МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет науки и технологий

имени академика М.Ф. Решетнева»

Направление подготовки

09.04.01 Информатика и вычислительная техника

**КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ**

**«РАЗРАБОТКА ПРИЛОЖЕНИЙ БАЗ ДАННЫХ»**

**ТЕМА: РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ОБМЕНА ТЕКСТОВЫМИ СООБЩЕНИЯМИ ГРУППЫ ПОЛЬЗОВАТЕЛЕЙ**

Выполнил:

студент группы МИВ19-01

А.З. Асроров

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Принял:

доцент, кандидат физ.-мат. наук

Н.В. Киреев

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Красноярск 2019

Содержание

[ВВЕДЕНИЕ 3](#_Toc28126195)

[Создание проекта в Android Studio 4](#_Toc28126196)

[Добавления аутентификации пользователя 7](#_Toc28126197)

[Обработка входа пользователя 8](#_Toc28126198)

[Обработка выхода пользователя 11](#_Toc28126199)

[СОЗДАНИЕ МОДЕЛИ 13](#_Toc28126200)

[ОТПРАВКА СООБЩЕНИЙ В ЧАТ 14](#_Toc28126201)

[ОТОБРАЖЕНИЕ СООБЩЕНИЙ ЧАТА 15](#_Toc28126202)

[REALTIME DATABASE 17](#_Toc28126203)

[ЗАКЛЮЧЕНИЕ 19](#_Toc28126204)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 20](#_Toc28126205)

# ВВЕДЕНИЕ

В современном мире смартфоны стали частью жизни любого человека. Смартфоны пришли на смену мобильным телефонам и принесли с собой огромный выбор функционала и возможностей как для связи между людьми, так и для частичной замены ноутбуков, компьютеров и планшетов. В данный момент уже нельзя представить современного человека без наличия смартфона или планшета.

В ходе изучения рынка смартфонов, планшетов и ноутбуков, стало известно, что большинство устройств работает на базе ОС Android, тем самым предоставляя огромные возможности разработчикам для реализации своих проектов.

С чем связано такое массовое распространение Android устройств? Широкое распространение ОС получила благодаря компании Google и сторонним производителям мобильных устройств. У мобильной системы очень высокая доступность средств разработки и низкая стоимость поддерживаемых устройств. Если же сравнивать с компанией Apple, то цена на их устройства значительна выше и располагает более значительными расходами, как на реализацию, так и на приобретения требуемых устройств.

В связи с высокой доступностью и низкой стоимостью на рынке формируется огромное количество различных Android приложений, который имеют свои уникальные функции и возможности.

Целью данной курсовой работы является создание мобильного приложения для обмена сообщениями в реальном времени между группой людей.

# Создание проекта в Android Studio

Запустим Android Studio и создадим новый проект с пустой Activity (рис. 1).

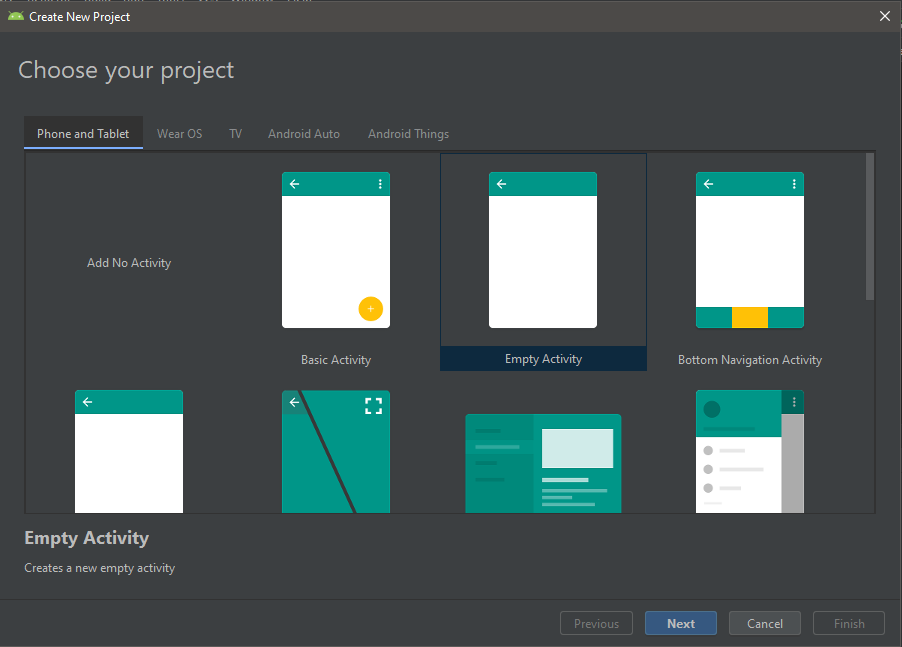


Рисунок 1. Создание нового проекта

Далее зададим макеты (layouts). При создании проекта к MainActivity уже привязан файл activity\_main.xml, который определяет содержимое главного экрана приложения. В нашем случае, он будет представлять чат-комнату. Наше приложение будет содержать следующие элементы интерфейса:

* Список, который будет отображать все сообщения группового чата в хронологическом порядке;
* Поле ввода, в котором пользователь группового чата может ввести новое сообщение;
* Кнопка, при нажатии которой будет происходить отправка сообщения.

После добавления вышеперечисленных элементов содержимое файла activity\_main.xml будет выглядеть следующим образом:

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"  
 android:id="@+id/activity\_main">  
  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/btnSend"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:clickable="true"  
 app:srcCompat="@android:drawable/ic\_input\_add"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentEnd="true"  
 android:layout\_margin="20sp"  
 app:fabSize="mini"/>  
  
 <com.google.android.material.textfield.TextInputLayout  
  
 android:id="@+id/text\_layout"  
 android:layout\_alignParentBottom="true"  
 android:layout\_toLeftOf="@id/btnSend"  
 android:layout\_margin="20sp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/MessageField"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Введите сообщение" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <ListView  
 android:id="@+id/List\_of\_messages"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentTop="true"  
 android:layout\_margin="20sp"  
 android:divider="@android:color/transparent"  
 android:dividerHeight="10dp"  
 android:layout\_above="@id/text\_layout"/>  
  
  
</RelativeLayout>

На рисунке 2 представлен графический макет основного окна мобильного приложения.

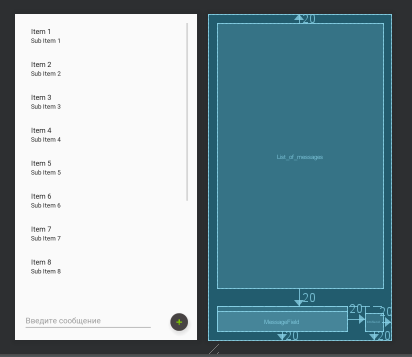


Рисунок 2. Макет основного окна мобильного приложения

После создания макета исходного экрана можно приступать к созданию макета для сообщений чата, которые будут элементами внутри нашего списка. Для этого создадим новый XML файл под названием list\_item.xml. Далее разместим виджеты TextView для отображения текста сообщения, времени его отправки и его автора. После добавления вышеперечисленных элементов содержимое файла activity\_main.xml будет выглядеть следующим образом:

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_alignParentStart="true"  
 android:id="@+id/message\_user"  
 android:textStyle="normal|bold"/>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentEnd="true"  
 android:id="@+id/message\_time"/>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_marginTop="5dp"  
 android:layout\_below="@id/message\_user"  
 android:textSize="18sp"  
 android:id="@+id/message\_text"/>  
  
  
</RelativeLayout>

На рисунке 3 показано выбранное расположение элементов.

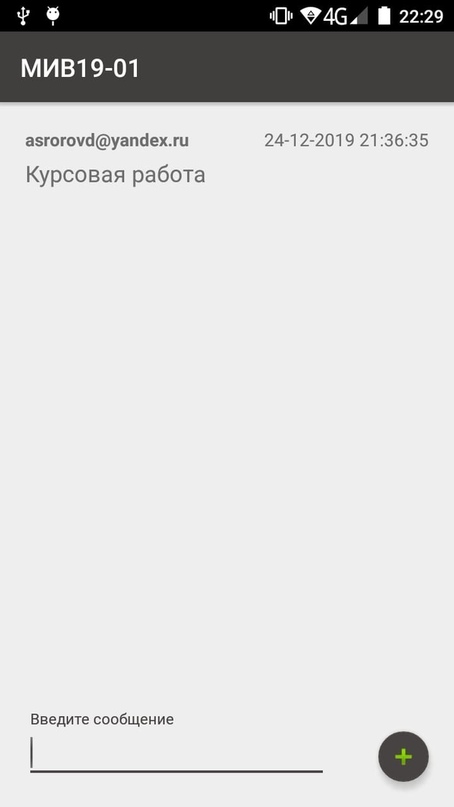


Рисунок 3. Расположение элементов сообщения внутри списка

# Добавления аутентификации пользователя

С целью решения проблем, связанных с безопасностью, настроим наше приложение так, чтобы читать и публиковать сообщения могли только зарегистрированные пользователи. Для этого перейдем в Firebase Console и включим возможность регистрации по адресу электронной почты и паролю (рис. 4).

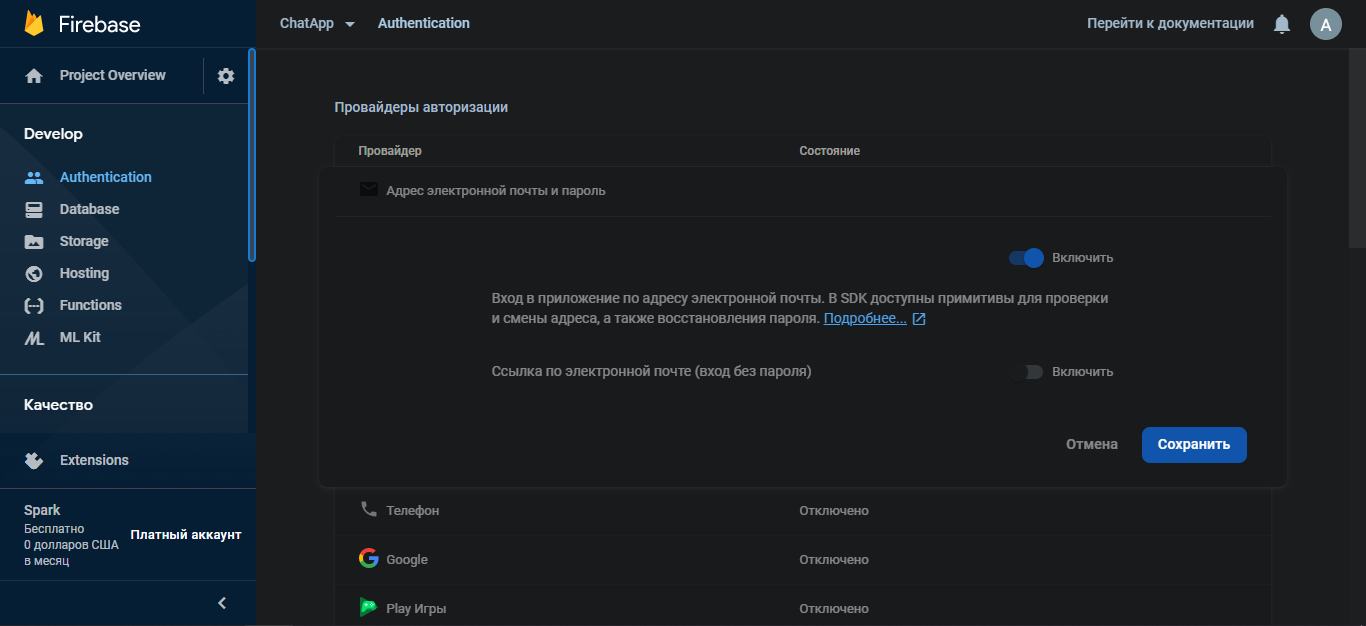


Рисунок 4. Настройка способа входа пользователей

## Обработка входа пользователя

Как только наше мобильное приложение запускается, необходимо сразу проверять, вошел ли пользователь в систему. Если да, то отобразить содержимое чат – комнаты. Иначе, оно должно перенаправить пользователя на экран входа или регистрации. Для решения этой проблемы будем использовать библиотеку FirebaseUI.

FirebaseUI – это библиотека с открытым исходным кодом, которая обеспечивает легкие привязки интерфейса для Firebase Auth и Farebase Database. Добавим зависимости в файл build.grable нашего приложения:

implementation 'com.firebaseui:firebase-ui:0.6.2'

implementation 'com.firebaseui:firebase-ui-auth:0.6.2'

implementation 'com.google.firebase:firebase-auth:19.2.0'

implementation 'com.google.firebase:firebase-database:19.2.0'

Далее внутри метода onCreate() в MainActivity проверим, был ли пользователь зарегистрирован, проверив, не является ли текущий объект FirebaseUser null. Если оно равно null, необходимо создать и настроить объект Intent, который открывает Activity входа в приложение. Для этого используем класс SignInIntentBuilder. После этого, запустим Activity входа, используя метод startActivityForResult(). Ниже представлен код для метода onCreate():

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 activity\_main = findViewById(R.id.*activity\_main*);  
  
 sendBtn = findViewById(R.id.*btnSend*);  
 sendBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 EditText textField = findViewById(R.id.*MessageField*);  
 if (textField.getText().toString() == "")  
 return;  
  
 FirebaseDatabase.*getInstance*().getReference().push().setValue(  
 new Message(  
 FirebaseAuth.*getInstance*().getCurrentUser().getEmail(),  
 textField.getText().toString()  
 )  
 );  
 textField.setText("");  
 }  
 });  
  
 if (FirebaseAuth.*getInstance*().getCurrentUser() == null)  
 startActivityForResult(AuthUI.*getInstance*().createSignInIntentBuilder().build(), *SIGN\_IN\_CODE*);  
 else {  
 Snackbar.*make*(activity\_main, "Вы авторизованы", Snackbar.*LENGTH\_LONG*).show();  
 displayAllMessages();  
 }  
 displayAllMessages();  
  
}

В приведенном выше коде, если пользователь уже вошел в систему, мы сначала отображаем SnackBar приветствуя пользователя, а затем вызываем метод displayAllMessages(), код которого опишем позже.

Как только пользователь выполнит вход, MainActivity получит результат от Intent. Чтобы обработать его, необходимо переопределить метод onActivityResult(). Если код результата – RESULT\_OK, это означает, что пользователь выполнил вход успешно и нам необходимо вызвать метод displayAllMessages(). В противном случае вызовем finish(), чтобы закрыть приложение. Код для метода onActivityResult() представлен ниже:

@Override  
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == *SIGN\_IN\_CODE*) {  
 if (resultCode == *RESULT\_OK*) {  
 Snackbar.*make*(activity\_main, "Вы авторизованы", Snackbar.*LENGTH\_LONG*).show();  
 displayAllMessages();  
 } else {  
 Snackbar.*make*(activity\_main, "Вы не авторизованы", Snackbar.*LENGTH\_LONG*).show();  
 finish();  
 }  
 }

}

На рисунке 5 представлено окно входа в приложение.

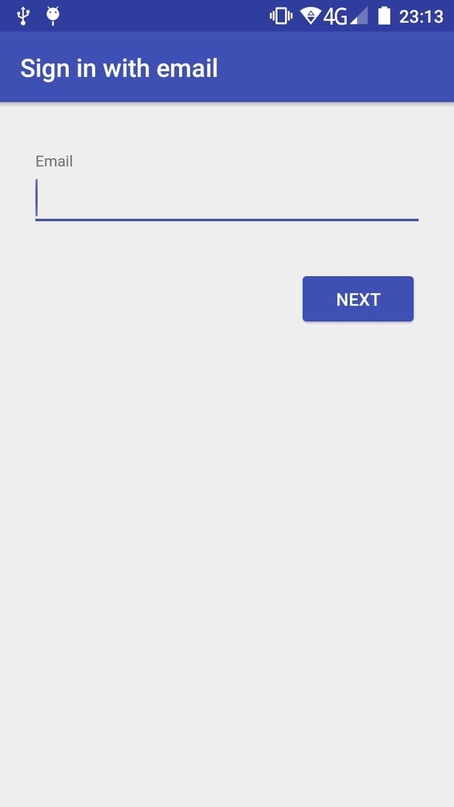


Рисунок 5. Окно входа в приложение

## Обработка выхода пользователя

По умолчанию, FirebaseUI использует Smart Lock для паролей. Поэтому, как только пользователи входят в систему, они останутся в системе, даже если приложение будет перезапущено. Чтобы пользователи могли выйти из системы, мы добавим параметр «Выход» в меню MainActivity.

Создайте новый файл меню ресурсов с именем main\_menu.xml и добавьте к нему один элемент item, название title которого — Выход. Содержимое файла должно выглядеть так:

<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_height="45dp"  
 android:layout\_width="45dp">  
  
 <item android:title="Sign out" app:showAsAction="never"  
 android:id="@+id/menu\_sign\_out"/>  
  
</menu>

Чтобы создать экземпляр ресурса меню внутри MainActivity, переопределим метод onCreateOptionsMenu() и вызовите метод inflate() объекта MenuInflater.

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.layout.*main\_menu*, menu);  
 return true;  
}

Затем переопределите метод onOptionsItemSelected() для обработки событий клика по пункту меню. Внутри этого метода мы вызовем метод signOut() класса AuthUI для выхода пользователя. Поскольку операция выхода из системы выполняется асинхронно, мы также добавим туда OnCompleteListener.

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
 if(item.getItemId() == R.id.*menu\_sign\_out*) {  
 AuthUI.*getInstance*().signOut(this)  
 .addOnCompleteListener(new OnCompleteListener<Void>() {  
 @Override  
 public void onComplete(@NonNull Task<Void> task) {  
 Toast.*makeText*(MainActivity.this,  
 "You have been signed out.",  
 Toast.*LENGTH\_LONG*)  
 .show();  
 // Close activity  
 finish();  
 }  
 });  
 }  
 return true;  
}

Когда пользователь выйдет из системы, приложение должно автоматически закрыться. Для этого мы вызываем метод finish() в коде выше.

# СОЗДАНИЕ МОДЕЛИ

Чтобы сохранять сообщения чата в базе данных Firebase в реальном времени, мы должны создать для них модель. Макет сообщения чата, который мы создали ранее в этом уроке, имеет три представления. Чтобы иметь возможность заполнить эти представления, модель также должна иметь как минимум три поля.

Создадим новый Java класс под названием Message.java и добавьте к нему три переменные: MessageText, messageUser и messageTime. Также добавьте конструктор для инициализации этих переменных.

Чтобы сделать модель совместимой с FirebaseUI, также нужно добавить конструктор по умолчанию, а также геттеры и сеттеры для всей группы переменных.

На этом этапе класс Message будет выглядеть следующим образом:

public class Message {  
 public String userName;  
 public String textMessage;  
 private long messageTime;  
  
 public Message() {}  
 public Message(String userName, String textMessage) {  
 this.userName = userName;  
 this.textMessage = textMessage;  
 this.messageTime = new Date().getTime();  
 }  
  
 public String getUserName() {  
 return userName;  
 }  
  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
  
 public String getTextMessage() {  
 return textMessage;  
 }  
  
 public void setTextMessage(String textMessage) {  
 this.textMessage = textMessage;  
 }  
  
 public long getMessageTime() {  
 return messageTime;  
 }  
  
 public void setMessageTime(long messageTime) {  
 this.messageTime = messageTime;  
 }  
}

# ОТПРАВКА СООБЩЕНИЙ В ЧАТ

Теперь, когда модель готова, мы можем легко добавлять новые сообщения чата в базу данных Firebase в реальном времени.

Чтобы отправить новое сообщение, пользователь будет нажимать FloatingActionButton. Для этого мы должны добавить к ней OnClickListener.

Внутри слушателя мы должны сначала получить объект DatabaseReference, используя метод getReference() класса FirebaseDatabase. Затем вызвать методы push() и setValue(), чтобы добавить новые экземпляры класса Message в базу данных реального времени.

Экземпляры Message должны, конечно, быть инициализированы с использованием содержимого EditText и отображать имя текущего пользователя.

Соответственно, метод onCreate() будет выглядеть следующим образом:

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 activity\_main = findViewById(R.id.*activity\_main*);  
  
 sendBtn = findViewById(R.id.*btnSend*);  
 sendBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 EditText textField = findViewById(R.id.*MessageField*);  
 if (textField.getText().toString() == "")  
 return;  
  
 FirebaseDatabase.*getInstance*().getReference().push().setValue(  
 new Message(  
 FirebaseAuth.*getInstance*().getCurrentUser().getEmail(),  
 textField.getText().toString()  
 )  
 );  
 textField.setText("");  
 }  
 });  
  
 if (FirebaseAuth.*getInstance*().getCurrentUser() == null)  
 startActivityForResult(AuthUI.*getInstance*().createSignInIntentBuilder().build(), *SIGN\_IN\_CODE*);  
 else {  
 Snackbar.*make*(activity\_main, "Вы авторизованы", Snackbar.*LENGTH\_LONG*).show();  
 displayAllMessages();  
 }  
 displayAllMessages();  
  
}

# ОТОБРАЖЕНИЕ СООБЩЕНИЙ ЧАТА

FirebaseUI имеет очень удобный класс под названием FirebaseListAdapter, что значительно снижает попытки, необходимые для заполнения нашего списка, используя данные, имеющиеся в базе данных Firebase. Сейчас мы будем использовать его для извлечения и отображения всех объектов Message, которые присутствуют в базе данных.

Добавим объект FirebaseListAdapter в качестве новой переменной класса MainActivity.

private FirebaseListAdapter<Message> adapter;

Внутри метода displayAllMessages() инициализируем адаптер, используя его конструктор, который ожидает следующие аргументы:

* Ссылка на Activity
* class объекта
* Расположение элементов списка
* Объект DatabaseReference

FirebaseListAdapter является абстрактным классом и имеет абстрактный метод populateView(), который должен быть переопределен.

Как следует из его названия, populateView() используется для заполнения представлений каждого элемента списка.

Внутри метода, мы должны сначала использовать findViewById(), чтобы получить ссылки на каждый TextView, который присутствует в файле макета message.xml. Затем вызовем их методы setText() и заполним их с помощью геттеров класса Message.

На этом этапе содержимое displayAllMessages() должно выглядеть следующим образом:

//Отображение сообщений  
private void displayAllMessages() {  
 ListView listofmessages = findViewById(R.id.*List\_of\_messages*);  
 adapter = new FirebaseListAdapter<Message>(this, Message.class, R.layout.*list\_item*, FirebaseDatabase.*getInstance*().getReference()) {  
 @Override  
 protected void populateView(View v, Message model, int position) {  
 TextView mess\_user, mess\_time, mess\_text;  
 mess\_user = v.findViewById(R.id.*message\_user*);  
 mess\_time = v.findViewById(R.id.*message\_time*);  
 mess\_text = v.findViewById(R.id.*message\_text*);  
  
 mess\_user.setText(model.getUserName());  
 mess\_text.setText(model.getTextMessage());  
 mess\_time.setText(DateFormat.*format*("dd-MM-yyyy HH:mm:ss", model.getMessageTime()));  
 }  
 };  
  
 listofmessages.setAdapter(adapter);  
}

На этом этапе разработка мобильного приложения группового чата готово. Запустим его и отправим тестовое сообщение (рис. 6).

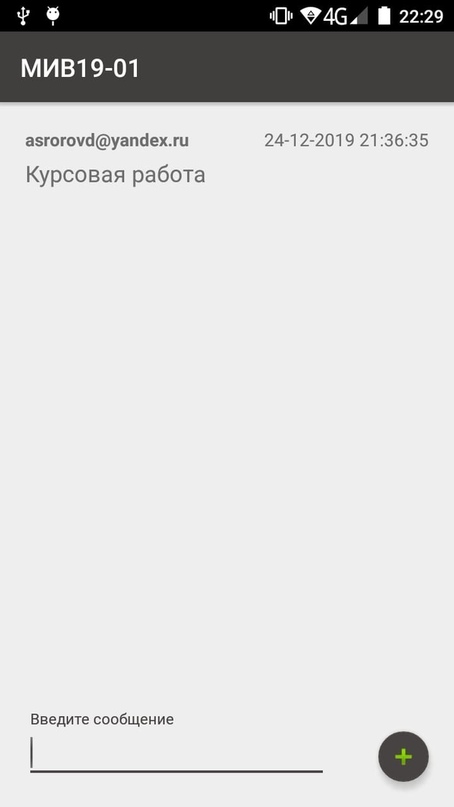


Рисунок 6. Главное окно приложения

# REALTIME DATABASE

На рисунке 7 представлена структура базы данных.

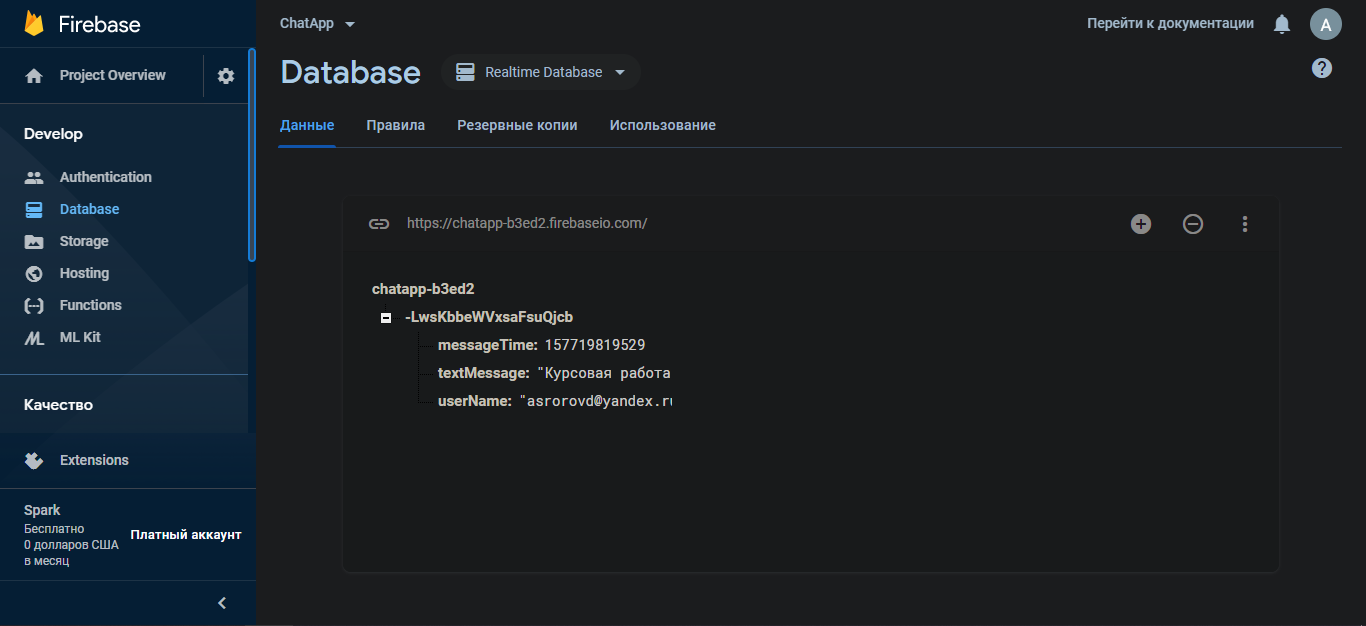


Рисунок 7. Структура базы данных

Правила для базы данных выглядят следующим образом:

{

"rules": {

".read": true,

".write": true

}

}

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы было разработано мобильное приложение группового чата в реальном времени. Был создан адаптивный дизайн для портретной и ландшафтной ориентации экранов. Изучены компоненты Firebase. Использовалась система контроля версий GitHub.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Голощапов А.Л. Google Android: программирование для мобильных устройств / А.Л Голощапов. Санкт-Петербург: БХВ Петербург, 2010. - 448 с. (Дата обращения 10.12.2019)
2. Гид пользователя Android Studio [Электронный ресурс] – Режим доступа URL: <https://developer.android.com/studio/intro/>
3. Documentation Android Developers [Электронный ресурс] – Режим доступа URL: <https://developer.android.com/guide/components/fundamentals>
4. Documentation Firebase [Электронный ресурс] –Режим доступа URL: <https://firebase.google.com/docs>
5. How to Create an Android Chat App Using Firebase [Электронный ресурс] – Режим доступа URL: <https://code.tutsplus.com/tutorials/how-to-create-an-android-chat-app-using-firebase--cms-27397>