МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Проектирование системы экранных форм в Qt для ведения локальной базы данных

Лабораторная работа №2

по дисциплине «Базы данных»

Выполнил студент Чекулаев В.Ю.

Факультет, группа ФКФН, ПО(аб)-81

Проверил Федосеев А.А.

Хабаровск – 2020г.

Задание:

В процессе выполнения второй лабораторной работы написать программу с использованием библиотеки Qt и СУБД SQLite. Программа должна

1. обеспечивать**:**

* систему экранных форм для редактирования содержания таблиц.
* отдельные экранные формы для ведения независимых и зависимых таблиц. Под зависимыми таблицами понимаются таблицы, при добавлении новой информации в которые требуются данные из других таблиц, в дальнейшем называемых справочными (фактически, это означает наличие ограничение целостности по существованию). При использовании для работы с зависимыми таблицами необходимо создавать многотабличные экранные формы со справочными таблицами, открытыми только для просмотра).
* обеспечить контроль ограничения целостности по существованию, путём выдачи сообщения об наличии связанных записей.
* обеспечить контроль ограничения целостности по возможным ключам путём выдачи сообщения об наличии связанных записей.

По результатам работы необходимо продемонстрировать возможность

1. корректного ведения локальной базы данных с использованием созданной в
2. работе системы экранных форм.

**Содержимое скрипта создания БД**

*pragma foreign\_keys=on;*

*create table Plan(name\_plan varchar(100) unique not null, ID\_plan integer primary key, ID\_status integer references Status, ID\_spec integer references Specialnost, ID\_forma integer references Forma\_obucheniya, add\_year integer, duration real, am\_of\_disc integer);*

*create table Status(name\_status varchar(20) unique not null, ID\_status integer primary key);*

*create table Specialnost(name\_spec varchar(100) unique not null, ID\_spec integer primary key, ID\_spez integer reference Spez, ID\_kaf integer reference Kafedra);*

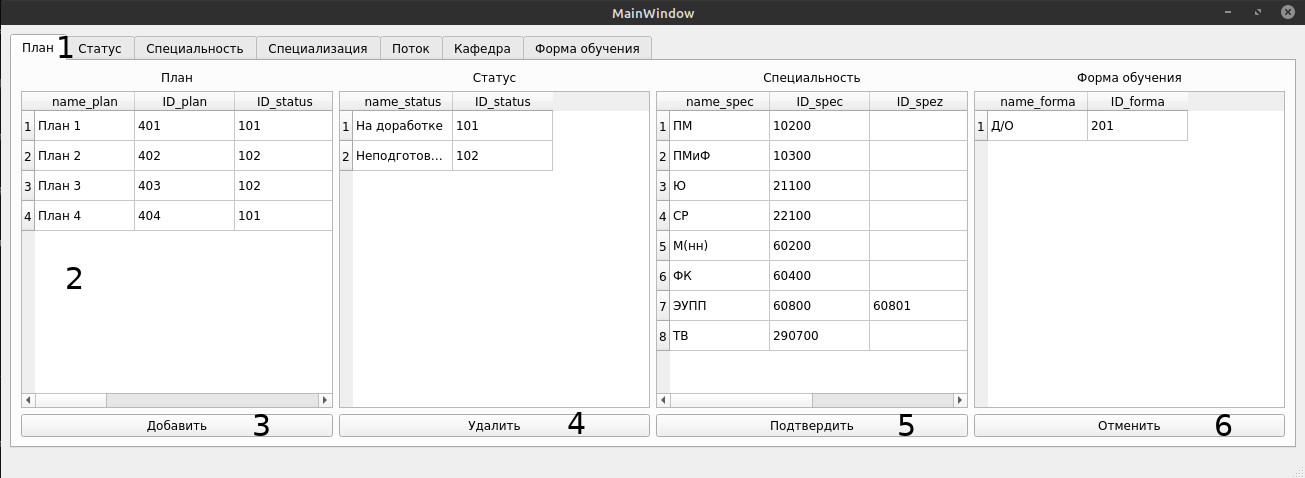
*create table Spez(ID\_spec integer primary key, name\_spez varchar(100) unique not null);*

*create table Potok(name\_potok varchar(10) unique not null, amount integer, ID\_potok integer primary key, ID\_plan integer reference Plan);*

*create table Kafedra(name\_kaf varchar(20) unique not null, ID\_kaf integer primary key);*

*create table Forma\_obucheniya(name\_forma varchar(100) unique not null, ID\_forma integer primary key);*

**Макеты экранных форм с перечнем и назначением визуальных компонентов**



1. Вкладки с таблицами (QTabWidget). Предназначены для удобства пользования приложением.
2. Таблица базы данных (QTableView).
3. Кнопка «Добавить» (QPushButton). Предназначена для внесения новой записи в таблицу базы данных.
4. Кнопка «Удалить»(QPushButton). Предназначена для удаления записи из таблицы базы данных.
5. Кнопка «Подтвердить»(QPushButton). Предназначена для подтверждения операций произведенных после предыдущего подтверждения.
6. Кнопка «Отменить»(QPushButton). Предназначена для отмены операций произведенных после предыдущего подтверждения.

**Фрагменты программы, отвечающие за взаимодействие с базой данных и визуальными компонентами**

int MainWindow::initDB(){

m\_database = QsqlDatabase::addDatabase("QSQLITE");

QString filename("../base.bd");

m\_database.setDatabaseName(filename);

if(QFileInfo::exists(filename)){

m\_database.open();

m\_database.exec("pragma foreign\_keys=on");

} else{

QMessageBox::critical(this, "Critical error", "Can not open database");

return 1;

}

*//-----------------------*

m\_Plan = new QSqlTableModel(nullptr, m\_database);

m\_Plan->*setTable*("Plan");

m\_Plan->*select*();

ui->Plan\_Plan->*setModel*(m\_Plan);

m\_Plan->*setEditStrategy*(QSqlTableModel::*OnManualSubmit*);

m\_Status\_SUB = new QSqlQueryModel;

ui->Plan\_Status->*setModel*(m\_Status\_SUB);

m\_Specialnost\_SUB = new QSqlQueryModel;

ui->Plan\_Specialnost->*setModel*(m\_Specialnost\_SUB);

m\_Forma\_obucheniya\_SUB = new QSqlQueryModel;

ui->Plan\_Forma->*setModel*(m\_Forma\_obucheniya\_SUB);

*//-----------------------*

m\_Status = new QSqlTableModel(nullptr, m\_database);

m\_Status->*setTable*("Status");

m\_Status->*select*();

ui->Status\_2->*setModel*(m\_Status);

m\_Status->*setEditStrategy*(QSqlTableModel::*OnManualSubmit*);

*//-----------------------*

...

updateDB();

return 0;

}