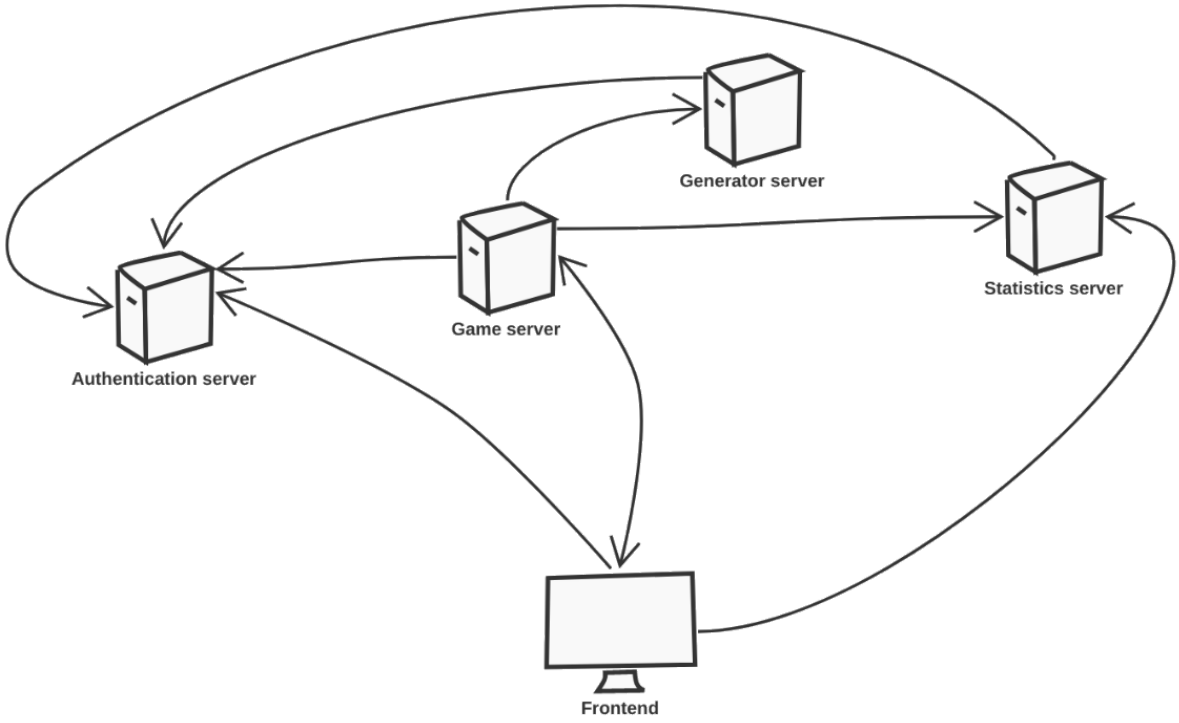
Advanced Programming Languages

Sudo-ku

Arena Gianluca - Lo Castro Alex



Introduzione

Sudo-ku è un sudoku multiplayer online in cui gli utenti hanno la possibilità di giocare secondo due modalità, challenge e collaborative. La modalità challenge è quella classica, due avversari si sfidano per risolvere lo stesso sudoku nel minor tempo possibile e al termine un giocatore risulterà vincitore e l'altro sconfitto. La modalità collaborativa è quella più interessante, due giocatori collaborano per risolvere lo stesso sudoku contemporaneamente e dunque vedendo in tempo reale le mosse dell'alleato. È possibile scegliere la difficoltà della board generata per ogni partita. Al termine di ogni partita viene generato un grafico che mostra il completamento del sudoku nel tempo ed è possibile visualizzare le statistiche relative a partite vinte, perse e tempo medio impiegato per una partita.

Architettura dell'applicazione

Come si evince dalla figura iniziale, i componenti logici della nostra architettura sono cinque: authentication server (AS), game server (GS), generator server (GenS), statistics server(SS) e frontend(FE).

L'AS fornisce l'autenticazione sia al FE, per loggarsi nel proprio account, sia ai server che possono contattarsi tra loro. Il FE contatta l'AS al momento del login e memorizza il token restituito per poi utilizzarlo per richiedere un nuovo match al GS e per ottenere le proprie statistiche dallo SS. Il GS utilizza il token ricevuto dall'AS per richiedere una nuova board al GenS e per inviare i dati allo SS alla fine di ogni partita.

Il GS è il responsabile del matching tra i giocatori e della gestione dello stato della partita. Dopo aver verificato l'autenticità del giocatore, lo mette in coda in base alla difficoltà e alla modalità scelta. Quando viene trovato un avversario/compagno compatibile, viene generata una nuova partita e i due giocatori possono iniziare a risolvere la board. Al termine della partita, i risultati vengono inviati allo SS e i giocatori vengono informati dell'esito.

Il GenS si occupa della generazione delle board, che fornisce al GS in base alla difficoltà richiesta.

Lo SS si occupa di memorizzare i dati dei match tra giocatori e fornirli al FE quando richiesti.

Il FE è il componente con cui si interfaccia il giocatore ed è il responsabile del login del giocatore, della gestione della partita lato client e di mostrare le statistiche al termine del match o successivamente.

Implementazione dei componenti

A fianco del nome del componente viene indicato chi se ne è occupato principalmente.

Authentication server (Lo Castro)

L'AS è stato implementato in Python come un web server Flask che espone tre route, "/register", "/login" e "/servers/auth". La prima è responsabile della registrazione dei

giocatori, che vengono salvati su un database MongoDB. La seconda è responsabile del login dei giocatori attraverso la generazione di un token che viene restituito. La struttura del token assomiglia a quella di JWT ma è stata customizzata e alleggerita per adattarsi meglio alla nostra applicazione. Sono state quindi implementate le funzioni per la verifica e la decodifica dei token sia in Python che in Go. La terza route serve per l'autenticazione dei server e viene restituito sempre un token che però utilizza un secret diverso da quello degli utenti.

Game server (Arena)

Il GS è l'elemento centrale dell'architettura, insieme al frontend. È implementato in Go come un server TCP. Alla connessione con un nuovo client, viene verificato che il token sia valido e la richiesta viene inoltrata alla goroutine matchServer. Questa si occupa di effettuare il matching tra giocatori, in base alla difficoltà e alla modalità scelta con politica FIFO, e una volta trovato il match vengono passate le informazioni dei due client accoppiati alla goroutine gameServerChallenge o gameServerCollaborative. Queste generano due goroutine per ogni client, una di lettura e l'altra di scrittura sulla socket, inviano ai due client la board generata del GenS e l'username dell'avversario e iniziano il match. Alla conclusione del match, le goroutine vengono terminate e sia SS che client vengono notificati. Il formato dello scambio di messaggi tra FE e GS è descritto nella sezione finale di questo documento.

Generator server (Arena)

Il GenS è implementato in Go e si occupa di fornire le board al GS attraverso la route "/board/difficulty={difficulty}". La generazione delle board non avviene in maniera sincrona, ovvero al momento della richiesta, ma viene effettuata offline. Per semplificare la valutazione del progetto le board vengono generate all'avvio del GenS ma in un deployment reale il refresh delle board verrebbe schedulato in maniera separata dal GenS, ad esempio come un cron job. La libreria per la generazione dei sudoku è stata implementata da zero in Go. L'elemento centrale della libreria è la funzione countSolutions, che restituisce il numero di soluzioni di una board, e makeFilledBoard che risolve la board in maniera randomizzata. Partendo da una board vuota e applicando makeFilledBoard viene generata una board risolta che rispetta le regole del sudoku by design e che cambia a ogni esecuzione per la randomness della funzione. Quindi la generazione della board con cui giocare viene realizzata eliminando un numero alla volta e verificando che la board

abbia ancora soluzione univoca (usando countSolutions). Una volta raggiunta la difficoltà richiesta, abbiamo ottenuto la board che cercavamo. Si noti che la difficoltà viene valutata in base ai numeri rimasti sulla board. Se volessimo una valutazione accurata della difficoltà in base a criteri umani dovremmo sottoporla a valutatori umani o a sistemi che simulino la risoluzione della board con metodi umani. Questo sistema sarebbe decisamente più complesso ed esula dallo scopo dell'elaborato e dunque ci atterremo al numero di valori rimasti sulla griglia come indicatore (in genere non eccessivamente inaccurato) della difficoltà.

Statistics server (Lo Castro)

Lo SS è implementato in Go come un web server e si occupa di registrare i risultati delle partite e di fornire le statistiche ai giocatori. Espone le routes "/user/{username}/result", "/stats" e "/ranking". La prima è utilizzata dal GS e serve a inserire il risultato di un match appena concluso. Il JSON postato comprende l'esito, il tempo impiegato e la data. La seconda e la terza sono utilizzate dal FE per ottenere, rispettivamente, il numero di match vinti/persi e il tempo medio impiegato e il ranking generale in base al numero di vittorie.

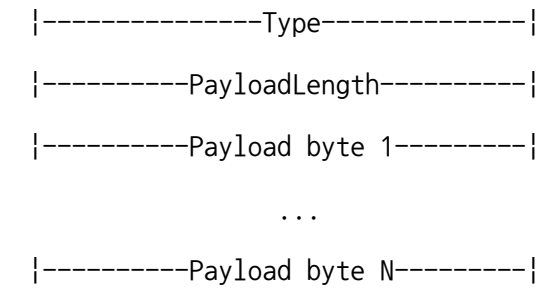
Frontend (Lo Castro-Arena)

Il frontend è stato implementato in Python utilizzando PyQt5 come libreria grafica. Dal punto di vista della logica applicativa, nella modalità challenge il client mantiene le mosse localmente e invia un messaggio al server solo quando la board è completa, nella versione collaborativa ogni mossa viene inviata al server per mantenere allineate le board dei due giocatori. Al termine di ogni match viene stampato, utilizzando Matplotlib, il grafico tempo-caselle riempite per avere un'idea dell'andamento del match nel tempo. La comunicazione con AS e SS avviene attraverso REST API mentre col GS attraverso socket. Il formato dei messaggi scambiati è specificato nella sezione successiva. Sono stati gestiti i casi in cui il server va in fault o l'altro giocatore abbandona la partita.

Comunicazione

La comunicazione tra quasi tutti i componenti dell'architettura avviene attraverso REST API (HTTP + JSON), l'unica eccezione è l'interazione tra FE e GS durante il match per cui sono state utilizzate socket TCP per ottenere performance migliori e comunicazione full duplex.

La struttura dei messaggi scambiati da FE e GS è la seguente:



Il campo “type” (1 byte) indica il tipo di pacchetto scambiato e può assumere uno dei seguenti valori:

Codifica	Tipo pacchetto
1	matchRequest
2	matchFound
3	checkSolution
4	validSolution
5	done
6	move
7	changeValue
8	ping
9	pingAck
10	error

Il byte seguente indica la dimensione del resto del payload. Il payload è in formato JSON e il parsing dipende dal valore del tipo del pacchetto.