SKILLBRANCH
Образовательное IT- сообщество

Компоненты платформы Android

Жизненный цикл Activity

API Level

Версия платформы	Уровень API	Ключевые особенности
Android 1.0	1	Первая стабильная версия системы. Появился магазин приложений Android Market
Android 1.5	3	Поддержка виджетов и папок на рабочем столе. Запись и воспроизведение видео в MPEG-4 и 3GP Поддержка Bluetooth-профиля A2DP и AVRCP Обновление WebKit и Squirrelfish Javascript Engine.
Android 2.0	5	Новый пользовательский интерфейс браузера и поддержка HTML5 Добавлена поддержка новых размеров и разрешений экранов.
Android 2.2.x	8	Интеграция в браузер JavaScript-движка V8, ранее реализованного в Chrome. Поддержка установки приложений в расширенную память
Android 2.3.2 Android 2.3.1 Android 2.3	9	Встроенная поддержка протокола SIP VoIP-телефонии Системная поддержка копирования и вставки Параллельная сборка мусора для улучшения производительности Встроенная поддержка большего числа сенсоров (например, гироскопы и барометры) Менеджер скачивания для длительных загрузок
Android 3.0.x	11	Поддержка многоядерных процессоров Улучшенная поддержка планшетов благодаря новому UI Улучшенная многозадачность Поддержка аппаратного ускорения
Android 3.2	13	Внесены оптимизации для поддержки более широкого спектра планшетов Лёгкий доступ приложений к файлам на SD-карте, например для синхронизации

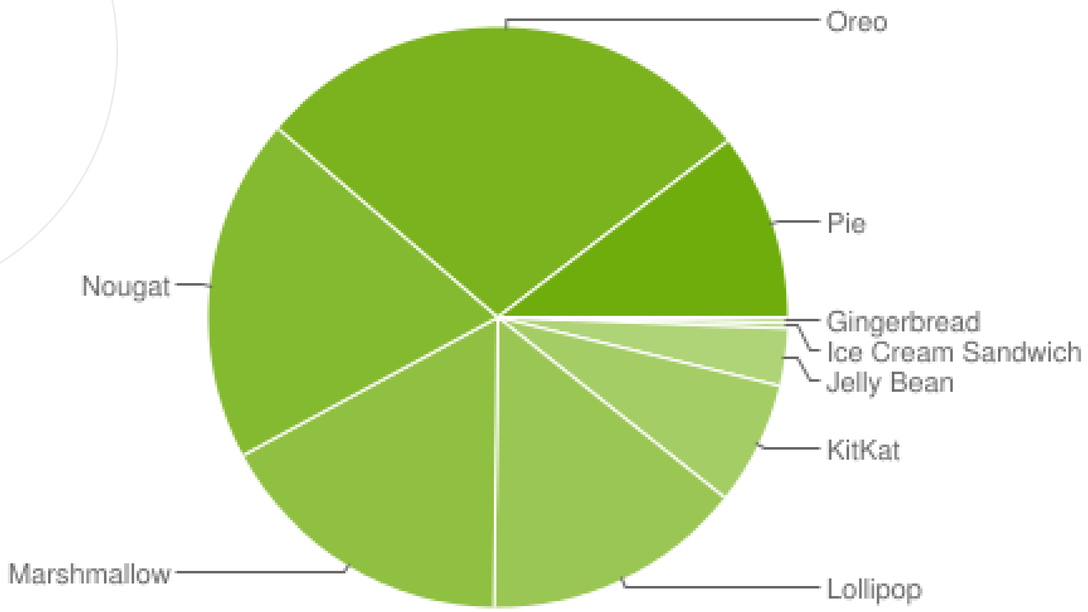
API Level

Версия платформы	Уровень API	Ключевые особенности
Android 4.0, 4.0.1, 4.0.2	14	Использование единой оболочки для планшетов, смартфонов и других устройств на базе OS Android Обновлённый браузер с поддержкой вкладок, синхронизацией закладок Google Chrome. Также обновлено ядро WebKit и движок V8
Android 4.2, 4.2.2	17	Обновление ядра Linux до ветки 3.4. поддержка профилей, теперь на одном устройстве может использоваться несколько учётных записей
Android 4.3	18	Smart или Bluetooth 4.0 LowEnergy включен в новую прошивку, что повышает энергоэффективность устройства. В области уведомлений теперь показаны все работающие приложения, даже в фоновом режиме. В настройках можно установить постоянную работу Wi-Fi для улучшенного гео-позиционирования. Доступна новая камера и галерея.
Android 4.4	19	Более быстрое переключение между задачами и оптимизированное распределение памяти. Осуществление NFC-платежей через Google Wallet и другие платежные системы. В стандартную клавиатуру добавлены Emoji
Android 5.0	21	Material design Теперь вместо компилятора Dalvik используется компилятор ART. Project Volta, благодаря которому операционная система обращается к процессору не одиночными запросами, а пакетами данных
Android 6.0	23	Настройка разрешений Защита данных с помощью отпечатка пальца Увеличена продолжительность работы от батареи (Doze Mode)

API Level

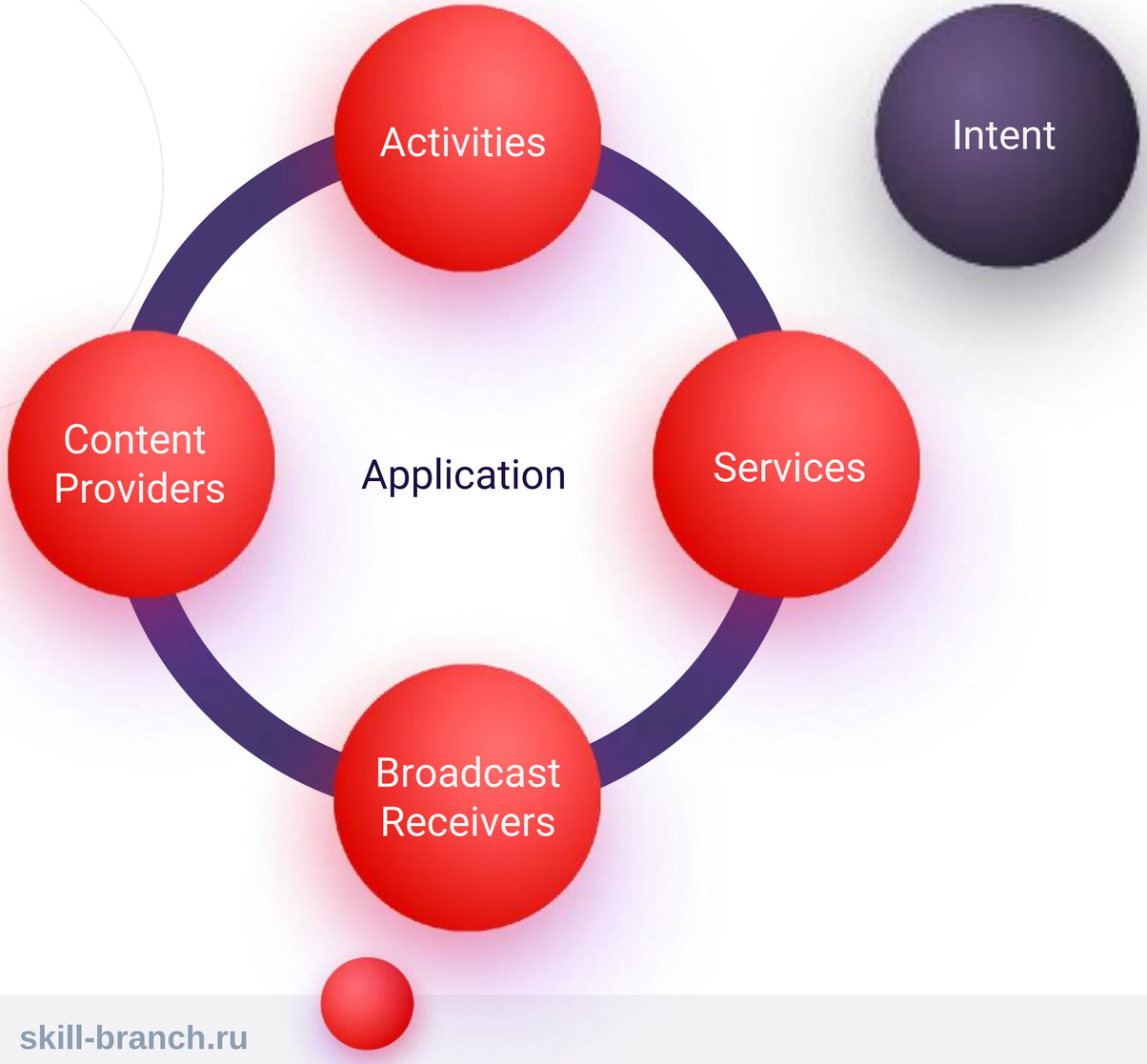
Версия платформы	Уровень API	Ключевые особенности
Android 7, 7.1	**24, 25**	Режим многооконного разделения экрана Улучшены уведомления, появилась возможность быстрого ответа. Ночной режим позволяет добиться оптимального отображения информации на экране с помощью автоматического повышения контрастности и яркости Усовершенствована функция энергосбережения «Doze» App shortcuts API.
Android 8	**26, 27**	Новый внешний вид уведомлений, каналы уведомлений для разных типов уведомлений Ограничение фоновых процессов, фонового отслеживания местоположений, сканирования WiFi. «Умное» выделение текста. Картинка в картинке Распознавание экранных жестов Автоматические светлые и темные темы
Android 9	**28**	Новый дополнительный системный интерфейс на основе жестов, позволяющий пользователям перемещаться по ОС с помощью swipes Расширенная версия режима «Не беспокоить» «Адаптивная батарея», которая использует Doze для спячки пользовательских приложений Функция автоматической яркости изменяет яркость экрана на основе пользовательских привычек Поддержка выреза на безрамочных устройствах (“бровь”)

API Level в процентах использования



Версия	Кодовое имя	API Level	Распространение
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Компоненты Android



Activity – UI приложения

- Может занимать весь экран или его часть
- Может быть запущена из других компонент приложения или из другого приложения
- Activity может возвращать результат

Service - компонент для выполнения длительных фоновых задач

- Не содержит графического интерфейса
- Может выполняться в том же процессе, что и само приложение, либо в отдельном
- Повышает приоритет процесса с точки зрения Android

Broadcast Receiver – приемник широковещательных сообщений

- Получает сообщения от Android или других приложений
- Должен обрабатывать сообщения быстро, длительные задачи можно делегировать сервису

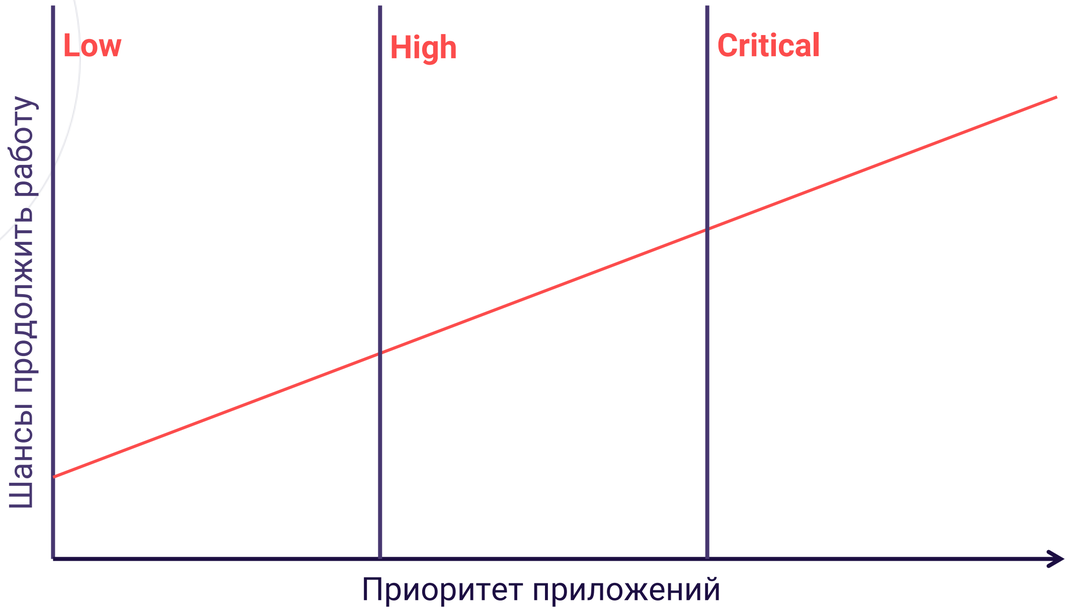
Content Provider – компонент для доступа к хранилищу данных

- Используется для доступа к данным, хранимым Android или другими приложениями
- Приложение может давать доступ к своим данным для других приложений, реализуя Content Provider
- Представляет данные в виде таблиц, реализует методы query, Insert, update, delete

Intent - сущность для описания операций, которую требуется выполнить

- Запуск Activity, Service, отправка Широковещательных сообщений
- Выполнения стандартных, предопределенных операций

Application Priority



Critical

- Активные Activity (с которыми происходит взаимодействие пользователя),
- Фоновые Service

High

- Видимые Activity,
- Запущенные Service (обновления/синхронизация)

Low

- Приложения в фоне (свернутые приложения)

Структура проекта

```
dev-intensive-2019
├─ app
│  ├─ libs
│  ├─ build
│  ├─ src
│  │  ├─ androidTest
│  │  │  └─ java
│  │  │     └─ ru.skillbranch.devintensive
│  │  ├─ test
│  │  │  └─ java
│  │  │     └─ ru.skillbranch.devintensive
│  │  └─ main
│  │     ├─ java
│  │     │  └─ ru.skillbranch.devintensive
│  │     ├─ assets
│  │     ├─ res
│  │     └─ AndroidManifest.xml
│  ├─ build.gradle
│  └─ proguard-rules.pro
├─ build.gradle
├─ local.properties
└─ settings.gradle
```

```
res
├─ anim / *.xml // Tween-анимация
├─ animator / *.xml // Анимация свойств
├─ drawable /*.png *.9.png *.jpg *.gif *.xml //Изображения
├─ layout / *.xml //пользовательский интерфейс
├─ menu / *.xml //Меню приложения
├─ mipmap / *.xml //Иконки
├─ raw / *.xml //Различные файлы в исходном виде (mp3, json)
├─ transition / *.xml //Анимация переходов
├─ values
│  ├─ strings.xml //Строки
│  ├─ colors.xml //Цвета
│  ├─ dimens.xml //Размеры
│  ├─ attr.xml //кастомные атрибуты View
│  └─ styles.xml //Стили
└─ xml / *.xml //Произвольные xml файлы
```

Наименование статических ресурсов

colors.xml

```
<resources>
    <!-- grayscale -->
    <color name="white">#FFFFFF</color>
    <color name="gray_light">#DBDBDB</color>
    <color name="gray">#939393</color>
    <color name="gray_dark">#5F5F5F</color>
    <color name="black">#323232</color>

    <!-- basic colors -->
    <color name="green">#27D34D</color>
    <color name="blue">#2A91BD</color>
    <color name="orange">#FF9D2F</color>
    <color name="red">#FF432F</color>
</resources>
```

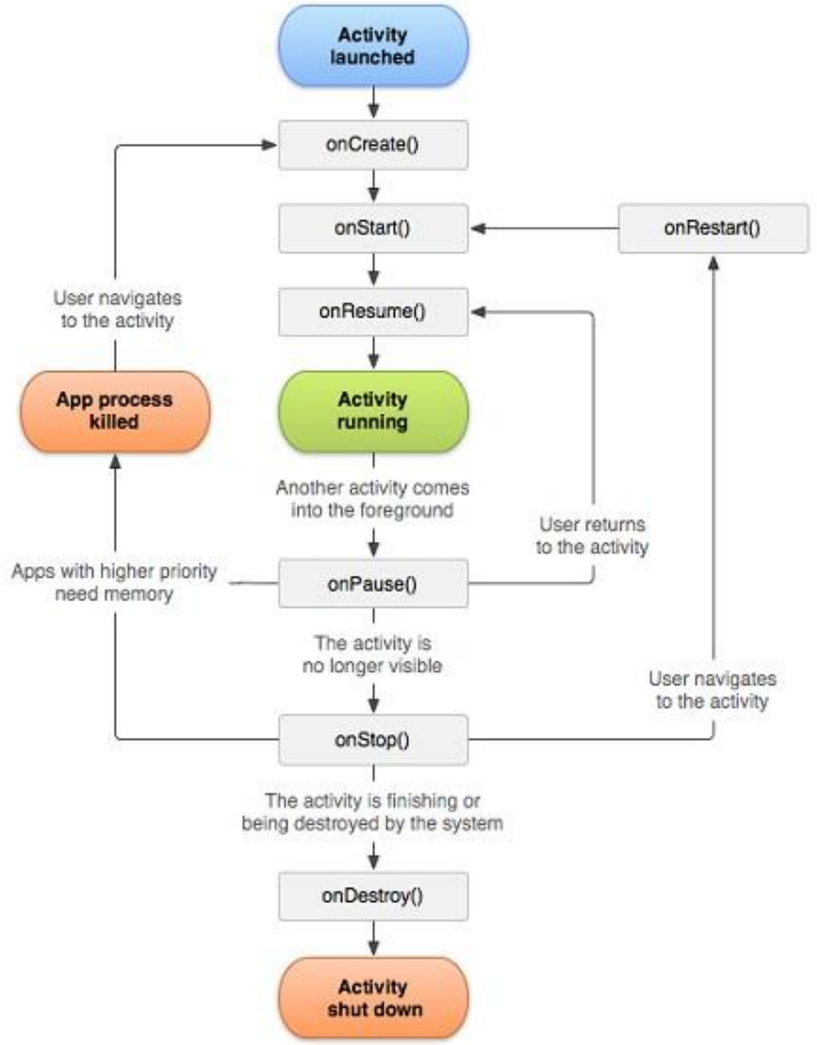
dimens.xml

```
<resources>
    <!-- font sizes -->
    <dimen name="font_large_24">24sp</dimen>
    <dimen name="font_normal_16">16sp</dimen>
    <dimen name="font_small_12">12sp</dimen>

    <!-- typical spacing between two views -->
    <dimen name="spacing_huge_32">32dp</dimen>
    <dimen name="spacing_large_24">24dp</dimen>
    <dimen name="spacing_normal_16">16dp</dimen>
    <dimen name="spacing_small_8">8dp</dimen>
    <dimen name="spacing_tiny_4">4dp</dimen>

    <!-- typical sizes of views -->
    <dimen name="icon_size_normal_24">24dp</dimen>
</resources>
```

strings.xml

```
<resources>
    <string name="error.message.network">Network error</string>
    <string name="error.message.call">Call failed</string>
    <string name="error.message.map">Map loading failed</string>
</resources>
```

Activity lifecycle



onCreate() вызывается при первом создании или перезапуске Activity

- задаётся внешний вид Activity (UI) через метод setContentView().
- инициализируются представления и модели
- представления связываются с необходимыми данными и ресурсами

onStart() UI еще не виден пользователю, но вскоре будет виден, вызывается непосредственно перед тем, как Activity становится видимой пользователю.

- чтение из базы данных
- запуск сложной анимации
- запуск потоков, отслеживания показаний датчиков, запросов к GPS, сервисов или других процессов, которые нужны исключительно для обновления пользовательского интерфейса

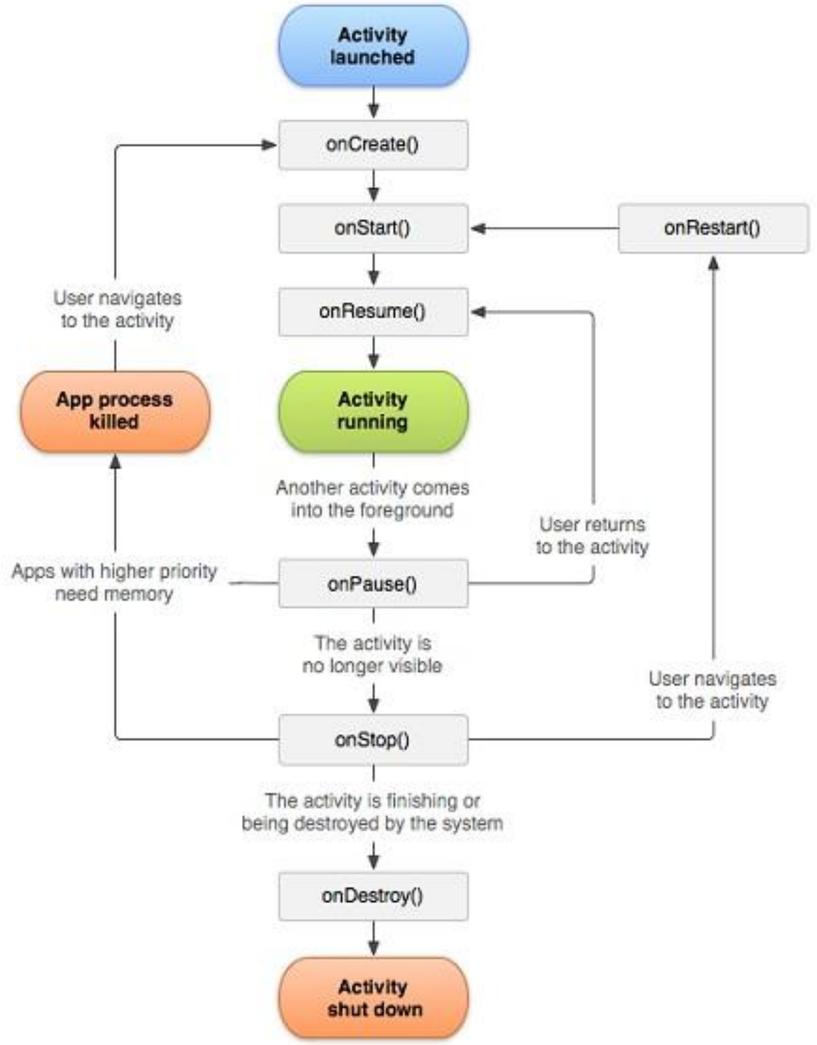
onRestart() вызывается если Activity возвращается в приоритетный режим после вызова onStop(), т.е. вызывается после того, как Activity была остановлена и снова была запущена пользователем. Всегда сопровождается вызовом метода onStart().

- используется для специальных действий, которые должны выполняться только при повторном запуске Activity

onResume() вызывается, когда Activity начнет взаимодействовать с пользователем.

- запуск воспроизведения анимации, аудио и видео
- регистрации любых BroadcastReceiver или других процессов, которые вы освободили/приостановили в onPause()
- выполнение любых другие инициализации, которые должны происходить, когда Activity вновь активна.

Activity lifecycle



onPause() вызывается после сворачивания текущей активности или перехода к новому. От onPause() можно перейти к вызову либо onResume(), либо onStop().

- остановка анимации, аудио и видео
- сохранение состояния пользовательского ввода (легкие процессы)
- сохранение в DB если данные должны быть доступны в новой Activity
- остановка сервисов, подписок, BroadcastReceiver

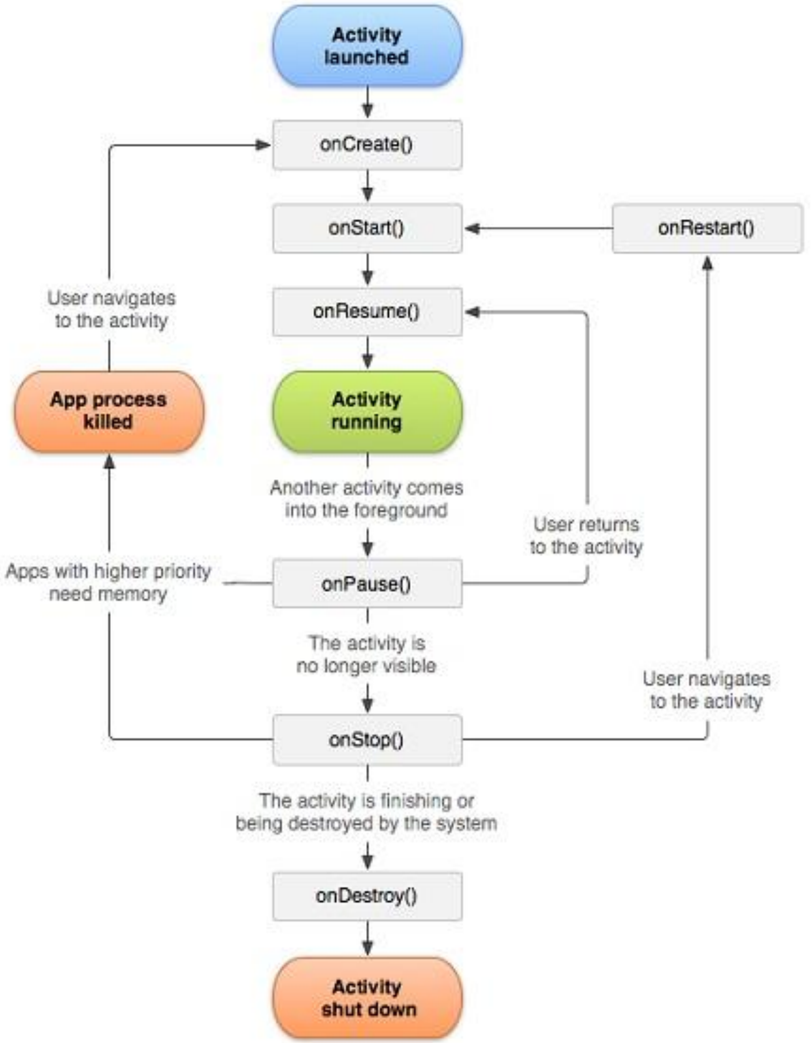
onStop() вызывается, когда Activity становится невидимым для пользователя. Это может произойти при её уничтожении, или если была запущена другая Activity (существующая или новая) перекрывшая окно текущей Activity.

- запись в базу данных
- приостановка сложной анимации
- приостановка потоков, отслеживания показаний датчиков, запросов к GPS, таймеров,
- сервисов или других процессов, которые нужны исключительно для обновления пользовательского интерфейса

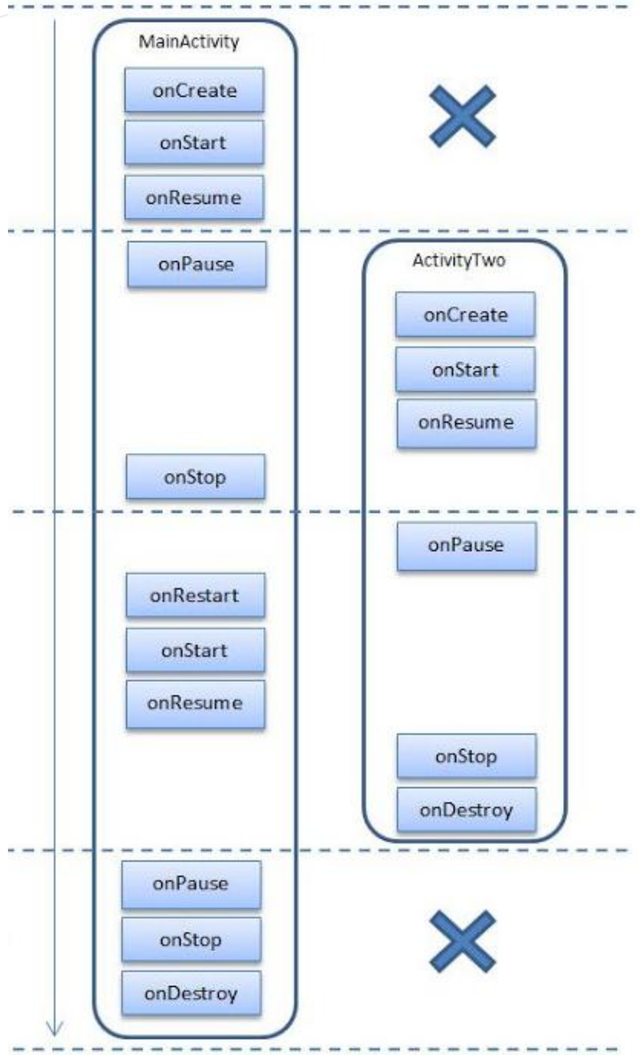
OnDestroy() вызывается по окончании работы Activity (пользователь закрывает приложение через клавишу back, или удаляет из списка активных приложений), при вызове метода finish()

- высвобождение ресурсов
- дополнительная перестраховка если ресурсы не были выгружены или процессы не были остановлены ранее

Activity lifecycle на примере 2 Activity



- Запуск приложения **Activity A**
- Переход к **Activity B**
- Возврат к **Activity A**
- Выход из приложения



Use case Activity lifecycle

Запуск приложения
onCreate() → onStart() → onResume()

Нажимаем кнопку Back для выхода из приложения
onPause() → onStop() → onDestroy()

Нажата кнопка Home
onPause() → onStop()

После нажатия кнопки Home, когда приложение запущено из списка недавно открытых приложений или через значок
onRestart() → onStart() → onResume()

Когда запускается другое приложение из области уведомлений
onPause() → onStop()

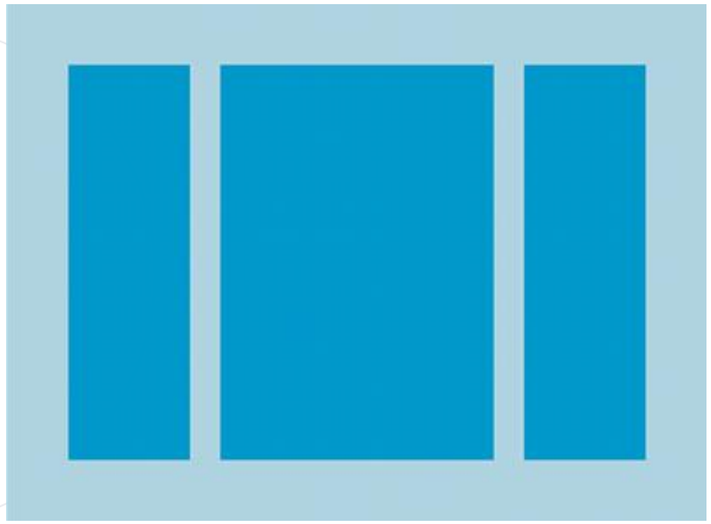
Нажата кнопка Back в другом приложении или в Настройках и ваше приложение стало снова видимым.
onRestart() → onStart() → onResume()

Пользователь отвечает на звонок → Разговор окончен
onPause() → onResume()

Экран телефона гаснет
onPause() → onStop()

Экран снова включён
onRestart() → onStart() → onResume()

Linear Layout



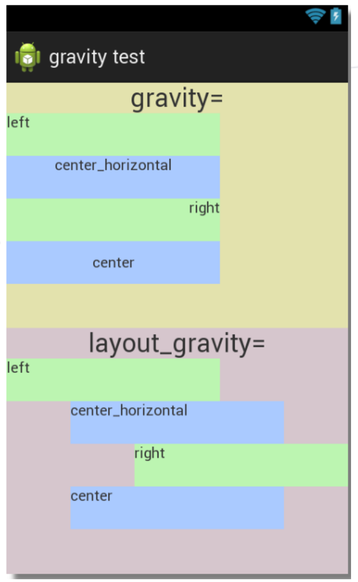
Размещает все дочерние View элементы в одном направлении – горизонтально или вертикально

android:orientation
vertical - расположение элементов сверху вниз
horizontal - расположение элементов слева направо

android:layout_weight - индивидуальный вес для дочернего элемента. Этот атрибут определяет "важность" представления и позволяет этому элементу расширяться, чтобы заполнить любое оставшееся пространство в родительском представлении. Заданный по умолчанию вес является нулевым.

android:gravity
Выравнивание внутреннего содержимого View элемента
top/bottom/left/right
center/center_vertical/center_horizontal

android:layout_gravity
Выравнивание View элемента относительно его родителя
top/bottom/left/right
center/center_vertical/center_horizontal



Организационные вопросы

Будущее Devintensive

останется в открытом доступе и будет доступен для прохождения

LMS новые фичи

frontend реализован с применением socket.io (обновление характеристик в реальном времени)
переработана система проверки тестов
добавлен функционал просмотра stdOut, stdErr в результате прохождения теста
добавлен функционал поддержать проект для web версии и через Telegram Payment API

Проверка 3 практического задания

начнется после тестирования менторами и отладки – предположительно 10.07.2019

Summary по практическим заданиям

будет реализовано и добавлено в личный кабинет после окончания курса

SKILLBRANCH
Образовательное IT- сообщество

Спасибо за внимание

пиши красивый код