# Documentazione progetto di Reti Informatiche

## Funzionamento generale

Nell'Escape Room sono implementate due stanze. Al primo accesso di un account in una stanza viene avviata una nuova partita; in ogni stanza si può giocare solo una partita alla volta perciò tutti gli account collegati alla stessa stanza giocano la stessa partita in modalità cooperativa, condividendo token, tempo, oggetti raccolti e così via.

Un giocatore può sbloccare un oggetto usando altri oggetti in un determinato ordine oppure risolvendo un enigma con risposta aperta oppure risolvendo un enigma a risposta multipla con numero limitato di tentativi.

Anche la vittoria (la raccolta di tutti i token) o la sconfitta (per esaurimento del tempo o per risposta sbagliata a una domanda a tentativi) sono comuni a tutti gli account nella stessa stanza. Quando un account si accorge che la partita è terminata avvisa tutti gli altri account collegati alla stessa stanza con l'esito della partita. Una partita può terminare anche perché tutti gli account sono usciti dalla stanza.

Dal punto di vista della scalabilità, per evitare che troppi giocatori si intralcino nella stessa stanza, si può pensare a creare vari gruppi di giocatori che giocano partite diverse nella stessa stanza, ma questa dinamica non è stata affrontata nel progetto.

La funzionalità a piacere consiste in un blocco note condiviso in cui gli utenti possono leggere e annotare idee, osservazioni, cose da fare; nel blocco note c'è un limite massimo di note che si possono scrivere, perciò le note sono gestite in modalità FIFO così da preservare solo le più recenti.

## Implementazione

L'Escape Room è implementata in modo che sia facilmente possibile aggiungere nuove stanze e comandi oppure modificare i parametri di funzionamento e di scambio dei messaggi. Le costanti e i parametri dell'Escape Room sono definite in `include.h`. Il file `strutture.h` definisce le strutture `Account`, `Locazione`, `Enigma`, `Oggetto` e `Partita` per rappresentare gli scenari di gioco.

Il server è basato su I/O multiplexing. Questo permette di gestire facilmente gli input da `stdin`, le nuove connessioni al socket di ascolto e i numerosi scambi di dati ai socket di comunicazione, limitando l'overhead e permettendo la condivisione delle stanze. Le interazioni client-server avvengono tramite socket TCP perché è preferibile garantire l'affidabilità delle comunicazioni rispetto alla velocità.

Il server attiva il socket di ascolto con il comando `start`. Il client tenta ripetutamente di connettersi al server poi si registra e accede con i comandi `signup` e `login`. Il server gestisce gli account dinamicamente tramite la lista `accounts` e associa socket e account tramite l'array `socketToAccount`. Dopodiché il client seleziona una room e il server inizializza la partita in quella room modificando l'opportuna entrata dell'array `stanze`.

A questo punto inizia la partita vera e propria. In generale, l'esecuzione dei comandi ha uno schema fisso:

1. Il client invia il comando
2. Il server invia lo stato della partita

3. Se la partita non è terminata il client invia l'eventuale argomento
4. Il server esegue il comando
5. Il client riceve l'esito del comando e le informazioni sulla partita
6. Il client stampa il risultato relativo all'esito del comando e le informazioni

In questo modo il client è costantemente informato sullo stato della partita e non si verificano inconsistenze. Per risparmiare traffico, fa eccezione il comando `look` senza argomenti, che viene gestito lato client, essendo le descrizioni delle stanze informazioni statiche. Anche il parsing dei comandi è gestito lato cliente nella maggior parte dei casi l'esito di un comando è comunicato numericamente, così che la gestione di lunghe stringhe di testo possa avvenire lato client. Il ciclo di comandi del client si interrompe quando la partita finisce o quando il giocatore esce con il comando `end` o con `Ctrl+C`; il server capta la disconnessione del client e aggiorna opportunamente le strutture dati.

La funzionalità a piacere è implementata tramite il comando `notes [add]`. Se il parametro `add` è assente, il server concatena le note in un'unica stringa e le invia al client; il client riceve dal server la stringa e stampa le note. Se è presente, il client scrive la nuova nota e la invia al server, il quale la aggiunge al blocco note della partita (una coda circolare di note).

**Scambio dei dati**
L'applicazione non definisce uno standard di comunicazione dei dati unico data la variabilità degli scambi.

Per le stringhe si adotta il protocollo text definendo la lunghezza massima della stringa in tre casi:

- I testi lunghi (es. descrizioni, stringhe di sblocco, enigmi, note) hanno lunghezza massima `BUFLEN` (256 Byte)
- I testi brevi (es. username, password, nomi, risposte, argomenti dei comandi) hanno lunghezza massima `WORDLEN` (20 Byte)
- I comandi hanno lunghezza massima `CMDLEN` (6 Byte)

Per l'esito dei comandi e per l'invio del numero della stanza in `start` si adotta il protocollo binary con tipo di dato `uint8_t` (non servono numeri maggiori di 255 e si evita la trasformazione in formato host/network).

Per l'invio dal server al client della stringa delle note, che può variare in dimensioni da zero a `MAXNOTES * BUFLEN` caratteri, si è preferito inviare prima la lunghezza della stringa e poi il numero di caratteri preciso, così da non sprecare grandi quantità di traffico quando il blocco note è vuoto.