RuStore

Анатолий Гусев

Установка приложений в Android

От доисторических времен
до наших дней

Обо мне

- 8 лет в Android разработке
- Работал в основном в Core командах
- Много работал с API установки приложений

Структура доклада

1/3 история развития API установки

2/3 Нюансы использования этих API

3/3 Баги и костыли

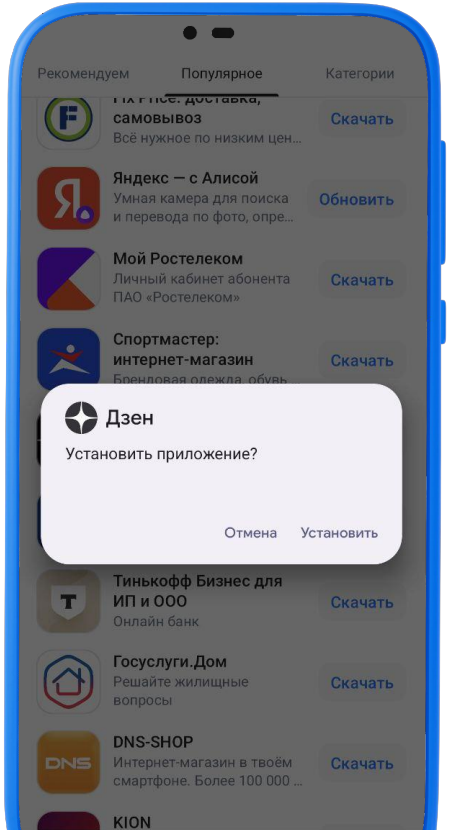
Установка через Intent

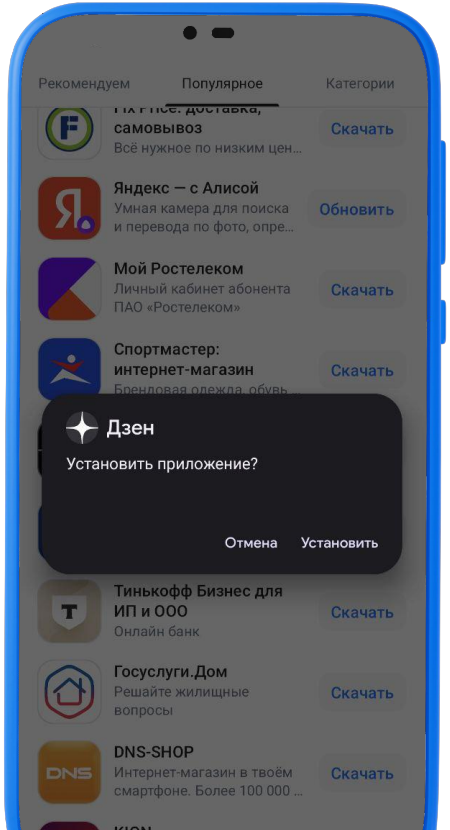
```
val uri = FileProvider.getUriForFile(context, providerAuthority, apkFile)
val intent = Intent(Intent.ACTION_VIEW).apply {
    setDataAndType(uri, "application/vnd.android.package-archive")
    addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
    addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
}
```

Установка через Intent

```
val uri = FileProvider.getUriForFile(context, providerAuthority, apkFile)
val intent = Intent(Intent.ACTION_VIEW).apply {
    setDataAndType(uri, "application/vnd.android.package-archive")
    addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
    addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
}
```

Установка через Intent





Установка через Intent

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == INSTALLING_REQUEST_CODE) {
        when (resultCode) {
            RESULT_OK -> processSuccess()

            RESULT_CANCELED -> processAborted()

            else -> processError()
        }
    }
}
```

Установка через Intent и MI UI

Установка через Intent и MI UI

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == INSTALLING_REQUEST_CODE) {
        when (resultCode) {
            RESULT_OK -> processSuccess()

            RESULT_CANCELED -> processAborted()

            else -> processError()
        }
    }
}
```

Android 2.3 APK Expansion Files

Решают проблему
приложений
большого размера

Android 2.3 APK Expansion Files

Решают проблему
приложений
большого размера

Допустимо
2 obb файла
для приложения,
main и patch

Android 2.3 APK Expansion Files

Решают проблему приложений большого размера

Допустимо 2 obb файла для приложения, main и patch

Доставка файла на устройство не гарантирована

Проблема фрагментации Android

В APK необходимо
включать все
для работы
на любом устройстве

Проблема фрагментации Android

В APK необходимо включать все для работы на любом устройстве

Приложение занимает много памяти на устройстве

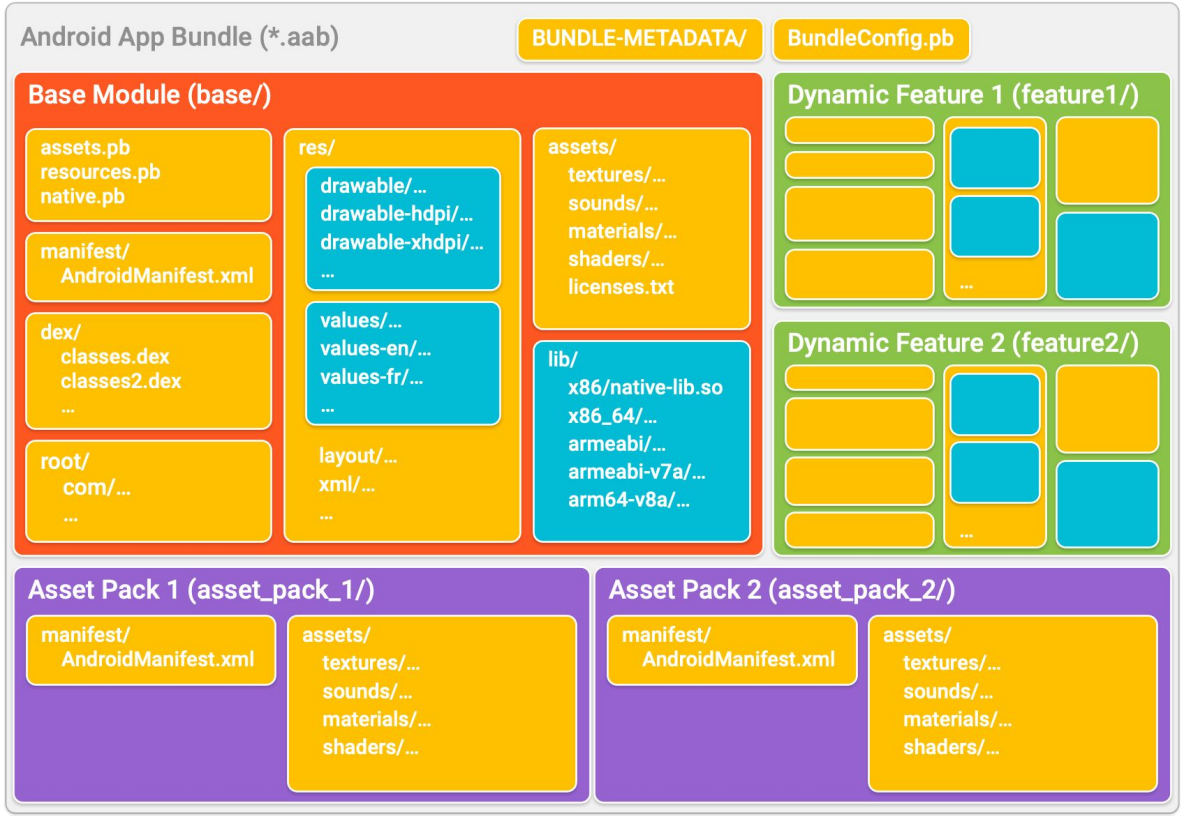
Проблема фрагментации Android

В APK необходимо включать все для работы на любом устройстве

Приложение занимает много памяти на устройстве

Вынуждены передавать ненужные данные по сети

Android 5 AAB формат



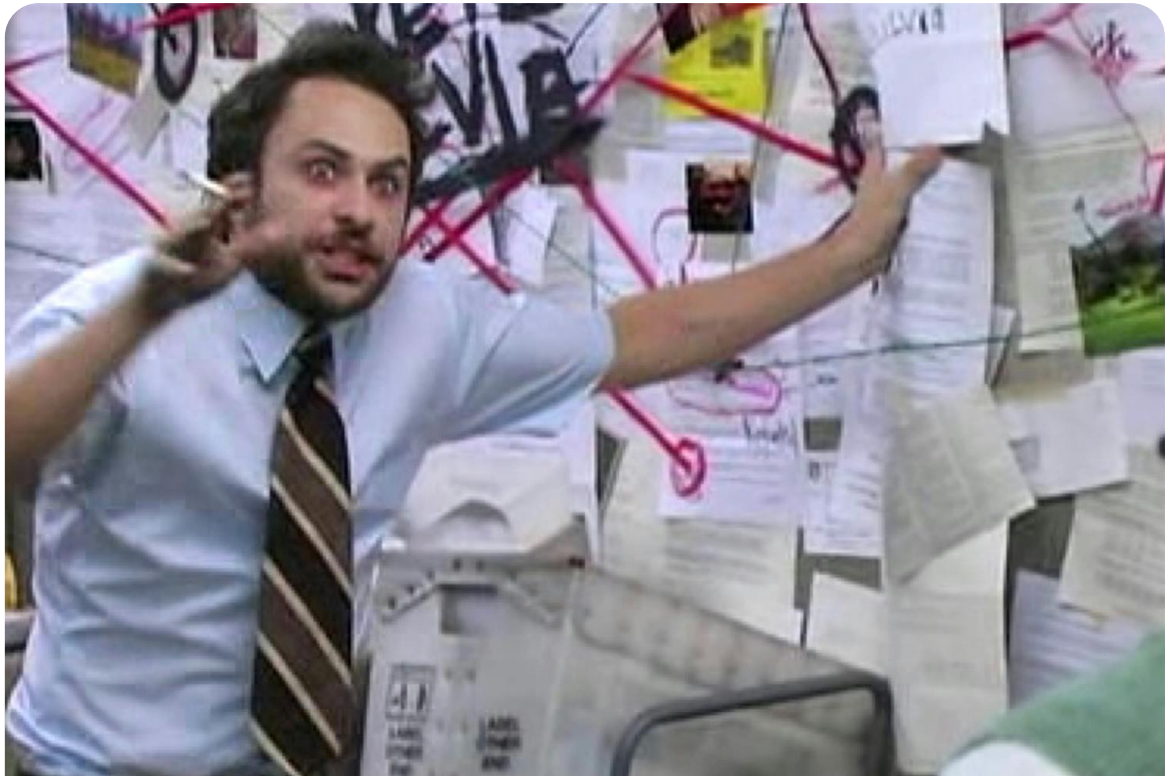
Android 5 AAB формат

В APK теперь включается только то, что нужно на целевом девайсе

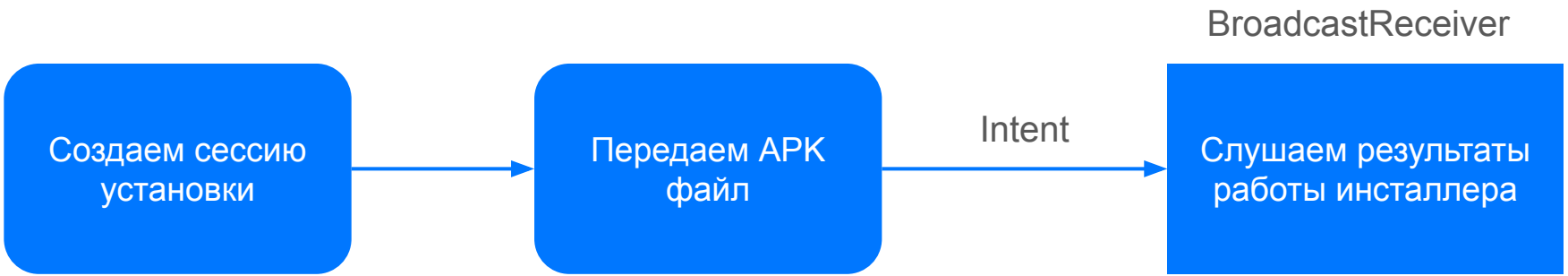
На замену obb файлам появились Asset-delivery

Появилась возможность налету загружать полноценные фичи приложения

Android 5 PackageInstaller session API



Android 5 PackageInstaller session API



Android 5 PackageInstaller session API

```
val sessionParams = SessionParams(SessionParams.MODE_FULL_INSTALL)
val sessionId = packageInstaller.createSession(sessionParams)
```

Android 5 PackageInstaller session API

```
packageInstaller.openSession(sessionId).use { session ->
    session.openWrite(apkFile.name, 0, apkFile.length()).use { sessionStream ->
        sessionStream.copyApkFrom(apkFile)
        session.fsync(sessionStream)
    }
    val statusReceiver = InstallBroadcastReceiver::class.java
    val intentsender = context.getIntentSender(statusReceiver)
    session.commit(intentsender)
}
```

Android 5 PackageInstaller session API

```
packageInstaller.openSession(sessionId).use { session ->
    session.openWrite(apkFile.name, 0, apkFile.length()).use { sessionStream ->
        sessionStream.copyApkFrom(apkFile)
        session.fsync(sessionStream)
    }
    val statusReceiver = InstallBroadcastReceiver::class.java
    val intentsender = context.getIntentSender(statusReceiver)
    session.commit(intentsender)
}
```

Android 5 PackageInstaller session API

```
packageInstaller.openSession(sessionId).use { session ->
    session.openWrite(apkFile.name, 0, apkFile.length()).use { sessionStream ->
        sessionStream.copyApkFrom(apkFile)
        session.fsync(sessionStream)
    }
    val statusReceiver = InstallBroadcastReceiver::class.java
    val intentsender = context.getIntentSender(statusReceiver)
    session.commit(intentsender)
}
```

Android 5 PackageInstaller session API

```
packageInstaller.openSession(sessionId).use { session ->
    session.openWrite(apkFile.name, 0, apkFile.length()).use { sessionStream ->
        sessionStream.copyApkFrom(apkFile)
        session.fsync(sessionStream)
    }
    val statusReceiver = InstallBroadcastReceiver::class.java
    val intentsender = context.getIntentSender(statusReceiver)
    session.commit(intentsender)
}
```

Android 5 PackageInstaller session API

```
internal class InstallBroadcastReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        val extras = intent?.extras
        if (context != null && extras != null) {
            val status = extras.getInt(EXTRA_STATUS)
            val sessionId = extras.getInt(EXTRA_SESSION_ID, INVALID_SESSION_ID)

            if (sessionId != INVALID_SESSION_ID) {
                processStatus(intent, status)
            }
        }
    }
    ...
}
```

Android 5 PackageInstaller session API

```
internal class InstallBroadcastReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        val extras = intent?.extras
        if (context != null && extras != null) {
            val status = extras.getInt(EXTRA_STATUS)
            val sessionId = extras.getInt(EXTRA_SESSION_ID, INVALID_SESSION_ID)

            if (sessionId != INVALID_SESSION_ID) {
                processStatus(intent, status)
            }
        }
    }
    ...
}
```

Android 5 PackageInstaller session API

```
internal class InstallBroadcastReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        val extras = intent?.extras
        if (context != null && extras != null) {
            val status = extras.getInt(EXTRA_STATUS)
            val sessionId = extras.getInt(EXTRA_SESSION_ID, INVALID_SESSION_ID)

            if (sessionId != INVALID_SESSION_ID) {
                processStatus(intent, status)
            }
        }
    }
    ...
}
```

Android 5 PackageInstaller session API

```
internal class InstallBroadcastReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        val extras = intent?.extras
        if (context != null && extras != null) {
            val status = extras.getInt(EXTRA_STATUS)
            val sessionId = extras.getInt(EXTRA_SESSION_ID, INVALID_SESSION_ID)

            if (sessionId != INVALID_SESSION_ID) {
                processStatus(intent, status)
            }
        }
    }
    ...
}
```

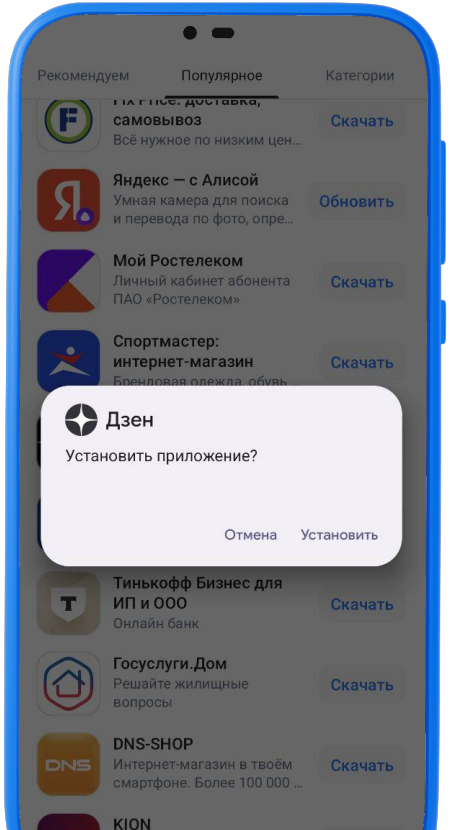
Android 5 PackageInstaller session API

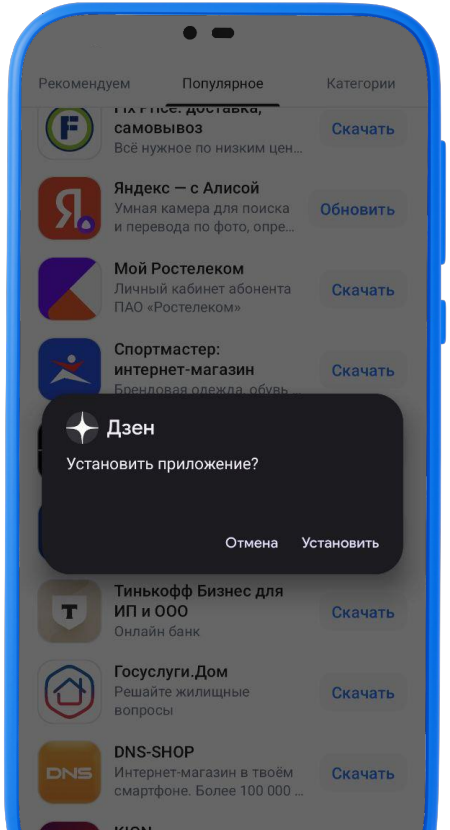
```
private fun processStatus(intent: Intent?, status: Int) {
    when (status) {
        PackageInstaller.STATUS_PENDING_USER_ACTION -> handleConfirmationStatus(intent, sessionId)

        PackageInstaller.STATUS_SUCCESS -> handleSuccess()

        else -> {
            processError(status)
        }
    }
}
```

Android 5 PackageInstaller session API

```
private fun processStatus(intent: Intent?, status: Int) {
    when (status) {
        PackageInstaller.STATUS_PENDING_USER_ACTION -> handleConfirmationStatus(intent, sessionId)

        PackageInstaller.STATUS_SUCCESS -> handleSuccess()

        else -> {
            processError(status)
        }
    }
}
```

Android 5 PackageInstaller session API





Android 5 PackageInstaller session API

```
private fun processStatus(intent: Intent?, status: Int) {
    when (status) {
        PackageInstaller.STATUS_PENDING_USER_ACTION -> handleConfirmationStatus(intent, sessionId)

        PackageInstaller.STATUS_SUCCESS -> handleSuccess()

        else -> {
            processError(status)
        }
    }
}
```

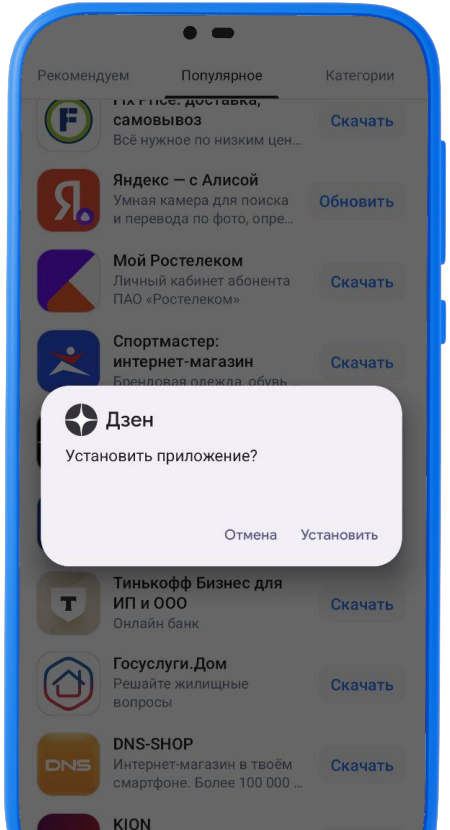
Android 5 PackageInstaller session API

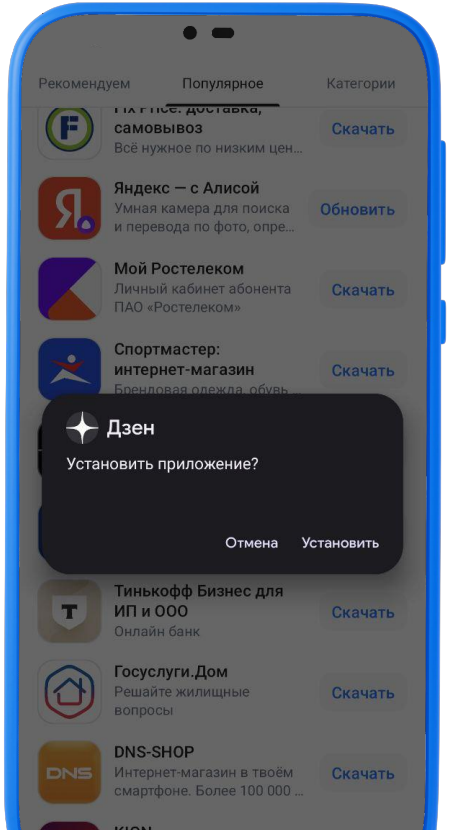
```
private fun processStatus(intent: Intent?, status: Int) {
    when (status) {
        PackageInstaller.STATUS_PENDING_USER_ACTION -> handleConfirmationStatus(intent, sessionId)

        PackageInstaller.STATUS_SUCCESS -> handleSuccess()

        else -> {
            processError(status)
        }
    }
}
```

Android 5 PackageInstaller session API

```
PackageInstaller.STATUS_FAILURE,

PackageInstaller.STATUS_FAILURE_BLOCKED,

PackageInstaller.STATUS_FAILURE_ABORTED,

PackageInstaller.STATUS_FAILURE_INVALID,

PackageInstaller.STATUS_FAILURE_CONFLICT,

PackageInstaller.STATUS_FAILURE_STORAGE,

PackageInstaller.STATUS_FAILURE_INCOMPATIBLE,
```

Android 5 PackageInstaller session API





Android 5 PackageInstaller session API

Android 5 PackageInstaller session API

Смотреть
на прогресс сессии
установки до диалога
и после

На onResume
проверять
установлено
ли приложение

Android 5 PackageInstaller session API

На некоторых версиях MI UI (~9-12) любая установка через PackageInstaller вернет STATUS_FAILURE при включенной оптимизации.

Разрешение на установку приложений

```
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.O) {
    Settings.Secure.getInt(contentResolver, Settings.Secure.INSTALL_NON_MARKET_APPS) == 1
} else {
    packageManager.canRequestPackageInstalls()
}
```

Разрешение на установку приложений

```
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.O) {
    Settings.Secure.getInt(contentResolver, Settings.Secure.INSTALL_NON_MARKET_APPS) == 1
} else {
    packageManager.canRequestPackageInstalls()
}
```

Разрешение на установку приложений

```
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.O) {
    Settings.Secure.getInt(contentResolver, Settings.Secure.INSTALL_NON_MARKET_APPS) == 1
} else {
    packageManager.canRequestPackageInstalls()
}
```

Пермишены установки

Привилегированные

android.permission.INSTALL_PACKAGES

android.permission.INSTALL_SELF_UPDATES

android.permission.INSTALL_PACKAGE_UPDATES

Стандартный

android.permission.REQUEST_INSTALL_PACKAGES

Android 12 Упрощение обновлений

```
val params = SessionParams(SessionParams.MODE_FULL_INSTALL).apply {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        setRequireUserAction(SessionParams.USER_ACTION_NOT_REQUIRED)
    }
}
```

Android 12 Упрощение обновлений

```
val params = SessionParams(SessionParams.MODE_FULL_INSTALL).apply {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        setRequireUserAction(SessionParams.USER_ACTION_NOT_REQUIRED)
    }
}
```

Android 14 Преаппрув установки

```
val session = packageInstaller.openSession(sessionId)
session.requestUserPreapproval(
    PreapprovalDetails.Builder()
        .setIcon(iconBitmap)
        .setLabel(message)
        .setLocale(locale)
        .build(),
    intentSender,
)
```

Android 14 Преаппрув установки

```
val session = packageInstaller.openSession(sessionId)

session.requestUserPreapproval(
    PreapprovalDetails.Builder()
        .setIcon(iconBitmap)
        .setLabel(message)
        .setLocale(locale)
        .build(),
    intentSender,
)
```

Android 14 Установка в удобный момент

```
PackageInstaller.InstallConstraints.Builder()
    .setNotInCallRequired()
    .build()
```

Android 14 Установка в удобный момент

```
packageInstaller.commitSessionAfterInstallConstraintsAreMet(
    sessionId,
    intentSender,
    installContraints,
    timeout = Duration.ofHours(12).toMillis()
)
```

Android 14 Update ownership

```
val sessionParams = SessionParams(SessionParams.MODE_FULL_INSTALL).apply {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.UPSIDE_DOWN_CAKE) {
        setRequestUpdateOwnership(true)
    }
}

packageInstaller.createSession(sessionParams)
```

Android 14 Update ownership

```
val sessionParams = SessionParams(SessionParams.MODE_FULL_INSTALL).apply {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.UPSIDE_DOWN_CAKE) {
        setRequestUpdateOwnership(true)
    }
}

packageInstaller.createSession(sessionParams)
```

Android 14 Update ownership

```
val sessionParams = SessionParams(SessionParams.MODE_FULL_INSTALL).apply {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.UPSIDE_DOWN_CAKE) {
        setRequestUpdateOwnership(true)
    }
}

packageInstaller.createSession(sessionParams)
```

Итоги

API установки приложений постепенно становится лучше

Иногда появляются спорные фичи

При работе с API установки стоит учитывать вендор-специфичные баги