

Kotlin

Multiplatform



A hands-on perspective

Melisa De la Garza, September 2024

First of all,
how dare I?



Story time!

What we know

Kotlin Multiplatform Mobile: The promises

- Allows us to share code between both platforms
- Reduce the amount of code to write and maintain
- Faster development cycles
- Consistent behavior across platforms

IDEA 😊

R&D 🧐

SETUP 🧑

FEATURE 😊

SHIP 🧑

GROW 🧑

Decisions. Decisions. Decisions.

- Monorepo? → **Submodules**
- Cocoapods? SPM? → **Cocoapods**
- What will be shared? → **Network layer and business logic**

IDEA 😊

R&D 🤔

SETUP 🧐

FEATURE 😊

SHIP 🧐

GROW 😊

Setting it up

Environment variables: JAVA_Home

```
/bin/sh -c /Users/melisadlg/Library/Developer/Xcode/DerivedData/Build/Intermediates.no
```

```
REPO_ROOT="$PODS_TARGET_SRCROOT"  
"$REPO_ROOT/../gradlew" -p "$REPO_ROOT" $KOTLIN_PROJECT_PATH:syncFramework -Pkotlin.native.cocoa  
-Pkotlin.native.cocoapods.archs="$ARCHS" -Pkotlin.native.cocoapods.configuration="$CONFIGURATION"  
The operation couldn't be completed. Unable to locate a Java Runtime.  
Please visit http://www.java.com for information on installing Java.
```

Command PhaseScriptExecution failed with a nonzero exit code

✖ Command PhaseScriptExecution failed with a nonzero exit code

✖ Build failed 01/09/2024, 10:57 56.2 seconds
1 error, 5 warnings

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🤖

GROW 😊

Setting it up

The circus act 🎪

coroutines:

Light-weight threads that allow you to write asynchronous non-blocking code.

Without coroutines:

- Blocking
- Poor UX
- Main Thread overload = crash!

With coroutines:

- Smooth performance
- Better UX
- Thread management

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🤖

GROW 🧘

Pain points

Not-so-smooth sailing: Serialization

```
409     func createOrder(_ order: [String : Any?], _ status: String?, _ cancelInfo: [String : Any?]? ) -> OrderData? {
410         let paymentData = order["payment"] as? [String: Any]
411         let paymentMethods = paymentData?["methods"] as? [[String: Any]]
412         var methods = [OrderMethods]()
413         paymentMethods?.forEach { element in
414             let method = OrderMethods.init(method: element["method"] as? String,
415                                             amount: (element["amount"] as? Float) as? KotlinFloat,
416                                             reference: element["reference"] as? String)
417             methods.append(method)
418         }
419         let orderPayment = OrderPayment.init(subTotal: (paymentData?["subTotal"] as? Float) as? KotlinFloat,
420                                             shipping: (paymentData?["shipping"] as? Float) as? KotlinFloat,
421                                             total: (paymentData?["total"] as? Float) as? KotlinFloat,
422                                             status: paymentData?["status"] as? String,
423                                             errorReason: paymentData?["errorReason"] as? String,
424                                             methods: methods)
425
426
427         let orderDocuments = order["documents"] as? [[String: Any]]
```

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🤖

GROW 😊

Pain points

Not-so-smooth sailing: Architecture

```
> Task :shared:compileKotlinIosX64 FAILED
```

```
FAILURE: Build failed with an exception.
```

```
* What went wrong:
```

```
Execution failed for task ':shared:compileKotlinIosX64'.
```

```
> Compilation finished with errors
```

```
* Try:
```

```
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.
```

```
* Get more help at https://help.gradle.org
```

```
BUILD FAILED in 7s
```

```
1 actionable task: 1 executed
```

```
! Command PhaseScriptExecution failed with a nonzero exit code
```

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🤖

GROW 🧘

Pain points

Not-so-smooth sailing: Reserved keywords

```
productDescription = product.description {  
    productDescription = productDescription  
}
```

description

A textual representation of the receiver.

```
var description: String { get }
```

Required

Return Value

A string that describes the object.

[Open in Developer Documentation](#)

```
productDescription = product.description_ {  
    productDescription = productDescription  
}
```

description_

```
var description_: String? { get set }
```

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🤖

GROW 😊

Pain points

Not-so-smooth sailing: ObjC

```
__attribute__((objc_subclassing_restricted))
__attribute__((swift_name("PaymentRepositoryKmm"))))
@interface KmmFeaturesPaymentRepositoryKmm : KmmFeaturesBaseRepository <KmmFeaturesKoin_coreKoinComponent>
- (instancetype)init __attribute__((swift_name("init()"))) __attribute__((objc_designated_initializer));
+ (instancetype)new __attribute__((availability(swift, unavailable, message="use object initializers instead")));

/**
 * @note This method converts instances of CancellationException to errors.
 * Other uncaught Kotlin exceptions are fatal.
 */
- (void)deletePaymentPaymentDeleteRequest:(KmmFeaturesPaymentDeleteRequest *)paymentDeleteRequest completionHandler:(void (^)(KmmFeaturesResponse<KmmFeaturesCheckoutPaymentResponse *> * _Nullable, NSError * _Nullable))completionHandler
__attribute__((swift_name("deletePayment(paymentDeleteRequest:completionHandler:)")));
```

```
class PaymentRepositoryKmm : BaseRepository(), KoinComponent {

    private val paymentApi: PaymentApi by inject()
    private val httpClient: HttpClient by inject()

    suspend fun deletePayment(paymentDeleteRequest: PaymentDeleteRequest): Response<CheckoutPaymentResponse> { ... }
```

Pain points

Recap

- No decoding → **RIP testing**
- Clashing namings → **Refactoring**
- CI/CD → **Manual releases**
- Libraries and plugins → **Balancing act**

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🤖

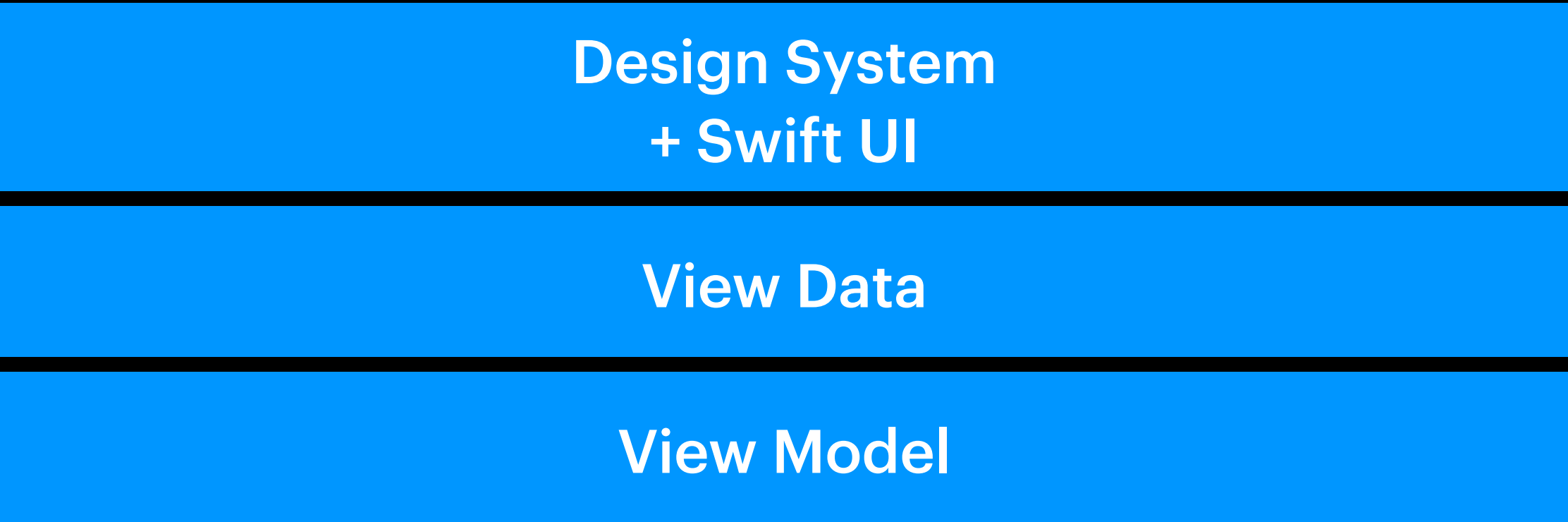
GROW 🧘

What would it look like?

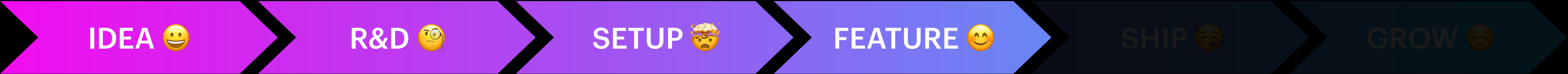
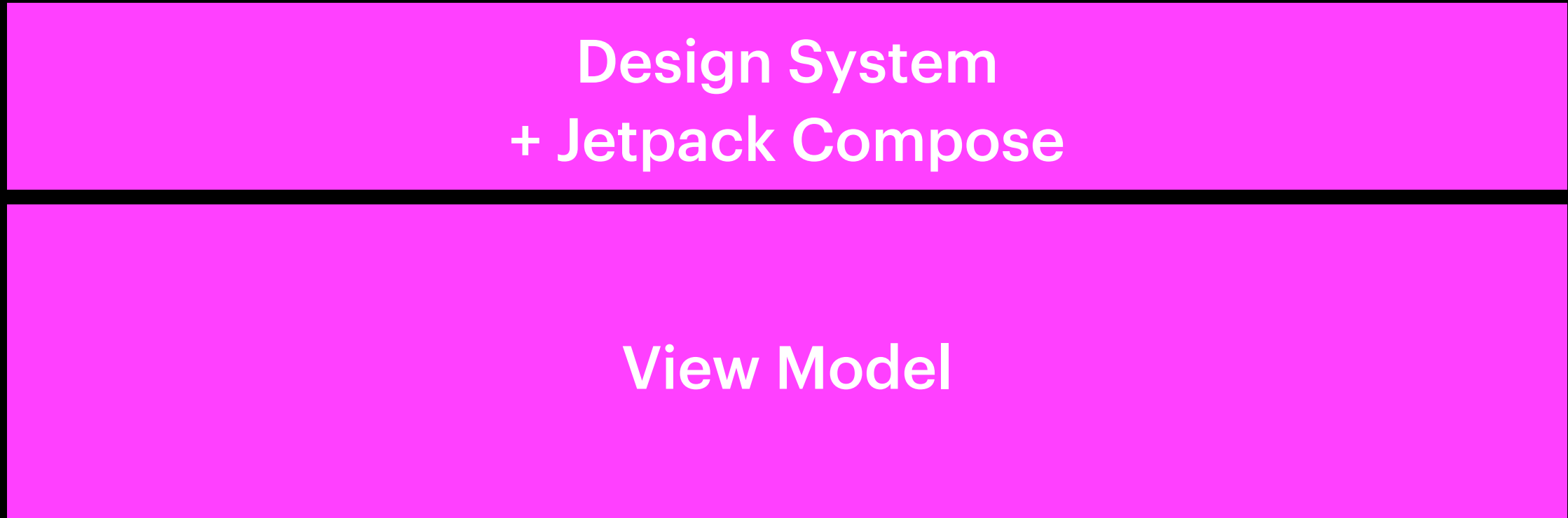
	THE PAST		THE FUTURE
	Android-specific	iOS-specific	D-KMP (multiplatform)
UI System	Views/Fragments	ViewControllers/UIKit	Compose & SwiftUI
Observable	LiveData	Combine	StateFlow
HTTP Client	Retrofit	Alamofire	Ktor Client
Serialization	Gson/Moshi	Codable/JSONSerializ	Kotlin Serialization
Structured Data	Room	CoreData	SQLDelight
Settings	SharedPreferences	NSUserDefaults	MultiplatformSettings
Concurrency	AsyncTask	GrandCentralDispatch	Coroutines

Architecture

iOS



Android



Happy thoughts

Good Technical Findings

- Reduced and unified business logic
- Scalable
- Single source of truth
- Stable performance
- Keep same architecture, enums and generics
- Feature flag 🙌

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🤖

GROW 🧘

Happy thoughts

Good Findings

- Team communication
- Knowledge sharing
- Flexible capacity
- Flexibility to changes
- Only fix it once!

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🤖

GROW 🤖

Ship it! 🚀

IDEA 😊

R&D 🤔

SETUP 🤯

FEATURE 😊

SHIP 🎉

GROW 🧘

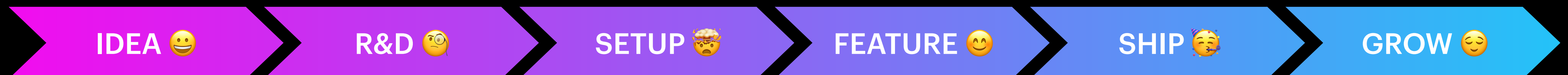
The girl with the list

Pros

- Code sharing
- Efficiency
- Interoperability
- Faster delivery

Cons

- Learning curve
- Tooling maturity
- Dependencies and configuration
- Setup on existing large projects



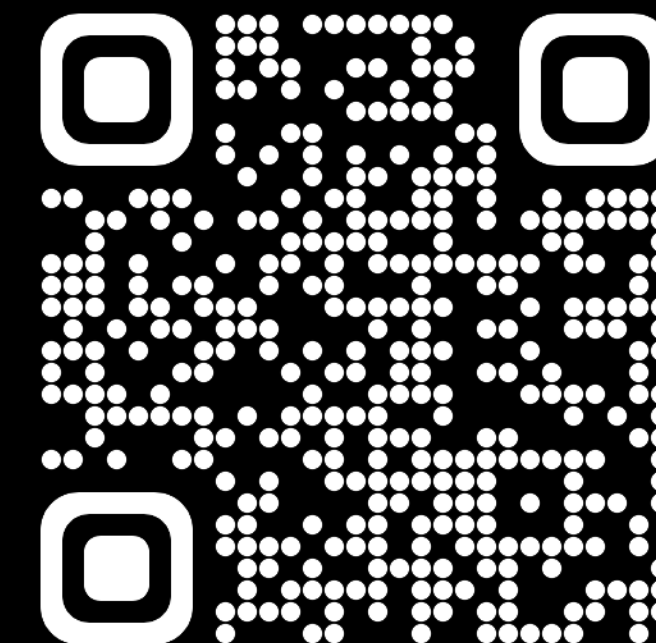
Is it worth it?

**“Technology is nothing.
What's important is that
you have a faith in people,
that they're basically good
and smart, and if you give
them tools, they'll do
wonderful things with them.”**

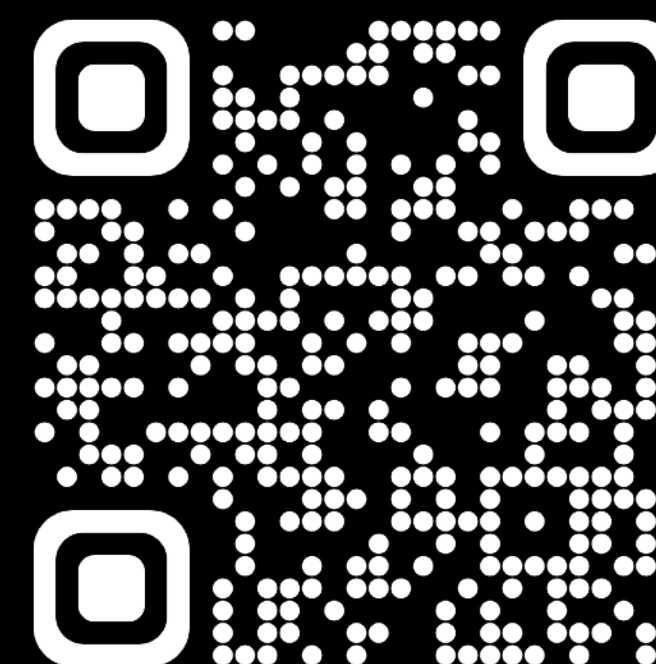
— Steve Jobs

Thank you!

Melisa De la Garza, September 2024



LinkedIn



Medium