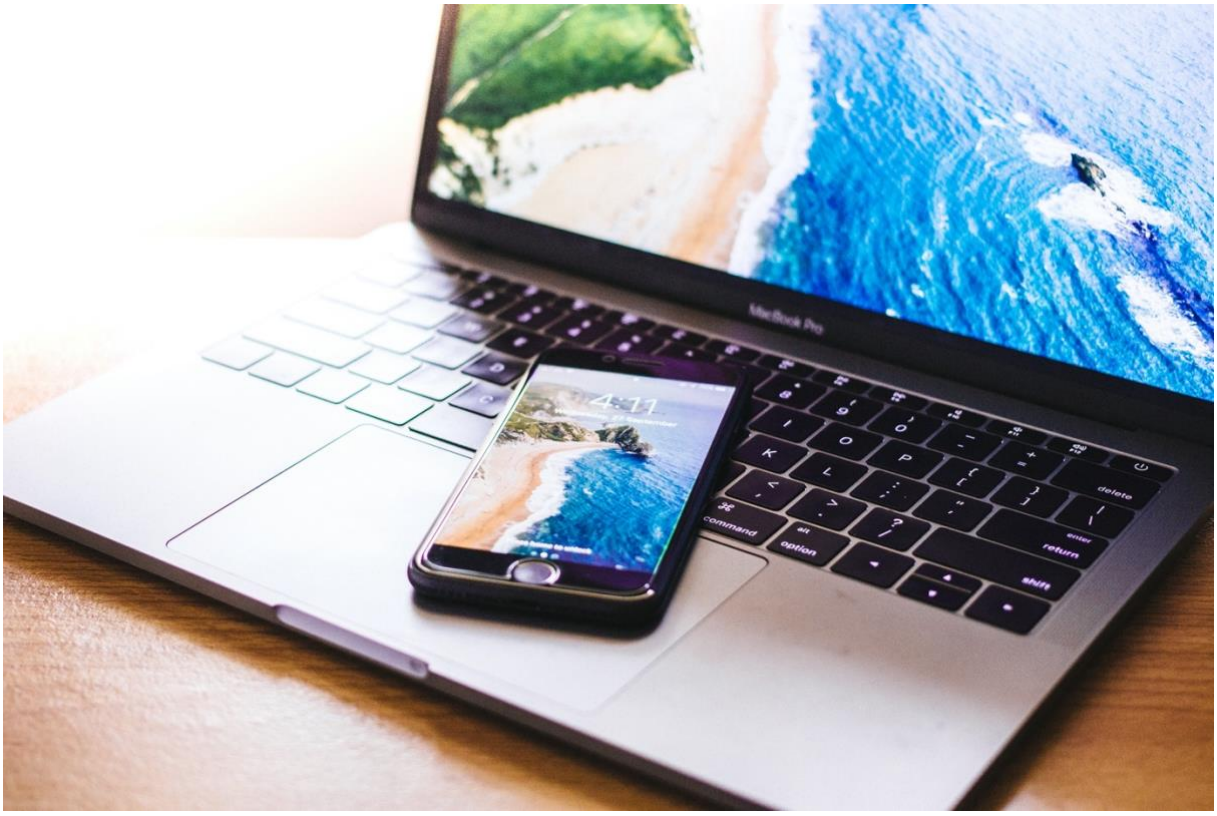
Aplikacje mobilne. Projektowanie aplikacji Android. Programowanie aplikacji dla różnych wielkości ekranów.



Urządzenia z Androidem mają różne kształty i rozmiary, więc układ aplikacji musi być elastyczny i responsywny. Oznacza to, że zamiast definiować układ za pomocą sztywnych wymiarów, które zakładają określony rozmiar ekranu i proporcje, układ powinien automatycznie reagować na różne rozmiary i orientacje ekranu. Android obsługuje dwie orientacje: pionową i poziomą.
Dostosowanie się do różnych rozmiarów ekranu niekoniecznie oznacza, że aplikacja jest kompatybilna ze wszystkimi urządzeniami Androida jak Android Wear, TV, Auto.

Najlepszym sposobem na stworzenie responsywnego układu dla różnych rozmiarów ekranu jest użycie ConstraintLayout jako układu podstawowego interfejsu użytkownika. ConstraintLayout umożliwia określenie położenia i rozmiaru każdego widoku zgodnie z relacjami z innymi widokami w układzie. W ten sposób wszystkie widoki mogą się przesuwać i rozciągać razem, gdy zmienia się rozmiar ekranu.

Aby mieć pewność, że układ jest elastyczny, responsywny i dostosowuje się do różnych rozmiarów ekranu, należy używać "wrap_content" oraz "match_parent" dla szerokości i wysokości większości komponentów widoku zamiast zakodowanych na stałe rozmiarów.

- "wrap_content" informuje widok, aby ustawił swój rozmiar na taki, jaki jest niezbędny do dopasowania zawartości w tym widoku.
- "match_parent" powoduje, że widok rozwija się tak bardzo, jak to możliwe w widoku macierzystym.

Zadanie :

W emulatorze AVD przetestuj kilka swoich ostatnich aplikacji wykonanych na zajęciach na urządzeniu typu smartfon, tablet oraz Android TV. Sprawdź jak wygląda aplikacja w układzie poziomym i pionowym. Jakie płyną wnioski z tego zadania?

Interfejs użytkownika zaprojektowany dla telefonu prawdopodobnie nie zapewnia dobrego widoku na tablecie. W związku z tym aplikacja powinna również zapewniać alternatywne zasoby układu, aby w przystępny sposób prezentować interfejs użytkownika dla określonych rozmiarów ekranu.

Można utworzyć dodatkowe układy w res/layout/ , tworząc dodatkowe pliki - po jednym dla każdej konfiguracji ekranu, która wymaga innego układu - a następnie dołączając kwalifikator konfiguracji ekranu do layoutu np. layout-w600dp dla ekranów, które mają dostępną szerokość 600 dp.

Najmniejsze wartości szerokości odpowiadają typowym rozmiarom ekranu:

- 320dp: typowy ekran telefonu (240x320 ldpi, 320x480 mdpi, 480x800 hdpi itp.).
- 480dp: duży ekran telefonu ~ 5 "(480x800 mdpi).
- 600dp: 7-calowy tablet (600x1024 mdpi).
- 720dp: 10-calowy tablet (720x1280 mdpi, 800x1280 mdpi itp.).

Zadanie :

Chcemy używać trybu pionowego na telefonach z Androidem oraz trybu pionowego i poziomego na tabletach. Jak to zrobić?

1. Stwórz nowy zasób xml w res / values jako portrait_landscape.xml.
2. Wklej do tego pliku poniższy kod :

```
<? xml version = "1.0" encoding = "utf-8"?>
<resources>
   <bool name = "portrait_only"> true </bool>
</resources>
```

3. Stwórz następnie nowy plik typu values w folderze res o nazwie values-sw600dp i dodaj poniższy kod :

```
<? xml version = "1.0" encoding = "utf-8"?>
<resources>
   <bool name = "portrait_only"> false </bool>
</resources>
```

4. Następnie w metodzie onCreate w MainActivity.kt należy dodać

```
if (getResources (). getBoolean (R.bool. portrait_only )) {
 setRequestedOrientation (ActivityInfo. SCREEN_ORIENTATION_PORTRAIT );
}
```

Więcej :
https://stackoverflow.com/questions/9627774/android-allow-portrait-and-landscape-for-tablets-but-force-portrait-on-phone

Możemy wyłączyć orientację w aplikacji na Androida. Aby to zrobić musimy w Manifeście zadeklarować na sztywno wybraną orientację. W tym przykładzie pokazuję jak wyłączyć tryb poziomy w systemie Android.
<activity android:name = ".MainActivity"

```
    android:screenOrientation = "portrait">
    <intent-filter>
      <action android:name = "android.intent.action.MAIN" />
      <category android:name = "android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
```

Zadanie:
Stwórzmy teraz aplikację, która będzie pobierać rzeczywiste dane : szerokość i wysokość ekranu.

1. Utwórz aplikację o nazwie displayMetricsApp oraz przejdź do pliku activity_main.xml z jednym TextView.
2. Przykładowy kod activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <TextView
    android:id="@+id/textV"
    android:textSize="30px"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

</android.support.constraint.ConstraintLayout>
```

3. Następnie musimy pobrać dane i je wyświetlić. Odpowiada za to poniższy kod:

```
val displayMetrics = DisplayMetrics()
windowManager.defaultDisplay.getMetrics(displayMetrics)

var width = displayMetrics.widthPixels
var height = displayMetrics.heightPixels
```

4. W MainActivity.kt umieść :

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*
import android.util.DisplayMetrics


class MainActivity : AppCompatActivity() {

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // get device dimensions
    val displayMetrics = DisplayMetrics()
    windowManager.defaultDisplay.getMetrics(displayMetrics)
```

```
        var width = displayMetrics.widthPixels
        var height = displayMetrics.heightPixels

        textV.text = width.toString() + " x " +height.toString()
    }
}
```

5. Przeanalizujmy kod:
   Aby uzyskać szerokość i wysokość ekranu Androida potrzebujemy,
   - Utworzyć obiekt DispalyMetrics ().
   - Przekazać obiekt displayMetrics do metody getMetrics() klasy Display.
   - WindowManager.defaultDisplay () zwraca obiekt Display.
   - Szerokość ekranu można uzyskać za pomocą metody displayMetrics.widthPixels
   - Wysokość ekranu można uzyskać za pomocą metody displayMetrics.heightPixels

PDFViewer

Aby korzystać z tej biblioteki, potrzebujesz pliku PDF w systemie. Następnie możesz napisać ten wiersz kodu, aby dołączyć go do tej biblioteki i wyświetlić plik w aplikacji.

1. Utwórz projekt o nazwie PDFViewer. W pliku build.gradle(Module:app) w dependencies należy umieścić wpis informujący o dołączeniu biblioteki.

   ```
   implementation "com.pdfview:pdfview-android:1.0.0"
   ```

2. Dodaj do swojego projektu w build.gradle.kts wpis:

   ```
   repositories {
       <...>
       jcenter()
   }
   ```

3. **W pliku MainActivity chcąc wyświetlać dokumenty .pdf musimy wywołać działanie:**

   ```
   findViewById<PDFView>(R.id.activityMainPdfView).fromAsset("dokument.pdf").show()
   ```

4. **W pliku activity_main.xml musimy dopisać komponent odpowiadający za wyświetlenie w ramce dokumentu .pdf.**

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <android.support.constraint.ConstraintLayout
   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">

     <com.pdfview.PDFView
       android:id="@+id/activityMainPdfView"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />

   </android.support.constraint.ConstraintLayout>
   ```

5. Sprawdź czy plik MainActivity.kt wygląda tak jak poniżej:

6.

```
import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import com.pdfview.PDFView

class MainActivity : AppCompatActivity() {

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
findViewById<PDFView>(R.id.activityMainPdfView).fromAsset("dokument.pdf").show()
   }
}
```

7. **Plik pdf o nazwie dokument.pdf trzeba umieścić w folderze Assets.**

ScrollView.

Jego cechą jest to, że obszar szablonu może wystawać poza wyznaczoną fizyczną wielkość wyświetlacza smartfona, stąd też istnieje możliwość przewijania w dół/górę.

1. Utwórz nowy projekt o nazwie ScrollView i pustej aktywności. W tym zadaniu będziemy tworzyć aktywność/podstronę naszej aplikacji, która będzie prezentowała informacje o kotach.
2. Do folderu drawable dodaj zdjęcie kota syberyjskiego o nazwie kot.jpg i umieść obrazek na samej górze w activity_main.xml.
3. Dodaj dwa TextView w którym to jeden będzie posiadał id o nazwie kot_syberyjski a drugi kot_opis, gdzie będzie przechowywana dość obszerna informacja na temat kota, powodująca nie zmieszczenie się tekstu na ekranie naszej aplikacji.
4. W strings.xml zadeklaruj:

<string name="kot_syberyjski">Kot syberyjski</string>
<string name="kot_opis">Kot syberyjski to rasa tak stara, że wprost niemożliwe jest odgadnięcie jej pierwotnego pochodzenia. Niegdyś koty tej rasy gościły u samego cara, dziś możemy cieszyć się ich obecnością w naszych domach. I choć „syberyjczyki” to prawdziwe kocie gwiazdy, w Rosji wciąż uważane są za zwykłe dachowce. Jaki jest kot syberyjski? Poznaj tę niezwykłą rasę.\n
1. Nie uczula alergików.Pojawiają się opinię, że kot syberyjski nie uczula alergików. Choć prawdopodobnie nie jest to do końca prawda, to faktycznie „syberian” nie wywołuje tak silnych objawów, jak inne rasy. Badania nad tym tematem wciąż trwają, jednak zdążono już udowodnić, że ślina tego kota posiada dużo mniejszą ilość białka Fel d 1 odpowiedzialnego za reakcje alergiczne. Kot syberyjski zdecydowanie rzadziej wywołuje objawy alergii. Nie oznacza to jednak, że jest zupełnie hipoalergiczny.\n
2. Mały lew „Sybirak” może pochwalić się wyjątkowo piękną kryzą. Ta „ozdoba” nie tylko przyciąga pełne podziwu spojrzenia, ale ma także ważne zadanie do spełnienia. To kołnierz chroniący wrażliwą szyję przed atakami innych drapieżników. Zupełnie jak u lwa! Kryza kota syberyjskiego nie powinna być zbyt długa. Musi natomiast mieć owalny kształt (inaczej niż w przypadku kota norweskiego leśnego, którego kryza powinna być trójkątna). Taka grzywa to bez wątpienia powód do dumy."\m
3. Jaki jest kot syberyjski? To prawdziwy przyjaciel
Opiekunowie kotów syberyjskich są zazwyczaj zachwyceni więzią, jaka łączy ich z pupilem. Mruczek daje się wychowywać i tworzy z człowiekiem silną relację. Jest niezwykle uczuciowy i, choć zachowuje niezależność, to zwykle nie kryje swoich emocji, śmiało tuląc się i mrucząc. Raz na jakiś czas spogląda przy tym na opiekuna

rozkochanym wzrokiem. To bardzo towarzyska rasa, nielubiąca samotności, a wprost kochająca przebywanie w otoczeniu ludzi. Niekoniecznie będą to nasze kolana, lecz z pewnością zawsze bliskie otoczenie.\n4. Mimo sporych rozmiarów, słusznej wagi i mocnej, muskularnej budowy ciała, zdziwić może fakt, że kot syberyjski jest niezwykle zwinny i atletyczny. Znany jest ze swojej skoczności i zamiłowania do wspinaczek. No cóż, to naturalna rasa wywodząca się z lasów. Nie straszne mu żadne wysokie cele. Zawsze znajdzie sposób, by się na nich zameldować. Wysokości to jego ulubienie miejsca do spędzania czasu. Jednak zanim wykona skok, dobrze sprawdzi, czy to aby na pewno bezpieczny ruch. W efekcie „syberian" staje na tylnych łapach, chcąc zobaczyć, co jest wyżej i dopiero gdy zbada teren wzrokiem – podejmuje decyzję o skoku.\n5. Czesanie? Wolę kąpiel ponieważ w naturze kot ten był często zmuszony do brodzenia w płytkiej wodzie, dosyć dobrze znosi kąpiele. Czasami nawet woli je od czesania. Na tę czynność pielęgnacyjną reaguje różnie. Niektóre koty wręcz nie znoszą szczotki. Choć kąpanie nie wpływa zbytnio na stan gęstej i wysoko wodoodpornej sierści, to bywa niezbędne, gdy pupil pobrudzi swoje długie „portki".
</string>

5. Jak widzisz w kot_opis znajduje się dość obszerna informacja. W activity_main.xml z poziomu Pallette dodaj komponent ScrollView. Poprawny plik powinien wyglądać tak:

```
<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="wrap_content"
android:layout_height="wrap_content"
tools:context=".MainActivity">
<ScrollView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_alignParentStart="true"
android:layout_alignParentTop="true">
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">
<ImageView
android:id="@+id/image"
android:layout_width="match_parent"
android:layout_height="294dp"
android:layout_gravity="center"
android:layout_marginTop="0dp"
android:src="@drawable/kot" />
<TextView
android:id="@+id/txtView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:text="@string/kot_syberyjski"
android:textSize="30sp" />
<TextView
android:id="@+id/txtView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:layout_gravity="center"
android:clickable="false"
android:justificationMode="inter_word"
android:text="@string/kot_opis"
android:textSize="15sp" />
</LinearLayout>
</ScrollView>
</RelativeLayout>
```

6. Sprawdź poprawność działania aplikacji. Czy aplikacja przesuwa się w dół i w górę?

HorizontalScrollView

HorizontalScrollView to komponent, który umożliwia przewijanie w poziomie. Może posłużyć jako slider zdjęć/obrazków w naszej aplikacji. Może być to dobra podstawa do stworzenia galerii zdjęć.

1. Utwórz aplikację HorizontalScrollView, a następnie **pliku activity_main.xml dodaj :**

```
<?xml version="1.0" encoding="utf-8"?>
<HorizontalScrollView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:orientation="horizontal">

    <ImageView
      android:id="@+id/image1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginLeft="20dp"
      android:layout_marginRight="20dp"
      android:contentDescription="@string/no_image"
      android:src="@drawable/pers"/>

    <ImageView
      android:id="@+id/image2"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginRight="20dp"
      android:contentDescription="@string/no_image"
      android:src="@drawable/pers"/>
    <ImageView
      android:id="@+id/image3"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginRight="20dp"
      android:contentDescription="@string/no_image"
```

```
        android:src="@drawable/pers"/>

    <ImageView
        android:id="@+id/image4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginRight="20dp"
        android:contentDescription="@string/no_image"
         android:src="@drawable/pers"/>

    <ImageView
        android:id="@+id/image5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginRight="20dp"
        android:contentDescription="@string/no_image"
         android:src="@drawable/pers"/>

    <ImageView
        android:id="@+id/image6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginRight="20dp"
        android:contentDescription="@string/no_image"
        android:src="@drawable/pers"/>

  </LinearLayout>
</HorizontalScrollView>
```

2. Dodaj 6 zdjęć/obrazków oraz zmień android:src="@drawable/pers" na właściwy dla nowych obrazków.
3. Zadeklaruj odpowiednie stringi w folderze res/values/strings.xml dla app_name oraz no_image.
4. W pliku MainActivity nie za wiele się zmienia.

```
import android.os.Bundle
import android.support.v7.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
  }
}
```

5. Spróbuj dostosować galerię zdjęć by zdjęcia ładowały się na cały ekran.

WebView.

WebView to widżet, który wyświetla stronę internetową. Ta klasa służy do implementacji własnej przeglądarki internetowej lub po prostu wyświetlania treści online w aktywności.

1. Stwórz nowy projekt o nazwie WebViewApp. W naszej aplikacji musimy mieć dostęp do Internetu, aby załadować stronę. Dodamy poniżesze uprawnienia internetowe w **AndroidManifest.xml przed <application>… </application>.**

   <uses-permission android:name="android.permission.INTERNET" />
2. W pliku activity_main.xml dodaj

```
<WebView
  android:id="@+id/webview"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
/>
```
3. W pliku MainActivity.kt musimy wywołać otwarcie strony poprzez dodanie w metodzie onCreate poniższego kodu:

   webView.loadUrl("https://www.google.com/")

4. **Sprawdź czy wyświetla się strona?**

Zadanie:
Stwórz aplikację zawierającą komponenty ScrollView, HorizontalScrollView oraz ListView dedykowaną jakiejś firmie z trybem poziomym i pionowym ekranu.