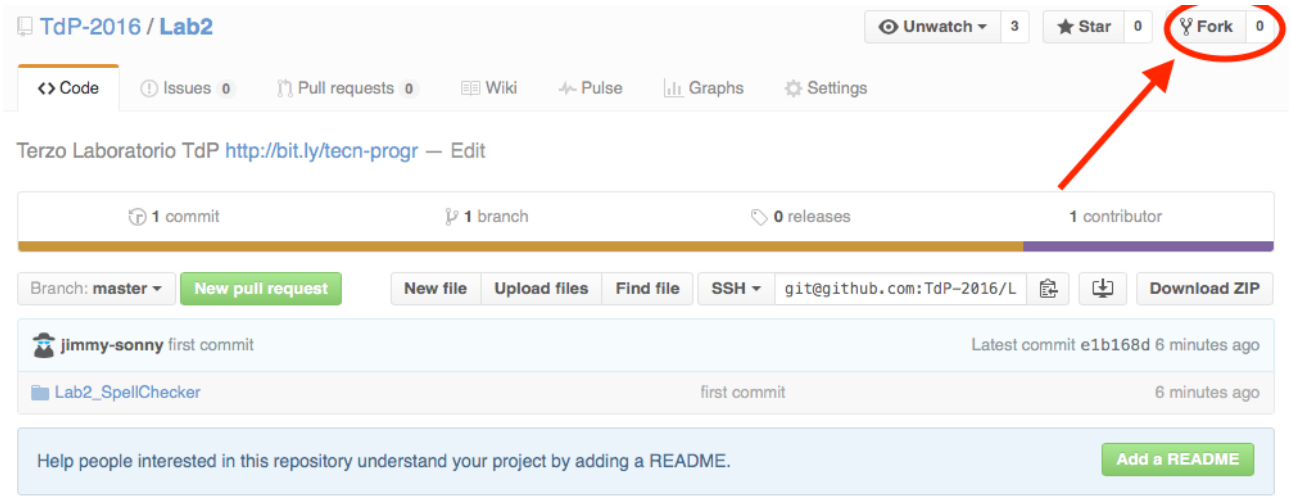
I03FYZ TECNICHE DI PROGRAMMAZIONE
Istruzioni per effettuare il fork di un repository GitHub

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

  https://github.com/TdP-2016/Lab11
- 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 accout GitHub con una copia dei file necessari per l'esecuzione del laboratorio.
- Aprire Eclipse, andare su *File -> Import*. Digitare *Git* e selezionare *Projects from Git -> Next -> Clone URI -> Next*.
- Utilizzare la URL del **proprio** repository che si vuole clonare (**non** quello in TdP-2016!), ad esempio:

  https://github.com/my-github-username/Lab11
- Fare click su *Next*. Selezionare il branch (*master* è quello di default) fare click su *Next*.
- Selezionare la cartella di destinazione (quella proposta va bene), fare click su *Next*.
- Selezionare *Import existing Eclipse projects*, fare click su *Next* e successivamente su *Finish*.
- Il nuovo progetto Eclipse è stato clonato ed è possibile iniziare a lavorare.
- A fine lavoro ricordarsi di effettuare Git commit e push, utilizzando il menù *Team in Eclipse*.

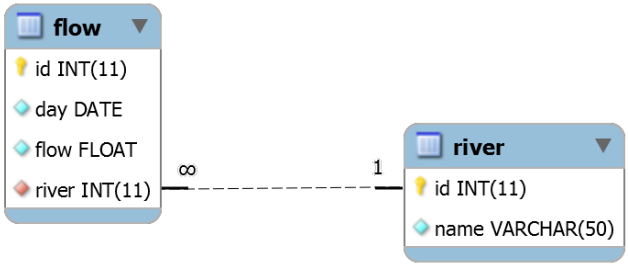
  ATTENZIONE: solo se si effettua Git **commit** e successivamente Git **push** le modifiche locali saranno propagate sui server GitHub e saranno quindi accessibili da altri PC e dagli utenti che ne hanno visibilità.

ESERCIZIO 1:

Si consideri il data-set **rivers.sql** [GitHub - DB/rivers.sql] ottenuto combinando un sottoinsieme delle informazioni presenti nella *Time Series Data Library*[1] (TSDL). Tale data-set contiene la portata dei flussi di acqua di alcuni fiumi, misurati quotidianamente per diversi anni. Notare che i diversi fiumi sono stati misurati in intervalli di tempo indipendenti tra loro. Si noti anche che le portate dei fiumi sono molto diverse tra loro, di vari ordini di grandezza.

La struttura del database è mostrata in *Fig 1*.

Fig 1

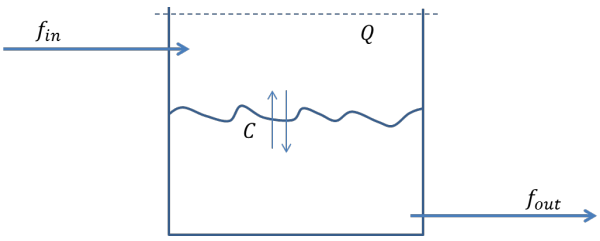


La tabella 'river' contiene i fiumi considerati nel data set, ciascuno rappresentato da un identificatore numerico 'id' e da una stringa descrittiva 'name'.

La tabella 'flow' contiene le misurazioni effettuate, nelle date specificate dal campo 'day' e per il fiume identificato dal campo (chiave esterna) 'river'. La colonna 'flow' riporta il flusso misurato nel giorno specificato, espresso in *metri cubi per secondo*. La chiave primaria 'id' è un semplice identificatore univoco della misurazione.

Scopo dell'esercitazione: Realizzare un'applicazione JavaFX che permetta di interrogare il data-set e di supportare un progettista che ha l'incarico di costruire la diga per la realizzazione di un bacino idrico che si inserisce nel flusso del fiume (Fig 2).

Fig 2



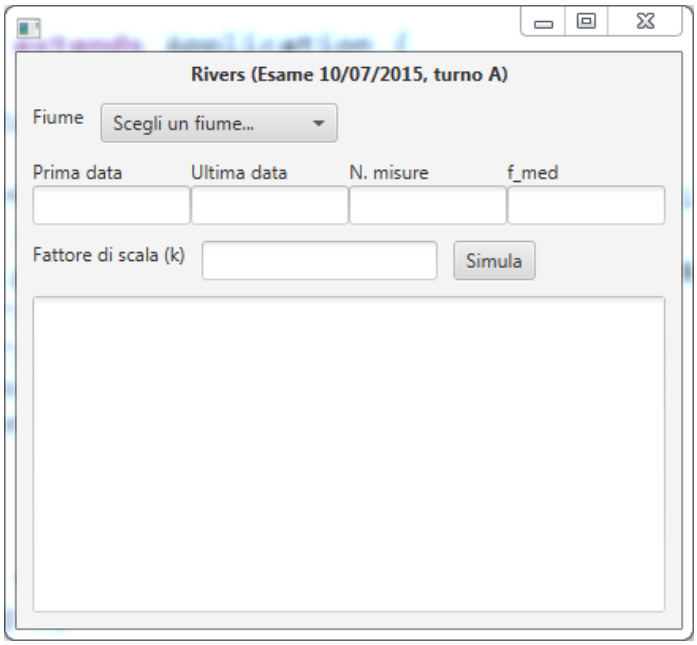
Descrizione dall'applicazione: L'applicazione deve fornire le seguenti funzionalità:

1. Permettere all'utente di selezionare, da una tendina, uno specifico fiume. Appena il fiume è selezionato, occorre riempire i campi dell'interfaccia che riportano, rispettivamente: la data della prima misurazione disponibile per tale fiume, la data dell'ultima misurazione disponibile per tale fiume, il numero totale di misurazioni, il valore medio del flusso misurato per tale fiume (detto f_{med}).

2. Permettere all'utente di simulare l'effetto di un bacino idrico della capienza totale Q (espressa in metri cubi). La simulazione dovrà aggiornare, giorno per giorno, la quantità di acqua presente nel bacino (detta C), in funzione dei dati sul flusso in ingresso (detto f_{in}) presente nel data set. Il calcolo deve tenere conto delle seguenti regole:
    a. L'utente può specificare il valore di Q inserendo un "fattore di scala" $k > 0$, dal quale si calcolerà[2] $Q = k \cdot f_{med} \cdot 30\ giorni$ (in altre parole, k indica la frazione di un mese a flusso "medio" che il bacino potrà contenere prima di riempirsi).
    b. L'occupazione iniziale del bacino è $C = {}^{Q}/_{2}$
    c. Il bacino dovrà garantire un flusso in uscita (detto f_{out}) pari ad almeno f_{out_min} , ogni giorno dell'anno. Per gli scopi della simulazione, si assuma $f_{out\_min} = 0.8 \cdot f_{med}$.
    d. Quando $f_{in} > f_{out}$, il livello C nel bacino salirà di una quantità legata alla differenza tra i due flussi.
    e. Non è possibile avere $C > Q$. In caso di flusso in ingresso eccessivo, questo deve essere scaricato in uscita (evento detto "tracimazione").
    f. In ogni giorno dell'anno, c'è una probabilità pari al 5% di avere un flusso richiesto in uscita più elevato, pari a $10 \cdot f_{out\_min}$, in quanto i campi devono essere irrigati.

    Il simulatore dovrà determinare, in funzione di k:
    - il numero di giorni in cui non si è potuta garantire l'erogazione minima;
    - l'occupazione media C_{med} del bacino nel corso della simulazione.

Fig 3



Nella realizzazione del codice, si lavori a partire dalle classi (Bean e DAO, FXML). È ovviamente permesso aggiungere o modificare classi e metodi.

Tutti i possibili errori di immissione o validazione dati devono essere gestiti, non sono ammesse eccezioni generate dal programma.

[2] https://datamarket.com/data/list/?q=interval:day%20provider:tsdl , create da Rob Hyndman, Professore di Statistica alla Monash University, Australia.

[2] Attenzione alle unità di misura, in quanto f_{med} è espresso in m^3 al secondo e non al giorno.