Immagine che contiene nero, oscurità

Descrizione generata automaticamente

PROGETTO “GUESS THE WORD”

LABORATORIO DI SISTEMI OPERATIVI

FEDERICO II - NAPOLI

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Prof.ssa:** | Alessandra Rossi |  |
|  |  |  |
| **Autori:** | Federico Gargiulo  Cristian Carrella  Gianmarco Lembo | N86002884  N86003677  N86003940 |
|  |  |  |

**INDICE**

[INTRODUZIONE 5](#_Toc136677171)

[FUNZIONAMENTO DELLE RICHIESTE 5](#_Toc136677172)

[FASE DI ACCETTAZIONE DEI NUOVI CLIENT 5](#_Toc136677174)

[GENERAZIONE DELLE PAROLE E DELLE DEFINIZIONI 6](#_Toc136677175)

[FUNZIONAMENTO DELLA LOBBY 7](#_Toc136677176)

[START DI UNA PARTITA 7](#_Toc136677178)

[FUNZIONAMENTO DI UNA PARTITA 8](#_Toc136677180)

[FINE DI UNA PARTITA 8](#_Toc136677182)

# INTRODUZIONE

“Guess the word” è un applicazione sviluppata - secondo le specifiche della traccia:

* Client : Android - Java/XML
* Server : C

Inoltre il server ha un collegamento ad un database relazionale PosgreSQL, mediante l’ausilio della libreria libpq, dove sono conservate le informazioni degli utenti che si registrano all’applicazione.

# FUNZIONAMENTO DELLE RICHIESTE

Affinché il server possa elaborare le richieste del client, quest'ultimo dovrà inviare al server le richieste in formato JSON. Le richieste verranno analizzate per determinare il tipo di operazione da eseguire. Il server leggerà il campo "operation" del JSON e, in base al suo contenuto, invocherà la corrispondente funzione che eseguirà l'operazione e restituirà un JSON appositamente creato.

Es: Client - invia { operation: login, email: email@example.com, password: pwd }

Server - capisce che l’operazione è di login allora chiama la funzione login(email, password) e successivamente costruisce il JSON di risposta { logged : true }

# Immagine che contiene schermata, cartone animato Descrizione generata automaticamente

OSS: Per l’utilizzo dei JSON lato server abbiamo utilizzato la libreria json-c

# 

# FASE DI ACCETTAZIONE DEI NUOVI CLIENT

Quando un utente si connette al server, al momento dell’accept(), quest’ultimo crea un thread che andrà a gestire tutte le richieste corrispondenti ad esso.

Immagine che contiene schermata, Policromia, linea

Descrizione generata automaticamente

# GENERAZIONE DELLE PAROLE E DELLE DEFINIZIONI

Le parole presenti nel gioco sono generate completamente in modo casuale facendo delle richieste a delle API che vengono fornite in modo gratuito:

* random-word-api.herokuapp.com: per generare la parola in modo casuale
* api.dictionaryapi.dev: da cui prendere le definizioni per far indovinare la parola

Immagine che contiene schermata, linea, cerchio, connettore

Descrizione generata automaticamente

# FUNZIONAMENTO DELLA LOBBY

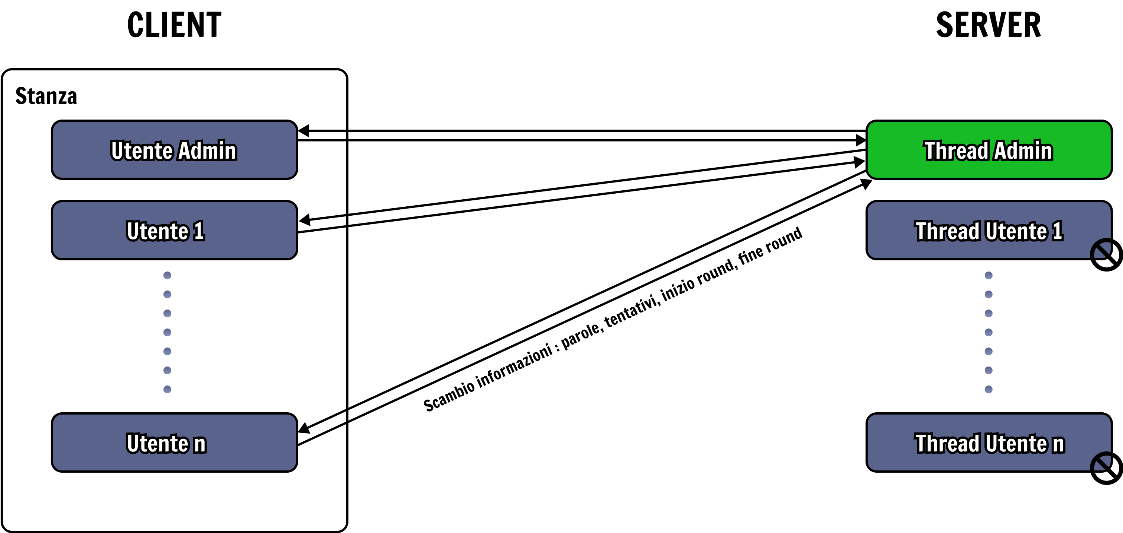
# 

# START DI UNA PARTITA

# 

# SVOLGIMENTO DEL GIOCO

Da questo momento la logica di gioco lato client viene affidata alla classe GameManager che estende la classe Thread e si occupa delle operazioni di rete e della modica dell’UI.



**Fase pre-round**

Il server designa il giocatore che dovrà scegliere la parola, dopodichè segnalerà ai client l’operazione da svolgere attraverso l’invio di una stringa:

* *WAIT:* il client si mette in attesa di ricevere la parola scelta dall’utente designato
* *CHOOSE:* il client si mette in attesa di ricevere un JSON contenente le parole tra cui l’utente potrà scegliere e le definizioni, nel seguente formato:  
   { “word0” : [“parola0”, “definizione0] }  
   …  
   { “word4” : [“parola4”, “definizione4] }
* *END:* la partita è finita e il client può procedere a rimuovere la stanza e tornare alla Home Activity. Il server provvederà a effettuare le opportune modifiche ai dati persistenti e non.

**Fase di svolgimento del round**

Una volta che tutti I client avranno ricevuto la parola scelta inizierà la fase di svolgimento del round. I client aspetteranno ciascuno il loro turno il cui inizio verrà segnalato dal server attraverso l’invio della stringa “*YOUR\_TURN*”. All’inizio del turno verrà attivato un timer di 15 secondi entro I quali l’utente potrà effettuare il proprio tentativo di indovinare la parola. Dopo che tutti gli utenti avranno tentato di indovinare la parola il server genererà in maniera randomica un aiuto che consiste nello sblocco di una lettera della parola. Informazioni aggiuntive potranno essere ricevute dal client durante l’attesa del turno:

* *NEW\_HINT:* il client si mette in attesa di ricevere un JSON contenente la lettera generata dal server e la posizione all’interno della parola:  
   *{“letter” : “lettera”, “position” : posizione}*
* *HINT\_END:* gli aiuti possibili sono terminati, il punto viene assegnato all’utente che ha scelto la parola e si ritorna alla fase pre-round.
* Una stringa in formato JSON contenente:
  + Guessed: valore booleano che indica se la parola è stata indovinata da un altro giocatore.
  + Word: stringa contenente la parola con cui ha provato ad indovinare un altro giocatore.
  + PlayerName: stringa contenente l’username del giocatore che ha provato ad indovinare.

# FINE DI UNA PARTITA

# 

# STRUTTURE UTILIZZATE LATO SERVER

## STRUTTURA UTENTE

Si occupa di gestire l’utente loggato.

## STRUTTURA STANZA

La struttura stanza si occupa di tenere traccia di tutti i dati relativi a tutte le stanze in gioco o in attesa dell’inizio.

Il server conserva all’interno di un’array di puntatori a stanze quelle create dagli utenti.

Per gestire gli accessi concorrenti alla struttura stanza si utilizzano **pthread\_mutex**.

# README

Come hostare il server sulla macchina attuale:

1. Installare Docker

2. Eseguire da bash "docker compose up" nella directory /Guess30L/server

Docker avvierà il server su porta 5001

Come avviare il client:

1. Aprire al path /Guess30L/Android il progetto con Android Studio

Avviare tramite Android studio l’applicazione che si collegerà al server locale avviato in precedenza