**Лабораторная работа №2**

**Цель работы:** создать приложение в Qt для работы с базой данных. Реализовать валидатор и написать к нему Unitest.

**Задача:** Цветочная оранжерея выращивает различные виды цветов и продает на заказ  
составленные из них композиции. Каждая композиция имеет свое название и может состоять  
как из цветов одного вида, так и из цветов разного вида.

Дирекция оранжереи владеет информацией:  
о цветах: название цветка, сорт, стоимость одного цветка;  
о композициях: название композиции, название входящего в композицию цветка, количество единиц;  
о выполнении заказов: дата принятия заказа, название композиции,  
количество единиц, покупатель. *о покупателе:* логин, пароль

**Необходимо выполнить:**

1. Создать таблицы БД
2. Обеспечить с помощью операторов Select, Insert, Update, Delete обновление и выборку информации в указанных таблицах
3. Создать оконное приложение, которое позволяет манипулировать данными с использованием вышеупомянутых операторов.
4. Написать тесты для валидатора

**Свою работу я разделила на 5 частей:**

1. Разработка необходимых таблиц, связи таблиц между собой и наполнения каждой таблицы.
2. Создание базы данных с помощью DB Browser
3. Реализация работы с базой данных через QT. Создание интерфейса для простейших операций над базами данных
4. Написание валидатора
5. Написание теста к валидатору

**Часть 1. Разработка необходимых таблиц, связи таблиц между собой и наполнения каждой таблицы.**

Предварительная заготовка для таблиц выглядела следующим образом:

Flower(название text, сорт text, cost integer, id PK integer)

Compositions(id integer, name text, cost integer)

Flower\_Comp(id integer, id\_flower integer, id\_comp integer, count integer)

Order(id integer, date\_v text, id\_comp integer, count integer, user\_id integer, summa integer)

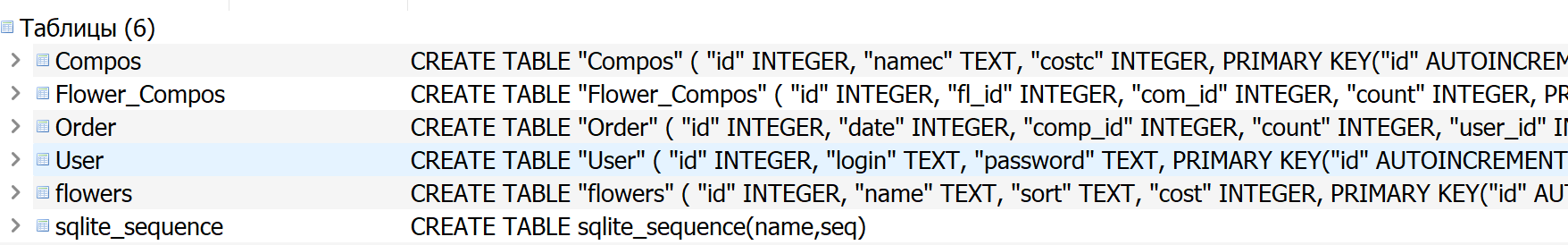
User(id integer, login text, password text)

Из заготовки видно, что таблицы Order и Flower\_Comp связаны с остальными таблицами путем указания id нужных столбцов.

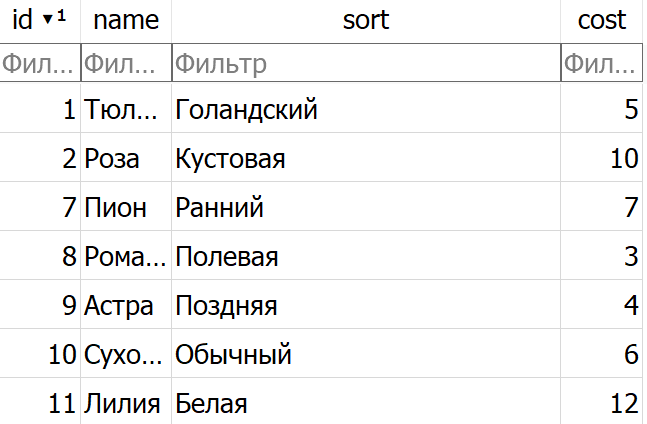
Типы данных выбраны соответственно доступным в db Browser, так как дальнейшая работа будет проводится в нем.

**Часть 2. Создание базы данных с помощью DB Browser**

В db Browser создается 5 таблиц:



Далее таблицы были заполнены вручную. Пример:



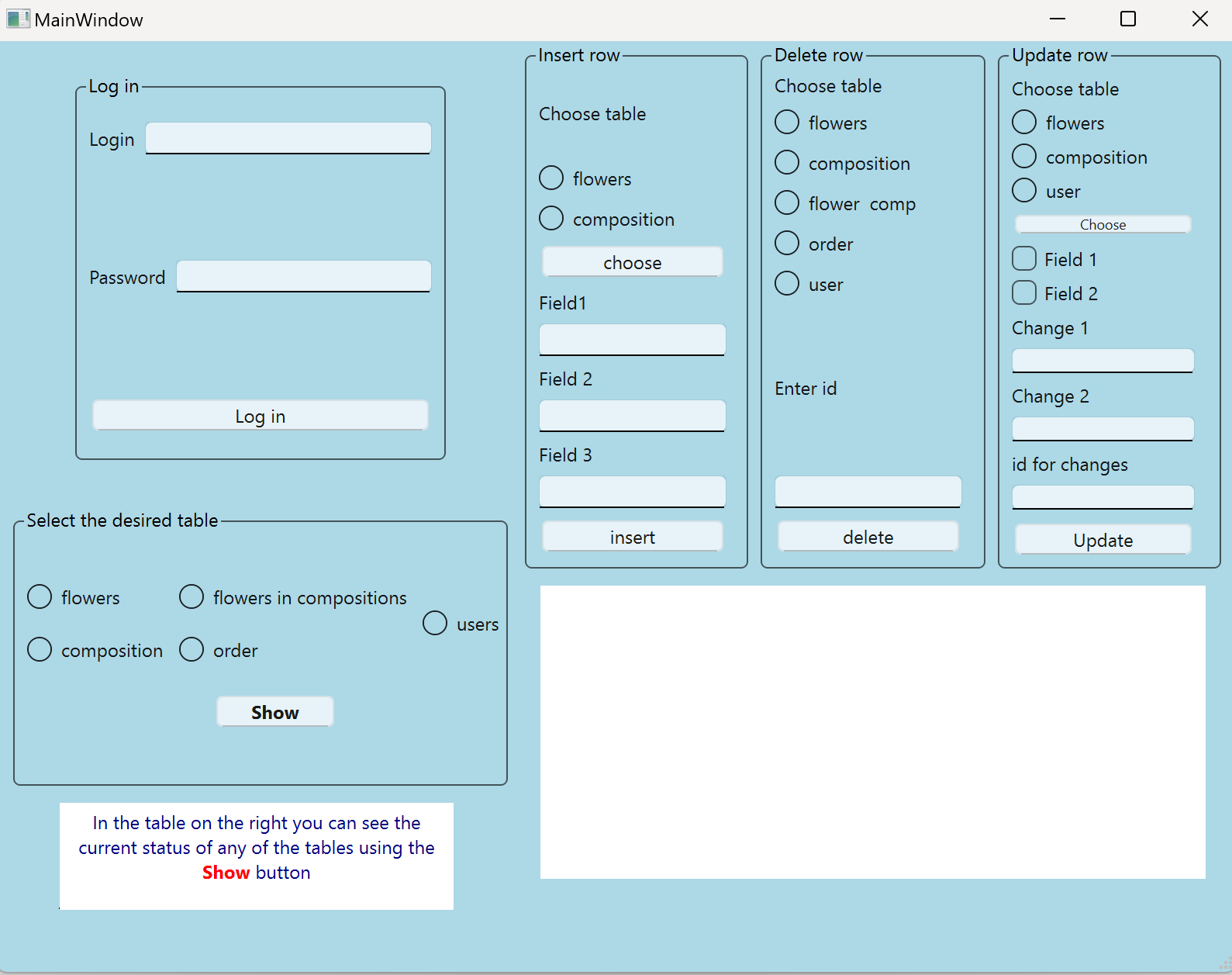
Далее файл был сохранен с расширение .db

**Часть 3. Реализация работы с базой данных через QT. Создание интерфейса для простейших операций над базами данных.**

Основной задачей было создать удобный интерфейс для пользоваля, который бы выполнял те же операции что и db Browser, однако пользователю не нужно было бы писать команды вручную.

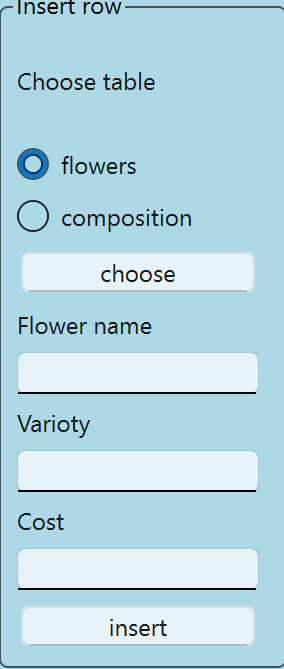
Для того, чтобы подать информацию структурировано, использовались такие инструменты QT как: QSqlTableModel(вывод таблиц в текущем состоянии на выбор), GroupBox(для объединения элементов, которые решают общую задачу), TextBrowser и Label( для вывода указаний пользователю).

Вот финальный вид приложения:



Пользователь в каждом действии может выбрать таблицу, с которой он работает, и в соответствии с этим изменятся Label для упрощения ввода, т. к. каждая таблица имеет специфичные столбцы.

Пример такого изменения на операции Insert:

(изменение происходит после нажатия кнопки choose)

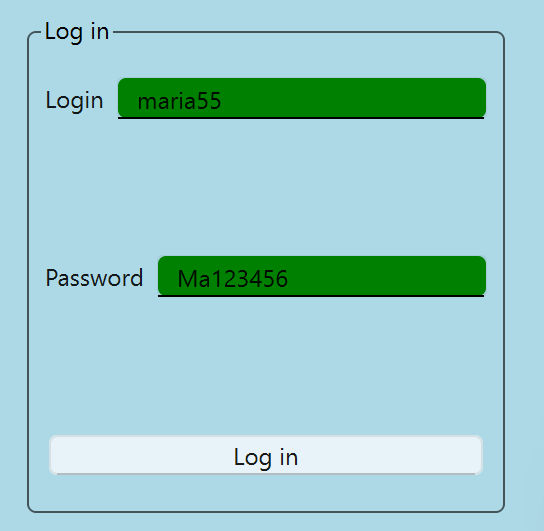
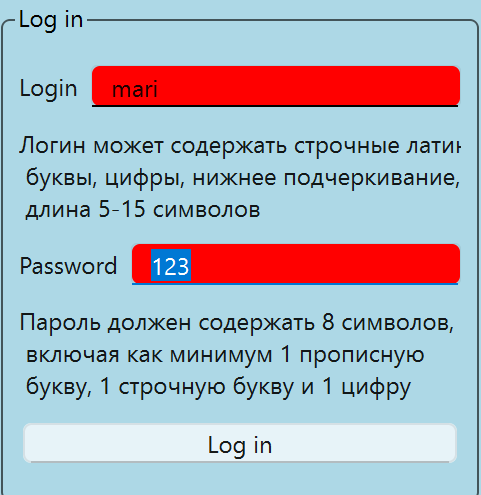
Для более глубокого разбора кода оставляю ссылку на GitHub:

<https://github.com/fominami/lab_2_flowers>

**Часть 4.** **Написание валидатора**

Валидатор был создан для операции добавления пользователя в БД. Чтобы добавить пользователя было необходимо ввести login и password. С помощью валидатора было ограничено количество символов для ввода и поставлены правила для создания надежного пароля. Для наглядности в случае подходящего логина и пароля lineEdit, куда вводились строки, подсвечивался зеленым, а в случае неудачи красным. Также при неверном вводе всегда показывались условия для правильного ввода.

Вот пример двух вариантов:

**Часть 5. Написание теста к валидатору**

Тестировавние проводилось с помощью Visual Studio, где был выбран шаблон google test. Валидатор, который нужно было протестировать, был написан как отдельный класс, поэтому мы подключали файл .h для дальнейшего подключения к тестам. Т. к. класс писался в Qt, он содержал библиотеки Qt, поэтому для качественного тестирования нужно было провести замены на эквивалентные элементы из Visual Studio: (QRegularExpression-> regex, QRegularExpressionMatch->regex\_match, QValidator::State->bool, QString->string)

Далее представлены тесты и их результаты для login и password:

