МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств ВТ и автоматизированных систем»

Выпускная квалификационная работа бакалавра

на тему

Чат с авторизацией базы данных для обмена данными в локальной сети

Студент 03.06.20 Белов Владислав Александрович
(подпись, дата) (ФИО полностью)

Руководитель 03.06.20 Зинкин С.А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр Кучин А.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 23.06.2020 № 15)

Секретарь ГЭК Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А.Митрохин
«30» апреля 2020г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Белов Владислав Александрович гр. 16вв2 факультета вычислительной техники.

2.Тема работы «Чат с авторизацией базы данных для обмена данными в локальной сети».

Тема утверждена приказом ПГУ № 0335а/о от "30" апреля 2020г.

3. Руководитель бакалаврской работы д.т.н., профессор каф. ВТ Зинкин С.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.).
Назначение разработки- клиент-серверное приложение, предназначенное для обмена данными в локальной сети, имеющее чат с авторизацией базы данных.

5. Перечень подлежащих разработке вопросов.
Требуется произвести анализ предметной области, определить архитектуру приложения, выбрать языки программирования для клиентского и серверного приложения, выбрать способ взаимодействия между клиентом и сервером, разработать протокол взаимодействия.

6. Календарный график выполнения работы(в сумме 90 часов)

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	30.04.2020-03.05.2020	
2.	Выбор архитектуры приложения, языков программирования, способа взаимодействия	10	03.05.2020-05.05.2020	
3.	Разработка протокола взаимодействия	10	05.05.2020-08.05.2020	
4.	Разработка серверной части приложения	20	08.05.2020-19.05.2020	
5.	Разработка клиентской части приложения	20	19.05.2020-28.05.2020	
6.	Тестирование и отладка	10	28.05.2020-31.05.2020	
7.	Оформления выпускной работы	20	31.05.2020-03.06.2020	

Дата выдачи задания "30"апреля 2020г.

Руководитель бакалаврской работы _____ 30.04.20 Зинкин С.А.
(подпись, дата)

Задание к исполнению принял студент _____ 30.04.20 Белов В.А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 95 листов, 22 рисунка, 2 таблицы, 10 источников, 4 приложения.

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, БАЗА ДАННЫХ, ПРОТОКОЛ TCP/IP, SQL-ЗАПРОСЫ, СУБД, ЭКСПОРТ ДАННЫХ.

Цель работы – разработка чата для обмена данными в локальной сети. Система должна обеспечивать взаимосвязь сотрудников компании между собой в развёрнутой сети, позволять обмениваться текстовыми сообщениями. Также система должна обрабатывать процесс авторизации и реализовывать добавление новых контактов в список контактов текущего пользователя.

Результатом разработки является проект, удовлетворяющий поставленной задаче. А именно, разработано, реализовано и протестировано приложение для обмена данными между сотрудниками компании в локальной сети.

Разработка производилась с использованием масштабного стека технологий, к которым относятся: язык программирования Java, язык структурированных запросов SQL, среда разработки NetBeans. Для хранения данных выбрана СУБД MySQL, а для её администрирования применялся инструмент MySQL Command Line.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ 6
ВВЕДЕНИЕ 7
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9
1.1 Анализ архитектуры системы работы чата 9
1.2 Анализ функциональных блоков чата 10
2 ТРЕБОВАНИЯ К РАЗРАБОТКЕ ПРИЛОЖЕНИЯ ДЛЯ ОБМЕНА СООБЩЕНИЯМИ В ЛОКАЛЬНОЙ СЕТИ 13
3 ВЫБОР ТЕХНОЛОГИЙ, ЯЗЫКОВ И СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ.15
3.1 Программные средства, используемые при разработке системы 15
3.2 NetBeans IDE 16
3.3 Java 17
3.4 SQL 18
3.5 MySQL Command Line 19
3.6 Maven 19
3.7 SWING 20
3.8 Server Socket 21
3.9 JDBC 23
3.10 Взаимодействие рассмотренных средств 24
4 РЕАЛИЗАЦИЯ 26
4.1 Алгоритм работы чата для обмена сообщениями 26
4.2 Описание структуры и таблиц БД 27
4.3 Основные функции доступные пользователям 28
4.4 Схема взаимодействия клиента и сервера 31
5 МЕТОДЫ ИСПЫТАНИЙ 33
5.1 Добавление нового сотрудника 33
5.2 Удаление сотрудника 35
5.3 Авторизацтя 37
5.4 Выход из сети 39
5.5 Изменение настроек подключения 40
5.6 Добавление контактов 42
5.7 Просмотр сообщений 42
5.8 Обмен сообщениями 42
6 БЖД 45

ЗАКЛЮЧЕНИЕ 45
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45
ПРИЛОЖЕНИЕ А 47
ПРИЛОЖЕНИЕ Б 50
ПРИЛОЖЕНИЕ В 51
ПРИЛОЖЕНИЕ Г 76

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БД - база данных

СУБД - система управления базой данных

ORM - (англ.Object-relational Mapping) - объектно-реляционное отображение

ПО - программное обеспечение

ОС - операционная система

ЯП - язык программирования

GUI - (англ. Graphical User Interface) - графический интерфейс пользователя

SQL - (англ. Structured Query Language) - язык структурированных запросов

URL - (англ. Uniform Resource Locator) - единый указатель ресурса

ВВЕДЕНИЕ

Сегодня информацию рассматривают как один из основных ресурсов развития общества, а информационные системы и технологии как средство повышения производительности и эффективности работы людей.

Создание современных электронных вычислительных машин позволило автоматизировать обработку данных во многих сферах человеческой деятельности, в первую очередь там, где необходимо максимально быстро обмениваться информацией. Без современных систем обмена данными трудно представить сегодня передовые производственные технологии, управление большим количеством людей, научные исследования, образование, издательское дело, функционирование средств массовой информации, проведение крупных спортивных состязаний. Значительно расширило сферу применения систем обработки данных появление персональных компьютеров.

Возможность передачи информации на расстоянии интересовала людей всегда. Для того чтобы обмениваться информацией не только при личной встрече, но и на больших расстояниях, люди изобретали всё новые технические средства, протягивали структурированные кабельные сети по всему миру, организовывали почтовые системы, запускали спутники связи. С развитием информационных технологий стали возможным еще более глобальные коммуникации. Сегодня люди с легкостью по всему миру могут обмениваться не только сообщениями, но и графикой и звуковой и другими видами информации.

Историческим «докомпьютерным» предшественником чатов, несомненно, являлся телефон. Ни телеграф, ни почта не могли позволить общаться в реальном времени и не были доступны в домашних условиях. Изобретение и распространение телефона вызвало настоящую революцию

способах и средствах общения. Возможность поговорить с собеседником в реальном времени на другом континенте казалась настоящим чудом.

Язык Java является основой практически для всевозможных типов сетевых приложений и всеобщим стандартом для разработки встроенных и мобильных приложений, веб-контента, игр, и корпоративного программного обеспечения. В мире насчитывается более девяти миллионов специалистов, разрабатывающих приложения на Java, которая позволяет эффективно разрабатывать, тестировать, внедрять и использовать программное обеспечение.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

При разработке программного продукта необходимо провести анализ существующих вариантов решения задач. В данном разделе рассматриваются общие теоретические вопросы, связанные с проектированием и реализацией чата для обмена сообщениями в локальной сети компании или офиса.

Для того чтобы создать программный продукт с заданными параметрами, следует рассмотреть его архитектуру и функциональные блоки, без которых приложение не сможет корректно работать.

1.1 Анализ архитектуры системы работы чата

Архитектура системы работы чата является трехзвенной архитектурой, которая состоит из трех основных частей:

1. Клиентская часть
2. Серверная часть
3. База данных

Рассмотрим конкретнее, что представляет каждая часть, какова роль каждой в проекте.

Клиентская часть - непосредственно та часть, с которой взаимодействует пользователь. Она представляет собой графический интерфейс, компоненты. Взаимодействие пользователя и программы происходит именно через клиентскую часть.

Серверная часть - программа, которая принимает и обрабатывает запросы от пользователей. Серверную часть можно реализовать с помощью различных технологий и языков программирования. Процесс взаимодействия с клиентской частью состоит из получения запроса от клиента, обработка его на сервере и отправка результата данного запроса обратно на клиентскую сторону.

База данных (БД) - совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает

их использование оптимальным образом для одного или нескольких приложений.

Система управления базами данных (СУБД) - программное обеспечение, необходимое для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации. При принятии запроса сервером на взаимодействие с какими- либо данными из БД, он в свою очередь посылает SQL-запрос СУБД, где происходит соответствующая операция по добавлению, модификации, получению или удалению данных. После принятия сервером ответа от БД, он посылает результат запроса обратно на клиентскую часть.

1.2 Анализ функциональных блоков чата по обмену сообщениями в локальной сети

Ни одна система по обмену сообщениями не может обойтись без некоторых функциональных блоков, то есть они являются обязательными для любой программы. К таким важным частям относятся:

- регистрация новых сотрудников;
- удаление сотрудников;
- авторизация персонала;
- отображение списка сотрудников в системе;
- добавление новых контактов;
- обмен сообщениями.

Функционал блока регистрации сотрудников служит для того, чтобы создать учетную запись в базе данных, с целью получения доступа к полному функционалу системы. Учетная запись хранит информацию о главных характеристиках клиента, таких как: фамилия, имя, отчество, отдел в котором он работает и занимаемая должность. Данная информация необходима для того, чтобы осуществить идентификацию пользователя для обеспечения

доступа к функции по обмену сообщениями. Данная информация хранится на стороне сервера.

Функционал удаления сотрудника является неотъемлемой частью по администрированию персоналом в компании. Для удаления работника из системы администратору сервера предоставляется список ранее зарегистрированных участников системы, которых он в свою очередь имеет возможность удалить из системы и БД развёртываемой системы. После удаления записи о служащем, он также удаляется из БД на сервере.

Блок авторизации необходим для входа сотрудника в систему. Данный блок обеспечивает ввод клиентом необходимого перечня данных для его идентификации на серверной стороне и предоставление ему необходимых полномочий для работы в системе.

Отображение списка сотрудников - это крайне необходимая функция для управления персоналом целого офиса. При помощи данного решения администратор сервера может наблюдать за количеством сотрудников в сети в данный момент времени. Из количества часов проведённого в сети и на рабочем месте может складываться статистика проведённого времени на рабочем месте. Вместе с фамилией, именем и отчеством работника также отображается отдел и занимаемая должность работника в текущей отрасли.

Неотъемлемой частью для удобного и непрерывного обмена данными является функционал по добавлению новых контактов. Каждый пользователь имеет право добавить необходимые ему контакты к себе в аккаунт. Все добавленные сотрудники в качестве контактов сохраняться на серверной стороне и будут доступны только пользователю системы, которых их присоединил в свой список.

Важнейшей функцией в разрабатываемой программе является функционал по обмену сообщениями. Все ранее спроектированные и разработанные блоки являются вспомогательными для этой функции. Они упрощают весь процесс обмена сообщениями и позволяют пользователю

делать это максимально быстро и удобно.

Для отправки сообщения сотруднику необходимо быть авторизованным в системе и иметь необходимый список контактов. В противном случае работник не будет иметь прав для реализации функционала по обмену данными с другими сотрудниками в развёртываемой сети.

После авторизации сотрудник имеет доступ к полному функционалу клиентского приложения, следовательно, может добавить необходимый перечень контактов для обмена сообщениями. После проделанной операции потребуется открыть нужный диалог с собеседником и начать процесс по обмену данными.

2 ТРЕБОВАНИЯ К РАЗРАБОТКЕ ПРИЛОЖЕНИЯ ДЛЯ ОБМЕНА СООБЩЕНИЯМИ В ЛОКАЛЬНОЙ СЕТИ

Система обмена сообщениями должна выполнять список основных функций для реализации поставленных требований и задач, к ним относятся:

- со стороны сервера:
  - регистрация нового сотрудника в сети, добавление перечня необходимой информации о нём и предоставление данных для входа в сеть;
  - отображение списка работников, которые находятся в сети и заняты обменом данными.
  - обработка поступающих команд об обмене сообщениями от авторизированных пользователей системы.
- со стороны клиента:
  - функционал авторизации, для идентификации работника в сети и предоставление ему всего перечня прав по эксплуатации системы обмена данными;
  - осуществление функции просмотра всего списка контактов для добавления данных контактов в личный список;
  - возможность выбора диалога, а также просмотра всех ранее созданных сообщений в диалоге с выбранным сотрудником;
  - настройка данных для подключения к серверу, возможность указать хост и порт, по которому будет производиться подключение к серверной стороне.

При запуске клиентского приложения пользователю предоставляется упрощённый интерфейс. После выполнения операции авторизации, где пользователь указывает свои персональные данные для входа и его

идентификации, он может войти в систему как зарегистрированный пользователь и получить статус авторизированный. С этого момента продавец получает все права по эксплуатации клиентского приложения и его функционала.

Сотрудник офиса после авторизации имеет возможность работать над содержимым чата. Подразумевается, что работников в развёртываемой сети по обмене данными будет несколько, так как в обслуживании такого ресурса необходимо много времени и сил.

Клиент-серверное приложение позволяет улучшить работу компании за счет упрощения работы и взаимодействия между собой сотрудников и управляющего персонала с сотрудниками, возможности работы из любого места. Используемая программа по обмену данными позволяет экономить время по осуществлению информационного обмена.

Приложение должно иметь простой интуитивно понятный интерфейс для работы, осуществлять правильную и быструю обработку запросов, хранить большое количество данных о пользователях и их сообщениях в рамках одной базы данных.

3 ВЫБОР ТЕХНОЛОГИЙ, ЯЗЫКОВ И СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

3.1 Программные средства, используемые при разработке системы

Для реализации программного продукта «Чат с авторизацией в базе данных и обменой сообщениями в локальной сети» были рассмотрены следующие технологии, языки и архитектуры:

- NetBeans IDE – среда разработки

- Java - типизированный объектно-ориентированный язык программирования;

- SQL - декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

- MySQL Command Line – инструмент для администрирования СУБД SQL;

- JDBC – библиотека для осуществления подключения к БД для программ языка Java.

- Maven – универсальный сборщик проектов для языка Java.

- Swing – библиотека для создания графического интерфейса для программ на языке Java.

- Server Socket – библиотека языка Java для осуществления обмена информации в сети клиент-сервер.

В разработанном приложении используется архитектура «Клиент - сервер».

Как правило компьютеры и программы, входящие в состав информационной системы, не являются равноправными. Некоторые из них владеют ресурсами (файловая система, база данных и т.д.), другие имеют возможность обращаться к этим ресурсам. Программу, управляющую ресурсом, называют сервером этого ресурса (файл-сервер, сервер базы данных, вычислительный сервер...). Клиент и сервер какого-либо ресурса могут находится как в рамках одной вычислительной системы, так и на различных компьютерах, связанных сетью.

Основной принцип архитектуры "клиент-сервер" заключается в разделении функций приложения на три группы:

- ввод и отображение данных (клиентская сторона);
- прикладные функции, характерные для данной предметной области;
- функции управления ресурсами (серверная сторона).

Поэтому, в любом приложении выделяются следующие компоненты:

- компонент представления данных;
- прикладной компонент;
- компонент управления ресурсом.

Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате. В этом случае двухзвенная архитектура клиент-сервер становится трехзвенной, а к некоторых случаях, она может включать и больше звеньев.

3.2 NetBeans IDE

NetBeans IDE - бесплатная интегрированная среда разработки с открытым исходным кодом для разработчиков программного обеспечения. Среда предоставляет все средства, необходимые для создания профессиональных десктоп приложений, корпоративных, мобильных и веб-приложений на платформе Java, а также C/C++, PHP, JavaScript, Groovy и Ruby.

Возможна разработка Java десктоп приложений с профессиональными графическими интерфейсами пользователя. Используется визуальный редактор - Swing GUI Builder. Работа осуществляется путем перетаскивания элементов графического интерфейса из палитры на холст. Предварительное позиционирование элементов можно осуществлять с помощью указателя мыши. Панель свойств и инспектор компонентов предоставляют возможность тонкой настройки каждого компонента интерфейса.

Основные преимущества данной среды разработки являются:

- рабочая область среды IDE является полностью настраиваемой - существует возможность пользовательской настройки действий, выполняемых с помощью панели, назначения "горячих" клавиш и т.д.

- Производится проверка ошибок во время ввода, отображение вариантов для автозавершения кода и фрагментов документации по требуемому языку программирования.

- Редактор может генерировать и вставлять в исходный код стандартные фрагменты кода на Java или других языках.

- IDE имеет в своем составе расширенный многоязыковой редактор для различных языков программирования - Java, C/C++, PHP, JavaScript, документацию Javadoc и тд. Существует возможность расширения функций редактора с целью поддержки любого другого языка.

3.3 Java

Java - в современном мире Java является одним из самых известных и используемых объектно-ориентированных языков программирования. Данный ЯП разрабатывался как универсальный для решения различного горизонта задач. Java прошел огромный путь развития, в текущий момент является уже не просто ЯП, а целой платформой, которая объединяет огромное количество различных технологий, библиотек и т.д. Все это позволяет вести разработку на Java серьезных enterprise приложений, а большое количество дополнительных модулей дает возможность подобрать конкретные средства для решения поставленной задачи. Многие разработчики выбирают Java, в результате чего растет количество различных решений для возникающих вопросов, а также увеличивается объем литературы, доступной для изучения.

К особенностям Java относится:

- Архитектура языка обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, из этого следует, что программам не нужно перекомпилироваться, чтобы быть используемыми на других платформах, таких как: Windows, Linux, Mac Os и других;

- Java является языком с Си-подобным синтаксисом и близок в этом отношении к C/C++ и C#. Данная особенность позволяет разработчикам, уже имеющих опыт работы с этими языками, быстро приступить к разработке приложений на Java;

- Одной из главных особенностей языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. Затем происходит выполнения данного кода специальной виртуальной машиной Java. Данный принцип отличается от других интерпретируемых языков, таких как PHP или Perl, с другой стороны Java не является чисто компилируемым языком как C или C++;

- В Java присутствует автоматический сборщик мусора, это позволяет разработчику не уделять времени на ручную очистку памяти от ранее использованных объектов, что значительно упрощает и ускоряет работу;

- Java является объектно-ориентированным языком. Объектно-ориентированная модель содержит основные принципы: полиморфизм, наследование, инкапсуляцию. Это позволяет разработчикам на Java создавать сложные, но при этом гибкие, хорошо масштабируемые и расширяемые приложения.

3.4 SQL

SQL - структурированный язык запросов. Он используется для создания, выполнения каких - либо изменений и управления данными в реляционной БД, которая в свою очередь управляется посредством СУБД.

В разрабатываемой программе SQL является основным способом работы с базой данных и позволяет выполнять следующий набор операций:

- создание в базе данных новой таблицы;
- добавление в таблицу новых записей;
- изменение записей;
- удаление записей;
- выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
- изменение структур таблиц.

В первую очередь язык SQL является информационно-логическим языком, что означает, что он предназначен для описания, модификации и получения данных из реляционных БД. SQL позволяет осуществлять создание в БД новых таблиц, добавление данных, изменение значений записей, удаление их, получение значений из таблиц, модификация таблиц.

Со временем SQL стал сложнее и расширился новыми особенностями, поэтому стал приобретать черты, делающие его похожим на языки программирования. Несмотря на все изменения, затронувшие SQL, он является самым распространенным средством для взаимодействия прикладных программ с БД.

3.5 MySQL Command Line

MySQL Command Line – это инструмент для администрирования SQL сервера направленный на постоянную поддержку базы данных с целью поддержания и увеличения её эффективности. Критерий эффективности все больше зависит от безопасности баз данных, а корректное управление учетными записями пользователей гарантирует эффективное функционирование всей компании.

Для предоставления возможности надежного и эффективного управления базами данных, MySQL Command Line был оснащена мощным средством SQL администрирования. Данный инструмент полностью отвечает всем современным требованиям первоклассного SQL-администратора.

3.6 Maven

Мавен - это инструмент для сборки Java проекта: компиляции, создания jar, создания дистрибутива программы, генерации документации. Простые проекты можно собрать в командной строке. Если собирать большие проекты с командной строки, то команда для сборки будет очень длинной, поэтому её иногда записывают в bat/sh скрипт. Но такие скрипты зависят от платформы. Для того чтобы избавиться от этой зависимости и упростить написание скрипта используют инструменты для сборки проекта. Основным приемуществом Maven является то, что основные среды разработки на java легко открывают проекты, которые собираются с помощью maven. При этом зачастую проект настраивать не нужно - он сразу готов к дальнейшей разработке.

3.7 SWING

Технология Swing — это UI Java-платформы. Она выступает как программное обеспечение, управляющее всем взаимодействием пользователя и компьютера. По существу, она служит посредником между пользователем и внутренностями компьютера. Swing предоставляет механизмы для управления аспектами UI:

- Клавиатура: Swing предоставляет способ перехвата пользовательского ввода.
- Цвета: Swing предоставляет способ менять цвета, которые вы видите на экране.
- Текстовое поле для ввода: Swing предоставляет текстовые компоненты для обработки всех повседневных задач.

В любом случае Swing предоставляет вам все инструменты, необходимые для создания вашего собственного UI.

Базовым строительным блоком всей библиотеки визуальных компонентов Swing является JComponent. Это суперкласс каждого компонента. Он является абстрактным классом, поэтому в действительности вы не можете создать JComponent, но он содержит буквально сотни функций, которые каждый компонент Swing может использовать как результат иерархии классов.

JComponent — базовый класс не только для Swing-компонентов, но также и для пользовательских компонентов. Он знает, как обрабатывать все нажатия клавиш на клавиатуре. Подклассы, следовательно, должны только прослушивать определенные клавиши. Также, он содержит метод add(), который позволяет добавлять другие JComponent. Этим способом вы можете добавить любой Swing-компонент к любому другому Swing-компоненту для создания вложенных компонентов (например, JPanel, содержащую JButton, или даже более причудливые комбинации, например JMenu, содержащее JButton).

3.8 Server Socket

Библиотека Server Socket реализует сокеты сервера. Сокет сервера ожидает запросов, чтобы войти по сети. Это выполняет некоторую работу, основанную на том запросе, и затем возможно возвращает результат запрашивающей стороне.

Сокеты обеспечивают механизм связи между двумя компьютерами, использующими TCP. Клиентская программа создает сокет на своем конце связи и пытается подключить этот сокет к серверу.

Когда соединение установлено, сервер создает объект сокета на своем конце связи. Клиент и сервер теперь могут общаться, записывая и считывая данные из сокета.

Класс java.net.Socket представляет собой сокет, а класс java.net.ServerSocket предоставляет механизм для серверной программы для прослушивания клиентов и установления соединений с ними.

Следующие шаги выполняются при установлении соединения TCP между двумя компьютерами с использованием сокетов:

- сервер создает экземпляр объекта ServerSocket, определяющий, по какому номеру порта должна происходить связь;
- сервер вызывает метод accept () класса ServerSocket. Этот метод ожидает, пока клиент не подключится к серверу по указанному порту;
- после ожидания сервера клиент создает экземпляр объекта Socket, указывая имя сервера и номер порта для подключения;
- конструктор класса Socket пытается подключить клиента к указанному серверу и номеру порта. Если связь установлена, у клиента теперь есть объект Socket, способный связываться с сервером;
- на стороне сервера метод accept () возвращает ссылку на новый сокет на сервере, который подключен к клиентскому сокету.

Сокеты обеспечивают механизм связи между двумя компьютерами, использующими TCP. Клиентская программа создает сокет на своем конце связи и пытается подключить этот сокет к серверу.

Когда соединение установлено, сервер создает объект сокета на своем конце связи. Клиент и сервер теперь могут общаться, записывая и считывая данные из сокета.

Класс java.net.Socket представляет собой сокет, а класс java.net.ServerSocket предоставляет механизм для серверной программы для прослушивания клиентов и установления соединений с ними.

Следующие шаги выполняются при установлении соединения TCP между двумя компьютерами с использованием сокетов:

- сервер создает экземпляр объекта ServerSocket, определяющий, по какому номеру порта должна происходить связь;

- сервер вызывает метод accept () класса ServerSocket. Этот метод ожидает, пока клиент не подключится к серверу по указанному порту;

- после ожидания сервера клиент создает экземпляр объекта Socket, указывая имя сервера и номер порта для подключения;

- конструктор класса Socket пытается подключить клиента к указанному серверу и номеру порта. Если связь установлена, у клиента теперь есть объект Socket, способный связываться с сервером;

- на стороне сервера метод accept () возвращает ссылку на новый сокет на сервере, который подключен к клиентскому сокету.

Сервер создает экземпляр объекта ServerSocket, определяющий, по какому номеру порта должна происходить связь.

Сервер вызывает метод accept () класса ServerSocket. Этот метод ожидает, пока клиент не подключится к серверу по указанному порту.

После ожидания сервера клиент создает экземпляр объекта Socket,

указывая имя сервера и номер порта для подключения.

Конструктор класса Socket пытается подключить клиента к указанному серверу и номеру порта. Если связь установлена, у клиента теперь есть объект Socket, способный связываться с сервером.

На стороне сервера метод accept () возвращает ссылку на новый сокет на сервере, который подключен к клиентскому сокету.

После того, как соединения установлены, связь может происходить с использованием потоков ввода / вывода. Каждый сокет имеет и OutputStream, и InputStream. OutputStream клиента подключен к InputStream сервера, а InputStream клиента подключен к OutputStream сервера.

TCP является двусторонним протоколом связи, поэтому данные могут передаваться по обоим потокам одновременно. Ниже приведены полезные классы, предоставляющие полный набор методов для реализации сокетов.

3.9 JDBC

Java Database Connectivity – это стандартный API для независимого соединения языка программирования Java с различными базами данных.

JDBC решает следующие задачи:

- создание соединения с БД;
- создание SQL выражений;
- выполнение SQL – запросов;
- просмотр и модификация полученных записей.

Если говорить в целом, то JDBC – это библиотека, которая обеспечивает целый набор интерфейсов для доступа к различным БД.

Для доступа к каждой конкретной БД необходим специальный JDBC – драйвер, который является адаптером Java – приложения к БД.

JDBC API использует менеджер драйверов и специальные драйверы БД

для обеспечения подключения к различным базам данных.

JBDC Manager проверяет соответствие драйвера и конкретной БД. Он поддерживает возможность использования нескольких драйверов одновременно для одновременной работы с несколькими видами БД.

JDBC API состоит из следующих элементов:

- Менеджер драйверов управляет списком драйверов БД. Каждой запрос на соединение требует соответствующего драйвера. Первое совпадение даёт нам соединение;

- Драйвер (Driver). Этот элемент отвечает за связь с БД. Работать с ним нам приходится крайне редко. Вместо этого мы чаще используем объекты DriverManager, которые управляют объектами этого типа;

- Соединение (Connection). Этот интерфейс обеспечивает нас методами для работы с БД. Все взаимодействия с БД происходят исключительно через Connection;

- Выражение (Statement). Для подтверждения SQL-запросов мы используем объекты, созданные с использованием этого интерфейса;

- Результат (ResultSet). Экземпляры этого элемента содержат данные, которые были получены в результате выполнения SQL – запроса. Он работает как итератор и “пробегает” по полученным данным;

- Исключения (SQL Exception). Этот класс обрабатывает все ошибки, которые могут возникнуть при работе с БД.

3.10 Взаимодействие рассмотренных средств

Взаимодействие можно рассмотреть в рамках клиент-серверной модели, где основными блоками являются серверная часть, клиентская часть и БД. Для разработки серверной части, взаимодействующей с базой данных и частью клиента, а также обеспечивающей полного функционала торговой сети используются следующие средства. Все начинается со среды

разработки, так как без нее создание проектного продукта будет гораздо более громоздким и долгим, для этого мы воспользуемся NetBeans, вдобавок к ней идут средства для автоматизации сборки проектов maven. Следующим очень мощными инструментами являются библиотеки Server Socket и JDBC, именно они позволяет нам быстро передавать данные от БД к клиенту. Java и все выше сказанные средства позволяют описывать работу веб-сервера, те запросы, которые он должен обрабатывать и ответный результат, который он должен формировать. Без сервера приложения просто не будет существовать. Ни одна подобная система не обойдется без хранения и взаимодействия с данными.

4 РЕАЛИЗАЦИЯ

Прежде чем начинать непосредственную разработку всех частей приложения, необходимо еще раз проанализировать систему, какой функционал необходимо реализовывать, какие сущность должны находиться в базе данных, какие должны быть атрибуты у этих сущностей. Также необходимо рассмотреть, как именно будет взаимодействовать клиент, сервер и БД, какие для этого нужны будут механизмы и т.д.

4.1 Алгоритм работы чата для обмена сообщениями в локальной сети

Для того чтобы реализовать систему нужно подготовить клиента, сервера и СУБД.

Вся последовательность работы состоит из следующего алгоритма: пользователь своими действиями создает определенные события, которые обрабатываются клиентской частью, в зависимости от того что сделал клиент, серверу посылается запрос на обработку соответствующей задачи.

Если во время выполнения запроса возникнет ошибка, то сервер вернет клиенту информацию об этой ошибке и не выполнит действие, если запрос пройдет успешно, то сервер обратится к СУБД для получения данных, запрашиваемых пользователем.

В случае, когда при обращении к СУБД возникнет ошибка, то сервер соответствующим способом обработает ее и вернет сообщение об ошибке клиенту, в том же случае, если исключительных ситуаций не возникло, то запрос является выполненным и сервер вернет клиенту информацию об успешном выполнении запроса.

Диаграмма последовательности работы системы и диаграмма развертывания системы обмена сообщениями изображена на рисунке 1, а на рисунке 2 отображается диаграмма развёртывания системы для обмены сообщениями.

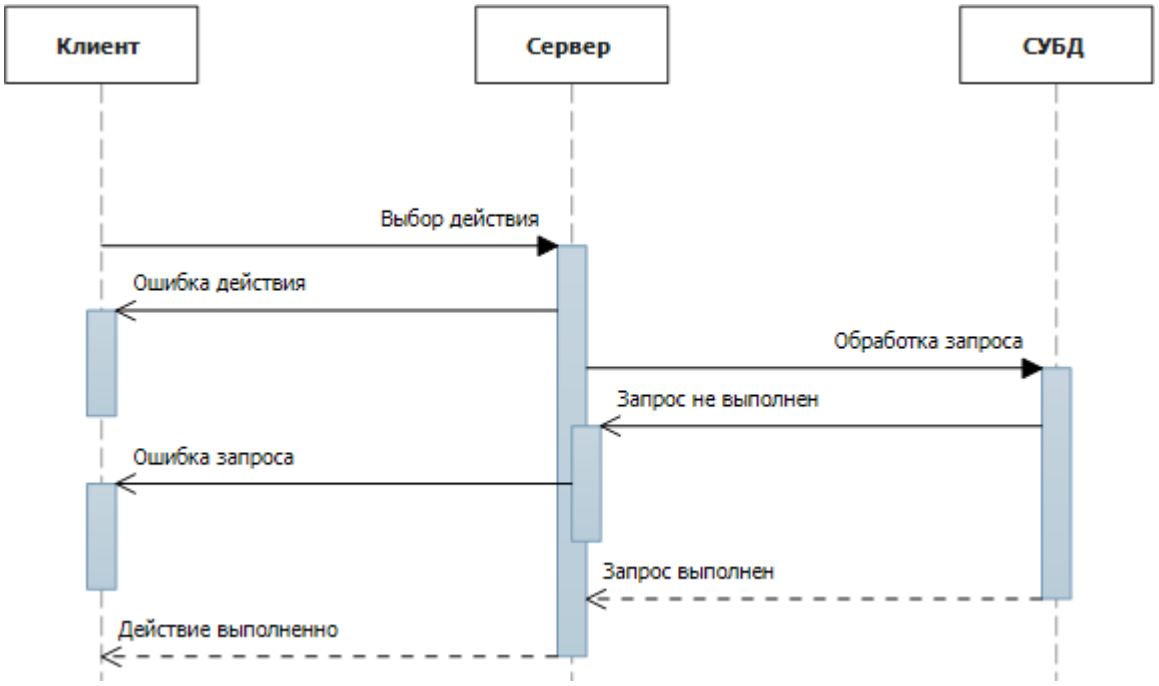


Рисунок 1 – UML диаграмма последовательности работы системы

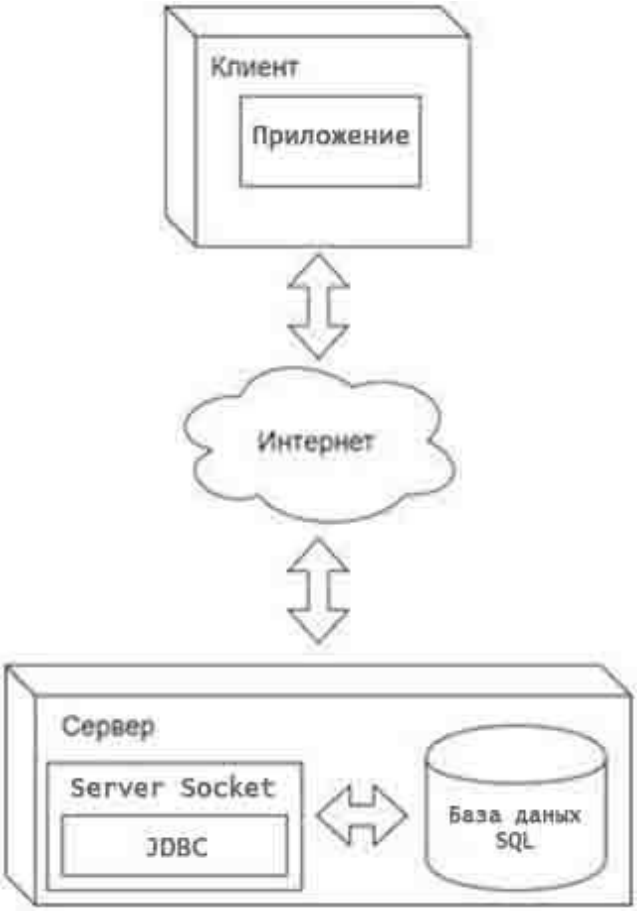


Рисунок 2 – UML диаграмма развертывания системы обмена сообщениями

4.2 Описание структуры и таблиц БД

Для разработки структуры БД, необходимо проанализировать какие сущности требуют рассмотрения, и какие атрибуты будут им принадлежать. Дополнительно необходимо выявить, как данные сущности будут взаимосвязаны. Все это является работой по созданию ERD (англ. Entity Relation Diagram), т.е. диаграммы взаимодействия сущностей.

Для нормальной работы системы по обмену сообщениями были разработаны следующие таблицы:

- users - таблица, содержащая список работников и данные о каждом зарегистрированном сотруднике;

- contacts - таблица, которая содержит данные о всех контактах для каждого из зарегистрированных пользователей;

Таблица 1 - Содержимое таблицы «users» (сотрудники).

Имя столбца	Описание	Тип данных	Пример
ID	Идентификатор пользователя	INTEGER	4038
NAME	ФИО работника	VARCHAR(40)	Терентьев М.П.
DEPARTMENT	Раздел, в котором работает сотрудник	VARCHAR(15)	IOS
POSITION	Занимаемая должность	VARCHAR(15)	Дизайнер
LOGIN	Логин для входа в систему	VARCHAR(10)	teren
PASSWORD	Пароль	VARCHAR(10)	teren123

Таблица 2 - Содержимое таблицы «contacts» (список контактов).

Имя столбца	Описание	Тип данных	Пример
ID	Идентификатор контакта	INTEGER	5970

4.3 Основные функции доступные пользователям

Система работы чата для обмена сообщениями в локальной сети организует доступ к функционалу только авторизированным пользователям. Пользователь в свою очередь может быть авторизован в системе или не авторизован. Распределение обязанностей сотрудников строится на договоренностях, построенных при личном взаимодействии, что означает, что программно все работники имеют одинаковый уровень доступа и взаимодействия со средой сайта.

Возможности, предоставленные категории идентифицированных пользователей сайта, можно рассмотреть в виде диаграммы использования. Данная диаграмма наглядно и полно показывает все функции, которыми они обладают.

Пользователю доступен следующий функционал по использованию клиентского приложения:

- добавление нового сотрудника;
- удаление сотрудника;
- авторизация;
- выход из сети;
- изменение настроек подключения;
- добавление контактов;
- просмотр личных сообщений;
- обмен сообщениями.

Диаграмма вариантов использования для пользователя представлена на рисунке 3.

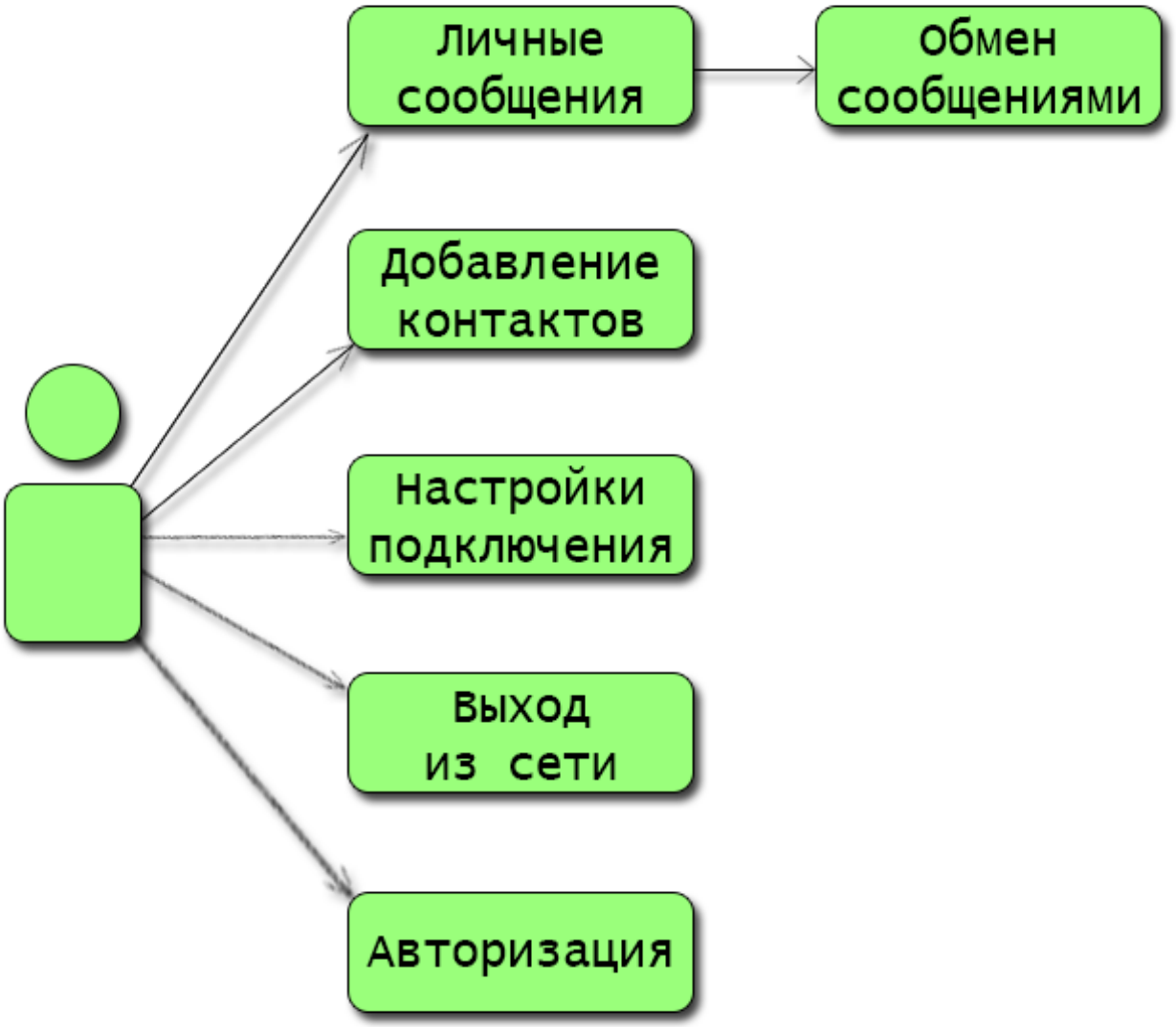


Рисунок 3 – UML диаграмма вариантов использования приложения пользователем

4.4 Схема взаимодействия клиента и сервера при обмене сообщениями

Рассмотреть процесс обмены сообщениями в клиент-серверном приложении можно наглядно благодаря схеме взаимодействия на рисунке 4. Она состоит из взаимодействия клиентской и серверной стороны.

Процесс начинается с того, что пользователю необходимо ввести контактные данные для входа в систему. После успешной идентификации

сотрудника, ему необходимо выбрать диалог с потенциальным собеседником. При выборе диалога, на сервер отправится команда о запросе содержимого выбранной беседы. В момент получения ответа от сервера откроется соответствующее окно, в котором отобразятся все существующие сообщения с данным сотрудником.

Диаграмма взаимодействия клиента и сервера 4.

Рисунок 4 – UML диаграмма взаимодействия клиента и сервера

5 МЕТОДЫ ИСПЫТАНИЙ

5.1 Добавление нового сотрудника

После выполнения работы, необходимо удостовериться в работоспособности программы, для чего необходимо провести действия по добавлению и удалению персонала, осуществить авторизацию и, в дальнейшем, осуществить обмен сообщениями.

Для проверки работы регистрации необходимо выполнить следующие действия:

- Запустить серверное приложение, рисунок 5;

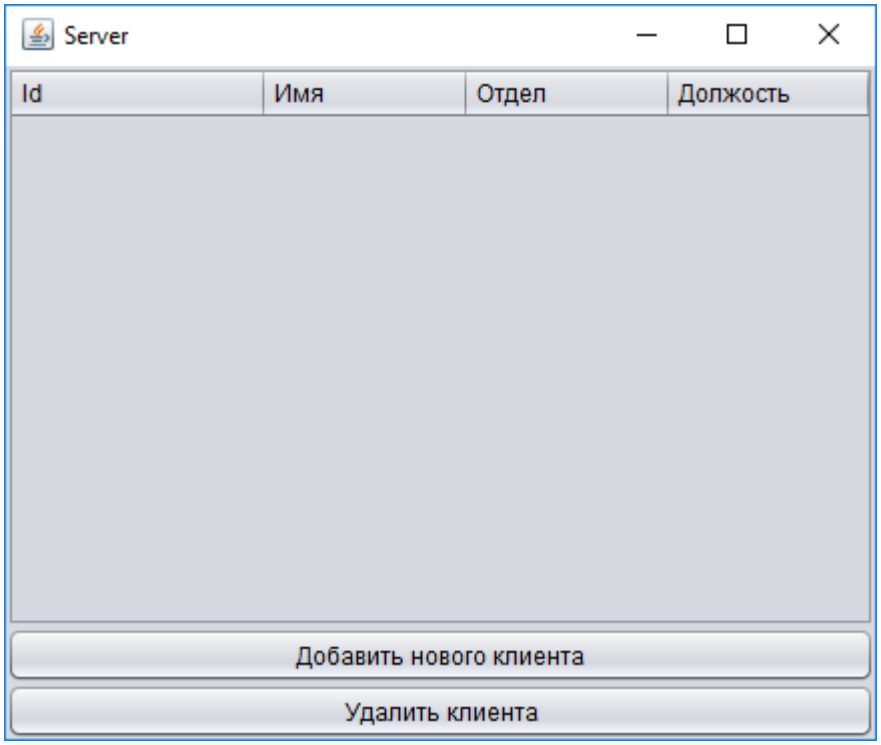


Рисунок 5 – Окно сервера

- Нажать на кнопку «Добавить нового клиента», после чего откроется окно добавления работника, данное окно представлено на рисунке 6;

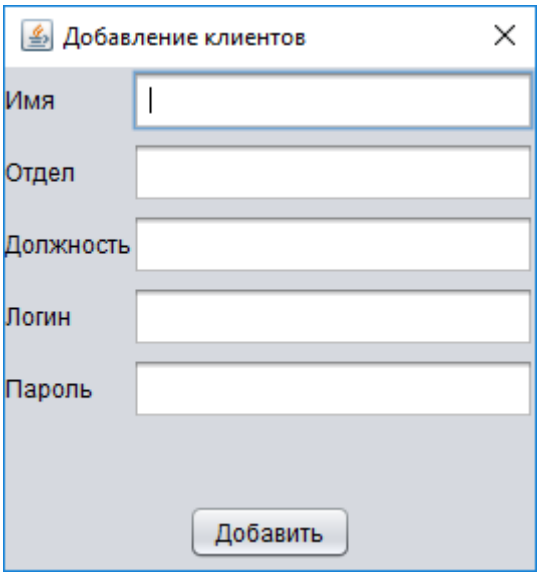


Рисунок 6 – Окно регистрации персонала

- Заполнить все необходимые поля информацией о новом сотруднике и нажать кнопку «Добавить», при соответствии введённой информации с условиями, произойдёт добавление нового члена сети по обмену сообщениями. Результат отображён на рисунке 7.

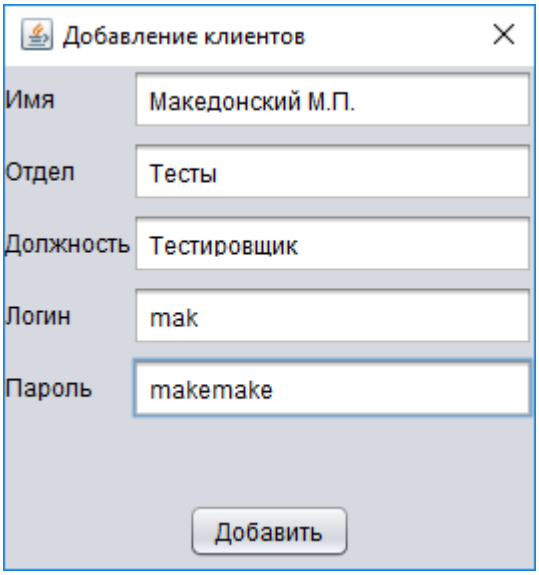


Рисунок 7 – Ввод перечня информации о новом сотруднике

5.2 Удаление сотрудника

В процессе удаления представителя персонала происходит удаление всей информации о выбранном сотруднике. Весь процесс протекает на серверной стороне и выглядит следующим образом:

- Перед удалением необходимо открыть соответствующее окно. Перед нами отобразится список всех зарегистрированных в сети сотрудников. В данном окне администратор сервера имеет доступ к перечню информации каждого сотрудника, рисунок 8;

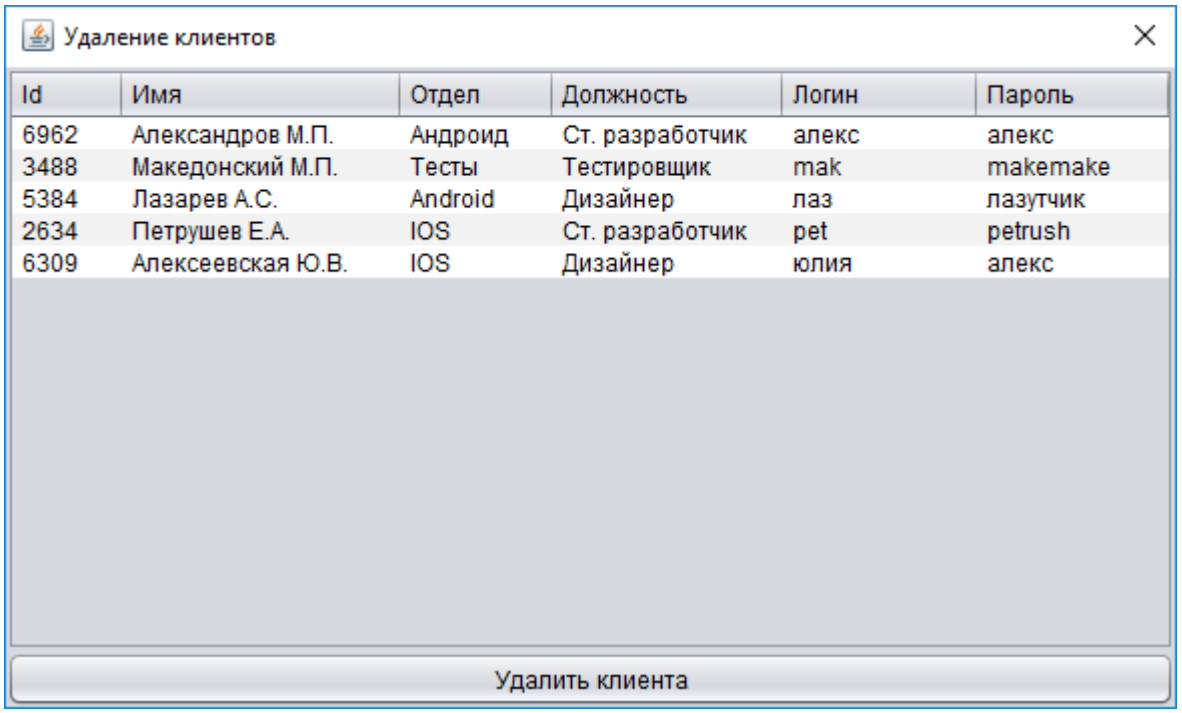


Рисунок 8 – Окно удаления персонала

- Для удаления требуется выбрать соответствующего сотрудника. После выбора можно осуществить удаление выбранного работника из системы, рисунок 9 и 10.

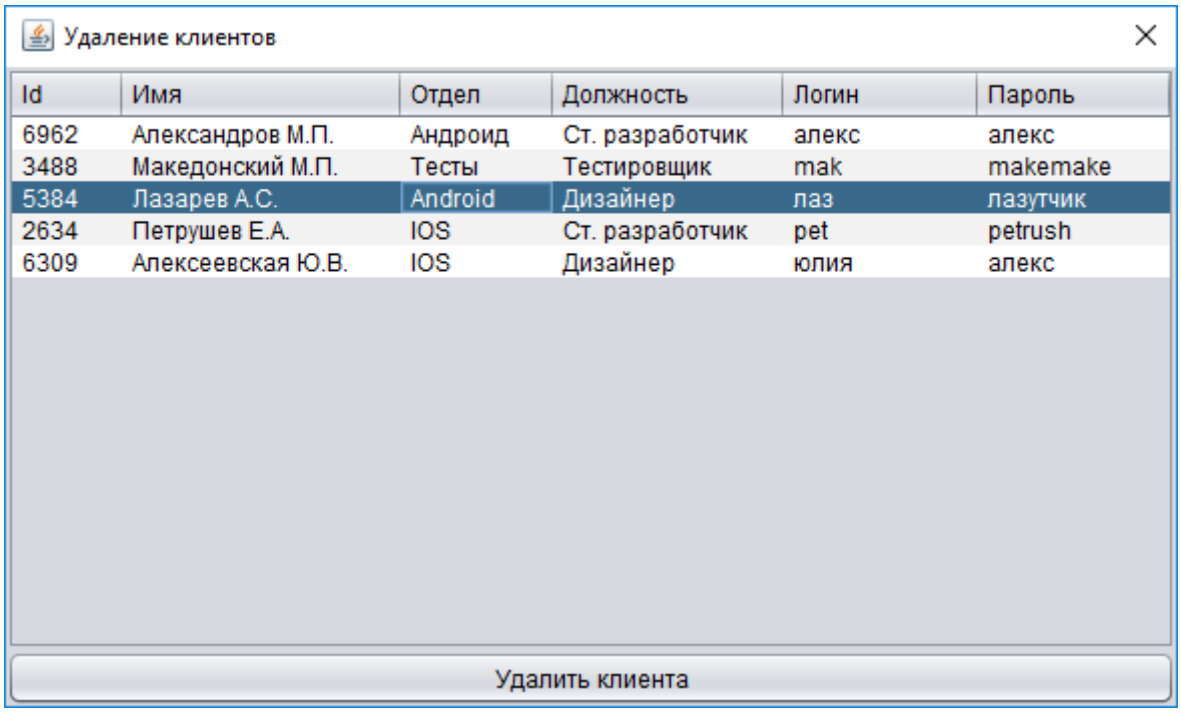


Рисунок 9 – Выбор работника

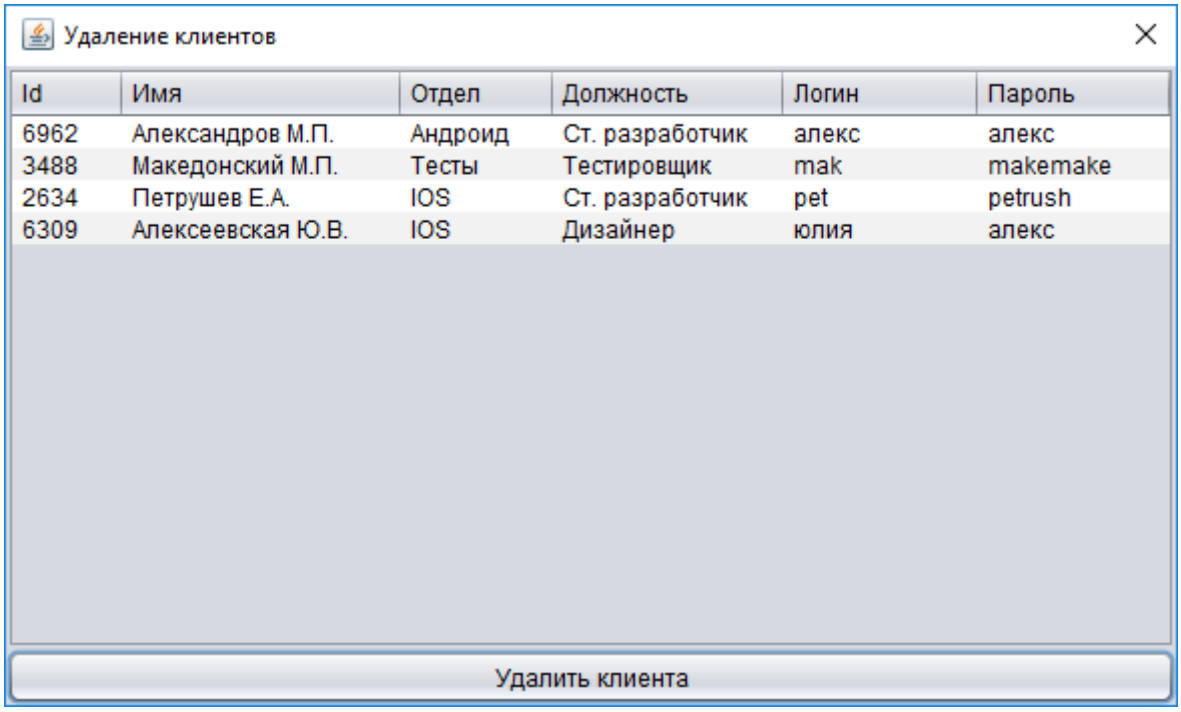


Рисунок 10 – Удаление сотрудника

5.3 Авторизация

Перед идентификацией работника в системе требуется развернуть серверное и клиентское приложение. Следующим шагом после запуска является ввод данных для авторизации на клиентской стороне. У каждого пользователя системы имеется логин и пароль. Данные для входа выдаются при добавлении сотрудника в систему.

Перед процессом авторизации сотруднику необходимо указать его логин и пароль в соответствующих графах главного окна клиентского приложения, рисунок 11.

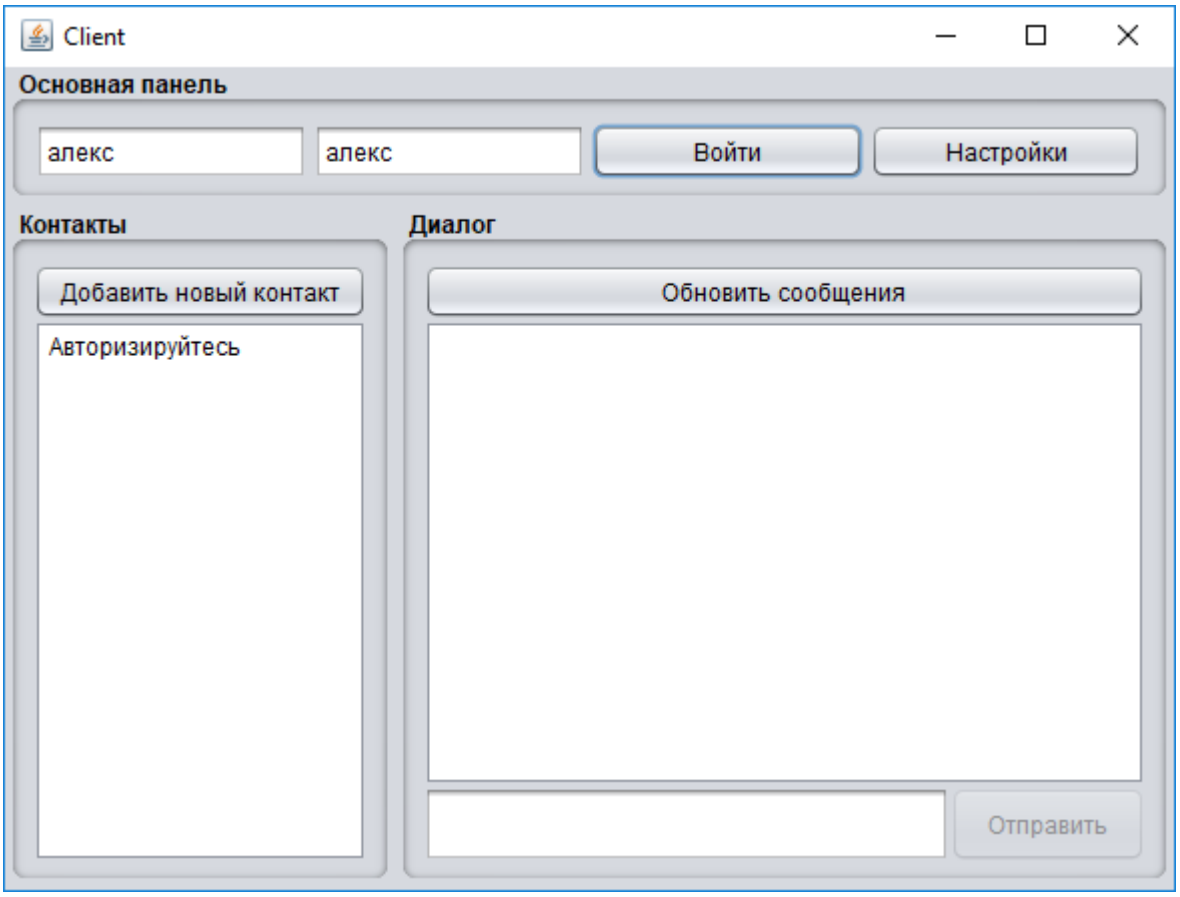


Рисунок 11 – Программа клиента

После нажатия кнопки «Вход» проверяется корректность введённых данных. На серверную сторону поступает команда о проверке введённых данных включающих логин и пароль.

При удачной идентификации пользователя на серверной стороне, он оповещает клиентское приложение об удачном входе и предоставляет права по использованию программы вошедшему сотруднику, результат отображён на рисунке 12.

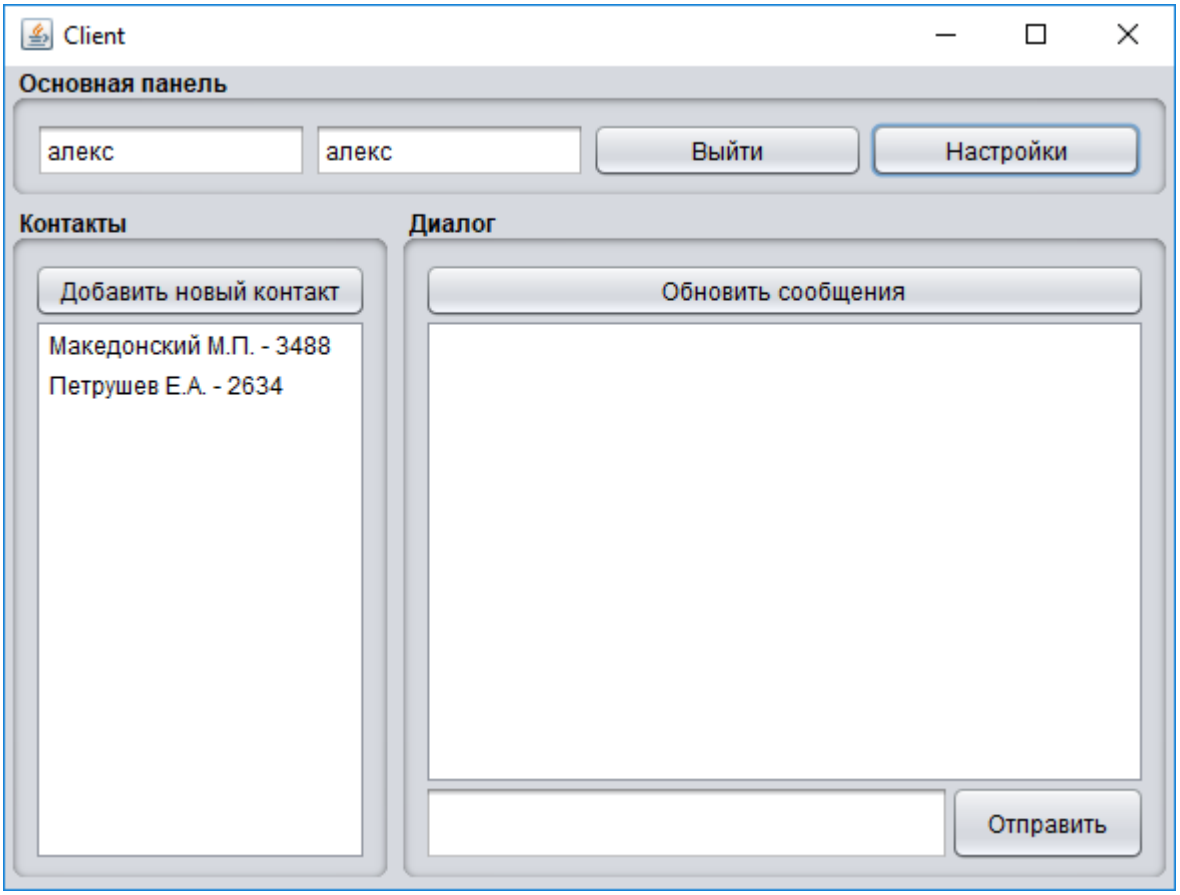


Рисунок 12 –Идентификация пользователя

При успешной авторизации серверное приложение получает информацию о вошедшем с этого аккаунта сотруднике, которому предоставляется возможность по эксплуатации системы. Отображение идентифицированного сотрудника в сети отображает рисунок 13.

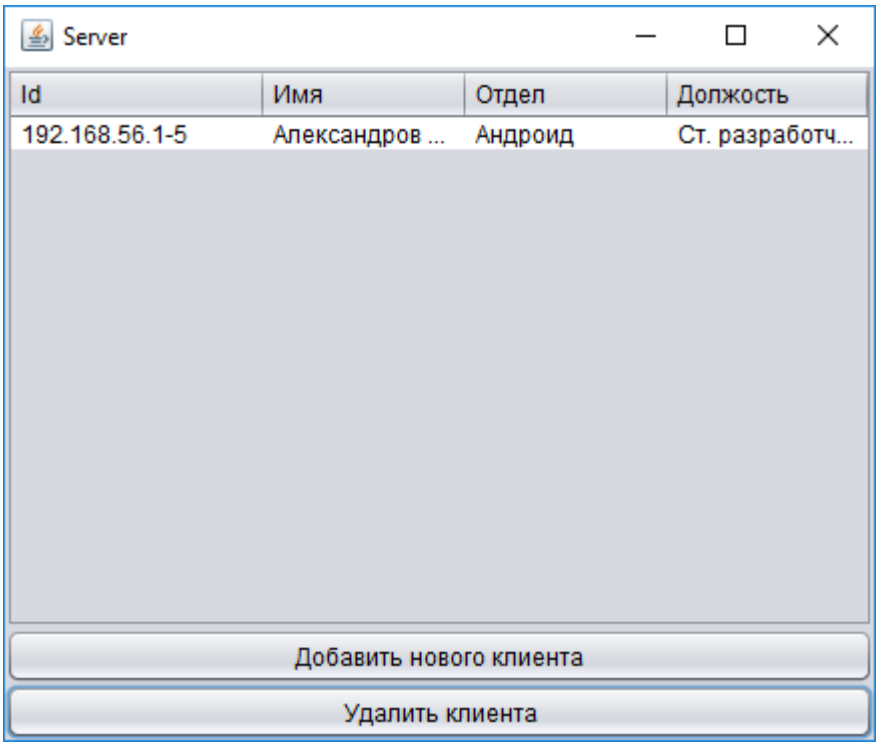


Рисунок 13 – Отображение сотрудника в сети

5.4 Выход из системы

Для корректного исполнения клиентского приложения, сотрудники имеют возможность производить отключение от серверной машины.

Данный функционал позволяет не загружать серверное приложение теми сотрудниками, чьи рабочие часы уже вышли. Также данное программное решение будет полезно для сотрудников, ведь всегда имеется возможность разорвать текущее подключение, не закрывая саму программу.

Демонстрация описанного функционала отображена на рисунке 14.

В любое время текущий сотрудник имеет возможность отключиться от сервера и прекратить работу в системе. При разрыве подключения сервер получает запрос об окончании сессии от указанного сотрудника. Затем он выполняет перечень функций по безопасному разъединению связи с клиентом, после этого клиентское приложение получает оповещение об удачном разрыве связи, рисунок 15.

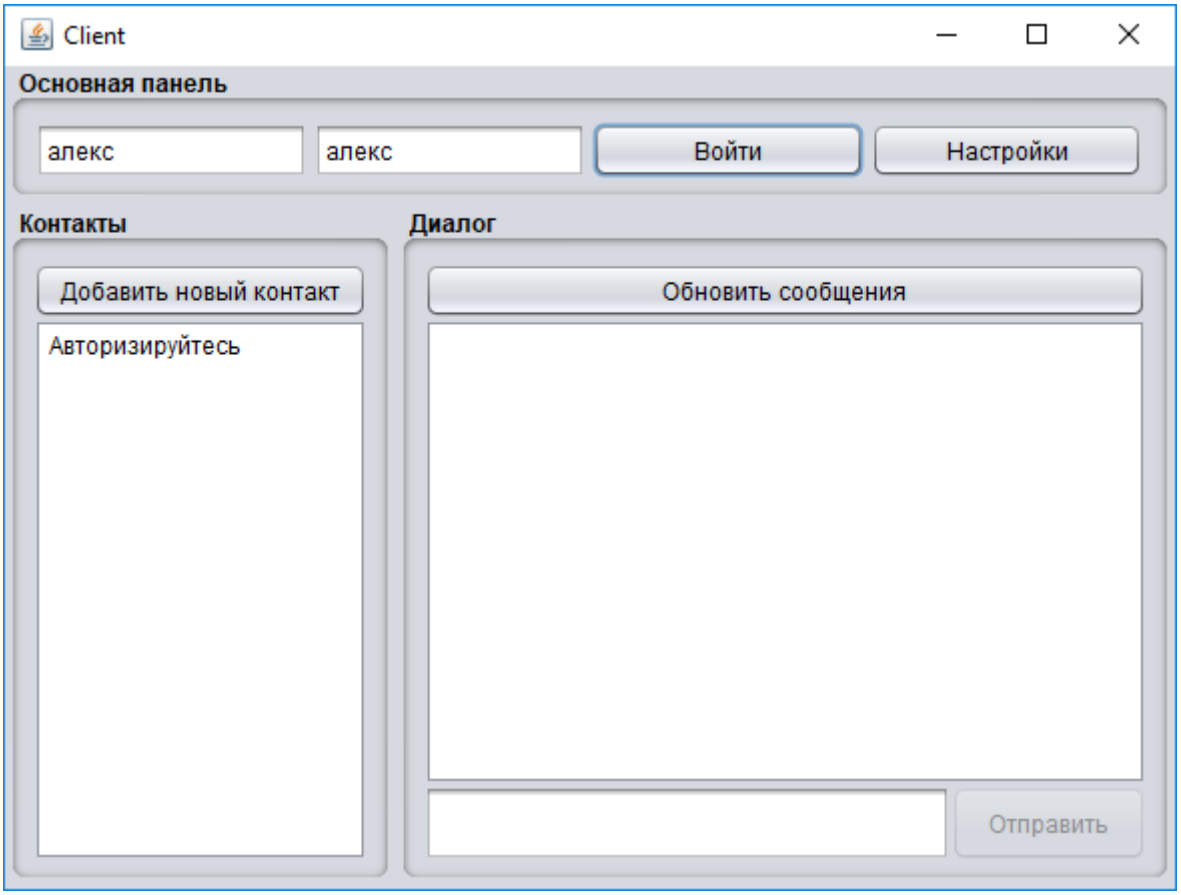


Рисунок 14 – Завершение подключения

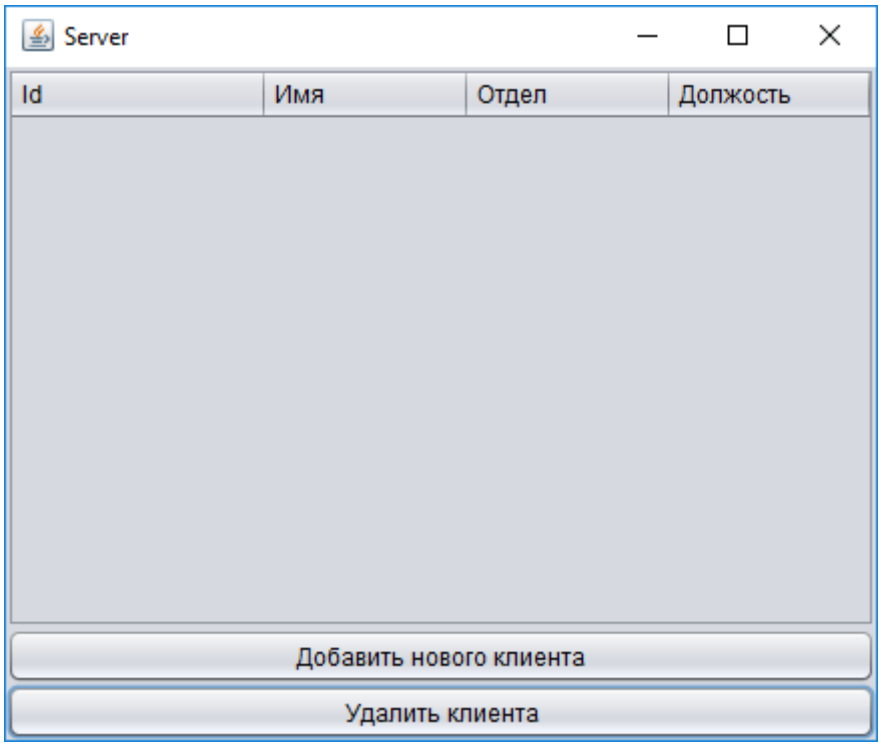


Рисунок 15 – Выход сотрудника из сети

5.5 Изменение настроек подключения

Любая информационная система с использованием архитектуры клиент-сервер подвержена воздействию технических трудностей и проблем по развёртыванию. Для гибкости и универсальности разработанного приложения было принято решение по реализации функционала для смены настроек подключения. В перечень данных для установления связи входят хост и порт. Данные параметры должны быть зарезервированны в локальной сети для развёртывания сервера с указанными параметрами.

Перед изменением настроек соединения необходимо узнать под каким IP-адресом функционирует серверное приложение, рисунок 16.

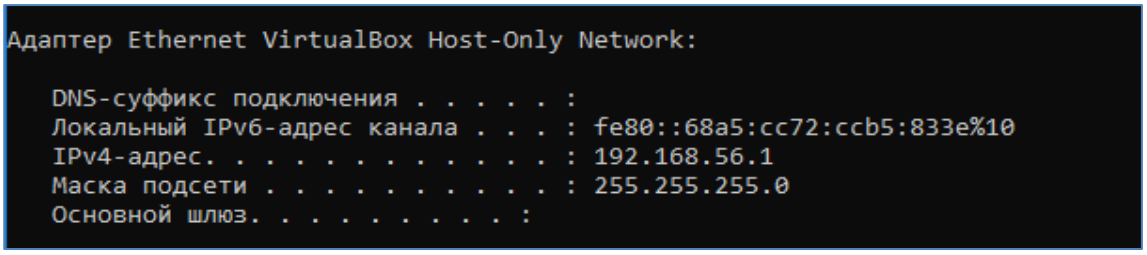


Рисунок 16 – Просмотр значения IPv4-адреса на серверной машине

В конфигурационном файле клиентского приложения хранится аналогичный адрес подключения, что и адрес на рисунке 17. Из этого следует, что ранее настроенное подключение является исправным.

Отображение содержимого файла настроек клиентского приложения отображено на рисунке 17.

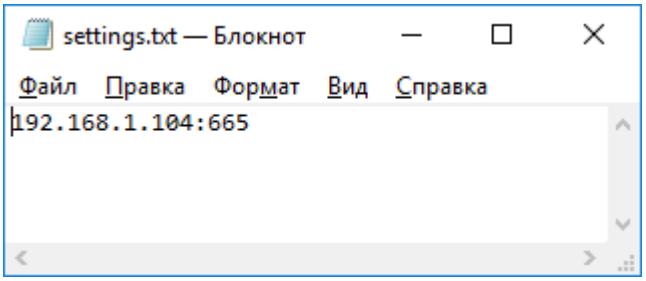


Рисунок 17 – Содержимое конфигурационного файла клиента

Для изменения данных для подключения необходимо нажать кнопку «Настройки». Ввести значения хоста и порта в соответствующие поля ввода открывшегося окна и подтвердить ввод нажатием на кнопку «Сохранить», рисунок 18.

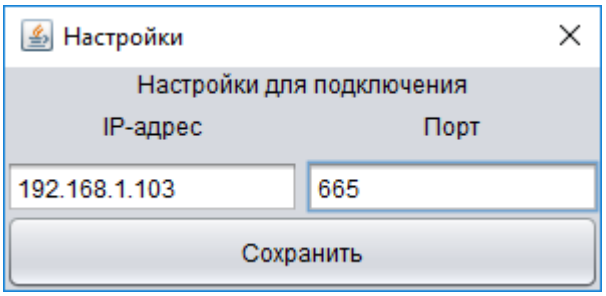


Рисунок 18 – Изменение конфигурации соединения

5.6 Добавление контактов

В подавляющем большинстве программных продуктах схожего назначения реализована система хранения контактов или иной быстрый доступ к диалогу с конкретным собеседником. Разрабатываемое приложение «Communica» не является исключением.

Перед добавлением контактов, пользователю необходимо идентифицировать свою личность в системе. Следующем шагом является непосредственное добавление новых собеседников, для этого необходимо в клиентском приложении нажать кнопку «Добавить новый контакт», рисунок 19.

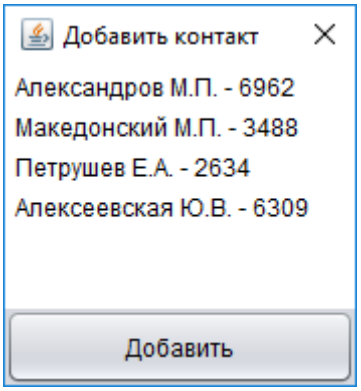


Рисунок 19 – Список сотрудников для добавления

Для осуществления добавления контакта, необходимо выбрать потенциального собеседника и нажать кнопку «Добавить», рисунок 20.

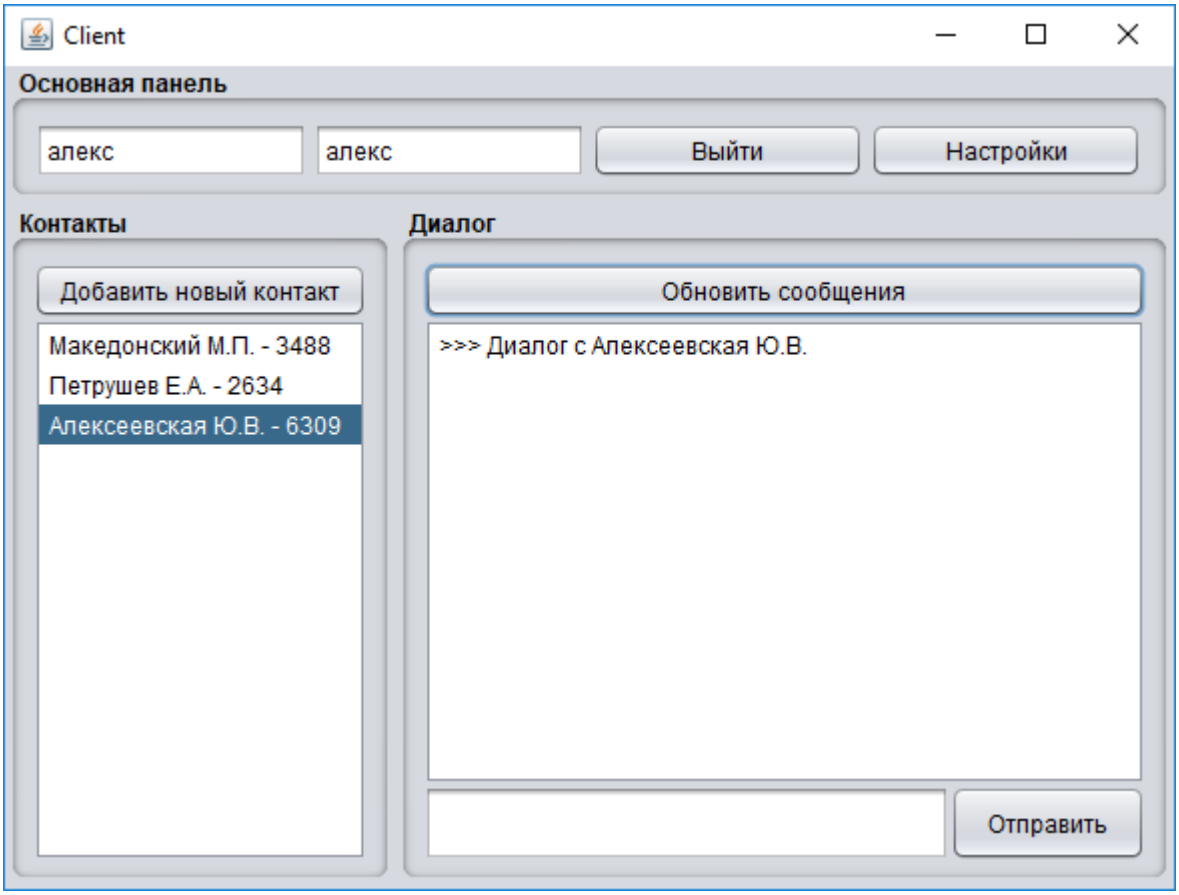


Рисунок 20 – Результат добавления нового контакта

5.7 Просмотр сообщений

Следующим испытываемым функционалом следует возможность просмотра сообщений. Все сообщения, когда-либо отправленные пользователями, хранятся на серверной стороне в виде отдельных файлов.

Файлы диалогов индексируются в соответствии с идентификаторами пользователей беседы. Таким образом, при разработке, получаем гибкую и распределённую систему по хранению и чтению существующих данных в системе.

Перед отображением списка сообщений, с клиентского приложения отправляется соответствующий запрос на сервер. Ответом на данный запрос является список сообщений в выбранной беседе, рисунок 21.

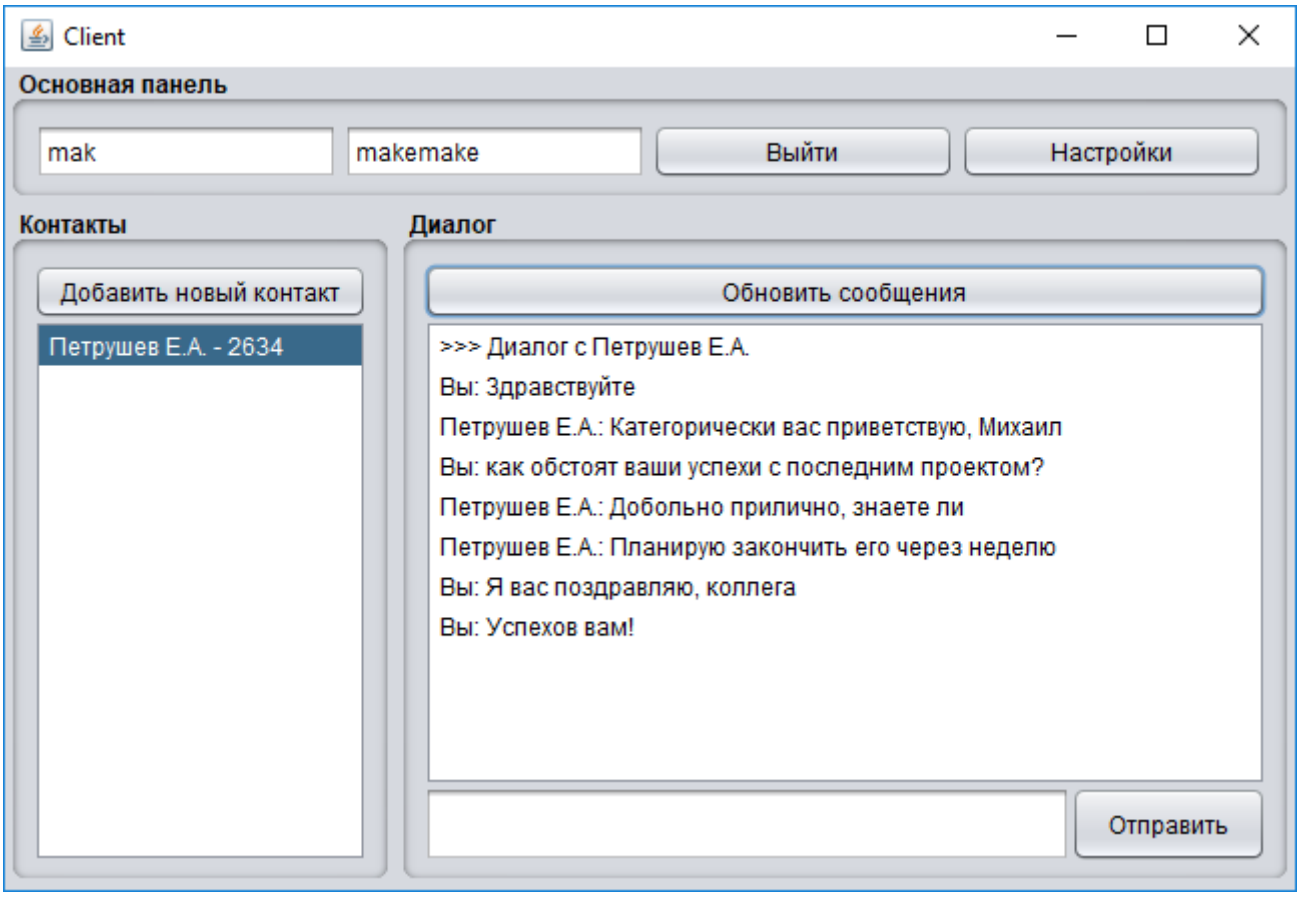


Рисунок 21 – Содержимое диалога между двумя сотрудниками

5.8 Обмен сообщениями

Реализация обмена сообщениями заняла значительную часть разработки от всего приложения. Строк кода, приходящихся на исполнение функций по обмену сообщениями, гораздо больше в сравнении с массивом кода для исполнения каких-либо иных функций в приложении. Исходя из этого, требуется провести подробное испытание данной программной реализации.

На рисунке 22 - 23 отображено тестирование возможности обмена строковыми сообщениями между клиентами. Сотрудник в качестве отправителя изображён на рисунке 22. Получатель письма демонстрируется на рисунке 23. После получения письма требуется обновить сообщения при помощи функционала заложенного в кнопке «Обновить сообщения».

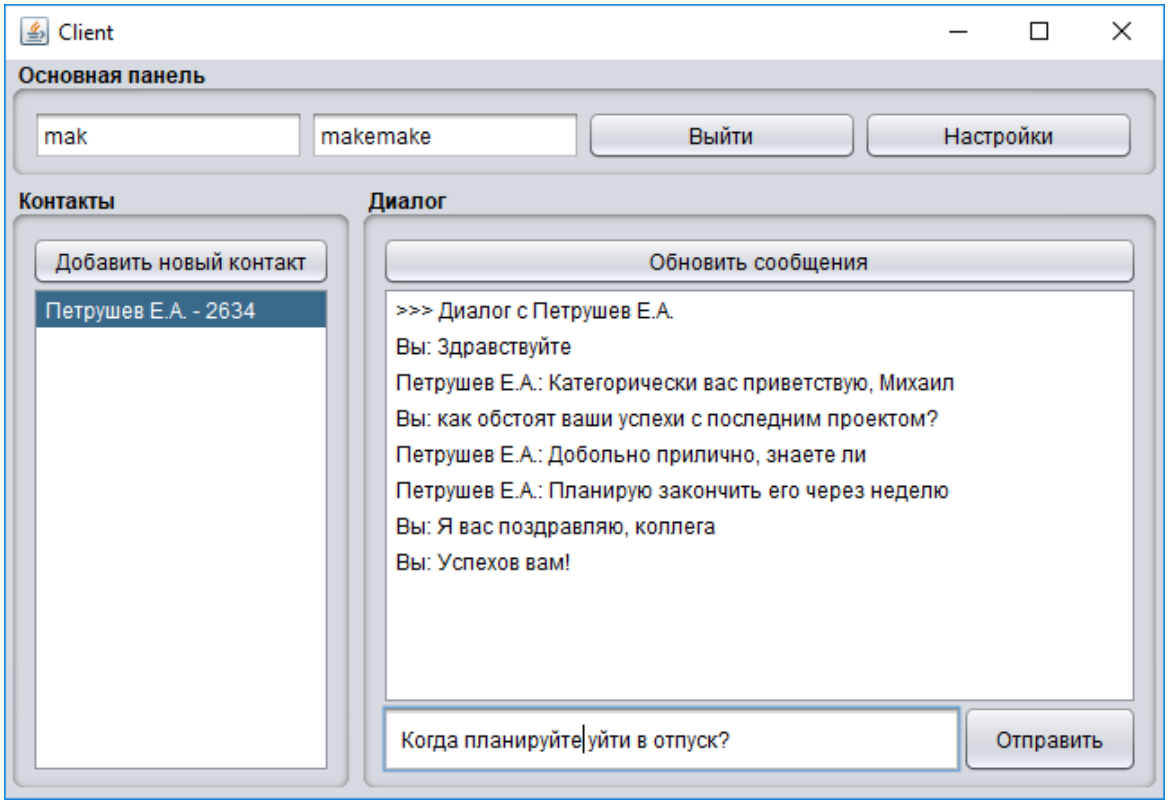


Рисунок 22 – Окно отправителя сообщения

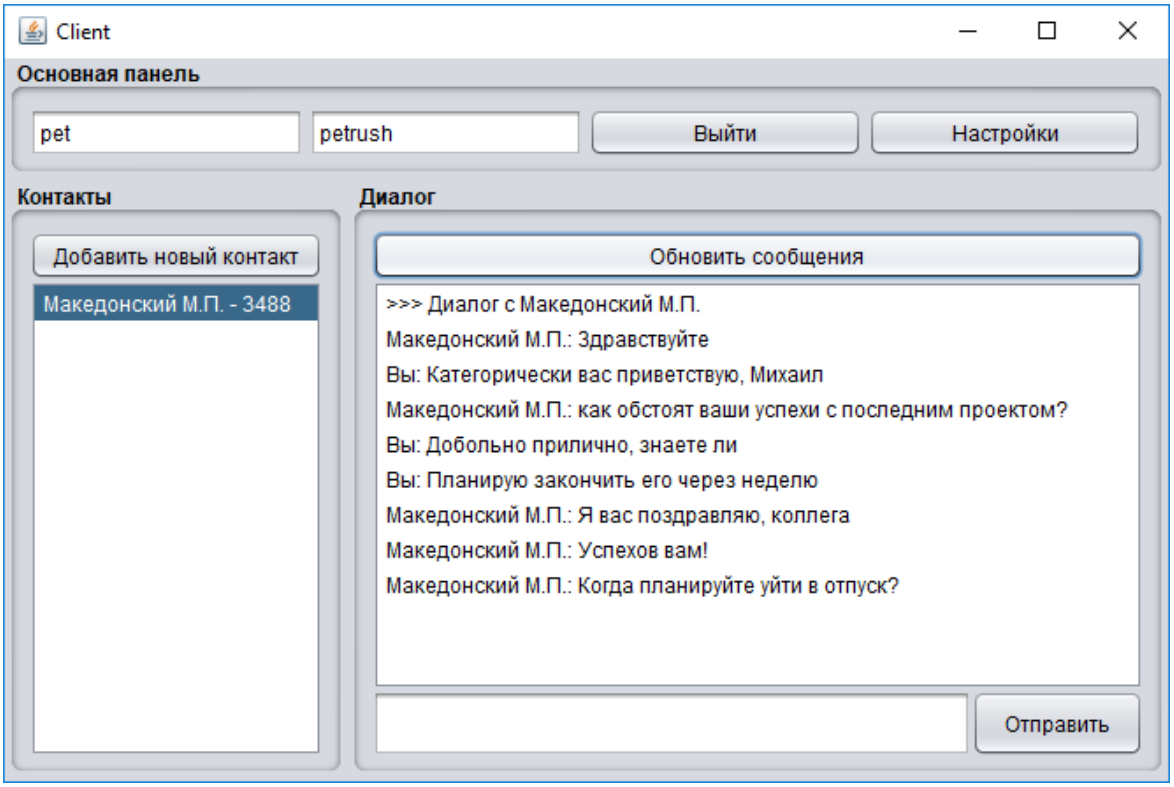


Рисунок 23 – Окно получателя сообщения

6 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

В последние годы большое внимание уделяется улучшению условий труда пользователей электронно-вычислительных машин и видеодисплейных терминалов, несмотря на то, что качество и безопасность самых ПЭВМ и ВДТ постоянно улучшаются. В развитых странах, в том числе в Германии, Швеции, США, вопрос об опасности работы за дисплеями поднялся до уровня национальной проблемы, а в Германии работа за дисплеями входит в список сорока наиболее опасных и вредных профессий.

Главным фактором, влияющим на производительность труда людей, работающих с электронно-вычислительными машинами и видеодисплейных терминалов, являются комфортные и безопасные условия их труда.

Условия труда пользователя, работающего с персональным компьютером, определяются:

- особенностями организации рабочего места;
- условиями производственной среды (освещением, микроклиматом, шумом, электромагнитными полями, визуальными эргономическими параметрами дисплея и т. д.);
- характеристиками информационного взаимодействия человека и персональных электронно-вычислительных машин.

При выполнении работ на персональном компьютере согласно ГОСТу 12.0.003-74 могут иметь место следующие факторы:

- повышенная температура поверхностей ПК;
- повышенная или пониженная температура воздуха рабочей зоны;
- выделение в воздух рабочей зоны ряда химических веществ;
- повышенная или пониженная влажность воздуха;
- повышенный или пониженный уровень аэроионов;
- отсутствие или недостаток естественного света;
- недостаточная искусственная освещенность рабочей зоны;

Повышенные статические и динамические нагрузки на организм у пользователей ПК приводят к симптомам похожим на боли в спине, шейном отделе позвоночника и руках. Из всего перечня недугов, обусловленных работой на компьютере, чаще встречаются те, которые непосредственно связаны с использованием клавиатуры. В период выполнения операций ввода значений количество незначительных однотипных манипуляций кистью и пальцами рук за смену может превысить 60 тысяч, что в соответствии с классификацией труда относится к категории вредных и опасных. Поскольку каждое нажатие на клавишу связано с сокращением мышц, сухожилия непрерывно скользят вдоль костей и соприкасаются с соединительными материалами, вследствие чего развиваются болезненные воспалительные процессы. Воспалительные процессы тканей сухожилий получили общее название «травма повторяющихся нагрузок».

Большинство работающих рано или поздно начинают предъявлять жалобы на боли в шее и спине. Эти недомогания накапливаются постепенно и получили название «синдром длительных статических нагрузок».

Другой причиной возникновения СДСН может быть длительное пребывание в положении «сидя», которое приводит к сильному перенапряжению мышц спины и ног, в результате чего возникают боли и неприятные ощущения в нижней части спины. Основной причиной перенапряжения мышц спины и ног являются нерациональная высота рабочей поверхности стола и сидения, отсутствие опорного механизма и опоры для локтей, непрвильное размещение монитора, клавиатуры и документов, отсутствие подставки для ног.

– Для симптомного уменьшения неприятных ощущений или боли, возникающих у пользователей ПК, необходимы частые перерывы в работе и эргономические усовершенствования, в том числе оборудование рабочего места так, чтобыустранять неудобные позы и длительные напряжения.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной бакалаврской работы был разработан алгоритм решения задачи, создана и отложена программа, оформлена документация. Проведенные испытания показали, что клиент-серверное приложение для обмена данными в локальной сети соответствует всем поставленным требованиям и выполняет свою работу корректно.

За время работы с данным проектом было прочитано большое количество технической документации, изучены новые способы и механизмы программирования и проектирования систем. Значительно развит навык разработки программ на языке Java, администрирования баз данных.

Опыт, полученный в ходе выполнения этого задания, неоспоримо поможет в дальнейшем трудоустройстве и развитии собственных навыков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. «Java. Полное руководство», Герберт Шилдт.
2. «Head First Java, 2nd Edition» (Изучаем Java), Кэти Сьера.
3. «Java. Методы программирования», Блинов, Романчик.
4. «Introduction to Programming in Java: An Interdisciplinary Approach», Robert Sedgewick.
5. «Effective Java: Programming Language Guide» (Java. Эффективное программирование) - Joshua Bloch, Джошуа Блох.
6. «Java Swing: Эффектные пользовательские интерфейсы» - Издание второе (2011).
7. «Программирование на SQL. Учебное пособие для СПО», Маркин А. В.
8. «Язык SQL в примерах и задачах», Толстобров А. П., Фертиков В. В.
9. Интернет ресурс «https://proselyte.net/tutorials/jdbc/introduction/»
10. Интернет ресурс «http://java-online.ru/jdbc.xhtml»

ПРИЛОЖЕНИЕ А

(Графическая часть)

ПРИЛОЖЕНИЕ Б

(UML диаграммы классов)

UML диаграмма классов приложения «CommunicaServer»

Разработанное серверное приложение несёт максимальную ответственность за выполнение всех поступающих команд от множества клиентских приложений. Для этого была спроектирована и реализована архитектура, построенная на выдаче персональных потоков каждому из идентифицированных пользователей.

Реализацию данной архитектуры можно наблюдать на рисунке 24.

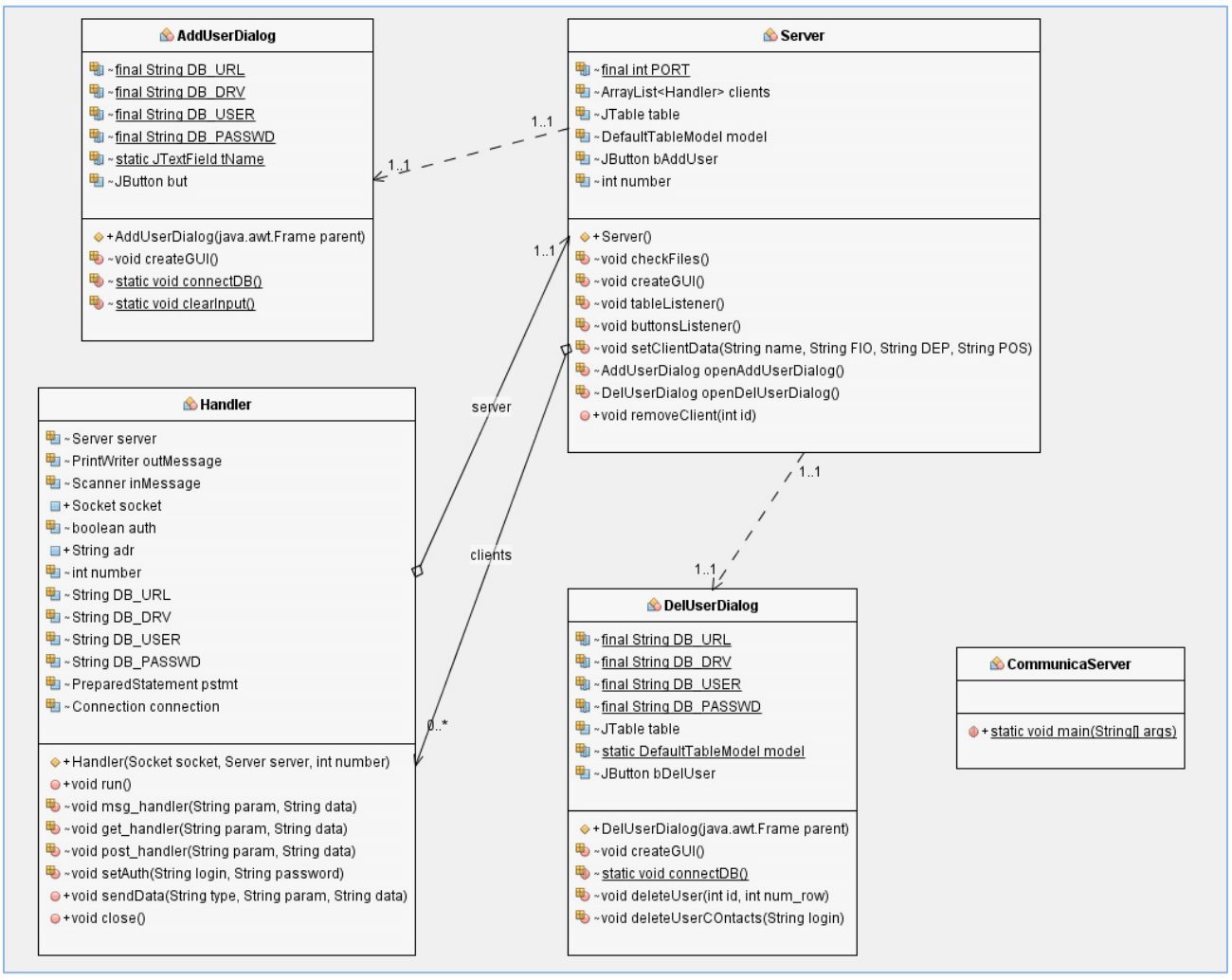


Рисунок 24 – UML диаграмма классов серверного приложения

UML диаграмма классов приложения «CommunicaClient»

Для корректного и исправного подключения к серверной машине требуется масштабный перечень переменных и большое количество кода. Для этого главный класс клиентской программы «CLientWindow» был разбит на множество функций и методов. Так же были созданы вспомогательные классы для реализации отдельного функционала.

На рисунке 25 отображён список всех классов, методов и необходимых переменных для построения работоспособного и функционального подключения к серверной программе.

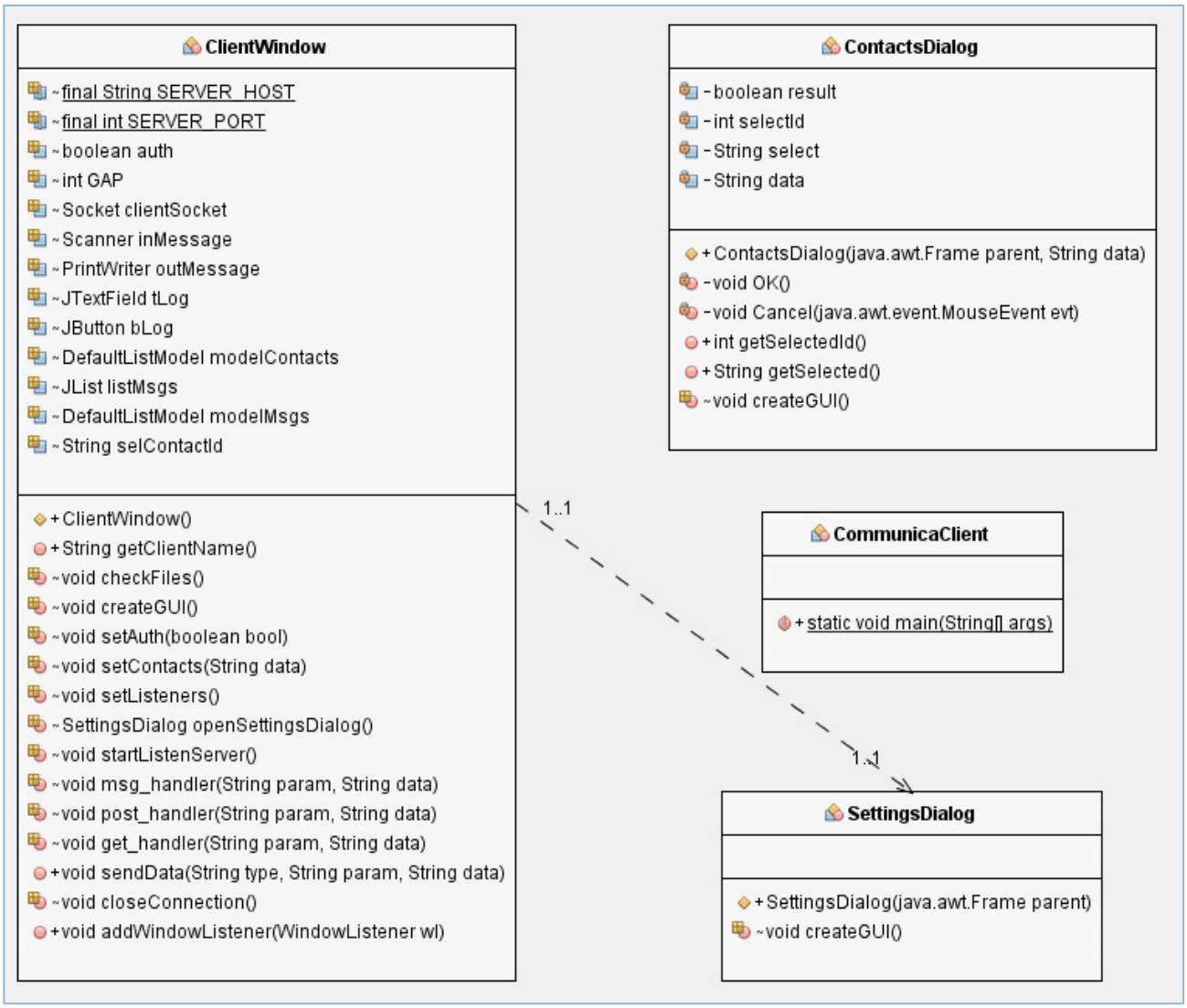


Рисунок 25 – UML диаграмма классов клиентского приложения

ПРИЛОЖЕНИЕ В

(Листинги программы)

Листинг серверного приложения «CommunicaServer»

Файл Server.java

```
package communicaserver;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.WindowConstants;
import javax.swing.table.DefaultTableModel;

public class Server extends JFrame {
    static final int PORT = 665;
    ArrayList<Handler> clients = new ArrayList<Handler>();
    JTable table;
    DefaultTableModel model;
    JButton bAddUser, bDelUser;
    int number = 0;

    public Server() {
        Socket clientSocket = null;
        ServerSocket serverSocket = null;

        try {
            serverSocket = new ServerSocket(PORT);
            System.out.println("### Сервер запущен! ###");
            checkFiles();
            createGUI();
            setVisible(true);
            tableListener();
            buttonsListener();

            while (true) {
                clientSocket = serverSocket.accept();
                Handler clientHandler =
                            new Handler(clientSocket, this,
number);
                clients.add(clientHandler);
                number++;
                new Thread(clientHandler).start();
            }
```

```
    } catch (IOException ex) {
                ex.printStackTrace();
            } finally {
                try {
                    clientSocket.close();
                    System.out.println("Сервер остановлен");
                    serverSocket.close();
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
        }
        void checkFiles() {
            try {
                if (!Files.exists(Paths.get("data"))) {
                    File dir = new File("data");
                    dir.mkdir();
                    System.out.println("Директория data создана");
                }
                if (!Files.exists(Paths.get("data/contacts.txt"))) {
                    new File("data/contacts.txt").createNewFile();
                    System.out.println("Файл contacts.txt создан");
                }
                if (!Files.exists(Paths.get("data/settings.txt"))) {
                    new File("data/settings.txt").createNewFile();
                    System.out.println("Файл settings.txt создан");
                }
            } catch(IOException ex) {

            }
        }

        final void createGUI() {
            // Задаём настройки элементов на форме
            setBounds(50, 115, 450, 375);
            setTitle("Server");
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

            // верхняя панель
            JPanel topPanel = new JPanel(new GridLayout(1,5));
            add(topPanel, BorderLayout.NORTH);

            // центральная панель
            String[] columnNames = {"Id", "Имя", "Отдел",
"Должость"};
            model = new DefaultTableModel(0, columnNames.length);
            model.setColumnIdentifiers(columnNames);
            table = new JTable(model) {
                @Override
                public boolean isCellEditable(int arg0, int arg1) {
                    return false;
                }
            };

table.getColumnModel().getColumn(0).setPreferredWidth(100);
```

```
            add(new JScrollPane(table), BorderLayout.CENTER);

            // нижняя панель
            JPanel bottomPanel = new JPanel(new GridLayout(2,1));
            add(bottomPanel, BorderLayout.SOUTH);
            bAddUser = new JButton("Добавить нового клиента");
            bottomPanel.add(bAddUser);
            bDelUser = new JButton("Удалить клиента");
            bottomPanel.add(bDelUser);
        }

        void tableListener() {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    int count = 0;
                    while (true) {
                        try {
                            if (count != clients.size()) {
                                count = clients.size();
                                model.setRowCount(0);
                                for (int i = 0; i < clients.size();
i++) {
                                    model.addRow(new Object[] {
clients.get(i).socket.getLocalAddress().toString()
                                            .substring(1) + "-" +
clients.get(i).number,
                                    clients.get(i).name,
                                    clients.get(i).dep,
                                    clients.get(i).pos});
                                }
                            }
                            model.fireTableDataChanged();
                            Thread.sleep(100);
                        } catch (InterruptedException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null,
ex);
                        }
                    }
                }
            }).start();
        }
        void buttonsListener() {
            bAddUser.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    AddUserDialog dialog = openAddUserDialog();
                    dialog.setVisible(true);
                }
            });
            bDelUser.addActionListener(new ActionListener() {
                @Override
```

```
    public void actionPerformed(ActionEvent e) {
                    DelUserDialog dialog = openDelUserDialog();
                    dialog.setVisible(true);
                }
            });
        }

        void setClientData(String name, String FIO, String DEP,
String POS) {
            try {
                // Задержка потока для добавления строки в таблицу о
новом потоке
                Thread.sleep(600);
            } catch (InterruptedException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null,
ex);
            }
            System.out.println("New client data from:" + name);
            for (int i = 0; i < model.getRowCount(); i++) {
                if (model.getValueAt(i, 0).toString().equals(name))
{
                    model.setValueAt(FIO, i, 1);
                    model.setValueAt(DEP, i, 2);
                    model.setValueAt(POS, i, 3);
                }
            }
        }
            AddUserDialog openAddUserDialog() {
            AddUserDialog dialog = new AddUserDialog(this);
            return dialog;
        }
        DelUserDialog openDelUserDialog() {
            DelUserDialog dialog = new DelUserDialog(this);
            return dialog;
        }
        public void removeClient(int id) {
            clients.remove(id);
        }
    }
```

Файл Handler.java

```
    package communicaserver;
    import java.io.File;
    import java.io.FileInputStream;
    import java.io.IOException;
    import java.io.PrintWriter;
    import java.net.Socket;
    import java.nio.file.Files;
    import java.nio.file.Paths;
    import java.nio.file.StandardOpenOption;
    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.PreparedStatement;
    import java.sql.ResultSet;
```

```
    import java.sql.SQLException;
    import java.sql.Statement;
    import java.util.Properties;
    import java.util.Scanner;

    public class Handler implements Runnable {
        Server server;                  // экземпляр нашего сервера
        PrintWriter outMessage;         // исходящее сообщение
        Scanner inMessage;              // входящее собщение
        public Socket socket = null;
        boolean auth = false;
        public String adr, name, dep, pos;
        int number;
        // SQL
        String DB_URL = "jdbc:mysql://localhost:3306/data";
        String DB_DRV = "com.mysql.jdbc.Driver";
        String DB_USER = "root";
        String DB_PASSWD = "root";
        PreparedStatement pstmt = null;
        Connection connection = null;
        public Handler(Socket socket, Server server, int number) {
            try {
                this.server = server;
                this.socket = socket;
                this.outMessage = new
PrintWriter(socket.getOutputStream());
                this.inMessage = new
Scanner(socket.getInputStream());
                this.adr =
socket.getLocalAddress().toString().substring(1) + "-" + number;
                this.number = number;
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
        @Override
        public void run() {
            try {
                System.out.println("Присоединился " + adr);
                sendData("get", "auth", "-");
                try {
                    if (!Files.exists(Paths.get("messages"))) {
                        File dir = new File("messages");
                        dir.mkdir();
                        System.out.println("Директория messages
создана");
                    }
                } catch(Exception ex) {
                    ex.printStackTrace();
                }
                while (true) {
                 if (inMessage.hasNext()) {
```

```
    String[] msg = inMessage.nextLine().split("#");

                        if (msg[0].equals("exit")) {
                            System.out.println("Покинул сеть " +
adr);
                            break;
                        } else {
                            switch(msg[0]) {
                                case "msg":
                                    msg_handler(msg[1], msg[2]);
                                    break;
                                case "get":
                                    get_handler(msg[1], msg[2]);
                                    break;
                                case "post":
                                    post_handler(msg[1], msg[2]);
                                    break;
                                default:
System.out.println(adr + ": " + msg[0] + msg[1] + msg[2]);
                                    break;
                            }
                        }
                    }
                }
            } catch (NumberFormatException ex) {
                //JOptionPane.showMessageDialog(null, "Неверный
формат введённых значений");
            } finally {
                this.close();
            }
        }
        void msg_handler(String param, String data) {
            try {
                String[] val = param.split("&");
                connection =
DriverManager.getConnection(DB_URL,DB_USER,DB_PASSWD);
                String ID = "SELECT id FROM users WHERE login='" +
val[1] + "'";
                pstmt = connection.prepareStatement(ID,
Statement.RETURN_GENERATED_KEYS);
                ResultSet rs = pstmt.executeQuery(ID);
                if (rs.next()) {
                    int senderId = Integer.valueOf(rs.getString(1));
                    int resipId = Integer.valueOf(val[0]);
                    StringBuilder dialogId = new StringBuilder("");

                    if (senderId > resipId) {
                        dialogId.append(resipId).append(senderId);
                    } else {
                        dialogId.append(senderId).append(resipId);
                    }
                 dialogId.append(".txt");
```

```
    //System.out.println("ЗАПРОС для диалога: " + dialogId);

                    // Проверка на существование данного диалога
(файла диалога)
                    if (!Files.exists(Paths.get("messages/" +
dialogId))) {
                        new File("messages/" +
dialogId).createNewFile();
                        System.out.println("СОЗДАНИЕ диалога " +
dialogId);
                        sendData("post", "list_msg", "none");
                    }

                    // Сохранение полученного сообщения
                    StringBuilder msg = new StringBuilder("");

msg.append(senderId).append(":").append(data).append(";");
                    Files.write(Paths.get("messages/" + dialogId),
                            msg.toString().getBytes(),
                            StandardOpenOption.APPEND);
                    System.out.println("ЗАПИСЬ сообщения в: " +
dialogId);
                                connection.close();
                pstmt.close();
            } catch(SQLException ex) {
                ex.printStackTrace();
            } catch(Exception ex) {
                ex.printStackTrace();
            }
        }
        void get_handler(String param, String data) {
            switch(param) {
                case "contacts":
                    try {
                        Properties prop = new Properties();
                        prop.setProperty("user", "root");
                        prop.setProperty("password", "root");
                        prop.setProperty("useUnicode", "true");
                        prop.setProperty("characterEncoding",
"cp1251");
                        connection =
DriverManager.getConnection(DB_URL, prop);
                        String DATA = "SELECT id,name FROM users";
                        pstmt = connection.prepareStatement(DATA,
Statement.RETURN_GENERATED_KEYS);
                        pstmt.execute(DATA);
                        final ResultSet rsData =
pstmt.getResultSet();

                        StringBuilder sb = new StringBuilder("");
                        while (rsData.next()) {
                            sb.append(rsData.getString(2)).append("
- ").append(rsData.getString(1)).append(";");
                        }
      sendData("post", "contacts", sb.toString());
```

```
                    connection.close();
                    pstmt.close();
                } catch(SQLException ex) {
                    System.out.println("ОШИБКА загрузки списка
контактов: " + ex.getLocalizedMessage());
                }
                break;
            case "client_contacts":
                try {
                    Properties prop = new Properties();
                    prop.setProperty("user", "root");
                    prop.setProperty("password", "root");
                    prop.setProperty("useUnicode", "true");
                    prop.setProperty("characterEncoding", "cp1251");
                    connection = DriverManager.getConnection(DB_URL,
prop);
                    String ID = "SELECT id FROM " + data;
                    pstmt = connection.prepareStatement(ID,
Statement.RETURN_GENERATED_KEYS);
                    pstmt.execute(ID);
                    final ResultSet rsId = pstmt.getResultSet();

                    rsId.next();
                    StringBuilder sbId = new StringBuilder("SELECT
id,name FROM users WHERE id IN("
                            + rsId.getString(1));
                    while (rsId.next()) {
                        sbId.append(",").append(rsId.getString(1));
                    }
                    sbId.append(")");

                    pstmt =
connection.prepareStatement(sbId.toString(),
Statement.RETURN_GENERATED_KEYS);
                    pstmt.execute(sbId.toString());
                    final ResultSet rsData = pstmt.getResultSet();

                    StringBuilder sb = new StringBuilder("");
                    while (rsData.next()) {
                        sb.append(rsData.getString(2)).append(" -
").append(rsData.getString(1)).append(";");
                    }
                    // Пересылаем персональный список контактов клиенту
                    sendData("post", "client_contacts", sb.toString());
                    connection.close();
                    pstmt.close();
                } catch(SQLException ex) {
                    System.out.println("ОШИБКА загрузки контактов: " +
ex.getLocalizedMessage());
                    sendData("post", "client_contacts", "none");
                }
                break;
```

```
case "list_msg":
                try {
                    // Определяем id диалога; val[0] - id получателя,
val[1] - логин отправителя
                    String[] val = data.split("&");
                    Properties prop = new Properties();
                    prop.setProperty("user", "root");
                    prop.setProperty("password", "root");
                    prop.setProperty("useUnicode", "true");
                    prop.setProperty("characterEncoding", "cp1251");
                    connection = DriverManager.getConnection(DB_URL,
prop);
                    String ID = "SELECT id FROM users WHERE login='" +
val[1] + "'";
                    pstmt = connection.prepareStatement(ID,
Statement.RETURN_GENERATED_KEYS);
                    ResultSet rs = pstmt.executeQuery(ID);
                    if (rs.next()) {
                        int senderId =
Integer.valueOf(rs.getString(1));
                        int resipId = Integer.valueOf(val[0]);
                        StringBuilder dialogId = new StringBuilder("");

                        if (senderId > resipId) {
                            dialogId.append(resipId).append(senderId);
                        } else {
                            dialogId.append(senderId).append(resipId);
                        }
                        dialogId.append(".txt");
                        System.out.println("ЗАПРОС для диалога: " +
dialogId);
                        if (!Files.exists(Paths.get("messages/" +
dialogId))) {
                            new File("messages/" +
dialogId).createNewFile();

                            System.out.println("СОЗДАНИЕ диалога " +
dialogId);
                            sendData("post", "list_msg", "none");
                        } else {
                            FileInputStream fin = new
FileInputStream("messages/" + dialogId);
                            int size = fin.available();
                            byte[] buffer = new byte[size];

                            fin.read(buffer, 0, size);

                            if (buffer.length != 0)
                                sendData("post", "list_msg", new
String(buffer));
                            else
                                sendData("post", "list_msg", "none");
                        }
                    }
```

```
                    connection.close();
                    pstmt.close();
                } catch(SQLException ex) {
                    ex.printStackTrace();
                } catch(Exception ex) {
                    ex.printStackTrace();
                }
                break;
            case "msgs":

                break;
            default:
                System.out.println(adr + ": message eror");
                break;
        }
    }
    void post_handler(String param, String data) {
        switch(param) {
            case "auth":
                String[] auth = data.split(":");
                setAuth(auth[0], auth[1]);
                break;
            case "contacts":
                try {
                    // Получение выбранного контакта от текущего
клиента
                    String[] addInfo = data.split(";");
                    System.out.println("From LOGIN:" + addInfo[0] + "
add contact with ID:" + addInfo[1]);

                    // Проверка на отсутствие добавляемого контакта в
таблице контактов текущего клиента
                    // Настройки подключения
                    Properties prop = new Properties();
                    prop.setProperty("user", "root");
                    prop.setProperty("password", "root");
                    prop.setProperty("useUnicode", "true");
                    prop.setProperty("characterEncoding", "cp1251");

                    // Подключение к БД
                    connection = DriverManager.getConnection(DB_URL,
prop);
                    String SEE = "SELECT id FROM " + addInfo[0] + "
WHERE id=" + addInfo[1];
                    pstmt = connection.prepareStatement(SEE,
Statement.RETURN_GENERATED_KEYS);
                    ResultSet rs = pstmt.executeQuery(SEE);
                    // Если результаты поиска отсутствуют
                    if (rs.next()) {
                        System.out.println("From LOGIN:" + addInfo[0] +
", выбранный контакт существует:" + addInfo[1]);
                    } else {
```

```
                        String ADD = "INSERT INTO " + addInfo[0] +
"(id) VALUES(?)";
                        pstmt = connection.prepareStatement(ADD,
Statement.RETURN_GENERATED_KEYS);
                        pstmt.setInt(1, Integer.valueOf(addInfo[1]));
                        pstmt.executeUpdate();
                        System.out.println("Login:" + addInfo[0] + "
add contact " + addInfo[1]);
                    }
                    connection.close();
                    pstmt.close();
                } catch(Exception ex) {
                    ex.printStackTrace();
                }
                break;
            default:
                System.out.println("Message eror from " + adr + param +
"|" + data);
                break;
        }
    }
    void setAuth(String login, String password) {
        try {
            // Настройки подключения
            Properties prop = new Properties();
            prop.setProperty("user", "root");
            prop.setProperty("password", "root");
            prop.setProperty("useUnicode", "true");
            prop.setProperty("characterEncoding", "cp1251");
            connection = DriverManager.getConnection(DB_URL, prop);
            String SEE = "SELECT password FROM users WHERE login IN ('"
+ login + "')";
            pstmt = connection.prepareStatement(SEE,
Statement.RETURN_GENERATED_KEYS);
            ResultSet rs = pstmt.executeQuery(SEE);

            if (rs.next() && rs.getString("password").equals(password))
{
                auth = true;
                sendData("post", "auth", "true");
                System.out.println(">>> Вход: " + login + "|" +
password);

                // Установка информации о клиенте в таблицу клиентов
                String DATA = "SELECT name, department, position FROM
users WHERE login='" + login + "'";
                pstmt = connection.prepareStatement(DATA,
Statement.RETURN_GENERATED_KEYS);
                pstmt.execute(DATA);
                final ResultSet rsData = pstmt.getResultSet();
                rsData.next();
                    name = rsData.getString(1);
```

```
                    dep = rsData.getString(2);
                    pos = rsData.getString(3);
                    server.setClientData(adr, name, dep, pos);
                    connection.close();
                    pstmt.close();
                } else {
                    sendData("post", "auth", "false");
                    System.err.println("Попытка входа: " + login +
"|" + password);
                }

                connection.close();
                pstmt.close();
            } catch(SQLException ex) {
                ex.printStackTrace();
            }
        }


        public void sendData(String type, String param, String data)
{
            try {
                String message = type + "#" + param + "#" + data;
                outMessage.println(message);
                outMessage.flush();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }

        public void close() {
            for (int i = 0; i < server.clients.size(); i++) {
                System.out.println("<<< Close handler with number: "
+ this.number);
                if (server.clients.get(i).number == this.number) {
                    server.removeClient(i);
                    break;
                }
            }
        }

    }
```

Файл DelUserDialog.java

```
package communicaserver;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
 import java.sql.SQLException;
```

```
    import java.sql.Statement;
    import java.util.Properties;
    import javax.swing.JButton;
    import javax.swing.JPanel;
    import javax.swing.JScrollPane;
    import javax.swing.JTable;
    import javax.swing.table.DefaultTableModel;

    public class DelUserDialog extends javax.swing.JDialog {
        static final String DB_URL =
"jdbc:mysql://localhost:3306/data";
        static final String DB_DRV = "com.mysql.jdbc.Driver";
        static final String DB_USER = "root";
        static final String DB_PASSWD = "root";
        JTable table;
        static DefaultTableModel model;
        JButton bDelUser;

        public DelUserDialog(java.awt.Frame parent) {
            super(parent, true);
            setSize(600,400);
            this.setLocation(parent.getLocation().x + 100,
parent.getLocation().y + 100);
            createGUI();
            connectDB();
        }
        void createGUI() {
            setTitle("Удаление клиентов");
            GridLayout grid = new GridLayout(1,6);
            grid.setHgap(10);
            grid.setVgap(5);

            String[] columnNames = {"Id", "Имя", "Отдел",
"Должность", "Логин", "Пароль"};
            model = new DefaultTableModel(0, columnNames.length);
            model.setColumnIdentifiers(columnNames);
            table = new JTable(model) {
                @Override
                public boolean isCellEditable(int arg0, int arg1) {
                    return false; // Запрет на редактирование
таблицы
                }
            };
table.getColumnModel().getColumn(0).setPreferredWidth(30);

table.getColumnModel().getColumn(1).setPreferredWidth(100);
            add(new JScrollPane(table), BorderLayout.CENTER);

            JPanel botPanel = new JPanel(new GridLayout(1,1));
            bDelUser = new JButton("Удалить клиента");
            botPanel.add(bDelUser);
            add(botPanel, BorderLayout.SOUTH);
            bDelUser.addActionListener(new ActionListener() {
                @Override
             public void actionPerformed(ActionEvent e) {
```

```
    int num_row = table.getSelectedRow();
                    int id =
Integer.parseInt(table.getValueAt(num_row, 0).toString());
                    String login = table.getValueAt(num_row,
4).toString();

                    deleteUser(id, num_row);
                    deleteUserCOntacts(login);
                }
            });
        }

        static void connectDB() {
            Connection connection = null;
            PreparedStatement pstmt = null;

            try {
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");
                connection = DriverManager.getConnection(DB_URL,
prop);
                String sql = "SELECT * FROM users";
                pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
                pstmt.execute(sql);
                final ResultSet rec = pstmt.getResultSet();
                while(rec.next()) {
                    model.addRow(new Object[] {
                        rec.getString(1),
                        rec.getString(2),
                        rec.getString(3),
                        rec.getString(4),
                        rec.getString(5),
                        rec.getString(6)});
                }
            } catch(SQLException ex) {
                ex.printStackTrace();
            } finally {
                try {
                    if (pstmt != null) pstmt.close();
                    if (connection != null) connection.close();
                } catch (SQLException ex) {

                }
            }
        }
        void deleteUser(int id, int num_row) {
            try {
                String DEL_USER = "DELETE FROM users WHERE id=" +
id;
```

```
    Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");

                // Подключение к БД
                Connection connection =
DriverManager.getConnection(DB_URL, prop);
                PreparedStatement pstmt =
connection.prepareStatement(DEL_USER,
                        Statement.RETURN_GENERATED_KEYS);
                pstmt.executeUpdate(DEL_USER);

                model.removeRow(num_row);
                if (pstmt != null) pstmt.close();
                if (connection != null) connection.close();
            } catch(SQLException ex) {
                ex.printStackTrace();
            }
        }

        void deleteUserCOntacts(String login) {
            try {
                String DEL_USER = "DROP TABLES " + login;
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");

                // Подключение к БД
                Connection connection =
DriverManager.getConnection(DB_URL, prop);
                PreparedStatement pstmt =
connection.prepareStatement(DEL_USER,
                        Statement.RETURN_GENERATED_KEYS);
                pstmt.executeUpdate(DEL_USER);
                if (pstmt != null)
                    pstmt.close();
                if (connection != null)
                    connection.close();
            } catch(SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
```

Файл CommunicaServer.java

```
package communicaserver;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class DelUserDialog extends javax.swing.JDialog {
    static final String DB_URL = "jdbc:mysql://localhost:3306/data";
    static final String DB_DRV = "com.mysql.jdbc.Driver";
    static final String DB_USER = "root";
    static final String DB_PASSWD = "root";

    JTable table;
    static DefaultTableModel model;
    JButton bDelUser;

    public DelUserDialog(java.awt.Frame parent) {
        super(parent, true);
        setSize(600,400);
        this.setLocation(parent.getLocation().x + 100,
parent.getLocation().y + 100);
        createGUI();
        connectDB();
    }

    void createGUI() {
        setTitle("Удаление клиентов");
        GridLayout grid = new GridLayout(1,6);
        grid.setHgap(10);
        grid.setVgap(5);

        String[] columnNames = {"Id", "Имя", "Отдел", "Должность",
"Логин", "Пароль"};
        model = new DefaultTableModel(0, columnNames.length);
        model.setColumnIdentifiers(columnNames);
        table = new JTable(model) {
            @Override
            public boolean isCellEditable(int arg0, int arg1) {
                return false; // Запрет на редактирование таблицы
            }
        };
        table.getColumnModel().getColumn(0).setPreferredWidth(30);
        table.getColumnModel().getColumn(1).setPreferredWidth(100);
        add(new JScrollPane(table), BorderLayout.CENTER);
                JPanel botPanel = new JPanel(new GridLayout(1,1));
```

```
        bDelUser = new JButton("Удалить клиента");
                botPanel.add(bDelUser);
                add(botPanel, BorderLayout.SOUTH);
                bDelUser.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        int num_row = table.getSelectedRow();
                        int id =
    Integer.parseInt(table.getValueAt(num_row, 0).toString());
                        String login = table.getValueAt(num_row,
    4).toString();

                        deleteUser(id, num_row);
                        deleteUserCOntacts(login);
                    }
                });
            }
            static void connectDB() {
                Connection connection = null;
                PreparedStatement pstmt = null;

                try {
                    Properties prop = new Properties();
                    prop.setProperty("user", "root");
                    prop.setProperty("password", "root");
                    prop.setProperty("useUnicode", "true");
                    prop.setProperty("characterEncoding", "cp1251");

                    // Подключение к БД
                    connection = DriverManager.getConnection(DB_URL,
    prop);
                    String sql = "SELECT * FROM users";
                    pstmt = connection.prepareStatement(sql,
    Statement.RETURN_GENERATED_KEYS);
                    pstmt.execute(sql);
                    final ResultSet rec = pstmt.getResultSet();

                    while(rec.next()) {
                        model.addRow(new Object[] {
                            rec.getString(1),
                            rec.getString(2),
                            rec.getString(3),
                            rec.getString(4),
                            rec.getString(5),
                            rec.getString(6)});
                    }
                } catch(SQLException ex) {
                    ex.printStackTrace();
                } finally {
                    try {
                        if (pstmt != null) pstmt.close();
                        if (connection != null) connection.close();
                    } catch (SQLException ex) {

                    }
            }
```

```
    }

        void deleteUser(int id, int num_row) {
            try {
                String DEL_USER = "DELETE FROM users WHERE id=" +
id;
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");
                Connection connection =
DriverManager.getConnection(DB_URL, prop);
                PreparedStatement pstmt =
connection.prepareStatement(DEL_USER,
                        Statement.RETURN_GENERATED_KEYS);
                pstmt.executeUpdate(DEL_USER);

                model.removeRow(num_row);
                if (pstmt != null) pstmt.close();
                if (connection != null) connection.close();
            } catch(SQLException ex) {
                ex.printStackTrace();
            }
        }


        void deleteUserCOntacts(String login) {
            try {
                String DEL_USER = "DROP TABLES " + login;
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");
                Connection connection =
DriverManager.getConnection(DB_URL, prop);
                PreparedStatement pstmt =
connection.prepareStatement(DEL_USER,
                        Statement.RETURN_GENERATED_KEYS);
                pstmt.executeUpdate(DEL_USER);

                if (pstmt != null)
                    pstmt.close();

                if (connection != null)
                    connection.close();
            } catch(SQLException ex) {
                ex.printStackTrace();
            }
        }
}
```

Файл AddUserDialog.java

```
package communicaserver;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class AddUserDialog extends javax.swing.JDialog {
    static final String DB_URL =
"jdbc:mysql://localhost:3306/data";
    static final String DB_DRV = "com.mysql.jdbc.Driver";
    static final String DB_USER = "root";
    static final String DB_PASSWD = "root";
    static JTextField tName, tDep, tPos, tLog, tPas;
    JButton but;
    public AddUserDialog(java.awt.Frame parent) {
        super(parent, true);
        setSize(280,290);
        this.setLocation(parent.getLocation().x + 100,
parent.getLocation().y + 100);
        createGUI();
    }
    void createGUI() {
        setTitle("Добавление клиентов");
        GridLayout grid = new GridLayout(6,1);
        grid.setHgap(5);
        grid.setVgap(5);
        JPanel pNorth = new JPanel(grid);
        JLabel lName = new JLabel("Имя");
        pNorth.add(lName);
        JLabel lDep = new JLabel("Отдел");
        pNorth.add(lDep);
        JLabel lPos = new JLabel("Должность");
        pNorth.add(lPos);
        JLabel lLog = new JLabel("Логин");
        pNorth.add(lLog);
        JLabel lPas = new JLabel("Пароль");
        pNorth.add(lPas);              add(pNorth,
BorderLayout.WEST);
```

```
        JPanel pCenter = new JPanel(grid);
                tName = new JTextField("");
                tName.setMargin(new Insets(3, 3, 3, 3));
                pCenter.add(tName);
                tDep = new JTextField("");
                tDep.setMargin(new Insets(3, 3, 3, 3));
                pCenter.add(tDep);
                tPos = new JTextField("");
                tPos.setMargin(new Insets(3, 3, 3, 3));
                pCenter.add(tPos);
                tLog = new JTextField("");
                tLog.setMargin(new Insets(3, 3, 3, 3));
                pCenter.add(tLog);
                tPas = new JTextField("");
                tPas.setMargin(new Insets(3, 3, 3, 3));
                pCenter.add(tPas);
                add(pCenter, BorderLayout.CENTER);

                JPanel pSouth = new JPanel();
                but = new JButton("Добавить");
                pSouth.add(but);
                but.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        char ch = tLog.getText().toCharArray()[0];
                        if (Character.isLetter(ch)) {
                            connectDB();
                        } else
                            JOptionPane.showMessageDialog(null,
                                "Логин должен начинаться с буквы!");
                    }
                });
                add(pSouth, BorderLayout.SOUTH);
            }

            static void connectDB() {
                Connection connection = null;
                PreparedStatement pstmt = null;

                try {
                    // Настройки подключения
                    Properties prop = new Properties();
                    prop.setProperty("user", "root");
                    prop.setProperty("password", "root");
                    prop.setProperty("useUnicode", "true");
                    prop.setProperty("characterEncoding", "cp1251");
                    connection = DriverManager.getConnection(DB_URL,
prop);
                    String LOG = "SELECT login FROM users WHERE login='"
+ tLog.getText() + "'";
                    pstmt = connection.prepareStatement(LOG,
Statement.RETURN_GENERATED_KEYS);            ResultSet rsLog =
pstmt.executeQuery(LOG);
```

```
                if (rsLog.next()) {
                    JOptionPane.showMessageDialog(null, "Данный
логин уже занят.");
                } else if (tLog.getText().equals("users")) {
                    JOptionPane.showMessageDialog(null, "Ндопутимое
значение в поле логина.");
                } else {
                    // Генерация id для нового клиента
                    int new_id = 0;
                    boolean isGen = false;
                    while (!isGen) {
                        new_id = 1000 + (int)(Math.random()*9000);
                        String SEE = "SELECT id FROM users WHERE
id=" + new_id;
                        pstmt = connection.prepareStatement(SEE,
Statement.RETURN_GENERATED_KEYS);
                        ResultSet rs = pstmt.executeQuery(SEE);

                        boolean isFind = true;
                        while (rs.next())
                            isFind = false;

                        if (isFind)
                            isGen = true;
                    }
                    String ADD = "INSERT INTO
users(id,name,department,position,login,password)
VALUES(?,?,?,?,?,?)";
                    pstmt = connection.prepareStatement(ADD,
Statement.RETURN_GENERATED_KEYS);
                    String name = tName.getText(),
                            dep = tDep.getText(),
                            pos = tPos.getText(),
                            log = tLog.getText(),
                            pas = tPas.getText();

                    if (name.length() != 0 &&
                        dep.length() != 0 &&
                        pos.length() != 0 &&
                        log.length() != 0 &&
                        pas.length() != 0) {
                        pstmt.setInt(1, new_id);
                        pstmt.setString(2, name);
                        pstmt.setString(3, dep);
                        pstmt.setString(4, pos);
                        pstmt.setString(5, log);
                        pstmt.setString(6, pas);
                        pstmt.executeUpdate();
                    }
                    String CREATE = "CREATE TABLE " + log + " (id
INTEGER not NULL)";                 pstmt =
connection.prepareStatement(CREATE,
Statement.RETURN_GENERATED_KEYS);
```

```
pstmt.executeUpdate(CREATE);

                clearInput();
            }
        } catch(SQLException ex) {
            ex.printStackTrace();
        } finally {
            try {
                if (pstmt != null) pstmt.close();
                if (connection != null) connection.close();
            } catch (SQLException ex) {

            }
        }
    }

    static void clearInput() {
        tName.setText("");
        tDep.setText("");
        tPos.setText("");
        tLog.setText("");
        tPas.setText("");
    }

}
```

Листинг клиентского приложения «CommunicaClient»
Файл SettnigsDialog.java

```
package communicaclient;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
public class SettingsDialog extends javax.swing.JDialog {
    public SettingsDialog (java.awt.Frame parent) {
        super(parent, true);
        setSize(200,150);
        this.setLocation(parent.getLocation().x + 100,
parent.getLocation().y + 100);
        createGUI();    }
    void createGUI() {
        setTitle("Настройки");
        JPanel panel = new JPanel(new BorderLayout());
        panel.add(new JLabel("Настройки для подключения",
SwingConstants.CENTER), BorderLayout.NORTH);
        GridLayout grid = new GridLayout(2,2);
        grid.setHgap(3);
        grid.setVgap(3);
        JPanel bodyPanel = new JPanel(grid);
        bodyPanel.add(new JLabel("IP-адрес",
SwingConstants.CENTER));
        bodyPanel.add(new JLabel("Порт",
SwingConstants.CENTER));
        JTextField tIP = new JTextField("",
SwingConstants.CENTER);
        bodyPanel.add(tIP);
        JTextField tPORT = new JTextField("",
SwingConstants.CENTER);
        bodyPanel.add(tPORT);
        panel.add(bodyPanel);
        JButton bSave = new JButton("Сохранить");
        bSave.setMargin(new Insets(5, 5, 5, 5));
        bSave.addActionListener(new ActionListener() {
```

```
    public void actionPerformed(ActionEvent e) {
                    if (tIP.getText().length() != 0 &&
                            tPORT.getText().length() != 0) {
                        try {
                            if (!Files.exists(Paths.get("data"))) {
                                File dir = new File("data");
                                dir.mkdir();
                                System.out.println("Директория data
создана");
                            }
                            if
(!Files.exists(Paths.get("data/settings.txt"))) {
                                new
File("data/settings.txt").createNewFile();
                                System.out.println("Файл
settings.txt создан");
                            }
                            StringBuilder msg = new
StringBuilder("");
                            msg.append(tIP.getText())
                                    .append(":")
                                    .append(tPORT.getText());
                            FileOutputStream fooStream = new
FileOutputStream("data/settings.txt", false);

fooStream.write(msg.toString().getBytes());

                            tIP.setText("");
                            tPORT.setText("");

                            Thread.sleep(100);
                        } catch (IOException ex) {
                            ex.printStackTrace();
                        } catch (InterruptedException ex) {
                            ex.printStackTrace();
                        }

                        System.out.println("ЗАПИСЬ новых настроек");
                    }
                }
            });
            panel.add(bSave, BorderLayout.SOUTH);
            add(panel);
        }
    }
```

Файл ContactsDialog.java

```
    package communicaclient;
    import java.awt.BorderLayout;
    import java.awt.Insets;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import java.util.Vector;
    import javax.swing.JButton;
    import javax.swing.JList;
    import javax.swing.JPanel;
```

```
    public class ContactsDialog extends javax.swing.JDialog {
        private boolean result;
        private int selectId = -1;
        private String select;
        private String data;
        public ContactsDialog (java.awt.Frame parent, String data) {
            super(parent, true);
            this.data = data;
            setSize(190,300);
            this.setLocation(parent.getLocation().x + 100,
parent.getLocation().y + 100);
            createGUI();
        }

        private void OK() {
            this.result = true;
            this.dispose();
        }
        private void Cancel(java.awt.event.MouseEvent evt) {
                this.result = false;
                this.dispose();
        }

        public int getSelectedId() {
            return selectId;
        }

        public String getSelected() {
            return select;
        }

        void createGUI() {
            setTitle("Добавить контакт");
            JPanel panel = new JPanel(new BorderLayout());

            String[] contacts = data.split(";");
            Vector<String> list = new Vector<String>();
            for (int i=0; i < contacts.length; i++) {
                list.add(contacts[i]);
            }
            JList<String> listContacts = new JList<String>(list);
            panel.add(listContacts, BorderLayout.CENTER);

            JButton bOK = new JButton("Добавить");
            bOK.setMargin(new Insets(5, 5, 5, 5));
            bOK.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    OK();
                    String[] contact =
listContacts.getSelectedValue().split(" - ");
                    try {
                        selectId = Integer.valueOf(contact[1]);
```

```
    select = listContacts.getSelectedValue();
                    } catch (Exception ex) {
                        System.err.print("ContactsDialog/Ошибка
добавления нового контакта");
                        selectId = -1;
                    }
                }
            });
            panel.add(bOK, BorderLayout.SOUTH);
            add(panel);
        }
    }
```

Файл CommunicaClient.java

```
    package communicaclient;
    import java.util.logging.Level;
    import java.util.logging.Logger;
    import javax.swing.UnsupportedLookAndFeelException;

    public class CommunicaClient {
        public static void main(String[] args) {
            for (javax.swing.UIManager.LookAndFeelInfo info
                    :
javax.swing.UIManager.getInstalledLookAndFeels()) {

                if ("Nimbus".equals(info.getName())) {
                    try {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    } catch (ClassNotFoundException ex) {
                        Logger.getLogger(CommunicaClient
                                .class.getName())
                                .log(Level.SEVERE, null, ex);
                    } catch (InstantiationException ex) {
                        Logger.getLogger(CommunicaClient
                                .class.getName())
                                .log(Level.SEVERE, null, ex);
                    } catch (IllegalAccessException ex) {
                        Logger.getLogger(CommunicaClient
                                .class.getName())
                                .log(Level.SEVERE, null, ex);
                    } catch (UnsupportedLookAndFeelException ex) {
                        Logger.getLogger(CommunicaClient
                                .class.getName())
                                .log(Level.SEVERE, null, ex);
                    }
                    break;
                }
            }
            ClientWindow clientWindow = new ClientWindow();
```

```
    }

}
```

Файл ClientWindow.java

```
package communicaclient;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.Socket;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.NoSuchElementException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;

public class ClientWindow extends JFrame {

    static final String SERVER_HOST = "192.168.56.1";
    static final int SERVER_PORT = 665;
    boolean auth = false;
    int GAP = 10;
    Socket clientSocket = null;
    Scanner inMessage;
    PrintWriter outMessage;

    JTextField tLog, tPas, tMsg;
    JButton bLog, bSet, bAdd, bSend, bUpdate;
    DefaultListModel modelContacts;
    JList listMsgs, listContacts;
    DefaultListModel modelMsgs;
    String selContactId, selContactName;

    public String getClientName() {
        return this.SERVER_HOST;
    }


    public ClientWindow() {
        checkFiles();
        createGUI();
        setAuth(auth);
        setListeners();
        setVisible(true);
    }
    void checkFiles() {
        try {
            if (!Files.exists(Paths.get("data"))) {
                File dir = new File("data");
                dir.mkdir();
```

```
    System.out.println("Директория data создана");
                }

                if (!Files.exists(Paths.get("data/settings.txt"))) {
                    new File("data/settings.txt").createNewFile();
                    System.out.println("Файл settings.txt создан");
                }
            } catch(IOException ex) {
                ex.printStackTrace();
            }
        }


        void createGUI() {
            setBounds(520, 125, 600, 450);
            setTitle("Client");
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            GridLayout grid = new GridLayout(1,3);
            grid.setHgap(3);
            JPanel topPanel = new JPanel(grid);

topPanel.setBorder(BorderFactory.createTitledBorder("Основная
панель"));
            add(topPanel, BorderLayout.NORTH);
            tLog = new JTextField("val");
            topPanel.add(tLog);
            tPas = new JTextField("val");
            topPanel.add(tPas);
            bLog = new JButton("Войти");
            topPanel.add(bLog);
            bSet = new JButton("Настройки");
            topPanel.add(bSet);
            JPanel leftPanel = new JPanel(new BorderLayout());

leftPanel.setBorder(BorderFactory.createTitledBorder("Контакты"));
            bAdd = new JButton("Добавить новый контакт");
            leftPanel.add(bAdd, BorderLayout.NORTH);
            modelContacts = new DefaultListModel();
            listContacts = new JList(modelContacts);
            MouseListener mouseListener = new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    if (e.getClickCount() == 1 &&
listContacts.getSelectedValue()!= null) {
                        String[] newSelId = listContacts

.getSelectedValue().toString().split(" - ");
                        if (newSelId.length == 2) {
                            selContactId = newSelId[1];
                            selContactName = newSelId[0];
                            bAdd.setEnabled(false);
                            listContacts.setEnabled(false);
                            listMsgs.setEnabled(false);

                            System.out.println("Открытие диалога с "
```

```
                                        +
listContacts.getSelectedValue());
                                modelMsgs.clear();
                                modelMsgs.addElement(">>> Диалог с "
+ listContacts.getSelectedValue().toString().split(" - ")[0]);
                                sendData("get", "list_msg", selContactId
+ "&" + tLog.getText());
                            }
                        }
                    }
                };
                listContacts.addMouseListener(mouseListener);
                leftPanel.add(new JScrollPane(listContacts),
BorderLayout.CENTER);
                modelContacts.addElement("Авторизируйтесь");
                add(leftPanel, BorderLayout.WEST);

                // центральная панель
                JPanel centralPanel = new JPanel(new BorderLayout());

centralPanel.setBorder(BorderFactory.createTitledBorder("Диалог"));
                // Лист с сообщениями
                modelMsgs = new DefaultListModel();
                listMsgs = new JList(modelMsgs);
                centralPanel.add(new JScrollPane(listMsgs),
BorderLayout.CENTER);
                // верхняя панель
                JPanel controlPanel = new JPanel(new BorderLayout());
                bUpdate = new JButton("Обновить сообщения");
                bUpdate.addMouseListener(mouseListener);
                controlPanel.add(bUpdate);
                centralPanel.add(controlPanel, BorderLayout.NORTH);
                // нижняя панель
                JPanel msgPanel = new JPanel(new BorderLayout());
                bSend = new JButton("Отправить");
                bSend.setMargin(new Insets(5, 5, 5, 5));
                msgPanel.add(bSend, BorderLayout.EAST);
                tMsg = new JTextField("");
                tMsg.setMargin(new Insets(5, 5, 5, 5));
                msgPanel.add(tMsg, BorderLayout.CENTER);
                centralPanel.add(msgPanel, BorderLayout.SOUTH);
                add(centralPanel, BorderLayout.CENTER);
            }
            void setAuth(boolean bool) {
                tLog.setEditable(!bool);
                tPas.setEditable(!bool);
                tMsg.setEditable(bool);
                bAdd.setEnabled(bool);
                bSend.setEnabled(bool);
            }
            void setContacts(String data) {
                try {
                    if (data.equals("none")) {
                        modelContacts.clear();
                        modelContacts.addElement("Нет контактов");
```

```
    } else if (data.equals("close")) {
                    modelContacts.clear();
                    modelContacts.addElement("Авторизируйтесь");
                } else {
                    String[] contacts = data.split(";");
                    modelContacts.clear();
                    for (int i = 0; i < contacts.length; i++) {
                        modelContacts.addElement(contacts[i]);
                    }
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
        void setListeners() {
            bLog.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    new Thread(new Runnable() {
                        @Override
                        public void run() {
                            try {
                                if (!auth) {
                                    FileInputStream fin = new
FileInputStream("data/settings.txt");
                                    int size = fin.available();
                                    if (size != 0) {
                                        byte[] buffer = new
byte[size];
                                        fin.read(buffer, 0, size);
                                        String[] settings = new
String(buffer, "Cp1251").split(":");
                                        if (settings.length == 2) {
                                            clientSocket = new
Socket(settings[0], Integer.valueOf(settings[1]));
                                            inMessage = new
Scanner(clientSocket.getInputStream());
                                            outMessage = new
PrintWriter(clientSocket.getOutputStream());
                                            startListenServer();
                                            bLog.setEnabled(false);
                                        } else
JOptionPane.showMessageDialog(null, " подключения");
                                    }  else
JOptionPane.showMessageDialog(null, "данные для подключения");
                                } else if (auth) {
                                    closeConnection();
                                    auth = false;
                                    setAuth(auth);
                                    bLog.setText("Войти");
                                    bAdd.setEnabled(true);
                                    listContacts.setEnabled(true);
                                    listMsgs.setEnabled(true);
                                }
                                Thread.sleep(100);
```

```
                            } catch (Exception ex) {
                                System.err.println(ex.getMessage());
                                JOptionPane.showMessageDialog(null,
                                        "Не удаётся подключиться к
серверу, проверьте настройки подключения");
                            }
                        }
                    }).start();
                }
            });

            bAdd.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    sendData("get", "contacts", "-");
                }
            });

            bSend.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    if (tMsg.getText().length() > 0 &&
tMsg.getText().length() < 100) {
                        sendData("msg",
                            selContactId + "&" + tLog.getText(),
                            tMsg.getText());
                        modelMsgs.addElement("Вы: " +
tMsg.getText());
                        tMsg.setText("");
                    } else
                        JOptionPane.showMessageDialog(null,
                            "Введите сообщение длиной от 1 до 100
символов!");
                }
            });
            bSet.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    SettingsDialog dialog = openSettingsDialog();
                    dialog.setVisible(true);
                }
            });
            // закрытие окна клиентского приложения
            addWindowListener(new WindowAdapter() {
                @Override
                public void windowClosing(WindowEvent e) {
                    super.windowClosing(e);
                    closeConnection();
                }
            });
        }
        SettingsDialog openSettingsDialog() {
            SettingsDialog dialog = new SettingsDialog(this);
            return dialog;
```

```
        }


        // слушаем сервер
        void startListenServer() {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        System.out.println("Подключен к серверу.");
                        while (true) {
                            if (inMessage == null) {
                                Thread.sleep(50);
                            } else {
                                String str = inMessage.nextLine();

                                if (str.equals("exit")) {
                                    System.err.println("Связь с
сервером прекратилась");
                                    break;
                                } else {
                                    String[] msg = str.split("#");
                                    switch(msg[0]) {
                                        case "msg":
                                            msg_handler(msg[1],
msg[2]);
                                            break;
                                        case "post":
                                            post_handler(msg[1],
msg[2]);
                                            break;
                                        case "get":
                                            get_handler(msg[1],
msg[2]);
                                            break;
                                        default:

System.out.println(msg[0] + msg[1] + msg[2]);
                                            break;
                                    }
                                }
                            }
                        } catch (NoSuchElementException ex) {

                        } catch (IndexOutOfBoundsException ex) {
                        } catch (IllegalStateException ex) {
                        } catch (Exception ex) {
                        }
                    }
                }).start();
        }
```

```
    void msg_handler(String param, String data) {
            switch(param) {
                case "msg":

                    break;
                default:
                    System.out.println("Message eror from server " +
param + "|" + data);
                    break;
            }
        }
        void post_handler(String param, String data) {
            switch(param) {
                case "contacts":
                    // Приём всего списка клиентов от сервера
                    ContactsDialog dialog = new ContactsDialog(this,
data);
                    dialog.setVisible(true);

                    if (dialog.getSelectedId() != -1) {
                        boolean exist = false;
                        for (int i = 0; i < modelContacts.getSize();
i++)
                            if
(modelContacts.getElementAt(i).toString().equals(dialog.getSelected(
)))
                                exist = true;
                        if (!exist)
modelContacts.addElement(dialog.getSelected());
                        sendData("post", "contacts", tLog.getText()
+ ";" + dialog.getSelectedId());
                    }
                    break;
                case "message":

                    break;
                case "auth":
                    try {
                        if (data.equals("true")) {
                            auth = true;
                            sendData("get", "client_contacts",
tLog.getText());
                            setAuth(auth);
                            bLog.setText("Выйти");
                            System.out.println("Вы авторизованы!");
                        } else {
                            auth = false;
                            closeConnection();
                            System.err.println("Не удалось
авторизоваться!");
                        }
                        bLog.setEnabled(true);
                    } catch(Exception ex) {
                        bLog.setEnabled(true);
                    }
```

```
                    break;
                case "client_contacts":
                    System.out.println("Получение списка контактов:
" + data);
                    modelContacts.clear();
                    setContacts(data);
                    break;
                case "list_msg":
                    // Загружаем полученные сообщения в окно
сообщений
                    bAdd.setEnabled(true);
                    listContacts.setEnabled(true);
                    listMsgs.setEnabled(true);

                    if (!data.equals("none")) {
                        String[] msgs = data.split(";");

                        /*try {
                            // Преобразование кодировки сообщений
                            msgs = new String(data.getBytes(),
"Cp1251").split(";");
                        } catch (UnsupportedEncodingException ex) {
                            ex.printStackTrace();
                        }*/
                        System.err.println("СООБЩЕИЕ: " + data);

                        for (int i = 0; i < msgs.length; i++) {
                            StringBuilder bodyMsg = new
StringBuilder("");
                            String[] message = msgs[i].split(":");
                            if (message[0].equals(selContactId)) {

bodyMsg.append(selContactName).append(": ").append(message[1]);
                            } else
                                bodyMsg.append("Вы:
").append(message[1]);

modelMsgs.addElement(bodyMsg.toString());
                            }
                        }
                    break;
                default:
                    System.err.println("message eror");
                    break;
            }
        }
        void get_handler(String param, String data) {
            switch(param) {
                case "auth":
                    sendData("post", "auth", tLog.getText() + ":" +
tPas.getText());
                    break;
                default:
                    System.err.println("message eror");
                    break;
```

```
            }
        }


        public void sendData(String type, String param, String data)
{
            try {
                String message = type + "#" + param + "#" + data;
                outMessage.println(message);
                outMessage.flush();
            } catch (Exception ex) {
                System.err.println("Ошибка отправки сообщения: " +
ex.getLocalizedMessage());
            }
        }


        void closeConnection() {
            try {
                // если подключение существует
                if (clientSocket != null) {
                    outMessage.println("exit");
                    outMessage.flush();
                    outMessage.close();
                    if (inMessage != null)
                        inMessage.close();
                    inMessage = null;
                    clientSocket.close();
                    bLog.setEnabled(true);
                    modelMsgs.clear();
                    setContacts("close");
                }
            } catch (IOException exc) {
                System.err.println("Не удалось разорвать текущее
подключение с сервером.");
                bLog.setEnabled(true);
            }
        }


        @Override
        public synchronized void addWindowListener(WindowListener
wl) {
            super.addWindowListener(wl);
        }

    }
```

ПРИЛОЖЕНИЕ Г

(Презентация)

ФГБО УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
КАФЕДРА «ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

ЧАТ С АВТОРИЗАЦИЕЙ В БАЗЕ ДАННЫХ ДЛЯ ОБМЕНА ДАННЫМИ В РАМКАХ ЛОКАЛЬНОЙ СЕТИ

Выполнил:
студент группы 16в82
Белов В.А.
Руководитель:
д.т.н., профессор кафедры ВТ
Зинкин С.А.

Пенза 2020

АКТУАЛЬНОСТЬ РАБОТЫ

- Технологии интернет жизни проникли во все сферы жизни, в том числе в социальную. В связи с этим, для сохранения положения при котором компания способна конкурировать, ни одна компания не в состоянии обойтись без привлечения сторонних технологий.
- Из-за этого, рост числа схожих программных реализаций и спрос на подобное программное обеспечение, неуклонно растёт.

ЦЕЛИ И ЗАДАЧИ

- Цель работы состоит в том, чтобы создать клиент-серверное приложение для осуществления процесса по обмену текстовыми данными в локальной сети.
- Задачи:
  - Изучение предметной области;
  - Анализ предъявляемых требований;
  - Выбор средств и технологий для разработки;
  - Разработка программных модулей;
  - Отладка и тестирование программы.

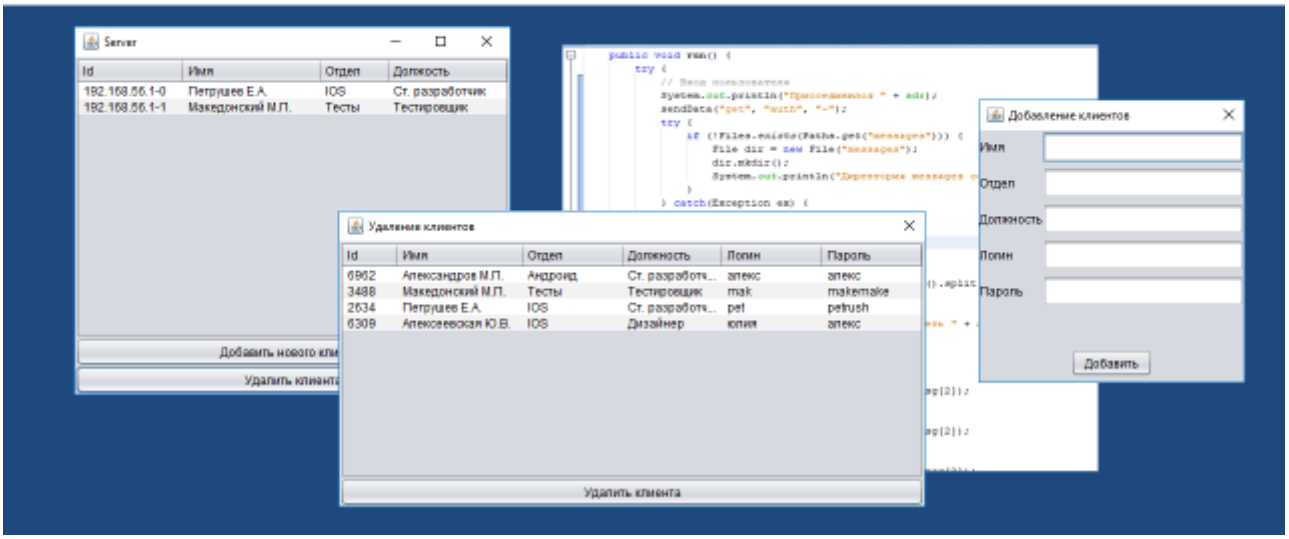
АНАЛИЗ ТРЕБОВАНИЙ К СИСТЕМЕ

- Реализация архитектуры клиент-сервер
- Осуществление клиент-серверного взаимодействия
- Авторизация в системе
- Хранение данных в БД
- Ведение списка контактов каждого пользователя
- Обмен сообщениями

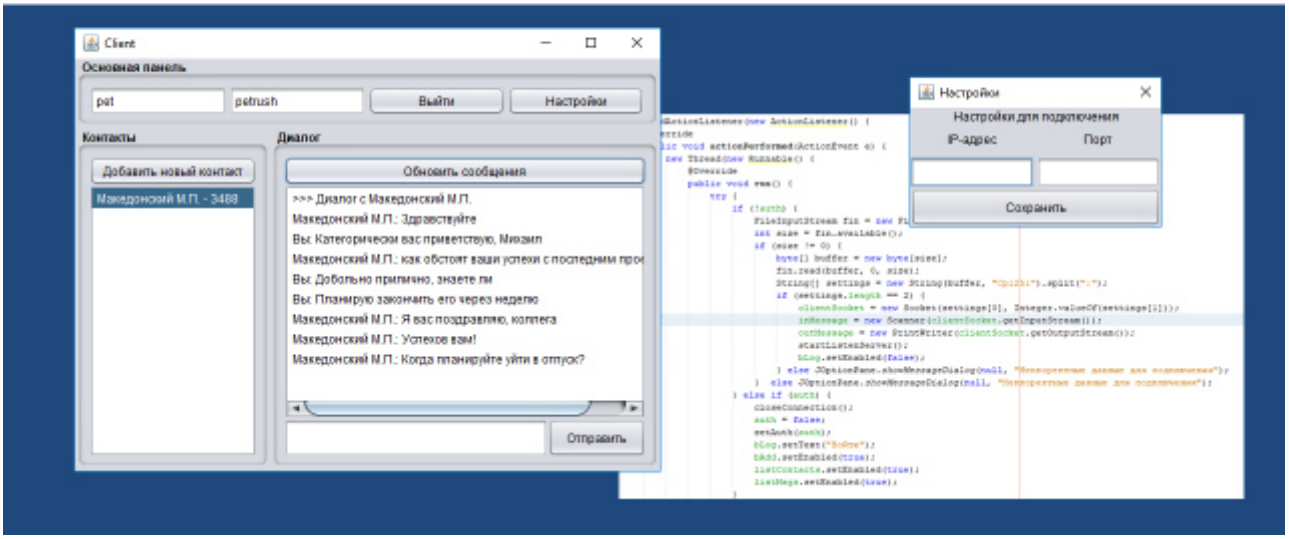
ВЫБОР ЯЗЫКОВ, СРЕДСТВ И ТЕХНОЛОГИЙ

- Средства и технологии
  - NetBeans IDE
  - Server Socket
  - JDBC
  - Maven
  - Swing
- Языки
  - Java
  - SQL

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ



РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ



ТЕСТИРОВАНИЕ

- Тестирование отдельных модулей
- Тестирование взаимодействия разработанных модулей
- Отладка процесса отправки сообщений
- Проверка цельности передачи данных в установленном соединении
- Исправление и отладка обнаруженных ошибок
- Регулярная повторная отладка

ВЫВОДЫ

- В результате проделанной работы:
  - Разработано и отлажено клиент-серверное приложение для обмена строковой информацией в локальной сети
  - Составлены и реализованы алгоритмы работы чата
  - Составлена и оформлена соответствующая документация

СПАСИБО ЗА ВНИМАНИЕ