0МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Chat»

Выполнил студент Романович Никита Русланович

(Ф.И.О.)

Руководитель проекта асс. Колмаков М.В. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: асс. Колмаков М.В. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: асс. Колмаков М.В. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc41319422)

[1 Обзор аналогов приложений «Средств обмена сообщениями» 4](#_Toc41319423)

[1.1 Приложение «Telegram» 4](#_Toc41319424)

[1.2 Приложение «WhatsApp» 5](#_Toc41319425)

[1.3 Приложение «Viber» 6](#_Toc41319426)

[1.4 Выводы по главе «Обзор аналогов приложений «Средств обмена сообщениями»» 7](#_Toc41319427)

[2 Проектирование программного продукта 8](#_Toc41319428)

[2.1 Проектирование мобильного приложения 8](#_Toc41319429)

[2.2 Проектирование базы данных 10](#_Toc41319430)

[2.3 Проектирование сервера 11](#_Toc41319431)

[2.4 Выводы по главе «Проектирование программного продукта» 11](#_Toc41319432)

[3 Программная реализация приложения 12](#_Toc41319433)

[3.1 Технические средства для разработки приложений 12](#_Toc41319434)

[3.2 Разработка баз данных 12](#_Toc41319435)

[3.3 Разработка мобильного приложения 13](#_Toc41319436)

[3.4 Разработка сервера 16](#_Toc41319437)

[3.5 Вывод по главе «Программная реализация приложений» 17](#_Toc41319438)

[4 Тестирование разработонного продукта 18](#_Toc41319439)

[4.1 Негативное тестирование 18](#_Toc41319440)

[4.2 Выводы по главе «Тестирование разработонного продукта» 20](#_Toc41319441)

[5 Руководство пользователя 21](#_Toc41319442)

[5.1 Выводы по главе «Руководство пользователя» 23](#_Toc41319443)

[ЗАКЛЮЧЕНИЕ 24](#_Toc41319444)

[Список использованных источников 25](#_Toc41319445)

[ПРИЛОЖЕНИЕ А 26](#_Toc41319446)

[ПРИЛОЖЕНИЕ Б 27](#_Toc41319447)

[ПРИЛОЖЕНИЕ В 28](#_Toc41319448)

# **ВВЕДЕНИЕ**

Цель курсового проекта: разработать программное средство «Chat».

Задачи курсового проекта:

1. Изучить аналогичные программные средства по теме курсового проекта.

Данная задача позволит нам изучить похожие программные средства для анализа, где пользователям может быть неудобно в использовании приложения. Также определить, чем новое приложение будет отличатся от уже имеющихся аналогов.

1. Спроектировать приложение.

Проектирование приложения позволит нам увидеть прозрачную картину предстоящей работы целиком и разрабатывать приложение с четким пониманием что, зачем и для кого делается, а также сколько это займет времени.

1. Реализовать приложение.

В курсовом проекте будет реализовано мобильное приложение для отправки и получения сообщений между пользователями. В век развития информационных технологий этот вариант будет более оптимальный и удобный, так как пользователи смогут обмениваться информацией находясь в любой точке мире где есть доступ в интернет.

1. Протестировать разработанный продукт.

Это один из важных этапов в процессе создания продукта. Не уделив ему должного внимания, нельзя добиться высокого качества на выходе. Проведется негативное тестирование использование приложения в рамках которого будут применятся сценарии, которые соответствуют внештатному поведению тестируемой системы. Например, исключительные ситуации или неверные данные.

1. Разработать руководство пользователя.

В рамках работы данного курсового проекта будет разработано руководство пользователя, призванное помочь пользователю разобраться с работой программы и её основными функциями. Важно разработать руководство пользователя, даже если вы уверены, что с вашим продуктом легко работать, а его интерфейс интуитивно понятен – всегда найдутся пользователи, которые любят читать инструкции, прежде чем начать освоение продукта.

# **1 Обзор аналогов приложений «Chat»**

# **1.1 Приложение «Telegram»**

Telegram — облачный мессенджер, его можно использовать одновременно на нескольких устройствах, и все чаты, и файлы (за исключением Секретных чатов) будут доступны на этих устройствах. Во всех чатах можно использовать голосовые сообщения, видеосообщения, прикрепление файлов, стикеры, gif-анимации и эмодзи; есть отметка о том, что собеседник прочитал сообщение и т. д.

Приложение предоставлено скриншоте ниже (рис 1.1).

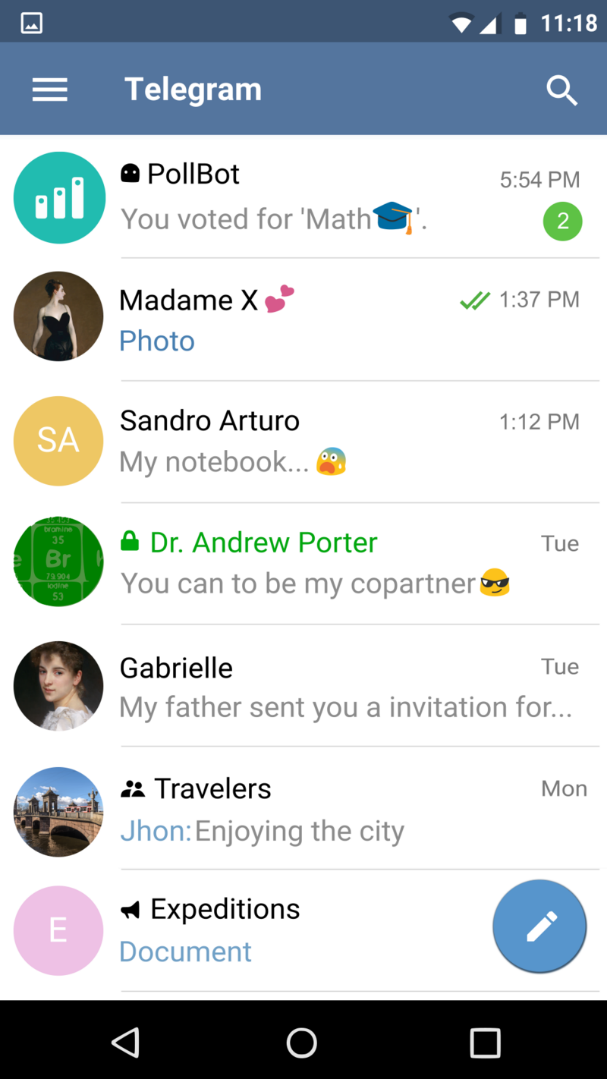


Рисунок 1.1 –Приложение «Telegram»

Telegram не предоставляет никому, кроме самих администраторов канала, информацию о том, кто ведёт канал и, кто на него подписан.

С концептуальной точки зрения каналы дают читателям, с одной стороны, возможность ощутить себя на одном уровне с автором (публикации каналов выглядят так же, как и обмен личными сообщениями), а с другой — позволяют пользователям потреблять контент в удобной системе координат в формате отдельного диалога (отталкиваясь от хронологии публикации материалов).

# **1.2 Приложение «WhatsApp»**

WhatsApp — популярная бесплатная система мгновенного обмена текстовыми сообщениями для мобильных и иных платформ с поддержкой голосовой и видеосвязи. Позволяет пересылать текстовые сообщения, изображения, видео, аудио, электронные документы и даже программные установки через Интернет.

Общее представление о приложении можно получить, посмотрев скриншот (рис 1.2).

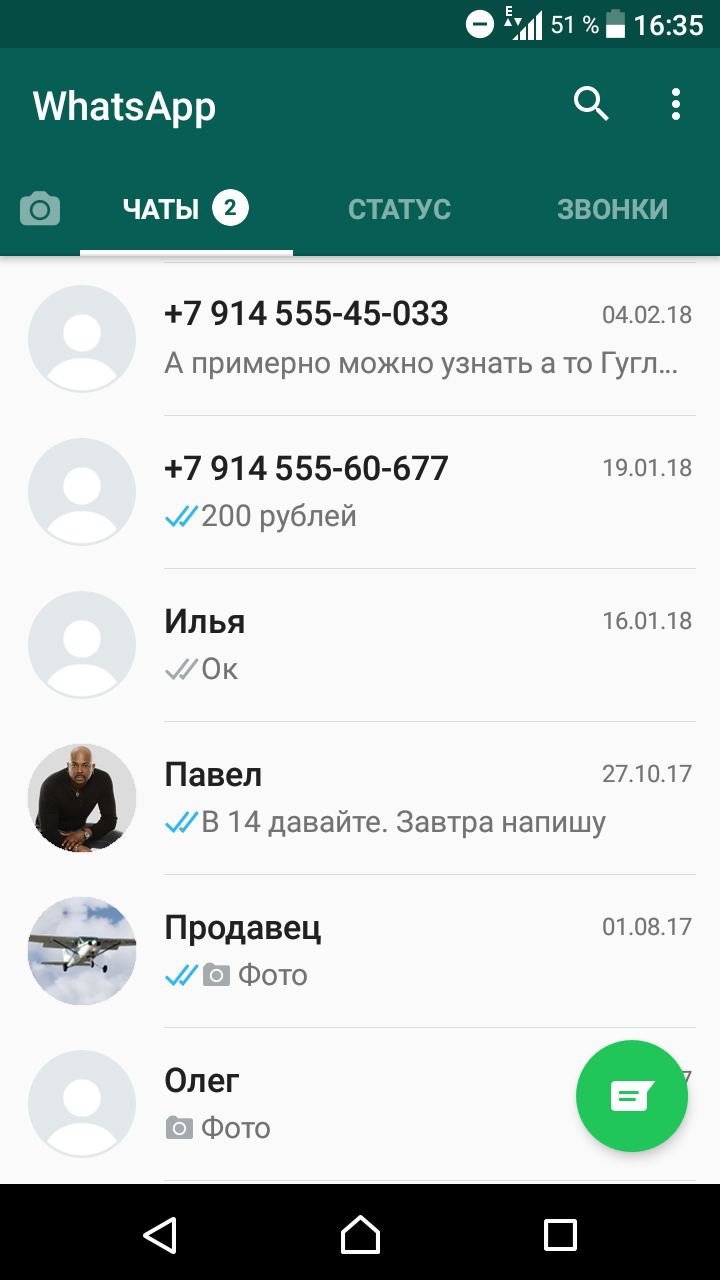


Рисунок 1. 2 – Приложение «WhatsApp»

Клиент работает на платформах Android, iOS, Windows Phone (отказ от поддержки с начала 2020 года), Nokia Symbian, Nokia S40, KaiOS, а также ОС Windows и в виде веб-приложения.

# **1.3 Приложение «Viber»**

Viber — приложение-мессенджер, которое позволяет отправлять сообщения, совершать видео- и голосовые VoIP-звонки через интернет. Голосовые вызовы между пользователями с установленным Viber бесплатны (оплачивается только интернет-трафик по тарифу оператора связи). Также в Viber имеется возможность передачи изображений, видео- и аудиосообщений, документов и файлов.

Приложение показано на скриншоте (рис 1.3):

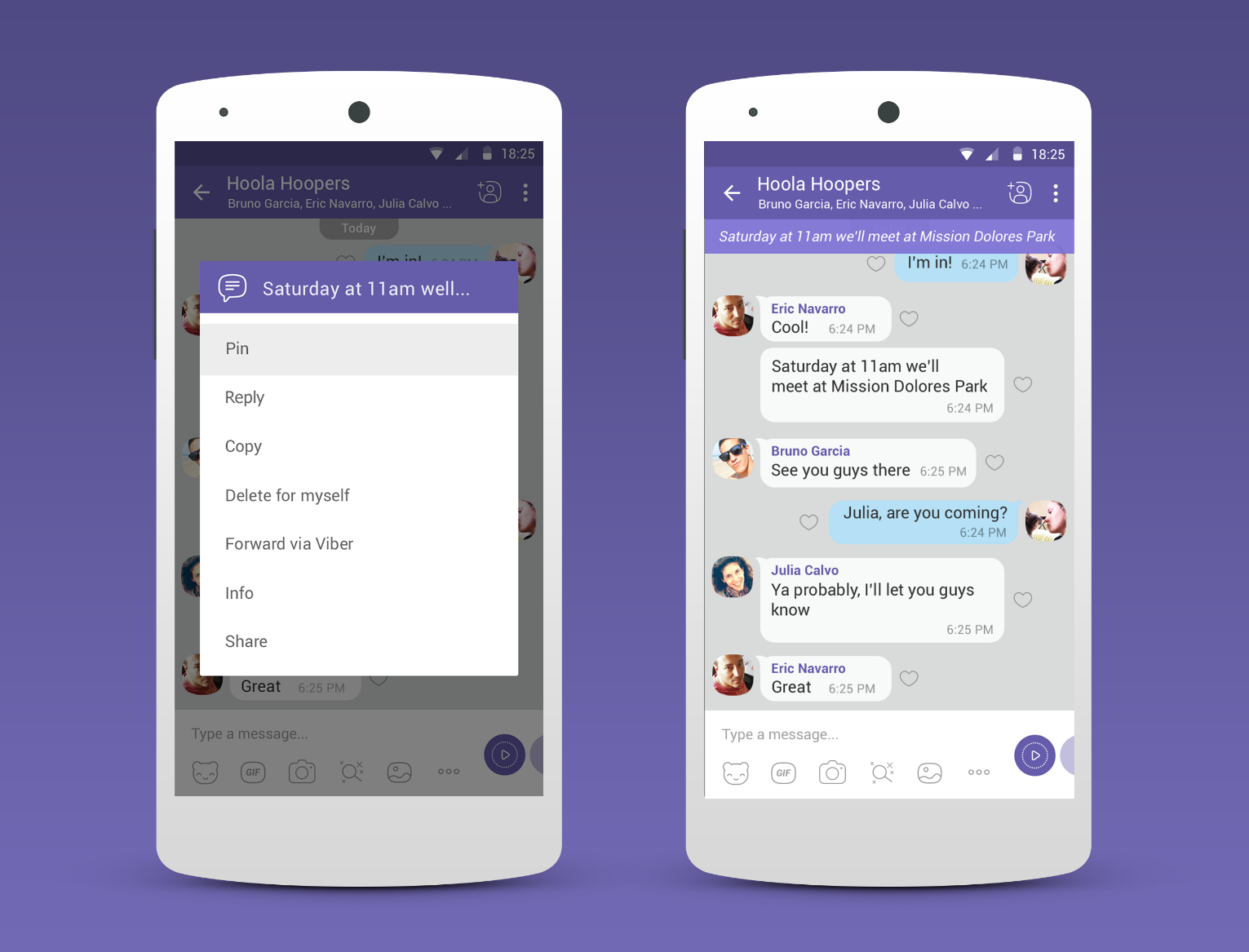


Рисунок 1.3 – Приложение «Viber»

Для авторизации пользователей и поиска контактов приложение использует номер телефона и передает содержимое телефонной адресной книги (имена и телефоны всех контактов) на серверы корпорации Viber Media, Люксембург Они же собирают информацию о совершенных звонках и переданных сообщениях, длительности звонков, участниках звонков и чатов — в целях улучшения качества обслуживания и в иных целях.

# **1.4 Выводы по главе «Обзор аналогов приложений «Средств обмена сообщениями»»**

В данной главе были изучены три приложения, которые являются аналогами приложения «Chat»: «Telegram», «WhatsApp» и «Viber». В результате были подведены выводы:

Дизайн у приложений «Telegram» и «WhatsApp» достаточно приятный для глаз. Однако дизайн «Viber» имеет нагромождение кнопок и текстового контента, который затрудняет восприятие информации. Из этого можно сделать вывод, что дизайн для будущего приложения необходимо делать простой и лаконичный.

Плюсы функционала приложения «Telegram»: анонимность, боты, многоязычность, обмен медиафайлами различных форматов.

Минусы функционала приложения «Telegram»: сами сообщения не зашифрованы. Они действительно передаются по зашифрованным протоколам, но на серверах сообщения хранятся в открытом виде.

Плюсы функционала приложения «WhatsApp»: приложение c 2016-го года полностью бесплатно. В данный момент у приложения нет никаких встроенных покупок.

Минусы функционала приложения «WhatsApp»: нет стикеров, таких как в Telegram или Viber.

Плюсы функционала приложения «Viber»: помимо обычной переписки можно совершать платные звонки на городские и мобильные телефоны, даже если на них не установлено приложение Viber.

Минусы функционала приложения «Viber»: плохая связь при голосовых и видео звонках.

Приложение «Telegram» является с ограниченным функционалом за использование всех функций необходимо платить.

Приложение «WhatsApp» приложение c 2016-го года полностью бесплатно.

Приложение «Viber» является с ограниченным функционалом за использование всех функций необходимо платить.

# **2 Проектирование программного продукта**

# **2.1 Проектирование мобильного приложения**

В рамках курсового проекта необходимо разработать приложение, позволяющее пользователю получить доступ к контактам, отправлять и получать текстовые сообщения, отправлять и получать изображения, регистрироваться.

На странице регистрации приложение должно предоставлять форму для заполнения данными пользователя для дальнейшей авторизации. Хранить имя, e-mail, фото и пароль пользователя, а также предотвращать не корректную регистрацию пользователями. Серверная часть приложения должна обеспечивать регистрацию пользователей. После регистрации предоставлять форму для авторизации.

На странице авторизации приложение должно предоставлять форму для авторизации пользователей. Серверная часть приложения должна обеспечивать авторизацию пользователей. После авторизации пользователя предоставлять пользователю список контактов, с которыми можно будет начать обмен информацией. В списке отображаться e-mail, фото и имя пользователей. Также на странице должно отображаться фото авторизированного пользователя.

Можно ознакомится с UML-диаграммой последовательности для окна «Чат» на рисунке 2.1.1.

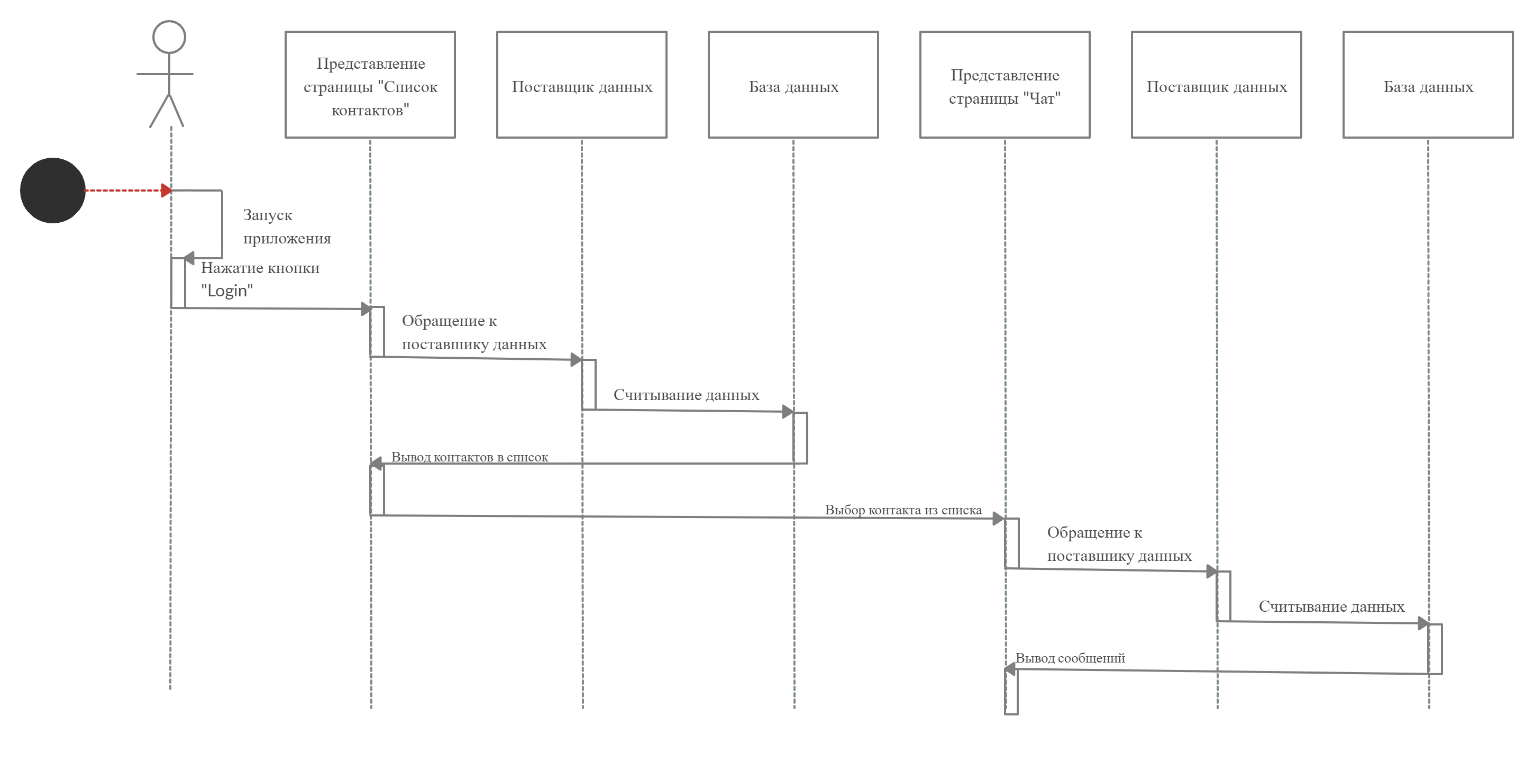


Рисунок 2.1.1 – Диаграмма последовательности

При нажатии на элемент списка содержащего данные о пользователе, должен осуществлять переход на страницу для обмена информацией, на этой странице отображаться текстовые сообщения и изображения, отравляемые между пользователями. Также на этой странице должен осуществляться обмен между пользователя текстовыми сообщениями и изображениями.

Можно ознакомиться с UML-диаграммой, а именно диаграммой вариантов использования. Она отображает функциональность программного продукта с точки зрения получения значимого результата для пользователя. (рис. 2.1.2)

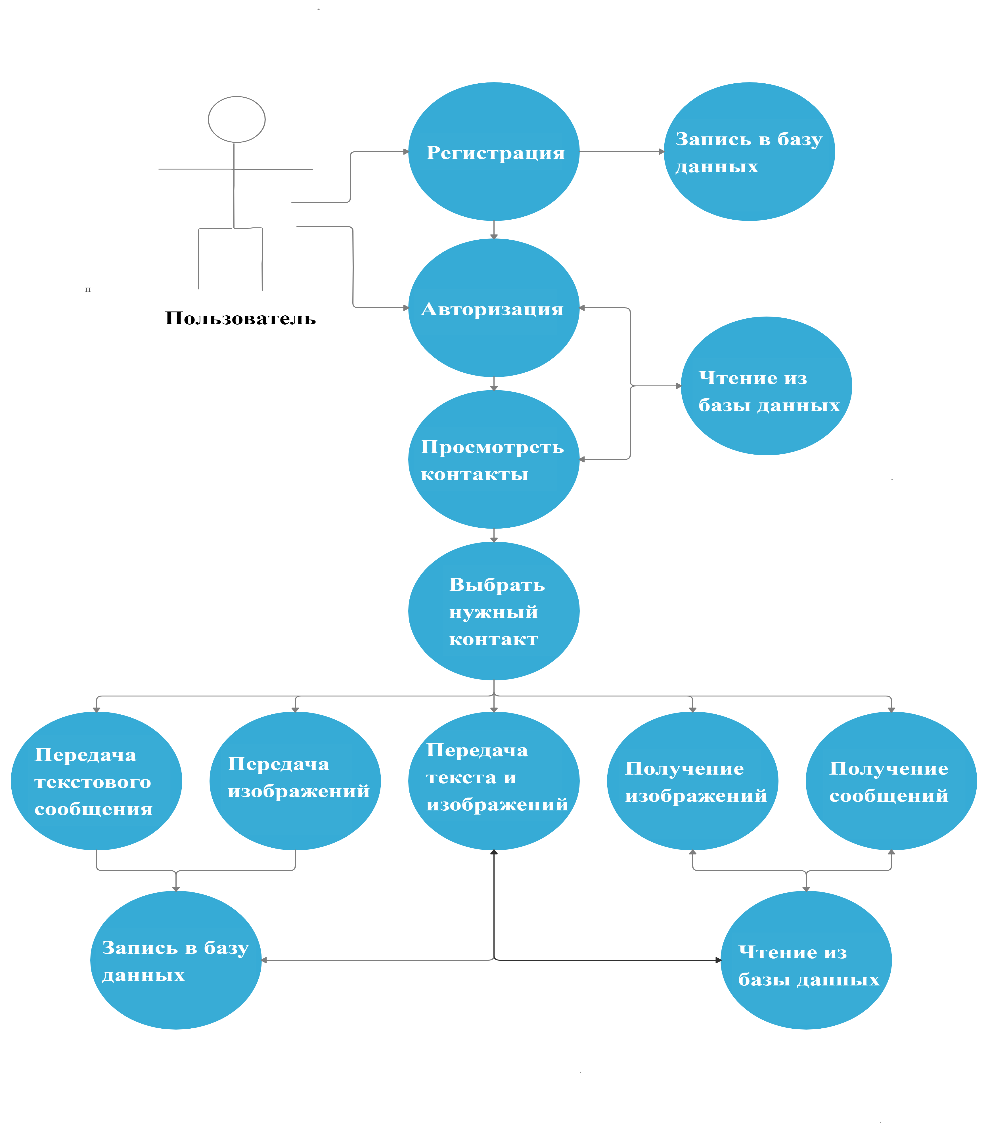


Рисунок 2.1.2 – Диаграмма вариантов использования

Таким образом приложение должно выполнять функции:

* предоставления списка пользователей из базы данных;
* предоставления списка текстовых сообщений из базы данных;
* предоставления списка изображений из базы данных;
* хранение списка пользователей в базе данных;
* хранение списка сообщений в базе данных;
* добавление пользователей в базу данных;
* шифрование данных;
* управление транзакциями;

# **2.2 Проектирование базы данных**

Названия таблиц, столбцы, которые в них есть, типы данных и ограничения целостности.

В базе данных данного приложения должны быть реализованы 2 таблицы.

В таблице «Users» содержится информация о пользователях. Столбцы данной таблицы:

* Id – целочисленный тип данных, не должен содержать значений null, является первичным ключом;
* UserName – строковый тип данных, максимальной длинной 50 символов;
* Email – строковый тип данных, максимальной длинной 50 символов;
* Password – строковый тип данных, максимальной длинной 50 символов;
* Image – строковый тип данных;
* IsAdmin – целочисленный тип данных, не должен содержать значений null

В таблице «Messages» хранится информация об ID категории и названии категории и поля для синхронизации с сервером

* Id – целочисленный тип данных, не должен содержать значений null, является первичным ключом;
* IdSender – целочисленный тип данных, внешний ключ;
* IdReceiver – целочисленный тип данных, внешний ключ;
* Message – строковый тип данных;
* Image – строковый тип данных;

Схема базы данных представлена на рисунке 2.2.1.

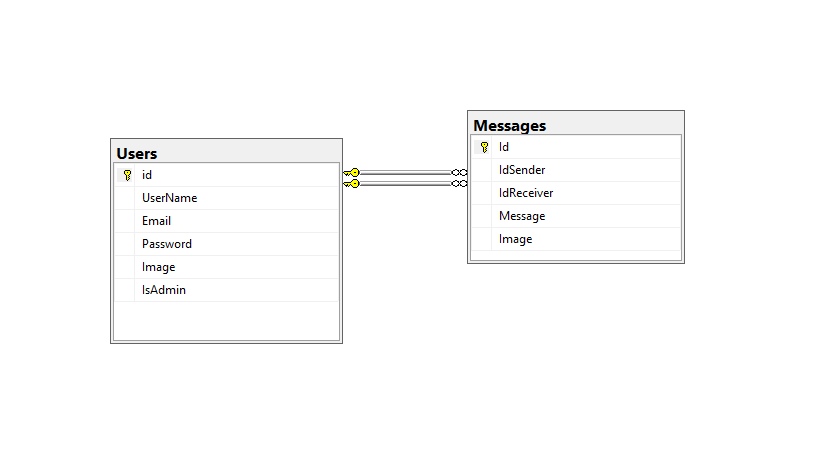


Рисунок 2.2.1 – Схема базы данных

Из данной схемы видно, что таблицы Users и Messages соединены связью Многие к одному.

# **2.3 Проектирование сервера**

Для синхронизации двух и более клиентов необходимо серверное приложение. Серверная часть приложения должна предоставлять API для доступа к каждой сущности в базе данных. Передаваемые или получаемые данные от сервера должны передаваться в формате JSON.

Серверное приложение должно быть построено на основе архитектурного стиля REST и будет использоваться для взаимодействия компонентов распределённого приложения в сети.

При обращении к серверу по заданному маршруту и определенному HTTP методу сервер будет определять с какой сущностью базы данных необходимо взаимодействовать и какие действия над данными необходимо будет произвести.

При помощи HTTP метода GET сервер будет возвращать данные клиенту, ссылаясь на аргументы в запросе, как на условие для выборки данных. Метод POST в свою очередь позволяет передать данные на сервер для последующей обработки.

Механизм синхронизации между серверным приложением и клиентским приложением представлено схемой в Приложении A.

# **2.4 Выводы по главе «Проектирование программного продукта»**

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения.

По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения и диаграмма последовательности одного из вариантов использования, а именно, отображение сообщений на странице «Чат».

Была также спроектирована база данных, где были определены необходимые таблицы, поля для каждой из них, определены типы данных и ограничения целостности. Также были установлены связи между таблицами, которые описаны на схеме базы данных.

При проектировании серверного приложения был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных, обозначили HTTP методы которые будут использованы в работе с серверной частью.

# **3 Программная реализация приложения**

# **3.1 Технические средства для разработки приложений**

Для разработки приложений курсового проекта были использованы следующие технические средства:

SQLite – компактная встраиваемая СУБД представляющая собой библиотеку.

MySQL – свободная реляционная система управления базами данных.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств.

Java – строго типизированный объектно-ориентированный язык программирования.

.NET Core – это модульная платформа для разработки программного обеспечения с открытым исходным кодом.

Entity Framework – специальная объектно-ориентированная технология на базе фреймворка .NET для работы с данными.

SQLCipher – библиотека с открытым исходным кодом, созданная компанией Zetetic, для прозрачного 256-битного AES шифрования баз данных SQLite.

OkHttp – библиотека предоставляющая собой простой и удобный клиент для работы с HTTP запросами.

Newtonsoft.Json – библиотека позволяющую преобразовывать объекты в JSON формат с большим количеством настроек.

# **3.2 Разработка базы данных**

Для хранения данных будет использована СУБД MySQL. Мобильное приложение отправляет данные серверу, который в свою очередь помещает их в базу данных.

Мобильное и серверное приложение имеют общие структурные объекты:

* User – таблица пользователей;
* Messages– таблица собщений;

Также будут разработаны три хранимые процедуры: «AddUser», «DeleteUser»,

«SendMessage».

# **3.3 Разработка мобильного приложения**

Мобильное приложение было реализовано в среде разработки – Android Studio 3.0. Для написания приложения был использован язык программирования Java.

Графическая структура проекта представлена на рисунке 3.3.1.

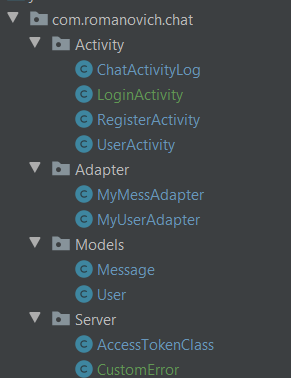


Рисунок 3.3.1 – Графическая структура проекта

Пакет Adapter хранит в себе классы для отображения данных в RecyclerView (см. таблицу 3.3.1).

**Таблица 3.3.1 – Содержание пакета Adapter**

|  |  |
| --- | --- |
| Класс MyUserAdapter – отображения данных пользователей | |
| onCreateViewHolder | Метод создания держателя вида |
| onBindViewHolder | Метод привязки данных к держателю вида |
| getItemCount | Метод считающий элементы списка |
| Класс ViewHolder – представляет собой держатель вида | |
| getItemViewType | Метод определяющий тип представления |
| StringToImage | Метод преобразующий строковый тип в изображение |
| Класс MyMessAdapter – отображение сообщений пользователей | |
| onCreateViewHolder | Метод создания держателя вида |
| onBindViewHolder | Метод привязки данных к держателю вида |
| getItemCount | Метод считающий элементы списка |

Пакет Models хранит в себе классы сущностей (см. таблицу 3.3.2).

**Таблица 3.3.2 – Содержание пакета Models**

|  |  |
| --- | --- |
| Класс User – описание пользователей | |
| getId | Метод получения id пользователя |
| setId | Метод установки id пользователя |
| getEmail | Метод получения Email пользователя |
| getUserName | Метод получения имени пользователя |
| getImage | Метод получения строкового представления изображения |
| Класс Message – описание сообщений | |
| getIdReceiver | Метод получения Id получателя |
| setIdReceiver | Метод установки Id получателя |
| getImage | Метод получения строкового представления изображения |
| getText | Метод получения текста сообщения |
| getId | Метод получения Id сообщения |

Пакет Activity хранит в себе классы для отображения данных (см. таблицу 3.3.3).

**Таблица 3.3.3 – Содержание пакета Activity**

|  |  |
| --- | --- |
| Класс RegisterActivity – Регистрация пользователей | |
| OnCreate | Метод содержащий в себе слушатели событий для кнопок «Select Photo» и «Register» |
| OnActivityResult | Метод позволяющий выбрать из галереи изображение и установить его в ImageView |
| registerClick | Метод регистрирующий пользователей |
| ImageToString | Метод позволяющий преобразовать изображение в сроковый тип |
| Класс LoginActivity – Регистрация пользователей | |
| OnCreate | Метод содержащий в себе слушатель событий для кнопки «Login» |
| loginClick | Метод авторизующий пользователей |
| Класс UserActivity – отображение пользователей | |
| onCreate | Метод обрабатывающий и отображающий данные с сервера |
| validate | Метод преобразующий данные с сервера |
| StringToImage | Метод преобразующий строковый тип в изображение |
| Класс ChatActivityLog – отображение и отправка сообщений | |
| onCreate | Метод обрабатывающий и отображающий данные с сервера |
| validate | Метод преобразующий данные с сервера |
| StringToImage | Метод преобразующий строковый тип в изображение |
| SendMessage | Метод отправляющий сообщения |

Пакет Server хранит в себе вспомогательные классы для работы с сервером (см. таблицу 3.3.4).

**Таблица 3.3.4 – Содержание пакета Server**

|  |  |
| --- | --- |
| Класс AccesTokenClass – преобразование токена | |
| toString | Метод выполняющий преобразование токена в строковый тип |
| Класс CustomError – отображение описания ошибок | |
| error\_description | Поле содержащее описание ошибок |

Ознакомится с диаграммой классов разработанного приложения можно на рисунке 3.3.2.

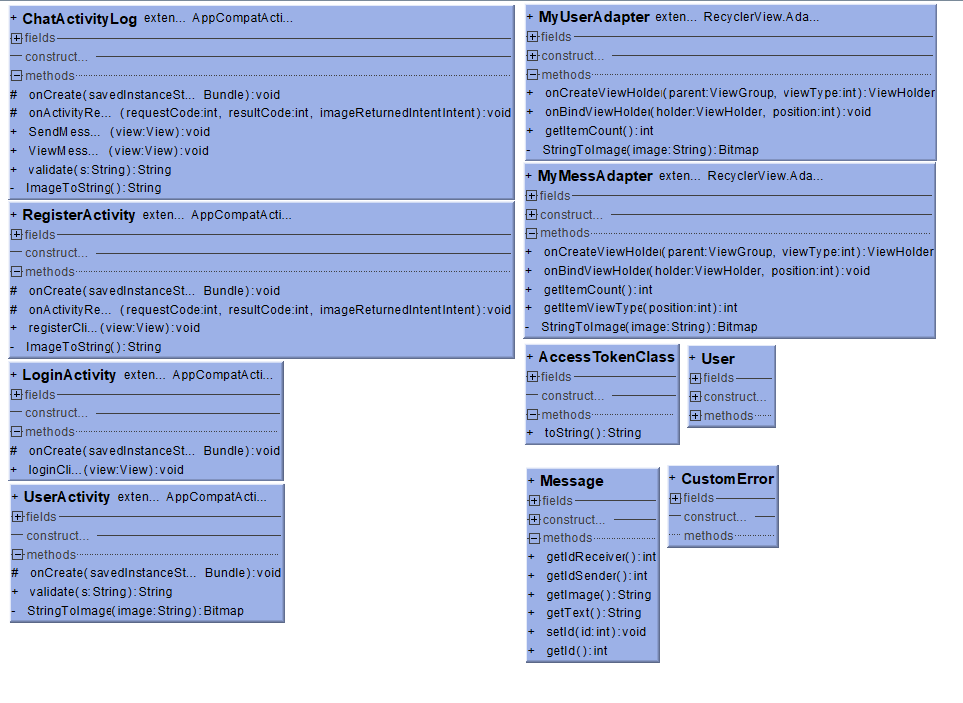


Рисунок 3.3.2 – Диаграмма классов

# **3.4 Разработка сервера**

Сервер было реализовано в среде разработки – Microsoft Visual Studio 2019. Для написания приложения был использован язык программирования C#.

Графическая структура проекта представлена на рисунке 3.4.1.

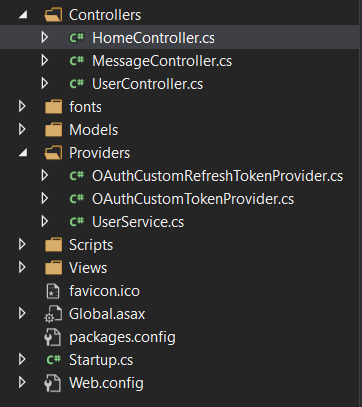


Рисунок 3.4.1 – Графическая структура проекта

Папка Controllers хранит в себе классы контроллеров обрабатывающие поступающие запросы к серверу(см. таблицу 3.4.1). ). Ознакомится с кодом класса MessageController можно в Приложении В.

**Таблица 3.4.1 – Содержание папки Controllers**

|  |  |
| --- | --- |
| Класс HomeController – контроллер по умолчанию | |
| Index | Метод отображающий главную страницу |
| Класс MessageController – контроллер сообщений | |
| SendMessage | Метод отправляющий и записывающий сообщения в базу данных |
| GetMyMessage | Метод возвращающий все сообщения конкретного пользователя из базы данных |
| Класс UserController – контроллер пользователей | |
| CreateUser | Метод создающий пользователей |
| GetUsers | Метод возвращающий всех пользователей |

Папка Providers хранит в себе классы, которые используются для авторизации и аутентификации пользователя (см. таблицу 3.4.2).

**Таблица 3.4.2 – Содержание пакета Providers**

|  |  |
| --- | --- |
| Класс UserService – обработка списка пользователей | |
| Validate | Метод осуществляющий поиск пользователей по логину и паролю |
| GetUserList | Метод возвращающий список пользователей |
| Класс OAuthCustomTokenProvider– авторизация пользователей | |
| GrantResourceOwnerCredentials | Метод проверяет подлинность клиента и учетные данные владельца ресурса |
| ValidateClientAuthentication | Метод проверяет действительность пользователя |
| TokenEndpoint | Метод выдачи токена |
| Класс OAuthCustomRefreshTokenProvider – создание и удаление токена | |
| Сreate | Метод обработки ошибок при создании токена |
| CreateAsync | Метод создания токена |
| Receive | Метод обработки ошибок при удаление токена |
| ReceiveAsync | Метод удаления токена |

Также можно ознакомиться с основными классами, которые не входят ни в одну из папок на таблице 3.4.3.

**Таблица 3.4.3 – Основные классы**

|  |
| --- |
| Класс Startup – определяет конфигурацию приложения |

# **3.5 Вывод по главе «Программная реализация приложений»**

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных на стороне сервера и клиента.

Третий этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы, в частности.

Четвертый этап – это разработка серверного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы.

# **4 Тестирование разработонного продукта**

Процесс тестирования продукта представляет собой столь же неотъемлемую часть процесса разработки, как и проектирование. Тестирование позволяет оценить качество разрабатываемого продукта.

# **4.1 Негативное тестирование**

В рамках курсового проекта будет выполнено негативное тестирование. Негативное тестирование – это один из видов тестирования, который позволяет проверить систему на некорректное поведение. В ходе такого тестирования мы можем узнать, что система справится с непредвиденными ситуациями.

Предусмотрена следующая обработка исключительных ситуаций:

* в случае ввода неверного опечатка пальца, не верного отпечатка пальца 5 раз, отмены ввода отпечатка пальца предусмотрена обработка данных ситуаций (рисунок 4.1.1).



Рисунок 4.1.1 – Обработка ошибок в аутентификации

* В случае ввода не корректных данных на странице авторизации будет всплывать предупреждающее сообщение (рис. 4.1.2).

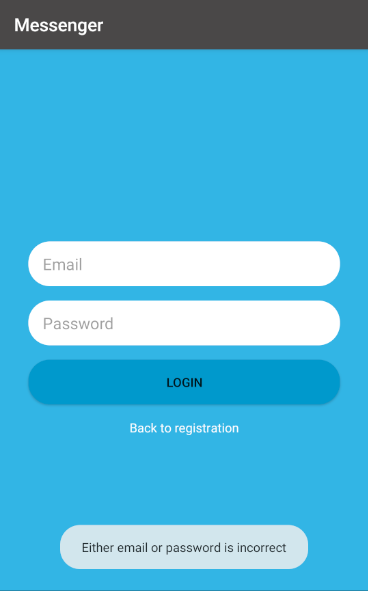


Рисунок 4.1.2 – Предупреждающее сообщение на странице авторизации

* Будет всплывать предупреждающее сообщение, если хотя бы одно из полей или все поля не заполнены при нажатии на кнопку «Login». (рис. 4.1.3)

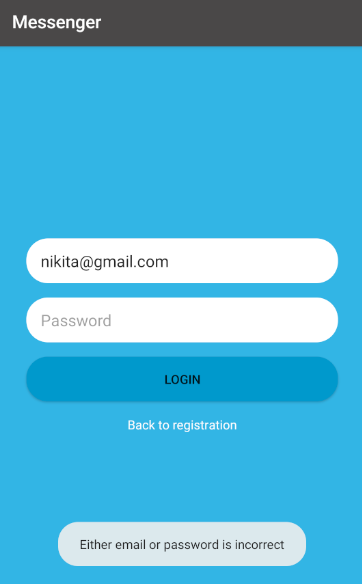


Рисунок 4.1.3 – Предупреждающие сообщения на странице авторизации

* Появится всплывающее сообщение «Either email or password is incorrect» также, если нет зарегистрированного пользователя с введёнными «Email» и «Password».
* На странице регистрации кнопка «Register» не будет активна пока не произойдет выборка из галереи изображения пользователем. (рис.4.1.4)

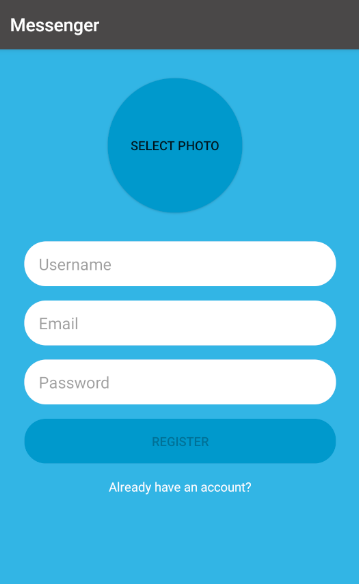


Рисунок 4.1.4 – Не активная кнопка на странице регистрации

* Появится всплывающее сообщение «fill in all fields» если будут пустые поля на странице регистрации.

# **4.2 Выводы по главе «Тестирование разработонного продукта»**

В данной главе было рассмотрено негативное тестирование наиболее уязвимых компонентов мобильного приложения.

Были произведены проверки: на существование пользователя в базе данных, на введение корректных данных.

Также на странице регистрации были проведены ограничения на выборку изображения для профиля и заполнения всех полей регистрации.

# **5 Руководство пользователя**

Для того чтобы установить приложения на телефон необходимо иметь версию Android минимум 8.0 и версию API не ниже 26. Версию Android и API вашего телефона можно узнать в настройках телефона.

Если версия Android подходит, тогда можно устанавливать приложение на мобильный телефон. В меню ваших приложений или на главном экране смартфона появится новая иконка с именем «Chat». Запустим приложение нажатием на иконку приложения. Откроется главная страница с аутентификацией по отпечатку пальца. Если ваш отпечаток пальца успешно подошел, тогда появится кнопка «Entry» (рис. 5.1).

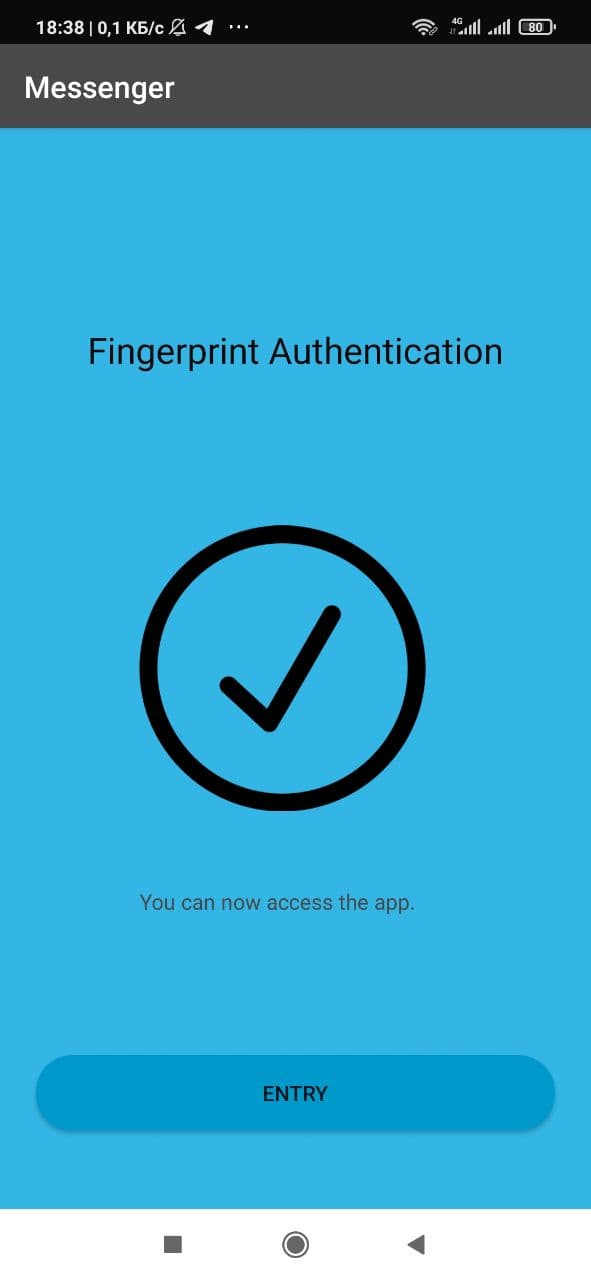


Рисунок 5.1 – Главная страница

Для работы с приложением, сначала нужно пройти регистрацию. (рис. 5.2). После выбора изображения из галереи и заполнения всех полей нужно нажать на кнопку «Register».

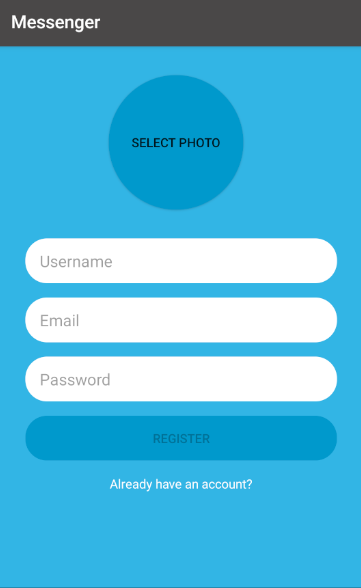


Рисунок 5.2 – Страница регистрации

После нажатия на кнопку «Register» произойдет переход на страницу авторизации, где нужно заполнить два поля «Email» и «Password».

Страница авторизации представлена на рисунке 5.3.

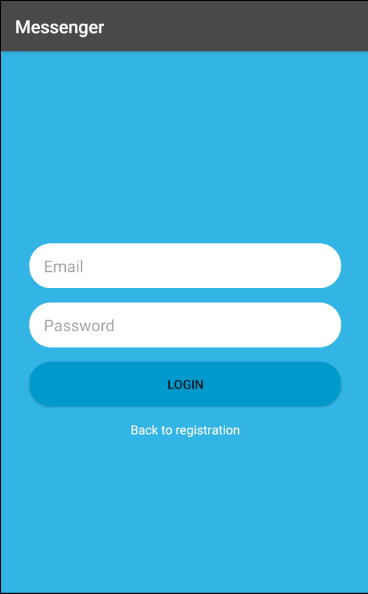


Рисунок 5.3 – Страница авторизации

После заполнения полей произойдет переход на страницу контактов, где будут отображены все зарегистрированные пользователи. (рис. 5.4).

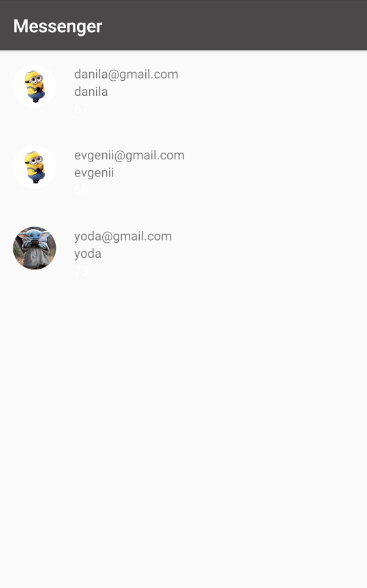


Рисунок 5.4 – Страница контактов

Для того чтобы начать обмен информацией между пользователем, требуется нажать по нужному вам пользователю. Далее произойдет переход на страницу чата. (рис.5.5)

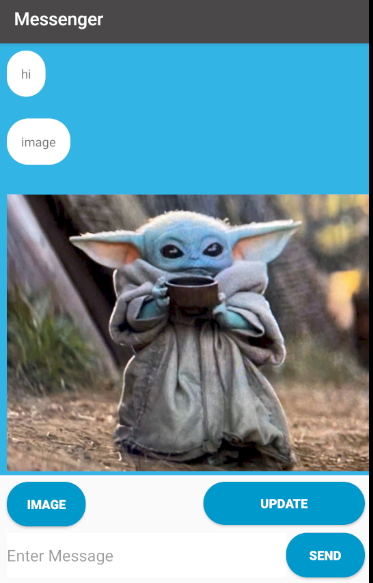


Рисунок 5.5 – Страница чата

Для того чтобы отправить текстовое сообщения нажмите на поле «Enter Message», далее вам нужно нажать на кнопку «Send». Чтобы отобразить сообщения, нужно нажать на кнопку «Update». Если вы хотите отправить изображение вам нужно нажать кнопку «Image» у вас отроется галерея и вы сможете выбрать нужное вам изображение, далее вам нужно нажать на кнопку «Send». Также вы можете отправить текстовое сообщение и изображение.

# **5.1 Выводы по главе «Руководство пользователя»**

В данной главе было описано руководство пользователя.

Данное руководство позволяет пользователю узнать требования для установки приложения на телефон, а также как использовать все функции приложения. Были предоставлены скриншоты самого приложения для большей наглядности использования.

# **ЗАКЛЮЧЕНИЕ**

Целью моего курсового проекта было разработка средства обмена сообщениями, чтобы люди могли обмениваться текстовыми сообщениями и изображениями на любом расстоянии.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено 3 приложения и были выявлены их положительные и отрицательные стороны. Эта информация была использована для улучшения функционала приложения и пользовательского интерфейса.

* проектирование приложения;

Был спроектирован каждый отдельный компонент приложения. В мобильном приложении был определен основной функционал, также были разработаны необходимые диаграммы. При проектировании базы данных, были определены необходимые таблицы и разработана схема базы данных. В серверном приложении был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных и HTTP методы для работы с серверной частью.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: СУБД SQLite, сред разработки Android Studio и Microsoft Visual Studio, языков программирования Java и C#, модульной платформы .NET Core, библиотек Entity Framework, SQLCipher, OkHttp, Newtonsoft.Json.

Были реализованы базы данных на стороне сервера и клиента.

При решении данной задачи была достигнута поставленная цель, и был создан проект «Средство обмена сообщениями», который включает в себя android-клиент и сервер.

* тестирование разработанного продукта.

Для реализации данной задачи проведено негативное тестирование наиболее уязвимых компонентов мобильного приложения.

* разработка руководства пользователя.

Разработано руководство, позволяющее пользователю узнать требования для установки приложения на телефон, а также как использовать все функции мобильного приложения.

Мобильное приложение имеет простой, интуитивно понятный пользователю интерфейс и обладает следующими особенностями:

* регистрация пользователей;
* авторизация пользователей;
* отправка и получение текстовых сообщений;
* отправка и получение изображений;
* отображение списка контактов;
* шифрование данных;
* управление транзакциями;

В соответствии с поставленными задачами и полученным результатом можно сделать вывод, что задачи были выполнены в полном объеме.

# **Список использованных источников**

1 Хабрахабр [Электронный ресурс]. – Архитектура REST – TM, 2006-2019. – Режим доступа: https://habrahabr.ru/post/38730 – Дата доступа: 29.04.2019.

1. Программирование под ОС Андроид [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/java/android. Дата доступа: 06.04.2018
2. СountDownTimer Tutorial [Электронный ресурс] – codinginflow.com – Режим доступа: https://codinginflow.com/tutorials/android/countdowntimer/part-4-time-input. – Дата доступа: 23.02.2020
3. Руководство по ASP.NET Core 3 [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sharp/aspnet5. Дата доступа: 23.01.2020
4. Руководство по MySQL [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sql/mysql. Дата доступа: 26.05.2018

**ПРИЛОЖЕНИЕ А**

Код UserController.cs

using Newtonsoft.Json;

using Romanovich.Models;

using System;

using System.Collections.Generic;

namespace Romanovich.Controllers

{

public class UserController : ApiController

{

[HttpPost]

public HttpResponseMessage CreateUser([FromBody] Users user)

{

RomanovichEntities db = new RomanovichEntities();

if (db.Users.FirstOrDefault(x => x.UserName == user.UserName) == null && db.Users.FirstOrDefault(x => x.Email == user.Email) == null)

{

db.AddUser(user.UserName, user.Email, user.Password, user.isAdmin, user.Image);

return new HttpResponseMessage(HttpStatusCode.OK);

}

else

{

return new HttpResponseMessage(HttpStatusCode.MovedPermanently);

}

}

[Authorize]

[HttpGet]

public string GetUsers()

{

RomanovichEntities db = new RomanovichEntities();

ClaimsPrincipal principal = Request.GetRequestContext().Principal as ClaimsPrincipal;

var Name = ClaimsPrincipal.Current.Identity.Name;

db.Configuration.LazyLoadingEnabled = false;

var res = db.Users.ToList();

var toDelete = db.Users.FirstOrDefault(x => x.UserName == Name);

res.Remove(toDelete);

return JsonConvert.SerializeObject(res);

}

}

}

**ПРИЛОЖЕНИЕ Б**

Код OAuthCustomTokenProvider.cs

namespace Romanovich.Providers

{

public class OAuthCustomTokenProvider : OAuthAuthorizationServerProvider

{

public override System.Threading.Tasks.Task GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext context)

{

return System.Threading.Tasks.Task.Factory.StartNew(() =>

{

String UserRole;

RomanovichEntities db = new RomanovichEntities();

var email = context.UserName;

var password = context.Password;

var userService = new UserService();

var user = userService.Validate(email, password);

if (user != null)

{

var claims = new List<Claim>()

{

new Claim(ClaimTypes.Sid, Convert.ToString(user.id)),

new Claim(ClaimTypes.Name, user.UserName)

};

if (user.IsAdmin == 1)

{

UserRole = "Admin";

}

else

{

UserRole = "User";

}

claims.Add(new Claim(ClaimTypes.Role, UserRole));

var data = new Dictionary<string, string>

{

{ "userName", user.UserName },

{ "roles", string.Join(",", UserRole)}

};

var properties = new AuthenticationProperties(data);

ClaimsIdentity oAuthIdentity = new ClaimsIdentity(claims,

Startup.OAuthOptions.AuthenticationType);

var ticket = new AuthenticationTicket(oAuthIdentity, properties);

context.Validated(ticket);

}

else

{

context.SetError("invalid\_grant", "Either email or password is incorrect");

}

});

}

**ПРИЛОЖЕНИЕ В**

Код MessageController.cs

using Newtonsoft.Json;

using Romanovich.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Security.Claims;

using System.Web.Http;

namespace Romanovich.Controllers

{

public class MessageController : ApiController

{

[HttpPost]

[Authorize]

public HttpResponseMessage SendMessage([FromBody]Message message)

{

RomanovichEntities db = new RomanovichEntities();

ClaimsPrincipal principal = Request.GetRequestContext().Principal as ClaimsPrincipal;

var Name = ClaimsPrincipal.Current.Identity.Name;

var sender = db.Users.FirstOrDefault(x => x.UserName == Name).id;

var reciever = db.Users.FirstOrDefault(x => x.UserName == message.reciever).id;

DateTime date = DateTime.Now;

db.SendMessage(sender, reciever, message.text, message.image, date);

return new HttpResponseMessage(HttpStatusCode.OK);

}

[HttpGet]

[Authorize]

public string GetMyMessage(string fr)

{

RomanovichEntities db = new RomanovichEntities();

db.Configuration.LazyLoadingEnabled = false;

ClaimsPrincipal principal = Request.GetRequestContext().Principal as ClaimsPrincipal;

var Name = ClaimsPrincipal.Current.Identity.Name;

var im = db.Users.FirstOrDefault(x => x.UserName == Name).id;

var hs = db.Users.FirstOrDefault(x => x.UserName == fr);

var messages = db.Messages.Where(x => (x.IdSender == im && x.IdReceiver == hs.id) || (x.IdSender == hs.id && x.IdReceiver == im)).OrderBy(x => x.Date).ToList();

List<ExtendedMessage> normalizedMessages = new List<ExtendedMessage>();

messages.ForEach(value => normalizedMessages.Add(new ExtendedMessage { id = value.id, IdSender = value.IdSender, IdReceiver = value.IdReceiver, Text = value.Message1, Date = value.Date, Image = value.Image }));

return JsonConvert.SerializeObject(normalizedMessages);

}

}

}