

Arquitectura Mobile





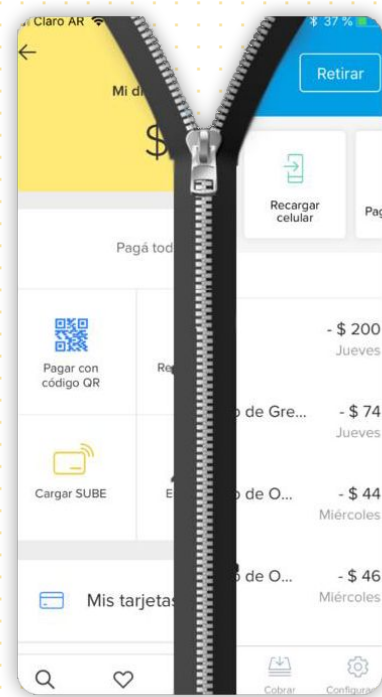
¿Quiénes somos?



Arquitectura Mobile

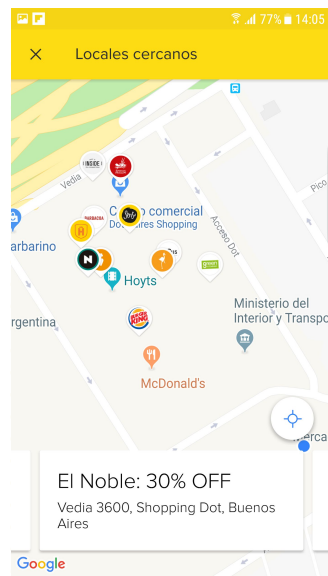
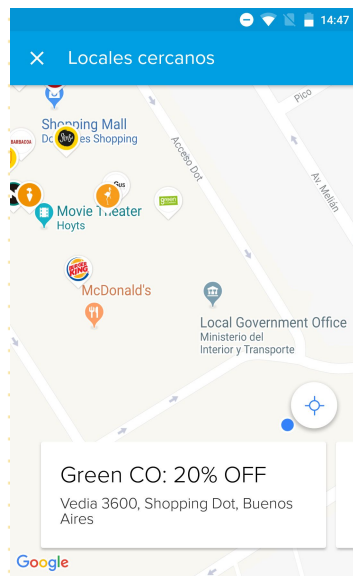
- Velar por el buen desarrollo las apps
- Mejorar la experiencia de trabajo a los equipos
- Soluciones de arquitectura
- Soporte
- Salidas a producción

¿A qué vinimos?



¿Qué impulsó a la nueva arquitectura?

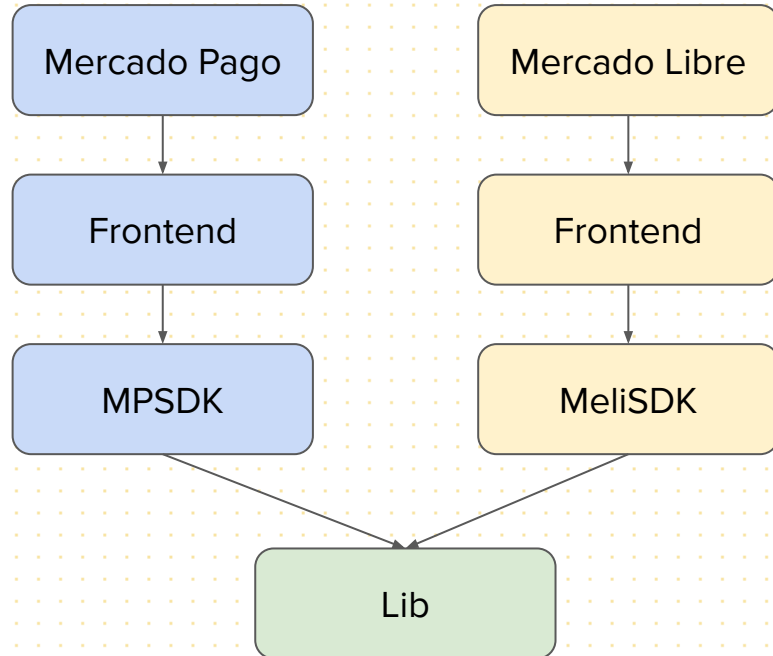
Módulos cross aplicación





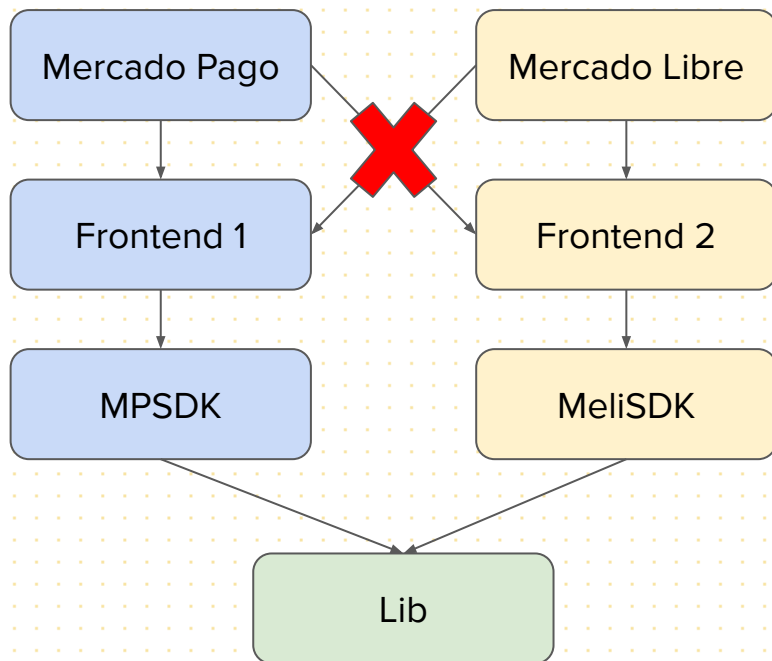
**¿Cómo estaba compuesta
nuestra arquitectura?**

SDK + Libs utilitarias



- Frontends específicos a la aplicación
- Existencia de SDKs, acoplada al negocio
- Librerías no agnósticas al consumidor

¿Que debíamos resolver?

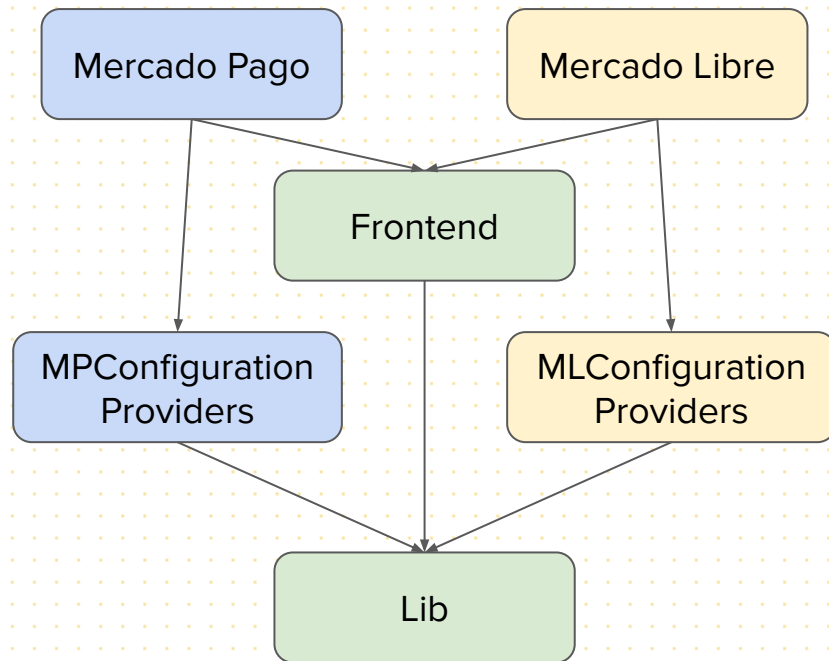


- Frontends ligados al contexto
- Librerías acopladas al negocio
- Diseño inextensible por uso excesivo de herencias

Configuration Providers

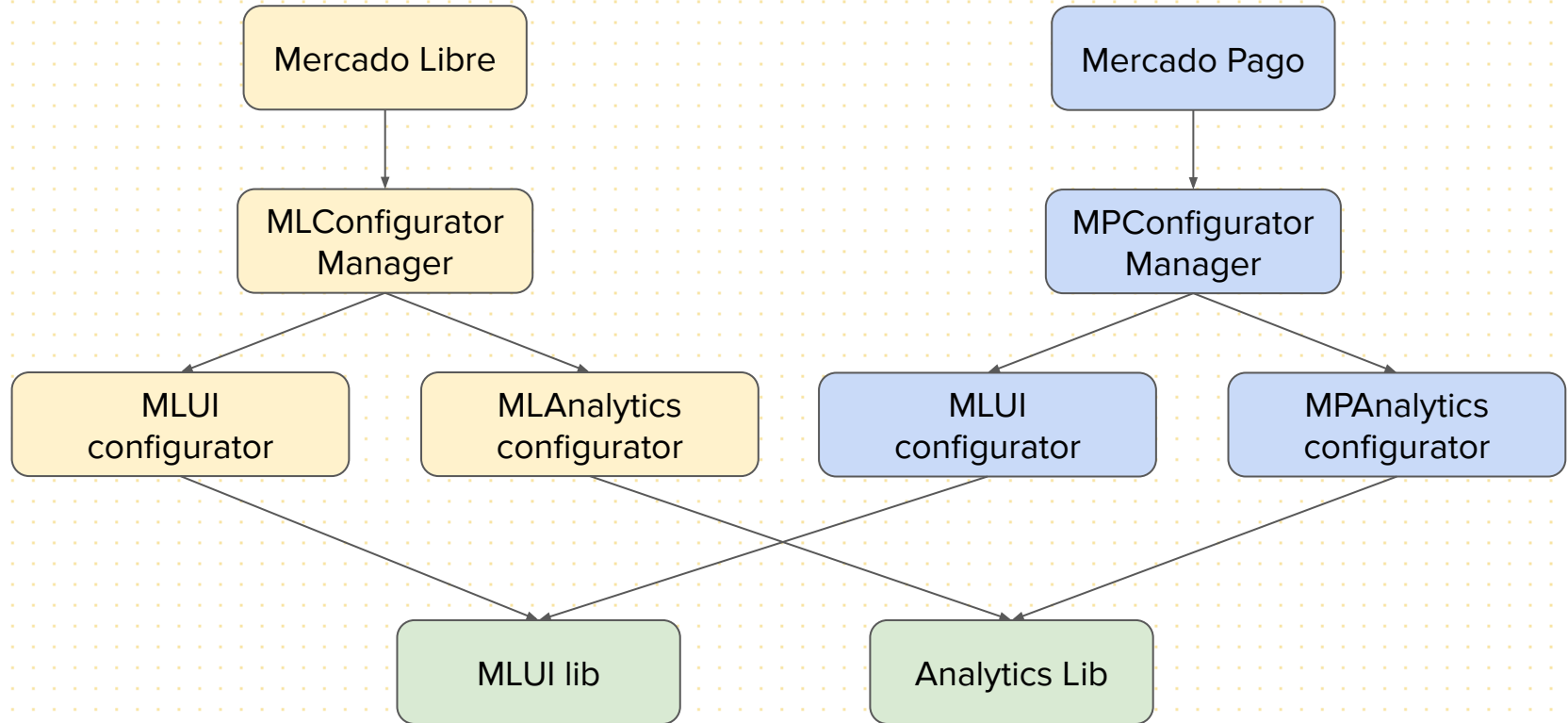


¿Qué es un Configuration Provider?



- Configura librerías
- Transparente para frontends
- Se ejecuta al inicio de la app

¿Cómo funcionan?





¿Como crear un Configuration Provider?

```
protocol MLCPConfigurable {  
    func configure()  
}
```



```
// Lib de UI
```

```
protocol MLUIStyleSheet {  
    func primaryColor() -> UIColor  
    // .  
    // .  
    // .  
}  
//
```

```
// Lib de Providers
```

```
class MLCPUIConfigurator: MLCPCConfigurable, MLUIStyleSheet {  
    func configure() {  
        MLIUStyleSheetManager.set(styleSheet: self)  
    }  
  
    func primaryColor() -> UIColor {  
        return UIColor.yellow  
    }  
}
```



```
class MLCPCConfigurationManager {
```

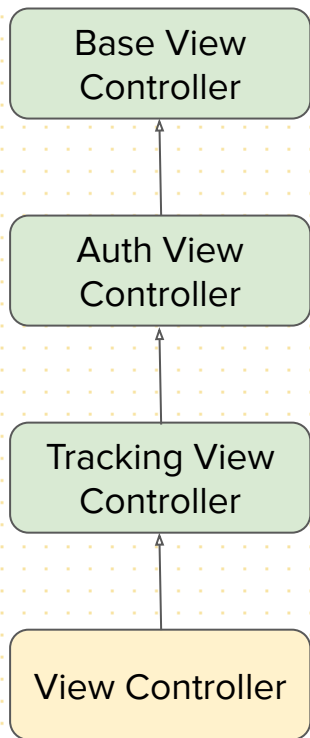
```
    static func configurators() -> [MLCPCConfigurable] {  
        return [  
            MLCPUIConfigurator()  
            // .  
            // .  
            // .  
        ]  
    }
```

```
    static func configure() {  
        for configurator in configurators() {  
            configurator.configure()  
        }  
    }  
}
```

Behaviours

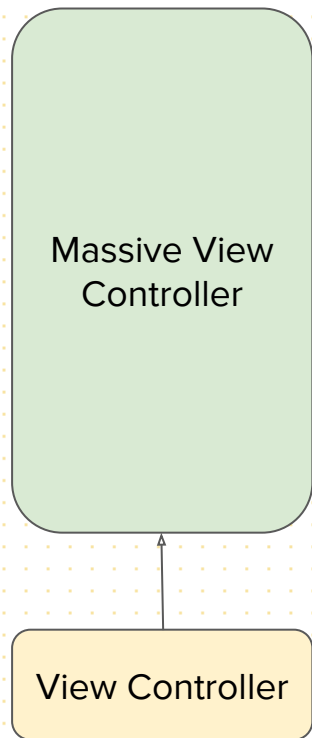


Diseño de vistas anterior



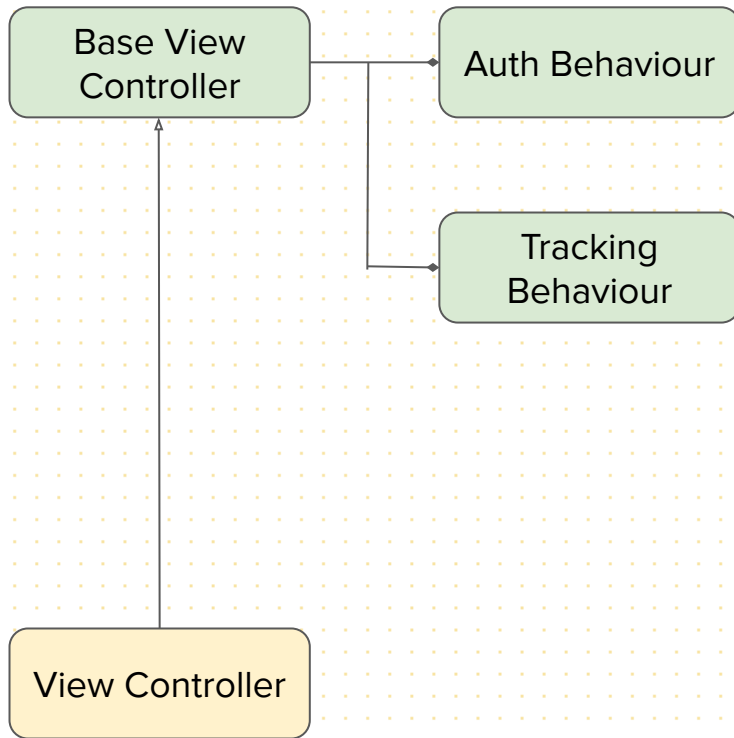
- **Falta de abstracción:** Teníamos que agregar herencias en X lugar para incorporar comportamientos
- **Diseño inextensible:** Sobreuso de overrides y herencias
- **Falta de portabilidad:** No podíamos modificar comportamientos a nivel aplicativo

Diseño de vistas anterior




- **Falta de abstracción:** Teníamos que agregar herencias en X lugar para incorporar comportamientos
- **Diseño inextensible:** Sobreuso de overrides y herencias
- **Falta de portabilidad:** No podíamos modificar comportamientos a nivel aplicativo

Behaviours



- Es un **hook al lifecycle** del View Controller
- **Evita la extensión en cadena** y **Massive** View Controller
- **Delega comportamientos** atados al lifecycle del View Controller
- **Desacopla funcionalidades** y módulos aislados



```
protocol MLBaseBehaviourProtocol {  
    func viewDidLoad()  
    func viewWillAppear(_ animated: Bool)  
    // .  
    // .  
    // .  
}
```

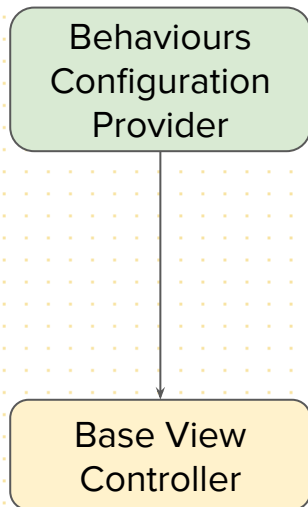


```
class MLBehaviourManager {  
  
    private var behaviours: [MLBaseBehaviourProtocol]  
  
    public func addBehaviour() {  
        // Add Behaviour  
    }  
  
    public func remove(behaviour: MLBaseBehaviourProtocol) {  
        // Removes Behaviour  
    }  
  
    public func getBehaviour(_ type: MLBaseBehaviourProtocol.Type) -> MLBaseBehaviourProtocol {  
        // Returns Behaviour  
    }  
  
    func viewDidLoad() {  
        for behaviour in behaviours {  
            behaviour.viewDidLoad()  
        }  
    }  
  
    // .  
    // .  
    // .  
}
```



```
class MLBaseViewController: UIViewController {  
  
    let behaviourManager: MLBehaviourManager  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        behaviourManager.viewDidLoad()  
    }  
  
    // .  
    // .  
    // .  
}
```

Behaviours Obligatorios



- Se **agregan al comienzo** de cada aplicación
- Se van a ejecutar en **todos los ViewControllers.**
- No pueden ser sobrescritos ni borrados.
- Ejemplos analytics, tracking



```
class MLRequiredBehaviours {  
  
    static let shared = MLRequiredBehaviours()  
  
    public var behaviours: [MLBaseBehaviourProtocol] = []  
  
    func add(requiredBehaviours: [MLBaseBehaviourProtocol]) {  
        // Add required behaviours  
    }  
}
```



```
class MLBehaviourManager {
```

```
    private var behaviours = MLRequiredBehaviours.shared.behaviours
```

```
    public func addBehaviour() {  
        // Add Behaviour  
    }
```

```
    public func remove(behaviour: MLBaseBehaviourProtocol) {  
        // Removes Behaviour  
    }
```

```
    public func getBehaviour(_ type: MLBaseBehaviourProtocol.Type) -> MLBaseBehaviourProtocol {  
        // Returns Behaviour  
    }
```

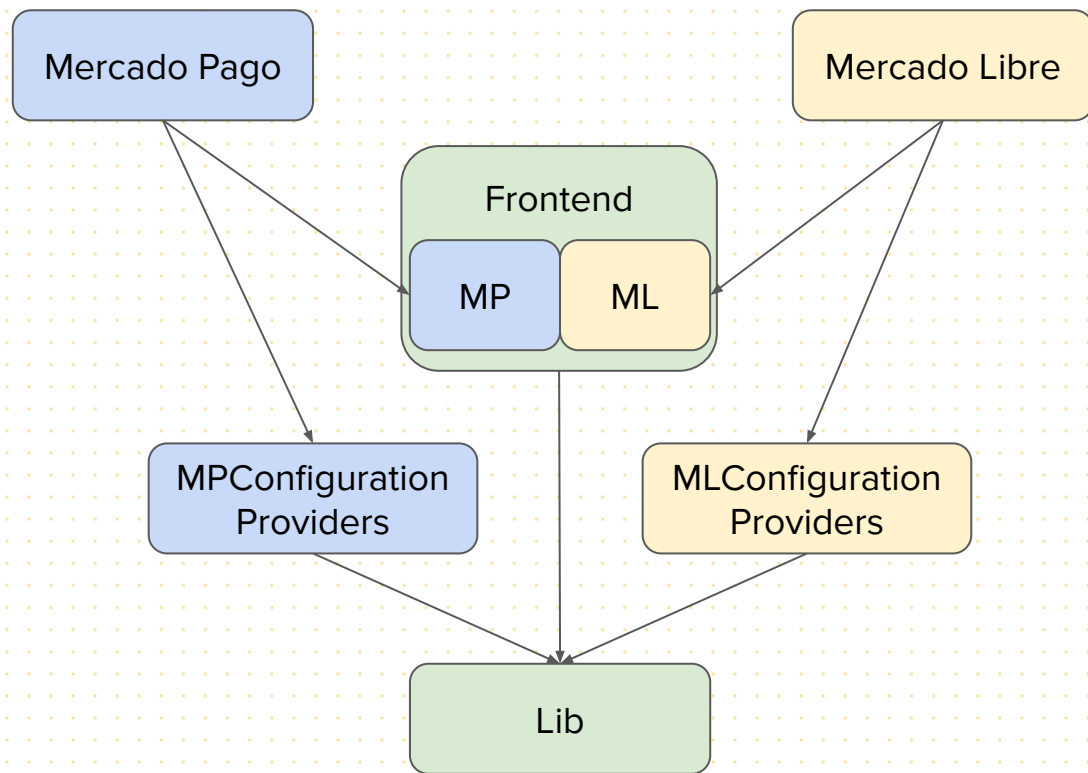
```
    func viewDidLoad() {  
        for behaviour in behaviours {  
            behaviour.viewDidLoad()  
        }  
    }
```


```
    // .  
    // .  
    // .
```

```
}
```


Flavours







```
s.subspec 'MercadoPago' do |mercadopago|  
  mercadopago.source_files = [ 'LibraryComponent' ]  
  mercadopago.resources = [ 'Lib' ]  
  mercadopago.ios.resource_bundle = { 'ISStri' }  
end
```

```
s.subspec 'MercadoLibre' do |mercadolibre|  
  mercadolibre.source_files = [ 'LibraryComponent' ]  
  mercadolibre.resources = [ 'Lib' ]  
  mercadolibre.ios.resource_bundle = { 'ISStri' }  
end
```

Preguntas

