[Capítulo 5. Análisis 2](#_Toc95589536)

[5.1 Determinación del Alcance del Sistema 2](#_Toc95589537)

[5.1.1 Módulo de entrenamiento 2](#_Toc95589538)

[5.1.2 Módulo de interfaz 2](#_Toc95589539)

[5.1.3 Módulo de controladores 3](#_Toc95589540)

[5.2 Requisitos del Sistema 3](#_Toc95589541)

[5.2.1 Obtención de los Requisitos del Sistema 3](#_Toc95589542)

[5.2.2 Identificación de Actores del Sistema 6](#_Toc95589543)

[5.2.3 Especificación de Casos de Uso 7](#_Toc95589544)

[5.3 Identificación de los Subsistemas en la Fase de Análisis 10](#_Toc95589545)

[5.3.1 Descripción de los Subsistemas 10](#_Toc95589546)

[5.3.2 Descripción de los Interfaces entre Subsistemas 10](#_Toc95589547)

# 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.