
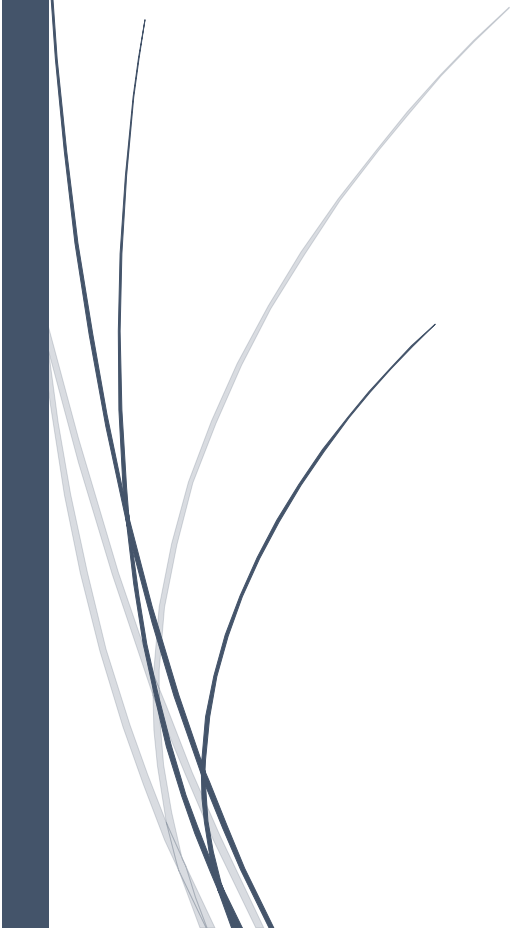


8 de diciembre de 2023



Obligatorio de Diseño y desarrollo de aplicaciones e Ingeniería del software

Documentación del Proyecto: UNO



Lautaro De León
Santiago Eguzquiza



Autor/es	Cambios	Fecha	Versión
Lautaro De León, Santiago Eguzquiza	Inicio del desarrollo del documento incluyendo requerimientos principales.	20/10/23	1.0
Lautaro De León, Santiago Eguzquiza	Extensión de requerimientos e inclusión del diagrama de clases.	24/10/23	1.0.1
Lautaro De León, Santiago Eguzquiza	Modificación en los requisitos funcionales, casos de uso y estimación del proyecto.	5/11/23	1.0.2
Lautaro De León, Santiago Eguzquiza	Actualización del Diagrama de Clases	12/11/23	1.1
Lautaro De León, Santiago Eguzquiza	Creación del documento SCM.	18/11/23	1.2
Lautaro De León, Santiago Eguzquiza	Elaboración del documento de pruebas del sistema.	3/12/23	1.3
Lautaro De León, Santiago Eguzquiza	Documento acerca de la Implementación de Scrum en el sistema.	7/12/23	1.4
Lautaro De León, Santiago Eguzquiza	Corrección de errores.	8/12/23	1.5

INDICE

1	INTRODUCCIÓN	4
1.1	IDENTIFICACIÓN	4
1.2	NECESIDADES	4
1.3	OBJETIVOS	4
1.4	GLOSARIO	5
1.5	EXPLICACIÓN UNO: THE GAME.....	6
2	DESCRIPCIÓN GENERAL	8
2.1	ALCANCE	8
2.2	LIMITACIONES	8
2.3	SUPUESTOS	8
2.4	ACTORES INVOLUCRADOS.....	9
3	ESPECIFICACIÓN DE REQUERIMIENTOS	10
3.1	REQUERIMIENTOS FUNCIONALES.....	10
3.2	REQUERIMIENTOS NO FUNCIONALES	14
4	CASOS DE USO	16
5	DIAGRAMAS	32
5.1	DIAGRAMA DE CLASES	32
6	GESTIÓN DE LA CONFIGURACIÓN (SCM).....	33
6.1	INTRODUCCIÓN	33
6.1.1	<i>Objetivo del documento</i>	<i>33</i>
6.1.2	<i>Alcance</i>	<i>34</i>
6.2	CONTROL DE VERSIONES	35
6.2.1	<i>Herramienta de control de versiones: GitHub</i>	<i>35</i>
6.2.2	<i>Repositorio.....</i>	<i>36</i>
6.2.3	<i>Convenciones de nomenclatura.....</i>	<i>37</i>
6.2.4	<i>Proceso de ramificación y fusión</i>	<i>38</i>
6.3	GESTIÓN DE CONFIGURACIÓN	39
6.3.1	<i>Configuración del entorno de desarrollo</i>	<i>39</i>
6.4	GESTIÓN DE CAMBIOS	41
6.4.1	<i>Proceso de solicitud de cambios.....</i>	<i>41</i>
6.4.2	<i>Control de versiones de documentación.....</i>	<i>43</i>
6.5	AUDITORIA Y SEGUIMIENTO	44
6.6	PROCEDIMIENTOS DE RESPUESTA A PROBLEMAS	46
6.6.1	<i>Resolución de conflictos</i>	<i>46</i>
6.6.2	<i>Recuperación ante fallos</i>	<i>48</i>
6.7	CONCLUSIONES	49

7	DOCUMENTO DE ESTIMACIÓN DEL PROYECTO	50
7.1	OBJETIVOS	50
7.2	ALCANCE	50
7.3	LIMITACIONES	50
7.3.1	<i>Restricciones de plataforma</i>	<i>50</i>
7.3.2	<i>Limitaciones de tiempo y recursos.....</i>	<i>50</i>
7.4	ESTIMACIÓN DE RECURSOS Y TIEMPO	51
7.4.1	<i>Equipo de desarrollo</i>	<i>51</i>
7.4.2	<i>Duración del proyecto</i>	<i>51</i>
7.4.3	<i>Recursos técnicos.....</i>	<i>51</i>
7.5	DESCOMPOSICIÓN DE TAREAS.....	52
7.5.1	<i>Desarrollo de la versión para un jugador</i>	<i>52</i>
7.5.2	<i>Desarrollo de la versión remota para un jugador</i>	<i>52</i>
7.5.3	<i>Desarrollo de la versión multijugador remoto.....</i>	<i>53</i>
8	IMPLEMENTACION DE SCRUM.....	54
8.1	ROLES EN SCRUM.....	54
8.2	ARTEFACTOS EN SCRUM	54
8.3	EVENTOS EN SCRUM.....	55
8.4	DOCUMENTO DE SCRUM.....	55
9	DOCUMENTO DE PRUEBAS DEL SISTEMA	57
9.1	OBJETIVOS.....	57
9.2	ALCANCE	57
9.3	CASOS DE PRUEBA.....	57

1 Introducción

El presente documento de especificación de requerimientos marca el comienzo de un emocionante proyecto académico, cuyo objetivo es desarrollar una versión del popular juego UNO en el entorno digital. Este proyecto se enmarca en un contexto educativo, con la finalidad de aplicar los conocimientos y habilidades adquiridos en las materias Ingeniería del Software y Diseño y Desarrollo de Aplicaciones.

A lo largo de este documento, definiremos los requerimientos funcionales y no funcionales que guiarán el desarrollo de esta aplicación. Nuestra intención es crear una experiencia de juego auténtica que respete las reglas y mecánicas tradicionales del juego UNO, al tiempo que aproveche las ventajas que ofrece el entorno digital.

1.1 Identificación

Nombre del producto: UNO: The Game ®

1.2 Necesidades

- Proporcionar una experiencia de juego digital del juego UNO que sea auténtica y entretenida.
- Aplicar conocimientos y habilidades en programación y desarrollo de software en un proyecto académico.

1.3 Objetivos

Desarrollar una versión (escalable) del juego "UNO", comenzando con una versión para un jugador, seguido de una versión remota para un solo jugador, y culminando con una versión multijugador remota.

1.4 Glosario

Caso de uso (CU): Un caso de uso es una descripción de cómo un sistema interactúa con actores externos para lograr un objetivo específico, representando una funcionalidad desde la perspectiva del usuario.

Requerimiento funcional (RF): Es una especificación de una función o capacidad que el sistema debe realizar, como una característica específica o una acción que el sistema debe llevar a cabo.

Requerimiento no funcional (RNF): Es una especificación que describe cómo debe funcionar el sistema en términos de cualidades, restricciones o atributos, como el rendimiento, la seguridad, la usabilidad o la escalabilidad, en lugar de funciones específicas.

Diagrama de clases: Es una representación gráfica que muestra las clases, sus atributos y métodos, y las relaciones entre ellas en un sistema de software. Este tipo de diagrama es ampliamente utilizado en la programación orientada a objetos para visualizar la estructura de un programa.

Java: Es un lenguaje de programación orientado a objetos ampliamente utilizado y conocido por su portabilidad, seguridad y versatilidad. Incorpora una máquina virtual (JVM) que facilita la ejecución de programas en múltiples plataformas. Java es esencial en el desarrollo de software, desde aplicaciones de escritorio hasta aplicaciones web y móviles.

1.5 Explicación UNO: The Game

Reglas del Juego

La finalidad de UNO es quedarse sin cartas en la mano, avisando con la palabra “UNO” cuando solo queda una carta por jugar. El que consigue 500 puntos primero gana. Los puntos se obtienen por las cartas que los demás jugadores aún tienen en sus manos.

Se mezclan las cartas y cada jugador recibe siete cartas. Las cartas sobrantes se colocan boca abajo en el centro y forman el mazo. La primera carta se descubre y se pone junto al mazo. Este montón es el mazo de descarte.

El primer jugador coloca una carta de su mano en el mazo de descarte. La regla es: una carta solo se puede poner encima de una carta del mismo color o del mismo número. Las cartas negras son cartas de acción especiales con reglas propias (explicadas más adelante). Si un jugador no puede poner la carta adecuada, tiene que levantar una carta de castigo del mazo. Puede jugar esta carta inmediatamente, si le sirve. Si no, le toca al siguiente jugador.

El que pone la penúltima carta, debe decir “UNO” y señala que solo tiene una última carta en la mano.

Cartas de acción

En el juego existen cartas de acción negras con distintas funciones, que se explican a continuación:

CARTA ROBA DOS: Al poner esta carta, el siguiente jugador debe levantar dos cartas y no puede poner ninguna carta en ese turno. Esta carta solo se puede poner sobre una carta del mismo color o sobre otras cartas “roba dos”. Si sale al principio del juego, se aplican las mismas reglas.

CARTA DE CAMBIO DE SENTIDO: Con esta carta se invierte la dirección. Si se estaba jugando hacia la izquierda, ahora se juega hacia la derecha y viceversa. La carta solo se puede poner sobre una carta del mismo color o sobre otra carta de cambio de sentido. Cuando esta carta sale al principio del juego, el repartidor empieza y luego sigue el jugador a su derecha.

CARTA DE PASAR TURNO: Al poner esta carta, el siguiente jugador se “salta”. La carta solo se puede poner sobre una carta del mismo color o sobre otra carta de pasar turno. Cuando esta carta sale al principio del juego, el jugador a la izquierda del repartidor se “salta” y el jugador a la izquierda de este jugador sigue el juego.

CARTA DE COMODÍN: Con esta carta el jugador elige qué color sigue en el juego. También se puede elegir el color que ya estaba. Una carta de comodín también se puede usar, aunque el jugador tenga otra carta que pueda jugar. Si sale una carta de comodín al principio del juego, el jugador a la izquierda del repartidor elige qué color va a seguir.

CARTA DE ROBA CUATRO COMODÍN: Esta es la mejor carta. El jugador elige qué color sigue en el juego. Además, el siguiente jugador debe levantar cuatro cartas. No puede poner ninguna carta en ese turno. Si esta carta sale al principio del juego, se vuelve a meter en el mazo y se saca otra carta.

2 Descripción General

2.1 Alcance

El proyecto tiene como objetivo principal la creación de una aplicación de software que permita a los jugadores disfrutar del juego UNO en un entorno digital. Esto incluye la implementación de reglas y mecánicas fieles al juego de cartas tradicional, la posibilidad de jugar partidas individuales y multijugador en línea, así como la interacción a través de una interfaz gráfica intuitiva.

2.2 Limitaciones

Si bien aspiramos a brindar una experiencia de juego completa y entretenida, reconocemos que hay ciertas limitaciones que debemos considerar. Estas pueden incluir restricciones de tiempo y recursos que pueden influir en la implementación de características adicionales o en la optimización del sistema.

Una de las limitaciones clave de este proyecto es que el juego solo estará disponible en ordenadores. No se contempla el soporte para dispositivos móviles o la accesibilidad a través de navegadores web.

2.3 Supuestos

En la planificación de este proyecto, partimos de algunos supuestos clave. Suponemos que los jugadores tendrán acceso a dispositivos con capacidades mínimas para ejecutar la aplicación, incluyendo conexión a internet para el modo multijugador. También suponemos que los jugadores estarán familiarizados con las reglas básicas del juego UNO.

2.4 Actores Involucrados

El éxito de este proyecto dependerá en gran medida de la colaboración y participación de varios actores clave. Estos actores incluyen, pero no se limitan a:

- **Desarrolladores:** El equipo encargado del diseño, implementación y pruebas del software.
- **Usuarios Finales:** Los jugadores que interactuarán con la aplicación.
- **Supervisores o Clientes:** Las partes interesadas que supervisarán el progreso y proporcionarán orientación a lo largo del desarrollo del proyecto.

3 Especificación de Requerimientos

3.1 Requerimientos Funcionales

RF1- Inicio de partida

Descripción: Los jugadores deben poder iniciar una nueva partida de UNO con un mínimo de 2 y un máximo de 4 jugadores.

Especificación: CU01

Prioridad: Alta

RF2- Distribución de cartas

Descripción: El sistema debe ser capaz de repartir 7 cartas a cada jugador al comienzo de la partida.

Especificación: CU02

Prioridad: Alta

RF3- Formar pila

Descripción: El sistema, luego de repartir las cartas debe ser capaz de dar vuelta la carta superior para dar comienzo a la pila de descarte.

Especificación: CU03

Prioridad: Alta

RF4- Reglas del juego

Descripción: El sistema debe aplicar las reglas del juego, como permitir a los jugadores jugar cartas que coincidan en número o color con la carta superior del mazo.

Especificación: CU04

Prioridad: Alta

RF5- Robar cartas

Descripción: El sistema debe permitir a un jugador robar una carta del mazo cuando no puede jugar una carta válida.

Especificación: CU05

Prioridad: Alta

RF6- Cambios de color

Descripción: El sistema debe permitir a los jugadores cambiar el color de juego mediante cartas especiales.

Especificación: CU06

Prioridad: Alta

RF7- Cambios de sentido

Descripción: El sistema debe permitir a los jugadores cambiar el sentido del juego mediante cartas especiales.

Especificación: CU07

Prioridad: Alta

RF8- Saltar

Descripción: El sistema debe permitir a los jugadores saltar turno mediante cartas especiales.

Especificación: CU08

Prioridad: Alta

RF9- Sumar puntos

Descripción: El sistema, luego de terminada la ronda, debe sumar al ganador los puntos de cada jugador dependiendo las cartas restantes de los mismos.

Especificación: CU9

Prioridad: Alta

RF10- Finalización de la ronda

Descripción: El sistema debe reconocer cuando un jugador se queda sin cartas y declarar al ganador de la ronda.

Especificación: CU10

Prioridad: Alta

RF11- Finalización de la partida

Descripción: El sistema debe reconocer cuando un jugador llega a 500 puntos y declarar al ganador de la partida.

Especificación: CU11

Prioridad: Alta

RF12 - Jugada de Carta por la CPU

Descripción: El sistema debe permitir que la CPU realice una jugada estratégica al seleccionar y jugar una carta durante su turno.

Especificación: CU12

Prioridad: Alta

RF13 - Recogida de Carta por la CPU

Descripción: El sistema debe permitir que la CPU recoja una carta según las reglas del juego.

Especificación: CU13

Prioridad: Alta

RF14 - Toma de Decisiones de la CPU para Seleccionar Carta

Descripción: El sistema debe incluir un mecanismo de toma de decisiones para que la CPU seleccione de manera inteligente la carta a jugar.

Especificación: CU14

Prioridad: Alta

RF15- Registro de Jugador

Descripción: Refiere al proceso de registro de un jugador en la aplicación, que permite a los usuarios generar una cuenta.

Especificación: CU15

Prioridad: Media

RF16- Inicio de sesión

Descripción: Refiere al proceso de inicio de sesión en la aplicación, que permite a los usuarios autenticarse en la plataforma para acceder a sus cuentas y utilizar sus servicios.

Especificación: CU16

Prioridad: Media

3.2 Requerimientos No Funcionales

RNF1- Interfaz de usuario amigable

Descripción: La interfaz de usuario debe ser intuitiva y fácil de usar, con una representación visual de las cartas y las opciones disponibles.

Prioridad: Alta

RNF2- Eficiencia

Descripción: El sistema debe responder de manera rápida a las acciones de los jugadores, minimizando el tiempo de espera entre turnos.

Prioridad: Alta

RNF3- Seguridad

Descripción: Debe haber medidas de seguridad para prevenir el fraude y proteger la privacidad de los jugadores en línea.

Prioridad: Alta

RNF4- Escalabilidad

Descripción: El sistema debe ser capaz de manejar partidas con un número variable de jugadores sin problemas de rendimiento.

Prioridad: Alta

RNF5- Rendimiento

Descripción: El juego debe ser capaz de funcionar sin problemas en hardware de gama baja y no consumir demasiados recursos del sistema.

Prioridad: Alta

RNF6- Tiempo de actividad

Descripción: Si el juego es en línea, es importante que esté disponible y en funcionamiento la mayor parte del tiempo, con tiempos de inactividad programados mínimos.

Prioridad: Alta

RNF7- Lenguaje

Descripción: El sistema deberá ser desarrollado en el lenguaje JAVA

Prioridad: Alta

4 Casos de uso

CU01 - Caso de Uso 01: INICIAR PARTIDA DE "UNO"

Actor Principal: Jugador

Descripción: Este caso de uso describe el proceso de inicio de una nueva partida de UNO, permitiendo que un mínimo de 2 y un máximo de 4 jugadores participen en la partida.

Precondiciones:

Los jugadores han accedido al juego UNO.

El sistema está en un estado de espera para iniciar una partida.

Flujo Básico:

El jugador inicia la aplicación UNO en su dispositivo.

El sistema muestra la pantalla principal del juego con la opción de "Iniciar Nueva Partida".

El jugador selecciona "Iniciar Nueva Partida".

El sistema permite al jugador configurar el número de jugadores (2, 3 o 4).

El jugador confirma la configuración.

El sistema distribuye 7 cartas a cada jugador y coloca una carta en el centro como carta de inicio.

El sistema determina al azar el jugador que inicia la partida.

El juego comienza con el primer jugador.

Flujo Alternativo:

Si en cualquier momento el número de jugadores configurado está fuera del rango permitido (menos de 2 o más de 4), el sistema mostrará un mensaje de error y permitirá al jugador ingresar un valor válido.

Postcondiciones:

Se ha iniciado una partida de UNO con el número de jugadores especificado.

Los jugadores tienen sus manos de cartas distribuidas.

El sistema ha seleccionado al primer jugador de manera aleatoria.

Prioridad: Alta

CU02 - Caso de Uso 02: DISTRIBUCIÓN DE CARTAS

Actor Principal: Sistema

Descripción: Este caso de uso describe el proceso de distribución de 7 cartas a cada jugador al comienzo de una partida.

Precondiciones:

La partida de UNO ha sido iniciada.

El juego se encuentra en el estado de preparación para la partida.

Flujo Básico:

El sistema verifica la cantidad de jugadores en la partida.

El sistema prepara un mazo de cartas UNO completo.

El sistema baraja el mazo de cartas UNO.

El sistema reparte 7 cartas a cada jugador, una por una en sentido horario.

Las cartas repartidas se retiran del mazo principal.

El sistema coloca la carta superior del mazo restante en el centro de la mesa como carta de inicio.

El juego está listo para comenzar.

Flujo Alternativo:

Si no hay suficientes cartas en el mazo para repartir 7 cartas a cada jugador, el sistema debe tomar medidas adecuadas, como rebarajar el mazo.

Postcondiciones:

Cada jugador tiene 7 cartas en su mano al comienzo de la partida.

El mazo se ha reducido en 7 cartas.

La carta de inicio se ha colocado en el centro de la mesa.

Prioridad: Alta

CU03 - Caso de Uso 03: FORMAR PILA DE DESCARTE

Actor Principal: Sistema

Descripción: Este caso de uso describe el proceso de dar vuelta a la carta superior del mazo para formar la pila de descarte al comienzo de la partida de UNO, después de que se hayan repartido las cartas a los jugadores.

Precondiciones:

Las cartas han sido repartidas a los jugadores.

El juego se encuentra en el estado de preparación para la partida.

Flujo Básico:

El sistema toma la carta superior del mazo, que no ha sido repartida a los jugadores.

El sistema voltea la carta y la coloca en el centro de la mesa, formando así la pila de descarte.

El juego está listo para comenzar a partir de la carta de inicio en la pila de descarte.

Postcondiciones:

La pila de descarte ha sido creada con la carta volteada.

La partida de UNO está lista para comenzar desde la carta de inicio.

Prioridad: Alta

CU04 - Caso de Uso 04: APLICAR REGLAS DEL JUEGO

Actor Principal: Jugador

Descripción: Este caso de uso describe el proceso de aplicar las reglas del juego UNO, lo que incluye la capacidad de los jugadores para jugar cartas que coincidan en número o color con la carta superior del mazo de descarte.

Precondiciones:

La partida de UNO está en progreso.

El sistema ha formado la pila de descarte con una carta inicial.

Flujo Básico:

El sistema muestra las cartas en la mano del jugador.

El sistema muestra la carta superior de la pila de descarte.

El jugador selecciona una carta de su mano que coincida en número o color con la carta superior de la pila de descarte.

El sistema verifica si la carta seleccionada es válida según las reglas del juego.

Si la carta es válida, el sistema la coloca en la pila de descarte y actualiza la carta superior de la pila de descarte.

Si la carta no es válida, el sistema muestra un mensaje de error y permite al jugador seleccionar otra carta.

El juego continúa con el siguiente jugador.

Flujo Alternativo:

Si un jugador no tiene cartas que coincidan en número o color con la carta superior de la pila de descarte, debe robar una carta del mazo.

Postcondiciones:

La pila de descarte se ha actualizado con la carta jugada por el jugador.

El turno se ha pasado al siguiente jugador, a menos que se requiera una acción adicional debido a una carta especial.

Prioridad: Alta

CU05 - Caso de Uso 05: ROBAR CARTAS

Actor Principal: Jugador

Descripción: Este caso de uso describe el proceso de permitir a un jugador robar una carta del mazo cuando no puede jugar una carta válida en su turno.

Precondiciones:

La partida de UNO está en progreso.

El jugador actual no tiene una carta que coincida en número o color con la carta superior de la pila de descarte.

El mazo de cartas no está vacío.

Flujo Básico:

El sistema muestra un mensaje al jugador actual indicando que no puede jugar una carta válida.

El jugador selecciona la opción "Robar Carta".

El sistema toma una carta del mazo principal, la agrega a la mano del jugador y muestra la carta robada.

El turno del jugador continúa.

El juego continúa con el siguiente jugador.

Postcondiciones:

El jugador ha robado una carta del mazo.

El juego continúa con el siguiente jugador.

Prioridad: Alta

CU06 - Caso de Uso 06: CAMBIO DE COLOR

Actor Principal: Jugador

Descripción: Este caso de uso describe el proceso de permitir a un jugador cambiar el color de juego utilizando cartas especiales, como las cartas "Cambio de Color" o "Cambio de Color +4".

Precondiciones:

La partida de UNO está en progreso.

El jugador actual ha jugado una carta especial que permite cambiar el color de juego.

Flujo Básico:

El sistema muestra las cartas en la mano del jugador actual.

El jugador selecciona una carta especial que permite cambiar el color de juego.

El sistema solicita al jugador elegir un nuevo color para el juego (Rojo, Amarillo, Verde o Azul).

El jugador elige un nuevo color.

El sistema actualiza el color de juego y la partida continúa con el siguiente jugador.

Flujo Alternativo:

Si el jugador no tiene una carta especial que permita cambiar el color de juego, el sistema no muestra la opción de cambio de color.

Postcondiciones:

El color de juego se ha cambiado según la elección del jugador.

La partida continúa con el siguiente jugador.

Prioridad: Alta

CU07 - Caso de Uso 07: CAMBIAR EL SENTIDO DEL JUEGO

Actor Principal: Jugador

Descripción: Este caso de uso describe el proceso de permitir a un jugador cambiar el sentido del juego utilizando cartas especiales, como las cartas "Cambio de Sentido" o "Salto".

Precondiciones:

La partida de UNO está en progreso.

El jugador actual ha jugado una carta especial que permite cambiar el sentido del juego.

Flujo Básico:

El sistema muestra las cartas en la mano del jugador actual.

El jugador selecciona una carta especial que permite cambiar el sentido del juego.

El sistema cambia el sentido del juego de horario a antihorario o viceversa, según corresponda.

La partida continúa con el siguiente jugador en el nuevo sentido del juego.

Flujo Alternativo:

Si el jugador no tiene una carta especial que permita cambiar el sentido del juego, el sistema no muestra la opción de cambio de sentido.

Postcondiciones:

El sentido del juego se ha cambiado según la elección del jugador.

La partida continúa con el siguiente jugador en el nuevo sentido del juego.

Prioridad: Alta

CU08 - Caso de Uso 08: SALTAR EL TURNO

Actor Principal: Jugador

Descripción: Este caso de uso describe el proceso de permitir a un jugador saltar su turno utilizando cartas especiales, como las cartas "Salto".

Precondiciones:

La partida de UNO está en progreso.

El jugador actual ha jugado una carta especial que permite saltar el turno.

Flujo Básico:

El sistema muestra las cartas en la mano del jugador actual.

El jugador selecciona una carta especial que permite saltar el turno.

El sistema salta el turno del jugador actual y pasa al siguiente jugador en sentido horario.

El juego continúa con el nuevo jugador.

Flujo Alternativo:

Si el jugador no tiene una carta especial que permita saltar el turno, el sistema no muestra la opción de salto de turno.

Postcondiciones:

El turno del jugador actual se ha saltado, y la partida continúa con el siguiente jugador.

Prioridad: Alta

CU9 - Caso de Uso 9: SUMAR PUNTOS AL GANADOR DE LA RONDA

Actor Principal: Sistema

Descripción: Este caso de uso describe el proceso de sumar puntos al jugador ganador de una ronda de UNO al final de la partida, dependiendo de las cartas restantes en las manos de los demás jugadores.

Precondiciones:

La partida de UNO ha llegado a su fin.

Se ha determinado al ganador de la ronda.

Flujo Básico:

El sistema verifica las manos de los jugadores que no ganaron la ronda.

El sistema calcula la suma de puntos en las cartas restantes en las manos de los jugadores perdedores.

El sistema asigna estos puntos al jugador ganador de la ronda.

Se muestra la puntuación actualizada de todos los jugadores.

Se inicia una nueva ronda o se declara al ganador del juego, según corresponda.

Postcondiciones:

El ganador de la ronda recibe los puntos de los jugadores perdedores basados en las cartas restantes en sus manos.

La puntuación de los jugadores se actualiza según las reglas del juego UNO.

Prioridad: Alta

CU10 - Caso de Uso 10: FINALIZACIÓN DE LA RONDA Y DECLARACION DEL GANADOR

Actor Principal: Sistema

Descripción: Este caso de uso describe el proceso de finalizar una ronda de UNO y declarar al ganador cuando un jugador se queda sin cartas.

Precondiciones:

La partida de UNO está en progreso.

Un jugador ha jugado su última carta y se ha quedado sin cartas en su mano.

Flujo Básico:

El sistema detecta que un jugador se ha quedado sin cartas en su mano.

El sistema declara al jugador que se quedó sin cartas como ganador de la ronda.

Se muestra un mensaje de victoria al jugador ganador.

La ronda se da por finalizada y se procede a la sumatoria de puntos o al inicio de una nueva ronda, según corresponda.

Postcondiciones:

La ronda se ha dado por finalizada.

El ganador de la ronda ha sido declarado.

Se ha avanzado al siguiente paso del juego, como la sumatoria de puntos o el inicio de una nueva ronda.

Prioridad: Alta

CU11 - Caso de Uso 11: FINALIZACION DE LA PARTIDA Y DECLARACION DEL GANADOR

Actor Principal: Sistema

Descripción: Este caso de uso describe el proceso de finalizar una partida de UNO y declarar al ganador cuando un jugador alcanza 500 puntos.

Precondiciones:

La partida de UNO está en progreso.

Un jugador ha alcanzado un total de 500 puntos.

Flujo Básico:

El sistema detecta que un jugador ha alcanzado un total de 500 puntos.

El sistema declara al jugador que ha alcanzado 500 puntos como el ganador de la partida.

Se muestra un mensaje de victoria al jugador ganador.

La partida se da por finalizada.

Postcondiciones:

La partida se da por finalizada.

El ganador de la partida ha sido declarado.

Prioridad: Alta

CU12 - Caso de Uso 12: JUGADA DE CARTA POR LA CPU

Actor Principal: CPU

Descripción: Este caso de uso describe el proceso de la CPU jugando una carta durante una partida de UNO.

Precondiciones:

La partida de UNO está en progreso.

Es el turno de la CPU para jugar una carta.

Flujo Básico:

La CPU evalúa las cartas en su mano y determina la mejor carta para jugar según las reglas de UNO.

La CPU selecciona la carta que jugará y realiza la acción correspondiente según el tipo y el color de la carta (saltar turno, invertir el sentido del juego, cambiar color, etc.).

El sistema actualiza el estado del juego, teniendo en cuenta la carta jugada por la CPU.

Se muestra la carta jugada por la CPU en el área de juego.

La CPU realiza cualquier acción adicional requerida por la carta jugada.

Postcondiciones:

La carta jugada por la CPU se encuentra en la pila de descarte.

Se ha actualizado el estado del juego según la acción de la carta jugada.

El turno pasa al siguiente jugador, según las reglas de UNO.

Flujo Alternativo:

Si la CPU no puede jugar ninguna carta válida, se procede a levantar una carta y su turno termina.

Prioridad: Alta

CU13 - Caso de Uso 13: RECOGIDA DE CARTA POR LA CPU

Actor Principal: CPU

Descripción: Este caso de uso describe el proceso de la CPU levantando una carta cuando no puede jugar ninguna carta válida durante su turno en una partida de UNO.

Precondiciones:

La partida de UNO está en progreso.

Es el turno de la CPU, y no puede jugar ninguna carta válida.

Flujo Básico: La CPU levanta una carta según las reglas de UNO. El sistema añade la carta recogida a la mano de la CPU. Se actualiza el estado del juego para reflejar la adición de la carta recogida. El turno pasa al siguiente jugador, según las reglas de UNO.

Postcondiciones:

La carta levantada se encuentra en la mano de la CPU.

Se ha actualizado el estado del juego para reflejar la recogida de la carta.

El turno pasa al siguiente jugador.

Prioridad: Alta

CU14 - Caso de Uso 14: TOMA DE DECISIONES DE LA CPU PARA SELECCIONAR CARTA

Actor Principal: CPU

Descripción: Este caso de uso describe el proceso de toma de decisiones de la CPU para seleccionar la mejor carta a jugar durante su turno en una partida de UNO.

Precondiciones:

La partida de UNO está en progreso.

Es el turno de la CPU.

Flujo Básico:

La CPU evalúa las cartas en su mano y determina la mejor carta para jugar según las reglas de UNO.

La CPU selecciona la carta que jugará y realiza la acción correspondiente según el tipo y el color de la carta (saltar turno, invertir el sentido del juego, cambiar color, etc.).

El sistema actualiza el estado del juego, teniendo en cuenta la carta jugada por la CPU.

Postcondiciones:

La carta seleccionada por la CPU se encuentra en la pila de descarte.

Se ha actualizado el estado del juego según la acción de la carta jugada.

El turno pasa al siguiente jugador, según las reglas de UNO.

Flujo Alternativo:

Si la CPU no puede decidir la mejor carta para jugar, levanta una carta y termina su turno.

Prioridad: Alta

CU15 – Caso de Uso 15: REGISTRO DE JUGADOR

Actor Principal: Jugador

Descripción: Este caso de uso describe el proceso de registro de un nuevo jugador en la aplicación, permitiéndole crear una cuenta para acceder a los servicios y funcionalidades del juego.

Precondiciones:

La aplicación está instalada en el dispositivo del jugador.

El jugador no tiene una cuenta registrada en la aplicación.

Flujo Básico:

El jugador inicia la aplicación.

La aplicación muestra la pantalla de registro.

El jugador proporciona la información requerida, que puede incluir nombre, dirección de correo electrónico, contraseña, y otros datos necesarios.

La aplicación valida la información ingresada por el jugador.

Si la información es válida, la aplicación crea una cuenta para el jugador y muestra un mensaje de confirmación.

El jugador puede ahora utilizar las credenciales de la cuenta recién creada para iniciar sesión.

Flujo Alternativo:

En el paso 4, si la información proporcionada es inválida o si la dirección de correo electrónico ya está registrada en el sistema, la aplicación muestra un mensaje de error y permite al jugador corregir la información.

Postcondiciones:

El jugador ha completado el registro con éxito y ahora tiene una cuenta válida en la aplicación.

El jugador puede utilizar las credenciales de inicio de sesión creadas durante el registro para acceder a la aplicación.

Prioridad: Alta

CU16 – Caso de Uso 16: INICIO DE SESIÓN

Actor Principal: Jugador

Descripción: Este caso de uso describe el proceso de inicio de sesión en la aplicación, que permite a los jugadores autenticarse en la plataforma para acceder a sus cuentas y utilizar sus servicios.

Precondiciones:

La aplicación está instalada en el dispositivo del jugador.

El jugador tiene una cuenta registrada en la aplicación.

Flujo Básico:

El jugador inicia la aplicación.

La aplicación muestra la pantalla de inicio de sesión.

El jugador ingresa su dirección de correo electrónico y contraseña.

La aplicación verifica la autenticidad de las credenciales ingresadas.

Si las credenciales son válidas, la aplicación permite el acceso a la cuenta del jugador y muestra la pantalla principal.

Si las credenciales son inválidas, la aplicación muestra un mensaje de error y permite al jugador volver a intentar el inicio de sesión.

Flujo Alternativo:

En el paso 4, si el sistema detecta múltiples intentos fallidos de inicio de sesión, puede aplicar medidas de seguridad adicionales, como un bloqueo temporal de la cuenta o la necesidad de restablecer la contraseña.

Postcondiciones:

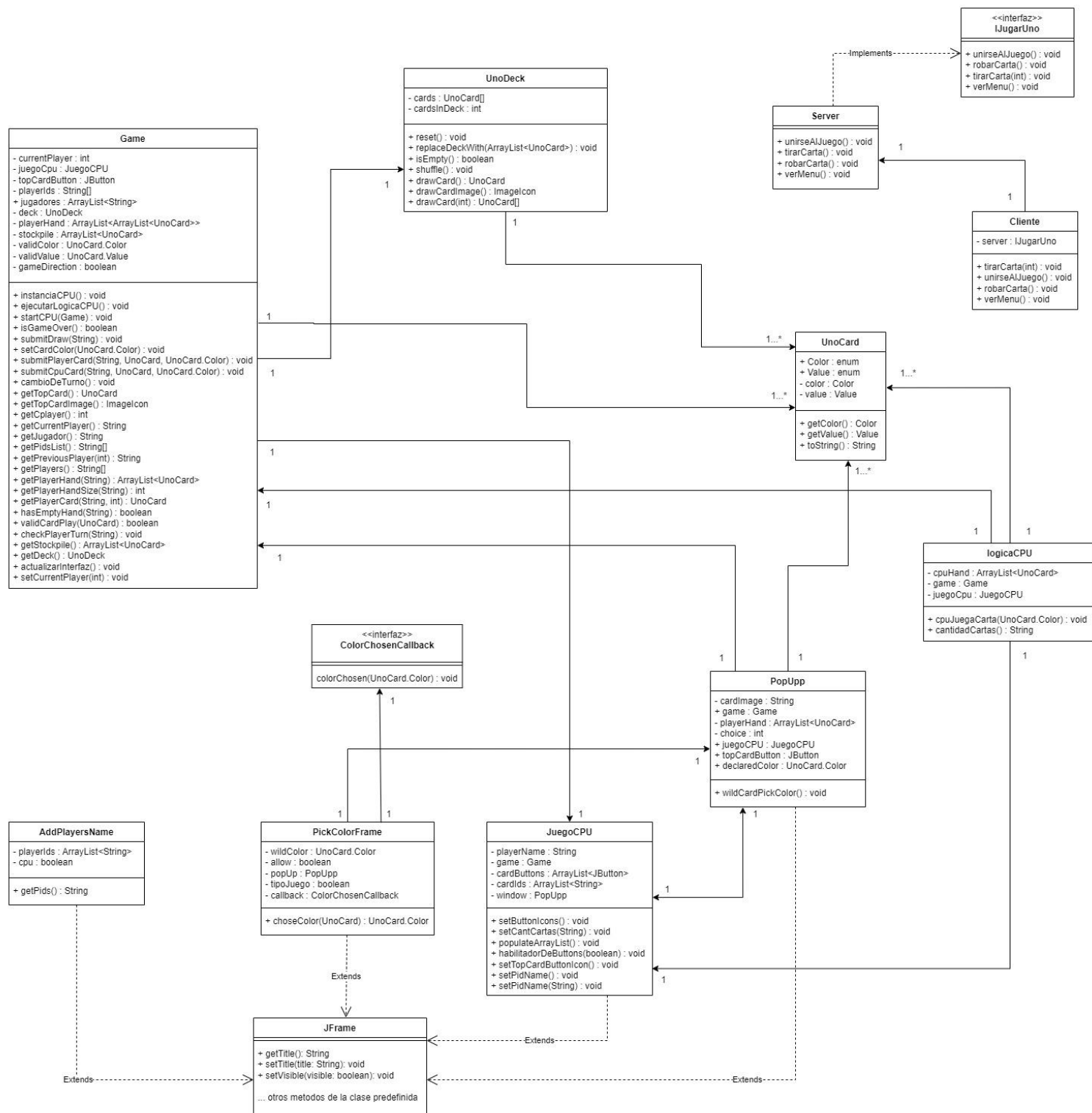
El jugador ha iniciado sesión en su cuenta, lo que le permite acceder a las funciones y características de la aplicación.

La pantalla principal de la aplicación se muestra al jugador si las credenciales son válidas.

Prioridad: Alta

5 Diagramas

5.1 Diagrama de clases



6 Gestión de la configuración (SCM)

6.1 Introducción

6.1.1 Objetivo del documento

El objetivo principal de este documento es proporcionar una guía detallada sobre la gestión de configuración y el control de versiones para el proyecto del juego UNO desarrollado en Java. Este documento servirá como referencia para todos los miembros del equipo de desarrollo y cualquier otra parte interesada.

Propósito:

- Establecer y mantener un proceso estructurado para el control de versiones y la gestión de configuración del juego UNO.
- Garantizar la coherencia y estabilidad del código fuente a lo largo del tiempo.
- Facilitar la colaboración eficiente entre los miembros del equipo de desarrollo.
- Proporcionar un marco para el seguimiento y documentación de los cambios en el sistema.

Utilización:

- **Desarrollo:**
 - Los desarrolladores utilizarán este documento como referencia para entender las prácticas y procedimientos de control de versiones.
 - Se seguirán las convenciones de nomenclatura y ramificación para garantizar un desarrollo fluido y colaborativo.
- **Mantenimiento:**
 - Los equipos de mantenimiento utilizarán esta documentación para comprender la configuración del sistema y aplicar cambios o actualizaciones de manera coherente.
- **Colaboración:**
 - Facilitará la colaboración entre los miembros del equipo, ya que todos seguirán un proceso estandarizado.
- **Auditoría:**
 - Permitirá realizar auditorías y revisiones de configuración para garantizar la integridad del sistema en cualquier momento.

6.1.2 Alcance

Este documento abarca las áreas cruciales del desarrollo del juego UNO en Java y se centra en la gestión de configuración y el control de versiones. Los aspectos específicos incluidos son:

1. **Control de Versiones:**

- Gestión de las diferentes versiones del código fuente del juego.
- Procedimientos para la creación, fusión y eliminación de ramas.
- Convenciones de nomenclatura para las versiones.

2. **Gestión de Configuración:**

- Configuración del entorno de desarrollo para garantizar uniformidad entre los desarrolladores.
- Procesos para la auditoría y seguimiento de la configuración.

3. **Gestión de Cambios:**

- Proceso para proponer y aprobar cambios en el código (Solicitud de Cambios).
- Control de versiones de la documentación relacionada con el proyecto.

4. **Auditoría y Seguimiento:**

- Registro detallado de los cambios realizados en el código fuente.
- Procedimientos para llevar a cabo auditorías de configuración.

5. **Procedimientos de Respuesta a Problemas:**

- Resolución de conflictos en la fusión de ramas.
- Procedimientos para la recuperación ante fallos del sistema.

Este documento se diseñó con el propósito de ser una referencia centralizada para todos los miembros del equipo de desarrollo del juego UNO. Abarca desde la fase inicial del desarrollo hasta el mantenimiento continuo, garantizando la coherencia y eficiencia en todas las etapas del ciclo de vida del proyecto. Cualquier cambio en estas áreas se documentará y actualizará en consecuencia.

6.2 Control de versiones

6.2.1 Herramienta de control de versiones: GitHub

GitHub es la plataforma de control de versiones que se utilizó para gestionar el código fuente del juego UNO. Proporciona una interfaz web amigable y potentes herramientas de colaboración para el desarrollo de software en equipo.

Características clave de GitHub:

- **Repositorios:** Almacén centralizado para nuestro código fuente y recursos relacionados.
- **Ramificación y Fusión:** Permite trabajar en nuevas funciones o correcciones sin afectar el código principal hasta que estemos listos para fusionar los cambios.
- **Seguimiento de Problemas:** Facilita el seguimiento de problemas, mejoras y tareas pendientes.
- **Colaboración:** Herramientas para revisar y discutir cambios antes de que se fusionen.

Documentación Relevante:

- [Documentación oficial de GitHub](#): Ofrece tutoriales y guías detalladas para aprovechar al máximo las funciones de GitHub.
- [Guía de ramificación en GitHub](#): Explica cómo utilizar ramas de manera efectiva en GitHub.

Uso recomendado:

- **Clonación del Repositorio:** Para comenzar a trabajar en el proyecto, clona el repositorio desde GitHub a tu entorno local.
- **Creación de Ramas:** Crea ramas separadas para nuevas características o correcciones.
- **Solicitudes de Fusión:** Utiliza las solicitudes de fusión en GitHub para revisar y fusionar cambios en el código principal.

GitHub es esencial para nuestro flujo de trabajo colaborativo y garantiza un control de versiones eficiente y transparente a lo largo del desarrollo del juego UNO.

6.2.2 Repositorio

El repositorio de nuestro proyecto en GitHub sigue una estructura organizada que facilita el desarrollo colaborativo y la gestión eficiente de versiones. A continuación, se describe la estructura y las ramas utilizadas:

1. Ramas principales:

- **main:** Esta es la rama principal que refleja el estado de producción del juego. Contiene el código fuente estable y probado que está listo para ser lanzado.
- **lautaro:** Esta rama se utiliza para el desarrollo activo de nuevas características o mejoras. Las contribuciones en esta rama se integran con **main** después de una revisión y prueba exitosas.
- **santiago:** Similar a la rama **lautaro**, **santiago** se utiliza para el desarrollo en paralelo. Los cambios se fusionan con **main** después de ser evaluados y confirmados.

2. Proceso de Fusiones:

- Cuando se completa el desarrollo en las ramas **lautaro** o **santiago** y los cambios han sido probados, se fusionan con la rama **main**. Esto asegura que solo las características completas y probadas lleguen a la versión principal del juego.

3. Eliminación de Ramas:

- Después de la fusión exitosa, las ramas **lautaro** y **santiago** se eliminan para mantener una estructura limpia del repositorio.

4. Creación de Nuevas Ramas:

- Después de la fusión en **main**, se crean nuevas ramas para futuros desarrollos. Esto garantiza que las ramas de desarrollo siempre se basen en la última versión estable del juego.

Este enfoque permite un desarrollo paralelo eficiente, donde múltiples funciones pueden estar en progreso al mismo tiempo sin interferir entre sí. La rama **main** siempre representa una versión estable y lista para lanzar del juego UNO.

6.2.3 Convenciones de nomenclatura

Para mantener la consistencia y facilitar la comprensión de las versiones y las ramas en nuestro proyecto, utilizamos las siguientes convenciones de nomenclatura:

Versiones:

- Seguimos un enfoque simple de numeración para nuestras versiones.
 - **Ejemplo: 1.0, 1.1, 1.2.**
- Cada número representa un cambio significativo, una nueva característica o una corrección de errores, respectivamente.

Ramas:

- Cada rama tiene un nombre descriptivo que refleja su propósito y contenido.
 - **main:** Rama principal que refleja la versión estable y probada del juego.
 - **lautaro:** Rama para el desarrollo activo de nuevas características por el desarrollador Lautaro.
 - **santiago:** Rama para el desarrollo activo de nuevas características por el desarrollador Santiago.

Proceso de Nomenclatura:

- Cuando se crea una nueva rama para el desarrollo, se le asigna un nombre relacionado con la función o tarea específica.
 - **Ejemplo: nueva-caracteristica, correccion-bug.**
- Al fusionar con la rama principal (**main**), se puede agregar una etiqueta descriptiva al commit para resaltar el propósito del cambio.

Estas convenciones facilitan la identificación y comprensión rápida de las versiones y las ramas, asegurando un proceso de desarrollo más claro y ordenado para todos los miembros del equipo.

6.2.4 Proceso de ramificación y fusión

Creación de Ramas:

1. Desarrollo de Funcionalidades:

- Cuando se inicia el desarrollo de una nueva funcionalidad, se crea una nueva rama específica para esa tarea.

2. Desarrollo Paralelo:

- Los desarrolladores Lautaro y Santiago crean sus propias ramas **lautaro** y **santiago** respectivamente para trabajar en paralelo en nuevas características o correcciones.

Fusión de Ramas:

1. Revisión y Pruebas:

- Antes de fusionar una rama de desarrollo en **main**, los cambios pasan por una revisión de código por parte del equipo y se someten a pruebas para garantizar la estabilidad.

2. Fusión con main:

- Una vez que los cambios en una rama se consideran listos, se fusionan con la rama **main**.

Eliminación de Ramas:

1. Después de la Fusión:

- Después de una fusión exitosa con **main**, la rama de desarrollo se elimina para mantener una estructura limpia.

2. Ciclo de Creación y Eliminación:

- Este proceso de creación, revisión, fusión y eliminación de ramas se repite para cada tarea o función, garantizando que siempre se trabaje sobre la última versión estable del código principal.

Este enfoque ayuda a evitar conflictos y mantiene una estructura de código organizada. Además, facilita la identificación de las características desarrolladas por cada miembro del equipo y mejora la trazabilidad de los cambios en el proyecto.

6.3 Gestión de configuración

6.3.1 Configuración del entorno de desarrollo

La configuración del entorno de desarrollo es crucial para garantizar que todos los desarrolladores trabajen en un entorno coherente y compatible. Aquí se detallan los requisitos de configuración del entorno de desarrollo para el proyecto del juego UNO:

1. Herramientas de Desarrollo:

- **Entorno de Desarrollo Integrado (IDE):**
 - Se recomienda el uso de un IDE específico, como NetBeans o Eclipse, con las configuraciones predeterminadas del proyecto.
- **JDK (Java Development Kit):**
 - Especificar la versión del JDK compatible con el proyecto. Por ejemplo, JDK 11.
- **Git:**
 - Asegurarse de que Git esté instalado y configurado en el sistema.

2. Gestión de Dependencias:

- **Sistema de Gestión de Dependencias:**
 - Utilizamos herramientas como Maven o Gradle para gestionar las dependencias del proyecto.
- **Archivo de Configuración:**
 - Incluir un archivo **pom.xml** (Maven) o **build.gradle** (Gradle) que liste las dependencias y configuraciones del proyecto.

3. Requisitos Específicos del Proyecto:

- **Configuración de Base de Datos:**
 - Si el juego UNO utiliza una base de datos, proporcionar detalles sobre cómo configurarla localmente para desarrollo.
- **Configuraciones del Servidor:**
 - Especificar cualquier configuración adicional necesaria para ejecutar el servidor del juego localmente.

4. Variables de Entorno:

- **Cualquier Variable Específica del Proyecto:**
 - Si hay variables de entorno específicas necesarias para el funcionamiento del juego, detallarlas aquí.

5. Instrucciones de Configuración:

- **Pasos para Configurar el Entorno:**
 - Proporcionar instrucciones paso a paso sobre cómo configurar el entorno de desarrollo, incluyendo la descarga de dependencias y la preparación del sistema.

6. Actualizaciones y Cambios:

- **Procedimiento para Actualizaciones del Entorno:**
 - Detallar cómo los desarrolladores deben manejar las actualizaciones y cambios en la configuración del entorno a medida que evoluciona el proyecto.

6.4 Gestión de cambios

6.4.1 Proceso de solicitud de cambios

La gestión de cambios es fundamental para mantener la integridad y la calidad del código en el proyecto del juego UNO. Aquí se describe el proceso de Solicitud de Cambios (CR) para proponer y aprobar modificaciones en el código:

1. Identificación de la Necesidad de Cambio:

- Cualquier miembro del equipo que identifique la necesidad de un cambio en el código, ya sea para corregir un error, mejorar una funcionalidad o agregar una nueva característica, inicia el proceso.

2. Creación de una Rama Específica:

- Antes de realizar cambios, el desarrollador crea una nueva rama específica para la tarea. Esta rama se basa en la rama **main** y tiene un nombre descriptivo de la tarea.

3. Implementación de Cambios:

- El desarrollador realiza los cambios necesarios en la rama específica de la tarea.

4. Pruebas Locales:

- Antes de solicitar la fusión, el desarrollador realiza pruebas locales para asegurarse de que los cambios no introduzcan nuevos problemas.

5. Creación de la Solicitud de Cambios:

- El desarrollador crea una Solicitud de Cambios (CR) en la plataforma de gestión de código (por ejemplo, GitHub). La solicitud incluye información detallada sobre la tarea, los cambios realizados y las pruebas realizadas.

6. Revisión por Pares:

- Otro miembro del equipo revisa los cambios propuestos en la Solicitud de Cambios.
- Se proporcionan comentarios, sugerencias o aprobación según sea necesario.

7. Pruebas de Integración:

- Después de la revisión por pares, se realizan pruebas de integración en un entorno de desarrollo compartido para asegurar que los cambios no afecten negativamente a otras partes del código.

8. Aprobación y Fusión:

- Una vez que la Solicitud de Cambios ha sido revisada y aprobada, se fusiona con la rama **main**.
- La Solicitud de Cambios debe ser aprobada por al menos un revisor antes de la fusión.

9. Eliminación de la Rama de Tarea:

- Después de la fusión exitosa, la rama específica de la tarea se elimina para mantener una estructura de ramificación limpia.

Este proceso asegura que todos los cambios en el código sean revisados y probados antes de integrarse con la rama principal. Facilita la colaboración entre los miembros del equipo y ayuda a mantener la calidad y estabilidad del código a lo largo del tiempo.

6.4.2 Control de versiones de documentación

La documentación es una parte fundamental de nuestro proyecto y, al igual que el código, debe gestionarse de manera estructurada. Aquí se describe cómo se gestionan las versiones de la documentación relacionada con el proyecto del juego UNO:

1. Formato de Documentación:

- La documentación se mantiene en formatos accesibles y colaborativos, o un formato compatible con la plataforma de gestión de documentos.

2. Ubicación del Repositorio:

- La documentación se almacena en un directorio específico dentro del repositorio del proyecto, junto con el código fuente.

3. Sincronización con Versiones del Código:

- La documentación se sincroniza con las versiones del código. Por ejemplo, cuando se realiza una fusión en la rama **main**, también se actualiza la documentación relevante para reflejar los cambios introducidos.

4. Revisión y Actualización Continua:

- La documentación se revisa y actualiza de manera continua para reflejar con precisión el estado actual del proyecto.
- Las actualizaciones pueden incluir nuevas características, cambios en el flujo de trabajo, requisitos del sistema, entre otros.

5. Compatibilidad con Versiones Anteriores:

- Se garantiza la disponibilidad de la documentación correspondiente a versiones anteriores del código. Esto es especialmente útil para mantener la documentación coherente con las versiones específicas del juego.

6. Indicación de la Versión en la Documentación:

- La documentación incluye una indicación clara de la versión del proyecto a la que hace referencia. Esto permite a los usuarios y desarrolladores acceder a la documentación correspondiente a una versión específica.

7. Revisión por Pares:

- Al igual que el código, la documentación pasa por revisiones por pares para garantizar precisión y claridad.

6.5 Auditoria y seguimiento

Registro de cambios

El registro de cambios es esencial para rastrear y documentar la evolución del código fuente a lo largo del tiempo. En nuestro proyecto del juego UNO, utilizamos un sistema de registro estructurado que incluye los siguientes pasos:

1. **Commits Significativos:**

- Cada cambio en el código se realiza a través de commits significativos y bien documentados.
- Cada commit describe claramente la naturaleza del cambio y su propósito.

2. **Referencias a Problemas o Tareas:**

- Si el cambio está relacionado con una tarea específica o resuelve un problema conocido, se hace referencia a la tarea correspondiente en el sistema de seguimiento de problemas.

3. **Inclusión de Cambios en Solicitudes de Fusión:**

- Cada cambio propuesto a la rama principal (main) se realiza a través de una Solicitud de Fusión (Pull Request) en GitHub o la plataforma de gestión de código utilizada.
- La Solicitud de Fusión incluye una descripción de los cambios y enlaces a los commits relacionados.

4. **Historial de Commits en Repositorio:**

- El historial completo de commits está disponible en el repositorio, permitiendo a los desarrolladores y colaboradores revisar y comprender la evolución del código.

5. **Uso de Etiquetas para Versiones:**

- Se utilizan etiquetas específicas para marcar versiones importantes del juego en el historial de commits.

6. **Registro de Cambios en la Documentación:**

- Se asegura de que los cambios realizados en el código también se reflejen en la documentación correspondiente.

Este proceso de registro proporciona una visión clara y detallada de la evolución del código, facilitando la identificación de cambios específicos, la revisión de historiales y la comprensión de la progresión del proyecto a lo largo del tiempo.

6.6 Procedimientos de respuesta a problemas

6.6.1 Resolución de conflictos

En el desarrollo colaborativo, los conflictos de fusión pueden surgir cuando dos o más ramas tienen cambios en las mismas líneas del código. Aquí se presenta un proceso para resolver conflictos de fusión:

1. Identificación de Conflictos:

- Antes de fusionar una rama con la principal (**main**), se realiza una revisión para identificar posibles conflictos.
- El equipo utiliza comandos como **git status** y **git diff** para revisar las diferencias entre las ramas.

2. Fusión Local y Resolución:

- Si se identifican conflictos, el desarrollador realiza la fusión local en su entorno de desarrollo.
- Se aborda cada conflicto individualmente, utilizando herramientas como **git mergetool** o editando manualmente los archivos.

3. Resolución Colaborativa:

- En el caso de conflictos más complejos, se pueden discutir y resolver de manera colaborativa con otros desarrolladores involucrados.
- Las discusiones sobre la resolución de conflictos se documentan en la plataforma de gestión de código.

4. Pruebas Locales:

- Después de resolver los conflictos, se realizan pruebas locales para asegurarse de que la fusión no haya introducido nuevos problemas.

5. Actualización de la Solicitud de Fusión:

- Si la fusión es parte de una Solicitud de Fusión (Pull Request), se actualiza la Solicitud de Fusión con información sobre cómo se resolvieron los conflictos y se solicita una revisión adicional.

6. Revisión por Pares:

- Otra revisión por pares se realiza después de resolver los conflictos para garantizar la calidad del código y la resolución adecuada de los problemas.

7. Fusión en la Rama Principal (main):

- Después de resolver los conflictos y recibir aprobación, la fusión se realiza en la rama principal (**main**).

8. Eliminación de Ramas Temporales:

- Las ramas temporales utilizadas para la resolución de conflictos se eliminan después de la fusión exitosa.

9. Registro de Conflictos y Resoluciones:

- Se mantiene un registro de los conflictos y las resoluciones realizadas para facilitar la identificación de patrones y mejorar prácticas futuras.

Este proceso asegura una resolución efectiva y colaborativa de conflictos durante el proceso de fusión, minimizando el impacto en el desarrollo y manteniendo la integridad del código.

6.6.2 Recuperación ante fallos

La recuperación ante fallos es esencial para garantizar la estabilidad y continuidad en el desarrollo del juego UNO. Aquí se describe cómo manejamos situaciones de falla del sistema y cómo nos recuperamos:

1. Monitoreo Continuo:

- Se implementa para identificar rápidamente cualquier anomalía en el sistema, ya sea en el rendimiento, la disponibilidad o la integridad de los datos.

2. Registro Detallado de Errores:

- Cada aplicación y componente del sistema registra detalladamente los errores y excepciones que puedan ocurrir.

3. Alertas Automáticas:

- Configuramos alertas automáticas que notifican al equipo de desarrollo ante la detección de eventos o errores críticos.

4. Respuesta Inmediata:

- Ante una alerta o un registro de error, el equipo de desarrollo responde de inmediato para evaluar la situación y determinar la causa raíz.

5. Análisis Post-Falla:

- Después de resolver la situación de fallo, realizamos un análisis post-falla para entender las causas del problema y tomar medidas preventivas.
- Esto puede incluir ajustes en el código, mejoras en la infraestructura o la implementación de medidas de seguridad adicionales.

6. Plan de Continuidad del Negocio:

- Mantenemos un plan de continuidad del negocio que detalla los pasos específicos a seguir en caso de una falla que afecte la operación normal del juego UNO.

7. Mejoras Continuas:

- Utilizamos la información recopilada de eventos de falla para realizar mejoras continuas en el sistema y prevenir problemas similares en el futuro.

Este enfoque integral asegura una rápida identificación y respuesta a situaciones de falla, minimizando el impacto en el desarrollo y garantizando una experiencia de desarrollo continua y robusta para el equipo.

6.7 Conclusiones

Lecciones aprendidas

Durante el desarrollo y la gestión de configuración del juego UNO, hemos adquirido valiosas lecciones que han contribuido a mejorar nuestro enfoque y eficiencia. Algunas lecciones clave incluyen:

1. **Claridad en Convenciones:**

- Establecer convenciones de nomenclatura y ramificación claras ha mejorado la comunicación y la colaboración entre los miembros del equipo. La consistencia en la nomenclatura facilita la identificación y comprensión de las versiones y las ramas.

2. **Revisión de Código:**

- La implementación de revisiones de código, incluyendo revisiones por pares y pruebas exhaustivas, ha demostrado ser esencial para la calidad del código y la identificación temprana de problemas.

3. **Registro Detallado de Cambios:**

- Mantener un registro detallado de cambios en el código y la documentación ha sido crucial para la identificación de problemas. Los mensajes de commit claros y descriptivos son esenciales para comprender el propósito de cada cambio.

4. **Respuesta Rápida a Fallas:**

- La implementación de un sistema de monitoreo continuo y la capacidad de respuesta inmediata ante fallas han minimizado el impacto de problemas en la producción y han mejorado la estabilidad del sistema.

5. **Formación Continua:**

- La formación continua sobre nuevas tecnologías, herramientas y prácticas de desarrollo ha mantenido al equipo actualizado y ha mejorado la capacidad de abordar desafíos complejos.

6. **Colaboración Efectiva:**

- Fomentar la colaboración efectiva entre los miembros del equipo, especialmente durante la resolución de conflictos y la implementación de nuevas características, ha mejorado la eficiencia y la calidad del trabajo.

7 Documento de estimación del proyecto

7.1 Objetivos

El objetivo principal del proyecto es desarrollar una versión escalable del juego "UNO". Este proyecto tiene una visión evolutiva, comenzando con la implementación de una versión para un jugador, seguida de la introducción de una versión remota para un solo jugador y, finalmente, culminando con el desarrollo de una versión multijugador remota. Cada fase tiene como propósito proporcionar una experiencia progresivamente más completa y atractiva para los jugadores.

7.2 Alcance

El proyecto se enfoca en la creación de una aplicación de software que permita a los jugadores disfrutar del juego UNO en un entorno digital. Se busca replicar las reglas y mecánicas del juego de cartas tradicional, asegurando una experiencia auténtica para los usuarios. Además, se implementarán funciones para partidas individuales y multijugador en línea, y se prestará especial atención a la creación de una interfaz gráfica intuitiva para mejorar la experiencia del usuario.

7.3 Limitaciones

7.3.1 Restricciones de plataforma

El juego se desarrollará exclusivamente para plataformas de ordenadores. La decisión de no incluir soporte para dispositivos móviles se basa en recursos limitados y en la necesidad de enfocarse en una experiencia de usuario sólida en una plataforma específica.

7.3.2 Limitaciones de tiempo y recursos

Se reconoce que existen limitaciones de tiempo y recursos que podrían influir en la implementación de características adicionales o en la optimización del sistema. Es importante gestionar de manera eficiente estos recursos para cumplir con los plazos y objetivos del proyecto.

7.4 Estimación de recursos y tiempo

7.4.1 Equipo de desarrollo

El equipo de desarrollo se conformará de la siguiente manera:

- Desarrolladores Frontend: 2
- Desarrolladores Backend: 2
- Diseñadores de Interfaz de Usuario (UI): 2
- Evaluadores y Pruebas: 2

La dedicación estimada por semana por cada miembro del equipo será de 10 horas.

7.4.2 Duración del proyecto

La estimación de la duración completa del proyecto se basa en la complejidad de las fases y características planificadas. Se prevé que el desarrollo tomará aproximadamente 3 meses.

7.4.3 Recursos técnicos

Para la implementación del juego y la plataforma en línea, se utilizarán las siguientes tecnologías:

- **Frontend:**

Tecnologías: Java Swing

Descripción: Utilizaremos Java Swing para el desarrollo del frontend, aprovechando su capacidad para crear interfaces gráficas de usuario en entornos de escritorio. El diseño de las ventanas, paneles y la interacción con el usuario se gestionarán mediante las bibliotecas de Java Swing.

- **Backend:**

Tecnologías: Java con NetBeans

Descripción: La lógica del juego y la gestión de datos estarán basadas en Java, implementadas en el entorno de desarrollo NetBeans. Esto garantizará una integración fluida con el frontend y permitirá un desarrollo coherente en todo el proyecto.

Esta elección de tecnologías se basa en la experiencia del equipo con Java Swing y NetBeans, asegurando una implementación coherente y eficiente en el desarrollo del juego UNO.

7.5 Descomposición de tareas

7.5.1 Desarrollo de la versión para un jugador

Implementación de Reglas del Juego:

- Detalles: Incluye la programación de reglas específicas de UNO.
- Estimación de Tiempo: 3 semanas

Desarrollo de la Interfaz Gráfica:

- Detalles: Creación de la interfaz de usuario para la versión de un jugador.
- Estimación de Tiempo: 1 semana

Pruebas y Correcciones:

- Detalles: Realización de pruebas exhaustivas y corrección de errores.
- Estimación de Tiempo: 1 semana

7.5.2 Desarrollo de la versión remota para un jugador

Implementación de Conexión Remota:

- Detalles: Integración de funciones para permitir partidas en línea para un solo jugador.
- Estimación de Tiempo: 1 semana

Integración de Partidas en Línea:

- Detalles: Desarrollo de la lógica para el juego en línea.
- Estimación de Tiempo: 2 semanas

Pruebas y Correcciones:

- Detalles: Realización de pruebas específicas para la versión remota y corrección de errores.
- Estimación de Tiempo: 1 semana

7.5.3 Desarrollo de la versión multijugador remoto

Escalabilidad de la Plataforma:

- Detalles: Asegurar que la plataforma pueda manejar múltiples jugadores simultáneamente.
- Estimación de Tiempo: 1 semana

Implementación de Funcionalidades Multijugador:

- Detalles: Desarrollo de funciones de juego interactivo entre múltiples jugadores.
- Estimación de Tiempo: 1 semana

Pruebas y Correcciones:

- Detalles: Pruebas intensivas para garantizar la estabilidad del juego multijugador y corrección de cualquier error identificado.
- Estimación de Tiempo: 1 semana

8 Implementacion de SCRUM

8.1 Roles en Scrum

- **Product Owner:**
 - Santiago: Responsable de definir y priorizar los elementos del Product Backlog. Se asegurará de que el equipo trabaje en las características más valiosas para el proyecto.
- **Scrum Master:**
 - Lautaro: Facilitador del proceso SCRUM. Ayudará al equipo a comprender y seguir las prácticas SCRUM, eliminando obstáculos y promoviendo un entorno de desarrollo ágil.
- **Development Team:**
 - Lautaro y Santiago: Responsables de la implementación de las historias de usuario y la entrega de incrementos funcionales en cada sprint.

8.2 Artefactos en Scrum

- **Product Backlog:**
 - <https://seguzquiza.atlassian.net/jira/software/projects/OBLI/boards/2/backlog>
Lista priorizada de todas las características, historias de usuario y tareas que deben realizarse en el proyecto.
- **Sprint Backlog:**
 - <https://seguzquiza.atlassian.net/jira/software/projects/OBLI/boards/2>
Lista de tareas seleccionadas del Product Backlog para el sprint actual.
- **Incremento del Producto:**
 - https://github.com/SantiagoEguzquiza/UNO_GAME
Versión funcional del producto al final de cada sprint.

8.3 Eventos en Scrum

- **Sprint Planning:**
 - **Frecuencia:** Al comienzo de cada sprint.
 - **Objetivo:** Planificación detallada de las tareas que se abordarán durante el próximo sprint.
- **Daily Scrum:**
 - **Frecuencia:** Diario.
 - **Objetivo:** Reunión diaria de 15 minutos para sincronizar al equipo, compartir actualizaciones y abordar posibles obstáculos.
- **Sprint Review:**
 - **Frecuencia:** Al final de cada sprint.
 - **Objetivo:** Revisión de las historias de usuario completadas y demostración del incremento del producto.
- **Sprint Retrospective:**
 - **Frecuencia:** Al final de cada sprint, después de la Sprint Review.
 - **Objetivo:** Reflexión sobre el sprint pasado, identificación de mejoras y planificación de ajustes para el próximo sprint.

8.4 Documento de Scrum

Sprint 1: 18/10/2023 – 25/10/2023

- **Sprint Planning:** 18/10/2023
 - Definición de objetivos y selección de tareas del Product Backlog.
- **Daily Scrum:** 21, 22, y 23/10/2023
 - Actualizaciones diarias y discusión de posibles impedimentos.
- **Sprint Review:** 25/10/2023
 - Demostración del trabajo completado durante el sprint.
- **Sprint Retrospective:** 25/10/2023
 - Reflexión sobre el sprint y planificación de mejoras.

Sprint 2: 26/10/2023 – 9/11/2023

- **Sprint Planning: 26/10/2023**
 - Definición de objetivos y selección de tareas del Product Backlog.
- **Daily Scrum: 28 y 29/10/2023, 4 y 5/11/2023**
 - Actualizaciones diarias y discusión de posibles impedimentos.
- **Sprint Review: 9/11/2023**
 - Demostración del trabajo completado durante el sprint.
- **Sprint Retrospective: 9/11/2023**
 - Reflexión sobre el sprint y planificación de mejoras.

Sprint 3: 10/11/2023 – 24/11/2023

- **Sprint Planning: 10/11/2023**
 - Definición de objetivos y selección de tareas del Product Backlog.
- **Daily Scrum: 11, 12, 18 y 19/11/2023**
 - Actualizaciones diarias y discusión de posibles impedimentos.
- **Sprint Review: 24/11/2023**
 - Demostración del trabajo completado durante el sprint.
- **Sprint Retrospective: 24/11/2023**
 - Reflexión sobre el sprint y planificación de mejoras.

Sprint 4: 25/11/2023 – 9/12/2023

- **Sprint Planning: 25/11/2023**
 - Definición de objetivos y selección de tareas del Product Backlog.
- **Daily Scrum: 25 y 26/11/2023, 2, 3, 5, 6, 7 y 8/12/2023**
 - Actualizaciones diarias y discusión de posibles impedimentos.
- **Sprint Review: 8/12/2023**
 - Demostración del trabajo completado durante el sprint.
- **Sprint Retrospective: 8/12/2023**
 - Reflexión sobre el sprint y planificación de mejoras.

9 Documento de Pruebas del Sistema

9.1 Objetivos

El propósito de este documento es describir los procedimientos y resultados de las pruebas realizadas en el proyecto UNO desarrollado en Java.

9.2 Alcance

Las pruebas se centran en garantizar la funcionalidad, usabilidad y rendimiento del juego UNO.

9.3 Casos de prueba

Caso de Prueba 1: Iniciar Partida

Descripción: Verificar que el juego permite iniciar una nueva partida correctamente.

1. Iniciar la aplicación UNO.
2. Seleccionar "Iniciar Nueva Partida".
3. Configurar el número de jugadores.
4. Confirmar la configuración.
5. Verificar que se ha iniciado la partida.

Resultado esperado: Se puede comenzar el juego.

Resultado obtenido: La partida se inicia sin errores.

Caso de Prueba 2: Número de Jugadores Inválido

Descripción: Verificar que el sistema maneja correctamente un número de jugadores fuera del rango permitido.

1. Iniciar la aplicación UNO.
2. Seleccionar "Iniciar Nueva Partida".
3. Configurar el número de jugadores con un valor inválido.
4. Intentar confirmar la configuración.

Resultado esperado: El sistema muestra un mensaje de error y no permite iniciar la partida.

Resultado obtenido: El sistema detecta valores fuera del rango y muestra un mensaje de error.

Caso de Prueba 3: Distribución de Cartas Exitosa

Descripción: Verificar que el sistema distribuye correctamente 7 cartas a cada jugador al comienzo de una partida.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Verificar que se han distribuido 7 cartas a cada jugador.

Resultado esperado: El sistema distribuye 7 cartas a cada jugador sin errores.

Resultado obtenido: El sistema verifica la cantidad de jugadores, prepara un mazo completo de cartas UNO, lo baraja y reparte 7 cartas a cada jugador. Las cartas repartidas se retiran del mazo principal, y la carta de inicio se coloca en el centro de la mesa. El juego está listo para comenzar.

Caso de Prueba 4: Insuficientes Cartas en el Mazo

Descripción: Verificar que el sistema toma medidas adecuadas si no hay suficientes cartas en el mazo para repartir 7 cartas a cada jugador.

1. Iniciar la aplicación UNO.
2. Configurar el mazo para que contenga menos de 7 cartas.
3. Iniciar una nueva partida.

Resultado esperado: El sistema rebaraja el mazo antes de intentar repartir las cartas.

Resultado obtenido: El sistema detecta la insuficiencia de cartas en el mazo, toma medidas adecuadas (como rebarajar el mazo), y luego reparte 7 cartas a cada jugador.

Caso de Prueba 5: Formación de Pila de Descarte Exitosa

Descripción: Verificar que el sistema forma correctamente la pila de descarte al comienzo de la partida de UNO.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Verificar que la pila de descarte se ha formado correctamente.

Resultado esperado: El sistema forma la pila de descarte con la carta superior del mazo sin errores.

Resultado obtenido: El sistema toma la carta superior del mazo, la voltea y la coloca en el centro de la mesa, formando así la pila de descarte. El juego está listo para comenzar a partir de la carta de inicio en la pila de descarte.

Caso de Prueba 6: Mazo Vacío al Intentar Formar Pila de Descarte

Descripción: Verificar que el sistema maneja adecuadamente la situación cuando el mazo está vacío al intentar formar la pila de descarte.

1. Iniciar la aplicación UNO.
2. Configurar el mazo para que esté vacío.
3. Iniciar una nueva partida.

Resultado esperado: El sistema toma medidas adecuadas para manejar un mazo vacío, como rebarajar el mazo antes de formar la pila de descarte.

Resultado obtenido: El sistema detecta que el mazo está vacío, toma medidas adecuadas (como rebarajar el mazo), y luego forma la pila de descarte con la carta superior del mazo.

Caso de Prueba 7: Jugar Carta Válida Exitosamente

Descripción: Verificar que un jugador puede jugar una carta válida según las reglas del juego UNO.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Asegurarse de que el jugador tenga una carta en la mano que coincida en número o color con la carta superior de la pila de descarte.
4. El jugador selecciona la carta de su mano.
5. Verificar que la carta se ha colocado correctamente en la pila de descarte y que la carta superior se ha actualizado.

Resultado esperado: El jugador puede jugar una carta válida, y la carta seleccionada se coloca en la pila de descarte. La carta superior de la pila de descarte se actualiza correctamente.

Resultado obtenido: El sistema muestra las cartas en la mano del jugador y la carta superior de la pila de descarte. El jugador selecciona una carta que coincide en número o color con la carta superior de la pila de descarte. La carta se coloca en la pila de descarte, y la carta superior se actualiza correctamente.

Caso de Prueba 8: Jugar Carta No Válida - Mensaje de Error

Descripción: Verificar que el sistema muestra un mensaje de error si un jugador intenta jugar una carta que no es válida.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Configurar la mano del jugador para que no tenga cartas válidas para jugar.
4. El jugador intenta seleccionar una carta inválida.
5. Verificar que se muestra un mensaje de error.

Resultado esperado: El sistema muestra un mensaje de error indicando que la carta seleccionada no es válida.

Resultado obtenido: El jugador intenta seleccionar una carta que no es válida, y el sistema muestra un mensaje de error adecuado.

Caso de Prueba 9: Robar Carta Exitosamente

Descripción: Verificar que un jugador puede robar una carta del mazo cuando no puede jugar una carta válida en su turno.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. El jugador selecciona la opción "Robar Carta".
4. Verificar que el jugador ha robado una carta del mazo y que la carta se ha agregado a su mano.

Resultado esperado: El jugador puede robar una carta del mazo con éxito.

Resultado obtenido: El jugador selecciona la opción "Robar Carta", y el sistema toma una carta del mazo, la agrega a la mano del jugador y muestra la carta robada.

Caso de Prueba 10: Intento de Robar Carta con Mazo Vacío

Descripción: Verificar que el sistema maneja adecuadamente la situación cuando el jugador intenta robar una carta, pero el mazo está vacío.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Configurar el mazo para que esté vacío.
4. El jugador selecciona la opción "Robar Carta".

Resultado esperado: El sistema muestra un mensaje indicando que el mazo está vacío y no permite al jugador robar una carta.

Resultado obtenido: El sistema detecta que el mazo está vacío y muestra un mensaje indicando que el jugador no puede robar una carta porque el mazo está vacío.

Caso de Prueba 11: Cambio de Color Exitoso

Descripción: Verificar que un jugador puede cambiar el color de juego correctamente al jugar una carta especial.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Asegurarse de que el jugador actual tenga una carta especial que permita cambiar el color de juego.
4. El jugador selecciona la carta especial.
5. El sistema solicita al jugador elegir un nuevo color para el juego.
6. El jugador elige un nuevo color.
7. Verificar que el color de juego se ha cambiado según la elección del jugador.

Resultado esperado: El jugador puede cambiar el color de juego correctamente al jugar una carta especial.

Resultado obtenido: El sistema muestra las cartas en la mano del jugador actual. El jugador selecciona una carta especial que permite cambiar el color de juego. El sistema solicita al jugador elegir un nuevo color para el juego, y el jugador elige con éxito. El sistema actualiza el color de juego, y la partida continúa con el siguiente jugador.

Caso de Prueba 12: Cambio de Sentido Exitoso

Descripción: Verificar que un jugador puede cambiar el sentido del juego correctamente al jugar una carta especial.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Asegurarse de que el jugador actual tenga una carta especial que permita cambiar el sentido del juego.
4. El jugador selecciona la carta especial.
5. Verificar que el sentido del juego se ha cambiado correctamente.

Resultado esperado: El jugador puede cambiar el sentido del juego correctamente al jugar una carta especial.

Resultado obtenido: El sistema muestra las cartas en la mano del jugador actual. El jugador selecciona una carta especial que permite cambiar el sentido del juego. El sistema cambia el sentido del juego según corresponda, y la partida continúa con el siguiente jugador en el nuevo sentido del juego.

Caso de Prueba 13: Salto de Turno Exitoso

Descripción: Verificar que un jugador puede saltar su turno correctamente al jugar una carta especial.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Asegurarse de que el jugador actual tenga una carta especial que permita saltar el turno.
4. El jugador selecciona la carta especial.
5. Verificar que el turno del jugador actual se ha saltado correctamente.

Resultado esperado: El jugador puede saltar su turno correctamente al jugar una carta especial.

Resultado obtenido: El sistema muestra las cartas en la mano del jugador actual. El jugador selecciona una carta especial que permite saltar el turno. El sistema salta el turno del jugador actual y pasa al siguiente jugador en sentido horario. El juego continúa con el nuevo jugador.

Caso de Prueba 14: Jugada de Carta por la CPU Exitosa

Descripción: Verificar que la CPU puede jugar una carta correctamente durante una partida de UNO.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Configurar la mano de la CPU para que tenga una carta válida para jugar.
4. Es el turno de la CPU para jugar una carta.
5. Verificar que la CPU selecciona y juega una carta correctamente.

Resultado esperado: La CPU puede jugar una carta correctamente durante su turno.

Resultado obtenido: La CPU evalúa las cartas en su mano y selecciona la mejor carta para jugar según las reglas de UNO. La CPU realiza la acción correspondiente según el tipo y el color de la carta, y el sistema actualiza el estado del juego. Se muestra la carta jugada por la CPU en el área de juego, y el turno pasa al siguiente jugador.

Caso de Prueba 15: Jugada de Carta por la CPU sin Carta Válida

Descripción: Verificar que la CPU levanta una carta si no puede jugar ninguna carta válida durante su turno.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Configurar la mano de la CPU para que no tenga cartas válidas para jugar.
4. Es el turno de la CPU para jugar una carta.
5. Verificar que la CPU levanta una carta y su turno termina.

Resultado esperado: La CPU levanta una carta si no puede jugar ninguna carta válida durante su turno.

Resultado obtenido: La CPU evalúa las cartas en su mano y determina que no puede jugar ninguna carta válida. La CPU levanta una carta del mazo.

Caso de Prueba 16: Recogida de Carta Válida

Descripción: Verificar si la carta recogida por la CPU es válida para ser jugada inmediatamente, la CPU la juega y continúa su turno.

1. Iniciar la aplicación UNO.
2. Iniciar una nueva partida.
3. Es el turno de la CPU para jugar una carta.
4. Al intentar jugar, la misma no posee ninguna carta válida.
5. Verificar que la CPU levanta una carta, la juega si es válida, y termina su turno.

Resultado esperado: Si la carta recogida por la CPU es válida para ser jugada inmediatamente, la CPU la juega y termina su turno.

Resultado obtenido: La CPU evalúa las cartas en su mano y determina que no puede jugar ninguna carta válida. La CPU levanta una carta, la juega, y el sistema actualiza el estado del juego. El turno de la CPU termina.

Caso de Prueba 17: Rendimiento General

Descripción: Evaluar el rendimiento general del juego durante una partida.

1. Iniciar una partida.
2. Jugar varias rondas de juego.
3. Observar el rendimiento del juego, incluyendo la respuesta a las acciones de los jugadores y la CPU.

Resultado esperado: El juego mantiene un rendimiento fluido sin retrasos significativos.

Resultado Obtenido: Se cumplió con el resultado esperado. Las acciones de los jugadores y la CPU se ejecutaron sin demoras notables, al igual que las transiciones entre las rondas de juego. Este resultado indica que el juego puede manejar eficientemente las acciones simultáneas de múltiples jugadores y mantener un nivel adecuado de fluidez durante las rondas de juego.