Белорусский государственный университет

Институт непрерывного образования

Кафедра прикладной математики и информатики

КУРСОВОЙ ПРОЕКТ

на тему

Программное средство «Сетевой чат»

Слушатель группы 15ПО11-04з

по специальности 1-40 01 73

«Программное обеспечение

информационных систем»

Селюк Иван Чеславович

Руководитель Кравцова А.А.

Допущен к защите:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Зав. кафедрой Лесун Б.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, ФИО, дата)

Минск 2013

Белорусский государственный университет

Институт непрерывного образования

Кафедра прикладной математики и информатики

|  |  |
| --- | --- |
| Группа 15ПО11-04з | УТВЕРЖДАЮ  Зав. кафедрой  Б.В. Лесун  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г. |

ЗАДАНИЕ

на курсовое проектирование

Слушателю Селюку Ивану Чеславовичу

1 Тема проекта Программное средство «Сетевой чат»

2 Срок сдачи законченного проекта «08» июля 2013 г.

3 Исходные данные по проекту:

Программное средство разрабатывается в архитектуре клиент-сервер на объектно-ориентированном языке программирования. Необходимо разработать собственную иерархию классов с использованием сокрытия данных (инкапсуляция), наследования, виртуальных методов (полиморфизм), перегрузки и переопределения методов. Так же можно использовать интерфейс программирования приложений (API) предоставляемый операционной системой. Предусмотреть обработку исключительных ситуаций.

Необходимо:

Разработанное программное средство позволяет обмениваться текстовыми сообщениями в локальной сети, поддерживает функции регистрации нового пользователя, входа и выхода из системы зарегистрированных пользователей, имеет возможность настройки сетевых подключений.

4 Состав проекта:

а) пояснительная записка:

Введение

Основная часть

- Анализ и разработка требований

- Проектирование программного средства

- Разработка программного средства

- Тестирование программного средства

- Руководство пользователя

Заключение

б) графическая часть проекта:

- Диаграмма вариантов использования

- Диаграмма взаимодействия при отправке сообщения

- Диаграмма классов модуля сообщений

- Диаграмма состояний UDP сервера

- Выбранные средства разработки

- Модель взаимодействия потоков при получении сообщений

- Пользовательский интерфейс

5 Календарный график работы на весь период проектирования:

17.06.2013 - 20.06.2013 – анализ и разработка требований;

20.06.2013 – 23.062013 – программное проектирование;

23.06.2013 – 30.06.2013 – реализация программного средства;

30.06.2013 – 02.07.2013 – тестирование программного средства;

02.07.2013 – 07.07.2013 – оформление пояснительной записки;

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Кравцова

Дата выдачи задания «17» июня 2013 г.

Задание принял к исполнению: «17» июня 2013 г.

Подпись слушателя: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

АННОТАЦИЯ

к курсовому проекту Селюка Ивана Чеславовича

слушателя группы 15ПО11-04з по специальности 1-40 01 73   
«Программное обеспечение информационных систем»

на тему

«Программное средство «Сетевой чат»

Ключевые слова: ПРОГРАММНОЕ СРЕДСТВО, СООБЩЕНИЕ, СОКЕТ, ЛОКАЛЬНАЯ СЕТЬ, СЕТЕВОЙ ПРОТОКОЛ, СТЕК СЕТЕВЫХ ПРОТОКОЛОВ.

Предметной областью курсового проектирования является общение пользователей в локальной сети, с помощью передачи текстовых сообщений.

Целью курсового проекта является создание программного средства, которое позволит обмениваться текстовыми сообщениями в локальной сети и, тем самым, организовать систему общения между работниками конкретного предприятия.

При выполнении курсового проектирования были использованы среда визуального программирования Visual Studio 2010, среда проектирования информационных систем Rational Rose 2000.

Пояснительная записка к курсовому проекту включает перечень сокращений, перечень определений, введение, пять глав и заключение, содержит 38 страниц, в том числе 1 таблица, 8 рисунков, 7 листингов, 2 приложения, 6 литературных источников.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Определения…………………………………………………………………..... | 6 |
| Введение………………………………………………………………………... | 8 |
| 1 Анализ и разработка требований к программному средству……………... | 10 |
| 1.1 Анализ предметной области…………………………………………. | 10 |
| 1.2 Разработка требований к программному средству…………………. | 11 |
| 1.2.1 Общее описание……………………………………………….. | 11 |
| 1.2.2 Функциональность системы………………………………….. | 11 |
| 1.2.3 Требования к внешним интерфейсам………………………... | 12 |
| 2 Проектирование программного средства…………………………………... | 14 |
| 2.1 Общие сведения………………………………………………………. | 14 |
| 2.2 Разработка диаграммы вариантов использования………………….. | 14 |
| 2.3 Разработка диаграмм взаимодействия………………………………. | 15 |
| 2.4 Разработка диаграмм классов………………………………………... | 17 |
| 2.5 Разработка диаграмм состояний……………………………………... | 18 |
| 3 Разработка программного средства…………………………………………. | 20 |
| 3.1 Обоснование выбора средств и инструментов разработки………... | 20 |
| 3.2 Разработка модулей программного средства……………………….. | 22 |
| 3.2.1 Разработка модуля пользовательского интерфейса………… | 22 |
| 3.2.2 Разработка модуля компонентов……………………………... | 23 |
| 3.2.3 Разработка модуля отправки и получения сообщений……... | 25 |
| 4 Тестирование программного средства……………………………………… | 28 |
| 4.1 Выбор методики тестирования………………………………………. | 28 |
| 4.2 Тестирование программного средства………………………………. | 28 |
| 5 Руководство пользователя…………………………………………………... | 31 |
| Заключение……………………………………………………………………... | 33 |
| Список использованных источников…………………………………………. | 34 |
| Приложение А………………………………………………………………….. | 35 |
| Приложение Б…………………………………………………………………... | 37 |

**ОПРЕДЕЛЕНИЯ**

Локальная вычислительная сеть (англ. Local Area Network, LAN) – компьютерная сеть, покрывающая обычно относительно небольшую территорию или небольшую группу зданий (дом, офис, фирму, институт).

Ethernet – пакетная технология передачи данных преимущественно локальных компьютерных сетей. Стандарты Ethernet определяют проводные соединения и электрические сигналы на физическом уровне, формат кадров и протоколы управления доступом к среде — на канальном уровне модели OSI.

Протокол передачи данных – набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами. Эти соглашения задают единообразный способ передачи сообщений и обработки ошибок при взаимодействии программного обеспечения, разнесённой в пространстве аппаратуры, соединённой тем или иным интерфейсом.

Стек протоколов – иерархически организованный набор сетевых протоколов, достаточный для организации взаимодействия узлов в сети.

Межпроцессное взаимодействие (англ. Inter-Process Communication, IPC) – набор способов обмена данными между множеством потоков в одном или более процессах. Процессы могут быть запущены на одном или более компьютерах, связанных между собой сетью. IPC-способы делятся на методы обмена сообщениями, синхронизации, разделяемой памяти и удаленных вызовов (RPC). Методы IPC зависят от пропускной способности и задержки взаимодействия между потоками и типа передаваемых данных.

Сокет (англ. socket – разъём) – название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет – абстрактный объект, представляющий конечную точку соединения.

UDP (англ. User Datagram Protocol – протокол пользовательских датаграмм) – один из ключевых элементов Internet Protocol Suite, набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.

Дейтаграмма (англ. datagram), также датаграмма – блок информации, посланный как пакет сетевого уровня через передающую среду без предварительного установления соединения и создания виртуального канала. Датаграмма представляет собой единицу информации в протоколе (protocol data unit, PDU) для обмена информацией на сетевом и транспортном (в случае протокола UDP, UDP-датаграммы) уровнях эталонной модели OSI.

Многопоточность – свойство платформы (например, операционной системы, виртуальной машины и т. д.) или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени. При выполнении некоторых задач такое разделение может достичь более эффективного использования ресурсов вычислительной машины.

Чат – средство обмена сообщениями по компьютерной сети в режиме реального времени, а также программное обеспечение, позволяющее организовывать такое общение.

Система обмена мгновенными сообщениями (англ. Instant messaging, IM) – служба мгновенных сообщений (Instant Messaging Service, IMS), программы онлайн-консультанты (OnlineSaler) и программы-клиенты (Instant Messenger, IM) для обмена сообщениями в реальном времени через Интернет. Могут передаваться текстовые сообщения, звуковые сигналы, изображения, видео, а также производиться такие действия, как совместное рисование или игры. Многие из таких программ-клиентов могут применяться для организации групповых текстовых чатов или видеоконференций.

**ВВЕДЕНИЕ**

С развитием информационных технологий стали возможными ещё более глобальные коммуникации. Историческим «докомпьютерным» предшественником чатов, несомненно, был телефон. Ни почта, ни телеграф не позволяли общаться в режиме реального времени, и не были доступны в домашней обстановке. Изобретение и распространение телефона по планете вызвало настоящую революцию в средствах и способах общения. Возможность поговорить с собеседником на другой стороне Земли казалась настоящим чудом.

Во второй половине XX века начали бурно развиваться компьютеры. Однако долгое время они были большими и слишком дорогими, что препятствовало тому, чтобы расходовать драгоценное машинное время на забавы с обменом сообщениями вместо расчётов атомных бомб. К тому же до конца 60-х годов они не были связаны друг с другом. Предок Интернета, сеть ARPANET, в 1969 году насчитывала только четыре связанных друг с другом научных компьютера. Чуть позже, в 1971 году, была придумана электронная почта, которая стала необычайно популярна ввиду своего удобства. Постепенно появились новые службы сообщений, такие, как списки почтовой рассылки, новостные группы и доски объявлений. Однако в то время сеть ARPANET ещё не могла легко взаимодействовать с другими сетями, построенными на других технических стандартах, что затрудняло её распространение.

Программы для обмена текстовыми строками, несмотря на простоту самой идеи, появились не сразу. Примерно в 1974 году для мэйнфрейма PLATO был разработана программа Talkomatic, потенциально позволявшая общаться между тысячей терминалов системы. В 1980-x появилась система Freelancing' Round table. Однако по-настоящему популярным стал разработанный в 1988 году протокол, названный Internet Relay Chat (IRC), что примерно можно перевести как ретранслируемый интернет-разговор. Где-то в это же время появилось и распространилось само понятие «чат».

Обычно под словом «чат» подразумевается обмен текстовыми сообщениями, реже встречаются голосовой чат, видеочат (или видеоконференция). Существует огромное множество средств общения с использованием сетевых технологий. Например: интернет форумы, Web-чаты, программы для обмена мгновенными сообщениями через Интернет в реальном времени такие, как ICQ, Jabber, AIM, MSN, Yahoo, Google Talk и соответствующие сетевые протоколы. Эти протоколы используют протоколы более низкого уровня, например TCP/IP, как транспорт для передачи данных. Передаваться могут текстовые сообщения, звуковые сигналы, картинки, видео. Такие программы могут применяться для организации телеконференций.

Для этого вида коммуникации необходима клиентская программа, так называемый мессенджер (англ. message – сообщение). Он отличается от электронной почты тем, что позволяет обмениваться сообщениями в реальном времени (instant – мгновенно). Большинство программ позволяет видеть, подключены ли в данный момент абоненты, занесенные в список контактов. В ранних версиях программ все, что печатал пользователь, тут же передавалось. Если он делал ошибку и исправлял ее, это тоже было видно. В таком режиме общение напоминало телефонный разговор. В современных программах сообщения появляются на мониторе собеседника только по окончанию редактирования и отправке.

Как правило, мессенджеры не работают самостоятельно, а подключаются к главному компьютеру данной сети обмена сообщениями, называемому сервером. Поэтому мессенджеры называют ещё клиентскими программами или клиентами (термин взят из понятий о клиент-серверных технологиях).

Широкому кругу пользователей известно некоторое количество популярных сетей обмена сообщениями, таких как ICQ, MSN, Yahoo!. Каждая из этих сетей разработана отдельной группой разработчиков, имеет отдельный сервер, отличается своими правилами и особенностями. Между различными сетями обычно нет никакой взаимосвязи. Таким образом, пользователь сети ICQ не может связаться с пользователем сети MSN. Однако, ничто не мешает быть одновременно пользователем нескольких сетей.

Для каждой из сетей есть свой мессенджер, разработанный той же командой разработчиков. Так, для пользования вышеуказанными сетями разработчиками предлагаются программы с одноименными названиями: ICQ, MSN Messenger, Yahoo! Messenger.

Большинство IM-сетей используют закрытые протоколы, поэтому альтернативные клиенты обычно обладают меньшим количеством функций, чем официальные. Также, в связи с изменениями протоколов на стороне сервера, альтернативные клиенты могут внезапно переставать работать.

Целью данной курсовой работы является разработка программного средства «Сетевой чат» для локальной сети не требующего собственного сервера.

**1 АНАЛИЗ И РАЗРАБОТКА ТРЕБОВАНИЙ   
К ПРОГРАММНОМУ СРЕДСТВУ**

**1.1 Анализ предметной области**

Создание компьютерных сетей вызвано практической потребностью пользователей удаленных друг от друга компьютеров в одной и той же информации. Сети предоставляют пользователям возможность быстрого обмена информацией, совместной работы на сетевых устройствах, и даже одновременной обработки документов.

Развитие компьютерных сетей привело к появлению различных средств общения, так называемых чатов. Обычно под словом «чат» подразумевается обмен текстовыми сообщениями, реже встречаются голосовой чат, видеочат.  
Разрабатывались не только определенные программные средства, но так же и специализированные протоколы передачи сообщений. Одним из самых первых и популярных стал разработанный в 1988 году протокол, названный Internet Relay Chat (IRC), что примерно можно перевести как ретранслируемый интернет-разговор. Впоследствии появились многие другие.

Существующие программные средства для передачи текстовых сообщений можно условно разделить на 2 группы:

- приложения, сочетающие в себе функциональные возможности, как сервера, так и клиента для приема передачи сообщений. Данные программные средства обычно используются в локальных сетях.

- приложения, выполняющие только роль клиента при передаче сообщений. Сервер чаще всего является удаленным и выполняет функции транзита переданной информации от одного клиента к другому. Данные программы-клиенты называются мессенджерами.

Разрабатываемое программное средство «Сетевой чат» относится к первой группе сетевых приложений, т.е. сочетать в себе функции, как сервера для приема сообщений, так и клиента для их отправки. Таким образом, не будет необходимости устанавливать в сети дополнительное серверное приложение.

Чаты в домашних локальных сетях чаще всего используются для общения с друзьями и знакомыми. В корпоративных и офисных сетях чаты предназначены для:

- рассылки объявлений или заданий коллегам и подчиненным;

- напоминаний работникам о проведении совещаний;

- получения оповещений от администратора сети;

Архитектура сетевых чатов, как и архитектура других сетевых приложений, разделяют прикладную систему на службы, которые могут совместно и многократно использоваться множеством приложений. Во время выполнения сетевого приложения поток управления реализуется на одном или на нескольких хостах. Все компоненты такой системы обмениваются информацией, передавая друг другу данные и управление потоком по мере необходимости. Если использовать совместимые протоколы обмена информацией, то можно добиться взаимодействия отдельных компонентов, даже если базовые сети, операционные системы, аппаратные средства и языки программирования являются неоднородными.

**1.2 Разработка требований к программному средству**

**1.2.1 Общее описание**

Программное средство «Сетевой чат» представляет собой оконное приложение, предназначенное для отправки и получения текстовых сообщений между пользователями (компьютерами) в локальной сети.

Уровень владения персональным компьютером пользователей данного программного средства должен соответствовать уровню «Опытный пользователь».

Программное средство функционирует в среде под управлением операционной системы Windows.

Компьютеры, использующие сетевой чат должны быть подключены к локальной сети Ethernet. При подключении чат использует UDP и TCP/IP стеки сетевых протоколов.

**1.2.2 Функциональность системы**

Программное средство поддерживает функцию регистрации нового пользователя. При регистрации необходимо ввести имя пользователя и пароль. В дальнейшем введенные данные изменить будет невозможно.

Функцию входа в систему (логина) пользователей осуществляется каждый раз при запуске программы. При этом пользователю необходимо ввести имя и пароль. При наличии ошибок в вводимых данных пользователь будет оповещен об этом. Программа имеет возможность смены текущего пользователя во время работы. Только вошедший в системы пользователь имеет возможность принятия и отправки сообщений.

Отправка сообщений осуществляется путем ввода текста и нажатием на кнопку «Отправить».

Отправленные сообщения помещаются в окно отправленных и полученных сообщений.

Все вошедшие в систему клиенты в сети отображаются в списке пользователей. Список пользователей отображает только онлайн пользователей в данный момент времени. Для каждого пользователя в списке предусмотрено свое окно отправленных и полученных сообщений.

При выходе из системы или отключении пользователя он автоматически удаляется из списка пользователей у других клиентов.

Предусмотрена возможность настройки программного средства для работы в определенной сети (при наличии у пользователя нескольких сетевых интерфейсов) и/или с использованием различных портов для приема и отправки сообщений.

При приеме и передаче сообщений программное средство использует сетевой протокол UDP.

**1.2.3 Требования к внешним интерфейсам**

Главное окно приложения имеет возможность сворачивания на панель задач, и разворачивания на всю ширину рабочего стола. Пользовать может изменить размер главного окна в зависимости от своих предпочтений.

Главное окно разделяется на три области:

- список онлайн пользователей, представляет собой вертикальный список пользователей, отображает их имена, указанные при регистрации.

- окно для ввода новых сообщений, имеет функцию редактирования.

- окно для отображения принятых и отправленных сообщений, функция редактирования отключена.

Размер каждой области можно изменить относительно других.

В правом нижнем углу главного окна расположена кнопка отправки сообщений.

Главное меню расположено вверху окна под заголовком. Меню имеет следующую структуру:

- Вход в систему (Sign In);

Вход в систему (Sign In);

Выход из системы (Sign Out);

Закрытие приложения (Exit);

- Настройки (Settings);

- Информация о программе (About);

Программное средство предназначено для работы в операционных системах Windows XP, Windows Vista, Windows 7.

Требования к оборудованию (определены экспериментально, зависят от выбранной операционной системы):

- CPU AMD Duron 1000 Mhz;

- оперативная память 256 Mb;

- 50 Mb свободного места на жестком диске;

- сетевой адаптер Ethernet;

- подключение к локальной сети.

**2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

**2.1 Общие сведения**

Использование объектно-ориентированное проектирования на основе графических диаграмм языка UML позволяет:

- сокращение цикла разработки приложения «заказчик – программист – заказчик»;

- увеличение продуктивности работы программистов вследствие уменьшения ошибок и времени на отладку;

- улучшение потребительских качеств программ за счет ориентации на пользователей и бизнес;

- возможность вести большие проекты и группы проектов;

- возможность повторного использования уже созданного ПО за счет упора на разбор их архитектуры и компонентов;

- использование языка UML как универсального «мостика» между разработчиками из разных подразделений.

В качестве программного пакета для визуального объектно-ориентированного моделирования систем на основе классов и их взаимодействия выбран визуальный редактор Rational Rose.

**2.2 Разработка диаграммы вариантов использования**

Одним из основных преимуществ применения диаграммы вариантов использования – это предоставляет важной информации. Она показывает, какие функциональные возможности будут заложены в систему, кто конкретно будет с ней взаимодействовать. Изучение всего множества вариантов использования и действующих лиц, позволяет определить сферу применения системы.

При проектировании диаграммы вариантов использования для разрабатываемого программного средства были определены следующие варианты использования:

- отправка текстовых сообщений;

- получение текстовых сообщений;

- настройка сетевого подключения;

- выбор сетевого адаптера;

- регистрация;

- вход и выход из системы;

Диаграмма вариантов использования для программного средства «Сетевой чат» представлена на рисунке 2.1.



**Рисунок 2.1 –** **Диаграмма вариантов использования   
программного средства «Сетевой чат»**

**2.3 Разработка диаграмм взаимодействия**

На диаграмме взаимодействия отображается один из процессов обработки информации в варианте использования. Различают два типа диаграмм взаимодействия – диаграммы последовательности и кооперативные диаграммы. Диаграммы первого типа организованы по времени, они заостряют внимание на управлении. Кооперативные диаграммы отображают поток данных.

С помощью диаграмм взаимодействия мы можем определить классы, которые нужно создать, связи между ними, а также операции и ответственности каждого класса. Диаграммы последовательности полезны для того, что бы понять логическую последовательность событий в сценарии. Кооперативные диаграммы полезны в тех случаях, когда нужно оценить последствия сделанных изменений.

Диаграммы последовательностей используются для моделирование конкретных экземпляров классов, интерфейсов, компонентов и узлов, а также сообщений, которыми они обмениваются, – и все это в контексте сценария, иллюстрирующего данное поведение. Диаграммы взаимодействий могут существовать автономно и служить для визуализации, специфицирования, конструирования и документирования динамики конкретного сообщества объектов, а могут использоваться для моделирования отдельного потока управления в составе прецедента.

Диаграмма последовательности при отправке текстового сообщения пользователю в сети представлена на рисунке 2.2.



**Рисунок 2.2 –** **Диаграмма последовательности   
при отправке сообщения пользователю в сети**

На основании построенной в Rational Rose диаграммы последовательности можно автоматически получить кооперативную диаграмму. Кооперативная диаграмма при отправке сообщения пользователю в сети представлена на рисунке 2.3.



**Рисунок 2.3 –** **Кооперативная диаграмма   
при отправке сообщения пользователю в сети**

**2.4 Разработка диаграмм классов**

На диаграммах классов отображаются некоторые классы и пакеты системы. Это статические картины фрагментов системы и связей между ними.

Обычно для описания системы создают несколько диаграмм Классов. На одних показывают некоторое подмножество классов и отношения между классами подмножества. На других отображают то же подмножество, но вместе с атрибутами и операциями классов. Третьи соответствуют только пакетам классов и отношениям между ними. Для представления полной картины системы можно разработать столько диаграмм классов, сколько требуется.

В ходе проектирования информационной системы было принято решение использовать объектно-ориентированный подход. При проектировании иерархии классов необходимо исходить из правила, что каждый класс, каждая сущность должно выполнять строго определенную задачу, функцию. Для построения гибкой и расширяемой системы были использованы паттерны объектно-ориентированного проектирования. Диаграмма классов модуля приема и отправки текстовых сообщений представлена на рисунке 2.4.



**Рисунок 2.4 –** **Диаграмма классов  
модуля отправки и приема текстовых сообщений**

Как видно из приведенной диаграммы в модуле отправки и получения сообщений находится UDP сервер, который считывает данные из сокета и передает их обработчику сообщений (Messages Handler). После обработки сообщения оно помещается в очередь сообщений (Messages Queue). После извлечения сообщения из очереди оно непосредственно отображается в окне пользователя. Объект (Server Settings Holder) хранит в себе настройки UDP сервера. Таким образом, применение конкретного хранителя настроек позволяет повторно использовать один и тот же класс UDP сервера.

**2.5 Разработка диаграмм состояний**

На диаграмме состояний отображается жизненный цикл одного объекта, начиная с момента его создания и заканчивая разрушением. С помощью таких диаграмм удобно моделировать динамику поведения класса. Как правило, диаграммы состояний не требуется создавать для каждого класса. Многие проекты вообще обходятся без них. Если динамика поведения класса важна, то для него полезно разработать диаграмму состояний. Такие классы обычно имеют несколько различных состояний.

В среде Rational Rose на основании диаграмм состояний не генерируется никакого исходного кода. Они нужны для того, чтобы документировать динамику поведения класса.



**Рисунок 2.5 –** **Диаграмма состояний  
объекта класса UDP Server**

На рисунке 2.5 представлена диаграмма состояний UDP сервера. Как видно из данной диаграммы объект сервера после создания и инициализации переходит в состояние ожидания, до момента входа пользователя в систему (логина). После логина сервер переходит в состояние приема сообщений. Если пользователь вновь выйдет из системы, сервер перейдет в состояние ожидания. После завершения работы приложения сервер завершает работу, освобождает используемые ресурсы и разрушается.

**3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**3.1 Обоснование выбора средств и инструментов разработки**

Для создания программного средства «Сетевой чат» была выбрана среда разработки MS Visual Studio 2010.

Интегрированная среда разработки (Integrated Development Environment, IDE) MS Visual Studio является последней по времени выпуска версией популярной и широко используемой среды разработки профессионального программного обеспечения (ПО) производства компании Microsoft.

Объединяя в своем составе все положительные стороны предыдущих версий, данная обеспечивает возможность использования всех преимуществ современной технологий разработки программного обеспечения. В числе основных достоинств MS Visual Studio, оцененных сообществом профессиональных программистов, можно отметить следующие моменты:

- Повышение производительности труда разработчиков. Среда разработки MS Visual Studio продолжает традиции корпорации icrosoft в области предоставления эффективных инструментальных средств для разработчиков сложного ПО. Обеспечивая среду разработки для всех языков программирования, дополненную набором окон с интуитивно понятными инструментальными средствами, контекстной справкой и автоматизированными механизмами выполнения разнообразных задач разработки, Visual Studio .NET позволяет в сжатые сроки проводить профессиональную разработку программ различного назначения;

• Поддержка нескольких языков программирования – В большинстве профессиональных групп разработчиков, как правило, используется несколько языков программирования – для поддержки такой практики в Visual Studio .NET впервые была обеспечена возможность использования сразу нескольких языков в рамках одной и той же среды. Благодаря применению общего конструктора для компонентов, для форматов XML и HTML, а также наличию единого отладчика, Visual Studio .NET предоставляет разработчикам эффективные средства, независимые от языка программирования. Разработчикам ПО при использовании Visual Studio .NET уже не придется ограничиваться одним языком программирования, адаптируя свою рабочую среду к особенностям этого языка. Более того, Visual Studio .NET позволяет программистам многократно использовать уже имеющиеся у них наработки, а также навыки разработчиков, создающих свои программы на разных языках программирования;

• Единая модель программирования для всех приложений - При создании приложений ранее разработчикам приходилось использовать различные приемы программирования, которые существенным образом зависели от типа приложения — технологии разработки клиентского программного обеспечения, общедоступных веб-приложений, программного обеспечения для мобильных устройств и бизнес-логики промежуточного уровня значительно различались между собой. Среда разработки Visual Studio .NET решает данную проблему, предоставляя в распоряжение разработчиков единую модель создания приложений всех категорий. Эта интегрированная модель обладает привычным и одновременно интуитивно понятным интерфейсом, позволяя разработчикам использовать свои навыки и знания для эффективного создания широкого спектра приложений,

• Всесторонняя поддержка жизненного цикла разработки - Среда Visual Studio .NET обеспечивает поддержку всего жизненного цикла разработки: начиная с планирования и проектирования через разработку и тестирование и вплоть до развертывания и последующего управления. Обеспечивая возможность легкого расширения среды разработки посредством включения продуктов независимых разработчиков, Visual Studio .NET предоставляет всестороннюю адаптируемую среду для создания всех приложений, жизненно необходимых для успешной работы современных компаний.

Microsoft Visual Studio 2010 Professional – необходимое средство для независимых разработчиков, позволяющее решать основные задачи разработки. Система упрощает создание, отладку и развертывание приложений на различных платформах, включает встроенную поддержку модели «разработка через тестирование», а также инструменты отладки, которые обеспечивают создание высококачественных решений.

Написание программного кода часто требует одновременной работы с несколькими конструкторами и редакторами. Visual Studio 2010 Professional помогает разработчику организовать цифровое окружение благодаря поддержке нескольких мониторов, что упрощает работу над программами.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов. Например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов процесса разработки программного обеспечения.

В качестве языка программирования был выбран язык C++. Это компилируемый, статически типизированный язык программирования общего назначения.

С++ поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений.

В качестве инструмента разработки пользовательского интерфейса была выбрана широко используемая библиотека Qt. Qt – кросс-платформенный инструментарий разработки программного обеспечения на языке программирования C++. На сегодняшний день Qt – это продукт, широко используемый разработчиками всего мира. Среди известных компаний, которые наиболее активно используют Qt можно назвать такие как: Adobe, AT&T, Cannon, HP, Bosch, Boeing, IBM, Motorola, NASA, NEC, Pioneer, Sharp, Siemens, Sony и Xerox и др.

Qt – полностью объектно-ориентированная библиотека. Новая концепция ведения межобъектных коммуникаций, именуемая «сигналы и слоты», полностью заменяет былую не вполне надежную модель обратных вызовов. Также имеется возможность обработки событий, например, нажатия клавиш клавиатуры, перемещения мыши и т.д. Предоставляемая система расширений позволяет создавать модули, расширяющие функциональные возможности приложений.

При разработке сетевых приложений, как правило, выделяют «собственную» и «случайную» сложность. Собственная сложность связана с ключевыми проблемами предметной области, затрудняющими разработку сетевого приложения, включая:

- выбор подходящих механизмов коммуникаций и создание протоколов для их эффективного использования.

- проектирование сетевых служб, которые рационально используют доступные вычислительные ресурсы.

- эффективное использование параллелизма для достижения надежной и высокой производительности системы.

Случайная сложность вытекает из ограничений, связанных с инструментальными средствами и методами, включая:

- отсутствие в операционной системе собственных типобезопасных, переносимых и расширяемых API.

- широко распространенное применение алгоритмической декомпозиции, что делает неоправданно трудными поддержку и развитие сетевых приложений.

Чтобы избежать «случайной» сложности при разработке, была выбрана популярная инструментальная библиотека для программирования сетевых приложений ACE. Адаптивная коммуникационная среда (ACE) свободно доступная, объектно-ориентированная библиотека, с открытым исходным кодом, реализующая основные модели программирования сетевых приложений. ACE предоставляет богатый набор многоразовых C ++ компонентов, которые выполняют общие задачи коммуникационного программного обеспечения в целом ряде платформ. Задачи коммуникационного программного обеспечение включают демультиплексирования и диспетчеризацию событий, обработка сигналов, инициализацию служб, взаимодействие процессов, разделяемую память управления, маршрутизации сообщений, динамическое конфигурирование распределенных служб, параллельное выполнение и синхронизацию.

ACE упрощает разработку объектно-ориентированных сетевых приложений и сервисов, которые используют взаимодействие процессов и событий, явное динамическое связывание и параллелизма. Кроме того, ACE автоматизирует конфигурирование системы и изменение конфигурации путем динамического связывания сервисов в приложении во время выполнения.

**3.2 Разработка модулей программного средства**

При проектировании структуры программного средства «Сетевой чат» вся система приложения была разделена на несколько модулей, каждый из которых выполняет определенные функции.

**3.2.1 Разработка модуля пользовательского интерфейса**

Модуль пользовательского интерфейса (UI Module) разработан с использованием библиотеки Qt и представляет собой исполняемый (.exe) файл приложения. Он состоит из классов, с помощью которых создается главное окно приложения, а так же элементы управления для удобной работы пользователя с программным средством (главное меню, диалоги настройки, логина, регистрации и др.).

Центральным классом данного модуля является класс MainFraim. Данный класс аккумулирует в себе все действия по созданию и отображению пользовательского интерфейса. Так же он предоставляет методы для управления добавлением и удалением пользователей, добавлением сообщений. В структурном плане класс MainFrame состоит из списка пользователей, окна отправленных и принятых сообщений, окна для ввода сообщений, главного меню. Создание пользовательского интерфейса разделено на несколько методов, каждый из которых отвечает за создание определенной области или объекта в главном окне приложения. Создание интерфейса главного окна приведено в листинге 3.1.

void MainFrame::SetupUI()

{

QWidget\* mainWidget = new QWidget();

mainWidget->setObjectName(MAIN\_WIDGET\_NAME);

mainWidget->setStyleSheet(MAIN\_WIDGET\_STYLE);

CreateMenuBar();

QHBoxLayout\* mainLayout = CreateMainLayout(CreateUsersWidget(),  
 CreateMessagesWidget());

mainWidget->setLayout(mainLayout);

setCentralWidget(mainWidget);

setWindowTitle(MAIN\_TITLE);

setWindowIcon(QIcon(MAIN\_ICON\_PATH));

resize(MAINFRAME\_WIDTH, MAINFRAME\_HEIGH);

}

**Листинг 3.1 –** **Метод SetupUI класса MainFrame**

**3.2.2 Разработка модуля компонентов**

Модуль компонентов (ComponentsModule) включает в себя несколько объектов для обеспечения дополнительной функциональности программного средства «Сетевой чат». Он представлен в виде отдельной статической библиотеки (.lib) которая используется остальными модулями, включая модуль пользовательского интерфейса.

В данном модуле представлены следующие классы: LoginManager (управление регистрацией и логином), SettingsManager (управление настройками приложения), NetUsersManger (управление сетевыми пользователями).

LoginManager хранит в себе данные всех зарегистрированных пользователей, считывает и сохраняет данные реестре, предоставляет методы для получения данных, а так же для проверки их на уникальности и правильность.

std::vector<UserDataPtr> LoginManagerImpl::GetUsersData() const

{

std::vector<UserDataPtr> users;

std::for\_each(m\_users.begin(), m\_users.end(),

[&users](const std::pair<std::wstring, UserDataPtr>& it)

{

users.push\_back(it.second);

});

return users;

}

**Листинг 3.2 –** **Получение данных зарегистрированных пользователей**

В листинге 3.2 представлен метод класса для получения данных всех зарегистрированных пользователей на данном компьютере

Кроме этого данный класс хранит в себе два состояния online и offline. Объекты, поведение которых зависит от данных состояний, подписываются на событие изменения данного состояния и оповещаются при его изменении.

Класс SettingsManager хранит в себе и управляет настройками программы, определяет IP-адреса сетевых адаптеров и выбирает наиболее подходящий. Данные настройки записываются в реестр и считываются из него при запуске приложении. Другие объекты могут обратиться к SettingsManager для получения текущих настроек. Сохранение настроек в реестре представлено в листинге 3.3.

void SettingsManager::SaveSettings()

{

CRegKey key;

if (ERROR\_SUCCESS == key.Open(HKEY\_CURRENT\_USER, SETTINGS\_KEYS\_PATH,  
 KEY\_WRITE))

{

key.SetStringValue(CURRENT\_NET\_ADDRESS, m\_currentNetAddress.c\_str());

key.SetDWORDValue(CURRENT\_STATE\_MSG\_PORT, m\_stateMsgPort);

key.SetDWORDValue(CURRENT\_CHAT\_MSG\_PORT, m\_chatMsgPort);

}

}

**Листинг 3.3 –** **Сохранение настроек приложения в реестре**

Объект класса NetUsersManager управляет списком подключенных пользователей, и предоставляет другим объектом информацию об именах, сетевых адресах пользователей.

Из-за того, что невозможно уникально идентифицировать пользователя по его сетевому адресу и имени, было принято решение при регистрации нового пользователя генерировать заведомо уникальный идентификатор. Данным идентификатором является GUID.

GUID (Globally Unique Identifier) – статистически уникальный 128-битный идентификатор. Его главная особенность – уникальность, которая позволяет создавать расширяемые сервисы и приложения без опасения конфликтов, вызванных совпадением идентификаторов. Хотя уникальность каждого отдельного GUID не гарантируется, общее количество уникальных ключей настолько велико (2128 или 3,4028×1038), что вероятность того, что в мире будут независимо сгенерированы два совпадающих ключа, практически не возможна. Алгоритм вычисления GUID учитывает текущие дату и время, последовательность «тиков таймера» и её запоминаемое состояние, чтобы можно было обрабатывать и ситуации обратного передвижения часов, простой инкрементальный счетчик, чтобы можно было правильно обрабатывать и очень часто следующие запросы на генерацию очередного GUID, истинно глобально уникальный IEEE-идентификатор машины, извлекаемый из сетевой карты.

Вспомогательная функция для генерирования GUID идентификатора представлена в листинге 3.4.

std::wstring GenerateUUID()

{

UUID uuid = {0};

if (::UuidCreate(&uuid) == RPC\_S\_OK)

{

RPC\_WSTR str = nullptr;

if (::UuidToString(&uuid, &str) == RPC\_S\_OK)

{

std::wstring result(reinterpret\_cast<wchar\_t\*>(str));

::RpcStringFree(&str);

return result;

}

}

return std::wstring();

}

**Листинг 3.4 –** **Вспомогательная функция для генерирования GUID**

**3.2.3 Разработка модуля отправки и получения сообщений**

Модуль отправки и получения сообщений является центральным логическим модулем всей системы. Он спроектирован в виде статический библиотеки, которая используется модулем пользовательского интерфейса.

При отправке и получении сообщений используются UDP сокеты, реализация которых представлена классами ACE\_SOCK\_Dgram и ACE\_SOCK\_Dgram\_Bcast ( отправка широковещательных запросов) библиотеки ACE. В механизме обмена сообщениями участвуют два типа сообщений: сообщения о состоянии клиентов, и непосредственно текстовые сообщения. Первый тип используется для обнаружения пользователей в сети и обмена контактами, другой тип для отправки текстовых сообщений чата.

Центральными классами данного модуля являются классы StateMessagesManager и ChatMessagesManager. В своей реализации они отвечают за отправку и прием сообщений. Для данной задачи они содержат сервер для приема и обработки сообщений и выполняют роль клиентов при их отправке. Метод отправки широковещательного сообщения представлен в листинге 3.5.

Работа менеджеров связана с несколькими потоками, поэтому для их синхронизации используются такие объекты как мьютексы и условные переменные.

Необходимо отметить, что сообщения отправляются в формате xml. Это гарантирует их правильный прием. При нарушении структуры сообщения из-за потери данных или других причин, данное сообщение игнорируется. Таким образом, гарантируется, что все принятые сообщения валидны.

void StateMessagesManager::SendBroadcastMessage(State currentState)

{

net::WSAStartupHolder wsaHolder(MAKEWORD(2, 2));

if (wsaHolder.GetErrorCode() == 0)

{

std::wstring message(CreateMessage(currentState));

ACE\_INET\_Addr currentAddr(TEMP\_PORT, m\_settingsHolder->GetAddress().c\_str());

ACE\_SOCK\_Dgram\_Bcast udpBroadcastSocket(currentAddr);

size\_t result = udpBroadcastSocket.send(message.c\_str(),  
 message.size() \* sizeof(wchar\_t), m\_settingsHolder->GetPort());

udpBroadcastSocket.close();

}

}

**Листинг 3.5 –** **Отправка широковещательных сообщений**

Механизм получения сообщений заключается в приеме и обработке сообщений потоком сервера. Затем сообщение помещается в очередь, после чего оповещается главный поток. Для оповещения главного потока мы создаем скрытое невидимое окно, задача которого принимать оповещения из других потоков. Нотификации происходят с помощью API функции PostMessage. Данная функция является асинхронной. Главный поток извлекает сообщение из очереди и обрабатывает его. В листинге 3.6 представлена функция обработки сообщений скрытым окном, все действия происходят уже в главном потоке.

LRESULT CALLBACK WindowProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

MessagesReceiver\* msgReceiver = reinterpret\_cast<MessagesReceiver\*>(wParam);

if (msgReceiver != nullptr)

{

switch(message)

{

case WM\_PROCESS\_STATE\_MSG:

msgReceiver->ProcessStateMessage();

return 0;

case WM\_PROCESS\_CHAT\_MSG:

msgReceiver->ProcessChatMessage();

return 0;

}

}

return DefWindowProc(hWnd, message, wParam, lParam);

}

**Листинг 3.6 –** **Функция обработки сообщений скрытым окном  
в главном потоке**

Для парсинга принятых xml данных используется библиотека с открытым исходным кодом PugiXml. Данная библиотека позволяет быстро и безопасно как создать, так и извлечь данные из xml строк и документов. В листинге 3.7 приведена функция для извлечения сообщения из принятых по сети данных.

ChatMessagePtr ChatMessagesHandler::GetMsgDataFromXml(const std::wstring& xmlMsg)

{

ChatMessagePtr chatMsg(nullptr);

pugi::xml\_document messageDoc;

pugi::xml\_parse\_result result = messageDoc.load(xmlMsg.c\_str());

if (result)

{

pugi::xml\_node messageNode = messageDoc.first\_child();

pugi::xml\_node userNode = messageNode.child(USER\_NODE);

std::wstring uuid = userNode.child\_value(USER\_UUID\_NODE);

pugi::xml\_node dataNode = messageNode.child(DATA\_NODE);

std::wstring message = dataNode.text().as\_string();

chatMsg.reset(new ChatMessage(uuid, message));

}

return chatMsg;

}

**Листинг 3.7 –** **Использования библиотеки PugiXml   
для извлечения данных сообщения**

**4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

**4.1 Выбор методики тестирования**

Цель испытания программного средства «Сетевой чат» является устранение возможных ошибок и неполадок в работе программного обеспечения. Задачей тестирования приложения было обеспечение высокого качество программного средства. Качество программного обеспечения (ПО) можно определить как совокупную характеристику исследуемого ПО с учётом таких составляющих как: надёжность, сопровождаемость, практичность, эффективность, мобильность, функциональность.

Одним из наиболее важных видов тестирования является функциональное. Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности приложения в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает. Был разработан набор функциональных тест-кейсов, сгруппированных в сценарии в соответствии с ролями пользователей системы.

Для тестирования качества и надежности программного кода было использовано юнит-тестирование или тестирование методом «белого ящика». Оно предполагает процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Целью обеспечения модульного тестирования было изолирование отдельных частей программы и демонстрация, что по отдельности эти части работоспособны.

**4.2 Тестирование программного средства**

Для тестирования основных функциональных требований программного средство были разработаны набор тест-кейсов. Они охватывают функции, которые непосредственно предоставляются пользователю после запуска приложения.

Таблица 4.1 – Пример функциональных тест-кейсов приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Модуль | Описание теста | Ожидаемый результат | Статус |
| 1 | 2 | 3 | 4 |
| Главное окно | Запуск приложения в операционной среде Windows. | Открытие главного окна приложения, появление диалогового окна регистрации нового пользователя. | выполнено успешно |
| Логин-модуль | Запуск приложения | Открытие главного окна приложения появление диалогового окна логина нового пользователя. | выполнено успешно |
| Ввод имени и пароля | Логин успешно выполнен | выполнено успешно |
| Логин-модуль | Запуск приложения | Открытие главного окна приложения, появление диалогового окна логина нового пользователя | выполнено успешно |
| Ввод неверного имени и пароля | В логине отказано | выполнено успешно |
| Регистрация | Запуск приложения | Открытие главного окна приложения, появление диалогового окна логина нового пользователя. | выполнено успешно |
| Ввод имени и пароля | Регистрация успешна | выполнено успешно |
| Логин-модуль | Запуск приложения | Открытие главного окна приложения, появление диалогового окна логина нового пользователя. | выполнено успешно |
| Ввод пустого имени и/или пароля | В регистрации отказано | выполнено успешно |
| UI  модуль | Запуск приложения | Открытие главного окна приложения, появление диалогового окна логина нового пользователя. | выполнено успешно |
| Запуск еще одного экземпляра приложения | Открытие главного окна приложения, появление диалогового окна логина нового пользователя | выполнено успешно |
| Сетевой модуль | Запуск приложения | Открытие главного окна приложения появление диалогового окна логина нового пользователя | выполнено успешно |
| Ввод имени и пароля | Логин успешно выполнен | выполнено успешно |
| Выбор пользователя в списке контактов и отправка текстового сообщения | Сообщение доставлено успешно | выполнено успешно |
| Выбор другого пользователя в списке контактов и отправка текстового сообщения | Сообщение доставлено успешно | выполнено успешно |

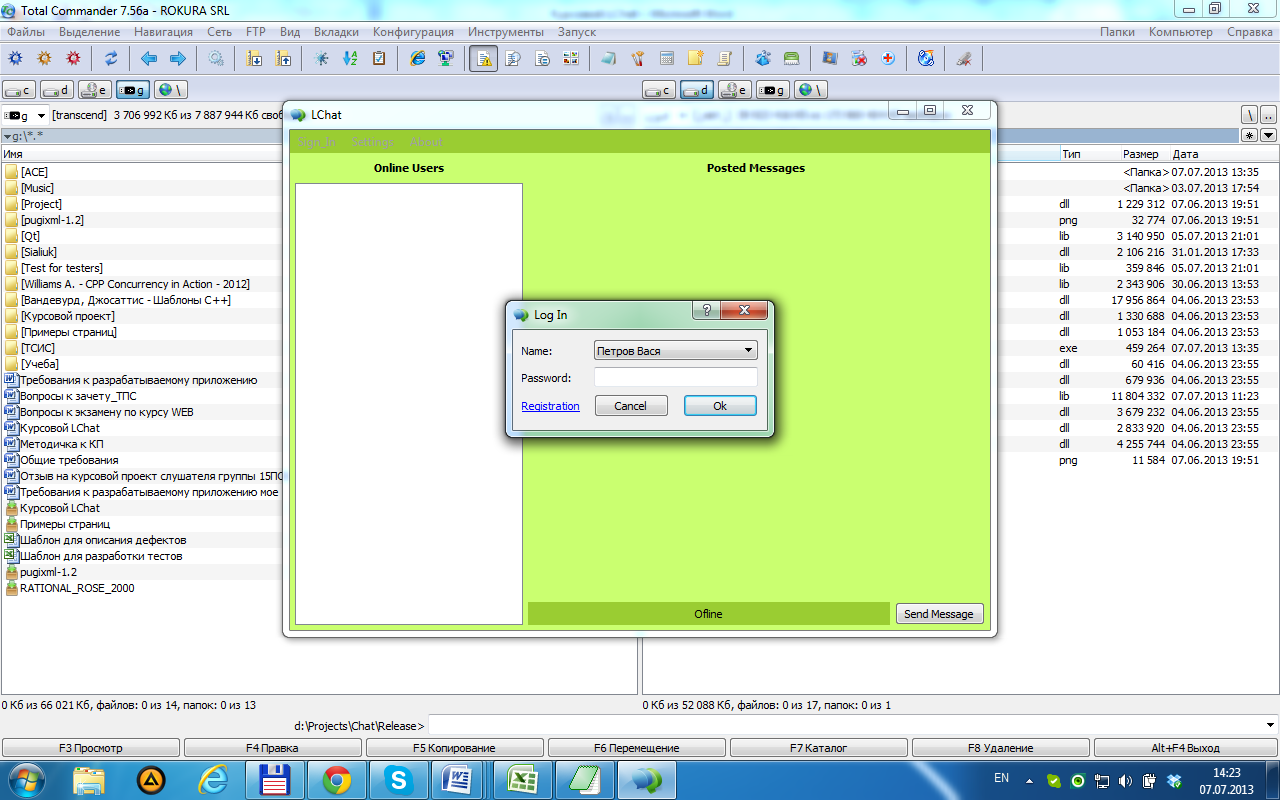
Проведенное модульное тестирование позволяет осуществлять рефакторинг существующего программного кода, гарантируя, что модуль по-прежнему работает корректно.

При проведении модульного тестирования использовались готовые автоматизированные средства юнит-тестирования из библиотеки Boost Test. Данная библиотека обеспечивает подходящий набор компонентов для написания тестовых программ, организации тестов и создания, как одиночных тестов, так и их наборов, при этом максимально обеспечивая контроль их выполнение.

**5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

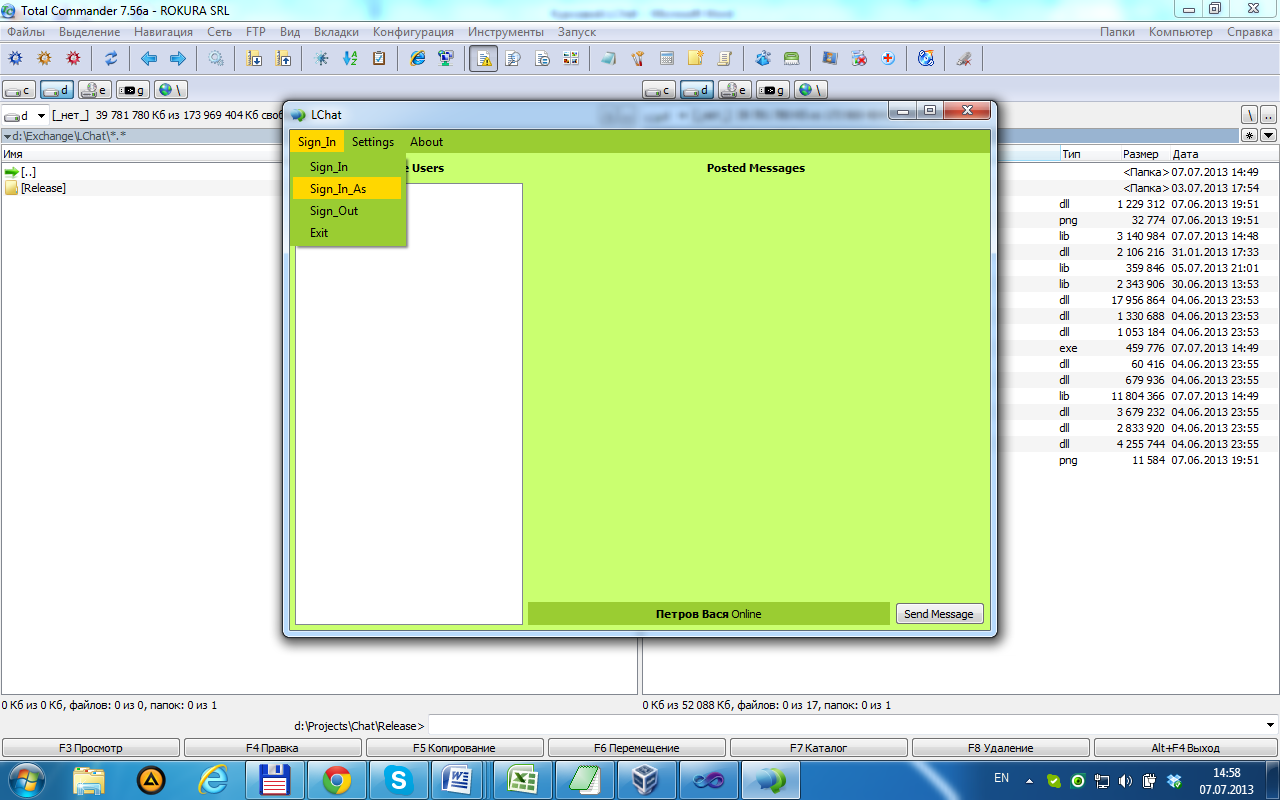
Программа, разработанная в данном курсовом проекте, является сетевым чатом и предназначена для общения пользователей локальной сети в режиме реального времени при помощи обмена короткими текстовыми сообщениями.

После запуска приложения появляется главное окно и диалог входа в систему. Если приложение запущено впервые на компьютере появляется диалог регистрации пользователей. Главное окно приложения представлено на рисунке 5.1.



**Рисунок 5.1 –** **Главное окно приложения «Сетевой чат»**

После успешного логина или регистрации главное окно активизируется, и пользователь переходит в режим онлайн. Если в сети есть подключенные пользователи, они отобразятся в списке контактов.

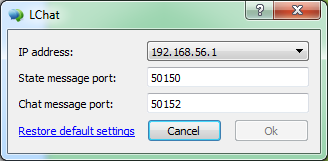


**Рисунок 5.2 –** **Главное меню приложения «Сетевой чат»**

Для отправки сообщения достаточно выбрать нужного пользователя в списке контактов набрать сообщение и нажать кнопку «Send Messаge».

С помощью главного меню пользователь может закрыть приложения, сменить пользователя, изменить настройки программы. Главное меню программного средства представлено на рисунке 5.2.

Диалог настроек приложения позволяет сменить текущий IP адрес, и номера портов в случае необходимости. Диалог настроек представлен на рисунке 5.3.



**Рисунок 5.3 –** **Диалог настроек** **приложения «Сетевой чат»**

Программное средство предназначено для работы в операционных системах Windows XP, Windows Vista, Windows 7, и имеет следующие требования к оборудованию:

- CPU AMD Duron 1000 Mhz;

- оперативная память 256 Mb;

- 50 Mb свободного места на жестком диске;

- сетевой адаптер Ethernet;

- подключение к локальной сети.

**ЗАКЛЮЧЕНИЕ**

В данной работе было разработано клиент-серверное программное средство «Сетевой чат» с возможностью обмена короткими текстовыми сообщениями.

Для разработки приложение использовалась интегрированная среда разработки Visual Studio. Данная система упрощает создание, отладку и развертывание приложений на различных платформах, включает встроенную поддержку модели «разработка через тестирование», а также инструменты отладки, которые обеспечивают создание высококачественных решений.

В ходе проектирования приложения было принято решение о реализации приложения в виде одной программы, выполняющей роль, как клиента, так и сервера. Был разработан примерный алгоритм работы каждого модуля приложения.

Заключительным этапом работы явилась непосредственная реализация модуля отправки и получения сообщений. Полученный программный продукт работоспособен, имеет необходимую функциональность – возможность передачи текстовых сообщений между двумя компьютерами, подключенными к локальной вычислительной сети.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма [и др.]; Питер; под общ. ред. Э. Гамма. – СПб, 2007. – 366 с.

2. Рихтер, Дж. Windows для профессионалов. Создание эффективных WIN32-приложений с учетом специфики 64-разрядной версии Windows / Дж. Рихтер. – СПб: Питер, 2001. – 752 с.

3. Шмидт, Д. Программирование сетевых приложений на С++ / Д. Шмидт, С. Хьюстон. – Москва: Бином-Пресс, 2003. – 304 с.

4. Boost C++ Libraries [Electronic resource]. – 2004. – Mode of access: http://www.boost.org. – Date of access: 25.06.2013.

5. Qt Project [Electronic resource]. – 2013. – Mode of access: http://qt-project.org/. – Date of access: 26.06.2013.

6. The Adaptive Communication Environment (ACE) [Electronic resource]. – 2003. – Mode of access: <http://www.cs.wustl.edu/~schmidt/ACE.html>. – Date of access: 20.06.2013.

**ПРИЛОЖЕНИЕ А**

**Исходный код класса UDP Server**

UDPMessageServer::UDPMessageServer(IMessagesHandler\* handler,  
 IServerSettingsHolder\* settingsHolder)

: m\_startupHolder(MAKEWORD(2, 2))

, m\_msgHandler(handler)

, m\_settingsHolder(settingsHolder)

, m\_shouldShutdown(false)

, m\_needReset(false)

, m\_udpSocket(nullptr, DeleteSocket)

, m\_thread(nullptr)

{

if (m\_startupHolder.GetErrorCode() == 0)

{

Initialize();

}

}

UDPMessageServer::~UDPMessageServer()

{

m\_thread->join();

}

void UDPMessageServer::Run()

{

while (!ShouldShutdown())

{

if (login::LoginManager::Instance()->IsOnline())

{

ACE\_INET\_Addr userAddr;

iovec buffer = {0};

size\_t result = m\_udpSocket->recv(&buffer, userAddr, 0,  
 &TIMEOUT);

if ((result > 0) && buffer.iov\_base)

{

std::wstring message((wchar\_t\*)buffer.iov\_base,  
 (wchar\_t\*)(buffer.iov\_base + buffer.iov\_len));

if (!message.empty())

{

m\_msgHandler->HandleMessage(message, userAddr);

}

delete[] buffer.iov\_base;

}

}

else

{

WaitForEvents();

}

if (NeedReset())

{

InitSocket();

}

}

}

void UDPMessageServer::Shutdown()

{

{

Lock lock(m\_mutex);

m\_shouldShutdown = true;

}

m\_condVariable.notify\_one();

}

void UDPMessageServer::Reset()

{

Lock lock(m\_mutex);

m\_needReset = true;

}

void UDPMessageServer::Notify()

{

m\_condVariable.notify\_one();

}

void UDPMessageServer::Initialize()

{

InitSocket();

m\_thread.reset(new Thread(&UDPMessageServer::Run, this)); //start thread

}

void UDPMessageServer::InitSocket()

{

ACE\_INET\_Addr serverAddr(m\_settingsHolder->GetPort(),  
 m\_settingsHolder->GetAddress().c\_str());

m\_udpSocket.reset(new ACE\_SOCK\_Dgram(serverAddr));

}

bool UDPMessageServer::ShouldShutdown()

{

Lock lock(m\_mutex);

return m\_shouldShutdown;

}

bool UDPMessageServer::NeedReset()

{

Lock lock(m\_mutex);

return m\_needReset;

}

void UDPMessageServer::WaitForEvents()

{

Lock lock(m\_mutex);

m\_condVariable.wait(lock);

}

**ПРИЛОЖЕНИЕ Б**

**Исходный код класса ChatMessageManager**

ChatMessagesManager\* ChatMessagesManager::Instance()

{

static ChatMessagesManager manager;

return &manager;

}

ChatMessagesManager::ChatMessagesManager()

: m\_msgQueue(new ChatMessagesQueue())

, m\_settingsHolder(new ChatServerSettingsHolder(sm::SettingsManager::Instance()))

{

login::LoginManager::Instance()->Subscribe(this);

}

ChatMessagesManager::~ChatMessagesManager()

{

m\_server->Shutdown();

}

void ChatMessagesManager::Send(const std::wstring& uuid, const std::wstring& txtMessage)

{

net::WSAStartupHolder wsaHolder(MAKEWORD(2, 2));

if ((wsaHolder.GetErrorCode() == 0)

&& net::NetUsersManager::Instance()->IsUserExist(uuid))

{

std::wstring message(CreateMessage(txtMessage));

std::wstring addr(net::NetUsersManager::Instance()->GetNetUserAddress(uuid));

ACE\_INET\_Addr userAddr(m\_settingsHolder->GetPort(), addr.c\_str());

ACE\_INET\_Addr currentAddr(TEMP\_PORT, m\_settingsHolder->GetAddress().c\_str());

ACE\_SOCK\_Dgram udpSocket(currentAddr);

udpSocket.send(message.c\_str(), message.size() \* sizeof(wchar\_t), userAddr);

udpSocket.close();

}

}

void ChatMessagesManager::Activate(MessagesReceiver\* receiver)

{

if (!m\_server)

{

m\_msgHandler.reset(new ChatMessagesHandler(receiver, m\_msgQueue.get()));

m\_server.reset(new UDPMessageServer(m\_msgHandler.get(),  
 m\_settingsHolder.get())); //init socket and run server

}

}

ChatMessagesQueue\* ChatMessagesManager::GetMessagesQueue()

{

return m\_msgQueue.get();

}

void ChatMessagesManager::ResetServer()

{

m\_server->Reset();

}

std::wstring ChatMessagesManager::CreateMessage(const std::wstring& txtMessage)

{

try

{

login::UserDataPtr user(login::LoginManager::Instance()->GetCurrentUser());

return boost::str(boost::wformat(CHAT\_MESSAGE\_FORMAT) % user->uuid   
 % txtMessage);

}

catch (const std::exception&)

{

return std::wstring();

}

}

void ChatMessagesManager::OnlineStateChanged()

{

if (login::LoginManager::Instance()->IsOnline())

{

m\_server->Notify();

}

}

РЕЦЕНЗИЯ

на курсовой проект Селюка Ивана Чеславовича

слушателя группы 15ПО11-04з по специальности 1-40 01 73   
«Программное обеспечение информационных систем»

на тему

«Программное средство «Сетевой чат»

Развитие компьютерных сетей привело к появлению различных средств общения, так называемых чатов. Использование чатов упрощает передачу информации между пользователями в рамках внутренних локальных сетей предприятий, а также помогает наладить общения в домашних сетях.

В курсовом проекте перед слушателем была поставлена задача разработки программного средства для передачи текстовых сообщений в локальной сети. Слушатель проанализировал предметную область, выполнил проектирование и разработку информационной системы для решения поставленной задачи.

При проектировании программного средства слушатель применил многопоточное программирование, паттерны объектно-ориентированного проектирования и др. средства. Разработка приложения велась в среде Visual Studio 2010, на языке программирования – С++. Модули программного средства используют в своей работе эффективные библиотеки, как для работы с пользовательским интерфейсом (Qt), так и для передачи данных по сети (ACE).

Среди достоинств разработанного программного средства можно выделить: простоту и легкость использования, быструю и простую настройку сетевых параметров для работы приложения. Среди недостатков: отсутствие модуля помощи для пользователей, недостаточный анализ существующего программного обеспечения для передачи сообщений в локальной сети.

Курсовой проект имеет дальнейшие пути развития и усовершенствования путем добавления передачи файлов, созданием книги контактов пользователей, сохранение переписки в базе данных.

Слушатель справился с поставленной задачей и заслуживает положительной оценки.

\_\_\_\_\_\_\_\_\_\_\_\_\_ Кравцова А.А.

(подпись)