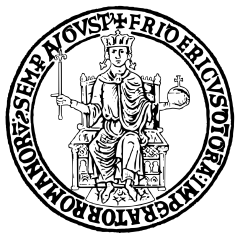
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI LABORATORIO DI SISTEMI OPERATIVI
ANNO ACCADEMICO 2022/2023

GESTIONE VENDITA BEVANDE

Autori

Gian Marco ADDATI
N86003795
gi.addati@studenti.unina.it

Simone GIORDANO
N86003660
simone.giordano5@studenti.unina.it

Docenti

Alessandra ROSSI

31 marzo 2023

Indice

Parte I

Introduzione

In questo documento presentiamo un'applicazione Android progettata per gestire la vendita di Drink in un Bar.

L'Utente potrà :

- **Registrarsi**
  L'Utente, inserendo Email, Nome, Cognome e Password, potrà registrarsi all'applicazione.

- **Autenticarsi**
  L'Utente, dopo essersi correttamente registrato, potrà accedere all'applicativo inserendo la propria email e la propria password.

- **Visualizzare le informazioni personali**
  L'Utente ha a disposizione una sezione, nella schermata di Home, in cui potrà visualizzare le informazioni del proprio account, modificare la password ed effettuare il Logout.

- **Visualizzare lo storico**
  L'Utente potrà controllare gli acquisti effettuati, potendo filtrare tra Cocktail e Frullati.

- **Visualizzare gli ingredienti di un Drink**
  L'Utente potrà visualizzare, oltre al prezzo del Drink, i suoi ingredienti.

- **Acquistare un Cocktail oppure un Frullato**
  In particolare l'Utente avrà sia la possibilità di visualizzare tutti i drink disponibili nel bar sia di poter effettuare una ricerca approfondita, scegliendo gli ingredienti che dovranno essere presenti nei drink e potendo visualizzare prima i drink già acquistati in precedenza.

- **Aggiungere un Drink al carrello**
  L'Utente potrà aggiungere un Drink al carrello per acquistarlo, insieme agli altri elementi nel carrello, in un secondo momento. Sarà possibile inoltre visualizzare il totale degli elementi nel carrello.

- **Ricevere la ricevuta per l'acquisto dei Drink**
  Nel momento in cui un utente effettua un acquisto, verrà rilasciata una ricevuta, in formato PDF, di conferma d'acquisto, scaricata sul dispositivo e dalla quale si potranno visualizzare le bevande acquistate ed il totale in Euro.

Analizzeremo dunque la Progettazione dell'Applicativo, le scelte implementative, aspetti relativi alla Base di Dati, visualizzeremo dei blocchi di codice significativi ed infine proporremo una guida all'uso.

Parte II

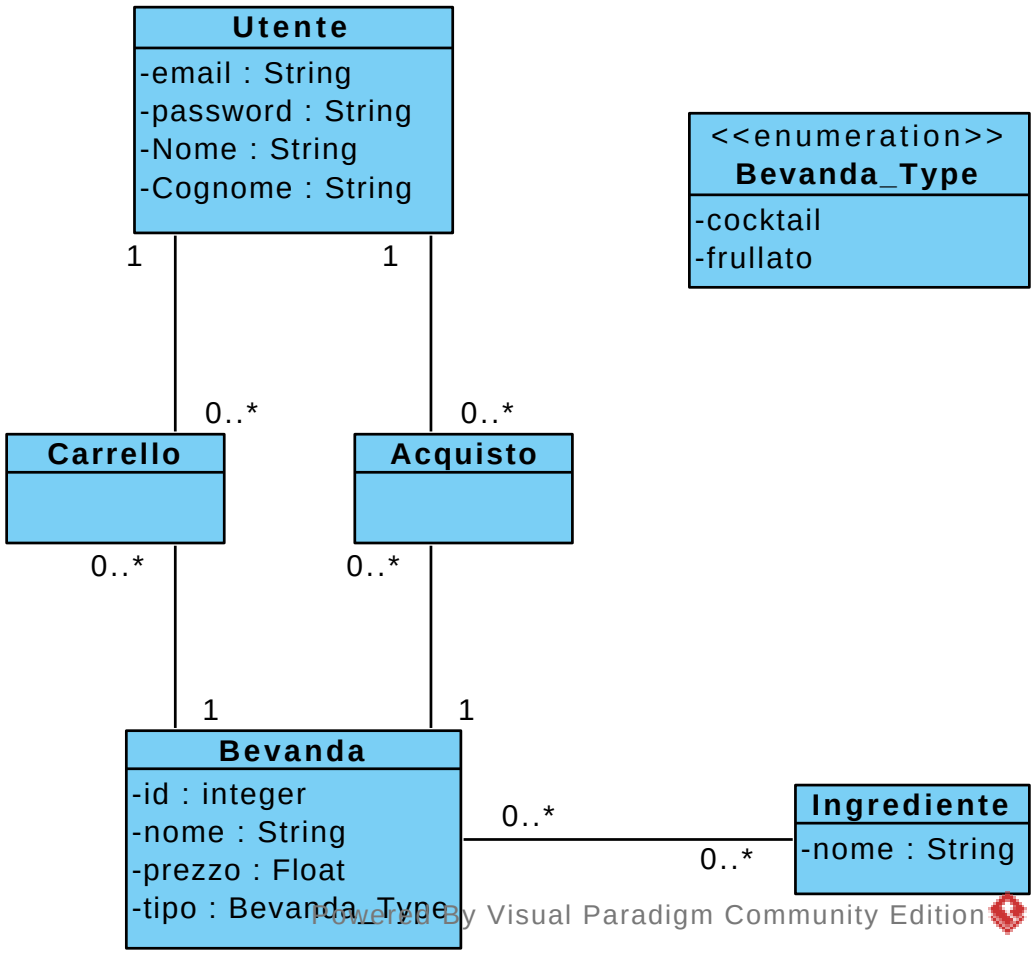
Progettazione

1 Database

I dati relativi all'applicazione saranno mantenuti su un **DataBase MYSQL**, dal quale il Server recupererà dati relativi alle bevande, agli Acquisti, agli Ingredienti e tanto altro su richiesta del Client.

Di seguito proponiamo un semplice **Class Diagram** che abbiamo progettato per realizzare successivamente la Base di Dati.
Le Tabelle individuate sono dunque :

- ACQUISTO
  Tiene traccia degli acquisti degli utenti.
- BEVANDA
  Le Bevande disponibili al bar, possono essere Cocktail o Frullati e contengono Ingredienti.
- CARRELLO
  Tabella in cui si registrano dati relativi all'aggiunta di un elemento al carrello di un utente.
- CONTIENE
  Tabella in cui sono presenti dati relativi agli Ingredienti contenuti in una determinata Bevanda.
- UTENTE
  Utenti registrati all'applicazione, si tiene traccia della loro Email e della loro Password (che serviranno in fase di Login) oltre che al loro Nome e Cognome.


Utente
-email : String
-password : String
-Nome : String
-Cognome : String
<<enumeration>>
Bevanda_Type
-cocktail
-frullato
1
1
0..*
0..*
Carrello
Acquisto
0..*
0..*
1
1
Bevanda
-id : integer
-nome : String
-prezzo : Float
-tipo : Bevanda_Type
0..*
0..*
Ingrediente
-nome : String
Powered By Visual Paradigm Community Edition

2 Architettura

L'Applicazione è disponibile su Android, e presenta un'architettura di tipo **Three Tier** :

- LIVELLO DI ARCHIVIAZIONE DEI DATI
  Composto dalla **Base di Dati**, è responsabile della memorizzazione dei dati dell'applicazione.
  Riceve una **Query SQL** dal Server, e restituisce una **Result Table**.

- LIBELLO LOGICO
  Il **Server** dell'applicazione, si occupa di ricevere una richiesta dal Client, interrogare la Base di Dati e rispondere al Client fornendo il risultato della richiesta.

- LIVELLO DI PRESENTAZIONE
  Il **Client**, responsabile dell'interfaccia utente, registra **Eventi** che inviano richieste al Server, il quale risponde con dei Dati.

Parte III

Sviluppo

L'idea di base dello sviluppo dell'applicativo era quella di realizzare un'applicazione MultiClient, quindi accessibile da più utenti contemporaneamente.

Per realizzare quest'idea ci siamo avvalsi dei **Thread**, ovvero unità di elaborazione all'interno di un processo eseguibile in maniera indipendente rispetto a quest'ultimo.

In particolare, volendo descrivere un normale flusso di utilizzo del software, i passaggi chiave sono i seguenti :

1. L'utente scatena un evento
2. L'evento è gestito dal Client, che crea un Client-Thread, apre una connessione sulla Socket e scrive la Richiesta
3. Il Server riceve la Richiesta, crea un Server-Thread per gestire questa richiesta e si rimette in attesa di una nuova richiesta
4. Il Server-Thread analizza la richiesta, estrapola i parametri, crea la Query corrispondente alla richiesta, si connette al DataBase MYSQL e lo interroga
5. Il DataBase risponde al Server-Thread con una Result Table
6. Il Server-Thread riceve una Result Table, la converte, la invia al Client-Thread e chiude la connessione
7. Il Client-Thread riceve la risposta del Server la converte a sua volta, elabora i dati e chiude la connessione

3 Client

Vediamo nello specifico cosa succede all'interno del Client, andando ad analizzare i punti 1, 2 e 7 elencati in precedenza.

- PUNTO 1 - EVENTO
  L'applicazione Android, presenta delle **View**, elementi ai quali si possono associare dei comportamenti.
  Questi comportamenti entrano in scena quando l'utente interagisce con una determinata View.

- PUNTO 2 - GESTIONE DELL'EVENTO
  Quando un Listener di una View si attiva, può generare, se previsto, la creazione di un **Thread** al fine di comunicare col Server. Infatti, nel momento in cui sarà necessario recuperare dati dal DB, il Client crea un Thread per gestire la richiesta.
  La "Richiesta-Tipo" del nostro applicativo è simile ad una **Richiesta HTTP** e contiene il metodo ed i parametri, opportunamente separati dai caratteri "$$".
  I parametri saranno composti tipicamente da **Oggetti JSON**, ottenuti da Oggetti Java e che verranno convertiti sal Server in Struct.
  La richiesta è scritta sulla **Socket** dopo aver effettuato la connessione ad essa.

- PUNTO 7 - RICEZIONE DELLA RISPOSTA Le risposte del Server al Client sono composte da vari Oggetti JSON, che vengono dunque convertiti in Oggetti Java ed utilizzati per riempire le View dell'Interfaccia Grafica.

4 Server

Vediamo nello specifico cosa succede all'interno del Server, andando ad analizzare i punti 3, 4 e 6 elencati in precedenza.

- PUNTO 3 - ACCETTAZIONE E CREAZIONE DEL THREAD
  Il server accetta la Connessione del Client e crea un Thread per gestire la richiesta arrivata.
  Una volta fatto questo, si rimette in attesa di una nuova connessione di un client.

- PUNTO 4 - ELABORAZIONE E RICHIESTA
  Il Thread legge dalla Socket la richiesta, la spacchetta, ne estrapola **Metodo** e **Parametri**, e delega ad un'altra funzione la gestione del metodo, la quale costruirà delle Struct a partire dai Parametri JSON passati dal client, si connetterà al DB MYSQL, costruirà la Query e la invierà alla Base di Dati

- PUNTO 6 - FORMAZIONE DELLA RISPOSTA
  Il Server riceve dalla Base di Dati una Result Table, la scorre, memorizza i contenuti in Strutture Dati le cui componenti verranno convertite in JSON ed inviate al Client come risposta alla richiesta originale.
  Dopo l'invio della risposta la connessione viene chiusa.

Parte IV

README

5 Compilazione ed Esecuzione del Server

Il Server, scritto in C, dovrà essere prima compilato e poi eseguito.

Per quanto riguarda la Compilazione, è presente uno **Script** nel codice sorgente che contiene i comandi necessari alla compilazione del Server, eseguibile digitando sul terminale il comando ‘`./build.sh`‘

Qualora il file non abbia i i permessi di esecuzione, occorrerà digitare, prima di eseguire lo script, il comando‘`chmod +x build.sh`‘

Per eseguire il Server invece basterà digitare sul terminale il comando ‘`./main`‘

6 Compilazione ed Esecuzione del Client

Il Client potrà essere compilato ed eseguito tranquillamente tramite un Software come Android Studio.

Occorrerà, a "Riga 17, ConnessioneController.java", impostare come "serverName" l’IP su cui il Server è in esecuzione.