**Практическая работа**

**Тема: кастомные виджеты на Android**

Теоретический материал

Создание кастомных виджетов на Android позволяет пользователям персонализировать свои устройства и улучшить их функциональность. Виджеты могут отображать информацию, предоставлять быстрый доступ к приложениям и выполнять различные задачи прямо с домашнего экрана. В этой статье мы рассмотрим, как создать кастомные виджеты на Android, начиная с основ и заканчивая добавлением интерактивности и тестированием. Мы разберем каждый шаг процесса, чтобы вы могли легко следовать инструкциям и создать свой собственный виджет.

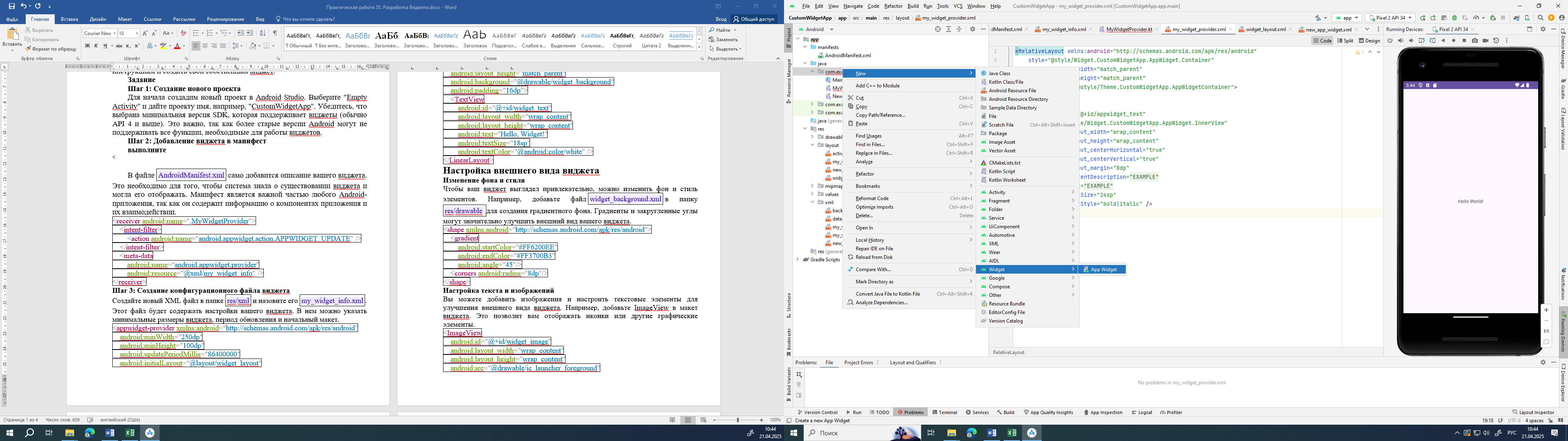
**Задание**

**Шаг 1: Создание нового проекта**

Для начала создадим новый проект в Android Studio. Выберите **"Empty Activity"** и дайте проекту имя, например, "CustomWidgetApp". Убедитесь, что выбрана минимальная версия SDK, которая поддерживает виджеты (обычно API 35). Это важно, так как более старые версии Android могут не поддерживать все функции, необходимые для работы виджетов.

**Шаг 2: Добавление виджета в манифест**

**выполните**



В файле AndroidManifest.xml само добавится описание вашего виджета. Это необходимо для того, чтобы система знала о существовании виджета и могла его отображать. Манифест является важной частью любого Android-приложения, так как он содержит информацию о компонентах приложения и их взаимодействии.

<receiver  
 android:name=".MyWidgetProvider"  
 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/my\_widget\_provider\_info" />  
</receiver>

**Шаг 3: Создание конфигурационного файла виджета**

В папке res/xml появится файлmy\_widget\_info.xml. Этот файл будет содержать настройки вашего виджета. В нем можно указать минимальные размеры виджета, период обновления и начальный макет.

<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"

android:minWidth="250dp"

android:minHeight="100dp"

android:updatePeriodMillis="86400000"

android:initialLayout="@layout/widget\_layout"

android:resizeMode="horizontal|vertical"

android:widgetCategory="home\_screen">

**Шаг 4: Создание макета виджета**

В папке res/layout создайте новый XML файл и назовите его widget\_layout.xml. Это будет основной макет вашего виджета. Макет определяет, как виджет будет выглядеть на экране.

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 style="@style/Widget.Widget.AppWidget.Container"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:theme="@style/Theme.Widget.AppWidgetContainer">  
  
 <TextView  
 android:id="@+id/appwidget\_text"  
 style="@style/Widget.Widget.AppWidget.InnerView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_centerVertical="true"  
 android:layout\_margin="8dp"  
 android:contentDescription="@string/appwidget\_text"  
 android:text="@string/appwidget\_text"  
 android:textSize="24sp"  
 android:textStyle="bold|italic" />  
</RelativeLayout>

**Настройка внешнего вида виджета**

**Изменение фона и стиля**

Чтобы ваш виджет выглядел привлекательно, можно изменить фон и стиль элементов. Например, добавьте файл widget\_background.xml в папку res/drawable для создания градиентного фона. Градиенты и закругленные углы могут значительно улучшить внешний вид вашего виджета.

<shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle">  
 <corners android:radius="?attr/appWidgetInnerRadius" />  
 <solid android:color="@color/light\_blue\_200" /><gradient  
 android:startColor="#2196F3"  
 android:endColor="#FF3700B3"  
 android:angle="45"/>  
 <corners android:radius="8dp"/>  
  
  
</shape>

**Настройка текста и изображений**

Вы можете добавить изображения и настроить текстовые элементы для улучшения внешнего вида виджета. Например, добавьте ImageView в макет виджета. Это позволит вам отображать иконки или другие графические элементы.

<ImageView  
 android:id="@+id/widget\_image"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:src="@drawable/ic\_launcher\_foreground"  
 app:srcCompat="@android:drawable/btn\_star\_big\_on" />

**Добавление интерактивности и функциональности**

**Обработка кликов**

Для добавления интерактивности виджету необходимо настроить обработку кликов. В классе MyWidgetProvider добавьте код для обработки нажатий. Это позволит пользователям взаимодействовать с виджетом, например, открывать приложения или обновлять данные.

override fun onUpdate(  
 context: Context,  
 appWidgetManager: AppWidgetManager,  
 appWidgetIds: IntArray  
) {  
 *// There may be multiple widgets active, so update all of them* for (appWidgetId in appWidgetIds) {  
 val intent = Intent(context, MyWidgetProvider::class.*java*)  
 intent.*action* = AppWidgetManager.*ACTION\_APPWIDGET\_UPDATE* val pendingIntent =  
 PendingIntent.getBroadcast(context, 0, intent, PendingIntent.*FLAG\_UPDATE\_CURRENT*)  
 val views = RemoteViews(context.*packageName*, R.layout.*my\_widget\_provider*)  
 views.setOnClickPendingIntent(R.id.*appwidget\_text*, pendingIntent)  
 appWidgetManager.updateAppWidget(appWidgetId, views)  
 }  
}

**Обновление данных виджета**

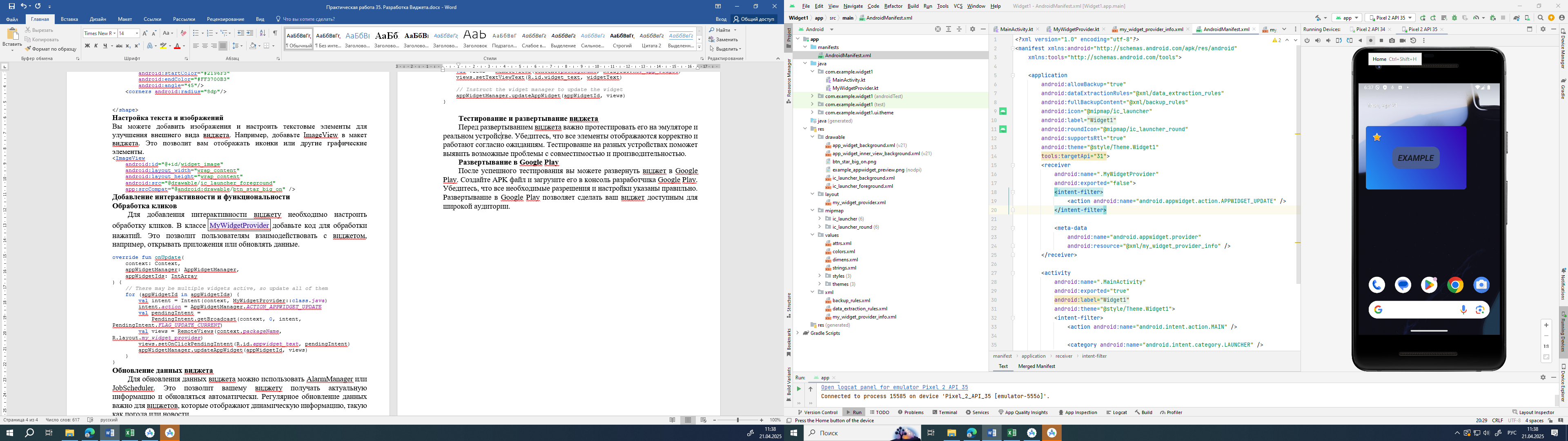
Для обновления данных виджета можно использовать AlarmManager или JobScheduler. Это позволит вашему виджету получать актуальную информацию и обновляться автоматически. Регулярное обновление данных важно для виджетов, которые отображают динамическую информацию, такую как погода или новости.

internal fun updateAppWidget(  
 context: Context,  
 appWidgetManager: AppWidgetManager,  
 appWidgetId: Int  
) {  
 val widgetText = context.getString(R.string.*appwidget\_text*)  
 *// Construct the RemoteViews object* val views = RemoteViews(context.*packageName*, R.layout.*new\_app\_widget*)  
 views.setTextViewText(R.id.widget\_text, widgetText)  
  
 *// Instruct the widget manager to update the widget* appWidgetManager.updateAppWidget(appWidgetId, views)  
}

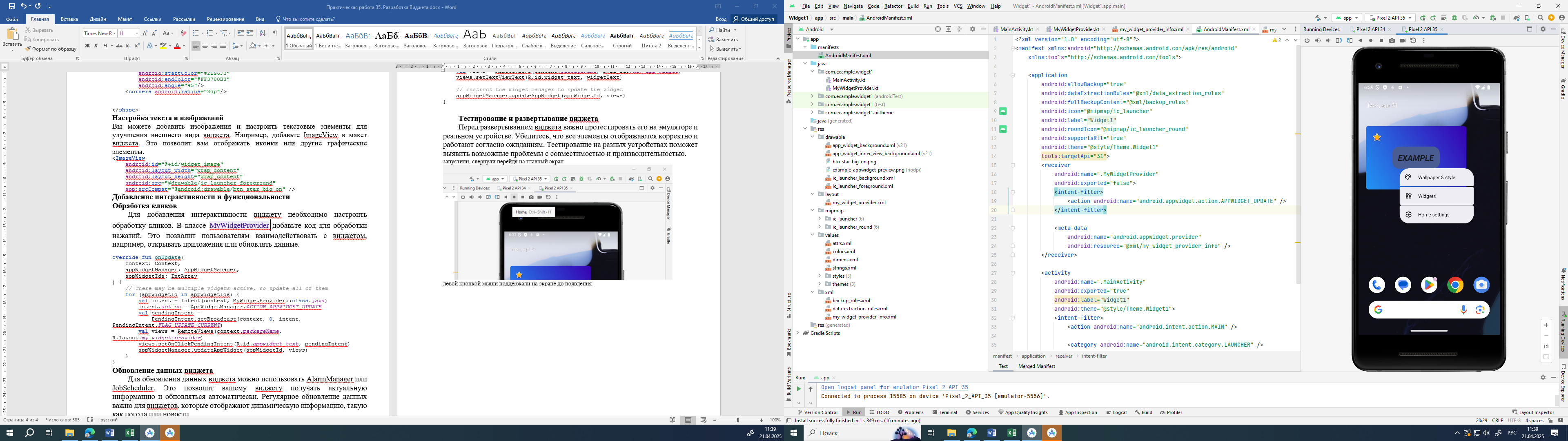
**Тестирование и развертывание виджета**

Перед развертыванием виджета важно протестировать его на эмуляторе и реальном устройстве. Убедитесь, что все элементы отображаются корректно и работают согласно ожиданиям. Тестирование на разных устройствах поможет выявить возможные проблемы с совместимостью и производительностью.

запустили, свернули перейдя на главный экран



левой кнопкой мыши поддержали на экране до появления



перешли в панель виджетов, выбрали свой и добавили его кнопкой ADD