МИНИСТЕРСТВО 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:**

«MyContactManager»

Выполнил студент Сташкевич Александр Валентинович

(Ф.И.О.)

Руководитель проекта ст. преподаватель Северинчик Н.А. (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: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ст. преподаватель Северинчик Н.А (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc42611655)

[1 Обзор аналогов приложений «Viber» 4](#_Toc42611656)

[1.1 Приложение «Viber» 4](#_Toc42611657)

[1.2 Приложение «WhatsApp» 5](#_Toc42611658)

[1.4 Выводы по главе «Обзор аналогов приложений «MyContactManager» 6](#_Toc42611659)

[2 Проектирование программного продукта 7](#_Toc42611660)

[2.1 Проектирование мобильного приложения 7](#_Toc42611661)

[2.2 Проектирование базы данных 8](#_Toc42611662)

[2.3 Проектирование сервера 9](#_Toc42611663)

[2.4 Выводы по главе «Проектирование программного продукта» 9](#_Toc42611664)

[3 Программная реализация приложений 9](#_Toc42611665)

[3.1 Программные средства разработки 9](#_Toc42611666)

[3.2 Разработка веб-приложения 10](#_Toc42611667)

[3.3 Разработка мобильного приложения 11](#_Toc42611668)

[3.4 Разработка сервера 13](#_Toc42611669)

[3.5 Вывод по главе «Программная реализация приложений» 14](#_Toc42611670)

[4 Руководство пользователя 15](#_Toc42611671)

[4.1 Выводы по главе «Руководство пользователя» 24](#_Toc42611672)

[ЗАКЛЮЧЕНИЕ 25](#_Toc42611673)

[Список использованных источников 26](#_Toc42611674)

[**ПРИЛОЖЕНИЕ А** 27](#_Toc42611675)

[**ПРИЛОЖЕНИЕ В** 35](#_Toc42611676)

# ВВЕДЕНИЕ

Цель курсового проекта: предоставление возможности обмена сообщениями между человеком и его контактами.

Задачи курсового проекта:

1. Изучить аналогичные программные средства по теме курсового проекта.

Выполнение данной задачи поможет нам сформировать требования к разрабатываемому продукту, определить, какие функциональные возможности предоставит пользователям мобильное приложение, а также выявить недостатки конкурентов, которые мы можем исправить.

1. Спроектировать приложение.

Проектирование приложения позволит получить более качественный, продуманный и проработанный в комплексе продукт и сэкономить время, которое уходило бы на внесение правок в уже готовом приложении.

1. Реализовать приложение.

В курсовом проекте будет реализовано мобильное приложение с возможностью создания контактов, передачи сообщений между ними, совершения звонков. В век развития информационных технологий этот вариант будет более оптимальный и удобный, так как вся информация будет хранится в одном приложении на телефоне, и пользователь легко сможет им воспользоваться, где бы он не был.

1. Разработать руководство пользователя.

Цель создания документа заключается в том, чтобы предоставить пользователю возможность самостоятельно решать свои прикладные задачи с помощью системы. Этой цели может служить и введение в предметную область, и ознакомление со всеми возможностями программы, и описание конкретных процедур решения задач, и приведение различных инструкций. Иногда Руководство пользователя больше похоже на справочник, к которому можно обращаться в процессе работы, а иногда – на учебник, который позволяет изучить принципы работы с программой и ее возможности, а затем применять их на практике

# 1 Обзор аналогов приложений «Viber»

# Приложение «Viber»

"Viber" – это мобильное приложение для совершения звонков, а также отправки сообщений, не только текстовых, но и видео, фото, аудио сообщений.(рисунок 1.1.1).



Рисунок 1.1.1 – Приложение «Viber»

Возможности приложения:

* Голосовое общение с высоким качеством
* Видеозвонки
* Отправка сообщений, снимков и стикеров
* Прикрепление данных о своем местонахождении
* Групповые чаты и звонки
* Передача медиаконтента
* Push-уведомления

Данные характеристики безусловно являются плюсами, однако здесь отсутствует возможность просмотра статистики по отправленным и полученным сообщениям, что может быть полезно, если вы следите за временем, которое проводите на экраном смартфона, а также за своей социальной активностью.

# 1.2 Приложение «WhatsApp»

WhatsApp — это приложение для популярных систем, способное работать и в виде онлайн-сервиса. Необходим для обмена сообщениями, создания групповых чатов и совершения звонков. (рисунок 1.1.2).

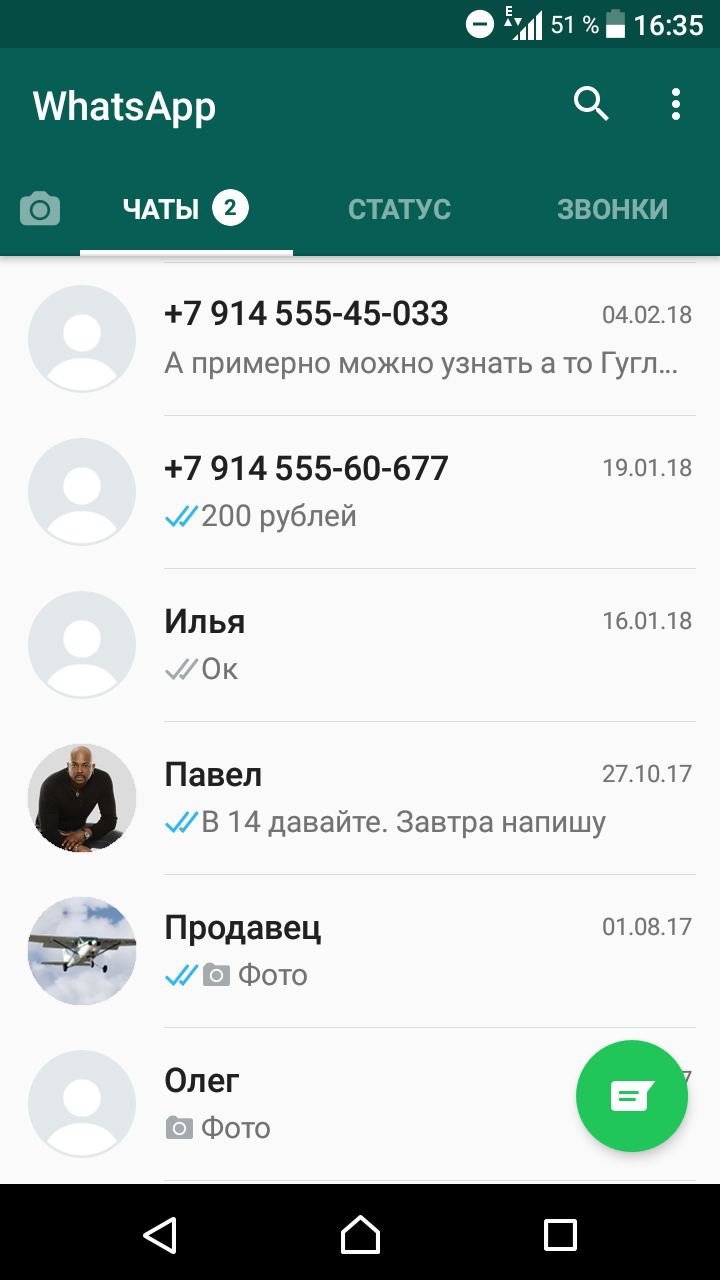


Рисунок 1.1.2 – Приложение «WhatsApp»

Основные характеристики:

* Бесплатный обмен сообщениями и совершение звонков;
* Отправка и получение фотографий, видео и голосовых сообщений;
* Групповые чаты;
* Интеграция с адресной книгой телефона;
* Быстрый переход из контактов телефона к чатам;

# 1.4 Выводы по главе «Обзор аналогов приложений «MyContactManager»

В данной главе были изучены два приложения, которые являются аналогами приложения, – это «Viber», «WhatsApp». В конечном результате были подведены выводы:

Дизайн у приложений получился очень лаконичным. Также функционал приложений покрывает практически все нужды современного человека в плане общения. Из этого можно сделать вывод, что дизайн для будущего приложения необходимо делать простой и лаконичный. Также функционал моего приложения должен покрывать основные запросы пользователя.

Плюсы функционала приложения «Viber»: большой набор стикеров, возможность устраивать групповые звонки и чаты.

Минусы функционала приложения «Viber»: нет просмотра статистики по отправленным сообщениям, дублирование контактов в записной книге.

Плюсы функционала приложения «WhatsApp»: простота в использовании, отсутствие рекламы, бесплатные международные звонки.

Минусы функционала приложения «WhatsApp»: перемешивание контактов в записных книгах.

# 2 Проектирование программного продукта

# 2.1 Проектирование мобильного приложения

В рамках курсового проекта необходимо разработать мобильное приложение, предоставляющее пользователю возможность создания, просмотра, удаления и редактирования контактов; возможность отправки сообщений по сети интернет, а также по мобильной связи. Также необходимо разработать веб-приложение, предоставляющее пользователю возможность регистрации и авторизации, отправки сообщений, просмотра статистки по отправленным и полученным сообщениям.

Первым этапом в использовании приложения будет создание нового контакта или выбор из уже существующих. Для этой цели будет создан список всех контактов и форма для создания нового. При заполнении всех полей и нажатии на кнопку подтверждения должен быть добавлен новый контакт, а также он должен отобразиться в списке всех контактов.

При нажатии на выбранный контакт должно открываться меню выбранного контакта. В нём должны быть доступны функции вызова, отправки сообщения, изменения или удаления контакта.

При нажатии на кнопку отправки сообщения должен открываться чат с историей сообщений между двумя пользователями. Также должна быть предоставлена возможность ввода и отправки сообщения.

При нажатии на кнопку вызова должен происходить автоматический набор номера через приложение «Телефон».

Также из любого места приложения должна быть доступна кнопка «Internet messages», нажатие на которую открывает режим авторизации для доступа к отправке сообщений через интернет.

После авторизации или регистрации должен отобразиться список всех контактов. При нажатии на один из них должен открыться чат, в котором можно написать сообщение и просмотреть своё.

Таким образом приложение должно выполнять следующие функции:

* Предоставление списка всех контактов;
* Возможность добавления нового контакта;
* Редактирование или удаление контакта;
* Возможность совершения звонков;
* Отправка сообщения по сети Интернет и по мобильной связи;
* Возможность регистрации и авторизации пользователей;
* Просмотр статистики по отправленным и полученным сообщениям;

# 2.2 Проектирование базы данных

В базе данных данного веб-приложения должны быть реализованы 2 таблицы.

В таблице «User» 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, не должен содержать значений null;
* Password – строковый тип данных, максимальной длинной, не должен содержать значений null;
* IsAdmin – логический тип данных;

В таблице «Message» хранится информация о сообщениях:

* id – целочисленный тип данных, не должен содержать значений null, является первичным ключом;
* id\_sender – целочисленный тип данных, не должен содержать значений null;
* id\_receiver – целочисленный тип данных, не должен содержать значений null;
* text – строковый тип данных, максимальной длинной, не должен содержать значений null;
* date – тип даты.

Схема базы данных представлена на рисунке 2.2.1.

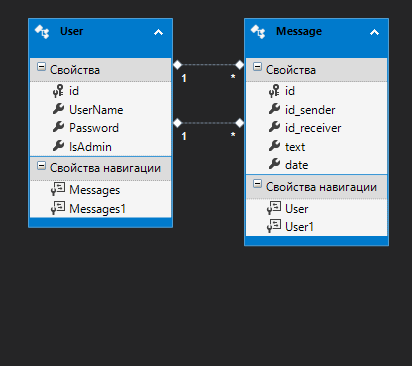


Рисунок 2.2.1 – Схема базы данных

# 2.3 Проектирование сервера

Серверная часть приложения должна предоставлять API для доступа к каждой хранимой процедуре 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 какую процедуру необходимо выполнить.

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. Метод PUT для обновления. DELETE для удаления ресурсов.

# 2.4 Выводы по главе «Проектирование программного продукта»

В данной главе было рассмотрено проектирование всех компонентов приложения.

По части мобильного приложения был определен основной функционал приложения.

Была спроектирована база данных, где были определены необходимые таблицы, поля для каждой из них. 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 приложений**

**3.1 Программные средства разработки**

Для разработки приложений курсового проекта были использованы следующие технические средства:

Android Studio – интегрированная среда разработки для работы с платформой Android [1].

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств [2].

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft [3].

Java – строго типизированный объектно-ориентированный язык программирования [4].

C# – объектно-ориентированный язык программирования [5].

.NET – это модульная платформа для разработки программного обеспечения с открытым исходным кодом [6].

Entity Framework – специальная объектно-ориентированная технология на базе фреймворка .NET для работы с данными [7].

OkHttp – библиотека предоставляющая собой простой и удобный клиент для работы с HTTP запросами [8].

Newtonsoft.Json – библиотека позволяющую преобразовывать объекты в JSON формат с большим количеством настроек [9].

Firebase Auth – предоставляет бэкенд, простой в использовании SDK и готовые библиотеки пользовательского интерфейса для реализации аутентификации пользователей в приложении [10].

**3.2 Разработка веб-приложения**

Отсутствие возможности у контроллера API генерировать HTML-разметку из представлений является причиной, по которой в одностраничных приложениях комбинируются стандартные приемы ASP.NET MVC Framework с Web API. Инфраструктура ASP.NET MVC Framework выполняет шаги, требуемые для доставки HTML-содержимого пользователю (включая аутентификацию, авторизацию, выбор и визуализацию представления). После того, как HTML-содержимое доставлено в браузер, запросы Ajax, генерируемые содержащимся внутри кодом JavaScript, будут обрабатываться контроллером Web API.

Определение контроллера Web API отличается от обычного контроллера MVC. Потому что контроллеры Web API применяют стиль REST (Representation State Transfer или "передача состояния представления").

REST-архитектура предполагает применение следующих методов или типов запросов HTTP для взаимодействия с сервером: GET, POST, PUT, DELETE.

REST-архитектура предполагает применение следующих методов или типов запросов HTTP для взаимодействия с сервером: GET, POST, PUT, DELETE.

Графическая структура классов представлена на рисунке 3.2.1.

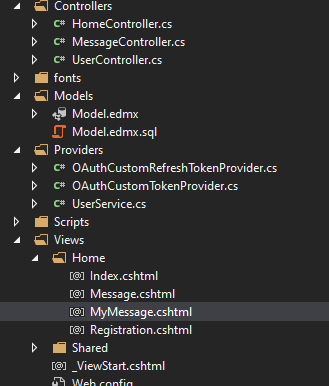


Рисунок 3.2.1 – Схема базы данных

Код классов, которые реализуют REST технологию, а также AJAX-запросов представлен в Приложении А.

**3.3 Разработка мобильного приложения**

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

Графическая структура проекта представлена на рисунке 3.3.1.

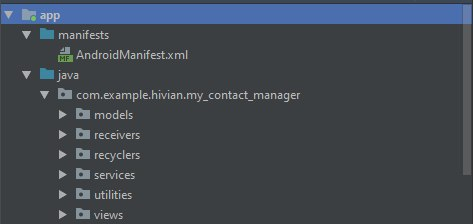


Рисунок 3.3.1 – Графическая структура проекта

Проект в себе содержит папку java, в которой организована удобная структура подпапок для различных функций мобильного приложения.

1. Models – содержит классы, которые организовывают работу с базой данных.  
   1.1. DBHandler – класс, описывающий работу с базой данных. В него входит создание таблиц, удаление, добавление и т.д.  
   1.2. Contact – класс, описывающий поля для контакта  
   1.3. Message – класс, описывающий поля для сообщений  
   1.4. Sms – класс, описывающий поля для содержимого сообщений  
   1.5. User – класс, описывающий поля для пользователя приложения
2. SmsReceiver – класс, представляющий собой передатчик сообщений по мобильной связи.
3. Recyclers – папка, в которой находятся классы-обработчики для контактов и смс.
4. Services – в данной папке находится класс SmsNotificationService, который служит для работы с уведомлениями и класс FingerPrintService для аутентификации по отпечатку пальца.
5. Utilities – папка, в которой хранятся различные классы-утилиты.  
   5.1. AccessTokenClass – класс для получения токена из приходящего JSON  
   5.2. BitmapUtility – класс для работы с Bitmap
6. Views – Содержит в себе классы активностей.  
   6.1. ContactCreationActivity – класс активности создания контакта  
   6.2. ContactEditionActivity - класс активности изменения контакта  
   6.3. ContactSmsActivity – класс активности чата  
   6.4. InternetContactList – класс активности пользователей при отправке сообщений через интернет

В папке layout находятся активности, которые представляют собой xml файлы, содержащие разметку. С помощью этой разметки в свою очередь отрисовываются макеты экранов, которые мы будем видеть при запуске приложения.

Все активности похожи своим наполнением, при этом отрисовываются по-разному, то есть используя похожую xml-разметку удается воспроизводить различные элементы на экране и обеспечить взаимодействие пользователя с программой. Также в мобильном приложении реализован запрет на осуществление снимков экрана и вход по отпечатку пальца.

# 3.4 Разработка сервера

Для взаимодействия мобильного приложения с сервером применяется архитектурный стиль REST. Основная задача сервера в этой архитектуре предоставить клиентам доступ к ресурсам по их идентификаторам. Под доступом подразумевается как получение информации, так и ее изменение или удаление. В зависимости от реализации сервиса, в ответ на наш запрос придёт ответ в формате JSON. На рисунке 3.4 представлен пример одного из post-запросов.

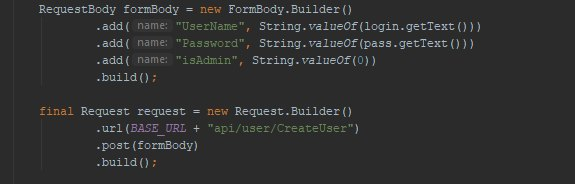


Рисунок 3.4 – post-запрос

В данном участке кода мы можем увидеть, как формируется тело POST запроса, а также происходит отправка на Base\_Url веб-приложения.

В данном случае мы заносим в тело запроса UserName, Password и проверку на администратора. В объекте request важную роль играют методы url, post и build.

На рисунке 3.4.2 приведен пример get запроса.

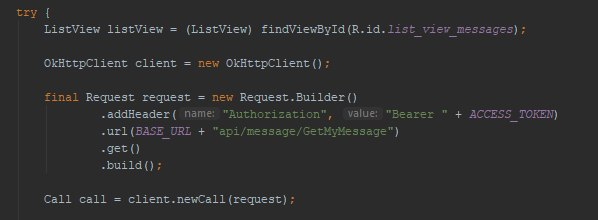


Рисунок 3.4.2– get-запрос

В данном случае также формируется объект request, но появляется новый метод addHeader, в котором мы указываем заголовок GET-запроса.

GET – позволяет получить данные, а также конкретный ответ по заданным параметрам. POST – позволяет передать данные на сервер.

Таким образом можно сказать, что REST – это взаимодействие «клиент-серверной» архитектуры с помощью HTTP-запросов и HTTP-ответов по протоколу, не подтверждающему состояние.

# 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 веб-приложения и сервера.

Третий этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрены основные папки с классами.

Четвертый этап – разработка REST.

# 4 Руководство пользователя

Для начала необходимо установить приложение на мобильное устройство с API не меньше 25 (Android 7.1.1 Nougat). Запустим наше приложение. Сразу после запуска мы увидим главный экран приложения, на котором отображены созданные контакты, а также кнопки для добавления нового контакта и перехода в режим отправки сообщений через интернет (Рисунок 4.1).

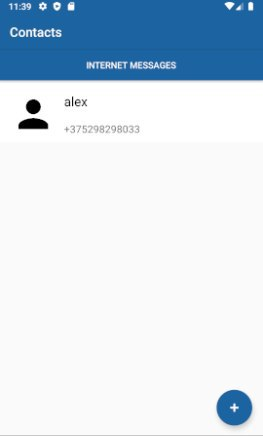


Рисунок 4.1 – главный экран

Если пользователь еще не создавал контакты, то ему необходимо нажать на кнопку «плюс», и он попадёт на экран создания контакта (Рисунок 4.2).

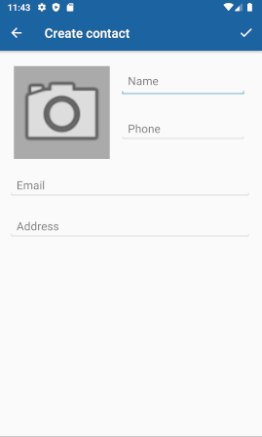


Рисунок 4.2 – экран создания контакта

Если контакт успешно создан, то пользователь снова попадает на главный экран. После этого он может выбрать определенный контакт и попадёт в меню выбранного контакта (Рисунок 4.3).

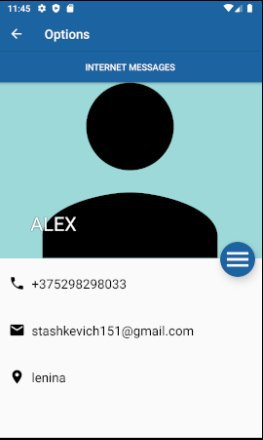


Рисунок 4.3 – экран контакта

При нажатии на кнопку «гамбургер» происходит раскрытие выпадающего меню, где пользователь может выбрать определенные действия, например, изменение данных о контакте, удаление контакта, звонок или сообщение через смс. (Рисунок 4.4).

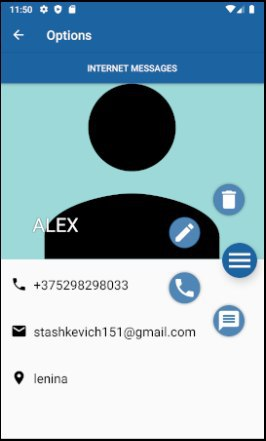


Рисунок 4.4 – выпадающее меню

Если нажать на кнопку удаления контакта (урна), то появится предупреждение о том, что контакт будет безвозвратно удалён.

Если нажать на кнопку изменения (карандаш), то мы попадём на экран редактирования контакта, схожий с экраном создания (Рисунок 4.2).

При нажатии на кнопку чата (всплывающее сообщение) пользователь попадёт на экран чата (Рисунок 4.5).

При нажатии на кнопку звонка (телефонная трубка), мы попадём на экран вызова (Рисунок 4.6).

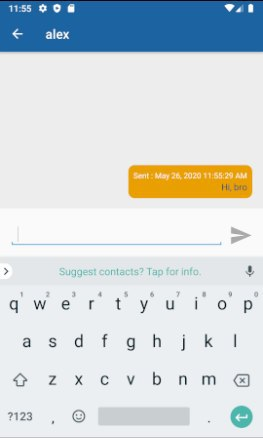


Рисунок 4.5 – чат

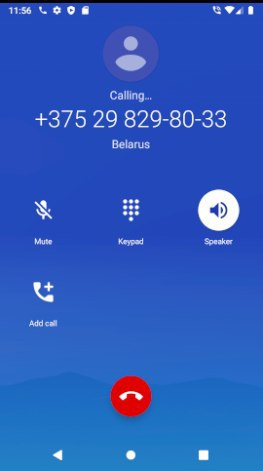


Рисунок 4.6 – экран вызова

После завершения вызова мы снова попадаем на экран контакта (Рисунок 4.3). Здесь мы можем нажать на кнопку INTERNET MESSAGES для доступа к отправке сообщений через интернет. Данная кнопка доступна практически из любого места программы. При нажатии на эту кнопку, мы попадаем на экран регистрации и аутентификации (Рисунок 4.7).

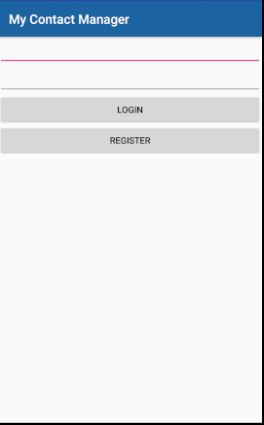


Рисунок 4.7 – экран регистрации и аутентификации

Если пользователь не зарегистрирован, ему необходимо нажать на кнопку «REGISTER», придумать логин и пароль, а потом ввести их в соответствующие поля. После успешной регистрации/логина, пользователь попадает в меню интернет сообщений (Рисунок 4.8)

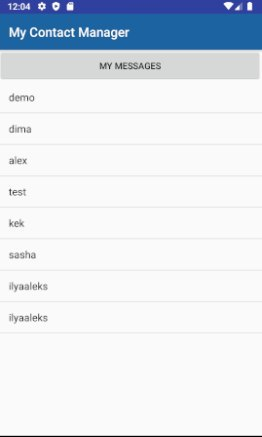


Рисунок 4.8 – меню интернет сообщений

Пользователь может выбрать любого другого человека и отправить ему сообщение (Рисунок 4.9), а может нажать на кнопку «MY MESSAGES», которая приведёт его на экран просмотра сообщений (Рисунок 4.10)

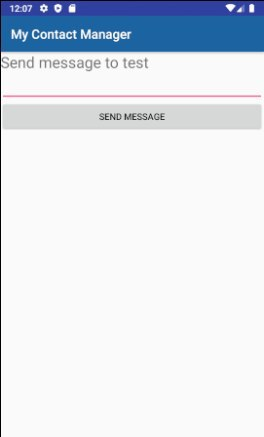


Рисунок 4.9 – отправка сообщения

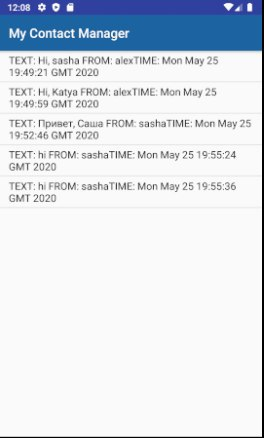


Рисунок 4.10 – просмотр сообщений

Теперь рассмотрим работу веб-приложения. Для размещения веб-приложения в сети Интернет был использован ресурс Azure. При первом попадании на страницу веб-приложения перед нами открывается страница регистрации и аутентификации (Рисунок 4.11).

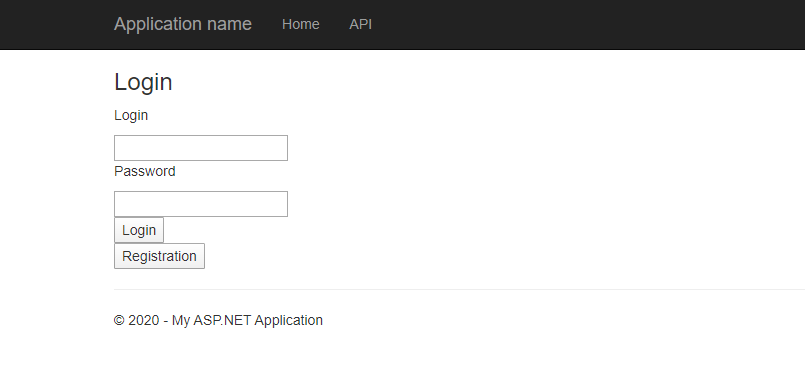


Рисунок 4.11 – экран регистрации и аутентификации

Если пользователь не зарегистрирован, ему необходимо нажать на кнопку «Registration», придумать логин и пароль, а потом ввести их в соответствующие поля. После успешной регистрации/логина, пользователь попадает в меню своего аккаунта (Рисунок 4.12)

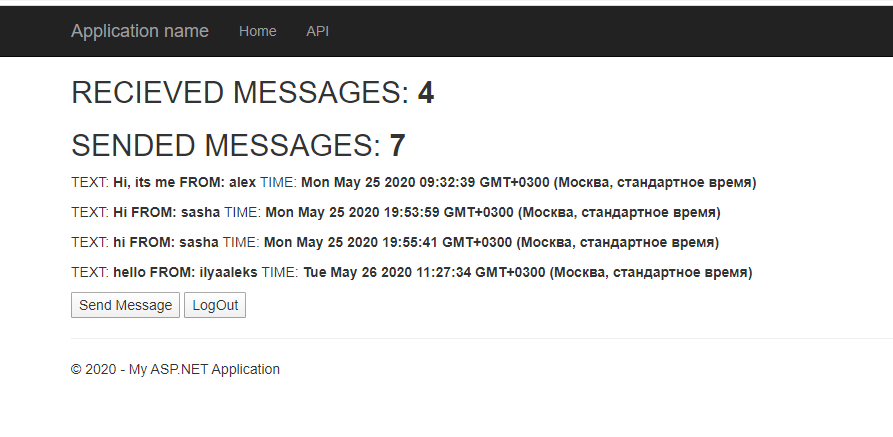


Рисунок 4.12 – данные аккаунта

Здесь мы можем видеть статистку по отправленным и полученным сообщениям, а также полную информацию по каждому сообщению. Из этой страницы мы можем сделать выход из аккаунта, а также отправку сообщения через интернет (Рисунок 4.13).

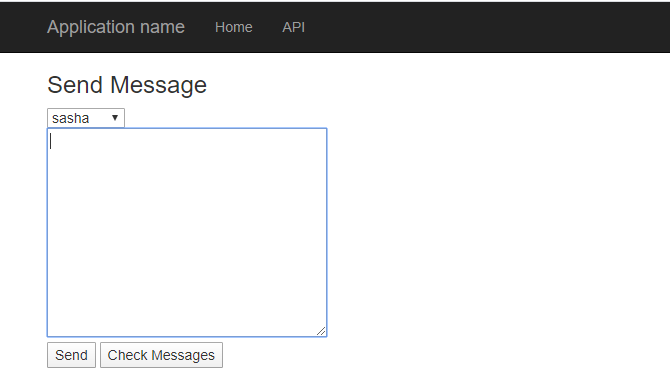


Рисунок 4.13 – меню отправки сообщения

Исходя из составленного руководства пользователя, можно смело говорить о том, что интерфейс приложения является дружелюбным, несложным и сам подсказывает пользователю, какие действия он может совершить.

# 4.1 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 предоставление возможности обмена сообщениями между пользователями и их контактами.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено 2 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. 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: среда разработки Android Studio и Microsoft Visual Studio, языков программирования Java и C#, модульной платформы .NET, библиотек Entity Framework, OkHttp.

Была реализована база данных на стороне сервера.

При решении данной задачи была достигнута поставленная цель, и был создан проект «MyContactManager», который включает в себя android-клиент и серверное веб-приложение.

* 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 интерфейс и обладает следующими особенностями:

* Предоставление списка всех контактов;
* Возможность добавления нового контакта;
* Редактирование или удаление контакта;
* Возможность совершения звонков;
* Отправка сообщения по сети Интернет и по мобильной связи;
* Возможность регистрации и авторизации пользователей;
* Просмотр статистики по отправленным и полученным сообщениям;

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

# Список использованных источников

1. Android Studio [Электронный ресурс]. / Android Developers – Режим доступа:<https://developer.android.com/studio> – Дата доступа: 29.04.2020
2. Visual Studio 2017 [Электронный ресурс]. / Microsoft 2020 – Режим доступа: <https://visualstudio.microsoft.com/ru/vs/older-downloads/> – Дата доступа: 29.04.2020
3. Microsoft SQL Server [Электронный ресурс]. / Microsoft 2020 – Режим доступа: https://www.microsoft.com/en-us/sql-server/sql-server-downloads – Дата доступа: 29.04.2020
4. The Java Language Specification [Электронный ресурс]. / James Gosling; Bill Joy; Guy Steele; Gilad Bracha; Alex Buckley; Daniel Smith – Режим доступа: https://docs.oracle.com/javase/specs/jls/se12/html/index.html – Дата доступа: 29.04.2020
5. C# 6.0 draft specification [Электронный ресурс]. / microsoft.com – Режим доступа: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction – Дата доступа: 29.04.2020
6. .NET Tutorial - Hello World in 10 minutes [Электронный ресурс]. / microsoft.com – Режим доступа: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/intro – Дата доступа: 29.04.2020
7. Database First Entity Framework 6 [Электронный ресурс] / Metanit.com – Режим доступа: https://metanit.com/sharp/entityframework/2.4.php. Дата доступа: 26.05.2019
8. OkHttp [Электронный ресурс] / okkhttp.com – Режим доступа: https://square.github.io/okhttp/. Дата доступа: 26.05.2019
9. Newtonsoft [Электронный ресурс] / newtonsoft.com – Режим доступа: https://www.newtonsoft.com/json / Дата доступа: 26.05.2019
10. Firebase Authentification [Электронный ресурс] / google developers – Режим доступа: https://firebase.google.com/docs/auth / Дата доступа: 26.05.2019
11. Build Web APIs using ASP.NET [Электронный ресурс]. / courses.edx.com – Режим доступа: https://courses.edx.org/courses/course- v1:Microsoft+DEV247x+3T2018/course – Дата доступа: 29.04.2020
12. Руководство по Web API [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sharp/mvc/12.1.php. Дата доступа: 23.01.2020

**ПРИЛОЖЕНИЕ А**

using Newtonsoft.Json;

using Stashkevich.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace Stashkevich.Controllers

{

public class UserController : ApiController

{

public class Users

{

public string UserName { get; set; }

public string Password { get; set; }

public int isAdmin { get; set; }

};

[HttpPost]

public HttpResponseMessage CreateUser([FromBody] Users user)

{

stashkevichEntities db = new stashkevichEntities();

db.CreateUser(user.UserName, user.Password, user.isAdmin);

return new HttpResponseMessage(HttpStatusCode.OK);

}

[Authorize]

[HttpGet]

public string GetUsers()

{

stashkevichEntities db = new stashkevichEntities();

return JsonConvert.SerializeObject(db.GetUsers());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

using Server.Models;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace Server.Controllers

{

[Route("api/[controller]")]

public class InsertEventController : Controller

{

CourseProjectContext dbContext = new CourseProjectContext();

// POST api/<controller>

[HttpPost]

public void Post([FromBody]Events value)

{

Events events = new Events();

events.CId = value.CId;

events.Title = value.Title;

events.Description = value.Description;

events.Date = value.Date;

try

{

dbContext.Add(events);

dbContext.SaveChanges();

}

catch (Exception e)

{

}

}

// PUT api/<controller>/5

//[HttpPut("{id}")]

public string Put([FromBody]Events value)

{

Events events = dbContext.Events.Where(u => u.EId.Equals(value.EId)).First();

//events.EId = value.EId;

//events.CId = value.CId;

events.Title = value.Title;

events.Description = value.Description;

events.Date = value.Date;

try

{

dbContext.Update(events);

dbContext.SaveChanges();

return JsonConvert.SerializeObject("User has been registered!");

}

catch (Exception ex)

{

return JsonConvert.SerializeObject(ex.Message);

}

}

// DELETE api/<controller>/5

[HttpDelete("{id}")]

public void Delete(int id)

{

try

{

Events events = dbContext.Events.FirstOrDefault(u => u.EId.Equals(id));

dbContext.Events.Remove(events);

dbContext.SaveChangesAsync();

}

catch (Exception e)

{

}

}

}

}

using Stashkevich.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;

using Microsoft.AspNet.Identity;

using Newtonsoft.Json;

using Stashkevich.Providers;

namespace Stashkevich.Controllers

{

public class MessageController : ApiController

{

public class Message

{

public string text { get; set; }

public int reciever { get; set; }

};

[HttpPost]

[Authorize]

public HttpResponseMessage SendMessage([FromBody]Message message)

{

stashkevichEntities db = new stashkevichEntities();

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

var Name = ClaimsPrincipal.Current.Identity.Name;

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

DateTime date = DateTime.Now;

db.CreateMessage(sender, message.reciever, message.text + " FROM: " + Name, date);

return new HttpResponseMessage(HttpStatusCode.OK);

}

[HttpGet]

[Authorize]

public string GetMyMessage()

{

stashkevichEntities db = new stashkevichEntities();

db.Configuration.LazyLoadingEnabled = false;

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

var Name = ClaimsPrincipal.Current.Identity.Name;

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

return JsonConvert.SerializeObject(db.Messages.OrderBy(x => x.date).Where(x => x.id\_receiver == im));

}

[HttpGet]

[Authorize]

public string GetMyStat()

{

stashkevichEntities db = new stashkevichEntities();

db.Configuration.LazyLoadingEnabled = false;

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

var Name = ClaimsPrincipal.Current.Identity.Name;

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

Stat stat = new Stat

{

recv = db.Messages.Where(x => x.id\_receiver == im).Count(),

send = db.Messages.Where(x => x.id\_sender == im).Count()

};

return JsonConvert.SerializeObject(stat);

}

}

}

<h3>Login</h3>

<div><p>Login</p><input type="text" name="UserName" id="UserName" required> </div>

<div><p>Password</p><input type="password" name="Password" id="Password" required> </div>

<div><button onclick="signIn()" />Login</div>

<div><button onclick="back()" />Registration</div>

<script>

function back() {

location.href = "/Home/Registration";

}

signIn = function () {

var tokenKey = "tokenInfo";

var data = {

grant\_type: 'password',

username: $("#UserName").val(),

password: $("#Password").val()

};

$.ajax({

async: false,

type: 'POST',

url: '/token',

contentType: 'application/x-www-form-urlencoded; charset=UTF-8',

dataType: 'json',

data: data

}).done(function (data) {

var expire = new Date();

expire.setHours(expire.getHours() + 4);

document.cookie = "tokenInfo=" + data.access\_token + "; path=/; expires=" + expire.toUTCString() + ";";

location.href = "/Home/MyMessage";

})

}

function getCookiePartByKey(key) {

var value = "; " + document.cookie;

var parts = value.split("; " + key + "=");

if (parts.length == 2) return parts.pop().split(";").shift();

}

function IfLoginedRedirectToMain() {

var tokenKey = "tokenInfo";

if (getCookiePartByKey(tokenKey) != null) location.href = "/Home/MyMessage";

}

IfLoginedRedirectToMain();

IfLoginedRedirectToMain();

</script>

<h3>Send Message</h3>

<select id="select"></select>

<div><textarea id="messageText"></textarea></div>

<button onclick="sendMessage()">Send</button>

<button onclick="back()">Check Messages</button>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>

<script>

$(document).ready(function () {

var tokenKey = "tokenInfo";

$.ajax({

type: 'GET',

url: '/api/user/GetUsers',

contentType: 'application/json; charset=UTF-8',

beforeSend: function (xhr) {

var token = getCookiePartByKey(tokenKey);

xhr.setRequestHeader("Authorization", "Bearer " + token);

}

}).done(function (data) {

var obj = jQuery.parseJSON(data);

console.log(obj);

for (i = 0; i < obj.length; i++) {

$("#select").append('<option value="' + obj[i].id + '">' + obj[i].UserName + '</option>')

}

})

});

getUsers = function () {

var tokenKey = "tokenInfo";

$.ajax({

type: 'GET',

url: '/api/user/GetUsers',

contentType: 'application/json; charset=UTF-8',

beforeSend: function (xhr) {

var token = getCookiePartByKey(tokenKey);

xhr.setRequestHeader("Authorization", "Bearer " + token);

}

}).done(function (data) {

var obj = jQuery.parseJSON(data);

console.log(obj);

})

}

signIn = function () {

var tokenKey = "tokenInfo";

var data = {

grant\_type: 'password',

username: $("#UserName").val(),

password: $("#Password").val()

};

$.ajax({

async: false,

type: 'POST',

url: '/token',

contentType: 'application/x-www-form-urlencoded; charset=UTF-8',

dataType: 'json',

data: data

}).done(function (data) {

var expire = new Date();

expire.setHours(expire.getHours() + 4);

document.cookie = "tokenInfo=" + data.access\_token + "; path=/; expires=" + expire.toUTCString() + ";";

})

}

function getCookiePartByKey(key) {

var value = "; " + document.cookie;

var parts = value.split("; " + key + "=");

if (parts.length == 2) return parts.pop().split(";").shift();

}

sendMessage = function () {

var tokenKey = "tokenInfo";

var data = {

text: $("#messageText").val(),

reciever: $("#select").val()

}

$.ajax({

type: 'POST',

url: '/api/message/SendMessage',

contentType: 'application/json; charset=UTF-8',

data: JSON.stringify(data),

beforeSend: function (xhr) {

var token = getCookiePartByKey(tokenKey);

xhr.setRequestHeader("Authorization", "Bearer " + token);

}

})

}

function back() {

location.href = "/Home/MyMessage";

}

</script>

@{

ViewBag.Title = "MyMessage";

}

<div id="stat"></div>

<div id="wrapper"></div>

<button onclick="back()">Send Message</button>

<button onclick="logout()">LogOut</button>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>

<script>

function logout() {

document.cookie.split(";").forEach(function (el) {

el = el.split("=")[0].trim();

if (!el.indexOf("tokenInfo")) {

document.cookie = el + "=; path=/; expires=Thu, 01 Jan 1970 00:00:00 UTC";

location.href = "/Home/Index";

}

})

}

$(document).ready(function () {

var tokenKey = "tokenInfo";

$.ajax({

async: false,

type: 'GET',

url: '/api/message/GetMyMessage',

contentType: 'application/json; charset=UTF-8',

beforeSend: function (xhr) {

var token = getCookiePartByKey(tokenKey);

xhr.setRequestHeader("Authorization", "Bearer " + token);

}

}).done(function (data) {

var obj = jQuery.parseJSON(data);

console.log(obj);

for (i = 0; i < obj.length; i++) {

var date = new Date(Date.parse(obj[i].date));

$('#wrapper').append('<p> TEXT: <b>' + obj[i].text + '</b> TIME: <b>' + date + '</b> </p>');

}

})

});

$(document).ready(function () {

var tokenKey = "tokenInfo";

$.ajax({

async: false,

type: 'GET',

url: '/api/message/GetMyStat',

contentType: 'application/json; charset=UTF-8',

beforeSend: function (xhr) {

var token = getCookiePartByKey(tokenKey);

xhr.setRequestHeader("Authorization", "Bearer " + token);

}

}).done(function (data) {

var obj = jQuery.parseJSON(data);

console.log(obj);

$('#stat').append('<h2> RECIEVED MESSAGES: <b>' + obj.recv + '</b> </h2>' + '<h2> SENDED MESSAGES: <b>' + obj.send + '</b></h2>');

})

});

function getCookiePartByKey(key) {

var value = "; " + document.cookie;

var parts = value.split("; " + key + "=");

if (parts.length == 2) return parts.pop().split(";").shift();

}

function back() {

location.href = "/Home/Message";

}

</script>

<h3>Register</h3>

<div><p>UserName</p><input type="text" name="UserName" id="UserName" required> </div>

<div><p>Password</p><input type="password" name="Password" id="Password" required> </div>

<div><button onclick="register()" />Login</div>

<script>

register = function () {

var tokenKey = "tokenInfo";

var data = {

UserName: $("#UserName").val(),

Password: $("#Password").val(),

isAdmin: 0

}

$.ajax({

type: 'POST',

url: '/api/user/CreateUser',

contentType: 'application/json; charset=UTF-8',

data: JSON.stringify(data),

beforeSend: function (xhr) {

var token = getCookiePartByKey(tokenKey);

xhr.setRequestHeader("Authorization", "Bearer " + token);

}

}).done(function (data) {

location.href ="/Home/Index"

})

}

function getCookiePartByKey(key) {

var value = "; " + document.cookie;

var parts = value.split("; " + key + "=");

if (parts.length == 2) return parts.pop().split(";").shift();

}

function IfLoginedRedirectToMain() {

var tokenKey = "tokenInfo";

if (getCookiePartByKey(tokenKey) != null) location.href = "/Home/MyMessage";

}

IfLoginedRedirectToMain();

IfLoginedRedirectToMain();

</script>

**ПРИЛОЖЕНИЕ В**

package com.example.hivian.my\_contact\_manager.models.db;  
  
import android.content.ContentValues;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.content.Context;  
import android.database.sqlite.SQLiteDatabase;  
  
import com.example.hivian.my\_contact\_manager.models.Contact;  
import com.example.hivian.my\_contact\_manager.models.Sms;  
  
import java.util.ArrayList;  
import java.util.List;  
  
  
  
public class DBHandler extends SQLiteOpenHelper {  
 // Database Version  
 private static final int *DB\_VERSION* = 1;  
 // Database Name  
 private static final String *DB\_NAME* = "DB";  
 // Contacts table name  
 private static final String *CONTACTS\_TABLE* = "contacts";  
 private static final String *SMS\_TABLE* = "sms";  
 // Shops Table Columns names  
 private static final String *KEY\_ID* = "id";  
 private static final String *KEY\_IMAGE* = "image";  
 private static final String *KEY\_NAME* = "name";  
 private static final String *KEY\_PHONE* = "phone";  
 private static final String *KEY\_EMAIL* = "email";  
 private static final String *KEY\_ADDRESS* = "address";  
 private static final String *KEY\_SMS\_HEADER* = "sms\_header";  
 private static final String *KEY\_SMS\_CONTENT* = "sms\_content";  
 private static final String *KEY\_CONTACT\_ID* = "contact\_id";  
 private static final String *KEY\_SMS\_TYPE* = "sms\_type";  
 private static final String *CONTACTS\_TABLE\_CREATE* =  
 "CREATE TABLE " + *CONTACTS\_TABLE* + " (" + *KEY\_ID* + " INTEGER PRIMARY KEY, " +  
 *KEY\_IMAGE* + " BLOB, " + *KEY\_NAME* + " TEXT, " + *KEY\_PHONE* + " TEXT, " +  
 *KEY\_EMAIL* + " TEXT, " + *KEY\_ADDRESS* + " TEXT)";  
 private static final String *SMS\_TABLE\_CREATE* =  
 "CREATE TABLE " + *SMS\_TABLE* + " (" + *KEY\_ID* + " INTEGER PRIMARY KEY, " +  
 *KEY\_SMS\_HEADER* + " TEXT, " + *KEY\_SMS\_CONTENT* + " TEXT, " +  
 *KEY\_CONTACT\_ID* + " INTEGER," + *KEY\_SMS\_TYPE* + " INTEGER)";  
  
 private static DBHandler *dbInstance* = null;  
  
 public static DBHandler getInstance(Context context) {  
 if (*dbInstance* == null) {  
 *dbInstance* = new DBHandler(context, *DB\_NAME*, *DB\_VERSION*);  
 }  
 return *dbInstance*;  
 }  
  
 DBHandler(Context context, String dbName, int dbVersion) {  
 super(context, dbName, null, dbVersion);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 db.execSQL(*CONTACTS\_TABLE\_CREATE*);  
 db.execSQL(*SMS\_TABLE\_CREATE*);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 // Drop older table if existed  
 db.execSQL("DROP TABLE IF EXISTS " + *CONTACTS\_TABLE*);  
 db.execSQL("DROP TABLE IF EXISTS " + *SMS\_TABLE*);  
 // Creating tables again  
 onCreate(db);  
 }  
  
 // Adding new contact  
 public void addContact(Contact contact) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
  
 values.put(*KEY\_IMAGE*, contact.getImage());  
 values.put(*KEY\_NAME*, contact.getName());  
 values.put(*KEY\_PHONE*, contact.getPhone());  
 values.put(*KEY\_EMAIL*, contact.getEmail());  
 values.put(*KEY\_ADDRESS*, contact.getAddress());  
 db.insert(*CONTACTS\_TABLE*, null, values);  
 }  
  
 // Adding new sms  
 public void addSms(Sms sms) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
  
 values.put(*KEY\_SMS\_HEADER*, sms.getHeader());  
 values.put(*KEY\_SMS\_CONTENT*, sms.getContent());  
 values.put(*KEY\_CONTACT\_ID*, sms.getContactId());  
 values.put(*KEY\_SMS\_TYPE*, sms.getType());  
 db.insert(*SMS\_TABLE*, null, values);  
 }  
  
 // Getting one contact  
 public Contact getContact(int id) {  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.query(*CONTACTS\_TABLE*, new String[]  
 { *KEY\_ID*, *KEY\_IMAGE*, *KEY\_NAME*, *KEY\_PHONE*, *KEY\_EMAIL*, *KEY\_ADDRESS* },  
 *KEY\_ID* + "=?", new String[] { String.*valueOf*(id) }, null, null, null, null);  
  
 if (cursor != null)  
 cursor.moveToFirst();  
 Contact contact = new Contact(cursor.getBlob(1), cursor.getString(2),  
 cursor.getString(3), cursor.getString(4), cursor.getString(5));  
 contact.setId(cursor.getInt(0));  
 cursor.close();  
 return contact;  
 }  
  
 // Getting one contact by name  
 public Contact getContactByName(String name) {  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.query(*CONTACTS\_TABLE*, new String[]  
 { *KEY\_ID*, *KEY\_IMAGE*, *KEY\_NAME*, *KEY\_PHONE*, *KEY\_EMAIL*, *KEY\_ADDRESS* },  
 *KEY\_NAME* + "=?", new String[] { name }, null, null, null, null);  
  
 if (cursor != null)  
 cursor.moveToFirst();  
 Contact contact = new Contact(cursor.getBlob(1), cursor.getString(2),  
 cursor.getString(3), cursor.getString(4), cursor.getString(5));  
 contact.setId(cursor.getInt(0));  
 cursor.close();  
 return contact;  
 }  
  
 // Getting one contact by phone  
 public Contact getContactByPhone(String phone) {  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.query(*CONTACTS\_TABLE*, new String[]  
 { *KEY\_ID*, *KEY\_IMAGE*, *KEY\_NAME*, *KEY\_PHONE*, *KEY\_EMAIL*, *KEY\_ADDRESS* },  
 *KEY\_PHONE* + "=?", new String[] { phone }, null, null, null, null);  
 if (!cursor.moveToFirst())  
 return null;  
 Contact contact = new Contact(cursor.getBlob(1), cursor.getString(2),  
 cursor.getString(3), cursor.getString(4), cursor.getString(5));  
 contact.setId(cursor.getInt(0));  
 cursor.close();  
 return contact;  
 }  
  
 // Getting one sms by contact id  
 public List<Sms> getSmsByContactId(Integer id) {  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.query(*SMS\_TABLE*, new String[]  
 { *KEY\_ID*, *KEY\_SMS\_HEADER*, *KEY\_SMS\_CONTENT*, *KEY\_CONTACT\_ID*, *KEY\_SMS\_TYPE* },  
 *KEY\_CONTACT\_ID* + "=?", new String[] { String.*valueOf*(id) }, null, null, null, null);  
 List<Sms> allSms = new ArrayList<>();  
  
 if (cursor.moveToFirst()) {  
 do {  
 Sms sms = new Sms();  
 sms.setId(Integer.*parseInt*(cursor.getString(0)));  
 sms.setHeader(cursor.getString(1));  
 sms.setContent(cursor.getString(2));  
 sms.setContactId(cursor.getInt(3));  
 sms.setType(cursor.getInt(4));  
 allSms.add(sms);  
 } while (cursor.moveToNext());  
 }  
 cursor.close();  
 return allSms;  
 }  
  
 // Getting all contacts  
 public List<Contact> getAllContacts() {  
 String selectQuery = "SELECT \* FROM " + *CONTACTS\_TABLE*;  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.rawQuery(selectQuery, null);  
 List<Contact> allContacts = new ArrayList<>();  
  
 if (cursor.moveToFirst()) {  
 do {  
 Contact contact = new Contact();  
 contact.setId(Integer.*parseInt*(cursor.getString(0)));  
 contact.setImage(cursor.getBlob(1));  
 contact.setName(cursor.getString(2));  
 contact.setPhone(cursor.getString(3));  
 contact.setEmail(cursor.getString(4));  
 contact.setAddress(cursor.getString(5));  
 allContacts.add(contact);  
 } while (cursor.moveToNext());  
 }  
 cursor.close();  
 return allContacts;  
 }  
  
 // Getting all sms  
 public List<Sms> getAllSms() {  
 String selectQuery = "SELECT \* FROM " + *SMS\_TABLE*;  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.rawQuery(selectQuery, null);  
 List<Sms> allSms = new ArrayList<>();  
  
 if (cursor.moveToFirst()) {  
 do {  
 Sms sms = new Sms();  
 sms.setId(Integer.*parseInt*(cursor.getString(0)));  
 sms.setHeader(cursor.getString(1));  
 sms.setContent(cursor.getString(2));  
 sms.setContactId(cursor.getInt(3));  
 sms.setType(cursor.getInt(4));  
 allSms.add(sms);  
 } while (cursor.moveToNext());  
 }  
 cursor.close();  
 return allSms;  
 }  
  
 // Getting all sms from contact  
 public List<Sms> getAllSmsFromContact(int id) {  
 String selectQuery = "SELECT \* FROM " + *SMS\_TABLE* + " WHERE " +  
 *KEY\_CONTACT\_ID* + " = " + id;  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.rawQuery(selectQuery, null);  
 List<Sms> allSms = new ArrayList<>();  
  
 if (cursor.moveToFirst()) {  
 do {  
 Sms sms = new Sms();  
 sms.setId(Integer.*parseInt*(cursor.getString(0)));  
 sms.setHeader(cursor.getString(1));  
 sms.setContent(cursor.getString(2));  
 sms.setContactId(cursor.getInt(3));  
 sms.setType(cursor.getInt(4));  
 allSms.add(sms);  
 } while (cursor.moveToNext());  
 }  
 cursor.close();  
 return allSms;  
 }  
  
 // Getting contacts count  
 public Integer getContactsCount() {  
 String countQuery = "SELECT \* FROM " + *CONTACTS\_TABLE*;  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.rawQuery(countQuery, null);  
  
 return cursor.getCount();  
 }  
  
 // Checking duplicates in table  
 public boolean isDuplicate(DBHandler db, String name) {  
 List<Contact> contacts = db.getAllContacts();  
 for (Contact cont : contacts) {  
 if (cont.getName().equals(name))  
 return true;  
 }  
 return false;  
 }  
  
 // Updating a contact  
 public Integer updateContact(Contact contact) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
  
 values.put(*KEY\_IMAGE*, contact.getImage());  
 values.put(*KEY\_NAME*, contact.getName());  
 values.put(*KEY\_PHONE*, contact.getPhone());  
 values.put(*KEY\_EMAIL*, contact.getEmail());  
 values.put(*KEY\_ADDRESS*, contact.getAddress());  
 return db.update(*CONTACTS\_TABLE*, values, *KEY\_ID* + " = ?",  
 new String[]{String.*valueOf*(contact.getId())});  
 }  
  
 // Updating a sms  
 public Integer updateSms(Sms sms) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
  
 values.put(*KEY\_SMS\_HEADER*, sms.getHeader());  
 values.put(*KEY\_SMS\_CONTENT*, sms.getContent());  
 values.put(*KEY\_CONTACT\_ID*, sms.getContactId());  
 values.put(*KEY\_SMS\_TYPE*, sms.getType());  
 return db.update(*SMS\_TABLE*, values, *KEY\_ID* + " = ?",  
 new String[]{String.*valueOf*(sms.getId())});  
 }  
  
 // Deleting a contact  
 public void deleteContact(Contact contact) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 db.delete(*CONTACTS\_TABLE*, *KEY\_ID* + " = ?",  
 new String[] { String.*valueOf*(contact.getId()) });  
 }  
  
 // Deleting a contact  
 public void deleteSms(Sms sms) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 db.delete(*SMS\_TABLE*, *KEY\_ID* + " = ?",  
 new String[] { String.*valueOf*(sms.getId()) });  
 }  
  
 // Deleting all contacts  
 public void deleteAllContacts(DBHandler db) {  
 List<Contact> contacts = db.getAllContacts();  
  
 for (Contact cont : contacts) {  
 db.deleteContact(cont);  
 }  
 }  
  
 // Deleting all sms  
 public void deleteAllSms(DBHandler db) {  
 List<Sms> allSms = db.getAllSms();  
  
 for (Sms sms : allSms) {  
 db.deleteSms(sms);  
 }  
 }  
  
}

package com.example.hivian.my\_contact\_manager.receivers;  
  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
import android.telephony.SmsMessage;  
import android.util.Log;  
  
import com.example.hivian.my\_contact\_manager.models.Contact;  
import com.example.hivian.my\_contact\_manager.models.Sms;  
import com.example.hivian.my\_contact\_manager.models.db.DBHandler;  
import com.example.hivian.my\_contact\_manager.recyclers.sms.SmsData;  
import com.example.hivian.my\_contact\_manager.services.SmsNotificationService;  
import com.example.hivian.my\_contact\_manager.views.activities.ContactSmsActivity;  
  
import java.text.DateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.Locale;  
  
  
  
public class SmsReceiver extends BroadcastReceiver {  
 private static final String *SMS\_RECEIVED* = "android.provider.Telephony.SMS\_RECEIVED";  
 private static final String *PDUS* = "pdus";  
 public static final String *SMS\_HEADER* = "sms\_header";  
 public static final String *SMS\_CONTENT* = "sms\_content";  
 public static final String *SMS\_TYPE* = "sms\_type";  
 public static final String *NOTIF\_TITLE* = "notification\_title";  
 public static final String *NOTIF\_MESSAGE* = "notification\_message";  
  
 Contact contact;  
 StringBuilder sb;  
  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 if (intent.getAction().equals(*SMS\_RECEIVED*)) {  
 DBHandler db = DBHandler.*getInstance*(context);  
  
 Bundle bundle = intent.getExtras();  
 if (bundle != null) {  
 Object[] pdus = (Object[]) bundle.get(*PDUS*);  
 if (pdus.length == 0) {  
 return;  
 }  
 SmsMessage[] messages = new SmsMessage[pdus.length];  
 sb = new StringBuilder();  
 for (int i = 0; i < pdus.length; i++) {  
 messages[i] = SmsMessage.*createFromPdu*((byte[]) pdus[i]);  
 sb.append(messages[i].getMessageBody());  
 }  
  
 String message = sb.toString();  
 String phone = messages[0].getOriginatingAddress();  
 contact = db.getContactByPhone(phone);  
 if (contact == null) {  
 phone = messages[0].getOriginatingAddress();//.replace("+33", "0");  
 contact = db.getContactByPhone(phone);  
 if (contact == null) {  
 db.addContact(new Contact(null, phone.replaceAll("\\s+",""),  
 phone.replaceAll("\\s+",""), "", ""));  
 }  
 }  
  
 DateFormat df = DateFormat.*getDateTimeInstance*(DateFormat.*MEDIUM*,  
 DateFormat.*MEDIUM*, Locale.*getDefault*());  
 contact = db.getContactByPhone(phone);  
 Sms sms = new Sms(df.format(new Date()), message, contact.getId(), Sms.*RECEIVED*);  
 db.addSms(sms);  
  
 Intent intentService = new Intent(context, SmsNotificationService.class);  
 intentService.putExtra(*NOTIF\_TITLE*, contact.getName());  
 intentService.putExtra(*NOTIF\_MESSAGE*, message);  
 context.startService(intentService);  
  
 Intent intentBroadcast = new Intent(ContactSmsActivity.*SMS\_RECEIVED*);  
 intentBroadcast.putExtra(*SMS\_HEADER*, sms.getHeader());  
 intentBroadcast.putExtra(*SMS\_CONTENT*, sms.getContent());  
 intentBroadcast.putExtra(*SMS\_TYPE*, sms.getType());  
 context.sendBroadcast(intentBroadcast);  
  
 }  
 }  
 }  
}

package com.example.hivian.my\_contact\_manager.services;  
  
import android.app.IntentService;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.Intent;  
import android.support.annotation.Nullable;  
import android.support.v4.app.NotificationCompat;  
  
import com.example.hivian.my\_contact\_manager.models.Contact;  
import com.example.hivian.my\_contact\_manager.receivers.SmsReceiver;  
import com.example.hivian.my\_contact\_manager.views.activities.ContactSmsActivity;  
import com.example.hivian.my\_contact\_manager.models.db.DBHandler;  
import com.example.hivian.my\_contact\_manager.utilities.BitmapUtility;  
import com.example.hivian.my\_contact\_manager.R;  
  
  
public class SmsNotificationService extends IntentService {  
 private final static String *SERVICE\_NAME* = "smsService";  
 private DBHandler db;  
  
 public SmsNotificationService() {  
 super(*SERVICE\_NAME*);  
 db = DBHandler.*getInstance*(this);  
 }  
  
 @Override  
 protected void onHandleIntent(@Nullable Intent intent) {  
 String contentTitle = (String) intent.getExtras().get(SmsReceiver.*NOTIF\_TITLE*);  
 String contentMessage = (String) intent.getExtras().get(SmsReceiver.*NOTIF\_MESSAGE*);  
  
 Contact contact = db.getContactByName(contentTitle);  
  
 PendingIntent pIntent = PendingIntent.*getActivity*(this, 0, intent, 0);  
 NotificationCompat.Builder n = new NotificationCompat.Builder(this)  
 .setContentTitle(contact != null ? contact.getName() : contentTitle)  
 .setContentText(contentMessage)  
 .setSmallIcon(R.drawable.*ic\_message\_white\_24dp*)  
 .setLargeIcon(contact.getImage() != null ? BitmapUtility.*getImage*(contact.getImage()) : null)  
 .setContentIntent(pIntent)  
 .setWhen(System.*currentTimeMillis*())  
 .setAutoCancel(true)  
 .setPriority(NotificationCompat.*PRIORITY\_MAX*)  
 .setOngoing(false);  
  
 Intent smsIntent = new Intent(this, ContactSmsActivity.class);  
 smsIntent.putExtra("name", contact.getName());  
 smsIntent.putExtra("phone", contact.getPhone());  
  
 PendingIntent contentIntent = PendingIntent.*getActivity*(this, 0,  
 smsIntent, PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 n.setContentIntent(contentIntent);  
  
 NotificationManager notificationManager =  
 (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 notificationManager.notify(0, n.build());  
 }  
  
}

package com.example.hivian.my\_contact\_manager.utilities;  
  
public class AccessTokenClass {  
 public String access\_token;  
  
 @Override  
 public String toString() {  
 return "AccessTokenClass{" +  
 "access\_token='" + access\_token + '\'' +  
 '}';  
 }  
}

package com.example.hivian.my\_contact\_manager.views.activities;  
  
import android.content.Intent;  
import android.os.StrictMode;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
import android.widget.ListView;  
  
import com.example.hivian.my\_contact\_manager.R;  
import com.example.hivian.my\_contact\_manager.models.User;  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
  
import java.io.IOException;  
import java.util.Date;  
  
import okhttp3.Call;  
import okhttp3.OkHttpClient;  
import okhttp3.Request;  
import okhttp3.Response;  
  
  
public class InternetContactList extends AppCompatActivity {  
  
 private static final String *BASE\_URL* = "https://stashkevich.azurewebsites.net/";  
 static String *ACCESS\_TOKEN*;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_internet\_contact\_list*);  
 StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();  
 StrictMode.*setThreadPolicy*(policy);  
 final Intent intents = new Intent(this, InternetSendMessageActivity.class);  
  
  
 final Intent intent = getIntent();  
 String aser = null;  
 *ACCESS\_TOKEN* = intent.getStringExtra(LoginActivity.*ACCESS\_TOKEN*);  
  
 try {  
 ListView listView = (ListView) findViewById(R.id.*list\_view\_users*);  
  
 OkHttpClient client = new OkHttpClient();  
  
 final Request request = new Request.Builder()  
 .addHeader("Authorization", "Bearer " + *ACCESS\_TOKEN*)  
 .url(*BASE\_URL* + "api/user/getUsers")  
 .get()  
 .build();  
  
 Call call = client.newCall(request);  
  
 Response response = null;  
 try {  
 response = call.execute();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 GsonBuilder gsonBuilder = new GsonBuilder();  
 Gson g = gsonBuilder.create();  
  
 aser = response.body().string();  
 aser = validate(aser);  
 final User[] users = g.fromJson(aser, User[].class);  
  
 String[] userNames = new String[users.length];  
  
 for (int i = 0; i < users.length; i++)  
 {  
 userNames[i] = users[i].UserName;  
 }  
 ArrayAdapter adapter = new ArrayAdapter<>(this,  
 android.R.layout.*simple\_list\_item\_1*, userNames);  
 listView.setAdapter(adapter);  
  
 listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
 intents.putExtra("name", users[i].UserName);  
 intents.putExtra("id", users[i].id);  
 intents.putExtra("ACCESS\_TOKEN", *ACCESS\_TOKEN*);  
 startActivity(intents);  
 }  
 });  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public String validate(String s) {  
 String b = s.substring(1, s.length() - 1);  
 b = b.replaceAll("\\\\", "");  
 return b;  
 }  
  
 public void GoToMyMessage(View view) {  
 Intent intent = new Intent(this, InternetMyMessageActivity.class);  
 intent.putExtra("ACCESS\_TOKEN", *ACCESS\_TOKEN*);  
 startActivity(intent);  
 }  
}