DOCUMENTATIE

TEMA *3*

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

NUME STUDENT: Toader Eric-Stefan
GRUPA: 302210

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei este implementarea folosind limbajul de programare Java a unei aplicatii desktop prin care sa se managerieze cu usurinta o baza de date continand clienti, produse si comenzi.

Obiectivele secundare ale temei sunt:

1. Crearea unei baze de date MySQL care sa contina 3 tabele: Client, Product si Orders si crearea unui cont de administrator al bazei de date
2. Popularea tabelelor cu date sugestive in tabelele Client si Product
3. Definirea claselor Client, Product, Orders: crearea de clase ce vor oglindi tabelele create, continand variabile instanta ce reproduc intocmai denumirile coloanelor bazei de date asociate
4. Implementarea operatiilor de interogare (select, insert, update, delete): crearea de metode care sa se ocupe cu operatiile clasice de interogare a bazei de date, in format generic
5. Realizarea unei interfete grafice prietenoase: dezvoltarea unei interfete grafice intuitive care sa faciliteze operatiile asupra bazei de date MySQL

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Pentru a rezolva cerintele problemei, va trebui sa realizam o interfata grafica prietenoasa si intuitiva ca utilizatorul sa poata realiza operatiile dorite asupra tabelelor bazei lui de date.

Asadar, ne putem imagina problema ca un black box cu doua posibile intrari (operatia asupra bazei de date si datele ce trebuie introduse/ actualizate) si o iesire (tabela pe care s-a efectuat operatia respectiva). In acest sens, vom avea o multitudine de ferestre, pentru a ajuta utilizatorul sa selecteze, in prima instanta, tabela pe care vrea sa efectueze operatii, operatia pe care vrea sa o efectueze si, eventual, datele de intrare necesare acesteia.

Scenariul in care utilizatorul doreste sa insereze, sa modifice, sau sa stearga un rand in oricare tabela a bazei de date, in afara de tabela Orders, este unul ce nu poate esua. Cu alte cuvinte, utilizatorul va avea intotdeauna rezultatul asteptat.

In scenariul in care utilizatorul va incerca sa plaseze o comanda cu o cantitate mai mare decat stocul curent al produsului selectat, aplicatia va trebui sa afiseze un mesaj de eroare enuntand acest fapt si sa nu efectueze plasarea comenzii. In caz contrar, daca stocul permite plasarea comenzii, operatia de inserare a unei noi comenzi in tabela Orders se va face normal.

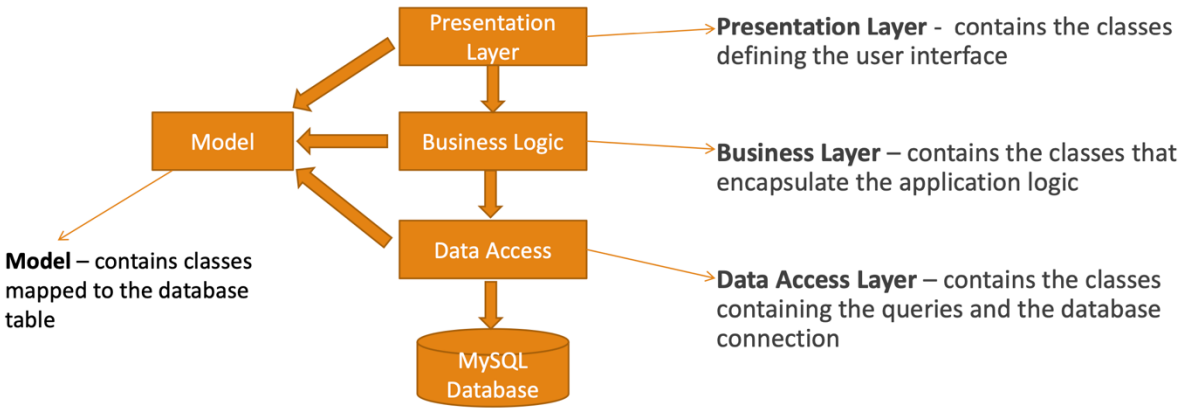
3. Proiectare

In cadrul proiectarii claselor, variabilele instanta din fiecare clasa au fost declarate cu modificatorul de acces “private”, pentru a respecta paradigma incapsularii OOP. Astfel, datele stocate intr-un obiect pot fi citite si/sau modificate doar prin intermediul metodelor de tip getter si setter.

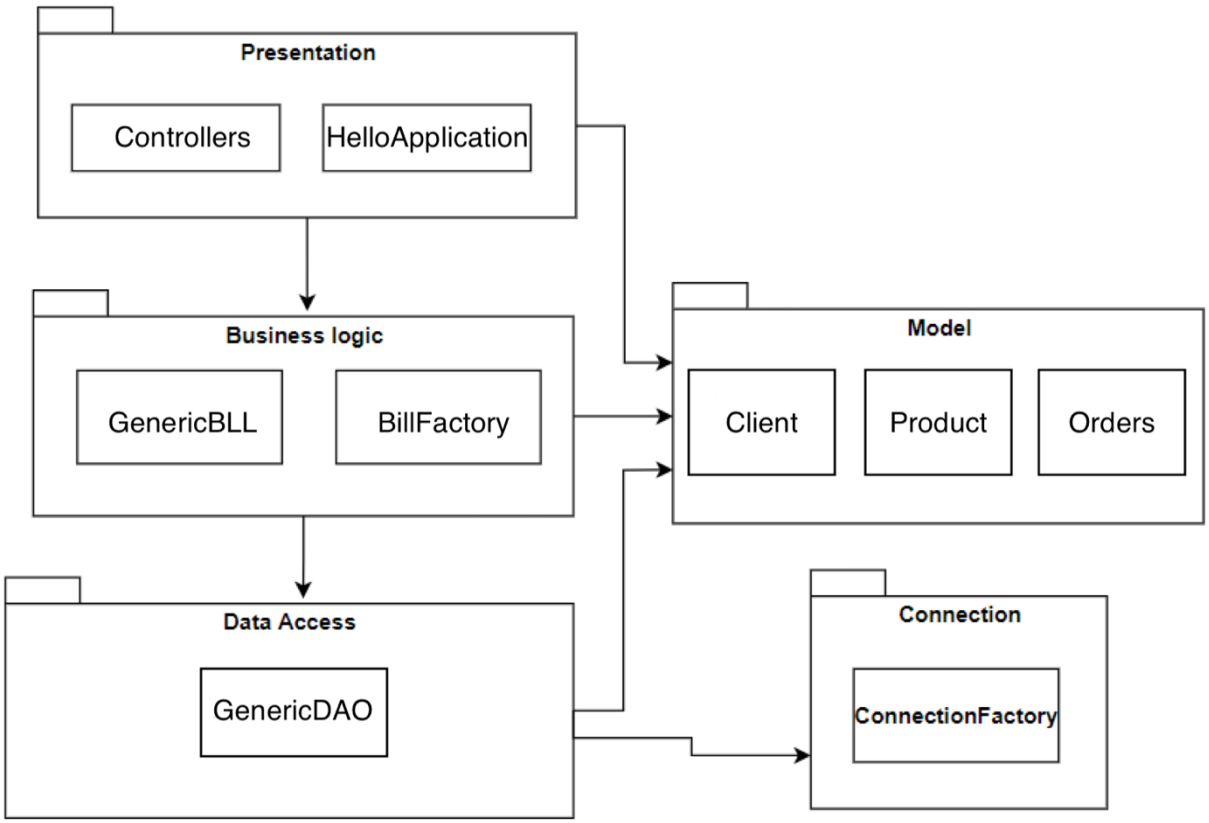
De asemenea, am folosit mostenirea in definirea claselor “concrete” de DAO si BLL si am implementat interfata Initializable, in contextul implementarii constructorilor ce se ocupa cu actualizarea tabelelor Client si Product.

In cadrul tuturor claselor, am definit variabile instanta de tip Integer. Asadar, am folosit paradigma impachetarii OOP pentru a reprezenta tipuri de date primitive in clase invelitoare ce stocheaza valori de tip int, pentru a ma putea folosi de metode statice implementate pentru aceste clase, in implementarea ulterioara a parsarii si a operatiilor matematice.

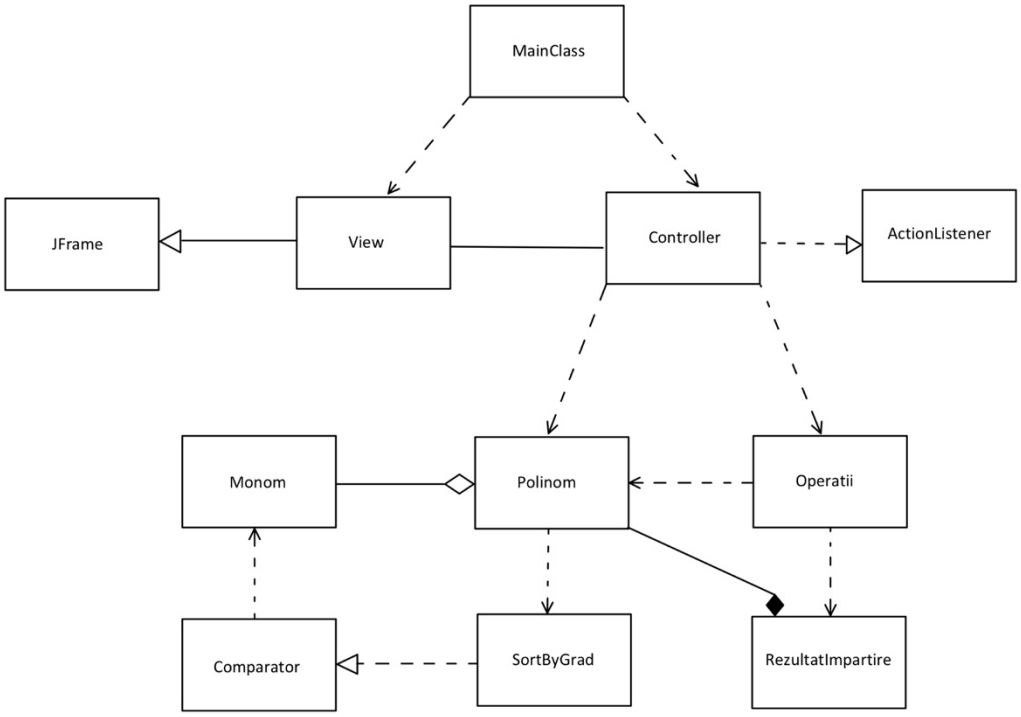
In cadrul proiectarii aplicatiei de management al bazei de date, clasele implementate sunt incluse in pachete stratificate ce respecta forma urmatoare:



Pachetele definite mai sus contin clasele:



Clasele definite in clasele de mai sus relationeaza conform urmatoarei diagrame UML:



In cadrul temei am folosit liste de tip ArrayList pentru a stoca succesiunea de monoame in obiectele de tip Polinom si pe tot parcursul operatiilor am iterat prin aceste liste folosind for each-ul specific OOP.

4. Implementare

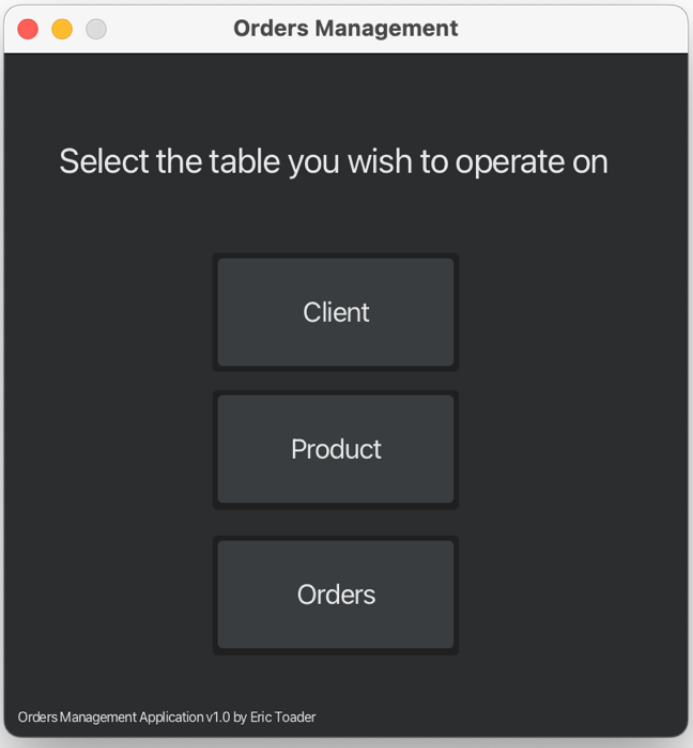
In cadrul implementarii temei am definit clasele prezentate mai jos, ale caror cele mai importante metode au fost si detaliate.

1. Clasa HelloApplication
   - Fara variabile instante sau de clasa
   - Metoda *main*, care initializeaza fereastra JavaFX afisabila

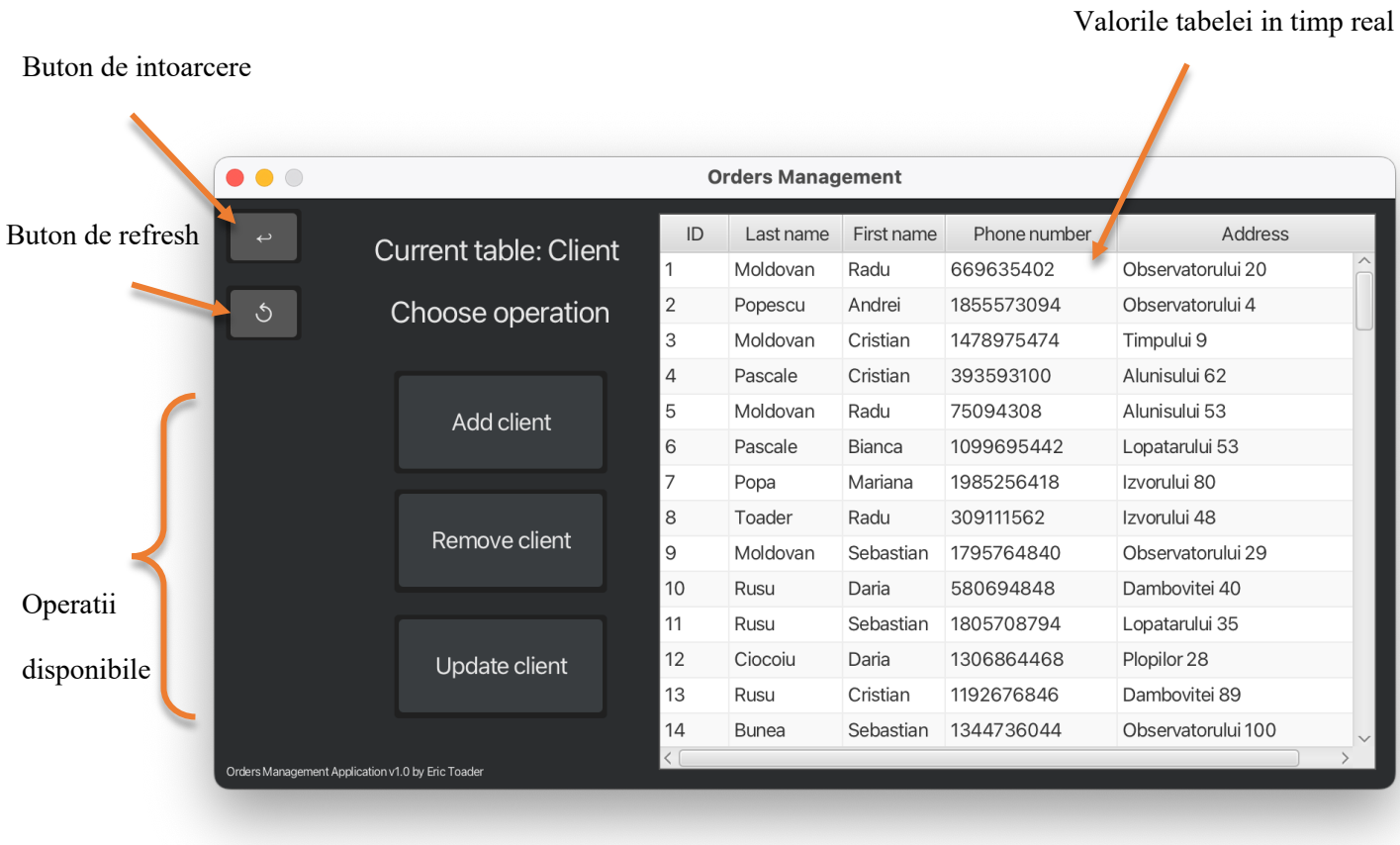
2. Clasele Client, Product si Orders
   - Variabilele instanta: variabile instanta de tip si nume care respecta in exactitate tabelele bazei de date
   - Metode *getter* si *setter* necesare returnarii corecte a setului de rezultate obtinut in urma interogarilor
3. Clasa ConnectionFactory
   - Fara variabile instanta, variabile de clasa ce stocheaza date necesare la conectarea cu baza de date si o variabila de clasa ConnectionFactory folosita drept “singleton”, permitand astfel intretinerea unei singure unice conexiuni cu baza de date, prin intermediul aplcatiei.
   - Metodele *createConnection* si *getConnection* ce sunt folosite pentru intializarea unei singure conexiuni cu baza de date si folosirea acesteia
   - Metodele *close* care se asigura ca statement-ul, conexiunea si resultSet-ul sunt inchise in urma finalizarii unei interogari

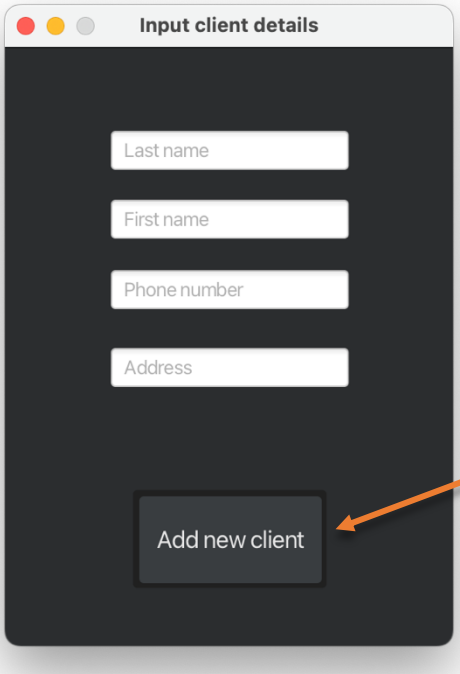
4. Clasa GenericBLL si derivatele
   - Variabile instanta: o clasa de tip generic T si o referinta catre un obiect data access generic de tip T; fara variabile de clasa
   - Metodele standard de inserare, actualizare, stergere si selectie a datelor ce vor comunica operatia si tabela mai departe stratului de Data Access Operation

5. Clasa GenericDAO
   - Variabile instanta: o clasa de tip generic T; variabile de clasa: o instanta de tip Logger care sa inregistreze erorile intalnite
   -
6. Clasele Controller
   - Metode de initialize (unde e cazul), metode de actualizare a tabelului si metode asociate butoanelor din ferestrele JavaFX



Butoane pentru selectarea tabelei pe care se vrea a se opera


Valorile tabelei in timp real
Buton de intoarcere
Buton de refresh
Operatii disponibile
Orders Management
Current table: Client
Choose operation
Add client
Remove client
Update client
Orders Management Application v1.0 by Eric Toader
ID
Last name
First name
Phone number
Address
1 Moldovan Radu 669635402 Observatorului 20
2 Popescu Andrei 1855573094 Observatorului 4
3 Moldovan Cristian 1478975474 Timpului 9
4 Pascale Cristian 393593100 Alunisului 62
5 Moldovan Radu 75094308 Alunisului 53
6 Pascale Bianca 1099695442 Lopatarului 53
7 Popa Mariana 1985256418 Izvorului 80
8 Toader Radu 309111562 Izvorului 48
9 Moldovan Sebastian 1795764840 Observatorului 29
10 Rusu Daria 580694848 Dambovitei 40
11 Rusu Sebastian 1805708794 Lopatarului 35
12 Ciocoiu Daria 1306864468 Plopilor 28
13 Rusu Cristian 1192676846 Dambovitei 89
14 Bunea Sebastian 1344736044 Observatorului 100

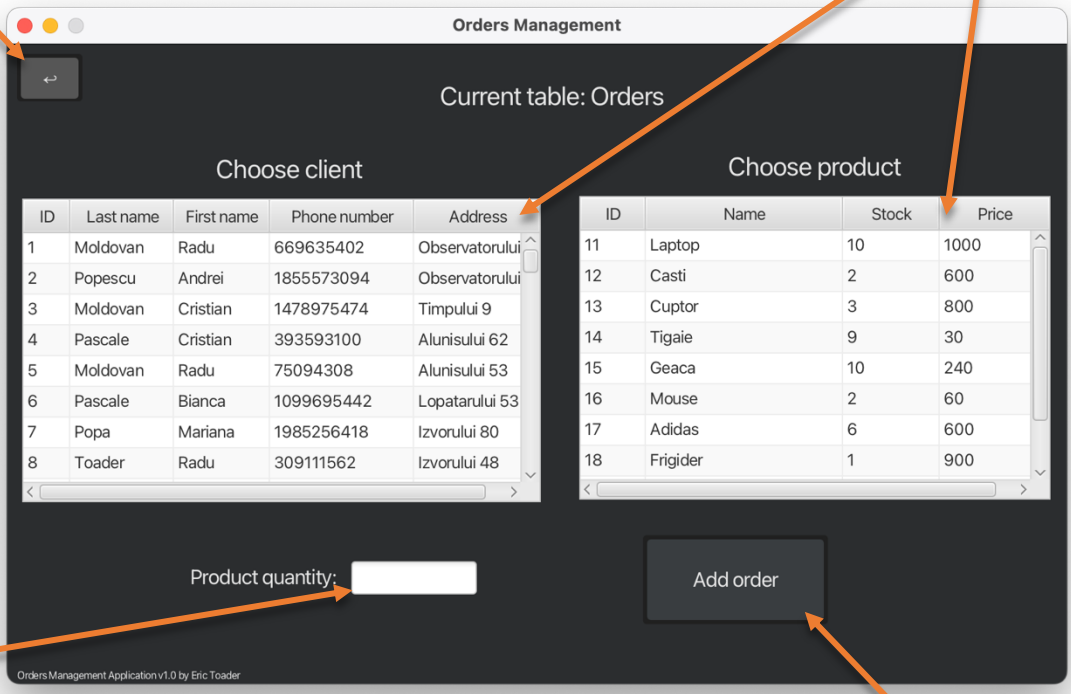


Campuri de completat

Buton de efectuare operatie

Tabele ce vor fi folosite pentru a selecta clienti si produse

Buton de intoarcere



Precizarea cantitatii

Buton de efectuare operatie

5. Rezultate

Pentru a testa aplicatia, am testat fiecare scenariu de utilizare posibil, rezultatul obtinut fiind cel asteptat in fiecare caz. De exemplu, la inserarea sau actualizarea oricarui produs sau client, aplicatia duce la bun sfarsit operatiile, dar trebuie reactualizat tabelul (apasand pe butonul de refresh) pentru a se observa modificarile. In cazul stergerii din tabelele Client sau Product, rezultatul se va observa imediat. De asemenea, pentru adaugarea comenzilor, aplicatia duce la bun sfarsit operatiile de inserare a comenzilor si de actualizare a stocului produsului comandat.

6. Concluzii

In urma realizarii acestei teme am deprins abilitatea de a crea o interfata grafica prietenoasa si intuitiva in Java FX, familiarizandu-ma simultan si cu Scene Builder.

De asemenea, am invatat sa folosesc bibliotecile mysql din Java pentru a ma conecta la baza mea de date si a efectua operatii de interogare.

In ceea ce priveste o posibila dezvoltare ulterioara a aplicatiei, aceasta poate fi extinsa la nesfarsit, datorita faptului ca metodele si clasele de Business Logic si Data Access au fost abstractizate, oferind astfel oportunitatea scalabilitatii si reutilizarii aplicatiei si in alte arii.

7. Bibliografie

Connect to MySql from a Java application
- https://www.baeldung.com/java-jdbc
- http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/

Layered architectures
- https://dzone.com/articles/layers-standard-enterprise

Reflection in Java
- http://tutorials.jenkov.com/java-reflection/index.html

Creating PDF files in Java
- https://www.baeldung.com/java-pdf-creation

JAVADOC
- https://www.baeldung.com/javadoc

SQL dump file generation
- https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html