Введение в мобильную разработку (Android/Kotlin)

Лекция 1: Подходы к созданию мобильных приложений, структура курса и обзор Kotlin

Александр Глускер

Курс «Мобильная разработка»

3 апреля 2026 г.

Содержание

Цели лекции

- Разобраться в требованиях курса и оценивании
- Понять ландшафт мобильной разработки
- Увидеть особенности Kotlin

Навигация по лекции

Как мы учимся на курсе

- **Лекции**: теория (только главное), обзор, мастер-классы (но на них будет не всё)
- **Семинары**: разбор заданий (если все будет хорошо), сдача
- **Дома**: выполнение задания

Инструменты

IntelliJ Idea, Android Studio

Оценивание и контроль

- **Практические задания:** на каждом семинаре
- **Полусеместр** письменный тест (от руки) + накопленная оценка за практику
- **Финальный экзамен:** или с нуля (теория + практика), или накопленная оценка
- **Проекты etc:** возможны к учету

Политика курса

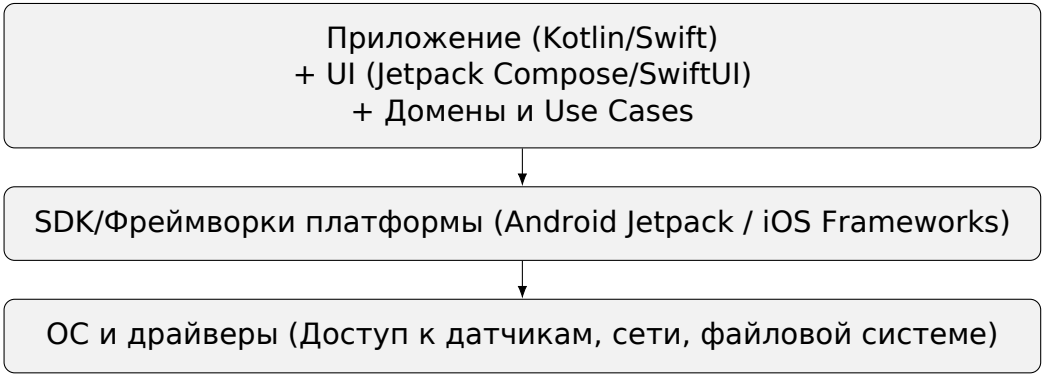
- Дедлайны по практикам, их можно сдавать позже, но оценка ниже и приоритет тем, кто сдает вовремя
- Код должен соответствовать стандартам (обсудим)
- Как можно использовать AI (методика проверки)
- Вопросы в начале пар

Навигация по лекции

Ландшафт: основные подходы

- Нативная разработка (Android: Kotlin/Java; iOS: Swift)
- Кросс-платформенная (Flutter, React Native, Kotlin Multiplatform, .NET MAUI)
- Гибридная (Ionic/Cordova)
- PWA (продвинутые веб-приложения)
- Low-code/No-code

Нативная разработка: архитектура



- **Плюсы:** производительность, полный доступ к API, лучший UX
- **Минусы:** отдельные коды для iOS/Android, выше стоимость

Кросс-платформенные решения

Flutter (Dart)

- Собственный рендеринг UI
- Отличная скорость разработки (hot reload)

Kotlin Multiplatform (KMP)

- Общая бизнес-логика, нативные UI
- Поддержка Авроры :-)

React Native (JS)

- JS/TS стек, мост к нативным компонентам

.NET MAUI

- C#/.NET, единый стек Microsoft

Риски: тонкости производительности, мосты к нативу, обновления экосистем

Гибридные приложения и PWA

- **Гибрид:** веб-страницы в нативном контейнере (WebView)
- **PWA:** офлайн-кеш, иконка на рабочем столе, уведомления (где поддерживается)

Когда это уместно

- Контентные приложения, MVP, быстрые пилоты
- Корпоративные внутренние инструменты

Сравнение подходов (краткая таблица)

	Натив	**Кросс-платф.**	**Гибрид**	**PWA**
Производительность	**Высокая**	Средняя/Высокая	Средняя	Низк./Средн.
Доступ к API	**Полный**	Почти полный	Ограничен	Ограничен
Стоимость	Высокая	**Ниже**	Низкая	**Низкая**
UX/Нативный вид	**Лучший**	Хороший	Посредственный	Зависит от браузера

Навигация по лекции

Kotlin в двух словах

- Современный, статически типизированный язык от JetBrains
- Официально поддерживается Google для Android
- Сильная совместимость с Java (JVM), есть KMP
- Философия: лаконичность, безопасность, практичность
- Поддерживает **функциональную / декларативную**, **объектно-ориентированную**, **алгоритмическую** парадигмы, обобщенное программирование
- Библиотека реализует идеи реактивного программирования

Hello, Kotlin

```kotlin
fun main() {
    println("Hello, Kotlin!")
}
```

Сравнение: Java vs Kotlin

Java

```java
// Java
public class User {
  private final String name;
  public User(String name) { this.name
       = name; }
  public String getName() { return
      name; }
}
```

Kotlin

```kotlin
// Kotlin
data class User(val name: String)
```

Null-safety

```
1 var name: String? = null      // Можно\ хранить\ null
2 println(name?.length)         // Безопасныйвызов  -> null
3 println(name ?: "нет имени") // Elvisоператор-
```

Важно

Оператор !! бросает NPE — используйте его осознанно.

Функции, лямбды, расширения

```kotlin
fun Int.isEven(): Boolean = this % 2 == 0

val nums = listOf(1, 2, 3, 4)
val evens = nums.filter { it.isEven() }  // [2, 4]
```

Сравнение с Java: обработка null и коллекции

Java

```
1 List<Integer> xs = Arrays.asList
      (1,2,3);
2 List<Integer> evens = xs.stream()
3   .filter(x -> x % 2 == 0)
4   .collect(Collectors.toList());
```

Kotlin

```
1 val xs = listOf(1, 2, 3)
2 val evens = xs.filter { it % 2 == 0 }
```

Где применяется Kotlin

- Android-разработка (официальный язык Google)
- Backend: Spring, Ktor, Micronaut
- Kotlin Multiplatform: общая логика для Android/iOS
- Kotlin/Native: десктопные, серверные, встраиваемые приложения
- Kotlin/JS: веб-разработка
- Скрипты и утилиты (JVM)

Навигация по лекции

Переменные: var и val

```
1 var x = 10      // изменяемаяпеременная
2 x = 20
3 val y = 30      // неизменяемаяаналог ( final)
```

Важно

Используйте `val` по умолчанию. `var` — только когда нужно изменять значение.

Базовые типы

- Числовые: `Int`, `Long`, `Double`, `Float`, `Short`, `Byte`
- Логический: `Boolean`
- Символьный: `Char`
- Строковый: `String`

```kotlin
val a: Int = 42
val b: Double = 3.14
val c: String = "Kotlin"
```

if: базовый оператор

```
1 val a = 10
2 if (a > 0) {
3     println("a положительное")
4 }
```

if-else

```kotlin
val a = -5
if (a >= 0) {
    println("Неотрицательное")
} else {
    println("Отрицательное")
}
```

if как выражение

```kotlin
val a = 5
val b = 7
val max = if (a > b) a else b
println("Максимум = $max")
```

Важно

`if` возвращает значение, как в функциональных языках.

if с несколькими ветками

```
1 val grade = 75
2 val result = if (grade >= 90) "Отлично"
3              else if (grade >= 70) "Хорошо"
4              else if (grade >= 50) "Удовлетворительно"
5              else "Плохо"
6
7 println(result)
```

when: базовый вариант

```
1 val x = 2
2 when (x) {
3     1 -> println("Один")
4     2 -> println("Два")
5     else -> println("Другое")
6 }
```

when без аргумента

```kotlin
val y = -3
when {
    y < 0 -> println("Отрицательное")
    y == 0 -> println("Ноль")
    else -> println("Положительное")
}
```

when с диапазонами и множеством значений

```
1 val x = 7
2 when (x) {
3     0, 1 -> println("Ноль илиодин ")
4     in 2..5 -> println("От двухдопяти  ")
5     !in 6..10 -> println("Не отшестидодесяти   ")
6     else -> println("От шестидодесяти  ")
7 }
```

when и типы (smart cast)

```
1 fun process(obj: Any) = when (obj) {
2     is String -> println("Строка длиной ${obj.length}")
3     is Int -> println("Целое число: $obj")
4     else -> println("Неизвестный тип")
5 }
```

when как выражение

```
1 val day = 3
2 val name = when (day) {
3     1 -> "Понедельник"
4     2 -> "Вторник"
5     3 -> "Среда"
6     else -> "Выходной"
7 }
8 println(name)
```

when с произвольными условиями

```
1 val a = 10
2 val b = 20
3 val relation = when {
4     a < b -> "a меньше b"
5     a > b -> "a больше b"
6     else -> "a равно b"
7 }
8 println(relation)
```

Приведение типов

```
1 val n: Int = 42
2 val d: Double = n.toDouble()  // явноепреобразование
3
4 val obj: Any = "строка"
5 if (obj is String) {
6     println(obj.length)  // smart cast
7 }
```

Выражения в Kotlin

```kotlin
val result = try {
    10 / 2
} catch (e: Exception) {
    -1
}
println(result)
```

Важно

В Kotlin почти всё — выражения: `if`, `when`, `try`, лямбды.

Ввод и вывод

```
1 fun main() {
2     println("Введите имя:")
3     val name = readln()
4     println("Привет, $name!")
5 }
```

Как надо писать практики

```
1  fun main() {
2      println("Введите первое\ число:")
3      val a = readlnOrNull()?.toIntOrNull()
4      println("Введите второе\ число:")
5      val b = readlnOrNull()?.toIntOrNull()
6
7      if (a != null && b != null) {
8          val sum: Int? = if ((b > 0 && a > Int.MAX_VALUE - b) ||
9                              (b < 0 && a < Int.MIN_VALUE - b)) {
10             null
11         } else {
12             a + b
13         }
```

Как надо писать практики (продолжение)

```
        if (sum != null) {
            println("Сумма: $sum")
        } else {
            println("Ошибка: переполнение Int")
        }
    } else {
        println("Ошибка: введитекорректныецелыечисла   ")
    }
}
```

Особенности

- Проверка до сложения исключает переполнение
- Абсолютно безопасный ввод: `readlnOrNull() + toIntOrNull()`

Навигация по лекции

Итоги лекции

- Обсудили формат курса, оценивание и ожидания
- Поняли различия подходов к мобильной разработке и критерии выбора
- Познакомились с Kotlin: синтаксис, null-safety, примеры фунционального подхода