Documentación
Pokemón Bytes

Documentación – Fase IV

Índice

1. Introducción y objetivos de la Fase

La Fase IV implementa el cierre del Core Loop del juego mediante la mecánica de captura. Desde el punto de vista arquitectónico, este módulo no crea nuevas entidades, sino que gestiona la transición de estado y propiedad de una entidad PokemonUsuario existente. Transforma una instancia "salvaje" (cuyo usuarioId es nulo o del sistema) en una instancia "propia" (vinculada al idUsuario del jugador), consumiendo recursos del módulo de economía de manera atómica.

2. Lógica Matemática de Captura (Gen II)

2.1. Fórmula Implementada (*valor a*)

La probabilidad de captura no es aleatoria, sino que sigue una implementación determinista de la fórmula oficial de la Generación II (Oro/Plata). Se ha programado un algoritmo en CalculoService que pondera cuatro variables críticas.

En el método calcularCaptura, se calcula un valor umbral *a* (variable a en el código). Si un número aleatorio generado entre 0 y 255 es menor que *a*, la captura es exitosa.

$$a = \frac{\left(\left(3 * HP\,max - 2 * HP\,actual\right) x\, rate\, x\, ball\right)}{\left(3 * HP\,max\right)} x\, estado$$

```
double numerador = (3 * hpMax - 2.0 * hpActual) * captureRate * bonoBall;

double denominador = 3.0 * hpMax;

double a = numerador / denominador;
```

2.2. Variables: Ratio, Ball y Status

Factor de Salud (HP):

- La fórmula (3 * hpMax - 2 * hpActual) penaliza la salud restante.
- *Análisis:* Un Pokémon con el 100% de vida reduce el numerador significativamente. Cuanto más cercano a 0 sea hpActual, más se acerca el multiplicador a 3.

Ratio de Especie (captureRate):

- Obtenido de PokedexMaestra (cargado vía PokeAPI).
- Rango: 0 a 255.
- *Código:* datosMaestros.getRatioCaptura() en BatallaService.

Bonificación de Poké Ball (bonoBall):

- Se ha implementado una lógica de control (switch/if) en BatallaService antes de llamar al cálculo:
    - Super Ball: Multiplicador **x1.5**.
    - Ultra Ball: Multiplicador **x2.0**.
    - Master Ball: Se asigna **255.0** directamente, garantizando que $a \ge 255$ (Captura 100%).

Multiplicador de Estado (Estado):

- Se aplica un bonus final al valor *a* dependiendo del Enum Estado:
- **x2.0:** Si el estado es DORMIDO o CONGELADO.
- **x1.5:** Si el estado es PARALIZADO, QUEMADO o ENVENENADO

3. Arquitectura Transaccional (BatallaService.java)

El método intentarCaptura es una operación que modifica dos tablas (INVENTARIO_USUARIO y POKEMON_USUARIO). Para garantizar la integridad, se ha optado con @Transactional.

3.1. Flujo de Ejecución

El servicio ejecuta la siguiente lógica secuencial:

1. **Inyección de Dependencias y Recuperación:**
    - Utiliza UserRepository y PokemonUsuarioRepository para cargar el contexto.
    - *Validación:* Si el Pokémon salvaje no existe (findById), lanza RuntimeException.
2. **Gestión de Inventario (Pre-Cálculo):**
    - Antes de calcular la probabilidad, se verifica el stock.
    - **Consulta:** inventarioRepository.findByUsuarioAndItem(usuario, ball).
    - **Regla de Negocio:** Si cantidad <= 0, se aborta la operación.
    - **Consumo Atómico:**

      ```
      inventario.setCantidad(inventario.getCantidad() - 1);

      inventarioRepository.save(inventario);
      ```

    Al estar dentro de una transacción, si la captura falla por error de sistema (ej. caída de BD), el ítem se devuelve automáticamente (Rollback).

3. **Persistencia Dinámica (El "Momento Captura"):**
    - Si calculoService.calcularCaptura() devuelve true, se ejecuta la lógica de cambio de propiedad.
    - Mutación de Entidad JPA:

      salvaje.setUsuarioId(usuario.getIdUsuario()); // Asignación de FK
      salvaje.setPosicionEquipo(2); // Lógica de hueco libre
      pokemonUsuarioRepository.save(salvaje);

    - Esto actualiza la columna id_usuario en MySQL, moviendo efectivamente el registro del "sistema" al "jugador".

4. Implementación en Código (Backend)

4.1. Algoritmo Matemático (CalculoService)

```
// Implementación de la fórmula de Gen II
public boolean calcularCaptura(int hpMax, int hpActual, int captureRate,
double bonoBall, Estado estado){
    // 1. Cálculo base
    double numerador = (3 * hpMax - 2.0 * hpActual) * captureRate *
bonoBall;
    double a = numerador / (3.0 * hpMax);

    // 2. Bonus de Estado
    if(estado == Estado.DORMIDO || estado == Estado.CONGELADO) a *= 2.0;
    else if (estado == Estado.PARALIZADO || estado == Estado.QUEMADO) a
*= 1.5;

    // 3. Check Captura Asegurada (Masterball o Ratio alto)
    if(a >= 255) return true;

    // 4. Probabilidad Estocástica
    int random = ThreadLocalRandom.current().nextInt(0, 256);
    return random < a;
}
```

4.2. Transacción de Captura (BatallaService)

```
@Transactional
public String intentarCaptura(String username, CapturaRequest request){
    // ... Carga de entidades omitida ...

    // Consumo de Ítem
    if(inventario.getCantidad() <= 0) throw new RuntimeException("Sin stock");
    inventario.setCantidad(inventario.getCantidad() - 1);
    inventarioRepository.save(inventario);

    // Determinación de Bono Ball
    double bonoBall = 1.0;
    if(ball.getNombre().equalsIgnoreCase("Ultra Ball")) bonoBall = 2.0;
    // ...

    // Ejecución
    if(calculoService.calcularCaptura(...)){
        salvaje.setUsuarioId(usuario.getIdUsuario()); // CAMBIO DE PROPIETARIO
        pokemonUsuarioRepository.save(salvaje);
        return "¡Capturado!";
    }
    return "Se escapó";
}
```

5. Diagramas y Esquemas Visuales

5.1. Diagrama de Secuencia (Captura)