KOTLIN MULTIPLATFORM IN ACTION

LAURA DE LA ROSA

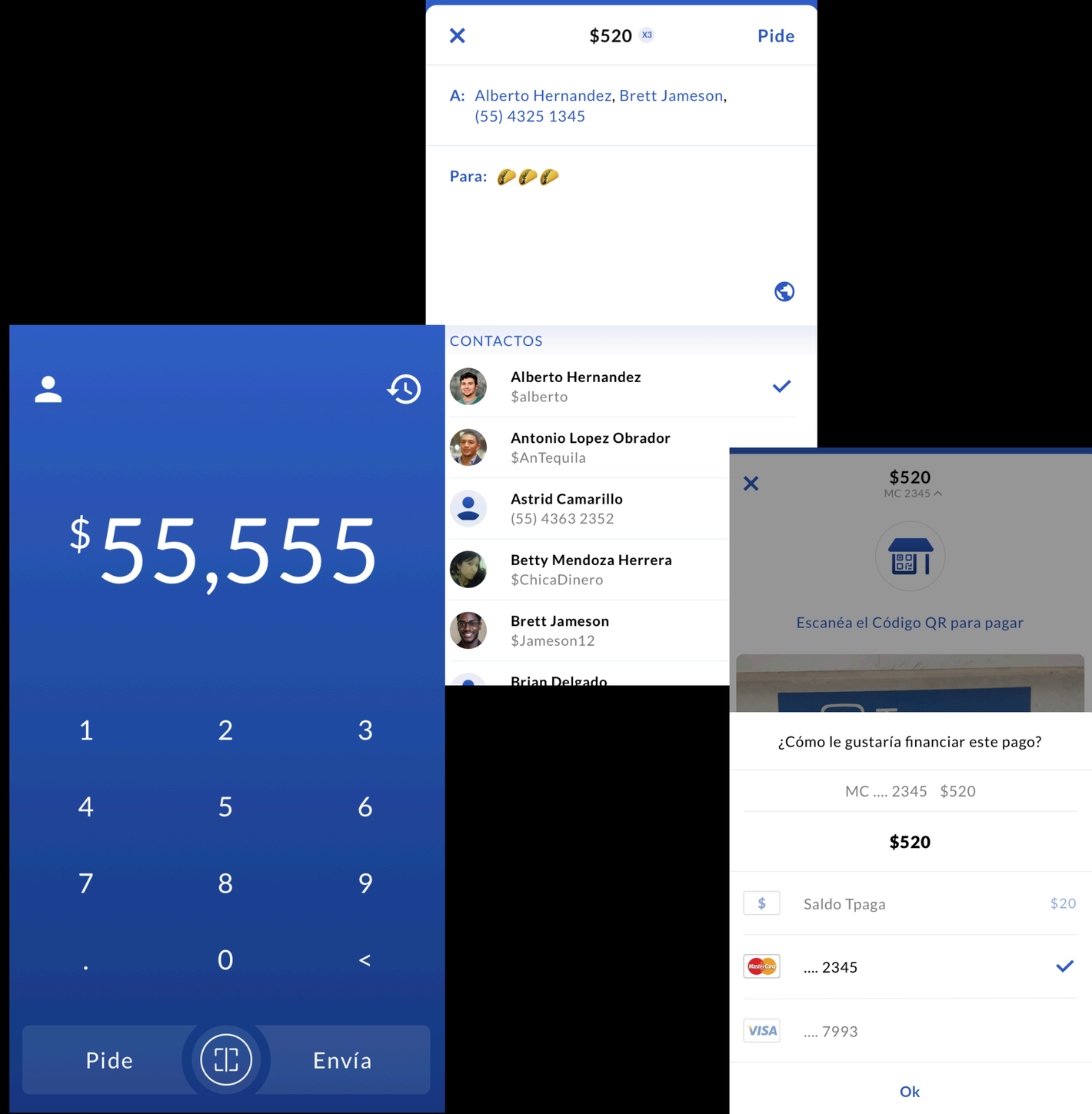
@lauramdelarosa

@MedellinAndroid

#KotlinConfGlobal2020

Tpaga MX

- Envío de dinero
- Pide dinero
- Pagos con Qr
- Formulario de regulación mexicana.
- Ver Informes de Pagos



Tpaga MX

- Usuarios pagan fácil y sin efectivo
- Changarros controlan sus ventas
- Bancarizan los negocios informales

Kotlin Multiplatform

- En producción hace 1 año en Android
- En producción hace 10 meses en iOS
- 18% iOS
- Equipo .

Orígenes

Orígenes

- App de Colombia
- Oportunidad en MX
- Poco tiempo
- Objetivos específicos MX
- Ambas Plataformas

Orígenes

Nueva App con KOTLIN MULTIPLATAFORMA

Proceso y Aprendizaje

Acoplamiento

- Android. ☺️
- iOS. 🙄
- Problemas?
  - Revisando PR
  - Cambiando lógica de negocio

Migración

- Todo en Kotlin
- Migrar la arquitectura actual - Probabilidad de mejorarla .
- Tener buena comunicación ante:
  - Nuevos features .
  - Revisando PR
  - iOS empoderados para aprender y crecer
- Tpaga NO migró

Tip

Cuando desarrollas en código compartido,
Piensa en compartir.

¿Qué compartir?

- Depende 🤫
  - Reglas y lógica de negocio
    - Network
    - Manejo de errores
    - Casos de Uso
    - Transformación de datos.

Tip

Estudia muy bien las librerías open source,
antes de implementarlas.

- No son estables
- Representan riesgos

¿De qué dependo?

Expect / Actual

```
package
mx.tpaga.wallet


expect class
DeviceInfo() {
    fun getOsName():
String
```

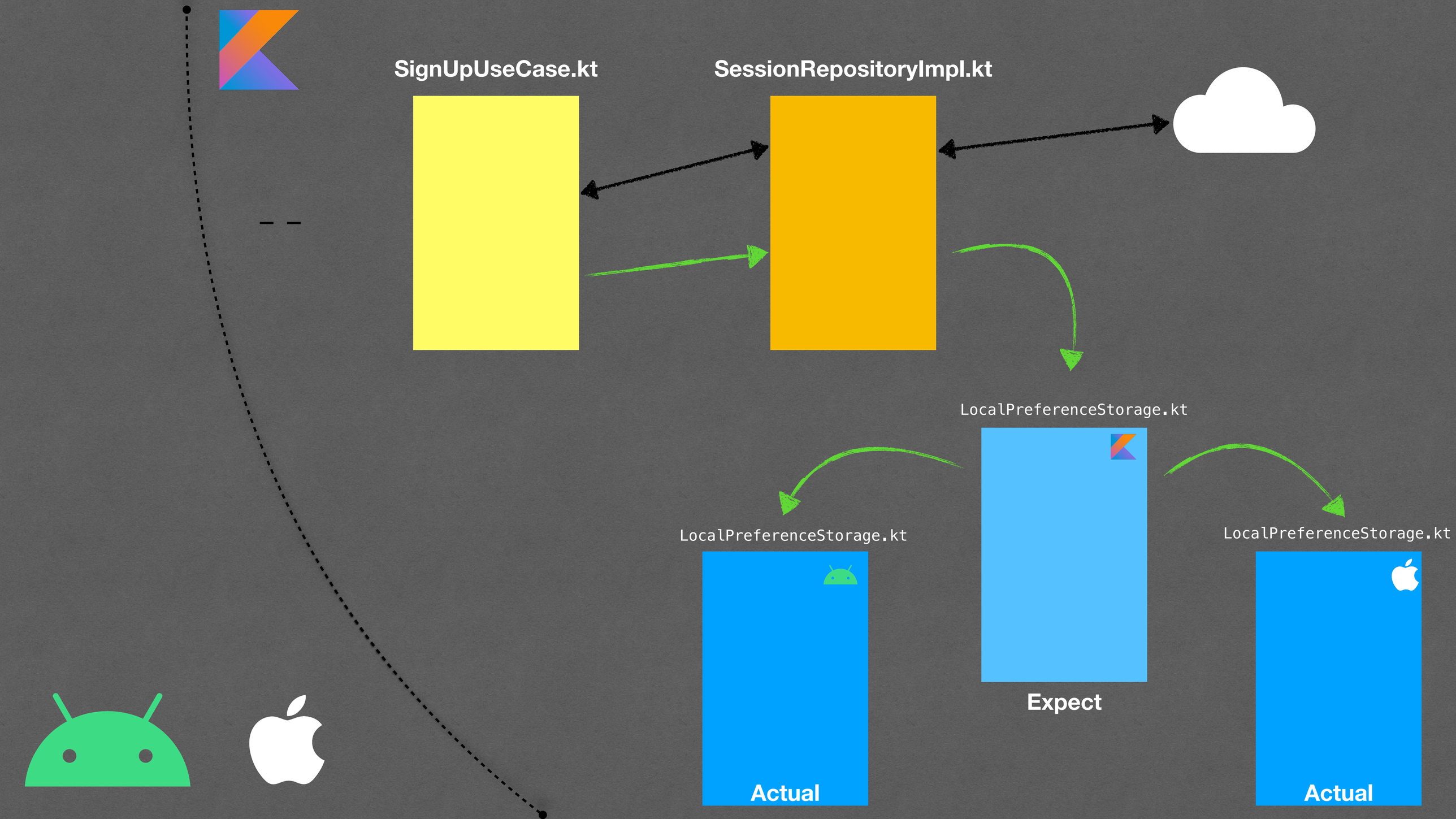
commonMain/kotlin/mx/tpaga/wallet/common/utils/DeviceInfo.kt

```kotlin
package mx.tpaga.wallet

actual class DeviceInfo {
    actual fun getOsName(): String =
"Android"
}
```

common/kotlin/java/mx/tpaga/wallet/common/utils/DeviceInfo.kt

```kotlin
package mx.tpaga.wallet

actual class DeviceInfo {
    actual fun getOsName(): String = "iOS"
}
```

common_iosMain/kotlin/mx/tpaga/wallet/common/utils/DeviceInfo.kt


SignUpUseCase.kt
SessionRepositoryImpl.kt
LocalPreferenceStorage.kt
LocalPreferenceStorage.kt
LocalPreferenceStorage.kt
Expect
Actual
Actual

```
expect class LocalPreferenceStorage: KeyValueStorage
```

commonMain/kotlin/mx/tpaga/wallet/common/local/LocalPreferenceStorage.kt

```kotlin
interface KeyValueStorage {

    fun save(key: String, value: String)
    fun retrieve(key: String) : String?
    fun remove(key: String)

}
```

commonMain/kotlin/mx/tpaga/wallet/common/local/KeyValueStorage.kt

```kotlin
import android.content.Context
import android.content.SharedPreferences

actual class LocalPreferenceStorage(context: Context) : KeyValueStorage {

    companion object {
        const val PREFERENCES = "package"
    }

    private val preferences: SharedPreferences by lazy {
        context.getSharedPreferences(PREFERENCES, Context.MODE_PRIVATE)
    }

    override fun save(key: String, value: String) {
        preferences.edit().putString(key, value).apply()
    }

    override fun retrieve(key: String): String? {
        return preferences.getString(key, null)
    }

    override fun remove(key: String) {
        preferences.edit().remove(key).apply()
    }

}
```

common/kotlin/java/mx/tpaga/wallet/common/local/LocalPreferenceStorage.kt

```kotlin
import platform.Foundation.NSUserDefaults

actual class LocalPreferenceStorage : KeyValueStorage {

    private val delegate: NSUserDefaults = NSUserDefaults.standardUserDefaults()

    override fun save(key: String, value: String) {
        delegate.setObject(value, key)
    }

    override fun retrieve(key: String): String? = delegate.stringForKey(key)

    override fun remove(key: String) {
        delegate.removeObjectForKey(key)
    }

}
```

common_iosMain/kotlin/java/mx/tpaga/wallet/common/local/LocalPreferenceStorage.kt

Consideraciones

- iOS lo debe implementar en Kotlin
- No es Kotlin puro por la interoperabilidad con Objective C -> código raro

expect / actual

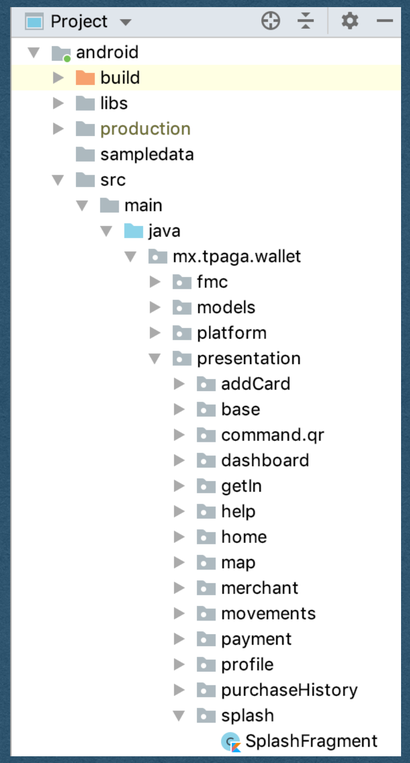
- Donde los uso?
  - Casos de usos
  - Presenters
- Cuales tenemos?
  - Time
  - DeviceInfo
  - LocalPreferenceStorage
  - Contactos
  - Dispatchers
  - UUID …
  - Problemas ? -> Location

Arquitectura

- Considere que tiene sentido compartir o no
- iOS != Android por el framework. iOS mas capas.

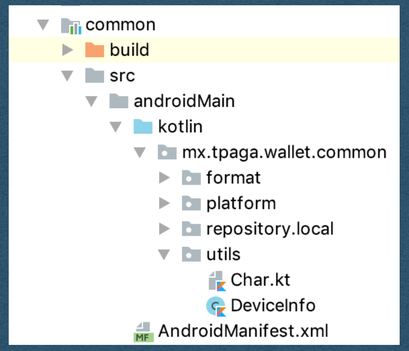
Estructura Proyecto

Android

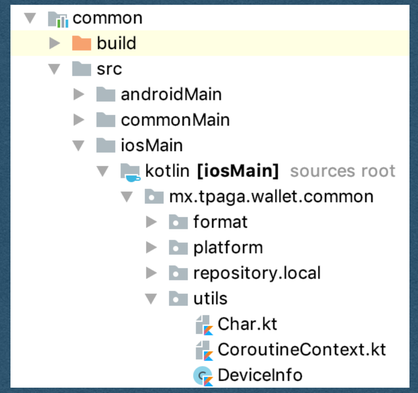


Common

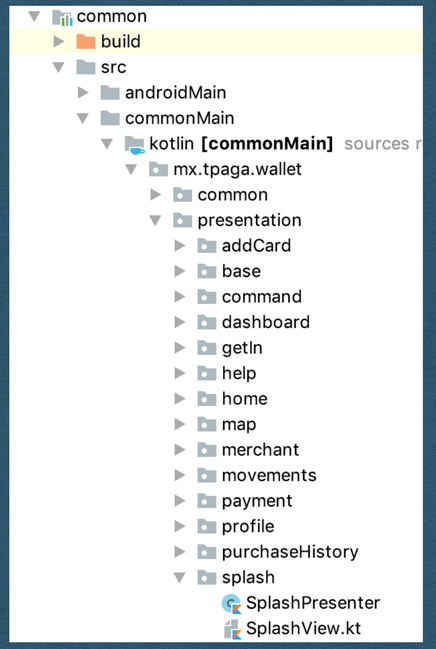
androidMain .



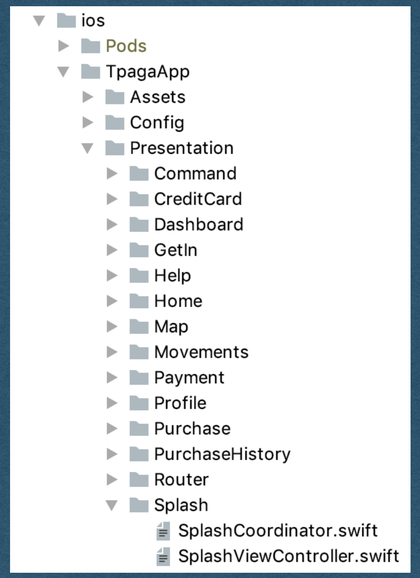
iosMain .



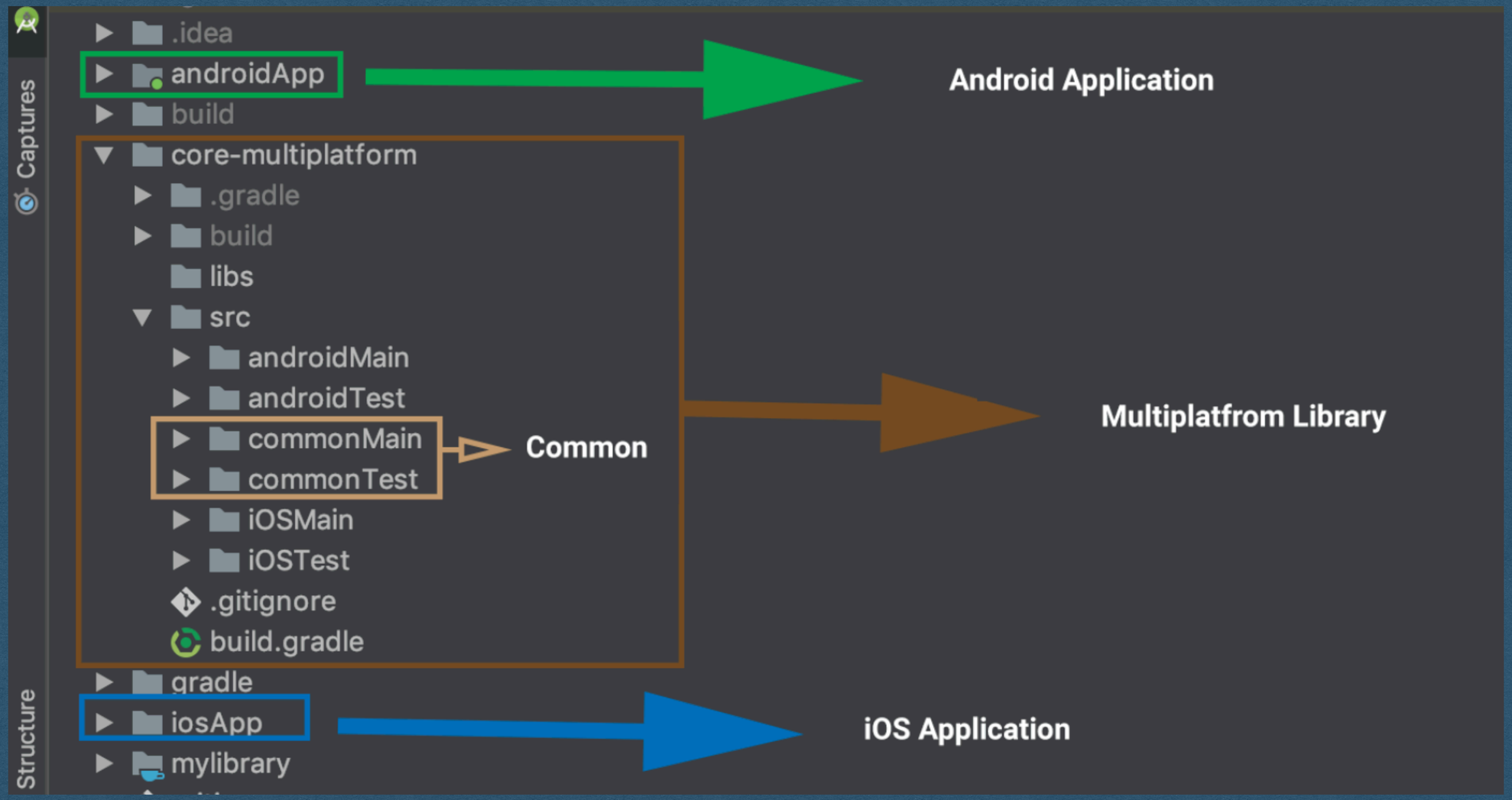
commonMain .



iOS



Estructura Proyecto



Common Gradle

```
apply plugin: 'kotlin-multiplatform'
apply plugin: 'kotlinx-serialization'
apply plugin: 'com.android.library'

kotlin {

    targets {
        fromPreset(presets.android, 'android')

        final def iOSTarget = presets.iosX64
        fromPreset(iOSTarget, 'ios')
    }
}
```

Common Gradle

```
sourceSets {

    commonMain {
        dependencies {
            // Http API communication client
            implementation "io.ktor:ktor-client-core:$ktor_version"
            implementation "io.ktor:ktor-client-json:$ktor_version"
            implementation "io.ktor:ktor-client-logging:$ktor_version"
        }
    }

    commonTest {
        dependencies {
            implementation "org.jetbrains.kotlin:kotlin-test-common"
            implementation "org.jetbrains.kotlin:kotlin-test-annotations-common"
        }
    }

    androidMain {
        dependencies {
            // Http API communication client runtime: JVM implementation
            implementation "io.ktor:ktor-client-core-jvm:$ktor_version"
            implementation "io.ktor:ktor-client-json-jvm:$ktor_version"
            implementation "io.ktor:ktor-client-logging-jvm:$ktor_version"
            implementation "io.ktor:ktor-client-okhttp:$ktor_version"
        }
    }

    androidTest {
        dependencies {
            implementation 'org.jetbrains.kotlin:kotlin-test'
            implementation 'org.jetbrains.kotlin:kotlin-test-junit'
        }
    }

    iosMain {
        dependencies {
            // Http API communication client runtime: iOS implementation
            implementation "io.ktor:ktor-client-ios:$ktor_version"
            implementation "io.ktor:ktor-client-core-native:$ktor_version"
            implementation "io.ktor:ktor-client-json-native:$ktor_version"
            implementation "io.ktor:ktor-client-logging-native:$ktor_version"
        }
    }
}
```

Common Gradle

```
task packForXCode(type: Sync) {
    final File frameworkDir = new File(buildDir, "xcode-frameworks")
    final String mode = project.findProperty("XCODE_CONFIGURATION")?.toUpperCase() ?: 'DEBUG'

    inputs.property "mode", mode
    dependsOn kotlin.targets.iOS.compilations.main.linkTaskName("FRAMEWORK", mode)

    from { kotlin.targets.iOS.compilations.main.getBinary("FRAMEWORK", mode).parentFile }
    into frameworkDir

    doLast {
        new File(frameworkDir, 'gradlew').with {
            text = "#!/bin/bash\nexport 'JAVA_HOME=${System.getProperty("java.home")}'\ncd '$
{rootProject.rootDir}'\n./gradlew \$@\n"
            setExecutable(true)
        }
    }
}

tasks.build.dependsOn packForXCode
```

Diferencias entre plataformas

- Localización
  - Android: FuseLocation - GpsLocation
  - iOS: CoreLocation

Notas

- La interoperabildad es con Objetive C y no con Swift
- Valores de parámetros por defecto: Kotlin- Swift - Objective C .
- Compilar en Android studio para generar el build del framework para iOS
- Generics

```kotlin
data class Node<T>(val value: T, val next: Node<T>? = nu
```

```
__attribute__((objc_subclassing_restricted))
__attribute__((swift_name("Node")))
@interface EverythingNode : KotlinBase
- (instancetype)initWithValue:(id _Nullable)value next:(EverythingNode * _Nullable)next
__attribute__((swift_name("init(value:next:)")))
__attribute__((objc_designated_initializer));
// more declarations here
@end;
```

```
let node = Node<NSString>(value: "hello!", next: nil)
```

Cannot specialize non-generic type 'Node'

Replace '<NSString>' with ''

Fix

```
object CarNode {
  fun getCarNode(): Node<Car> = Node(Car(2019, "Tesla", "Model S"))
}
```

```
func test() {
    let currentCarNode = CarNode.init().getCarNode()
    let currentCar = currentCarNode.value
    print(currentCar.model)
}
```

Value of type 'Any?' has no member 'model'

```
func test() {
    let currentCarNode = CarNode.init().getCarNode()
    let currentCar = currentCarNode.value as! Car
    print(currentCar.model)
}
```

```
kotlin {
    iosX64 {
        binaries {
            framework {
                freeCompilerArgs += "-Xobjc-generics"
            }
        }
    }
}
```

```
__attribute__((objc_subclassing_restricted))
__attribute__((swift_name("Node")))
@interface EverythingNode<T> : KotlinBase
- (instancetype)initWithValue:(T _Nullable)value next:(EverythingNode<T> * _Nullable)next
__attribute__((swift_name("init(value:next:)")))
__attribute__((objc_designated_initializer));
// more declarations here
@end;
```

```
let node = Node<NSString>(value: "hello!", next: nil)
let length = node.value.length
```

Value of optional type 'NSString?' must be unwrapped to refer to member 'length' of wrapped base type 'NSString'

Chain the optional using '?' to access member 'length' only for non-'nil' base values — Fix

Force-unwrap using '!' to abort execution if the optional value contains 'nil' — Fix

```
let node = Node<NSString>(value: "hello!", next: nil)
let length = node.value?.length
```

```
data class Node<T : Any>(val value: T, val next: Node<T>? =
```

```objc
__attribute__((objc_subclassing_restricted))
__attribute__((swift_name("Node")))
@interface EverythingNode<T> : KotlinBase
- (instancetype)initWithValue:(T)value next:(EverythingNode<T> * _Nullable)next
__attribute__((swift_name("init(value:next:)")))
__attribute__((objc_designated_initializer));
// more declarations here
@end;
```

```
func test() {
    let node = Node<NSString>(value: "hello!", next: nil)
    let length = node.value.length
    print(length)

    let currentCarNode = CarNode.init().getCarNode()
    let currentCar = currentCarNode.value
    print(currentCar.model)
}
```

Dificultades

- Debuggear para iOS
- Hacerlo en Android primero

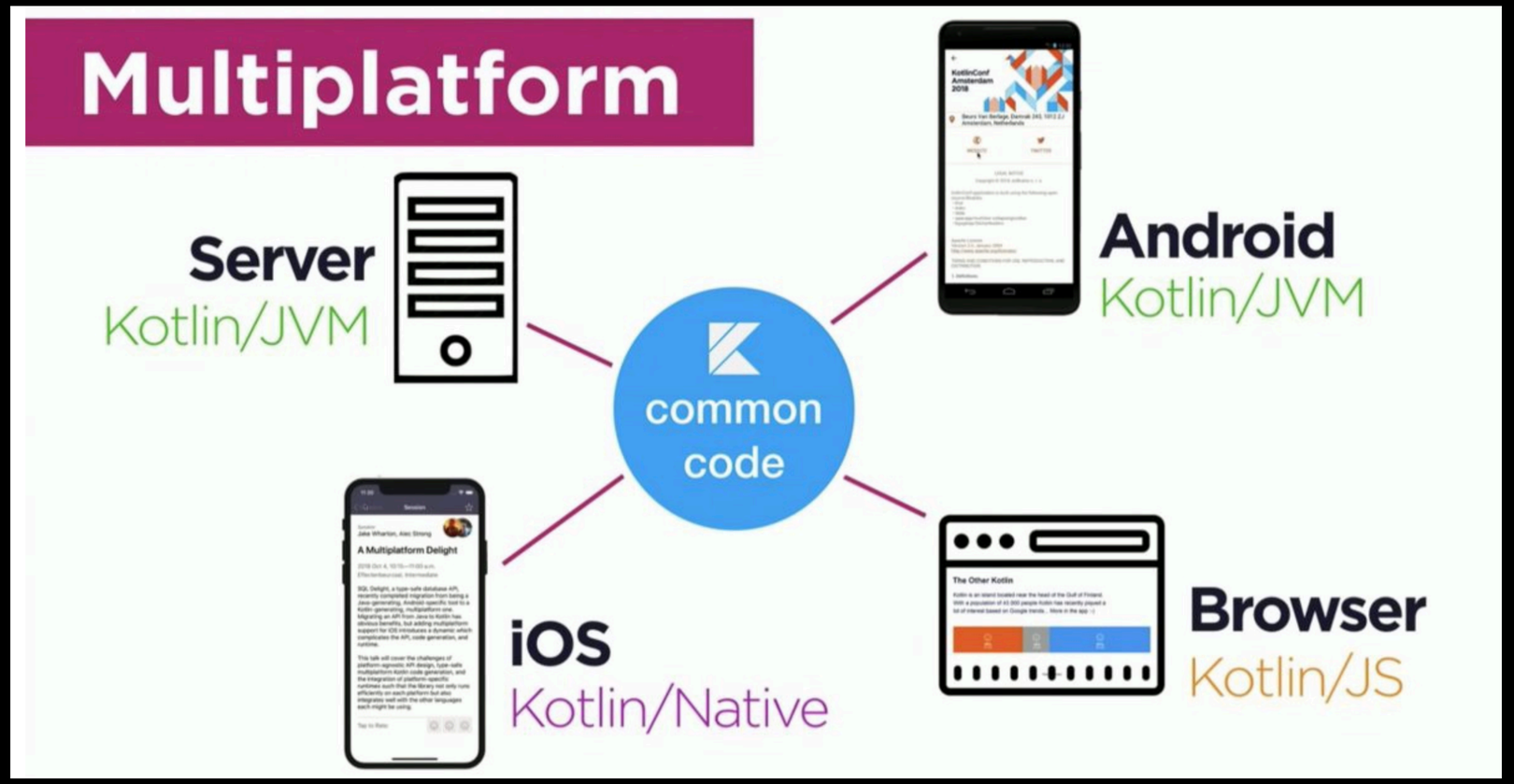
Para pensar..

- Errores con la librería mockk -> No es soportada
- Test con JVM
- Pensar en que más se puede compartir

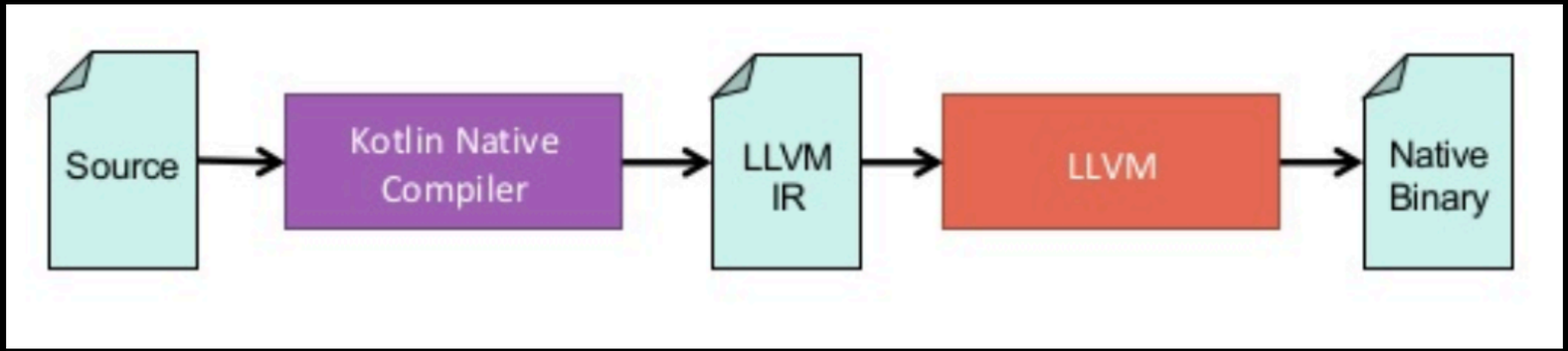
Beneficios

- 1 código base que escribir, testear, mantener
- Errores centralizados
- Implementaciones de diseño nativas
- Android - iOS - Web - Backend (1 sola base de código)

Beneficios


Multiplatform
Server
Kotlin/JVM
common
code
Android
Kotlin/JVM
iOS
Kotlin/Native
Browser
Kotlin/JS

Kotlin Native



Resumen

- Piensa en compartir
- La comunicación lo es todo
- Empodérate
- Comparte lo que ambas plataformas estén dispuestas
- Empieza de a pocos
- Construye confianza
- Con el tiempo el impacto aumenta

GRACIAS

Laura De la Rosa @lauramdelarosa

@MedellinAndroid

#KotlinConfGlobal2020