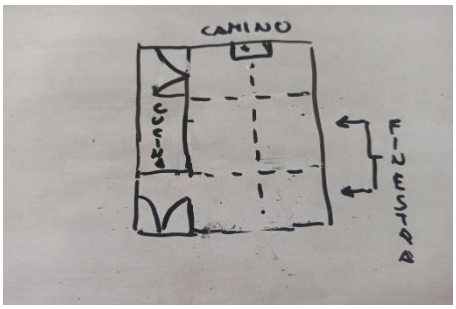
Documentazione

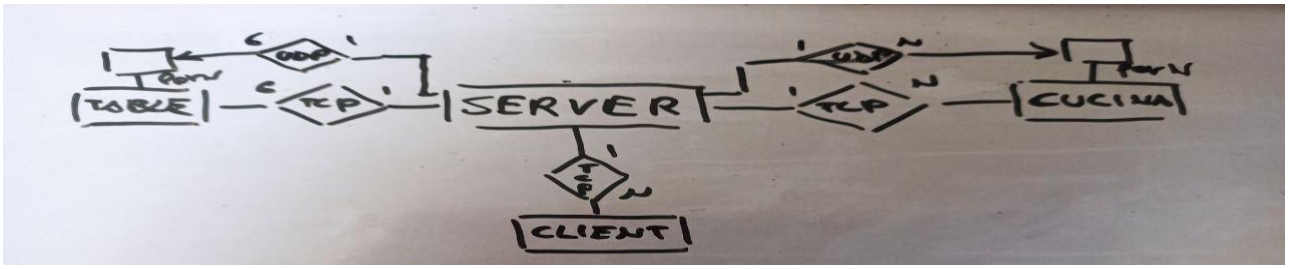
Si è immaginato una situazione.

Dove è stato possibile collocare sei tavoli tra i quali 4 con collocazioni speciali: vicino la finestra o vicino il camino.



I tavoli vengono inizializzati tutti all'avvio del server e gli vengono dati dei valori di default.

Il progetto è stato sviluppato secondo questa struttura



Presenta quindi un server al quale è possibile collegare vari dispositivi che fungono da client. Le tipologie dei client sono 3: Tavolo, Cucina e Cliente.

I client comunicano esclusivamente con il server tramite socket bloccanti. Per questo motivo, il server gestisce i vari socket in modo tale da non lasciare che il processo si blocchi: il server gli inserisce all'interno un set, in modo tale che vangano letti solo a seguito della primitiva select() solo quando risultano pronti. I comandi del server sono delegati a un processo figlio che gestisce gli input da tastiera li comunica al padre in TCP. Anche la cucina e il tavolo sono gestiti mediante l'uso di set, ma in questo caso vengono utilizzati per non bloccare il processo sull'attesa di un input da tastiera. Essi, inoltre, avranno un processo figlio con lo scopo di ricevere le notifiche da mostrare a video. Il Server fa da Single Point of Failure.

Le strutture sono le seguenti: Tavolo, prenotazione, comanda pietanza. Saranno gestite tutte tramite liste. In particolare, ci saranno liste come attesa, preparazione, servita per comande una lista dei tavoli e una per le presentazioni; quindi, tutte le strutture dati presenteranno un campo puntatore che punterà all'elemento successivo della lista.

Il tavolo avrà un identificativo che lo rende unico. Poi si avranno le informazioni quali il numero dei posti la sala, il codice per accedere al tavolo, un'aggiuntiva per descrivere una particolarità del tavolo. La lista di comande che il tavolo prenota con un contatore.

```
struct tavolo{
    int id;
    int n_posti;
    int n_comande;
    int cod_attivazione;
    struct comanda *comande;
    int sala;
    char aggiuntive[10];
    struct tavolo *puntatore;
};
```

```
struct comanda{
    int tavolo;
    status stato;
    int n_portate;
    int count;
    struct portata *pietanze;
    struct comanda *puntatore;
};
```

Per quanto riguarda le comande esse verrà inserite in più liste, dovranno avere anche l'identificatore del tavolo che l'ha ordinata che insieme al contatore la renderà unica. Essa avrà il numero e il puntatore al primo elemento della lista dei piatti che la compongono. Portata che sarà composta da un nome e la quantità di quel determinato piatto.

```
struct prenotazione{
    int tavolo;
    int codice;
    struct tm *data;
    struct prenotazione *puntatore;
};
```

Infine, avremo le prenotazioni composte dal tavolo con la data e il codice inviato al cliente. Le prenotazioni vengono salvate in prenotazioni.txt, da cui vengono poi caricate al momento di una nuova apertura. Esse inoltre sono inserite in una lista ordinate secondo l'ordine di prenotazione, quindi, la ricerca potrebbe essere molto onerosa.

Un ultimo argomento è la gestione delle connessioni. Esse sono quasi tutte di tipo TCP per dare più significato all'affidabilità. Le uniche UDP sono i segnali che vengono inviati dal server ai client quando c'è un avanzamento di stadio o di completamento richiesta (hanno tutte solo l'obbiettivo di notifica).

PS: Il tavolo si collega solo dopo aver inserito il codice. Quando si chiama il comando stop i tavoli non collegati rimangono accesi

Il server possiede una funzione con il quale è possibile cambiare giorno anche manualmente. Il Formato è il seguente: "data -> invio" "gg-mm-yy ora" dove mm va da 0 a 11 e l'ora deve rimanere all'interno dell'orario di apertura 12-15