República Bolivariana de Venezuela

Universidad Católica Andrés Bello

Puerto Ordaz, Ciudad Guayana

Escuela de Ingeniería Informática

Algoritmos y Programación III

´PROYECTO – DONKEY KONG

Aray, Jonás

Betancourt, Jesús

Rojas, Cesar

Youssef, Teresa

Profesora. Ing. Bello Jannelly

Puerto Ordaz, mayo 2021

ÍNDICE

INTRODUCCIÓN

El presente proyecto es realizado por los alumnos de la catedra Algoritmos y Programación III. En el que se evaluara las habilidades en cuanto a desarrollo del software utilizando conceptos aprendidos a lo largo del estudio de la catedra como la programación orientada a objetos y la aplicación de modelos de proceso de desarrollo de software.

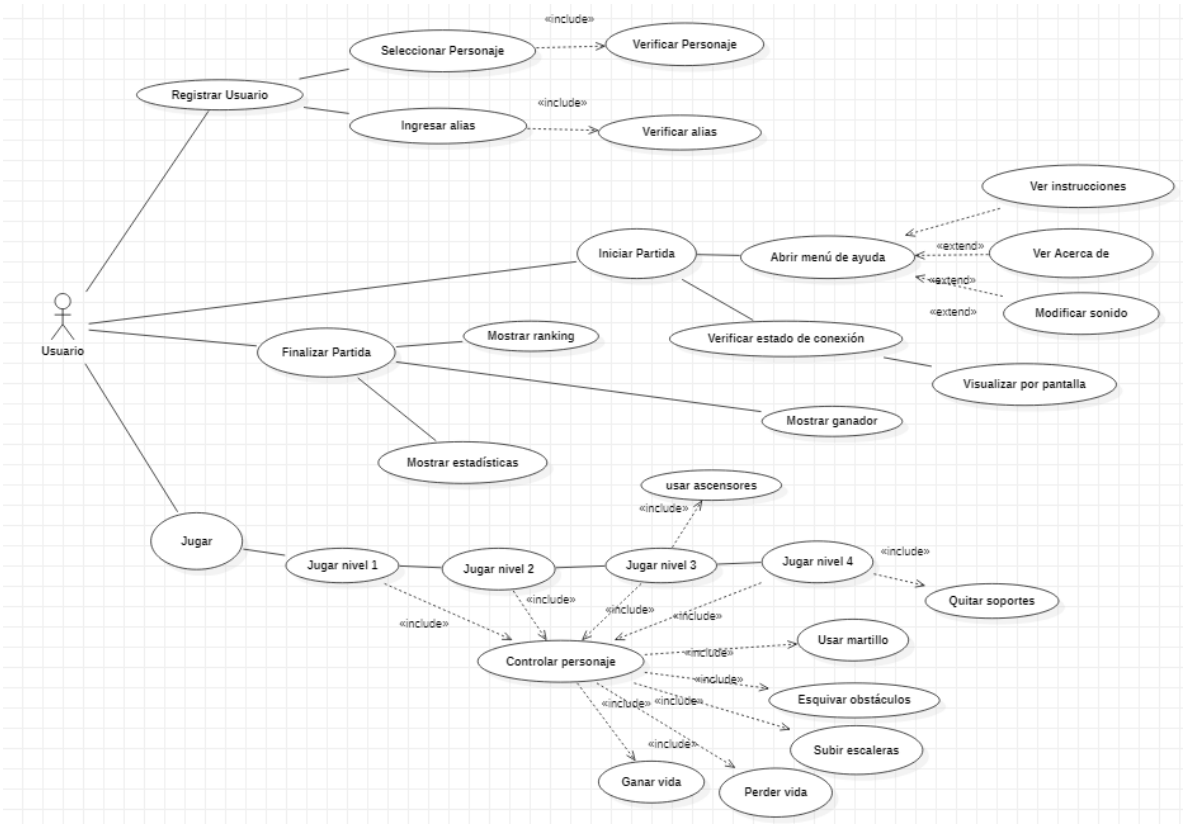
Se trata de una simulación del videojuego Donkey Kong creado por Nintendo en el año 1981. Es un primitivo videojuego del genero plataformas, este tipo de genero se refiere a los que se caracterizan por tener que caminar, correr, saltar o escalar sobre una serie de plataformas y acantilados, con enemigos, mientras se recogen objetos para poder completar el juego. El juego se basa en que Mario debe rescatar a Pauline, la cual fue capturada por Donkey Kong.

El juego consta de diversos niveles en donde a medida que van pasando va aumentando la dificultad, cada nivel representa 25 metros del edificio que Donkey Kong va escalando. El jugador debe recoger, esquivar y destruir objetos, para así poder obtener puntos.

El juego es multijugador, es decir, dos jugadores competirán por rescatar a la dama Pauline. La partida inicia cuando haya dos jugadores conectados desde computadoras diferentes, donde ambos visualizaran en sus pantallas el juego completo. Los jugadores inician la partida con cinco vidas, deben subir escaleras hasta llegar a la cima, y también, como se mencionó anteriormente, deben esquivar los obstáculos móviles, los cuales son lanzados por Donkey Kong y saltar obstáculos fijos, si estos chocan con algún obstáculo, pierden una vida. Si al llegar a la cima estos obtienen 7000 puntos, ganan una vida extra.

CAPITULO I – ANÁLISIS

Diagrama de casos de usos:



Tablas descriptivas de cada caso de uso:

CASO DE USO	Registrar Usuario
ACTORES	Usuario
PROPOSITO	Registrar en la base de datos los datos del usuario.
PRECONDICIONES	- Personaje seleccionado - Alias
FLUJO PRINCIPAL	De ser validos todos los datos ingresados por el usuario, se guardaran en la base de datos y se indicara que todo salio correcto

FLUJO ATERNATIVO	De faltar algún campo o algún dato introducido es invalido se mostrara mensaje de error al usuario y se pedirá que se vuelva a ingresar.

CASO DE USO	Iniciar partida
ACTORES	Usuario
PRÓPOSITO	Iniciar la partida para ambos jugadores.
PREPCONDICIONES	- Estar registrado. - Haber seleccionado un personaje y un alias.
FLUJO PRINCIPAL	Al estar registrado el usuario y ya haber seleccionado el alias y el personaje se inicia la partida
FLUJO SECUNDARIO	Al no estar registrado el usuario se envía mensaje de error y se pedí que se registre. De no haber seleccionado un alias o un personaje se muestra mensaje de error y se pide que lo seleccione.

CASO DE USO	Finalizar partida
ACTORES	Usuario
PRÓPOSITO	Dar por finalizada la partida.
PREOCONDICIONES	Haber iniciado la partida
FLUJO PRINCIPAL	Se finaliza la partida, se muestra el ganar
FLUJO ALTERNATIVO	No hay flujo alternativo.

CASO DE USO	Jugar
ACTORES	Usuario
PRÓPOSITO	Empezar el juego. Cuando el usuario ya va por este caso de uso puede realizar todas

	las siguientes acciones: moverse de un lado a otro, subir escaleras, esquivar obstáculos, golpear obstáculos
PRECONDICIONES	- El usuario debe estar registrado con su alias y personaje. - El usuario debe estar unido a un servidor o partida. - En el servidor deben estar dos jugadores
FLUJO PRINCIPAL	Se verifican el alias cada jugador se le da un martillo y así pueden empezar la partida.
FLUJO ALTERNATIVO	En todo caso que el registro no sea correcto se muestra un mensaje de error al usuario y se regresa a la pantalla principal para que se registre.

Caso de uso	Seleccionar personaje
Actores	Usuario
Propósito	Que el usuario seleccione el personaje que utilizará durante la partida
Precondiciones	Ingresar al juego e iniciar partida
Flujo Principal	De estar disponible el personaje ingresado, este personaje quedará asignado para que el usuario en cuestión pueda utilizarlo
Flujo Alternativo	Si el personaje ya fue seleccionado por otro jugador, se dirá al usuario que el personaje ya no se encuentra disponible para su uso

CASO DE USO	Ingresar alias
ACTORES	Usuario

PROPÓSITO	Registrar un alias o nickname con el que el usuario en cuestión pueda jugar durante la partida, y sirva para que pueda saber sus estadísticas, posición en la partida y posición en el ranking
PRECONDICIONES	- Haber ingresado al juego - Haber seleccionado un personaje disponible
FLUJO PRINCIPAL	De ser ingresado correctamente el alias y no haber sido seleccionado por otro usuario, quedará registrado correctamente en el juego con ese alias
FLUJO ALTERNATIVO	Si el alías ya fue escogido por otro jugador, se avisará al usuario de que el alias no está disponible y se le dará la opción para que escoja otro alias

CASO DE USO	Abrir menú ayuda
ACTORES	Usuario
PROPÓSITO	Que el jugador pueda acceder a través de un menú a las instrucciones del juego ayudándolo a su mejor comprensión y también pueda acceder a la información general del juego
PRECONDICIONES	Haber ingresado al juego correctamente con un alias válido y un personaje válido
FLUJO PRINCIPAL	Se ingresa al menú de ayuda y el jugador podrá escoger si desea visualizar las instrucciones del juego o la información general del juego
FLUJO ALTERNATIVO	En caso de no poder acceder al menú de ayuda, mostrar un mensaje de error explicando el motivo al usuario

CASO DE USO	Ver instrucciones
ACTORES	usuario
PROPÓSITO	Que el usuario en cuestión pueda acceder a las instrucciones y explicación del propósito

	general del juego para aclarar las dudas que tenía al respecto
PRECONDICIONES	- Haber accedido correctamente al juego con un alias y un personaje válido - Haber ingresado correctamente en el menú de ayuda
FLUJO PRINCIPAL	El jugador puede acceder correctamente a las instrucciones y leer sobre la funcionalidad y modalidad del juego para aclarar dudas al respecto
FLUJO ALTERNATIVO	Se muestra un mensaje de error al usuario indicando que no se pudo acceder correctamente a las instrucciones junto con sus motivos

CASO DE USO	Ver acerca de
ACTORES	Usuario
PROPÓSITO	Que el usuario en cuestión pueda acceder a la información general del juego para que pueda conocer cuáles son sus desarrolladores, el lenguaje de programación utilizado, las librerías utilizadas y la versión actualizada del juego
PRECONDICIONES	- Haber ingresado correctamente en la partida con un alias y un personaje válidos - Haber accedido correctamente al menú de opciones
FLUJO PRINCIPAL	El usuario puede acceder al apartado “Acerca de” del menú de opciones y puede visualizar la información general del juego
FLUJO ALTERNATIVO	Se muestra un mensaje de error al usuario indicando que no se pudo acceder correctamente al apartado de “acerca de” para que intente con realizar otra acción.

CASO DE USO	Modificar sonido
ACTORES	Usuario

PROPÓSITO	Que el usuario en cuestión pueda configurar el sonido del juego (activarlo o desactivarlo)
PRECONDICIONES	- Haber accedido correctamente a la partida (alias y personaje válido) - Haber accedido correctamente al menú de opciones - Tener el volumen del dispositivo activado
FLUJO PRINCIPAL	El usuario podrá configurar el sonido del juego correctamente indicando si desea activarlo o desactivarlo
FLUJO ALTERNATIVO	En caso de no poder acceder correctamente a las opciones de sonido se indica a través de un mensaje de error al usuario y se regresa a la partida en cuestión

CASO DE USO	Verificar estado de conexión
ACTORES	Usuarios
PROPÓSITO	Verificar que dos jugadores estén conectados desde computadoras diferentes para poder dar inicio al juego
PRECONDICIONES	- Que los dos jugadores hayan podido acceder correctamente al juego (alias y personajes validos) - Que cada jugador tenga acceso al juego desde su pc - Que los jugadores tengan acceso para conectarse de forma multijugador al juego
FLUJO PRINCIPAL	Los jugadores pueden conectarse al juego desde dispositivos diferente y pueden dar inicio correctamente a la partida, visualizando por pantalla el juego completo, incluyendo el desempeño de su oponente
FLUJO ALTERNATIVO	En caso de no poder acceder desde computadoras diferentes al juego se lanzará un error por pantalla indicando el error de conexión a los dos usuarios

CASO DE USO	Visualizar por pantalla
ACTORES	Usuario
PROPÓSITO	Que el jugador pueda observar correctamente por pantalla el desarrollo del juego y también el desempeño de su contrincante
PRECONDICIONES	- Haber accedido correctamente al juego - Haberse conectado correctamente al juego de forma multijugador
FLUJO PRINCIPAL	El usuario en cuestión puede visualizar correctamente por pantalla el desarrollo del juego, junto con el desempeño de su oponente
FLUJO ALTERNATIVO	En caso de no poder visualizarse correctamente por pantalla el juego o el desempeño de su oponente indicar el problema a través de un mensaje de error a ambos jugadores

CASO DE USO	Mostrar ganador
ACTORES	Usuario
PROPÓSITO	Una vez finalizada la partida mostrar por un mensaje el ganador de la misma y incrementarle a ese jugador en 1 la cantidad de partidas ganadas
PRECONDICIONES	- Haber finalizado correctamente la partida - Que uno de los jugadores haya cumplido el objetivo en el tiempo previsto del nivel
FLUJO PRINCIPAL	Mostrar al jugador ganador de la partida por pantalla junto con un mensaje de felicitaciones
FLUJO ALTERNATIVO	Indicar a través de un mensaje de error que no se pudo mostrar al ganador de la partida ya que ninguno de los jugadores cumplió con los requisitos establecidos para el nivel

CASO DE USO	Mostrar ranking
ACTORES	Usuario
PROPÓSITO	Visualizar por pantalla el ranking de los jugadores ordenados por el puntaje obtenido durante la partida
PRECONDICIONES	- Haber finalizado correctamente la partida
FLUJO PRINCIPAL	En caso de haber finalizado exitosamente la partida ya sea porque todos los jugadores perdieron o porque alguno de ellos logró ganar mostrar el ranking colocando los alias de los jugadores ordenados de acuerdo a su puntaje obtenido
FLUJO ALTERNATIVO	En caso de una excepción que impida mostrar el ranking de los jugadores mostrar a través de un mensaje de error a los usuarios

CASO DE USO	Mostrar estadísticas
ACTORES	Usuarios
PROPÓSITO	Una vez finalizada la partida mostrar para cada jugador sus estadísticas: partidas jugadas, partidas ganadas y partidas perdidas
PRECONDICIONES	- Haber jugado y finalizado correctamente el juego - Que los datos de las partidas del alías del usuario ya estén registrados en la base de datos
FLUJO PRINCIPAL	Una vez finalizada correctamente la partida mostrar las estadísticas de cada jugador con los datos registrados de cada alias en el sistema
FLUJO ALTERNATIVO	En caso de no poder mostrar las estadísticas del jugador por motivos como que no se encontraron sus registros de partidas indicar por un mensaje de error al usuario

CASO DE USO	Jugar nivel 1
ACTORES	Usuario
PROPÓSITO	Jugar el nivel 1 del juego escalando una zona de construcción formada por vigas torcidas, mientras esquivan saltando los barriles y bolas de fuego de Donkey Kong
PRECONDICIONES	- HABER ACCEDIDO CORRECTAMENTE A LA PARTIDA (ALIAS Y PERSONAJES VÁLIDOS) - HABERSE CONECTADO CORRECTAMENTE EN LA PARTIDA CON EL OTRO JUGADOR
FLUJO PRINCIPAL	Si se realizaron correctamente las precondiciones, acceder para posteriormente jugar el nivel 1 del juego
FLUJO ALTERNATIVO	De no poder jugarse este nivel mostrar a través de un mensaje de error al usuario lo sucedido

CASO DE USO	Jugar nivel 2
ACTORES	Usuario
PROPÓSITO	Que el usuario pueda acceder correctamente y posteriormente jugar el nivel 2 del juego en el tiempo límite, en un nivel donde se debe subir una construcción de 5 pisos, esquivando obstáculos diversos arrojados por Donkey Kong
PRECONDICIONES	- Haber accedido correctamente a la partida (alias y personajes válidos) - Haberse conectado correctamente en la partida con el otro jugador - Haber culminado el nivel1
FLUJO PRINCIPAL	Una vez cumplidos correctamente con las precondiciones acceder para posteriormente jugar el nivel 2 del juego en el tiempo límite establecido

FLUJO ALTERNATIVO	De no poderse acceder a este nivel debido a un error en el sistema o a un incumplimiento en las precondiciones entonces mostrar un mensaje de error al usuario en cuestión

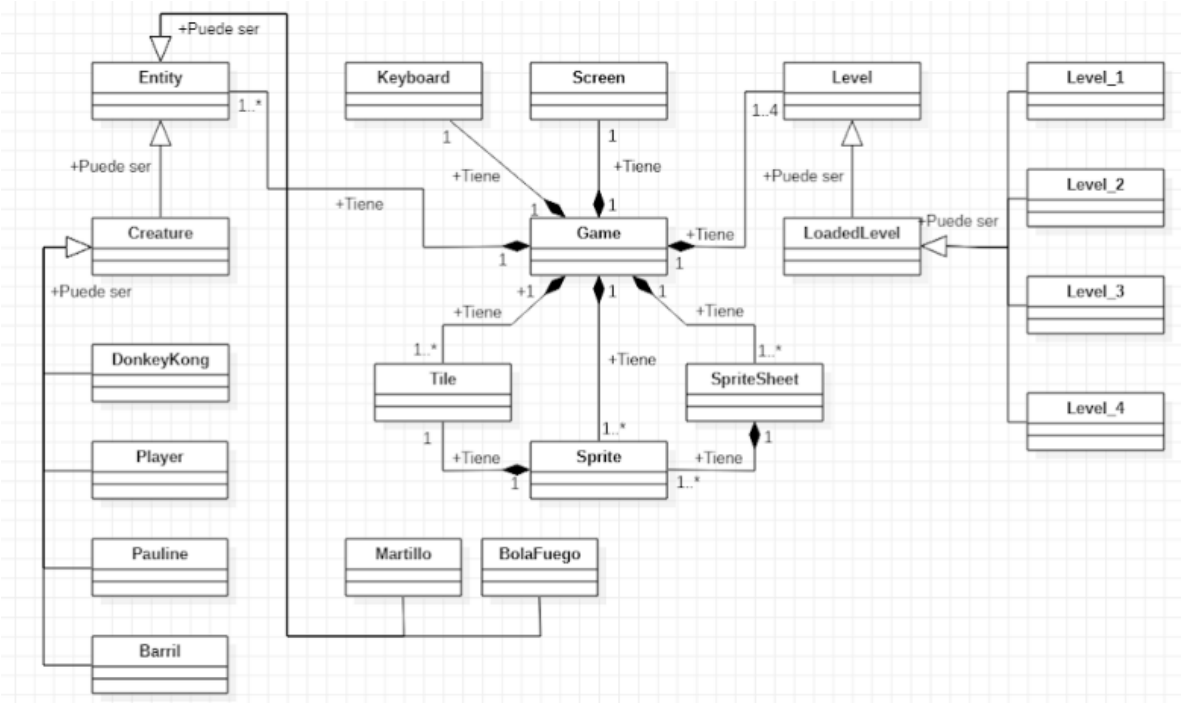
CASO DE USO	Jugar nivel 3
ACTORES	Usuario
PROPÓSITO	Poder acceder y jugar correctamente el nivel 3 del juego, donde los jugadores pueden subir y bajar por ascensores a la vez que evitan los obstáculos arrojados por Donkey Kong
PRECONDICIONES	- Haber accedido correctamente a la partida (alias y personajes válidos) - Haberse conectado correctamente en la partida con el otro jugador - Haber culminado el nivel 2
FLUJO PRINCIPAL	En caso de cumplidas correctamente las precondiciones, acceder para posteriormente jugar el nivel 3 del juego en cuestión
FLUJO ALTERNATIVO	En caso de no poder acceder al nivel 3 por un error en el sistema o el incumplimiento de una de las precondiciones notificar al usuario a través de un mensaje de error

CASO DE USO	Jugar nivel 4
ACTORES	Usuario
PROPÓSITO	Poder acceder para posteriormente jugar el nivel 4 del juego, donde los jugadores deben quitar 8 soportes de 16 que hay en el suelo, ganando de esa forma el nivel el primero que lo logre y en el tiempo límite
PRECONDICIONES	- Haber accedido correctamente a la partida (alias y personajes válidos) - Haberse conectado correctamente en la partida con el otro jugador - Haber culminado el nivel 3

FLUJO PRINCIPAL	Si son cumplidas todas las precondiciones entonces acceder al nivel 4 del juego para posteriormente jugarlo
FLUJO ALTERNATIVO	En caso de no poder acceder al nivel 4 por un error en el sistema o por el incumplimiento de alguna de las precondiciones, notificar al usuario a través de un mensaje de error

CASO DE USO	Controlar jugadores
ACTORES	Usuario
PROPÓSITO	Que los jugadores tengan disponibles todo el control de sus personajes manejados pudiendo realizar las acciones características de estos en el juego
PRECONDICIONES	- Haber accedido correctamente al juego (alias y personaje validos) - Haber accedido correctamente a la partida (conexión multijugador) a través de distintos dispositivos conectados
FLUJO PRINCIPAL	En caso delo correcto acceso a la partida, los jugadores podrán hacer con sus personajes las siguientes acciones: usar martillo, esquivar obstáculos, subir escalera, perder vida, ganar vida
FLUJO ALTERNATIVO	En caso de no poder conseguir el control del personaje para uno o varios jugadores en específico por problemas en el incumplimiento de las precondiciones como no haber ingresado correctamente, notificar al usuario en cuestión a través de un error por pantalla

Diagrama de clases:



Tablas descriptivas de cada clase:

CLASE	Game
	Es la base principal del programa. Se crea el canvas para mostrar la ventana, se le asigna el tamaño de sus dimensiones. Aquí está el reloj, que es lo que hace que se actualice el juego cada cierto tiempo, esto a una velocidad que se vea óptima para el usuario, la frecuencia es 60 veces por segundo que se actualiza.
ATRIBUTOS	- int WIDTH: Constante que tiene el valor del ancho de la ventana el cual es 256. - int HEIGHT: Constante que tiene el valor del alto de la ventana el cual es 288 - int SCALE: Factor de re-escalado de la ventana. - String NAME: Nombre de la aplicación. - string counterUPS: Mostrat las actualizaciones por segundo. - string counterFPS: Mostrat las actualizaciones de los fotogramas por segundo. - int ups: Cuentas las actualizaciones por segundo.

	- int fps: Cuenta las actualizaciones de los fotogramas por segundo. - boolean running: Indica el estado del juego. - JFrame frame: El frame en donde se dibujar el juego. - Thread thread: Hilo para que empiece a correr el programa. - Keyboard keyboard: Nos permite llevar los controles del personaje. - Screen screen: Con esto se muestran todos los sprites. - Level level: Carga una imagen desde una ruta especificiada para generar el nivel. - Player player: Mostrar el sprite del jugador y efectuar sus respectivos movimientos. - BufferedImage image: Genera una imagen para ser posteriormente mostrada. - int[] pixels: Rasteriza la imagen y los guarda en un arreglo correspondientemente. - final ImageIcon icon: Muestra el icono de la aplicación.
MÉTODOS	- synchronized void start(): Inicia el hilo para que arranque el juego. - synchronized void stop(): Detiene el hilo para parar el juego. - void tick(): Actualiza el juego. - void render(): Actualiza el frame y también actualiza cada vez las imágenes que se muestran en el mismo. + void run(): Es el reloj que permite llevar la frecuencia en la que se actualiza el juego infinitamente hasta que el usuario quiera parar.

CLASE	SpriteSheet
	Esta se encarga de procesar una hoja de sprites para posteriormente para posteriormente generar los sprites indivicuales.
ATRIBUTOS	- int width: El ancho de la imagen. - int height: El alto de la imagen. - int[] pixels: Arreglo de pixeles donde se guarda cada pixel de la imagen que le estamos pasando.
MÉTODOS	+ int getWidth(): Mostrar el valor del ancho. + int getHeight(): Mostar el valor del alto.

CLASE	Sprite
	Esta clase se encarga de separar la hoja de sprites en sprites individuales.
ATRIBUTOS	-int side: Esto representa el tamaño del lado del sprite. -int x: Coordenadas del sprite en la hoja (fila). -int y: Coordenada del sprite en la hoja (columna). -int[] pixels: Arreglo de pixiles. -SpriteSheet sheet: Hoja de sprites de donde se sacarán los sprites.
MÉTODOS	+ int getSide(): Mostrar el tamaño del lado del sprite. - void loadSprite(int): Recibe un entero, de ser 0 genera el sprite normal, de lo contrario, lo genera invertido. - void normal(): Cargar el sprite normal. - void rotarX(): Carga el sprite invertido. - int[] initPixels(): Genera un arreglo que se usa para el método rotarX y asi invertir el arreglo.

CLASE	Tile
	Esta clase se encarga de generar los tiles que será mostrado en la pantalla del juego.
ATTRIBUTOS	+int x: Coordenadas de los sprites en el cuadrado (fila). +int y: Coordenadas de los sprites en el cuadrado (columna). + Sprite sprite: El sprite que se va a cargar. + int SIDE: Es el valor constante del tamaño del sprite.
MÉTODOS	+ void render(int, int, Screen): Muestra en la clase pantalla el tile.

CLASE	Screen
	Esta clase se encarga de mostrar los tiles tanto del escenarios del nivel, como los del jugador.
ATRIBUTOS	-int width: El ancho de la ventana. -int height: El largo de la ventana. +int[] pixels: Arreglo de pixeles. -int deltaX: La nueva posición de las coordenadas del sprite en la pantalla (fila).

	-int deltaY: La nueva posición de las coordenadas del sprite en la pantalla (columna).
MÉTODOS	+int getWidth(): Mostrar el ancho de la ventana. +getHeight(): Mostrar el largo de la ventana. +void setDelta(int, int): Actualizar los valores de ddeltax y deltay. +void clear(): Limpia la pantalla. +void renderTile(int, int, Tile): Se encarga de mostrar el tile por pantalla. +void renderPlayer(int, int, Player): Muestra el tile del jugador por pantalla.

CLASE	Level
	Esta es una clase abstracta que contiene los atributos y métodos pertinentes para generar el nivel.
ATRIBUTOS	#int width: Ancho de la hoja donde están los colores de los sprites. #int height: Largo de la hoja donde están los colores de los sprites. #Tile[] catalogueOfTiles: Arreglo donde se guardan los tiles.
MÉTODOS	+int getWidth(): Mostrar el ancho de hoja. +Tile getCatalogueOfTiles(int): Muestra un tile en una posición especificada. #void loadLevel(String): Método que se hereda a la clase hija. #void generateLevel(): Método que se hereda a la clase hija. + void tick():Método que se hereda a la clase hija. + void render(int, int, Screen): Muestra por pantalla en la posición dada.

CLASE	LoadeLevel
	Esta clase es hija de level y se encarga de generar un nivel, guardando en un arreglo todos los tiles a mostrar.
ATRIBUTOS	- int[] pixels: Arreglo de pixeles
MÉTODOS	# void loadLevel(String): Recibe el nombre de un archivo (imagen .png) la cual contiene el mapeado de lo que será el

	nivel. Se encarga de almacenar los colores en el arreglo de pixeles. # void generateLevel(): Se encarga de examinar los valores contenidos en el arreglo de pixeles y dependiendo del mismo almacena un tile que se haya especificado.

CLASE	Keyboard
	Esta clase se encarga de los controles el teclado.
ATRIBUTOS	- int keysAvailable: Valor constante que representa el número de teclas que están disponibles. - boolean[] keys: Arreglo que representa las teclas. Cuando se presiona una tecla la posición del arreglo correspondiente se coloca true. + boolean up: Tecla subir. + boolean down:Tecla bajar. + boolean left: Tecla izquierda. + boolean right: Tecla derecha. + boolean exit: Tecla de salir.
MÉTODOS	+ void tick(): Actualiza cada una de las teclas. + void keyPressed(KeyEvent):Recibe un evento del teclad y la convierte verdadera en el arreglo keys. + void keyReleased(KeyEvent): Recibe un evento del teclad y la convierte falsa en el arreglo keys si se detecta que no es pulsada.

CLASE	Entidy
	Esta clase abstracta contiene los atributos y métodos a heredar.
ATRIBUTOS	# int x: Coordenadas de la entidad (fila). # int y: Coordenadas de la entidad (columna). - boolean isDead: Indica si el jugador murió. # Level level: La entidad.
MÉTODOS	+ void setX(int): Actualizar el valor de X. + int getX(): Mostar el valor de X. + void setY(int): Actualizar el valor de Y. + int getY(): Mostrar el valor de Y. + void setIsDead(): Actualizar el estado del jugador.

	+ boolean getIsDead(): Mostrar el estado del jugador. + void tick(): Método que se hereda. + void render(): Método que se hereda. - boolean isColliding(int, int): Maneja las colisiones.

CLASE	Creature
	Esta clase se encarga de las colisiones y movimientos de los personajes.
ATRIBUTOS	# Sprite sprite: Representa el sprite del personaje. # char direction: Se encarga de tomar la dirección del personaje. # boolean isMoving: Indica si se mueve el personaje.
MÉTODOS	+ void setSprite(Sprite): Actualiza el valor del sprite. + Sprite getSprite(): Muestra el sprite. + void move(int, int): Se encarga de mover el personaje.

CLASE	Player
	Esta clase se encarga de manejar al personaje del jugador.
ATRIBUTOS	- Keyboard keyboard: Representa el teclado. - int animation: Representa a un entero que va a manejar las animaciones del personaje.
MÉTODOS	+ void tick(): Se encarga de mostrar las animaciones y del estado del teclado para así poder mover al personaje. + void render(Screen): Se le pasa una pantalla y se llama al método para mostrar al jugador en la pantalla.

CLASE	Level_1
	Clase hija de LoadLevel. Esta se encarga de generar los sprites para el nivel 1.
ATRIBUTOS	atributos heredados de la clase padre.
MÉTODOS	Métodos heredados de la clase padre.

CLASE	Level_2
	Clase hija de LoadLevel. Esta se encarga de generar los sprites para el nivel 2.
ATRIBUTOS	atributos heredados de la clase padre.
MÉTODOS	Métodos heredados de la clase padre.

CLASE	Level_3
	Clase hija de LoadLevel. Esta se encarga de generar los sprites para el nivel 3.
ATRIBUTOS	atributos heredados de la clase padre.
MÉTODOS	Métodos heredados de la clase padre.

CLASE	Level_4
	Clase hija de LoadLevel. Esta se encarga de generar los sprites para el nivel 4.
ATRIBUTOS	atributos heredados de la clase padre.
MÉTODOS	Métodos heredados de la clase padre.

CLASE	DonkeyKong
	Clase hija de Creature. Esta se encarga de representar al enemigo principal del juego, Donkey Kong, y contiene todas las acciones que este puede realizar.
ATRIBUTOS	-int animation: Se encarga de llevar el control de las animaciones de Donkey Kong.
MÉTODOS	+ void lanzarbarril(): Hace que el Donkey Kong lance un barril por el escenario.

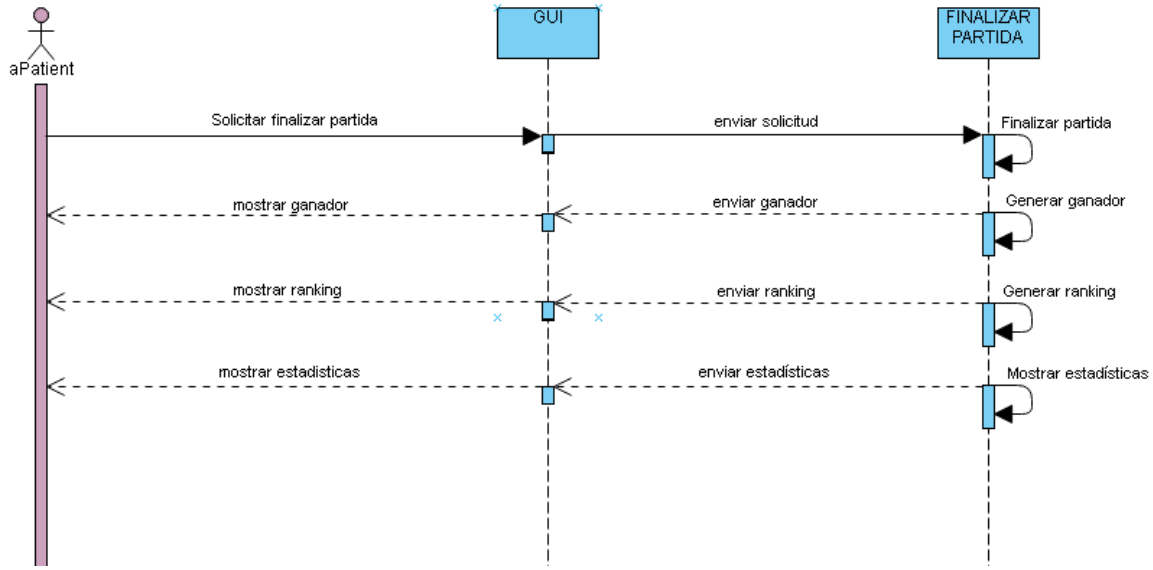
CLASE	Pauline
	Clase hija de Creature. Esta se encarga de representar a la damisela en peligro que los jugadores deben rescatar, Pauline..
ATRIBUTOS	-int animation: Se encarga de llevar el control de las animaciones de Pauline.
MÉTODOS	---------- no tiene métodos ----------

CLASE	Barril
	Clase hija de Creature. Esta se encarga de representar los barriles que va a lanzar Donkey Kong a lo largo de la partida.
ATRIBUTOS	-int animation: Se encarga de llevar el control de las animaciones de los barriles.
MÉTODOS	Métodos heredados de la clase padre.

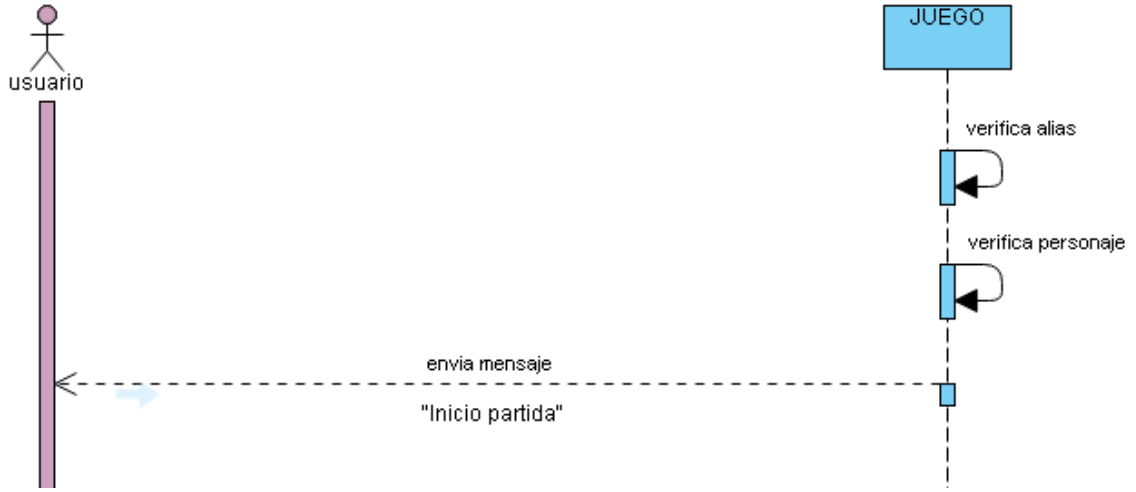
CLASE	Martillo
	Clase hija de Entity. Esta se encarga de representar el power-up de los jugadores cuando tienen el martillo.
ATRIBUTOS	-int animation: Se encarga de llevar el control de las animaciones del martillo.
MÉTODOS	Métodos heredados de la clase padre.
CLASE	BolaFuego
	Clase hija de Entity. Esta se encarga de representar al otro enemigo, las bolas de fuego.
ATRIBUTOS	-int animation: Se encarga de llevar el control de las animaciones de bolas de fuego.
MÉTODOS	Métodos heredados de la clase padre.

CAPITILO II – DISEÑO

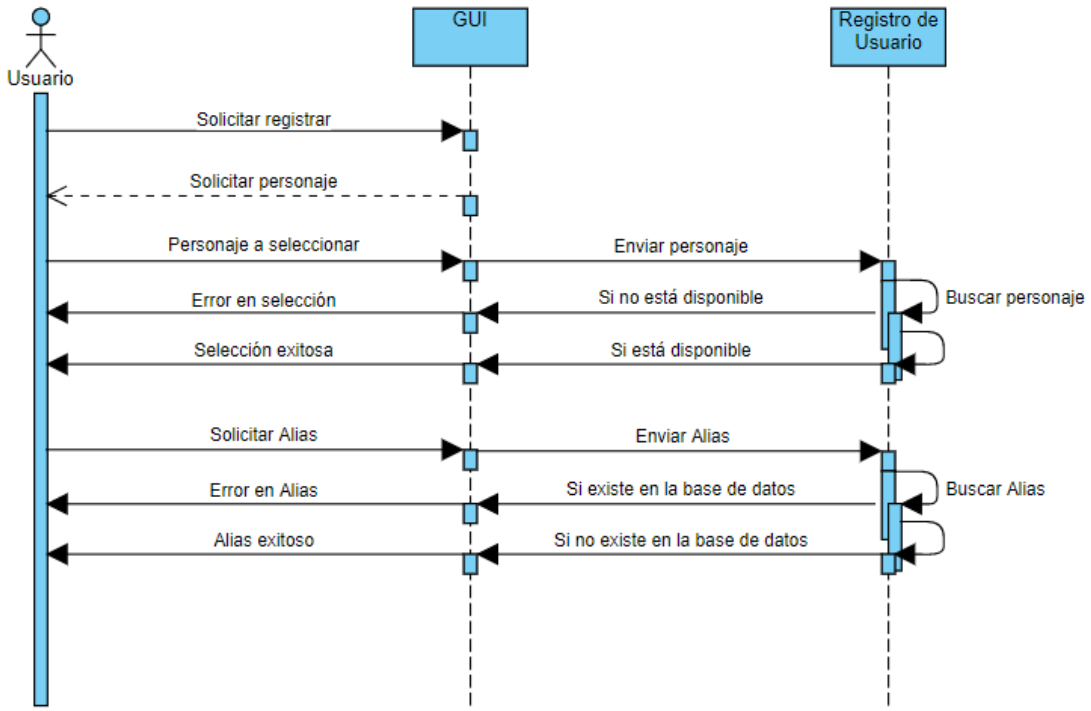
Diagramas de secuencia



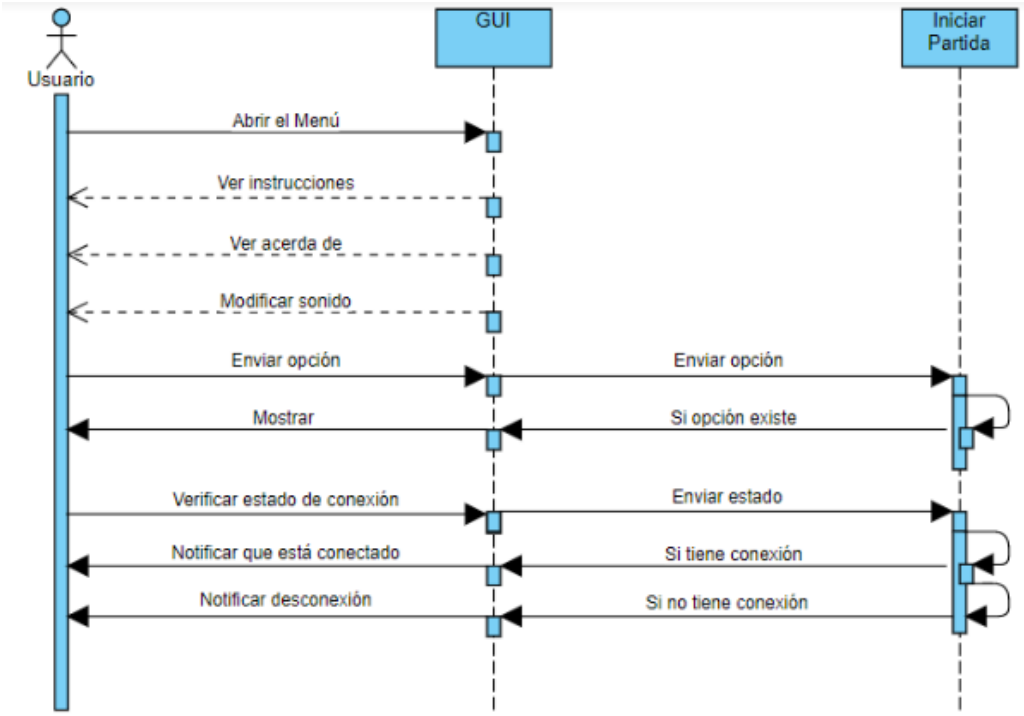
Caso de uso: Finalizar partida.



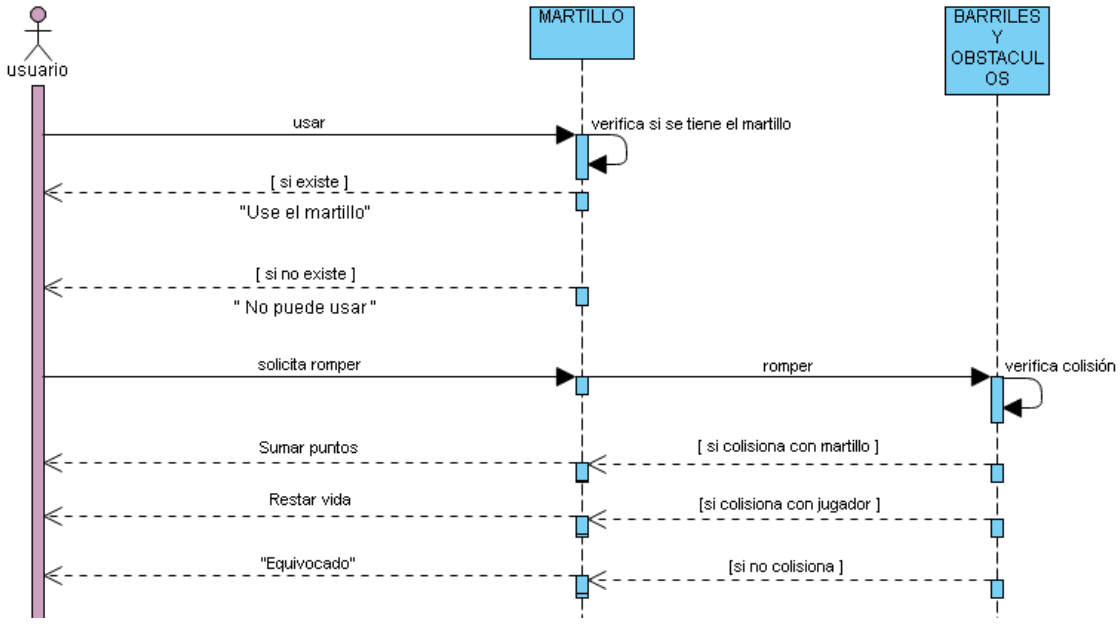
Caso de uso: Juego



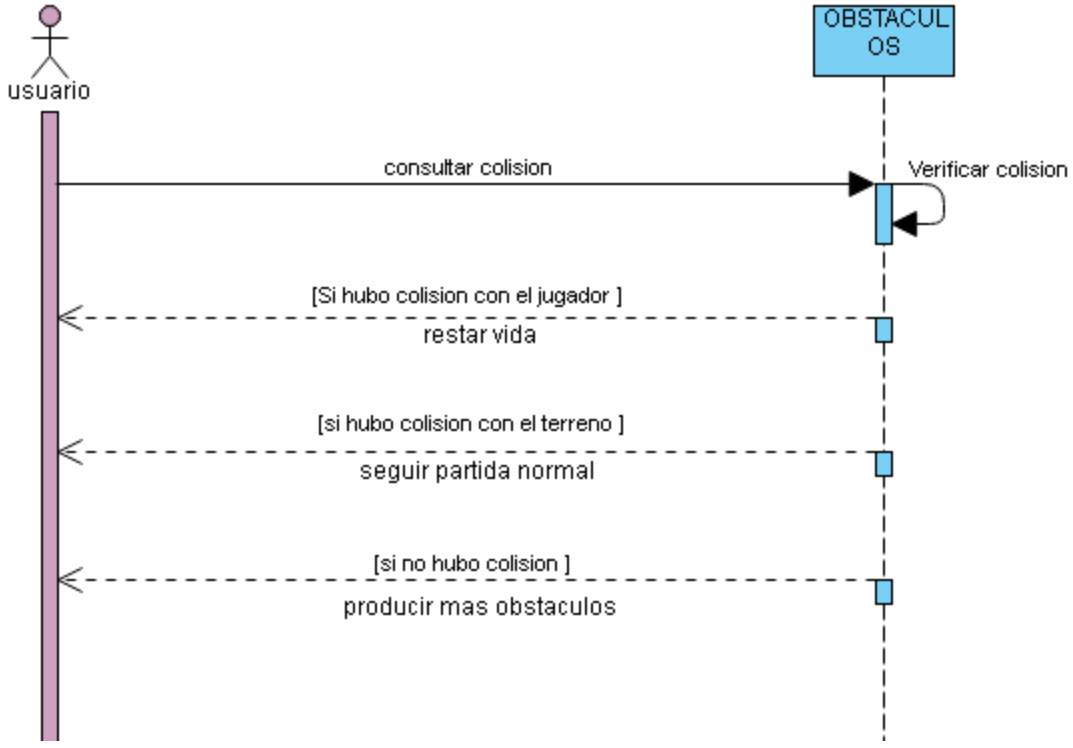
Caso de uso: Registrar usuario



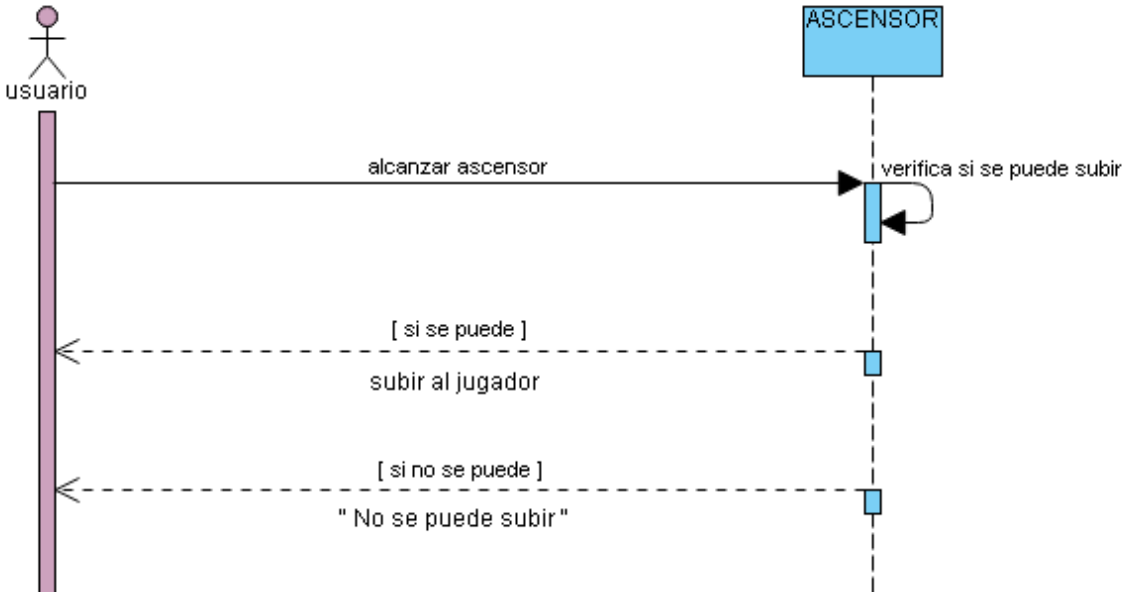
Caso de uso: Iniciar partida



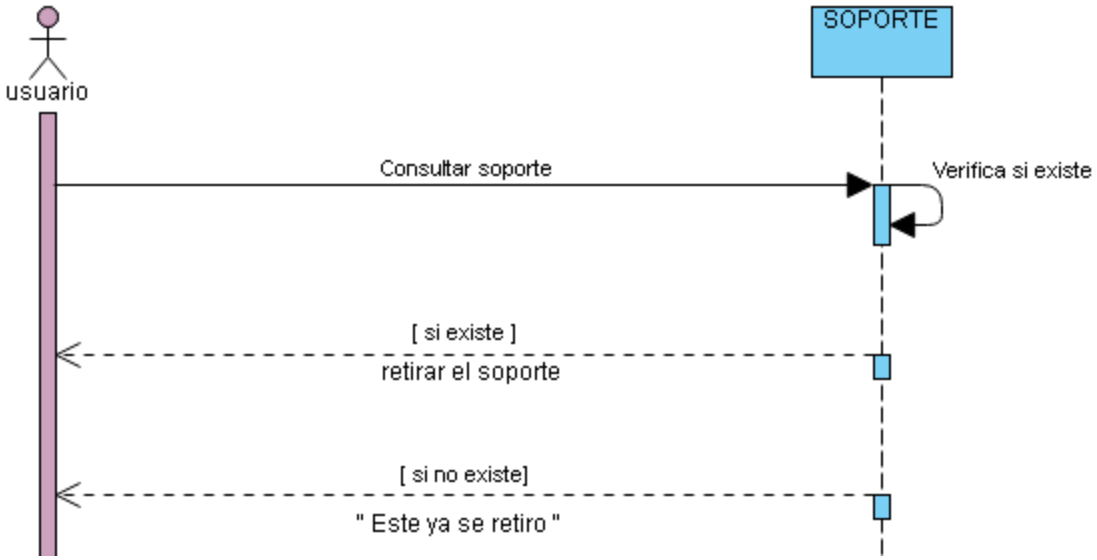
Caso de uso: Usar martillo



Caso de uso: esquivar obstáculos



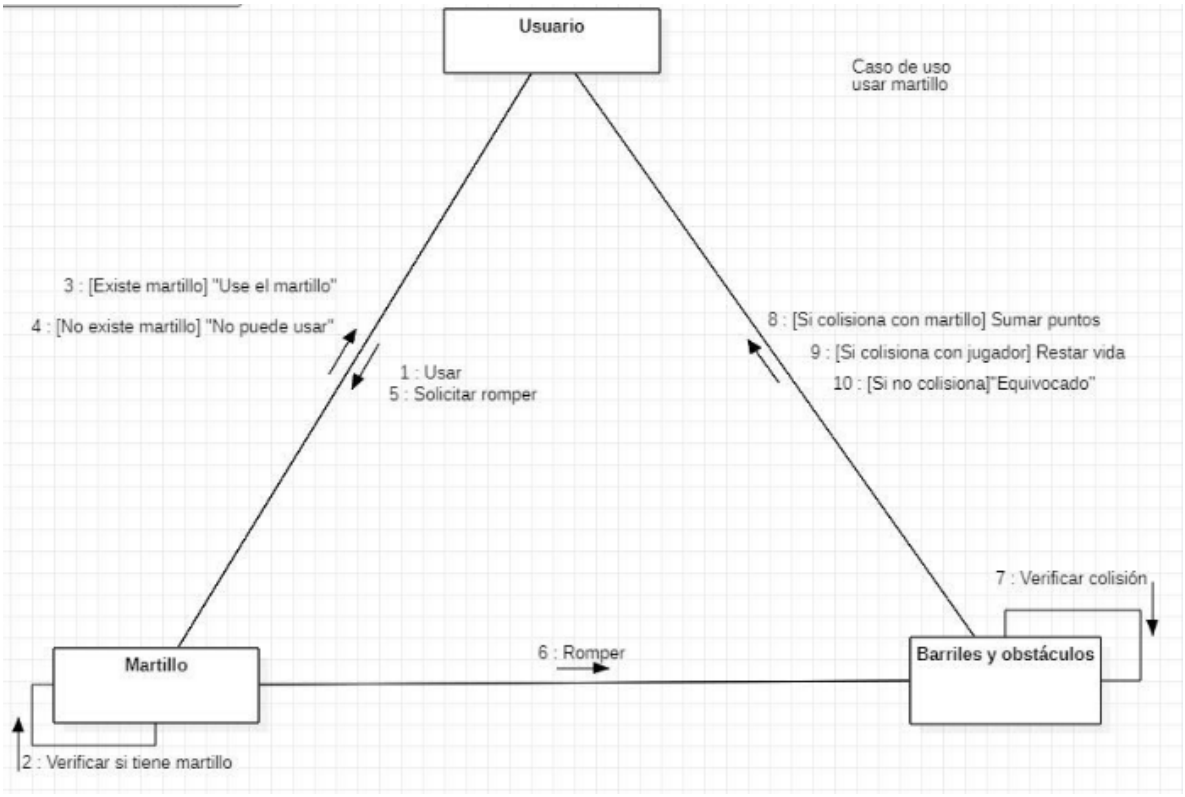
Caso de uso: Usar ascensor

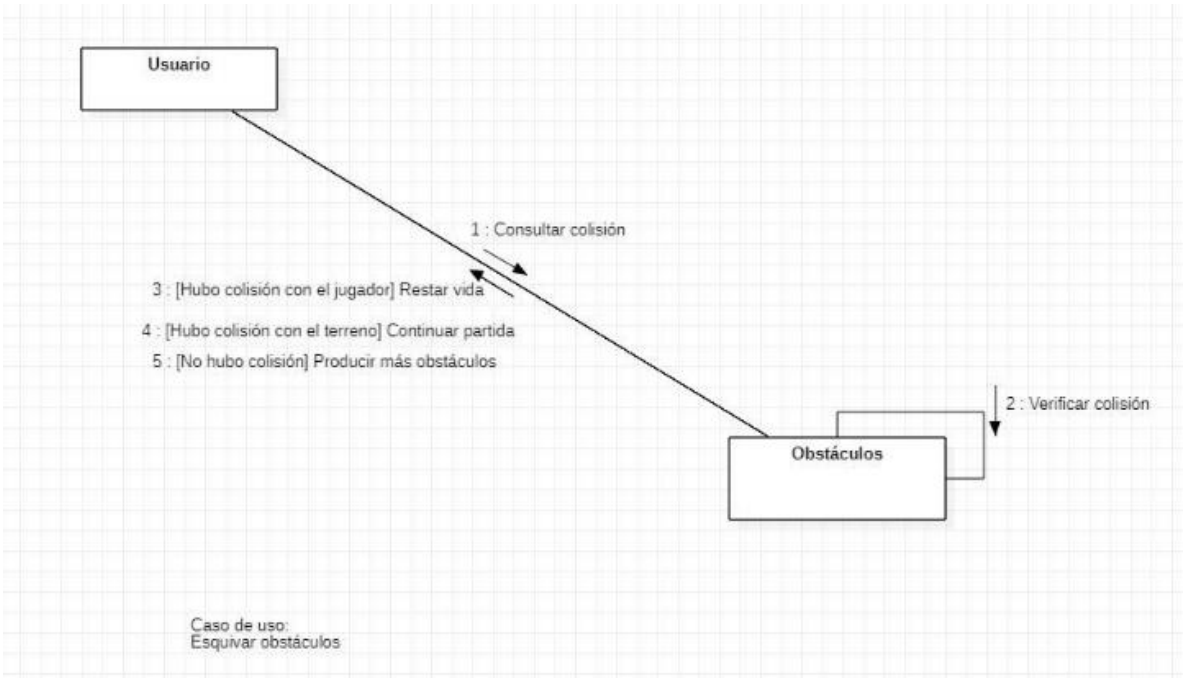


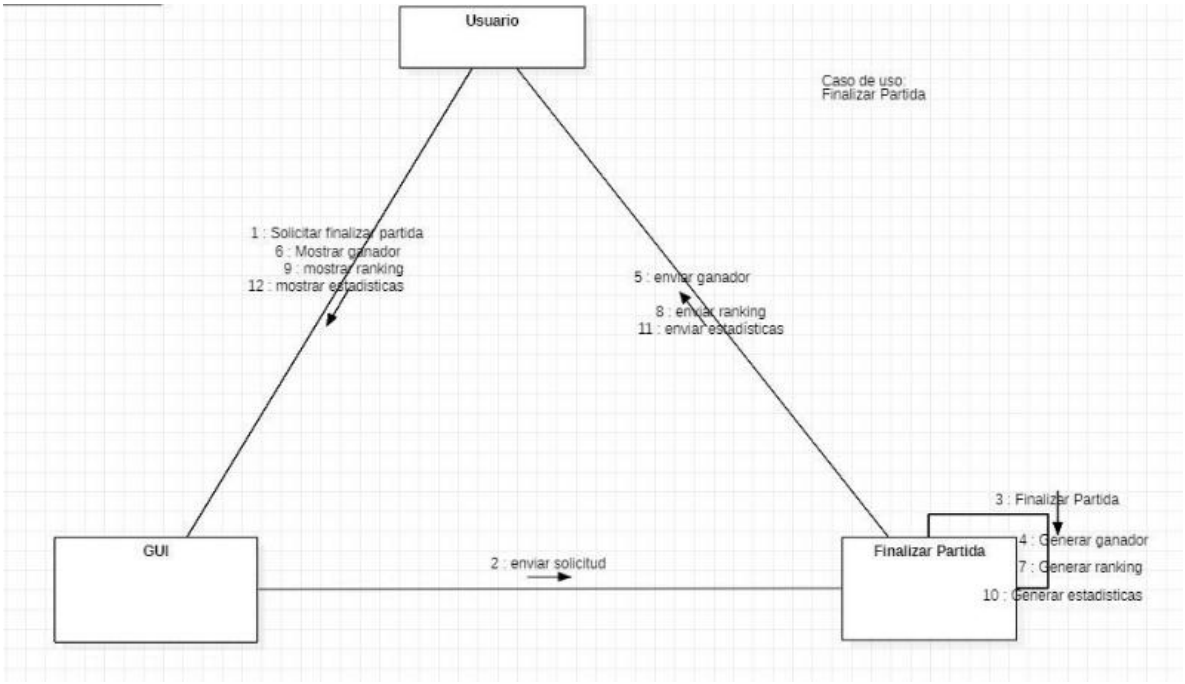
Caso de uso: quitar soporte

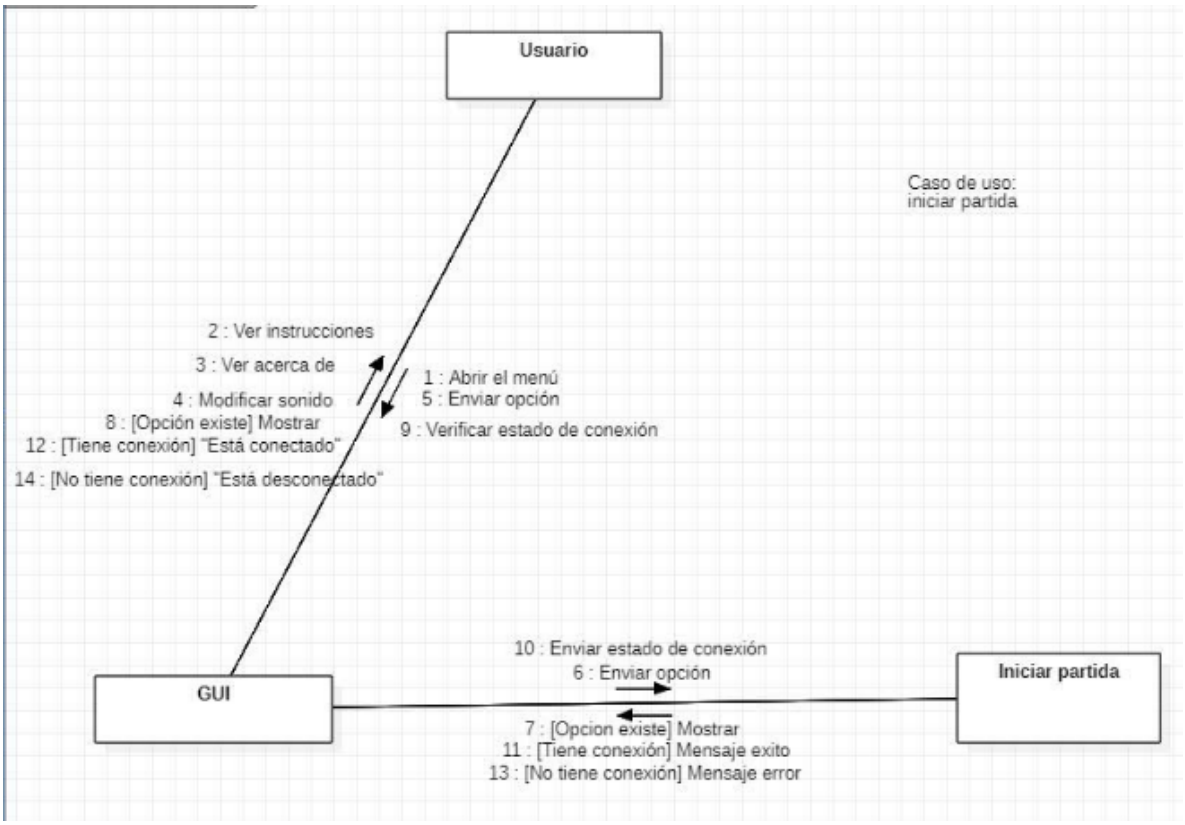
Diagramas de colaboración

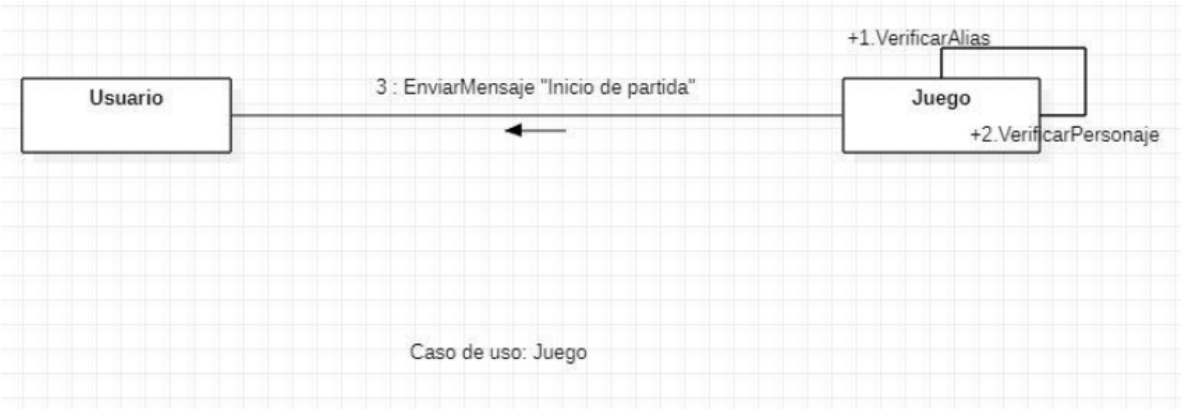
(De cada diagrama de secuencia)



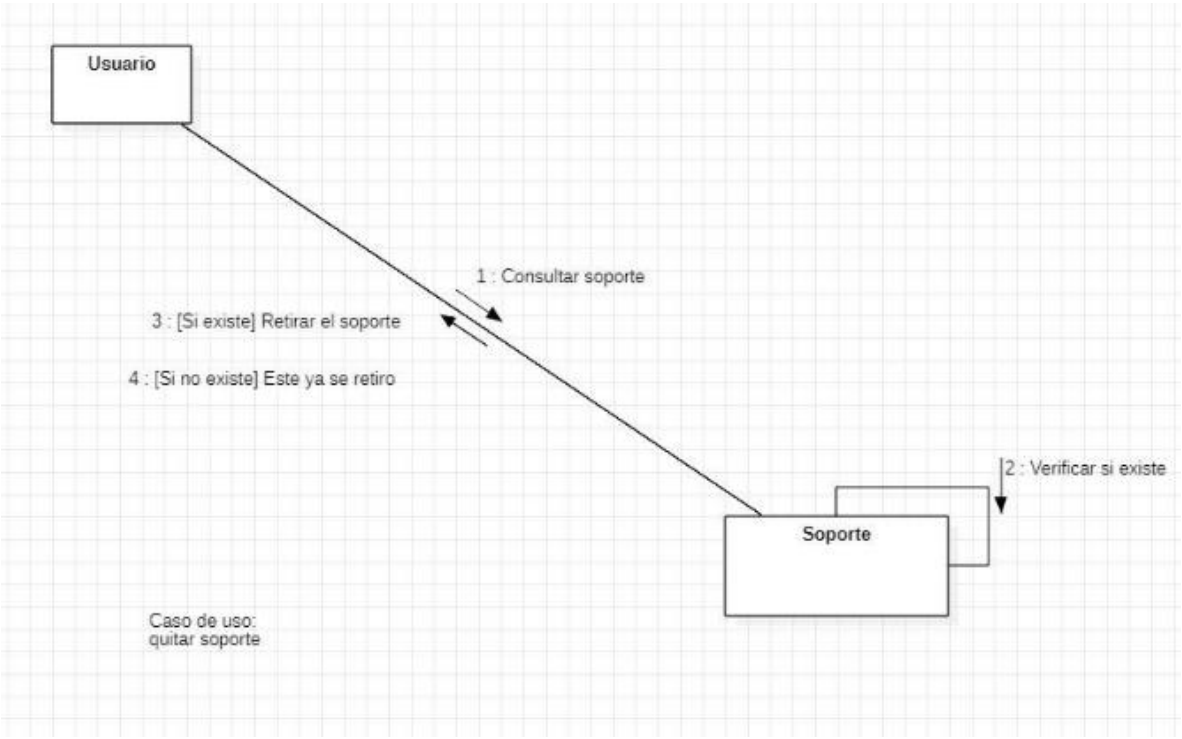



Usuario
Caso de uso
Finalizar Partida
1 : Solicitar finalizar partida
6 : Mostrar ganador
9 : mostrar ranking
12 : mostrar estadisticas
5 : enviar ganador
8 : enviar ranking
11 : enviar estadisticas
3 : Finalizar Partida
4 : Generar ganador
7 : Generar ranking
10 : Generar estadisticas
GUI
2 : enviar solicitud
Finalizar Partida

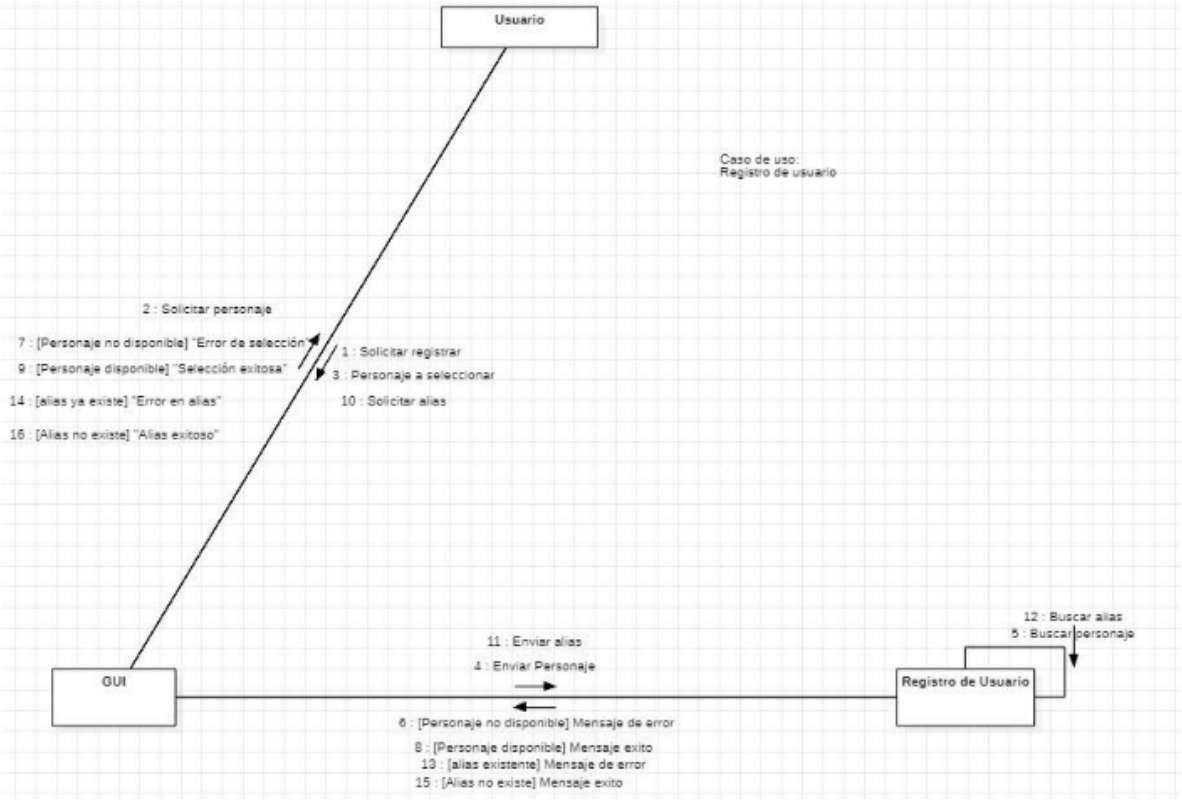

Usuario
Caso de uso:
iniciar partida
2 : Ver instrucciones
3 : Ver acerca de
4 : Modificar sonido
8 : [Opción existe] Mostrar
12 : [Tiene conexión] "Está conectado"
14 : [No tiene conexión] "Está desconectado"
1 : Abrir el menú
5 : Enviar opción
9 : Verificar estado de conexión
10 : Enviar estado de conexión
6 : Enviar opción
GUI
Iniciar partida
7 : [Opcion existe] Mostrar
11 : [Tiene conexión] Mensaje exito
13 : [No tiene conexión] Mensaje error

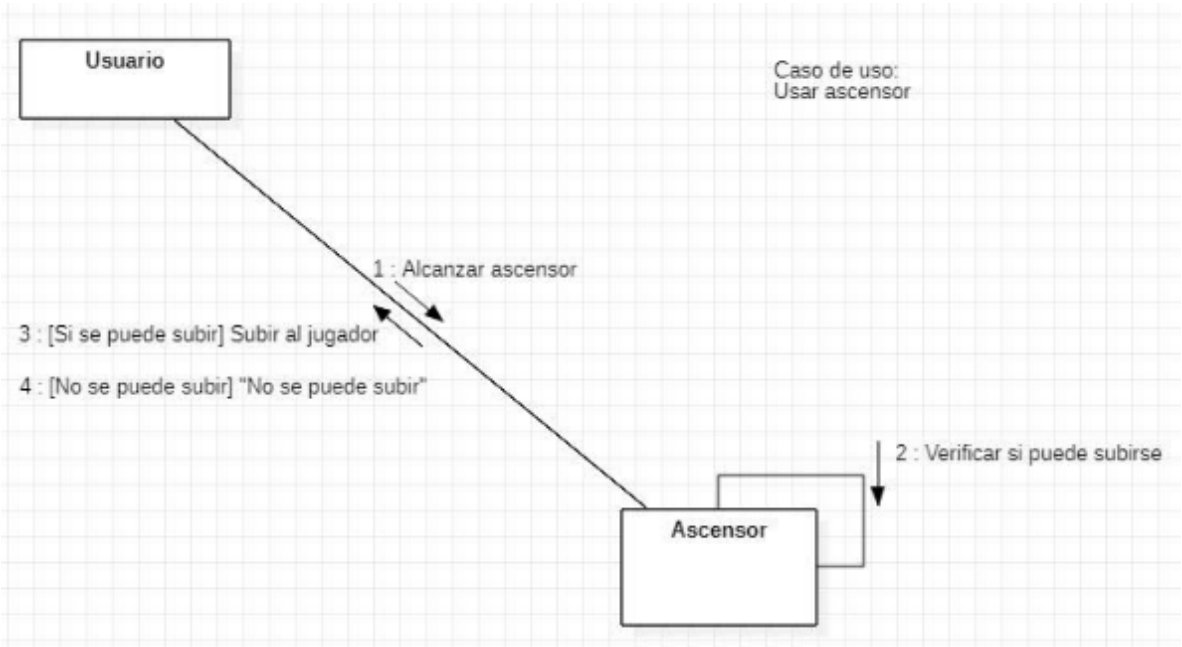


Caso de uso: Juego



Caso de uso:
quitar soporte

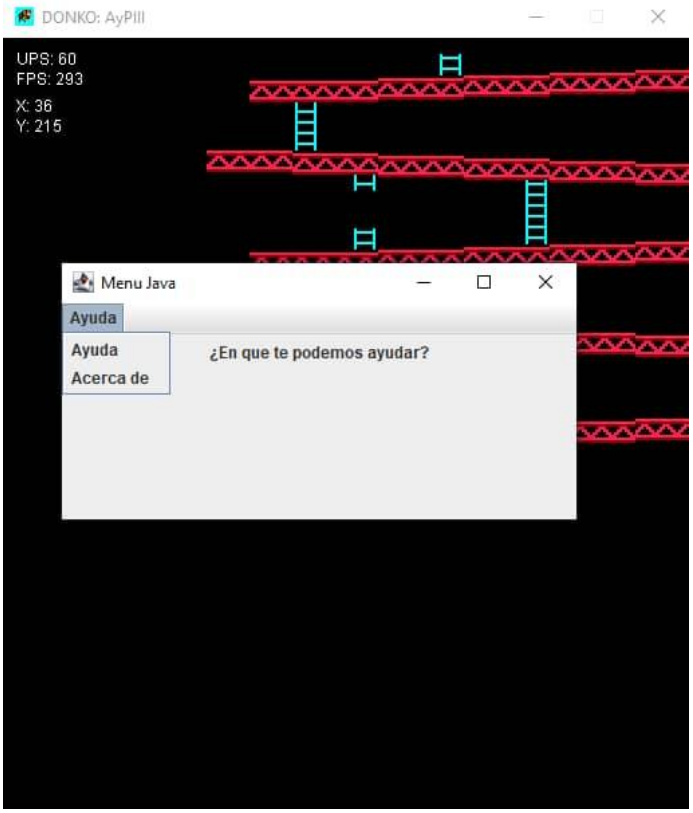

Usuario
Caso de uso:
Registro de usuario
2 : Solicitar personaje
7 : [Personaje no disponible] "Error de selección"
9 : [Personaje disponible] "Selección exitosa"
1 : Solicitar registrar
3 : Personaje a seleccionar
14 : [alias ya existe] "Error en alias"
10 : Solicitar alias
16 : [Alias no existe] "Alias exitoso"
12 : Buscar alias
5 : Buscar personaje
11 : Enviar alias
4 : Enviar Personaje
GUI
Registro de Usuario
6 : [Personaje no disponible] Mensaje de error
8 : [Personaje disponible] Mensaje exito
13 : [alias existente] Mensaje de error
15 : [Alias no existe] Mensaje exito


Usuario
Caso de uso:
Usar ascensor
1 : Alcanzar ascensor
3 : [Si se puede subir] Subir al jugador
4 : [No se puede subir] "No se puede subir"
2 : Verificar si puede subirse
Ascensor

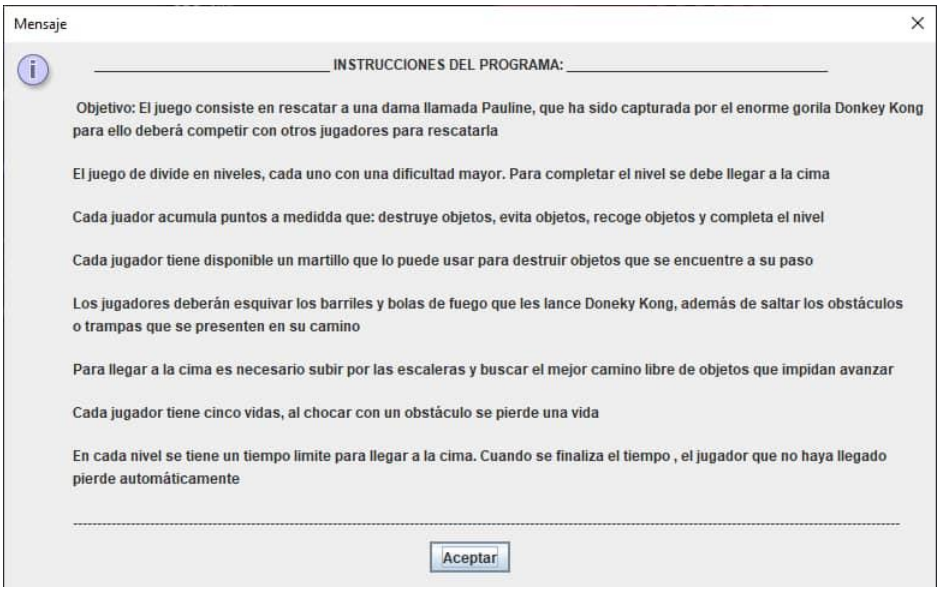
Diseño de interfaz gráfica

Pantalla que aparece apenas se abre el juego. Aquí es en donde se debería visualizar la entrada del juego, que te redirige tanto al menú como al juego, de la misma forma para salir del juego.

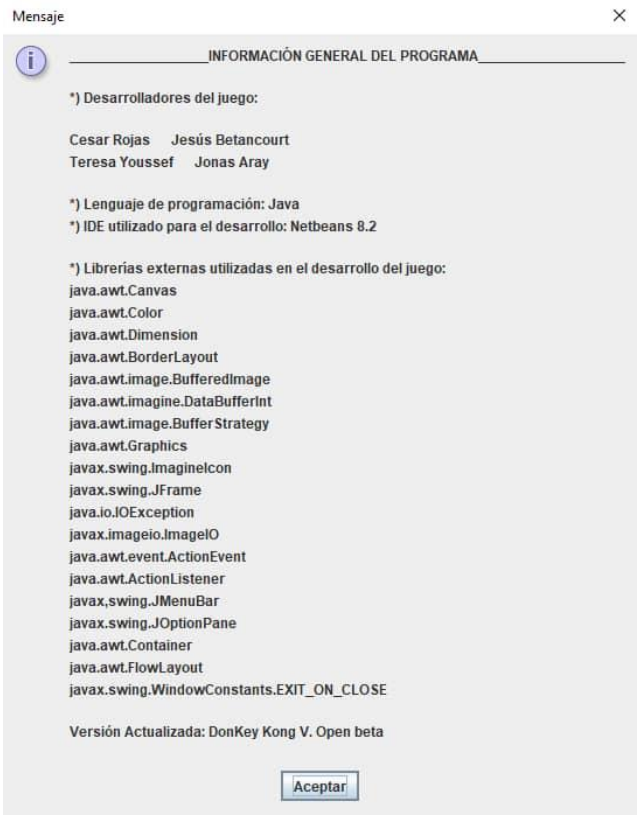
Ahora se observa cómo se visualiza el menú junto con las opciones del mismo. Esta pantalla le permite al usuario oprimir cualquiera de las dos opciones posibles, dependiendo de lo que desee.



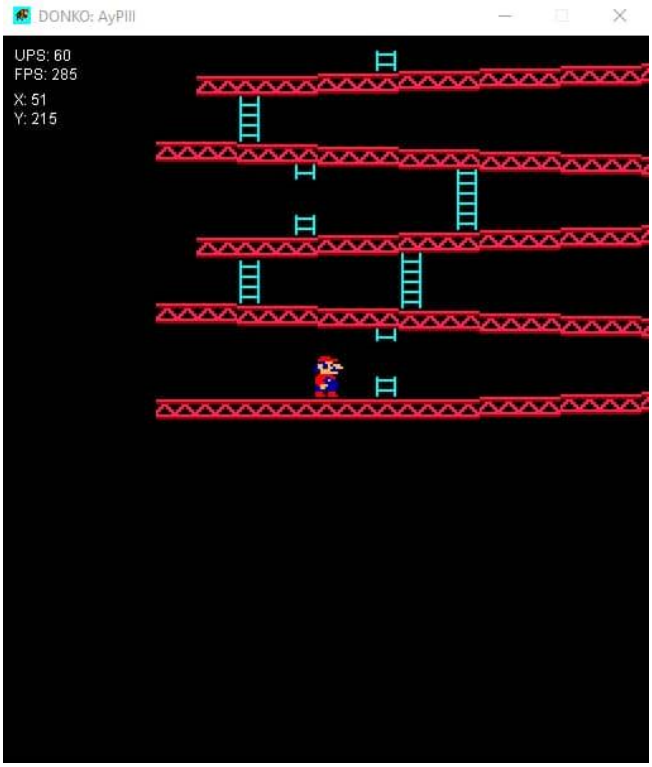
Esta pantalla aparece si el usuario presiona el botón de ayuda. Donde se muestran las instrucciones del programa.



Esta pantalla aparece si el usuario oprime la opción acerca de. Donde se muestra quienes desarrollaron el juego, el tipo de lenguaje utilizado para el mismo, el IDE y las librerías usadas.



Así se observa la pantalla a penas se inicia el juego.



Y así se observa cuando ya el usuario empezó a jugar