Nombre del grupo	Bonfire Studio
Titulación	Software D y Computadores A
Grupo reducido	GR1
Número de grupo	03
Repositorio Github	https://github.com/alvarolh33/marcelo
Espacio Trello	https://trello.com/invite/b/28mBT12B/ATTI98661c3078d615b0004ebe57ae78b524CE17531A/bonfire-studio
Memoria del proyecto	https://docs.google.com/document/d/1ISR2ngOB4Au9SglrSnYng9WFSheibC2c-5RWcu3rXZA/edit?usp=sharing

Número	**Nombre**	**Apellidos**	**Correo UMA**
1	Álvaro	Leal Huertas	alealhuer995@uma.es
2	Yago	Salas Trujillo	yagosalast@uma.es
3	Ignacio	González González	ignaciogg@uma.es
4	Jesús	Quiñones de las Peñas	jesusqdlp2003@uma.es

Nuestro proyecto: *La mazmorra de Marcelo*

Nuestro equipo, al que hemos apodado *Bonfire Studio*, va a desarrollar un videojuego llamado *La mazmorra de Marcelo*. Al ser un videojuego, el principal objetivo es entretener al usuario. Queremos ofrecer un videojuego sencillo y al que sea fácil aficionarse.

Tenemos en mente tres perfiles de usuarios: el jugador, el padre o madre que controlará los controles parentales de un jugador menor y el creador de niveles.

Nuestro juego pertenece al género *Dungeon Crawler*. Es un videojuego con perspectiva cenital. El jugador debe moverse en un escenario cuadriculado. El objetivo es avanzar entre salas derrotando enemigos, obteniendo objetos y subiendo de nivel hasta, finalmente, alcanzar la sala del jefe final y derrotarlo.

Hay un sistema de dinero y tienda para obtener objetos consumibles, como pociones, bombas o flechas, así como un banco que permite al jugador mantener el dinero obtenido entre partidas. Por otra parte, ciertos objetos clave, como la espada del jugador, suben de nivel tras derrotar suficientes enemigos. La tienda ofrece la posibilidad al jugador de almacenar el dinero obtenido en lugar de gastarlo, de manera que el jugador puede acceder a este más adelante desde cualquier archivo de guardado.

Hay enemigos de distintos tipos, divididos en dos categorías: melé y distancia, así como un jefe final, que conforma su propia categoría. Cada tipo de enemigo dará distintos objetos al jugador tras ser derrotado.

De la misma forma, también hay distintos tipos de sala según el contenido: salas de enemigos, salas de recompensas y de jefe final, entre otros.

Para mantener el progreso, el juego tiene un sistema de archivos de guardado que almacena tanto la posición del jugador como los objetos obtenidos, el nivel de los objetos clave y las salas completadas.

El juego tiene a su vez un sistema multijugador, tanto local como en línea. Cada jugador tiene su propio perfil y objetos. El jugador puede utilizar el mismo perfil tanto en partidas individuales como en partidas multijugador. La dificultad es proporcional al número de jugadores.

Roles

- Product Manager: Yago y Jesús
- Project Manager: Álvaro e Ignacio
- Diseño audiovisual y de interfaces gráficas: Ignacio y Jesús
- Control del personaje y diseño de enemigos y escenarios: Álvaro y Yago

Gestión de riesgos

Riesgos tecnológicos

Limitaciones del motor gráfico elegido que den lugar a problemas de rendimiento en el juego.

- Probabilidad: baja, ya que el motor gráfico elegido es una herramienta con un gran número de usuarios y con capacidad para desarrollar juegos de magnitudes muy superiores a la de nuestro proyecto.
- Efectos: catastróficos, puesto que un problema causado por el motor requeriría un gran número de cambios a nuestro proyecto, ya que el código estaría estructurado acorde al motor elegido.
- Estrategia: en el caso de presentarse una limitación del motor que nos impida desarrollar el proyecto de acuerdo a nuestra visión, podríamos mitigar dicha limitación modificando el código fuente del motor, ya que es un motor de código abierto y, por tanto, se puede modificar libremente.

Riegos de personal

Falta de conocimientos técnicos necesarios para la implementación de la funcionalidad requerida para el desarrollo del proyecto.

- Probabilidad: Alta, ya que somos un grupo formado por estudiantes y no tenemos experiencia previa con el motor y el lenguaje de programación escogidos.
- Efectos: serios, ya que una falta de conocimientos podría enlentecer considerablemente el desarrollo.
- Estrategia: hacer uso de la documentación oficial ofrecida por el proyecto de Godot, donde se explica la funcionalidad completa del motor gráfico, así como

Riesgos de organización

Fallos en la comunicación entre los miembros del equipo debidos a una dificultad usando las herramientas de control de versiones como Git y Github debido a nuestra falta de experiencia.

- Probabilidad: moderada, ya que aunque nos falte experiencia no son herramientas difíciles de entender y usar.
- Efectos: tolerables, ya que la escala del proyecto no es tan grande como para que errores en el control de versiones supongan graves consecuencias.
- Estrategia: utilizar los recursos disponibles en internet para aprender la forma correcta de utilizar las herramientas de control de versiones, así como practicar lo aprendido en repositorios de prueba.

Riesgos de herramientas

Las herramientas escogidas no son eficientes o tienen *bugs*, tenemos problemas a la hora de trabajar con distintas herramientas de desarrollo, ya sea por problemas de compatibilidad de formatos o de integración con el motor.

- Probabilidad: muy baja, ya que la gran mayoría del desarrollo se hará dentro del propio motor gráfico. Las únicas partes del proyecto que se desarrollarán con herramientas ajenas al motor serán los *sprites* y sonidos.

- Efectos: insignificantes, ya que los formatos de imágen y vídeo gozan de una alta capacidad de conversión. Si alguna incompatibilidad se produjese, sería sencillo solventarla.
- Estrategia: para evitar bugs, usaremos versiones estables de las herramientas escogidas (en contraposición a las versiones *nightly*) y trataremos de limitar el número de herramientas esenciales para el desarrollo del proyecto a una cantidad razonable.

Riesgos de requisitos

Vamos a emplear un modelo incremental, de manera que los requisitos cambiarán a lo largo del desarrollo del proyecto. Esto puede llevar a que un nuevo requisito del proyecto sea incompatible o requiera efectuar grandes cambios a la estructura del proyecto.

- Probabilidad: moderada, ya que los requisitos pueden cambiar pero la visión global del proyecto será estable, es decir, no pretendemos cambiar la idea general del proyecto.
- Efectos: tolerables, ya que el motor elegido ofrece mucha flexibilidad a la hora de cambiar la estructura de un proyecto. El motor utiliza una metodología basada en clases y objetos, cosa que facilita una posible reestructuración de parte del proyecto
- Estrategia: llevar un control estricto sobre qué requisitos nuevos pueden incluirse en el proyecto y considerar detenidamente si su implementación colisiona de alguna manera con los sistemas ya implementados.

Riesgos de estimación

No hacemos una correcta valoración del tiempo necesario para desarrollar el proyecto. Invertimos más tiempo del estimado en labores que no contribuyen o contribuyen poco en la finalización del proyecto.

- Probabilidad: alta, ya que es el primer proyecto que desarrollamos en equipo y con las herramientas escogidas, es muy probable que nuestras estimaciones de tiempo sean erróneas.
- Efectos: serios, ya que tenemos una fecha de entrega con la que tenemos que cumplir.
- Estrategia: empezar el desarrollo del proyecto con suficiente tiempo de antelación para poder así acostumbrarnos a las nuevas herramientas y metodologías de trabajo, así como mantener bajo control la escala del proyecto, teniendo en cuenta el número de personas involucradas y la experiencia.

Planificación

Hemos decidido implementar un modelo de desarrollo incremental. Consideramos que es el modelo que mejor se ajusta al desarrollo de un videojuego, ya que la separación en versiones nos permite probar mecánicas de manera gradual y cambiar el diseño de aspectos del juego de acuerdo a nuestras impresiones al jugarlo.

El desarrollo estará basado en versiones iterativas, de manera que el proyecto se irá refinando de manera gradual. Cada sistema o mecánica del juego deberá tener una implementación inicial que se irá mejorando de manera iterativa.

Hemos decidido clasificar las tareas en cuatro grupos principales:

🟥 Interfaz de usuario

- Menú del juego: Un menú navegable por el usuario con un conjunto de opciones relativas a los gráficos, el audio y los controles parentales.
- Gestión de inventarios: Implementar una interfaz para la gestión de objetos y equipamiento por parte del jugador.
- HUD (Heads-Up Display): La implementación del HUD, la interfaz gráfica que puede ver el jugador durante el ciclo normal de juego. Muestra la vida del jugador, las monedas, la puntuación, el tiempo de juego, etc.

🟩 Controles del usuario

- Control del personaje: Desarrollar el sistema de control de movimiento y ataque del personaje.
- Control del menú: Sistema de navegación por las diferentes interfaces gráficas como los menús o el inventario.
- Navegación entre salas: Implementación de la navegación entre las diferentes salas y escenarios del juego, basado en puertas que llevan de un escenario a otro.

🟦 Diseño de mecánicas

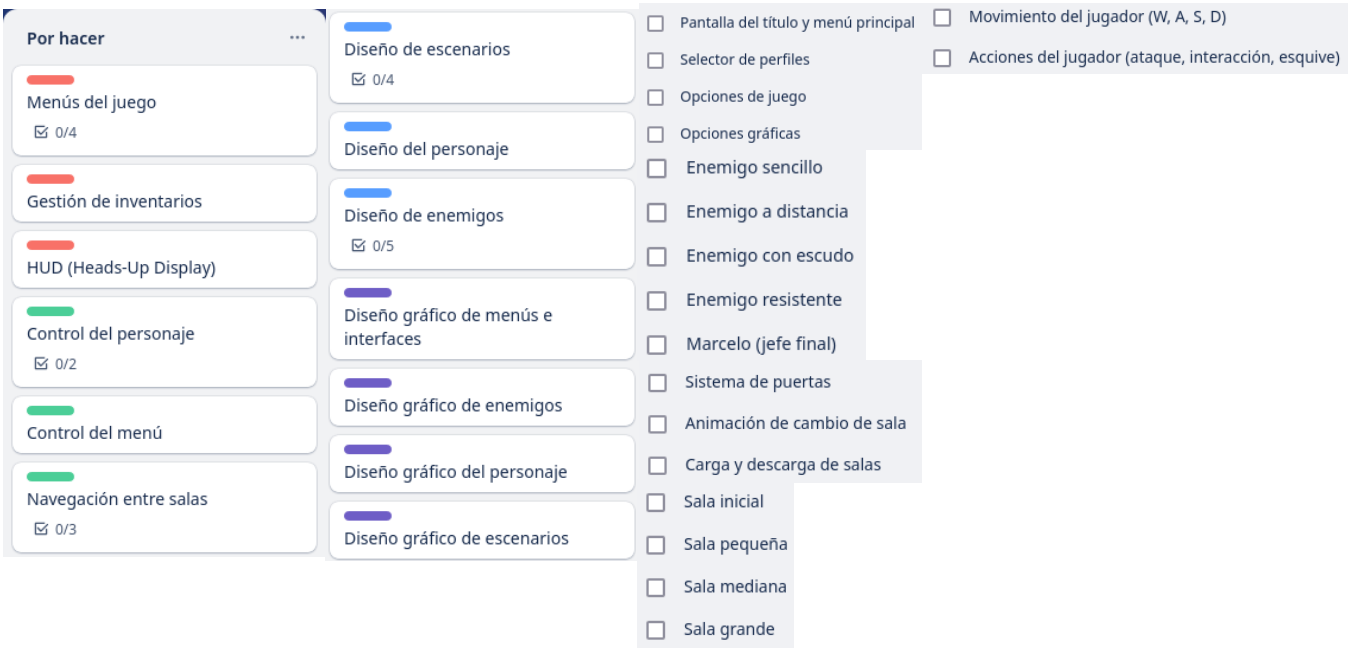
- Diseño de niveles y escenarios: La distribución de salas, tipos de salas, y su contenido, así como la progresión del jugador.
- Diseño del personaje principal: El desarrollo de su diseño mecánico, como lo serían sus diferentes habilidades, poderes o características del mismo.
- Diseño de enemigos: Diseño de su dificultad, formas de derrotarlos y principalmente su función de obstáculo ante el jugador.

🟪 Diseños gráficos

Esta categoría incluye el desarrollo y diseño de los diferentes sprites y elementos gráficos o visuales requeridos. Estos serían los diseños gráficos de:

- Menús e interfaces
- Enemigos
- Personaje
- Escenarios y gráficos (e.g. partículas, efectos visuales, etc.)

Planificación en Trello



Requisitos

Historias de usuario:

Requisitos funcionales

RF1: Menú principal
Como usuario quiero poder navegar por un menú con diferentes opciones para poder ajustarlas a mis necesidades durante la ejecución del programa.

RF2: Opciones
Como usuario quiero un botón de opciones en el menú principal que me permita hacer cambios, como controles, gráficos y sonido.

RF3: Botón Jugar
Como usuario quiero tener en el menú principal, una opción de inicio de partida.

RF4: Botón Salir
Como usuario quiero un botón para salir en el menú principal que acabe con la ejecución del programa.

RF5: Controles
Como usuario quiero que haya una sección en opciones en el que pueda cambiar los controles a mi gusto.

RF6: Gráficos
Como usuario me gustaría una opción que altere la calidad de la partida, como la resolución del juego.

RF7: Sonido
Como usuario quiero una opción en el apartado de opciones que me permita cambiar el volumen de la música y los efectos del juego.

RF8: Selector de perfiles
Como usuario quiero poder tener diferentes tipos de perfiles que me ofrezcan una experiencia de uso diferente con el juego.

RF9: Administrador
Como usuario quiero utilizar el programa como administrador para tener acceso a opciones adicionales que ajusten el perfil de control parental.

RF10: Perfil de control parental
Como padre, madre o tutor quiero poder utilizar el juego en un modo de control parental que permita una experiencia restringida y controlada del mismo. Eliminando la sangre, los ruidos etc, para poder ofrecer una experiencia a los usuarios más jóvenes.

RF11: Usuario normal
Como usuario quiero que haya una opción de usuario predeterminada en la que tenga una experiencia sin cambios.

RF12: Heads-Up Display
Como usuario debe haber una interfaz HUDs durante el desarrollo de la partida para poder ver información sobre el desarrollo de la partida sin tener que abrir menús.

RF13: Medidor de puntos de vida (HUD)
Como usuario debo poder ver en el HUD la cantidad de vida que tiene el jugador principal para poder verlas rápidamente y así poder conocer esta información en todo momento.

RF14: Modificadores y estadísticas (HUD)
Como usuario debo poder ver en el HUD las diferentes estadísticas y modificadores que tiene el personaje para no tener que acceder a un menú para ver esa información,

RF15: Mapa (HUD)
Como usuario debo poder ver en el HUD un minimapa donde aparezca la posición del jugador así como la de otros puntos de interés para poder orientarme por el escenario.

RF16: Input
Como usuario quiero que el programa sea capaz de recibir los inputs de teclado correspondientes a los controles asignados.

RF17: Abrir el menú
Como usuario quiero que haya una opción para abrir el menú durante la partida al pulsar una tecla.

RF18: Movimiento del personaje
Como usuario quiero que el personaje responda a los movimientos de los controles predeterminados o alterados por el usuario.

RF19: Navegación por el menú
Como usuario quiero que el selector del menú responda a los movimientos de los controles predeterminados o alterados por el usuario.

RF20: Acciones del personaje
Como usuario quiero que el personaje responda a las acciones que le indico por teclado, con los controles predeterminados o los alterados por el usuario.

RF21: Sistema de vidas (S.V)
Como usuario quiero que haya una forma de que el jugador pueda perder y ganar vida y que al llegar esta a 0 la partida finalice.

RF22: Perder vidas (S.V)
Como usuario quiero que haya una forma de perder vidas al ser el jugador golpeado por un enemigo o por un proyectil.

RF23: Ganar vidas (S.V)
Como usuario quiero que haya una forma de ganar vidas al obtener objetos que den esta vida, o estos en áreas determinadas donde sanarte.

RF24: Mapas basado en salas
Como usuario quiero que haya varios tipos de escenarios en forma de salas, avanzando de unas a otras a la vez en la que avanza en el juego.

RF25: Movimiento entre salas a través de puertas
Como usuario quiero que haya puertas en algunas paredes específicas de las salas que me permitan avanzar entre salas.

RF26 : Carga y descarga de salas
Como usuario quiero que las salas que no sean visibles por el jugador se descarguen para ahorrar recursos. Y que cuando sea necesario el juego vuelva a cargar las salas.

RF27 : Enemigos
Como usuario quiero que haya enemigos que supongan un reto a la hora de pasarse el juego.

RF28: Acciones de los enemigos
Debe haber enemigos capaces de seguir al personaje a una velocidad constante e iniciar su ataque cuando estén a una distancia determinada, estos valores dependen de la dificultad.

RF29: Atacar
Como usuario quiero que el enemigo sea capaz de atacar a tu personaje, y le quite vida

RF30 : Objetos
Como usuario quiero que haya varios tipos de objetos, dependiendo de la utilidad de cada uno de ellos.

RF31 : Objetos consumibles
Como usuario quiero que haya objetos de tipo consumible que modifiquen las estadísticas del jugador o le den efectos adicionales.

Requisitos no funcionales

RNF1: Botón Idioma
Como usuario quiero un botón en el menú principal que me permita cambiar entre diferentes idiomas el texto del juego.

RNF2: Multijugador
Como usuario quiero una opción en el menú principal que permite que un usuario externo se una a mi partida (con los permisos correspondientes) y que yo me pueda unir a la de otro usuario.

RNF3: Crear Partida
Como usuario quiero que en el menú de Multijugador haya una opción para crear una partida con la opción de recibir a usuarios externos.

RNF4: Unirse a partida
Como usuario quiero que en el menú de Multijugador haya una opción para unirse a una partida de un usuario externo.

RNF5: Compatibilidad con mandos de videoconsolas
Como usuario me gustaría tener la posibilidad de poder usar los controles de otras plataformas para poder jugar.

RNF6: Efectos de sonido
Como usuario quiero que haya diferentes efectos de sonido que respondan a los eventos ocurridos en pantalla por el jugador y su entorno.

RNF7: Tipos de salas
Como usuario quiero que las salas que se generen sean diferentes y nunca entres en dos escenarios iguales.

RNF9: Salas de enemigos
Como usuario me gustaría que uno de los tipos de salas, sea en la que estén los enemigos que tengo que derrotar para continuar el juego. Además las puertas de las salas se cerrarán hasta que no termines con todos los enemigos de la sala.

RNF10 : Sala de recompensas
Como usuario me gustaría que uno de los tipos de salas, sea una en la recibas recompensas, como vida, dinero y objetos varios.

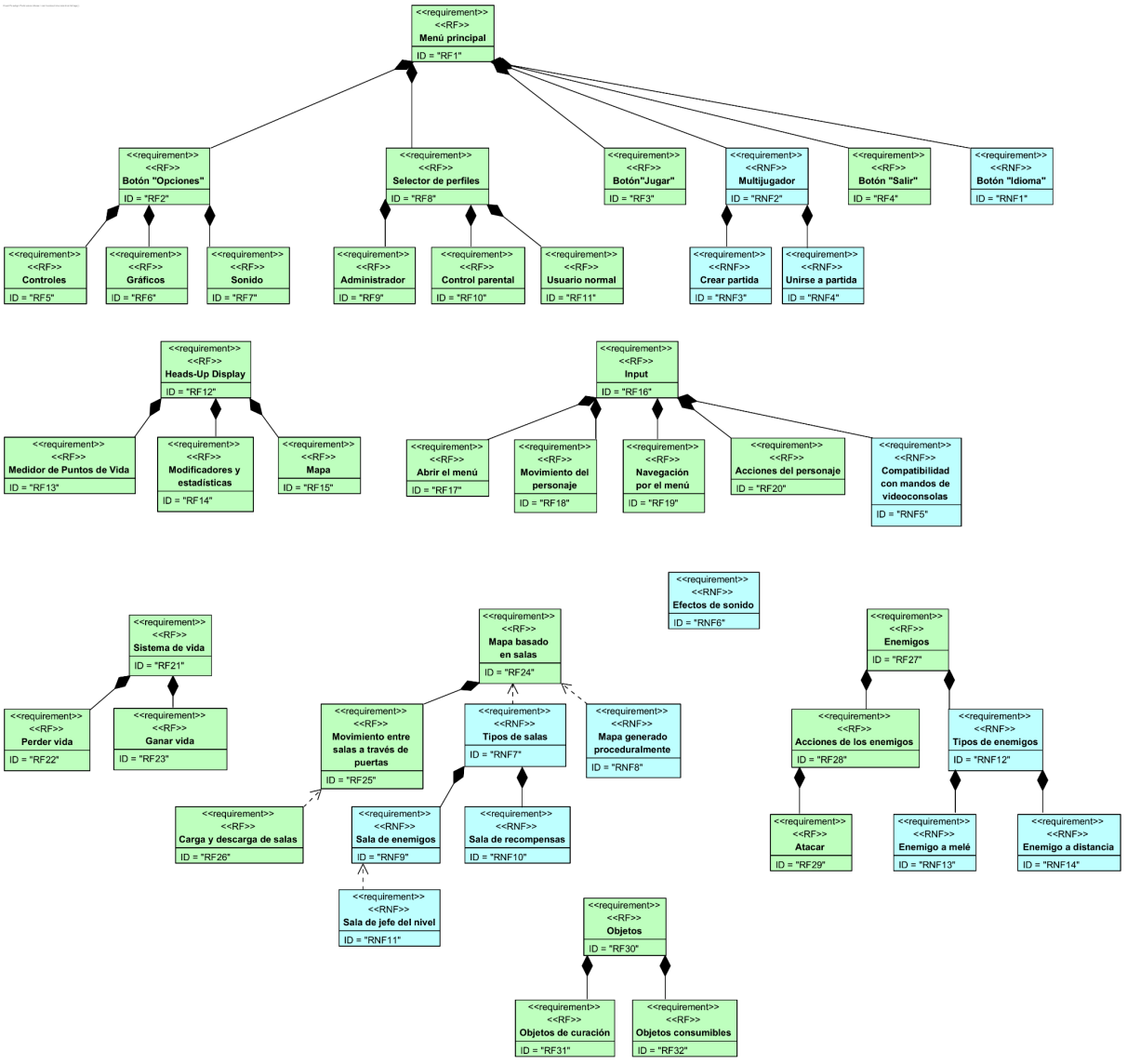
RNF11: Sala de jefe de nivel
Como usuario quiero que exista un tipo de sala especial donde haya uno o varios tipos de enemigos más difíciles de lo normal.

RNF12 : Tipos de enemigos
Como usuario quiero que haya diferentes enemigos dependiendo de en qué parte nos encontremos del juego y para darle dificultad al mismo.

RNF13: Enemigo a melé
Como usuario quiero que exista un tipo de enemigo que trate de dañar al jugador por medio del contacto directo

RNF14: Enemigo a distancia
Como usuario quiero que exista un tipo de enemigo que trate de dañar al jugador desde la distancia.

Diagrama de requisitos de Visual Paradigm:



Casos de uso

CU1: Guardar partida

- Contexto de uso:
  Cuando el jugador lo desee, podrá guardar el estado de su partida.
- Precondiciones y activación:
  El jugador está jugando una partida y en el apartado de guardado en el menú de opciones.
- Garantías de éxito / Postcondición:
  Se crea una nueva entrada en el menú de guardado con la fecha y hora actuales.
- Escenario principal:
  1. El jugador pulsa la opción de guardar la partida.
  2. El juego crea una nueva entrada en el listado de partidas guardadas.
- Escenarios alternativos:
  1. La partida no se ha guardado correctamente y se muestra un mensaje al jugador.

CU2: Cargar partida

- Contexto de uso:
  Cuando el jugador lo desee, podrá cargar una partida previamente guardada.
- Precondiciones y activación:
  El jugador está en el menú principal y en el apartado de jugar.
- Garantías de éxito / Postcondición:
  Se carga la partida seleccionada por el jugador.
- Escenario principal:
  1. El jugador selecciona la partida que quiere cargar.
  2. El jugador selecciona la opción “Jugar”.
  3. El juego carga la partida seleccionada.
- Escenarios alternativos:
  1. La partida no se ha cargado correctamente y se muestra un mensaje al jugador.

CU3: Seleccionar idioma

- Contexto de uso:
  Cuando el jugador lo desee, podrá cambiar el idioma del juego.
- Precondiciones y activación:
  El jugador está en el menú principal.
- Garantías de éxito / Postcondición:
  El idioma del juego ha cambiado al seleccionado por el jugador.
- Escenario principal:
  1. El jugador selecciona la opción para cambiar el idioma en el menú principal.
  2. El jugador selecciona el idioma deseado de la lista de idiomas disponibles.
  3. El jugador selecciona la opción de aplicar.
  4. El idioma del juego cambia al seleccionado.
- Escenarios alternativos:
  1. El cambio de idioma falla y se muestra un mensaje de error al jugador.
  2. El idioma deseado no forma parte de la lista.

CU4: Seleccionar el perfil de jugador

- Contexto de uso:
  Cuando el jugador lo desee, podrá cambiar el perfil de usuario en uso.
- Precondiciones y activación:
  El jugador está en el menú principal.
- Garantías de éxito / Postcondición:
  El perfil en uso ha cambiado al seleccionado por el jugador.
- Escenario principal:
  1. El jugador selecciona la opción para cambiar el perfil de usuario en el menú principal.
  2. El jugador selecciona el perfil deseado de la lista de perfiles existentes.
  3. De ser un perfil protegido, el jugador rellena el campo de contraseña.
  4. El jugador selecciona la opción para cambiar al perfil seleccionado.
  5. El perfil en uso cambia al seleccionado.
- Escenarios alternativos:
  1. La contraseña no es válida y no se efectúa el cambio de perfil.

CU5: Jugar en modo multijugador

- Contexto de uso:
  Cuando el jugador lo desee, podrá jugar en modo multijugador.
- Precondiciones y activación:
  El jugador está en el menú principal.
- Garantías de éxito / Postcondición:
  Se carga la partida seleccionada en modo multijugador.
- Escenario principal:
  1. El jugador selecciona la opción de juego multijugador.
  2. El jugador selecciona la partida guardada que desea cargar.
  3. El jugador introduce el número de jugadores en el campo de jugadores.
  4. El jugador selecciona los perfiles de jugador que jugarán.
  5. El jugador selecciona la opción de jugar.
  6. El juego carga la partida seleccionada con el número de jugadores seleccionado.
- Escenarios alternativos:
  1. El jugador introduce un número de jugadores menor o igual a cero y se muestra un mensaje de error.

CU6: Cambiar los ajustes

- Contexto de uso:
  Cuando el jugador lo desee, podrá cambiar los ajustes del juego.
- Precondiciones y activación:
  El jugador está en el menú principal.
- Garantías de éxito / Postcondición:
  Se aplican los cambios seleccionados por el jugador.
- Escenario principal:
  1. El jugador selecciona la opción de ajustes.
  2. El jugador selecciona los ajustes que desea aplicar de la lista.
  3. El jugador selecciona la opción de aplicar los ajustes.
  4. Los ajustes seleccionados se aplican.

- Escenarios alternativos:
    1. El cambio de ajustes falla y se notifica al jugador.

CU7: Salir al menú principal

- Contexto de uso:
  Cuando el jugador se encuentre dentro de una partida, podrá salir al menú principal.
- Precondiciones y activación:
  El jugador se encuentra dentro de una partida.
- Garantías de éxito / Postcondición:
  La partida en proceso se cierra y se accede al menú principal.
- Escenario principal:
    1. El jugador abre el menú con el botón ESC.
    2. El jugador selecciona la opción de cerrar la partida.
    3. La partida se cierra y se vuelve al menú principal.
- Escenarios alternativos:
    1. No se logra cerrar la partida y se notifica al jugador.

CU8: Cerrar el juego

- Contexto de uso:
  Cuando el jugador lo desee, podrá cerrar el juego.
- Precondiciones y activación:
  El jugador se encuentra en el menú principal.
- Garantías de éxito / Postcondición:
  El juego se cierra.
- Escenario principal:
    1. El jugador selecciona la opción de cerrar el juego del menú principal.
- Escenarios alternativos:
    1. No se logra cerrar el juego y se notifica al jugador.

CU9: Atacar con la espada

- Contexto de uso:
  Ante un enemigo, el jugador podrá atacar con la espada.
- Precondiciones y activación:
  El jugador se encuentra dentro de una partida.
- Garantías de éxito / Postcondición:
  El jugador efectúa la animación de ataque con la espada y los enemigos que se encuentren en la trayectoria del ataque reciben daño.
- Escenario principal:
    1. El jugador pulsa el botón de ataque.
    2. El ataque se efectúa.
- Escenarios alternativos:
    1. Se produce un error y el ataque no se efectúa.

CU10: Atacar con el arco

- Contexto de uso:
  Ante un enemigo, el jugador podrá atacar con el arco.
- Precondiciones y activación:

El jugador se encuentra dentro de una partida.

- Garantías de éxito / Postcondición:
  El jugador efectúa la animación de ataque con el arco, una flecha es disparada y los enemigos que se encuentren en la trayectoria del ataque reciben daño. El contador de flechas del jugador disminuye.
- Escenario principal:
  1. El jugador pulsa el botón de ataque a distancia.
  2. El ataque se efectúa.
- Escenarios alternativos:
  1. El jugador no tiene flechas y no se efectúa el ataque.

CU11: Usar un objeto consumible

- Contexto de uso:
  Durante la partida, el jugador podrá cambiar sus estadísticas con objetos consumibles.
- Precondiciones y activación:
  El jugador se encuentra dentro de una partida.
- Garantías de éxito / Postcondición:
  El jugador efectúa la animación de consumir un objeto y la característica afectada aumenta (e.g. si se trata de un elixir de curación, aumentan los puntos de vida).
- Escenario principal:
  1. El jugador pulsa el botón de usar consumibles.
  2. El objeto se consume.
- Escenarios alternativos:
  1. El jugador no tiene objetos consumibles y no se efectúa la acción.

CU12: Comprar objetos

- Contexto de uso:
  Estando en la tienda, el jugador podrá comprar objetos consumibles.
- Precondiciones y activación:
  El jugador está jugando una partida y se encuentra dentro de la tienda.
- Garantías de éxito / Postcondición:
  El objeto comprado aparece en el inventario del jugador y el dinero del jugador disminuye en igual cantidad al precio del objeto.
- Escenario principal:
  1. El jugador entra en la tienda.
  2. El jugador selecciona el objeto que desea comprar.
  3. El jugador pulsa el botón de comprar.
  4. El objeto comprado aparece en el inventario del jugador y su dinero se ve disminuido.
- Escenarios alternativos:
  1. El jugador no tiene el dinero necesario para comprar el objeto y se muestra un mensaje al jugador.

CU13: Depositar dinero

- Contexto de uso:
  Estando en la tienda, el jugador podrá depositar dinero en el cajero.

- Precondiciones y activación:
  El jugador está jugando una partida y se encuentra dentro de la tienda.
- Garantías de éxito / Postcondición:
  El jugador ha podido depositar dinero y se ha restado esta cantidad a su dinero. El dinero almacenado ha aumentado en igual proporción al dinero depositado.
- Escenario principal:
  1. El jugador entra en la tienda.
  2. El jugador selecciona la opción del cajero.
  3. El jugador selecciona la opción de depositar dinero.
  4. El jugador introduce la cantidad a depositar.
  5. El jugador selecciona la opción de realizar la transacción y el dinero depositado es restado al dinero poseído.
- Casos alternativos:
  1. El jugador no tiene suficiente dinero para efectuar la transacción, se deposita todo el dinero poseído.
  2. Se ha introducido una operación incorrecta (e.g. dinero negativo) y el cajero muestra un mensaje de error.

CU14: Sacar dinero

- Contexto de uso:
  Estando en la tienda, el jugador podrá sacar dinero depositado previamente.
- Precondiciones y activación:
  El jugador está jugando una partida y se encuentra dentro de la tienda.
- Garantías de éxito / Postcondición:
  El jugador ha podido sacar dinero y se ha sumado esta cantidad a su dinero. El dinero almacenado ha disminuido en igual proporción al dinero sacado.
- Escenario principal:
  1. El jugador entra en la tienda.
  2. El jugador selecciona la opción del cajero.
  3. El jugador selecciona la opción de sacar dinero.
  4. El jugador introduce la cantidad a sacar.
  5. El jugador selecciona la opción de realizar la transacción y el dinero sacado es sumado al dinero poseído.
- Casos alternativos:
  1. El cajero no tiene almacenado suficiente dinero para efectuar la transacción, se saca todo el dinero del cajero y se queda vacío.
  2. Se ha introducido una operación incorrecta (e.g. dinero negativo) y el cajero muestra un mensaje de error.

CU15: Configurar la visibilidad de contenido explícito

- Contexto de uso:
  Desde el perfil de padre/madre se podrá configurar el contenido explícito.
- Precondiciones y activación:
  El perfil seleccionado es el de padre/madre y el usuario se encuentra en el menú principal.
- Garantías de éxito / Postcondición:
  La opción de visibilidad seleccionada entra en vigor.

- Escenario principal:
    1. El usuario selecciona la opción de controles parentales.
    2. El usuario selecciona la opción de contenido explícito.
    3. El usuario selecciona el ajuste deseado (✔/✘).
    4. El usuario selecciona la opción de aplicar.
    5. El ajuste se aplica.
- Escenarios alternativos:
    1. El cambio de ajustes falla y se notifica al usuario.

CU16: Establecer un límite de horas de juego

- Contexto de uso:
    Desde el perfil de padre/madre se podrá limitar el número de horas de juego diarias.
- Precondiciones y activación:
    El perfil seleccionado es el de padre/madre y el usuario se encuentra en el menú principal.
- Garantías de éxito / Postcondición:
    Se inicia el contador de horas de juego y tras finalizar se bloquea la opción de jugar..
- Escenario principal:
    1. El usuario selecciona la opción de controles parentales.
    2. El usuario selecciona la opción de límite de tiempo de juego.
    3. El usuario introduce el número de horas deseado.
    4. El usuario selecciona la opción de aplicar.
    5. El ajuste se aplica.
- Escenarios alternativos:
    1. El usuario introduce un número de horas negativo y se muestra un mensaje de error.

CU17: Cambiar el PIN de control parental

- Contexto de uso:
    Desde el perfil de padre/madre se podrá cambiar el PIN del perfil.
- Precondiciones y activación:
    El perfil seleccionado es el de padre/madre y el usuario se encuentra en el menú principal.
- Garantías de éxito / Postcondición:
    El viejo PIN deja de funcionar y el nuevo PIN procede a hacerlo.
- Escenario principal:
    1. El usuario selecciona la opción de controles parentales.
    2. El usuario selecciona la opción de cambiar el PIN.
    3. El usuario introduce el viejo PIN en el campo de viejo PIN.
    4. El usuario introduce el nuevo PIN en el campo de nuevo PIN.
    5. El usuario selecciona la opción de aplicar.
    6. El cambio se aplica.
- Escenarios alternativos:
    1. El usuario introduce caracteres inválidos en el campo de nuevo PIN y se muestra un error.

CU18: Crear un perfil

- Contexto de uso:
  Desde el perfil de administrador se podrán crear nuevos perfiles.
- Precondiciones y activación:
  El perfil seleccionado es el de administrador y el usuario se encuentra en el menú principal.
- Garantías de éxito / Postcondición:
  Aparecen los nuevos perfiles creados en el listado de perfiles.
- Escenario principal:
  1. El administrador selecciona la opción de seleccionar perfiles.
  2. El administrador selecciona la opción de crear un perfil.
  3. El administrador selecciona el tipo de perfil (jugador, padre/madre o administrador) e introduce el nombre y salvo para el caso del jugador, una contraseña.
  4. El administrador selecciona la opción de crear el perfil.
  5. El nuevo perfil aparece en la lista de perfiles.
- Escenarios alternativos:
  1. El nombre introducido coincide con un perfil preexistente o la contraseña no es válida y se muestra un mensaje de error.

CU19: Borrar un perfil

- Contexto de uso:
  Desde el perfil de administrador se podrán borrar perfiles.
- Precondiciones y activación:
  El perfil seleccionado es el de administrador y el usuario se encuentra en el menú principal.
- Garantías de éxito / Postcondición:
  Los perfiles borrados desaparecen de la lista de perfiles.
- Escenario principal:
  1. El administrador selecciona la opción de seleccionar perfiles.
  2. El administrador selecciona un perfil de la lista de perfiles.
  3. El administrador selecciona la opción de borrar perfil.
  4. Se muestra una ventana de confirmación y el administrador confirma que desea borrar el perfil.
  5. El perfil borrado desaparece de la lista.
- Escenarios alternativos:
  1. El perfil seleccionado es el perfil del administrador activo y se muestra un mensaje de error ya que no se puede borrar el perfil activo.

CU20: Configurar un perfil

- Contexto de uso:
  Desde el perfil de administrador se podrán configurar perfiles.
- Precondiciones y activación:
  El perfil seleccionado es el de administrador y el usuario se encuentra en el menú principal.
- Garantías de éxito / Postcondición:
  Los cambios sobre los perfiles entran en vigor.
- Escenario principal:

1. El administrador selecciona la opción de seleccionar perfiles.
2. El administrador selecciona un perfil de la lista de perfiles.
3. El administrador selecciona la opción de configurar el perfil.
4. El administrador cambia los campos que desea configurar de entre los tres posibles: tipo, nombre y contraseña.
5. El administrador selecciona la opción de aplicar los cambios.
6. Los cambios se aplican.

- Escenarios alternativos:
    1. El administrador introduce un nombre inválido o que coincide con el de otro perfil o la contraseña contiene caracteres inválidos y se muestra un mensaje de error.

<u>Diagrama de casos de uso de Visual Paradigm:</u>

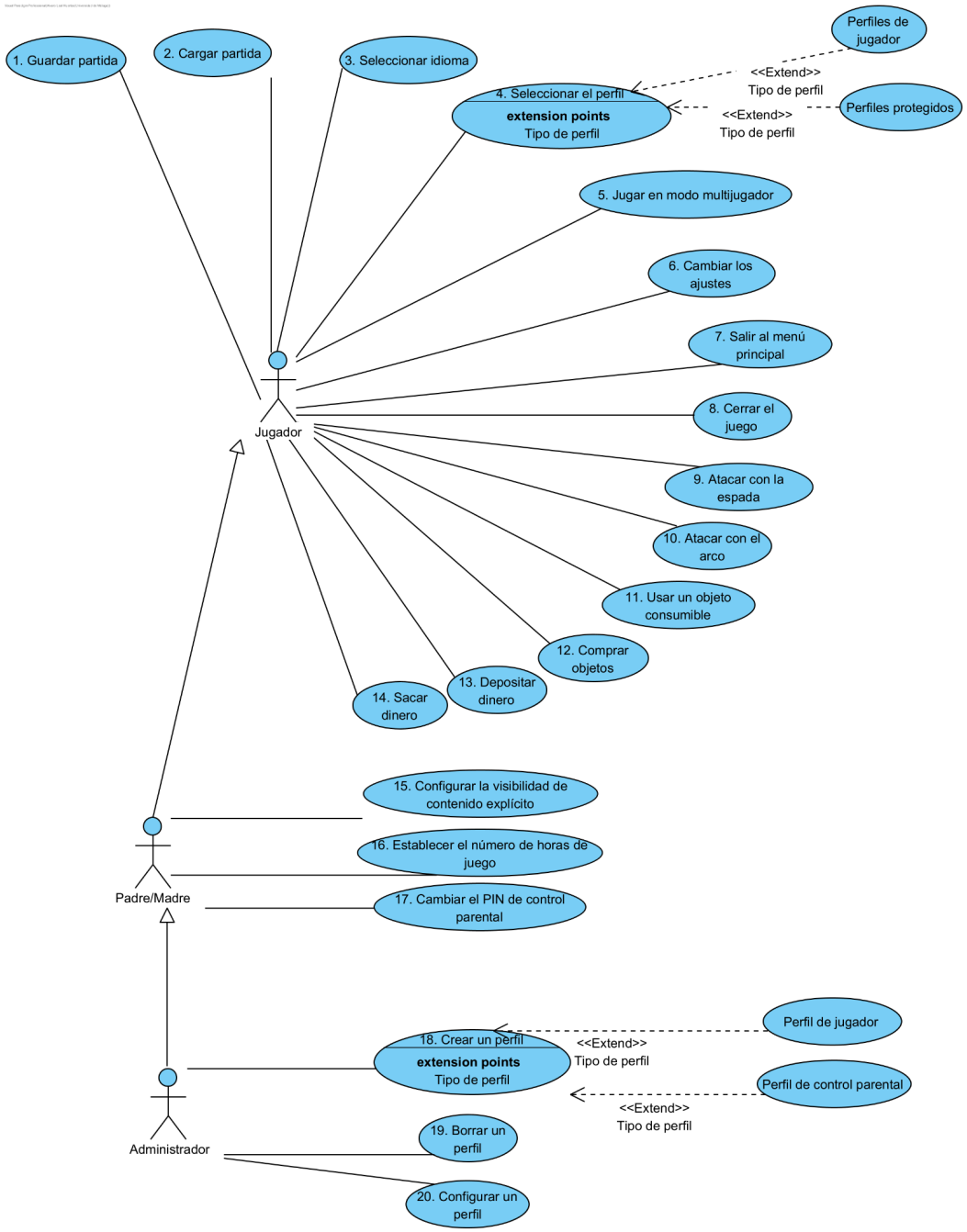


Diagrama de clases:

Partida:
Es la clase que se crea cuando comienza una partida. Contiene a las distintas entidades que conforman el juego (enemigos, jugadores y vendedores), así como el mapa. Mantiene el registro del tiempo de juego.

Entidad:
Entidad modela a enemigos, jugadores y demás elementos movibles del juego. Contiene sus coordenadas, así como métodos para detectar colisiones y ubicar a la entidad en el mapa de juego.

Enemigo:
Enemigo hereda de Entidad y modela a un enemigo en el juego. Contiene un nivel que sirve de multiplicador a sus estadísticas, unos puntos de vida y una velocidad, así como una instancia de la clase Comportamiento y una instancia de la clase Arma.

Jugador:
Jugador hereda de Entidad y modela a un jugador. Está formado por un dinero, unos puntos de vida y una velocidad, así como una instancia de la clase Arma y una lista de instancias de la clase Objeto o clases que hereden de esta.

Vendedor:
La clase Vendedor hereda de Entidad y modela un vendedor. Contiene una lista de instancias de la clase Objeto o subclases de esta.

Arma:
La clase Arma contiene una variable booleana que determina si es un arma de rango o melé, una variable entera que determina el poder de ataque y una velocidad que mide el tiempo entre ataques.

Comportamiento:
La clase Comportamiento define una serie de métodos que son llamados desde Enemigo y que definen el comportamiento de este. La instancia de la clase Comportamiento almacenada en Enemigo se crea en función del tipo de Arma que porta. Es decir, un enemigo con un arma a melé tendrá un comportamiento diferente a uno que porte un arma a distancia.

Menú:
La clase Menú contiene una etiqueta que se muestra en la cabecera del menú correspondiente y una lista de instancias de la clase Botón. Además, contiene un perfil según el cuál decidirá qué botones mostrar.

Botón:
La clase Botón contiene una etiqueta y unas coordenadas. Contiene métodos que llaman a las correspondientes funciones cuando el botón es activado.

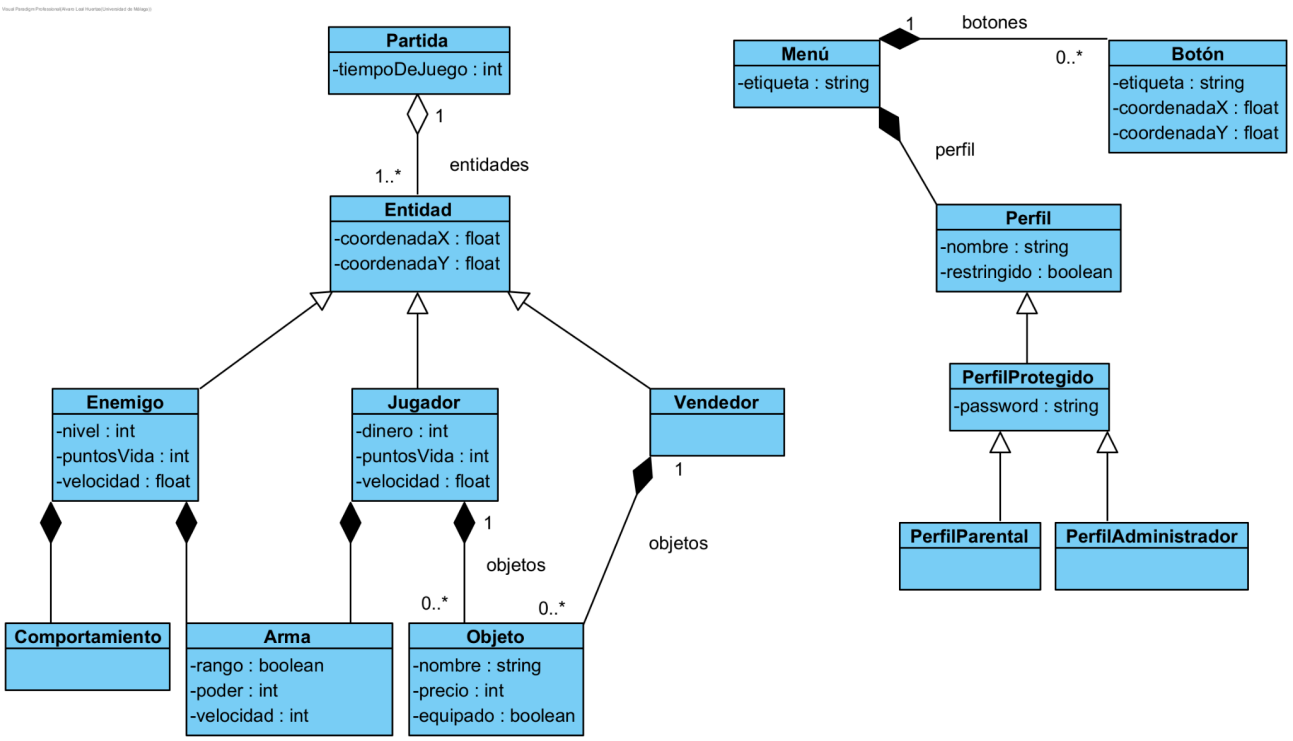
Perfil:
La clase perfil contiene un nombre y un valor booleano que indica si es un perfil restringido por controles parentales o no.

PerfilProtegido:
La clase PerfilProtegido hereda de perfil y añade un campo de contraseña.

PerfilParental:
La clase PerfilParental hereda de PerfilProtegido. La clase implementa métodos para configurar los controles parentales.

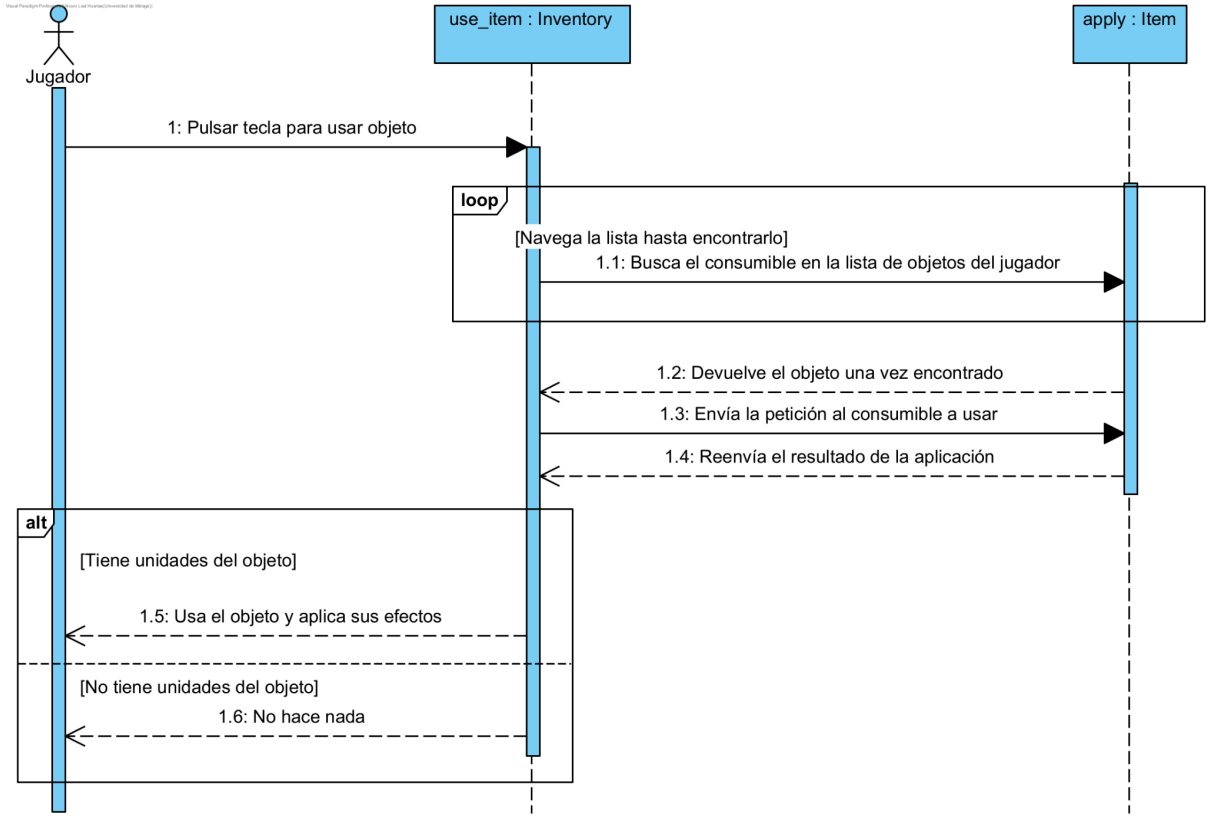
PerfilAdministrador:
La clase PerfilAdministrador hereda de PerfilProtegido. La clase implementa métodos para administrar usuarios.

Diagrama de clases de Visual Paradigm:



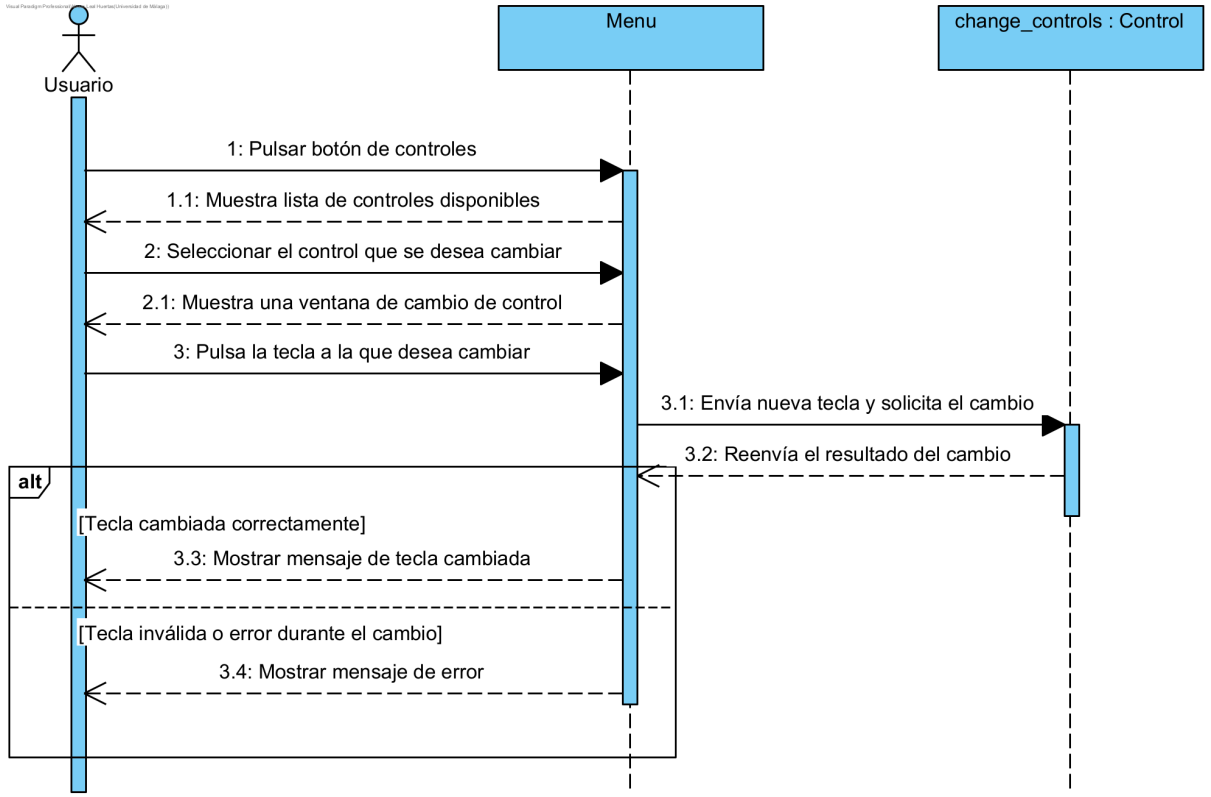
Diagramas de secuencia

Usar consumible



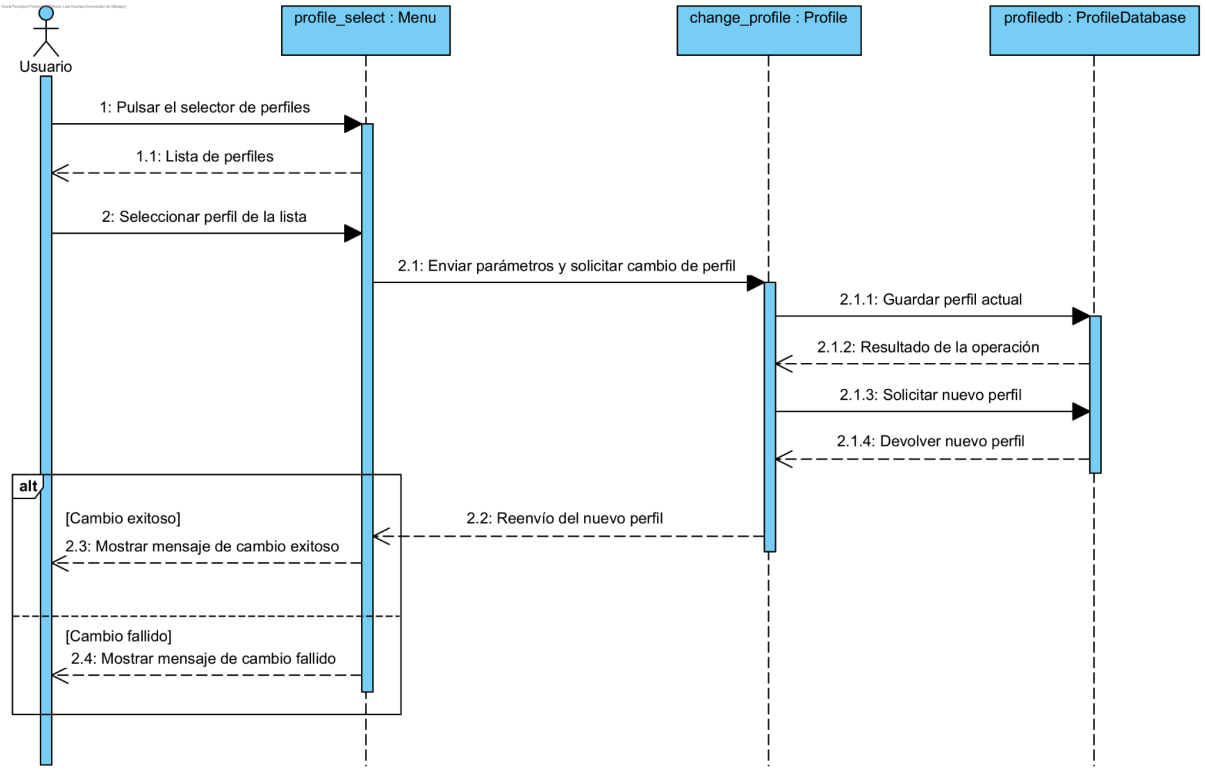
Al usar un consumible, se pulsa una tecla que comunica la intención de usar el objeto al juego. Posteriormente, se busca en un bucle el inventario hasta encontrar el objeto que se desea usar y se devuelve. Se consume el objeto y decrementa en 1 su número de usos y se aplican los efectos del objeto o, si no tiene usos disponibles, no hace nada.

Cambiar controles



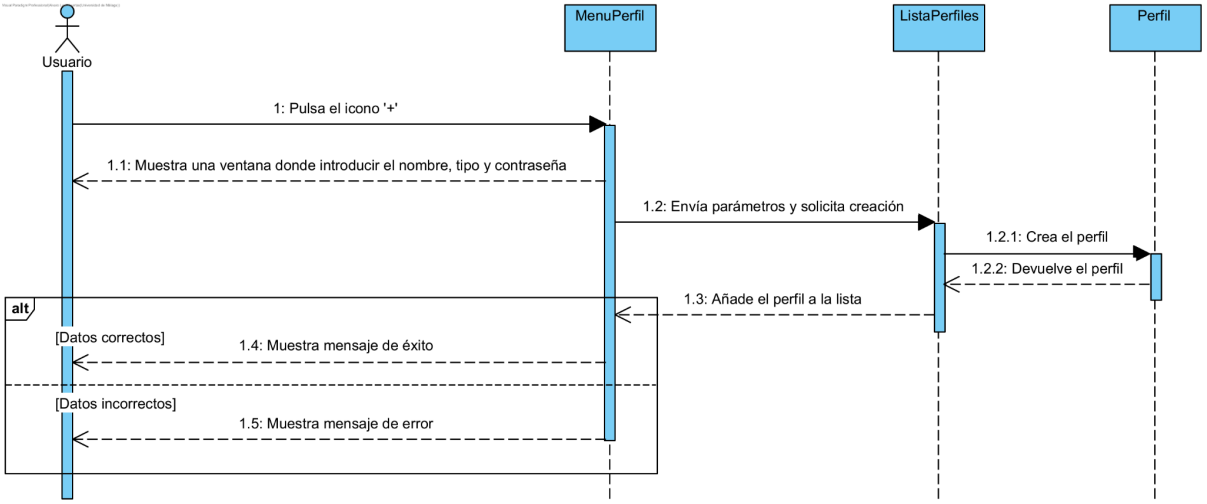
En el menú, pulsa el botón de controles. El juego muestra una lista de controles disponibles y el usuario selecciona el control a cambiar. Se abre una ventana de cambio de control que espera a una entrada de teclado. Cuando el usuario pulsa la tecla que se desea cambiar, se envía la solicitud de cambio a los controles y se recibe una respuesta. Según la respuesta, se muestra un mensaje de éxito o uno de fracaso.

Seleccionar perfil



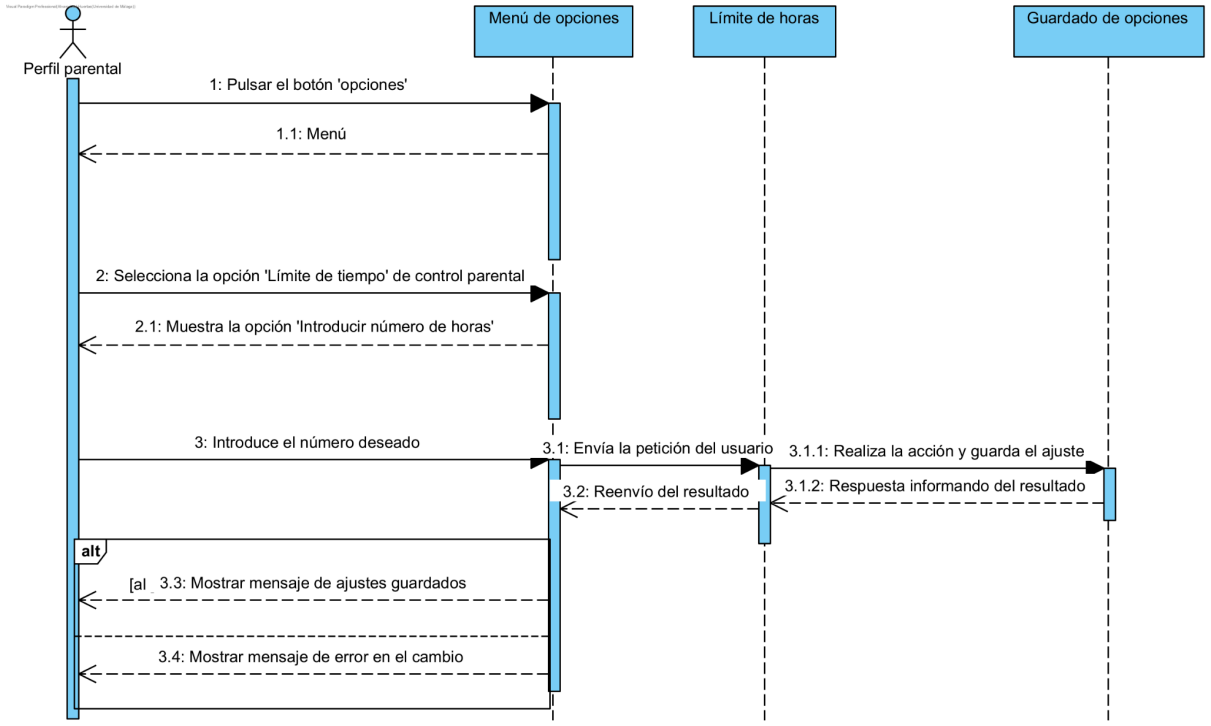
Se pulsa la opción de seleccionar perfil en el menú de opciones. Se muestra una lista de perfiles y se pide al usuario que seleccione uno. Al hacer esto, se envía una solicitud al perfil para que acceda al nuevo perfil y guarde el actual. Esta información se devuelve y según la respuesta se muestra un mensaje de cambio exitoso o fallido.

Crear perfil de administrador



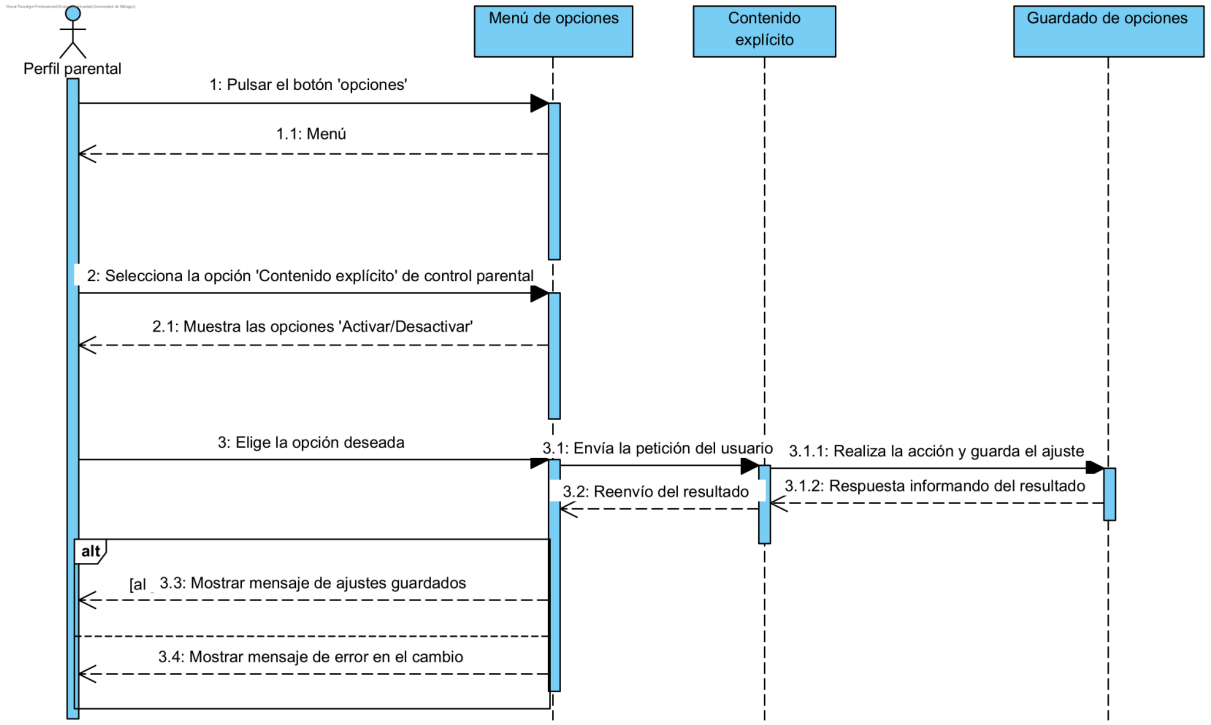
Se pulsa el icono ‘+’ en el selector de perfiles y se muestra una ventana donde introducir los datos del nuevo perfil. Se envía una solicitud a la lista de perfiles para añadir el nuevo perfil y esta solicita la creación del nuevo perfil. Así, añade el perfil a la lista y muestra en caso de éxito un mensaje de éxito y en caso contrario, un mensaje de error.

Configurar límite de horas



El perfil parental pulsa el botón de opciones y accede al menú. Selecciona la opción del límite de tiempo de juego y se muestra un submenú con un campo de texto. Introduce el número de horas deseado y se envía la petición a las opciones. Al recibir el resultado, se muestra un mensaje de error o éxito según el resultado de la operación.

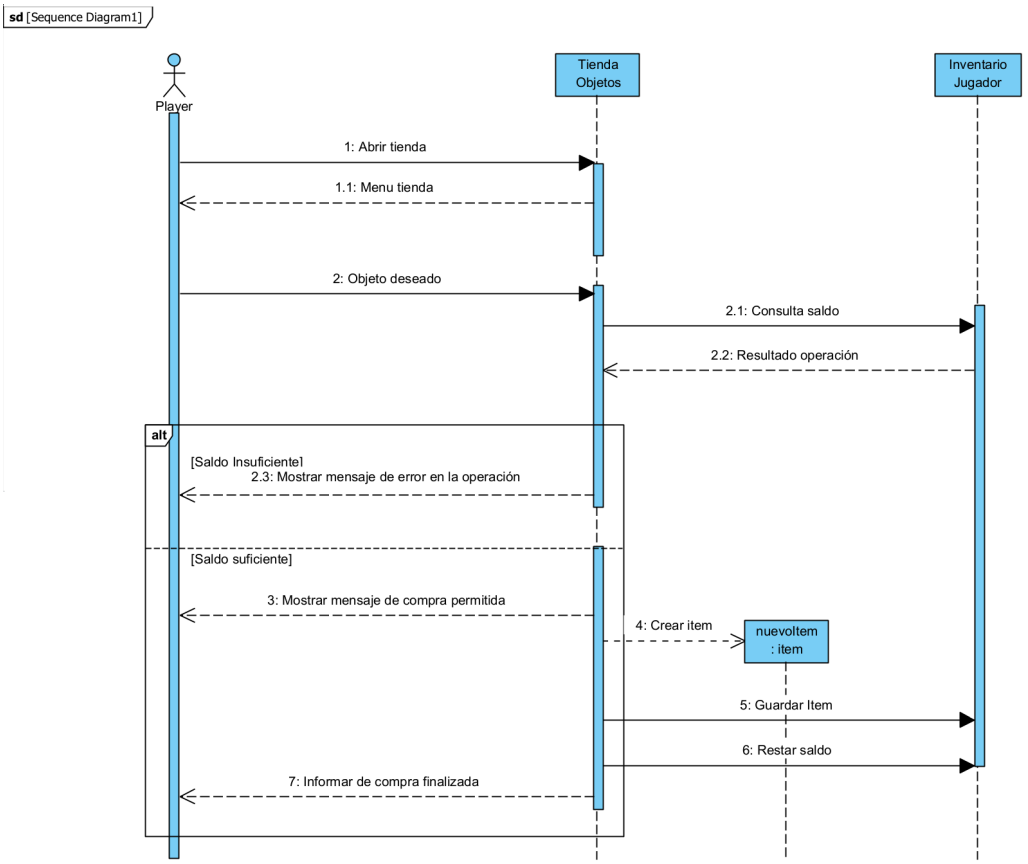
Configurar visibilidad de contenido explícito



El perfil parental pulsa el botón de opciones y accede al menú, posteriormente selecciona la opción de contenido explícito y se muestra la opción de activar o desactivar el ajuste.

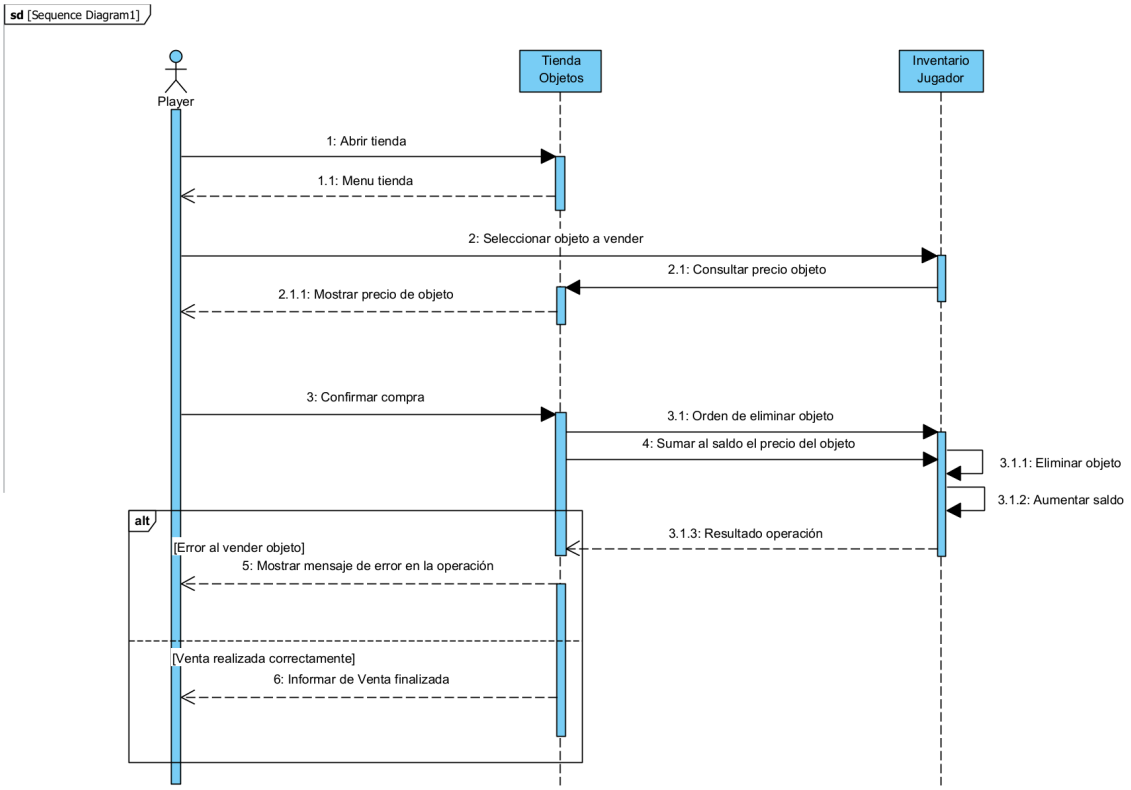
Selecciona la opción deseada, envía la petición y tras recibir el resultado, muestra un mensaje de error o éxito acorde al mismo.

Comprar un objeto



El jugador abre la tienda y accede al menú de la tienda. A continuación selecciona el objeto deseado y la tienda consulta el saldo del jugador y lo retorna. En caso de tener un saldo insuficiente, se muestra un mensaje de error y en caso contrario, se informa al jugador de que la compra se ha realizado con éxito y se crea un nuevo objeto que se añade al inventario del jugador. Finalmente, se informa de que la compra ha finalizado.

Vender un objeto



El jugador abre la tienda y accede al menú de la tienda. A continuación, selecciona el objeto a vender de su inventario y la tienda consulta el precio de dicho objeto, que es retornado. El jugador confirma la compra y el inventario elimina el objeto y suma el saldo correspondiente al jugador. Se retorna el resultado de la operación, de acuerdo al cual se muestra un mensaje de error o éxito en la operación.

Arquitectura de Software

La estructura de nuestro programa sigue las reglas y principios de la llamada y retorno orientada a objetos. La lógica de nuestro programa modela los elementos reales de nuestro juego (Player, Coin, Enemy, etc.), componentes que contienen un estado local y exponen ciertos métodos a otras clases. Como conectores utilizamos dos medios de comunicación: invocación de objetos y señales lanzadas ante eventos.

Esta arquitectura de software nos permite separar nuestro proyecto en módulos tangibles, que son escalables, fáciles de mantener y reutilizables.

Un ejemplo de esta reutilización de módulos está en las clases Player y Enemy: ambas modelan cosas diferentes pero reutilizan gran parte de su lógica como consecuencia de heredar de Entity, que contiene los métodos y estado comunes a ambas.

En cuanto a la comunicación entre clases, nuestro proyecto tiene una gran presencia de señales. Scripts como menu.gd y profile.gd (que son clases anónimas) efectúan la totalidad de su lógica con capturas de señales. También se hace uso de la instanciación de clases, ya sea para modelar un patrón de composición o herencia entre clases.

Un problema de esta arquitectura es que por lo general los datos no se manejan de la manera más eficiente cuando cada uno modela a una entidad en lugar de al conjunto. Un ejemplo es modelar una colección *Edades* como una lista de instancias de un objeto *Persona* que contiene una variable entera *edad*, en lugar de modelar una clase *Edades* que contiene una lista de enteros. La primera forma maneja referencias a objetos y el acceso es ineficiente, ya que no se aprovecha la capacidad de la CPU para manejar arrays. Este diseño menos eficiente es muy habitual en arquitecturas orientadas a objetos que tratan de modelar los problemas con entidades reales.

Sin embargo, esta arquitectura también tiene virtudes, en particular en lo que concierne al mantenimiento del software, siempre y cuando no se abuse de la herencia, que puede llevar a *coupling* excesivo.

Herramientas usadas en el desarrollo del software

- Godot: Es el motor gráfico elegido para desarrollar nuestro videojuego. Utiliza un sistema de escenas y objetos con scripts asociados, los cuales usan un lenguaje de scripting propio llamado GDScript. Es un motor de código abierto, bajo la licencia permisiva MIT.
- Git y Github: Hemos escogido Git como software de control de versiones de nuestro proyecto y el repositorio está hosteado en GitHub.
- Discord: Es el medio de comunicación informal que hemos usado cuando hemos necesitado hablar acerca del proyecto.
- Trello: Hemos usado Trello como forma de organizar nuestras tareas pendientes y en proceso.