Progetto Di Laboratorio di Sistemi Operativi 2024/2025
Applicazione Client-Server che Gestisce Aree di Gioco del Tris

NOME GRUPPO: TicTacTeam

Murano Monte Liberti
Stefano Cristian Simone
N86004383 N86004326 N86004252

MAGGIO 2025

Indice

Capitolo 1

Analisi del Progetto

1.1 Requisiti Richiesti Dal Cliente

Lo studente dovrà realizzare un server multi-client per giocare a Tris (con due giocatori per ogni partita). Un giocatore può creare una o più partite, ma può giocare solo ad una partita alla volta. Il creatore di una partita può accettare o rifiutare la richiesta di partecipazione alla partita da un nuovo giocatore. Gli stati di gioco possono essere terminata, in corso, in attesa, nuova creazione. Gli stati di terminazione di partita possono essere vittoria, sconfitta, pareggio rispetto al giocatore. In base allo stato di gioco di ogni singola partita, tutti i giocatori collegati al server dovranno ricevere un messaggio diverso. Per esempio, "in attesa" tutti i giocatori vengono invitati a partecipare alla partita. Le partite devono essere identificate in maniera univoca. A fine partita (terminata), i giocatori di ogni partita possono scegliere se iniziare o meno un'altra partita.
Opzionale: Il vincitore di una partita può decidere se fare un'altra partita - in questo caso, se non era il proprietario, diventa il proprietario della partita e attende un nuovo giocatore. Il perdente deve lasciare la partita. Se c'è pareggio, entrambi i giocatori possono decidere se farne un'altra.

1.2 Requisiti Individuati

Segmento	Requisito
Lo studente dovrà realizzare un server multi-cliente	Architettura client-server multiclient
Un giocatore può creare una o più partite,	Possibilità da parte di ogni giocatore di creare una o più partite
ma può giocare solo ad una partita alla volta	Possibilità da parte di ogni giocatore di partecipare ad un'unica partita per volta
Il creatore di una partita può accettare o rifiutare la richiesta di partecipazione alla partita da un nuovo giocatore	Possibilità di un giocatore di accettare o rifiutare richieste di partecipazione a partite da lui create
Gli stati di gioco possono essere terminata, in corso, in attesa, nuova creazione	Ogni partita deve essere dotata di uno stato di gioco che può essere: "nuova creazione", "in corso", "in attesa", "terminata"
Gli stati di terminazione di partita possono essere vittoria, sconfitta, pareggio rispetto al giocatore	Quando una partita entra nello stato di gioco "terminata" deve essere specificato lo stato di terminazione: "vittoria del giocatore..." - "sconfitta del giocatore...", "pareggio"
In base allo stato di gioco di ogni singola partita, tutti i giocatori collegati al server dovranno ricevere un messaggio diverso. Per esempio, "in attesa" tutti i giocatori vengono invitati a partecipare alla partita	Ogni partita è affiancata ad un messaggio relativo al proprio stato di gioco che il server invia ad ogni client collegato
Il vincitore di una partita può decidere se fare un'altra partita - in questo caso, se non era il proprietario, diventa il proprietario della partita e attende un nuovo giocatore	Il vincitore di una partita ne diventa il proprietario e può accettare o rifiutare richieste di accesso da parte di altri giocatori
Il perdente deve lasciare la partita	Il perdente viene rimosso dalla partita e, se era il proprietario, ne perde il possesso
Se c'è pareggio, entrambi i giocatori possono decidere se farne un'altra.	In caso di pareggio, i giocatori possono decidere se rimanere e ricominciare la partita o uscire

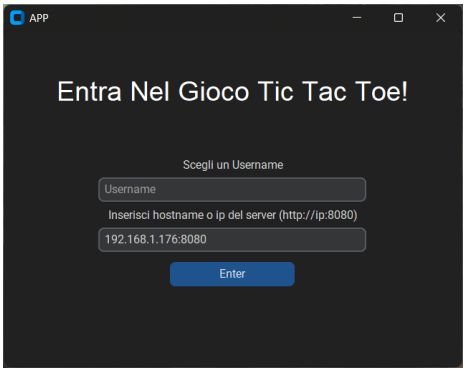
1.3 Risoluzione dei Requisiti

Collegamento al server

Per collegarsi al server, un client deve specificare l'indirizzo ip e porta del server e scegliere un username (non per forza univoco) che sarà visibile agli altri giocatori collegati

Per le comunicazioni con il server viene utilizzato esclusivamente l'indirizzo di socket.

Nel caso il server non sia raggiungibile verrà segnalato con un apposito popup



Gestione partite

Una volta connesso, il client riceve dal server l'elenco delle partite disponibili e può decidere se richiedere l'accesso alle partite di altri giocatori o crearne a sua volta.

Un giocatore può richiedere l'accesso solo alle partite il cui stato di gioco è "creazione".

Un giocatore può accettare o rifiutare le richieste di accesso alle proprie partite nella sezione notifiche. Qualora venisse accettata una richiesta da parte di un giocatore disconnesso o già in partita, verrà segnalato con un apposito popup.

Un giocatore può aggiornare l'elenco delle partite disponibili clickando sull'apposito pulsante chiamato Ricarica.

Quando un giocatore si disconnette vengono cancellate in automatico tutte le partite da lui create.

Gestione partite in corso

Ogni client può giocare al più una partita per volta

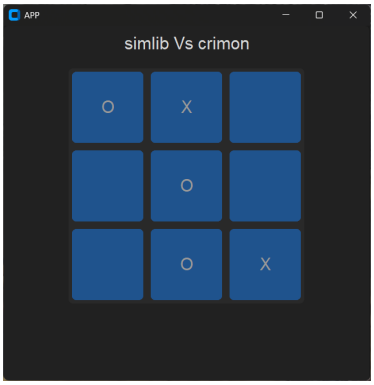
La partita si svolge in modo alternato tra i due giocatori. Per poter inserire una mossa ogni giocatore deve aspettare il proprio turno

In caso di vittoria il giocatore diventa il nuovo proprietario della partita

In caso di sconfitta il giocatore viene rimosso dalla partita

In caso di pareggio si aspettano le decisioni dei due giocatori, se entrambi accettano la partita ricomincia, se entrambi rifiutano la partita viene cancellata, se un solo giocatore accetta vengono applicati i casi vittoria-sconfitta

Se un giocatore si disconnette dalla partita, l'avversario vince automaticamente la partita



Capitolo 2

Scelte Implementative

La scelte implementative dell'applicazione Client-Server, vertono sull'utilizzo di un Server in linguaggio C, rispettando il vincolo imposto dal cliente che ha commissionato tale applicazione, la scelta per il client, verte sul linguaggio Python, che fà parte dei linguaggi di scripting, tale scelta è stata ponderata basandoci su quelle che sono le peculiarità di tale linguaggio, sotto diversi punti come:

1. **Semplicità e leggibilità**: Python è noto per la sua sintassi chiara e leggibile, il che rende più facile scrivere e mantenere il codice. Questo è particolarmente utile quando si lavora in team o si devono apportare modifiche nel tempo.

2. **Ampia libreria di moduli**: Python offre una vasta gamma di librerie e framework, come Flask e Django per il server, e Requests per il client. Questi strumenti possono semplificare notevolmente lo sviluppo e ridurre il tempo necessario per implementare funzionalità comuni.

3. **Supporto per la programmazione asincrona**: Con librerie come asyncio, Python consente di gestire operazioni di rete in modo efficiente, migliorando le prestazioni delle applicazioni Client-Server, specialmente quando si tratta di gestire molte connessioni simultanee.

4. **Portabilità**: Python è un linguaggio multipiattaforma, il che significa che le applicazioni possono essere eseguite su diversi sistemi operativi senza modifiche significative al codice.

5. **Comunità attiva**: La comunità di Python è molto attiva e offre un ampio supporto. Ci sono molte risorse, tutorial e forum dove puoi trovare aiuto e condividere esperienze.

6. **Integrazione con altre tecnologie**: Python si integra bene con altre tecnologie e linguaggi, il che facilita l'interazione con database, API e servizi esterni.

7. **Rapidità di sviluppo**: Grazie alla sua sintassi semplice e alle librerie disponibili, Python consente uno sviluppo rapido, il che è ideale per prototipazione e iterazione veloce.

Per gestire la comunicazione tra client e server è stato sfruttato il formato Json, data la sua versatilità e portabilità.
Il server è stato configurato per essere eseguito all'interno di un ambiente Docker, includendo tutti i requisiti necessari nel relativo Dockerfile.
Il client è stato sviluppato utilizzando Python e distribuito sotto forma di eseguibile nativo, eliminando la necessità di ambiente di runtime esterni.

Capitolo 3

Funzionamento

3.1 Organizzazione dei File dell'Applicazione

Tale applicazione client Server è organizzata nel seguente modo:

1. Package Server:
   (a) file main.c: File contenente la struttura del server.
   (b) file handler.c : File contenente le funzionalità di connessione, gestione e disconnessione dei client.
   (c) file logic.c : File contenente la logica per la gestione delle partita.
   (d) file utils.c : File contenente le funzionalità di costruzione dei messaggi.
   (e) file Makefile : File che genera i files con estensione .o ed il file eseguibile.
   (f) Package include:
      i. file buffer.h : File contenente le strutture dei buffer.
      ii. file common.h : File contenente le strutture dati utilizzate.
      iii. file messaging.h : File contenente le tipologie di messaggi.
   (g) Package lib : File di libreria JSON per la conversione da buffer-JSON e JSON-buffer.
   (h) Package build: Package contenente i files generati dal Makefile.
2. Package Client:
   (a) file main.py : File contenente il template delle interfacce e della finestra principale.
   (b) file models.py : File contenente i dizionari e le strutture dati utilizzate.
   (c) file utils.py : File contenente le funzioni per la gestione del Client.

(d) Package core:

i. file buffers.py :File contenente le strutture dei buffer.

ii. file connection.py : File contenente le funzioni per la connesione con il Server.

iii. file controller.py : File contenente le funzioni per la comunicazione con il server.

iv. file globals.py : File contenente le variabili globali.

(e) Package gui:

i. Package frames:

A. file loginframe.py : File contenente il LoginFrame.

B. file homeframe.py: File contenente l'HomeFrame.

C. file matchframe.py : File contenente il MatchFrame.

ii. file app.py: File contenente la logica di cambio frame.

iii. file Popups.py : File contenente i popups.

3. file .gitignore : Per riformattare i file.

4. file Dockerfile : Per rendere l'applicazione Client-Server eseguibile da diversi Sistemi Operativi.

3.2 Descrizione Comunicazione Client e Server

Ogni client connesso è un thread e può comunicare esclusivamente con il server, non c'è comunicazione diretta tra i client. Tutti i messaggi sono marcati da un segnale che ne specifica la struttura ed il contenuto. Il server mette a disposizione un'area di memoria condivisa in cui salvare fino ad un massimo di 100 partite. Ogni volta che un client tenta di modificare una partita: blocca l'area di memoria che intende utilizzare, esegue le operazioni e sblocca l'area di memoria utilizzata.