PokeSharp

Autor: Marcos Zahonero
Ciclo formativo: 2º DAM

Centro: IES Dr. Faustí Barberá
Tutor individual: Belén López Pérez

Índice

1. Introducción

PokeSharp es un fan game de Pokémon basado en texturas del juego Pokémon Blanco y Negro 2. A diferencia de otros juegos fan games, este se diferencia sobre todo por sus tecnologías utilizadas, ya que la mayoría de juegos usan RPG Maker XP, el cual permite la facilidad de incorporar texturas y sistemas ya hechos, en este caso se partió desde cero, con el objetivo de crear una historia original y entretenida para el jugador.

Este proyecto incluye también una página web que sirve para su promoción, facilitando el seguimiento de las actualizaciones, el lanzamiento de tráilers y otros contenidos relacionados importantes del proyecto.

2. Justificación

El motivo de desarrollar este proyecto es principalmente personal, más que el realizar un proyecto complejo. Desde hace mucho sentía la necesidad de crear un juego, especialmente de Pokémon, que incluyera una historia única, con el objetivo de capturar el interés del jugador, haciendo que te sea agradable cada segundo que lo juegues.

3. Objetivos del proyecto

- Desarrollar un juego que mantenga la esencia de Pokémon, pero que al mismo tiempo contenga elementos diferenciadores.
- Implementar sistemas complejos (Combate, inventario, diálogos…) para poner a prueba mis conocimientos en programación.
- Explorar una nueva área, los videojuegos, y nuevas tecnologías, para así explorar más el mundo de la programación.
- Crear un proyecto que pueda ser continuado en un futuro y que pueda llegar a tener éxito a largo plazo.
- Realizar un proyecto con el que me sienta cómodo haciéndolo y satisfecho de lo conseguido.

4. Planteamiento inicial

Teniendo claros los objetivos ahora falta hacer una investigación profunda para decidir el entorno y las tecnologías utilizadas para el proyecto, normalmente alrededor de un 95% de los fan games de Pokémon utilizan RPG Maker XP con una librería llamada Pokémon Essentials para hacer todo, veamos un ejemplo:

Esta imagen corresponde a un fangame hecho en RPG Maker XP utilizando Pokémon Essentials, un paquete que replica completamente la experiencia de Pokémon. Este motor y esta librería facilitan la implementación de sistemas complejos, como el combate y los encuentros con Pokémon, permitiendo que el “desarrollador” se concentre principalmente en el diseño del juego.

Como mencioné, la mayoría de los fangames utilizan RPG Maker XP junto con Pokémon Essentials. Sin embargo, los juegos más destacados modifican esta librería mediante Ruby para sacar el máximo provecho y marcar la diferencia.

El 5 % restante de los juegos que no están hechos en RPG Maker XP suelen estar desarrollados en páginas web usando JavaScript, como es el caso de grandes títulos como Pokémon Showdown y Pokémon Eclipse RPG. Otros optaron por Unity para crear juegos en 3D, y uno de los más conocidos es PokéMMO, que desarrolló su propio motor de videojuegos utilizando Java para el cliente y otros lenguajes para el servidor. Este juego usa directamente la información oficial de los juegos en cuanto a texturas, pero es el más distinto en cuanto a desarrollo.

Teniendo toda esta información, vamos a recopilarla y presentar las ventajas y desventajas de las opciones más viables y accesibles para un proyecto con una duración aproximada de tres meses, cuyo objetivo sea desarrollar una versión funcional con lo fundamental y con posibilidades de crecimiento futuro.

Teniendo toda esta información vamos a recopilar todo y hacer unas ventajas y desventajas de las opciones más viables y accesibles para un proyecto de duración 3 meses con el objetivo de hacer una versión con lo fundamental puesto y con opciones a que tenga un futuro.

Plataforma / Motor	Lenguaje principal	Ventajas	Desventajas
RPG Maker XP + Essentials	Ruby	- Facilidad de hacer, hay muchas cosas hechas - Mucha comunidad	- Limitado para hacer algo grande - Poco margen para demostrar habilidades de programación
Página web	JavaScript	- Multiplataforma	- Poco conocimiento en entorno web - Complicado de hacer todo
Unity	C#	- Entorno conocido y estable - Graficos 2D/3D - Multiplataforma - Sin límites	- Complicado de hacer un juego profesional sin conocer bien la herramienta - Programar desde cero
Godot	GDScript / C#	- Entorno menos usado que Unity pero en constante crecimiento con mucha comunidad - Entorno 2D más amigable - Facilidad de aprender a usar el entorno - Sin limites - Sería de los primeros proyectos casi en Godot de Pokémon	- Pocos proyectos de Pokémon para poder basarse en ellos y ver posibles dificultades - Programar todo desde cero

Teniendo esta tabla, podemos descartar las opciones de RPG Maker y de página web. Nos quedamos con dos motores bastante conocidos: Unity y Godot. Lo bueno de Unity es que lleva bastante tiempo en el mercado, tiene una gran comunidad y, realmente, es el más usado en este ámbito. Sin embargo, puede ser un poco caótico aprenderlo si quieres hacer algo más profesional, como era mi objetivo. El haberlo probado en clase me hizo darme cuenta de que su interfaz no era fácil de entender.

En cambio, al probar Godot y ver un par de vídeos para entender el funcionamiento de árboles y nodos, pude comenzar con el proyecto mientras investigaba las cosas que necesitaba en cada momento.

Mi opinión, tras llevar un tiempo trabajando en el proyecto, es que Godot 2D ha superado a Unity por bastante. Me informé mediante videos de desarrolladores indie de videojuegos y realmente decían lo mismo: Godot es un entorno de código abierto desarrollado por personas que seguramente hayan probado Unity. Realmente, el usuario de un producto es quien sabe qué es lo que le falta, y eso es precisamente lo bueno de Godot, que han sabido identificar lo que le falta a Unity.

Finalmente, me decidí por Godot, ya que lo veo como un motor con mucho futuro, fácil de aprender, ligero y muy sencillo de usar.

5. Tecnologías utilizadas

5.1. Juego

C# = Lenguaje de programación principal.
Godot Engine = Motor de desarrollo del videojuego.
Supabase = Plataforma utilizada para conectividad y almacenamiento de bases de datos.

Aplicaciones utilizadas

FontForge = Edición de las tipografías utilizadas, para añadir caracteres como “¡” “¿”.
GIMP = Edición de los sprites de entrenadores
Paint = Edición de los tilesets utilizados para arreglar detalles o facilitar en si su uso.

5.2. Web

HTML = Estructura base de la página web.
CSS = Diseño y estilo de la web.
JavaScript = Funcionalidades dinámicas, como por ejemplo traducción automática de idiomas y otras interacciones.

5.3. Otros

Adobe Premiere Pro 2022 = Programa para editar el trailer del juego.
Draw.io = Página web utilizada para diseñar el diagrama de casos de uso.

6. Temporalización inicial

Para los tiempos que debo seguir hice una hoja de cálculo para llevar una organización y porque también van a ser subidos a un canal de YouTube unos blogs de desarrollo, que son vídeos durante el desarrollo de un videojuego enseñando mecánicas y en sí el desarrollo del juego.

Fecha límite	26/02/25	23/03/25	06/04/25	20/04/25	04/05/25
Nº dev blog	1#	2#	3#	4#	5#
Nombre dev blog	**Movement, hitboxes y entorno**	**NPCs in game, Dialogos y sonidos**	**Equipo, inventario y captura de pokemon**	**Combates e Items**	**Inicio, ajustes y sistema de guardado**
Contenido en el video	Movement Grid System	NPCs in game	Encuentro de pokemon	Combate vs. NPCs	Pantalla inicio
	Y Sorting	Mapa spawn hecho	Ventana equipo	Cambio de mapa	Crear/Cargar partida
	Poder entrar en sitios	Sistema Dialogos	Ventana inventario	Recoger items	Crear personaje
	Ventana combate provisional	Hierba alta y sonidos		Menu	Ajustes
	Diseño provisional spawn				Sistema de guardado automatico

Antes de hacer eso también realicé la página web, que fue durante el curso hasta básicamente terminar el curso (Diciembre 2024 – Febrero 2025).

Para el mes de abril quisiera tener una posibilidad de otorgar una beta privada para si alguien le apetece probarlo y encontrar algunos fallos y también así ver un poco la evolución y opinión.

7. Temporalización final

La temporalización final que realicé fue bastante distinta a la temporalización inicial, esto se debe a la falta de experiencia con el entorno utilizado y también a la poca consideración de trabajo que llevaría hacer ciertas tareas.

Al final la idea de hacer los blogs de desarrollo fue cancelada por falta de tiempo y también porque se consideró que requería demasiado esfuerzo para el beneficio que aportaba realmente al proyecto.

Esto fue lo que se realizó en el plazo de los vídeos, y también lo que se mostró en ellos.

Fecha límite	06/04/25	13/04/25	20/04/25
Nº vídeo	1#	2#	3#
Contenido en el video	Pantalla inicio	Cambios en la estructura (Clases, carpetas, etc..)	Funcionalidad de ocultar opciones de combate
	Sistema movimiento del jugador	Navegación entre escenas	Mochila en combate
	Sistema de interiores para uso de puertas su sonido	Menú de pausa	Nuevo mapa
	Sistema de encuentros pokemon sin acabar	Estados de juego (Menú, Pausa, jugando, en combate)	Menú principal (Tu equipo, tus pokemon, mochila y pokedex)
	Vinculación de base de datos	Asignación de nivel a las hierbas para encuentros	

Después de estos vídeos plantee la siguiente temporalización final con lo que veía esencial, con fechas límite o fechas para empezar a realizar esa tarea.

- Pantalla de equipo → Día 04/05/25
- Inicio documentación/memoria → Día 09/05/25
- Música → Día 10/05/25
- NPCs → Día 15/05/25
- Lanzar versión alfa → Día 11/05/25
- Acabar documento/memoria hasta lo necesario para la entrega → Día 15/05/25
- Empezar Powerpoint → Día 16/05/25
- Sistema de encuentros pokémon → Día 17/05/25
- Pantalla de inventario → Día 18/05/25
- Sistema de combate contra NPCs → Día 24/05/25
- Empezar trailer → Día 24/05/25
- Terminar documentación/memoria → Día 01/05/25
- Terminar Powerpoint → Día 02/05/25
- Terminar trailer → Día 04/05/25

8. Descripción del proyecto

El proyecto se divide en sistemas que lo hacen un fan game de Pokémon, como por ejemplo los encuentros pokémon o los combates contra entrenadores, aunque ambos sistemas comparten un sistema de combate.

8.1. Combate

El combate en Pokémon aparece en ambos sistemas más importantes: los encuentros Pokémon y los combates contra entrenadores. Al fin y al cabo, estás combatiendo contra un Pokémon en ambos casos. Veamos cuál es su funcionamiento, dejando de lado la parte gráfica, que será gestionada por otros apartados. Aquí se hablará principalmente del código del combate.

Estados del combate

Cuando comienza el combate, hay que gestionar los estados. ¿Qué estados existen?

TurnoEstado.Jugador → Indica que el turno es del jugador.
TurnoEstado.Enemigo → Indica que el turno es del enemigo.
TurnoEstado.Atacando → Indica que actualmente se está atacando (enemigo o jugador).
TurnoEstado.AnimacionEnCurso → Indica que se está reproduciendo una animación.
TurnoEstado.EsperandoAccion → Indica que se está lanzando una Poké Ball.
TurnoEstado.EsperandoChange → Indica que se está esperando a que el jugador seleccione un Pokémon, ya que el actual fue derrotado.
TurnoEstado.CombateTerminado → Indica que el combate ha finalizado.

Estos estados facilitan la gestión del combate y permiten seguir un flujo ordenado.

¿Quién comienza el combate?

Volviendo al inicio del combate, y entendiendo estos estados, hay que saber quién empieza su turno y por qué.

El primer turno se decide por la velocidad del Pokémon. Los Pokémon tienen una serie de estadísticas: Ataque, Defensa, Ataque Especial, Defensa Especial y Velocidad.

Si la velocidad del Pokémon del jugador es **mayor** que la del enemigo → El jugador comienza.
Si es **igual o menor** → Comienza el enemigo.

Estos estados también determinan qué puedes hacer y cuándo puedes hacerlo.
Por ejemplo, solo podrás seleccionar un ataque cuando turnoActual sea igual a TurnoEstado.Jugador.

Gestión de turnos

Cada vez que se gestiona un turno, es decir, cuando turnoActual == TurnoEstado.Jugador y el jugador ya ha atacado, se cambia a TurnoEstado.Enemigo y se llama a un método llamado IniciarTurnoAsync para gestionar el turno actual.

En este caso, como ataca el enemigo, se debe llamar a su método de ataque.

Si el enemigo derrota al Pokémon que tienes en el campo y tienes más Pokémon vivos, el juego te obliga a sacar otro Pokémon, asignando turnoActual = TurnoEstado.EsperandoChange, mostrando la pantalla de equipo y evitando que se pueda hacer cualquier otra acción, podrás cambiar de menú pero no te dejará capturar el pokémon o huir.

Fin del combate

En caso del TurnoEstado.CombateTerminado actúa de forma distinta dependiendo la situación:

En un **encuentro** Pokémon, hay dos posibles escenarios:

Si se captura el Pokémon:
- Se guardará en el equipo si hay espacio disponible.
- Si el equipo está completo, se enviará al PC.
- En cualquier caso, se inserta en la base de datos manteniendo todos los datos.
- No se otorga experiencia en este caso.

Si se derrota al Pokémon:
- Se otorgará experiencia a todos los Pokémon del equipo.

En un **combate** contra un entrenador:
Al finalizar el combate, se ocultan los Pokémon y se muestra al entrenador diciendo una frase, la cual varía dependiendo de si ha ganado o perdido.

Después de eso en ambas situaciones se cambia de pantalla mostrando el mapa de nuevo y el jugador se puede mover con normalidad.

8.2. Encuentro pokémon

¿Cómo llegamos a un encuentro pokémon?

Primero debemos dirigirnos a buscar unas hierbas y pasar por encima, como vemos en la imagen:

Cuando estemos pasando por encima moviéndonos entre ellas, calcula internamente un número del 1 al 100. Si este número está entre 1 al 25, saltará el encuentro pokémon si sale un número que no se encuentre entre el 1 y el 25 tocará seguir buscando pasando entre más hierbas.

El procedimiento al iniciar el encuentro pokémon es el siguiente:

1. Salta la animación haciendo zoom, la música empieza y aparece la común animación gráfica de Pokémon en la pantalla.
2. Busca un aleatorio del 1 al 649 y consulta a la base de datos, en la tabla de pokemons, ese poke_id en busca de un pokémon.
3. Saca sus datos convirtiendo ese JSON a un objeto y empieza a asignar la parte gráfica, asignando el nivel del pokémon, nombre, etc.
4. Hace una búsqueda de sus movimientos. En este juego es distinto al juego original, ya que aquí existe una fórmula distinta con porcentajes, y es la siguiente:

> 45 % -> Movimiento que puede aprender el Pokémon por tutor
> 35 % -> Del mismo tipo del Pokémon
> 20 % -> Movimientos aleatorios sin requisitos

Esto contiene unos porcentajes para decidir qué movimientos sacar. Esto lo decide con cada movimiento, es decir, un total de 4 veces.

5. Salta la pantalla de combate y aparece con una animación de negro a su color original.
6. Empieza realmente el combate y se empiezan a gestionar los estados, calculando el primer turno dependiendo la velocidad de ambos pokémon. Si uno es más veloz, empieza.

Aquí un ejemplo de un encuentro pokémon cuando ya ha cargado todo. En este caso, el juego consultó las dos velocidades y vio que en este caso el pokémon “Darkrai” era más veloz que

Dialga. Es por eso que atacó primero, como se puede ver abajo a la izquierda, que es una caja que contiene el registro de combate, para también ver qué ataques usa el enemigo. Como se puede ver, al pokémon enemigo realizar ese ataque a mi pokémon, le ha bajado la vida en consecuencia.

Ahora que el enemigo ha atacado, se le acaba su turno y nos toca a nosotros. Podemos realizar cualquier acción: usar un movimiento, cambiar de pokémon o usar una Pokéball para atraparlo. Probemos primero con un movimiento:

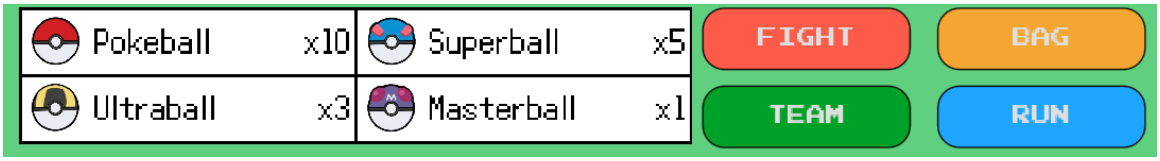
Aquí vemos los movimientos. Su fondo depende del tipo que son: tres son de tipo dragón, por eso tienen ese color y, el Meteor mash es tipo acero.

Lo usamos y le bajamos la vida hasta bastante. Después cambia el turno y consigue atacarnos. Vamos a aprovechar que se encuentra a poca vida para poder ver la captura de pokémon.

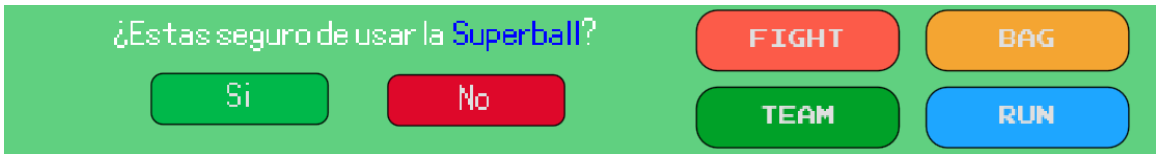
Cuando le damos a “Bag”, podemos usar objetos de captura.

Vamos a usar una Superball, este pokémon que tenemos en contra es un pokémon legendario, es decir, tiene la probabilidad más baja de todas. Cada Pokeball que vemos aquí tiene un multiplicador propio:

- Pokeball x1
- Superball x1.5
- Ultraball x2
- Masterball captura asegurada.



Una confirmación, le damos a “Sí”.



No ha habido suerte y nos atacó de vuelta al acabar nuestro turno, ya que lo hemos utilizado para lanzar la Superball, vamos a probar con una Ultraball.

Tampoco, las probabilidades fueron altas, pero no fue suficiente. Al atacarnos, nos dejó sin vida restante, es decir, nos debilitó al Pokémon, ahora nos fuerza a cambiar de pokémon mostrándonos lo siguiente:

Una vez seleccionemos uno, perdemos el turno y le toca atacar. Vamos a dar por finalizado este encuentro con una Masterball.

Sale que lo hemos capturado y finaliza el encuentro. Este pokémon se colocará dependiendo de nuestro equipo, si nuestro equipo actual se encuentra lleno se mandará al PC, sino se añadirá al equipo actual.

8.3. Combate contra entrenadores

El combate contra entrenadores es similar a los encuentros Pokémon. Las diferencias más destacables son que los entrenadores tienen modificadores y pueden tener más de un Pokémon, llegando incluso a tener hasta seis.

En el apartado de Entrenadores se detallan las propiedades que contiene un entrenador. Aquí se aborda lo relacionado específicamente con el combate.

Los combates contra entrenadores siempre son distintos: los Pokémon reciben movimientos aleatorios en cada partida, lo que permite sorprenderte cada vez que te enfrentas a ellos. Estos Pokémon no se rigen por la fórmula original de aleatoriedad para la generación de movimientos como los del jugador. Esto se explica con más detalle en el apartado de Entrenadores.

Al inicio del combate se muestra una pequeña animación en la que ambos entrenadores (tú y el entrenador rival) se acercan, el entrenador dice una frase, y luego se reproduce una animación en la que retrocede. A continuación, se muestran los Pokémon mediante otra animación.

Por supuesto, al tratarse de un combate contra un entrenador, no puedes capturar sus Pokémon. Aunque uses una pokeball, no sucederá nada.

Durante el combate, si derrotas a uno de sus pokémon, esto se registra en el historial de combate en la parte inferior izquierda. Después de eso, el entrenador enviará a otro pokémon. En este caso, se trata de su último pokémon disponible. Él nos supera en velocidad y recibimos un gran impacto.

Finalmente, logramos derrotar a Arceus. Entonces se muestra nuevamente la animación de ambos entrenadores, y el entrenador rival dice su frase de derrota.

8.4. Sistema de puertas

Esta parte es de alta importancia a la larga en el proyecto, ya que este sistema permite el traslado del personaje entre distintas escenas/mapas sin hacer al usuario pensar que están cambiando muchas cosas de forma interna.

Al entrar a un interior, por ejemplo, al laboratorio, debemos crear en un json un nuevo registro con la siguiente estructura:

```
  "Nombre": { "label": "Laboratorio", "interior": "false", "scene": "Escena
que carga", "xSpawnPoint": "-84", "ySpawnPoint": "191,6667" }
```

Esta permite que cuando nos acerquemos a la puerta en este caso desplegué un texto encima de la puerta para hacernos saber donde vamos a entrar, llegados a este punto si le damos a la tecla “E” nos llevará dentro, haciendo sonar un sonido de una puerta, mostrando una transición y mientras la transición se realiza hace el cambio de escena sin que se vea visualmente.

Cuando ponemos si es un interior o no es para poder hacer ciertas cosas como lo siguiente, si nosotros entramos de espaldas al laboratorio ocurre algo curioso, y es que da igual como nos coloquemos que nos orienta hacia arriba.

Esto también ocurre al revés.

8.5. Pantalla menú principal

Esta pantalla se puede abrir mediante la tecla “R”. A la parte izquierda tenemos la sección de **Equipo**, donde se muestra de forma visual los Pokémon que tienes en tu equipo. ¿Cómo identifica el juego qué pokémon tienes en tu equipo o no? ¿Y su orden? Esto es gracias a una variable que contiene los pokémon que capturas: inTeam. Es un entero que funciona como orden, y para identificar si está en tu equipo o si se encuentra en el PC es sencillo: si el número es -1, este lo enviará al PC. En el apartado de inicio de sesión se entrará más a profundidad sobre esto.

Por otro lado, en la parte de la derecha tenemos un menú con 2 cosas a futuro.

1. Tus Pokémon, esta pantalla nos abre la pantalla del PC, para poder gestionar los pokémon que tienes fuera del equipo y poder intercambiarlos para ponerlos en tu equipo.

2. Bolsa, es un apartado que contiene los objetos que posees. Esta opción es a futuro, es por eso que actualmente existe pero no tiene funcionalidad, es algo a futuro.

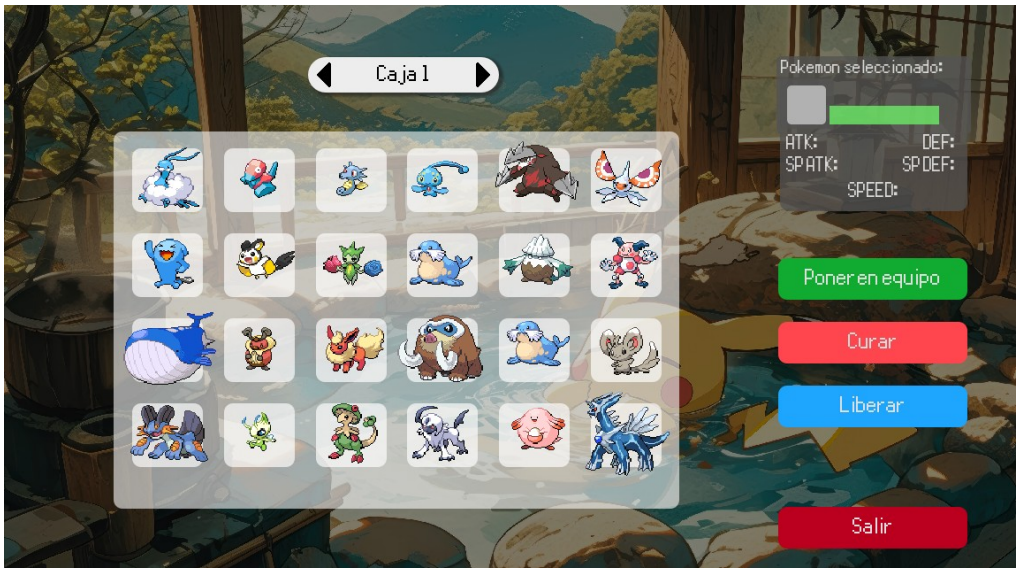
3. Pokedex, esto es algo común en todos los juegos Pokémon, pero es algo que cuesta bastante realizar y también es algo para la gente que de verdad quiera invertir más de 40 horas al juego. Ya que consiste en atrapar a todos los pokémon, eso quiere decir poder atrapar a los 649 que se encuentran actualmente en el juego, ya que este juego contiene la generación 1 hasta la 5 de pokémon. Adjunto un enlace con más detalle sobre esto de las generaciones: [Generación Pokémon]

Volviendo a la pantalla, en general, el apartado del equipo se irá actualizando. Es decir, si acabas de tener un combate y tu Pokémon Blaziken ha recibido daños o incluso esta a 0 HP, lo podrás consultar aquí.

8.6. Pantalla Tus Pokémon (PC)

Como se ha visto con anterioridad esta es una de las opciones del menú principal, esta es la pantalla donde puedes gestionar tus Pokémon. A simple vista verás en el centro un gran panel semitransparente con iconos de todos los Pokémon que has capturado, cada caja corresponde a una caja distinta (aquí estamos en “Caja 1”, que puedes cambiar con las flechas izquierda/derecha en la parte superior), solo podrás cambiar de caja si tienes suficientes pokémon para ocupar al menos un hueco en esa caja. Los pequeños sprites te permiten reconocer rápidamente quiénes están ahí: desde ese adorable Phione hasta ese imponente Dialga, esto nos permite identificarlos.

A la derecha tienes el área de información del Pokémon seleccionado. Cuando haces clic en uno de tus pokémon, su sprite aparece ahí mismo y aparecerán sus estadísticas: HP, ataque, defensa, ataque especial, defensa especial y velocidad. Además verás una barra de vida que te indica la vida actual del pokémon, por si necesita una curación antes de volver a tu equipo.



Debajo de esos datos encuentras cuatro botones de acción:

Poner en equipo (verde): mueve al Pokémon de la caja a tu equipo.

Curar (rojo): restaura su HP y elimina estados alterados. Ideal para dejarlo listo antes de lanzarlo a la batalla.

Liberar (azul): elimina al Pokémon de tu colección para siempre, liberando espacio en las cajas. Ten cuidado: no hay marcha atrás.

Salir (rojo oscuro): cierra esta pantalla y regresa al mapa o menú principal sin hacer cambios.

Poner en equipo: mueve al Pokémon de la caja a tu equipo, en la primera ranura libre. Si tu equipo ya está lleno (máximo seis), te avisará y te saldrá una ventana para reemplazarlo por uno de ellos, aquí veremos un ejemplo al respecto, seleccionando incluso un pokémon para ver como se ve:

Moverse entre las cajas y los Pokémon es muy fluido: usa el ratón para seleccionar. Conforme cambias de caja, verás cómo los sprites se actualizan y la etiqueta superior indica “Caja 2”, “Caja 3”, etc. Así puedes ver todos los pokémon que tienes guardados.

En resumen, esta interfaz está pensada para que gestiones tu “inventario pokémon” de forma rápida y visual, sin complicaciones. Donde decides quién entra en tu equipo, quién descansa en el PC.

8.7. Pantallas secundarias

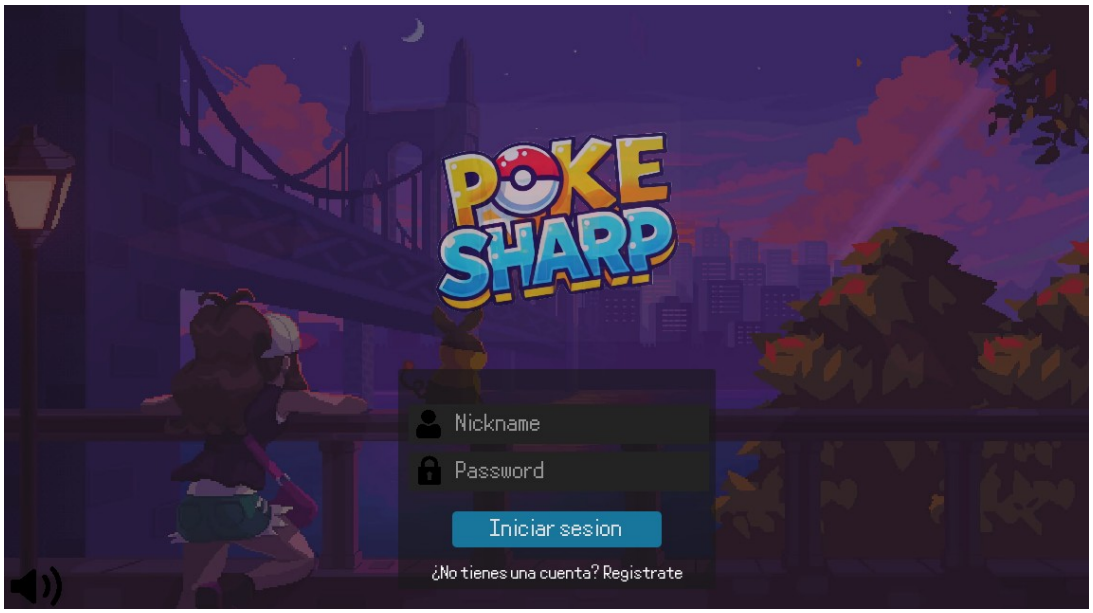
Estas pantallas son secundarias debido a su impacto al juego, carecen de contenido avanzado técnico y no suponen algo demasiado complejo de realizar.

8.7.1. Inicio de sesión

Esta pantalla se muestra al ejecutar el juego, siempre, sin excepción, es una forma de ver que jugador se encuentra jugando para poder consultar a la base de datos los datos requeridos, después iremos paso a paso de como se gestiona todo esto por la parte de backend.

Primero veamos la interfaz, empieza a sonar la música, el logo de PokeSharp hace una animación de tipo latido, volviéndose más grande y pequeño como si se tratará de un corazón latiendo, el fondo es animado, y a la parte de abajo a la izquierda vemos un altavoz para silenciar la música del inicio de sesión, esta simplemente la silencia de inmediato.

La parte más importante es la parte central de la parte inferior, donde tenemos una caja que contiene para introducir el usuario y la contraseña



Si introducimos la contraseña nos aparecerá de la siguiente forma, para hacer más segura la interfaz:

Vamos a probar a registrarnos, ponemos un usuario y una contraseña, ¡Pero cuidado! Este usuario debe ser único, es decir, no pueden existir dos iguales, veamos lo que ocurre si ya existe uno, nos salta un aviso de que el nickname, el usuario, ya está creado en la base de datos, es por eso que no nos deja, al probar con otro que no exista nos dejará, le damos a “Ya tienes una cuenta? Inicia sesión” y cambiaremos el modo para poder iniciar sesión.

Le damos a “Iniciar sesión” y en caso de que las credenciales coincidan cambiará la parte inferior a lo siguiente, de aquí podemos darle a “¡Empieza tu aventura!” y nos llevará directos al juego, sin ningún tipo de pantalla de carga:

Vamos a ver la parte de backend, al registrarse como hemos visto antes hace alguna comprobación, vamos a ir por partes.

Aquí es el inicio del método que se llama al presionar el botón, esta todo encapsulado con un try catch, primero intentamos obtener el usuario mediante el nickname que ha introducido en el TextEdit de nickname.

```
public async void OnBtnRegistrarsePressedAsync()
{
    try
    {
        // Comprobar el nickname por si es único
        PlayersControllers playersControllers = new PlayersControllers();
        Player player = null;

        player = await PlayersControllers.GetPlayerByNickname(nickname.Text);
```

Aquí si es null ya sabemos que no es un usuario que ya ha sido introducido, y si no es null ya que nos está devolviendo un jugador ponemos que este nickname ya está utilizado, como hemos visto anteriormente.

Para el registro lo que hace es serializar un JSON con el nombre de usuario y la contraseña y lo manda.

```
if (player != null) {
    lblError.Text = "¡El  nickname  utilizado  ya  existe!";
} else {
    await playersControllers.InsertPlayer(nickname.Text, password.Text);
```

Aquí por una parte, otorgamos un pokémon inicial para que tenga algún pokémon, calculamos sus estadísticas, que básicamente en lo que se basa es en coger el nivel y hacer cálculos con las estadísticas base que tiene el pokémon y se lo otorgamos al jugador, asignamos en una variable estática.

```
Pokemon pokemonStarter = new Pokemon();

pokemonStarter = await PokemonController.GetPokemonById(4);
pokemonStarter.CalcularStats();

Game.PlayerPlaying = player;

var pokemonPlayersController = new PokemonPlayersController();
bool added = await pokemonPlayersController.CapturarPokemon(pokemonStarter, GetTree().Root.GetNode("Game"));
```

Ahora vamos a ver el inicio de sesión.

El inicio es similar al registro, hace el login y si encuentra el login en este caso asigna directamente el jugador y luego asigna los pokémon del jugador, que ahora entraremos en detalle sobre eso.

```
PlayersControllers playersControllers = new PlayersControllers();
Player player = null;

player = await PlayersControllers.Login(nickname.Text, password.Text);

if (player != null) {
    Game.PlayerPlaying = player;

    await player.asignarPokemons();
```

En asignar pokémon obtenemos todos los pokémon sin excepción del jugador, a partir de ahí a cada pokémon se le vuelven a calcular las estadísticas y dependiendo de la variable inTeam identificamos si se encuentra en el PC o en el equipo, para evitar duplicaciones hacemos un contains en negativo para que no haya pokémon duplicados.

A cada pokémon que entre al equipo se le obtiene un moveset, que es un set de movimientos pokémon, en caso de que no lo tenga creado se le genera introduciendo los movimientos en la base de datos, finalmente se añade el pokémon a la lista del equipo del jugador y nos muestra como hemos visto anteriormente la pantalla para poder seguir con nuestra partida.

```
public async Task asignarPokemons()
{
    var pokemonPlayersController = new PokemonPlayersController();
    var listaAllPokemons = await pokemonPlayersController.GetPokemonsByPlayer(id);

    foreach (Pokemon poke in listaAllPokemons) {
        poke.CalcularStats();

        if (poke.inTeam == -1) {
            if (!listPokemonsCaja.Contains(poke)) {
                listPokemonsCaja.Add(poke);
            }
        } else {
            if (!listPokemonsTeam.Contains(poke)) {
                await poke.getMovesetDB();

                if (poke.Movimientos == null || poke.Movimientos.Count < 4) {
                    GD.Print($"Pokémon {poke.NombreCamelCase} con menos de 4 moves

                    await poke.generateMoveset(true);
                }

                GD.Print($"Pokémon en team:\n {poke}");
                listPokemonsTeam.Add(poke);
            }
        }
    }
}
```

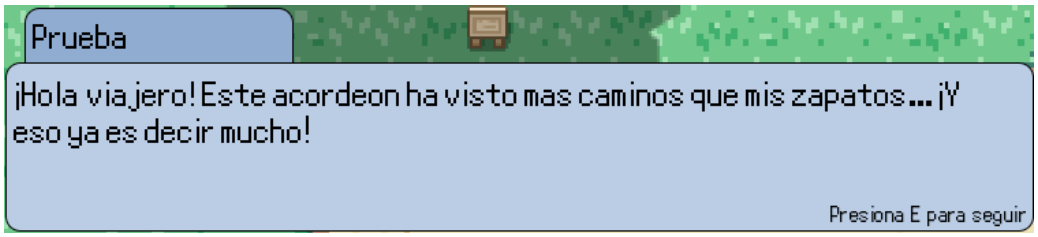
8.7.2. Menú pausa

Este menú pausa se accede mediante la tecla “ESC” y también se quita, actualmente no tiene muchas funcionalidades, únicamente el pausar el jugador y también poder volver a la pantalla de inicio.

8.8. NPCs

Los NPCs (Non-Player Character) del juego son escasos ya que su sistema actual solo permite el dialogo sin poder conversar tu con el NPC, veamos un ejemplo:

Este NPC tiene una animación mientras se encuentra quieto, cuando nos acercamos y le damos a la “E”, es decir, la tecla para poder interactuar nos salta esta ventana que nos muestra su nombre, que en este caso le puse “Prueba” y abajo el texto que nos dice, y abajo derecha nos avisa de que podemos seguir adelante con el dialogo

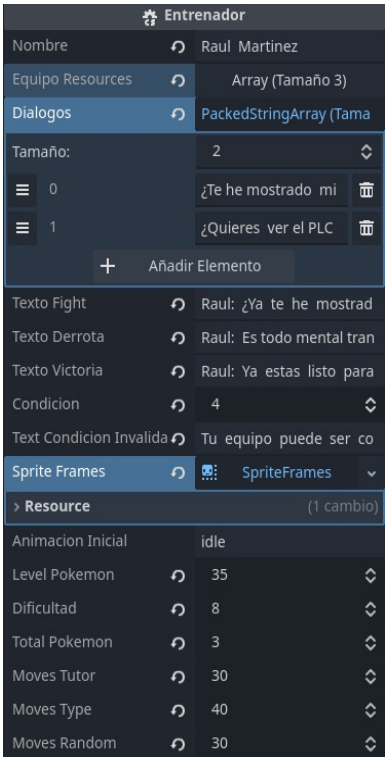


El propósito de estos NPCs es tener únicamente texto y contar curiosidades, historias y en un futuro que puedas desarrollar una mejor conversación con el NPC.

8.9. Entrenadores

Los entrenadores son realmente unos NPCs pero con la posibilidad de poder combatir contra ellos, estos también contienen un dialogo como un NPC cualquiera, pero a su vez contienen muchas propiedades para modificar un combate pokémon.

Contiene estas series de propiedades, veamos lo también con un ejemplo:



- **Nombre** → Nombre del entrenador.
- **Equipo Resources** → Es la lista de recursos, que básicamente contiene los pokémon en formato Recurso, permitiendo poder indicarle una ID y una experiencia base, para que cuando te acerques lo suficiente este consulte a la base de datos esa ID y obtenga al pokémon asignando a su lista interna de pokémon.
- **Diálogos** → Son los textos que muestra al hablar con él.
- **Texto Fight** → El texto que muestra de entrada cuando entras en combate.
- **Texto Derrota** → Texto que muestra cuando has sido derrotado por el entrenador.
- **Texto Victoria** → Texto que es mostrado cuando ganas el combate.
- **Condición** → Cantidad mínima de pokémon necesarios para poder combatir contra este entrenador, sino no podrás.
- **Text Condición Invalida** → Texto que se mostrará si intentas hablar con el sin tener el mínimo de pokémon de la condición.
- **Animación Inicial** → Animación interna que se muestra de su sprite.
- **Level Pokemon** → Nivel de todos los pokémon mínimo, esto no aplica si la media de tu equipo supera este nivel.
- **Dificultad** → Este número positivo o negativo modifica las estadísticas base de los pokémon para hacer el combate más difícil o más fácil.
- **Total Pokemon** → Número total de pokémon que contiene el entrenador, en caso de que la lista de recursos contenga menos de este número forzará a obtener un pokémon aleatoriamente, con esto se consigue hacer que los combates contra entrenadores sean más inesperados y distintos.
- **Moves Tutor** → Porcentaje del que obtiene los movimientos del pokémon que se aprenden por tutor.
- **Moves Type** → Porcentaje del que obtiene los movimientos del pokémon que es del mismo tipo.
- **Moves Random** → Porcentaje del que obtiene los movimientos del pokémon de forma aleatoria.

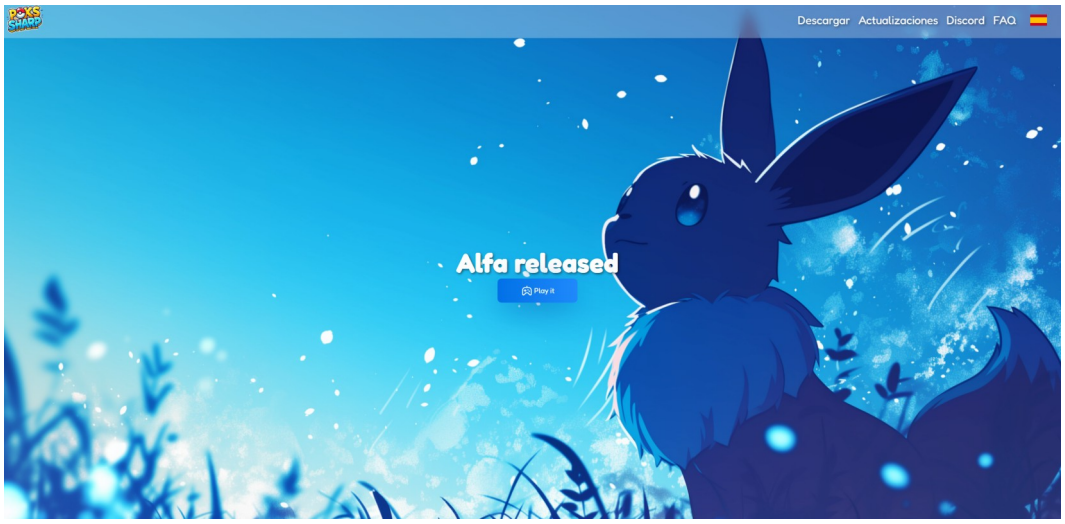
8.10. Página web

A inicios del segundo año se empezó el desarrollo de la idea y una página web para publicitar el juego, y que el seguimiento de las actualizaciones del proyecto sea más fácil de seguir. Se ha utilizado HTML, CSS y JavaScript para el desarrollo de la página web.

Página web publicada con **GitHub Pages**: https://maek0s.github.io/PokeSharpWeb/

Página principal:

Contiene una pequeña descripción del proyecto y un botón de enlace directo para ver las actualizaciones.



Actualizaciones

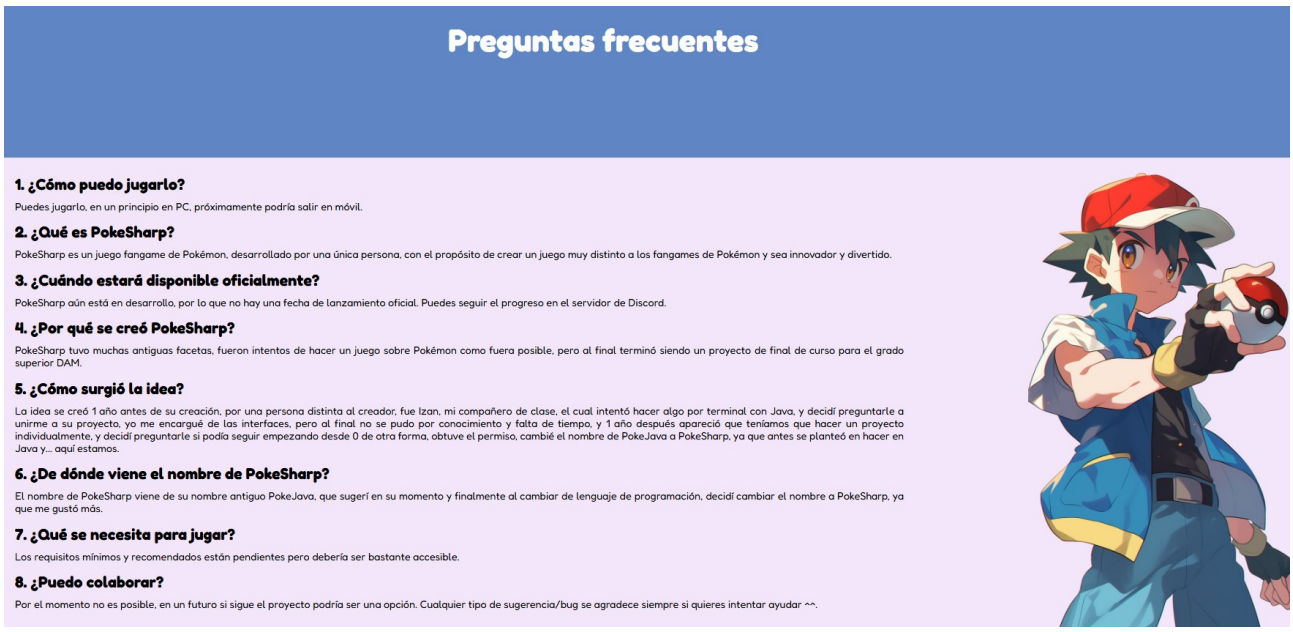
Contiene las distintas versiones con sus funcionalidades añadidas y bugs arreglados, dividiendo la última versión para hacerlo más claro visualmente.

Discord

No contiene una página propia, es un enlace directo al discord del proyecto.

FAQ

Es una página con preguntas y respuestas frecuentes que puede llegar a tener un usuario.



8.11. Control de versiones

Se ha gestionado el proyecto del juego y la página web con 2 repositorios desde GitHub.

Proyecto principal - Maek0s/PokeSharp: Repositorio oficial de PokeSharp
Página web - Maek0s/PokeSharpWeb: Web of PokeSharp

En el caso del proyecto principal se realizó los commits de forma dividida para que las funcionalidades añadidas se vean separadas con el objetivo de detectar fallos en el juego más rápidamente, a raíz de empezar a sacar versiones alfa se dividieron los commits en directamente las versiones sacadas.

9. Discord

¿Qué es Discord?

Discord es algo muy conocido en el mundo de los videojuegos, es una plataforma digital que permite unir a comunidades y también poder comunicarte con otros usuarios, aquí dejo con más detalle al respecto: Discord: qué es y para qué sirve

¿Por qué hice un servidor de Discord?

Normalmente los juegos suelen tener un servidor de Discord oficial, es la mejor forma de notificar a los usuarios de nuevas actualizaciones, avances o directamente recibir comentarios sobre que les parece el juego, me parecía una gran oportunidad el hacer uno y decidí crear uno, para mantener a una comunidad unida si existe en un futuro.

También porque puede llegar a ser una forma de publicitar el juego, entre amigos en vez de mandar la página web o el repositorio de GitHub puedes mandar una invitación al servidor y de ahí que le interese o no el proyecto.

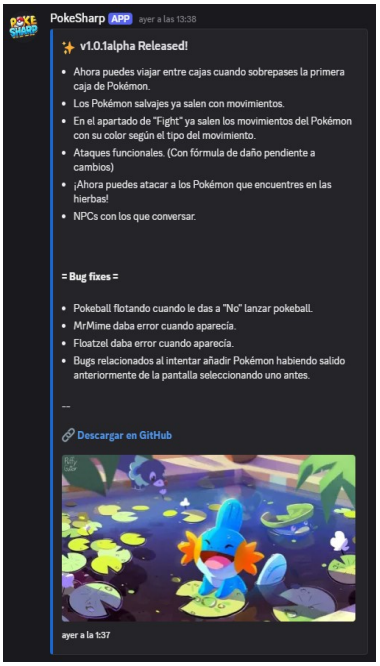
¿Realmente es necesario?

Si y no, al fin al cabo el unir a una comunidad cuesta, y que se mantengan unidos es más difícil, por eso existen cosas como Reddit, que podría llegar a decirse que es parecido a Discord pero en este caso Reddit es más una especie de foro.

El compartir tu experiencia jugando a un juego solitario con otros puede llegar a derivar en más entretenimiento para todos.

¿Cuál es el uso que le doy para el proyecto?

Como ya dije con anterioridad principalmente es para mantener a una comunidad unida, pero también es para ir anunciando las nuevas versiones del proyecto, posibles avances, recibir reportes de bugs encontrados y sugerencias de las jugadores.



Durante su desarrollo cuando ya era un juego más o menos con posibilidad de jugarlo publiqué dos versiones alfa, que son versiones muy tempranas de su desarrollo y hago una recopilación de lo que está incorporado en la versión, aunque esto también se encuentra en GitHub.

10. Diagrama Entidad-Relación

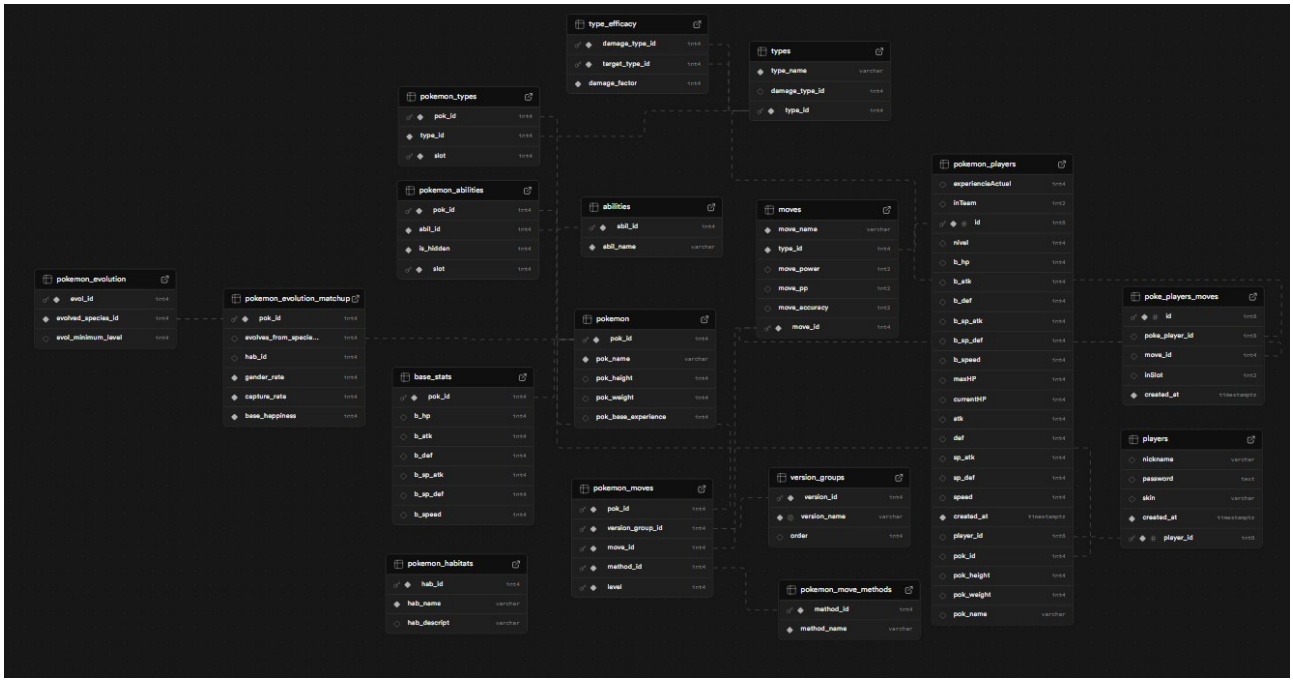
La base de datos utilizada en el proyecto está compuesta por un total de **17 tablas**. De estas, **14** provienen de una base de datos pública relacionada con Pokémon, lo que permitió ahorrar tiempo en la estructuración inicial y centrarme en la lógica del juego.

Las **3 tablas restantes** han sido diseñadas específicamente para gestionar la información de los jugadores. Gracias a ellas, es posible relacionar a cada usuario con sus respectivos Pokémon, así como con otros datos importantes como:

- Sus **credenciales de inicio de sesión**, necesarias para acceder al juego con su cuenta.
- La **asignación de pokémon a cada jugador**, permitiendo así almacenar su progreso.
- Los **movimientos que conoce cada uno de sus pokémon**, lo que permite que cada pokémon tenga habilidades únicas dentro del combate.

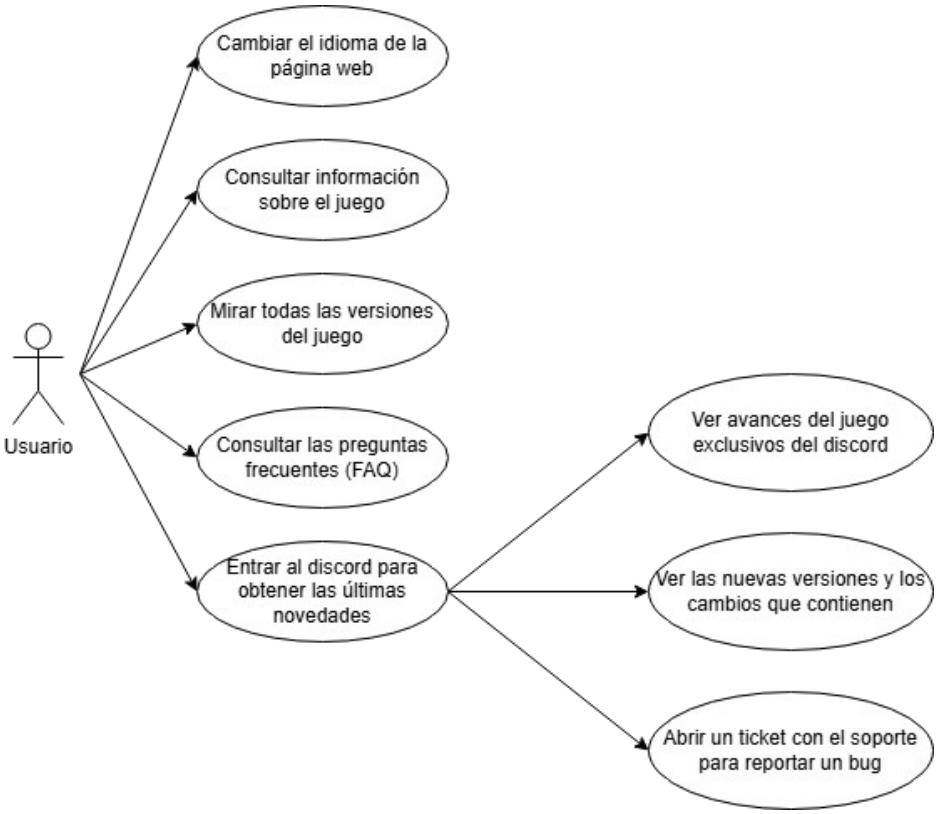
Estas tablas personalizadas han sido necesarias para otorgar al proyecto de un sistema de guardado automático, permitiendo un sistema de usuario funcional que se guarda cada vez que ocurre algo importante.

Base de datos de Pokémon: [brianr852/Pokemon-Database]



11. Esquema funcional

Casos de uso de la página web.



12. Dificultades encontradas y cómo las ha superado

12.1. Aprendizaje de nuevas tecnologías

Una de las principales dificultades a lo largo del proyecto ha sido la falta de conocimientos previos en muchas de las tecnologías necesarias. Para desarrollar el juego, fue imprescindible aprender desde cero aspectos como:

- El motor de videojuegos **Godot Engine**
- El lenguaje de programación **C#**, de forma más profunda
- El uso de **Supabase**, una plataforma que utiliza **PostgreSQL**

Superar esta dificultad supuso dedicar muchas más horas de lo realmente planeado. Principalmente usando el método de prueba y error, junto a mucha investigación. Este enfoque me permitió no solo resolver los problemas concretos que iban surgiendo, sino también adquirir un conocimiento sólido de cada nueva herramienta que incorporaba al proyecto.

12.2. Problemas con mayúsculas en los scripts

Al realizar la exportación de la primera versión alfa disponible tuve bastantes problemas que me dejaron atascado unos días en ello, el problema era que al probar el juego exportado no funcionaba nada como de normal, lo que hacia que esto suceda era las mayúsculas en algunos scripts, esto por supuesto se solucionó cambiando a todos los scripts a minúsculas, a raíz de eso todo funcionó.

12.3. Ambición frente a limitaciones de tiempo

Desde el inicio, la idea del proyecto era mucho más ambiciosa que el resultado actual. Sin embargo, pronto me di cuenta de que muchas de las funcionalidades que había planteado eran más complejas de lo que parecían, y que el tiempo disponible no iba a ser suficiente para realizarlas todas.

Ante esta situación, decidí reorganizar el desarrollo del juego, dividiendo las tareas en tres niveles de prioridad: sistemas principales, secundarias y complementarias. De esta forma me permitió centrarme en construir una base sólida del juego, asegurando un resultado funcional, sobre el que se podrían seguir construyendo futuras mejoras.

12.4. Problemas en los tilesets demasiado grandes

En el proyecto a la hora de diseñar el mapa usaba tilesets, que son imágenes con texturas que introduce en el juego para poder diseñarlo, lo que ocurría es que en un portátil donde probé el juego habían texturas que no cargaban, al ocurrir esto investigué como dependiendo el dispositivo utilizado se veían y no. Se trata de un límite que tiene la memoria VRAM de la gráfica, al ser un portátil la tarjeta gráfica integrada no era demasiado bueno y no cargaba imágenes tan excesivamente grandes, suele rondar igual 4000 pixeles, en este caso usaba tilesets de un

tamaño de entre 20000 y 4000 pixeles de altura, ya que son imágenes muy largas para contener todas las texturas.

Pensé en dos posibles soluciones:

1. Crear un tileset propio de lo que todo lo que utilizo.

Ventajas:
- Esta todo más ordenado.
- Nunca darán fallos de exceso de tamaño ya que sería también más ancho y utilizaría menos espacio.

Desventajas:
- Poner todas las texturas usadas en un tileset propio.
- Mantener ese tileset propio actualizado cuando utilice nuevas texturas.

2. Dividir las imágenes de los sprites actuales a imágenes más cortas.

Ventajas:
- Fácil de hacer y sigo teniendo todo para poder añadir más texturas.
- No debe dar problemas de tamaño ya que lo reduzco lo suficiente para evitar esto.

Desventajas:
- Distintas imágenes para realmente solo una.

Finalmente me quedé con la segunda opción que era más sencilla y simple, empecé a dividir los tilesets que daban problemas y a dibujar de nuevo con el nuevo tileset todo lo que antes no cargaba correctamente y finalmente ya cargaban las texturas en un portátil de gama media.

13. Diseño

La parte de diseño del proyecto

13.1. Estructura carpetas

La estructura de carpetas del proyecto esta estructurada de una forma común en los videojuegos también esta algo adaptado a el sistema de escenas que contiene Godot, la estructura es la siguiente:

📁 **addons** → Son las extensiones que contienes en tu entorno de trabajo (Plugins)
📁 **assets** → Imágenes utilizados que están dentro del juego como algo visible.
📁 **data** → JSONs que maneja el juego (JSON con las coordenadas de las puertas)
📁 **multimedia** → Fuentes, videos e imágenes que se usa más de tipo presentación o algo no tan jugable.
📁 **resources** → Recursos creados mediante el editor (Los pokémon que contienen los entrenadores son recursos que se almacenan aquí)
📁 **scenes** → Las escenas creadas en el juego ordenadas por más carpetas.
    ├── 📁 **important** → Escenas importantes, como el juego en si y el personaje principal.
    ├── 📁 **interfaces** → Escenas que se usa como interfaces.
    ├── 📁 **mapas** → Escenas que son mapas o interiores.
    ├── 📁 **others** → Escenas no contempladas por el resto de carpetas.
    ├── 📁 **smallthings** → Escenas con contenido mínimo.
    └── 📁 **ui** → Escenas que aparecen en la interfaz del jugador.
📁 **scripts** → Código del juego dividido en carpetas.
    ├── 📁 **database** → Código relacionado con la gestión de la base de datos.
    ├── 📁 **models** → Clases que se utilizan para el código.
    ├── 📁 **others** → Archivos de código que no se caracterizan por el resto de carpetas.
    ├── 📁 **systems** → Archivos de los sistemas del juego.
    ├── 📁 **transitions** → Transiciones del juego.
    ├── 📁 **ui** → Código de las interfaces del juego.
    └── 📁 **utils** → Archivos que contienen utilidades que se utilizan por el código.

Archivos sin carpeta propia

📄 Game.cs
📄 MainCharacter.cs
📄 Transition.cs

14. Implementación

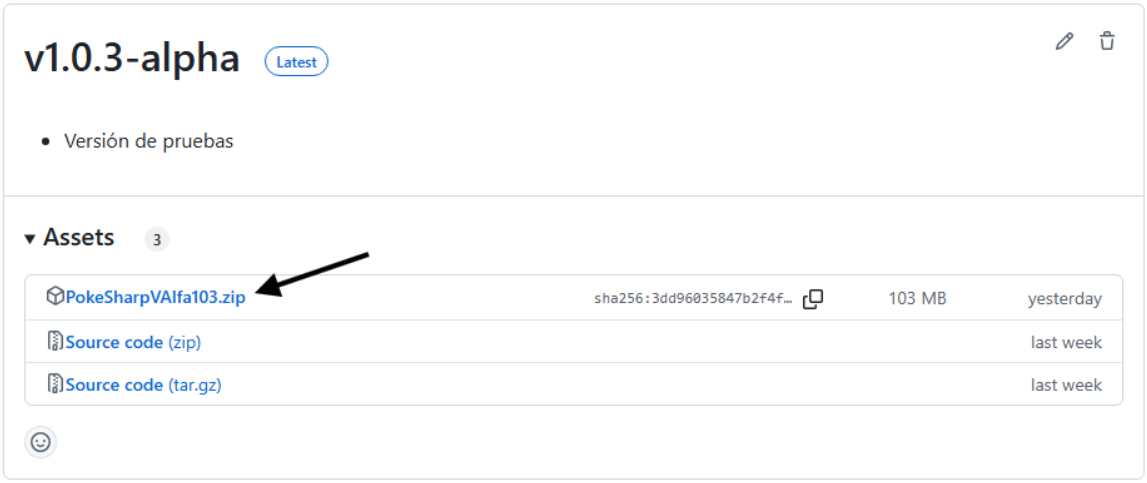
14.1. Entorno de desarrollo

Motor de juego: Godot 4.4.1.stable – versión para C#
Lenguaje de programación: C#
Editor de código: Visual Studio Code
Sistema operativo: Windows 11
Control de versiones: GitHub, mediante GitHub Desktop

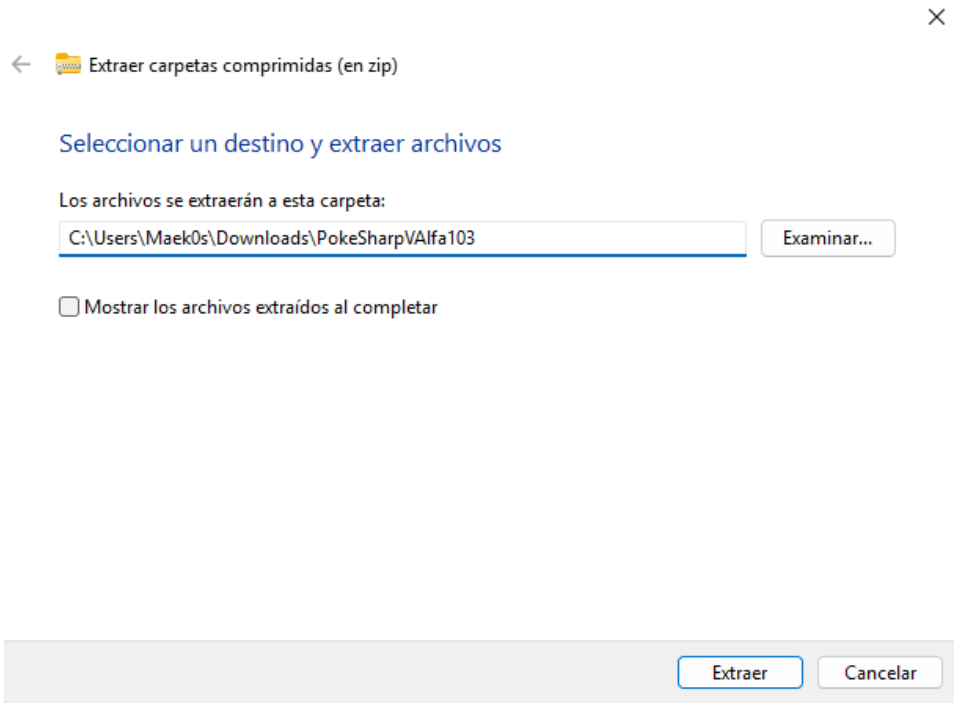
14.2. Instalación

La instalación de este videojuego es muy simple, debemos acceder al repositorio del juego y entrar a las “Releases” Releases · Maek0s/PokeSharp

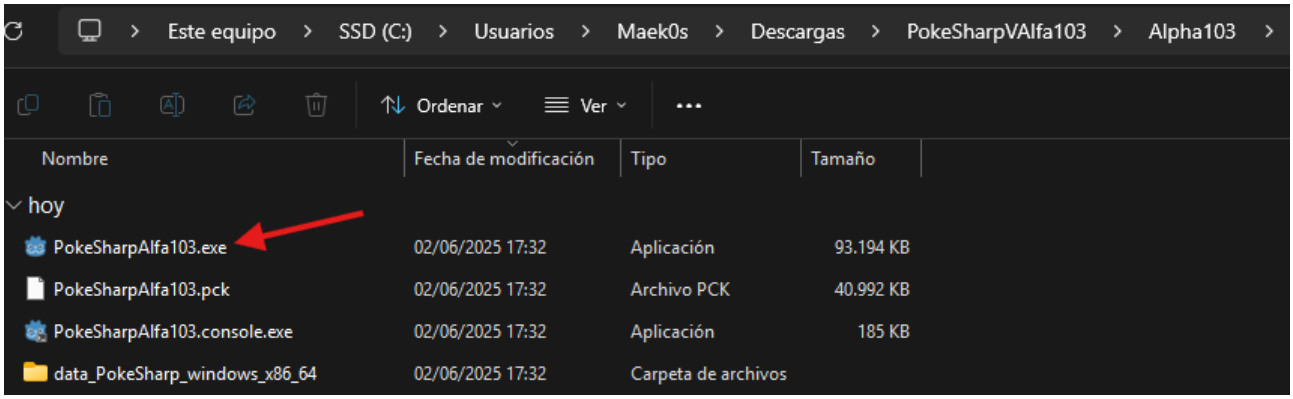
Desde ahí accedemos preferiblemente a la última versión > Assets y le damos a **PokeSharpVAlfa103.zip** en este caso



Extraemos el .zip en cualquier sitio para que se guarde en una carpeta única con todo.



Accedemos a la carpeta extraída > Alpha103 y ejecutamos el **PokeSharpAlfa103.exe**



Como recomendación se puede llegar a crear un acceso directo a este .exe y dejarlo en el escritorio para su acceso más cómodo ya que el .exe debe estar en la carpeta junto al data.

15. Ideas a futuro

- Más pokéballs, más mapas, más variación de texturas por el mapa.
- Más NPCs y un sistema de diálogos que permita diversas tipos de dialogo con un mismo NPC.
- Incorporación de Shinys en el juego.
- Combate dos contra dos.
- Gimnasios con medallas.
- Pokedex.
- Gestión de pokémon de otros jugadores desde un panel admin.
- Sistema de dinero in-game para comprar objetos.
- Mejora en el sistema de combate, funcionalidades y animaciones.
- Curar los pokémon con las pociones o un centro pokémon.
- Incorporación de maquinas arcade o en si minijuegos.
- Cliente para actualizar automáticamente el juego.
- Página web más útil y conectada con el juego.
- Página web con las actualizaciones de forma dinámica.

16. Experiencia personal y conclusiones finales

La experiencia personal durante este proyecto ha sido muy positiva. De hecho, ha sido tan interesante que me ha llevado a plantearme la posibilidad de especializarme en desarrollo de videojuegos. Volviendo al proyecto en sí, ha sido una experiencia que sin duda repetiría, ya que he aprendido muchísimo tanto a nivel técnico como a nivel personal. Uno de los aspectos más valiosos ha sido mejorar mi capacidad de organización y gestión del tiempo para cumplir con los plazos establecidos.

Gracias a este trabajo, he comprendido que llevar a cabo un proyecto sólido y bien hecho no depende exclusivamente del talento, sino de la dedicación y el tiempo que se le invierte.

La principal conclusión que saco de este proyecto es que ha sido, sin duda, la mejor elección que podría haber hecho. Es un proyecto con el que he disfrutado incluso en los momentos que estuve cansado.

Siento que el resultado cumple con las expectativas que tenía al inicio. A futuro, me gustaría seguir trabajando en este juego como hobby, ampliando sus funcionalidades y puliendo aquellos aspectos que podrían mejorarse. Este proyecto ha sido una excelente oportunidad para aprender, crecer y confirmar mi interés por el desarrollo de videojuegos.

17. Referencias bibliográficas y webgrafía

Imágenes utilizadas:

- Imagen utilizada en la pantalla de “Tus Pokémon” para desplazarte entre cajas: Flaticon – flecha
- Fondo de inicio de sesión: Hilda And Tepig Watching Castelia City Pokemon - MoeWalls

Enlaces consultados:

- Documentación de Godot 4 C#: C#/.NET — Godot Engine (latest) documentation in English

18. Agradecimientos

- Brianr852, un usuario de GitHub que publicó una base de datos muy completa de Pokémon.

- A todas esas personas que suben texturas de Pokémon por internet.

- A las personas encargadas de Pokémon Essentials por hacer los fan games de Pokémon posibles, otorgando muchas texturas e información de ayuda.

- Especial agradecimiento a Izan López por concederme el permiso para poder empezar de cero el proyecto que tuvimos pensado hacer en un pasado, y también por haber probado el alfa 1.0.0.

- También agradecer a ciertos amigos por probar el juego en su versión alfa, y a una amiga por hacer retoques al logo del juego.