# droidconlisbon





# One codebase to rule them all:

Cross-Platform apps with Kotlin and Compose Multiplatform



Andrea Della Porta Mobile Tech Lead @ Cegeka





# The Cross-Platform challenge



#### Multiple platforms = multiple codebases

 Android, iOS, Desktop, Web... each needs its own code

#### Duplicated effort & inconsistent UX

- Features must be reimplemented for each platform
- UI and behavior often differ, leading to a fragmented user experience

#### High maintenance cost

- Bug fixes and updates have to be applied everywhere
- Harder to keep features in sync across platforms

# The Cross-Platform challenge

#### Login

Username

Password

Sign In

Forgot password?

```
@Composable
   var username by remember { mutableStateOf("") }
   var password by remember { mutableStateOf("") }
       modifier = Modifier.fillMaxSize().padding(32),
       horizontalAlignment = Alignment.CenterHorizontally
       Text("Login", fontSize = 28.sp, fontWeight = FontWeight.Bold)
       Spacer(Modifier.height(24.dp))
           value = username, onValueChange = { username = it },
           label = { Text("Username") }
       Spacer(Modifier.height(16.dp))
       OutlinedTextField(
           value = password, onValueChange = { password = it },
           label = { Text("Password") }, visualTransformation = PasswordVisualTransformation()
       Spacer(Modifier.height(32.dp))
       Button(onClick = { /* Handle login */ }, shape = RoundedCornerShape(22.dp)) {
           Text("Sign In")
       Spacer(Modifier.height(16.dp))
       Text("Forgot password?", color = Color.Gray)
```

```
struct LoginView: View {
   @State private var username = ""
   var body: some View {
       VStack(spacing: 24) {
           Text("Login").font(.largeTitle).bold()
           TextField("Username", text: $username)
              .textFieldStyle(RoundedBorderTextFieldStyle())
           SecureField("Password", text: $password)
               .textFieldStyle(RoundedBorderTextFieldStyle())
           Button(action: { /* Handle login */ }) {
               Text("Sign In")
                   .frame(maxWidth: .infinity)
                   .padding()
                   .background(Color.blue)
                  .foregroundColor(.white)
                   .cornerRadius(22)
           Text("Forgot password?")
               .foregroundColor(.gray)
               .font(.footnote)
```



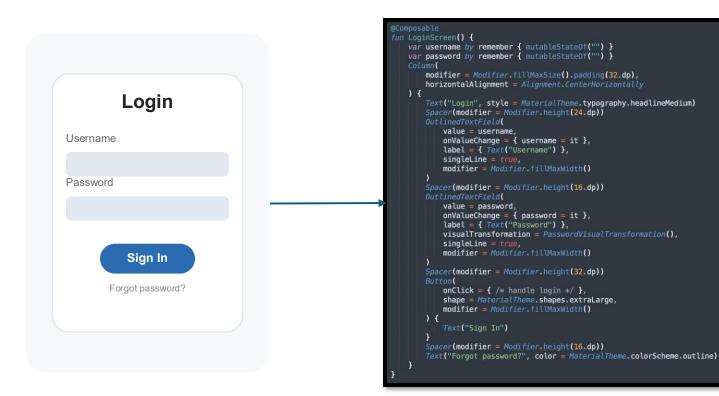


# The Dream

What if you could write your app **once** and run it **everywhere**?

Imagine: a single codebase powering Android, iOS, Desktop, Web...

#### Consistent experience, faster development, easier maintenance







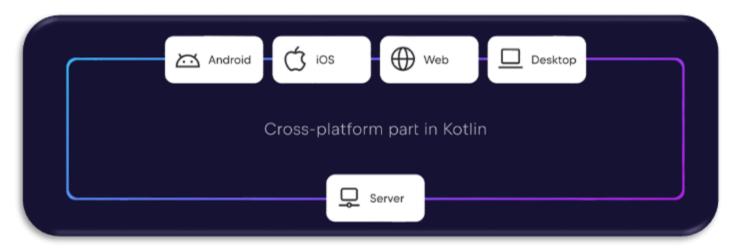


# **Introducing Kotlin**

**Kotlin Multiplatform** is a technology from JetBrains and Google → Share code across Android, iOS, Desktop, Web, Server.

#### How does it work?

- Write core logic (business rules, networking, models) in Kotlin
- KMP compiles your code for each platform (JVM, JS, Native)
- Platform-specific code only where needed (UI, device APIs)



#### **Benefits**

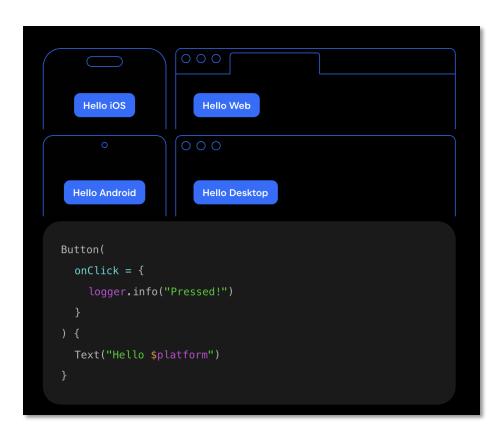
- No need to rewrite business logic for each platform
- One set of tests, one source of truth
- Great interoperability with platform code (Swift, JavaScript, Java, etc.)



# **Compose Multiplatform**

Compose Multiplatform brings Jetpack Compose's declarative UI to

→ Android, iOS, Desktop and Web (alpha)



#### How does it work?

- Compose UI code is written in Kotlin
- The toolkit adapts rendering to each platform (native widgets, HTML/CSS, etc.)
- Supports platform-specific features when needed

#### **Benefits**

- Rapid UI development with hot reload and live preview
- Consistent look & feel across devices
- Easy to share components and design system





# How it works: Architecture

#### **Shared Code**

- Business logic, data models, networking written in Kotlin
- UI screens in Compose Multiplatform

# JVM iosArm64 JS JVM classfiles JVM iosArm64 JS Native executables for 64-bit iPhone device

# Platform-specific code

- Small parts for device APIs (camera, sensors, notifications)
- UI integration with host app if needed (navigation, lifecycle)

#### **Project structure**

- commonMain/: Kotlin Multiplatform code (logic, Compose UI)
- androidMain/, iosMain/, desktopMain/, wasmJsMain/: platform modules



# What can be shared?



### **Business logic**

- ViewModels, data models, repositories
- Networking, state management, validation

#### UI

- Screens, components, layouts (Compose Multiplatform)
- Navigation logic, design system

#### **Tests**

- Unit tests for shared logic and UI
- Integration tests



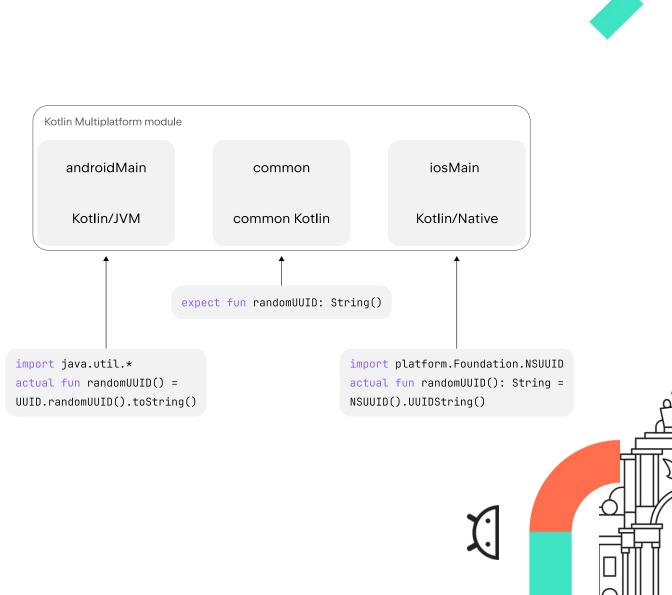
# When to go native?

## **Platform-specific APIs**

Camera, sensors, notifications, device permissions

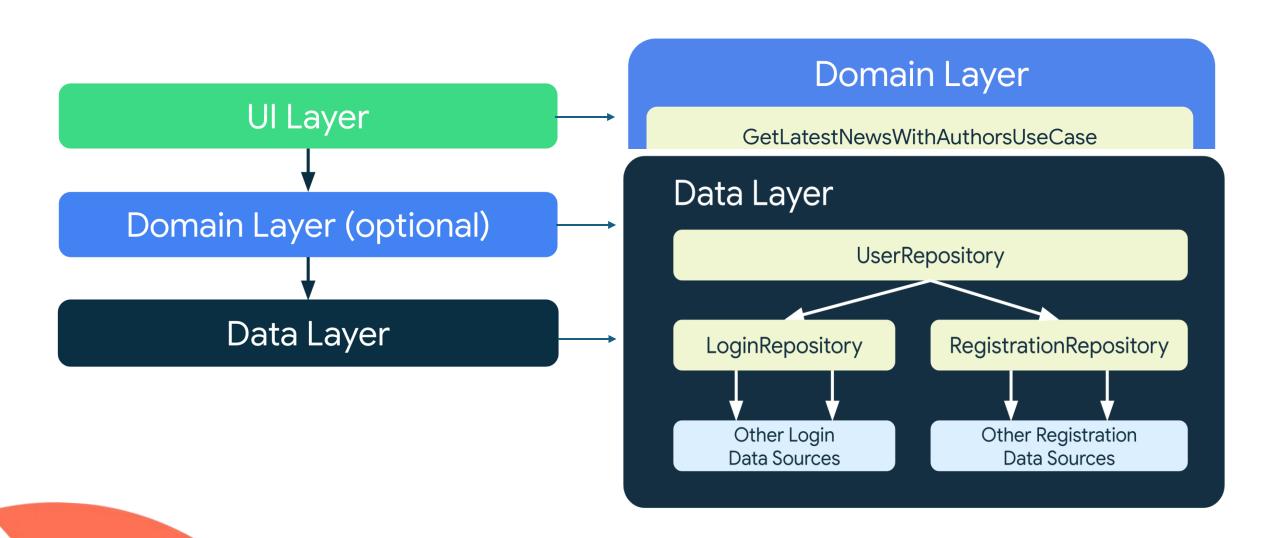
# **Custom platform UI**

Native controls, advanced animations, platform look & feel



# **Project structure**

Clean code, clear structure: let's build it with clean architecture!



# Let's build a WeatherApp!

https://github.com/andreadellaporta01/weatherapp



iOS

















**Any Question?** 







andreadellaporta01



andrea-della-porta-01



**AndreaDellaPor5**