Разработка мобильных приложений

То, что скрыто от глаз

План

1. Потоки
2. Сетевые запросы
3. Сервисы
4. Архитектурные решения

Потоки

Demo: Thread1

Главный поток

1. Правило 16 мс (60 кадров в секунду).
2. Проблема запаздывания реакции системы при загрузке главного потока тяжелой работой.
3. ANR.

ANR

1. I/O - operation
2. Long-running operation
3. Wait/sleep/lock
4. Deadlock (demo: Deadlock)
5. data/anr/traces.txt
6. https://developer.android.com/topic/performance/vitals/anr.html

Главный поток

1. Activity Lifecycle Callbacks
2. Service Lifecycle Callbacks
3. ContentProvider Lifecycle Callbacks
4. Application Lifecycle Callback
5. Broadcast#onReceive()
6. View Lifecycle Callbacks

Demo: Thread2

Фоновый поток

1. UI можно обновлять только с главного потока: handler.post, runOnUiThread, view.post, view.postDelayed.
2. ContentProvider#(insert, update, query, e.t.c.).

Demo: Thread3

Аннотации

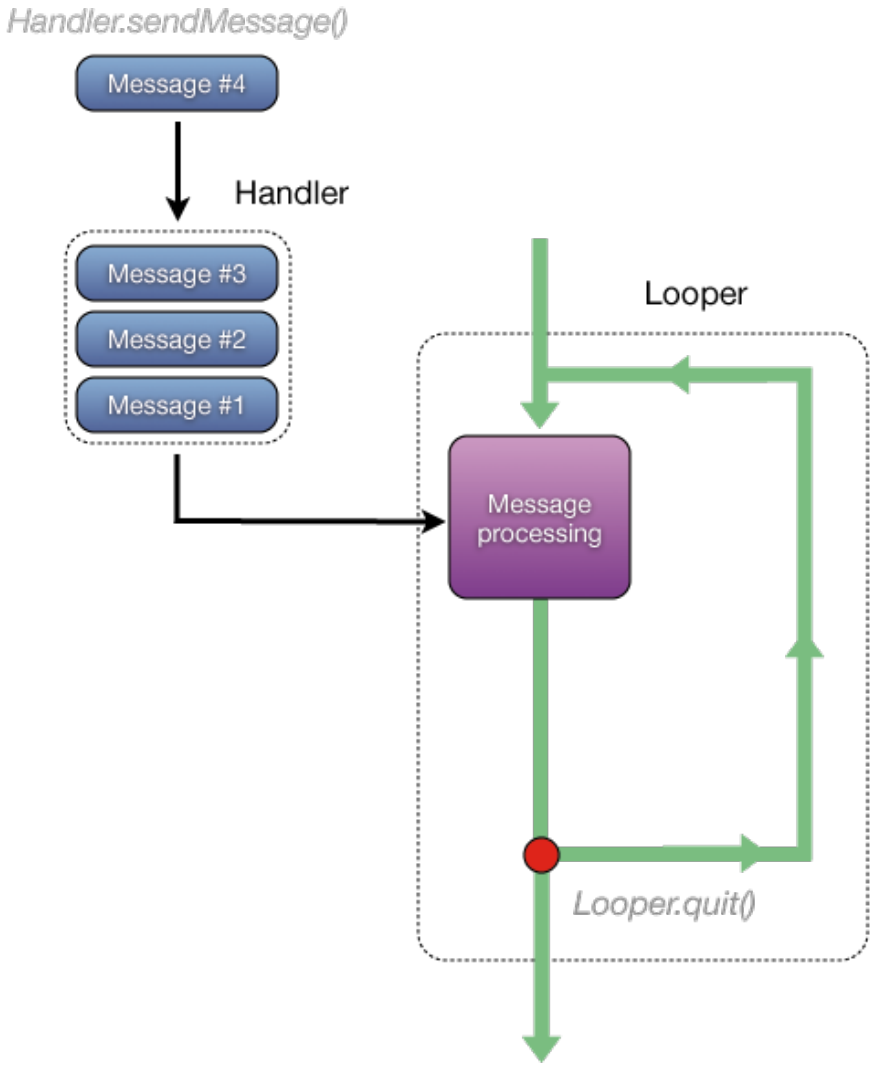
- @MainThread
- @WorkerThread
- https://developer.android.com/studio/write/annotations.html - thread-annotations

Demo: Thread4

Поток

1. Thread.
2. HandlerThread = Thread + Looper.

Handler


Handler.sendMessage()
Message #4
Handler
Message #3
Message #2
Message #1
Looper
Message processing
Looper.quit()

Handler

1. https://developer.android.com/training/multiple-threads/communicate-ui.html
2. https://developer.android.com/reference/android/os/Handler.html

Demo: HandlerThread

AsyncTask

1. https://developer.android.com/reference/android/os/AsyncTask.html
2. Deprecated в Android 11

AsyncTask: проблемы

- Взаимодействие с не существующими более компонентами активити
- Утечка памяти: ссылка на активити хранится для onPostExecute
- Потеря результатов
- При пересоздании активити (e.g. поворот) ссылка станет недействительной

Demo: AsyncTask

Что еще?

Executor & Executors:

https://developer.android.com/reference/java/util/concurrent/Executor.html

https://developer.android.com/reference/java/util/concurrent/Executors.html

https://developer.android.com/reference/java/util/concurrent/ThreadPoolExecutor

Сетевые операции

Сетевые соединения

1. Wifi(a/b/g/n/ac)
2. Mobile networks: GPRS, EDGE, UTMS, HSDPA, HSUPA, HSPA, HSPA+, LTE.

Demo: ConnectionTypes

Оптимизация

1. Настройки: когда можно ходить в сеть.
2. Data Saver
3. Предзагружать данные, которые скорее всего будут использоваться.
4. Запускаться из кэша, если возможно.
5. Группировать запросы.
6. Cache-control, etag.

DataSaver

https://developer.android.com/training/basics/network-ops/data-saver

Demo: DataSaver

UrlConnection

1. TLS.
2. Streaming uploads and downloads.
3. Timeouts.
4. IPv6.

UrlConnection

1. https://developer.android.com/training/basics/network-ops/connecting.html
2. https://developer.android.com/reference/java/net/HttpURLConnection.html
3. https://developer.android.com/reference/java/net/URLConnection.html

Demo: ImageLoaderMain

Предупреждение!

Не ходите в сеть с главного потока!

Demo: ImageLoaderAsyncTask

OkHttp

https://github.com/codepath/android_guides/wiki/Using-OkHttp

Demo: ImageLoaderOkHttp

Demo: ImageLoaderOkHttpAsync

Сервисы

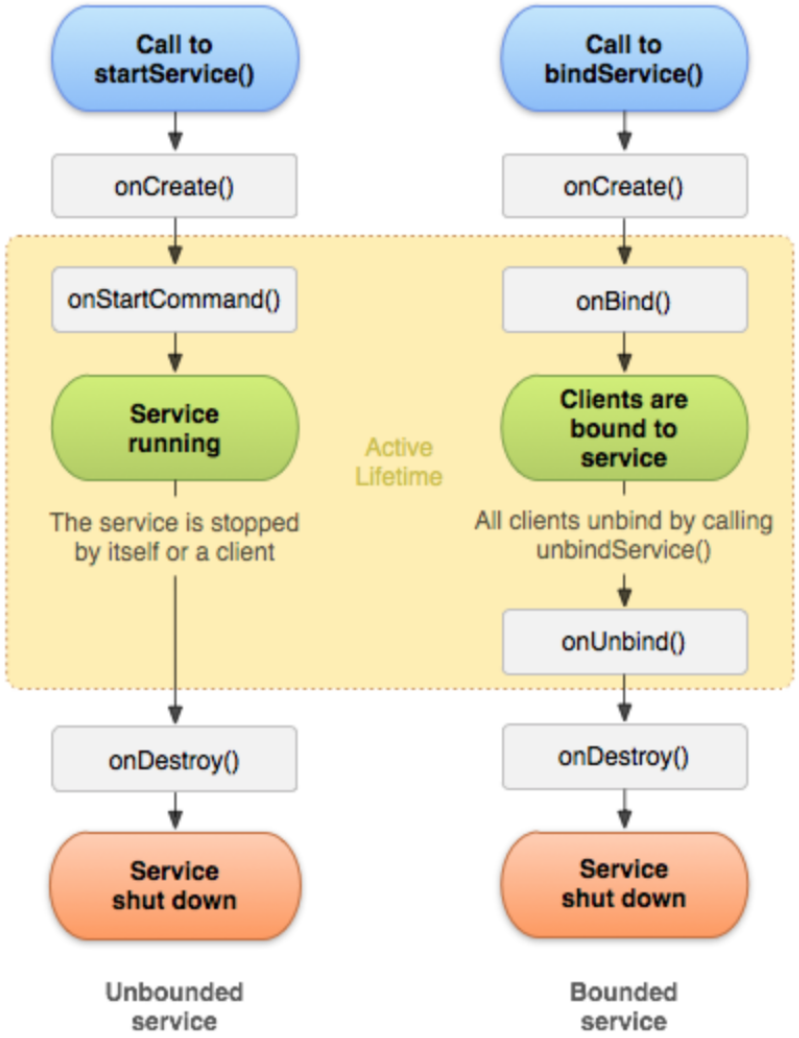
Приоритет компонентов

- Foreground: видимая Activity, BroadcastReceiver, колбэки сервисов
- Visible: свернутая активити, foreground сервис
- Service: запущенный сервис
- Cached: старые свернутые активити, все остальное неважное

Сервисы

1. Запущенные фоновые
2. Привязанные
3. Запущенные foreground

Сервисы


Call to startService()
onCreate()
onStartCommand()
Service running
The service is stopped by itself or a client
onDestroy()
Service shut down
Unbounded service
Call to bindService()
onCreate()
onBind()
Clients are bound to service
All clients unbind by calling unbindService()
onUnbind()
onDestroy()
Service shut down
Bounded service
Active Lifetime

Сервисы

1. https://developer.android.com/guide/components/services.html.

Bounded Services

1. https://developer.android.com/guide/components/bound-services.html.

Bounded Services

1. Binder: если используется тем же приложением в том же процессе
2. AIDL
3. Messenger: использует AIDL, очередь запросов на одном потоке

Demo: ImageLoaderService

Foreground Service

- Нужно разрешение android.permission.FOREGROUND_SERVICE
- startService – как и с обычным Service
- Нужно создать уведомление: установить иконку и текст, а также Intent, который будет выполняться по клику
- Вызвать startForeground
- stopForeground

Demo: ImageLoaderFgService

Service vs IntentService

	Service	IntentService
Нужно создавать свой Executor для задач	+	-
Нужно следить за жизненным циклом	+	-

IntentService

1. https://developer.android.com/reference/android/app/Service.html

Demo: ImageLoaderIntentService

JobService

1. Можно задавать рекомендации по запуску.
2. Запуском задачи управляет JobScheduler.
3. Существует лимит на длительность выполнения задачи (около 10 минут).
4. Выполняется на главном потоке
5. Нужно переопределить onStartJob, onStopJob (если система прервала выполнение) и вызывать jobFinished по завершении работы

JobService. Рекомендации по запуску

- setMinimumLatency – запустить не раньше, чем
- setOverrideDeadline – запустить не позже, чем
- setPeriodic – периодически повторять
- setPersisted – важно ли запустить даже после перезагрузки
- setRequiredNetworkType
- setRequiresBatteryNotLow
- setRequiresCharging
- setRequiresDeviceIdle
- setRequiresStorageNotLow

JobService. Доп. параметры

- setTransientExtras – передать любые параметры в onStartJob
- setBackoffCriteria – поддержка ретраев

JobService

1. https://developer.android.com/reference/android/app/job/JobScheduler.html
2. https://developer.android.com/topic/performance/scheduling.html

Demo: ImageLoaderJobService

JobIntentService

1. https://developer.android.com/reference/android/support/v4/app/JobIntentService.html

Demo: CalculatorJobIntentService

WorkManager

https://developer.android.com/topic/libraries/architecture/workmanager/

WorkManager – обертка над:

1. JobScheduler
2. Firebase JobDispatcher
3. AlarmManager + BroadcastReceiver

WorkManager

- Нужен, когда задачу необязательно выполнять немедленно, но выполнить надо, даже если был ребут
- Позволяет задавать input/output
- Позволяет создавать очередь зависимых задач
- Позволяет создавать уникальные задачи: задача с таким же именем второй раз проигнорируется
- Позволяет подписываться на изменение статуса задачи
- doWork вызывается на фоновом потоке

Demo: ImageLoaderWorkManager

DownloadManager

- Работает на фоновом потоке
- Повторят попытки в случае неудачи или проблем с соединением
- Можно отменять
- Отображает прогресс в уведомлении
- Рассылает уведомления о завершении загрузки

DownloadManager

https://developer.android.com/reference/android/app/DownloadManager

Demo: ImageLoaderDownloadManager

Что еще?

- AlarmManager: https://developer.android.com/reference/android/app/AlarmManager

Архитектурные решения

Типовые проблемы

1. Фоновая загрузка данных.
2. Синхронизация жизненного цикла UI и фоновой загрузки данных.
3. Слежение за изменением данных.
4. Хранение состояния UI.

Рекомендуемые принципы

1. "Разделяй и властвуй".
2. UI должен обновляться из одного источника данных.
3. На UI потоке нужно только обновлять UI.
4. При обновлении UI необходимо учитывать жизненный цикл activity и fragment.

Рекомендуемые принципы

5. Подготовку данных для UI необходимо осуществлять на фоновом потоке.

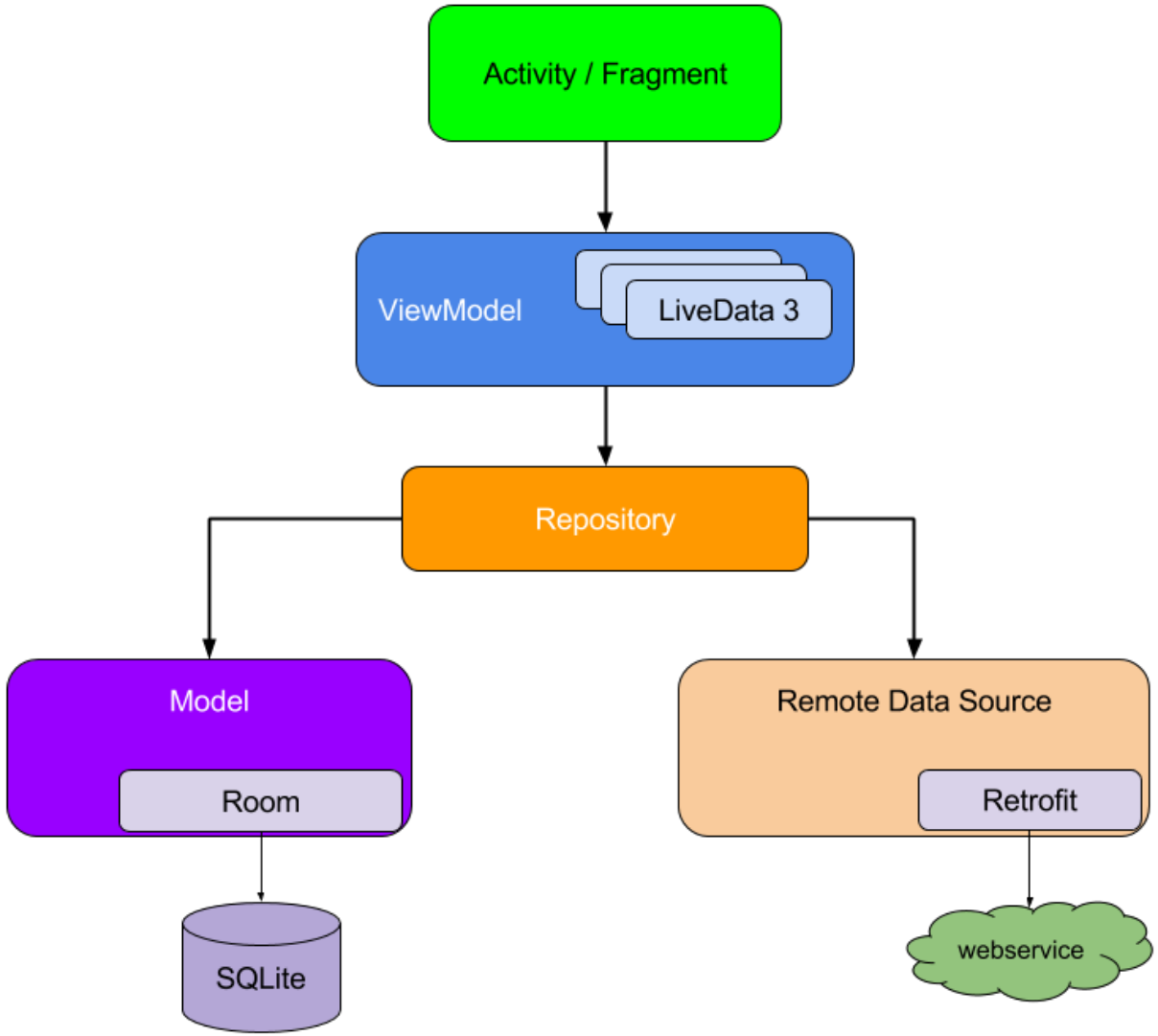
6. Всю работу, не связанную с подготовкой данных для UI, лучше выносить на сервис.

Android Architecture Components

1. https://developer.android.com/topic/libraries/architecture/guide.html

Android Architecture Components

1. LiveData
2. ViewModel
3. Room Persistence Library
4. Paging Library
5. WorkManager
6. …..


Activity / Fragment
ViewModel
LiveData 3
Repository
Model
Room
SQLite
Remote Data Source
Retrofit
webservice

LiveData

1. Слежение за обновлением данных в контексте жизненного цикла компонентов приложения.
2. Объединение данных из разных источников - MediatorLiveData.
3. Подготовка данных к отображению на UI.
4. После разворота экрана callback сразу дергается с имеющимися данными

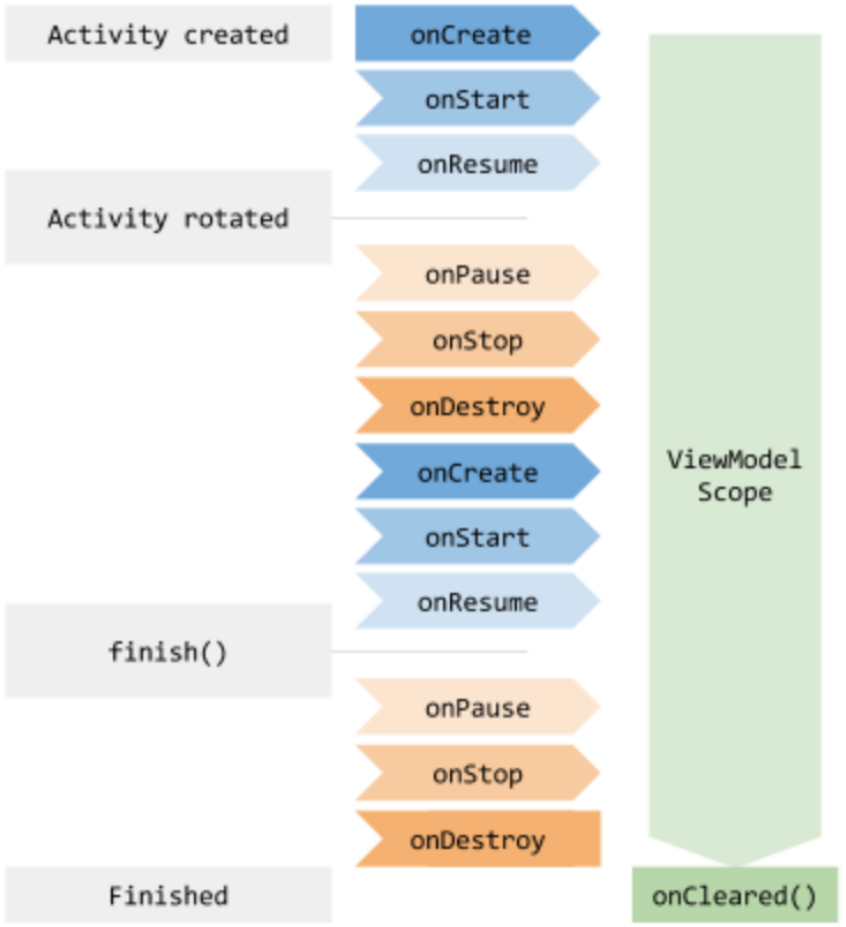
LiveData

1. https://developer.android.com/topic/libraries/architecture/livedata.html

ViewModel

1. Хранение и управление данными, связанными с UI
2. Продолжают существовать, когда меняется конфигурация
3. Уничтожается при уничтожении активити
4. Не должен содержать ссылки на View
5. Позволяет легко обмениваться данными между UI компонентами

ViewModel


Activity created
onCreate
onStart
onResume
Activity rotated
onPause
onStop
onDestroy
onCreate
onStart
onResume
finish()
onPause
onStop
onDestroy
Finished
ViewModel Scope
onCleared()

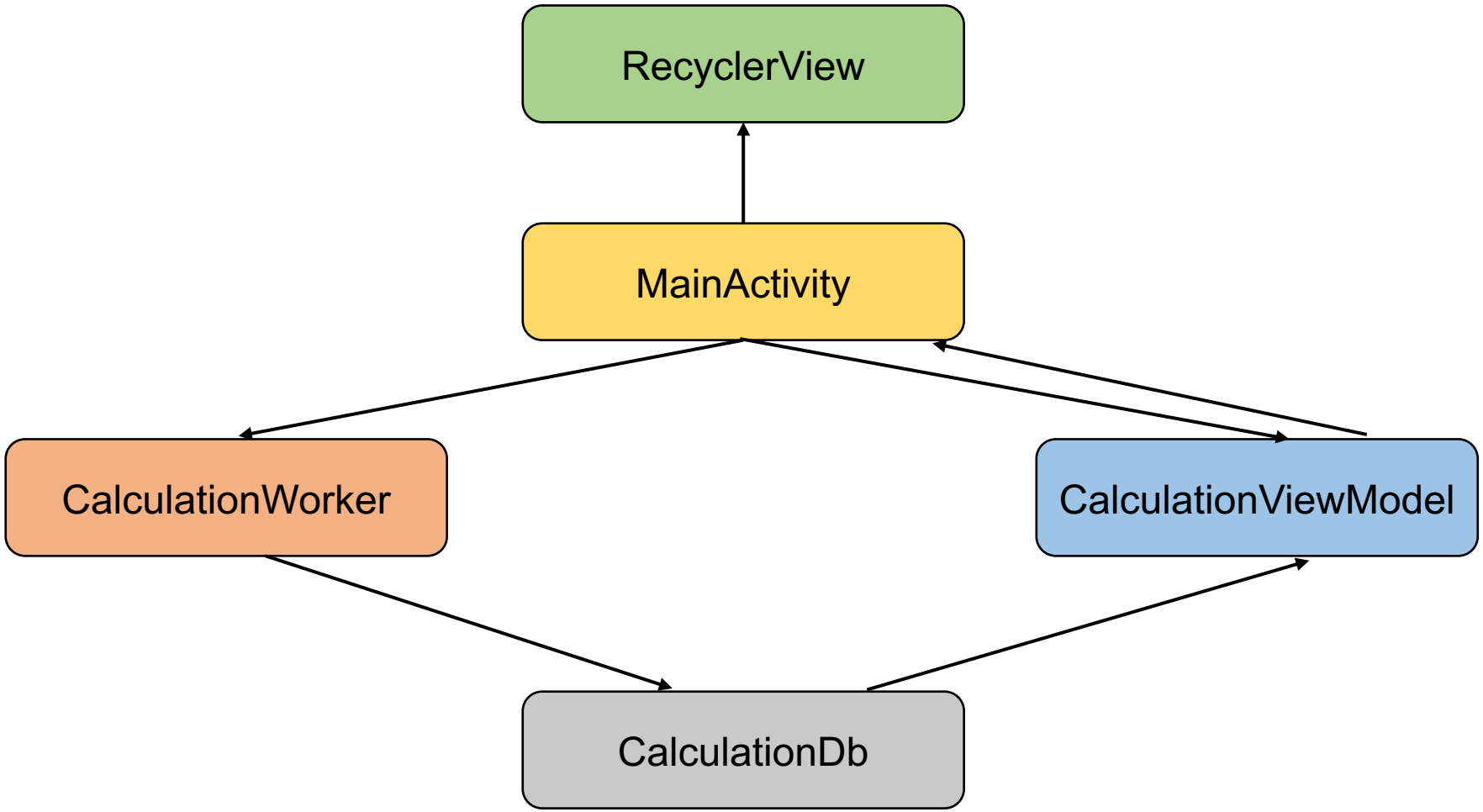
ViewModel

1. https://developer.android.com/topic/libraries/architecture/viewmodel.html

Room Persistence Library

1. https://developer.android.com/training/data-storage/room/index.html

Demo: FactorialCalculator


RecyclerView
MainActivity
CalculationWorker
CalculationViewModel
CalculationDb

Что осталось за кадром

- Корутины: https://kotlinlang.org/docs/reference/coroutines-overview.html
- Paging library: https://developer.android.com/topic/libraries/architecture/paging