Dadi

Stefano Rodighiero

November 5, 2016

Contents

1 Scenario

In un gioco di ruolo i giocatori costruiscono colletivamente una storia. Nella maggioranza dei casi il meccanismo di gioco prevede la presenza di un Narratore (chiamato alternativamente Master e talvolta Arbitro) che descrive la situazione agli altri giocatori, interpreta i personaggi non giocanti (antagonisti e in generale tutti i personaggi non controllati dai giocatori) e applica il regolamento per risolvere i conflitti che avvengono nel gioco. Gli altri Giocatori interpretano ciascuno un Personaggio, reagendo alle situazioni proposte dal Narratore e facendo proseguire la storia con le loro azioni.

Esistono molteplici sistemi (regolamenti) che guidano le decisioni dei giocatori e descrivono come risolvere i conflitti che capitano durante il gioco. È comune usare un elemento aleatorio (ad esempio il lancio di un dado) per simulare la parziale imprevedibilità dell'esito delle azioni.

Nel gioco tradizionale i giocatori si riuniscono nello stesso luogo e la risoluzione dei conflitti è demandata a qualche genere di processo fisico verificabile da tutti: lancio di un dado, estrazione di una carta o cose simili. Per il nostro esercizio, invece, immaginiamo un ambiente di gioco distribuito geograficamente: i giocatori ricorrono alla rete per giocare insieme, e hanno bisogno di un sistema centralizzato e imparziale per regolare i conflitti. Lo scopo dell'esercizio è realizzare tale sistema.

Tale sistema prevede:

- Creazione di check da parte del Narratore. Un check è definito da una probabilità di riuscita e da una descrizione.
- Consultazione dell'elenco dei check, aperti o già risolti.
- Provare a risolvere un check. Tale operazione modifica lo stato del check. Successive consultazioni di questo check mostrano la probabilità di riuscita definita, e l'esito della prova avvenuta.

Per rendere l'idea, ecco la trascrizione di una ipotetica sessione di gioco che utilizza questo sistema.

Dario: Siete in una radura. A Nord c'è una parete di roccia. Tra i rampicanti che la ricoprono riconoscete chiaramente il contorno di una porta di ferro.

Dario è il narratore.

Stefano: Il mio personaggio prova ad aprire la porta spingendola.

Stefano è un giocatore.

Dario: Va bene. Il tuo personaggio però non è molto forte, e la porta sembra molto pesante, sarà difficile.

Dario accede al servizio web per definire un check. Stabilisce che la probabilità di riuscita è pari a 20%. Ottiene in cambio la URL del check appena creato.

Dario: Stefano, ecco la URL del check che devi passare.

Stefano punta il suo browser sulla URL ricevuta. Trova una descrizione dell'azione che vuole provare, e la probabilità di riuscita stabilita da Dario. Cè un pulsante per tentare il check. Purtroppo l'esito è negativo.

Stefano: Purtroppo non ci sono riuscito!

Dario ricarica la URL che ha passato a Stefano. Poichè il check è già stato risolto, vede solo l'esito della prova.

2 Il progetto

Il progetto proposto è composto da una API REST e da una interfaccia Web per creare, consultare e risolvere check. Propongo di usare il framework Servant per costruire il backend.

Serve un layer di persistenza per memorizzare i dati che definiscono i check.

Non ho le idee chiare a proposito della realizzazione dell'interfaccia web, ma probabilmente basta un sistema di template che vada bene accoppiato a Servant.

3 Specifiche

3.1 Entità e azioni dell'interfaccia REST

- /check/new
  Una chiamata GET restituisce un form dove inserire i parametri che definiscono il check.

  Una chiamata POST crea un nuovo check e restituisce un identificatore per il check appena creato, o un errore se i parametri forniti sono incompleti o malformati. Il payload della chiamata POST è un documento JSON che contiene i parametri di creazione del check: la descrizione è una stringa di testo arbitraria, la probabilità di riuscita è un valore numerico.

```
{
  description: "Provi a spingere la porta di ferro."
  prob: 20
}
```

- /check/:id
  Una chiamata GET restituisce una pagina con la descrizione completa del check e un pulsante per provare a passarlo.

  Una chiamata PUT aggiorna lo stato del check, generando un valore random e confrontandolo con la probabilità di riuscita dell'azione. Una chiamata PUT su un check già risolto non ha alcun effetto.

- /checks
  Restituisce la lista di tutti i check, ordinati per stato e per data di creazione descrescente.