03FYZPL TECNICHE DI PROGRAMMAZIONE
Esercitazione di Laboratorio – 8/9 Aprile 2025

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

  https://github.com/TdP-2025/Lab07

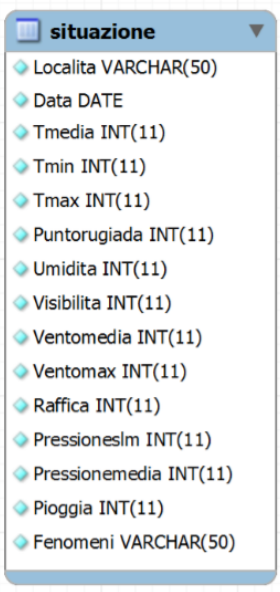
1. Utilizzare il pulsante *Fork* in alto a destra per creare una propria copia del progetto. L'azione di Fork crea un nuovo repository nel proprio account GitHub con una copia dei file necessari per l'esecuzione del laboratorio.

- 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-2025!), ad esempio:
  https://github.com/my-github-username/Lab07
- 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.

Estratto e semplificato dal tema d'esame del 03/07/2014

Il noto sito di previsioni meteorologiche "ilMeteo" (http://www.ilmeteo.it/) mette a disposizione, gratuitamente, un archivio storico[1] delle principali variabili climatiche nelle città italiane, con cadenza quotidiana.
Le informazioni relative alla situazione meteorologica sono rappresentate nella base dati avente la struttura schematizzata a fianco. Il database è denominato 'meteo' e contiene un'unica tabella, chiamata 'situazione'.



Le informazioni presenti nella base dati fornita sono relative alle città di *Torino*, *Milano* e *Genova*, e coprono tutto e solo l'anno 2013.

Si intende costruire un'applicazione in Python che permetta di interrogare tale base dati, e calcolare informazioni a proposito dell'andamento climatico.

L'applicazione dovrà svolgere le seguenti funzioni:

1. Permettere all'utente di scegliere un mese dell'anno (valore intero tra 1 e 12) e visualizzare il valore dell'umidità media per quel mese in ciascuna delle città presenti nel database.

[1] https://www.ilmeteo.it/portale/archivio-meteo/

2. Risolvere il seguente problema di ottimizzazione **mediante un algoritmo ricorsivo**:

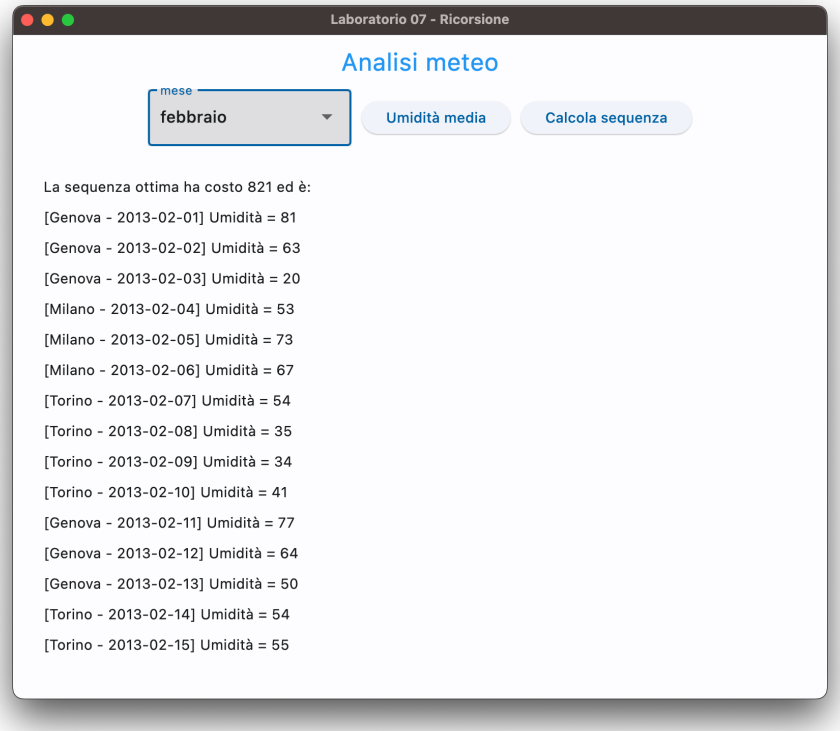
Sapendo che nel database sono presenti 3 città (Milano, Torino, Genova), supponiamo che un tecnico debba compiere delle analisi tecniche della durata di un giorno in ciascuna città. Le analisi hanno un **costo per ogni giornata**, determinato dalla somma di due contributi: un fattore costante (di valore 100) ogniqualvolta il tecnico si deve spostare da una città ad un'altra in due giorni successivi, ed un fattore variabile pari al valore numerico dell'umidità della città nel giorno considerato. Si trovi la sequenza delle città da visitare nei primi 15 giorni del mese selezionato, tale da minimizzare il costo complessivo rispettando i seguenti vincoli:

- In nessuna città si possono trascorrere più di 6 giornate (anche non consecutive)
- Scelta una città, il tecnico non si può spostare prima di aver trascorso 3 giorni consecutivi.

L'applicazione va sviluppata seguendo il pattern MVC e il pattern DAO per l'accesso al database.

Al fine di semplificare lo svolgimento, alcune classi e l'interfaccia grafica sono già fornite.

Esempio di soluzione



Per ragionare sulla ricorsione

Utilizzare lo schema seguente per pensare come impostare la ricorsione. Può esser utile ragionare su carta per capire come impostare l'algoritmo

```
def recursion(…, level):
    // E – instructions that should be always executed (rarely needed)
    do_always(…)

    // A
    if terminal_condition:
        do_something(…)
        return …

    for … //a loop, if needed
        //B
        compute_partial()

        if filtro: //C
            recursion(…, level+1)

        //D
        back_tracking
```

- ○