J Reto #7

Proyecto 3

Pokemón

Equipo 6

Contreras Martínez Alan Gael
Gómez Rosales Roberto Josué.
Guzmán Fernández Andrés Rogelio
Martínez Ruiz Abdiel Barush
Santos Mateos Oswaldo

Objetivo

Desarrollar un simulador de batallas Pokémon en Flutter/Dart

Aplicar conceptos de POO, UML, patrones y manejo de errores

Integrar lógica de batalla, UI, audio y persistencia

Demostrar un diseño modular, mantenible y extensible

Arquitectura general del sistema

División en capas:

models: lógica de datos

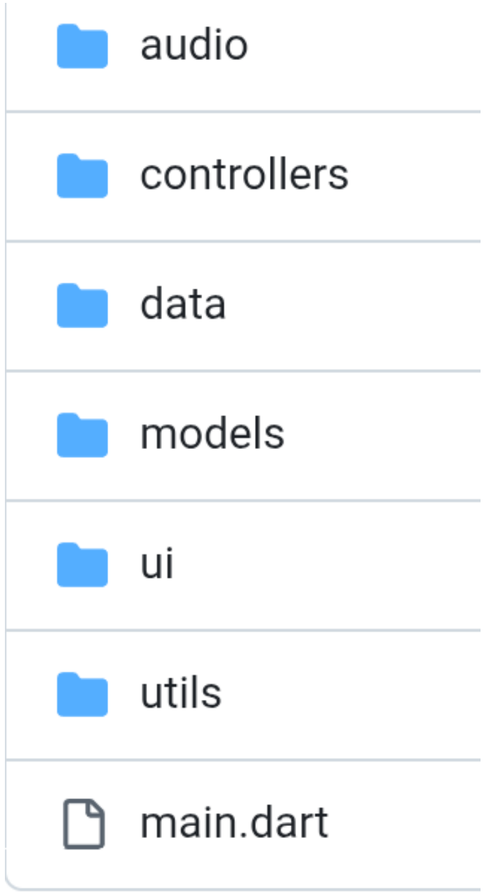
controllers: motor de batalla y flujo de juego

ui: pantallas y widgets GBA

data: catálogos, tablas y configuraciones

audio: música y efectos

Evita que la lógica, la UI y los datos se mezclen



Modelos principales

pokemon.dart

Define estadísticas, tipos, vida, estado, movimientos

movimiento.dart

Tipo, categoría (físico/especial/estado), poder, precisión, prioridad

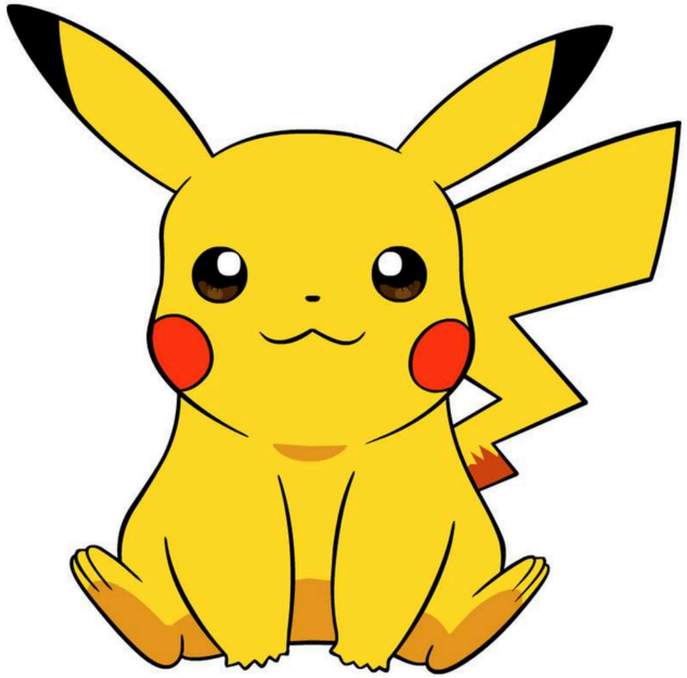
item.dart

Ítems de batalla: curar HP, curar estado, revivir

```
//Constructor
Pokemon({
  required this.nombre,
  required this.tipoPrimario,
  required this.baseHP,
  required this.baseAtk,
  required this.baseDef,
  required this.baseSpAtk,
  required this.baseSpDef,
  required this.baseSpeed,
  this.nivel = 50,
  required this.movimientos,
}) {
```

Clase Pokemon

Un pokemon tiene nombre, tipo primario, HP, ataque, defensa, ataque especial, defensa especial, velocidad de ataque y nivel además de una lista de movimientos para cada uno. Entonces para aprovechar el paradigma tenemos que hallar los puntos de intersección de los pokemons, para empezar, todos son precisamente pokemons, así que es conveniente crear una clase de pokemons, pero hay diferentes tipos, así que de acuerdo a su tipo algunos tendrán ciertas debilidades o fortalezas, es ahí donde entra la herencia y el polimorfismo, pues ya con eso solo quedaría modificar estadísticas y darle nombre propio a cada objeto pokemon teniendo almacenados **53** de estos para utilizarse, cada uno con ciertas estadísticas, tipo y set de movimientos



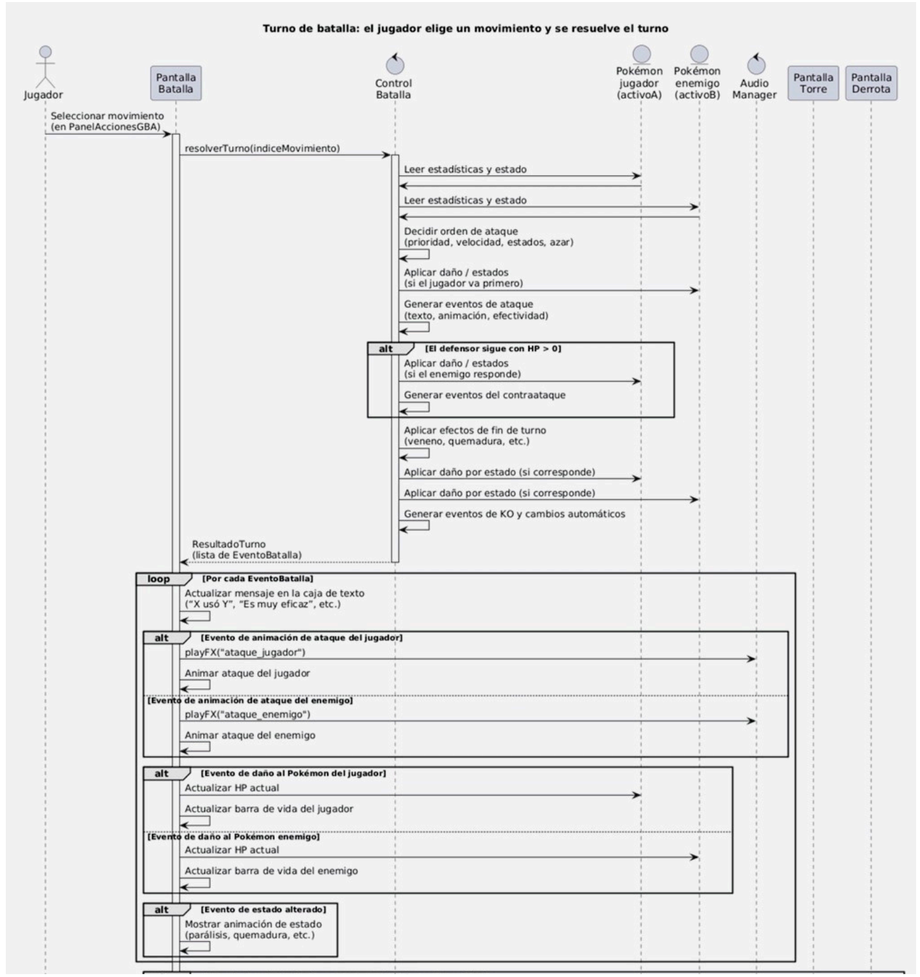
Clase Movimiento

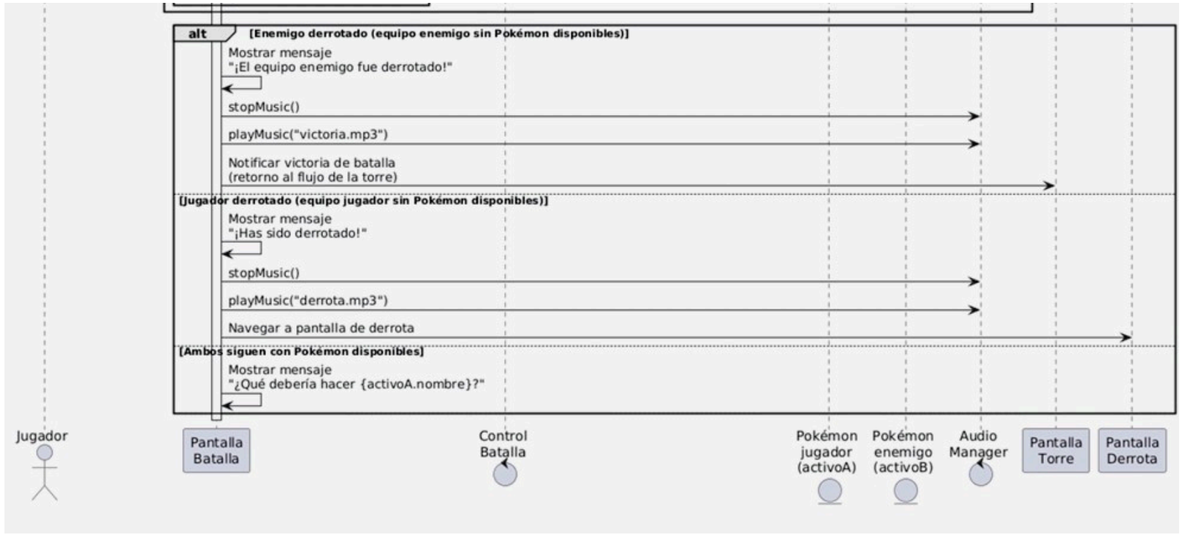
Para poder usar los movimientos se da un caso similar, pues estos pueden ser físicos, especiales o de estado, cada uno ejercerá un daño diferente teniendo en cuenta la defensa o el tipo del pokemon, e inclusive si es que tiene prioridad de atacar primero como lo es el ataque rápido mediante un booleano. Entonces para poder tener una correcta gestión de elementos, sus pros y sus contra debemos hacer uso de tablas de tipo

Clase Item

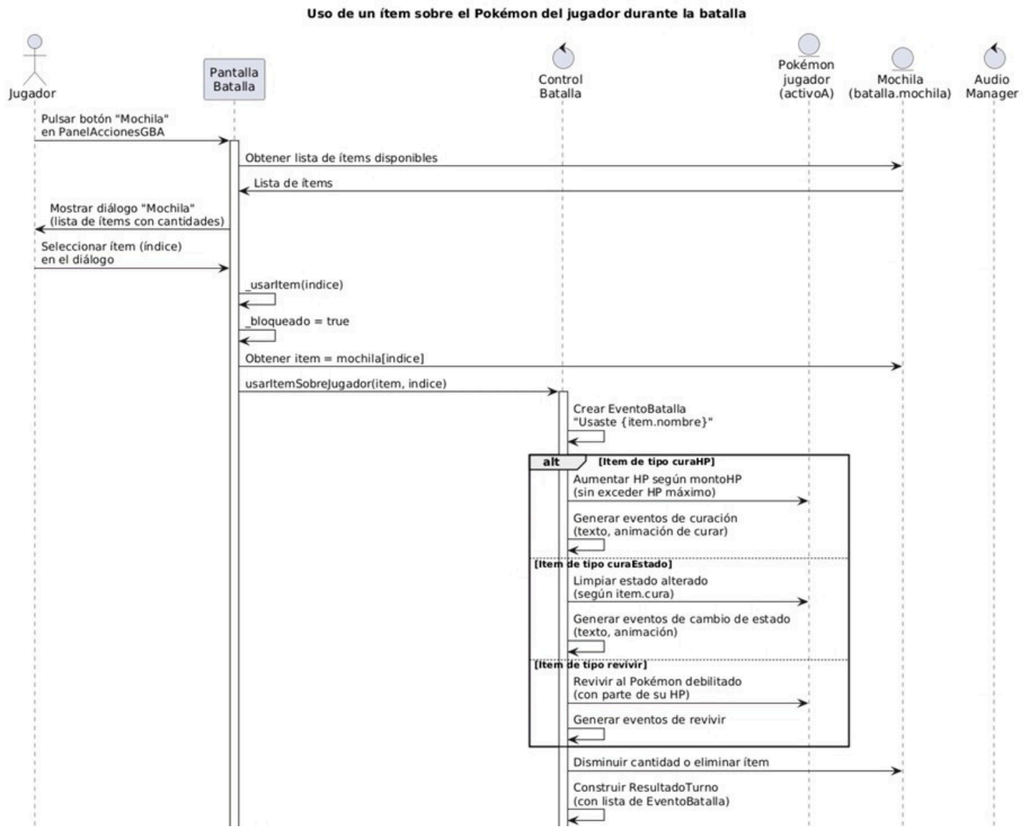
En estos objetos también se puede ver rasgos de herencia y polimorfismo pues no importa cual dificultad elijas todos son un tipo de Item el cual dependiendo de su función es de tipo curaH, tipo curaEstado o tipo revivir, cada uno posee un nombre, una cantidad dentro de la bolsa y un monto si es que recupera vida o repara un estado. Entonces podemos decir que montoHP y cura son opcionales porque cada tipo de item usa solo uno de ellos.

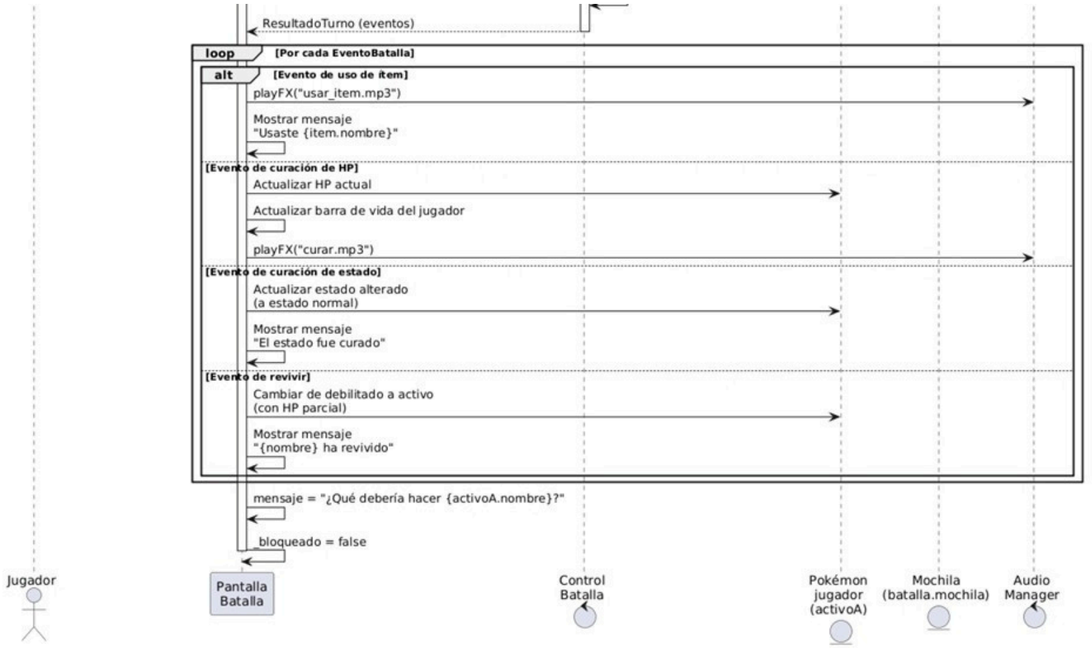
Flujo de trabajo al realizar un movimiento y resolver un turno





Flujo de trabajo al usar un item





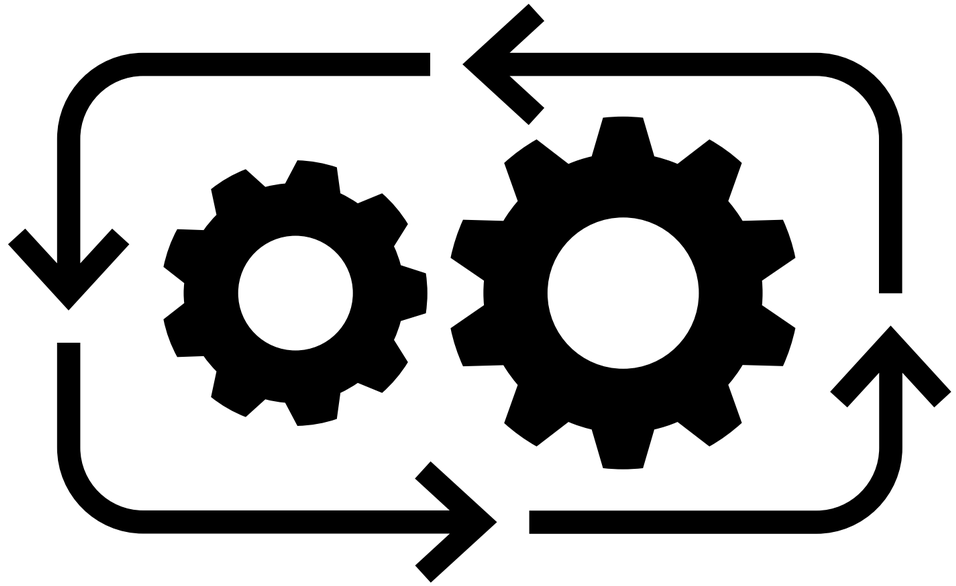
Audio:

Se importará los paquetes de just _audio y audio player con los cuales crearemos una clase audioManager manejando música, efectos de sonido y volumen, con justaudio para música, audioplayers para efectos FX, shredpreferences para guardar el volumen. Se utilizo singleton para su operación, manejándolo como un servidor global de audio. Para mantener la variedad se optó por cierta aleatoriedad en la música por lo que a veces cambiará dependiendo de la partida que estés jugando, aunque si es que pasas un considerable tiempo en un fase será notable que la música está en loop

Controladores

Batalla.dart:

- Maneja la lógica de las batallas específicamente
- Funcionamiento mediante eventos
- Administración de eventos
- Estructurar animaciones
- Estructurar sonido



Manager.dart:

- Maneja la lógica del juego
- Progreso en la torre batalla
- Items en la mochila
- Estructura singleton
- Guarda nombre, spray, datos en general

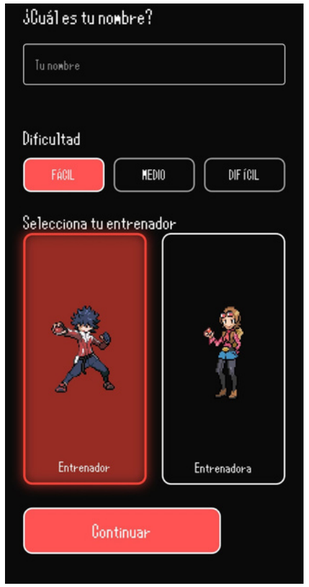
Estructura del proyecto | Resultado

UI:

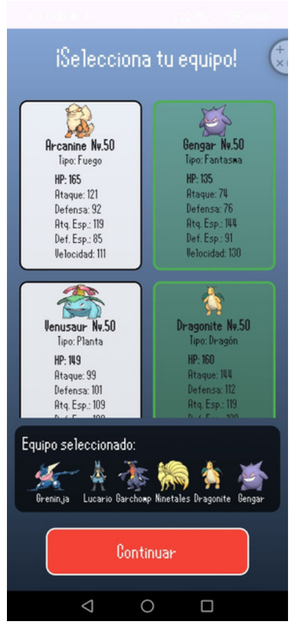
main.dart
Inicia la app y configura la ruta inicial

pantalla_inicio.dart
Menú principal: nueva partida u opciones

pantalla_nombre.dart
Nombre del jugador, sprite, dificultad

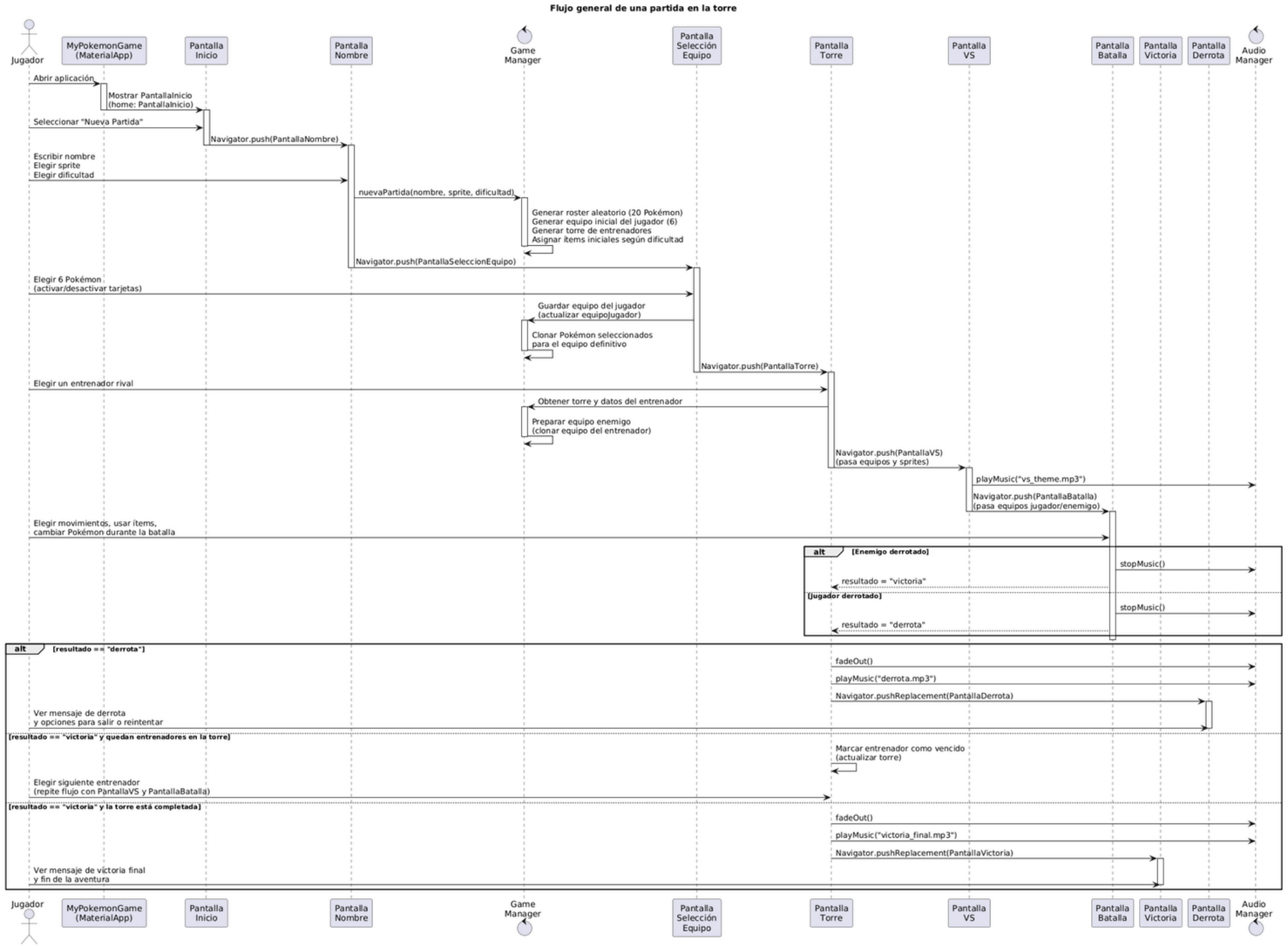


pantalla_seleccion.dart
Selección de equipo: 20 → 6 Pokémon



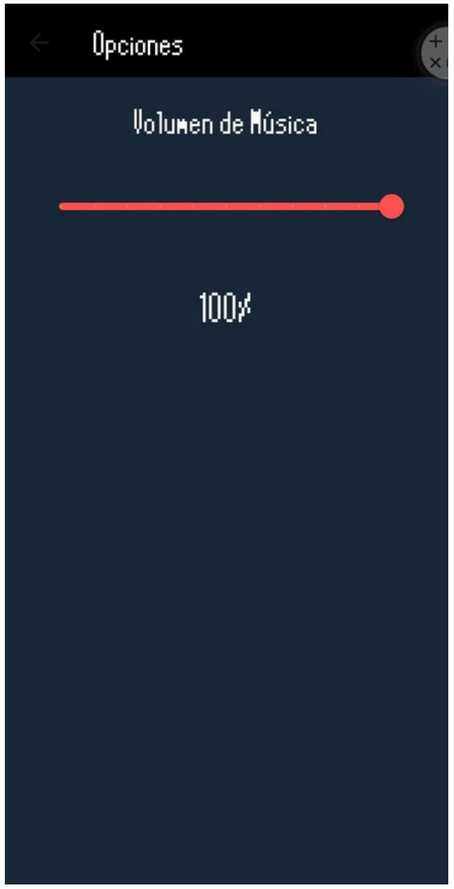
pantalla_torre.dart
Lista de entrenadores a desafiar

UI (UML)


Flujo general de una partida en la torre
Jugador
MyPokemonGame (MaterialApp)
Pantalla Inicio
Pantalla Nombre
Game Manager
Pantalla Selección Equipo
Pantalla Torre
Pantalla VS
Pantalla Batalla
Pantalla Victoria
Pantalla Derrota
Audio Manager
Abrir aplicación
Mostrar PantallaInicio (home: PantallaInicio)
Seleccionar "Nueva Partida"
Navigator.push(PantallaNombre)
Escribir nombre
Elegir sprite
Elegir dificultad
nuevaPartida(nombre, sprite, dificultad)
Generar roster aleatorio (20 Pokémon)
Generar equipo inicial del jugador (6)
Generar torre de entrenadores
Asignar ítems iniciales según dificultad
Navigator.push(PantallaSeleccionEquipo)
Elegir 6 Pokémon (activar/desactivar tarjetas)
Guardar equipo del jugador (actualizar equipoJugador)
Clonar Pokémon seleccionados para el equipo definitivo
Navigator.push(PantallaTorre)
Elegir un entrenador rival
Obtener torre y datos del entrenador
Preparar equipo enemigo (clonar equipo del entrenador)
Navigator.push(PantallaVS) (pasa equipos y sprites)
playMusic("vs_theme.mp3")
Navigator.push(PantallaBatalla) (pasa equipos jugador/enemigo)
Elegir movimientos, usar ítems, cambiar Pokémon durante la batalla
alt [Enemigo derrotado]
stopMusic()
resultado = "victoria"
[Jugador derrotado]
stopMusic()
resultado = "derrota"
alt [resultado == "derrota"]
fadeOut()
playMusic("derrota.mp3")
Navigator.pushReplacement(PantallaDerrota)
Ver mensaje de derrota y opciones para salir o reintentar
[resultado == "victoria" y quedan entrenadores en la torre]
Marcar entrenador como vencido (actualizar torre)
Elegir siguiente entrenador (repite flujo con PantallaVS y PantallaBatalla)
[resultado == "victoria" y la torre está completada]
fadeOut()
playMusic("victoria_final.mp3")
Navigator.pushReplacement(PantallaVictoria)
Ver mensaje de victoria final y fin de la aventura

Ejecucion final

Inicio

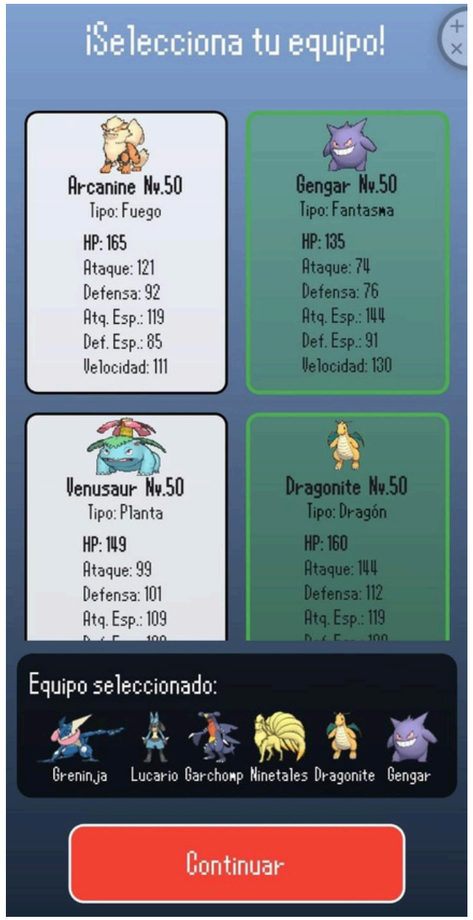


Volumen

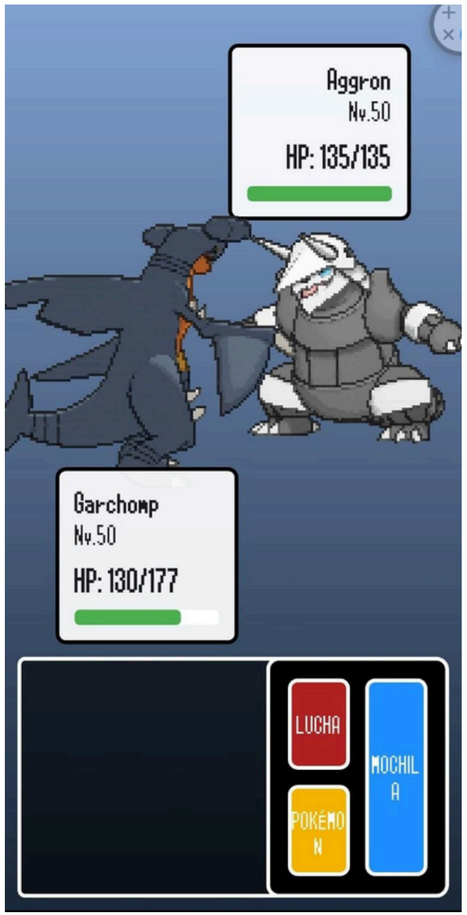
Personaje

Selección de equipo

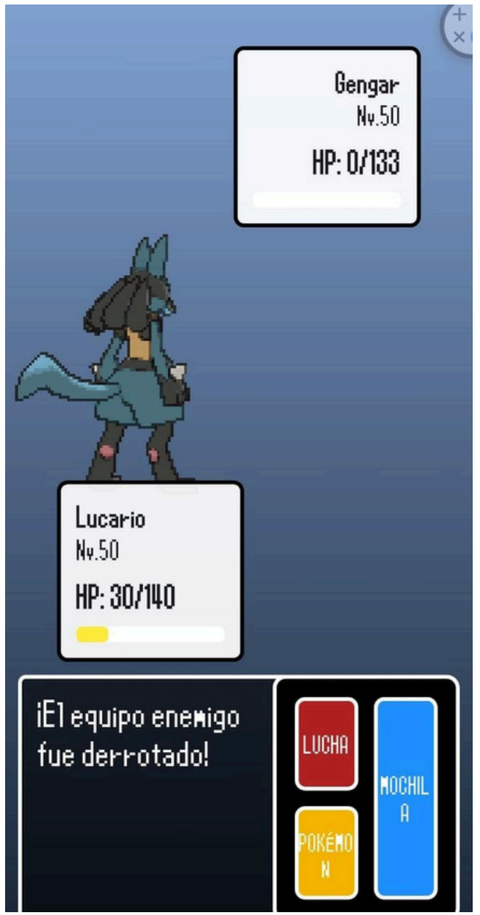
Ejecucion final



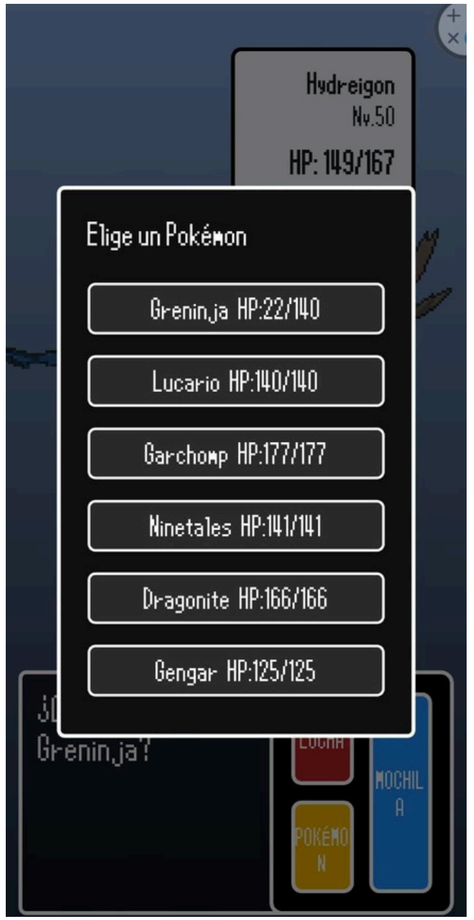
Equipo seleccionado



Batalla



Victoria contra rival

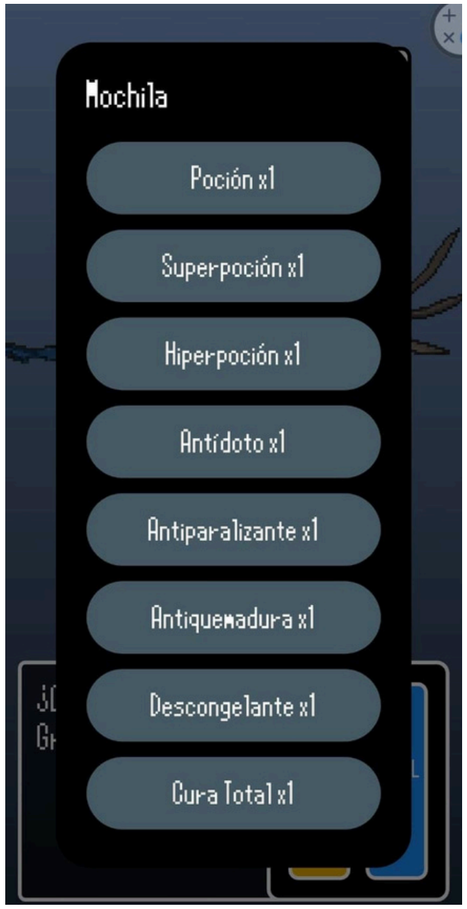


Cambio de pokemon

Estructura del Reto

Resultado

Ejecucion final



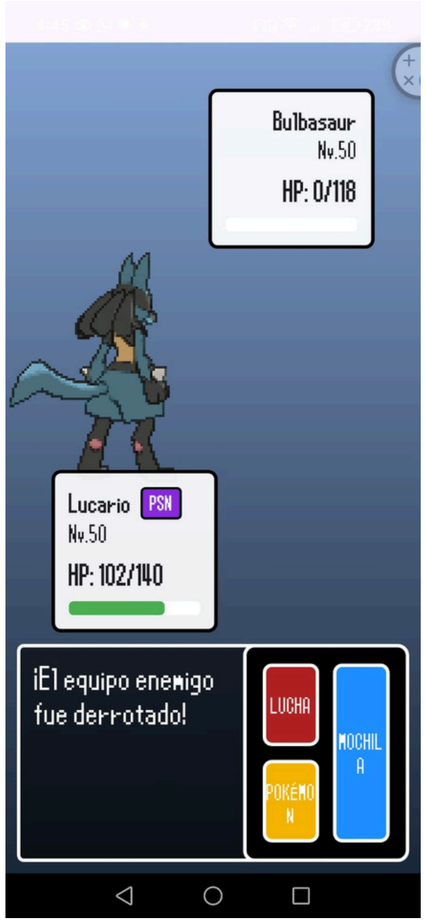
Mochila

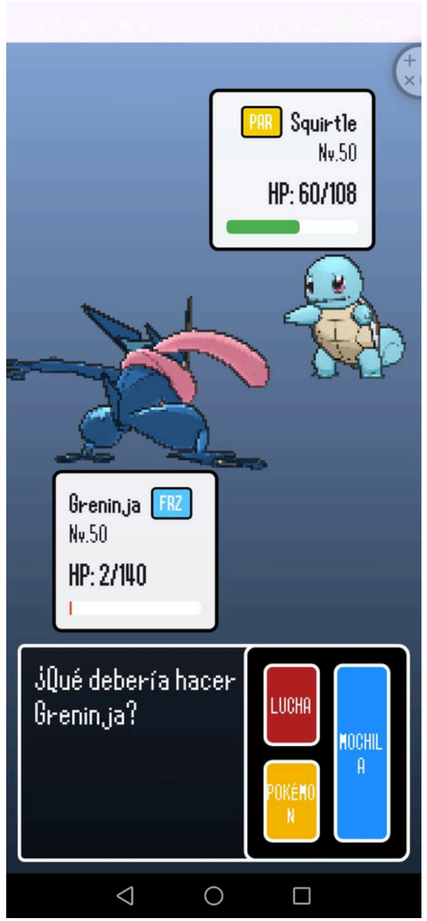
Movimientos

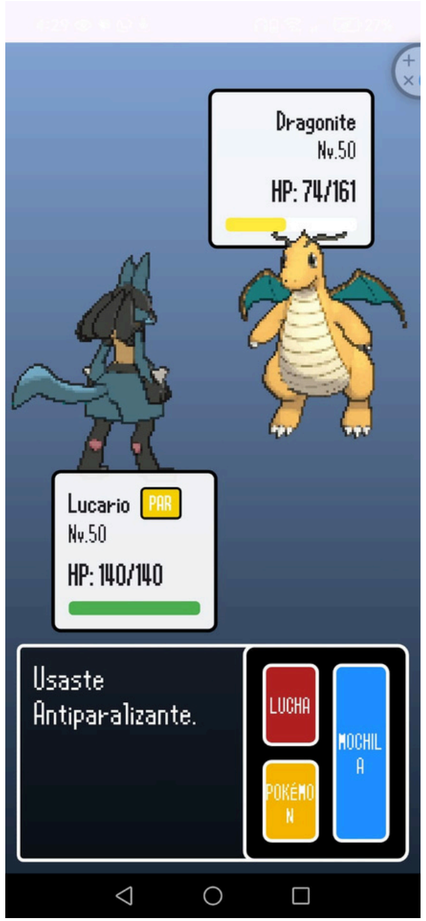
Victoria

Derrota

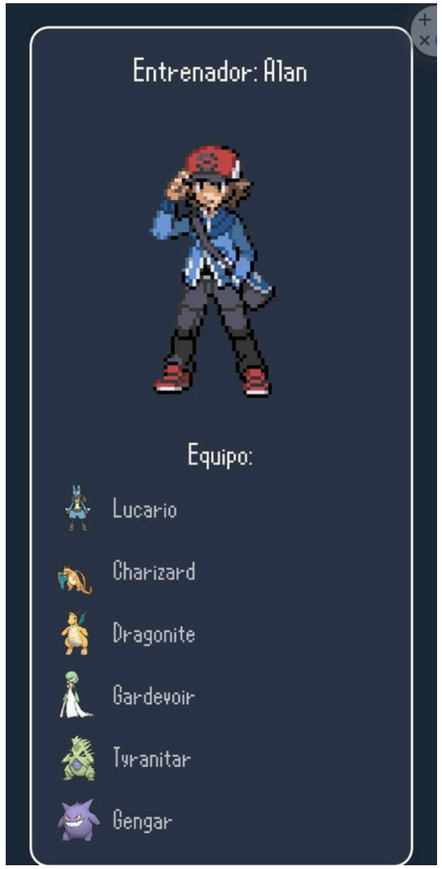
Estados

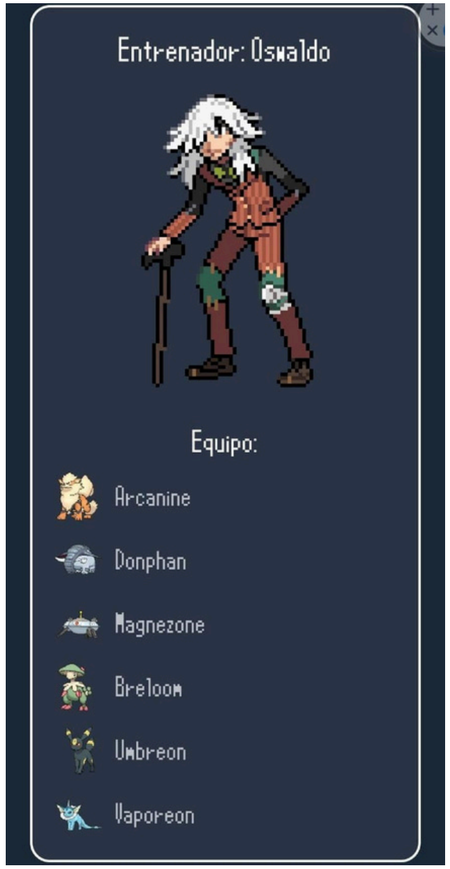

Bulbasaur
Nv.50
HP: 0/118
Lucario PSN
Nv.50
HP: 102/140
¡El equipo enemigo fue derrotado!
LUCHA
MOCHILA
POKÉMON

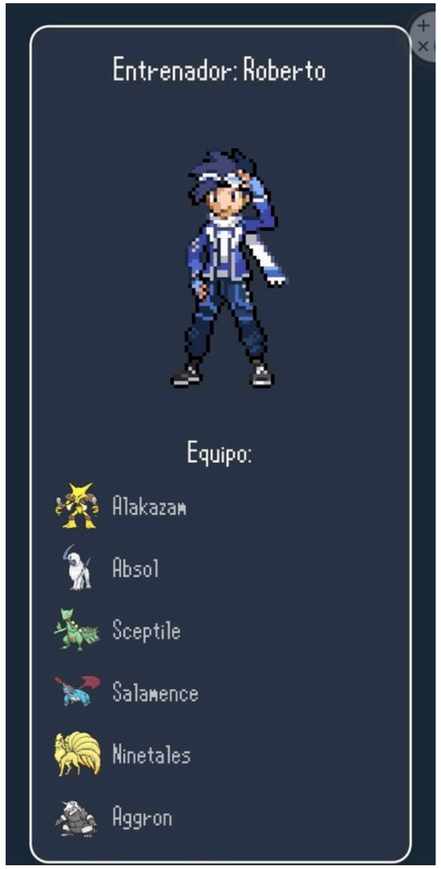

PAR Squirtle
Nv.50
HP: 60/108
Greninja FRZ
Nv.50
HP: 2/140
¿Qué debería hacer Greninja?
LUCHA
MOCHILA
POKÉMON


Dragonite
Nv.50
HP: 74/161
Lucario PAR
Nv.50
HP: 140/140
Usaste Antiparalizante.
LUCHA
MOCHILA
POKÉMON

Rivales


Entrenador: Alan
Equipo:
Lucario
Charizard
Dragonite
Gardevoir
Tyranitar
Gengar


Entrenador: Oswaldo
Equipo:
Arcanine
Donphan
Magnezone
Breloom
Umbreon
Vaporeon


Entrenador: Roberto
Equipo:
Alakazam
Absol
Sceptile
Salamence
Ninetales
Aggron


Entrenador: Abdiel
Equipo:
Hydreigon
Goodra
Tyranitar
Metagross
Dragonite


Entrenador: Andres
Equipo:
Gliscor
Metagross
Swampert
Togekiss
Ferrothorn
Infernape