|  |
| --- |
|  |

**Введение**

Клиент – серверное приложение «Двадцать одно» является программным средством для игры в «Двадцать одно» карты.

В современном мире люди стали больше времени уделять развлечениям особенно в интернете. Старая добрая игра «Двадцать одно» будет перенесена в виртуальный мир. Люди смогут играть с другими пользователями по всему миру. Больше людям не надо хранить карты и заботиться чтобы они не терялись.

Основная задача данного курсового проекта – создать игру для развлечения людей, к которой они имели доступ в любой момент времени.

Технологии, которые были использованы в работе:

1. Объектно-ориентированный язык программирования C#;
2. Среда разработки Visual Studio 2019;
3. Система для построения клиентских приложений WPF;
4. Microsoft SQL Management studio 18
5. Спецификация UML для создания диаграмм и схем.

# Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству

PokerStar – карточная игра в виде покера, доступны турниры, игра как на игровую валюту так и на реальные деньги. В игре реализована механика поиска лобби с свободными местами, заполненные лобби не видны пользователю. Внутренний чат, помощь игрокам с комбинациями, настройка звука, возможность выбрать ставку для игры.



На рисунке 1.1 представлен внешний вид приложения «PokerStar».

Дурак Онлайн – мобильная карточная игра. Доступен чат, стикеры, добавление в друзья. Присутствует настройка профиля: фото, имя в игре. В самой игре можно настроить поиск комнат с подходящими для вас условиями. Создать свои комнаты с паролем или без, выбор скинов на карты, достижения, турниры. Во время игры можно за отдельную плату вернуть карту которую вы положили на стол, подсвечивать карты, которые подходят для отбоя карты с которой на вас походил соперник.



На рисунке 1.2 представлен внешний вид приложения «Дурак Онлайн».

**Вывод:**

В результате изучения аналогов были выделены основные функции, которые должны быть в нашем приложении, общаться в чате, создавать комнаты, пополнять баланс, просматривать результаты игр, поиск комнат из списка свободных.

# Анализ требований к программному средству и разработка функциональных требований

## **Описание инструментов разработки программного средства**

При разработке приложения были использованы:

* Интегрированная среда разработки Microsoft Visual Studio 2019;
* Программная платформа .NET Framework 4.7.2;
* Язык программирования C#;
* Расширяемый язык разметки XAML;
* Технология WPF;
* Спецификация UML;
* Microsoft SQL Management studio 18

C# — объектно-ориентированный язык программирования. Разработан в 1998 — 2001 под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework.

XAML — это декларативный язык разметки, разработанный компанией Microsoft. Он используется для инициализации объектов в технологиях на платформе .NET.  Декларативные языки, позволяют описать состояние, которого мы хотим добиться, но не требуют описать, как прийти к нему.

WPF (Windows Presentation Foundation) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

UML — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML— язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

С помощью языка UML построим диаграмму вариантов использования для демонстрации функционала приложения.



Рис. 2.1 – Описание функций приложения

Рассмотрим каждый из прецедентов подробнее.

Прецедент «Регистрация» заключается в создании нового пользователя с указанием его позывного в игре, логина и пароля. Данные учитываются в базе данных.

Прецедент «Авторизация» заключается во входе в приложение. Используются уже ранее записанные данные, а именно логин и пароль.

Прецедент «Поиск лобби» заключается в поиске комнат для игры с другими пользователями.

Прецедент «Создание лобби» заключается в создании собственной комнаты для игры.

Прецедент «Чат» заключается в коммуникации пользователей в приложении.

Прецедент «Пополнение баланса» заключается в пополнении баланса реальными деньгами с помощью карты.

## **Проектирование логической структуры базы данных**

Для реализации поставленной задачи была создана база данных, для этого в Visual Studio необходимо было установить соединение с SQL Server.

Прежде всего, необходимо спроектировать корректную базу данных для приложения. В данном случае, база данных содержит следующие таблицы:

* Информация о пользователях;
* Информация о лобби;
* Информация о пополнении баланса;

Диаграмма базы данных представлена на рисунке 3.1

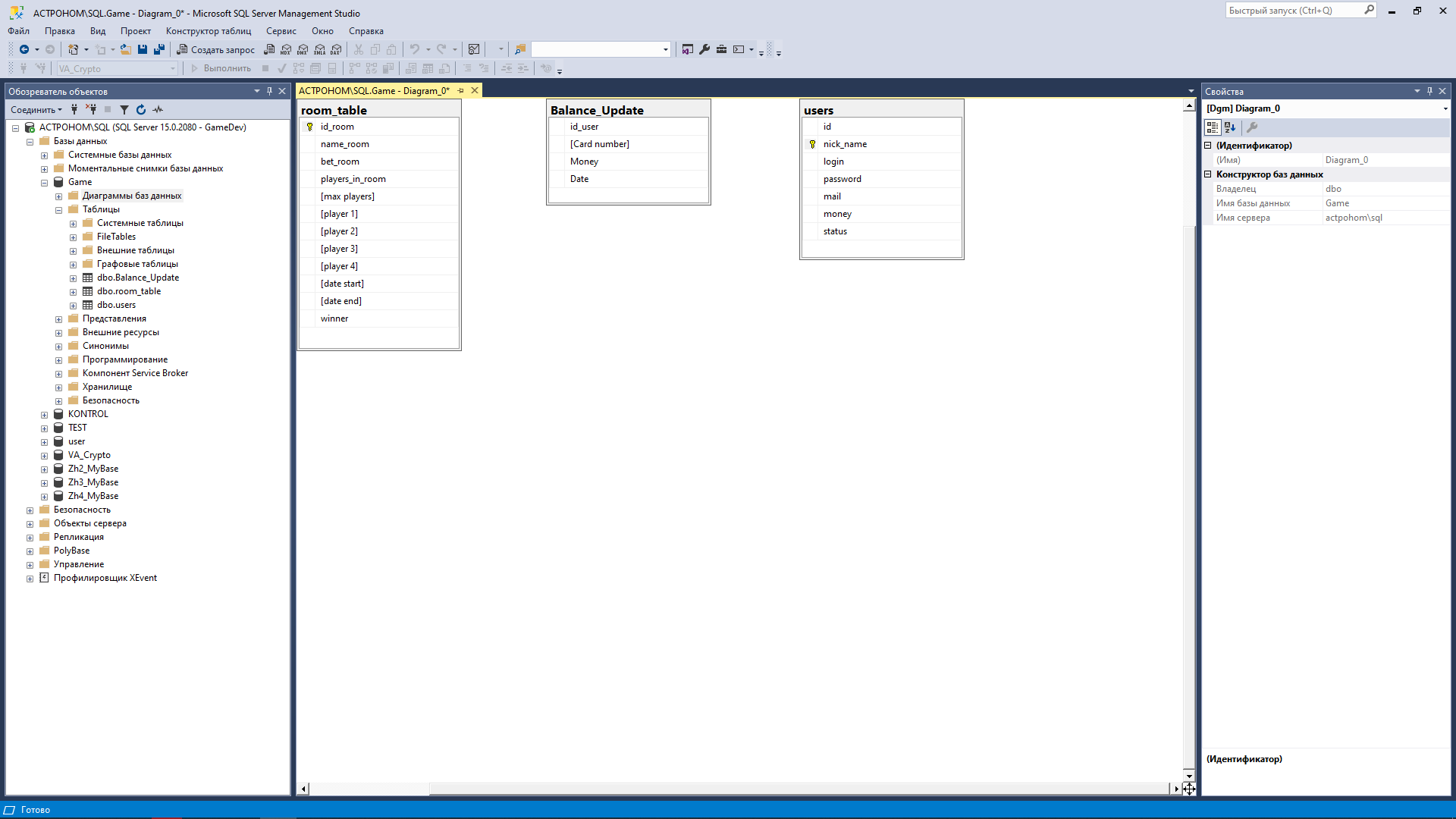


Рис. 2.1 – Диаграмма базы данных

Далее рассмотрим подробнее структуру каждой таблицы базы данных.

Таблица users хранит информацию о пользователях приложения. Данная таблица содержит поля:

* Id – уникальный идентификатор пользователя
* Nick\_name – имя пользователя
* Login – логин пользователя
* Password – значение хеш-функции для пароля пользователя
* Mail – почта привязанная к аккаунту
* Money – количество игровых денег на аккаунте
* Status – показатель онлайн пользователь или нет

Структура таблицы users представлена на рисунке 3.2.

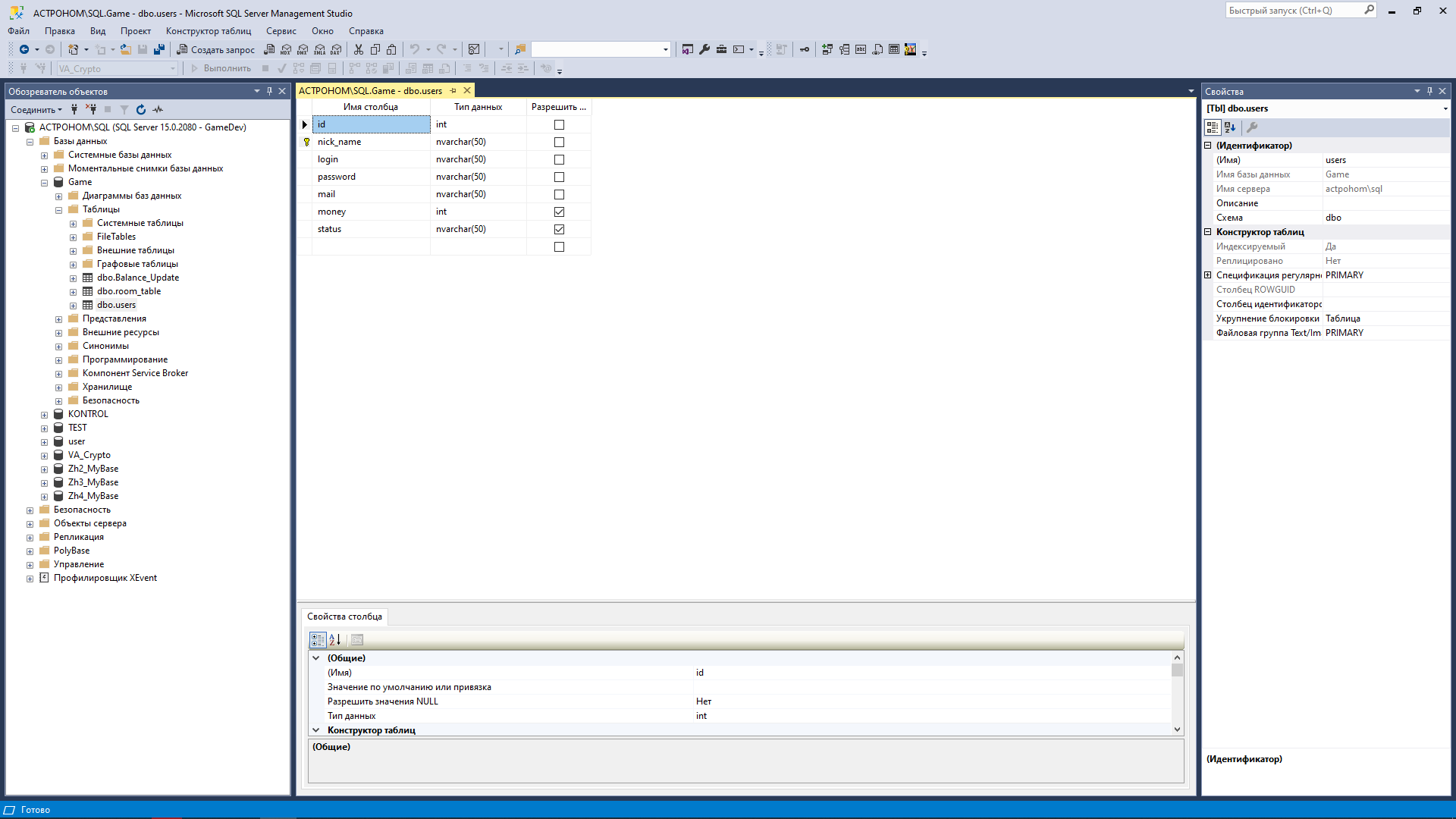


Рис. 2.2 – структура таблицы «Users»

Таблица Room\_table содержит информацию о доступных комнатах.

* Id\_room – уникальный идентификатор комнаты
* Name\_room – название комнаты
* Bet\_room – ставка в этой комнате
* Players\_in\_room – количество игроков в комнате
* Max players – максимально количество игроков в лобби
* Player 1 – никнейм игрока под номером 1
* Player 2 – никнейм игрока под номером 2
* Player 3 – никнейм игрока под номером 3
* Player 4 – никнейм игрока под номером 4
* Date start – время начала игры
* Date end – время конца игры
* Winner – победитель

Структура таблицы Room\_table представлена на рисунке 3.3.

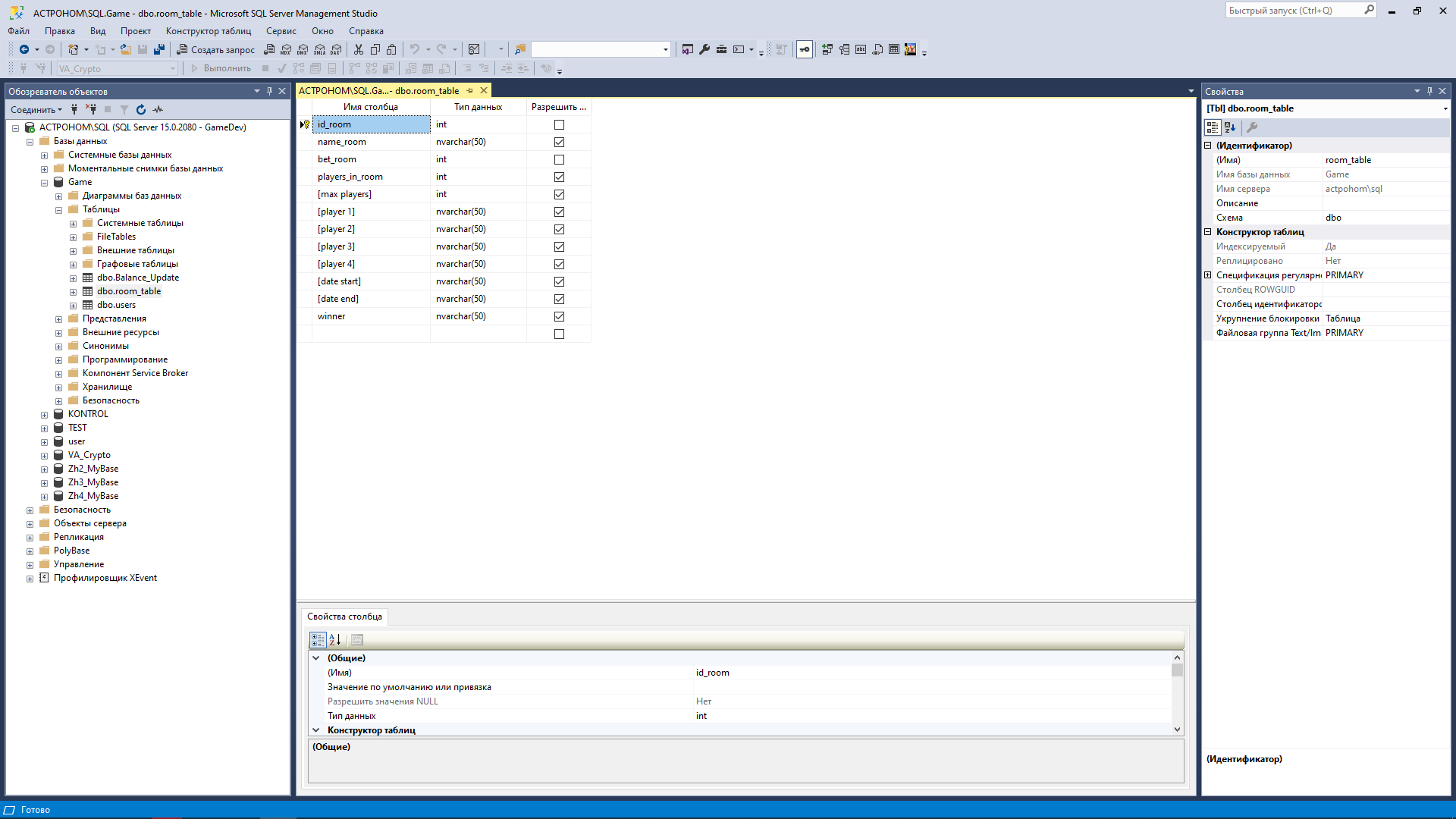


Рис. 2.3 – структура таблицы «Room\_table»

Таблица Balance\_Update содержит информацию о доступных комнатах.

* Id\_user – уникальный идентификатор игрока
* Card number – номер карты
* Money – на какое количество пополнили баланс
* Date – время пополнение

Структура таблицы Balance\_Update представлена на рисунке 3.4.



Рис. 2.4 – структура таблицы «Balance\_Update»

## **Диаграмма последовательности**

Далее была разработана диаграмма последовательностей, которая используется для уточнения диаграмм функций, которые представлены выше. Она показывает более детальное описание логики сценариев использования. На рисунке 2.5 представлена диаграмма последовательности.

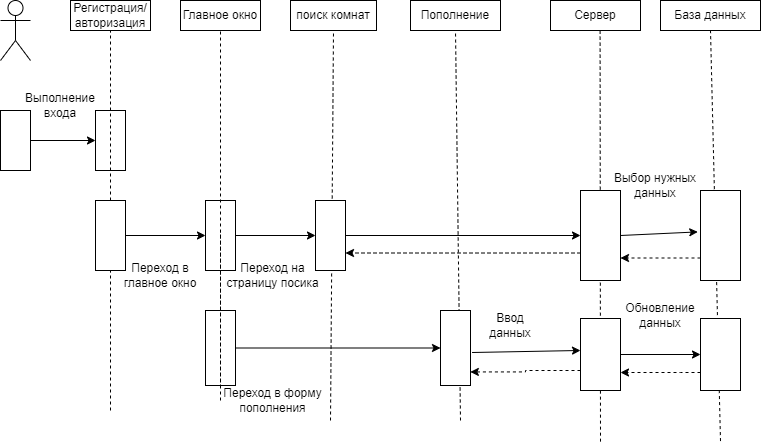


Рис. 2.5 – диаграмма последовательности

На данной диаграмме показана последовательность действий, происходящая при пополнении средств, поиске комнат. Сначала пользователь входит в приложение. Далее переходит на страницу поиска, после чего данные передаются серверу, сервер отправляет данных из бд пользователю. Аналогичным способом происходит общение в чате.

# Реализация программного средства

## **Обобщенная структура проекта**

Структура клиентской части проекта для разрабатываемого приложения представлен на рисунке 3.1.

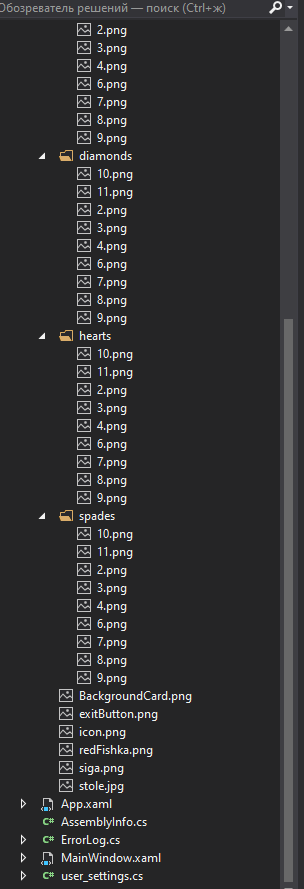
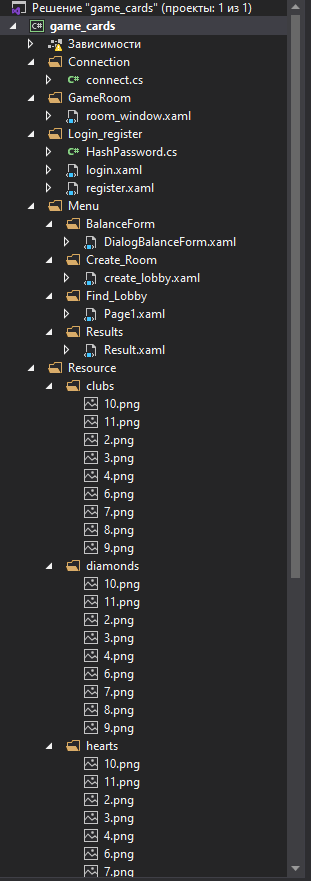


Рис. 3.1 – Структура клиентской части проекта приложения

В решении клиентской части присутствуют разные виды классов. Опишем их более подробно:

* файлы MainWindow.xaml, login.xaml, Page1.xaml, room\_window.xaml, register.xaml, FindLobby.xaml, DialogBalanceForm.xaml, Result.xaml являются основными классами, реализующими окна WPF приложения, и содержат xaml разметку. Их аналоги с расширением xaml.cs содержат всю необходимую логику взаимодействия, описанную на языке C#. Login представляет собой окно в котором мы производим авторизацию, перенаправляющий нас на Register – страницу регистрации пользователя или на MainWindow – главную страницу, в которой пользователь может найти подходящую для себя комнату, общаться в чате, просматривать результаты игр, создавать комнату. После подбора комнаты, пользователь попадает на страницу Room\_Window, в которой он ожидает других игроков и играет.
* папка Resource содержит всю графическую информацию, необходимую для приложения, такую как изображения и иконки (рис. 3.3);

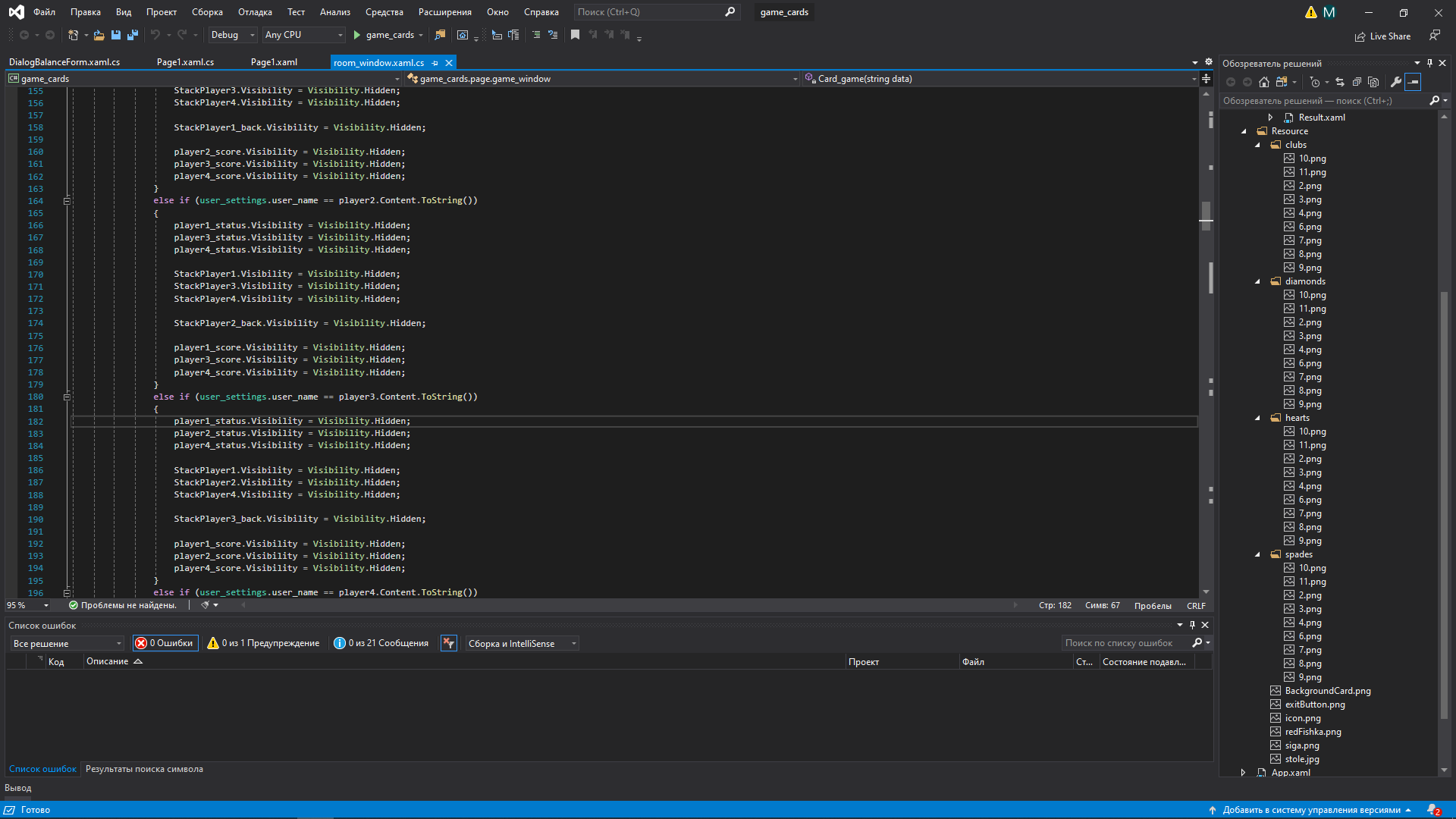


Рисунок 3.2 –Папка Resource

* FindLobby нужен чтобы найти свободные комнаты для игры.
* Create\_Lobby нужен чтобы создать новую комнату для игры.

Структура серверной части проекта для разрабатываемого приложения представлен на рисунке 3.2.

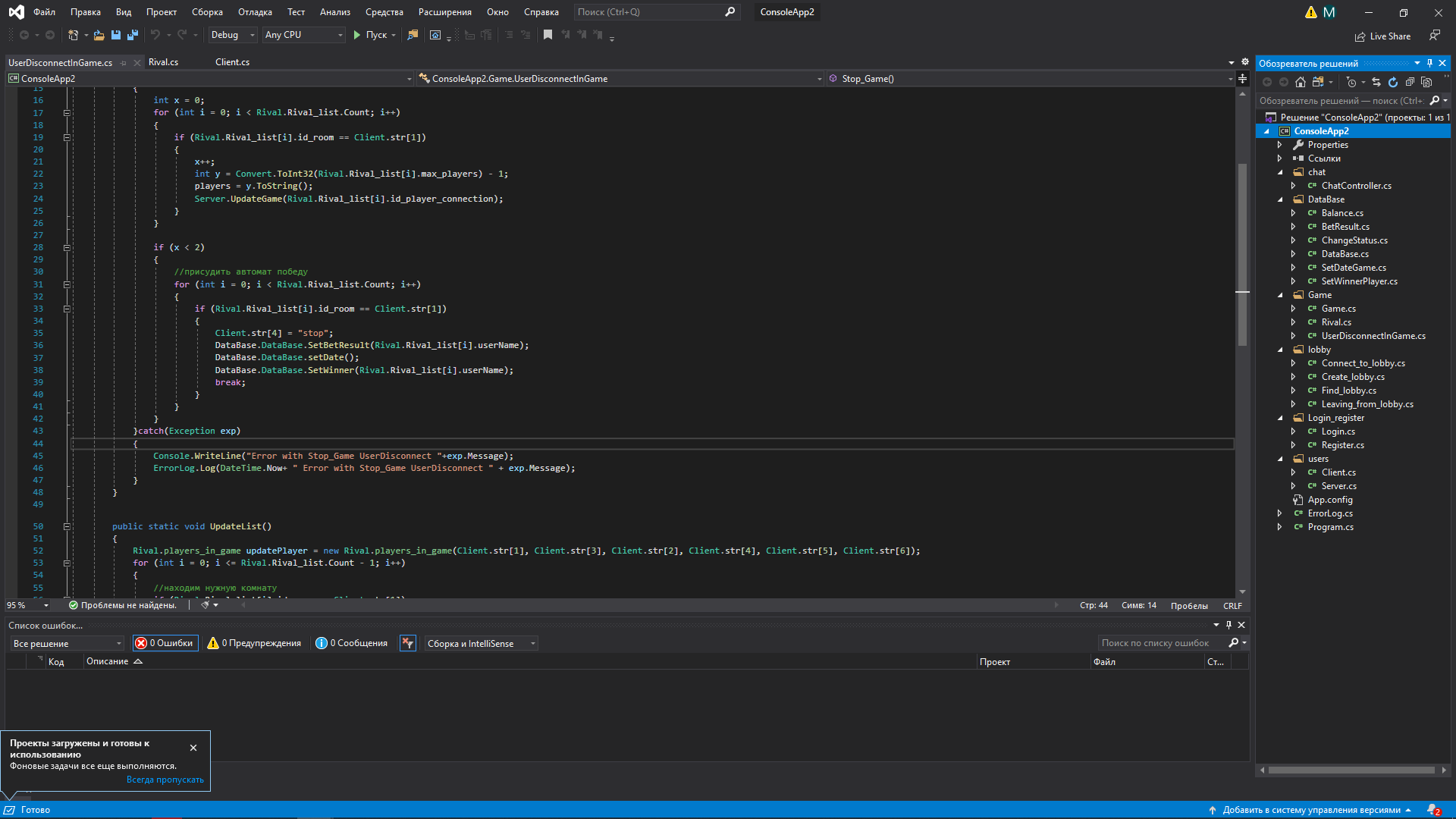


Рис. 3.3 – Структура серверной части проекта приложения

В решении серверной части присутствуют разные виды классов. Опишем их более подробно:

* файлы Program.cs, ErrorLog.cs, Server.cs, Client.cs, DataBase.cs, Find\_lobby.cs, Login.cs, Register.cs являются основными классами, на сервере. Стартовой точкой в сервере является Program.cs в нем реализован слушатель подключающихся пользователей. Поле чего они попадают в Server.cs где создается список подключений. В классе Client.cs создался новый поток, в котором сервер слушает запросы от пользователя и отдает ответы. Например, пользователь запрашивает список комнат, он отправляет запрос на сервер, сервер распознает команду и если команда связана с базой данных, то переходит в класс DataBase.cs где перенаправляет запрос в другой класс где читает БД и возвращает ответ.
* В папке lobby находятся все классы, относящиеся к поиску, созданию, подключению и выходу из лобби.
* В папке Game описана основная логика игры.
* В папке chat находиться класс контролирующий чат.
* В папке Login\_register находятся классы для регистрации и авторизации.
* В папке DateBase находятся файлы, работающие с базой данных.

## **Работа с базой данных в проекте**

Для того, чтобы осуществить подключение к базе данных MS SQL в WPF приложении, нужно создать базу данных (БД) и таблицы.

В качестве подключения в проекте используется ADO.Net. Для подключения нам нужно вручную указать имя сервера, логин и пароль если он имеется, имя базы данных и тип подключения изображение 3.5.

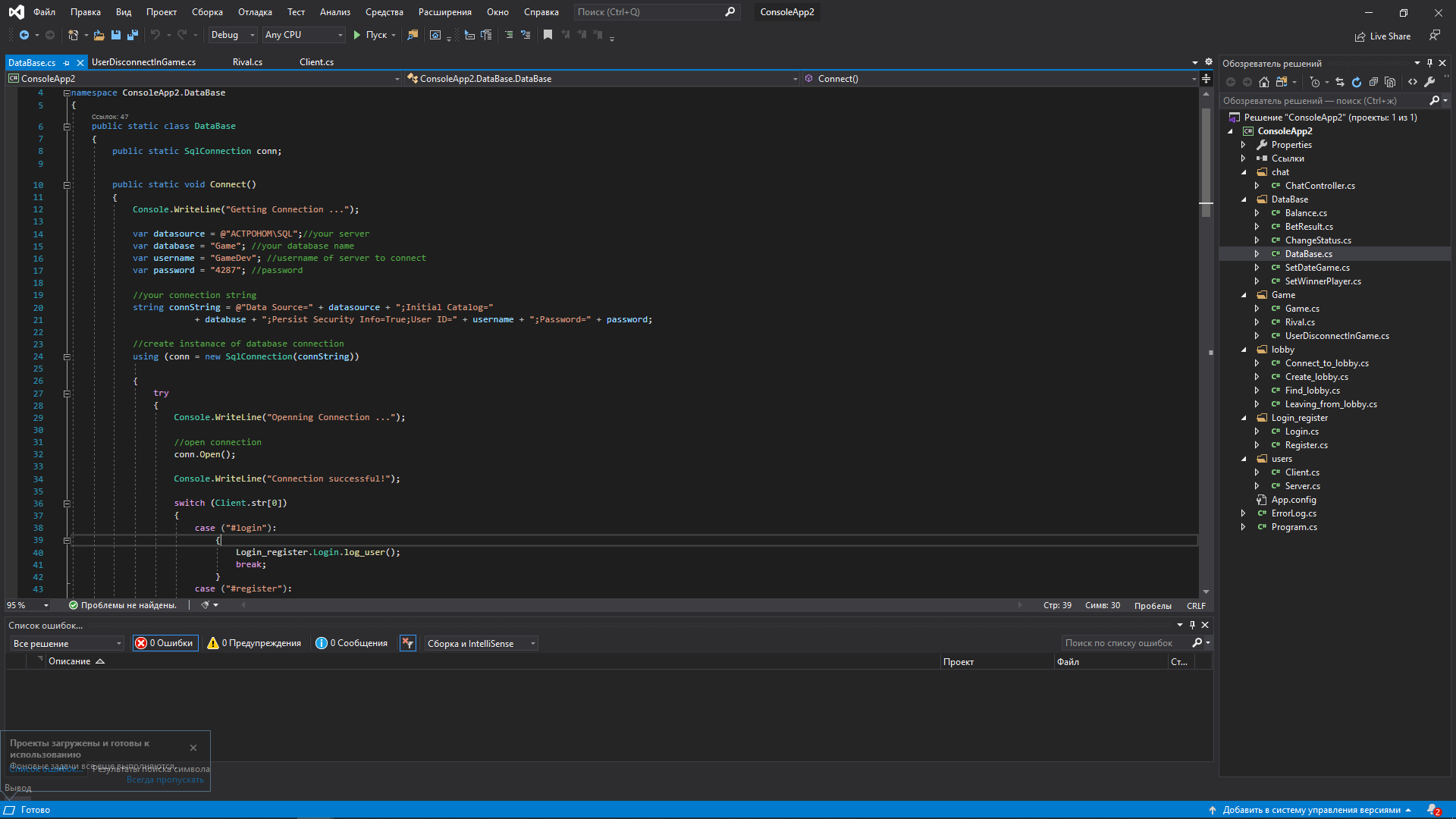


Рис. 3.5 – Строка подключения к БД

Далее вся работа с БД реализована через класс DataBase.

## **Создание связи между клиентом и сервером.**

Для создания связи между клиентом и сервером был использован C# Socket. На сервере открывается порт и задается IP-адрес (Рис. 3.6).

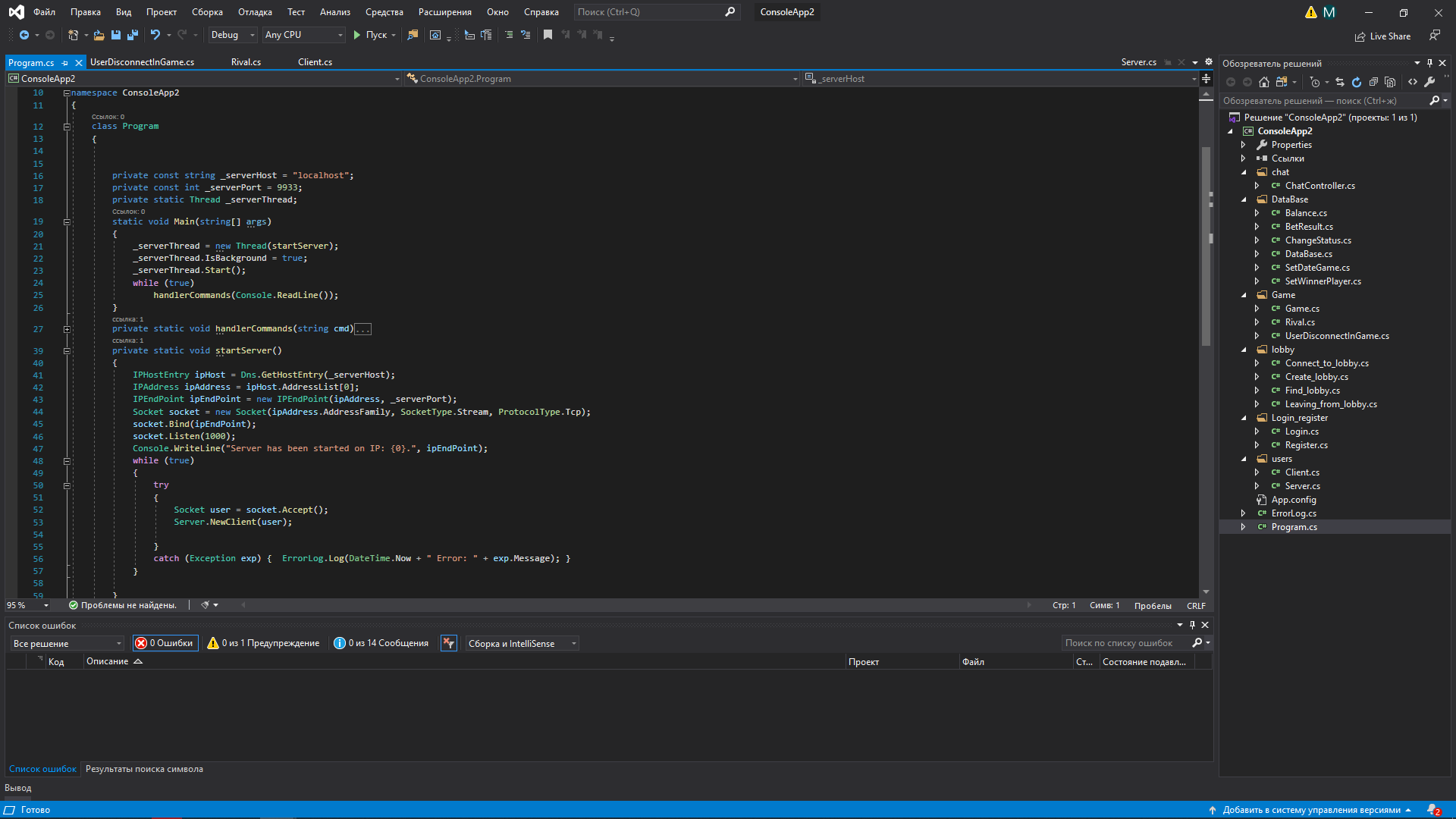


Рис. 3.6 – Точка открытия Socket-а

Используемые библиотеки: using System.Net;

using System.Net.Sockets;

* AddressFamily: возвращает все адреса, используемые сокетом. Данное свойство представляет одно из значений, определенных в одноименном перечислении **AddressFamily**. Перечисление содержит 18 различных значений, наиболее используемые:
  + InterNetwork: адрес по протоколу IPv4
  + InterNetworkV6: адрес по протоколу IPv6
  + Ipx: адрес IPX или SPX
  + NetBios: адрес NetBios
* SocketType: возвращает тип сокета. Представляет одно из значений из перечисления **SocketType**:
  + Dgram: сокет будет получать и отправлять дейтаграммы по протоколу Udp. Данный тип сокета работает в связке с типом протокола - Udp и значением AddressFamily.InterNetwork
  + Raw: сокет имеет доступ к нижележащему протоколу транспортного уровня и может использовать для передачи сообщений такие протоколы, как ICMP и IGMP
  + Rdm: сокет может взаимодействовать с удаленными хостами без установки постоянного подключения. В случае, если отправленные сокетом сообщения невозможно доставить, то сокет получит об этом уведомление
  + Seqpacket: обеспечивает надежную двустороннюю передачу данных с установкой постоянного подключения
  + Stream: обеспечивает надежную двустороннюю передачу данных с установкой постоянного подключения. Для связи используется протокол TCP, поэтому этот тип сокета используется в паре с типом протокола Tcp и значением AddressFamily.InterNetwork
  + Unknown: адрес NetBios

При работе с сокетами вне зависимости от выбранных протоколов мы будем опираться на методы класса Socket:

* Accept(): создает новый объект Socket для обработки входящего подключения
* Bind(): связывает объект Socket с локальной конечной точкой
* Close(): закрывает сокет
* Connect(): устанавливает соединение с удаленным хостом
* Listen(): начинает прослушивание входящих запросов
* Poll(): определяет состояние сокета
* Receive(): получает данные
* Send(): отправляет данные
* Shutdown(): блокирует на сокете прием и отправку данных

В зависимости от применяемого протокола (TCP, UDP и т.д.) общий принцип работы с сокетами будет немного различаться.

# Тестирование

## **Использование приложения пользователем**

При запуске приложения пользователь видит начальную страницу, в которой ему предлагается либо авторизоваться или зарегистрироваться (Рис 4.1).

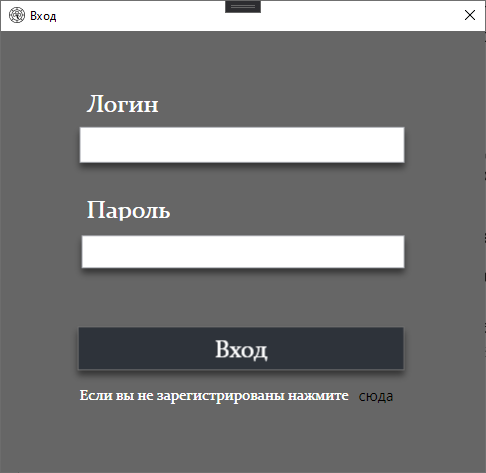


Рис. 4.1 Начальная страница.

На данной странице присутствует валидация в случаи не ввода пароля или же логина вас оповестит система сообщением (рис 4.2).

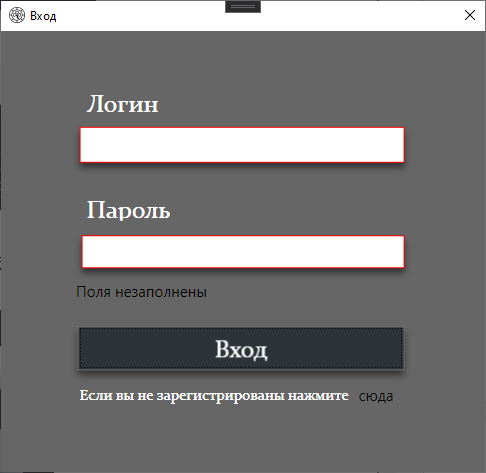


Рис. 4.2 – Оповещение что поля пустые

А при попытке отправить данные на выключенный сервер возникает ошибка (рис 4.3).

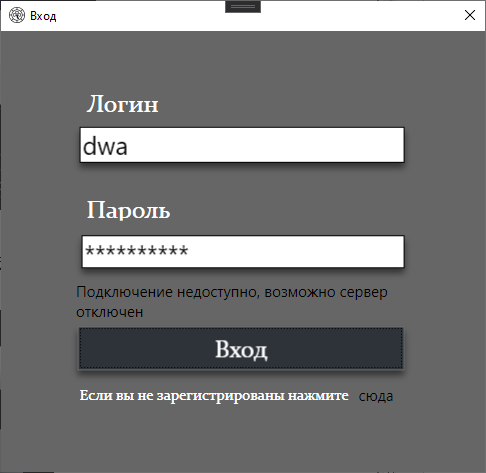


Рис. 4.3 – При выключенном сервере

Если же данные не верны то возникает оповещение (рис 4.4).

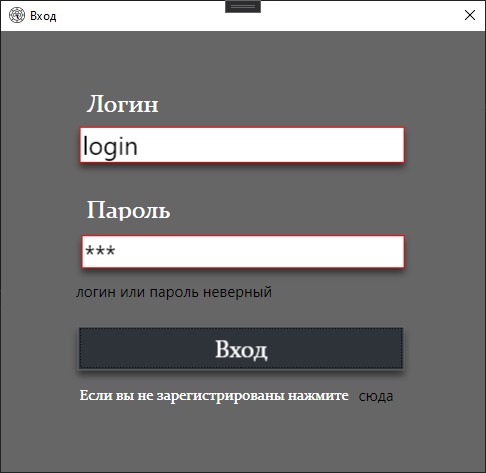


Рис. 4.4 – Введены неверные данные

При выборе «Зарегистрироваться» открывается страница для регистрации нового пользователя рисунок 4.5.

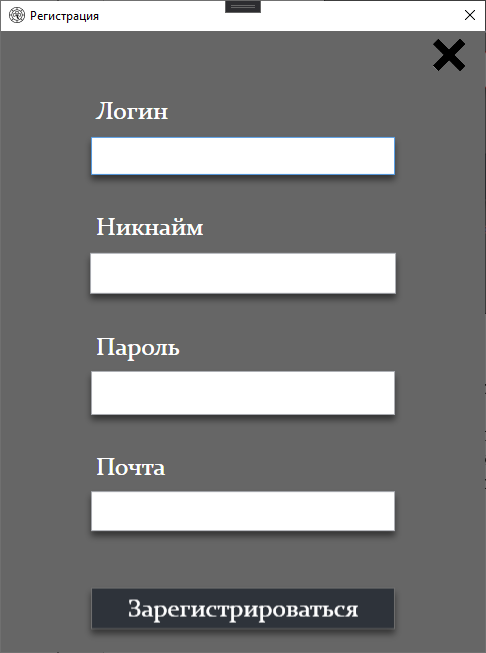


Рис. 4.5 – окно регистрации

На данной странице также присутствует валидация, а именно при вводе логина запрещаются все символы кроме латинских и цифр рисунок 4.6.

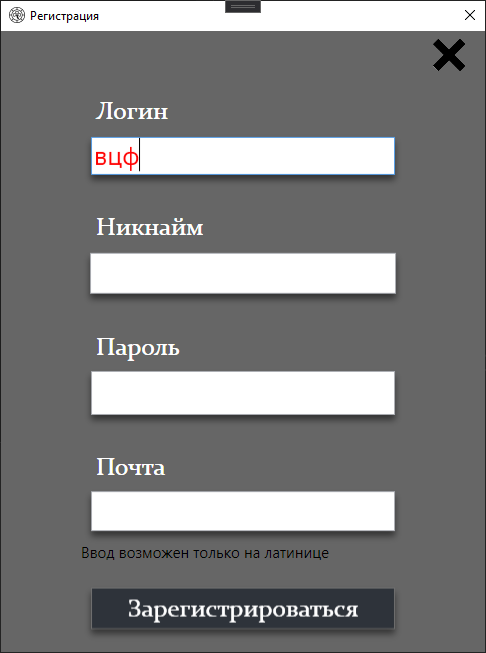


Рис. 4.6 – валидация на символы

Есть проверка на корректный ввод mail почты на которую будет отправляется письмо с кодом рисунок 4.7.



Рис 4.7 – валидация почты

Есть проверка на совпадение Логина и Никнейма а так же Почты рисунок 4.7 4.8 4.9.

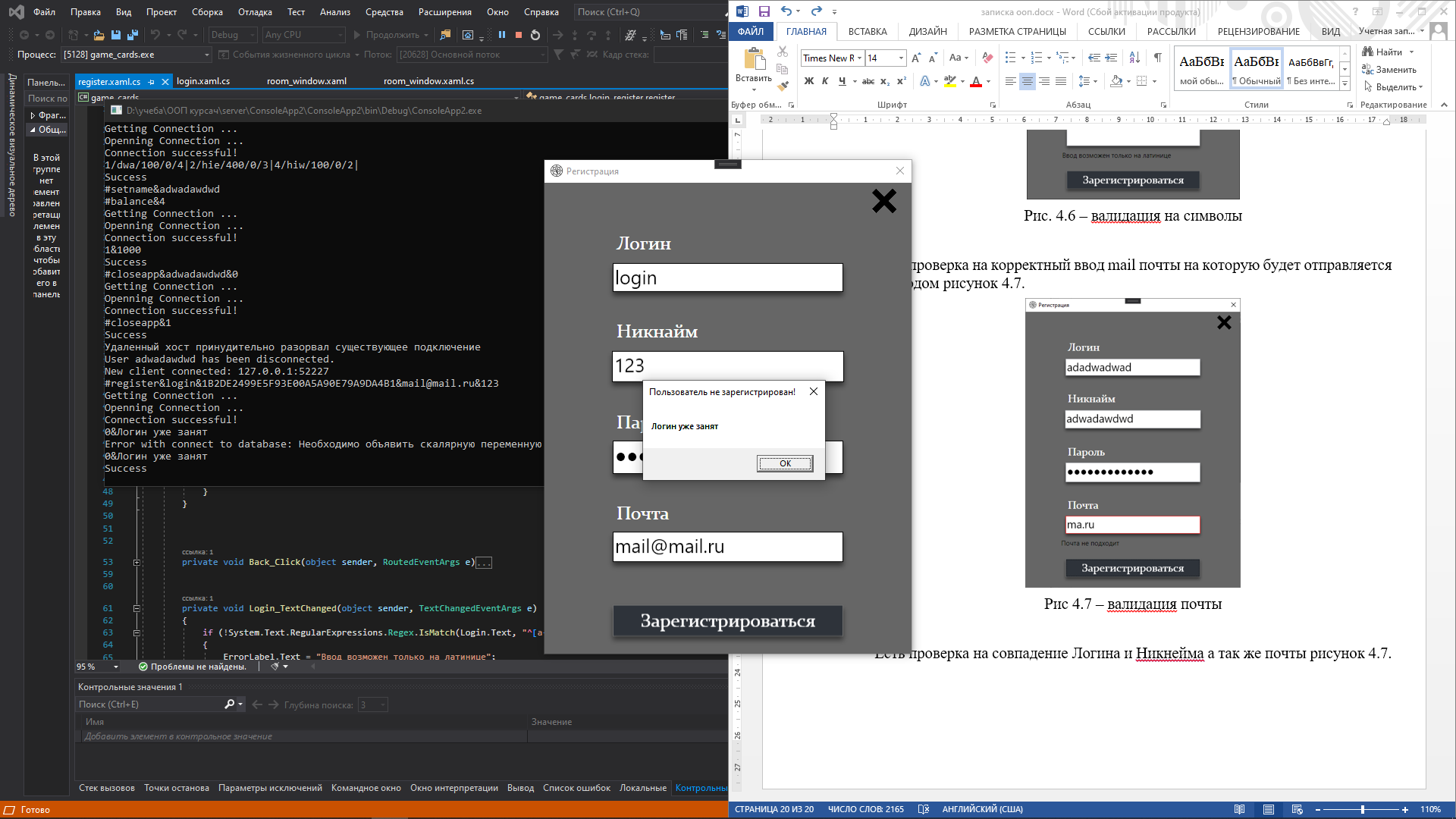


Рис 4.7 – проверка на логин

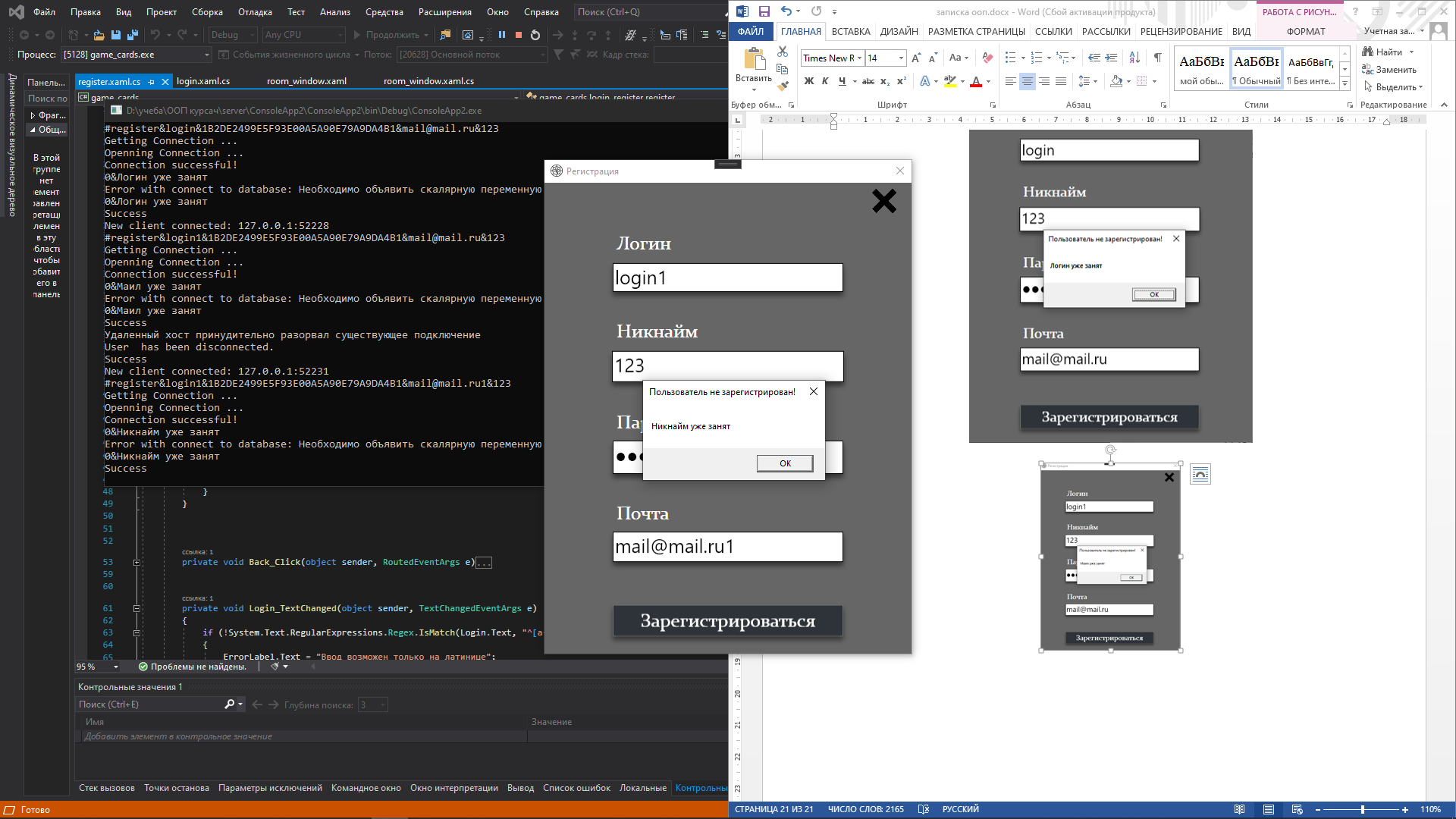


Рис 4.8 – проверка на никней

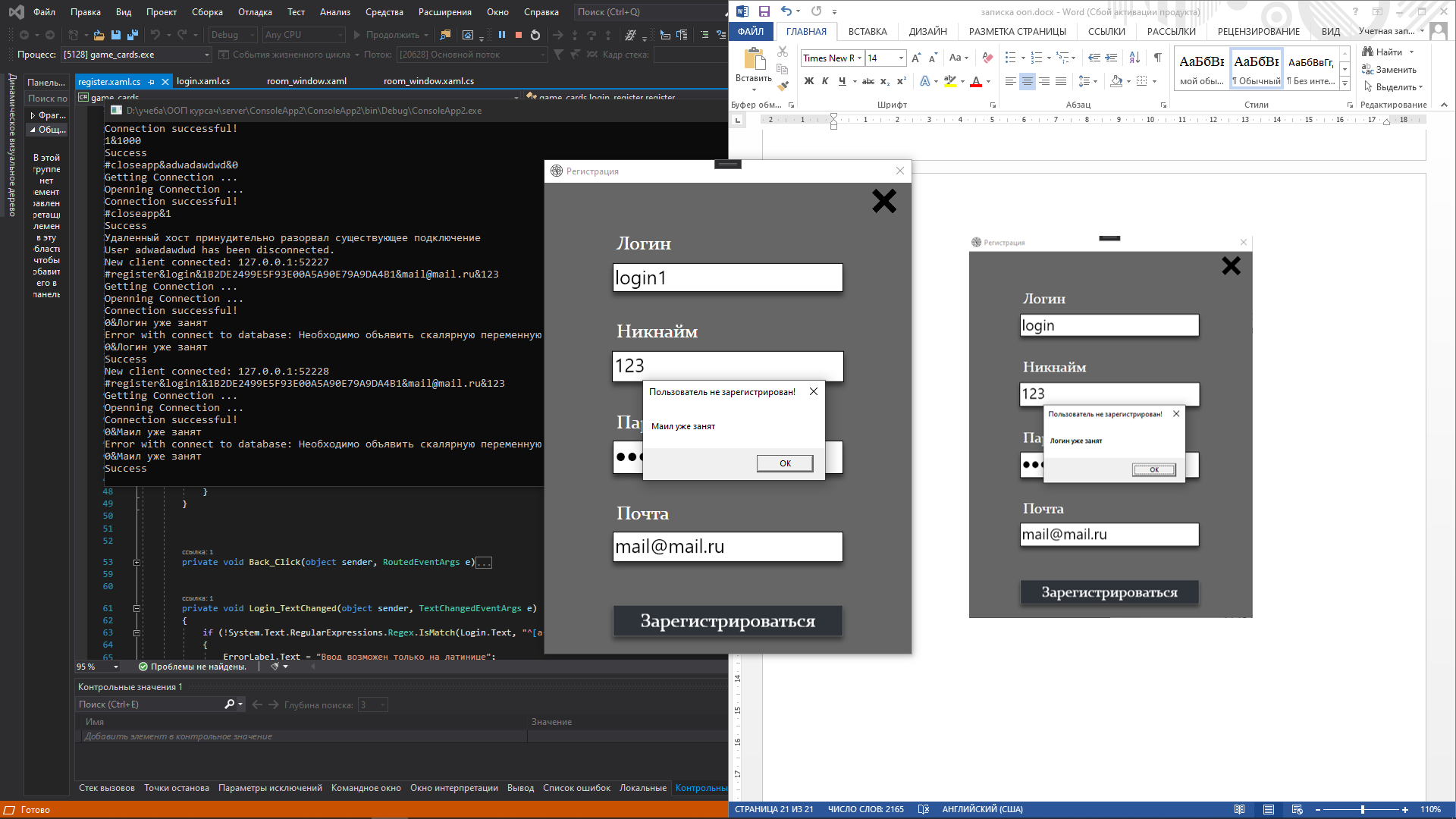


Рис 4.9 – проверка на маил

После успешного прохождения аутентификации пользователь попадает на главную страницу приложения с его всеми кошельками рисунок 4.10.

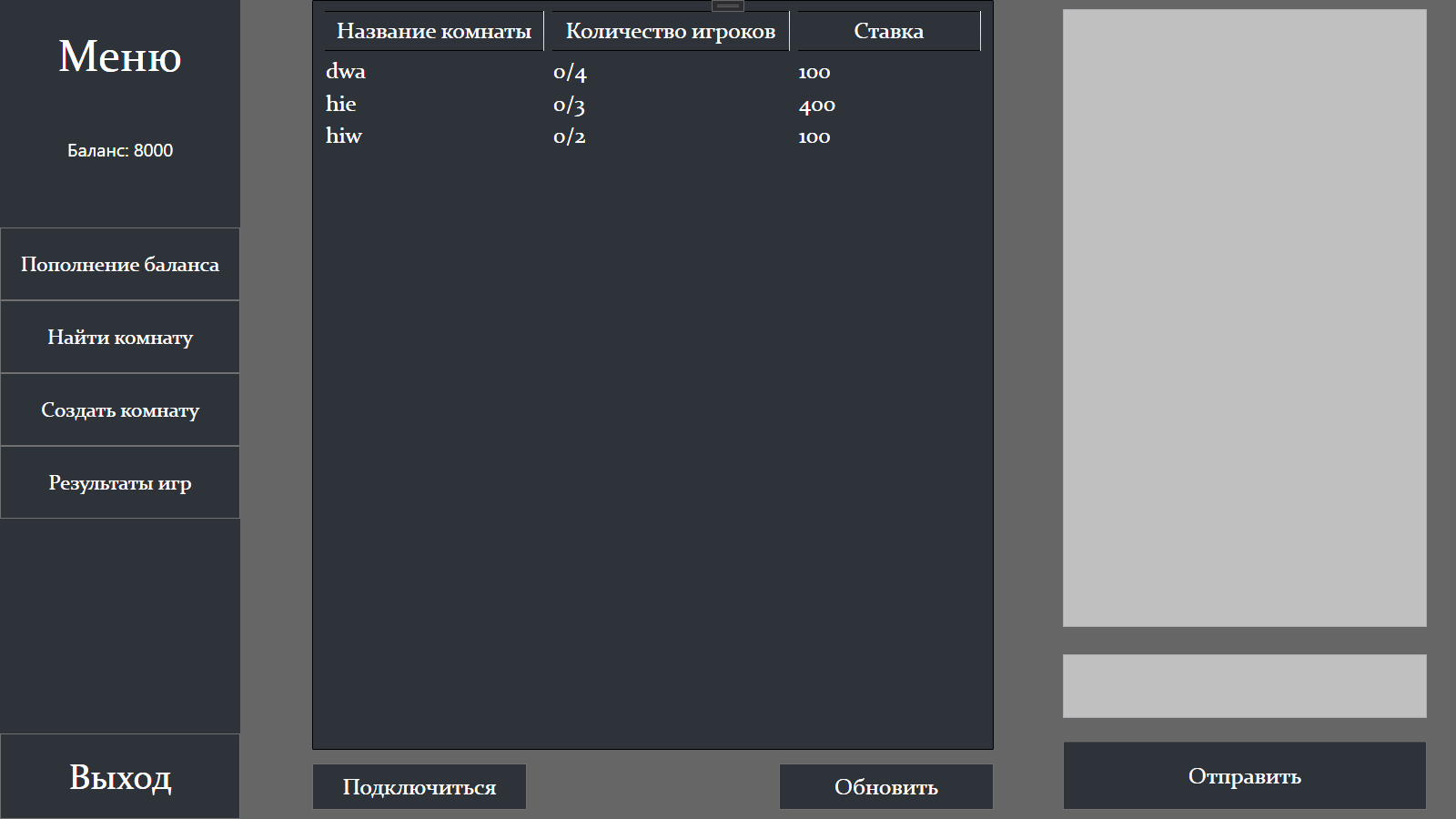


Рис 4.10 – Главное окно

Пользователь может написать в чат (Рис. 4.11).

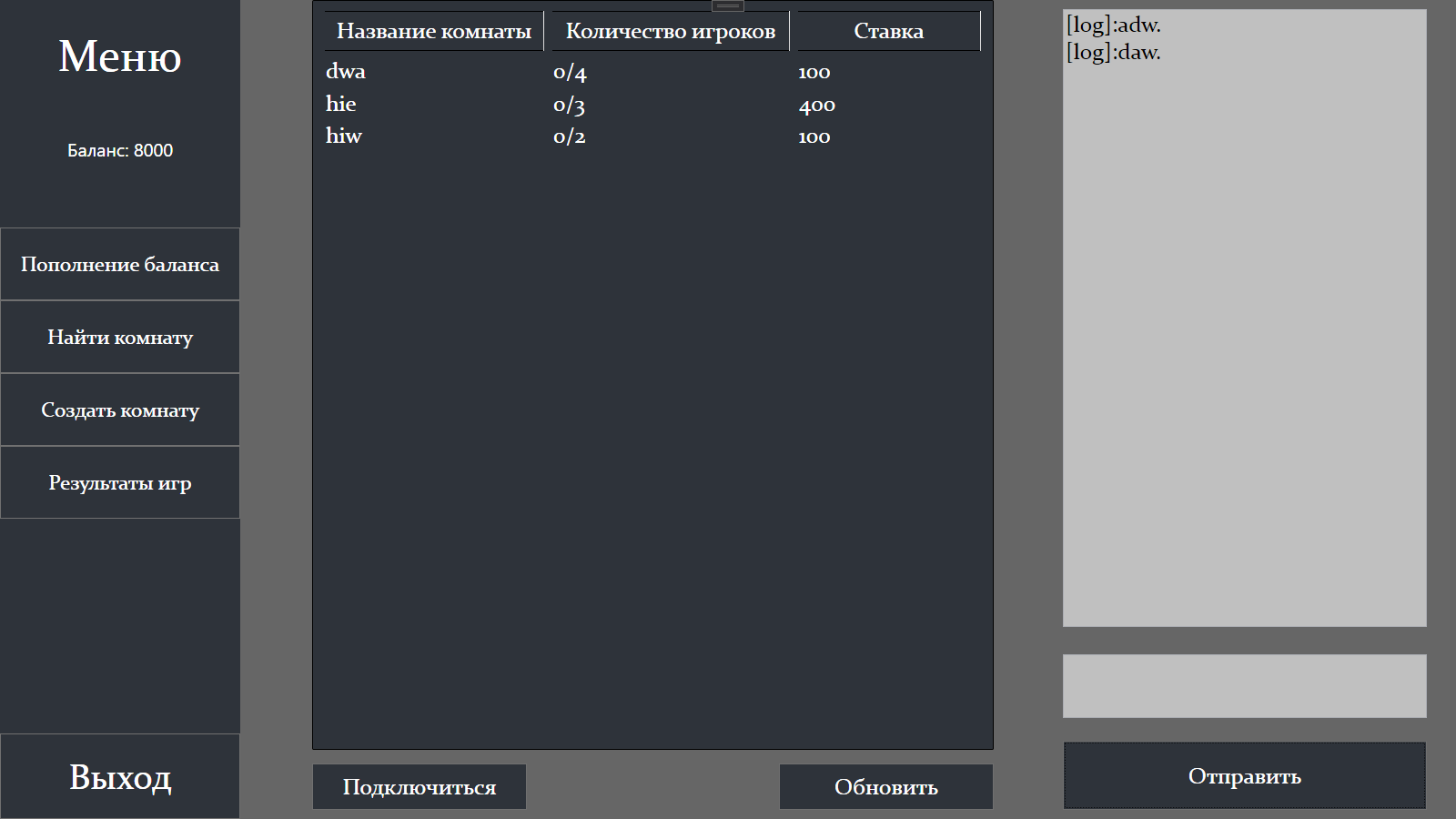


Рис 4.11 – Использование чата

Может пополнить баланс (Рис 4.12).

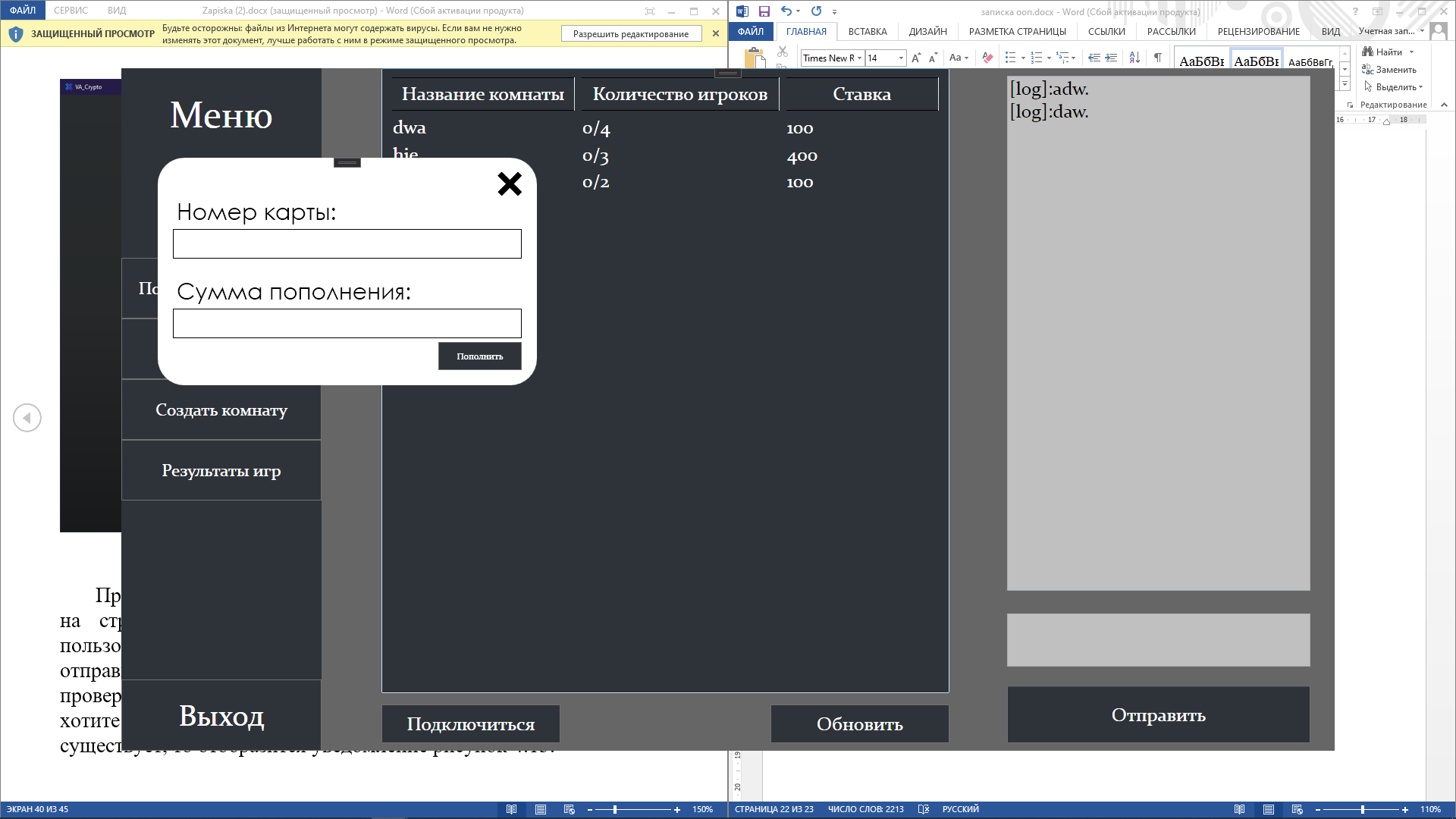


Рис 4.12 – окно пополнения баланса

Пользователь может создать комнату с ставкой от 100 до 1000 и от 2 до 4 человек (Рис 4.13).

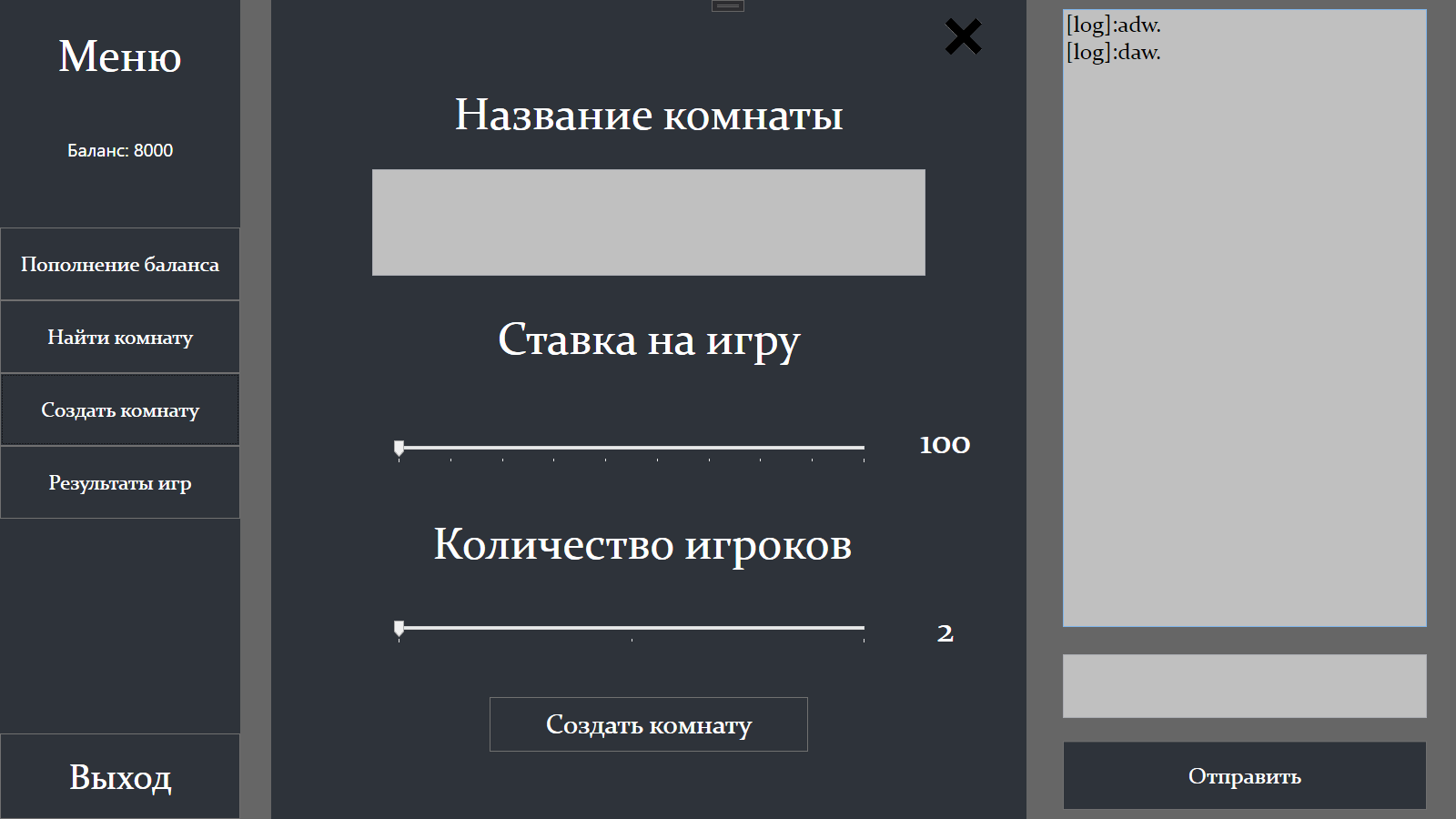


Рис 4.13 – Создание комнаты

Пользователь может посмотреть результаты игр, в которых он принимал участие (Рис 4.14).

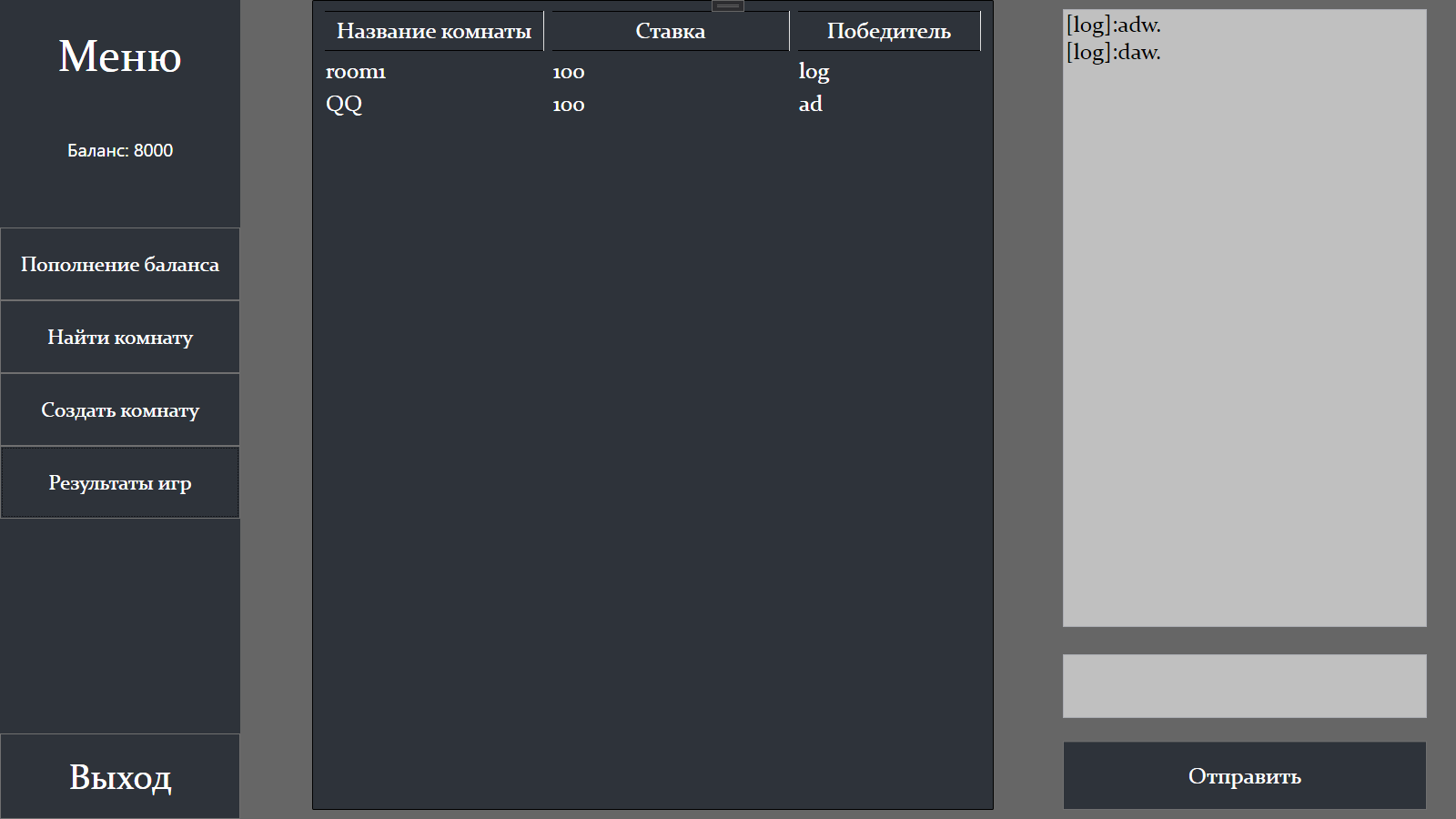


Рис 4.14 – Просмотр результатов игры

Также он может найти и сыграть в одной из комнат (Рис 4.15).



Рис 4.15 – Поиск комнат для игры

Игра в Двадцать одно (Рис 4.16).

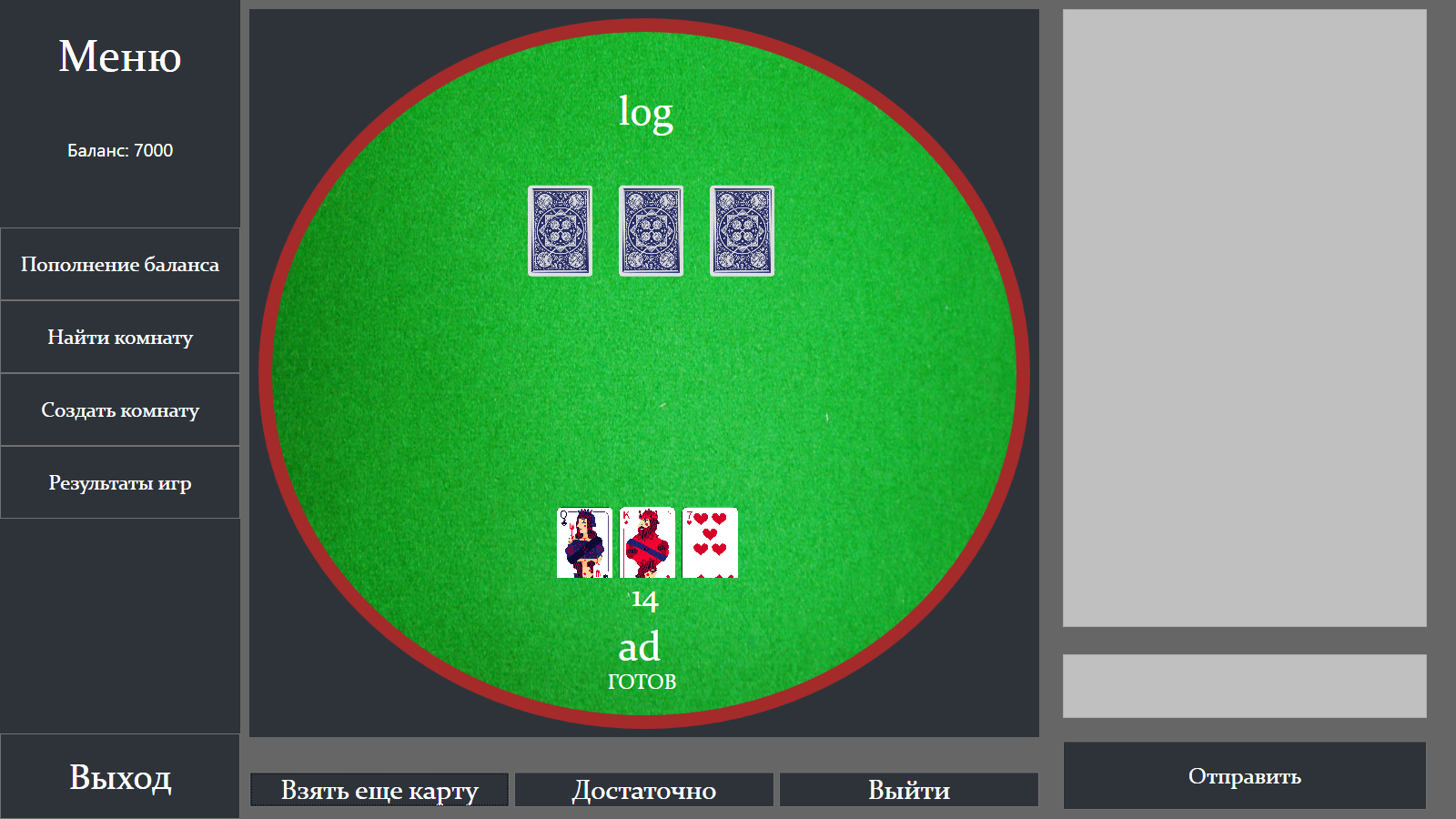


Рис 4.16 – Игра в Двадцать одно

# Руководство по использованию

## **Настройка перед использованием**

Для начала вам понадобиться Microsoft SQL Server Management Studio, после чего надо установить сервер и задать ему IP-адрес, в нашем случае localhost, так же откроем любой порт на вкус. И настроем подключение к БД

Так как у меня есть аккаунт для подключения к базе данных через Username и password, то я использую этот код

var datasource = @"ACTPOHOM\SQL";//your server

var database = "Game"; //your database name

var username = "GameDev"; //username of server to connect

var password = "4287"; //password

//your connection string

string connString = @"Data Source=" + datasource + ";Initial Catalog="

+ database + ";Persist Security Info=True;User ID=" + username + ";Password=" + password;

Если у вас нет такого пользователя, то можно подключится через проверку подленности Microsoft через такой код:

connString = @"Data Source=ACTPOHOM\SQL;Initial Catalog=TEST;Integrated Security=True";

Где @"Data Source=ACTPOHOM\SQL это ваш сервер и Catalog=TEST это база данных.

Как только вы это сделаете вам надо создать таблицы в бд (Рис 5.1):

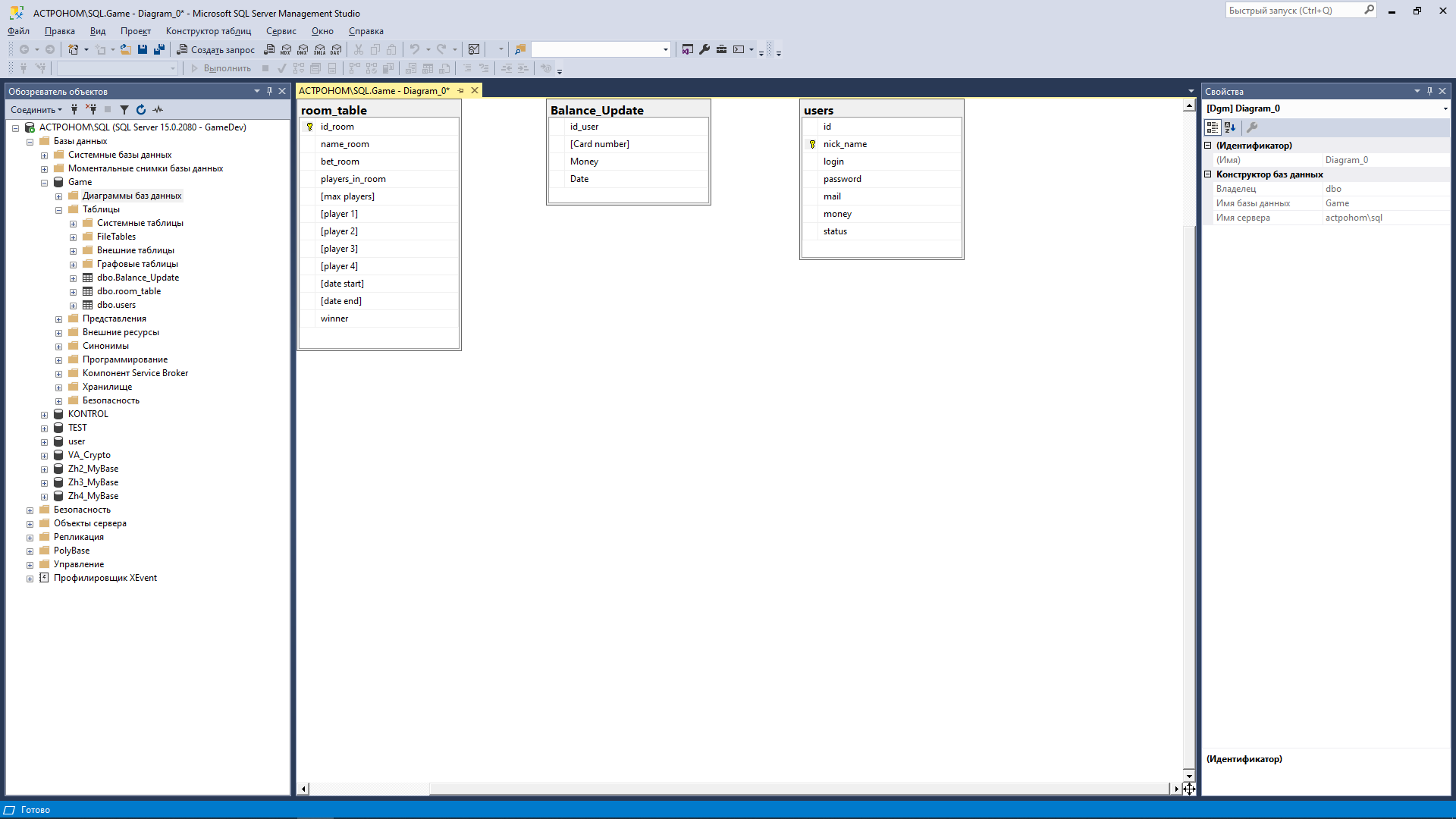


Рис 5.1 – Таблицы базы данных

В клиенте в папке Connection есть класс connect, где находиться подключение к серверу. Тут указывает IP-адрес и порт.

Также советую запустить ServerController, его задача включать сервер если тот по какой-то причине был выключен.

Дальше программа готова к работе.

## **Регистрация и авторизация**

При запуске приложения запускается окно авторизации, в которое выводится представление с формой для авторизации. Если у пользователя еще нет аккаунта, ему следует нажать на кнопку «Регистрация», которая его перенаправит на представление с формой для регистрации (рис. 4.5). Представление авторизации представлено на рисунке 4.1.

## **Главное окно**

После авторизации вы попадаете в главное окно, слева от вас меню, в котором вы можете выбрать действия: Пополнить баланс, Найти комнату, Создать комнату, Результаты игр, а также кнопка выхода (Рис 5.1).

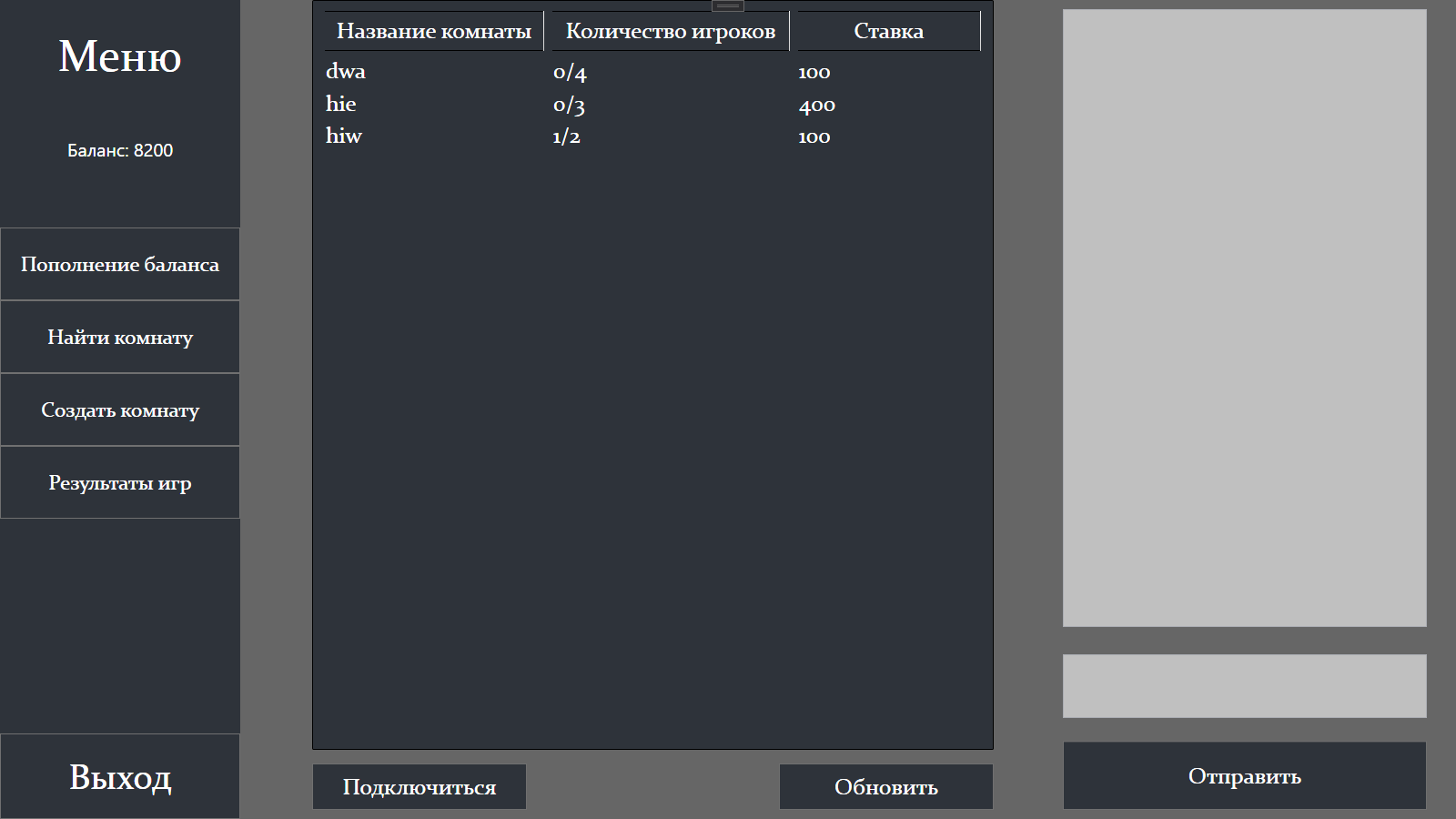


Рис 5.1

Подключение к комнате происходит через выделение строчки в списке (ListView) и последующим нажатием на кнопку “Подключиться” или двойным нажатием на нужный элемент в ListView.

После того как вы войдете в игровую комнату вам придется ждать вашего соперника для того чтобы начать игру.

При нажатии на кнопку “обновить”, обновляется список комнат.

Пользователю также доступен чат в любой момент времени, для общения с игроками.

В вкладке Пополнить баланс, пользователь должен ввести номер карты и сумму пополнения.

В вкладке Создать комнату, пользователь должен указать название комнаты, ставку и количество игроков, после чего нажать “СОЗДАТЬ”.

Для дальнейшей корректной работы выход осуществлять через кнопку “Выход”.

**Заключение**

Итогом курсовой работы стало клиент – серверное приложение, предназначенное для игры в Двадцать одно. В ходе выполнения был проведен тщательный анализ, который включал в себя обзор аналогов, проверка актуальности данного ПО, постановка целей и задач, формирование технического задания.

Было проведено проектирование, включающее разработку структуры базы данных, разработку архитектуры приложения, разработка пользовательского интерфейса, разработку классов и других функциональных частей.

Благодаря подготовительным этапам получилась стабильная реализация приложения. Тщательно спроектированная база данных позволит в будущем без значительных её изменений добавить огромную функциональную составляющую, позволяющую превратить клиент – серверное приложение в одну из лучших программ для онлайн игры Двадцать одно.

На последних этапах разработки клиент – серверное приложения было протестировано для выявления неточностей в работе, а также для всех пользователей было предложено руководство по пользованию.

Для выполнения курсовой работы одним из важнейших этапов стало изучение материала по таким направлениям, как «Объектно-ориентированное программирование на языке С#», «Реляционные базы данных SQL на основе SQL Server» и «Система для построения клиентских приложений WPF».

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список использованной литературы

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com. Дата доступа: 23.04.2021
2. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
3. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
4. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> – Дата доступа: 10.04.2021.

**Приложение А**

