



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ8-34 \_\_\_\_\_

КАФЕДРА ИУ8 \_\_\_\_\_

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

**НА ТЕМУ:**

***Разработка электронного магазина компьютерных игр.***

Студент ИУ8-34  
(Группа)

  
(Подпись, дата)

Федоров В. М.  
(И.О.Фамилия)

Руководитель курсового проекта

\_\_\_\_\_  
(Подпись, дата)

Бородин А. А.  
(И.О.Фамилия)

2021 г.

УТВЕРЖДАЮ  
Заведующий кафедрой 1198  
(И.О.Фамилия)  
«18» » 03 2020 г.

## ЗАДАНИЕ на выполнение курсового проекта

по дисциплине «Технологии и методы программирования»

Студент группы ИУ8-34

Федоров Василий Михайлович  
(Фамилия, имя, отчество)

Тема курсового проекта *Разработка электронного магазина компьютерных игр.*  
Направленность КП (учебный, исследовательский, практический, производственный, др.)  
*практический*  
Источник тематики (кафедра, предприятие, НИР) *кафедра*

График выполнения проекта: 25% к 6 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

### **Задание**

Разработать ПО с графическим интерфейсом – «Электронного магазина компьютерных игр».  
ПО позволяет зарегистрировать новых пользователей и использовать в дальнейшем учетные записи пользователей. Пользователи могут покупать игры, добавлять/удалять игры в избранное, передавать свою копию игры другому пользователю, пополнять свой баланс. Предусмотрено стороннее ПО для добавления игр и установления цены на игры.



### **Оформление курсового проекта:**

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.  
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания «1» сентября 2020 г.

Руководитель курсового проекта

Студент

 18.09.20 А. А. Бородин  
(Подпись, дата) (И.О.Фамилия)  
 18.09.20 В. М. Федоров  
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

## Оглавление

<b>Вступление.....</b>	<b>4</b>
<b>Описание работы программы .....</b>	<b>5</b>
<b>База данных.....</b>	<b>12</b>
<b>GameStoreGUI.exe.....</b>	<b>15</b>
<b>GameStoreServerSide.exe .....</b>	<b>21</b>
<b>Заключение.....</b>	<b>23</b>

# Вступление

Целью проекта является создание электронного магазина компьютерных игр, в котором пользователи смогут регистрироваться, покупать игры, добавлять/удалять игры в избранное, передавать (дарить) свою копию игры другому пользователю, пополнять свой баланс. Так же пользователь может добавлять игры и устанавливать на них цены.

В ходе работы я должен:

- Получить представление о программировании ПО на C++
- Освоить сетевое взаимодействие программ, работу с серверами.
- Научится работать с фреймворком Qt

Используемые материалы в данной работе:

- C++ - распространенный язык программирования
- Qt – фреймворк для разработки ПО на языке программирования C++
- Microsoft Azure – облачная платформа, предоставляющая сервер для работы магазина игр
- sha256 – безопасный алгоритм хеширования

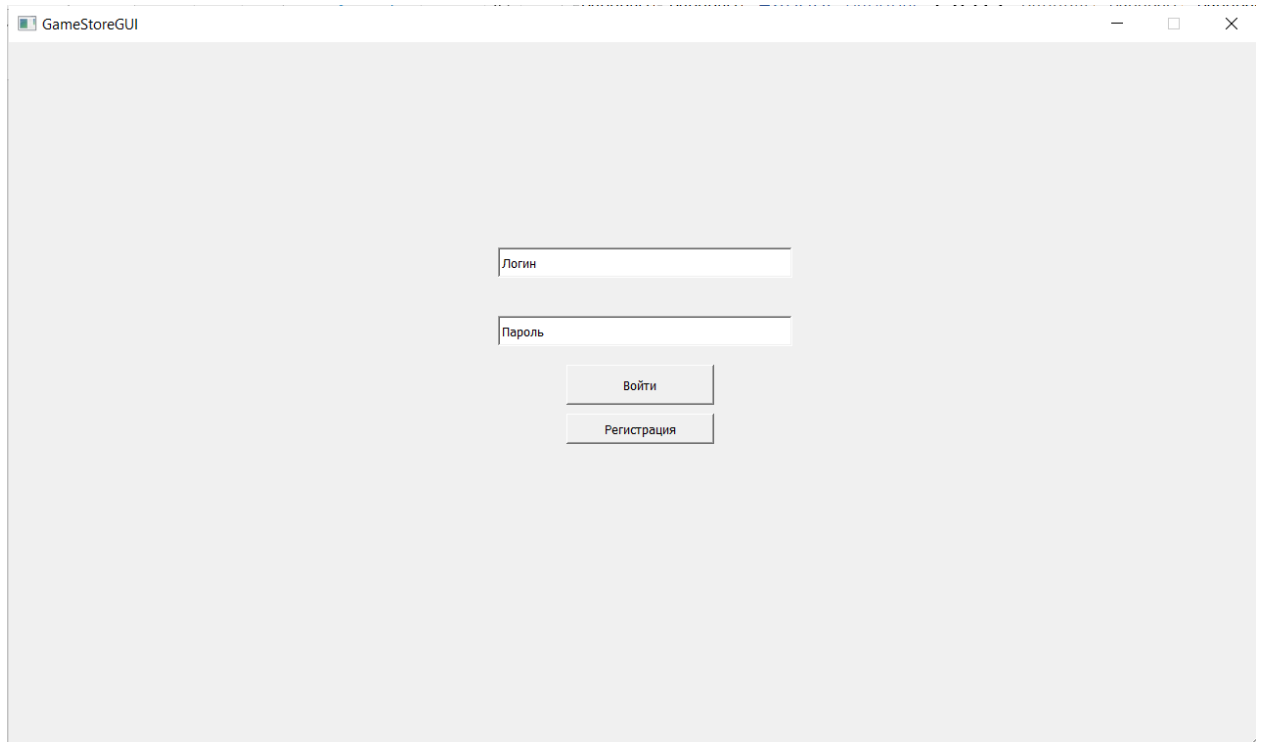
Причины использования данных материалов:

- C++ - язык, на котором я учусь работать уже продолжительное время и хочу лучше изучить его.
- Qt – качественный и распространенный фреймворк. Комплектуется визуальной средой разработки графического интерфейса.
- Microsoft Azure – позволяет удобно использовать сервер для расположения базы данных на нем и работы с электронным магазином.

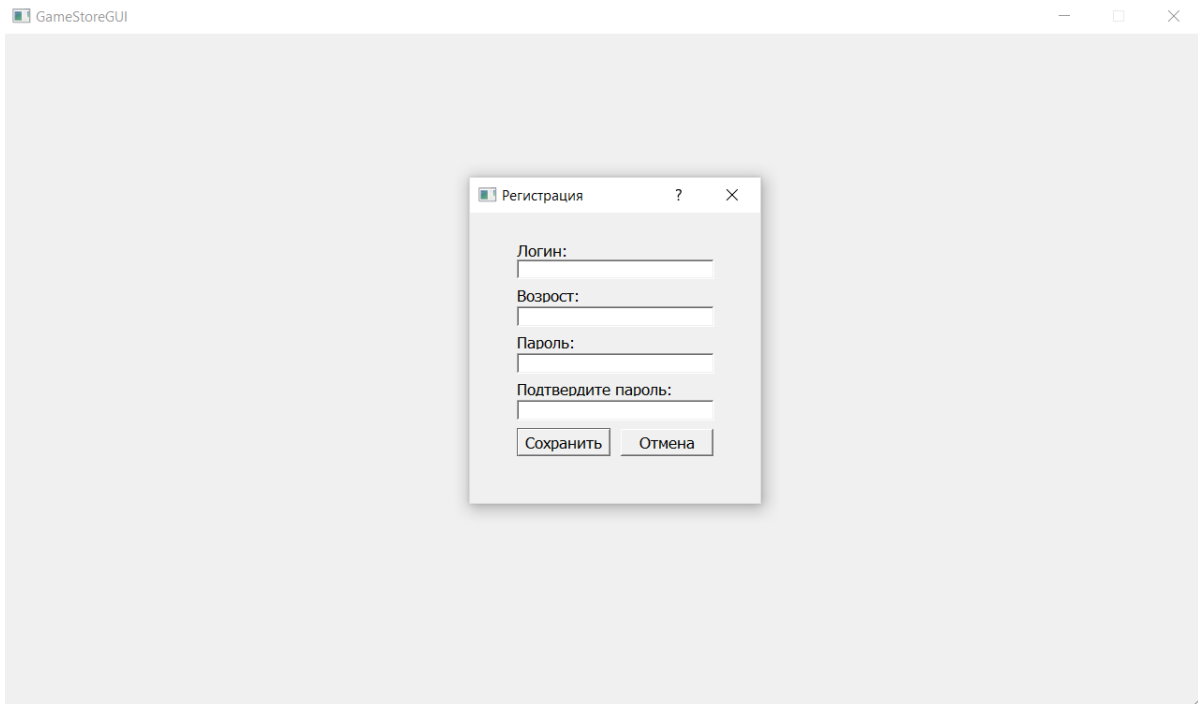
- sha256 – позволяет шифровать пароли и, в последствии, передавать и хранить их в зашифрованном виде.

## Описание работы программы

При запуске программы пользователя встречает диалоговое окно, позволяющее ему либо войти в уже имеющуюся учетную запись, либо создать новую (зарегистрироваться)



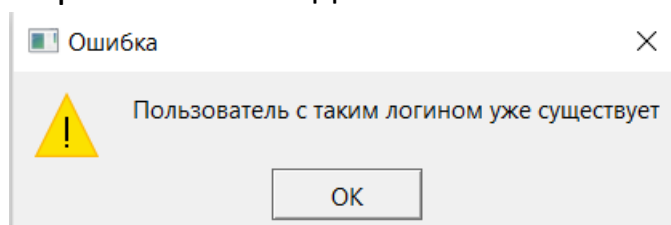
Давайте нажмем кнопку *регистрация*. Открылось новое диалоговое окно регистрации, при этом взаимодействие с первым окном временно заблокировано.



Окно предлагает пользователю ввести необходимые данные для регистрации.

При попытке сохранить данные нового пользователя, программа проверит корректность введенных данных и выдаст ошибку в следующих случаях:

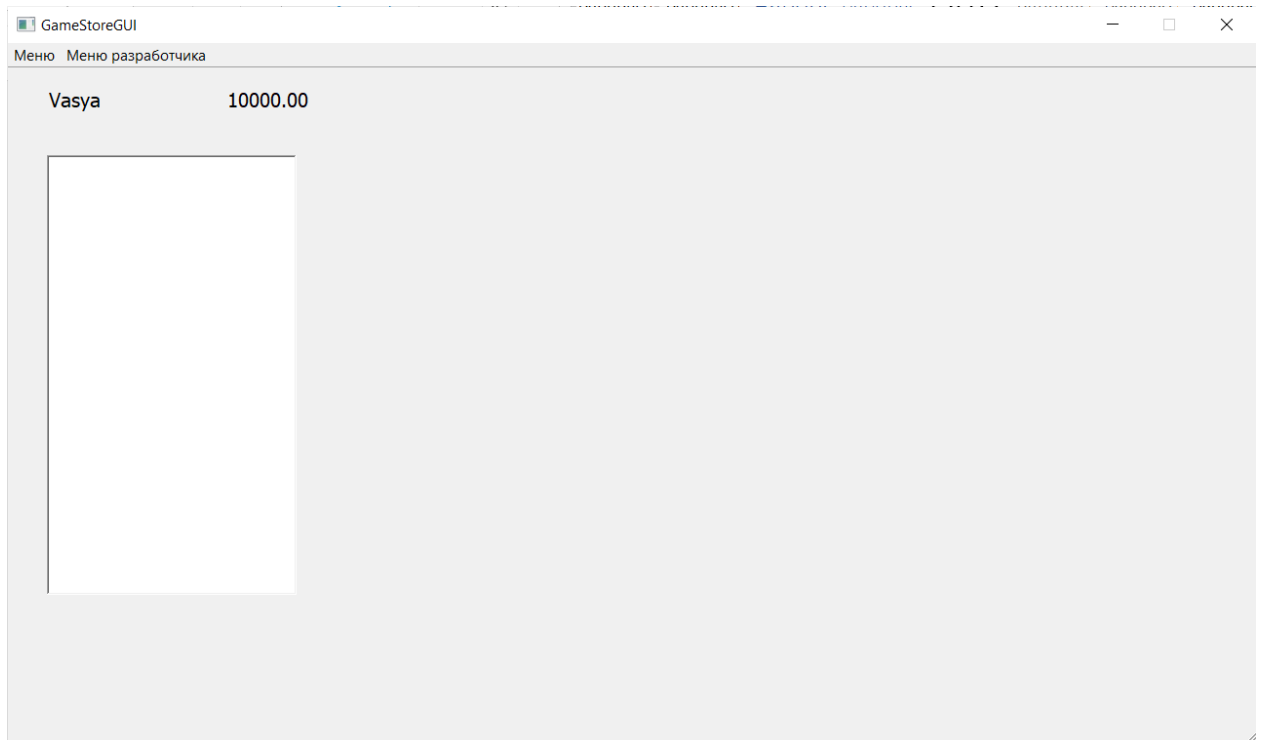
- Логин повторяется с логином уже существующего пользователя
- Возраст имеет значения отличные от целых чисел в промежутке от 3-х до 99-ти
- Пароль имеет длину менее 8-ми символов
- Пароли не совпадают



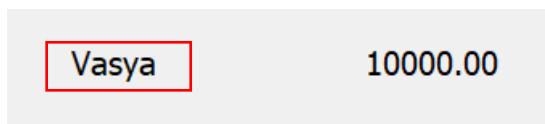
После закрытия окна ошибки пользователь сможет исправить введенные данные и создать пользователя.

При сохранении данных или нажатии кнопки *отмена* диалоговое окно регистрации закрывается, и пользователь может войти в свою учетную запись введя логин и пароль.

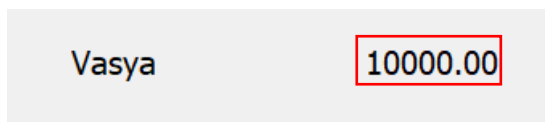
Войдя в свою учетную запись, пользователь видит следующее диалоговое окно.



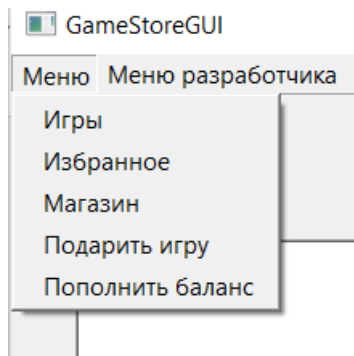
В левом верхнем углу отображается логин пользователя



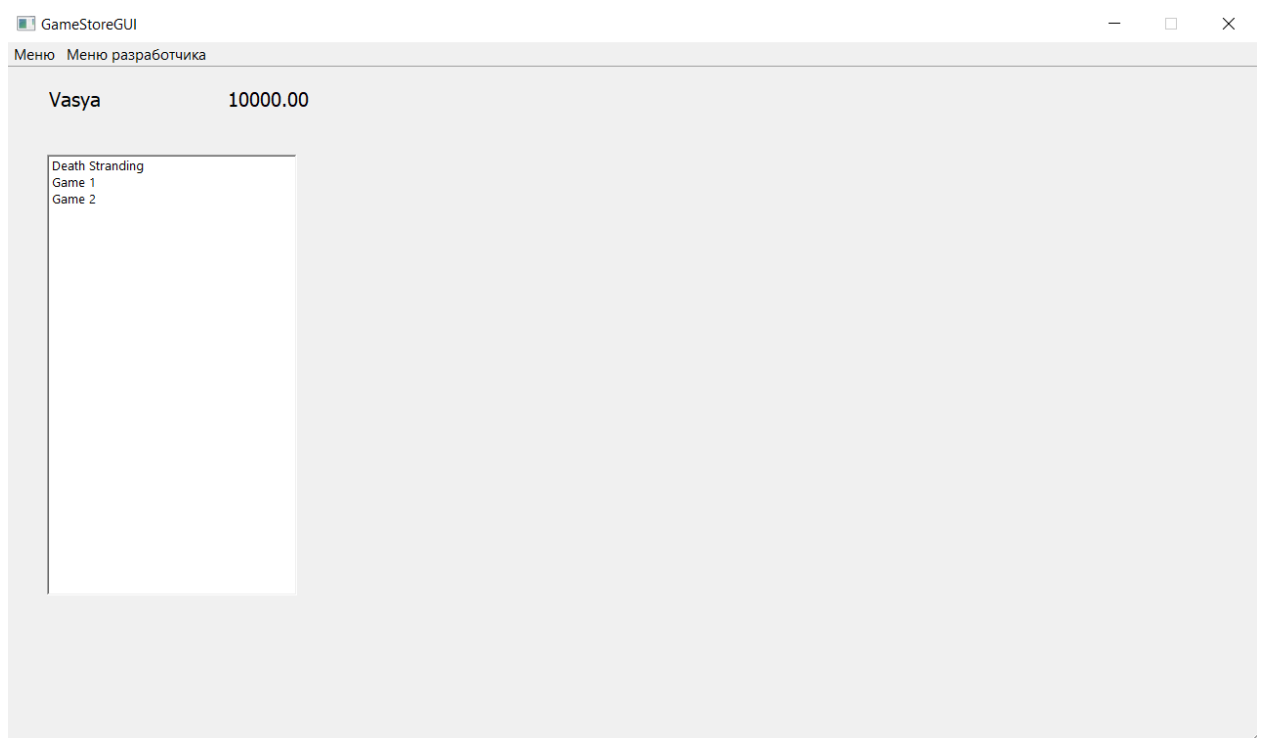
и баланс



При нажатии на кнопку *меню* появляются доступные команды.

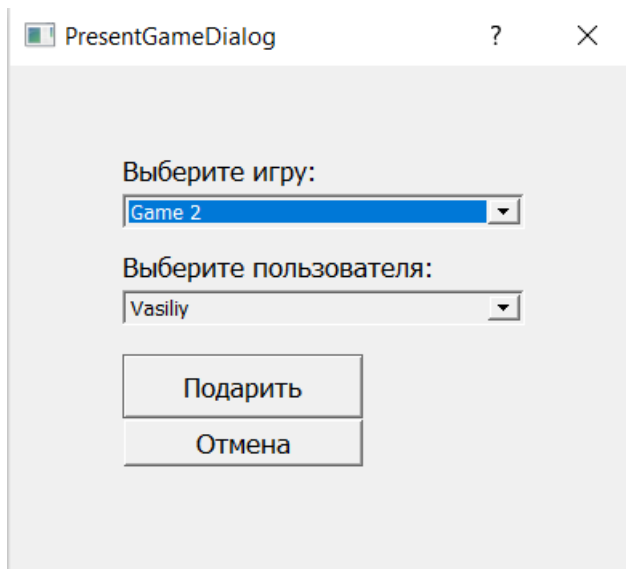


Кнопки *Игры*, *Избранное* и *Магазин* высвечивают соответственно игры, купленные пользователем, добавленные в избранное, и все существующие в магазине.



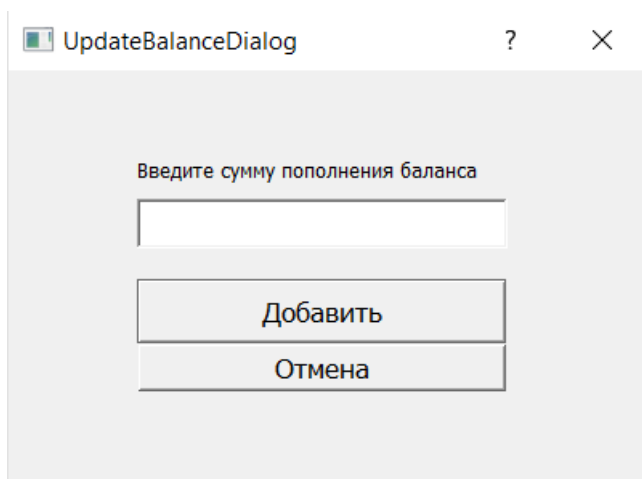
Если пользователь нажмет кнопку *Подарить игру*, то откроется диалоговое окно.



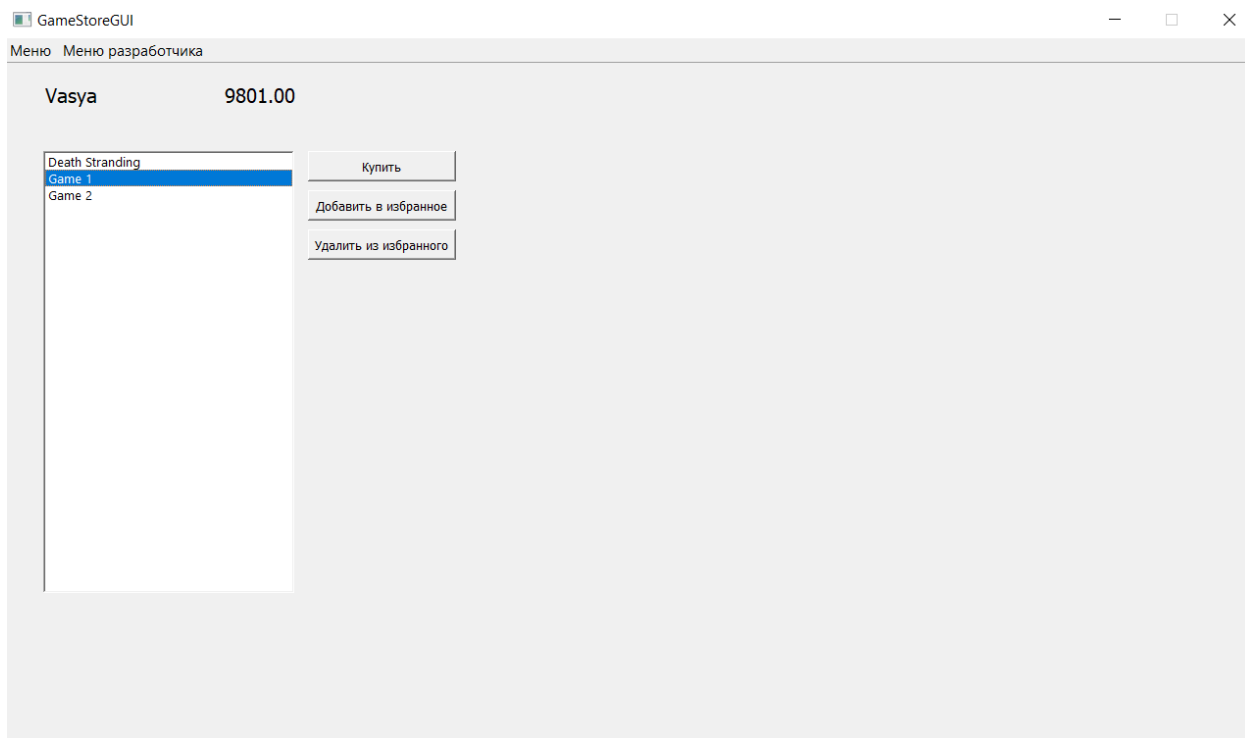


В нем пользователь может с помощью разворачивающихся списков выбрать игру, которую он хочет подарить и пользователя, кому будет передана копия этой игры. Если же игр у пользователя или других пользователей нет, то программа откроет соответствующее окно ошибки, уведомляющее пользователя о невозможности операции.

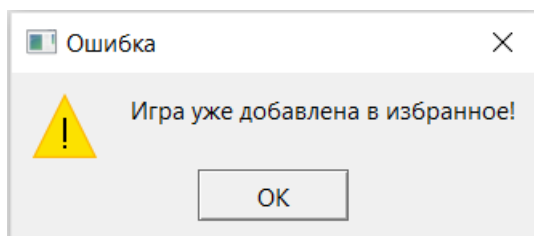
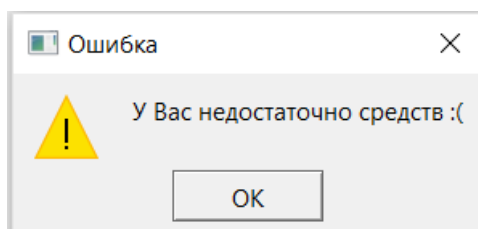
При нажатии кнопки *Пополнить баланс*, открывается окно, в котором пользователь может ввести необходимую сумму пополнения.



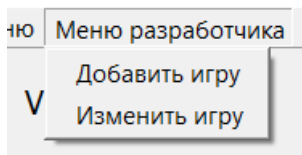
Если в главном окне выбрать игру в любом из списков(игры, избранное, магазин), то появятся дополнительные кнопки, позволяющие взаимодействовать с игрой определенным образом.



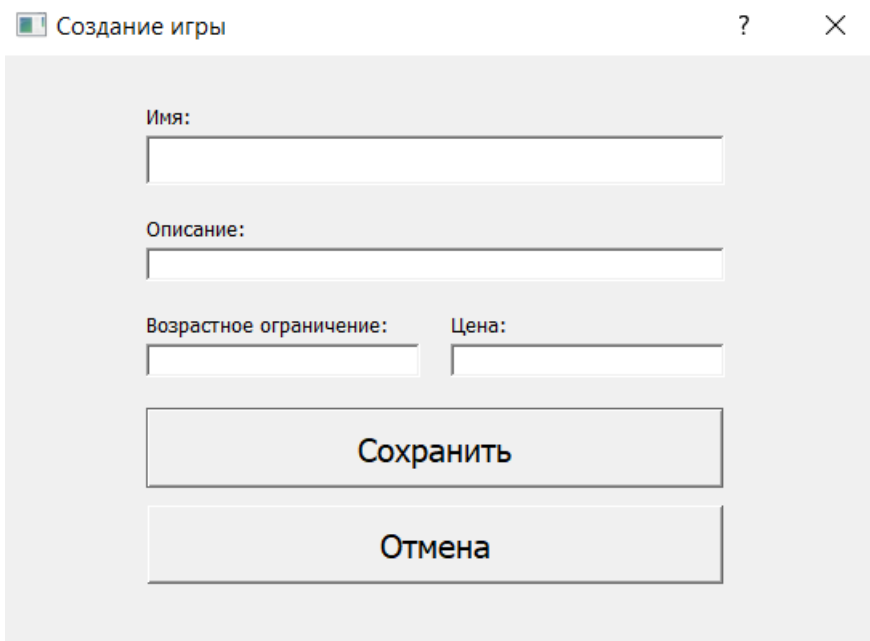
Приложение уведомит вас, если действие выполнить невозможно



При открытии меню разработчика пользователю отрываются возможности добавления и изменения игр.



При нажатии кнопки *добавить игру* отрывается соответствующее окно, позволяющее взаимодействовать с библиотекой игр.



Создание игры

Имя:

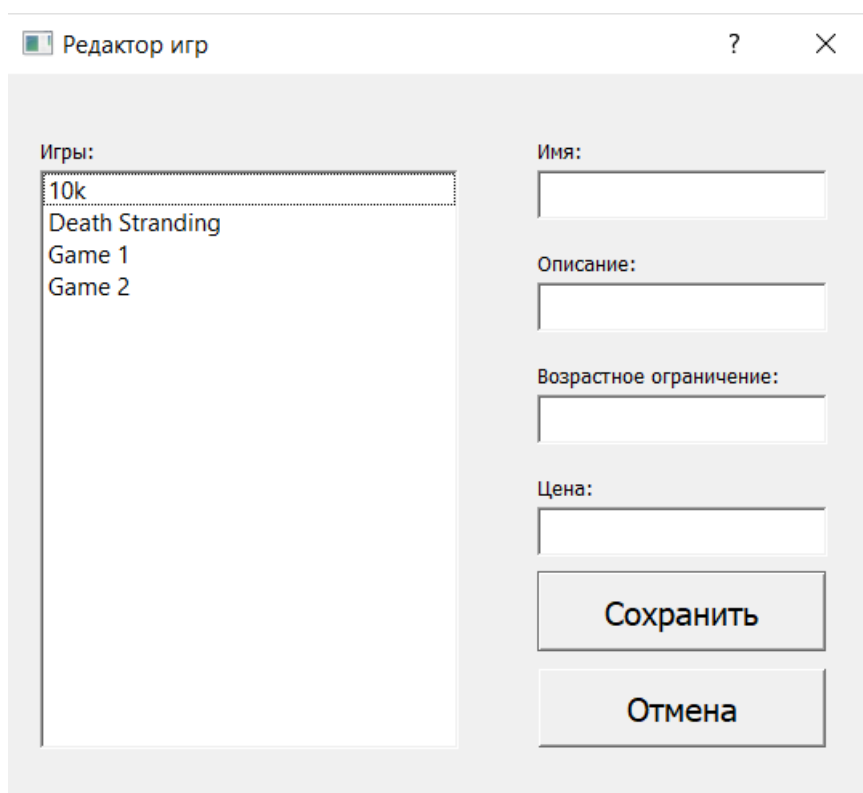
Описание:

Возрастное ограничение: Цена:

Сохранить

Отмена

При нажатии кнопки *изменить игру* открывается следующее окно



Редактор игр

Игры:

- 10k
- Death Stranding
- Game 1
- Game 2

Имя:

Описание:

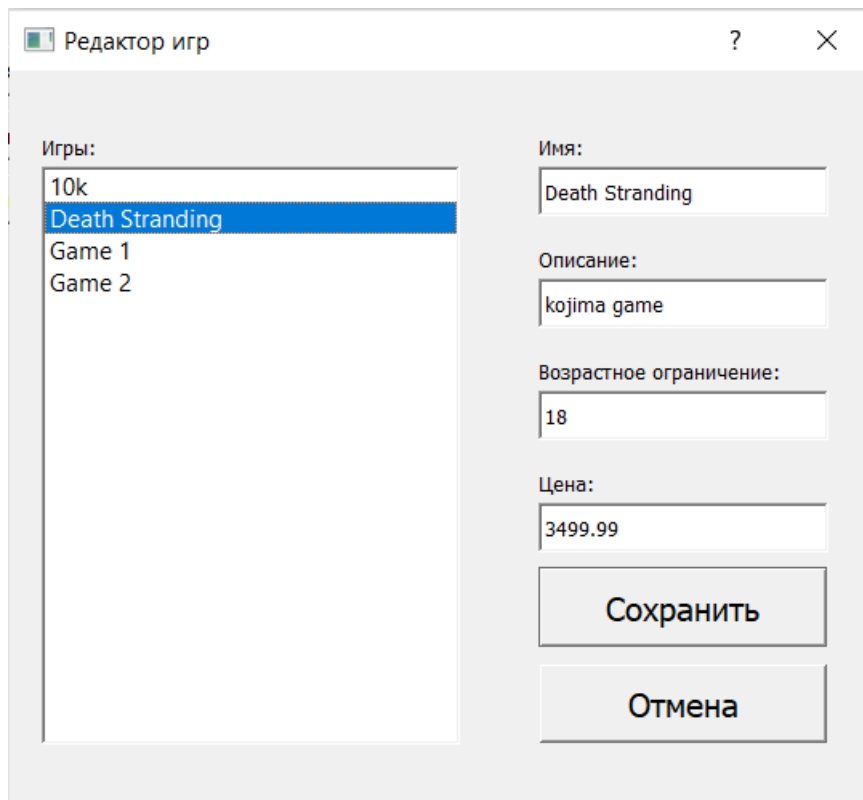
Возрастное ограничение:

Цена:

Сохранить

Отмена

устроенное следующим образом: при нажатии на игру в списке, поля редактирования справа заполняются соответствующими данными об игре.



Затем пользователь может отредактировать данные игры и сохранить их.

## База данных

Базу данных я решил расположить в двух текстовых файлах

GamesData.txt	3/29/2021 10:59 A...	Text Document	1 KB
usersData.txt	3/29/2021 10:58 A...	Text Document	1 KB

- *GamersData.txt* содержит информацию об играх:
  - Имя
  - Описание
  - Возрастное ограничение
  - Цену
- *usersData.txt* содержит информацию об пользователях:
  - Имя (он же логин)
  - Пароль (в зашифрованном виде)
  - Возраст
  - Баланс
  - Купленные игры

- Игры, добавленные в избранное

Данные записываются построчно:

```
GamesData.txt - Notepad
File Edit Format View Help
Death Stranding
kojima game
18
3499.99
Game 1
qwert
14
199
Game 2
qwert
12
250
10k
qweq
12
10000
|
```

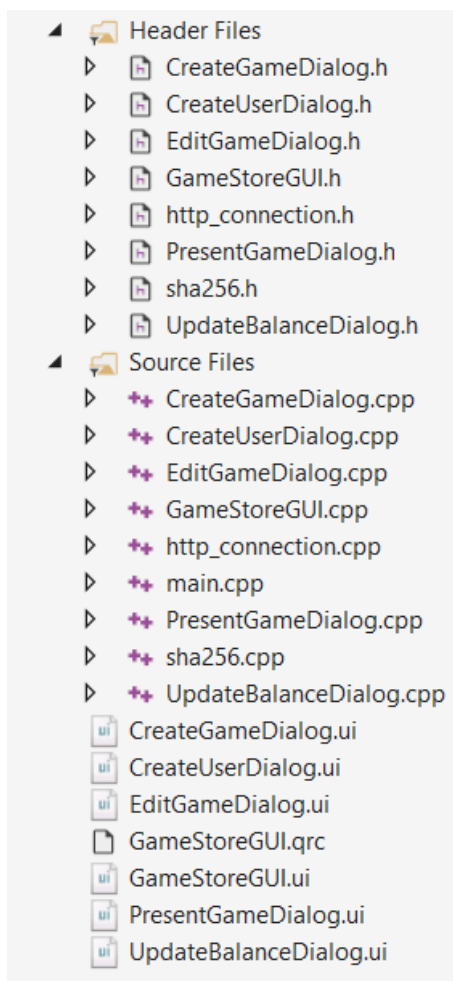
В *usersData.txt* данные записываются так же. Причем купленные игры и игры, добавленные в избранное в строке, разделяются символом '|'.  
 |

```
usersData.txt - Notepad
File Edit Format View Help
Vasya
5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5
18
9801.00
Game 1|Game 2|
Game 1|
Vasiliy
ef797c8118f02dfb649607dd5d3f8c7623048c9c063d532cc95c5ed7a898a64f
18
0
|
```

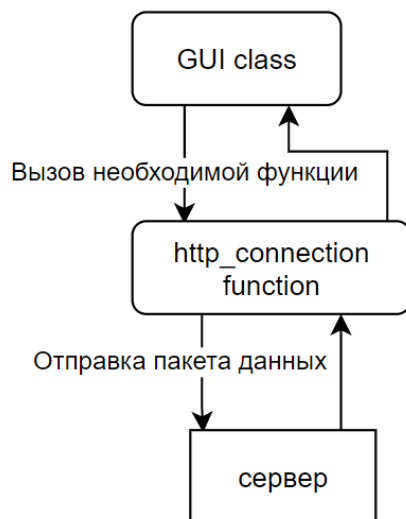
База данных хранится на сервере Microsoft Azure. Работа с базой данных осуществляется специальной программой *GameStoreServerSide.exe*, расположенной на сервере и написанной на C++. Эта же программа осуществляет обмен данными с магазином игр по сети интернет.

## GameStoreGUI.exe

Список файлов:



Блок-схема работы программы:



Все файлы программы можно разделить на 3 типа:

- Dialog файлы. Это файлы описывающие окна программы (GameStoreGUI файлы относятся к этой же категории, просто описывают главное окно). Такие файлы можно в свою очередь разделить на 2 типа:
  - .ui файлы. Эти файлы описывают интерфейс окон и создаются с помощью визуальной среды Qt Designer.
  - .cpp и .h файлы. Эти файлы отвечают за работу окон. В этих файлах описываются действия кнопок, вызываются необходимые функции для загрузки с сервера информации и многое другое.
- http\_connection файлы. Эти два файла содержат одноименный класс, который осуществляет «общение» с сервером: отправляет на него необходимые запросы, получает данные.
- sha256 файлы. Эти два файла содержат одноименный класс, используемый в шифровании паролей.
- main файл. Содержит main функцию, которая запускает основное окно.

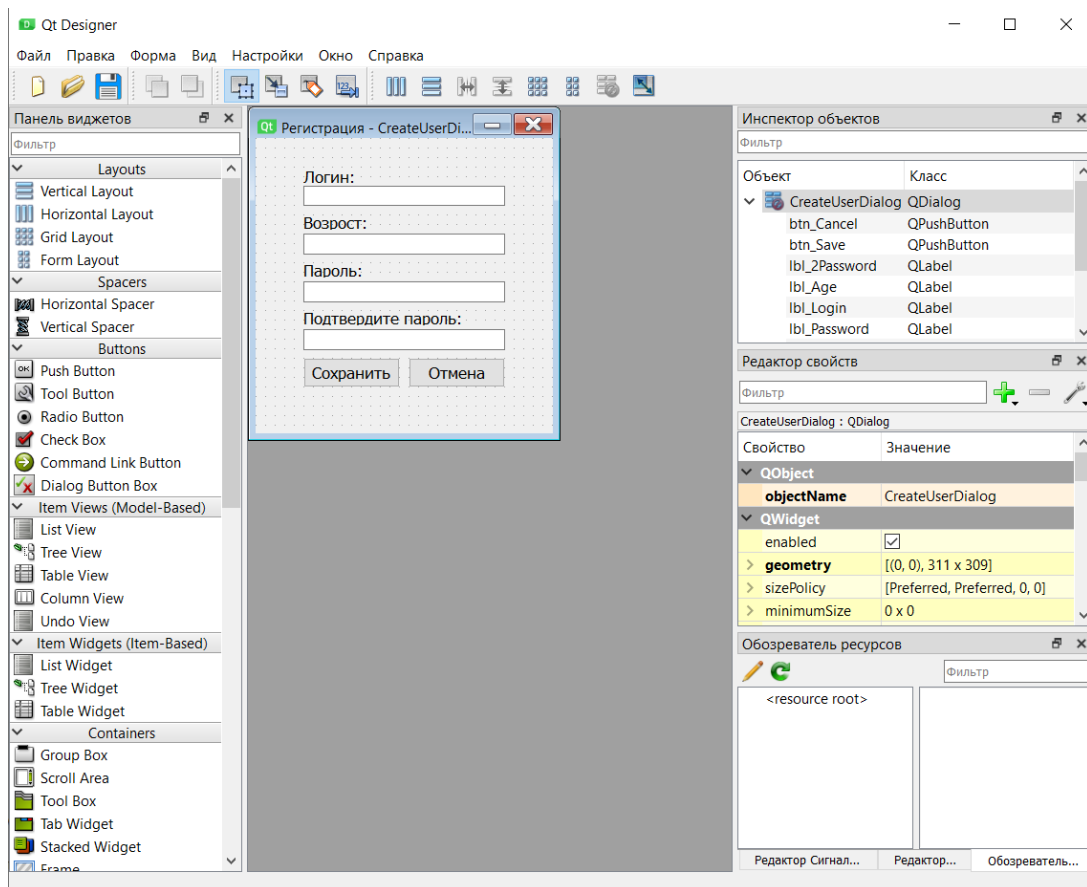
Содержимое файла *main.cpp*:

```

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

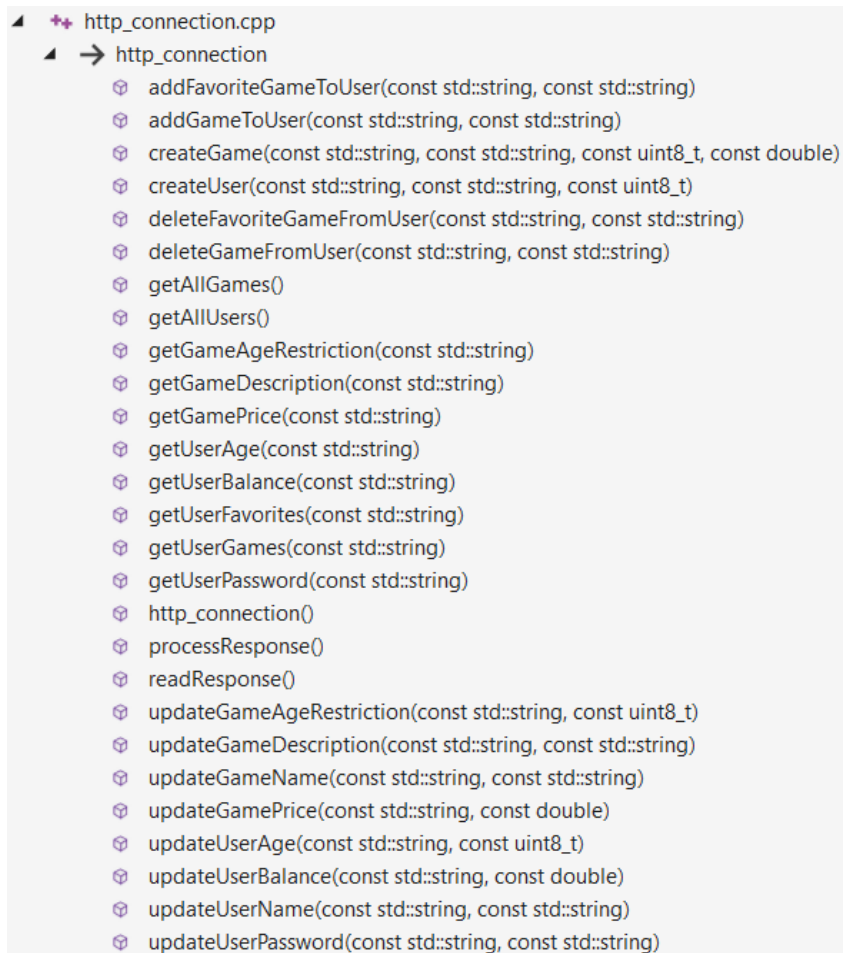
```

## Редактирование окна в Qt Designer:



Класс `http_connection` содержит множество функций





которые выполняют действия, соответствующие их названию.

Давайте рассмотрим, как устроен *GameStoreGUI.cpp*.

Начнем с конструктора:

```

GameStoreGUI::GameStoreGUI(QWidget *parent)
: QMainWindow(parent)
{
    setlocale(LC_ALL, "Russian");

    ui.setupUi(this);
    ui.menuBar->setVisible(false);
    ui.lbl_UserName->setVisible(false);
    ui.lbl_Balance->setVisible(false);
    ui.lst_Games->setVisible(false);
    ui.btn_Buy->setVisible(false);
    ui.btnAddToFavorites->setVisible(false);
    ui.btnDeleteFromFavorites->setVisible(false);
    this->setFixedSize(1280, 720);

    user = "";

    QList<QAction*> menus = ui.menuBar->actions();
    for (QAction* elem : menus[0]->menu()->actions()) {
        wstring a = elem->text().toStdWString();
        if (elem->text() == L"Избранное")
            QObject::connect(elem, &QAction::triggered, this, &GameStoreGUI::click_show_favorite);
        else if (elem->text() == L"Игры")
            QObject::connect(elem, &QAction::triggered, this, &GameStoreGUI::click_show_games);
        else if (elem->text() == L"Магазин")
            QObject::connect(elem, &QAction::triggered, this, &GameStoreGUI::click_show_store);
        else if (elem->text() == L"Подарить игру")
            QObject::connect(elem, &QAction::triggered, this, &GameStoreGUI::click_Present);
        else if (elem->text() == L"Пополнить баланс")
            QObject::connect(elem, &QAction::triggered, this, &GameStoreGUI::click_add_money);
    }
    for (QAction* elem : menus[1]->menu()->actions()) {
        wstring a = elem->text().toStdWString();
        if (elem->text() == L"Изменить игру")
            QObject::connect(elem, &QAction::triggered, this, &GameStoreGUI::click_update_game);
        else if (elem->text() == L"Добавить игру")
            QObject::connect(elem, &QAction::triggered, this, &GameStoreGUI::click_create_game);
    }

    QObject::connect(ui.btn_Login, &QPushButton::clicked, this, &GameStoreGUI::login);
    QObject::connect(ui.btn_Registration, &QPushButton::clicked, this, &GameStoreGUI::click_create_user);

    QObject::connect(ui.lst_Games, &QListWidget::itemClicked, this, &GameStoreGUI::game_selected);
    QObject::connect(ui.btn_Buy, &QPushButton::clicked, this, &GameStoreGUI::click_buy);
    QObject::connect(ui.btnAddToFavorites, &QPushButton::clicked, this, &GameStoreGUI::click_add_to_favorites);
    QObject::connect(ui.btnDeleteFromFavorites, &QPushButton::clicked, this, &GameStoreGUI::click_delete_from_favorites);
}

```

1. По умолчанию все объекты в окне видимы, поэтому некоторые объекты в начале необходимо скрыть.
2. Здесь мы к кнопкам в меню привязываем определенное действие и теперь при нажатии на кнопку будет вызываться определенная функция.
3. То же самое, но уже для кнопок в самом окне.

Далее идут функции, объявленные в *GamestoreGUI.h*

```

void login();
void click_show_favorite();
void click_show_games();
void click_show_store();

void click_create_user();
void click_create_game();
void click_update_game();

void game_selected(QListWidgetItem* item);
void click_open();
void click_buy();
void click_add_to_favorites();
void click_delete_from_favorites();
void click_Present();
void click_add_money();

```

Давайте рассмотрим функцию login, она вызывается если пользователь нажмет кнопку Войти.

```

void GameStoreGUI::login() {
    SHA256* cipher = new SHA256();
    std::string encrypted_password = cipher->hash(ui.le_Password->text().toStdString());
    delete cipher;
    if (conn.getUserPassword(ui.le_UserName->text().toStdString()) == encrypted_password) {
        ui.le_Password->setVisible(false);
        ui.le_UserName->setVisible(false);
        ui.btn_Login->setVisible(false);
        ui.btn_Registration->setVisible(false);

        ui.menuBar->setVisible(true);
        ui.lbl_UserName->setVisible(true);
        ui.lbl_Balance->setVisible(true);
        ui.lst_Games->setVisible(true);

        ui.lbl_UserName->setText(ui.le_UserName->text());
        user = ui.le_UserName->text().toStdString();
        ui.lbl_Balance->setText(QString(conn.getUserBalance(ui.le_UserName->text().toStdString()).c_str()));

        for (const std::string &game : conn.getUserGames(ui.le_UserName->text().toStdString())) {
            new QListWidgetItem(tr(game.c_str()), ui.lst_Games);
        }
    }
}

```

1. Шифруем пароль, введенный пользователем, с помощью класса SHA256.
2. Сравниваем зашифрованный пароль и зашифрованный пароль, хранящийся в базе данных.
3. Если пароли совпали, то можно изменить интерфейс окна, сделав невидимыми ненужное...
4. ... и сделав видимым нужное

5. Заполняем поля имени(логина) и баланса.
6. Заполняем список игр купленными играми.

А теперь давайте рассмотрим функцию `click_create_user`, она вызывается если пользователь нажмет кнопку Регистрация.

```
void GameStoreGUI::click_create_user() {
    CreateUserDialog win(this, &conn);
    if (win.exec() == QDialog::Accepted) {}
}
```

Тут все просто:

1. Объявляем объект класса.
2. Открываем диалоговое окно.

А теперь давайте рассмотрим функцию `getUserBalance` в классе `http_connection`.

```
std::string http_connection::getUserBalance(const std::string name)
{
    data.clear(); // 1

    QTimer tim;
    tim.setInterval(10000);
    tim.setSingleShot(true); // 2

    QNetworkAccessManager http_client;
    QByteArray req_data = "getUserBalance\r\n1\r\n" + QByteArray(name.c_str());
    QNetworkReply* http_response = http_client.post(QNetworkRequest(QUrl("http://" + server_ip_address + ":" + QString::number(port) + "/")), req_data);
    if (http_response == nullptr) {
        QMessageBox(QMessageBox::Warning, QString::fromWCharArray(L"Ошибка"), QString::fromWCharArray(L"Ошибка подключения"), QMessageBox::NoButton).exec();
        return data;
    } // 3

    QEventLoop loop; // 4
    connect(&tim, &QTimer::timeout, &loop, &QEventLoop::quit);
    connect(http_response, &QNetworkReply::finished, &loop, &QEventLoop::quit); // 5

    tim.start(); // 6
    loop.exec(); // 7

    if (tim.isActive()) {
        tim.stop();
        if (http_response->error() != QNetworkReply::NoError) {
            QMessageBox(QMessageBox::Warning, QString::fromWCharArray(L"Ошибка"), http_response->errorString(), QMessageBox::NoButton).exec();

            http_response->deleteLater();
            return data;
        }
        int http_status = http_response->attribute(QNetworkRequest::HttpStatusCodeAttribute).toInt();
        if (http_status == 200)
            data = http_response->readAll();

        http_response->deleteLater();
        return data;
    }

    disconnect(http_response, &QNetworkReply::finished, &loop, &QEventLoop::quit);
    http_response->abort();

    http_response->deleteLater();

    return data;
}
```

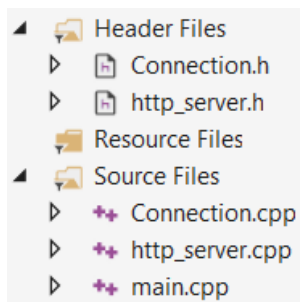
1. Очищаем data. data, как и server\_ip\_address, и как port создана в конструкторе http\_connection:

```
http_connection::http_connection() {  
    server_ip_address = "13.69.21.71";  
    port = 80;  
    data = "";  
}
```

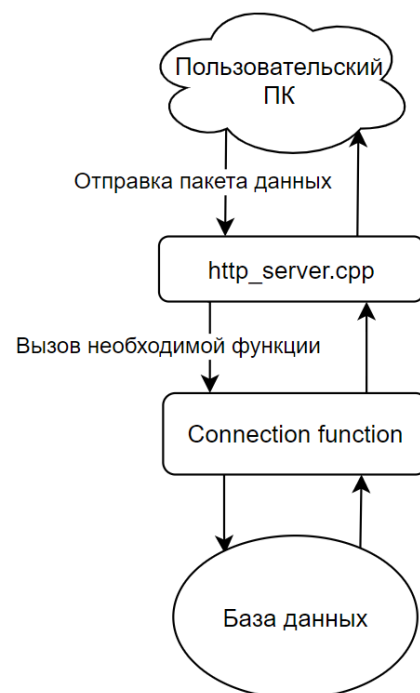
2. Создаем таймер. Он нам понадобится, когда мы будем принимать ответный пакет данных.
3. Отправляем на сервер три строки:
  - 1) Команда, которую мы выполняем (getUserBalance)
  - 2) Число 1 (означает количество дополнительных строк)
  - 3) Имя пользователя, чей баланс мы хотим получить.
4. QEventLoop предотвращает возможные ошибки при параллельном обращении к серверу.
5. Принимаем ответный пакет данных.
6. Запускаем таймер. Пока он работает программа ждет пакет данных от сервера.
7. Если таймер работает, то возвращаем принятый пакет данных

## GameStoreServerSide.exe

Список файлов:



Блок-схема работы программы:



Все файлы программы можно разделить на 3 типа:

- `http_server` файлы. Эти два файла содержат одноименный класс, который в реальном времени слушает порт. При получении входящего запроса обрабатывает его и вызывает необходимую функцию в классе `Connection` для работы с базой данных. Отправляет ответный пакет данных на ПК пользователя.
- `Connection` файлы. Эти два файла содержат одноименный класс, который содержит функции для работы с базой данных.
- `main` файл. Содержит одноименную функцию, которая создает объект класса `http_server` и запускает прослушивание порта.

Содержимое файла *main.cpp*:

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    Connection::init();

    http_server *serv = new http_server(&a);

    if (!serv->isListening()) {
        std::cout << "Server is not listening" << std::endl;
        return 1;
    }
    else
        std::cout << "Server is listening" << std::endl;

    return a.exec();
}
```

## Заключение

В результате программа работает и делает все, что от нее ожидалось, в будущем можно будет улучшить программу добавив дополнительный функционал.

Работа для меня была сложной так как это первый опыт разработки крупного проекта. При его разработке я изучил фреймворк Qt, освоил сетевое взаимодействие программ и закрепил знания программирования на C++ на практике.

На данном этапе программа работает исправно, но не в состоянии выполнять полноценные функции электронного магазина так как в нем отсутствуют такие возможности как загрузка файла игры на сервер, загрузка файла игры с сервера на ПК пользователя и запуск игры. В дальнейшем я планирую, после наработки дополнительных теоретических и практических знаний, вернуться к этому и реализовать в нем вышеперечисленные функции.