МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка базы данных мессенджера с реализацией технологии шифрования»

Выполнил студент Гунько Дмитрий Витальевич

(Ф.И.О.)

Руководитель проекта Нистюк О.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: Нистюк О.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: Нистюк О.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

**Содержание**

[Реферат 3](#_Toc58798048)

[1. Аналитический обзор литературы 4](#_Toc58798049)

[2. Архитектура базы данных 7](#_Toc58798050)

[2.1. Таблицы базы данных 7](#_Toc58798051)

[2.2. Процедуры для решения поставленных задач 8](#_Toc58798052)

[2.3. Функции для решения поставленных задач 10](#_Toc58798053)

[2.4. Индекс базы данных 10](#_Toc58798054)

[2.5. Пользователи базы данных 11](#_Toc58798055)

[3. Реализация задач 12](#_Toc58798056)

[3.1. Регистрация и авторизация пользователей 12](#_Toc58798058)

[3.2. Добавление и удаление контактов 13](#_Toc58798059)

[3.3. Реализация нескольких ролей 14](#_Toc58798060)

[3.5. Административный поиск и блокировка/разблокировка пользователей 17](#_Toc58798061)

[4. Импорт и экспорт данных. Тестирование производительности 18](#_Toc58798062)

[5. Описание технологии 20](#_Toc58798063)

[6. Руководство пользователя 23](#_Toc58798064)

[7. Тестирование 29](#_Toc58798065)

[Заключение 31](#_Toc58798066)

[Список использованных источников 32](#_Toc58798067)

[ПРИЛОЖЕНИЕ A 33](#_Toc58798068)

[ПРИЛОЖЕНИЕ B 35](#_Toc58798069)

[ПРИЛОЖЕНИЕ C 37](#_Toc58798070)

# Реферат

Курсовая работа изложена на 52 страницах, которые включают в себя 7 разделов, 16 подразделов, введение, архитектуру, реализацию, руководство пользователя, тестирование, заключение, используемую литературу и приложения. В работу добавлено 17 изображения.

В курсовой работе разработана модель базы данных, которая состоит из 5 основных таблиц и динамически генерируемых таблиц для чатов пользователей. В базу данных включено 3 табличных пространства, 2 профиля безопасности, 2 роли, 1 пакет с 24 процедурами и 2 функциями, 5 индексов, 2 пользователя.

Используемая СУБД – Oracle Database 12c.

Изучаемая технология – шифрование.

Дополнительная технология – работа с мультимедийными данными.

Основной целью курсовой работы является изучение и применение технологий шифрования и работы с мультимедийными данными.

## Аналитический обзор литературы

Telegram — программа, которая призвана заменить обмен привычными сообщениями. SMS, Hangouts, Viber, WhatsApp, может, даже Gmail. Ведь пересылать можно не только текст, но и файлы разных форматов — от картинок до документов. Приложение привязывается к номеру телефона, приходит проверочное сообщение с кодом (на мобильной версии для Android код подтягивается автоматически). Общаться можно как с конкретными пользователями, которые уже установили приложение, так и организовывать групповые чаты до 200 человек и отправлять пригласительные сообщения с призывом установить себе эту крутотенюшку. Еще раз — двести человек в чате! Разработчики считают это одной из самых главных фишек своего приложения (рис 1.1 ).

Благодаря защищенности мессенджера и защите от перехвата приложение стало популярно в Гонконге во время протестов. Изначально о митинге договаривались через Facebook, но эту информацию легко заполучить.

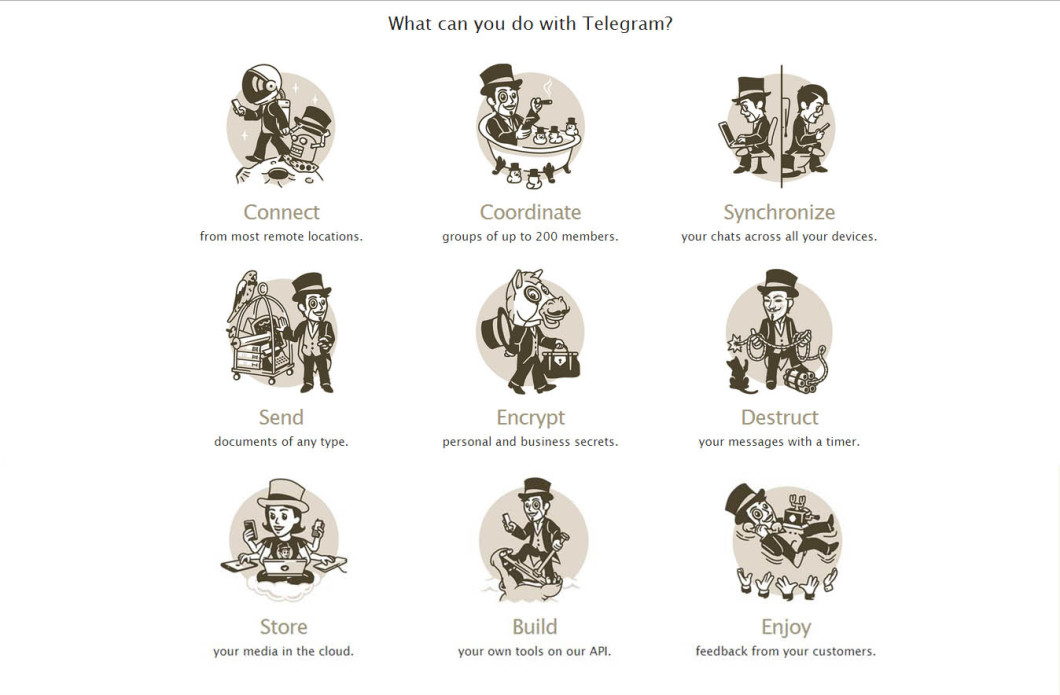


Рисунок 1.1 – Возможности Мессенджера Telegram

Viber — приложение-мессенджер, которое позволяет отправлять сообщения, совершать видео- и голосовые VoIP-звонки через интернет. Голосовые вызовы между пользователями с установленным Viber бесплатны. Также в Viber имеется возможность передачи изображений, видео- и аудиосообщений, документов и файлов.

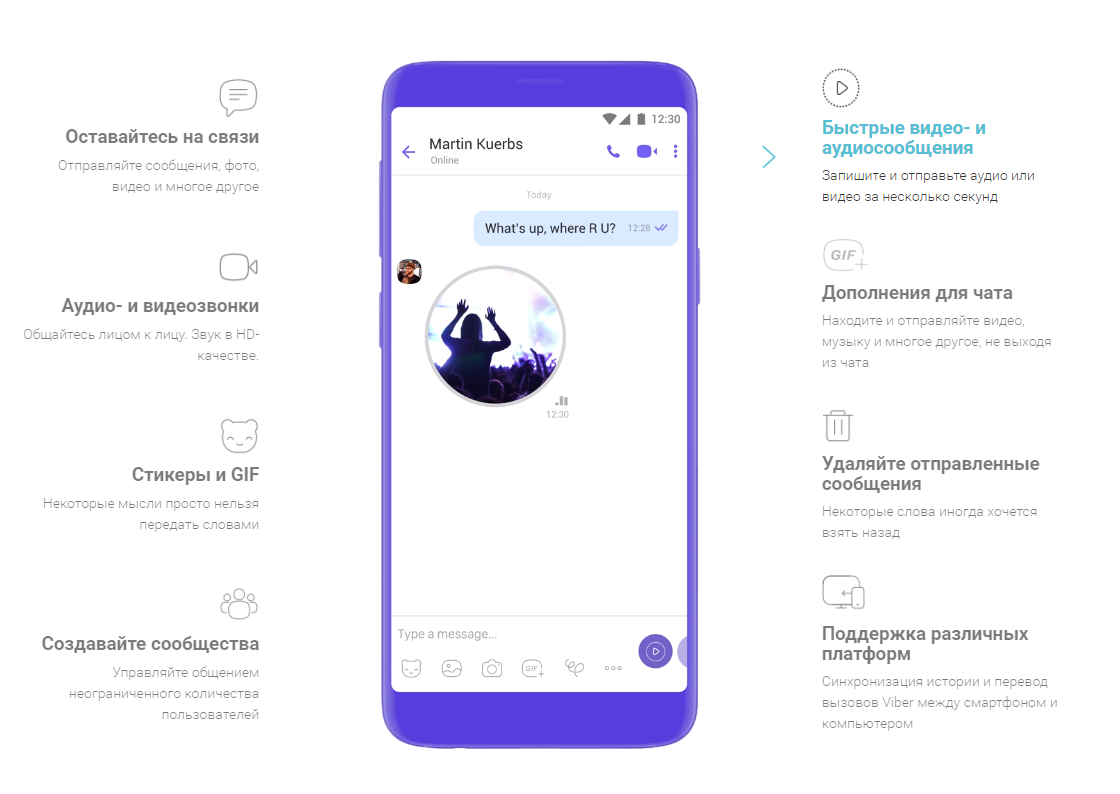


Рисунок 1.2 – Возможности Мессенджера Viber

ICQ (акроним от англ. I seek You/"Я ищу тебя") – популярный мессенджер для мгновенного обмена текстовыми сообщениями и анимационными изображениями, ныне находящийся в собственности российской группы Mail.ru.

Достоинства

1. красивый пользовательский интерфейс (привлекательная цветовая гамма, новые смайлы, радующая взор анимация и т.д.);
2. исключительная простота пользования;
3. возможность общения одновременно с несколькими собеседниками (с опцией параллельного просмотра уведомлений в разных окнах);
4. простор для творчества в контексте "подгонки" оформления под свой вкус (замена иконок, изменение "обоев", опция добавления смайликов от сторонних разработчиков и пр.);
5. опция группирования контактов (объединение собеседников в группы);
6. возможность установки на различные девайсы (от настольных ПК до смартфонов);
7. опция отправки файлов;
8. возможность "привязки" программы к соцсетям (позволяет отыскать потерявшихся одноклассников, коллег, сослуживцев и т.д. по их сетевым аккаунтам);
9. опция проведения групповых видеоконференций;
10. малая нагрузка программы на компьютер;
11. быстрая закачка и установка.

Недостатки

1. надоедливые окна с рекламой;
2. периодическая "потеря" сообщений;
3. на самая надежная защита от спама;
4. отсутствие истории переписки (в отдельных версиях).

## Архитектура базы данных

## Таблицы базы данных

Для хранения и предоставления данных пользователю, была разработана база данных, которая содержит в себе 5 основных таблиц и 3 динамически создаваемых таблиц для чатов, все связаны друг с другом внешними ключами. Диаграмма представлена на рисунке 2.1.

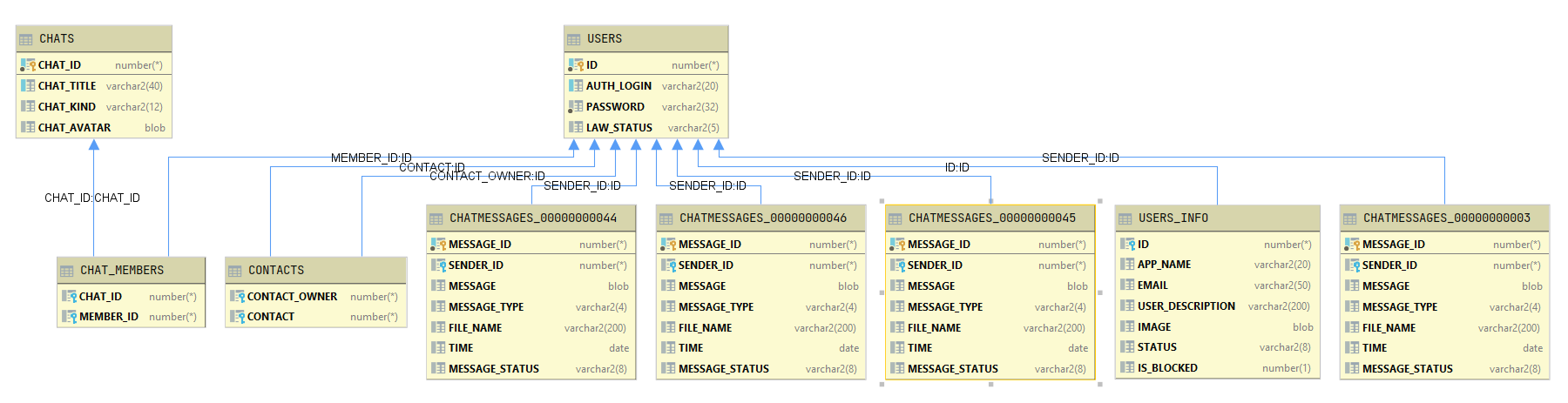


Рисунок 2.1 – Диаграмма базы данных

Таблица users содержит минимальную информацию о пользователях для их авторизации, а именно: идентификатор (id, int, pk), логин (auth\_login, varchar2(20), unique), пароль (password, varchar2(32), not null, шифруется), правовой статус (law\_status, varchar2(5), check(law\_status in('Admin', 'User')). Также к таблице применена технология интервального секционирования с интервалом в 1000 идентификаторов и включенным перемещением строк между сегментами.

Таблица users\_info содержит расширенную информацию о пользователях, а именно: идентификатор пользователя (id, int, pk, references users(id) on delete cascade enable), имя внутри приложения (app\_name varchar2(20)), почта (email varchar2(200)), графа «о себе» (user\_description varchar2(200)), аватарка (image blob), статус(status varchar2(8) check(status in('Offline', 'Online'), заблокирован ли пользователь (number(1)). Также к таблице применена технология интервального секционирования с интервалом в 1000 идентификаторов и включенным перемещением строк между сегментами.

Таблица contacts содержит список контактов для всех пользователей, а именно: идентификатор владельца контакта (contact\_owner, int, references users(id)), идентификатор контакта (contact, int, references users(id)). Также к таблице применена технология интервального секционирования с интервалом в 1000 идентификаторов и включенным перемещением строк между сегментами.

Таблица chats информацию о чатах, а именно: идентификатор чата (chat\_id, int, pk), название чата (chat\_title varchar2(40), unique, для чатов 1 на 1 паттерн названия [id]\_[id], для бесед [название беседы]), тип чата (chat\_kind varchar2(12), check(chat\_kind in('Chat', 'Conversation'))), аватарка чата (chat\_avatar, blob). Также к таблице применена технология интервального секционирования с интервалом в 1000 идентификаторов и включенным перемещением строк между сегментами.

Таблица chat\_members содержит список участников чата, а именно: идентификатор чата (chat\_id, int, references chats(id)), идентификатор участника чата(chat\_member, int, references users(id)). Также к таблице применена технология интервального секционирования с интервалом в 1000 идентификаторов и включенным перемещением строк между сегментами.

Таблица ChatMessages\_{chatId} является динамически генерируемой для каждого чата, где chatId – идентификатор чата из таблицы chats и имеет следующие колонки: идентификатор сообщения (message\_id, int, pk), идентификатор отправителя сообщения(sender\_id, references users(id)), тело сообщения (message, blob, шифруется)

## Процедуры для решения поставленных задач

Все процедуры хранятся в пакете katran\_package для удобного менеджмента.

Процедура GetUserIdByLogin позволяет получить идентификатор пользователя по его логину и имеет следующие параметры: логин (inAuthLogin, входящий, users.auth\_login%type), идентификатор (outFoundUserId, выходящий, users.id%type).

Процедура AddUser позволяет добавить пользователя в таблицу users и имеет следующие параметры: логин (inAuthLogin, входящий, users.auth\_login%type), пароль (inPassword, входящий, users.password%type), привилегии в приложении (inLawStatus, входящий, users.law\_status%type), идентификатор добавленного пользователя (outAddedUserId, выходящий, users.id%type).

Процедура AddUserInfo позволяет добавить пользователя в таблицу users\_info и имеет следующие параметры: идентификатор (inId, входящий, users\_info.id%type), имя в приложении (inAppName, входящий, users\_info.app\_name%type), электронная почта (inEmail, входящий, users\_info.email%type), графа «о себе» (inUserDescription, входящий, users\_info.user\_description%type), аватарка (inImage, входящий, users\_info.image%type), статус (inStatus, входящий, users\_info.status%type), идентификатор добавленного пользователя (outAddedUserId, выходящий, users\_info.id%type).

Процедура CheckUserByLoginAndPassword позволяет проверить есть ли в базе данных пользователь с определенным логином и паролем и имеет следующие параметры: логин (inAuthLogin, входящий, users.auth\_login%type), пароль (inPassword, входящий, users.password%type), идентификатор найденного пользователя (outFoundUserId, выходящий, users.id%type).

Процедура GetUserInfoById позволяет получить информацию о пользователе из таблицы users\_info и имеет следующие параметры: идентификатор (in\_outUserId, входящий-выходящий, int), (in\_outAppName, входящий-выходящий, varchar2), (in\_outEmail, входящий-выходящий, varchar2), графа «о себе» (in\_outUserDescription, входящий-выходящий, varchar2), аватарка (in\_outImage, входящий-выходящий, blob), статус (in\_outStatus, входящий-выходящий, varchar2), (in\_outLawStatus, входящий-выходящий, varchar2), заблокирован ли пользователь (in\_outIsBlocked, входящий-выходящий, number).

Процедура ChangeUserStatusById позволяет изменить статус пользователя по его идентификатору и имеет следующие параметры: новый статус пользователя (inNewUserStatus, входящий, users\_info.status%type), идентификатор (in\_outUserId, входящий-выходящий, users\_info.id%type).

Процедура AddContact позволяет добавить запись в таблицу contacts и имеет следующие параметры: владелец контакта (in\_outContactOwner, входящий-выходящий, users.id%type), контакт (inTargetContact, входящий, users.id%type).

Процедура RemoveContact позволяет удалить запись из таблицы contacts и имеет следующие параметры: владелец контакта (in\_outContactOwner, входящий-выходящий, users.id%type), контакт (inTargetContact, входящий, users.id%type).

Процедура AddChat позволяет добавить запись в таблицу chats и имеет следующие параметры: идентификатор чата (out\_AddedChatId, выходящий, chats.chat\_id%type), имя чата (inChatTitle, входящий, chats.chat\_title%type), тип чата (inChatKind, входящий, chats.chat\_kind%type default 'Chat'), аватарка чата (inChatAvatar, входящий, chats.chat\_avatar%type default null).

Процедура AddChatMember позволяет добавить участника чата и имеет следующие параметры: идентификатор чата (inChatId, входящий, chat\_members.chat\_id%type), иднтификатор участника чата (in\_outMemberId, входящий-выходящий, chat\_members.member\_id%type).

Процедура RemoveChatMember позволяет удалить участника из чата и имеет следующие параметры: идентификатор чата (inChatId, входящий, chat\_members.chat\_id%type), идентификатор участника (inMemberId, входящий, chat\_members.member\_id%type).

Процедура RemoveAllChatMembers позволяет удалить всех участников чата и имеет следующие параметры: идентификатор чата (inChatId, входящий, chat\_members.chat\_id%type).

Процедура GetChatMembersCount позволяет получить количество участников чата и имеет следующие параметры: идентификатор чата (inChatId, входящий, chat\_members.chat\_id%type), количество участников (outMembersCount, выходящий, number).

Процедура GetChatMembersIds позволяет получить список идентификаторов всех участников чата и имеет следующие параметры: курсор на идентификаторы (outChatMembersIds, выходящий, sys\_refcursor), идентификатор чата (inChatId, входящий, chat\_members.chat\_id%type).

Процедура RemoveChat позволяет удалить чат из таблицы chats и имеет следующие параметры: идентификатор чата (in\_outChatId, входящий-выходящий, chat\_members.chat\_id%type).

Процедура GetConvsInfo позволяет получить информацию о беседе, ее участниках и сообщениях, имеет следующие параметры: курсор на результирующий набор (outResult, выходящий, sys\_refcursor), идентификатор участника (inMemberId, входящий, chat\_members.member\_id%type).

Процедура GetConvMembersInfo позволяет получить информацию об участнике беседы и имеет следующие параметры: курсор на результирующий набор (outResult, выходящий, sys\_refcursor), идентификатор участника (inChatId, входящий, chat\_members.member\_id%type).

Процедура BlockUnblockUser позволяет заблокировать или разблокировать пользователя и имеет следующие параметры: идентификатор пользователя (in\_outUserId, входящий-выходящий, users\_info.id%type), (раз)заблокировать (inIsBlocked, входящий, users\_info.is\_blocked%type).

Процедура GetUserLawStatusById позволяет правовые полномочия пользователя в приложении по его идентификатору и имеет следующие параметры: идентификатор пользователя (in\_outUserId, входящий-выходящий, users.id%type), правовые полномочия (in\_outUserLawStatus, входящий-выходящий, varchar2).

Процедура AdminSearch позволяет произвести поиск по всем пользователям приложения и узнать их статус блокировки, имеет следующие параметры: курсор на результирующий набор (outResult, выходящий, sys\_refcursor), идентификатор администратора (inAdminId, входящий, users\_info.id%type), паттерн поиска по пользователям (inSearchPattern, входящий, varchar2).

Процедура SearchOutOfContacts позволяет производить поиск вне контактов пользователя и имеет следующие параметры: курсор на результирующий набор (outResult, выходящий, sys\_refcursor), паттерн поиска среди пользователей (inPattern, входящий, varchar2), идентификатор ищущего (inSearcherUserID, входящий, users\_info.id%type).

Процедура GetUserContacts позволяет получить список контактов пользователя и имеет следующие параметры: курсор на результирующий набор (outResult, выходящий, sys\_refcursor), идентификатор ищущего (inContactsOwner, входящий, users\_info.id%type).

## Функции для решения поставленных задач

Функция DecryptBlob позволяет расшифровать сообщение в чате и принимает следующие параметры: сообщение в зашифрованном виде (enc\_value, blob, входящий), ключ шифрования (crypt\_key, входящий, varchar2(24)).

Функция EncryptBlob позволяет зашифровать сообщение для добавления в чат и принимает следующие параметры: сообщение (enc\_value, blob, входящий), ключ шифрования (crypt\_key, входящий, varchar2(24)).

## Индекс базы данных

Индексы Oracle обеспечивают быстрый доступ к строкам таблиц, сохраняя отсортированные значения указанных столбцов и используя эти отсортированные значения для быстрого нахождения ассоциированных строк таблицы. Индексы позволяют находить строку с определенным значением столбца, просматривая при этом лишь небольшую часть общего объема строк таблицы. Таким образом правильное использование индексов сокращает до минимума количество дорогостоящих операций ввода-вывода.

В данном курсовом проекте также были созданы индексы для каждой таблицы, относящиеся к первичному ключу.

## Пользователи базы данных

Для базы данных было разработано два типа пользователей. Один из них разработан в качестве пользователя приложения – katran\_user, а второй для администрирования в базе данных – katran\_admin.

Пользователь katran\_user имеет грант на подключение к базе данных, гранты на выборку/вставку/обновление/чтение из таблиц, грант выполнение процедур из пакета katran\_package и sys.dbms\_crypto. Все эти гранты содержаться в роле katran\_user\_role.

Пользователь имеет все гранты в рамках базы данных, исключая те, что требуют sys права. Также имеет грант на выполнение процедур из пакета sys.dbms\_crypto. Все эти гранты содержаться в роле katran\_admin\_role.

Для каждого из пользователей были созданы профили безопасности: katran\_admin\_security\_profile для katran\_admin и katran\_user\_security\_profile для katran\_user. Данные профили безопасности представлены на рисунке 2.2.

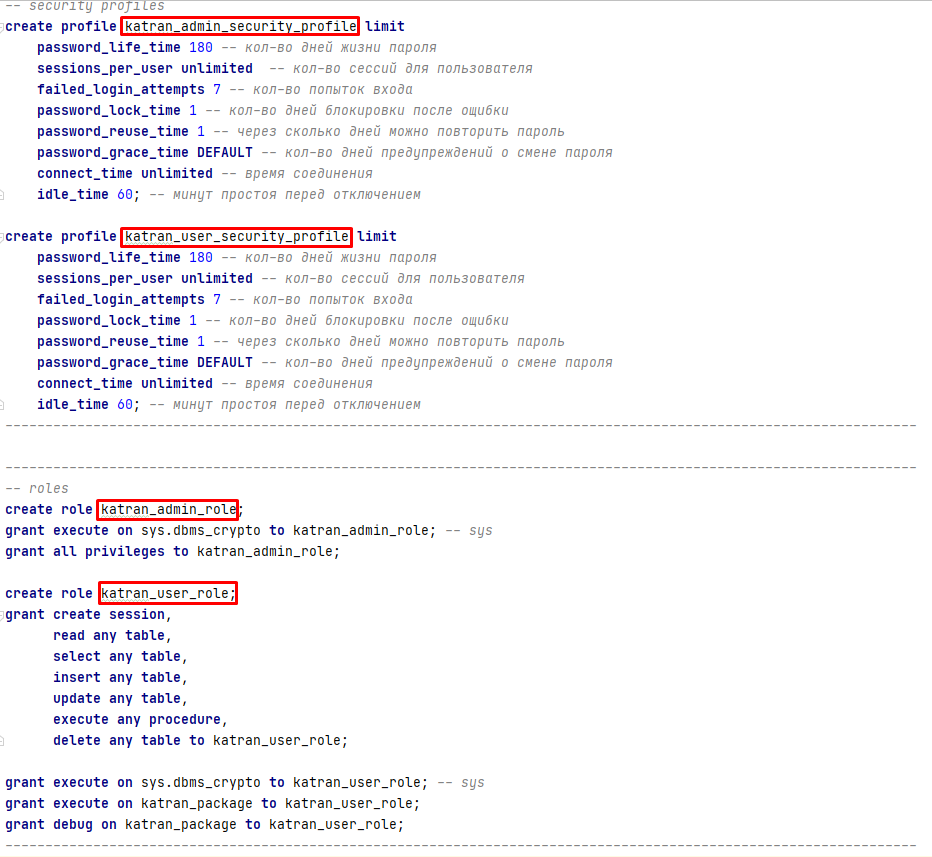


Рисунок 2.2 – профили безопасности и роли пользователей

## Реализация задач



## Регистрация и авторизация пользователей

Ниже приведены процедуры, реализующие регистрацию и авторизацию пользователей в базе данных.

**procedure** *AddUser*(inAuthLogin **in users**.**auth\_login**%**type**,  
 inPassword **in users**.**password**%**type**,  
 inLawStatus **in users**.**law\_status**%**type**,  
 outAddedUserId **out users**.**id**%**type**)  
**as  
begin  
  
 insert into users** (**auth\_login**, **password**, **law\_status**)  
 **values** (inAuthLogin, inPassword, inLawStatus)  
 **returning id into** outAddedUserId;  
 **commit**;  
**exception  
 when others then** outAddedUserId := -1;  
 **rollback**;  
**end**;

**procedure** *AddUserInfo*(inId **in** users\_info.**id**%**type**, inAppName **in** users\_info.**app\_name**%**type**, inEmail **in** users\_info.**email**%**type**, inUserDescription **in** users\_info.**user\_description**%**type**, inImage **in** users\_info.**image**%**type**, inStatus **in** users\_info.**status**%**type**, **out** users\_info.**id**%**type**)  
**as  
begin  
 insert into** users\_info (**id**, **app\_name**, **email**, **user\_description**, **image**, **status**)  
 **values**(inId, inAppName, inEmail, inUserDescription, inImage, inStatus)  
 **returning id into** outAddedUserId;  
 **commit**;  
**exception  
 when others then** outAddedUserId := -1;  
 **rollback**;  
**end**;

**procedure** *CheckUserByLoginAndPassword*(inAuthLogin **in users**.**auth\_login**%**type**, inPassword **in users**.**password**%**type**, outFoundUserId **out users**.**id**%**type**)  
**as  
begin  
 select id into** outFoundUserId **from users where auth\_login** = inAuthLogin **and** inPassword = inPassword;  
**exception  
 when others then** outFoundUserId := -1;  
**end**;

## Добавление и удаление контактов

Ниже приведены процедуры, реализующие регистрацию и авторизацию пользователей в базе данных.

**procedure** *AddContact*( in\_outContactOwner **in out users**.**id**%**type**, inTargetContact **in users**.**id**%**type**)  
**as  
begin  
  
 insert into** contacts (**contact\_owner**, **contact**)  
 **values**(in\_outContactOwner, inTargetContact);  
 **commit**;  
 **insert into** contacts (**contact\_owner**, **contact**)  
 **values**(inTargetContact, in\_outContactOwner);  
 **commit**;  
**exception  
 when others then** in\_outContactOwner := -1;  
 **rollback**;  
**end**;

**procedure** *RemoveContact*( in\_outContactOwner **in out users**.**id**%**type**, inTargetContact **in users**.**id**%**type**)  
**as  
begin  
 delete from** contacts **where contact\_owner** = in\_outContactOwner **and contact** = inTargetContact;  
 **commit**;  
**exception  
 when others then** in\_outContactOwner := -1;  
 **rollback**;  
**end**;

## Реализация нескольких ролей

В приложении есть две роли: администратор и пользователь, но с точки зрения БД у них одинаковые права. У администратора в приложении доступна вкладка, где он может заблокировать или разблокировать любого пользователя.

* 1. Менеджмент чатов и динамическое добавление таблиц

Ниже представлены процедуры для взаимодействия с чатами: их создание/удаление, добавление/удаление пользователей из чата, получение информации о участниках чата.

**procedure** *AddChat*( out\_AddedChatId **out** chats.**chat\_id**%**type**, **in** chats.**chat\_title**%**type**, **in** chats.**chat\_kind**%**type default 'Chat'**, inChatAvatar **in** chats.**chat\_avatar**%**type default null**)  
**as  
begin  
  
 insert into** chats (**chat\_title**, **chat\_kind**, **chat\_avatar**)  
 **values**(inChatTitle, inChatKind, inChatAvatar)  
 **returning chat\_id into** out\_AddedChatId;  
 **commit**;  
**exception  
 when others then** out\_AddedChatId := -1;  
 **rollback**;  
**end**;

**procedure** *RemoveChat*( in\_outChatId **in out** chat\_members.**chat\_id**%**type**)  
**as  
begin  
 delete from** chats **where chat\_id** = in\_outChatId;  
 **commit**;  
**exception  
 when others then** in\_outChatId := -1;  
 **rollback**;  
**end**;

**procedure** *AddChatMember*( inChatId **in** chat\_members.**chat\_id**%**type**, in\_outMemberId **in out** chat\_members.**member\_id**%**type**)  
**as  
begin  
  
 insert into** chat\_members (**chat\_id**, **member\_id**)  
 **values**(inChatId, in\_outMemberId)  
 **returning member\_id into** in\_outMemberId;  
 **commit**;  
**exception  
 when others then** in\_outMemberId := -1;  
 **rollback**;  
**end**;

**procedure** *RemoveChatMember*( inChatId **in** chat\_members.**chat\_id**%**type**, inMemberId **in** chat\_members.**member\_id**%**type**)  
**as  
begin  
 delete from** chat\_members **where chat\_id** = inChatId **and member\_id** = inMemberId;  
 **commit**;  
**end**;

**procedure** *RemoveAllChatMembers*( inChatId **in** chat\_members.**chat\_id**%**type**)  
**as  
begin  
 delete from** chat\_members **where chat\_id** = inChatId;  
 **commit**;  
**end**;

**procedure** *GetChatMembersCount*( inChatId **in** chat\_members.**chat\_id**%**type**, outMembersCount **out number**)  
**as  
begin  
 select** *count*(**chat\_id**) **into** outMembersCount **from** chat\_members **where chat\_id** = inChatId;  
**exception  
 when others then** outMembersCount := 0;  
**end**;

**procedure** *GetChatMembersIds*( outChatMembersIds **out sys\_refcursor**,  
 inChatId **in** chat\_members.**chat\_id**%**type**)  
**as  
begin  
 open** outChatMembersIds **for  
 select member\_id from** chat\_members **where chat\_id** = inChatId;  
**end**;

**procedure** *GetConvsInfo*( outResult **out sys\_refcursor**, inMemberId **in** chat\_members.**member\_id**%**type**)  
**as  
begin  
 open** outResult **for  
 select** chat.**chat\_id**, chat.**chat\_title**, chat.**chat\_avatar  
 from** Chats chat  
 **where** chat.**chat\_id in** (**select** cm.**chat\_id from** chat\_members cm **where** cm.**member\_id** = inMemberId) **and** chat.**chat\_kind** = **'Conversation'**;  
**end**;

**procedure** *GetConvMembersInfo*( outResult **out sys\_refcursor**, inChatId **in** chat\_members.**member\_id**%**type**)  
**as  
begin  
 open** outResult **for  
 select** ui.**id**, ui.**app\_name**, ui.**image**, ui.**status  
 from** Users\_info ui **join** chat\_members cm  
 **on** ui.**id** = cm.**member\_id and** cm.**chat\_id** = inChatId;  
**end**;

Добавление таблиц для новых чатов и взаимодействие с ними невозможно через процедуры, поэтому используя технологию ADO.NET в С# на стороне сервера был реализован скрипт для взаимодействия с ними. Объекты Oracle команд представлены ниже.

OracleCommand chatMessagesCommand = new OracleCommand($"select message\_id, sender\_id, {GetPackageName()}.DecryptBlob(message, '{GetChatMessagesTableName(chatId)}'), message\_type, file\_name, time, message\_status, app\_name from {GetChatMessagesTableName(chatId)} c join Users\_info u on c.sender\_id = u.id order by message\_id desc", adminConnection);

OracleCommand command = new OracleCommand($"update {GetSchemaName()}.{GetChatMessagesTableName(rmessT.ChatId)} set message\_status = :status where message\_id = :id", sql)

Создание таблицы для чата реализовано как показано на Рисунке 3.1. ChatId – идентификатор чата из таблицы chats. Над полем message осуществляется шифрование. Таблица 2х уровнево сегментирована: сначала интервально по идентификатору столбца, а после по хэшу на 10 подчастей.

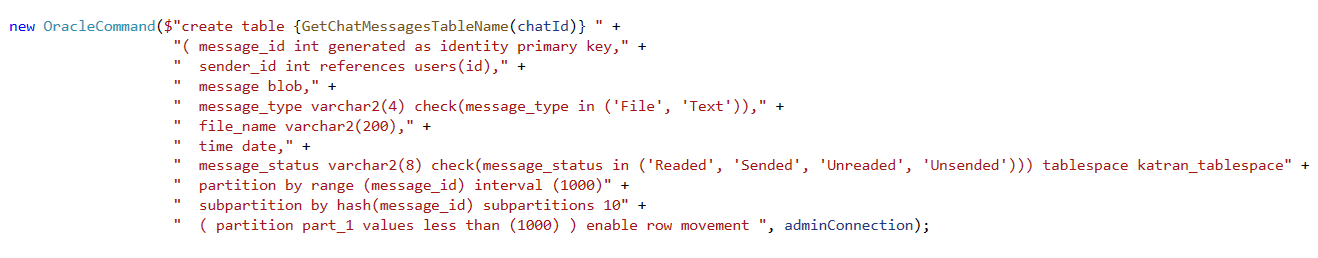


Рисунок 3.1 – создание таблицы для чата

## Административный поиск и блокировка/разблокировка пользователей

Администратор на клиентской части имеет доступ ко вкладке гдеон может осуществить поиск по всем пользователям приложения, узнать их статус блокировки и изменить его с помощью следующих процедур:

**procedure** *AdminSearch*( outResult **out sys\_refcursor**,  
 inAdminId **in** users\_info.**id**%**type**,  
 inSearchPattern **in varchar2**)  
**as  
begin  
 open** outResult **for  
 select** ui.**id**, ui.**app\_name**, ui.**image**, ui.**status**, ui.**is\_blocked  
 from** Users\_info ui  
 **where** *instr*(ui.**app\_name**, inSearchPattern) > 0 **and** ui.**id** != inAdminId;  
**end**;

**procedure** *BlockUnblockUser*( in\_outUserId **in out** users\_info.**id**%**type**, inIsBlocked **in** users\_info.**is\_blocked**%**type**)  
**as  
begin  
 update** users\_info **set is\_blocked** = inIsBlocked **where id** = in\_outUserId;  
 **commit**;  
**exception  
 when others then** in\_outUserId := -1;  
 **rollback**;  
**end**;

## Импорт и экспорт данных. Тестирование производительности

Для реализации процедур импорта и экспорта потребовалось создать объект директории, которая бы указывала путь для xml файла, в который будет производиться импорт и экспорт. В нашем случае имя директории – «KATRAN\_XML», путь – «C:\katran\_db\xml». Таблица, над которой будем производить операции – CHATMESSAGES\_00000000046.

Процедура экспорта данных:

**CREATE OR REPLACE PROCEDURE** *chatmessages\_export***IS**rc **sys\_refcursor**;  
doc DBMS\_XMLDOM.DOMDocument;  
**BEGIN  
OPEN** rc **FOR SELECT** *\** **FROM** katran\_admin.CHATMESSAGES\_00000000046;  
doc := DBMS\_XMLDOM.NewDOMDocument(**XMLTYPE**(rc));  
DBMS\_XMLDOM.WRITETOFILE(doc, **'KATRAN\_XML\CHATMESSAGES\_00000000046.xml'**);  
**END** *chatmessages\_export*;

Курсорная переменная rc создаётся для того, чтобы выгрузить большое число строк. Для создания xml файла и экспорта в него потребуется использовать пакет DBMS\_XMLDOM. Объявляем переменную типа DOMDocument.

В блоке begin построчно считываем необходимые строки из таблицы и помещаем их в курсорную переменную. Создаем новый DOMDocument, содержимое которого – строки курсорной переменной. После чего при помощи функции WRITETOFILE записываем содержимое DOMDocument’а в созданный файл с именем CHATMESSAGES\_00000000046.xml.

Процедура импорта из xml файла:

**CREATE OR REPLACE PROCEDURE** *chatmessages\_import***IS  
BEGIN  
 DELETE FROM** CHATMESSAGES\_00000000045;  
 **INSERT INTO** CHATMESSAGES\_00000000045 (**SENDER\_ID**, **MESSAGE**, **MESSAGE\_TYPE**, **TIME**, **MESSAGE\_STATUS**)  
 **SELECT** *ExtractValue*(*VALUE*(product\_order\_xml), **'//SENDER\_ID'**) **AS** SENDER\_ID,  
 *ExtractValue*(*VALUE*(product\_order\_xml), **'//MESSAGE'**) **AS** MESSAGE,  
 *ExtractValue*(*VALUE*(product\_order\_xml), **'//MESSAGE\_TYPE'**) **AS** MESSAGE\_TYPE,  
 *ExtractValue*(*VALUE*(product\_order\_xml), **'//TIME'**) **AS TIME**,  
 *ExtractValue*(*VALUE*(product\_order\_xml), **'//MESSAGE\_STATUS'**) **AS** MESSAGE\_STATUS  
 **FROM TABLE**(*XMLSequence*(*EXTRACT*(**XMLTYPE**(*bfilename*(**'KATRAN\_XML'**, **'CHATMESSAGES\_import.xml'**),  
  
 *nls\_charset\_id*(**'UTF-8'**)),**'/ROWSET/ROW'**))) product\_order\_xml;  
**END** *chatmessages\_import*;

Перед вставкой данных из xml файла производим очистку всех данных из таблицы (предполагается, что импорт будет использоваться после вставки 100000 строк). Указываем таблицу и столбцы, в которые будем производить вставку. При помощи функции ExtractValue даем установку, что записи с определенными тегами будут трактоваться как данные определенных столбцов. После чего при помощи той же функции указываем местоположение файла и его название, используемую кодировку и корень файла.

Данная процедура вставляет в таблицу product\_order\_t строки с фиксированным значением. Время выполнения – примерно 5 секунд.

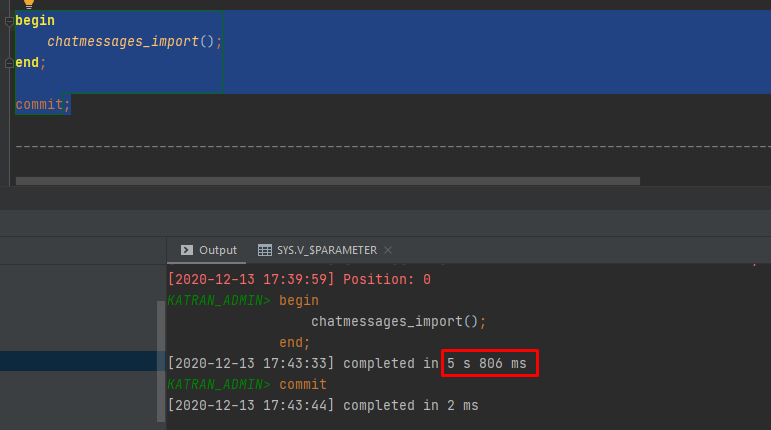


Рисунок 4.1 – Тестирование производительности

## Описание технологии

Всем прекрасно известно, что самое дорогое, что может хранится в приложение – это пользовательские, личные, конфиденциальные данные. Каждый злоумышленник стремится заполучить их любым возможным, доступным ему образом. Чтобы защититься от незаконного хищения данных, каждый добросовестный разработчик, или команда разработки, используют различные алгоритмы шифрования. Для реализации шифрования данных в Oracle можно использовать пакеты *DBMS\_OBFUSCATION\_TOOLKIT* и *DBMS\_CRYPTO*. Однако оба эти пакета увеличивают трудоемкость в том смысле, что приложению приходится управлять ключами шифрования, а также обращаться к API-интерфейсам для выполнения шифрования и дешифровки данных. И уже в Oracle Database 10g для преодоления этих недостатков было предложено новое средство TDE (Transparent Data Encryption — прозрачное шифрование данных), которое позволяет шифровать столбцы в таблице. Для Oracle характерны следующие алгоритмы шифрования:

* DES3. В этой схеме, в зависимости от режима вызова данные шифруются дважды или трижды. DES3 использует 128- или 192-разрядный ключ; его длина определяется количеством проходов.
* DES (Data Encryption Standard). Алгоритм DES использует 64-разрядный ключ, но 8 бит ключа не используются. Чтобы подобрать ключ, злоумышленнику придется перебрать до 72 057 594 037 927 936 комбинаций.
* AES. В ноябре 2001 года был одобрен новый стандарт AES (Advanced Encryption Standard), вступивший в силу в мае 2002 года.

Блок данных обычно не шифруется как единое целое. Чаще всего он разбивается на фрагменты по 8 байт, после чего каждый фрагмент шифруется независимо от других. Конечно, длина данных может быть не кратной 8 — в этом случае алгоритм добавляет символы в последний фрагмент до 8 байт. Этот процесс называется заполнением (padding). Если злоумышленник угадает, какие данные использовались для заполнения, это может упростить подбор ключа. Для безопасного заполнения следует использовать метод, реализованный в Oracle, называемый PKCS#5 (Public Key Cryptography System #5).

Если данные делятся на фрагменты, также должен существовать способ объединения смежных фрагментов; этот процесс называется сцеплением (chaining). Безопасность системы шифрования также зависит от того, как происходит соединение и шифрование фрагментов (независимо или в сочетании со смежными фрагментами). Самым распространенным форматом сцепления является формат CBC (Cipher Block Chaining). В Oracle он выбирается при помощи константы, определенной во встроенном пакете CHAIN\_CBC. Также используются режимы сцепления Electronic Code Book (CHAIN\_ECB), Cipher Feedback (CHAIN\_CFB) и Output Feedback (CHAIN\_OFB).

Шифрация данных происходит при помощи функции DBMS\_CRYPTO.ENCRYPT. Функция получает четыре аргумента:

* src — исходные данные, подлежащие шифрованию (должны иметь тип данных RAW).
* key — ключ шифрования (также RAW). Длина ключа должна соответствовать выбранному алгоритму. Например, для алгоритма DES она должна быть не менее 64 бит.
* typ — определение трех компонентов (алгоритм, механизм заполнения и метод сцепления) в виде суммы соответствующих констант.
* iv — необязательный вектор инициализации (IV), еще один компонент схемы шифрования, затрудняющий анализ «закономерностей» в зашифрованном тексте.

В своем приложении я использовал:

* алгоритм — AES c 192-разрядным ключом;
* метод сцепления — CBC;
* механизм заполнения — PKCS#5.

Они задаются следующим значением параметра typ при вызове функции:

*l\_mod NUMBER := DBMS\_CRYPTO.encrypt\_aes192 + DBMS\_CRYPTO.chain\_cbc + DBMS\_CRYPTO.pad\_pkcs5;*

Для шифрования необходимо выбрать ключ. В приложении в качестве ключа используется параметр вводимый в процедуру, в моем случае на стороне сервере в этот параметр записывается имя таблицы сообщений чата. Это значение относится к типу данных VARCHAR2. Чтобы использовать его в функции ENCRYPT он должен быть длиной 24 символа и необходимо преобразовать его к типу RAW. Для этого мы воспользуемся функцией STRING\_TO\_RAW встроенного пакета UTL\_I18N. Также вторым параметром указываем кодировку.

Переменная l\_in\_val типа VARCHAR2 также должна быть преобразована к типу RAW. Переменную, которая будет передаваться в качестве возвращаемого значения функции, требуется преобразовать в шестнадцатеричный формат. Для этого используем функцию RAWTOHEX.

Процедура шифрования пароля представлена ниже:

**function** *EncryptBlob* (enc\_value **in blob**,  
 crypt\_key **in varchar2**) *-- should be 24 symbols length***return blob  
is** encrypted\_blob **blob**;  
 l\_mod **number** := dbms\_crypto.encrypt\_aes192 + dbms\_crypto.chain\_cbc + dbms\_crypto.pad\_pkcs5;  
**begin** encrypted\_blob := dbms\_crypto.*encrypt*(enc\_value, l\_mod, utl\_i18n.*string\_to\_raw*(crypt\_key, **'AL32UTF8'**));  
**return** encrypted\_blob;  
**end** *EncryptBlob*;

Шифрование данных имеет смысл только в том случае, если зашифрованные данные в какой-то момент будут прочитаны и использованы в приложении. Эта задача решается при помощи функции DECRYPT. По структуре вызова она идентична функции ENCRYPT и получает те же четыре аргумента:

* src — зашифрованные данные.
* key — ключ, использованный для шифрования.
* typ — три компонента (алгоритм, механизм заполнения и метод сцепления), использованные при вызове ENCRYPT.
* iv — вектор инициализации, использованный при вызове ENCRYPT.

Функция DECRYPT тоже возвращает дешифрованные данные в формате RAW; для нормального просмотра их необходимо преобразовать в другой формат. Прежде всего преобразуем шестнадцатеричный формат зашифрованного пароля в RAW. Для этого воспользуемся функцией HEXTORAW.

Используя тот же самый ключ, передаем его в качестве одного из параметров в функцию DECRYPT, перед этим преобразовав его в RAW при помощи функции STRING\_TO\_RAW. Так же не забываем передать в функцию параметр typ и тело сообщения типа RAW.

Результат функции преобразуем к строке при помощи функции RAW\_TO\_CHAR пакета UTL\_I18N.

Процедура дешифрования пароля представлена ниже:

**function** *DecryptBlob* (enc\_value **in blob**,  
 crypt\_key **in varchar2**) *-- should be 24 symbols length***return blob  
is** decrypted\_blob **blob**;  
 l\_mod **number** := dbms\_crypto.encrypt\_aes192 + dbms\_crypto.chain\_cbc + dbms\_crypto.pad\_pkcs5;  
**begin** decrypted\_blob := dbms\_crypto.*decrypt*(enc\_value, l\_mod, utl\_i18n.*string\_to\_raw*(crypt\_key, **'AL32UTF8'**));  
**return** decrypted\_blob;  
**end** *DecryptBlob*;

## Руководство пользователя

После запуска приложения пользователя встречает страница авторизации, она представлена на рисунке 6.1.

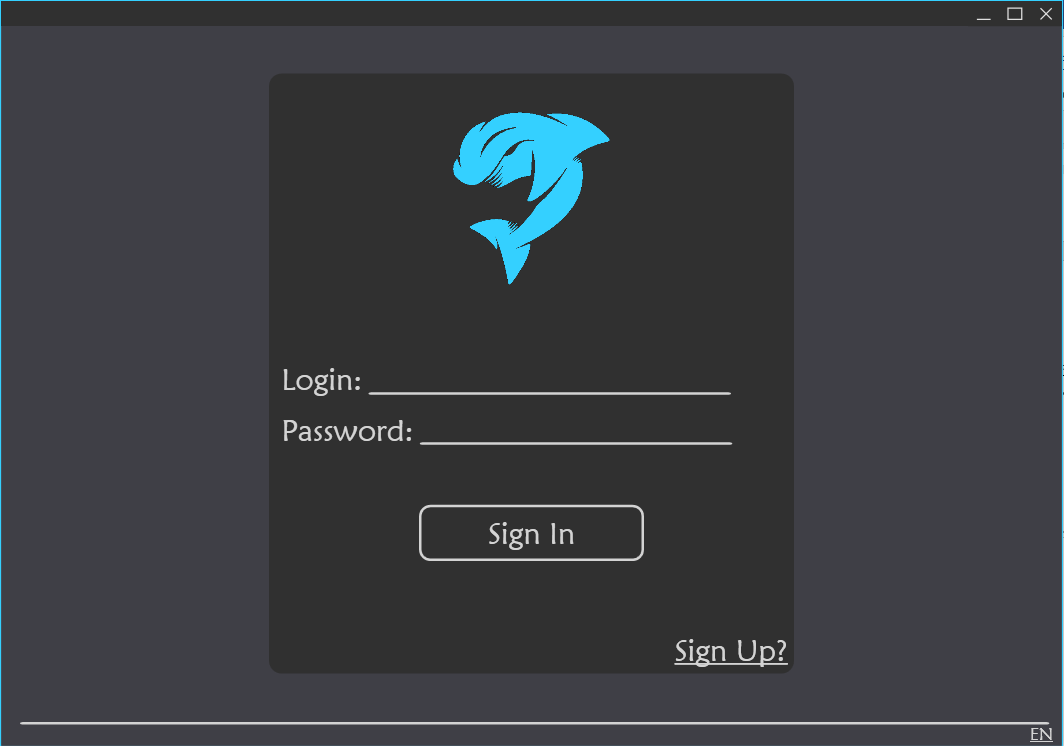


Рисунок 6.1 – Страница авторизации

В случае если пользователь не имеет аккаунт, ему необходимо нажать кнопку «Зарегистрироваться?», после чего он увидит страницу регистрации, где пользователю будет необходимо ввести информацию о себе. Окно регистрации представлено на рисунке 6.2.

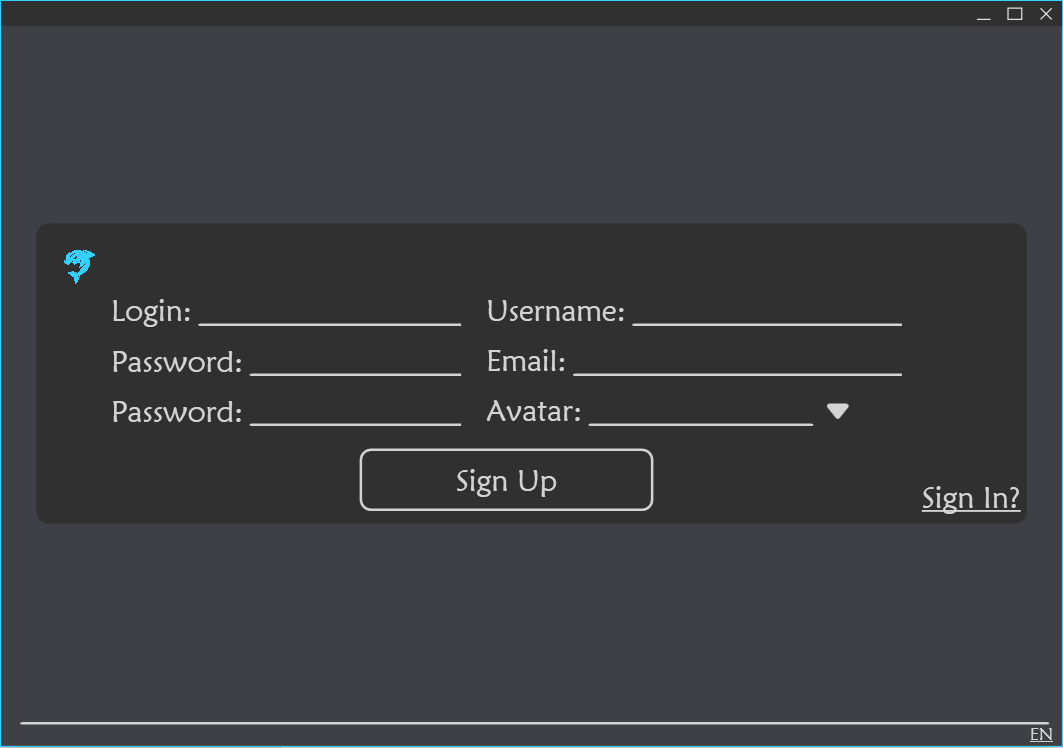


Рисунок 6.2 – Страница авторизации

После успешной авторизации или регистрации пользователю будет доступно основная страница приложения, рисунок 6.3.

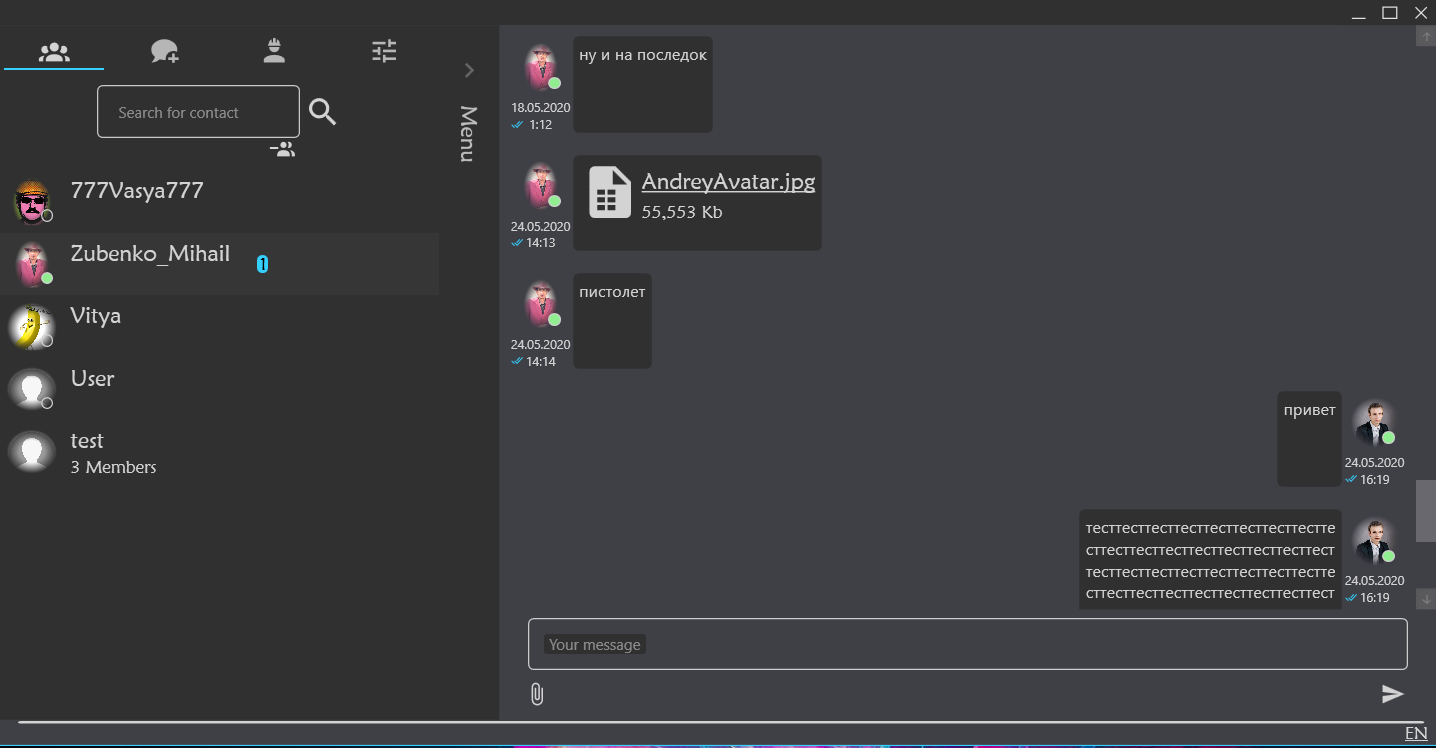


Рисунок 6.3 – Основная страница приложения

Здесь доступно 3 обязательные вкладки и 1 дополнительная, если у пользователя есть права админа, то появится вкладка «Панель Администратора». А панели контактов можно выбрать текущий чат, удалить контакт из своей сети, найти новый контакт вне своей сети, выйти из беседы, каждое такое действие сопровождается подтверждающим диалоговым окном, рисунок 6.4.

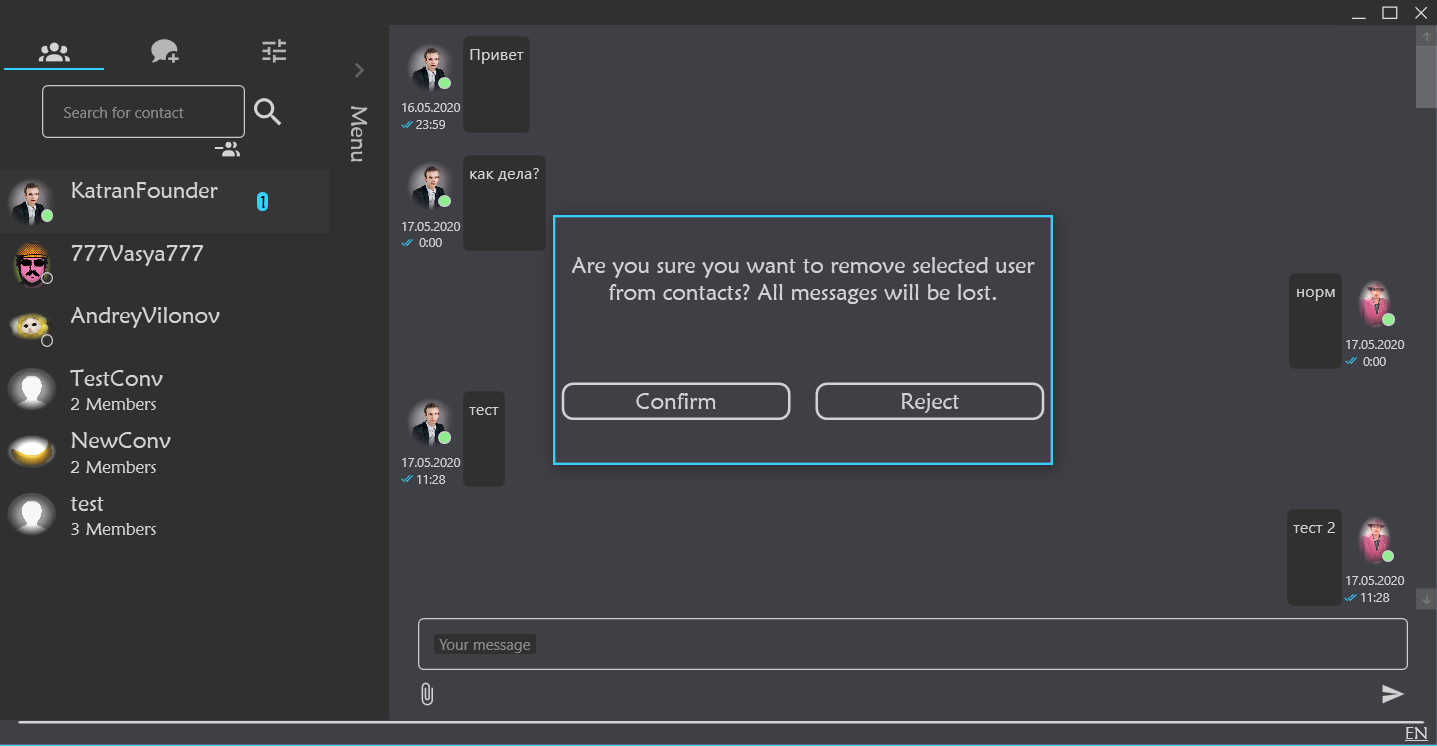


Рисунок 6.4 – Модальные диалоговые окна

Во вкладке «Создание Беседы» пользователь может создать беседу и добавить туда людей из своих контактов, после того, как беседа будет создана во вкладке «Контакты» появится соответствующий элемент, рисунок 6.5.

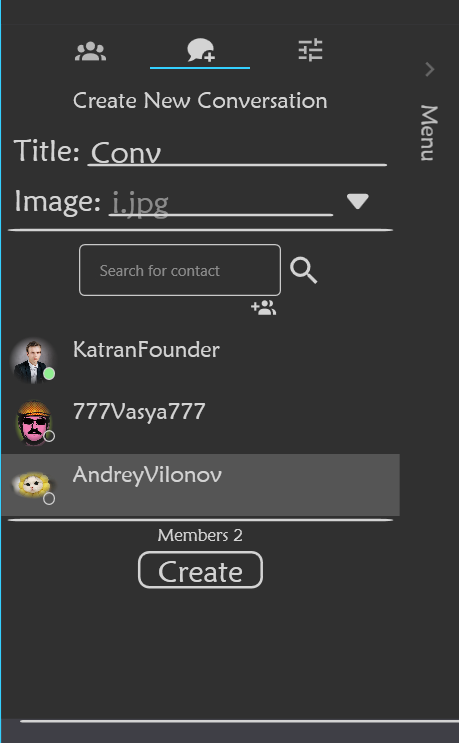


Рисунок 6.5 – Создание бесед

Во вкладке администратора можно найти пользователя по его нику в приложении и разблокировать или заблокировать, это сразу же отобразится у него на экране и в сети контактов у других пользователей он будет помечен как «Заблокирован», рисунок 6.5, 6.6 и 6.7 соответственно.

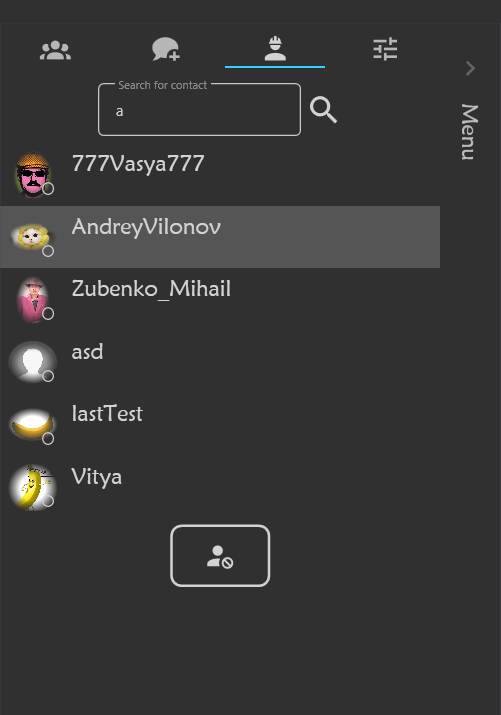


Рисунок 6.5 – Вкладка блокировки пользователей

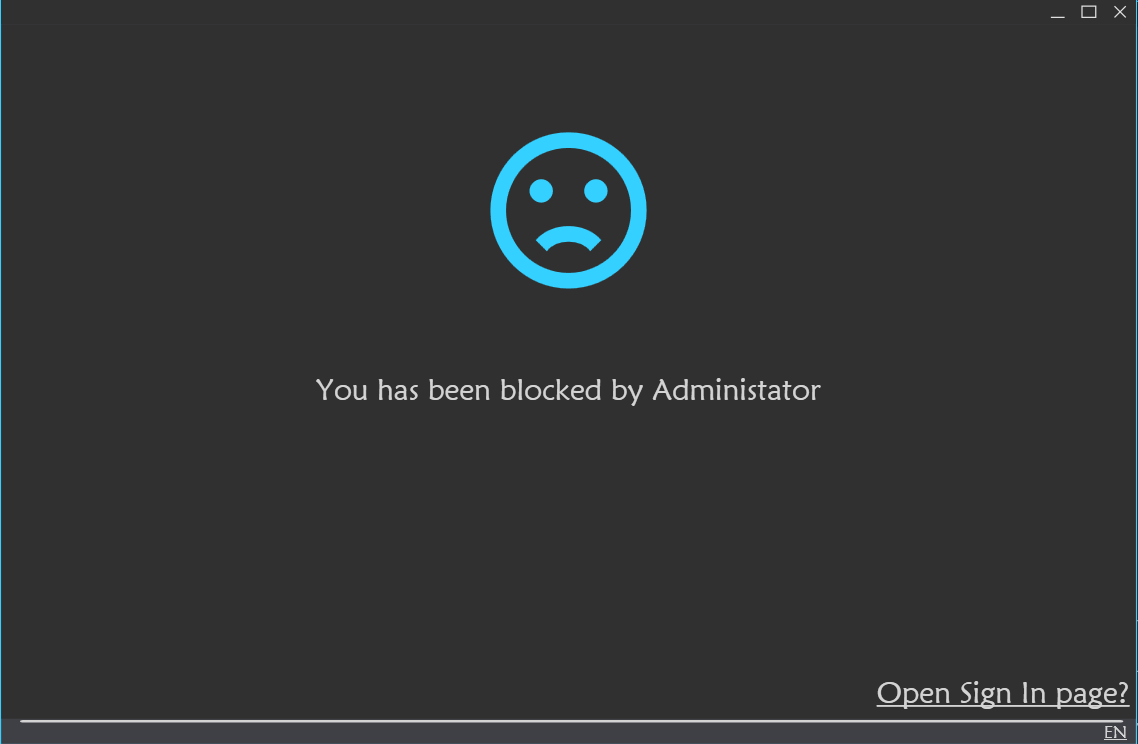


Рисунок 6.6 – Страница, появляющаяся у заблокированных пользователей

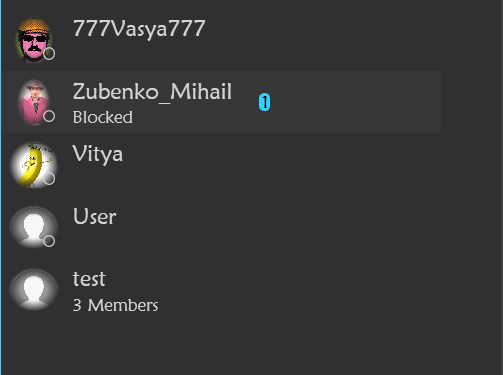


Рисунок 6.7 – Отображение в сети контактов, что пользователь заблокирован

Вкладка «Настройки», здесь можно сменить тему приложения и язык, все изменения сохранятся и при следующем входе будет так, как было настроено. Так же присутствует кнопка выхода из аккаунта, рисунок 6.8.

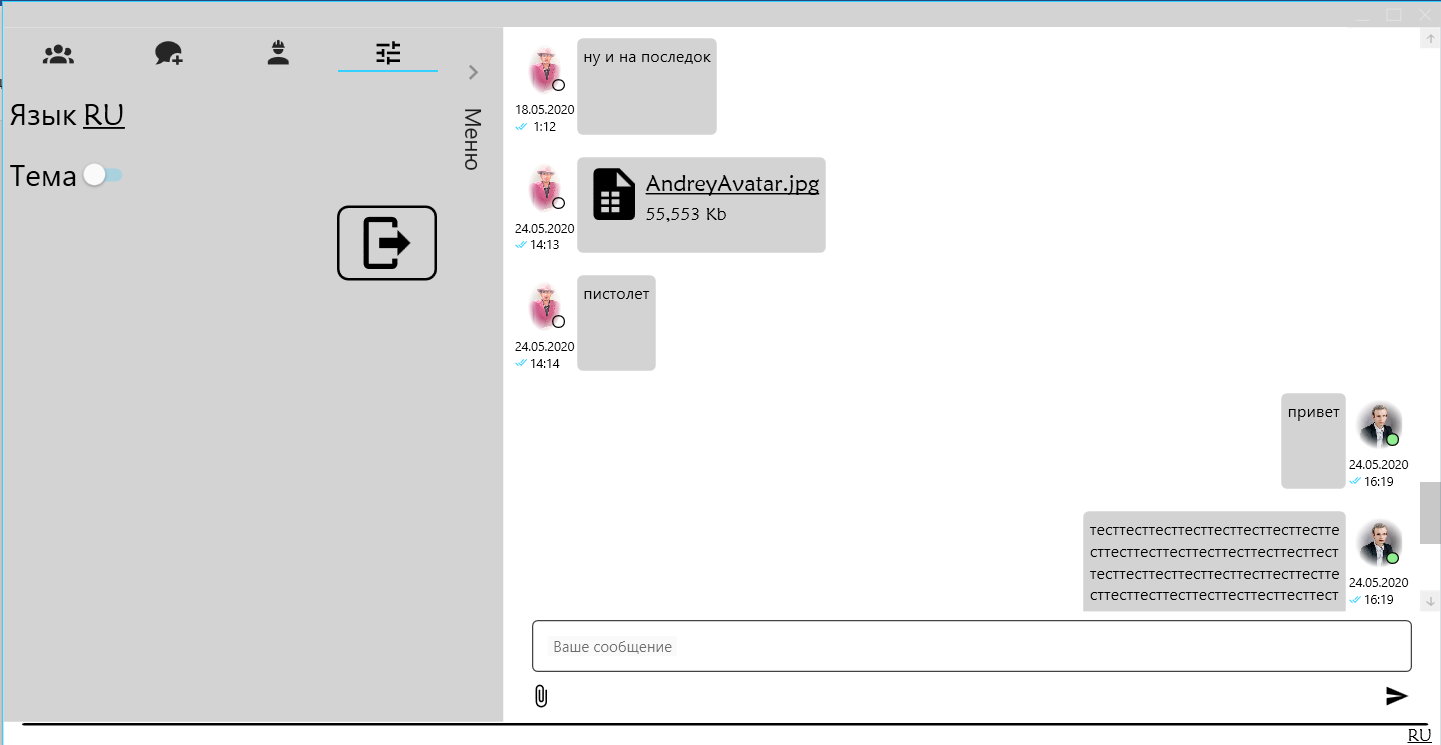


Рисунок 6.8 – Вкладка настроек

В чате отображается не только текст сообщения, но и аватарка отправителя, его статус(онлайн/оффлайн), состояние сообщения(не отправлено/отправлено/прочитано), время. Так же присутствует возможность обмена файлами. При наведении на аватарку собеседника будет указан его ник в приложении, рисунок 6.9. Чтобы сообщение засчиталось как прочитанное на него нужно навестись мышкой. При нажатии на «скрепку» предложится выбрать файл для отправки, после выбора он сразу же отправится в чат. Если нужно скачать файл из чата, то достаточно нажать на имя файла и выбрать куда сохранить.

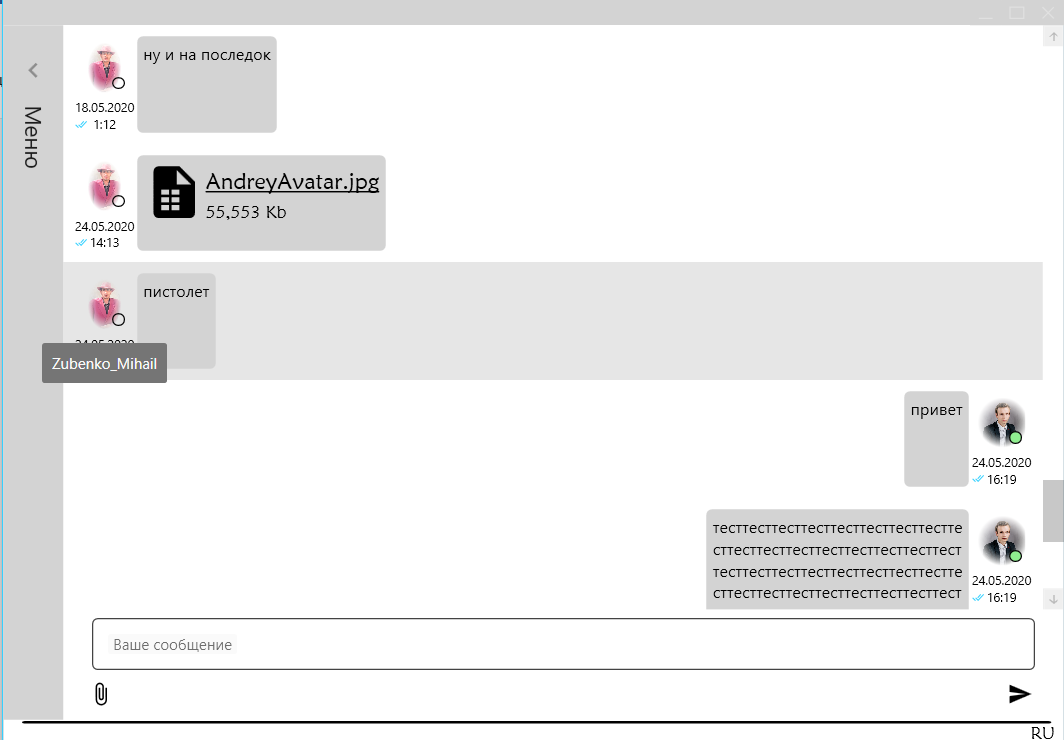


Рисунок 6.9 – Чат и его возможности

## Тестирование

При разработке приложение было сконструировано так, чтобы все данные, с которыми работает пользователь вводились корректно. Это относится к форме регистрации и входа.

Валидация данных в приложении предусмотренна таким образом, что пользователю просто не предоставляется возможность ввести некорректные данные, так же если он забудет заполнить какое-то поле, то ему придет уведомление в строке состояния.

На рисунке 7.1 продемонстрировано поведение страницы авторизации при отсутствии записей в полях. Такое же предупреждение будет возникать и в случае неправильного ввода пароля или логина. Так же в левом нижнем углу присутствует появляющаяся строка состояния, которая уведомляет пользователя, о какой-то ошибке.

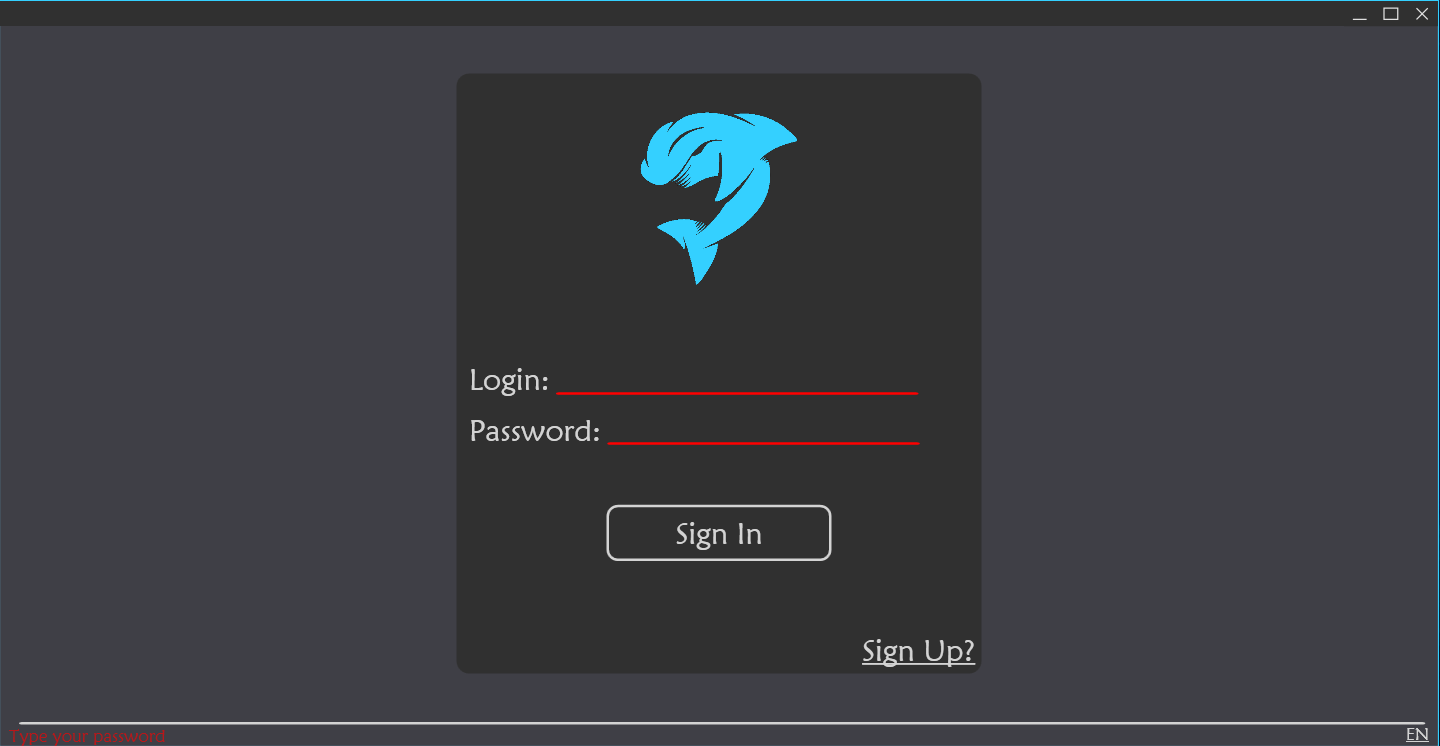


Рисунок 7.1 – Страница авторизации

Такие же действия системы мы можем наблюдать и на странице регистрации, рисунок 7.2.

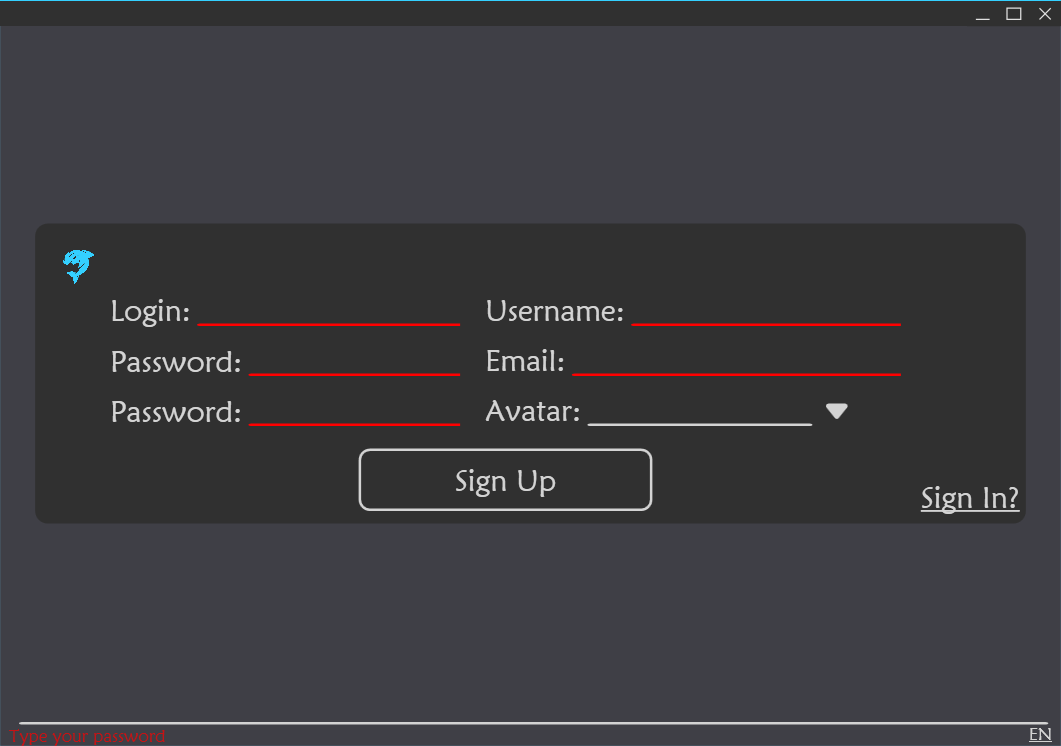


Рисунок 5.2 – Страница регистрации

При отсутствии подлючения к серверу в левом нижнем углу, в строке состояния будет отображаться соответствующее сообщение, рисунок 5.3.

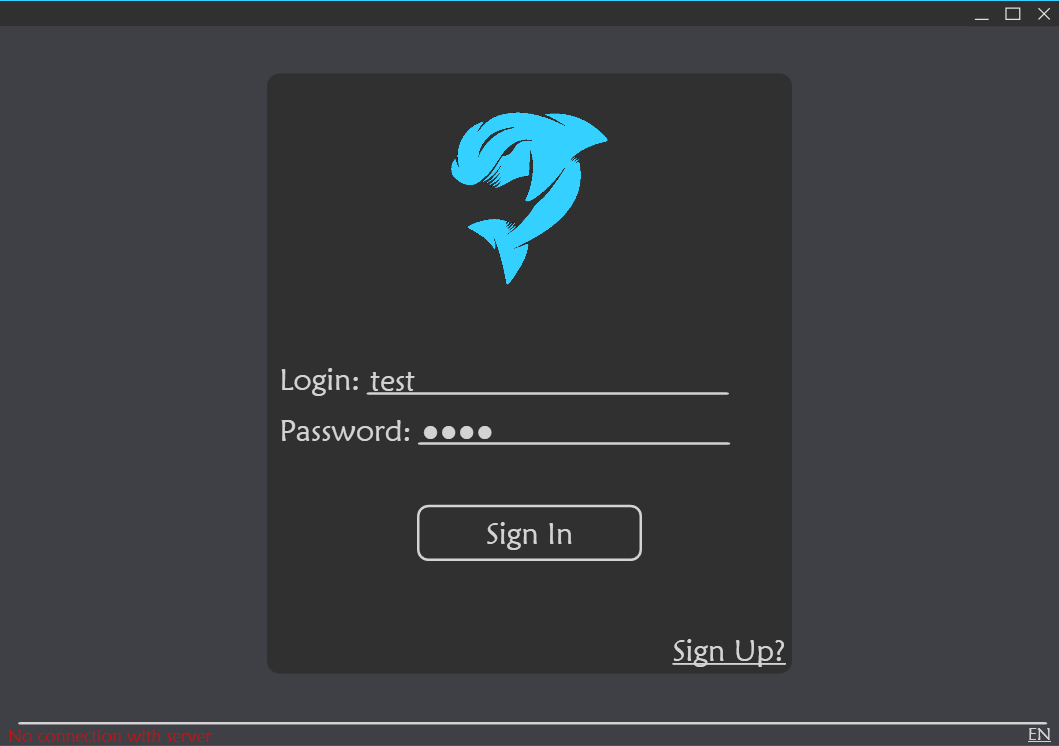


Рисунок 5.3 – Уведомление об отсутствии подключения к серверу

## Заключение

В ходе курсовой работы основные задачи были выполнены: была спроектирована база данных для клиент-серверного мессенджера на базе СУБД Oracle 12c. Приложение было разработано с использованием технологии WPF.NET. Основные требования, которые были поставлены к работе, были выполнены:

* Был реализован ряд процедур и функций, благодаря которому в итоговом серверном приложении поддерживается весь необходимый функционал;
* Возможность авторизации/регистрации пользователей;
* Администратору клиентской части приложения доступен административный поиск среди всех пользователей чатов, а так же блокировка и разблокировка аккаунтов.
* Вся переписка внутри чатов зашифрована по алгоритму AES 192 битным ключом, который является уникальным для каждого из чатов и не храниться в базе данных;
* Реализована технология хранения мультимедийных данных на базе DBMS\_LOB – для хранения аватаров пользователей, заставок бесед и тел сообщений, которые отображаются в профиле пользователя;
* На клиентской части имеется возможность создавать беседы, добавлять пользователей в список контактов, отправлять не только сообщения, но и файлы любого формата не больше 2 гб;
* Реализация нескольких ролей представляет из себя создание двух пользователей на уровне базы данных: администратора и пользователя;

# Список использованных источников

[1] METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com Дата доступа: 10.12.2019

[2] Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru Дата доступа 10.12.2019

[3] stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com Дата доступа: 12.12.2019

[4] Шифрование в PL/SQL: уроки программирования [Электронный ресурс] / Режим доступа: <https://oracle-patches.com/db/sql/3939-shifrovanie-v-pl-sql-uroki-programmirovaniya> Дата доступа: 13.12.2019

[5] DBMS\_LOB [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/database/121/ARPLS/d_lob.htm#ARPLS600> Дата доступа: 13.12.2019

[6] DBMS\_XMLDOM [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B19306_01/appdev.102/b14258/d_xmldom.htm#i1076719>Дата доступа: 03.12.2019

[7] Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/tdddg_procedures.htm> Дата доступа: 10.11.2019

# ПРИЛОЖЕНИЕ A

Табличные пространства, профили безопасности, роли, пользователи

*-----------------------------------------------------------  
-- tablespaces***create tablespace** katran\_tablespace  
 **datafile 'C:\katran\_db\tablespaces\katran\_tablespace.dbf'  
 size** 1000M  
 **autoextend on next** 1m  
 **maxsize unlimited**;  
  
**create temporary tablespace** katran\_tablespace\_temp  
 **tempfile 'C:\katran\_db\tablespaces\katran\_tablespace\_temp.dbf'  
 size** 15 m  
 **autoextend on next** 1m  
 **maxsize unlimited**;  
  
**create undo tablespace** katran\_tablespace\_undo  
 **datafile 'C:\katran\_db\tablespaces\katran\_tablespace\_undo.dbf'  
 size** 5 m  
 **autoextend on next** 1m  
 **maxsize unlimited**;  
*-----------------------------------------------------------  
  
-----------------------------------------------------------  
-- security profiles***create profile** katran\_admin\_security\_profile **limit  
 password\_life\_time** 180 *-- кол-во дней жизни пароля* **sessions\_per\_user unlimited** *-- кол-во сессий для пользователя* **failed\_login\_attempts** 7 *-- кол-во попыток входа* **password\_lock\_time** 1 *-- кол-во дней блокировки после ощибки* **password\_reuse\_time** 1 *-- через сколько дней можно повторить пароль* **password\_grace\_time DEFAULT** *-- кол-во дней предупреждений о смене пароля* **connect\_time unlimited** *-- время соединения* **idle\_time** 60; *-- минут простоя перед отключением***create profile** katran\_user\_security\_profile **limit  
 password\_life\_time** 180 *-- кол-во дней жизни пароля* **sessions\_per\_user unlimited** *-- кол-во сессий для пользователя* **failed\_login\_attempts** 7 *-- кол-во попыток входа* **password\_lock\_time** 1 *-- кол-во дней блокировки после ощибки* **password\_reuse\_time** 1 *-- через сколько дней можно повторить пароль* **password\_grace\_time DEFAULT** *-- кол-во дней предупреждений о смене пароля* **connect\_time unlimited** *-- время соединения* **idle\_time** 60; *-- минут простоя перед отключением  
-----------------------------------------------------------  
  
-----------------------------------------------------------  
-- roles***create role** katran\_admin\_role;  
**grant execute on sys**.dbms\_crypto **to** katran\_admin\_role; *-- sys***grant all privileges to** katran\_admin\_role;  
  
**create role** katran\_user\_role;  
**grant create session**,  
 **read any table**,  
 **select any table**,  
 **insert any table**,  
 **update any table**,  
 **execute any procedure**,  
 **delete any table to** katran\_user\_role;  
  
**grant execute on sys**.dbms\_crypto **to** katran\_user\_role; *-- sys***grant execute on** katran\_package **to** katran\_user\_role;  
**grant debug on** katran\_package **to** katran\_user\_role;  
*-----------------------------------------------------------  
  
-----------------------------------------------------------  
-- users***create user** katran\_admin **identified by** qw2SF24xvGGS  
**default tablespace** katran\_tablespace **quota unlimited on** katran\_tablespace  
**temporary tablespace** katran\_tablespace\_temp  
**profile** katran\_admin\_security\_profile  
**account unlock**;  
  
**grant** katran\_admin\_role **to** katran\_admin;  
  
**create user** katran\_user **identified by** fu6djHH763  
**default tablespace** katran\_tablespace **quota unlimited on** katran\_tablespace  
**temporary tablespace** katran\_tablespace\_temp  
**profile** katran\_user\_security\_profile  
**account unlock**;  
  
**grant** katran\_user\_role **to** katran\_user;  
  
*-----------------------------------------------------------*

# ПРИЛОЖЕНИЕ B

Таблицы

*-----------------------------------------------------------  
-- tables***create table users**(  
 **id int generated as identity constraint** users\_id\_pk **primary key**,  
 **auth\_login varchar2**(20) **unique**,  
 **password varchar2**(32) **not null**,  
 **law\_status varchar2**(5) **default 'User' check**(**law\_status in**(**'Admin'**, **'User'**))  
) **tablespace** katran\_tablespace  
**partition by range** (**id**) **interval** (1000)  
( **partition** part\_1 **values less than** (1000) ) **enable row movement**;  
  
**create table** users\_info  
(  
 **id int constraint** users\_info\_users\_id\_fk **references users**(**id**) **on delete cascade enable**,  
 **app\_name varchar2**(20),  
 **email varchar2**(200),  
 **user\_description varchar2**(200),  
 **image blob**,  
 **status varchar2**(8) **default 'Offline' check**(**status in**(**'Offline'**, **'Online'**)),  
 **is\_blocked number**(1) **default** 0  
) **tablespace** katran\_tablespace  
**partition by range** (**id**) **interval** (1000)  
( **partition** part\_1 **values less than** (1000) ) **enable row movement**;  
  
**create table** contacts  
(  
 **contact\_owner int constraint** contacts\_contact\_owner\_id\_fk **references users**(**id**),  
 **contact int constraint** contacts\_contact\_id\_fk **references users**(**id**)  
) **tablespace** katran\_tablespace  
**partition by range** (**contact\_owner**) **interval** (1000)  
( **partition** part\_1 **values less than** (1000) ) **enable row movement**;  
  
**create table** chats  
(  
 **chat\_id int generated as identity constraint** chats\_chat\_id\_pk **primary key**,  
 **chat\_title varchar2**(40) **unique**, *--для чатов 1 на 1 паттерн названия [id]\_[id], для бесед [название беседы]* **chat\_kind varchar2**(12) **default 'Chat' check**(**chat\_kind in**(**'Chat'**, **'Conversation'**)),  
 **chat\_avatar blob**) **tablespace** katran\_tablespace  
**partition by range** (**chat\_id**) **interval** (1000)  
( **partition** part\_1 **values less than** (1000) ) **enable row movement**;  
  
**create table** chat\_members  
(  
 **chat\_id int constraint** chat\_members\_chat\_id\_fk **references** chats (**chat\_id**),  
 **member\_id int constraint** chat\_members\_member\_id\_fk **references users** (**id**)  
) **tablespace** katran\_tablespace  
**partition by range** (**chat\_id**) **interval** (1000)  
( **partition** part\_1 **values less than** (1000) ) **enable row movement**;

*--template for dynamic table of messages for each Chat creating***create table** ChatMessages\_{chatId}  
(  
 message\_id **int generated as identity primary key**,  
 sender\_id **int references users**(**id**),  
 message **blob**,  
 message\_type **varchar2**(4) **check**(message\_type **in** (**'File'**, **'Text'**)),  
 file\_name **varchar2**(200),  
 **time date**,  
 message\_status **varchar2**(8) **check**(message\_status **in** (**'Readed'**, **'Sended'**, **'Unreaded'**, **'Unsended'**))  
) **tablespace** katran\_tablespace  
**partition by range** (message\_id) **interval** (1000)  
**subpartition by hash**(message\_id) **subpartitions** 10  
( **partition** part\_1 **values less than** (1000) ) **enable row movement**;  
  
  
*-----------------------------------------------------------*

# ПРИЛОЖЕНИЕ C

Пакет процедур и функций

*-----------------------------------------------------------  
-- procedures and functions***create or replace package** katran\_package **AS  
  
 procedure** *GetUserIdByLogin*(inAuthLogin **in users**.**auth\_login**%**type**,  
 outFoundUserId **out users**.**id**%**type**);  
  
 **procedure** *AddUser*(inAuthLogin **in users**.**auth\_login**%**type**,  
 inPassword **in users**.**password**%**type**,  
 inLawStatus **in users**.**law\_status**%**type**,  
 outAddedUserId **out users**.**id**%**type**);  
  
 **procedure** *AddUserInfo*(inId **in** users\_info.**id**%**type**,  
 inAppName **in** users\_info.**app\_name**%**type**,  
 inEmail **in** users\_info.**email**%**type**,  
 inUserDescription **in** users\_info.**user\_description**%**type**,  
 inImage **in** users\_info.**image**%**type**,  
 inStatus **in** users\_info.**status**%**type**,  
 outAddedUserId **out** users\_info.**id**%**type**);  
  
 **procedure** *CheckUserByLoginAndPassword*(inAuthLogin **in users**.**auth\_login**%**type**,  
 inPassword **in users**.**password**%**type**,  
 outFoundUserId **out users**.**id**%**type**);  
  
 **procedure** *GetUserInfoById*(in\_outUserId **in out int**,  
 in\_outAppName **in out varchar2**,  
 in\_outEmail **in out varchar2**,  
 in\_outUserDescription **in out varchar2**,  
 in\_outImage **in out blob**,  
 in\_outStatus **in out varchar2**,  
 in\_outLawStatus **in out varchar2**,  
 in\_outIsBlocked **in out number**);  
  
 **procedure** *ChangeUserStatusById*( inNewUserStatus **in** users\_info.**status**%**type**,  
 in\_outUserId **in out** users\_info.**id**%**type**);  
  
 **procedure** *AddContact*( in\_outContactOwner **in out users**.**id**%**type**,  
 inTargetContact **in users**.**id**%**type**);  
  
 **procedure** *RemoveContact*( in\_outContactOwner **in out users**.**id**%**type**,  
 inTargetContact **in users**.**id**%**type**);  
  
 **procedure** *AddChat*( out\_AddedChatId **out** chats.**chat\_id**%**type**,  
 inChatTitle **in** chats.**chat\_title**%**type**,  
 inChatKind **in** chats.**chat\_kind**%**type default 'Chat'**,  
 inChatAvatar **in** chats.**chat\_avatar**%**type default null**);  
  
 **procedure** *AddChatMember*( inChatId **in** chat\_members.**chat\_id**%**type**,  
 in\_outMemberId **in out** chat\_members.**member\_id**%**type**);  
  
 **procedure** *RemoveChatMember*( inChatId **in** chat\_members.**chat\_id**%**type**,  
 inMemberId **in** chat\_members.**member\_id**%**type**);  
  
 **procedure** *RemoveAllChatMembers*( inChatId **in** chat\_members.**chat\_id**%**type**);  
  
 **procedure** *GetChatMembersCount*( inChatId **in** chat\_members.**chat\_id**%**type**,  
 outMembersCount **out number**);  
  
 **procedure** *GetChatMembersIds*( outChatMembersIds **out sys\_refcursor**,  
 inChatId **in** chat\_members.**chat\_id**%**type**);  
  
 **procedure** *RemoveChat*( in\_outChatId **in out** chat\_members.**chat\_id**%**type**);  
  
 **procedure** *GetConvsInfo*( outResult **out sys\_refcursor**,  
 inMemberId **in** chat\_members.**member\_id**%**type**);  
  
 **procedure** *GetConvMembersInfo*( outResult **out sys\_refcursor**,  
 inChatId **in** chat\_members.**member\_id**%**type**);  
  
 **procedure** *BlockUnblockUser*( in\_outUserId **in out** users\_info.**id**%**type**,  
 inIsBlocked **in** users\_info.**is\_blocked**%**type**);  
  
 **procedure** *GetUserLawStatusById*( in\_outUserId **in out users**.**id**%**type**,  
 in\_outUserLawStatus **in out varchar2**);  
  
 **procedure** *AdminSearch*( outResult **out sys\_refcursor**,  
 inAdminId **in** users\_info.**id**%**type**,  
 inSearchPattern **in varchar2**);  
  
 **procedure** *SearchOutOfContacts*(outResult **out sys\_refcursor**,  
 inPattern **in varchar2**,  
 inSearcherUserID **in** users\_info.**id**%**type**);  
  
 **procedure** *GetUserContacts*(outResult **out sys\_refcursor**,  
 inContactsOwner **in** users\_info.**id**%**type**);  
  
 **function** *EncryptBlob* (enc\_value **in blob**,  
 crypt\_key **in varchar2**)  
 **return blob**; *-- should be 24 symbols length* **function** *DecryptBlob* (enc\_value **in blob**,  
 crypt\_key **in varchar2**)  
 **return blob**; *-- should be 24 symbols length***END**;  
  
**create or replace package body** katran\_package **as***------------------------------------------------- Is user already registered* **procedure** *GetUserIdByLogin*( inAuthLogin **in users**.**auth\_login**%**type**,  
 outFoundUserId **out users**.**id**%**type**)  
 **as  
 begin  
 select id into** outFoundUserId **from users where users**.**auth\_login** = inAuthLogin **and** *ROWNUM* <= 1;  
 **exception  
 when** no\_data\_found **then** outFoundUserId := -1;  
 **end**;  
  
*------------------------------------------------- Is user already registered  
  
------------------------------------------------- AddUser* **procedure** *AddUser*(inAuthLogin **in users**.**auth\_login**%**type**,  
 inPassword **in users**.**password**%**type**,  
 inLawStatus **in users**.**law\_status**%**type**,  
 outAddedUserId **out users**.**id**%**type**)  
 **as  
 begin  
  
 insert into users** (**auth\_login**, **password**, **law\_status**)  
 **values** (inAuthLogin, inPassword, inLawStatus)  
 **returning id into** outAddedUserId;  
 **commit**;  
 **exception  
 when others then** outAddedUserId := -1;  
 **rollback**;  
 **end**;  
  
*------------------------------------------------- AddUserInfo* **procedure** *AddUserInfo*(inId **in** users\_info.**id**%**type**,  
 inAppName **in** users\_info.**app\_name**%**type**,  
 inEmail **in** users\_info.**email**%**type**,  
 inUserDescription **in** users\_info.**user\_description**%**type**,  
 inImage **in** users\_info.**image**%**type**,  
 inStatus **in** users\_info.**status**%**type**,  
 outAddedUserId **out** users\_info.**id**%**type**)  
 **as  
 begin  
 insert into** users\_info (**id**, **app\_name**, **email**, **user\_description**, **image**, **status**)  
 **values**(inId, inAppName, inEmail, inUserDescription, inImage, inStatus)  
 **returning id into** outAddedUserId;  
 **commit**;  
 **exception  
 when others then** outAddedUserId := -1;  
 **rollback**;  
 **end**;  
  
*------------------------------------------------- AddUserInfo  
  
------------------------------------------------- CheckUserByLoginAndPassword* **procedure** *CheckUserByLoginAndPassword*(inAuthLogin **in users**.**auth\_login**%**type**,  
 inPassword **in users**.**password**%**type**,  
 outFoundUserId **out users**.**id**%**type**)  
 **as  
 begin  
 select id into** outFoundUserId **from users where auth\_login** = inAuthLogin **and** inPassword = inPassword;  
 **exception  
 when others then** outFoundUserId := -1;  
 **end**;  
  
*------------------------------------------------- CheckUserByLoginAndPassword  
  
------------------------------------------------- GetUserInfoById* **procedure** *GetUserInfoById*(in\_outUserId **in out int**,  
 in\_outAppName **in out varchar2**,  
 in\_outEmail **in out varchar2**,  
 in\_outUserDescription **in out varchar2**,  
 in\_outImage **in out blob**,  
 in\_outStatus **in out varchar2**,  
 in\_outLawStatus **in out varchar2**,  
 in\_outIsBlocked **in out number**)  
 **as  
 cursor** query\_cursor **is  
 select** ui.**id**, ui.**app\_name**, ui.**email**, ui.**user\_description**, ui.**image**, ui.**status**, u.**law\_status**, ui.**is\_blocked  
 from** users\_info ui **join users** u **on** ui.**id** = u.**id where** ui.**id** = in\_outUserId;  
 **begin  
 open** query\_cursor;  
 **fetch** query\_cursor **into** in\_outUserId, in\_outAppName, in\_outEmail, in\_outUserDescription, in\_outImage, in\_outStatus, in\_outLawStatus, in\_outIsBlocked;  
 **close** query\_cursor;  
 **exception  
 when others then** in\_outUserId := -1;  
 **end**;  
  
*------------------------------------------------- GetUserInfoById  
  
------------------------------------------------- ChangeUserStatusById* **procedure** *ChangeUserStatusById*( inNewUserStatus **in** users\_info.**status**%**type**,  
 in\_outUserId **in out** users\_info.**id**%**type**)  
 **as  
 begin  
 update** users\_info **set status** = inNewUserStatus **where id** = in\_outUserId  
 **returning id into** in\_outUserId;  
 **commit**;  
 **exception  
 when others then** in\_outUserId := -1;  
 **rollback**;  
 **end**;  
  
*------------------------------------------------- ChangeUserStatusById  
  
------------------------------------------------- AddContact* **procedure** *AddContact*( in\_outContactOwner **in out users**.**id**%**type**,  
 inTargetContact **in users**.**id**%**type**)  
 **as  
 begin  
  
 insert into** contacts (**contact\_owner**, **contact**)  
 **values**(in\_outContactOwner, inTargetContact);  
 **commit**;  
 **insert into** contacts (**contact\_owner**, **contact**)  
 **values**(inTargetContact, in\_outContactOwner);  
 **commit**;  
 **exception  
 when others then** in\_outContactOwner := -1;  
 **rollback**;  
 **end**;  
  
*------------------------------------------------- AddContact  
  
------------------------------------------------- RemoveContact* **procedure** *RemoveContact*( in\_outContactOwner **in out users**.**id**%**type**,  
 inTargetContact **in users**.**id**%**type**)  
 **as  
 begin  
 delete from** contacts **where contact\_owner** = in\_outContactOwner **and contact** = inTargetContact;  
 **commit**;  
 **exception  
 when others then** in\_outContactOwner := -1;  
 **rollback**;  
 **end**;  
  
*------------------------------------------------- RemoveContact  
  
-------------------------------------------------*

*AddChat* **procedure** *AddChat*( out\_AddedChatId **out** chats.**chat\_id**%**type**,  
 inChatTitle **in** chats.**chat\_title**%**type**,  
 inChatKind **in** chats.**chat\_kind**%**type default 'Chat'**,  
 inChatAvatar **in** chats.**chat\_avatar**%**type default null**)  
 **as  
 begin  
  
 insert into** chats (**chat\_title**, **chat\_kind**, **chat\_avatar**)  
 **values**(inChatTitle, inChatKind, inChatAvatar)  
 **returning chat\_id into** out\_AddedChatId;  
 **commit**;  
 **exception  
 when others then** out\_AddedChatId := -1;  
 **rollback**;  
 **end**;  
  
*-------------------------------------------------*

*AddChat  
  
------------------------------------------------- AddChatMember* **procedure** *AddChatMember*( inChatId **in** chat\_members.**chat\_id**%**type**,  
 in\_outMemberId **in out** chat\_members.**member\_id**%**type**)  
 **as  
 begin  
  
 insert into** chat\_members (**chat\_id**, **member\_id**)  
 **values**(inChatId, in\_outMemberId)  
 **returning member\_id into** in\_outMemberId;  
 **commit**;  
 **exception  
 when others then** in\_outMemberId := -1;  
 **rollback**;  
 **end**;  
  
*------------------------------------------------- AddChatMember  
  
------------------------------------------------- RemoveChatMember* **procedure** *RemoveChatMember*( inChatId **in** chat\_members.**chat\_id**%**type**,  
 inMemberId **in** chat\_members.**member\_id**%**type**)  
 **as  
 begin  
 delete from** chat\_members **where chat\_id** = inChatId **and member\_id** = inMemberId;  
 **commit**;  
 **end**;  
  
*------------------------------------------------- RemoveChatMember  
  
------------------------------------------------- RemoveAllChatMembers* **procedure** *RemoveAllChatMembers*( inChatId **in** chat\_members.**chat\_id**%**type**)  
 **as  
 begin  
 delete from** chat\_members **where chat\_id** = inChatId;  
 **commit**;  
 **end**;  
  
*------------------------------------------------- RemoveAllChatMembers  
  
------------------------------------------------- GetChatMembersCount* **procedure** *GetChatMembersCount*( inChatId **in** chat\_members.**chat\_id**%**type**,  
 outMembersCount **out number**)  
 **as  
 begin  
 select** *count*(**chat\_id**) **into** outMembersCount **from** chat\_members **where chat\_id** = inChatId;  
 **exception  
 when others then** outMembersCount := 0;  
 **end**;  
  
*------------------------------------------------- GetChatMembersCount  
  
------------------------------------------------- GetChatMembersIds* **procedure** *GetChatMembersIds*( outChatMembersIds **out sys\_refcursor**,  
 inChatId **in** chat\_members.**chat\_id**%**type**)  
 **as  
 begin  
 open** outChatMembersIds **for  
 select member\_id from** chat\_members **where chat\_id** = inChatId;  
 **end**;  
  
*------------------------------------------------- GetChatMembersIds  
  
------------------------------------------------- RemoveChat* **procedure** *RemoveChat*( in\_outChatId **in out** chat\_members.**chat\_id**%**type**)  
 **as  
 begin  
 delete from** chats **where chat\_id** = in\_outChatId;  
 **commit**;  
 **exception  
 when others then** in\_outChatId := -1;  
 **rollback**;  
 **end**;  
  
*------------------------------------------------- RemoveChat  
  
------------------------------------------------- GetConvsInfo* **procedure** *GetConvsInfo*( outResult **out sys\_refcursor**,  
 inMemberId **in** chat\_members.**member\_id**%**type**)  
 **as  
 begin  
 open** outResult **for  
 select** chat.**chat\_id**, chat.**chat\_title**, chat.**chat\_avatar  
 from** Chats chat  
 **where** chat.**chat\_id in** (**select** cm.**chat\_id from** chat\_members cm **where** cm.**member\_id** = inMemberId) **and** chat.**chat\_kind** = **'Conversation'**;  
 **end**;  
  
*------------------------------------------------- GetConvsInfo  
  
------------------------------------------------- GetConvMembersInfo* **procedure** *GetConvMembersInfo*( outResult **out sys\_refcursor**,  
 inChatId **in** chat\_members.**member\_id**%**type**)  
 **as  
 begin  
 open** outResult **for  
 select** ui.**id**, ui.**app\_name**, ui.**image**, ui.**status  
 from** Users\_info ui **join** chat\_members cm  
 **on** ui.**id** = cm.**member\_id and** cm.**chat\_id** = inChatId;  
 **end**;  
  
*------------------------------------------------- GetConvMembersInfo  
  
------------------------------------------------- GetUserLawStatusById* **procedure** *GetUserLawStatusById*( in\_outUserId **in out users**.**id**%**type**,  
 in\_outUserLawStatus **in out varchar2**)  
 **as  
 begin  
 select law\_status into** in\_outUserLawStatus **from users where id** = in\_outUserId;  
 **exception  
 when others then** in\_outUserLawStatus := **''**;  
 **end**;  
  
*------------------------------------------------- GetUserLawStatusById  
  
------------------------------------------------- BlockUnblockUser* **procedure** *BlockUnblockUser*( in\_outUserId **in out** users\_info.**id**%**type**,  
 inIsBlocked **in** users\_info.**is\_blocked**%**type**)  
 **as  
 begin  
 update** users\_info **set is\_blocked** = inIsBlocked **where id** = in\_outUserId;  
 **commit**;  
 **exception  
 when others then** in\_outUserId := -1;  
 **rollback**;  
 **end**;  
  
*------------------------------------------------- BlockUnblockUser  
  
------------------------------------------------- AdminSearch* **procedure** *AdminSearch*( outResult **out sys\_refcursor**,  
 inAdminId **in** users\_info.**id**%**type**,  
 inSearchPattern **in varchar2**)  
 **as  
 begin  
 open** outResult **for  
 select** ui.**id**, ui.**app\_name**, ui.**image**, ui.**status**, ui.**is\_blocked  
 from** Users\_info ui  
 **where** *instr*(ui.**app\_name**, inSearchPattern) > 0 **and** ui.**id** != inAdminId;  
 **end**;  
  
*------------------------------------------------- AdminSearch  
  
------------------------------------------------- SearchOutOfContacts* **procedure** *SearchOutOfContacts*(outResult **out sys\_refcursor**,  
 inPattern **in varchar2**,  
 inSearcherUserID **in** users\_info.**id**%**type**)  
 **as  
 begin  
 open** outResult **for  
 select** ui.**id**, ui.**app\_name**, ui.**image**, ui.**status**, ui.**is\_blocked  
 from** Users\_info ui  
 **where** *instr*(ui.**app\_name**, inPattern) > 0 **and** ui.**id** != inSearcherUserID **and** ui.**id NOT IN**(**select** ui.**id from** Users\_info ui **join** Contacts **c on** ui.**id** = **c**.**contact and c**.**contact\_owner** = inSearcherUserID);  
 **end**;  
  
*------------------------------------------------- SearchOutOfContacts  
  
------------------------------------------------- GetUserContacts* **procedure** *GetUserContacts*(outResult **out sys\_refcursor**,  
 inContactsOwner **in** users\_info.**id**%**type**)  
 **as  
 begin  
 open** outResult **for  
 select** ui.**id**, ui.**app\_name**, ui.**image**, ui.**status**, ui.**is\_blocked  
 from** Users\_info ui  
 **join** Contacts **c  
 on** ui.**id** = **c**.**contact and c**.**contact\_owner** = inContactsOwner;  
 **end**;  
  
*------------------------------------------------- GetUserContacts  
  
-------------------------------------------------*

*Crypto* **function** *EncryptBlob* (enc\_value **in blob**,  
 crypt\_key **in varchar2**) *-- should be 24 symbols length* **return blob  
 is** encrypted\_blob **blob**;  
 l\_mod **number** := dbms\_crypto.encrypt\_aes192 + dbms\_crypto.chain\_cbc + dbms\_crypto.pad\_pkcs5;  
 **begin** encrypted\_blob := dbms\_crypto.*encrypt*(enc\_value, l\_mod, utl\_i18n.*string\_to\_raw*(crypt\_key, **'AL32UTF8'**));  
 **return** encrypted\_blob;  
 **end** *EncryptBlob*;  
  
 **function** *DecryptBlob* (enc\_value **in blob**,  
 crypt\_key **in varchar2**) *-- should be 24 symbols length* **return blob  
 is** decrypted\_blob **blob**;  
 l\_mod **number** := dbms\_crypto.encrypt\_aes192 + dbms\_crypto.chain\_cbc + dbms\_crypto.pad\_pkcs5;  
 **begin** decrypted\_blob := dbms\_crypto.*decrypt*(enc\_value, l\_mod, utl\_i18n.*string\_to\_raw*(crypt\_key, **'AL32UTF8'**));  
 **return** decrypted\_blob;  
 **end** *DecryptBlob*;  
  
*-------------------------------------------------*

*Crypto***end** katran\_package;  
  
  
*-----------------------------------------------------------  
-- import & export;***create or replace directory** KATRAN\_XML **as 'C:\katran\_db\xml'**;  
  
**CREATE OR REPLACE PROCEDURE** *chatmessages\_export***IS**rc **sys\_refcursor**;  
doc DBMS\_XMLDOM.DOMDocument;  
**BEGIN  
OPEN** rc **FOR SELECT** *\** **FROM** katran\_admin.CHATMESSAGES\_00000000046;  
doc := DBMS\_XMLDOM.NewDOMDocument(**XMLTYPE**(rc));  
DBMS\_XMLDOM.WRITETOFILE(doc, **'KATRAN\_XML\CHATMESSAGES\_00000000046.xml'**);  
**END** *chatmessages\_export*;  
  
**begin** *chatmessages\_export*();  
**end**;  
  
**CREATE OR REPLACE PROCEDURE** *chatmessages\_import***IS  
BEGIN  
 DELETE FROM** CHATMESSAGES\_00000000045;  
 **INSERT INTO** CHATMESSAGES\_00000000045 (**SENDER\_ID**, **MESSAGE**, **MESSAGE\_TYPE**, **TIME**, **MESSAGE\_STATUS**)  
 **SELECT** *ExtractValue*(*VALUE*(product\_order\_xml), **'//SENDER\_ID'**) **AS** SENDER\_ID,  
 *ExtractValue*(*VALUE*(product\_order\_xml), **'//MESSAGE'**) **AS** MESSAGE,  
 *ExtractValue*(*VALUE*(product\_order\_xml), **'//MESSAGE\_TYPE'**) **AS** MESSAGE\_TYPE,  
 *ExtractValue*(*VALUE*(product\_order\_xml), **'//TIME'**) **AS TIME**,  
 *ExtractValue*(*VALUE*(product\_order\_xml), **'//MESSAGE\_STATUS'**) **AS** MESSAGE\_STATUS  
 **FROM TABLE**(*XMLSequence*(*EXTRACT*(**XMLTYPE**(*bfilename*(**'KATRAN\_XML'**, **'CHATMESSAGES\_import.xml'**),  
  
 *nls\_charset\_id*(**'UTF-8'**)),**'/ROWSET/ROW'**))) product\_order\_xml;  
**END** *chatmessages\_import*;