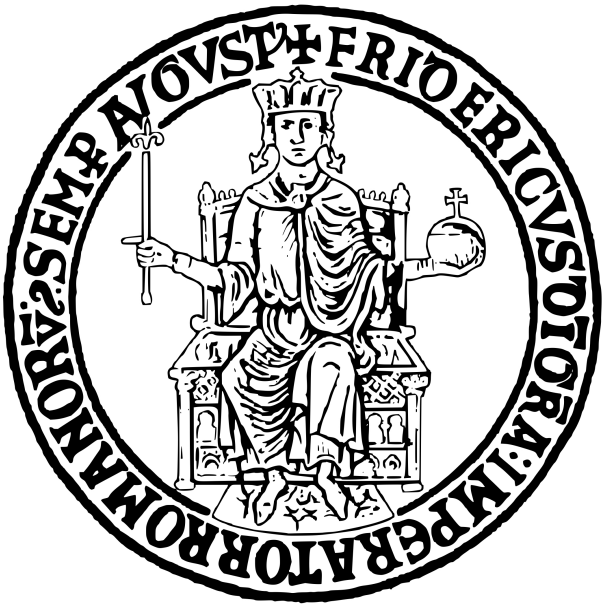
Università degli Studi di Napoli Federico II

Corso di Laurea in Informatica



Autori:

Giuliano Savino	Matricola: N86004523
Riccardo Regina	Matricola: N86004614
Francesco Rossetti	Matricola: N86004505

Anno Accademico: 2024/2025

Indice

1 Analisi dei requisiti

In questa sezione descriviamo i requisiti del progetto.

1.1 Obiettivo del progetto

Il progetto consiste nella simulazione di un sistema che riproduce il **Telefono Senza Fili** tra giocatori di lingue diverse. Gli utenti devono potersi registrare e accedere al server per iniziare a giocare. Successivamente possono entrare in stanze preesistenti o crearne di nuove, all'interno delle quali invieranno messaggi agli altri partecipanti. Quando nella stanza sono presenti almeno quattro giocatori, il creatore può scegliere la direzione del turno (oraria o antioraria) e inviare il primo messaggio. Il sistema tradurrà automaticamente il testo nella lingua del secondo giocatore, che potrà aggiungere una parola prima di inoltrarlo al terzo partecipante, e così via fino all'ultimo. Al termine del giro, il sistema mostrerà a tutti i giocatori l'evoluzione completa della frase, rivelando come il messaggio è cambiato lungo il percorso.

1.2 Analisi dei requisiti

Le principali funzionalità del sistema sono:

- **Registrazione e autenticazione degli utenti**: gli utenti devono potersi registrare e accedere al server per iniziare a giocare.
- **Creazione delle stanze**: tutti gli utenti possono creare stanze a cui altri giocatori possono unirsi.
- **Modalità oraria e antioraria**: il creatore di una stanza può scegliere la direzione del gioco, in senso orario o antiorario.
- **Traduzione delle frasi**: un giocatore in una stanza inserisce una frase che viene tradotta dal sistema e passata al giocatore successivo.
- **Gestione della coda**: un giocatore che entra in una stanza in cui una partita che è già iniziata viene inserito in coda e, al termine della partita, può partecipare al gioco.

2 Implementazione

2.1 Tecnologie utilizzate

Il progetto utilizza tre linguaggi principali :

- **C** : Il linguaggio C è utilizzato per lo sviluppo dell'applicazione client-server. Il server e il client comunicano tramite socket, permettendo lo scambio di dati in rete.
- **SQL**: il linguaggio SQL è utilizzato per la gestione del database **SQLite**. Abbiamo scelto SQLite per la sua semplicità architetturale e di gestione: ogni database è contenuto in un unico file sul filesystem, semplificando backup, copia e distribuzione.
- **Python**: per la realizzazione dell'interfaccia grafica è stato scelto il linguaggio Python, utilizzando in particolare la libreria standard **Tkinter**. Tale scelta è giustificata dalla semplicità di integrazione con il sistema di socket del server e dalla natura **event-driven** di Tkinter, che si basa su un loop principale dedicato all'ascolto di eventi e all'invocazione di callback associate.
- **Docker**: consente di creare ambienti virtualizzati leggeri e indipendenti. In particolare, poiché LibreTranslate fornisce un'immagine Docker ufficiale, abbiamo creato un file `docker-compose.yml` per:
  - scaricare e avviare l'immagine di LibreTranslate;
  - costruire e avviare l'immagine del nostro server, definita nel `Dockerfile`.

  Al termine del deploy, saranno attivi due container — uno per LibreTranslate e uno per il server — che comunicano tra loro sulla stessa rete virtuale Docker. Inoltre,l'immagine del server è stata fornita di uno script *wait-for-libretranslate.sh*, che permette al container di aspettare l'health check del container di LibreTranslate.
- **Git** : Abbiamo scelto Git come sistema di versionamento per la sua affidabilità e diffusione nell'industria. Grazie a Git, possiamo gestire il lavoro in team in modo collaborativo, creare branch per sperimentazioni o nuove funzionalità, e tracciare facilmente ogni modifica al codice. Inoltre, la sua compatibilità con piattaforme come GitHub e GitLab semplifica l'integrazione con il nostro flusso di lavoro

2.2 Struttura del codice

Il progetto è organizzato in una struttura di cartelle che separa logicamente i vari componenti del progetto.

Di seguito, è riportata la struttura gerarchica delle cartelle:

```
project_root
+-- client
|   +-- client.py
+-- server
|   +--translator.h
|   +--server.c
|   +--translator.c
|   +--Makefile
|   +--Dockerfile
|   +--docker-compose.yaml
|   +--wait-for-libretranslate.sh
```

3 Architettura del sistema

3.1 Componenti principali

Il sistema si compone da tre principali figure:

- **Server** : Scritto in `C`, utilizza le socket per la comunicazione TCP, gestisce lobby e partite e si integra con `LibreTranslate` via `HTTP` per la traduzione delle frasi.
- **Client**: Client GUI in Python basato su Tkinter per un'esperienza interattiva; include anche un client da riga di comando in C per il gioco nel terminale.
- **LibreTranslate**: Il server comunica con `LibreTranslate` tramite HTTP Requests per tradurre le frasi fra giocatori di diverse lingue durante il gioco.

3.2 Multithreading

Il server è completamente multithreaded. Per ogni nuova connessione client viene creato un thread dedicato utilizzando i POSIX threads. Questo permette al server di gestire più client in contemporanea, garantendo reattività e scalabilità. Le risorse condivise, come le tabelle globali dei giocatori e delle lobby, sono protette tramite mutex per prevenire race condition.Per garantire la sicurezza dei thread, attorno alle sezioni critiche — ad esempio la modifica della lista dei giocatori, delle lobby e l'invio di dati sui socket — vengono

utilizzati dei mutex. La socket di ogni giocatore è protetto da un mutex dedicato per evitare scritture concorrenti.

3.3 Comunicazione tramite socket

Il server utilizza socket TCP per una comunicazione affidabile. Ascolta su una porta configurabile e accetta le connessioni in arrivo dai client. Ogni client comunica con il server tramite un semplice protocollo testuale, in cui ogni messaggio inizia con un codice operazione seguito dai parametri richiesti.

3.4 Protocollo utilizzato

Codici di operazioni Client –> Server

Codice	Operazione	Formato / Parametri
201	Signup	`201 <lang> <username> <password>`
202	Login	`202 <username> <password>`
100	Create Lobby	`100`
101	Join Lobby	`101 <lobby_id>`
102	Get Lobbies	`102`
103	Leave Lobby	`103`
110	Start Match	`110 <direction> (1 for clockwise, 0 for counter)`
111	Speak (add word)	`111 <len> <word>`

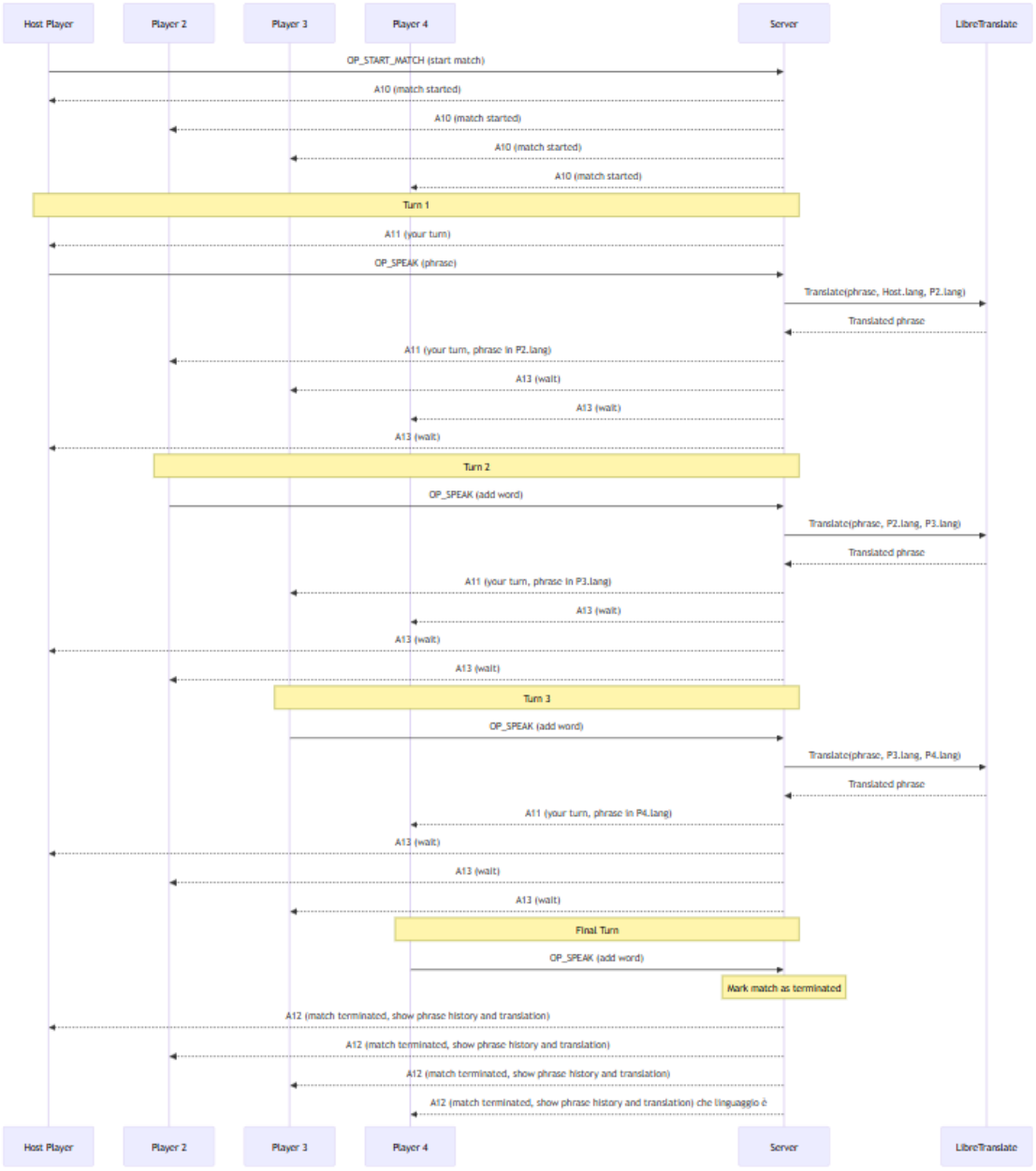
Tabella 1: Codici di operazione (Client → Server)

Codici di operazioni Server->Client

Codice	Significato	Descrizione
A00	Lobby Created	Lobby successfully created
A01	Lobby Joined	Successfully joined a lobby
A02	Host Left	Host left, lobby closed
A03	Player Left	Player left the lobby
A04	Player Enqueued	Lobby full, player added to queue
A05	Lobbies List	List of available lobbies
A06	Queue Left	Player left the queue
A07	Queue Joined	Match started, player added to queue
A08	Player Joined	A player joined the lobby
A10	Match Started	Match has started
A11	Your Turn	It's your turn
A12	Match Terminated	Match ended, phrase history shown
A13	Wait for Others	Wait for other players
B01	Signed Up	Signup successful
B02	Logged In	Login successful
Z00	Server Error	Internal server error
Z01	Bad Request	Invalid request format or parameters
Z02	Conflict	Username exists, already logged in, etc.
Z03	Unauthorized	Not authenticated or wrong credentials

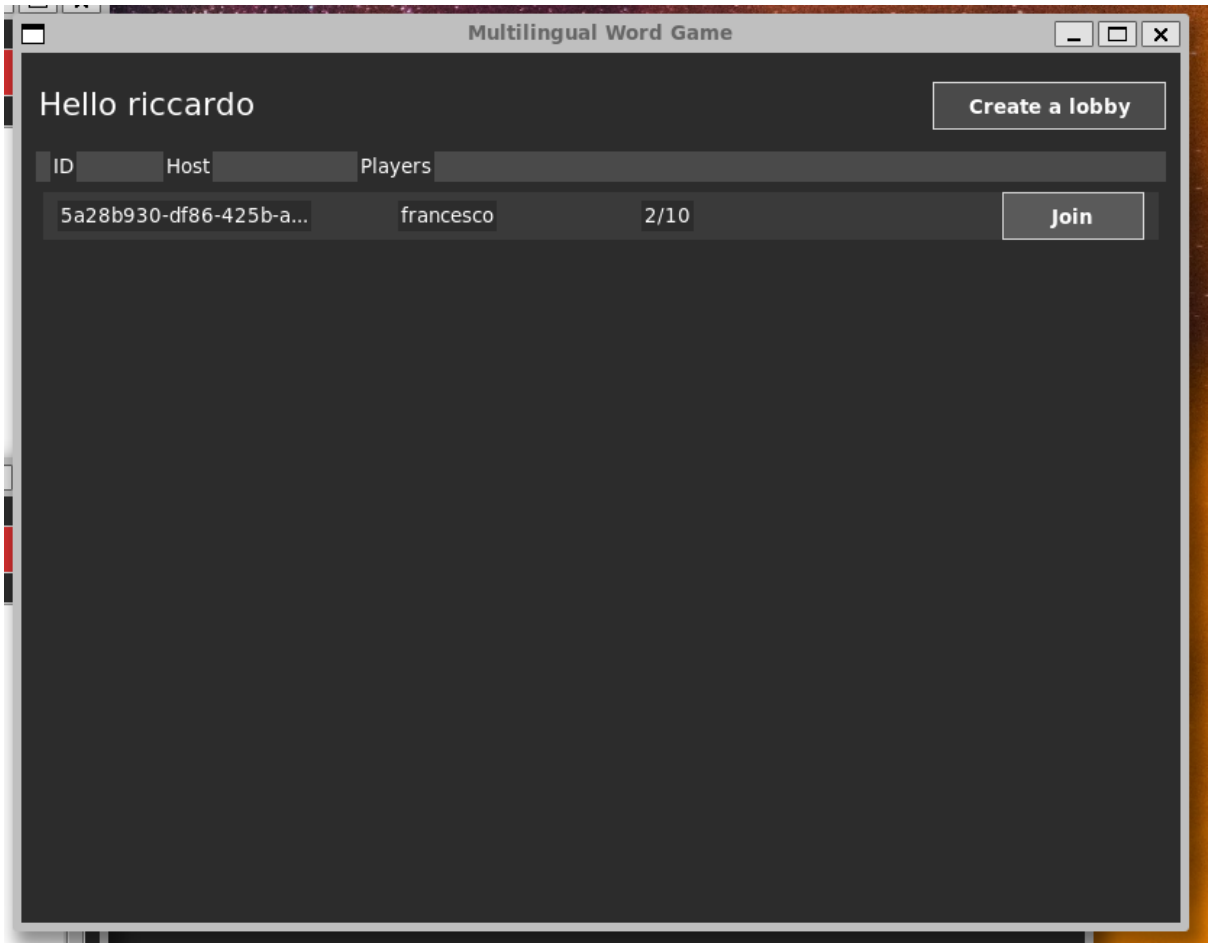
Tabella 2: Codici di operazione (Server → Client)

3.5 Funzionamento di una partita

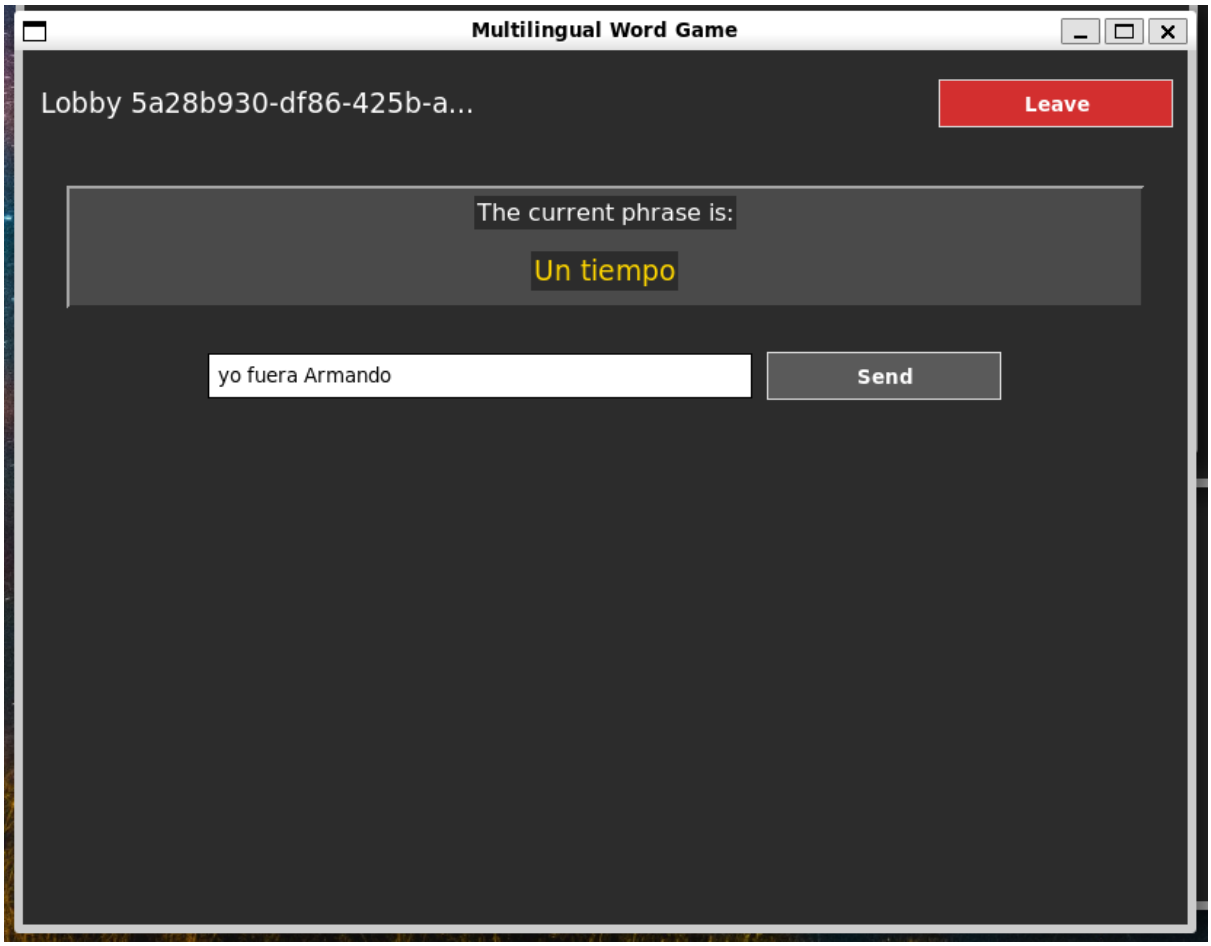


4 Interfaccia del sistema

4.1 Schermata accesso alla lobby



4.2 Schermata inserimento frase



4.3 Schermata visualizza frase finale