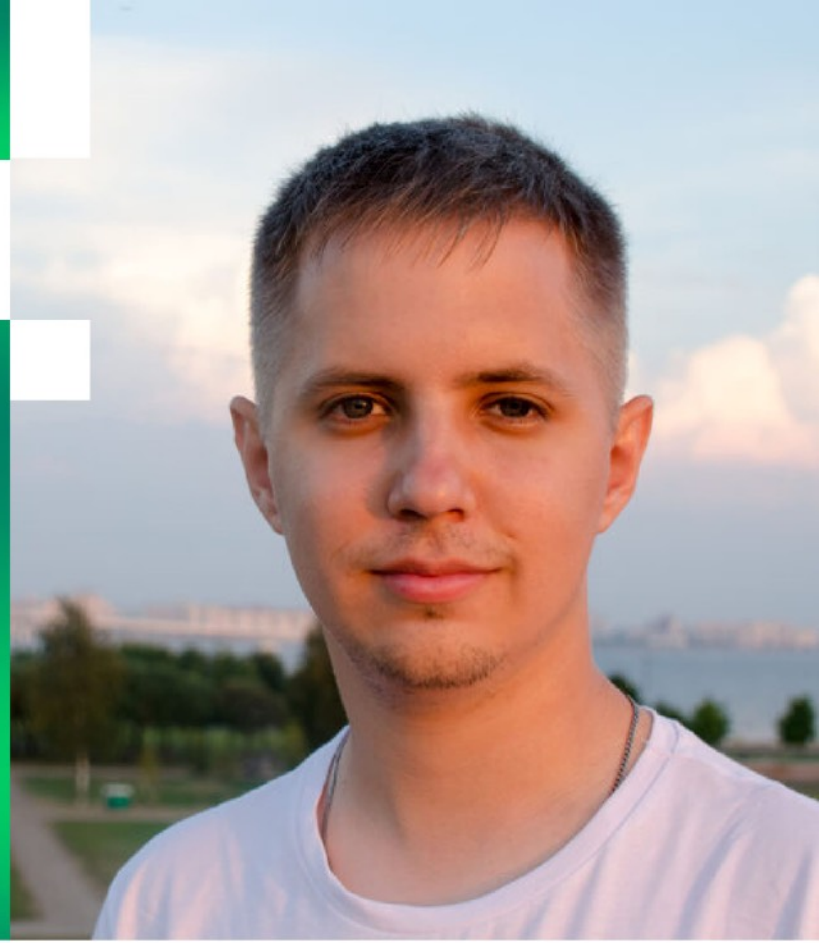
podlodka

Android Crew
14-18 февраля

Jetpack Compose



Андрей Берюхов
Авито

Доклад
Compose for Widgets & Wearables

avito.tech

Обо Мне

Android Engineer @ Avito.Tech

Статьи и доклады про:

- Jetpack (Compose)
- Мультиплатформу
- Многомодульность

Open-source проекты:

- Coffeegram (Compose Android & Desktop) - 155+ ★

Ментор и спикер Android Academy

Compose

01. Android (Jetpack)
`Compose → Skia`

02. Desktop (JetBrains)
`Compose → Skia`

03. Web (JetBrains)
`Compose → DOM`

04. Glance
`Composable → RemoteViews/Tiles/XML`

Android UI (ex. XML)

01. Activity

`Phones, Watch, TV, Auto`

02. Remote Views

`Widgets, Notifications`

03. Tiles

`Watch`

04. Previews

`Widgets`

План

01. Историческая справка Wear OS & Widgets

02. Compose for Wearables

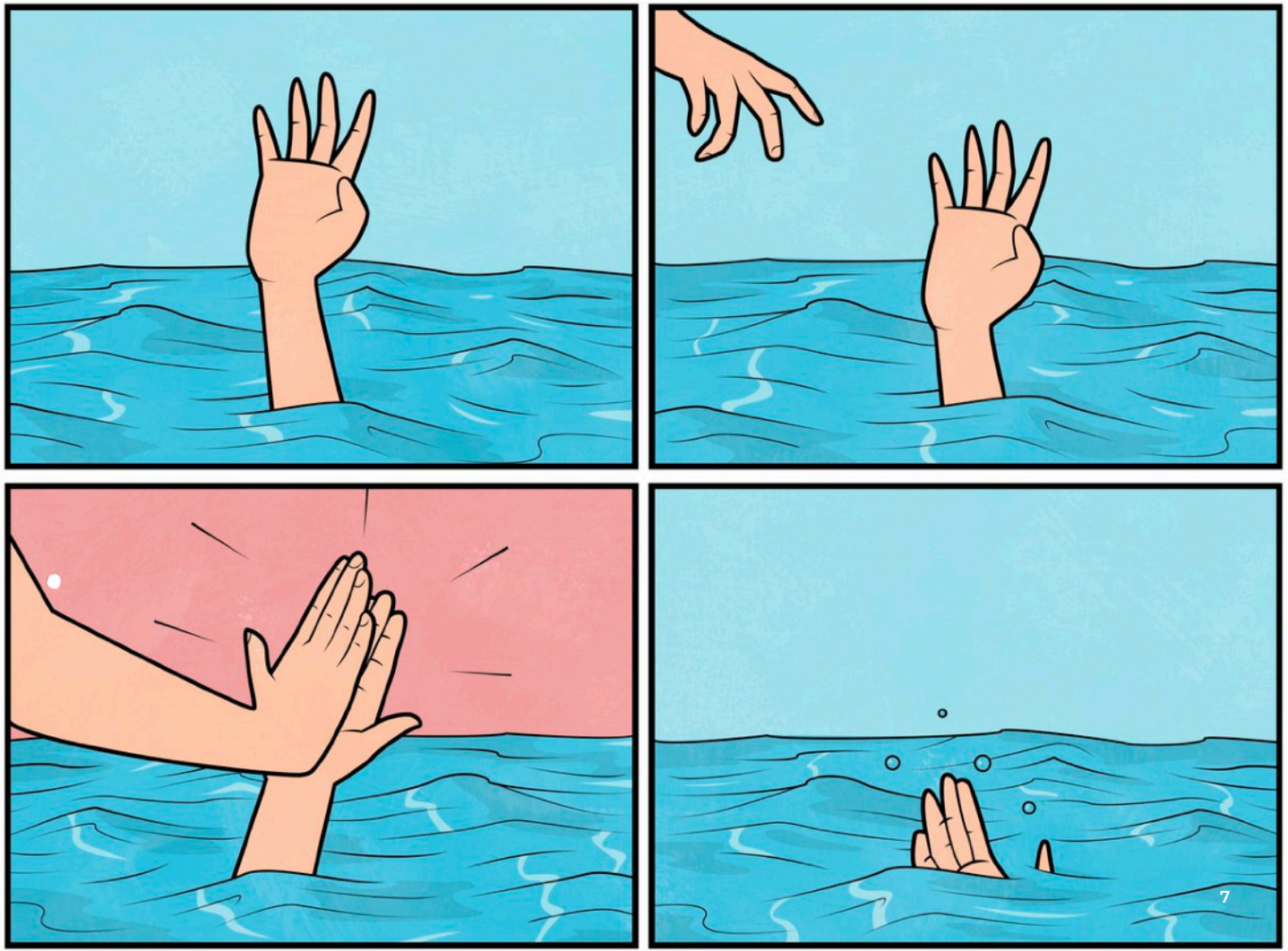
03. Jetpack Glance (Widgets + Wear Tiles + ...)

04. Куда движутся Compose / Часы / Виджеты?

Но сначала

Измерим температуру

Кто делал приложение для часов?




Кто делал виджеты в приложении?


THAT'S
CLASSIC
memegenerator.net

Немного статистики

Wear OS by Google

Количество установок

10 000 000+

`< 50 000 000`

Google Pay

Количество установок

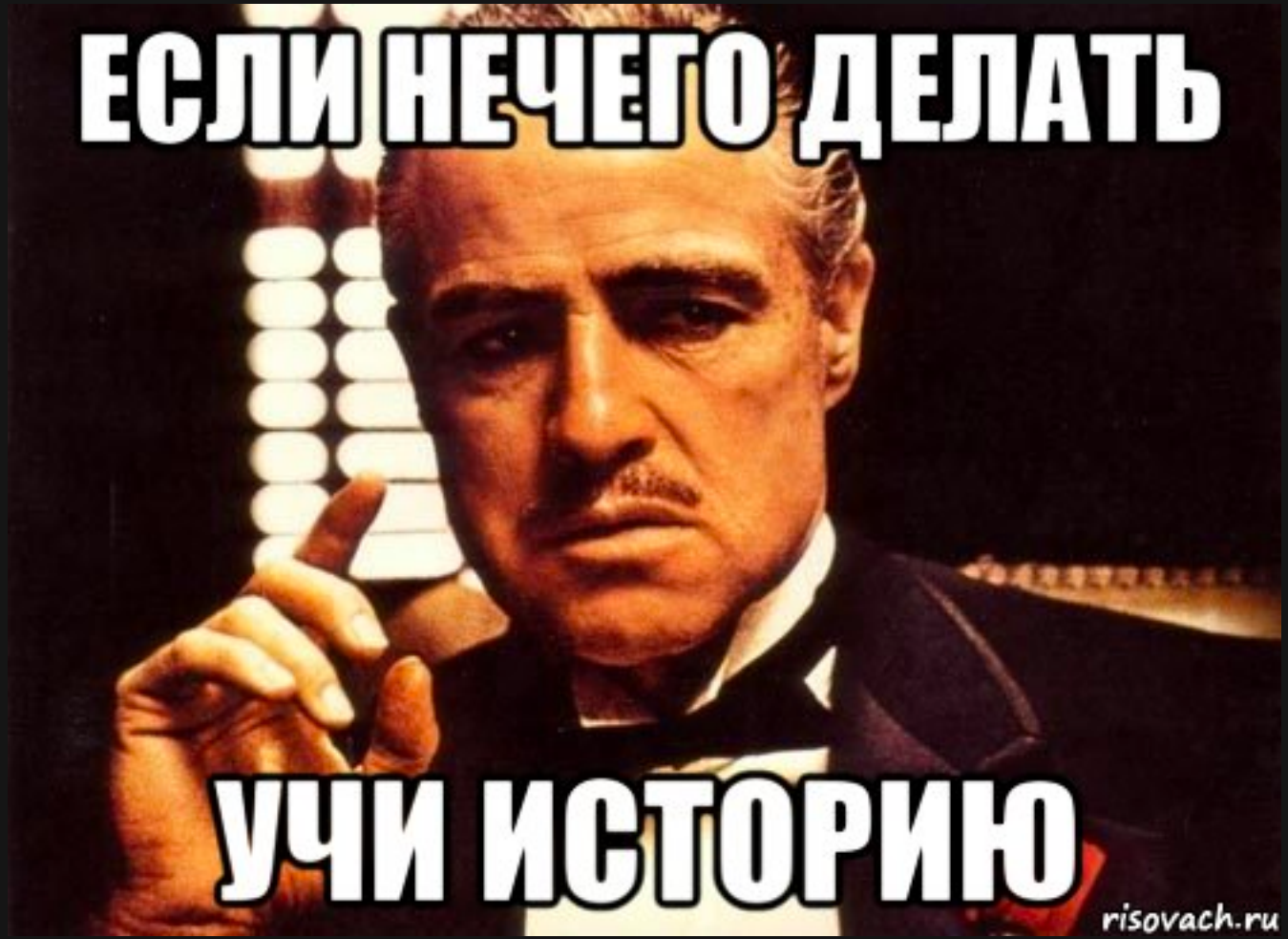
100 000 000+

Google

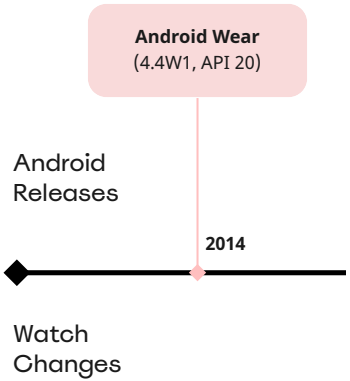
Количество установок

10 000 000 000+

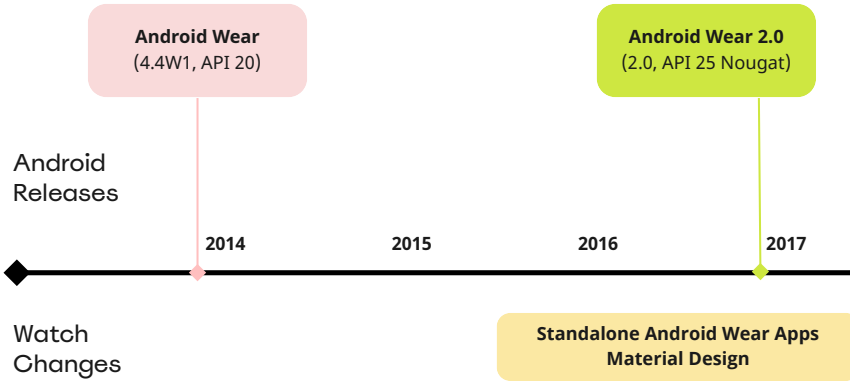
`В ~1000 раз больше`


ЕСЛИ НЕЧЕГО ДЕЛАТЬ
УЧИ ИСТОРИЮ
risovach.ru

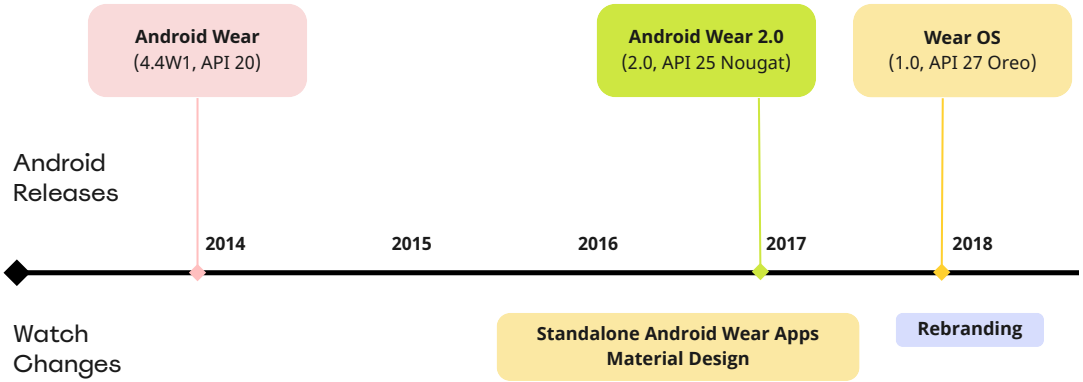
История часов



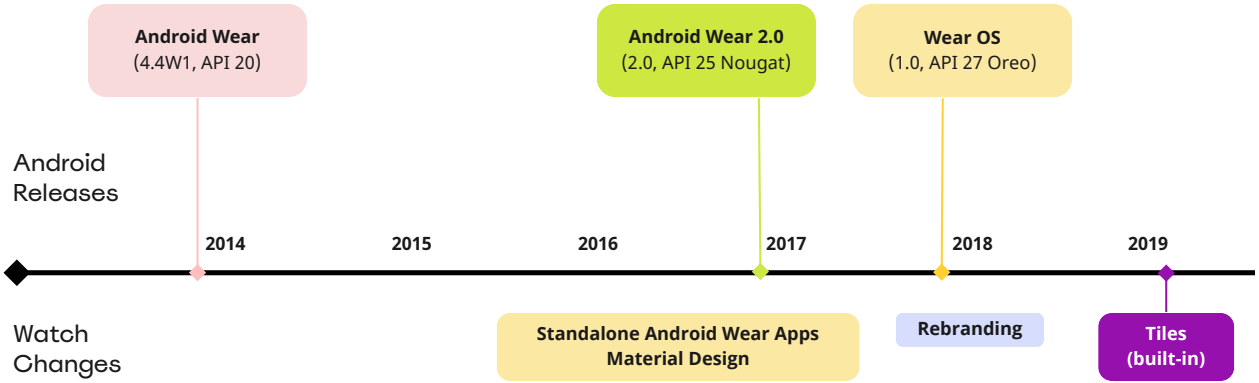
История часов



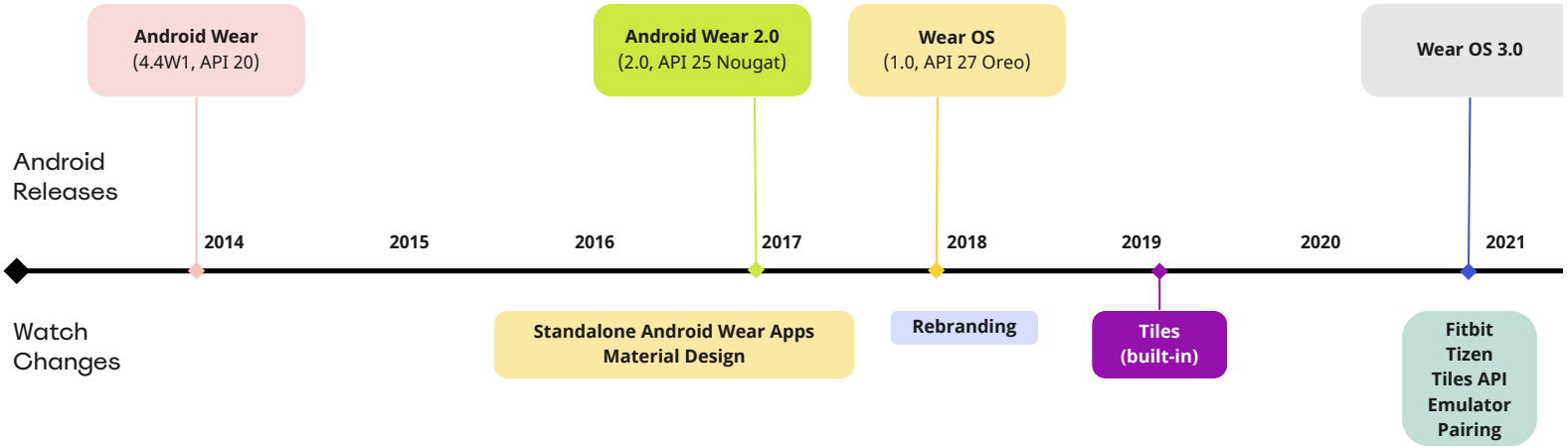
История часов



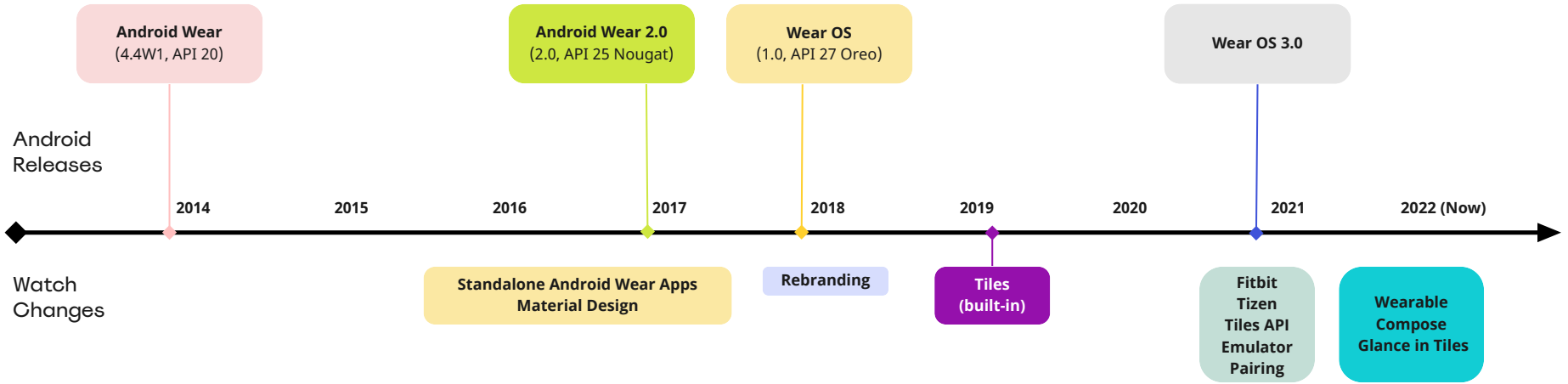
История часов



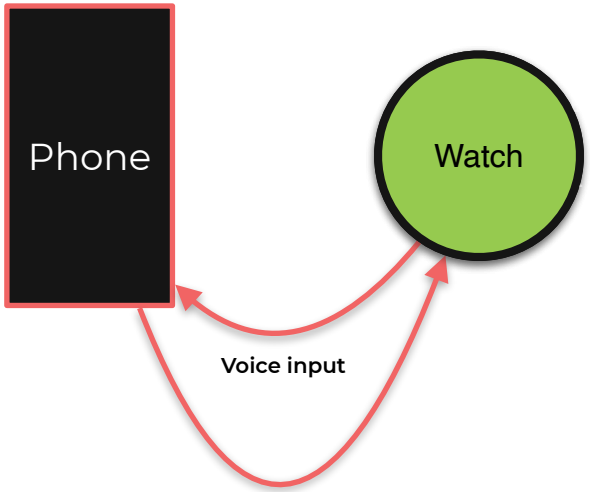
История часов



История часов



Companion vs Standalone



Bluetooth

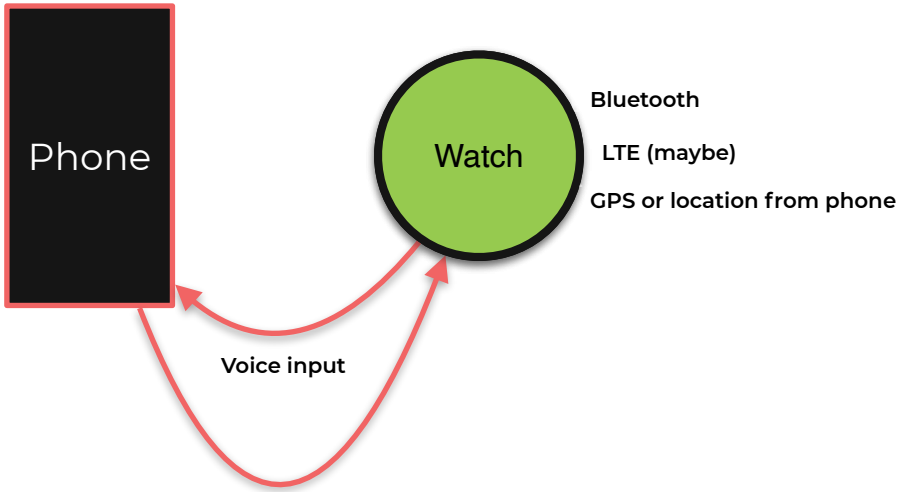
LTE (maybe)

GPS or location from phone

Completing a task from another device

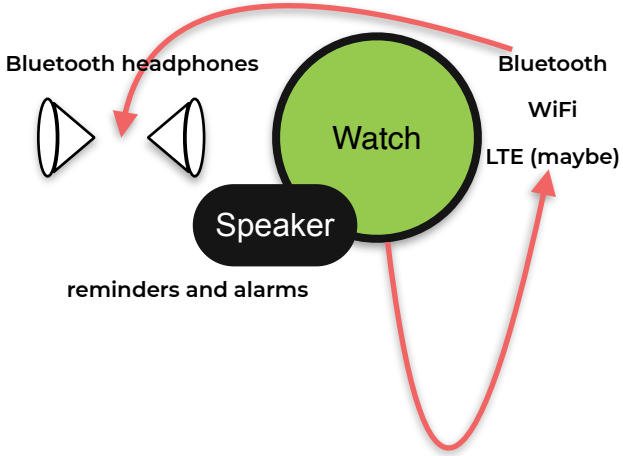
Playback controls for music

Companion vs Standalone



Completing a task from another device

Playback controls for music



Listen for music without phone

Часы бывают двух видов

Из чего состоит приложение для часов

Overlay = Full App

- В целом похоже на обычное приложение (с ограничениями часов)
- Может быть несколько экранов => Нужна навигация

Overlay
P1: What's the weather right now?

P2: What's the weather today?

P3: What's the hourly breakdown?

P3: Preferences

Из чего состоит приложение для часов

Complication ~ Виджет для циферблата
Notification - Нотификация для часов

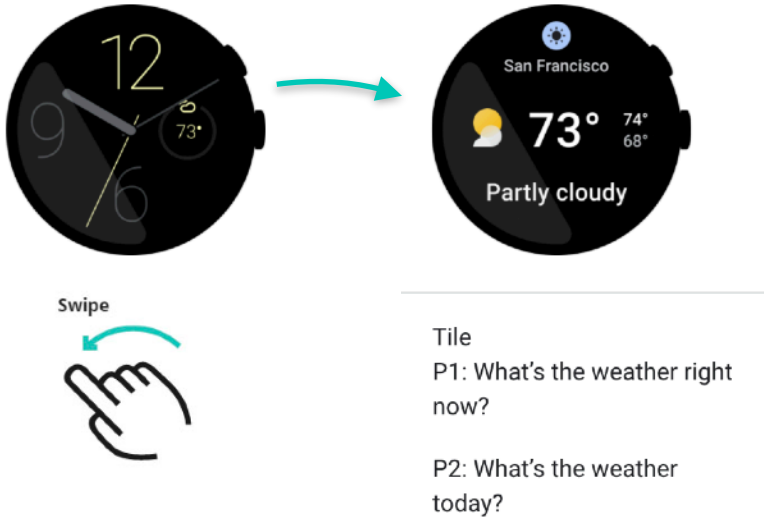
Complication
P1: What's the weather right now?

Notification
P1 Tell me about a severe weather advisory

Из чего состоит приложение для часов

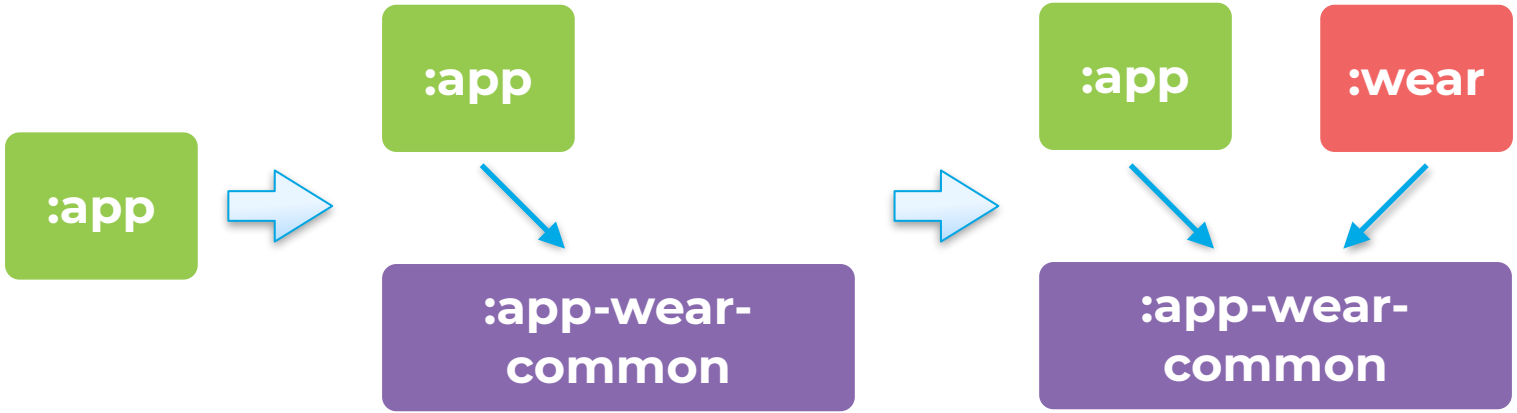
Tile - один экран в меню

- По сути это сервис с пермишном и интент-фильтром `BIND_TILE_PROVIDER`
- Ограниченный набор собственных View
- API добавлен в 2021



https://developer.android.com/training/wearables/tiles

Как сделать приложение для часов



Commit "Add wear" in Coffeegram

Как сделать приложение для часов

:wear

- Gradle:

```
plugins { id("com.android.application") … }
```

- Manifest:

```
<uses-feature
android:name="android.hardware.type.watch" />
```

Commit "Add wear" in Coffeegram

Как сделать Overlay

```
Manifest → WearActivity (intent-filters = [Main + Launcher])


class WearActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.wear_activity)
    }
}
```

Как сделать Overlay после Podlodka

```
class WearActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            CoffeegramTheme {
                PagesContent()
            }
        }
    }
}
```

Compose “телефонный"


```
@Composable //псевдокод
fun PagesContent() {
    LazyColumn( ... ) {
        Chip( ... ) {
            leadingIcon = {},
            content = { Text() }
        }
    }
}
```

[Больше кода в ветке]

С часами всё, теперь виджеты

~~С часами всё, теперь виджеты~~

Compose for Wearables

Новые артефакты

```
wearCompose = “1.0.0-alpha16"

"androidx.wear.compose:compose-foundation"
"androidx.wear.compose:compose-material"
"androidx.wear.compose:compose-navigation"
```

MinSDK = 25

Compose “телефонный"


```
@Composable //псевдокод
fun PagesContent() {
    LazyColumn( ... ) {
        Chip( ... ) {
            leadingIcon = {},
            content = { Text() }
        }
    }
}
```

Compose Wearable

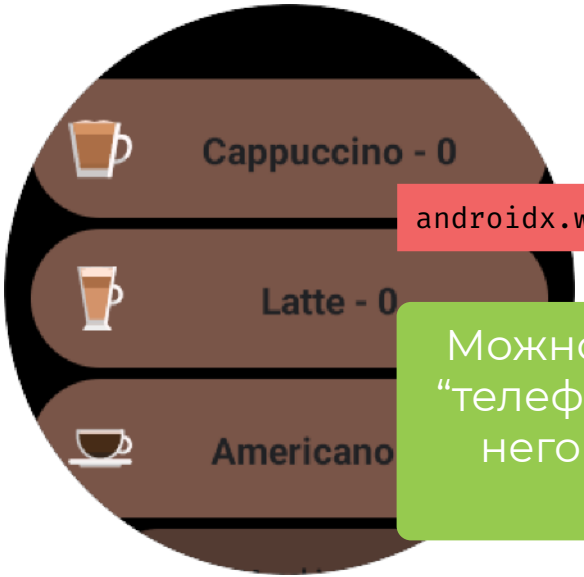

```
@Composable //псевдокод
fun PagesContent() {
    ScalingLazyColumn( ... ) {
        Chip( ... ) {
            icon = {},
            label = { Text() }
        }
    }
}
```

Compose Wearable

androidx.wear.compose.material.

```
@Composable //псевдокод
fun PagesContent() {
    ScalingLazyColumn( ... ) {
        Chip( ... ) {
            icon = {},
            label = { Text() }
        }
    }
}
```

Compose Wearable



androidx.wear.compose.material.

```
@Composable //псевдокод
fun PagesContent() {
    ScalingLazyColumn( ... ) {
        Chip( ... ) {
            ico
            t() }
    }
}
```

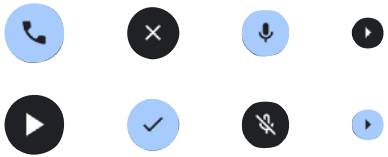
Можно использовать и “телефонный” Chip, но у него другие цвета и размеры

Что в библиотеке Foundation

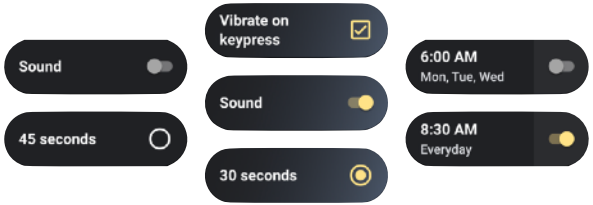
- Пока только код для скруглений
- BasicCurvedText, CurvedRowScope

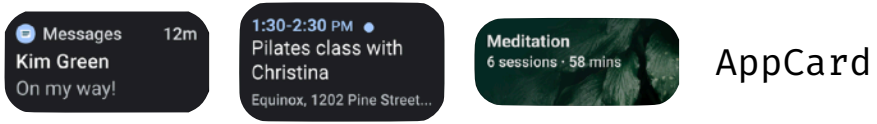
Что в библиотеке Material

Button



ToggleChip





AppCard

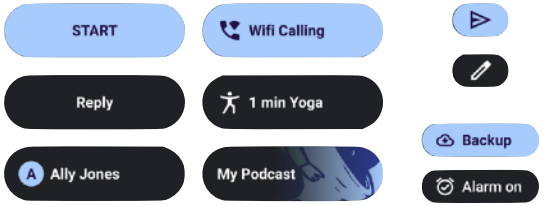
TimeText

CurvedText

ScalingLazyColumn

SwipeToDismissBox

Chip

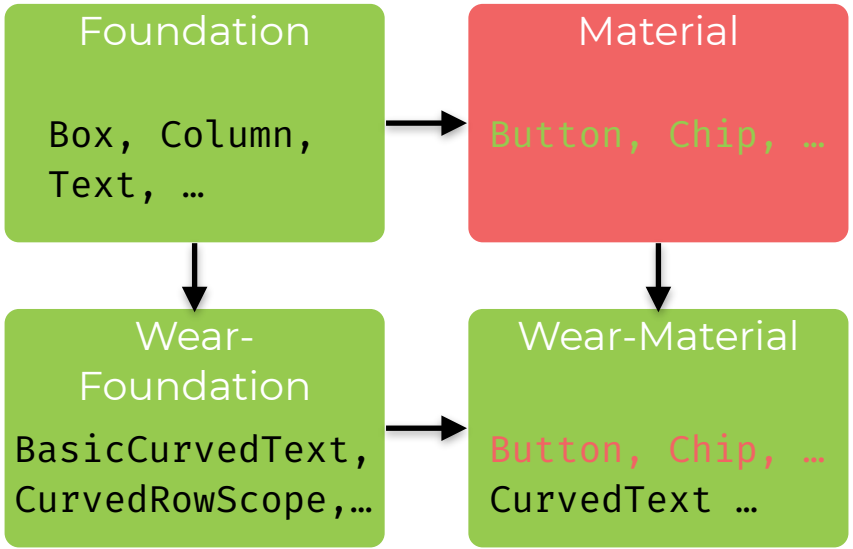


Scaffold

https://developer.android.com/training/wearables/compose

Почему лучше не смешивать Material

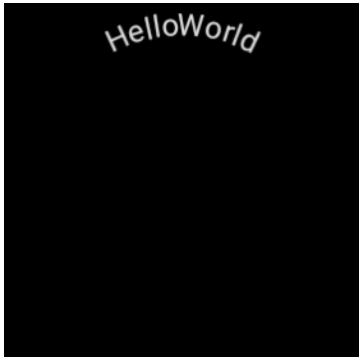
- Технически возможно
- Телефонная версия не оптимизирована для Wear OS
- Непредсказуемое поведение
  - Собственные `MaterialTheme`



https://developer.android.com/training/wearables/compose

CurvedText и CurvedRow

```
CurvedRow {
    CurvedText("Hello ")
    CurvedText("World")
}
```



CurvedText и CurvedRow

```
CurvedRow(clockwise = false) {
    CurvedText("Hello ")
    CurvedText("World")
}
```

CurvedText и CurvedRow

```
CurvedRow(anchor = 270f) {
    CurvedText("Hello North World")
}

CurvedRow(anchor = 90f) {
    CurvedText("Hello South World", clockwise = false)
}
```

CurvedText и CurvedRow

```
CurvedRow {
    Icon(imageVector = Icons.Default.Face, "")
    CurvedText("Hello ")
    Icon(imageVector = Icons.Default.ThumbUp, "")
    CurvedText("World")
    Icon(imageVector = Icons.Default.LocationOn, "")
}
```

CurvedText и CurvedRow

- Круглый
- Но остается круглым и на квадратных часах
- Можно проверять “круглость” и отдавать нужный Composable

https://commonsware.com/blog/2022/01/17/compose-wear-curvedrow-curvedtext.html

ScalingLazyColumn

- Основное отличие - уменьшение карточки при приближении к краю списка

SwipeToDismissBox

```
val state = rememberSwipeToDismissBoxState()
    SwipeToDismissBox(
        state = state
    ) { swipeBackgroundScreen →
        if (swipeBackgroundScreen) {
            Text(text = "Are you sure?")
        } else {
            Content()
        }
    }
```

Scaffold

- Vignette
- PositionIndicator
- TimeText

Scaffold Vignette

```
Scaffold(
    vignette = { Vignette(vignettePosition = VignettePosition.Top) }
) {
    Content()
}
```

None

Top

TopAndBottom

Scaffold PositionIndicator

```
val scalingLazyListState = rememberScalingLazyListState()
Scaffold(
    positionIndicator = { PositionIndicator(scalingLazyListState) }
) {
    ScalingLazyColumn( ... ,
        state = scalingLazyListState
    ) { ... }
}
```

None

With

Scaffold TimeText

```
Scaffold(
    timeText = { TimeText() }
) {
    Content()
}
```

None

With

Что в библиотеке Navigation

- Основное отличие = `SwipeDismissableNavHost`

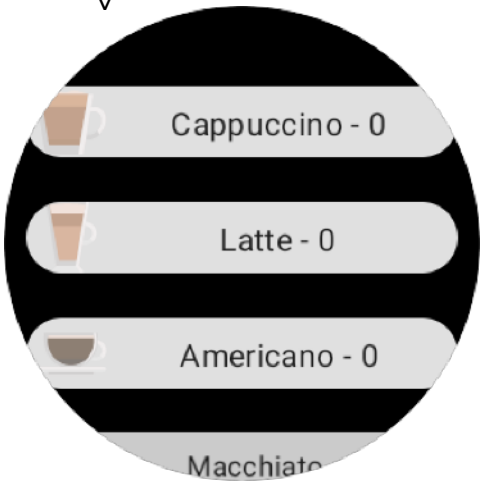
Интероп

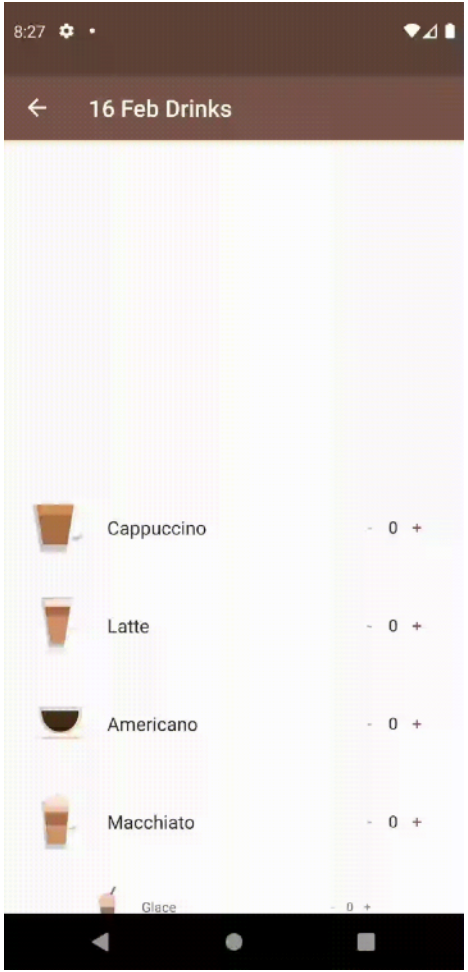
Compose for Wearable (ScalingLazyList) на телефоне ->

“Телефонный” Compose на часах

|

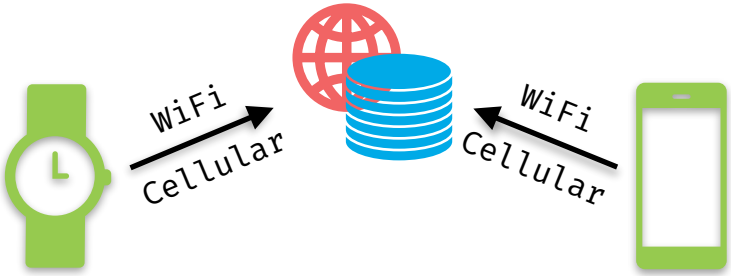
V





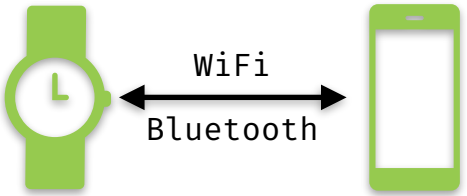
Откуда взять данные

- Ходить в сеть напрямую



Откуда взять данные

- Использовать Data Layer
  - Обертка над транспортами (Bluetooth или Wi-Fi) для более быстрой синхронизации данных с телефоном
  - Основан на Google Services

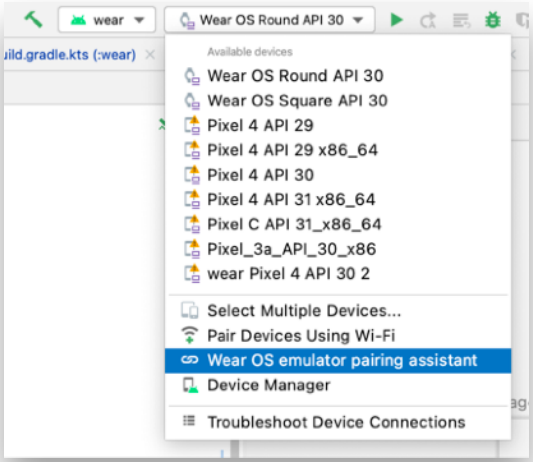


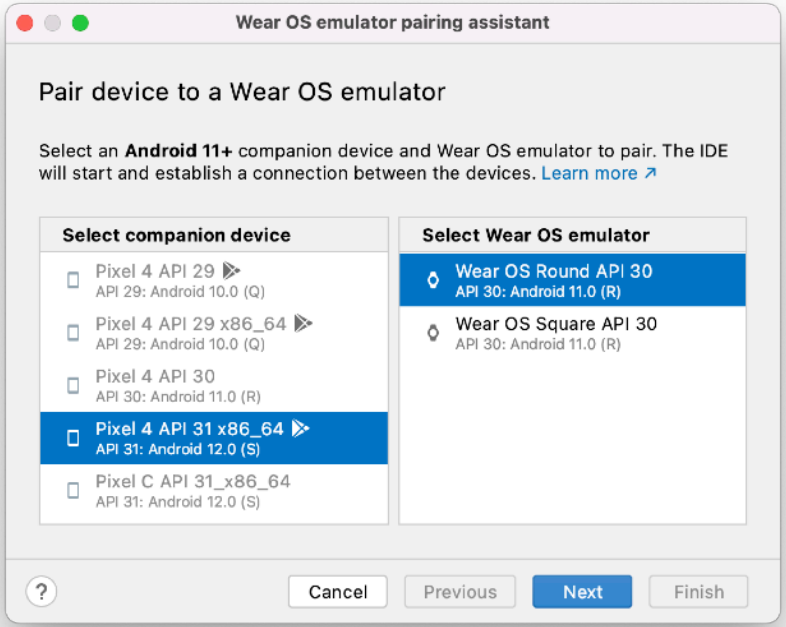
Send and sync data on Wear OS

Пример DataLayerListenerService в Coffeegram

Не про Compose, но всё же

- Pairing assistant





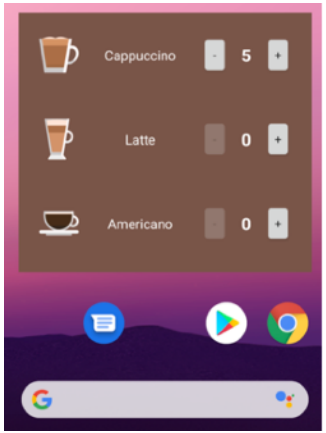
[Use the Wear OS emulator pairing assistant]

Больше про часы

- Курс с Codelabs https://developer.android.com/courses/pathways/wear
- Если тема интересна, накидайте **:watch:** ⌚ или комментарии под анонсом в Slack => подумаем над отдельным докладом

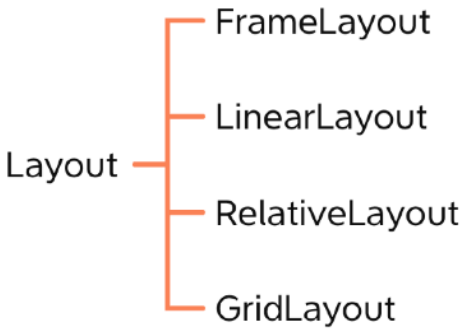
С часами всё, теперь виджеты

Что же такое виджет



- По своей сути - BroadcastReceiver
- Запускается в Remote процессе = AppWidgetHost (e.g. HomeScreenLauncher)
- Использует особый вид View = RemoteView

- Stateless
- В режиме энергосбережения могут не обновляться данные (WorkManager)

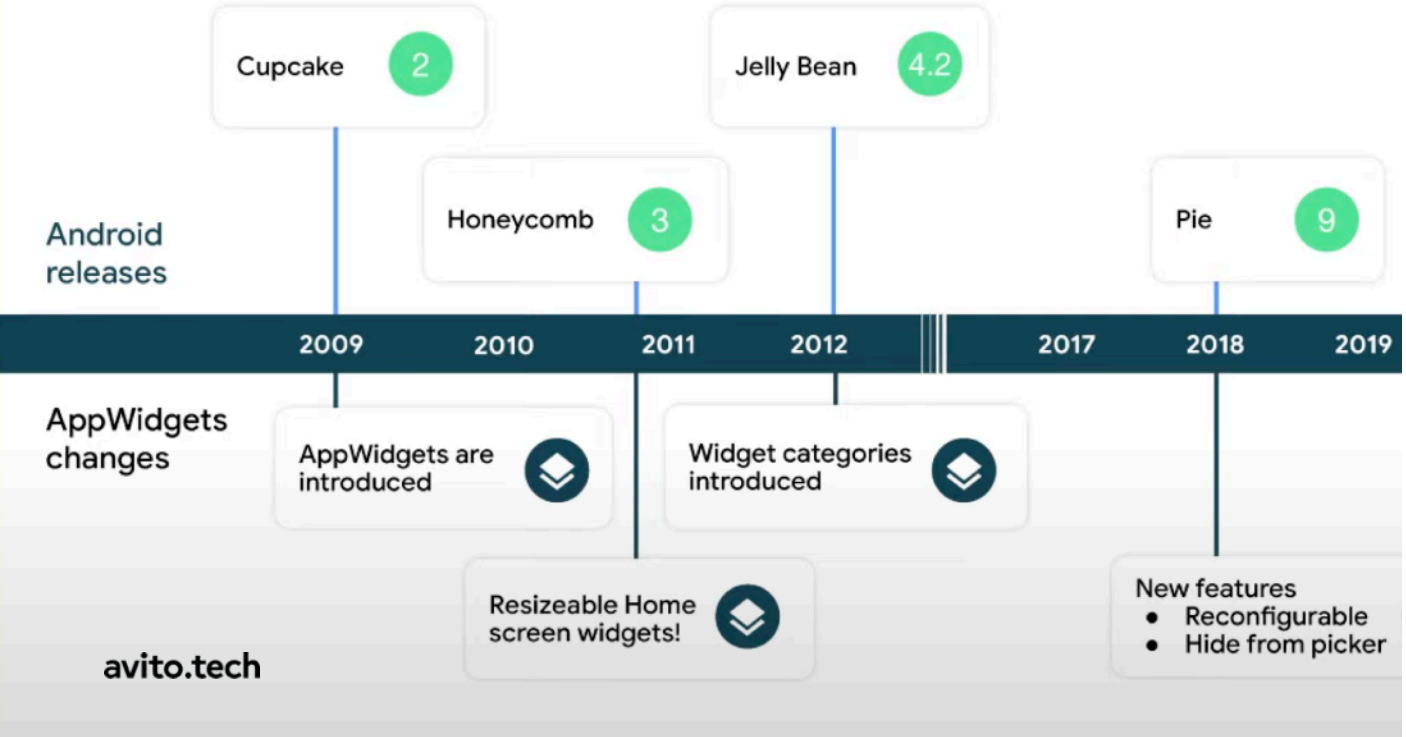


Улучшения в виджетах

https://youtu.be/15Q7xqxBGG0

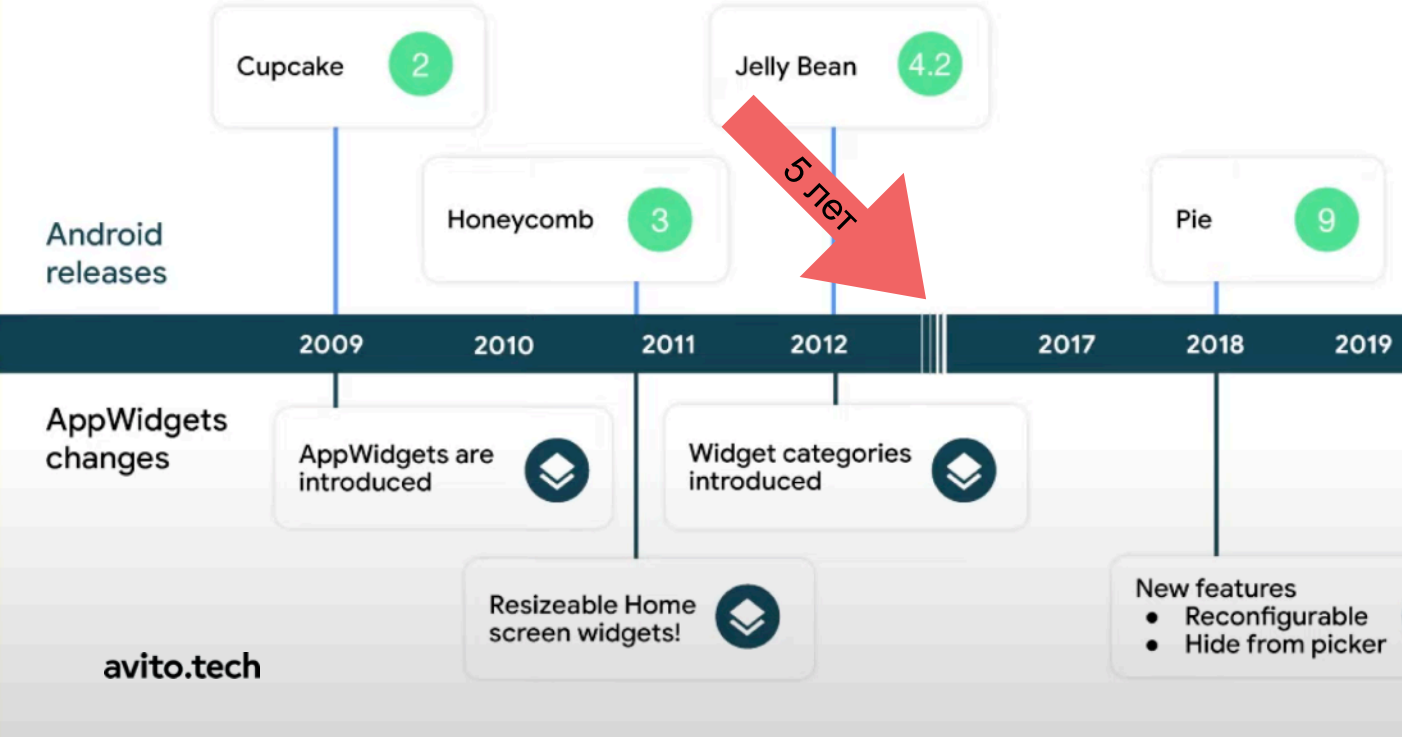
Улучшения в виджетах

https://youtu.be/15Q7xqxBGG0



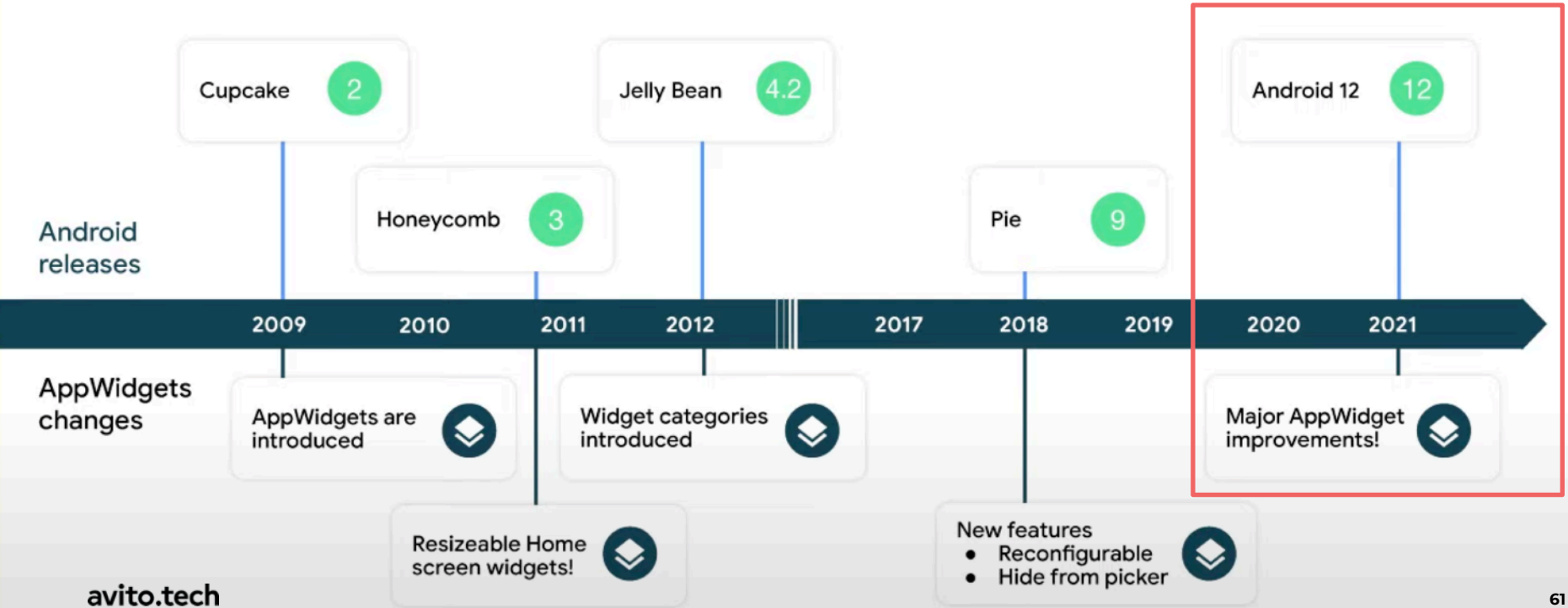
Улучшения в виджетах

https://youtu.be/15Q7xqxBGG0



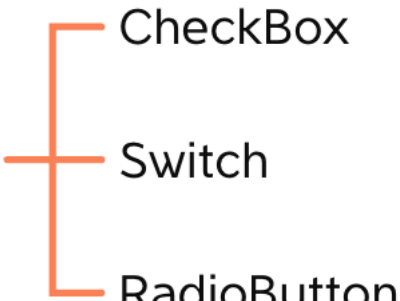
Улучшения в виджетах

https://youtu.be/15Q7xqxBGG0



Улучшения в Android 12

- Внешние изменения (углы) + поддержка системной темы
- Поддержка state-like поведения с новыми View
- Виджеты всё ещё stateless. Нужно сохранять состояние в приложении и подписываться на изменения в виджете.
- API для адаптивных макетов (Sizemodes)



[Android 12 widgets improvements]

Как сделать виджет

```
// AndroidManifest
<receiver
    android:name=".widget.WidgetReceiver"
    android:enabled="true"
    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/widget_info" />
</receiver>
```


Commit "Add sample widget" in Coffeegram

Декларация виджета

```
<!- - widget_info.xml - - >
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"

    // Общее
    android:widgetCategory="home_screen|keyguard|searchbox"
    android:initialLayout="@layout/widget_loading"
    android:resizeMode="horizontal|vertical"
```

Декларация виджета

```
<!- - widget_info.xml - - >
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"

    // Общее
    android:widgetCategory="home_screen|keyguard|searchbox"
    android:initialLayout="@layout/widget_loading"
    android:resizeMode="horizontal|vertical"

    // До Android 12
    android:minWidth="50dp"
    android:minHeight="50dp"

    android:previewImage="@drawable/logo_splash"
```

```
// Замена в Android 12
android:minResizeHeight="50dp"
android:minResizeWidth="50dp"
android:maxResizeHeight="1000dp"
android:maxResizeWidth="1000dp"

android:previewLayout="@layout/widget_preview"
```

Декларация виджета

```
<!- - widget_info.xml - - >
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"

    // Общее
    android:widgetCategory="home_screen|keyguard|searchbox"
    android:initialLayout="@layout/widget_loading"
    android:resizeMode="horizontal|vertical"

    // До Android 12
    android:minWidth="50dp"
    android:minHeight="50dp"

    android:previewImage="@drawable/logo_splash"

    // Замена в Android 12
    android:minResizeHeight="50dp"
    android:minResizeWidth="50dp"
    android:maxResizeHeight="1000dp"
    android:maxResizeWidth="1000dp"

    android:previewLayout="@layout/widget_preview"

    // Только с Android 12
    android:description="@string/widget_description"
    android:targetCellWidth="3"
    android:targetCellHeight="3"
/>
```

Old way

```
class WidgetReceiver : AppWidgetProvider() {
    override fun onUpdate(
        context: Context?,
        appWidgetManager: AppWidgetManager?,
        appWidgetIds: IntArray?
    ) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        val intent = Intent(context, SplashActivity::class.java)
        val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0)
        appWidgetIds?.forEach { appWidgetId →
            val views = RemoteViews(
                context!!.packageName,
                R.layout.app_widget
            )
            views.setOnClickPendingIntent(R.id.count, pendingIntent)
            ...
            appWidgetManager?.updateAppWidget(appWidgetId, views);
        }
    }
}
```

Old way

```
class WidgetReceiver : AppWidgetProvider() {
    override fun onUpdate(
        context: Context?,
        appWidgetManager: AppWidgetManager?,
        appWidgetIds: IntArray?
    ) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        val intent = Intent(context, SplashActivity::class.java)
        val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0)
        appWidgetIds?.forEach { appWidgetId →
            val views = RemoteViews(
                context!!.packageName,
                R.layout.app_widget
            )
            views.setOnClickPendingIntent(R.id.count, pendingIntent)
            ...
            appWidgetManager?.updateAppWidget(appWidgetId, views);
        }
    }
}
```

Old way

```
class WidgetReceiver : AppWidgetProvider() {
    override fun onUpdate(
        context: Context?,
        appWidgetManager: AppWidgetManager?,
        appWidgetIds: IntArray?
    ) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        val intent = Intent(context, SplashActivity::class.java)
        val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0)
        appWidgetIds?.forEach { appWidgetId →
            val views = RemoteViews(
                context!!.packageName,
                R.layout.app_widget
            )
            views.setOnClickPendingIntent(R.id.count, pendingIntent)
            ...
            appWidgetManager?.updateAppWidget(appWidgetId, views);
        }
    }
}
```

Old way

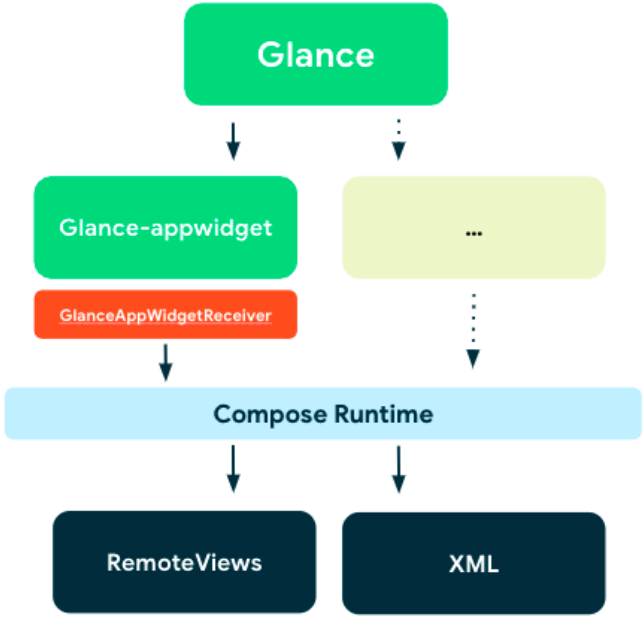
```
class WidgetReceiver : AppWidgetProvider() {
    override fun onUpdate(
        context: Context?,
        appWidgetManager: AppWidgetManager?,
        appWidgetIds: IntArray?
    ) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        val intent = Intent(context, SplashActivity::class.java)
        val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0)
        appWidgetIds?.forEach { appWidgetId ->
            val views = RemoteViews(
                context!!.packageName,
                R.layout.app_widget
            )
            views.setOnClickPendingIntent(R.id.count, pendingIntent)
            ...
            appWidgetManager?.updateAppWidget(appWidgetId, views);
        }
    }
}
```

Old way

```
class WidgetReceiver : AppWidgetProvider() {
    override fun onUpdate(
        context: Context?,
        appWidgetManager: AppWidgetManager?,
        appWidgetIds: IntArray?
    ) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        val intent = Intent(context, SplashActivity::class.java)
        val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0)
        appWidgetIds?.forEach { appWidgetId →
            val views = RemoteViews(
                context!!.packageName,
                R.layout.app_widget
            )
            views.setOnClickPendingIntent(R.id.count, pendingIntent)
            ...
            appWidgetManager?.updateAppWidget(appWidgetId, views);
        }
    }
}
```

Jetpack Glance

- Транслирует Composables в RemoteViews
  - `Box, Row, Column, Text, Button, LazyColumn, Image, Spacer`
  - **`GlanceModifier`**
- Interop with RemoteViews
  - `AndroidRemoteViews(RemoteView)`




```
androidx.glance:glance-appwidget:1.0.0-alpha02
```

[Announcing Jetpack Glance Alpha for app widgets]

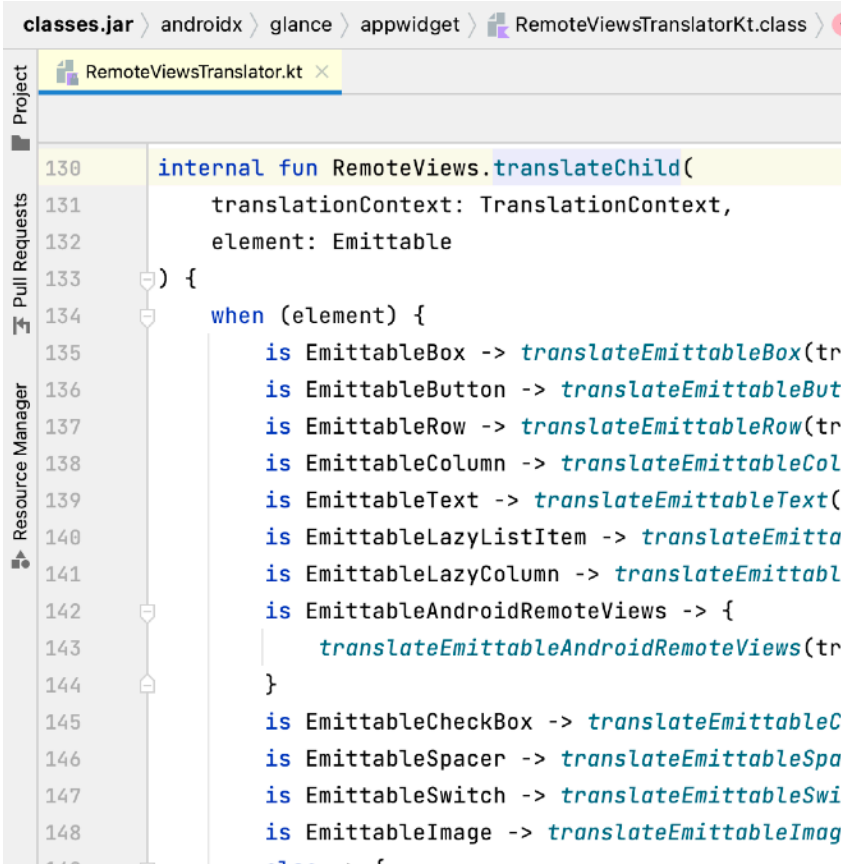
Actions

- Обертки над интентами
- `actionStartActivity`
- `actionStartService`
- `actionSendBroadcast`
- `actionRunCallback`

```
Button(
    text = "-",
    enabled = ...,
    modifier = GlanceModifier...,
    onClick = actionStartActivity
                <MainActivity>()
)
```

Трансляция Composable в RemoteView

- Composable API
- По факту всё транслируется в RemoteViews
- Actions достаются из Modifier и превращаются в ClickListeners
- Затем RemoteViews сохраняются в DataStore Proto



avito.tech

[Мой доклад про DataStore]

Как сделать виджет (New Way)

```
class WidgetReceiver : GlanceAppWidgetReceiver() {
    override val glanceAppWidget: GlanceAppWidget = FirstGlanceWidget()
}
```

Commit "Add sample widget" in Coffeegram

Как сделать виджет (New Way)

```
class FirstGlanceWidget : GlanceAppWidget() {

    @Composable
    override fun Content() {
        Column(
            modifier = GlanceModifier
                .fillMaxSize()
                .background(color = Color.White)
                .padding(8.dp)
        ) {
            Text(
                text = "First Glance widget",
                modifier = GlanceModifier.fillMaxWidth(),
            )
        }
    }
}
```

Commit "Add sample widget" in Coffeegram

Actions

- Обертки над интентами
- `actionStartActivity`
- `actionStartService`
- `actionSendBroadcast`
- `actionRunCallback`

```
Button(
    text = "-",
    enabled = ...,
    modifier = GlanceModifier...,
    onClick = actionStartActivity
                <MainActivity>()
)
```

ActionCallback

```
class SomeAction : ActionCallback {
    override suspend fun onRun(context: Context, glanceId: GlanceId,
                               parameters: ActionParameters) {
        parameters[SOME_KEY]?.let {
            ...
        }

        Widget().update(context, glanceId)
        //Widget().updateAll(context)
    }
}
```

actionRunCallback

```
Button(
    text = "+",
    onClick = actionRunCallback<SomeAction>(
        actionParametersOf(SOME_KEY to smth)
    )
)

Image(
    ...
    modifier = GlanceModifier
        .clickable(actionRunCallback<SomeAction>())
)
```

Другие возможности GlanceAppWidget

```
public abstract class GlanceAppWidget(
    @LayoutRes
    private val errorUiLayout: Int = R.layout.glance_error_layout
) {
    @Composable
    public abstract fun Content()

    public open val sizeMode: SizeMode = SizeMode.Single

    public open val stateDefinition: GlanceStateDefinition<*>? = null
}
```

GlanceAppWidget - Content ✅

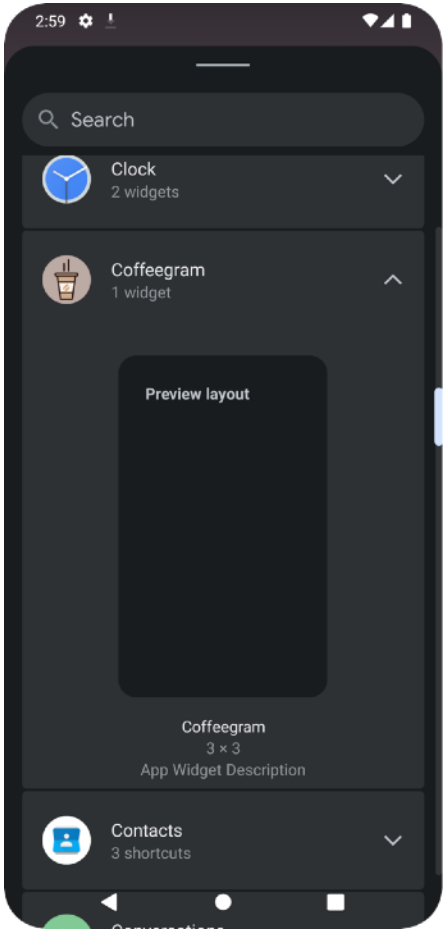
```
public abstract class GlanceAppWidget(
    @LayoutRes
    private val errorUiLayout: Int = R.layout.glance_error_layout
) {
    @Composable
    public abstract fun Content()

    public open val sizeMode: SizeMode = SizeMode.Single

    public open val stateDefinition: GlanceStateDefinition<*>? = null
}
```

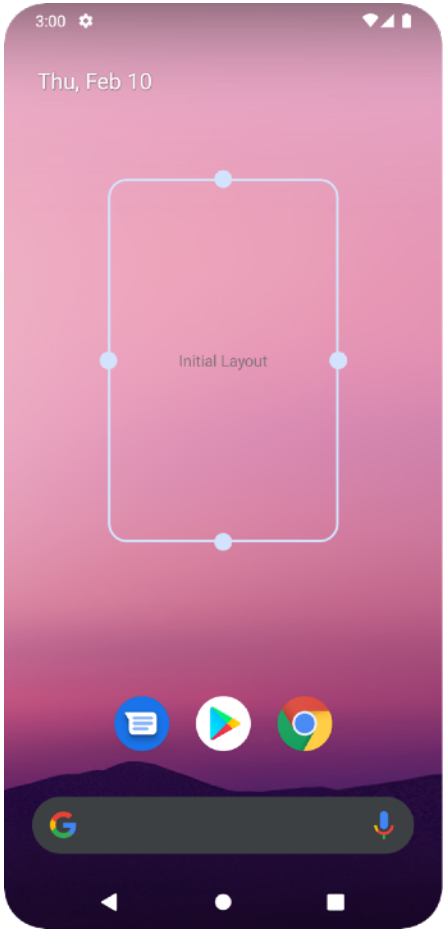
GlanceAppWidget - errorUiLayout

```
public abstract class GlanceAppWidget(
    @LayoutRes
    private val errorUiLayout: Int = R.layout.glance_error_layout
) {
    @Composable
    public abstract fun Content()

    public open val sizeMode: SizeMode = SizeMode.Single

    public open val stateDefinition: GlanceStateDefinition<*>? = null
}
```

XML

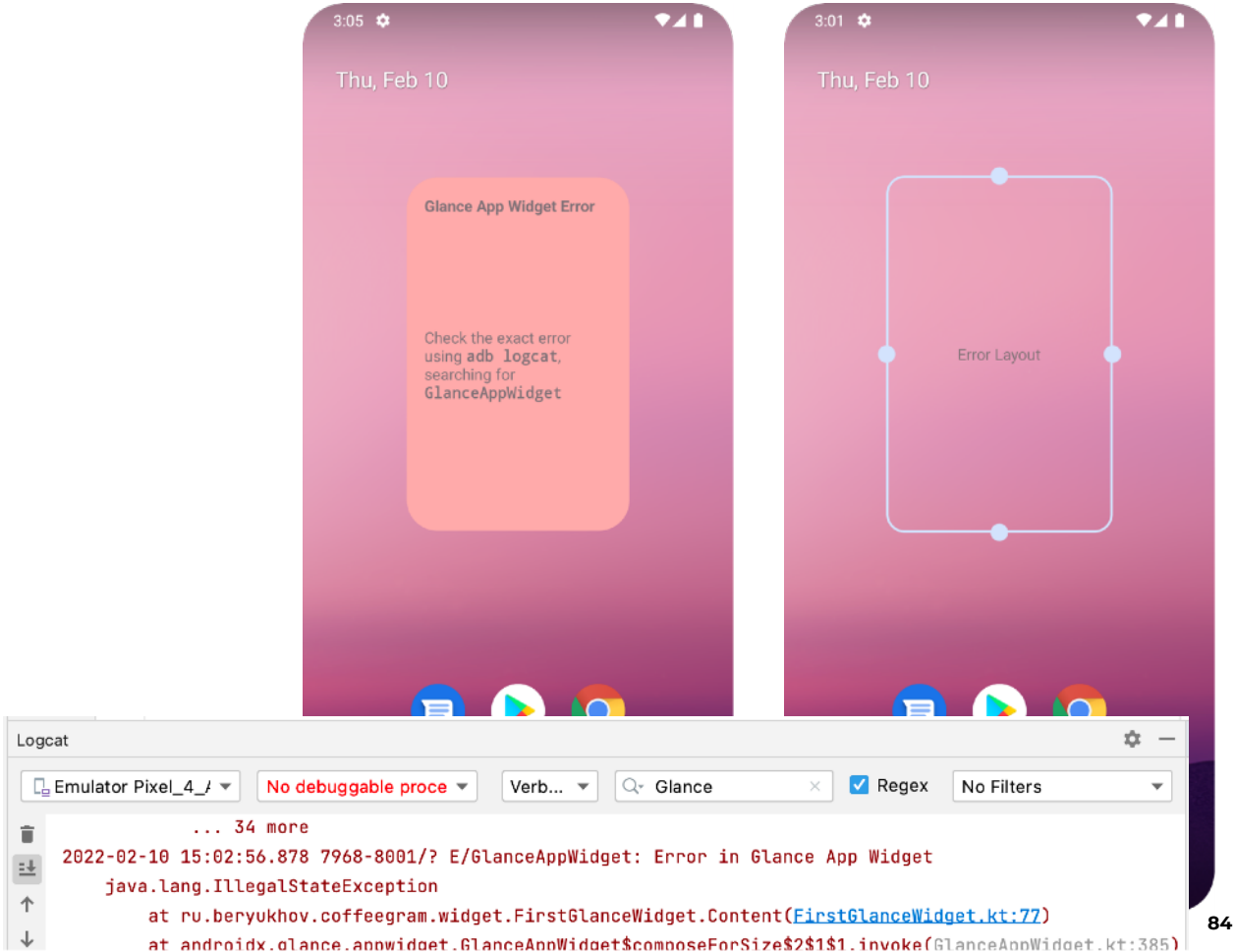
- widget_info.xml
  - android:previewLayout
  - android:initialLayout





XML

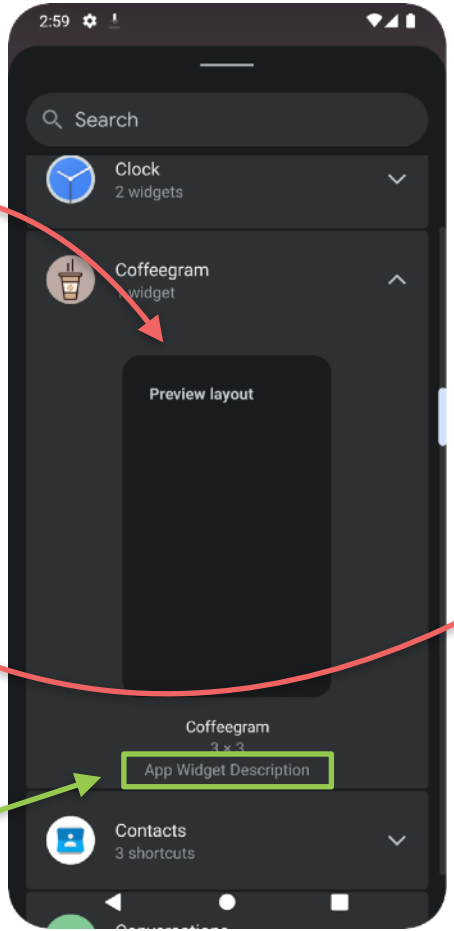
- GlanceAppWidget
  - errorUiLayout

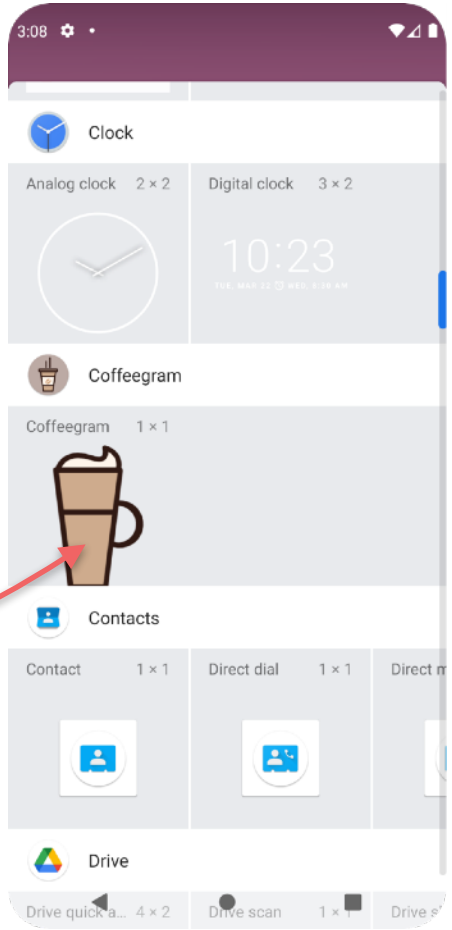


PNG

Android 12, XML

- widget_info.xml
  - android:previewLayout
  - android:previewImage





Android <12, PNG

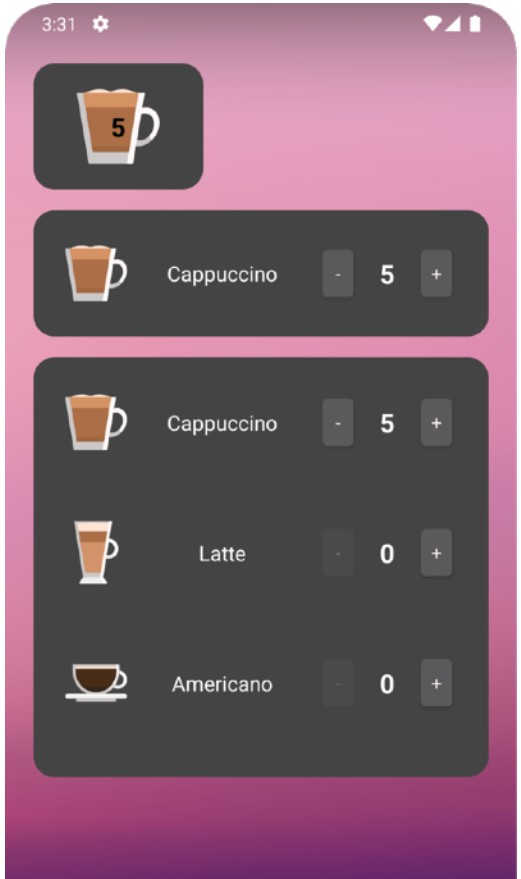
+ Android 12 - Description

GlanceAppWidget - SizeMode

```
public abstract class GlanceAppWidget(
    @LayoutRes
    private val errorUiLayout: Int = R.layout.glance_error_layout
) {
    @Composable
    public abstract fun Content()

    public open val sizeMode: SizeMode = SizeMode.Single

    public open val stateDefinition: GlanceStateDefinition<*>? = null
}
```

Sizemodes (Android 12)

- Разные лейауты в зависимости от доступного размера



Sizemodes (Android 12)

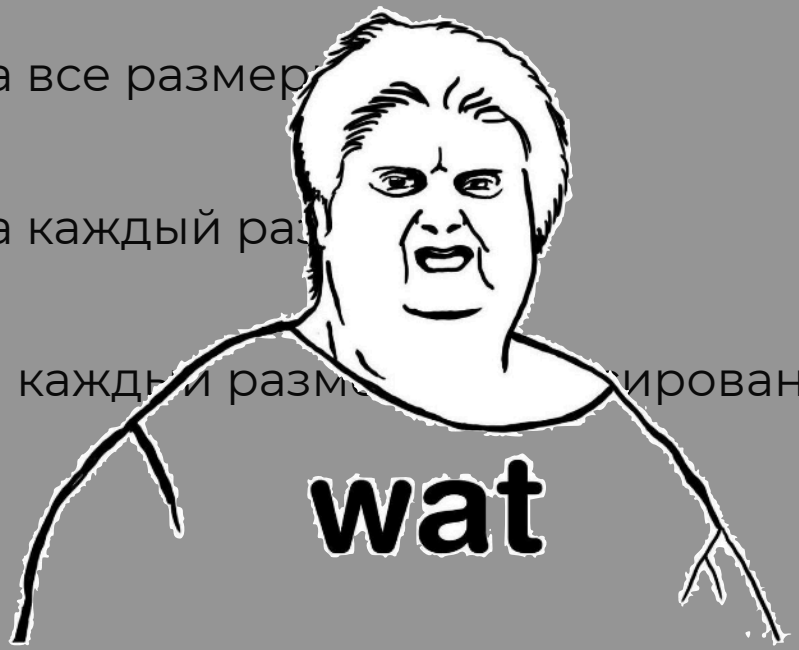
- Single
  - Один UI на все размеры
- Exact
  - Один UI на каждый размер
- Responsive
  - Свой UI на каждый размер из фиксированного набора

https://developer.android.com/reference/kotlin/androidx/glance/appwidget/SizeMode

https://developer.android.com/guide/topics/appwidgets/layouts

Sizemodes (Android 12)

- Single
  - Один UI на все размер
- Exact
  - Один UI на каждый ра
- Responsive
  - Свой UI на кажды и разм ированного набора

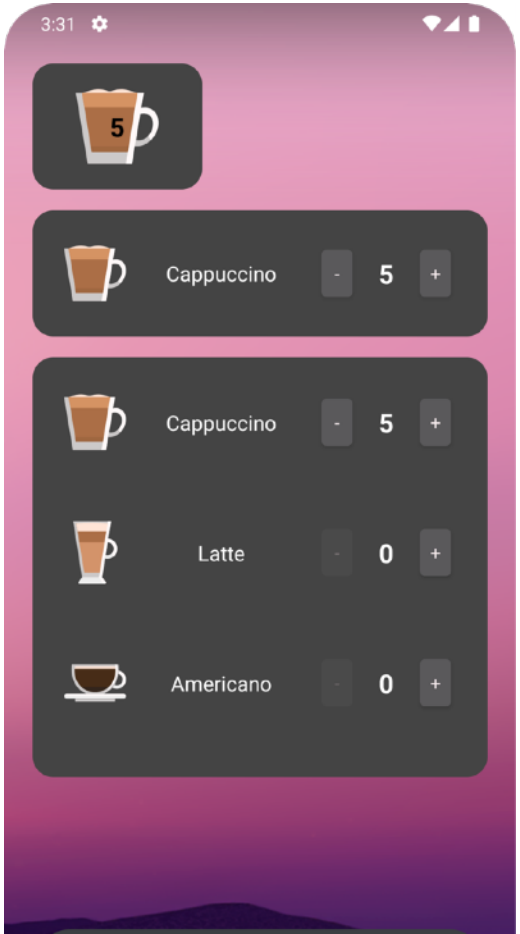


https://developer.android.com/reference/kotlin/androidx/glance/appwidget/SizeMode

https://developer.android.com/guide/topics/appwidgets/layouts

Смотрим код

```
private val SMALL_SQUARE =
                DpSize(110.dp, 110.dp)
private val HORIZONTAL_RECTANGLE =
                DpSize(270.dp, 70.dp)
private val BIG_SQUARE =
                DpSize(270.dp, 140.dp)
```



Больше кода

Смотрим код

```
@Composable
override fun Content() {
    val size = LocalSize.current
    Box(
        modifier = GlanceModifier.background(
            day = brown500,
            night = Color.DarkGray
        )
    ) {
        when {
            size.lessThan(SMALL_SQUARE) → SmallWidget()
            size.lessThan(HORIZONTAL_RECTANGLE) → HorizontalWidget()
            else → BigWidget()
}}}
```

Больше кода

Смотрим код

```
@Composable
override fun Content() {
    val size = LocalSize.current
    Box(
        modifier = GlanceModifier.background(
            day = brown500,
            night = Color.DarkGray
        )
    ) {
        when {
            size.lessThan(SMALL_SQUARE) → SmallWidget()
            size.lessThan(HORIZONTAL_RECTANGLE) → HorizontalWidget()
            else → BigWidget()
}}}
```

Больше кода

Смотрим код

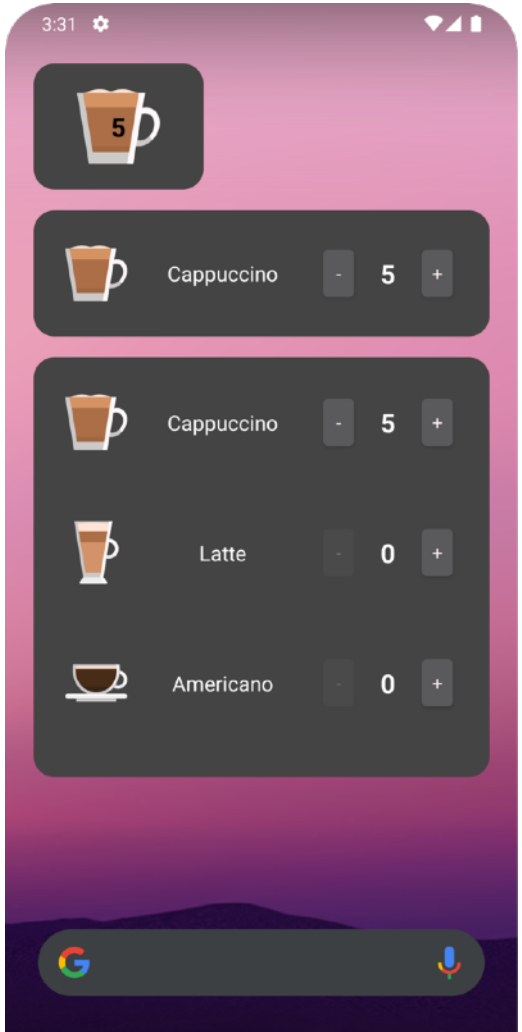
```
@Composable
override fun Content() {
    val size = LocalSize.current
    Box(
        modifier = GlanceModifier.background(
            day = brown500,
            night = Color.DarkGray
        )
    ) {
        when {
            size.lessThan(SMALL_SQUARE) → SmallWidget()
            size.lessThan(HORIZONTAL_RECTANGLE) → HorizontalWidget()
            else → BigWidget()
}}}
```

Больше кода

Responsive

```
override val sizeMode = SizeMode.Responsive(
    setOf(
        SMALL_SQUARE,
        HORIZONTAL_RECTANGLE,
        BIG_SQUARE
    )
)
```

UI будет срендерен и сохранен во время создания виджета для последующей реактивности



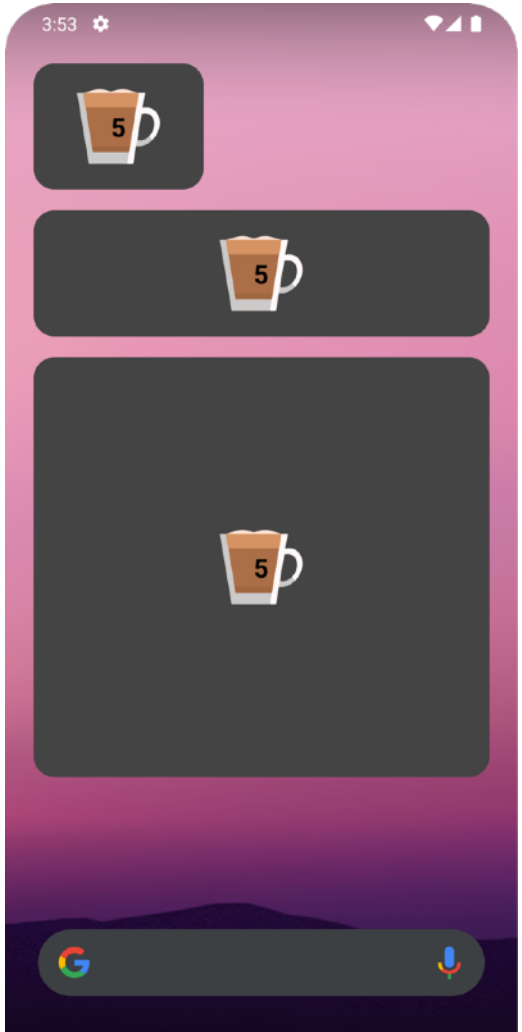
[Больше кода]

Single

```
override val sizeMode = SizeMode.Single
```

Content() вызывается один раз при создании виджета

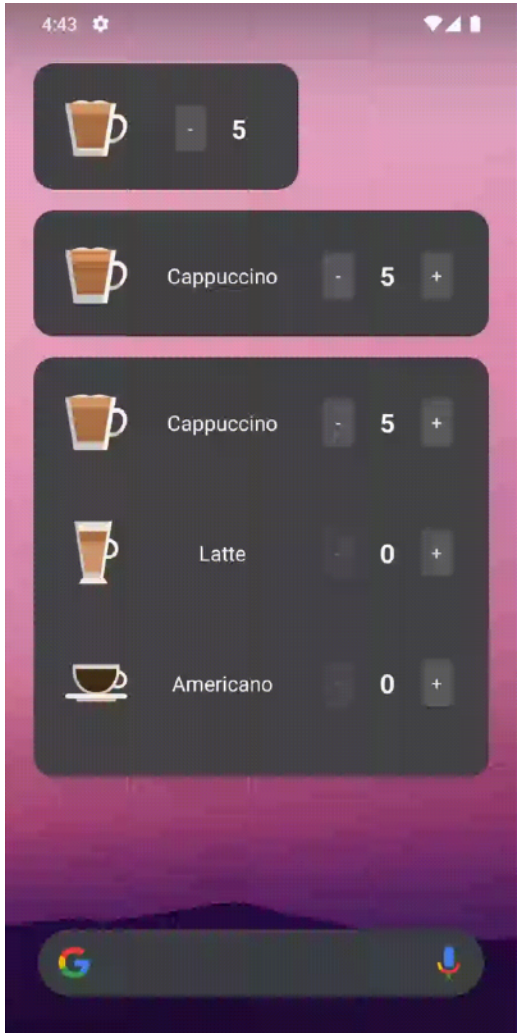
Больше кода



Exact

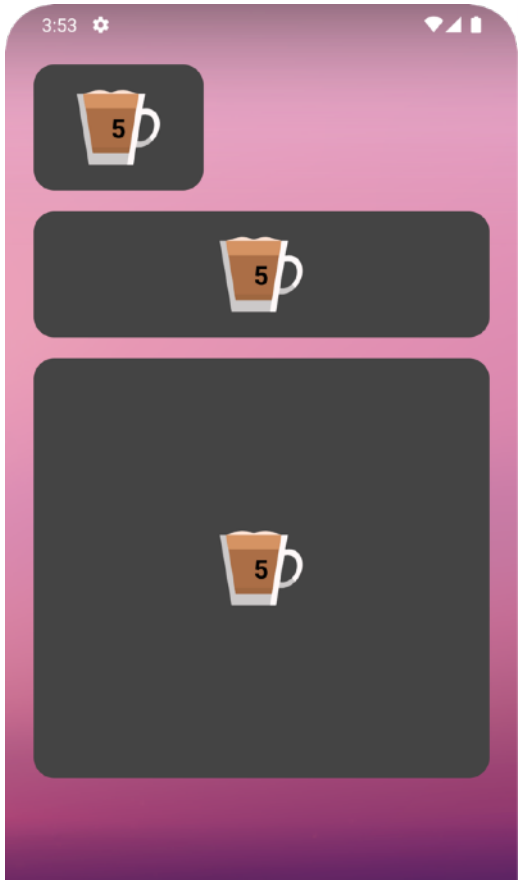
```
override val sizeMode = SizeMode.Exact
```

Content() вызывается каждый раз при смене размера

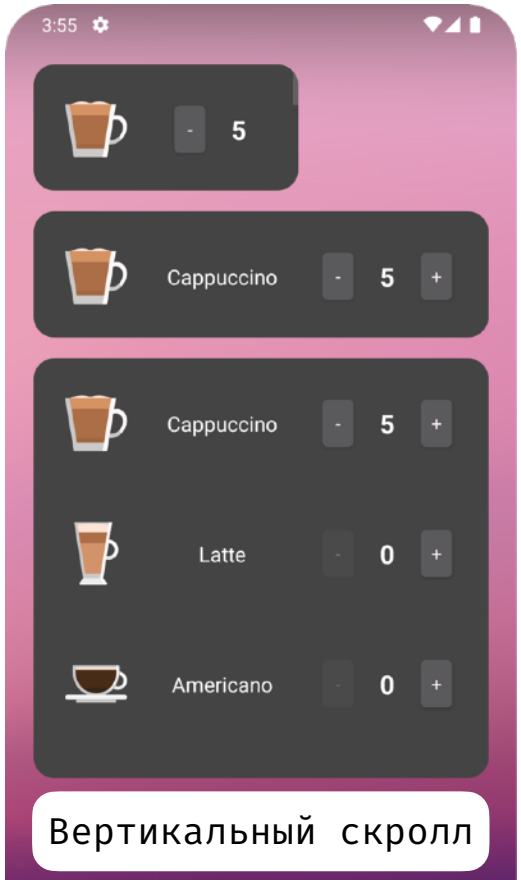


Больше кода

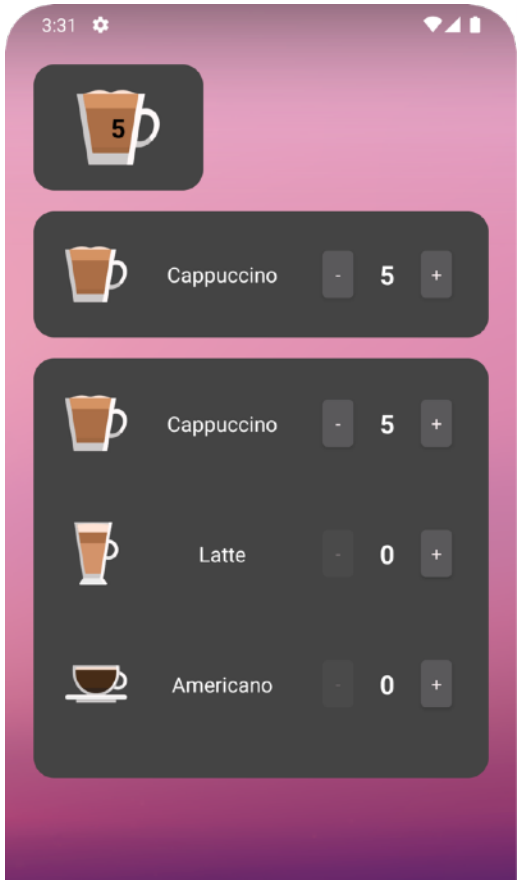
Single



Exact



Responsive



Переключение темы

```
@Composable
override fun Content() {
    val size = LocalSize.current
    Box(
        modifier = GlanceModifier.background(
            day = brown500,
            night = Color.DarkGray
        )
    ) {
        ...
}}}
```

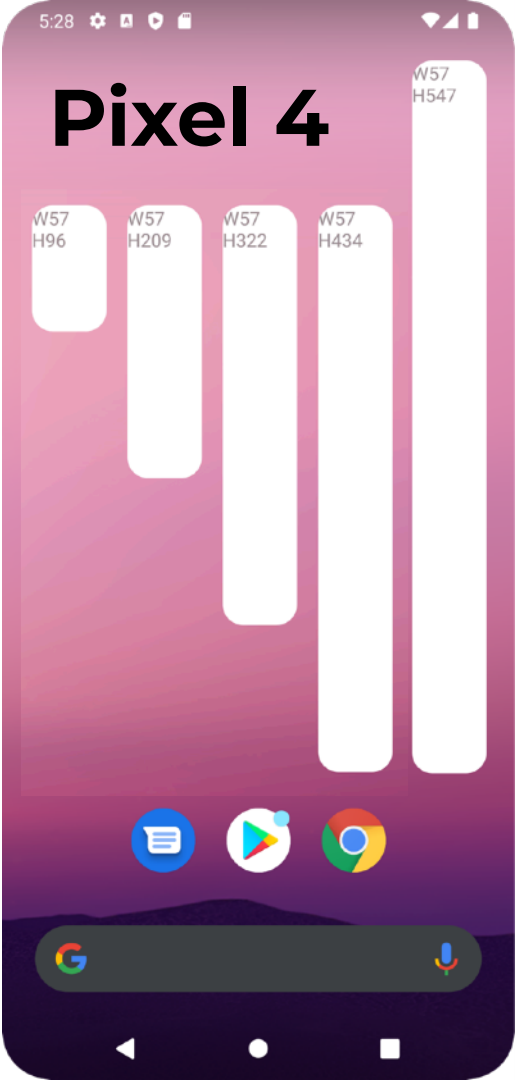
Больше кода

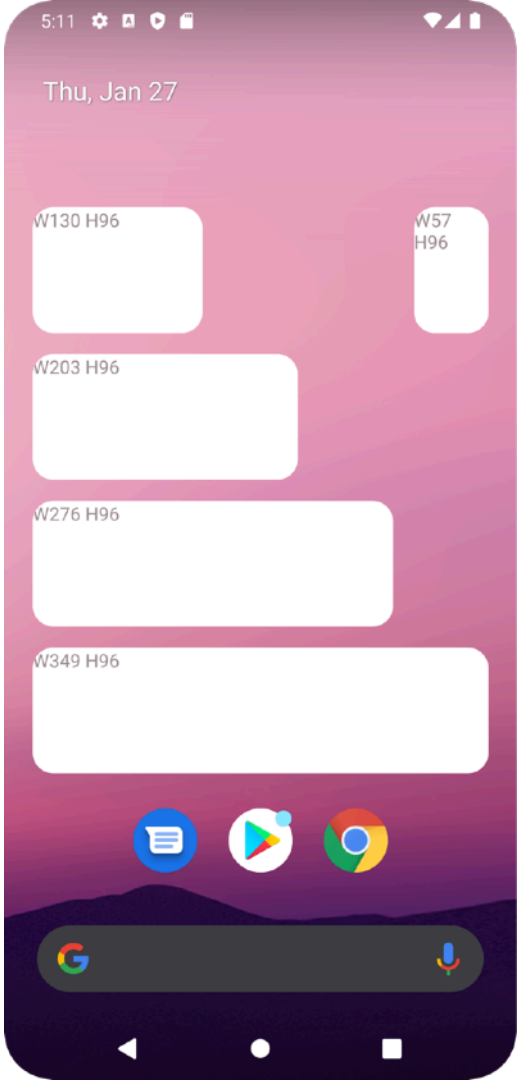
Что-то работает и до Android 12

Про вычисление количества клеток

- У клеток разные размеры в горизонтальной и вертикальной ориентациях
- Эти размеры сложно считать (из-за отступов между виджетами)
- (Недавно) добавили формулу для Pixel 4

https://developer.android.com/guide/topics/appwidgets/layouts#anatomy_determining_size


5:28
Pixel 4
W57
H547
W57
H96
W57
H209
W57
H322
W57
H434


5:11
Thu, Jan 27
W130 H96
W57
H96
W203 H96
W276 H96
W349 H96


5:24
W647 H53
W116 H53
W116 H53
W116 H53
W116 H53


5:25
W116 H333
W116 H53
W116 H53
W116 H53
W116 H53

GlanceAppWidget - StateDefinition

```
public abstract class GlanceAppWidget(
    @LayoutRes
    private val errorUiLayout: Int = R.layout.glance_error_layout
) {
    @Composable
    public abstract fun Content()

    public open val sizeMode: SizeMode = SizeMode.Single

    public open val stateDefinition: GlanceStateDefinition<*>? = null
}
```

У виджетов в Compose
есть Стейт

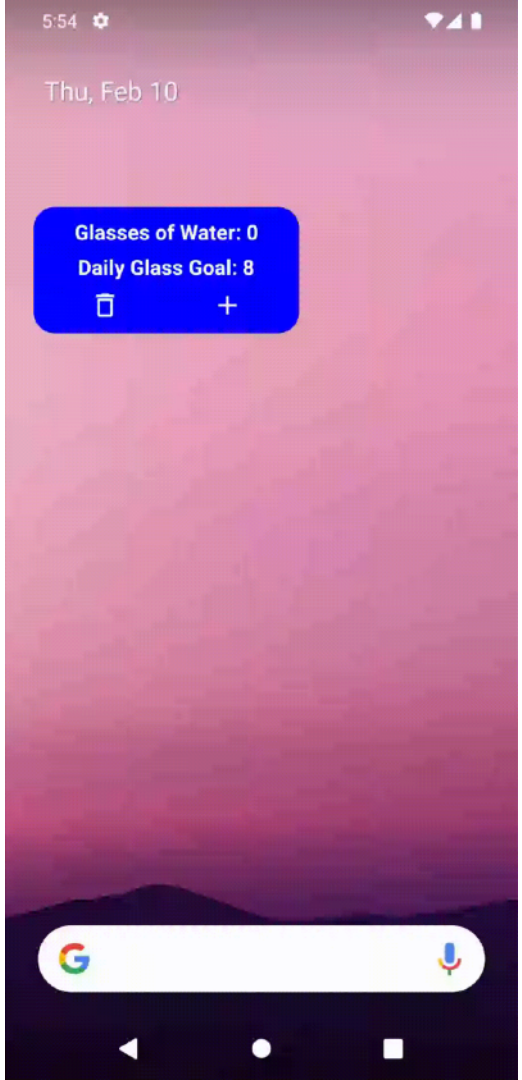
У виджета в Android
Стейта нет

Парам-парам-пам

ВСЁ!

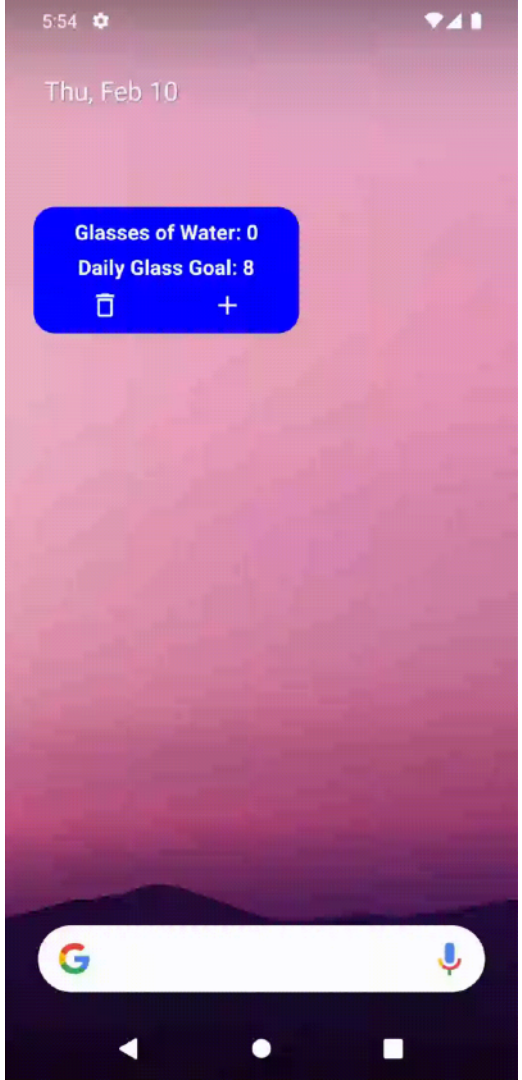
GlanceStateDefinition

```
class WaterWidget : GlanceAppWidget() {

    override val stateDefinition:
            GlanceStateDefinition<*>
    = PreferencesGlanceStateDefinition
     ...
}
```



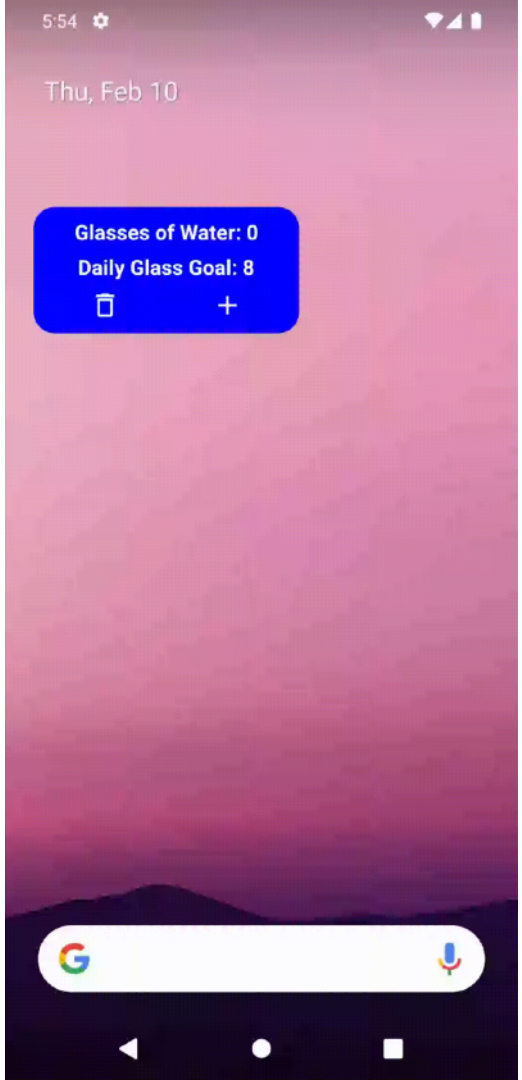
Read Preferences

```
@Composable
fun WaterWidgetText() {
    val prefs = currentState<Preferences>()
    val glassesOfWater = prefs[
            intPreferencesKey(PREFS_KEY)
    ] ?: 0
    Text("$glassesOfWater")
}
```



Read Preferences

```
@Composable
fun WaterWidgetText() {                                  Glance
    val prefs = currentState<Preferences>()
    val glassesOfWater = prefs[
        intPreferencesKey(PREFS_KEY)
    ] ?: 0                          Datastore Preferences
    Text("$glassesOfWater")
}
```



Write Preferences

```
class AddWaterClickAction : ActionCallback {
    override suspend fun onRun(
        context: Context, glanceId: GlanceId,
        parameters: ActionParameters
    ) {
        updateAppWidgetState(context, PreferencesGlanceStateDefinition, glanceId) {
            it.toMutablePreferences()
                .apply {
                    val glassesOfWater = this[intPreferencesKey(PREFS_KEY)] ?: 0
                    this[intPreferencesKey(PREFS_KEY)] = glassesOfWater + 1
                }
        }
        WaterWidget().update(context, glanceId)
    }
}
```

Write Preferences

```
class AddWaterClickAction : ActionCallback {  Glance
    override suspend fun onRun(
        context: Context, glanceId: GlanceId,
        parameters: ActionParameters
    ) {
        updateAppWidgetState(context, PreferencesGlanceStateDefinition, glanceId) {
            it.toMutablePreferences()
                .apply {
                    val glassesOfWater = this[intPreferencesKey(PREFS_KEY)] ?: 0
                    this[intPreferencesKey(PREFS_KEY)] = glassesOfWater + 1
                }
        }
        WaterWidget().update(context, glanceId)
    }
}
```

Write Preferences

```
class AddWaterClickAction : ActionCallback {
    override suspend fun onRun(
        context: Context, glanceId: GlanceId,
        parameters: ActionParameters
    ) {
        updateAppWidgetState(context, PreferencesGlanceStateDefinition, glanceId) {
            it.toMutablePreferences()
                .apply {
                    val glassesOfWater = this[intPreferencesKey(PREFS_KEY)] ?: 0
                    this[intPreferencesKey(PREFS_KEY)] = glassesOfWater + 1
                }
        }
        WaterWidget().update(context, glanceId)
    }
}
```

Datastore Preferences

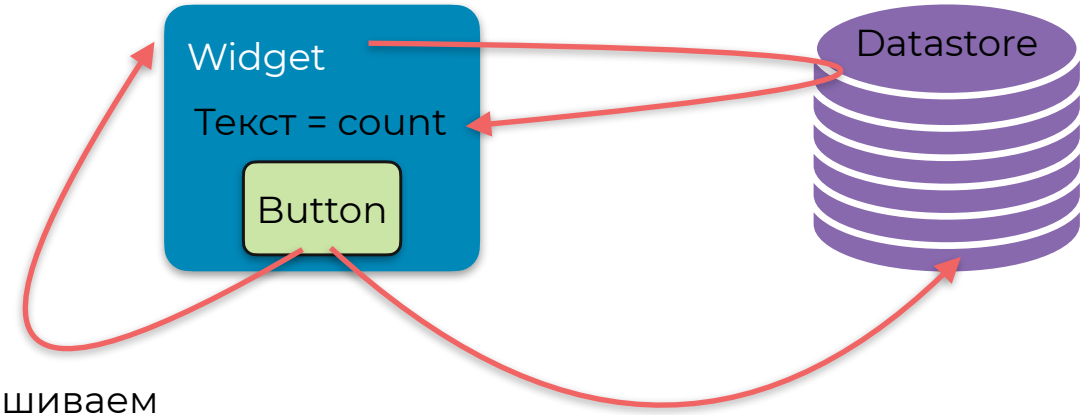
Write Preferences

```
class AddWaterClickAction : ActionCallback {
    override suspend fun onRun(
        context: Context, glanceId: GlanceId,
        parameters: ActionParameters
    ) {
        updateAppWidgetState(context, PreferencesGlanceStateDefinition, glanceId) {
            it.toMutablePreferences()
                .apply {
                    val glassesOfWater = this[intPreferencesKey(PREFS_KEY)] ?: 0
                    this[intPreferencesKey(PREFS_KEY)] = glassesOfWater + 1
                }
        }
        WaterWidget().update(context, glanceId)
    }
}
```

Glance

Как это работает?

#3
Виджет в `Content()` запрашивает
Стейт из Datastore и использует в UI

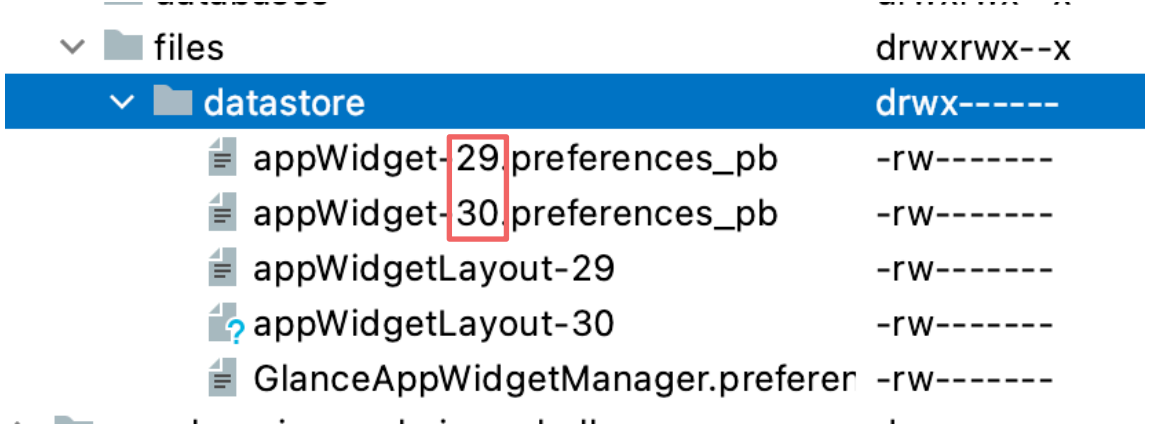


#2
Запрашиваем
обновление виджета

#1
Кладем Новый
Стейт в Datastore

У каждого виджета свой Стейт

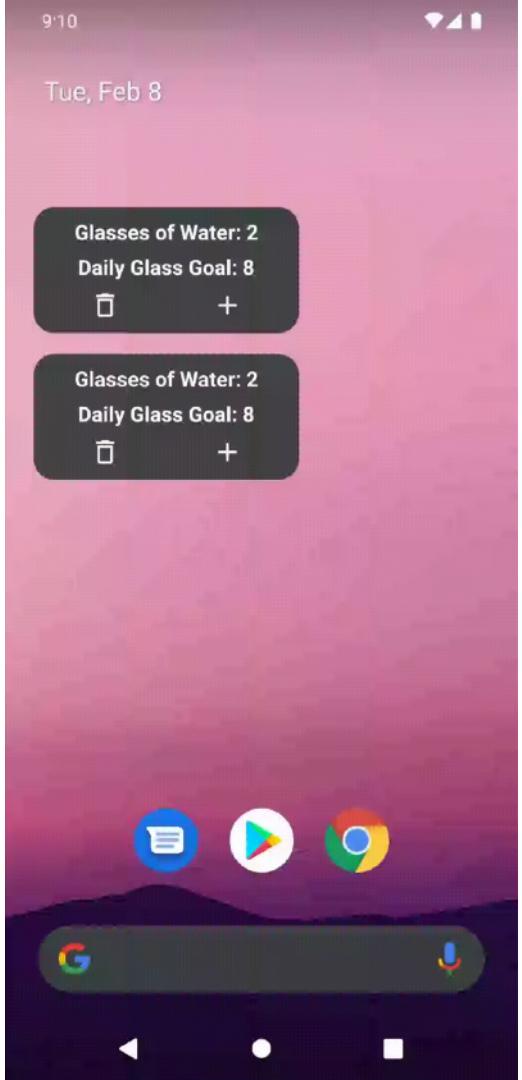
Id виджета на экране



Мой доклад про DataStore

Можно свой Datastore

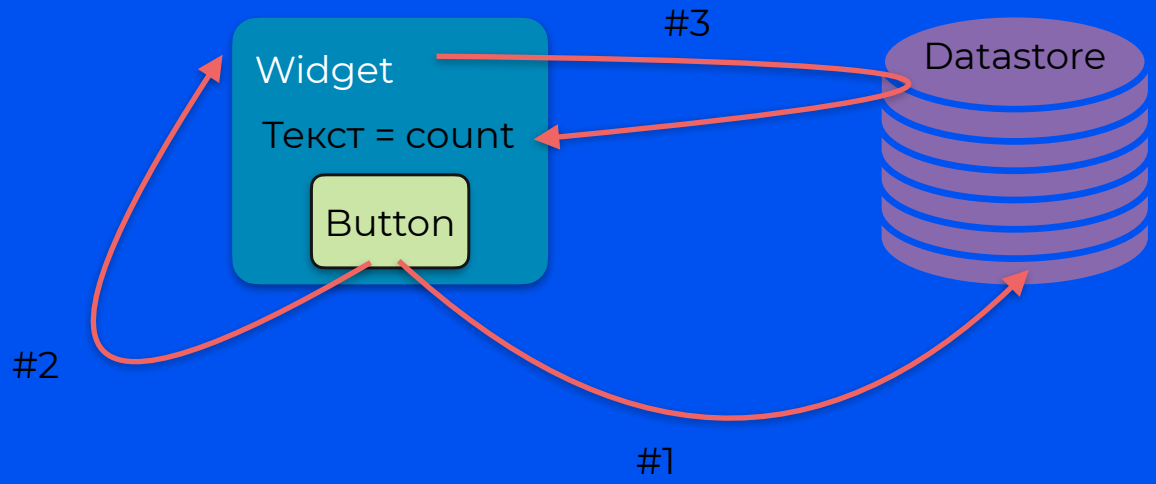
Код больше про Datastore,
чем про Glance
его можно найти по ссылке



[Замена встроенного Datastore на общий для приложения]

У виджета в Android
всё ещё нет Стейта

У виджетов в ~~Compose~~
Glance
теперь есть Стейт

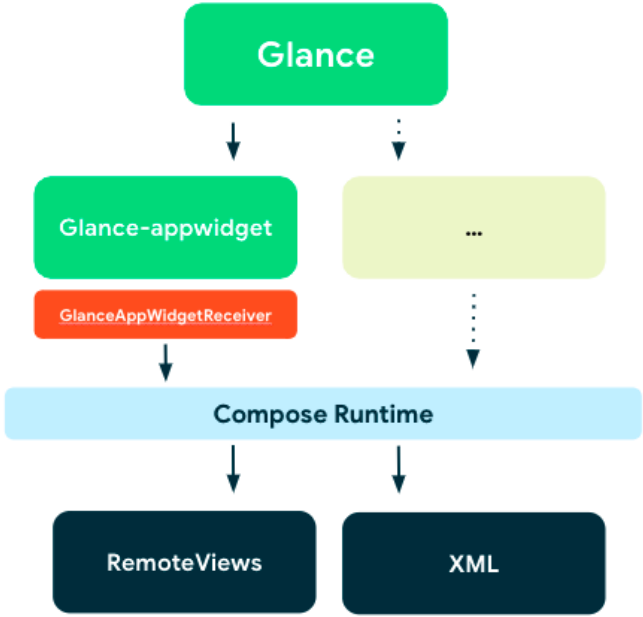


Он вам не Compose

- Никак не совместим с Compose
- `@Preview` пока не работает -> `java.lang.IllegalStateException: Invalid applier`
- Утилиты из Compose часто не работают
  - `isSystemInDarkTheme()` -> `java.lang.IllegalStateException: CompositionLocal LocalConfiguration not present`

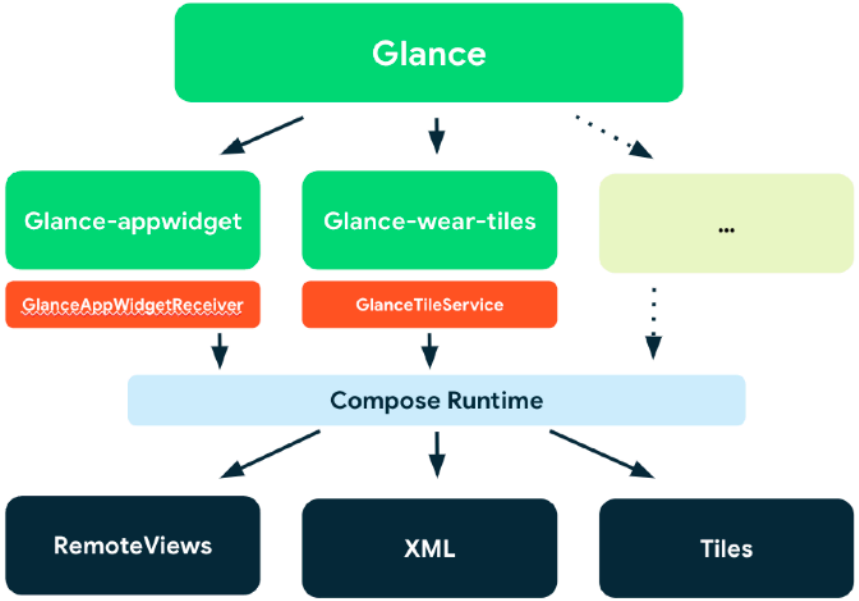
С часами не всё, возвращаемся

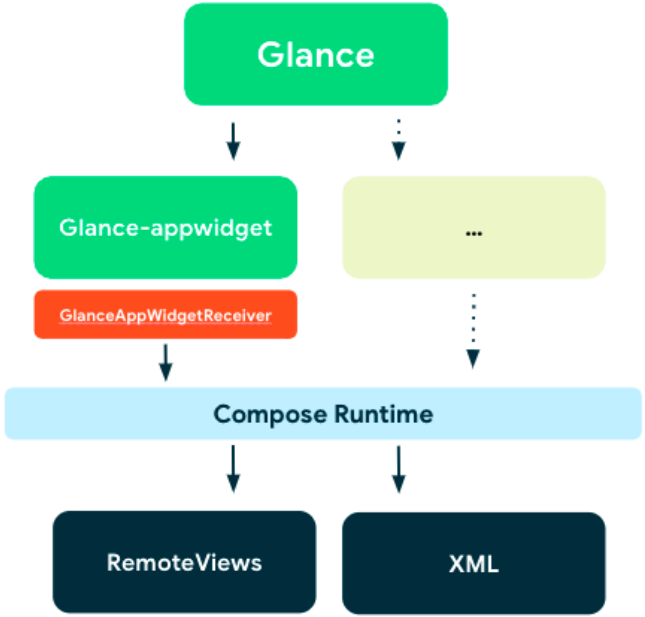
Jetpack Glance



[Announcing Jetpack Glance Alpha for app widgets]

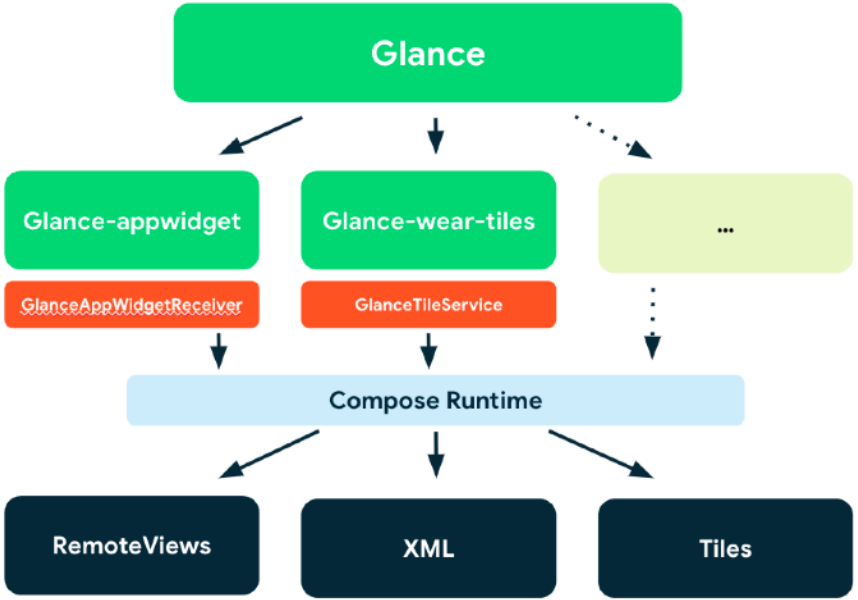
Jetpack Glance for Tiles





[Announcing Glance: Tiles for Wear OS made simple]

Jetpack Glance for Tiles



- Box, Row, Column, Text, Image, Spacer, CurvedRow, CurvedText
  - -Button, LazyColumn
  - +CurvedRow + CurvedText
- GlanceTileService
- Тоже есть GlanceStateDefinition
- Из actions только actionStartActivity()

androidx.glance:glance-wear-tiles:1.0.0-alpha02

Выводы

Android UI vs Compose + Glance

01. Activity
Phones, Watch, …

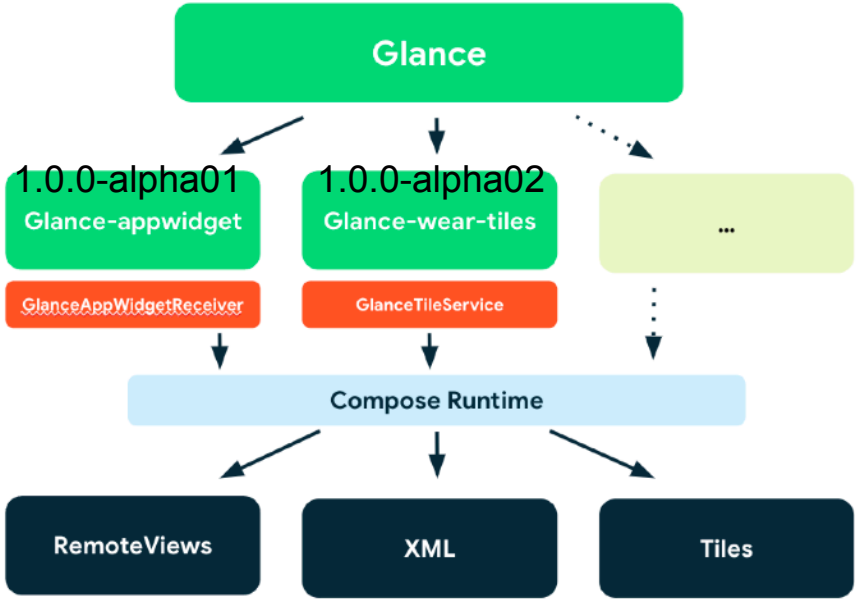
02. Remote Views

Widgets, Notifications

03. Tiles
Watch

04. Previews (XML)
Widgets

Jetpack Glance



1.0.0-alpha03

Notifications?

Замкнутый круг Wear OS

Не выпускают
Android-часы
и приложения

Пользователи

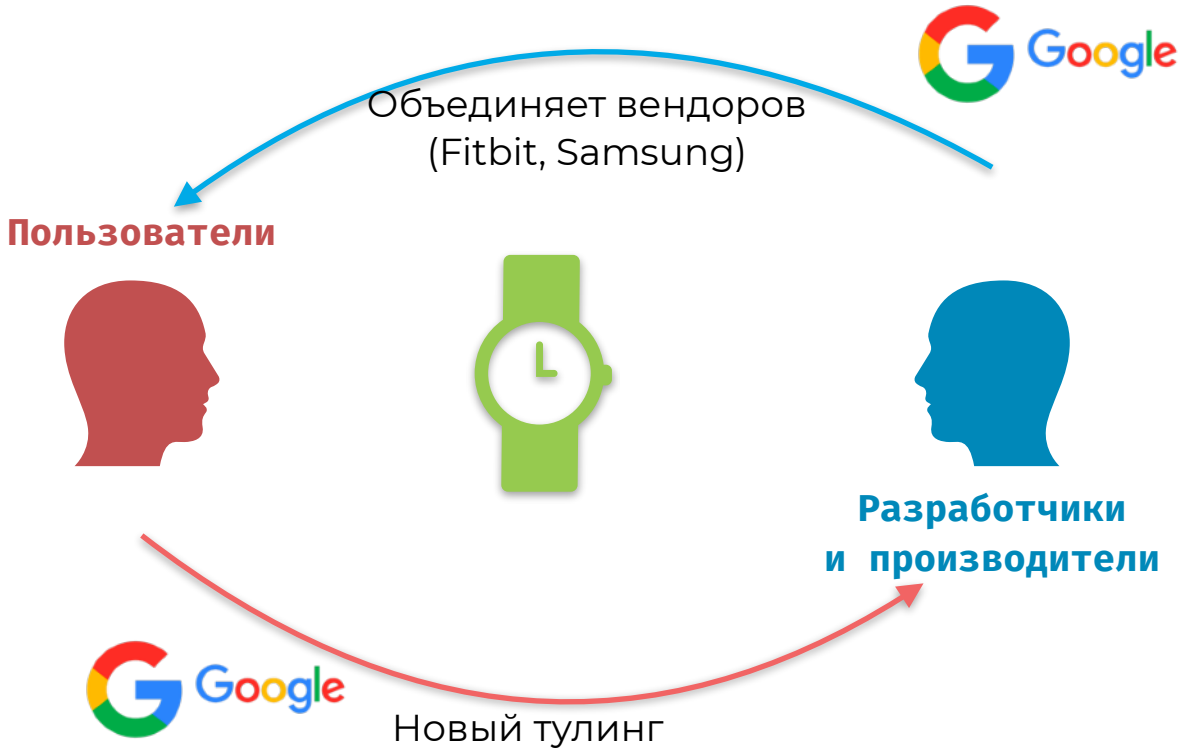
Не пользуются
Android-часами

Разработчики
и производители

так как

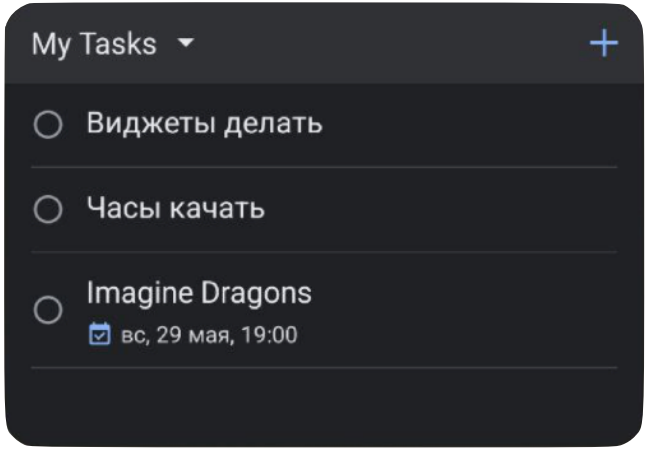

так как
Не выпускают
Android-часы
и приложения
Пользователи
Не пользуются
Android-часами
Разработчики
и производители
так как

Замкнутый круг Wear OS



Виджеты

- Android сделал первым
- Забыл про них
- iOS сделал виджеты
- Android вспомнил про виджеты



Noteit Case

Noteit - Drawing App 4+

Vitor Bukovitz

Разработано для iPhone

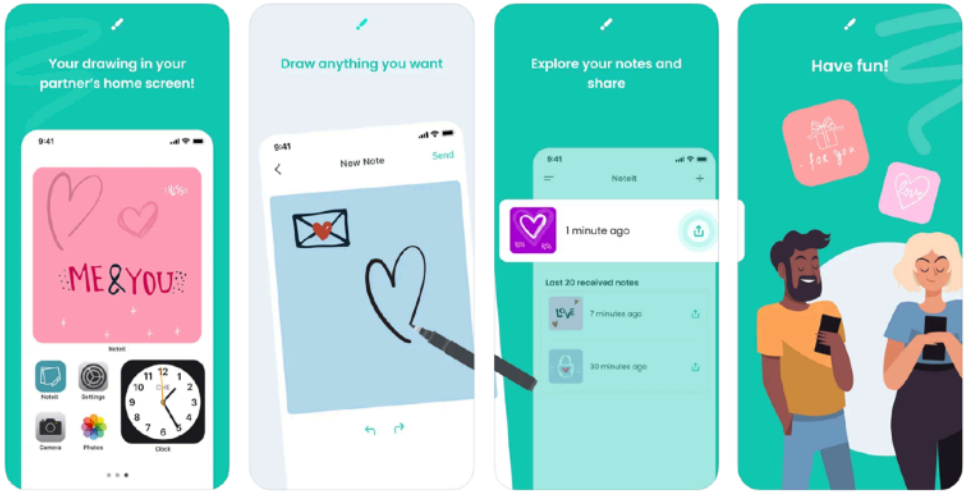
Развлечения: № 24 в этой категории

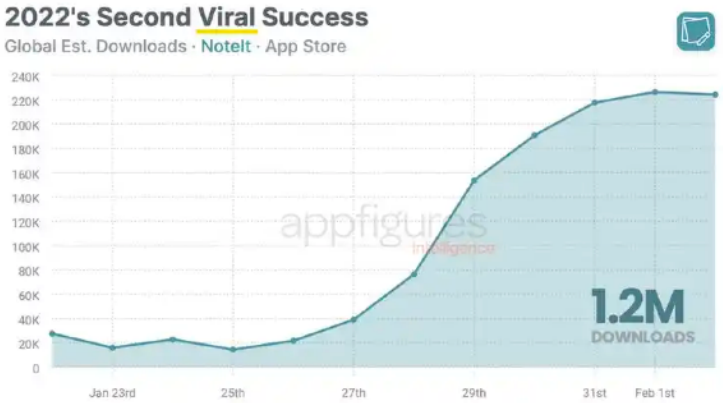
★★★★☆ 3,8 • Оценок: 13

Бесплатно

Просмотреть в Mac App Store ↗

Снимки экрана (iPhone)





https://apptractor.ru/measure/app-store-analytics/noteit.html

Виджеты

- Можно найти новые пользовательские сценарии
- Разработка станет удобнее и понятнее

Зачем это всё

- Потренировать Compose на часах/виджетах, если не дают в приложении
- Compose развивается, становится чем-то большим чем просто Flutter на Kotlin

avito.tech

Podlodka
Android Crew #7

Compose for Wearables & Widgets

Андрей Берюхов

https://github.com/phansier

https://t.me/phansier

Ссылки:

https://github.com/phansier/Coffeegram
- все оттенки Compose

Доклады на Podlodka:
- WorkManager
- Jetpack DataStore

beryukhov.ru - все доклады