03FYZPL TECNICHE DI PROGRAMMAZIONE

Istruzioni per effettuare il fork di un repository GitHub

- Effettuare il login su GitHub utilizzando il proprio username e password.
- Aprire il repository su GitHub relativo al quarto laboratorio:

  https://github.com/TdP-2024/Lab06
- Utilizzare il pulsante *Fork* in alto a destra per creare una propria copia del progetto. L'azione di Fork crea un nuovo repository nel proprio account GitHub con una copia dei file necessari per l'esecuzione del laboratorio.
- Aprire Pycharm, assicurandosi che eventuali precedenti progetti siano chiusi, selezionare *Get From VCS.* Utilizzare la URL del **proprio** repository che si vuole clonare (**non** quello in TdP-2024!), ad esempio:
  https://github.com/my-github-username/Lab06
- Selezionare la cartella di destinazione (quella proposta va bene), fare click su *Clone*.
- Il nuovo progetto è stato clonato ed è possibile iniziare a lavorare.
- A fine lavoro ricordarsi di effettuare Git commit e push, utilizzando l'apposito menù.

ATTENZIONE: solo se si effettua Git **commit** e successivamente Git **push** le modifiche locali saranno propagate sui server GitHub e saranno quindi accessibili da altri PC e dagli utenti che ne hanno visibilità.

03FYZPL TECNICHE DI PROGRAMMAZIONE

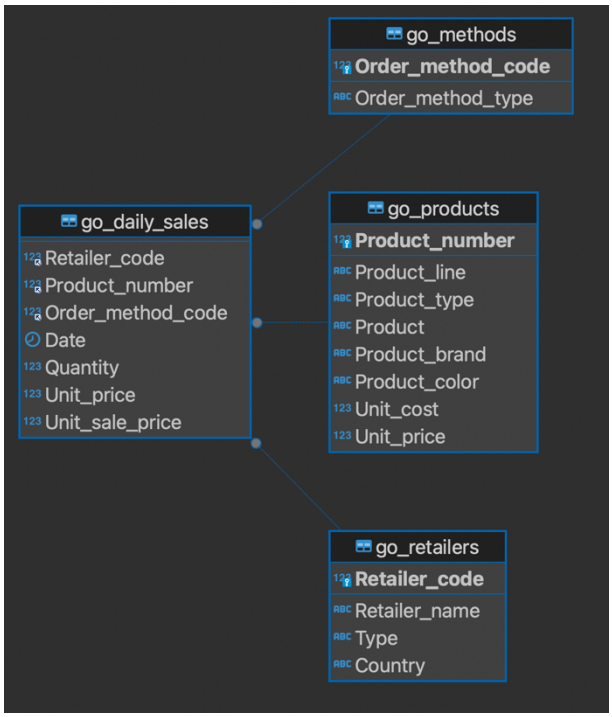
Esercitazione di Laboratorio 09/10 Aprile 2024

Obiettivi dell'esercitazione:

- Utilizzo del Pattern MVC
- Utilizzo di mysql-connector-python
- Utilizzo del Pattern DAO

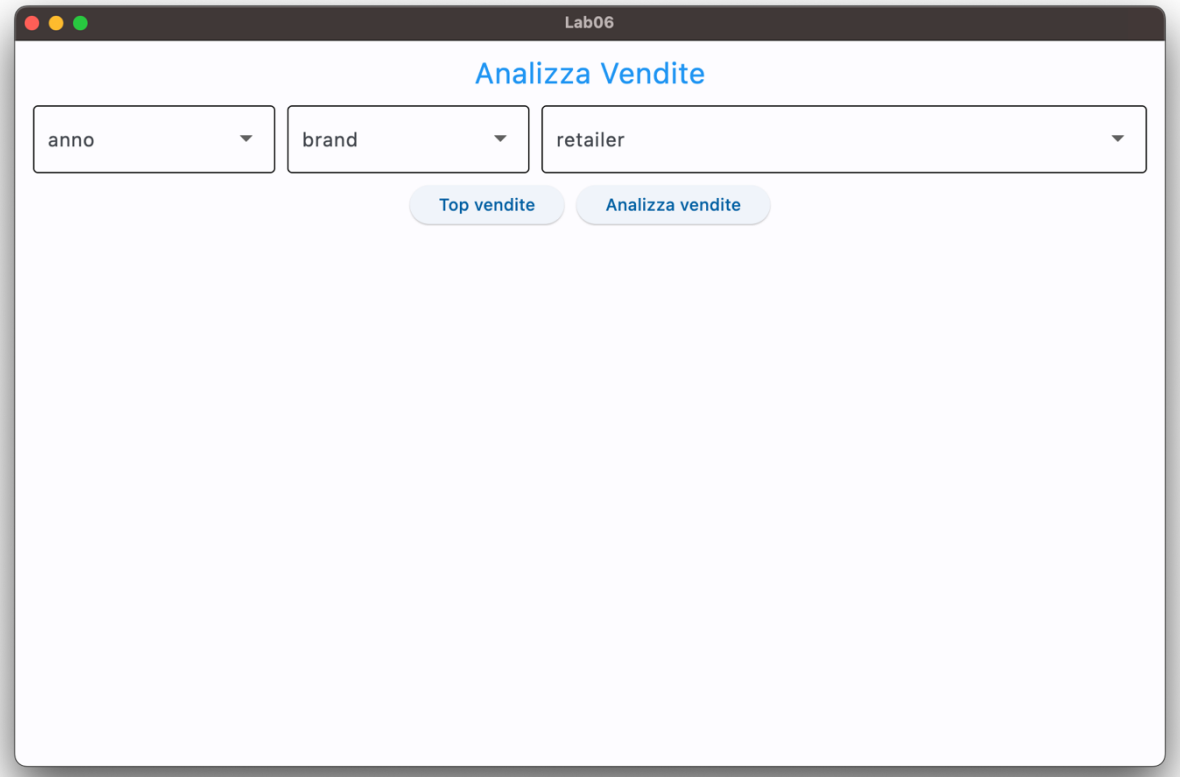
Prima di iniziare

Nella cartella del progetto è presente lo script *go_sales.sql* . Occorre eseguire questo script in DBeaver per importare il database. Il diagramme ER del database è riportato di seguito:



Dopo aver fatto il fork del progetto relativo al laboratorio, realizzare in linguaggio Python un'applicazione dotata di interfaccia grafica per analizzare le vendite, con la possibilità di filtrarle per *anno*, *brand*, *retailer*. I filtri possono essere impostati dall'utente tramite appositi menu a tendina. L'interfaccia grafica viene già fornita ma bisogna implementarne il controller

Di seguito si mostra l'interfaccia grafica



Fare uso dei patterns **MVC** e **DAO**, usando i pacchetti **flet** e **mysql-connector-python**, come spiegato a lezione.

Traccia

Punto 1

Come primo punto, bisogna popolare i tre menu a tendina. Tutti i menu contengono già una opzione "Nessun filtro". Occorre aggiungervi le altre opzioni, partendo dai dati del database.

Per il menu degli anni, bisogna aggiungere tutti gli anni delle vendite presenti nel database. Si ricorda, che in SQL si può estrarre l'anno da una colonna di tipo date utilizzando il comando `YEAR()` https://www.w3schools.com/sql/func_sqlserver_year.asp . In questo caso, occorre utilizzarlo sulla colonna `Date` della tabella `go_daily_sales`.

Per il menu brand, bisogna leggerlo dai prodotti presenti nel database, nella tabella `go_products`.

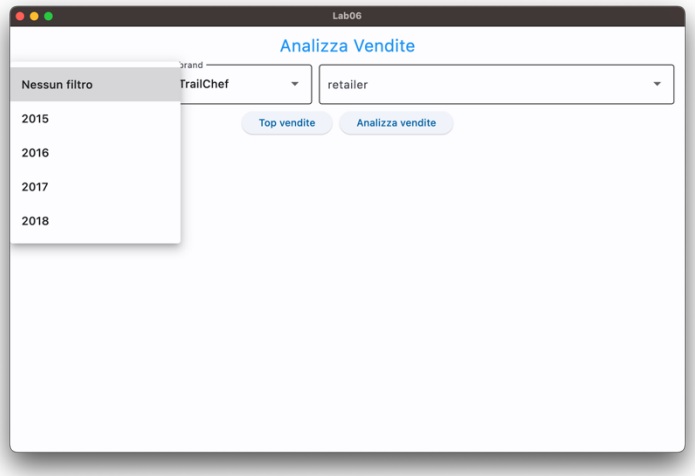
Per il menu retailer, bisogna compilarlo con tutti i retailers presenti nella tabella `go_products`.

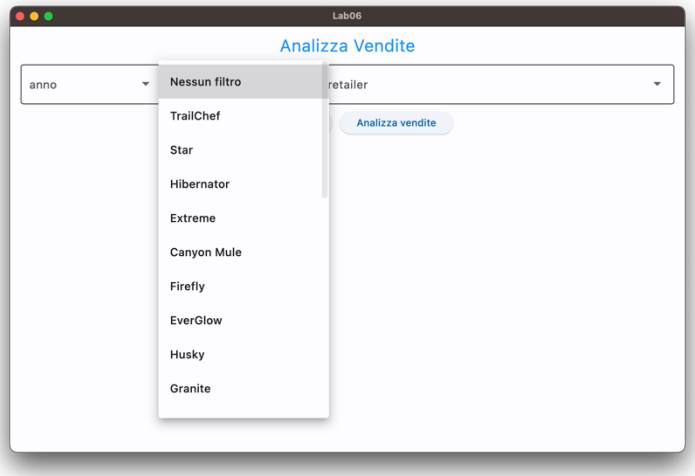
Nel precedente laboratorio, abbiamo visto come sia possibile popolare una tendina con stringhe. In realtà, si può anche popolare con oggetti (ad esempio oggetti di tipo Retailer) e si possono leggere da essa gli oggetti stessi. Per fare questo si può fare una cosa del tipo:

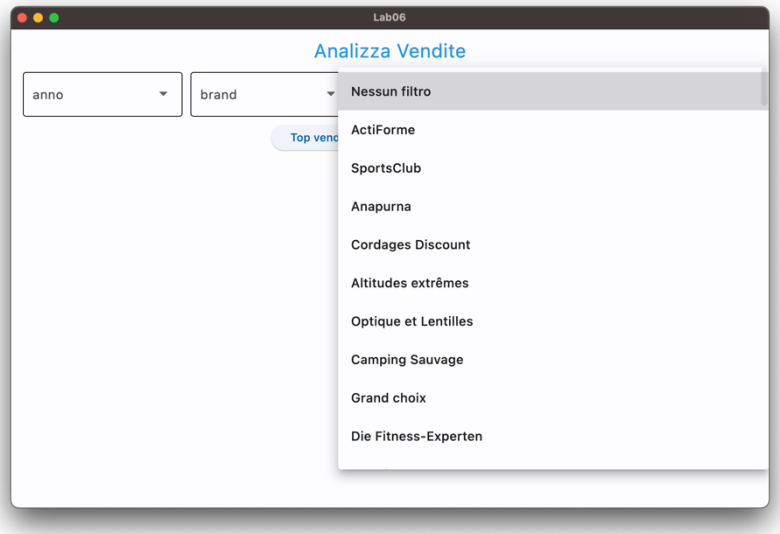
Assumendo che retailer sia un oggetto, possiamo fare
`self._view.dd_retailer.options.append(ft.dropdown.Option(key=retailer.retailer_code, text=retailer.retailer_name, data=retailer, on_click=self.read_retailer))`
In questo modo, l'oggetto è disponibile nel campo data dell'opzione. Per poterlo leggere però, bisogna usare un event handler di tipo `on_click` sull'opzione stessa.

Questo handler, può avere una forma del tipo:
```
def read_retailer(self, e):
    retailer = e.control.data
```

Per il menu retailers, si consiglia di provare a popolarlo con oggetti.

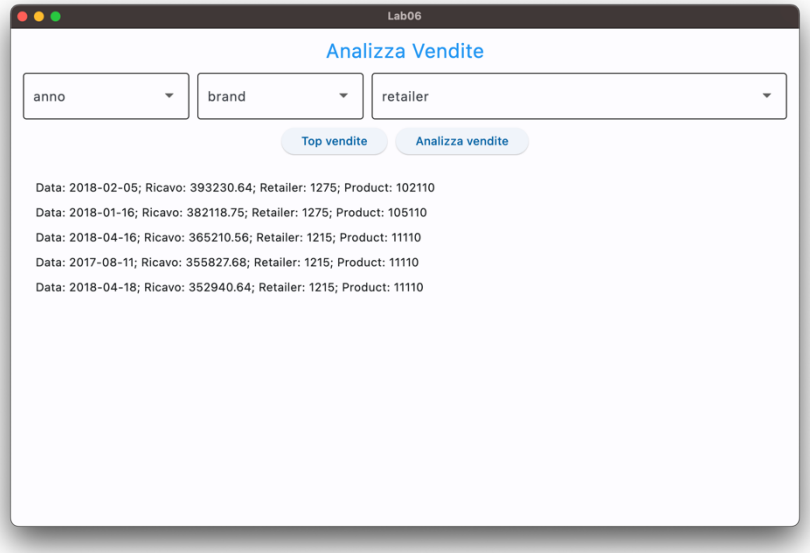


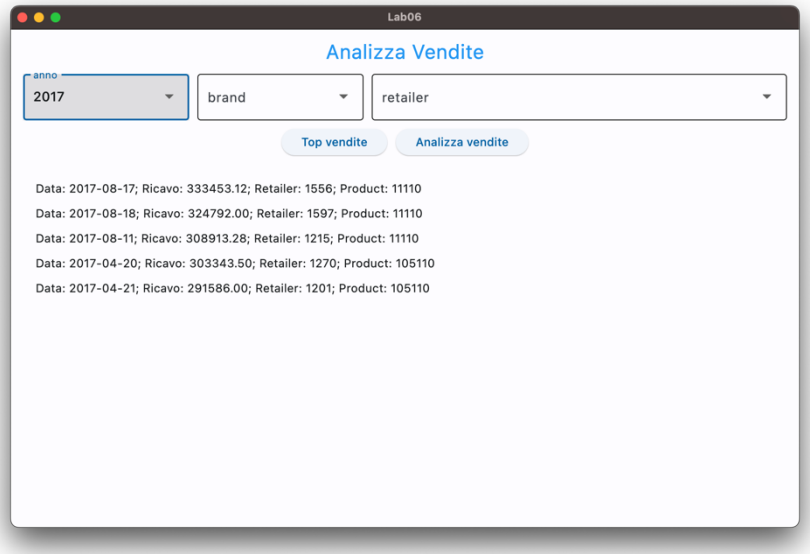


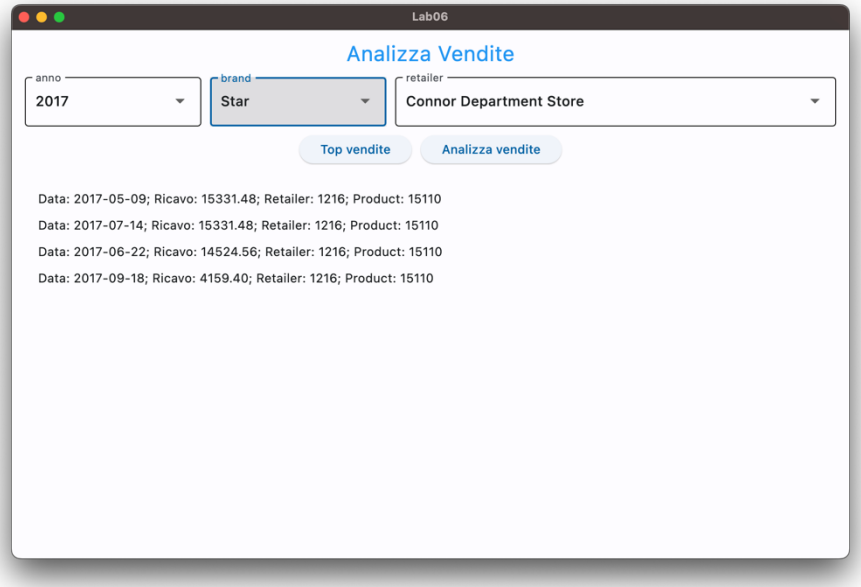


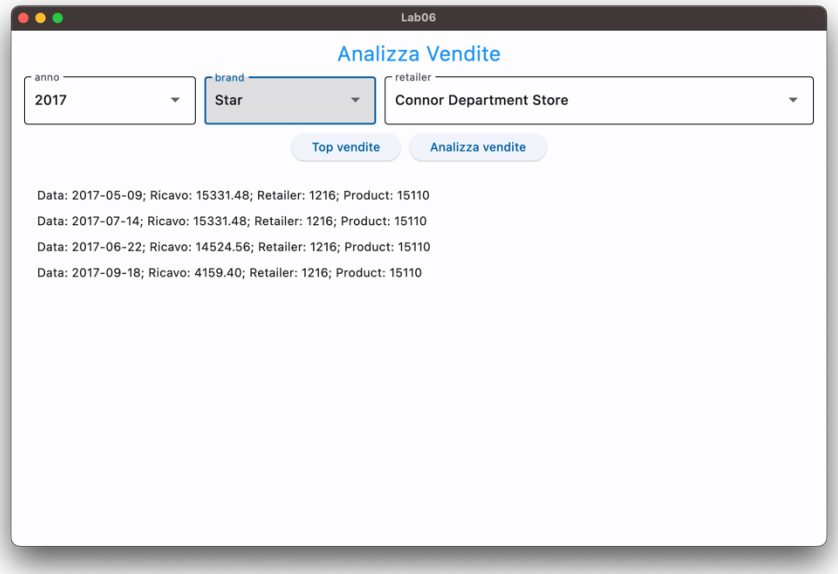
Punto 2

Implementare la funzionalità_ **Top vendite**, che ha come scopo quello di stampare (al massimo) le migliori 5 vendite del database per cui *l'anno*, il *brand del prodotto venduto* e il *retailer* che ha effettuato la vendita. Le migliori vendite sono stabilite in base al **ricavo**, dato come `Unit_sale_price * Quantity` (vedere la tabella `go_daily_sales` del database).
Il sorting delle vendite deve essere fatto in senso decrescente di ricavo.









Punto 3 Implementare la **Analizza vendite**: come al punto 2, questa funzione considera solamente le vendite che soddisfano i filtri inseriti dall'utente. Deve stampare su schermo le seguenti statistiche:

- Giro d'affari (ovvero volume totale dei ricavi)
- Numero di vendite
- Numero dei retailers coinvolti

- Numero di prodotti coinvolti