**UNIVERSIDAD DE OVIEDO**

|  |  |
| --- | --- |
| Dibujo1PortadaPFC.jpg | Dibujo2PortadaPFC.jpg |

ESCUELA DE INGENIERÍA INFORMÁTICA

**TRABAJO FIN DE GRADO**

Versión digital del juego Hanamikoji utilizando redes neuronales

|  |  |
| --- | --- |
| **Vº Bº del Director del**  **Proyecto** | **DIRECTOR: Vicente García Diaz**  **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 Victoria por estar siempre ahí para cualquier cosa.

Gracias a mis compañeros de carrera: Santi Rangel, Santy Vargas, Manu Mateos, Manu Valencia, Trivi, Diego Gonzalez, Miguel Torres, Antonio Muñoz, Emmanuel, 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, Jesus, Marta Rasines, Sibila, Masaaki, Angel, Lucía, Marta Aroba, Yandru, Tomas, Darisee, Calesio, Isma, Alexis

*Resumen*

El trabajo consiste en una versión en digital del juego de mesa Hanamikoji, en el que el jugador se enfrenta a una inteligencia artificial entrenada usando una red neuronal que aprende a jugar usando aprendizaje profundo. Para ello se ha implementado la lógica del juego en Python y se le suministrará a la librería de Python TensorFlow, en la que se programará la red neuronal. De manera que llegue a tener un nivel suficiente como para llegar a ser un reto para el jugador.

*Palabras Clave*

Índice General

[Capítulo 1. Memoria del Proyecto 7](#_Toc104634625)

[Capítulo 2. Introducción 8](#_Toc104634626)

[Capítulo 3. Aspectos Teóricos 9](#_Toc104634627)

[Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales 10](#_Toc104634628)

[Capítulo 5. Análisis 11](#_Toc104634629)

[Capítulo 6. Diseño del Sistema 12](#_Toc104634630)

[Capítulo 7. Implementación del Sistema 13](#_Toc104634631)

[Capítulo 8. Desarrollo de las Pruebas 14](#_Toc104634632)

[Capítulo 9. Manuales del Sistema 15](#_Toc104634633)

[Capítulo 10. Conclusiones y Ampliaciones 16](#_Toc104634634)

[Capítulo 11. Planificación del Proyecto y Presupuesto finales 17](#_Toc104634635)

[Capítulo 12. Referencias Bibliográficas 18](#_Toc104634636)

[Capítulo 13. Apéndices 19](#_Toc104634637)

Índice de Figuras

# Memoria del Proyecto

# Introducción

## 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.

## Evaluación de alternativas tecnológicas

En lo que respecta a la inteligencia artificial, actualmente está en auge el uso de redes neuronales convolucionales 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

### Relativas a la inteligencia artificial

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

#### 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.

#### 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.

### Relativas a la interfaz gráfica de usuario

#### Servicios Web

En un principio se planteó la posibilidad de implementar la aplicación como un servicio web, por lo que la interfaz gráfica sería implementada en algún framework web. Esta idea fue desechada ya que llevaría demasiado tiempo hacer esta implementación y no es el objetivo del trabajo.

#### 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, 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.

#### Tkinter

Al final se optó por utilizar Tkinter, 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.

# Aspectos Teóricos

Esta sección está destinada a describir brevemente aquellos conceptos, herramientas y tecnologías existentes que vamos a usar en nuestro proyecto.

## Hanamikoji

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

### Reglas del juego original

Este es un extracto de (*Hanamikoji - Reglas Del Juego de Cartas - Happy Meeple*, n.d.) que son las reglas que vienen en la caja física del juego

#### 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

#### 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.

##### 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.

##### 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:

###### 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.

Puedes revisar esta carta en cualquier momento.

###### Renuncia

Imagen que contiene caja, alimentos

Descripción generada automáticamente 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.

###### Regalo

Imagen que contiene taza

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

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.

Imagen de la pantalla de un celular con texto e imágenes

Descripción generada automáticamente con confianza baja

Ilustración 3.2 Ejemplo de acción de regalo

###### Una caja de cartón Descripción generada automáticamente con confianza bajaCompetición

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

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.

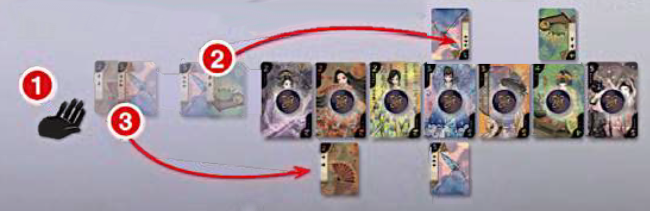


Ilustración 3.3 Ejemplo de acción de competición

##### 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.

#### Finalización del Juego



Ilustración 3.4 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.

### Cambio de nomenclatura

Para este proyecto 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**

### Mas información

Para más información acerca del juego Hanamikoji se pueden consultar las siguientes fuentes:

* (*Reseña: Hanamikoji | Misut Meeple*, n.d.)
* (*Hanamikoji y Shadows in Kyoto*, n.d.)

## Redes neuronales

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

### Machine Learning

Para entender el término *Redes neuronales* primero hay que tener claro que significa el concepto de *Machine Learning. 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 dentro de los datos de entrenamiento tenemos la solución deseada
* **Aprendizaje no supervisado**: El conjunto de datos de entrenamiento no contiene 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 el 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.

### **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.

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.

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 formulas 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.

Diagrama

Descripción generada automáticamente

Ilustración 3.5 Ejemplo de red neuronal

### 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.

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

#### **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.

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 siguiente imagen.

Imagen que contiene Diagrama

Descripción generada automáticamente

Ilustración 3.6 Ejemplo de operación de convolución

#### **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 max-pooling, que transforma el valor promedio de un conjunto de pixeles a su valor máximo, generando así las imágenes.

Por ejemplo, para la siguiente imagen se utiliza una operación de max-pooling de 2x2.

Imagen que contiene biombo, reloj

Descripción generada automáticamente

Ilustración 3.7 Ejemplo de operación de max-pooling

### Mas información

Para más información se puede consultar las siguientes fuentes, de donde se han sacado algunas imágenes utilizadas en este documento:

(*Deep Learning – Introducción Práctica Con Keras - Jordi TORRES.AI*, n.d.)

(*Pasos Para Entrenar Una Red Neuronal Profunda - Código Fuente*, n.d.)

(*Layer Activation Functions*, n.d.)

(*¡Redes Neuronales CONVOLUCIONALES! ¿Cómo Funcionan? - YouTube*, n.d.)

# Planificación del Proyecto y Presupuesto Iniciales

Para poder llevar a cabo el proyecto de una manera ordenada se procede a realizar una planificación inicial 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.

## Planificación Inicial

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 figura 4.1.

|  |  |  |
| --- | --- | --- |
|  | 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 |

Figura 4.1. Tareas de la planificación inicial

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 inicial tal y como se muestra en el diagrama de la imagen 4.2

Gráfico

Descripción generada automáticamente

Figura 4.2. Diagrama de planificación

## Presupuesto Inicial

El presupuesto inicial 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.

### Coste de personal

Para la elaboración del presupuesto inicial 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 salarial[[1]](#footnote-1) 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 salaril1 de 45.370€ anuales, tendría un coste total de 23,63€ la hora.
* **Desarrollador junior:** 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.

### Coste de la compra de hardware

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

### Costes indirectos

Para los costes indirectos hay que aclarar que el software a utilizar será sido software libre con la excepción de Windows 10 y el paquete de office incluidos en los costes de material de oficina. De esta manera los costes indirectos serían los siguientes:

|  |  |  |  |
| --- | --- | --- | --- |
| **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 € |
| Gastos de material de oficina | 4 | 20,00 € | 80,00 € |

Tabla 4.1 Costes indirectos

Con un total de 480,00 €

### 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](#_Planificación_Inicial).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 4.2 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 480,00 €

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** |  | 4.464,83 € |
| Definición de Objetivos | 156,95 € |  |
| Definición de Requisitos | 313,90 € |  |
| Presupuesto | 156,95 € |  |
| Investigación de Tecnologías existentes | 2.825,11 € |  |
| Diseño de Interfaces | 404,76 € |  |
| Diseño de Clases | 607,15 € |  |
| **Implementación** |  | 1.500,63 € |
| Módulo de Controlador | 441,36 € |  |
| Módulo de Interfaz Gráfica | 353,09 € |  |
| Módulo de Red Neuronal | 706,18 € |  |
| **Pruebas** |  | 911,82 € |
| Diseño de Pruebas | 113,98 € |  |
| Pruebas Unitarias | 341,93 € |  |
| Pruebas de Integración | 227,95 € |  |
| Pruebas de Usabilidad | 113,98 € |  |
| Pruebas de Accesibilidad | 113,98 € |  |
| **Documentación** |  | 1.255,61 € |
| Elaboración de la Documentación | 941,70 € |  |
| Revisión de la Documentación | 313,90 € |  |

Tabla 4.3 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 4.708,52 €, que supone un incremento de la parte que se va a mostrar al cliente de un 138%.

### 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** |  | **2.403,30 €** |
| Diseño de Interfaces | 961,32 € |  |
| Diseño de Clases | 1.441,98 € |  |
| **Implementación** |  | **3.564,01 €** |
| Módulo de Controlador | 1.048,24 € |  |
| Módulo de Interfaz Gráfica | 838,59 € |  |
| Módulo de Red Neuronal | 1.677,18 € |  |
| **Pruebas** |  | **2.165,57 €** |
| Diseño e implementación de pruebas | 2.165,57 € |  |
|  |  |  |
| **Total** |  | **8.132,88 €** |

Tabla 4.4 Partida simplificada

# 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.

## 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 por Takamagahara en japones, 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.

Para llevar a cabo esto la aplicación se separará en 3 módulos diferentes.

### Módulo de entrenamiento

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 2.2.1.1 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.

### 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.

### Módulo de controladores

Ya que los módulos de entrenamiento e 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.

## 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.

### Obtención de los Requisitos del Sistema

Aquí veremos los requisitos funcionales y no funcionales identificados.

#### Requisitos funcionales

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

##### Módulo entrenamiento

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 | Creación del modelo de la red neuronal | La red neuronal creará un modelo que será entrenado con los datos procesados. |
| R.F.1.1.3 | Generación y guardado del 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 5.1 – Requisitos funcionales de la red neuronal

##### 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 5.2 – Requisitos funcionales del usuario

##### 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 5.3 – Requisitos funcionales del sistema

#### 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. |
| 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. |

Tabla 5.4 – Requisitos no funcionales

### 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.

#### 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.

#### 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 solidos que los ofrecidos por la aplicación podrían usarse para generar nuevos valores para el modelo de la red neuronal.

### Especificación de Casos de Uso

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

#### Casos de uso para el Usuario Jugador

Diagrama

Descripción generada automáticamente

Ilustración 5.2 Casos de uso del Usuario Jugador

|  |
| --- |
| **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 5.5 Caso de Uso 1: Seleccionar acción.

|  |
| --- |
| **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 5.6 Caso de Uso 2: Cambiar acción seleccionada.

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

Tabla 5.7 Caso de Uso 3: Selecciona cartas.

|  |
| --- |
| **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 5.8 Caso de Uso 4: Seleccionar acción.

|  |
| --- |
| **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 5.9 Caso de Uso 5: Seleccionar cartas acción pendiente.

#### Casos de uso para el encargado del entrenamiento

Diagrama

Descripción generada automáticamente

Ilustración 5.2 Casos de uso del Encargado del Entrenamiento

|  |
| --- |
| **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 5.10 Caso de Uso 5: Generar datos de prueba.

|  |
| --- |
| **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 5.11 Caso de Uso 5: Entrenar modelo.

## 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.

### Descripción de los Subsistemas

A continuación, se veremos la descripción de los subsistemas.

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

**Red neuronal**: Forma parte de fundamental del módulo de entrenamiento, 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, 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 accione 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.

### 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 un matiz 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 entrenamiento mediante unos ficheros con el modelo y los pesos.

## 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.

### 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 representa 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.

Diagrama

Descripción generada automáticamente

Figura 5.3. Diagrama de clases

### Descripción de las Clases

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

#### Módulo de Entrenamiento

|  |
| --- |
| **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. 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. | |

|  |
| --- |
| **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 | |

#### Módulo de Controladores

|  |
| --- |
| **Nombre de la Clase** |
| Partida Controller |
| 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 | |
| **Tablero controller:** Instancia del controlador del tablero  **Win:** Flag (bandera) para saber cuando 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. | |

|  |
| --- |
| **Nombre de la Clase** |
| Tablero Controller |
| 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 | |

|  |
| --- |
| **Nombre de la Clase** |
| 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. | |

|  |
| --- |
| **Nombre de la Clase** |
| Neural Network Controller |
| 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  **Prediccion:** 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. | |

|  |
| --- |
| **Nombre de la Clase** |
| Jugador Controller |
| 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. | |

#### 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 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 accion:** Devuelve la acción seleccionada por el usuario  **Cerrar:** Termina el bucle de Tkinter y cierra la ventana | |

## 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.

### Descripción de la Interfaz

Diagrama

Descripción generada automáticamente

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 cuenteo de final de ronda.

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

### 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.

#### 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 el se le enviará dicha acción al controlador de la partida para que la ejecute y siga la ronda.

#### 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.

# Diseño del Sistema

En este apartado se describe el diseño final del sistema.

## Arquitectura del sistema: Vista de bloques de construcción

En la siguiente imagen (*Imagen 6.2.a*) 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.

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Imagen 6.2.a – Vista en bloques completa

Leyenda:

|  |  |
| --- | --- |
| Icono  Descripción generada automáticamente | Persona que interactua con la aplicación |
| Interfaz de usuario gráfica, Aplicación, PowerPoint  Descripción generada automáticamente | Bloque de caja negra |
| Forma  Descripción generada automáticamente con confianza media | Bloque de caja blanca |
| Interfaz de usuario gráfica, Aplicación  Descripción generada automáticamente | Ficheros de caja negra |
| Interfaz de usuario gráfica, Aplicación  Descripción generada automáticamente | Ficheros de caja blanca |
| Tabla  Descripción generada automáticamente con confianza baja | Clase o archivo de funciones |
| Imagen en blanco y negro  Descripción generada automáticamente con confianza media | Llamada a función sin paso de datos |
| Imagen en blanco y negro  Descripción generada automáticamente con confianza baja | Llamada a función con paso de datos |

### Nivel 1

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Imagen 6.2.1.a – 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.

### Nivel 2

Diagrama

Descripción generada automáticamente

Imagen 6.2.2.a – 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 comunicase con el módulo de controladores.

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

### Nivel 3

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Imagen 6.2.3.a – 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*.

#### Controladores

Diagrama

Descripción generada automáticamente

Imagen 6.2.3.1.a – 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 numero 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.

#### Red Neuronal

Diagrama

Descripción generada automáticamente

Imagen 6.2.3.2.a - 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 *Predicción*.

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

La clase de *Predicción* 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.

#### Interfaz grafica

Diagrama

Descripción generada automáticamente

Imagen 6.2.3.2.a - 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.

## 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).

### Controladores

|  |
| --- |
| **Nombre de la Clase** |
| ControladorPartida |
| **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** |
| \_\_controladorTablero | Instancia del controlador del tablero en el que se guarda el estado del tablero y se ejecutan las acciones. |
| \_\_accionesJ1 | Cadena de texto en la que se guardan las acciones que ha ejecutado el jugador 1 para su futuro guardado. |
| \_\_accionesJ2 | Cadena de texto en la que se guardan las acciones que ha ejecutado el jugador 2 para su futuro guardado. |
| \_\_tableroJ1 | Cadena de texto en la que se guardan los tableros que le llegaron al jugador 1 y sobre los cuales formo la acción correspondiente. |
| \_\_tableroJ2 | Cadena de texto en la que se guardan los tableros que le llegaron al jugador 2 y sobre los cuales formo la acción correspondiente. |
| \_\_winnnerNumero | Guarda el número del ganador al final de cada ronda para poder saber si se ha terminado la partida. |
| \_\_winner | Copia de la instancia del jugador que ha ganado la partida. |
| \_\_jugador1 | Instancia del jugador correspondiente al orden 1. |
| \_\_jugador2 | Instancia del jugador correspondiente al orden 2. |
| \_\_jugador1Inicio | 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. |
| \_\_jugador2Inicio | 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** |
| \_\_init\_\_ | Método constructor de la clase ControladorPartida, se definen todos los atributos privados de la misma. |
| run | Método ejecutor de la clase ControladorPartida, inicializa los jugadores y ejecuta el método \_\_start con una captura de posibles errores que se mostraran en un popup. |
| \_\_initJugadores | Método que instancia los atributos de jugador1 y jugador2 dependiendo del modo y nivel seleccionados en la parametrización. |
| \_\_start | Método encargado de realizar el bucle de rondas hasta que en una de ellas haya un ganador. |
| \_\_ronda | Método encargado de inicializar la ronda correspondiente y realizar el bucle de las acciones de los jugadores. |
| \_\_accion | 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. |
| \_\_guardarAccion | Método encargado de guardar la acción en formato cadena de texto y el tablero dados en su correspondiente atributo de la clase. |
| getAccionesJ1 | Método para obtener el atributo de tipo cadena de texto: accionesJ1. |
| getAccionesJ2 | Método para obtener el atributo de tipo cadena de texto: accionesJ2. |
| getTablerosJ1 | Método para obtener para el atributo de tipo cadena de texto: tablerosJ1. |
| getTablerosJ2 | Método para obtener para el atributo de tipo cadena de texto: tablerosJ1. |
| getWinner | Método para obtener para el atributo winner que implementa una clase jugador. |

Tabla 6.2.1.a – Descripción de la clase ControladorPartida

|  |
| --- |
| **Nombre de la Clase** |
| ControladorTablero |
| **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** |
| \_\_tablero | Guarda una matriz con el estado actual del tablero. |
| \_\_mazoArmas | Guarda un array con las cartas que quedan en el mazo de robo. |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| \_\_init\_\_ | Método constructor de la clase ControladorTablero, se definen e inicializan todos los atributos privados de la misma. |
| \_\_initMazo | Método que inicializa el mazo de las cartas con todas las cartas de armas disponibles. |
| \_\_borrarAcciones | Método que se encarga de borrar del tablero las acciones realizadas por ambos jugadores. |
| \_\_borrarArmas | Método que se encarga de borrar del tablero las armas usadas por ambos jugadores. |
| \_\_robarCarta | Método que elimina del mazo de cartas una carta aleatoria y la devuelve. Lanza una excepción si el mazo no tiene cartas que robar. |
| \_\_repartoDeCartas | Método que se encarga de repartir 6 cartas del mazo a cada jugador y ordenar las manos de dichos jugadores. |
| \_\_conseguirCarta | Método que asigna una carta del mazo a la mano del jugador con el numero dado por parámetro. |
| \_\_soltarCarta | Método que recibe el array de la mano y el valor de la carta que sustituye por 0. |
| \_\_eliminarCarta | Método que recibe un array y un valor, devuelve el array eliminando el valor una vez. |

|  |  |
| --- | --- |
| \_\_getFilaAcciones | Método que recibe el índice del jugador y devuelve el índice de sus acciones usadas. |
| \_\_getMano | Método que recibe el índice del jugador y devuelve el índice de su mano. |
| \_\_ordenarMano | Método que recibe el índice de la mano del jugador y ordena su mano. |
| \_\_getNumeroCartasEn  AccionSeleccionada | 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. |
| \_\_comprobarAccion1 | Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción 1 o si esta acción ya ha sido usada. |
| \_\_comprobarAccion2 | Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción 2 o si esta acción ya ha sido usada. |
| \_\_comprobarAccion3 | Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción 3 o si esta acción ya ha sido usada. |
| \_\_comprobarAccion4 | Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción 4 o si esta acción ya ha sido usada. |
| \_\_comprobarAccion  Decision3 | Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción de decisión 3 o si esta acción no concuerda con la esperada. |
| \_\_comprobarAccion  Decision4 | Método que lanza una excepción si el número de cartas no concuerda con el tipo de acción de decisión 4 o si esta acción no concuerda con la esperada. |
| \_\_guardarAccion1 | Método que guarda la carta seleccionada del array dado por parámetro para la acción 1 en la fila de acciones del tablero dada por parámetro y elimina esa carta de la mano dada por parámetro. |
| \_\_guardarAccion2 | Método que guarda la carta seleccionada del array dado por parámetro para la acción 2 en la fila de acciones del tablero dada por parámetro y elimina esa carta de la mano dada por parámetro. |
| \_\_guardarAccion3 | Método que guarda las cartas seleccionadas del array dado por parámetro la acción 3 en la fila de acción pendiente, establece la acción 3 como usada en la fila de acciones del tablero dada por parámetro y elimina esas cartas de la mano dada por parámetro. |
| \_\_guardarAccion4 | Método que guarda las cartas seleccionadas del array dado por parámetro la acción 4 en la fila de acción pendiente, establece la acción 4 como usada en la fila de acciones del tablero dada por parámetro y elimina esas cartas de la mano dada por parámetro. |
| \_\_guardarAccionDecision3 | Método que guarda las cartas seleccionadas del array dado por parámetro para la acción de decisión 3 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. |

|  |  |
| --- | --- |
| \_\_guardarAccionDecision4 | 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. |
| \_\_guardarSecreto | 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. |
| \_\_sumarCarta | Método que dada una carta suma 1 a la columna correspondiente a esa carta y a la fila dada. |
| \_\_getGanador | Método que calcula todos los puntos de cada jugador y devuelve el ganador. |
| \_\_manoLlena | Método que comprueba el número de cartas en la mano dada y devuelve y está llena o no. |
| initRonda | Método que inicializa el mazo con todas las cartas menos una y reparte las cartas iniciales. |
| jugadorRobaCarta | Método que, dado el índice del jugador, roba una carta del mazo de cartas y la guarda en la mano de dicho jugador. |
| getVistaTablero | Método que, dado el índice del jugador, devuelve una matriz de información parcial para ese jugador. |
| realizarAccion | 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. |
| hayAccionPendiente | 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. |
| finalizarTurno | 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 6.2.1.b – Descripción de la clase ControladorTablero

|  |
| --- |
| **Nombre de la Clase** |
| ControladorBot |
| **Descripción de la clase** | |
| Clase controladora del jugador de bot aleatorio. | |
| **Atributos** | |
| **Nombre** | **Descripción** |
| \_\_miNombre | Define el nombre para leerlo en los logs. |
| \_\_miNumero | Define el orden del jugador, puede ser 1 o 2. |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| \_\_init\_\_ | Método constructor de la clase ControladorBot, recibe el nombre y el numero para guardarlo en sus respectivos atributos. |
| decidirAccion | 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. |
| decidirAccionDeSeleccion | 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. |
| finish | Método que sirve para cerrar los hilos pendientes de los jugadores, en este caso no es necesario cerrar ninguno. |
| getMiNombre | Método para obtener para el atributo de tipo cadena de texto: miNombre. |
| getMiNumero | Método para obtener para el atributo de tipo cadena de texto: miNumero. |

Tabla 6.2.1.c – Descripción de la clase ControladorBot

|  |
| --- |
| **Nombre de la Clase** |
| ControladorJugador |
| **Descripción de la clase** | |
| Clase controladora del jugador controlado por el usuario a través de la interfaz gráfica. | |
| **Atributos** | |
| **Nombre** | **Descripción** |
| \_\_miNombre | Define el nombre para leerlo en los logs. |
| \_\_miNumero | Define el orden del jugador, puede ser 1 o 2. |
| \_\_GUI | Instancia la clase GUI\_Tkinter que corresponde a la interfaz gráfica de usuario con la que interactuará el usuario. |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| \_\_init\_\_ | Método constructor de la clase ControladorJugador, recibe el nombre y el numero para guardarlo en sus respectivos atributos. Además, inicializa el atributo GUI que implementa la clase GUI\_Tkinter. |
| decidirAccion | 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 \_\_pedirAccion. |
| decidirAccionDeSeleccion | 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 \_\_pedirAccion. |
| \_\_pedirAccion | Método que unifica los Métodos de decidirAccion y decidirAccionDeSeleccion en el que se llama a los Métodos de printTabla, start y obtenerAccion para pintar el tablero en la interfaz de usuario y esperar a que el usuario seleccione la acción que desee. |
| finish | Método que sirve para cerrar el hilo de la interfaz gráfica. |
| getMiNombre | Método para obtener para el atributo de tipo cadena de texto: miNombre. |
| getMiNumero | Método para obtener para el atributo de tipo cadena de texto: miNumero. |

Tabla 6.2.1.d – Descripción de la clase ControladorJugador

|  |
| --- |
| **Nombre de la Clase** |
| ControladorRedNeuronal |
| **Descripción de la clase** | |
| Clase controladora del jugador controlado por la red neuronal entrenada anteriormente. | |
| **Atributos** | |
| **Nombre** | **Descripción** |
| \_\_miNombre | Define el nombre para leerlo en los logs. |
| \_\_miNumero | Define el orden del jugador, puede ser 1 o 2. |
| \_\_prediccion | Instancia la clase Prediccion que corresponde a la parte de la red neuronal encargada de generar predicciones. |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| \_\_init\_\_ | Método constructor de la clase ControladorRedNeuronal, recibe el nombre y el numero para guardarlo en sus respectivos atributos. Además, inicializa el atributo Prediccion que implementa la clase Prediccion. |
| decidirAccion | 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 \_\_procesarAccion. |
| \_\_procesarAccion | 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. |
| \_\_obtenerCartasEnMano  YAccionesPosibles | 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 \_\_procesarAccion. |
| \_\_obtenerAccionCount | Método que devuelve el número de cartas que debe tener la acción a realizar dada por parámetro. Este método es necesario para el Método \_\_procesarAccion. |
| \_\_eliminarCarta | 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 \_\_procesarAccion. |
| decidirAccionDeSeleccion | 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 \_\_procesarAccionDeSeleccion. |
| \_\_procesarAccionDe  Seleccion | 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. |
| \_\_seleccionarCartasAccion  DeSeleccionRegalo | Método encargado de procesar el resultado emitido por la red neuronal para la acción de selección específica de tipo regalo para una acción pendiente dada. |
| \_\_seleccionarCartasAccion  DeSeleccionCompeticion | Método encargado de procesar el resultado emitido por la red neuronal para la acción de selección específica de tipo competición para una acción pendiente dada. |

|  |  |
| --- | --- |
| \_\_crearAccionCompleta | Método que crea un array acción a partir del tipo de acción y las cartas seleccionadas. |
| finish | Método que sirve para cerrar los hilos pendientes de los jugadores, en este caso no es necesario cerrar ninguno. |
| getMiNombre | Método para obtener para el atributo de tipo cadena de texto: miNombre. |
| getMiNumero | Método para obtener para el atributo de tipo cadena de texto: miNumero. |

Tabla 6.2.1.e – Descripción de la clase ControladorRedNeuronal

|  |
| --- |
| **Nombre de la Clase** |
| ControladorGeneradorDatos |
| **Descripción de la clase** | |
| Clase controladora de la generación de datos para el entrenamiento de la red neuronal. | |
| **Atributos** | |
| **Nombre** | **Descripción** |
| \_\_partidasGanadas1 | Define el número de partidas que ha ganado el primer jugador. |
| \_\_partidasGanadas2 | Define el número de partidas que ha ganado el segundo jugador. |
| \_\_controladorPartida | Instancia la clase ControladorPartida, que se va instanciando con cada nueva simulación. |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| \_\_init\_\_ | Método constructor de la clase ControladorGeneradorDatos, se inicializan los atributos de la clase. |
| run | Método ejecutor de la clase ControladorGeneradorDatos, reinicia los archivos y ejecuta partidas un número de veces parametrizado. |
| \_\_guardarGanador | Método que dado un jugador obtiene los datos de su partida y los guarda en los ficheros de partidas ganadas. |
| \_\_resetArchivo | Método que elimina los archivos generados anteriormente y los deja limpios para una nueva generación. |
| \_\_guardarEnArchivo | Método de escritura en fichero del texto dado. |

Tabla 6.2.1.f – Descripción de la clase ControladorGeneradorDatos

### Red Neuronal

|  |
| --- |
| **Nombre de la Clase** |
| Entrenamiento |
| **Descripción de la clase** | |
| Clase encargada de utilizar los datos de entrenamiento para guardar el modelo entrenado. | |
| **Atributos** | |
| **Nombre** | **Descripción** |
| \_\_cnn | Contendrá una instancia de la clase Sequential de TensorFlow. |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| \_\_init\_\_ | Método constructor de la clase Entrenamiento, se define el atributo cnn. |
| run | Método ejecutor de la clase Entrenamiento, se realiza el preprocesado de datos, la creación del modelo y el entrenamiento. |
| \_\_preProcesadoDeDatos | Método encargado de cargar los datos de entrenamiento desde los ficheros y transformarlos en matrices y arrays que reconoce la red neuronal. |
| \_\_creacionModelo | Método encargado de instanciar el atributo cnn con la clase Sequential. |
| \_\_establecerCapas | Método encargado de definir las capas y filtros que va a tener la red neuronal. |
| \_\_complileAndFit | Método encargado de compilar el modelo para su entrenamiento. |
| \_\_guardarModelo | 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 Prediccion. |

Tabla 6.2.2.a – Descripción de la clase Entrenamiento

|  |
| --- |
| **Nombre de la Clase** |
| Prediccion |
| **Descripción de la clase** | |
| Clase encargada de utilizar el modelo entrenado para generar predicciones. | |
| **Atributos** | |
| **Nombre** | **Descripción** |
| \_\_cnn | Contendrá una instancia de la clase Sequential. |
| \_\_resultado | Contendrá el resultado de la última predicción realizada. |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| \_\_init\_\_ | Método constructor de la clase Entrenamiento, se define los atributos e inicializarlos. |
| \_\_cargarModelo | Método encargado de cargar el modelo entrenado de los ficheros .h5. |
| predecir | Método que, dado un tablero de entrada, genera una predicción y la guarda en el atributo resultado. |
| obtenerPrediccionCampo | 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 6.2.2.b – Descripción de la clase Prediccion

### Interfaz Gráfica

|  |
| --- |
| **Nombre de la Clase** |
| GUI\_Tkinter |
| **Descripción de la clase** | |
| Clase encargada de generar la ventana principal de la aplicación usando la librería tkinter. | |
| **Atributos** | |
| **Nombre** | **Descripción** |
| \_\_accionGuardada | Array en el que se van guardando la información de la acción que se está generando. |
| \_\_cartasRestantes | El número de cartas que faltan para terminar de completar la acción actual. |
| \_\_accionPendiente | Array con la acción pendiente que llega. |
| \_\_window | Instancia la clase Tk de tkinter. |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| \_\_init\_\_ | Método constructor de la clase GUI\_Tkinter, se definen e inicializan todos los atributos privados de la misma. |
| printTabla | Método encargado de pintar el tablero que nos llega. |
| \_\_limpiarAccion | Método encargado de vaciar el array de acción guardada y borrar del tablero la acción que se había seleccionado. |
| \_\_comprobarAccion  Pendiente | Método encargado de comprobar si existe una acción pendiente y modificar el tablero para que se muestre la misma. |
| \_\_bloquearAcciones  NormalesYMano | Método encargado de deshabilitar todos los botones de las acciones y de la mano. |
| \_\_addGuerreras | Método encargado de añadir una fila de imágenes de cartas grandes. |
| \_\_addMarcadores | Método encargado de añadir una fila de marcadores dada una lista de números. |
| \_\_addSusAcciones | Método encargado de añadir una fila con las acciones usadas del adversario. |
| \_\_addMisAcciones | Método encargado de añadir una fila con las acciones usadas del jugador. |
| \_\_addMiMano | Método encargado de añadir una fila con las cartas de la mano del jugador. |
| \_\_addAccionSeleccionada | Método encargado de añadir el label con la acción seleccionada. |
| \_\_addAccionPendiente | Método encargado de añadir las cartas de la acción pendiente. |
| \_\_addAccionPendiente5 | Método encargado de añadir las cartas de la acción pendiente para el tipo 5. |
| \_\_addAccionPendiente6 | Método encargado de añadir las cartas de la acción pendiente para el tipo 6. |
| \_\_addCartaGrande | Método para añadir una imagen de carta grande en la posición dada con la imagen dada. |
| \_\_addMarcadorValor | Método para añadir el marcador del valor en la posición dada con la imagen dada. |
| \_\_addMarcador | Método para añadir marcadores en la posición dada con el texto dado. |
| \_\_addAccionPropia | Método para añadir un botón, en la posición dada, de acción activa/inactiva dependiendo del tipo dado. |

|  |  |
| --- | --- |
| \_\_addAccionEnemiga | Método encargado de añadir una imagen de acción realizada/no realizada en la posición dada. |
| \_\_getTextoAccion | Método encargado de seleccionar el texto para un tipo dado. |
| \_\_addCartaPeque | Método encargado de añadir un botón con una carta pequeña. |
| \_\_addCartaAccionPendiente | Método encargado de añadir un botón de una carta de acción pendiente. |
| \_\_addCartaOculta | Método encargado de añadir un botón con una carta oculta. |
| \_\_addMarcoExplicativo | Método encargado de añadir un botón con una carta oculta. |
| \_\_addLabelConImagen | 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. |
| \_\_addLabelConImagenY  TamanioFilas | Método encargado de añadir un label con una imagen para una posición, con un tamaño, imagen, texto, numero de filas y borde dados. |
| \_\_addLabelConSoloTexto | Método encargado de añadir un label in imagen para una posición, con un tamaño, texto y borde dados. |
| \_\_addBotonConImagen | Método encargado de añadir un botón con una imagen para una posición, con un tamaño, imagen, texto, numero de filas y borde dados. |
| \_\_addButtonConTexto | Método encargado de añadir un botón con un texto para una posición, con una imagen, texto y borde dados. |
| \_\_seleccionarAccion | 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. |
| \_\_seleccionarCarta | 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. |
| \_\_seleccionarCarta  Pendiente5 | Método que se ejecuta al seleccionar una carta cuando esta la acción de selección pendiente de tipo 5. |
| \_\_seleccionarCarta  Pendiente6 | Método que se ejecuta al seleccionar una carta cuando esta la acción de selección pendiente de tipo 6. |
| \_\_noAccion | Método auxiliar para no hacer nada al pulsar acciones que no están disponibles. |
| \_\_printAceptar | Método que pinta el botón de aceptar para enviar la acción seleccionada. |
| \_\_borrarAceptar | Método que borra el botón de aceptar para que no se envíe una acción a medias. |
| start | Método que inicia el bucle de tkinter. |
| \_\_pressAceptar | Método sale del bucle de tkinter para poder seguir con la ejecución del programa y aplicar la acción seleccionada. |
| cerrar | Método que destruye la ventana de tkinter cuando se termina la partida. |
| obtenerAccion | Método que devuelve el atributo de accionGuardada. |

Tabla 6.2.3.a – Descripción de la clase GUI\_Tkinter

|  |
| --- |
| **Nombre de la Clase** |
| Popup\_Tkinter |
| **Descripción de la clase** | |
| Clase encargada de generar la ventana pequeña de información (o popup). | |
| **Métodos** | |
| **Nombre** | **Parámetros y descripción** |
| sendMensaje | Método que construye el popup con el texto dado y un botón de aceptar para cerrarlo. |

Tabla 6.2.3.b – Descripción de la clase Popup\_Tkinter

## Vista de tiempo de ejecución

### 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* llaman 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.

### 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 imagen 6.3.2.a.

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 y 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.

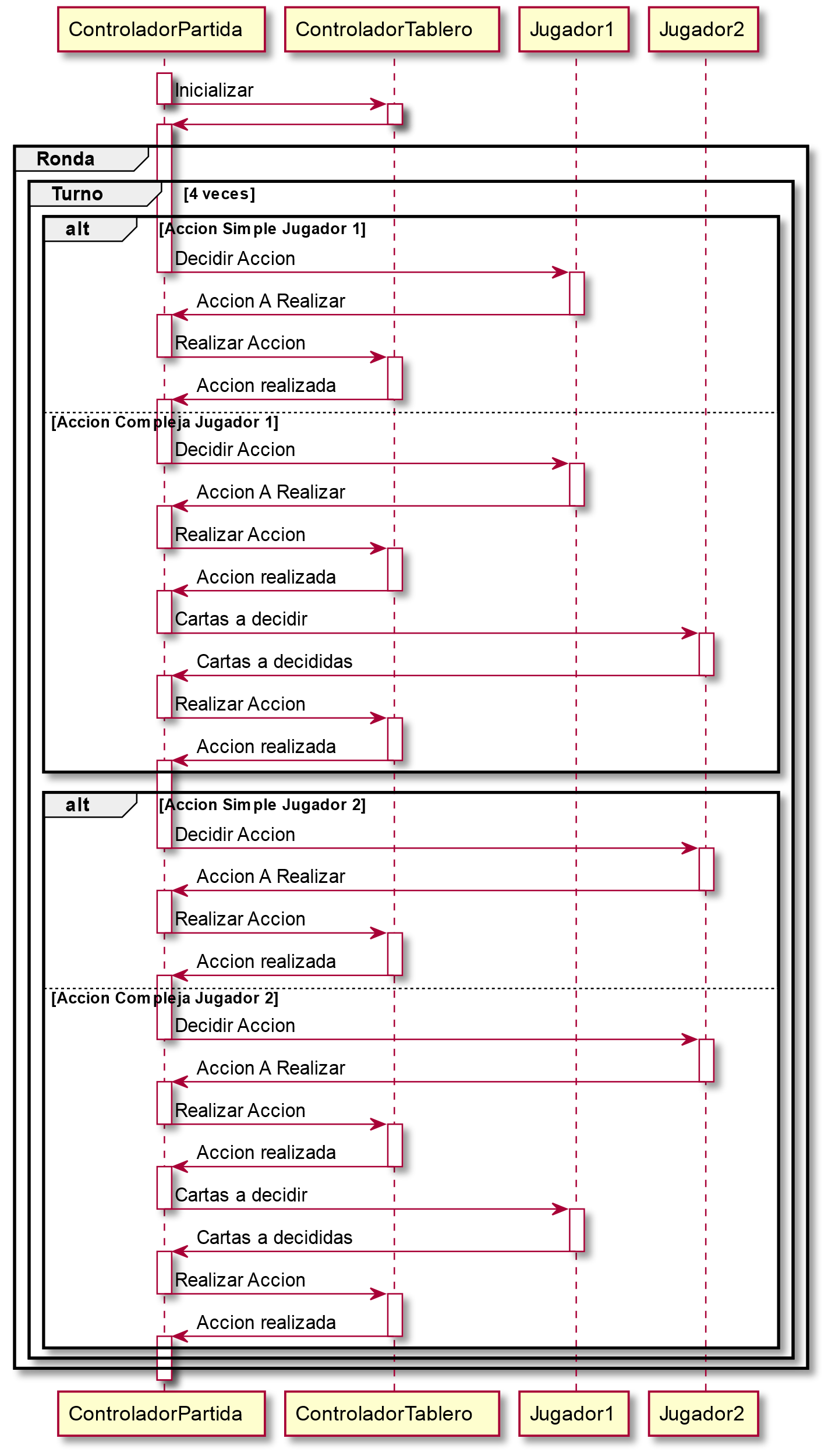


Imagen 6.3.2.a – Diagrama de flujo de una partida

## 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.

Interfaz de usuario gráfica, Diagrama, Aplicación, Teams

Descripción generada automáticamente

Imagen 6.4.a – Ejemplo de interfaz de usuario

### Acciones del adversario

Forma

Descripción generada automáticamente con confianza media

Imagen 6.4.1.a – 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 Imagen 4.1.1.a.

Una pantalla de proyector

Descripción generada automáticamente con confianza media

Imagen 6.4.1.b – 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.

### Guerreras, armas usadas y favor

*Imagen de la pantalla de un celular con texto e imágenes

Descripción generada automáticamente con confianza baja*

Imagen 6.4.2.a – 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 (imagen 4.1.2.b) o en la parte inferior en caso de este a favor del jugador (imagen 4.1.2.c).

|  |  |
| --- | --- |
| Un dibujo de una persona  Descripción generada automáticamente con confianza media  Imagen 6.4.2.b Ejemplo de guerrera con el favor del adversario | Imagen que contiene monitor, reloj, pantalla, firmar  Descripción generada automáticamente  Imagen 6.4.2.c – 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 jugador 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.

### Acciones del jugador

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Imagen 6.4.3.a – 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 selecciones 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.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 6.4.3.b – 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 interactiva, ya que el usuario no puede volver a utilizarlas.

Forma

Descripción generada automáticamente con confianza media

Imagen 6.4.3.b – 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.

### Mano del jugador

Pintura de un reloj

Descripción generada automáticamente con confianza media

Imagen 6.4.4.a – 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.

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

Imagen 6.4.4.b – 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.

Un dibujo de una persona

Descripción generada automáticamente con confianza media

Imagen 6.4.4.c – Ejemplo de mano del jugador bloequeada

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.

### Acción elegida

Imagen que contiene reloj, objeto, pantalla, tabla

Descripción generada automáticamente

Imagen 6.4.5.a – 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.

#### Acción por realizar

En la primera casilla se muestra la acción que ha seleccionado el jugador (ver sección 4.1.3). A continuación, se muestran las cartas que ha seleccionado el jugador. Es importante destacar que en 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.

#### Acción de decisión

Pantalla de video juego

Descripción generada automáticamente con confianza media

Imagen 6.4.5.2.a – 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.

### 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 numero 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.

### 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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Imagen 6.4.7.a – Ventana de información de resultado

# Implementación del Sistema

# Desarrollo de las Pruebas

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

## Pruebas Unitarias

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

### Pruebas del controlador del Bot

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

#### 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 1 libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción 1 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 1 libre y tiene solo 1 carta en mano a elegir. Devuelve la acción 1 con la carta de la mano. | 5 | 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 2 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 2 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 3 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 3 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 4 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 4 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 |

#### 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 3 y tiene todas las cartas iguales. Devuelve la acción de selección 3 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 3 y tiene dos cartas iguales. Devuelve la acción de selección 3 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 3 y tiene las tres cartas iguales. Devuelve la acción de selección 3 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 4 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 4 y tiene las dos opciones iguales. Devuelve la acción de selección 4 con la opción posible. | 17 | Correcto | OK |

### Pruebas del controlador de la Red Neuronal

#### 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 1 libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción 1 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 1 libre y tiene solo 1 carta en mano a elegir. Devuelve la acción 1 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 2 libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción 2 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 2 libre y tiene solo 2 cartas, las 2 iguales, en mano a elegir. Devuelve la acción 2 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 3 libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción 3 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 3 libre y tiene solo 3 cartas, todas iguales, en mano a elegir. Devuelve la acción 3 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 4 libre y tiene 7 cartas distintas en mano a elegir. Devuelve la acción 4 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 4 libre y tiene solo 4 cartas, todas iguales, en mano a elegir. Devuelve la acción 4 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 |

#### 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 3 y tiene todas las cartas iguales. Devuelve la acción de selección 3 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 3 y tiene dos cartas iguales. Devuelve la acción de selección 3 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 3 y tiene las tres cartas iguales. Devuelve la acción de selección 3 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 4 y tiene las dos opciones distintas. Devuelve la acción de selección 4 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 4 y tiene las dos opciones iguales. Devuelve la acción de selección 4 con la opción posible. | 34 | Correcto | OK |

### Pruebas del controlador del Tablero

#### 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 |

#### 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 |

#### 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 adversario, pero no las cartas. | Cuando el tablero está en la situación del final de la segunda ronda, devuelve una vista en la que se ven las acciones usadas por el adversario, pero no se ven las cartas que han sido usadas en esas acciones 1 y 2. | 39 | Correcto | OK |
| 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 3, 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 4, devuelve una vista en la que se ve dicha acción pendiente. | 41 | Correcto | OK |

#### 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 |

#### 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 secreto 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 secreto 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 secreto 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 secreto 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 renuncia 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 renuncia 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 renuncia 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 renuncia 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 regalo 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 regalo 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 regalo 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 regalo que contiene un numero incorrecto de cartas y devuelve una excepción. | 53 | Excepción | OK |
|  |  |  |  |  |
| Acción de tipo competición y el tipo de acción está disponible. | Se realiza una acción de tipo competición 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 competición 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 competición 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 competición 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 regalo 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 regalo 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 regalo 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 regalo 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 competición 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 competición 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 competición para un tablero en el que no está a la espera de esa acción y devuelve una excepción. | 61 | Excepción | OK |
| Acción de tipo decisión competición y la acción tiene un numero incorrecto de cartas. | Se realiza una acción de selección pendiente de tipo competición que contiene un numero incorrecto de cartas y devuelve una excepción. | 62 | Excepción | OK |

# Manuales del Sistema

## Despliegue del entorno

Descargar en instalar [Anaconda](https://www.anaconda.com/products/individual)

Una vez instalado Anaconda, desde la aplicación de promt 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>*

Abrir el navegador de Anaconda Anaconda Navigator, en Aplications 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.

## Ejecución y configuración

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

[Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja](https://github.com/samuelmorenov/El-Favor-de-las-Guerreras/blob/main/doc/images/spyder.png)

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

Para cambiar la configuración, abrir el archivo *param.properties* que está en \src\main\resources. Hay 3 campos configurables:

* MODO: Para elegir entre **Generar datos** (1), **Entrenar a la red neuronal** (2) o **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) o **Red neuronal entrenada** (2)

# Conclusiones y Ampliaciones

# Planificación del Proyecto y Presupuesto finales

# Referencias Bibliográficas

# Apéndices

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. [↑](#footnote-ref-1)