

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

учреждение образования
«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики
Кафедра современных технологий программирования

Огурцов Александр Андреевич

Мессенджер «HisMes»

Курсовая работа
студента 2 курса специальности
1-40 01 01 «Программное обеспечение информационных технологий»
дневной формы получения образования

Научный руководитель
Куриян Николай Николаевич
Старший преподаватель кафедры
современных технологий
программирования

Гродно, 2020

РЕЗЮМЕ

Тема курсовой работы

Мессенджер «HisMes»

Работа содержит: 37 страниц, 17 рисунков, 5 использованных источников литературы.

Ключевые слова: мессенджер, онлайн, TCP, java, javafx, socket, sockeysServer.

Цель курсовой работы – получения практических навыков программирования сетевых протоколов, получение навыков проектирования компьютерных сетей.

Объект исследования – реализация онлайн мессенджера.

Предмет исследования – реализация и разработка алгоритмов windows приложений.

Методы исследования: теоретическое исследование.

Данное приложение разработано с целью коммуникаций людей в сети интернет, по принципу общего чата и личных сообщений.

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	3
ВВЕДЕНИЕ	5
ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	6
1.1 Основный аспекты создания приложения	6
1.2 Обзор существующих решений для созданий приложений с ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ.....	8
1.3 Вывод по первой главе	10
ГЛАВА 2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ	11
2.1 Постановка задач для реализации приложения.....	11
2.2 Проектирование функций приложения	14
2.3 Выводы по главе 2	15
ГЛАВА 3. РЕАЛИЗАЦИЯ СИСТЕМЫ.....	16
3.1 Описание структуры проекта	16
3.2 Работа с приложением	28
3.3 Вывод по главе 3	35
ЗАКЛЮЧЕНИЕ	36
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	37

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ

Java – Язык программирования

Javafx – технологии для создания графических приложений на языке программирования Java.

Бд – база данных

Введение

В данном документе описывается программа, написанная при помощи языка java с использованием технологий для разработки графических приложений javafx. По теме «Мессенджер «HisMes»». Данная программа даёт возможность общаться с людьми зарегистрированными в данном приложении, на основе общего чата или личных сообщений. Реализована система аккаунтов с личной информации для удобного поиска.

В современном мире ни один человек не может обойтись без коммуникаций с другими людьми. Одни встречаются и вместе гуляют на улице, другие звонят друг другу по телефону. Но оба варианта не подходят если вы находитесь далеко друг от друга, в разных странах. В таком случае вам поможет мессенджер. В котором вы можете общаться по средствам интернет связи, в любое время суток вы можете отправить сообщение любому человеку, который находится даже в самом отдаленном уголке мира.

В чем польза мессенджера?

Общение с человеком на больших расстояниях, одно из самых важных плюсов всех мессенджеров. Так же нельзя не учесть цену общения, она зависит от стоимости интернета которым вы пользуетесь каждый день. Связь людей в разных часовых поясах.

Предусмотрено решение следующих задач:

- * разработка приложения на языке программирования java;
- * разработка удобного графического пользовательского интерфейса;
- * разработка серверной части мессенджера;
- * разработка соединения клиентской части с серверной.

ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Основные аспекты создания приложения.

Этап 1. Уточнение задач. На самом первом этапе разрабатывается список задач, которые должны выполняться данным приложением, включая и те, которые не используются в данный момент, но могут появиться в последующей работе программы. Под «основными» задачами понимаются функции, которые должны быть реализованы в формах или отчетах приложения.

Этап 2. Последовательность выполнения задач. Чтобы программа работала стабильно и удобно, лучше всего сгруппировать важные задачи в группы, а потом отсортировать задачи каждой группы так, чтобы они были расположены в порядке их выполнения. Может произойти так, что определенные задачи будут связаны с разными группами или за выполнением отдельной задачи будет идти выполнение другой, не принадлежащей к этой группе.

Этап 3. Анализ данных. После определения задач, один из важнейших этапов это реализация подробного списка всех данных, которые будут необходимы для решения последующей задачи. Определенные данные будут использоваться как исходные и изменяться не будут. Какие-то данные изменяться в ходе выполнения задачи. Возможно, какие-то элементы будут удалены или добавлены. Некоторые данные получатся при помощи вычислительных операции: их результат будет элементом задачи, но записываться в бд они не будут.

Этап 4. Определение структуры данных. По завершению первоначального анализа все необходимые элементы данных необходимо упорядочить по объектам и сопоставить элементы(объекты) с таблицами и командами бд. Для реляционных бд таких, как Access используется процесс, называемый нормализацией, в результате данного этапа разрабатывается наиболее эффективная структура данных.

Этап 5. Разработка макета приложения и пользовательского интерфейса. По завершению предыдущего этапа, в Microsoft Access достаточно просто можно создать его макет с помощью форм и объединить их в одно целое, используя достаточно простые макросы или процедуры обработки событий. Такой макет достаточно просто можно предоставить заказчику и получить его согласие на выполнение поставленных задач еще до практической реализации приложения.

Этап 6. Создание приложения. Если же задачи достаточно просты, то созданный макет является почти законченным приложением. Но все же достаточно часто приходится прибегать к написанию процедур, позволяющие упростить решение и реализацию всех поставленных задач. Значит, придется создать какие-то объединяющие формы, которые организуют переход от задачи к задаче.

Этап 7. Тестирование и усовершенствование. Финальным этапом разработки является проверка функционирования приложения в каждом из существующих режимов. Следует протестировать работу макросов, прибегая к использованию режим отладки, где будет исполняться определенная макрокоманда. В ходе разработки автономных разделов приложения рекомендуется передать их заказчику для проверки их правильности и выслушать мнение о необходимости внесения каких либо изменений. После ознакомления заказчика с выполненной работой, у него могут возникнуть какие-либо предложения по улучшению или модернизации. Поиск необходимых изменений на первых стадиях разработки приложения позволяет сократить время на следующие переделки.

1.2. Обзор существующих решений для созданий приложений с использование сетевой передачи данных и графическим интерфейсом

На данный момент существует большое количество решений для создания приложений: WhatsApp

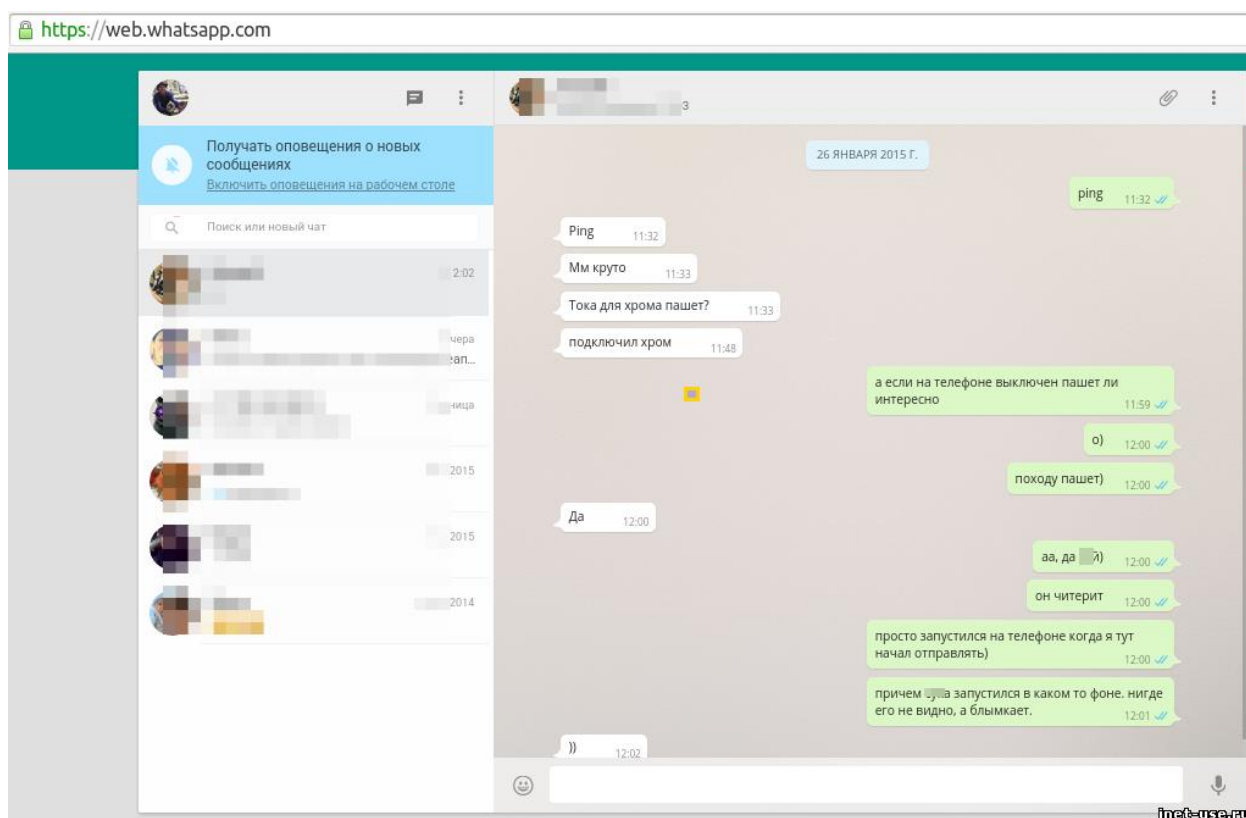


Рис. 1.1.1. «WhatsApp»

WhatsApp — это бесплатное приложение, которое предлагает простой, безопасный и надёжный обмен сообщениями и звонками, доступное на мобильных телефонах по всему миру. Приложение создавалось для замены sms сообщений между людьми. Для упрощения коммуникации.

Достоинства:

- * простой и понятный интерфейс;
- * возможность отправлять медиа файлы.

* возможность видео/аудио связи

2) Telegram — кроссплатформенный мессенджер, позволяющий обмениваться сообщениями и медиафайлами многих форматов. Используются проприетарная серверная часть с закрытым кодом, работающая на мощностях нескольких компаний США и Германии, финансируемых Павлом Дуровым в объёме порядка 13 млн долларов США ежегодно, и несколько клиентов с открытым исходным кодом, в том числе под лицензией GNU GPL. (рис.1.1.2)

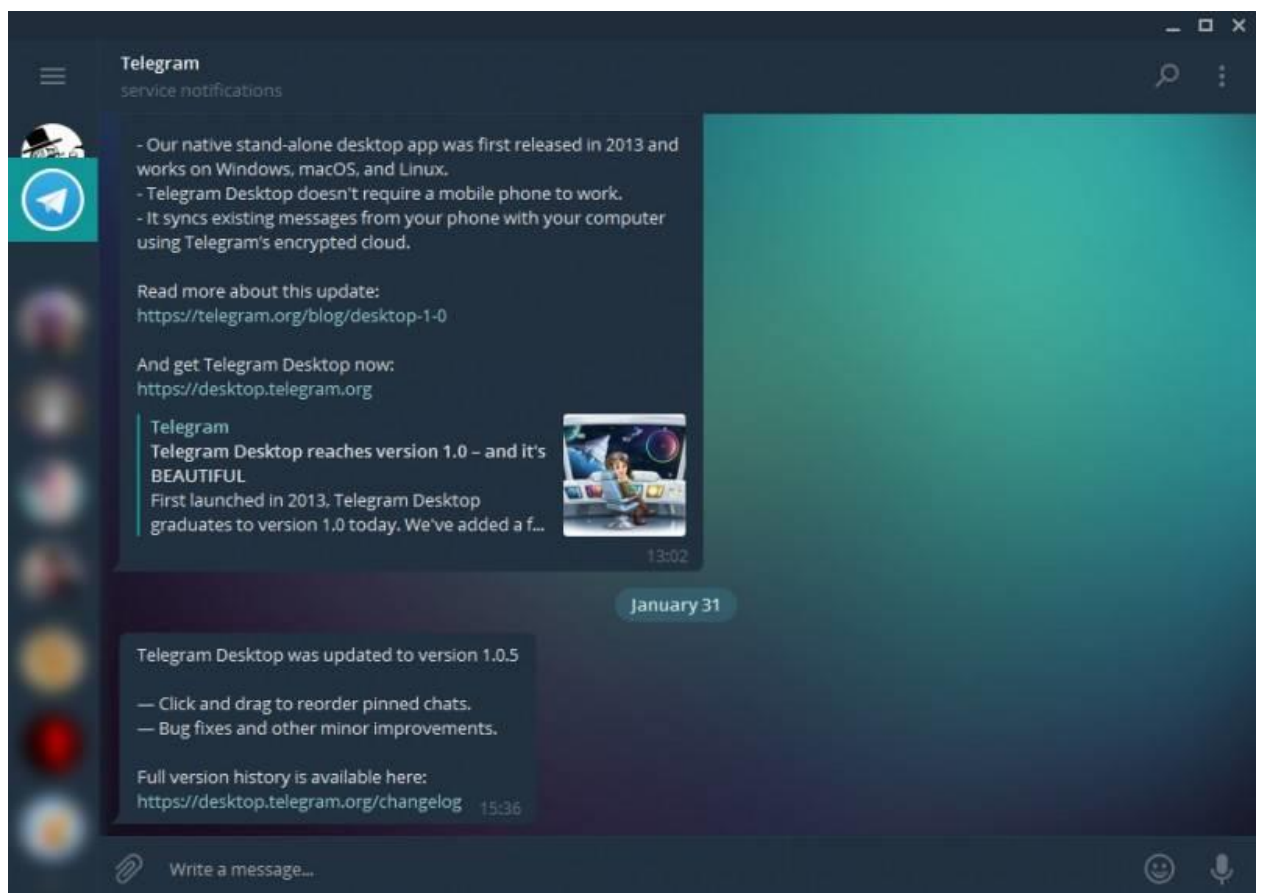


Рис. 1.1.2. «Telegram»

Достоинства:

- * Красивый дизайн
- * Возможность изменять цвет фона
- * простота

1.3. Выводы по главе 1

В первой главе были рассмотрены глобальные и наиболее подходящие решения в области создания приложений с графическим интерфейсом. Были разобраны несколько приложений, такие как: WhatsApp и Telegram. Каждое решение было описано. В ходе исследования существующих решений были отмечены основные достоинства приложений такие как простота, удобство использования, недостатков не было выявлено, так как компании глобального масштаба, такие как WhatsApp Inc и Telegram Messenger LLP, постоянно оптимизируют и улучшают свои продукт.

Глава 2

Проектирование приложения

2.1 Постановка задач для реализации приложения

Требуется реализовать приложение. Программа должна выполнять авторизацию или регистрацию пользователя, общение пользователей в общем чате, поиск профилей по критериям, передача личных сообщений. Программа должна иметь удобный интерфейс, обновлять сообщения в реальном времени. Должна выполняться аутентификация пользователя или регистрацию.

Для создания графического интерфейса на языке java был использован интерфейс программирования приложений javafx.

JavaFX — это инструмент для создания графического интерфейса пользователя для Java.

Graphical user interface — графический интерфейс пользователя — это разновидность пользовательского интерфейса, в котором все элементы представленные пользователю на дисплее, выполнены в виде картинок, графики.

В отличие от интерфейса командной строки, в GUI у пользователя есть произвольный доступ к видимым объектам с помощью устройств ввода. Зачастую элементы интерфейса реализованы в виде метафор и отображают их свойства и назначение для облегчения понимания пользователя.

JavaFX ориентирован на создание игр и настольных приложений на Java. По сути им заменят Swing из-за нового инструмента GUI для Java. Также, он позволяет нам стилизовать файлы компоновки и сделать их элегантнее с помощью CSS, подобно тому, как реализовано в сетевых приложениях.

JavaFX дополнительно работает со встроенной 3D-графикой, а также аудио, видео и встроенными сетевыми приложениями. Он прост в освоении и

хорошо оптимизирован. Он поддерживает множество операционных систем, а также Windows, UNIX системы и Mac OS.

JavaFX изначально поставляется с огромным набором частей графического интерфейса, таких как кнопки, текстовые поля, таблицы, деревья, меню, диаграммы и т.д., что в свою очередь сэкономит нам вагон времени.

JavaFX часто использует стили CSS, и может использовать специальный формат FXML для создания GUI, а не делать это в коде Java. Это облегчает быстрое размещение графического интерфейса пользователя или изменение внешнего вида или композиции без необходимости долго играть в коде Java.

JavaFX имеет готовые к использованию части диаграммы, поэтому не требуется писать их с нуля в любое время, когда вам нужна базовая диаграмма.

JavaFX дополнительно поставляется с поддержкой 3D графики, которая часто полезна, если мы разрабатываем какую-то игру или подобные приложения.

Stage — это окружающее окно, которое используется как начальное полотно и содержит в себе остальные компоненты. У приложения может быть несколько stage, но один такой компонент должен быть в любом случае. По сути Stage является основным контейнером и точкой входа.

Scene — отображает содержание stage (прям матрёшка). Каждый stage может содержать несколько компонентов — scene, которые можно между собой переключать. Внутри это реализуется графом объектов, который называется — Scene Graph (где каждый элемент — узел, ещё называемый как Node).

Node — это элементы управления, например, кнопки метки, или даже макеты (layout), внутри которых может быть несколько вложенных компонентов. У каждой сцены (scene) может быть один вложенный узел (node), но это может быть макет (layout) с несколькими компонентами.

Вложенность может быть многоуровневой, когда макеты содержат другие макеты и обычные компоненты. У каждого такого узла есть свой идентификатор, стиль, эффекты, состояние, обработчики событий.

В работе представлено полное описание программы и её действий, что она выполняет и каким образом работает. Прилагаются скриншоты результата работы программы.

2.2. Проектирование функций приложения

В первой главе были разобраны реализации решений по данной тематике. Сейчас необходимо разработать набор функций приложения, которые пользователь сможет использовать.

- * авторизация;
- * регистрация;
- * загрузка профиля пользователя;
- * подключение и общение в общем чате с авторизованными пользователями;
- * возможность отключаться/подключаться/очищать общий чат;
- * запись логов общего чата;
- * поиск профилей других людей по разным критериям;
- * загрузка профилей других пользователей и отображение основной информации;
- * общение с пользователем по средствам личных сообщений;
- * шифрование данных;
- * реализация серверов для получения и обработки сообщений и личной информации.

Данный набор функций позволит пользователям и администраторам удобно и просто пользоваться данным приложением.

2.3. Выводы по главе 2

В ходе выполнения работы по проектированию приложения были выведены основные функции приложения:

- * авторизация;
- * регистрация;
- * загрузка профиля пользователя;
- * подключение и общение в общем чате с авторизованными пользователями;
- * возможность отключаться/подключаться/очищать общий чат;
- * запись логов общего чата;
- * поиск профилей других людей по разным критериям;
- * загрузка профилей других пользователей и отображение основной информации;
- * общение с пользователем по средствам личных сообщений;
- * шифрование данных;
- * реализация серверов для получения и обработки сообщений и личной информации.

ГЛАВА 3. Реализация системы

3.1 Описание структуры проекта

Проект состоит из двух частей клиентской и серверной.

Клиентская часть проекта состоит из 8 классов (рис 3.1.1).

(«appController», «CheckValidDate » , «InputOutputData» , «LoginWin» , «Main» , «MessageHandling» , «PrivateMessage » , «UserInfo»)

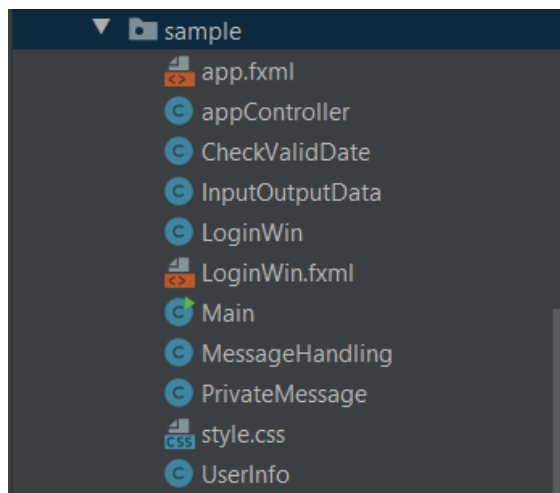


Рис. 3.1.1 Клиентская часть

Серверная часть состоит из трех серверов. Для обработки личных данных пользователя, авторизации и регистрации используется DataServer (рис.3.1.2)

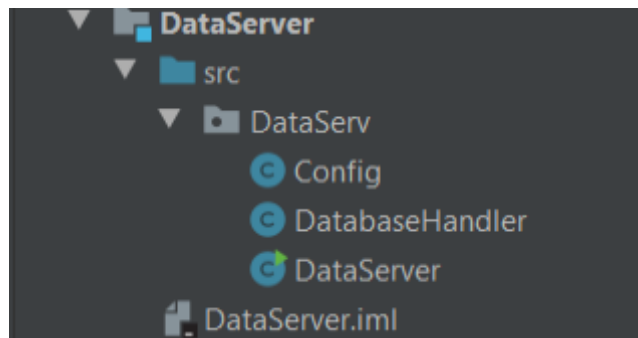


Рис. 3.1.2 Серверная часть (сервер для обработки данных пользователя)

Для обработки сообщений общего чата, сохранения логов используется ChatServer (рис.3.1.3).

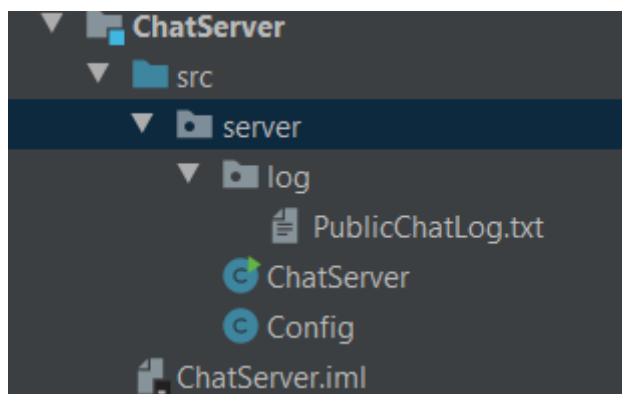


Рис. 3.1.3 Серверная часть (сервер для обработки сообщений общего чата)

Для обработки личных сообщений PrivateChatServ (рис.3.1.4).

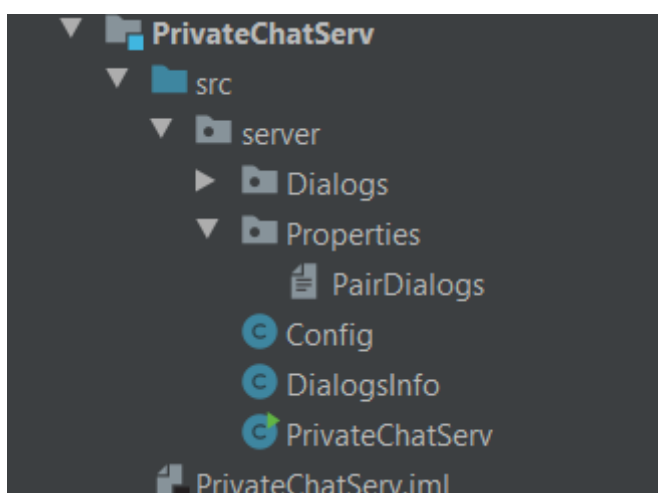


Рис. 3.1.4 Серверная часть (сервер для обработки личных сообщений)

Вся информация о пользователях хранится в базе данных на базе phpMyAdmin (рис. 3.1.5).

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	login	varchar(120)	utf8_general_ci		No	None			Change Drop More
3	pass	varchar(1000)	utf8_general_ci		No	None			Change Drop More
4	salt	varchar(120)	utf8_general_ci		No	None			Change Drop More
5	Name	varchar(32)	utf8_general_ci		No	None			Change Drop More
6	Surname	varchar(32)	utf8_general_ci		No	None			Change Drop More
7	ip	varchar(32)	utf8_general_ci		No	None			Change Drop More
8	country	varchar(120)	utf8_general_ci		No	None			Change Drop More
9	mail	varchar(32)	utf8_general_ci		No	None			Change Drop More
10	bDay	varchar(120)	utf8_general_ci		No	None			Change Drop More
11	regDay	varchar(120)	utf8_general_ci		No	None			Change Drop More
12	gender	varchar(11)	utf8_general_ci		No	None			Change Drop More

Рис. 3.1.5 - Структура базы данных

В таблице хранится информация о пользователях. Таблица имеет следующие поля:

- Id – индекс пользователя;
- Login – логин пользователя;
- Pass – пароль пользователя;
- Salt – ключ для шифрования;
- Name – имя;
- Surname – фамилия;
- ip – ип адрес пользователя;
- Country – страна пользователя;
- Mail – email адрес;
- bDay – дата рождения пользователя;
- regDay – дата регистрации пользователя;
- gender – пол пользователя;

Для работы программы были написаны 3 сервера для передачи/обработки информации.

При запуске сервера ожидаю подключения к ним. При подключений к серверу, по средствам TCP протокола с использованием socket и socketServer, создается поток для обработки подключения.

Для взаимодействия сервера и клиента используется интерфейс ConnectionLis и класс Connection в котором определены конструкторы для подключения и принятия подключения сервером. Создается поток для обработки подключения. Метод void sendString (String msg) отправляет сообщения, метод void disconnected() прерывает поток и закрывает сокет (соединение). Так же переопределен метод toString().

В интерфейсе ConnectionLis определены 4 метода:

- void ConnectionReady(Connection connection);
- void ReceiveString(Connection connection, String value);
- void Disconnection (Connection connection);
- void Exception(Connection connection, Exception e);

Запуск клиентской программы начинается с класса Main с функции main, в которой создается окно (LoginWin.fxml) авторизации и основные элементы управления. Для обработки события окна используется LoginWin.java. Для передачи данных используется класс InputOutputData.java.

Для авторизации пользователю необходимо ввести данные логин и пароль и нажать кнопку войти. После нажатия кнопки, при помощи метода boolean CheckTextField (Object object) проверяется поля логина и пароля на пустоту. Если поля не пусты создается соединение с сервером для обработки данных (DataServer) куда передается логин для получения ключа шифрования. А так же запускается таймер (TimerLogin(String value)) для обработки полученных сообщений от сервера. При получении от сервера ключа идет генерация зашифрованного пароля по средства ключа и введенного пароля, отправка данных на сервер для проверки валидной комбинации, при помощи метода boolean SetPass(). В методе SetPass() вызывается метод класса InputOutputData, boolean LoginUserSendConnection

(String login, String pass) в котором на сервер отправляются данные и активация таймера для ожидания ответа сервера. При успешном ответе от сервера вызывается метод ShowAndWaitNewWindow() для отображения главного меню приложения (app.fxml), получение от сервера основной информации о текущем пользователе, списка всех пользователей, а так же закрытие окна авторизации. Если аккаунта не существует, то необходимо зарегистрироваться. При нажатии кнопки «зарегистрироваться» все поля проверяются на пустоту при помощи метода CheckField() и checkFieldIsEmpty(). Если все поля заполнены вызывается метод OutputDate(String msg) для записи данных в базу данных. Запускается таймер для ожидания ответа сервера.

Загрузка основного окна начинается с заполнения информации о текущем пользователе, создания объекта MessageHandling и PrivateMessage которые используют интерфейс ConnectionLis. При нажатиях кнопки «Общий чат» запускается таймер (CheckInputMsg(boolean flag)) для обработки полученных сообщений от сервера, создается подключение с сервером обработки сообщений общего чата. Метод void CheckInputMsg(boolean flag) при получении от сервера сообщения вызывает метод void printMsg(String msg) для вывода сообщения на экран. Для списка кнопок chatSettings реализованы события:

```
EventHandler<ActionEvent>logClearEvent=new  
EventHandler<ActionEvent>()  
Event> disconnectEvent = new EventHandler<ActionEvent>()  
EventHandler<ActionEvent> connectEvent = new EventHandler<  
ActionEvent
```

Для быстрого подключения/отключения от сервера передачи сообщений, а так же для очистки чата.

Отправка сообщений реализована при нажатий кнопки Enter или кнопки SendMesButton при помощи метода EnterTextField(). Метод EnterTextField() проверят поле ввода сообщения на пустоту и если поле не

пустое, то сообщение отправляется на сервер для отправки всем пользователям при помощи метода `SendToAllConnection(String msg)` сервера `ChatServer`.

При нажатии кнопки `searchPeopleButton` («Поиск») в `ListView` загружается список всех пользователей, отображается панель со списком всех пользователей и поля поиска, а так же кнопки для дополнительных параметров поиска. Для загрузки списка всех пользователей используется метод `SetListView()` класса `appController`. В методе вызываются методы `SetImageToListView()`, `SetActionToClickListView()` для отображения картинок пользователей и обработки события выбора элемента в `ListView` соответственно. Дополнительное меню поиска открывается/закрывается при помощи метода `advantagesMenu(boolean flag)`. При выборе пользователя из списка заполняется информация, на панели `UserInfoPanel`, по текущему пользователю. Панель `UserInfoPanel` становится активной.

На панели `UserInfoPanel` активна кнопка `privateDialogButton` («Написать сообщение») при нажатии на которую вызывается метод `privateDialogsButtonClick()`. В методе создается подключение к серверу передаче и обработке личных сообщений, на сервер отправляется запрос для получения ключа диалога, активируется таймер (`CheckInputPrivateMsg`) для получения сообщений. После получения ключа диалога вызывается метод `InputArrayWithFile` класса `PrivateMessage` для получения списка сообщений которые находятся в файле название которого есть ключ диалога. Если сообщения в файле присутствуют они загружаются в `TextArea` и количество сообщений передается на сервер для проверки новых сообщений. Если количество сообщений меньше чем на сервере, то пересылаются зашифрованные недостающие сообщения, которые при помощи метода `InputLogFile` записываются в файл и добавляются в список сообщений в `TextArea`. По таймеру `CheckInputMsg(Boolean flag)` отправляются запросы на сервер для проверки новых сообщений.

Методы и переменные класса `LoginWin`:

@FXML

void RegistrationButton() – событие кнопки регистрации («Зарегистрироваться»)

@FXML

void BackButtonClick() - событие кнопки, вернуться назад к авторизации

public static void showErrorMessage(String Teg, String message) – сообщение

@FXML

void LoginButtonClick() – событие кнопки «Войти»

public void authorization() - авторизация

protected boolean SetPass() – создание пароля по ключу и отправка на сервер

@FXML

void RegButtonClick() – событие кнопки регистрации.

boolean checkFieldsEmpty() – проверка полей на пустоту

public boolean CheckField() – проверка полей

public void ShowAndWaitNewWindow() – закрытие окна и создание главного окна

boolean CheckTextField(Object object) – проверка object на пустоту

Методы и переменные класса InputOutputData :

protected static final String IP_ADDR = "localhost"; - ип сервера

protected static final int PORT = 8152; - порт сервера

protected static ArrayList<String> log = new ArrayList<>(); - список для логов

protected static boolean flag = false; - флаг для обработки ответов сервера

protected static String currentKey = null; - ключ авторизации

private Connection connection;

Timer timer = new java.util.Timer(); - таймер для обработки ответов сервера

ArrayList<String> Users = new ArrayList<>(); - список пользователей

private boolean CreateConnection() – соединение с сервером

protected void GetKeySendConnection(String login) – получение ключа по логину

public synchronized boolean TimerLogin(String value) - таймер для обработки ответов

private void CheckLogin() – проверка существования логин

protected boolean GetUserInfo(String login) - получение информации о пользователе

private void setUserInfo(String user) - запись данных в список пользователей

protected boolean LoginUserSendConnection(String login, String pass) – авторизация пользователя

protected void LoginUser() – попытка авторизации

public void CheckKey() – проверка ключа

public void CheckRepeatL() – проверка логина на повторение

public void CheckRepeatE() – проверка почты на повторение

protected boolean OutputDate(String msgToOut) - отправка данных на сервер для регистрации

public void RepeatFieldSendConnection(String value) - отправка запроса на проверку повтора данных
@Override
public void ReceiveString(Connection connection, String value) - возврат сообщения сервера
public void NextThreadToGetUserList() – поток для получения списка пользователей
public void ParseStrToUserInfo()
public void GetAllUserList() – список всех пользователей
public void CheckInputMsg(boolean flag) – таймер получения списка пользователей.

Методы и переменные класса appController:

PrivateMessage privateDialogs = new PrivateMessage();
ArrayList<String> currentDialog = new ArrayList<>(); - список сообщений
MessageHandling tt = new MessageHandling();
Timer timer = new java.util.Timer();
Timer timerToPrivateMSG;
String TwoLogins;
@FXML
void initialize() – инициализация окна
public void SetListView() – список пользователей для поиска
private void SetImageToListView() – картинки пользователей

private void SetActionToClickListView() – обработка событий ListView

public void SetButtonToSettingChat() – кнопки в общем чате

EventHandler<ActionEvent> logClearEvent = new EventHandler<ActionEvent>() – событие очищения экрана

EventHandler<ActionEvent> disconnectEvent = new EventHandler<ActionEvent>() – событие отключения

EventHandler<ActionEvent> connectEvent = new EventHandler<ActionEvent>() – событие подключения
@FXML
public void HomeButtonClick() – событие нажатия кнопки «Главная»
@FXML
public void EnterPrivateTextField() – отправить личное сообщение
@FXML
public void PublicChatButtonClick() – событие нажатия кнопки «Общий чат»
@FXML
public void SearchButton() – событие нажатия кнопки «Поиск»
@FXML
public void privateChatButtonClick() - событие нажатия «Сообщения»
@FXML
public void privateBackButtonClick() – событие нажатия кнопки «назад» в личных сообщениях
@FXML
public void AdvancedSettingButtonClick() – событие кнопки «Дополнительные параметры поиска»
@FXML
public void backAdvantagesButtonClick() – событие кнопки «скрыть»

@FXML

public void backUnfoUserButtonClick() – событие кнопки «назад» поиск пользователей

@FXML

public void privateDialogsButtonClick() – событие кнопки «Написать сообщение»

public void advantagesMenu(boolean flag) – дополнительное меню поиска

public void CheckInputMsg(boolean flag) – таймер сообщений общего чата

public void CheckInputPrivateMsg(boolean flag) - таймер сообщений личного чата

public synchronized void printMsg(String msg) - отобразить сообщение в общем чате

public synchronized void printPrivateMsg(String msg) – отобразить сообщение в личном чате

public void StopPrivateChat() – остановка таймеров личных сообщений

public void EnterTextField() – отправить текст общего чата

public void EnterTextFieldSearchUser() отправить текст личных сообщений

Методы и переменные класса MessageHandling:

protected static final String IP_ADDR = "localhost"; - ип для подключения к серверу

protected static final int PORT = 8085; - порт для подключения к серверу

public ArrayList<String> logs = new ArrayList<>(); - запись сообщений от сервера

public Connection connection;

protected boolean CreateConnection(String login) – создание соединения с сервером

public void SendMes(String msg) - отправить сообщение на сервер

@Override

public void ConnectionReady(Connection connection) - соединение готово

@Override

public void ReceiveString(Connection connection, String value) - ответ сервера

@Override

public void Disconnection(Connection connection) – отключение

@Override

public void Exception(Connection connection, Exception e) - ошибка

Методы и переменные класса CheckValidDate:

public void CheckRepeatLogin(String login) - проверка на повтор логина

public void CheckRepeatEmail(String mail) - проверка на повтор почты

public boolean CheckCorrectEmail(String email) - проверка на правильность ввода почты

public String generateSalt() – генерирование ключа

public String bytetoString(byte[] input)

public byte[] getHashWithSalt(String input, byte[] salt) throws NoSuchAlgorithmException - шифрование

public byte[] stringToByte(String input)

public static String convertStringToHex(String str)

Методы и переменные класса PrivateMessage :

protected static final String IP_ADDR = "localhost"; - ип сервера

protected static final int PORT = 8015; - порт сервера

public Connection connection;


```

private String Currentkey ; - ключ диалога
public ArrayList<String> logs = new ArrayList<>();
private final String pathToLog = "F:\\sem4\\kys4\\src\\cache\\"; - путь диалогов
protected boolean CreateConnection() – соединение с сервером
public void CheckDialogKey(String msg) - проверка ключа диалога
private int CountLogStr() – количество сообщений с кэше
public void SendLoginToGetKey(String CurrentUserLogin, String Login) – получение ключа диалога
private void SendCountStr(int counter) - отправка количества сообщений
public Timer timerS;
public void CheckInputMsg(boolean flag) – таймер проверка новых сообщений
public void InputLogFile(String msg) –запись нового сообщения в кэш
public void InputArrayWithFile(ArrayList<String> temp) – чтение из кэша
public void SendMes(String msg) - отправить сообщение
@Override
public void ConnectionReady(Connection connection) - соединение готово
@Override
public void ReceiveString(Connection connection, String value) – ответ сервера
@Override
public void Disconnection(Connection connection) – отключение
@Override
public void Exception(Connection connection, Exception e) – ошибка

```

Серверная часть:

PrivateChatServ:

Методы и переменные класса PrivateChatServ:

```

public static void main(String[] args) – запуск программы, вызов конструктора
private final ArrayList<Connection> connections = new ArrayList<>(); - подключения к серверу
DialogsInfo dialogs; - объекта класса
private PrivateChatServ() – конструктор
@Override
public synchronized void ConnectionReady(Connection connection) - соединение готово
@Override
public synchronized void ReceiveString(Connection connection, String value) – запрос клиента
@Override
public synchronized void Disconnection(Connection connection) – отключение
@Override
public synchronized void Exception(Connection connection, Exception e) – ошибка

```

Методы и переменные класса DialogsInfo:

```

private final ArrayList<DialogsInfo> DialogsName = new ArrayList<>(); список всех диалогов
String key; - ключ
String UserOne; - пользователь 1
String UserTwo; - пользователь 2

```

```

    int countStr; - количество сообщений
private final String pathToPairLogin =
"F:\\sem4\\kyrs4\\PrivateChatServ\\src\\server\\Properties\\PairDialogs"; - путь до пар
логин/логин/ключ
private final String dialogPath = "F:\\sem4\\kyrs4\\PrivateChatServ\\src\\server\\Dialogs\\"; - путь к
диалогам
public DialogsInfo() - конструктор
public DialogsInfo(String key,String UserOne, String UserTwo,int countStr) - конструктор
private void GetDialName() – получение ключа
private int GetCountOfStr(String path) – количество сообщений
private int GenerateKey() – генерация ключа
public String searchDialogKey(String msg) - поиск ключа
private void SetKeyToFile(DialogsInfo dialog) – добавление диалога
public boolean AddMessageToFile(String str) – добавление сообщения
@Override
public String toString() – переопределение строки вывода
public ArrayList<String> DifferenceStr(int count, String key) – сообщения разности.

```

DataServer:

Методы и переменные класса DataServer:

```

public static void main(String[] args) – старт сервера
private final ArrayList<Connection> connections = new ArrayList<>(); - подключенные клиенты
private DataServer() – конструктор
@Override
public synchronized void ConnectionReady(Connection connection) – соединение готово
public int counter(ResultSet result) - количество записей
public String CheckRepeat(ResultSet result, String value, String suffix) – проверка на повтор
private String sendMes(ResultSet resultSet, String suffix) – отправка сообщения
@Override
public synchronized void ReceiveString(Connection connection, String value) – запрос клиента
public String ParseUserInfo(ResultSet resultSet) - перевод в строку
@Override
public synchronized void Disconnection(Connection connection) - отключение
@Override
public synchronized void Exception(Connection connection, Exception e) – ошибка

```

Методы и переменные класса DatabaseHandler:

```

Connection dbConnections;
public Connection getDbConnections() throws ClassNotFoundException, SQLException - соединение с
бд
public boolean signUpUser(String msg) – регистрация пользователя
public ResultSet LoginUser(String loginPas) - авторизация
public ResultSet RepeatField(String value) – проверка на повтор
public ResultSet GetKeyUser(String value) - получение ключа пользователя
public ResultSet GetUser(String login) – информация о пользователе
public ResultSet GetCountUser() – информация о пользователях

```

ChatServer:

Методы и переменные класса ChatServer:

```
public static void main(String[] args) – запуск сервера
private final ArrayList<Connection> connections = new ArrayList<>(); - подключенные устройства
private FileWriter writer;
private ChatServer() – конструктор
@Override
public synchronized void ConnectionReady(Connection connection) – соединение готово
@Override
public synchronized void ReceiveString(Connection connection, String value) – запрос клиента
@Override
public synchronized void Disconnection(Connection connection) – отключение
@Override
public synchronized void Exception(Connection connection, Exception e) – ошибка
private void SendToAllConnection(String value) – отправка всем пользователям
private void PublicMessFile(String value) – запись сообщений в лог
private void CheckAliveConnection() – проверка соединений
```

Общий класс Connection:

```
private final Socket socket;
private final Thread nextThread;
private final BufferedReader in;
private final BufferedWriter out;
private ConnectionLis eventList;
public Connection(ConnectionLis eventList, String ipAddr, int port) throws IOException – конструктор
public Connection(ConnectionLis eventList, Socket socket) throws IOException – конструктор
public synchronized void sendString (String msg) – отправка сообщения
public synchronized void disconnected() – отключиться
public String OutputAdr() – вывод текущего адреса
public boolean CheckAliveConnect() – проверка соединения
@Override
public String toString() – переопределения сообщения
```

Интерфейс ConnectionLis:

void ConnectionReady(Connection connection); - соединение готово

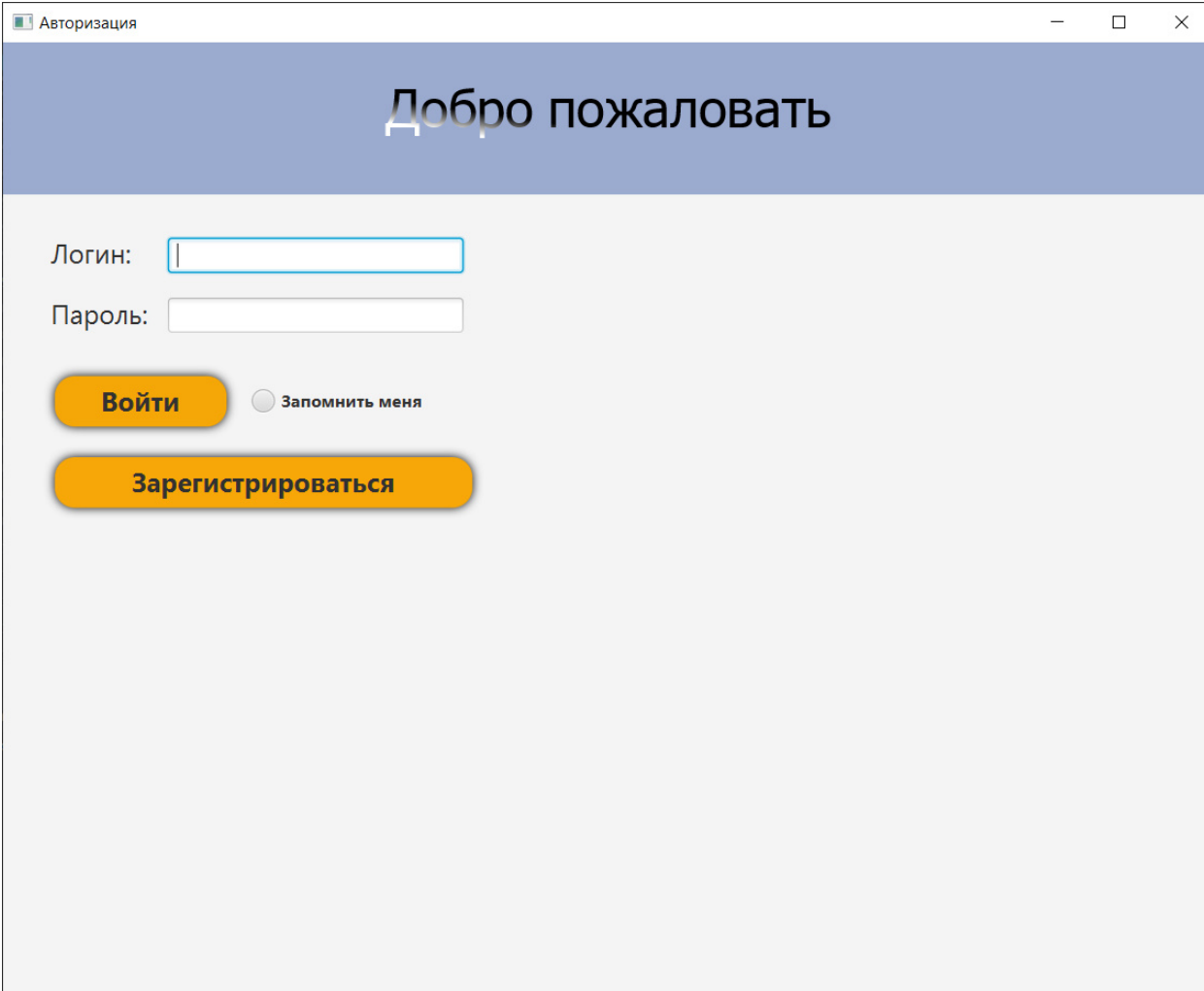
void ReceiveString(Connection connection, String value); - получение сообщения

void Disconnection (Connection connection); - отключение

void Exception(Connection connection, Exception e); - ошибка.

3.2 Работа с приложением

Запуск приложения выполняется с появления окна авторизации



Авторизация

Добро пожаловать

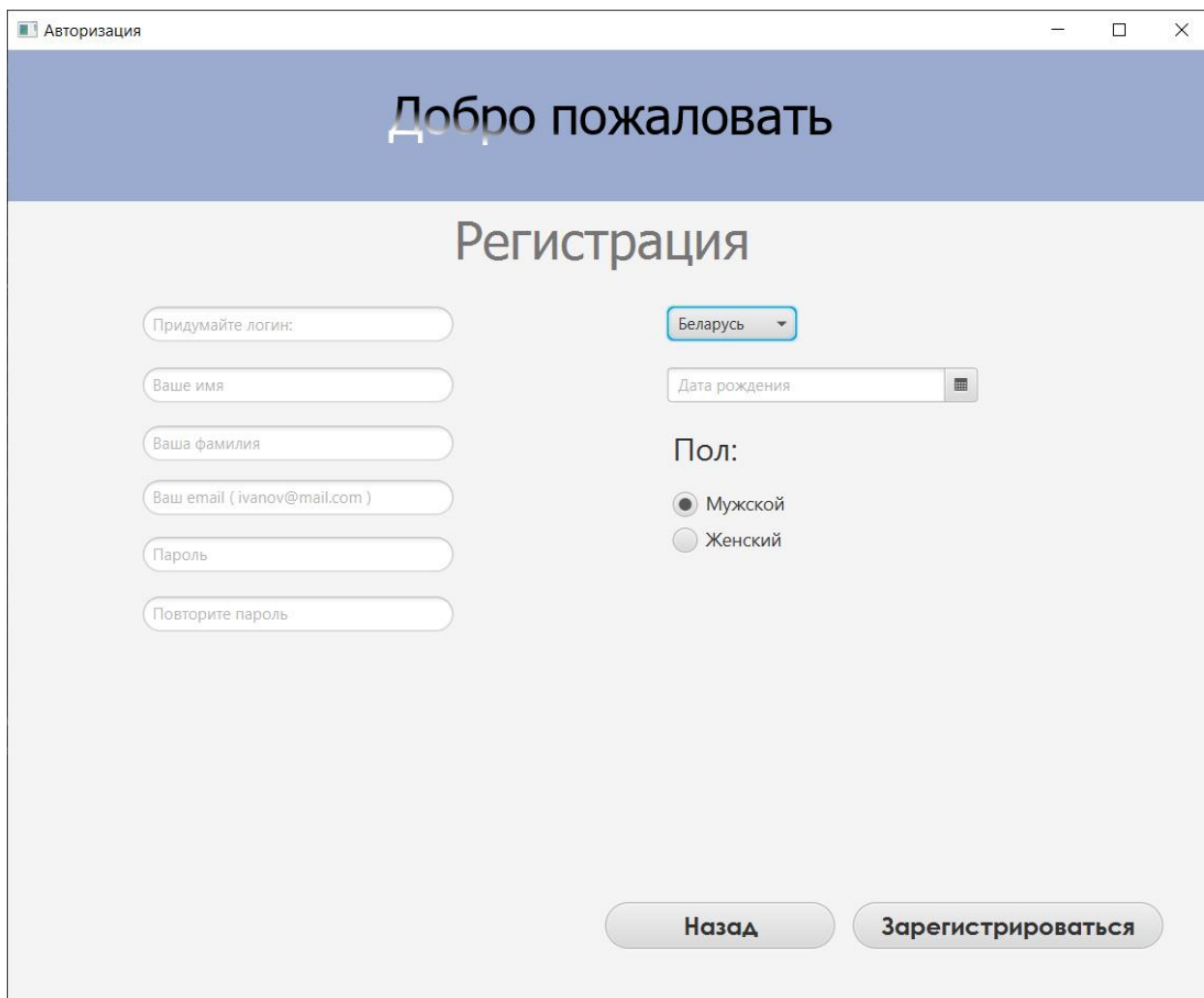
Логин:

Пароль:

☐ Запомнить меня

Рис. 3.2.1. Окно при запуске приложения

У пользователя есть несколько вариантов действий, ввод данных в виде логин и пароль и нажатие кнопки «Войти» или вызов окна регистрации при помощи кнопки «Зарегистрироваться» (рис.3.2.2).



Авторизация

Добро пожаловать

Регистрация

Придумайте логин:

Ваше имя

Ваша фамилия

Ваш email (ivanov@mail.com)

Пароль

Повторите пароль

Беларусь

Дата рождения

Пол:

☒ Мужской

☐ Женский

Назад

Зарегистрироваться

Рис. 3.2.2. Окно регистрации

При заполнении всех полей и получении сообщений о успешной регистрации, пользователю необходимо авторизоваться под новым логином и паролем.

При успешной авторизации появится окно с главным экраном программы (рис. 3.2.3).



Рис. 3.2.3. Главное окно программы.

Главное окно программы состоит из меню основного окна. При загрузке отображается главная страница на которой отображается профиль пользователя с основной информацией.

При нажатии кнопки «Общий чат» отображается чат с возможностью отправлять сообщения и информацией о подключениях/отключений пользователей (рис 3.2.4). В правом верхнем углу общего чата расположен список кнопок для быстрого отключения и подключения к чату, а так же для очищения окна чата (рис 3.2.5). Отправление сообщений возможно при помощи клавиши Enter или кнопки возле поля ввода сообщения.

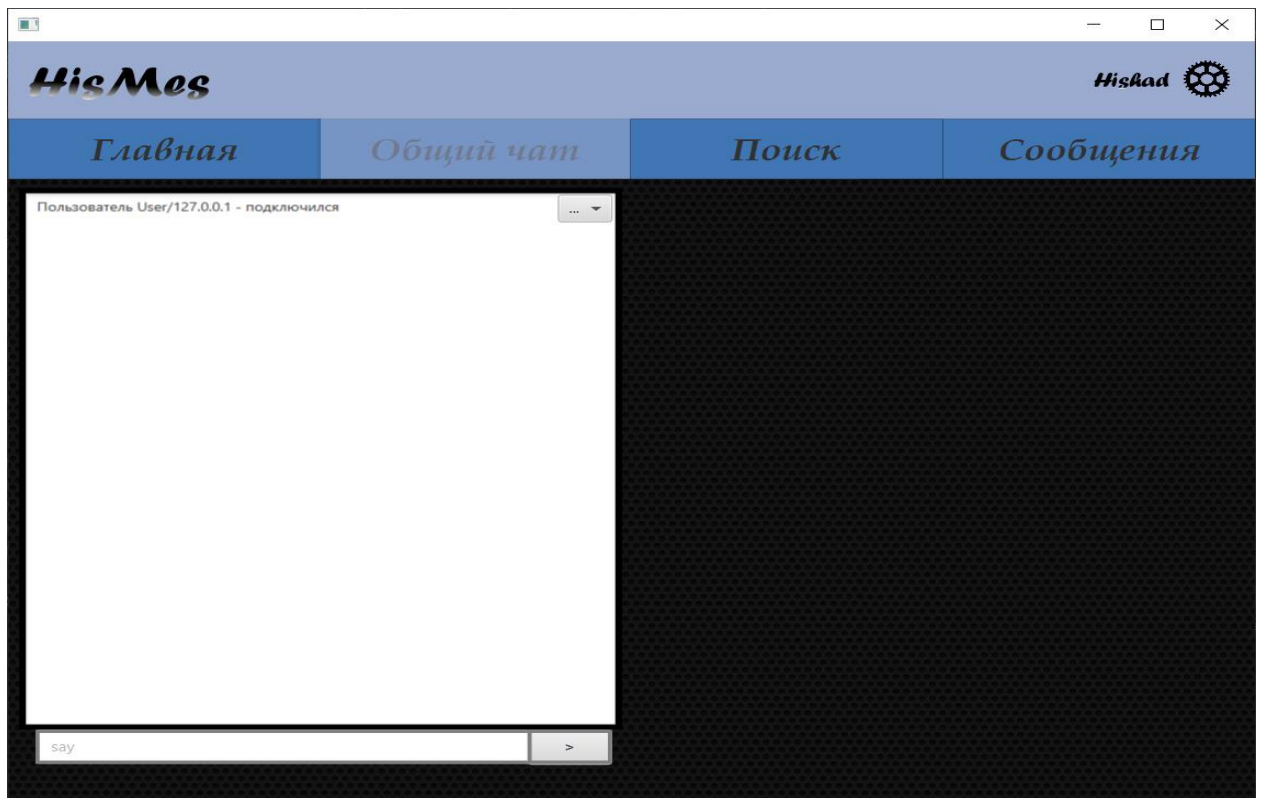


Рис. 3.2.4. Общий чат

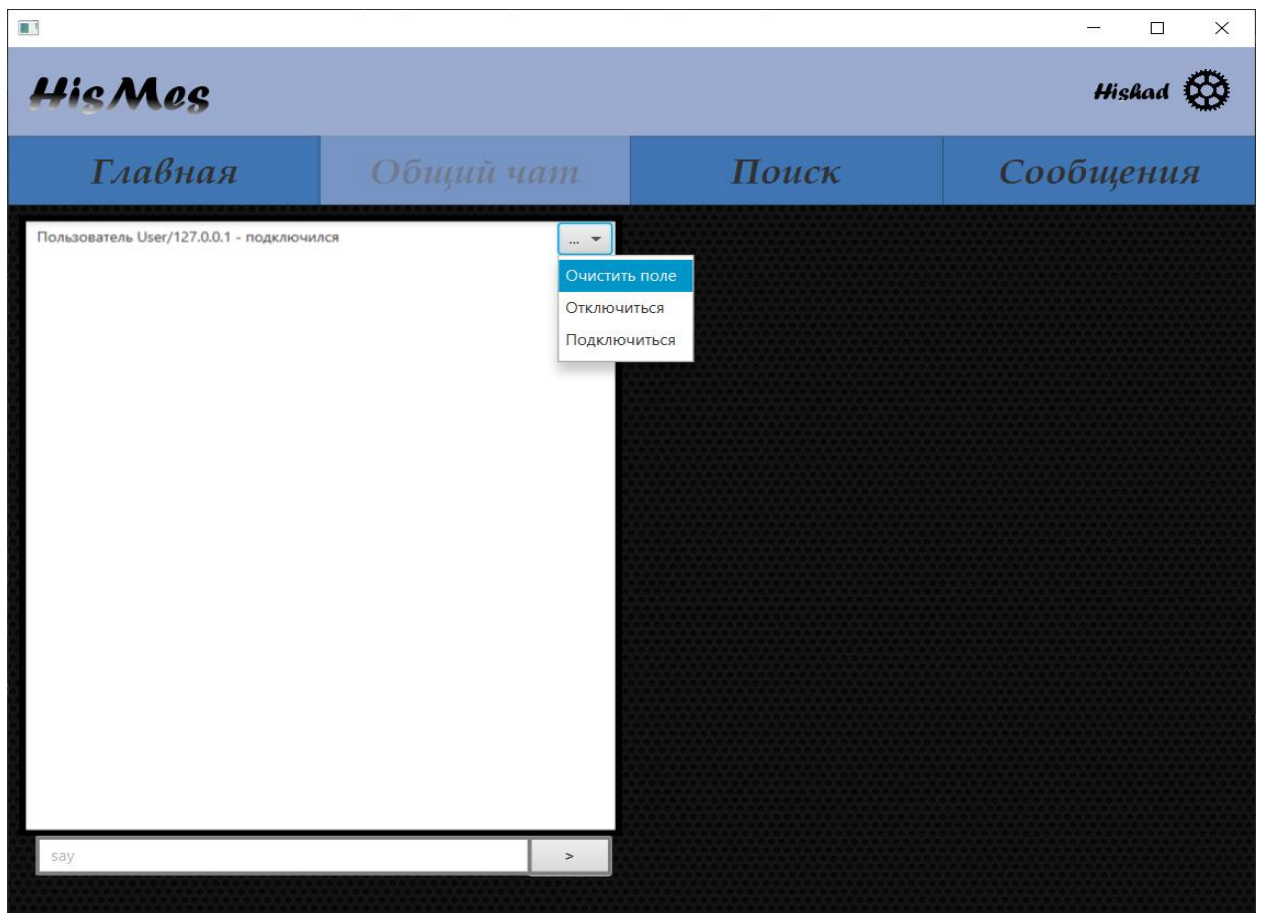


Рис. 3.2.5. Кнопки общего чата

Кнопка меню «Поиск» дает возможность искать пользователей в списке всех зарегистрированных пользователей (рис 3.2.6). Так же присутствует возможность расширенного поиска (по имени, по фамилии, по стране). По нажатию кнопки «Дополнительные параметры поиска» функция расширенного поиска станет доступной (рис 3.2.7). Пример поиска пользователя (рис 3.2.8).

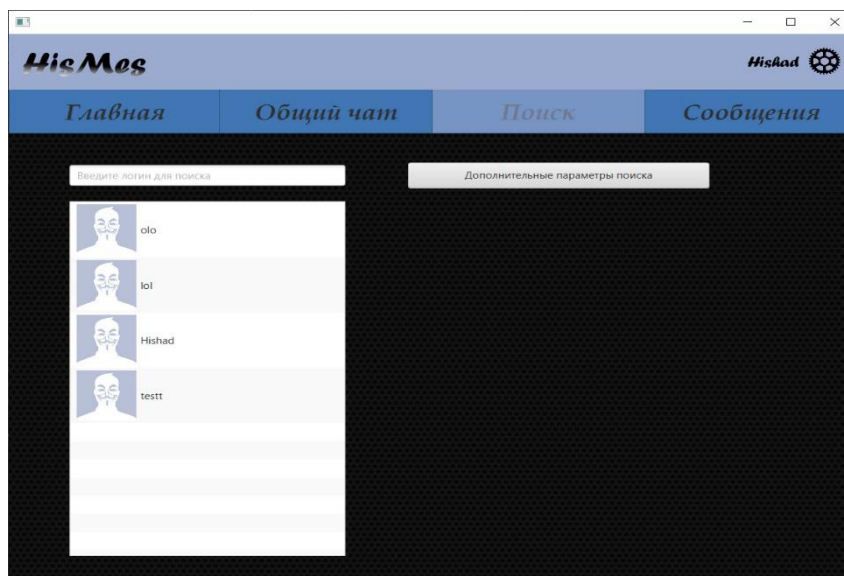


Рис. 3.2.6. Кнопка «Поиск»

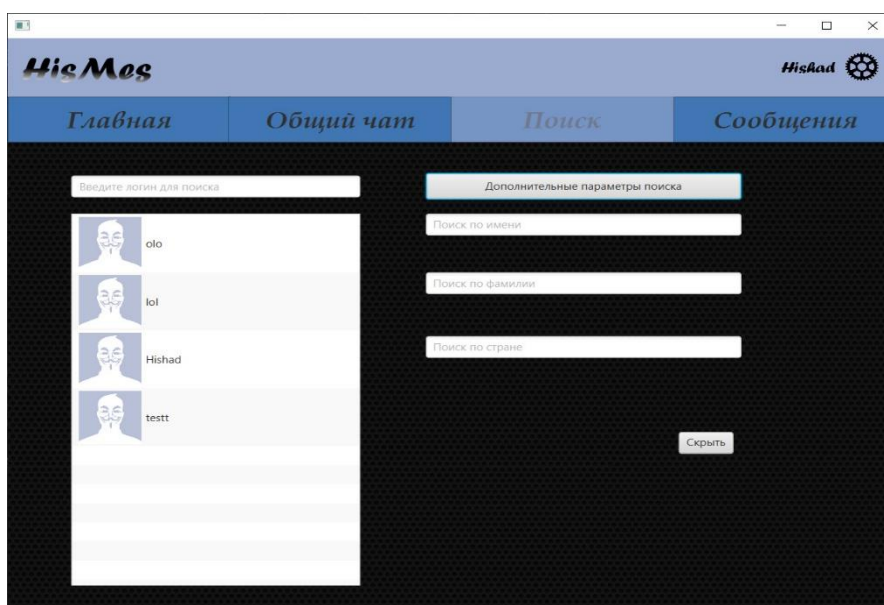


Рис. 3.2.7. Расширенный поиск

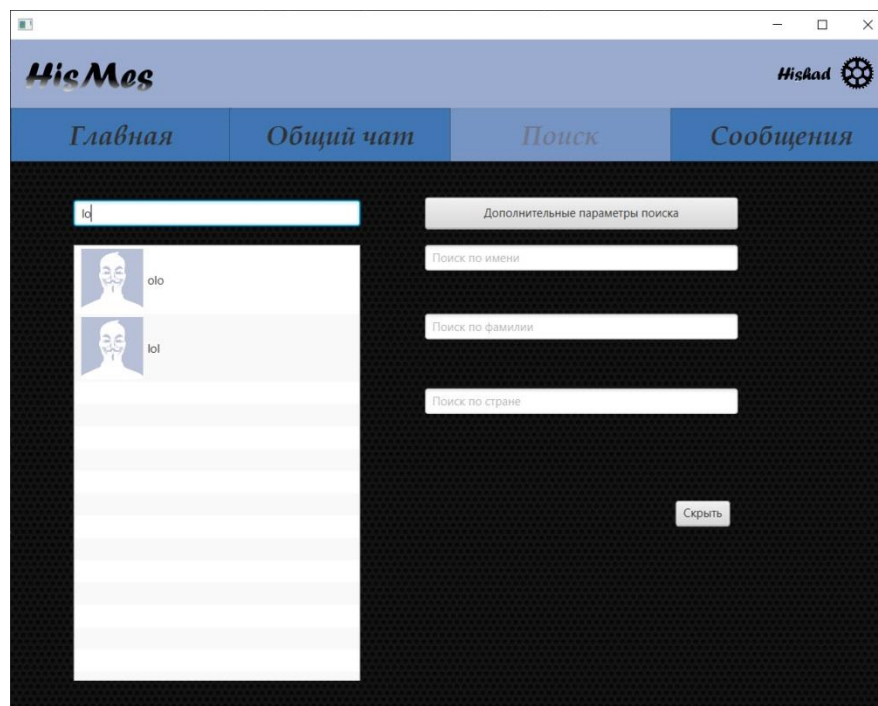


Рис. 3.2.8. Пример поиска пользователя.

При выборе пользователя отобразится его профиль с возможностью написать личное сообщение. Кнопкой «назад» происходит возврат в меню поиска (рис 3.2.9).

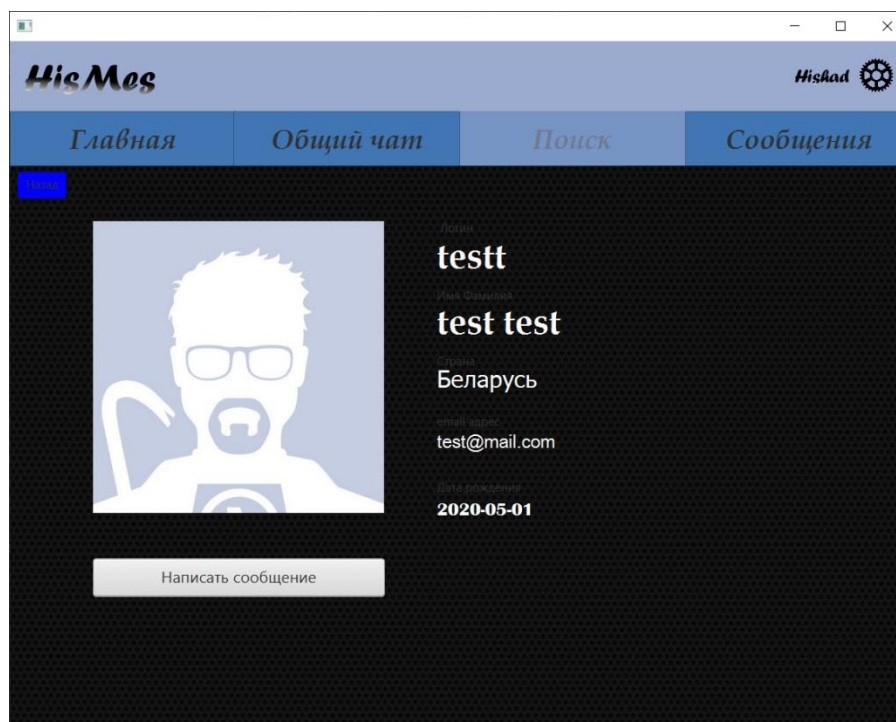


Рис. 3.2.9. Профиль пользователя.

При нажатии кнопки «Написать сообщение» происходит переход в окно диалога (рис 3.2.10). Отображается логин пользователя с которым ведется переписка, а так же сам диалог.

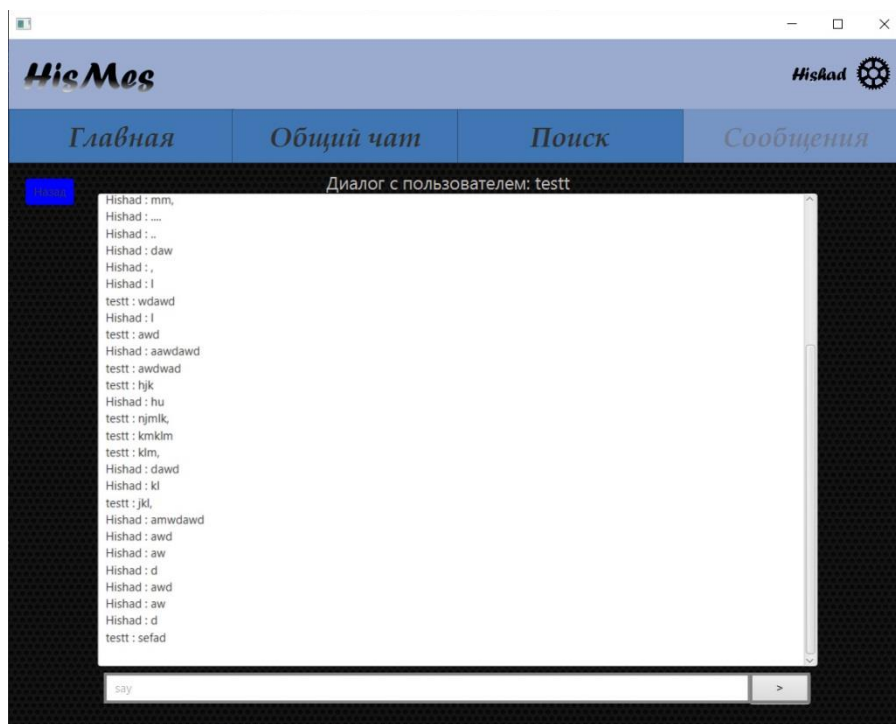


Рис. 3.2.10. Управление пользователями.

При нажатии на логотип «HisMes» откроется профиль текущего пользователя.

Важно! При переходе из меню «Общий чат» и «Сообщения» происходит отключения от сервера обмена сообщениями.

3.3. Выводы по главе 3

Таким образом, поставленные задачи к программе и разработанный алгоритм были осуществлены при помощи знаний языка высокого уровня java. В ходе выполнения работы было реализовано приложение.

В данной главе была описана структура проекта, а именно содержимое заголовочных и исходных файлов и их назначение. Разобраны пути взаимодействия пользователя с приложением на основе личного опыта. Наглядно показаны функции приложения.

ЗАКЛЮЧЕНИЕ

Результатом данной работы является полноценное приложение для общения пользователей между собой. Приложение позволяет общаться в общем чате со всеми пользователями приложения, общаться в личном чате с определенным пользователем, производить поиск пользователей, просматривать профиль пользователей. Разработан простой и удобный пользовательский интерфейс. В ходе разработки данного приложения была изучена предметная область данной темы, а именно проведена исследовательская работа по изучению основных подходов к разработке приложения. Оценены основные способы воплощения данного проекта в готовое решение. Использованы технологии, такие как язык программирования java и графический интерфейс javafx.

На основе данного заключения можно сказать, что данное приложение будет эффективным помощником пользователям, которые хотят общаться с друзьями и не только. Все поставленные задачи были выполнены и проектирование приложения прошло успешно.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Head First Java, Изучаем Java", Кэти Сьерра, Берт Бэйтс.
2. Разработка алгоритмов с использованием принципов ООП. А.А. Калинин
3. <https://metanit.com/sql/mysql/>
4. <https://www.mamp.info/en/mamp/>
5. <https://docs.microsoft.com/>