Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет радиоэлектроники

Кафедра информационных радиотехнологий

Дисциплина: Конструирование программ и языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ПРОГРАММНОЕ СРЕДСТВО ОБМЕНА СООБЩЕНИЯМИ

Студентка: гр.444601 Вашкевич А. С.

Руководитель: Кучук С. А.

Минск, 2016

# СОДЕРЖАНИЕ

Введение3  
1 Обзор литературы5

1.1 Выбор языка разработки5

1.2 Выбор СУБД6

1.3 Entity Framework7

1.4 Выбор между WinForms и WPF8

2 Структурное проектирование10  
3 Функциональное проектирование12

3.1 Модуль подключений по сети12

3.2 Модуль взаимодействия с клиентом12

3.3 Модуль подключения к серверу12

3.4 Модуль UI13  
4 Тестирование14

5 Руководство пользователя17  
Заключение21

Список использованных источников22

# ВВЕДЕНИЕ

В данном курсовом проекте будет разработано программное средство обмена сообщениями со следующим функционалом: авторизация, получение и отправка сообщения в режиме реального времени, добавление и удаление контактов из своего списка, отправление добавленному контакту различные файлы, а также шифрование своих данных. В этом проекте будет применена база данных MySQL, и вся переписка пользователя с контактом будет хранится именно там.

Интернет общение в современном мире приобрело широкое распространение по причине того, что, возможно, человеку не хватает общения в реальных контактах, или, например, для реализации качеств личности, которые невозможно воплотить в реальной жизни или они скрываются, в том числе для людей, которые хотят общения с теми, кто находится, к примеру, в другой стране. Интернет позволяет связаться с человеком в любой точке мира в любое время – это одно из преимуществ мессенджеров. Второе – это приватность и персональность, ведь в отличии от переписки в социальных сетях, мессенджеры обеспечивают достаточным уровнем приватности. Третье положительное качество мессенджеров – это разнообразный контент: можно пересылать не только текст, но и фото, видео, геопозицию. Также через основные мессенджеры можно звонить, причём бесплатно.

В нынешнее время средства обмена сообщениями используются абсолютно везде. Чаще всего используют мессенджеры в бизнесе для консультации клиентов, приема товаров или, к примеру, для записи на прием. В маркетинге или в интернет-магазинах. Мессенджеры могут использовать турфирмы, авиаперевозчики, гостиницы как удобное средство для консультаций или бронирования. Так же телеком-компании и операторы связи могут организовать информирование абонентов о балансе через мессенджеры и сэкономить на СМС-рассылке. Или такие средства можно использовать просто для развлечения.

Конечно же, существует огромное количество аналогов данного средства: Skype, GoogleTalk, Mail.Agent, VZOchat, Viber и многие другие. Например, в Skype для пользователя представлена возможность не только обмена сообщениями, но и возможности звонков, видеозвонков как между абонентами, так и групповые, переадресация звонков, а также звонки на мобильные и стационарные телефоны. Помимо этого, существует отправка сообщений, которые можно видеть и слышать, смайлов и эмодзи, контактных данных, голосовой почты, файлов. В Skype перед пользователями открывается возможность демонстрации экрана для человека, с которым вы разговариваете, либо для группы людей.

Еще одним популярным мессенджером в наши дни является Viber. Функционал публичного чата позволяет пользователям Viber распространять информационные материалы, участвовать в разговорах и создавать публичные чаты с участием других пользователей, а также обеспечивает возможность знакомиться с содержанием того или иного публичного чата и просматривать все представленные в нём информационные материалы. Также Viber позволяет совершать звонки через Интернет бесплатно или осуществлять голосовой вызов на городские и мобильные телефоны, у которых отсутствует выход в Интернет.

В курсовом проекте планируется реализовать чат со следующими функциями: авторизация, управление списком контактов, обмен сообщениями с добавленным контактом, передача файлов, шифрование данных.

# 1 ОБЗОР ЛИТЕРАТУРЫ

**1.1 Выбор языка разработки**

Когда начинается проектирование часто возникает вопрос о выборе средств разработки и если в одних случаях ответ на этот вопрос следует из потребности совместимости с библиотеками и приложениями, то в других ответ не очевиден.

Для того, чтобы выбрать более подходящее средство разработки проекта, проанализируем и сравним такие языки программирования как C# и C++. Сравнение будет происходит по критериям, которые наиболее значимы для решаемой задачи.

Рассмотрим критерий скорости разработки.

С# позволяет стартовать разработку быстрее, а это позволяет быстрее получить прототип решения. Скорость разработки на С# на начальных этапах проекта значительно выше по сравнению с С++.

Однако, когда инфраструктура проекта создана, основные подходы и библиотеки выбраны, скорость разработки на С++ и скорость разработки на С# становятся примерно одинаковыми.

Таким образом, в коротких проектах С# будет иметь преимущество по скорости разработки, но в длинных данное преимущество будет незначительным.

Рассмотрим критерий производительности кода и требовательности к ресурсам.

Если говорить о совокупности субъективных «простоты разработки», «красоты кода» и объективной производительности, то используя C# проще написать код, удовлетворяющий этим критериям одновременно. Однако это не значит, что производительный код на С++ обязательно будет сложным для восприятия, просто при его написании потребуется более «творческий» подход для удовлетворения перечисленных критериев одновременно.

Рассмотрим предоставляемые библиотеки.

Отличие ассортимента С++ и С# библиотек в том, что С++ библиотек больше, они имеют большую историю, за которую стали неплохо отлажены и оптимизированы, часто под разные операционные системы, многие с открытым кодом. Однако при всех положительных сторонах С++ библиотеки имеют очень разную, часто даже архаичную архитектуру, часто не объектный, а структурно-процедурный интерфейс. Связано это с тем, что многие С++ библиотеки это С библиотеки.

Другая неприятная особенность С++ библиотек — это создание и переопределение своих базовых типов. Многие С++ библиотеки заводят свои типы строк, контейнеров, переопределяют некоторые базовые типы. Базовые же С++ библиотеки дают не так много, как дают стандартные библиотеки С#, поэтому подбор правильных библиотек для проекта С++ — это задача, необходимая даже в сравнительно простых проектах.

В С# перечисленных выше проблем значительно меньше. Огромное количество библиотек включено в состав .Net Framework, также имеется большое количество множество свободно доступных библиотек, что покрывает практически все первостепенные задачи разработки под Windows.

Рассмотрим критерий синтаксиса.

С первого взгляда код С++ и С# очень похож внешне. Но многообразие кода на С++ больше, ведь С++ является одновременно и С, и С++, и С++0х и все это можно использовать одновременно, если это поддерживает компилятор.

С# – это только C#, хотя его синтаксис постоянно расширяется. Код на С#, как правило, выглядит проще и лаконичнее, чем код С++. Языковые конструкции С++ и С# очень схожи, однако существенные различия можно найти в деталях.

Чаще всего сложный код легче пишется и анализируется, если написан более простым языком. С этой позиции, используя С#, меньше шансов допустить ошибку в принципиально сложном коде и больше шансов написать чистый код, обладая теми же ресурсами. Это может быть полезно при решении достаточно сложных, но не требовательных к производительности задач. Однако при этом большее количество «синтетики» в С# делает меньше оценку производительности кода по его «внешнему виду».

Рассмотрим критерий удобства сборки.

Сборка С++ проектов заметно сложнее сборки проектов С#. Однако стоит понимать, что большая сложность предоставляет и дополнительную гибкость, которая рано или поздно может стать полезной.

Проанализировав все критерии было принято использовать для разработки программного средства обмена сообщениями язык программирования C#, так как он больше подходит для создания небольших проектов.

**1.2 Выбор СУБД**

В этом проекте будет применена база данных для хранения переписки пользователя с контактом. Кратко рассмотрим наиболее популярные бесплатные СУБД.

MySQL используется во многих крупных программных продуктах и на многих веб-сайтах, в том числе YouTube, Wikipedia, Facebook, и многих других. Система полностью бесплатна (в том числе и для коммерческого использования). MySQL создан с помощью языков C/C++, однако большое количество языков программирования включают в себя API для работы с MySQL (MySQLConnector / Net для C# и VB, JDBC driver для Java, MyODBC для языков, поддерживающих интерфейс ODBC).

MS SQL – серверная база данных от Microsoft. Для написания скриптов используется Transact-SQL. В полной версии поддерживается куб (OLAP) и присутствуют функции для сбора статистики и добычи данных (data mining). Среди редакций есть бесплатная, экспресс версия — MS SQL Express.

SQLite – бесплатная СУБД для работы с реляционными БД, написанная на C. Очень легковесна и проста в использовании. Хорошо документирована и доступна для скачивания либо в формате исходного кода, либо уже скомпилированной. Отлично подходит для проектов, не требующих сложных схем доступа к данным и не предъявляющих серьезных требований к безопасности.

PostgreSQL– мощная и тяжелая система, отвечающая всем современным стандартам СУБД. Больше подходит для серьезных проектов, требующих сложных баз данных. По скорости работы PostgreSQL уступает MySQL. Также данная СУБД сложнее в администрировании, но является не чисто реляционной, а реляционно-объектной, что позволяет ей легко отражать иерархии объектов на реляционные таблицы.

Для решения поставленной задачи лучше всего подходит СУБД MS SQL, так как она имеет ряд преимуществ:

- многопоточность, поддержка нескольких одновременных запросов;

- интерфейс с языками C и Perl, PHP;

- быстрая работа, масштабируемость;

- быстрая поддержка транзакций через механизм InnoDB.

**1.3 Entity Framework**

Раньше SQL запросы писали прямо в исходном коде либо хранили в базе данных хранимые процедуры. Такой подход приводил к ухудшению читабельности и безопасности кода, что привело к созданию такого подхода как ORM: технологии программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

Задачей ORM библиотек являлась необходимость обеспечить работу с данными в терминах классов, а не таблиц данных и напротив, преобразовать термины и данные классов в данные, пригодные для хранения в СУБД. В общем, предполагалось избавиться от необходимости писать SQL-код для взаимодействия в СУБД.

Примерами ORM библиотек для C# являются BLToolkit, NHibernate.

NHibernate позволяет отображать объекты бизнес-логики на реляционную базу данных. По заданному XML-описанию сущностей и связей NHibernate автоматически создает SQL-запросы для загрузки и сохранения объектов.

Поддерживаемые СУБД: Microsoft SQL Server, Oracle, Microsoft Access, Firebird, PostgreSQL, DB2 UDB, MySQL, SQLite.

В качестве альтернативы ORM библиотекам Microsoft реализовал LINQ: проект по добавлению синтеза языка запросов, напоминающего SQL, в языке программирования платформы .NET Framework.

Платформа Entity Framework представляет собой набор технологий ADO.NET, которые обеспечивают разработку приложений, связанных с обработкой данных. Архитекторам и разработчикам приложений, ориентированных на обработку данных, приходится учитывать необходимость достижения двух совершенно различных целей. Они должны моделировать сущности, связи и логику решаемых бизнес-задач, а также работать с ядрами СУБД, используемыми для сохранения и получения данных. Данные могут распределяться по нескольким системам хранения данных, в каждой из которых применяются свои протоколы, но даже в приложениях, работающих с одной системой хранения данных, необходимо поддерживать баланс между требованиями системы хранения данных и требованиями написания эффективного и удобного для обслуживания кода приложения.

**1.4 Выбор между WinForms и WPF**

WinForms и WPF представляют собой две различные архитектуры для создания интерфейсов приложений.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (Extensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

WPF предоставляет широкий спектр возможностей по созданию интерактивных настольных приложений, одним из них является привязка данных – это гибкий механизм, который позволяет через расширения разметки XAML связывать различные данные. Привязка данных представлена классом Binding, который в свою очередь унаследован от MarkupExtension, что позволяет использовать привязки не только в коде, но и в разметке.

Технология Windows Forms представляет собой набор управляемых библиотек, упрощающих выполнение стандартных задач, таких как чтение из файловой системы и запись в нее.

В Windows Forms форма — это видимая поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы.

В состав Windows Forms входят многофункциональные элементы пользовательского интерфейса, позволяющие воссоздавать возможности таких сложных приложений, как Microsoft Office. Используя элементы управления ToolStrip и MenuStrip, можно создавать панели инструментов и меню, содержащие текст и рисунки, подменю и другие элементы управления, такие как текстовые поля и поля со списками.

С помощью поддерживающего перетаскивание конструктора Windows Forms в Visual Studio можно легко создавать приложения Windows Forms. Достаточно выделить элемент управления курсором и поместить его в нужное место на форме. Для преодоления трудностей, связанных с выравниванием элементов управления, конструктор предоставляет такие средства, как линии сетки и линии привязки.

Для разработки программного средства обмена сообщениями была выбрана технология WinForms, благодаря легкому и быстрому созданию приложений.

Для данного программного средства выбраны следующие технологии: C#, MySQL, Entity Framework, WinForms.

# 2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

Структуру программного средства можно представить в виде диаграммы компонентов (рис. 2.1, рис. 2.2).

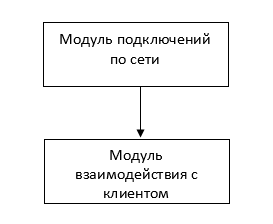


Рисунок 2.1 – Диаграмма компонентов программного средства для серверной части

Описание компонентов приложения для серверной части:

Модуль подключений по сети – ожидает соединения с пользователем, после чего перенаправляет обработку входящих сообщений к обработчикам.

Модуль взаимодействия с клиентом – классы, отвечающие за обработку действий со стороны клиента. К примеру, авторизация, отправка или прием сообщений, поиск контактов и т.д.

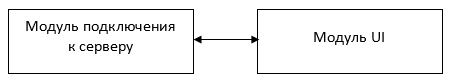


Рисунок 2.2 – Диаграмма компонентов программного средства для клиентской части

Описание компонентов приложения для клиентской части:

Модуль подключения к серверу – модуль, который отвечает за соединение с сервером.

Модуль UI – модуль, который отвечает за отображение пользовательского интерфейса.

База данных разрабатываемого приложения будет иметь структуру, показанную на рисунке 2.3.

Таблица «Friends» будет использоваться для хранения списка друзей пользователей.

Таблица «Users» будет использоваться для хранения данных о всех зарегистрировавшихся пользователей.

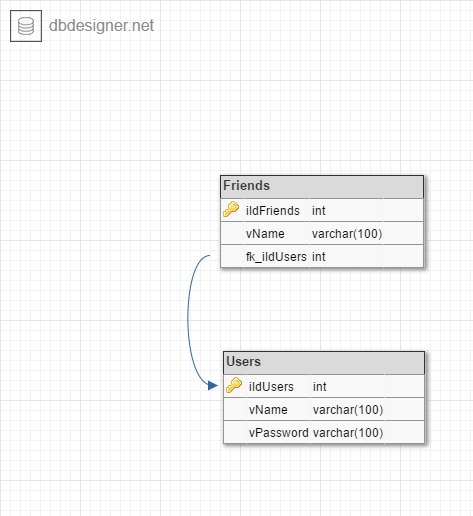


Рисунок 2.3 – Схема базы данных

В ходе структурного проектирования были выделены следующие компоненты в серверной части: модуль подключений по сети и модуль взаимодействия с клиентом. В том числе клиентской части: модуль подключения к серверу и модуль UI.

# 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Проектируемое программное средство будет включать в себя следующие модули:

- модуль подключений по сети;

- модуль взаимодействия с клиентом;

- модуль подключения к серверу;

- модуль UI;

**3.1 Модуль подключений по сети**

Модуль подключений по сети ожидает соединения с пользователем, после чего перенаправляет обработку входящих сообщений к обработчикам.

**3.2 Модуль взаимодействия с клиентом**

Модуль взаимодействия с клиентом представляет собой классы, отвечающие за обработку действий со стороны клиента, которые представлены на рисунке 3.1.

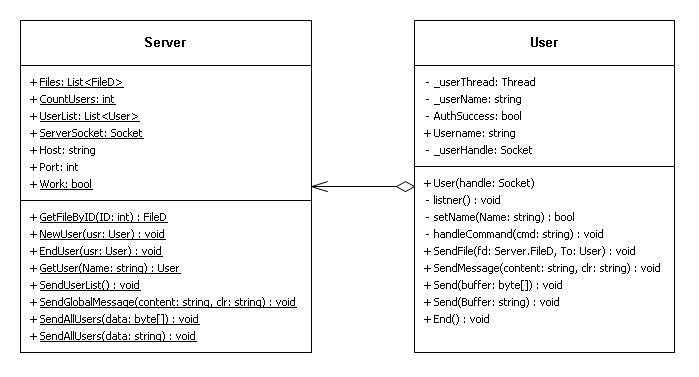


Рисунок 3.1 – Диаграмма классов модуля взаимодействия с клиентом

**3.3 Модуль подключения к серверу**

Модуль подключения к серверу предназначен для соединения с сервером. Модуль состоит из набора классов, представленных на рисунке 3.3.

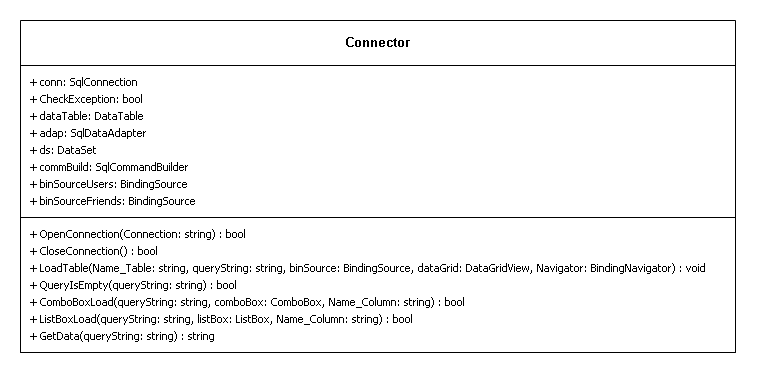


Рисунок 3.3 – Диаграмма классов модуля подключения к серверу

**3.4 Модуль UI**

Модуль UI отвечает за отображение пользовательского интерфейса.

# 4 ТЕСТИРОВАНИЕ

Тестирование – процесс исследования, испытания программного продукта с целью продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям, и выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации. При выявлении несоответствий работы программы техническому заданию, либо ошибок, требуется доработка программного обеспечения и (или) документации.

Для проведения корректного тестирования сначала необходимо разработать порядок испытаний. Порядок испытаний - это список последовательности действий направленных на проверку корректности работы программы и (или) ее отдельных функциональных частей.

Существует множество признаков, по которым принято производить классификацию видов тестирования. Наиболее часто пользуются следующими критериями при классификации методов тестирования:

* По объекту тестирования (функциональное, производительности, конфигурационное, юзабилити, интерфейс пользователя, безопасность, локализация, совместимость).
* По знанию системы (тестирование чёрного, белого или серого ящиков).
* По степени автоматизации (ручное, автоматизированное или полуавтоматизированное).
* По степени изолированности компонентов (модульное, интеграционное, системное).
* По времени проведения (альфа, бета).
* По степени подготовленности (тестирование по документации или интуитивное).

Автоматизированное тестирование программного обеспечения — часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

Одной из главных проблем автоматизированного тестирования является его трудоемкость: несмотря на то, что оно позволяет устранить часть рутинных операций и ускорить выполнение тестов, большие ресурсы могут тратиться на обновление самих тестов. При изменении интерфейса приложения необходимо заново переписать все тесты, которые связаны с обновленными окнами, что при большом количестве тестов может отнять значительные ресурсы.

Автоматические тесты не могут полностью заменить ручное тестирование. Автоматизация всех испытаний — очень дорогой процесс, и потому автоматическое тестирование является лишь дополнением ручного тестирования.

Ручное тестирование – это процесс поиска дефектов в работе программы, когда тестировщик проверяет работоспособность всех компонентов программы с точки зрения пользователя.

Начинать тестирование программы лучше с самого начала. Когда будет написан шаблон, необходимо провести быстрое тестирование. После внесения в работу программы каких-либо серьезных изменений необходимо снова протестировать ее.

Ручное тестирование – это ключевой этап разработки программного обеспечения. Тестер может не придерживаться строго плана тестирования, а отклоняться от него для более полного тестирования, приближенного к использованию программы обычным пользователем.

Безусловно, ручное тестирование уступает автоматизированному хотя бы в экономической составляющей. Ведь для ручных тестов зачастую требуется значительный человеческий ресурс, в несколько раз больше времени и усилий. Однако именно человеческий фактор помогает избежать многих ошибок в тестировании.

Тестирование всего приложения проводилось методом ручного тестирования путём многократных запусков, разнообразных оправляемых событиях и моделируемых ошибочных ситуаций.

Таблица 4.1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| Компонент | Тест | Сценарий теста | Результат |
| Авторизация в клиенте | Неавторизованный пользователь не может войти в приложение. | 1. Запустить приложение.  2. Ввести некорректные авторизационные данные.  3. Убедиться, что форма не закрылась и отобразилось сообщение «Неверный логин или пароль». | Успешно |
| Авторизованный пользователь может войти в приложение. | 1. Запустить приложение.  2. Ввести корректные авторизационные данные.  3. Убедиться, что открылась форма списка контактов. | Успешно |
|  |  |  |  |
| Список контактов | Добавление пользователя в список друзей | 1. Запустить приложение.  2. Ввести корректные авторизационные данные.  3. Убедиться, что открылась форма списка контактов. | Успешно |
| Продолжение таблицы 4.1 | | | |
|  |  | 4. Добавить пользователя в список друзей.  5. Убедиться, что данный пользователь отобразился в списке друзей. |  |
| Удаление пользователя из списка друзей | 1. Запустить приложение.  2. Ввести корректные авторизационные данные.  3. Убедиться, что открылась форма списка контактов.  4. Удалить пользователя из списка друзей.  5. Убедиться, что данный пользователь удалился из списка друзей. | Успешно |
| Окно чата | Отправка сообщения | 1. Запустить приложение.  2. Ввести корректные авторизационные данные.  3. Убедиться, что открылась форма списка контактов.  4. Выбрать пользователя.  5. Отправить сообщение.  6. Убедиться, что адресату дошло сообщение. | Успешно |
| Передача файлов | 1. Запустить приложение.  2. Ввести корректные авторизационные данные.  3. Убедиться, что открылась форма списка контактов.  4. Выбрать пользователя.  5. Отправить файл.  6. Убедиться, что адресату дошел файл. | Успешно |

В результате тестирования была получено более защищенное приложения, адекватно реагирующее на возможные ошибочные ситуации.

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для работы программного средства обмена сообщениями компьютер должен удовлетворять следующим системным требованиям:

Операционная система: Windows 7, 8, 10,

- RAM: 256 Мб,

- HDD: 14 Мб,

- Устройства ввода: клавиатура, компьютерная мышь.

Попадая на главную форму программного средства, пользователь видит набор полей для ввода данных (рис. 5.1).

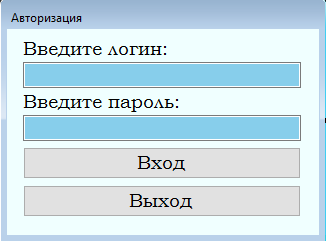


Рисунок 5.1 – Скриншот главной формы приложения

Поля для ввода «Введите логин» и «Введите пароль» позволяют пользователю ввести логин и пароль соответственно.

Кнопка «Вход» после авторизации переходит в главное окно чата (рис. 5.2), кнопка «Выход» закрывает приложение.

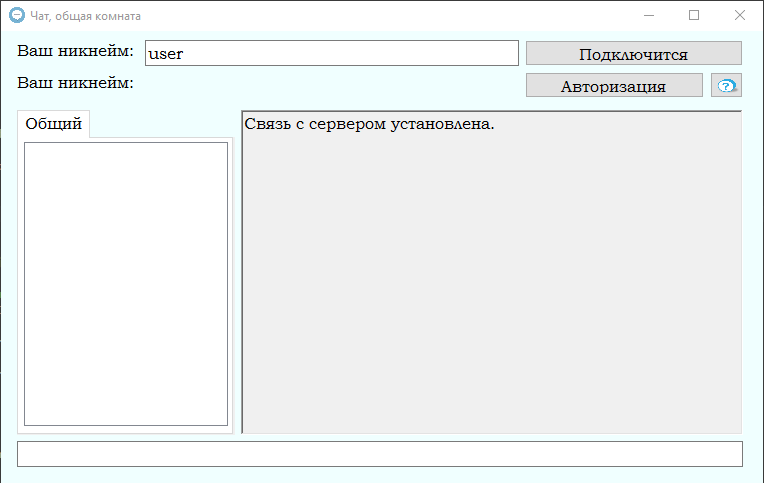


Рисунок 5.2 – Скриншот главного окна чата

Далее заполнить поле «Ваш никнейм» и нажать кнопку «Подключиться». После подключения попадаем в окно общего чата (рис. 5.3).

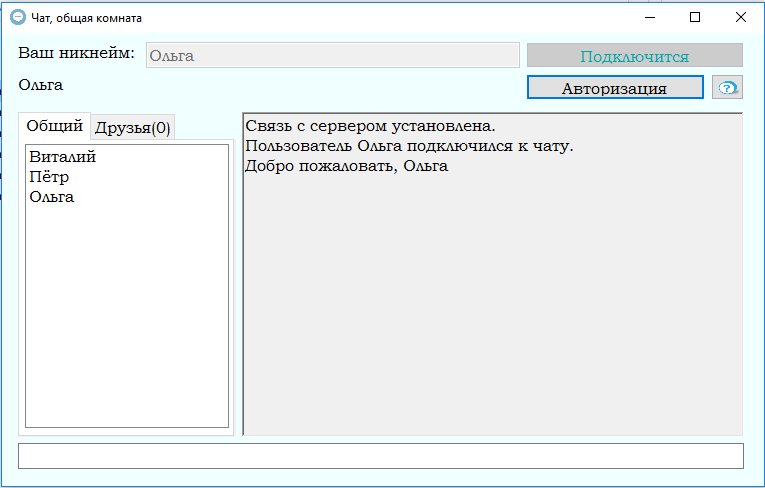


Рисунок 5.3 – Скриншот окна общего чата

Для того, чтобы отправить сообщение в общий чат, необходимо написать в поле ввода сообщение и нажать кнопу Enter (рис. 5.4).

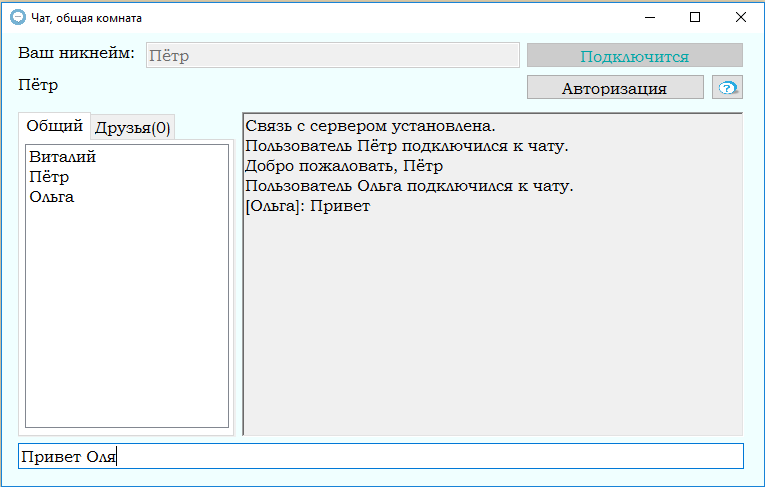


Рисунок 5.4 – Скриншот передачи сообщений в общем чате

При необходимости написать пользователю личное сообщение надо нажать правую кнопку мыши и выбрать в контекстном меню пункт «Личное сообщение» (рис. 5.5).

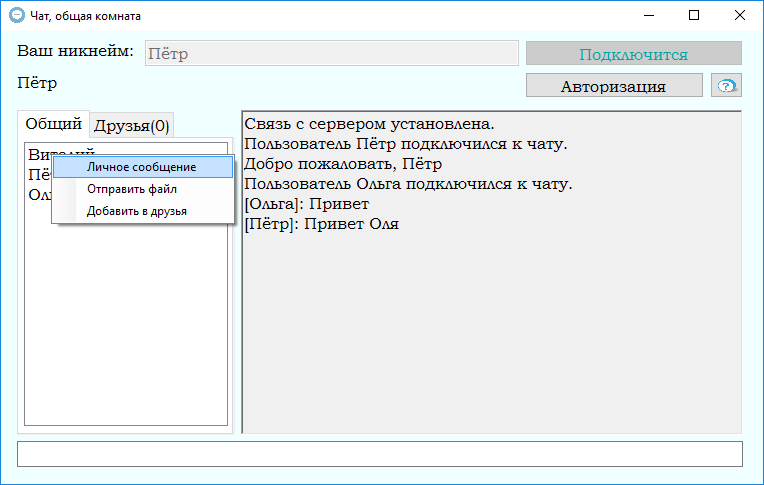


Рисунок 5.5 – Скриншот отправки личного сообщения пользователю

Чтобы добавить пользователя из общего чата в список друзей следует нажать правую кнопку мыши и выбрать в контекстном меню пункт «Добавить в друзья» (рис. 5.6), после этого пользователь отобразится в списке друзей (рис. 5.7).

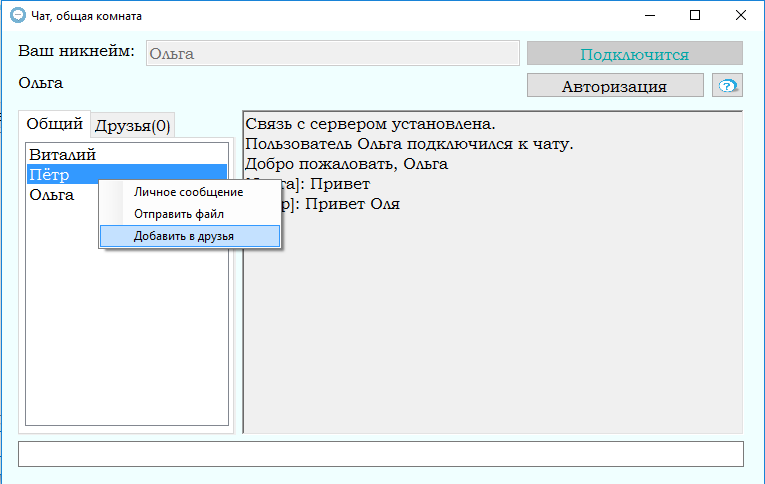


Рисунок 5.6 – Скриншот добавления пользователя в список друзей

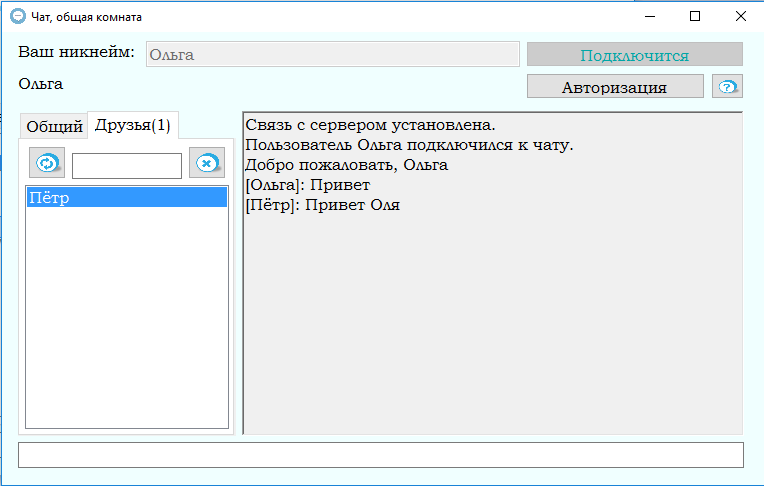


Рисунок 5.7 – Скриншот списка друзей программного средства

Для удаления пользователя из списка друзей необходимо ввести в поле ввода его никнейм и нажать кнопку «Удалить», которая находится справа от поля ввода (рис. 5.8).

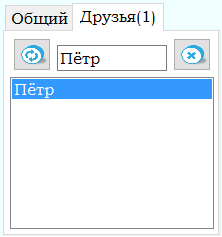


Рисунок 5.8 – Скриншот удаления пользователя из списка друзей

Чтобы передать пользователю файл требуется нажать на правую кнопку мыши и выбрать в контекстном меню пункт «Отправить файл».

# ЗАКЛЮЧЕНИЕ

В курсовом проекте было разработано программное средство обмена сообщениями со следующими функциями: авторизация, управление списком контактов, обмен сообщениями с добавленным контактом, передача файлов, шифрование данных.

Для данного программного средства были выбраны следующие технологии: C#, MySQL, Entity Framework, WinForms.

В процессе разработки было проведено проектирование, включающее в себя создание архитектуры приложения, определение набора классов необходимых для написания, определены методы и подходы для написания каждого конкретного класса.

Для работы программного средства обмена сообщениями компьютер должен удовлетворять следующим системным требованиям:

Операционная система: Windows 7, 8, 10,

- RAM: 256 Мб,

- HDD: 14 Мб,

- Устройства ввода: клавиатура, компьютерная мышь.

Тестирование всего приложения проводилось методом ручного тестирования путём многократных запусков, разнообразных оправляемых событиях и моделируемых ошибочных ситуаций.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Буч, Г. Язык UML Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон – ДМК Пресс, Питер, 2004. – 432 с.

[2] Грабер, М. Введение в SQL/ М. Грабер– М.: Издательский дом «Лори», 2010. – 228 с.

[3] Джеффри, Р. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / Р. Джеффри – 3-е издание –Питер Год: 2012 –437 с.

[4] Петкович, Д. Microsoft SQL Server 2008. Руководство для начинающих / Д. Петкович – БХВ.: Петербург, 2009. – 752 с.