

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт Компьютерных Наук и Технологий

Высшая школа искусственного интеллекта

Курсовая работа
по дисциплине «Объектно-ориентированное программирование»
Разработка приложения телефонного справочника с использованием библиотеки Qt

Студент гр. 3530201/00001 _____ Кузнецов А.Д.

Преподаватель _____ Глазунов В.В.

Содержание

Введение	2
1 Описание задачи	2
1.1 Постановка задачи	2
1.2 Пояснение к пунктам постановки задачи	2
2 Описание реализации	3
2.1 Класс MainWindow	3
2.2 Поля класса MainWindow	3
2.2.1 int rowSelected	3
2.2.2 Ui::MainWindow *ui	3
2.2.3 QSqlDatabase database	3
2.2.4 QSqlQuery *query	3
2.2.5 QSqlTableModel *db_model	3
2.3 Методы класса MainWindow	3
2.3.1 MainWindow(QWidget *parent = nullptr)	3
2.3.2 void on_pushButton_clicked()	4
2.3.3 void on_tableView_clicked(const QModelIndex &index)	4
2.3.4 void on_pushButton_2_clicked()	5
2.3.5 void on_lineEdit_editingFinished()	5
2.3.6 void on_lineEdit_textEdited(const QString &arg1)	5
2.4 Классы-делегаты	5
2.4.1 Проверка ФИО	5
2.4.2 Проверка электронного адреса	6
2.4.3 Проверка номера телефона	6
2.4.4 Проверка даты рождения	7
3 Тестирование программы	8
Заключение	13
Список источников	14
Приложение. Исходный код	15
3.1 main.cpp	15
3.2 mainwindow.h	15
3.3 mainwindow.cpp	15
3.4 delegate0.h	16
3.5 delegate0.cpp	16
3.6 delegate4.h	17
3.7 delegate4.cpp	17
3.8 delegate5.h	17
3.9 delegate5.cpp	17
3.10 delegateem.h	18
3.11 delegateem.cpp	18

Введение

В данной работе представлено описание разработки и реализации оконного приложения "Телефонный справочник" средствами фреймворка Qt. Данная программа используется для работы с базой данных, которая содержит такие данные как:

- Фамилия
- Имя
- Отчество
- Место рождения
- E-mail
- Телефонный номер

и позволяет удобно с ними взаимодействовать. Под удобным взаимодействием имеются в виду возможности сортировки, поиска по значению, внесения изменений в базу данных напрямую (без заполнения лишних форм и нажатия дополнительных кнопок).

1 Описание задачи

1.1 Постановка задачи

Для выполнения данной работы требуется:

- Реализовать оконный интерфейс, позволяющий добавлять/удалять любые записи, а также редактировать все поля
- Реализовать проверку вводимых данных ФИО, телефона, даты рождения с помощью регулярных выражений или другого подходящего способа
- Реализовать возможность сортировки данных по любому из указанных полей
- Реализовать сохранение полученных данных в файл и загрузку данных из файла
- Реализовать поиск по значению по полному совпадению

1.2 Пояснение к пунктам постановки задачи

Под проверкой вводимых данных с помощью регулярных выражений имеются в виду уникальные проверки на ввод данных. Для полей фамилии, имени и отчества осуществляется проверка на прописную первую букву, строчные остальные буквы (а также дефис), один язык (либо латиница, либо кириллица). В конце знак дефиса стоять не может.

Для значений даты рождения выполняется проверка на дату рождения, меньшую текущей даты, на соответствие введенной даты календарному дню (проверка на високосный год, на количество дней в месяце).

Для полей электронного адреса выполняется проверка на написание значения латинскими буквами, на наличие знака '@', на разделение доменной части на два уровня символом '.'.

Для полей телефона осуществляется проверка на соответствие ввода международному формату:

+ 'Код_страны' ('Код_региона') 'Семь_цифр'

Для полей адреса проверка не осуществляется

Под сохранением файла имеется в виду создание файла с расширением .db. Из этого файла должна производиться и загрузка данных

Под сортировкой имеется в виду возможность пользователя отсортировать данные по возрастанию или по убыванию по любому из полей в лексикографическом порядке

Под поиском значения имеется в виду выделение ячейки/ячеек, полностью соответствующих запрашиваемому значению

2 Описание реализации

В основе реализации лежит работа с модулем QtSql и его составляющими классами:

- QSqlTableModel - класс третьего уровня, предоставляет модели для отображения результатов запросов в представлениях.
- QSqlQuery - класс второго уровня, предоставляет программный интерфейс для обращения к базе данных
- QSqlDatabase - также класс второго уровня
- QSqlField - класс второго уровня
- QSqlRecord - класс второго уровня

Чтобы начать использование модуля QtSql, необходимо в проектный файл добавить строку:

```
1 QT +=sql
```

Для реализации визуального представления на макет формы добавлен виджет TableView. В настройках TableView активирована возможность сортировки.

2.1 Класс MainWindow

В связи с возможностью простой реализации программы, создание интерфейса программы, самой базы данных и реализация методов осуществляются в стандартном классе MainWindow, который наследуется от QMainWindow и создается по-умолчанию. В качестве идентификатора драйвера СУБД в данной реализации выбран *QSQLITE*.

2.2 Поля класса MainWindow

2.2.1 int rowSelected

Поле со спецификатором доступа типа private, используется для хранения номера строки, в которой пользователь выделил запись.

2.2.2 Ui::MainWindow *ui

Стандартное поле класса MainWindow со спецификатором доступа типа private. Данный указатель содержит адрес объекта интерфейса. Выделение памяти под объект интерфейса и присвоение его адреса происходит в конструкторе

2.2.3 QSqlDatabase database

Класс QSqlDatabase предоставляет интерфейс для подключения к базе данных через соединение. Экземпляр класса предоставляет соединение.

2.2.4 QSqlQuery *query

Указатель на экземпляр класса QSqlQuery. QSqlQuery обеспечивает интерфейс для выполнения SQL-запросов и навигации по результирующей выборке

2.2.5 QSqlTableModel *db_model

Указатель на экземпляр QSqlTableModel. Данный класс необходим для отображения базы данных, позволяет осуществлять редактирование данных и может отображать данные в табличной и иерархической форме.

2.3 Методы класса MainWindow

2.3.1 MainWindow(QWidget *parent = nullptr)

Конструктор класса MainWindow. В данном методе происходит создание базы данных под названием ContactDB.db, если такой базы не существует, или соединение с ранее созданной базой данных ContactDB.db. Для того чтобы подключиться к базе данных, требуется четыре следующих параметра:

- имя базы данных - передается в метод `QSqlDatabase::setDatabaseName()`;
- имя пользователя, желающего к ней подключиться, - передается в метод `QSqlDatabase::setUserName()`;
- имя компьютера, на котором размещена база данных, - передается в метод `QSqlDatabase::setHostName()`;
- пароль - передается в метод `QSqlDatabase::setPassword()`.

Методы вызываются из объекта, созданного с помощью статического метода `QSqlDatabase::addDatabase()`

Само соединение осуществляется методом `QSqlDatabase::open()`. Возвращаемое значение проверяется. В случае возникновения ошибки в `qDebug()` будет выведена информация об ошибке.

Для исполнения команд SQL после установки соединения используется класс `QSqlQuery`. Запросы оформляются в виде обычной строки и затем выполняются методом `QSqlQuery::exec()`. Для создания таблицы и выставления заголовков используется запрос

```
1 query->exec("CREATE TABLE Contacts ( Фамилия TEXT, Имя TEXT, Отчество TEXT, Адрес TEXT, Дата
   рождения TEXT, email TEXT, Номер TEXT);");
```

Для создания графического отображения создается объект `QSqlTableModel`.

Для проверки вводимых значений к каждому из столбцов подключается соответствующий делегат.

Исходный код метода:

```
1 MainWindow::MainWindow(QWidget *parent)
2     : QMainWindow(parent)
3     , ui(new Ui::MainWindow)
4 {
5     ui->setupUi(this);
6     QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
7     db.setDatabaseName("../ContactDB.db");
8     db.setUserName("userdefault");
9     db.setHostName("userdefault");
10    db.setPassword("password");
11    if(!db.open()){
12        qDebug() << "Database couldn't be opened";
13    }
14    query = new QSqlQuery(db);
15    query->exec("CREATE TABLE Contacts ( Фамилия TEXT, Имя TEXT, Отчество TEXT, Адрес TEXT, Дата
   рождения TEXT, email TEXT, Номер TEXT);");
16    db_model = new QSqlTableModel(this, db);
17    db_model->setTable("Contacts");
18    db_model->select();
19    ui->tableView->setModel(db_model);
20    ui->tableView->setItemDelegateForColumn(0, new Delegate0());
21    ui->tableView->setItemDelegateForColumn(1, new Delegate0());
22    ui->tableView->setItemDelegateForColumn(2, new Delegate0());
23    ui->tableView->setItemDelegateForColumn(4, new Delegate4());
24    ui->tableView->setItemDelegateForColumn(6, new Delegate5());
25    ui->tableView->setItemDelegateForColumn(5, new DelegateEm());
26    ui->tableView->resizeColumnsToContents();
27    ui->lineEdit->setText("for searching");
28    this->setWindowTitle("QSQL DataBase");
29 }
```

2.3.2 void on_pushButton_clicked()

Слот, реализующий добавление строки в базу данных, если пользователь нажал кнопку "Добавить запись". Не принимает и не возвращает значения. Осуществляет вставку новой строки под самой последней строкой.

Исходный код:

```
1 void MainWindow::on_pushButton_clicked(){
2     db_model->insertRow(db_model->rowCount());
3 }
```

2.3.3 void on_tableView_clicked(const QModelIndex &index)

Слот, реализующий считывание номера ряда, в котором пользователь выделил ячейку. Принимает значение типа `QModelIndex&`. Не возвращает значения. Исходный код:

```
1 void MainWindow::on_tableView_clicked(const QModelIndex& index){
2     rowSelected=index.row();
3 }
```

2.3.4 void on_pushButton_2_clicked()

Слот, реализующий удаление ряда, в котором пользователь выделил ячейку. Обновляет таблицу после удаления ряда. Не принимает и не возвращает значения. Исходный код:

```
1 void MainWindow::on_pushButton_2_clicked(){
2     db_model->removeRow(rowSelected);
3     ui->tableView->update();
4     db_model->select();
5     return;
6 }
```

2.3.5 void on_lineEdit_editingFinished()

Слот, реализующий поиск по значению, которое пользователь ввел в окно ввода. Слот вызывается при окончании редактирования, т.е. нажатии кнопки ввода. Для считывания введенного текста используется метод QLineEdit::text(). Поиск осуществляется с помощью прохождения двумя циклами по всем значений базы данных. При нахождении полного соответствия происходит выделение ячейки, в который было найдено искомое значение. Не принимает и не возвращает значения

Исходный код:

```
1 void MainWindow::on_lineEdit_editingFinished()
2 {
3     QString strToSearch = ui->lineEdit->text();
4     for(int i = 0; i<db_model->rowCount();i++){
5         QSqlRecord data = db_model->record(i);
6         for(int j = 0; j<data.count();j++){
7             QSqlField tempField = data.field(j);
8             if(strToSearch==tempField.value()){
9                 qDebug()<<"It's here";
10                QModelIndex index = ui->tableView->model()->index(i,j);
11                qDebug()<<index;
12                ui->tableView->selectionModel()->select(index,QItemSelectionModel
13                ::Select);
14            }
15        }
16 }
```

2.3.6 void on_lineEdit_textEdited(const QString &arg1)

Слот, снимающий все выделения с таблицы при любом изменении введенных пользователем данных. Исходный код:

```
1 void MainWindow::on_lineEdit_textEdited(const QString &arg1)
2 {
3     ui->tableView->clearSelection();
4 }
```

2.4 Классы-делегаты

Для проверки вводимых пользователем данных используются классы-делегаты, наследуемые от QItemDelegate. Все классы-делегаты не имеют полей и имеют лишь два метода: пустой конструктор и метод createEditor(). В методе createEditor() и выполняется проверка вводимого значения и установка валидатора. Если вводимое значение не соответствует формату, делегат не позволит ввести его в базу данных.

2.4.1 Проверка ФИО

Для проверки фамилии, имени и отчества используется класс Delegate0. В нем задано регулярное выражение, обеспечивающее ввод согласно шаблону:

1. Заглавная буква
2. От одной до 30 строчных букв или дефис
3. Одна строчная буква в конце (так как дефис не должен быть последним символом)

Данный шаблон применяется либо для латинских букв, либо для кириллических. Ввод и тех, и других одновременно запрещен.

Исходный код:

```
1 QWidget* Delegate0::createEditor(QWidget*parent, const QStyleOptionViewItem&,const QModelIndex
  &)const{
2     QLineEdit*lineEdit = new QLineEdit(parent);
3     QRegExp regExp("[А - Я]{1}[а - я ё\\s-]{1,30}[а - я ё]{1}|[A-Z]{1}[a-z\\s-]{1,30}[a-z]{1})");
4     QValidator*validator = new QRegExpValidator(regExp,lineEdit);
5     lineEdit->setValidator(validator);
6     return lineEdit;
7 }
```

2.4.2 Проверка электронного адреса

Для проверки электронного адреса используется класс DelegateEm. В нем задано регулярное выражение, обеспечивающее ввод согласно шаблону:

1. Любые цифры/буквы латинского алфавита
2. Знак '@'
3. Строчные буквы латинского алфавита
4. Знак '.'
5. Строчные буквы латинского алфавита

Исходный код:

```
1 QWidget* DelegateEm::createEditor(QWidget*parent, const QStyleOptionViewItem&,const
  QModelIndex&)const{
2     QLineEdit*lineEdit = new QLineEdit(parent);
3     QRegExp regExp("[0-9A-Za-z]*[@][a-z]{1,}.[a-z]{1,})");
4     QValidator*validator = new QRegExpValidator(regExp,lineEdit);
5     lineEdit->setValidator(validator);
6     return lineEdit;
7 }
```

2.4.3 Проверка номера телефона

Для проверки номера телефона используется класс Delegate5. В нем задано регулярное выражение, обеспечивающее ввод согласно шаблону:

1. Знак '+'
2. От одной до трех цифр
3. Знак '('
4. От одной до трех цифр
5. Знак ')'
6. Семь цифр

Исходный код:

```
1 QWidget* Delegate5::createEditor(QWidget*parent, const QStyleOptionViewItem&,const QModelIndex
  &)const{
2     QLineEdit*lineEdit = new QLineEdit(parent);
3     QRegExp regExp("[+]{1}\\d{1,3}([]{1}\\d{1,3}){1}\\d{7})");
4     QValidator*validator = new QRegExpValidator(regExp,lineEdit);
5     lineEdit->setValidator(validator);
6     return lineEdit;
7 }
```

2.4.4 Проверка даты рождения

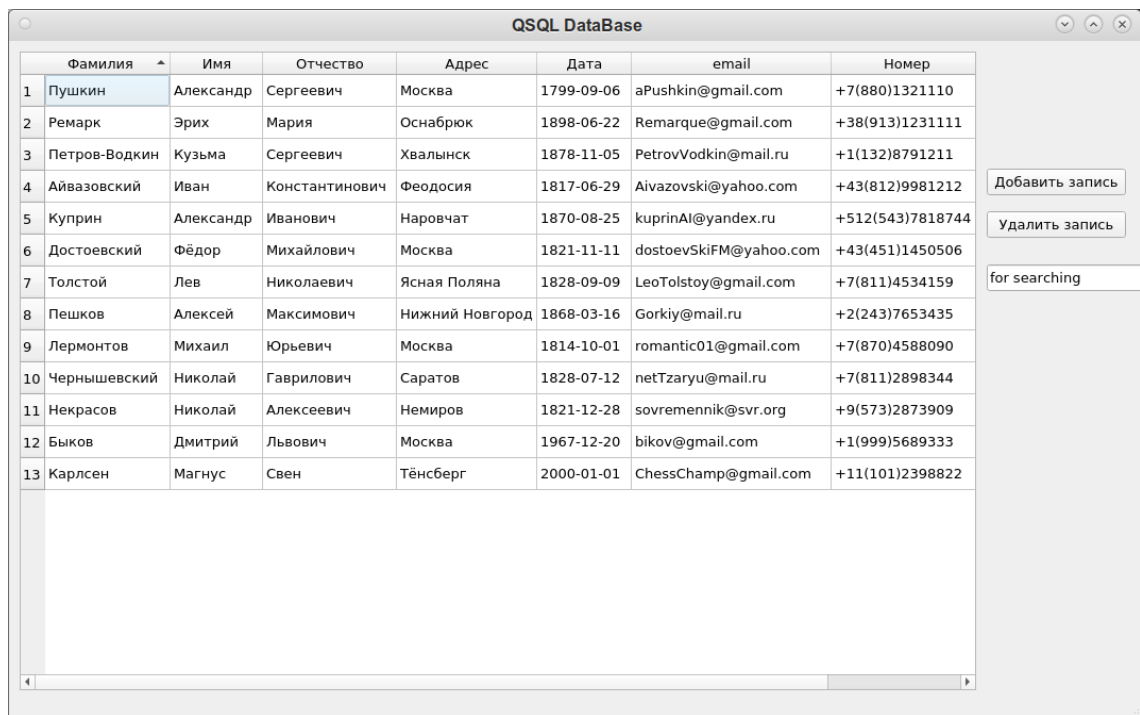
Для проверки даты рождения используется класс Delegate4. Это единственный класс-делегат, в котором не используются регулярные выражения. Проверка ввода заменяется изначальной установкой типа QDateEdit, который самостоятельно обеспечивает соответствие количества дней месяцам и проверяет високосный ли год. Дополнительно устанавливается ограничение на максимально допустимую к выставлению дату - она не должна превышать текущую.

Исходный код:

```
1 QWidget *Delegate4::createEditor(QWidget *parent, const QStyleOptionViewItem &, const
   QModelIndex &) const
2 {
3     QDateEdit*dateEdit = new QDateEdit(parent);
4     dateEdit->setMaximumDate(QDate::currentDate());
5     return dateEdit;
6 }
```


3 Тестирование программы

На рис.1 представлено окно сразу после запуска программы. База данных загружается автоматически из файла Contacts.db, если такой файл существует. Если такого файла не существует, создается пустая база данных.

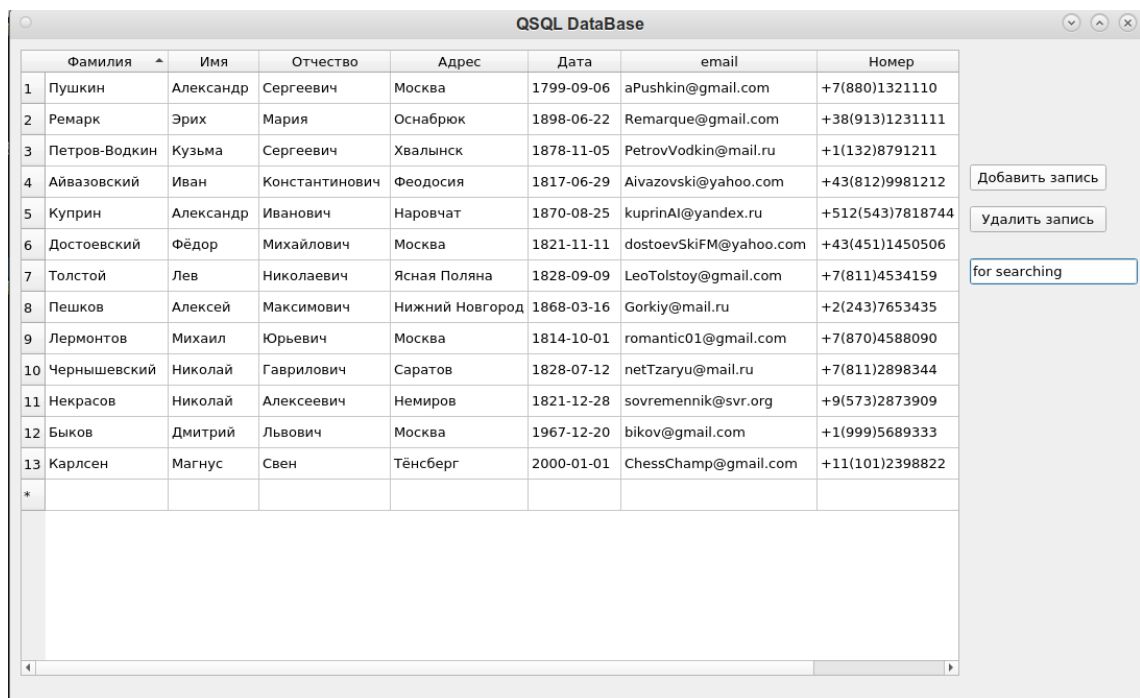


The screenshot shows the QSQL DataBase application window. It contains a table with 7 columns: Фамилия, Имя, Отчество, Адрес, Дата, email, and Номер. The table lists 13 contacts. To the right of the table are two buttons: «Добавить запись» and «Удалить запись», and a text input field labeled «for searching».

	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
2	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
3	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
4	Айвазовский	Иван	Константинович	Феодосия	1817-06-29	Aivazovski@yahoo.com	+43(812)9981212
5	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAl@yandex.ru	+512(543)7818744
6	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevSkiFM@yahoo.com	+43(451)1450506
7	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
8	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
9	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
10	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
11	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
12	Быков	Дмитрий	Львович	Москва	1967-12-20	bikov@gmail.com	+1(999)5689333
13	Карлсен	Магнус	Свен	Тёнсберг	2000-01-01	ChessChamp@gmail.com	+11(101)2398822

Рис. 1: Запуск программы

На рис.2 изображено добавление новой пустой строки с помощью нажатия кнопки «Добавить запись».

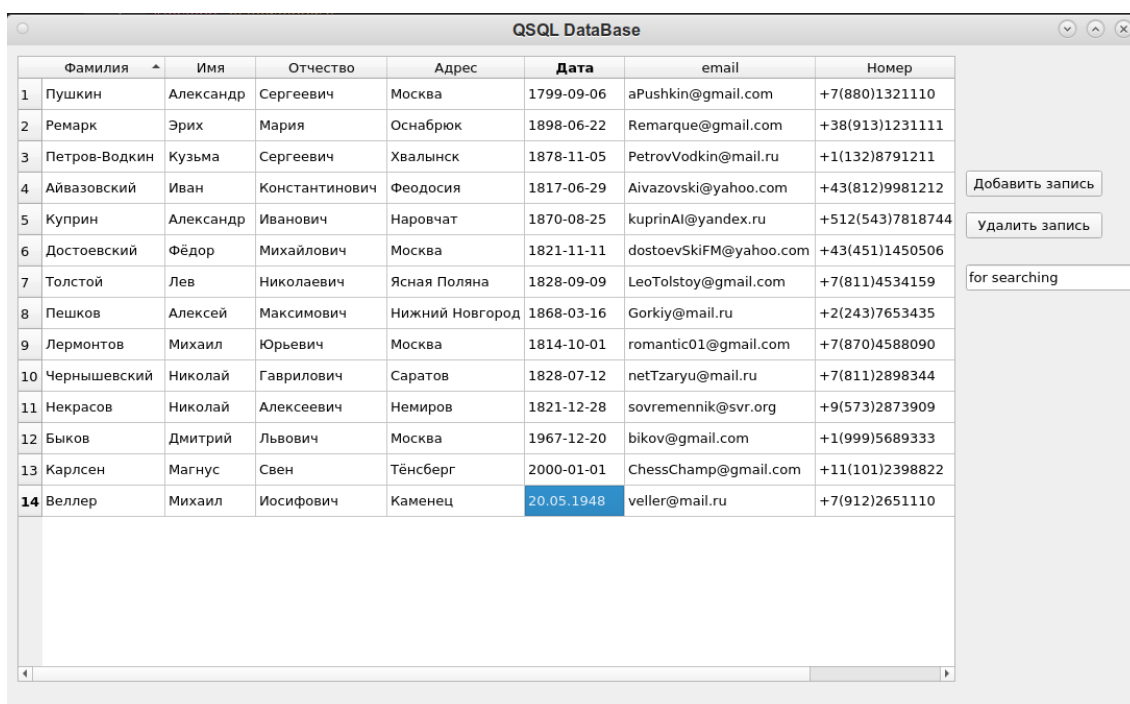


The screenshot shows the QSQL DataBase application window after clicking the «Добавить запись» button. A new empty row with an asterisk (*) in the first column has been added to the table. The search field now contains the text «for searching».

	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
2	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
3	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
4	Айвазовский	Иван	Константинович	Феодосия	1817-06-29	Aivazovski@yahoo.com	+43(812)9981212
5	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAl@yandex.ru	+512(543)7818744
6	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevSkiFM@yahoo.com	+43(451)1450506
7	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
8	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
9	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
10	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
11	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
12	Быков	Дмитрий	Львович	Москва	1967-12-20	bikov@gmail.com	+1(999)5689333
13	Карлсен	Магнус	Свен	Тёнсберг	2000-01-01	ChessChamp@gmail.com	+11(101)2398822
*							

Рис. 2: Добавление записи

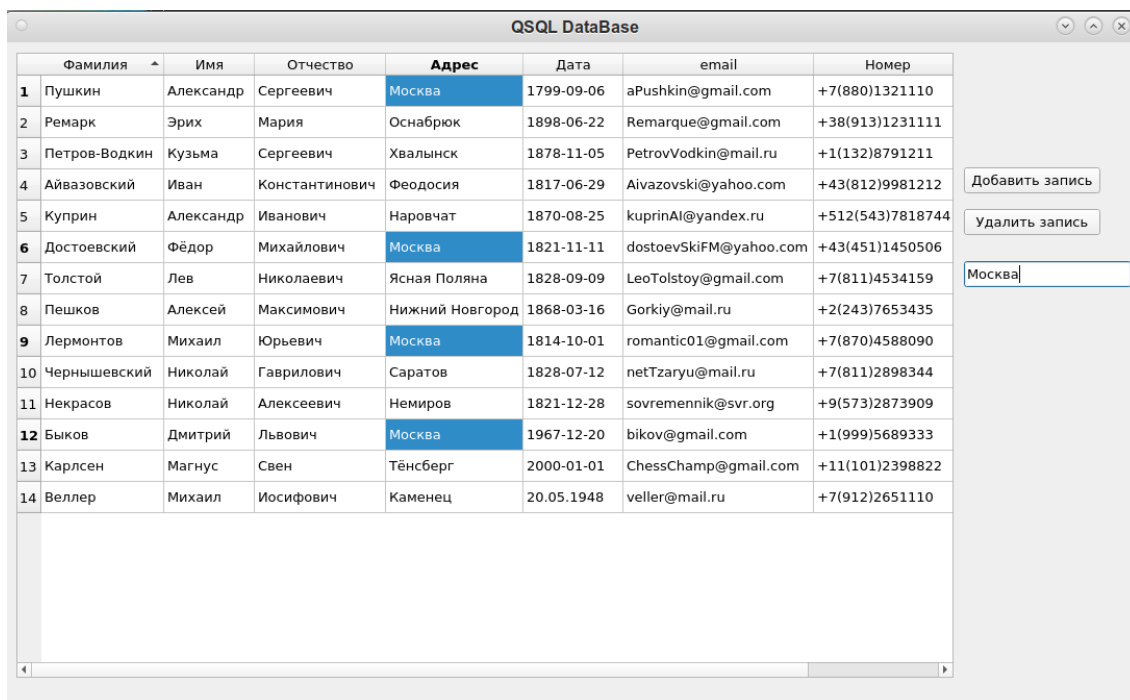
На рис. 3 представлено заполнение пустой строки новыми данными. Все введенные данные обязательно должны соответствовать требуемому формату. Если требуемый формат нарушается в вводимом символе, такой символ не отобразится и не будет введен.



	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
2	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
3	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
4	Айвазовский	Иван	Константинович	Феодосия	1817-06-29	Aivazovski@yahoo.com	+43(812)9981212
5	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAI@yandex.ru	+512(543)7818744
6	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevSkiFM@yahoo.com	+43(451)1450506
7	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
8	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
9	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
10	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
11	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
12	Быков	Дмитрий	Львович	Москва	1967-12-20	bikov@gmail.com	+1(999)5689333
13	Карлсен	Магнус	Свен	Тёнсберг	2000-01-01	ChessChamp@gmail.com	+11(101)2398822
14	Веллер	Михаил	Иосифович	Каменец	20.05.1948	veller@mail.ru	+7(912)2651110

Рис. 3: Заполнение

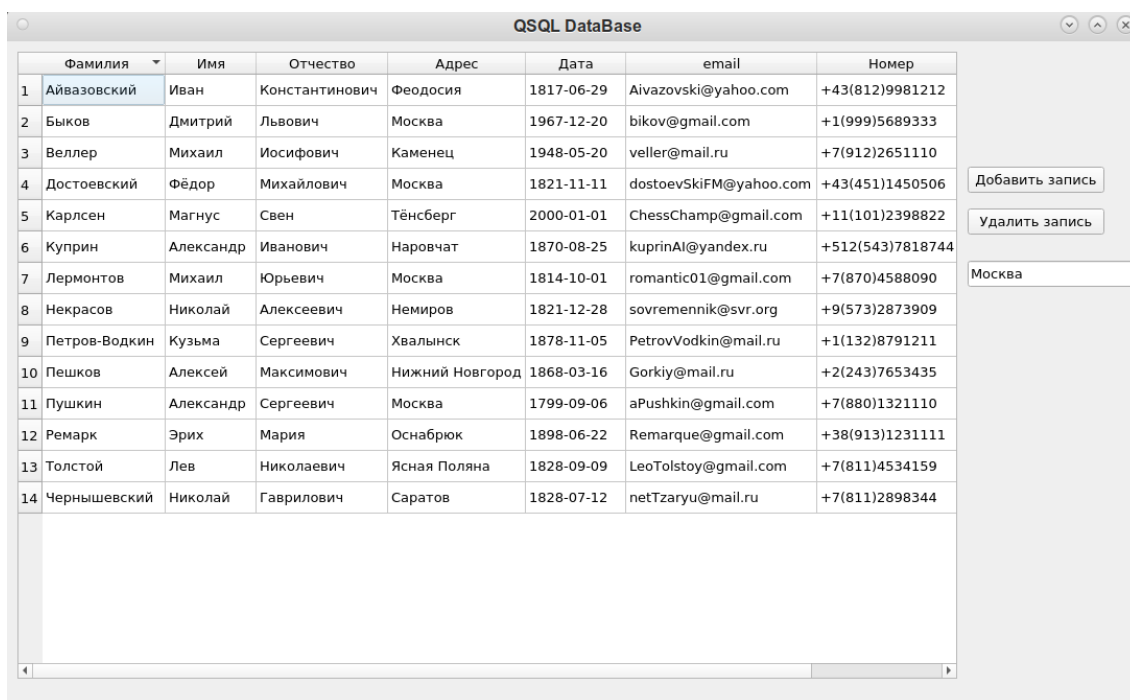
На рис.4 представлен поиск по слову «Москва». Для поиска используется окно ввода. После окончания ввода требуется нажать клавишу «Enter». Поиск осуществляется по полному соответствию. При нахождении соответствия найденные ячейки станут выделены.



	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
2	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
3	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
4	Айвазовский	Иван	Константинович	Феодосия	1817-06-29	Aivazovski@yahoo.com	+43(812)9981212
5	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAI@yandex.ru	+512(543)7818744
6	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevSkiFM@yahoo.com	+43(451)1450506
7	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
8	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
9	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
10	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
11	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
12	Быков	Дмитрий	Львович	Москва	1967-12-20	bikov@gmail.com	+1(999)5689333
13	Карлсен	Магнус	Свен	Тёнсберг	2000-01-01	ChessChamp@gmail.com	+11(101)2398822
14	Веллер	Михаил	Иосифович	Каменец	20.05.1948	veller@mail.ru	+7(912)2651110

Рис. 4: Поиск по слову «Москва»

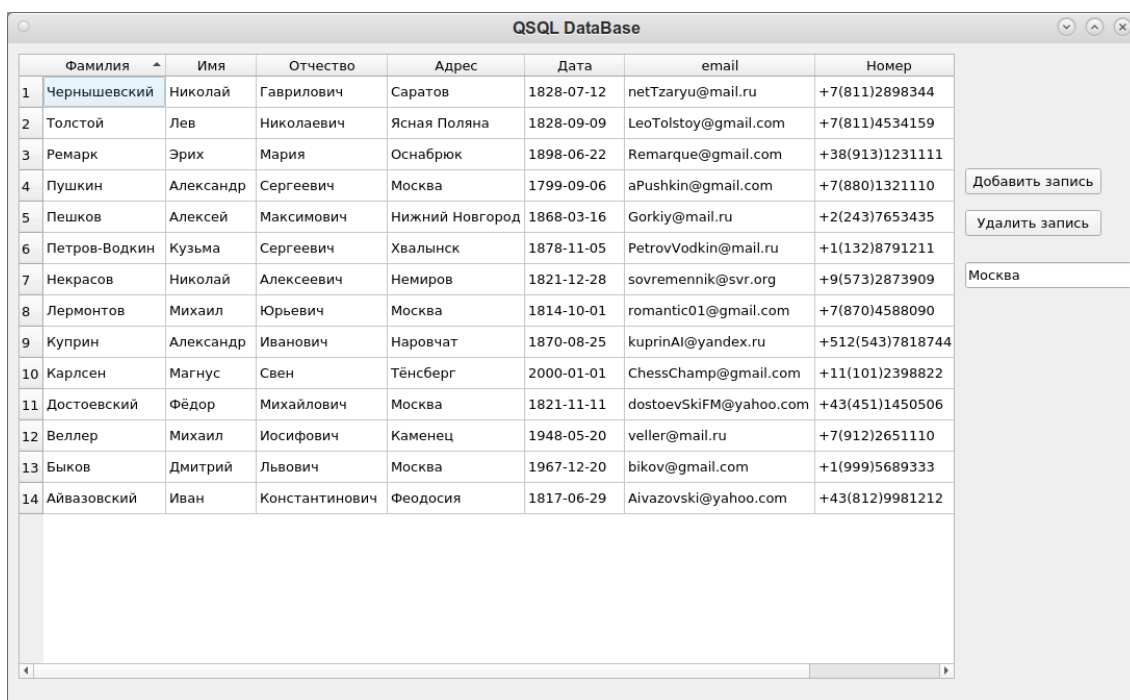
На рис.5 представлена сортировка данных по полю «Фамилия» в возрастающем лексикографическом порядке. Для использования сортировки необходимо нажать на название столбца.



	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Айвазовский	Иван	Константинович	Феодосия	1817-06-29	Aivazovski@yahoo.com	+43(812)9981212
2	Быков	Дмитрий	Львович	Москва	1967-12-20	bikov@gmail.com	+1(999)5689333
3	Веллер	Михаил	Иосифович	Каменец	1948-05-20	veller@mail.ru	+7(912)2651110
4	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevskiFM@yahoo.com	+43(451)1450506
5	Карлсен	Магнус	Свен	Тёнсберг	2000-01-01	ChessChamp@gmail.com	+11(101)2398822
6	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAI@yandex.ru	+512(543)7818744
7	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
8	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
9	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
10	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
11	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
12	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
13	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
14	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344

Рис. 5: Сортировка по фамилии от А до Я

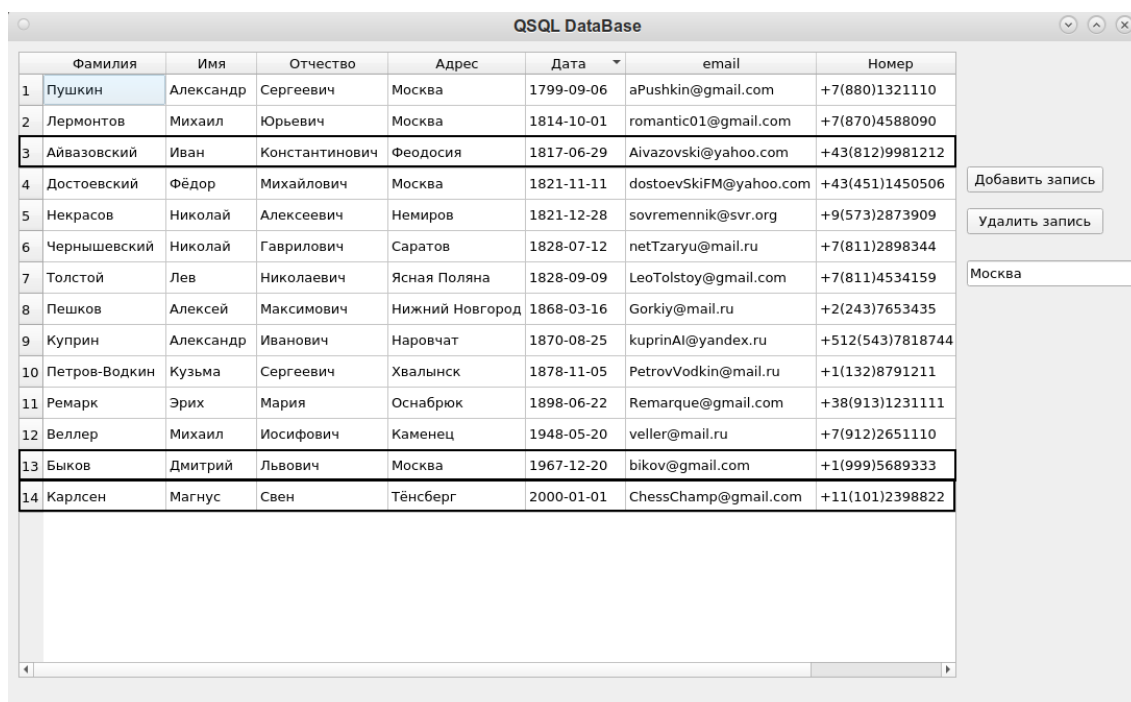
На рис.6 изображена аналогичная сортировка по полю «Фамилия» в обратном порядке.



	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
2	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
3	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
4	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
5	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
6	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
7	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
8	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
9	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAI@yandex.ru	+512(543)7818744
10	Карлсен	Магнус	Свен	Тёнсберг	2000-01-01	ChessChamp@gmail.com	+11(101)2398822
11	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevskiFM@yahoo.com	+43(451)1450506
12	Веллер	Михаил	Иосифович	Каменец	1948-05-20	veller@mail.ru	+7(912)2651110
13	Быков	Дмитрий	Львович	Москва	1967-12-20	bikov@gmail.com	+1(999)5689333
14	Айвазовский	Иван	Константинович	Феодосия	1817-06-29	Aivazovski@yahoo.com	+43(812)9981212

Рис. 6: Сортировка по фамилии от Я до А

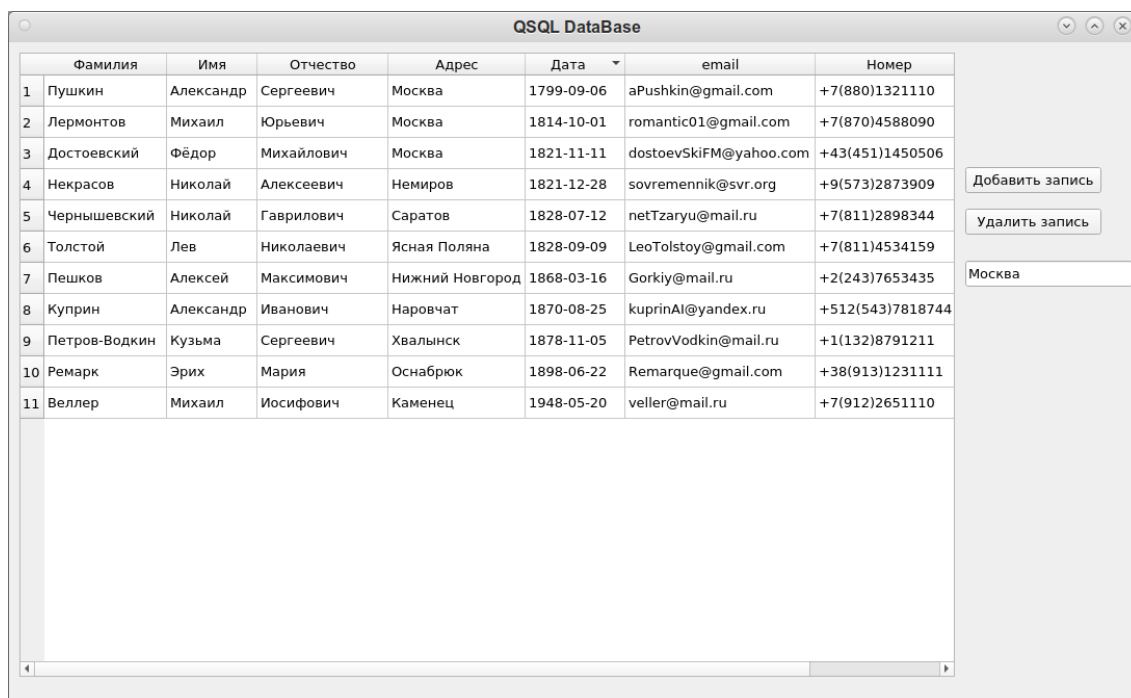
На рис.7 представлена сортировка по полю «Дата». Для сортировки также используется нажатие на наименование столца. Строки выделены для следующего рисунка.



	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
2	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
3	Айвазовский	Иван	Константинович	Феодосия	1817-06-29	Aivazovski@yahoo.com	+43(812)9981212
4	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevskiFM@yahoo.com	+43(451)1450506
5	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
6	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
7	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
8	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
9	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAl@yandex.ru	+512(543)7818744
10	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
11	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
12	Веллер	Михаил	Иосифович	Каменец	1948-05-20	veller@mail.ru	+7(912)2651110
13	Быков	Дмитрий	Львович	Москва	1967-12-20	bikov@gmail.com	+1(999)5689333
14	Карлсен	Магнус	Свен	Тёнсберг	2000-01-01	ChessChamp@gmail.com	+11(101)2398822

Рис. 7: Сортировка по дате рождения

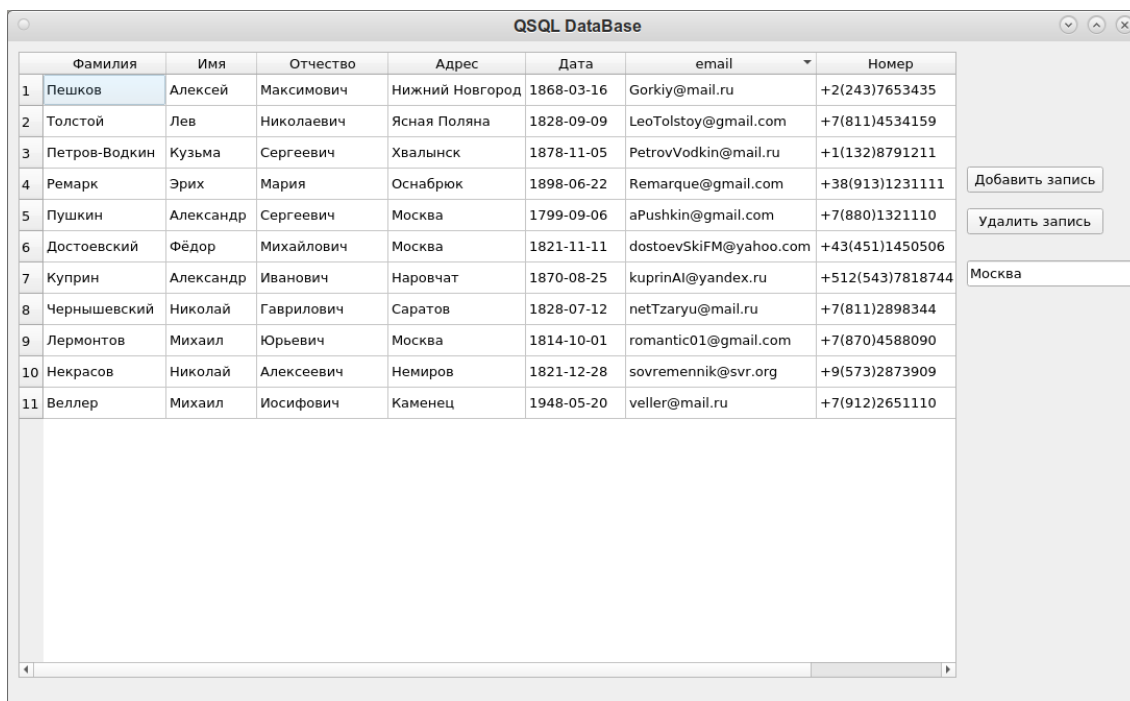
На рис.8 представлено удаление строк 3,13,14, выделенных на рис.7. Для удаления требуется выделить строку или ячейку строки и нажать кнопку «Удалить запись»



	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
2	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
3	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevskiFM@yahoo.com	+43(451)1450506
4	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
5	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
6	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
7	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
8	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAl@yandex.ru	+512(543)7818744
9	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
10	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
11	Веллер	Михаил	Иосифович	Каменец	1948-05-20	veller@mail.ru	+7(912)2651110

Рис. 8: Удаление записей 3,13,14

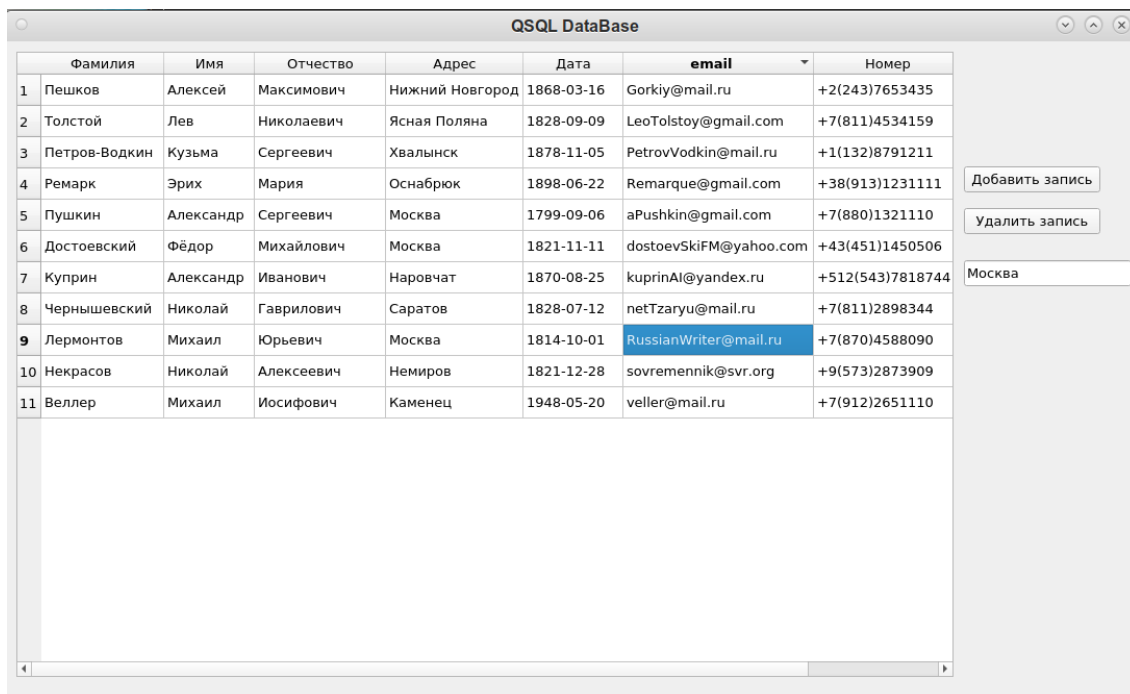
Для дополнительной проверки стабильности работы программы на рис.9 представлен результат сортировки по полю «email» после процедуры удаления строк. Как видно из рисунка, программа работает корректно.



	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
2	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
3	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
4	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
5	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
6	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevskiFM@yahoo.com	+43(451)1450506
7	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAI@yandex.ru	+512(543)7818744
8	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
9	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	romantic01@gmail.com	+7(870)4588090
10	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
11	Веллер	Михаил	Иосифович	Каменец	1948-05-20	veller@mail.ru	+7(912)2651110

Рис. 9: Сортировка после удаления

На рис.10 представлено изменение поля «email». После окончания ввода необходимо нажать «Enter».



	Фамилия	Имя	Отчество	Адрес	Дата	email	Номер
1	Пешков	Алексей	Максимович	Нижний Новгород	1868-03-16	Gorkiy@mail.ru	+2(243)7653435
2	Толстой	Лев	Николаевич	Ясная Поляна	1828-09-09	LeoTolstoy@gmail.com	+7(811)4534159
3	Петров-Водкин	Кузьма	Сергеевич	Хвалынский	1878-11-05	PetrovVodkin@mail.ru	+1(132)8791211
4	Ремарк	Эрих	Мария	Оснабрюк	1898-06-22	Remarque@gmail.com	+38(913)1231111
5	Пушкин	Александр	Сергеевич	Москва	1799-09-06	aPushkin@gmail.com	+7(880)1321110
6	Достоевский	Фёдор	Михайлович	Москва	1821-11-11	dostoevskiFM@yahoo.com	+43(451)1450506
7	Куприн	Александр	Иванович	Наровчат	1870-08-25	kuprinAI@yandex.ru	+512(543)7818744
8	Чернышевский	Николай	Гаврилович	Саратов	1828-07-12	netTzaryu@mail.ru	+7(811)2898344
9	Лермонтов	Михаил	Юрьевич	Москва	1814-10-01	RussianWriter@mail.ru	+7(870)4588090
10	Некрасов	Николай	Алексеевич	Немиров	1821-12-28	sovremennik@svr.org	+9(573)2873909
11	Веллер	Михаил	Иосифович	Каменец	1948-05-20	veller@mail.ru	+7(912)2651110

Рис. 10: Изменение значения ячейки

Заключение

В результате выполнения курсовой работы было реализовано оконное приложение «Телефонный справочник», обеспечивающее работу с базой данных.

- Реализация оконного интерфейса была сделана с помощью фреймворка Qt
- Реализация добавления/редактирования записей была сделана с помощью средств виджета TableView и механизма сигналов/слотов.
- Реализация проверки всех вводимых значений на соответствие заданному шаблону была выполнена с помощью классов-делегатов, регулярный выражений и типа QDateEdit.
- Реализация сортировки была выполнена с помощью встроенного механизма сортировки виджета TableView.
- Реализация сохранения базы данных и загрузки из файла была реализована классами второго уровня модуля QSql.
- Реализация поиска по значению была реализована с помощью механизма сигналов/слотов и обхода всех данных двумя циклами.

Работа выполнялась в среде QtCreator v.5.12.0 GCC 64bit в операционной системе Linux 5.10.0-9-amd64.

Список источников

- [1] Макс Шлее Qt 5.10 Профессиональное программирование на C++. 1-е изд. 1072с – СПб.: БХВ-Петербург, 2018г.
- [2] Qt Assistant Manual - Qt Documentation

Приложение. Исходный код

3.1 main.cpp

```
1 #include "mainwindow.h"
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     MainWindow w;
8     w.show();
9     return a.exec();
10 }
```

3.2 mainwindow.h

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3 #include <QMainWindow>
4 #include <QtSql/QtSqlDatabase>
5 #include <QtSql/QtSqlQuery>
6 #include <QtSql/QtSqlTableModel>
7 #include <QDebug>
8 #include <QLineEdit>
9 #include <QTableWidget>
10 #include <QSqlRecord>
11 #include <QSqlField>
12 #include <QModelIndex>
13 #include <QThread>
14 QT_BEGIN_NAMESPACE
15 namespace Ui { class MainWindow; }
16 QT_END_NAMESPACE
17
18 class MainWindow : public QMainWindow
19 {
20     Q_OBJECT
21
22 public:
23     MainWindow(QWidget *parent = nullptr);
24     ~MainWindow();
25
26 public slots:
27     void on_pushButton_clicked();
28     void on_pushButton_2_clicked();
29     void on_tableView_clicked(const QModelIndex&index);
30     void on_lineEdit_editingFinished();
31     void on_lineEdit_textEdited(const QString &arg1);
32
33 private:
34     int rowSelected;
35     Ui::MainWindow *ui;
36     QSqlDatabase database;
37     QSqlQuery*query;
38     QSqlTableModel*db_model;
39 };
40 #endif // MAINWINDOW_H
```

3.3 mainwindow.cpp

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include "delegate0.h"
4 #include "delegate4.h"
5 #include "delegate5.h"
6 #include "delegateem.h"
7 MainWindow::MainWindow(QWidget *parent)
8     : QMainWindow(parent)
9     , ui(new Ui::MainWindow)
10 {
11     ui->setupUi(this);
12     QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
```



```

13 db.setDatabaseName("../ContactDB.db");
14 db.setUserName("userdefault");
15 db.setHostName("userdefault");
16 db.setPassword("password");
17 if(!db.open()){
18     qDebug() << "Database couldn't be opened";
19 }
20 query = new QSqlQuery(db);
21 query->exec("CREATE TABLE ContactsФамилия( TEXT, Имя TEXT, Отчество TEXT, Адрес TEXT, Дата
рождения TEXT, email TEXT, Номер TEXT);");
22 db_model = new QSqlTableModel(this,db);
23 db_model->setTable("Contacts");
24 db_model->select();
25 ui->tableView->setModel(db_model);
26 ui->tableView->setItemDelegateForColumn(0,new Delegate0());
27 ui->tableView->setItemDelegateForColumn(1,new Delegate0());
28 ui->tableView->setItemDelegateForColumn(2,new Delegate0());
29 ui->tableView->setItemDelegateForColumn(4,new Delegate4());
30 ui->tableView->setItemDelegateForColumn(6,new Delegate5());
31 ui->tableView->setItemDelegateForColumn(5,new DelegateEm());
32 ui->tableView->resizeColumnsToContents();
33 ui->lineEdit->setText("for searching");
34 this->setWindowTitle("QSQL DataBase");
35 }
36 void MainWindow::on_pushButton_clicked(){
37     db_model->insertRow(db_model->rowCount());
38     return;
39 }
40 void MainWindow::on_pushButton_2_clicked(){
41     db_model->removeRow(rowSelected);
42     ui->tableView->update();
43     db_model->select();
44     return;
45 }
46 void MainWindow::on_tableView_clicked(const QModelIndex&index){
47     rowSelected=index.row();
48 }
49 MainWindow::~MainWindow()
50 {
51     delete ui;
52 }

```

3.4 delegate0.h

```

1 #ifndef DELEGATE0_H
2 #define DELEGATE0_H
3
4 #include <QItemDelegate>
5 #include <QWidget>
6 #include <QLineEdit>
7 class Delegate0 : public QItemDelegate
8 {
9     QWidget* createEditor(QWidget*parent, const QStyleOptionViewItem&,const QModelIndex&)const
10     ;
11 public:
12     Delegate0();
13 };
14 #endif // DELEGATE0_H

```

3.5 delegate0.cpp

```

1 #include "delegate0.h"
2
3 Delegate0::Delegate0()
4 {
5
6 }
7 QWidget* Delegate0::createEditor(QWidget*parent, const QStyleOptionViewItem&,const QModelIndex
&)const{
8     QLineEdit*lineEdit = new QLineEdit(parent);
9     QRegExp regExp("( [А- Я]{1}[а - я ё\\s-]{1,30}[а - я ё]{1}|[А-З]{1}[а-з\\s-]{1,30}[а-з]{1})
");
10     QValidator*validator = new QRegExpValidator(regExp,lineEdit);

```

```

11     lineEdit->setValidator(validator);
12     return lineEdit;
13 }

```

3.6 delegate4.h

```

1  #ifndef DELEGATE4_H
2  #define DELEGATE4_H
3
4  #include <QItemDelegate>
5  #include <QWidget>
6  #include <QDateEdit>
7  #include <QDateTime>
8  #include <QLineEdit>
9  class Delegate4 : public QItemDelegate
10 {
11     QWidget* createEditor(QWidget*parent, const QStyleOptionViewItem&,const QModelIndex&)const
12     ;
13 public:
14     explicit Delegate4(QObject *parent = nullptr);
15 };
16 #endif // DELEGATE4_H

```

3.7 delegate4.cpp

```

1  #include "delegate4.h"
2
3  QWidget *Delegate4::createEditor(QWidget *parent, const QStyleOptionViewItem &, const
    QModelIndex &) const
4  {
5      QDateEdit*dateEdit = new QDateEdit(parent);
6      dateEdit->setMaximumDate(QDate::currentDate());
7      return dateEdit;
8  }
9  Delegate4::Delegate4(QObject *parent) : QItemDelegate(parent)
10 {
11
12 }

```

3.8 delegate5.h

```

1  #ifndef DELEGATE5_H
2  #define DELEGATE5_H
3
4  #include <QItemDelegate>
5  #include <QWidget>
6  #include <QLineEdit>
7  class Delegate5 : public QItemDelegate
8  {
9      QWidget* createEditor(QWidget*parent, const QStyleOptionViewItem&,const QModelIndex&)const
10     ;
11 public:
12     explicit Delegate5(QObject *parent = nullptr);
13 };
14 #endif // DELEGATE5_H

```

3.9 delegate5.cpp

```

1  #include "delegate5.h"
2  #include <QDebug>
3  Delegate5::Delegate5(QObject *parent) : QItemDelegate(parent)
4  {
5
6  }
7
8  QWidget* Delegate5::createEditor(QWidget*parent, const QStyleOptionViewItem&,const QModelIndex
    &)const{

```

```

9     QLineEdit*lineEdit = new QLineEdit(parent);
10    QRegExp regExp("(\\+){1}\\d{1,3}([\\]{1}\\d{1,3}){1}\\d{7})");
11    QValidator*validator = new QRegExpValidator(regExp,lineEdit);
12    lineEdit->setValidator(validator);
13    return lineEdit;
14 }

```

3.10 delegateem.h

```

1  #ifndef DELEGATEEM_H
2  #define DELEGATEEM_H
3
4  #include <QItemDelegate>
5  #include<QLineEdit>
6
7  class DelegateEm : public QItemDelegate
8  {
9      QWidget* createEditor(QWidget*parent, const QStyleOptionViewItem&,const QModelIndex&)const
10     ;
11 public:
12     explicit DelegateEm(QObject *parent = nullptr);
13 };
14 #endif // DELEGATEEM_H

```

3.11 delegateem.cpp

```

1  #include "delegateem.h"
2
3  DelegateEm::DelegateEm(QObject *parent) : QItemDelegate(parent)
4  {
5
6  }
7  QWidget* DelegateEm::createEditor(QWidget*parent, const QStyleOptionViewItem&,const
8      QModelIndex&)const{
9      QLineEdit*lineEdit = new QLineEdit(parent);
10     QRegExp regExp("([0-9A-Za-z]*[\\]{1,}\\.[a-z]{1,})");
11     QValidator*validator = new QRegExpValidator(regExp,lineEdit);
12     lineEdit->setValidator(validator);
13     return lineEdit;
14 }

```