МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

**Кафедра современных технологий программирования**

ЛАТОШ ЕВГЕНИЙ АЛЕКСАНДРОВИЧ

**Клиент-серверное приложение "интернет-мессенджер"**

Курсовая работа

по дисциплине «Компьютерные системы и сети»

студента 2 курса специальности

1-40 01 01 «Программное обеспечение информационных технологий»

дневной формы получения образования

|  |  |
| --- | --- |
|  | Научный руководитель  Урбан Ольга Ивановна,  старший преподователь  кафедры современных технологий программирования |
|  |  |

Гродно, 2022

#### РЕЗЮМЕ

Тема курсовой работы

**«Клиент-серверное приложение "интернет-мессенджер"»**

Работа содержит 43 страницы, 31 иллюстраций, 4 использованных источников литературы.

Ключевые слова: клиент, сервер, аккаунт, сокет, мессенджер.

Цель курсовой работы – разработка клиент-серверного приложения для обмена сообщениями, изображениями и файлами.

Объектом исследования выступают клиент-серверные приложения для обмена данными между пользователями.

Предметом исследования курсовой работы является архитектура клиент-серверных приложений.

В работе были использованы следующие методы: сравнительный анализ, моделирование, абстрагирование.

**Авторская характеристика работы.**

Программа написана на языке высокого уровня Java. В ней реализованы общие алгоритмы функционирования приложения. Основной задачей данной работы является создание законченного продукта, готового к использованию. Так же реализованные функции могут найти применение в создании новых приложений и сервисов.

#### SUMMARY

Course work topic

**«Client-server application "Internet messenger"»**

The work contains 43 pages, 31 illustrations, 4 used literature sources.

Keywords: client, server, account, socket, messenger.

The purpose of the course work is to develop a client-server application for exchanging messages, images and files.

The object of research is client-server applications for data exchange between users.

The subject of the course work is the architecture of client-server applications.

The following methods were used in the work: comparative analysis, modeling, abstraction.

**Author's characteristics of the work.**

The program is written in the Java high-level language. It implements the general algorithms for the operation of the application. The main objective of this work is to create a finished product, ready for use. The implemented functions can also be used in the creation of new applications and services.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc104810710)

[ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc104810711)

[1.1 Основные аспекты 8](#_Toc104810712)

[1.2 Обзор существующих решений 9](#_Toc104810713)

[1.2.1 Приложение Viber 9](#_Toc104810714)

[1.2.2 Приложение WhatsApp 10](#_Toc104810715)

[1.2.3 Приложение Telegram 11](#_Toc104810716)

[1.3 Вывод по главе 1 12](#_Toc104810717)

[ГЛАВА 2 ПРОЕКТИРОВАНИЕ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ "ИНТЕРНЕТ-МЕССЕНДЖЕР" 13](#_Toc104810718)

[2.1 Проектирование архитектуры клиент-серверного приложения 13](#_Toc104810719)

[2.2 Сохранение и чтение пользовательской информации 15](#_Toc104810720)

[2.3 Отправление и получение сообщений 16](#_Toc104810721)

[2.4 Создание и удаление чата 18](#_Toc104810722)

[2.5 Поиск существующих чатов по названию 19](#_Toc104810723)

[2.6 Проектирование взаимодействия с аккаунтом 19](#_Toc104810724)

[2.6.1 Проектирование авторизации 19](#_Toc104810725)

[2.6.2 Проектирование регистрации 20](#_Toc104810726)

[2.6.3 Изменение пароля 21](#_Toc104810727)

[2.6.4 Удаление аккаунта 22](#_Toc104810728)

[2.7 Вывод по главе 2 22](#_Toc104810729)

[ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ "ИНТЕРНЕТ-МЕССЕНДЖЕР" 23](#_Toc104810730)

[3.1 Сохранение и чтение пользовательской информации 23](#_Toc104810731)

[3.2 Отправление и получение сообщений 24](#_Toc104810732)

[3.3 Создание и удаление чата 29](#_Toc104810733)

[3.4 Поиск существующих чатов по названию 32](#_Toc104810734)

[3.5 Реализация взаимодействия с аккаунтом 34](#_Toc104810735)

[3.5.1 Реализация авторизации 34](#_Toc104810736)

[3.5.2 Реализация регистрации 35](#_Toc104810737)

[3.5.3 Изменение пароля 35](#_Toc104810738)

[3.5.4 Удаление аккаунта 36](#_Toc104810739)

[3.6 Демонстрация работы реализованного приложения 37](#_Toc104810740)

[3.7 Вывод по главе 3 41](#_Toc104810741)

[ЗАКЛЮЧЕНИЕ 42](#_Toc104810742)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc104810743)

# **ВВЕДЕНИЕ**

Ещё в прошлом веке трудно было представить, что в будущем человечество начнёт отходить от живого общения. Неизбежное развитие технологий коммуникации сделало людей с разных концов земного шара ближе друг к другу.

Благодаря Интернету то, как мы общаемся друг с другом, сильно изменилось. Мы просто обмениваемся фотографиями или видео с помощью приложения для обмена сообщениями. Кроме того, мы на связи с друзьями и семьей в любое время и в любом месте. Мы не просто общаемся с помощью текстовых сообщений, мы отправляем смайлики, изображения или видеофайлы, чтобы подчеркнуть конкретную ситуацию. Таким образом, изменилось не только «как», но и «что» мы сообщаем через этот канал. Это явление можно наблюдать не только в частной сфере, но и между компаниями и их клиентами.

Мессенджер незаменим для современного общения с клиентами. Как и в личной жизни, клиентам нужна немедленная помощь, независимо от того, что их беспокоит. Мессенджер — идеальный ответ. Способность мессенджера быстро реагировать на эти запросы также говорит вашим клиентам, что вы считаете их потребности и время ценными. Чем лучше они себя чувствуют, тем более лояльными они становятся.

Мессенджеры не просто предоставляют возможность комфортного общения. Они еще и являются мощными инструментами для продвижения бизнеса. Сегодня многие компании пользуются этими программами как для внутреннего общения, так и для рекламы.

Цель курсовой работы – разработка двух приложений: 1) Мобильное приложение “мессенджер”, которое выполняет роль клиента. Данное приложение позволяет получать и отправлять данные (текстовые сообщения, изображения, файлы), создавать и удалять чаты, а также производить поиск чатов по названию. Также предусмотрена возможность регистрации, авторизации и удаления аккаунта. Приложение должно иметь возможность отправлять запросы на сервер, а также получать от него ответы. 2) Приложение сервер. Данное приложение организует хранение и доступ к данным, а также занимается получением и обработкой клиентских запросов с последующей отправкой ответа клиенту.

Предусмотрено решение следующих задач:

1) Разработать архитектуру клиент-серверного приложения

2) Организовать сохранение и чтение пользовательской информации

3) Реализовать отправку и получение сообщений

4) Реализовать создание и удаление чата

5) Реализовать поиск существующих чатов по названию

6) Реализовать взаимодействие с аккаунтом (авторизацию, регистрацию, изменение пароля, удаление аккаунта)

# **ГЛАВА 1**

# **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **Основные аспекты**

Развитие Интернет-ресурсов дало толчок появлению социальных сетей, мессенджеров, приложений. Со временем появился мобильный интернет, который сделал возможной работу в сети Интернет прямо с мобильного телефона. Все функции и ресурсы стали продуцироваться и на мобильной версии. Все это повлияло на развитие индустрии в сфере IT-технологий.

Появление мобильных мессенджеров обусловлено развитием различных мобильных приложений для смартфонов и гаджетов. Мессенджеры дают возможность не только обмениваться мгновенной информацией. Некоторые программы предоставляют возможность звонить на мобильные телефоны в любую точку мира, необходимым является только наличие сети. На данный момент мессенджеры предусматривают возможность обмена текстовыми файлами, звонками, аудио- или видеофайлами, документами и многое другое.

Используя мессенджер для общения с клиентом, вы экономите финансовые, человеческие и временные ресурсы. Во-первых, использование мессенджеров совершенно бесплатно. Во-вторых, вместо пяти человек на телефоне вам будет достаточно одного (максимум двоих) специалистов, которые будут отвечать на сообщения клиентов. Вы также экономите время, поскольку телефонный разговор может затянуться на бесконечное количество времени, а сообщения предполагают лаконичный разговор по делу.

Мессенджер совмещает режим реального времени и возможность отложенного ответа. С одной стороны, вы получаете сообщение от клиента мгновенно, с другой — можете ответить через некоторое время, не потеряв потенциального покупателя.

Мобильные мессенджеры как канал общения постепенно заменяют все другие - телефонные звонки, SMS, электронную почту и даже социальные сети. Популярность сервисов мгновенных сообщений объясняется активным развитием мобильной инфраструктуры, удешевлением самих мобильных устройств и Интернет-соединения для пользователей. Конкурируя между собой, мессенджеры вынуждены “наращивать” свой функционал, предлагать пользователям новые услуги и возможности.

## **1.2 Обзор существующих решений**

Их разработано большое количество. Предназначенные для разных операционных систем, с разной функциональностью и задачами, платные и бесплатные – вариантов достаточно, чтобы выбрать программу по душе.

Существуют приложения, которые рассчитаны на применение внутри локальной сети компании. Такие помогают сотрудникам общаться между собой, совместно работать над проектами и задачами. Есть программы, которые функционируют на определенном сайте – онлайн-консультанты. Наибольшую популярность и применение получили сервисы, доступные широкому кругу пользователей.

### **1.2.1 Приложение Viber**

Этот мессенджер делит первое место по уровню использования с программой WhatsApp. Работает с конца 2010 года, установлен на 90% смартфонов. Количество его клиентов почти достигло миллиарда человек. С его помощью можно писать текстовые сообщения, организовывать общение групп участников, звонить, совершать видео звонки абонентам по всему миру. Пересылка файлов, фотографий, интерактивных картинок, наклеек, геолокация – функции, присутствующие в приложении. Из последних новинок – возможность создать секретный чат, в котором сообщение нельзя переадресовать другим. А спустя некоторое время, оно удалится само.

В чате можно оценивать комментарии, выбрать администратора, настроить разные графические интерфейсы. Программа может работать как со смартфона, планшета, так и на компьютере. Установка и использование программы – бесплатное. Контакты для общения из телефонной книги подтянутся автоматически.

Основным недостатком Viber пользователи считают большое количество рекламы. Она приходит в виде сообщений от всевозможных служб такси, магазинов, ресторанов и других компаний. В этом сервисе, чтобы прислать что-то, не нужно запрашивать согласие получателя.

### **1.2.2 Приложение WhatsApp**

Самый популярный и один из первых мировых мессенджеров – им пользуются более миллиарда людей. Последние годы популярность падает из-за появления новых программ с расширенной функциональностью. За его использование бралась плата в размере 1 $ в год, с 2017 года – стал бесплатным. Разработан в 2010 году в США. Приложение устанавливается на популярные операционные системы, может использоваться как онлайн-сервис.

**Основные возможности, доступные в WhatsApp:**

* общение при помощи сообщений или звонков (через приложение);
* групповые чаты;
* определение и показ местоположения абонента;
* интеграция контактов из телефонной книги;
* пересылка контактных данных другим пользователям;
* персональная настройка оформления интерфейса;
* настраиваемые уведомления;
* пересылка истории переписки по электронной почте;
* отправка и получение фото.

Из недостатков можно отметить то, что один аккаунт устанавливается только на одном мобильном устройстве (дополнительно – десктоп-приложение), так как он связан с номером телефона. По отзывам пользователей, синхронизация между двумя версиями не всегда работает корректно.

### **1.2.3 Приложение Telegram**

Разработан Павлом Дуровым – российским бизнесменом и программистом. Лидирует по количеству клиентов в странах Ближнего Востока, Южной Америке, некоторых европейских странах. Для регистрации потребуется номер мобильного телефона.

**Основные возможности Тelegram:**

* Общаться при помощи текстовых сообщений и голосовых звонков.
* Обмениваться с другими пользователями файлами разных форматов.
* Участвовать в групповых чатах – до 200 участников и супергруппах – до 10 тысяч участников.
* Засекреченные чаты – информация удаляется спустя время и не сохраняется нигде.
* Пересылать надиктованные голосовые сообщения.
* Создавать и просматривать каналы – записанное или онлайн содержимое на разные темы – политика, финансы, мода, образование и другие.
* Встроенный медиа плеер.
* Хранить файлы (неограниченного объема и количества) в облачном сервере.
* Пользоваться ботами – специальными программами, которые могут выполнять самые разные действия – конвертацию файлов из одного формата в другой, поиск информации, прогноз погоды и прочие функции. Можно как применять уже существующих, так и создать своего.

Известен Telegram также и как защищенный мессенджер – пользовательская переписка закрыта даже от государственных служб. Информация не предоставляется по запросам, а изъять её силой не получится – сервера расположены в разных частях света и одновременный доступ к ним невозможен. Без этого сообщения полностью не восстановить. Это и послужило причиной запрета мессенджера в некоторых странах.

## **1.3 Вывод по главе 1**

В последнее время мессенджеры стали не только способом обмена текстовыми сообщениями или звонками, но и центром интерактивной связи. Возможность бесплатного обмена информацией, мобильность и скорость передачи информации, удобство общения с друзьями и коллегами в групповом чате способствует активной популяризации мессенджеров во всем мире.

Приложения, рассмотренные в данной главе, позволили выявить основные аспекты, связанные с реализацией приложения. А именно:

1. Реализовать отправку и получение сообщений, изображений, файлов
2. Реализовать хранение сообщений и файлов
3. Реализовать создание и удаление чатов
4. Реализовать взаимодействие с аккаунтом пользователя

# **ГЛАВА 2**

# **ПРОЕКТИРОВАНИЕ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ "ИНТЕРНЕТ-МЕССЕНДЖЕР"**

## **Проектирование архитектуры клиент-серверного приложения**

Архитектура клиент/сервер — это вычислительная модель, в которой несколько компонентов работают в строго определенных ролях для связи. Сервер размещает, доставляет и управляет большей частью ресурсов и услуг, потребляемых клиентом. Этот тип архитектуры общих ресурсов имеет один или несколько клиентских компьютеров, подключенных к центральному серверу через сеть или подключение к Интернету.

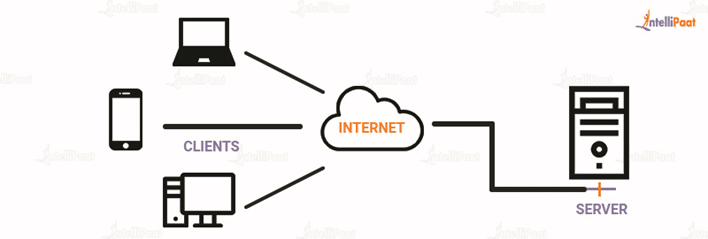


Рис. 2.1 ‒ Схема клиент-серверного приложения

В клиент-серверной архитектуре сервер действует как производитель, а клиент — как потребитель. На сервере размещаются и предоставляются высокопроизводительные, ресурсоемкие услуги клиенту по запросу. Эти услуги могут включать в себя доступ к приложениям, хранилище, совместное использование файлов, доступ к принтеру и/или прямой доступ к необработанной вычислительной мощности сервера.

Архитектура клиент/сервер работает, когда клиентский компьютер отправляет запрос ресурса или процесса на сервер по сетевому соединению, который затем обрабатывается и доставляется клиенту. Серверный компьютер может управлять несколькими клиентами одновременно, тогда как один клиент может одновременно подключаться к нескольким серверам, каждый из которых предоставляет различный набор услуг.

**Типы клиент-серверной архитектуры:**

Функциональность клиент-серверной архитектуры находится на разных уровнях.

**1-уровневая архитектура:**

Такая архитектура содержит всевозможные настройки на одном устройстве. Настройка включает настройку конфигурации, данные и маркетинговую логику. 1-уровневая архитектура является одним из надежных источников из-за служб; однако работать с такой архитектурой немного сложно, поскольку данные находятся в разных вариациях.

Все слои работают интегрировано с помощью уникального программного пакета. Данные хранятся на локальных устройствах.

**2-уровневая архитектура**

Эта архитектура имеет лучшую среду. В этой архитектуре пользовательский интерфейс хранится на стороне клиента, а база данных хранится на сервере, а логика базы данных и бизнес-логика поддерживаются либо на стороне клиента, либо на стороне сервера.

Двухуровневая архитектура быстрее по сравнению с одноуровневой архитектурой; это связано с тем, что в двухуровневой архитектуре нет посредника между клиентом и сервером. Это часто используется, чтобы избежать путаницы между клиентами. Одним из популярных примеров двухуровневой архитектуры является система онлайн-бронирования билетов.

**3-уровневая архитектура**

В отличие от двухуровневой архитектуры, в которой нет посредника, в трехуровневой клиент-серверной архитектуре между клиентом и сервером находится промежуточное программное обеспечение. Если клиент отправляет запрос на получение определенной информации с сервера, запрос сначала будет получен промежуточным программным обеспечением. Затем он будет отправлен на сервер для дальнейших действий. Тот же шаблон будет применяться, когда сервер отправляет ответ клиенту.

Благодаря наличию третьего уровня, обеспечивающего контроль данных, трехуровневая архитектура более безопасна, имеет невидимую структуру базы данных и обеспечивает целостность данных.

**N-уровневая архитектура**

N-уровневая архитектура также называется многоуровневой архитектурой. Это масштабная форма трех других типов архитектуры. В этой архитектуре предусмотрена возможность размещения каждой функции в виде изолированного уровня, который включает представление, обработку приложений и функции управления данными.

Так как для приложения "интернет-мессенджер" важна скорость при передаче сообщений между клиентами, я остановлюсь на 2-уровневой архитектуре клиент-серверного приложения.

## **Сохранение и чтение пользовательской информации**

В данном курсовом проекте для сохранения или чтения пользовательской информации будет использоваться сериализация и десериализация.

Сериализация - это процесс преобразования объекта в форму, чтобы его можно было сохранить в файле, базе данных или памяти; или он может быть передан по сети. Его основная цель - сохранить состояние объекта, чтобы его можно было воссоздать при необходимости.

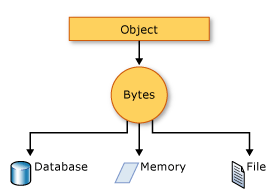


Рис. 2.2 ‒ Схема сериализации

Как следует из названия, десериализация - это процесс, обратный сериализации. Это процесс возврата сериализованного объекта, чтобы его можно было загрузить в память. Он воскрешает состояние объекта, задавая свойства, поля и т. д.

Существует несколько типов сериализации:

1. Двоичная сериализация
2. XML-сериализация
3. Сериализация JSON

Поскольку в этом приложении данные сохраняются в JSON файлах, я буду использовать тип сериализации JSON.

## **Отправление и получение сообщений**

Основная задача данного приложения — это обмен сообщениями между пользователями. Для реализации данной возможности клиентское приложение должно иметь возможность отправлять запрос о добавлении сообщения в базу данных, после чего серверное приложение оправит обновленный список сообщений каждому пользователю, подписанному на обновления данного чат.

Для реализации общения серверного приложения с клиентским приложением в языке программирования Java существует Socket.

**На стороне сервера (server-side):**

Обычно сервер, работающий на определенном компьютере, имеет серверный сокет (Server Socket), привязанный к определенному номеру порта. Серверные программы просто ждут, слушая серверный сокет для клиента, чтобы выполнить запрос соединения.

**На стороне клиента (client-side):**

Клиент знает ip-адрес машины, на котором работает сервер, и номер порта, на котором сервер прослушивает. Чтобы сделать запрос на соединение, клиент пытается встретиться с сервером на компьютере и в порту сервера. Клиент также должен идентифицировать себя с сервером, чтобы он связался с локальным портом, который будет использоваться во время этого соединения.

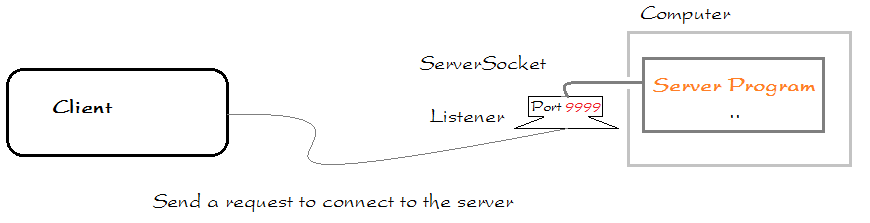


Рис. 2.3 ‒Схема подключения к серверу

Если все пройдет хорошо, серверная программа примет соединение клиента. После принятия сервер получает новый сокет, привязанный к тому же локальному порту, а также его удаленная конечная точка является адресом и портом клиента. Он создал новый сокет, чтобы заботиться о клиенте, который только что принял соединение, и мог продолжать прослушивать исходный сокет для других запросов на подключение.

После подключения клиента к серверу, можно получить входной и выходной потоки (inputstream, outputstream). После чего их можно преобразовать в ObjectInputStream и ObjectOutputStream для удобно передачи данных между клиентом и сервером.

## **Создание и удаление чата**

Другая основная функция интернет-мессенджера — это создание и удаление чата. В мобильном приложении предусмотрен экран создания чата. На данном экране находятся поле для указания названия нового чата, список существующих пользователей, для присоединения их к чату при его создании, поле для поиска пользователей по имени, а также кнопка для отправки запроса на создание чата. На данном этапе предусмотрена проверка на пустое название чата.

После нажатия на кнопку создания чата, клиентское приложение отправляет запрос на создание на сервер. Далее на серверной стороне проходит проверка на существование чата с таким названием, если чата с таким названием не существует, то создается новый чат и в него добавляются пользователи, которые были выбраны. После попытки создания чата серверное приложение отправляет результат (положительный или отрицательный) о создании на клиентское приложение.

Если клиентское приложение получает отрицательный результат при создании чата, то оно выводит соответствующую ошибку на экран. При получении положительного результат приложение переходит на основной экран навигации приложения.

Также предусмотрена возможность удаления чата. На экране отправки сообщения в чат существует кнопка отвечающая за удаление чат. При нажатии на данную кнопу открывается диалоговое окно для подтверждения удаления чата. Если удаление чата не было подтверждено, то диалоговое окно закрывается и удаление чата не происходит. В случае подтверждения удаления чата, на сервер отправляется запрос на удаление, после чего сервер удаляет чат из базы данных, а также отправляет рассылку всем пользователям в сети, подписанным на обновление данного чата, о том, что чат был удален. После чего клиентские приложения обновляют списки чатов, а также если данный чат был открыт происходит переход на основной экран навигации приложения.

## **Поиск существующих чатов по названию**

В данном приложении предусмотрена возможность присоединится к существующему чату. Для этого на основном экране навигации приложения предусмотрено кнопка поиска, при нажатии на нее приложение переходит на экран поиска существующих чатов. При переходе на дынный экран клиентское приложение отправляет запрос на сервер о получении списка всех существующих чатов. После получения списка всех чатов, он выводится на экран поиска. Также на данном экране предусмотрена возможность поиска чата по названию. При нажатии на чат, приложение переходит на основной экран навигации приложения, и пользователь подписывается на обновления этого чата, а также получает возможность отправлять сообщения в этот чат.

## **Проектирование взаимодействия с аккаунтом**

### **2.6.1 Проектирование авторизации**

При запуске программы, пользователя встречает окно авторизации. В данном окне есть два текстовых поля (для логина и пароля), кнопка, для попытки авторизации, а также кнопка для перехода на экран настроек приложения. Кроме того, при введении неверных данных предусмотрен вывод на экран соответствующих ошибок. Также существует кнопка для перехода на экран регистрации. Экран авторизации продемонстрирован на рисунке 2.4.

При нажатии на кнопку Login происходит запрос на сервер с попыткой авторизоваться. На серверном приложении происходит проверка на корректность логина и пароля. В последствии сервер оправляет ответ клиенту. Если ответ положительный, то клиентское приложение запоминает авторизованного пользователя и переходит на основной экран навигации приложения. В случае отрицательного ответа, приложение выводит соответствующую ошибку.

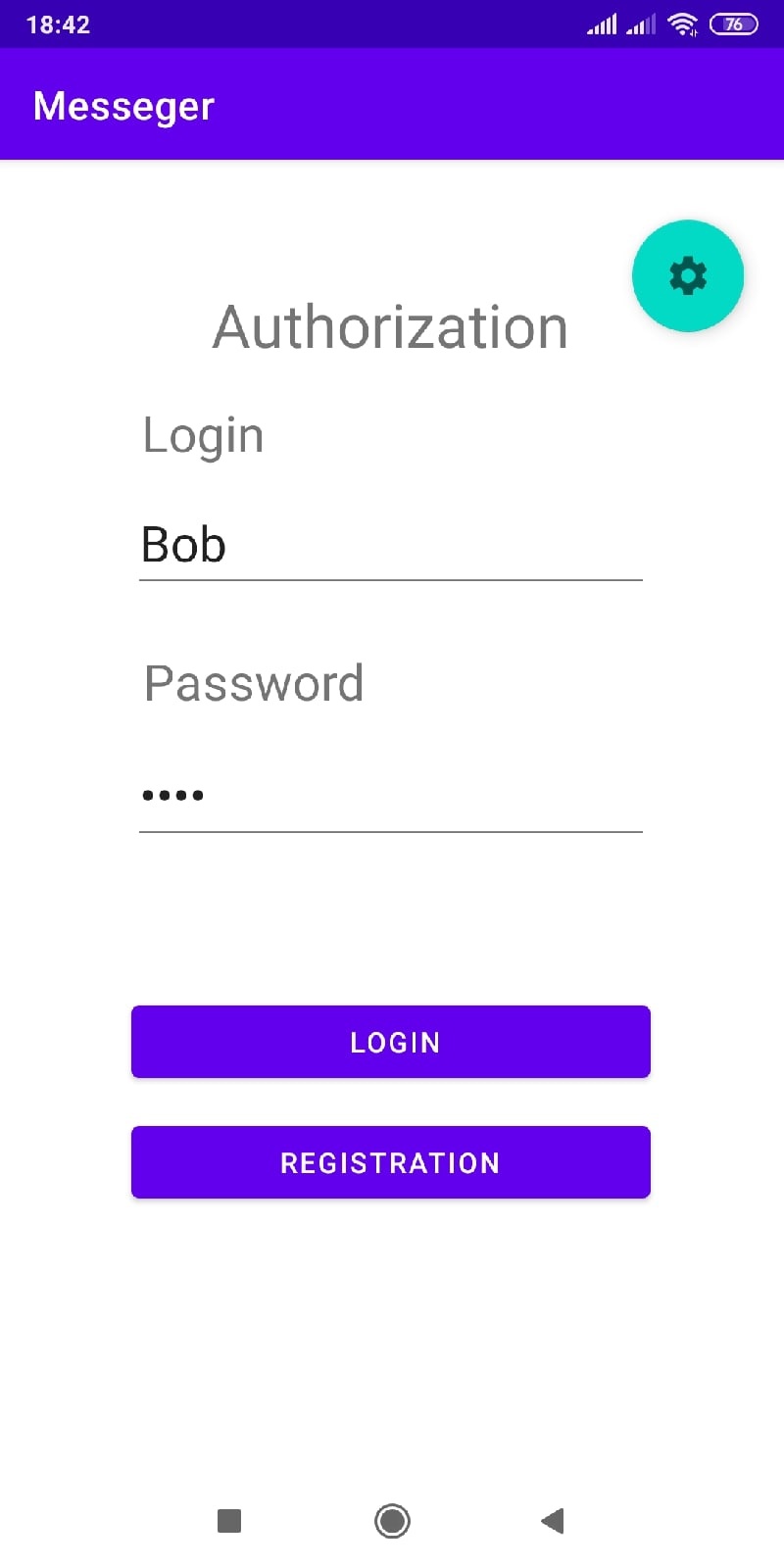


Рис. 2.4 ‒ Экран авторизации

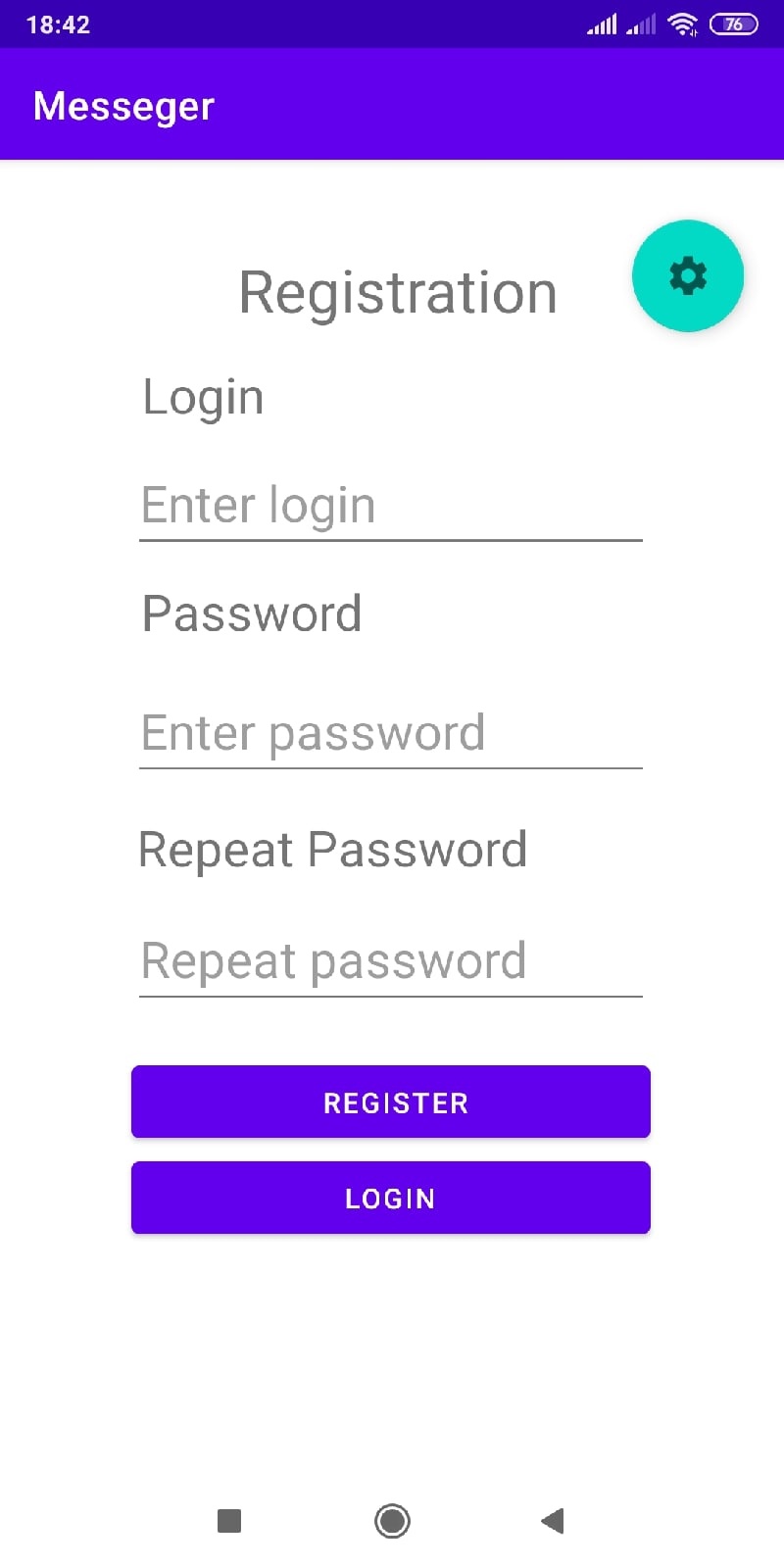


Рис. 2.5 ‒ Экран регистрации

### **2.6.2 Проектирование регистрации**

В окне регистрации есть три текстовых поля (для логина, пароля и повторного ввода пароля), кнопка, для попытки регистрации, а также кнопка для перехода на экран настроек приложения. Кроме того, при введении неверных данных предусмотрен вывод на экран соответствующих ошибок. Также существует кнопка для перехода на экран авторизации. Экран регистрации продемонстрирован на рисунке 2.5.

При нажатии на кнопку Register происходит проверка на совпадение пароля и повторного ввода пароля, если они совпадают отправляется запрос на сервер с попыткой регистрации. На серверном приложении происходит проверка на существование пользователя с таким логином. В последствии сервер оправляет ответ клиенту. Если ответ положительный, то клиентское приложение запоминает зарегистрированного пользователя и переходит на основной экран навигации приложения. В случае отрицательного ответа, приложение выводит соответствующую ошибку.

### **2.6.3 Изменение пароля**

В данном приложении предусмотрена возможность изменения пароля в аккаунте. Изменение пароля можно провести на экране настроек аккаунта. На данном экране отображается логин пользователя, а также находятся два текстовых поля (для текущего пароля и для нового пароля). Кроме того, на этом экране находятся три кнопки (для выхода, изменения пароля и удаления аккаунта). Также при введении неверных данных предусмотрен вывод на экран соответствующих ошибок.

При нажатии на кнопку изменения пароля, если строки не пустые на сервер происходит отправка запроса на изменение пароля данного пользователя. На серверном приложении проходит проверка текущего пароля данного пользователя, если пароль совпадает, то он изменяется на новый и клиентскому приложению отправляется положительный ответ. Если текущий пароль не совпадает клиентскому приложению отправляется отрицательный ответ.

Если клиентское приложение получает отрицательный ответ, оно выводит на экран соответствующую ошибку.

### **2.6.4 Удаление аккаунта**

Еще одна важная функция взаимодействия с аккаунтом — это удаление аккаунта. Удаление аккаунта также, как и изменение пароля можно провести на экране настроек аккаунта.

При нажатии на кнопку удаления аккаунта появляется диалоговое окно с подтверждением удаления. Если удаление не было подтверждено, диалоговое окно закрывается и удаления не происходит. В случае если удаление аккаунта было подтверждено, приложение отправляет на сервер запрос на удаление аккаунта и производится выход на экран авторизации. На серверном приложении из базы данных удаляются все данные пользователя и его подписки на чаты.

## **2.7 Вывод по главе 2**

Во второй главе была выбрана 2-уровневая архитектура. На основе которой будет построена архитектура клиент-серверного приложения. Также были спроектированы основные возможности приложения и выбраны способы их реализации. После того, как были спроектированы основные возможности приложения можно приступать к их реализации программным кодом.

В качестве среды разработки приложения будут использованы IntelliJ IDEA и Android Studio. Используемый язык программирования – Java.

# **ГЛАВА 3**

# **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ "ИНТЕРНЕТ-МЕССЕНДЖЕР"**

## **Сохранение и чтение пользовательской информации**

Для сохранения в JSON файл или чтения из JSON файла будет использоваться сериализация и десериализация.

Сериализация реализована в обобщенном методе public <T> void Write(T item) класса JsonWork.

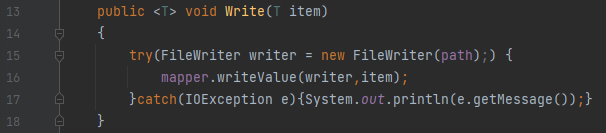


Рис. 3.1 ‒ Метод реализующий сериализацию

Данный метод принимает один параметр (Объект, который нужно сериализавать).

В строке 15 объявляется поток, при помощи которого производится запись информации в файл. Путь к файлу присваивается в конструкторе класса.

В строке 16 производится сериализация и запись объекта при помощи созданного потока.

В строке 17 производится вывод сообщения об ошибке при ее возникновении.

Десериализация реализована в обобщенном методе public <T> T Read(Class<T> type) класса JsonWork.

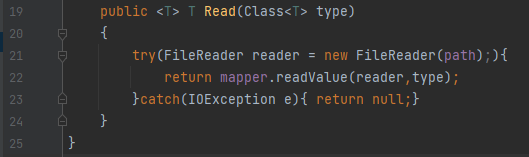


Рис. 3.2 ‒ Метод реализующий десериализацию

Данный метод принимает один параметр (Тип объекта, которого нужно десериализавать).

В строке 21 объявляется поток, при помощи которого будет производится чтение из файла. Путь к файлу присваивается в конструкторе класса.

В строке 22 производится десериализация и возвращение объекта переданного типа.

В строке 23 производится возвращение значения null при возникновении ошибки.

Хочется отметить что в рассмотренных методах созданные потоки закрываются при завершении блоков try-catch.

## **Отправление и получение сообщений**

Обмен данными между клиентским приложением и серверным приложением происходит при помощи классов ServerSocket и Socket.

На серверной стороне подключение к клиенту реализует метод public void Start() класса Server.

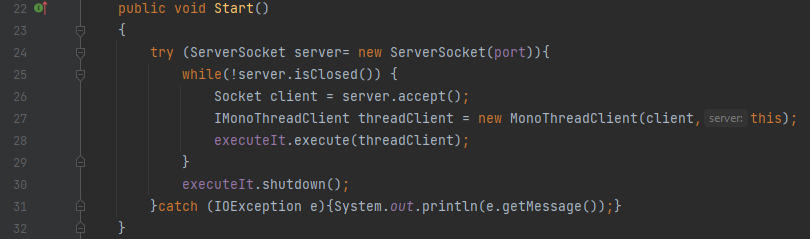


Рис. 3.3 ‒ Метод реализующий подключение к клиенту

Данный метод не принимает входных параметров и не возвращает никаких значений.

В строке 24 создается серверный сокет прослушивающий определенный порт. При завершении блоков try-catch серверный сокет закрывается. В случае возникновения ошибки в строке 31 выводится сообщение об данной ошибке.

В строке 25 создается цикл, работающий пока серверный сокет не закрыт.

В строке 26 ловится подключение нового клиента и создается сокет, через который можно обмениваться данными с этим клиентом.

В строке 27 создается объект нового класса, который работает в отдельном потоке и обрабатывает запросы данного пользователя.

В строке 28 поток, обрабатывающий запросы клиента, запускается и добавляется в список потоков.

В строке 30 после закрытия серверного сокета, закрывается добавление новых потоков в список.

На клиентской стороне подключение к серверу реализуется методом public void Connect() класса Client.

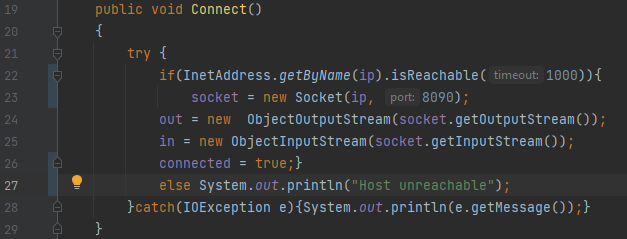


Рис. 3.4 ‒ Метод реализующий подключение к серверу

Данный метод не принимает входных параметров и не возвращает никаких значений.

В строке 22 происходит проверка доступности данного ip-адреса, в случае если данный адрес не доступен в строке 27 выводится соответствующие сообщение.

В строке 24 происходит получение выходного потока серверного сокета. А в строке 25 происходит получение входного потока серверного сокета.

В строке 26 присваивается значение true флагу, показывающему наличие подключения к серверу.

В строке 28 происходит вывод сообщения об ошибке при возникновении ошибки подключения.

Отправка запросов реализована в методе public void pushObject(Object obj) throws IOException.

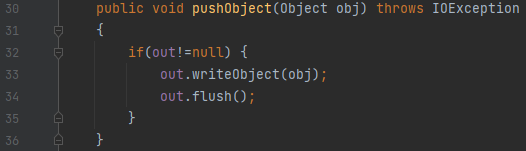


Рис. 3.5 ‒ Метод реализующий отправку запроса

Данный метод принимает один входной параметр. Также данный метод может кинуть исключение IOException.

В строке 32 происходит проверка на наличие выходного потока.

В строке 33 происходит добавление объекта в буфер. А в строке 34 происходит отправление всего буфера.

Получение запросов реализовано в методе public Object receiveObject() throws IOException, ClassNotFoundException.

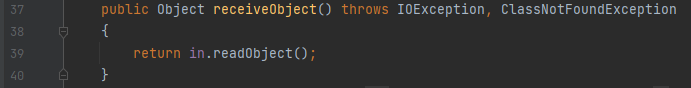


Рис. 3.6 ‒ Метод реализующий получение запроса

Данный метод не принимает входных параметров. Также данный метод может кинуть исключения IOException, ClassNotFoundException.

В строке 39 метод возвращает пришедший запрос.

Методы отправки и получения запросов аналогично для серверного и клиентского приложения.

Отправка сообщений пользователей реализована в методе public void pushMessage(Message message) класса ClientAccess.

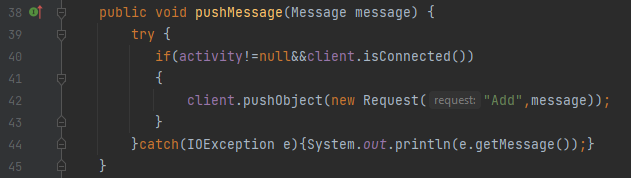


Рис. 3.7 ‒ Метод реализующий отправку сообщений

Данный метод принимает один входной параметр (Сообщение пользователя).

В строке 40 происходит проверка на подключение к серверу, а также открытие экрана чата.

В строке 42 создается соответствующий запрос и от отправляется на сервер.

В строке 44 Выводится сообщение при возникновении ошибки.

Добавление сообщения на серверной стороне реализуется в части метода public void handle(Request request)throws IOException класса RequestHandler.



Рис. 3.8 ‒ Часть метода обрабатывающая добавление нового сообщения

В строках 34-40 происходит обработка добавления нового сообщения.

В строке 35 данные запроса приводятся к типу Message. В строке 36 данное сообщение добавляется в базу данных.

В строках 37-38 происходит получение данного чата и проверка корректность.

В строке 39 происходит рассылка обновленного чата всем пользователям, подписавшимся на обновления.

Обновление сообщений на клиентской стороне реализуется в части метода public void handle(Request request) класса RequestHandler.

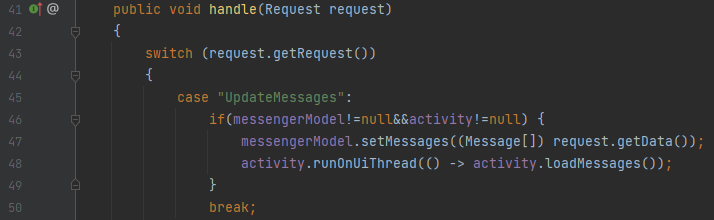


Рис. 3.9 ‒ Часть метода обрабатывающая обновление сообщений

В строках 45-50 реализуется обработка запроса обновления сообщений.

В строке 46 происходит проверка на наличие обязательных объектов для обновления сообщений.

В строке 47 обновляется список сообщений, а в строке 48 сообщения выводятся на экран.

## **Создание и удаление чата**

На клиентской стороне собираются данные чате и отправляется запрос на сервер, где происходит основная работа при создании или удалении чата. Поэтому имеет смысл рассмотреть только серверную часть этих функций.

Создание чата реализовано в методе public boolean addChat(Chat chat) класса DB.

Данный метод принимает один входной параметр (Информацию о чате, который нужно создать).

В строках 209-225 происходит проверка на ошибки, если возникает ошибка ее текст выводится на экран.

В строках 210-212 объявляются объекты для работы с файлами.

В строке 213 происходит попытка создать файл чата, если не получается создать файл чата метод возвращает отрицательное значение.

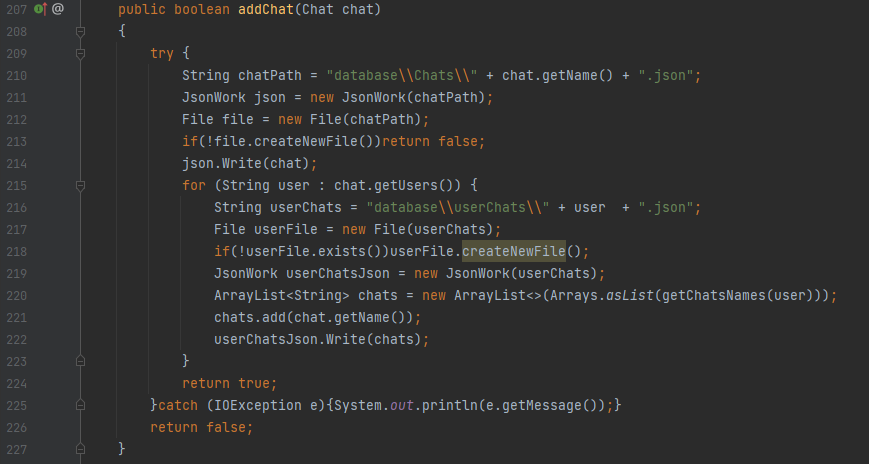


Рис. 3.10 ‒ Метод создающий новый чат

В строке 214 записывается информация о чате в созданный файл.

В строках 215-223 происходит запись нового чата для каждого пользователя, подписанного на данный чат по умолчанию.

В строках 216, 217, 219 происходит объявление объектов для работы с файлами.

В строке 218 создается файл пользователя, если он не был создан.

В строке 220 происходит получение названий всех чатов, на которые подписан данный пользователь. В строке 221 к списку добавляется новый чат и в строке 222 файл перезаписывается.

Удаление чата реализовано в методе public void DeleteChat(String chatName) класса DB.

Данный метод принимает один входной параметр (Название чата, которого надо удалить).

В строках 165-166 происходит получение чата по его названию и проверка на корректность.



Рис. 3.11 ‒ Метод удаляющий чат

В строках 167-174 реализуется удаление чата из подписок каждого пользователя, подписанного на данный чат. В строках 169-170 объявляются объекты для работы с файлами. В строке 171 происходит получение списка названий чатов. В строке 172 из списка удаляется данный чат и в строке 173 перезаписывается файл.

В строках 175-188 происходит удаление дополнительных файлов к сообщениям. В строках 178-182 если существует файл, прикрепленный к сообщению, происходит его удаление. В строках 183-187 если существует изображение, прикрепленное к сообщению, происходит его удаление.

В строках 189-190 происходит удаление файла с информацией о чате.

## **Поиск существующих чатов по названию**

При открытии окна поиска, клиентское приложение запрашивает список чатов у сервера. Далее поиск чата по названию реализуется в методе public boolean onQueryTextChange(String s) класса TextChangeHandler.



Рис. 3.12 ‒ Метод реализующий поиск

В строках 29-30 происходит получение доступа к нужным для поиска данным. И проверка на их наличие.

В строке 31 происходит получение списка всех чатов. Далее в строке 32 происходит проверка на наличие чатов. В строках 34-35 происходит выбор чатов, у которых в названии присутствует строка, введенная пользователем.

В строке 36 выбранные чаты выводятся на экран.

Подписка на чат реализована в методе public void onClick(View view, int position) класса AddChatHandler.

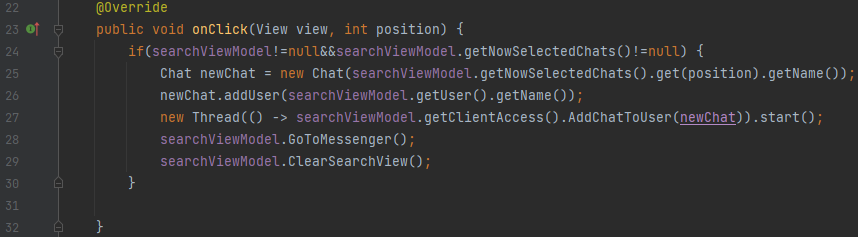


Рис. 3.13 ‒ Метод реализующий подписку на чат

В строке 24 происходит проверка на наличие необходимых объектов.

В строке 25 создается новый чат с названием выбранного чата из списка. В строке 26 в чат добавляется данный пользователь.

В строке 27 отправляется запрос на сервер о добавлении пользователя в чат.

В строке 28 производится переход на основной экран навигации приложения. И в строке 29 сбрасывается состояние элемента поиска.

На серверной стороне добавление пользователя в чат реализовано в методе public void AddChatToUser(Chat chat) класса DB.

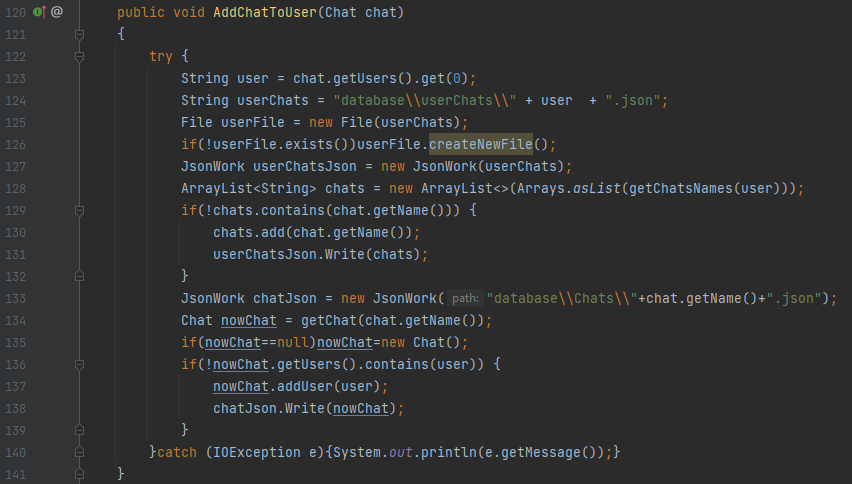


Рис. 3.14 ‒ Метод реализующий добавление пользователя в чат

В строке 123происходит получение имя пользователя.

В строках 124, 125, 127, 133 происходит объявление объектов для работы с файлами.

В строках 128-132 происходит получение списка чатов, на которые подписан данный пользователь и, если нового чата среди них нет, он добавляется в список.

В строках 134-139 происходит получение информации о данном чате и, если на него не подписан данный пользователь происходит добавление его в список.

## **Реализация взаимодействия с аккаунтом**

### **3.5.1 Реализация авторизации**

Авторизация пользователя происходит на серверном приложении в методе public boolean CheckUserPassword(User user) класса DB.

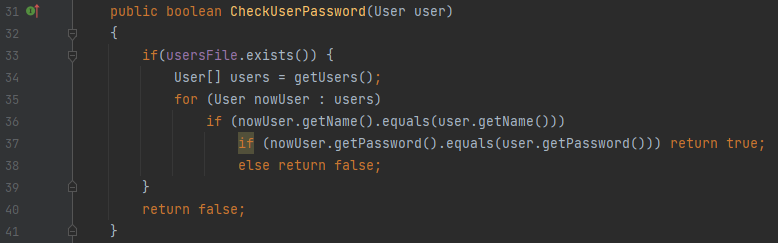


Рис. 3.15 ‒ Метод реализующий авторизацию

В строке 33 происходит проверка на наличие файла со списком пользователей. Далее если данный файл существует, в строке 34 происходит получение списка пользователей.

В строках 35-36 происходит поиск пользователя по имени. Если данный пользователь найден, в строке 37 происходит проверка на пароль. Если пароль верный авторизация возвращает положительный результат. Если же пароль не верный или данный пользователь не найден, возвращается отрицательный результат.

### **3.5.2 Реализация регистрации**

Регистрация пользователя происходит на серверном приложении в методе public boolean UserRegistration(User user) класса DB.

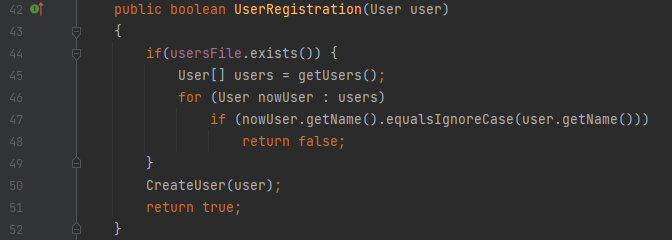


Рис. 3.16 ‒ Метод реализующий регистрацию

В строке 44 происходит проверка на наличие файла со списком пользователей. Далее если данный файл существует, в строке 45 происходит получение списка пользователей.

В строках 46-47 происходит поиск пользователя по имени. Если аккаунт с таким именем уже существует возвращается отрицательный ответ. В противном случае создается новый аккаунт и возвращается положительный ответ.

### **3.5.3 Изменение пароля**

Изменение пароля происходит на серверном приложении в методе public boolean UpdateUser(User newUser) класса DB.

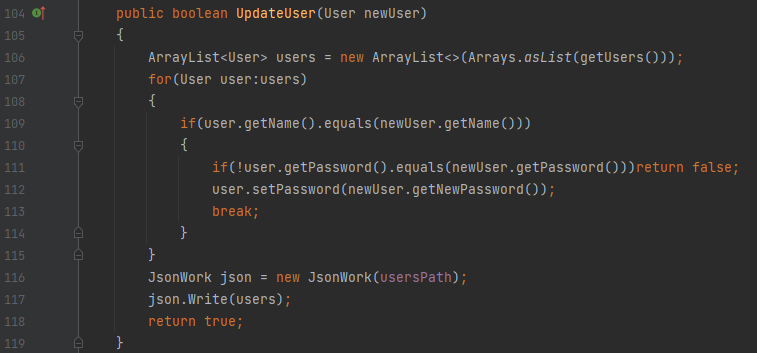


Рис. 3.17 ‒ Метод реализующий изменение пароля

В строке 106 происходит получение списка всех пользователей. Далее в строках 107-109 производится поиск пользователя по имени.

В строках 111-112 происходит проверка текущего пароля пользователя. Если пароль не совпадает возвращается отрицательное значение. В противном случае пользователю присваивается новый пароль и в строках 116-117 перезаписывается информация о пользователе. После чего возвращается положительный результат.

### **3.5.4 Удаление аккаунта**

Удаление аккаунта происходит на серверном приложении в методе public void DeleteUser(User user) класса DB.

В строках 144, 151, 159, 160 объявляются объекты для работы с файлами.

В строках 145-147 происходит получение списка пользователей, поиск и удаление данного пользователя, а также перезапись файла с информацией о пользователях.



Рис. 3.18 ‒ Метод реализующий удаление аккаунта

В строке 148 происходит получение названий всех чатов на который подписан пользователь

В строках 149-158 происходит удаление пользователя из каждого чата, на которые он был подписан.

В строке 161 удаляется файл с информацией о чатах, на которые он был подписан.

## **Демонстрация работы реализованного приложения**

При запуске программы, пользователя встречает окно авторизации. В данном окне есть два текстовых поля (для логина и пароля), кнопка (для входа в приложение), а также кнопка для перехода на экран настроек приложения. Кроме того, при введении неверных данных предусмотрен вывод на экран соответствующих ошибок. Также существует кнопка для перехода на экран регистрации.

После перехода на экран регистрации, на нем мы можем увидеть три текстовых поля (для логина, пароля и повторного ввода пароля), кнопка, для попытки регистрации, а также кнопка для перехода на экран настроек приложения. Кроме того, при введении неверных данных предусмотрен вывод на экран соответствующих ошибок. Также существует кнопка для перехода на экран авторизации.

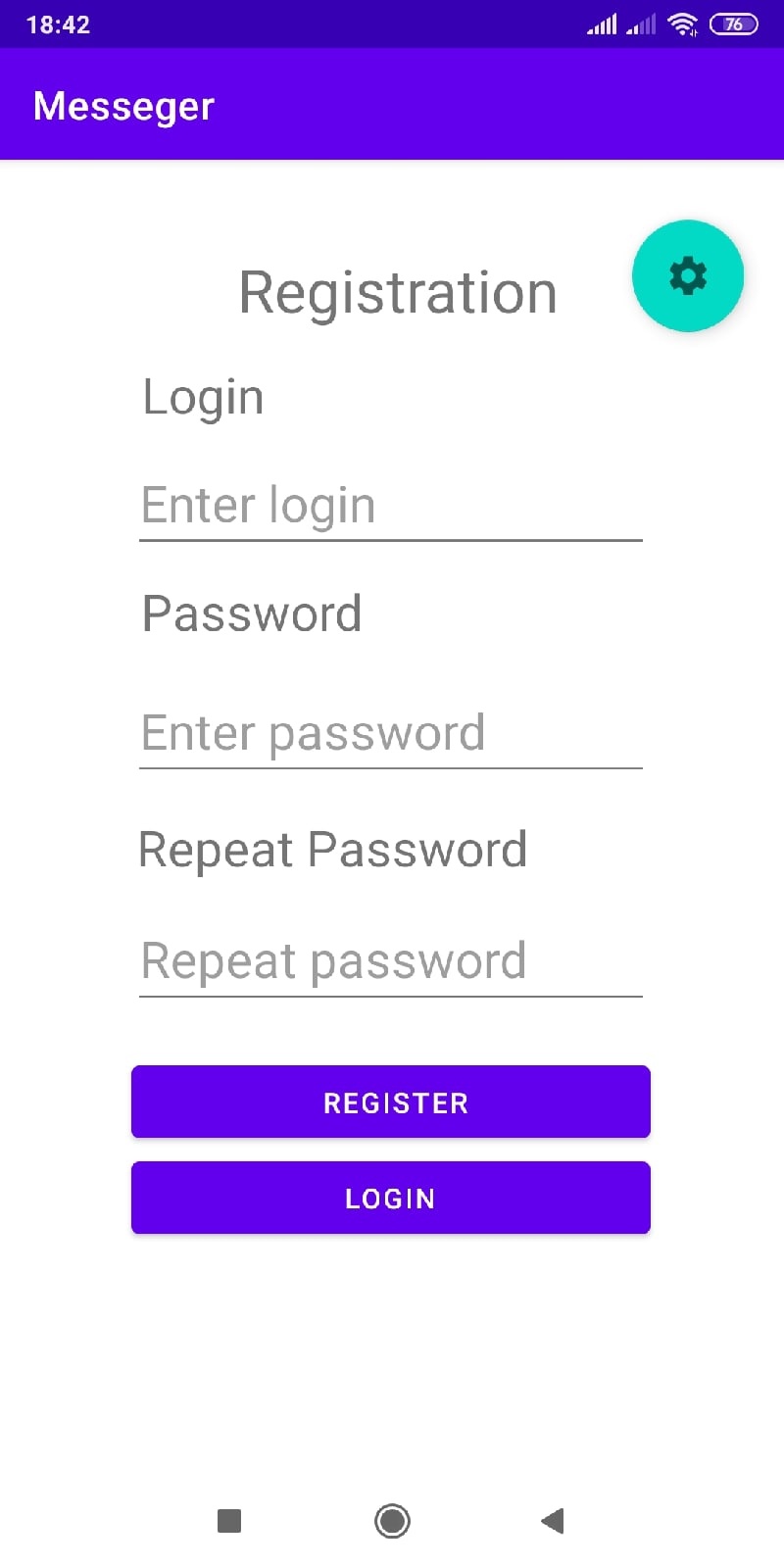


Рис. 3.20 ‒ Экран регистрации

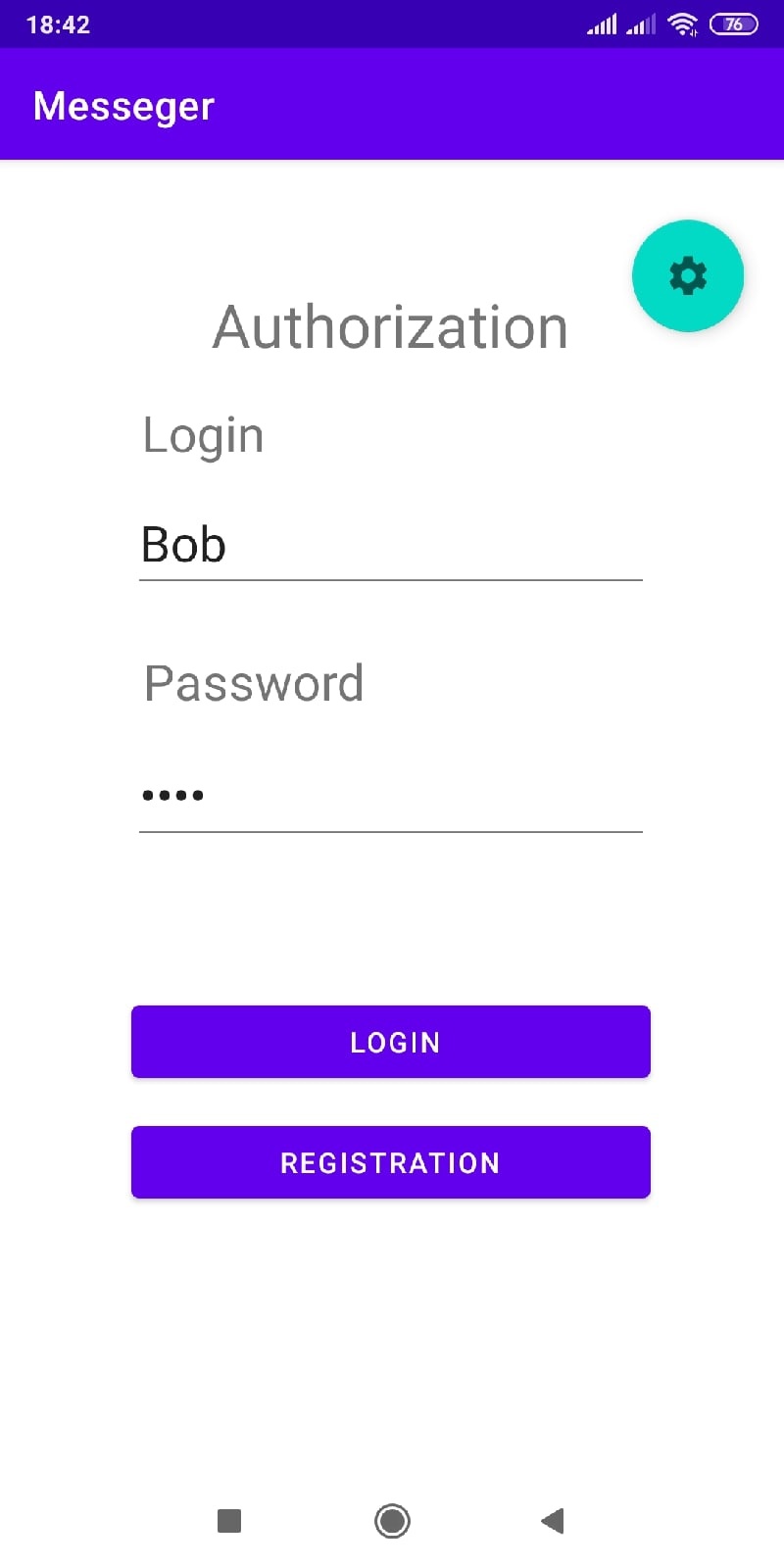


Рис. 3.19 ‒ Экран авторизации

После авторизации мы попадаем на основной экран навигации приложения. На данном экране находится список чатов, на которые подписан данный пользователь, кнопка для создания чата, а также поисковая строка, при нажатии на которую пользователь переходит на экран поиска чатов. Помимо это на данном экране находится выдвижное окно навигации по приложению, через которое можно попасть на окна настроек аккаунта и настроек приложения.

В списке чатов существует возможность выйти из чата, нажав на крестик напротив него. Кроме того, при нажатии на чат можно перейти на окно просмотра сообщений чата.

При нажатии на кнопку добавления чата происходит переход на экран создания чата. На данном экране находятся поле для указания названия нового чата, список существующих пользователей, для присоединения их к чату при его создании, поле для поиска пользователей по имени, а также кнопка для отправки запроса на создание чата. На данном этапе предусмотрена проверка на пустое название чата.

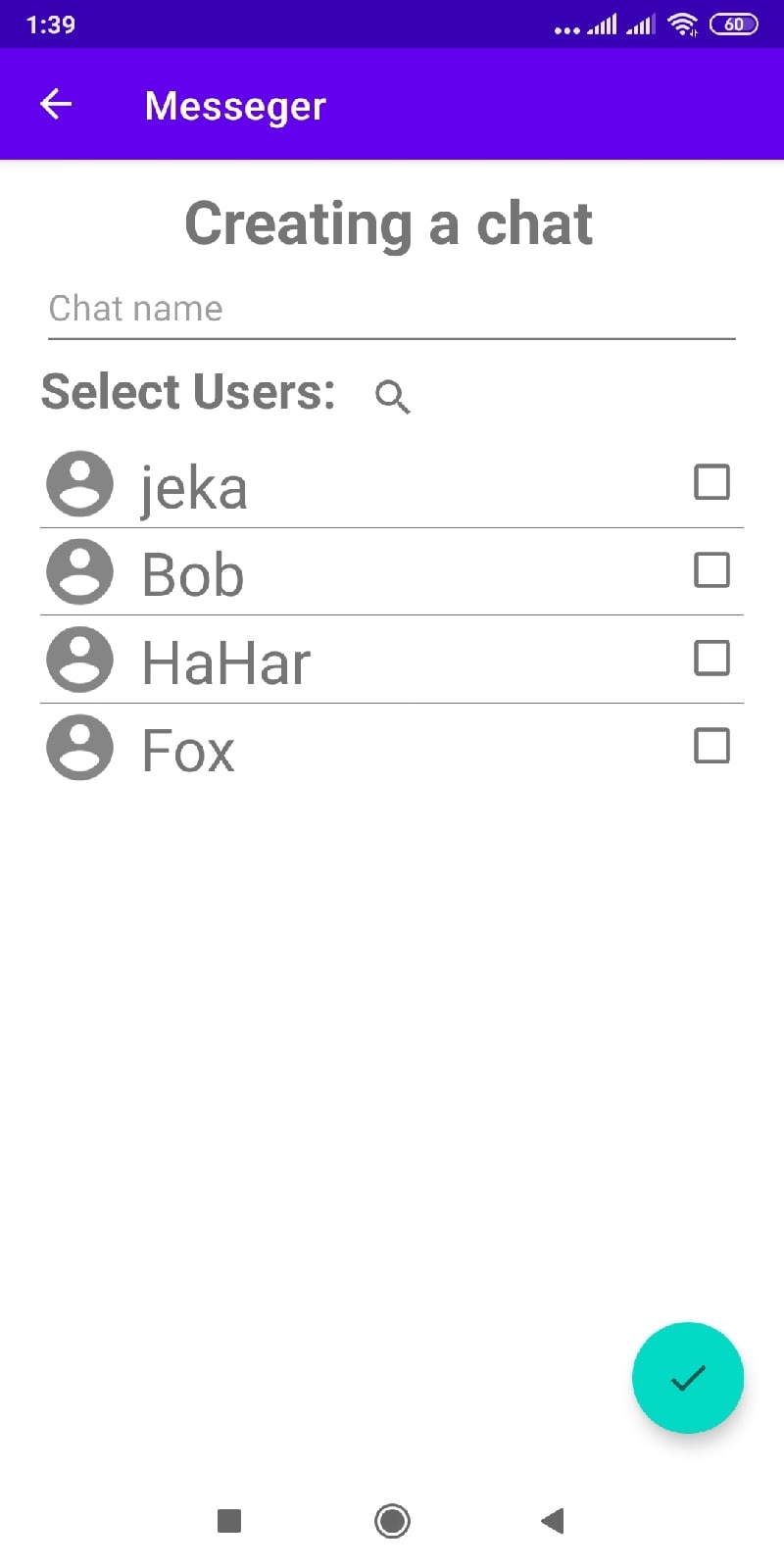


Рис. 3.22 ‒ Экран создания чата

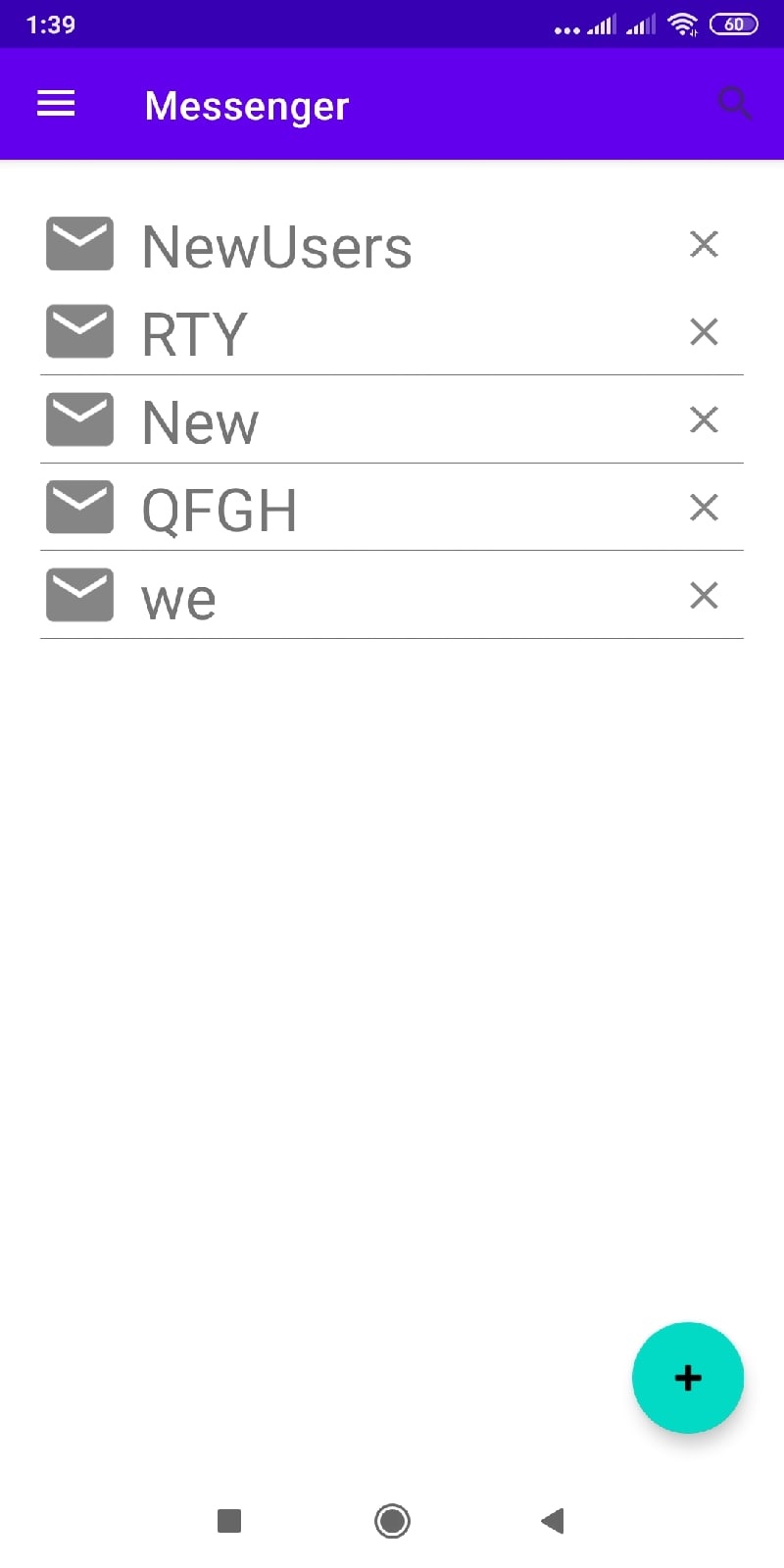


Рис. 3.21 ‒ Основной экран навигации

При помощи панели навигации можно переместится на экран настроек аккаунта. На данном экране отображается логин пользователя, а также находятся два текстовых поля (для текущего пароля и для нового пароля). Кроме того, на этом экране находятся три кнопки (для выхода, изменения пароля и удаления аккаунта). Также при введении неверных данных предусмотрен вывод на экран соответствующих ошибок.

При помощи панели навигации можно переместится на экран настроек приложения. На данном экране находится текстовое поле для ввода нового ip-адреса, кнопка для смены ip-адреса, и ползунок для переключения темы приложения между черной и белой.

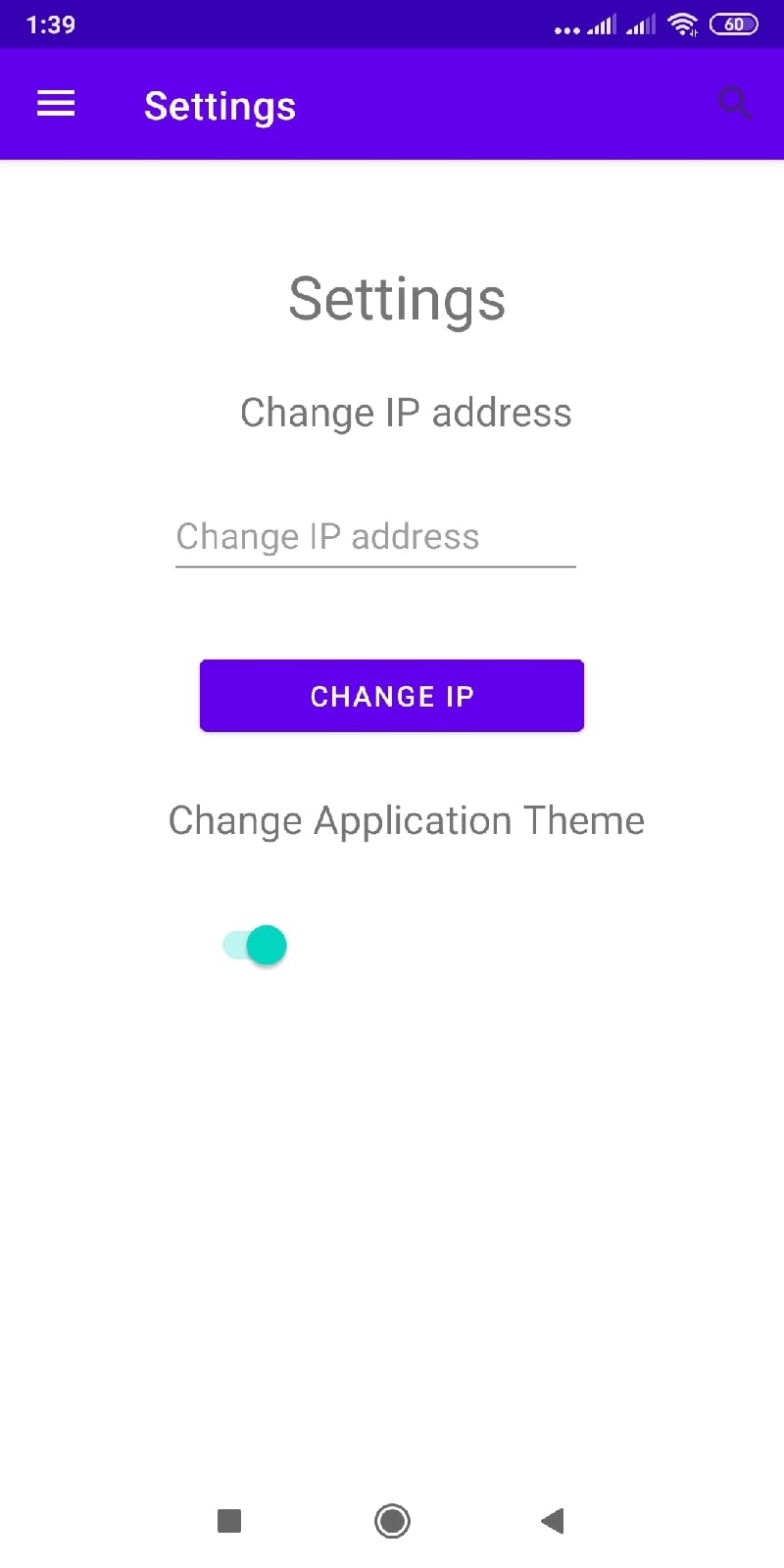


Рис. 3.24 ‒Экран настроек приложения

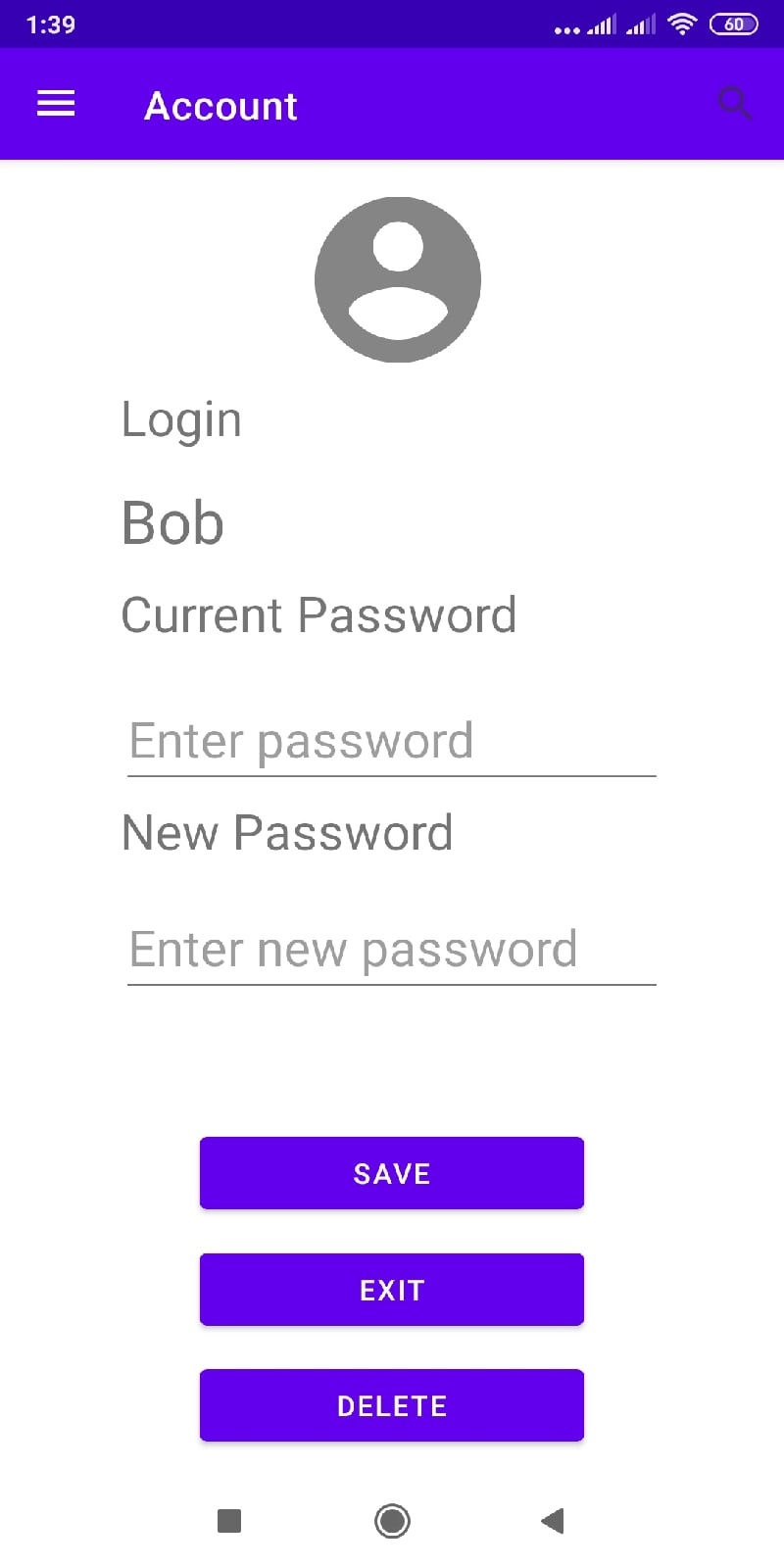


Рис. 3.23 ‒ Экран настроек аккаунта

При нажатии на поисковую строку приложение переходит на экран поиска чата по названию. На данном экране пользователь может найти чат по названию и при нажатии на интересующий его чат присоединиться к нему. Также предусмотрена возможность вернуться на предыдущий экран.

При нажатии на чат, на главном экране навигации, приложение переходит на экран чата. На данном экране пользователь может писать сообщения в чат, прикреплять к ним изображения и файлы. Также на данном экране можно удалить текущий чат. Помимо всего предусмотрена возможность вернуться на предыдущий экран.

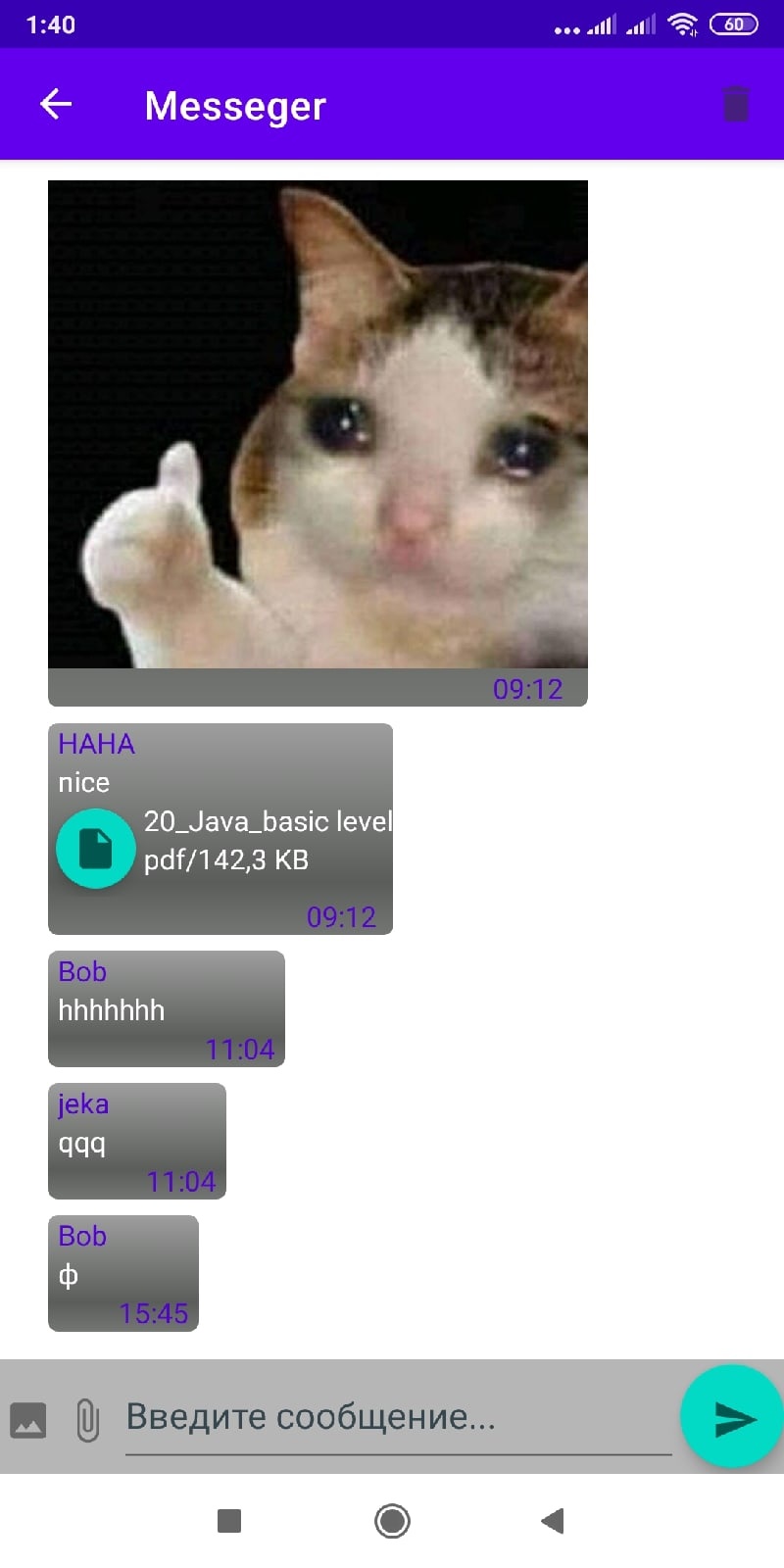


Рис. 3.26 ‒ Экран чата

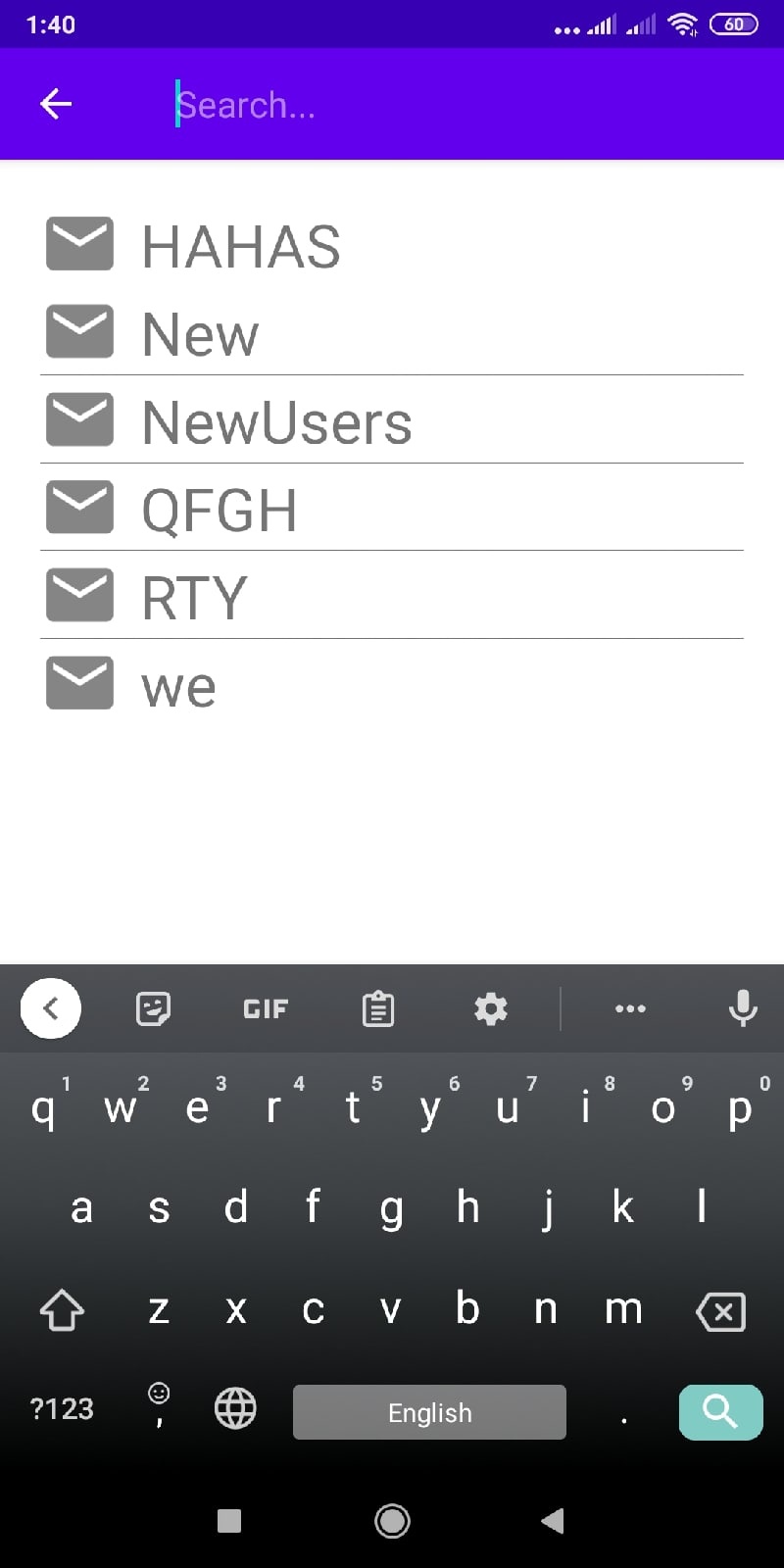


Рис. 3.25 ‒ Экран поиска чата

## **Вывод по главе 3**

Таким образом мы разобрались с функционалом приложения и его реализацией. В данной главе была описана реализация основных методов данного проекта.

В будущем, проект может представлять, как практическую, так и академическую пользу. Поскольку в случае его удачной реализации, его можно будет развивать дальше, а конечная версия, написанная в максимально простом виде, может в дальнейшем использоваться как шаблон, которым смогут воспользоваться все желающие для того, чтобы ознакомиться с архитектурой приложений на конкретном примере.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была изучена клиент-серверная архитектура и в результате был выбран подходящий для данного приложения тип архитектуры. На этапе разработки программы были получены практические навыки создания клиент-серверного приложения. Благодаря использованию дополнительной учебной литературы, особых трудностей при написании работы не возникло.

Дальнейшую модификацию программы вижу в возможности реализации приватных чатов, передачу аудио сообщений, выделение и пересылку сообщений и т.д.

В первой главе курсовой работы была доказана актуальность выбранной темы. Также были рассмотрены существующие решения и были выявлены основные аспекты, связанные с реализацией приложения.

Во второй главе была спроектирована архитектура приложения. А также были спроектированы основные методы, использованные в курсовой работе.

В третьей главе была описана программная реализация основных методов, использованных в курсовой работе.

В заключении хочется отметить, что данное приложение можно будет развивать дальше, добавляя новы возможности. Кроме того, реализованные методы могут быть применены не только в данном проекте, их можно использовать в различных приложениях. Также данный проект может использоваться в качестве примера для ознакомления с архитектурой клиент-серверных приложений.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Мурат Йенер, Алекс Фидом Java ЕЕ. Паттерны проектирования для профессионалов. - СПб.: Питер, 2016. - 240 с.: ил.
2. intellipaat.com [Электронный ресурс]: What is Client Server Architecture? -URL: <https://intellipaat.com/blog/what-is-client-server-architecture/> (дата обращения: 07.03.2022)
3. betacode.net [Электронный ресурс]: Руководство Программирование Java Socket. -URL: <https://betacode.net/10393/java-socket> (дата обращения: 13.03.2022)
4. semantica.in [Электронный ресурс]: Что такое мессенджер. -URL: <https://semantica.in/blog/chto-takoe-messendzher.html> (дата обращения: 02.03.2022)