МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

«ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра технологий программирования

КУРСОВАЯ РАБОТА

**по дисциплине «Операционные системы и системное программирование»**

**Тема: «Клиент мессенджера на Android»**

Выполнил: студент группы 15-ИТ-1

\_\_\_\_\_\_ И.И. Павинич

Проверил: ассистент кафедры ТП

Е.С. Гатихо

Отметка:

Дата защиты: « » \_\_\_\_\_\_\_\_\_\_ 2017г.

Члены комиссии:  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

Полоцк, 2017

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc501660498)

[1. Анализ исходных данных 5](#_Toc501660499)

[2. Программное проектирование 7](#_Toc501660500)

[3. Программная реализация 10](#_Toc501660501)

[3.1 Детальная реализация функциональных частей ПО 10](#_Toc501660502)

[3.2 Сопроводительная документация 14](#_Toc501660503)

[3.3 Анализ ПО 14](#_Toc501660504)

[3.4 Тестирование ПО 14](#_Toc501660505)

[Заключение 20](#_Toc501660506)

[Список источников 21](#_Toc501660507)

[ПРИЛОЖЕНИЕ А 22](#_Toc501660508)

[Приложение Б 25](#_Toc501660522)

# Введение

В рамках данного курсового проекта необходимо реализовать клиента мессенджера на ОС Android. Наименование программного продукта – «Messenger».

Мессенджер - это программа, мобильное приложение или веб-сервис для мгновенного обмена сообщениями. Английское слово messenger является производным от message - сообщение.

Чаще всего под мессенджером понимают программу, в которую вы пишите сообщения и где вы их читаете. Однако, за каждой такой программой стоит сеть обмена сообщениями, которая тоже входит в понятие «мессенджер». Мессенджеры становятся все более популярным каналом коммуникации, захватывая в том числе и публичное пространство с помощью чатов и каналов.

Мессенджеры приходят на смену социальным сетям. Недавно Павел Дуров назвал друзей в соцсетях «явлением 2010 года», заявив, что «все, кто нужен, уже давно в мессенджерах». Тем временем, мировая аудитория Facebook продолжает расти, давно перевалив за миллиард, однако мессенджеры демонстрируют более активный рост аудитории. Согласно исследованию, опубликованному Business Insider, в 2016-м году ежемесячная мировая аудитория мессенджеров превзошла мировую ежемесячную аудиторию соцсетей.

Люди все чаще используют мессенджеры как самый быстрый и простой способ общаться в среде, где нет ничего лишнего. Теперь таким образом можно общаться не только с друзьями, но и с брендами, получать доступ к самым нужным услугам, минуя поисковики и сайты [5].

# Анализ исходных данных

Требования к функциональным характеристикам расположены в приложении А.

Приложение должно состоять из следующих частей:

1. Вход в существующий аккаунт.
2. Регистрация нового пользователя.
3. Список диалогов.
4. Список сообщений с выбранным пользователем.
5. Список друзей.
6. Поиск пользователей.
7. Просмотр профиля пользователя.

Активность входа в существующий аккаунт должна содержать поля ввода для логина и пароля, кнопку входа и возможность перехода на активность регистрации.

Активность регистрации нового пользователя должна содержать поля ввода для имени, фамилии, логина и пароля нового пользователя, возможность выбора аватара аккаунта и возможность перехода на активность входа.

Список диалогов должен состоять из последних диалогов пользователя. В каждом элементе списка должны отображаться: имя и фамилия собеседника, статус (онлайн или нет), время последнего сообщения, текст последнего сообщения, аватар собеседника. Элемент списка должен иметь отдельный интерфейс, если у диалога имеется непрочитанное сообщение.

Список сообщений с выбранным пользователем должен иметь все отправленные сообщения между двумя пользователями. Сообщения, отправленные собеседником должны иметь другой дизайн, для удобности определения отправителя сообщения. Каждое сообщение должно иметь время отправления и содержимое.

В списке друзей должны отображаться пользователи, которые были добавлены в друзья. Каждый элемент списка представляет собой аватар друга, его имя, фамилию и статус (онлайн или нет).

Поиск пользователей должен быть возможен по имени, фамилии или логину.

В просмотре профиля пользователя должны отображаться: аватар, имя, фамилия и логин. Если пользователь просматривает свой профиль, то он должен иметь возможность его изменения и выхода из аккаунта. Если пользователь просматривает чужой профиль, то оттуда он должен иметь возможность перейти к сообщениям с выбранным пользователем, либо добавить его в друзья.

В качестве среды разработки выбрана Android Studio, потому что эта среда разработки является лучшим средством для разработки приложений на ОС Android, имеет приятный дизайн, автодополнение кода, подсказывает о недочетах, является официальной средой разработки на Android. Язык программирования – Java [1], т.к. примеры кода в официальной документация для разработки на Android написаны на Java. Приложение должно запускаться на устройстве с установленной OS Android (версия от 5.0), и с ОЗУ не менее 512 мегабайт.

Сервер должен уметь принимать запросы от клиента, обрабатывать их и посылать ответ. В качестве среды разработки для сервера была выбрана Intellij IDEA, т.к. она очень удобна в использовании, имеет приятный дизайн, имеет возможность автодополнения кода.

Язык программирования – Java, используемая технология – Java Servlet. Сервлет взаимодействует с клиентами по принципу запрос – ответ. Протокол взаимодействия – HTTP. HTTP – протокол передачи гипертекста. Наиболее распространенные методы – GET и POST. Метод GET используется, например, при обычном открытии страницы в браузере. Метод POST используется в случае, если параметры нельзя передать обычной строкой, например, изображение.

Для соединения с базой данных сервер использует JDBC-драйвер. Обращение к содержимому базы данных осуществляется с помощью SQL-запросов. Системой управления базами данных была выбрана MySQL. Причины выбора MySQL:

1. Поддержка языка структурированных запросов SQL.
2. Быстродействие в сравнении с конкурентами.
3. Высокий уровень безопасности.
4. Инструмент для визуального проектирования баз данных Workbench.
5. Доступ к базе данных с нескольких устройств.

Сервером приложений выбран Apache Tomcat. Tomcat – контейнер сервлетов, позволяет запускать веб-приложения. Преимуществом является маленький вес, поддержка в Intellij IDEA, простая конфигурация и развертывание сервера.

# Программное проектирование

При первом запуске на смартфоне должна появиться активность входа в аккаунт. Макет представлен на рисунке 2.1.

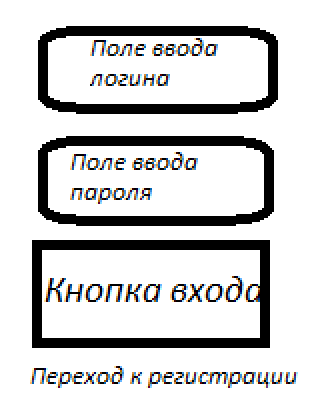


Рисунок 2.1 — Макет входа в аккаунт

При переходе к регистрации откроется активность регистрации (рисунок 2.2).



Рисунок 2.2 — Макет регистрации

Кнопка «Назад» должна вернуть пользователя на активность входа. После нажатия на кнопку регистрации, если регистрация пройдет успешно, откроется список диалогов. Пример интерфейса списка диалогов представлен на рисунке 2.3.



Рисунок 2.3 — Макет интерфейса списка диалогов

Пример интерфейса списка друзей и поиска пользователей представлен на рисунке 2.4.



Рисунок 2.4 — Макет списка пользователей

Пример интерфейса списка сообщений между пользователями представлен на рисунке 2.5.

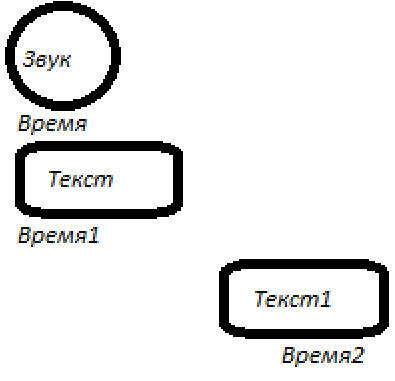


Рисунок 2.5 — Макет списка сообщений

Пример просмотра профиля представлен на рисунке 2.6.

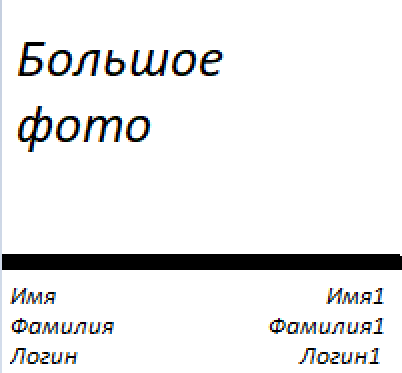


Рисунок 2.6 — Макет просмотра профиля

Для проектирования отношений между классами внутри приложения используется язык UML.

UML — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур [6].

Диаграмма вариантов использования представлена в приложении Б.

# Программная реализация

## Детальная реализация функциональных частей ПО

Класс HttpConnect используется при входе в аккаунт, этот класс наследуется от AsyncTask. AsyncTask используется для выполнения задач в другом потоке [2]. При наследовании от класса обязательно надо переопределить метод doInBackground(), данный метод вызывается во время работы потока. Также в классе переопределен метод inPostExecute(), данный метод вызывается после завершения работы метода doInBackground(). В методе doInBackground() открывается соединение по указанному адресу, в параметрах передается введенный логин и пароль. После получения логина и пароля сервер возвращает код результата входа. После этого клиент получает код и если код обозначает успешный вход, то переходит на активность с диалогами.

**Листинг 1** - Класс HttpConnect

1. **class** HttpConnect **extends** AsyncTask<String, Void, Void>{
2. @Override
3. **protected** **void** onPostExecute(Void aVoid) {
4. **if**(code == HttpURLConnection.HTTP\_OK) {
5. CookiesWork.saveCookie(getSharedPreferences("SharPrefs"
6. , MODE\_PRIVATE));
7. startActivity(**new** Intent(SignIn\_Activity.**this**,
8. DialogsActivity.**class**));
9. }
10. progressBar.setVisibility(View.GONE);
11. }
12. @Override
13. **protected** Void doInBackground(String... params) {
14. HttpURLConnection connection = **null**;
15. URL url;
16. **try** {
17. url = **new** URL(getResources().getString
18. (R.string.url) + "?operation=login&login=" +
19. URLEncoder.encode(params[0], "UTF-8") + "&password=" +
20. SHA.encrypt(params[1]));
21. connection = (HttpURLConnection)
22. url.openConnection();
23. connection.setRequestMethod("GET");
24. List<String> cookies =
25. connection.getHeaderFields().get(CookiesWork.COOKIES\_HEADER);
26. String[] vals = cookies.get(0).split("=");
27. CookiesWork.cookie = vals[1];
28. code = connection.getResponseCode();
29. }
30. **catch** (Exception e){
32. }
33. **finally** {
34. **if**(connection!=**null**)
35. connection.disconnect();
36. }
37. **return** **null**;
38. }
39. }

TextWatcher textWatcher является реализацией наблюдателя изменения текста. При изменение текста в полях логина и пароля вызывается метод OnTextChaged(), в котором осущесвляется проверка на правильность ввода (листинг 2).

**Листинг 2** - textWatcher

1. TextWatcher textWatcher = **new** TextWatcher() {
2. @Override
3. **public** **void** beforeTextChanged
4. (CharSequence charSequence, **int** i, **int** i1, **int** i2) {
5. }
6. @Override
7. **public** **void** onTextChanged(CharSequence charSequence,
8. **int** i, **int** i1, **int** i2) {
9. **if**(checkLogin(etLogin.getText().toString()))
10. etLogin.setCompoundDrawablesWithIntrinsicBounds
11. (**null**, **null**, getResources().getDrawable
12. (R.drawable.icon\_valid), **null**);
13. **else** etLogin.
14. setCompoundDrawablesWithIntrinsicBounds(**null**,**null**,**null**,**null**);
15. **if**(checkPassword(etPassword.getText().toString()))
16. etPassword.setCompoundDrawablesWithIntrinsicBounds
17. (**null**, **null**, getResources().getDrawable
18. (R.drawable.icon\_valid), **null**);
19. **else** etPassword.
20. setCompoundDrawablesWithIntrinsicBounds(**null**,**null**,**null**,**null**);
21. **if**(checkLogin(etLogin.getText().
22. toString()) && checkPassword(etPassword.getText().toString()))
23. bSignIn.setEnabled(**true**);
24. **else** bSignIn.setEnabled(**false**);
25. }
26. @Override
27. **public** **void** afterTextChanged(Editable editable) {
28. }
29. };

Класс SearchAdapter наследуется от BaseAdapter и является адаптером для списка пользователей в поиске. Метод getView() возвращает представление как должен выглядеть один элемент списка (листинг 3).

**Листинг 3** - SearchAdapter

1. **public** **class** SearchAdapter **extends** BaseAdapter {
2. ArrayList<User> users;
3. Context mContext;
5. **public** SearchAdapter(Context mContext, ArrayList<User> users) {
6. **this**.mContext = mContext;
7. **this**.users = users;
8. }
10. **public** **int** getCount() {
11. **return** users.size();
12. }
14. **public** Object getItem(**int** arg0) {
15. **return** users.get(arg0);
16. }
18. **public** **long** getItemId(**int** position) {
19. **return** position;
20. }
22. **public** View getView(**int** position, View arg1,
23. ViewGroup viewGroup){
24. LayoutInflater inflater =
25. (LayoutInflater)mContext.getSystemService
26. (Context.LAYOUT\_INFLATER\_SERVICE);
27. View row = inflater.inflate(R.layout.user\_item,
28. viewGroup, **false**);
29. TextView fullName = row.findViewById(R.id.user\_name);
30. fullName.setText(users.get(position).getSurname()
31. + " " + users.get(position).getName());
32. fullName.setTypeface(Typeface.createFromAsset
33. (mContext.getAssets(), "Roboto-Bold.ttf"));
34. ImageView imageView = row.findViewById(R.id.user\_img);
35. Bitmap img = users.get(position).getPicture();
36. **if**(img!=**null**)
37. imageView.setImageBitmap(users.get(position).
38. getPicture());
39. **else** imageView.setImageResource
40. (R.drawable.ic\_mood\_black\_72dp);
41. ImageView online = row.findViewById(R.id.onlineImg);
42. **if**(users.get(position).isOnline())
43. online.setVisibility(View.VISIBLE);
44. **return** row;
45. }
46. }

Класс SHA – класс для шифрования пароля по алгоритму SHA-512 (листинг 4).

**Листинг 4** – Шифрование SHA

1. **public** **class** SHA {
3. **public** **static** String encrypt(String passwordToHash)
4. {
6. passwordToHash = "#Void\_"+passwordToHash+"\_Chat$";
8. String generatedPassword = **null**;
9. **try** {
10. MessageDigest md = MessageDigest.getInstance
11. ("SHA-512");
12. md.update(passwordToHash.getBytes());
13. **byte**[] bytes = md.digest();
14. StringBuilder sb = **new** StringBuilder();
15. **for**(**int** i=0; i< bytes.length ;i++)
16. {
17. sb.append(Integer.toString(
18. (bytes[i] & 0xff) + 0x100, 16).substring(1));
19. }
20. generatedPassword = sb.toString();
21. }
22. **catch** (NoSuchAlgorithmException e)
23. {
24. e.printStackTrace();
25. }
26. **return** generatedPassword;
27. }
28. }

Для работы сервера мною были написаны следующие его части:

1. Фильтрация запросов по наличию куков.
2. Соединение с базой данных.
3. Проверка статуса пользователя (онлайн/оффлайн).
4. Отправка данных о пользователе по его логину.
5. Отправка списка диалогов.
6. Отправка списка сообщений.
7. Получение и отправка голосового сообщения.
8. Отправка списка друзей.

Для соединения сервера с базой данных используется класс ConnectDB. При инициализации сервлета открывается соединение с базой данных. В процессе работы сервера, при необходимости обращения к базе данных, получается объект созданного соединения.

**Листинг 5** - ConnectDB

1. **public** **class** ConnectDB {
2. **private** **final** String URL ="jdbc:mysql://localhost:3306/messengerdb" +
3. "?autoReconnect=true&useSSL=false";
4. **private** **final** String USER = "root";
5. **private** **final** String PASSWORD = "root";
6. Connection connection;
8. **public** Connection getConnection() {
9. **return** connection;
10. }
12. **public** ConnectDB() {
13. **try**{
14. Class.forName("com.mysql.jdbc.Driver");
15. connection = DriverManager.getConnection(URL,
16. USER,PASSWORD);
17. **if**(!connection.isClosed())
18. System.out.println("CONNECTED");
19. }
20. **catch** (Exception e){
21. System.out.println("ERROR CONNECTION");
22. }
23. }
24. }

## Сопроводительная документация

Сопроводительная документация по разработанному программному продукту предоставляется в составе технического задания (приложение А) согласно ГОСТ 19.201-78.

Требования к сопроводительной документации устанавливаются государственными стандартами ЕСПД.

## Анализ ПО

Анализ ПО был проведен с помощью встроенного анализатора «Android Profiler». Анализ ПО содержит в себе, проверку потребления ресурсов процессора, оперативной памяти и сети в фоновом режиме, и в активном использовании. Результаты анализа представлены на рисунке 3.1.

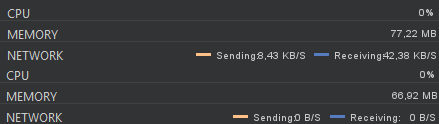


Рисунок 3.1 – Сравнение активного и фонового режима

На рисунках заметно, что приложение использует небольшое количество оперативной памяти как в активном, так и в фоновом режиме (в сравнении с другими приложениями похожего типа меньше в 2 раза).

В верхней части рисунка приложение анализировалось в активном режиме, скриншот представлен во время получения списка всех диалогов. Учитывая, что на тот момент приложение получило 7 диалогов (один диалог содержит картинку пользователя, текст, дату, статус пользователя), можно сделать вывод, что приложение расходует малое количество трафика.

## Тестирование ПО

Данная программа будет проходить smoke test. Этот тест наиболее подходит так как он проводится самим программистом.

Список тестов, которые проходило приложение, представлен ниже:

1. Запуск приложения.
2. Ввод данных.
3. Регистрация пользователя.
4. Просмотр диалогов.
5. Прием нового сообщения.
6. Просмотр списка сообщений.
7. Отправка нового сообщения.
8. Просмотр профиля.
9. Редактирование профиля.
10. Просмотр друзей.
11. Поиск пользователей.
12. Выход из аккаунта.

Запуск приложения (рисунок 3.2).

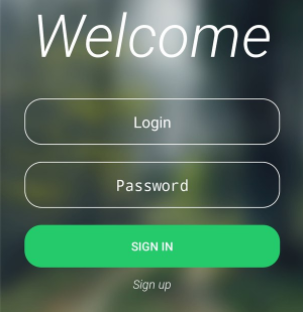


Рисунок 3.2 - Запуск приложения

Ввод логина и пароля (рисунок 3.3).



Рисунок 3.3 — Ввод логина и пароля

Попытка регистрации нового пользователя (рисунок 3.4).

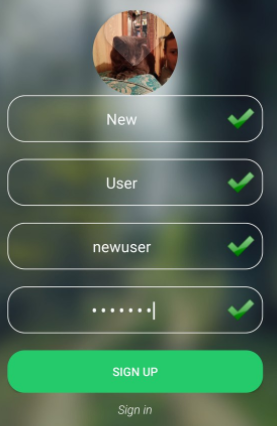


Рисунок 3.4 — Регистрация пользователя

После регистрации или входа в аккаунт открывается список диалогов (рисунок 3.5).

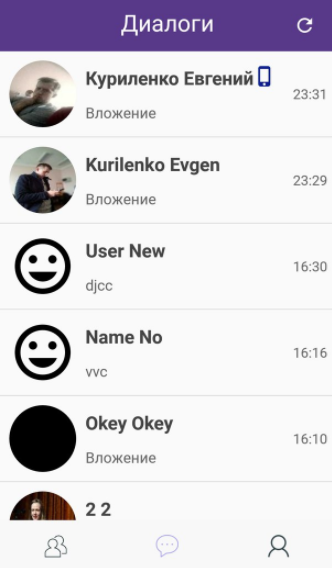


Рисунок 3.5 — Список диалогов

При поступлении нового сообщения диалоги автоматически обновляются и непрочитанное сообщение выделяется другим цветом (рисунок 3.6).

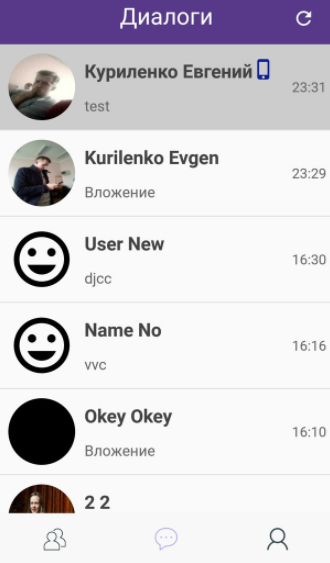


Рисунок 3.6 — Обновление диалогов

Просмотр сообщений с выбранным пользователем (рисунок 3.7).

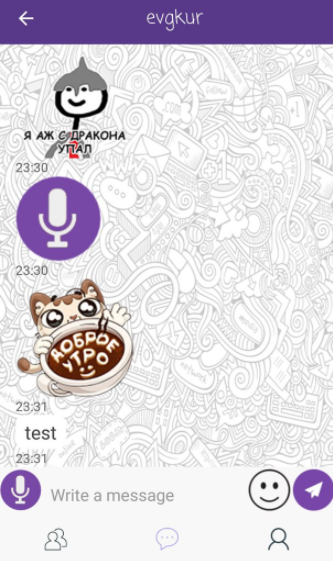


Рисунок 3.7 - Просмотр сообщений

Отправка сообщения (рисунок 3.8).

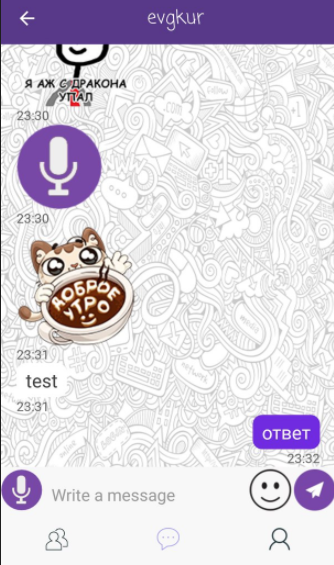


Рисунок 3.8 – Отправка сообщения

Просмотр профиля (рисунок 3.9).

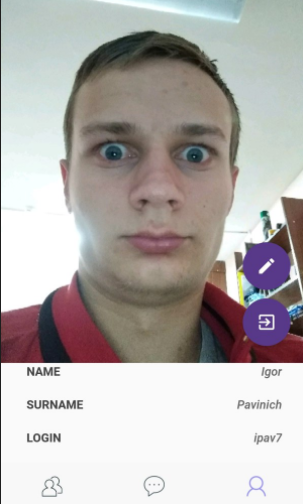


Рисунок 3.9 – Профиль

Профиль после обновления аватара (рисунок 3.10).

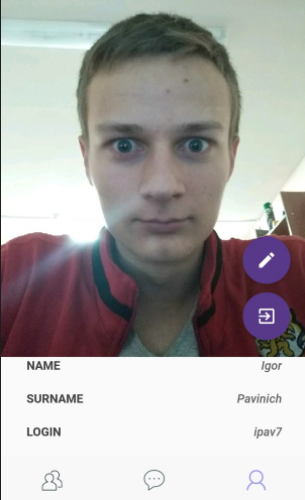


Рисунок 3.10 – Отредактированный профиль

Просмотр друзей (рисунок 3.11).

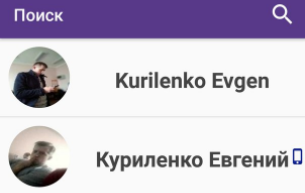


Рисунок 3.11 – Друзья

Поиск пользователя (рисунок 3.12).

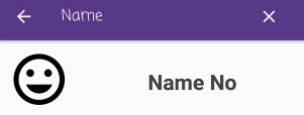


Рисунок 3.12 – Поиск

После выхода из аккаунта открывается активность входа в аккаунт.

В итоге 100% тестов были пройдены.

# Заключение

В результате выполнения данной курсовой работы получены навыки по созданию приложений на ОС Android. Это является довольно важным навыком, так мобильные платформы развиваются все больше. Закреплены навыки программирования на языке Java. При написании интерфейса использовался язык разметки XML. Были получены и закреплены навыки по использованию данного языка разметки.

В результате были получены навыки по развертыванию сервера на языке Java с использованием технологии Java Servlet. Более тщательно был изучен протокол взаимодействия HTTP, изучена структура его запросов, основные методы.

В ходе работы научился работать с базами данных, проектировать их, изучил множество SQL-запросов.

Помимо сетевого программирования в данной работе активно использовалось многопоточное программирование, в результате чего были получены теоретические и практические знания по использованию потоков на Android.

В процессе проектирования работы научился выбирать необходимые для разработки технологии, чтобы они соответствовали требованиям, поддерживались производителями.

В ходе разработки программы были реализованы алгоритмы передачи изображения, а также его сжатия.

В ходе тестирования критических ошибок не обнаружено. Все запланированные тесты были пройдены.

В рамках работы были выполнены все представляемые задачи. Имеется возможность отправлять текстовые, голосовые сообщения, стикеры, добавлять пользователей в друзья, смотреть находится ли другой пользователь онлайн.

В итоге получился мессенджер, пригодный для использования.

# Список источников

1. Java [Электронный ресурс] — режим доступа: https://ru.wikipedia.org/wiki/Java. Дата обращения: 12.12.2017.
2. Выполнение задач в бэкграунде [Электронный ресурс] — режим доступа: https://habrahabr.ru/post/136942/. Дата обращения: 13.12.2017.
3. MediaPlayer – аудио/видео плеер, основные возможности [Электронный ресурс] — режим доступа: http://startandroid.ru/ru/uroki/vse-uroki-spiskom/236-urok-126-media-mediaplayer-audiovideo-pleer-osnovnye-vozmozhnosti.html. Дата обращения: 14.12.2017.
4. Java Кастомизация списка. Создаем свой адаптер [Электронный ресурс] — режим доступа: http://startandroid.ru/ru/uroki/vse-uroki-spiskom/113-urok-54-kastomizatsija-spiska-sozdaem-svoj-adapter.html. Дата обращения: 14.12.2017.
5. Почему мессенджеры лучше остальных средств коммуникации и как их использует бизнес [Электронный ресурс] – режим доступа: https://vc.ru/14092-messengers-2016. Дата обращения: 14.12.2017.
6. UML [Электронный ресурс] – режим доступа: https://ru.wikipedia.org/wiki/UML. Дата обращения: 14.12.2017.

# ПРИЛОЖЕНИЕ А

(обязательное)

Техническое задание

Введение

Наименование программного продукта – клиент мессенджера для ОС Android. Клиент рассчитан на любого человека, имеющего устройство с установленной операционной системой Android и выходом в интернет. С помощью приложения пользователь может общаться с друзьями, семьей, коллегами, заводить новые знакомства.

А.1 Основание для разработки

Клиент мессенджера на Android разрабатывается в рамках курсового проекта студента учреждения образования «Полоцкий государственный университет» Павинича И.И. Основанием для разработки является выданное задание к курсовому проекту по курсу «Операционные системы и системное программирование».

А.2 Назначение разработки

Функциональное и эксплуатационное назначение клиента – общение с другими людьми посредством выхода в интернет.

А.3 Требования к программе или программному изделию

А.3.1 Требования к функциональным характеристикам

При разработке приложения были выдвинуты следующие требования к функциональным характеристикам:

1. Вход в свою учетную запись по логину и паролю или регистрации нового пользователя (ввод имени, фамилии, логина, пароля, загрузка фотографии).
2. Текстовое сообщение.
3. Голосовое сообщение.
4. Просмотр или редактирование профиля.
5. Проверка статуса пользователя (онлайн или нет).
6. Добавление пользователя в друзья.
7. Поиск пользователей.

А.3.2 Требования к надежности

Клиент должен надежно функционировать и в случае программного или аппаратного сбоя должен продолжать работу, извещая пользователя о произошедшем.

**А.3.3 Условия эксплуатации**

Для эксплуатации должен быть настроен сервер и расположенная на нём база данных, а также у пользователя должен быть доступ к интернету. Минимальные требования к пользователю – умение обращаться с телефоном на базе ОС Android.

А.3.4 Требования к составу и параметрам технических средств

Для обеспечения устойчивости работы программного средства требуется:

1. Количество ядер процессора – не менее 2.
2. ОЗУ не менее 512 МБ.
3. До 10 МБ доступного дискового пространства.

А.3.5 Требования к информационной и программной   
совместимости

Требования совместимости:

1. ОС Android 5.0 и выше.
2. Микрофон.
3. Динамик.
4. Доступ к сети интернет.

А.3.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке отсутствуют.

А.3.7 Требования к транспортированию и хранению

Программное средство должно храниться на электронном носителе в виде исполняемых файлов программного продукта.

А.4 Требования к программной документации

Программная документация по программе должна быть предоставлена в техническом задании согласно ГОСТ 19.201-78.

Требования к перечисленным программным документам устанавливаются государственными стандартами ЕСПД.

А.5 Стадии и этапы разработки

Разработка клиента заключается в следующем:

1. Анализ исходных данных и постановка задачи проектирования, разработка технического задания.
2. Проектирование и разработка базы данных.
3. Развертывание сервера.
4. Разработка интерфейса, архитектуры и структуры.
5. Реализация и тестирование программы.
6. Разработка программной документации.

А.6 Порядок контроля и приемки

Контроль и приемка программного средства осуществляется в соответствии с программой и методикой испытаний.

Для проверки корректности приложения применялись следующие программные средства:

1. Flyme OS.
2. Среда разработки Android Studio.
3. MySQL Workbench.

Тестирование программы состояло из проверки корректности работы ранее перечисленных функций.

Методы испытаний:

Основным методом испытания программы является визуальный контроль выполнения программой требующихся функций, корректное выполнение юнит-тестов.

# Приложение Б

**Диаграмма вариантов использования**

