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 №6

«Программирование виджетов рабочего экрана»

ДИСЦИПЛИНА: «Разработка мобильного ПО»

Выполнил: студент гр. ИУК4-52Б _____ (_____Боков А.А._____)
(Подпись) (Ф.И.О.)

Проверил: _____ (_____Прудяк П.Н._____)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2024 г.

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.

Формулировка задания:

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:

RandomImageWidget.java:

```
package com.example.android_dev_lab6;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.ComponentName;
import android.content.ContentUris;
import android.content.Context;
import android.content.Intent;
import android.graphics.BitmapFactory;
import android.os.Looper;
import android.os.SystemClock;
import android.widget.RemoteViews;
import android.net.Uri;
import android.database.Cursor;
import android.content.ContentResolver;
import android.graphics.Bitmap;
import android.provider.MediaStore;
import android.util.Log;
```

```
import android.os.Handler;
import android.app.PendingIntent;
import android.os.Bundle;

import java.io.IOException;
import java.io.InputStream;
import java.lang.ref.WeakReference;
import java.util.ArrayList;
import java.util.Random;

public class RandomImageWidget extends AppWidgetProvider {

    private static ArrayList<Uri> imageUris = new ArrayList<>();
    private static final String ACTION_UPDATE_WIDGET =
"com.example.randomimagewidget.UPDATE_WIDGET";
    private static final int UPDATE_INTERVAL = 10000; // 10 секунд
    private static ArrayList<WeakReference<Bitmap>> imageCache = new ArrayList<>();


    public static void clearImageCache() {
        for (WeakReference<Bitmap> weakBitmap : imageCache) {
            Bitmap bitmap = weakBitmap.get();
            if (bitmap != null && !bitmap.isRecycled()) {
                bitmap.recycle(); // Освобождаем память
                Log.d("RandomImageWidget", "Recycling bitmap");
            }
            Log.d("RandomImageWidget", "Recycling bitmap");
        }
        imageCache.clear();
    }


    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[]
appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);



        // Инициализация списка изображений из галереи
        if (imageUris.isEmpty()) {
            fetchImagesFromGallery(context);
        }

        // Обновляем виджеты
        for (int appWidgetId : appWidgetIds) {
            updateAppWidget(context, appWidgetManager, appWidgetId);
        }

        // Настроить AlarmManager для периодического обновления
        setAlarmToUpdateWidget(context);
    }
```

```
    private void fetchImagesFromGallery(Context context) {
        ContentResolver contentResolver = context.getContentResolver();
        Uri externalUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;

        // Запрос для получения всех изображений
        String[] projection = {MediaStore.Images.Media._ID, MediaStore.Images.Media.DATA};
        Cursor cursor = contentResolver.query(externalUri, projection, null, null, null);

        if (cursor != null) {
            while (cursor.moveToNext()) {
                int columnIndex = cursor.getColumnIndex(MediaStore.Images.Media._ID);
                long imageId = cursor.getLong(columnIndex);

                // Формируем URI для доступа к файлу изображения через контент-провайдер
                Uri imageUri =
ContentUris.withAppendedId(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
imageId);
                imageUris.add(imageUri);
                Log.d("RandomImageWidget", "Image found: " + imageUri.toString()); // Логируем
найденные изображения
            }
            cursor.close();
        }
    }

    private void updateAppWidget(Context context, AppWidgetManager appWidgetManager, int
appWidgetId) {
        RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.widget_layout);

        // Случайное изображение
        if (!imageUris.isEmpty()) {
            Random random = new Random();
            Uri randomImageUri = imageUris.get(random.nextInt(imageUris.size()));

            // Попробуйте преобразовать Uri в Bitmap
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        clearImageCache();
                        // Загружаем изображение с уменьшением размера
                        Bitmap bitmap = decodeSampledBitmapFromUri(randomImageUri, context, 125,
125); // Уменьшаем до 200x200
                        if (bitmap != null) {
                            views.setImageViewBitmap(R.id.widget_image, bitmap);
                        } else {
                            // Если изображение слишком большое, устанавливаем заглушку
                            views.setImageViewResource(R.id.widget_image,
```

```
R.drawable.placeholder_image);
                    }
                } catch (IOException e) {
                    Log.e("RandomImageWidget", "Error loading image", e);
                }
                appWidgetManager.updateAppWidget(appWidgetId, views);
            }
        }).start();
    }

    // Обновление по нажатию
    Intent intentUpdate = new Intent(context, RandomImageWidget.class);
    intentUpdate.setAction(ACTION_UPDATE_WIDGET);
    PendingIntent pendingUpdate = PendingIntent.getBroadcast(context, 0, intentUpdate,
PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);
    views.setOnClickPendingIntent(R.id.widget_refresh, pendingUpdate);

    clearImageCache();
    // Обновляем виджет
    appWidgetManager.updateAppWidget(appWidgetId, views);
  }


  private void setAlarmToUpdateWidget(Context context) {
    AlarmManager alarmManager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
    Intent intent = new Intent(context, RandomImageWidget.class);
    intent.setAction(ACTION_UPDATE_WIDGET);

    PendingIntent pendingIntent = PendingIntent.getBroadcast(context, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);

    // Используем setExact для точного времени срабатывания
    long triggerAtMillis = SystemClock.elapsedRealtime() + UPDATE_INTERVAL;
    alarmManager.setExact(AlarmManager.ELAPSED_REALTIME, triggerAtMillis,
pendingIntent);
  }

  private Bitmap decodeSampledBitmapFromUri(Uri uri, Context context, int reqWidth, int
reqHeight) throws IOException {
    // Сначала получаем размеры изображения
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inJustDecodeBounds = true;
    InputStream inputStream = context.getContentResolver().openInputStream(uri);
    BitmapFactory.decodeStream(inputStream, null, options);
    inputStream.close();

    // Вычисляем коэффициент масштабирования
    options.inSampleSize = calculateInSampleSize(options, reqWidth, reqHeight);
    options.inJustDecodeBounds = false;

    // Декодируем изображение с учетом выбранного масштаба
```

```
        inputStream = context.getContentResolver().openInputStream(uri);
        Bitmap bitmap = BitmapFactory.decodeStream(inputStream, null, options);
        inputStream.close();

        // Проверяем размер изображения и уменьшаем его, если необходимо
        if (bitmap != null && (bitmap.getByteCount() > 2000000)) {  // 2 MB лимит для виджета
            bitmap.recycle();  // Убираем старое изображение из памяти
            return null; // Возвращаем null, если изображение слишком большое
        }
        return bitmap;
    }


    // Вычисляем коэффициент масштабирования
    private int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int
reqHeight) {
        // Ширина и высота изображения
        final int height = options.outHeight;
        final int width = options.outWidth;
        int inSampleSize = 1;

        if (height > reqHeight || width > reqWidth) {
            final int halfHeight = height / 2;
            final int halfWidth = width / 2;

            // Пока одно из измерений больше нужного размера
            while ((halfHeight / inSampleSize) >= reqHeight && (halfWidth / inSampleSize) >=
reqWidth) {
                inSampleSize *= 2;
            }
        }
        return inSampleSize;
    }


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

        Log.d("RandomImageWidget", "onReceive called with action: " + intent.getAction());
        clearImageCache();
        if (ACTION_UPDATE_WIDGET.equals(intent.getAction())) {
            AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
            int[] appWidgetIds = appWidgetManager.getAppWidgetIds(new
ComponentName(context, RandomImageWidget.class));
            onUpdate(context, appWidgetManager, appWidgetIds);
        }
    }

}
```

MyApplication.java:

```
package com.example.android_dev_lab6;
import android.app.Application;
import android.util.Log;

public class MyApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    public void onTrimMemory(int level) {
        super.onTrimMemory(level);

        // Логирование уровня памяти
        Log.d("MyApplication", "Memory trim level: " + level);

        if (level >= TRIM_MEMORY_UI_HIDDEN) {
            // Очистка кэша изображений
            Log.d("MyApplication", "Clearing image cache due to low memory.");
            RandomImageWidget.clearImageCache(); // Вызов метода очистки кэша
        }
    }
}
```

MainActivity.java:

```
package com.example.android_dev_lab6;
import android.Manifest;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private static final int PERMISSION_REQUEST_CODE = 100;
```

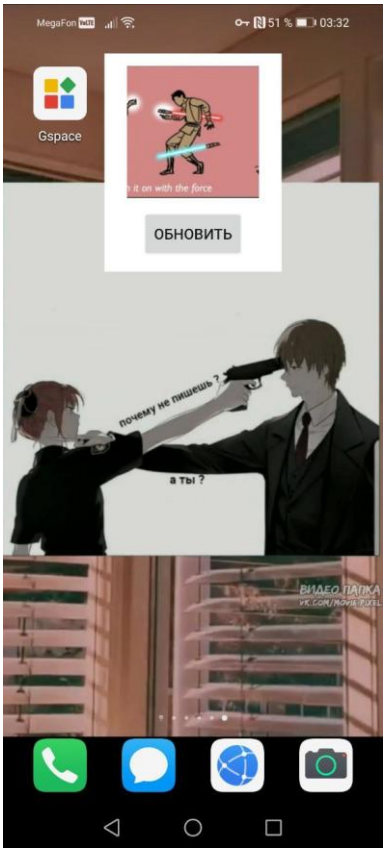
```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
//        clearImageCache();


        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
                    PERMISSION_REQUEST_CODE);
        } else {
            // Если разрешение уже предоставлено, можно продолжать работу
            Toast.makeText(this, "Permission granted!", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);

        if (requestCode == PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "Permission granted!", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "Permission denied!", Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

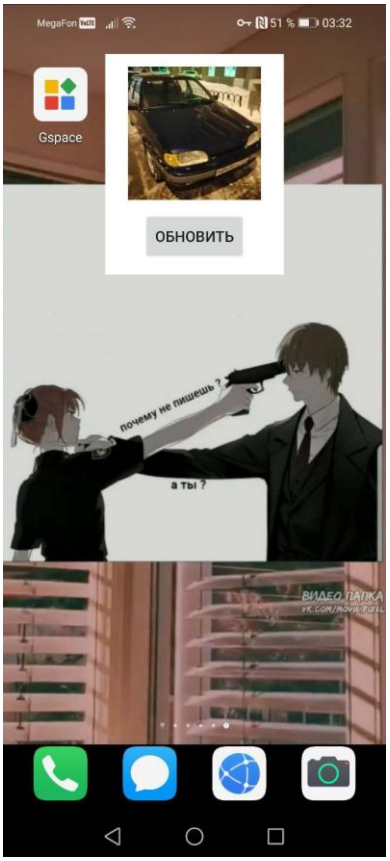
AndroidManifest.xml:

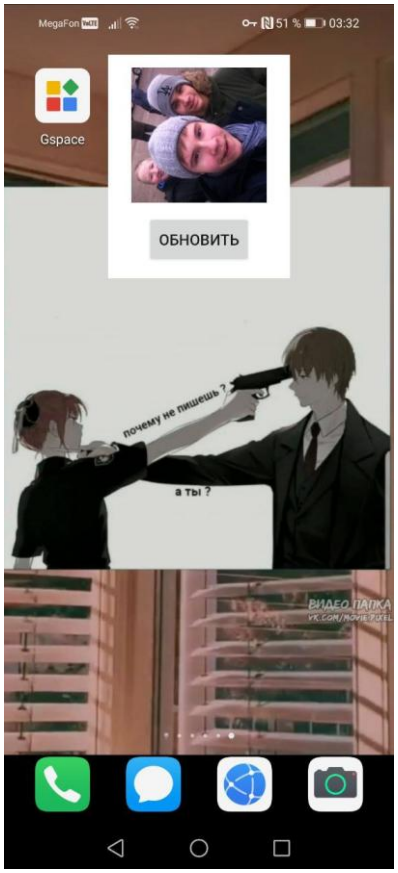
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:name=".MyApplication"
        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:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
```

```
        android:theme="@style/Theme.Android_dev_lab6"
        tools:targetApi="31">

        <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>


        <receiver android:name=".RandomImageWidget"
            android:exported="true"
            tools:ignore="WrongManifestParent">
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>
            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/widget_info" />
        </receiver>




    </application>



    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />


</manifest>
```

Результаты выполнения работы:







Вывод: в ходе лабораторной работы был реализован виджет отображающий случайные картинки из галереи пользователя и предусматривающий принудительное обновление.