|  |  |
| --- | --- |
|  | **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 \_\_\_\_\_\_\_\_\_«Информатика и системы управления» (ИУ)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_«Информационная безопасность» (ИУ8)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

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

Студент ИУ8-34 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. М. Федоров**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Колесников**

(Подпись, дата) (И.О.Фамилия)

*2021 г.*

**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)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ 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. Предусмотрена возможность 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 2021 г.

**Руководитель курсового проекта**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **А. В. Колесников**

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. М. Федоров**

(Подпись, дата) (И.О.Фамилия)

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

Оглавление

**Вступление4**

**Описание работы программы5**

**База данных12**

**GameStoreGUI.exe15**

**GameStoreServerSide.exe21**

**Заключение23**

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 С++
* Освоить сетевое взаимодействие программ, работу с серверами.
* Научится работать с фреймворком Qt

Используемые материалы в данной работе:

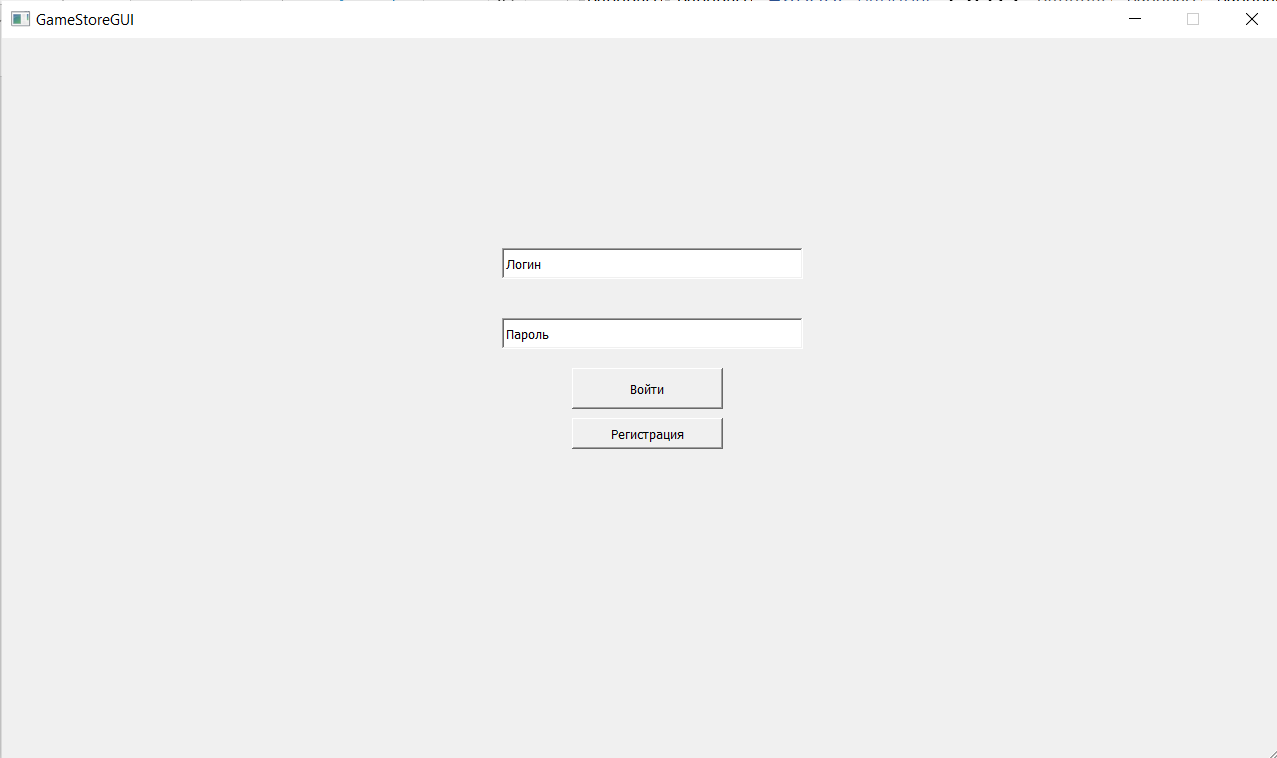
* С++ - распространенный язык программирования
* Qt – фреймворк для разработки ПО на языке программирования С++
* Microsoft Azure – облачная платформа, предоставляющая сервер для работы магазина игр
* sha256 – безопасный алгоритм хеширования

Причины использования данных материалов:

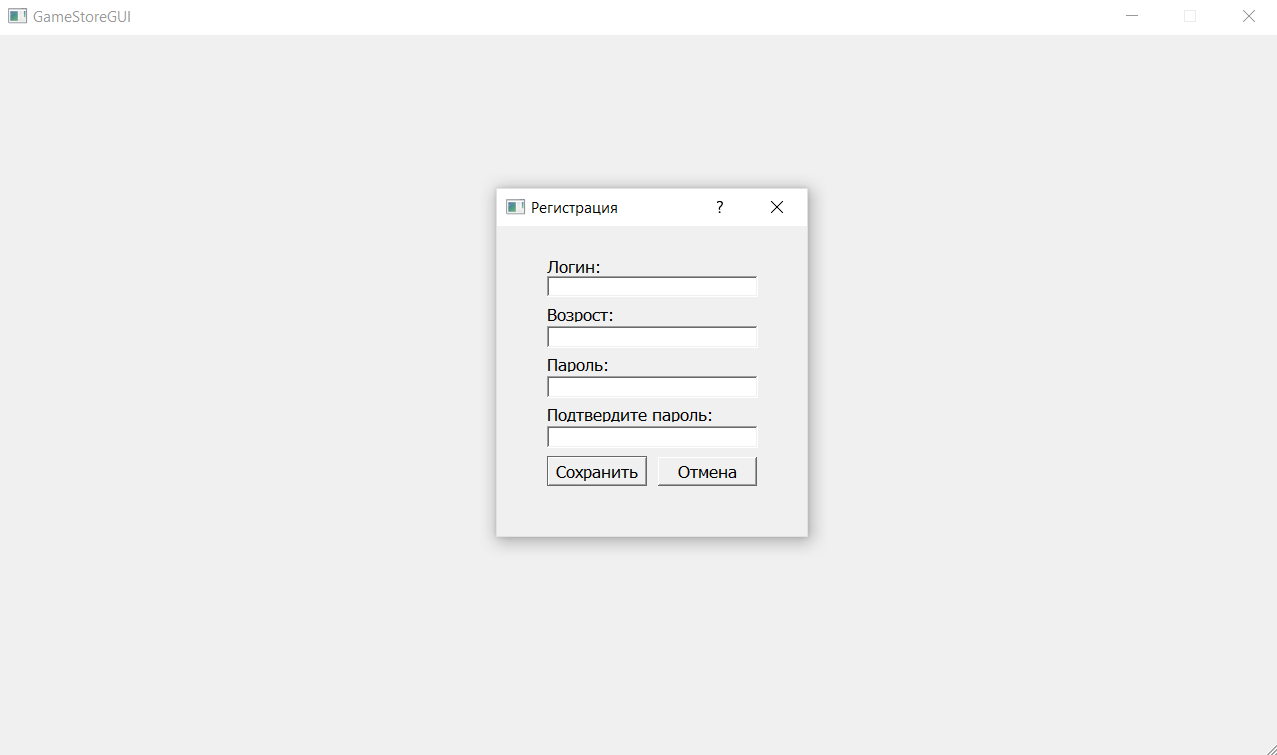
* С++ - язык, на котором я учусь работать уже продолжительное время и хочу лучше изучить его.
* Qt – качественный и распространенный фреймворк. Комплектуется визуальной средой разработки графического интерфейса.
* Microsoft Azure – позволяет удобно использовать сервер для расположения базы данных на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



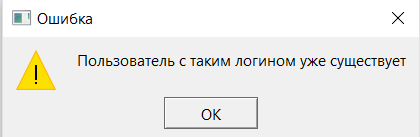
Давайте нажмем кнопку *регистрация*. Открылось новое диалоговое окно регистрации, при этом взаимодействие с первым оком временно заблокировано.



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:

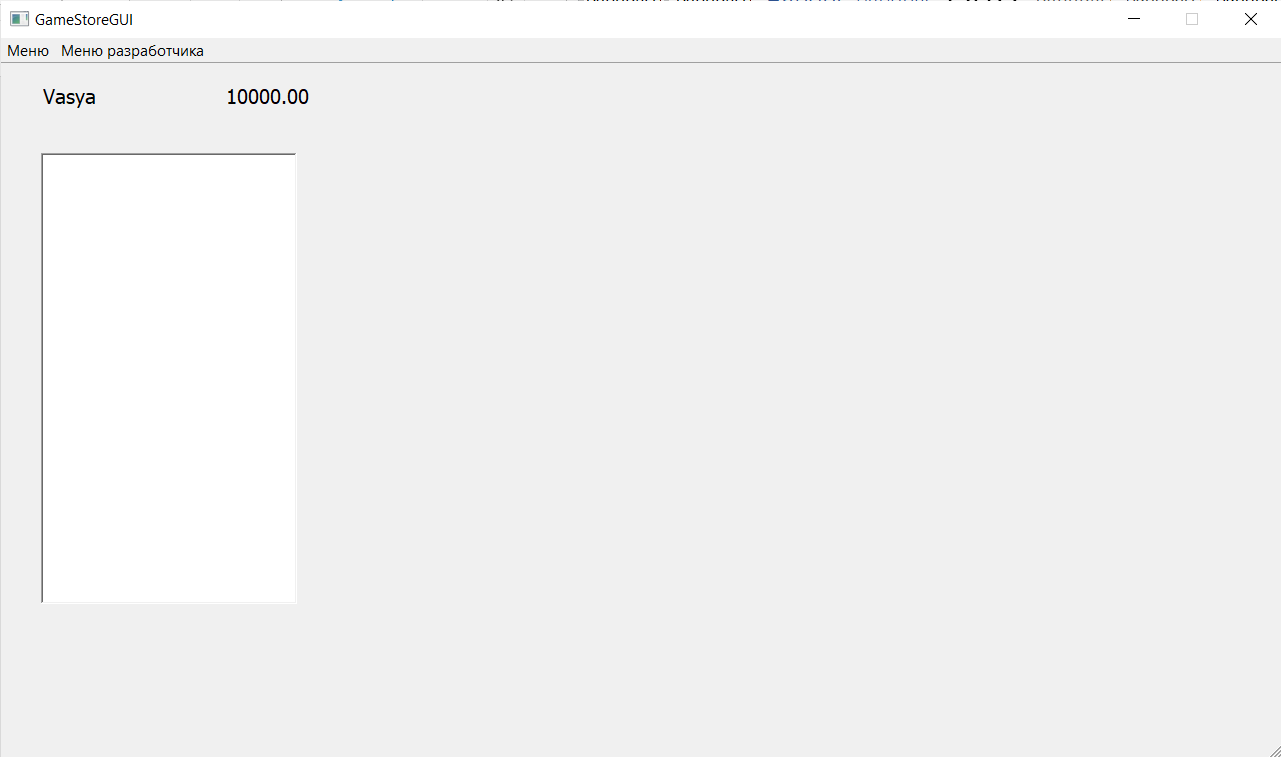
* Логин повторяется с логином уже существующего пользователя
* Возраст имеет значения отличные от целых чисел в промежутке от 3-х до 99-ти
* Пароль имеет длину менее 8-ми символов
* Пароли не совпадают



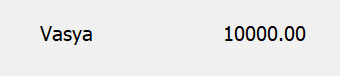
После закрытия окна ошибки пользователь сможет исправить введенные данные и создать пользователя.

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

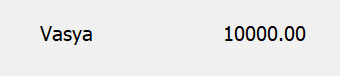
Войдя в свою учетную запись, пользователь видит следующее диалоговое окно.



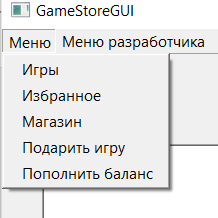
В левом верхнем углу отображается логин пользователя



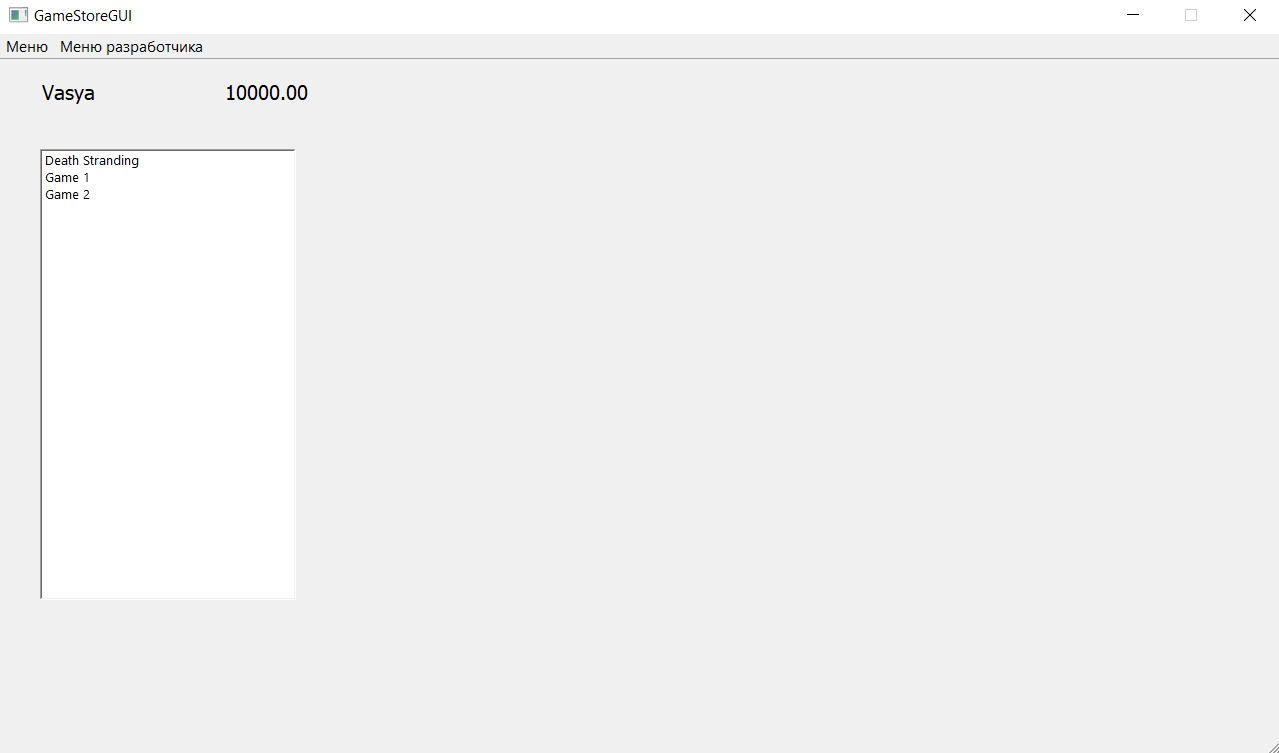
и баланс



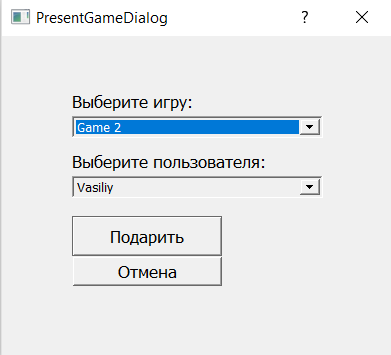
При нажатии на кнопку *меню* появляются доступные команды.



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

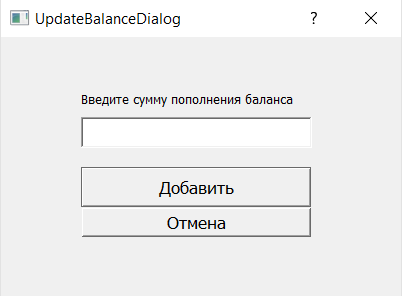


Если пользователь нажмет кнопку *Подарить игру*, то откроется диалоговое окно.

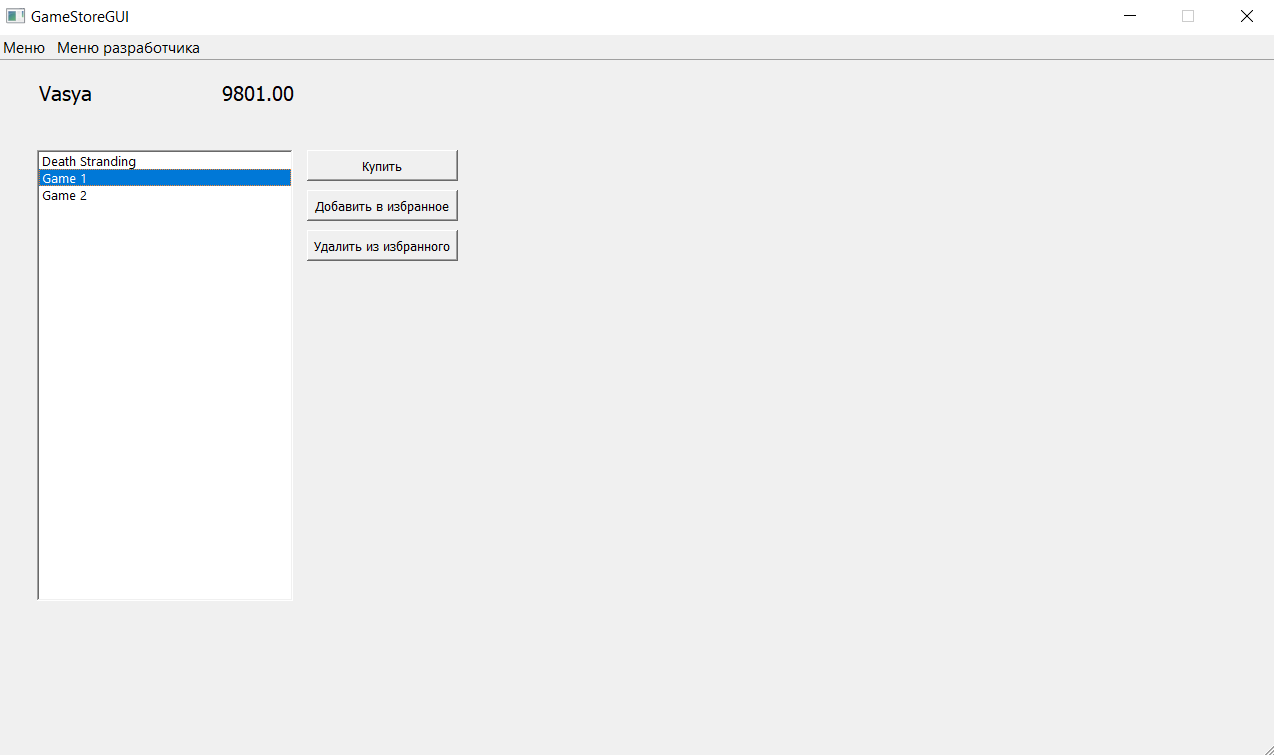


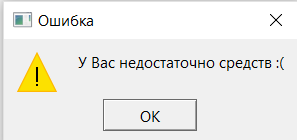
В нем пользователь может с помощью развертывающихся списков выбрать игру, которую он хочет подарить и пользователя, кому будет передана копия этой игры. Если же игр у пользователя или других пользователей нет, то программа откроет соответствующее окно ошибки, уведомляющее пользователя о невозможности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

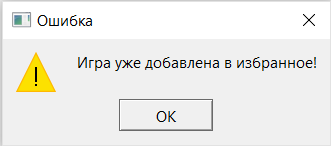
При нажатии кнопки *Пополнить баланс*, открывается окно, в котором пользователь может ввести необходимую сумму пополнения.



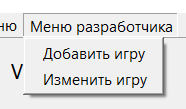
Если в главном окне выбрать игру в любом из списков(игры, избранное, магазин), то появятся дополнительные кнопки, позволяющие взаимодействовать с игрой определенным образом.

Приложение уведомит вас, если действие выполнить невозможно

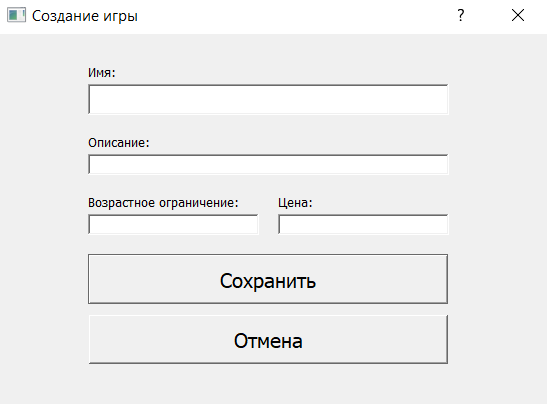




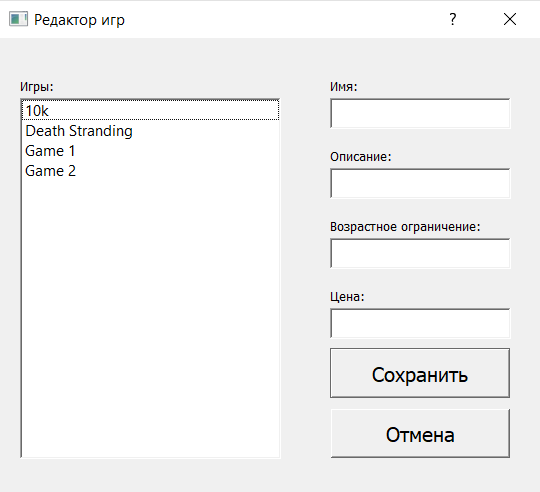
При открытии меню разработчика пользователю отрываются возможности добавления и изменения игр.



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

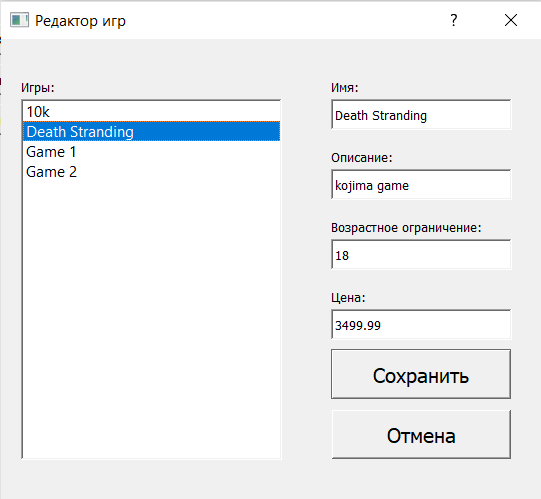


При нажатии кнопки *изменить игру* открывается следующее окно



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.

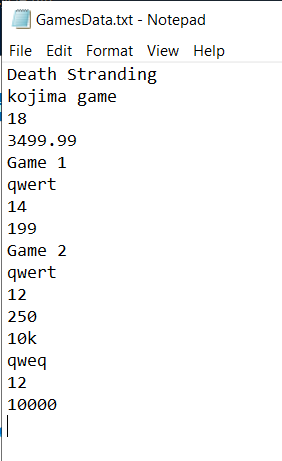
База данных

Базу данных я решил расположить в двух текстовых файлах

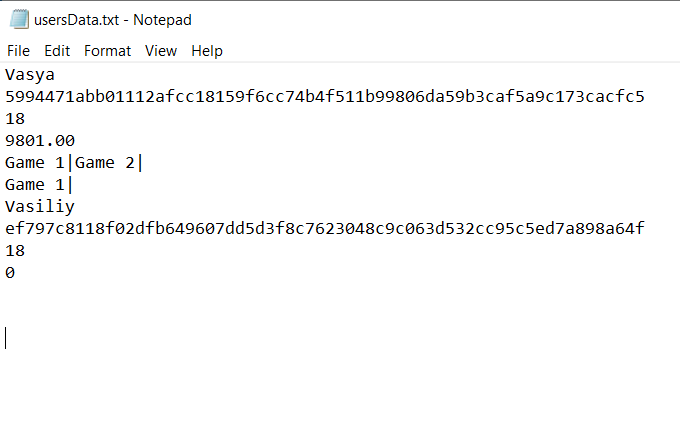


* *GamersData.txt* содержит информацию об играх:
* Имя
* Описание
* Возрастное ограничение
* Цену
* *usersData.txt* содержит информацию об пользователях:
* Имя (он же логин)
* Пароль (в зашифрованном виде)
* Возраст
* Баланс
* Купленные игры
* Игры, добавленные в избранное

Данные записываются построчно:



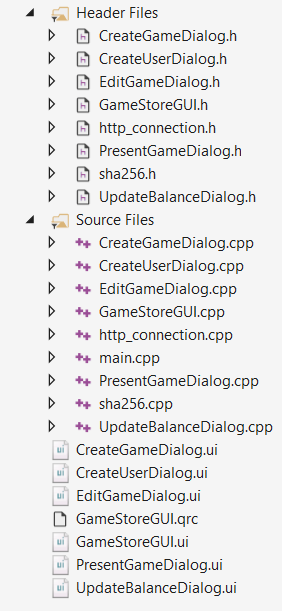
В *usersData.txt* данные записываются так же. Причем купленные игры и игры, добавленные в избранное в строке, разделяются символом ‘|’.



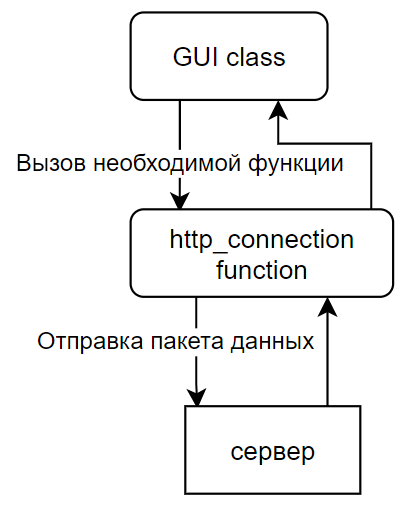
База данных хранится на сервере Microsoft Azure. Работа с базой данных осуществляется специальной программой *GameStoreServerSide.exe*, расположенной на сервере и написанной мной на С++. Эта же программа осуществляет обмен данными с магазином игр по сети интернет.

GameStoreGUI.exe

Список файлов:



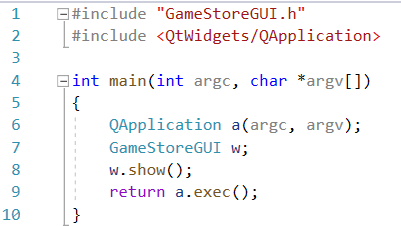
Блок-схема работы программы:



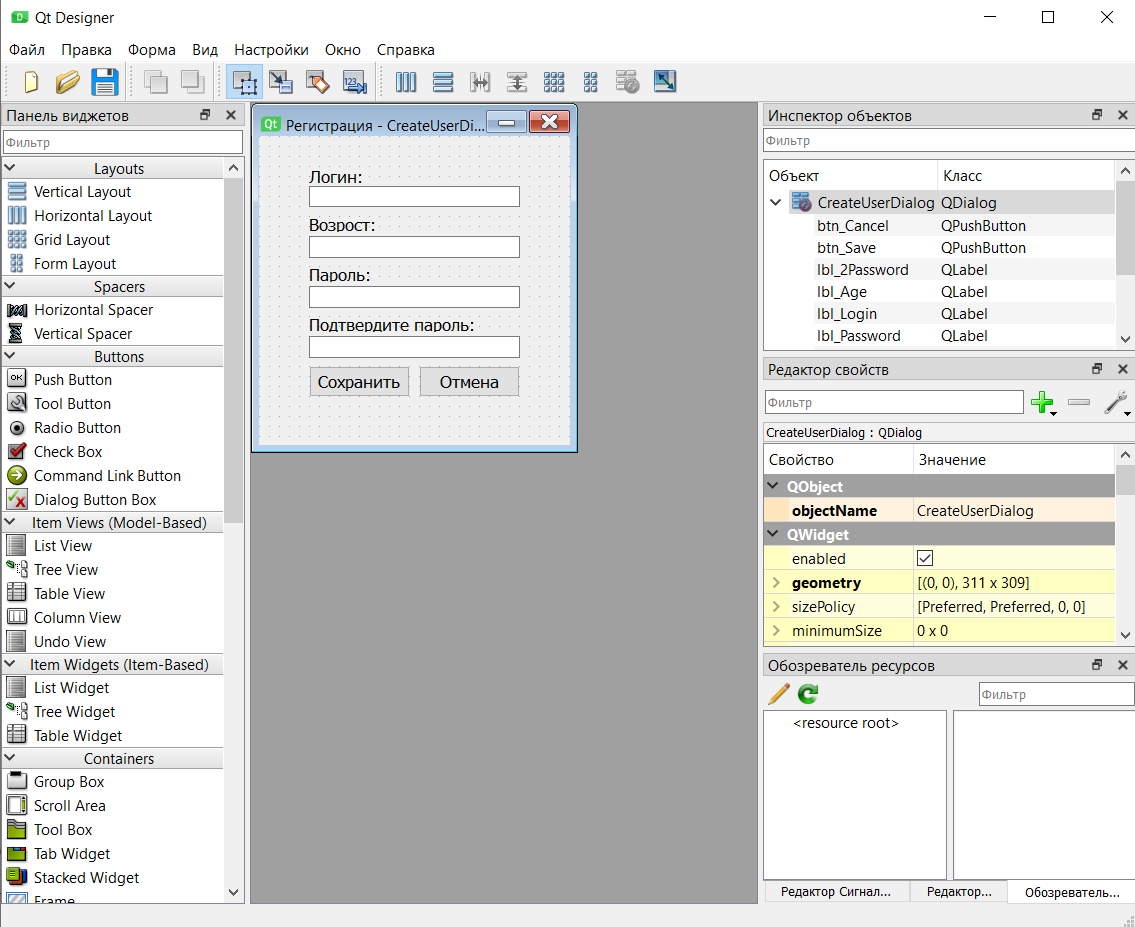
Все файлы программы можно разделить на 3 типа:

* 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 подгрузки 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.

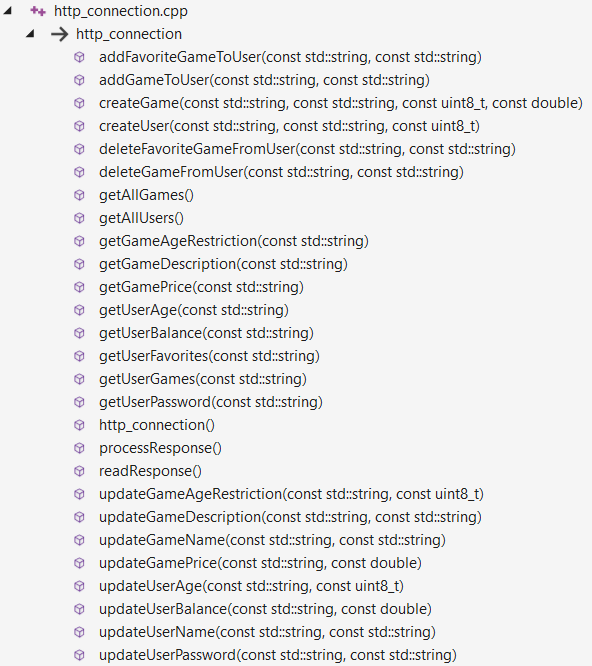
Содержимое файла *main.cpp*:



Редактирование окна в Qt Designer:



Класс http\_connection содержит множество функций



которые выполняют действия, соответствующие их названию.

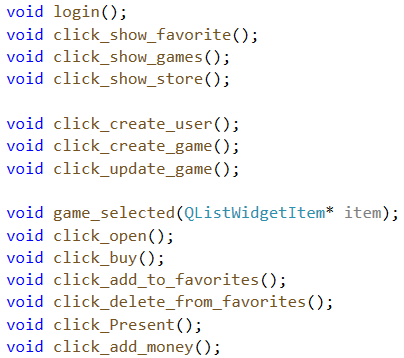
Давайте рассмотрим, как устроен *GameStoreGUI.cpp*.

Начнем с конструктора:



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*

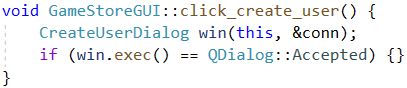


Давайте рассмотрим функцию login, она вызывается если пользователь нажмет кнопку Войти.



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.

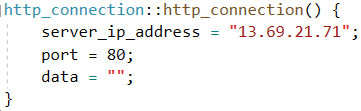


Тут все просто:

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

А теперь давайте рассмотрим функцию getUserBalance в классе http\_connection.

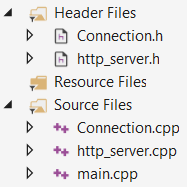
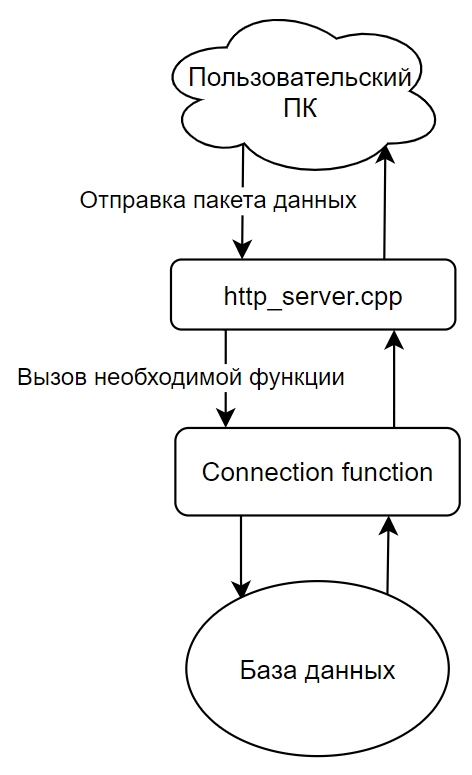
1. Отчищаем data. data, как и server\_ip\_address, и как port создана в конструкторе http\_connection:



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

GameStoreServerSide.exe

Список файлов:

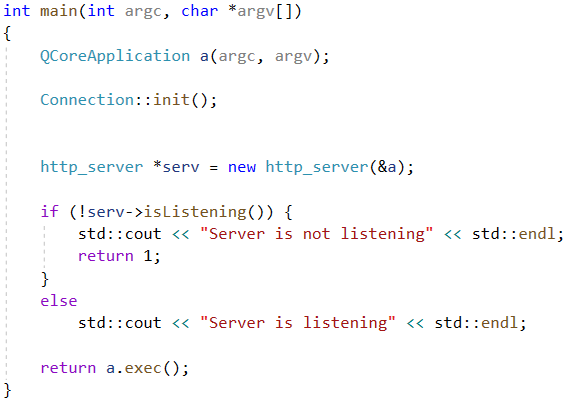


Блок-схема работы программы:

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*:



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 С++ 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, вернутся к этому и реализовать в нем вышеперечисленные функции.