

Progetto di

Laboratorio di Sistemi Operativi

Il supermercato

Davide Di Pierro  
Emilia Napolano

Immagine che contiene testo, Carattere, logo, simbolo

Descrizione generata automaticamente

Questa pagina è stata lasciata vuota intenzionalmente.

Sommario

[Analisi e specifica dei requisiti 4](#_Toc159502801)

[Il supermercato 4](#_Toc159502802)

[Ingresso e uscita del supermercato 4](#_Toc159502803)

[I cassieri 4](#_Toc159502804)

[Acquisti del cliente 4](#_Toc159502805)

[Requisiti 5](#_Toc159502806)

[Vincoli 5](#_Toc159502807)

[Architettura 5](#_Toc159502808)

[Protocollo di comunicazione client-server 6](#_Toc159502809)

[Concorrenza 7](#_Toc159502810)

[Numero clienti 7](#_Toc159502811)

[Coda casse 7](#_Toc159502812)

[Coda ingresso 7](#_Toc159502813)

[Carrelli 7](#_Toc159502814)

[Metodi 8](#_Toc159502815)

[Main 8](#_Toc159502816)

[Cliente 8](#_Toc159502817)

[Carrello 10](#_Toc159502818)

[Cassiere 10](#_Toc159502819)

[Coda Ingresso 11](#_Toc159502820)

[Coda Cassa 11](#_Toc159502821)

[Flusso di esecuzione 12](#_Toc159502822)

[Interfaccia grafica 13](#_Toc159502823)

[Cronologia lavoro 14](#_Toc159502824)

## Analisi e specifica dei requisiti

### Il supermercato

Il nostro supermercato è dotato di un numero predefinito di casse con annessi cassieri, apre le sue porte ai suoi clienti affezionati e fornisce loro dei prodotti di prima necessità con ottimo rapporto qualità prezzo.

Ingresso e uscita del supermercato

L’ingresso dei clienti al supermercato è supervisionato: non è possibile la presenza di più di C clienti all’interno del supermercato.  
Quando il supermercato è pieno, non appena si raggiungono C-E clienti allora è possibile farne entrare altri E.  
I clienti possono girovagare all’interno del negozio per un tempo variabile, non appena un cliente ha ultimato di decidere quali prodotti acquistare inseriti nel proprio carrello si mette in fila e aspetta il suo turno per pagare, una volta pagati i prodotti lascia il supermercato.  
Un cliente che non ha trovato i prodotti desiderati non si mette in coda alla cassa ma esce direttamente dal negozio.

I cassieri  
Per ogni cassa c’è un cassiere che chiama i clienti in ordine First in First out ovvero il primo a mettersi in coda sarà il primo ad essere servito, questo servizio avviene in un tempo costante o variabile a seconda di quanta spesa è stata fatta.

### Acquisti del cliente

Il cliente può comodamente acquistare i prodotti dall’applicazione facile ed intuitiva messa in dotazione dal supermercato.

## Requisiti

|  |  |  |
| --- | --- | --- |
| Nome | | Descrizione |
| Visualizza catalogo | Un cliente deve poter visualizzare il catalogo dei prodotti disponibili nel negozio. | |
| Inserimento prodotto nel carrello | Un cliente all’interno del supermercato deve poter inserire un prodotto nel proprio carrello. | |
| Visualizza carrello | Un cliente deve poter visualizzare i prodotti all’interno del proprio carrello. | |
| Togliere un prodotto dal carrello | Un cliente deve poter togliere un prodotto non desiderato dal carrello. | |
| Inserimento in coda | Un cliente deve poter mettersi in coda una volta scelti i prodotti da acquistare. | |
| Pagamento | Una volta raggiunto il suo turno un cliente deve poter pagare la merce presente nel suo carrello. | |
| Uscita con prodotti | Un cliente dopo aver acquistato i prodotti deve poter lasciare il supermercato. | |
| Uscita senza prodotti | Un cliente che non ha acquistato nulla non deve passare dalle casse ma deve poter uscire direttamente dal supermercato. | |
| Coda | Un cassiere deve poter servire i clienti in ordine FIFO. | |

## Vincoli

|  |  |  |
| --- | --- | --- |
| Nome | | Descrizione |
| Numero clienti nel supermercato | Non ci possono essere più di C clienti all’interno del supermercato. | |
| Entrata condizionata | Una volta raggiunti C-E clienti è possibile farne entrare altri E. | |
| Fuori uscita di un cliente | Un cliente che non interagisce più con il sistema viene automaticamente considerato come se avesse abbandonato il supermercato, perdendo così il suo carrello. | |

## Architettura

L'architettura adoperata è basata sul paradigma client-server.

Il server, implementato come un programma in linguaggio C, è responsabile della gestione completa del supermercato e dei cassieri.

L'elaborazione del server è suddivisa in differenti thread:

* Il thread principale (main thread) si occupa di accettare le richieste e di assegnarle a un thread dedicato per l'elaborazione;
* N thread dedicati ai cassieri, dove N rappresenta il numero di cassieri disponibili per assistere i clienti in coda;
* Un singolo thread dedicato alla pulizia dei carrelli, il quale periodicamente libera i carrelli non utilizzati da un certo periodo di tempo.
* Inoltre, è presente un thread dedicato all'interfaccia grafica per la visualizzazione e l'interazione con il sistema da parte degli utenti.

Ogni client è un'applicazione Android sviluppata in Java e rappresenta un utente interessato all'acquisto all'interno del supermercato.

## Protocollo di comunicazione client-server

Le richieste rivolte al server sono di tre tipologie: da parte del cliente, da parte del cassiere e per la richiesta del catalogo.

Il tipo di richiesta avviene secondo la seguente sintassi:

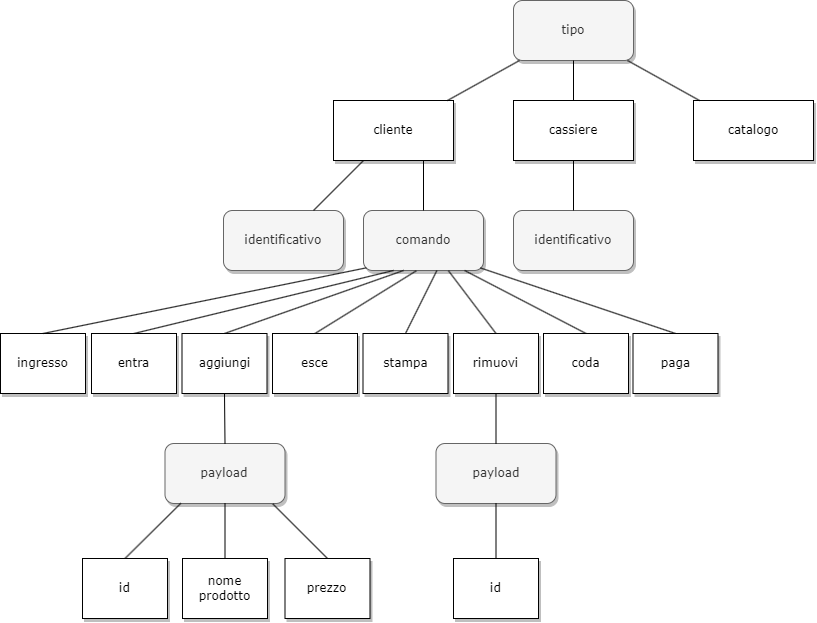
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tipo | : | identificativo | : | comando | \n |
| payload |  |  |  |  |  |

* **tipo**: definisce il tipo di richiesta e può assumere i seguenti valori:
  + **cliente**: utilizzato per una richiesta da parte del cliente
  + **cassiere**: utilizzato per una richiesta da parte del cassiere.

*Questa funzionalità non è stata ancora implementata in quanto i requisiti del cassiere sono stati soddisfatti da un thread interno al server.*

* + **catalogo**: utilizzato per richiedere il catalogo dei prodotti disponibili.

*In questo caso i campi identificativo, comando e payload sono assenti. La decisione di creare una richiesta separata per il catalogo è stata presa per permettere ad utenti non identificati di visualizzarlo.*

* **identificativo**: contiene l’id dell’utente (cliente o cassiere) che effettua una richiesta al server.
* **comando***:* contiene il comando da eseguire sul server.
  + **Per il cliente:**
    - *ingresso* permette all’utente di accodarsi all’entrata del negozio;
    - *entra* permette all’utente di entrare nel negozio;
    - *esce* permette all’utente di uscire dal negozio;
    - *aggiungi* permette all’utente di aggiungere un prodotto al carrello;
    - *rimuovi* permette all’utente di rimuovere un prodotto dal carrello;
    - *stampa* permette di stampare il contenuto del carrello;
    - *coda* permette di mettersi in coda alle casse;
    - *paga* permette di effettuare il pagamento.
  + ***Per il cassiere:***

*Riservato per scopi futuri.*

* **payload**: permette di

specificare informazioni

aggiuntive per il comando.

## Concorrenza

### Numero clienti

Il numero dei clienti è una risorsa a cui accede ciascun thread-cliente, l’accesso alla risorsa è consentito in mutua esclusione ai thread che ne richiedono l’accesso tramite l’utilizzo di un mutex gestito dalle funzioni:

* [get\_n\_clienti](#_Cliente)
* [incrementa\_n\_clienti](#_Cliente)
* [decrementa\_n\_clienti](#_Cliente)

### Coda casse

In quanto ciascun cliente necessita di mettersi in coda alle casse, è necessario gestire l’accesso alla coda in modo che due clienti non provino ad accedere allo stesso posto in coda.

La gestione è effettuata attraverso un mutex nelle funzioni per la struttura della coda.

### Coda ingresso

Allo stesso modo è necessario gestire l’accesso alla coda per accedere al supermercato.

### Carrelli

Per evitare che due clienti tentino di accedere allo stesso carrello, ciascun carrello è gestito da un mutex e da una variabile di stato (LIBERO, IN\_NEGOZIO, IN\_CODA, IN\_CASSA, PAGAMENTO, PAGATO) che permettono di identificare un carrello libero all’interno del supermercato.

Quando un cassiere necessita di elaborare il carrello di un cliente, quest’ultimo cede il mutex al cassiere in modo che possa elaborarlo.

Al termine del lavoro del cassiere, il mutex viene rilasciato e lo stato riportato su LIBERO. In modo che un nuovo cliente possa utilizzarlo.

Immagine che contiene robot, giocattolo, macchina, cartone animato

Descrizione generata automaticamente

## Metodi

### Main

Nel main vengono inizializzati i carelli e vengono fatti entrare i cassieri per ogni cassa.  
Viene inizializzato il server e vengono creati i thread per i clienti e per pulire i carrelli (ogni tot secondi se un cliente non interagisce con il sistema si dà per scontato che abbia abbandonato il negozio e quindi si libera il carrello).  
Di seguito vengono collegati i socket all’indirizzo e alla porta specificati e si mette il server in ascolto in attesa di connessioni da parte dei clienti.

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Parametri | Tipo | Descrizione |
| process | void \* ptr | void\* | Serve a processare la richiesta del client. Processa richieste da parte del tipo cliente, cassiere o catalogo. |
| send\_response | int socket, char \* response | void | Invia la risposta alla socket del client. |
| read\_request | int socket, char\* request | void | Legge la richiesta della socket del client. |
| inviaCatalogo | char\* response | void | Server a leggere il file catalogo.txt dove è presente la lista dei prodotti. |
| riordinaCarrelli |  | void\* | Serve a riordinare i carrelli inutilizzati. |
| ui |  | void\* | Serve per la rappresentazione grafica da terminale dei clienti che interagiscono con il supermercato. |
| stampa\_stickman | int num\_stickman | void | Serve disegnare gli stickman che rappresentano persone all’interno del negozio. |

### Cliente

Nel file cliente.c ci si occupa di effettuare il parsing delle richieste, di gestire l’ingresso e l’uscita di un cliente, di poter permettere al cliente l’aggiunta e la rimozione di prodotti nel carello, di stampare il prezzo da pagare in cassa.  
Inoltre si occupa di poter permettere l’accodarsi di un cliente in cassa con l’annesso pagamento.

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Parametri | Tipo | Descrizione |
| get\_n\_clienti |  | int | Serve a reperire il numero di clienti presenti all’interno del negozio. |
| incrementa\_n\_clienti |  | void | Serve ad incrementare il numero di clienti nel negozio. |
| decrementa\_n\_clienti | |  | void | Serve a decrementare il numero di clienti nel negozio. |
| clienteParser | | char\* request, char\* response, carrello\_t\* carrelli, coda\_casse\_t\* coda\_casse, coda\_ingresso\_t\* coda\_ingresso | void | Serve ad effettuare il parsing così come spiegato nella [sezione dedicata al protocollo.](#_Protocollo) |
| clienteEntra | | int\* id, char\* response, carrello\_t\* carrelli, coda\_ingresso\_t\* coda\_ingresso | void | Permette ad un utente autorizzato di entrare una volta eseguito il comando *entra,* così facendo rimuove l’utente dalla coda esterna al negozio e gli assegna un carrello. |
| puoEntrare | | coda\_ingresso\_t\* coda ingresso | bool | Verifica se l’utente può entrare in base ai vincoli definiti [qui.](#_Vincoli) |
| clienteEntraInCodaIngresso | | int id, char\* response, coda\_ingresso\_t\* coda\_ingresso | void | Aggiunge un utente alla coda all’esterno del negozio e gli assegna una posizione. |
| clienteEsce | | int id, char\* response, carrello\_t\* carrelli | void | Permette ad un utente di uscire nel momento immediatamente successivo al pagamento oppure se la sua sessione è scaduta. |
| clienteAggiunge | | int id, char\* equest, char\* response, carrello\_t\* carrelli | void | Permette ad un cliente di aggiungere un prodotto al suo carrello tramite il comando aggiungi. |
| clienteRimuove | | int id, char\* request, char\* response, carrello\_t\* carrelli | void | Permette ad un cliente di rimuovere un prodotto dal carrello tramite il comando rimuovi. |
| clienteStampa | | int id, char\* response, carrello\_t\* carrelli | void | Stampa il prezzo totale da pagare in cassa. |
| clienteSiMetteInCodaAllaCassa | | int id, char\* response, carrello\_t\* carrelli, coda\_casse\_t\* casse | void | Permette al cliente di accodarsi alla cassa con il suo carrello. |
| clientePaga | | int id, char\* response, carrello\_t\* carrelli | void | Permette il pagamento dei prodotti all’interno del carrello di un cliente. |

### Carrello

Tramite il file carrello.c è possibile inizializzare i carrelli all’interno del supermercato, aggiungere e rimuovere un prodotto dal carrello, stamparlo e calcolarne il totale da pagare.   
Inoltre, dà la possibilità di svuotare il carrello.

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Parametri | Tipo | Descrizione |
| aggiungi\_prodotto | carrello\_t\* carrello, prodotto\_t prodotto | void | Aggiunge un prodotto al carrello utilizzando una struttura di tipo coda. |
| rimuovi\_prodotto | carrello\_t\* carrello, int id\_prodotto | void | Rimuove un prodotto preciso dal carrello. |
| stampa\_carrello | char\* stringa, carrello\_t\* carrello | void | Stampa i prodotti all’interno del carrello. |
| calcola totale | carrello\_t\* carrello | float | Calcola il prezzo totale di tutti i prodotti all’interno del carrello. |
| inizializza\_carrelli | carrello\_t\* carrelli | void | Inizializza i carrelli in base a quante persone possono entrare nel negozio. |
| svuota\_carrello | carrello\_t\* carrello | void | Svuota un carrello. |

### Cassiere

Tramite il file cassiere.c è possibile far entrare un cassiere nel supermercato per poter servire i clienti.

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Parametri | Tipo | Descrizione |
| cassiereEntra | int tempoCassiere, int tempoElaborazioneProdotto, carrello\_t\* carrelli, coda\_casse\_t\* coda\_casse | void | Permette al cassiere di entrare nel supermercato e di inizializzare il suo thread. |
| aspettaFila | void\* ptr | void\* | Attende continuamente che una persona si accodi alla cassa per poterla servire ed elaborare il suo carello. |
| elaboraCarrello | void\* ptr | void\* | Permette al cassiere di elaborare un carrello e di procedere al pagamento. |

### Coda Ingresso

Il file codaIngresso.c permette la gestione del formarsi della coda all’esterno del supermercato (assegnando a ciascun cliente una posizione, proprio come si fa con i numerini della posta) per poter poi successivamente scegliere in che modo possono entrare gli utenti.

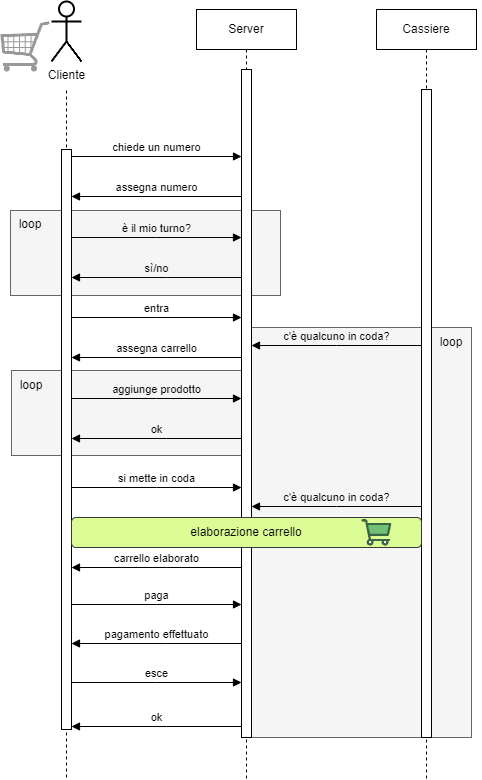
|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Parametri | Tipo | Descrizione |
| aggiungi\_cliente\_coda\_ingresso | int id\_cliente, coda\_ingresso\_t\* coda\_ingresso | void | Permette l’aggiunta di un utente alla coda esterna al supermercato, anche in questo caso con una struttura di tipo coda. |
| rimuovi\_cliente\_coda\_ingresso | coda\_ingresso\_t\* coda\_ingresso | int | Permette la rimozione di un cliente dalla coda. |
| rimuovi\_cliente\_coda\_ingresso\_id | int id\_cliente, coda\_ingresso\_t\* coda\_ingresso | int | Permette la rimozione di un cliente dalla coda in base al proprio id. |
| numero\_cienti\_coda\_ingresso | coda\_ingresso\_t\* coda\_ingresso | int | Restituisce il numero di clienti attualmente in coda per entrare nel supermercato. |
| posizione\_cliente\_coda\_ingresso | int id\_cliente, coda\_ingresso\_t\* coda\_ingresso | int | Restituisce la posizione attuale di un cliente. |

### Coda Cassa

Il file codaCassa.c permette di gestire la coda alle casse secondo una politica di tipo FIFO.

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Parametri | Tipo | Descrizione |
| aggiungi\_cliente\_coda | int id\_cliente, coda\_casse\_t\* coda\_casse | void | Permette l’aggiunta di un utente alla coda per la cassa, anche in questo caso con una struttura di tipo coda. |
| rimuovi\_cliente\_coda | coda\_casse\_t\* coda\_casse | int | Permette la rimozione di un cliente dalla coda. |
| rimuovi\_cliente\_coda\_id | int id\_cliente, coda\_casse\_t\* coda\_casse | int | Permette la rimozione di un cliente dalla coda in base al proprio id. |
| numero\_cienti\_coda | coda\_casse\_t\* coda\_casse | int | Restituisce il numero di clienti attualmente in coda per la cassa. |
| posizione\_cliente\_coda | int id\_cliente, coda\_casse\_t\* coda\_casse | int | Restituisce la posizione attuale di un cliente. |

## Flusso di esecuzione



## Interfaccia grafica

## Cronologia lavoro

|  |  |  |  |
| --- | --- | --- | --- |
| Modifiche apportate da | In data | Modifiche | In relazione a |
| Davide Di Pierro Emilia Napolano | 28 Dicembre 2023 | Impostazione del protocollo con creazione del parser per le diverse richieste.  I clienti riescono ad entrare e ad uscire dal negozio. | Server |
| Davide Di Pierro Emilia Napolano | 29 Dicembre 2023 | Inserita la gestione del carrello e del catalogo. | Server |
| Davide Di Pierro Emilia Napolano | 30 Dicembre 2023 | Ci siamo resi conto che non siamo soddisfatti della resa del parsing che abbiamo costruito, quindi stiamo pensando ad altre soluzioni. | Riflessione |
| Davide Di Pierro Emilia Napolano | 12 Gennaio 2024 | Inizio stesura documentazione. | Documentazione |
| Davide Di Pierro Emilia Napolano | 13 Gennaio 2024 | Impostazione del client su android studio. | Client |
| Davide Di Pierro Emilia Napolano | 15 Febbraio 2024 | Implementazione pagamento e coda alla cassa. | Server |
| Davide Di Pierro Emilia Napolano | 19 Febbraio 2024 | Implementazione rimozione del cliente dalla coda. | Server |
| Davide Di Pierro Emilia Napolano | 20 Febbraio 2024 | Aggiunte funzioni per la gestione de cliente e del menu principale con annessa rappresentazione grafica da terminale per il server. | Server |
| Davide Di Pierro Emilia Napolano | 21 Febbraio 2024 | Stesura dei metodi utilizzati all’interno della documentazione. | Documentazione |
| Davide Di Pierro  Emilia Napolano | 22 Febbraio 2024 | Implementazione della coda e della logica per l’ingresso. | Server |