Ingegneria del Software
Progetto: Negozio Online di CD

Andrea Cancellieri VR387757
Mario A. Barbara VR387794

2017

1 Introduzione

Il testo seguente rappresenta la documentazione relativa all'elaborato di Ingegneria del software, riguardante la progettazione di un negozio virtuale di CD musicali.

In particolare sono stati progettati tre differenti eseguibili, corrispondenti alle figure che possono interagire col negozio: **cliente**, **personale**, **sistema**. Per il cliente e il personale è prevista un'interfaccia grafica per gestire le operazioni possibili, il sistema invece ha il compito di comunicare con cliente e personale per svolgere funzioni di controllo e modifica, poichè è l'unico ad avere le autorizzazioni per modificare le informazioni.

I dati da gestire sono gli attributi dei CD e delle loro quantità, dei clienti registrati e delle vendite effettuate; per ognuno di questi è stato creato un file *.json*, dove le informazioni sono organizzate e possono essere modificate dal sistema o inviate al cliente e al personale.

La documentazione tratterà in ordine:

- specifiche del progetto;
- pattern di progettazione;
- pattern architetturali;
- diagrammi UML;
- design pattern;
- differenze tra progettazione e implementazione;
- test.

2 Specifiche del progetto

Si vuole progettare un sistema informativo per gestire le informazioni relative alla gestione di un negozio virtuale di CD e DVD musicali (vende solo via web).

Il negozio mette in vendita CD di diversi generi: jazz, rock, classica, latin, folk, world-music, e così via. Per ogni CD o DVD il sistema memorizza: un codice univoco, il titolo, i titoli di tutti i pezzi contenuti, eventuali fotografie della copertina, il prezzo, la data dalla quale è presente sul sito web del negozio, il musicista/band titolare, una descrizione, il genere del CD o DVD, i musicisti che vi suonano, con il dettaglio degli strumenti musicali usati. Per ogni musicista il sistema registra il nome d'arte, il genere principale, l'anno di nascita, se noto, gli strumenti che suona.

Sul sito web del negozio è illustrato il catalogo dei prodotti in vendita. Cliccando sul nome del prodotto, appare una finestra con i dettagli del prodotto stesso. I clienti possono acquistare on-line selezionando gli oggetti da mettere in un "carrello della spesa" virtuale.

Deve essere possibile visualizzare il contenuto del carrello, modificare il contenuto del carrello, togliendo alcuni articoli.

Al termine dell'acquisto va gestito il pagamento, che può avvenire con diverse modalità.

Il sistema supporta differenti ricerche: per genere, per titolare del CD o DVD, per musicista partecipante, per prezzo. Coerentemente, differenti modalità di visualizzazione, sono altresì supportate.

Ogni vendita viene registrata indicando il cliente che ha acquistato, i prodotti acquistati, il prezzo complessivo, la data di acquisto, l'ora, l'indirizzo IP del PC da cui è stato effettuato l'acquisto, la modalità di pagamento (bonifico, carta di credito, paypal) e la modalità di consegna (corriere, posta, ...).

Per ogni cliente il sistema registra: il suo codice fiscale, il nome utente (univoco) con cui si è registrato, la sua password, il nome, il cognome, la città di residenza, il numero di telefono ed eventualmente il numero di cellulare.

Per i clienti autenticati, il sistema propone pagine specializzate che mostrano suggerimenti basati sul genere dei precedenti prodotti acquistati.

Se il cliente ha fatto già 3 acquisti superiori ai 250 euro l'uno entro l'anno, il sistema gli propone sconti e consegna senza spese di spedizione.

Il personale autorizzato del negozio può inserire tutti i dati dei CD e DVD in vendita. Il personale inserisce anche il numero di pezzi a magazzino. Il sistema tiene aggiornato il numero dei pezzi a magazzino durante la vendita e avvisa il personale del negozio quando un articolo (CD o DVD) scende sotto i 2 pezzi presenti in magazzino.

3 Pattern di progettazione

La progettazione dell'elaborato ha visto l'utilizzo di un misto tra il **plan-driven** e quello **agile**.

In particolare all'inizio della progettazione, in fase di analisi, abbiamo adottato il pattern **plan-driven** per la scelta dei *modelli architetturali* da seguire e per la stesura dei *diagrammi UML*.

Dopo una prima pianificazione delle specifiche e del design generale del progetto, con l'inizio della scrittura del codice abbiamo cambiato il pattern di sviluppo a un approccio **agile**. Abbiamo quindi iniziato l'implementazione del codice e continuato la progettazione e documentazione in parallelo, modificando anche alcuni punti decisi nella fase di analisi. Seguendo la tecnica agile dell'**extreme programming**, abbiamo seguito un approccio *incrementale*, implementato piccole parti di codice alla volta, testandole singolarmente prima di passare ai task successivi. In base alle funzioni che dovevamo implementare, abbiamo applicato, quando necessari, i *design pattern* che pensavamo fossero adatti.

Come già accennato, durante lo sviluppo abbiamo modificato e aggiunto alcune parti ai *diagrammi UML* (soprattutto nei class, activity e sequence diagram), per risolvere alcune problematiche e aggiungere funzioni che non erano state considerate in fase di analisi.

Per la maggior parte dell'elaborato è stata attuata la tecnica del **pair-designing** per la progettazione e del **pair-programming** per l'implementazione del codice; ciò è stato utile per un'istantanea revisione e correzione del progetto, facilitando la fase di **refactoring** del codice.

4 Pattern Architetturali

Abbiamo scelto un'architettura di tipo **client-server** per modellare il progetto, poichè rispondeva adeguatamente alle specifiche date (basti pensare ad altre applicazioni simili come Amazon o Ebay).

Nello specifico il lato **server** è rappresentato dal *sistema*, il lato **client** è composto da due figure: *cliente* e *personale*. Il server è l'unico ad avere accesso diretto alle informazioni nei database, i client possono richiedere operazioni sui dati e l'accesso a questi comunicando col sistema tramite segnali. Il server è sempre in esecuzione ed è in attesa di tutti questi possibili segnali inviabili dal lato client.

5 Diagrammi UML

5.1 Use Case Diagram

Figura 1: Use Case Diagram Cliente-Sistema

Figura 2: Use Case Diagram Personale-Sistema

5.2 Activity Diagram

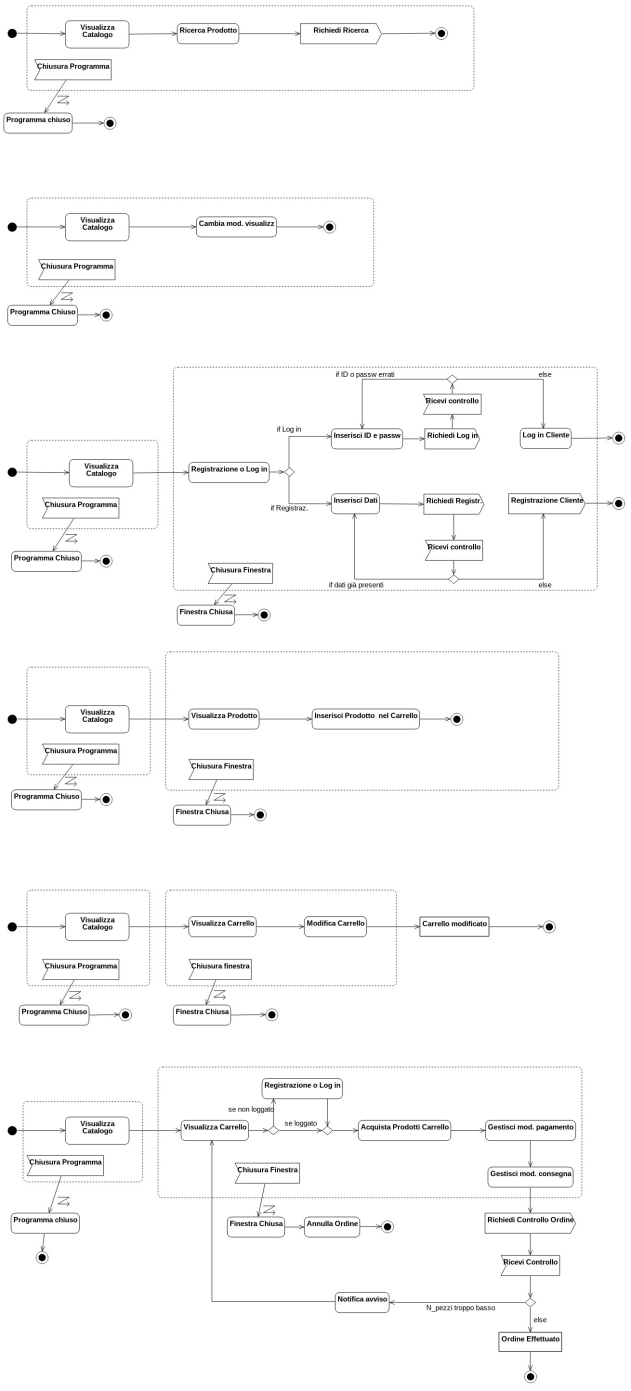

Figura 3: Activity Diagram Cliente

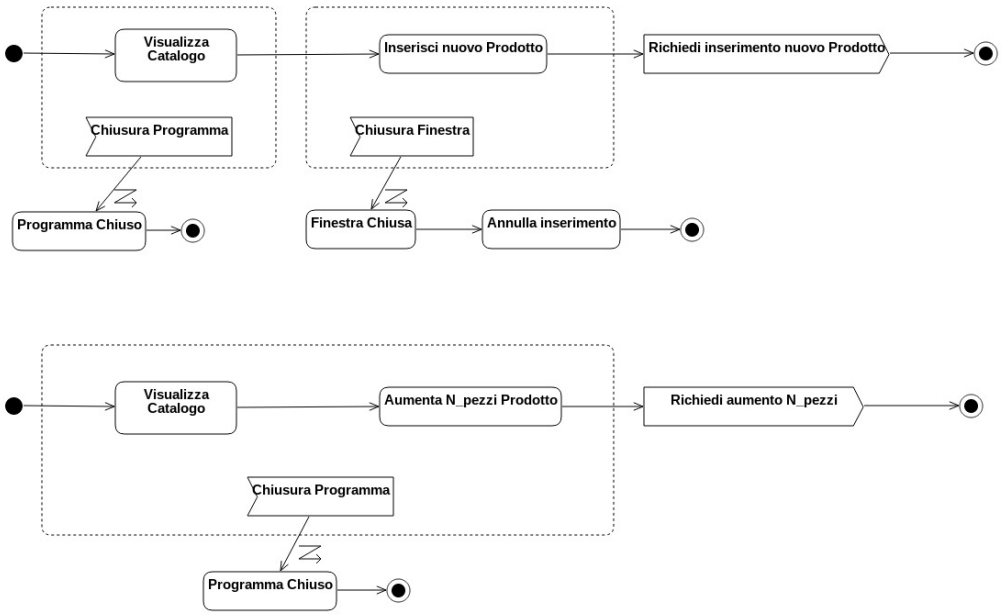

Figura 4: Activity Diagram Personale

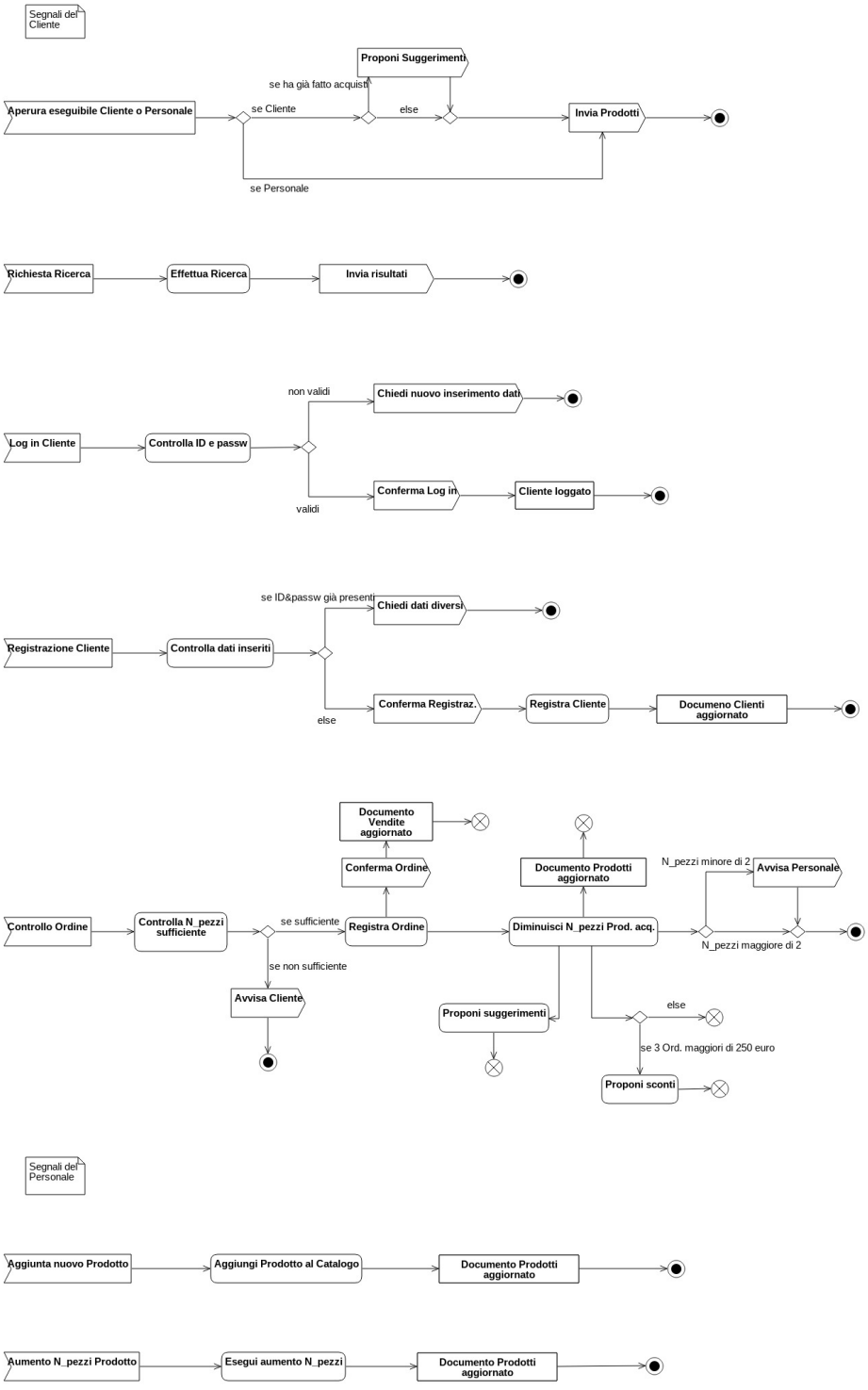

Figura 5: Activity Diagram Sistema

5.3 Class Diagram

5.3.1 Cliente

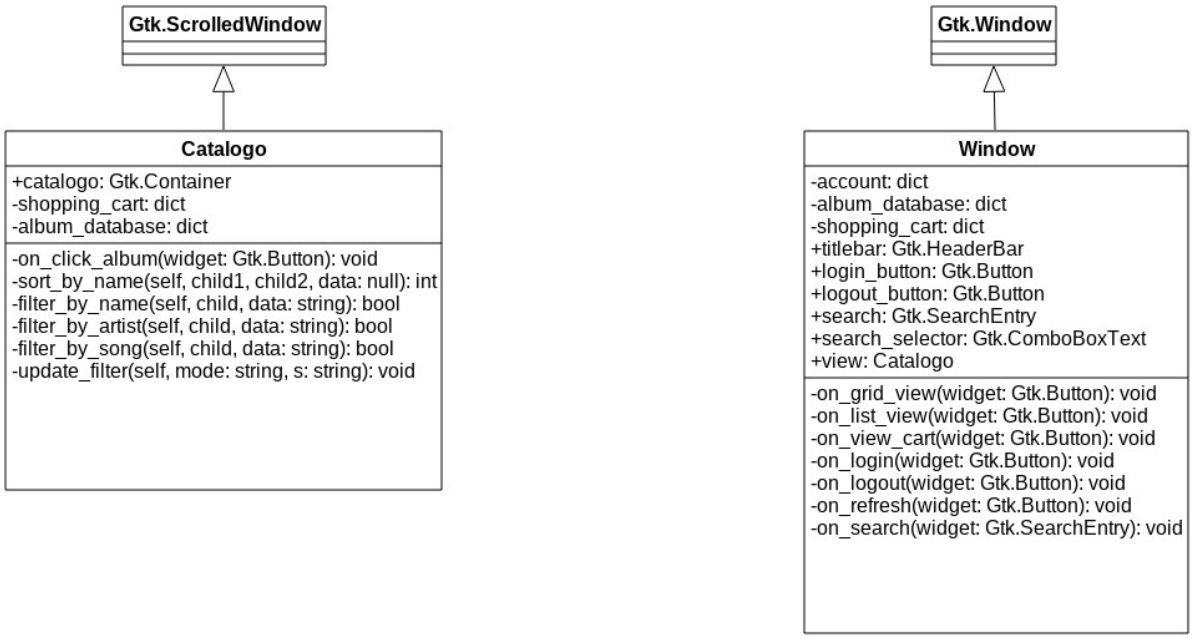

Figura 6: main.py

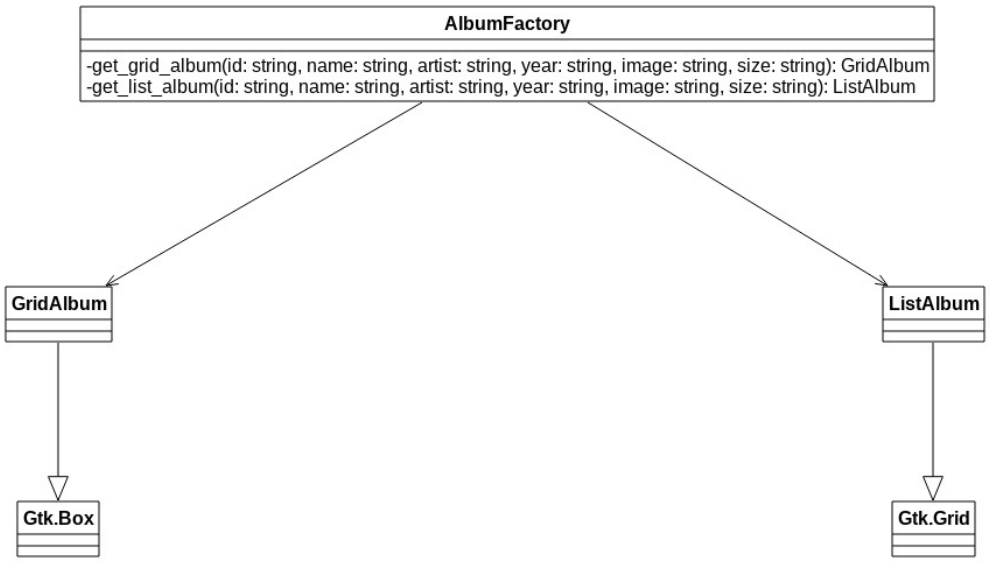

Figura 7: album.py

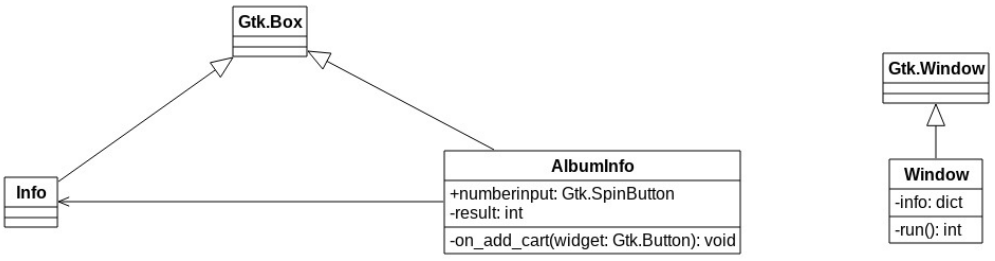

Figura 8: albumInfo.py

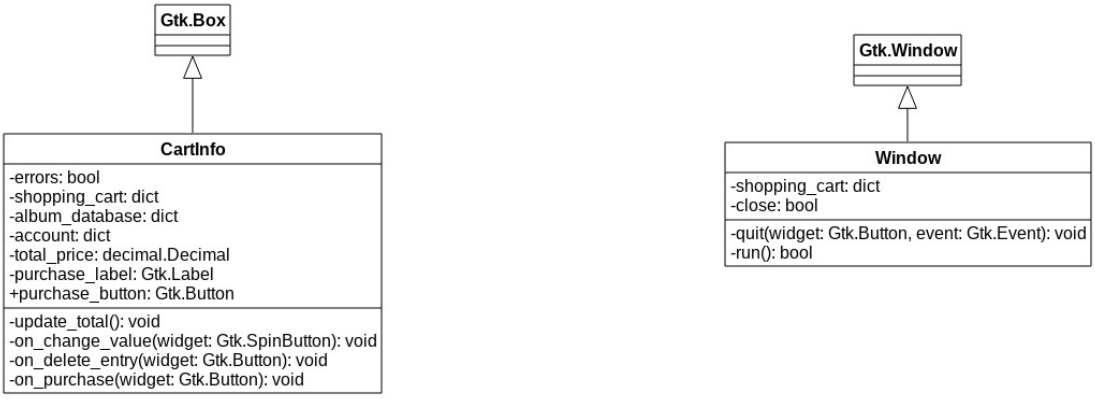

Figura 9: cart.py

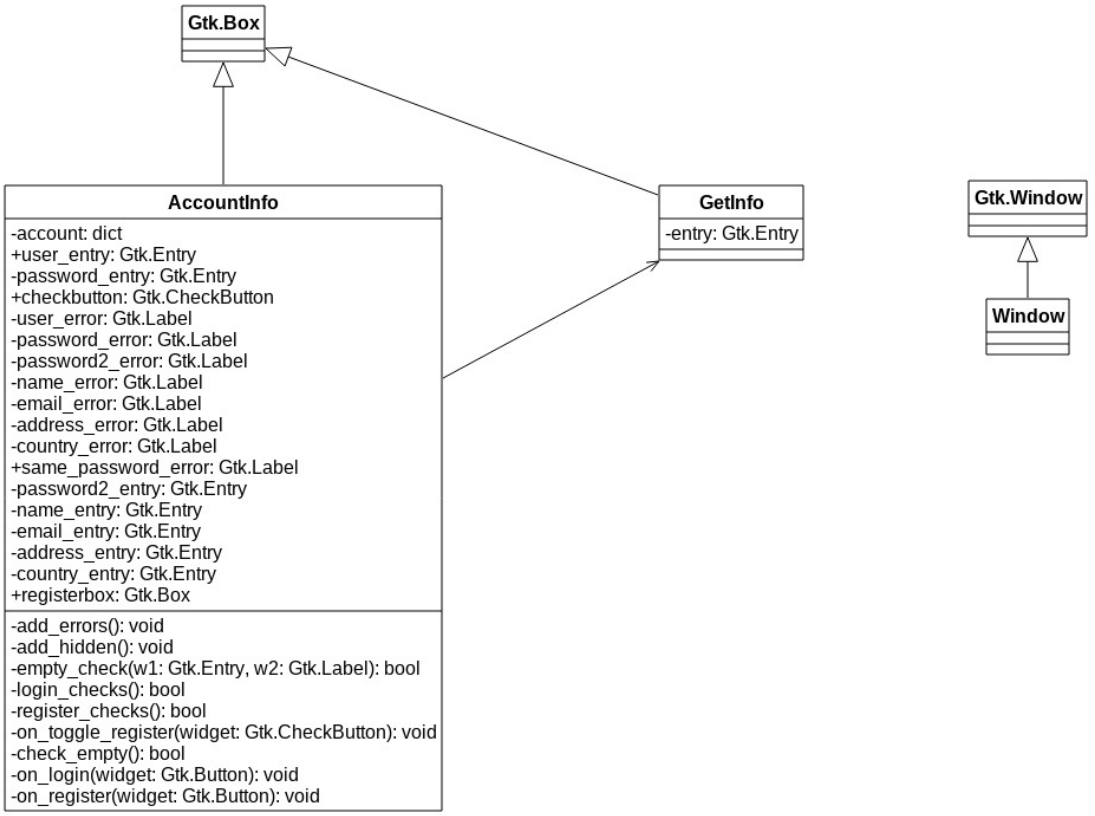

Figura 10: login.py

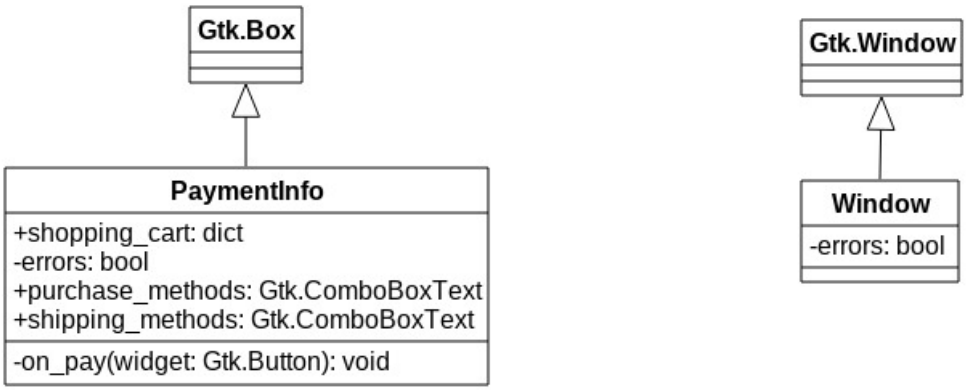

Figura 11: purchase.py

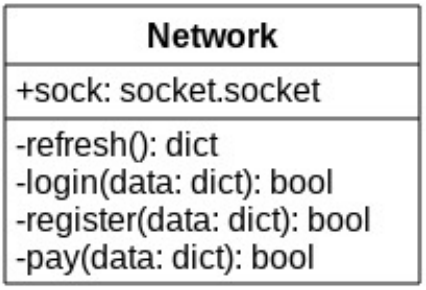

Figura 12: network.py

5.3.2 Personale

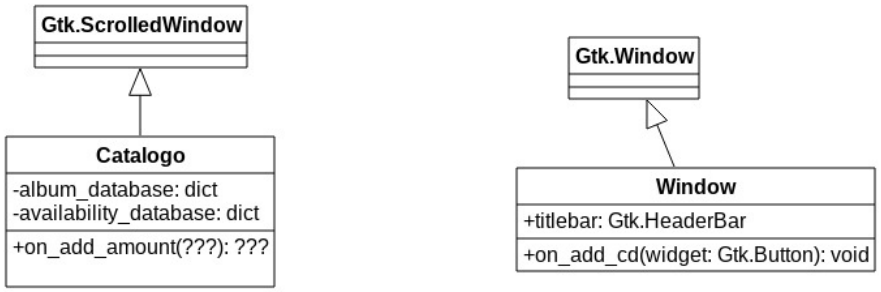

Figura 13: main.py

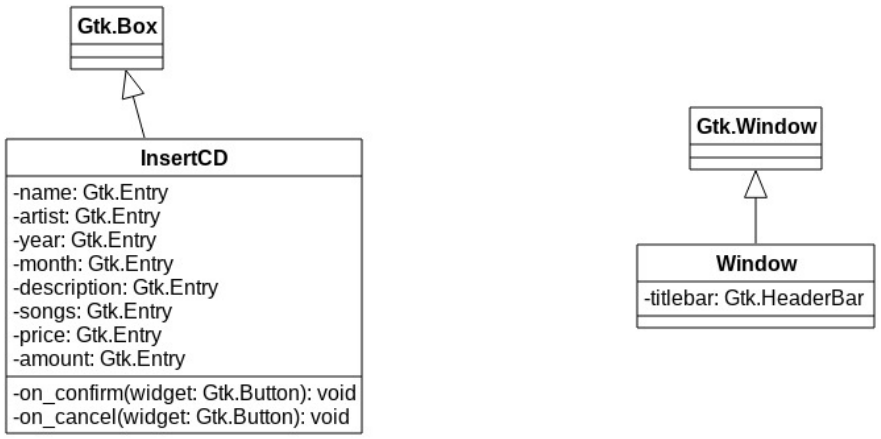

Figura 14: InsertCD.py

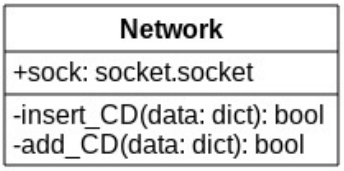

Figura 15: network.py

5.3.3 Sistema

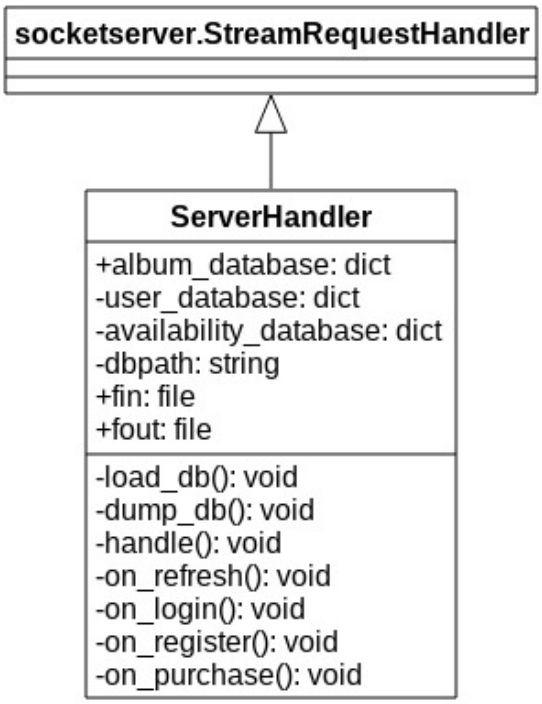

Figura 16: main.py

5.4 Sequence Diagram

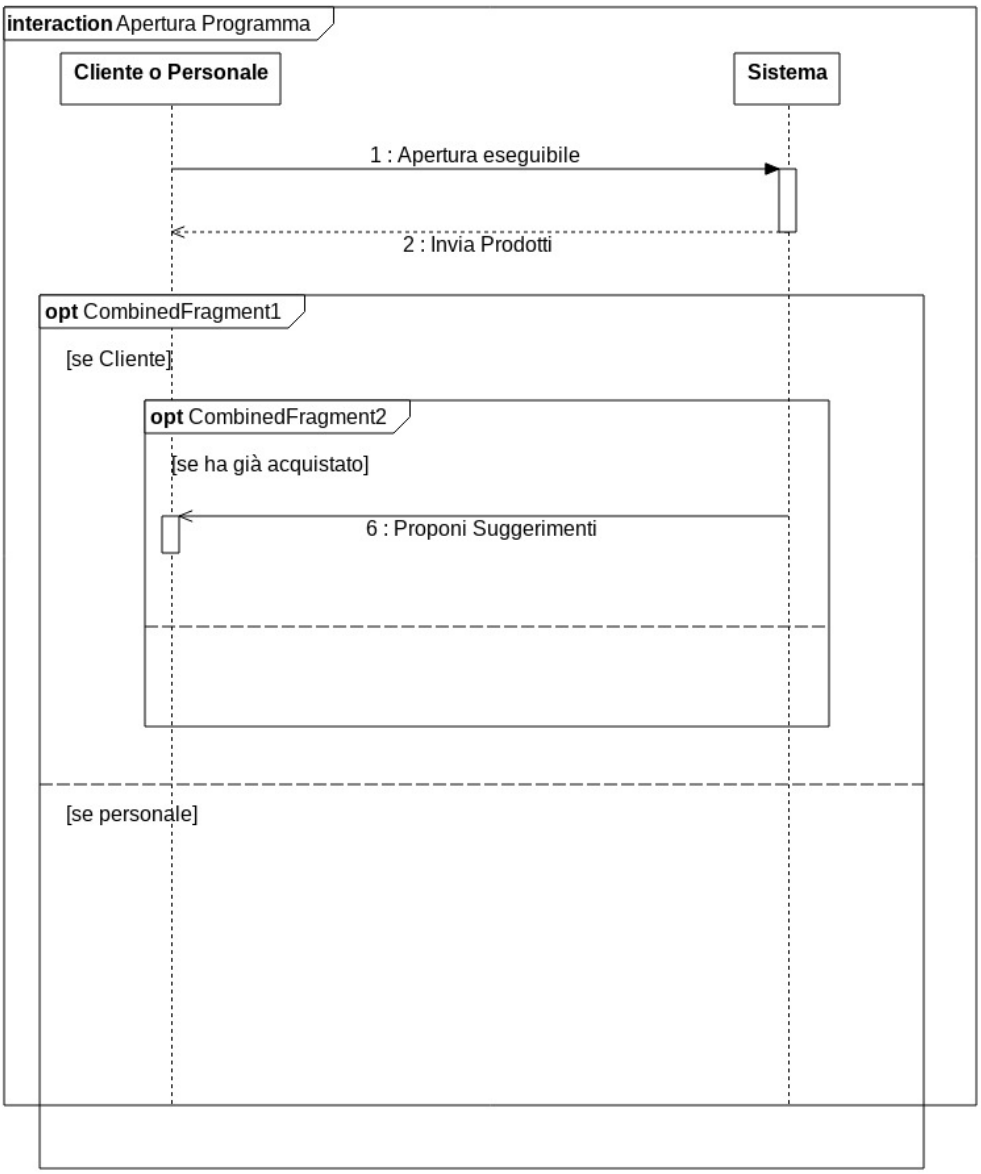

Figura 17: Apertura Programma

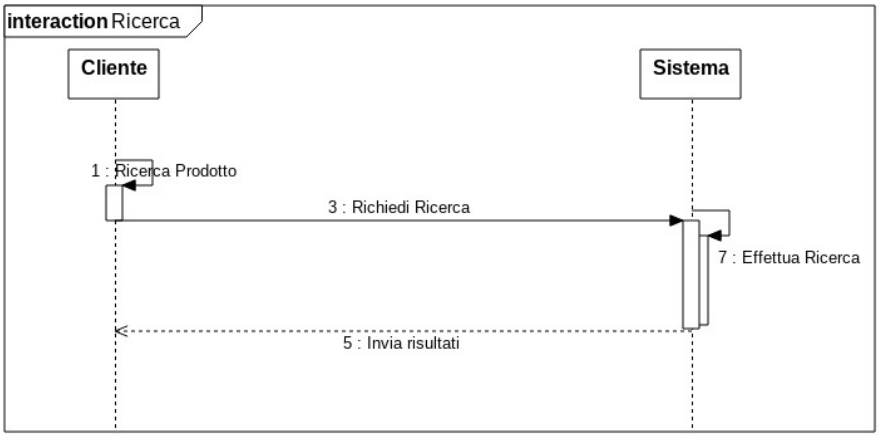

Figura 18: Ricerca

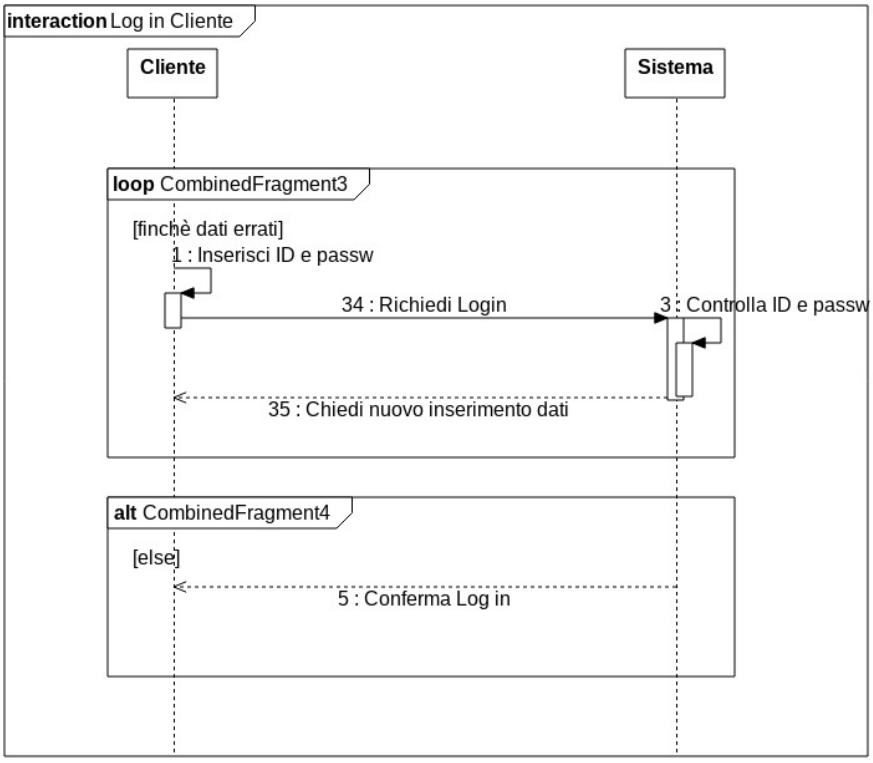

Figura 19: Log in

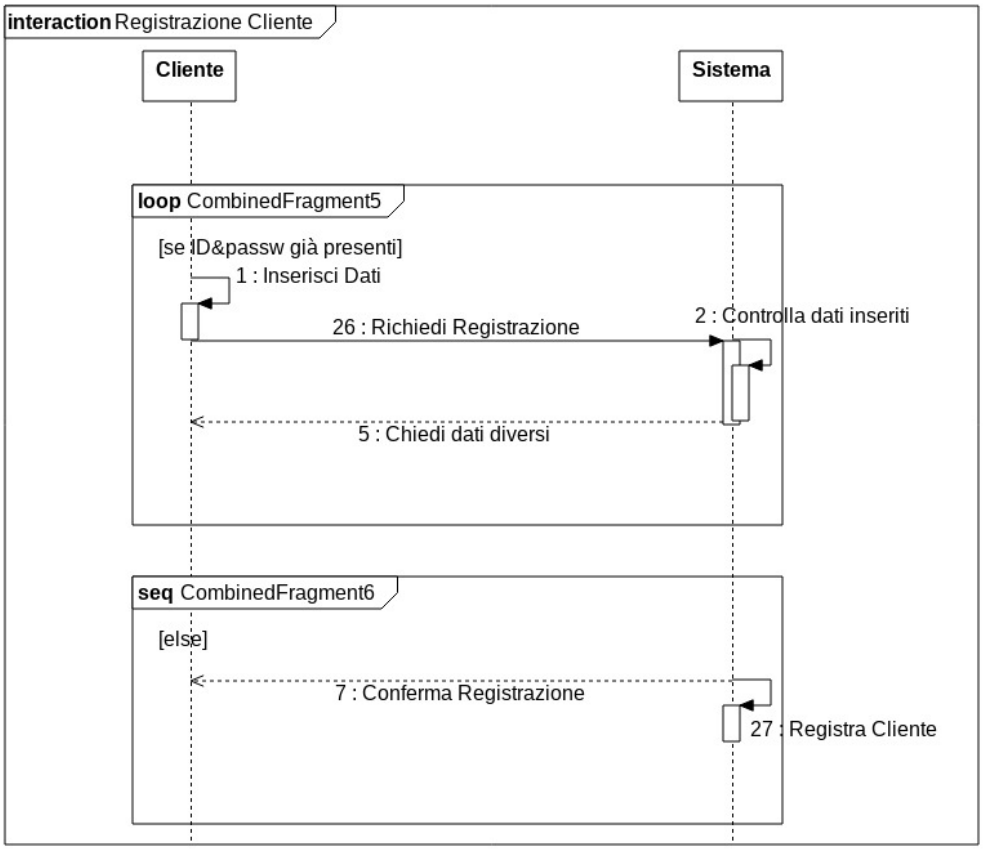

Figura 20: Registrazione

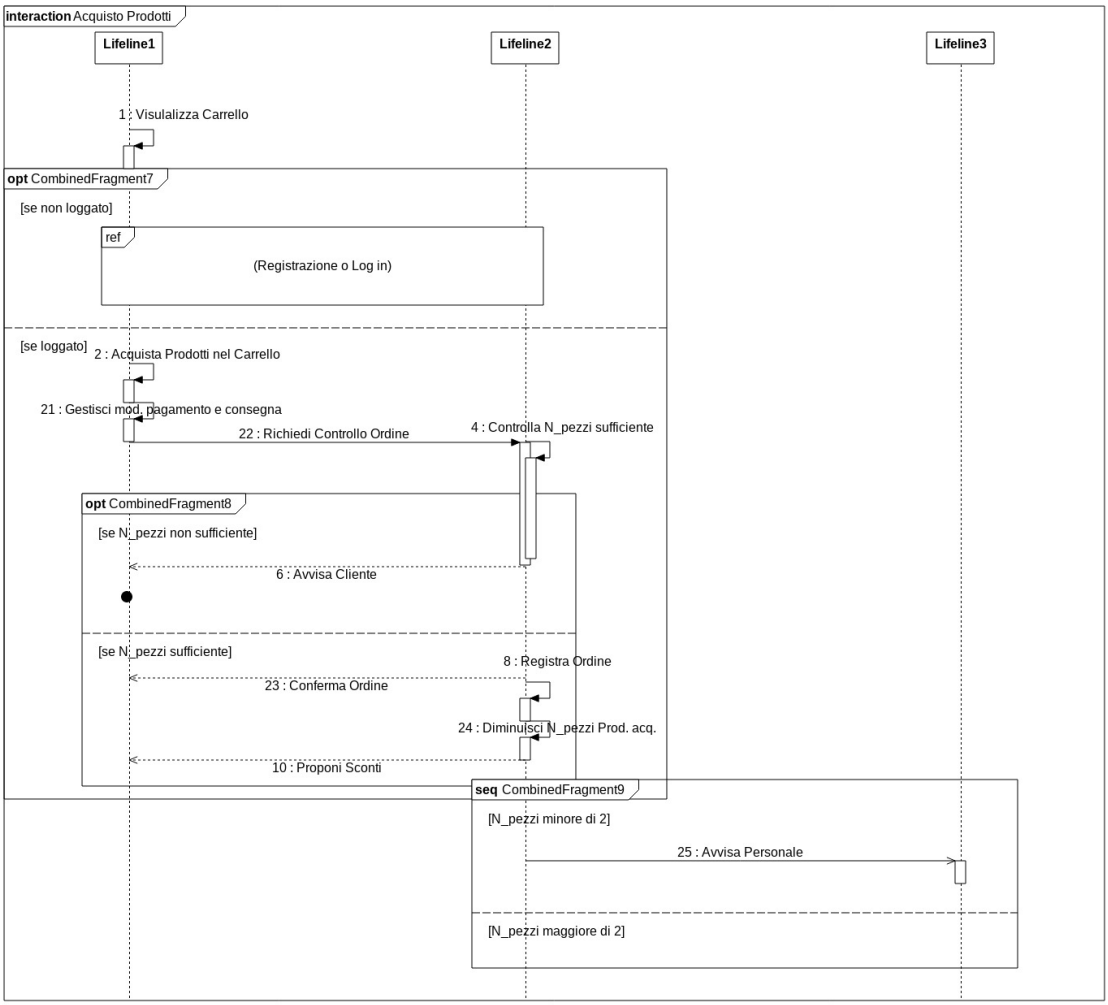

Figura 21: Acquisto Prodotto

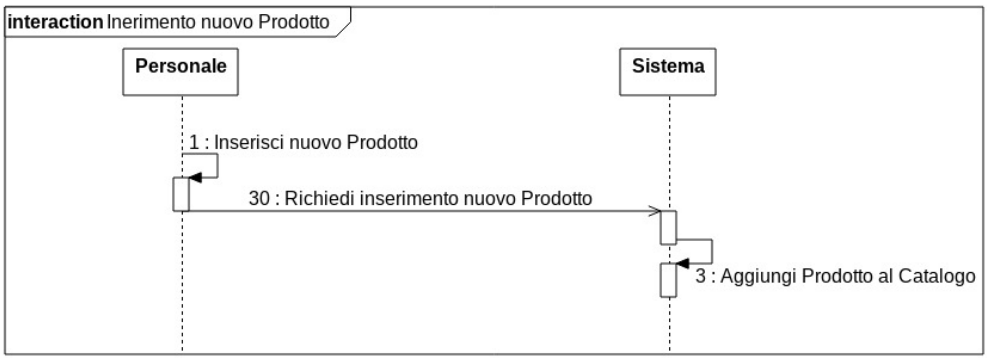

Figura 22: Inerimento nuovo Prodotto

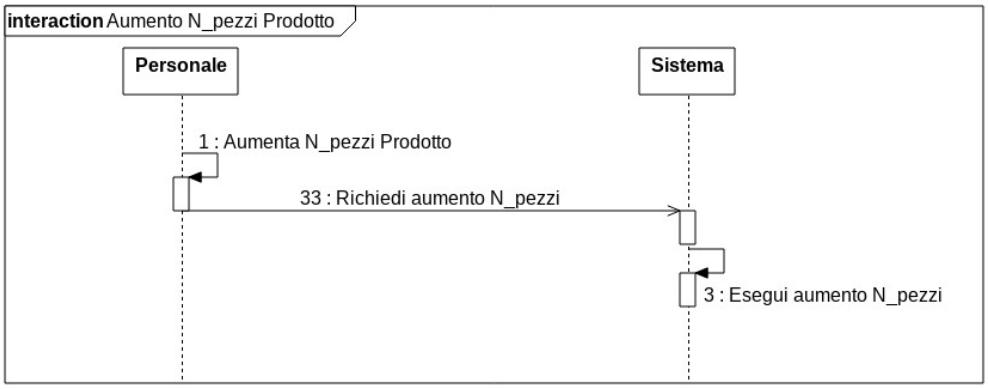

Figura 23: Aumento N-pezzi Prodotto

6 Design Pattern

Durante la scrittura del codice abbiamo utilizzato alcuni design pattern per implementare determinati comportamenti:

- **Factory pattern**:

  file: *album.py*

  Usato per distinguere la modalità di visualizzazione a griglia a quella a lista. La classe *AlbumFactory* è servita come Shapefactory, poichè in base alla richiesta ritorna (crea) le classi *GridAlbum* e *ListAlbum*.

- **Decorator pattern**:

  file: *utilities.py*

  Il file utilities contiene delle funzioni che caricano le immagini degli album e delle icone dei pulsanti. Una prima funzione carica tutte le immagini, successivamente altre due funzioni caricano rispettivamente una gli album e l'altra le icone; per implementare la funzione per le icone abbiamo utilizzato un decoratore che la estende in modo che oltre alle icone crei direttamente anche i rispettivi pulsanti.

- **Iterator pattern**:

  Usato in più file (es: *Cliente/main.py, Cliente/purchase.py*) per accedere sequenzialmente a liste di elementi come gli album nel visualizzabili nel catalogo, i CD presenti nel carrello, o quelli in un ordine d'acquisto.

7 Test

Il seguente paragrafo prende in considerazione i test più significativi effettuati alla fine dell'implementazione del codice, vengono tralasciati i test delle singole unità durante la scrittura. Vengono proposti sia test d'esecuzione per validare il programma sia test estremizzati per verificare l'esecuzione del programma con input non previsti.

- **Ricerca**:

  Cercando una stringa in una determinata categoria il programma ritorna i risultati aspettati. Il numero di caratteri inseribili nella casella di ricerca è stato fissato a un numero finito, in modo tale da evitare inserimenti troppo grandi da utenti maldestri.

- **Log in**:

  Dopo che l'utente inserisce ID e password, il programma rivela con successo se i dati sono errati o meno, determinando una successiva esecuzione corretta del codice. Anche in questo caso i caratteri inseribili nei campi sono stati resi finiti.

- **Registrazione**:

  In questo caso, oltre al limitare il numero di caratteri inseribili, era necessario che il programma controllasse l'equivalenza dei due campi password e che vietasse la registrazione di due account con lo stesso ID Questo controllo viene eseguito efficaciemente, anche se non sono stati implementati controlli avanzati in campi come addres, country, email ecc.

- **Inserimento nel Carrello**:

  L'inserimento e la visualizzazione di più CD nel carrello viene eseguita con successo. Al momento dell'inserimento di un CD, il numero di pezzi acquistabili può essere scelto tramite un widget con valori predefiniti e con un limite finito, per evitare inserimenti spropositati dagli utenti. Inoltre, se un utente inserisce per due volte lo stesso CD nel carrello, la seconda quantità inserita viene sommata alla prima già presente; questo risolve possibili problemi nell'acquisto.

- **Acquisto**:

  Una normale simulazione di acquisto viene eseguita con successo, chiedendo Log in/Registrazione se necessario ecc. Sono stati inseriti nel database alcuni album con 0 disponibilità, se si cerca di acquistarli il programma impedisce l'ordine come previsto, anche nel caso in cui viene ordinata una quantità superiore alla disponibilità.