



x @ mail.ru
group

Android и библиотеки

Клещин Никита





Напоминание отметиться на портале



Кратко

#03

Что помним?

- Как находить проблемы
- Как их решаем?:)

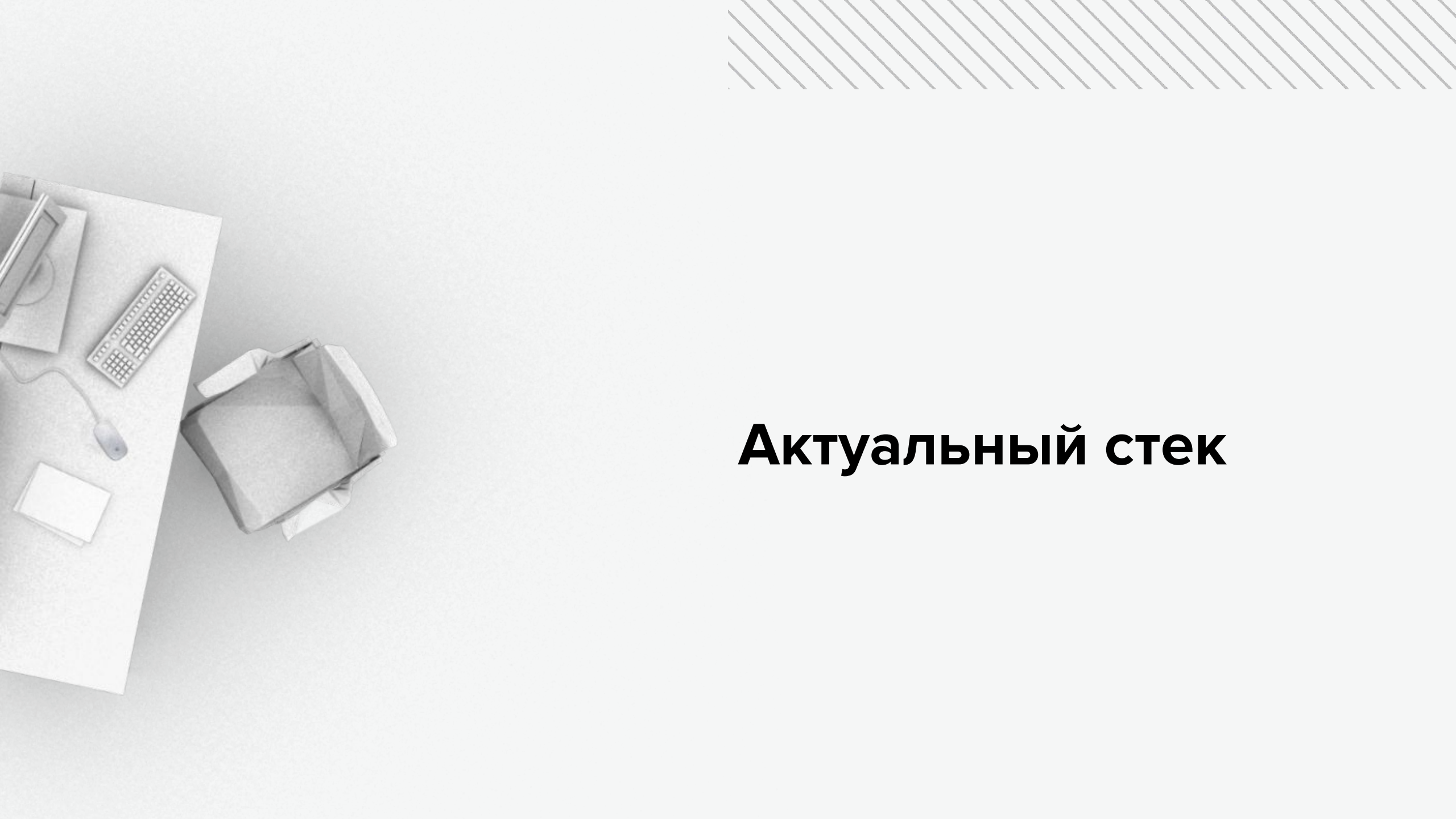
Что успели сделать?

- MVP1?
- MVP2? :)



Содержание занятия

1. Актуальный стек
2. Новый Android
3. Не такой новый Android



Актуальный стек

У разных команд - разный стек

Требования

- ...
- Умение оценивать сложность и время выполнения задач, git;
- Уверенное знание Kotlin, Java;
- Опыт разработки клиент-сервер приложений, работа с сетью, работа в фоне;
- Опыт создания кастомных UI компонентов.

Будет плюсом

Знание JavaRX, OKHTTP, Android Architecture Components, Coroutines

Требования

- ...
- опыт работы с Rx;
- уверенное знание kotlin и Java;

Требования

- Опыт профессионального программирования на языке Java;
- Опыт профессиональной разработки Android-приложений;
- Опыт работы с web-технологиями;

Требования

- Знание Kotlin + Coroutines&Flow
- Знание и опыт применения clean architecture и design patterns
- Знание и опыт применения инструментов командной работы: git / CI / CD

Требования

- Знание Java, Kotlin. Опыт коммерческого программирования минимум 3 года;
- Знание SOLID принципов и MV* паттернов (MVP, MVVM и другие);
- Понимание Single Activity Architecture, Multi-module project;
- Приветствуется опыт работы с медиаконтентом - фото, видео, музыка;

Все равно есть пересечения

- Когда написано про взаимодействие клиент-сервер, обычно:
 - **OkHttpClient**
 - **retrofit2** + **OkHttpClient**
 - **GSON**
 - **moshi**
- Когда говорят про многопоточность:
 - В редких случаях - знание как работать с **java.util.concurrent**, и как многопоточность обеспечивается в java
 - **RxJava**
 - **Coroutines**
- Видео и Аудио - **ExoPlayer**
- Картинки:
 - **Glide**
 - **Picasso**
 - Реже - **fresco**

... продолжение

- Android Architecture Components:
 - **ViewModel**
 - **LiveData**
 - **Room**
 - **Paging** (Используют с недоверием:)
 - **ViewBinding** (не всегда)
 - **Navigation** (не всегда)
 - ... тут еще много полезного для ознакомления
- Инверсия зависимостей (DI):
 - **Dagger2**
 - **KOIN**
 - **ToothPick**
- База данных
 - **Room**
 - **Realm**
 - Остаются еще ORM варианты или NoSql

Нет ничего вечного

Поскольку появляются новые технологии и идеи, то и “стандартные библиотеки” со временем тоже меняются.

Для примера

- Асинхронное взаимодействие:
 - **AsyncTask/Executors** -> **RoboSpice/Volley** -> **RxJava/Coroutines**
- Загрузка изображений:
 - Свои разработки -> **UniversalImageLoader/Picasso** -> **Glide**
- База данных:
 - **SQLite** -> **ORM (greenDAO)** -> **Realm** -> **Room** + возможны старые реализации

ViewBinding

app level build.gradle

```
android {  
    ...  
  
    buildFeatures {  
        viewBinding true  
    }  
}
```

```
class DashboardFragment : Fragment() {  
    private var binding: FragmentDashboardBinding? = null  
    private val binding get() = _binding!!  
  
    ...  
  
    override fun onCreateView(inflater: LayoutInflater  
        , container: ViewGroup?, savedInstanceState: Bundle?  
    ): View? {  
        _binding = FragmentDashboardBinding.inflate(inflater, container, false)  
  
        return _binding.root  
    }  
  
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
        super.onViewCreated(view, savedInstanceState)  
  
        binding.textDashboard.setOnClickListener { ... }  
    }  
  
    override fun onDestroyView() {  
        super.onDestroyView()  
        _binding = null  
    }  
}
```

Navigation Component installation

top level build.gradle

```
buildscript {  
    ...  
    dependencies {  
        ...  
        classpath  
        "androidx.navigation:navigation-safe-args-gradle-plugin: $nav_version"  
    }  
}
```

app level build.gradle

```
plugins {  
    id 'androidx.navigation.safeargs.kotlin'  
}  
  
dependencies {  
    ...  
    // navigation  
    implementation "androidx.navigation:navigation-fragment-ktx: $nav_version"  
    implementation "androidx.navigation:navigation-ui-ktx: $nav_version"  
}
```

Navigation Component usage

layout

```
<fragment
    android:id="@+id/nav_host_fragment_activity_main"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"

    app:defaultNavHost="true"
    app:navGraph="@navigation/mobile_navigation"
/>
```

navigate

```
binding.textDashboard.setOnClickListener {
    val direction = DashboardFragmentDirections

    .actionNavigationDashboardToFullscreenFragment( "BlahBla")
        findNavController().navigate(direction)
}
```

or

```
binding.textDashboard.setOnClickListener {
    val args = FullscreenFragmentArgs( "BlahBlah")
        .toBundle()
    findNavController().navigate(R.id.fullscreenFragment, args)
}
```

Paging Library (v3)

app level build.gradle

```
dependencies {  
    ...  
    implementation "androidx.paging:paging-runtime: $paging_version"  
}
```

objects

```
data class Dummy(val id: Int, val title: String)  
class DummyComparator: DiffUtil.ItemCallback<Dummy>() {...}  
class DummyHolder(val binding: ItemDummyBinding): RecyclerView.ViewHolder(binding.root)  
{...}  
class DummyAdapter: PagingDataAdapter<Dummy, DummyHolder>(DummyComparator()) {...}
```

observing

```
val dummies = Pager(PagingConfig(pageSize = 20, prefetchDistance = 10))  
    { DummyProvider(DummyAccessor()) }  
    .flow.cachedIn(viewModelScope)  
  
viewLifecycleOwner.lifecycleScope.launch {  
    notificationsViewModel.dummies.collectLatest {  
        dummyAdapter.submitData(it)  
    }  
}
```

Room

app level build.gradle

```
plugins {  
    ...  
    id 'kotlin-kapt'  
}  
...  
dependencies {  
    ...  
    implementation "androidx.room:room-runtime: $room_version"  
    implementation "androidx.room:room-ktx: $room_version"  
    kapt "androidx.room:room-compiler: $room_version"  
}
```

initialization

```
database = Room.databaseBuilder(context, AbstractDatabase::class.java, "database-name").build()
```

objects

```
@Entity  
data class Dummy(  
    @PrimaryKey val id: Int  
    , val title: String  
)  
  
@Dao  
interface IDummyOfflineAccessor {  
    @Query("SELECT * FROM dummy")  
    suspend fun getAll(): List<Dummy>?  
  
    @Query("SELECT * FROM dummy WHERE id IN (:ids)" )  
    suspend fun load(ids: IntArray): List<Dummy>?  
  
    @Insert suspend fun insertAll(list: List<Dummy>)  
    @Delete suspend fun delete(item: Dummy)  
}  
  
@Database(entities = arrayOf(Dummy::class), version = 1)  
abstract class AbstractDatabase: RoomDatabase() {  
    abstract fun dummyOfflineAccessor():  
        IDummyOfflineAccessor  
}
```

Simple Image Loading

app level build.gradle

```
plugins {  
    id 'kotlin-kapt'  
}  
...  
dependencies {  
    ...  
    implementation "com.github.bumptech.glide:glide: $glide_version"  
    kapt "com.github.bumptech.glide:compiler: $glide_version"  
}
```

usage in code

```
Glide.with(context / activity / fragment)  
    .load(url)  
    .into(imageView)
```

app level build.gradle

```
dependencies {  
    ...  
    implementation 'com.squareup.picasso:picasso:2.71828'  
}
```

usage in code

```
Picasso.get()  
    .load(url)  
    .into(imageView)
```

Dependency Injection Framework install

top level build.gradle

```
buildscript {  
    dependencies {  
        classpath  
        "com.google.dagger:hilt-android-gradle-plugin: $hilt_version"  
    }  
}
```

app level build.gradle

```
plugins {  
    ...  
    id 'dagger.hilt.android.plugin'  
}  
  
dependencies {  
    ...  
    implementation "com.google.dagger:hilt-android: $hilt_version"  
    kapt "com.google.dagger:hilt-android-compiler: $hilt_version"  
}
```


Dependency Injection Framework usage

```
@Module
@InstallIn (SingletonComponent :: class)
class DummyModule {
    @Provides @Singleton
    fun provideDummyProvider (onlineAccessor: IDummyAccessor, offlineAccessor: IDummyOfflineAccessor): DummyProvider
    {}

    @Provides @Singleton
    fun provideDummyOnlineAccessor (): IDummyAccessor {}

    @Provides @Singleton
    fun provideDatabase (@ApplicationContext context: Context): AbstractDatabase {}

    @Provides @Singleton
    fun provideDummyOfflineAccessor (database: AbstractDatabase): IDummyOfflineAccessor {}
}

@HiltAndroidApp
class DummyApplication: Application()

@AndroidEntryPoint
class MainActivity : AppCompatActivity() {...}

@AndroidEntryPoint
class NotificationsFragment : Fragment() {...}

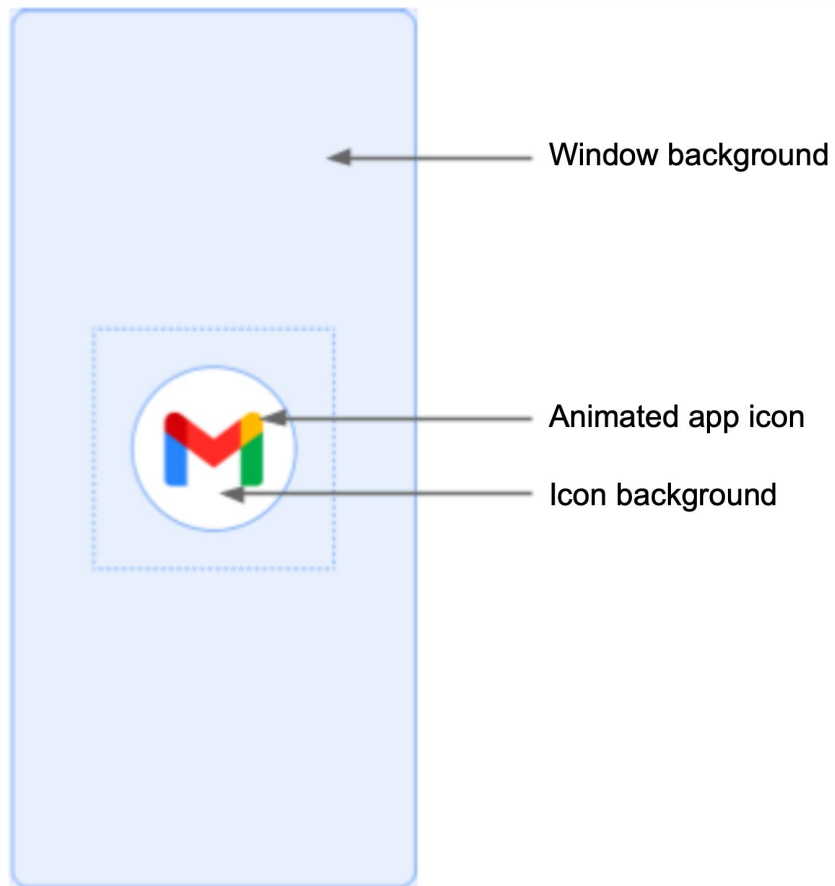
@HiltViewModel
class NotificationsViewModel @Inject constructor(): ViewModel() {
    @Inject lateinit var dummyProvider: DummyProvider

    ...
}
```

Hello Android 12



SplashScreen API



Кастомизация стартового экрана.

Главное изменение - это добавления анимации центральной иконки

```
<style name="Theme.Base" parent="Theme.MaterialComponents.DayNight.DarkActionBar" >
  <item name="android:windowSplashScreenBackground" >#ffffff</item>
  <item name="android:windowSplashScreenAnimatedIcon" >@mipmap/...</item>
  <item name="android:windowSplashScreenIconBackgroundColor" >#ffffff</item>
  <item name="android:windowSplashScreenBrandingImage" >@drawable/...</item>
</style>
```

Widgets Improvements



Закругленные углы:)

```
<style name="MyWidgetTheme" parent="@android:style/Theme.DeviceDefault.DayNight">  
  <item name="backgroundRadius">@android:dimen/system_app_widget_background_radius</item>  
</style>
```

Использование цветов темы приложения, например:

```
?android:attr/colorAccent
```

```
?android:attr/textColorPrimary
```

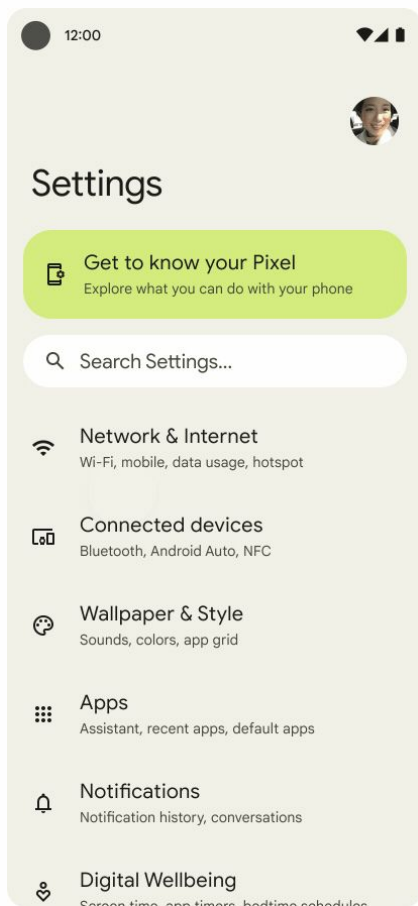
Удобный ресайзинг

```
<appwidget-provider  
  ...  
  android:configure="com.myapp.WidgetConfigActivity"  
  android:widgetFeatures="reconfigurable">  
</appwidget-provider>
```

Обновлены контроллы - чекбоксы, свитчи радиокнопки

Плавный переход в приложение из виджета

Overscroll effect



Он включается по умолчанию для ViewGroup, которые используют **EdgeEffect**.

(RecyclerView, ListView, ScrollView, NestedScrollView, HorizontalScrollView, ViewPager, ViewPager2)

Но можно отключить:

```
<ScrollView
...
    android:overScrollMode="never"
...
>
```

или в коде

```
...
recyclerview.overScrollMode = View.OVER_SCROLL_NEVER
```

Hello, Native Render Effect

Выставить RenderEffect для View

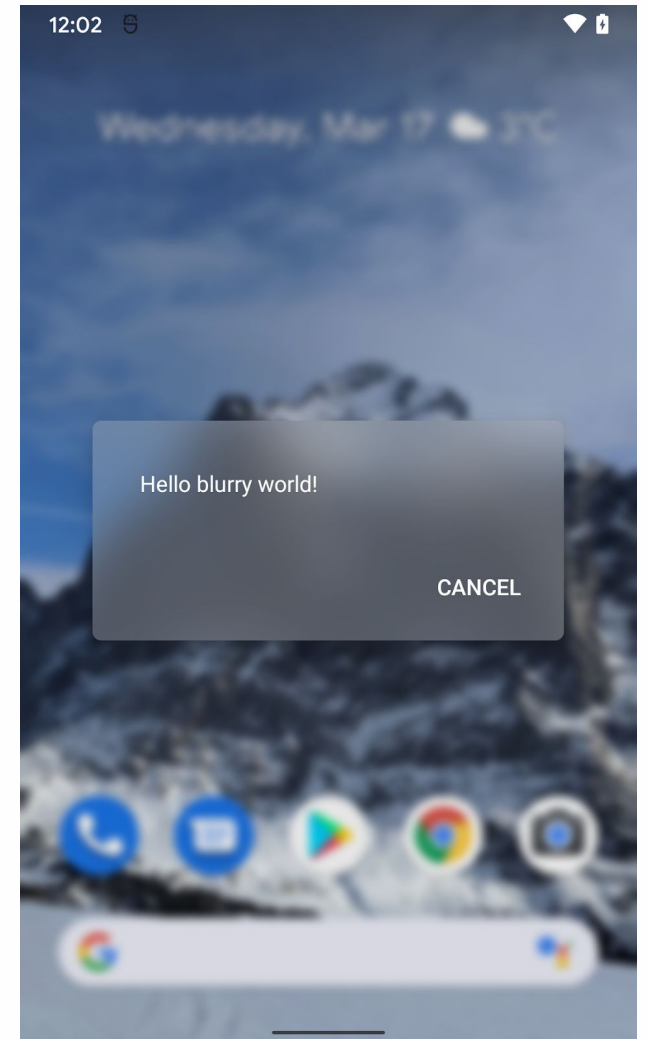
```
val RENDER_EFFECT_BLUR = RenderEffect.createBlurEffect( 20f, 20f, Shader.TileMode.CLAMP)
view.setRenderEffect( RENDER_EFFECT_BLUR )
```

Проставить blur для экрана

```
<item name="android:windowIsTranslucent">true</item>
<item name="android:windowBlurBehindEnabled" >true</item>
<item name="android:windowBlurBehindRadius" >40dp</item>
<item name="android:windowBackgroundBlurRadius" >20dp</item>
```

В коде

```
window.setBackgroundBlurRadius( 40)
```





И еще улучшения работы и приватности:

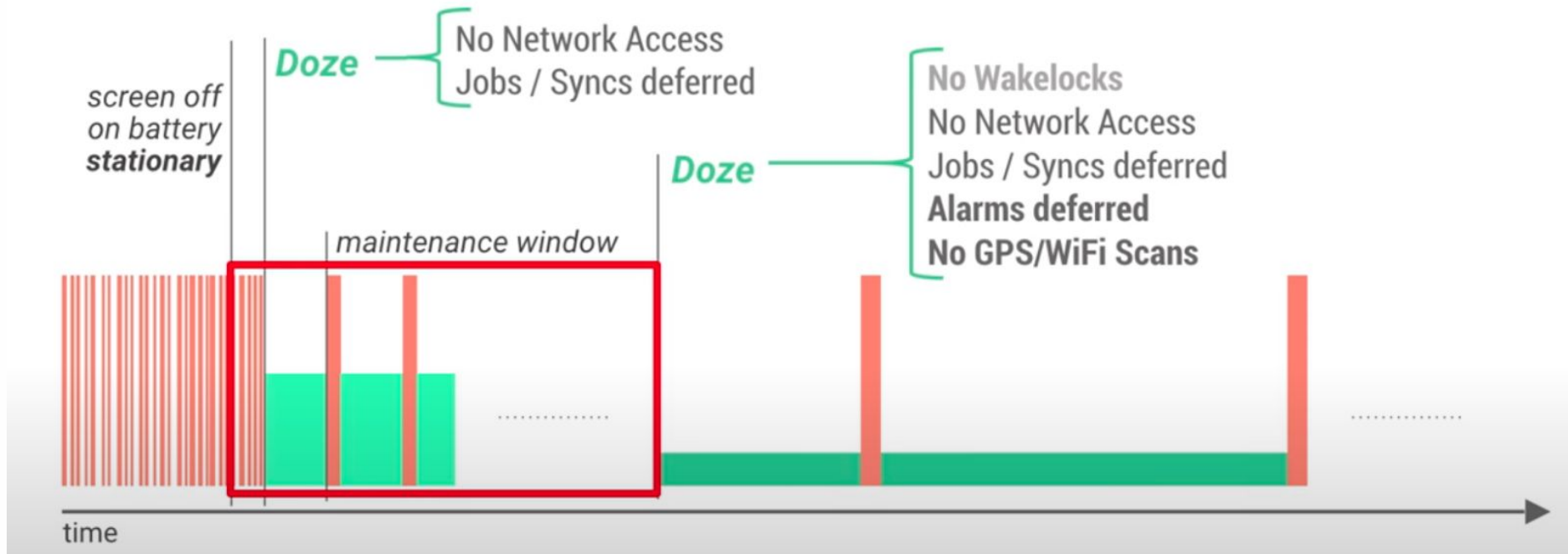
- Улучшенная производительность за счет оптимизации системы
 - Расширены возможности Picture In Picture
 - Копирование-вставка не только текста
-
- Ограничения на запрос геолокации
 - Еще ограничения на работу в фоне
 - Сброс выданных разрешений
 - Автоочистка кэша малоиспользуемых приложений
 - Дополнительные ограничения на старт Foreground Service

и прочее



pre-Android S

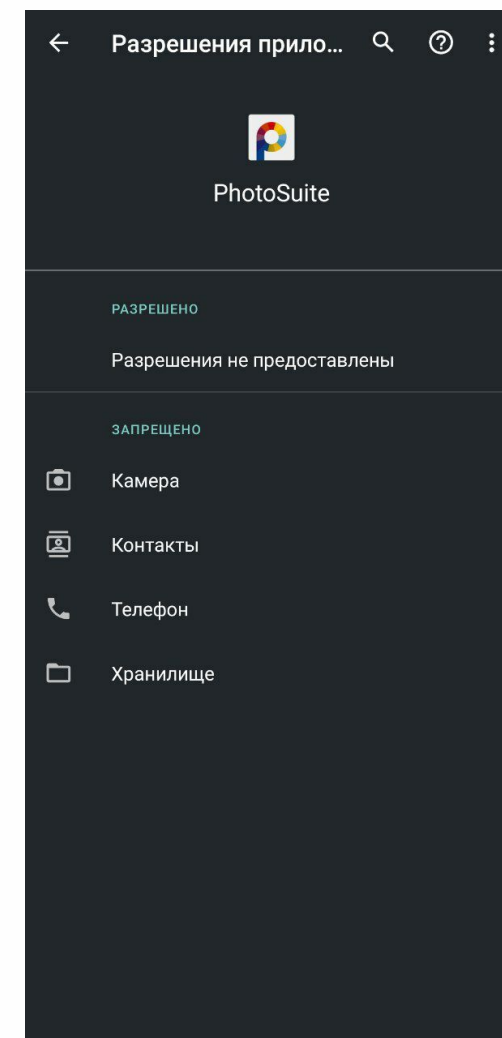
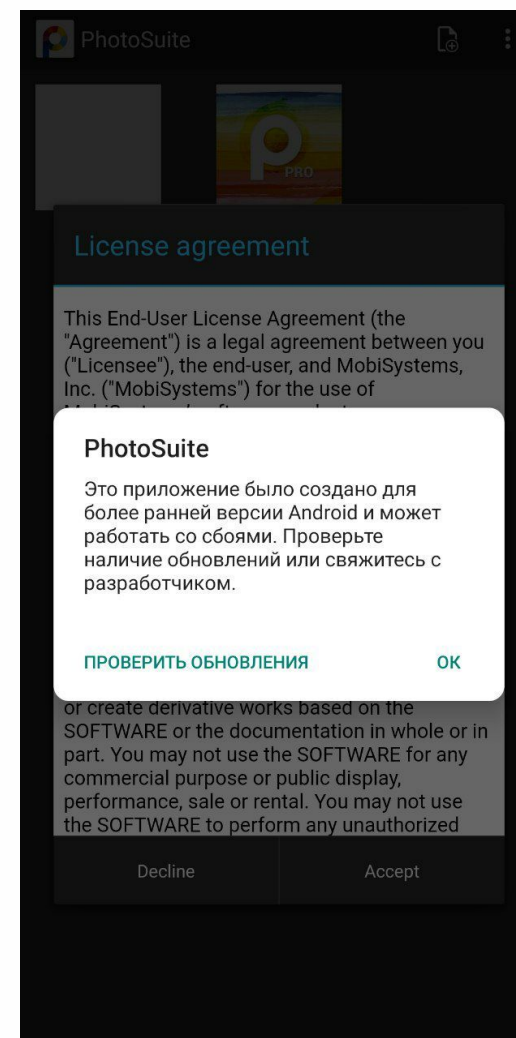
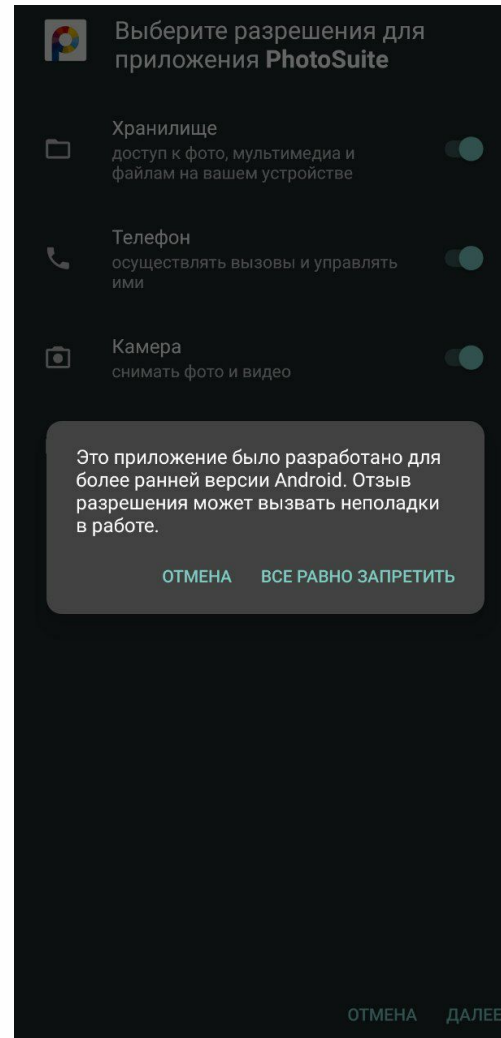
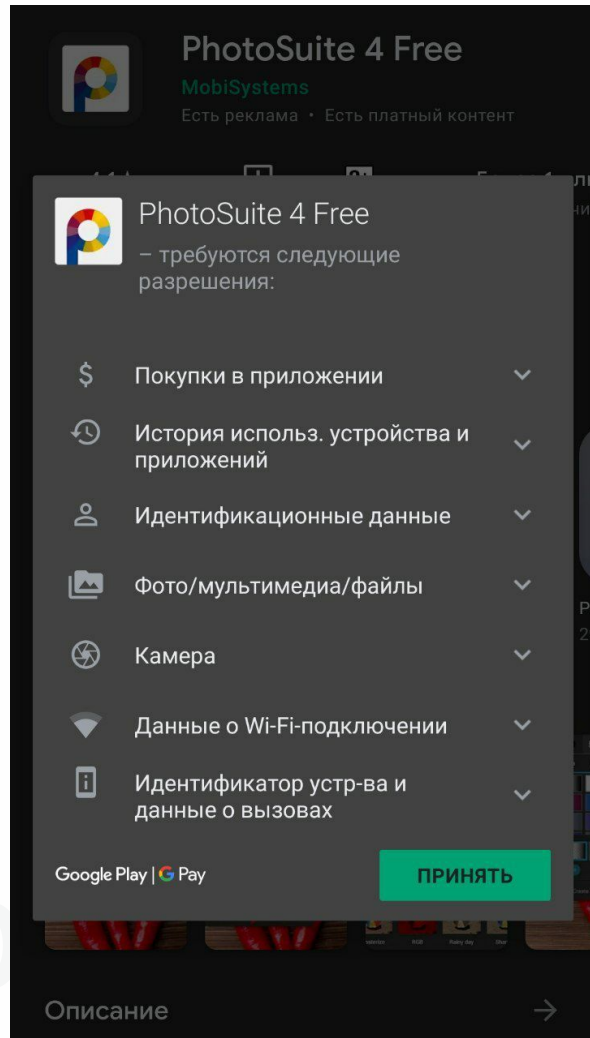
Doze and App Standby (Android API \geq 23)



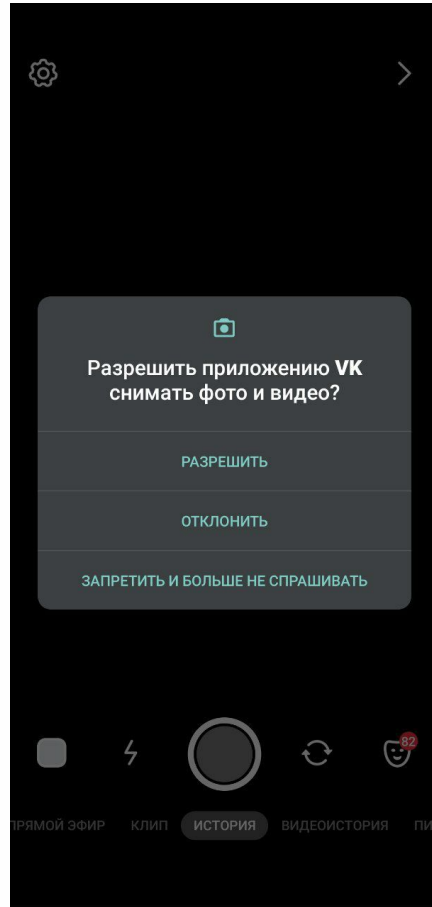
Android Service

	Запуск сервиса		Вывод на передней план
Api level	Context	ContextCompat	Service
< 26	startService(Intent i)	startForegroundService(Contex c, Intent i)	startForeground(int id, Notification n)
< 29	startForegroundService(Intent i)		startForeground(int id, Notification n)
>=29			startForeground(int id, Notification n, int serviceType)
>=31	Ограничения на старт из Background		

Uses Permissions Target Api < 23



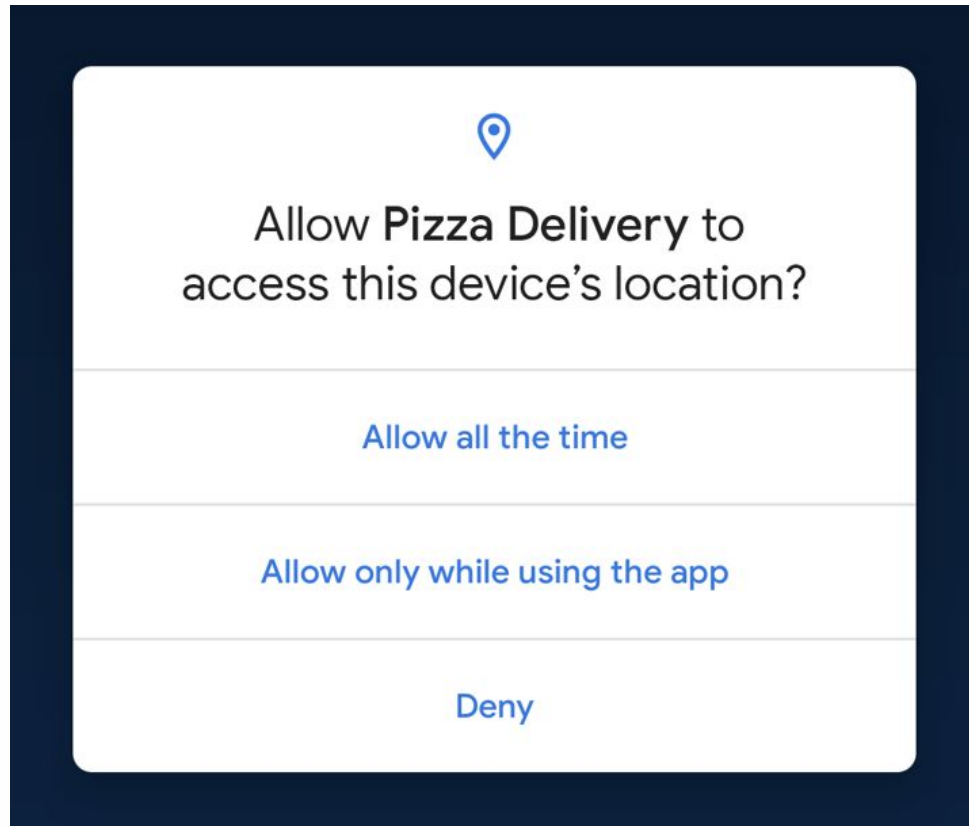
Runtime Permission Target Api >= 23



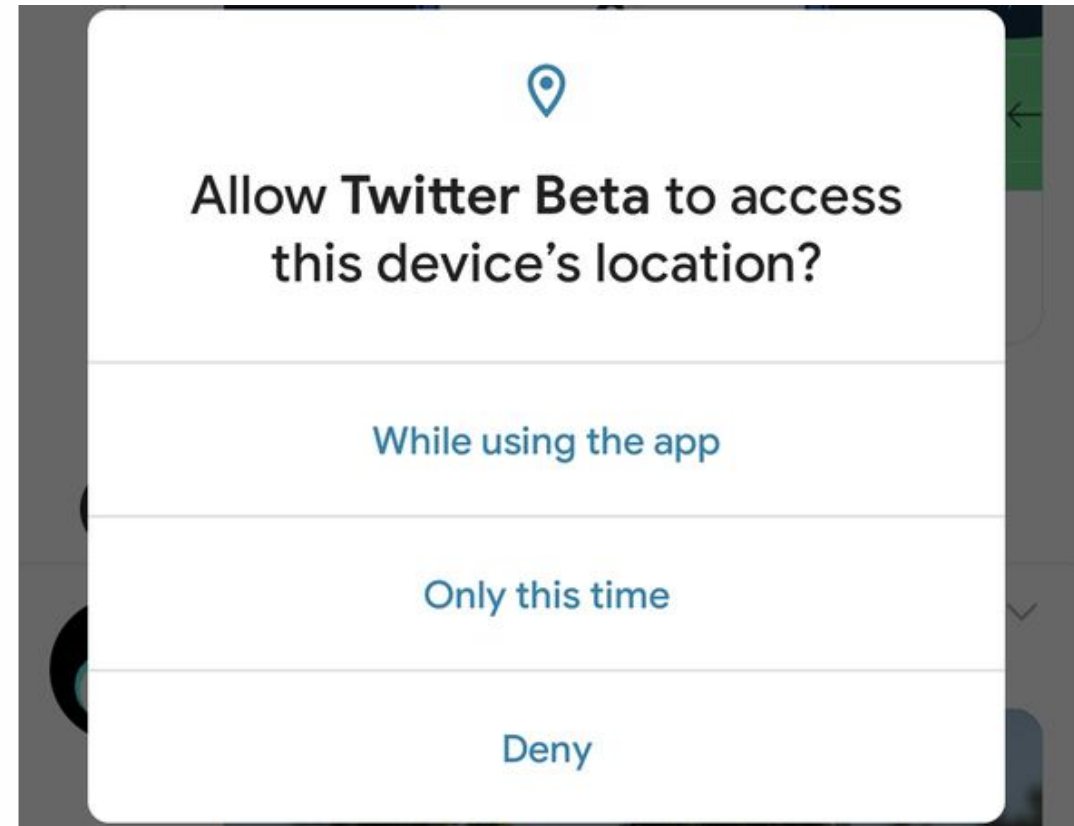
Уровни защиты

1. Normal (Интернет)
2. Dangerous (Геолокация, микрофон)
3. Signature
4. SignatureOrSystem

Ограничения



Api 29



Api 30

Ограничения. Арі 30

1. Автоматический сброс разрешений спустя некоторое время
2. Изменение логики отображения диалога
 - Если пользователь отклонил запрос более одного раза, то диалог не показывается

Использование

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="ru.hse.lesson10">
    ...
    <uses-permission android:name="android.permission.READ_CONTACTS" />

    requestPermissionLauncher = registerForActivityResult( ActivityResultContracts .RequestPermission()) { isGranted: Boolean ->
        if (isGranted) {
            loadContacts()
        } else {
            Toast.makeText(requireContext(), "PERMISSION is not granted", Toast.LENGTH_LONG).show()
            ...
        }
    }

    fun checkAndLoadContacts () {
        when {
            ContextCompat.checkSelfPermission(requireActivity(), Manifest.permission.READ_CONTACTS) == PackageManager.PERMISSION_GRANTED -> {
                loadContacts()
            }

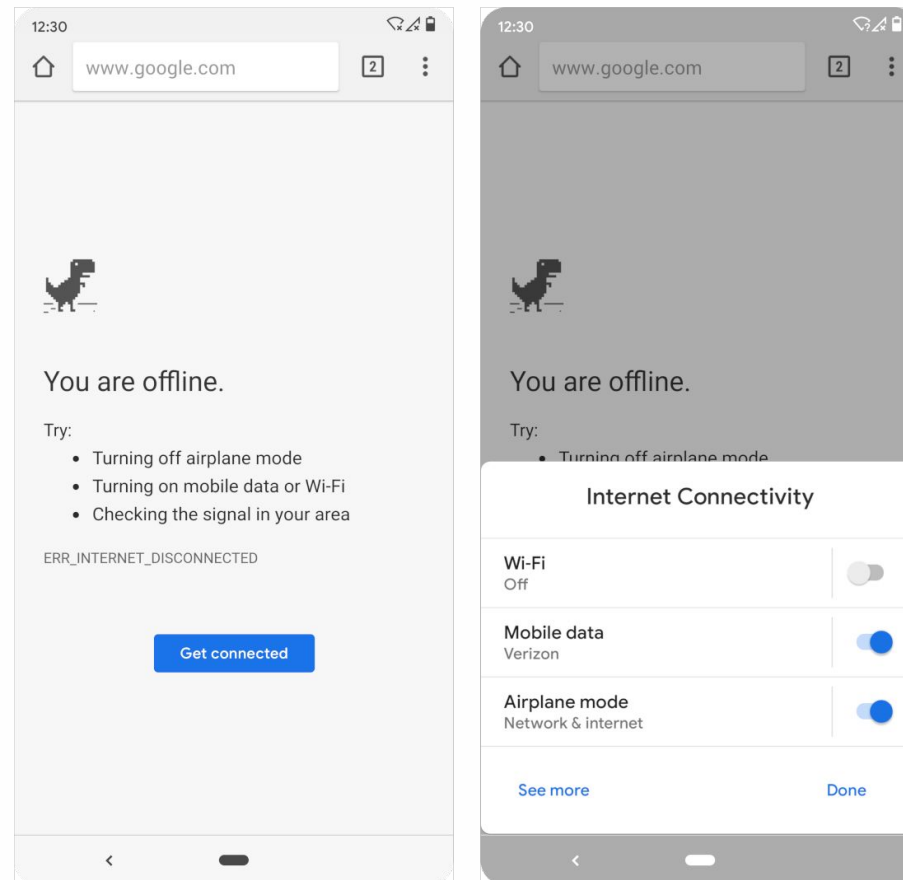
            ActivityCompat.shouldShowRequestPermissionRationale(requireActivity(), Manifest.permission.READ_CONTACTS) -> {
                showRationaleDialog()
            }

            else -> requestContactPermission()
        }
    }

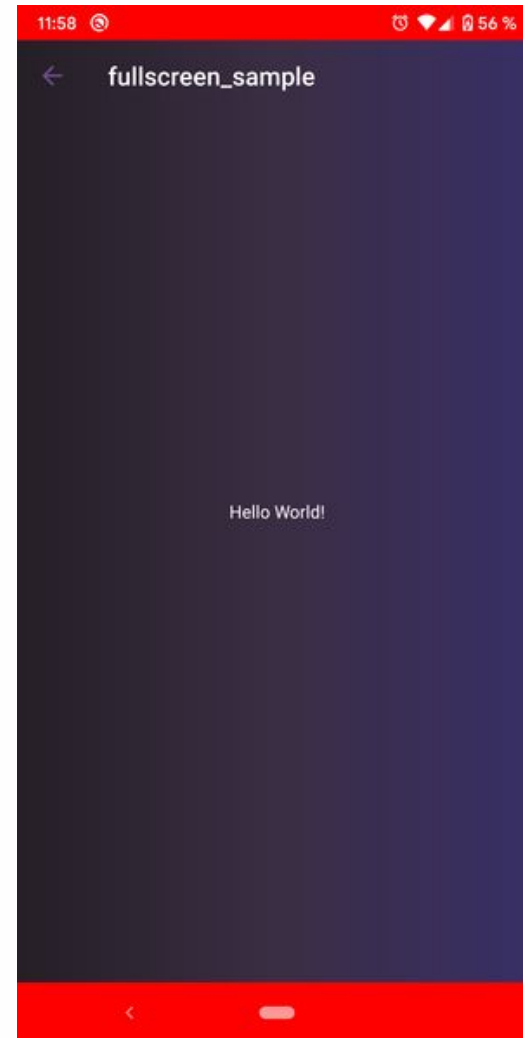
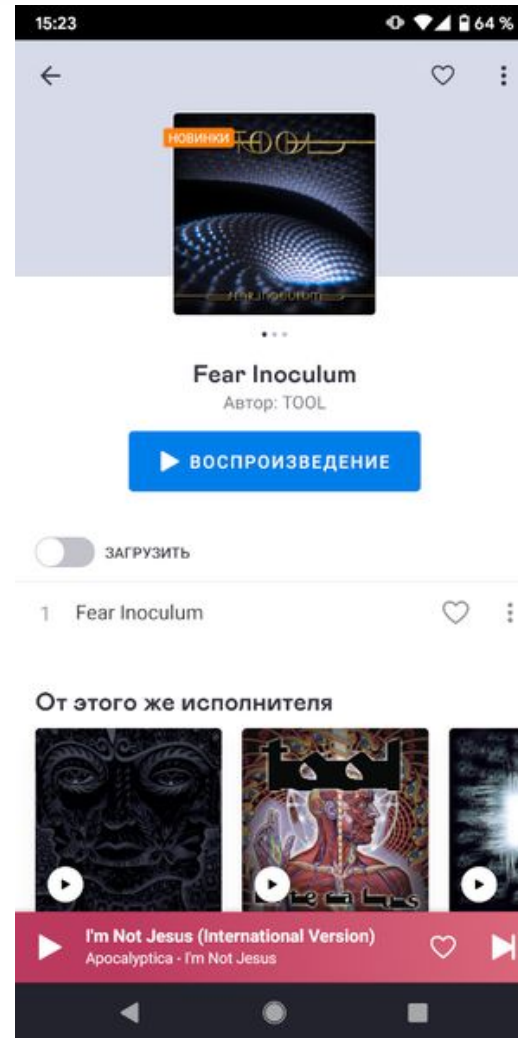
    fun requestContactPermission () {
        requestPermissionLauncher.launch(Manifest.permission.READ_CONTACTS)
    }
}
```

Setting Panel (Android API \geq 29)

```
val panelIntent = Intent(Settings.Panel.settings_panel_type)  
startActivity(panelIntent)
```




WindowInsets (Android API ≥ 21)





WindowInsets simple implementation (layout)



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

    android:fitsSystemWindows="true"
>
```

WindowInsets simple implementation (code)

```
typealias BarChangeListener = (offsetTop: Int, offsetBottom: Int) -> Unit

fun setBarTransparency(listener: BarChangeListener) {
    setCustomInsetHandler(window.decorView, listener)
    window.navigationBarColor = Color.TRANSPARENT
    window.statusBarColor = Color.TRANSPARENT
}

fun setCustomInsetHandler(view: View, listener: BarChangeListener) {
    ViewCompat.setOnApplyWindowInsetsListener(view) { , insets ->
        val desiredBottomInset = calculateDesiredBottomInset(
            insets.systemWindowInsetTop,
            insets.systemWindowInsetBottom,
            listener
        )

        val tmp = Insets.of(0, 0, 0, desiredBottomInset)
        val compatInsets = WindowInsetsCompat.Builder()
            .setSystemWindowInsets(tmp)
            .build()

        ViewCompat.onApplyWindowInsets(view, compatInsets)
    }
}
```

WindowInsets simple implementation (code) 2

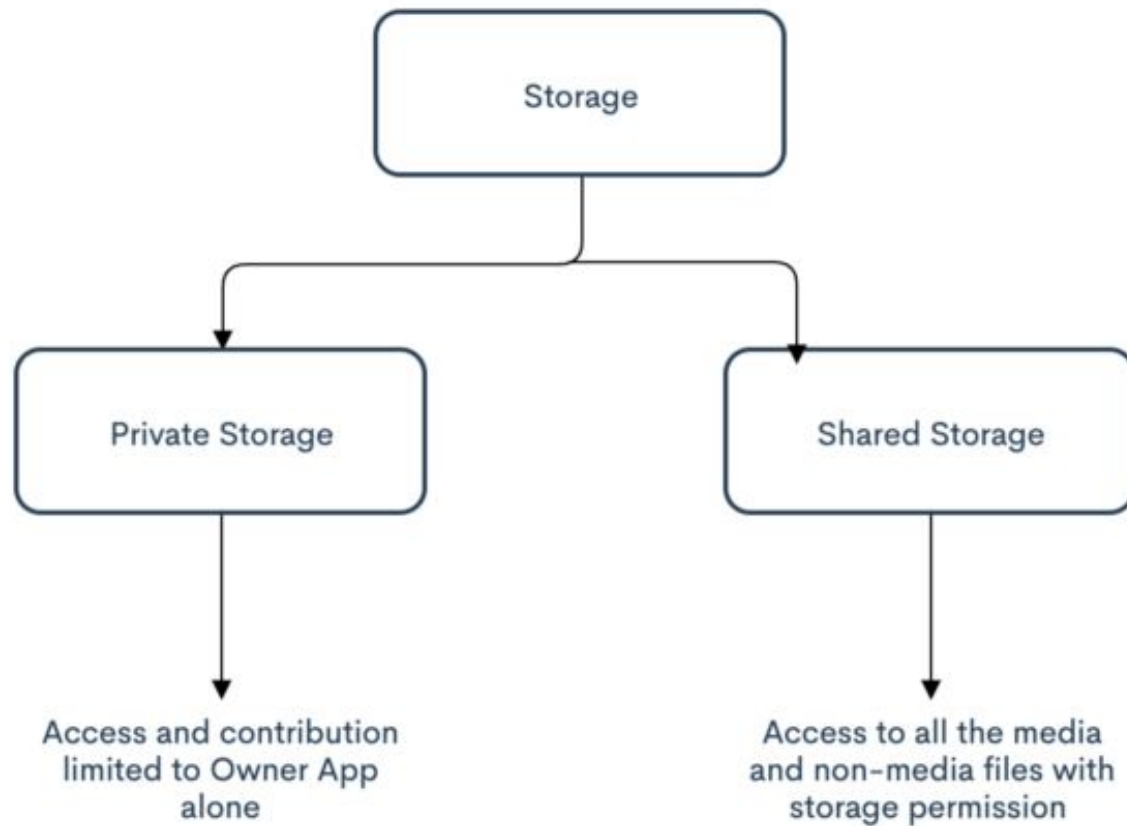
```
fun calculateDesiredBottomInset (topInset: Int, bottomInset: Int, listener: BarChangeListener): Int {  
    val hasKeyboard = isKeyboardAppeared(bottomInset)  
    val desiredBottomInset = if (hasKeyboard) bottomInset else 0  
    listener(topInset, if (hasKeyboard) 0 else bottomInset)  
    return desiredBottomInset  
}  
  
fun isKeyboardAppeared (bottomInset: Int) = bottomInset / resources.displayMetrics.heightPixels.toDouble() > .25
```

Usage

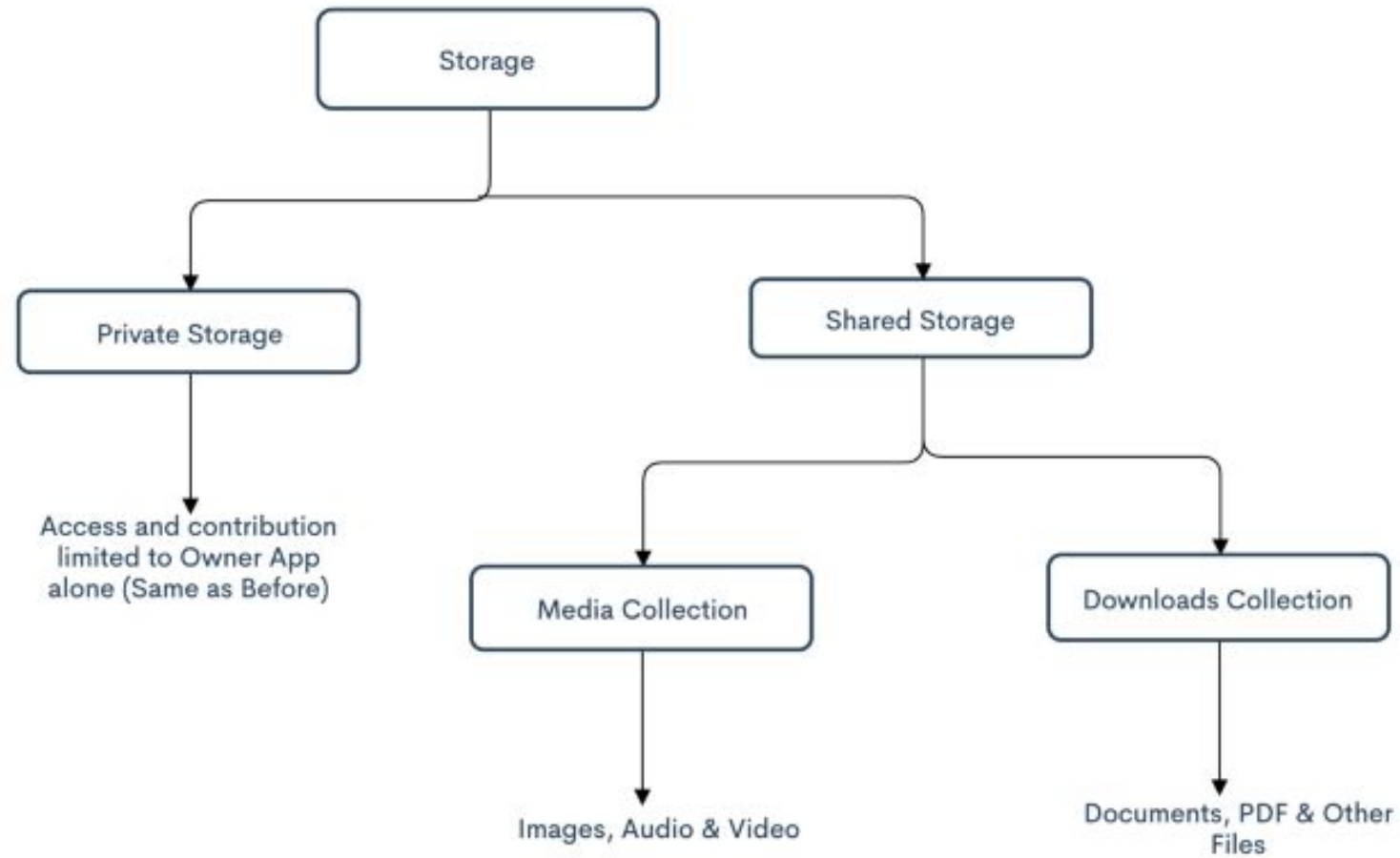
```
setBarTransparency { offsetTop, offsetBottom ->  
    ...  
}
```

Доступ к внешнему хранилищу

Как было....



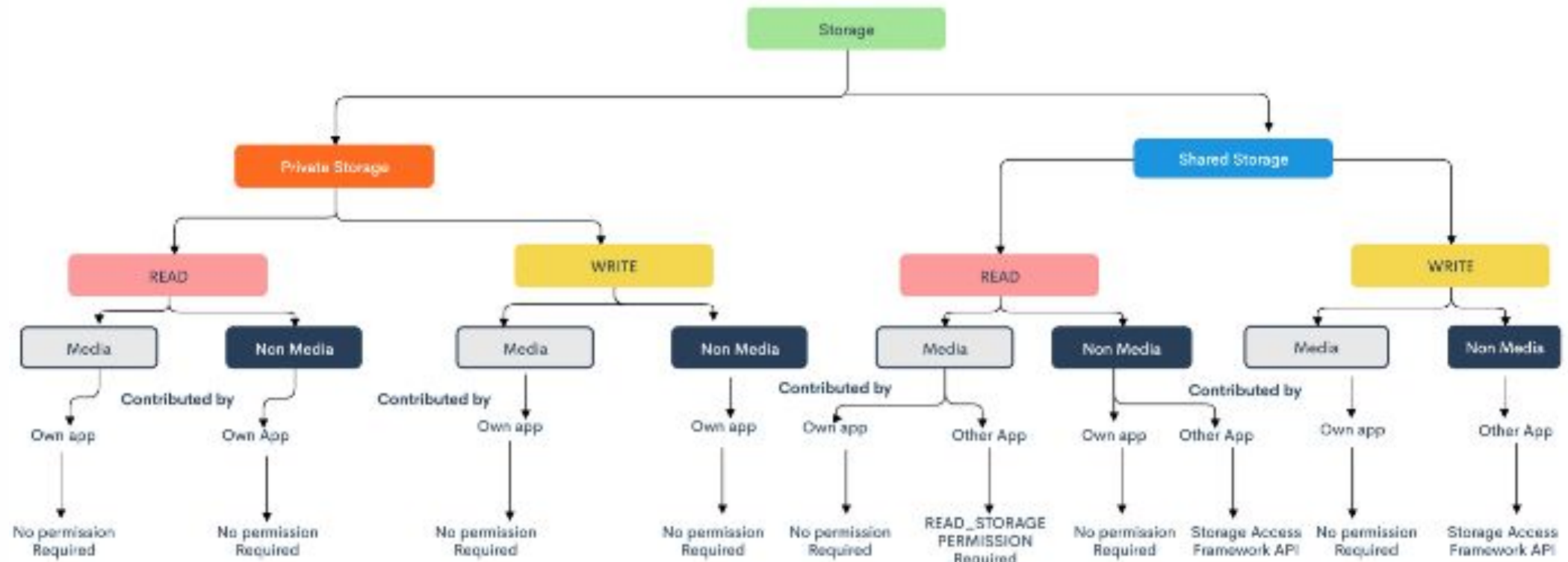
Scoped Storage



Scoped Storage. Кратко

1. Нет доступа к файлу по его пути. Вместо этого нужно использовать его Uri.
2. Приложение будет иметь **неограниченный доступ к своему** внутреннему и внешнему хранилищу для операций чтения и записи.
3. Доступ к коллекции медиафайлов других приложений можно получить с помощью разрешения `READ_STORAGE_PERMISSION`

Scoped Storage



Scoped Storage

Operation	File types	Storage type	CONTRIBUTED BY	Permission
READ	Media	Private Storage	Self	NO PERMISSION
READ	Media	Shared Storage	Self	NO_PERMISSION
READ	Media	Shared Storage	Other App	READ_STORAGE_PERMISSION
READ	Non media	Private Storage	Self	NO PERMISSION
READ	Non media	Shared Storage	Self	NO PERMISSION
READ	Non media	Shared Storage	Other APP	Storage access framework API
WRITE	Media	Private Storage	Self	NO PERMISSION
WRITE	Media	Shared Storage	Self	NO PERMISSION
WRITE	Non media	Private Storage	Self	NO PERMISSION
WRITE	Non media	Shared Storage	Other APP	Storage access framework API

Directories

Type	Access method	Permissions	Access	Uninstalled
app-specific files (internal storage)	getFilesDir() /data/user/...	Not needed	No other app can access	Data removed
app-specific files (external storage)	getExternalFilesDir() /storage/emulated/...	Not needed	Yes	Data removed
Media	MediaStore API	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE (<API 28)	Yes	Nothing
Other files	SAF	None	Through the system file picker	Nothing

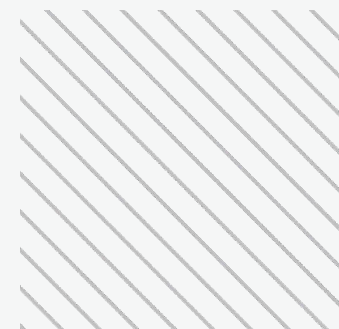
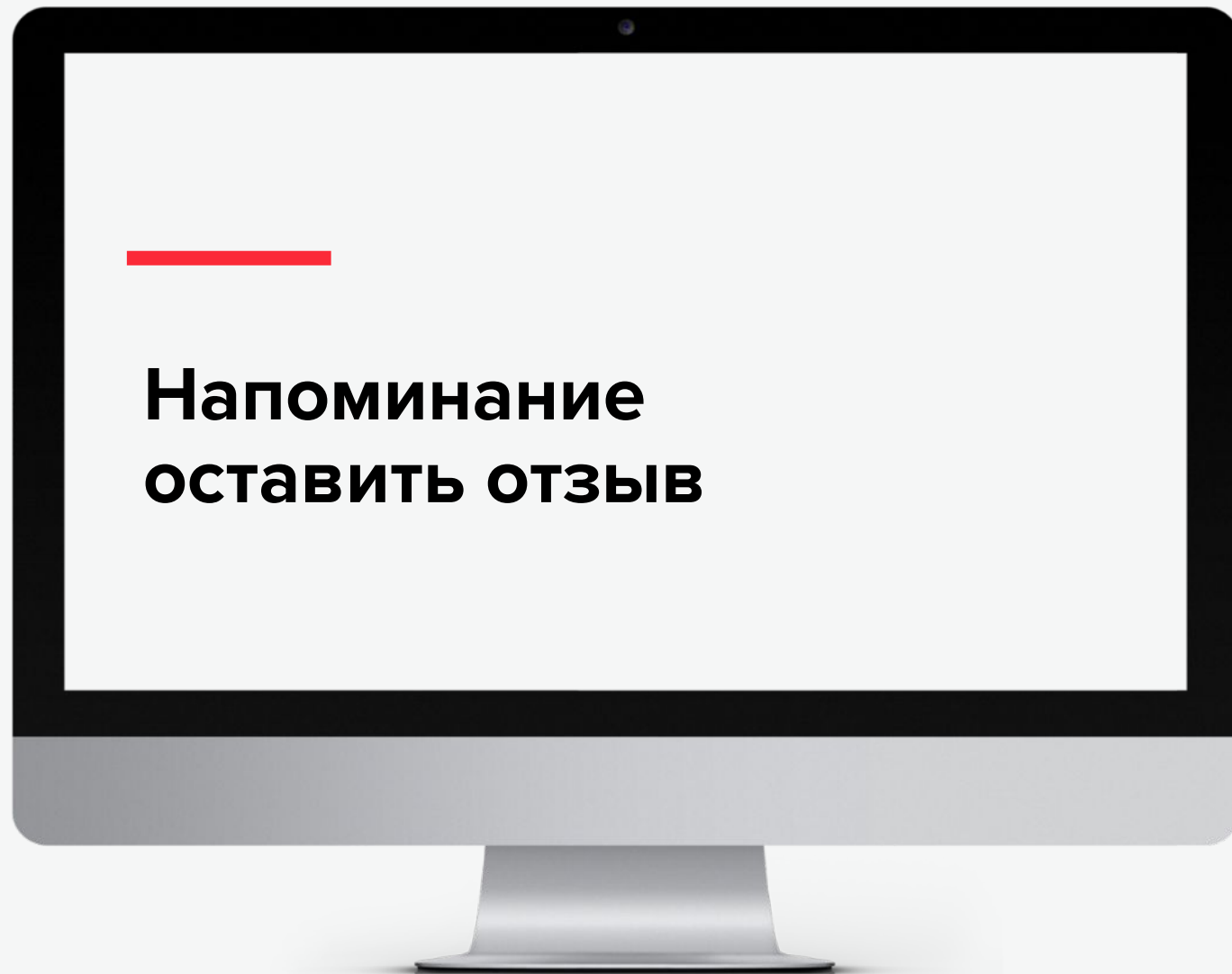
Советы Google

- Target api < 29 - отказаться от Scoped Storage
- Target api = 29 - отказаться от Scoped Storage
 - `android:requestLegacyExternalStorage="true"`
- Target Api >= 30 - использовать Scoped Storage

Storage Access Framework



```
fun openFile() {  
    val intent = Intent(Intent.ACTION_OPEN_DOCUMENT).apply {  
        addCategory(Intent.CATEGORY_OPENABLE)  
        type = "*/*"  
    }  
  
    startActivityForResult(intent, 10003)  
}
```



**СПАСИБО
ЗА ВНИМАНИЕ**

