Курс Kotlin Jetpack Compose

Лекция 1

ОС Android. Среда разработки Android Studio. Приложение HelloWorld

Д. Макаренков

https://dmpsy.club

JetpackCompose <= (Kotlin && Android)

Целевая аудитория

- Энтузиасты программирования на Kotlin для Android, желающие следовать современным принципам разработки пользовательских интерфейсов (UI) с использованием библиотеки Jetpack Compose

История названия:

Andy Rubin, “отец-основатель” ОС Android был настолько увлечен роботами, что друзья даже прозвали его Андроидом.

(см. https://en.wikipedia.org/wiki/Andy_Rubin)

Варианты разработки моб. приложений

Поставщик	Apple	Samsung, Huawei, Google, Vivo, Xiaomi *etc.*	Huawei
Операционная система	iOS	Android OS	HarmonyOS
Установочные файлы	*.dmg (installer) *.ipa (archive)	*.apk (installer), *.aab (bundles)	*.app (installer), *.hap (bundles)
Среда разработки (IDE)	Xcode	Android Studio	Dev Eco Studio
“Родные” ЯВУ	Swift Objective C	Kotlin (JetPack Compose) Java (не рекомендован Google Play с 2019)	Java, JS, eTS, ArkTS
Популярные альтернативы	React Native (JS) KMM (Kotlin Multiplatform Mobile)	React Native (JS)	Kotlin (?) React Native (?)

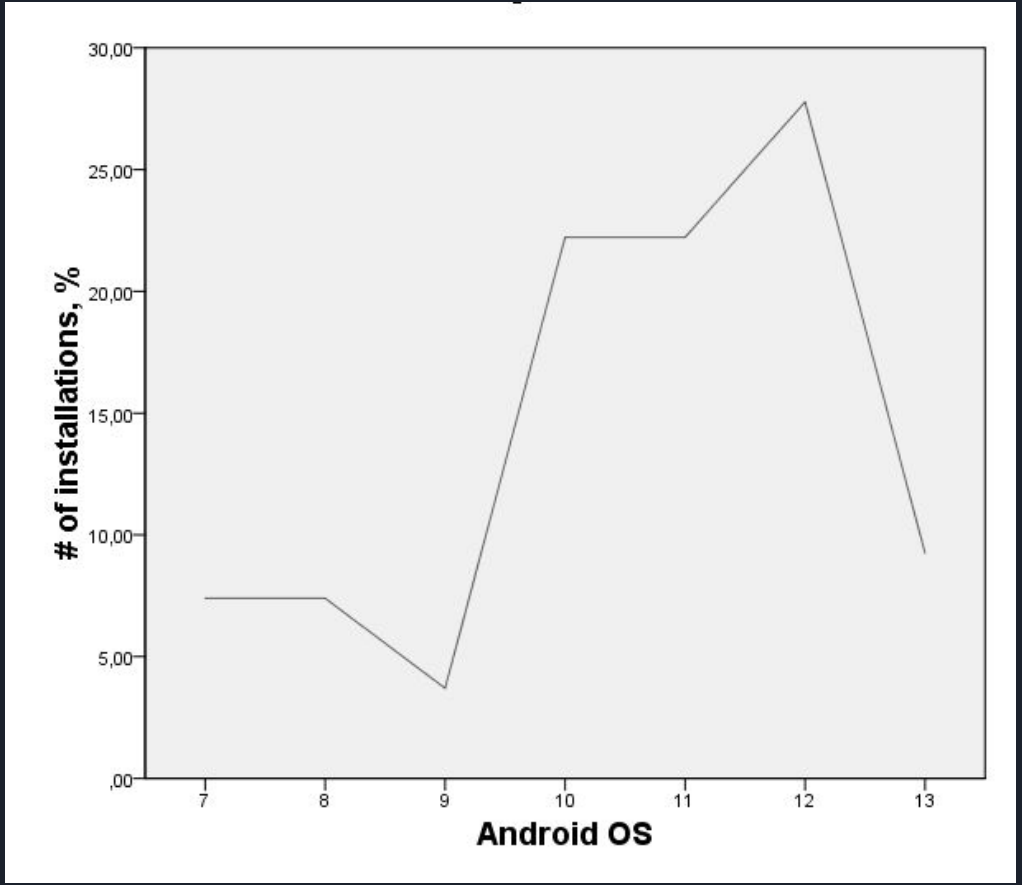
ОС Android: Вводные замечания

- Первый релиз ОС Android датируется 2008 г. (спустя год после выпуска iPhone)
- ОС Android - продукт Google с открытым кодом, основан на Linux
- Приложения запускаются под управлением виртуальной машины Java (JVM), также производства Google
- Java как язык разработки постепенно вытесняется Kotlin’ом, считающимся “предпочтительным”, начиная с 2019 г.
- Google поставляет инструментарий SDK для разработки приложений под каждый уникальный уровень API.
- Для удобства разработки, используйте последнюю версию Android Studio IDE, наиболее популярную среду, основанную на IntelliJ IDEA от JetBrains
- Выходные форматы файлов: (*.apk, installers) или (*.aab, bundles)
- Известные магазины приложений: Google Play, AppGallery, RuStore (РФ)

Используемые версии ОС Android OS и SDK

Название ОС	Версия ОС	Уровень API (SDK)
Android 13, Tiramisu	13	33
Android 12L, Snow Cone	12.1	32
Android 12, Snow Cone	12	31
Android 11, R, Red Velvet Cake	11	30
Android 10, Q, Quince Tart	10	29
Pie	9	28
Oreo	8.1.0	27
	8.0.0	26
Nougat	7.1	25
	7.0	24
Marshmallow	6.0	23
Lollipop	5.1	22
	5.0	21

Статистика (%) скачиваний Eysenck App с RuStore



Основные параметры приложений Android

- Язык разработки: Kotiln, как предпочтительный для Google Play с 2019 г.
- Выходные форматы: как правило, *.apk, т.к. не все магазины поддерживают загрузку “bundles”, кроме того, вы не сможете запустить *.aab на реальных телефонах или эмуляторах
- Учтите, что релизная версия вашего приложения должна быть подписана
- minSdk: 21 (по умолчанию), минимальная поддерживаемая версия ОС Android
- targetSdk: 32 (по умолчанию) или 33 (максимальный уровень API в н.в.)
- Начиная с августа 2021, приложения должны “целиться” в Android 11 (уровень API 30) или выше
- versionCode приложения, число из возрастающей последовательности 1, 2, 3..
- versionName "1.0" строка, которую видит конечный пользователь в

Параметры “сборки” (Build) в проекте Android Studio

```
android {
    signingConfigs {
        debug {
            storeFile file('K:\\KeyStore\\TestKey.jks')
            storePassword 'TestPassword'
            keyAlias 'keyTest'
            keyPassword 'TestPassword'
        }
    }
    namespace 'club.dmpsy.hellocompose'
    compileSdk 33

    defaultConfig {
        applicationId "club.dmpsy.hellocompose"
        minSdk 21
        targetSdk 33
        versionCode 1
        versionName "1.0"
```

Файлы с параметрами:

build.gradle (для проекта)
build.gradle (для приложения)

Варианты сборки:

1. ***Неподписанный debug***
2. ***Неподписанный release***
3. ***Подписанный debug***
4. ***Подписанный release***

Параметры “сборки” (Build) в проекте Android Studio

```
buildscript {
    ext {
        compose_version = '1.3.2'
    }
}// Top-level build file where you can add configuration options com
plugins {
    id 'com.android.application' version '7.3.1' apply false
    id 'com.android.library' version '7.3.1' apply false
    id 'org.jetbrains.kotlin.android' version '1.6.10' apply false
}
```

```
dependencies {

    implementation 'androidx.core:core-ktx:1.9.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.5.1'
    implementation 'androidx.activity:activity-compose:1.6.1'
    implementation "androidx.compose.ui:ui:$compose_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_version"
    implementation 'androidx.compose.material3:material3:1.1.0-alpha03'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.4'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.0'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_version"
    debugImplementation "androidx.compose.ui:ui-test-manifest:$compose_version"
}
```

конфигурируемые зависимости (раздел dependencies) вашего проекта

Параметры запуска (Launch) в проекте Android Studio

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="HelloCompose"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.HelloCompose"
    tools:targetApi="31">
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:label="HelloCompose"
        android:theme="@style/Theme.HelloCompose">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
```

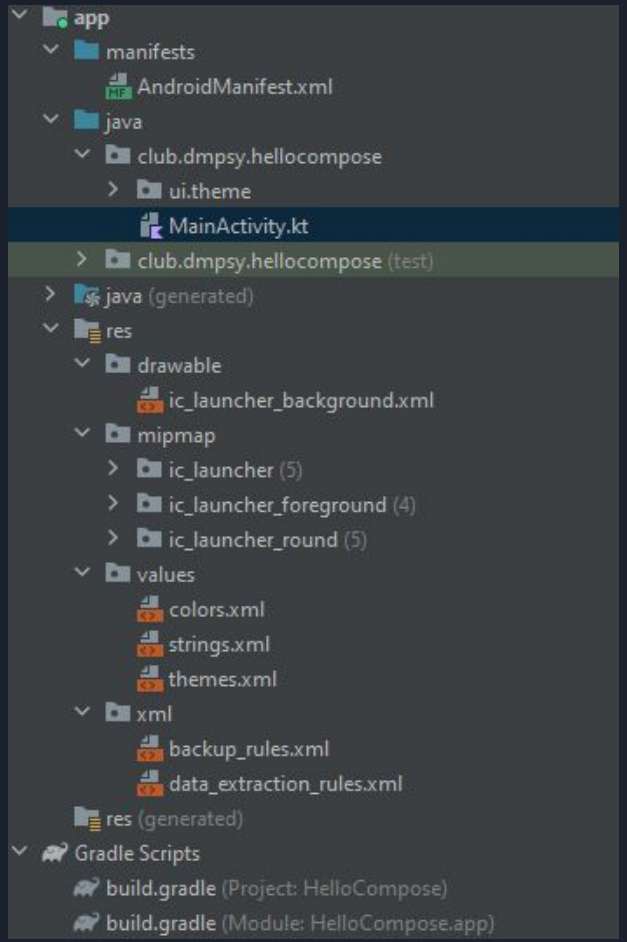
Файл Manifest.xml содержит информацию, необходимую ОС Android для запуска:

иконку приложения,

название приложения,

“Точку входа”, т.е. первый модуль, который должен быть запущен (MainActivity)

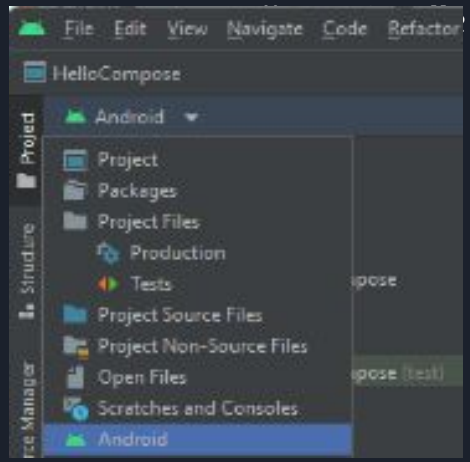
Дерево “App” проекта Android Studio



Дерево “app” содержит:

1. ***Manifest.xml, файл запуска***
2. ***MainActivity.kt , основной исходник приложения на языке Kotlin***
3. ***Папка res содержит иконки, строки и прочие ресурсы вроде цветов и тем системы Material Design***

для сборки проекта



NB. При необходимости, используйте другие “древовидные” представления (см. слева)

Класс MainActivity, “точка входа” приложения

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            HelloComposeTheme {
                // A surface container using the 'background' color from the th
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    Greeting( name: "Android")
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String) {
```

MainAcitivy.kt содержит основной класс приложения MainActivity, который:

1. *Переопределяет метод onCreate создания приложения*
2. *В функции setContent задает тему Material Design (с параметрами рабочей поверхности для отображения элементов интерфейса)*
3. *Вызывает функцию, аннотированную @Compose, для отображения элементов пользовательского интерфейса*

Рассмотрим архитектуру

Трехслойная архитектура приложения

UI Layer
(UI elements and UI state holders)
(Элементы интерфейса и их состояния)

Domain Layer
(Опциональный, модули логики взаимодействия UI Layer и Data Layer)

Data Layer
Repository
(бизнес-логика приложения, доступ к источникам данных)
Data source 1, …, Data source N
(источники данных различной природы)

Jetpack Compose - на уровне UI Layer:

- современный инструментарий (toolkit) для создания пользовательских интерфейсов
- впервые выпущен в 2021 году с целью дальнейшего упрощения и ускорения разработки приложений Android

Jetpack Compose vs. Classic MDC approach

Jetpack Compose заменяет:

1. Xml layouts - описания визуального представления компонентов интерфейса, отделенные от
2. Программного кода Kotlin, определяющего собой поведение этих компонентов

на:

- ***Единое представление компонента интерфейса с помощью “composable”-функции,***

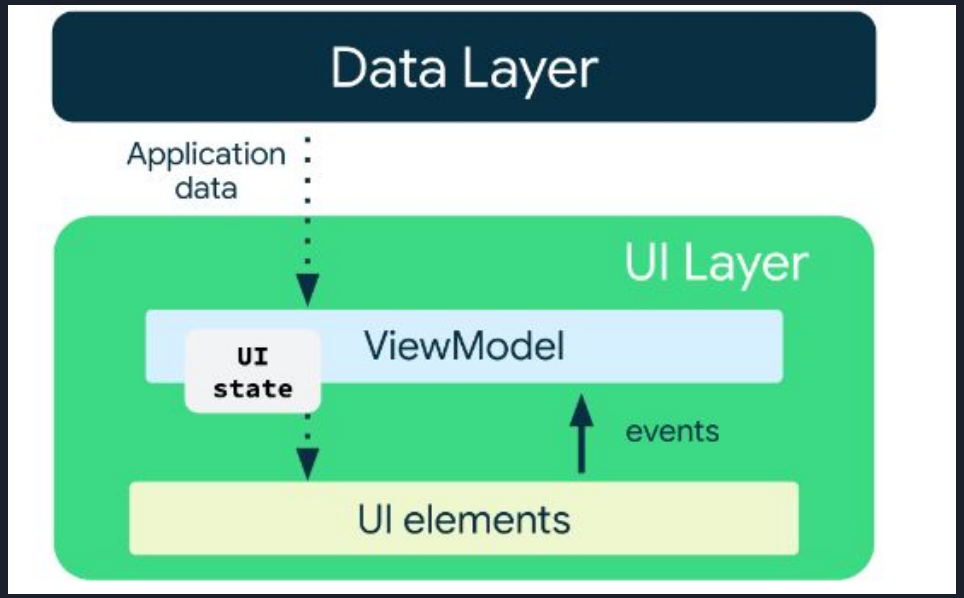
А также реализует:

- ***Подход UDF (однонаправленного потока данных) - “Events up”, “States down”:***

Элементы интерфейса отсылают свои события (Events) на верхний по иерархии уровень

Элементы интерфейса меняют свое визуальное представление в зависимости от состояний (States), полученных от верхнего по иерархии уровня

Однонаправленный поток данных (UDF)

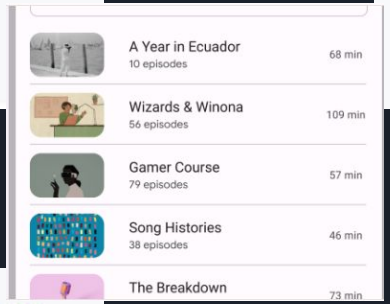


- События (events) поднимаются вверх на уровень ViewModel
- ViewModel меняет состояния (UI states) в соответствии с полученными событиями и/или данными (Application data), пришедшими от слоя данных (Data Layer)
- ViewModel отсылает обновленные состояния вниз к элементам интерфейса для изменения ими своего визуального представления

(см. https://developer.android.com/topic/architecture/ui-layer)

Замена Layout + Code на @Composable

```
<com.google.android.material.divider.MaterialDivider
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:dividerInsetStart="16dp"
    app:dividerInsetEnd="16dp"/>
```




```
@Composable
fun Divider(
    modifier: Modifier = Modifier,
    thickness: Dp = DividerDefaults.Thickness,
    color: Color = DividerDefaults.color
): Unit
```

```
divider.setDividerInsetStart(insetStart)
divider.setDividerInsetEnd(insetEnd)
```

(см. https://m3.material.io/components/divider/overview)

Свойства функции Composable:

1. Должна быть аннотирована как @Composable
2. Должна быть аннотирована @Preview для предпросмотра в Android Studio Previewer
3. Ее название форматируется в PascalCase, в отличие от camelCase обычных котлиновских функций
4. Не возвращает значения
5. [illegible] льзовать их (см.

```
@Composable
fun Divider(
    modifier: Modifier = Modifier,
    thickness: Dp = DividerDefaults.Thickness,
    color: Color = DividerDefaults.color
): Unit
```

Ключевые “строительные блоки” Jetpack Compose:

1. Элементы - контейнеры, невидимые, но составляющие каркас видимой части приложения:

   Row, Column, Box, Scaffold (буквально “строительные леса”) и т.д..

2. Компоненты системы Material 3 Design, составляющие реальное наполнение пользовательского интерфейса, реализующие взаимодействие конечного пользователя и приложения:

   Button, Card, Chip, Dialog, TopAppBar ***etc.***,

   (см. https://m3.material.io/components)

План практикума:

1. Скачать и установить последнюю версию Android Studio с https://developer.android.com/studio
2. Создать в Android Studio новый проект по шаблону Empty Compose Activity (Material 3)
3. Уяснить структуру проекта
4. Скомпилировать и запустить в эмуляторе debug-версию приложения
5. Из папки app/build/outputs/apk/debug загрузить и установить файл *.apk на платформу BlueStacks
6. Скомпилировать неподписанный release и попытаться установить его *.apk на BlueStacks. Система должна отказать в установке
7. Обновить приложение до последней targetSdk = 33, также проапгрейдить dependencies
8. Установить versionCode = 2 и versionName = “1.1”
9. Заменить иконку приложения с помощью Resource manager -> “+” -> “Image Asset”
10. Добавить функциональности приложению (Row/Column/Box/Button/, “кастомный” Button Block)
11. Пересобрать приложение, убедиться в его работоспособности с помощью эмулятора
12. Подписать приложение, сгенерировать подписанный release (app/release/ *.apk), установить этот окончательный вариант приложения на BlueStacks и убедиться в его работоспособности

Следующее видео

- Лекция 1 Практикум

P.S. Презентация доступна для скачивания здесь:
https://dmpsy.club/references/Kotlin/lesson_001_helloWorld_rus.pdf

https://dmpsy.club
postmaster@dmspy.club