使用 KMP 开发支付服务：回顾一卡通面临的技术挑战与成果

林宇軒

Developed by JetBrains | @b95505017

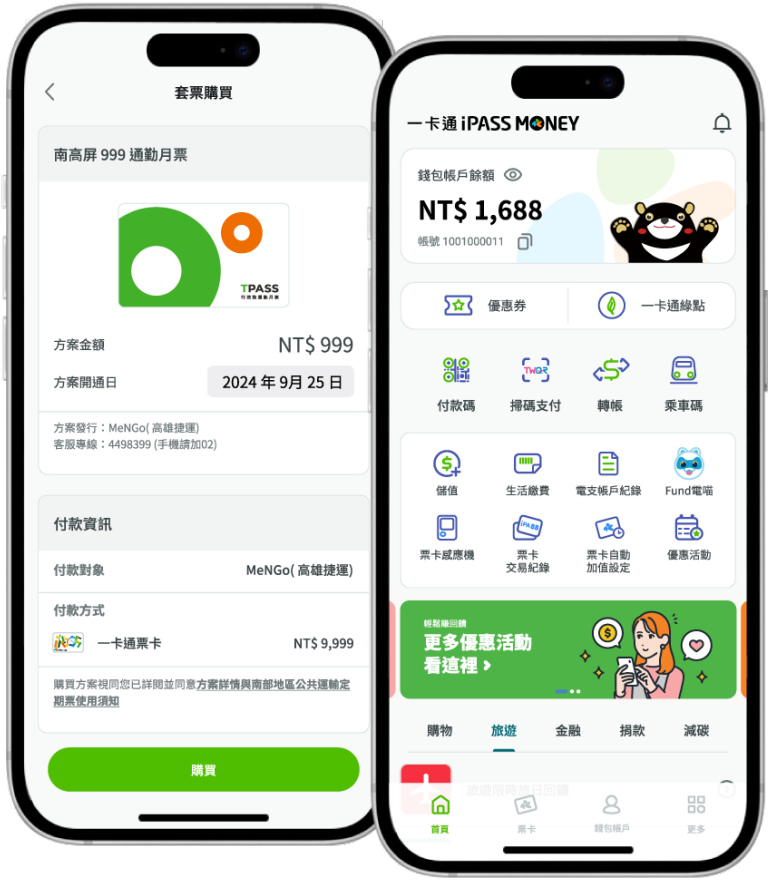
About me

- Android development since 2010
- Using Kotlin for the past 6 years
- 3 years of experience with KMP

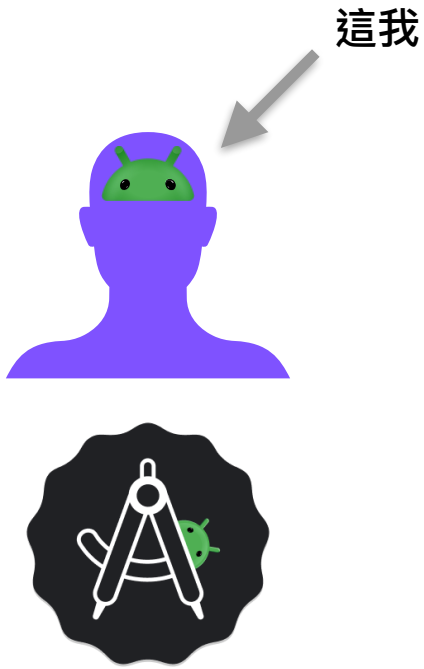
林宇軒　@b95505017

iPASS MONEY app

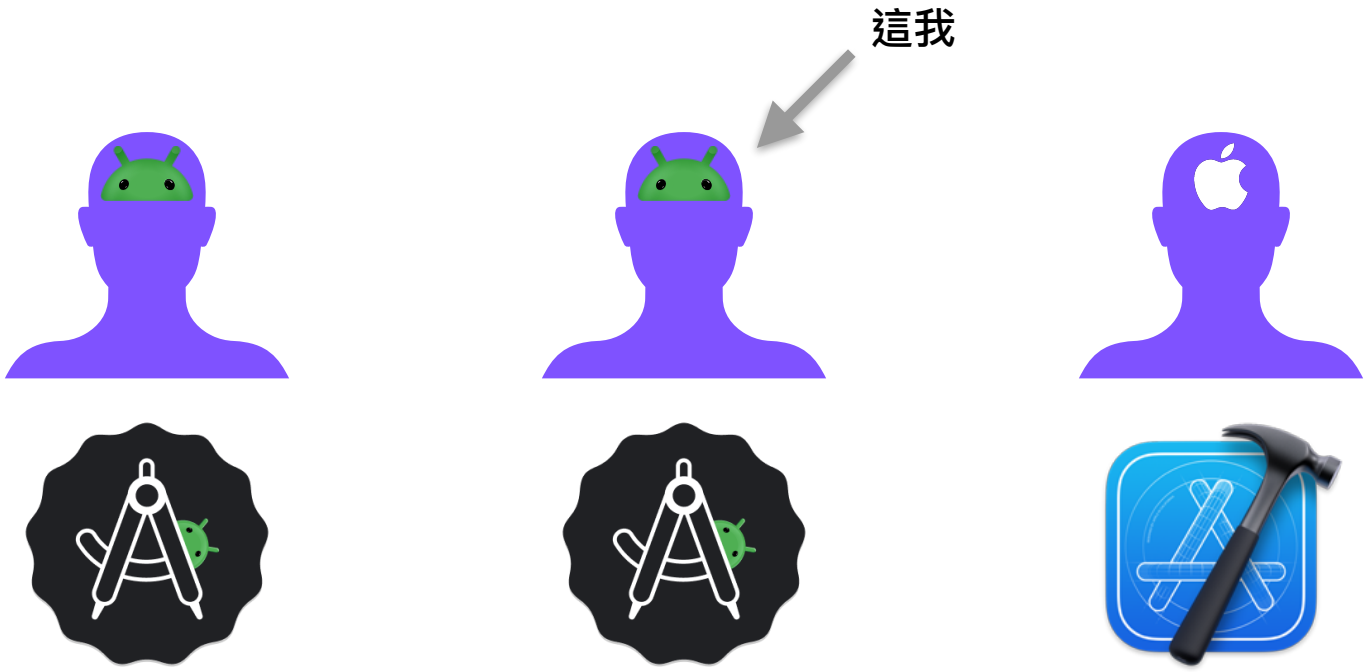
- 電子支付
- 電子票證



Team members



Team members



Team members

Get started from multiple options

Share a UI with Compose Multiplatform or keep it native?
Start from scratch or migrate from an existing Android app?
Do it your way!

Share logic and keep UI native

Create your first Kotlin Multiplatform app with a native UI for Android (Jetpack Compose UI) and iOS (SwiftUI)

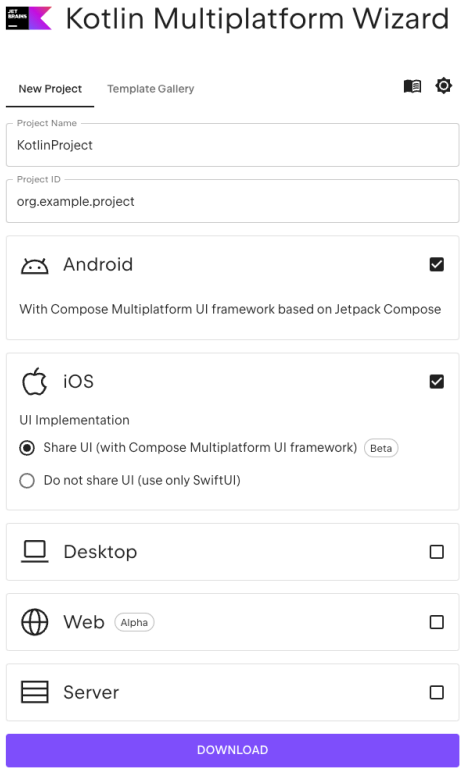
Share both logic and UI

Create your first Kotlin Multiplatform app with a shared UI across Android, iOS, and desktop using Compose Multiplatform

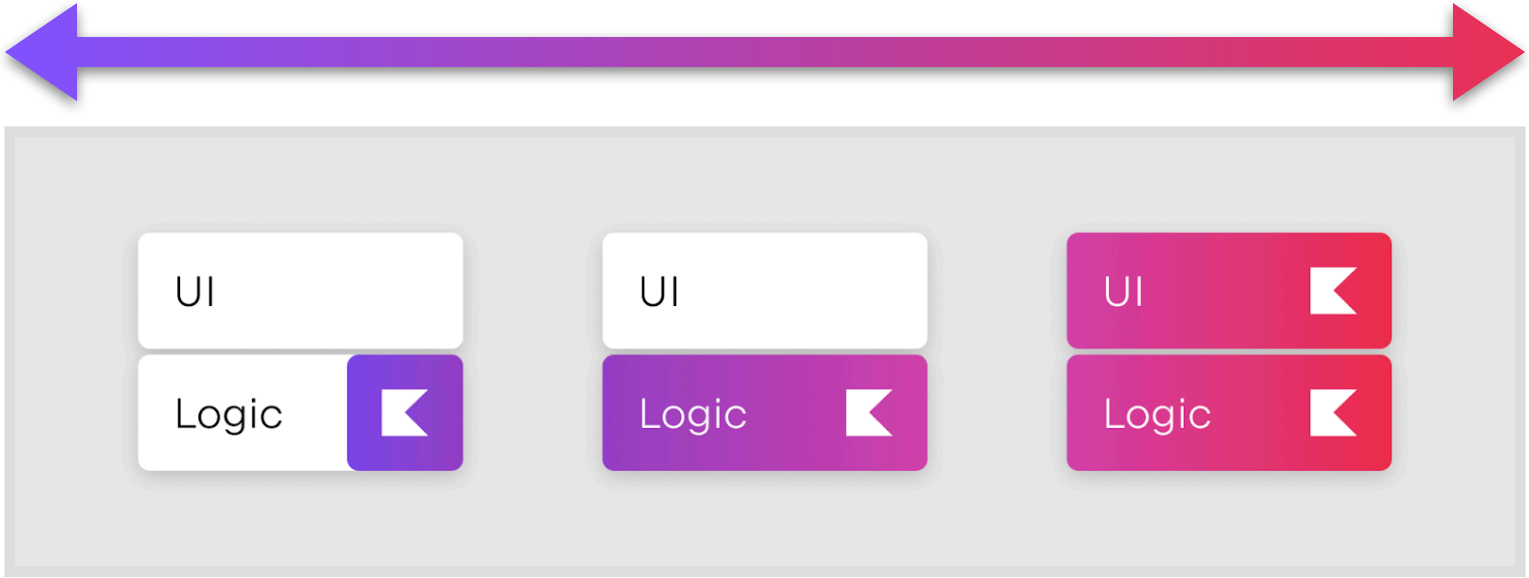
Migration

Migrate an existing Android app to iOS

Learn how to make your existing Android app work on iOS with SwiftUI



Code-sharing freedom with KMP



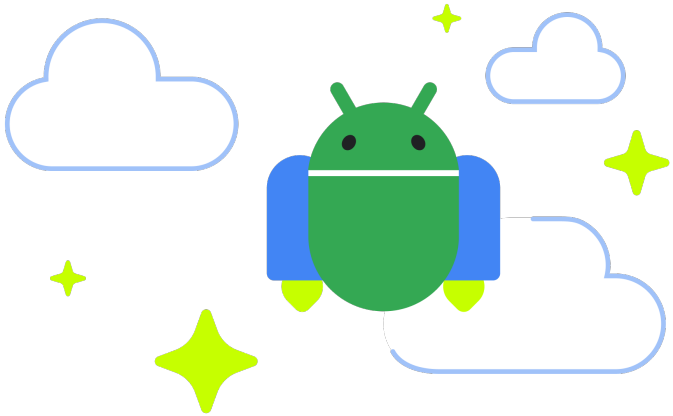
Multiplatform Jetpack libraries

Multiplatform Jetpack libraries

Kotlin Multiplatform is officially supported by Google for sharing business logic between Android and iOS. Many of our Jetpack libraries have already been migrated to take advantage of KMP.

The following Jetpack libraries provide KMP support:

Maven Group ID	Latest Update	Stable Release	Release Candidate	Beta Release	Alpha Release	Documentation
annotation (*)	October 30, 2024	1.9.1	-	-	-	
collection	November 13, 2024	1.4.5	-	-	1.5.0-alpha06	
datastore	May 1, 2024	1.1.1	-	-	-	Documentation
lifecycle (*)	November 13, 2024	2.8.7	-	-	2.9.0-alpha07	
paging (*)	November 13, 2024	3.3.4	-	-	-	
room	October 30, 2024	2.6.1	-	-	2.7.0-alpha11	Documentation
sqlite	October 30, 2024	2.4.0	-	-	2.5.0-alpha11	Documentation



NFC 過卡購買套票


套票購買
感應過卡時，請將卡片緊貼手機 NFC 感應位置
操作過程中，請勿移動卡片
感應中，請將卡片緊貼
手機 NFC 感應位置
接下來的操作流程中，請勿移動卡片
取消


套票購買
感應過卡時，請將卡片緊貼手機 NFC 感應位置
操作過程中，請勿移動卡片
感應中，請將卡片緊貼手機 NFC 感應位置
接下來的操作流程中，請勿移動卡片
取消


套票購買
全部
南高屏
高雄
台南
屏東
南高屏 999 通勤月票
了解方案詳情
NT$ 999
購買方案
南高屏 999 通勤季票
了解方案詳情
NT$ 2,997
購買方案
高雄市區 399 通勤月票
了解方案詳情
NT$ 399
購買方案
大台南公車 299 無限搭
了解方案詳情
NT$ 299
購買方案
高雄市區 399 通勤季票
了解方案詳情


套票購買
南高屏 999 通勤月票
TPASS
方案金額
NT$ 999
方案開通日
2024/07/09
方案發行：MeNGo(高雄捷運)
客服專線：4498399 (手機請加02)
付款資訊
付款對象
MeNGo(高雄捷運)
付款方式
一卡通票卡
NT$ 9,999
購買方案同意已詳閱並同意方案詳情與南部地區公共運輸定期票使用須知
購買

NFC 過卡購買套票

iOS:

- `NFCTagReaderSession`
  - `alertMessage/errorMessage` (i18n String)
- UI controlled by system

Android:

- `NFCAdapter`
- Fully custom UI

該從哪開始共用？

NFC 過卡購買套票 (*Flutter-nfc_manager)*

```dart
/// Start the session and register callbacks for tag discovery.
///
/// This uses the NFCTagReaderSession (on iOS) or NfcAdapter#enableReaderMode (on Android).
/// Requires iOS 13.0 or Android API 19, or later.
///
/// `onDiscovered` is called whenever the tag is discovered.
///
/// `pollingOptions` is used to specify the type of tags to be discovered. All types by default.
///
/// (iOS only) `alertMessage` is used to display the message on the popup shown when the session is started.
///
/// (iOS only) `invalidateAfterFirstRead` is used to specify whether the session should be invalidated
/// after the first tag is discovered. Default is true.
///
/// (iOS only) `onError` is called when the session is stopped for some reason after the session has started.
Future<void> startSession({
  required NfcTagCallback onDiscovered,
  Set<NfcPollingOption>? pollingOptions,
  String? alertMessage,
  bool? invalidateAfterFirstRead = true,
```

NFC 過卡購買套票 (*Flutter-nfc_manager)*

Handling Platform Tag

The following platform-tag-classes are available:

- Ndef
- FeliCa (iOS only)
- Iso7816 (iOS only)
- Iso15693 (iOS only)
- MiFare (iOS only)
- NfcA (Android only)
- NfcB (Android only)
- NfcF (Android only)
- NfcV (Android only)
- IsoDep (Android only)
- MifareClassic (Android only)
- MifareUtralight (Android only)
- NdefFormatable (Android only)

```
if (Platform.isAndroid) {
  // Handle NFC-F
} else if (Platform.isIOS) {
  // Handle FeliCa
}
```

NFC 過卡購買套票 (*KMP*)


```kotlin
expect class CardTag {
    suspend fun sendCommand(command: ByteString): ByteString
}

// Android
actual class CardTag(private val tag: NfcF) {
    actual suspend fun sendCommand(command: ByteString) = withContext(IO) {
        tag.transceive(command.toByteArray()).toByteString()
    }
}

// iOS
actual class CardTag(private val tag: NFCFeliCaTagProtocol) {
    actual suspend fun sendCommand(command: ByteString) = suspendCoroutine {
        tag.sendFeliCaCommandPacket(command.toByteArray().toNSData()) { _, _ ->
            // …
        }
    }
}
```

NFC 過卡購買套票

```
var tag by remember { mutableStateOf<NfcF?>( value: null) }
val nfcAdapter = remember(activity) {
    activity?.let(NfcAdapter::getDefaultAdapter)
}
val enableReaderMode = remember {
    {
        runCatching {
            nfcAdapter?.enableReaderMode(
                activity,
                { it: Tag!
                    val nfcFTag = NfcF.get(it)
                    // ...
                },
                NFC_FLAGS,
                extras: null,
            )
        }
    }
}
val closeResource = remember {
    {
        runCatching { tag?.close() }
        tag = null
        runCatching { nfcAdapter?.disableReaderMode(activity) } ^lamb
    }
}
```

```
// MARK: - NFCTagReaderSessionDelegate

extension NFCReader: NFCTagReaderSessionDelegate {

    func tagReaderSessionDidBecomeActive(_ session: NFCTagReaderSession) {
        updateStatus(.active)
    }

    func tagReaderSession(_ session: NFCTagReaderSession, didDetect tags: [NFCTag]) {
        guard let tag = tags.first else {
            return
        }

        guard case let .feliCa(felicaTag) = tag else {
            updateStatus(.unsupportedCardDetected)
            return
        }

        session.connect(to: tag) { [weak self] error in
            guard let self else { return }

            if let error {
                self.session?.invalidate(errorMessage: error.localizedDescription)
            } else {
                self.setAlertMessage(String(localized: "nfc__session_alert__processing"))
                self.updateStatus(.connected(felicaTag))
            }
        }
    }

    func tagReaderSession(_ session: NFCTagReaderSession, didInvalidateWithError error: Error) {
        guard let nfcError = error as? NFCReaderError else {
            return
        }
        updateStatus(.invalidated(nfcError))
    }
}
```

共用層核心架構的好幫手

- Decompose (https://arkivanov.github.io/Decompose)
  - Lifecycle-aware business logic components (aka BLoC)
  - Navigation routing functionality
- Koin (https://github.com/InsertKoinIO/koin)
  - lightweight dependency injection framework

團隊開發方式

- 單兵作戰模式
  - 獨自處理完商業邏輯共用層 + 雙平台 UI
- 分工合作模式
  - 輪流負責商業邏輯共用層
  - UI 交給各自熟悉的人負責

Thanks!
Have a
Nice KMP

Kotlin @b95505017