Relazione progetto Reti Informatiche

Il paradigma utilizzato nel progetto è il **client-server**, in particolare si evidenzia la presenza di un unico server e più client che svolgono la funzione di client utilizzato dall'utente per prenotare, kitchen device utilizzati in cucina e table device utilizzati dagli utenti per ordinare i piatti durante il pasto. Il server implementato è di tipo **iterativo**, più semplice da gestire ma non consente la gestione di più richieste in contemporanea, risulta inefficiente in presenza di numerosi client, i quali dovrebbero aspettare prima di poter ricevere risposta dal server.

Nel server si utilizza il **multiplexing dell'I/O** che consente di controllare più socket, e di gestire lo standard input, grazie a ciò è possibile inserire comandi nel server mentre questo attende messaggi dalla rete. Appena un dispositivo si connette al server invia un intero, in base ad esso il server riconosce il tipo di dispositivo, se è un client non fa nulla, se è un table o un kitchen device salva il socket descriptor ed assegna un codice 'BBB' per il table e 'CCC' per il kitchen in un array di strutture, appena un utente inserisce il codice corretto nel table, il codice da 'BBB' diventa il codice della prenotazione così si identifica un device in modo univoco, e facilmente si riesce ad informare un table device quando una sua comanda cambia di stato, mentre per il kitchen non si aggiorna il codice, dato che appena c'è una nuova comanda si informano tutti i kitchen con il numero di comande (asterischi) in attesa. Se un client si disconnette, il relativo elemento dell'array viene azzerato, appena si connette un nuovo client questo sarà inserito nella prima posizione libera, il vettore ha dimensione 20, supponendo che ci siano 10 tavoli al massimo e 10 kitchen device.

Per quanto riguarda il client, l'utente può controllare la disponibilità di una prenotazione utilizzando il comando **find**, il formato di tale comando è il seguente: **find stringa_cognome numero_partecipanti data ora**, dove per semplicità si è assunto che i tavoli abbiano massimo 8 posti, il cognome sia minimo di tre lettere, l'ora non contiene i minuti e si suppone che la prenotazione sia valida per un'ora, quindi ad esempio è possibile prenotare per le 14, un tavolo già prenotato per le 13, la data è nel formato gg-mm-aa. Una volta ricevuti i tavoli disponibili, l'utente può prenotare tramite il comando **book numero_tavolo**, può accadere che prenotando un tavolo libero, la prenotazione non possa essere effettuata, questo perché nel frattempo qualche altro utente potrebbe averlo prenotato, in questo caso si invita l'utente a fare una nuova find. Questo controllo viene fatto inviando con il comando book, i dati inseriti nella find precedente, con questa soluzione si inviano dati già mandati in precedenza ma risulta necessaria in quanto il server non memorizza lo stato di una find, essa viene sovrascritta con altri comandi. Dopo la prenotazione l'utente riceve un codice di tre caratteri alfanumerici minuscoli, il numero e la sala del tavolo prenotato esso può scollegarsi digitando il comando **esc**. La prenotazione viene salvata nel file **prenotazioni**, e il codice creato nel file **codici**.

Il table device inizialmente richiede l'inserimento del codice, una volta inserito e verificata la correttezza lato server, mostra i comandi disponibili, il comando **help** indica all'utente il formato e l'utilità dei comandi, **menu** mostra il menu giornaliero all'utente, per semplicità stampato direttamente nel codice, come alternativa migliore si potrebbe scrivere in un file e leggerlo da esso, in modo tale che le modifiche vengano apportate solamente al file. L'utente può richiedere i piatti con il comando **comanda piatto$_1$-quantità$_1$ ……. piatto$_n$-quantità$_n$**, utilizzabile più volte durante il pasto fino ad un massimo di 9 volte, per semplificare la gestione del numero di comanda e il relativo inserimento nel file **comande**, appena una comanda viene inviata, viene inserita nel file e messa in

stato di attesa. L'utente può richiedere il conto, tramite il comando **conto**, quando tutte le sue comande sono state servite e non mentre sono in attesa o in preparazione, a questo punto il server calcola il conto, anche in questo caso non essendoci un file del menu per semplicità, calcola direttamente il conto con i valori inseriti nel codice, quando l'utente riceve il conto il table device non viene chiuso, ma riparte dal punto in cui si chiede il codice, così eventuali nuovi clienti possono utilizzarlo a loro volta. Il tavolo da servire viene ricavato in base al codice inserito nel table device controllando nel file **prenotazioni**, questa è una limitazione in quanto l'utente se inserisse un codice corretto diverso dal suo, potrebbe risultare ad un tavolo diverso da quello effettivamente prenotato e lo stesso se si sedesse in un tavolo diverso dal suo e se inserisse il suo codice, il tavolo risulterebbe quello prenotato nonostante sia fisicamente un tavolo diverso. Si suppone che ogni utente si sieda nel tavolo effettivamente prenotato e inserisca il codice che gli appartiene.

Il kitchen device è utilizzato dagli chef in cucina per accettare comande in preparazione, con il comando **take**, si accetta la comanda in attesa da più tempo, la prima in attesa nel file **comande**, questa comanda passa in preparazione, le comande in preparazione accettate dal device vengono mantenute in memoria e sono concatenate attraverso il separatore **/**, lo chef può visualizzare le comande accettate tramite il comando **show**. Una volta che tutte le portate di una comanda sono pronte, si segnala al server tramite il comando **ready comanda$_n$-T$_n$**, con questo comando viene rimossa la comanda dalla memoria. L'aggiornamento del file delle comande avviene in questo dispositivo per ridurre le operazioni effettuate lato server. Quando l'utente richiede il conto, il server rimuove le comande relative, dal file delle comande. Il server stesso ha dei comandi utilizzabili come **stat**, mostra tutte le comande presenti ed il loro stato, **stat option**, con option=a/p/s per mostrare le comande in stato di attesa/preparazione/servizio, **stat T$_n$**, mostra lo stato delle comande del tavolo selezionato, il comando **stop** arresta il server e gli altri device solo se non ci sono più comande nel file.

I controlli sui comandi nel client e device vengono fatti lato client per alleggerire il carico del server, quando il server si disconnette dovrebbe cancellare le prenotazioni della giornata trascorsa dal file e i relativi codici dal file **codici**, in modo che possano essere riciclati, questa parte non è stata implementata, dato che si fa riferimento ad una singola giornata.

Si riscontra un difetto nel table device, quando il server invia l'aggiornamento dello stato di una comanda, la prima volta viene visualizzata solo premendo un tasto o eseguendo un comando sul device, nelle volte successive alla prima viene visualizzato normalmente. Nel table device quando si inserisce il codice, il numero di comanda parte da 1, se all'avvio del sistema sono presenti nel file altre comande relative a tale codice, come realizzato per verificare il funzionamento, il numero parte da 1 e non riprende la sequenza memorizzata nel file, ciò è irrilevante dato che si suppone che quando un utente si collega al table device non ci siano già comande con tale codice nel file.

L'**I/O multiplexing** è stato utilizzato anche nel table e kitchen device, questo per consentire di ricevere le notifiche dal server non appena le comande vengono inserite o cambiano stato e alternare la scrittura dei comandi da parte degli chef e dei commensali nel device.