**Компьютерная академия «ШАГ»**

**Донецкий филиал**

**Кафедра Разработки программного обеспечения**

**КУРСОВОЙ ПРОЕКТ**

**“** **Программа - Чат ”**

**Студента группы ППВ-23-01  
Москалёва Д. А.**

**Научный руководитель:  
Стрюк Д. О.**

**Донецк, 2015**

**Техническое задание**

Чат на основе протокола TCP. В чате должна бать история сообщений, список контактов, приватная, групповая и массовая рассылка, черный список, изменение состояний пользователей. У чата должен быть сервер, написанный на WCF, который управляет работой чата, позволяет администратору менять настройки.

Администратор сайта должен иметь возможность регистрировать активность пользователей, видеть историю активности по каждому пользователю, добавлять пользователей в существующие группы. Администратору предоставляется возможность добавлять пользователей в чёрный список, удалять их из чёрного списка и блокировать на время.

Клиенты будут видеть список групп и, при желании, будут отсылать запрос администратору на добавление. Администратор оставляет за собой право отклонить заявку.

Для каждой группы ведётся отдельная история сообщений. Клиенты, выбирая определённые группы, видят сообщения, находящиеся в этой группе.

Вся история хранится в базе данных и восстанавливается при каждом запуске сервера администратором.

Технологии

При выполнении курсовой работы были использованы следующие технологии:

* C#
* WPF
* WCF
* SP
* .NET
* Transact-SQL
* ADO.NET

Программа написана в системе **Windows Presentation Foundation (WPF)**

Решение состоит из трёх проектов. Первый проект содержит в себе инструментарий для работы с базой данных и адаптирует данные для работы в программе (создание новых классов: User, Group и т.д.) Второй проект отвечает за внешний вид серверной части приложения и функциональные возможности. Третий проект - клиентская часть и её функциональные возможности.

При запуске программы запускается панель администратора. В ней отображаются существующие пользователи, группы и заявки пользователей на вступление в группы.

Данные для работы вынимаются из БД и сохраняются в List<T> соответствующих классов, определённых в библиотеке классов ChatInfoDAL. После вызова метода FirstInit(). После первичной инициализации форма администратора заполняется основными данными: все зарегистрированные пользователи и группы.

Далее создаются три объекта класса Socket

reg\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

IPEndPoint ipEndPointReg = new System.Net.IPEndPoint(IPAddress.Any, 3000);

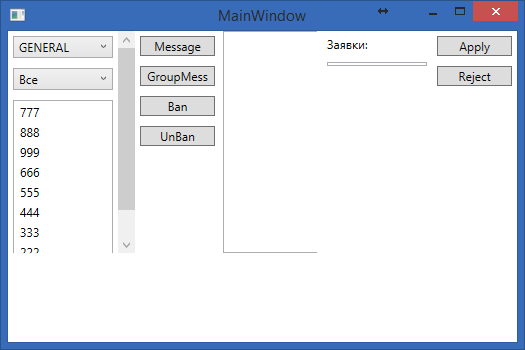
login\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

IPEndPoint ipEndPointLog = new System.Net.IPEndPoint(IPAddress.Any, 4000);

server\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

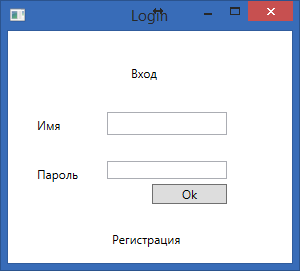
IPEndPoint ipEndPoint = new System.Net.IPEndPoint(IPAddress.Any, 2000);

Объекты отличаются только портами. 3000 – для регистрации; 4000 – для входа; 2000 – для работы с подключившимся пользователем. Все сокеты становятся на прослушку входящих соединений в отдельных потоках. При обнаружении подключения сервер и клиент обмениваются данными. При регистрации проверяется существование пользователя. Если пользователь ещё не зарегистрирован ему оправляется ответ в приветствием. Если форма для регистрации или входа получает приветствие, она прекращает свою работы, предварительно запустив форму клиента и передав ей имя залогинившегося(зарегистрировавшегося) пользователя.

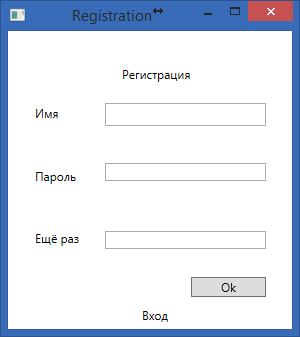


Окно входа пользователя имеет следующий вид. При желании можно перейти на форму регистрации, нажав на Labal с текстом Регистрация.

После нажатия кнопки OK данные из текстовых полей сохраняются в экземпляре класса LoginInfo. Класс помечен как сериализуемый для отправки данных на сервер через соединение Socket. Дальнейшие действия описаны выше.



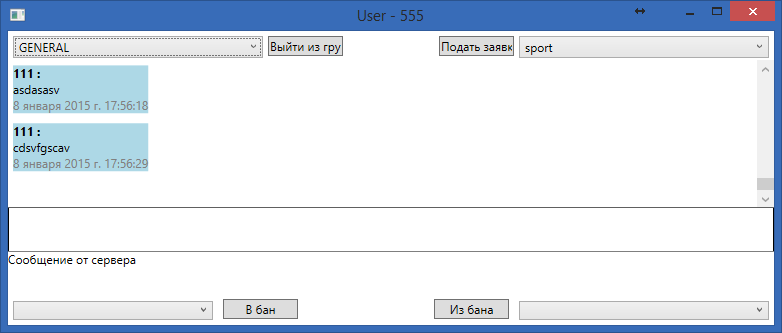
Форма для регистрации:



После нажатия клавиши OK данные записываются в экземпляр класса RegInfo, предварительно помеченного как сериализуемый.

Следует отметить, что одновременно могут быть запущенны несколько форм для регистрации или входа, которые в свою очередь в дальнейшем могут не однократно запускать экземпляры Client.

Клиентская форма выглядит следующим образом:



Во время её инициализации сервер отправляет поочерёдно все необходимые данные на клиента. Список групп, список пользователей, сообщения из групп, в которых состоит пользователь.

Рабочая область разделена на секции в помощью элементов Grid, DockPanel, StackPanel.

В верхней части расположен ComboBox с возможность выбора групп сообщений, выхода из них и подачи заявки на вступление.

Выход из группы происходит путём отправки сообщения серверу в обектом класса ClaimToGroup с данными пользователя и группы. Сервер удаляет объект MemberOfGroupManager.MembersOfGroup.Remove(mem); из List с данными о членстве в группах и перезаписывает данные в БД

string query = string.Format("delete from MemeberOfGroups where UserID={0} and GroupID={1}", claimToGroup.User.UserID, claimToGroup.Group.Id);

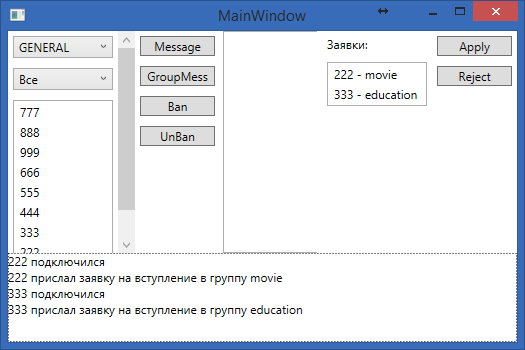
SqlCommand command = new SqlCommand(query, DataBase.Sql);

command.ExecuteNonQuery();

В конце делается рассылка на всех активных пользователей этой группы о выходе пользователя из группы.

Если пользователь хочет вступить в группу, то он выбирав название группы, отправляет заявку на вступление адинистратору, путём создания и отправки экземпляра класса ClaimToGroup.

Все заявки отображаются в админской панели. Администратор может принять или отклонить заявку.



Если администратор принимает заявку, уведомление отправляется на всех пользователей группы. На нового пользователя отправляется ещё и List<Message> сообщений группы.

В БД делаются соответствующие изменения.

Основная часть окна – сообщения пользователей. Технически каждое сообщение представляет собой три элемента TextBlock, содержащихся в объекте StackPanel, который свою очередь помещается в StackPanel со всеми сообщениями, обёрнутый в ScrollViewer для их прокрутки.

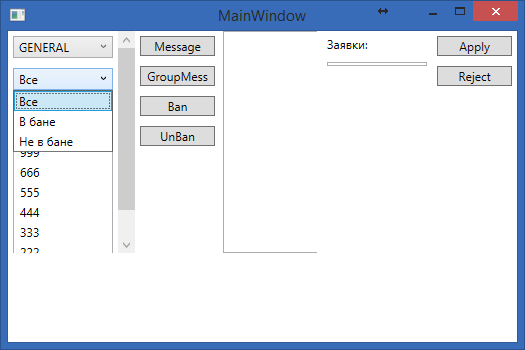
Ниже расположен TextBox для ввода сообщения пользователя.

После нажатия клавиши Enter текст, информация о группе и пользователе инкапсулируются в объект класса Message и отправляются серверу. Сервер проверяет не забанен ли пользователь и отправляет сообщение на всех активных юзеров из группы, к которой относится сообщение.

При приёме нового сообщения пользователями, алгоритм решает выводить сообщение на экран или нет, исходя из List<BanPair> (списка индивидуальных банов).

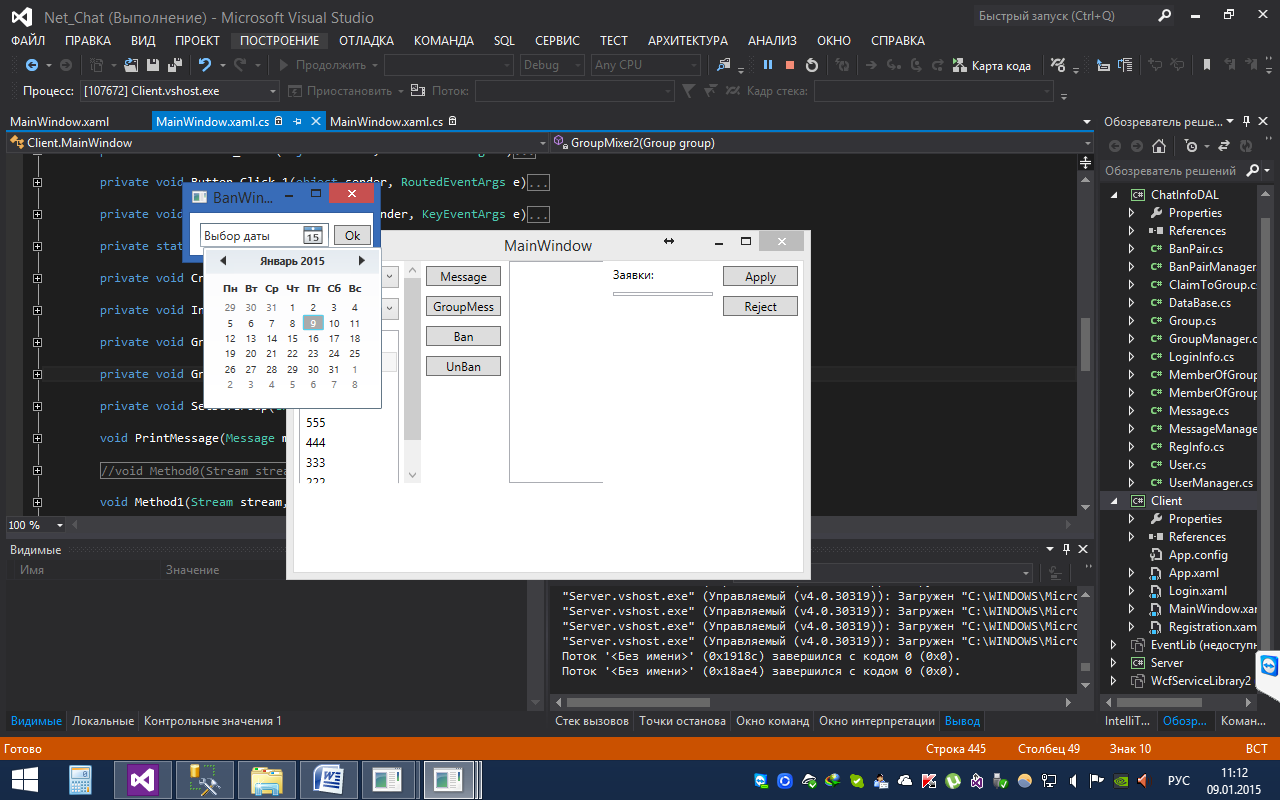
Ещё ниже в клиентском окне рсположена панель для вывода сообщений от сервера о входе и выходе пользователей, о банах сервера и банах от других пользователей, а также обычных текстовых сообщений от сервера при групповой или адресной рассылке.

В самом низу два ComboBox с пользователями забанеными и активными. При желании пользователь может забанить одного из пользователей. После этого происходит перестроение сообщений, для сообщений от забаненых пользователей устанавливается состояние отображения Visibility.Collapsed. Отсылается информация на сервер. Сервер влёт уведомления забаненому пользователю.



В приложении администратора расположены два ComboBox для фильтрации спичка пользователей (группы и состояние возможной активности). При выборе конкретного пользователя администратор имеет возможность отправлять сообщение, забанить или раз банить.

Окно бана выглядит следующим образом



Групповое сообщение означает отправку сообщения на всех пользователей, присутствующих в списке.

База данных склада состоит из пяти таблиц:

1. Пользователи
   1. Идентификатор пользователя
   2. Имя пользователя
   3. Пароль
   4. Дата бана
2. Сообщения
   1. Идентификатор сообщения
   2. Текст сообщения
   3. Идентификатор группы
   4. Идентификатор пользователя
   5. Дата и время сообщения
3. Группы
   1. Идентификатор группы
   2. Название группы
4. Членство в группах
   1. Идентификатор группы
   2. Идентификатор пользователя
5. Индивидуальные баны
   1. Идентификатор первого пользователя
   2. Идентификатор второго пользователя

Таблицы связаны между собой следующими ключами:

ALTER TABLE MemeberOfGroups ADD

CONSTRAINT FK\_MemeberOfGroups\_Users FOREIGN KEY(UserID)

REFERENCES Users(UserID);

ALTER TABLE MemeberOfGroups ADD

CONSTRAINT FK\_MemeberOfGroups\_Groups FOREIGN KEY(GroupID)

REFERENCES Groups(GroupID);

ALTER TABLE Messagese ADD

CONSTRAINT FK\_Messagese\_Users FOREIGN KEY(UserID)

REFERENCES Users(UserID);

ALTER TABLE Messagese ADD

CONSTRAINT FK\_Messagese\_Groups FOREIGN KEY(GroupID)

REFERENCES Groups(GroupID);

ALTER TABLE BanList ADD

CONSTRAINT FK\_BanList\_Users FOREIGN KEY(UserID1)

REFERENCES Users(UserID);

ALTER TABLE BanList ADD

CONSTRAINT FK2\_BanList\_Users FOREIGN KEY(UserID2)

REFERENCES Users(UserID);

База заполняется следующими данными:

INSERT INTO Users (NickName, Pass, Ban) VALUES ('777', 777, '1900-01-01')

INSERT INTO Users (NickName, Pass, Ban) VALUES ('888', 888, '1900-01-01')

INSERT INTO Users (NickName, Pass, Ban) VALUES ('999', 999, '1900-01-01')

INSERT INTO Users (NickName, Pass, Ban) VALUES ('666', 666, '1900-01-01')

INSERT INTO Users (NickName, Pass, Ban) VALUES ('555', 555, '1900-01-01')

INSERT INTO Groups (GroupName) VALUES ('sport')

INSERT INTO Groups (GroupName) VALUES ('movie')

INSERT INTO Groups (GroupName) VALUES ('music')

INSERT INTO Groups (GroupName) VALUES ('education')

INSERT INTO Groups (GroupName) VALUES ('science')

INSERT INTO Groups (GroupName) VALUES ('GENERAL')

INSERT INTO MemeberOfGroups(UserID, GroupID) VALUES (1, 6)

INSERT INTO MemeberOfGroups(UserID, GroupID) VALUES (2, 6)

INSERT INTO MemeberOfGroups(UserID, GroupID) VALUES (3, 6)

INSERT INTO MemeberOfGroups(UserID, GroupID) VALUES (4, 6)

INSERT INTO MemeberOfGroups(UserID, GroupID) VALUES (5, 6)

Список использованных источников:

1. <http://msdn.microsoft.com/>
2. <http://stackoverflow.com/>
3. <http://www.cyberforum.ru/>
4. Герберт Шилдт - C# 4.0: Полное руководство