SKAMPER

Komunikator audio-wideo na urządzenia mobilne z system Android

Spis Treści

1. Charakterystyka tematu i zespołu realizującego

Projekt zsotał wykonany w składzie 3-osobowym w postaci:

- Wojciech Kwaśny (koordynator składu)
- Sebastian Koba
- Adam Bielawny

Prace na projektem zostały podzielone w następujący sposób:

Wojciech Kwaśny odpowiadał za interfejs graficzny, ustanowienie połączenia audio oraz video, utworzenie czatu oraz przysłanie pakietu w rozmowach każdego typu.

Sebastian Koba był odpowiedzialny za ustanowienie połączenia, archiwum wiadomości, ustanowienie połączenia, utworzenie czatu oraz przesyłanie pakietów w rozmowach każdego typu.

Adam Bielawny odpowiadał za interfejs graficzny, obsługę reklam w technologii RTB oraz prezentacje wraz z dokumentacją ogólną.

Za testowanie aplikacji oraz utworzenie dokumentacji był odpowiedzialny cały zespół.

Prace nad projektem były przeprowadzane na bieżąco bez wiążących terminów. Jako zespół komunikowaliśmy się za pomocą Messengera na portalu społecznościowym Facebook. Wykorzystywaliśmy również repozytorium Githuba do przechowywania najnowszej wersji projektu oraz do szybkiego udostępniania aktualizacji. Głównymi założeniami pracy zespołowej było pracowanie nad projektem na bieżąco, tak zwana metoda małych kroków.

Celem powstania projektu było utworzenie lekkiego oprogramowania z minimalistycznym podejściem do interfejsu i funkcji. Skamper znajduje zastosowanie gdy nie chcemy płacić za połączenie oraz nasze łącze internetowe posiada ograniczoną przepustowość. Nasza aplikacja była tworzona tak aby w jak najmniejszym stopniu wykorzystywała dostępne zasoby.

Po wstępnych ustaleniach utworzyliśmy harmonogram, który pozwala podzielić prace nad projektem na poszczególne etapy.

I Etap

- Podział obowiązków

- Utworzenie repozytorium

- Ustanowienie wstępnych założeń i koncepcji projektu

II Etap

- Utworzenie wstępnego interfejsu graficznego

- Ustanowienie połączenia między urządzeniami

III Etap

- Utworzenie chatu oraz archiwum wiadomości

IV Etap

- Utworzenie komunikacji głosowej

- Wstępne wykorzystanie technologii umieszczania reklam i rozmieszczenie reklam w aplikacji

V Etap

- Utworzenie wideokomunikacji

VI Etap

- Stworzenie panelu użytkownika oraz wykończenie interfejsu graficznego aplikacji

VII Etap

- Wykańczanie prac dotyczących obsługi reklam

- Testowanie aplikacji

- Ostateczne modyfikacje projektu

2. Użytkownicy

W projekcie komunikatora kierując się oszczędnością aplikacji została została zaimplementowana jedna grupa użytkowników z jednakowym poziomem dostępu do aplikacji .

Ilość użytkowników naszej aplikacji pozostaje otwarta. Nie zostały wprowadzone żadne ograniczenia, jednak nie nastawiamy się na specjalnie wysoką popularność naszego projektu.

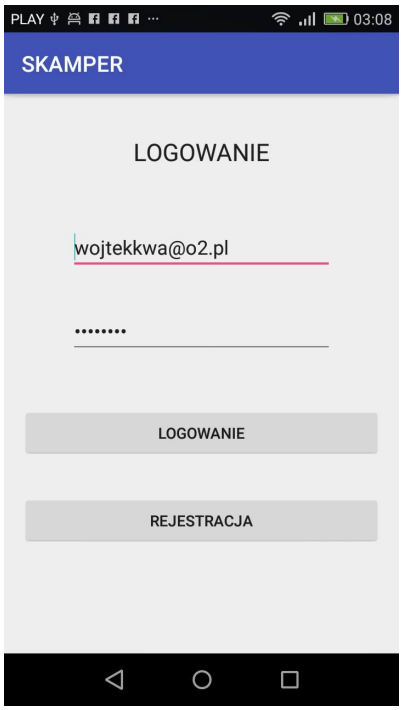
Interfejs w aplikacji klienckiej charakteryzuje się prostą dlatego aplikacja jest skierowana dla każdego kto potrzebuje komunikatora wykorzystującego połączenie internetowe. Do obsługi nie są wymagane żadne specjalne umiejętności czy też kwalifikacje.

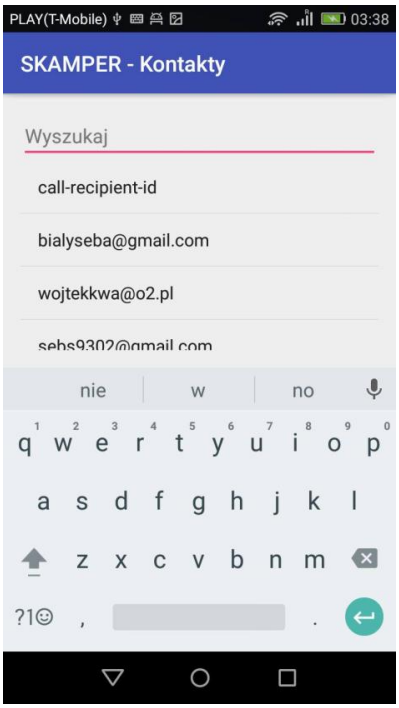
Na obecną chwilę nie przewidujemy kontaktu z odbiorcami naszej aplikacji.

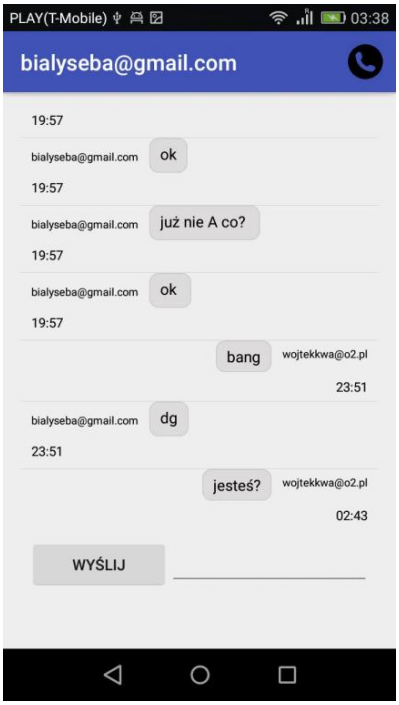
Ze względu na brak podziału na grupy odbiorców, aplikacja zostaje oddana bez specjalnych udogodnień dla jednostek.

3. Ergonomia

Aplikacja ze względu na fakt, iż jest aplikacją mobilną musi mieć czytelny interfejs, a jednocześnie zawierać wszystkie założone funkcjonalności.







Jak widać na powyższych zrzutach ekranu postawiliśmy na przejrzystość widoku. Pierwszym ekranem wyświetlanym po uruchomieniu aplikacji jest panel logowania i rejestracji. Sam interfejs, naszym zdaniem, jest intuicyjny. Przyciski są opisane drukowanymi literami w celu lepszej widoczności.

Po zalogowaniu otrzymujemy ekran listy znajomych, aby usprawnić korzystanie z komunikatora. Dodatkową funkcją jest wyszukiwarka kontaktów z modułem autouzupełnienia, który po wpisaniu paru liter pokaże kontakty zaczynające się ów frazą.

Okno chatu przypomina większość istniejących komunikatorów wiadomości tekstowych w czasie rzeczywistym. Zdecydowaliśmy, że bezcelowym byłoby wymyślanie nowej konstrukcji gdy obecny model jest wszystkim znany i ergonomiczny. Obok wiadomości tekstowych znajdują się informacje o godzinie przesłania danej wiadomości oraz o nadawcy.

Aby rozpocząc rozmowę należy wybrać ikonę słuchawki w prawym górnym rogu, która również obecnie stała się symbolem rozmowy audio. Przy odbieraniu zaproszenia do rozmowy zostają wyświetlone 2 ikony słuchawek w kolorach zielony i czerwony. Kolejno pierwsza z nich oznacza akceptację i rozpoczęcie rozmowy, druga odrzuca połączenie.

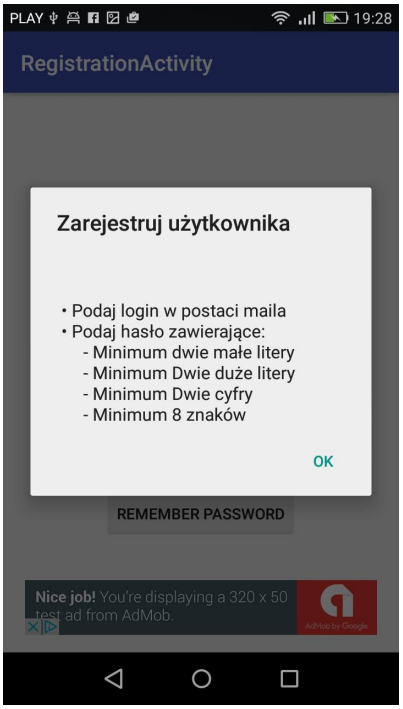
Ponieważ nasz projekt jest aplikacją mobilną, dane wprowadzane są poprzez klawiaturę smartfona.

Z powodu prostego interfejsu aplikacji uznaliśmy, iż funkcja edycji ów interfejsu byłaby niepotrzebnym udziwnieniem i obciążeniem dla samej aplikacji.

Moduł pomocy jest zawarty jedynie w tej dokumentacji w postaci intrukcji obsługi. Aplikacja ze względu na swój minimalistyczny charakter względem zużycia dostępnych zasobów smartfona nie posiada wbudowanej pomocy w oprogramowaniu, jednak jego oszczędność oznacza również prosty i czytelny interfejs, który nie powinien sprawiać problemów nawet początkującym użytkownikom.

Cała aplikacja posiada tło w kolorze (238,238,238) w standardzie RGB, ze względu na fakt, iż biały kolor emitowany przez matryce jest męczący dla ludzkiego oka. Kolor ten jest możliwie jasny w celu kontrastu z czarnymi napisami.

W punktach, w których aplikacja posiada wymagania względem wprowadzanych danych użytkownik jest informowany za pomocą komunikatów pokazujących się na ekranie, jak jest to pokazane na poniższym zrzucie ekranu.



W przypadku niepoprawnie wprowadzanych danych aplikacja poinformuje użytkownika o popełnionym błędzie, jak na poniższym zrzucie ekranu.

4. Założenia wstępne

Do wykonania projektu został wykorzystany język Java, jako środowisko wykorzystaliśmy Android Studio. Do uruchomienia aplikacji wymagane jest urządzenie z systemem zgodnym z Android 4.0.3 lub nowszym, co na obecną chwilę obejmuje 97 % ze wszystkich urządzeń z systemem Android.

Wykorzystywanymi bibliotekami są biblioteki SINCH, które umożliwiają komunikację pomiędzy urządzeniami mobilnymi. Dodatkowo wykorzystujemy biblioteki FireBase, umożliwiające podłączenie bazy danych platformy FireBase z aplikacją.

W naszym projekcie korzystamy z technologii bazy danych w chmurze – FireBase. Rozwiązanie to spełnia nasze wymagania nie obciąża urządzenia i pozwala na szybki dostęp do swojego konta.

W bazie danych przechowujemy dane niebędne do logowania oraz informacje o koncie użytkownika.

Do poprawnego działania aplikacji jak już wcześniej zaznaczyliśmy jest dostęp do internetu.

Aplikacja sama w sobie nie posiada wymagań co do rozmiarów ekranu smartphone'a, jednak reklamy w technologii RTB mają odgórnie narzucone rozmiary banerów dlatego, dlatego minimalną szerokością ekranu umożliwiającą wyświetlenie ekranu jest 320dp (Density-independent Pixels). Nie jest to jednak problemem ze względu na to, iż jest to wartość osiągalna przez praktycznie wszystkie modele obsługujące Android 4.0.3.

Nasza aplikacja jest udostępniana poprzez plik APK, czyli plik instalacyjny systemu Android.

W celu otrzymania ów pliku należy zgłosić się do jednego z twórców ów aplikacji, bądź osoby znajomej posiadającej ten plik. Naszym zamiarem było aby jedną z metod dystrybucji naszej aplikacji stała się tzw. poczta pantoflowa.

Co do praw autorskich kopiowanie oraz pobieranie aplikacji SKAMPER jest całkowicie legalne. Jako twórcy udostępniamy aplikację za darmo, bez żadnych zastrzeżeń ani licencji.

Aplikacja nie posiada złożonych algorytmów kryptograficznych, umożliwiających bezpieczne przechowywanie badź przesyłanie danych. Jedynym zabiezpieczeniem jest zapisywanie haseł do kont za pomocą funkcji skrótu, oraz wewnętrzne zabezpieczenia platformy FireBase.

Z wyżej wymienionych powodów odradzamy przesyłanie oraz przechowywanie wrażliwych informacji.

Ze względu na akademicki charakter projektu, prace nad aktualizacjami jak i system ich przeprowadzania zostały zawieszone.

5. Dokumentacja techniczna

Diagram przypadków użycia

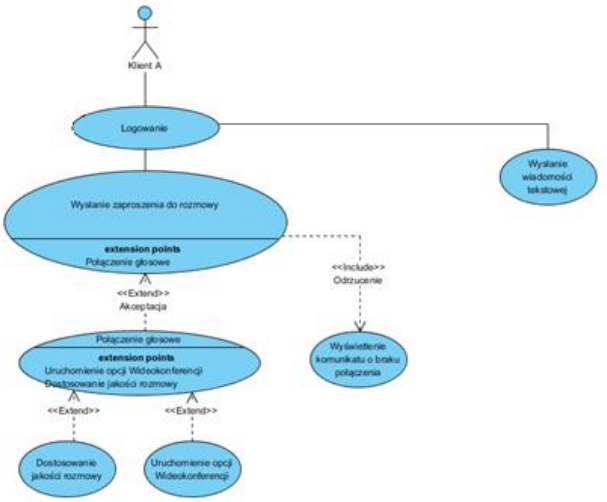
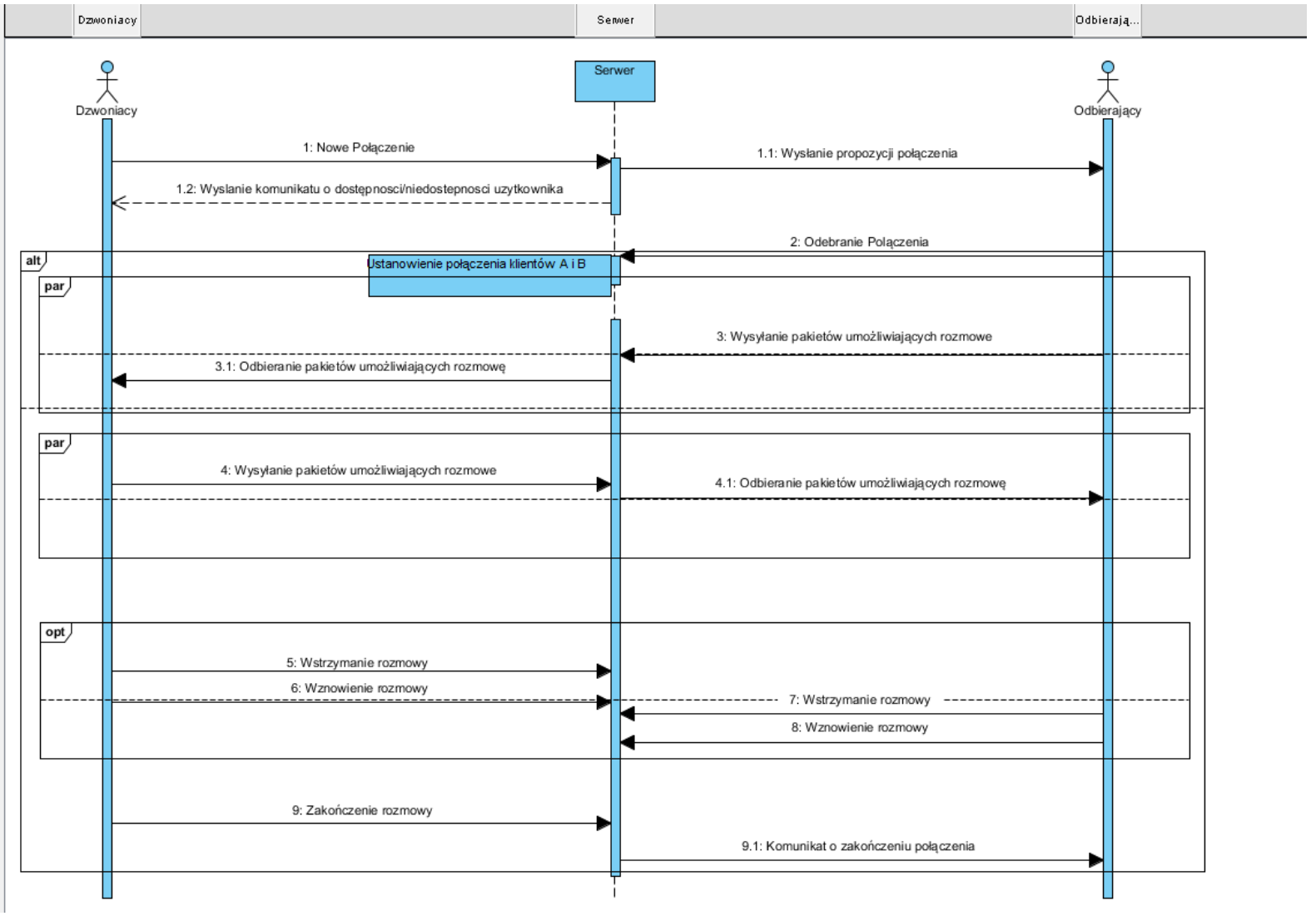


Diagram sekwnecji sesji połączenia głosowego pomiędzy dwoma urządzeniami.



XML Activity głównego aplikacji – activity logowania

```
<?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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.wojtek_asus.app.MainActivity"
    tools:showIn="@layout/activity_main">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Rejestracja"
        android:id="@+id/btRejestracja"
        android:layout_below="@+id/btLogowanie"
        android:layout_marginTop="32dp"
        android:layout_alignRight="@+id/btLogowanie"
        android:layout_alignEnd="@+id/btLogowanie"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:onClick="Register"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Logowanie"
        android:id="@+id/btLogowanie"
        android:layout_below="@+id/InputHaslo"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="46dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:onClick="Firsts"/>

    <EditText
        android:layout_width="240dp"
        android:layout_height="wrap_content"
        android:id="@+id/InputLogin"
        android:layout_below="@+id/LogowanieLabel"
        android:layout_centerHorizontal="true"
        android:hint="login"
        android:text="wojtekkwa@o2.pl"
        android:layout_marginTop="50dp"
        />

    <EditText
        android:layout_width="240dp"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:hint="password"
        android:ems="10"
        android:id="@+id/InputHaslo"
        android:layout_below="@+id/InputLogin"
        android:layout_alignLeft="@+id/InputLogin"
```

```
        android:layout_alignStart="@+id/InputLogin"
        android:layout_marginTop="31dp"
        android:text="KKaa1111" />

    TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Login"
        android:id="@+id/LoginLabel"
        android:layout_alignTop="@+id/InputLogin"
        android:layout_toLeftOf="@+id/InputLogin" />

    TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Haslo"
        android:id="@+id/HasloLabel"
        android:layout_alignTop="@+id/InputHaslo"
        android:layout_toLeftOf="@+id/InputHaslo"
        android:layout_toStartOf="@+id/InputHaslo" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="LOGOWANIE"
        android:id="@+id/LogowanieLabel"
        android:layout_marginTop="22dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

Oraz metody które wywoływane są w tym activity:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Firebase.setAndroidContext(this);

    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    username_tv = (TextView) findViewById(R.id.InputLogin);
    password_tv = (TextView) findViewById(R.id.InputHaslo);
    mInterstitialAd = new InterstitialAd(this);
    mInterstitialAd.setAdUnitId("ca-app-pub-3940256099942544/1033173712");
    mInterstitialAd.setAdListener(new AdListener() {
        @Override
        public void onAdClosed() {

            requestNewInterstitial();
            Intent intent = new Intent(con, ContactsList.class);
            startActivity(intent);
        }
    });
    requestNewInterstitial();
}
```

Po starcie activity wywoływany oncreate odpowiada za przekazanie polaczenia z baza danych, przypisanie do zmiennych danych pobranych z edittextow jak login i haslo oraz wywolanie reklamy.

```
private void requestNewInterstitial()
{
    AdRequest adRequest = new AdRequest.Builder()
            .addTestDevice(AdRequest.DEVICE_ID_EMULATOR)
            .build();
    mInterstitialAd.loadAd(adRequest);
}
```

Funkcja odpowiadająca za załadowanie reklamy.

```
public void Firsts(View view)
{
    if (mInterstitialAd.isLoaded())
    {

        Login();
        mInterstitialAd.show();

    }
    else
    {
        Login();

    }

}
```

Funkcja odpowiadająca za wyświetlenie reklamy oraz wywołanie funkcji logowania po jej wyłączeniu.

```
public void Login()
{

    User.getInstance().password = password_tv.getText().toString();
        loginClicked();
}
```

funkcja wywolywana przez buton logowania, wywolujaca metode loginclicked().

```
public void Register(View view)
{
    Intent intent = new Intent (this, RegistrationActivity.class);
    startActivity(intent);

}
```

Po wywołaniu klawisza rejestracja zostajemy przeniesieni do activity rejestracji (RegistrationActivity)

```
private void loginClicked() {
    EditText Logintmp = (EditText) findViewById(R.id.InputLogin);
    EditText Passwordtmp = (EditText) findViewById(R.id.InputHaslo);
    String userName = Logintmp.getText().toString();
    if (userName.isEmpty()) {

        AlertDialog.Builder builder12 = new AlertDialog.Builder(con);
        builder12.setMessage("Wprowadź Login").show();

    }
    else
    {

        Firebase ref = new Firebase("https://fiery-torch-
1348.firebaseio.com");
        ref.authWithPassword(Logintmp.getText().toString(),
Passwordtmp.getText().toString(), new Firebase.AuthResultHandler() {
            @Override
            public void onAuthenticated(AuthData authData) {
                AlertDialog.Builder builder112 = new
AlertDialog.Builder(con);
                builder112.setMessage("Zalogowano" + authData.getUid() + "
Haslo" + authData.getProvider());

                User.getInstance().username =
username_tv.getText().toString();

                startService(new Intent(getApplicationContext(),
CallingService.class));

            }

            @Override
            public void onAuthenticationError(FirebaseError firebaseError)
{
                AlertDialog.Builder builder1112 = new
AlertDialog.Builder(con);
                builder1112.setMessage("Nie zalogowano").show();
            }
        });


    }
```

Funkcja odpowiadająca za przekazanie parametrów logowania, sprawdzanie poprawności wprowadzonych danych, gdy zostanie wprowadzony niepoprawny login wyświetli się błąd, jeśli nie zostanie wprowadzony wyświetli się alert z informacja by wprowadzić nazwę loginu. Następuje polaczenie z baza danych pod adresem: https://fiery-torch-1348.firebaseio.com, następuje autentykacja użytkownika, jeśli następuje poprawne potwierdzenie danych użytkownik jest logowany, zostaje wyświetlony komunikat o zalogowaniu danego usera. Następnie zostaje wystartowany serwis dzwoniący(CallingService) wraz z parametrami użytkownika. Kolejno startuje activity listy kontaktów (ContactsList). Jeśli wystąpił jakiś błąd logowania np. brak polaczenia z Internetem lub blednie wprowadzone dane logowania, wyskakuje alert z informacja ze nie zostaliśmy zalogowani.

XML Activity rejestracji:

```
<?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"
    xmlns:ads="http://schemas.android.com/apk/res-auto"

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"

    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.wojtek_asus.app.RegistrationActivity"
    tools:showIn="@layout/activity_registration"
    android:id="@+id/qqqqqqqqqqeqweqweqwewqe">
    <com.google.android.gms.ads.AdView
        android:id="@+id/adView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        ads:adSize="BANNER"
        ads:adUnitId="@string/banner_ad_unit_id">
    </com.google.android.gms.ads.AdView>

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/LoginRejestration"
        android:hint="login"
        android:layout_marginTop="156dp"
        android:layout_alignParentTop="true"
        android:layout_alignLeft="@+id/PasswordRejestration"
        android:layout_alignStart="@+id/PasswordRejestration"
        android:layout_alignRight="@+id/PasswordRejestration"
        android:layout_alignEnd="@+id/PasswordRejestration" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:id="@+id/PasswordRejestration"
        android:hint="hasło"
        android:layout_below="@+id/LoginRejestration"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Zarejestruj"
        android:id="@+id/btRejestration"
        android:layout_below="@+id/PasswordRejestration"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="49dp"
        android:onClick="SignUp"
        />
</RelativeLayout>
```

```
public void SignUp(View view) throws IOException {
    try {


        Firebase myFirebaseRef = new Firebase("https://fiery-torch-
1348.firebaseio.com");
        Firebase.setAndroidContext(getApplicationContext());
        final EditText login = (EditText)
findViewById(R.id.LoginRejestration);
        EditText password = (EditText)
findViewById(R.id.PasswordRejestration);

        myFirebaseRef.createUser(login.getText().toString(),
password.getText().toString(), new Firebase.ValueResultHandler<Map<String,
Object>>() {
            @Override
            public void onSuccess(Map<String, Object> result) {

                AlertDialog.Builder builder119 = new
AlertDialog.Builder(getApplicationContext());
                builder119.setMessage("Dodano Użytkownika" +
login.getText().toString());
                ;
            }

            @Override
            public void onError(FirebaseError firebaseError) {
                AlertDialog.Builder builder119 = new
AlertDialog.Builder(getApplicationContext());
                builder119.setMessage(firebaseError.toString() +
"Użytkownik już istnieje");
            }
        });

        loginClicked();
        passwordClicked();
    } catch (Exception ex) {
        AlertDialog.Builder builder113 = new
AlertDialog.Builder(getApplicationContext());
        builder113.setMessage(ex.getMessage()).show();
    }
    if (sum < 1) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
}

;
```

Główna funkcja wywoływana przez buton **@+id/btRejestration.** Funkcja odpowiada za zarejestrowanie nowego użytkownika. Na początku następuje nawiązanie polaczenia z baza danych fierytorch, następnie pobranie danych do rejestracji z edittextow. W postaci mapy użytkownicy i ich hasła są rejestrowani w bazie danych. Jeśli wszystko przebiegło poprawnie to otrzymujemy komunikat o zarejestrowaniu użytkownika. Jeśli dany użytkownik o takim samym loginie już istnieje dostajemy komunikat ze taki user już istnieje i nie jest rejestrowany drugi raz. Metody loginClicked() oraz passwordClicked() odpowiadają za sprawdzenie wymagań loginu oraz hasła. Na koniec sprawdzamy czy wszystkie wymagania są spełnione. Jeśli tak po rejestracji przechodzimy do activity logowania.

```
protected void onCreate(Bundle savedInstanceState) {


    super.onCreate(savedInstanceState);

    builder11 = new AlertDialog.Builder(
RegistrationActivity.this).create();
    builder11.setTitle("Zarejestruj użytkownika");
    builder11.setMessage(" \n \n • Podaj login w postaci maila \n \u2022
Podaj hasło zawierające: \n\t\t - Minimum dwie małe litery \n" +
            "\t\t - Minimum Dwie duże litery \n \t\t - Minimum Dwie cyfry
\n \t\t - Minimum 8 znaków");

    builder11.setButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            // Write your code here to execute after dialog closed
            //Toast.makeText(getApplicationContext(), "You clicked on OK",
Toast.LENGTH_SHORT).show();
            builder11.cancel();
        }
    });
    builder11.setCancelable(false);
    builder11.show();


    setContentView(R.layout.activity_registration);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    MobileAds.initialize(getApplicationContext(), "ca-app-pub-
3940256099942544~3347511713");
    AdView mAdView = (AdView) findViewById(R.id.adView);
    AdRequest adRequest = new AdRequest.Builder().build();
    mAdView.loadAd(adRequest);
}
```

Po przejściu z activityMain do activity rejestracji oncreate wywołuje nam na początku alert zawierający wymaganie jakie musi spełniać login oraz hasło by użytkownik został zalogowany, po kliknięciu ok przechodzimy do właściwej rejestracji. Wraz z załadowaniem activity na dole zostaje załadowana reklama.

```
public void onBackPressed() {
    super.onBackPressed();
    Intent inte = new Intent(this, MainActivity.class);
    startActivity(inte);
}
```

funkcja odpowiadająca za blokowanie activity by nie można było cofnąć rejestracji do main inaczej niż przyciskiem do tego służącym, ponieważ inaczej przechodzilibyśmy do alertów.

```
private void loginClicked() {
    EditText Logintmp = (EditText) findViewById(R.id.LoginRejestration);
    String userName = Logintmp.getText().toString();
    if (userName.isEmpty()) {
        sum++;
        AlertDialog.Builder builder13 = new AlertDialog.Builder(this);
        builder13.setMessage("Wprowadź Login").show();
    }
}
```

Funkcja sprawdzająca czy login nie jest pusty.

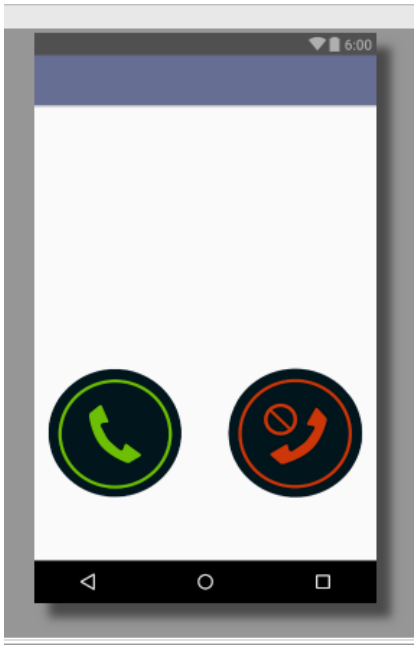
```
private void passwordClicked() {
    EditText Passwordtmp = (EditText)
findViewById(R.id.PasswordRejestration);
    String UserPass = Passwordtmp.getText().toString();
    if (UserPass.length() < 8) {
        sum++;

        AlertDialog.Builder builder14 = new AlertDialog.Builder(this);
        builder14.setMessage("Za krótkie hasło").show();
    }
    int countnum = 0;
    int countupp = 0;
    int countlow = 0;

    for (int i = 0; i < UserPass.length(); i++) {
        if (Character.isDigit(UserPass.charAt(i))) {
            countnum++;
        }
    }

    for (int i = 0; i < UserPass.length(); i++) {
        if (Character.isUpperCase(UserPass.charAt(i))) {
            countupp++;
        }
    }
    for (int i = 0; i < UserPass.length(); i++) {
        if (Character.isLowerCase(UserPass.charAt(i))) {
            countlow++;
        }
    }
    if (countnum < 2) {
        sum++;
        AlertDialog.Builder builder15 = new AlertDialog.Builder(this);
        builder15.setMessage("Hasło powinno zawierać minimum 2
cyfry").show();
    }
    if (countlow < 2) {
        sum++;
        AlertDialog.Builder builder17 = new AlertDialog.Builder(this);
        builder17.setMessage("Hasło powinno zawierać minimum 2 małe
litery").show();
    }
    if (countupp < 2) {
        sum++;
        AlertDialog.Builder builder16 = new AlertDialog.Builder(this);
        builder16.setMessage("Hasło powinno zawierać minimum 2 duże
litery").show();
    }
}
```

Funkcja sprawdzająca wymaganie dotyczące hasła. Sprawdzamy tu miedzy innymi czy podane hasło zawiera 2 małe litery, 2 duże litery, 2 cyfry oraz czy składa się z minimum 8 znaków. Jeśli któreś z wymagań nie jest spełnione wyświetla się alert które dokładnie nie jest oraz dodawana wartość do sum które wykorzystywane jest później do blokowania rejestracji.

```
class TempUser {
    String TmpLogin;
    String TmpPass;

    TempUser(String TmpLogin, String TmpPass) {
        this.TmpLogin = TmpLogin;
        this.TmpPass = TmpPass;
    }
}
```

klasa wewnątrz klasy RejestrationActivity.java, utworzona by przekazywać dane.
Xml RingingActivity:

```
<?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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.wojtek_asus.app.RingingActivity"
    tools:showIn="@layout/activity_ringing">

    <ImageButton
        android:layout_width="150dip"
        android:layout_height="150dip"
        android:id="@+id/AcceptCallBtn"
        android:background="@drawable/accepticon"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:onClick="receivecall"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="107dp" />

    <ImageButton
        android:layout_width="150dip"
        android:layout_height="150dip"
        android:id="@+id/imageButton"
        android:background="@drawable/declineicon"
        android:onClick="abortcall"
        android:layout_alignTop="@+id/AcceptCallBtn"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```




```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_ringing);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    mMediaPlayer = new MediaPlayer();
    mMediaPlayer = MediaPlayer.create(this, R.raw.ring);
    mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
    mMediaPlayer.setLooping(true);
    mMediaPlayer.setVolume(1f, 1f);

    mMediaPlayer.start();

}
```

Po wywołaniu dzwonienia do danego użytkownika, użytkownik do którego dzwonimy i jego listener wykrył próbę nawiązanie polaczenia, otrzymuje informacje o tym w postaci wyświetlającego się activityRinging w którym może zaakceptować lub odrzucić polaczenie. onCreate wywołuje dzwonienie dzwonka i konfiguruje MediaPlayer.

```
public void receivecall(View view){
   User.getInstance().call.answer();
   Intent intent = new Intent(this, CallActivity.class);
   mMediaPlayer.stop();
   startActivity(intent);

}
```

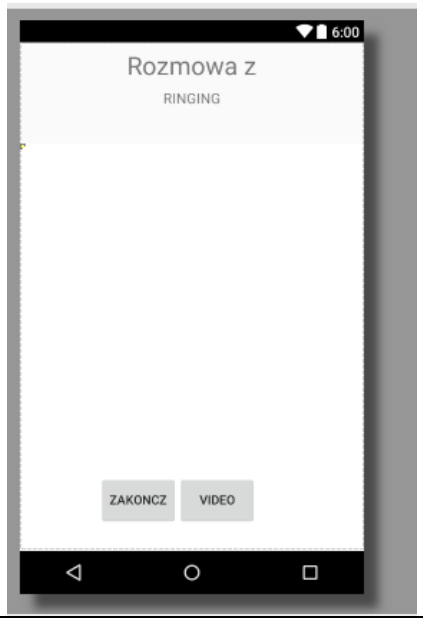
Jest to funkcja wywoływana przez buton **@+id/AcceptCallBtn** i odpowiada za pozytywna odpowiedz na polaczenie, zatrzymanie MediaPlayera, oraz wywołanie activityCall w celu sfinalizowania polaczenia.

```
public void abortcall(View view){
    User.getInstance().call.hangup();
    Intent intent = new Intent(this, ContactsList.class);
    mMediaPlayer.stop();
    startActivity(intent);

}
```

Jest to funkcja wywoływana przez button **@+id/imageButton** i odpowiadająca za zerwanie polaczenia i powrót do activity z lista kontaktów.
Xml callActivity:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="5"
    android:background="#ffffffff"
    >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="vertical"
        android:layout_weight="1"
        android:background="#fffafafa"

        >

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_margin="10dp"
            android:text="Rozmowa z"

            android:textSize="28sp"/>

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="Ringing"

            android:textSize="16sp"
            android:textAllCaps="true"
            />



    </LinearLayout>

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:padding="0dip"
        android:layout_weight="3"
         android:orientation="vertical">
```

```
        <LinearLayout android:id="@+id/remoteVideo"
android:layout_width="wrap_content"
            android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
android:orientation="horizontal"/>

        <RelativeLayout android:id="@+id/localVideo"
android:layout_width="150dp"
            android:layout_alignParentRight="true"
android:layout_height="200dp"
            android:layout_alignParentTop="true" />

    </RelativeLayout>

    <RelativeLayout
        android:id="@+id/bottomPanel"
        android:layout_width="match_parent"
        android:layout_height="0dp"

        android:layout_weight="1">
        <Button
            android:text="Zakoncz"
            android:id="@+id/button2"
            android:onClick="endCall"
            android:paddingBottom="20dp"
            android:paddingTop="20dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="88dp"
            android:layout_marginStart="88dp"
            android:layout_centerVertical="true"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Video"
            android:id="@+id/btvideo"
            android:onClick="VideoCallButtonClicked"
            android:layout_gravity="bottom"
            android:layout_alignTop="@+id/button2"
            android:layout_toRightOf="@+id/button2"
            android:layout_alignBottom="@+id/button2"
            />

    </RelativeLayout>

</LinearLayout>
```




```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_call);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        TextView zkimgadam = (TextView) findViewById(R.id.textView2);
        zkimgadam.setText(User.getInstance().call.getRemoteUserId());
        User.getInstance().call.addCallListener(new SinchCallListener());
if(User.getInstance().call.getDetails().isVideoOffered() == true)
{
    onVideoTrackAdded();

}
    }
```

Funkcja onCreate odpowiada za przechwycenie informacji z kim prowadzimy rozmowę oraz sprawdzaniu czy użytkownik nie wywołuje rozmowy video.

```
public class SinchCallListener implements CallListener {
    @Override
    public void onCallEnded(Call endedCall) {
        //if(User.getInstance().call.getDetails().isVideoOffered() == true)
{
        //recreate();
        //}

        setVolumeControlStream(AudioManager.USE_DEFAULT_STREAM_TYPE);
        User.getInstance().call = endedCall;

        if(videoinviter)
        {
            videoinviter = false;
            //Intent intent = new
Intent(getApplicationContext(),CallActivity.class);
            //audioManager.setSpeakerphoneOn(false);
          // startActivity(intent);
        }
        else {
```

```
            //'finish();
            Intent intent = new Intent(getApplicationContext(),
ContactsList.class);
            //audioManager.setSpeakerphoneOn(false);
            startActivity(intent);
        }
    }
```

Klasa SinchCallListener w której mamy zaimplementowana funkcje kończenia rozmowy. Po jej zakończeniu przechodzimy do activity ContactsList.

```
@Override
public void onCallEstablished(Call establishedCall) {
    //setVolumeControlStream(AudioManager.STREAM_VOICE_CALL);
    // Intent intent = new Intent(getApplicationContext(),
CallActivity.class);
    //startActivity(intent);
    //incoming CallActivity was picked up

    audioManager = (AudioManager)getSystemService(Context.AUDIO_SERVICE);
    audioManager.setMode(AudioManager.MODE_IN_CALL);
    audioManager.setSpeakerphoneOn(false);
}
```

Funkcja wywoływana gdy polaczenie zostało zestawione, ustawia parametry audioManagera.

```
public void endCall(View view) {
        //mAudioPlayer.stopProgressTone();

        if (User.getInstance().call != null) {
            User.getInstance().call.hangup();
                    }
finish();
    }
```

Funkcja kończenia polaczenia.

```
public void onVideoTrackAdded()
    {
vc = User.getInstance().sinchClient.getVideoController();

        LinearLayout view = (LinearLayout) findViewById(R.id.remoteVideo);
        view.addView(vc.getRemoteView());
        RelativeLayout localView = (RelativeLayout)
findViewById(R.id.localVideo);
        localView.addView(vc.getLocalView());
        //View myPreview = vc.getLocalView();
        //View remoteView = vc.getRemoteView();
        User.getInstance().sinchClient.getAudioController();

    }
```

Funkcja wywoływana gdy nawiązujemy polaczenie video, przekazuje strumień video do utworzonego layoutu gdzie wyświetlane jest video.

```
public void VideoCallButtonClicked(View view) {
    User.getInstance().call.hangup();
    videoinviter = true;
   onVideoTrackAdded();

    String userName = User.getInstance().call.getRemoteUserId();
    if (userName.isEmpty()) {
        Toast.makeText(this, "Nie przeczytałem odbiorcy",
Toast.LENGTH_LONG).show();
        return;
    }
    User.getInstance().call =
User.getInstance().sinchClient.getCallClient().callUserVideo(userName);

    User.getInstance().call.getCallId();

}
```

Wywoływana po kliknięciu buttona @+ **id/btvideo** odpowiada za zestawienie polaczenia video z określonym odbiorcą.

Activity ConversationActivity

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_conversation);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    chosenuser = (String) getIntent().getSerializableExtra("Chosen");
    //getIntent().getStringExtra("Chosen");
    getSupportActionBar().setTitle(chosenuser);


    buttonSend = (Button) findViewById(R.id.send_button);
    listView = (ListView) findViewById(R.id.messages_view);

    chatArrayAdapter = new ChatArrayAdapter(getApplicationContext(),
R.layout.right);
    listView.setAdapter(chatArrayAdapter);
    chatText = (EditText) findViewById(R.id.message_input);
    messagesSaver = new MessagesSaver();
    User.getInstance().messages = new ArrayList<ChatMessage>();







    User.getInstance().messages = messagesSaver.readmessages(this);

    for(int i=0;i< User.getInstance().messages.size();i++)
    {
        if(User.getInstance().messages.get(i).left == false &&
User.getInstance().messages.get(i).sender.equals(chosenuser))
        {
```

```
            chatArrayAdapter.add(User.getInstance().messages.get(i));
        }

        if(User.getInstance().messages.get(i).left == true &&
User.getInstance().messages.get(i).recipient.equals(chosenuser))
        {
            chatArrayAdapter.add(User.getInstance().messages.get(i));
        }
    }



    chatText.setOnKeyListener(new View.OnKeyListener() {
        public boolean onKey(View v, int keyCode, KeyEvent event) {
            if ((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode ==
KeyEvent.KEYCODE_ENTER)) {
                try {
                    return sendChatMessage();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            return false;
        }

    });


    buttonSend.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {

            try {
                sendChatMessage();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });

    listView.setTranscriptMode(AbsListView.TRANSCRIPT_MODE_ALWAYS_SCROLL);
    listView.setAdapter(chatArrayAdapter);

    //to scroll the list view to bottom on data change
    chatArrayAdapter.registerDataSetObserver(new DataSetObserver() {
        @Override
        public void onChanged() {
            super.onChanged();
            listView.setSelection(chatArrayAdapter.getCount() - 1);
        }


    });


    receiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String mes = intent.getStringExtra("message");
            String who = intent.getStringExtra("messagewho");
            Calendar calendar = Calendar.getInstance();
            SimpleDateFormat simpleDateFormat = new
```

```
SimpleDateFormat("HH:mm");
            final String hour =
simpleDateFormat.format(calendar.getTime());

            if(who.equals(chosenuser)) {
                chatArrayAdapter.add(new ChatMessage(false, mes, who, hour,
User.getInstance().username));
            }
            // do something here.
        }
    };


}
```

Metoda onCreate ConversationActivity. Na początku następuje utworzenie obiektów w Javie oraz przypisanie do nich elementów designu. Następnie wiadomości czytane są z pliku do listy obiektów ChatMessage po to by mogły zostać wyświetlone za pomocą widoku ListView za pomocą ChatArrayAdapter. Stworzony jest listener, który po naciśnięciu Buttona „Wyślij” wysyła wiadomość i czyści EditText.

```
private boolean sendChatMessage()
{
    Calendar c = Calendar.getInstance();
    SimpleDateFormat df = new SimpleDateFormat("HH:mm");
    String formattedDate = df.format(c.getTime());
    chatArrayAdapter.add(new ChatMessage(true,
chatText.getText().toString(), User.getInstance().username,
formattedDate,chosenuser));
    messagesSaver.savemessage((new ChatMessage(true,
chatText.getText().toString(), User.getInstance().username,
formattedDate,chosenuser)), User.getInstance().messages,
ConversationActivity.this);
    WritableMessage msg = new WritableMessage(chosenuser,
chatText.getText().toString());
    User.getInstance().messageClient.send(msg);
    chatText.setText("");
    return true;
}
```

Metoda odpowiadająca za wysłanie wiadomości do określonego użytkownika.

```
public class ChatMessage implements Serializable {
    public boolean left;
    public String message;
    public String sender;
    public String date;
    public String recipient;

    public ChatMessage(boolean left, String message, String sender, String
date,String recipient) {
```

```
        super();
        this.left = left;
        this.message = message;
        this.sender = sender;
        this.recipient = recipient;
        this.date = date;
    }
}
```

Klasa przechowująca wiadomości oraz informacje o wiadomościach. Implementuje interfejs Serializable aby możliwe było zapisywanie wiadomości w pliku na urządzeniu.

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;


class ChatArrayAdapter extends ArrayAdapter<ChatMessage> {

    private TextView chatText;
    private TextView dateField;
    private TextView userField;
    private List<ChatMessage> chatMessageList = new
ArrayList<ChatMessage>();
    private Context context;


    @Override
    public void add(ChatMessage object) {
        chatMessageList.add(object);
        super.add(object);
    }

    public ChatArrayAdapter(Context context, int textViewResourceId) {
        super(context, textViewResourceId);
        this.context = context;
    }

    public int getCount() {
        return this.chatMessageList.size();
    }

    public ChatMessage getItem(int index) {
        return this.chatMessageList.get(index);
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        ChatMessage chatMessageObj = getItem(position);
        View row = convertView;
        LayoutInflater inflater = (LayoutInflater)
this.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        if (chatMessageObj.left) {
```

```
            row = inflater.inflate(R.layout.right, parent, false);
        }else{
            row = inflater.inflate(R.layout.left, parent, false);
        }
        chatText = (TextView) row.findViewById(R.id.checkpointid);
        chatText.setText(chatMessageObj.message);
        dateField = (TextView) row.findViewById(R.id.date);
        dateField.setText(chatMessageObj.date);
        userField = (TextView) row.findViewById(R.id.duration);
        userField.setText(chatMessageObj.sender);
        return row;
    }
}
```

Adapter służący do wyświetlania listy wiadomości pobieranych z urządzenia dla danego użytkownika(wysyłanych I odbieranych w czasie rzeczywistym również)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.wojtek_asus.app.ConversationActivity"
    tools:showIn="@layout/activity_conversation">

    <ListView
        android:id="@+id/messages_view"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_centerHorizontal="true"
        android:layout_weight="5" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_margin="8dp"
        android:layout_weight="1"
        android:orientation="horizontal">

        <Button

            android:id="@+id/send_button"
            android:layout_width="128dp"
            android:layout_height="wrap_content"
            android:text="Wyślij"

            />


        <EditText
            android:id="@+id/message_input"
```

```
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"


            />
    </LinearLayout>
</LinearLayout>
```

Plik XML odpowiadający za wygląd Activity czatu. Głównym layoutem jest LinearLayout, który zawiera w sobie kolejny LinearLayout do wyświetlania wiadomości oraz Button i EditText do wysyłania oraz wpisywania wiadomości.

Activity ContactsList

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_contactslist);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);


    final ListView myListView = (ListView)
findViewById(R.id.contactslistview);
    ArrayList<String> contacts = new ArrayList<String>();
    contacts.add("call-recipient-id");
    contacts.add("bialyseba@gmail.com");
    contacts.add("wojtekkwa@o2.pl");
    contacts.add("sebs9302@gmail.com");

    LV = (ListView) findViewById(R.id.contactslistview);
    InputSearch = (EditText) findViewById(R.id.InputFilter);
    adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, contacts);
    LV.setAdapter(adapter);
    InputSearch.addTextChangedListener(new TextWatcher() {

        @Override
        public void onTextChanged(CharSequence cs, int arg1, int arg2, int
arg3) {
            // When user changed the Text
            ContactsList.this.adapter.getFilter().filter(cs);

        }

        @Override
        public void beforeTextChanged(CharSequence arg0, int arg1, int
arg2, int arg3) {
        }

        @Override
        public void afterTextChanged(Editable arg0) {
        }
    });
    //
```

```
    // myListView.setAdapter(adapter);


    myListView.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {

            String chosen = (String)
myListView.getAdapter().getItem(position);
            Toast.makeText(getApplicationContext(), chosen,
                    Toast.LENGTH_SHORT).show();
            Intent intent = new
Intent(getApplicationContext(),ConversationActivity.class);
            intent.putExtra("Chosen", chosen);
            startActivity(intent);
        }
    });



    // ATTENTION: This was auto-generated to implement the App Indexing
API.
    // See https://g.co/AppIndexing/AndroidStudio for more information.
    client = new
GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
}
```

Metoda onCreate ContactList Activity. Występuje przypisanie do obiektu ListViev jego widoku a następnie przypisanie elementów listy kontaktów do adaptera wyświetlającego. Zadeklarowany jest również EditText, który posiada listener pozwalający filtrować listę kontaktów. Kolejny listener przypisany do ListView pozwala uruchomić ekran konwersacji z danym użytkownikiem oraz wyświetlić komunikat wyświetlający jakiego użytkownika wybraliśmy.

```
<?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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".ContactsList"
    tools:showIn="@layout/activity_contactslist">

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <EditText
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/InputFilter"

        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:hint="Wyszukaj" />

        <ListView
        android:id="@+id/contactslistview"
        style="@style/ContactsListView"
        android:entries="@array/SKAMPER"
        android:layout_height="437dp" />



</LinearLayout>
</RelativeLayout>
```

Plik XML opisujący wygląd Activity ContactList. Na samej górze struktury jest RealativeLayout, który zawiera LinearLayout w którym znajdują się ListView wyświetlający listę kontaktów oraz EditText do wpisywania fraz filtrujących liste.

Service CallingService

```
public void onStart(Intent intent, int startId) {
    final MessagesSaver messagesSaver = new MessagesSaver();

    broadcaster = LocalBroadcastManager.getInstance(this);



    final Thread t = new Thread() {
        @Override
        public void run() {
            try {

                mHandler.post(new Runnable() {
                    @Override
                    public void run() {

                        // User.getInstance().username = username;
                        Toast.makeText(getApplicationContext(), "Serwis
gotowy", Toast.LENGTH_SHORT).show();
                        Toast.makeText(getApplicationContext(), "Zalogowano
Użytkownika  " + User.getInstance().username, Toast.LENGTH_SHORT).show();

                        User.getInstance().sinchClient =
Sinch.getSinchClientBuilder()
                                .context(getApplicationContext())
                                        //.userId(username)
                                .userId(User.getInstance().username)
                                .applicationKey("3cc0c725-63fb-4410-a505-
c438eeea1041")

.applicationSecret("2V4L1bcagE+VcapWvc8gig==")
```

```
                                .environmentHost("sandbox.sinch.com")
                                .build();

                        //Rozpoczęcie nasłuchiwania połączeń przychodzących
                        messages =
messagesSaver.readmessages(getApplicationContext());



                        if(!User.getInstance().sinchClient.isStarted()) {
                            try{


User.getInstance().sinchClient.setSupportMessaging(true);

User.getInstance().sinchClient.setSupportCalling(true);
                            User.getInstance().sinchClient.start();


User.getInstance().sinchClient.addSinchClientListener(new
SinchClientListener() {
                                    public void onClientStarted(SinchClient
client) {

User.getInstance().sinchClient.startListeningOnActiveConnection();}
                                    public void onClientStopped(SinchClient
client) {
                                        User.getInstance().sinchClient =
null;
                                    }
                                    public void onClientFailed(SinchClient
client, SinchError error) {

Toast.makeText(getApplicationContext(), error.getMessage(),
Toast.LENGTH_LONG).show();}
                                    public void
onRegistrationCredentialsRequired(SinchClient client, ClientRegistration
registrationCallback) { }
                                    public void onLogMessage(int level,
String area, String message) { }
                                });


User.getInstance().sinchClient.getCallClient().addCallClientListener(new
SinchCallClientListener());
                            Calendar calendar = Calendar.getInstance();
                            SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("HH:mm");
                            final String hour =
simpleDateFormat.format(calendar.getTime());
                                User.getInstance().messageClient =
User.getInstance().sinchClient.getMessageClient();

User.getInstance().messageClient.addMessageClientListener(new
MessageClientListener() {
                                    @Override
                                    public void onIncomingMessage(MessageClient
messageClient, Message message) {
                                        messages =
messagesSaver.readmessages(getApplicationContext());
                                        messagesSaver.savemessage((new
ChatMessage(false, message.getTextBody(), message.getSenderId(), hour,
```

```
User.getInstance().username)), messages, getApplicationContext());
                                    User.getInstance().messages =
messagesSaver.readmessages(getApplicationContext());

                                    usr = message.getSenderId();
                                    Handler mainHandler = new
Handler(getApplicationContext().getMainLooper());

                                    Runnable myRunnable = new Runnable() {
                                        @Override
                                        public void run() {

Toast.makeText(getApplicationContext(), "Wiadomość od " + usr,
Toast.LENGTH_LONG).show();
                                            try {
                                                Uri notification =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
                                                Ringtone r =
RingtoneManager.getRingtone(getApplicationContext(), notification);
                                                r.play();
                                            } catch (Exception e) {
                                                e.printStackTrace();
                                            }
                                        } // This is your code
                                    };
                                    mainHandler.post(myRunnable);

                                    Toast.makeText(getApplicationContext(),
"Wiadomość od " + message.getSenderId(), Toast.LENGTH_LONG).show();
                                    Intent intent = new
Intent("messenger");
                                    intent.putExtra("message",
message.getTextBody());
                                    intent.putExtra("messagewho",
message.getSenderId());
                                    broadcaster.sendBroadcast(intent);
                                }

                                @Override
                                public void onMessageSent(MessageClient
messageClient, Message message, String s) {

                                }

                                @Override
                                public void onMessageFailed(MessageClient
messageClient, Message message, MessageFailureInfo messageFailureInfo) {

                                }

                                @Override
                                public void
onMessageDelivered(MessageClient messageClient, MessageDeliveryInfo
messageDeliveryInfo) {

                                }

                                @Override
                                public void
onShouldSendPushData(MessageClient messageClient, Message message,
List<PushPair> list) {
```

```
                                }
                            });}catch (Exception
e){Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_LONG).show();}
                        }




                    }
                });

            } finally {

            }
        }
    };
    t.start();



    super.onStart(intent, startId);
}
```

Serwis uruchamiany po poprawnym zalogowaniu użytkownika służący do nasłuchiwania połączeń przychodzących oraz monitorujący działania użytkownika. Na początku odbywa się ustalenie wszelkich parametrów klienta oraz poinformowanie serwera że klient będzie używał interfejsów rozmowy oraz czatu. Metoda onIncomingMessage definiuje zdarzenia dziejące się po otrzymaniu wiadomości natomiast onIncomingCall zdarzenia dziejące gdy inny użytkownik chce się z nami połączyć głosowo.

Klasa typu Singleton – User

```
public class User implements Serializable{
    public static User instance = new User();
    public String username;
    public String password;

    public MessageClient messageClient= null;
    public transient Call call;
    public transient SinchClient sinchClient = null;
    public List<ChatMessage> messages;


    protected User() {
        // Exists only to defeat instantiation.
    }
    public static User getInstance() {
        if(instance == null) {
            instance = new User();
        }
        return instance;
    }
}
```

Klasa User przechowuje informacje na temat użytkownika, do której instancji dostęp musi mieć kilka innych klas(stąd wykorzystanie Singletona)

Pola klasy:

- Pola username i password to informacje o loginie i haśle użytkownika.

- messageClient – klient serwera odpowiadającego za wiadomości

- call – zmienna opisująca połączenie. Jeśli jest nieaktywne ma wartość null

- sinchClient – klient serwera Sinch

- messages – lista przechowująca wiadomości

Plik AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Pozwolenia na wykorzystywanie funkcjonalności system Android.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="APP"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

Główne informacje na temat aplikacji.

```
<service android:name=".CallingService" />
```

Informacja o wykorzystaniu serwisu działającego w tle.

```
<activity
    android:name=".MainActivity"
```

```
    android:label="SKAMPER"
    android:screenOrientation="portrait"
    android:theme="@style/AppTheme.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:screenOrientation="portrait"
    android:name=".ContactsList"
    android:label="SKAMPER - Kontakty"
    android:theme="@style/AppTheme.NoActionBar" />
<!--
     ATTENTION: This was auto-generated to add Google Play services to your
project for
     App Indexing.  See https://g.co/AppIndexing/AndroidStudio for more
information.
-->
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />

<activity
    android:screenOrientation="portrait"
    android:name=".CallActivity"
    android:label="Połączenie"
    android:theme="@style/AppTheme.NoActionBar" />
<activity
    android:screenOrientation="portrait"
    android:name=".ConversationActivity"
    android:label="@string/title_activity_conversation"
    android:parentActivityName=".ContactsList"
    android:theme="@style/AppTheme.NoActionBar">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.example.wojtek_asus.app.ContactsList" />
</activity>
<activity
    android:screenOrientation="portrait"
    android:name=".RingingActivity"
    android:label="@string/title_activity_ringing"
    android:theme="@style/AppTheme.NoActionBar" />
<activity
    android:screenOrientation="portrait"
    android:windowSoftInputMode="adjustPan"
    android:name=".RegistrationActivity"
    android:label="SKAMPER - Rejestracja"
    android:theme="@style/AppTheme.NoActionBar" />
```

Informacje na temat poszczególnych Activities.

Grafiki wykorzystane do projektu:

- ikona akceptacji połączenia przychodzącego

- ikona odrzucenia połączenia przychodzącego

- ikona buttona inicjującego połączenie

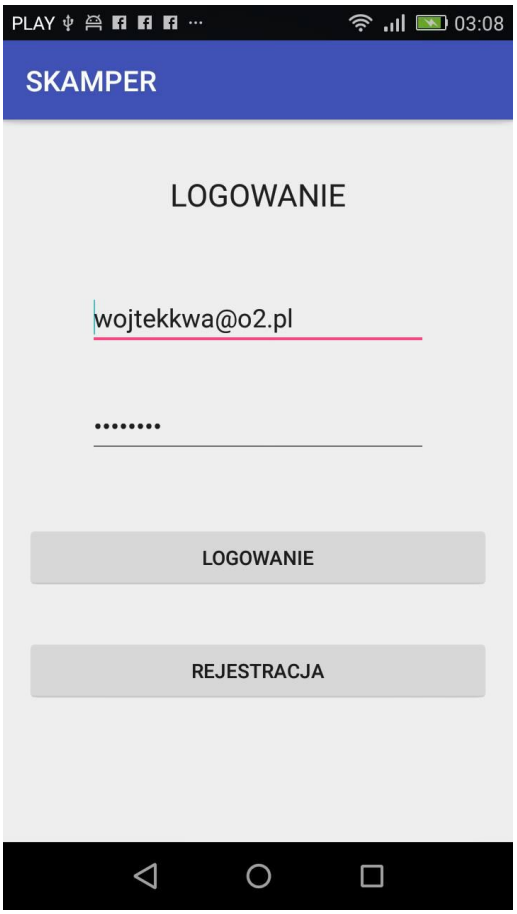
Pliki dźwiękowe wykorzystane do projektu:

- ring.mp3(znajduje się w folderze raw projektu)

6. Instrukcja obsługi aplikacji

(Jeżeli posiadasz już konto przejdź do punktu 3.)

1. Jeżeli uruchamiasz aplikację po raz pierwszy należy utworzyć konto, klikając przycisk REJESTRACJA.



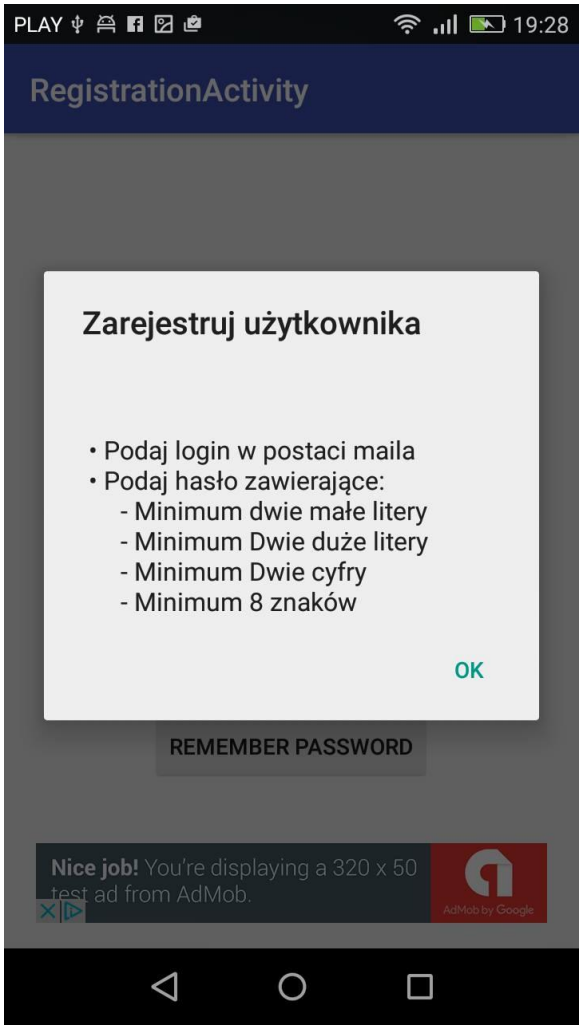
2. Po przejściu do okna rejestracji należy wpisać login (login musi przyjąć formę adresu e-mail, np. przykladowy@login.pl). Login również musi być unikatowy w przypadku wpisania loginu użytkownika już istniejącego program poinfromuje nas o zajętości ów nazwy. Po wpisaniu loginu należy wpisać hasło, które musi spełniać trzy wymogi:

   - ❖ Dwie duże litery
   - ❖ Dwie małe litery
   - ❖ Dwie cyfry

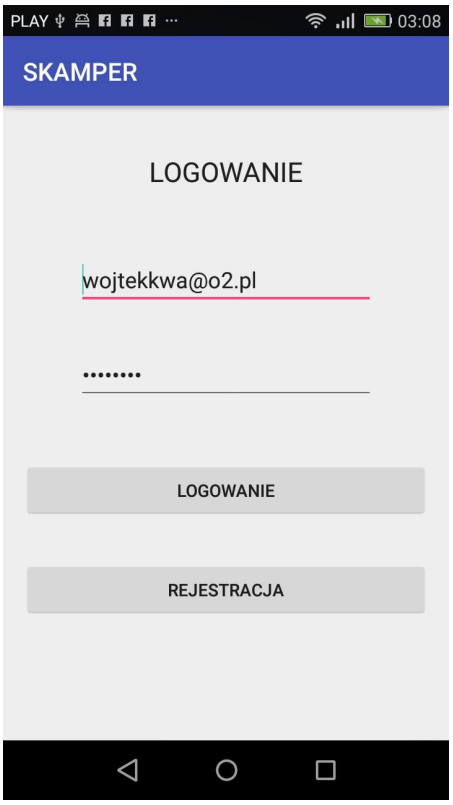
   Przykładowe akceptowane hasła : HaSlO123,qwe321ASD, SK4mp3R

   Po wypełnieniu obu pól prawidłowo naciśnij przycisk ZAREJESTRUJ.

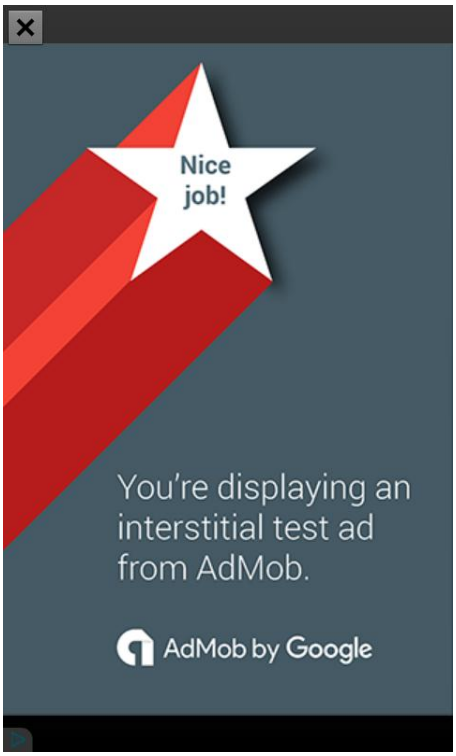
   Powinieneś otrzymać komunikat „Dodano użytkownika”



3. Jeżeli posiadasz już konto należy wpisać odpowiedni login oraz hasło i przycisnąć przycisk LOGOWANIE

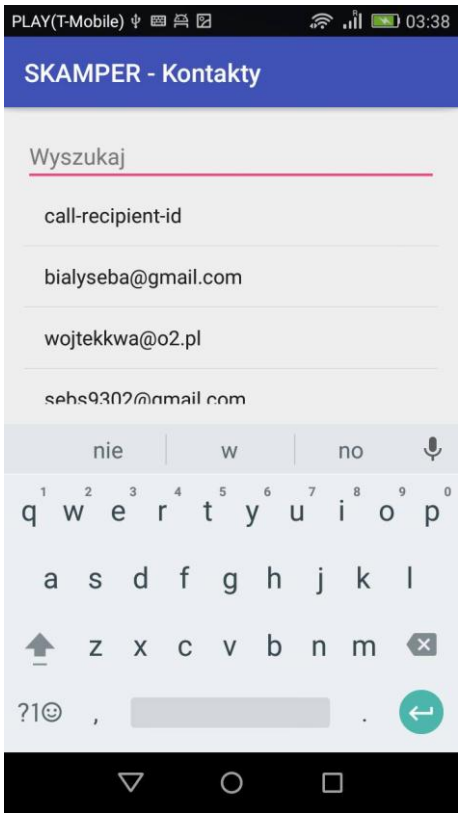


4. Jeżeli podane dane były prawidłowe wyświetli się nam ekran z reklamą w technologii RTB, w przeciwnym wypadku otrzymamy komunikat o braku powodzenia operacji logowania.

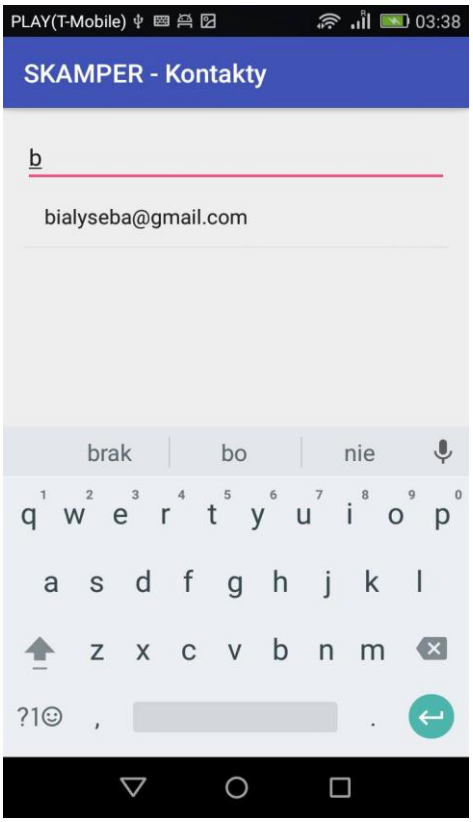


Aby zamknąć reklamę i przejść do aplikacji należy przycisnąć krzyżyk w lewym górnym rogu ekranu.

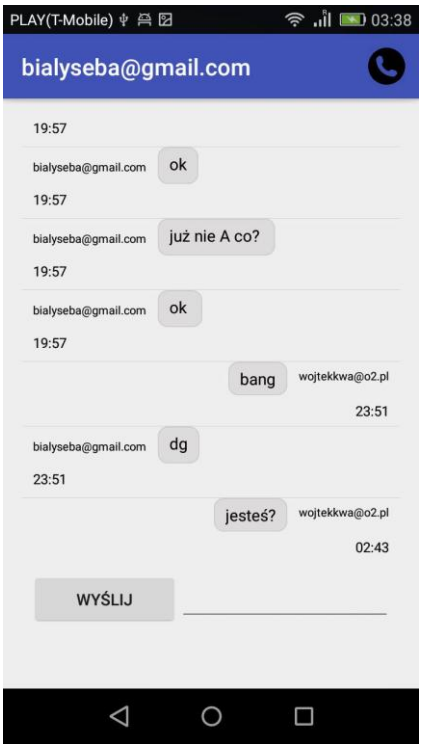
5. Po przejściu do następnego okna wyświetlona zostaje lista twoich kontaktów.



6. W celu odnalezienia jednego z kontaktów na liście należy przycisnąć pole do edycji powyżej czerwonej linii z napisem Wyszukaj. Następnie wpisywać kolejne litery loginu, aplikacja za pomocą autouzupełnienia pokaże nam znajomych o loginach zaczynających się wpisaną frazą.



7. Po wybraniu jednego ze znajomych wyświetla się okno chatu wraz z zarchiwizowanymi wiadomościami.

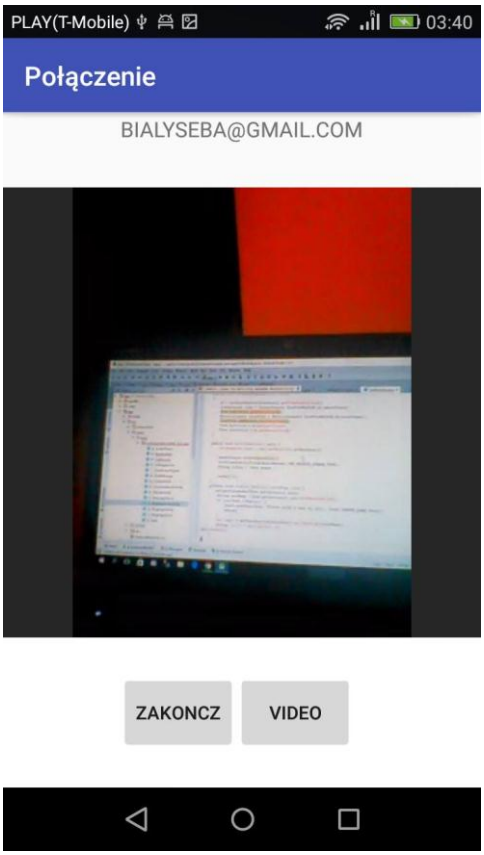


Aby wysłać wiadomość najpierw należy przycisnąć pole w okolicy poziomej linii w dole ekranu, pokaże się wtedy klawiatura po wpisaniu wiadomości należy przycisnąć przycisk WYŚLIJ.

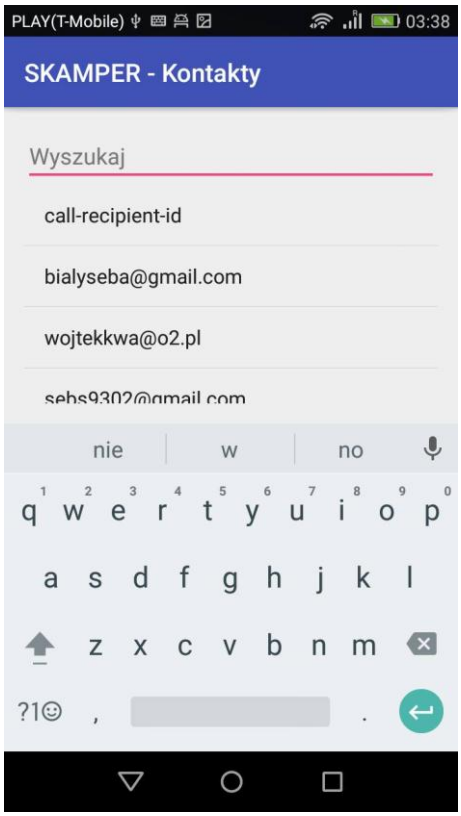
W celu wysłania zaproszenia do rozmowy należy przycisnąć przycisk słuchawki w prawym górnym rogu.

8. Po wysłaniu zaproszenia i akceptacji ze strony adresata, na ekranie urządzenia zostanie wyświetlone okno rozmowy.

9. Aby rozszerzyć rozmowę do wideorozmowy należy wybrać przycisk VIDEO.



10. Aby zakończyć połączenie należy wybrać przycisk ZAKONCZ, aplikacja powróci do okna listy kontaktów.



7. Instrukcja instalacji

W celu instalacji naszej aplikacji, należy pobrać plik APK, który jest plikiem instalacyjnym dla systemów Android. Po pobraniu badź skopiowaniu pliku na pamięć telefonu badź na kartę pamięci, należy za pomocą menadżera plików odnaleźć ów plik i wybrać go.

Po naciśnięciu system operacyjny Android poprosi o wybranie aplikacji, która przeprowadzi instalację. Domyślnie jest to wbudowana aplikacja systemu Eksplorator plików.

Po wybraniu narzędzia instalacyjnego system poprosi nas o akceptację możliwości ingerencji w nasz telefon. Należy tutaj zaznaczyć, iż nie chodzi tutaj o żadne szpiegowanie czy pozyskiwanie prywatnych danych, a akceptację, iż aplikacja może korzystać z internetu, słuchawki oraz głośnika, mikrofonu oraz kamery.

Należy tu zaznaczyć, iż nie jest to żadne złośliwe oprogramowanie. Każda aplikacja instalowana na systemach urządzeń mobilnych musi ów manifest przedstawić, gdyż na jego podstawie określane są możliwości aplikacji.

8. Podsumowanie i możliwe kierunki rozwoju aplikacji

Finalna wersja projektu spełnia założenia początkowe. Niestety ze względu zbyt dużą dokładność urządzenia zbierającego w postaci mikrofonu, implementacja trybu głośnomówiącego nie dała oczekiwanych efektów. Niestety dźwięk był niewyraźny, ilość szumów z tła nie pozwalała na zrozumienie przekazywanych wiadomości. Z tego też powodu porzuciliśmy dalsze prace nad tym modułem. Nie licząc tego modułu prezentowany projekt posiada wszystkie funkcjonalności zakładane przy jego planowaniu. Rozwój aplikacji jest jak najbardziej możliwy gdyż zostaje wiele dodatkowych funkcjonalności, które bez większego problemy mogły być zaimplementowane. Należy jednak pamiętać, iż idea powstania ów aplikacji zakłada oszczędność zasobów kosztem dodatkowych funkcji. Największym problemem podczas pracy nad projektem był konflikt na linii Android studio, a system operacyjny Windows. Emulator środowiska, w którym tworzyliśmy aplikacje ewidentnie miał problem ze stabilnym działaniem w systemie Windows. Środowiska informatyczne polecają do współpracy z Android studio system iOS, który można rzec jest kompatybilny z Android studio. Co do reszty wykorzystanych narzędzi nie mam żadnych zastrzeżeń.

9. Literatura

https://www.sinch.com/

https://firebase.google.com/

https://www.google.pl/admob/

https://www.wikipedia.org/

http://stackoverflow.com/