**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Отчет

По лабораторной работе №7

На тему: “Исследование способов реализации сетевых запросов в QT-приложениях”

По дисциплине: “Кроссплатформенное программирование”

Выполнил:

ст.гр. ИС/б-17-2

Долженко И.А.

Проверил:

Строганов В.А.

Севастополь

2021

1 ЦЕЛЬ РАБОТЫ

Исследование способов работы Qt-приложения с HTTP GET/POST запросами. Приобретение навыков разработки простейших сетевых приложений в Qt.

2 ПОСТАНОВКА ЗАДАЧИ

2.1. Изучить основные возможности модуля QtNetwork для реализации HTTP-запросов (выполняется в ходе самостоятельной подготовки к лабораторной работе).

2.2. Создать Qt GUI приложение.

2.3. В дизайнере создать форму и реализовать логику приложения по инструкции из методических указаний.

2.4. Исследовать работу программы при отправке GET и POST запросов, проанализировать формат получаемых HTTP-ответов.

3 ТЕКСТ ПРОГРАММЫ

mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QtNetwork/QNetworkAccessManager>

#include <QtNetwork/QNetworkReply>

#include <QJsonDocument>

#include <QJsonArray>

#include <QJsonObject>

#include <QStandardItemModel>

#include <QMessageBox>

#include <QDateTime>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

private:

Ui::MainWindow \*ui;

QNetworkAccessManager \*nam;

QStandardItemModel model;

QHash<QNetworkReply\*, int> hash;

int removedIndex;

private slots:

void **getUsers**();

void **getUsersCompleted**(QNetworkReply \*reply);

void **addUser**();

void **addUserCompleted**(QNetworkReply \*reply);

void **removeUser**();

void **removeUserCompleted**(QNetworkReply \*reply);

void **finished**(QNetworkReply \*reply);

};

#endif // MAINWINDOW\_H

mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

nam = new QNetworkAccessManager(this);

ui->tableView->*setModel*(*&model*);

this->getUsers();

connect(ui->buttonAdd, SIGNAL(clicked()), this, SLOT(addUser()));

connect(ui->buttonRemove, SIGNAL(clicked()), this, SLOT(removeUser()));

connect(nam, SIGNAL(finished(QNetworkReply\*)), this, SLOT(finished(QNetworkReply\*)));

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**getUsers**()

{

QNetworkReply \*reply = nam->get(QNetworkRequest(QUrl("http://localhost:3000/users")));

hash[reply] = 1;

}

void MainWindow::**addUser**()

{

if (ui->lineEdit->text() != "" && ui->lineEdit\_2->text() != "" && ui->lineEdit\_3->text() != "")

{

QString fullname = "fullname=" + ui->lineEdit->text() + "&";

QString age = "age=" + ui->lineEdit\_2->text() + "&";

QString address = "address=" + ui->lineEdit\_3->text() + "&";

QString createdAt = "created\_at=" + QDateTime().currentDateTime().toString();

QByteArray postData;

postData.append(fullname.toUtf8());

postData.append(age.toUtf8());

postData.append(address.toUtf8());

postData.append(createdAt.toUtf8());

QNetworkRequest request;

request.setUrl(QUrl("http://localhost:3000/users"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/x-www-form-urlencoded");

QNetworkReply \*reply = nam->post(request, postData);

hash[reply] = 2;

}

else

{

QMessageBox msgBox;

msgBox.setWindowTitle("Ошибка");

msgBox.setText("Заполните все поля");

msgBox.setStandardButtons(QMessageBox::Ok);

msgBox.*exec*();

}

}

void MainWindow::**removeUser**()

{

QItemSelectionModel \*select = ui->tableView->selectionModel();

if (select->hasSelection())

{

QString id = select->selectedRows(0).value(0).data().toString();

removedIndex = select->selectedIndexes().first().row();

QNetworkReply \*reply = nam->deleteResource(QNetworkRequest(QUrl("http://localhost:3000/users/" + id)));

hash[reply] = 3;

}

}

void MainWindow::**finished**(QNetworkReply \*reply)

{

switch(hash[reply])

{

case 1:

getUsersCompleted(*reply*);

break;

case 2:

addUserCompleted(*reply*);

break;

case 3:

removeUserCompleted(*reply*);

break;

}

hash.remove(reply);

}

void MainWindow::**getUsersCompleted**(QNetworkReply \*reply)

{

if (reply->error() == QNetworkReply::NoError)

{

QJsonDocument doc = QJsonDocument::fromJson(reply->readAll());

QJsonArray data = doc.array();

QStringList headers = QStringList() << "ID" << "Fullname" << "Age" << "Address" << "Created At";

model.setHorizontalHeaderLabels(headers);

model.setColumnCount(headers.size());

const int rowCount = data.count();

const int columnCount = headers.size();

model.setRowCount(rowCount);

model.setColumnCount(columnCount);

for (int i = 0; i < rowCount; i++)

{

QJsonObject row = data[i].toObject();

model.setItem(i, 0, new QStandardItem*(*QString*::number(row[*"id"*].toInt()))*);

model.setItem(i, 1, new QStandardItem*(row[*"fullname"*].toString())*);

model.setItem(i, 2, new QStandardItem*(row[*"age"*].toString())*);

model.setItem(i, 3, new QStandardItem*(row[*"address"*].toString())*);

model.setItem(i, 4, new QStandardItem*(row[*"created\_at"*].toString())*);

}

}

else

{

QMessageBox msgBox;

msgBox.setWindowTitle("Ошибка");

msgBox.setText("Упс, сервер не отвечает");

msgBox.setStandardButtons(QMessageBox::Ok);

msgBox.*exec*();

}

}

void MainWindow::**addUserCompleted**(QNetworkReply \*reply)

{

if (reply->error() == QNetworkReply::NoError)

{

QJsonDocument doc = QJsonDocument::fromJson(reply->readAll());

QJsonObject data = doc.object();

QList<QStandardItem\*> newRow;

newRow.append(new QStandardItem(QString::number(data["id"].toInt())));

newRow.append(new QStandardItem(data["fullname"].toString()));

newRow.append(new QStandardItem(data["age"].toString()));

newRow.append(new QStandardItem(data["address"].toString()));

newRow.append(new QStandardItem(data["created\_at"].toString()));

model.appendRow(newRow);

ui->lineEdit->setText("");

ui->lineEdit\_2->setText("");

ui->lineEdit\_3->setText("");

}

else

{

QMessageBox msgBox;

msgBox.setWindowTitle("Ошибка");

msgBox.setText("Упс, сервер не отвечает");

msgBox.setStandardButtons(QMessageBox::Ok);

msgBox.*exec*();

}

}

void MainWindow::**removeUserCompleted**(QNetworkReply \*reply)

{

if (reply->error() == QNetworkReply::NoError)

{

model.removeRow(removedIndex);

}

else

{

QMessageBox msgBox;

msgBox.setWindowTitle("Ошибка");

msgBox.setText("Упс, сервер не отвечает");

msgBox.setStandardButtons(QMessageBox::Ok);

msgBox.*exec*();

}

}

4 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Начальное окно приложения представлено на рисунке 1. При старте приложения делается запрос на сервер и в таблицу выводится список пользователей.

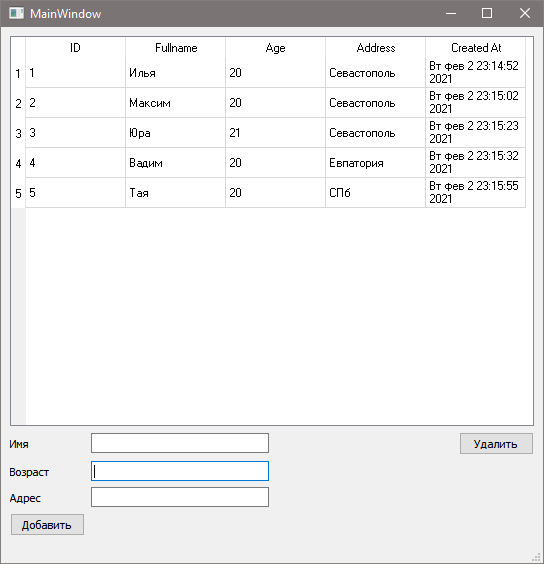


Рисунок 1 – Результат выполнения запроса на получение пользователей

Добавим пользователя с именем “Виктор”, введя данные в форму и выполнив Post запрос на http://localhost:3000/users. Результат выполнения запроса представлен на рисунке 2.

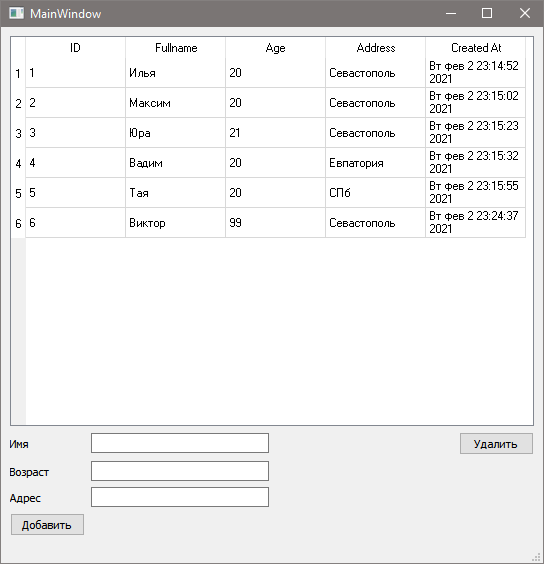


Рисунок 2 – Результат добавления нового пользователя

Теперь удалим пользователя с именем “Юра” из таблицы, путем выбора необходимой строки и нажатия кнопки удаления. Результат представлен на рисунке 3.

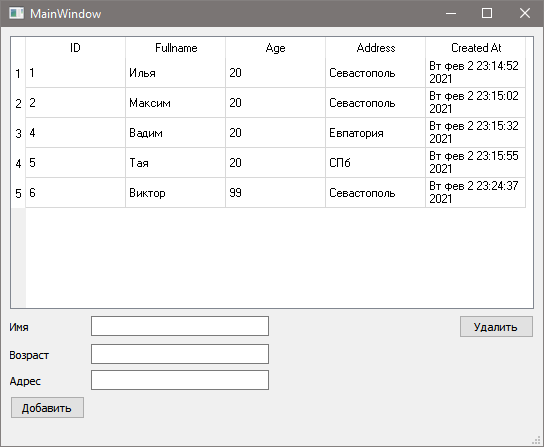


Рисунок 3 – Результат удаления пользователя

ВЫВОДЫ

В ходе выполнения лабораторной работы была написана программа, которая выполняет HTTP GET/POST/DELETE запросы с помощью класса QNetworkAccessManager. Он содержит общую конфигурацию и настройки для посылаемых запросов. Также был изучен объект QNetworkReply, который используется для получения любых данных, возвращаемых в ответ на соответствующий запрос.