МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра прикладной математики**

**КУРСОВАЯ РАБОТА**

**КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ НА ANDROID STUDIO**

Работу выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Т.Е. Кулиш

(подпись, дата) (инициалы,фамилия)

Факультет компьютерных технологий и прикладной математики курс 3

Направление 02.03.03 Математическое обеспечение и администрирование информационных систем

Научный руководитель,

д.физ.-мат.н., профессор\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.-Х.Х. Уртенов

(подпись, дата) (инициалы, фамилия)

Нормоконтролер,

к.ф.м.н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Г.В. Калайдина

(подпись, дата) (инициалы, фамилия)

Краснодар 2018

СОДЕРЖАНИЕ

Введение 3

1 Архитектура клиент-сервер 5

1.1 Описание предметной области 5

1.2 Актуальность разработки клиент-серверного приложения 7

2 Разработка клиент-серверного приложения на Android Studio 8

2.1 Техническая часть написания клиент-серверного приложения 8

2.2 Реализация серверной части 13

2.3 Оформление страниц и реализация клиентской части 14

Заключение 22

Список использованных источников 23

ВВЕДЕНИЕ

В наши дни активно развивается сфера информационных технологий, благодаря чему расширяется область её применения в человеческой деятельности. Значительную часть современных информационных технологий составляют сетевые технологии и их более глобальное проявление – Internet-технологии.

Одной из наиболее активно развивающихся областей компьютерных технологий, является автоматизация обработки информации на основе программно-вычислительных комплексов информационных систем, которые пользуются большими объемами данных, хранящимися во внешней памяти. Такие хранилища имеют сложную структуру, в основании которой лежит такой объект, как таблица. Автоматизация практически всех видов деятельности строится по принципу реляционной базы данных, что позволяет сократить финансовые и ресурсоемкие затраты, что, безусловно, является актуальной задачей в настоящее время.

Клиент-сервер – это вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемые серверами, и заказчиками услуг, называемыми клиентами.

 Клиент-серверная информационная система состоит из трех основных компонентов:

* Сервер баз данных, управляющий хранением данных, доступом и защитой, резервным копированием, отслеживающий целостность данных и выполняющий запросы клиента;
* Клиент, выполняющий логику приложения, посылающий запросы к серверу и получающий ответы от него;
* Сеть и коммуникационное программное обеспечение, которое осуществляет взаимодействие между клиентом и сервером посредством сетевых протоколов;

Рассматривая вопрос о том, какие преимущества имеет архитектура клиент-сервер, стоит сказать, что создание таких приложений позволяет добиться ряда важнейших факторов:

* Сохранность информации;
* Устойчивость к сбоям;
* Масштабируемость;
* Большая защищенность информации от несанкционированного доступа;

Цель курсовой работы – разработка клиент-серверного приложения на Android Studio.

Задача курсовой работы – изучение современных технологий разработки клиент-серверного приложения.

Курсовая работа состоит из двух глав, введения, заключения и списка используемой литературы.

В первой главе представлено описания предметной области, а также актуальность разработки клиент-серверного приложения на Android Studio.

Вторая глава повествует о технической разработки клиент-серверного приложения. В данной главе представлены основные аспекты взаимодействия установленного сервера с разработанным приложением, которое предоставляет возможность передавать сообщения между пользователями. Можно ознакомиться с реализацией серверной и клиентской частями.

Разработанное клиент-серверное приложение предоставляет возможность общения между пользователями, сообщения которых хранятся в базе данных. Данный проект позволяет ознакомится с технологией интернет-мессенджера.

В заключении подведены основные итоги написания курсовой работы, а также представлен результат.

1 Архитектура клиент-сервер

* 1. Описание предметной области

Эффективность функционирования информационной системы во многом зависит от ее архитектуры.

В настоящее время перспективной является архитектура клиент-сервер.

Система клиент-сервер – это гармоничная композиция трех отдельных технологий, работающих в неразрывной связке, чтобы обеспечить эффективное хранение и быстрый доступ к данным. Рассмотрим более подробно о взаимосвязанных компонентов.

Программное обеспечение на компьютере клиенте, так называемое фронтальное ПО, отвечает за экран и ввод-вывод информации пользователем. Программное обеспечение на сервере несет ответственность за обработку введенной информации и доступ к дискам данных. К примеру, пользователь на машине-клиенте создает запрос данных в базе, фронтальная программа посылает данный запрос через сеть к серверу. Сервер базы данных проводит поиск, и отправляет обратно соответствующие запросу данные.

В достаточно распространенном варианте она предполагает наличие компьютерной сети и распределенной базы данных.

Компьютерная сеть – это множество компьютеров, соединенных линиями связи и работающих под управлением специального программного обеспечения. Под линией связи обычно понимают совокупность технических устройств, и физической среды, обеспечивающих передачу сигналов от передатчика к приемнику.

Распределенная база данных -– набор логически связанных между собой разделяемых данных, которые физически распределены в некоторой компьютерной сети.

Распределенная СУБД (система управления базами данных) – программный комплекс, предназначенный для управления распределенными базами данных и позволяющий сделать информацию прозрачной для конечного пользователя.

Сервером определенного ресурса в компьютерной сети называется компьютер, управляющий этим ресурсом, клиентом - компьютер (программа), использующий этот ресурс. В качестве ресурса компьютерной сети могут выступать: базы данных, файловые системы, службы печати, почтовые службы. Тип сервера определяется видом ресурса, которым он управляет. Например, если управляемым ресурсом является база данных, то соответствующий сервер называется сервером базы данных.

Структура распределенной информационной системы, которая построенна по архитектуре клиент-сервер с использованием сервера баз данных, обеспечивает выполнение основного объема обработки данных. Формируемые пользователем или приложением запросы поступают к серверу базы данных в виде инструкции языка SQL. Сервер базы данных выполняет поиск и извлечение нужных данных, которые затем передаются на компьютер пользователя.

Важнейшим достоинством применения базы данных в информационной системе является обеспечение независимости данных от прикладных программ, это дает возможность пользователям не заниматься проблемами представления данных на физическом уровне: размещение данных в памяти, методов доступа к ним.

Такая независимость достигается поддерживаемым СУБД многоуровневым представлением информации в базе данных на логическом (пользовательском) и физическом уровнях. Благодаря системе управления базами данных и наличию логического уровня его представления обеспечивается отделение понятийной модели базы данных от ее физического представления в памяти ЭВМ. Важнейшим параметром крупной информационной системы является быстродействие при значительном количестве пользователей, а также надежность, масштабируемость и безопасность. Всё это обеспечивает архитектура клиент-сервер.

Представленная архитектура позволяет оптимально распределить работу между клиентскими и серверной частями системы. Приложение, работающее на рабочей станции, не читает записи базы данных напрямую, а посылает запросы на сервер, где они принимаются и последовательно отрабатываются специальными программами. В результате на рабочую станцию поступают только обработанные данные, что радикально сокращает информационные потоки в локально вычислительную сеть.

* 1. Актуальность разработки клиент-серверного приложения

На сегодняшний день технология клиент-сервер получает все большее распространение. Она дает общее представление о том, как должна быть организована современная распределенная информационная система, и в то же время реализации этой технологии в конкретных программных продуктах и даже в видах программного обеспечения различаются весьма существенно.

Одним из важнейших преимуществ клиент-серверных информационных систем является снижение сетевого трафика при выполнении запросов.

Вторым преимуществом архитектуры клиент-сервер является возможность хранения бизнес-правил на сервере, что позволяет избежать дублирования кода в различных приложениях, использующих общую базу данных. Кроме того, в этом случае любое редактирование данных, в том числе и редактирование нештатными средствами, может быть произведено только в рамках этих правил.

Помимо перечисленных возможностей, современные серверные СУБД обладают широкими возможностями управления пользовательскими привилегиями и правами доступа к различным объектам базы данных, резервного копирования и архивации данных, а нередко и оптимизации выполнения запросов. Они также, как правило, предоставляют возможность параллельной обработки данных, особенно в случае использования многопроцессорных компьютеров в качестве сервера баз данных.

2 Разработка клиент-серверного приложения на Android Studio

2.1 Техническая часть написания клиент-серверного приложения

В начале разработки клиент-серверного приложения необходимо ознакомиться с языками программирования и базами данных, которые будут использоваться в данном проекте. Основные языки программирования, которые задействованы в нашей программе, являются Java, PHP (Hypertext Preprocessor) и язык запросов SQL. В данный момент PHP и SQL активно поддерживают значительное большинство хостинг-провайдеров.

В данном проекте для хранения сообщений используем SQL БД как на сервере, так и на устройствах-клиентах. Вся работа клиентов интернет-мессенджеров сводится к постоянной синхронизации локальной и удаленной БД с сообщениями. Реализация клиентской части осуществлена на языке Java, серверная часть – на языке PHP, а взаимодействие между ними будет происходить путем HTTP запросов и JSON ответов. Рассмотрим более подробную информацию.

Технология Java – это объектно-ориентированная, многопоточная среда программирования.

При компиляции программного продукта, написанный на языке программирования Java, получается байт-код. Виртуальная машина Java может интерпретировать этот байт-код на любой платформе, на которой установлена виртуальная машина Java. Это означает, что нет необходимости в портировании программ.

Задачам распределенных систем клиент-сервер отвечает объектно-ориентированная парадигма: использование концепций инкапсуляции, наследования и полиморфизма. Java предоставляет ясную и действенную платформу разработки.

Платформа Java разработана для создания высоконадежного прикладного программного обеспечения. Большое внимание уделено проверке программ на этапе компиляции, за которой следует второй уровень - динамическая проверка (на этапе выполнения).

Java разработана для оперирования в распределенных средах, это означает, что на первом плане должны стоять вопросы безопасности. В сетевых средах приложения, написанные на Java, защищены от вторжения неавторизованного кода, пытающегося внедрить вирус или разрушить файловую систему.

Далее познакомимся с JSON (JavaScript Object Notation), предоставляющий простой формат обмена данными, удобен для чтения и написания как человеком, так и компьютером. JSON является полностью независимым от языков реализации, но использование соглашения, знакомые программистам таких языков, как C, C++, C#, Java, JavaScript, Perl, Python и многих других. Эти свойства делают JSON идеальным языком обмена данными.

JSON основан на двух структурах данных:

* коллекция пар ключ-значение. В разных языках, эта концепция реализована как объект, запись, структура, словарь, хэш, именованный список или ассоциативный массив;
* упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность.

В разработанном клиент-серверном приложение, основные запросы и функции реализованы на языке Java. Далее мы сможем ознакомиться с файлами, которые осуществляют непосредственно отправку сообщений, а также предупреждают пользователя при успешном результате выполнения команд или же некорректной работе приложения или сервера. Для начала кратко изучим за что отвечают представленные файлы: MainActivity.java, ChatActivity.java, FoneService.java, AutoRun.

В фоновом режиме работает FoneService.java, который, в отдельном потоке, каждые 15 секунд отправляет запрос на сервер. Если ответ сервера содержит новые сообщения, то данный файл записывает их в локальную базу данных, для хранения переданного сообщения между указанными пользователями, и отправляет данный текст в файл ChatActivity.java о необходимости обновить ListView, параметр которого отвечает за поток сообщений. Обновление содержимого данного параметра позволяет собеседникам в диалоге, между определенными пользователями, видеть переданную информацию.

Отправка нового сообщения из ChatActivity.java происходит сразу на сервер, но переданная информация не записывается в локальную БД. Запись осуществляется только лишь после получения его назад в виде ответа сервера.

Такая реализация обусловлена тем, что работа интернет-мессенджера обязательно должна группировать сообщения по времени. Учитывая, что клиентские приложения не могут быть синхронизированы с точным временем, а возможно даже работа будет проходить в разных часовых поясах, то следует использовать время сервера. Это позволяет не нарушать последовательность переданной информации.

Создавая новое сообщение, на сервер передается: имя адресанта, имя получателя и отправленный текст. Получая эти данные обратно, в виде ответа сервера, получаем еще четвертый параметр, который отвечает за время записи сообщения в локальной БД.

Для наглядности в файл MainActivity.java была добавлена возможность удаления сообщений из локальной БД. Данная процедура отправляет на сервер запрос на получение всех сообщений выбранного чата, что обеспечивает чистую установку приложения. Так же есть возможность послать запрос на удаление всех сообщений из БД, расположенной на сервере.

Далее рассмотрим, как происходит взаимодействие между клиентом и сервером.

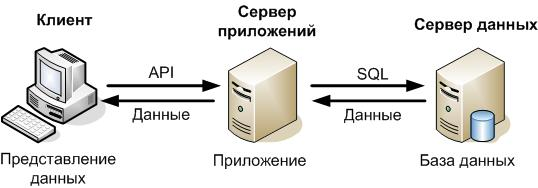


Рисунок 1 – Архитектура клиент – сервер

Как видно из рисунка 1, клиент, отправляя сообщение другому пользователю, передает данные на сервер, который играет связывающую роль между объектами передачи информации. Сервер накапливает сообщения, для возможности восстановления в случае удаления на одном из клиентских устройств.

Для реализации сервера необходимой частью является выбор хостинг-провайдера, услуги которого являются наиболее удобными. Рассмотрев различные предложения компаний, в большей степени соответствует потребностям и предложенному обслуживанию крупнейший хостинг-провайдер TimeWeb. Для получения доменного имени и адреса хоста необходимо создать базу данных, как представлено на рисунке 2.

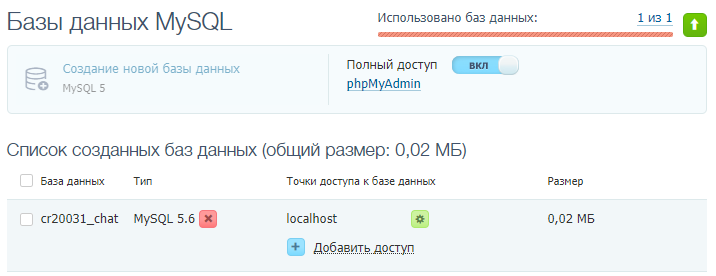


Рисунок 2 – Обзор базы данных

Основные идеи современной информационной технологии базируются на концепции, согласно которой данные должны быть организованы в базы данных. Они создаются и функционируют под управлением специальных программных комплексов, называемых системами управления базами данных (СУБД). Рассмотрим более подробно.

База данных – это организованное хранение информационных ресурсов, которые представляются в виде интегрированной совокупности файлов, обеспечивающей удобное взаимодействие и быстрый доступ к данным. Значения преобразуются при изменении состояния отражаемой предметной областью, под которой понимаются объекты, процессы, которые могут быть представлены в базе данных в полном информационном объеме.

Информация, хранимая в базе, генерируется в единую целостную структуру, что обеспечивает более эффективную работу пользователей с большим объемом данных.

Система управления базой данных (СУБД) – это программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ.

На рисунке 3 представлено размещение полученных данных на web-сервере.

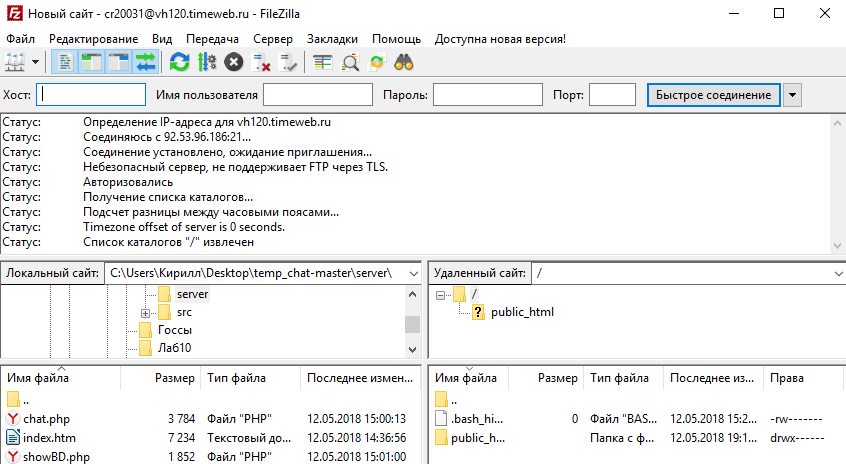


Рисунок 3 – Обзор FTP-клиента(FileZilla)

Хостинг-провайдер предоставляет возможность создания таблицы chat с помощью SQL запроса, представленный на рисунке 4.

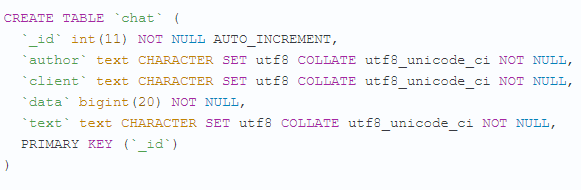


Рисунок 4 – SQL запрос

2.2 Реализация серверной части

Для передачи данных между пользователями, необходимо создать файл Chat.php, который отправляет информационный текст в базу данных.

Фрагмент, отвечающий за считывание переданных параметров, показан на рисунке 5.

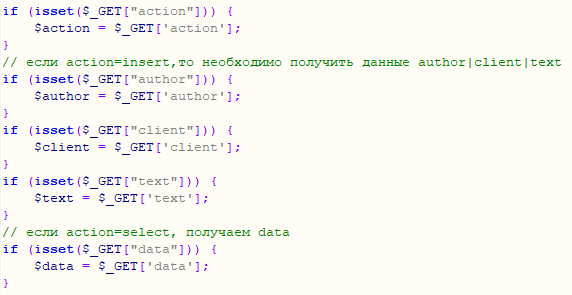


Рисунок 5 – Фрагмент кода из файла Сhat.php

Время передачи сообщения, которое обозначено параметром data, берется текущее на сервере.

Теперь можно преступить непосредственно к разработке клиент-серверного приложения, который предоставляет возможность передавать сообщения между пользователями, сохраняя информационный текст в базе данных. Принцип данного проекта заключается в создание простого интернет-мессенджера.

2.3 Оформление страниц и реализация клиентской части

Макет приложения предназначен для удобного и технически простого использования интернет-мессенджера. Пользователь при запуске программы должен легко ориентироваться с открытым интерфейсом.

В данном проекте пользователю представлена возможность выбора собеседника, удобного перемещения в диалог, с выбранным клиентом, а также удаления всей информации с данного устройства и базы данных, которая располагается на сервере.

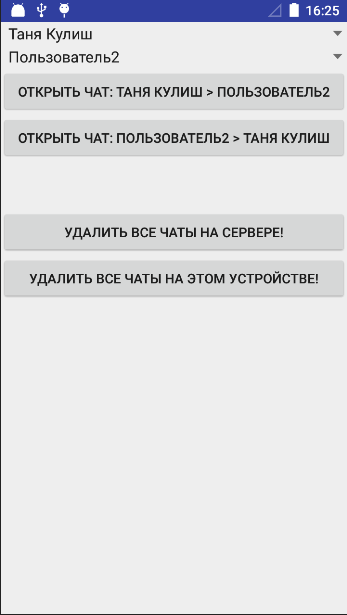


Рисунок 6 – Главная страница интернет-мессенджера.

Ознакомившись с рисунком 6, который демонстрирует главную страницу интернет-мессенджера, можно рассмотреть более подробно реализацию представленных кнопок, выпадающих списков с выбором собеседника и перехода непосредственно к личному диалогу между пользователями.

В данной программе параметры author и client отвечают за пользователей, показывающих направление передачи информации. Первый параметр выполняет роль адресанта, второй­­ – получателя. Если же значения имени пользователя совпадут, то есть author = client, то приложение выдаст ошибку и выбранный нами чат не будет открыт. Реализацию данного аспекта можно увидеть на рисунке 7.

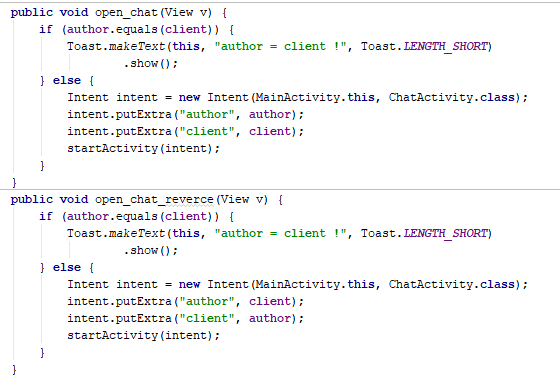


Рисунок 7 – Фрагмент кода из файла MainActivity

Представленные процедуры open\_chat, open\_chat\_reverce, считывающие значения параметров author и client, выполняют проверку на несовпадение имен пользователей и записывают их на кнопки, для перехода в чат. Иначе, при совпадении имен, доступ перехода в диалог запрещен.

При выборе чата с данными именами пользователей, переходим из MainActivity в ChatActivity. Для начала необходимо проверить введенный текст на пустоту (рисунок 8). Если текст отсутствует, то кнопка отправки, значение которой передается в параметр bt, блокируется.

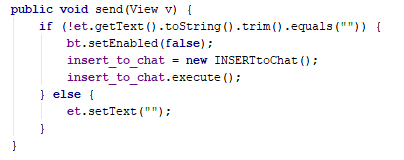


Рисунок 8 – Фрагмент кода из файла ChatActivity

Процедура, представленная на рисунке 9, обновляет значения параметра lv, который выполняет роль полосы сообщений, считывая информацию из базы данных. Успешное выполнение данного действия позволяет вывести переданное сообщение на экран, указывая дату отправки. Обновление полей чата происходит автоматически.

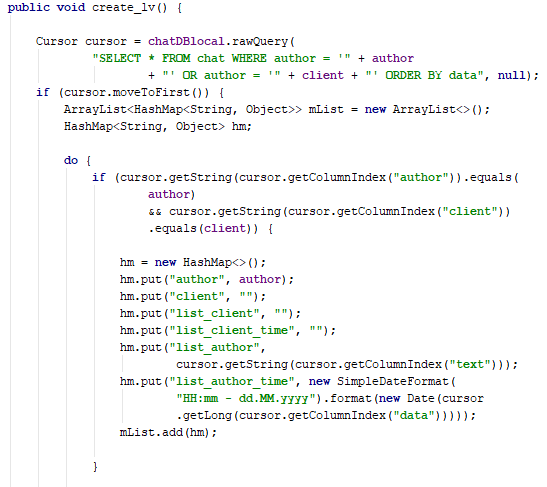


Рисунок 9 – Фрагмент кода из файла ChatActivity

В случае, если пользователь решиться удалить сообщения на данном устройстве, необходимо нажать кнопку «Удалить все чаты на данном устройстве». Это позволит вызвать процедуру delete\_local\_chats, представление которой рассмотрено на рисунке 10.

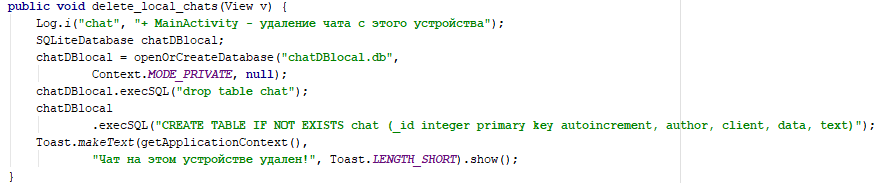


Рисунок 10 – Фрагмент кода из файла MainActivity

Данный метод удаляет локальную таблицу чатов, создавая при этом новую c параметрами: id, author, client, data, text. Также реализовано удаление таблицы чатов на сервере.

Немало важной частью данного проекта является вывод сообщений о возможных ошибках, которые могут возникать при отправке сообщений между пользователями, некорректной работы приложения, неправильных запросов, а также при отсутствии соединения на сервере. Но не стоит забывать о том, что пользователь хочет убедиться в правильности обработки запросов приложения. Рассмотрим более подробно о каждом из возможных предупреждений.

За корректность передачи информации между пользователями отвечает процедуры onPostExecute, которая активирует кнопку отправки для дальнейшей передачи информации, в случае успешного результата выполнения запроса. Сообщение об ошибке передачи данных предупреждает адресанта о том, что собеседник не сможет получить требуемую информацию. Реализация данной процедуры представлена на рисунке 11.

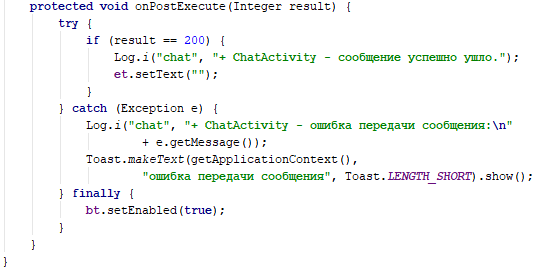


Рисунок 11 – Фрагмент кода из файла ChatActivity

Одной из частых проблем может служить сбой передачи информации на сервер. Такие ситуации могут возникнуть при отсутствии интернета или закрытом доступе сервера. Пользователь должен знать, что служит некорректной работе используемого приложения. Данную проверку осуществляет процедура doInBackground (рисунок 12).

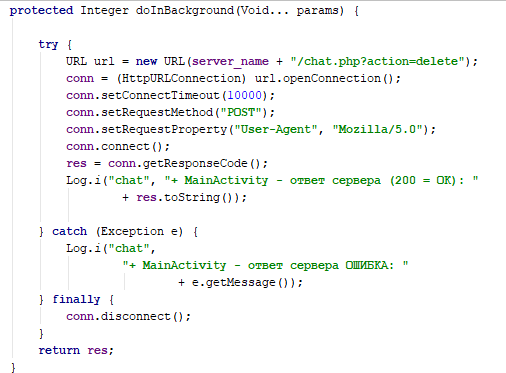


Рисунок 12 – Фрагмент кода из файла MainActivity

При безуспешном подключении к серверу, соединение закрывается, за это отвечает функция disconnect.

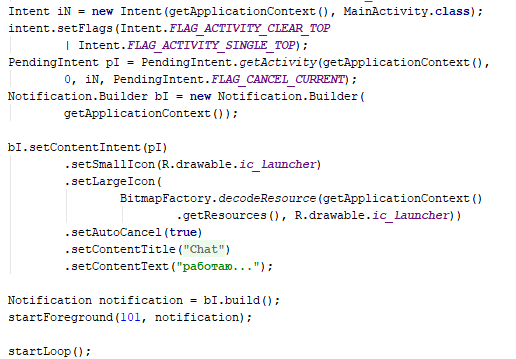


Рисунок 13 – Фрагмент кода из файла FoneService

Рисунок 13 демонстрирует алгоритм запуска работы в фоновом режиме. Процедура onStart устанавливает соединение с сервером, при этом создается и выводится на экран уведомление о запущенном приложении на Android Studio для пользователей. Это позволяет визуально облегчить управление интернет-мессенджером. Параметр ic\_launcher позволяет в фоновом режиме устанавливать значок, который соответствует Android приложению.

При запуске клиент-серверного приложения происходит запуск потока, внутри которого будет происходить регулярное соединение с сервером для чтения новых записей в БД. Если информационный текст был найден, то обновляется параметр ListView в файле ChatActivity и формируется запрос, по которому получаем только свежие сообщения. Сортировка осуществляется при помощи параметра data, которое задается сервером, поэтому является всегда уникальным.

Установив соединение с сервером, формируем JSON ответ (рисунок 14).



Рисунок 14 – Фрагмент кода из файла FoneService

Параметр new\_mess разбивает переданную информацию в соответствие наименования столбцов, т.е. происходит разделение на адресанта, получателя, времени отправления и текста сообщений, и записывает их в исходную таблицу БД.

ЗАКЛЮЧЕНИЕ

Результатом выполнения курсовой работы является разработанное клиент-серверное приложение на Android Studio, которое предоставляет возможность передавать сообщения между двумя пользователями, данные которых хранятся в БД.

В ходе работы были рассмотрены в первую очередь вопросы, касающиеся архитектуры Android-приложений, а также затронутые языки программирования для реализации интернет-мессенджера.

Макет приложения имеет удобный и технически понятный интерфейс, что позволяет клиентам с легкостью ориентироваться в данном приложении. При запуске интернет-мессенджера пользователю представлена возможность выбора собеседника, удобного перемещения в диалог, а также удаления всей информации с данного устройства и базы данных, которая располагается на сервере.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Курс по архитектуре клиент-серверных приложений на Android Studio. URL: http://www.fandroid.info/ (дата обращения: 28.04.2018)
2. Start Android – учебник по Android для начинающих. URL: [http://startandroid.ru](http://startandroid.ru/ru/uroki/vse-uroki-spiskom.html/)/ (дата обращения:15.04.2018)
3. Герберт Шилдт Java: Руководство для начинающих. – Издательство: Вильямс, 2015, 133 с.
4. Алексей Васильев Самоучитель Java. – Издательство: Наука и Техника, 2017, 368 с.
5. Гурвиц Г. Разработка реального приложения в среде клиент-сервер –ДВГУПС, 2005, 206 с.
6. Леон Шкляр, Рич Розен Архитектура веб-приложений – Издательство: Эксмо, 2010, 640с.