otras aportaciones:

  En el segundo sprint Cristina y yo creamos un **diccionario** que consistía de una **lista de palabras** que almacenamos e inicializamos en el Game para poder utilizarlo en el método Verificar.

  Durante el desarrollo de la ventana de compra de ventajas se hizo evidente que había que **cambiar el sistema de recogida de excepciones** para poder mostrar mensajes de error a través de la GUI. Por ello, eliminé el punto de recogida de excepciones para que se pudieran recoger en la GUI y añadí **mensajes de error para las funcionalidades del sprint que ya habían sido realizadas** que lo necesitaban

  Para la refactorización de las IA me vi con la necesidad de disponer de un diccionario en el que buscar palabras fuera más eficiente. Al final, opté por crear una clase **Diccionario** desde cero aplicando el patrón singleton para que todo el juego tuviera el mismo diccionario. Este nuevo diccionario lo creé utilizando una estructura de datos basada en la **estructura de datos Trie**. Además, para crear este nuevo diccionario también creé un **nuevo documento de carga en el que se almacenan las palabras sin tildes** dado que algunas palabras no se podían verificar porque nuestras fichas no tienen tilde y las palabras originales sí.

  Al juntar las IA con el resto del proyecto pasó algo extraño digno del programa Cuarto Milenio al hacer **merge en GitHub** y la rama principal quedó inutilizada. Para corregir este error **restauré el estado en otra rama que llamamos antesDeMerge** sobre la que trabajamos hasta el final de la entrega. Después procedí a **introducir los elementos de las máquinas de uno en uno en la rama manualmente** para evitar otro desastre catastrófico.

  Al introducir las máquinas surgieron una serie de problemas originados en otros puntos de la rama que nada tenían que ver con las IA así que tuve que dedicar mucho tiempo a encontrar el origen de estos fallos y arreglarlos.

  El primero fue que en el menú principal **cuando se creaban las máquinas se creaban con un constructor vacío y en ningún momento se les daba el resto de la información** que necesitaban para funcionar aunque se disponía de parte de esa información incluso antes de crear las máquinas. Para solucionarlo tuve que **crear otro constructor** con la información de la que se disponía previamente e

incluí una forma de darle a las máquinas la información que les faltaba en el administrador de turnos.

Otros dos fallos se producían por la **ejecución de los turnos**. El **primer turno** se creaba en el constructor del adminTurnos pero **en ningún momento se mandaba que se ejecutara**. Para los jugadores humanos esto no supone un problema puesto que disponen de la IA y los cambios se hacían correctamente. Sin embargo, **cuando la máquina tiene el primer turno** al no hacer que se ejecute el primer turno internamente en el modelo no se entraba nunca en el sistema de colocación de la primera palabra y la máquina **no hacía nada**. Para solucionarlo **hice que se ejecutara el primer turno**. El otro error se daba porque l**a ejecución de los turnos no estaba bien adaptada para ejecutar los turnos de las máquinas**. Al final **pude solucionarlo cambiando de sitio las llamadas y forzando la finalización del turno de la máquina**.

También se dieron una serie de errores debido a que al copiar el contenido manualmente algunas partes no se copiaron bien y fue necesario **revisar de nuevo todas las partes introducidas**.

Como parte de mis aportaciones, **ayudé a arreglar diversos errores y a recuperar información borrada por algunos merge**.

Debido a la refactorización del modelo que se realizó vi que **había** muchos **errores** en el proyecto así que **aparqué durante al menos un día mi trabajo para poder ayudar a solventarlos**.

Al final del proyecto, hice una **revisión de mis diagramas** porque había algunos que era necesario actualizar por la refactorización y me **encontré con que varias funcionalidades habían dejado de hacer lo que se acordó e incluso algunas habían desaparecido así que me encargué de solventar esos problemas.**

En cuanto a mis aportaciones externas al desarrollo del juego, como Product Owner me he encargado del **mantenimiento del Product Backlog**. También he elaborado el **documento de evaluación de la evolución del Product Backlog** y he realizado una **revisión rigurosa del Product Backlog antes de la entrega** dado que buscando información por internet para la elaboración del documento de estudio de la evolución llegué a la conclusión de que había **aspectos mejorables**. Además, al preguntar sobre este tema en clase **el profesor nos habló de la posibilidad de tener historias que se vuelven a realizar a posteriori y que esto podía incluirse como una evolución de la historia en vez de como una historia nueva** y pensé que esta idea sería perfecta para juntar las historias que hablan de la GUI con las originales.

Creo que he mantenido una **buena comunicación con mis compañeros. Siempre me he mostrado dispuesta a ayudar y he intentado mantener a mis compañeros informados de mi trabajo** lo máximo posible. Además, **he procurado informarme antes de tocar nada que no hubiera desarrollado yo** para no fastidiar el trabajo de nadie.

Además, **me encargué de hablar con el profesor sobre el miembro de nuestro grupo que dejó de dar señales de vida** tras el primer sprint.

Por último, **siempre que mis compañeros han necesitado mi opinión** o se ha iniciado un debate en las Issues de GitHub **me he esforzado por responderles lo antes posible de manera clara y justificada**.