DOCUMENTO FUNCIONAL DEL PROYECTO

ALUMNO/GRUPO: Adrian Lara Zafra

---

1. Introducción y contexto

En muchos juegos de acción roguelike o de supervivencia, la jugabilidad tiende a centrarse exclusivamente en la mecánica de la acción sin dar mucha profundidad a la historia o la motivación del jugador. Aunque la jugabilidad puede ser muy adictiva, algunos jugadores buscan una experiencia que combine la acción frenética con una narrativa envolvente que les permita conectar con los personajes y el mundo que están explorando.

El videojuego está dirigido a jugadores que disfrutan de títulos de acción rápida y roguelikes, como *Vampire Survivors*, donde el objetivo es sobrevivir a oleadas de enemigos con un sistema de progresión dinámico. Además, está orientado a aquellos que buscan una historia que se desarrolle durante el juego, ofreciendo un trasfondo narrativo que añade peso emocional a la experiencia de combate.

Se desarrollará un videojuego en *Unity* que combine mecánicas de acción al estilo *Vampire Survivors* con elementos narrativos que se desenvuelven a lo largo del juego. El jugador enfrentará hordas de enemigos, pero también podrá desbloquear fragmentos de una historia que se desarrolla mientras avanza. A medida que el jugador progresa y sobrevive más tiempo, se revelarán más detalles sobre el mundo, los personajes y las motivaciones detrás de las oleadas de enemigos. El propósito es ofrecer una experiencia jugable emocionante y desafiante, en la que los jugadores no solo sientan que están sobreviviendo, sino también que están descubriendo algo más profundo a medida que avanzan.

---

2. Análisis de requisitos

2.1. Requisitos funcionales (RF)

RF1	Se contará con una web donde los jugadores podrán descargar el juego. En esta web habrá una breve explicación de qué trata y también se mostrarán algunas capturas para que el usuario pueda ver cómo es antes

	de jugar.
RF2	El juego debe permitir que el jugador inicie sesión mediante una cuenta personal, para guardar su progreso y personalización. Se deberá ofrecer la opción de iniciar sesión mediante un sistema de cuenta tradicional o, en su caso, un sistema de invitado sin necesidad de registro.
RF3	Una vez iniciada sesión, el jugador deberá poder sincronizar su progreso de juego, configuraciones y personalizaciones (como skins o mejoras de personaje) en la nube. Este sistema permitirá al jugador continuar su partida desde cualquier dispositivo, asegurando que el progreso esté siempre disponible.
RF4	El juego debe permitir el guardado automático del progreso en puntos de control clave (por ejemplo, después de cada ola de enemigos). Además, el jugador podrá realizar guardados manuales en cualquier momento, para poder retomar su partida exactamente donde la dejó.
RF5	El sistema debe proporcionar una experiencia de juego en un entorno 3D, en la cual se generen oleadas de enemigos distribuidas aleatoriamente por el mapa. Al completar cada oleada, el jugador debe poder acceder a mejoras para su personaje. Además, el sistema debe generar un jefe al finalizar cada oleada, que deberá aparecer tras la derrota de los enemigos estándar.
RF6	En el juego se avanzará principalmente superando oleadas de enemigos. Las oleadas consistirán en un número X de enemigos

---

2.2. Requisitos no funcionales (RNF)

RNF 1	El juego debe ser compatible con Windows, funcionando correctamente en las versiones más recientes y al menos en dos versiones anteriores de los sistemas operativos.
RNF 2	La interfaz del juego debe ser intuitiva y accesible, permitiendo que un nuevo jugador entienda cómo jugar sin necesidad de tutoriales extensos. Los menús y opciones deberán ser fácilmente navegables.
RNF 3	El juego deberá soportar múltiples idiomas (por ejemplo, español e inglés) para llegar a un público más amplio, y facilitar la incorporación de más idiomas en futuras actualizaciones.
RNF 4	El juego debe funcionar de manera estable en las plataformas soportadas, manteniendo un rendimiento fluido (mínimo 60 FPS) incluso en situaciones de alta carga gráfica, como cuando se generan grandes cantidades de enemigos o efectos visuales.

2.3. Restricciones

- Lenguajes o tecnologías obligatorias:

El videojuego se desarrollará en Unity, usando C# como lenguaje principal para programar mecánicas, enemigos y progresión. Este proyecto será mi primera experiencia con C#, por lo que también implicará un proceso de aprendizaje del lenguaje.

Se usarán herramientas complementarias como Visual Studio para la edición de código y programas básicos de diseño gráfico (GIMP, Photoshop o Canva) para los recursos visuales. También se podrán integrar recursos gratuitos de Unity Asset Store si es necesario.

- Recursos disponibles (tiempo, equipo, materiales):

El proyecto se desarrollará de manera individual, contando con tiempo limitado según el calendario académico, un ordenador personal capaz de ejecutar Unity y realizar pruebas, y herramientas de desarrollo gratuitas o con licencia educativa, incluyendo recursos gráficos y sonoros libres de derechos.

- Dependencias o limitaciones técnicas:

El proyecto presenta las siguientes limitaciones: es mi primer videojuego, por lo que la experiencia en C# y Unity es inicial; se prioriza el funcionamiento sobre la calidad gráfica, limitando efectos o cantidad de enemigos; la compatibilidad inicial será solo para Windows; algunos recursos visuales y de sonido dependerán de fuentes externas gratuitas, y la historia se desarrollará de forma básica para asegurar un prototipo funcional.

- Posibilidad de trasladar el juego a otros dispositivos:

Aunque actualmente el desarrollo está enfocado en la versión para PC (Windows), no se descarta la posibilidad de adaptar el juego a otros dispositivos, como Android, en el futuro, si el tiempo y los recursos lo permiten.

---

3. Análisis de usuarios y roles

Rol	**Descripción**	**Permisos principales**
Jugador registrado	Usuario que crea una cuenta o inicia sesión para guardar su progreso, personalizaciones y acceder a funciones en la nube	- Acceso completo a todas las funcionalidades del juego - Subir y descargar archivos de guardado a la nube - Modificar configuraciones personales

Jugador invitado o no registrado	Usuario que juega sin crear una cuenta o iniciar sesión, puede probar el juego sin comprometerse a registrarse	- Acceso a la mayoría de las funciones del juego, excepto la sincronización en la nube y el guardado permanente más allá de la sesión actual
Administrador / Equipo de desarrollo	Usuarios con permisos para gestionar y mantener el sistema (normalmente el equipo técnico).	- Acceso total al backend y sistemas de gestión - Capacidad para modificar y actualizar el contenido y las bases de datos - No tienen acceso directo al juego como jugador

---

4. Casos de uso / Escenarios de uso

Código	**Nombre del caso de uso**	**Actor principal**	**Descripción**	**Resultado esperado**
CU1	Registrar usuario	Visitante	El usuario introduce sus datos y se crea una cuenta.	Usuario registrado correctamente.
CU2	Iniciar sesión	Usuario	Introduce credenciales y accede al sistema.	Acceso concedido.
CU3	Atacar automáticamente	Jugador	El personaje realiza ataques automáticos a los enemigos cercanos según sus armas o habilidades.	El personaje ataca a los enemigos cercanos solo
CU4	Subir de Nivel	Jugador	Al subir de nivel el personaje durante la partida, se abrira un menu en el que se permitirá la selección de dos mejoras u objetos	Se selecciona una mejora que potencia al personaje o añade un arma nueva.

CU5	Muerte del jugador	Jugador/Juego	Si la vida llega a 0, el juego termina y se muestra el resumen de progreso.	Fin de la partida y registro de puntuación.
CU6	Desbloquear fragmento de historia	Jugador/Sistema	Al subir de nivel o cumplir una condición (minuto alcanzado, enemigo derrotado, partida completada), el jugador desbloquea un fragmento narrativo.	Desbloqueo del fragmento de historia.

---

5. Modelo de datos o estructura de la información

BASE DE DATOS

En cuanto a la base de datos, se gestionarán diversas entidades clave relacionadas con el juego, estructuradas en varias tablas para facilitar su almacenamiento y manipulación:

Tabla PLAYER: Contendrá la información básica de los jugadores.

Campo	Tipo	Descripción
player_id (PK)	INT/UUID	Identificador único del jugador
player_name	VARCHAR	Nombre del jugador
level	INT	Nivel actual
survival_time	INT	Tiempo total de supervivencia
story_progress	INT	Progreso interno de historia

Tabla Rewards: Almacena las recompensas disponibles en el juego. Cada recompensa tiene un identificador único y un nombre descriptivo.

Campo	Tipo	Descripción
reward_id (PK)	INT	Identificador de recompensa
reward_name	VARCHAR	Nombre de la recompensa

Tabla PLAYER_REWARDS: Relaciona a los jugadores con las recompensas obtenidas.

Campo	Tipo	Descripción
player_id (FK)	INT/UUID	Referencia a Players
reward_id (FK)	INT	Referencia a Rewards
obtained_at	DATETIME	Fecha en que la obtuvo

Tabla ITEMS: Esta tabla almacena los objetos disponibles en el juego.

Campo	Tipo	Descripción
item_id (FK)	INT	Identificador de objeto
item_name	VARCHAR	Nombre del objeto

Tabla PLAYER_UNLOCKED_ITEMS: Relaciona a los jugadores con los objetos que han desbloqueado.

Campo	Tipo	Descripción
player_id (FK)	INT/UUID	Referencia a Players
item_id (FK)	INT	Referencia a Items
unlocked_at	DATETIME	Fecha de desbloqueo

*Cloud Save usa **JSON con estructura key/value**, así que el modelo debe ser **simple, plano y eficiente**.*

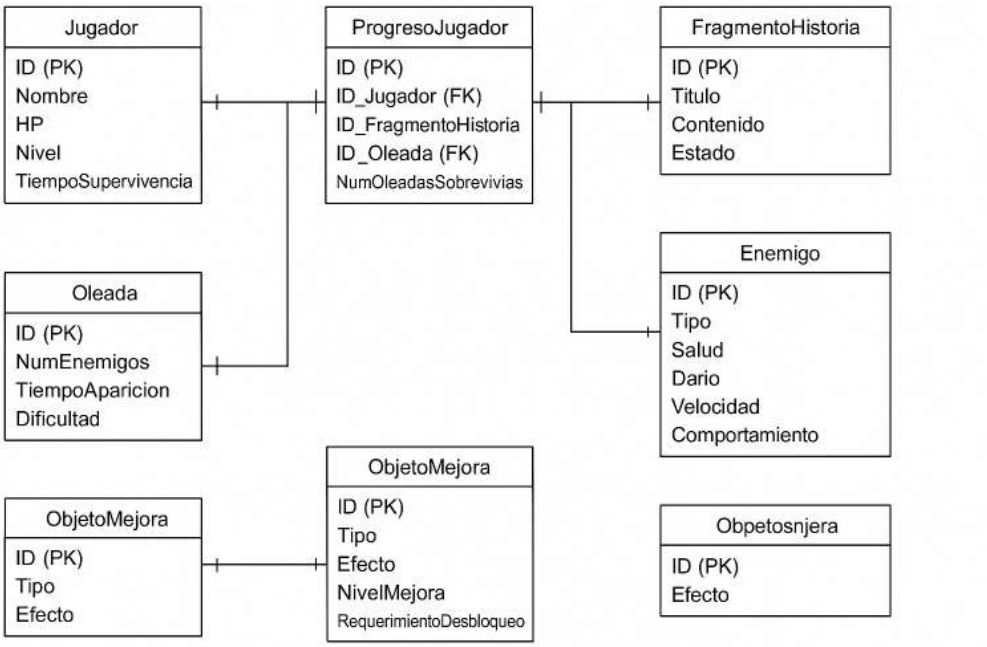
JUEGO

Respecto al juego, los datos que se manejarán a nivel de programación, serán:

Entidad	Descripción	Atributos
Jugador	Representa la jugador y su progreso en el juego	- Nombre del jugador - Puntos de vida (HP) - Nivel del jugador - Tiempo de supervivencia - Progreso en la historia (no visible) - Recompensas obtenidas - Objetos desbloqueados
Enemigo	Representa a los enemigos que atacarán al jugador	- Tipo - Salud - Daño

		- Velocidad - Comportamiento (ataque cc, ataque a distancia, curador…)
Oleada	Representa las oleadas de enemigos que debe sobrevivir el jugador	- Número de enemigos - Tiempo de duración - Tipo de enemigos - Dificultad - Intervalo entre oleadas
Historia	Fragmentos narrativos que se desbloquean jugando	- Contenido del fragmento - Método de obtención (Nivel jugador, nº oleada, acción específica…) - Estado (desbloqueado o no)
Objeto/Mejora	Elementos que el jugador obtiene o mejora	- Tipo (arma, habilidad, consumible…) - Efecto - Nivel de mejora - Duración (si es temporal)

Relación	**Descripción**
Jugador → Progreso del Jugador	Un jugador puede tener múltiples progresos (una por cada sesión o partida)
Progreso del Jugador → Fragmento de Historia	Un progreso puede desbloquear múltiples fragmentos narrativos
Jugador ↔ Objeto / Mejora	Un jugador puede desbloquear múltiples objetos; un objeto puede ser desbloqueado por varios jugadores
Oleada → Enemigo	Una oleada puede contener múltiples enemigos
Enemigo ↔ Oleada	Un enemigo puede aparecer en distintas oleadas
Fragmento de Historia → Oleada / Progreso del Jugador	Un fragmento de historia puede estar vinculado al progreso (por tiempo sobrevivido o número de oleadas completadas)

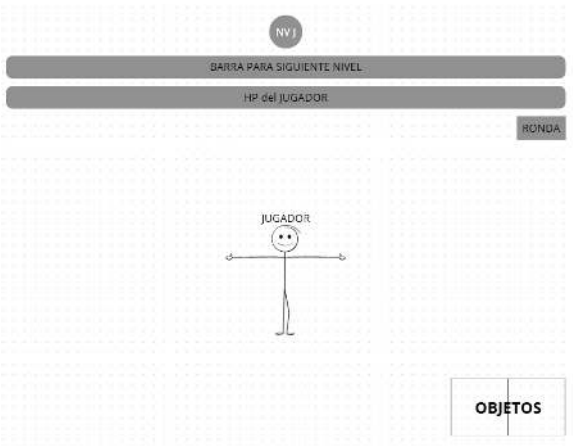
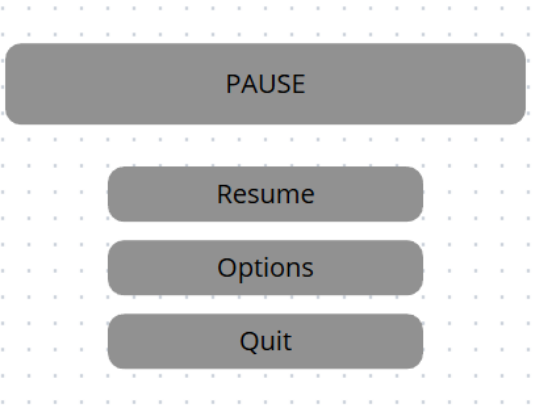


DATOS DEL JUEGO

---

6. Diseño de la interfaz

Nombre	**Imagen**
Menú de inicio: Será lo primero que verá el Jugador al iniciar el juego, consta del título del juego (aún sin nombre) y de las opciones de empezar partida nueva, seguir con la que ya empezaste, opciones y salir del juego. Además he pensado en añadir un botón que sirva como link a la web para que los usuarios que jueguen al juego puedan dar feedback con comentarios.	

Layout in game: Aquí encontraremos lo que veremos en la partida mientras jugamos. Contará con dos barras, una que indicará la salud del jugador y la otra cuánto le falta para alcanzar el siguiente nivel. Encima de esta última saldrá el nivel actual del jugador En la esquina inferior tendremos los objetos consumibles que tendremos disponibles, si no lo tenemos saldrán en blanco.	
Menú de pausa: Este menú aparecerá cuando se le dé al botón de pausa. Al pulsar aparecerán diferentes cuadros de texto que serán: - Para continuar la partida - Acceder a opciones - Salir al menú de inicio	
Pantalla de selección de objeto: Esta pantalla saldrá al subir de nivel y aparecerán dos objetos o mejoras con una imagen y debajo un texto explicativo.	
Diseño de la web: Se contará con una web sencilla con una explicación del juego, un slider con fotos del mismo y su descarga.	

7. Planificación técnica

Indica las tecnologías y herramientas que se utilizarán, y cómo se organizará el trabajo.

- ## Lenguajes y frameworks:

Motor de desarrollo: *Unity (versión LTS)*.

Se ha elegido Unity para desarrollar este videojuego 3D debido a que ofrece la mayoría de características necesarias para el proyecto. Unity permite trabajar con entornos tridimensionales, gestionar físicas, animaciones, efectos visuales y crear una interfaz de usuario completa.

Además, ofrece integración con servicios en la nube y autenticación de usuarios, lo que amplía las posibilidades del proyecto.

Lenguaje de programación: *C#*.

Se emplea para programar la lógica del juego, la interacción entre objetos, el control de personajes, las colisiones, la gestión de puntuaciones y el guardado de datos.

Servicios y frameworks adicionales:

- **Unity Cloud Save:** se utilizará para permitir que el jugador pueda subir y guardar su progreso en la nube, utilizando los servidores de Unity.
- **Unity Authentication:** permitirá que el jugador inicie sesión, de modo que el sistema reconozca su usuario y pueda asociar sus datos de guardado.
- **Cinemachine:** control avanzado de cámaras en entornos 3D.
- **TextMeshPro:** renderizado de texto de alta calidad en la interfaz.
- **Post Processing Stack:** mejora visual mediante efectos como bloom, color grading o profundidad de campo.

- ## Base de datos:

Unity Cloud Save: permitirá guardar en la nube los datos de progreso y configuración del jugador.

PlayerPrefs: para almacenar de forma local ajustes simples (volumen, resolución, última sesión).

- ## Herramientas de diseño o edición:

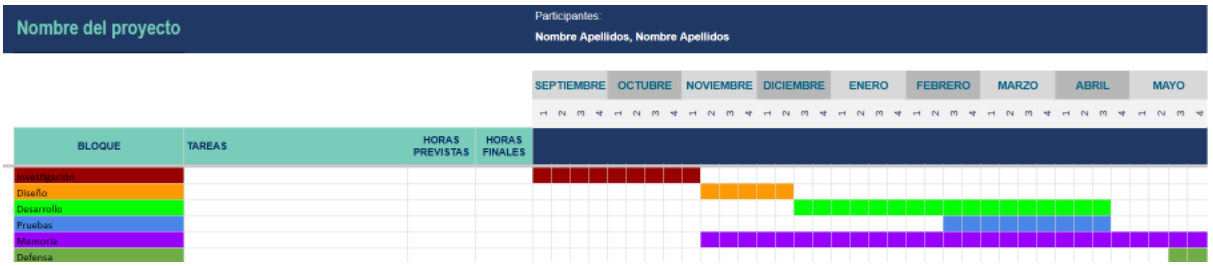
GitHub: control de versiones y respaldo del proyecto.

Trello: organización y seguimiento del progreso del desarrollo.

Blender: modelado 3D de objetos, escenarios y personajes.

Unity Editor: diseño de niveles, configuración de escenas, iluminación, materiales, animaciones y pruebas de jugabilidad.

- Cronograma (puede incluir un diagrama de Gantt):



---

8. Análisis de riesgos

8.1. Identificación de riesgos

- Falta de tiempo
- Mala planificación
- Problemas de rendimiento
- Pérdida o corrupción de datos
- Bugs y errores críticos

8.2. Valoración y respuesta

Clasifica cada riesgo según su probabilidad e impacto, e indica cómo se mitigará.

Riesgo	**Probabilidad**	**Impacto**	**Plan de prevención o contingencia**
Falta de tiempo	Media	Alta	Dividir tareas y fijar entregas intermedias.
Mala planificación	Alta	Media	Planificar bien qué es lo que quiero y como lo quiero con antelidad
Problemas de rendimiento	Baja	Medio	Asegurarse de que cada implementación no haga que el programa baje de rendimiento
Pérdida o corrupción de datos	Baja	Alta	Ir haciendo copias de seguridad cada semana y asegurarse de que GitHub sube bien todos los datos

Bugs y errores críticos	Media	Alta	Implementar *testing* constante (QA interno o externo). Realizar *playtests* frecuentes con usuarios reales. Utilizar Unity LTS

---

9. Validación y criterios de éxito

- Criterios de aceptación (qué debe cumplirse para darlo por válido).
    - Funcionamiento básico completo: El juego inicia correctamente desde el menú principal y puede completarse sin errores críticos (sin bloqueos ni cierres inesperados).
    - Jugabilidad estable: Las mecánicas principales (movimiento, colisiones, IA, recolección de objetos, puntuación, etc.) funcionan como se diseñaron.
    - Interfaz funcional y clara: Los menús, botones e indicadores son visibles, legibles y responden correctamente.
    - Rendimiento aceptable: El juego mantiene una tasa estable de 60fps.
    - Experiencia de usuario positiva: Los jugadores comprenden fácilmente los controles y objetivos del juego.

- Pruebas previstas (funcionales, de usuario, de rendimiento).

Tipo de prueba	**Descripción**	**Responsable**	**Resultado esperado**
Pruebas funcionales	Verificar que todas las mecánicas (movimiento, disparo, colisiones, puntuación) funcionan correctamente.	Desarrollador	Todas las funciones operan sin errores.
Prueba de usuario (jugabilidad)	Pedir a 2-3 personas que jueguen y den feedback sobre controles, dificultad y claridad de objetivos	Desarrollador/testers	Jugabilidad fluida y comprensión rápida del objetivo
Pruebas de rendimiento	Medir FPS, tiempos de carga y consumo de memoria	Desarrollador	Rendimiento estable y sin caídas graves de FPS
Pruebas de compatibilidad	Probar diferentes resoluciones o dispositivos	Desarrollador	El juego corre sin errores en todas las configuraciones

			probadas.

- Indicadores de calidad o resultados esperados.

Tasa de errores críticos: 0 (ningún bug que impida jugar).
Estabilidad: el juego se puede jugar durante al menos 10 minutos sin bloqueos.
Rendimiento: FPS medio ≥ 30 fps.
Satisfacción de usuario: al menos el 80 % de los jugadores de prueba considera el juego “divertido” o “fácil de entender”.
Entrega puntual: proyecto final entregado dentro del plazo establecido.

---

10. Conclusión

El juego será uno estilo *Vampire Survivor* en el que el jugador irá completando hordas de enemigos y completando niveles para avanzar en la historia.

Se desarrollará completamente en Unity con ayuda de GitHub como control de versiones o Blender para modelados de personajes u objetos.

Aunque este proyecto no busca generar un impacto social o económico directo, su valor radica en el crecimiento personal y creativo que representa. Desarrollar mi propio videojuego me permite unir mi pasión por los videojuegos con el aprendizaje práctico del proceso de creación, programación y diseño, convirtiéndose en una experiencia significativa y motivadora a nivel personal.

PRÓXIMOS PASOS:

- Preparar el entorno de desarrollo con los assets y todo listo.
- Crear la estructura inicial del repositorio y la base de datos.
- Empezar la implementación de los casos de uso prioritarios.

El análisis funcional proporciona una visión clara, estructurada y realista del videojuego, permitiendo definir con precisión sus mecánicas, características y objetivos principales. A partir de esta base, se establecen los cimientos técnicos y conceptuales necesarios para iniciar de manera organizada la fase de desarrollo en Unity.