

UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

“El Favor de las guerras: Una versión digital del juego Hanamikoji
utilizando redes neuronales”

DIRECTOR: Vicente García Díaz

AUTOR: Samuel Moreno Vincent

Agradecimientos

Quiero mostrar mi grandísimo agradecimiento a las personas que me han apoyado durante toda carrera y han contribuido a que llegue hasta aquí.

Gracias a mi familia, a mis padres que siempre me animaron a seguir adelante aunque me costase, y a mi hermano y a Victoria por estar siempre ahí para cualquier cosa.

Gracias a mis compañeros de carrera: Antonio Muñoz, Santi Rangel, Santy Vargas, Manu Mateos, Manu Valencia, Trivi, Diego González, Miguel Torres, Emmanuel y Álvaro Baños. Vuestra ayuda ha sido un elemento importantísimo para hacer un poco menos dura esta experiencia.

Gracias a mis compañeros de aventuras: Pedro Juan, Jesús, Masaaki, Sibila, Marta Rasines, Angeloso, Lucía, Marta Aroba, Isma, Calesio, Yandru, Tomás, Darisee, Alexis, Dayvo y Eduardo Álvarez. Gracias a vosotros estos años han sido mucho más llevaderos.

Resumen

El Favor de las Guerreras es un proyecto de desarrollo de software que busca implementar una versión digital del juego de cartas *Hanamikoji* [1] en el que el usuario pueda jugar contra una inteligencia artificial creada utilizando redes neuronales.

El propósito del proyecto se enmarca en la necesidad personal y académica de profundizar en el conocimiento de la Inteligencia Artificial, un campo en actual evolución más concretamente el uso de aprendizaje profundo con redes neuronales. Además, utilizando un lenguaje de programación menos visto en el Grado de Ingeniería Informática del Software , como es Python [2].

Aunque en un principio se valoró utilizar estas tectologías en la implementación de un juego clásico como el ajedrez, finalmente el contexto elegido para su aplicación ha sido el juego de mesa Hanamikoji, un juego de cartas publicado en 2013 en japonés. Esta opción supone un nuevo reto al tener que analizar y diseñar los algoritmos necesarios para controlar la lógica del juego, lo que implica tener que crear mi propio catálogo de datos debido a la inexistencia de estos en internet.

Dentro del amplio catálogo de diseños de redes neuronales [3] y sus posibilidades, este proyecto va a utilizar redes neuronales convolucionales [4] para la generación de las respuestas a la interacción del usuario. Estas redes neuronales están diseñadas principalmente para encontrar patrones en imágenes, esto será utilizado dentro de la aplicación para que aprenda a encontrar los patrones den las situaciones puntuales del tablero de juego y así realizar jugadas que la lleven a la victoria, lo que supondrá un desafío para el jugador.

Para permitir la interacción del usuario con esta inteligencia artificial, se desarrollará una interfaz gráfica en Python, que permita trasladar a un entorno digital la experiencia del juego original, con el objetivo de que una persona sin experiencia en el uso de la aplicación pueda jugar sin dificultad, siempre y cuando conozca las mecánicas originales.

De esta manera la aplicación será construida como una aplicación de escritorio que una persona pueda ejecutar desde su ordenador y pueda jugar de una manera simple y rápida.

Palabras Clave

Inteligencia artificial, Redes neuronales, Redes neuronales convolucionales, TensorFlow, Juegos de mesa, Hanamikoji.

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO	13
1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO	13
1.2 RESUMEN DE TODOS LOS ASPECTOS	14
CAPÍTULO 2. INTRODUCCIÓN	15
2.1 OBJETIVOS DEL PROYECTO	15
2.2 EVALUACIÓN DE ALTERNATIVAS TECNOLÓGICAS.....	15
2.2.1 <i>Relativas a la inteligencia artificial</i>	15
2.2.2 <i>Relativas a la interfaz gráfica de usuario</i>	16
CAPÍTULO 3. ASPECTOS TEÓRICOS	17
3.1 HANAMIKOJI	17
3.1.1 <i>Descripción y objetivo del juego original</i>	17
3.1.2 <i>Reglas del juego original</i>	18
3.1.3 <i>Cambio de nomenclatura</i>	22
3.2 INTELIGENCIA ARTIFICIAL Y REDES NEURONALES	23
3.2.1 <i>Machine Learning</i>	23
3.2.2 <i>Redes neuronales</i>	23
3.2.3 <i>Redes neuronales convolucionales</i>	24
CAPÍTULO 4. PLANIFICACIÓN DEL PROYECTO Y PRESUPUESTO	27
4.1 PLANIFICACIÓN	27
4.2 PRESUPUESTO	29
4.2.1 <i>Coste de personal</i>	29
4.2.2 <i>Coste de la compra de hardware</i>	29
4.2.3 <i>Costes indirectos</i>	29
4.2.4 <i>Desarrollo de Presupuesto Interno</i>	30
4.2.5 <i>Desarrollo de Presupuesto Del Cliente</i>	32
CAPÍTULO 5. ANÁLISIS	33
5.1 DETERMINACIÓN DEL ALCANCE DEL SISTEMA.....	33
5.1.1 <i>Módulo de la red neuronal</i>	33
5.1.2 <i>Módulo de interfaz</i>	33
5.1.3 <i>Módulo de controladores</i>	34
5.2 REQUISITOS DEL SISTEMA	34
5.2.1 <i>Obtención de los Requisitos del Sistema</i>	34
5.2.2 <i>Identificación de Actores del Sistema</i>	36
5.2.3 <i>Especificación de Casos de Uso</i>	36
5.3 IDENTIFICACIÓN DE LOS SUBSISTEMAS EN LA FASE DE ANÁLISIS	40
5.3.1 <i>Descripción de los Subsistemas</i>	40
5.3.2 <i>Descripción de los Interfaces entre Subsistemas</i>	41
5.4 DIAGRAMA DE CLASES PRELIMINAR DEL ANÁLISIS.....	41
5.4.1 <i>Diagrama de Clases</i>	41
5.4.2 <i>Descripción de las Clases</i>	42
5.5 ANÁLISIS DE INTERFACES DE USUARIO	45

5.5.1	<i>Descripción de la Interfaz</i>	46
5.5.2	<i>Descripción del Comportamiento de la Interfaz</i>	47
CAPÍTULO 6. DISEÑO DEL SISTEMA		49
6.1	ARQUITECTURA DEL SISTEMA: VISTA DE BLOQUES DE CONSTRUCCIÓN	49
6.1.1	<i>Nivel 1</i>	50
6.1.2	<i>Nivel 2</i>	51
6.1.3	<i>Nivel 3</i>	51
6.2	DESCRIPCIÓN DETALLADA DE LAS CLASES	55
6.2.1	<i>Controladores</i>	55
6.2.2	<i>Red Neuronal</i>	61
6.2.3	<i>Interfaz Gráfica</i>	62
6.2.4	<i>Ficheros de parametrización y configuración</i>	65
6.3	VISTA DE TIEMPO DE EJECUCIÓN	66
6.3.1	<i>Opciones</i>	66
6.3.2	<i>Ciclo de una partida</i>	67
6.4	DISEÑO DE LA RED NEURONAL	69
6.4.1	<i>Transformación de información en datos utilizables por la red neuronal</i>	69
6.4.2	<i>Preprocesamiento de datos</i>	71
6.4.3	<i>Definición de las capas de la red neuronal</i>	71
6.4.4	<i>Procesamiento del resultado</i>	73
6.5	DISEÑO DE LA INTERFAZ	74
6.5.1	<i>Acciones del adversario</i>	75
6.5.2	<i>Guerreras, armas usadas y favor</i>	75
6.5.3	<i>Acciones del jugador</i>	76
6.5.4	<i>Mano del jugador</i>	77
6.5.5	<i>Acción elegida</i>	78
6.5.6	<i>Botón de aceptar</i>	79
6.5.7	<i>Pantalla de información final</i>	79
CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA		80
7.1	ESTÁNDARES Y NORMAS SEGUIDOS	80
7.1.1	<i>Arquitectura de N-capas</i>	80
7.1.2	<i>Estándar de mensajería CSV</i>	81
7.2	LENGUAJES DE PROGRAMACIÓN	81
7.2.1	<i>Python</i>	81
7.3	HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO	81
7.3.1	<i>Anaconda</i>	82
7.3.2	<i>Spyder</i>	82
7.3.3	<i>Sourcetree</i>	82
7.3.4	<i>Gimp</i>	83
7.4	CREACIÓN DEL SISTEMA	83
7.4.1	<i>Problemas Encontrados</i>	83
7.4.2	<i>Descripción Detallada de las Clases</i>	85
CAPÍTULO 8. DESARROLLO DE LAS PRUEBAS		86
8.1	PRUEBAS UNITARIAS	86
8.1.1	<i>Pruebas del controlador del Bot</i>	86
8.1.2	<i>Pruebas del controlador de la Red Neuronal</i>	89
8.1.3	<i>Pruebas del controlador del Tablero</i>	92

8.1.4	Resultados obtenidos y cambios realizados	97
8.2	PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA	97
8.2.1	Pruebas para los Casos de uso para el Usuario Jugador.....	97
8.2.2	Pruebas para los Casos de uso para el encargado del entrenamiento	99
8.2.3	Resultados obtenidos y cambios realizados	100
8.3	PRUEBAS DE USABILIDAD.....	100
8.3.1	Primer usuario entrevistado	100
8.3.2	Segundo usuario entrevistado	103
8.3.3	Tercer usuario entrevistado	106
8.3.4	Resultados obtenidos y cambios realizados	109
8.4	PRUEBAS DE RENDIMIENTO.....	109
8.4.1	Rendimiento en generación de datos	109
8.4.2	Rendimiento en entrenamiento de la red neuronal.....	110
8.4.3	Rendimiento en la ejecución de las pruebas.....	111
8.4.4	Rendimiento en la respuesta de la red neuronal	111
8.5	PRUEBAS DE EFICACIA DE LA RED NEURONAL	112
CAPÍTULO 9.	MANUALES DEL SISTEMA.....	113
9.1	DESPLIEGUE DEL ENTORNO.....	113
9.2	EJECUCIÓN Y CONFIGURACIÓN	113
9.2.1	Ejecución desde el entorno de desarrollo	113
9.2.2	Ejecución desde el archivo ejecutable.....	114
9.2.3	Configuración del archivo de parámetros.....	114
9.3	MANUAL DE USUARIO	115
9.3.1	Elementos del tablero.....	115
9.3.2	Comportamiento del tablero	121
9.4	MANUAL DEL PROGRAMADOR.....	123
9.4.1	Modificar la red neuronal	123
9.4.2	Modificar la interfaz	124
9.4.3	Modificar las reglas del juego.....	124
CAPÍTULO 10.	CONCLUSIONES Y AMPLIACIONES.....	125
10.1	CONCLUSIONES.....	125
10.2	AMPLIACIONES	126
10.2.1	Nuevo entrenamiento con nuevos datos.....	126
10.2.2	Nuevo modo de entrenamiento con redes neuronales adversarias	126
10.2.3	Nuevo diseño de la red neuronal.....	126
10.2.4	Ampliación del alcance de la aplicación: Uso de servicios web.....	127
10.2.5	Ampliación del alcance de la aplicación: Gestión de usuarios	127
CAPÍTULO 11.	PLANIFICACIÓN Y PRESUPUESTOS FINALES DEL PROYECTO	128
11.1	PLANIFICACIÓN FINAL.....	128
11.1.1	Cambio en los grupos de tareas	128
11.1.2	Tiempos reales dedicados	129
11.1.3	Comparativa y conclusiones.....	129
11.2	PRESUPUESTO FINAL	130
11.2.1	Coste de personal y de hardware	130
11.2.2	Costes indirectos.....	130
11.2.3	Desarrollo del presupuesto final.....	130
11.2.4	Comparativa con el presupuesto inicial.....	131

CAPÍTULO 12.	REFERENCIAS BIBLIOGRÁFICAS	132
CAPÍTULO 13.	APÉNDICES	135
13.1	GLOSARIO Y DICCIONARIO DE DATOS	135
13.2	CONTENIDOS.....	135
13.2.1	<i>Contenido del repositorio en Git.....</i>	<i>135</i>
13.2.2	<i>Contenido del archivo adjunto</i>	<i>136</i>
13.2.3	<i>Contenido del fichero de ejecución.....</i>	<i>136</i>
13.3	ÍNDICE ALFABÉTICO	137

Índice de Figuras

Ilustración 3.1 - Ejemplo de preparación del tablero.....	18
Ilustración 3.2 - Símbolo de acción de Secreto	19
Ilustración 3.3 - Símbolo de acción de Renuncia	20
Ilustración 3.4 - Símbolo de acción de Regalo	20
Ilustración 3.5 - Ejemplo de acción de Regalo	20
Ilustración 3.6 - Símbolo de acción de Competición	21
Ilustración 3.8 - Ejemplo de acción de competición.....	21
Ilustración 3.9 - Ejemplo de fin del juego	22
Ilustración 3.10 - Ejemplo de red neuronal	24
Ilustración 3.11 - Ejemplo de operación de convolución	25
Ilustración 3.12 - Ejemplo de operación de max-pooling.....	25
Ilustración 4.1 - Diagrama de planificación	28
Ilustración 5.1 - Casos de uso: Usuario Jugador.....	37
Ilustración 5.2 - Descripción del caso de uso: Encargado del Entrenamiento	39
Ilustración 5.3 - Diagrama de comunicación entre los Subsistema	41
Ilustración 5.4 - Diagrama de clases	42
Ilustración 5.5 - Primer esbozo (a mano) de la interfaz	46
Ilustración 5.6 - Segundo esbozo (ya en digital) de la interfaz	47
Ilustración 6.1 - Vista en bloques completa.....	49
Ilustración 6.2 - Vista en bloques de nivel 1	50
Ilustración 6.3 - Vista en bloques de nivel 2	51
Ilustración 6.4 - Vista en bloques de nivel 3	51
Ilustración 6.5 - Detalle del módulo de controladores de la vista en bloques de nivel 3.....	52
Ilustración 6.6 - Detalle del módulo de la red neuronal de la vista en bloques de nivel 3.....	53
Ilustración 6.7 - Detalle del módulo de la interfaz gráfica de la vista en bloques de nivel 3	54
Ilustración 6.8 - Descripción del fichero ParametrosCNN	65
Ilustración 6.9 - Descripción del fichero ParametrosDatos.....	65
Ilustración 6.10 - Descripción del fichero ParametrosGUI	65
Ilustración 6.11 - Descripción del fichero ParametrosImagenes	65
Ilustración 6.12 - Descripción del fichero ParametrosMenu	66
Ilustración 6.13 - Descripción del fichero ParametrosTablero	66
Ilustración 6.14 - Diagrama de flujo de una partida	68
Ilustración 6.15 - Ejemplo de interfaz de usuario	74
Ilustración 6.16 - Ejemplo de acciones del adversario sin utilizar	75
Ilustración 6.17 - Ejemplo de acciones del adversario utilizadas	75
Ilustración 6.18 - Ejemplo de visualización de las guerreras.....	75
Ilustración 6.19 - Ejemplo de guerrera con el favor del adversario.....	76
Ilustración 6.20 - Ejemplo de guerrera con el favor del jugador	76
Ilustración 6.21 - Ejemplo de acciones del jugador sin utilizar.....	76
Ilustración 6.22 - Ejemplo de acciones del jugador utilizadas.....	77
Ilustración 6.23 - Ejemplo de acciones del jugador bloqueadas.....	77
Ilustración 6.24 - Ejemplo de mano del jugador con todas las cartas disponibles.....	77
Ilustración 6.25 - Ejemplo de mano del jugador con cartas seleccionadas.....	77
Ilustración 6.26 - Ejemplo de mano del jugador bloqueada	78
Ilustración 6.27 - Ejemplo de acción elegida	78

Ilustración 6.28 - Ejemplo de acción de decisión	78
Ilustración 6.29 - Ventana de información de resultado.....	79
Ilustración 7.1 - Representación del modelo de n-capas de la aplicación	80
Ilustración 7.2 - Logotipo del programa Anaconda.....	82
Ilustración 7.3 - Logotipo del programa Spyder.....	82
Ilustración 7.4 - Logotipo del programa Sourcetree	82
Ilustración 7.5 - Logotipo del programa Gimp	83
Ilustración 9.1 - Apariencia de la aplicación Spyder en Anaconda	113
Ilustración 9.2 - Contenido de la carpeta del ejecutable.....	114
Ilustración 9.3 - Imagen de la ventana de la aplicación	115
Ilustración 9.4 - Sección del favor de las guerreras.....	116
Ilustración 9.5 - Relación entre guerreras y armas	117
Ilustración 9.6 - Sección de los marcadores de las armas usadas	117
Ilustración 9.7 - Sección de las acciones del adversario	118
Ilustración 9.8 - Ejemplo de acción sin usar.....	118
Ilustración 9.9 - Ejemplo de acción usada	118
Ilustración 9.10 - Sección de las acciones del jugador	119
Ilustración 9.11 - Sección de las cartas de la mano del jugador	120
Ilustración 9.12 - Sección de la acción actual.....	121
Ilustración 9.13 - Ejemplo de acción de selección.....	123
Ilustración 11.1 - Comparación de planificación inicial y final	129
Ilustración 13.1 - Contenido del archivo adjunto.....	136
Ilustración 13.2 - Contenido del fichero de ejecución.....	136

Índice de Tablas

Tabla 1 - Tareas de la planificación	27
Tabla 2 - Costes indirectos	30
Tabla 3 - Presupuesto completo del desarrollo del proyecto	30
Tabla 4 - Partida completa	31
Tabla 5 - Partida simplificada	32
Tabla 6 - Requisitos funcionales de la red neuronal	34
Tabla 7 - Requisitos funcionales del usuario	35
Tabla 8 - Requisitos funcionales del sistema	36
Tabla 9 - Requisitos no funcionales	36
Tabla 10 - Descripción del caso de uso: Seleccionar acción	37
Tabla 11 - Descripción del caso de uso: Cambiar acción seleccionada	37
Tabla 12 - Descripción del caso de uso: Seleccionar cartas	38
Tabla 13 - Descripción del caso de uso: Enviar acción	38
Tabla 14 - Descripción del caso de uso: Seleccionar cartas acción pendiente	38
Tabla 15 - Descripción del caso de uso: Generar datos de prueba	39
Tabla 16 - Descripción del caso de uso: Entrenar modelo	39
Tabla 17 - Clase Entrenamiento	43
Tabla 18 - Clase Predicción	43
Tabla 19 - Clase Controlador Partida	43
Tabla 20 - Clase Controlador Tablero	44
Tabla 21 - Clase Controlador Bot	44
Tabla 22 - Clase Controlador Red Neuronal	44
Tabla 23 - Clase Controlador Jugador	45
Tabla 24 - Clase GUI	45
Tabla 25 - Leyenda común a todas las vistas	50
Tabla 26 - Descripción de la clase ControladorPartida	56
Tabla 27 - Descripción de la clase ControladorTablero	58
Tabla 28 - Descripción de la clase ControladorBot	59
Tabla 29 - Descripción de la clase ControladorJugador	59
Tabla 30 - Descripción de la clase ControladorRedNeuronal	61
Tabla 31 - Descripción de la clase ControladorGeneradorDatos	61
Tabla 32 - Descripción de la clase Entrenamiento	62
Tabla 33 - Descripción de la clase Prediccion	62
Tabla 34 - Descripción de la clase GUI_Tkinter	64
Tabla 35 - Descripción de la clase Popup_Tkinter	65
Tabla 36 - Pruebas para el método: seleccionar acción	87
Tabla 37 - Pruebas para el método: seleccionar acción selección	88
Tabla 38 - Pruebas para el método: seleccionar acción	90
Tabla 39 - Pruebas para el método: seleccionar acción selección	91
Tabla 40 - método: crear tablero	92
Tabla 41 - Pruebas para el método: iniciar ronda	92
Tabla 42 - Pruebas para el método: vista del tablero	94
Tabla 43 - Pruebas para el método: robar carta	94
Tabla 44 - Pruebas para el método: realizar acción	97
Tabla 45 - Descripción de las pruebas del Caso de Uso: Seleccionar acción	97

Tabla 46 - Descripción de las pruebas del Caso de Uso: Cambiar acción seleccionada.....	98
Tabla 47 - Descripción de las pruebas del Caso de Uso: Seleccionar cartas	98
Tabla 48 - Descripción de las pruebas del Caso de Uso: Enviar acción	99
Tabla 49 - Descripción de las pruebas del Caso de Uso: Seleccionar cartas acción pendiente.....	99
Tabla 50 - Descripción de las pruebas del Caso de Uso: Generar datos de prueba	99
Tabla 51 - Descripción de las pruebas del Caso de Uso: Entrenar modelo	100
Tabla 52 - Preguntas de carácter general del primer usuario	101
Tabla 53 - Preguntas sobre la aplicación del primer usuario	103
Tabla 54 - Cuestionario para el responsable del primer usuario.....	103
Tabla 55 - Preguntas de carácter general del segundo usuario.....	104
Tabla 56 - Preguntas sobre la aplicación del segundo usuario	106
Tabla 57 - Cuestionario para el responsable del segundo usuario	106
Tabla 58 - Preguntas de carácter general del segundo usuario.....	107
Tabla 59 - Preguntas sobre la aplicación del segundo usuario	109
Tabla 60 - Cuestionario para el responsable del segundo usuario	109
Tabla 61 - Resultado de la transformación de la planificación inicial.....	128
Tabla 62 - Planificación con tiempos finales	129
Tabla 63 - Costes indirectos finales	130
Tabla 64 - Presupuesto final del desarrollo del proyecto.....	130
Tabla 65 - Comparativa de presupuestos inicial y final.....	131

Capítulo 1. Memoria del Proyecto

En este capítulo se va a mostrar un resumen de los puntos más importantes del proyecto, tanto del proyecto completo como de la documentación presente.

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Dado el auge actual de la inteligencia artificial y el uso de las redes neuronales en muchos ámbitos de la informática, la primera motivación del proyecto es obtener un conocimiento y experiencia en este campo, tanto para tener una base para futuros proyectos como para tener una visión más específica y práctica.

La segunda motivación para la creación de este proyecto es la motivación personal de la realización de juegos en versión digital, en este caso el desarrollo de un juego de mesa conocido a nivel personal por haberlo jugado con anterioridad. El juego *Hanamikoji* [1] no cuenta actualmente con una versión digital que de la opción de enfrentarse a un adversario no jugador, por lo que me pareció un reto realizar mi propia versión de este juego en vez de un juego con más estudiado como el ajedrez.

Otra de las motivaciones por la que elegir este juego de cartas en específico es la búsqueda de patrones que puedan llevar a un jugador a obtener la victoria en el juego con mayor facilidad. Por ello el uso de inteligencia artificial y redes neuronales supone una elección perfecta para este objetivo.

También es una motivación para la realización de este proyecto el hecho de que durante la carrera se han visto lenguajes de programación como Java [5], JavaScript [6], C [7], o C++ [8] dejando de lado otros lenguajes de programación interesantes. En este caso, y aprovechando su uso en redes neuronales, se ha decidido utilizar el lenguaje de programación Python, que es muy utilizado en ciertas ramas científicas, de ingeniería y de investigación y puede ser muy útil en caso de trabajar en una de esas ramas en un futuro.

El objetivo principal de la aplicación Favor de las Guerreras, por tanto, es conseguir una implementación del juego de cartas original *Hanamikoji* que permita a una persona, o jugador, llevar a cabo partidas de la misma manera que lo haría en la vida real, pero apoyándonos en una inteligencia artificial como adversario y utilizando un entorno grafico agradable y similar al que tiene el juego en su versión original que permita llevar a cabo partidas sin la necesidad de un aprendizaje extra del uso de la interfaz de la aplicación.

Para ello, uno de los objetivos a conseguir es diseñar una red neuronal convolucional [4] que sea capaz de entrenarse utilizando un conjunto de datos generados por la propia aplicación utilizando jugadores virtuales que realizan acciones aleatorias.

Una vez entrenada la red neuronal [3], la aplicación será capaz de ofrecer partidas en las que un jugador humano pueda enfrentarse a esta inteligencia artificial la cual basará sus jugadas en la experiencia adquirida.

Para que se pueda obtener la experiencia del juego original de cartas sin la necesidad de jugarlo en físico con otros jugadores, se desarrollará una interfaz gráfica en Python que nos ofrecerá una manera sencilla e intuitiva de interactuar con la inteligencia artificial.

1.2 Resumen de Todos los Aspectos

Capítulo 1. Memoria del Proyecto: Contiene un resumen de los puntos más importantes del proyecto, tanto del proyecto completo como de la documentación presente.

Capítulo 2. Introducción: Se describe brevemente los objetivos del proyecto y las alternativas que se tuvieron en cuenta antes de comenzar.

Capítulo 3. Aspectos Teóricos: Se describe los principales conceptos, herramientas y tecnologías existentes que se van a usar en el proyecto.

Capítulo 4. Planificación del Proyecto y Presupuesto: Se describe la planificación en la que se estimará la duración de las tareas y se hará un reparto de estas, así como un presupuesto de cuánto costará realizar el proyecto.

Capítulo 5. Análisis: Se describe la documentación de todos los puntos realizados del análisis del sistema, los cuales se utilizarán para la realización del diseño.

Capítulo 6. Diseño del Sistema: Se describe el diseño final del sistema.

Capítulo 7. Implementación del Sistema: Se describen estándares, lenguajes de programación, herramientas y programas utilizados para la implementación del sistema, así como el proceso de creación.

Capítulo 8. Desarrollo de las Pruebas: Se describe el diseño y resultado de las pruebas realizadas.

0.

Manuales del Sistema: Se describen los manuales de despliegue, ejecución, para el usuario y el programador.

Capítulo 10. Conclusiones y Ampliaciones: Se explican las conclusiones de realizar el proyecto y posibles aplicaciones.

Capítulo 11. Planificación y Presupuestos finales del Proyecto: Se describe la planificación final realizada y su comparativa con la inicial.

Capítulo 12 Referencias Bibliográficas: Se describen las referencias que se han utilizado para la realización del proyecto.

Capítulo 13. Apéndices: Se describen los diferentes apéndices incluidos en la documentación.

Capítulo 2. Introducción

En este capítulo se va a describir brevemente los objetivos del proyecto y las alternativas que se tuvieron en cuenta antes de comenzar.

2.1 Objetivos del Proyecto

El principal objetivo del proyecto es aprender más a cerca de las redes neuronales y su aplicación en diferentes campos, en este caso en un juego de mesa. Además, se escogió un juego de mesa con azar para poder ver el comportamiento de una red neuronal dentro de este campo.

2.2 Evaluación de alternativas tecnológicas

En lo que respecta a la inteligencia artificial, actualmente está en auge el uso de redes neuronales convolucionales [4] para el tratamiento de imágenes ya que son capaces de reconocer patrones lo que las hace tener una gran versatilidad.

Ya que el juego de mesa estaba claro desde un inicio, solo quedaban dos grandes decisiones a tomar, que tipo de inteligencia artificial se iba a usar y que tipo de interfaz gráfica de usuario se iba a implementar.

2.2.1 Relativas a la inteligencia artificial

Tras el estudio inicial de los sistemas de inteligencia artificial con redes neuronales se

2.2.1.1 Redes neuronales convolucionales

Estas redes neuronales se basan en tomar una matriz de datos que representa una imagen y pasarla por capas de transformación convolucional (de ahí su nombre) que buscan patrones en las imágenes y pueden predecir que objetos hay en la imagen.

Esta es una gran opción para la aplicación ya que se puede transformar la información del juego en una matriz de datos que simule una imagen y así poder utilizar las propiedades de la red neuronal convolucional para “predecir” el resultado de la jugada que se debe realizar.

El inconveniente es la gran cantidad de datos necesarios para entrenarla, algo que se tendrá en cuenta más adelante.

Esta fue la alternativa que se eligió.

2.2.1.2 *Redes neuronales adversarias*

Estas redes neuronales consisten en enfrentar a dos redes neuronales, normalmente con objetivos contrarios, por ejemplo, una red generadora de imágenes y una red encargada de saber si estas imágenes son reales o generadas.

Esta opción no resulta tan interesante como la anterior ya que requiere encontrar una función que describa cuan buena es una acción realizada por la red neuronal y esto solo puede ser descrito al finalizar la partida.

2.2.2 *Relativas a la interfaz gráfica de usuario*

En esta sección se van a contemplar las alternativas que se barajaron para crear la interfaz de usuario de la aplicación.

2.2.2.1 *HTML y JavaScript*

En un principio se planteó la posibilidad de implementar la aplicación como una página web, por lo que la interfaz gráfica sería implementada en algún framework web que usase HTML [9] y JavaScript [6] como podría ser ReactJS [10] o AngularJS [11]. Esta idea fue desechada ya que llevaría demasiado tiempo hacer esta implementación ya que me hubiera obligado a dedicarme al aprendizaje de tecnologías que escapan al alcance de este proyecto.

2.2.2.2 *Pygame*

Una vez que se decidió utilizar Python para la red neuronal por la cantidad de librerías existentes en este campo, una opción lógica era utilizar otra librería para realizar la interfaz gráfica. En un principio se pensó utilizar Pygame [12], que es una librería para realizar videojuegos en Python. Sin embargo, una vez que se empezó la implementación con esta librería se descartó ya que está más enfocada en el uso de elementos móviles por la pantalla, y al ser la aplicación un tablero de juego inmóvil resultaba ineficiente programar la interfaz con Pygame.

2.2.2.3 *Tkinter*

Al final se optó por utilizar Tkinter [13], que es una librería gráfica de Python. Aunque no esté pensada para ser usada específicamente en juegos y resultó un poco engorrosa de utilizar dio un buen resultado.

Capítulo 3. Aspectos Teóricos

Este capítulo está destinado a describir brevemente los principales conceptos, herramientas y tecnologías más representativas que vamos a usar en nuestro proyecto.

3.1 Hanamikoji

Aquí vamos a ver las reglas del juego original Hanamikoji [14] en el que se ha basado el proyecto y los cambios de nomenclatura que se hicieron para personalizarlo para este caso en específico.

3.1.1 Descripción y objetivo del juego original

Bienvenido a la calle de Geishas mejor conocida de la antigua capital, Hanamikoji.

Las Geishas, las agraciadas mujeres que dominan elegantemente el arte, la música, la danza, y toda una variedad de representaciones artísticas tras años de aprendizaje, son ampliamente respetadas y adoradas. Geisha podría traducirse como “artista” y con sus bailes y canciones entretienen a todo el mundo.

En Hanamikoji, dos jugadores compiten por conseguir el favor de las siete mejores Geishas coleccionando los objetos representativos con los que tienen mayor talento. Con cada cuidadosa apuesta y en ocasiones algunos movimientos audaces, puedes obtener los objetos esenciales regalando los menos significativos. ¿Podrás superar a tu oponente y ganar más favores de las Geishas?

3.1.1.1 Objetivo del juego

En Hanamikoji el objetivo del juego es ganar el Favor de 4 Geishas u 11 (o más) Puntos de Carisma.

Tú y tu oponente os turnáis realizando acciones para conseguir distintas clases de “Cartas de Objeto”. Por cada Geisha, si tienes más “Cartas de Objeto” que tu oponente del tipo correspondiente, ganas el Favor de la Geisha.

El juego continuará hasta que uno de los jugadores alcance el objetivo de victoria en la Fase de Puntuación.

3.1.2 Reglas del juego original

Este es un extracto de Hanamikoji - Reglas del juego de cartas [14] que son las reglas que vienen en la caja física del juego.

3.1.2.1 Componentes del juego

7 “Cartas de Geisha” con sus respectivos 7 “Marcadores de Victoria”, que representan el a quien deben su favor las Geishas.

8 “Marcadores de Acción”, que representan las 4 acciones de cada jugador y si han sido o no utilizadas.

21 “Cartas de Objeto”, que estarán en el mazo de robo, pasarán a la mano un jugador y se utilizarán para realizar las acciones.

3.1.2.2 Preparación

1. Coloca 7 “Cartas de Geisha” en una fila, en el siguiente orden de izquierda a derecha, entre los jugadores.
2. Coloca 1 “Marcador de Victoria” en el centro de cada “Carta de Geisha”.
3. Apila las “Cartas de Objeto” boca abajo en un mazo y colócalo a un lado.
4. Cada jugador coge 4 “Marcadores de Acción” del mismo color con el lado coloreado hacia arriba y los coloca frente a él.
5. El jugador más joven es el jugador inicial.



Ilustración 3.1 - Ejemplo de preparación del tablero

3.1.2.3 Secuencia de Juego

El juego se desarrolla durante una o varias rondas. Cada ronda consiste en 3 fases en el siguiente orden:

- Fase 1: Reparto
- Fase 2: Acción
- Fase 3: Puntuar y Actualizar

Si cualquier jugador consigue el objetivo de victoria en la fase de Puntuar, el juego finaliza inmediatamente. Si ningún jugador consigue el objetivo de victoria, el juego prosigue a la siguiente ronda. El juego continuará hasta que alguno de los jugadores gane.

3.1.2.3.1 Fase 1: Reparto

El jugador inicial baraja las 21 “Cartas de Objeto” y las apila boca abajo, y al azar retira 1 carta del mazo y la devuelve a la caja del juego sin mirarla. Esta carta no se usará en esta ronda. Ningún jugador tiene permitido revisarla.

Reparte a cada jugador 6 “Cartas de Objeto” para conformar su mano, que se mantendrá oculta.

Apila el resto de “Cartas de Objeto” boca abajo como “Mazo de Objetos” y colócalo a un lado de la fila de “Cartas de Geisha”.

3.1.2.3.2 Fase 2: Acción

Empezando por el jugador inicial, los jugadores se alternan en turnos (Jugador A -> Jugador B -> Jugador A -> Jugador B -> Etcétera) hasta que ambos jugadores hayan realizado 4 turnos.

En tu turno, debes coger una carta del “Mazo de Objetos” y realizar una acción.

Cuando realizas una acción, escoges 1 de tus “Marcadores de Acción” del lado colorado y realizas la correspondiente acción. Tras resolverla, coloca boca abajo el marcador. No puedes usar marcadores boca abajo (los “Marcadores de Acción” de cada jugador se podrán usar una sola vez por ronda).

Hay 4 acciones en Hanamikoji:

3.1.2.3.2.1 Secreto

Escoge una carta de tu mano y colócala boca abajo debajo del “Marcador de Acción” usado (Secreto). Esta carta se revelará en la Fase de Puntuar y se puntuará.



Ilustración 3.2 - Símbolo de acción de Secreto

Puedes revisar esta carta en cualquier momento.

3.1.2.3.2.2 *Renuncia*

Escoge 2 cartas de tu mano y colócalas boca abajo frente a ti, debajo del “Marcador de Acción” usado (Renuncia). Estas cartas no puntuarán en esta ronda.

Puedes revisar estas cartas en cualquier momento.



Ilustración 3.3 - Símbolo de acción de Renuncia

3.1.2.3.2.3 *Regalo*

Escoge 3 cartas de tu mano y colócalas boca arriba frente a ti.



Ilustración 3.4 - Símbolo de acción de Regalo

Tu oponente escoge 1 de esas cartas y la coloca en su lado junto a la correspondiente Geisha. Tú colocas las otras 2 cartas en tu lado junto a la(s) correspondiente(s) Geisha(s). Estas cartas puntuarán.



Ilustración 3.5 - Ejemplo de acción de Regalo

3.1.2.3.2.4 *Competición*

Escoge 4 cartas de tu mano y colócalas boca arriba frente a ti. Divídelas en dos grupos, cada uno con 2 cartas.



Ilustración 3.6 - Símbolo de acción de Competición

Tu oponente escoge 1 grupo y coloca las 2 cartas en su lado junto a la(s) correspondiente(s) Geisha(s). Tú colocas las otras 2 cartas en tu lado junto a la(s) correspondiente(s) Geisha(s). Estas cartas puntuarán.



Ilustración 3.7 - Ejemplo de acción de competición

3.1.2.3.3 Fase 3: Puntuar y Actualizar

Después de que ambos jugadores hayan realizado 4 acciones, el juego prosigue a la Fase 3.

Los jugadores revelan la carta debajo del “Marcador de Acción” (Secreto) y la colocan en su lado junto a la correspondiente Geisha. Compara el número de “Cartas de Objeto” en ambos lados de cada Geisha.

- Un lado tiene más cartas que el otro: El lado con más “Cartas de Objeto” gana el Favor de la Geisha. Mueve el Marcador de Victoria al lado vencedor.

- Ambos lados empatan o no hay cartas: No se mueve el Marcador de Victoria.

Tras puntuar, los jugadores calculan el número de Geishas cuyo Favor han ganado y suman sus Puntos de Carisma. Si algún jugador alcanza el objetivo de victoria, el juego finaliza inmediatamente (ver *Finalización del Juego*).

Si ningún jugador consigue el objetivo de victoria, se procede a Actualizar:

- Recoge TODAS las “Cartas de Objeto” de la mesa y de la caja, apílalas boca abajo en un mazo y colócalo a un lado.

- Los Marcadores de Victoria permanecerán en su lugar. Nota: No los devuelvas al centro de cada “Carta de Geisha”.

- Los jugadores colocan boca arriba sus “Marcadores de Acción”.

- El segundo jugador se convierte en el jugador inicial.
- La siguiente ronda está lista para empezar.

3.1.2.4 Finalización del Juego



Ilustración 3.8 - Ejemplo de fin del juego

Si algún jugador gana el Favor de 4 Geishas u 11 (o más) Puntos de Carisma, el juego finaliza.

Si un único jugador alcanza el objetivo de victoria, es el vencedor.

Si un jugador gana el Favor de 4 Geishas y el otro gana 11 (o más) Puntos de Carisma, éste último vence.

3.1.3 Cambio de nomenclatura

Para darle un toque más personal al proyecto y poder cambiarle la estética, se ha decidido hacer un cambio de nomenclatura en alguno de los términos:

Hanamikoji: Pasa a llamarse **El Favor de las Guerreras**

“Cartas de Geisha”: Pasan a llamarse Cartas de guerreras o **Guerreras**

“Cartas de Objeto”: Pasan a llamarse Cartas de armas o **Armas**

3.2 Inteligencia Artificial y Redes Neuronales

A continuación, vamos a ver la definición de algunos términos importantes para entender en qué consisten las redes neuronales.

3.2.1 Machine Learning

Para entender el término *Redes neuronales* [3] primero hay que tener claro que significa el concepto de *Machine Learning* [15]–[17]. *Machine Learning* es un campo de investigación y desarrollo dentro del campo de la inteligencia artificial cuyo objetivo es proporcionar a los ordenadores la capacidad de aprender sin ser explícitamente programados, para ello se desarrollan algoritmos que permitan encontrar patrones para a partir de unos datos construir un modelo que pueda predecir y clasificar elementos.

Existen tres grandes categorías dentro del *Machine Learning*:

- **Aprendizaje supervisado:** Cuando los datos de entrenamiento vienen etiquetados previamente con la solución deseada o esperada.
- **Aprendizaje no supervisado:** El conjunto de datos de entrenamiento no está etiquetado con la solución deseada y será el propio sistema el encargado de generarla.
- **Aprendizaje por refuerzo:** El sistema actuará como agente que deberá explorar un espacio desconocido y será premiado o castigado en función de sus acciones, creando así la mejor estrategia posible.

Como se ha mencionado anteriormente, el objetivo final del *Machine Learning* es generar un modelo que nos permita hacer predicciones, por lo que se pueden identificar dos fases fundamentales en el uso de esta tecnología:

- **Fase de entrenamiento:** Es aquella en la que el modelo se entrena mediante una de las categorías de aprendizaje mencionadas para generar un modelo predictivo.
- **Fase de predicción:** Es aquella en la que se utiliza el modelo generado para inducir resultados para datos nuevos de los que no conocemos la solución.

3.2.2 Redes neuronales

Las redes neuronales son un caso específico de *Machine Learning* en el que el modelo generado consiste en una estructura de datos en forma de grafo dirigido en el que los nodos son llamados neuronas y estas se agrupan en diferentes capas o *layers* [18].

Dentro de estas capas podemos diferenciar tres tipos básicos. La capa de entrada, que recibe los datos de entrada. La capa de salida, que devuelve la predicción realizada. Y las capas intermedias o capas ocultas que son las encargadas de realizar las operaciones necesarias para transformar la entrada en la predicción, que quedarían tal y como se muestra en la *Ilustración 3.9 - Ejemplo de red neuronal* (imagen original en [introducción práctica con Keras](#)).

Dentro del concepto de neurona en las redes neuronales, es importante saber que consisten en nodos de información que se encargan de transformar los datos que les vienen de las neuronas de la capa anterior y devolver esta información a la capa siguiente. Esta transformación consiste en un conjunto de fórmulas matemáticas con ciertos valores, estos valores se irán cambiando durante el proceso de aprendizaje de manera que serán los últimos los que se guarden con el modelo para realizar las predicciones.

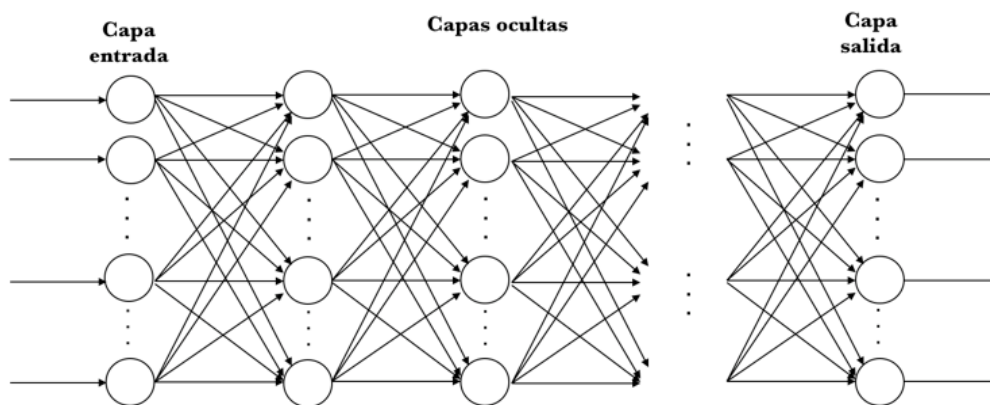


Ilustración 3.9 - Ejemplo de red neuronal

3.2.3 Redes neuronales convolucionales

Las redes neuronales convolucionales son un caso particular de redes neuronales especializadas en reconocimiento de patrones en imágenes, de manera que cada neurona de la capa de entrada representa un píxel de la imagen que va a analizar la red neuronal.

Estas redes neuronales se diferencian de las demás principalmente por utilizar capas de neuronas que realizan operaciones de *convolución* [4] (de ahí su nombre) y de *pooling* [19].

En el caso de la aplicación a desarrollar en este proyecto se ha decidido utilizar este tipo de red neuronal ya que la entrada que tenemos es una matriz de información similar a una imagen en la que necesitamos encontrar patrones para conseguir la mejor jugada en cada caso.

3.2.3.1 Operación de convolución

A modo de resumen, las capas convolucionales se encargan de detectar características en zonas concretas de la imagen, como líneas, ángulos o ciertos colores.

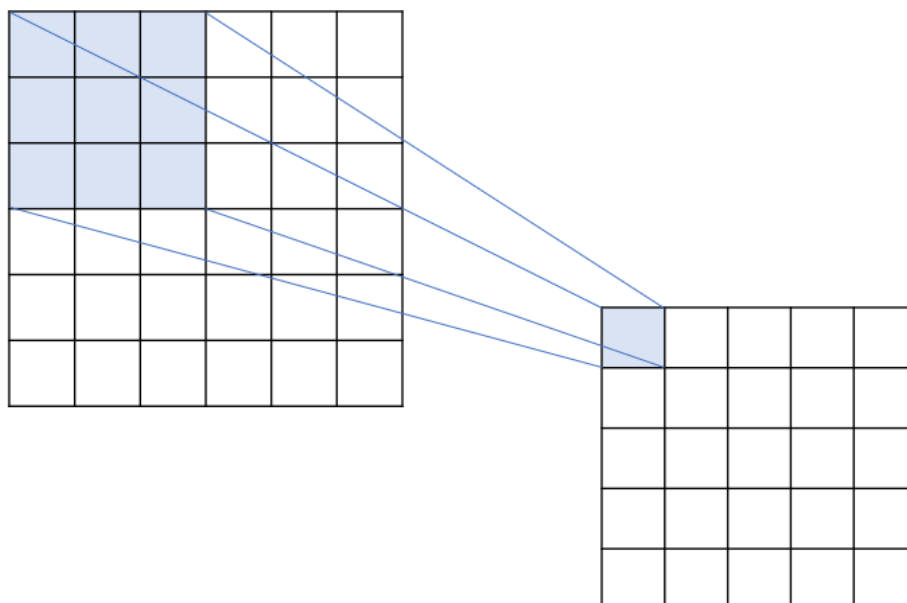


Ilustración 3.10 - Ejemplo de operación de convolución

Para realizar esta operación, se conectan un conjunto de neuronas correspondiente a un área de la capa de origen con una neurona de la capa destino, mapeando toda la matriz de la imagen tal y como se puede ver en la *Ilustración 3.10 - Ejemplo de operación de convolución*.

Como resultado, obtendríamos que las neuronas de la siguiente capa representan cada una característica de la zona correspondiente en la capa anterior.

3.2.3.2 Operación de pooling

En las redes neuronales convolucionales suele haber capas intermedias con la operación de pooling o capas de pooling. Estas capas sintetizan la información de una zona en un punto, para ello se utiliza la función pooling. Por ejemplo, la operación max-pooling transforma el valor promedio de un conjunto de píxeles a su valor máximo, como se puede ver en la siguiente imagen, que se utiliza una operación de max-pooling de 2x2.

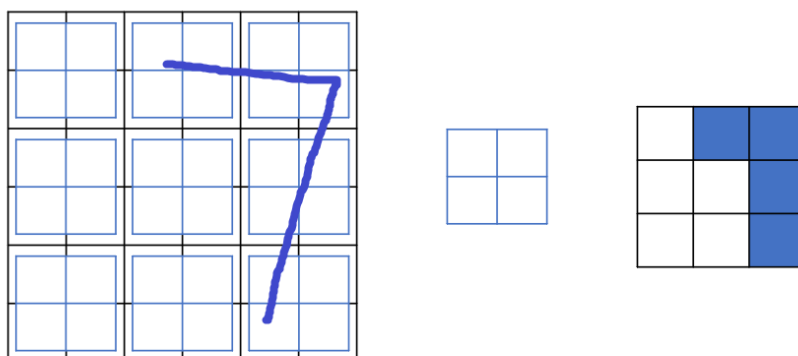


Ilustración 3.11 - Ejemplo de operación de max-pooling

Como resultado obtenemos una imagen más compacta en la que se remarcan más las características, tal y como se puede observar en la *Ilustración 3.11 - Ejemplo de operación de max-pooling*.

Capítulo 4. Planificación del Proyecto y Presupuesto

Para poder llevar a cabo el proyecto de una manera ordenada se procede a realizar una planificación en la que se estimará la duración de las tareas y se hará un reparto de estas, así como un presupuesto de cuánto costará realizar el proyecto.

4.1 Planificación

Para la realización del proyecto se han separado las tareas en 4 grandes grupos: Análisis y Diseño, Implementación, Pruebas y Documentación repartidos de la forma y con la duración que se muestra en la *Ilustración 4.1 - Diagrama de planificación*.

	Nombre de la tarea	Duración en horas
	Proyecto Completo	300
1	Análisis y Diseño	135
1.1	Definición de Objetivos	5
1.2	Definición de Requisitos	10
1.3	Presupuesto	5
1.4	Aprender uso de GUI en Python	20
1.5	Aprender sobre el uso de redes neuronales	50
1.6	Redes neuronales aplicadas a juegos	20
1.7	Diseño de Interfaces	10
1.8	Diseño de Clases	15
2	Implementación	85
2.1	Módulo de Controlador	25
2.2	Módulo de Interfaz Gráfica	20
2.3	Módulo de Red Neuronal	40
3	Pruebas	40
3.1	Diseño de Pruebas	5
3.2	Pruebas Unitarias	15
3.3	Pruebas de Integración	10
3.4	Pruebas de Usabilidad	5
3.5	Pruebas de Accesibilidad	5
4	Documentación	40
4.1	Elaboración de la Documentación	30
4.2	Revisión de la Documentación	10

Tabla 1 - Tareas de la planificación

Para poder distribuir las tareas de manera que se pudiese llegar a la fecha límite de entrega del trabajo se dividió el tiempo disponible entre las 300 horas del proyecto. La fecha límite es el día 14 de junio de 2021, y este se comenzaría la segunda semana de febrero, es decir, el 8 de febrero de 2021.

Entre estas fechas hay un total de 126 días. Como se decidió que se trabajarían todos los días de la semana por igual, para repartir las 300 hora en los 126 días habría que hacer un total de 2,38 horas al día. Para que fuese más fácil de cumplir y así además acabar antes y tener tiempo de sobra por si hubiese algún imprevisto se decidió trabajar un total de 2 hora y media cada día. Con este cálculo se prevé terminar el proyecto el día 4 de junio de 2021.

De esta forma y repatriando todas las tareas a una única persona con diferentes roles de manera consecutiva queda una planificación tal y como se muestra en el diagrama de la *Ilustración 4.1 - Diagrama de planificación*. En el caso de la documentación se irán realizando borradores durante el desarrollo del resto de las tareas, siendo la tarea final darles forma a todos.

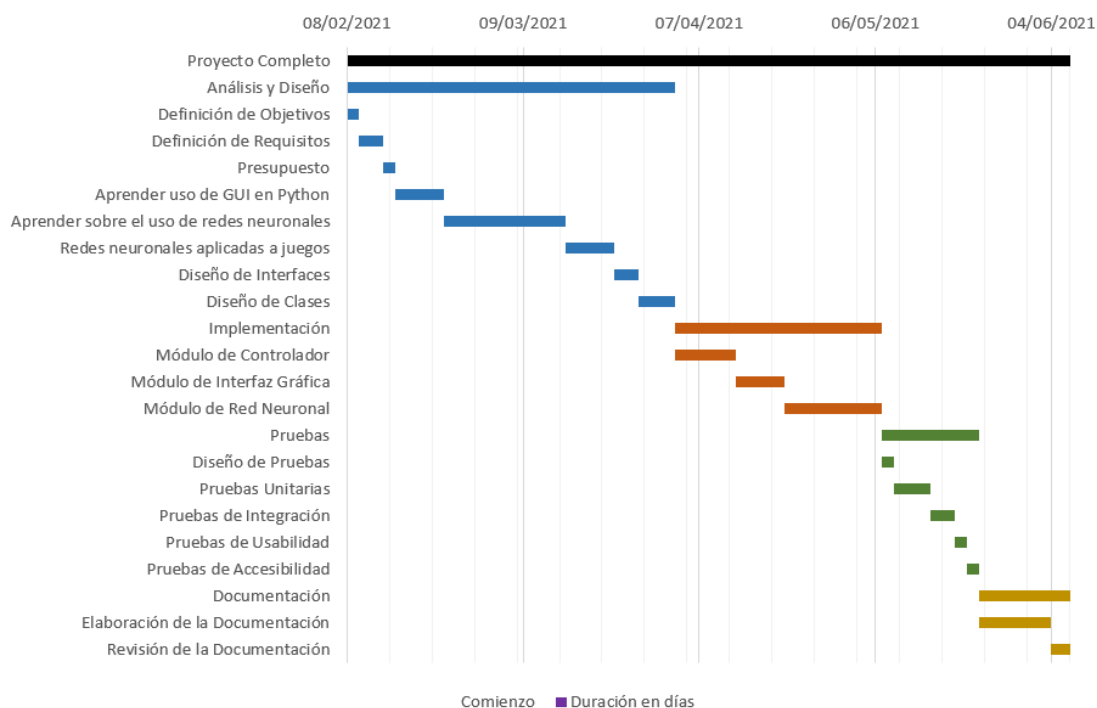


Ilustración 4.1 - Diagrama de planificación

4.2 Presupuesto

El presupuesto se va a realizar calculando los costes de manera individual para poder crear el presupuesto interno, del cual se hará un resumen que conformará el presupuesto del cliente.

4.2.1 Coste de personal

Para la elaboración del presupuesto y como se ha dicho anteriormente se contará con un único miembro en el personal que se ocupará de realizar las tareas con los diferentes perfiles profesionales, que serán los siguientes:

- **Arquitecto de Software:** Se encargará de las tareas de 1.5. *Diseño de interfaces* y 1.6. *Diseño de clases*. Con un sueldo anual de 45.000€ que corresponden a un coste salarial¹ de 58.500€ anuales, tendría un coste total de 30,47€ la hora.
- **Analista de sistemas:** Se encargará de las tareas de 1.1. *Definición de Objetivos*, 1.2. *Definición de Requisitos*, 1.3. *Presupuesto*, 1.4. *Investigación de Tecnologías existentes*, y 4. *Documentación*. Con un sueldo anual de 34.900€ que corresponden a un coste salarial¹ de 45.370€ anuales, tendría un coste total de 23,63€ la hora.
- **Desarrollador:** Se encargará de las tareas de 2. *Implementación*. Con un sueldo anual de 19.630€ que corresponden a un coste salarial¹ de 25.519€, tendría un coste total de 13,29€ la hora.
- **Tester:** Se encargará de las tareas de 3. *Pruebas*. Con un sueldo anual de 25.350€ que corresponden a un coste salarial¹ de 32.955€ anuales, tendría un coste total de 17,16€ la hora.

4.2.2 Coste de la compra de hardware

El coste de la compra de hardware es nulo ya que se trata de un proyecto íntegramente software.

4.2.3 Costes indirectos

Para los costes indirectos hay que aclarar que el software a utilizar será software libre con la excepción de Windows 10 y el paquete de office incluidos en los costes de alquiler de software. Puesto que el equipo de hardware (los ordenadores utilizados) es reutilizado de otros proyectos, se añaden los gastos de reparación de hardware correspondiente al tiempo de vida útil de estos. Quedando los costes indirectos de la siguiente manera:

¹ El cálculo del coste salarial se obtiene al añadirle un 30% al coste anual, este coste adicional se debe al coste de los seguros y demás prestaciones que debe pagar la empresa.

Servicio	Tiempo de uso (Meses)	Coste por mes	Coste total
Consumo de electricidad	4	50,00 €	200,00 €
Gastos de Internet	4	50,00 €	200,00 €
Alquiler de software	4	40,00 €	160,00 €
Gastos de material de oficina	4	40,00 €	160,00 €
Gastos de reparación de hardware	4	150,00 €	600,00 €

Tabla 2 - Costes indirectos

Con un total de 1.320,00 €.

4.2.4 Desarrollo de Presupuesto Interno

A continuación, se detalla el presupuesto completo del desarrollo del proyecto con los datos que se han especificado anteriormente para poder calcular el coste total. Cabe aclarar que el valor de la columna de total es, para las tareas compuesta de subtareas, la suma de las subtareas en las que se descompone y aparecen justo debajo en la tabla, de manera que siguen la estructura mostrada en la figura 4.1 en el [apartado 4.1](#).

Descripción	Cantidad	Unidades	Participantes	Coste por hora	Subtotal	Total
Proyecto Completo	300	horas				6.122,30 €
Análisis y Diseño	135	horas				3.361,05 €
Definición de Objetivos	5	horas	Analista de sistemas	23,63 €	118,15 €	
Definición de Requisitos	10	horas	Analista de sistemas	23,63 €	236,30 €	
Presupuesto	5	horas	Analista de sistemas	23,63 €	118,15 €	
Aprender uso de GUI en Python	20	horas	Analista de sistemas	23,63 €	472,60 €	
Aprender sobre el uso de redes neuronales	50	horas	Analista de sistemas	23,63 €	1.181,50 €	
Redes neuronales aplicadas a juegos	20	horas	Analista de sistemas	23,63 €	472,60 €	
Diseño de Interfaces	10	horas	Arquitecto software	30,47 €	304,70 €	
Diseño de Clases	15	horas	Arquitecto software	30,47 €	457,05 €	
Implementación	85	horas				1.129,65 €
Módulo de Controlador	25	horas	Desarrollador	13,29 €	332,25 €	
Módulo de Interfaz Gráfica	20	horas	Desarrollador	13,29 €	265,80 €	
Módulo de Red Neuronal	40	horas	Desarrollador	13,29 €	531,60 €	
Pruebas	40	horas				686,40 €
Diseño de Pruebas	5	horas	Tester	17,16 €	85,80 €	
Pruebas Unitarias	15	horas	Tester	17,16 €	257,40 €	
Pruebas de Integración	10	horas	Tester	17,16 €	171,60 €	
Pruebas de Usabilidad	5	horas	Tester	17,16 €	85,80 €	
Pruebas de Accesibilidad	5	horas	Tester	17,16 €	85,80 €	
Documentación	40	horas				945,20 €
Elaboración de la Documentación	30	horas	Analista de sistemas	23,63 €	708,90 €	
Revisión de la Documentación	10	horas	Analista de sistemas	23,63 €	236,30 €	

Tabla 3 - Presupuesto completo del desarrollo del proyecto

Para el cálculo del presupuesto detallado se han tenido en cuenta los siguientes datos.

- El **coste total** es de 6.122,30 €
- La **compra del hardware** supone un coste de 0€
- Se ha decidido obtener un **beneficio esperado** del 25%
- Los **costes indirectos** son 1.320,00 €

Esto supone que queremos obtener un **beneficio esperado** de 1.530,58 €. Por lo que la factura final (sumando el coste total, la compra del hardware, el beneficio esperado y los costes indirectos) ha de ser de 8.132,88 €. Para alcanzar esta cifra debemos dividir la diferencia entre todas las tareas programadas en la partida, siendo una cantidad para dividir de 2.010,58 €, que supone un incremento del 33%.

Para la realización del presupuesto simplificado se han decidido ocultar algunos costes de la partida, de manera que la partida quedaría de la siguiente manera:

Partida	Coste	Total
Análisis y Diseño		4.925,97 €
Definición de Objetivos	173,16 €	
Definición de Requisitos	346,32 €	
Presupuesto	173,16 €	
Aprender uso de GUI en Python	692,65 €	
Aprender sobre el uso de redes neuronales	1.731,61 €	
Redes neuronales aplicadas a juegos	692,65 €	
Diseño de Interfaces	446,57 €	
Diseño de Clases	669,85 €	
Implementación		1.655,62 €
Módulo de Controlador	486,95 €	
Módulo de Interfaz Gráfica	389,56 €	
Módulo de Red Neuronal	779,12 €	
Pruebas		1.005,99 €
Diseño de Pruebas	125,75 €	
Pruebas Unitarias	377,25 €	
Pruebas de Integración	251,50 €	
Pruebas de Usabilidad	125,75 €	
Pruebas de Accesibilidad	125,75 €	
Documentación		1.385,29 €
Elaboración de la Documentación	1.038,97 €	
Revisión de la Documentación	346,32 €	

Tabla 4 - Partida completa

Leyenda de colores:

Se muestra al cliente
Se ha de diluir
No se muestra al cliente

La cantidad para diluir suma un total de 1.385,29 € que supone un incremento de la parte que se va a mostrar al cliente de un 18%.

4.2.5 Desarrollo de Presupuesto Del Cliente

El presupuesto que se va a mostrar al cliente, habiendo diluido los costes que no se muestran entre los que si se muestran quedaría de la siguiente manera:

Partida	Coste	Total
Análisis y Diseño		5.825,32 €
Diseño de Interfaces	2.330,13 €	
Diseño de Clases	3.495,19 €	
Implementación		1.957,89 €
Módulo de Controlador	575,85 €	
Módulo de Interfaz Gráfica	460,68 €	
Módulo de Red Neuronal	921,36 €	
Pruebas		1.189,66 €
Diseño e implementación de pruebas	1.189,66 €	
Total		8.972,88 €

Tabla 5 - Partida simplificada

Capítulo 5. Análisis

En este apartado se desarrollará la documentación de todos los puntos realizados del análisis del sistema, los cuales se utilizarán para la realización del diseño.

5.1 Determinación del Alcance del Sistema

El objetivo principal del sistema consiste en una versión digital del juego de mesa *Hanamikoji*, en el que el jugador se enfrentará a una inteligencia artificial entrenada usando una red neuronal que aprenderá a jugar usando aprendizaje reforzado. Las reglas del juego están ya descritas por el juego original que fue publicado por primera vez en 2013 en japonés y se pueden ver en la sección de *Reglas del juego original*, por lo que se respetarán dichas reglas modificando únicamente la apariencia y los nombres de los elementos para darles un diseño más adaptado al formato digital tal y como se describe en el apartado de *Cambio de nomenclatura*.

Para llevar a cabo esto la aplicación se separará en 3 módulos diferentes que formarán parte del mismo programa, comunicándose entre sí por medio de llamadas a los métodos públicos de las diferentes clases de cada módulo.

5.1.1 Módulo de la red neuronal

Para la red neuronal contra la que se enfrentará el jugador se ha optado por una red neuronal convolucional tal y como se menciona en el apartado *Redes neuronales convolucionales*, para ello será necesario transformar la información que le llega a la red neuronal (que será el estado actual del tablero de juego) en una matriz que pueda ser interpretada por la red neuronal buscando patrones. Así mismo la información que salga de la red neuronal, al ser una red neuronal convolucional, será un vector de probabilidades, que deberá ser interpretado como una acción del juego.

Este módulo contendrá el diseño de la red neuronal convolucional, que será entrenada a partir de partidas generadas aleatoriamente de las cuales se tomarán los estados del tablero y las acciones que llevaron a una victoria. De esta manera la red neuronal aprenderá a generar acciones potencialmente ganadoras cuando reciba un estado del tablero.

5.1.2 Módulo de interfaz

Para que el usuario pueda tener una experiencia al jugar con el juego similar a la que se obtiene al jugar con el juego de cartas original, el sistema debe ofrecerle una interfaz gráfica simple y fácil de usar. Consistirá en una única ventana en la que se muestre el estado actual del tablero de manera que el usuario pueda ver todos los elementos de un vistazo rápido, con la opción de seleccionar entre las jugadas disponibles para generar su acción de juego.

5.1.3 Módulo de controladores

Ya que el *Módulo de la red neuronal* y el *Módulo de interfaz* reciben un estado del tablero y generan una acción, el módulo de controladores será el encargado integrar dicha acción con el estado actual del tablero para generar un nuevo tablero que se enviará al siguiente oponente. Además, se encargará de generar el tablero inicial y comprobar si el estado actual del tablero es de finalización y generar el resultado final de la partida.

5.2 Requisitos del Sistema

En este capítulo se realiza la identificación de los requisitos del sistema, los actores y los casos de uso para su posterior uso en el diseño del sistema.

5.2.1 Obtención de los Requisitos del Sistema

Aquí veremos los requisitos funcionales y no funcionales identificados.

5.2.1.1 Requisitos funcionales

Los requisitos funcionales se agruparán en función de la parte de la aplicación a la que pertenezcan.

5.2.1.1.1 Módulo de la red neuronal

Módulo perteneciente a la red neuronal en el que se entrenará y usará la misma.

Código	Nombre Requisito	Descripción del Requisito
R.F.1.1	Entrenamiento del modelo	La red neuronal podrá entrenarse para generar un modelo entrenado.
R.F.1.1.1	Preprocesamiento de los datos	La red neuronal recibirá un conjunto de datos de entrenamiento que deberá procesar para poder usarlos en la generación del modelo entrenado.
R.F.1.1.2	Crear el modelo de la red neuronal	La red neuronal creará un modelo que será entrenado con los datos procesados.
R.F.1.1.3	Generar y guardar el modelo entrenado	La red neuronal se entrenará con los datos procesados para generar el modelo entrenado y lo guardará para su futuro uso.
R.F.1.2	Generar predicción	La red neuronal generará una predicción de una acción.
R.F.1.2.1	Carga del modelo entrenado	La red neuronal cargará el modelo previamente entrenado para su uso en la predicción
R.F.1.2.2	Generar predicción ganadora	La red neuronal usará el modelo entrenado para generar una acción ganadora.

Tabla 6 - Requisitos funcionales de la red neuronal

5.2.1.1.2 Módulo de interfaz

Módulo que se ocupará de mostrar y recibir la información necesaria para el usuario.

Código	Nombre Requisito	Descripción del Requisito
R.F.2.1	Mostrar las acciones usadas	El usuario podrá ver que acciones han sido usadas en turnos anteriores, tanto propias como del adversario.
R.F.2.2	Mostrar los marcadores de cada guerrera	El usuario podrá ver los marcadores de cada guerrera, tanto el marcador que indica a quien debe su favor esa guerrera como el número de cartas usadas en esa guerrera por cada jugador.
R.F.2.3	Mostrar las cartas de la mano	El usuario podrá ver las cartas disponibles que tiene en su mano en todo momento.
R.F.2.4	Crear acción a realizar	El usuario podrá crear la acción que desea realizar en el turno actual.
R.F.2.4.1	Seleccionar acción	El usuario podrá seleccionar una acción dentro de las disponibles, mostrando claramente cuáles son las disponibles y las no disponibles.
R.F.2.4.2	Mostrar acción seleccionada	El usuario podrá ver la acción que ha seleccionado.
R.F.2.4.3	Cambiar la acción seleccionada	El usuario podrá cambiar la acción seleccionada antes del envío pulsando de nuevo en una de las acciones disponibles.
R.F.2.4.4	Seleccionar cartas necesarias para la acción	El usuario podrá seleccionar las cartas dentro de las disponibles en su mano para completar la acción. No se podrá seleccionar dos veces la misma carta.
R.F.2.4.5	Mostrar cartas seleccionadas	El usuario podrá ver las cartas que ha seleccionado junto a la acción seleccionada.
R.F.2.4.6	Enviar acción una vez terminada la selección	El usuario podrá enviar la acción seleccionada para que sea añadida al tablero de juego. Solo se podrá enviar la acción una vez se haya seleccionado la acción y el número exacto de cartas para esa acción.
R.F.2.5	Seleccionar cartas de la acción pendiente del adversario	El usuario podrá seleccionar la carta o cartas de la acción pendiente cuando el adversario haya seleccionado una acción que requiera esta selección. Una vez seleccionada dicha acción se podrá enviar para añadirla al tablero de juego.

Tabla 7 - Requisitos funcionales del usuario

5.2.1.1.3 Módulo de controladores

Código	Nombre Requisito	Descripción del Requisito
R.F.3.1	Creación de tablero inicial	El sistema creará el tablero inicial repartiendo las cartas iniciales en la mano de los jugadores y establecerá la configuración
R.F.3.2	Añadir una acción al tablero	El sistema se encargará de integrar la información proveniente de los jugadores en el tablero de juego
R.F.3.3	Repartir cartas nuevas	El sistema se encargará de repartir cartas a los jugadores simulando que roban una cada turno del

		mazo de cartas
R.F.3.4	Generar datos de prueba	El sistema se encargará de generar los datos de prueba necesarios para el entrenamiento de la red neuronal.

Tabla 8 - Requisitos funcionales del sistema

5.2.1.2 Requisitos no funcionales

Código	Nombre Requisito	Descripción del Requisito
R.N.F.1	Generación de un log	Se realizará un seguimiento de la ejecución del programa guardándolo con un sistema de logs.
R.N.F.2	Uso de Git	Para mantener un sistema de versiones se utilizará Git.
R.N.F.3	Uso de TensorFlow	La red neuronal será diseñada usando la librería de TensorFlow [20].
R.N.F.4	Uso de una red neuronal convolucional	Como modelo de la red neuronal se utilizará una red neuronal convolucional.
R.N.F.5	Uso de Python	Todos los módulos de la aplicación serán programados usando Python.
R.N.F.6	Guardado de datos de entrenamiento en CSV	El sistema de guardado de datos será mediante el uso de ficheros con formato CSV [21].

Tabla 9 - Requisitos no funcionales

5.2.2 Identificación de Actores del Sistema

Existe un único actor principal del sistema una vez entrenada la red neuronal, pudiendo contar como actor secundario al encargado de entrenarla, ya que se podrá configurar para que exista esta opción.

5.2.2.1 Actor principal: Usuario Jugador

El usuario jugador será el que interactúe con la aplicación con el objetivo de jugar partidas contra la red neuronal entrenada, de manera que usará la interfaz gráfica para interactuar con el tablero seleccionando las jugadas.

5.2.2.2 Actor secundario: Encargado del entrenamiento

Este actor será el que se ocupe del entrenamiento de la red neuronal. En principio solo sería necesario entrenarla una única vez, pero se deja la opción abierta de reentrenamiento, de manera que si en el futuro se consiguieran unos datos de entrenamiento más sólidos que los ofrecidos por la aplicación podrían usarse para generar nuevos valores para el modelo de la red neuronal.

5.2.3 Especificación de Casos de Uso

A continuación, se describen los casos de usos identificados para los actores del sistema.

5.2.3.1 Casos de uso para el Usuario Jugador

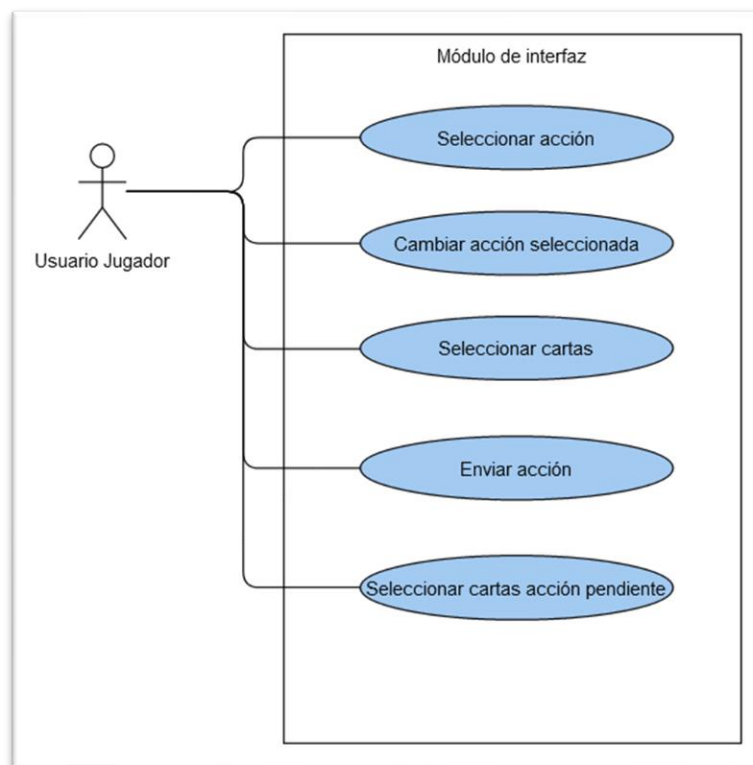


Ilustración 5.1 - Casos de uso: Usuario Jugador

5.2.3.1.1 Caso de Uso: Seleccionar acción

Nombre del Caso de Uso
Seleccionar acción
Descripción
El usuario podrá visualizar que acciones tiene disponibles y seleccionar una de ellas para poder crear la acción completa que será enviada. Una vez seleccionada la acción esta se mostrará en la zona de acción seleccionada.

Tabla 10 - Descripción del caso de uso: Seleccionar acción

5.2.3.1.2 Caso de Uso: Cambiar acción seleccionada

Nombre del Caso de Uso
Cambiar acción seleccionada
Descripción
El usuario podrá seleccionar una acción nueva (o la misma en caso de querer cambiar las cartas) una vez haya seleccionado una carta de manera que pueda modificarla las veces que necesite antes de enviarla.

Tabla 11 - Descripción del caso de uso: Cambiar acción seleccionada

5.2.3.1.3 Caso de Uso: Seleccionar cartas

Nombre del Caso de Uso
Seleccionar cartas
Descripción
El usuario podrá seleccionar desde su mano las cartas necesarias para completar la acción.

Tabla 12 - Descripción del caso de uso: Seleccionar cartas

5.2.3.1.4 Caso de Uso: Enviar acción

Nombre del Caso de Uso
Enviar acción
Descripción
El usuario podrá enviar la acción seleccionada. Esta opción solo estará disponible cuando el usuario haya seleccionado una acción dentro de las disponibles y haya seleccionado las cartas necesarias para realizar esa acción.

Tabla 13 - Descripción del caso de uso: Enviar acción

5.2.3.1.5 Caso de Uso: Seleccionar cartas acción pendiente

Nombre del Caso de Uso
Seleccionar cartas acción pendiente
Descripción
El usuario podrá seleccionar las cartas de una acción que es del adversario y requiera su interacción.

Tabla 14 - Descripción del caso de uso: Seleccionar cartas acción pendiente

5.2.3.2 Casos de uso para el encargado del entrenamiento

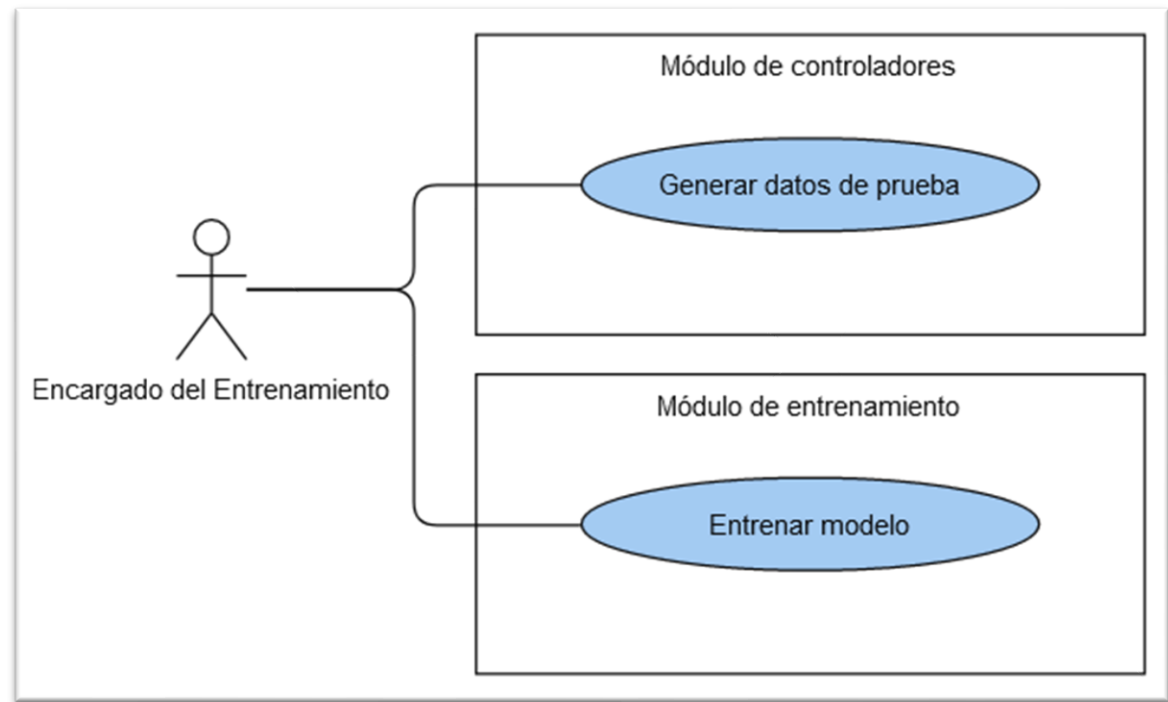


Ilustración 5.2 - Descripción del caso de uso: Encargado del Entrenamiento

5.2.3.2.1 Caso de Uso: Generar datos de prueba

Nombre del Caso de Uso	
Generar datos de prueba	
Descripción	
El encargado del entrenamiento podrá utilizar la aplicación para simular partidas de manera que se generen el número deseado de datos de prueba para que pueda entrenar al modelo.	

Tabla 15 - Descripción del caso de uso: Generar datos de prueba

5.2.3.2.2 Caso de Uso: Entrenar modelo

Nombre del Caso de Uso	
Entrenar modelo	
Descripción	
El encargado del entrenamiento podrá generar unos nuevos valores para el modelo entrenándolo con los datos que desee (en principio serán los generados en el caso de uso anterior), de manera que la aplicación quedará configurada para funcionar con esos nuevos valores a partir de ese momento.	

Tabla 16 - Descripción del caso de uso: Entrenar modelo

5.3 Identificación de los Subsistemas en la Fase de Análisis

En esta sección veremos las partes que componen los módulos del sistema desde un punto de vista de alto nivel.

5.3.1 Descripción de los Subsistemas

A continuación, se veremos la descripción de los subsistemas.

Pantallas de interacción: Forman parte del módulo de interfaz, serán las encargadas de la interacción con el usuario jugador. Existirá una pantalla principal y pantallas pop-up auxiliares.

Red neuronal: Tal como dice su nombre, forma parte fundamental del módulo de la red neuronal, será la encargada de generar las jugadas como adversario al usuario jugador.

Ficheros: Se usarán ficheros para el guardado de datos. En ellos se guardarán las jugadas necesarias para el entrenamiento de la red neuronal en formato CSV [21], [22], y el modelo de la red neuronal en formato H5.

Bot aleatorio: Forma parte de módulo de controladores, sirve para generar jugadas aleatorias, que serán utilizadas para guardar las acciones que lleven a la victoria y así poder entrenar a la red neuronal.

Generador de datos: Forma parte del módulo de controladores, usa al Bot para generar una gran cantidad de partidas y guardar las acciones que llevaron a la victoria.

Procesamiento de la partida: Forma parte del módulo de controladores, es la parte encargada de gestionar la partida y el tablero.

Estos subsistemas se comunican entre sí tal y como se muestra en la siguiente imagen:

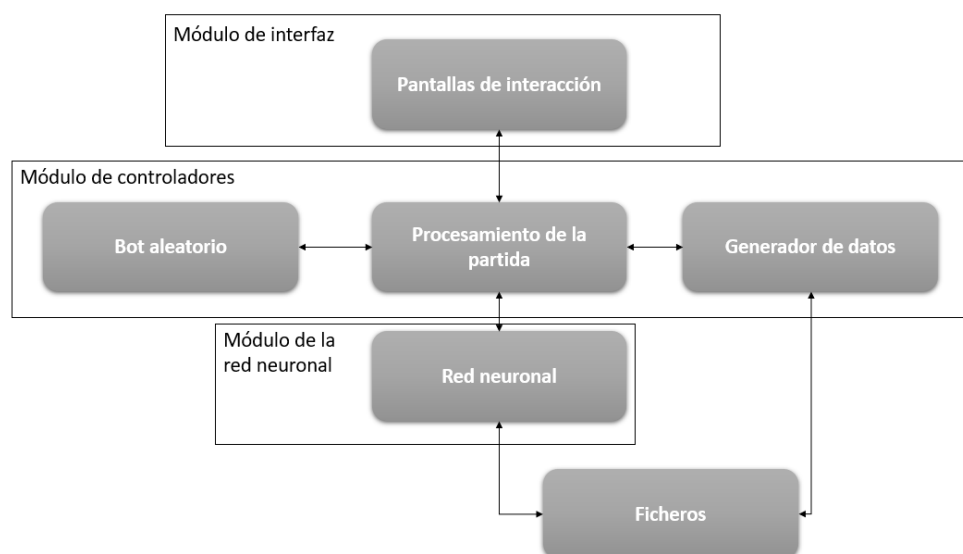


Ilustración 5.3 - Diagrama de comunicación entre los Subsistema

5.3.2 Descripción de los Interfaces entre Subsistemas

El subsistema de procesamiento de la partida es el eje central de la aplicación, se comunica con el resto de los subsistemas con una interfaz común: tanto la red neuronal, la interfaz del jugador y el Bot tendrán un método llamado *decidirAccion*, que recibe una matriz con la situación actual del tablero y devuelve un vector con la acción realizada. Dentro del módulo de la red neuronal, la parte de entrenamiento se comunica con la parte de predicción mediante unos ficheros con el modelo y los pesos.

5.4 Diagrama de Clases Preliminar del Análisis

En este apartado vamos a ver una primera aproximación al diagrama de clases. Es posible que el diagrama no se corresponda con exactitud al diagrama final de lo que se implementará.

5.4.1 Diagrama de Clases

Como se puede observar en la imagen del diagrama de clases, este sigue la separación en los 3 módulos mencionados anteriormente. Las líneas de comunicación representan las clases que interaccionan entre ellas de manera directa, excepto la de la clase predicción con la clase de entrenamiento que se comunica mediante los ficheros del modelo y los pesos.

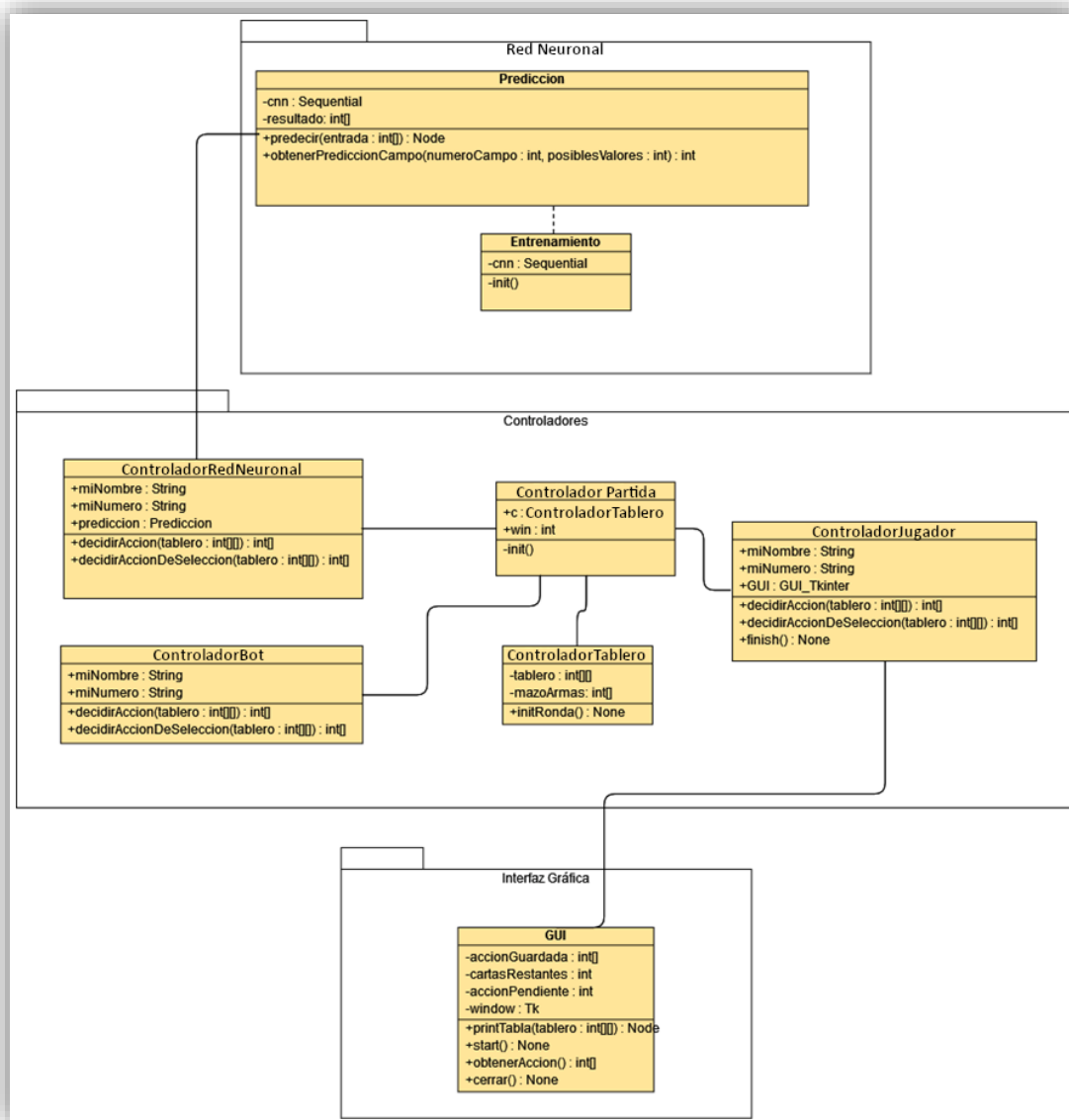


Ilustración 5.4 - Diagrama de clases

5.4.2 Descripción de las Clases

A continuación, se hará una breve descripción de los métodos de cada clase, así como de sus atributos.

5.4.2.1 Módulo de Red neuronal

Nombre de la Clase
Entrenamiento
Descripción
Esta clase contendrá la red neuronal convolucional, de manera que será la clase a la que se le pasan los datos de entrenamiento y genera el modelo entrenado para su posterior uso.
Responsabilidades
Generación del modelo y los valores del modelo.
Atributos Propuestos
cnn: Es la clase Sequential de Keras [15]. Se usará para crear el modelo
Métodos Propuestos
init: Al ser una clase únicamente funcional, no tendrá métodos públicos, ya que se hará todo desde el constructor.

Tabla 17 - Clase Entrenamiento

Nombre de la Clase
Predicción
Descripción
Esta clase contendrá la red neuronal convolucional cargada de los archivos generados por la clase de entrenamiento y servirá para generar acciones para un tablero dado.
Responsabilidades
Carga del modelo y los valores para generación de acciones.
Atributos Propuestos
cnn: Es la clase Sequential de Keras. Se usará para cargar el modelo
resultado: Es el vector de salida generado por la red neuronal
Métodos Propuestos
predecir: Dada una entrada genera y guarda un resultado para ser consultado obtenerPrediccionCampo: Dada la posición del campo y un vector de posibles valores, comprueba que valor ha devuelto la red neuronal con el método predecir

Tabla 18 - Clase Predicción

5.4.2.2 Módulo de Controladores

Nombre de la Clase
Controlador Partida
Descripción
Es la clase controladora de la partida, las rondas y la comunicación entre los demás controladores.
Responsabilidades
Controlar las rondas y comunicar las acciones entre los jugadores y el tablero
Atributos Propuestos
Controlador Tablero: Instancia del controlador del tablero
Win: Flag (bandera) para saber cuándo se ha terminado la partida y quien la ha ganado.
Métodos Propuestos
Init: Al ser la clase principal no necesita métodos públicos, todo será un único hilo de ejecución que arrancará cuando se cree la clase.

Tabla 19 - Clase Controlador Partida

Nombre de la Clase
Controlador Tablero
Descripción
Es la clase encargada de gestionar el tablero y el mazo de armas.
Responsabilidades
Gestión de acciones nuevas. Reparto de cartas.
Atributos Propuestos
Mazo de armas: Array con las cartas restantes en el mazo. Tablero: Matriz con toda la información del tablero.
Métodos Propuestos
Iniciar ronda: Para el inicio de cada ronda, reinicia el mazo de armas y reparte las cartas iniciales a cada jugador Jugador – Robar carta: Roba una carta y se la da a un jugador Jugador – Vista del tablero: Devuelve la matriz del tablero censurada para que solo se vea la información que puede ver el jugador que la pide. Jugador – Realizar acción: El jugador realizar una acción, modificando el estado del tablero

Tabla 20 - Clase Controlador Tablero

Nombre de la Clase
Controlador Bot
Descripción
Simulador de jugador que realiza acciones correctas pero aleatorias.
Responsabilidades
Responder a un tablero con una acción aleatoria
Atributos Propuestos
Mi nombre: Atributo común a todos los jugadores que sirve para identificarlos en los logs Mi número: Atributo común a todos los jugadores que sirve para saber el orden de juego
Métodos Propuestos
Decidir acción: Dado un estado del tablero genera una acción aleatoria correcta. Decidir acción de selección: Dado un estado del tablero en el que hay que seleccionar una respuesta a una acción del adversario, genera dicha respuesta de manera aleatoria.

Tabla 21 – Clase Controlador Bot

Nombre de la Clase
Controlador Red Neuronal
Descripción
Clase controladora de la red neuronal que se encarga de realizar acciones usando la red neuronal entrenada.
Responsabilidades
Responder a un tablero con una acción generada por la red neuronal
Atributos Propuestos
Mi nombre: Atributo común a todos los jugadores que sirve para identificarlos en los logs Mi número: Atributo común a todos los jugadores que sirve para saber el orden de juego Predicción: Instancia de la clase predicción de la red neuronal para poder acceder a esta
Métodos Propuestos
Decidir acción: Dado un estado del tablero genera una acción usando la red neuronal. Decidir acción de selección: Dado un estado del tablero en el que hay que seleccionar una respuesta a una acción del adversario, genera dicha respuesta usando la red neuronal.

Tabla 22 - Clase Controlador Red Neuronal

Nombre de la Clase
Controlador Jugador
Descripción
Clase controladora de la interfaz gráfica que se encarga de pedir acciones a esta para que el jugador las seleccione.
Responsabilidades
Responder a un tablero con una acción seleccionada por el jugador
Atributos Propuestos
Mi nombre: Atributo común a todos los jugadores que sirve para identificarlos en los logs Mi número: Atributo común a todos los jugadores que sirve para saber el orden de juego GUI: Instancia de la interfaz gráfica para poder acceder a esta.
Métodos Propuestos
Decidir acción: Dado un estado del tablero pide una acción a la interfaz de usuario para que el jugador la seleccione. Decidir acción de selección: Dado un estado del tablero en el que hay que seleccionar una respuesta a una acción del adversario, pide dicha acción a la interfaz de usuario para que el jugador la seleccione. Finish: Termina el proceso que mantiene la ventana de la interfaz gráfica abierta.

Tabla 23 - Clase Controlador Jugador

5.4.2.3 Módulo de Interfaz Gráfica

Nombre de la Clase
GUI
Descripción
Es la clase encargada de gestionar la interfaz con el usuario usando la librería Tkinter
Responsabilidades
Mostrar todos los elementos necesarios para que el usuario visualice su tablero y le permita generar una acción correcta.
Atributos Propuestos
Window: Instancia de la clase Tk de Tkinter en la se van a pintar los elementos Acción guardada: Array con los elementos de la acción que el usuario va seleccionando Cartas restantes: Numero de cartas que faltan para completar la acción seleccionada Acción pendiente: Numero de la acción pendiente de seleccionar cuando el usuario tiene que elegir cartas de la acción del adversario.
Métodos Propuestos
Start: Inicia el bucle de Tkinter Print tabla: Muestra por pantalla la situación actual del tablero Obtener acción: Devuelve la acción seleccionada por el usuario Cerrar: Termina el bucle de Tkinter y cierra la ventana

Tabla 24 - Clase GUI

5.5 Análisis de Interfaces de Usuario

Para realizar la interfaz de usuario se ha decidido usar diseño minimalista, de manera que toda la información esté en una única pantalla, simulando lo que sería una mesa de juego en la realidad.

5.5.1 Descripción de la Interfaz

A continuación, se muestra la estructura del tablero para la realización de la interfaz. En primer lugar la *Ilustración 5.5 - Primer esbozo (a mano) de la interfaz*, y después la *Ilustración 5.6 – Segundo esbozo (ya en digital) de la interfaz*.

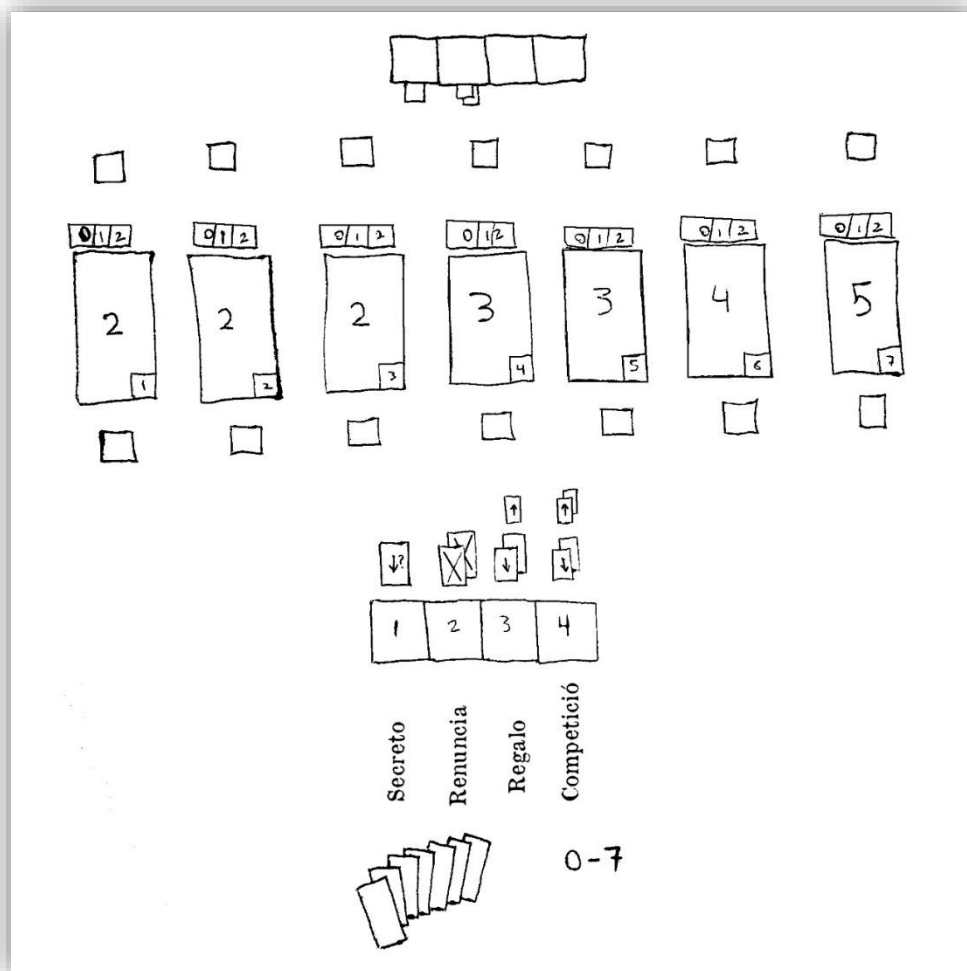


Ilustración 5.5 - Primer esbozo (a mano) de la interfaz

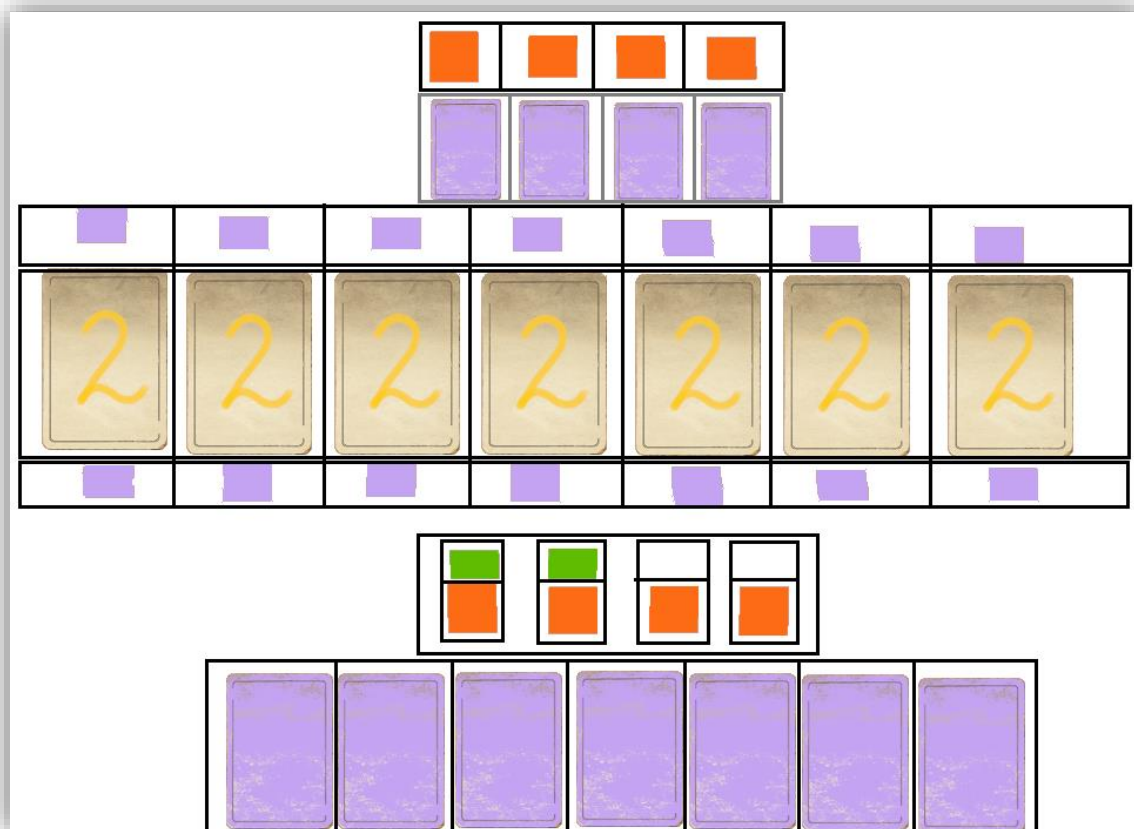


Ilustración 5.6 – Segundo esbozo (ya en digital) de la interfaz

La pantalla del tablero constará de 3 partes.

La parte superior corresponde a la información de la situación del adversario, en la que se mostrarán las acciones utilizadas por el adversario y las acciones disponibles.

En la parte central se dará información del estado del favor actual que tienen cada una de las guerreras. Sobre las cartas de las guerreras se mostrará a quien deben su favor actualmente, mientras que en la parte superior e inferior se mostrará con un contador que cantidad de armas ha ofrecido cada jugador a cada guerrera, que servirá para el conteo de final de ronda.

En la parte inferior se muestra la información del jugador, en la sé que informará sobre las acciones utilizadas, las cartas utilizadas en dichas acciones y las cartas actuales que tiene en la mano.

5.5.2 Descripción del Comportamiento de la Interfaz

Además de mostrar la información actual del tablero tal y como se describe en el apartado anterior, la interfaz de usuario también se encargará de facilitar al mismo la creación de una acción válida para poder seguir la partida.

5.5.2.1 *Acción normal*

En la parte inferior el usuario podrá seleccionar una de las acciones que le aparezcan como acciones no usadas o disponibles.

Una vez seleccionada una acción el usuario podrá cambiar de acción siempre que lo desee antes de enviar la acción completa.

Para completar una acción el usuario seleccionará las cartas necesarias para completarlas, estas cartas las podrá seleccionar desde el conjunto de cartas que tenga en la mano.

Las cartas seleccionadas se marcarán como desactivadas y aparecerán en la interfaz en la parte de la acción seleccionada.

Para reiniciar las cartas seleccionadas el usuario marcará de nuevo la misma acción, lo que hará que se eliminen todas las cartas de la parte de acción seleccionada y aparecerán como disponibles de nuevo.

Una vez el usuario haya seleccionado una acción y sus correspondientes cartas le aparecerá un botón de envío de acción. Al pulsarlo se le enviará dicha acción al controlador de la partida para que la ejecute y siga la ronda.

5.5.2.2 *Acción de selección*

Además de la selección de una acción normal la interfaz mostrará en el lugar de la acción una acción de selección cuando el usuario tenga que elegir cartas de la acción del adversario. Para esta opción el usuario únicamente podrá seleccionar un grupo de entre los dos grupos de dos cartas para la acción de tipo *Competición* y una de las tres cartas cuando tenga que elegir para una acción de tipo *Regalo*.

El usuario podrá cambiar la elección siempre que lo desee antes de enviar la acción.

Al igual que en la acción normal, una vez haya seleccionado la acción que desea, aparecerá un botón de envío para poder continuar con la ronda al ser pulsado.

Capítulo 6. Diseño del Sistema

En este apartado se describe el diseño final del sistema.

6.1 Arquitectura del sistema: Vista de bloques de construcción

En la siguiente imagen (*Ilustración 6.1 - Vista en bloques completa*) podemos ver una vista general de los bloques de construcción que se describirán en los siguientes puntos. La leyenda de la imagen se comparte con la de todos los puntos de esta sección de la documentación. Esta imagen será analizada más adelante, de manera que se podrá ver en mejor calidad.

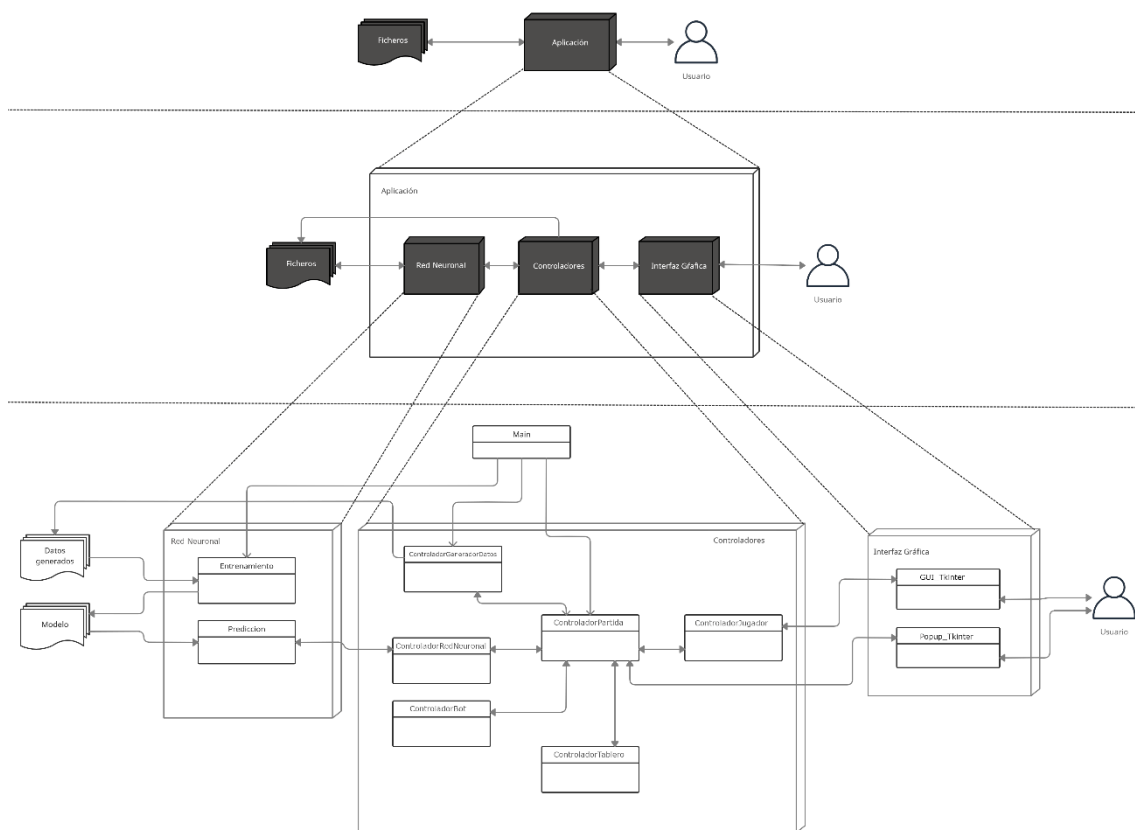


Ilustración 6.1 - Vista en bloques completa

Leyenda:





Figura del diagrama	Nombre	Descripción
 Usuario	Persona que interactúa con la aplicación	Representa a un actor o usuario que interactúa con la aplicación.
 Aplicación	Bloque de caja negra	Representa un conjunto de módulos o clases del cual no se puede ver su contenido.
 Aplicación	Bloque de caja blanca	Representa un conjunto de módulos o clases del cual se puede ver su contenido.
 Ficheros	Ficheros de caja negra	Representa un conjunto de ficheros del cual no se puede ver su contenido.
 Datos generados	Ficheros de caja blanca	Representa un fichero o conjunto de ficheros del cual se puede ver su contenido.
 Partida	Clase o archivo de funciones	Representa una clase de la aplicación o un archivo ejecutable con una serie de funciones.
	Llamada a función sin paso de datos	Representa la llamada a una función de una clase o la interacción con un elemento en la que no hay intercambio de datos.
	Llamada a función con paso de datos	Representa la llamada a una función de una clase o la interacción con un elemento en la que hay intercambio de datos.

Tabla 25 - Leyenda común a todas las vistas

6.1.1 Nivel 1

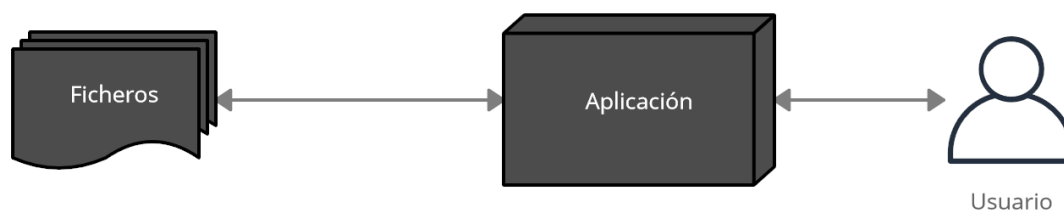


Ilustración 6.2 - Vista en bloques de nivel 1

En el nivel 1 podemos ver como la aplicación puede comunicarse y recibir la información del usuario que la esté ejecutando. Cabe destacar que en caso de que se esté ejecutando en modo generación de datos o entrenamiento de la red neuronal el usuario no interactúa con la aplicación, esta comunicación se produce únicamente cuando se ejecuta en modo jugar.

También podemos ver como la aplicación se comunica con ficheros externos. Esta comunicación sí se realiza para los 3 modos de ejecución como veremos más adelante.

6.1.2 Nivel 2

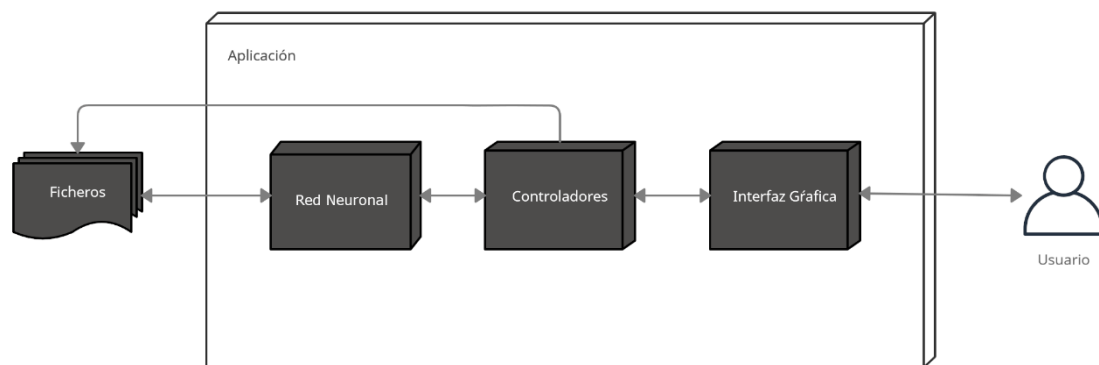


Ilustración 6.3 - Vista en bloques de nivel 2

En el nivel 2 podemos ver como la aplicación tiene 3 módulos principales.

El módulo de controladores está en el centro de ellos porque es el encargado de gestionar el resto, este se comunica con el paquete de la red neuronal tanto para pedir datos como para recibirlos, así como con el de la interfaz gráfica. También puede generar información que se guarda en ficheros.

El módulo de la red neuronal es el encargado de gestionar la parte de entrenamiento y predicción de esta. Hace uso y guarda archivos, además de comunicarse con el módulo de controladores.

El paquete de interfaz gráfica es el encargado de comunicar el módulo de controladores con el del usuario.

6.1.3 Nivel 3

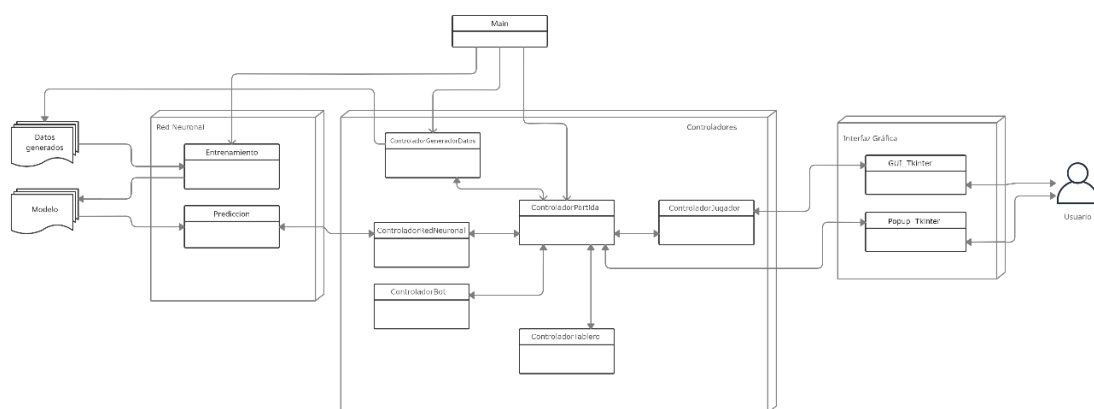


Ilustración 6.4 - Vista en bloques de nivel 3

En este nivel podemos ver todas las clases de la aplicación y la relación entre ellas.

Desde el *main* se puede llamar a 3 clases dependiendo del modo de ejecución en el que estemos. Para el modo de generación de datos se llama la clase *ControladorGeneradorDatos*, para el modo de entrenamiento de la red neuronal se llama a la clase *Entrenamiento* y para el modo jugar se llama a la clase *ControladorPartida*.

6.1.3.1 Controladores

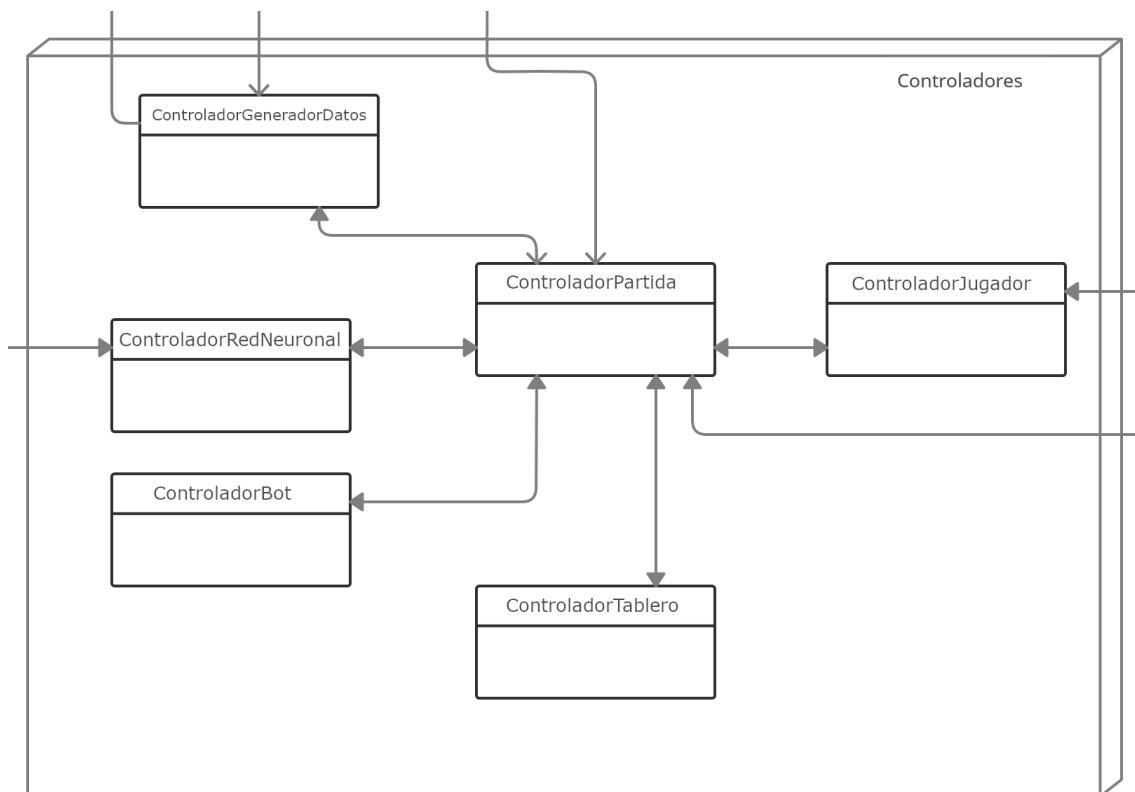


Ilustración 6.5 - Detalle del módulo de controladores de la vista en bloques de nivel 3

En el módulo de controladores tenemos 2 clases que se llaman desde el *main*.

La clase *ControladorGeneradorDatos* sirve para llamar a la clase *ControladorPartida* el número de veces que se establezcan en la parametrización, de esta clase recibe los datos de los tableros y acciones que se han realizado en cada partida y se guardan en los ficheros de datos generados.

La clase *ControladorPartida* es el centro de la aplicación, ya que es la que se encarga de gestionar y comunicar entre sí gran parte del resto de clases. En el caso de que la aplicación se ejecute en modo de generación de datos, la *ControladorPartida* será llamada un número parametrizado de veces, pero en todas ellas hará lo mismo que si fuera una ejecución en modo jugar, como podemos ver en el apartado 6.3.2 de la documentación. Recibe el tablero de la clase *ControladorTablero* y se la envía a la clase Red Neuronal (*ControladorRedNeuronal*), *ControladorJugador* o *ControladorBot* dependiendo de que jugadores estén jugando esa partida. De esas clases recibirá la acción realizada y se la pasará a clase *ControladorTablero*.

La clase *ControladorJugador* sirve de interfaz intermediaria entre la clase *ControladorPartida* y la interfaz gráfica de usuario.

La clase Red Neuronal (*ControladorRedNeuronal*) sirve de intermediaria entre la clase *ControladorPartida* y la clase de *Predicción* que forma parte del módulo de la red neuronal.

La clase *ControladorTablero* es la que se encarga de toda la gestión del tablero, lo inicializa al inicio de cada turno para enviárselo a la clase *ControladorPartida*, luego recibe las acciones que realicen los diferentes jugadores desde la clase *ControladorPartida* y guarda los cambios que realizan esas acciones en el tablero, devolviéndoselo de nuevo para la siguiente acción.

La clase *ControladorBot* sirve como jugador aleatorio. Esta clase recibe un tablero y realiza una acción completamente aleatoria dentro de las acciones posibles y con unas cartas aleatorias suficientes para realizar esa acción que estén disponibles en la mano de este jugador. Sirve como jugador inicial para el entrenamiento y como jugador de dificultad baja para el modo jugar si así se define en el archivo de parametrización.

Hay que destacar que todas las clases que sirven como jugadores de la partida tienen un patrón de métodos públicos definido. Este patrón se compone de los métodos:

- `__init__(miNombre, miNumero)`
- `decidirAccion(tablero)`
- `decidirAccionDeSeleccion(tablero)`
- `finish()`

Estos métodos serán descritos en su correspondiente clase más adelante.

6.1.3.2 Red Neuronal

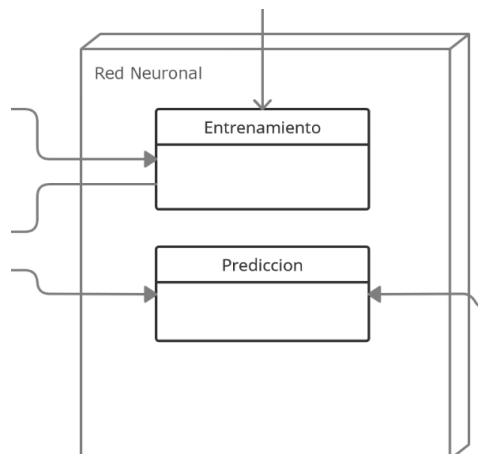


Ilustración 6.6 - Detalle del módulo de la red neuronal de la vista en bloques de nivel 3

En el módulo de la red neuronal hay únicamente 2 clases, la clase de *Entrenamiento* y la de *Prediccion*.

La clase de *Entrenamiento* se ejecuta directamente desde el *main*, lee los archivos de datos y genera los archivos del modelo.

La clase de *Prediccion* carga los archivos del modelo, recibe el tablero de la clase de red neuronal y le devuelve una predicción de una acción que será la que realice como jugador.

6.1.3.3 Interfaz grafica

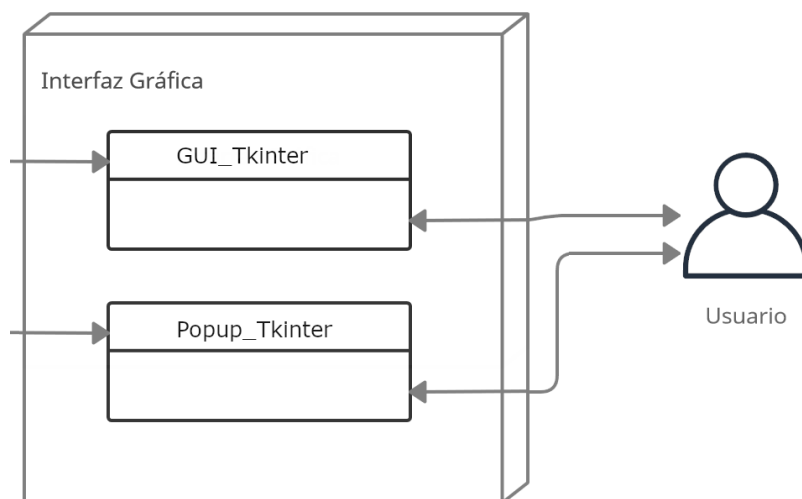


Ilustración 6.7 - Detalle del módulo de la interfaz gráfica de la vista en bloques de nivel 3

En la interfaz gráfica tenemos dos clases que utilizan Tkinter, la clase *GUI_Tkinter* lo utiliza para hacer la ventana gráfica donde el usuario interactuará como jugador con la aplicación, de manera que esta clase pinta el tablero y ofrece al usuario las posibles acciones que puede realizar. Una vez seleccionada la acción esta vuelve a los controladores para seguir la partida.

En este módulo también existe la clase auxiliar *Popup_Tkinter* sirve para informar al usuario de la finalización del programa.

6.2 Descripción detallada de las clases

A continuación, se va a hacer una descripción de las clases con sus de la aplicación separadas por los paquetes que se han explicado en el punto anterior (*Arquitectura del sistema: Vista de bloques de construcción*).

6.2.1 Controladores

Nombre de la Clase	
<i>ControladorPartida</i>	
Descripción de la clase	
Clase controladora de la partida, se encarga de los turnos y las rondas además de guardar las acciones que han llevado a cabo los jugadores.	
Atributos	
Nombre	Descripción
<code>__controladorTablero</code>	Instancia del controlador del tablero en el que se guarda el estado del tablero y se ejecutan las acciones.
<code>__accionesJ1</code>	Cadena de texto en la que se guardan las acciones que ha ejecutado el jugador 1 para su futuro guardado.
<code>__accionesJ2</code>	Cadena de texto en la que se guardan las acciones que ha ejecutado el jugador 2 para su futuro guardado.
<code>__tableroJ1</code>	Cadena de texto en la que se guardan los tableros que le llegaron al jugador 1 y sobre los cuales formo la acción correspondiente.
<code>__tableroJ2</code>	Cadena de texto en la que se guardan los tableros que le llegaron al jugador 2 y sobre los cuales formo la acción correspondiente.
<code>__winnerNumero</code>	Guarda el número del ganador al final de cada ronda para poder saber si se ha terminado la partida.
<code>__winner</code>	Copia de la instancia del jugador que ha ganado la partida.
<code>__jugador1</code>	Instancia del jugador correspondiente al orden 1.
<code>__jugador2</code>	Instancia del jugador correspondiente al orden 2.
<code>__jugador1Inicio</code>	Instancia del jugador copia de jugador 1, pero que no cambia a lo largo de la partida dependiendo del turno, se usa para declarar el ganador.
<code>__jugador2Inicio</code>	Instancia del jugador copia de jugador 1, pero que no cambia a lo largo de la partida dependiendo del turno, se usa para declarar el ganador.
Métodos	
Nombre	Parámetros y descripción
<code>__init__</code>	Método constructor de la clase <i>ControladorPartida</i> , se definen todos los atributos privados de la misma.
<code>run</code>	Método ejecutor de la clase <i>ControladorPartida</i> , inicializa los jugadores y ejecuta el método <i>start</i> con una captura de posibles errores que se mostraran en un popup.
<code>__initJugadores</code>	Método que instancia los atributos de <i>jugador1</i> y <i>jugador2</i> dependiendo del modo y nivel seleccionados en la parametrización.
<code>__start</code>	Método encargado de realizar el bucle de rondas hasta que en una de ellas haya un ganador.

<code>__ronda</code>	Método encargado de inicializar la ronda correspondiente y realizar el bucle de las acciones de los jugadores.
<code>__accion</code>	Método encargado de pedir a los jugadores que realicen una acción, si la acción requiere de interacción por parte del otro jugador también se le pedirá que haga la selección.
<code>__guardarAccion</code>	Método encargado de guardar la acción en formato cadena de texto y el tablero dados en su correspondiente atributo de la clase.
<code>getAccionesJ1</code>	Método para obtener el atributo de tipo cadena de texto: <code>accionesJ1</code> .
<code>getAccionesJ2</code>	Método para obtener el atributo de tipo cadena de texto: <code>accionesJ2</code> .
<code>getTablerosJ1</code>	Método para obtener para el atributo de tipo cadena de texto: <code>tablerosJ1</code> .
<code>getTablerosJ2</code>	Método para obtener para el atributo de tipo cadena de texto: <code>tablerosJ1</code> .
<code>getWinner</code>	Método para obtener para el atributo <code>winner</code> que implementa una clase jugador.

Tabla 26 - Descripción de la clase *ControladorPartida*

Nombre de la Clase	
<i>ControladorTablero</i>	
Descripción de la clase	
Clase controladora del tablero de juego, almacena la información de la situación actual del tablero en la que permite realizar acciones.	
Atributos	
Nombre	Descripción
<code>__tablero</code>	Guarda una matriz con el estado actual del tablero.
<code>__mazoArmas</code>	Guarda un array con las cartas que quedan en el mazo de robo.
Métodos	
Nombre	Parámetros y descripción
<code>__init__</code>	Método constructor de la clase <i>ControladorTablero</i> , se definen e inicializan todos los atributos privados de la misma.
<code>__initMazo</code>	Método que inicializa el mazo de las cartas con todas las cartas de armas disponibles.
<code>__borrarAcciones</code>	Método que se encarga de borrar del tablero las acciones realizadas por ambos jugadores.
<code>__borrarArmas</code>	Método que se encarga de borrar del tablero las armas usadas por ambos jugadores.
<code>__robarCarta</code>	Método que elimina del mazo de cartas una carta aleatoria y la devuelve. Lanza una excepción si el mazo no tiene cartas que robar.
<code>__repartoDeCartas</code>	Método que se encarga de repartir 6 cartas del mazo a cada jugador y ordenar las manos de dichos jugadores.
<code>__conseguirCarta</code>	Método que asigna una carta del mazo a la mano del jugador con el número dado por parámetro.
<code>__soltarCarta</code>	Método que recibe el array de la mano y el valor de la carta que sustituye por 0.
<code>__eliminarCarta</code>	Método que recibe un array y un valor, devuelve el array eliminando el valor una vez.

<code>__getFilaAcciones</code>	Método que recibe el índice del jugador y devuelve el índice de sus acciones usadas.
<code>__getMano</code>	Método que recibe el índice del jugador y devuelve el índice de su mano.
<code>__ordenarMano</code>	Método que recibe el índice de la mano del jugador y ordena su mano.
<code>__getNumeroCartasEnAccionSeleccionada</code>	Método que devuelve el número de cartas que contiene el array que se le pasa por parámetro, sin incluir la primera posición que corresponde al tipo de acción. Además, lanza una excepción si las cartas no están bien ordenadas en el array y contiene ceros en medio.
<code>__comprobarAccion1</code>	Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción <i>Secreto</i> o si esta acción ya ha sido usada.
<code>__comprobarAccion2</code>	Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción <i>Renuncia</i> o si esta acción ya ha sido usada.
<code>__comprobarAccion3</code>	Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción <i>Regalo</i> o si esta acción ya ha sido usada.
<code>__comprobarAccion4</code>	Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción <i>Competición</i> o si esta acción ya ha sido usada.
<code>__comprobarAccionDecision3</code>	Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción de decisión <i>Regalo</i> o si esta acción no concuerda con la esperada.
<code>__comprobarAccionDecision4</code>	Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción de decisión <i>Competición</i> o si esta acción no concuerda con la esperada.
<code>__guardarAccion1</code>	Método que guarda la carta seleccionada del array dado por parámetro para la acción <i>Secreto</i> en la fila de acciones del tablero dada por parámetro y elimina esa carta de la mano dada por parámetro.
<code>__guardarAccion2</code>	Método que guarda la carta seleccionada del array dado por parámetro para la acción <i>Renuncia</i> en la fila de acciones del tablero dada por parámetro y elimina esa carta de la mano dada por parámetro.
<code>__guardarAccion3</code>	Método que guarda las cartas seleccionadas del array dado por parámetro la acción <i>Regalo</i> en la fila de acción pendiente, establece la acción <i>Regalo</i> como usada en la fila de acciones del tablero dada por parámetro y elimina esas cartas de la mano dada por parámetro.
<code>__guardarAccion4</code>	Método que guarda las cartas seleccionadas del array dado por parámetro la acción <i>Competición</i> en la fila de acción pendiente, establece la acción <i>Competición</i> como usada en la fila de acciones del tablero dada por parámetro y elimina esas cartas de la mano dada por parámetro.
<code>__guardarAccionDecision3</code>	Método que guarda las cartas seleccionadas del array dado por parámetro para la acción de decisión <i>Regalo</i> en la fila de acción pendiente, sumamos las cartas seleccionadas a los jugadores en las filas de armas usadas y deja la fila de acción pendiente vacía.

<code>__guardarAccionDecision4</code>	Método que guarda las cartas seleccionadas del array dado por parámetro para la acción de decisión 4 en la fila de acción pendiente, sumamos las cartas seleccionadas a los jugadores en las filas de armas usadas y deja la fila de acción pendiente vacía.
<code>__guardarSecreto</code>	Método que toma el valor de la carta de secreto de la fila del jugador seleccionada y la guarda en la fila de armas del jugador seleccionada.
<code>__sumarCarta</code>	Método que dada una carta suma 1 a la columna correspondiente a esa carta y a la fila dada.
<code>__getGanador</code>	Método que calcula todos los puntos de cada jugador y devuelve el ganador.
<code>__manoLlena</code>	Método que comprueba el número de cartas en la mano dada y devuelve y está llena o no.
<code>initRonda</code>	Método que inicializa el mazo con todas las cartas menos una y reparte las cartas iniciales.
<code>jugadorRobaCarta</code>	Método que, dado el índice del jugador, roba una carta del mazo de cartas y la guarda en la mano de dicho jugador.
<code>getVistaTablero</code>	Método que, dado el índice del jugador, devuelve una matriz de información parcial para ese jugador.
<code>realizarAccion</code>	Método que, dado el índice del jugador y una acción, comprueba que ese jugador puede realizar la acción, que esta está bien formada y de ser así la ejecuta haciendo los cambios correspondientes en el tablero.
<code>hayAccionPendiente</code>	Método que, dado el índice del jugador y una acción, comprueba que ese jugador puede realizar la acción, que esta está bien formada y de ser así ejecuta haciendo los cambios correspondientes en el tablero.
<code>finalizarTurno</code>	Método que realiza todas las acciones necesarias para dar el turno por terminado y devuelve el jugador ganador en caso de que lo hubiera.

Tabla 27 - Descripción de la clase *ControladorTablero*

Nombre de la Clase	
<i>ControladorBot</i>	
Descripción de la clase	
Clase controladora del jugador de Bot aleatorio.	
Atributos	
Nombre	Descripción
<code>__miNombre</code>	Define el nombre para leerlo en los logs.
<code>__miNumero</code>	Define el orden del jugador, puede ser 1 o 2.
Métodos	
Nombre	Parámetros y descripción
<code>__init__</code>	Método constructor de la clase <i>ControladorBot</i> , recibe el nombre y el numero para guardarlo en sus respectivos atributos.
<code>decidirAccion</code>	Método para generar una acción aleatoria, recibe la matriz del tablero y devuelve un array con una acción correcta aleatoria dentro de las posibles con las cartas aleatorias que tenga en mano.
<code>decidirAccionDeSeleccion</code>	Método para generar la acción de selección pendiente con cartas aleatorias, recibe la matriz del tablero y devuelve un array con la acción correctamente formada con las cartas aleatorias dadas por

	la acción pendiente.
<i>finish</i>	Método que sirve para cerrar los hilos pendientes de los jugadores, en este caso no es necesario cerrar ninguno.
<i>getMiNombre</i>	Método para obtener para el atributo de tipo cadena de texto: <i>miNombre</i> .
<i>getMiNumero</i>	Método para obtener para el atributo de tipo cadena de texto: <i>miNumero</i> .

Tabla 28 - Descripción de la clase *ControladorBot*

Nombre de la Clase	
<i>ControladorJugador</i>	
Descripción de la clase	
Clase controladora del jugador controlado por el usuario a través de la interfaz gráfica.	
Atributos	
Nombre	Descripción
<i>__miNombre</i>	Define el nombre para leerlo en los logs.
<i>__miNumero</i>	Define el orden del jugador, puede ser 1 o 2.
<i>__GUI</i>	Instancia la clase <i>GUI_Tkinter</i> que corresponde a la interfaz gráfica de usuario con la que interactuará el usuario.
Métodos	
Nombre	Parámetros y descripción
<i>__init__</i>	Método constructor de la clase <i>ControladorJugador</i> , recibe el nombre y el numero para guardarlo en sus respectivos atributos. Además, inicializa el atributo <i>GUI</i> que implementa la clase <i>GUI_Tkinter</i> .
<i>decidirAccion</i>	Método para generar una acción, recibe la matriz del tablero y devuelve un array con una acción correcta que será seleccionada por el usuario a través del Método <i>pedirAccion</i> .
<i>decidirAccionDeSeleccion</i>	Método para generar la acción de selección pendiente, recibe la matriz del tablero y devuelve un array con la acción correctamente formada con las cartas seleccionadas por el usuario a través del Método <i>pedirAccion</i> .
<i>__pedirAccion</i>	Método que unifica los Métodos de <i>decidirAccion</i> y <i>decidirAccionDeSeleccion</i> en el que se llama a los Métodos de <i>printTabla</i> , <i>start</i> y <i>obtenerAccion</i> para pintar el tablero en la interfaz de usuario y esperar a que el usuario seleccione la acción que desee.
<i>finish</i>	Método que sirve para cerrar el hilo de la interfaz gráfica.
<i>getMiNombre</i>	Método para obtener para el atributo de tipo cadena de texto: <i>miNombre</i> .
<i>getMiNumero</i>	Método para obtener para el atributo de tipo cadena de texto: <i>miNumero</i> .

Tabla 29 - Descripción de la clase *ControladorJugador*

Nombre de la Clase	
<i>ControladorRedNeuronal</i>	
Descripción de la clase	
Clase controladora del jugador controlado por la red neuronal entrenada anteriormente.	
Atributos	
Nombre	Descripción
<code>__miNombre</code>	Define el nombre para leerlo en los logs.
<code>__miNumero</code>	Define el orden del jugador, puede ser 1 o 2.
<code>__prediccion</code>	Instancia la clase <i>Prediccion</i> que corresponde a la parte de la red neuronal encargada de generar predicciones.
Métodos	
Nombre	Parámetros y descripción
<code>__init__</code>	Método constructor de la clase <i>ControladorRedNeuronal</i> , recibe el nombre y el numero para guardarlo en sus respectivos atributos. Además, inicializa el atributo <i>Prediccion</i> que implementa la clase <i>Prediccion</i> .
<code>decidirAccion</code>	Método para generar una acción, recibe la matriz del tablero y devuelve un array con una acción correcta que será seleccionada por la red neuronal y procesada por el Método <i>procesarAccion</i> .
<code>__procesarAccion</code>	Método encargado de procesar el resultado emitido por la red neuronal para transformarlo en una acción correctamente formada y válida para el tablero actual. Esto se debe a que la red neuronal devuelve el valor en porcentajes de acierto, que no están exentos de fallos.
<code>__obtenerCartasEnMano Y AccionesPosibles</code>	Método encargado de devolver una lista con las cartas que hay en la mano y otra lista con las acciones posibles que puede realizar el jugador para el tablero dado. Este método es necesario para el método <i>procesarAccion</i> .
<code>__obtenerAccionCount</code>	Método que devuelve el número de cartas que debe tener la acción a realizar dada por parámetro. Este método es necesario para el método <i>procesarAccion</i> .
<code>__eliminarCarta</code>	Método que devuelve un array con las cartas en mano a la que se le ha retirado la carta seleccionada. Este método es necesario para el método <i>procesarAccion</i> .
<code>decidirAccionDe Seleccion</code>	Método para generar la acción de selección pendiente, recibe la matriz del tablero y devuelve un array con la acción correctamente formada con las cartas seleccionadas por red neuronal y procesada por el método <i>procesarAccionDeSeleccion</i> .
<code>__procesarAccionDe Seleccion</code>	Método encargado de procesar el resultado emitido por la red neuronal para transformarlo en una acción de selección correctamente formada y válida para el tablero actual. Esto se debe a que la red neuronal devuelve el valor en porcentajes de acierto, que no están exentos de fallos.
<code>__seleccionarCartas AccionDeSeleccionRegalo</code>	Método encargado de procesar el resultado emitido por la red neuronal para la acción de selección específica de tipo <i>Regalo</i> para una acción pendiente dada.
<code>__seleccionarCartasAccion</code>	Método encargado de procesar el resultado emitido por la

<i>DeSeleccionCompeticion</i>	red neuronal para la acción de selección específica de tipo <i>Competición</i> para una acción pendiente dada.
<i>__crearAccionCompleta</i>	Método que crea un array acción a partir del tipo de acción y las cartas seleccionadas.
<i>finish</i>	Método que sirve para cerrar los hilos pendientes de los jugadores, en este caso no es necesario cerrar ninguno.
<i>getMiNombre</i>	Método para obtener para el atributo de tipo cadena de texto: <i>miNombre</i> .
<i>getMiNumero</i>	Método para obtener para el atributo de tipo cadena de texto: <i>miNumero</i> .

Tabla 30 - Descripción de la clase *ControladorRedNeuronal*

Nombre de la Clase	
<i>ControladorGeneradorDatos</i>	
Descripción de la clase	
Clase controladora de la generación de datos para el entrenamiento de la red neuronal.	
Atributos	
Nombre	Descripción
<i>__partidasGanadas1</i>	Define el número de partidas que ha ganado el primer jugador.
<i>__partidasGanadas2</i>	Define el número de partidas que ha ganado el segundo jugador.
<i>__controladorPartida</i>	Instancia la clase <i>ControladorPartida</i> , que se va instanciando con cada nueva simulación.
Métodos	
Nombre	Parámetros y descripción
<i>__init__</i>	Método constructor de la clase <i>ControladorGeneradorDatos</i> , se inicializan los atributos de la clase.
<i>run</i>	Método ejecutor de la clase <i>ControladorGeneradorDatos</i> , reinicia los archivos y ejecuta partidas un número de veces parametrizado.
<i>__guardarGanador</i>	Método que dado un jugador obtiene los datos de su partida y los guarda en los ficheros de partidas ganadas.
<i>__resetArchivo</i>	Método que elimina los archivos generados anteriormente y los deja limpios para una nueva generación.
<i>__guardarEnArchivo</i>	Método de escritura en fichero del texto dado.

Tabla 31 - Descripción de la clase *ControladorGeneradorDatos*

6.2.2 Red Neuronal

Nombre de la Clase	
<i>Entrenamiento</i>	
Descripción de la clase	
Clase encargada de utilizar los datos de entrenamiento para guardar el modelo entrenado.	
Atributos	
Nombre	Descripción
<i>__cnn</i>	Contendrá una instancia de la clase <i>Sequential</i> de TensorFlow.

Métodos	
Nombre	Parámetros y descripción
<code>__init__</code>	Método constructor de la clase <i>Entrenamiento</i> , se define el atributo <i>cnn</i> .
<code>run</code>	Método ejecutor de la clase <i>Entrenamiento</i> , se realiza el preprocesado de datos, la creación del modelo y el entrenamiento.
<code>__preProcesadoDeDatos</code>	Método encargado de cargar los datos de entrenamiento desde los ficheros y transformarlos en matrices y arrays que reconoce la red neuronal.
<code>__creacionModelo</code>	Método encargado de instanciar el atributo <i>cnn</i> con la clase <i>Sequential</i> .
<code>__establecerCapas</code>	Método encargado de definir las capas y filtros que va a tener la red neuronal.
<code>__compileAndFit</code>	Método encargado de compilar el modelo para su entrenamiento.
<code>__guardarModelo</code>	Método encargado de guardar el modelo generado con los pesos de este en los archivos .h5 para su posterior carga por parte de la clase <i>Prediccion</i> .

Tabla 32 - Descripción de la clase Entrenamiento

Nombre de la Clase	
<i>Prediccion</i>	
Descripción de la clase	
Clase encargada de utilizar el modelo entrenado para generar predicciones.	
Atributos	
Nombre	Descripción
<code>__cnn</code>	Contendrá una instancia de la clase <i>Sequential</i> .
<code>__resultado</code>	Contendrá el resultado de la última predicción realizada.
Métodos	
Nombre	Parámetros y descripción
<code>__init__</code>	Método constructor de la clase <i>Entrenamiento</i> , se define los atributos e inicializarlos.
<code>__cargarModelo</code>	Método encargado de cargar el modelo entrenado de los ficheros .h5.
<code>predecir</code>	Método que, dado un tablero de entrada, genera una predicción y la guarda en el atributo resultado.
<code>obtenerPrediccionCampo</code>	Método que, dado un el índice del campo del que se desea obtener la predicción realizada y los posibles valores aceptables, devuelve el valor más probable.

Tabla 33 - Descripción de la clase Prediccion

6.2.3 Interfaz Gráfica

Nombre de la Clase	
<i>GUI_Tkinter</i>	
Descripción de la clase	
Clase encargada de generar la ventana principal de la aplicación usando la librería Tkinter.	

Atributos	
Nombre	Descripción
<code>__accionGuardada</code>	Array en el que se van guardando la información de la acción que se está generando.
<code>__cartasRestantes</code>	El número de cartas que faltan para terminar de completar la acción actual.
<code>__accionPendiente</code>	Array con la acción pendiente que llega.
<code>__window</code>	Instancia la clase <i>Tk</i> de Tkinter.
Métodos	
Nombre	Parámetros y descripción
<code>__init__</code>	Método constructor de la clase <i>GUI_Tkinter</i> , se definen e inicializan todos los atributos privados de la misma.
<code>printTabla</code>	Método encargado de pintar el tablero que nos llega.
<code>__limpiarAccion</code>	Método encargado de vaciar el array de acción guardada y borrar del tablero la acción que se había seleccionado.
<code>__comprobarAccionPendiente</code>	Método encargado de comprobar si existe una acción pendiente y modificar el tablero para que se muestre la misma.
<code>__bloquearAccionesNormalesYMano</code>	Método encargado de deshabilitar todos los botones de las acciones y de la mano.
<code>__addGuerreras</code>	Método encargado de añadir una fila de imágenes de cartas grandes.
<code>__addMarcadores</code>	Método encargado de añadir una fila de marcadores dada una lista de números.
<code>__addSusAcciones</code>	Método encargado de añadir una fila con las acciones usadas del adversario.
<code>__addMisAcciones</code>	Método encargado de añadir una fila con las acciones usadas del jugador.
<code>__addMiMano</code>	Método encargado de añadir una fila con las cartas de la mano del jugador.
<code>__addAccionSeleccionada</code>	Método encargado de añadir el label con la acción seleccionada.
<code>__addAccionPendiente</code>	Método encargado de añadir las cartas de la acción pendiente.
<code>__addAccionPendiente5</code>	Método encargado de añadir las cartas de la acción pendiente para el tipo <i>Regalo</i> .
<code>__addAccionPendiente6</code>	Método encargado de añadir las cartas de la acción pendiente para el tipo <i>Competición</i> .
<code>__addCartaGrande</code>	Método para añadir una imagen de carta grande en la posición dada con la imagen dada.
<code>__addMarcadorValor</code>	Método para añadir el marcador del valor en la posición dada con la imagen dada.
<code>__addMarcador</code>	Método para añadir marcadores en la posición dada con el texto dado.
<code>__addAccionPropia</code>	Método para añadir un botón, en la posición dada, de acción activa/inactiva dependiendo del tipo dado.
<code>__addAccionEnemiga</code>	Método encargado de añadir una imagen de acción realizada/no realizada en la posición dada.
<code>__getTextoAccion</code>	Método encargado de seleccionar el texto para un tipo dado.
<code>__addCartaPeque</code>	Método encargado de añadir un botón con una carta

	pequeña.
<code>__addCartaAccionPendiente</code>	Método encargado de añadir un botón de una carta de acción pendiente.
<code>__addCartaOculto</code>	Método encargado de añadir un botón con una carta oculta.
<code>__addMarcoExplicativo</code>	Método encargado de añadir un botón con una carta oculta.
<code>__addLabelConImagen</code>	Método encargado de añadir un label con una imagen para una posición, con un tamaño, imagen, texto y borde dados y numero de filas = 1.
<code>__addLabelConImagenY TamanoFilas</code>	Método encargado de añadir un label con una imagen para una posición, con un tamaño, imagen, texto, numero de filas y borde dados.
<code>__addLabelConSoloTexto</code>	Método encargado de añadir un label in imagen para una posición, con un tamaño, texto y borde dados.
<code>__addBotonConImagen</code>	Método encargado de añadir un botón con una imagen para una posición, con un tamaño, imagen, texto, numero de filas y borde dados.
<code>__addButtonConTexto</code>	Método encargado de añadir un botón con un texto para una posición, con una imagen, texto y borde dados.
<code>__seleccionarAccion</code>	Método que se ejecuta al dar a un botón de selección de acción. Elimina la acción anteriormente seleccionada y pinta la nueva para poder seleccionar las cartas correspondientes.
<code>__seleccionarCarta</code>	Método que se ejecuta al seleccionar una carta, en caso de que la acción admita una nueva carta pinta la carta seleccionada en la sección de la acción seleccionada y bloquea la carta de la mano.
<code>__seleccionarCarta Pendiente5</code>	Método que se ejecuta al seleccionar una carta cuando esta la acción de selección pendiente de tipo <i>Regalo</i> .
<code>__seleccionarCarta Pendiente6</code>	Método que se ejecuta al seleccionar una carta cuando esta la acción de selección pendiente de tipo <i>Competición</i> .
<code>__noAccion</code>	Método auxiliar para no hacer nada al pulsar acciones que no están disponibles.
<code>__printAceptar</code>	Método que pinta el botón de aceptar para enviar la acción seleccionada.
<code>__borrarAceptar</code>	Método que borra el botón de aceptar para que no se envíe una acción a medias.
<code>start</code>	Método que inicia el bucle de Tkinter.
<code>__pressAceptar</code>	Método sale del bucle de Tkinter para poder seguir con la ejecución del programa y aplicar la acción seleccionada.
<code>cerrar</code>	Método que destruye la ventana de Tkinter cuando se termina la partida.
<code>obtenerAccion</code>	Método que devuelve el atributo de <i>accionGuardada</i> .

Tabla 34 - Descripción de la clase *GUI_Tkinter*

Nombre de la Clase	
<i>Popup Tkinter</i>	
Descripción de la clase	
Clase encargada de generar la ventana pequeña de información (o popup).	
Métodos	
Nombre	Parámetros y descripción
<i>sendMensaje</i>	Método que construye el popup con el texto dado y un botón de

	aceptar para cerrarlo.
--	------------------------

Tabla 35 - Descripción de la clase *Popup_Tkinter*

6.2.4 Ficheros de parametrización y configuración

Además de las clases descritas anteriormente, cabe mencionar que existen ficheros Python que no contienen clases, pero en los que se encuentran funciones y parámetros que se utilizan en la aplicación.

Nombre del fichero
<i>ParametrosCNN</i>
Descripción de la clase
Contiene los parámetros necesarios para la configuración y entrenamiento de la red neuronal convolucional.

Ilustración 6.8 - Descripción del fichero *ParametrosCNN*

Nombre del fichero
<i>ParametrosDatos</i>
Descripción de la clase
Contiene los parámetros de las direcciones de los archivos de generación y guardado de datos de la red neuronal.

Ilustración 6.9 - Descripción del fichero *ParametrosDatos*

Nombre del fichero
<i>ParametrosGUI</i>
Descripción de la clase
Contiene los parámetros necesarios para la configuración de la red neuronal convolucional.

Ilustración 6.10 - Descripción del fichero *ParametrosGUI*

Nombre del fichero
<i>ParametrosImagenes</i>
Descripción de la clase
Contiene los parámetros referentes a las direcciones y nombre de los archivos de imágenes utilizados por la aplicación.

Ilustración 6.11 - Descripción del fichero *ParametrosImagenes*

Nombre del fichero
<i>ParametrosMenu</i>
Descripción de la clase
Contiene los parámetros necesarios arrancar la aplicación con los diferentes modos que se leen del archivo <i>param.properties</i> donde están definidos.

Ilustración 6.12 - Descripción del fichero *ParametrosMenu*

Nombre del fichero
<i>ParametrosTablero</i>
Descripción de la clase
Contiene los parámetros referentes a las posiciones de las cartas dentro de la matriz del tablero.

Ilustración 6.13 - Descripción del fichero *ParametrosTablero*

6.3 Vista de tiempo de ejecución

En esta sección se va a describir el flujo de la aplicación durante su ejecución.

6.3.1 Opciones

Como hemos visto anteriormente existen 3 opciones parametrizables a la hora de ejecutar la aplicación. Estas opciones se arrancan desde el mismo *main* y llaman a su clase inicial correspondiente.

- Generar datos
- Entrenar a la red neuronal
- Jugar

La vista de tiempo de ejecución de entrenar a la red neuronal es la más simple de las 3, ya que únicamente se establece el modelo de la red neuronal y se arranca, no hay más que el uso de una sola clase.

Las vistas de generación de datos y jugar es prácticamente la misma, únicamente se diferencian en quienes son los jugadores de cada partida y el número de partidas que se juegan.

Para la opción de generación de datos el *main* llama a la clase *ControladorGeneradorDatos*, que se encarga de hacer las llamadas a la clase *ControladorPartida* y guardar los datos en un archivo.

Para la opción de Jugar se llama desde el *main* directamente a la clase *ControladorPartida* que es la encargada de gestionar el ciclo completo de ejecución de una partida.

6.3.2 Ciclo de una partida

Como acabamos de ver la clase *ControladorPartida* se encarga de gestionar la partida con un bucle que realiza una ronda completa hasta que uno de los 2 jugadores gana.

Cada ronda se compone de a su vez de un bucle de 4 turnos tal y como se ve representado en la Ilustración 6.14 - Diagrama de flujo de una partida.

Al inicio de cada ronda la clase *ControladorPartida* pide a la clase *ControladorTablero*.

En cada ciclo del turno cada jugador realiza una acción, siempre en el mismo orden de jugadores para cada turno e intercambiando el orden para cada ronda. El jugador inicial recibe el estado actual del tablero y decide que acción realizar. Si la acción es la acción de *Secreto* o la acción de

Renuncia (es decir, es una acción simple) esta acción llega a la clase *ControladorPartida* que se encarga de enviarla a la clase *ControladorTablero* para que haga los cambios correspondientes en el tablero y devolvérselo a la clase *ControladorPartida*. Si la acción es de *Regalo* o *Competición* (es decir, una acción compleja) después de que el jugador realice la acción y la clase *ControladorTablero* devuelva el resultado en el tablero igual que en una acción simple, la clase *ControladorPartida* envía la acción de decisión en el tablero al jugador 2 para que elija las cartas, este una vez ha elegido las cartas le devuelve esa acción a la clase *ControladorPartida* y esta vuelve a enviársela a la clase *ControladorTablero* para que guarde dicha decisión en el tablero.

Una vez terminada la acción del jugador 1 se realiza la acción del jugador 2.

A modo de resumen, cada partida consta de al menos 1 ronda. Cada ronda consta de 4 turnos en los que se realizan 2 acciones, una por cada jugador.

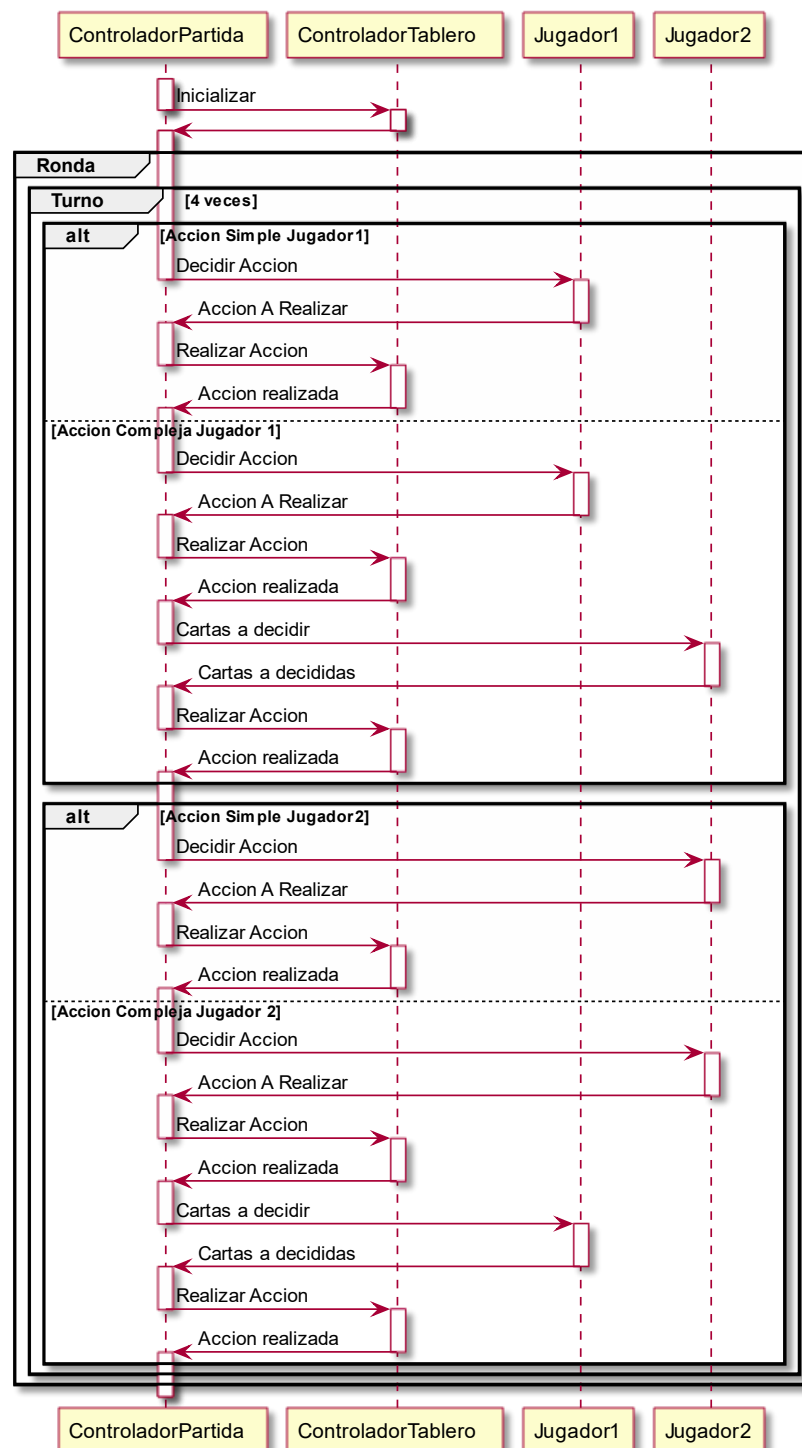


Ilustración 6.14 - Diagrama de flujo de una partida

6.4 Diseño de la Red Neuronal

En esta sección se va a analizar capa por capa el diseño de la red neuronal.

6.4.1 Transformación de información en datos utilizables por la red neuronal

El diseño de la red neuronal comienza con el preprocesamiento de datos. Al ser una red neuronal convolucional, en un principio se pensó en pasarle una captura de la imagen del tablero y dejar que la red neuronal interpretase las cartas y situación general de este. Pero al final se decidió que, ya que esta información la teníamos que usar para realizar el controlador del tablero, se podía configurar la red neuronal para que obtuviese esa información directamente.

Por lo tanto, el preprocesado de datos comienza incluso fuera de la propia clase de entrenamiento de la red neuronal, empieza en la obtención y guardado de datos.

El objetivo de la red neuronal va a ser recibir un tablero con la información de la situación actual del tablero y devolver una acción concreta. La situación actual del tablero se va a representar como una matriz de datos, donde cada fila representa una parte de la información, y la acción un array, donde la primera posición representa la acción a realizar y el resto las cartas usadas en dicha acción.

La matriz del tablero se distribuirá de la siguiente manera:

- La primera fila contiene las cartas de la mano del jugador (máximo de 7).
- La segunda fila contiene las cartas de la mano del adversario.
- La tercera fila contiene las acciones realizadas por el jugador.
- La cuarta fila contiene las acciones realizadas por el adversario.
- La quinta fila contiene los contadores de armas usadas por el jugador (Para cada guerrera guarda la cantidad de armas que le ha dado el jugador).
- La sexta fila contiene los contadores de armas usadas por el adversario (Para cada guerrera guarda la cantidad de armas que le ha dado el adversario).
- La séptima fila contiene el favor actual de las guerreras (0: Nadie, 1: Jugador 1, 2: Jugador 2).
- La octava fila contiene la acción de decisión en caso de haber una acción pendiente.

De esta manera tenemos una matriz de 8 filas y 7 columnas (este número depende del número máximo de cartas en la mano) en la que se contiene toda la información del tablero.

Para el entrenamiento de la red neuronal, la matriz completa será truncada, ya que hay datos a los que un jugador no tiene acceso, como la mano del adversario y las cartas que ha usado en las acciones de tipo *Secreto* y

Renuncia.

Ejemplo de matriz del tablero:

```
0000023
0000000
4001100
1111100
0001213
0110122
0201210
0000000
```

- 0000023 -> Cartas en la mano del jugador (2 y 3).
- 0000000 -> Mano vacía del adversario.
- 4001100 -> Acciones realizadas por el jugador (ha realizado las acciones *Secreto* (usando la carta 4), *Regalo* y *Competición*).
- 1111100 -> Acciones realizadas por el adversario (ha realizado todas sus acciones).
- 0001213 -> Armas utilizadas por el jugador (1 en la guerrera 4, 2 en la guerrera 5, 1 en la guerrera 6 y 3 en la guerrera 7).
- 0110122 -> Armas utilizadas por el adversario (1 en la guerrera 2, 1 en la guerrera 3, 1 en la guerrera 5, 2 en la guerrera 6 y 2 en la guerrera 7).
- 0201210 -> Favor de las guerreras (el jugador 1 gana en las 4 y 6 y el jugador 2 en las 2 y 5).
- 0000000 -> Acción de decisión, al estar vacía indica que hay que realizar una acción normal.

Para el entrenamiento de la red neuronal se utilizarán como datos un conjunto de tableros con las acciones correspondientes, acciones que llevaron a la victoria. Así la red neuronal se encargará de aprender a devolver estas acciones cuando se le presente ese tablero.

Para ello necesitamos definir también la forma de las acciones.

Con la información descrita por el tablero anterior, la red neuronal realizará una acción dentro de las 4 posibles o una acción de decisión dentro de las 2 posibles. Además, todas las acciones van acompañadas de un número de cartas entre 1 y 4 dependiendo de la acción. Por lo tanto, la acción completa se enviará como un array de 5 campos, donde el primer campo es la acción por realizar (0, 1, 2 o 3 para las acciones normales, 4 para la acción de decisión de la acción de *Regalo* y 5 para la acción de decisión de la acción 4) y el resto de los campos las cartas seleccionadas (0 en caso de que la acción no requiera todos los campos).

Ejemplo de acción:

13200 -> El jugador realiza la acción de

Renuncia con las cartas 3 y 2.

6.4.2 Preprocesamiento de datos

En el preprocesamiento de los datos se leerán los datos obtenidos de las simulaciones realizadas y guardados en los documentos .CSV.

Como la información se ha guardado en un array, para que cada línea del CSV corresponda a un tablero, el primer paso es deshacer este cambio, volviendo a transformar el array en una matriz de entrada, para que en el futuro la red neuronal entrenada pueda obtener también una matriz.

6.4.3 Definición de las capas de la red neuronal

A continuación, se van a enumerar las capas que forman la red neuronal.

6.4.3.1 Primera capa

Esta es una capa de tipo *Input*, sirve exclusivamente para definir qué entrada va a recibir la red neuronal. En este caso se define el tamaño de los datos de entrada: (8, 7, 1), es decir, va a recibir una matriz de 8 de altura, 7 de ancho y 1 de profundidad, que corresponde a la matriz del tablero.

6.4.3.2 Segunda capa

La segunda capa de la red neuronal consiste en una capa de convolución: *Convolution2D*. Como ya se explicó en el punto 3.2.3.1 *Operación de convolución*, la función de esta capa va a ser encontrar patrones dentro de los datos de entrada.

Al resultado dado por la capa anterior, se le van a aplicar un filtro de tamaño 1,7 para que vaya recorriendo las filas de la tabla y así poder encontrar patrones en estas. El tipo de *padding* es *valid*, ya que no queremos que añada 0 en los bordes, estos bordes implican la finalización de los datos y añadir información podría alterar el resultado esperado.

6.4.3.3 Tercera capa

La tercera capa es una capa de pooling: *MaxPooling2D*. Como ya se explicó en el punto 3.2.3.2 *Operación de pooling*, la función de esta capa obtener una matriz de información más compacta en la que se remarcan más las características.

Al resultado dado por la capa anterior, se le va a realizar una operación de max-pooling de 2,2 que correspondería a algo similar a la *Ilustración 3.11 - Ejemplo de operación de max-pooling*.

6.4.3.4 Cuarta capa

Esta capa es una capa de tipo *Flatten* [23], que va a realizar transformación tomando el resultado de la capa anterior y lo va a transformándolo en un array.

Esta transformación es necesaria para la siguiente capa.

6.4.3.5 Quinta capa

Esta capa es una capa densa: *Dense* [24]. Este tipo de capas se utilizan para modificar el número de neuronas en relación con la capa anterior.

En este caso se va a realizar una ampliación de neuronas a un total de 448×4 , que corresponde a un escalado de la proporción inicial de neuronas.

6.4.3.6 Sexta capa

Esta capa es una capa de desactivación: *Dropout* [25]. Este tipo de capas se utiliza para desactivar cierto número de neuronas. Esto se hace para que en cada simulación haya algunas neuronas que no se utilicen, haciendo que sea más complicado que la red neuronal utilice siempre un mismo camino de neuronas atrofiando el resultado y haciendo que haya neuronas que no se utilicen nunca.

En este caso se va a aplicar una desactivación del 50% de las neuronas, para así obligar a la red neuronal a que pueda encontrar varios caminos para una misma solución esperada.

6.4.3.7 Séptima capa

Esta capa vuelve a ser una capa densa: *Dense*. Tiene una función similar a la de la *Quinta capa*, solo que en este caso se va a buscar un escalado correspondiente al número inicial de neuronas (8×7) para poder repetir el proceso.

6.4.3.8 Octava capa

Esta capa es una capa de reescalado: *Reshape* [26]. Tiene la función inversa a la capa de tipo *Flatten*, en este caso se va a reescalar para que la siguiente capa tenga una entrada de dimensión: (8, 7, 1).

6.4.3.9 Novena capa

Esta capa es de tipo *Convolution2D*, realiza una repetición de la *Segunda capa* para volver a buscar patrones que puedan ser útiles en la búsqueda de la solución esperada.

6.4.3.10 Decima capa

Esta capa es de tipo *MaxPooling2D*, realiza una repetición de la *Tercera capa* para volver a buscar patrones que puedan ser útiles en la búsqueda de la solución esperada.

6.4.3.11 Undécima capa

Esta capa es de tipo *Flatten*, va a realizar transformación tomando el resultado de la capa anterior y lo va a transformándolo en un array.

6.4.3.12 Duodécima capa

Esta capa vuelve a ser una capa densa: *Dense*. Tiene una función similar a la de la *Quinta capa*, solo que en este caso se va a buscar un escalado correspondiente al número final de neuronas para poder obtener el resultado final.

6.4.3.13 Decimotercera capa

Esta capa es una capa de reescalado: *Reshape*. Tiene la función similar a la de la *Octava capa*, en este caso se va a reescalar para que la salida de la red neuronal sea (5, 8), es decir el número de salidas será 5 y el número de posibilidades de salidas para cada salida será 8.

6.4.4 Procesamiento del resultado

Al realizar una predicción con la red neuronal nos va a devolver una matriz de 5x8, en la que tendremos el resultado del array de 5 con los 8 valores posibles puntuados con un porcentaje de estimación al resultado querido.

De esta forma, y para transformar ese resultado en una acción correcta, debemos procesar el resultado.

En primer lugar, se obtiene la acción a realizar en base el valor del primer array de los 5. Para que la acción sea correcta y no de error al ser procesada en el futuro, se calculan las posibles acciones que se pueden realizar, una vez realizado este cálculo se obtiene el valor del porcentaje (guardado en el resultado de la predicción) para estas acciones y se toma el porcentaje más alto. Esta será la acción realizada.

Una vez obtenida la acción realizada se calcula el número de cartas requeridas para esa acción, y se repite el proceso para cada una de las cartas (se busca en el array de valores predichos por la red neuronal los porcentajes más altos para las cartas posibles dentro de las cartas en mano).

Por último, ya con todos los datos obtenidos, se forma el array de la acción realizada por la red neuronal, llenando los espacios extra con 0 para seguir el patrón establecido.

6.5 Diseño de la Interfaz

La interfaz gráfica se compone de una única pantalla en la que el usuario tiene una visión parcial de la situación actual del tablero.

Se compone de una matriz de 7x7 casillas donde cada fila corresponde a un conjunto diferente de datos que explicaremos a continuación.



Ilustración 6.15 - Ejemplo de interfaz de usuario

6.5.1 Acciones del adversario



Ilustración 6.16 - Ejemplo de acciones del adversario sin utilizar

Al inicio de la partida el adversario no habrá utilizado ninguna de las acciones, por lo que se mostraran todas en un color oscuro indicando esto tal y como se muestra en la *Ilustración 6.16 - Ejemplo de acciones del adversario sin utilizar*.



Ilustración 6.17 - Ejemplo de acciones del adversario utilizadas

Cuando el adversario utilice alguna acción esta pasará a estar en un color azul más claro.

Existe un espacio entre la acción de *Secreto* y la acción de

Renuncia que representa la carta que el adversario haya usado para la acción de *Secreto*. Si no se ha utilizado el hueco estará vacío, mientras que si se ha utilizado aparecerá una carta gris, ya que es una carta oculta y el jugador no puede saber que carta es.

También existen dos espacios entre la acción de

Renuncia y la de *Regalo* que representan las dos cartas de renuncia del adversario en caso de haber usado ya esta acción al igual que con la carta de secreto.

6.5.2 Guerreras, armas usadas y favor

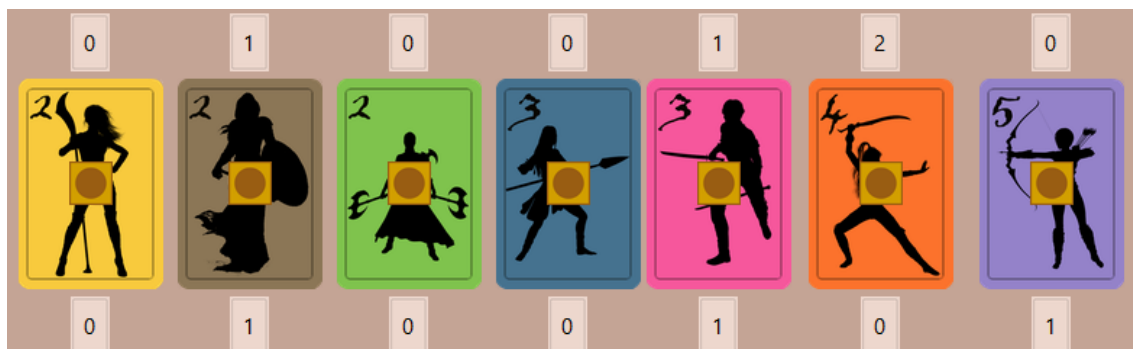


Ilustración 6.18 - Ejemplo de visualización de las guerreras

Estas 3 filas están orientadas, es decir, la parte superior corresponde al adversario y la parte inferior corresponde al jugador.

La fila del medio representa a las guerreras, y el marcador central representan su favor actual. Al inicio de la partida este favor estará siempre en el centro representando que son neutrales. Sin embargo, tras la primera ronda, si ningún jugador ha ganado el favor cambiará dependiendo de las acciones de la ronda anterior de manera que estará en la parte superior de la carta de la guerrera en caso de que esa guerrera este a favor del adversario (*Ilustración 6.19 - Ejemplo de guerrera con el favor del adversario*) o en la parte inferior en caso de este a favor del jugador (*Ilustración 6.20 - Ejemplo de guerrera con el favor del jugador*).



Ilustración 6.19 - Ejemplo de guerrera con el favor del adversario



Ilustración 6.20 - Ejemplo de guerrera con el favor del jugador

Los marcadores numéricos que aparecen encima y debajo de las cartas de las guerreras representan el número de cartas que cada jugador ha jugado en favor de esa guerrera durante esta ronda. Los marcadores superiores corresponden al adversario y los inferiores al jugador. Hay que tener en cuenta, las acciones de *Secreto* y

Renuncia guardan las cartas en la parte de acciones usadas y no se tienen en cuenta hasta el final de la ronda.

6.5.3 Acciones del jugador



Ilustración 6.21 - Ejemplo de acciones del jugador sin utilizar

Esta fila es muy similar a la de acciones del adversario al inicio, pero tiene dos diferencias fundamentales. Para empezar las acciones son interactivas, por lo que el usuario puede pulsar encima de ellas para seleccionarla. Cuando se seleccione una acción aparecerá marcada más abajo. Si se desea se puede cambiar de acción seleccionando una diferente, además

seleccionar de nuevo una acción reinicia las cartas que hayan sido seleccionadas para esta acción.



Ilustración 6.22 - Ejemplo de acciones del jugador utilizadas

Al igual que con las acciones del adversario cuando ya han sido utilizadas en esta ronda se marcan en azul más claro, pero a diferencia del adversario en esta ocasión si se podrán ver las cartas seleccionadas para las acciones de *Secreto* y

Renuncia. Además, las acciones utilizadas dejan de ser interactivas, ya que el usuario no puede volver a utilizarlas.

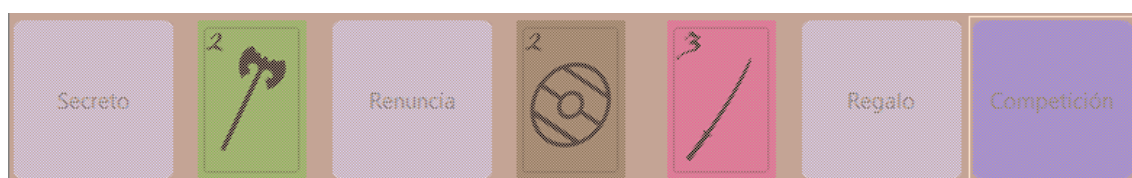


Ilustración 6.23 - Ejemplo de acciones del jugador bloqueadas

Cuando el jugador se encuentre con una acción de decisión en la que tenga que elegir cartas del adversario, las acciones del jugador quedarán bloqueadas, tornándose en un tono más grisáceo, pero mostrando diferencia entre las acciones usadas y no usadas, así como las cartas usadas para las acciones de *Secreto* y

Renuncia para poder tomar una decisión con toda la información del tablero.

6.5.4 Mano del jugador



Ilustración 6.24 - Ejemplo de mano del jugador con todas las cartas disponibles

En esta fila encontramos las cartas que tiene en la mano el jugador. El color y el símbolo representan a que guerrera corresponden. Estas cartas son interactivas, para poder seleccionarlas hay que haber elegido antes una acción, y dependiendo de la acción elegida se podrán seleccionar un numero distinto de cartas.



Ilustración 6.25 - Ejemplo de mano del jugador con cartas seleccionadas

Cuando se seleccione alguna carta esta quedará bloqueada, de manera que no se podrá volver a seleccionar. Si se desea cambiar de cartas se puede volver a seleccionar la acción borrando así las cartas seleccionadas.

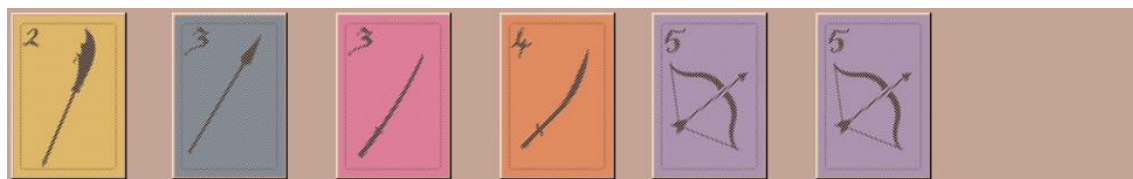


Ilustración 6.26 - Ejemplo de mano del jugador bloqueada

Cuando el jugador se encuentre con una acción de decisión en la que tenga que elegir cartas del adversario, las cartas del jugador quedarán bloqueadas al igual que las acciones.

6.5.5 Acción elegida



Ilustración 6.27 - Ejemplo de acción elegida

Esta fila aparecerá en un principio vacía, pero irá cambiando, dependiendo de la acción y las cartas que seleccione el jugador. No es interactiva, ya que el jugador no puede pulsar en ninguna casilla. En caso de que el jugador tenga que realizar una acción de decisión esta aparecerá en esta fila.

6.5.5.1 Acción por realizar

En la primera casilla se muestra la acción que ha seleccionado el jugador. A continuación, se muestran las cartas que ha seleccionado el jugador. Es importante destacar que para la acción de *Competición* el orden en el que se seleccionen las cartas tiene relevancia, ya que las dos primeras formarán un grupo y las dos segundas el otro grupo. Para el resto de las acciones el orden es indiferente.

6.5.5.2 Acción de decisión



Ilustración 6.28 - Ejemplo de acción de decisión

Cuando el jugador tenga que realizar una acción de decisión en el lugar de la acción aparecerá una breve descripción de lo que debe hacer. En caso de ser una acción de *Regalo* tendrá que seleccionar una carta de las 3, la cual aparecerá bloqueada después de ser seleccionada, pudiendo seleccionar cualquiera de las otras dos si el jugador cambia de opinión.

En caso de ser una acción de *Competición* el jugador podrá elegir entre el grupo de las dos primeras o el grupo de las dos últimas. Al pulsar sobre una carta se quedan marcadas las dos cartas de ese grupo, pudiendo cambiar de grupo en caso de cambio de opinión.

6.5.6 Botón de aceptar

En último lugar tenemos el botón de aceptar, este botón aparece únicamente cuando el jugador ha seleccionado una acción y el correspondiente grupo de cargas o ha seleccionado las cartas que desea en la acción de decisión. En caso de que el jugador cambie de acción y las cartas seleccionadas se reinicien el botón desaparecerá hasta que el jugador vuelva a seleccionar el número correcto de cartas.

Al pulsar este botón la acción queda registrada y se envía a la aplicación, para que el adversario realice su acción y le vuelva a tocar el turno al jugador.

6.5.7 Pantalla de información final

Cuando se acaba la partida se cierra la pantalla principal y aparece una pequeña ventana anunciando el resultado de la partida.

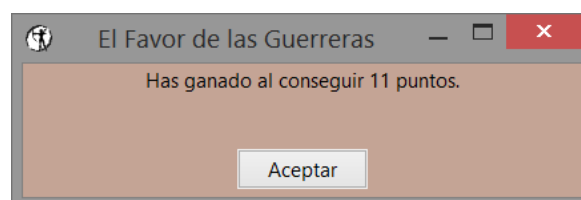


Ilustración 6.29 - Ventana de información de resultado

Capítulo 7. Implementación del Sistema

En esta sección se va a realizar una descripción de los estándares, lenguajes de programación, herramientas y programas utilizados para la implementación del sistema, así como una descripción del proceso de creación.

7.1 Estándares y Normas Seguidos

A continuación, se van a describir brevemente los estándares utilizados en la aplicación para su desarrollo.

7.1.1 Arquitectura de N-capas

La arquitectura basada en capas [27] consiste en una diferenciación de roles de forma jerárquica. En el caso de este proyecto se ha implementado gracias a la separación de los 3 módulos existentes:

- *Módulo de interfaz*
- *Módulo de controladores*
- *Módulo de la red neuronal*

Para el caso de este proyecto, el módulo de controladores es el eje central, esta capa se comunica con el módulo de la red neuronal y con el módulo de la interfaz, pero estas últimas capas no pueden comunicarse entre sí.

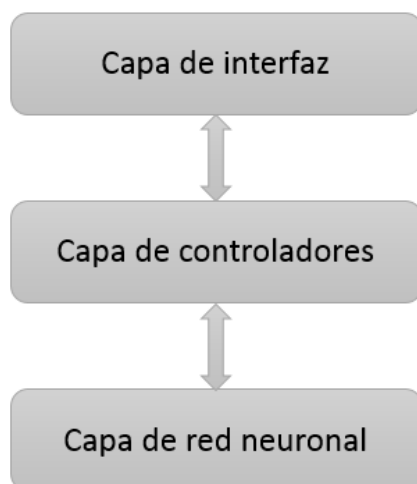


Ilustración 7.1 - Representación del modelo de n-capas de la aplicación

7.1.2 Estándar de mensajería CSV

El formato CSV [21] (*Comma Separated Value*, o Valores Separados por Comas) es un formato muy utilizado para intercambio de información entre bases de datos o aplicaciones.

En esta aplicación se ha utilizado el formato CSV para guardar los datos generados por las simulaciones de partidas.

Este formato tiene una gran ventaja en la integración con sistemas de Python y el uso de las redes neuronales.

7.2 Lenguajes de Programación

En esta sección se va a describir el lenguaje de programación usados, su versión y algunas librerías importantes dentro del mismo.

7.2.1 Python

Se ha utilizado Python [2] en la versión 3.7.9 en todo el proyecto. Como ya se comentó en la sección 2.2 *Evaluación de alternativas tecnológicas*, al utilizar la librería TensorFlow [20] para la implementación de la red neuronal, que está disponible para Python, se decidió utilizar este lenguaje de programación para toda la aplicación, incluyendo el Módulo de interfaz y el Módulo de controladores.

7.2.1.1 Librería TensorFlow

La librería de TensorFlow [20] ha sido un pilar fundamental en la implementación de la aplicación. Se ha utilizado la versión 2.3.0 para Python.

7.3 Herramientas y Programas Usados para el Desarrollo

En este apartado vamos a describir los programas que se han utilizado para facilitar el desarrollo de la aplicación y el seguimiento del proyecto.

7.3.1 Anaconda

Anaconda [28] es un programa de distribución libre y abierta que se utiliza mucho en ciencia de datos, y aprendizaje automático. Permite tener un control sobre las librerías y aplicaciones del entorno, así como sus versiones. En el caso de este proyecto toda la configuración de anaconda está guardada en el archivo `anaconda.yaml` tal y como se indica en el apartado del *Despliegue del entorno*.



Ilustración 7.2 - Logotipo del programa Anaconda

7.3.2 Spyder

Spyder [29] es un entorno de desarrollo integrado multiplataforma de código abierto para programación científica en el lenguaje Python. Es uno de los programas que vienen integrados con Anaconda y es muy útil para utilizar las librerías instaladas en Anaconda.



Ilustración 7.3 - Logotipo del programa Spyder

7.3.3 Sourcetree

Sourcetree [30] es un programa de administración de repositorios que utiliza git para gestionar las versiones de un proyecto. Para este proyecto se ha creado un repositorio de git subido en github en el que están todos los cambios por los que ha pasado el proyecto.



Ilustración 7.4 - Logotipo del programa Sourcetree

7.3.4 Gimp

Gimp [31] es un programa de edición de imágenes libre y gratuito. Forma parte del proyecto GNU. En este proyecto se ha utilizado para poder modificar y crear las imágenes usadas tanto en la aplicación como en la documentación.



Ilustración 7.5 - Logotipo del programa Gimp

7.4 Creación del Sistema

En esta sección se van a detallar los aspectos que nos hemos encontrado durante el desarrollo del proyecto

7.4.1 Problemas Encontrados

A continuación, se van a describir los problemas encontrados.

7.4.1.1 Desconocimiento sobre la realización del diseño de una red neuronal

Aunque desde un principio el aprendizaje sobre redes neuronales estaba contemplado, resultó más complicado de lo previsto. Aunque ya se conocían términos y conceptos básicos sobre las redes neuronales a nivel divulgativo, esta información estaba muy alejada al conocimiento necesario para la implementación desde cero de una red neuronal.

Caben destacar dos campos importantes donde se notó una falta de conocimiento.

En primer lugar, la transformación de los conceptos básicos sobre redes neuronales en su uso con herramientas como TensorFlow. Una vez se había aprendido como se implementa una red neuronal, a la hora de hacerlo también había que conocer la librería y los parámetros con los que se maneja para poder transformar un diseño en un desarrollo.

En segundo lugar, a la hora de diseñar una red neuronal no es suficiente con tener un conocimiento del funcionamiento de estas, se requiere una experiencia, ya que para obtener un resultado satisfactorio se debe realizar un diseño apropiado. Desgraciadamente para realizar ese diseño no existe una metodología común a todos los desarrollos, por lo que

requiere “intuición” para realizar un diseño que funcione en el ámbito deseado, y esa “intuición” solo se obtiene con la experiencia de haber realizado muchos proyectos similares.

7.4.1.2 *Poca experiencia en el lenguaje de programación utilizado*

Aunque desde un principio se conocía que la decisión de utilizar Python, y el *¡Error! No se encuentra el origen de la referencia.* era una de las motivaciones, esto también era un riesgo, por lo que cabe destacarlo en este apartado. No fue un problema que retrasase el desarrollo del proyecto, pero fue una dificultad con la que se tuvo que lidiar.

7.4.1.3 *Problemas en los resultados obtenidos*

Una vez realizado el desarrollo y la implementación del proyecto no se ha podido realizar una de las motivaciones del proyecto: *¡Error! No se encuentra el origen de la referencia.* En el resultado final de la aplicación no se han podido encontrar patrones claros que haya desarrollado la red neuronal.

Hay diferentes posibles causas por la que se puede haber dado este problema, y no está claro cuál de ellas ha sido o si ha sido el conjunto de todas.

Para empezar el uso de la clase *ControladorBot* (ver: *Tabla 28 - Descripción de la clase ControladorBot*) para la generación de datos supone un posible problema. Al ser este “Bot” un generador de acciones aleatorias es posible que los datos generados, aunque se guarden solo aquellos de las partidas ganadas, no sean suficientemente coherentes como para llegar a obtener patrones de victoria. Es decir, que estos datos pueden resultar en que la red neuronal aprenda a realizar acciones aleatorias en vez de acciones que la lleven a la victoria. Este problema tiene dos posibles soluciones, utilizar otros datos de entrenamiento (posible ampliación, ver: *Nuevo entrenamiento con nuevos datos*), o bien cambiar el enfoque del entrenamiento de la red neuronal (ver: *Nuevo modo de entrenamiento con redes neuronales adversarias*).

Otra posible causa es un mal diseño de la red neuronal. Como se ha mencionado anteriormente, al realizar este proyecto se tenía un *Desconocimiento sobre la realización del diseño de una red neuronal*. Esta posible causa requeriría de un aprendizaje y obtención de experiencia en la implementación de redes neuronales a base de la realización de varios proyectos de esta temática. Por eso se deja abierta la posibilidad de un *Nuevo diseño de la red neuronal* en el futuro, si se ha obtenido la experiencia suficiente como para poder realizar un mejor diseño.

Por último, existe la posibilidad de que realmente el juego no tenga patrones ganadores. Al ser un juego de cartas en el que el azar y la interacción con el adversario juegan un papel fundamental, puede que no existan patrones como en el Ajedrez o en el Go que supongan una ventaja para el jugador y una manera más fácil de obtener la victoria. Ya que es un juego poco conocido, aunque es divertido, la victoria puede ser una cuestión de suerte y de intentar engañar al rival para que se piense que las cartas ocultas que has jugado sean unas que te benefician, y estos parámetros no son aplicables al caso del desarrollo de este proyecto.

7.4.2 Descripción Detallada de las Clases

Las clases del proyecto pueden encontrarse en el fichero adjunto.

En la ruta `\src\main\python` se encuentran las clases de implementación de la aplicación.

En la ruta `\src\main\recursos` se encuentran los ficheros estáticos y de configuración de la aplicación.

En la ruta `\src\test\python` se encuentran las clases de las pruebas unitarias de la aplicación.

En la ruta `\src\main\recursos\entrenamiento` se encuentran los ficheros `.h5` con el modelo y los pesos del modelo de la red neuronal.

Capítulo 8. Desarrollo de las Pruebas

En este capítulo se va a describir el diseño y resultado de las pruebas realizadas.

8.1 Pruebas Unitarias

A continuación, se van a describir el diseño y el estado de las pruebas unitarias realizadas.

8.1.1 Pruebas del controlador del Bot

Estas pruebas corresponden a la clase *ControladorBot*, separadas por los métodos a probar.

8.1.1.1 Método: seleccionar acción

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
4 acciones libres con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que las 4 acciones están disponibles y tiene 7 cartas distintas en la mano a elegir. Devuelve una acción cualquiera con cartas existentes en la mano.	1	Correcto	OK
4 acciones libres con 4 cartas iguales para elegir.	El método recibe un tablero de entrada en el que las 4 acciones están disponibles y tiene 4 cartas iguales en la mano a elegir. Devuelve una acción con la carta repetida de la mano.	2	Correcto	OK
4 acciones libres con 0 cartas para elegir.	El método recibe un tablero de entrada en el que las 4 acciones están disponibles y tiene 0 cartas en la mano a elegir. Devuelve una excepción al no poder seleccionar ninguna carta.	3	Excepción	OK
Acción libre de tipo 1 con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Secreto</i> libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción <i>Secreto</i> con cartas existentes en la mano.	4	Correcto	OK
Acción libre de tipo 1 con 1 carta para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Secreto</i> libre y tiene solo 1 carta en mano a	5	Correcto	OK

	elegir. Devuelve la acción <i>Secreto</i> con la carta de la mano.			
Acción libre de tipo 2 con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo Renuncia libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción 2 con cartas existentes en la mano.	6	Correcto	OK
Acción libre de tipo 2 con 2 cartas iguales para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo Renuncia libre y tiene solo 2 cartas, las 2 iguales, en mano a elegir. Devuelve la acción 2 con las cartas de la mano.	7	Correcto	OK
Acción libre de tipo 3 con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Regalo</i> libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción 3 con cartas existentes en la mano.	8	Correcto	OK
Acción libre de tipo 3 con 3 cartas iguales para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Regalo</i> libre y tiene solo 3 cartas, todas iguales, en mano a elegir. Devuelve la acción 3 con las cartas de la mano.	9	Correcto	OK
Acción libre de tipo 4 con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Competición</i> libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción 4 con cartas existentes en la mano.	10	Correcto	OK
Acción libre de tipo 4 con 4 cartas iguales para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Competición</i> libre y tiene solo 4 cartas, todas iguales, en mano a elegir. Devuelve la acción 4 con las cartas de la mano.	11	Correcto	OK
Sin acciones libres.	El método recibe un tablero de entrada en el que no tiene acciones libres y tiene solo 4 cartas, todas iguales, en mano a elegir. Devuelve una excepción.	12	Excepción	OK

Tabla 36 - Pruebas para el método: seleccionar acción

8.1.1.2 Método: seleccionar acción selección

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
Es de tipo 3 con todas las cartas son distintas.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Regalo</i> y tiene todas las cartas iguales. Devuelve la acción de selección <i>Regalo</i> con una de las cartas posibles.	13	Correcto	OK
Es de tipo 3 con algunas cartas iguales.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Regalo</i> y tiene dos cartas iguales. Devuelve la acción de selección <i>Regalo</i> con una de las cartas posibles.	14	Correcto	OK
Es de tipo 3 con todas las cartas son iguales.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Regalo</i> y tiene las tres cartas iguales. Devuelve la acción de selección <i>Regalo</i> con la única carta posible.	15	Correcto	OK
Es de tipo 4 con las opciones distintas.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Competición</i> y tiene las dos opciones distintas. Devuelve la acción de selección 4 con una de las opciones posible.	16	Correcto	OK
Es de tipo 4 con las opciones iguales.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Competición</i> y tiene las dos opciones iguales. Devuelve la acción de selección 4 con la opción posible.	17	Correcto	OK

Tabla 37 - Pruebas para el método: seleccionar acción selección

8.1.2 Pruebas del controlador de la Red Neuronal

Estas pruebas corresponden a la clase *ControladorRedNeuronal*, separadas por los métodos a probar.

8.1.2.1 Método: seleccionar acción

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
4 acciones libres con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que las 4 acciones están disponibles y tiene 7 cartas distintas en la mano a elegir. Devuelve una acción cualquiera con cartas existentes en la mano.	18	Correcto	OK
4 acciones libres con 4 cartas iguales para elegir.	El método recibe un tablero de entrada en el que las 4 acciones están disponibles y tiene 4 cartas iguales en la mano a elegir. Devuelve una acción con la carta repetida de la mano.	19	Correcto	OK
4 acciones libres con 0 cartas para elegir.	El método recibe un tablero de entrada en el que las 4 acciones están disponibles y tiene 0 cartas en la mano a elegir. Devuelve una excepción al no poder seleccionar ninguna carta.	20	Excepción	OK
Acción libre de tipo 1 con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Secreto</i> libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción <i>Secreto</i> con cartas existentes en la mano.	21	Correcto	OK
Acción libre de tipo 1 con 1 carta para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Secreto</i> libre y tiene solo 1 carta en mano a elegir. Devuelve la acción <i>Secreto</i> con la carta de la mano.	22	Correcto	OK
Acción libre de tipo 2 con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Renuncia</i> libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción <i>Renuncia</i> con cartas existentes en la mano.	23	Correcto	OK

Acción libre de tipo 2 con 2 cartas iguales para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo Renuncia libre y tiene solo 2 cartas, las 2 iguales, en mano a elegir. Devuelve la acción Renuncia con las cartas de la mano.	24	Correcto	OK
Acción libre de tipo 3 con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Regalo</i> libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción <i>Regalo</i> con cartas existentes en la mano.	25	Correcto	OK
Acción libre de tipo 3 con 3 cartas iguales para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Regalo</i> libre y tiene solo 3 cartas, todas iguales, en mano a elegir. Devuelve la acción <i>Regalo</i> con las cartas de la mano.	26	Correcto	OK
Acción libre de tipo 4 con 7 cartas distintas para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Competición</i> libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción <i>Competición</i> con cartas existentes en la mano.	27	Correcto	OK
Acción libre de tipo 4 con 4 cartas iguales para elegir.	El método recibe un tablero de entrada en el que tiene solo la acción de tipo <i>Competición</i> libre y tiene solo 4 cartas, todas iguales, en mano a elegir. Devuelve la acción <i>Competición</i> con las cartas de la mano.	28	Correcto	OK
Sin acciones libres.	El método recibe un tablero de entrada en el que no tiene acciones libres y tiene solo 4 cartas, todas iguales, en mano a elegir. Devuelve una excepción.	29	Excepción	OK

Tabla 38 - Pruebas para el método: seleccionar acción

8.1.2.2 Método: seleccionar acción selección

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
Es de tipo 3 con todas las cartas son distintas.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Regalo</i> y tiene todas las cartas iguales. Devuelve la acción de selección <i>Regalo</i> con una de las cartas posibles.	30	Correcto	OK
Es de tipo 3 con algunas cartas iguales.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Regalo</i> y tiene dos cartas iguales. Devuelve la acción de selección <i>Regalo</i> con una de las cartas posibles.	31	Correcto	OK
Es de tipo 3 con todas las cartas son iguales.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Regalo</i> y tiene las tres cartas iguales. Devuelve la acción de selección <i>Regalo</i> con la única carta posible.	32	Correcto	OK
Es de tipo 4 con las opciones distintas.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Competición</i> y tiene las dos opciones distintas. Devuelve la acción de selección <i>Competición</i> con una de las opciones posible.	33	Correcto	OK
Es de tipo 4 con las opciones iguales.	El método recibe un tablero de entrada en el que tiene la acción de selección de tipo <i>Competición</i> y tiene las dos opciones iguales. Devuelve la acción de selección <i>Competición</i> con la opción posible.	34	Correcto	OK

Tabla 39 - Pruebas para el método: seleccionar acción selección

8.1.3 Pruebas del controlador del Tablero

Estas pruebas corresponden a la clase *ControladorTablero*, separadas por los métodos a probar.

8.1.3.1 Método: crear tablero

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
Todo vacío.	Al crear un tablero nuevo y guarda la matriz completamente vacía.	35	Correcto	OK

Tabla 40 - método: crear tablero

8.1.3.2 Método: iniciar ronda

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
Primera ronda y todo vacío excepto las manos de los jugadores.	Se inicia la ronda siendo la primera ronda. Guarda la matriz del tablero con todo vacío a excepción de las manos de ambos jugadores.	36	Correcto	OK
No primera ronda y todo vacío excepto las manos de los jugadores y el favor.	Se inicia ronda siendo la segunda ronda o posterior. Guarda la matriz del tablero con todo vacío a excepción de las manos de ambos jugadores y el favor de las guerreras.	37	Correcto	OK

Tabla 41 - Pruebas para el método: iniciar ronda

8.1.3.3 Método: vista del tablero

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
Inicio primera ronda, comprobar que se ve la mano del jugador.	Cuando el tablero está en la situación de la primera ronda, devuelve una vista en la que se ven las cartas de la mano del jugador.	36	Correcto	OK
Inicio primera ronda, comprobar que no se ve la mano del adversario.	Cuando el tablero está en la situación de la primera ronda, devuelve una vista en la que se ven las cartas de la mano del jugador.	36	Correcto	OK

Inicio primera ronda, comprobar que todo lo demás está vacío.	Cuando el tablero está en la situación de la primera ronda, devuelve una vista en la que se ve vacío las acciones realizadas, el favor de las guerreras y las armas usadas.	36	Correcto	OK
Mitad de la segunda ronda, comprobar que se ve la mano del jugador.	Cuando el tablero está en la situación de la mitad de la segunda ronda, devuelve una vista en la que se ven las cartas de la mano del jugador.	38	Correcto	OK
Mitad de la segunda ronda, comprobar que no se ve la mano del adversario.	Cuando el tablero está en la situación de la mitad de la segunda ronda, devuelve una vista en la que se ven las cartas de la mano del jugador.	38	Correcto	OK
Mitad de la segunda ronda, comprobar que hay acciones realizadas por ambos jugadores.	Cuando el tablero está en la situación de la mitad de la segunda ronda, devuelve una vista en la que se ven las acciones realizadas por ambos jugadores.	38	Correcto	OK
Mitad de la segunda ronda, comprobar que se ven las armas en las guerreras.	Cuando el tablero está en la situación de la mitad de la segunda ronda, devuelve una vista en la que se ven las armas usadas por cada jugador.	38	Correcto	OK
Mitad de la segunda ronda, comprobar que se ve el favor de las guerreras.	Cuando el tablero está en la situación de la mitad de la segunda ronda, devuelve una vista en la que se ve el favor de las guerreras.	38	Correcto	OK
Fin de la segunda ronda, comprobar que mano del jugador vacía.	Cuando el tablero está en la situación del final de la segunda ronda, devuelve una vista en la que se ve la mano del jugador vacía.	39	Correcto	OK
Fin de la segunda ronda, comprobar que no se ve la mano del adversario.	Cuando el tablero está en la situación del final de la segunda ronda, devuelve una vista en la que no se ve la mano del adversario.	39	Correcto	OK
Fin de la segunda ronda, comprobar que se ven las cartas de las acciones del jugador.	Cuando el tablero está en la situación del final de la segunda ronda, devuelve una vista en la que se ven las acciones realizadas por el jugador.	39	Correcto	OK
Fin de la segunda ronda, comprobar que se ven las acciones del	Cuando el tablero está en la situación del final de la segunda ronda, devuelve una vista en la	39	Correcto	OK

adversario, pero no las cartas.	que se ven las acciones usadas por el adversario, pero no se ven las cartas que han sido usadas en las acciones <i>Secreto</i> y <i>Renuncia</i> .			
Fin de la segunda ronda, comprobar que se ven las armas en las guerreras.	Cuando el tablero está en la situación del final de la segunda ronda, devuelve una vista en la que se ven las armas usadas por cada jugador.	39	Correcto	OK
Fin de la segunda ronda, comprobar que se ve el favor de las guerreras.	Cuando el tablero está en la situación del final de la segunda ronda, devuelve una vista en la que se ve el favor de las guerreras.	39	Correcto	OK
Con acción pendiente de selección de tipo 3.	Cuando el tablero está en la situación de que el jugador tiene que realizar una acción de selección pendiente de tipo <i>Regalo</i> , devuelve una vista en la que se ve dicha acción pendiente.	40	Correcto	OK
Con acción pendiente de selección de tipo 4.	Cuando el tablero está en la situación de que el jugador tiene que realizar una acción de selección pendiente de tipo <i>Competición</i> , devuelve una vista en la que se ve dicha acción pendiente.	41	Correcto	OK

Tabla 42 - Pruebas para el método: vista del tablero

8.1.3.4 Método: robar carta

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
Tiene la mano con un solo hueco.	Cuando se llama a esta acción y el jugador tiene un hueco no devuelve una excepción.	42	Correcto	OK
Tiene la mano llena.	Cuando se llama a esta acción y el jugador tiene la mano llena devuelve una excepción.	43	Excepción	OK
El mazo está vacío.	Cuando se llama a esta acción y el mazo de cartas está vacío devuelve una excepción.	44	Excepción	OK

Tabla 43 - Pruebas para el método: robar carta

8.1.3.5 Método: realizar acción

Prueba	Descripción	Numero de caso	Resultado esperado	Estado de la prueba
Acción de tipo secreto y el tipo de acción está disponible.	Se realiza una acción de tipo <i>Secreto</i> para un tablero en el que está disponible y se comprueba que se ha realizado.	45	Correcto	OK
Acción de tipo secreto y la acción está bien formada.	Se realiza una acción de tipo <i>Secreto</i> bien formada y se comprueba que se ha realizado.	45	Correcto	OK
Acción de tipo secreto y el tipo de acción no está disponible.	Se realiza una acción de tipo <i>Secreto</i> para un tablero en el que no está disponible y devuelve una excepción.	46	Excepción	OK
Acción de tipo secreto y la acción tiene un numero incorrecto de cartas.	Se realiza una acción de tipo <i>Secreto</i> que contiene un numero incorrecto de cartas y devuelve una excepción.	47	Excepción	OK
Acción de tipo renuncia y el tipo de acción está disponible.	Se realiza una acción de tipo <i>Renuncia</i> para un tablero en el que está disponible y se comprueba que se ha realizado.	48	Correcto	OK
Acción de tipo renuncia y la acción está bien formada.	Se realiza una acción de tipo <i>Renuncia</i> bien formada y se comprueba que se ha realizado.	48	Correcto	OK
Acción de tipo renuncia y el tipo de acción no está disponible.	Se realiza una acción de tipo <i>Renuncia</i> para un tablero en el que no está disponible y devuelve una excepción.	49	Excepción	OK
Acción de tipo renuncia y la acción tiene un numero incorrecto de cartas.	Se realiza una acción de tipo <i>Renuncia</i> que contiene un numero incorrecto de cartas y devuelve una excepción.	50	Excepción	OK
Acción de tipo regalo y el tipo de acción está disponible.	Se realiza una acción de tipo <i>Regalo</i> para un tablero en el que está disponible y se comprueba que se ha realizado.	51	Correcto	OK
Acción de tipo regalo y la acción está bien formada.	Se realiza una acción de tipo <i>Regalo</i> bien formada y se comprueba que se ha realizado.	51	Correcto	OK
Acción de tipo regalo y el tipo de acción no está disponible.	Se realiza una acción de tipo <i>Regalo</i> para un tablero en el que no está disponible y devuelve una excepción.	52	Excepción	OK
Acción de tipo regalo y la acción tiene un numero incorrecto de cartas.	Se realiza una acción de tipo <i>Regalo</i> que contiene un numero incorrecto de cartas y devuelve	53	Excepción	OK

	una excepción.			
Acción de tipo competición y el tipo de acción está disponible.	Se realiza una acción de tipo <i>Competición</i> para un tablero en el que está disponible y se comprueba que se ha realizado.	54	Correcto	OK
Acción de tipo competición y la acción está bien formada.	Se realiza una acción de tipo <i>Competición</i> bien formada y se comprueba que se ha realizado.	54	Correcto	OK
Acción de tipo competición y el tipo de acción no está disponible.	Se realiza una acción de tipo <i>Competición</i> para un tablero en el que no está disponible y devuelve una excepción.	55	Excepción	OK
Acción de tipo competición y la acción tiene un numero incorrecto de cartas.	Se realiza una acción de tipo <i>Competición</i> que contiene un numero incorrecto de cartas y devuelve una excepción.	56	Excepción	OK
Acción de tipo decisión regalo y el tipo de acción es el esperado.	Se realiza una acción de selección pendiente de tipo <i>Regalo</i> para un tablero en el que está a la espera de esa acción y se comprueba que se ha realizado.	57	Correcto	OK
Acción de tipo decisión regalo y la acción está bien formada.	Se realiza una acción de selección pendiente de tipo <i>Regalo</i> bien formada y se comprueba que se ha realizado.	57	Correcto	OK
Acción de tipo decisión regalo y el tipo de acción no es el esperado.	Se realiza una acción de selección pendiente de tipo <i>Regalo</i> para un tablero en el que no está a la espera de esa acción y devuelve una excepción.	58	Excepción	OK
Acción de tipo decisión regalo y la acción tiene un numero incorrecto de cartas.	Se realiza una acción de selección pendiente de tipo <i>Regalo</i> que contiene un numero incorrecto de cartas y devuelve una excepción.	59	Excepción	OK
Acción de tipo decisión competición y el tipo de acción es el esperado.	Se realiza una acción de selección pendiente de tipo <i>Competición</i> para un tablero en el que está a la espera de esa acción y se comprueba que se ha realizado.	60	Correcto	OK
Acción de tipo decisión competición y la acción está bien formada.	Se realiza una acción de selección pendiente de tipo <i>Competición</i> bien formada y se comprueba que se ha realizado.	60	Correcto	OK
Acción de tipo decisión competición y el tipo de acción no es el esperado.	Se realiza una acción de selección pendiente de tipo <i>Competición</i> para un tablero en el que no está a la espera de esa acción y	61	Excepción	OK

	devuelve una excepción.			
Acción de tipo decisión competición y la acción tiene un numero incorrecto de cartas.	Se realiza una acción de selección pendiente de tipo <i>Competición</i> que contiene un numero incorrecto de cartas y devuelve una excepción.	62	Excepción	OK

Tabla 44 - Pruebas para el método: realizar acción

8.1.4 Resultados obtenidos y cambios realizados

Durante la ejecución de las pruebas se pudieron detectar algunos errores en el código de la aplicación. Estos errores fueron corregidos de inmediato ya que no eran complejos de resolver.

8.2 Pruebas de Integración y del Sistema

En este apartado se van a describir las pruebas de integración y del sistema relacionadas a los casos de uso descritos en el apartado de *Especificación de Casos de Uso*.

8.2.1 Pruebas para los Casos de uso para el Usuario Jugador

Pruebas relacionadas a los *Casos de uso para el Usuario Jugador*.

8.2.1.1 Pruebas del Caso de Uso: Seleccionar acción

Pruebas relacionadas al

Caso de Uso: Seleccionar acción.

<i>Caso de Uso: Seleccionar acción</i>		
Prueba	Resultado Esperado	Resultado Obtenido
El usuario podrá visualizar que acciones tiene disponibles.	Se pueden ver en todo momento las acciones utilizadas y libres están visibles en todo momento.	Superada.
Prueba	Resultado Esperado	Resultado Obtenido
Seleccionar una de las acciones libres para poder crear la acción completa.	Al pulsar sobre una acción marcada como libre esta acción se mostrará en la zona de acción seleccionada.	Superada.

Tabla 45 - Descripción de las pruebas del Caso de Uso: Seleccionar acción

8.2.1.2 Pruebas del Caso de Uso: Cambiar acción seleccionada

Pruebas relacionadas al Caso de Uso: Cambiar acción seleccionada.

Caso de Uso: Cambiar acción seleccionada		
Prueba	Resultado Esperado	Resultado Obtenido
El usuario podrá seleccionar una acción nueva para cambiar la seleccionada anteriormente.	Al pulsar sobre otra acción marcada como libre esta acción sustituirá a la seleccionada anteriormente, borrando las cartas seleccionadas en dicha acción.	Superada.
Prueba	Resultado Esperado	Resultado Obtenido
El usuario podrá seleccionar la misma acción de nuevo.	Al pulsar sobre la misma acción ya seleccionada, esta acción se mostrará en la zona de acción seleccionada, borrando las cartas seleccionadas en dicha acción.	Superada.

Tabla 46 - Descripción de las pruebas del Caso de Uso: Cambiar acción seleccionada

8.2.1.3 Pruebas del Caso de Uso: Selecciona cartas

Pruebas relacionadas al Caso de Uso: Seleccionar cartas.

Caso de Uso: Seleccionar cartas		
Prueba	Resultado Esperado	Resultado Obtenido
El usuario podrá seleccionar desde su mano las cartas necesarias para completar la acción.	Al pulsar sobre una carta de arma de la mano, dicha carta aparecerá como bloqueada y se mostrará en la sección de acción seleccionada.	Superada.
Prueba	Resultado Esperado	Resultado Obtenido
No se pueden seleccionar más cartas de las requeridas	Una vez seleccionado el número requerido de cartas para una acción, al pulsar sobre una carta nueva no ocurrirá nada.	Superada.

Tabla 47 - Descripción de las pruebas del Caso de Uso: Seleccionar cartas

8.2.1.4 Pruebas del Caso de Uso: Enviar acción

Pruebas relacionadas al Caso de Uso: Enviar acción.

Caso de Uso: Enviar acción		
Prueba	Resultado Esperado	Resultado Obtenido
El usuario podrá enviar la acción seleccionada.	Una vez seleccionada la acción con su número exacto de cartas, aparecerá el botón de <i>Aceptar</i> . Esto hará que se envíe la acción seleccionada y se cargue el resultado junto a la acción del adversario.	Superada.
Prueba	Resultado Esperado	Resultado Obtenido
Esta opción solo	Si la acción no está seleccionada o el número de	Superada.

estará disponible cuando el usuario haya seleccionado una acción completa.	cartas para la acción seleccionada es incorrecto, el botón de <i>Aceptar</i> no aparecerá, y en caso de estar anteriormente se ocultará.	
--	--	--

Tabla 48 - Descripción de las pruebas del Caso de Uso: Enviar acción

8.2.1.5 Pruebas del Caso de Uso: Seleccionar cartas acción pendiente

Pruebas relacionadas al Caso de Uso: Seleccionar cartas acción pendiente.

Caso de Uso: Seleccionar cartas acción pendiente		
Prueba	Resultado Esperado	Resultado Obtenido
El usuario podrá seleccionar las cartas de una acción que es del adversario y requiera su interacción.	Cuando el adversario seleccione una acción de <i>Regalo</i> o <i>Competición</i> que requiera interacción por parte del jugador, aparecerá la acción de selección en la pantalla y el jugador podrá seleccionar la opción que desee. Una vez seleccionada aparecerá el botón <i>Aceptar</i> .	Superada.

Tabla 49 - Descripción de las pruebas del Caso de Uso: Seleccionar cartas acción pendiente

8.2.2 Pruebas para los Casos de uso para el encargado del entrenamiento

Pruebas relacionadas a los Casos de uso para el encargado del entrenamiento.

8.2.2.1 Pruebas del Caso de Uso: Generar datos de prueba

Pruebas relacionadas al Caso de Uso: Generar datos de prueba.

Caso de Uso: Generar datos de prueba		
Prueba	Resultado Esperado	Resultado Obtenido
Generación de nuevo datos de pruebas	Al ejecutar la aplicación en modo de generación de datos, se simularán el número de partidas parametrizado, generando dos ficheros con las jugadas de las partidas ganadas.	Superada.

Tabla 50 - Descripción de las pruebas del Caso de Uso: Generar datos de prueba

8.2.2.2 Pruebas del Caso de Uso: Entrenar modelo

Pruebas relacionadas al Caso de Uso: Entrenar modelo.

Caso de Uso: Entrenar modelo		
Prueba	Resultado Esperado	Resultado Obtenido
Entrenamiento de	El encargado del entrenamiento podrá generar	Superada.

nuevo de la red neuronal con los datos generados.	unos nuevos valores para el modelo entrenándolo con los datos que desee (en principio serán los generados en el caso de uso anterior), de manera que la aplicación quedará configurada para funcionar con esos nuevos valores a partir de ese momento.	
---	--	--

Tabla 51 - Descripción de las pruebas del Caso de Uso: Entrenar modelo

8.2.3 Resultados obtenidos y cambios realizados

Al ser estas pruebas muy similares a las realizadas durante el desarrollo de la aplicación no se vieron errores ni problemas durante su reproducción.

8.3 Pruebas de Usabilidad

En esta sección se van a mostrar los resultados obtenidos tras realizar las entrevistas para las pruebas de usabilidad con diferentes usuarios. Estas pruebas se realizaron después de explicarles las *Reglas del juego original*, y hacer una breve explicación del funcionamiento de la aplicación con relación a estas reglas.

8.3.1 Primer usuario entrevistado

A continuación, se van a mostrar los resultados del primer usuario entrevistado.

8.3.1.1 Datos personales

- Ocupación: Desarrollador de software
- Edad: 37

8.3.1.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días. X
2. Varias veces a la semana.
3. Ocasionalmente.
4. Nunca o casi nunca.

¿Qué tipo de actividades realiza con el ordenador?
<p>1. Es parte de mi trabajo o profesión. X</p> <p>2. Lo uso básicamente para ocio.</p> <p>3. Solo empleo aplicaciones estilo Office.</p> <p>4. Únicamente leo el correo y navego ocasionalmente.</p>
¿Ha usado alguna vez software como el de esta prueba?
<p>1. Sí, he empleado software similar.</p> <p>2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares. X</p> <p>3. No, nunca.</p>
¿Qué busca Vd. Principalmente en un programa?
<p>1. Que sea fácil de usar.</p> <p>2. Que sea intuitivo. X</p> <p>3. Que sea rápido.</p> <p>4. Que tenga todas las funciones necesarias.</p>
¿Suele jugar a juegos de azar en plataformas digitales?
<p>1. Sí, con mucha frecuencia.</p> <p>2. Sí, con poca frecuencia.</p> <p>3. Rara vez. X</p> <p>4. No, nunca.</p>
¿Suele jugar a juegos de azar en formato físico?
<p>1. Sí, con mucha frecuencia. X</p> <p>2. Sí, con poca frecuencia.</p> <p>3. Rara vez.</p> <p>4. No, nunca.</p>

Tabla 52 - Preguntas de carácter general del primer usuario

8.3.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿Le resulta sencillo el uso de la aplicación?	X			
¿Sabe a dónde tiene que pulsar para realizar las acciones que desea?	X			
¿Es capaz de saber en cada momento que significa la información mostrada?	X			
¿Se ha sentido perdido en algún momento durante el empleo de la aplicación?				X
¿Le ha costado ver con claridad los elementos de la aplicación?				X
Funcionalidad	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿La información mostrada del tablero ha sido suficiente?	X			
¿Es capaz de reconocer los elementos del juego original visto en la guía?		X		
¿Es capaz de deshacer una acción antes de enviarla?	X			
¿Es capaz de cambiar de acción antes de enviarla?	X			
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son	X			
Los colores empleados son		X		
La distribución de los diferentes elementos en el tablero es	X			
La correlación entre las guerreras y sus armas es	X			
Diseño de la Interfaz	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿Le resulta fácil de usar?	X			
¿El diseño del tablero es claro y atractivo?		X		
¿Cree que el juego está bien estructurado?	X			
¿Piensa que es adecuado la utilización de ventanas emergentes?	X			

Observaciones

Mostrar mayor contraste entre letras y fondos

Tabla 53 - Preguntas sobre la aplicación del primer usuario

8.3.1.4 Cuestionario para el responsable de las pruebas

Aspecto Observado	Notas
<i>El usuario comienza a trabajar de forma rápida por las tareas</i>	Empieza a usar la aplicación y se desenvuelve con fluidez en poco tiempo.
<i>Tiempo en realizar cada tarea</i>	Las tareas se realizaron en el tiempo esperable, teniendo en cuenta el tiempo de pensar en la acción que se desea realizar habitual.
<i>Errores leves cometidos</i>	Ninguno.
<i>Errores graves cometidos</i>	Ninguno.
<i>Preguntas</i>	Ninguna.
<i>Mayor dificultad</i>	No se ha observado ninguna dificultad en el uso de la aplicación.

Tabla 54 - Cuestionario para el responsable del primer usuario

8.3.2 Segundo usuario entrevistado

A continuación, se van a mostrar los resultados del segundo usuario entrevistado.

8.3.2.1 Datos personales

- Ocupación: Fontanero
- Edad: 28

8.3.2.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?

1. Todos los días. **X**
2. Varias veces a la semana.
3. Ocasionalmente.
4. Nunca o casi nunca.

¿Qué tipo de actividades realiza con el ordenador?
<p>1. Es parte de mi trabajo o profesión.</p> <p>2. Lo uso básicamente para ocio. X</p> <p>3. Solo empleo aplicaciones estilo Office.</p> <p>4. Únicamente leo el correo y navego ocasionalmente.</p>
¿Ha usado alguna vez software como el de esta prueba?
<p>1. Sí, he empleado software similar.</p> <p>2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares. X</p> <p>3. No, nunca.</p>
¿Qué busca Vd. Principalmente en un programa?
<p>1. Que sea fácil de usar.</p> <p>2. Que sea intuitivo.</p> <p>3. Que sea rápido. X</p> <p>4. Que tenga todas las funciones necesarias.</p>
¿Suele jugar a juegos de azar en plataformas digitales?
<p>1. Sí, con mucha frecuencia.</p> <p>2. Sí, con poca frecuencia. X</p> <p>3. Rara vez.</p> <p>4. No, nunca.</p>
¿Suele jugar a juegos de azar en formato físico?
<p>1. Sí, con mucha frecuencia. X</p> <p>2. Sí, con poca frecuencia.</p> <p>3. Rara vez.</p> <p>4. No, nunca.</p>

Tabla 55 - Preguntas de carácter general del segundo usuario

8.3.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿Le resulta sencillo el uso de la aplicación?	X			
¿Sabe a dónde tiene que pulsar para realizar las acciones que desea?		X		
¿Es capaz de saber en cada momento que significa la información mostrada?	X			
¿Se ha sentido perdido en algún momento durante el empleo de la aplicación?				X
¿Le ha costado ver con claridad los elementos de la aplicación?				X
Funcionalidad	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿La información mostrada del tablero ha sido suficiente?		X		
¿Es capaz de reconocer los elementos del juego original visto en la guía?	X			
¿Es capaz de deshacer una acción antes de enviarla?	X			
¿Es capaz de cambiar de acción antes de enviarla?	X			
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son	X			
Los colores empleados son	X			
La distribución de los diferentes elementos en el tablero es	X			
La correlación entre las guerreras y sus armas es	X			
Diseño de la Interfaz	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿Le resulta fácil de usar?	X			
¿El diseño del tablero es claro y atractivo?		X		
¿Cree que el juego está bien estructurado?		X		
¿Piensa que es adecuado la utilización de ventanas emergentes?	X			

Observaciones

Cambiar las imágenes de las cartas por unas más trabajadas.

Tabla 56 - Preguntas sobre la aplicación del segundo usuario

8.3.2.4 Cuestionario para el responsable de las pruebas

Aspecto Observado	Notas
<i>El usuario comienza a trabajar de forma rápida por las tareas</i>	Empieza a usar la aplicación ha tardado un poco en reconocer los elementos, pero una vez aprendidos se ha desenvuelto fácilmente
<i>Tiempo en realizar cada tarea</i>	Las tareas se realizaron en el tiempo esperable, teniendo en cuenta el tiempo de pensar en la acción que se desea realizar habitual.
<i>Errores leves cometidos</i>	Ninguno.
<i>Errores graves cometidos</i>	Ninguno.
<i>Preguntas</i>	Ninguna.
<i>Mayor dificultad</i>	No se ha observado ninguna dificultad en el uso de la aplicación.

Tabla 57 - Cuestionario para el responsable del segundo usuario

8.3.3 Tercer usuario entrevistado

A continuación, se van a mostrar los resultados del tercer usuario entrevistado.

8.3.3.1 Datos personales

- Ocupación: Profesor
- Edad: 58

8.3.3.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?

1. Todos los días.
2. Varias veces a la semana.
3. Ocasionalmente. **X**
4. Nunca o casi nunca.

¿Qué tipo de actividades realiza con el ordenador?
<p>1. Es parte de mi trabajo o profesión.</p> <p>2. Lo uso básicamente para ocio.</p> <p>3. Solo empleo aplicaciones estilo Office. X</p> <p>4. Únicamente leo el correo y navego ocasionalmente.</p>
¿Ha usado alguna vez software como el de esta prueba?
<p>1. Sí, he empleado software similar.</p> <p>2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares.</p> <p>3. No, nunca. X</p>
¿Qué busca Vd. Principalmente en un programa?
<p>1. Que sea fácil de usar. X</p> <p>2. Que sea intuitivo.</p> <p>3. Que sea rápido.</p> <p>4. Que tenga todas las funciones necesarias.</p>
¿Suele jugar a juegos de azar en plataformas digitales?
<p>1. Sí, con mucha frecuencia.</p> <p>2. Sí, con poca frecuencia.</p> <p>3. Rara vez.</p> <p>4. No, nunca. X</p>
¿Suele jugar a juegos de azar en formato físico?
<p>1. Sí, con mucha frecuencia.</p> <p>2. Sí, con poca frecuencia.</p> <p>3. Rara vez.</p> <p>4. No, nunca. X</p>

Tabla 58 - Preguntas de carácter general del segundo usuario

8.3.3.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿Le resulta sencillo el uso de la aplicación?		X		
¿Sabe a dónde tiene que pulsar para realizar las acciones que desea?	X			
¿Es capaz de saber en cada momento que significa la información mostrada?		X		
¿Se ha sentido perdido en algún momento durante el empleo de la aplicación?			X	
¿Le ha costado ver con claridad los elementos de la aplicación?				X
Funcionalidad	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿La información mostrada del tablero ha sido suficiente?		X		
¿Es capaz de reconocer los elementos del juego original visto en la guía?			X	
¿Es capaz de deshacer una acción antes de enviarla?		X		
¿Es capaz de cambiar de acción antes de enviarla?		X		
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son	X			
Los colores empleados son	X			
La distribución de los diferentes elementos en el tablero es		X		
La correlación entre las guerreras y sus armas es	X			
Diseño de la Interfaz	Siempre	Con frecuencia	Ocasionalmente	Nunca
¿Le resulta fácil de usar?		X		
¿El diseño del tablero es claro y atractivo?			X	
¿Cree que el juego está bien estructurado?		X		
¿Piensa que es adecuado la utilización de ventanas emergentes?	X			

Observaciones

Siendo que no es fan de los juegos, este juego no le hará cambiar de opinión.

Tabla 59 - Preguntas sobre la aplicación del segundo usuario

8.3.3.4 Cuestionario para el responsable de las pruebas

Aspecto Observado	Notas
<i>El usuario comienza a trabajar de forma rápida por las tareas</i>	Empieza a usar la aplicación ha tardado un poco en reconocer los elementos, y le ha costado adaptarse.
<i>Tiempo en realizar cada tarea</i>	Las tareas se realizaron en un tiempo ligeramente superior al esperado, teniendo en cuenta el tiempo de pensar en la acción que se desea realizar habitual.
<i>Errores leves cometidos</i>	Ninguno.
<i>Errores graves cometidos</i>	Cerró la ventana sin querer.
<i>Preguntas</i>	Ninguna.
<i>Mayor dificultad</i>	No saber que jugadas hacer al no estar acostumbrado a juegos del estilo.

Tabla 60 - Cuestionario para el responsable del segundo usuario

8.3.4 Resultados obtenidos y cambios realizados

Tras realizar los cuestionarios se decidió cambia los colores de la interfaz para que fueran más intuitivos y el contraste entre las letras y los fondos permitiesen una mejor lectura.

8.4 Pruebas de Rendimiento

A continuación, se van a detallar las pruebas de rendimiento que se han realizado.

Para las pruebas de rendimiento se ha utilizado un ordenador con las siguientes características:

- Procesador: Intel Core i7-4790K 4.0Ghz
- RAM: G. Skill Ripjaws X DDR3 1600 PC3-12800 16GB (2x4GB CL7 + 1x8GB CL9)
- Disco duro 1: SanDisk Ultra II SSD 120GB SATA3

8.4.1 Rendimiento en generación de datos

Dentro del rendimiento de la generación de datos se puede observar la respuesta de la parte de la aplicación compuesta por los controladores de partida, tablero, generación de datos y Bot. Para probar el rendimiento de la generación de datos se realizó una generación de 100.000 partidas simuladas, de las cuales se obtuvieron los siguientes resultados:

```
INFO:root:Partidas simuladas: 100000
[INFO ] Duracion del programa: 1643 segundos.
[INFO ] Fin del programa.
```

De aquí podemos observar que las 100.000 partidas simuladas llevaron un tiempo total de 27 minutos y 23 segundos. Como las simulaciones se hacen de forma secuencial, esto supone un promedio de 0.016 segundos por partida simulada.

Además, se ha generado un total de 1.159.560 jugadas ganadoras. Haciendo un promedio de que por cada partida el 50% de las jugadas son ganadoras, nos quedaría un aproximado de 2.320.000 jugadas ejecutadas en 27 minutos y 23 segundos. Esto hace un tiempo promedio de 0.0007 segundos por acción.

8.4.2 Rendimiento en entrenamiento de la red neuronal

Siguiendo con los datos del punto anterior. Para entrenar a la red neuronal se utilizaron las 1.159.560 jugadas ganadoras con sus respectivos 1.159.560 tableros de entrada.

Para estos datos el entrenamiento se obtuvieron los siguientes resultados:

```
[INFO ] Inicio del programa.
[INFO ] Seleccionado modo de entrenamiento de la red neuronal
Train on 1159560 samples
Epoch 1/20
1159560/1159560 [=====] - 277s
Epoch 2/20
1159560/1159560 [=====] - 239s
Epoch 3/20
1159560/1159560 [=====] - 181s
Epoch 4/20
1159560/1159560 [=====] - 196s
Epoch 5/20
1159560/1159560 [=====] - 189s
Epoch 6/20
1159560/1159560 [=====] - 197s
Epoch 7/20
1159560/1159560 [=====] - 197s
Epoch 8/20
1159560/1159560 [=====] - 208s
Epoch 9/20
1159560/1159560 [=====] - 208s
Epoch 10/20
1159560/1159560 [=====] - 211s
Epoch 11/20
1159560/1159560 [=====] - 201s
Epoch 12/20
1159560/1159560 [=====] - 212s
Epoch 13/20
1159560/1159560 [=====] - 207s
Epoch 14/20
1159560/1159560 [=====] - 248s
Epoch 15/20
1159560/1159560 [=====] - 264s
Epoch 16/20
1159560/1159560 [=====] - 264s
Epoch 17/20
1159560/1159560 [=====] - 281s
Epoch 18/20
1159560/1159560 [=====] - 224s
Epoch 19/20
1159560/1159560 [=====] - 204s
```



```
Epoch 20/20
1159560/1159560 [=====] - 200s
[INFO ] Duracion del programa: 4414 segundos.
[INFO ] Fin del programa.
```

Lo que supone un tiempo total de entrenamiento de 1 hora, 13 minutos y 34 segundos.

8.4.3 Rendimiento en la ejecución de las pruebas

El rendimiento de la ejecución de las pruebas se va a utilizar más adelante para el cálculo de la respuesta de la red neuronal a una petición de predicción.

En el caso del rendimiento general, el resultado de la ejecución de todas las pruebas unitarias definidas en el apartado *8.1 Pruebas Unitarias* es el siguiente:

```
Ran 62 tests in 2.575s
```

Teniendo en cuenta que cada prueba es muy diferente, es complicado obtener valores objetivos para estas pruebas.

8.4.4 Rendimiento en la respuesta de la red neuronal

Tal y como se ha hablado en el apartado anterior, para el cálculo del rendimiento de la respuesta de la red neuronal se va a utilizar las pruebas unitarias. En concreto se va a utilizar la prueba 19 definida en el apartado Método: seleccionar acción.

Tras ejecutar esta prueba un total de 10 veces se ha obtenido estos resultados:

```
• Ran 1 test in 0.239s
• Ran 1 test in 0.196s
• Ran 1 test in 0.190s
• Ran 1 test in 0.204s
• Ran 1 test in 0.287s
• Ran 1 test in 0.193s
• Ran 1 test in 0.259s
• Ran 1 test in 0.264s
• Ran 1 test in 0.189s
• Ran 1 test in 0.187s
```

Con estos datos podemos calcular un promedio de 0.22 segundos de rendimiento en la respuesta. Este resultado supone un retraso en la carga de datos de la pantalla del jugador de sobras asumible.

8.5 Pruebas de Eficacia de la Red Neuronal

Las pruebas para poder observar la eficiencia de la red neuronal han consistido en dos formas distintas.

La primera forma de probarla ha sido enfrentando a la red neuronal contra el Bot que usa acciones aleatorias. De las 1000 partidas simuladas la red neuronal ganó 5259 y el Bot ganó 4741. Esto supone un porcentaje de victorias de cerca del 53%. Como se menciona en el punto *7.4.1.3 Problemas en los resultados obtenidos*. No es trivial saber hasta qué punto esto supone una gran mejora, ya que, al existir azar dentro de las propias reglas del juego, este influye en gran medida en las posibilidades de victoria bases.

La segunda forma de probarla ha sido enfrentarla a usuarios reales. Por desgracias estas pruebas han tenido un número muy inferior de partidas realizadas, por lo que no se ha podido medir empíricamente un porcentaje de victorias representativo. Sin embargo, al entrevistar a los usuarios comentaron que no ganaban siempre y que debían tomarse las partidas en serio si querían obtener la victoria.

Capítulo 9. Manuales del Sistema

En este capítulo se va a realizar la descripción de los manuales de despliegue, ejecución, para el usuario y el programador.

9.1 Despliegue del entorno

Descargar e instalar [Anaconda](#)

Una vez instalado Anaconda, desde la aplicación de prompt de anaconda *Anaconda Prompt* crear un entorno nuevo con el comando:

```
conda create --name <nombre del entorno>
```

Para activar el entorno creado ejecutar el comando:

```
conda activate <nombre del entorno>
```

Abir el navegador de Anaconda *Anaconda Navigator*, en *Applications on* seleccionar el entorno creado. A continuación, abrir la sección *Environments* y buscar e importar el archivo *anaconda.yaml* para que se instalen las librerías necesarias.

9.2 Ejecución y configuración

A continuación, se van a mostrar las diferentes formas de ejecución de la aplicación y su configuración.

9.2.1 Ejecución desde el entorno de desarrollo

Para la ejecución de la aplicación abrir el programa Spyder instalado dentro de Anaconda

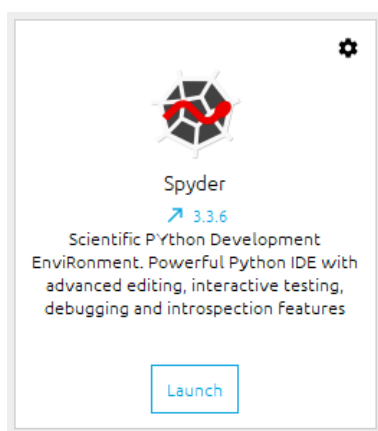


Ilustración 9.1 - Apariencia de la aplicación Spyder en Anaconda

Desde Spyder abrir el archivo `main.py` que está en `\src\main\python` y pulsar F5 para ejecutar.

9.2.2 Ejecución desde el archivo ejecutable

Como el fichero de ejecución es demasiado pesado para poder ser subido a las plataformas, se ha decidido subir al OneDrive personal: https://unioviedo-my.sharepoint.com/:u:/g/personal/uo266321_uniovi_es/EUaR1pmkJhVNgnO7Lyq8WcQBjwBxRc6gLBvEtZuh356BoQ?e=Im9cCB

Descomprimir la carpeta `ElFavorDeLasGuerreras.zip` en un directorio de su preferencia.

El contenido de la carpeta debería quedar de la siguiente manera:

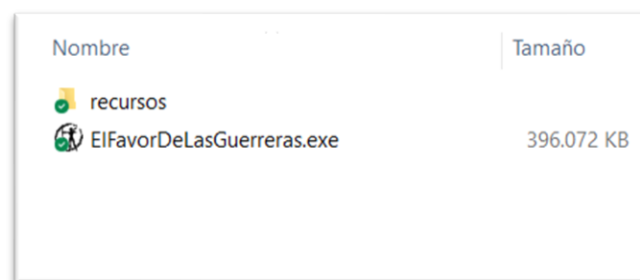


Ilustración 9.2 - Contenido de la carpeta del ejecutable

Es importante que la carpeta de recursos quede en la misma altura que el archivo ejecutable.

Una vez completado todo, únicamente hay que ejecutar el archivo `ElFavorDeLasGuerreras.exe` (es posible que tome algo de tiempo en aparecer la pantalla mientras carga el modelo de la red neuronal).

9.2.3 Configuración del archivo de parámetros

Para cambiar la configuración, abrir el archivo `parametros.properties` que está en `\src\main\recursos` para el caso de ejecutarlo desde el entorno de desarrollo y en `\recursos` si lo ejecutamos desde el archivo ejecutable. Hay 3 campos configurables:

- MODO: Para elegir entre
 - **Generar datos** (1)
 - **Entrenar a la red neuronal** (2)
 - **Jugar** (3)
- NUM_SIMULACIONES: En caso de haber elegido el modo de generar datos, este número indica el número de partidas que se van a simular.
- DIFICULTAD: Para elegir entre
 - **Bot de acciones aleatorias** (1)
 - **Red neuronal entrenada** (2)

9.3 Manual de Usuario

Una vez leídas las instrucciones del juego original de *Hanamikoji* descritas en el punto 3.1.1 *Descripción y objetivo del juego original* y teniendo en cuenta el *Cambio de nomenclatura*, se va a explicar la aplicación.

9.3.1 Elementos del tablero



Ilustración 9.3 - Imagen de la ventana de la aplicación

El tablero está compuesto por diferentes secciones, en las que aparecen diferentes cartas y contadores. Estas secciones corresponden a diferentes partes del juego original descrito en el punto 3.1.1 *Descripción y objetivo del juego original*. A continuación, se va a describir la analogía y comportamiento de cada una de estas secciones de forma detallada.

9.3.1.1 Favor de las guerreras

Esta sección (marcada en rojo en la *Ilustración 9.4 - Sección del favor de las guerreras*), corresponde a las 7 “Cartas de Geisha” con sus respectivos marcadores que se pueden ver en la *Ilustración 3.1 - Ejemplo de preparación del tablero*.



Ilustración 9.4 - Sección del favor de las guerreras

Aquí se muestran las cartas de las guerreras, en las que se muestra en la esquina superior izquierda el número de armas totales relacionado a esta guerrera existente en el mazo de las guerreras. Esta relación se puede ver por el color del fondo de la carta y el arma que usa la guerrera de esta como se muestra en la *Ilustración 9.5 - Relación entre guerreras y armas*.

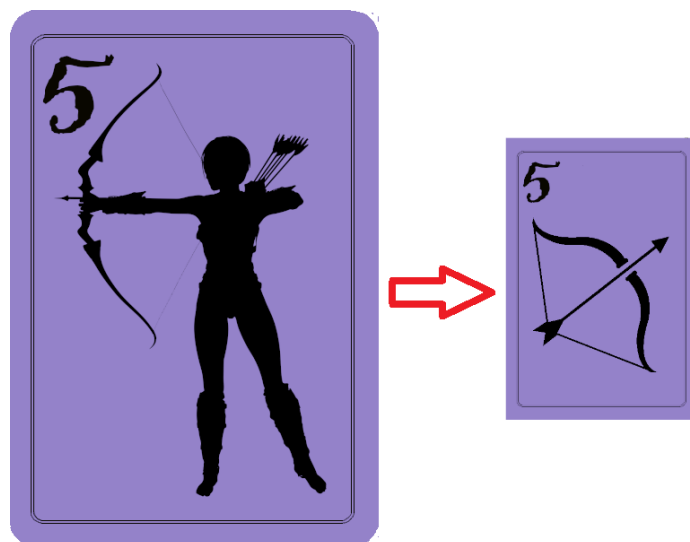


Ilustración 9.5 - Relación entre guerreras y armas

9.3.1.2 Armas usadas

Durante la partida, las cartas usadas por las acciones de tipo *Regalo* y *Competición* se mostrarán como contadores encima y debajo de las cartas de las guerreras para marcar las del adversario y las del jugador respectivamente. Se pueden ver remarcadas en rojo en la *Ilustración 9.6 - Sección de los marcadores de las armas usadas*.

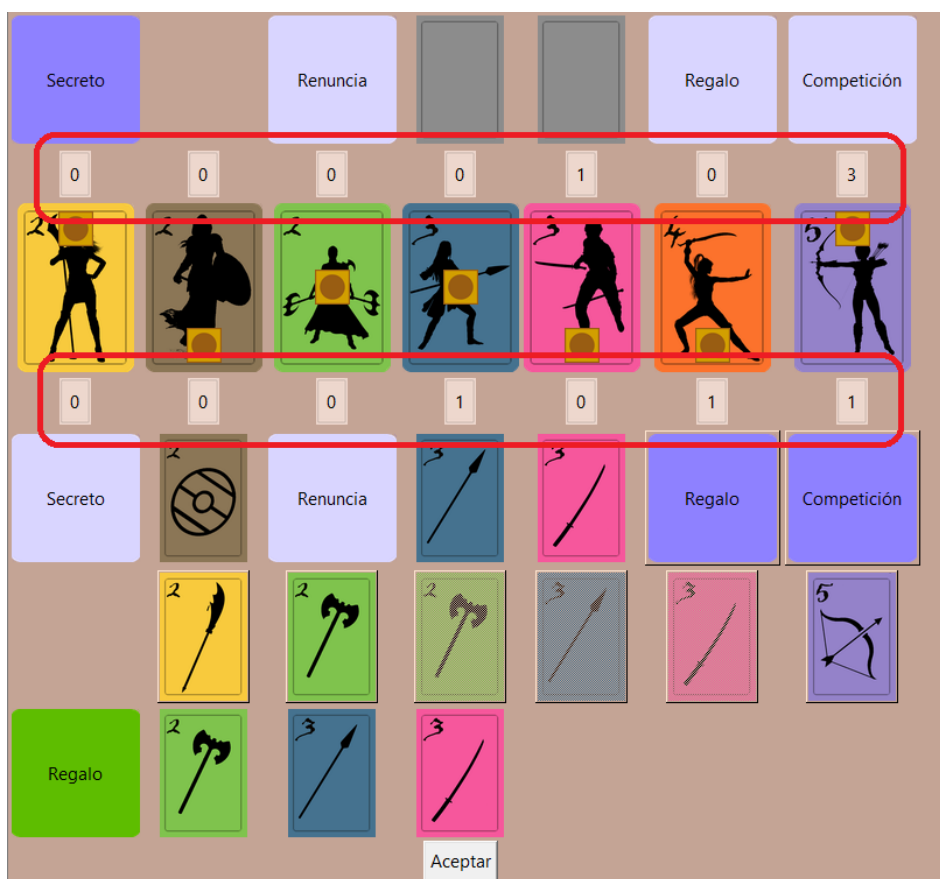


Ilustración 9.6 - Sección de los marcadores de las armas usadas

9.3.1.3 Acciones del adversario

Las acciones realizadas por el adversario se mostrarán en la parte superior del tablero, tal y como se muestra en la *Ilustración 9.7 - Sección de las acciones del adversario*. Se marcarán con colores distintos según estén usadas (*Ilustración 9.9 - Ejemplo de acción usada*) o no (*Ilustración 9.8 - Ejemplo de acción sin usar*). Además, para las acciones de *Secreto* y *Renuncia* se mostrarán las cartas como ocultas a la derecha de dicha acción.

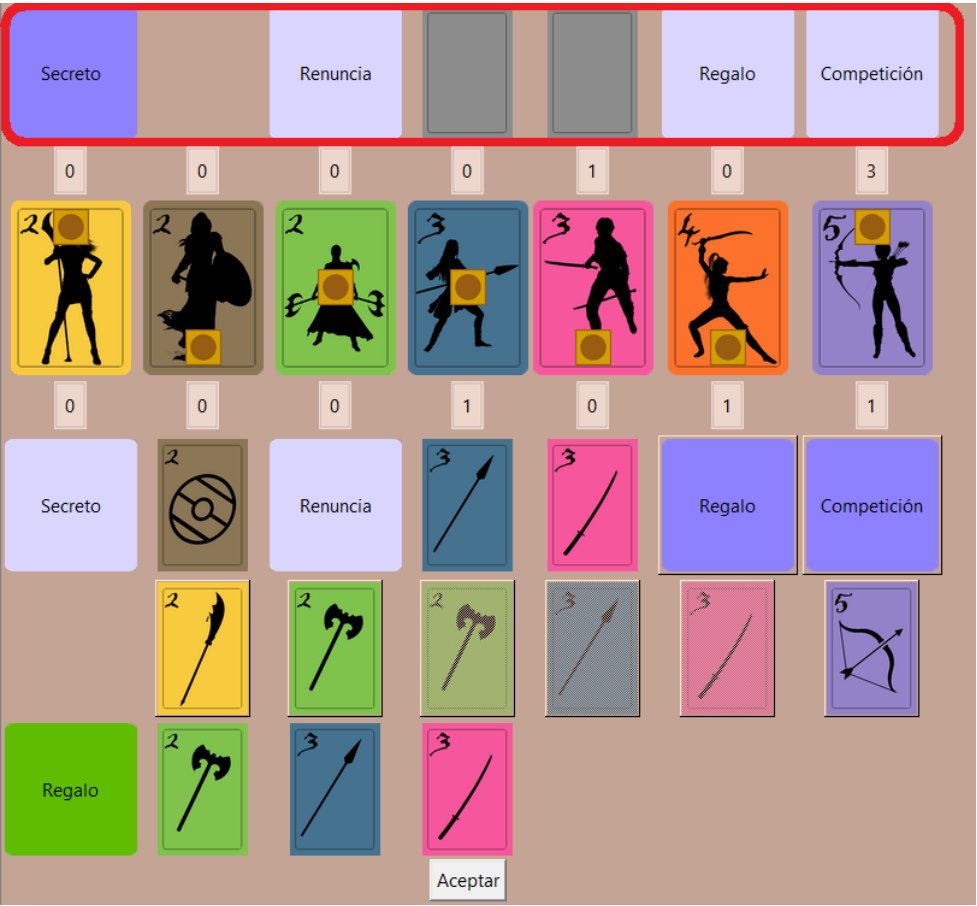


Ilustración 9.7 - Sección de las acciones del adversario



Ilustración 9.8 - Ejemplo de acción sin usar

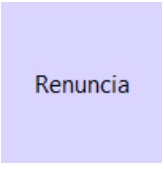


Ilustración 9.9 - Ejemplo de acción usada

9.3.1.4 Acciones del jugador

Al igual que las acciones del adversario, las acciones del jugador se marcarán con colores distintos según estén usadas (*Ilustración 9.9 - Ejemplo de acción usada*) o no (*Ilustración 9.8 - Ejemplo de acción sin usar*), con la diferencia de que las acciones disponibles serán botones que se pueden pulsar para poder seleccionar una acción tal y como se describe en el punto 9.3.2.1 *Pasos para seleccionar una acción*. Estas acciones estarán debajo de los marcadores de armas usados por el jugador, tal y como se muestra en la *Ilustración 9.10 - Sección de las acciones del jugador*. Además, para las acciones de *Secreto* y

Renuncia se mostrarán las cartas con las que se realizó esa acción a la izquierda de esta.

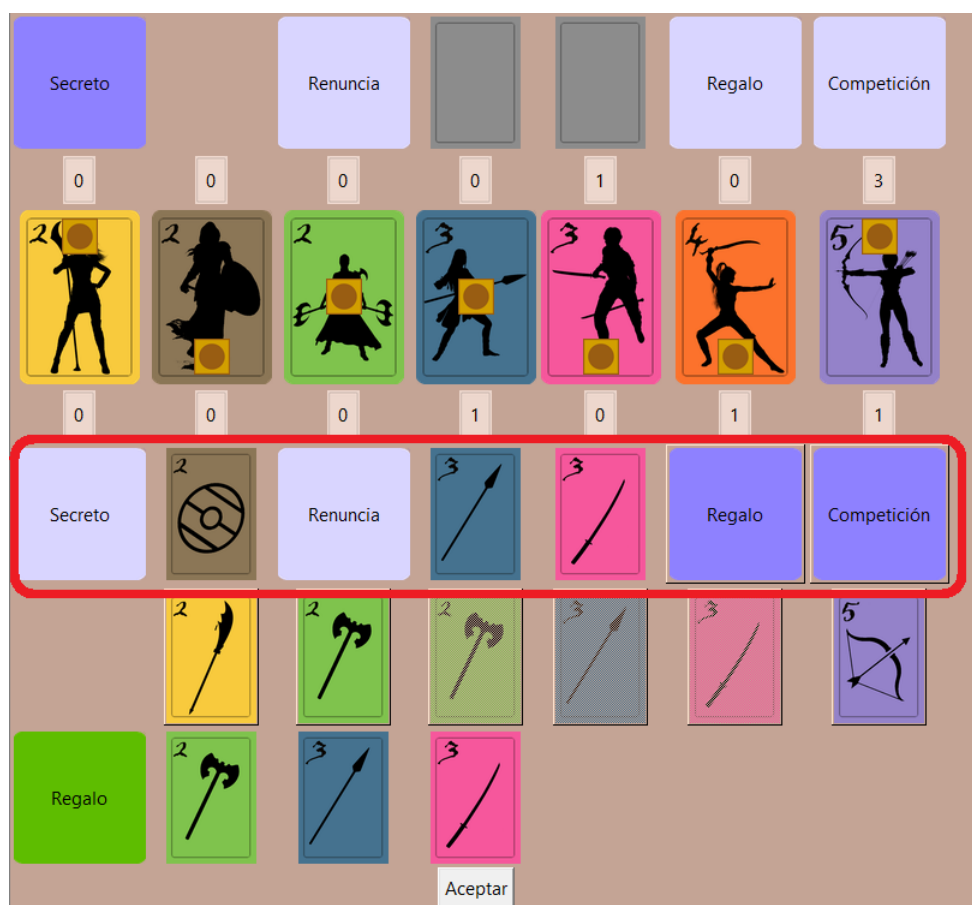


Ilustración 9.10 - Sección de las acciones del jugador

9.3.1.5 Mano del jugador

La mano del jugador constará de 0 a 7 cartas que se mostrarán debajo de la sección anterior (*Acciones del jugador*), tal y como se muestra remarcado en la *Ilustración 9.11 - Sección de las cartas de la mano del jugador*. Estas cartas se podrán pulsar una vez se ha seleccionado una acción para poder completar la acción tal y como se describe en el punto 9.3.2.1 *Pasos para seleccionar una acción*. Las cartas ya usadas quedarán deshabilitadas mientras se termina de completar la acción.

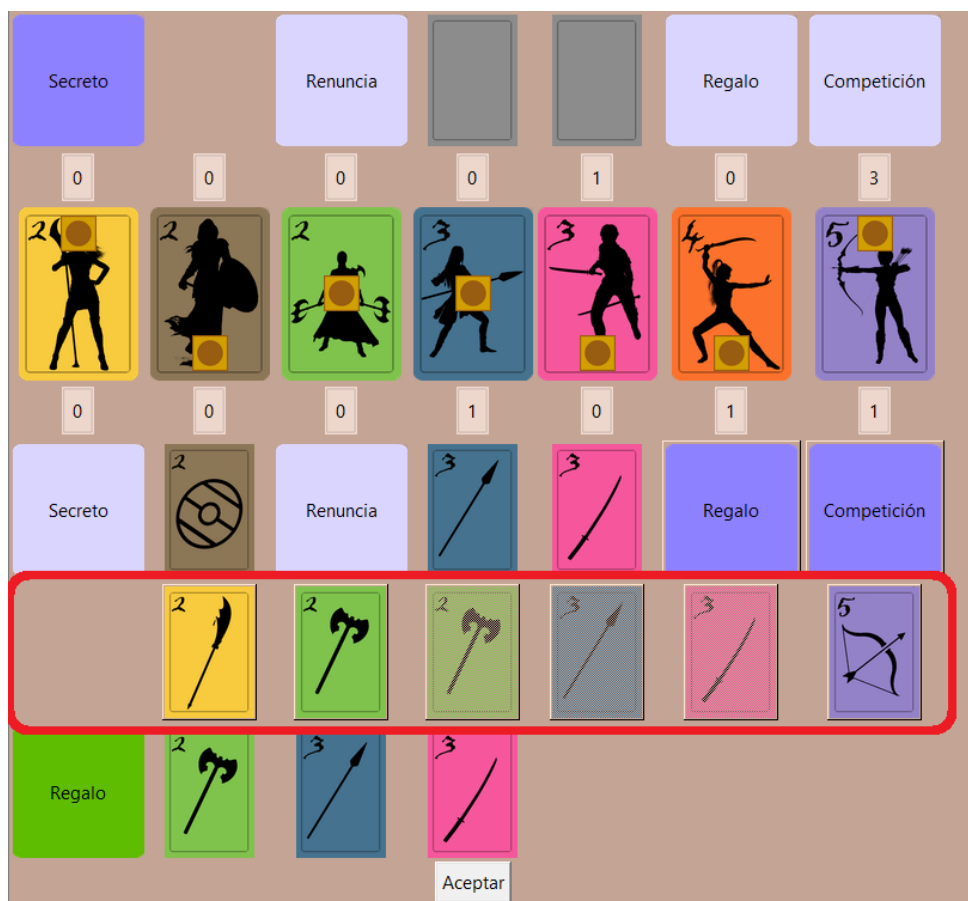


Ilustración 9.11 - Sección de las cartas de la mano del jugador

9.3.1.6 Acción actual

Una vez seleccionada una acción, esta aparecerá en la sección de la acción actual. Esta sección se encuentra debajo de las cartas de la mano del jugador, tal y como se muestra en la zona remarcada en la *Ilustración 9.12 - Sección de la acción actual*. En esta sección aparecerán también las cartas de la mano que el jugador vaya seleccionado. Una vez se ha seleccionado el número de cartas necesarias aparecerá el botón de aceptar debajo. Cuando se pulse este botón se enviará la jugada seleccionada y la partida continuará.

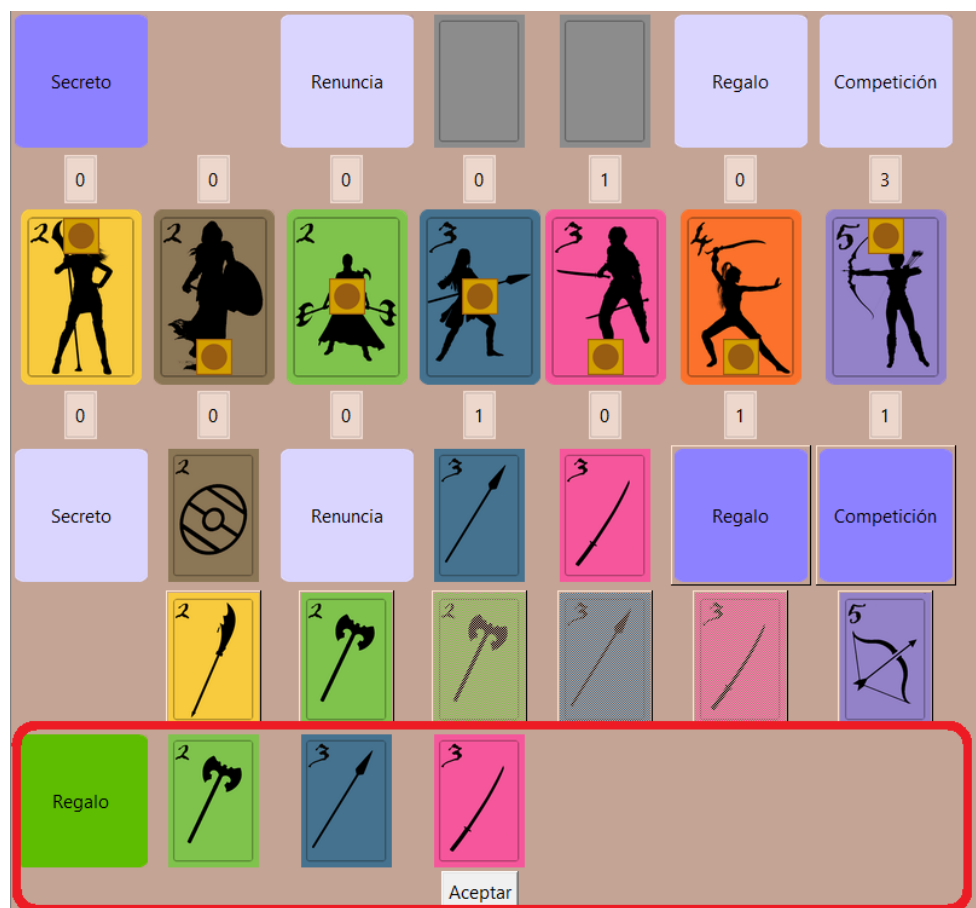


Ilustración 9.12 - Sección de la acción actual

9.3.2 Comportamiento del tablero

Una vez vistos los Elementos del tablero y comprendidas las Reglas del juego original se va a describir el comportamiento del tablero en general.

Como la parte del juego de repartir las cartas y colocar las acciones en su lugar lo hace la aplicación, el jugador únicamente tiene el objetivo de decidir qué acción realizar. Para poder tomar esta decisión el jugador tiene a su disposición una vista parcial de la situación del tablero (no puede ver las cartas de la mano del adversario ni las cartas usadas en sus acciones de *Secreto* y

Renuncia

Renuncia).

Una vez el jugador ha visto el tablero y se ve disposición para realizar una acción tendrá que seguir los siguientes pasos.

9.3.2.1 Pasos para seleccionar una acción

En primer lugar, el jugador seleccionará una de las acciones disponibles en la sección de *Acciones del jugador*, esta acción pasará a estar marcada en la sección de *Acción actual*. El jugador puede cambiar de opinión y seleccionar una acción diferente siempre que quiera antes de enviarla, lo que sustituirá la acción seleccionada anteriormente y borrará las cartas de armas que haya seleccionado y las hubiera.

En segundo lugar, el jugador seleccionará las cartas de su mano para poder completar la acción seleccionada. Tal y como se mencionó en la sección de la *Mano del jugador*, las cartas seleccionadas quedarán bloqueadas, ya que no se puede usar la misma carta dos veces. Si el jugador quisiera cambiar las cartas que ha seleccionado antes de enviarlas bastaría con volver a pulsar la misma acción en la sección de *Acciones del jugador* y así tendría disponible la acción vacía y todas las cartas de la mano habilitadas.

Por último, cuando se haya seleccionado la acción con su número de cartas correspondientes, se activará el botón de aceptar. Al pulsar este botón se enviará la acción y la partida continuará, de manera que al jugador se le mostrará el tablero con los nuevos cambios correspondiente a la acción que ha realizado y la acción que ha realizado el adversario.

9.3.2.2 Pasos para seleccionar una acción de selección

Esta situación corresponde a cuando el adversario ha realizado una acción de tipo *Regalo o Competición* ya que estas acciones requieren una decisión por parte del otro jugador. En este caso se mostrará dicha acción en la sección de *Acciones del jugador*, con las secciones de la *Mano del jugado* y de las *Acciones del jugador* deshabilitada, ya que el jugador solo podrá ver la situación del tablero y seleccionar las cartas que quiera.

En el caso de la acción de selección para una acción de tipo *Regalo* al jugador se le planteará una decisión entre 3 cartas, una vez seleccione una de ellas aparecerá deshabilitada y aparecerá el botón de enviar. El jugador podrá cambiar de carta las veces que quiera antes de dar al botón de enviar. Una vez pulsado el botón de enviar la selección se enviará y la partida continuará.

En el caso de la acción de selección para una acción de tipo *Competición* al jugador se le planteará una decisión entre 4 cartas, una vez seleccione una de ellas aparecerán deshabilitadas 2 cartas, correspondientes al grupo de la carta seleccionada y aparecerá el botón de enviar. El jugador podrá cambiar de opción las veces que quiera antes de dar al botón de enviar. Una vez pulsado el botón de enviar la selección se enviará y la partida continuará.



Ilustración 9.13 - Ejemplo de acción de selección

9.4 Manual del Programador

Para poder hacer las posibles *Ampliaciones* que comentaremos más adelante, el programador necesitará tener unos conceptos básicos de cómo está construida la aplicación. Por eso en esta sección se van a describir cómo están hechos y como se podrían modificar los diferentes módulos de la aplicación.

9.4.1 Modificar la red neuronal

En caso de querer modificar la red neuronal hay que tener en cuenta el alcance de la modificación. Así pues, tendríamos dos opciones, un *Nuevo diseño de la red neuronal*, para lo cual se necesitaría modificar únicamente el modelo de la red neuronal actual; o un nuevo modo de entrenamiento, como por ejemplo un *Nuevo modo de entrenamiento* con redes neuronales adversarias, para lo cual habría que redefinir gran parte de la aplicación.

En caso de querer únicamente modificar el diseño de la red neuronal actual, modificando las capas existentes, basta con cambiar el método `__establecerCapas` en la clase `Entrenamiento`. Este método se compone de 3 partes claramente comentadas dentro del código, una primera parte en la que se definen los parámetros de las capas, a continuación, se definen los tipos de capas que se van a utilizar y por último el orden dentro de la red neuronal en el que van a estar esas capas. Si se quieren definir nuevas capas, habría que declararlas en la sección de definición de tipos de capas y luego añadirlas al modelo, si lo que se desea es únicamente cambiar el orden de las capas existentes o cambiar su número, sería suficiente con cambiar las capas que se añaden al modelo en la sección de definición de orden de las capas.

En el caso de querer cambiar el modo de entrenamiento de la red neuronal, es decir, cambiar el tipo de red neuronal que se va a usar, habría que tener en cuenta que tipo de entrada y salida necesita esa red neuronal en concreto. Si se va a utilizar una red neuronal con datos que se generen durante el entrenamiento, hay que tener en cuenta que en toda la aplicación se están utilizando como datos entre módulos las situaciones puntuales de los tableros y las acciones utilizadas para esos tableros. En este caso se podría describir un nuevo tipo de jugador teniendo en cuenta siempre la interfaz de los jugadores: unos atributos privados `__miNombre` y `__miNumero` con sus respectivos *getters* y unos métodos públicos `decidirAccion`, `decidirAccionDeSeleccion` y `finish`. En caso de querer usar otro tipo de red neuronal en el que se le pasen datos generados con anterioridad, habría que definir el método de generación de datos y su etiquetado, pudiendo simular partidas de una forma similar a la clase `ControladorGeneradorDatos`.

9.4.2 Modificar la interfaz

Para modificar la interfaz de usuario existen dos opciones, crear una nueva interfaz independiente a la ya existente, o modificar la ya existente con intención de añadirle mejoras.

Si se desea crear una nueva interfaz independiente, bastaría con tener en cuenta la interfaz de los jugadores: unos atributos privados `__miNombre` y `__miNumero` con sus respectivos *getters* y unos métodos públicos *decidirAccion*, *decidirAccionDeSeleccion* y *finish*. En caso de que la nueva interfaz gráfica tenga dificultades para adaptarse a esta definición, se puede utilizar un controlador intermedio como se está utilizando en la versión implementada con la clase *ControladorJugador*.

En caso de querer mejorar la interfaz gráfica existente hay que tener en cuenta que está definida en su totalidad en la clase *GUI_Tkinter* del módulo *interfaz*. Esta interfaz está definida a modo de matriz para adaptarse a la matriz de entrada que va a llegar por parte del módulo de *controladores*, (para ver en detalle el diseño ir a la sección 6.5 *Diseño de la Interfaz*). En esta clase *GUI_Tkinter*, hay definidas una gran cantidad de métodos que configuran botones e imágenes que muestran información en la pantalla, si se quiere mostrar información con elementos ya existentes basta con utilizar estos métodos de configuración, en caso de querer cambiar los elementos existentes habría que cambiar la configuración descrita en estos métodos de configuración. Cabe destacar que hay parte de la configuración que está parametrizada en los archivos de parametrización: *ParametrosGUI.py* para los parámetros de los elementos de la interfaz, *ParametrosImagenes.py* para las rutas de las imágenes. Si se quieren modificar imágenes existentes basta con modificar los ficheros *.png* en `\src\main\recursos\estaticos`.

9.4.3 Modificar las reglas del juego

Aunque la idea de modificar las reglas del juego podría no tener mucho sentido, ya que dejaría de ser una versión del *Hanamikoji*, este podría ser el objetivo de dichas modificaciones. También existe la posibilidad de querer cambiar los algoritmos de gestión de las reglas del juego. En cualquiera de los dos casos, para poderlo hacer habrá que tener en cuenta las siguientes consideraciones.

La clase *ControladorPartida* es la encargada de gestionar los turnos y la interacción de los diferentes jugadores con el tablero, por lo que toda esta gestión está separada del controlador del tablero. En la clase *ControladorTablero* se encuentra la lógica de negocio encargada de aplicar las reglas del juego, esta es la clase encargada de transformar las acciones de los jugadores en una siguiente fase del tablero, también calcula el ganador de la partida y gestiona el mazo de cartas. Es importante destacar que la configuración del tablero está parametrizada en *ParametrosTablero.py*, este fichero se utiliza en casi todas las partes del proyecto, por lo que si se desea realizar algún cambio es conveniente revisar las repercusiones que pueda tener.

Capítulo 10. Conclusiones

Ampliaciones

y

En este capítulo se va a realizar una explicación de las conclusiones al realizar el proyecto y posibles aplicaciones.

10.1 Conclusiones

Gracias a la realización de este proyecto he conseguido desarrollar una versión digital de un juego de cartas existente nunca digitalizado de esta manera, de forma que el jugador puede jugar partidas de *Hanamikoji* contra una inteligencia artificial entrenada con el uso de redes neuronales convolucionales.

La realización de este proyecto ha supuesto un reto, tanto a nivel técnico como a nivel personal. La inteligencia artificial y las redes neuronales es una parte de la informática por la que se pasa muy por encima en la carrera, y es un campo de estudio relativamente joven y que está en constante evolución.

Como ya se ha mencionado en el apartado de *Problemas Encontrados*, el diseño de una red neuronal es algo que requiere una buena base de experiencia práctica para poder realizarse.

En un principio, pensaba que una red neuronal era una caja negra a la que se le pasaban datos de entrenamiento y se entrenaba ella sola, pero durante el desarrollo de este proyecto me he dado cuenta de que en realidad el interior de esa caja negra debe ser diseñado y desarrollado con cuidado para cada tarea específica en la que se quiere entrenar a la red neuronal. Si es cierto que hay cierta parte que se entrena automáticamente, y es en la obtención de los valores de cada una de las neuronas del modelo, pero este modelo debe ser descrito con anterioridad, y no es una tarea trivial.

Existen grandes grupos de aplicación de redes neuronales como puedan ser el análisis o generación del lenguaje natural (conocido como NLP [32]), el análisis o generación de imágenes (como DALL-E [33]) o el aprendizaje de videojuegos (como AlphaStar [34]). Para estos grupos existen ciertas descripciones de diseños de muy alto nivel, pero una vez se necesita un problema concreto el diseño concreto pasa a ser algo muy especializado y se requiere cierta “intuición” y una gran cantidad de prueba y error para poder llegar a alcanzar el resultado deseado.

Ciertamente, el desarrollo de este proyecto se ha sentido como para un músico con experiencia en un instrumento, cambiar a un instrumento de una familia distinta. La experiencia en el conocimiento de la música es compartida, pero la técnica es completamente nueva. En esta analogía el conocimiento de la música sería el conocimiento en programación, y el instrumento nuevo sería aprender sobre en la rama de inteligencia artificial y redes neuronales.

10.2 Ampliaciones

A continuación, se van a describir posibles ampliaciones y mejoras que podrían desarrollarse en el proyecto. No están descritas en orden de importancia y puede que algunas sean contradictorias entre ellas, ya que suponen cambios sobre la misma área.

10.2.1 Nuevo entrenamiento con nuevos datos

Actualmente el entrenamiento de la red neuronal se está haciendo con el uso de la simulación de jugadas completamente aleatorias. Una posible mejora sería guardar una enorme cantidad de jugadas realizadas por usuarios de la aplicación, para así poder enseñar a la red neuronal jugadas que ya han sido pensadas y estudiadas por anterioridad.

10.2.2 Nuevo modo de entrenamiento con redes neuronales adversarias

Actualmente se está utilizando aprendizaje supervisado con redes neuronales convolucionales. Una posible ampliación sería crear una neuronal del tipo de redes neuronales adversarias, en la que se entrenaría contra sí misma, siendo este un aprendizaje por refuerzo.

Una vez realizada la nueva red neuronal y entrenada, se podría comparar los resultados de dicha red con la red desarrollada en para la aplicación actual y comprobar cuál de las dos opciones sería más eficaz en la tarea de ganar partidas contra usuarios reales.

10.2.3 Nuevo diseño de la red neuronal

Como se ha mencionado con anterioridad, al realizar el diseño de la red neuronal se carecía de una experiencia anterior, por lo que es posible que el diseño actual tenga un rango de mejora notable.

Una posible mejora es un nuevo análisis y diseño en un futuro en el que se tenga más experiencia, o actualmente por una persona más experimentada en esta área. Al contrario que la ampliación anterior, esta no sería un cambio en el tipo de red neuronal, se seguiría usando una red neuronal convolucional, pero con un nuevo diseño.

10.2.4 Ampliación del alcance de la aplicación: Uso de servicios web

Si se quisiera transformar la aplicación en una aplicación online añadiendo el uso de servicios web, de manera que los módulos de controladores y de red neuronal quedarían intactos, pero el módulo de la interfaz gráfica se cambiaría por un módulo de puerta de enlace de un *back-end* de un servicio web o incluso la definición de microservicios. A su vez se debería realizar una parte de *front-end* que consuma dichos servicios y sirviese de interfaz gráfica para que el usuario pudiese interactuar con la aplicación tal y como se comentaba en el punto 2.2 *Evaluación de alternativas tecnológicas* concretamente en la del uso de *HTML* y *JavaScript*.

10.2.5 Ampliación del alcance de la aplicación: Gestión de usuarios

Para el desarrollo de este nuevo alcance, convendría tener una base de datos en la que se guarden los datos de los usuarios, por lo que habría que añadir una manera en la que puedan registrarse y ver datos de partidas anteriores. Esta ampliación podría ser interesante desarrollarla junto con el apartado anterior, ya que con el uso de servicios web es conveniente tener un registro de usuarios.

Capítulo 11. Planificación y Presupuestos finales del Proyecto

En este capítulo se va a mostrar la planificación final realizada y los cambios en los presupuestos debidos a la diferencia con la previsión inicial.

11.1 Planificación final

En esta sección se va a describir la comparativa con la realizada en el punto 4.1 *Planificación*.

11.1.1 Cambio en los grupos de tareas

Para la realización del proyecto se decidió agrupar la contabilización de horas dedicadas en grupos más grandes y flexibles, ya que con frecuencia se estaban realizando tareas que formaban parte a la vez de subgrupos diferencias.

De esta forma, la planificación inicial quedaría transformada de la manera que se muestra en la *Tabla 61 - Resultado de la transformación de la planificación inicial* mostrada a continuación.

	Nombre de la tarea	Duración Prevista
	Proyecto Completo	300
1	Investigación	90
2	Análisis y Diseño	45
3	Implementación	85
4	Pruebas	40
5	Documentación	40

Tabla 61 - Resultado de la transformación de la planificación inicial

11.1.2 Tiempos reales dedicados

A continuación, se va a mostrar una tabla con las horas dedicadas una vez finalizado el proyecto.

	Nombre de la tarea	Duración Final
	Proyecto Completo	456
1	Investigación	92
2	Análisis y Diseño	101
3	Implementación	106
4	Pruebas	66
5	Documentación	91

Tabla 62 - Planificación con tiempos finales

11.1.3 Comparativa y conclusiones

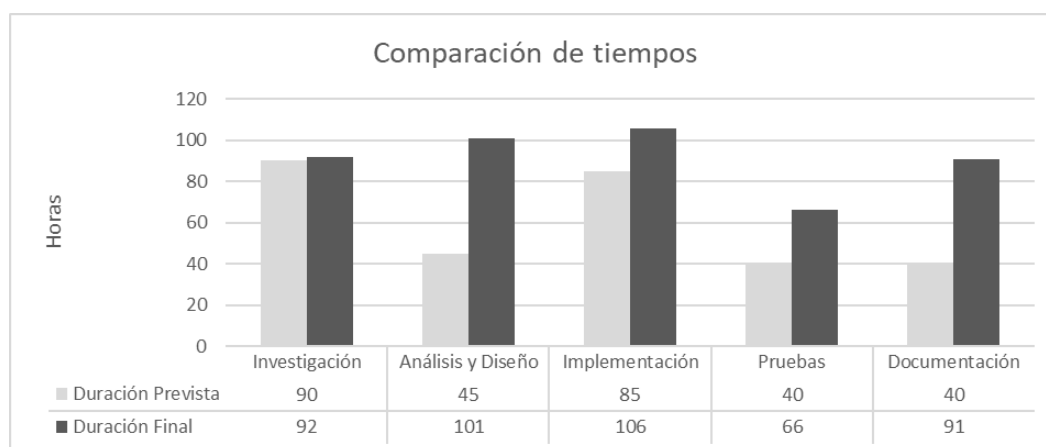


Ilustración 11.1 - Comparación de planificación inicial y final

Como se puede observar en la *Ilustración 11.1 - Comparación de planificación inicial y final*, ha habido un exceso de trabajo en comparación con la previsión inicial en casi todas las áreas.

El cambio más significativo se produce en el análisis y diseño de la aplicación. Este se ha producido por la dificultad mencionada en el punto sobre el *Desconocimiento sobre la realización del diseño de una red neuronal*. Aunque la investigación realizada fue suficiente para tener los conceptos básicos para poder realizar el diseño de una red neuronal desde cero, a la hora de empezar a realizarlo y comprobar en la implementación que no daba los resultados esperados supuso un tiempo extra de prueba y error hasta que se consiguió un diseño que funcionase dentro del proyecto.

11.2 Presupuesto final

En esta sección se va a realizar el presupuesto final teniendo en cuenta la planificación final realizada.

11.2.1 Coste de personal y de hardware

El coste de personal no ha cambiado, ya que es calculado en base al coste por horas, por lo que es independiente a la planificación. Puede verse en el punto: *4.2.1 Coste de personal*.

El coste de hardware tampoco ha cambiado, ya que en un inicio era nulo, al ser un proyecto completamente de software y así ha seguido siendo.

11.2.2 Costes indirectos

El número de los costes indirectos y su coste por mes no ha cambiado desde la planificación inicial, pero al haberse utilizado un total de 16 meses en lugar de los 4 previstos, el coste final si ha cambiado:

Servicio	Tiempo de uso (Meses)	Coste por mes	Coste total
Consumo de electricidad	16	50,00 €	800,00 €
Gastos de Internet	16	50,00 €	800,00 €
Alquiler de software	16	40,00 €	640,00 €
Gastos de material de oficina	16	40,00 €	640,00 €
Gastos de reparación de hardware	16	150,00 €	2.400,00 €

Tabla 63 - Costes indirectos finales

Con un total de 1.920,00 €

11.2.3 Desarrollo del presupuesto final

A continuación, se va a mostrar una tabla con la relación de las horas invertidas y los costes de cada parte del proyecto.

Descripción	Cantidad	Unidades	Participantes	Coste por hora	Total
Proyecto Completo	456	horas			9.943,06 €
Investigación	92	horas	Analista de sistemas	23,63 €	2.173,96 €
Análisis y Diseño	101	horas	Arquitecto software	30,47 €	3.077,47 €
Implementación	106	horas	Desarrollador	13,29 €	1.408,74 €
Pruebas	66	horas	Tester	17,16 €	1.132,56 €
Documentación	91	horas	Analista de sistemas	23,63 €	2.150,33 €

Tabla 64 - Presupuesto final del desarrollo del proyecto

11.2.4 Comparativa con el presupuesto inicial

En comparación con el presupuesto inicial, podemos observar que el número de horas extra invertido y el tiempo total utilizado para el proyecto ha supuesto un coste extra significativo.

	Inicial	Final	Diferencia
Total facturado	6.122,30 €	9.943,06 €	3.820,76 €
Beneficio esperado del 25%	1.530,58 €	2.485,77 €	955,19 €
Costes indirectos	480,00 €	5.280,00 €	4.800,00 €
Total a facturar	8.132,88 €	17.708,83 €	9.575,95 €

Tabla 65 - Comparativa de presupuestos inicial y final

Capítulo 12. Referencias Bibliográficas

- [1] “Reseña: Hanamikoji | Misut Meeple.” <https://misutmeeple.com/2020/01/resena-hanamikoji/> (accessed May 04, 2022).
- [2] “Welcome to Python.org.” <https://www.python.org/> (accessed Jun. 08, 2022).
- [3] “¿Qué es una Red Neuronal? Parte 1: La Neurona | DotCSV - YouTube.” https://www.youtube.com/watch?v=MRlv2lwFTPg&t=2s&ab_channel=DotCSV (accessed Jun. 08, 2022).
- [4] “¡Redes Neuronales CONVOLUCIONALES! ¿Cómo funcionan? - YouTube.” https://www.youtube.com/watch?v=V8j1oENVz00&ab_channel=DotCSV (accessed May 05, 2022).
- [5] “Java | Oracle.” <https://www.java.com/es/> (accessed Jun. 08, 2022).
- [6] “JavaScript | MDN.” <https://developer.mozilla.org/es/docs/Web/JavaScript> (accessed Jun. 08, 2022).
- [7] “The C Programming Language - Brian W. Kernighan, Dennis M. Ritchie - Google Libros.” https://books.google.es/books?hl=es&lr=&id=OpJ_0zpF7jIC&oi=fnd&pg=PP11&dq=c+programming+language&ots=2dL0fareNX&sig=Mqw76NFQ-p6EANDZ26-GHbYYfbw#v=onepage&q=c%20programming%20language&f=false (accessed Jun. 08, 2022).
- [8] “Qué es C++: Características y aplicaciones | OpenWebinars.” <https://openwebinars.net/blog/que-es-cpp/> (accessed Jun. 08, 2022).
- [9] “HTML: Lenguaje de etiquetas de hipertexto | MDN.” <https://developer.mozilla.org/es/docs/Web/HTML> (accessed Jun. 08, 2022).
- [10] “React – Una biblioteca de JavaScript para construir interfaces de usuario.” <https://es.reactjs.org/> (accessed Jun. 08, 2022).
- [11] “Angular.” <https://angular.io/> (accessed Jun. 08, 2022).
- [12] “Pygame Front Page — pygame v2.1.1 documentation.” <https://www.pygame.org/docs/> (accessed Jun. 08, 2022).
- [13] “tkinter — Interface de Python para Tcl/Tk — documentación de Python - 3.10.5.” <https://docs.python.org/es/3/library/tkinter.html> (accessed Jun. 08, 2022).
- [14] “Hanamikoji - Reglas del juego de cartas - Happy Meeple.” <https://www.happymeeple.com/es/juegos/hanamikoji/reglas-del-juego/> (accessed May 04, 2022).

- [15] “Deep Learning – Introducción práctica con Keras - Jordi TORRES.AI.” <https://torres.ai/deep-learning-inteligencia-artificial-keras/#Funciones de activacion> (accessed May 04, 2022).
- [16] “python 3.x - Incremental learning in keras - Stack Overflow.” <https://stackoverflow.com/questions/63907380/incremental-learning-in-keras> (accessed May 21, 2022).
- [17] “Pasos para entrenar una red neuronal profunda - Código Fuente.” <https://www.codigofuente.org/pasos-entrenar-una-red-neuronal-profunda/> (accessed May 04, 2022).
- [18] “Layer activation functions.” <https://keras.io/api/layers/activations/> (accessed May 04, 2022).
- [19] “CNN | Introduction to Pooling Layer - GeeksforGeeks.” <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/> (accessed Jun. 08, 2022).
- [20] “Explora el Aprendizaje Profundo: un Tutorial de TensorFlow Python | Toptal.” <https://www.toptal.com/machine-learning/de-resolver-ecuaciones-a-aprendizaje-profundo-un-tutorial-de-tensorflow-python> (accessed May 04, 2022).
- [21] “Estándar de mensajería CSV - Documentación de IBM.” <https://www.ibm.com/docs/es/integration-bus/10.0?topic=formats-csv-messaging-standard> (accessed May 29, 2022).
- [22] “Cargar datos CSV | TensorFlow Core.” https://www.tensorflow.org/tutorials/load_data/csv (accessed May 04, 2022).
- [23] “Flatten layer.” https://keras.io/api/layers/reshaping_layers/flatten/ (accessed Jun. 08, 2022).
- [24] “Dense layer.” https://keras.io/api/layers/core_layers/dense/ (accessed Jun. 08, 2022).
- [25] “Dropout layer.” https://keras.io/api/layers/regularization_layers/dropout/ (accessed Jun. 08, 2022).
- [26] “Reshape layer.” https://keras.io/api/layers/reshaping_layers/reshape/ (accessed May 04, 2022).
- [27] “Estilo de arquitectura de n niveles - Azure Architecture Center | Microsoft Docs.” <https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/n-tier> (accessed May 29, 2022).
- [28] “Anaconda | The World’s Most Popular Data Science Platform.” <https://www.anaconda.com/> (accessed May 29, 2022).
- [29] “Home — Spyder IDE.” <https://www.spyder-ide.org/> (accessed May 29, 2022).

- [30] “Sourcetree | Free Git GUI for Mac and Windows.” <https://www.sourcetreeapp.com/> (accessed May 29, 2022).
- [31] “GIMP Descargas, tutoriales y foros. Alternativa a Photoshop gratis y libre.” <http://www.gimp.org.es/> (accessed May 29, 2022).
- [32] “Procesamiento del Lenguaje Natural (NLP) | Aprende Machine Learning.” <https://www.aprendemachinelearning.com/procesamiento-del-lenguaje-natural-nlp/> (accessed Jun. 08, 2022).
- [33] “DALL-E | IDIS.” <https://proyectoidis.org/dall-e/> (accessed Jun. 08, 2022).
- [34] “AlphaStar: Mastering the real-time strategy game StarCraft II.” <https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii> (accessed Jun. 08, 2022).

Capítulo 13. Apéndices

En este capítulo se van a describir los diferentes apéndices incluidos en la documentación.

13.1 Glosario y Diccionario de Datos

Por orden alfabético, todos los términos que se consideren importantes en la aplicación con una descripción breve de su significado dentro de la aplicación.

- **Acción:** Se refiere a una fase del juego en la que el jugador interactúa con el tablero.
- **Arma:** Se refiere a las cartas usadas dentro del juego.
- **Carta:** Es la representación de las cartas físicas del juego, hace referencia a las cartas de la baraja de cartas de armas.
- **Convolución:** Es un tipo de transformación que se utiliza y da nombre a las redes neuronales convolucionales.
- **Guerrera:** Este término hace referencia al elemento central del juego, que dictamina el ganador de cada partida.
- **Jugador:** Hace referencia a uno de los dos participantes de una partida del juego, puede referirse al usuario o a la red neuronal.
- **Python:** Lenguaje de programación utilizado en todo el proyecto.
- **Redes neuronales:** Es un tipo de tecnología utilizada en la inteligencia artificial usadas para aprender patrones.
- **Tablero:** Hace referencia al conjunto de datos total en un momento determinado dentro de una partida.

13.2 Contenidos

A continuación, se va a describir el contenido del archivo adjunto y online

13.2.1 Contenido del repositorio en Git

Este repositorio se encuentra en la url: <https://github.com/samueldomorenov/El-Favor-de-las-Guerreras> y tiene el mismo contenido que el descrito en el siguiente punto.

13.2.2 Contenido del archivo adjunto

Directorio	Contenido
<i>./ Directorio raíz del Archivo adjunto</i>	Contiene un fichero <i>anaconda.yaml</i> para el despliegue del entorno de desarrollo, así como un fichero <i>.gitignore</i> para la configuración de Git.
<i>./src</i>	Contiene la estructura de los archivos fuente del proyecto.
<i>./src/main/python</i>	Ficheros correspondientes al código fuente de la aplicación.
<i>./src/main/recursos</i>	Contiene los ficheros estáticos, los modelos generados y los ficheros de parametrización de la aplicación.
<i>./src/test/python</i>	Contiene los ficheros correspondientes al código fuente de las pruebas unitarias.
<i>./doc</i>	Contiene el documento de documentación tanto en formato <i>.docx</i> como <i>.pdf</i> , así como un directorio con anexos utilizados para el desarrollo de la documentación.

Ilustración 13.1 - Contenido del archivo adjunto

13.2.3 Contenido del fichero de ejecución

Como el fichero de ejecución es demasiado pesado para poder ser subido a las plataformas, se ha decidido subir al OneDrive personal: https://unioviedo-my.sharepoint.com/:u:/g/personal/uo266321_uniovi_es/EUaR1pmkJhVNgnO7Lyq8WcQBjwBxRc6gLBvEtZuh356BoQ?e=Im9cCB

En este fichero zip tenemos los siguientes contenidos:

Directorio	Contenido
<i>./ Directorio raíz del Archivo adjunto</i>	Contiene un fichero <i>ElFavorDeLasGuerreras.exe</i> que es el archivo ejecutable.
<i>./recursos</i>	Contiene los ficheros estáticos, los modelos generados y los ficheros de parametrización de la aplicación.

Ilustración 13.2 - Contenido del fichero de ejecución

13.3 Índice Alfabético

A

aprendizaje, 3, 13, 17, 23, 24, 33, 82, 83, 84, 125, 126
Aprendizaje, 23

B

Bot, 40, 41, 44, 58, 84, 86, 109, 114

C

Competición, 20

E

entrenamiento, 23, 34, 36, 39, 40, 41, 43, 50, 51, 52, 53, 61, 62, 65, 69, 70, 84, 99, 110, 111, 123, 125, 126
Entrenamiento, 34, 39, 43, 52, 53, 54, 61, 62, 99, 123

G

Geisha, 17, 18, 19, 20, 21, 22, 116

H

Hanamikoji, 1, 3, 4, 17, 18, 19, 22, 33, 124

P

Python, 3, 13, 14, 16, 27, 30, 36, 65, 81, 82, 84, 135

R

red neuronal, 13, 14, 15, 16, 24, 33, 34, 36, 40, 41, 43, 44, 50, 51, 52, 53, 54, 60, 61, 62, 65, 66, 69, 70, 71, 72, 73, 80, 81, 83, 84, 85, 99, 110, 111, 114, 123, 125, 126, 127, 135
Regalo, 20
Renuncia, 20

S

Secreto, 19, 21
Software, 3, 29

T

tablero, 3, 16, 18, 33, 34, 35, 36, 40, 41, 43, 44, 45, 47, 52, 53, 54, 55, 56, 57, 58, 59, 60, 62, 63, 66, 67, 69, 70, 71, 74, 77, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 102, 105, 108, 109, 115, 118, 121, 122, 124, 135
TensorFlow, 4, 36, 61, 81, 83
Tkinter, 16, 45, 54, 59, 62, 63, 64, 65, 124