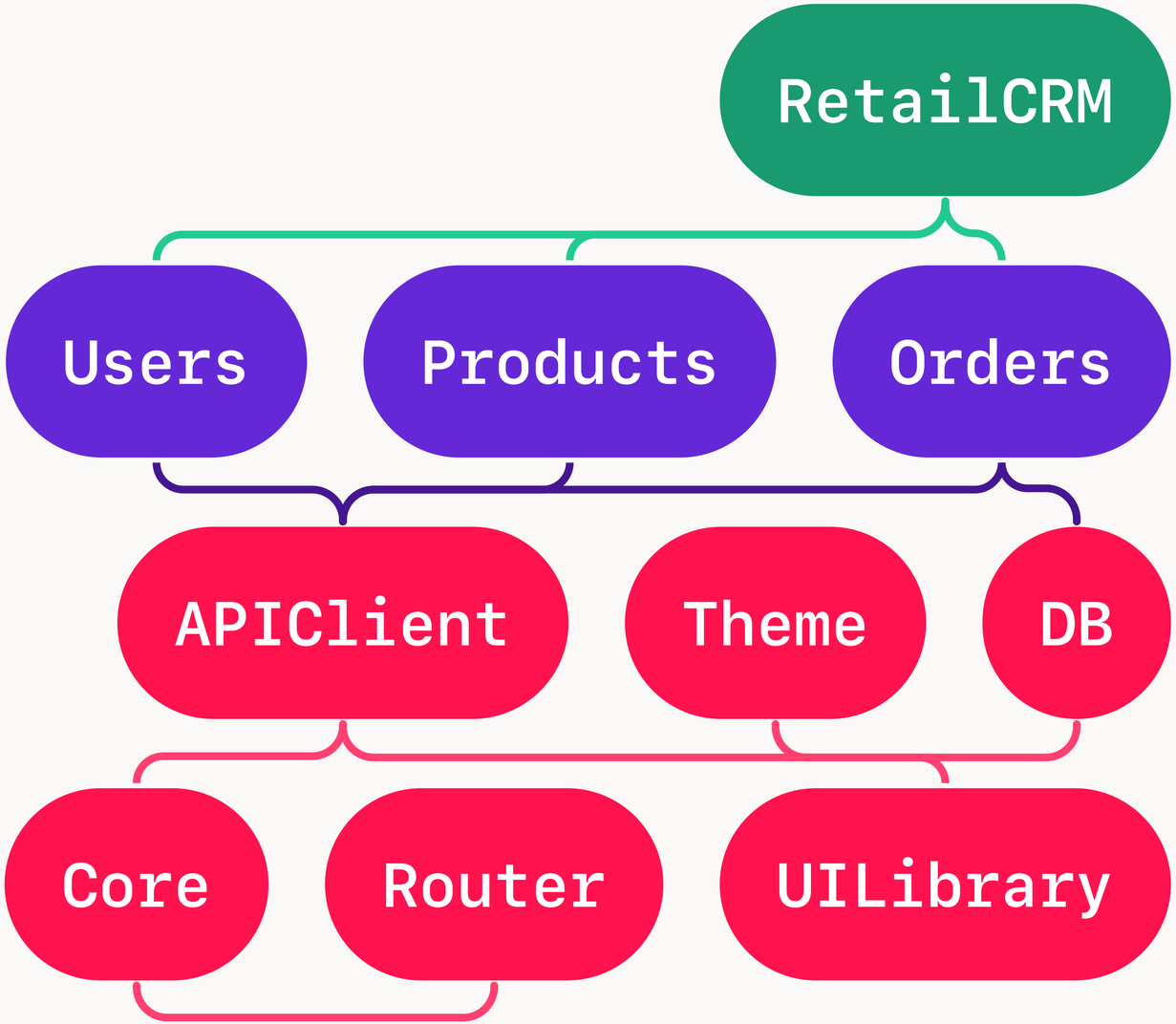
Зачем и как дробить мобильное приложение на µфичи

Илья Харламов

Head of Mobile department at RetailCRM / Simla.com

Кто такие эти ваши микрофичи?



👉 Ключевой принцип

Разработчик должен тратить **минимум** времени на рутинные процессы

ординг #индексация #сборка #тестирование #онбординг #индек

- А зачем оно нам надо?

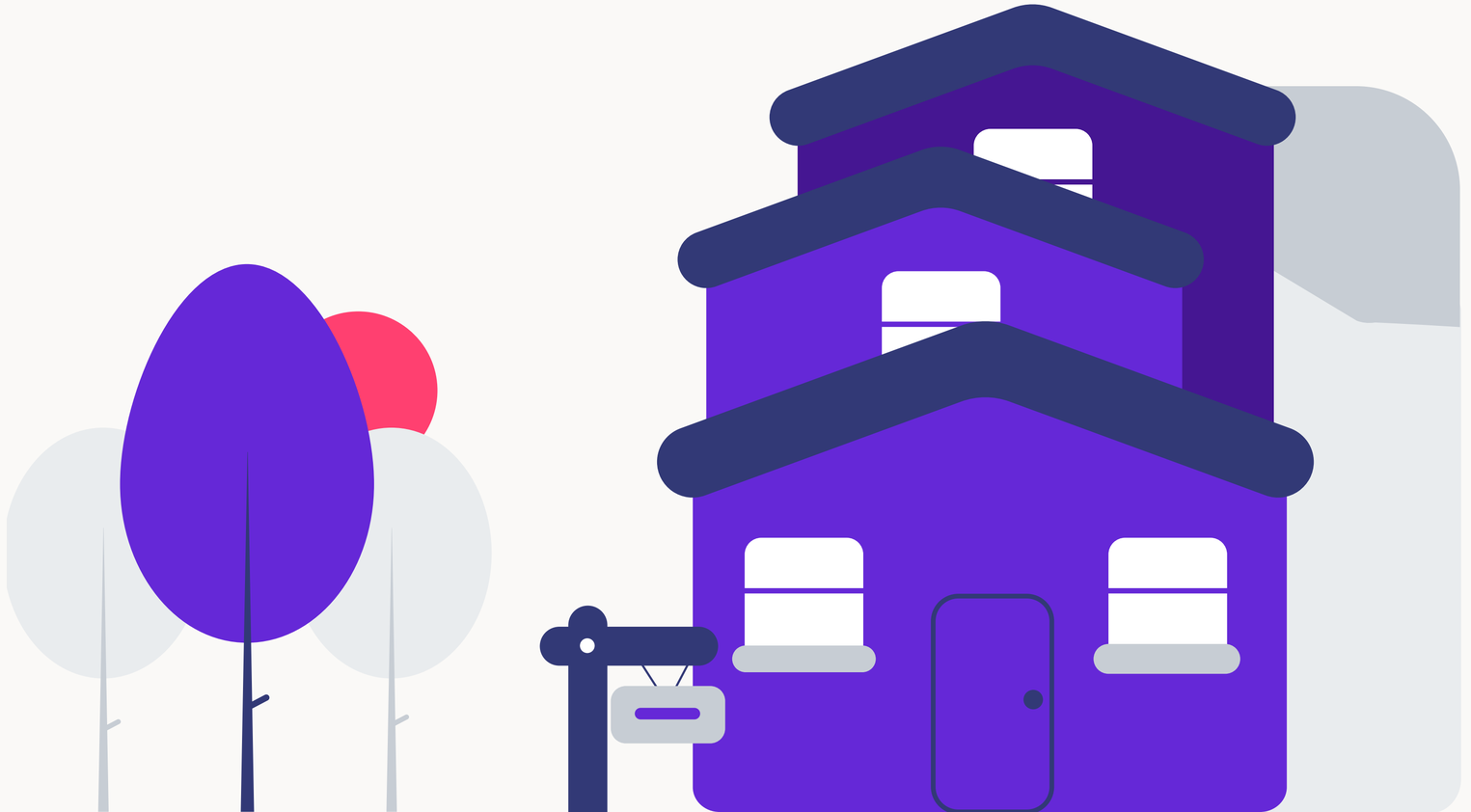
- Да, мне и с монолитом неплохо

Щас расскажу!

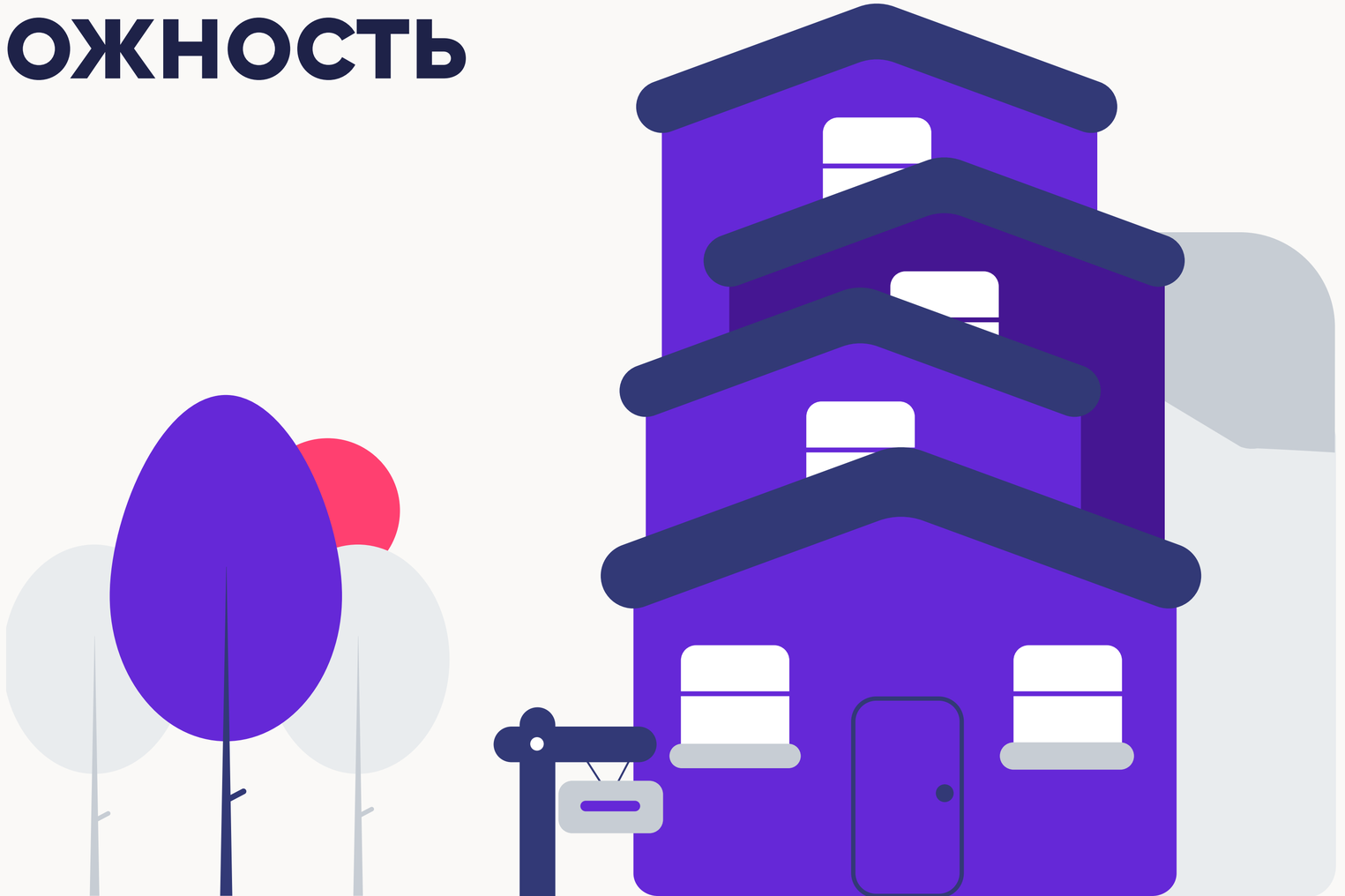
Продукт 📈 растет,
а с ним и сложность

Продукт 📈 растет,
а с ним и сложность

Продукт 📈 растет,
а с ним и сложность

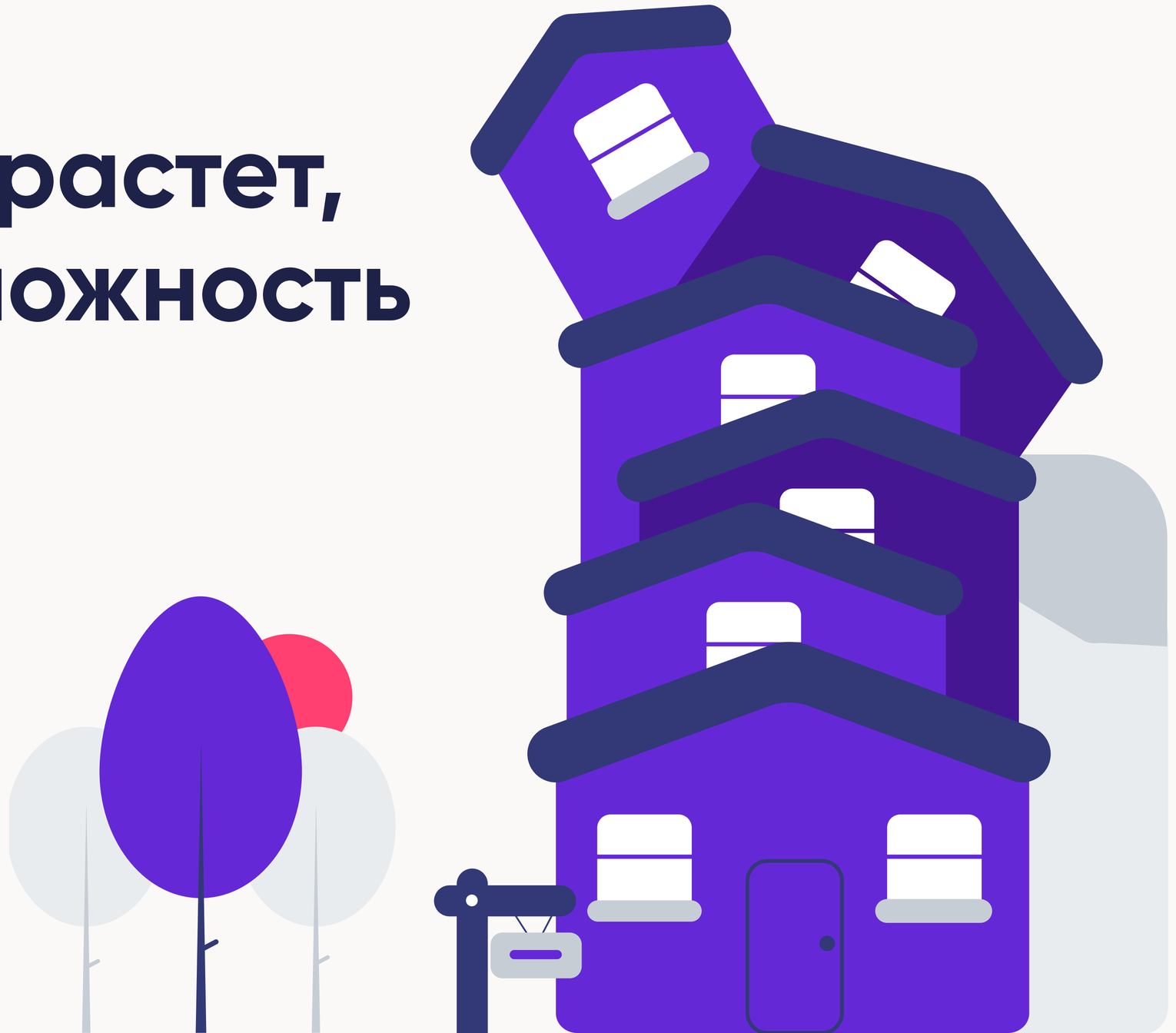


Продукт 📈 растет,
а с ним и сложность



Продукт 📈 растет,
а с ним и сложность

Продукт 📈 растет, а с ним и сложность



Продукт 📈 растет,
а с ним и ~~сложность~~ качество

Профит #1

Скорость сборки / индексации 

До дробления: ❄️ 10 мин / ☀️ 20 сек

После: ❄️ 6 мин / ☀️ 12 сек

Профит #2

Xcode стал меньше глючить

Конец. Спасибо за внимание

Профит #3

Меньше кофликтов и не только в коде

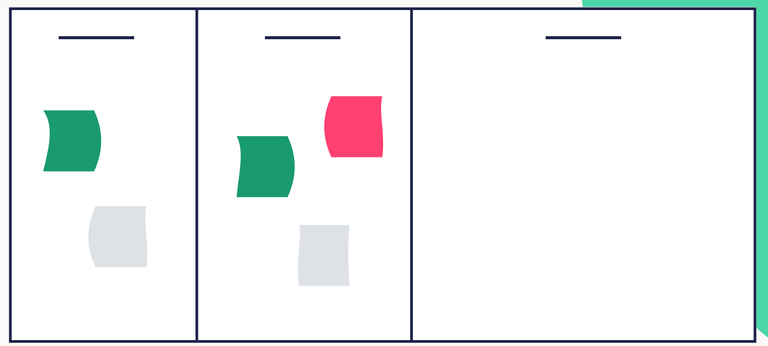
Каждый член команды
работает со своим модулем

И другим не мешает

Профит #4

побочный эффект

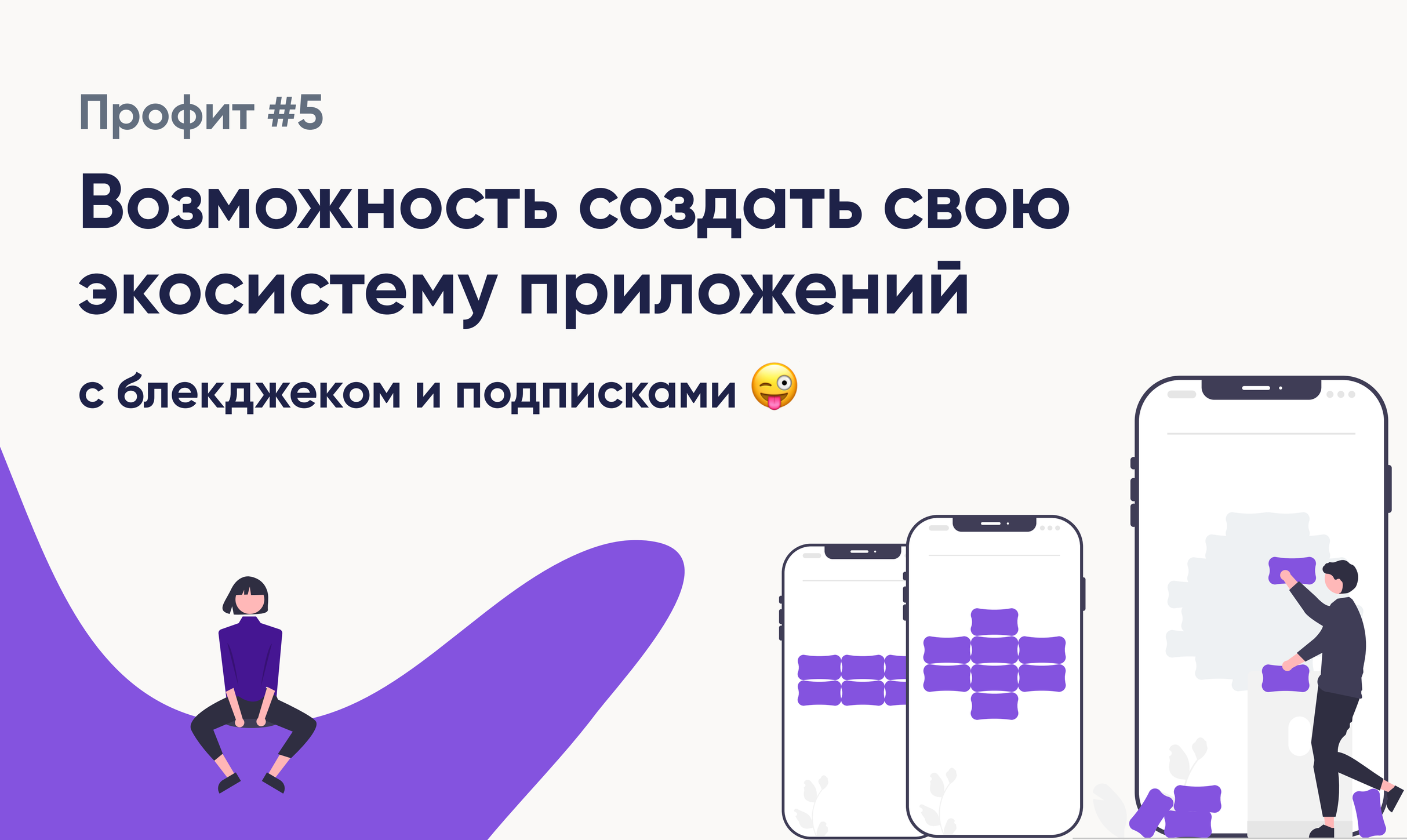
Учимся поддерживать код в порядке



Профит #5

Возможность создать свою экосистему приложений

с блекджеком и подписками 😜



Профит #6

Возможность выделять модули в open-source библиотеки

Graphus

★ 7

★ 64

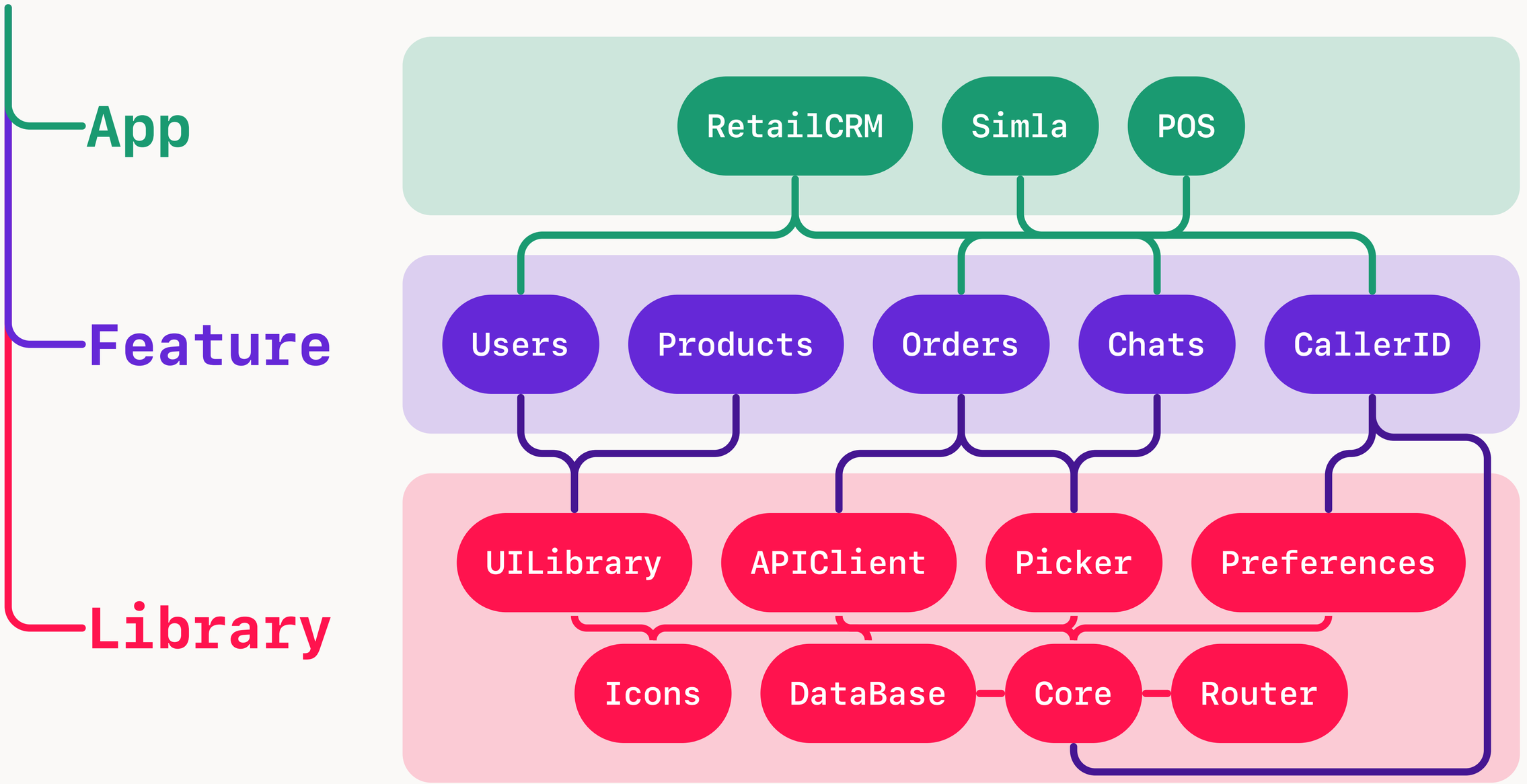
PrettyCards

★ 7

Пример наших библиотек github.com/simla-tech

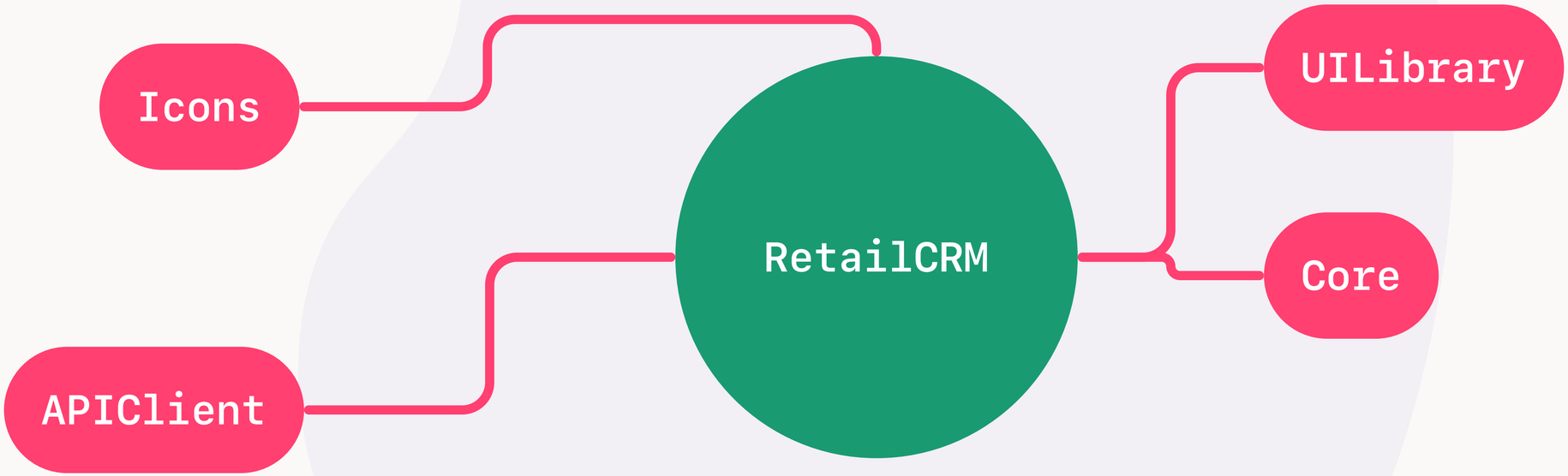
Переходим к архитектуре нашего проекта

Main.workspace

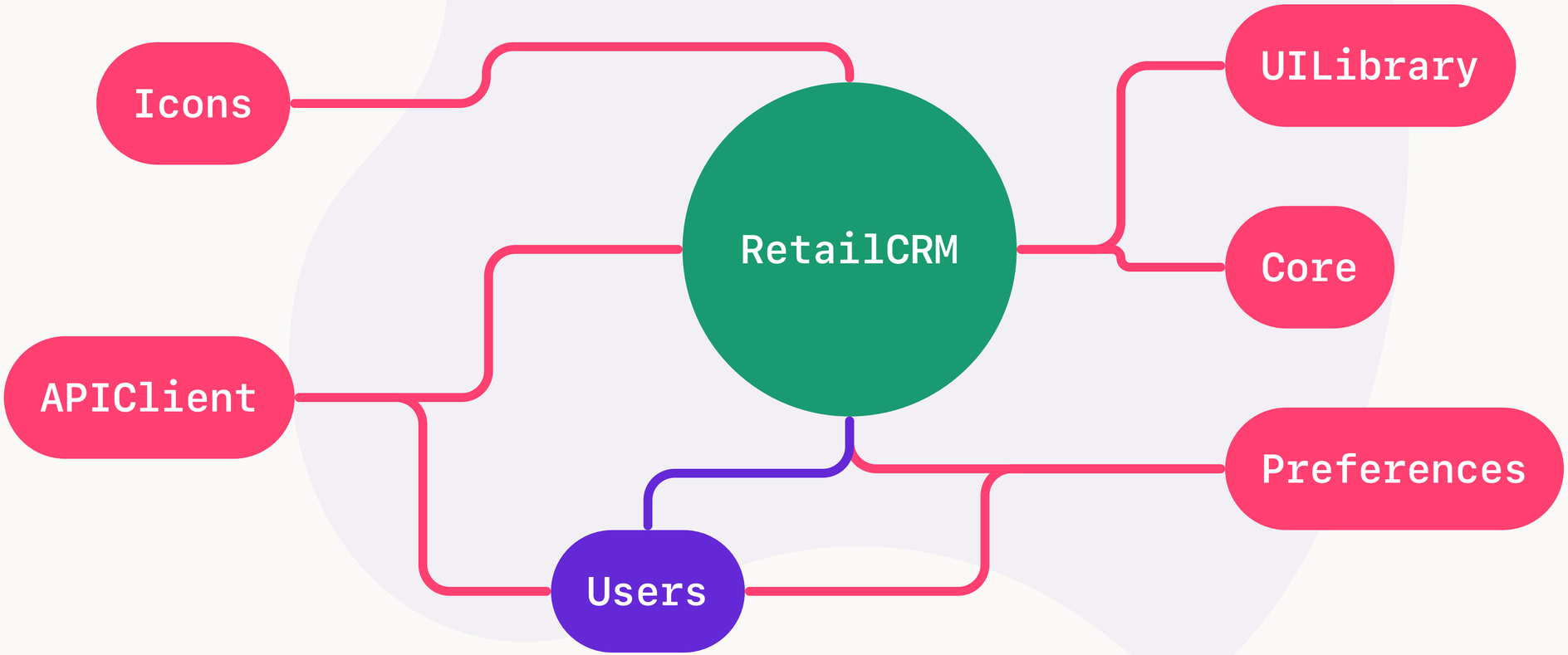


С чего начать?

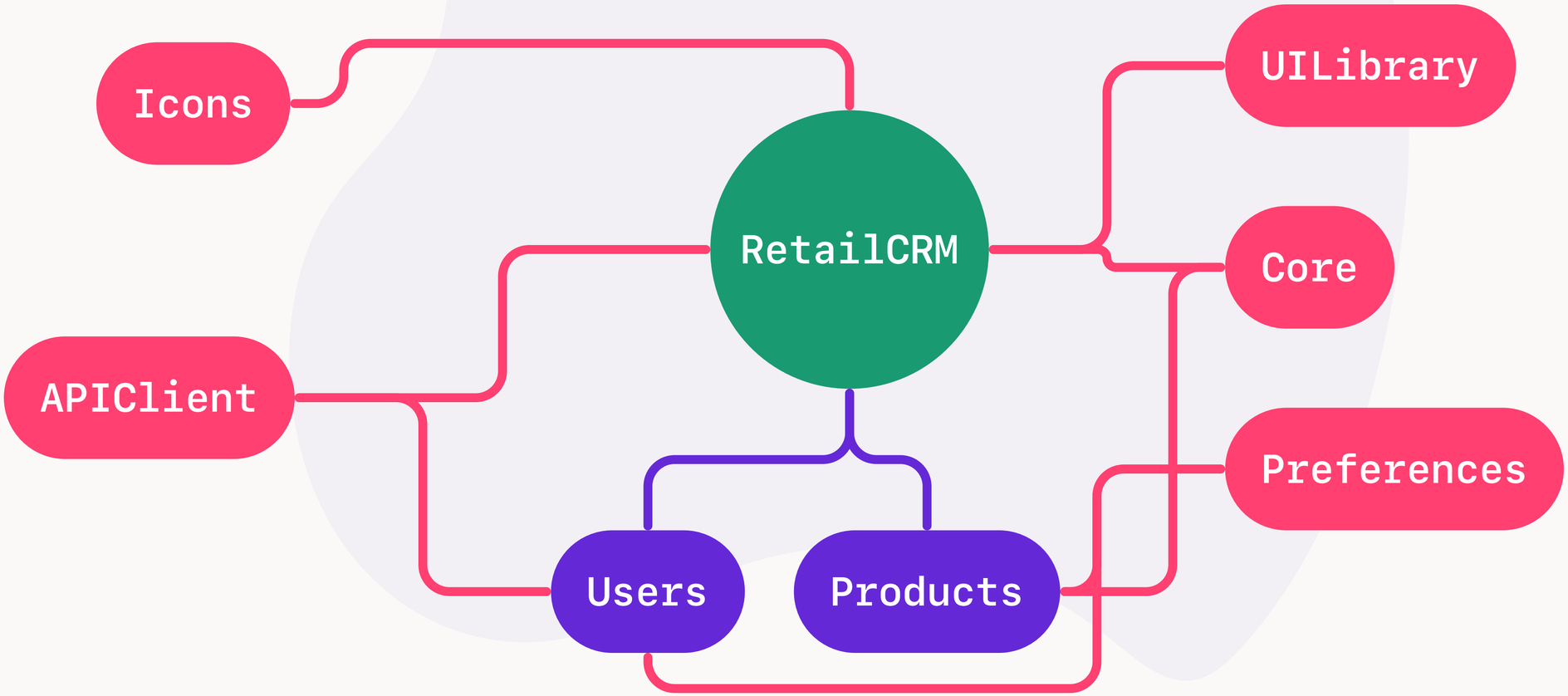
С чего начать?



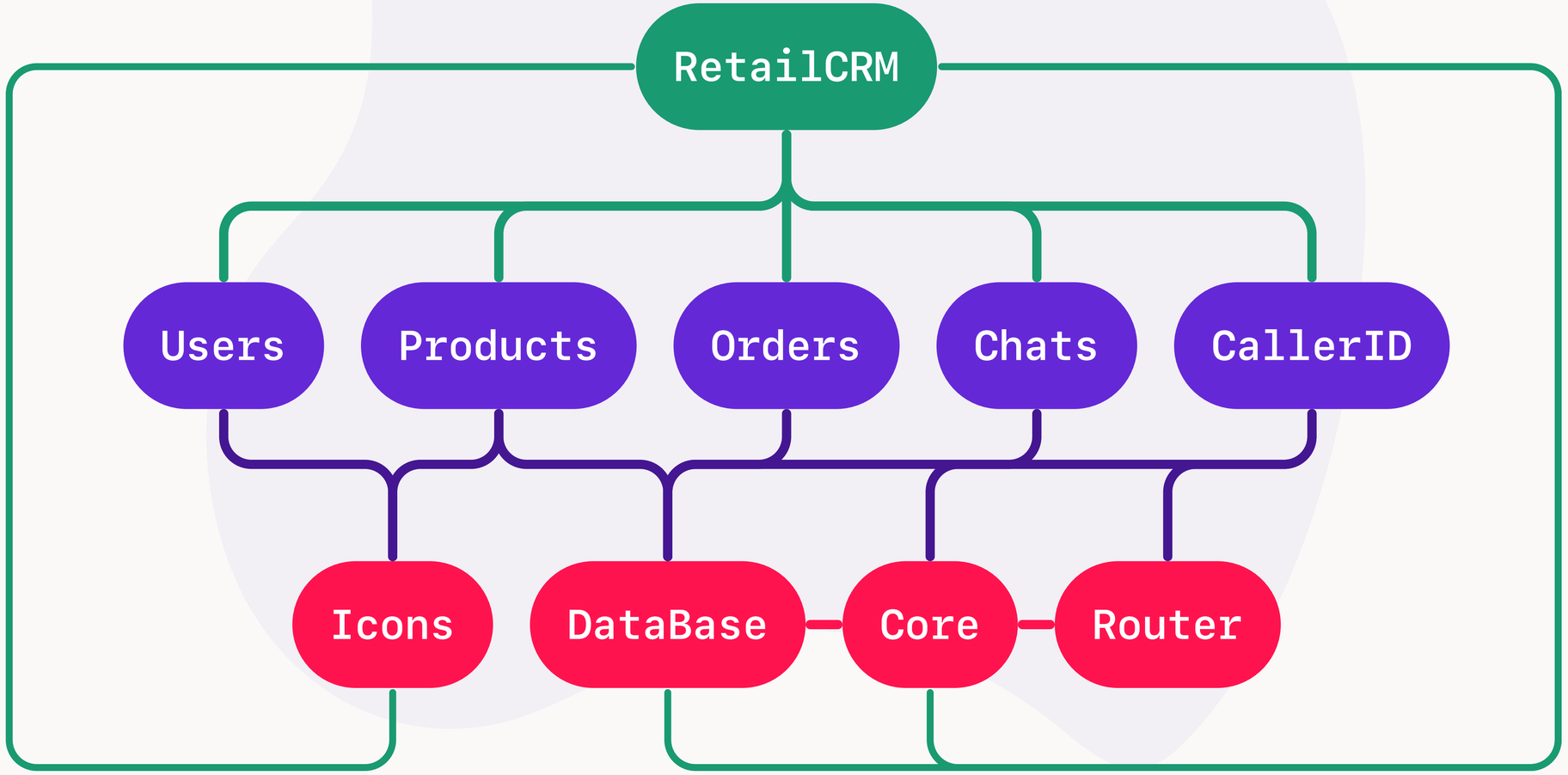
С чего начать?



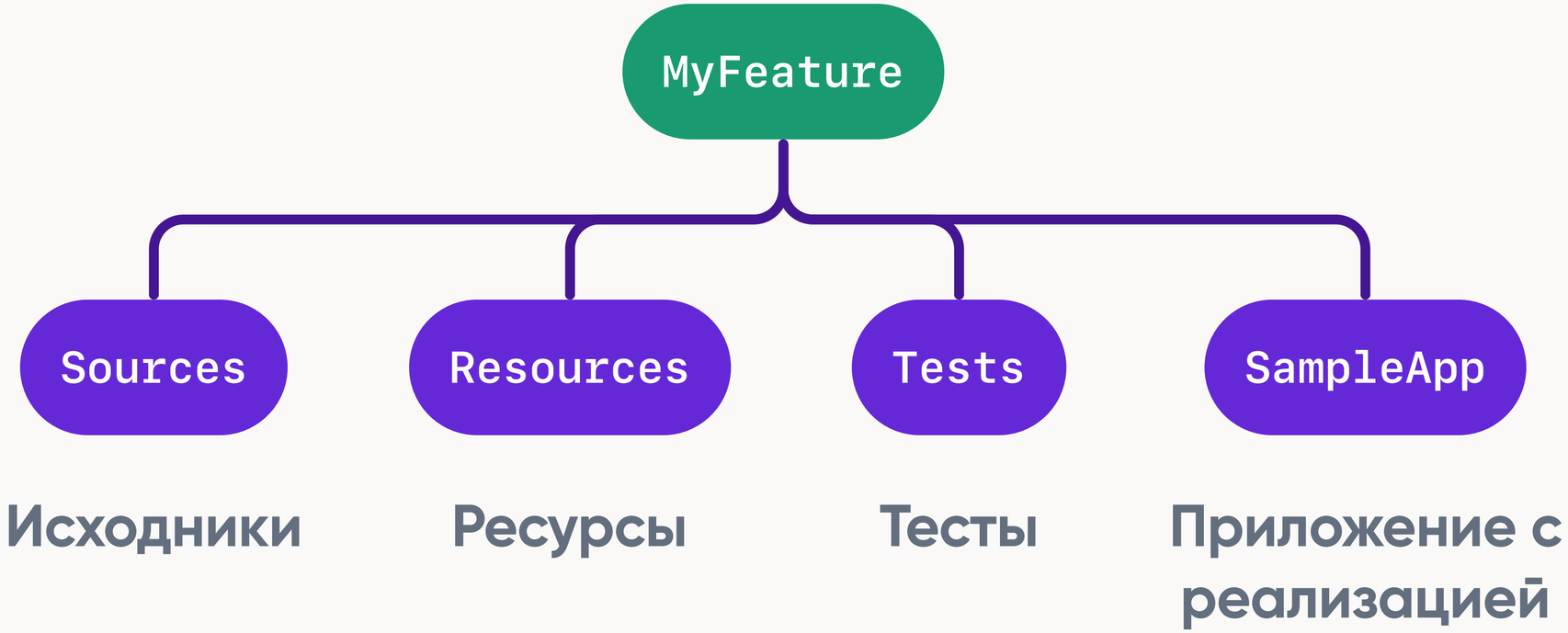
С чего начать?



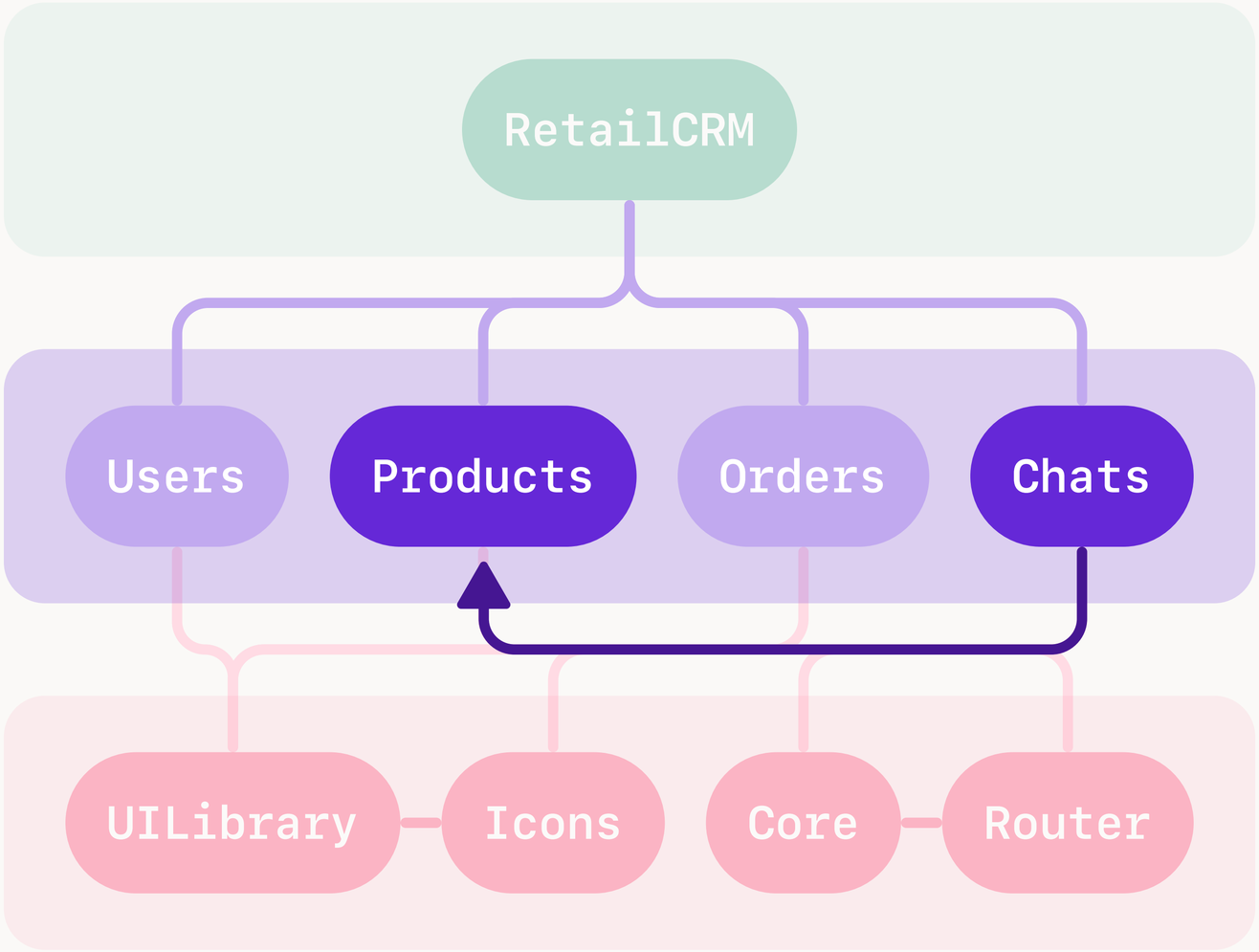
С чего начать?



Как организовать фичу?



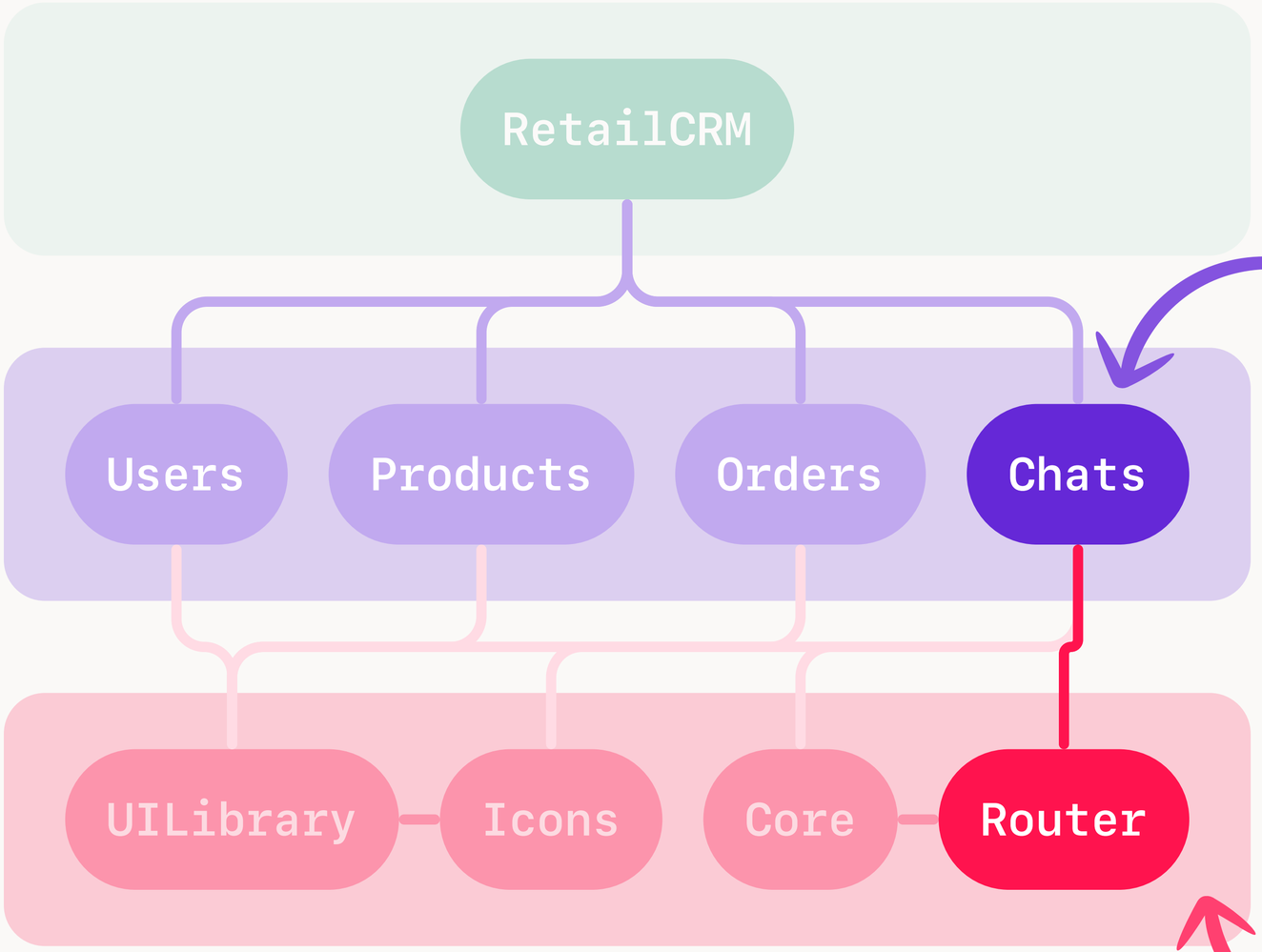
```
import Foundation


public class MyFeature {

    // ...

    public func productDetail(id: Int) -> UIViewController {
        let controller = ProductDetailViewController(id: id, /*...*/)
        return controller
    }

}
```



Роутинг

Как нам из чатов попасть в товар?

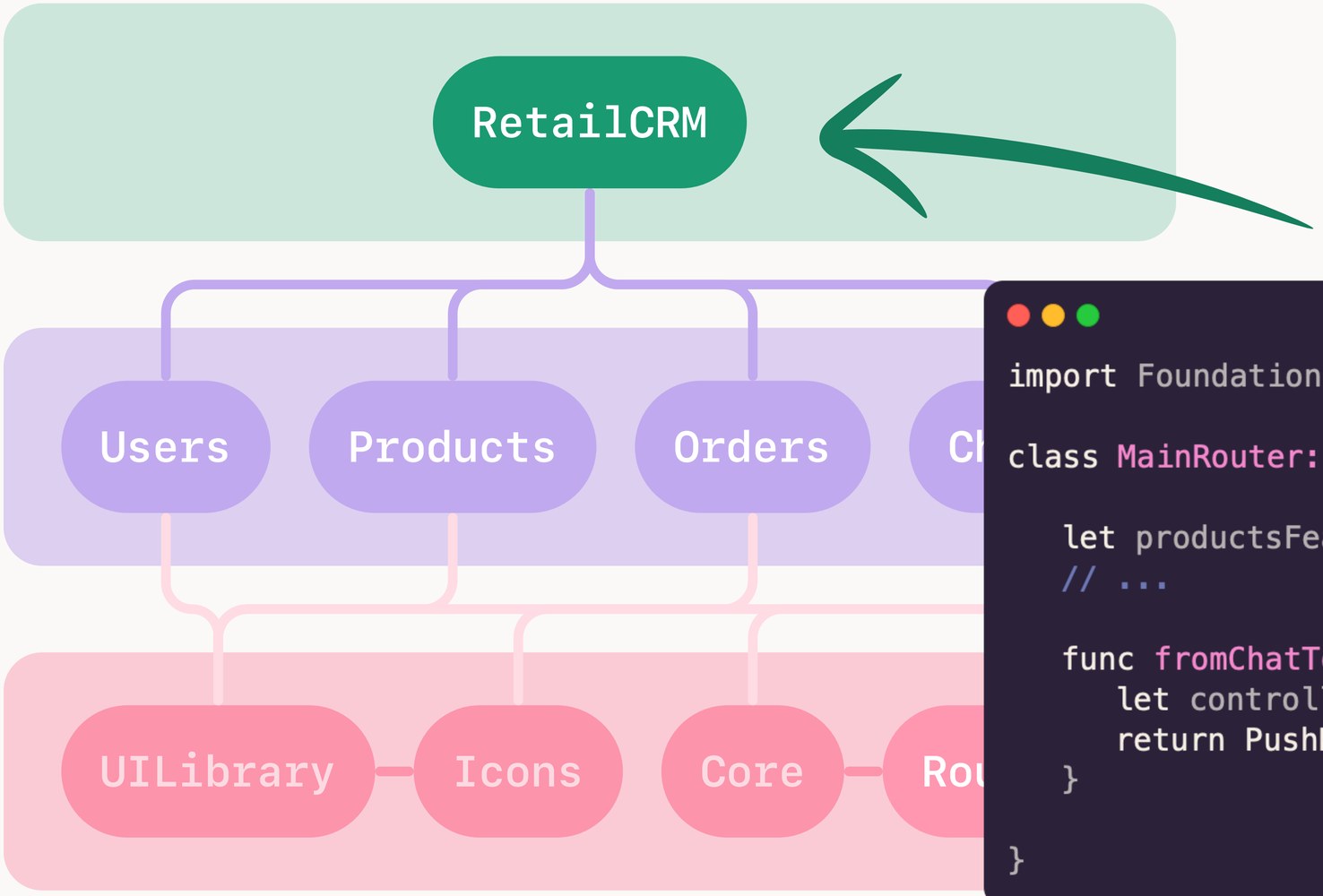
Роутинг




```swift
import Foundation

public protocol ChatsRouter: Router {

    func fromChatToProduct(id: Int) -> Route

}
```

```swift
import Foundation

public protocol Router {

    // ...

}
```

Роутинг




```swift
import Foundation

class MainRouter: ChatsRouter, ProductsRouter, ... {

   let productsFeature: ProductsFeature
   // ...

   func fromChatToProduct(id: Int) -> Route {
      let controller = productsFeature.product(id: id)
      return PushRoute(to: controller)
   }

}
```

Как быть с часто-используемыми экранами?

1. Сделать отдельную фичу
2. Фича-солянка Service
3. Сделать Library компонент

Обзор инструментария

Знакомьтесь,

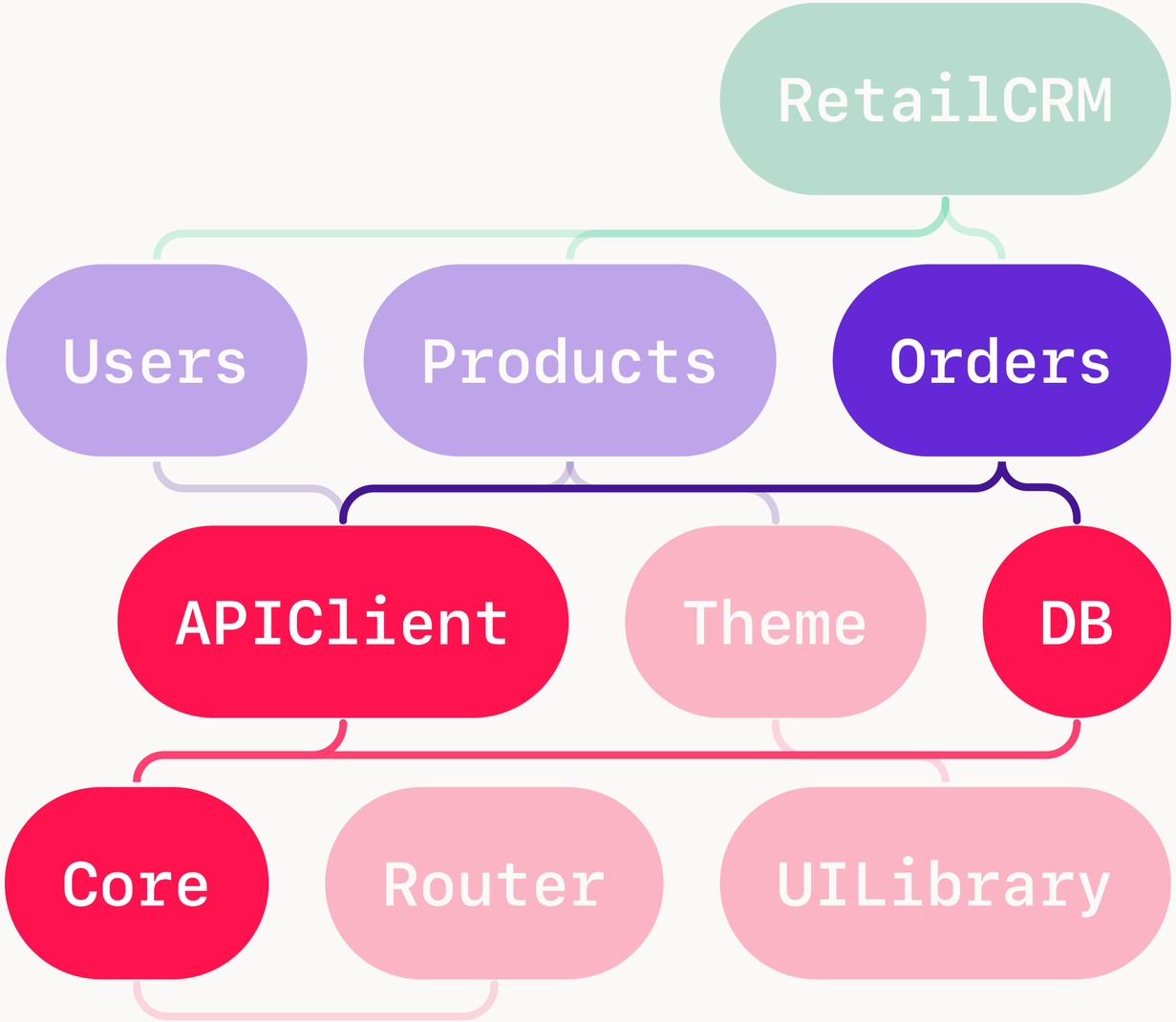
Tuist


```
Workspace.swift

App / RetailCRM / Project.swift
App / POS / Project.swift

Feature / Chats / Project.swift
Feature / Products / Project.swift

Library / Core / Project.swift
Library / APIClient / Project.swift

Tuist / Dependencies.swift
Tuist / Helpers / *.swift
```

Управление проектом в Tuist

```
import ProjectDescription
import ProjectDescriptionHelpers

let project = Project(
    name: .Notifications,
    targets: [
        Target(
            name: .NotificationsFeature,
            sources: .defaultSourcesPath,
            resources: .defaultResourcesPath,
            dependencies: [
                .target(name: .UILibrary),
                .target(name: .APIClient),
                .target(name: .Router),
                .target(name: .ActionSheet),
                .target(name: .PreferencesManager),
                .target(name: .NotificationsRepository),
                .target(name: .TasksRepository),
                .external(name: .SnapKit),
                .external(name: .Ashton)
            ]
        )
    ]
)
```

Tuist / Focus


RetailCRM
Users
Products
Orders
APIClient
Theme
DB
Core
Router
UILibrary

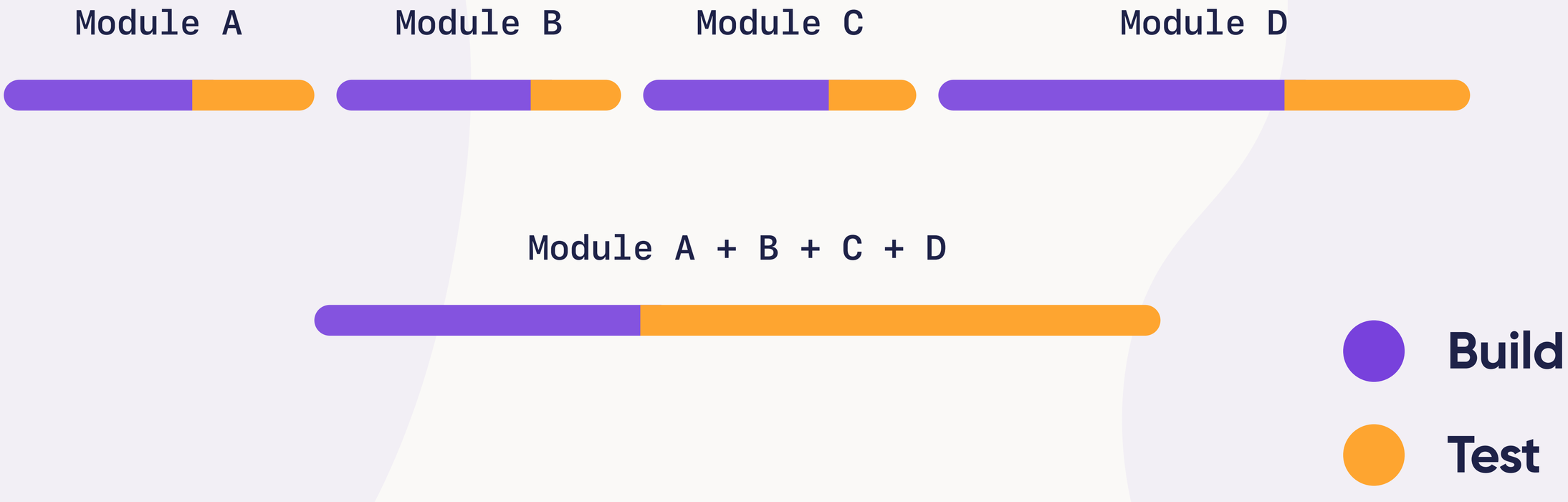
Tuist vs XcodeGen

Для DI используем Resolver

```
public class MyFeature {

    private let resolver: Resolver

    public init(resolver: Resolver) {
        self.resolver = Resolver(parent: resolver)
        self.registerDependencies()
    }

    private func registerDependencies() {
        self.resolver.register(Logger.self, {
            // ...
        })
    }

    public func product(id: Int) -> UIViewController {
        let logger = resolver.resolve(Logger.self)
        let database = resolver.resolve(DataBase.self)
        let viewModel = ProductViewModel(logger, database)
        return ProductViewController(viewModel: viewModel)
    }

}
```

Тестирование

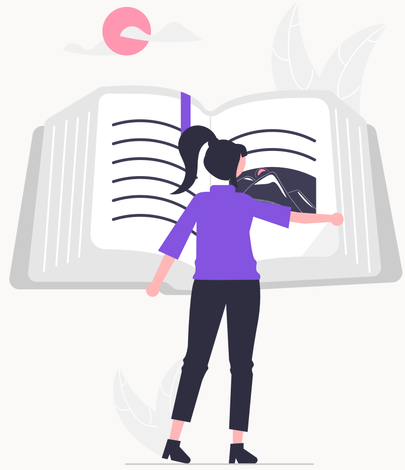
По возможности можно объединять тесты



Не забываем покурить матчасть

Полезные ссылочки

- Building µFeatures
- Framework Oriented Programming
- A Journey into frameworks and Swift
- Leveraging frameworks to speed up our development on iOS - Part 1
- Library Oriented Programming
- Building Modern Frameworks



Спасибо за внимание

linkedin.com/in/ilya-kharlamov

@ilya_kharlamov

github.com/ilia3546

with ❤ from simla.com