

Programador de Videojuegos

CLASE 25/06/25



Índice de temas de la clase

- Etapas de diseño y desarrollo de un videojuego:
 - Concepción y definición
 - Diseño
 - Planificación
 - Producción
 - Pruebas

Etapas de diseño y desarrollo de un videojuego

Etapas de diseño y desarrollo de un videojuego

- Concepción y definición: Idea, objetivos, estilo y plataforma
- Diseño: Mecánicas, niveles, personajes, interfaz
- Planificación: Asignación de tareas, fechas, tecnología
- Producción: Programación, gráficos, sonidos, niveles
- Pruebas: Testeo de errores, jugabilidad y equilibrio

Opcional:

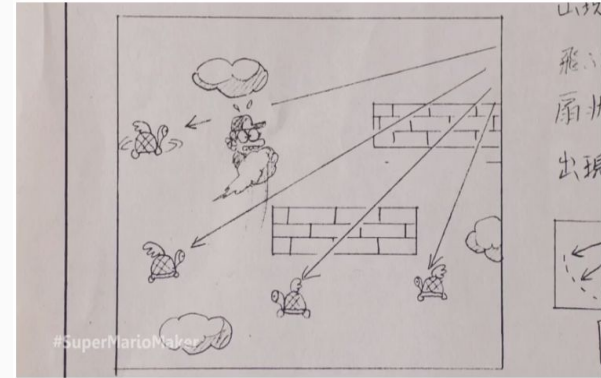
- Publicación: Distribución del juego final
- Mantenimiento: Correcciones, mejoras, actualizaciones

Concepción y definición

¿Qué se hace?

- Idea inicial: ¿Qué tipo o género de videojuego será? (plataformas, puzzle, shooter, etc.)
- Objetivo del juego: ¿Qué tiene que lograr el jugador?
- Game play: ¿Cómo será el proceso de juego? ¿Cuál es la mecánica central?
- Story board: ¿Qué historia se quiere contar? Estilo de personajes, ambiente, música, etc.
- Público objetivo: ¿Para quién está pensado el juego?
- Estilo visual y temático: ¿Será futurista, medieval, infantil?
- Plataforma: ¿Para qué plataformas se quiere lanzar? ¿PC, móvil, navegador?
- ¿Qué lo hace diferente o único?

Herramientas: En esta etapa es habitual realizar una lluvia de ideas para explotar la creatividad y encontrar la respuesta a los ítems anteriores.



Víñeta de un Storyboard de los primeros Super Mario

Rol	Función dentro de esta etapa
Game Designer	Define la idea, mecánicas generales y estilo de juego.
Product Owner / Productor	Ayuda a definir los objetivos del proyecto.
Director de arte	Colabora con el estilo visual inicial.

Ejemplo:

Crear un juego 2D de naves espaciales donde el jugador debe esquivar meteoritos y disparar a enemigos. Estilo retro arcade, para jugar en PC con teclado.

Diseño (Game Design)

¿Qué se hace?

Una vez se haya dado con una idea o concepto de videojuego viable, será momento de desarrollarla de una forma más profunda. Para ello se generará un documento de diseño en el que se profundizará en conceptos como la historia, los personajes, el arte, los sonidos, la música, las mecánicas del juego o la programación.

¿Qué se diseña?

- Mecánicas del juego: cómo se juega, controles, reglas.
- Niveles: ¿Cuántos hay? ¿Cómo se estructuran?
- Personajes, enemigos y objetos: habilidades, comportamiento.
- Interfaz (UI): menús, barras de vida, puntajes.
- Sonido y música: ambientación sonora.

Herramientas:

- Documentos de diseño (Game Design Document - GDD)
- Diagramas UML
- Bocetos (sketches) o prototipos en papel

Rol	Función dentro de esta etapa
Game Designer	Detalla las reglas y lógica del juego.
Narrative Designer	(si hay historia) crea diálogos, contexto, narrativa.
Diseñador de niveles	Diseña los escenarios, obstáculos y rutas del jugador.
Programador	Comienza a colaborar en la estructura técnica del diseño.

Planificación

¿Qué se hace?

Al acabar la fase de diseño del desarrollo de un videojuego se tiene una imagen clara de cómo debería ser el juego y todo lo que se necesitará para alcanzar dicho resultado. Por lo tanto, ahora será momento de planificar todas esas acciones y establecer objetivos realistas a medio y largo plazo.

¿Qué se organiza?

- Tareas: quién hace qué (gráficos, programación, música)
- Cronograma: fechas de entrega, hitos.
- Tecnología: motor de juego (ej: Unity, Godot, Pygame)
- Recursos necesarios: sonidos, sprites, tipografías, etc.

Ejemplo:

Se usará Pygame. El programador arma el sistema de colisiones.
Un diseñador crea los sprites. Se divide el proyecto en 4 semanas.

Rol	Función dentro de esta etapa
Productor / Scrum Master	Organiza tiempos, tareas y revisa el progreso.
Game Designer	Prioriza características del juego (backlog).
Líder técnico	Toma decisiones sobre el motor, lenguaje y herramientas.
Líder artístico	Planifica estilo gráfico, herramientas de arte.

Producción

Una vez se tiene claro lo que hay que hacer, cómo hacerlo, y se ha planificado el tiempo para llevarlo a cabo, entonces se empieza la producción con el objetivo de crear el juego, como mínimo en una versión inicial o prototipo a mejorar gradualmente.

¿Qué se hace?

- Programación de las mecánicas y lógica del juego.
- Diseño gráfico: creación de sprites, animaciones y fondos.
- Integración de sonido y música.
- Implementación de niveles, interfaz, puntaje, menú, etc.

 Es una etapa iterativa: se prueba y se mejora constantemente.

Ejemplo:

Se programa el movimiento de la nave, los enemigos que bajan, los disparos. Se agregan vidas, puntaje y fondo estrellado en movimiento.

Rol	Función dentro de esta etapa
Programadores	Codifican el comportamiento del juego.
Artistas gráficos	Dibujan e integran los elementos visuales.
Diseñador de sonido	Agrega efectos de sonido y música ambiental.
Diseñador de niveles	Construye escenarios jugables.
Game Designer	Supervisa que se mantenga la idea central del juego.

Pruebas (testing)

¿Qué se prueba?

- Errores (bugs): colisiones que fallan, movimientos incorrectos.
- Dificultad del juego: ¿es muy fácil o imposible?
- Interfaz: que los botones y menús funcionen correctamente.
- Compatibilidad: que funcione en todas las plataformas previstas.

Generalmente encontraremos dos tipos: las pruebas alpha, realizadas por un pequeño grupo de personas generalmente involucradas en el desarrollo, y las pruebas beta, realizadas por un equipo externo de jugadores. Las primeras tienen el objetivo de corregir defectos graves y mejorar características fundamentales no contempladas en el documento de diseño, mientras que las segundas se enfocan en detectar fallos menores y perfilar la experiencia de usuario, usando feedback de jugadores testers.

Ejemplo:

Se prueba el juego y nota que los disparos no afectan al enemigo. Se corrige el bug de colisión entre la bala y el enemigo.

Rol	Función dentro de esta etapa
Tester / QA	Prueba sistemáticamente el juego.
Programadores	Corrigen errores detectados.
Game Designer	Ajusta mecánicas para equilibrar dificultad o diversión.
Usuarios finales / jugadores	Dan feedback en pruebas abiertas o internas.

Publicación (opcional)

¿Qué se hace?

- Se compila el juego en el formato final (ejecutable, APK, HTML5).
- Se publica en plataformas: Itch.io, Steam, Play Store, etc.
- Se prepara documentación o tutorial para el usuario.

Mantenimiento / Actualizaciones (si aplica)

¿Qué se hace?

- Se corrigen errores reportados.
- Se agregan nuevos niveles, personajes o modos de juego.
- Se actualiza para nuevas versiones de sistema operativo.

La fase de mantenimiento es el momento de arreglar nuevos errores, mejorarlo, etc. Ésto se hace sacando parches o actualizaciones al mercado.

Etapas del diseño y desarrollo de un videojuego con metodologías ágiles

Etapas del diseño y desarrollo de un videojuego con metodologías ágiles

En lugar de hacer todo el diseño al principio y luego desarrollarlo completo al final (modelo en cascada), en metodologías ágiles se trabaja por iteraciones cortas, llamadas sprints, en las que el juego evoluciona paso a paso con entregas parciales que se van mejorando.

Concepción y visión del juego (Inicio del proyecto)

Objetivo:

Establecer la idea general del videojuego, sin entrar aún en detalles técnicos.

Actividades:

- Lluvia de ideas con el equipo
- Definir el género, estilo, y plataforma del juego
- Crear una breve descripción
- Identificar al jugador objetivo
- Redactar una visión del producto

Backlog del producto y diseño inicial

Objetivo:

Crear una lista priorizada de características (Product Backlog) y una base inicial de diseño para comenzar a trabajar.

Actividades:

- Listar las funcionalidades (user stories) en lenguaje simple:
 - "Como jugador, quiero moverme con el teclado"
 - "Como jugador, quiero que el enemigo me persiga"
- Definir MVP (Producto Mínimo Viable)
- Crear los primeros diagramas:
- Casos de uso (UML)
- Diagrama de clases básico
- Estimación de esfuerzo (por puntos, horas, etc.)

Iteraciones (Sprints)

Objetivo: Construir el videojuego por partes en ciclos cortos (sprints de 1 o 2 semanas). En cada sprint se agregan nuevas funcionalidades jugables.

En cada sprint se hace:

1. Planificación del sprint

El equipo selecciona del backlog qué funcionalidades se desarrollarán. Se define una meta del sprint (ej: movimiento + colisiones).

1. Diseño técnico y artístico

Se diseñan las clases, diagramas, sprites, pantallas necesarias solo para este sprint.

1. Desarrollo

Se programan las funcionalidades acordadas. Se usan tableros ágiles (como Trello, GitHub Projects o herramientas físicas) para seguir el progreso.

1. Pruebas y revisión

Se prueba el juego: jugabilidad, errores, interfaz. Se muestran los avances.

1. Retrospectiva

Se analiza qué funcionó bien y qué mejorar para el siguiente sprint.



Se repite este ciclo hasta terminar el proyecto.

Pruebas finales

Objetivo:

Asegurarse de que el juego funcione bien en su conjunto: sin errores, con dificultad balanceada y jugabilidad fluida.

Actividades:

- Pruebas cruzadas entre compañeros o testers externos
- Registro y corrección de bugs
- Pruebas de compatibilidad (pantallas, resoluciones, dispositivos)
- Ajuste de rendimiento y optimización

Entrega del producto final / Publicación

Objetivo:

Tener una versión jugable y presentable del videojuego, lista para publicar o mostrar.

Actividades:

- Se genera el archivo final (ejecutable, HTML5, APK)
- Preparar presentación y demo
- Se entrega todo el proyecto (código, documentación, arte)

Iteración post-lanzamiento (si aplica)

Objetivo:

- Hacer mejoras después de recibir feedback de jugadores o docentes.

Actividades:

- Corrección de errores detectados post-entrega
- Agregado de niveles, enemigos u opciones
- Actualización de gráficos o interfaz

Ejercitación

1) Concepción y definición -> Visión del juego

Supongamos que estamos en la etapa de “Concepción y definición”, en grupo se pide idear y redactar su visión inicial del juego respondiendo a las siguientes preguntas:

- ¿Qué tipo o género de videojuego es?
- ¿De qué trata? ¿Cuál es el objetivo?
- ¿Cómo será el proceso de juego? ¿Cuál es la mecánica central?
- ¿Qué historia se quiere contar? Detalle estilo de personajes, ambiente, música, etc.
- ¿Para qué público está dirigido?
- ¿Para qué plataformas se quiere lanzar?
- ¿Qué lo hace diferente o único?

Diseño

Game Design Document (GDD) - Documento de diseño

El Game Design Document es un documento colaborativo que contiene toda la información necesaria para diseñar, desarrollar y probar un videojuego.

Es el plano maestro del juego: lo usan diseñadores, programadores, artistas y testers.

¿Para qué sirve el GDD?

- Define con claridad cómo será el videojuego.
- Sirve como guía técnica y creativa para todo el equipo.
- Se usa para tomar decisiones de diseño sin perder la coherencia.
- Permite documentar cambios y mejoras durante el desarrollo.

Ejemplos de documentos de diseño de juegos (obtenidos de la página:

<https://www.nuclino.com/articles/game-design-document-template>) :

- Mike Dailly, uno de los diseñadores principales de Grand Theft Auto, originalmente llamado "Race'n'Chase", compartió los GDD : <https://www.gamedevs.org/uploads/grand-theft-auto.pdf>
- Brian Freyermuth compartió el GDD del cancelado Fallout: Brotherhood of Steel 2 : <https://drive.google.com/file/d/1b6TVJHAjtsK12gmDn9M8CdoUSHS1OHwY/view>
- GDD de Silent Hill 2: https://drive.google.com/file/d/1nxvdXasP-HsRCt62cHK3wF_plrJpYx5T/view

Game Design Document (GDD) - Estructura clásica

Sección	¿Qué incluye?
1. Visión del juego	Breve descripción del juego: género, experiencia deseada, inspiración, plataforma.
2. Mecánicas principales	¿Cómo se juega? Movimientos, objetivos, reglas básicas.
3. Personajes y enemigos	Qué personajes hay, habilidades, comportamiento de enemigos.
4. Historia (si aplica)	Narrativa básica, ambientación, inicio, conflictos.
5. Niveles y progresión	Cómo se avanza en el juego, dificultad, desbloques, niveles.
6. Interfaz de usuario (UI)	Menús, pantallas, barras, indicadores. Bocetos.
7. Gráficos y estilo visual	Estilo artístico, paleta de colores, sprites, animaciones.
8. Sonido y música	Efectos sonoros, música de fondo, estilo auditivo.
9. Controles e interacción	Cómo se juega (teclado, mouse, joystick), qué teclas hace qué.
10. Reglas y condiciones de juego	Puntos, vidas, condiciones de victoria o derrota.

UML - Lenguaje de Modelado Unificado

- UML (LENGUAJE UNIFICADO DE MODELADO) es un conjunto de diagramas que nos ayuda a representar cómo será nuestro sistema o videojuego. Sirve para visualizar, diseñar y organizar antes de programar.
- No es código: es un dibujo, un plano del juego.
- Cada diagrama en UML conduce al desarrollo de otro diagrama de UML. Teniendo así que a partir de la generación del diagrama de casos de uso, se pueden generar los escenarios de los casos de uso. Además, cada caso de uso podría generar un diagrama de actividades y cada escenario de caso de uso podría generar un diagrama de secuencias.
Y a su vez, los casos de uso y los diagramas de secuencias ayudan a determinar las clases. Los diagramas de clase se usan para representar la estructura del sistema, mostrando las características estáticas del sistema y las relaciones entre las clases.

¿Por qué usar UML en videojuegos?

- Para planificar personajes, enemigos, objetos
- Para organizar el código (clases y relaciones)
- Para que todos los integrantes del equipo entiendan cómo funciona el juego
- Para evitar errores antes de programar

UML - Lenguaje de Modelado Unificado

- Permiten una especificación completa de los requisitos y a una representación del diseño general del software a construir. Mediante el modelo de análisis, se obtiene la primera representación técnica de un sistema.
- Los diagramas permiten visualizar la construcción de un sistema orientado a objetos.
- El análisis y diseño orientado a objetos se centra en la definición de clases y en el modo en que colaboran una con otra para cumplir con los requerimientos del cliente.
- Características del diseño orientado a objetos:
 - Adecuadas para sistemas cambiantes y complejos.
 - Fácil para el rediseño.
 - Adaptación al cambio.
 - Fácil mantenimiento.

Diagrama de Caso de Uso

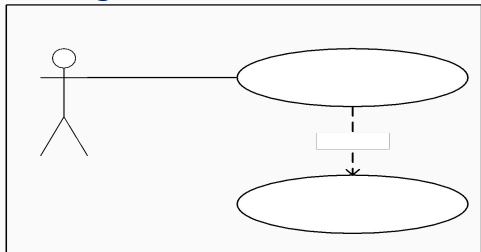
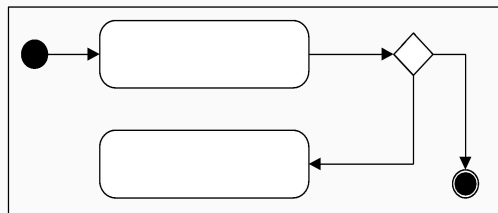


Diagrama de Actividades



Escenario del caso de uso

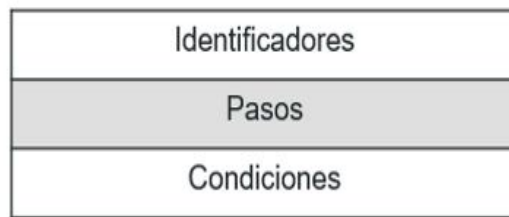


Diagrama de secuencias

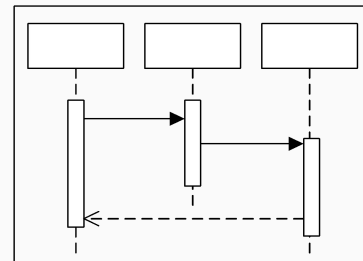
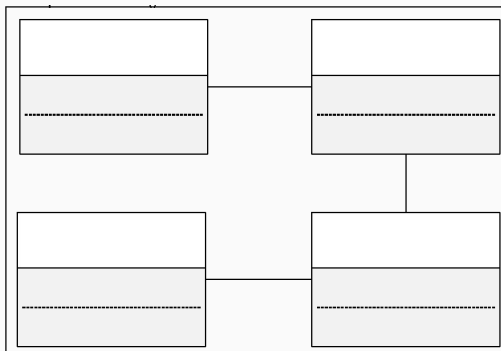


Diagrama de clases



1. Diagrama de casos de uso

Diagrama de casos de uso

Un diagrama de casos de uso es un tipo de diagrama UML (Lenguaje de Modelado Unificado) que representa las funcionalidades principales de un sistema desde el punto de vista de los usuarios externos (actores). Permite visualizar quién interactúa con el sistema y qué espera obtener de él. Describiendo así lo que hace un sistema sin describir cómo lo hace.

¿Para qué sirve?

- Capturar los requerimientos funcionales del sistema.
- Comunicar qué hace el sistema, sin entrar en detalles técnicos.
- Visualizar quién interactúa con el sistema y qué puede hacer.
- Facilitar la comunicación entre desarrolladores, usuarios y stakeholders.
- Servir como base para diseño posterior (diagramas de secuencia, clases, etc.).
- Ideal en etapas iniciales del desarrollo para definir el alcance.

IMPORTANTE! Representar sólo el qué, no el cómo.

Elementos: ACTOR

- Representa a un usuario o entidad externa (existe fuera del sistema) que interactúa con el sistema de una forma específica. Es decir, representar quién usa el sistema y desde qué rol.
- Puede ser:
 - Una persona (ej. Jugador, un Administrador)
 - Otro sistema o proceso externo (ej. Servidor de pagos)
- **Restricción:** No es parte del sistema; sólo se comunica con él.



Nombre actor

Elementos: CASO DE USO

- Es una función del sistema, una acción concreta que un actor puede realizar. Es decir, en un videojuego, son las acciones posibles dentro del juego que realiza cada actor.
- Se representa como un óvalo con el nombre de la acción dentro (en infinitivo).
- Ejemplos: Iniciar partida, Pausar juego, Guardar progreso, Saltar obstáculo, Evadir enemigo, Reiniciar juego, etc.
- **Restricción:** No debe detallar la lógica interna, solo lo que hace el sistema desde fuera.



Mover personaje

Ver puntaje

Elementos: SISTEMA

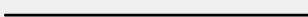
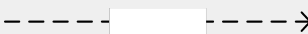
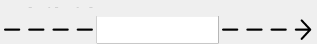
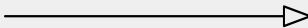
- Borde exterior del diagrama, se dibuja como un rectángulo grande que representa al sistema (en este caso, un videojuego).
- Es el marco que contiene todos los casos de uso. Indica qué está dentro del alcance del sistema (lo que el sistema ofrece).
- Todo lo que ocurre dentro es parte del videojuego.

Restricción: No se deben representar procesos internos o detalles técnicos.

Solo debe contener a los casos de usos y las relaciones. Los actores están por fuera del rectángulo que representa al sistema.

Elementos: RELACIONES

Relaciones entre actores y casos de uso

Relación	Símbolo	Significado
Comunica		Un actor se conecta a un caso de uso.
Incluye		Un caso de uso contiene un comportamiento que es más común que otro caso de uso. La flecha apunta al caso de uso común.
Extiende		Un caso de uso diferente maneja las excepciones del caso de uso básico. La flecha apunta desde el caso de uso extendido hacia el básico.
Generaliza		Una “cosa” de UML es más general que otra “cosa”. La flecha apunta a la “cosa” general.

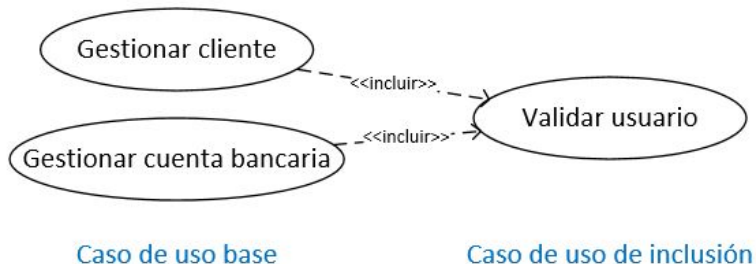
Incluye

Apunta al caso de uso a ser incluido.

Utilice relaciones de inclusión para comportamientos que se comparten entre dos o más casos de uso.



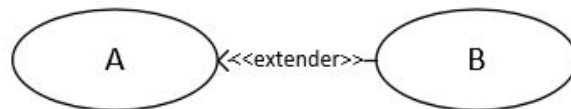
Ej.:



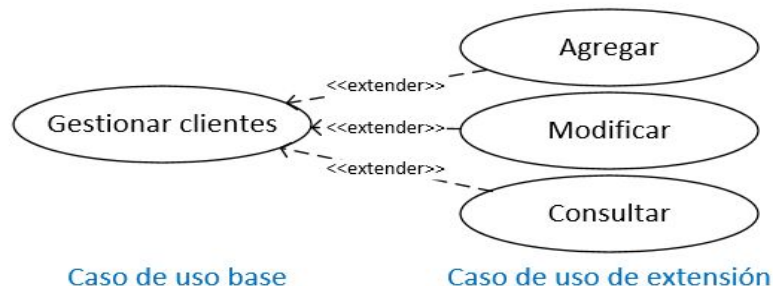
Extiende

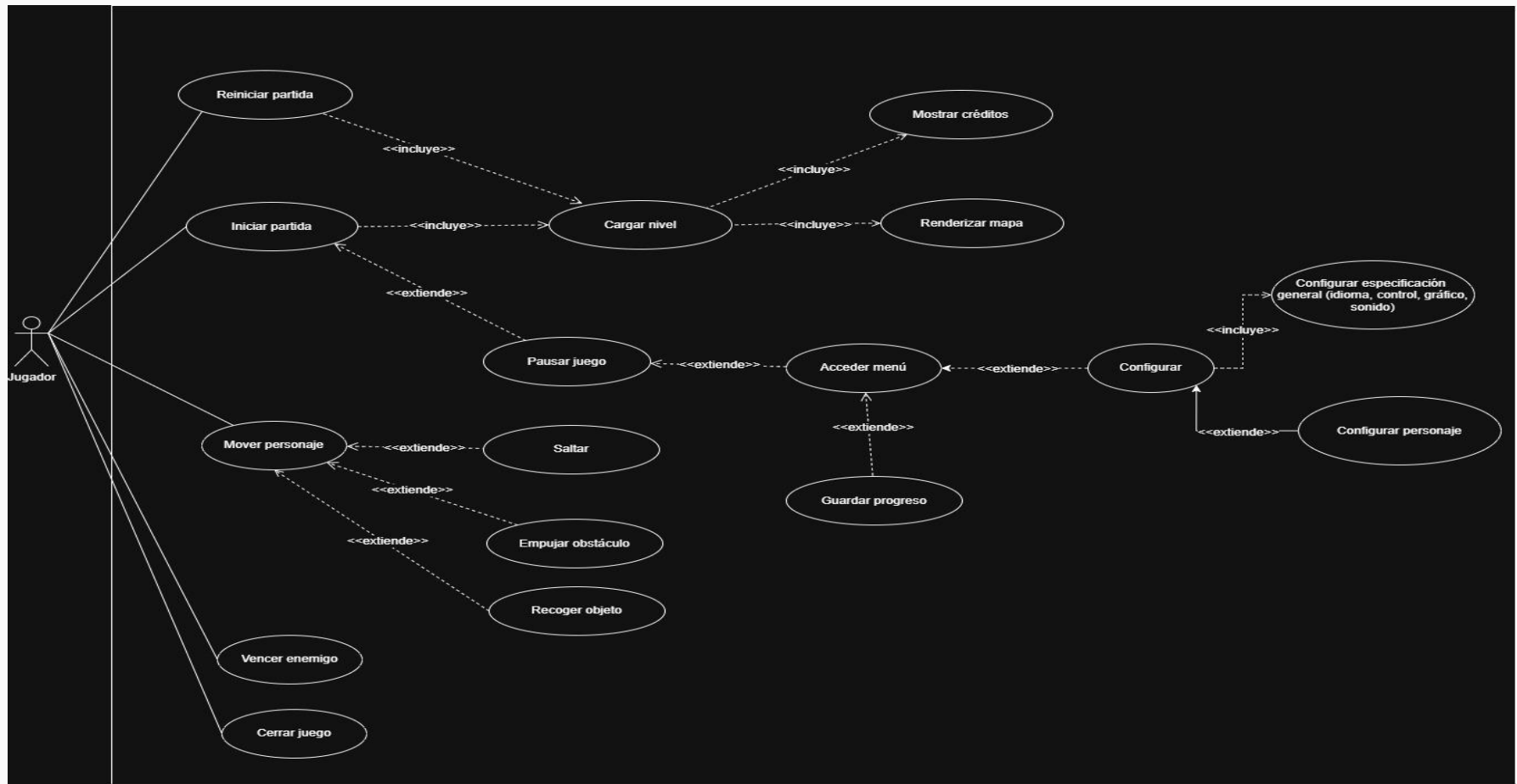
Apunta desde el caso de uso extendido hacia el básico .

Utilice la relación “extender” para comportamientos excepcionales, opcionales o que rara vez suceden.



Ej.:





2. Escenario de caso de uso

Escenario de caso de uso

Un escenario de caso de uso (aunque técnicamente no es un diagrama) es una articulación verbal de excepciones para el comportamiento principal descrito por el caso de uso principal.

Datos especificados en el escenario de caso de uso:

- El nombre del caso de uso para el cual lo realizamos
- Tenemos el área de aplicación o sistema al que pertenece este caso de uso
- el o los actores involucrados
- Una breve descripción de lo que logra el caso de uso
- La activación del evento, es decir lo que ocasionó que empezara el caso de uso
- Y el tipo de activación, pudiendo ser externa (si son empezados por un actor, ya sea una persona u otro sistema) o temporal (si lo activa el tiempo).
- Por otra parte, tenemos los pasos desempeñados y la información requerida para cada uno de los pasos, esto representa el flujo estándar de eventos, son los pasos para la realización exitosa del caso de uso. En donde tenemos que (leer pasos desempeñados).
- Por último, en la tercer área del caso de uso tenemos las precondiciones (que sería la condición del sistema antes de que se ejecute el caso de uso).
- Las postcondiciones, es decir el estado del sistema después de que el caso de uso terminó.

Caso de Uso	[Nombre del Caso de Uso]	Identificador: [Del caso de uso]
Actores	[Listado de los actores que tienen participación en el caso de uso]	
Tipo	[Tipo de caso de uso, primario, secundario, opcional]	
Referencias	[Requerimientos o funcionalidades incluidas en este caso de uso. Casos de uso relacionados.]	
Precondición	[Condiciones sobre el estado del sistema que deben cumplirse para iniciar el caso de uso]	
Postcondición	[Efectos inmediatos que tienen la ejecución del caso de uso sobre el estado del sistema]	
Descripción	[Descripción del caso de uso]	
Resumen	[Resumen de alto nivel del funcionamiento]	

Nro.	Ejecutor	Paso o Actividad
[Nro. de paso]	[Actor ejecutor o especifica si es el sistema o subsistema]	[Descripción del paso actividad ejecutado]

[Se describe el proceso o secuencia de pasos ejecutadas usando frases cortas]

[Cada paso del proceso puede ser ejecutado por los Actores o por el sistema]

[Se describe la secuencia de acciones realizadas por los actores y la secuencia de actividades realizada por el sistema como respuesta].

Curso
normal

Nro.	Descripción de acciones alternas
[Número de paso]	[Descripción de la secuencia de acciones alternas para el número de actividad indicado. Debe hacer referencia al número de paso en el curso normal]

[Cada paso descrito en el curso normal, puede tener actividades alternas, según la distribución de escenarios que ocurra en el flujo de procesos, en esta ficha se completa para cada actividad (haciendo referencia a su número) las posibles secuencias alternas]

Curso
alternativo

Nombre del caso de uso:		ID Única:	
Área:			
Actor(es):			
Descripción:			
Activar evento:			
Tipo de señal:	Interna/Externa		
Pasos desempeñados (ruta principal)		Información para los pasos	
1.			
2.			
3.			
N.			
Precondiciones:			
Postcondiciones:			
Suposiciones:			
Reunir requerimientos:			
Aspectos sobresalientes:			
Prioridad:			
Riesgo:			