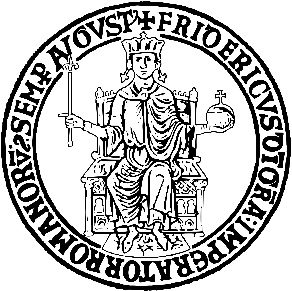
**UNIVERSITÀ DEGLI STUDI DI NAPOLI “FEDERICO II”**



Dipartimento di Ingegneria Elettrica e delle Tecnologie dell’Informazione

Corso di Laurea Triennale in Informatica

**PROGETTO DI LABORATORIO DI SISTEMI OPERATIVI**

**IL SUPERMERCATO**

**Studenti**

Gianfranco Duminuco

N86 004061

Fabrizio Formicola

N86 003487

Anno Accademico 2023/2024

Sommario

[Traccia 3](#_Toc172627396)

[Analisi e specifica dei requisiti 3](#_Toc172627397)

[Architettura e scelte gestionali 3](#_Toc172627398)

[Protocollo per la comunicazione client-server 3](#_Toc172627399)

[Moduli e rispettive funzionalità 3](#_Toc172627400)

[Gestione della concorrenza 3](#_Toc172627401)

[Sequence Diagram per il flusso di esecuzione 3](#_Toc172627402)

[Simulazione del flusso di esecuzione tramite interfaccia grafica 3](#_Toc172627403)

# Traccia

**Progetto 1: Il supermercato** - Lo studente dovrà realizzare la simulazione di un sistema che modella un supermercato con K casse e frequentato da un certo numero di clienti. Il numero dei clienti nel supermercato è contingentato: non ci possono essere più di C clienti che fanno acquisti (o che sono in coda alle casse) in ogni istante. All’inizio, tutti i clienti entrano contemporaneamente nel supermercato, successivamente, non appena il numero dei clienti scende a C-E (0<E<C), ne vengono fatti entrare altri E. Ogni cliente spende un tempo variabile T all’interno del supermercato per fare acquisti, quindi si mette in fila in una delle casse che sono in quel momento aperte ed aspetta il suo turno per “pagare” la merce acquistata. Dopo aver pagato, il cliente esce dal supermercato. Ogni cassa attiva ha un cassiere che serve i clienti in ordine FIFO con un certo tempo di servizio. Il tempo di servizio del cassiere ha una parte costante (diversa per ogni cassiere) più una parte variabile che dipende linearmente dal numero di prodotti acquistati dal cliente che sta servendo. I clienti che non hanno acquistato prodotti (P=0), non si mettono in coda alle casse.

Opzionale: i clienti senza acquisti non si mettono in coda, ma prima di uscire dal supermercato devono attendere il permesso di uscire. Il permesso può essere dato da un direttore del supermercato.

Opzionale: la scelta degli acquisti di un cliente può essere fatta o in ordine casuale oppure scelta dal cliente su una semplice interfaccia grafica/testuale

# Analisi e specifica dei requisiti

Asdsad

# Architettura e scelte gestionali

Il progetto consiste nella produzione un applicativo **client-server**, usando **Docker-compose**.

Il Server è stato sviluppato (come da requisiti) in **linguaggio C**, mentre il Client è stato sviluppato in linguaggio **Dart**, in particolare con l'utilizzo del framework **Flutter**.

Client e Server sono in **locale** e comunicano attraverso l'uso di **Socket** (come da requisiti).

La scelta di utilizzare Docker-compose è dovuta alla facilità di apprendimento del tool.

La scelta di utilizzare Flutter è dovuta ad:

ottime capacità del framework nella gestione delle socket attraverso funzioni e widgets forniti dalla libreria ‘dart:io’;

possibilità di creare, con un codice snello e facilmente intuibile, degli applicativi funzionanti sia su dispositivi Android che iOS, nonché applicazioni desktop o web-based;

esperienza pregressa degli sviluppatori nell’utilizzo di tale framework.

ghhgh

## Gestione della concorrenza

# Protocollo per la comunicazione client-server

Sdfsdf

# Moduli e rispettive funzionalità

## Main

**Descrizione**: gestisce la logica generale del server.

**Funzionalità**:

|  |  |
| --- | --- |
| main | Entry point del server. Inizializza i carrelli, la coda alla cassa, la coda all'ingresso, i mutex e i cassieri. |
| process | Processa la richieste che arrivano dal client. |
| read\_request | Legge le richieste dal client. |
| send\_response | Invia le risposte al client. |
| sendCatalog | Restituisce il catalogo coi prodotti all’utente. |
| reorderCarts | Pulisce e “risistema” i carrelli se non vengono utilizzati per un certo periodo di tempo. |
| bouncerAtEntrance | Controlla che ci siano ancora clienti in fila all’ingresso, e nel caso in cui questi non entrino in un certo tempo, li butta fuori. |
| ui | Serve ad aggiornare costantemente l’interfaccia grafica |

## Cart

**Descrizione**: classe per la gestione del carrello e degli oggetti al suo interno.

**Funzionalità**:

|  |  |
| --- | --- |
| add\_product | Per aggiungere prodotti al carrello. |
| remove\_product | Per rimuovere prodotti dal carrello. |
| print\_cart | Per la stampa nella socket del contenuto del carrello. |
| calculate\_total | Calcola il prezzo totale degli oggetti nel carrello. |
| initialize\_carts | Svuota completamente un carrello e ne setta lo stato a “FREE”, rendendolo disponibile per i prossimi clienti. |
| clear\_cart | Svuota il carrello. |

## Cashier

**Descrizione**: classe per la gestione delle casse e delle funzionalità dei cassieri.

**Funzionalità**:

|  |  |
| --- | --- |
| cashierEnters | Crea ed inizializza un cassiere, impostando anche i valori “variabili” sui suoi tempi per il processamento dei prodotti. |
| waitQueue | Funzione che consente al cassiere di restare in attesa di eventuali carrelli che si trovano nella coda per andare alla cassa. |
| processCart | Funzione che gestisce le operazioni del cassiere nel processare un carrello che arriva dalla coda, consentendogli di effettuare il “checkout” e di pagare. |

## Client

**Descrizione**: classe per la gestione di tutte le operazioni effettuabili da un client(e).

**Funzionalità**:

|  |  |
| --- | --- |
| get\_clients\_number | Per ottenere il numero di clienti nel supermercato. |
| increase\_clients\_number | Per incrementare il numero di clienti nel supermercato. |
| decrease\_clients\_number | Per decrementare il numero di clienti nel supermercato. |
| clientParser | Per effettuare il parsing delle richieste ricevute dal client. |
| clientEnters | Gestisce l’entrata di un cliente nel supermercato: se il cliente è libero di poter entrare poiché ci sono carrelli disponibili, viene fatto entrare, altrimenti viene messo in coda. |
| canEnter | Per verificare se un utente può entrare nel supermercato. |
| clientEntersInEntranceQueue | Gestisce le operazioni del cliente alla coda di ingresso: se un cliente è appena arrivato, gli viene assegnato un ticket. |
| clientExits | Gestisce le operazioni da effettuare dopo l’uscita dal supermercato di un cliente, come la pulizia del database. |
| clientAddItem | Per aggiungere un prodotto scelto al carrello. |
| clientRemoveItem | Per rimuovere un prodotto scelto dal carrello. |
| clientPrintCartContent | Per visualizzare l’intero contenuto del carrello. |
| clientEntersCheckoutQueue | Gestisce le operazioni del cliente alla coda per le casse: se il cliente non ha prodotti nel carrello, viene mandato avanti senza entrare nella coda; se il cliente è nel supermercato ed è pronto per andare alla cassa, viene inserito alla coda. |
| clientPays | Per la gestione del pagamento da parte del cliente. Se il cliente paga, lo stato del carrello viene settato a “PAYED” ed il cliente è pronto per essere accompagnato fuori dal Direttore. |

## EntranceQueue

**Descrizione**: classe per la gestione della coda di ingresso al supermercato.

**Funzionalità**:

|  |  |
| --- | --- |
| add\_client\_to\_checkout\_queue | Aggiunge un cliente alla coda. |
| remove\_client\_from\_cash\_queue | Rimuove un cliente dalla coda. |
| remove\_client\_from\_cash\_queue\_by\_id | Rimuove un cliente dalla coda dato il sui ID. |
| clients\_number\_checkout\_queue | Per ottenere il numero di clienti nella coda. |
| position\_client\_checkout\_queue | Per ottenere la posizione di un cliente all’interno della coda. |

## CheckoutQueue

**Descrizione**: classe per la gestione della coda per andare alle casse.

**Funzionalità**:

|  |  |
| --- | --- |
| add\_client\_to\_checkout\_queue | Aggiunge un cliente alla coda. |
| remove\_client\_from\_cash\_queue | Rimuove un cliente dalla coda. |
| remove\_client\_from\_cash\_queue\_by\_id | Rimuove un cliente dalla coda dato il sui ID. |
| clients\_number\_checkout\_queue | Per ottenere il numero di clienti nella coda. |
| position\_client\_checkout\_queue | Per ottenere la posizione di un cliente all’interno della coda. |

## Director

**Descrizione**: classe per il controllo delle uscite dal supermercato gestite dal Direttore.

**Funzionalità**:

|  |  |
| --- | --- |
| checkExit | Controlla periodicamente se c’è qualche carrello in stato di “CONFIRM” o “PAYED”, ossia gli stati in cui, per uscire dal supermercato, si attende solo l’approvazione del Direttore. |

# Sequence Diagram per il flusso di esecuzione

Dsf

# Simulazione del flusso di esecuzione tramite interfaccia grafica

Asdasd