# **Звіт до модульної роботи**

з дисципліни «комп’ютерні мережі»

Виконав: студент гр.

ПМ-13-1

Кривоносов О. Д.

Перевірив:

Мащенко Л. В.

# ЗМІСТ

[**Звіт до модульної роботи** 0](#_Toc439029015)

[ЗМІСТ 1](#_Toc439029016)

[ПОСТАНОВКА ЗАДАЧІ 2](#_Toc439029017)

[ТЕОРЕТИЧНА ЧАСТИНА 2](#_Toc439029018)

[ОПИС ПРОГРАМИ ТА ДАНИХ 6](#_Toc439029019)

[ТЕСТУВАННЯ 11](#_Toc439029020)

[СПИСОК ЛІТЕРАТУРИ 16](#_Toc439029021)

[ДОДАТОК (код програми) 16](#_Toc439029022)

[Код, що описує базу даних та збереженні процедури 16](#_Toc439029023)

# ПОСТАНОВКА ЗАДАЧІ

Т е м а. Чат (програма, яка забезпечує передавання текстових повідомлень).

Ц i л ь. Ознайомитись з API сервера MYSQL, API MYSQL у c++, API WINSOCK у c++, використанням класу сокетів у .NET «System.Net.Sockets», написати обгортку (клас) над API WINSOCK у c++.

*Постановка завдання*

Реалізувати чат, який матиме такий архітектурний вигляд:  
 інтерфейс ------- сервер менеджер чату ------- сервер бази даних.

Виконаті такі умови:

1. всі повідомлення між архітектурними вузлами «интерфейс» та «сервер менеджер чату» будуються з двох частин:  
    **1** – перші два байти кожного повідомлення означають собою кількість байтів, що треба буде прийняти;

**2** – саме повідомлення, що складається з тієї кількості байтів, що було зазначено першою частиною повідомлення.  
Можна сказати, що ця вимога стосовно сеансового рівня стеку протоколів.  
Таким чином кожне повідомлення не може бути представленим більш ніж 2^16 байтами (пакет TCP/IP теж не може бути быльшим).

1. Всі повідомлення мають зберігатися на сервері бази даних.
2. Користувач може мати друзів, може подати заявку в друзі, може прийняти заявку в друзі, може переглянути, чи є інший користувач у мережі, може пересвідчитись (не перечитуючи повідомлення, чи були прочитані ім повідомлення від конкретного друга).

# ТЕОРЕТИЧНА ЧАСТИНА

Застосування СУБД для роботи з інтегрованими БД виявило особливу важливість проблеми цілісності БД. База даних може використовуватися у випадку, коли її стан відповідає стану предметної області. Ці стани називають цілісними.

Будь-яка база даних придатна до використання тільки тоді, коли її стан відповідає стану предметної області. Такі стани називають цілісними. Очевидно, що при зміні даних БД повинна переходити від одного цілісного стану до іншого. Однак, в процесі оновлення даних можливі ситуації, коли стан цілісності порушується, наприклад:

У банківській системі проводиться переведення грошових коштів з одного рахунку на інший. На мові SQL ця операція описується послідовністю двох команд UPDATE:

UPDATE accounts SET summa=summa-1000 WHERE account="PC\_1"

UPDATE accounts SET summa=summa+1000 WHERE account="PC\_2"

Як бачимо, після виконання першої команди і до завершення другої команди база даних не знаходиться в цілісному стані, тобто шукана сума списана з першого рахунку, але не зарахована на другий. Якщо в цей момент в системі відбудеться збій, наприклад, виключення електроживлення, то цілісний стан бази даних буде безповоротно втрачено.

Цілісність бази даних може порушуватися і під час обробки однієї команди SQL. Нехай виконується операція збільшення зарплати всіх співробітників фірми на 20%:

UPDATE employers SET salary=salary\*1.2

При цьому СУБД послідовно обробляє всі записи, що підлягають оновленню, тобто існує часовий інтервал, коли частина записів містить нові значення, а частина – старі.

Щоб уникнути таких ситуацій в СУБД вводиться поняття транзакції. Під транзакцією розуміється неподільна з точки зору впливу на БД послідовність операторів маніпулювання даними (читання, видалення, вставки, модифікації), що приводить до одного з двох можливих результатів: або послідовність виконується, якщо всі оператори правильні, або вся транзакція відкочується, якщо хоча б один оператор не може бути успішно виконаний. Обробка транзакцій гарантує цілісність інформації в базі даних. Таким чином, транзакція переводить базу даних з одного цілісного стану в інший.

Методом контролю за транзакціями є ведення журналу, в якому фіксуються всі зміни, що здійснюються транзакцією в БД. Якщо під час обробки транзакції відбувається збій, транзакція відкочується – з журналу встановлюється стан БД на момент початку транзакції.

У СУБД початок транзакції може задаватися явно, наприклад, командою BEGIN TRANSACTION, або передбачатися неявно, тобто чергова транзакція відкривається автоматично одразу ж після вдалого чи невдалого завершення попередньої. Для завершення транзакції зазвичай використовують команди SQL:

• COMMIT – успішно завершити транзакцію

• ROLLBACK – відкотити транзакцію, тобто повернути БД в стан, в якому вона перебувала на момент початку транзакції.

Стандарт SQL визначає, що транзакція починається з першого SQL-оператора, ініційованого користувачем або міститься в прикладній програмі. Усі наступні SQL-оператори складають тіло транзакції. Транзакція завершується одним з можливих способів:

1. оператор COMMIT означає успішне завершення транзакції, всі зміни, внесення в базу даних робляться постійними;
2. оператор ROLLBACK перериває транзакцію і скасовує всі внесені нею зміни;
3. успішне заверешення програми, яка ініціювала транзакцію, означає успішне завершення транзакції (як використання COMMIT);
4. помилкове завершення програми перериває транзакцію (як ROLLBACK).

Приклад явно заданої транзакції:

BEGIN TRANSACTION; /\*Почати транзакцію\*/

DELETE ...; /\*Зміни\*/

UPDATE ...; /\*даних\*/

if (знайдена помилка) ROLLBACK;

else COMMIT; /\*Завершити транзакцію\*/

Прилад неявно заданої транзакції:

СOMMIT; /\*Закінчення попередньої транзакції\*/

DELETE ...; /\*Зміни\*/

UPDATE ...; /\*даних\*/

if (знайдена помилка) ROLLBACK;

else COMMIT; /\*Завершити транзакцію\*/

Описаний механізм транзакцій гарантує забезпечення цілісного стану бази даних тільки в тому випадку, коли всі транзакції виконуються послідовно, тобто в кожну одиницю часу активна тільки одна транзакція. Якщо роботу з даними ведуть одночасно кілька користувачів, такий спосіб організації обробки запитів не приємний, тому що це призведе до збільшення часу реакції системи. У той же час, якщо одночасно виконуються дві транзакції, можуть виникнути такі помилкові ситуації:

• брудне читання (Dirty Read) – транзакція Т1 модифікувала якийсь елемент даних. Після цього інша транзакція Т2 прочитала вміст цього елементу даних до завершення транзакції Т1. Якщо Т1 завершується операцією ROLLBACK, це означає, що транзакція Т2 прочитала неіснуючі дані.

• неповторюване (розмите) читання (Non-repeatable or Fuzzy Read) – транзакція Т1 прочитала вміст елементу даних. Після цього інша транзакція Т2 модифікувала або видалила цей елемент. Якщо Т1 прочитає вміст цього елементу знову, то вона отримає інше значення або виявить, що елемент даних більше не існує.

• Фантом (фіктивні елементи) (Phantom) – транзакція Т1 прочитала вміст декількох елементів даних, які відповідають деякій умові. Після цього Т2 створила елемент даних, який задовольняє цій умові і стала фіксованою. Якщо Т1 повторить читання з цією ж умовою, тоді вона отримає інший набір даних.

Жодна з цих ситуацій не може виникнути при послідовному виконанні транзакцій. Звідси виникло поняття серіалізація – здатність до впорядкування паралельної обробки транзакцій. Тобто почергове (паралельне) виконання заданої множини транзакцій буде вірним, якщо при його виконанні буде отриманий той самий результат, як і при послідовному виконанні тих самих транзакцій.

Такі ситуації виникають тільки тому, що виконання транзакцій Т1 і Т2 не було впорядковано, тобто не було еквівалентно виконанню спочатку транзакції Т1, а потім Т2, або, навпаки, спочатку транзакції Т2, а потім Т1.

Примусове впорядкування транзакцій забезпечується за допомогою механізму блокувань. Суть цього механізму в наступному: якщо для виконання деякої транзакції необхідно, щоб деякий об'єкт бази даних (кортеж, набір кортежів, ставлення, набір відносин, і т.д.) не змінювався непередбачувано і без відома цієї транзакції, такий об'єкт блокується.

Основними видами блокувань є:

* **блокування з взаємним доступом**, також має назву *S-блокування* (від Shared locks) і *блокуванням із читання*.
* **монопольне блокування** (без взаємного доступу), також має назву *X-блокування (*від eXclusive locks) або *блокування із запису*. Цей режим використовується при операціях зміни, додавання та видалення об'єктів.

При цьому:

* Якщо транзакція накладає на об'єкт X-блокування, то будь-який запит іншої транзакції з блокуванням цього об'єкту буде відкинутий.
* Якщо транзакція накладає на об'єкт S-блокування, тоді:
  + запит з боку іншої транзакції з X-блокуванням на цей об'єкт буде відкинутий;
  + запит з боку іншої транзакції з S-блокуванням цього об'єкту буде прийнятий.

Доведено, що серіалізація транзакцій або, інакше, їх ізоляція забезпечується при використанні двофазного протоколу блокувань 2LP –   
Two-Phase Locks, згідно з яким всі блокування, здійснені транзакцією, знімаються тільки при її завершенні. Тобто виконання транзакції розбивається на дві фази: (1) – накопичення блокувань, (2) – звільнення блокувань в результаті фіксації.

Застосування механізму блокування призводить до уповільнення обробки транзакцій, оскільки система змушена очікувати поки звільняться дані, захоплені конкуруючої транзакцією. Вирішити цю проблему можна за рахунок зменшення фрагментів даних, які захоплені транзакцією. Залежно від захоплених об'єктів розрізняють кілька рівнів блокування:

* блокування усієї бази даних;
* блокування окремих таблиць;
* блокування сторінок;
* блокування записів;
* блокування окремих полів.

Сучасні СУБД, можуть здійснювати блокування на рівні записів або сторінок.

Мова SQL також надає спосіб непрямого управління швидкістю виконання транзакцій за допомогою зазначення рівня ізоляції транзакції. Під рівнем ізоляції транзакції розуміють можливість виникнення однієї з описаних вище помилкових ситуацій. У стандарті SQL визначено 4 рівня ізоляції:

*Таблиця 1*

Рівні ізоляції

|  |  |  |  |
| --- | --- | --- | --- |
| **Рівень ізоляції** | **Брудне читання** | **Розмите читання** | **Фантом** |
| Незафіксоване читання (READ UNCOMMITTED) | можливо | можливо | можливо |
| Зафіксоване читання (READ COMMITED) | можливо | можливо | можливо |
| Повторюване читання (REPEATABLE READ) | можливо | неможливо | можливо |
| Серіалізація (SERIALIZABLE) | можливо | неможливо | неможливо |

Для визначення характеристик транзакції використовується оператор

**SET TRANSACTION <режим доступу>, <рівень ізоляції>**

Список рівнів ізоляції наведено в табл. 1. Режим доступу за замовчуванням використовується READ WRITE (читання запис), якщо заданий рівень ізоляції READ UNCOMMITED, то режим доступу повинен бути READ ONLY (тільки читання).

Одним з найбільш суттєвих недоліків методу серіалізації транзакцій на основі механізму блокувань є можливість виникнення глухих кутів (тупиків) (dead locks) між транзакціями. Наприклад, транзакція Т1 наклала монопольне блокування на об'єкт О1 і претендує на доступ до об'єкту О2, який вже монопольно заблокований транзакцією Т2, яка чекає доступу до об'єкта О1. У цьому випадку жодна з транзакцій тривати не може, отже, блокування об'єктів О1 і О2 ніколи не будуть зняті. Природного виходу з такої ситуації не існує, тому тупикові ситуації виявляються і усуваються штучно.

# ОПИС ПРОГРАМИ ТА ДАНИХ

Програма має такий архітектурний вигляд:  
 інтерфейс ------- сервер менеджер чату ------- сервер бази даннх.

Будемо описувати від серверів до інтерфейсу.

**Сервер бази данних:**

Є база даних **my\_chat**.

Основою бази даних е таблиці:

1. my\_chat.users
2. my\_chat.r\_c\_m

Коментарі  
«   
-- список ссылок на сообщения (такая таблица нужна, чтоб знать, каки сообщения из списка сообщений доступны конкретному пользователю  
-- эта таблица не создается для приватных переписок). имя такой таблицы для каждого пользователя задается так:  
-- CONCAT(id этого рользователя, '\_', имя таблицы с сообщениями)  
»

1. my\_chat.private\_chats  
   Коментарі  
   «  
   -- список личных переписок

-- имя такой таблицы для каждого пользователя задается так:

-- CONCAT('p', some id from users)  
»

1. my\_chat.conferences

Коментарі  
«  
-- список конференций

-- имя такой таблицы для каждого пользователя задается так:

-- CONCAT('с', some id from users)  
»

1. my\_chat.requests

Коментарі  
«  
-- список запросов в друзья

-- имя такой таблицы для каждого пользователя задается так:

-- CONCAT('r', some id from users)

»

1. my\_chat.chat\_m\_g\_d

Коментарі  
«  
-- участники чата

-- поле acs принимает 4 значения (NULL, '', 'g', 'r')

-- NULL - доступ закрыт

-- '' - обычный пользователь

-- 'g' - пользователь, который может добавлять в чат других пользователей и менять имя чата

-- 'r' - один пользователь, который назначать предыдущие уравни доступа всем пользователям и может добавлять в чат других пользователей

-- имя такой таблицы для каждого чата задается так:

-- CONCAT('part', имя таблиции сообщений, соответствующее этому списку участников переписки)

»

1. my\_chat.chat\_m\_u\_d

Коментарі  
«

-- chat\_members\_user\_default

»

1. my\_chat.chat\_name

Коментарі  
«

-- хранит имя чата (всего одна строка)

-- имя такой таблицы для каждого чата задается так:

-- CONCAT('name', имя таблиции сообщений)  
»

1. my\_chat.messages

Коментарі  
«

-- сообщения

-- имя такой таблицы для каждого чата задается, как индекс пользователя, что создал этот чат, и номер созданного им чата, что хранится в my\_chat.users.n\_created\_chats  
»

Далі маються такі збережені процедури

1. AUT відповідає за авторизацію користувача чату
2. GET\_USER\_ID
3. ADDUSER
4. GET\_FRIEND\_ID
5. GET\_FRIEND\_LOG
6. SEND\_P\_M відіслати приватне повідомлення
7. SEND\_C\_M відіслати повідомлення у конференцію
8. SEND\_REQUEST відіслати запит на прийняття в друзі
9. ACCEPT\_REQUEST прийняти запит на прийняття в друзі
10. CREATE\_CHAT
11. IS\_FRIEND\_IN\_CHAT
12. ADD\_PART додати користувача у конференцію
13. SET\_ACCESS задати рівень доступу
14. ONLINE
15. GNM\_PART\_F\_CHAT взяти M членів конференції починаючи з N-го
16. GNM\_REQUESTS
17. GNM\_MESSAGES\_P
18. GNM\_MESSAGES\_С
19. GNM\_CHATS
20. GET\_LAST\_USER\_TICK
21. GET\_USER\_DATA
22. GNM\_USERS
23. GET\_UNREAD\_LAST\_TICK;

**Сервер менеджер чату:**

Є реалізація обгортки для API WINSOCK яку можна знайти в додатку.

Є основні команди:

1. "check"

Коментарі  
«  
перевірити правильність комбінації логін пароль  
»

1. "smp"  
   Коментарі  
   «  
   надислати приватне повідомлення  
   »
2. "smc"  
   Коментарі  
   «  
   надислати повідомлення до конференції  
   »
3. "sr"  
   Коментарі  
   «  
   надіслати заявку в друзі  
   »
4. "ar"
5. "cc"  
   Коментарі  
   «  
   створити переписку  
   »
6. "ap"  
   Коментарі  
   «  
   додати користувача  
   »
7. "line"  
   Коментарі  
   «  
   сповістити про че, що користувач онлайн  
   »
8. "sa"  
   Коментарі  
   «  
   задати рівень доступу  
   »
9. "gud"  
   Коментарі  
   «  
   взяти данні користувача  
   »
10. "glut"  
    Коментарі  
    «  
    взяти останній тік користувача  
    »
11. "gnmp"  
    Коментарі  
    «  
    geter\_n\_m\_partners\_from  
    »
12. "gnmr"  
    Коментарі  
    «  
    geter\_n\_m\_requests  
    »
13. "gnmmp"  
    Коментарі  
    «  
    geter\_n\_m\_messages\_p  
    »
14. "gnmc"  
    Коментарі  
    «  
    geter\_n\_m\_messages\_c  
    »
15. "gnmf"  
    Коментарі  
    «  
    geter\_n\_m\_friens  
    »
16. "gnmco"  
    Коментарі  
    «  
    geter\_n\_m\_confs  
    »
17. "gnmu"  
    Коментарі  
    «  
    geter\_n\_m\_users  
    »
18. "gful"  
    Коментарі  
    «  
    geter\_freand\_unread\_last\_tick  
    »
19. "gcul"  
    Коментарі  
    «  
    geter\_conf\_unread\_last\_tick  
    »
20. "gfid"  
    Коментарі  
    «  
    get\_friend\_id  
    »

**Інтерфейс:**

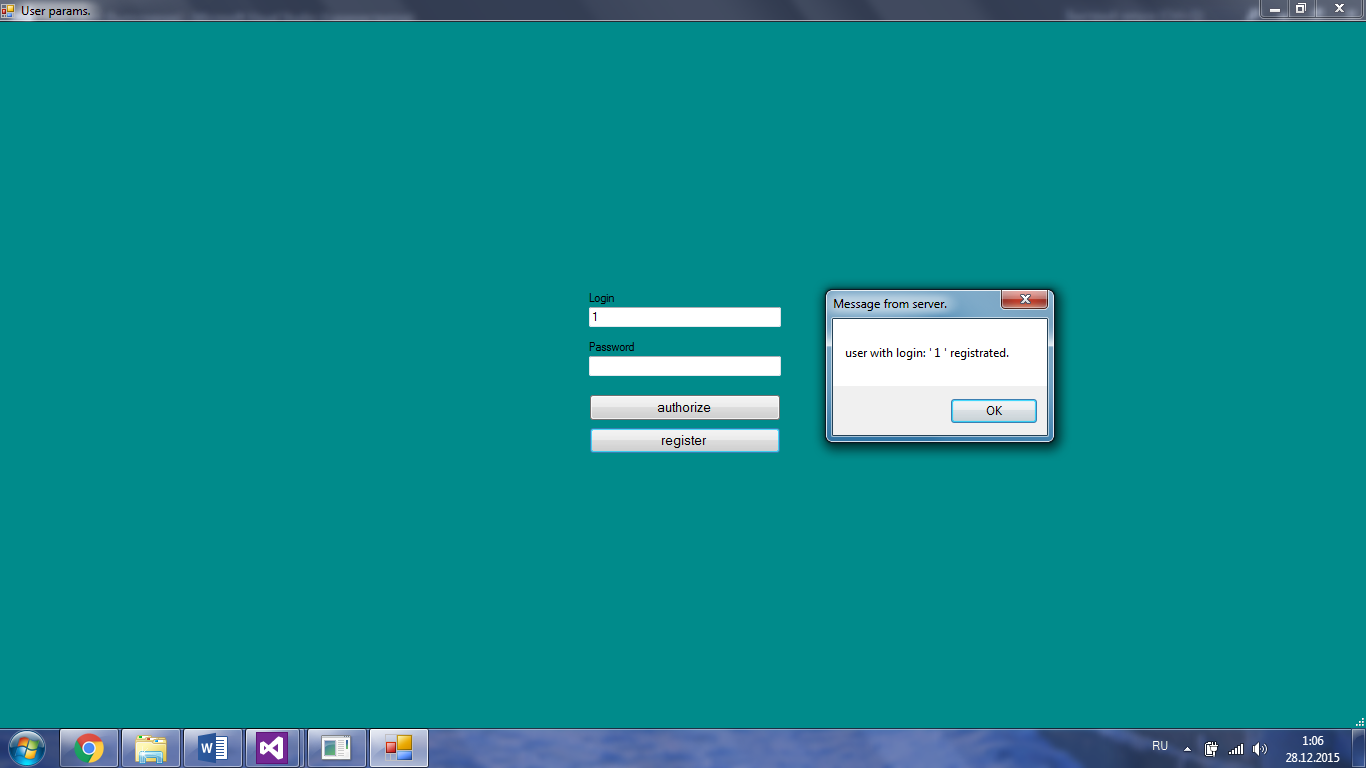
Є такі класи:

1. Button\_more – відповідає за кнопку, що означає «завантажити більше»
2. Central – відповідає за всі основні процеси інтерфейсу
3. Conference – відповідає за контрол, що для користувача є інтерфейсом до конференції.
4. Conference\_fields – відповідає за управління даними стосовно конкретної конфереції.
5. Form1 – основна форма.
6. Form2 – форма авторизації та регистрації
7. Form3 – форма інформації
8. ItemList\_with\_Button\_more – контрол, для колекціонування інших контролів; вміщає у себе контрол Button\_more.
9. Menager\_for\_conference – контрол, що є комплексом контролів для керування.
10. Message – контрол повідомлення
11. User – відповідає за контрол, що для користувача є інтерфейсом до конференції.
12. User\_fields – відповідає за управління даними стосовно конкретної конфереції.
13. Waiter – клас для управління чергою запитів.

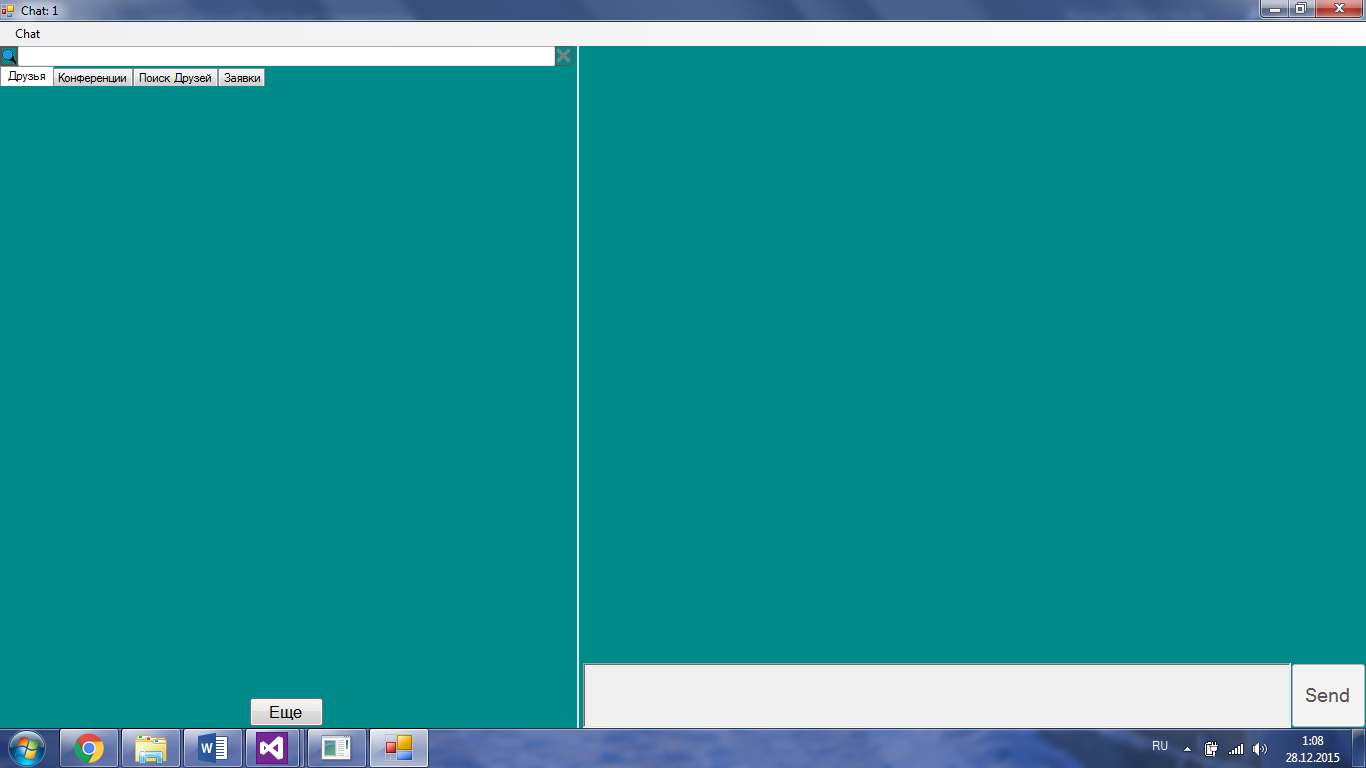
# ТЕСТУВАННЯ



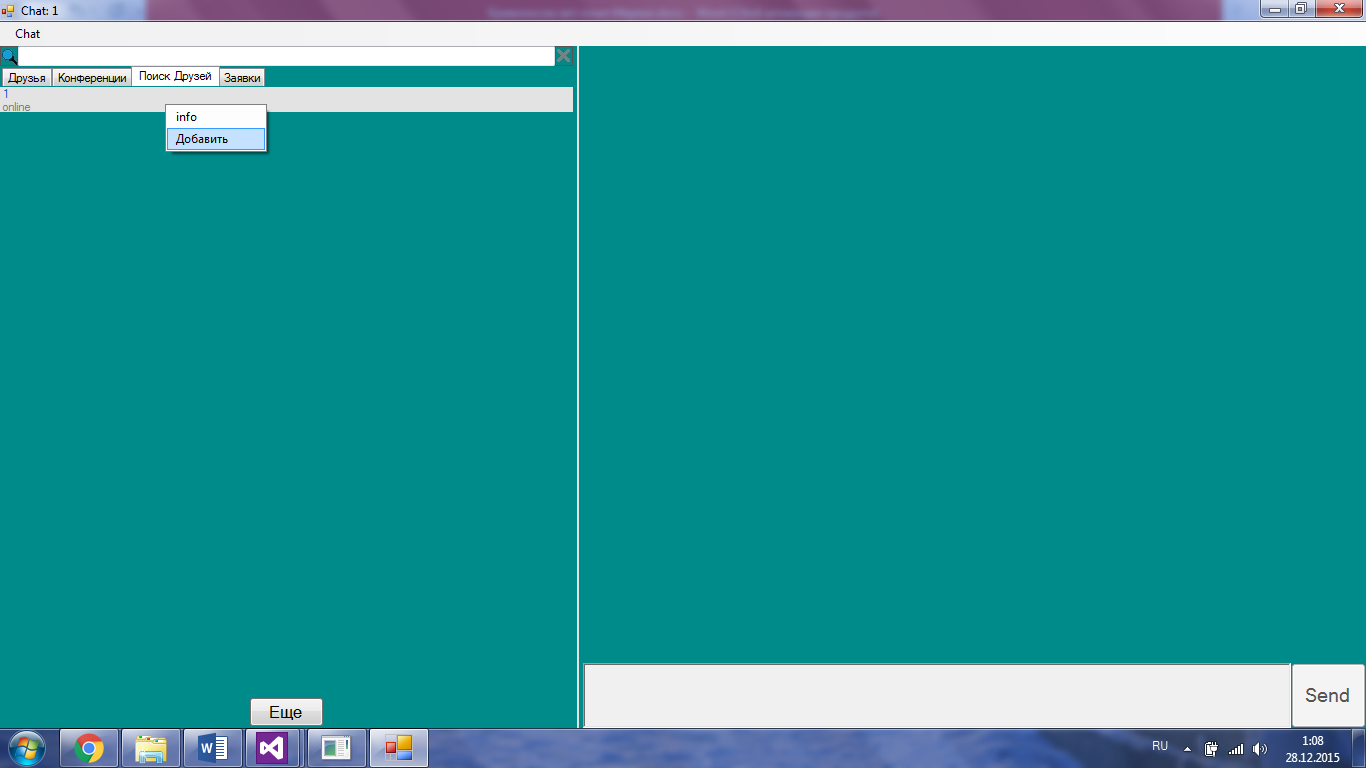
(Рис.1) початок роботи.



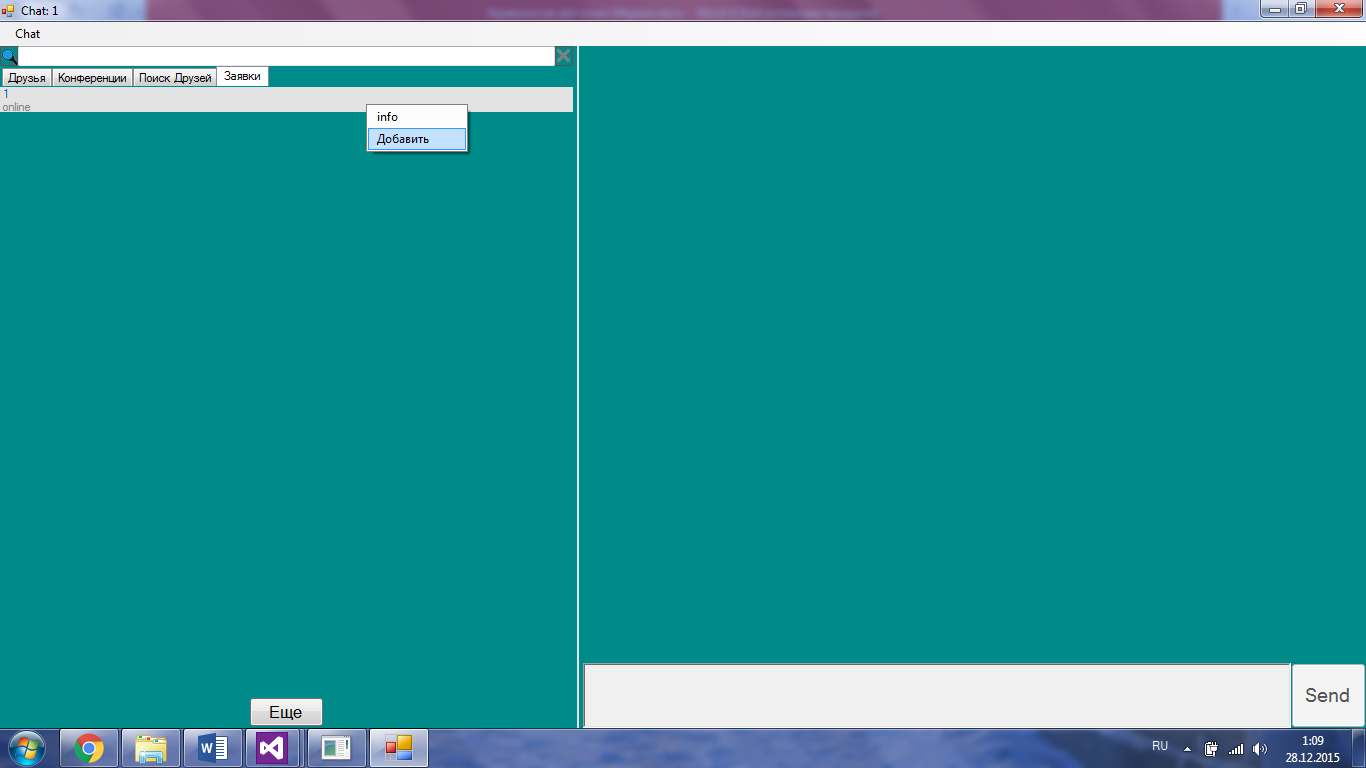
(Рис.2) користувач зареєструвався під логіном «1»



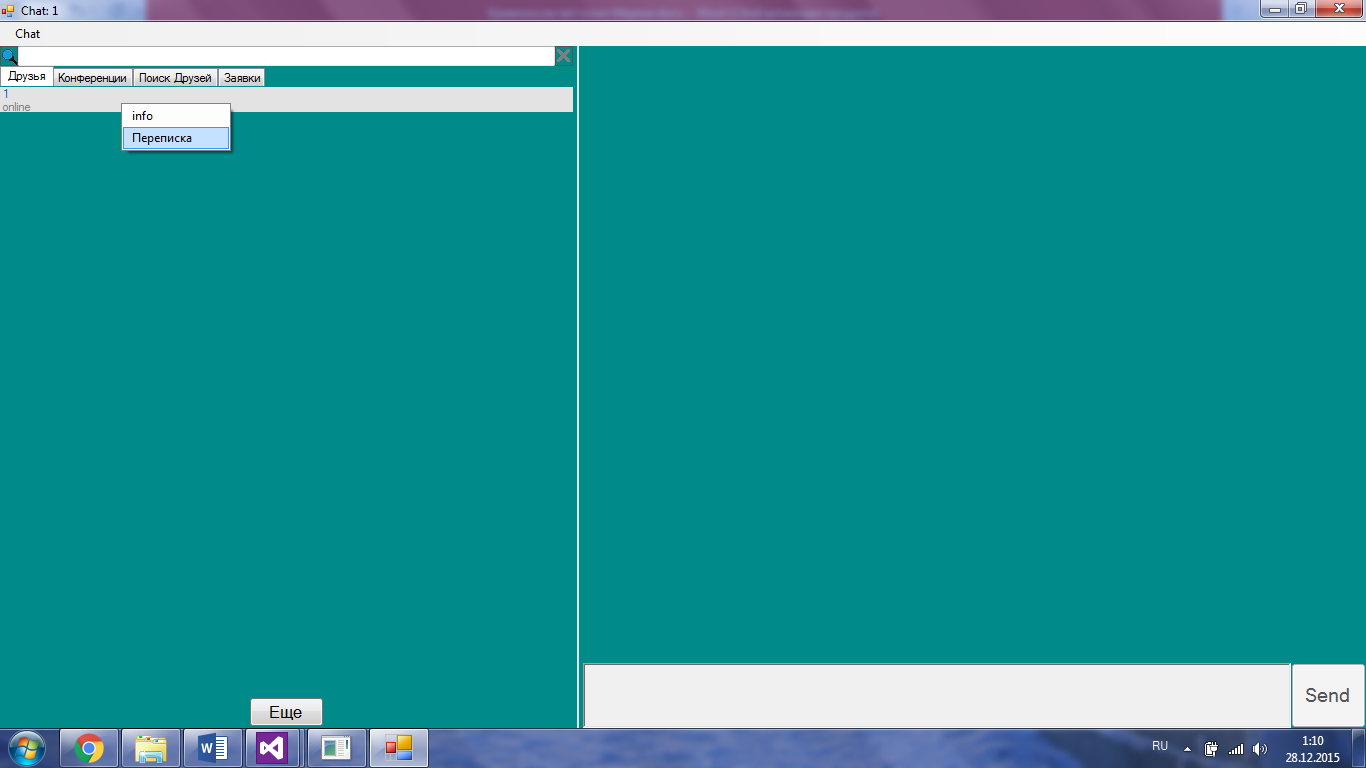
(Рис.3) користувач авторизувався



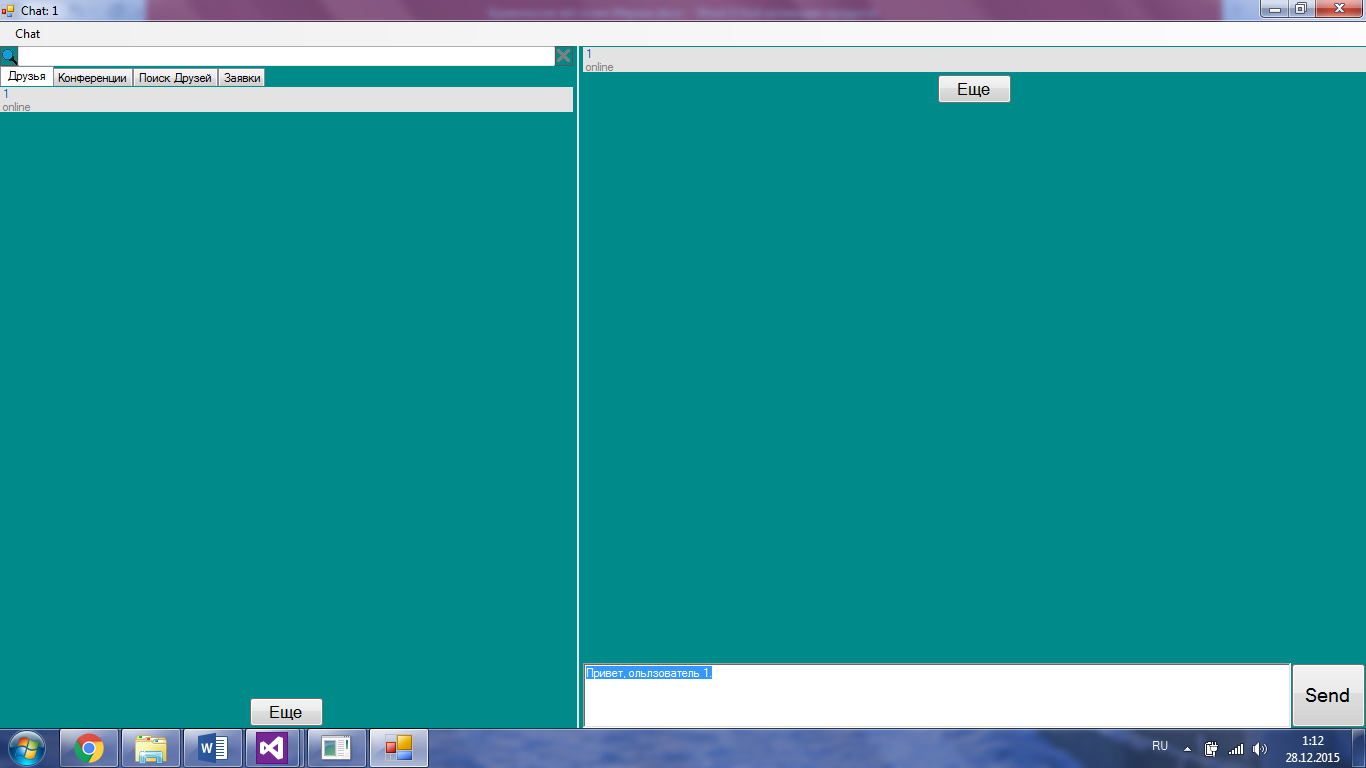
(Рис.4) користувач перейшов до пошуку друзів і хоче собі додати себе у друзі.



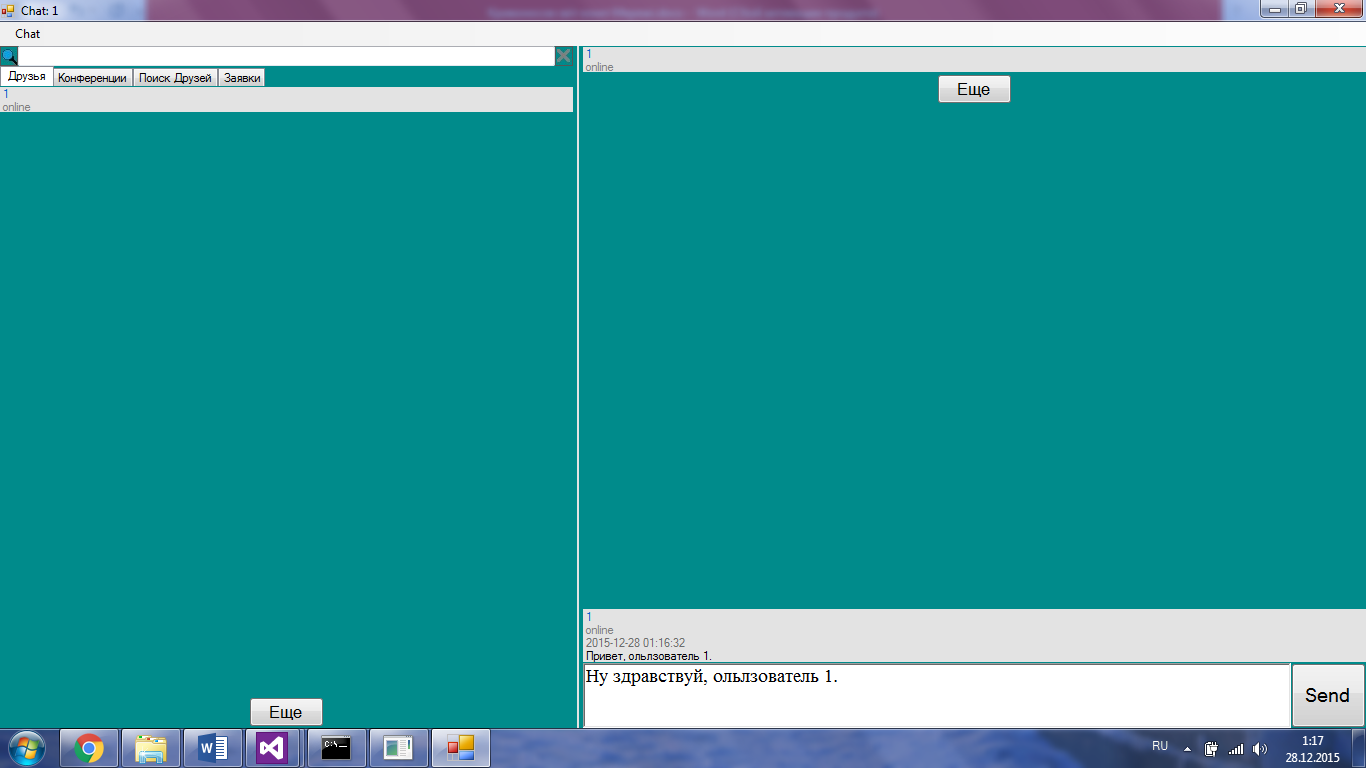
(Рис.5) користувач послав собі заявку в друзі, і тепер хоче прийняти цю заявку.



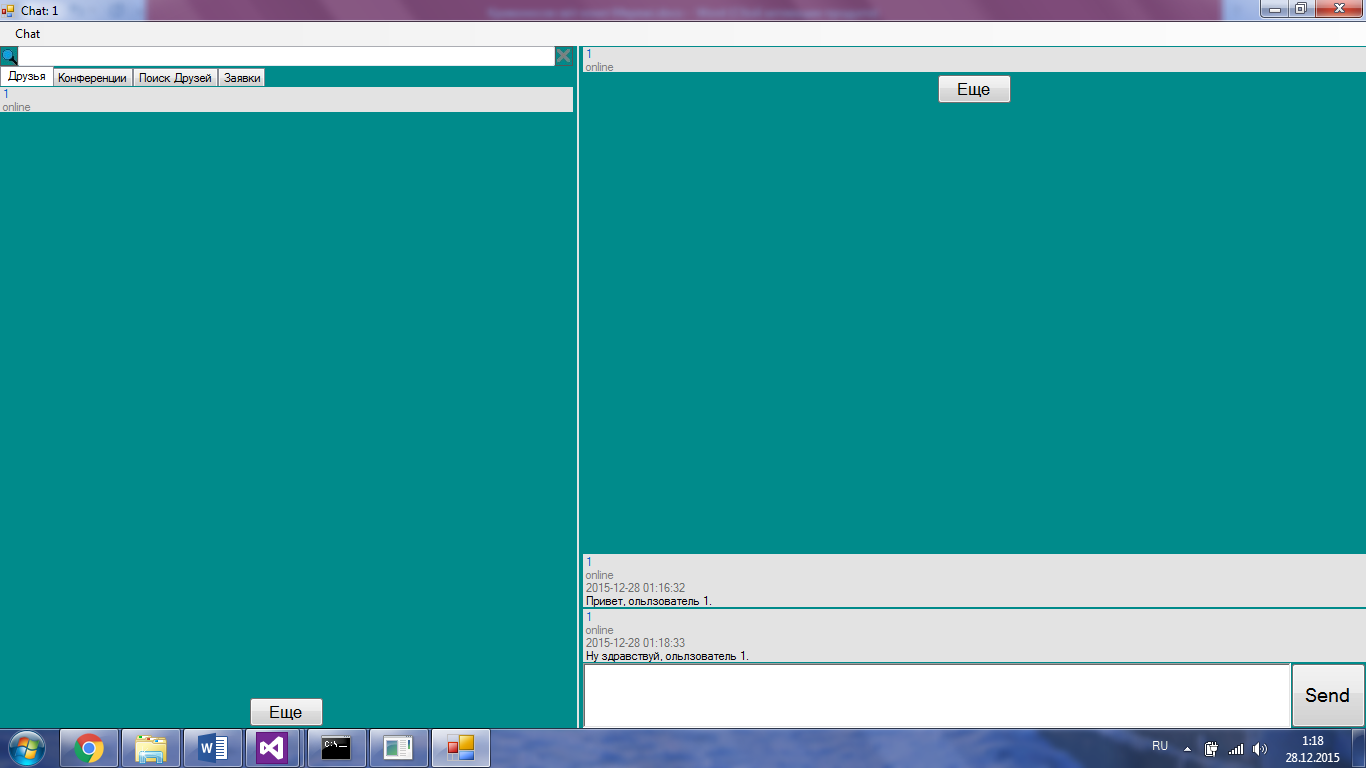
(Рис.6) користувач прийняв заявку і тепер хоте перейти до переписки.



(Рис.7) перейшовши до чату, користувач хоче надіслати собі повідомлення «Привет, ольлзователь 1.»



(Рис.8) користувач відправив собі повідомлення і хоче відповісти собі «Ну здравствуй, ольлзователь 1.»



(Рис.9)

# СПИСОК ЛІТЕРАТУРИ

1. [API MYSQL C++](https://dev.mysql.com/doc/refman/5.7/en/connector-cpp-info.html)
2. «MYSQL\_Кузнецов М.В., Симдянов И.В. - MySQL 5 (в подлиннике) – 2010»
3. «У. Р. Стивенс - UNIX. Разработка сетевых приложений, 3-е изд. - 2007»

# ДОДАТОК (код програми)

## Код, що описує базу даних та збереженні процедури

drop database my\_chat;

create database if not exists my\_chat;

USE my\_chat;

-- comment

#comment

/\*comment\*/

-- --------------------------------------------------------------------------------------

create table if not exists my\_chat.users(

id SERIAL PRIMARY KEY, -- SERIAL = BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT UNIQUE

log VARCHAR(40) NOT NULL UNIQUE, -- UNIQUE - не повторяестя значение этого поля в каждой строке

pass VARCHAR(40) NOT NULL,

status VARCHAR(255) NOT NULL DEFAULT '', -- статус

last\_tick TIMESTAMP NOT NULL, -- если не задать значение, то в переменной будет храниться время изменения (добавления) записи в диапазоне от «1970-01-01 00:00:00» до некоторой даты в 2038 г, ДЛЯ ОБНОВЛЕНИЯ ВРЕМЕНИ ИСПОЛЬЗУЙ ФУНКЦИЮ NOW() или CURRENT\_TIMESTAMP //insert into dt1 values(now());

unread\_private\_chats BOOL NOT NULL DEFAULT FALSE, -- пользоваетль проверяет, есть ли что-то непрочитанное (значение меняется автоматически)

unread\_chats BOOL NOT NULL DEFAULT FALSE,

request BOOL NOT NULL DEFAULT FALSE,

n\_unread\_private\_chats BIGINT UNSIGNED NOT NULL DEFAULT 0, -- количество непросмотренных записей

n\_unread\_chats BIGINT UNSIGNED NOT NULL DEFAULT 0,

n\_request BIGINT UNSIGNED NOT NULL DEFAULT 0,

n\_created\_chats BIGINT UNSIGNED NOT NULL DEFAULT 0, -- количество созданых чатов.

-- name VARCHAR(20),

INDEX(log)

-- ,INDEX(name)

)ENGINE InnoDB CHARACTER SET utf8;

-- --------------------------------------------------------------------------------------

-- --------------------------------------------------------------------------------------

-- список ссылок на сообщения (такая таблица нужна, чтоб знать, каки сообщения из списка сообщений доступны конкретному пользователю. эта таблица не создается для приватных переписок)

-- имя такой таблицы для каждого пользователя задается так:

-- CONCAT(id этого рользователя, '\_', имя таблицы с сообщениями)

CREATE TABLE if not exists my\_chat.r\_c\_m(-- ref\_chats\_messages

id SERIAL PRIMARY KEY,

id\_m BIGINT UNSIGNED NOT NULL -- указывет на сообщение (не надо делать внешний ключ потому, что невозможно подобрать таблицу, по которой этот ключ будет строиться (значения в этом поля будут из таблицы с неизвестным именем))

)ENGINE InnoDB CHARACTER SET utf8;

-- --------------------------------------------------------------------------------------

-- --------------------------------------------------------------------------------------

-- список личных переписок

-- имя такой таблицы для каждого пользователя задается так:

-- CONCAT('p', some id from users)

CREATE TABLE if not exists my\_chat.private\_chats(

id SERIAL PRIMARY KEY,

name\_of\_messages\_table VARCHAR(64) NOT NULL UNIQUE,

unread BOOL NOT NULL DEFAULT FALSE,

last\_tick TIMESTAMP NOT NULL -- чтоб сортировать потом по этому полю

)ENGINE InnoDB CHARACTER SET utf8;

-- список конференций

-- имя такой таблицы для каждого пользователя задается так:

-- CONCAT('с', some id from users)

CREATE TABLE if not exists my\_chat.conferences

LIKE my\_chat.private\_chats;

-- список запросов в друзья

-- имя такой таблицы для каждого пользователя задается так:

-- CONCAT('r', some id from users)

CREATE TABLE if not exists my\_chat.requests(

user\_id BIGINT UNSIGNED NOT NULL UNIQUE,

message TEXT NOT NULL,

last\_tick TIMESTAMP NOT NULL,

FOREIGN KEY(user\_id) REFERENCES my\_chat.users (id)

ON UPDATE CASCADE

ON DELETE RESTRICT

)ENGINE InnoDB CHARACTER SET utf8;

-- --------------------------------------------------------------------------------------

-- --------------------------------------------------------------------------------------

-- участники чата

-- поле acs принимает 4 значения (NULL, '', 'g', 'r')

-- NULL - доступ закрыт

-- '' - обычный пользователь

-- 'g' - пользователь, который может добавлять в чат других пользователей и менять имя чата

-- 'r' - один пользователь, который назначать предыдущие уравни доступа всем пользователям и может добавлять в чат других пользователей

-- имя такой таблицы для каждого чата задается так:

-- CONCAT('part', имя таблиции сообщений, соответствующее этому списку участников переписки)

CREATE TABLE if not exists my\_chat.chat\_m\_g\_d( -- chat\_members\_granting\_default

user\_id BIGINT UNSIGNED NOT NULL UNIQUE,

acs ENUM('g','r','') DEFAULT 'g', -- access

FOREIGN KEY(user\_id) REFERENCES my\_chat.users (id) ON UPDATE CASCADE ON DELETE RESTRICT

)ENGINE InnoDB CHARACTER SET utf8;

CREATE TABLE if not exists my\_chat.chat\_m\_u\_d -- chat\_members\_user\_default

LIKE my\_chat.chat\_m\_g\_d;

ALTER TABLE my\_chat.chat\_m\_u\_d ALTER acs SET DEFAULT '';

-- хранит имя чата (всего одна строка)

-- имя такой таблицы для каждого чата задается так:

-- CONCAT('name', имя таблиции сообщений)

CREATE TABLE if not exists my\_chat.chat\_name(

name VARCHAR(64) NOT NULL

)ENGINE InnoDB CHARACTER SET utf8;

-- сообщения

-- имя такой таблицы для каждого чата задается, как индекс пользователя, что создал этот чат, и номер созданного им чата, что хранится в my\_chat.users.n\_created\_chats

create table if not exists my\_chat.messages(

id SERIAL PRIMARY KEY,

user\_id BIGINT UNSIGNED, -- если в этом поле будет NULL - это сообщение от сервера (например сообщение о том, что кто-то из пользователей добавил в чат еще одного пользователя)

message TEXT NOT NULL,

last\_tick TIMESTAMP NOT NULL,

FOREIGN KEY(user\_id) REFERENCES my\_chat.users (id) ON UPDATE CASCADE ON DELETE RESTRICT

);

-- --------------------------------------------------------------------------------------

-- --------------------------------------------------------------------------------------

drop user chat;

CREATE USER chat

IDENTIFIED BY 'chat';

GRANT ALL ON my\_chat.\* TO 'chat';

/\*

show grants for chat\_server;

тут я хотел запретить пользоваетлю chat\_server смотреть на пароли и удалять строки в my\_chat.users,

но пишет, что не было назначено GRANT для (SELECT(pass) ON my\_chat.users) и (DELETE ON my\_chat.users) и пожтому отменять нечего...

REVOKE SELECT(pass)

ON my\_chat.users

FROM 'chat\_server';

REVOKE DELETE

ON my\_chat.users

FROM chat\_server;

\*/

-- --------------------------------------------------------------------------------------

-- --------------------------------------------------------------------------------------

DELIMITER //

DROP PROCEDURE IF EXISTS GNMF;

CREATE PROCEDURE GNMF(n BIGINT UNSIGNED, m BIGINT UNSIGNED, tabl VARCHAR(64)) -- Get N M From (взять m начиная с n из таблици где есть поле id)

BEGIN

START TRANSACTION;

SET @var = CONCAT('SELECT id FROM ', tabl, ' ORDER BY id DESC LIMIT ', n, ', ', m); -- LIMIT <номер строки выборки с которого начинать, количество строк>;

PREPARE query FROM @var; -- формировка запроса query

EXECUTE query; -- выполнение запроса query

COMMIT;

END//

DROP PROCEDURE IF EXISTS AUT;

CREATE PROCEDURE AUT(login VARCHAR(40), password VARCHAR(40)) -- authorize

BEGIN

START TRANSACTION;

SELECT id FROM my\_chat.users WHERE log = login AND pass = password;

COMMIT;

END//

DROP PROCEDURE IF EXISTS GET\_USER\_ID;

CREATE PROCEDURE GET\_USER\_ID(login VARCHAR(40))

BEGIN

START TRANSACTION;

SELECT id FROM my\_chat.users WHERE log = login;

COMMIT;

END//

DROP PROCEDURE IF EXISTS ADDUSER;

CREATE PROCEDURE ADDUSER(login VARCHAR(40), password VARCHAR(40))

BEGIN

START TRANSACTION;

insert into my\_chat.users (log, pass) values(login, password);

SET @var := CONCAT('CREATE TABLE if not exists my\_chat.p', (SELECT @id\_ := id FROM my\_chat.users WHERE log = login), ' LIKE my\_chat.private\_chats');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('CREATE TABLE if not exists my\_chat.c', @id\_, ' LIKE my\_chat.conferences');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('CREATE TABLE if not exists my\_chat.r', @id\_, ' LIKE my\_chat.requests');

PREPARE query FROM @var; EXECUTE query;

COMMIT;

END//

DROP PROCEDURE IF EXISTS GET\_FRIEND\_ID;

CREATE PROCEDURE GET\_FRIEND\_ID(login VARCHAR(40))

BEGIN

SET @user\_id := (SELECT id FROM my\_chat.users WHERE log = login);

IF (@user\_id > 0) THEN

START TRANSACTION;

SET @var := CONCAT('CREATE TEMPORARY TABLE addressees SELECT LALKA.id, LALKA.name\_of\_messages\_table FROM my\_chat.p', @userID, ' AS LALKA');

PREPARE query FROM @var; EXECUTE query;

ALTER TABLE addressees ADD COLUMN num SERIAL;

ALTER TABLE addressees ADD INDEX (num);

SET @i := 1;

SET @max\_ := (SELECT MAX(num) FROM addressees);

SET @res := 0;

IF (@user\_id = @userID) THEN

WHILE (@i <= @max\_ AND @res = 0) DO

SET @var := CONCAT('SET @res := (SELECT id FROM addressees WHERE num = @i AND (1 = (SELECT COUNT(user\_id) FROM my\_chat.part', (SELECT name\_of\_messages\_table FROM addressees WHERE num = @i), ')))');

PREPARE query FROM @var; EXECUTE query;

IF (@res > 0) THEN

SET @tmp := 0;

ELSE

SET @res := 0;

END IF;

SET @i := @i + 1;

END WHILE;

ELSE

WHILE (@i <= @max\_ AND @res = 0) DO

SET @var := CONCAT('SET @res := (SELECT id FROM addressees WHERE num = @i AND (@user\_id IN (SELECT user\_id FROM my\_chat.part', (SELECT name\_of\_messages\_table FROM addressees WHERE num = @i), ')))');

PREPARE query FROM @var; EXECUTE query;

IF (@res > 0) THEN

SET @tmp := 0;

ELSE

SET @res := 0;

END IF;

SET @i := @i + 1;

END WHILE;

END IF;

ALTER TABLE addressees DROP COLUMN num;

ALTER TABLE addressees DROP COLUMN name\_of\_messages\_table;

SELECT @res;

DROP TABLE addressees;

COMMIT;

END IF;

END//

/\*

-- переделать

DROP PROCEDURE IF EXISTS GET\_FRIEND\_ID\_FUNC;

CREATE PROCEDURE GET\_FRIEND\_ID\_FUNC(login VARCHAR(40))

BEGIN

START TRANSACTION;

-- переделать

SET @var = CONCAT('SET @res\_GET\_FRIEND\_ID\_FUNC = (SELECT id FROM my\_chat.p', @userID, ' WHERE (SELECT id FROM my\_chat.users WHERE log = ', login,' ) IN (SELECT user\_id FROM CONCAT(''part'', name\_of\_messages\_table)))');

PREPARE query FROM @var; EXECUTE query;

COMMIT;

END//

\*/

DROP PROCEDURE IF EXISTS GET\_FRIEND\_LOG;

CREATE PROCEDURE GET\_FRIEND\_LOG(friendID BIGINT UNSIGNED)

BEGIN

START TRANSACTION;

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.p', @userID,' WHERE id = ', friendID, ')');

PREPARE query FROM @var; EXECUTE query;

SET @var = CONCAT('SELECT log FROM my\_chat.users WHERE id IN (SELECT user\_id FROM my\_chat.part', @name, ') AND id != @userID');

PREPARE query FROM @var; EXECUTE query;

COMMIT;

END//

DROP PROCEDURE IF EXISTS SEND\_P\_M;

CREATE PROCEDURE SEND\_P\_M(freindID BIGINT UNSIGNED, message\_ TEXT)

BEGIN

START TRANSACTION;

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.p', @userID, ' WHERE id = ', freindID, ')');

PREPARE query FROM @var; EXECUTE query;

IF (@name != 'wwe') THEN -- wwe просто так. важно что name не NULL

-- отсылаем собщение

SET @var := CONCAT('insert into my\_chat.', @name, ' (user\_id, message) values(@userID, ''', message\_,''')');

PREPARE query FROM @var; EXECUTE query;

-- SET @var := CONCAT('SET @id\_m\_ := (SELECT MAX(id) FROM my\_chat.', @name, ')');

-- PREPARE query FROM @var; EXECUTE query;

-- говорим об этом всем участникам

SET @var := CONCAT('CREATE TEMPORARY TABLE addressees SELECT user\_id FROM my\_chat.part', @name);

PREPARE query FROM @var; EXECUTE query;

ALTER TABLE addressees ADD COLUMN id SERIAL PRIMARY KEY;

SET @i := 1;

SET @max\_ := (SELECT MAX(id) FROM addressees);

WHILE @i <= @max\_ DO

SET @id\_ := (SELECT user\_id FROM addressees WHERE id = @i);

SET @var := CONCAT('SET @unread\_ := (SELECT unread FROM my\_chat.p', @id\_, ' WHERE id = ', freindID, ')');

PREPARE query FROM @var; EXECUTE query;

IF (@unread\_ = FALSE) THEN

UPDATE my\_chat.users SET unread\_private\_chats := TRUE, n\_unread\_private\_chats := n\_unread\_private\_chats + 1 WHERE id = @id\_;

SET @var := CONCAT('UPDATE my\_chat.p', @id\_, ' SET unread := TRUE WHERE id = ', freindID);

PREPARE query FROM @var; EXECUTE query;

ELSE

UPDATE my\_chat.users SET unread\_private\_chats := TRUE WHERE id = @id\_;

END IF;

SET @i := @i + 1;

END WHILE;

DROP TABLE addressees;

ELSE

SELECT 1;

END IF;

COMMIT;

END//

DROP PROCEDURE IF EXISTS SEND\_C\_M;

CREATE PROCEDURE SEND\_C\_M(chatID BIGINT UNSIGNED, message\_ TEXT)

BEGIN

START TRANSACTION;

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.c', @userID, ' WHERE id = ', chatID,')');

PREPARE query FROM @var; EXECUTE query;

SET @chek := NULL;

IF (@name != NULL) THEN

SET @var = CONCAT('SET @chek := (SELECT acs FROM my\_chat.part', @name, ' WHERE user\_id = @userID)');

PREPARE query FROM @var; EXECUTE query;

END IF;

IF (@chek != 'wwe') THEN -- wwe просто так. важно что chek не NULL

-- отсылаем собщение

SET @var := CONCAT('insert into my\_chat.', @name, ' (user\_id, message) values(@userID, ''', message\_,''')');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('SET @id\_m\_ := (SELECT MAX(id) FROM my\_chat.', @name, ')');

PREPARE query FROM @var; EXECUTE query;

-- говорим об этом всем участникам

SET @var := CONCAT('CREATE TEMPORARY TABLE addressees SELECT user\_id FROM my\_chat.part', @name, ' WHERE acs != NULL');

PREPARE query FROM @var; EXECUTE query;

ALTER TABLE addressees ADD COLUMN id SERIAL PRIMARY KEY;

SET @i := 1;

SET @max\_ := (SELECT MAX(id) FROM addressees);

WHILE @i <= @max\_ DO

SET @id\_ := (SELECT user\_id FROM addressees WHERE id = @i);

--

SET @var := CONCAT('SET @unread\_ := (SELECT unread FROM my\_chat.c', @id\_, ' WHERE id = ', chatID, ')');

PREPARE query FROM @var; EXECUTE query;

IF (@unread\_ = FALSE) THEN

UPDATE my\_chat.users SET unread\_chats := TRUE, n\_unread\_chats := n\_unread\_chats + 1 WHERE id = @id\_;

SET @var := CONCAT('UPDATE my\_chat.c', @id\_, ' SET unread = TRUE WHERE id = ', chatID);

PREPARE query FROM @var; EXECUTE query;

ELSE

UPDATE my\_chat.users SET unread\_chats := TRUE WHERE id = @id\_;

END IF;

--

SET @var := CONCAT('insert into my\_chat.', @id\_, '\_', @name, ' (id\_m) values(@id\_m\_)');

PREPARE query FROM @var; EXECUTE query;

--

SET @i := @i + 1;

END WHILE;

DROP TABLE addressees;

ELSE

SELECT 1;

END IF;

COMMIT;

END//

DROP PROCEDURE IF EXISTS SEND\_REQUEST;

CREATE PROCEDURE SEND\_REQUEST(login VARCHAR(40), message\_ TEXT)

BEGIN

START TRANSACTION;

-- делаем таблицу реальных id друзей

SET @var := CONCAT('CREATE TEMPORARY TABLE addressees SELECT LALKA.name\_of\_messages\_table FROM my\_chat.p', @userID, ' AS LALKA');

PREPARE query FROM @var; EXECUTE query;

ALTER TABLE addressees ADD COLUMN num SERIAL;

ALTER TABLE addressees ADD INDEX (num);

ALTER TABLE addressees ADD COLUMN user\_id BIGINT;

SET @i := 1;

SET @max\_ := (SELECT MAX(num) FROM addressees);

WHILE @i <= @max\_ DO

SET @tmp\_name := (SELECT name\_of\_messages\_table FROM addressees WHERE num = @i);

SET @var := CONCAT('UPDATE addressees SET user\_id := (SELECT user\_id FROM my\_chat.part', @tmp\_name, ' WHERE (user\_id != @userID OR (1 = (SELECT COUNT(user\_id) FROM my\_chat.part', @tmp\_name, ')))) WHERE num = @i');

PREPARE query FROM @var; EXECUTE query;

SET @i := @i + 1;

END WHILE;

ALTER TABLE addressees DROP COLUMN num;

ALTER TABLE addressees DROP COLUMN name\_of\_messages\_table;

-- закончили

-- проверка, что login - не друг

-- SET @id\_ := NULL;

SET @id\_ := (SELECT id FROM my\_chat.users WHERE log = login AND id IN (SELECT user\_id FROM addressees));

IF (@id\_ > 0) THEN

SELECT 'NULL';

ELSE

SET @id\_ := (SELECT id FROM my\_chat.users WHERE log = login);

SET @var := CONCAT('SET @chek := (SELECT user\_id FROM my\_chat.r', @id\_, ' WHERE user\_id = @userID)');

PREPARE query FROM @var; EXECUTE query;

IF (@chek > 0) THEN

SET @var := CONCAT('UPDATE my\_chat.r', @id\_, ' SET message = ''', message\_, ''' WHERE user\_id = @userID');

ELSE

SET @var := CONCAT('insert into my\_chat.r', @id\_, ' (user\_id, message) values(@userID,''', message\_, ''')');

UPDATE my\_chat.users SET n\_request := n\_request + 1 WHERE id = @id\_;

END IF;

PREPARE query FROM @var; EXECUTE query;

UPDATE my\_chat.users SET request := TRUE WHERE id = @id\_;

END IF;

DROP TABLE addressees;

COMMIT;

END//

DROP PROCEDURE IF EXISTS ACCEPT\_REQUEST;

CREATE PROCEDURE ACCEPT\_REQUEST(login VARCHAR(40))

BEGIN

START TRANSACTION;

SET @var = CONCAT('SET @requesterID := (SELECT user\_id FROM my\_chat.r', @userID, ' WHERE user\_id = (SELECT id FROM my\_chat.users WHERE log = ''', login, '''))');

PREPARE query FROM @var; EXECUTE query;

IF(@requesterID > 0) THEN

IF(@userID < @requesterID) THEN

SET @name = CONCAT(@userID, @requesterID);

ELSE

SET @name = CONCAT(@requesterID, @userID);

END IF;

SET @var := CONCAT('CREATE TABLE if not exists my\_chat.', @name,' LIKE my\_chat.messages');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('CREATE TABLE if not exists my\_chat.part', @name,' LIKE my\_chat.chat\_m\_u\_d');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('insert into my\_chat.p', @userID, ' (name\_of\_messages\_table) values(@name)');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('insert into my\_chat.part', @name, ' (user\_id) values(@userID)');

PREPARE query FROM @var; EXECUTE query;

IF(@userID != @requesterID) THEN

SET @var := CONCAT('insert into my\_chat.p', @requesterID, ' (name\_of\_messages\_table) values(@name)');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('insert into my\_chat.part', @name, ' (user\_id) values(@requesterID)');

PREPARE query FROM @var; EXECUTE query;

END IF;

SET @var := CONCAT('DELETE FROM my\_chat.r', @userID, ' WHERE user\_id = @requesterID');

PREPARE query FROM @var; EXECUTE query;

UPDATE my\_chat.users SET n\_request := n\_request - 1 WHERE id = @userID;

ELSE

SELECT CONCAT('you have no requester with login: '' ', login, ' ''.');

END IF;

COMMIT;

END//

DROP PROCEDURE IF EXISTS CREATE\_CHAT;

CREATE PROCEDURE CREATE\_CHAT(name\_of\_chat VARCHAR(60), def\_user ENUM('g', 'u'))

BEGIN

START TRANSACTION;

IF(def\_user = 'g' OR def\_user = 'u') THEN

UPDATE my\_chat.users SET n\_created\_chats := (SELECT @n\_chat := n\_created\_chats + 1) WHERE id = @userID;

SET @name := CONCAT(@userID, '\_', @n\_chat);

SET @var := CONCAT('CREATE TABLE my\_chat.', @name,' LIKE my\_chat.messages');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('CREATE TABLE my\_chat.part', @name,' LIKE my\_chat.chat\_m\_', def\_user, '\_d');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('insert into my\_chat.part', @name, ' (user\_id, acs) values(@userID, ''r'')');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('CREATE TABLE my\_chat.name', @name,' LIKE my\_chat.chat\_name');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('insert into my\_chat.name', @name, ' (name) values(name\_of\_chat)');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('insert into my\_chat.с', @userID, ' (name\_of\_messages\_table) values(@name)');

PREPARE query FROM @var; EXECUTE query;

SET @var := CONCAT('CREATE TABLE my\_chat.', @userID, '\_', @name,' LIKE my\_chat.r\_c\_m');

PREPARE query FROM @var; EXECUTE query;

ELSE

SELECT CONCAT('wrong second parameter ', def\_user);

END IF;

COMMIT;

END//

DROP PROCEDURE IF EXISTS IS\_FRIEND\_IN\_CHAT;

CREATE PROCEDURE IS\_FRIEND\_IN\_CHAT(friendID BIGINT UNSIGNED, chatID BIGINT UNSIGNED)

BEGIN

START TRANSACTION;

-- берем реальное id друга

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.p', @userID,' WHERE id = friendID)');

PREPARE query FROM @var; EXECUTE query;

SET @var = CONCAT('SET @frienduserID := (SELECT id FROM my\_chat.users WHERE id IN (SELECT user\_id FROM my\_chat.part', @name, '))');

PREPARE query FROM @var; EXECUTE query;

-- проверяем есть ли он в данном чате

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.c', @userID,' WHERE id = chatID)');

PREPARE query FROM @var; EXECUTE query;

-- вернет непусто, если есть

SET @var = CONCAT('SELECT 1 FROM my\_chat.part', @name, ' WHERE user\_id = @frienduserID');

PREPARE query FROM @var; EXECUTE query;

COMMIT;

END//

DROP PROCEDURE IF EXISTS ADD\_PART;

CREATE PROCEDURE ADD\_PART(friendID BIGINT UNSIGNED, chatID BIGINT UNSIGNED)

BEGIN

START TRANSACTION;

-- берем реальное id друга

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.p', @userID,' WHERE id = friendID)');

PREPARE query FROM @var; EXECUTE query;

SET @var = CONCAT('SET @frienduserID := (SELECT id FROM my\_chat.users WHERE id IN (SELECT user\_id FROM part', @name, '))');

PREPARE query FROM @var; EXECUTE query;

-- проверяем, может ли userID добавлять друзей

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.c', @userID,' WHERE id = chatID)');

PREPARE query FROM @var; EXECUTE query;

SET @chek := NULL;

SET @var = CONCAT('SET @chek := (SELECT 1 FROM my\_chat.part', @name, ' WHERE user\_id = @userID AND acs IN (''g'',''r''))');

PREPARE query FROM @var; EXECUTE query;

IF(@chek != NULL) THEN

-- добавляем пользователя в чат

SET @var := CONCAT('insert into my\_chat.part', @name, ' (user\_id) values(@frienduserID)');

PREPARE query FROM @var; EXECUTE query;

-- создаем список ссылок на сообщения в списке сообщений для @frienduserID

SET @var := CONCAT('CREATE TABLE my\_chat.', @frienduserID, '\_', @name,' LIKE my\_chat.r\_c\_m');

PREPARE query FROM @var; EXECUTE query;

ELSE

SELECT 1;

END IF;

COMMIT;

END//

DROP PROCEDURE IF EXISTS SET\_ACCESS;

CREATE PROCEDURE SET\_ACCESS(login VARCHAR(40), chatID BIGINT UNSIGNED, acs\_ ENUM('g','u'))

BEGIN

START TRANSACTION;

SET @\_acs\_ := acs\_;

IF (@\_acs\_ = 'u') THEN SET @\_acs\_ := ''; END IF;

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.c', @userID,' WHERE id = chatID)');

PREPARE query FROM @var; EXECUTE query;

SET @chek := NULL;

SET @var = CONCAT('SET @chek := (SELECT 1 FROM my\_chat.part', @name, ' WHERE user\_id = @userID AND acs = ''r'')');

PREPARE query FROM @var; EXECUTE query;

IF(@chek != NULL) THEN

SET @id\_ := NULL;

SET @id\_\_ := (SELECT id FROM my\_chat.users WHERE log = login);

SET @var = CONCAT('SET @id\_ := (SELECT user\_id FROM my\_chat.part', @name, ' WHERE user\_id = @id\_\_)');

PREPARE query FROM @var; EXECUTE query;

IF(@id\_ != NULL AND @id\_ != @userID) THEN

SET @var = CONCAT('UPDATE my\_chat.part', @name, ' SET acs := ', \_acs\_,' WHERE user\_id = @id\_');

PREPARE query FROM @var; EXECUTE query;

ELSE

SELECT 2;

END IF;

ELSE

SELECT 1;

END IF;

COMMIT;

END//

DROP PROCEDURE IF EXISTS ONLINE;

CREATE PROCEDURE ONLINE()

BEGIN

START TRANSACTION;

SELECT unread\_private\_chats, unread\_chats, request FROM my\_chat.users WHERE id = @userID;

UPDATE my\_chat.users SET unread\_private\_chats := FALSE, unread\_chats := FALSE, request := FALSE, last\_tick := NOW() WHERE id = @userID;

COMMIT;

END//

DROP PROCEDURE IF EXISTS GNM\_PART\_F\_CHAT;

CREATE PROCEDURE GNM\_PART\_F\_CHAT(n BIGINT UNSIGNED, m BIGINT UNSIGNED, like\_ VARCHAR(50), chatID BIGINT UNSIGNED)

BEGIN

-- START TRANSACTION;

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.c', @userID,' WHERE id = chatID)');

PREPARE query FROM @var; EXECUTE query;

SET @var = CONCAT('SELECT log, acs FROM my\_chat.users, my\_chat.part', @name,' WHERE id = user\_id AND log LIKE ''', like\_,''' ORDER BY log ASC LIMIT ', n, ', ', m);

PREPARE query FROM @var; EXECUTE query;

-- COMMIT;

END//

DROP PROCEDURE IF EXISTS GNM\_REQUESTS;

CREATE PROCEDURE GNM\_REQUESTS(n BIGINT UNSIGNED, m BIGINT UNSIGNED, like\_ VARCHAR(50))

BEGIN

START TRANSACTION;

SET @var := CONCAT('SELECT U.log, M.message FROM my\_chat.r', @userID,' AS M, my\_chat.users AS U WHERE U.log LIKE ''', like\_,''' AND U.id = M.user\_id ORDER BY M.last\_tick DESC LIMIT ', n, ', ', m);

PREPARE query FROM @var; EXECUTE query;

COMMIT;

END//

DROP PROCEDURE IF EXISTS GNM\_MESSAGES\_P;

CREATE PROCEDURE GNM\_MESSAGES\_P(n BIGINT UNSIGNED, m BIGINT UNSIGNED, friendID BIGINT UNSIGNED)

BEGIN

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.p', @userID,' WHERE id = ', friendID, ')');

PREPARE query FROM @var; EXECUTE query;

SET @var = CONCAT('SET @unread\_ := (SELECT unread FROM my\_chat.p', @userID,' WHERE id = ', friendID, ')');

PREPARE query FROM @var; EXECUTE query;

IF (@unread\_ = TRUE AND n = 0) THEN

START TRANSACTION;

SET @var = CONCAT('UPDATE my\_chat.p', @userID, ' SET unread := FALSE, last\_tick := last\_tick WHERE id = ', friendID);

PREPARE query FROM @var; EXECUTE query;

UPDATE my\_chat.users SET n\_unread\_private\_chats := n\_unread\_private\_chats - 1 WHERE id = @userID;

COMMIT;

END IF;

SET @var = CONCAT('SELECT M.id, U.log, M.message, M.last\_tick FROM my\_chat.users AS U, my\_chat.', @name,' AS M WHERE user\_id = U.id ORDER BY M.id DESC LIMIT ', n, ', ', m);

PREPARE query FROM @var; EXECUTE query;

END//

DROP PROCEDURE IF EXISTS GNM\_MESSAGES\_С;

CREATE PROCEDURE GNM\_MESSAGES\_С(n BIGINT UNSIGNED, m BIGINT UNSIGNED, chatID BIGINT UNSIGNED)

BEGIN

-- START TRANSACTION;

SET @var = CONCAT('SET @name := (SELECT name\_of\_messages\_table FROM my\_chat.c', @userID,' WHERE id = ', chatID, ')');

PREPARE query FROM @var; EXECUTE query;

SET @var = CONCAT('SET @unread\_ := (SELECT unread FROM my\_chat.c', @userID,' WHERE id = ', chatID, ')');

PREPARE query FROM @var; EXECUTE query;

IF (@unread\_ = TRUE AND n = 0) THEN

START TRANSACTION;

SET @var = CONCAT('UPDATE my\_chat.c', @userID, ' SET unread := FALSE, last\_tick := last\_tick WHERE id = ', chatID);

PREPARE query FROM @var; EXECUTE query;

UPDATE my\_chat.users SET n\_unread\_chats := n\_unread\_chats - 1 WHERE id = @userID;

COMMIT;

END IF;

SET @var = CONCAT('SELECT ID\_M\_.id, U.log, M.message, M.last\_tick FROM my\_chat.users AS U, my\_chat.', @userID, '\_', @name,' AS ID\_M\_, my\_chat.', @name,' AS M WHERE M.user\_id = U.id AND ID\_M\_.id\_m = M.id ORDER BY ID\_M\_.id DESC LIMIT ', n, ', ', m);

PREPARE query FROM @var; EXECUTE query;

-- COMMIT;

END//

DROP PROCEDURE IF EXISTS GNM\_CHATS;

CREATE PROCEDURE GNM\_CHATS(n BIGINT UNSIGNED, m BIGINT UNSIGNED, like\_ VARCHAR(50), ch CHARACTER(1))

BEGIN

START TRANSACTION;

IF (ch = 'p' OR ch = 'c') THEN

SET @var := CONCAT('CREATE TEMPORARY TABLE addressees SELECT \* FROM my\_chat.', ch, @userID, ' AS LALKA');

PREPARE query FROM @var; EXECUTE query;

ALTER TABLE addressees ADD COLUMN num SERIAL;

ALTER TABLE addressees ADD INDEX (num);

ALTER TABLE addressees ADD COLUMN name VARCHAR(64);

SET @i := 1;

SET @max\_ := (SELECT MAX(num) FROM addressees);

IF (ch = 'p') THEN

WHILE @i <= @max\_ DO

SET @tmp\_name := (SELECT name\_of\_messages\_table FROM addressees WHERE num = @i);

SET @var := CONCAT('UPDATE addressees SET name := (SELECT log FROM my\_chat.users WHERE id IN (SELECT user\_id FROM my\_chat.part', @tmp\_name, ') AND (id != @userID OR (1 = (SELECT COUNT(user\_id) FROM my\_chat.part', @tmp\_name, '))) AND log LIKE ''', like\_,''') WHERE num = @i');

PREPARE query FROM @var; EXECUTE query;

SET @i := @i + 1;

END WHILE;

ELSE

WHILE @i <= @max\_ DO

SET @var := CONCAT('UPDATE addressees SET name := (SELECT N.name FROM my\_chat.name', (SELECT name\_of\_messages\_table FROM addressees WHERE num = @i), ' AS N WHERE N.name LIKE ''', like\_,''')');

PREPARE query FROM @var; EXECUTE query;

SET @i := @i + 1;

END WHILE;

END IF;

SELECT id, name FROM addressees ORDER BY last\_tick DESC LIMIT n, m;

DROP TABLE addressees;

END IF;

COMMIT;

END//

DROP PROCEDURE IF EXISTS GET\_LAST\_USER\_TICK;

CREATE PROCEDURE GET\_LAST\_USER\_TICK(login VARCHAR(40))

BEGIN

START TRANSACTION;

SELECT last\_tick FROM my\_chat.users WHERE log = login;

COMMIT;

END//

DROP PROCEDURE IF EXISTS GET\_USER\_DATA;

CREATE PROCEDURE GET\_USER\_DATA(login VARCHAR(40))

BEGIN

START TRANSACTION;

SELECT status FROM my\_chat.users WHERE log = login;

COMMIT;

END//

DROP PROCEDURE IF EXISTS GNM\_USERS;

CREATE PROCEDURE GNM\_USERS(n BIGINT UNSIGNED, m BIGINT UNSIGNED, login VARCHAR(40))

BEGIN

START TRANSACTION;

SET @var := CONCAT('CREATE TEMPORARY TABLE addressees SELECT LALKA.name\_of\_messages\_table FROM my\_chat.p', @userID, ' AS LALKA');

PREPARE query FROM @var; EXECUTE query;

ALTER TABLE addressees ADD COLUMN num SERIAL;

ALTER TABLE addressees ADD INDEX (num);

ALTER TABLE addressees ADD COLUMN user\_id BIGINT;

SET @i := 1;

SET @max\_ := (SELECT MAX(num) FROM addressees);

WHILE @i <= @max\_ DO

SET @tmp\_name := (SELECT name\_of\_messages\_table FROM addressees WHERE num = @i);

SET @var := CONCAT('UPDATE addressees SET user\_id := (SELECT user\_id FROM my\_chat.part', @tmp\_name, ' WHERE (user\_id != @userID OR (1 = (SELECT COUNT(user\_id) FROM my\_chat.part', @tmp\_name, ')))) WHERE num = @i');

PREPARE query FROM @var; EXECUTE query;

SET @i := @i + 1;

END WHILE;

ALTER TABLE addressees DROP COLUMN num;

ALTER TABLE addressees DROP COLUMN name\_of\_messages\_table;

SET @var := CONCAT('SELECT log FROM my\_chat.users WHERE id <> ALL (SELECT user\_id FROM addressees) AND id <> ALL (SELECT user\_id FROM my\_chat.r', @userID, ') AND log LIKE ''', login,''' ORDER BY last\_tick DESC LIMIT ', n, ', ', m);

PREPARE query FROM @var; EXECUTE query;

DROP TABLE addressees;

COMMIT;

END//

DROP PROCEDURE IF EXISTS GET\_UNREAD\_LAST\_TICK;

CREATE PROCEDURE GET\_UNREAD\_LAST\_TICK(id\_ BIGINT UNSIGNED, ch CHARACTER(1))

BEGIN

-- START TRANSACTION;

IF (ch = 'p' OR ch = 'c') THEN

SET @var := CONCAT('SELECT unread, last\_tick FROM my\_chat.', ch, @userID, ' WHERE id = ', id\_);

PREPARE query FROM @var; EXECUTE query;

END IF;

-- COMMIT;

END//

DELIMITER ;

-- --------------------------------------------------------------------------------------

## Код, що описує сервер менеджер

### Class MyException

#pragma once

#include<string>

using namespace std;

class MyException{

protected:

string errorCause;

int errorCode;

public:

typedef int j;

MyException(int errorCode, string errorCause = "") : errorCode(errorCode), errorCause(errorCause) {}

// получить код ошибки

int getErrorCode() const { return errorCode; }

// получить объяснение

string getErrorCause() { return string(errorCause.c\_str()); }

// оператор преобразования в стринг

operator string(){

return "Error:\nErrorCode is " + to\_string(errorCode) + "\nErrorCause is " + errorCause;

}

virtual string toString(){

return this->operator string();

}

virtual ~MyException(){}

};

### class MyData

#pragma once

//обертка MyData позволяет хранить обект и непариться про засорение памяти при копировании.

//Если нужны данные, чтохранятся внутри - можно воспользоваться getData(),

//но тогда необходим нормальный публичным (public) копирующий конструктор у класса Т;

//в противном случае пользователь класса MyData берет ответственность сохранности данных на себя.

template<class T>

class MyData{

private:

void\* constructed\_check;

//отмечает конструирование объекта

inline void constructing(){ constructed\_check = (void\*)this; }

//проверка конструирования объекта

inline bool constructed() const {

if (constructed\_check == (void\*)this) return true;

return false;

}

struct Node{ T data; unsigned long n; } \*node;

void init(const T &data){

Node tmp = {data, unsigned long(1)};

node = new Node(tmp);

constructing();

}

void init(const MyData<T> &obj){

node = obj.node;

(node->n)++;

constructing();

}

void destruct(){

if (--(node->n) == 0)

delete node;

}

public:

//возвращает количесто существующих объектов с тем же занчением поля data

inline unsigned long num\_of\_copies() const { return (node->n) - 1; }

inline MyData(const T &data){ init(data); }

inline MyData(const MyData<T> &obj){ init(obj); }

virtual inline ~MyData(){ destruct(); }

inline MyData& operator=(const MyData<T>& obj){

if (constructed()) destruct();

init(obj);

return \*this;

}

(const T)\* operator->(){ return (T\*)node; }

//возвращает данные, что хранятся тут,

//в надежде на надежность конструктора копирования класса T

inline T getData(){

return node->data;

}

inline void setData(const T &data){

destruct();

init(data);

}

};

### class MySocket

#pragma once;

#include<string>

#include"MyException.h"

#pragma warning(disable : 4290)

using namespace std;

class MySocketException : public MyException{

public:

MySocketException(int errorCode, string errorCause = "") : MyException(errorCode, errorCause){}

};

struct MySocketAddress{

virtual char\* get\_ip () const = 0;

virtual unsigned short get\_port () const = 0;

virtual ~MySocketAddress(){};

};

class MySocket{

public:

virtual ~MySocket () throw (MySocketException){}

// привязка сокета к локальномы "имени сокета" (здесь)

virtual void bind (const MySocketAddress \* const &address) throw (MySocketException) = 0;

// отправить сообщение (отправляются len байт по адресу data)

// возвращает количество отосланых байт

virtual int sendto (const MySocketAddress \* const &address, const char \* const &data, const int &len) throw (MySocketException) = 0;

// принять сообщение ожидаемой длины len

// вернет либо количество полученных байт, либо 0. 0 вовсе не означает, что было получено 0 байт, это значение означает, сокет был закрыт до вызова recvfrom().

virtual int recvfrom (MySocketAddress \* &address, char \* const &data, const int &len) throw (MySocketException) = 0;

virtual int recvfrom (char \* const &data, const int &len) throw (MySocketException) = 0;

// отправить len байт (сокет уже должен быль привязан к внегнему адресу)

// возвращает количество отосланых байт

virtual int send (const char \* const &data, const int &len) throw (MySocketException) = 0;

// принять байты (не больше len)

// вернет либо количество полученных байт, либо 0. 0 вовсе не означает, что было получено 0 байт, это значение означает, сокет был закрыт до вызова recv().

virtual int recv (char \* const &data, const int &len) throw (MySocketException) = 0;

// обертка над send. пишет все len байт из data, когда send может записать не все

virtual int write (const char \* const &data, const int &len) throw (MySocketException) = 0;

// обертка над recv. пытается читать все len байт в data (ждет, пока не придут все len байт), когда recv может читать только те данные, которые пришли на данный момент

virtual int read (char \* const &data, const int &len) throw (MySocketException) = 0;

// привязка сокета к глобальному "имени сокета" (в сети)

virtual void connect (const MySocketAddress \* const &address) throw (MySocketException) = 0;

// можно вызвать только для сокетов после вызова bind и перед вызовом accept

// переводит совет в состояние listen

// параметр backlog задает задает максимальное число соединений.

virtual void listen (const int &backlog) throw (MySocketException) = 0;

// возвращает указатель на сокет, который образовался в результате запроса на подсоединение к серверу, за который отвечает этот сокет.

// в address записывается адресс, от куда поступил запрос

virtual MySocket\* accept (MySocketAddress \* &address) throw (MySocketException) = 0;

virtual MySocket\* accept () throw (MySocketException) = 0;

};

### MyWinSocket.h

#pragma once

#pragma comment(lib, "ws2\_32.lib")//подключение библиотеки win\_sock\_2.0 (как я думаю, для 32 бит)

#include<WinSock2.h>

#include"MySocket.h"

#include"MyData.h"

class MyWinSocketException : public MySocketException{

public:

MyWinSocketException(int errorCode, string errorCause = "") : MySocketException(errorCode, errorCause){}

};

class MyWinSocketAddress : public MySocketAddress{

private:

MyData<sockaddr\_in> ad;

public:

MyWinSocketAddress(const char \* const &ip = "", const unsigned short &port = 0) throw (MyWinSocketException);

MyWinSocketAddress(const sockaddr& addr) throw (MyWinSocketException);

MyWinSocketAddress(const sockaddr\_in& addr) throw (MyWinSocketException);

virtual char\* get\_ip() const{ return inet\_ntoa(((MyData<sockaddr\_in>)ad)->sin\_addr); }

virtual unsigned short get\_port() const { return ntohs(((MyData<sockaddr\_in>)ad)->sin\_port); }

operator sockaddr(){return \*((sockaddr\*)&ad.getData()); }

};

class MyWinSocket : public MySocket{

protected:

MyData<SOCKET> socket;

//счетчик объектов этого класса

static unsigned long n;

public:

MyWinSocket(const int &type = SOCK\_STREAM) throw (MyWinSocketException);

virtual ~MyWinSocket() throw (MyWinSocketException);

// привязка сокета к "имени сокета"

virtual void bind (const MySocketAddress \* const &address) throw (MyWinSocketException);

// отправить сообщение (отправляются len байт по адресу в data)

// возвращает количество отосланых байт

virtual int sendto (const MySocketAddress \* const &address, const char \* const &data, const int &len) throw (MySocketException);

// принять сообщение ожидаемой длины len

// вернет либо количество полученных байт, либо 0. 0 вовсе не означает, что было получено 0 байт, это значение означает, сокет был закрыт до вызова recvfrom().

virtual int recvfrom (MySocketAddress \* &address, char \* const &data, const int &len) throw (MySocketException);

virtual int recvfrom (char \* const &data, const int &len) throw (MySocketException);

// отправить len байт (сокет уже должен быль привязан к внегнему адресу)

// возвращает количество отосланых байт

virtual int send (const char \* const &data, const int &len) throw (MySocketException);

// принять байты (не больше len)

// вернет либо количество полученных байт, либо 0. 0 вовсе не означает, что было получено 0 байт, это значение означает, сокет был закрыт до вызова recv().

virtual int recv (char \* const &data, const int &len) throw (MySocketException);

// привязка сокета к глобальному "имени сокета" (в сети)

// обертка над send. пишет все len байт из data, когда send может записать не все

virtual int write (const char \* const &data, const int &len) throw (MySocketException);

// обертка над recv. пытается читать все len байт в data (ждет, пока не придут все len байт), когда recv может читать только те данные, которые пришли на данный момент

virtual int read (char \* const &data, const int &len) throw (MySocketException);

// привязка сокета к глобальному "имени сокета" (в сети)

virtual void connect (const MySocketAddress \* const &address) throw (MySocketException);

// можно вызвать только для сокетов после вызова bind и перед вызовом accept

// переводит совет в состояние listen

// параметр backlog задает задает максимальное число соединений.

virtual void listen (const int &backlog = SOMAXCONN) throw (MySocketException);

// возвращает указатель на сокет, который образовался в результате запроса на подсоединение к серверу, за который отвечает этот сокет.

// в address записывается адресс, от куда поступил запрос

virtual MySocket\* accept (MySocketAddress \* &address) throw (MySocketException);

virtual MySocket\* accept () throw (MySocketException);

};

### MyWinSocket.cpp

#include"MyWinSocket.h"

inline static void init\_winSocket() throw (MyWinSocketException) {

WORD winSock\_version = MAKEWORD(2, 0); // запрашиваем winsock версии 2.0

WSADATA winSock\_data; // сюда будут помещены данные об инициализированном интерфейсе winsock

int error = WSAStartup (winSock\_version, &winSock\_data); // непосредственно инициализция

if (error != 0)

throw MyWinSocketException(GetLastError(), "initSock: could not initialize winSock.");

}

inline static void cleanup\_winSocket() throw (MyWinSocketException){

int error = WSACleanup();

if (error != 0)

throw MyWinSocketException(GetLastError(), "cleanSock: could not perform cleaning up.");

}

inline static sockaddr\_in make\_sockaddr\_in(const char \* const &ip, const unsigned short &port){

sockaddr\_in ad;

ad.sin\_family = AF\_INET;

if (strcmp(ip, "") == 0) ad.sin\_addr.s\_addr = INADDR\_ANY;

else ad.sin\_addr.s\_addr = inet\_addr (ip);

if (ad.sin\_addr.s\_addr == INADDR\_NONE) throw MyWinSocketException(GetLastError(), "WinSocketAddress: the provided IP address seems to be invalid.");

ad.sin\_port = htons (port);

return ad;

}

MyWinSocketAddress::MyWinSocketAddress(const char \* const &ip, const unsigned short &port) throw (MyWinSocketException) : ad(make\_sockaddr\_in(ip, port)) {}

inline static sockaddr\_in make\_sockaddr\_in(const sockaddr\_in &addr){

if (addr.sin\_addr.s\_addr == INADDR\_NONE) throw MyWinSocketException(GetLastError(), "WinSocketAddress: the provided IP address seems to be invalid.");

return addr;

}

MyWinSocketAddress::MyWinSocketAddress(const sockaddr\_in &addr) throw (MyWinSocketException) : ad(make\_sockaddr\_in(addr)) {}

MyWinSocketAddress::MyWinSocketAddress(const sockaddr& addr) throw (MyWinSocketException) : ad(make\_sockaddr\_in(\*((sockaddr\_in\*)&addr))) {}

unsigned long MyWinSocket::n = 0;

MyWinSocket::MyWinSocket(const int &type) throw (MyWinSocketException) : socket(0){

if (n++ == 0) init\_winSocket();

SOCKET sock = ::socket(PF\_INET, type, 0);

if (sock == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket: could not create a WinSocket.");

socket.setData(sock);

}

MyWinSocket::~MyWinSocket() throw (MyWinSocketException){

if (this->socket.num\_of\_copies() == 0){

int error = closesocket (socket.getData()); // закрываем сокет

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "~WinSocket: could not close WinSocket properly");

if (--n == 0) cleanup\_winSocket();

}

}

void MyWinSocket::bind(const MySocketAddress \* const &address) throw (MyWinSocketException){

MyWinSocketAddress addr(address->get\_ip(), address->get\_port());

int error = ::bind(socket.getData(), &((sockaddr)addr), sizeof(sockaddr\_in));

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::bind: could not bind socket to a provided address.");

}

int MyWinSocket::sendto (const MySocketAddress \* const &address, const char \* const &data, const int &len) throw (MySocketException){

MyWinSocketAddress addr(address->get\_ip(), address->get\_port());

int error = ::sendto (socket.getData(), data, len, 0, &((sockaddr)addr), sizeof(sockaddr\_in));

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::sendto: could not send data.");

return error;

}

int MyWinSocket::recvfrom (MySocketAddress \* &address, char \* const &data, const int &len) throw (MySocketException){

int address\_len = sizeof(sockaddr\_in);

sockaddr \*address\_ = new sockaddr;

int error = ::recvfrom (socket.getData(), data, len, 0, address\_, &address\_len);

if (error == SOCKET\_ERROR || address\_len != sizeof(sockaddr\_in)) throw MyWinSocketException(GetLastError(), "WinSocket::recvfrom: could not receive data.");

address = (MySocketAddress\*)(new MyWinSocketAddress(\*address\_));

delete address\_;

return error;

}

int MyWinSocket::recvfrom (char \* const &data, const int &len) throw (MySocketException){

int error = ::recvfrom (socket.getData(), data, len, 0, nullptr, nullptr);

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::recvfrom: could not receive data.");

return error;

}

int MyWinSocket::send (const char \* const &data, const int &len) throw (MySocketException){

int error = ::send (socket.getData(), data, len, 0);

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::send: could not send data.");

return error;

}

int MyWinSocket::recv (char \* const &data, const int &len) throw (MySocketException){

int error = ::recv (socket.getData(), data, len, 0);

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::recv: could not receive data.");

return error;

}

int MyWinSocket::write (const char \* const &data, const int &len) throw (MySocketException){

int \_len = len, error = 0;

char \*ptr = (char\*)data;

do{ error = ::send (socket.getData(), (ptr = ptr + error), \_len, 0); }while(error > 0 && (\_len = \_len - error) > 0 || error == SOCKET\_ERROR && GetLastError() == WSAEINTR);

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::write: could not write data.");

return len - \_len;

}

int MyWinSocket::read (char \* const &data, const int &len) throw (MySocketException){

int \_len = len, error = 0;

char \*ptr = data;

do{ error = ::recv (socket.getData(), (ptr = ptr + error), \_len, 0); }while(error > 0 && (\_len = \_len - error) > 0 || error == SOCKET\_ERROR && GetLastError() == WSAEINTR);

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::read: could not read data.");

return len - \_len;

}

void MyWinSocket::connect (const MySocketAddress \* const &address) throw (MySocketException){

MyWinSocketAddress addr(address->get\_ip(), address->get\_port());

int error = ::connect(socket.getData(), &((sockaddr)addr), sizeof(sockaddr\_in));

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::connect: could not connect socket to a provided address.");

}

void MyWinSocket::listen (const int &backlog) throw (MySocketException){

int error = ::listen(socket.getData(), backlog);

if (error == SOCKET\_ERROR) throw MyWinSocketException(GetLastError(), "WinSocket::listen: could not make listen socket.");

}

MySocket\* MyWinSocket::accept (MySocketAddress \* &address) throw (MySocketException){

int address\_len = sizeof(sockaddr\_in);

sockaddr \*address\_ = new sockaddr;

SOCKET sock = ::accept(socket.getData(), address\_, &address\_len);

if (sock == INVALID\_SOCKET || address\_len != sizeof(sockaddr\_in)) throw MyWinSocketException(GetLastError(), "WinSocket::accept: could not accsept connection.");

address = (MySocketAddress\*)(new MyWinSocketAddress(\*address\_));

delete address\_;

MyWinSocket \* res = new MyWinSocket(\*this);

res->socket.setData(sock);

n++;

return (MySocket\*)res;

}

MySocket\* MyWinSocket::accept () throw (MySocketException){

SOCKET sock = ::accept(socket.getData(), nullptr, nullptr);

if (sock == INVALID\_SOCKET) throw MyWinSocketException(GetLastError(), "WinSocket::accept: could not accsept connection.");

MyWinSocket \* res = new MyWinSocket(\*this);

res->socket.setData(sock);

n++;

return (MySocket\*)res;

}

### usingClasses.h

#pragma once

#include"MyWinSocket.h"

#include"MyData.h"

#pragma region usingClasses

unsigned int my\_strlen(const char \* const &s){

if (s == nullptr) return 0;

const char \*res = s;

for (; \*res != '\0'; res++){}

return res - s;

}

class charPtr{

typedef char T;

MyData<T\*> ptr;

unsigned short len;//не менять тип!

void destr(){

if(ptr.num\_of\_copies() == 0) delete ptr.getData();

}

public:

charPtr() : ptr(nullptr), len(0){}

//использовать только когда в конце есть нулевой байт

charPtr(const T\* const &ptr) : ptr(nullptr){

init(ptr, my\_strlen(ptr));

}

charPtr(const T\* const &ptr, const unsigned short &len) : ptr(nullptr){

init(ptr, len);

}

charPtr(T\* const &ptr, const unsigned short &len) : ptr(nullptr){

init(ptr, len);

}

void init(T\* const &ptr, const unsigned short &len){

destr();

this->len = len;

this->ptr.setData((T\*)ptr);

}

void init(const T\* const &ptr, const unsigned short &len){

char \*buf = new char[len];

for (char\* tmp1 = buf, \*tmp2 = (char\*)ptr, \*end = buf + len; tmp1 != end; \*tmp1++ = \*tmp2++){}

this->init(buf, len);

}

~charPtr(){

destr();

}

unsigned short getLen() const{

return len;

}

//за объект класса Т, что возвращается этим методом по указателю Т\*, создатель класса template<class T> class Ptr не несет ответственности

T\* getPtr() const{

return ((MyData<T\*>)ptr).getData();

}

operator bool(){

if(this->getPtr()) return true;

return false;

}

};

ostream& operator<<(ostream& stream, const charPtr &obj){

char \*ptr = obj.getPtr(), \*end = ptr + obj.getLen();

for (;ptr < end;) stream.put(\*ptr++);

return stream;

}

bool operator<(const charPtr &left, const charPtr &right){

if (left.getLen() < right.getLen()) return true;

else if(left.getLen() > right.getLen()) return false;

for(char \*ptr1 = left.getPtr(), \*ptr2 = right.getPtr(), \* const end = left.getPtr() + left.getLen() - 1; ptr1 <= end; ptr1++, ptr2++){

if((\*ptr1) < (\*ptr2)) return true;

else if((\*ptr1) > (\*ptr2)) return false;

}

return false;

}

bool operator ==(const charPtr &left, const charPtr &right){

if(left.getLen() != right.getLen()) return false;

for(char \*ptr1 = left.getPtr(), \*ptr2 = right.getPtr(), \* const end = left.getPtr() + left.getLen() - 1; ptr1 <= end; ptr1++, ptr2++){

if((\*ptr1) != (\*ptr2)) return false;

}

return true;

}

bool operator >(const charPtr &left, const charPtr &right){

if (left.getLen() > right.getLen()) return true;

else if(left.getLen() < right.getLen()) return false;

for(char \*ptr1 = left.getPtr(), \*ptr2 = right.getPtr(), \* const end = left.getPtr() + left.getLen() - 1; ptr1 <= end; ptr1++, ptr2++){

if((\*ptr1) > (\*ptr2)) return true;

else if((\*ptr1) < (\*ptr2)) return false;

}

return false;

}

bool operator !=(const charPtr &left, const charPtr &right){

return !(left == right);

}

charPtr operator+(const charPtr &left, const charPtr &right){

char \*buf = new char[left.getLen() + right.getLen()], \*ptr1 = left.getPtr(), \*ptr2 = right.getPtr();

for (char\* end = buf + left.getLen(); buf < end; \*buf++ = \*ptr1++){}

for (char\* end = buf + right.getLen(); buf < end; \*buf++ = \*ptr2++){}

buf -= left.getLen() + right.getLen();

return charPtr(buf, left.getLen() + right.getLen());

}

charPtr operator+(const charPtr &left, const char \* const &right){

return left + charPtr(right);

}

charPtr operator+(const char \* const &left, const charPtr &right){

return charPtr(left) + right;

}

class MySocketPtr{

typedef MySocket\* T;

MyData<T> data;

public:

MySocketPtr(const T &sock) : data(sock){}

~MySocketPtr(){

if(data.num\_of\_copies() == 0) delete data.getData();

}

T operator->(){

return data.getData();

}

};

#pragma region for\_socket

/\*\*

общение с клиентом:

-по протоколу TCP.

-в заглавие каждго сообщения ложется 2 байта, которые представляют из себя целое неотрицательное число [0; 256 \* 256 - 1] записаное в порядке байт от старшего к младшему.

таким образом каждая сторона будет знать, какой длины сообшение ожидать.

\*\*/

//отправляет len байт по адресу str, отправляя перед ними количество отправляемых байт (то есть len)

void writestr(MySocketPtr sock, const char\* const &str, const unsigned short &len) throw (MySocketException) {

char len\_[] = {len / 256, len % 256};

sock->write(len\_, 2);

if (len != 0) sock->write(str, len);

}

MySocketPtr& operator<<(MySocketPtr &sock, const charPtr &ptr){

writestr(sock, ptr.getPtr(), ptr.getLen());

return sock;

}

MySocketPtr& operator<<(MySocketPtr &sock, const char \* const &ptr){

writestr(sock, ptr, strlen(ptr));

return sock;

}

//принимает в str, отправленное к нам некоторое количество байт, и записывает это количество в len

//str на входе должна быть не инициализированой

void readstr(MySocketPtr sock, char \*&str, unsigned short &len) throw (MySocketException) {

char len\_[2];

if (sock->read(len\_, 2)) len = len\_[0] \* 256 + len\_[1];

else len = 0;

try{

if (len != 0){

str = new char[len];

sock->read(str, len);

} else str = nullptr;

} catch(...) {

delete str;

throw;

}

}

MySocketPtr& operator>>(MySocketPtr &sock, charPtr &ptr){

char\* buf;

unsigned short len;

readstr(sock, buf, len);

ptr.init(buf, len);

return sock;

}

#pragma endregion

#pragma endregion

### main.cpp

#include "usingClasses.h"

#pragma comment(lib, "libmysql.lib") // подключаем библиотеку

#include <mysql.h>

#include<iostream>

#pragma region TheProgramm

//возможные команды

enum command {authorization, registration};

enum key {check, sender\_private\_message, sender\_chat\_message, sender\_request, accepter\_request, creater\_chat, adder\_part, line, seter\_access, get\_user\_data, get\_last\_user\_tick, geter\_n\_m\_partners\_from, geter\_n\_m\_requests, geter\_n\_m\_messages\_p, geter\_n\_m\_messages\_c, geter\_n\_m\_friens, geter\_n\_m\_confs, geter\_n\_m\_users, geter\_freand\_unread\_last\_tick, geter\_conf\_unread\_last\_tick, get\_friend\_id};

const char\* buf;

string sbuf;

const unsigned short

//количество команд, количество ответов, количество ключей команды авторизации.

n\_commands = 2, n\_answers = 2, n\_aut\_keys = 21,

//максимальное количество байт для логина, максимальное количество байт для пароля.

max\_login\_len = 40, max\_password\_len = 40;

const charPtr

commands[n\_commands] = {"aut", "reg"},//массив команд

answers[n\_answers] = {"0", "1"},//массив ответов //0 - OK, 1 - ERROR

aut\_keys[n\_aut\_keys ] = {"check", "smp", "smc", "sr", "ar", "cc", "ap", "line", "sa", "gud", "glut", "gnmp", "gnmr", "gnmmp", "gnmc", "gnmf", "gnmco", "gnmu", "gful", "gcul", "gfid"},//массив ключей команды авторизации

authorized("authorized."),

registered("registered."),

error("ERROR: "),

warning("WARNING: "),

key("key \' "),

user("user \' "),

authorization\_error("incorrect autorisation parametres."),

authorization\_key\_error(" \' is unknown authorization key."),

registration\_error(" \' is already exist."),

wasnt\_authorizated("You are not logged in."),

unknown\_command(" is unknown command."),

null\_symbol("\0", 1),

login\_len\_error((string("\nYoure login is to long. max length of login is ") + to\_string(max\_login\_len) + ".").c\_str()),

password\_len\_error((string("\nYoure password is to long. max length of password is ") + to\_string(max\_password\_len) + ".").c\_str());

/\*\*

общение с клиентом:

-клиент подключается к серверу и выполняет либо регистрацию, либо авторизацию.

-при авторизации клиент указывае ключ, который определяет, за какое действие будет отвечать это соединение.

смысл заключается в том, что клиент должен авторизроваться для того, чтоб получить соеденение, которое будет отвечать за конкретное действие, а действий может быть много и для полноценной работы клиента необходимо иметь хотя бы по одному соединению для каждого действия (так проще писать программу)

клиент знает количество команд, значение кажной команды (смысл), и порядок, в котором они должны придти.

-каждый запрос - команда и некоторые параметры.(параметры присылаются отдельно от запроса - не склеянные в одно сообщение, а разделенные байтами, которые отвечают за длину)

\*\*/

/\*\*

смысл команд:

-authorization login password key: авторизация

(length of login <= max\_login\_len и length of password <= max\_password\_len)

при удачной авторизации клиенту вернется сообщение соответствующего содержания; сервер переводит соеденение с клиентом в состояние соответствующие ключу key.

-registration login password: регистрация

(length of login <= max\_login\_len и length of password <= max\_password\_len)

при удачной регистрации клиенту вернется сообщение соответствующего содержания.

\*\*/

#pragma region UserBaseManager

typedef MYSQL UBD;//UserBaseDescriptor

UBD \*MD;//MainDescriptor

string Mhost, Muser, Mpassword;

unsigned int Mport;

//возвращает объект типа charPtr с удвоенными апострофами.

//если непоместится - вернет пустой объект типа charPtr.

charPtr mono\_to\_di\_apostrophe(charPtr ptr){

unsigned short n = ptr.getLen();

unsigned int new\_n = 0;

char \*source\_ptr = ptr.getPtr(), \*new\_ptr;

for (unsigned short i = 0; i < n; i++){ if(source\_ptr[i] == '\'') new\_n++; }

new\_n += n;

if(new\_n > 64000) return charPtr();

new\_ptr = new char[new\_n];

for (unsigned short i = 0, j = 0; i < n; i++, j++){

new\_ptr[j] = source\_ptr[i];

if(source\_ptr[i] == '\'') new\_ptr[++j] = '\'';

}

return charPtr(new\_ptr, new\_n);

}

//возвращает объект типа charPtr в котором удвоенные апострофы стали одиназными.

//если есть одинарный апостроф - вернет пустой объект типа charPtr.

charPtr di\_to\_mono\_apostrophe(charPtr ptr){

unsigned short n = ptr.getLen();

unsigned int new\_n = 0;

char \*source\_ptr = ptr.getPtr(), \*new\_ptr;

for (unsigned short i = 0; i < n; i++){

if(source\_ptr[i] == '\''){

if(((i + 1) != n) && (source\_ptr[i + 1] == '\'')){

new\_n++;

i++;

} else return charPtr();

}

}

new\_n = n - new\_n / 2;

new\_ptr = new char[new\_n];

for (unsigned short i = 0, j = 0; i < n; i++, j++){

if(source\_ptr[i] == '\'') {

new\_ptr[j] = '\'';

i++;

}

else new\_ptr[j] = source\_ptr[i];

}

return charPtr(new\_ptr, new\_n);

}

charPtr insert\_first\_bifor\_second(charPtr ptr, char first, char second){

unsigned short n = ptr.getLen();

unsigned int new\_n = 0;

char \*source\_ptr = ptr.getPtr(), \*new\_ptr;

for (unsigned short i = 0; i < n; i++){ if(source\_ptr[i] == second) new\_n++; }

new\_n += n;

if(new\_n > 64000) return charPtr();

new\_ptr = new char[new\_n];

for (unsigned short i = 0, j = 0; i < n; i++, j++){

if(source\_ptr[i] == second) new\_ptr[j++] = first;

new\_ptr[j] = source\_ptr[i];

}

return charPtr(new\_ptr, new\_n);

}

// вернет ID пользователя, если такой есть, и charPtr с nullptr указателем на строку, если такого нет

charPtr authorize(charPtr login, charPtr passwodr) throw (MyException){

charPtr res;

UBD \*desc = mysql\_init(NULL);

mysql\_real\_connect(desc, Mhost.c\_str(), Muser.c\_str(), Mpassword.c\_str(), NULL, Mport, NULL, CLIENT\_MULTI\_STATEMENTS);

if(mysql\_query(desc, ("CALL my\_chat.AUT('" + mono\_to\_di\_apostrophe(login) + "','" + mono\_to\_di\_apostrophe(passwodr) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

MYSQL\_RES \*query\_res = mysql\_store\_result(desc);

auto num\_rows = mysql\_num\_rows(query\_res); // количество строк.

if (num\_rows){

const char \* const buf = mysql\_fetch\_row(query\_res)[0]; //const char \* const потому, что тогда в res будет копироваться строка и после вызова mysql\_free\_result(query\_res) деструктор объекта res выполнется нормально

res.init(buf, strlen(buf));

}

mysql\_free\_result(query\_res); // Очищаем результаты

mysql\_close(desc); // Закрываем соединение

return res;

}

bool existUser(charPtr login) throw (MyException){

bool res = false;

UBD \*desc = mysql\_init(NULL);

mysql\_real\_connect(desc, Mhost.c\_str(), Muser.c\_str(), Mpassword.c\_str(), NULL, Mport, NULL, CLIENT\_MULTI\_STATEMENTS);

if(mysql\_query(desc, ("CALL my\_chat.GET\_USER\_ID('" + mono\_to\_di\_apostrophe(login) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

MYSQL\_RES \*query\_res = mysql\_store\_result(desc);

if (mysql\_num\_rows(query\_res)) res = true;

mysql\_free\_result(query\_res);

mysql\_close(desc);

return res;

}

void addUser(charPtr login, charPtr passwodr){

bool res = false;

UBD \*desc = mysql\_init(NULL);

mysql\_real\_connect(desc, Mhost.c\_str(), Muser.c\_str(), Mpassword.c\_str(), NULL, Mport, NULL, CLIENT\_MULTI\_STATEMENTS);

if(mysql\_query(desc, ("CALL my\_chat.ADDUSER('" + mono\_to\_di\_apostrophe(login) + "','" + mono\_to\_di\_apostrophe(passwodr) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

mysql\_close(desc);

}

//void dellUser(charPtr login){

//

//}

////отвечает за соединение, для отправки приватных сообщений

//void sender\_p\_m(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr friendID, message;

// my\_ulonglong n;

// MYSQL\_RES \*query\_res;

// while(true){

// //friendID - не id пользователя базы данных, а id записи в списке друзей

// sock >> friendID >> message;

// if(mysql\_query(desc, ("CALL SEND\_P\_M(" + friendID + ",'" + mono\_to\_di\_apostrophe(message) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n) throw MyException(1, "could not send the message.");

// else sock << answers[0];

// }

//}

//void sender\_c\_m(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr chatID, message;

// my\_ulonglong n;

// MYSQL\_RES \*query\_res;

// while(true){

// sock >> chatID >> message;

// if(mysql\_query(desc, ("CALL SEND\_C\_M(" + chatID + ",'" + mono\_to\_di\_apostrophe(message) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n) throw MyException(1, "could not send the message.");

// else sock << answers[0];

// }

//}

////если пользователю отправляется заявка не первый раз, то она обновится.

//void send\_request(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr userLOGIN, diuserLOGIN, message;

// my\_ulonglong n;

// MYSQL\_RES \*query\_res;

// while(true){

// sock >> userLOGIN >> message;

// diuserLOGIN = mono\_to\_di\_apostrophe(userLOGIN);

// if(mysql\_query(desc, ("CALL GET\_USER\_ID('" + diuserLOGIN + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n == 0){

// sock << answers[1] << warning + "there is no user with login:\n" + userLOGIN;

// continue;

// }

// if(mysql\_query(desc, ("CALL SEND\_REQUEST('" + diuserLOGIN +"', '" + mono\_to\_di\_apostrophe(message) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n){

// sock << answers[1] << warning + "user \" " + userLOGIN + " \" is your friend already.";

// continue;

// }

// else sock << answers[0];

// }

//}

//void accept\_request(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr login;

// my\_ulonglong n;

// MYSQL\_RES \*query\_res;

// while(true){

// sock >> login;

// if(mysql\_query(desc, ("CALL ACCEPT\_REQUEST(" + mono\_to\_di\_apostrophe(login) + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n) throw MyException(1, "could not accept the request.");

// else sock << answers[0];

// }

//}

//void creat\_chat(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr name\_of\_chat, DEFAULT;

// my\_ulonglong n;

// MYSQL\_RES \*query\_res;

// while(true){

// sock >> name\_of\_chat >> DEFAULT;

// if (DEFAULT.getLen() != 1 || (DEFAULT.getPtr()[0] != 'u' && DEFAULT.getPtr()[0] != 'g')) throw MyException(1, "could not accept the request.");

// if (name\_of\_chat.getLen() > 64){

// sock << answers[1] << warning + "maximal length of name for chat is 64.";

// continue;

// }

// if(mysql\_query(desc, ("CALL CREATE\_CHAT('" + mono\_to\_di\_apostrophe(name\_of\_chat) + "', '" + DEFAULT + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n) throw MyException(1, "could not accept the request.");

// else sock << answers[0];

// }

//}

//void add\_part(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr friendID, chatID;

// my\_ulonglong n;

// MYSQL\_RES \*query\_res;

// while(true){

// sock >> friendID >> chatID;

// if(mysql\_query(desc, ("CALL IS\_FRIEND\_IN\_CHAT(" + friendID + ", " + chatID + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n){

// sock << answers[1] << warning + "your friend have been in chat allready.";

// continue;

// }

// //выполнит ся при условии, что friendID - в друзьях и отправитель в chatID как минимум granted и что friendID не в chatID

// if(mysql\_query(desc, ("CALL ADD\_PART(" + friendID + ", " + chatID + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n) throw MyException(1, "could not accept the request.");

// else sock << answers[0];

// }

//}

//void set\_access(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr login, chatID, acs;

// my\_ulonglong n;

// MYSQL\_RES \*query\_res;

// while(true){

// sock >> login >> chatID >> acs;

// if(mysql\_query(desc, ("CALL SET\_ACCESS('" + mono\_to\_di\_apostrophe(login) + "', " + chatID + ", '" + acs + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n = mysql\_num\_rows(query\_res);

// mysql\_free\_result(query\_res);

// if (n) throw MyException(1, "could not accept the request.");

// else sock << answers[0];

// }

//}

//void online(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr tmp;

// MYSQL\_RES \*query\_res;

// MYSQL\_ROW row;

// while(true){

// sock >> tmp;

// if(mysql\_query(desc, "CALL ONLINE()") > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// row = mysql\_fetch\_row(query\_res);

// sock << answers[0] << row[0] << row[1] << row[2];

// mysql\_free\_result(query\_res);

// }

//}

//void get\_n\_m\_partners\_from(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// //если все ок - вернет логины участников чата и к каждому из них 1 - (незабанин) или 0 - (забанен)

// charPtr n, m, chatID; //n - начиная с какого (счет нач с 0), m - сколько

// charPtr like;//для поиска по имени (будет искать имя, что начинается на значение, что в like)

// MYSQL\_RES \*query\_res;

// my\_ulonglong n\_rows;

// MYSQL\_ROW row;

// while(true){

// sock >> n >> m >> like >> chatID;

// if(mysql\_query(desc, ("CALL GNM\_PART\_F\_CHAT(" + n + ", " + m + ", '" + mono\_to\_di\_apostrophe(like) + "%', " + chatID + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

// for (int i = 0; i < n\_rows; i++){

// row = mysql\_fetch\_row(query\_res);

// sock << di\_to\_mono\_apostrophe((const char\*)row[0]);

// if (charPtr((const char\*)row[1]) == charPtr("NULL")) sock << "0";

// else sock << "1";

// }

// mysql\_free\_result(query\_res);

// }

//}

//void get\_n\_m\_requests(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// //смотрит приглашения в друзь

// //если все ок - вернет логины и сообщение и сообщение к каждому из логинов

// charPtr n, m; //n - начиная с какого (счет нач с 0), m - сколько

// charPtr like;//для поиска по имени (будет искать имя, что начинается на значение, что в like)

// MYSQL\_RES \*query\_res;

// my\_ulonglong n\_rows;

// MYSQL\_ROW row;

// while(true){

// sock >> n >> m >> like;

// if(mysql\_query(desc, ("CALL GNM\_REQUESTS(" + n + ", " + m + ", '" + mono\_to\_di\_apostrophe(like) + "%')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

// for (int i = 0; i < n\_rows; i++){

// row = mysql\_fetch\_row(query\_res);

// sock << di\_to\_mono\_apostrophe((const char\*)row[0]) << di\_to\_mono\_apostrophe((const char\*)row[1]);

// }

// mysql\_free\_result(query\_res);

// }

//}

////p - false, c - true;

//void get\_n\_m\_messages(UBD \*desc, MySocketPtr sock, bool p\_or\_c) throw (MyException, MySocketException){

// //читает сообщения из конференции

// //если все ок - вернет количество сообщений, {id сообщения, сообщение, логин отправителя, время}

// charPtr n, m, ID; //n - начиная с какого (счет нач с 0), m - сколько

// MYSQL\_RES \*query\_res;

// my\_ulonglong n\_rows;

// MYSQL\_ROW row;

// while(true){

// sock >> n >> m >> ID;

// if (p\_or\_c){

// if(mysql\_query(desc, ("CALL GNM\_MESSAGES\_С(" + n + ", " + m + ", " + ID + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// } else {

// if(mysql\_query(desc, ("CALL GNM\_MESSAGES\_P(" + n + ", " + m + ", " + ID + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// }

// query\_res = mysql\_store\_result(desc);

// sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

// for (int i = 0; i < n\_rows; i++){

// row = mysql\_fetch\_row(query\_res);

// sock << row[0] << di\_to\_mono\_apostrophe((const char\*)row[1]) << di\_to\_mono\_apostrophe((const char\*)row[2]) << row[3];

// }

// mysql\_free\_result(query\_res);

// }

//}

////friends - false, confs - true;

//void get\_n\_m\_chats(UBD \*desc, MySocketPtr sock, bool friends\_or\_confs) throw (MyException, MySocketException){

// //возвращает список конференций или друзей

// //если все ок - вернет количество сообщений, {id чата, имя чата}

// charPtr n, m, ch; //n - начиная с какого (счет нач с 0), m - сколько

// charPtr like;//для поиска по имени (будет искать имя, что начинается на значение, что в like)

// MYSQL\_RES \*query\_res;

// my\_ulonglong n\_rows;

// MYSQL\_ROW row;

// while(true){

// sock >> n >> m >> like;

// if (friends\_or\_confs) ch = charPtr("c");

// else ch = charPtr("p");

// if(mysql\_query(desc, ("CALL GNM\_CHATS(" + n + ", " + m + ", '" + mono\_to\_di\_apostrophe(like) + "%', '" + ch + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

// for (int i = 0; i < n\_rows; i++){

// row = mysql\_fetch\_row(query\_res);

// sock << row[0] << di\_to\_mono\_apostrophe((const char\*)row[1]);

// }

// mysql\_free\_result(query\_res);

// }

//}

//void get\_n\_m\_users(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// //возвращает пользователей

// //если все ок - вернет количество сообщений, {логин пользователя}

// charPtr n, m; //n - начиная с какого (счет нач с 0), m - сколько

// charPtr like;//для поиска по имени (будет искать имя, что начинается на значение, что в like)

// MYSQL\_RES \*query\_res;

// my\_ulonglong n\_rows;

// MYSQL\_ROW row;

// while(true){

// sock >> n >> m >> like;

// if(mysql\_query(desc, ("CALL GNM\_USERS(" + n + ", " + m + ", '" + mono\_to\_di\_apostrophe(like) + "%')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

// for (int i = 0; i < n\_rows; i++){

// row = mysql\_fetch\_row(query\_res);

// sock << row[0];

// }

// mysql\_free\_result(query\_res);

// }

//}

//void get\_u\_data(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr login;

// MYSQL\_RES \*query\_res;

// my\_ulonglong n\_rows;

// while(true){

// sock >> login;

// if(mysql\_query(desc, ("CALL GET\_USER\_DATA('" + mono\_to\_di\_apostrophe(login) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n\_rows = mysql\_num\_rows(query\_res);

// if (n\_rows) sock << answers[0] << mysql\_fetch\_row(query\_res)[0];

// mysql\_free\_result(query\_res);

// if (n\_rows == 0) throw MyException(1, "could not get user data.");

// }

//}

//void get\_last\_u\_tick(UBD \*desc, MySocketPtr sock) throw (MyException, MySocketException){

// charPtr login;

// MYSQL\_RES \*query\_res;

// my\_ulonglong n\_rows;

// while(true){

// sock >> login;

// if(mysql\_query(desc, ("CALL GET\_LAST\_USER\_TICK('" + mono\_to\_di\_apostrophe(login) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

// query\_res = mysql\_store\_result(desc);

// n\_rows = mysql\_num\_rows(query\_res);

// if (n\_rows) sock << answers[0] << mysql\_fetch\_row(query\_res)[0];

// mysql\_free\_result(query\_res);

// if (n\_rows == 0) throw MyException(1, "could not get last user tick.");

// }

//}

#pragma endregion

//вернет тру, если в ptr есть символ из tabu\_chars.

bool tabu\_chars(charPtr ptr, charPtr tabu\_chars){

unsigned short n1 = ptr.getLen(), n2 = tabu\_chars.getLen();

char \*c1 = ptr.getPtr(), \*c2 = tabu\_chars.getPtr();

for (int i = 0; i < n1; i++){

for (int j = 0; j < n2; j++){

if (c1[i] == c2[j]) return true;

}

}

return false;

}

void init\_connection\_for\_work(UBD \*&desc, charPtr userID) throw (MyException) {

try{ if(mysql\_query(desc, "USE my\_chat") > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));//два раза, потому, что первый раз может не зайти

}catch(MyException ex){

if (ex.getErrorCode() != 2013){

mysql\_close(desc);

desc = mysql\_init(NULL);

mysql\_real\_connect(desc, Mhost.c\_str(), Muser.c\_str(), Mpassword.c\_str(), NULL, Mport, NULL, CLIENT\_MULTI\_STATEMENTS);// без флага CLIENT\_MULTI\_STATEMENTS не получится запускать хранимые процедуры

}

if(mysql\_query(desc, "USE my\_chat") > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

}

if(mysql\_query(desc, ("SET @userID := " + userID + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

}

//вернуть номер команды

//если такой команды нет - вернет количество команд

unsigned short get\_cmd\_num(charPtr ptr){

for(unsigned short i = 0; i < n\_commands; i++){

if (ptr == commands[i]) return i;

}

return n\_commands;

}

unsigned short get\_aut\_key\_num(charPtr ptr){

for(unsigned short i = 0; i < n\_aut\_keys; i++){

if (ptr == aut\_keys[i]) return i;

}

return n\_aut\_keys;

}

void for\_client(MySocket \*sock\_){

try{

void \*tmp1 = sock\_;

void \*tmp2 = sock\_;

MySocketPtr sock(sock\_);

//bool authorizated = false;//для проверки, авторизирован ли пользыватель (на этом сокете хоть раз - однозначно проверке на то, что добавлин ли сокет в socketBase)

//for(int i = 0; i < n\_commands; i++) sock << commands[i];

//for(int i = 0; i < n\_answers; i++) sock << answers[i];

charPtr ptr1, ptr2, ptr3, ptr4;//для логина, пароля, сообщений...

unsigned short for\_cmd;

while(true){

sock >> ptr1; //читаем команду

for\_cmd = get\_cmd\_num(ptr1);

if(for\_cmd == authorization || for\_cmd == registration){

#pragma region TheProgramm

sock >> ptr1 >> ptr2;

if (for\_cmd == authorization) sock >> ptr3;//так надо

//проверка длины

if (ptr1.getLen() > max\_login\_len || ptr2.getLen() > max\_password\_len) {

ptr3.init((char\*)nullptr, 0);

if (ptr1.getLen() > max\_login\_len) ptr3 = login\_len\_error;

if (ptr2.getLen() > max\_password\_len) ptr3 = ptr3 + password\_len\_error;

sock << answers[1] << error + ptr3;

continue;

}

//проверка на запрещенные символы

if(tabu\_chars(ptr1, charPtr("\0", 1)) || tabu\_chars(ptr2, charPtr("\0", 1))){

sock << answers[1] << error + "It is forbidden to use symbols \\0'\" for login or password.";

continue;

}

if (for\_cmd == authorization){

charPtr userID(authorize(ptr1, ptr2));

if (userID) {

unsigned short for\_key = get\_aut\_key\_num(ptr3);

if (for\_key == n\_aut\_keys){

sock << answers[1] << error + key + ptr3 + authorization\_key\_error;

continue;

}

sock << answers[0];

if (for\_key == check) break;

UBD \*desc = mysql\_init(NULL);

mysql\_real\_connect(desc, Mhost.c\_str(), Muser.c\_str(), Mpassword.c\_str(), NULL, Mport, NULL, CLIENT\_MULTI\_STATEMENTS);// без флага CLIENT\_MULTI\_STATEMENTS не получится запускать хранимые процедуры

#pragma region TheProgramm

try{

MYSQL\_RES \*query\_res;

my\_ulonglong n\_rows;

MYSQL\_ROW row;

charPtr ch;

char \*fail\_message = nullptr;

while(true){

sock >> ptr1;

init\_connection\_for\_work(desc, userID);

//init\_connection\_for\_work(desc, userID);

if (for\_key <= get\_last\_user\_tick){

#pragma region TheProgramm

///////////////////////////////////////////////////////////////////////////////////////

if (for\_key == sender\_private\_message || for\_key == sender\_chat\_message){

sock >> ptr2;//id , message

//ptr2 = insert\_first\_bifor\_second(ptr2, '\\', '\\');

if (for\_key == sender\_private\_message) ch.init("P", 1);

if (for\_key == sender\_chat\_message) ch.init("C", 1);

if(mysql\_query(desc, ("CALL SEND\_" + ch + "\_M(" + ptr1 + ",'" + mono\_to\_di\_apostrophe(ptr2) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

fail\_message = "could not send the message.";

} else if (for\_key == sender\_request){

sock >> ptr2;//userLOGIN , message

ptr3 = mono\_to\_di\_apostrophe(ptr1);

if(mysql\_query(desc, ("SELECT id FROM my\_chat.users WHERE log = '" + ptr3 + "'" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

n\_rows = mysql\_num\_rows(query\_res);

mysql\_free\_result(query\_res);

if (n\_rows == 0){ sock << answers[1] << warning + "there is no user with login:\n" + ptr1; continue;}

else {

init\_connection\_for\_work(desc, userID);

if(mysql\_query(desc, ("CALL SEND\_REQUEST('" + ptr3 +"', '" + mono\_to\_di\_apostrophe(ptr2) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

fail\_message = "user is your friend already.";

}

} else if(for\_key == accepter\_request){

//sock >> ptr1;//login

if(mysql\_query(desc, ("CALL ACCEPT\_REQUEST('" + mono\_to\_di\_apostrophe(ptr1) + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

fail\_message = "could not accept the request.";

} else if(for\_key == creater\_chat){

sock >> ptr2;//name\_of\_chat, DEFAULT

if (ptr2.getLen() != 1 || (ptr2.getPtr()[0] != 'u' && ptr2.getPtr()[0] != 'g')) throw MyException(1, "could not accept the request.");

if (ptr1.getLen() > 64){ sock << answers[1] << warning + "maximal length of name for chat is 64."; continue;}

else {

if(mysql\_query(desc, ("CALL CREATE\_CHAT('" + mono\_to\_di\_apostrophe(ptr1) + "', '" + ptr2 + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

fail\_message = "could not accept the request.";

}

} else if(for\_key == adder\_part){

sock >> ptr2;//friendID , chatID

if(mysql\_query(desc, ("CALL IS\_FRIEND\_IN\_CHAT(" + ptr1 + ", " + ptr2 + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

n\_rows = mysql\_num\_rows(query\_res);

mysql\_free\_result(query\_res);

if (n\_rows){ sock << answers[1] << warning + "your friend have been in chat allready."; continue;}

else {

init\_connection\_for\_work(desc, userID);

//выполнит ся при условии, что friendID - в друзьях и отправитель в chatID как минимум granted и что friendID не в chatID

if(mysql\_query(desc, ("CALL ADD\_PART(" + ptr1 + ", " + ptr2 + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

fail\_message = "could not accept the request.";

}

} else if(for\_key == line){

//sock >> tmp;

if(mysql\_query(desc, "SELECT unread\_private\_chats, unread\_chats, request FROM my\_chat.users WHERE id = @userID") > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

row = mysql\_fetch\_row(query\_res);

sock << answers[0] << row[0] << row[1] << row[2];

mysql\_free\_result(query\_res);

if(mysql\_query(desc, "UPDATE my\_chat.users SET unread\_private\_chats := FALSE, unread\_chats := FALSE, request := FALSE, last\_tick := NOW() WHERE id = @userID") > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

continue;

} else if(for\_key == seter\_access){

sock >> ptr2 >> ptr3;//login >> chatID >> acs;

if(mysql\_query(desc, ("CALL SET\_ACCESS('" + mono\_to\_di\_apostrophe(ptr1) + "', " + ptr2 + ", '" + ptr3 + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

fail\_message = "could not accept the request.";

} else if(for\_key == get\_user\_data || for\_key == get\_last\_user\_tick){

//sock >> login;

if (for\_key == get\_user\_data){if(mysql\_query(desc, ("SELECT status FROM my\_chat.users WHERE log = '" + mono\_to\_di\_apostrophe(ptr1) + "'" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));}

else if(mysql\_query(desc, ("SELECT last\_tick FROM my\_chat.users WHERE log = '" + mono\_to\_di\_apostrophe(ptr1) + "'" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

n\_rows = mysql\_num\_rows(query\_res);

if (n\_rows) sock << answers[0] << mysql\_fetch\_row(query\_res)[0];

mysql\_free\_result(query\_res);

if (n\_rows == 0){

if (for\_key == get\_user\_data) throw MyException(1, "could not get user data.");

else throw MyException(1, "could not get last user tick.");

}

continue;

}

if (query\_res = mysql\_store\_result(desc)){

n\_rows = mysql\_num\_rows(query\_res);

mysql\_free\_result(query\_res);

if (n\_rows) throw MyException(1, fail\_message);

else sock << answers[0];

} else sock << answers[0];

continue;

///////////////////////////////////////////////////////////////////////////////////////

#pragma endregion

}

#pragma region TheProgramm

if(for\_key == geter\_freand\_unread\_last\_tick || for\_key == geter\_conf\_unread\_last\_tick){

//вернет ({0; 1}, время последнего сообщения)

//sock >> ID;

if (for\_key == geter\_conf\_unread\_last\_tick) ch = charPtr("c");

else ch = charPtr("p");

if(mysql\_query(desc, ("CALL GET\_UNREAD\_LAST\_TICK(" + ptr1 + ", '" + ch + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

n\_rows = mysql\_num\_rows(query\_res);

if (n\_rows){

row = mysql\_fetch\_row(query\_res);

sock << answers[0] << row[0] << row[1];

}

else{

mysql\_free\_result(query\_res);

throw MyException(1, "could not get unread and last\_tick.");

}

mysql\_free\_result(query\_res);

continue;

}

if(for\_key == get\_friend\_id){

//вернет ({0; 1}, id друга в списке друзей (если у вас нет такого друга, то вернет 0, а если такого пользователя нет - вернет 1 и сообщение об этом))

//sock >> ID;

if(mysql\_query(desc, ("CALL GET\_FRIEND\_ID(" + ptr1 + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

n\_rows = mysql\_num\_rows(query\_res);

row = mysql\_fetch\_row(query\_res);

if (n\_rows) sock << answers[0] << row[0];

else{

mysql\_free\_result(query\_res);

throw MyException(1, ("User ' " + ptr2 + " ' is not exsists." + null\_symbol).getPtr());

}

mysql\_free\_result(query\_res);

continue;

}

sock >> ptr2 >> ptr3;

ptr3 = insert\_first\_bifor\_second(ptr3, '\\', '\_');

ptr3 = insert\_first\_bifor\_second(ptr3, '\\', '%');

//ptr3 = insert\_first\_bifor\_second(ptr3, '\\', '\\');

if(for\_key == geter\_n\_m\_partners\_from){

//если все ок - вернет логины участников чата и к каждому из них 1 - (незабанин) или 0 - (забанен)

//charPtr n, m, chatID; //n - начиная с какого (счет нач с 0), m - сколько

//charPtr like;//для поиска по имени (будет искать имя, что начинается на значение, что в like)

sock >> ptr4;//sock >> n >> m >> like >> chatID;

if(mysql\_query(desc, ("CALL GNM\_PART\_F\_CHAT(" + ptr1 + ", " + ptr2 + ", '" + mono\_to\_di\_apostrophe(ptr3) + "%', " + ptr4 + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

for (int i = 0; i < n\_rows; i++){

row = mysql\_fetch\_row(query\_res);

sock << di\_to\_mono\_apostrophe((const char\*)row[0]);

if (charPtr((const char\*)row[1]) == charPtr("NULL")) sock << "0";

else sock << "1";

}

mysql\_free\_result(query\_res);

}

if(for\_key == geter\_n\_m\_requests){

//смотрит приглашения в друзь

//если все ок - вернет логины и сообщение к каждому из логинов

//charPtr n, m; //n - начиная с какого (счет нач с 0), m - сколько

//charPtr like;//для поиска по имени (будет искать имя, что начинается на значение, что в like)

//>> n >> m >> like

if(mysql\_query(desc, ("SELECT U.log, M.message FROM my\_chat.r" + userID + " AS M, my\_chat.users AS U WHERE U.log LIKE '" + mono\_to\_di\_apostrophe(ptr3) + "%' AND U.id = M.user\_id ORDER BY M.last\_tick DESC LIMIT " + ptr1 + ", " + ptr2 + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

for (int i = 0; i < n\_rows; i++){

row = mysql\_fetch\_row(query\_res);

sock << di\_to\_mono\_apostrophe((const char\*)row[0]) << di\_to\_mono\_apostrophe((const char\*)row[1]);

}

mysql\_free\_result(query\_res);

}

if(for\_key == geter\_n\_m\_messages\_p || for\_key == geter\_n\_m\_messages\_c){

//читает сообщения из конференции

//если все ок - вернет количество сообщений, {id сообщения, сообщение, логин отправителя, время}

//charPtr n, m, ID; //n - начиная с какого (счет нач с 0), m - сколько

//sock >> n >> m >> ID;

if (for\_key == geter\_n\_m\_messages\_c){

if(mysql\_query(desc, ("CALL GNM\_MESSAGES\_С(" + ptr1 + ", " + ptr2 + ", " + ptr3 + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

}

else if(mysql\_query(desc, ("CALL GNM\_MESSAGES\_P(" + ptr1 + ", " + ptr2 + ", " + ptr3 + ")" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

for (int i = 0; i < n\_rows; i++){

row = mysql\_fetch\_row(query\_res);

sock << row[0] << di\_to\_mono\_apostrophe((const char\*)row[2]) << di\_to\_mono\_apostrophe((const char\*)row[1]) << row[3];

}

mysql\_free\_result(query\_res);

}

if (for\_key == geter\_n\_m\_users){

//возвращает список пользовтелей, которые не друзья и не делали заявки в друзья данному пользователю

//если все ок - вернет количество сообщений, {имя пользователя}

//charPtr n, m, ch; //n - начиная с какого (счет нач с 0), m - сколько

//charPtr like;//для поиска по имени (будет искать имя, что начинается на значение, что в like)

//sock >> n >> m >> like;

if(mysql\_query(desc, ("CALL GNM\_USERS(" + ptr1 + ", " + ptr2 + ", '" + mono\_to\_di\_apostrophe(ptr3) + "%')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

for (int i = 0; i < n\_rows; i++) sock << di\_to\_mono\_apostrophe((const char\*)mysql\_fetch\_row(query\_res)[0]);

mysql\_free\_result(query\_res);

}

if(for\_key == geter\_n\_m\_friens || for\_key == geter\_n\_m\_confs){

//возвращает список конференций или друзей

//если все ок - вернет количество сообщений, {id чата, имя чата}

//charPtr n, m, ch; //n - начиная с какого (счет нач с 0), m - сколько

//charPtr like;//для поиска по имени (будет искать имя, что начинается на значение, что в like)

//sock >> n >> m >> like;

if (for\_key == geter\_n\_m\_confs) ch = charPtr("c");

else ch = charPtr("p");

if(mysql\_query(desc, ("CALL GNM\_CHATS(" + ptr1 + ", " + ptr2 + ", '" + mono\_to\_di\_apostrophe(ptr3) + "%', '" + ch + "')" + null\_symbol).getPtr()) > 0) throw MyException(mysql\_errno(desc), mysql\_error(desc));

query\_res = mysql\_store\_result(desc);

sock << answers[0] << to\_string(n\_rows = mysql\_num\_rows(query\_res)).c\_str();

for (int i = 0; i < n\_rows; i++){

row = mysql\_fetch\_row(query\_res);

sock << row[0] << di\_to\_mono\_apostrophe((const char\*)row[1]);

}

mysql\_free\_result(query\_res);

//

//query\_res = mysql\_store\_result(desc);

//mysql\_free\_result(query\_res);

//

}

#pragma endregion

}

} catch(MySocketException) {

mysql\_close(desc); // Закрываем соединение

} catch(MyException ex) {

sock << answers[1] << ex.toString().c\_str();

mysql\_close(desc); // Закрываем соединение

}

#pragma endregion

return;

} else sock << answers[1] << error + authorization\_error;

} else {

if (!existUser(ptr1)) {

addUser(ptr1, ptr2);

sock << answers[0];

} else sock << answers[1] << error + (user + ptr1 + registration\_error);

}

continue;

#pragma endregion

} else {

sock << answers[1] << error + (ptr1 + unknown\_command);

continue;

}

}

} catch(MyException e) {

//if (e.getErrorCode() != WSAECONNRESET) throw;

//cerr << (string)e << endl << '\a';

}

}

int main(int argc, char \*argv[]){

#pragma region Connecting to DATDBASE

MD = mysql\_init(NULL);

if(MD == nullptr){

cerr << error << "can not create MYSQL descriptor.\n";

return 1;

}

string port\_s;

if (argc != 5){

cout << "Enter host:\n"; cin >> Mhost;

cout << "Enter port:\n"; cin >> port\_s;

cout << "Enter login:\n"; cin >> Muser;

cout << "Enter password:\n"; cin >> Mpassword;

} else { Mhost = argv[1]; port\_s = argv[2]; Muser = argv[3]; Mpassword = argv[4]; }

try{

Mport = atoi(port\_s.c\_str());

if (atoi(port\_s.c\_str()) < 0 || (Mport == 0 && port\_s != "0")) throw 2;

}

catch(...){

cerr << error << "bad port.\n";

return 2;

}

if( !mysql\_real\_connect(MD, Mhost.c\_str(), Muser.c\_str(), Mpassword.c\_str(), NULL, Mport, NULL, CLIENT\_MULTI\_STATEMENTS) ){

cerr << error << mysql\_error(MD) << endl;

return 3;

} else {

if(mysql\_query(MD, "USE my\_chat") > 0){

cerr << error << mysql\_error(MD) << endl;

return 4;

}

cout << "connected.\n";

}

#pragma endregion

try{

MySocketPtr sock(new MyWinSocket(SOCK\_STREAM));

sock->bind((MySocketAddress\*)new MyWinSocketAddress("", 30000));

sock->listen(SOMAXCONN);

while (true) CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)for\_client, (LPVOID)sock->accept(), NULL, NULL);

}

catch(MyException ex){ cerr << ex.toString(); }

catch(...){

cerr << error << "server crashed.\n";

return 10;

}

//mysql\_close(MD);

return 0;

}

#pragma endregion

## Код, що описуе інтерфейс

### AdderFriend.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class AdderFriend : Form

{

User u;

public AdderFriend(User u)

{

InitializeComponent();

this.u = u.copy();

this.Controls.Add(this.u);

}

void add() {

Waiter w = new Waiter(ref Central.counter[(int)Central.key.sender\_request, 0], ref Central.counter[(int)Central.key.sender\_request, 1], new method\_empty(this.add\_));

}

void add\_(){

int k = (int)Central.key.sender\_request;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.write(u.Name, Central.socks[k]);

Central.write(richTextBox1.Text, Central.socks[k]);

if (Central.read(Central.socks[k]) != Central.answers[0])

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

else {

//Central.searching\_panel.Controls.RemoveByKey(u.Name);

this.Close();

}

}

bool shiftDown;

delegate void no\_params();

private void richTextBox1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Shift || e.KeyCode == Keys.ShiftKey) shiftDown = true;

if (e.KeyCode == Keys.Enter && !shiftDown) { add(); }

}

private void richTextBox1\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Shift || e.KeyCode == Keys.ShiftKey) shiftDown = false;

}

private void richTextBox1\_Leave(object sender, EventArgs e)

{

shiftDown = false;

}

private void button1\_Click(object sender, EventArgs ev)

{

add();

}

}

}

### AdderFriend.Designer.cs

namespace chat\_client

{

partial class AdderFriend

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.richTextBox1 = new System.Windows.Forms.RichTextBox();

this.button1 = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// richTextBox1

//

this.richTextBox1.Location = new System.Drawing.Point(-1, 139);

this.richTextBox1.Name = "richTextBox1";

this.richTextBox1.Size = new System.Drawing.Size(205, 96);

this.richTextBox1.TabIndex = 0;

this.richTextBox1.Text = "";

this.richTextBox1.KeyDown += new System.Windows.Forms.KeyEventHandler(this.richTextBox1\_KeyDown);

this.richTextBox1.KeyUp += new System.Windows.Forms.KeyEventHandler(this.richTextBox1\_KeyUp);

this.richTextBox1.Leave += new System.EventHandler(this.richTextBox1\_Leave);

//

// button1

//

this.button1.Location = new System.Drawing.Point(210, 139);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(95, 96);

this.button1.TabIndex = 1;

this.button1.Text = "ADD";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// AdderFriend

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.DarkCyan;

this.ClientSize = new System.Drawing.Size(310, 240);

this.Controls.Add(this.button1);

this.Controls.Add(this.richTextBox1);

this.Name = "AdderFriend";

this.Text = "AdderFriend";

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.RichTextBox richTextBox1;

private System.Windows.Forms.Button button1;

}

}

### Button\_more.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class Button\_more : UserControl

{

method do\_some;

public Button\_more(method do\_)

{

InitializeComponent();

do\_some = do\_;

}

private void more\_Click(object sender, EventArgs e)

{

do\_some(sender, e);

}

}

}

### Button\_more.Designer.cs

namespace chat\_client

{

partial class Button\_more

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором компонентов

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.more = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// more

//

this.more.Anchor = System.Windows.Forms.AnchorStyles.None;

this.more.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F);

this.more.ImeMode = System.Windows.Forms.ImeMode.NoControl;

this.more.Location = new System.Drawing.Point(0, 1);

this.more.Name = "more";

this.more.Size = new System.Drawing.Size(75, 30);

this.more.TabIndex = 17;

this.more.Text = "Еще";

this.more.UseVisualStyleBackColor = true;

this.more.Click += new System.EventHandler(this.more\_Click);

//

// Button\_more

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.Transparent;

this.Controls.Add(this.more);

this.MaximumSize = new System.Drawing.Size(0, 32);

this.MinimumSize = new System.Drawing.Size(75, 32);

this.Name = "Button\_more";

this.Size = new System.Drawing.Size(75, 32);

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.Button more;

}

}

### Central.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.Threading;

using System.Net.Sockets;

using System.Windows.Forms;

namespace chat\_client

{

public static class Central

{

public static method delegate\_load\_more\_to = new method(Central.load\_more\_to);

[STAThread]

public static void Main() {

Init();

Chat.ShowDialog();

}

//чтоб проверять впервые ли запущ прогр.

static private bool first = true;

public static void Init() {

if (first)

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

//Application.Run(new Form1());

first = false;

}

Central.closing = false;

Central.counter = new UInt64[Central.n\_keys, 2];

Central.socks = new Socket[Central.n\_keys];

Central.threads = new List<Thread>();

Form2 form = new Form2("User params.", "Login", "Password", Central.commands, Central.answers, Central.keys[0]);

form.ShowDialog();

if (!form.aut) Environment.Exit(0);

friends = new ItemList\_with\_Button\_more(delegate\_load\_more\_to, DockStyle.Fill, DockStyle.Bottom);

friends.Tag = (object)Central.key.geter\_n\_m\_friens;

conferences = new ItemList\_with\_Button\_more(delegate\_load\_more\_to, DockStyle.Fill, DockStyle.Bottom);

conferences.Tag = (object)Central.key.geter\_n\_m\_confs;

searching\_panel = new ItemList\_with\_Button\_more(delegate\_load\_more\_to, DockStyle.Fill, DockStyle.Bottom);

searching\_panel.Tag = (object)Central.key.geter\_n\_m\_users;

requests = new ItemList\_with\_Button\_more(delegate\_load\_more\_to, DockStyle.Fill, DockStyle.Bottom);

requests.Tag = (object)Central.key.geter\_n\_m\_requests;

Conference\_fields.Conferences = new Dictionary<string, Conference\_fields>();

User\_fields.Users = new Dictionary<string, User\_fields>();

Central.log = form.getParam1();

Central.pass = form.getParam2();

Chat = new Form1();

refresh\_data();

Thread online\_ = new System.Threading.Thread(new System.Threading.ThreadStart(Central.online));

Central.threads.Add(online\_);

Thread online\_checker\_ = new System.Threading.Thread(new System.Threading.ThreadStart(Central.online\_checker));

Central.threads.Add(online\_checker\_);

Thread message\_loader\_ = new System.Threading.Thread(new System.Threading.ThreadStart(Central.message\_loader));

Central.threads.Add(message\_loader\_);

online\_.Start();

online\_checker\_.Start();

message\_loader\_.Start();

}

public static void Close(){

if (Central.socks != null)

{

Central.closing = true;

for (int i = 0; i < Central.socks.Length; i++)

{

if (Central.socks[i] != null) Central.socks[i].Close();

Central.socks[i] = null;

}

}

if (Central.threads != null)

{

foreach (Thread t in Central.threads) { t.Abort(); }

Central.threads.Clear();

}

Central.threads = null;

Central.socks = null;

friends.Clear();

conferences.Clear();

searching\_panel.Clear();

requests.Clear();

User\_fields.Users.Clear();

Conference\_fields.Conferences.Clear();

}

//public static delegate string[] int\_parsms\_string(int k, int n\_read, params string[] s);

//public static int\_parsms\_string for\_write\_and\_read = new int\_parsms\_string(write\_and\_read);

////пишет и читает данны по к-тому сокету в порядке очереди запросов;

////n\_read - еоличество строк, что надо прочесть

//public static string[] write\_and\_read(int k, int n\_read, params string[] s) {

// while (counter[k, 0] != counter[k, 1]) { System.Threading.Thread.Sleep(5); }

// if (socks[k] == null) set\_right(k);

// ++counter[k, 0];

// for (int i = 0; i < s.Length; i++) write(s[i], socks[k]);

// string[] res = new string[n\_read];

// for (int i = 0; i < n\_read; i++) res[i] = read(socks[k]);

// ++counter[k, 1];

// return res;

//}

public static void start\_dialog(int k)

{

while (counter[k, 0] != counter[k, 1]) { System.Threading.Thread.Sleep(5); }

++counter[k, 0];

}

public static void stop\_dialog(int k) { ++counter[k, 1]; }

public const int n\_commands = 2, n\_answers = 2, n\_keys = 21;

static public string[] commands = { "aut", "reg" }, answers = { "0", "1" }, keys = { "check", "smp", "smc", "sr", "ar", "cc", "ap", "line", "sa", "gud", "glut", "gnmp", "gnmr", "gnmmp", "gnmc", "gnmf", "gnmco", "gnmu", "gful", "gcul", "gfid"};//массив ключей команды авторизации

public enum key { check, sender\_private\_message, sender\_chat\_message, sender\_request, accepter\_request, creater\_chat, adder\_part, line, seter\_access, get\_user\_data, get\_last\_user\_tick, geter\_n\_m\_partners\_from, geter\_n\_m\_requests, geter\_n\_m\_messages\_p, geter\_n\_m\_messages\_c, geter\_n\_m\_friens, geter\_n\_m\_confs, geter\_n\_m\_users, geter\_freand\_unread\_last\_tick, geter\_conf\_unread\_last\_tick, get\_friend\_id };

public const ushort max\_message\_len = 255 \* 257;

static public string log, pass;

public static ItemList\_with\_Button\_more friends, conferences, searching\_panel, requests;

public static void load\_more\_to(object sender, EventArgs e){

ItemList\_with\_Button\_more list = (ItemList\_with\_Button\_more)sender;

int k = (int)list.Tag;

if (socks[k] == null) set\_right(k);

if (socks[k] == null) Fatal\_Error();

start\_dialog(k);

write("0", Central.socks[k]);

write((list.get\_n\_visible\_children() - 1 + 10).ToString(), Central.socks[k]);

write(Chat.textBox1.Text, Central.socks[k]);

//if (list == conferences || list == friends) Central.write("0", Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

int n = Convert.ToInt32(Central.read(Central.socks[k]));

if (list == friends) for (int i = 0; i < n; i++) User\_fields.getUser\_fields(Central.read(Central.socks[k]), Central.read(Central.socks[k]));

else if (list == conferences)

for (int i = 0; i < n; i++)

{

}

else {

User[] variable = new User[n];

if (list == searching\_panel) for (int i = n - 1; i >= 0; i--) variable[i] = User\_fields.getUser\_fields(null, Central.read(Central.socks[k])).getUser();

else if (list == requests){

for (int i = n - 1; i >= 0; i--){

variable[i] = User\_fields.getUser\_fields(null, Central.read(Central.socks[k])).getUser();

variable[i].Tag = Central.read(Central.socks[k]);

}

}

list.Clear();

list.Controls.AddRange(variable);

}

}

else{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

stop\_dialog(k);

}

public static void refresh\_data()

{

//this.Controls.SetChildIndex - испозьзовать!

Central.no\_selected();

load\_accaunt\_data();

Central.load\_more\_to(Central.friends, null);

Central.load\_more\_to(Central.conferences, null);

}

public static string get\_user\_status(string name)

{

int k = (int)Central.key.get\_user\_data;

Waiter w = new Waiter(ref Central.counter[k, 0], ref Central.counter[k, 1], new method\_obj1(Central.get\_user\_status\_), name);

return (string)w.res;

}

public static object get\_user\_status\_(object name)

{

int k = (int)Central.key.get\_user\_data;

if (Central.socks[k] == null) Central.socks[k] = Central.make\_authorised\_sock(k);

if (Central.socks[k] == null) return null;

Central.write((string)name, Central.socks[k]);

return Central.read(Central.socks[k]);

}

//для развития программы (авки загрузить,если сделаю)

public static void load\_accaunt\_data() { }

public static void no\_selected()

{

Central.selected\_num = 0;

Central.selected\_obj = null;

Chat.panel6.Controls.Clear();

Chat.panel7.Enabled = false;

}

public static void ShowMessages(User\_fields user)

{

if (user.ID != null){

bool exec = false;

if (selected\_num != 1) exec = true;

else if(user.Name != ((User\_fields)selected\_obj).Name) exec = true;

if (exec) {

selected\_obj = user;

selected\_num = 1;

Chat.panel6.Controls.Clear();

Chat.panel6.Controls.Add(user.messages);

Chat.panel6.Controls.Add(user.getUser());

if (user.messages.Controls.Count <= 1) { user.load\_more\_messages(); }//сделать!

Chat.panel7.Enabled = true;

}

}

else no\_selected();

}

public static void ShowMessages(Conference\_fields conf)

{

if (conf.ID != null){

bool exec = false;

if (selected\_num != 2) exec = true;

else if(conf.ID != ((Conference\_fields)selected\_obj).ID) exec = true;

if (exec) {

selected\_obj = conf;

selected\_num = 2;

Chat.panel6.Controls.Clear();

Chat.panel6.Controls.Add(conf.messages);

Chat.panel6.Controls.Add(conf.getConference());

if (conf.messages.Controls.Count <= 1) { conf.load\_more\_messages(); }//сделать!

Chat.panel7.Enabled = true;

}

}

else no\_selected();

}

public static void insert\_friend(User\_fields fields) {

if (fields.ID != null)

{

if (!friends.Controls.ContainsKey(fields.Name))

{

User u = fields.getUser();

friends.Controls.Add(u);

u.Last\_tickChanged();//чтоб поставить на нужное место

u.ContextMenuStrip = Chat.contextMenuStrip1FRIEND;

requests.Controls.RemoveByKey(fields.Name);

searching\_panel.Controls.RemoveByKey(fields.Name);

}

}

}

public static void insert\_conference(Conference\_fields fields) {

if (!conferences.Controls.ContainsKey(fields.Name))

{

Conference c = fields.getConference();

conferences.Controls.Add(c);

c.Last\_tickChanged();//чтоб поставить на нужное место

}

}

static public Form1 Chat;

static public Socket[] socks;

public static List<System.Threading.Thread> threads = new List<System.Threading.Thread>();

public static Thread send\_mess\_in\_new\_thread;

//для выполнения взятия результата в нужном порядке

static public UInt64[,] counter;

static public bool closing;//хранит значение true, если сейчас должен произойти разрыв связи с сервером.(чтоб эксепшн не вылетал)

//востанавливает сокет под номером k

public static void set\_right(int k){

socks[k] = make\_authorised\_sock(k);

}

public static Socket make\_authorised\_sock(int k)

{

Socket sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

connect(sock);

write(commands[0], sock);

write(log, sock);

write(pass, sock);

write(keys[(int)k], sock);

if (read(sock) != answers[0])

{

MessageBox.Show(read(sock), "Message from server.");

return null;

}

return sock;

}

public static int still\_online = 2;//количество минут

public static void online\_checker(){

Thread.Sleep(200);

int k = (int)Central.key.get\_last\_user\_tick;

DateTime now = new DateTime();

while(true){

Central.start\_dialog(k);

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.write(Central.log, Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0]) now = DateTime.Parse(Central.read(Central.socks[k]));

else{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

Central.stop\_dialog(k);

Thread.Sleep(100000);//100 sec

continue;

}

Central.stop\_dialog(k);

var ref\_ = User\_fields.Users.Values.GetEnumerator();

ref\_.MoveNext();

int count = User\_fields.Users.Values.Count;

try { for (int i = 0; i < count; i++, ref\_.MoveNext()) { ref\_.Current.online\_checker(now, still\_online); } }

catch (Exception) {}

Thread.Sleep(100000);//100 sec

}

}

//отсылает сообщение текущему собеседнику

public static void send\_mess(string mess)

{

int k = 0;

string ID = "";

if (mess == "" || selected\_obj != null && (selected\_num == 1 || selected\_num == 2))

{

if (Central.selected\_num == 1)

{

k = (int)Central.key.sender\_private\_message;

ID = ((User\_fields)selected\_obj).ID;

}

else if (Central.selected\_num == 2)

{

k = (int)Central.key.sender\_chat\_message;

ID = ((Conference\_fields)selected\_obj).ID;

}

}

else { return; }

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.start\_dialog(k);

Central.write(ID, Central.socks[k]);

Central.write(mess, Central.socks[k]);

if (Central.read(Central.socks[k]) != Central.answers[0])

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Central.stop\_dialog(k);

}

public static void send\_mess(object mess)

{

send\_mess((string)mess);

}

public static void there\_is\_unreed\_user()

{

var ref\_ = User\_fields.Users.Values.GetEnumerator();

ref\_.MoveNext();

int count = User\_fields.Users.Values.Count;

try { for (int i = 0; i < count; i++, ref\_.MoveNext()) { ref\_.Current.unread\_and\_last\_tick\_checker(); } }

catch (Exception) { }

}

public static void there\_is\_anreed\_conference()

{

var ref\_ = Conference\_fields.Conferences.Values.GetEnumerator();

ref\_.MoveNext();

int count = Conference\_fields.Conferences.Values.Count;

try { for (int i = 0; i < count; i++, ref\_.MoveNext()) { ref\_.Current.unread\_and\_last\_tick\_checker(); } }

catch (Exception) { }

}

public static void there\_is\_anreed\_request()

{

load\_more\_to(requests, null);

}

public static void online()

{

Thread.Sleep(200);

int k = (int)Central.key.line;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

while (true)

{

Central.write("", Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

if (Central.read(Central.socks[k]) == "1") Chat.panel6.BeginInvoke(new method\_empty(there\_is\_unreed\_user));

if (Central.read(Central.socks[k]) == "1") Chat.panel6.BeginInvoke(new method\_empty(there\_is\_anreed\_conference));

if (Central.read(Central.socks[k]) == "1") Chat.panel6.BeginInvoke(new method\_empty(there\_is\_anreed\_request));

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Thread.Sleep(500);

}

}

delegate void method\_one\_obj\_param(object obj);

delegate void method\_two\_obj\_param(object obj1, object obj2);

delegate void method\_four\_obj\_param(object obj1, object obj2, object obj3, object obj4);

public static void User\_fields\_set\_Unread(object fields, object bool\_var)

{

((User\_fields)fields).Unread = (bool)bool\_var;

}

public static void Conference\_fields\_set\_Unread(object fields, object bool\_var)

{

((Conference\_fields)fields).Unread = (bool)bool\_var;

}

public static void message\_loader() {

int k = (int)Central.key.geter\_n\_m\_messages\_p;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

k = (int)Central.key.geter\_n\_m\_messages\_c;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

string ID = null;

int n\_messages = 2; //количество сообщени которые можно читать одним блоком

ItemList\_with\_Button\_more messages = null;

object selected = null;

method\_two\_obj\_param User\_fields\_set\_Unread\_delegate = new method\_two\_obj\_param(User\_fields\_set\_Unread);

method\_two\_obj\_param Conference\_fields\_set\_Unread\_delegate = new method\_two\_obj\_param(Conference\_fields\_set\_Unread);

while (true)

{

Thread.Sleep(100);

if (selected\_num == 0 || selected\_obj == null) continue;

selected = selected\_obj;

if (selected\_num == 1){

k = (int)Central.key.geter\_n\_m\_messages\_p;

messages = ((User\_fields)selected).messages;

ID = ((User\_fields)selected).ID;

}

else if (selected\_num == 2){

k = (int)Central.key.geter\_n\_m\_messages\_c;

messages = ((Conference\_fields)selected).messages;

ID = ((Conference\_fields)selected).ID;

}

Central.start\_dialog(k);

bool enought = false;

int numb = 0;

while (!enought){

Central.write((numb).ToString(), Central.socks[k]);

Central.write((numb = numb + n\_messages).ToString(), Central.socks[k]);

Central.write(ID, Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

int n = Convert.ToInt32(Central.read(Central.socks[k]));

Message[] messages\_ = new Message[n];

string message\_ID, message, login, time;

for (int i = 0; i < n; i++)

{

message\_ID = Central.read(Central.socks[k]);

message = Central.read(Central.socks[k]);

login = Central.read(Central.socks[k]);

time = Central.read(Central.socks[k]);

if (!messages.Controls.ContainsKey(message\_ID)){

if (k == (int)Central.key.geter\_n\_m\_messages\_p) Chat.BeginInvoke(new method\_four\_obj\_param(((User\_fields)selected).insert\_message), message\_ID, time, message, login);

else if (k == (int)Central.key.geter\_n\_m\_messages\_c) Chat.BeginInvoke(new method\_four\_obj\_param(((Conference\_fields)selected).insert\_message), message\_ID, time, message, login);

}

else {

i++;

for (; i < n; i++){

Central.read(Central.socks[k]);

Central.read(Central.socks[k]);

Central.read(Central.socks[k]);

Central.read(Central.socks[k]);

}

enought = true;

break;

}

}

if (n < n\_messages) enought = true;

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

}

Central.stop\_dialog(k);

if (k == (int)Central.key.geter\_n\_m\_messages\_p) Chat.BeginInvoke(User\_fields\_set\_Unread\_delegate, selected, false);

else if (k == (int)Central.key.geter\_n\_m\_messages\_c) Chat.BeginInvoke(Conference\_fields\_set\_Unread\_delegate, selected, false);

}

}

public static object selected\_obj;

public static int selected\_num = 0; //0 - nothing, 1 - User\_fields, 2 - Conference\_fields

public static void Fatal\_Error(string s)

{

MessageBox.Show(s, "ERROR.");

Environment.Exit(1);

}

public static void Fatal\_Error()

{

Fatal\_Error("Программа не может работать.");

}

public static void write(string s, Socket sock)

{

try

{

write(Encoding.UTF8.GetBytes(s), sock);

}

catch (Exception ex) {

if (!closing)

{

MessageBox.Show(ex.ToString(), "ERROR.");

Environment.Exit(4);

}

}

}

public static void write(byte[] buf, Socket sock)

{

if (buf.Length > max\_message\_len) throw new Exception("Could not send so long message.\nYour message weighs " + buf.Length.ToString() + " bytes, which is greater than the maximum value " + max\_message\_len.ToString() + ".");

byte[] len = { (byte)(buf.Length / 256), (byte)(buf.Length % 256) };

try

{

sock.Send(len, 2, SocketFlags.None);

if (buf.Length != 0) sock.Send(buf, buf.Length, SocketFlags.None);

}

catch (Exception) {

if (!closing) throw;

}

}

public static string read(Socket sock)

{

try

{

return Encoding.UTF8.GetString(readBytes(sock));

}

catch (Exception ex) {

if (!closing) {

MessageBox.Show(ex.ToString(), "ERROR.");

Environment.Exit(5);

}

return null;

}

}

public static byte[] readBytes(Socket sock)

{

byte[] buf = new byte[2];

try

{

sock.Receive(buf, 2, SocketFlags.None);

buf = new byte[buf[0] \* 256 + buf[1]];

if (buf.Length != 0) sock.Receive(buf, buf.Length, SocketFlags.None);

return buf;

}

catch (Exception) {

throw;

}

}

////////////////////////////////////////////////////////////////

//для ip, port

const string IPEnvironmentVariable = "IP\_of\_chat\_server";

const string PortEnvironmentVariable = "Port\_of\_chat\_server";

public static void connect(Socket sock)

{

string ip = Environment.GetEnvironmentVariable(IPEnvironmentVariable);

if (ip == null)

{

MessageBox.Show("Could not get environment variable \"" + IPEnvironmentVariable + "\".", "ERROR.");

Environment.Exit(1);

}

string port = Environment.GetEnvironmentVariable(PortEnvironmentVariable);

if (port == null)

{

MessageBox.Show("Could not get environment variable \"" + PortEnvironmentVariable + "\".", "ERROR.");

Environment.Exit(2);

}

try

{

sock.Connect(ip, Convert.ToUInt16(port));

}

catch (Exception e)

{

MessageBox.Show(e.ToString(), "ERROR.");

Environment.Exit(3);

}

}

}

}

### Conference.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class Conference : UserControl{

public Conference\_fields fields;

public bool Unread{

get {

if (panel1.BackColor == SystemColors.ControlLight) return false;

else return true;

}

set {

if (value) panel1.BackColor = SystemColors.ControlDark;

else panel1.BackColor = SystemColors.ControlLight;

}

}

public Conference(string id, string name, ContextMenuStrip contextMenuStrip){

InitializeComponent();

this.Dock = DockStyle.Top;

this.fields = Conference\_fields.getConference\_fields(id, name);

this.ContextMenuStrip = contextMenuStrip;

this.Unread = this.fields.Unread;

this.Name = this.fields.ID;

this.name.Text = this.fields.Name;

this.fields.UnreadChanged.Add(new method\_empty(this.UnreadChanged));

this.fields.Last\_tickChanged.Add(new method\_empty(this.Last\_tickChanged));

}

public virtual Conference copy() {

return new Conference(fields.ID, fields.Name, this.ContextMenuStrip);

}

protected virtual void Contact\_Click(object sender, EventArgs e){

Central.ShowMessages(this.fields);

}

public void UnreadChanged() {

this.Unread = fields.Unread;

}

public void Last\_tickChanged(){

if (this.Parent != null)

{

int n = this.Parent.Controls.Count;

DateTime this\_last\_tick = DateTime.Parse(this.fields.Last\_tick);

for (int i = 0; i < n; i++)

{

try

{

Conference C = (Conference)this.Parent.Controls[i];

if (DateTime.Parse(C.fields.Last\_tick) >= this\_last\_tick)

{

this.Parent.Controls.SetChildIndex(this, i);

break;

}

}

catch (Exception) { }

}

}

}

}

}

### Conference.Designer.cs

namespace chat\_client

{

partial class Conference

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором компонентов

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

protected void InitializeComponent()

{

this.panel1 = new System.Windows.Forms.Panel();

this.name = new System.Windows.Forms.Label();

this.panel1.SuspendLayout();

this.SuspendLayout();

//

// panel1

//

this.panel1.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.panel1.BackColor = System.Drawing.SystemColors.ControlLight;

this.panel1.Controls.Add(this.name);

this.panel1.Location = new System.Drawing.Point(0, 1);

this.panel1.MinimumSize = new System.Drawing.Size(35, 15);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(35, 15);

this.panel1.TabIndex = 20;

this.panel1.Click += new System.EventHandler(this.Contact\_Click);

//

// name

//

this.name.AutoEllipsis = true;

this.name.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F);

this.name.ForeColor = System.Drawing.SystemColors.HotTrack;

this.name.ImeMode = System.Windows.Forms.ImeMode.NoControl;

this.name.Location = new System.Drawing.Point(0, 0);

this.name.MaximumSize = new System.Drawing.Size(0, 15);

this.name.MinimumSize = new System.Drawing.Size(35, 15);

this.name.Name = "name";

this.name.Size = new System.Drawing.Size(35, 15);

this.name.TabIndex = 20;

this.name.Text = "имя";

this.name.Click += new System.EventHandler(this.Contact\_Click);

//

// Conference

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.Transparent;

this.Controls.Add(this.panel1);

this.MaximumSize = new System.Drawing.Size(0, 17);

this.MinimumSize = new System.Drawing.Size(35, 17);

this.Name = "Conference";

this.Size = new System.Drawing.Size(35, 17);

this.Click += new System.EventHandler(this.Contact\_Click);

this.panel1.ResumeLayout(false);

this.ResumeLayout(false);

}

#endregion

protected System.Windows.Forms.Panel panel1;

public System.Windows.Forms.Label name;

}

}

### Conference\_fields.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Threading;

namespace chat\_client

{

delegate void Conference\_fields\_delegate(Conference\_fields sender);

public class Conference\_fields

{

public static Dictionary<string, Conference\_fields> Conferences;//id contact для конфере

public static Conference\_fields getConference\_fields(string id){

if (Conferences.ContainsKey(id)) return Conferences[id];

else return null;

}

public static Conference\_fields getConference\_fields(string id, string name)

{

if (Conferences.ContainsKey(id)){

Conferences[id].Name = name;

return Conferences[id];

}

Conference\_fields res = (Conferences[id] = new Conference\_fields(id, name));

Central.insert\_conference(res);

return res;

}

private bool unread;

private string last\_tick, name, id;

public ItemList\_with\_Button\_more messages, partners;

public List<method\_empty> UnreadChanged, Last\_tickChanged;

private Conference\_fields(string id, string name)

{

UnreadChanged = new List<method\_empty>();

Last\_tickChanged = new List<method\_empty>();

this.id = id;

this.name = name;

messages = new ItemList\_with\_Button\_more(new method(this.load\_more\_messages), DockStyle.Fill, DockStyle.Top);

messages.AutoScroll = true;

partners = new ItemList\_with\_Button\_more(new method(this.load\_more\_partners), DockStyle.Fill, DockStyle.Top);

partners.AutoScroll = true;

unread\_and\_last\_tick\_checker();

}

public Conference getConference()

{

return new Conference(this.ID, this.Name, Central.Chat.contextMenuStrip2CONFER);

}

public void unread\_and\_last\_tick\_checker()

{

int k = (int)Central.key.geter\_conf\_unread\_last\_tick;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.start\_dialog(k);

Central.write(this.ID, Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

this.Unread = Convert.ToBoolean(Convert.ToInt16(Central.read(Central.socks[k])));

this.Last\_tick = Central.read(Central.socks[k]);

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Central.stop\_dialog(k);

}

private void run\_listeners(List<method\_empty> methodsList){

int n = methodsList.Count;

for (int i = 0; i < n; i++)

{

try { methodsList[i](); }

catch (Exception e)

{

MessageBox.Show(e.ToString(), "Error.");

}

}

}

public bool Unread{

get { return unread; }

set

{

if (value != unread) {

unread = value;

run\_listeners(UnreadChanged);

}

}

}

public string Last\_tick{

get { return last\_tick; }

set {

if (value != last\_tick) {

last\_tick = value;

run\_listeners(Last\_tickChanged);

}

}

}

public string Name{

get { return name; }

private set {

if (name != value) name = value;

}

}

public string ID{

get { return id; }

}

//для вызова из двугого потока с использованием инвок

public void insert\_message(object message\_ID, object time, object message, object login)

{

insert\_message((string)message\_ID, (string)time, (string)message, (string)login);

}

public void insert\_message(object m) { insert\_message((Message)m); }

//при использовании этой перегрузки метода не приходится создавать лишних слушателей событий

public void insert\_message(string message\_ID, string time, string message, string login)

{

if (!messages.Controls.ContainsKey(message\_ID)) insert\_message(new Message((string)message\_ID, (string)time, (string)message, User.getUser((string)login)));

}

public void insert\_message(Message m)

{

if (!messages.Controls.ContainsKey(m.Name))

{

messages.Controls.Add(m);

if (messages.Controls.Count > 2)

{

DateTime time\_of\_message = new DateTime();

time\_of\_message = DateTime.Parse(m.time.Text);

foreach (object obj\_ in messages.Controls)

{

try

{

Message obj = (Message)obj\_;

if (DateTime.Parse(obj.time.Text) >= time\_of\_message)

{

messages.Controls.SetChildIndex(m, messages.Controls.IndexOf(obj));

if (messages.Controls.IndexOf(m) == messages.Controls.Count - 1) messages.VerticalScroll.Value = messages.VerticalScroll.Maximum;

break;

}

}

catch (Exception) { }

}

}

}

}

public void load\_more\_messages()

{

int k = (int)Central.key.geter\_n\_m\_messages\_p;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.start\_dialog(k);

Central.write((messages.Controls.Count - 1).ToString(), Central.socks[k]);

Central.write((messages.Controls.Count - 1 + 10).ToString(), Central.socks[k]);

Central.write(ID, Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

int n = Convert.ToInt32(Central.read(Central.socks[k]));

Message[] messages\_ = new Message[n];

User messager;

string message\_ID, message, login, time;

for (int i = 0; i < n; i++)

{

message\_ID = Central.read(Central.socks[k]);

message = Central.read(Central.socks[k]);

login = Central.read(Central.socks[k]);

time = Central.read(Central.socks[k]);

messager = User.getUser(login);

this.insert\_message(new Message(message\_ID, time, message, messager));

}

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Central.stop\_dialog(k);

}

private void load\_more\_messages(object sender, EventArgs e)

{

load\_more\_messages();

}

public void load\_more\_partners(object sender, EventArgs e) {

}

}

}

### Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

using System.Threading;

namespace chat\_client

{

public delegate void method(object sender, EventArgs e);

public delegate void method\_empty();

public partial class Form1 : Form

{

#region \_.\_.\_.\_.Готово

////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////

//для управления расположения панелей на форме

int x;

private void panel11\_MouseDown(object sender, MouseEventArgs e)

{

x = e.X;

}

private void panel11\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left && e.X != x)

{

if ((panel11.Location.X + e.X - x >= menuStrip1.Size.Width / 4) && ((panel11.Location.X + e.X - x) + panel11.Size.Width <= (menuStrip1.Size.Width \* 4) / 10))

panel11.Location = new Point(panel11.Location.X + e.X - x, panel11.Location.Y);

else

{

if (panel11.Location.X + e.X - x < menuStrip1.Size.Width / 4)

panel11.Location = new Point(menuStrip1.Size.Width / 4, panel11.Location.Y);

else

panel11.Location = new Point((menuStrip1.Size.Width \* 4) / 10 - panel11.Size.Width, panel11.Location.Y);

}

}

}

private void panel11\_Move(object sender, EventArgs e)

{

panel1.Size = new Size(panel11.Location.X, panel1.Size.Height);

panel2.Size = new Size(menuStrip1.Size.Width - (panel11.Location.X + panel11.Size.Width), panel2.Size.Height);

panel2.Location = new Point(panel11.Location.X + panel11.Size.Width, menuStrip1.Size.Height);

}

////////////////////////////////////////////////////////////////

private void toolStripMenuItem5\_Click(object sender, EventArgs e){

this.Close();

this.Dispose();

Central.Main();

}

////////////////////////////////////////////////////////////////

//управление воодом сообщения

bool shiftDown;

delegate void no\_params();

private void richTextBox2\_KeyDown(object sender, KeyEventArgs e){

if (e.KeyCode == Keys.Shift || e.KeyCode == Keys.ShiftKey) shiftDown = true;

if (e.KeyCode == Keys.Enter && !shiftDown) {

(new System.Threading.Thread(new ParameterizedThreadStart(Central.send\_mess))).Start(this.richTextBox2.Text);

(new System.Threading.Thread(delegate() { this.BeginInvoke(new no\_params(richTextBox2.Clear)); })).Start();

}

}

private void richTextBox2\_KeyUp(object sender, KeyEventArgs e){

if (e.KeyCode == Keys.Shift || e.KeyCode == Keys.ShiftKey) shiftDown = false;

}

private void richTextBox2\_Leave(object sender, EventArgs e){

shiftDown = false;

}

private void button1\_Click(object sender, EventArgs ev){

Central.send\_mess(this.richTextBox2.Text);

}

////////////////////////////////////////////////////////////////

private void toolStripMenuItem6\_Click(object sender, EventArgs e){

Central.refresh\_data();

}

////////////////////////////////////////////////////////////////

private void textBox1\_TextChanged(object sender, EventArgs e)

{

search();

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == '\r') search();

}

private void button2\_Click(object sender, EventArgs e)

{

textBox1.Text = "";

}

private void button9\_Click(object sender, EventArgs e)

{

search();

}

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

foreach (Control obj in panel5.Controls) { if (obj.Dock != DockStyle.Top) panel5.Controls.Remove(obj); }

panel5.Controls.Add((Control)tabControl1.SelectedTab.Tag);

Central.load\_more\_to(tabControl1.SelectedTab.Tag, null);

}

void search()

{

//ItemList\_with\_Button\_more list = (ItemList\_with\_Button\_more)tabControl1.SelectedTab.Tag;

//list.Clear();

Central.load\_more\_to(tabControl1.SelectedTab.Tag, null);

}

////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

Central.Close();

}

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

User u = (User)((ContextMenuStrip)(((ToolStripMenuItem)sender).GetCurrentParent())).SourceControl;

Label label = new Label();

label.Text = "\n\nStatus:\n" + Central.get\_user\_status(u.Name);

Form3 form = new Form3(u, label);

form.ShowDialog();

}

private void toolStripMenuItem11\_Click(object sender, EventArgs e)

{

User u = (User)((ContextMenuStrip)(((ToolStripMenuItem)sender).GetCurrentParent())).SourceControl;

Label label = new Label();

label.Text = "\n\nStatus:\n" + Central.get\_user\_status(u.Name);

if (u.Tag != null) label.Text += "\n\nMessage:\n" + (string)u.Tag;

Form3 form = new Form3(u, label);

form.ShowDialog();

}

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

Conference c = (Conference)((ContextMenuStrip)(((ToolStripMenuItem)sender).GetCurrentParent())).SourceControl;

Form3 form = new Form3(c, c.fields.partners);

form.ShowDialog();

}

private void перепискаToolStripMenuItem\_Click(object sender, EventArgs e)

{

try { Central.ShowMessages(User\_fields.Users[((User)(((ContextMenuStrip)(((ToolStripMenuItem)sender).GetCurrentParent())).SourceControl)).Name]); }

catch(Exception) { Central.ShowMessages(User\_fields.Users[((Conference)(((ContextMenuStrip)(((ToolStripMenuItem)sender).GetCurrentParent())).SourceControl)).Name]); }

}

private void toolStripMenuItem9\_Click(object sender, EventArgs e)

{

try { Central.ShowMessages(User\_fields.Users[((User)(((ContextMenuStrip)(((ToolStripMenuItem)sender).GetCurrentParent())).SourceControl)).Name]); }

catch (Exception) { Central.ShowMessages(User\_fields.Users[((Conference)(((ContextMenuStrip)(((ToolStripMenuItem)sender).GetCurrentParent())).SourceControl)).Name]); }

}

private void toolStripMenuItem13\_Click(object sender, EventArgs e)

{

User u = (User)((ContextMenuStrip)(((ToolStripMenuItem)sender).GetCurrentParent())).SourceControl;

if (!Central.requests.Controls.Contains(u))

{

AdderFriend form = new AdderFriend(u);

form.Show();

}

else accept(u);

}

void accept(User u)

{

Waiter w = new Waiter(ref Central.counter[(int)Central.key.accepter\_request, 0], ref Central.counter[(int)Central.key.accepter\_request, 1], new method\_obj1(this.accept\_), u);

}

object accept\_(object u\_)

{

User u = (User)u\_;

int k = (int)Central.key.accepter\_request;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.write(u.Name, Central.socks[k]);

if (Central.read(Central.socks[k]) != Central.answers[0])

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Central.requests.Controls.RemoveByKey(u.Name);

Central.searching\_panel.Controls.RemoveByKey(u.Name);

return null;

}

#endregion

public Form1(){

InitializeComponent();

this.Text += ": " + Central.log;

tabPage1.Tag = Central.friends;

tabPage2.Tag = Central.conferences;

tabPage3.Tag = Central.searching\_panel;

tabPage4.Tag = Central.requests;

this.panel5.Controls.Add(Central.friends);

}

}

}

### Form1.Designer.cs

namespace chat\_client

{

partial class Form1

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(Form1));

this.menuStrip1 = new System.Windows.Forms.MenuStrip();

this.toolStripMenuItem4 = new System.Windows.Forms.ToolStripMenuItem();

this.toolStripMenuItem5 = new System.Windows.Forms.ToolStripMenuItem();

this.toolStripMenuItem6 = new System.Windows.Forms.ToolStripMenuItem();

this.toolStripMenuItem7 = new System.Windows.Forms.ToolStripMenuItem();

this.аккаунтToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.panel1 = new System.Windows.Forms.Panel();

this.panel5 = new System.Windows.Forms.Panel();

this.panel3 = new System.Windows.Forms.Panel();

this.textBox1 = new System.Windows.Forms.TextBox();

this.button2 = new System.Windows.Forms.Button();

this.button9 = new System.Windows.Forms.Button();

this.panel2 = new System.Windows.Forms.Panel();

this.panel6 = new System.Windows.Forms.Panel();

this.panel7 = new System.Windows.Forms.Panel();

this.richTextBox2 = new System.Windows.Forms.RichTextBox();

this.button1 = new System.Windows.Forms.Button();

this.panel11 = new System.Windows.Forms.Panel();

this.panel4 = new System.Windows.Forms.Panel();

this.contextMenuStrip1FRIEND = new System.Windows.Forms.ContextMenuStrip(this.components);

this.toolStripMenuItem1 = new System.Windows.Forms.ToolStripMenuItem();

this.перепискаToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.contextMenuStrip2CONFER = new System.Windows.Forms.ContextMenuStrip(this.components);

this.toolStripMenuItem2 = new System.Windows.Forms.ToolStripMenuItem();

this.toolStripMenuItem9 = new System.Windows.Forms.ToolStripMenuItem();

this.contextMenuStrip3USERS = new System.Windows.Forms.ContextMenuStrip(this.components);

this.toolStripMenuItem11 = new System.Windows.Forms.ToolStripMenuItem();

this.toolStripMenuItem13 = new System.Windows.Forms.ToolStripMenuItem();

this.tabPage4 = new System.Windows.Forms.TabPage();

this.tabPage3 = new System.Windows.Forms.TabPage();

this.tabPage2 = new System.Windows.Forms.TabPage();

this.tabPage1 = new System.Windows.Forms.TabPage();

this.tabControl1 = new System.Windows.Forms.TabControl();

this.menuStrip1.SuspendLayout();

this.panel1.SuspendLayout();

this.panel3.SuspendLayout();

this.panel2.SuspendLayout();

this.panel7.SuspendLayout();

this.panel11.SuspendLayout();

this.contextMenuStrip1FRIEND.SuspendLayout();

this.contextMenuStrip2CONFER.SuspendLayout();

this.contextMenuStrip3USERS.SuspendLayout();

this.tabControl1.SuspendLayout();

this.SuspendLayout();

//

// menuStrip1

//

this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.toolStripMenuItem4});

resources.ApplyResources(this.menuStrip1, "menuStrip1");

this.menuStrip1.Name = "menuStrip1";

//

// toolStripMenuItem4

//

this.toolStripMenuItem4.BackColor = System.Drawing.Color.DarkCyan;

this.toolStripMenuItem4.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.toolStripMenuItem5,

this.toolStripMenuItem6,

this.toolStripMenuItem7,

this.аккаунтToolStripMenuItem});

this.toolStripMenuItem4.Name = "toolStripMenuItem4";

resources.ApplyResources(this.toolStripMenuItem4, "toolStripMenuItem4");

//

// toolStripMenuItem5

//

this.toolStripMenuItem5.Name = "toolStripMenuItem5";

resources.ApplyResources(this.toolStripMenuItem5, "toolStripMenuItem5");

this.toolStripMenuItem5.Click += new System.EventHandler(this.toolStripMenuItem5\_Click);

//

// toolStripMenuItem6

//

this.toolStripMenuItem6.Name = "toolStripMenuItem6";

resources.ApplyResources(this.toolStripMenuItem6, "toolStripMenuItem6");

this.toolStripMenuItem6.Click += new System.EventHandler(this.toolStripMenuItem6\_Click);

//

// toolStripMenuItem7

//

this.toolStripMenuItem7.Name = "toolStripMenuItem7";

resources.ApplyResources(this.toolStripMenuItem7, "toolStripMenuItem7");

//

// аккаунтToolStripMenuItem

//

this.аккаунтToolStripMenuItem.Name = "аккаунтToolStripMenuItem";

resources.ApplyResources(this.аккаунтToolStripMenuItem, "аккаунтToolStripMenuItem");

//

// panel1

//

this.panel1.BackColor = System.Drawing.Color.Transparent;

this.panel1.Controls.Add(this.panel5);

this.panel1.Controls.Add(this.tabControl1);

this.panel1.Controls.Add(this.panel3);

resources.ApplyResources(this.panel1, "panel1");

this.panel1.Name = "panel1";

//

// panel5

//

resources.ApplyResources(this.panel5, "panel5");

this.panel5.Name = "panel5";

//

// panel3

//

this.panel3.Controls.Add(this.textBox1);

this.panel3.Controls.Add(this.button2);

this.panel3.Controls.Add(this.button9);

resources.ApplyResources(this.panel3, "panel3");

this.panel3.Name = "panel3";

//

// textBox1

//

resources.ApplyResources(this.textBox1, "textBox1");

this.textBox1.Name = "textBox1";

this.textBox1.TextChanged += new System.EventHandler(this.textBox1\_TextChanged);

this.textBox1.KeyPress += new System.Windows.Forms.KeyPressEventHandler(this.textBox1\_KeyPress);

//

// button2

//

resources.ApplyResources(this.button2, "button2");

this.button2.Image = global::chat\_client.Properties.Resources.крестик;

this.button2.Name = "button2";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// button9

//

resources.ApplyResources(this.button9, "button9");

this.button9.Image = global::chat\_client.Properties.Resources.лупа3;

this.button9.Name = "button9";

this.button9.UseVisualStyleBackColor = true;

this.button9.Click += new System.EventHandler(this.button9\_Click);

//

// panel2

//

resources.ApplyResources(this.panel2, "panel2");

this.panel2.BackColor = System.Drawing.Color.Transparent;

this.panel2.Controls.Add(this.panel6);

this.panel2.Controls.Add(this.panel7);

this.panel2.Name = "panel2";

//

// panel6

//

resources.ApplyResources(this.panel6, "panel6");

this.panel6.Name = "panel6";

//

// panel7

//

this.panel7.Controls.Add(this.richTextBox2);

this.panel7.Controls.Add(this.button1);

resources.ApplyResources(this.panel7, "panel7");

this.panel7.Name = "panel7";

//

// richTextBox2

//

this.richTextBox2.BackColor = System.Drawing.SystemColors.Window;

resources.ApplyResources(this.richTextBox2, "richTextBox2");

this.richTextBox2.Name = "richTextBox2";

this.richTextBox2.KeyDown += new System.Windows.Forms.KeyEventHandler(this.richTextBox2\_KeyDown);

this.richTextBox2.KeyUp += new System.Windows.Forms.KeyEventHandler(this.richTextBox2\_KeyUp);

this.richTextBox2.Leave += new System.EventHandler(this.richTextBox2\_Leave);

//

// button1

//

resources.ApplyResources(this.button1, "button1");

this.button1.Name = "button1";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// panel11

//

resources.ApplyResources(this.panel11, "panel11");

this.panel11.BackColor = System.Drawing.Color.Transparent;

this.panel11.Controls.Add(this.panel4);

this.panel11.Cursor = System.Windows.Forms.Cursors.SizeWE;

this.panel11.Name = "panel11";

this.panel11.MouseDown += new System.Windows.Forms.MouseEventHandler(this.panel11\_MouseDown);

this.panel11.MouseMove += new System.Windows.Forms.MouseEventHandler(this.panel11\_MouseMove);

this.panel11.Move += new System.EventHandler(this.panel11\_Move);

//

// panel4

//

resources.ApplyResources(this.panel4, "panel4");

this.panel4.BackColor = System.Drawing.SystemColors.ControlLight;

this.panel4.Name = "panel4";

this.panel4.MouseDown += new System.Windows.Forms.MouseEventHandler(this.panel11\_MouseDown);

this.panel4.MouseMove += new System.Windows.Forms.MouseEventHandler(this.panel11\_MouseMove);

//

// contextMenuStrip1FRIEND

//

this.contextMenuStrip1FRIEND.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.toolStripMenuItem1,

this.перепискаToolStripMenuItem});

this.contextMenuStrip1FRIEND.Name = "contextMenuStrip1";

this.contextMenuStrip1FRIEND.ShowImageMargin = false;

resources.ApplyResources(this.contextMenuStrip1FRIEND, "contextMenuStrip1FRIEND");

//

// toolStripMenuItem1

//

this.toolStripMenuItem1.Name = "toolStripMenuItem1";

resources.ApplyResources(this.toolStripMenuItem1, "toolStripMenuItem1");

this.toolStripMenuItem1.Click += new System.EventHandler(this.toolStripMenuItem1\_Click);

//

// перепискаToolStripMenuItem

//

this.перепискаToolStripMenuItem.Name = "перепискаToolStripMenuItem";

resources.ApplyResources(this.перепискаToolStripMenuItem, "перепискаToolStripMenuItem");

this.перепискаToolStripMenuItem.Click += new System.EventHandler(this.перепискаToolStripMenuItem\_Click);

//

// contextMenuStrip2CONFER

//

this.contextMenuStrip2CONFER.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.toolStripMenuItem2,

this.toolStripMenuItem9});

this.contextMenuStrip2CONFER.Name = "contextMenuStrip1";

this.contextMenuStrip2CONFER.ShowImageMargin = false;

resources.ApplyResources(this.contextMenuStrip2CONFER, "contextMenuStrip2CONFER");

//

// toolStripMenuItem2

//

this.toolStripMenuItem2.Name = "toolStripMenuItem2";

resources.ApplyResources(this.toolStripMenuItem2, "toolStripMenuItem2");

this.toolStripMenuItem2.Click += new System.EventHandler(this.toolStripMenuItem2\_Click);

//

// toolStripMenuItem9

//

this.toolStripMenuItem9.Name = "toolStripMenuItem9";

resources.ApplyResources(this.toolStripMenuItem9, "toolStripMenuItem9");

this.toolStripMenuItem9.Click += new System.EventHandler(this.toolStripMenuItem9\_Click);

//

// contextMenuStrip3USERS

//

this.contextMenuStrip3USERS.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.toolStripMenuItem11,

this.toolStripMenuItem13});

this.contextMenuStrip3USERS.Name = "contextMenuStrip1";

this.contextMenuStrip3USERS.ShowImageMargin = false;

resources.ApplyResources(this.contextMenuStrip3USERS, "contextMenuStrip3USERS");

//

// toolStripMenuItem11

//

this.toolStripMenuItem11.Name = "toolStripMenuItem11";

resources.ApplyResources(this.toolStripMenuItem11, "toolStripMenuItem11");

this.toolStripMenuItem11.Click += new System.EventHandler(this.toolStripMenuItem11\_Click);

//

// toolStripMenuItem13

//

this.toolStripMenuItem13.Name = "toolStripMenuItem13";

resources.ApplyResources(this.toolStripMenuItem13, "toolStripMenuItem13");

this.toolStripMenuItem13.Click += new System.EventHandler(this.toolStripMenuItem13\_Click);

//

// tabPage4

//

resources.ApplyResources(this.tabPage4, "tabPage4");

this.tabPage4.Name = "tabPage4";

this.tabPage4.UseVisualStyleBackColor = true;

//

// tabPage3

//

resources.ApplyResources(this.tabPage3, "tabPage3");

this.tabPage3.Name = "tabPage3";

this.tabPage3.UseVisualStyleBackColor = true;

//

// tabPage2

//

this.tabPage2.BackColor = System.Drawing.Color.Transparent;

resources.ApplyResources(this.tabPage2, "tabPage2");

this.tabPage2.Name = "tabPage2";

//

// tabPage1

//

resources.ApplyResources(this.tabPage1, "tabPage1");

this.tabPage1.Name = "tabPage1";

this.tabPage1.UseVisualStyleBackColor = true;

//

// tabControl1

//

this.tabControl1.Controls.Add(this.tabPage1);

this.tabControl1.Controls.Add(this.tabPage2);

this.tabControl1.Controls.Add(this.tabPage3);

this.tabControl1.Controls.Add(this.tabPage4);

resources.ApplyResources(this.tabControl1, "tabControl1");

this.tabControl1.Name = "tabControl1";

this.tabControl1.SelectedIndex = 0;

this.tabControl1.SelectedIndexChanged += new System.EventHandler(this.tabControl1\_SelectedIndexChanged);

//

// Form1

//

resources.ApplyResources(this, "$this");

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.DarkCyan;

this.Controls.Add(this.panel11);

this.Controls.Add(this.panel2);

this.Controls.Add(this.panel1);

this.Controls.Add(this.menuStrip1);

this.Name = "Form1";

this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Form1\_FormClosing);

this.menuStrip1.ResumeLayout(false);

this.menuStrip1.PerformLayout();

this.panel1.ResumeLayout(false);

this.panel3.ResumeLayout(false);

this.panel3.PerformLayout();

this.panel2.ResumeLayout(false);

this.panel7.ResumeLayout(false);

this.panel11.ResumeLayout(false);

this.contextMenuStrip1FRIEND.ResumeLayout(false);

this.contextMenuStrip2CONFER.ResumeLayout(false);

this.contextMenuStrip3USERS.ResumeLayout(false);

this.tabControl1.ResumeLayout(false);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Panel panel11;

private System.Windows.Forms.MenuStrip menuStrip1;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem4;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem5;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem6;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem7;

private System.Windows.Forms.Button button9;

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Panel panel2;

public System.Windows.Forms.ContextMenuStrip contextMenuStrip1FRIEND;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem1;

private System.Windows.Forms.ToolStripMenuItem перепискаToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem аккаунтToolStripMenuItem;

public System.Windows.Forms.Panel panel7;

private System.Windows.Forms.RichTextBox richTextBox2;

private System.Windows.Forms.Button button1;

private System.Windows.Forms.Panel panel3;

private System.Windows.Forms.Button button2;

private System.Windows.Forms.Panel panel4;

public System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Panel panel5;

public System.Windows.Forms.ContextMenuStrip contextMenuStrip2CONFER;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem2;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem9;

public System.Windows.Forms.ContextMenuStrip contextMenuStrip3USERS;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem11;

private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem13;

public System.Windows.Forms.Panel panel6;

private System.Windows.Forms.TabControl tabControl1;

private System.Windows.Forms.TabPage tabPage1;

private System.Windows.Forms.TabPage tabPage2;

private System.Windows.Forms.TabPage tabPage3;

private System.Windows.Forms.TabPage tabPage4;

}

}

### Form2.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

namespace chat\_client

{

public partial class Form2 : Form{

public bool aut;

private Socket sock;

string[] commands, answers;

string key;

public Form2(string Title, string label1, string label2, string[] commands, string[] answers, string key)

{

InitializeComponent();

setTitle(Title);

setLabel1(label1);

setLabel2(label2);

aut = false;

sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

Central.connect(sock);

this.commands = commands;

this.answers = answers;

this.key = key;

}

private void button1\_Click(object sender, EventArgs e){

Central.write(commands[0], sock);

Central.write(getParam1(), sock);

Central.write(getParam2(), sock);

Central.write(key, sock);

string res = Central.read(sock);

if (res == answers[0]){

sock.Close();

aut = true;

this.Close();

} else MessageBox.Show(Central.read(sock), "Message from server.");

}

private void button2\_Click(object sender, EventArgs e){

if (getParam1() == ""){

MessageBox.Show("User cannot have empty login.", "Warning.");

return;

}

Central.write(commands[1], sock);

Central.write(getParam1(), sock);

Central.write(getParam2(), sock);

string res = Central.read(sock);

if (res == answers[1]) MessageBox.Show(Central.read(sock), "Message from server.");

else MessageBox.Show("user with login: \' " + getParam1() + " \' registrated.", "Message from server.");

}

public string getParam1() { return textBox1.Text; }

public string getParam2() { return textBox2.Text; }

public void setTitle(string s) { this.Text = s; }

public void setLabel1(string s) { label1.Text = s; }

public void setLabel2(string s) { label2.Text = s; }

}

}

### Form2.Designer.cs

namespace chat\_client

{

partial class Form2

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.button1 = new System.Windows.Forms.Button();

this.textBox1 = new System.Windows.Forms.TextBox();

this.textBox2 = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.button2 = new System.Windows.Forms.Button();

this.panel1 = new System.Windows.Forms.Panel();

this.panel1.SuspendLayout();

this.SuspendLayout();

//

// button1

//

this.button1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.button1.Location = new System.Drawing.Point(23, 119);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(192, 27);

this.button1.TabIndex = 0;

this.button1.Text = "authorize";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(23, 32);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(192, 20);

this.textBox1.TabIndex = 1;

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(23, 81);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(192, 20);

this.textBox2.TabIndex = 2;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(20, 16);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(33, 13);

this.label1.TabIndex = 3;

this.label1.Text = "Login";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(20, 65);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(53, 13);

this.label2.TabIndex = 4;

this.label2.Text = "Password";

//

// button2

//

this.button2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.button2.Location = new System.Drawing.Point(23, 152);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(192, 27);

this.button2.TabIndex = 5;

this.button2.Text = "register";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// panel1

//

this.panel1.Anchor = System.Windows.Forms.AnchorStyles.None;

this.panel1.Controls.Add(this.label1);

this.panel1.Controls.Add(this.button2);

this.panel1.Controls.Add(this.button1);

this.panel1.Controls.Add(this.label2);

this.panel1.Controls.Add(this.textBox1);

this.panel1.Controls.Add(this.textBox2);

this.panel1.Location = new System.Drawing.Point(167, 77);

this.panel1.MaximumSize = new System.Drawing.Size(235, 201);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(235, 201);

this.panel1.TabIndex = 6;

//

// Form2

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.DarkCyan;

this.ClientSize = new System.Drawing.Size(568, 355);

this.Controls.Add(this.panel1);

this.MinimumSize = new System.Drawing.Size(251, 239);

this.Name = "Form2";

this.Text = "chat";

this.panel1.ResumeLayout(false);

this.panel1.PerformLayout();

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.Button button1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.TextBox textBox2;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Button button2;

private System.Windows.Forms.Panel panel1;

}

}

### Form3.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class Form3 : Form

{

public Form3(Control first, Control second){

InitializeComponent();

panel1.Controls.Add(first);

if ((Label)second != null)

{

second.MaximumSize = new Size(this.Size.Width, 0);

this.AutoSize = second.AutoSize;

}

panel2.Controls.Add(second);

}

}

}

### Form3.Designer.cs

namespace chat\_client

{

partial class Form3

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.panel1 = new System.Windows.Forms.Panel();

this.panel2 = new System.Windows.Forms.Panel();

this.SuspendLayout();

//

// panel1

//

this.panel1.Dock = System.Windows.Forms.DockStyle.Top;

this.panel1.Location = new System.Drawing.Point(0, 0);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(362, 58);

this.panel1.TabIndex = 0;

//

// panel2

//

this.panel2.Dock = System.Windows.Forms.DockStyle.Fill;

this.panel2.Location = new System.Drawing.Point(0, 58);

this.panel2.Name = "panel2";

this.panel2.Size = new System.Drawing.Size(362, 338);

this.panel2.TabIndex = 1;

//

// Form3

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.DarkCyan;

this.ClientSize = new System.Drawing.Size(362, 396);

this.Controls.Add(this.panel2);

this.Controls.Add(this.panel1);

this.Name = "Form3";

this.Text = "Info";

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Panel panel2;

}

}

### ItemList\_with\_Button\_more.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class ItemList\_with\_Button\_more : UserControl

{

private method load\_more\_to\_this;

public ItemList\_with\_Button\_more(method load\_more\_to\_this, DockStyle dockStyle, DockStyle Button\_more\_dockStyle){

InitializeComponent();

more = new Button\_more(new method(this.more\_Click));

more.Dock = Button\_more\_dockStyle;

this.Dock = dockStyle;

this.Controls.Add(more);

this.load\_more\_to\_this = load\_more\_to\_this;

}

//очищает компонент от когтроллов, и добавляет свой Button\_more

public void Clear(){

this.Controls.Clear();

this.Controls.Add(more);

}

public int get\_n\_visible\_children() {

int res = 0;

foreach (Control obj in Controls) {

if (obj.Visible) res++;

}

return res;

}

private void more\_Click(object sender, EventArgs e) {

load\_more\_to\_this(this, e);

}

}

}

### ItemList\_with\_Button\_more.Designer.cs

namespace chat\_client

{

partial class ItemList\_with\_Button\_more

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором компонентов

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.SuspendLayout();

//

// ItemList\_with\_Button\_more

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.AutoScroll = true;

this.BackColor = System.Drawing.Color.Transparent;

this.Name = "ItemList\_with\_Button\_more";

this.Size = new System.Drawing.Size(160, 163);

this.ResumeLayout(false);

}

#endregion

Button\_more more;

}

}

### Menager\_for\_conferens.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class Menager\_for\_conference : UserControl

{

public Menager\_for\_conference(User u, Conference c)

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button9\_Click(object sender, EventArgs e)

{

}

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

### Menager\_for\_conferens.Designer.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class Menager\_for\_conference : UserControl

{

public Menager\_for\_conference(User u, Conference c)

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button9\_Click(object sender, EventArgs e)

{

}

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

### Message.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class Message : UserControl

{

public string message\_ID;

private Control oldParent;

private EventHandler Message\_SizeChanged\_Listener;

public void init(string message\_ID, string time, string message, User messager)

{

InitializeComponent();

this.Size = new Size(40, 32);

//this.panel1.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

//| System.Windows.Forms.AnchorStyles.Left)

//| System.Windows.Forms.AnchorStyles.Right)));

this.oldParent = null;

this.Dock = DockStyle.Bottom;

this.time.Text = time;

this.messageText.Text = message;

panel1.Controls.Add(messager);

this.Name = message\_ID;

this.message\_ID = message\_ID;

this.SizeChanged += (Message\_SizeChanged\_Listener = new EventHandler(this.Message\_SizeChanged));

}

public Message(string message\_ID, string time, string message, User messager)

{

init(message\_ID, time, message, messager);

}

private void Parent\_SizeChanged(object sender, EventArgs e)

{

this.MaximumSize = new Size(this.Parent.Size.Width, 0);

}

private void Message\_ParentChanged(object sender, EventArgs e)

{

if (oldParent != null) oldParent.SizeChanged -= Parent\_SizeChanged;

oldParent = this.Parent;

if (this.Parent != null) this.Parent.SizeChanged += Parent\_SizeChanged;

this.MaximumSize = new Size(this.Parent.Size.Width, 0);

}

private void Message\_SizeChanged(object sender, EventArgs e)

{

this.SizeChanged -= Message\_SizeChanged\_Listener;

this.messageText.MaximumSize = new Size(this.MaximumSize.Width, 0);

//panel2.Size = new Size(this.Size.Width, this.messageText.Location.Y + this.messageText.Size.Height);

panel1.Size = new Size(this.Size.Width, this.panel2.Location.Y + this.messageText.Location.Y + this.messageText.Size.Height);

this.Size = new Size(this.Size.Width, panel1.Size.Height + 2);

this.SizeChanged += Message\_SizeChanged\_Listener;

messageText.MaximumSize = new Size(panel2.Size.Width, panel2.MaximumSize.Height);

}

}

}

### Message.Designer.cs

namespace chat\_client

{

partial class Message

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором компонентов

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.panel1 = new System.Windows.Forms.Panel();

this.panel2 = new System.Windows.Forms.Panel();

this.messageText = new System.Windows.Forms.Label();

this.time = new System.Windows.Forms.Label();

this.panel1.SuspendLayout();

this.panel2.SuspendLayout();

this.SuspendLayout();

//

// panel1

//

this.panel1.BackColor = System.Drawing.SystemColors.ControlLight;

this.panel1.Controls.Add(this.panel2);

this.panel1.Location = new System.Drawing.Point(0, 1);

this.panel1.MinimumSize = new System.Drawing.Size(40, 30);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(40, 30);

this.panel1.TabIndex = 0;

//

// panel2

//

this.panel2.AutoSize = true;

this.panel2.Controls.Add(this.messageText);

this.panel2.Controls.Add(this.time);

this.panel2.Dock = System.Windows.Forms.DockStyle.Fill;

this.panel2.Location = new System.Drawing.Point(0, 0);

this.panel2.Name = "panel2";

this.panel2.Size = new System.Drawing.Size(40, 30);

this.panel2.TabIndex = 4;

//

// messageText

//

this.messageText.AutoSize = true;

this.messageText.Location = new System.Drawing.Point(0, 13);

this.messageText.Name = "messageText";

this.messageText.Size = new System.Drawing.Size(31, 13);

this.messageText.TabIndex = 6;

this.messageText.Text = "сооб";

//

// time

//

this.time.AutoSize = true;

this.time.Dock = System.Windows.Forms.DockStyle.Top;

this.time.ForeColor = System.Drawing.SystemColors.GrayText;

this.time.Location = new System.Drawing.Point(0, 0);

this.time.Name = "time";

this.time.Size = new System.Drawing.Size(39, 13);

this.time.TabIndex = 5;

this.time.Text = "время";

//

// Message

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.Transparent;

this.Controls.Add(this.panel1);

this.MinimumSize = new System.Drawing.Size(40, 32);

this.Name = "Message";

this.Size = new System.Drawing.Size(43, 34);

this.ParentChanged += new System.EventHandler(this.Message\_ParentChanged);

this.panel1.ResumeLayout(false);

this.panel1.PerformLayout();

this.panel2.ResumeLayout(false);

this.panel2.PerformLayout();

this.ResumeLayout(false);

}

#endregion

public System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Panel panel2;

private System.Windows.Forms.Label messageText;

public System.Windows.Forms.Label time;

}

}

### User.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace chat\_client

{

public partial class User : UserControl

{

public static User getUser(string login){

return User\_fields.getUser\_fields(login).getUser();

}

private User\_fields fields;

public bool Unread{

get {

if (panel1.BackColor == SystemColors.ControlLight) return false;

else return true;

}

set {

if (value) panel1.BackColor = SystemColors.ControlDark;

else panel1.BackColor = SystemColors.ControlLight;

}

}

public bool Online

{

get

{

return this.online.Visible;

}

set

{

this.online.Visible = value;

}

}

public User(string id, string name, ContextMenuStrip contextMenuStrip){

InitializeComponent();

this.Dock = DockStyle.Top;

this.fields = User\_fields.getUser\_fields(id, name);

this.ContextMenuStrip = contextMenuStrip;

this.Unread = this.fields.Unread;

this.Online = this.fields.Online;

this.Name = this.fields.Name;

this.name.Text = this.fields.Name;

this.fields.UnreadChanged.Add(new method\_empty(this.UnreadChanged));

this.fields.Last\_tickChanged.Add(new method\_empty(this.Last\_tickChanged));

this.fields.OnlineChanged.Add(new method\_empty(this.OnlineChanged));

}

public User(User\_fields fields, ContextMenuStrip contextMenuStrip)

{

InitializeComponent();

this.Dock = DockStyle.Top;

this.fields = fields;

this.ContextMenuStrip = contextMenuStrip;

this.Unread = this.fields.Unread;

this.Online = this.fields.Online;

this.Name = this.fields.Name;

this.name.Text = this.fields.Name;

this.fields.UnreadChanged.Add(new method\_empty(this.UnreadChanged));

this.fields.Last\_tickChanged.Add(new method\_empty(this.Last\_tickChanged));

this.fields.OnlineChanged.Add(new method\_empty(this.OnlineChanged));

}

public virtual User copy() {

return new User(fields.ID, fields.Name, this.ContextMenuStrip);

}

protected virtual void Contact\_Click(object sender, EventArgs e){

if (this.fields.ID != null) Central.ShowMessages(this.fields);

}

public void UnreadChanged()

{

this.Unread = fields.Unread;

}

public void Last\_tickChanged(){

if (this.Parent != null)

{

int n = this.Parent.Controls.Count;

DateTime this\_last\_tick = DateTime.Parse(this.fields.Last\_tick);

for (int i = 0; i < n; i++)

{

try

{

User U = (User)this.Parent.Controls[i];

if (DateTime.Parse(U.fields.Last\_tick) >= this\_last\_tick)

{

this.Parent.Controls.SetChildIndex(this, i);

break;

}

}

catch (Exception) { }

}

}

}

public void OnlineChanged() {

this.Online = fields.Online;

}

}

}

### User.Designer.cs

namespace chat\_client

{

partial class User

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором компонентов

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.panel1 = new System.Windows.Forms.Panel();

this.online = new System.Windows.Forms.Label();

this.name = new System.Windows.Forms.Label();

this.panel1.SuspendLayout();

this.SuspendLayout();

//

// panel1

//

this.panel1.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.panel1.BackColor = System.Drawing.SystemColors.ControlLight;

this.panel1.Controls.Add(this.online);

this.panel1.Controls.Add(this.name);

this.panel1.Location = new System.Drawing.Point(0, 1);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(35, 25);

this.panel1.TabIndex = 0;

this.panel1.MouseClick += new System.Windows.Forms.MouseEventHandler(this.Contact\_Click);

//

// online

//

this.online.AutoSize = true;

this.online.Dock = System.Windows.Forms.DockStyle.Top;

this.online.ForeColor = System.Drawing.Color.Gray;

this.online.Location = new System.Drawing.Point(0, 13);

this.online.Name = "online";

this.online.Size = new System.Drawing.Size(35, 13);

this.online.TabIndex = 1;

this.online.Text = "online";

this.online.MouseClick += new System.Windows.Forms.MouseEventHandler(this.Contact\_Click);

//

// name

//

this.name.AutoEllipsis = true;

this.name.AutoSize = true;

this.name.Dock = System.Windows.Forms.DockStyle.Top;

this.name.ForeColor = System.Drawing.SystemColors.HotTrack;

this.name.Location = new System.Drawing.Point(0, 0);

this.name.Name = "name";

this.name.Size = new System.Drawing.Size(36, 13);

this.name.TabIndex = 0;

this.name.Text = "логин";

this.name.MouseClick += new System.Windows.Forms.MouseEventHandler(this.Contact\_Click);

//

// User

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.Transparent;

this.Controls.Add(this.panel1);

this.Name = "User";

this.Size = new System.Drawing.Size(35, 27);

this.panel1.ResumeLayout(false);

this.panel1.PerformLayout();

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Label name;

private System.Windows.Forms.Label online;

}

}

### User\_fields.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

using System.Windows.Forms;

namespace chat\_client

{

delegate void User\_fields\_delegate(User\_fields sender);

delegate void for\_setUserOnline(bool ch);

public class User\_fields

{

public static Dictionary<string, User\_fields> Users;//name, user

public static User\_fields getUser\_fields(string name){

if (Users.ContainsKey(name)) return Users[name];

int k = (int)Central.key.get\_friend\_id;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.start\_dialog(k);

Central.write(name, Central.socks[k]);

User\_fields res = null;

if (Central.read(Central.socks[k]) == Central.answers[0])

{

string id = Central.read(Central.socks[k]);

if (id == "0") res = new User\_fields(name);

else res = new User\_fields(id, name);

Users[name] = res;

Central.insert\_friend(res);

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Central.stop\_dialog(k);

return res;

}

public static User\_fields getUser\_fields(string id, string name)

{

if (Users.ContainsKey(name)){

Users[name].ID = id;

if (id == null && Central.friends.Controls.ContainsKey(name)) Central.friends.Controls.RemoveByKey(name);

else Central.insert\_friend(Users[name]);

return Users[name];

}

User\_fields res = (Users[name] = new User\_fields(id, name));

Central.insert\_friend(res);

return res;

}

private bool unread, online;

private string last\_tick, name, id;

public ItemList\_with\_Button\_more messages;

public List<method\_empty> UnreadChanged, OnlineChanged, Last\_tickChanged;

private User\_fields(string id, string name) : this(name)

{

ID = id;

}

private User\_fields(string name)

{

UnreadChanged = new List<method\_empty>();

OnlineChanged = new List<method\_empty>();

Last\_tickChanged = new List<method\_empty>();

this.name = name;

int k = (int)Central.key.get\_last\_user\_tick;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

DateTime now;

Central.start\_dialog(k);

Central.write(Central.log, Central.socks[k]);

Central.stop\_dialog(k);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

now = DateTime.Parse(Central.read(Central.socks[k]));

online\_checker(now, Central.still\_online);

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

}

public User getUser() {

if (Central.friends.Controls.ContainsKey(this.Name)) return new User(this, Central.Chat.contextMenuStrip1FRIEND);

else return new User(this, Central.Chat.contextMenuStrip3USERS);

}

public void unread\_and\_last\_tick\_checker()

{

if (this.ID == null || this.ID == "0") return;

int k = (int)Central.key.geter\_freand\_unread\_last\_tick;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.start\_dialog(k);

Central.write(this.ID, Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

this.Unread = Convert.ToBoolean(Convert.ToInt16(Central.read(Central.socks[k])));

this.Last\_tick = Central.read(Central.socks[k]);

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Central.stop\_dialog(k);

}

public void online\_checker(DateTime now, int still\_online)

{

int k = (int)Central.key.get\_last\_user\_tick;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

//for\_setUserOnline set\_userOnline = new for\_setUserOnline(this.setUserOnline);

Central.start\_dialog(k);

Central.write(this.name, Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

if (DateTime.Parse(Central.read(Central.socks[k])) > now.AddMinutes(still\_online)) this.Online = false;

else this.Online = true;

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Central.stop\_dialog(k);

}

private void run\_listeners(List<method\_empty> methodsList)

{

int n = methodsList.Count;

for (int i = 0; i < n; i++)

{

try { methodsList[i](); }

catch (Exception e)

{

MessageBox.Show(e.ToString(), "Error.");

}

}

}

public bool Unread

{

get { return unread; }

set

{

if (value != unread)

{

unread = value;

run\_listeners(UnreadChanged);

}

}

}

public bool Online

{

get { return online; }

set

{

if (value != online)

{

online = value;

run\_listeners(OnlineChanged);

}

}

}

public string Last\_tick

{

get { return last\_tick; }

set

{

if (value != last\_tick)

{

last\_tick = value;

run\_listeners(Last\_tickChanged);

}

}

}

public string Name

{

get { return name; }

}

public string ID

{

get { return id; }

private set {

if (id != value)

{

id = value;

if (value != null)

{

messages = new ItemList\_with\_Button\_more(new method(this.load\_more\_messages), DockStyle.Fill, DockStyle.Top);

messages.AutoScroll = true;

unread\_and\_last\_tick\_checker();

} else messages = null;

}

}

}

//для вызова из двугого потока с использованием инвок

public void insert\_message(object message\_ID, object time, object message, object login) {

insert\_message((string)message\_ID, (string)time, (string)message, (string)login);

}

public void insert\_message(object m) { insert\_message((Message)m); }

//при использовании этой перегрузки метода не приходится создавать лишних слушателей событий

public void insert\_message(string message\_ID, string time, string message, string login) {

if (!messages.Controls.ContainsKey(message\_ID)) insert\_message(new Message((string)message\_ID, (string)time, (string)message, User.getUser((string)login)));

}

public void insert\_message(Message m)

{

if (!messages.Controls.ContainsKey(m.Name))

{

messages.Controls.Add(m);

if (messages.Controls.Count > 2)

{

DateTime time\_of\_message = new DateTime();

time\_of\_message = DateTime.Parse(m.time.Text);

foreach (object obj\_ in messages.Controls)

{

try

{

Message obj = (Message)obj\_;

if (DateTime.Parse(obj.time.Text) >= time\_of\_message)

{

messages.Controls.SetChildIndex(m, messages.Controls.IndexOf(obj));

if (messages.Controls.IndexOf(m) == messages.Controls.Count - 1) messages.VerticalScroll.Value = messages.VerticalScroll.Maximum;

break;

}

}

catch (Exception) { }

}

}

}

}

private void load\_more\_messages(object sender, EventArgs e){

load\_more\_messages();

}

public void load\_more\_messages()

{

int k = (int)Central.key.geter\_n\_m\_messages\_p;

if (Central.socks[k] == null) Central.set\_right(k);

if (Central.socks[k] == null) Central.Fatal\_Error();

Central.start\_dialog(k);

Central.write((messages.Controls.Count - 1).ToString(), Central.socks[k]);

Central.write((messages.Controls.Count - 1 + 10).ToString(), Central.socks[k]);

Central.write(ID, Central.socks[k]);

if (Central.read(Central.socks[k]) == Central.answers[0])

{

int n = Convert.ToInt32(Central.read(Central.socks[k]));

Message[] messages\_ = new Message[n];

User messager;

string message\_ID, message, login, time;

for (int i = 0; i < n; i++)

{

message\_ID = Central.read(Central.socks[k]);

message = Central.read(Central.socks[k]);

login = Central.read(Central.socks[k]);

time = Central.read(Central.socks[k]);

messager = User.getUser(login);

this.insert\_message(new Message(message\_ID, time, message, messager));

}

}

else

{

MessageBox.Show(Central.read(Central.socks[k]), "Message from server.");

Central.socks[k].Close();

Central.set\_right(k);

}

Central.stop\_dialog(k);

}

}

}

### Waiter.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace chat\_client

{

delegate object method\_obj1(object obj1);

class Waiter{

public object res;

public Waiter(ref UInt64 first, ref UInt64 second, method\_obj1 m, object obj1){

while (first != second) { System.Threading.Thread.Sleep(5); }

++first;

res = m(obj1);

second++;

}

public Waiter(ref UInt64 first, ref UInt64 second, method\_empty m){

while (first != second) { System.Threading.Thread.Sleep(5); }

++first;

m();

second++;

}

}

}