Proyecto Formativo de Software

Evidencia

José Ignacio Botero Osorio

Instructor

Lyam Acosta Forero

Aprendiz

SENA – Servicio Nacional de Aprendizaje

Tecnología en análisis y Desarrollo de Software

3118526

Bogotá D.C.

Contenido

1. Introducción

A continuación, veremos el Proyecto de desarrollo de Software, un videojuego llamado “Juego Secreto” aquí veremos toda la creación del videojuego con los procesos básicos como caracterización de los sistemas, entre otros.

Durante la ejecución de este proyecto revisaremos los diferentes procesos de un software, como la ejecución de requisitos, la identificación de procesos de organización, también revisaremos algunos procesos en el software como los instrumentos de recolección de información los cuales nos guiaran en el proceso para la creación de nuestro videojuego.

También al momento de formular el proyecto crearemos documentos de especificación de requisitos, e informes de evaluación de los requerimientos del sistema esto por medio de infografías de metodologías y otros medios.

En este proyecto, revisaremos también en base a nuestros usuarios sus historias con nuestro Software con listas de chequeo estructuradas y con presupuestos para una propuesta económica sólida.

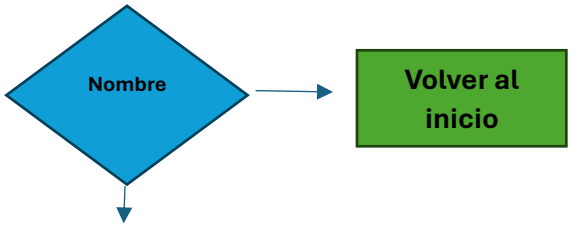
También revisaremos el Software en algunas secciones como Arquitectura del software, la validación del documento, los modelos de dominio del proyecto, y allí revisaremos un prototipo navegable donde también crearemos un interfaz web, modelos de bases de datos, revisando los estándares de codificación.

Por último, haremos el Front-end y el Back-end de nuestro videojuego crearemos los servicios web de nuestro juego, realizaremos pruebas y documentaremos todo lo relacionado al juego junto son sus plataformas tecnológicas, junto con la capacitación del usuario y un acta de entrega del juego, entre otros.

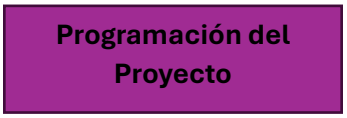
2. Procesos organizacionales

A continuación, veremos el proceso de desarrollo inicial del videojuego en su “Ideación” y cada uno de los procesos.

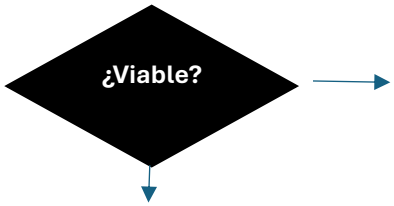
- Iniciamos con la idea inicial del videojuego.



- Decidimos el nombre del Juego y revisamos si existe, si existe lo replanteamos, si no podemos continuar



- Se toman algunas decisiones de diseño y se inicia la creación del proyecto en su programación.



- Se decide en que motor gráfico y de acuerdo con la toma de decisiones anteriores se revisa si es viable.

- En caso de ser viable, si inicia con el Proceso de identificación de errores y pruebas de calidad del proyecto, mecánicas y otros.

- Se inicia la producción de copias del contenido en sus formatos físico o Digitales una ve testeado el proyecto.

- Fin del proceso resumido

3. Ingeniería de requisitos

¿Qué es la ingeniería de Requisitos?

“El proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema de software es llamado Ingeniería de Requerimientos. La meta de la ingeniería de requerimientos es entregar una especificación de requerimientos de software correcta y completa. La ingeniería de requerimientos apunta a mejorar la forma en que comprendemos y definimos sistemas de software complejos. Existen varias definiciones de requerimientos”[1]

“La ingeniería de requisitos se define como el proceso de definición, documentación y mantenimiento de los requisitos. La disciplina incluye todas las técnicas, métodos y procedimientos relacionados con la definición y gestión de las necesidades de los usuarios relacionadas con el sistema en estudio”[2]

Fase 1

Se revisarán las actividades Claves para obtener los requisitos necesarios o funcionales

Fase 2

Se realizará unas clasificaciones de los requisitos de una categoría y se prioriza el requisito de los procesos e identificar cosas las cuales no sean importantes para el proceso

Fase 3

Aquí se documenta toda la información para realizar el proceso Revisar criterios importantes y evitar ambigüedades

Fase 4

Se revisan los Stakeholders y se revisan las necesidades del usuario en ello se revisa restricciones importantes como el tiempo de entrega o el presupuesto

Fase 5

Se realizan control de cambios en el momento de realizar el proceso con la trazabilidad de requisitos adicional se revisará el proceso y los procesos anteriores

Fase 6

Aquí se revisa los requisitos del negocio o necesidades estrategias de la organización

[1] https://sedici.unlp.edu.ar/bitstream/handle/10915/4057/2_-_Ingeniería_de_requerimientos.pdf?sequence=4

[2] https://visuresolutions.com/es /blog/requirementsengineering-process/

4. Recolección de información

La recolección de datos es un paso importante en el proceso de investigación.
El instrumento que elijas para recolectar los datos dependerá de los tipos de datos que pienses recolectar (cualitativos o cuantitativos) y de cómo pienses recolectarlos.

En la investigación se utilizan varios instrumentos para recopilar información:

- Entrevistas
- Observaciones
- Documentos de archivo y fuentes gubernamentales
- Experimentos de laboratorio
- Cuestionario de papel o cuestionarios online
- Focus groups presenciales or focus groups online
- Comunidades online"[3]

Usaremos entrevistas para el propósito de este proyecto formativo crearemos este formato de recolección de información donde verificaremos junto con el usuario si encontró problemas durante la jugabilidad inicial del prototipo, este formato lo usaremos en cada versión para que junto con QA podamos revisar cada versión y con ello evitar la máxima cantidad de Bugs posible en el Software aquí

Información General del Juego

- Nombre
- Género
- Plataformas en las que se lanzo
- Fecha de lanzamiento.
- Versión actual.

Mecánicas del Juego

- Objetivo principal del juego.
- Mecánicas
- ¿Es Single Player?
- Progresión del Juego

Aspectos Técnicos

- Motor de Desarrollo
- Lenguajes de programación.
- Requisitos del sistema.

[3] https://www.questionpro.com/blog/es/instrumentos-para-recopilar-informacion/

Diseño Visual y Sonoro

- Estilo gráfico
- Modelos
- Música de fondo y efectos de sonido.
- Diseño de niveles.

Contenido y Narrativa

- Historia o lore del juego.
- Personajes principales y secundarios.
- Escenarios, mundos o niveles.
- Sistema de misiones o eventos.

Experiencia del Usuario y Calidad

- Interfaz de usuario (UI/UX).
- Dificultad y curva de aprendizaje.
- Opiniones de los jugadores (reseñas, calificaciones).
- Frecuencia de errores reportados o bugs.
- Rendimiento en diferentes plataformas.

Herramienta N°	
Fecha	Código Formulario
Código empleado	
Identificación	
Nombre del Software	*Fecha de creación*
Personal a cargo	*Código*
Propósito del estudio	
Aquí se revisará el propósito del estudio ya se por marketing ante un nuevo software o la revisión de un software ya existente	
Técnico	
Lenguaje de programación usado	versión (En caso de que ya haya sido lanzado al público)
Usuario	
¿Cómo califica el software en una escala de 1 a 10?	¿Ha sufrido algún error usando el Software?
¿Ha tenido buenas experiencias con el software?	Si la respuesta anterior es no ¿Qué sucedió?
¿Siente que alguna característica en el Software es inútil para usted?	¿Cuál considera la característica más importante para usted?
¿Cuál sería una característica la cual usted desearía?	¿Siente que este software cumplió su necesidad?
¿Siente responsivo la navegación en el software?	¿Tras la última actualización ha sentido una mejora en el uso?
Observaciones o preguntas adicionales	

5. Formulación de Proyecto

5.1 Objetivo General

Crear y desarrollar un software de Videojuegos con la calidad que exige la industrial actualmente que sea sencillo y fácil para el usuario, también integrar modelos dinámicos para personas en condición de discapacidad conforme a las opciones de accesibilidad de las consolas y plataformas.

5.2 Objetivo Especifico

1- Análisis y desarrollo de un nuevo Software de Videojuegos
2- Crear un producto que muestre habilidades técnicas y creativas del equipo de desarrollo
3- Desarrollar las actividades para el desarrollo efectivo del software
4- Abarcar todo el proceso de desarrollo, tanto económico como del software.
5-

5.3 Alcance

Construir un videojuego con funcionalidades básicas con sus funcionalidades básicas e interfaz antes del 17 de febrero de 2027.

5.4 Captura de Requisitos

Para la captura de requisitos en el desarrollo de nuestro videojuego, nos enfocaremos en las historias de los usuarios y visualizar el flujo de un storyboard los cuales utilizaremos herramientas como:

1. Jira
2. Trello
3. Figma/Miro
4. Google Docs.

Después de revisar utilizaremos la opción de Jira para la gestión de historias de usuario y Miro para la elaboración de un Storyboard

“La captura de requisitos se refiere de dónde vienen los requisitos y como se realizará la obtención de estos, en este proceso el desarrollador por primera vez podrá tener una comprensión adecuada del problema que se requiere solucionar,

en esta etapa es importante la participación de todos los agentes mencionados en la lección anterior. Es la etapa más temprana en el ciclo de vida Proceso de vida que hace referencia a las etapas por las cuales pasa un desarrollo de software realizando una analogía del ciclo de la vida de una persona. del desarrollo del aplicativo web"[4]

5.5 Definición de Funcionalidades

Definimos funcionalidad utilizando una plantilla clásica de historias de usuarios utilizando las plantillas básicas de estas historias.

Ejemplos:

- Como Jugador quiero controlar mi progreso para poder continuar con tranquilidad cada nivel
- Como usuario quiero guardar mi progreso para después
- Como jugador quiero poder tener un HUD muy intuitivo y fácil de entender

Estas historias se pueden registrar en Jira como tarjetas y agrupar por módulos del videojuego.

5.6 Especificaciones de Captura de requisitos

La técnica aplicada será captura de mediante historias de usuarios, combinando sesiones de Coworking y elaboración de Storyboards en Miro el cual incluye:

1. Reunión con Stakeholders: identificación de necesidades.
2. Redacción colaborativa con las historias de usuarios.
3. Traducción visual de historias de usuario clave en Storyboards que muestren escenas importantes junto con flujos de juegos importantes.

5.7 Plantillas

En esta plantilla revisaremos algunas historias de usuario:

Como _____ (Tipo de usuarios) _____ Quiero ___(Funcionalidades)__ para _____(Objetivo)_____
Criterios de aceptación:

[4] https://contenidos.sucerman.com/nivel4/desarrollo/unidad1/leccion3.html

- Criterio 1
- Criterio 2

Notas Adicionales

{Observaciones, Ideas y demás}

4. **Escena:** Describe lo que sucede.
5. **Actores:** Quienes Participan.
6. **Interacción:** Que sucede o que reacción nos da.
7. **Resultado (Esperado):** Experimentación del usuario.

Escena	Actores	Interacción	Resultado
Pantalla de Presentación	Jugador/Usuario	Seleccionar una nueva partida	Inicia una nueva partida
Selección de personaje	Jugador/Usuario	Avatar para el juego	Personaje listo para Jugar
Tutorial	Jugador/QA	Superar los desafíos básicos	Aprender los controles del Juego

Para la formulación de este proyecto se fundamenta en la captura estructurada de requisitos usando historias de usuarios y Storyboards, asegurando claridad y movimientos entre equipos y los jugadores

5.8 Funcionalidades

Sumando cada historia, se pueden adjuntar modelos de Storyboards visualizando la experiencia en Miro, donde las funcionalidades del juego deben ser claramente listadas como resultado de la captura de requisitos combinando las historias de usuarios y paneles visuales

ID	Historia de Usuario	Priorización	Estado
Histo1	Como jugador quiero un avatar bonito y que resalte del ambiente	Media	Iniciando
Histo2	Como jugador quiero guardar y cargar una partida fácilmente	Alta	Desarrollo
Histo3	Como jugador quiero tutoriales para entender los controles	Media	En revisión
Histo4	Quiero poder conectarme en línea con mis amigos en línea dentro del juego	Baja	Definiendo

6. Formulario de recolección de información

6.1 Objetivo

Para esta sección revisaremos la técnica de recolección de información de entrevista para revisar que quieren los jugadores. Allí podremos revisar si existen o no algún conflicto con algún género o mecánica especifica y así obtendremos la información básica para poder desarrollar nuestro videojuego conforme a las necesidades reales de entretenimiento siguiendo los estándares de calidad de la industria

6.2 Herramienta

Realizando un total de 30 preguntas revisaremos los estándares básicos de los jugadores:

Encuesta de Sondeo de Gaming								
Preguntas	**Respuestas y Comentarios**							
¿Eres Jugador Casual?	Si	No						
¿Eres jugador Profesional?	Si	No						
¿Cuándo fue tu ultima partida?								
¿Cuántas horas tienes para jugar al dia?								
¿Juegas Juegos AAA?	Si	No						
¿Te gustan?	Si	No						
¿Juegas Juegos AA?	Si	No						
¿Te gustan?	Si	No						
¿Juegas Juegos Indie?	Si	No						
¿Te gustan?	Si	No						
¿Qué genero de Juegos te gusta?	Acción	Aventura	Plataformas	MOBA	Sigilo	Survival H	Deportes/Carreras	Otros ¿Cuál?:_____
¿Cuál es tu mecanica Favotira?	Gestion Recursos	Combate	Movimientos	Exploración	Puzzles	Creación	Resolucion Problemas	Otros ¿Cuál?:_____
¿Qué mecanica menos te gusta?	Acción	Aventura	Plataformas	MOBA	Sigilo	Survival H	Deportes/Carreras	Otros ¿Cuál?:_____
¿Qué genero de juego menos te gusta?	Gestion Recursos	Combate	Movimientos	Exploración	Puzzles	Creación	Resolucion Problemas	Otros ¿Cuál?:_____
¿Cuál fue el primer Juego que jugaste?								
¿Lo terminaste?								
¿Por qué dejaste tu ultimo Videojuego?								
¿Qué esperas de un videojuego?								
¿Qué clase de personaje te gustan?								
¿Qué tipo de personaje menos te gusta?								
¿Recuerdas la ultima compañía del juego que jugaste?	Si	No						
¿Por qué?								
¿Cuál es tu personaje de Videojuegos favorito?								
¿En que plataforma juegas?								
¿Has tenido siempre la misma plataforma?	Si	No						
¿Por qué la dejaste (Si la pregunta anterior fue si)?								
¿Tienes presupuestados videojuegos een tus gastos?	Si	No						
¿Cuánto presupuesto gastas al mes en juego?								
¿Te gustan mas los juegos Free to play o pagos?								
¿Te gustan mas los juegos 3D o 2D?	2D	3D						
¿Por qué te gustan? (3D o 2D)								
¿Qué sientes cuando juegas?								
¿Conoces juegos de tu pais?	Si	No						
¿Por qué?								

Ilustración 1 Entrevista a Jugadores

6.3 Análisis

Por medio de esta encuesta podremos revisar algún sondeo básico para nuestros clientes finales.

En la primera sección podremos revisar si el jugador es casual o profesional con ello podremos revisar nuestra base de jugadores, dado que si encontramos más una clase de jugador podríamos enfocar nuestro juego con dinámicas especiales para los jugadores.

En la segunda sección revisaremos si las personas desean más los juegos creados por grandes compañías.

En la tercera sección revisaremos si los jugadores les gustan más juegos con presupuestos moderados.

En la cuarta sección revisaremos si los jugadores les gusta nuestro modelo de videojuego, también en recientes estudios demuestran que los jugadores cada día le gustan más los juegos indies https://www.hd-tecnologia.com/hoy-en-dia-los-jugadores-estan-mas-interesados-en-juegos-indies-que-en-juegos-aaa/ sin embargo se hará esta revisión conforme a los jugadores colombianos.

En las siguientes secciones revisaremos si los jugadores colombianos tienen más tiempo para jugar videojuegos o si terminan los juegos o no.

Por último, podremos hacer un análisis claro de lo que espera un jugador, si tiene un videojuego favorito o le gustan más los videojuegos 3D o 2D, nos permite revisar si nuestro juego es interesante a los jugadores debido al aumento significativo de juegos 2D en la industria internacional.

7. Especificación de Requerimientos

7.1 Alcance

El videojuego será desarrollado para plataformas de consola (PlayStation, Xbox) y PC. El jugador controlará a un personaje principal en un entorno del cual hablamos en la introducción, con un enfoque en la exploración, la narrativa y las mecánicas para mostrarnos la historia de nuestro personaje principal el cual nos contará una historia entrelazada de su mundo. Durante nuestra aventura nos encontraremos diferentes mecánicas que vamos a revisar a continuación

7.2 Glosario

- **NPC (Non-Player Character):** Personajes que no puede usar directamente el jugador.
- **IA (Inteligencia Artificial):** Debido al que muchas veces el presupuesto es limitado usaremos la Inteligencia artificial como uso para los enemigos
- **HUD (Heads-Up Display):** Aquí podemos ver cosas como Mapas, y la vida del personaje entre otras cosas
- **FPS (Frames Per Second):** Si bien en cada videojuego es importante la cantidad de fotogramas que no muestra en pantalla en la industria de los videojuegos indie no es tan relevante
- **Indie:** Videojuego independiente con poco presupuesto

7.3 Visión General

Aquí revisaremos una descripción detallada de los requisitos funcionales y no funcionales del juego. Se detalla la jugabilidad, las características del personaje, la interacción del jugador con el mundo que lo rodea con Documento de diseño del videojuego y la referencia a los libros de arte y guía de narrativa.

¿Qué se espera de nuestro juego? El videojuego es una experiencia de mundo abierto en 2D con mecánicas lineales las cuales nos llevará la aventura dentro de su imaginación.

7.4 Funciones

- **Exploración**: Entendiendo los mundos que lleva este juego
- **Misiones:** Al entender este mundo el cual veremos el jugador podrá aceptar misionales para ir progresando en el juego que le queremos dar y con ello que el jugador pueda ir conectando.
- **Progresión del personaje:** El personaje podrá mejorar sus habilidades, adquirir nuevos poderes y equipo a medida que avanza en el juego.
- **Interacción con objetos:** Los objetos que encontremos a lo largo podrán generar un nuevo puzzle importante en la historia
- **Sistema de salud y recursos:** El jugador tendrá una barra de vida y recursos limitados que podrá gestionar a lo largo de la historia.

7.5 ¿Quién jugara nuestro juego?

- Para una mejor distribución del juego queremos llegar a la clasificación E + (Todos) según la clasificación PEGI a nivel mundial.
- Se estima un juego indie el cual esperamos que puedan jugar diferentes consumidores las cuales pueden llegar a ser jugadores profesionales o expertos en nuestra industria como personas que nunca hayan tocado un juego ya que esperamos poder sacar nuestro videojuego en consolas, PC y teléfonos

7.6 Restricciones

- El juego debe ser compatible por ello es importante revisar un motor gráfico compatible para la actualidad
- Esperamos tener un modelo estable de 30 FPs sin embargo en la parte de arriba aclaramos que en el mundo indie esto no es tan importante

7.7 Suposiciones y dependencias

- Se asume que los jugadores tendrán acceso a Internet para descargar actualizaciones y contenido adicional.
- El desarrollo dependerá del uso del motor gráfico que se adapte a varios modelos de consolas y teléfonos.

8. Evaluación de los requerimientos

Para nuestro videojuego tenemos que realizar una validación de algunos requerimientos clave para el videojuego y la industria debido a que este documento nos puede ayudar a identificar el uso innecesario de recursos y tiempo para ejecutar correctamente le desarrollo.

“Este estándar reemplaza los estándares IEEE 830, IEE 1233, IEEE 1362, y contiene disposiciones para los procesos y productos relacionados con la ingeniería de requisitos para sistemas, productos y servicios de software a lo largo del ciclo de vida (Penzenstadler, 2021).

Además, define la construcción de un buen requisito, proporciona atributos y características de los requisitos, y analiza la aplicación iterativa y recursiva del proceso de requisitos a lo largo del ciclo de vida. También proporciona orientación adicional en la aplicación de procesos de ingeniería y gestión de requerimientos relacionados con la ingeniería de requisitos al tiempo que define los elementos de información aplicables a la ingeniería de requisitos y su contenido”[5]

8.1 Casos

8.1.1 Objetivo del caso de prueba.

Verificar que el jugador pueda pausar y reanudar completamente

8.1.2 Identificador

8.1.3 Nombre del requerimiento o historia de usuario asociada.

Hisu1: Como jugador quiero poder pausar el juego a voluntad dado que es un Single Player y poder volver a la partida cuando quiera

[5] https://zajuna.sena.edu.co/Repositorio/Titulada/institution/SENA/Tecnologia/228118/Contenido/OVA/CF4/index.html#/curso/tema2

8.1.4 Precondiciones

- Se necesita que el juego este en una partida activa
- El botón de pausa funciona en el hardware

8.1.5 Lista de pasos con los resultados esperados

Caso de uso	
Hisu1	**Nombre de la historia de usuario: Pausa**
Usuario: Tester QA	
Prioridad: Alta	
Descripción: Durante una partida revisar que funcione el botón de pausa	
Observaciones: El juego se tiene que detener y quede con el menú básico de pausa	
Criterios de aceptación: Al revisar que si funcione el botón de pausa se debe revisar correctamente la información entregada y con ello poder aprobar el menú	
Caso de uso	
Hisu1	**Nombre de la historia de usuario: Reanudar**
Usuario: Tester QA	
Prioridad: Alta	
Descripción: Durante la pausa revisar que el botón "Reanudar" Funcione	
Observaciones: Mientras el juego está detenido, al momento de dar clic en reanudar o solo el boto de pausa, este vuelva a funcionar	
Criterios de aceptación: Durante la revisión se espera que el botón de "Reanudar" o el momento de pausa reactive el juego	

8.6.1 Objetivo del caso de prueba

Verificar que el jugador pueda guardar el progreso correctamente

8.2.1 Identificador

8.2.2 Nombre del requerimiento o historia de usuario asociada.

Hisu2: Como jugador quiero poder guardar la partida en todo momento esto debido que posiblemente me tenga que retirar del juego en medio de una partida

8.2.3 Precondiciones

- Se necesita que el juego este en una partida activa
- Se habilite un sistema de guardado de partida

8.2.4 Lista de pasos con los resultados esperados

Caso de uso	
Hisu 2	Nombre de la historia de usuario: Guardar
Usuario: Tester QA	
Prioridad: Alta	
Descripción: Uso del botón "Guardar" y guardado automático	
Observaciones: Durante la partida se crearán puntos de control de guardado automático sin embargo adicional se debe crear un botón en el menú de pausa para guardar	
Criterios de aceptación: En este punto el menú de pausa funciona perfectamente y se revisa que dentro de dicho menú funcione el guardar partida	

Caso de uso	
Hisu 2	Nombre de la historia de usuario: Cargar
Usuario: Tester QA	
Prioridad: Alta	
Descripción: Uso del botón "Cargar"	
Observaciones: Se creará un botón en el menú que mostrará las partidas guardadas para que el jugador pueda variar entre partidas	
Criterios de aceptación: En este punto el menú de pausa funciona perfectamente y también el botón "Cargar" mostrara las partidas guardadas	

8.2.5 Objetivo del caso de prueba

Verificar que el HUD del juego

8.3.1 Identificador

8.3.2 Nombre del requerimiento o historia de usuario asociada.

Hisu3: Como jugador deseo un HUD intuitivo con información básica para poder progresar en el juego

8.3.3 Precondiciones

- Se necesita un mundo creado
- Tener los menús ya funcionando
- Tener definidos los controles y mecánicas

8.3.4 Lista de pasos con los resultados esperados

Caso de uso	
Hisu 3	**Nombre de la historia de usuario: HUD**
Usuario: Tester QA	
Prioridad: Media	
Descripción: Revisión del HUD del juego	
Observaciones: Durante la partida saldrán unos pequeños carteles con información básica del juego	
Criterios de aceptación: Aquí ya deberemos tener la información clara de mecánicas y un mundo ya creado, por lo cual se procede a revisar la funcionalidad de los carteles en momentos claves del Gameplay	

8.3.5 Objetivo del caso de prueba

Verificar el progreso entre mundos

8.4.1 Identificador

8.4.2 Nombre del requerimiento o historia de usuario asociada.

Hisu4: Como jugador deseo revisar el progreso en cada nivel y si olvide algo durante el mismo

8.4.3 Precondiciones

- Se necesita que el juego este en una partida activa

- Se debe tener claro el botón el cual abrirá el menú de progreso
- Se debe tener claro la cantidad de niveles y coleccionables o cosas dentro del mismo

8.4.4 Lista de pasos con los resultados esperados

Caso de uso	
Hisu 4	**Nombre de la historia de usuario: Progreso**
Usuario: Tester QA	
Prioridad: Baja	
Descripción: Tener un menú de progreso especializado	
Observaciones: Al momento de apretar un botón saldrá una pantalla con información básica del nivel y su progreso en el mismo	
Criterios de aceptación: Funcionalidad completa del menú “Progreso”	

9. Metodologías de desarrollo de software

Al revisar las metodologías de los clientes deberemos tener en cuenta varios aspectos:

Ilustración 2 Prototipo https://www.redalyc.org/journal/3783/378366538003/html/

“Con frecuencia recibimos en Innevo la pregunta de cuál es la mejor metodología para desarrollar software. Siempre respondemos: la mejor metodología es la que utilices tal y como está definida, no parcialmente. Cada una tiene diferencias más bien sutiles que profundas porque todas contemplan el Ciclo de Vida de Desarrollo, que básicamente consiste en las siguientes etapas:

- Levantamiento y aprobación de requerimientos funcionales.
- Análisis y diseño (requerimientos funcionales, no funcionales y diseño de pruebas).
- Codificación.
- Pruebas (funcionales, no funcionales y de eficiencia de código).
- Liberación (pruebas piloto, validación final y capacitación a usuarios).
- Despliegue (puesta en producción).”[6]

[6] : https://innevo.com/blog/metodologias-desarrollo-software

Ilustración 3 Metodología Desarrollo de Software 1

[7]

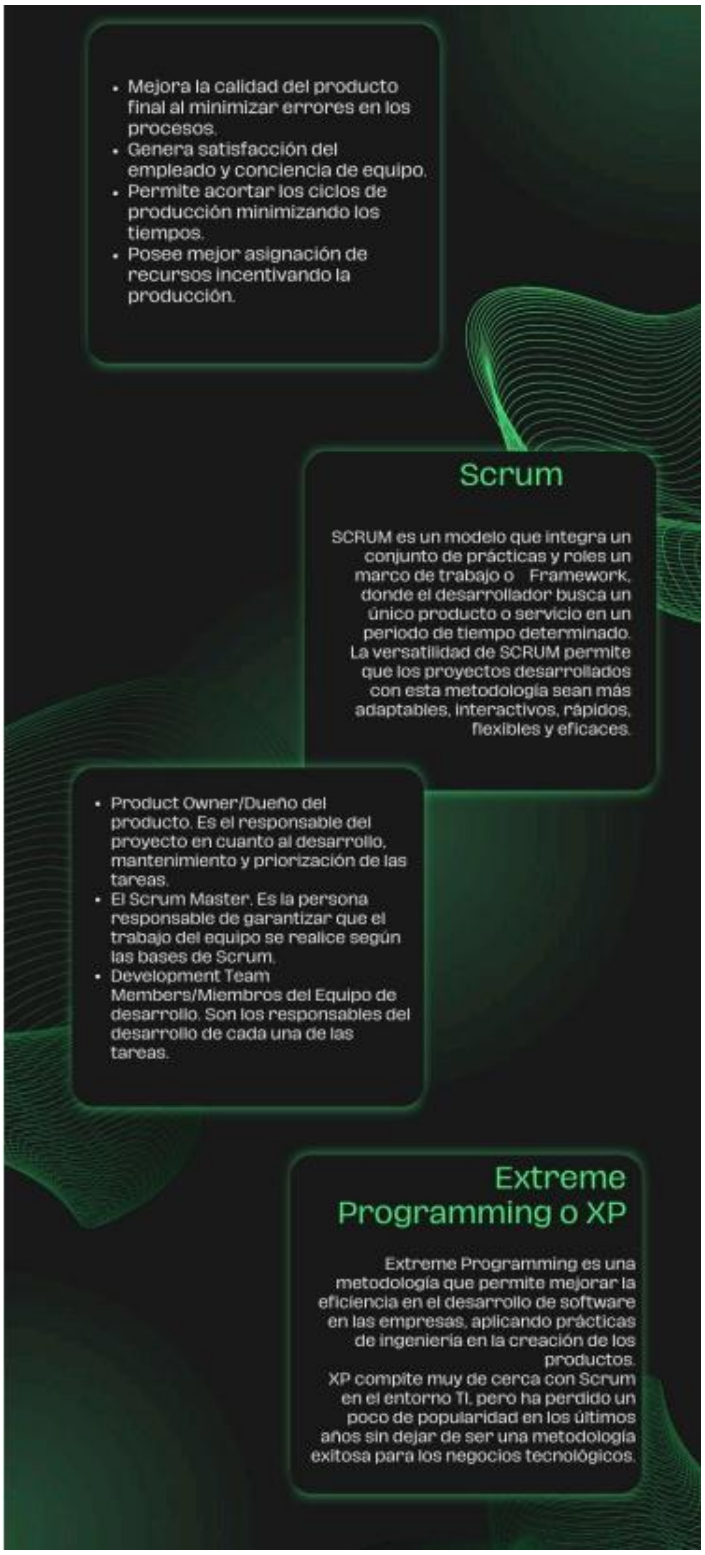


Ilustración 4 Metodología Desarrollo de Software 2

[7]https://www.valtx.pe/blog/metodologias-para-el-desarrollo-de-software-que-son-y-para-que-sirven

Ilustración 5 Metodología Desarrollo de Software 3

10. Metodología para el proyecto

En esta sección revisaremos la metodología correcta de nuestro videojuego y revisaremos cuales son las metodologías de un proyecto para revisar cual es la más útil para nuestro videojuego:

- Ágil.
- Modelo de cascada.
- **Metodología** Scrum.
- **Metodología** Kanban.
- **Metodología** Scrumban.
- **Metodología** PRINCE2.
- **Metodología** Six Sigma.
- Método de la ruta crítica (CPM)[8]

Nuestro desarrollo se usaremos Agile basado en Scrum o sus variantes

“Los equipos ágiles tienen un "por qué" sólido detrás de lo que hacen y una claridad en cuanto a cómo lo hacen. Los principios de la metodología ágil ayudan a los equipos a dividir objetivos enormes y ambiciosos en partes de trabajo gestionables que puedan cumplir de forma coherente. Los desarrolladores ágiles se ven fortalecidos por innumerables historias de equipos pequeños y ágiles que superan a los grandes competidores que utilizan la entrega en cascada. Los equipos ágiles también se benefician del "complejo industrial ágil". Hay una gran cantidad de recursos y herramientas para quienes necesitan conocer la metodología ágil y todo un ejército de consultores deseosos de ayudar con su implementación.”[9]

[8] https://asana.com/es/resources/project-management-methodologies
[9] https://www.atlassian.com/es/agile/scrum/agile-vs-scrum

“Para lograr una gestión eficiente de proyectos utilizando enfoques ágiles, es fundamental adoptar ciertas características y prácticas clave.

- Enfoque centrado en el cliente: El cliente es el eje central del proceso y siempre debe existir una comunicación constante con él para entender y adaptarse a las necesidades que le vayan surgiendo.
- La colaboración y el trabajo en equipo: Promover la colaboración y una comunicación abierta entre todos los integrantes del equipo del proyecto contribuye a generar un ambiente de trabajo cooperativo donde no existan jerarquías rígidas. Esto posibilita que el equipo tome decisiones de manera conjunta y resuelva eficientemente los problemas.
- Llevar a cabo entregas incrementales de valor y continuas, en lugar de intentar desarrollar el producto completo de una sola vez. Al trabajar en etapas pequeñas y manejables, se obtiene retroalimentación temprana que facilita realizar ajustes durante el proceso.
- Mantener la transparencia y visibilidad en cuanto al progreso del proyecto, de esta manera, todos los miembros del equipo y las partes interesadas estarán informados y alineados con los objetivos y la dirección del proyecto.

Finalmente, utilizar herramientas como DoneTonic para plasmar los tableros como los tableros Kanban o el Sprint Backlog, y así visualizar y gestionar el flujo de trabajo del proyecto, identificando cuellos de botella y áreas de mejora, contribuye a una gestión más efectiva y adaptativa.”[10]

[10] https://donetonic.com/es/agile-y-scrum/

Ilustración 6 Scrum
Tomado de: https://donetonic.com/es/agile-y-scrum/

10.1 Justificación

Decidimos usar este método por:

- **Alta variabilidad y cambios frecuentes**: Los videojuegos muchas veces experimentan cambios más actualmente con los DLC, los parches incluso mecánicas, debido a nuevas funciones o feedback de los usuarios
    - Por eso esta metodología permiten adaptarse a los cambios de manera más eficaz, ya que trabaja en varios ciclos cortos y se revisan las actividades más importantes

- **Entrega temprana:** Agil permite presentar versiones o prototipos desde la primera etapa, adicionalmente podemos tener feedback de inmediato y realizar los ajustes necesarios, por ellos en los videojuegos donde estamos evaluando en todo momento la experiencia y la jugabilidad de niveles, mecánicas y demás, es mejor siempre con prototipos especiales de nuevas funciones o las funciones actuales.

- **Colaboración en varias disciplinas:** Un videojuego muchas veces depende la colaboración de los jugadores, por eso las reuniones con estas metodologías garantizan comunicaciones constantes y más efectivas dentro del equipo, lo que significa que podemos evitar procesos innecesarios o problemas creativos.

- **Enfoque en lo más importante:** Esta metodología permite identificar las funcionalidad o mecánicas básicas de nuestro videojuego las cuales deben tener mayor prioridad en el desarrollo, así evitando tener tareas innecesarias y asegurando que las cosas claves tengan más prioridad para las pruebas

- **Gestiones más eficientes:** Los desarrolladores de videojuegos indie como el nuestro muchas veces no pasa de 20 personas por ello en todo momento tenemos dificultades de conocimiento, tiempo o presupuesto, por esto esta metodología permite el control de avance de nuestro videojuego ya sea que toque darles prioridad a nuevas tareas y así sacar el proyecto más rápidamente.

11. Software y servicios de internet

La tecnología de software y los servicios de internet son fundamentales en la comunicación global y el acceso a información diversa. Las plataformas como redes sociales, blogs, correos electrónicos y foros de discusión permiten la interacción entre usuarios de todo el mundo.[11]

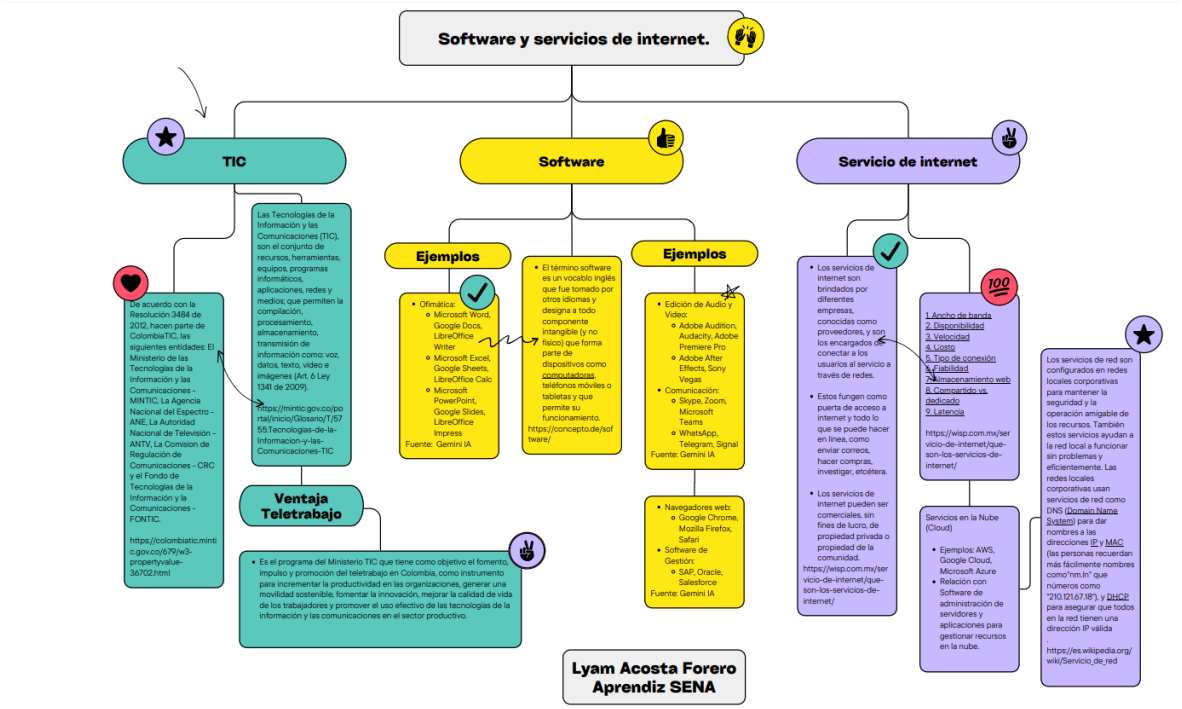


Ilustración 7: Servicios de Internet (Ilustración Propia)

[11] https://www.mindomo.com/es/mind-maps/software-y-servicios-de-internet-a8452fbd8a7c532964b3f20f1603dd6a

12. Mejora de productos y procesos con la incorporación de TIC

En nuestro videojuego podemos revisar las opciones TIC actuales para poder disminuir costos o tiempos de desarrollo

“Las Tecnologías de la Información y las Comunicaciones (TIC), son el conjunto de recursos, herramientas, equipos, programas informáticos, aplicaciones, redes y medios; que permiten la compilación, procesamiento, almacenamiento, transmisión de información como: voz, datos, texto, video e imágenes (Art. 6 Ley 1341 de 2009).

Las Tecnologías de la Información y las Comunicaciones (TIC), son el conjunto de recursos, herramientas, equipos, programas informáticos, aplicaciones, redes y medios; que permiten la compilación, procesamiento, almacenamiento, transmisión de información como: voz, datos, texto, video e imágenes (Art. 6 Ley 1341 de 2009).”[12]

Ilustración 8 Controles
Tomada https://www.saludcastillayleon.es/ventanafamilias/es/infancia/videojuegos-tecnologia-informacion-comunicacion-tic

12.1 Beneficios

- Automatización de tareas repetitivas: en los motores podemos usar IA y otros beneficios de la tecnología para búsqueda de errores, tanto en el código como en el motor gráfico.
- Comunicación: Usando nuevas plataformas podemos tener talento internacional con conocimientos especiales y la comunicación se pueda dar por medio de plataformas como Slack o Teams.

[12] https://mintic.gov.co/portal/inicio/Glosario/T/5755:Tecnologias-de-la-Informacion-y-las-Comunicaciones-TIC

- Mejora de Marketing y soporte al usuario: Podemos usar Chatbots en nuestra página para poder hacer rutas básicas de resolución de problemas en nuestro videojuego.
- Aumento de la seguridad de los datos: Para la gestión de pagos a través de nuestra página web, tales como Certificados de seguridad y otros

"La tecnología en desarrollo de videojuegos crece exponencialmente y está generando un mayor impacto. Cada vez más, los desarrolladores pueden contar con nuevas herramientas para sorprender al público, ya sea a través de periféricos y otros dispositivos, o aplicaciones de inteligencia artificial.

La industria de los videojuegos supone un 0.11 % del PIB de España y, a nivel global, es un sector que genera 3.577 millones de euros, según informa la Asociación Española de Videojuegos, AEVI. Se trata de un sector que representa, tal y como subraya un informe económico elaborado por esta asociación, "el 14,3% del sector de la edición, el 9,6% del sector de la producción audiovisual (cine, video, televisión y música), el 3,8% del sector de la programación y tratamiento de datos y el 3,2% del sector de las telecomunicaciones". [13]

Ilustración 9 Jugador
Tomado de https://www.mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/238562:Accesibilidad-en-videojuegos-todos-podemos-jugar

12.2 Procesos TIC en los Videojuegos

"El desarrollo tecnológico ha supuesto un importante avance en la sociedad actual, influyendo en el ámbito educativo. Mismamente, las Tecnologías de la Información y la Comunicación (TIC) han dado lugar a procesos de enseñanza y aprendizaje más flexibles y significativos. En este sentido, los Exergames constituyen una herramienta con la que realizar ejercicio en edad escolar, por lo que resulta de interés su empleo en sesiones de Educación Física. Este estudio pretende determinar parámetros de ocio-digital, actividad física y experiencias previas con Exergames en una muestra de alumnado de Educación Primaria, analizando en qué medida se relacionan con la posibilidad de utilizar esta tecnología en Educación Física. El estudio contó con la participación de 520 escolares, utilizando para la recogida de datos un cuestionario de elaboración propia. Los resultaros mostraron

[13] https://www.telefonica.com/es/sala-comunicacion/blog/tecnologia-en-desarrollo-de-videojuegos/

que los alumnos de esta edad tienen una actitud positiva a la realización de Educación Física a través de Exergames; sin encontrar diferencias estadísticamente significativas en la relación con las demás variables. Como conclusión, señalamos la necesidad de innovar en los procesos educativos para propiciar un aprendizaje más significativo, siendo los Exergames un instrumento a tener en cuenta en el área de Educación Física."[14]

Proceso	Mejora TIC Implementada	Resultado
Desarrollo y Diseño	Durante el desarrollo se pueden usar plataformas de comunicación entre los desarrolladores para el uso el motor grafico	Producción Mas Eficiente
QA y Testing	Se pueden realizar Testing básico de las funciones	Mejora en la calidad y las pruebas
Venta	Podemos mejorar la venta de nuestros videojuegos	Acceso a los usuarios con más facilidad
Atención al Cliente	Análisis más rápidos de los datos, con foros o automatizaciones de Chatbots	Soporte Personalizado
Actualización / Parches	Con las actualizaciones podemos usar automatizaciones para descubrir los errores más rápido que reporten los usuarios	Juegos con mayor Re-jugabilidad y duración

12.3 Estrategias

- **Herramientas de grupo Colaborativa:** Aquí podremos usar cosas como Google Drive o Trello para crear un entorno más colaborativo.
- Analítica de Datos: Para revisar de nuevo la experiencia de nuestros jugadores a medida que pasen uno u otro nivel.
- Capacitación Continua: De nuevos lenguajes o motores gráficos.
- Procesos de revisión y Validación: Por medio de Stakeholders o los usuarios quienes pueden ayudarnos con feedback o prototipos.

12.4 Caso Practico

Accesibilidad a los Videojuegos, que según el ministerio de las TIC cada día hay mayor accesibilidad a los mismos tal como lo muestra este reportaje

"El Internet y las TIC han permeado todos los aspectos de la vida humana. Además de los grandes avances y desarrollos tecnológicos en ámbitos laborales o

[14] https://dialnet.unirioja.es/servlet/articulo?codigo=5580046

educativos, también han representado una revolución en materia de entretenimiento y de las industrias creativas, como en el caso de los videojuegos.

No solo es cuestión de entretenimiento, los videojuegos traen muchos beneficios para las personas que los practican, entre ellos, mejorar la capacidad de respuesta, la estrategia y el liderazgo estimular la creatividad, favorecer el pensamiento crítico, fomentar el trabajo en equipo, aumentar la capacidad para resolver conflictos, mejorar la concentración y la atención visual, fortalecer las relaciones sociales o adquirir nuevos conocimientos."[15]

[15] https://www.mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/238562:Accesibilidad-en-videojuegos-todos-podemos-jugar

13. Diagramas para la especificación y análisis de requisitos

La especificación y el análisis de requisitos para nuestro videojuego es una parte importante por ello utilizar diagramas nos ayudaría a una identificación clara de funcionalidades, actores y como interactúan entre si dentro del sistema del videojuego sirviendo de base para el diseño, la estimación y la validación del producto a desarrollar.

13.1 Tipos de diagramas

- **“Diagrama de clases**

Un diagrama de clases UML representa un sistema estático orientado a objetos. Define proyectos por clases, atributos y funciones; como tal, es un bloque de construcción fundamental de cualquier solución orientada a objetos. Muestra las clases de un sistema y las operaciones de cada una de ellas.

- **Diagrama de paquetes**

Los diagramas de paquetes agrupan las clases en paquetes. Muestran las distintas dependencias y relaciones entre los paquetes de un sistema.

- **Diagrama de objetos**

Los diagramas de objetos son similares a los diagramas de clases en el sentido de que muestran las relaciones entre los objetos de un software. La diferencia es que los diagramas de objetos utilizan ejemplos del mundo real. Los diagramas de objetos también se denominan diagramas de instancia porque muestran el aspecto del sistema en un momento determinado.

- **Diagrama de componentes**

Un diagrama de componentes UML ofrece a los desarrolladores una comprensión global de los objetos físicos de un sistema. Este tipo de diagrama UML muestra la relación estructural entre cada componente físico y subcomponente de un sistema de software complejo. Esto ayuda a las partes interesadas a comprender cómo se organizan y conectan los componentes.

- **Diagrama de estructura compuesta**

Los diagramas de estructura compuesta visualizan la estructura interna de una clase. Estos diagramas desglosan la red de clases, interfaces y componentes. Este tipo de diagrama UML también muestra cómo interactúan estos elementos entre sí y por qué son esenciales para la estructura general del software.

- **Diagrama de despliegue**

Un diagrama de despliegue muestra la relación entre los componentes de software y hardware de un sistema. Representa la disposición física de los nodos en un sistema distribuido. Estos diagramas son especialmente útiles cuando el software que se está desarrollando funcionará en varios sistemas de hardware diferentes."[16]

13.2 Diagrama

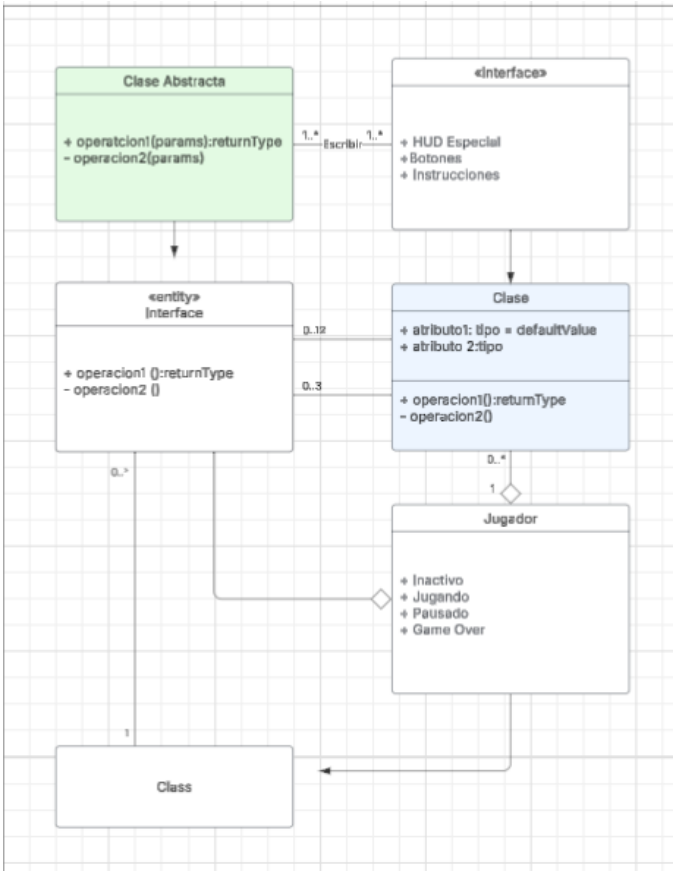


Ilustración 10 Diagrama UML

[16] https://miro.com/es/diagrama/que-es-diagrama-uml/#diagramas-estructurales

14. Diagramas y plantillas para casos de uso del proyecto

“¿Qué es un diagrama de casos de uso?

Un diagrama de casos de uso UML es el formato principal para los requisitos del sistema/software de un nuevo programa de software en desarrollo. Los casos de uso especifican el comportamiento esperado (qué), no el método exacto para lograrlo (cómo). Una vez especificados, los casos de uso pueden representarse tanto textualmente como visualmente (es decir, un diagrama de casos de uso). Un concepto clave del modelado de casos de uso es que nos ayuda a diseñar un sistema desde la perspectiva del usuario final. Es una técnica eficaz para comunicar el comportamiento del sistema en términos del usuario, especificando todo el comportamiento visible externamente del sistema.”[17]

Se revisan los casos de uso:

14.1 Estructurales

- Clase
- Componentes
- Estructuras compuestas
- Despliegue
- Objetos
- Paquetes
- Perfiles

14.2 De comportamiento

- Actividades
- Comunicación
- Interacciones
- Máquinas de Estado
- Secuencias
- Tiempos
- Casos de Uso[18]
-

[17] https://www-visual--paradigm-com.translate.goog/guide/uml-unified-modeling-language/what-is-use-case-diagram/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

[18] https://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso

14.3 Tipos de diagramas

"¿Cuáles son los diferentes tipos de diagramas UML? Hay dos categorías principales: diagramas de estructura y diagramas de comportamiento. Haga clic en los enlaces para obtener más información sobre cada tipo de diagrama.

- Diagramas de estructura
  - Diagrama de clases
  - Diagrama de componentes
  - Diagrama de implementación
  - Diagrama de objetos
  - Diagrama de paquete
  - Diagrama de perfil
  - Diagrama de estructura compuesta

- Diagramas de comportamiento
  - Diagrama de casos de uso
  - Diagrama de actividades
  - Diagrama de máquina de estados
  - Diagrama de secuencia
  - Diagrama de comunicación
  - Diagrama de descripción general de la interacción
  - Diagrama de tiempos"[19]

14.4 Plantillas

Campo	Descripción
Identificador	CU001
Nombre	Guardar Partida
Descripción	Propósito del caso de uso (qué objetivo cumple el usuario)
Actores	Usuario Final
Precondiciones	Qué debe cumplirse antes
Flujo Principal	Secuencia regular de pasos
Flujo Alternó	Secuencias distintas por excepción o variante

[19] https://creately-com.translate.goog/blog/diagrams/uml-diagram-types-examples/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

```
+ Identificador: CU001

  Nombre: Guardar Progreso
  Descripción: El jugador almacena su avance actual en el servidor.
  Actores: Jugador, Servidor
  Precondiciones: El jugador debe haber iniciado sesión.
  Flujo principal:
      1. El jugador selecciona "guardar progreso".
      2. El sistema almacena los datos.
      3. Se confirma la operación.
  Flujos alternos:
  A1. Error de red: Se muestra mensaje de error.
  Poscondiciones: Progreso actualizado en la base de datos.
  Reglas de negocio: Se puede guardar hasta 3 veces por hora.
+ Observaciones: Ver diagrama de secuencia adjunto.
```

Ilustración 11 Plantilla

15. Historias de usuario del proyecto

Durante esta sección revisaremos las historias de nuestros clientes con nuestro producto, con descripciones pequeñas para poder ver el punto de vista del usuario final, experiencia y valor añadido adicional.

Para esta historia de usuario revisaremos una estructura básica de un cliente tal y como:

Como _____ quiero _____ para _____.

Como "Jugador" Quiero "poder guardar mi proceso" para "que en caso de que cierre mi juego de manera sorpresiva".

Para estas historias de usuario debemos tener un lenguaje claro, breves, negociables, los cuales servirán para que en el futuro post venta nos volvamos a encontrar con estos usuarios finales y se revisen las proyecciones.

En estas historias queremos que se priorice la experiencia del usuario y los objetivos del estudio.

Estas historias se pueden adaptar según el perfil del equipo de desarrollo las cuales se usan para desarrollar y planificar las funciones de mayor importancia.

Historia de Usuario	Criterios de Aceptación
Como jugador, quiero un mapa interactivo, para no perderme dentro de los niveles.	El jugador puede abrir/cerrar el mapa; el mapa muestra su ubicación.
Como desarrollador, quiero registrar cada error durante la partida, para facilitar su corrección.	Se crea un log interno accesible desde el panel de administración.
Como jugador competitivo, quiero ver una tabla de líderes online, para comparar mis resultados.	Visualización de ranking y puntuaciones actualizadas en tiempo real.
Como jugador nuevo, quiero un tutorial interactivo, para aprender las mecánicas básicas sin leer manuales.	Creación de un tutorial con las dinámicas básicas
Como jugador frecuente, quiero poder guardar y cargar mi partida, para no perder mi avance si debo cerrar el juego.	Crear nuevos métodos de guardar y cargar partida por medio de un menú interactivo
Como programador, quiero contar con una pantalla de estadísticas de uso, para analizar el comportamiento de los usuarios y mejorar la experiencia.	Se crea un programa de estadísticas en Excel donde se ven la cantidad de ventas y demás
Como artista, quiero una galería donde visualizar los assets visuales en distintos fondos, para asegurar la calidad en todos los escenarios.	Se generará una biblioteca donde se encuentren los assets tantos internos como de bibliotecas gratuitas que podamos usar para poder ahorrar tiempos y costos
Como usuario con discapacidad visual, quiero un modo de alto contraste, para poder jugar con mayor comodidad.	Revisar junto con las plataformas métodos de accesibilidad y con ello poder desarrollarlo e incluirlo en nuestro videojuego

Una historia de usuario es la unidad de trabajo más pequeña en un marco ágil. Es un objetivo final, no una función, expresado desde la perspectiva del usuario del software.
Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final o cliente.
El propósito de una historia de usuario es articular cómo un elemento de trabajo entregará un valor particular al cliente. Ten en cuenta que los "clientes" no tienen por qué ser usuarios finales externos en el sentido tradicional, también pueden ser clientes internos o colegas dentro de tu organización que dependen de tu equipo.[20]

[20] https://www.atlassian.com/es/agile/project-management/user-stories

16. Modelo del dominio del Proyecto

Para el modelo de dominio del proyecto por medio de diagramas de clases para el software de nuestro videojuego con ello dar una adecuada gestión de software y con ello poder manejar el correcto desarrollo de este.

“Un modelo de dominio describe los tipos de dominio que admite una organización y sus restricciones.

Un modelo de dominio está formado por un grupo de dominios atómicos. Un dominio atómico representa un tipo de datos abstracto que se puede restringir mediante la adicción de restricciones. Los tipos de datos de dominio se basan en tipos de datos de base. Por ejemplo, puede definir tipos de datos de dominio para definiciones utilizadas con frecuencia, como el número de la seguridad social, el sexo, la altura o el estado civil.

Las definiciones de restricción siguen las definiciones de restricción de esquemas XML. Por ejemplo, puede utilizar una restricción de enumeración para limitar el número posible de valores; por ejemplo, un dominio denominado "prioridad" podría tener tres valores posibles: "alta", "media" o "baja". Puede utilizar una restricción de patrón para especificar una restricción en una expresión regular, por ejemplo "ABC". En este producto, las restricciones sólo se utilizan para la documentación. No hay construcciones adicionales generadas para restricciones durante la transformación del modelo lógico de datos en modelo físico.*

Los objetos de modelo de dominio se pueden almacenar en un archivo de modelo de dominio (.ddm) o en un archivo de modelo lógico de datos (*.ldm).*

Con el área de trabajo, puede crear un modelo de dominio a partir de una plantilla o importar tipos simples a partir de un archivo de definición de esquemas XML (.xsd) en un modelo de dominio como tipos de dominio. También puede exportar un modelo de dominio a un archivo .xsd. Un modelo de dominio se puede asociar con un modelo lógico para que cada dominio del modelo se pueda utilizar como un tipo de datos de atributo.”[21]

[21] https://www.ibm.com/docs/es/ida/9.1.2?topic=types-domain-models

16.1 Objetivo

- Elaboración de un diagrama de clases del proyecto
- Relación entre clases
- Relación entre categorías
- Identificar las relaciones del proyecto de categoría y clase
- Con un diagrama de clases crear un mapa de ruta
- Revisar el modelo del juego

16.2 Pasos

- **Identificación de entidades principales**

Analiza la temática y objetivos del videojuego para definir las entidades fundamentales: Jugador, Escenario, Personajes, Enemigos, Recompensas, Niveles, Etc.

- **Definición de Atributos y Relaciones**

Especifica los atributos más relevantes para cada entidad

- **Elaboración del diagrama UML**

Representa cada entidad como una clase UML con sus atributos principales

- **Refinamiento y comunicación**

Agrega detalles solo si son necesarias para comprender el contexto del juego (No incluimos lógica ni métodos de software

16.3 Modelo de Clase

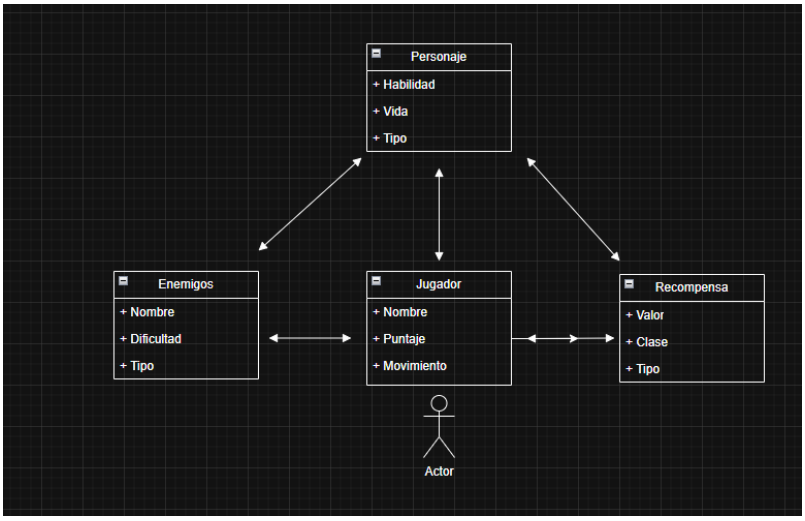


Ilustración 12 Diagrama UML

17. Validación de Documentos

A continuación, revisaremos un mapa conceptual sobre la validación de documentos de nuestro videojuego:

Ilustración 13 Validación de Documentos

18. Listas de chequeo para la validación de artefactos

En esta sección revisaremos la documentación necesaria por medio de informes y listas de chequeo básicas para la actividad de desarrollo de nuestro videojuego.

Debido a la complejidad del desarrollo de nuestro software debemos tener una información básica de cada paso del desarrollo para así evitar reprocesos durante el camino por vemos importantes listas de chequeo en cada caso.

18.1 Paso a paso de Cumplimiento

Este videojuego entra a la industria con el propósito de competir con una industria saturada de juegos los cuales no satisfacen a los jugadores actuales

Pasos:

- **Recolección de Requerimientos del videojuego**

Historias, mecánicas, personajes, interfaces, gráficos y sonidos

- **Identificación y desglose de Artefactos**

Documento de diseños de juegos, prototipos, diagramas de flujo, sprites, audio, código fuente, documentos de prueba

- **Asignación de Responsabilidades**

Artistas, programadores, diseñador de niveles, QA

- **Desarrollo de artefactos**

Creación de cada elemento según los requerimientos

- **Preparación para la validación**

Compilación y organización de todos los artefactos listos para revisión

18.2 Estrategias de Cumplimiento

- **Revisión de pares**

Se asignan miembros del equipo para revisar artefactos de otros roles (Diseñadores revisan folletos, programadores revisan el código, entro otros)

- **Validación iterativa**

Revisiones frecuentes de artefactos en cada Sprint de Desarrollo

- **Cheklist de cumplimiento**

Uso de lista de chequeo para cada tipo de artefactos

- **Participación del cliente y usuarios finales**

Validación temprana de prototipos mediante pruebas de jugabilidad y retroalimentación

- **Herramientas Digitales**

Uso de sistemas de control de versiones (Git) o tableros como Trello para una trazabilidad del proyecto

18.3 Informe de Artefactos Entregados

Artefacto	Descripción	Responsable
Documento de Guion	Narrativa	Persona 1
Bocetos	Arte conceptual	Persona 2
Prototipo	Versión jugable	Persona 3
Música y sonidos	Música	Persona 4
Código fuente	Movimientos del Jugador	Persona 5

18.4 Procesos y usuarios identificados

Procesos involucrados

- Recolección y especificación de Requisitos (Game Desing)
- Desarrollo de Artefactos (Arte, Programación, Sonidos)
- Pruebas de jugabilidad y usabilidad
- Validación final y entrega

Usuarios interesados

- Desarrollador
- Diseñador del juego y narrativa
- Productor
- Manager de Proyecto
- Clientes (Si aplica)
- Jugadores (Beta y finales)

18.5 Validación de usuarios

- Revisión guiada de prototipos y demos junto con el usuario final o QA
- Encuestas y entrevistas a usuarios que prueben versiones jugables
- Recopilación de Feedback de los usuarios y asignación de tareas para posibles ajustes
- Registro de validaciones con cheklist firmadas o bitácoras

18.6 Lista de chequeo Paso a Paso

Esta lista de cheque nos permite asegurar que se estén cumpliendo las actividades que se esperan de este proyecto, para tener un correcto lanzamiento del juego sin casi errores en el producto u otros

- Inventario de artefactos a Validar
- Definir criterios de aceptación para cada artefacto
- Crear preguntas o ítems de verificación
- Preparar el formato de registro
- Ejecutar la verificación y anotar resultados
- Registro de observaciones y Responsables
- Almacenamiento de evidencias (Videos, Archivos y más)

Ítems	Si	No	Observaciones
el personaje principal se mueve con los controles seleccionados			
El escenario esta correcto en cada nivel y no se camufla con el personaje			
Las mecánicas funcionan correctamente			
Existe música de ambiente funciona			
Los gráficos están alineados con el ambiente del juego			
No hay Bugs críticos que impidan a los usuarios continuar entre niveles			
Los menús y HUDs funcionan y son entendibles			
Los prototipos funcionan perfectamente			

19. Especificación de los referentes técnicos del hardware - software y estimación de las condiciones económicas

A continuación, se presenta la especificación de los referentes técnicos y la estimación de las condiciones económicas y para el desarrollo del videojuego, siguiendo las normas y estructura recomendada en los componentes formativos de especificación del software.

19.1 Especificaciones Técnicas

Para el desarrollo de nuestro videojuego se requieren las siguientes condiciones mínimas para el hardware de desarrollo

Componentes	Especificaciones
Procesador	Cuatro núcleos físicos, frecuencia ≥3.0GHz
RAM	16 GB 4RRD
Almacenamiento	1 Tera de almacenamiento SSD
Tarjeta Grafica	Tarjeta dedicada de 4GB, Compatible con Direct x12
Monitor	Full HD (1920x1080), 24 pulgadas o más
Accesorios	Teclado y ratón óptico
Internet	Cable Ethernet o servicio de internet Wi-fi Propio
Sistema Operativo	Windows 11 de 64 bits

Esto aplica para equipos de desarrollo como para estaciones de prueba y usuarios finales si el videojuego es para PC, para Consola o teléfonos, los requerimientos se ajustan al estándar especifico de la plataforma de publicación.

19.2 Especificaciones del Software

Para el desarrollo de nuestro videojuego se requieren las siguientes condiciones mínimas para el software, así podremos crear el código fuente y el arte de este.

Software	Especificación mínima
Motor Grafico	Unity, Unreal Engine y Godot, todos en la versión actual
Plataforma Código	Visual Studio code Actual
Arte	Blender, Photoshop/GIMP, Krita
Versiones	Git
Sistema Operativo	Windows 11 de 64 bits
Librerías y Módulos	Adecuada al motor grafico elegido

19.3 Requerimientos funcionales o no Funcionales

- Multiplataforma para PC, consola y teléfonos
- Rendimiento fluido a 60fps en hardware de mínima referencia
- Compatibilidad con controles estándar
- Ciberseguridad de los datos de compra si es en nuestra página web dado que plataformas como Steam o PlayStation Store ya las tienen
- Accesibilidad básica Como subtítulos o cambios de colores en caso de personas daltónicas.

19.4 Estimación de condiciones Económicas

La estimación económica debe considerar tanto los gastos directos como indirectos asociados al desarrollo de nuestro videojuego, para nosotros como indie lo que necesitaríamos en dinero es:

19.4.1 Costo de Hardware

Los costos de Hardware serian:

Producto	Proveedor / Modelo	Costo Aproximado (COP)
CPU	Ryzen 5 5600X	$1,229,034
GPU	RTX 3060	$1,644,192
Memoria RAM	Corsair Vengeance	$349,391
SSD 1TB	Samsung 980 PRO	$575,467
Monitor	LG 24MK430H	$698,782
Total		$4,496,866

19.4.2 Costo de Software

Los costos del Software seria:

Producto	Proveedor / Modelo	Modalidad	Costo Aproximado
Motor de Juego	Unity	Anual	$7,398,864
IDE Programación	Microsoft	Gratis*	0
Suite de Arte	Adobe CC	Mensual	$205,524
Control de Versiones	GitHub	Anual	$205,524
Servicios en la nube	AWS	Mensual	$328,838
Total			$8,138,750

19.4.3 Mano de obra

La mano de obra tiene es estimado de:

- Desarrollador: $2.000.000
- Artista: $1.423.000
- QA: $1.900.000
- Sonido: $1.423.000
- Productor: $2.000.000
- Servicios tercerizados: $1.423.000 cada empleado tercerizado

19.4.4 Otros

- Costos Legales: y de publicación: de $200.000 a $1.530.000
- Costos de Marketing y distribución: 30% del total, especialmente en plataformas de distribución como Steam o PlayStation Store

19.5 Conclusiones

Según los componentes formativos y estándares de especificación la estructura de nuestro juego es:

- Propósito
- Alcance
- Descripción general (Tipo de videojuego y público objetivo)
- Requerimientos del hardware y software
- Requerimientos funcionales y no funcionales
- Estimación de recursos y costos
- Referencias y normas, estándares y anexos técnicos

Unas recomendaciones adicionales para evitar el aumento de costos o generales son:

- Evitar el uso de marcas o Sprites registrados a menos que se tenga un convenio previo
- Incluir siempre los anexos y la información detallada del documento de desarrollo
- Revisar en todo momento la evolución tecnológica de nuestro videojuego
- Revisiones en todo momento y obtener feedback de nuestros usuarios finales y del equipo de desarrollo antes de la adquisición de nuevo hardware

20. Ficha técnica de los productos requeridos

La ficha técnica nos proporcionara la información esencial sobre los productos necesarios para el desarrollo de nuestro videojuego, incluyendo descripciones, características, funciones y una estimación de costos, lo cual nos facilita un presupuesto estimado y con ello poder manejar mejor los recursos

20.1 Tabla

Producto	Descripción	Características Principales	Función en el Proyecto	Proveedor / Modelo	Costo
CPU	Procesador de alto rendimiento para diseño y compilación	6 núcleos, ≥3.5GHz, tecnología multihilo	Procesos de desarrollo y pruebas	Ryzen 5 5600X	$1,229,034
GPU	Tarjeta gráfica para gráficos y simulación	12GB VRAM, arquitectura moderna, soporte DirectX 12/OpenGL 4.5	Renderizado de gráficos, pruebas visuales	RTX 3060	$1,644,192
Memoria RAM	Memoria para multitarea y rendimiento	16GB DDR4, 3200MHz, dual channel	Ejecución fluida de software de desarrollo	Corsair Vengeance	$349,391
SSD 1TB	Almacenamiento rápido y confiable	NVMe, velocidad ≥2,000MB/s lectura/escritura	Instalación de OS, motores y repositorios	Samsung 980 PRO	$575,467
Monitor	Pantalla para diseño y pruebas	24”, resolución Full HD, panel IPS	Visualización de arte y código	LG 24MK430H	$698,782
Motor de Juego (anual)	Entorno para la creación de videojuegos	Unity Pro, soporte multiplataforma, editor visual	Desarrollo del videojuego	Unity	$7,398,864
IDE Programación	Entorno integrado de desarrollo	Visual Studio 2022, depurador avanzado, soporte C#	Codificación y pruebas	Microsoft	Gratis*
Suite de Arte (mensual)	Herramientas para gráficos y animación	Adobe CC, soporte PSD/AI, integración cloud	Diseño 2D/3D y edición multimedia	Adobe	$205,524
Control de Versiones (anual)	Plataforma de colaboración y versionado	GitHub Teams, repositorio privado, manejo ramas	Control y seguimiento de cambios	GitHub	$205,524
Servicios en la nube (mensual)	Infraestructura para pruebas o backend	AWS, 200GB almacenamiento, 2TB transferencia/mensual	Hosting, almacenamiento, pruebas online	Amazon Web Services	$328,838

20.2 Objetivo

- Analizar funcionalidades y capacidad de cada producto
- Comparar proveedores y costos para optimizar el presupuesto
- Facilitar la gestión de compra y el control del inventario técnico del proyecto

21. Diseño de tablas comparativas sobre presupuestos de hardware y software

El presente documento tiene como propósito exponer el análisis comparativo del presupuesto de hardware y software ofertados por diversos proveedores, en el contexto del desarrollo de un videojuego, siguiendo las condiciones establecidas en los lineamientos de la propuesta técnica, se utilizará una ficha de comparación, que nos ayudará a tomar la mejor decisión técnica y económica en relación Calidad-Precio

21.1 Metodología

- Se han identificado los componentes esenciales del Hardware y el software requerido para el desarrollo eficiente de nuestro videojuego, siguiendo los estándares profesionales de la industria
- Se contactaron 3 proveedores nacionales donde se cotizaron y se hicieron simulación en precios de 2025
- Este análisis contempla precios unitarios, garantías y demás

21.2 Tabla comparativa Hardware

Componente	Proveedor A	Proveedor B	Proveedor C
CPU	Ryzen 5 5600X $2.000.000	Intel i5-12600K $2.900.000	Intel i5-11600K $2.500.000
GPU	RTX 3060 $400.000	RX 6600 XT $365.000	RTX 3060 $410.000
RAM	16GB DDR4 $85.000	16GB DDR4 $95.000	16GB DDR4 $80.000
SSD 1TB	NVMe $140.000	NVMe $130.000	SSD SATA $120.000
Motherboard	B550 $150.000	Z590 $180.000	B560 $140.000
Fuente poder	650W Gold $100.000	650W Bronze $90.000	600W Gold $110.000
Torre/Case	ATX Airflow $900.000	ATX Básica $700.000	Micro-ATX $850.000

21.3 Tabla comparativa Software

Software	Proveedor A	Proveedor B	Proveedor C
Motor de juego	Unity Pro (licencia anual) $3.500.000	Unreal (regalías)	Godot (libre)
IDE programación	Visual Studio Gratis	JetBrains Rider $250.0000/año	Visual Studio Code Gratis
Suite de arte digital	Adobe CC $250.000/mes	Affinity Suite $150.000 (pago único)	GIMP/Krita Gratis
Gestión de versiones	GitHub Teams $100.000/año	GitLab Gratis/Plan Pro $100.0000/año	Bitbucket Gratis
Servicios en la nube	AWS $100.000/mes	Azure $90.000/mes	Google Cloud $85.000/mes

21.4 Análisis

- **Hardware:** La diferencia entre precios de los proveedores para el kit completo no supera el 3% de la decisión y lo que se busca es la mayor calidad posible en el software
- **Software:** Seleccionar
- **Proveedores:** Las propuestas de Hardware de todos los proveedores son técnicamente equivalentes y cumplen estándares recomendados para el desarrollo de videojuegos actuales
- **Condiciones Comerciales:** Considerar en todo momento el tiempo de entrega, facilidades de pago, acceso a soporte tenido postventa y posibles actualizaciones en el futuro

22. Propuesta técnica y económica para la implementación del proyecto

Nuestro videojuego que presenta una solución tecnológica de entretenimiento, presentaremos una propuesta técnica y económica para la creación de este orientado a cubrir las necesidades y expectativas de la industria

22.1 Alcance

La presente propuesta cubre el desarrollo integral de un videojuego:

- Diseño conceptual y narrativo
- Programación y desarrollo artístico
- Implementación en plataformas tales como Pc, consola o teléfono
- Prueba de calidad y ajuste de rendimiento
- Entrega final e implementación
- Capacitación básica y soporte post-lanzamiento

22.2 Valoración

En la situación actual y las necesidades de la empresa necesitamos:

- Nicho del mercado de videojuegos
- Plataformas requeridas y alcance geográfico
- Requerimientos funcionales y técnicos principales de jugabilidad, diseño y conectividad
- Recursos disponibles y limitaciones presupuestales

22.3 Opciones

Se presentan dos opciones principales

Opción 1: Videojuego Estándar Multiplataforma

- Gráficos 2D de alta calidad
- Mecánicas clásicas ya sea Puzzle o runners
- Soporte básico para actualizaciones
- Integración básica de estadísticas y logros

Opción 2: Videojuego Avanzado

- Gráficos 2.5D a 3D
- Multijugador Local
- Funcionalidades Sociales
- Motor de físicas avanzados
- Sistema de micro transacciones

22.4 Estimación de costos

Una estimación de costos del desarrollo seria:

Gasto	Opción 1	Opción 2
Hardware y Periféricos	$ 5,332,000	$ 6,589,200
Licencia de software y motores	$ 2,000,000	$ 3,000,000
Mono de obra durante el desarrollo	$10,500,000	$10,500,000
QA y pruebas	$ 2,000,000	$ 5,000,000
Marketing y publicación	$ 5,600,000	$ 9,800,000
Costos legales y demás	$10,000,000	$15,000,000
Totales	$35,432,000	$49,889,200

Esta cotización formal el valor final podría ajustarse según requerimientos específicos, tiempo adicional de desarrollo o funcionalidad extra solicitadas durante el desarrollo.

22.5 Términos y condiciones

- Formas de pago: 30% del anticipo si es posible, 40% en la versión beta y 30% en la entrega final en caso de que sean tercerizados
- Proyecto estimado a 2 años, sujeto a alcance pactado
- Propiedad intelectual dado que según el licenciamiento el usuario podrá disfrutar la licencia de uso del juego excepto por tecnologías de terceros
- Soporte por hasta 60 días después de la compra
- Revisión de precios, la cotización será válida por 30 días
- Confidencialidad: Se asegura la protección de la información en caso de que se compre por nuestra página web.

23. Modelos conceptual y lógico para el proyecto

En esta sección revisaremos diferentes modelos lógicos y conceptuales para nuestro videojuego, basados en modelos entidad-relación para una base de datos aquí consideraremos varias cosas como el modelo conceptual, modelos lógicos, bases de datos y la especificación del análisis junto con un diagrama visual.

23.1 Modelo Conceptual

El modelo conceptual representa entidades principales del sistema y sus relaciones de alto nivel, sin considerar aspectos técnicos de implementación.

1. **Usuario/Jugador:** Entidad Central que representa a los usuarios del sistema.
2. **Partida:** Sesión de juego individual de un usuario.
3. **Nivel:** Etapas o fases del juego.
4. **Puntaje:** Sistema de puntuación o logros.
5. **Configuración:** Preferencias y ajustes del usuario.
6. **Estadística:** Datos de rendimiento o progreso.

23.2 Modelo Lógico

El modelo lógico describe de forma detallada la estructura de las tablas, los campos, los tipos de datos y las relaciones necesarias para la implementación en una base de datos.

La primera tabla, **USUARIOS**, contiene la información de los jugadores o participantes. Sus campos son:

- **id_usuario (PK)**: un número entero que se incrementa automáticamente y actúa como identificador único.
- **nombre_usuario**: una cadena de texto de hasta 50 caracteres, con restricción de unicidad.
- **email**: dirección de correo electrónico del usuario, con un máximo de 100 caracteres y también única.
- **fecha_registro**: almacena la fecha y hora en que se registró el usuario.

- **estado_activo**: un valor booleano que indica si el usuario se encuentra activo o no.

La segunda tabla, **PARTIDAS**, almacena la información referente a las sesiones de juego. Incluye los siguientes campos:

- **id_partida (PK)**: identificador único de cada partida, generado automáticamente.
- **id_usuario (FK)**: clave foránea que hace referencia al identificador del usuario que jugó la partida.
- **fecha_inicio** y **fecha_fin**: registran el momento de inicio y finalización de cada partida.
- **puntaje_total**: almacena la puntuación obtenida en la sesión.
- **nivel_alcanzado**: indica el nivel más alto alcanzado por el jugador en esa partida.

Finalmente, la tabla **NIVELES** define las características de cada nivel disponible en el juego. Está compuesta por los siguientes campos:

- **id_nivel (PK)**: identificador único del nivel.
- **nombre_nivel**: nombre descriptivo del nivel, con un máximo de 100 caracteres.
- **dificultad**: establece el nivel de dificultad, que puede ser “Fácil”, “Medio” o “Difícil”.
- **tiempo_limite**: indica el tiempo máximo permitido para completar el nivel.
- **puntaje_objetivo**: define el puntaje necesario para superar el nivel.

23.3 Especificaciones del Análisis

Requisitos de datos

1. **Almacenamiento de usuarios:** Gestión de cuentas y perfiles.
2. **Registro de partidas:** Historial completo de sesión de juego.
3. **Sistema de Niveles:** Progresión estructurada del juego.
4. **Estadísticas en tiempo real:** Monitoreo de Rendimiento**.**
5. **Configuración personalizada:** Preferencias del usuario.

Integridad de datos

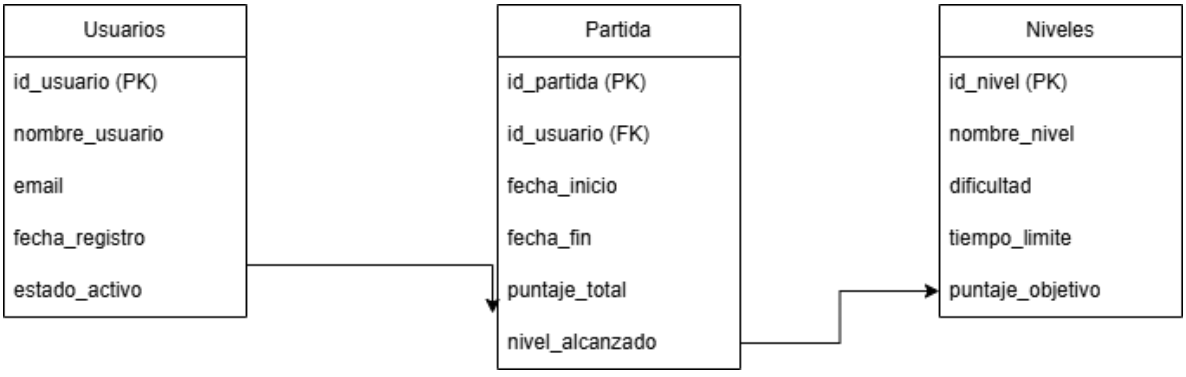
1. Claves primarias únicas para cada entidad.
2. Claves foráneas para mantener relación.
3. Restricción de integridad referencial.
4. Validación de datos de entrada.
5. Índice para optimización de consultas.

23.4 Tipos de Bases de Datos

1. **MySQL:** Base de datos relacional robusta, que es ideal para aplicaciones web con alta demanda.
2. **PostgreSQL:** Base de datos avanzadas con soporte para JSON y características modernas.
3. **MongoDB:** Base de datos NoSQL orientada a documentos, flexible para datos no estructurados.
4. **SQLite:** Base de datos ligera embebida, perfecta para aplicaciones Standalone.

Para nuestro videojuego la más recomendada es SQL debido a su amplia compatibilidad con Frameworks web, excelente rendimiento para aplicaciones de mediana escala, comunidad activa y documentación extensa, Herramientas de administración maduras y por último un costo/beneficio favorable

23.5 Diagrama



PK (Primary Key)

FK (Foreing Key)

Unique (Valor único de Tabla)

Podemos concluir que el modelo conceptual y lógico del proyecto propuesto es una base sólida para el desarrollo de nuestro videojuego y con eso aseguramos:

1. **Escalabilidad:** Estructura que puede crecer con el proyecto.
2. **Integridad:** Datos consistentes y relaciones bien definida.
3. **Rendimiento:** Optimización para consultas frecuentes.
4. **Mantenibilidad:** Estructura clara y documentada.
5. **Flexibilidad:** Capacidad de adaptación a nuevos requerimientos.

24. Informe de entregables para el proyecto de desarrollo de software

En el siguiente informe revisaremos nuestro proyecto de software donde revisaremos algunos conceptos básicos ya construidos en nuestro videojuego, donde revisaremos a detalle los usuarios, las historias de este o los casos de usos, entre muchos otras

Aquí revisaremos un resumen a detalle del proyecto mejorado.

24.1 Identificación de usuarios

Aquí revisaremos un Brief de marca de nuestro videojuego iniciando con los tipos de usuarios:

7. **Jugador Principal:** El usuario que experimenta el juego, controla los personajes y avanza los niveles
8. **Administrador:** Hace la gestión de usuarios, los niveles y estadísticas generales del juego.
9. **Desarrollador/Testers:** Se encarga del manteamiento, las pruebas y actualizaciones de contenido

- Cuál es el tipo de público o clientes a los que la empresa se dirige actualmente:

- **Edad:** Entre 13 a 40 años dado que algunos adolescentes y adultos gustan de juegos con la estética como la de nuestro videojuego
- **Género:** Predominantemente mixto, los juegos de plataformas atraen tanto a hombres como a mujeres gracias a su estilo visual.
- **Lugar de origen:** Bogotá y ciudades de Latinoamérica donde los juegos indie 2D tienen fuerte presencial digital y cultural
- **Gustos:** Fan de los juegos 2D estilo Píxel, art o “Retro” y mecánicas inspiradas en juegos de plataforma sin depender de “Gráficos hiperrealistas”
- **Estilo de vida**: Jugadores casuales o profesionales que consuman claramente videojuegos por medio de redes sociales o plataformas digitales y también algunos jugadores que solo busquen una partida de camino al trabajo o algún otro lugar.

- **Cuál es el tipo de negocio de la empresa**

Nuestra empresa pertenece al sector de desarrollo y publicación de videojuegos, los cuales nos hace fabricantes y digitales y distribuidores independiente (Indie) de títulos para PC, consolas y teléfonos gracias a que usamos varios motores de desarrollo.

- **¿Qué unidades de negocio y/o productos maneja la empresa?**

Desarrollo y publicación de videojuegos, producción de contenido multimedia, monetización y soporte técnico y por último tiendas propias digitales o de terceros.

- **¿Qué lo diferencia de la competencia?**

El arte visual distintivo como uso del píxel artístico y animaciones fluidas, con jugabilidad original que mezcla plataformas clásicas con mecánicas modernas como físicas mejoradas o narrativas interactivas.

También tenemos una jugabilidad original que mezclan plataformas clásicas con mecánicas modernas como físicas mejoradas o narrativa interactiva.

El modelo escalable que tiene integración con eventos de streaming y eventos de videojuegos.

La historia envolvente que se diseñó con un mundo con identidad propia y desafíos adaptativos.

- **¿Cuáles son las fortalezas y debilidades de su empresa?**

Las fortalezas son el uso de motores eficientes como Unity, que permiten rapidez y multiplataforma, la alta interacción de la comunidad gamers mediante las redes sociales y algunas betas abiertas, y por último la creatividad artística y narrativa.

Las debilidades es nuestro limitado presupuesto frente a estudios AAA como Sony o Xbox, también vemos una saturación en el mercado de juegos 2D que nos dificulta mantener la visibilidad constante sin campañas de marketing intensivas.

- **¿Cuál o cuáles son las redes sociales donde el cliente tiene mayor presencia digital?**

Las plataformas de mayor impacto y al alcance para la empresa son principalmente YouTube y TikTok, también debemos tener como prioridad Twitch donde varios gamers transmiten videojuegos donde podría hacerse viral

- **¿Cuánto tiempo lleva la empresa en el entorno digital?**

Se esperas que la empresa tenga presencial digital activa entre 4 a 5 años conociendo el auge del mercado independiente y el crecimiento de motores 2D accesibles entre 2020 a 2025.

- **¿Cómo está estructurado su plan de marketing digital?**

El plan de marketing se basa en nuestro enfoque multicanal y orientado al engagement.

Por medio de SEO y posicionamiento digital en tiendas de videojuegos con optimización de descripciones y palabras clave en plataformas de ventas de videojuegos digitales.

También queremos hacer una publicidad segmentada en anuncios pagados en redes sociales y Google Ads dirigidos a jugadores casuales, retros y profesionales.

También debemos tener en cuenta los influencers de marketing con colaboraciones con eventos e influencers de Twitch o YouTube

También queremos tener como opción el contenido orgánico tales como videos virales, blogs con los desarrolladores y teasers en redes sociales.

Queremos hacer una analítica retargeting con seguimientos de conversaciones (Con autorización) también haciendo una optimización de campañas y creación de comunidades mediante Discord o Patreon.

Y por último hacer eventos digitales como lanzamientos, betas abiertas y desafíos en redes sociales de forma semanal.

24.2 Historias de Usuarios

Aquí veremos 5 nuevas historias de usuarios

1. Historia 1: Como jugador, quiero guardar mi progreso para continuar la partida más tarde sin perder lo que ya jugué.
2. Historia 2: Como jugador, quiero poder personalizar mi personaje para sentir que tengo una identidad única dentro del juego.
3. Historia 3: Como jugador, quiero recibir una o muchas recompensas al completar las misiones para sentir que progreso y sentirme motivado.
4. Historia 4: Como administrador, quiero visualizar las estadísticas de los jugadores para analizar el rendimiento general del juego.
5. Historia 5: Como desarrollador, quiero probar las funciones del juego en modo debug para detectar errores y mejorarlos.

24.3 Prototipo de Aplicación

El prototipo de aplicación incluye las siguientes pantallas principales

1. **Pantalla de inicio:** Con el logo del juego y los botones de “Jugar”, “Opciones” y “Salir”.
2. **Pantalla de Selección:** Permite crear o cargar un perfil y configuración de personajes.
3. **Interfaz de juego:** Muestra el entorno 2D como barra de vida, el mapa y las instrucciones si se requieren además de objetivos activos.
4. **Menú de pausa:** Opciones de guardar, reiniciar, configuraciones y salir al menú principal.
5. **Pantallas de superación y logros:** Visualizamos las estadísticas, récords y misiones completadas.

24.4 Casos de Uso

Aquí revisaremos un cuadro con los casos de usos más importantes:

Caso de uso	Actor	Descripción	Flujo Principal
Iniciar Sesión	Jugador	Accede con su cuenta o crea un nuevo perfil	El jugador ingresa datos → el sistema valida → entra al juego
Guardar Progreso	Jugador	Almacena datos de misión y nivel	Jugador puede seleccionar guardar → el sistema lo guarda en bases de datos
Configuración de personajes	Jugador	Podemos cambiar apariencia y habilidades	El jugador elige diseños → el sistema aplica los cambios
Ver las estadísticas	Administrador	Podemos consultar datos globales	El administrador solicita información → el sistema muestra resultados
Actualizar las versiones	Desarrollador	Aquí subiremos parches o mejoras	El desarrollador sube archivos → el sistema actualiza contenidos

24.5 Diseño del modelo Relacional

A continuación, veremos un diseño de modelo relacional de nuestro videojuego

Tabla	Campos	Clave primaria o foránea
Jugadores	id_jugador,nombre,email,experiencia	PK: id_jugadores
Personajes	id_personaje, nombre,tipo,puntis_vida,habilidad	PK: id_personajes
Misiones	id_mision,descripcion,recompensa,estado	PK: id_mision
Progreso	id_progreso,id_jugador,id_mision,fecha,estado	PK: id_progreso FK: id_jugadores, id_mision
Administradores	id_admin,nombre,permiso	PK: id_admin

24.6 Modelo UML

En nuestro videojuego tenemos varias variables UML tales como:

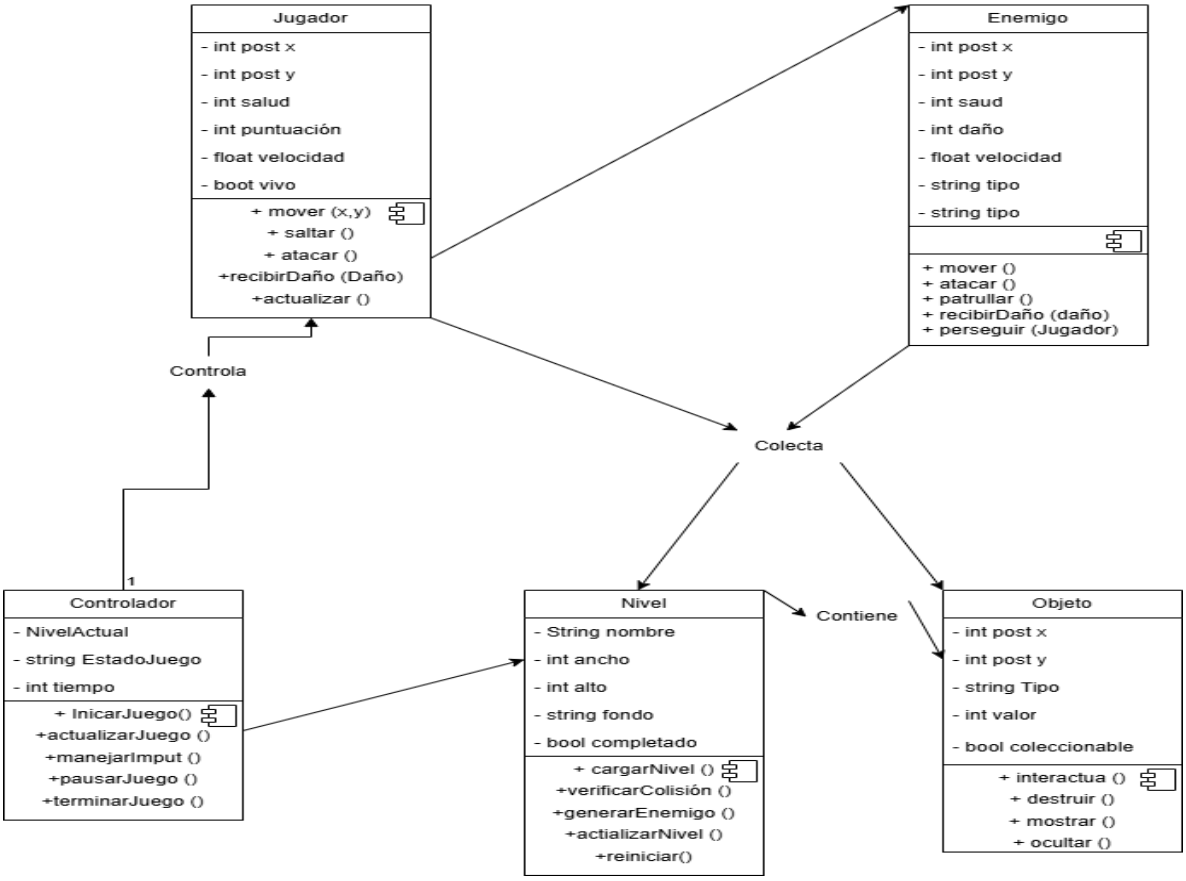


Ilustración 14 Diagrama UML Entregable

24.7 Conclusión

El diseño conceptual de nuestro videojuego establece la base sólida tanto la experiencia del jugador como para la administración técnica. A través de una arquitectura clara y un modelo relacional optimizado, facilitando la escalabilidad y el mantenimiento del proyecto.

Los próximos pasos son realizar la implantación del motor (Unity o Unreal) y desarrollar un prototipo funcional jugable que permita probar mecánicas básicas y recoger retroalimentación temprana.

25. Diagrama de Clases del proyecto de software

En nuestro proyecto elaboraremos un diagrama de clases aplicando buenas prácticas de diseño orientado a objetos, es importante seguir unos pasos claves y considerar ciertos elementos estructuras y diseños.

25.1 Diagrama de clases para nuestro proyecto

El diagrama de clases define la estructura del sistema, mostrando clases con sus atributos y métodos, así como la relación entre ellas tales como herencias, asociaciones y otros.

En nuestro videojuego las clases que incluiremos son:

5. **Jugador (Player):** Representa al usuario o personaje principal con atributos como salud, posición, puntaje y métodos para moverse o atacar.
6. **Enemigos (Enemy):** Diferentes tipos de enemigos. Con atributos y acciones propias.
7. **PowerUps o ítems:** Las clases que representan objetos que el jugador puede recoger.
8. **Nivel (Level):** Controlamos el escenario, enemigos y condiciones de finalización.
9. **Controlador del juego (GameController):** Maneja la lógica general y el estado del juego.

25.2 Buenas prácticas de diseño orientado a objetos aplicados

1. Encapsulamiento con atributos privados o protegidos con método público de acceso.
2. Uso de herencias para reutilización, por ejemplo, la clase Enemy como base para tipos específicos de enemigo.
3. Polimorfismo para manejar diferentes comportamientos a través e interfaces o métodos sobrecargados.
4. Aplicación de patrones de diseño como Factory para crear enemigos o ítems, y Observer para eventos del juego.
5. Nombres claros y constantes para clases, atributos o métodos.
6. Separación lógica de responsabilidad según el principio de responsabilidad única.

Estas prácticas aseguran que el diagrama refleje un diseño sólido, mantenible y escalable para nuestro videojuego.

25.3 Diagrama

A continuación, revisaremos el diagrama visual:

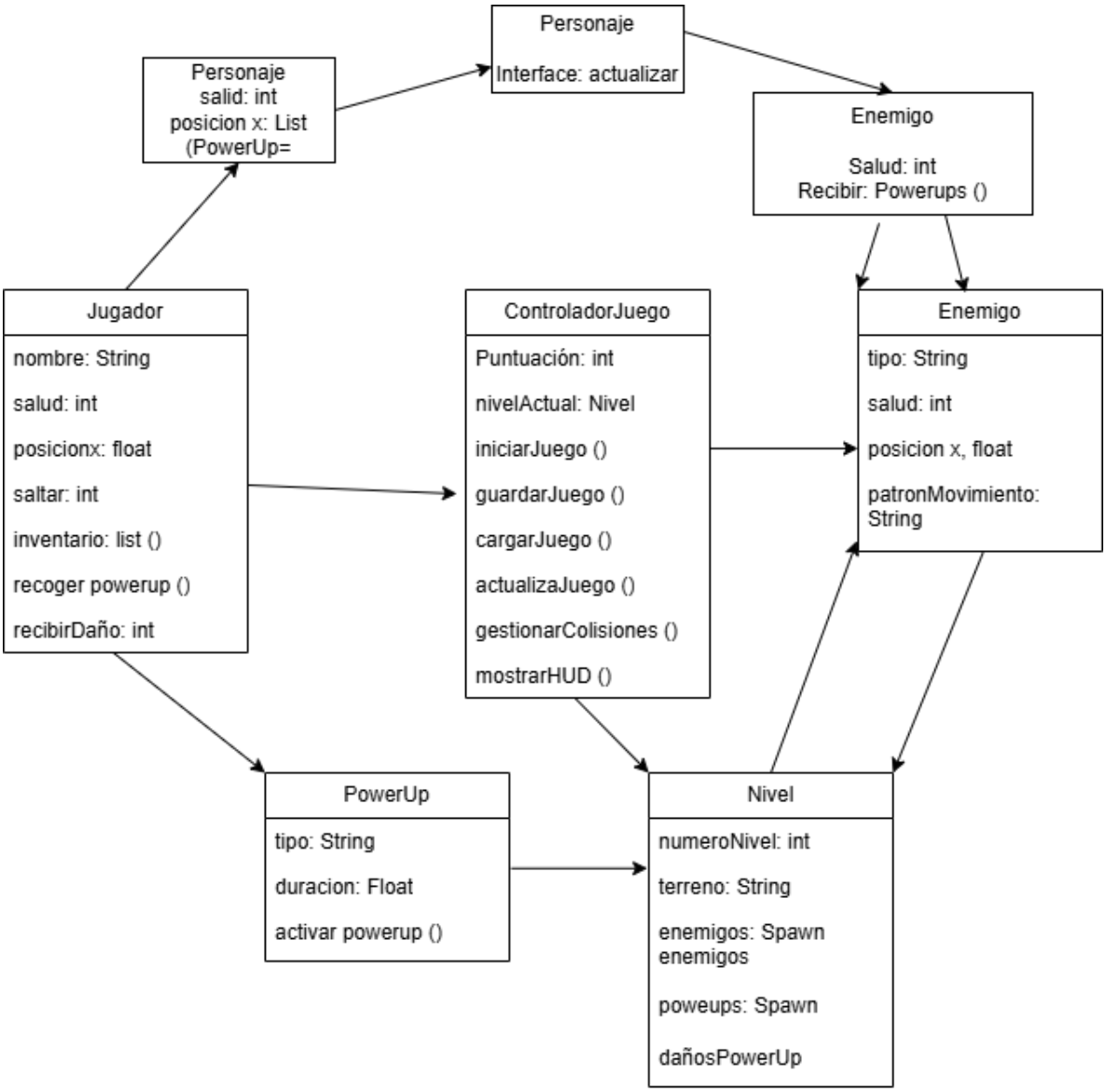


Ilustración 15 Diagrama de Clases

26. Desarrollar la arquitectura de software de acuerdo con el patrón de diseño seleccionado

La arquitectura de Software para nuestro videojuego debe basarse en patrones de diseño, principios de mantenibilidad, vistas de componentes y despliegue con herramientas adecuadas para garantizar escalabilidad y calidad, desarrollaremos una arquitectura completa.

26.1 Incorporación de patrones de diseños y buenas practicas

En nuestro proyecto tendremos:

1. **Patrón MVC (Model-View-Control):** Separación clara entre la lógica de datos, la presentación y el control del flujo del juego.
2. **Patrón Observer:** Notificación automática de cambios de estado entre los componentes del juego.
3. **Patrón Factory:** Creación centralizada de objetos del juego (Enemigos, Power-Ups, Efectos)
4. **Patron State:** Gestión de estado del juego (Menú, jugando, pausa y Game Over)

En nuestro software las buenas prácticas implementadas serán:

1. **Código Limpio:** Nombres descriptivos, funciones pequeñas y responsabilidad única.
2. **Testing Automático:** Pruebas unitarias de integración para cada componente.
3. **Documentación:** Comentarios claros y documentación técnica actualizada.
4. **Versionado:** Control de versiones con Git y flujos de trabajo definido.

26.2 Vista de componentes del Software

Los componentes del software de nuestro juego son:

1. **Capa de Presentación**
   - Game UI: Interfaz de usuario principal.
   - Hud Manager: Gestión de elementos HUD.
   - MenuStyle: Sistema de menú y navegaciones.
2. **Capa de lógica del juego**
   - GameManager: Control principal de juego.
   - PlayerController: Contorl de Personaje Principal.
   - EnemyAI: Inteligencia Artifical de los Enemigos.
   - LevelManager: Gestion de Niveles y progresión.
3. **Capa de Renderizado**
   - GraphicEngine: Motor de Renderizado 2D.
   - AnimationSystem: Sistema de Animaciones.
   - ParticleSystem: Efectos de Partículas
4. **Capa de Datos**
   - SaveSystem: Sistema de Guardado.
   - DataManager: Gestion de datos del Juego.
   - ConfigManager: Configuraciones y ajustes.

26.3 Vista del despliegue del Software

La vista de despliegue de nuestro juego es:

1. **Cliente (Dispositivo de Usuarios):**
2. **Servidor (Si más adelante decidimos crear multijugador):** Game server
3. **Plataforma de Distribución:** Steam, Epic Games Store, Itch.io, Nintendo Switch, PlayStation 5, Xbox.

26.4 Herramientas para optimizar los procesos

En nuestro videojuego tendremos las siguientes herramientas para optimizar los procesos:

1. Herramientas de Desarrollo tales como Unity/Unreal Engine, Visual Code Studio como editor con extensiones para desarrollo y Blender para Modelado 3D y animación.
2. Las herramientas de análisis que usaremos es Unity Profiler para el análisis en tiempo real, también Memory Profile para un análisis de uso de memoria y por último Frame Debugger para análisis Frame por Frame.
3. Para las herramientas de control de versiones usaremos Git para control de versiones distribuidas, GitHub para colaboración CI/CD y Git LFS para manejo de archivos grandes.
4. Las herramientas de Testing serán Unity Test Runner para el Testing automático integrado, también el PlayModeTests para pruebas en modo de juego y por último el Device Testing para pruebas en múltiples dispositivos.

27. Arquitectura de Software

En esta sección revisarnos la arquitectura de Software de nuestro proyecto, revisando primero algunos conceptos claves.

27.1 ¿Qué entiende por arquitectura de software?

La arquitectura de Software es la estructura y diseño de sistema de un software a alto nivel, define como los componentes del software se organizan, interactúan y cumplen con los requisitos funcionales y no funcionales, como la escalabilidad y el rendimiento.

Es el plano que guía la construcción del sistema y facilita la comprensión y colaboración entre los desarrolladores.

En nuestro proyecto la arquitectura es la estructura organizativa que define como se organizan y comunican los diferentes componentes y subsistemas del juego, como gráficos, físicas, inteligencia artificial y sonidos.

Esta arquitectura permite que el videojuego funcione de manera eficiente, escalable y mantenible durante su desarrollo y evolución.

27.2 ¿Cuál es su función?

La principal función de la arquitectura de Software es:

1. **Definir Estructura:** Establece la organización de componentes y sus relaciones.
2. **Facilitar Comunicaciones:** Permite la interacción entre diferentes módulos del sistema.
3. **Garantizar Escalabilidad:** Asegura que el sistema pueda crecer y adaptarse.
4. **Optimizar Rendimiento:** Mejora del sistema y velocidad del sistema.

27.3 ¿Cómo se elabora la arquitectura de software?

En nuestro videojuego aplicaremos la arquitectura del software así:

1. **Análisis de Requisitos:** Identificar y analizar los requisitos funcionales y no funcionales del sistema.
2. **Identificación de componentes:** Definir los módulos principales y sus responsabilidades.
3. **Diseño de Interacciones:** Establecer como se comunican los componentes entre sí.
4. **Selección de patrones:** Elegir diseño de patrones adecuados para el proyecto.
5. **Documentación:** Crear diagrama y documentación de la arquitectura.

27.4 ¿Cómo lograr una buena arquitectura?

En nuestro juego realizaremos una buena arquitectura con:

1. **Modularidad:** Dividir el sistema en módulos independientes y actualizados.
2. **Reutilización:** Diseñar componentes que puedas ser reutilizados.
3. **Robustez:** Crear un sistema resistente a fallos y errores.
4. **Mantenibilidad:** Facilita la modificación y actualización del código.
5. **Equilibrio:** Balancear rendimiento, escalabilidad y complejidad.
6. **Documentación:** Mantener documentación clara y actualizada

27.5 ¿Cuáles son los elementos de diseño de una arquitectura de software?

Los componentes de diseño de nuestro juego son:

1. **Componentes:** Unidades modulares que encapsulaban funcionalidades específicas.
2. **Conectores:** Mecanismos que permiten la comunicación entre componentes.
3. **Configuraciones:** Arreglo estructural de componentes y conectores.
4. **Restricciones:** Reglas y limitaciones que rigen el diseño.
5. **Atributos de Calidad:** Propiedades no funcionales como rendimiento, seguridad y escalabilidad.

27.6 Aplicación en nuestro proyecto

En nuestro videojuego la arquitectura de software es la estructura y diseño de un sistema de software a nivel alto, nos define componentes del software que se organiza, interactúan y cumplen con los requisitos funcionales y no funcionales, como la escalabilidad y el rendimiento.

Arquitectura de nuestro videojuego:

6. **Capa de Presentación:** Interfaz de usuario, menús, HUD y elementos visuales.
7. **Capa lógica de Juego:** Mecánicas de juego, reglas de negocio y control de flujo.
8. **Capas de renderizado:** Motor gráfico, animaciones y efectos visuales.
9. **Capa de Audio:** Música, efectos de sonido y gestión de audio.
10. **Capa de Datos:** Almacenamiento, guardado de partidas y persistencia.

Es como el plano maestro que nos guía la construcción del sistema.

28. Validación de Documentos

En esta sección revisaremos la validación de los documentos de nuestros videojuegos, mejorando la validación anterior de requisitos o documentos que se presentaron para ellos revisaremos los artefactos de nuestro proyecto, los tipos de artefactos para nuestro videojuego y la evaluación para nuestro videojuego, entre muchos otros, todos estos instrumentos nos permiten verificar el estado de desarrollo de nuestro videojuego y tomar acciones a tiempos de manera más detallada.

28.1 ¿Qué es un artefacto?

"Un **artefacto** es cualquier producto tangible o intangible generado durante el proceso de desarrollo de software. Puede incluir documentos, código fuente, modelos, plantillas, diagramas o incluso bases de datos. En términos generales, representa un elemento que captura y comunica información dentro del ciclo de vida del software.

Ejemplos: especificaciones de requisitos, diagramas UML, manuales de usuario, reportes de pruebas o scripts de instalación."[22]

"En el desarrollo de software, un artefacto de software es un elemento que se produce durante el proceso de desarrollo. Puede ser un modelo de datos, un prototipo, un diagrama de flujo de trabajo, un documento de diseño o un script de configuración. De hecho, existen artefactos específicos que se requieren durante un ciclo de desarrollo y que deben almacenarse de forma accesible."[23]

En nuestro proyecto todo producto generado durante un ciclo de desarrollo ya sea técnico, artístico o de diseño, incluyen tanto documentos conceptuales como elementos digitales del propio juego los cuales son:

7. Documento de diseño (GDD: Game Desing Document)
1. Modelos 3D y texturas
2. Scripts de programación (Mecánicas y lógica de programación)
3. Archivos de audio, música y voces
4. Niveles, escenarios y prototipos jugables

[22] Texto Generado por IA - Perplexity

[23] https://www-leanix-net.translate.goog/en/wiki/trm/software-artifacts?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

28.2 ¿Tipos de artefactos?

1. **“Código fuente:** El código real escrito por desarrolladores en lenguajes de programación como Java, Python o JavaScript. Constituye la base de cualquier aplicación de software.
2. **Binarios compilados:** archivos ejecutables generados al compilar el código fuente. Son las versiones ejecutables de la aplicación.
3. **Bibliotecas:** módulos de código reutilizables que proporcionan funcionalidades específicas. Las bibliotecas pueden compartirse entre diferentes proyectos para evitar redundancias.
4. **Archivos de configuración:** archivos que definen la configuración y los parámetros de la aplicación. Garantizan su correcto funcionamiento en diferentes entornos.
5. **Imágenes de contenedor:** Aplicaciones empaquetadas y sus dependencias en una única unidad portátil. Las imágenes de Docker son un ejemplo común, utilizadas para una implementación consistente en diferentes entornos.
6. **Documentación:** documentos, especificaciones y comentarios de código que proporcionan contexto y orientación para el desarrollo y uso del software.
7. **Scripts de compilación:** scripts que compilan el código fuente en binarios ejecutables para garantizar la coherencia en el proceso de compilación.
8. **Scripts de implementación:** scripts automatizados que manejan la instalación y configuración de la aplicación en entornos de producción.
9. **Casos de prueba:** procedimientos utilizados para verificar que el software se comporta según lo previsto. Son cruciales para el control de calidad.
10. **Registros:** registros del comportamiento y el rendimiento de la aplicación para la resolución de problemas y la optimización.”[24]

En nuestro proyecto necesitaremos los siguientes artefactos:

1. Artefactos de diseño: La cual nos da descripciones de mecánicas, niveles, narrativas, reglas, interfaces y progresión del juego.
2. Artefactos artísticos: Modelos 3D, Sprites, Efectos visuales, animaciones y arte conceptual.
3. Artefactos de Programación: Scripts en motores como Unity o Unreal, control de físicas, Inteligencia Artificial o las interacciones.
4. Artefactos de sonido: Música de ambientación, también los efectos de sonido, voces y mezclas de audio.
5. Artefactos de Prueba: Reporte de testeo, lista de errores y evaluaciones de rendimiento.

[24] https://jfrog-com.translate.goog/learn/devops/software-artifact/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

28.3 ¿Qué es la evaluación de artefactos?

“La evaluación de artefactos es un proceso mediante el cual se **evalúa** el impacto de un **cambio** en un **software** existente. Se realiza mediante el **análisis de los artefactos** generados durante el **desarrollo del software**.

Los resultados de la evaluación de artefactos permiten **determinar el impacto** de un cambio en el **software** y el **grado de madurez** del mismo.”[25]

En nuestro videojuego tenemos que revisar, probar y validar los recursos generados para asegurar que funcionan correctamente, sean coherentes con la visión del juego y proporcionen buena experiencia al jugador lo que significa:

1. Jugabilidad y mecánicas
2. Consistencia visual y artística
3. Calidad de audio y sincronización
4. Interfaz y respuesta al usuario
5. Rendimiento del motor de juego

28.4 ¿Cómo se realizan?

El proceso de evaluación suele incluir los siguientes pasos:

1. **Revisión del artefacto:** análisis detallado del contenido, estructura y cumplimiento de criterios.
2. **Identificación de no conformidades:** detección de errores, omisiones o inconsistencias.
3. **Registro de observaciones:** documentación de comentarios y sugerencias de mejora.
4. **Corrección y seguimiento:** ajuste del artefacto y validación de las correcciones.[26]

En nuestro proyecto el proceso de validación sería el siguiente:

- Pruebas Funcionales y de jugabilidad: Debido a que necesitamos revisar las mecánicas respondan correctamente y sin errores.
- Revisión visual y artística: Se asegura que el arte y las animaciones mantenga coherencia de estilo.

[25] https://brainly.lat/tarea/65663185
[26] Texto Generado por IA - Perplexity

- Pruebas de Rendimiento: Se evalúan los Frames por segundo (FPS), temperatura de la GPU y estabilidad general.
- Prueba de usuario (Playtesting): Se analiza la satisfacción del jugador y su interacción con la interfaz.
- Revisión entre equipos: Artistas, Programadores y diseñadores validan los artefactos propios y cruzados.

28.5 ¿Qué instrumentos se utilizan?

“Los instrumentos o herramientas más comunes para la validación de artefactos son:

- **Listas de verificación:** guías con criterios o preguntas específicas que ayudan a controlar la calidad del artefacto.
- **Plantillas:** estructuras estandarizadas que garantizan la uniformidad de la documentación.
- **Guías o manuales de evaluación:** documentos que ofrecen procedimientos o criterios calificados.
- **Herramientas de análisis estático:** detectan errores de código sin necesidad de ejecutarlo (por ejemplo, SonarQube o PMD).
- **Herramientas de pruebas dinámicas:** validan el comportamiento del software frente a los requisitos mediante pruebas funcionales o de integración.”[27]

En nuestro videojuego usaremos:

- Lista de chequeo (Lista QA): Que definen criterios técnicos y artísticos
- Herramientas de prueba automatizada: Por ejemplo, Unity Test Runner o Unreal Automation Tool.
- Benchmark Visuales: 3DMark o Unigine Superposition para medir el rendimiento gráfico.
- Tracker de errores: Como Jira, Trello o Bugzilla
- Sistema de control de Versiones: Como Git, Perforce o Plastic SCM para asegurar la integridad y trazabilidad
- Plataforma de testeo masivo: Como game Tester para que nos dé una buena retroalimentación de jugadores.

[27] Texto Generado por IA - Perplexity

28.6 ¿Qué resultados se obtienen?

Para nuestro videojuego se producen resultados medibles que garanticen la calidad de nuestro juego con los siguientes resultados:

1. Confirmación funcional: Las mecánicas y sistemas operas según lo que se planteó en el documento y en las reuniones
2. Uniformidad visual y de sonido: Coherencia entre el arte, animación y audio
3. Lista de incidencias: Registro de Bugs y su estado de Corrección
4. Informes de rendimiento y estabilidad: FPS, carga de CPU y GPU, uso de memoria
5. Feedback del jugador: Datos reales sobre la experiencia de juego y la diversión.

29. Instrumentos para verificación de artefactos

A continuación, revisaremos más a fondo la verificación de los artefactos de nuestro videojuego, para ello crearemos un instrumento para poder hacer una evaluación y poder controlar la calidad de nuestro videojuego, esto es importante para garantizar que los artefactos cumplan con los estándares establecidos y los requisitos específicos del proyecto.

“Los instrumentos de verificación son formatos de control elaborados para registrar mediciones o el cumplimiento total o parcial de tareas, acciones, funciones o el desempeño de actividades particulares en sistemas, procesos o procedimientos. Según Romero Alvarado (2018), estos formatos "son herramientas necesarias para garantizar la calidad de los productos ya sea por medición, verificación o control"[28]

Los componentes de nuestro proceso son:

1. **Definición de Objetivos**: Es necesario establecer claridad sobre los objetivos específicos de la verificación, determinado que aspectos de los artefactos se evaluaran y que preguntas se pretenden responder.
2. Id**entificación de criterios**: Se debe definir los criterios de evaluación, que debe incluir precisión, confiabilidad, validez y consistencia de los artefactos.
3. **Selección de instrumentos**: La elección de las herramientas apropiadas es crucial, pudiendo incluir cuestionarios, escalas de mediciones, pruebas y observaciones.

Algunos tipos que podemos tener en cuenta son:

1. Herramienta de análisis estático
2. Herramienta de prueba unitaria
3. Metodología de implementación
4. Consideraciones de calidad
5. Técnicas de verificación aplicada

Y por último debemos incluir las actividades de riesgo, pruebas de seguridad, análisis de código estático y cobertura, además de tares específicas para sistemas críticos como los regulados por la ISO 26262

[28] Texto Generado por IA - Perplexity

29.1 Ejemplo de Instrumento

Como parte del manual de software le solicitamos a la Inteligencia artificial un ejemplo el cual nos mostró lo siguiente:

Ítem	Descripción	Cumple (Sí/No)	Observaciones
Portada y datos del documento	¿El documento tiene portada completa con título, fecha y autor?		
Introducción	¿Incluye una introducción clara del propósito y alcance del documento?		
Cumplimiento de requisitos	¿El artefacto cumple con todos los requisitos especificados?		
Claridad y lenguaje	¿Está redactado en lenguaje sencillo y con buena ortografía?		
Compleción de contenido	¿Contiene toda la información necesaria para la verificación?		
Formato y presentación	¿El formato es amigable y facilita su lectura y uso?		
Resultados de pruebas y validación	¿Incluye resultados claros y conclusiones fundamentadas?		

29.2 Aplicación en nuestro proyecto

En nuestro proyecto la aplicación del formato que usaremos es:

RS Estudios				
Código	A-001			
Software	Videojuego “Videojuego Secreto”			
Fecha	22/10/2025			
Responsable	Pepito Pérez			
Aspecto Que Verificar	**Descripción**	**Si**	**No**	**Observaciones**
Diseño visual y UI/UX	Los menús, botones y Tipografía son legibles y coherentes con la estética de nuestro juego	X	X	
Fluidez del Gameplay	Las animaciones se ejecutan correctamente sin bugs visibles o interrupciones	X	X	
Pruebas de rendimiento	El juego corre a la tasa de fotogramas (FPS) estable en varias plataformas	X	X	
Audio	Los efectos de sonido, diálogos (si tenemos) y música estén sincronizado y libres de distorsiones	X	X	
Jugabilidad Funcional	Las mecánicas del juego como movimiento, salto, combate y recolección de ítems funcionan correctamente	X	X	
Gestión de errores	El juego maneja adecuadamente errores o cierres inesperados, mostrando mensajes de errores que nos sean útiles	X	X	
Pantalla de inicio y guardado de partida	Las opciones del menú como nuevo juego, cargar una partida las configuraciones y la opción de salir funcionan sin problema	X	X	
Compatibilidad	El juego se ejecuta correctamente en difernetes sistemas operativos o plataformas	X	X	
Localización y textos	Los textos esta traducidos correctamente en caso de que los vendamos al extranjero y sin errores ortográficos o culturales	X	X	
Pruebas de estrés	Pondremos el juego a prueba con diferentes cosas tales como: exceso de enemigo, un jugador que hace acciones inesperadas, efectos raros, etc.	X	X	
Cumplimientos de requisitos	Los artefactos implementan todas las funcionalidades del documento de requisitos iniciales	X	X	
Firma de Aprobación				

29.3 Listas de Chequeo para validación de documentos de diseño.

En esta sección revisaremos las listas de chequeo, pero mejoradas, con listas de chequeo para la validación de documentos de diseño específicos del desarrollo de videojuegos, asegurando la calidad y consistencia de cada fase del proyecto.

Los objetivos de las listas que de chequeo para validación de documentos de diseño de nuestro videojuego son herramientas fundamentales que nos permiten

1. **Control de calidad:** Garantizar que todos los documentos cumplan con los estándares establecidos y mantengan la coherencia visual y funcional del proyecto.
2. **Estandarización:** Establecer procesos uniformes de revisión y validación que aseguren la consistencia en la documentación del videojuego.
3. **Eficiencia:** Optimizar el tiempo de revisión mediante listas estructuradas que cubran todos los aspectos críticos del diseño.
4. **Especialización:** Adaptar los procesos de validación a las particularidades específicas del desarrollo de videojuegos.

29.3.1. Importancia en el Desarrollo de nuestro videojuego

En el desarrollo de nuestro videojuego, la validación de documentación de diseño es especialmente critica debido a:

1. **Complejidad Multidisciplinaria**: Los videojuegos requieren la coordinación de múltiples disciplinas tales como arte, programación, audio, diseño y narrativa que deben trabajar de manera integrada y coherente.
2. **Iteración constante:** El desarrollo de videojuegos es un proceso que es demasiado iterativo donde los documentos de diseño evolucionan constantemente, requiriendo validación continua.
3. **Experiencia del usuario:** La calidad de la experiencia del jugador depende directamente de la coherencia y la calidad de todos los elementos de diseño.
4. **Plazos de Entrega:** Los proyectos de videojuegos tienen cronogramas ajustados que requieren procesos de validación eficientes para evitar retrasos costosos.
5. **Múltiples Plataformas:** La necesidad de adaptar el juego a diferentes plataformas requiere documentación especifica y validación de compatibilidad.

29.3.2. Metodología de Aplicación

Las listas de chequeo se aplican siguiendo una metodología estructurada:

1. **Fase de Preparación:** Selección de la lista de chequeo apropiada según el tipo de documento y fase del proyecto.
2. **Fase de Revisión:** Evaluación sistemática de cada elemento de la lista, marcando el cumplimiento y documentando observaciones.
3. **Fase de Análisis:** Cálculo de porcentaje de cumplimiento e identificación de áreas que requieren atención.
4. **Fase de Validación:** Firma de aprobación por parte de los responsables correspondientes una vez cumplidos todos los criterios.

29.3.3. Beneficios de la implementación

La implementación de estas listas de chequeo proporciona beneficios tangibles al proyecto:

1. **Reducción de errores:** Disminuye significativamente la probabilidad de errores en la documentación y el diseño del juego.
2. **Ahorro de tiempo:** Optimiza los procesos de revisión, reduciendo el tiempo necesario para validar documentos.
3. **Mejor comunicación:** Facilita la comunicación entre equipos al proporcionar criterios claros y objetivos.
4. **Mejora Continua:** Permite identificar patrones de sistema y mejora los procesos de desarrollo.
5. **Control de calidad:** Establece un estándar mínimo de calidad que debe cumplir toda la documentación del proyecto.
6. **Documentación Histórica:** Crea un registro historia de las validaciones realizadas para futuras referencias.

29.3.4. Aplicación en nuestro Proyecto

Para nuestro proyecto, estas listas de chequeo son especialmente valiosas ya que:

1. **Garantizar la coherencia visual:** Aseguran que todos los elementos artísticos mantengan el estilo pixel art y la estética retro definida para el juego.
2. **Optimizar el desarrollo indie:** Proporciona estructura y organización a un equipo de desarrollo independiente con recursos limitados.

3. **Facilitan la iteración:** Permiten evaluar rápidamente los cambios y mejoras propuestas durante el desarrollo.
4. **Aseguran la calidad multiplataforma:** Verifican que el juego funcione correctamente en PC, consolas y dispositivos móviles.
5. **Preparan para la publicación:** Garantizan que todos los documentos estén listos para el proceso de certificación y lanzamiento.

29.3.5. Lista de chequeo 1: Documentos del diseño Conceptual

A continuación, veremos la lista de chequeo numero 1

Ítem	Descripción	Cumple	Observaciones
1.1	Documento de Visión del Juego (GDD) está completo y actualizado	☐	
1.2	Arte conceptual de personajes principales está definido y aprobado	☐	
1.3	Diseño de niveles y ambientes está documentado visualmente	☐	
1.4	Paleta de colores y estilo visual está establecida y es consistente	☐	
1.5	Storyboard o guion visual de secuencias clave está completo	☐	
1.6	Referencias de arte y modo boards están organizadas	☐	
1.7	Documentación de UI/UX incluye wireframes y mockups	☐	
1.8	Especificaciones técnicas de arte están definidas (resoluciones, formatos)	☐	

29.3.6. Lista de chequeo 2: Documentos de Mecánicas de Juego

A continuación, veremos la lista de chequeo numero 2

Ítem	Descripción	Cumple	Observaciones
2.1	Mecánicas principales del juego están documentadas y probadas	☐	
2.2	Sistema de controles está definido para todas las plataformas objetivo	☐	
2.3	Balance de dificultad está calculado y documentado	☐	
2.4	Sistema de progresión del jugador está especificado	☐	
2.5	Mecánicas de recompensas y logros están implementadas	☐	
2.6	Sistema de guardado y carga está funcional	☐	
2.7	Mecánicas de multijugador (si aplica) están documentadas	☐	
2.8	Tutorial y sistema de ayuda están integrados	☐	

29.3.7. Lista de chequeo 3: Documentos Técnicos

A continuación, veremos la lista de chequeo numero 3

Ítem	Descripción	Cumple	Observaciones
3.1	Arquitectura del software está documentada con diagramas	☐	
3.2	Especificaciones de rendimiento están definidas (FPS, memoria, etc.)	☐	
3.3	Requisitos de hardware están especificados para cada plataforma	☐	
3.4	API y servicios externos están documentados	☐	
3.5	Estructura de base de datos está diseñada y documentada	☐	
3.6	Flujo de datos del juego está mapeado	☐	
3.7	Procedimientos de backup y recuperación están definidos	☐	
3.8	Documentación de APIs para mods o contenido adicional	☐	

29.3.8. Lista de chequeo 4: Documentos de Audio y música

A continuación, veremos la lista de chequeo numero 4

Ítem	Descripción	Cumple	Observaciones
4.1	Banda sonora principal está compuesta y masterizada	☐	
4.2	Efectos de sonido están categorizados y organizados	☐	
4.3	Diálogos están grabados y sincronizados correctamente	☐	
4.4	Niveles de audio están balanceados y normalizados	☐	
4.5	Implementación de audio 3D está configurada	☐	
4.6	Opciones de configuración de audio están implementadas	☐	
4.7	Licencias de música y efectos están documentadas	☐	
4.8	Archivos de audio están optimizados para diferentes plataformas	☐	

29.3.9. Lista de chequeo 5: Documentos de Testing y QA

A continuación, veremos la lista de chequeo numero 5

Ítem	Descripción	Cumple	Observaciones
5.1	Plan de pruebas está documentado y aprobado	☐	
5.2	Casos de prueba para todas las mecánicas principales están definidos	☐	
5.3	Pruebas de compatibilidad en diferentes dispositivos están completas	☐	
5.4	Pruebas de estrés y rendimiento están ejecutadas	☐	
5.5	Pruebas de usabilidad con usuarios reales están realizadas	☐	
5.6	Reportes de bugs están categorizados y priorizados	☐	
5.7	Pruebas de accesibilidad están implementadas	☐	
5.8	Documentación de regresiones y fixes está actualizada	☐	

29.3.10. Lista de chequeo 6: Documentos de Localización

A continuación, veremos la lista de chequeo numero 6

Ítem	Descripción	Cumple	Observaciones
6.1	Textos del juego están extraídos para traducción	☐	
6.2	Idiomas objetivo están definidos y priorizados	☐	
6.3	Elementos culturales sensibles están identificados	☐	
6.4	Interfaz de usuario está adaptada para diferentes idiomas	☐	
6.5	Audio localizado está grabado y sincronizado	☐	
6.6	Pruebas de localización están ejecutadas	☐	
6.7	Documentación de localización está actualizada	☐	
6.8	Certificaciones regionales están obtenidas	☐	

29.3.11. Resumen de validación

Para Finalizar nuestra validación debemos tener algunos ítems claros los cuales son:

1. **Total de ítems**
2. **Ítems cumplidos**
3. **Porcentaje de cumplimiento**
4. **Fechas de Validación**

Y por último una ficha de validación de documentos de diseño donde aparecen los responsables de diseño. Responsable técnico y de QA.

29.4 Informe de evaluación a los artefactos de diseño

El informe de evaluación de artefactos de diseño del software es un documento fundamental de nuestro proyecto, ya que proporciona una evaluación integral y sistemática de todos los elementos de diseño del desarrollados durante el ciclo de vida del software

Esto nos garantiza que cada artefacto cumpla con los estándares de calidad establecidos y contribuya a que nuestro juego tenga una excelente calidad.

La evaluación sistemática de artefactos de diseño es crucial para mantener la coherencia, calidad y eficiencia en el desarrollo de software.

Permite identificar fortalezas, debilidades y oportunidades de mejora en cada componente del sistema.

29.4.1 Objetivos del informe de evaluación

Los objetivos de realizar este informe son:

1. **Evaluación integral:** Realizar una revisión completa de todos los artefactos de diseño para identificar su calidad, coherencia y cumplimiento de requisitos.
2. **Medición de Progreso:** Establecer métricas cuantitativas y cualitativas para medir el avance del proyecto y la calidad de los entregables.
3. **Identificación de Riesgos:** Detectar tempranamente problemas potenciales que puedan afectar el cronograma Presupuesto o calidad del proyecto.
4. **Mejora continua:** Proporcionar recomendaciones especificas para optimizar procesos y mejorar la calidad de futuros artefactos.

29.4.2 Metodología de Evaluación

La evaluación de los artefactos de diseño del software sigue una metodología estructurada que garantiza la objetividad y consistencia

1. **Preparación y planificación**
   - Definición de criterios de evaluación específicos
   - Selección de herramientas y técnicas de análisis
   - Asignación de evaluadores especificados
   - Establecimiento de cronograma de evaluación
2. **Análisis individual de Artefactos**
   - Revisión detallada de cada artefacto según criterios establecidos
   - Documentación de hallazgo y observaciones
   - Asignación de puntuaciones cuantitativas
   - Identificación de fortalezas y debilidades
3. **Evaluación integrada**
   - Análisis de coherencia entre artefactos relacionados
   - Verificación de cumplimiento de requisitos globales
   - Evaluación de la integración y compatibilidad
   - Análisis del impacto en el sistema completo
4. **Consolidación y reporte**
   - Agregar resultados individuales
   - Generación de métricas consolidadas
   - Elaboración de recomendaciones priorizadas
   - Preparación del informe final

29.4.3 Criterios de Evaluación

Los artefactos de diseño del software son evaluados según criterios específicos adaptados a las particularidades del desarrollo de videojuegos

1. **Criterios de diseño visual**
   - Coherencia visual: Consistencia en el estilo, paleta de colores, tipografías y elementos gráficos a lo largo del juego.
   - Usabilidad: Facilidad de uso, navegación intuitiva y accesibilidad para diferentes tipos de usuarios.
   - Rendimiento Visual: Optimización de recursos gráficos para mantener un rendimiento fluido en diferentes plataformas.
2. **Criterios Técnicos**
   - Arquitectura: Diseño modular, escalable y mantenible que facilite futuras actualizaciones y modificaciones.
   - Funcionalidad: Cumplimiento de todos los requisitos funcionales especificados en la documentación del proyecto.
   - Seguridad: Implementación de medidas de seguridad apropiadas para proteger datos y funcionalidad criticas
3. **Criterios específicos de Videojuegos**
   - Gameplay: Diseño de mecánicas de juego equilibradas, divertidas y que mantengan el interés del jugador.
   - Audio: Integración efectiva de música, efectos de sonido y audio que complementen la experiencia del juego.
   - Multiplataforma: Adaptación exitosa del diseño para diferentes plataformas manteniendo la calidad y la funcionalidad.

29.4.4 Herramientas de Evaluación

Para la evaluación efectiva de los artefactos de diseño, se utilizan diversas herramientas especializadas

1. **Herramientas de análisis estático**
   - SonarQube: Análisis de calidad del código, detección de bugs y vulnerabilidades de seguridad.
   - ESLint/Stylelint: Verificación de estándares de codificación y consistencia en el estilo del código.
2. **Herramientas de Testing**
   - Jest/Mocha: Framework de Testing para pruebas unitarias y de integración del código.
   - Unity Test Runner: Herramienta integrada para pruebas específicas de videojuegos desarrollador en Unity

3. **Herramientas de métricas**
   - Unity Profiler: Análisis de rendimiento en tiempo real para identificar cuellos de botella.
   - Google Analytics: Análisis de comportamiento del usuario y métricas de engagement.

29.4.5 Métricas de Evaluación

El informe de evaluación incluye métricas cuantitativas y cualitativas que proporcionan una visión objetiva del estado de los artefactos

1. **Métricas Cuantitativas**
   - Porcentaje de cumplimiento: Porcentaje de requisitos funcionales y no funcionales que han sido implementados correctamente.
   - Densidad de Defectos: Número de defectos encontrados por unidad de código o por artefacto evaluado.
   - Rendimiento: Métricas de FPS, tiempo de carga, uso de memoria y otros indicadores de rendimiento.
   - Cobertura de Pruebas: Porcentaje de código cubierto por pruebas automatizadas y manuales.
2. **Métricas Cualitativas**
   - Satisfacción del usuario: Evaluación de la experiencia del usuario basada en pruebas de usabilidad y feedback.
   - Calidad visual: Evaluación subjetiva de la calidad artística y coherencia visual del juego.
   - Mantenibilidad: Facilidad para realizar modificaciones, actualizaciones y correcciones en el código.
   - Documentación: Completitud, claridad y utilidad de la documentación técnica y de usuario.

29.4.6 Aplicación en nuestro proyecto

Para nuestro videojuego, el informe de evaluación de artefactos de diseño es especialmente importante porque

1. **Estilo Pixel Art:** Verifica la consistencia del estilo visual retro y pixel art a lo largo de todos los artefactos manteniendo la estética nostálgica del videojuego.
2. **Multiplataforma:** Asegura que el diseño funcione correctamente en PC, consolas y dispositivos móviles, adaptándose a las diferentes interfaces de cada plataforma.

3. **Rendimiento Optimizado:** Evalúa que el juego mantenga un rendimiento fluido en dispositivos con especificaciones modestas, importante para el acceso educativo.

29.4.7 Estructura de informe de Evaluación

El informe de evaluación sigue una estructura estándar que facilita la comprensión y el seguimiento de los resultados

1. **Resumen ejecutivo:** Visión general de los resultados, hallazgos principales y recomendaciones prioritarias para la toma de decisiones.
2. **Metodología Aplicada:** Descripción detallada de los criterios, herramientas y y procesos utilizados en la evaluación.
3. **Evaluación de artefactos:** Análisis detallado de cada artefacto individual, incluyendo fortalezas debilidades, y puntuaciones especifica.
4. **Métricas consolidadas:** Tablas y gráficos que representan las métricas cuantitativas y cualitativas de manera visual y comprensible.
5. **Análisis de riesgos:** identificación de riesgos potenciales y su impacto en el proyecto, con estrategias de mitigación.
6. **Recomendaciones:** Lista priorizada de acciones recomendadas para mejorar la calidad y resolver problemas identificados.
7. **Plan de Seguimiento:** Cronograma y responsabilidades para la implementación de las recomendaciones y futuras evaluaciones.

29.4.8 Beneficios del informe de Evaluación

La implementación del informe de evaluación sistemáticos proporciona beneficios significativos al proyecto

1. **Control de calidad:** Garantizar que todos los artefactos cumplan con los estándares de calidad establecidos, y contribuyan al éxito del proyecto.
2. **Visibilidad del progreso:** Proporciona métricas claras del avance y la calidad de los entregables para Stakeholders y equipos.
3. **Detección temprana:** Identifica problemas potenciales antes de que se conviertan en obstáculos mayores ahorrando tiempo y recursos.
4. **Mejora continua:** Facilita la identificación de oportunidades de mejora y la optimización de procesos de desarrollo.
5. **Documentación histórica:** Crea un registro detallado de la evolución del proyecto para futuras referencias y lecciones aprendidas.

6. **Comunicación efectiva:** Facilita la comunicación entre equipos y Stakeholders mediante reportes estructurados y objetivos.

29.4.9 Conclusión

El informe de evaluación de artefactos de diseño del software es una herramienta fundamental que asegura la calidad, coherencia y éxito de nuestro juego, mediante la aplicación sistemática de criterios de evaluación, herramientas especializadas y métricas objetivas, este proceso garantiza que cada componente del videojuego contribuya efectivamente al objetivo y la experiencia final del usuario.

La implementación de este informe no solo mejora la calidad del producto final, sino que también establece un marco de trabajo para la mejora continua y la optimización de procesos de desarrollo asegurando que nuestro juego cumpla con todos los estándares de calidad y le guste a nuestro usuario final.

30. Elaborar el prototipo navegable del software aplicando estándares de usabilidad y accesibilidad

31. Crear un modelo de base de datos con base a los requerimientos del cliente.

32. Manipular datos en el Sistema Administrador de Bases de Datos (SMBD)

33. Crear interfaces gráficas de usuario en aplicaciones de escritorio, web y móviles

34. Generar plantillas y estilos

35. Configurar herramientas de versionamiento para control de código

36. Aplicar estándares de codificación

37. Codificar los módulos del software Stand-alone, web y móvil

38. Codificar el frontend utilizando Framework

39. Crear servicios web

40. Integrar módulos

41. Incorporar tecnologías emergentes y disruptivas

42. Realizar plan de pruebas

43. Definir casos y ambiente de prueba

44. Documentar y realizar pruebas

45. Preparar la plataforma tecnológica

46. Verificar el cumplimiento de las características mínimas de hardware requeridas para el software desarrollado

47. Elaborar el plan de instalación

48. Configurar los servicios requeridos

49. Configurar el software en el servidor y la base de datos

50. Cargar archivos en el sitio de publicación

51. Realizar pruebas de funcionalidad del software

52. Elaborar planes de mantenimiento y soporte del software

53. Documentar el proceso de migración y respaldo de los datos

54. Elaborar manuales de instalación, técnico y de usuario

55. Elaborar el manual de usuario

56. Capacitar a los usuarios y realizar pruebas de aceptación

57. Elaborar acta de entrega

58. Aplicar los instrumentos de calidad de software

59. Realizar el informe de evaluación de la calidad de software y bitácora de lecciones aprendidas

60. Ajustar procesos del desarrollo de software