**Listado de Tecnologías Empleadas**

1. **Unity**

* Versión: 2023.3.0a15
* Descripción: Motor de desarrollo de videojuegos utilizado para la creación del videojuego. Elegido por su robustez, soporte multiplataforma, y amplia comunidad de desarrolladores.

1. **C#**

* Descripción: Lenguaje de programación utilizado para el desarrollo de scripts dentro del motor Unity. Elegido por su integración nativa con Unity y su facilidad para gestionar la lógica del juego.

1. **Rigidbody2D (UnityEngine)**

* Descripción: Componente físico utilizado para añadir comportamiento físico a los objetos 2D en Unity. Elegido para gestionar movimientos y colisiones del personaje de forma realista.

1. **Animator (UnityEngine)**

* Descripción: Componente de Unity utilizado para controlar y gestionar las animaciones del personaje. Permite sincronizar movimientos y acciones con las animaciones correspondientes.

1. **Input.GetAxis (UnityEngine)**

* Descripción: Función de Unity utilizada para obtener la entrada del usuario en los ejes horizontal y vertical, facilitando el control del movimiento del personaje.

1. **Instantiate (UnityEngine)**

* Descripción: Método de Unity para crear instancias de objetos prefabricados en tiempo de ejecución. Utilizado para generar armas y proyectiles durante el juego.

1. **AudioSource (UnityEngine)**

* Descripción: Componente de Unity utilizado para reproducir sonidos en el juego. Permite gestionar efectos de sonido como música de fondo y efectos sonoros específicos de acciones.

1. **AudioClip (UnityEngine)**

* Descripción: Clase de Unity utilizada para almacenar archivos de audio que se pueden reproducir mediante AudioSource. Elegido por su integración nativa en Unity y su facilidad de uso.

1. **SceneManager (UnityEngine.SceneManagement)**

* Descripción: Clase de Unity utilizada para la gestión de escenas en el juego. Permite cargar y cambiar entre diferentes escenas de manera eficiente. Elegido por su integración en el flujo de trabajo de Unity y su capacidad para manejar transiciones de escenas.

1. **IEnumerator (System.Collections)**

* Descripción: Interfaz utilizada en C# para crear funciones de corrutinas, permitiendo ejecutar métodos en pasos múltiples sin bloquear el hilo principal. Utilizado para manejar transiciones suaves y retardos en el juego.

1. **Application (UnityEngine)**

* Descripción: Clase de Unity utilizada para gestionar aspectos de la aplicación, como cerrar el juego. Elegido por su capacidad para manejar operaciones de alto nivel en la aplicación.

1. **OnTriggerEnter2D (UnityEngine)**

* Descripción: Método de Unity utilizado para detectar colisiones con otros objetos 2D con colisionadores. Elegido para implementar la lógica de daño y colisiones con enemigos y otros objetos en el juego.

1. **TextMeshProUGUI (TMPro)**

* Descripción: Componente de TextMesh Pro utilizado para mostrar y gestionar texto en la interfaz de usuario del juego. Elegido por su capacidad para renderizar texto de alta calidad y su integración con Unity.

1. **Random (UnityEngine)**

* Descripción: Clase utilizada para generar números aleatorios dentro de un rango específico. Utilizado para implementar comportamientos aleatorios en el juego, como la generación de enemigos.

1. **Mathf (UnityEngine)**

* Descripción: Clase estática que proporciona métodos matemáticos comunes, como Atan2 y Rad2Deg, utilizados para calcular ángulos y realizar conversiones matemáticas necesarias para el movimiento y la física en el juego.

1. **Invoke (UnityEngine)**

* Descripción: Método de Unity utilizado para llamar a una función después de un retraso específico. Elegido por su simplicidad y eficacia para manejar acciones temporizadas en el juego.

1. **Unity.VisualScripting**

* Descripción: Framework visual de scripting en Unity que permite a los desarrolladores crear lógica de juego sin necesidad de escribir código, mediante nodos visuales. Elegido para facilitar el desarrollo rápido y la iteración de prototipos.

1. **SpriteRenderer (UnityEngine)**

* Descripción: Componente de Unity utilizado para representar sprites (imágenes 2D) en la pantalla. Utilizado para cambiar las imágenes de las armas y personajes durante el juego, permitiendo visuales dinámicos y responsivos a las acciones del jugador.

1. **Transform (UnityEngine)**

* Descripción: Componente de Unity utilizado para gestionar la posición, rotación y escala de los objetos en el espacio 3D. Fundamental para la manipulación y control de objetos en el juego.

1. **ForceMode2D (UnityEngine)**

* Descripción: Enum utilizado para definir el tipo de fuerza aplicada a los objetos 2D en Unity. Utilizado en la física del juego para determinar cómo se aplican las fuerzas a los objetos dinámicos como las flechas disparadas.

1. **WaitForSeconds (UnityEngine)**
   * Descripción: Clase utilizada para crear un retraso temporal en una corrutina. Elegido para manejar los tiempos de espera entre las acciones de ataque en el juego.
2. **Quaternion (UnityEngine)**
   * Descripción: Estructura utilizada para manejar la rotación de objetos en Unity. Utilizado en el script para rotar los objetos de manera precisa al instanciarlos.
3. **Color (UnityEngine)**
   * Descripción: Estructura utilizada para definir colores en Unity. Utilizada en el script para cambiar los colores del enemigo basados en su estado de salud.
4. **SpriteRenderer (UnityEngine)**
   * Descripción: Componente de Unity utilizado para representar sprites (imágenes 2D) en la pantalla. Utilizado para cambiar las imágenes de las armas y personajes durante el juego, permitiendo visuales dinámicos y responsivos a las acciones del jugador.