

Progetto di Laboratorio di Sistemi Operativi

Il supermercato

Davide Di Pierro
Emilia Napolano



UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II



Questa pagina è stata lasciata vuota intenzionalmente.

Sommario

Analisi e specifica dei requisiti	4
Il supermercato	4
Ingresso e uscita del supermercato	4
I cassieri.....	4
Acquisti del cliente	4
Requisiti.....	5
Vincoli	5
Spiegazione delle decisioni prese	5
Architettura.....	6
Disconnessione	6
Protocollo di comunicazione client-server.....	7
Concorrenza	8
Numero clienti.....	8
Coda casse	8
Coda ingresso	8
Chiocciola.....	8
Cassieri.....	8
Carrelli	8
Moduli	9
Main.....	9
Cliente	9
Direttore.....	10
Carrello	11
Cassiere.....	11
Coda Ingresso	12
Coda Cassa.....	12
Flusso di esecuzione	13
Flusso cliente-server-cassiere	13
Flusso server-direttore-buttafuori-addetto	14
Interfaccia grafica	15
Grafica Client Android.....	15
Grafica Server da terminale.....	17
Cronologia lavoro	17

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

Vincoli

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

Spiegazione delle decisioni prese

Si è deciso di rappresentare il cassiere, il direttore, il buttafuori e l'addetto ai carrelli come thread e non come processi separati poiché la facilità della gestione di questo tipo di parallelizzazione permette di gestire una quantità di client maggiore grazie alla leggerezza ottenuta.

Abbiamo deciso di sviluppare il client anche su Android dato che ci permette di poter far visionare il nostro prodotto finale in maniera interattiva ed intuitiva.

Vista la sua facilità di apprendimento abbiamo deciso di utilizzare docker-compose rispetto ad ADE.

La struttura del protocollo che abbiamo utilizzato ci è sembrata la più facile con la quale lavorare sia dal client c che dal client Android.



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:

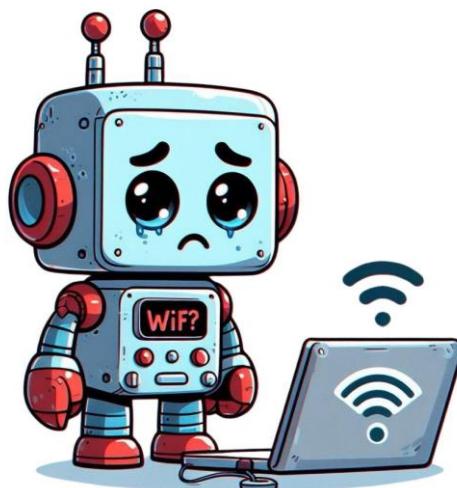
- Il thread principale (main thread) si occupa di accettare le richieste e di assegnarle a un thread dedicato per l'elaborazione;
- N thread dedicati ai cassieri, dove N rappresenta il numero di cassieri disponibili per assistere i clienti in coda;
- Un singolo thread dedicato alla pulizia dei carrelli, il quale periodicamente libera i carrelli non utilizzati da un certo periodo di tempo;
- Un singolo thread dedicato alla rimozione di clienti che non interagiscono con il sistema quando si trovano in coda all'ingresso;
- Inoltre, è presente un thread dedicato all'interfaccia grafica per la visualizzazione e l'interazione con il sistema da parte degli utenti;
- Un ultimo thread che rappresenta il direttore del supermercato che controlla che i clienti senza acquisti possano uscire.

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

Disconnessione

Nel caso in cui un client si disconnetta abbiamo optato per due soluzioni:

- Buttafuori: si occupa di controllare se i clienti hanno abbandonato la coda all'ingresso ogni tot di tempo e in tal caso si preoccupa di farla scorrere;
- Addetto ai carrelli: si occupa di controllare se un cliente all'interno del supermercato è andato via abbandonando così il suo carrello, in tal caso il carrello viene impostato come libero in modo da poter essere riutilizzato da un altro cliente.



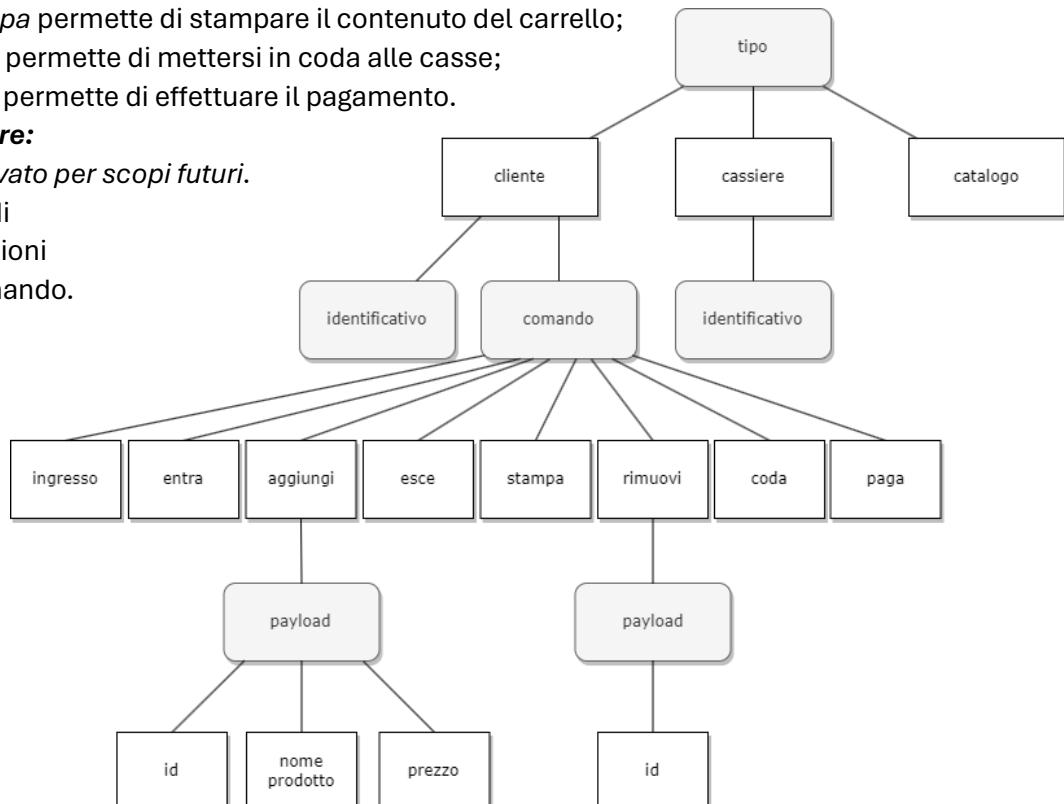
Protocollo di comunicazione client-server

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](#)
- [incrementa_n_clienti](#)
- [decrementa_n_clienti](#)

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.

Chiocciola

Per evitare che più clienti prendano lo stesso numero dalla chiocciola all'esterno del supermercato è stato aggiunto un mutex anche in questa sezione.

Cassieri

Per evitare che più cassieri provino ad assistere un cliente nello stesso momento, è stato aggiunto un mutex al cassiere.

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, CONFERMA) che permettono di identificare un carrello libero all'interno del supermercato.

Quando un cliente si mette in coda alle casse, cede il mutex del proprio carrello, che verrà preso dal cassiere in caso di acquisti o dal direttore in caso di uscita senza acquisti.

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



Moduli

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 cliente, del 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	Serve a leggere il file catalogo.json 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.
buttafuoriAllIngresso		void*	Serve a cacciare utenti che non interagiscono più con il sistema.

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.

Tiene conto di quando è stata effettuata l'ultima operazione.

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*	void	Serve ad effettuare il parsing così come spiegato nella sezione dedicata al protocollo .

	coda_casse, coda_ingresso_t* coda_ingresso		
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 <i>entra</i> , 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 .
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* request, char* response, carrello_t* carrelli	void	Permette ad un cliente di aggiungere un prodotto al suo carrello tramite il comando <i>aggiungi</i> .
clienteRimuove	int id, char* request, char* response, carrello_t* carrelli	void	Permette ad un cliente di rimuovere un prodotto dal carrello tramite il comando <i>rimuovi</i> .
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.

Direttore

Il file *direttore.c* si occupa di verificare che un cliente senza acquisti possa uscire senza problemi.

Nome	Parametri	Tipo	Descrizione
controllaUscita	void * ptr	void*	Permette ad un cliente che non ha fatto acquisti di uscire dal negozio marcando il suo carrello come "pagato".

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 id, int tempoCassiere, int tempoElaborazioneProd otto, 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 carrello.
elaboraCarrello	void* ptr	void*	Permette al cassiere di elaborare un carrello e di procedere al pagamento.

Coda Ingresso

Il file `codalngresso.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

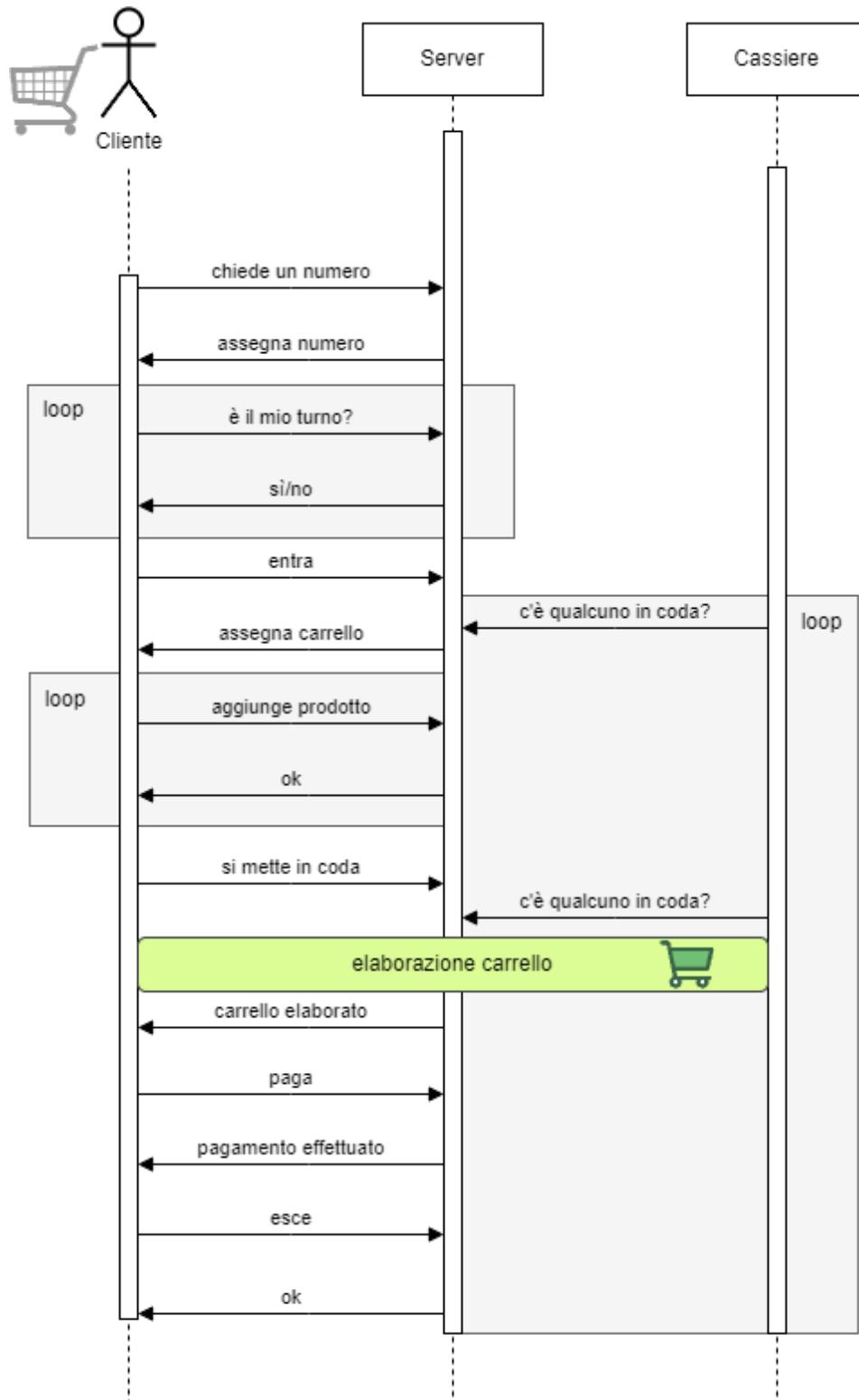
Il file `codaCassa.c` permette di gestire la coda alle casse secondo una politica di tipo FIFO.

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

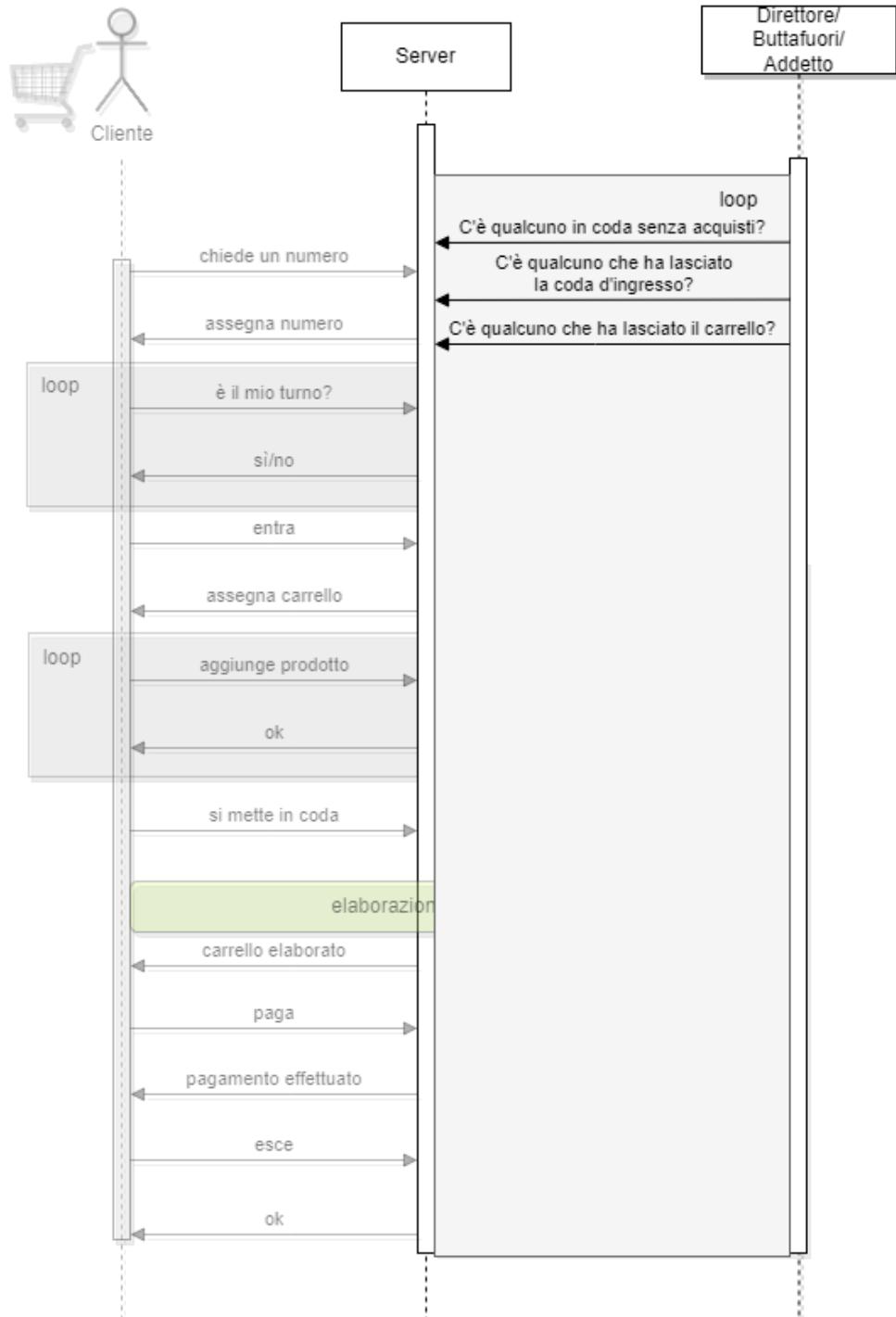
Flusso cliente-server-cassiere

Normale flusso di esecuzione con un cliente, il server e il thread per il cassiere.



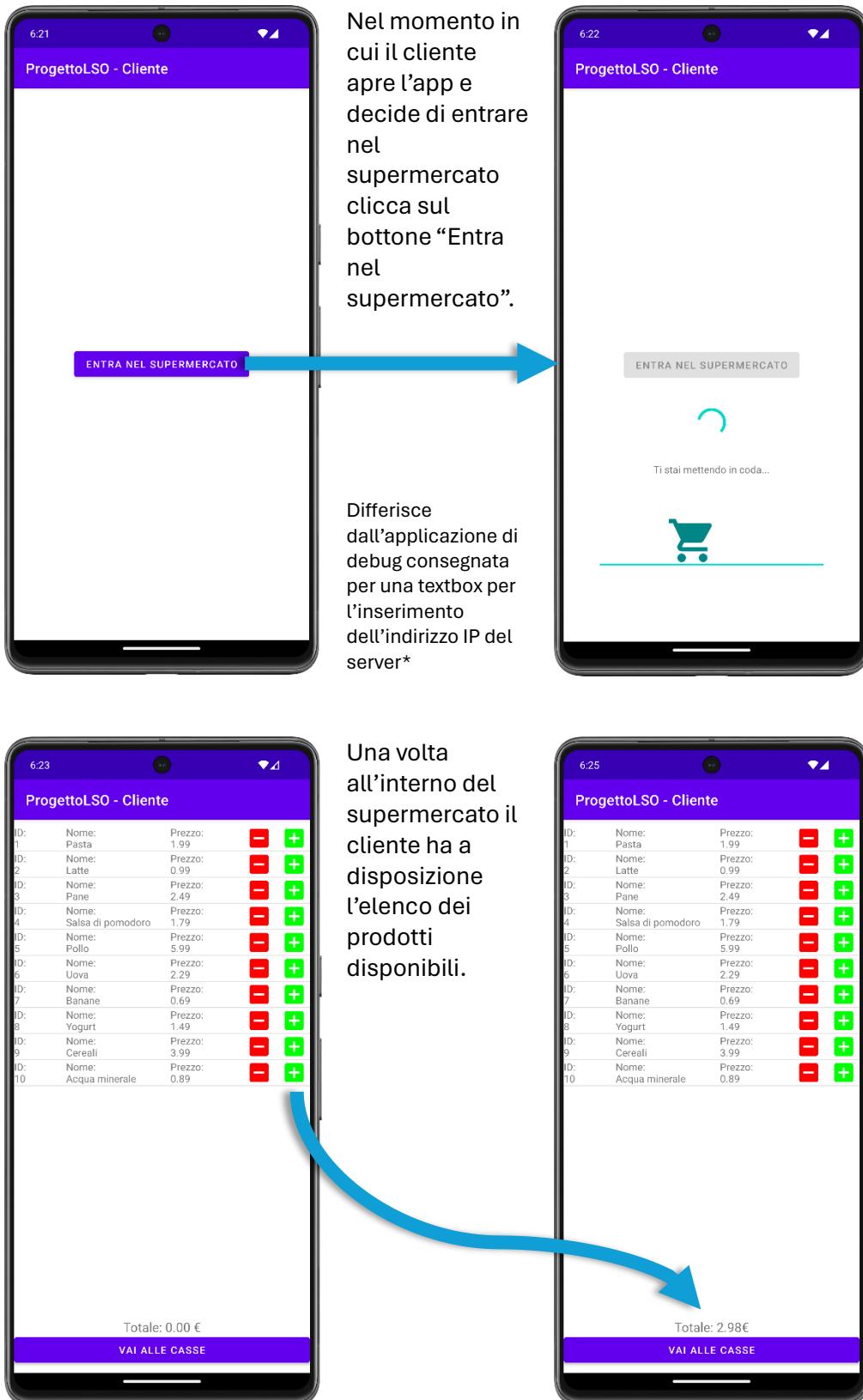
Flusso server-direttore-buttafuori-addetto

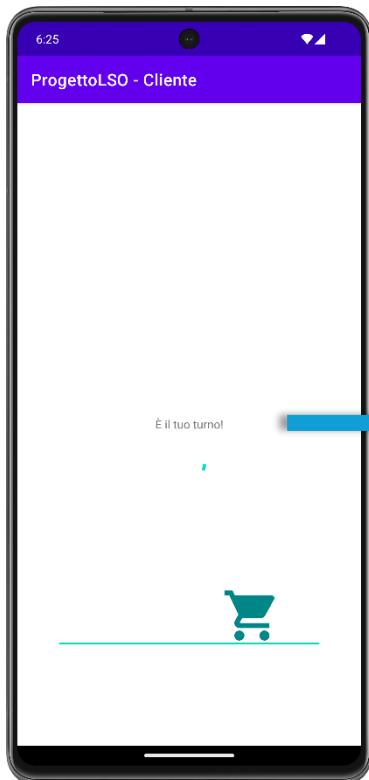
Flusso di esecuzione comprensivo dei thread Direttore, Buttafuori e Addetto ai carrelli.



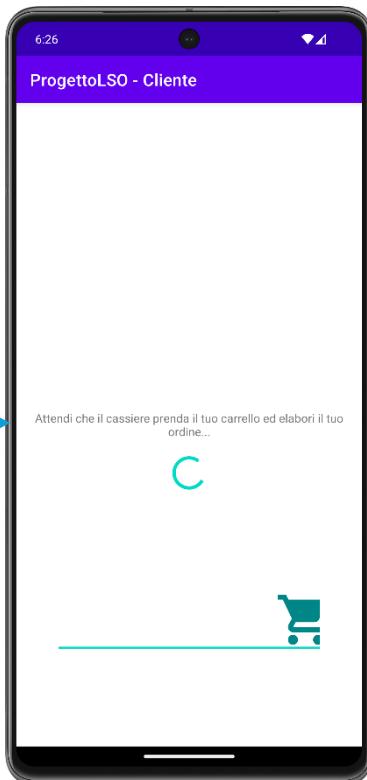
Interfaccia grafica

Grafica Client Android

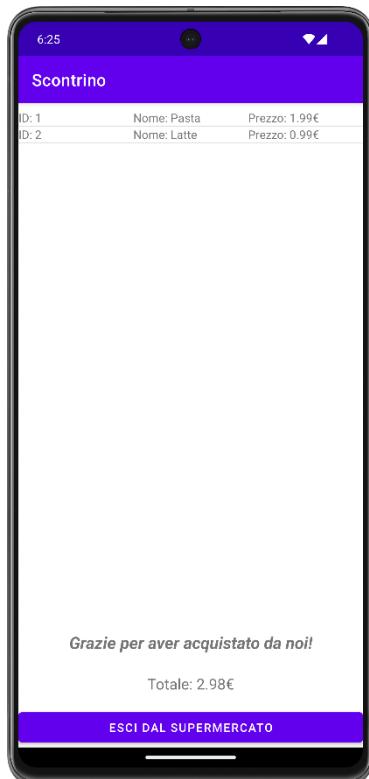




Quanto il cliente decide di procedere alle casse si mette in coda e attende il suo turno.



Il cliente viene notificato quando è arrivato il suo turno e il cassiere sta elaborando il carrello.



Una volta che il cassiere ha terminato di elaborare il carrello, il cliente visualizza lo scontrino ed esce dal supermercato.



Il cliente viene riportato alla schermata iniziale.

Grafica Server da terminale

Sul terminale del server è possibile visionare graficamente la situazione all'interno del supermercato, diviso così in cinque sezioni:

- *Fuori*: permette di visualizzare quante persone si trovano attualmente all'esterno del supermercato in coda per poter entrare;
- *In negozio*: permette di visualizzare quante persone si trovano attualmente all'interno del negozio intente ad acquistare prodotti;
- *In coda*: permette di visualizzare quante persone si trovano attualmente in coda per le casse;
- *In cassa*: permette di visualizzare quante persone si trovano alle casse mentre sono serviti dai cassieri disponibili;
- *Uscita senza acquisti*: permette di visualizzare le persone che escono dal negozio senza aver effettuato acquisti.

```

Update: 78
serverls0
serverls0 | FUORI:
serverls0 | 0   0   0   0   0   0
serverls0 | / \ / \ / \ / \ / \
serverls0 | |   |   |   |   |
serverls0 | / \ / \ / \ / \ / \
serverls0 |
serverls0 |
serverls0 | IN NEGOZIO:
serverls0 | 0
serverls0 | / \
serverls0 | |
serverls0 | / \
serverls0 |
serverls0 | IN CODA:
serverls0 | 0   0   0   0   0   0
serverls0 | / \ / \ / \ / \ / \
serverls0 | |   |   |   |   |
serverls0 | / \ / \ / \ / \ / \
serverls0 |
serverls0 |
serverls0 | IN CASSA:
serverls0 | 0   0
serverls0 | / \ / \
serverls0 | |   |
serverls0 | / \ / \
serverls0 |
serverls0 | USCITA SENZA ACQUISTI:
serverls0 | 0
serverls0 | / \
serverls0 | |
serverls0 | / \
serverls0 |

```

Cronologia lavoro

Modifiche apportate da	In data	Modifiche	In relazione a
<i>Davide Di Pierro Emilia Napolano</i>	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
<i>Davide Di Pierro Emilia Napolano</i>	29 Dicembre 2023	Inserita la gestione del carrello e del catalogo.	Server

<i>Davide Di Pierro Emilia Napolano</i>	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
<i>Davide Di Pierro Emilia Napolano</i>	12 Gennaio 2024	Inizio stesura documentazione.	Documentazione
<i>Davide Di Pierro Emilia Napolano</i>	13 Gennaio 2024	Impostazione del client su android studio.	Client
<i>Davide Di Pierro Emilia Napolano</i>	15 Febbraio 2024	Implementazione pagamento e coda alla cassa.	Server
<i>Davide Di Pierro Emilia Napolano</i>	19 Febbraio 2024	Implementazione rimozione del cliente dalla coda.	Server
<i>Davide Di Pierro Emilia Napolano</i>	20 Febbraio 2024	Aggiunte funzioni per la gestione de cliente e del menu principale con annessa rappresentazione grafica da terminale per il server.	Server
<i>Davide Di Pierro Emilia Napolano</i>	21 Febbraio 2024	Stesura dei metodi utilizzati all'interno della documentazione.	Documentazione
<i>Davide Di Pierro Emilia Napolano</i>	22 Febbraio 2024	Implementazione della coda e della logica per l'ingresso.	Server
<i>Davide Di Pierro Emilia Napolano</i>	23 Febbraio 2024	Stesura autopilota per l'invio simultaneo di più client. Aggiustata l'entrata vincolata.	Client Server
<i>Davide Di Pierro Emilia Napolano</i>	25 Febbraio 2024	Ultimazione interfaccia grafica Android per il client e terminazione interfaccia grafica da terminale per il server.	Client e Server
<i>Davide Di Pierro Emilia Napolano</i>	4 Aprile 2024	Verifica finale e consegna del progetto.	Documentazione

Info utili:

- [Github](#);
- [Download apk](#);
- Customer service 😊 :
 - d.dipierro@studenti.unina.it
 - emi.napolano@studenti.unina.it