|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №7**

**«Программирование виджетов рабочего экрана»**

**ДИСЦИПЛИНА: «Разработка мобильного ПО»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-62Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Шаматрин А.Г. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2023

**Цель:** формирование практических навыков создания пользовательских виджетов.

**Задачи:**

1. Научиться создавать пользовательские виджеты для рабочих экранов.
2. Научиться использовать конфигурационные окна для настройки виджетов.
3. Уметь понимать схемы взаимодействия виджета с другими элементами платформы Android.
4. Разработать эффективное приложение с учетом аппаратных ограничений мобильных устройств.
5. Научиться реализовывать логику работы приложения с учетом специфики платформы Android.

**Вариант 4**

Создать виджет отображающий случайные изображения галереи, имеющиеся на устройстве. Время отображения задается программно. Предусмотреть принудительное обновление виджета. При каждом принудительном обновлении должна меняться фотография.

**Листинг:**

***RandomPictureWidget.kt***

package com.blackline.lw3\_2\_7  
  
import android.app.PendingIntent  
import android.appwidget.AppWidgetManager  
import android.appwidget.AppWidgetProvider  
import android.content.ComponentName  
import android.content.Context  
import android.content.Intent  
import android.net.Uri  
import android.os.Handler  
import android.os.Looper  
import android.provider.MediaStore  
import android.widget.RemoteViews  
  
class RandomPictureWidget : AppWidgetProvider() {  
 var imagePaths: List<String>? = null  
 private val ACTION\_WIDGET\_CLICKED = "com.blackline.lw3\_2\_7.WIDGET\_CLICKED"  
  
 private val handler = Handler(Looper.getMainLooper())  
 private lateinit var runnable: Runnable  
  
 override fun onUpdate(context: Context, appWidgetManager: AppWidgetManager?, appWidgetIds: IntArray?) {  
 super.onUpdate(context, appWidgetManager, appWidgetIds)  
  
 if (imagePaths == null)  
 imagePaths = *getImagePaths*(context)  
  
 runnable = object : Runnable {  
 override fun run() {  
 val views = RemoteViews(context?.*packageName*, R.layout.*random\_picture\_widget*)  
  
 views.setImageViewUri(R.id.*widgetImageView*, Uri.parse(imagePaths!![(0..9).*random*()]))  
  
 appWidgetManager?.updateAppWidget(appWidgetIds, views)  
 handler.postDelayed(this, 2000)  
 }  
 }  
 handler.postDelayed(runnable, 0)  
  
 val intent = Intent(context, RandomPictureWidget::class.*java*)  
 intent.*action* = ACTION\_WIDGET\_CLICKED  
 val pendingIntent = PendingIntent.getBroadcast(context, 0, intent, PendingIntent.*FLAG\_MUTABLE*)  
  
 val views = RemoteViews(context.*packageName*, R.layout.*random\_picture\_widget*)  
 views.setOnClickPendingIntent(R.id.*widgetImageView*, pendingIntent)  
 appWidgetManager?.updateAppWidget(appWidgetIds, views)  
 }  
  
 override fun onReceive(context: Context, intent: Intent) {  
 super.onReceive(context, intent)  
  
 if (intent.*action*?.equals(ACTION\_WIDGET\_CLICKED) == true) {  
 val appWidgetManager = AppWidgetManager.getInstance(context)  
 val appWidgetIds = appWidgetManager.getAppWidgetIds(  
 ComponentName(context, RandomPictureWidget::class.*java*)  
 )  
  
 if (imagePaths == null)  
 imagePaths = *getImagePaths*(context)  
  
 val views = RemoteViews(context?.*packageName*, R.layout.*random\_picture\_widget*)  
  
 views.setImageViewUri(R.id.*widgetImageView*, Uri.parse(imagePaths!![(0..9).*random*()]))  
  
 appWidgetManager?.updateAppWidget(appWidgetIds, views)  
 }  
 }  
  
 override fun onDisabled(context: Context?) {  
 super.onDisabled(context)  
 handler.removeCallbacks(runnable)  
 }  
}  
  
internal fun getImagePaths(context: Context): List<String> {  
 val imagePaths = *mutableListOf*<String>()  
 val projection = *arrayOf*(MediaStore.Images.Media.*DATA*)  
 val cursor = context.*contentResolver*.query(  
 MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*,  
 projection,  
 null,  
 null,  
 MediaStore.Images.ImageColumns.*DATE\_MODIFIED* + " DESC"  
 )  
 cursor!!.moveToFirst()  
 while (!cursor.*isAfterLast*) {  
 val pathIndex = cursor.getColumnIndex(MediaStore.Images.Media.*DATA*)  
 val imagePath = cursor.getString(pathIndex)  
 imagePaths.add(imagePath)  
 cursor.moveToNext()  
 if (imagePaths.*count*() >= 10) break  
 }  
 cursor.close()  
  
 return imagePaths  
}

***random\_picture\_widget.xml***

<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"  
 style="@style/Widget.LW327.AppWidget.Container"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:theme="@style/Theme.LW327.AppWidgetContainer">  
  
 <ImageView  
 android:id="@+id/widgetImageView"  
 android:layout\_width="400dp"  
 android:layout\_height="400dp"  
 tools:srcCompat="@tools:sample/avatars" />  
</RelativeLayout>

***random\_picture\_widget\_info.xml***

*<?*xml version="1.0" encoding="utf-8"*?>*<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"  
 android:description="@string/app\_widget\_description"  
 android:initialKeyguardLayout="@layout/random\_picture\_widget"  
 android:initialLayout="@layout/random\_picture\_widget"  
 android:minWidth="110dp"  
 android:minHeight="110dp"  
 android:previewImage="@drawable/example\_appwidget\_preview"  
 android:previewLayout="@layout/random\_picture\_widget"  
 android:resizeMode="horizontal|vertical"  
 android:targetCellWidth="2"  
 android:targetCellHeight="2"  
 android:updatePeriodMillis="3000"  
 android:widgetCategory="home\_screen" />

***AndroidManifest.xml***

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools" >  
  
 <uses-permission android:name="android.permission.READ\_MEDIA\_IMAGES" />  
 <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.LW327"  
 tools:targetApi="31" >  
  
 <receiver  
 android:name=".RandomPictureWidget"  
 android:exported="false" >  
 <intent-filter>  
 <action android:name="android.appwidget.action.APPWIDGET\_UPDATE" />  
 </intent-filter>  
  
 <meta-data  
 android:name="android.appwidget.provider"  
 android:resource="@xml/random\_picture\_widget\_info" />  
 </receiver>  
  
 <activity  
 android:name=".MainActivity"  
 android:exported="true" >  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

**Результат:**



**Рис. 1.** Результат

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки создания пользовательских виджетов.