

Curso:

Arquitectura de Software Empresarial

Grupo:

Teoría 1 – Laboratorio B

Tema:

Caso De Estudio: Juego del Jankenpon

Autores y Códigos:

Díaz Diaz Melio Josue (0201814005)

García Romero Antonio Alfons (0201814022)

Lujan Rojas Nelvin Eduardo (0201814007)

Romero Loli Harby Breyner (0201814025)

Escuela:

Escuela Profesional de Ingeniería de Sistemas e Informática - VI Ciclo

Docente:

Mg. Macedo Alcántara Dayán Fernando

Nuevo Chimbote-Perú

**2020**

**Contenido**

[**1. Caso de Estudio** 2](#_Toc61108420)

[**2. Requerimientos** 4](#_Toc61108421)

[**2.1 Requisitos Funcionales** 4](#_Toc61108422)

[**2.2 Requisitos No Funcionales** 4](#_Toc61108423)

[**3. Diagramas** 5](#_Toc61108424)

[**3.1 Diagrama de Casos de Uso** 5](#_Toc61108425)

[**3.2 Especificación de Casos de Uso** 6](#_Toc61108426)

[**3.3 Diagramas de Robustez** 8](#_Toc61108427)

[**4. Wireframe** 13](#_Toc61108428)

[**5. MockUps** 13](#_Toc61108429)

[**6. Estrategia utilizada para obtener las clases** 14](#_Toc61108430)

[**7. Diagrama de Clases** 15](#_Toc61108431)

[ **Modelo** 15](#_Toc61108432)

[ **MVC** 15](#_Toc61108433)

[ **MVP Vista Pasiva** 15](#_Toc61108434)

[ **Persistencia** 16](#_Toc61108435)

[**8. Clases** 17](#_Toc61108436)

[**Modelo** 17](#_Toc61108437)

[**MVC** 18](#_Toc61108438)

[**MVP\_VISTA PASIVA** 18](#_Toc61108439)

[**PERSISTENCIA** 18](#_Toc61108440)

[**9. RUN** 19](#_Toc61108441)

[**10. Repositorio** 21](#_Toc61108442)

**Termómetro MVP**

# **1. Caso de Estudio**

Se requiere implementar el juego del JanKenPon. Este juego es también llamado PIEDRA PAPEL o TIJERA que consiste en mostrar una elección en forma simultánea entre dos jugadores. EL ganador de la jugada se designará de acuerdo a la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| ELEMENTO 1 | ELEMENTO 2 | **GANADOR** |
| PIEDRA | PAPEL | **PAPEL** |
| PAPEL | TIJERA | **TIJERA** |
| PIEDRA | TIJERA | **PIEDRA** |

* En caso de que los dos jugadores mostraran el mismo elemento, será un EMPATE
* La partida consistirá de 5 jugadas, el ganador será el que tenga más jugadas ganadas.
  + **Modificación:** Se establecerá un máximo de puntos y cuando uno de los 2 llegue, gana
* El juego debe funcionar tanto en consola, como en escritorio.
* El juego debe considerar la partida entre dos jugadores y también una partida contra la máquina.
  + **Modificación:** Se considera la partida entre 2 maquinas
* El juego debe estar bien validado (validaciones de entrada y del negocio)
* El juego permitirá guardar en cualquier momento de la partida.
* A menos que el juego que haya finalizado se podrá recuperar una partida *para continuarlo y terminarlo.*
* El juego debe soportar la opción de guardar en un archivo de tipo texto (txt)

**Se solicita lo siguiente:**

* Realizar la captura de Requisitos tanto funcionales como no funcionales:
* Realizar los diagramas de casos de uso y la especificación de cada uno.
* Realizar las interfaces de usuario (Mockups)
* Realizar el diagrama del dominio.
* Realizar el análisis y diseño de la aplicación. (Diagramas de Robustez y Secuencia)
* Diagrama de clases.
* Implementar la arquitectura MVC para la aplicación en Consola
* Implementar la arquitectura MVP con vista Pasiva para la aplicación en Consola y de Escritorio.
* Implementar una arquitectura de persistencia que soporte varios orígenes de datos.

# **2. Requerimientos**

## **2.1 Requisitos Funcionales**

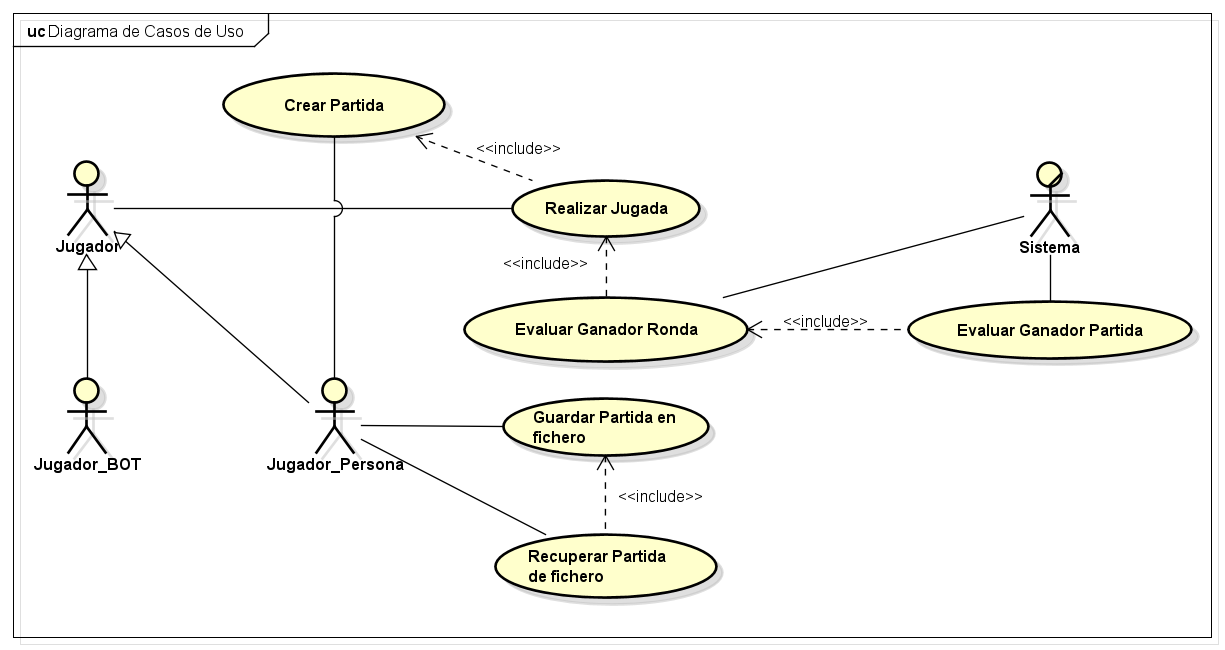
|  |  |
| --- | --- |
| **RF\_N°** | **Descripción de los Requisitos Funcionales -Administradores** |
| RF\_01 | Permitir realizar partidas entre jugadores, jugador contra la máquina o maquina contra máquina. |
| RF\_02 | Calcular con exactitud los puntos de las rondas que han sido jugadas, y determinar un ganador. |
| RF\_03 | Guardar una partida en cualquier momento de su desarrollo. |
| RF\_04 | Recuperar una partida guardada, y continuarla desde donde esta se quedó, excepto cuando ya finalizó. |
| RF\_05 | El resultado de las rondas será determinado por las reglas básicas del yankenpo. |

## **2.2 Requisitos No Funcionales**

|  |  |
| --- | --- |
| **RNF\_N°** | **Descripción de los Requisitos No Funcionales** |
| RNF\_01 | El sistema debe tener un tiempo de respuesta no mayor a 5 segundos. |
| RNF\_02 | Debe ser capaz de validar los datos de entrada para evitar excepciones. |
| RNF\_03 | Debe mostrar mensajes de error en caso de que se produzcan excepciones. |
| RNF\_04 | Debe ser capaz de funcionar tanto en modo gráfico como en modo consola. |

# **3. Diagramas**

## **3.1 Diagrama de Casos de Uso**



## **3.2 Especificación de Casos de Uso**

* Crear Partida

|  |  |  |
| --- | --- | --- |
| **Especificación de Caso de Uso** | | |
| **Caso de Uso** | Crear Partida | |
| **Precondiciones** | Iniciar Programa. | |
| **Flujo de Eventos** | **Básico** | El usuario del sistema al ver la ventana de tipos de partida hace click en la que desea elegir, y junto a esto ingresar los puntos de victoria de la partida para que el sistema los valide (Estos deben ser positivos).  A continuación, el usuario registra los jugadores depende el tipo de partida que se Eligio.  Se muestra la Ventana Pantalla en la que se realizara el siguiente caso de Uso. |
| **Alternativos** | * Puntos Negativos: El sistema muestra un mensaje que informa que los puntos de victoria no son válidos. * El usuario cierra la Ventana Menú: El sistema Termina * Error al registrar Jugadores: El sistema captura el tipo de error y muestra un mensaje que informa el tipo de error capturado. |
| **Post condiciones** | El sistema guarda la partida creada en una entidad | |

* Realizar Jugada

|  |  |  |
| --- | --- | --- |
| **Especificación de Caso de Uso** | | |
| **Caso de Uso** | Realizar jugada. | |
| **Precondiciones** | Haber creado una partida. | |
| **Flujo de Eventos** | **Básico** | El usuario del sistema ingresa a la Ventana Jugada, y el sistema le muestra una lista de las jugadas disponibles que pueden realizarse.  El usuario da clic en la jugada de su elección, y el sistema se encarga de validarla. Si la jugada elegida es correcta, el sistema guarda la jugada realizada, y la agrega a su respectiva ronda. |
| **Alternativos** | * Jugada incorrecta: El sistema muestra un mensaje que informa que la jugada elegida no es válida. * El usuario cierra la Ventana Jugada: El sistema muestra un mensaje preguntando al usuario si desea guardar la partida. * Error al registrar Jugada: El sistema captura el tipo de error y muestra un mensaje que informa el tipo de error capturado. |
| **Post condiciones** | El sistema guarda la jugada realizada por el usuario. | |

* Evaluar Ganador Ronda

|  |  |  |  |
| --- | --- | --- | --- |
| **Especificación de Caso de Uso** | | | |
| **Caso de Uso** | Evaluar Ganador Ronda | | |
| **Descripción** | Es el proceso en el cual se evalúa al ganador de la ronda. | | |
| **Precondiciones** | El jugador debe haberse registrado con anterioridad. | | |
| **Flujo de Eventos** | 1 | Básico | El sistema verifica la vista ronda, evalúa al ganador de la ronda, muestra los resultados y los guarda en su ronda respectiva. Después de concluir el análisis de la ronda, agrega otra ronda y comienza una partida. |
| 2 | Alternativos | Sí el usuario desea guardar el resultado de la ronda, necesita dar clic en “guardar resultados”. |
| **Postcondiciones** | El sistema guarda la ronda jugada. | | |

* Evaluar Ganador Partida

|  |  |  |  |
| --- | --- | --- | --- |
| **Especificación de Caso de Uso** | | | |
| **Caso de Uso** | Evaluar Ganador Partida | | |
| **Descripción** | Es el proceso en el cual se evalúa al ganador de la partida. | | |
| **Precondiciones** | El sistema evalúe el ganador de la ronda | | |
| **Flujo de Eventos** | 1 | Básico | El sistema verifica la vista partida, evalúa al ganador de la partida, muestra los resultados y los guarda en caso que exista un ganador. Después de concluir la evaluación de la partida almacena los datos. |
| 2 | Alternativos | Sí se desea guardar los resultados finales, tendría que existir un ganador obligatoriamente, en caso contrario no se podría efectuar el guardado. |
| **Postcondiciones** | Se debe realizar las jugadas para luego evaluarlas. | | |

* Guardar Partida en Fichero

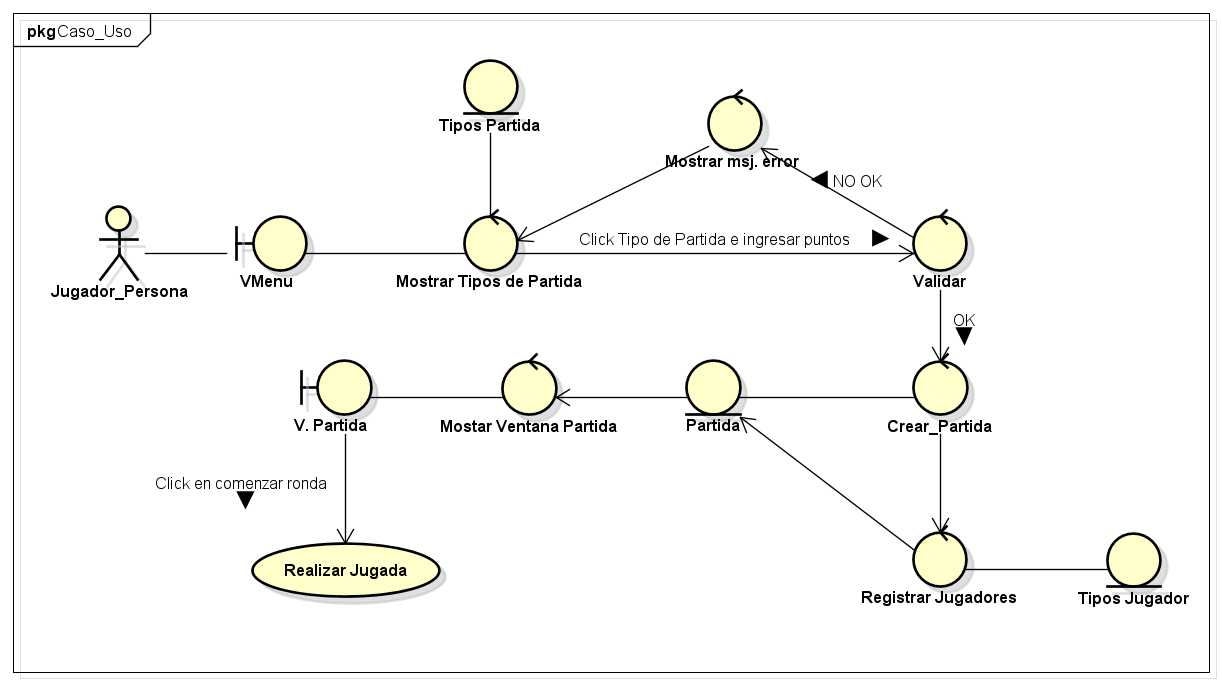
|  |  |  |
| --- | --- | --- |
| **Especificación de Caso de Uso** | | |
| **Caso de Uso** | Guardar Partida en Fichero | |
| **Precondiciones** | Partida aún no Finalizada | |
| **Flujo de Eventos** | **Básico** | El usuario del sistema al estar en una partida, tiene la opción de guardarla en fichero.  A continuación, el usuario presiona en guardar en fichero para salvar lo avanzado hasta ese momento. Se crea un nuevo fichero con nombre el id de la partida actual y se llenan con los datos de los jugadores y las jugadas en cada ronda existente Cada vez que se guarde una partida existe la opción de iniciar una nueva sin afectar los datos de la guardada. |
| **Alternativos** | * Partida ya Finalizada: El sistema deshabilita la opción de seguir jugando. * Error al guardar datos: El sistema captura el tipo de error y muestra un mensaje que informa el tipo de error capturado. * Fichero no Existente: El sistema muestra un mensaje de error informando que no existe la dirección del fichero. |
| **Post condiciones** | El sistema guarda la partida creada en una entidad | |

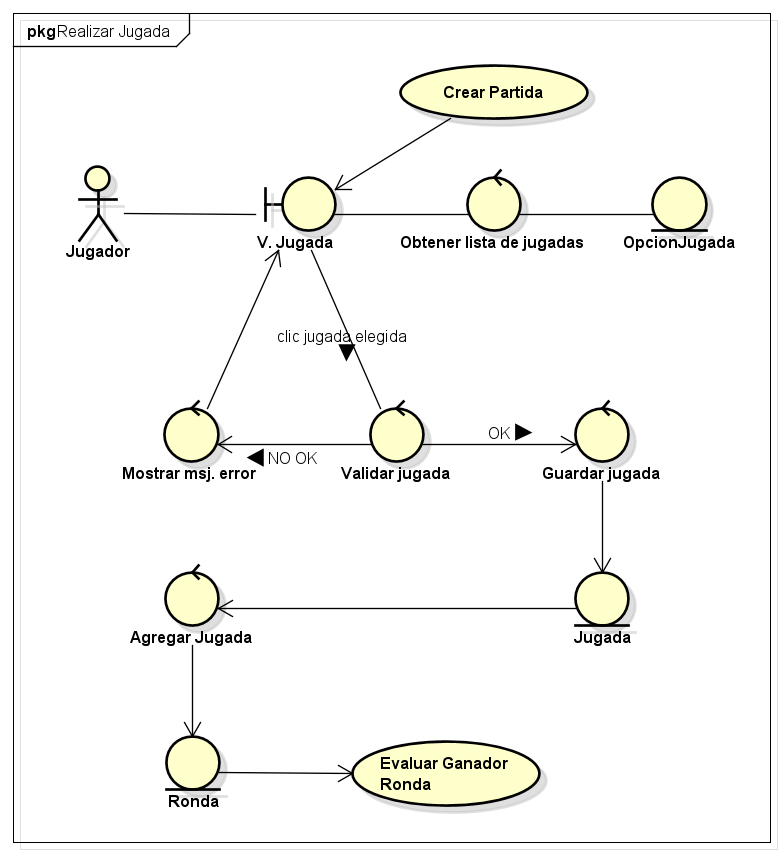
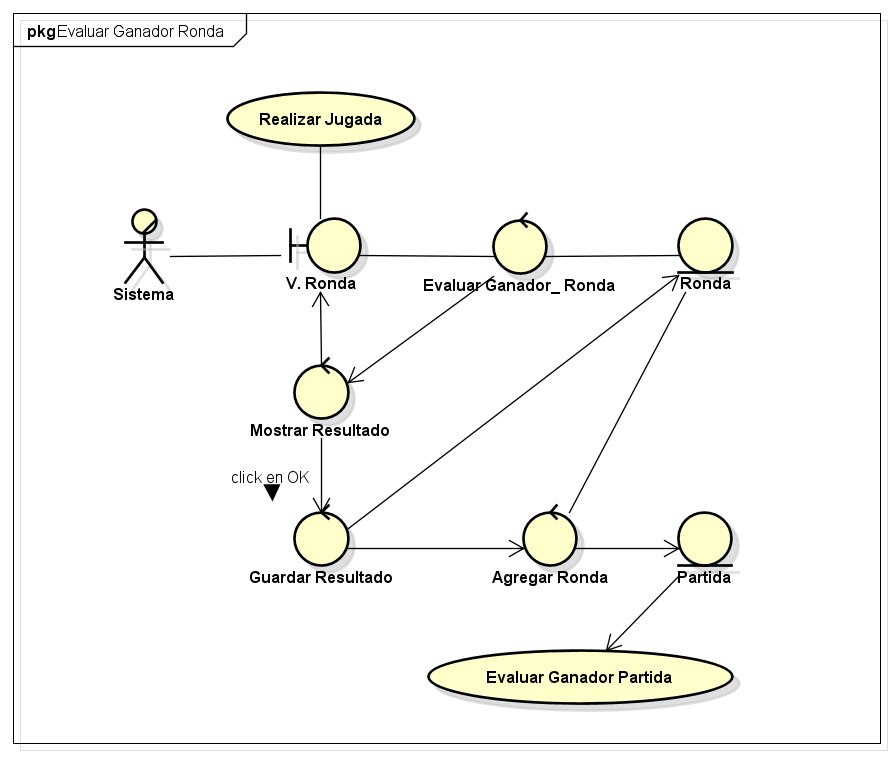
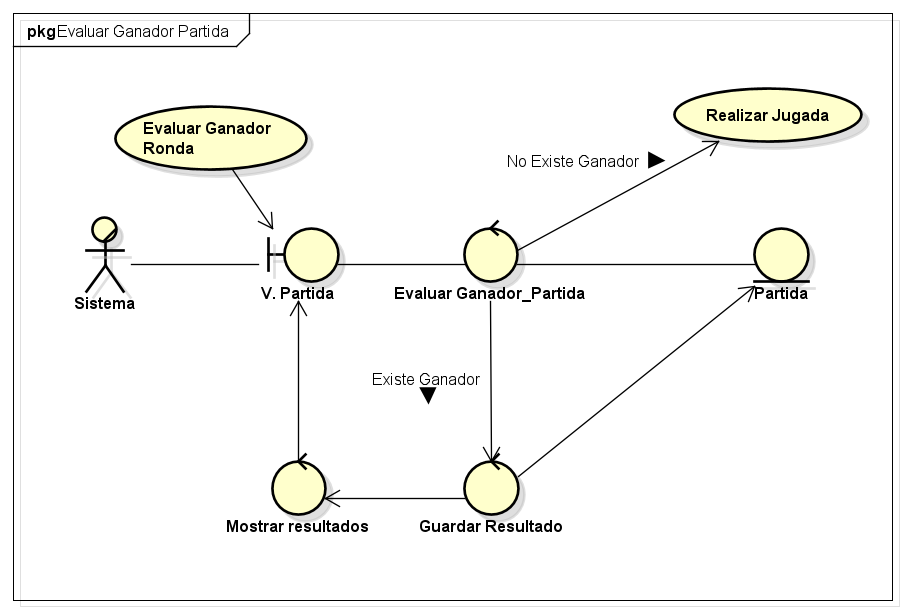
* Recuperar Partida de Fichero

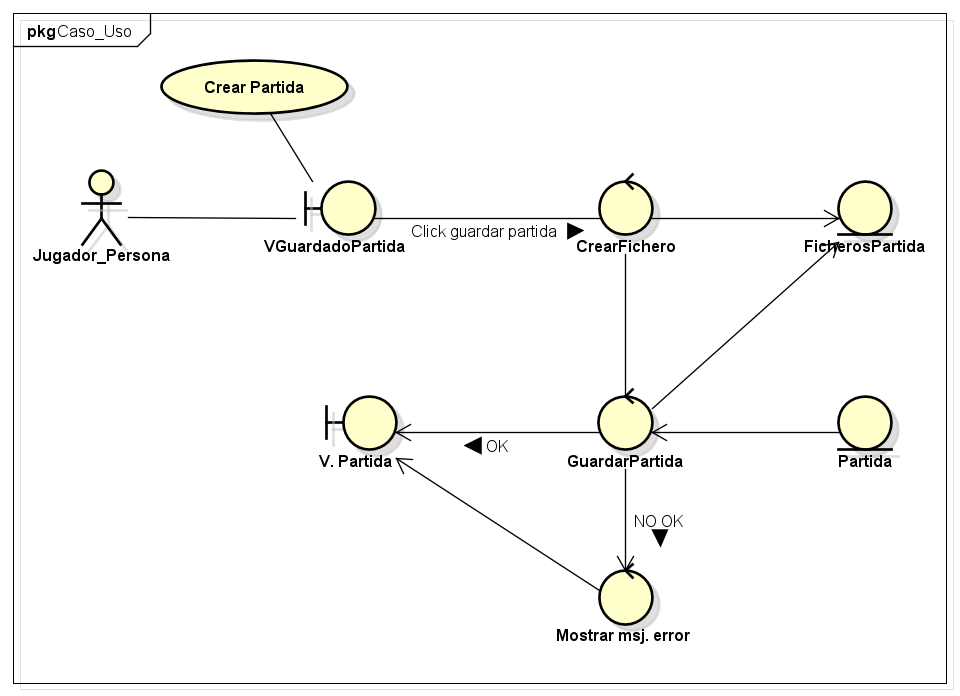
|  |  |  |
| --- | --- | --- |
| **Especificación de Caso de Uso** | | |
| **Caso de Uso** | Recuperar Partida de Fichero | |
| **Precondiciones** | Partida guardada en fichero | |
| **Flujo de Eventos** | **Básico** | El usuario tiene la opción de iniciar una nueva partida o reanudar una partida ya comenzada que se ha guardado en el fichero. En este caso el usuario tiene que dar clic en recuperar partida para comenzar a jugar una partida ya guardada anteriormente. |
| **Alternativos** | * Partida guardada: El sistema habilita la reanudación de la partida guardada. * Fichero no Existente: El sistema muestra un error de fichero de una partida que no ha sido guardado, haya finalizado o no existe. |
| **Post condiciones** | Reanudación de Partida | |

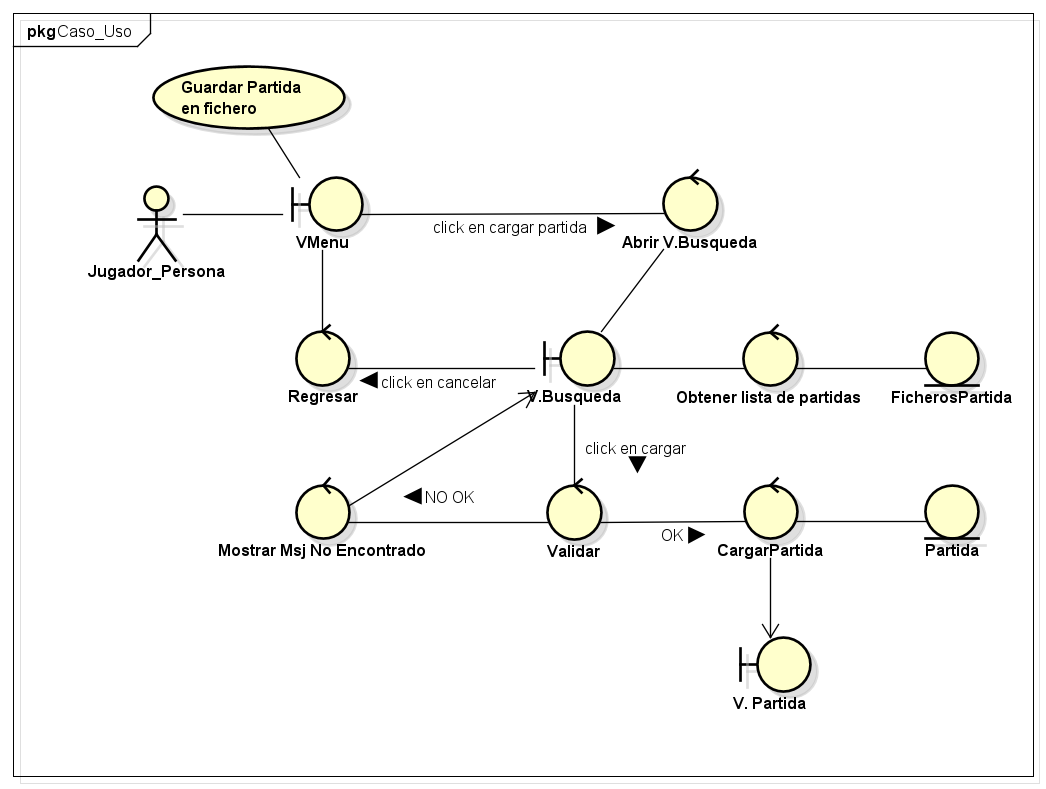
## **3.3 Diagramas de Robustez**

* Crear Partida



* Realizar Jugada
* Evaluar Ganador Ronda
* Evaluar Ganador Partida
* Guardar Partida en Fichero



* Recuperar Partida de Fichero

# **4. Wireframe**

**https://lucid.app/lucidchart/invitations/accept/18249a9c-1e79-487e-a65d-957d1107dc75**

# **5. MockUps**

# **6. Estrategia utilizada para obtener las clases**

# **7. Diagrama de Clases**

* **Modelo**
* **MVC**
* **MVP Vista Pasiva**
* **Persistencia**

# **8. Clases**

**Modelo**

* MTemperatura

**public** **class** **Mtemperatura** {

**private** Unidades unidadEntrada;

**private** Unidades unidadResultado;

**private** **double** valorEntrada;

**private** **double** valorResultado;

*//getters and setters*

}

* Unidades

**public** **enum** Unidades {

CELSIUS('C'), FAHRENHEIT('F'), KELVIN('K');

**private** **char** caracter;

Unidades (**char** c){

**this**.caracter=c;

}

**public** **char** getCaracter(){

**return** **this**.caracter;

}

}

**MVC**

* Vista
  + IVTermometro
  + VTermometroSwing
* Controlador
  + Presentador

**MVP\_VISTA PASIVA**

* Vista
  + IVTermometro
* Presentador
  + Presentador

**PERSISTENCIA**

# **9. RUN**

# **10. Repositorio**

* GIT
* OneDrive