CocktailShop

N86003179 Ciacciarella Alex

N86003216 De Biasio Guglielmo

N86003302 Buonanno Elvino

Sommario

Descrizione Generale

CocktailShop è un applicativo Client-Server che permette agli utenti di acquistare bevande divise in due categorie quali: Frullati e Cocktail.

È possibile registrarsi creando un account oppure accedere utilizzando un account creato in precedenza.

Il carrello dell'utente non è memorizzato in maniera persistente dunque, al log-out, tutti i prodotti al suo interno verranno rimossi e le quantità nel database non verranno alterate.

Funzionalità principali:

- **Database per la gestione degli account utente e delle bevande;**
- **Selezione delle bevande divise in categorie;**
- **Bevande consigliate in base agli acquisti;**
- **Acquisto delle bevande tramite carrello dedicato;**

Client

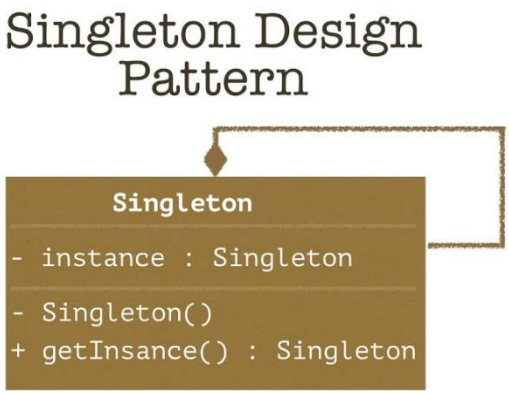
Consiste in un'applicazione Android scritta in Java.

È stato utilizzato il kit di sviluppo per Android fornito da Google: Software Development Kit (SDK), e l'interfaccia grafica è basata sul Material Design 3.

Il Client comunica con il Server tramite l'utilizzo di Socket Java e le informazioni al Carrello vengono trasmesse tramite l'utilizzo di una classe "Carrello" istanziata tramite Singleton Design Pattern, il quale garantisce la creazione di un'unica istanza dell'oggetto accessibile globalmente nel programma.

Si è deciso di usare questo sistema in quanto c'erano più punti nei quali c'era la possibilità di effettuare operazioni sugli articoli del carrello, quali l'inserimento e l'aggiornamento della quantità di un prodotto già presente.

La rimozione invece, avviene direttamente all'interno della schermata del carrello che offre anche la possibilità di modificare la quantità dei prodotti presenti al suo interno.



Le richieste di rete, svolte con le Socket, avvengono con degli appositi Thread chiamati tramite l'utilizzo dell'interfaccia "ExecutorService" , per evitare stalli al Thread principale dell'Applicazione a causa delle molteplici operazioni bloccanti.

È però necessario in alcuni casi, come ad esempio quello dell'invio delle informazioni di pagamento al server, ottenere il risultato dell'operazione e per questo si è deciso di utilizzare l'accoppiata "ExecutorService" e "Future".

Questi ultimi in particolare, sono oggetti che consentono di aspettare il Thread che sta compiendo un'operazione, per poi ottenere il risultato da essi restituito e utilizzarlo in seguito. Il catalogo viene ricevuto dal client sotto forma di stringhe composte da più campi, ognuno separato da un delimitatore.

Le stringhe possono essere di due tipologie:

- **Stringa dei Cocktail;**
- **Stringa degli Shakes;**

Entrambe hanno un campo Nome, un campo Ingredienti, un campo Prezzo ed un campo quantità, inoltra le stringhe dei Cocktail hanno un campo aggiuntivo relativo alla Gradazione Alcolica.

Le stringhe dei Cocktail hanno la seguente forma:

```
Mojito, [Rum;Lime;Zucchero;Menta], 18, 6, 10
Bloody Mary, [Vodka;Succo di pomodoro;Tabasco;Sedano;Sale;Pepe;nero;Succo di limone;Salsa Worchestershire], 25, 6, 13
White Russian, [Vodka;Liquore al caffè;Ghiaccio;Panna fresca], 25, 7, 16
```

I campi sono ordinati nel seguente modo: "Nome", "Ingredienti" ,"Gradazione Alcolica", "Prezzo", "Quantità".

Le stringhe degli Shakes hanno la seguente forma:

```
Frullato di frutta, [banana;fragola;kiwi;latte], 5, 10
Frullato tropicale, [ananas;mango;succo d'arancia;latte], 10, 10
Frullato di bacche, [fragole;mirtilli;lamponi;latte di mandorla], 7, 10
```

I campi sono ordinati nel seguente modo: "Nome", "Ingredienti", "Prezzo", "Quantità".

Le stringhe vengono convertite in Oggetti tramite delle apposite funzioni di Parsing con l'obbiettivo di estrarne i campi che verranno utilizzati successivamente.

L'operazione di pagamento, che consiste nell' invio di 2 liste di stringhe (una per i cocktail e un'altra per i frullati) per inviare le informazioni riguardo agli acquisti, è eseguita su un Thread dedicato il cui risultato viene ottenuto tramite l'utilizzo di un oggetto Future<boolean> con il quale si decide quale messaggio mostrare all'utente.

In qualsiasi caso, al termine dell'operazione di pagamento, il carrello viene svuotato peró, in caso di esito positivo, al momento del ritorno alla schermata del catalogo, esso viene aggiornato contattando il server per scaricare i dati e, successivamente, modificarli graficamente.

L'applicazione prevede una sezione dedicata alle bevande consigliate dal sistema che si basa sugli acquisti effettuati dagli utenti.

Nel caso in cui gli acquisti effettuati siano troppo pochi, il sistema non consiglierà nessuna bevanda all'utente lasciando la schermata vuota.

Server

Il Server è realizzato in Linguaggio C per poi essere eseguito in ambiente UNIX, ha lo scopo di ricevere dati e comandi dal Client per effettuare operazioni relative agli utenti e all'acquisto dei vari prodotti.

La comunicazione tra il Server ed il Client avviene tramite Socket con il protocollo TCP.

Il Server avvia la propria Socket e attende le connessioni dagli eventuali Client per ognuno dei quali viene creato un Thread specifico che gestisce lo scambio di informazioni.

Tutto quello che riguarda il Server è contenuto nel file Socket.c.

Quando viene accettata una connessione viene creato un nuovo Thread il quale chiama la funzione receive_data, questa funzione ha il compito di attendere un comando inviato dal Client.

Il Client invia dei bytes al Server, in base al quantitativo di questi bytes il server puó agire nei seguenti modi:

- **Procede alla disconnessione del Client;**
- **Effettua il Parsing della stringa ricevuta per ottenere il comando;**
- **Chiude la connessione del Client;**

Il primo caso accade quando il Server riceve 0 bytes.

Il secondo caso accade quando il Server riceve un numero di bytes maggiore di 0.

Il terzo caso accade quando la Receive (funzione che ha il compito di ricevere un buffer dal Client) restituisce -1.

Se ci troviamo nel secondo caso, cioè nell'unico caso positivo, si controlla se la stringa è vuota oppure no, se è vuota disconnette il Cliente, se invece non è vuota procede con il Parsing.

Il compito del Parsing è quello di ricevere una stringa ed estrarre il numero del comando inviato dal Client ed eventuali parametri in essa contenuti.

Ogni comando è identificato da un numero intero, e ad ogni numero ricevuto corrispondono una serie di operazioni quali:

- **Log-in;**
- **Registrazione;**
- **Download lista dei Cocktail;**
- **Download lista degli Shakes;**
- **Rimozione dei Cocktail e/o degli Shakes acquistati;**
- **Download lista dei Cocktail consigliati;**
- **Download lista degli Shakes consigliati;**

Il comando di Log-in ci permette di far accedere l'utente all'applicativo lavorando con il Database effettuando controlli sulle credenziali ricevute dal Client quali: Email e password.

Il comando di Registrazione ci permette di far registrare l'utente all'applicativo.

Vengono effettuati dei controlli sulla correttezza delle credenziali nel database e il risultato viene inviato al Server che procederà ad inoltrarlo al Client.

I 4 comandi di Download delle liste non presentano parametri, quando vengono ricevuti dal Server esso crea una stringa formattata che subirà il Parsing da parte del Cliente che la convertirà in oggetti pronti all'uso.

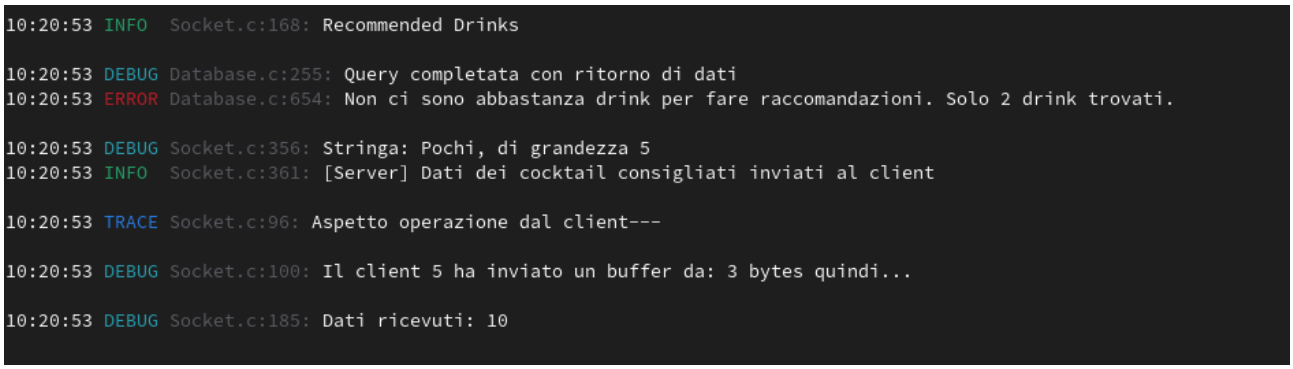
Al momento della ricezione del comando di Rimozione, il Server rimane in attesa di ricevere delle stringhe appositamente formattate contenenti le informazioni necessari per la rimozione al termine della quale invierà un comando di Fine.

Nel caso in cui il Server non riconosca il comando inviato procederà ad informare il Client della problematica riscontrata per poi tornare in attesa di un nuovo comando.

Logging

È stato implementato un sistema di Logging per identificare con facilità le eventuali problematiche nelle fasi di test e debug del Server, questo sistema si basa sulla libreria esterna log.h, la quale offre diversi livelli di informazioni permettendo di distinguerle in TRACE, DEBUG, INFO, WARN, ERROR, FATAL.

Di seguito viene mostrato un esempio di Log:



Database

Il sistema prevede l'utilizzo del Servizio di gestione di Database PostgreSQL nel quale vengono conservate le informazioni in merito ai Clienti, le informazioni in merito ai Prodotti e le informazioni in merito alle Vendite.

Il Database contiene tre tabelle che sono:

1. **Cliente;**
2. **Prodotti;**
3. **Vendite;**

La tabella Cliente è la seguente:

```
dbcocktail=# select * from cliente;
 email | password | islogged
-------+----------+----------
(0 rows)
```

La tabella Prodotti è la seguente:

```
dbcocktail=# select * from prodotti;
 prodotto_id |       nome       |   tipo   |                                  ingredienti                                  | gradazione_alcolica | prezzo | quantita
-------------+------------------+----------+-------------------------------------------------------------------------------+---------------------+--------+----------
```

La tabella Vendite è la seguente:

```
dbcocktail=# select * from vendite;
 vendita_id | utente_email | prodotto_id | quantita | tipo
------------+--------------+-------------+----------+------
(0 rows)
```

Istruzioni d'uso

SERVER

È necessaria la libreria libpq di PostgreSQL.

Bisogna installare PostgreSQL sulla propria macchina Unix e avere un compilatore quale GCC o G++.

Assicurarsi di aver modificato nel file Socket.c la costante IP con il proprio indirizzo IP locale.

La porta è sempre la porta 5978 ma è modificabile e c'è una costante apposita nella classe Socket.c.

Per compilare il server, segui questi passaggi:

1. **Posizionarsi all'interno della directory ../ProgettoLSO/ProgettoCocktailSellerApp/Server.**
2. **Aprire un terminale e lanciare lo script ./build.sh.**
3. **Lanciare il file ./main appena creato.**

CLIENT

È necessario Android Studio con SDK minimo versione 34 e un dispositivo Android con versione Android dalla 9.0 a 14.0.

Per configurare il client, segui questi passaggi:

1. **Aprire il progetto con Android Studio e modificare il proprio indirizzo IP che si trova nel file Client.java con lo stesso indirizzo del server (ovvero quello privato su cui runnerà il server).**

È possibile modificare la porta con una libera.

La modifica si fa sempre nel file Client.java