Александр Невьянцев
TradingView

ANDROID WIDGETS

1

Что такое виджеты?

Как запускаются и как работают

ВИДЖЕТ

- Это BroadcastReceiver
- Обрабатывает события лаунчера
- Верстка в виджете реализуется с помощью класса RemoteViews

У приложения нет прямого доступа к виджету. Доступ есть только у лаунчера, т.к. он является хостом

```
<receiver android:name=".view.WatchlistWidgetProvider">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE"/>
    </intent-filter>

    <meta-data
            android:name="android.appwidget.provider"
            android:resource="@xml/watchlist_widget_info"/>
    <meta-data
            android:name="test.provider"
            android:resource="@xml/hidden_watchlist_widget_info"/>
</receiver>
```

```xml
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="250dp"
    android:minHeight="100dp"
    android:minResizeWidth="180dp"
    android:minResizeHeight="100dp"
    android:updatePeriodMillis="300000"
    android:initialLayout="@layout/layout_watchlist_widget"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen"
    android:previewImage="@drawable/widget_preview_no_logo"
    android:configure=".view.ConfigurationActivity"
    android:widgetFeatures="reconfigurable">
</appwidget-provider>
```

```
class WatchlistWidgetProvider : AppWidgetProvider() {
    override fun onUpdate(
        context: Context,
        manager: AppWidgetManager,
        appWidgetIds: IntArray
    ) {
      ...
    }

    override fun onAppWidgetOptionsChanged(
        context: Context, manager: AppWidgetManager,
        widgetId: Int, newOptions: Bundle
    ) {
      ...
    }
}
```

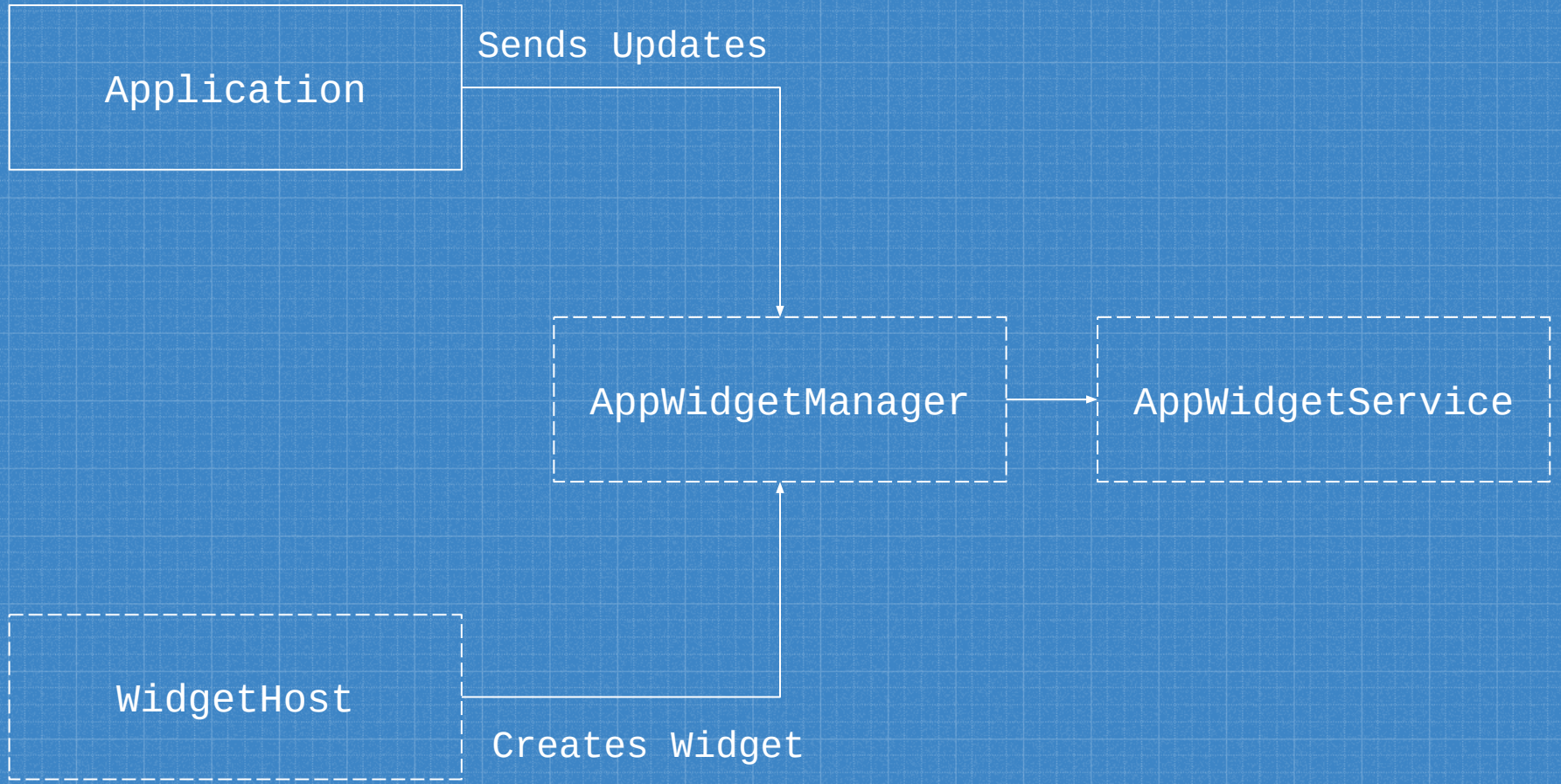
AppWidgetProvider

- onEnabled
- onDisabled
- onDeleted
- onWidgetOptionsChanged
- onUpdate
- onRestored
- onReceive

2

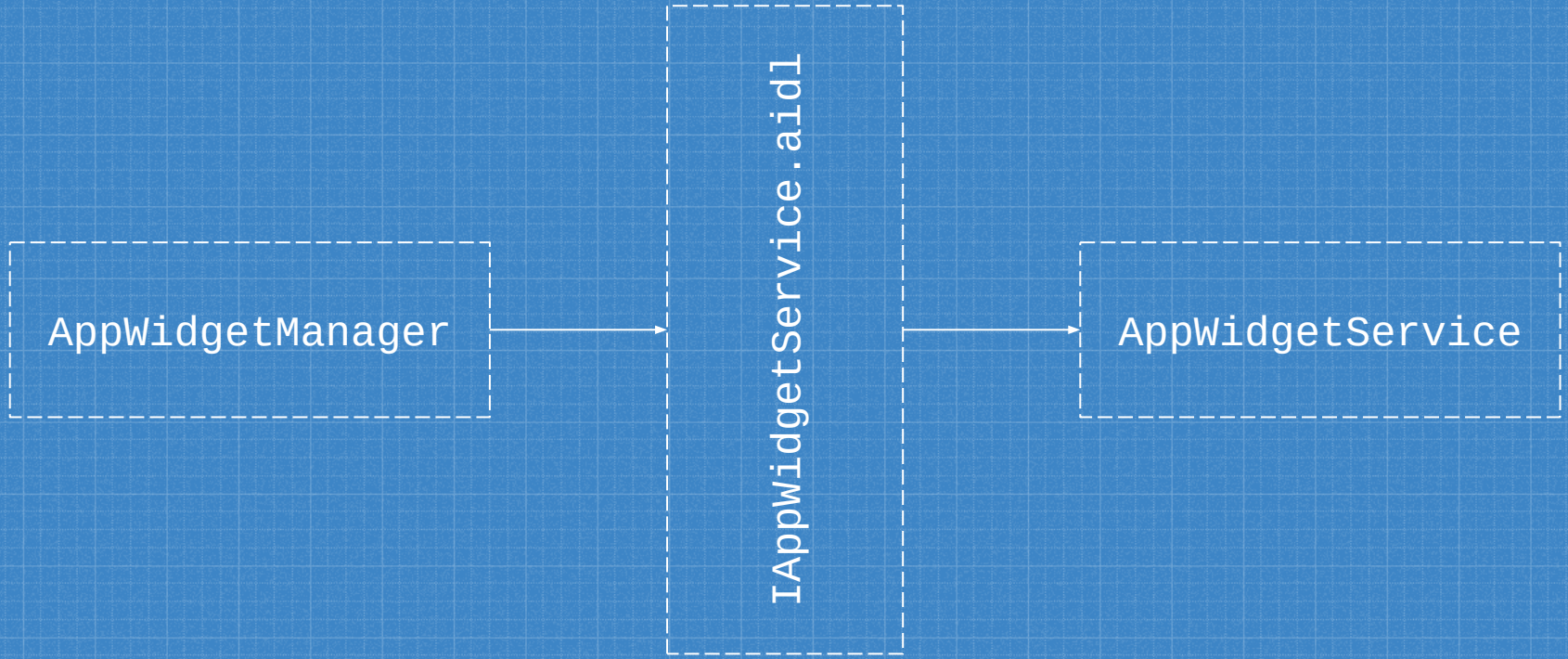
Взгляд изнутри

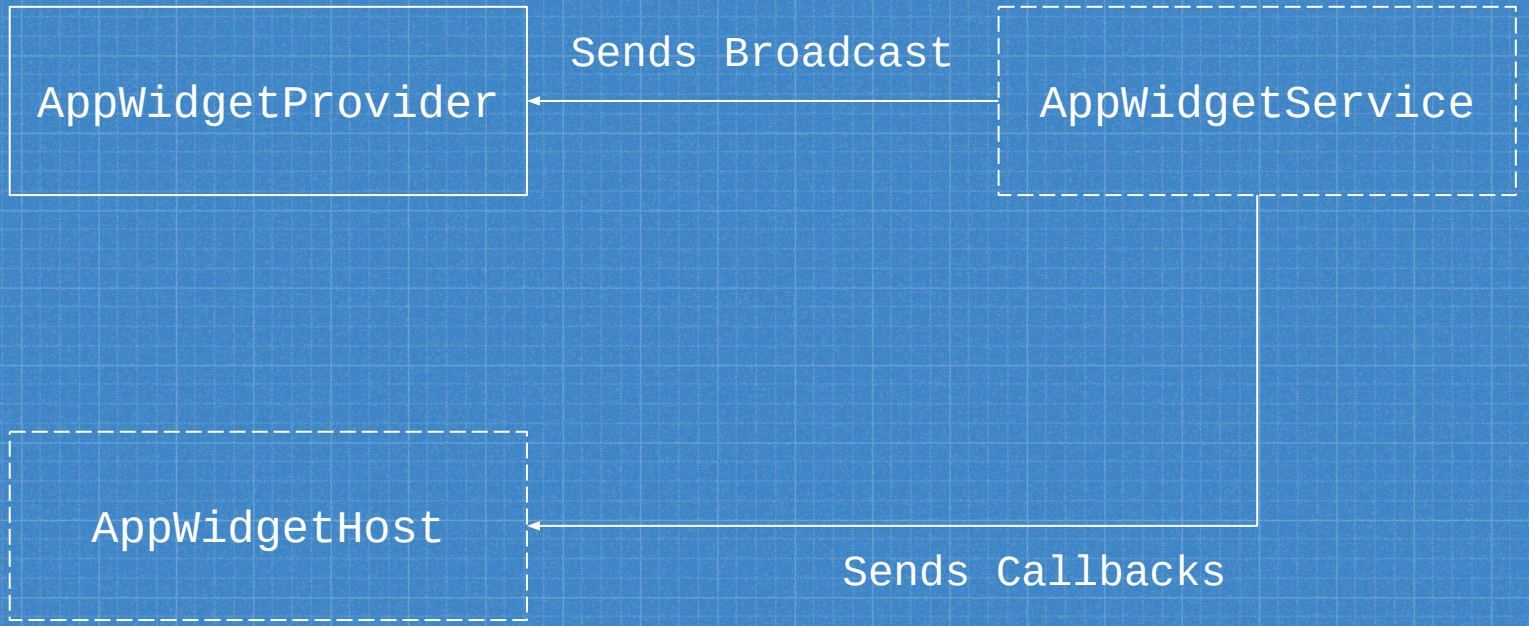
Архитектура компонентов виджета

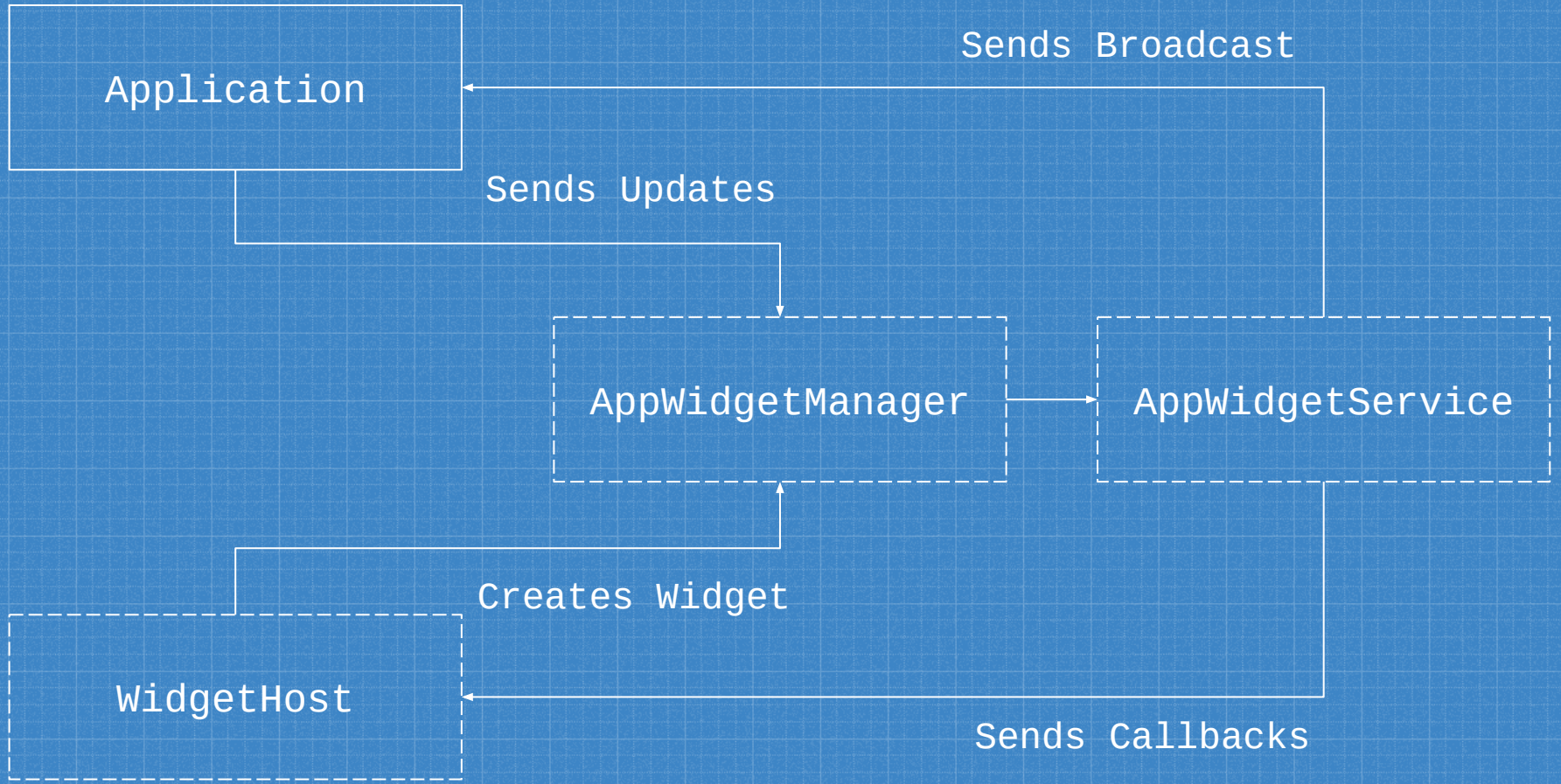

Application
Sends Updates
AppWidgetManager
AppWidgetService
WidgetHost
Creates Widget

AppWidgetManager

Предоставляет методы для работы с виджет сервисом


AppWidgetManager
IAppWidgetService.aidl
AppWidgetService


AppWidgetProvider
Sends Broadcast
AppWidgetService
AppWidgetHost
Sends Callbacks


Application
Sends Broadcast
Sends Updates
AppWidgetManager
AppWidgetService
Creates Widget
WidgetHost
Sends Callbacks

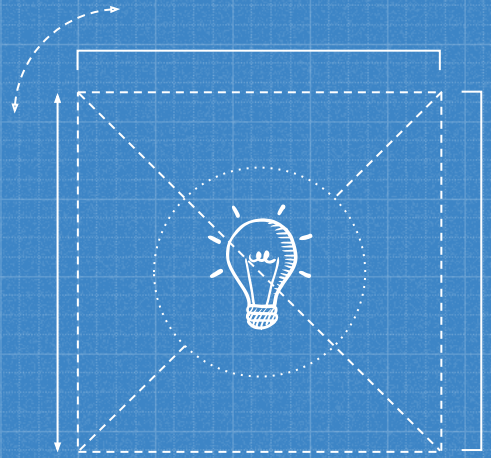
AppWidgetService

- Создание виджета
- Bind сервиса лаунчера и сервиса коллекций приложения
- Отправление сообщений бродкаст ресиверу приложения

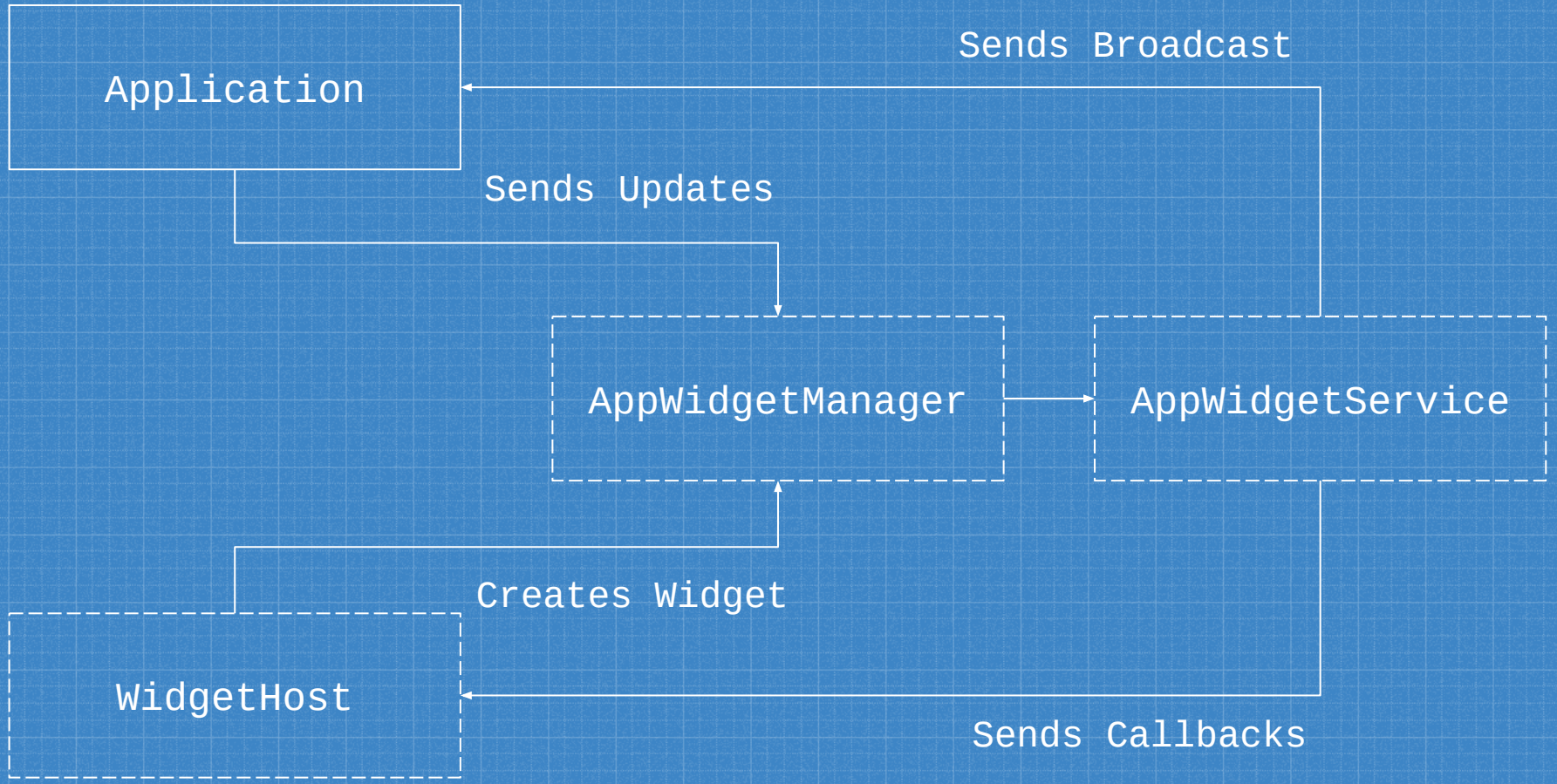
3

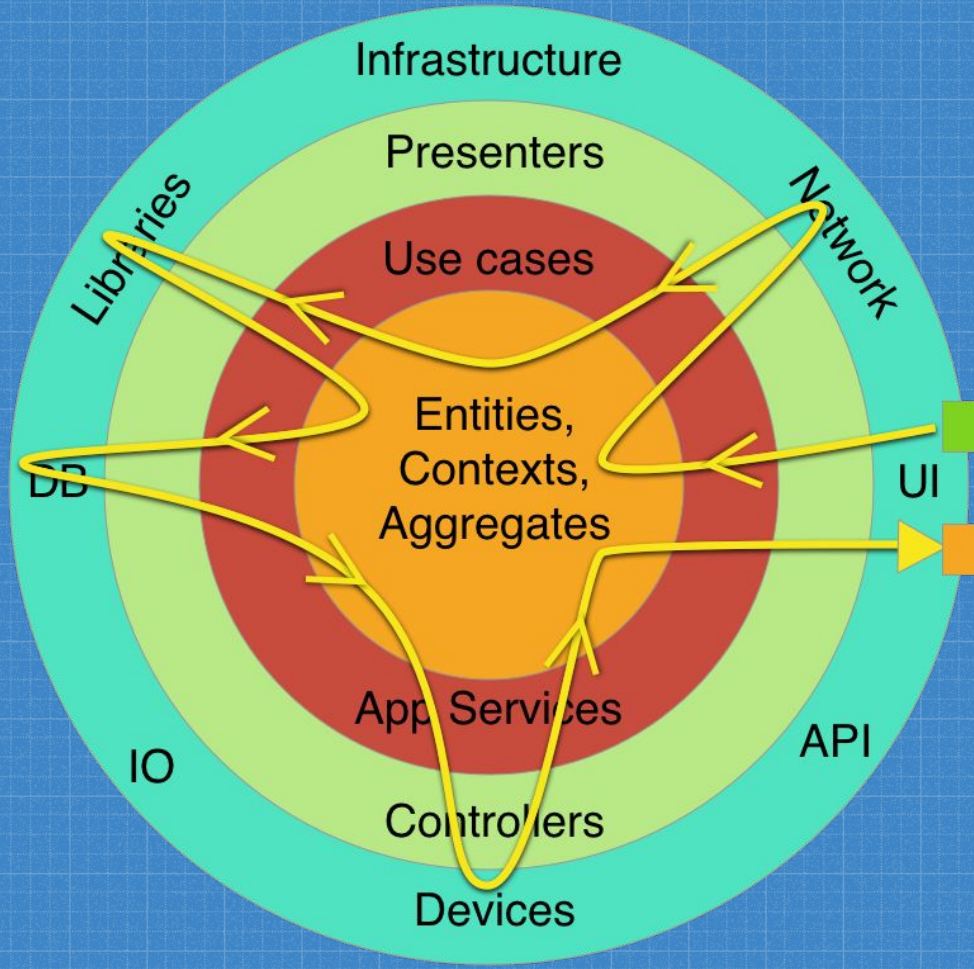
Данные и события

Учим виджет показывать данные
и реагировать на события



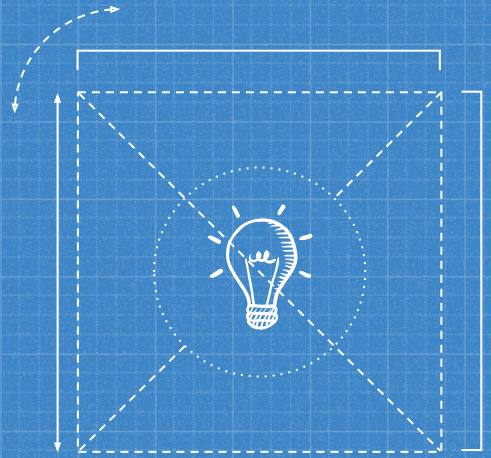
Виджет не должен содержать чувствительные данные


Application
Sends Broadcast
Sends Updates
AppWidgetManager
AppWidgetService
Creates Widget
WidgetHost
Sends Callbacks


Infrastructure
Presenters
Use cases
Entities,
Contexts,
Aggregates
App Services
Controllers
Devices
Libraries
Network
DB
UI
IO
API
Input
Output
BroadcastReceiver
AppWidgetManager

```kotlin
override fun onUpdate(
    context: Context,
    appWidgetManager: AppWidgetManager,
    appWidgetIds: IntArray
) {
    appWidgetIds.forEach { id ->
        val remoteViews = RemoteViews(context.packageName, R.layout.layout_watchlist_widget)
        val watchlistViewModel = WatchlistViewModel(watchlistId = 0)
        remoteViews.setTextViewText(R.id.widget_watchlist_name, watchlistViewModel.name)
        appWidgetManager.updateAppWidget(id, remoteViews)
    }
    super.onUpdate(context, appWidgetManager, appWidgetIds)
}
```

```
class WatchlistViewModel(private val watchlistId: Int) : ViewModel {
    private val watchlistInteractor: WatchlistInteractor =
        WatchlistInteractorImpl(PricesServiceImpl(), WatchlistServiceImpl())

    val symbols: List<Symbol>
        get() = watchlistInteractor.getWatchlist(watchlistId).symbols

    val name: String
        get() = watchlistInteractor.getWatchlist(watchlistId).name
}
```



onUpdate метод предполагает работу с готовыми данными. Для подготовки данных используйте сервисы

```kotlin
fun updateWidgets() {
    val widget = WidgetsManager[WidgetType.WATCHLIST]

    val appWidgetManager = AppWidgetManager.getInstance(context)
    val ids = appWidgetManager.getAppWidgetIds(ComponentName(context, widget.provider.java))

    if (ids.isNotEmpty()) {
        val intent = Intent(context, widget.service.java)
        intent.action = AppWidgetManager.ACTION_APPWIDGET_UPDATE
        when {
            DeviceInfo.apiLevelAtLeast(Build.VERSION_CODES.O) ->
                context.startForegroundService(intent)
            else -> context.startService(intent)
        }
    }
}
```

ГДЕ ДЕЛАТЬ ОБНОВЛЕНИЕ UI ВИДЖЕТА?

В BroadcastReceiver

После обновления данных можно отправить бродкаст.

+ Обновление UI в одном месте
- onReceive может вызываться с задержкой

В Service

Можно сделать обновление UI и в сервисе

+ Тратится меньше времени на вызов обновления UI
- Появляется несколько мест для обновления UI

ГДЕ ДЕЛАТЬ ОБНОВЛЕНИЕ UI ВИДЖЕТА?

В BroadcastReceiver

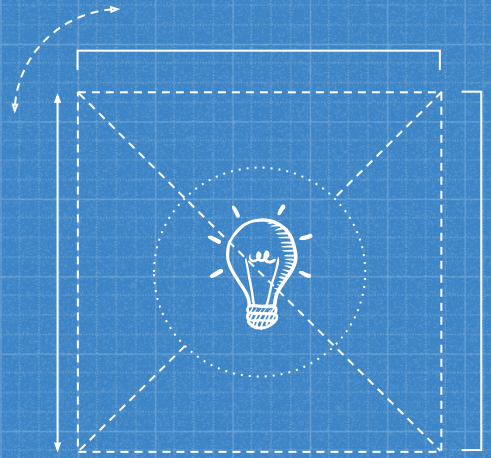
Когда нет жестких требований для обновления UI.

- Виджет с редко обновляемыми данными
- Виджет со статическими данными

В Service

Жесткие рамки для обновления UI

- Виджет с часто обновляемыми данными (плеер, датчик IoT)



Нужно помнить про lifecycle приложения.
Запуск ресиверов и сервисов может
изменить поведение приложения

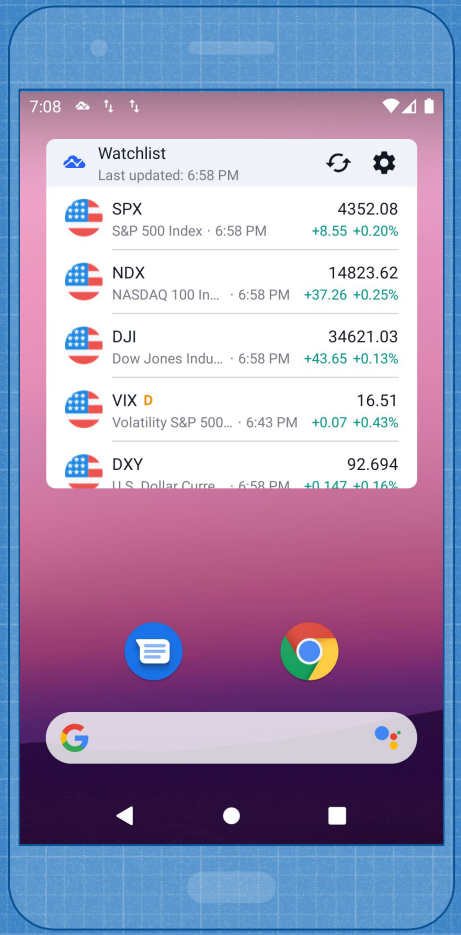
4

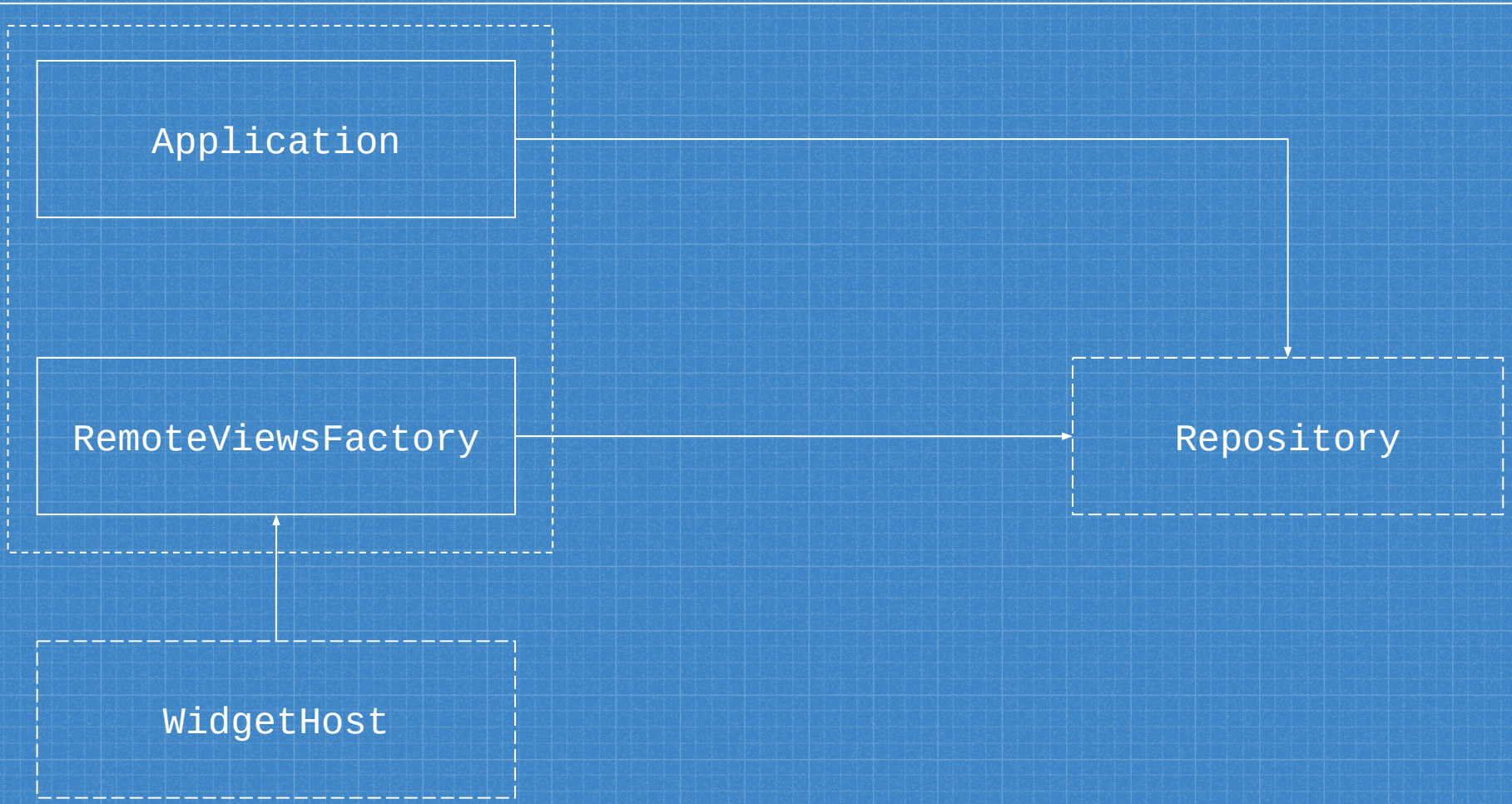
Списки

Дружим со списками
внутри виджета

Списки

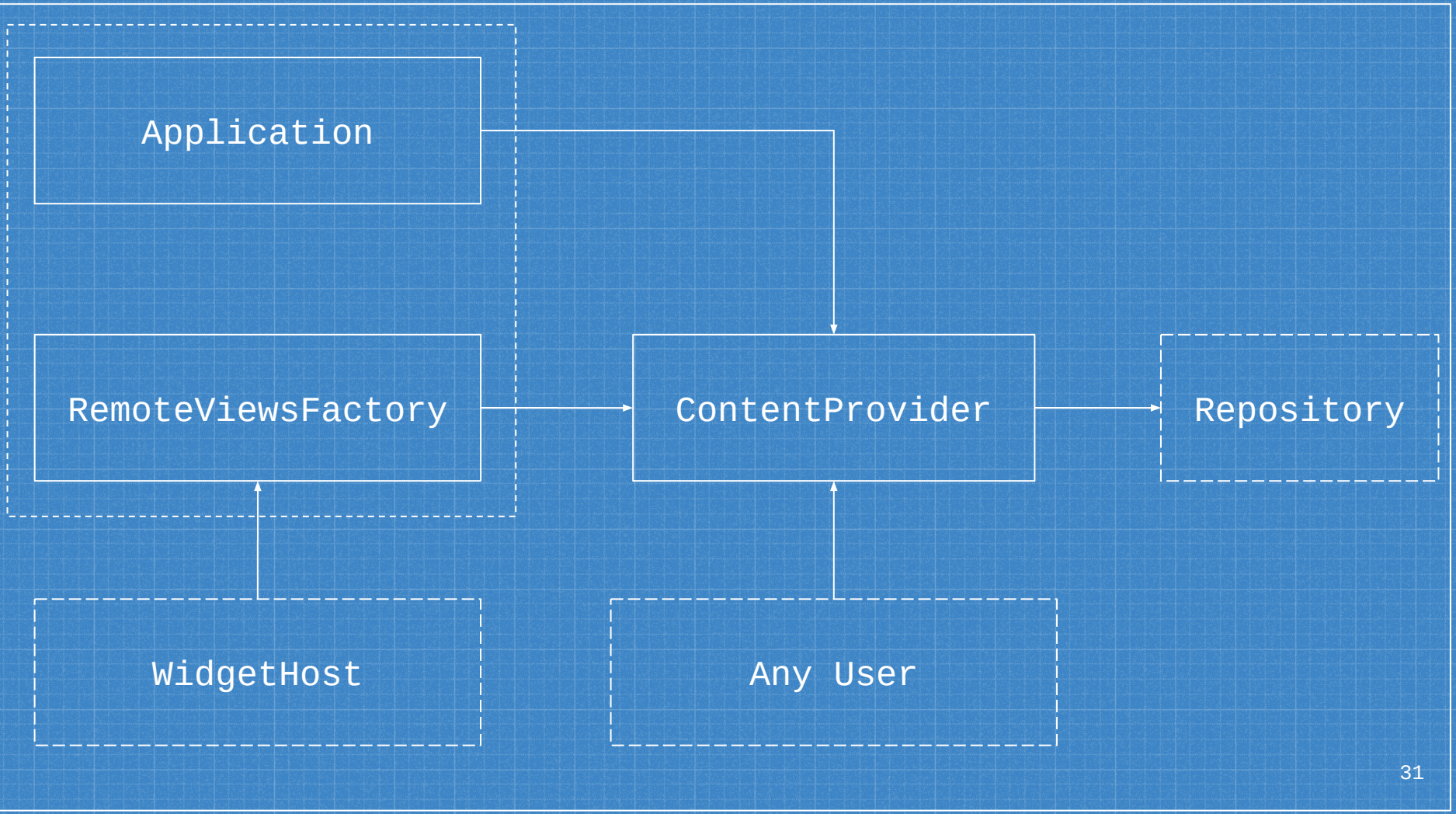
Списки реализуются с помощью фабрики **`RemoteViewsFactory`**




Application
RemoteViewsFactory
Repository
WidgetHost

```
public interface RemoteViewsFactory {
  public void onCreate();
  public void onDataSetChanged();
  public void onDestroy();
  public int getCount();
  public RemoteViews getViewAt(int position);
  public RemoteViews getLoadingView();
  public int getViewTypeCount();
  public long getItemId(int position);
  public boolean hasStableIds();
}
```

```
class WatchlistItemsViewsFactory(private val widgetId: Int)
        : RemoteViewsService.RemoteViewsFactory {
    ...
    override fun getViewAt(position: Int): RemoteViews {
        val packageName = WidgetsApp.instance.applicationContext.packageName
        val view = RemoteViews(packageName, R.layout.item_watchlist_widget)
        //Get data and set to view
        ...
        return view
    }
    ...
}
```


Application
RemoteViewsFactory
ContentProvider
Repository
WidgetHost
Any User

ГДЕ ПОЛУЧАТЬ ДАННЫЕ?

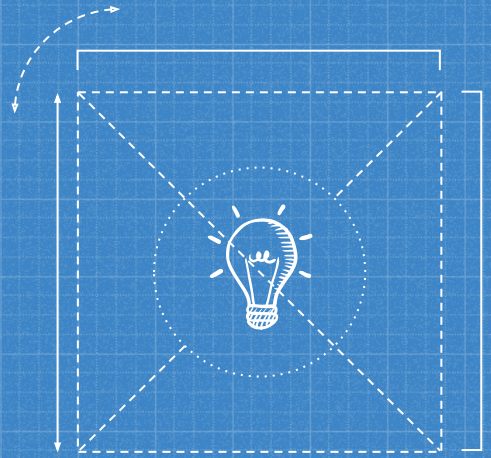
В ContentProvider

+ Единый источник данных для всех типов пользователей
- Неудобный API, завязанный на Cursor
- Время на инициализацию

В ViewModel

+ Проще доступ к данным
+ Выше гибкость
- Меньше контроля за данными

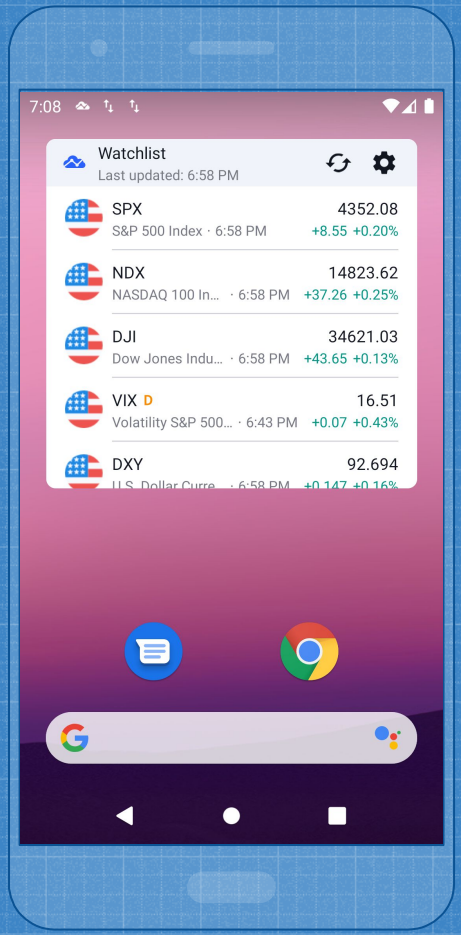
```
class WatchlistItemsViewsFactory(private val widgetId: Int)
        : RemoteViewsService.RemoteViewsFactory {
    ...
    override fun getViewAt(position: Int): RemoteViews {
        val packageName = WidgetsApp.instance.applicationContext.packageName
        val view = RemoteViews(packageName, R.layout.item_watchlist_widget)
        //Get data and set to view
        ...
        return view
    }
    ...
}
```



getViewAt синхронный метод. Во время его выполнения будет вызван метод **getLoadingView**

Проблема

Загрузка иконок занимает длительное время.



ГРУЗИМ ДАННЫЕ ДЛЯ UI

Асинхронно

Собираем кэш целиком, потом показываем данные

+ Можно быстрее показать данные на виджете
- Нужно дополнительно обновлять UI, что ведет к “прыганию” ячеек под пальцами

Синхронно

Блокируем показ до момента загрузки данных

+ UI более предсказуем, нет “прыжков”
- Дольше первая загрузка

4

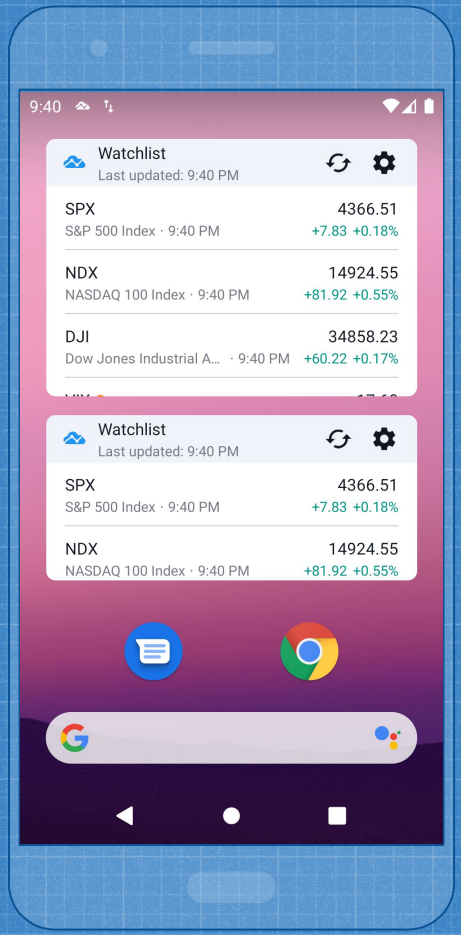
Обновления

Обновляем виджет
с нужным интервалом

Проблема

Несколько виджетов на экране

- Обновления виджетов могут занять минуты
- Стандартное ограничение на обновление - 30 минут

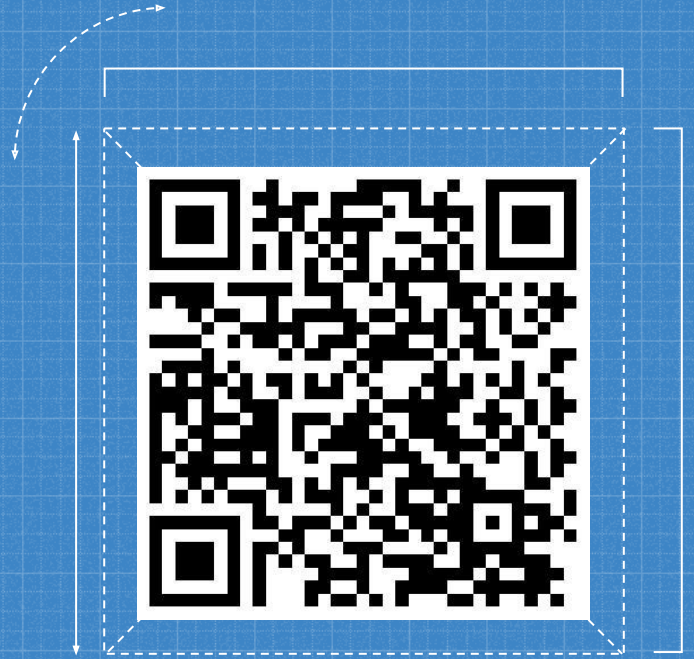


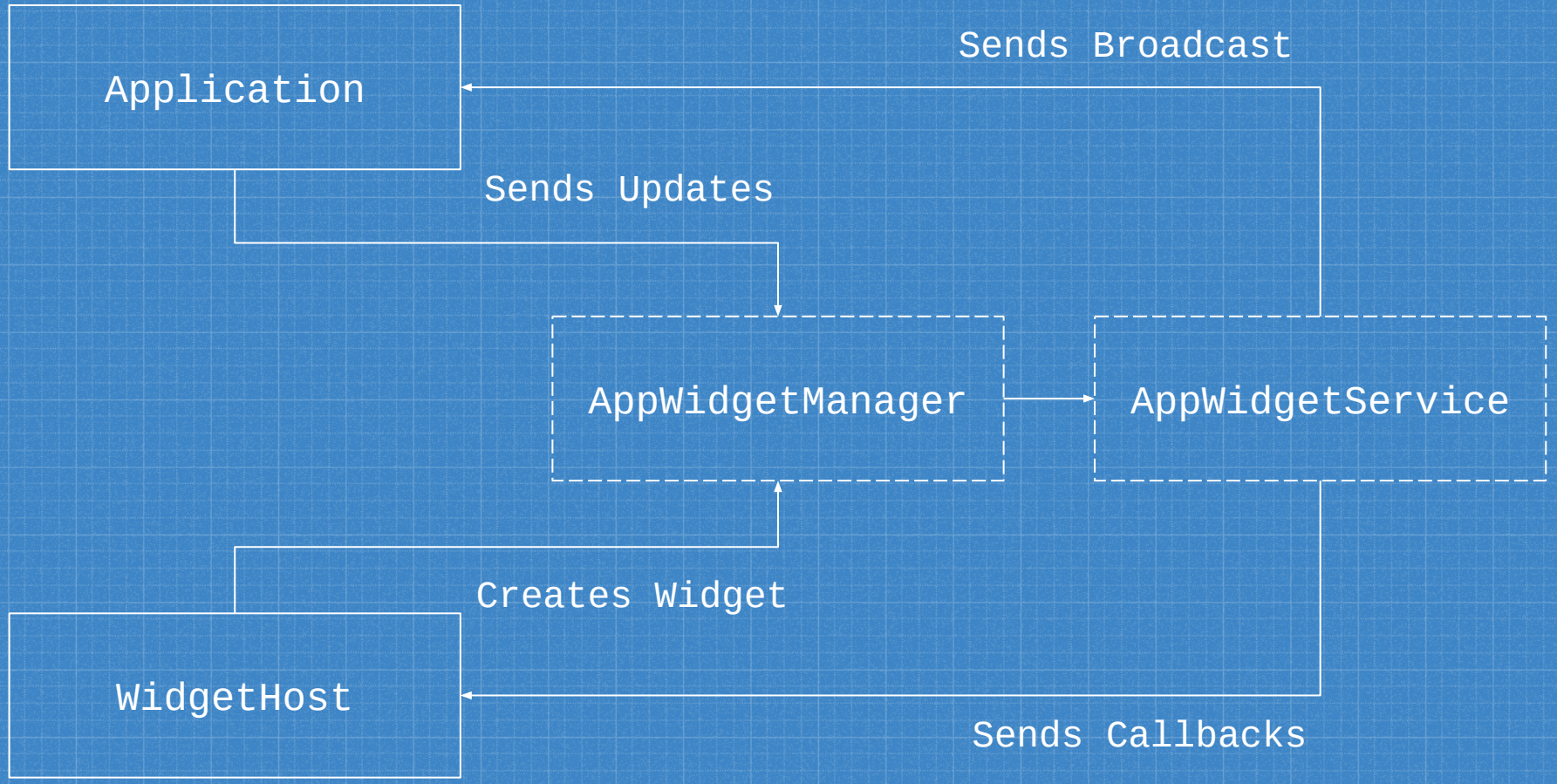
Обновление данных, ограничения

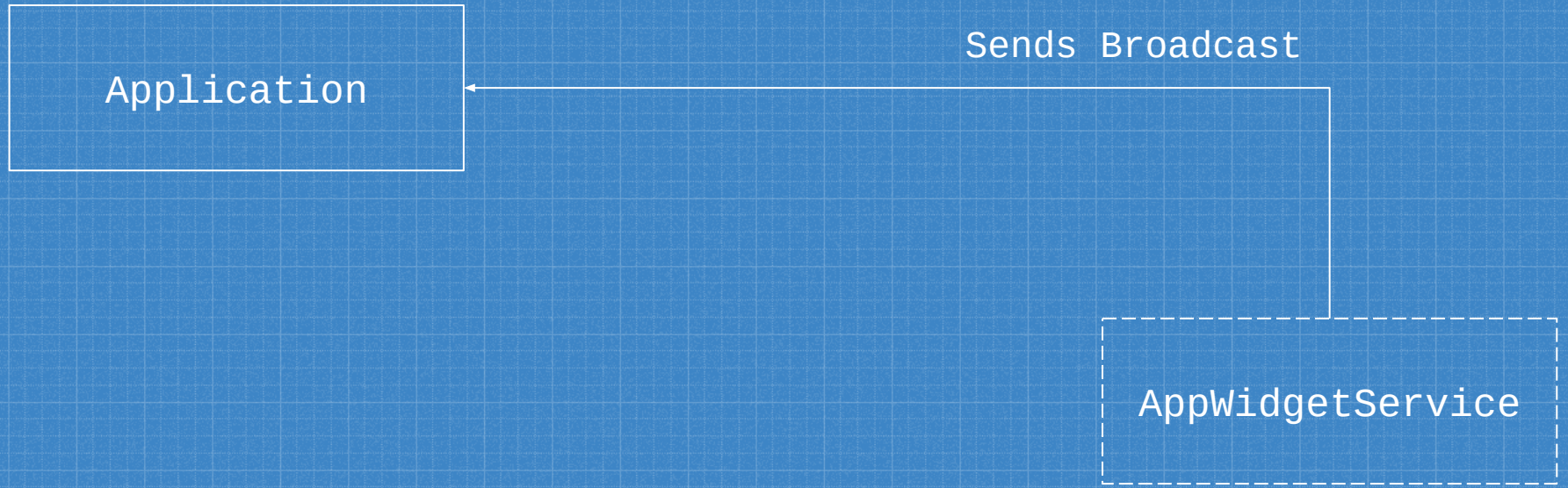
- Слабый сигнал мобильной сети
- Высокая нагрузка на сервер
- Тех работы на сервере
- Неисправность маршрутизаторов
- Любые другие проблемы

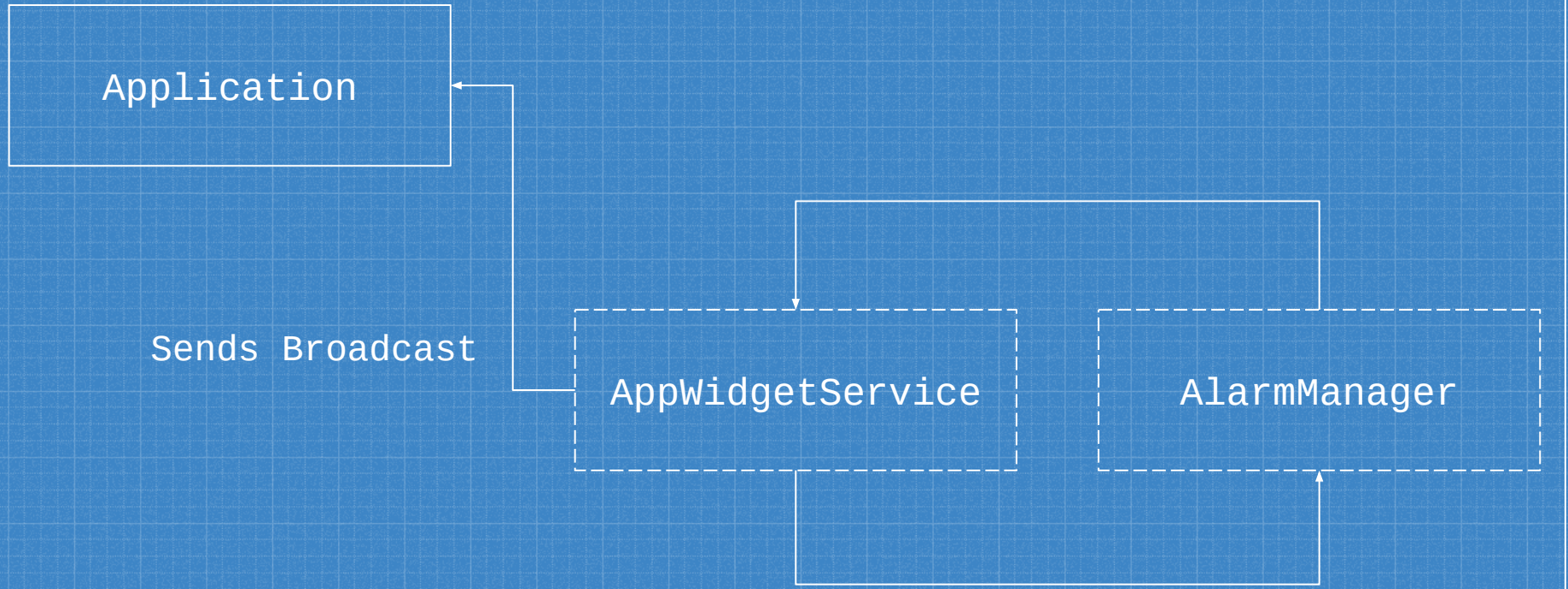
Foreground Service

Предназначен для выполнения длительных операций




Application
Sends Broadcast
Sends Updates
AppWidgetManager
AppWidgetService
Creates Widget
WidgetHost
Sends Callbacks

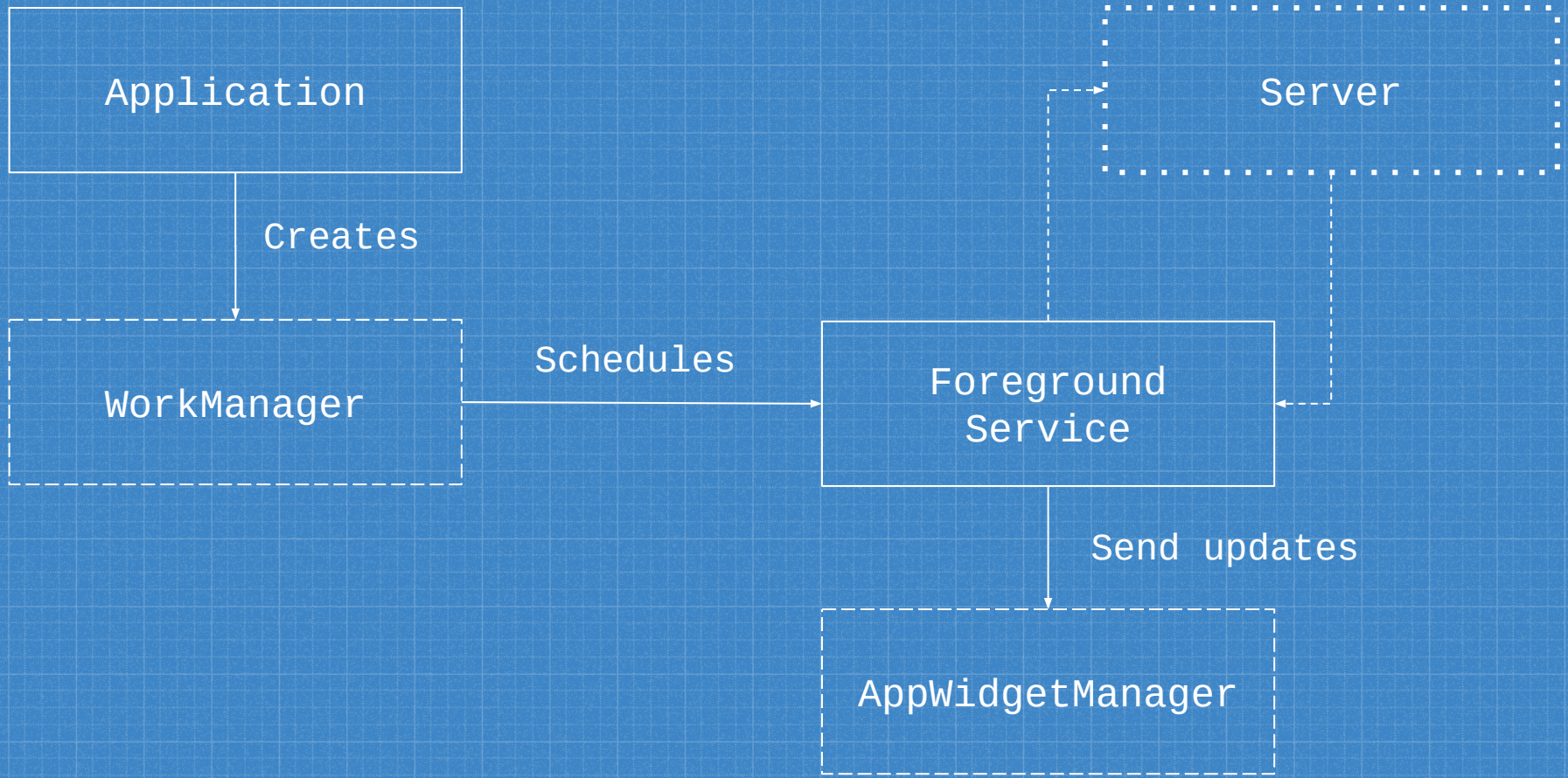

Application
Sends Broadcast
AppWidgetService


Application
Sends Broadcast
AppWidgetService
AlarmManager

WorkManager

Getting started

Официальная документация


Application
Server
Creates
WorkManager
Schedules
Foreground
Service
Send updates
AppWidgetManager

5

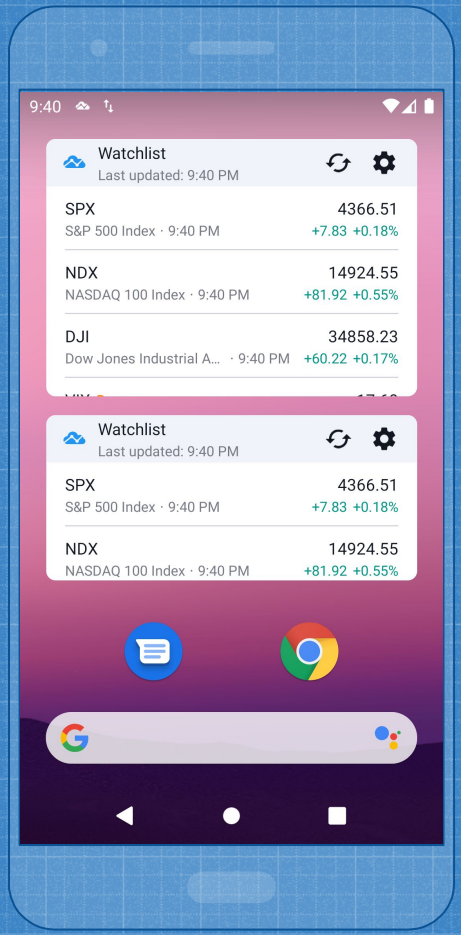
Хранение данных

Храним конфигурацию виджета

Проблема

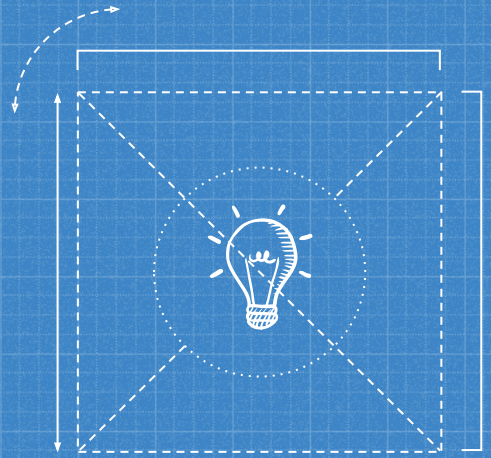
Несколько виджетов на экране

- Требуется хранить данные для каждого виджета

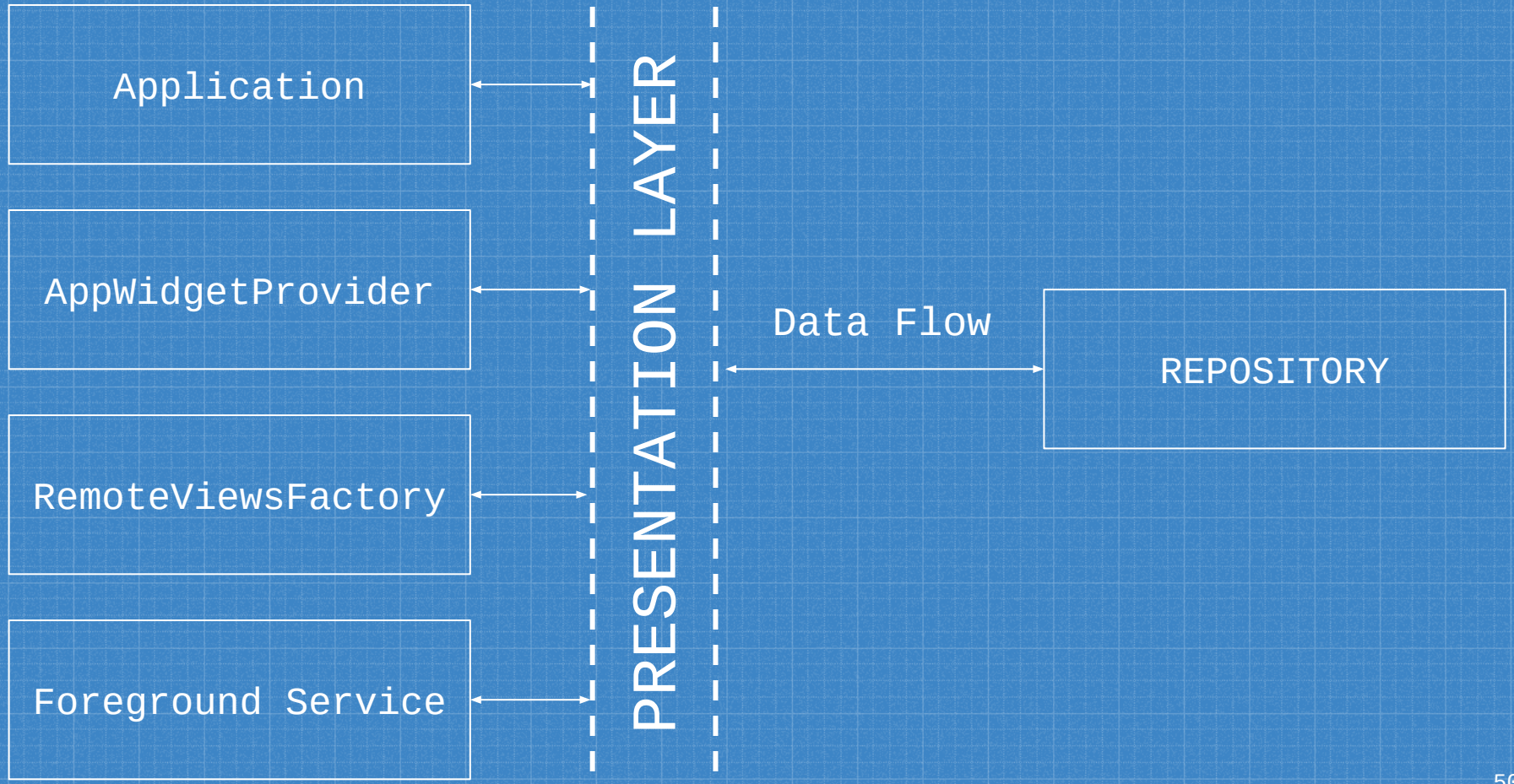


Хранение данных

- Можно добавить неограниченное количество виджетов
- Любой виджет можно удалить

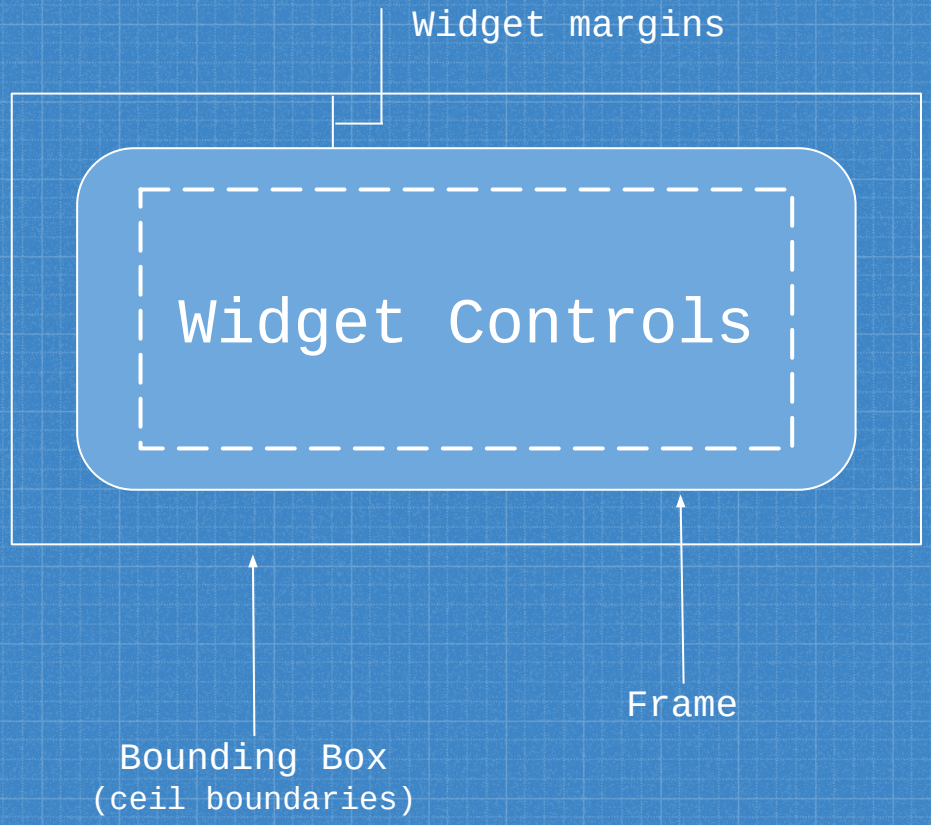


Для виджета требуется выстроить высокий контроль за потоком данных


Application
AppWidgetProvider
RemoteViewsFactory
Foreground Service
PRESENTATION LAYER
Data Flow
REPOSITORY

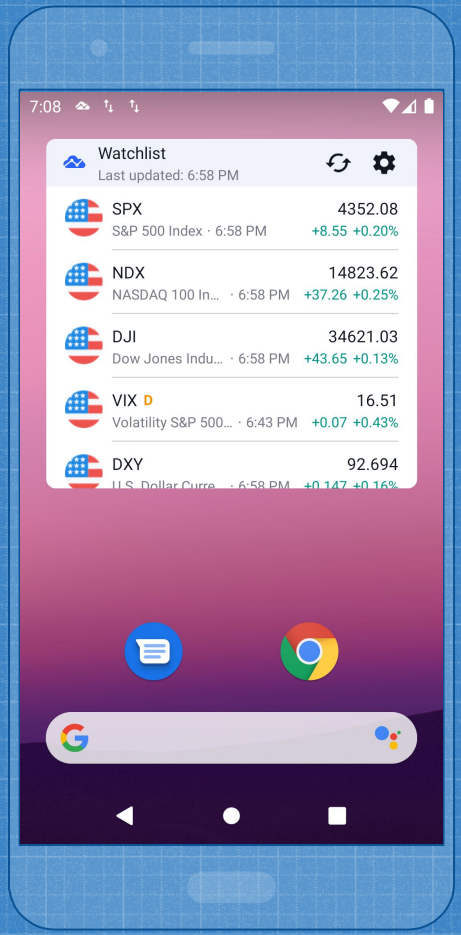
6 Дизайн

Учитываем особенности виджета


Widget margins
Widget Controls
Bounding Box
(ceil boundaries)
Frame

Компоненты

View должны сериализоваться, поэтому доступен только определенный набор компонентов

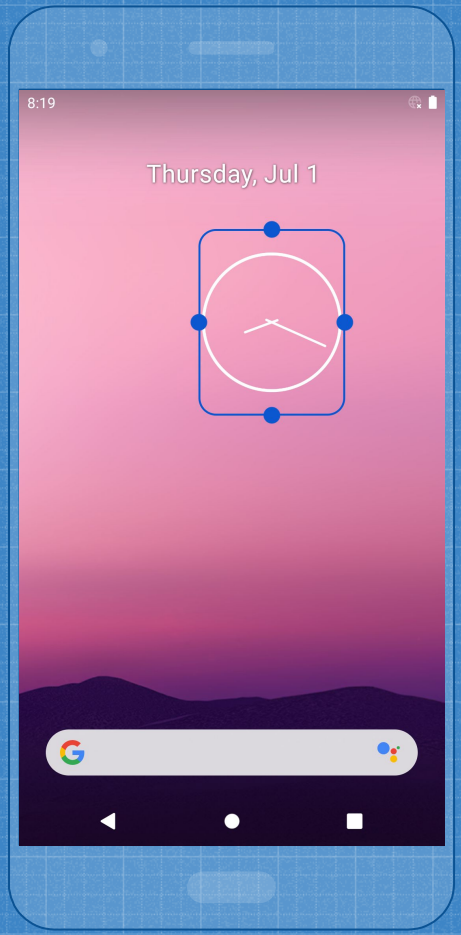


resizeMode

Правило, которое описывает
как будет ресайзиться
контейнер с виджетом.
Доступные значения:

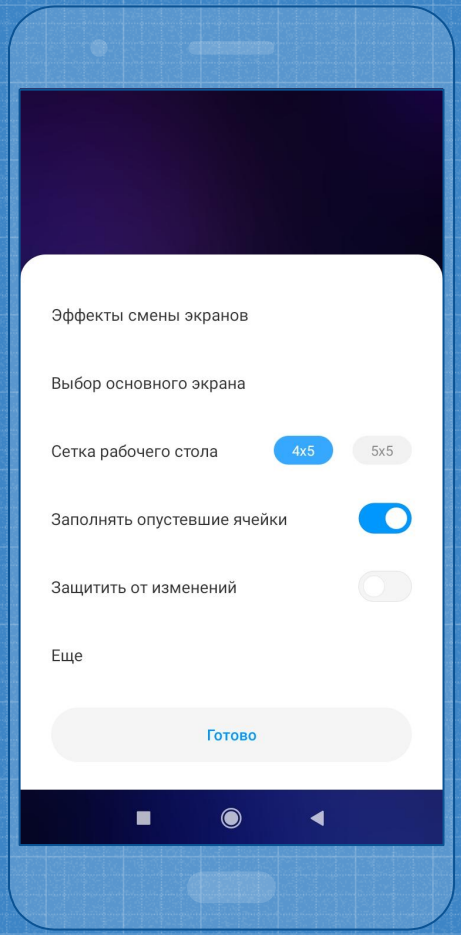
- horizontal
- vertical
- none

resizeMode=”horizontal|vertical”



Размер ячеек

Ячейки имеют определенный размер. Но, ресайзом виджета управляет лаунчер и он может настраивать размер ячеек на свое усмотрение.



OpenLauncher

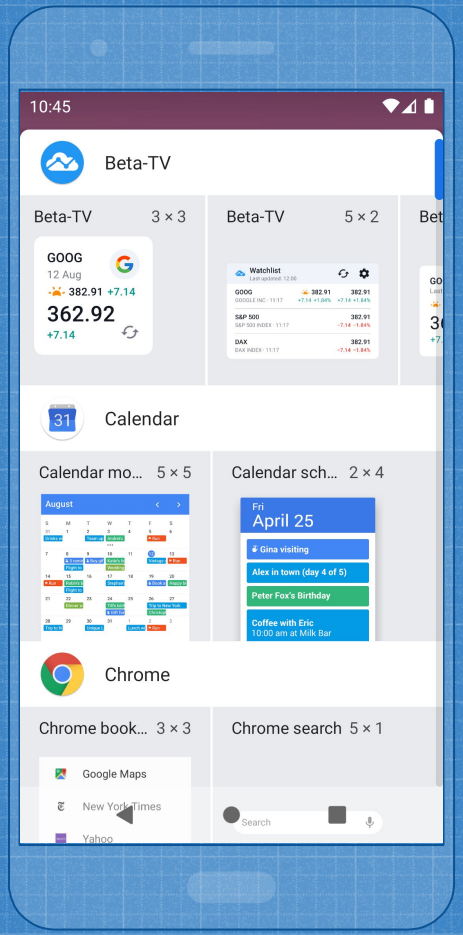
CellContainer

Пример того, как можно реализовать сетку в кастомном лаунчере

previewImage

Указывается картинка, которая покажет пользователю, как будет выглядеть виджет.

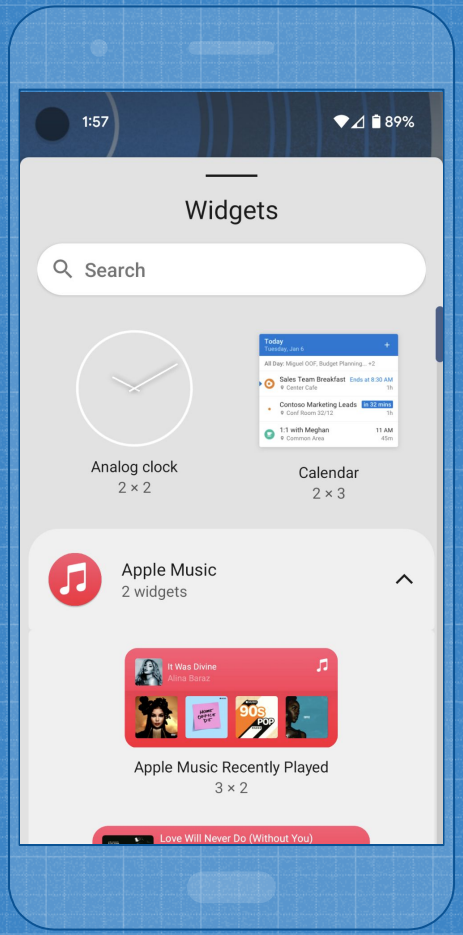
Но наличие сетки и растягивания виджета по ячейкам может сбить с толку



previewLayout

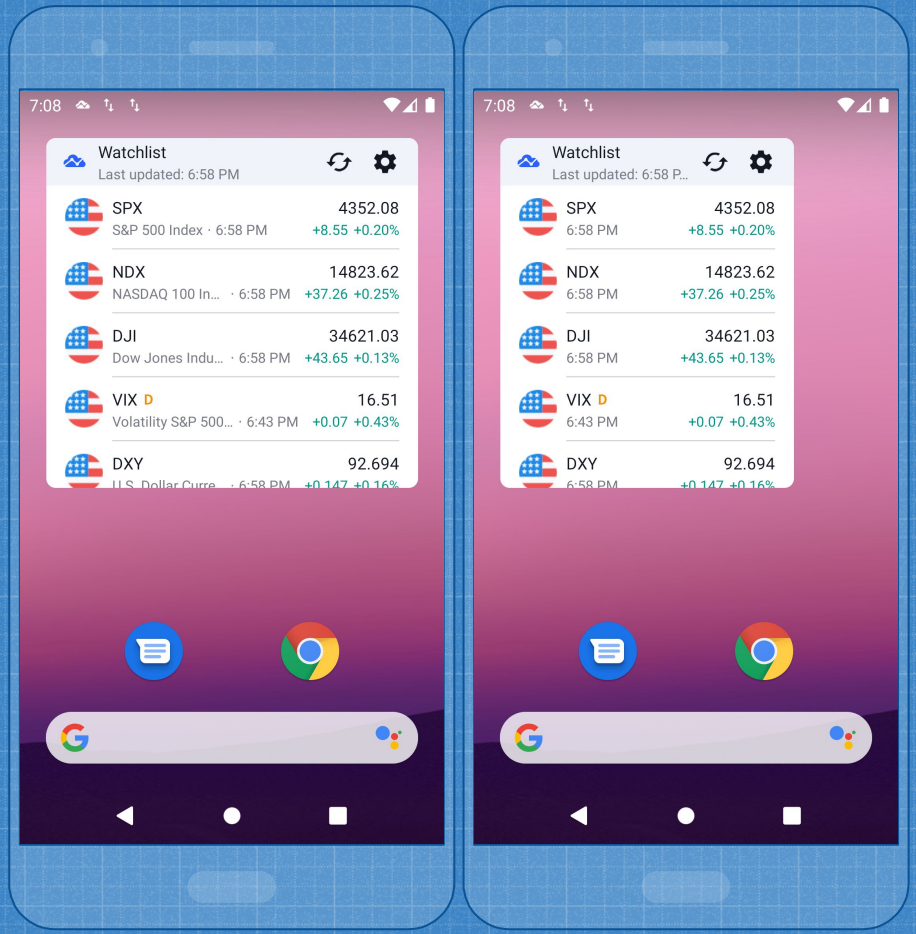
Вместо картинки будет отрисовываться верстка.

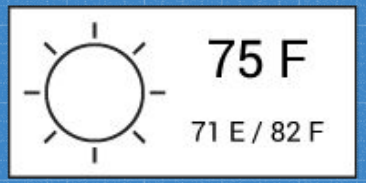
Но по прежнему пользователь будет “угадывать” как виджет будет выглядеть

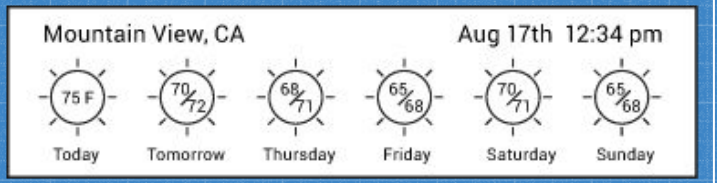


Резиновая верстка

Следует заранее продумать как виджет будет изменять визуальное отображение в зависимости от размеров контейнера







Наполнение виджета может меняться в зависимости от его размера

```
class WatchlistWidgetProvider : AppWidgetProvider() {
    override fun onUpdate(
        context: Context,
        manager: AppWidgetManager,
        appWidgetIds: IntArray
    ) {
      ...
    }

    override fun onAppWidgetOptionsChanged(
        context: Context, manager: AppWidgetManager,
        widgetId: Int, newOptions: Bundle
    ) {
      ...
    }
}
```

```kotlin
override fun onUpdate(context: Context, manager: AppWidgetManager, appWidgetIds: IntArray) {
    appWidgetIds.forEach { id ->
        val options = manager.getAppWidgetOptions(id)
        //size in dip
        val maxSize = Size(
            options.getInt(OPTION_APPWIDGET_MAX_WIDTH),
            options.getInt(OPTION_APPWIDGET_MAX_HEIGHT))

        val remoteViews = if (maxSize.width > 360) {
            RemoteViews(context.packageName, R.layout.layout_watchlist_widget)
        } else {
            RemoteViews(context.packageName, R.layout.layout_watchlist_widget_small)
        }
        manager.updateAppWidget(id, remoteViews)
    }
}
```

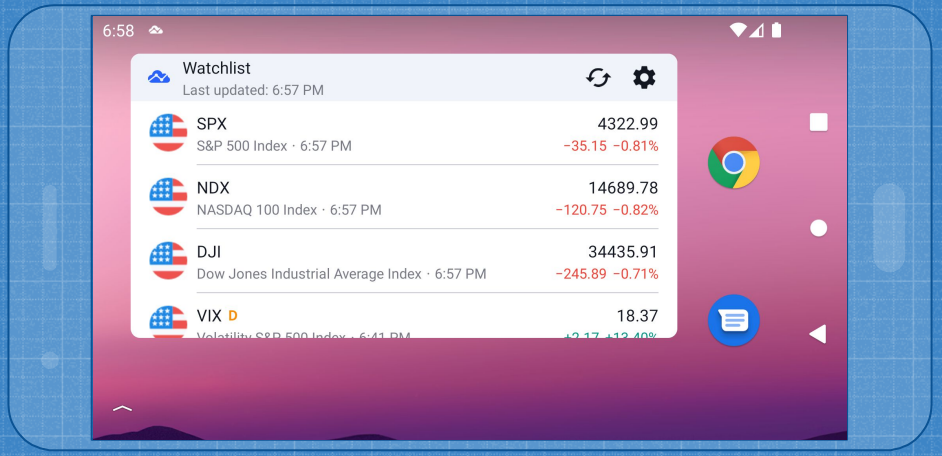
```kotlin
override fun onAppWidgetOptionsChanged(
    context: Context, manager: AppWidgetManager,
    widgetId: Int, newOptions: Bundle
) {
    val maxSize = Size(
        newOptions.getInt(OPTION_APPWIDGET_MAX_WIDTH),
        newOptions.getInt(OPTION_APPWIDGET_MIN_HEIGHT))
    val remoteViews = if (maxSize.width > 360) {
        RemoteViews(context.packageName, R.layout.layout_watchlist_widget)
    } else {
        RemoteViews(context.packageName, R.layout.layout_watchlist_widget_small)
    }
    manager.updateAppWidget(appWidgetId, remoteViews)
}
```

```
override fun onUpdate(...) {
  val smallView = ...
  val tallView = ...
  val wideView = ...

  val viewMapping: Map<SizeF, RemoteViews> = mapOf(
    SizeF(100f, 100f) to smallView,
    SizeF(100f, 200f) to tallView,
    SizeF(200f, 100f) to wideView
  )
  val remoteViews = RemoteViews(viewMapping)

  appWidgetManager.updateAppWidget(id, remoteViews)
}
```

Landscape

Лаунчеры умеют работать в альбомном режиме.

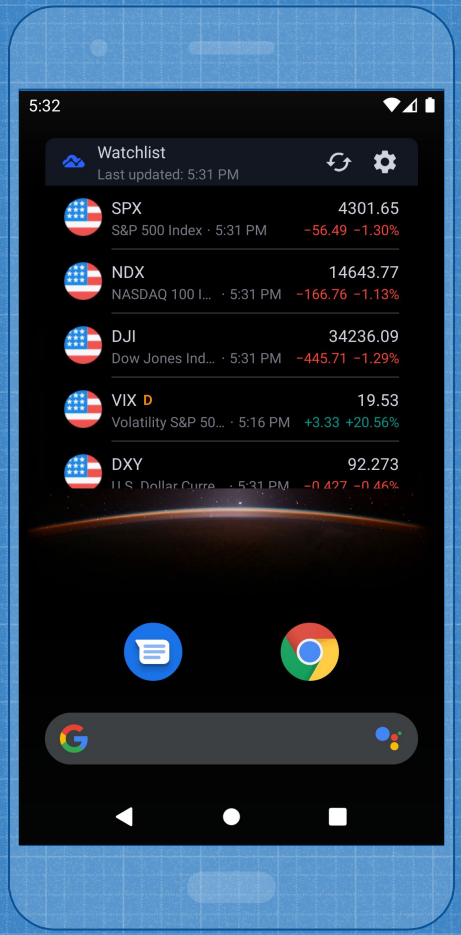
В случае смены ориентации ячейки меняют свои размеры



```
/**
 * RemoteViews(RemoteViews landscape, RemoteViews portrait)
 *
 * Create a new RemoteViews object that will inflate as
 * the specified landspace or portrait RemoteViews,
 * depending on the current configuration.
 */
val remoteViews = RemoteViews(landscape, portrait)
```

Dark Theme

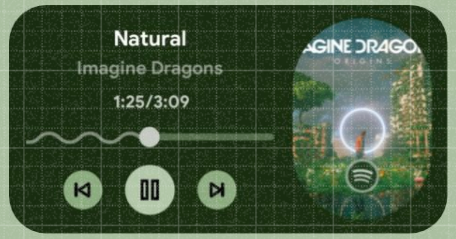
Как можно раньше начните проработку темной темы.

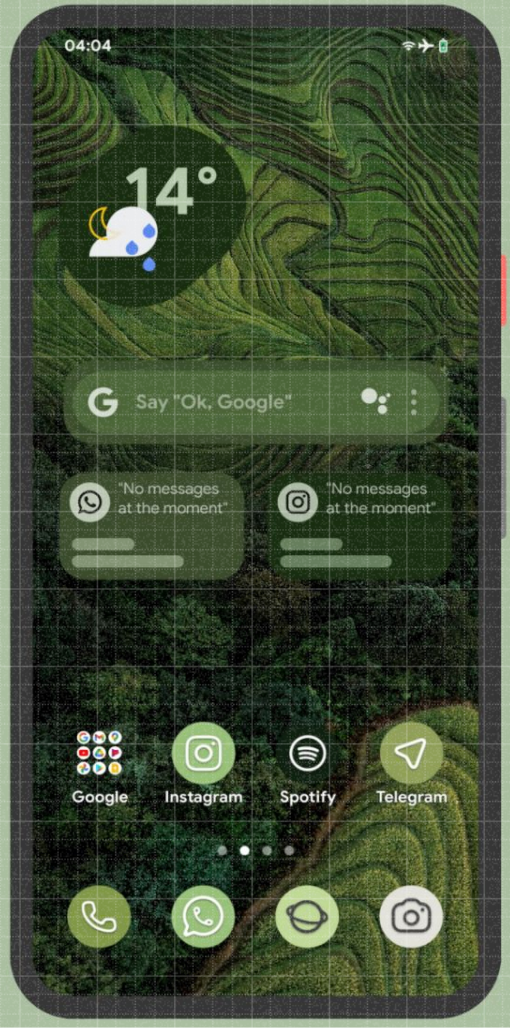


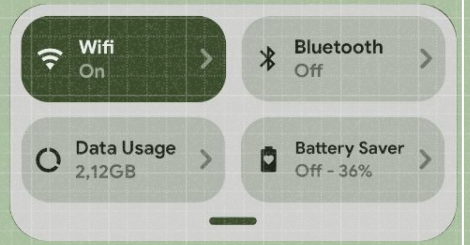
12

Auto theming

Material You Update



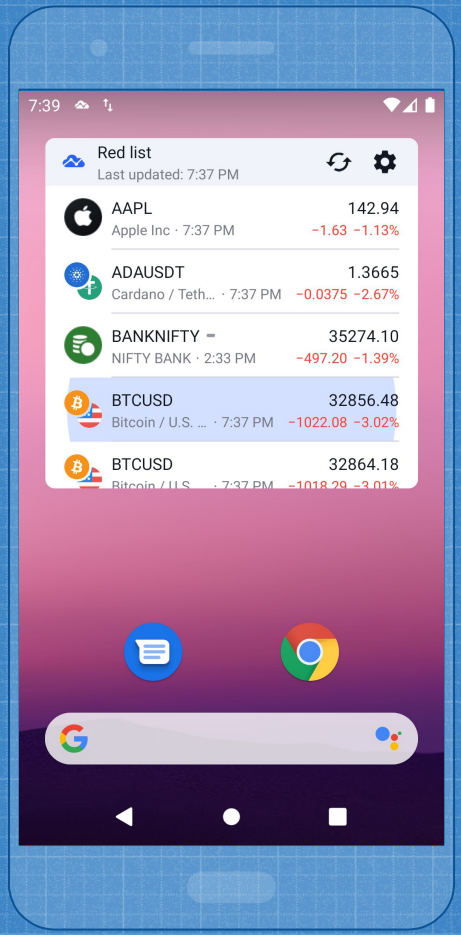




Next

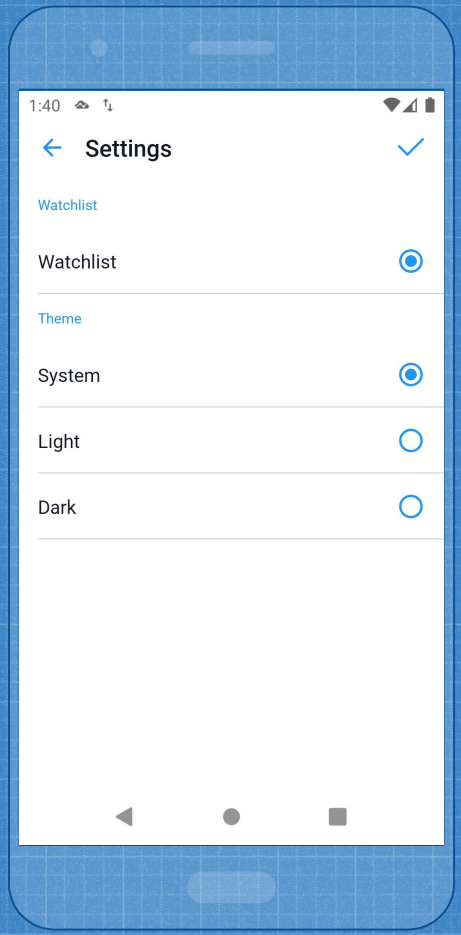
Touches

Уделяйте внимание к кликабельным элементам



Configuration

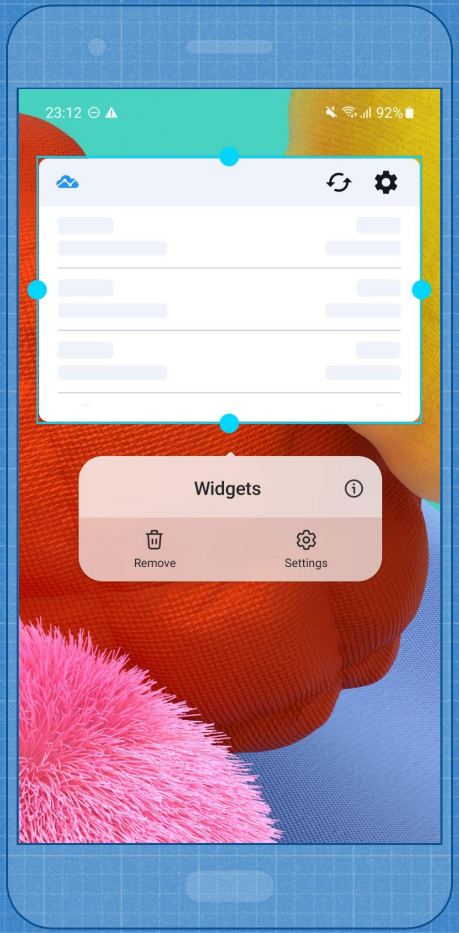
Это место для инициализации виджета, либо изменения его настроек

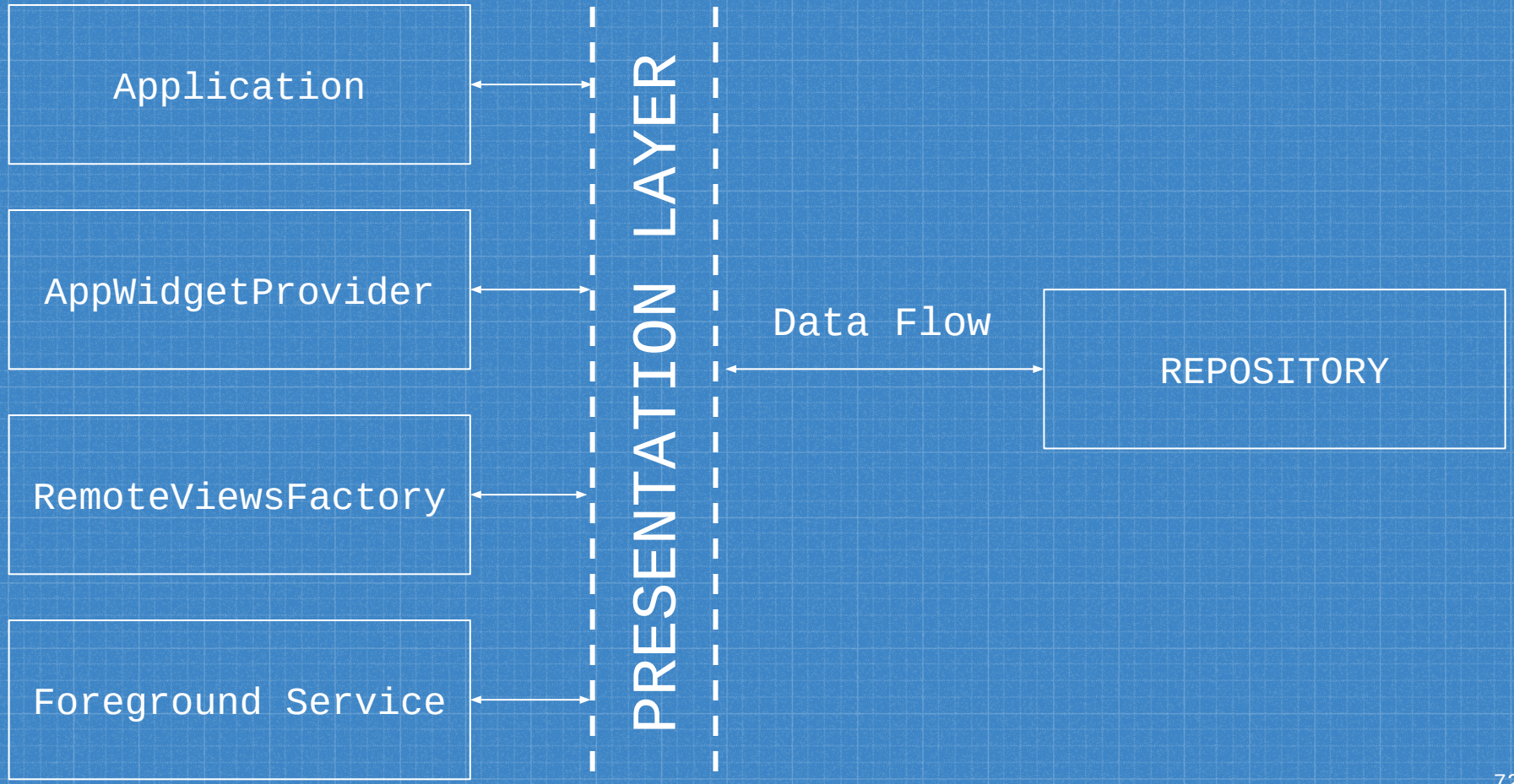


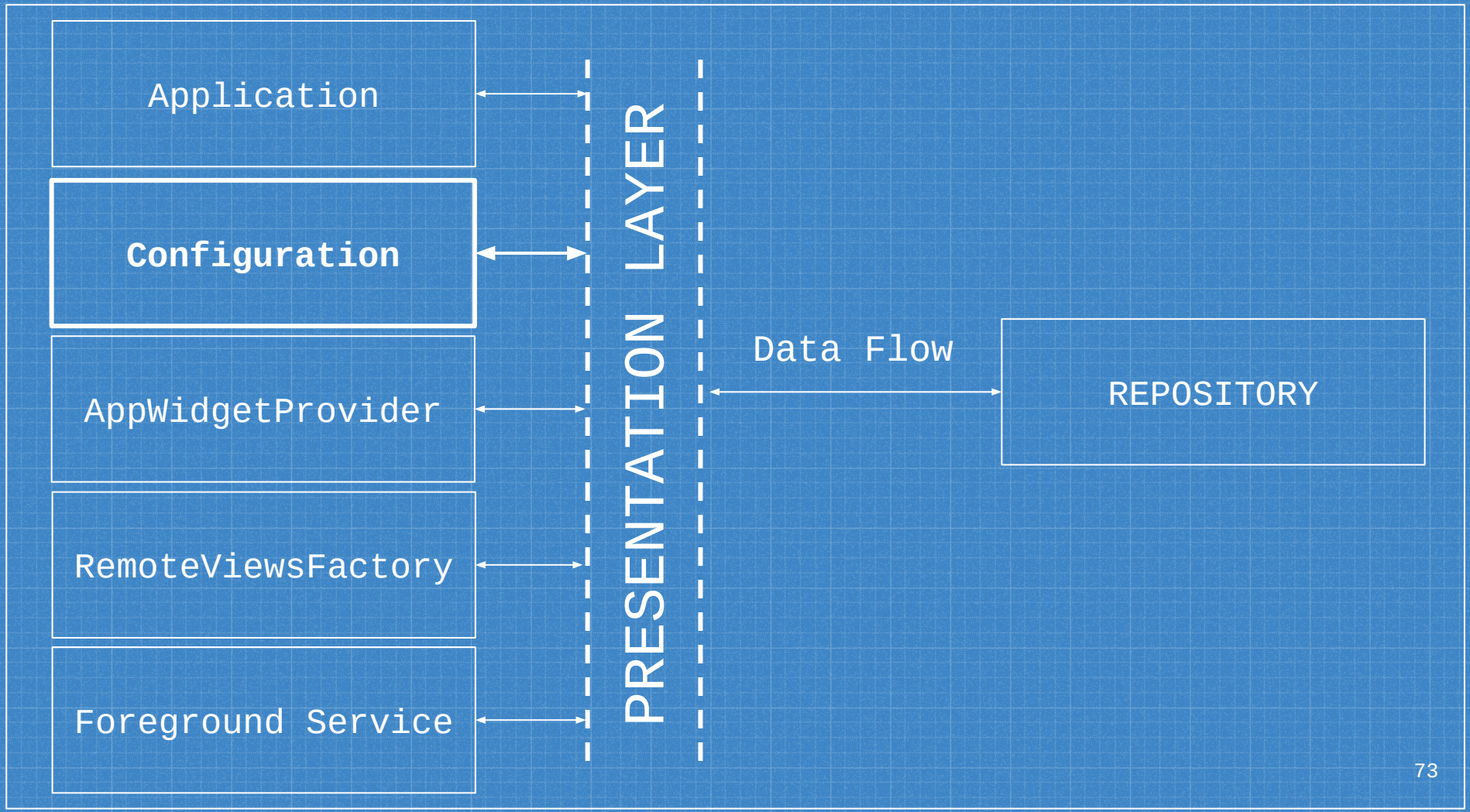
widgetFeatures

reconfigurable

Начиная с 9 андроида функционал доступен, но не широко распространен в лаунчерах

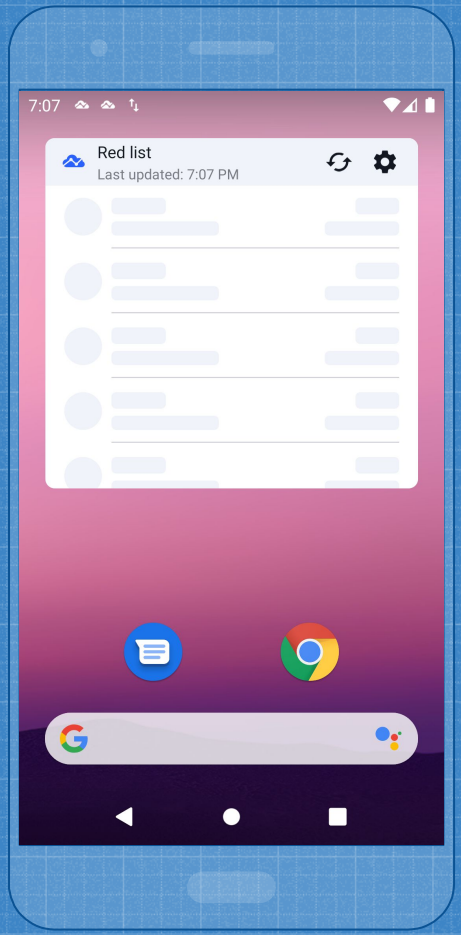


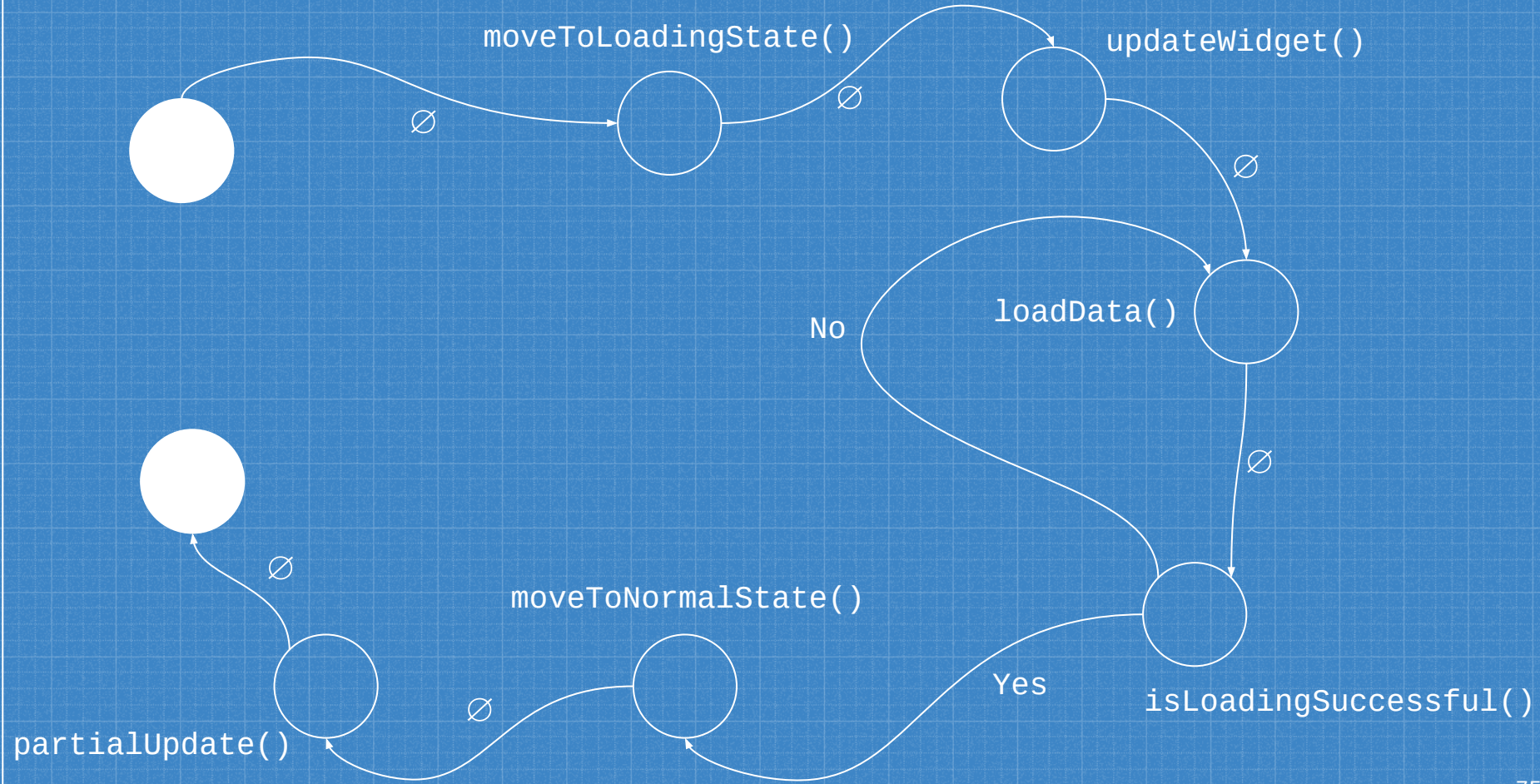

Application
AppWidgetProvider
RemoteViewsFactory
Foreground Service
PRESENTATION LAYER
Data Flow
REPOSITORY

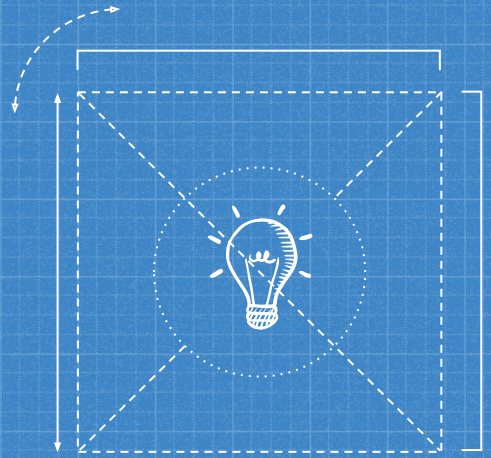

Application
Configuration
AppWidgetProvider
RemoteViewsFactory
Foreground Service
PRESENTATION LAYER
Data Flow
REPOSITORY

Loading

Продумайте состояние загрузки



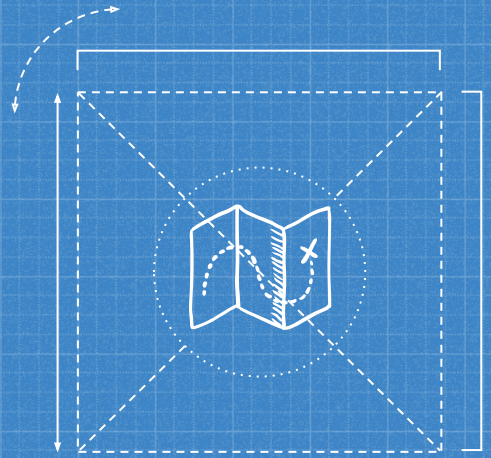

moveToLoadingState()
updateWidget()
∅
∅
∅
No
loadData()
∅
∅
moveToNormalState()
Yes
isLoadingSuccessful()
∅
partialUpdate()



Итеративный подход к разработке и дизайну не самый гладкий

Проблемы итераций

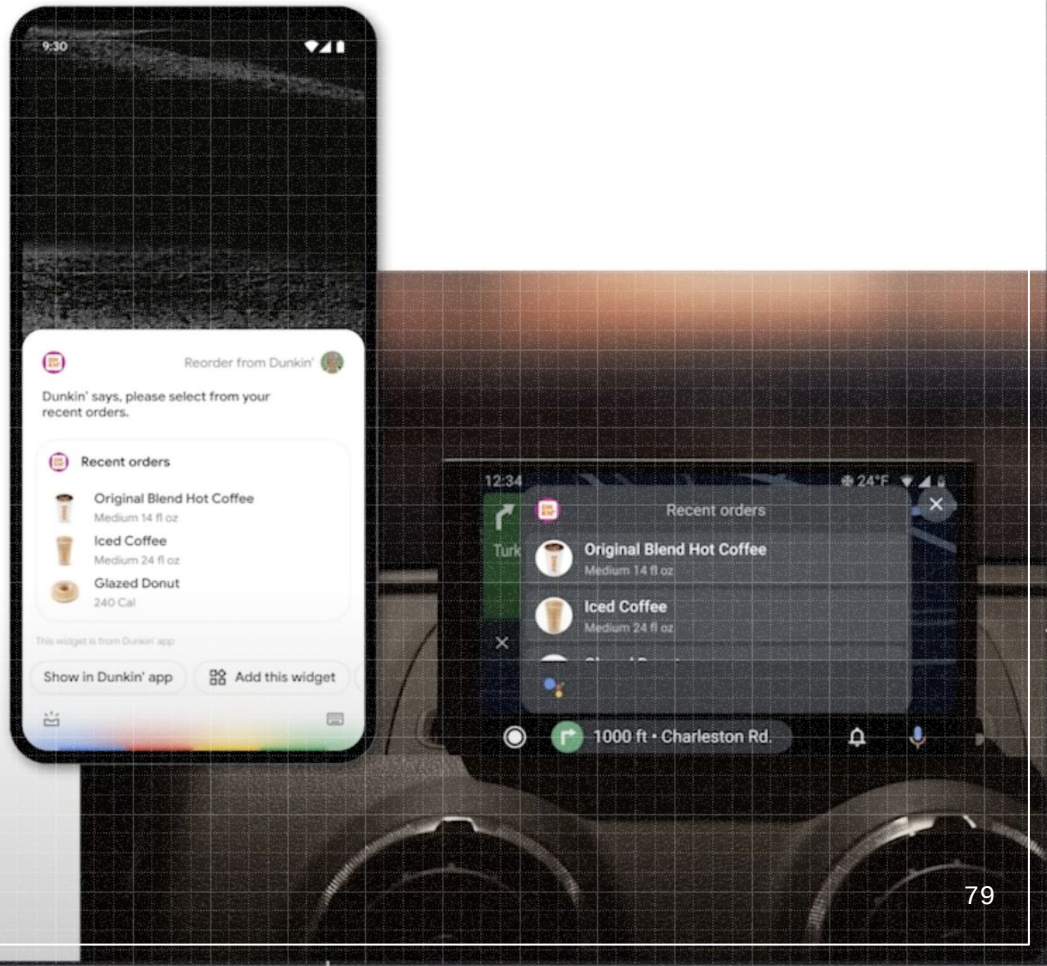
- Смена пакета провайдера влечет к удалению виджета
- Изменение resizeWidth и resizeHeight не изменяют уже установленные виджеты
- Отключение resizeMode может принести проблемы
- hide_from_picker настройка работает с Android 9



ЧТО ДАЛЬШЕ?

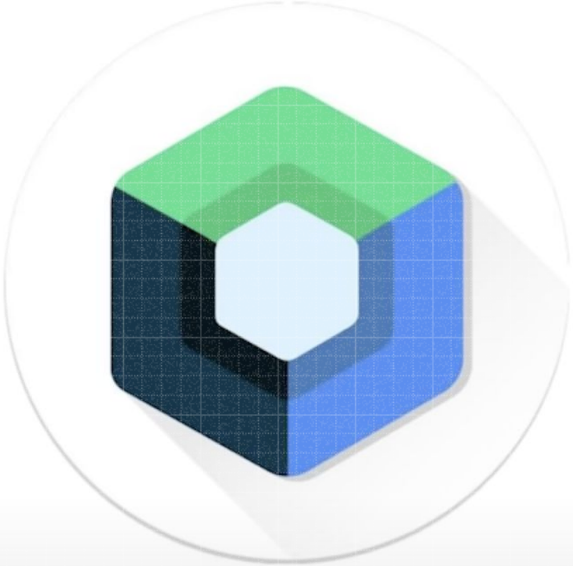
More Widgets in More Places

Enhanced integrations in **Android** and **Assistant** ensure users can find and use widgets in more places (via touch and voice!).



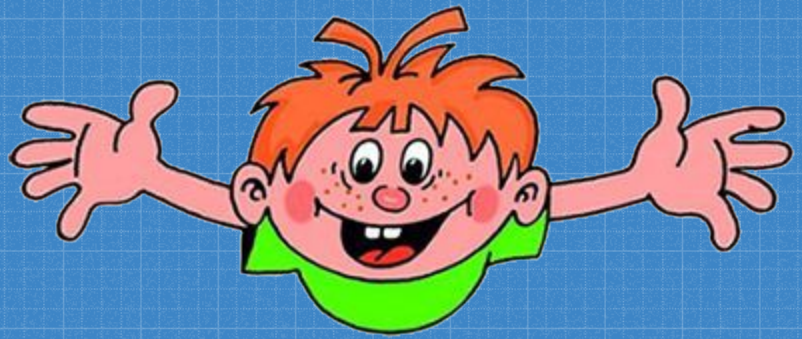
Easy to Build with Compose

Our new **Jetpack Compose** library and Studio support for Widgets helps you create great-looking, responsive, and backwards compatible widgets in less time (and code)



Conceptual code

```
@Composable
fun ListWidget(items: List<Item>) {
  Column {
    Text("Shopping list")
    items.forEach { CheckListItem(it) }
  }
}

@Composable
fun CheckListItem(item: Item) {
  CheckBox(
    item.name,
    isChecked=item.isChecked,
    onStateChanged={isChecked -> updateItem(item, isChecked)}
  )
}
```



TradingView

Мы набираем крутых iOS & Android разработчиков!

Спасибо!

ВОПРОСЫ?

Вы можете найти меня:

@makedonsky94

nevyantsev.alexandr@gmail.com