

1. Цель и задачи

Цель: познакомиться с основами работы REST приложений.

Задачи:

* изучить основы работы с REST запросами;
* создать простейшее приложение для получения данных по API со стороннего сайта.

1. Задание

Общее ограничение на лабораторную работу:

* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.

Требуется написать программу, которая по REST протоколу получает публичные данные с сайта http://vk.com/.

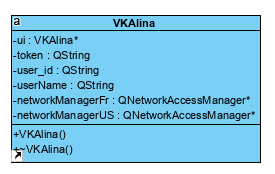
Этапы создания приложения:

* создать новое приложение в настройках Вконтакте;
* получить сервисный токен для отправки запросов на API Вконтакте;
* изучить API Вконтакте (https://vk.com/dev/methods).

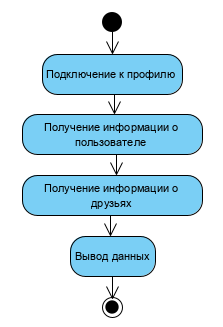
Само приложение должно предоставлять функцию ввода уникального ID профиля и кнопку его получения. При нажатии на кнопку данные должны показаться в заранее подготовленной форме. Данные, которые необходимо вывести (если у профиля они не заданы или скрыты, оставить пустую строку):

* фамилия;
* имя;
* ник-нейм;
* день рождения;
* пол;
* фотография профиля;
* список друзей (минимум ID каждого друга).

1. ДИАГРАММЫ



*Рисунок 1 – диаграмма классов*

**

*Рисунок 2 – диаграмма деятельности функции получения странички пользователя*

1. Код программы

Основной выполняемый файл main.cpp:

#include "widget.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

VKOreh w;

w.show();

return a.exec();

}

Файл widget.cpp:

#include <QUrl>

#include <QDebug>

#include <QString>

#include "ui\_widget.h"

#include "widget.h"

#include <QUrlQuery>

#include <QLabel>

#include <QLibrary>

#include <QJsonDocument>

#include <QJsonObject>

#include <QJsonArray>

#include <QNetworkReply>

VKOreh::VKOreh(QWidget \*parent) :

QWidget(parent),

ui(new Ui::VKOreh)

{

ui->setupUi(this);

connect(this, SIGNAL(auth\_success()),this,SLOT(on\_Poisk\_clicked()));

}

VKOreh::~VKOreh()

{

delete ui;

}

void VKOreh::on\_Poisk\_clicked()

{

userName = ui->ID\_users->text();

get\_user();

get\_friends();

}

void VKOreh::onResult(QNetworkReply \*reply)

{

ui->textEdit->clear();

if(!reply->error()){

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

QJsonObject root = document.object();

QJsonObject jv = root["response"].toObject();

QJsonArray ja = jv["items"].toArray();

for(int i = 0; i < ja.count(); i++){

QJsonObject subtree = ja.at(i).toObject();

QString q = QString::number(subtree.value("id").toInt());

ui->textEdit->append("id: "+ q);

ui->textEdit->append(subtree.value("first\_name").toString()+" "+subtree.value("last\_name").toString());

}

}

reply->deleteLater();

}

void VKOreh::onResultUser(QNetworkReply \*replyUs)

{

if(!replyUs->error()){

QJsonDocument document = QJsonDocument::fromJson(replyUs->readAll());

QJsonObject root = document.object();

QJsonArray ja = root["response"].toArray();

for(int i = 0; i < ja.count(); i++){

QJsonObject subtree = ja.at(i).toObject();

ui->label\_2->setText(subtree.value("first\_name").toString());

ui->label\_5->setText(subtree.value("last\_name").toString());

ui->label\_7->setText(subtree.value("bdate").toString());

if(subtree.value("sex").toInt() == 2){ui->label\_9->setText("Мужской");}

else{ui->label\_9->setText("Женский");}

ui->Photo->load(QUrl(subtree.value("photo\_max\_orig").toString()));

}

}

replyUs->deleteLater();

}

void VKOreh::get\_user()

{

QUrl current("https://api.vk.com/method/users.get");

QUrlQuery Param;

Param.addQueryItem("user\_id", userName);

Param.addQueryItem("access\_token",token);

Param.addQueryItem("v","5.2");

Param.addQueryItem("fields","photo\_max\_orig,bdate, sex, online");

current.setQuery(Param);

networkManagerUS = new QNetworkAccessManager();

connect(networkManagerUS, &QNetworkAccessManager::finished, this, &VKOreh::onResultUser);

networkManagerUS->get(QNetworkRequest(current));

}

void VKOreh::get\_friends()

{

QUrl current("https://api.vk.com/method/friends.get");

QUrlQuery Param;

Param.addQueryItem("user\_id", userName);

Param.addQueryItem("access\_token",token);

Param.addQueryItem("v","5.2");

Param.addQueryItem("fields","1");

current.setQuery(Param);

networkManagerFr = new QNetworkAccessManager();

connect(networkManagerFr, &QNetworkAccessManager::finished, this, &VKOreh::onResult);

networkManagerFr->get(QNetworkRequest(current));

}

Файл widget.h:

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QWidget>

#include <QUrl>

#include <QString>

#include <QUrlQuery>

#include <QNetworkAccessManager>

#include <QNetworkReply>

#include <QNetworkRequest>

#include <QTimer>

#include <QEventLoop>

#include <QLabel>

#include <QLineEdit>

namespace Ui {

class VKOreh;

}

class VKOreh: public QWidget

{

Q\_OBJECT

public:

explicit VKOreh(QWidget \*parent = 0);

~VKOreh();

signals:

void auth\_success();

private:

Ui::VKOreh \*ui;

QString token = "6021c2296021c2296021c229826051322c660216021c2293eaeb0e9bd93804e08cfe920";

QString user\_id;

QString userName;

QNetworkAccessManager \*networkManagerFr;

QNetworkAccessManager \*networkManagerUS;

private slots:

void on\_Poisk\_clicked();

void onResult(QNetworkReply \*reply);

void onResultUser(QNetworkReply \*replyUs);

public slots:

void get\_user();

void get\_friends();

};

#endif // WIDGET\_H

В качестве результата работы программы представлен скриншот выходного файла на рисунке 3.

*Рисунок 3 – результат работы программы*

# ВЫВОД

В процессе выполнения лабораторной работы были изучены основы работы с REST запросами.