МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КЛИЕНТ-СЕРВЕРНОЕ ОБНОВЛЕНИЕ DLL

ОТЧЁТ ЛАБОРАТОРНОЙ РАБОТЫ №4
ПО ДИСЦИПЛИНЕ «ОПЕРАЦИОННЫЕ СИСТЕМЫ И СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ»
ЗА II СЕМЕСТР

Выполнил:
студент 3-го курса
V-го семестр
факультета ЭИС
группы ПО-4(1)
зачётная книжка №190333
Галанин П. И.
«__» _____ 2021 г.

Проверил:
ассистент
кафедры ИИТ
Дряпко А. В.
«__» _____ 2021 г.

Брест 2021

Цель работы: ознакомиться с возможностями, предлагаемыми Qt для поддержки сетевого взаимодействия программ

1 Сервер

Сервер возвращает JSON с версиями и dll файлы.

- http://localhost:3002/versions - получаем JSON
- http://localhost:3002/gpi_helper.dll - загрузка dll
- http://localhost:3002/gpi_helper_class.dll - загрузка dll
- http://localhost:3002/gpi_about.dll - загрузка dll

Листинг: ../gpi_server/package.json

```
{
    "dependencies": {
        "express": "^4.17.1",
        "nodemon": "^2.0.12"
    },
    "scripts": {
        "start": "nodemon gpi_app.js"
    }
}
```

Листинг: ../gpi_server/gpi_versions.json

```
{
    "helper": "1.0.0",
    "helper_class": "1.0.0",
    "about": "1.0.0"
}
```

Листинг: ../gpi_server/gpi_app.js

```
const path = require('path');
const express = require('express');

const app = express();
app.listen(3002);
console.log(`Open http://localhost:3002/`);

app.use(express.static(path.join(__dirname, 'public')));

app.get("/", function (req, res) {
    res.send("Hello, world!");
});

app.get("/versions", function (req, res) {
    res.send(require('./gpi_versions.json'));
});
```

2 Клиент проверяет обновления

По нажати кнопки обновить программу, будет произведена проверка версий. При различии версий будет скачан новый dll и обновлен файл с версиями (gpi_versions.json).

Листинг: ../gpi_osisp5_lab4/gpi_osisp5_option5/gpi_updater.cpp

```
1  #include "gpi_mainwindow.hpp"
2
3  #include <QJsonDocument>
4  #include <QJsonObject>
5  #include <QJsonArray>
6  #include <QUrlQuery>
7  #include <QNetworkReply>
8  #include <QUrl>
9  #include <QFile>
10
11 void gpi_MainWindow::on_gpi_pushButton_update_clicked()
12 {
13     qDebug() << "on_gpi_pushButton_update_clicked()";
14     QMessageBox::information(this, "Процесс обновления", "Происходит запрос на сервер");
15     networkManager = new QNetworkAccessManager();
16
17     // Подключаем networkManager к обработчику ответа
18     connect(networkManager, &QNetworkAccessManager::finished, this, &gpi_MainWindow::onResult);
19
20     // Получаем данные, а именно JSON файл с сайта по определённому url
21     networkManager->get(QNetworkRequest(QUrl("http://localhost:3002/versions")));
22 }
23
24 void gpi_MainWindow::onResult(QNetworkReply *reply)
25 {
26 QString client_helper;
27 QString client_helper_class;
28 QString client_about;
29
30 QString server_helper;
31 QString server_helper_class;
32 QString server_about;
33
34 QFile file;
35 QString file_inner;
36
37 QJsonDocument client_document;
38 QJsonObject client_json;
39
40 QJsonDocument server_document;
41 QJsonObject server_json;
42
43 QString gpi_str_verions;
44 // = = = = = = = = = = = = = = = = =
45     if (reply->error() != QNetworkReply::NoError)
46     {
47         QMessageBox::critical(this, "Проблема со сервером", "Ошибка подключения к серверу");
48     }
49
50     // Если ошибки отсутсвуют
51     if(!reply->error())
52     {
53         QMessageBox::information(this, "Процесс обновления", "Происходит сверка версий");
54
```

```
55          // = = = Работа с клиентом = = =
56
57          file.setFileName(":/@/gpi_versions.json"); // Назначаем имя
58          file.open(QIODevice::ReadOnly | QIODevice::Text); // Открываем файл
59          file_inner = file.readAll(); // Читаем текст из файла
60          file.close(); // Закрываем файл
61          qDebug() << file_inner;
62
63          client_document = QJsonDocument::fromJson(file_inner.toUtf8());
64          client_json = client_document.object();
65
66          client_helper = client_json["helper"].toString();
67          client_helper_class = client_json["helper_class"].toString();
68          client_about = client_json["about"].toString();
69
70          // = = = Работа с сервером = = =
71
72          // То создаём объект Json Document, считав в него все данные из ответа
73          server_document = QJsonDocument::fromJson(reply->readAll());
74          server_json = server_document.object();
75          qDebug() << "document:" << server_document;
76          qDebug() << "json:" << server_json;
77
78          server_helper = server_json.value("helper").toString();
79          server_helper_class = server_json.value("helper_class").toString();
80          server_about = server_json.value("about").toString();
81
82          // = = = Сравнение версий = = =
83
84          gpi_str_verions
85              = client_helper + " ~ " + server_helper + " - helper.dll \n"
86              + client_helper_class + " ~ " + server_helper_class + " - helper_class.dll \n"
87              + client_about + " ~ " + server_about  + " - about.dll \n";
88
89          QMessageBox::information(this, "Процесс обновления", gpi_str_verions);
90
91          if (client_helper.compare(server_helper)) {
92              QMessageBox::information(this, "Процесс обновления", "Найдено обновление helper.dll");
93              Downloader* dd = new Downloader();
94              dd->getData_helper();
95          }
96
97          if (client_helper_class.compare(server_helper_class)) {
98              QMessageBox::information(this, "Процесс обновления", "Найдено обновление
        helper_class.dll");
99              Downloader* dd = new Downloader();
100             dd->getData_helper_class();
101         }
102
103         if (client_about.compare(server_about)) {
104             QMessageBox::information(this, "Процесс обновления", "Найдено обновление about.dll");
105             Downloader* dd = new Downloader();
106             dd->getData_about();
107         }
108
109         QString jsonString = server_document.toJson(QJsonDocument::Indented);
110         qDebug() << "==============================" << jsonString;
111         QFile file_version;
112         file_version.setFileName("gpi_versions.json");
113         file_version.open(QIODevice::WriteOnly | QIODevice::Text);
114         QTextStream stream( &file_version );
115         stream << jsonString;
116         file_version.close();
```

```
117
118         }
119         reply->deleteLater();
120     }
```

Листинг: ../gpi_osisp5_lab4/gpi_osisp5_option5/gpi_downloader.hpp

```
1   #ifndef DOWNLOADER_H
2   #define DOWNLOADER_H
3
4   #include <QObject>
5   #include <QNetworkAccessManager>
6   #include <QNetworkRequest>
7   #include <QNetworkReply>
8   #include <QFile>
9   #include <QUrl>
10  #include <QDebug>
11
12  class Downloader : public QObject
13  {
14      Q_OBJECT
15  public:
16      explicit Downloader(QObject *parent = 0);
17
18  signals:
19      void downloader_onReady();
20
21  public slots:
22      void getData_helper();      // Метод инициализации запроса на получение данных
23      void onResult_helper(QNetworkReply *reply);     // Слот обработки ответа о полученных данных
24      void getData_helper_class();      // Метод инициализации запроса на получение данных
25      void onResult_helper_class(QNetworkReply *reply);     // Слот обработки ответа о полученных данн
        ых
26      void getData_about();      // Метод инициализации запроса на получение данных
27      void onResult_about(QNetworkReply *reply);     // Слот обработки ответа о полученных данных
28
29  private:
30      QNetworkAccessManager *manager_helper; // менеджер сетевого доступа
31      QNetworkAccessManager *manager_helper_class; // менеджер сетевого доступа
32      QNetworkAccessManager *manager_about; // менеджер сетевого доступа
33  };
34
35  #endif // DOWNLOADER_H
```

Листинг: ../gpi_osisp5_lab4/gpi_osisp5_option5/gpi_downloader.cpp

```
1   #include "gpi_downloader.hpp"
2
3   Downloader::Downloader(QObject *parent) : QObject(parent)
4   {
5       qDebug() << "Downloader";
6       // Инициализируем менеджер ...
7       manager_helper = new QNetworkAccessManager();
8       manager_helper_class = new QNetworkAccessManager();
9       manager_about = new QNetworkAccessManager();
10      // ... и подключаем сигнал о завершении получения данных к обработчику полученного ответа
11      connect(manager_helper, &QNetworkAccessManager::finished, this, &Downloader::onResult_helper);
12      connect(manager_helper_class, &QNetworkAccessManager::finished, this,
        &Downloader::onResult_helper_class);
13      connect(manager_about, &QNetworkAccessManager::finished, this, &Downloader::onResult_about);
14  }
15
16  void Downloader::getData_helper()
```

```
17 {
18     qDebug() << "getData";
19     QUrl url("http://localhost:3002/gpi_helper.dll"); // URL, к которому будем получать данные
20     QNetworkRequest request;     // Отправляемый запрос
21     request.setUrl(url);         // Устанавлвиваем URL в запрос
22     manager_helper->get(request);       // Выполняем запрос
23 }
24
25 void Downloader::getData_helper_class()
26 {
27     qDebug() << "getData";
28     QUrl url("http://localhost:3002/gpi_helper_class.dll"); // URL, к которому будем получать данны
       е
29     QNetworkRequest request;     // Отправляемый запрос
30     request.setUrl(url);         // Устанавлвиваем URL в запрос
31     manager_helper_class->get(request);       // Выполняем запрос
32 }
33
34 void Downloader::getData_about()
35 {
36     qDebug() << "getData";
37     QUrl url("http://localhost:3002/gpi_about.dll"); // URL, к которому будем получать данные
38     QNetworkRequest request;     // Отправляемый запрос
39     request.setUrl(url);         // Устанавлвиваем URL в запрос
40     manager_about->get(request);       // Выполняем запрос
41 }
42
43 void Downloader::onResult_helper(QNetworkReply *reply)
44 {
45     qDebug() << "onResult";
46     // Если в процесе получения данных произошла ошибка
47     if(reply->error())
48     {
49         // Сообщаем об этом и показываем информацию об ошибках
50         qDebug() << "ERROR";
51         qDebug() << reply->errorString();
52     }
53     else
54     {
55         // В противном случае создаём объект для работы с файлом
56         QFile *file = new QFile("./gpi_helper.dll");
57         // Создаём файл или открываем его на перезапись ...
58         if(file->open(QFile::WriteOnly))
59         {
60             qDebug() << "write helper.dll";
61             file->write(reply->readAll());  // ... и записываем всю информацию со страницы в файл
62             file->close();                  // закрываем файл
63             qDebug() << "Downloading is completed";
64             emit downloader_onReady(); // Посылаем сигнал о завершении получения файла
65         }
66     }
67 }
68
69 void Downloader::onResult_helper_class(QNetworkReply *reply)
70 {
71     qDebug() << "onResult";
72     // Если в процесе получения данных произошла ошибка
73     if(reply->error())
74     {
75         // Сообщаем об этом и показываем информацию об ошибках
76         qDebug() << "ERROR";
77         qDebug() << reply->errorString();
78     }
```

```
79      else
80      {
81          // В противном случае создаём объект для работы с файлом
82          QFile *file = new QFile("./gpi_helper_class.dll");
83          // Создаём файл или открываем его на перезапись ...
84          if(file->open(QFile::WriteOnly))
85          {
86              qDebug() << "write helper_class.dll";
87              file->write(reply->readAll());  // ... и записываем всю информацию со страницы в файл
88              file->close();                  // закрываем файл
89              qDebug() << "Downloading is completed";
90              emit downloader_onReady(); // Посылаем сигнал о завершении получения файла
91          }
92      }
93  }
94
95  void Downloader::onResult_about(QNetworkReply *reply)
96  {
97      qDebug() << "onResult";
98      // Если в процесе получения данных произошла ошибка
99      if(reply->error())
100     {
101         // Сообщаем об этом и показываем информацию об ошибках
102         qDebug() << "ERROR";
103         qDebug() << reply->errorString();
104     }
105     else
106     {
107         // В противном случае создаём объект для работы с файлом
108         QFile *file = new QFile("./gpi_about.dll");
109         // Создаём файл или открываем его на перезапись ...
110         if(file->open(QFile::WriteOnly))
111         {
112             qDebug() << "write about.dll";
113             file->write(reply->readAll());  // ... и записываем всю информацию со страницы в файл
114             file->close();                  // закрываем файл
115             qDebug() << "Downloading is completed";
116             emit downloader_onReady(); // Посылаем сигнал о завершении получения файла
117         }
118     }
119 }
```

3 Новый уровень

Новый уровень записан в файле helper.dll.

Листинг: ../gpi_osisp5_lab4/gpi_helper/gpi_helper.h

```
1  #ifndef GPI_HELPER_H
2  #define GPI_HELPER_H
3
4  #include "gpi_helper_global.h"
5  #include <QStringList>
6
7  extern "C" __declspec(dllexport)
8  QStringList gpi_get_1_level();
9
```

```
10 extern "C" __declspec(dllexport)
11 QStringList gpi_get_2_level();
12
13 extern "C" __declspec(dllexport)
14 QStringList gpi_get_3_level();
15
16 extern "C" __declspec(dllexport)
17 QStringList gpi_get_4_level();
18
19 #endif // GPI_HELPER_H
```

Листинг: ../gpi_osisp5_lab4/gpi_helper/gpi_helper.cpp

```
1  #include "gpi_helper.h"
2
3  // gpi_ Функция, которая возвращает текстовый 1-ый уровень
4  QStringList gpi_get_1_level()
5  {
6  QStringList gpi_str_map;
7  // = = = = = = = = = = = = = = = = = =
8      gpi_str_map = {
9          "xxxxxxxxxxxxxxx",
10         "xxxxxxxxxxxxxxx",
11         "xxxxxxxxxxxxxxx",
12         "xxxxxxxxxxxxxxx",
13         "xxxxxx...xxxxxx",
14         "xxxxf@b..xxxxxx",
15         "xxxxxx.bfxxxxxx",
16         "xxxxfxxb.xxxxxx",
17         "xxxx.x.f.xxxxxx",
18         "xxxxb.Bbbfxxxxx",
19         "xxxx...f..xxxxx",
20         "xxxxxxxxxxxxxxx",
21         "xBxxxxxxxxxxxxx",
22         "xBxxxxxxxxxxxxx",
23         "xBxxxxxxxxxxxxx",
24         "xBxxxxxxxxxxxxx",
25         "xxxxxxxxxxxxxxx"
26     };
27     return gpi_str_map;
28 }
29
30 // gpi_ Функция, которая возвращает текстовый 2-ой уровень
31 QStringList gpi_get_2_level()
32 {
33 QStringList gpi_str_map;
34 // = = = = = = = = = = = = = = = = = =
35     gpi_str_map = {
36         "xxxxxxx.xxxxxxxxxxxxxxxxxxxxx",
37         "xxxxxxx.xxxxxxxxxxxxxxxxxxxxx",
38         "xx...xx.xx.................xx",
39         "xx....x.x.................xx",
40         "xx.........bbbbbbbbbb.....xx",
41         "xxx.......................xxx",
42         "xxxx.....................xxxx",
43         "........@.........b..........",
44         "xxxx.............fff.....xxxx",
45         "xxx....x.........fff......xxx",
46         "xx...............fff.......xx",
47         "xx....x.x........fff.......xx",
48         "xx...xx.xx.................xx",
49         "xxxxxxx.xxxxxxxxxxxxxxxxxxxxx",
50         "xxxxxxx.xxxxxxxxxxxxxxxxxxxxx",
51     };
```

```
52      return gpi_str_map;
53  }
54
55  // gpi_ Функция, которая возвращает текстовый 3-ий уровень
56  QStringList gpi_get_3_level()
57  {
58  QStringList gpi_str_map;
59  // = = = = = = = = = = = = = = = = = =
60      gpi_str_map = {
61          "xxxxxxxxxxxxxxx",
62          "xxxxxxxxxxxxxxx",
63          "xxxxxxxxxxxxxxx",
64          "xxxxxxxxxxxxxxx",
65          "xxxxxx...xxxxxx",
66          "xxxxf@b..xxxxxx",
67          "xxxxxx.bfxxxxxx",
68          "xxxxfxxb.xxxxxx",
69          "xxxx.x.f.xxxxxx",
70          "xxxxb.Bbbfxxxxx",
71          "xxxx...f..xxxxx",
72          "xxxxxxxxxxxxxxx",
73          "xBxBxBxxxxxxxxx",
74          "xBxBxBxxxxxxxxx",
75          "xBxBxBxxxxxxxxx",
76          "xBxBxBxxxxxxxxx",
77          "xxxxxxxxxxxxxxx"
78      };
79      return gpi_str_map;
80  }
81
82  // gpi_ Функция, которая возвращает текстовый 4-ый уровень
83  QStringList gpi_get_4_level()
84  {
85  QStringList gpi_str_map;
86  // = = = = = = = = = = = = = = = = = =
87      gpi_str_map = {
88          "xxxxxxxxxxxxxxx",
89          "xxxxxxxxxxxxxxx",
90          "xxxxxxxxxxxxxxx",
91          "xxxxxxxxxxxxxxx",
92          "xxxxxx...xxxxxx",
93          "xxxxf@b..xxxxxx",
94          "xxxxxx.bfxxxxxx",
95          "xxxxfxxb.xxxxxx",
96          "xxxx.x.f.xxxxxx",
97          "xxxxb.Bbbfxxxxx",
98          "xxxx...f..xxxxx",
99          "xxxxxxxxxxxxxxx",
100         "xBxBxBxxxxxxxxx",
101         "xBxBxBxxxxxxxxx",
102         "xBxxBxxxxxxxxxx",
103         "xBxxBxxxxxxxxxx",
104         "xxxxxxxxxxxxxxx"
105     };
106     return gpi_str_map;
107 }
```

Вывод: Ознакомился с возможностями Qt для межсерверного взаимодействия. Научился работать с JSON файлами. Научился отправлять запросы GET на сервер. Научился скачивать файл с сервера.