Instytut Informatyki
Kolegium Nauk Przyrodniczych
Uniwersytet Rzeszowski

Przedmiot:

Programowanie urządzeń mobilnych

Dokumentacja techniczna projektu:

Motivational App

Wykonał: Krystian Burbano-Marek

Prowadzący: mgr inż. Adam Szczur

Rzeszów 2023

1. Temat projektu i nazwa aplikacji

Motivaitonal App ma służyć użytkownikom, którzy szukają motywacji do działania. Tak jak nazwa wskazuje, aplikacja motywuje do działania i sprawia, że na twarzy użytkownika pojawia się szeroki uśmiech.

2. Cel projektu

Zaprojektowanie i wykonanie oprogramowania umożliwiającego wyświetlanie motywacyjnych cytatów mniej lub bardziej znanych autorów.

3. Funkcjonalności aplikacji

- Pobieranie i wyświetlanie motywacyjnych cytatów
- Ustawienie powiadomień na daną godzinę z motywacyjnym cytatem
- Latarka

4. Technologie

- Java 17
- jsoup 1.15.3
- Android Studio Dolphin 2021.3.1

5. Interesariusze aplikacji

Osoby potrzebujące motywacji lub szukające sensu istnienia, bądź działania.

6. Projekt GUI

GUI stanowią dwa pliki .xml, które nadają wygląd aplikacji. GUI zostało zaprojektowane tak, by przyciągało uwagę żywymi i uspokajającymi kolorami. Aplikacja składa się z widoku głównego oraz widoku ustawień.

activity_main.xml

activity_settings.xml

7. Struktura programu

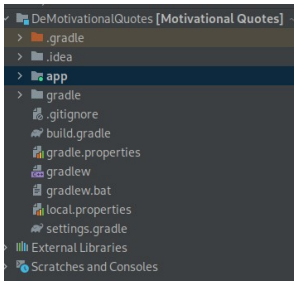
Struktura programu na pierwszy rzut oka może wydawać się złożona, lecz po zaznajomieniu się z programem Android Studio Dolphin można łatwo manewrować w plikach projektu.

7.1. Dane wykorzystywane przez program

Jedynymi danymi z których program korzysta są dane ze strony www.goodreads.com z której biorą się motywacyjne cytaty i ich autorzy wyświetlane w aplikacji.

7.2. Opis plików zewnętrznych

Struktura programu wygląda następująco:

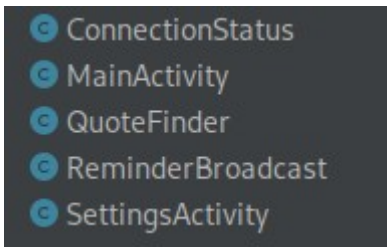


Najważniejszym folderem jest folder **app** w którym znajdują się wszystkie pliki zewnętrzne, które wchodzą w skład projektu (m.in. pliki źródłowe .java, .jpg, .xml, itp.).

W folderze **app/main/res** znajdują się pliki potrzebne do budowy aplikacji, a w folderze **app/main/java** są wszystkie pliki .java, które tworzą logikę aplikacji motywacyjnej.

7.3. Podział na moduły, komunikacja między modułami

W aplikacji istnieje 5 klas .java, które tworzą aplikację:



ConnectionStatus służy do sprawdzania czy jest połączenie z internetem do pobrania cytatów.

```java
public class ConnectionStatus {
    public boolean isConnectedToInternet() {
        try {
            String command = "ping -c 1 google.com";
            return (Runtime.getRuntime().exec(command).waitFor() == 0);
        } catch (Exception e) {
            return false;
        }
    }
}
```

QuoteFinder służy do znajdywania losowego cytatu.

```java
public class QuoteFinder {
    protected List<String> searchQuote() throws IOException {


        List<String> tags = new ArrayList<~>();
        // generate random number for page and quote
        Random rand = new Random();

        int n = rand.nextInt( bound: 99);

        // setting up the url to a random page
        String url = "https://www.goodreads.com/quotes/tag/motivational?page=" + (n + 1);


        // connecting and getting quotes
        Document doc = Jsoup.connect(url).get();
        Elements e = doc.select( cssQuery: "div.quoteText");


        int quotes = e.size();

        // get random quote from quotes
        n = rand.nextInt(quotes);

        String quote = e.get(n).text();

        // filter strings to get quote and author separately
        String quoteText = quote.replaceAll( regex: "\\—(.*)",  replacement: "");

        String quoteAuthor = quote.substring(quote.indexOf("—"), quote.length());

        System.out.println(quoteText);

        System.out.println(quoteAuthor);

        tags.add(quoteText);
        tags.add(quoteAuthor);

        return tags;
    }
}
```

Funkcja **SearchQuote()** Najpierw losujemy numer strony z której będzą pobrane cytaty (stron jest 100). Później łączymy się z tą stroną i czekamy na jej pełne załadowanie, bo cytaty wyświetlane są za pomocą JavaScripta, który wykonuje się po załadowaniu strony. Później ze strony jest wyciągany cytat i autor i przerabiany na potrzebę aplikacji. Funkcja zwraca listę znaków z cytatem i autorem.

MainActivity wyświetla na stronie startowej aplikacji cytat i autora. Pozwala też na losowanie innych cytatów, oraz wejście w ustawienia aplikacji.

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();

        StrictMode.setThreadPolicy(policy);
        Button newQuoteButton = findViewById(R.id.newQuoteButton);

        TextView quoteText = findViewById(R.id.quoteText);

        TextView authorText = findViewById(R.id.authorText);

        QuoteFinder quoteFinder = new QuoteFinder();
        List<String> quotes = null;
        try {
            quotes = quoteFinder.searchQuote();
        } catch (IOException e) {
            e.printStackTrace();
        }

        if (quotes == null) {
            quoteText.setText("No internet connection");
        } else {
            quoteText.setText(quotes.get(0));
            authorText.setText(quotes.get(1));

            String quote = quotes.get(0);
        }
        newQuoteButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                List<String> quotes = null;
                try {
                    quotes = quoteFinder.searchQuote();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                if (quotes == null) {
                    quoteText.setText("No internet connection");
                    authorText.setText("");
                } else {
                    quoteText.setText(quotes.get(0));
                    authorText.setText(quotes.get(1));
                }
            }
        });

        Button settingsButton = findViewById(R.id.settingsButton);

        settingsButton.setOnClickListener(view -> {
            startActivity(new Intent( packageContext: MainActivity.this, SettingsActivity.class));
        });
    }
}
```

SettingsActivity jest sekcją ustawień aplikacji do której przeniesiemy się po kliknięciu w przycisk „SETTINGS” na stronie głównej aplikacji. Można tam między innymi ustawić godzinę na codzienne powiadomienie z motywacyjnym cytatem, lub włączyć latarkę.

```
public class SettingsActivity extends AppCompatActivity {

    int hour = -1;
    int minute = -1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        createNotificationChannel();

        Button backButton = findViewById(R.id.notifyButtonBack);

        CameraManager cameraManager = (CameraManager) getSystemService(CAMERA_SERVICE);

        Switch lightSwitch = findViewById(R.id.lightSwitch);
        lightSwitch.setEnabled(true);

        lightSwitch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {

                if(isChecked){
                    try {
                        cameraManager.setTorchMode( cameraId: "0", enabled: true);
                    } catch (CameraAccessException e) {
                        e.printStackTrace();
                    }
                    lightSwitch.setText("FLASH ON");
                }
                else {
                    try {
                        cameraManager.setTorchMode( cameraId: "0", enabled: false);
                    } catch (CameraAccessException e) {
                        e.printStackTrace();
                    }
                    lightSwitch.setText("FLASH ON");
                }
            }
        });

        backButton.setOnClickListener(view -> {
            startActivity(new Intent( packageContext: SettingsActivity.this, MainActivity.class));
        });


        Button selectTimeButton = findViewById(R.id.selectTimeButton);

        selectTimeButton.setOnClickListener(view -> {
            TimePickerDialog.OnTimeSetListener onTimeSetListener = new TimePickerDialog.OnTimeSetListener() {
                @Override
                public void onTimeSet(TimePicker timePicker, int selectedHour, int selectedMinute) {
                    hour = selectedHour;
                    minute = selectedMinute;

                    selectTimeButton.setText(String.format(Locale.getDefault(), format: "%02d:%02d", hour, minute));
                }
            };

            int style = AlertDialog.THEME_DEVICE_DEFAULT_DARK;

            TimePickerDialog timePickerDialog = new TimePickerDialog( context: this, style, onTimeSetListener, hour, minute, is24HourView: true);

            timePickerDialog.setTitle(("Select time for a daily reminder"));
            timePickerDialog.show();
        });

        Button notifyButton = findViewById(R.id.notifyButton);

        notifyButton.setOnClickListener(view -> {
            ConnectionStatus connectionStatus = new ConnectionStatus();
            if (connectionStatus.isConnectedToInternet()) {
                if (hour != -1 && connectionStatus.isConnectedToInternet()) {

                    Intent intent = new Intent( packageContext: SettingsActivity.this, ReminderBroadcast.class);
```

```
                PendingIntent pendingIntent;
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    pendingIntent = PendingIntent.getBroadcast( context: this,
                            requestCode: 0, intent,  flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_MUTABLE);

                } else {
                    pendingIntent = PendingIntent.getBroadcast( context: this,
                            requestCode: 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);

                }

                AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);

                Calendar setTimeCalendar = Calendar.getInstance();

                int hourNow = setTimeCalendar.get(setTimeCalendar.HOUR_OF_DAY);
                int minuteNow = setTimeCalendar.get(setTimeCalendar.MINUTE);
                int secondNow = setTimeCalendar.get(setTimeCalendar.SECOND);

                System.out.println("current hour: " + hourNow);
                System.out.println("current minute: " + minuteNow);
                System.out.println("current second: " + secondNow);
                setTimeCalendar.set(Calendar.HOUR_OF_DAY, hour);
                setTimeCalendar.set(Calendar.MINUTE, minute);
                setTimeCalendar.set(Calendar.SECOND, 00);

                if (setTimeCalendar.getTimeInMillis() < System.currentTimeMillis()) {
                    Log.e( tag: "setAlarm", msg: "time is in past");
                    setTimeCalendar.add(Calendar.DAY_OF_YEAR,  i1: 1); // it will tell to run to next day
                }

                System.out.println("reminder hour: " + setTimeCalendar.get(setTimeCalendar.HOUR_OF_DAY));
                System.out.println("reminder minute: " + setTimeCalendar.get(setTimeCalendar.MINUTE));
                System.out.println("reminder second: " + setTimeCalendar.get(setTimeCalendar.SECOND));

                System.out.println("reminder in: " + (setTimeCalendar.get(setTimeCalendar.HOUR_OF_DAY) - hourNow));


                alarmManager.setRepeating(AlarmManager.RTC_WAKEUP, setTimeCalendar.getTimeInMillis(),AlarmManager.INTERVAL_DAY, pendingIntent);
                alarmManager.set(AlarmManager.RTC_WAKEUP, setTimeCalendar.getTimeInMillis(), pendingIntent);
                Toast.makeText( context: this,  text: "Reminder set!", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText( context: this,  text: "Time is not set!", Toast.LENGTH_SHORT).show();
            }
        } else Toast.makeText( context: this,  text: "No internet connection", Toast.LENGTH_SHORT).show();
    });

}

private void createNotificationChannel() {

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = "notificationChannel";
        String description = "Channel for motivational notification";
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel( id: "notcha1", name, importance);
        channel.setDescription(description);

        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}

}
```

Fukncja **createNotificationChannel()** jest potrzebna do stworzenia kanału powiadomień, które tworzone są w klasie **ReminderBroadcast**

ReminderBroadcast to klasa w której tworzone jest ciało powiadomienia. Jest ona później używana do wyświetlenia powiadomienia od strony „SETTINGS”.

```
public class ReminderBroadcast extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        QuoteFinder quoteFinder = new QuoteFinder();

        List<String> quotes = null;
        try {
            quotes = quoteFinder.searchQuote();
        } catch (IOException e) {
            e.printStackTrace();
        }

        NotificationCompat.Builder builder = new NotificationCompat.Builder(context,  channelId: "notcha1")
                .setSmallIcon(R.drawable.ic_thumb)
                .setContentTitle("Your motivational quote!")
                .setContentText("")

                .setSound(null)

                .setStyle(new NotificationCompat.BigTextStyle().bigText(quotes.get(0) + quotes.get(1)))

                .setPriority(NotificationCompat.PRIORITY_DEFAULT);

        NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        Notification notification = builder.build();

        notificationManager.notify( id: 100, notification);

        playNotificationSound(context);
    }

    void playNotificationSound(Context context){
        Uri sound = Uri.parse("android.resource://" + context.getPackageName() + "/" + R.raw.sound);
        Ringtone r = RingtoneManager.getRingtone(context, sound);
        r.play();
    }
}
```

Funkcja **onReceive()** tworzy ciało powiadomienia, które potem jest wyświetlanie w stronie „SETTINGS”. Dźwięk powiadomienia jest ustawiony w funkcji **playNotificationSound()**.

8. Literatura

- https://developer.android.com/guide

- https://stackoverflow.com/questions/72423948/show-multiple-daily-repeating-notifications-in-android-studio