Архитектура, и как её готовить

Вихляев Сергей, Бубнов Иван

Портфолио

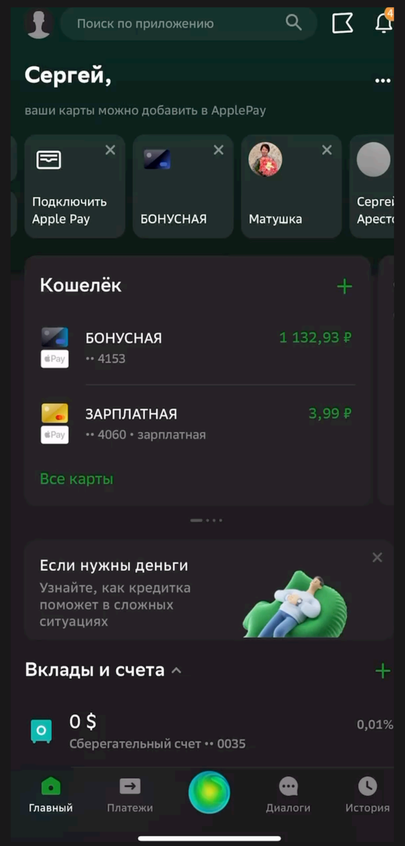
- Вихляев Сергей
- 5 лет в iOS-разработке
- Lead iOS в команде Потребительского кредитования **СберБанк Онлайн**
- Основатель сообщества iOSnick Community в Telegram
- Проекты, в которых участвовал:

План доклада

- Вступление
- Значимость архитектурного проектирования
- Цели доклада
- В чем отличие архитектур?
- Тезис об архитекрутах
- Порядок решения архитектурных задач
- Практика - пример адаптации
- Вывод по докладу

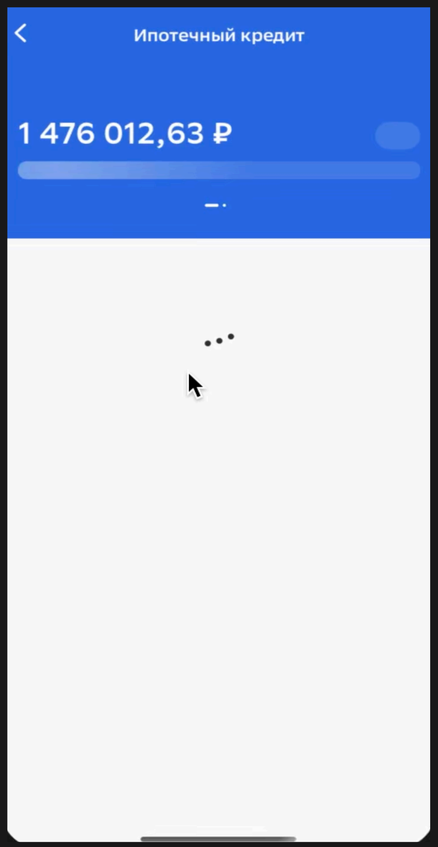
Проектирование в приложении СберБанк Онлайн

Примеры:



- Предварительный расчет кредита с конструктором страховок

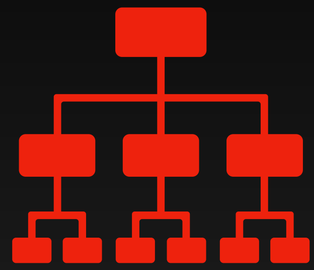
- Редизайн детальной информации по кредиту



Каждая задача так или иначе связана с архитектурой

Цели доклада

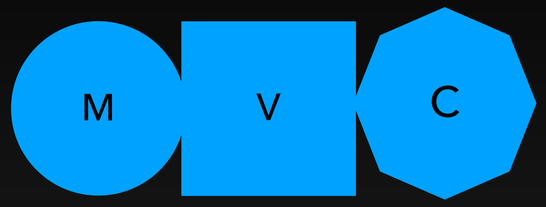
1. Упразднить сложность в вопросах выбора архитектур

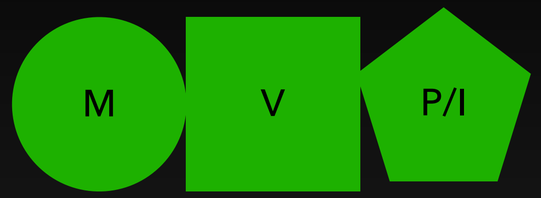


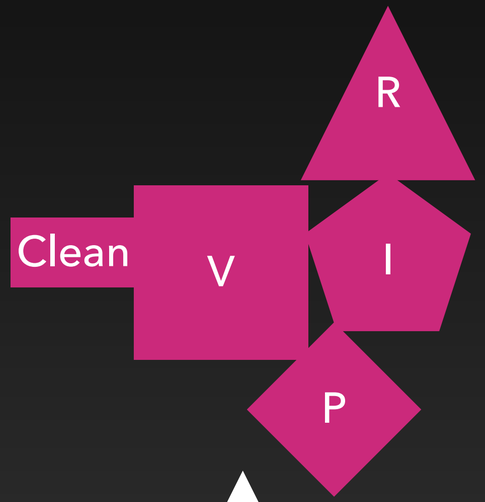
2. Предложить **свой flow** решения архитектурных задач

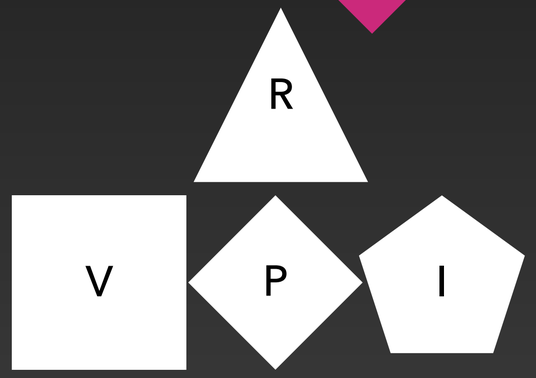
3. Продемонстрировать **flow** на примере

В чем отличие?









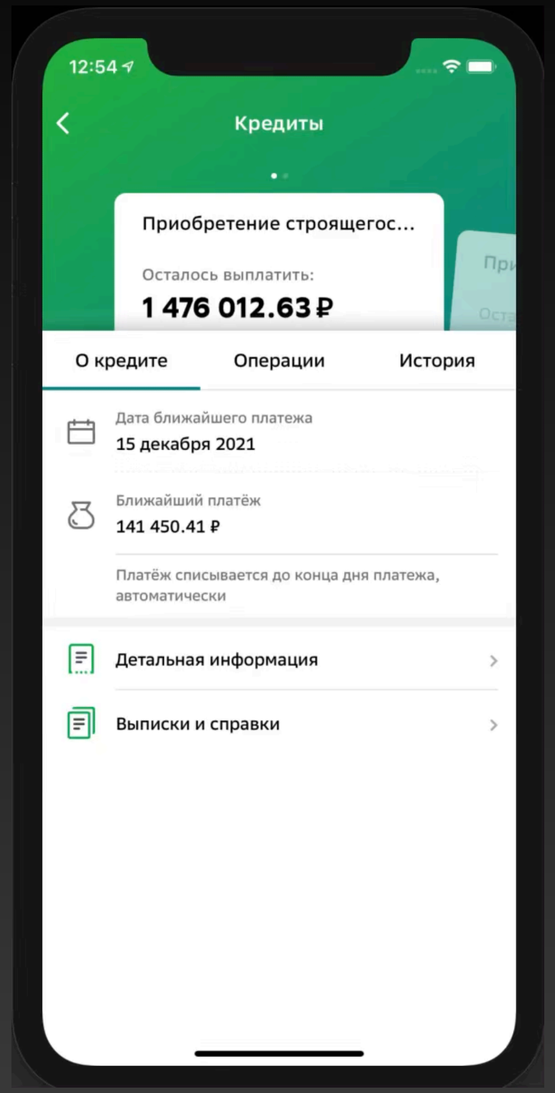
- вариации MVX-архитектуры, где X – обоснованная компоновка более низкоуровневых слоёв, чем VIEW.

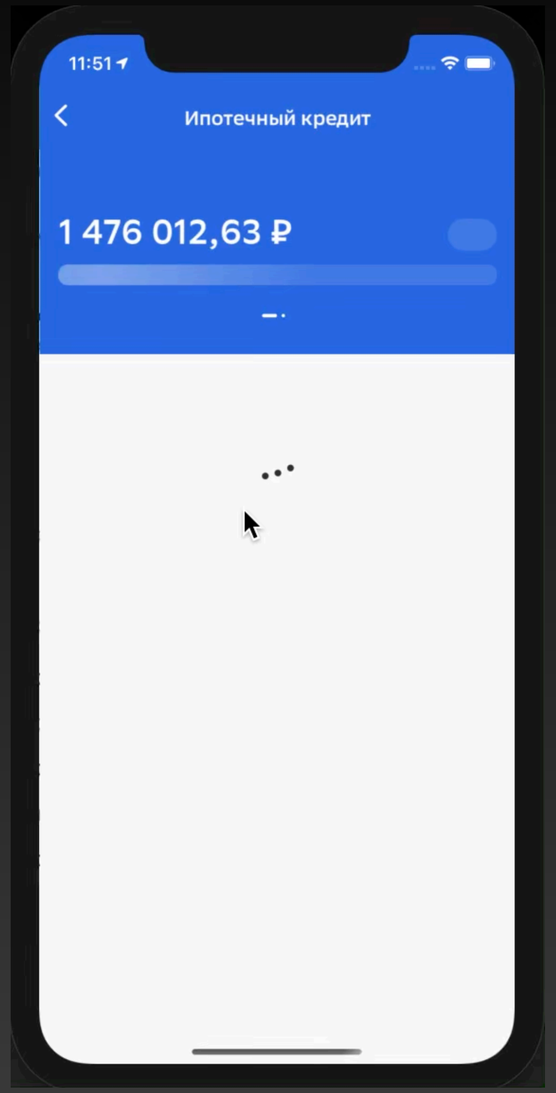
Архитектура - это всего лишь рецепт.
Гораздо важнее - как Вы готовите.

Порядок решения архитектурных задач

- Постановка задачи
- Определение проблематики
- Определение технических аспектов
- Реализация
  - Адаптация наиболее подходящего решения
    - Выбор наиболее подходящей архитектуры
    - Определение конфликтных областей
  - Разработка
- Определение результата

Задача

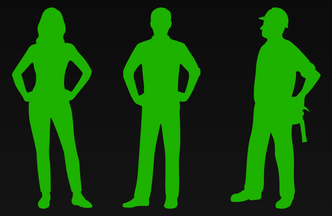

12:54
Кредиты
Приобретение строящегос...
Осталось выплатить:
1 476 012.63 ₽
О кредите
Операции
История
Дата ближайшего платежа
15 декабря 2021
Ближайший платёж
141 450.41 ₽
Платёж списывается до конца дня платежа, автоматически
Детальная информация
Выписки и справки

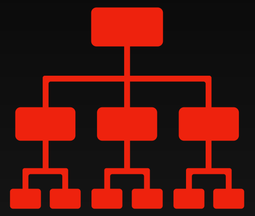

11:51
Ипотечный кредит
1 476 012,63 ₽

Порядок решения архитектурных задач

- **Постановка задачи**
- **Определение проблематики**
- Определение технических аспектов
- Реализация
  - Адаптация наиболее подходящего решения
    - Выбор наиболее подходящей архитектуры
    - Определение конфликтных областей
  - Разработка
- Определение результата

Проблемы





Продуктовые:

- Несколько продуктов, команд…
- Разная кастомизация для каждой команды
- Независимость разработки

Технические:

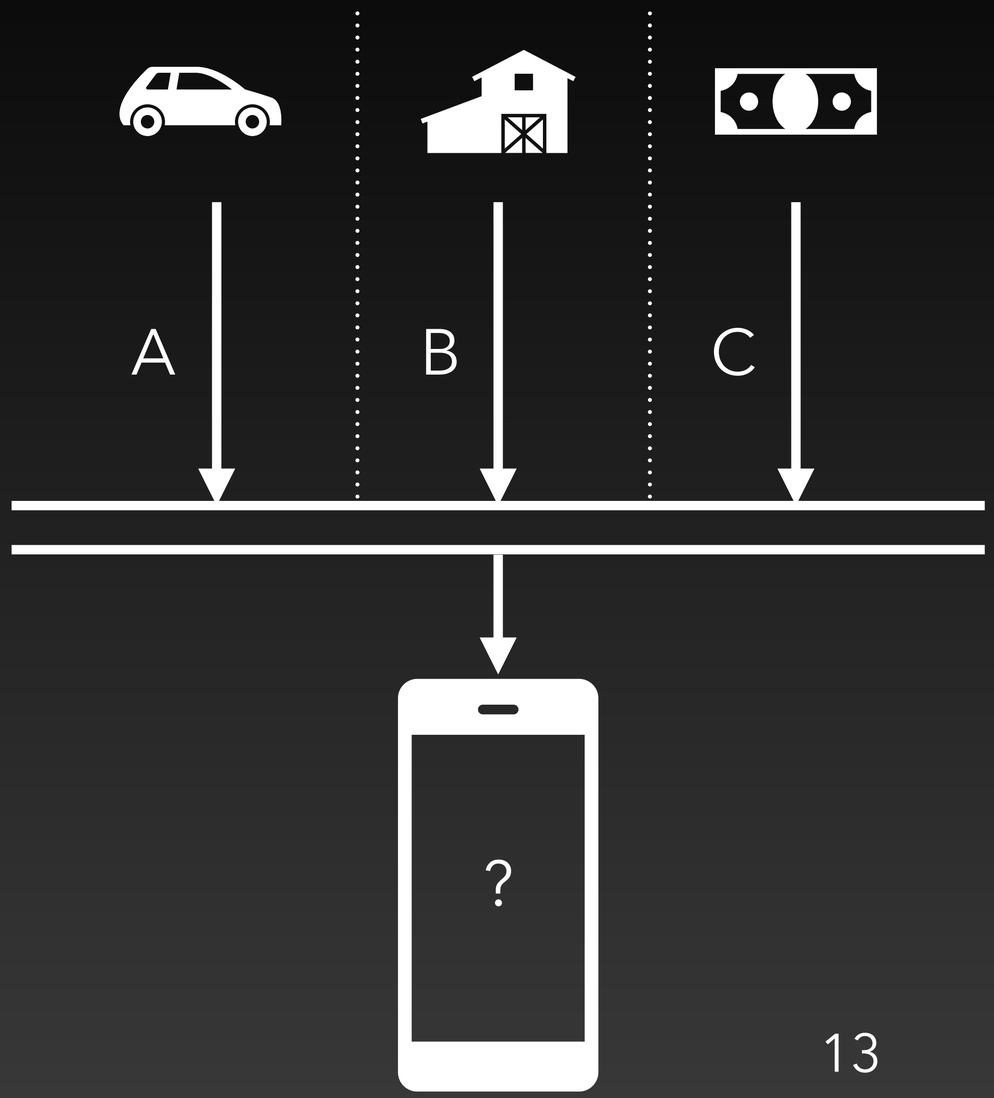
- Продукты из разных Target-ов
- Стек (не содержит react)
- Frameworks
- Обратная совместимость

Порядок решения архитектурных задач

- **Постановка задачи**
- **Определение проблематики**
- **Определение технических аспектов**
- Реализация
  - Адаптация наиболее подходящего решения
    - Выбор наиболее подходящей архитектуры
    - Определение конфликтных областей
  - Разработка
- Определение результата

Технические аспекты

- Несколько продуктов = Расширяемость
- Независимость разработки = Масштабируемость
- Единая реализация общих компонентов = Унификация
- Опциональная реализация уникальных компонентов = Кастомизация
- Актуальность технического стека

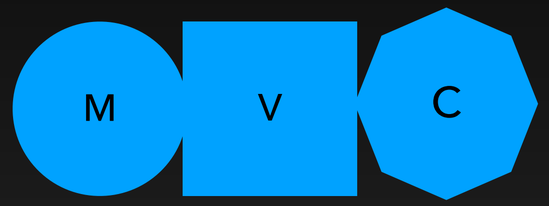


Порядок решения архитектурных задач

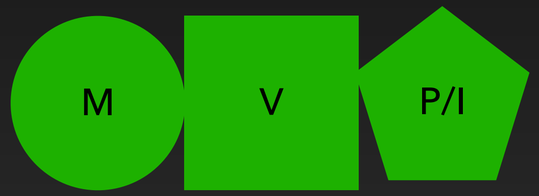
- **Постановка задачи**
- **Определение проблематики**
- **Определение технических аспектов**
- **Реализация**
  - **Адаптация наиболее подходящего решения**
    - **Выбор наиболее подходящей архитектуры**
    - Определение конфликтных областей
  - Разработка
- Определение результата

Адаптация - процесс изменения готового решения под требования конечной реализации

Наиболее подходящая архитектура



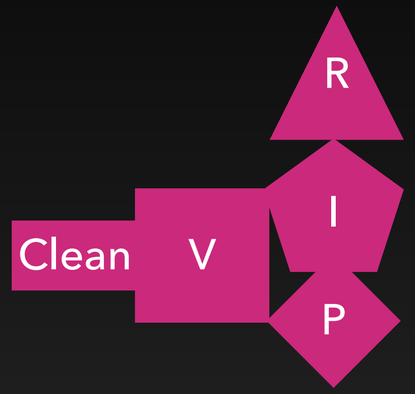
❌ Нет ярко выраженных слоёв

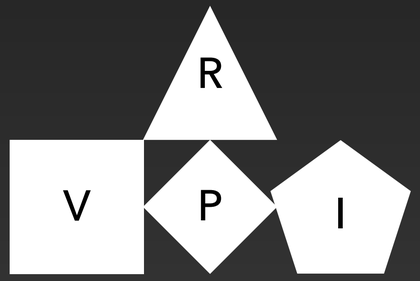


❌ Риск перегрузить слой бизнес-логики

❌ Без react неудобно

Наиболее подходящая архитектура

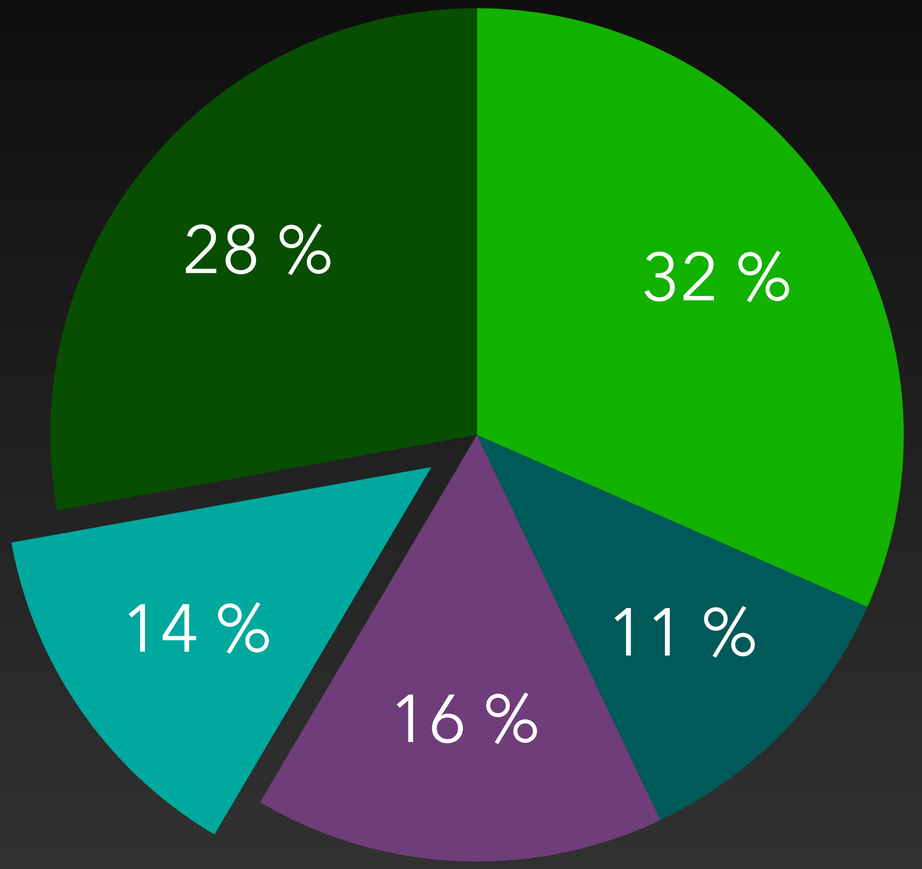




- ✅ Слои ярко выражены
- ✅ Бизнес-логика в отдельном слое

- ✅ Нет замкнутости
- ✅ Личное предпочтение команды

Предпочтения сообщества iOS-разработчиков



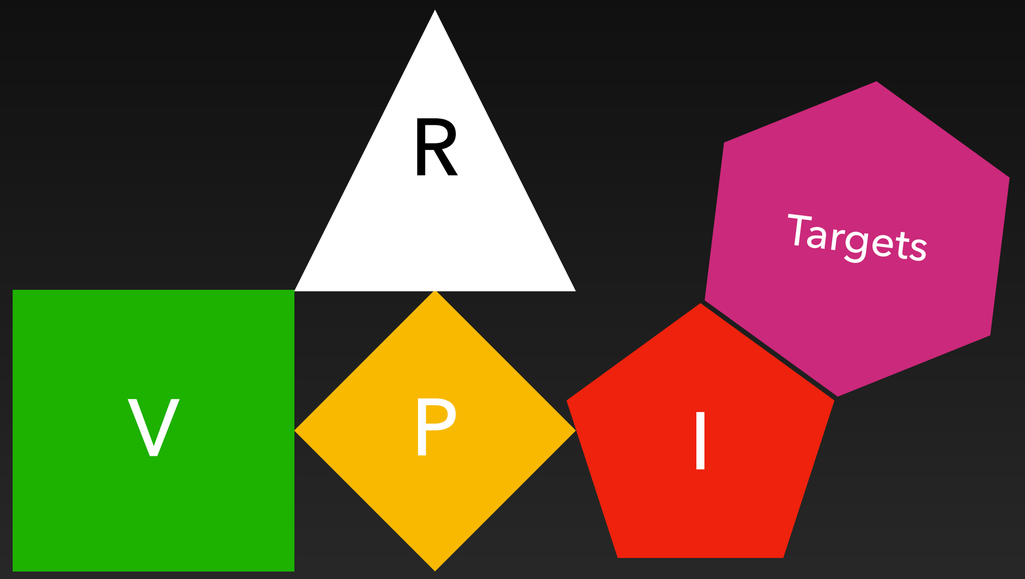
Порядок решения архитектурных задач

- **Постановка задачи**
- **Определение проблематики**
- **Определение технических аспектов**
- **Реализация**
  - **Адаптация наиболее подходящего решения**
    - **Выбор наиболее подходящей архитектуры**
    - **Определение конфликтных областей**
  - Разработка
- Определение результата

Конфликтная область - часть реализации, в которой, в силу специфики выполняемой задачи, подразумеваются изменения.

Конфликтные области

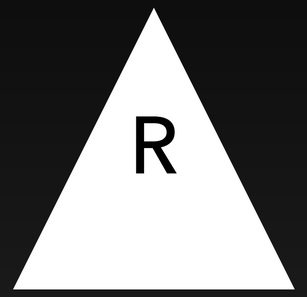
- Несколько продуктов и команд = несколько бизнес-логик
- Продукты из других Target-ов = фасады передачи данных
- Своя DS = кастомная обработка событий UI
- Кастомизация отображения = унификация работы View-слоя



Порядок решения архитектурных задач

- **Постановка задачи**
- **Определение проблематики**
- **Определение технических аспектов**
- **Реализация**
  - **Адаптация наиболее подходящего решения**
    - **Выбор наиболее подходящей архитектуры**
    - **Определение конфликтных областей**
  - **Разработка**
- Определение результата

VIPER для нашей задачи



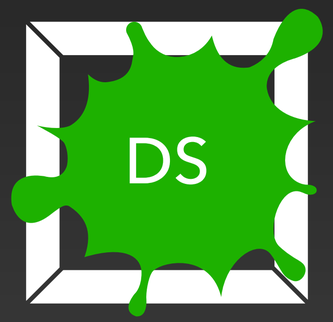
Реализация - View

```
/// Протокол управления View-слоем
protocol ViewInputProtocol: AnyObject {

 /// Обновить набор ViewModel-ей
 /// - Parameter items: новый набор вью моделей
 /// - Parameter currentLoanIndex: индекс выбранного
 /// в данный момент продукта
 func update(items: [ProductViewModelProtocol],
     and currentLoanIndex: Int)

 /// Обновить конкретную ViewModel
 /// - Parameter item: новая viewModel
 /// - Parameter index: индекс обновлемой ViewModel-и
 func update(item: ProductViewModelProtocol, at index: Int)

}
```

ProductViewModelProtocol

```swift
/// Протокол View Model-и для экрана кредитного продукта
public protocol ProductViewModelProtocol {

 /// View model заголовка кредита
 var headerItem: DesignSystemHeaderItemProtocol { get }

 /// View models содержимого кредита
 /// (доступные функции, операции,... и т.д)
 var contentItems: [DesignSystemTableItemProtocol] { get }
}
```

VIPER для нашей задачи

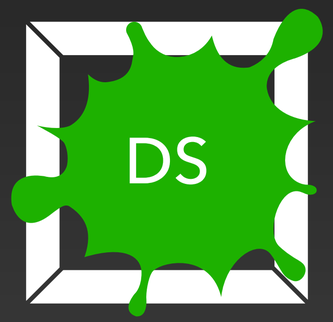


ProductViewModelProtocol

Реализация - Presenter ч.1 (для ViewLayer)

```swift
/// Интерфейс делегирования событий View
protocol ViewDelegate: AnyObject {

 /// Произошла смена страницы (продукта)
 /// - Parameter index: Индекс нового кредита
 func didTurn(toLoanWithIndex index: Int)

 /// Повторная загрузка списка кредитов
 func reloadAllProducts()

 /*... Another events */
}
```

VIPER для нашей задачи

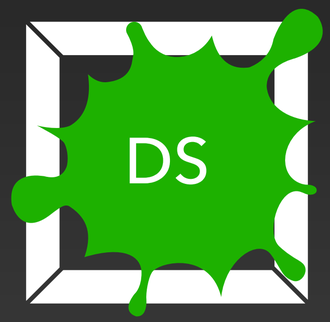


ProductViewModelProtocol

Реализация - Presenter ч.2 (для InteractionLayer)

```
/// Интерфейс взаимодействия с Presenter-ом
/// со стороны Interactor-а
protocol PresenterInputProtocol: AnyObject {

 /// Обновить кредитный продукт по индексу
 /// - Parameters:
 ///    - loan: кредитный продукт
 ///    - index: индекс обновляемого
 ///    кредитного продукта
 func represent(loan: ViewModelConvertable, at index: Int)
}
```

VIPER для нашей задачи

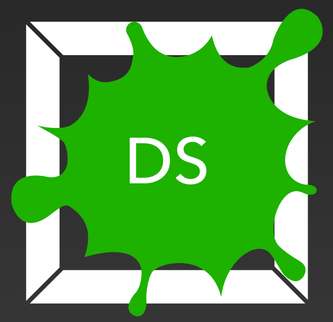


ProductViewModelProtocol

Реализация - Interactor ч.1 (для PresentationLayer)

```
/// Протокол хранения параметров, необходимых Presenter-у
protocol PresenterDataStore {

 /// Возвращает все кредитные продукты в строгом порядке
 /// - Parameter completion: closure по выполнению запроса
 func getAllProducts() -> [ViewModelConvertable]?


 /// Возвращает отображаемый кредитный продукт
 func getCurrentProduct() -> ViewModelConvertable?
}
```

VIPER для нашей задачи

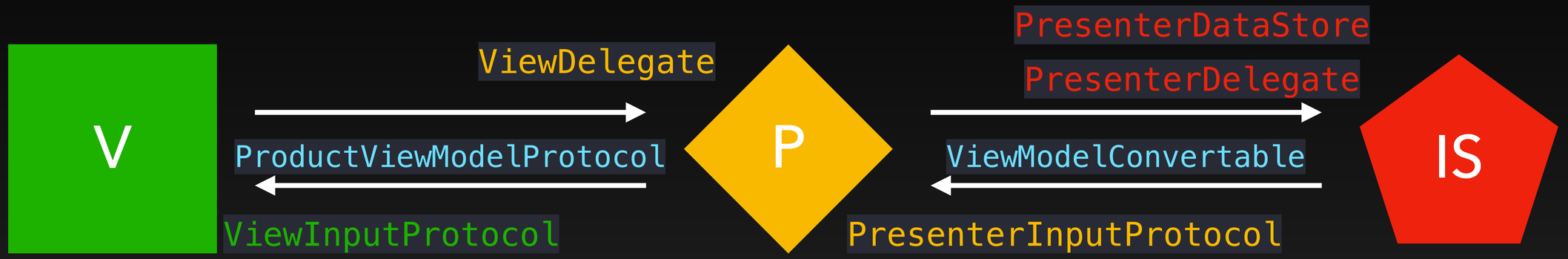


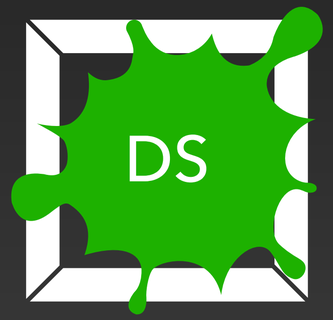
ProductViewModelProtocol

Реализация - Interactor ч.2 (для PresentationLayer)

```swift
/// Протокол делегирования бизнес-событий Presenter-а
protocol PresenterDelegate {

 /// Инициирует выбор для отображения
 /// кредитного продукта
 /// - Parameter index: индекс выбранного
 /// кредитного продукта
 func select(loanWithIndex index: Int)
}
```

VIPER для нашей задачи





ProductViewModelProtocol

Interactor...-ы ?

```
/// Класс объекта, отвечающего за стратегию бизнес-взаимодействия
/// нескольких interactor-ов (сервисов) каждый из
/// которых реализует бизнес-логику управления
/// продуктами конкретного типа
final class InteractionStrategist {

 /// Слабая ссылка на объект PresentationLayer
 weak var presenter: PresentationLayerInputProtocol?

 private let interactors: [ProductBusinessLogicProtocol]

 /// Инициализвтор объекта InteractionStrategist
 /// - Parameter interactors: массив interactor-ов (сервисов)
 /// каждый из которых реализует бизнес-логику управления
 /// продуктами конкретного типа
 init(interactors: [ProductBusinessLogicProtocol]) {
  self.interactors = interactors
 }
}
```

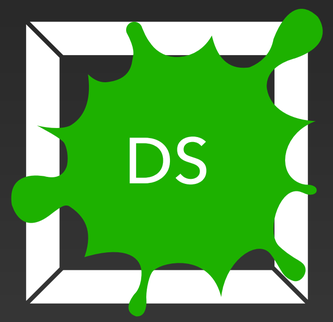
ProductBusinessLogicProtocol

```swift
/// typealias-объединение протоколов EveryProductProtocol & ViewModelConvertable
typealias ProductProtocol = EveryProductProtocol & ViewModelConvertable

/// Общий протокол для каждого interactor-а (сервиса), реализующего
/// бизнес-логику управления конкретным продуктом
protocol ProductBusinessLogicProtocol {

 /// Тип поддерживаемого этим interactor-ом (сервисом) продукта
 var supportingProductType: ProductNamespace.ProductType { get }

 /// Запрос всех продуктов (поддерживаемого типа)
 /// - Parameter completion: closure по выполнению
 func allProducts(completion: @escaping (Result<[ProductProtocol], Error>) -> Void)

 /// Запрос подробной информации по проукту с конкретным id
 /// - Parameters:
 ///   - id: id продукта
 ///   - completion: closure по выполнению
 func productWithDetails(by id: Int,
         completion: @escaping (Result<ProductProtocol, Error>) -> Void)
}
```

VIPER для нашей задачи

EveryProductProtocol &
ViewModelConvertable

ProductBusinessLogicProtocol



ProductViewModelProtocol

DataModel: EveryProductProtocol & ViewModelConvertable

Магия Presenter-a

```
// MARK: - PresenterInputProtocol
extension Presenter: PresenterInputProtocol {

 /*...*/

 func represent(loan: ViewModelConvertable, at index: Int) {
  guard
   let viewModel = loan.getMainViewModel(itemsDelegate: self)
  else { return }

  view?.update(item: viewModel, at: index)
 }
}
```

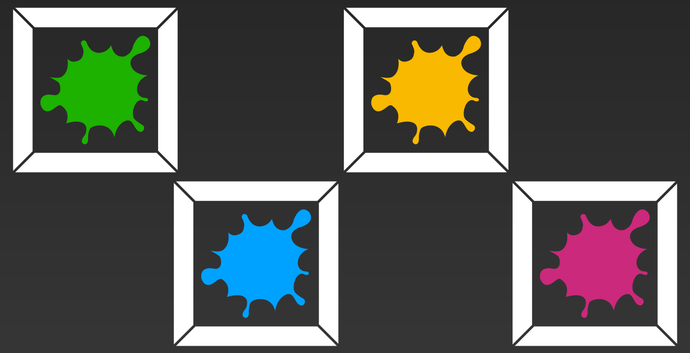
ViewModelConvertable

```
/// Протокол объединения кредитных продуктов на уровне преобразования DataModel-ей
/// Каждый кредитный продукт должен иметь возможность быть представленным
/// в таком виде для отображения в экране детальной информации
protocol ViewModelConvertable {

 /// Возвращает viewModel для главного экрана детальной информации
 /// - Parameter itemsDelegate: delegate для событий
 /// кастомных UI элементов
 func getMainViewModel(itemsDelegate: CustomItemsDelegate?) ->
 ProductViewModelProtocol?

 /// Возвращает viewModel для экрана полной детальной информации
 func getViewModelForDetailsInfo() -> Any?

 /// Возвращает viewModel для шторки "Из чего состоит продукт"
 func getViewModelForMoreAboutProductBS() -> Any?

 /*... Another ViewModels */
}
```

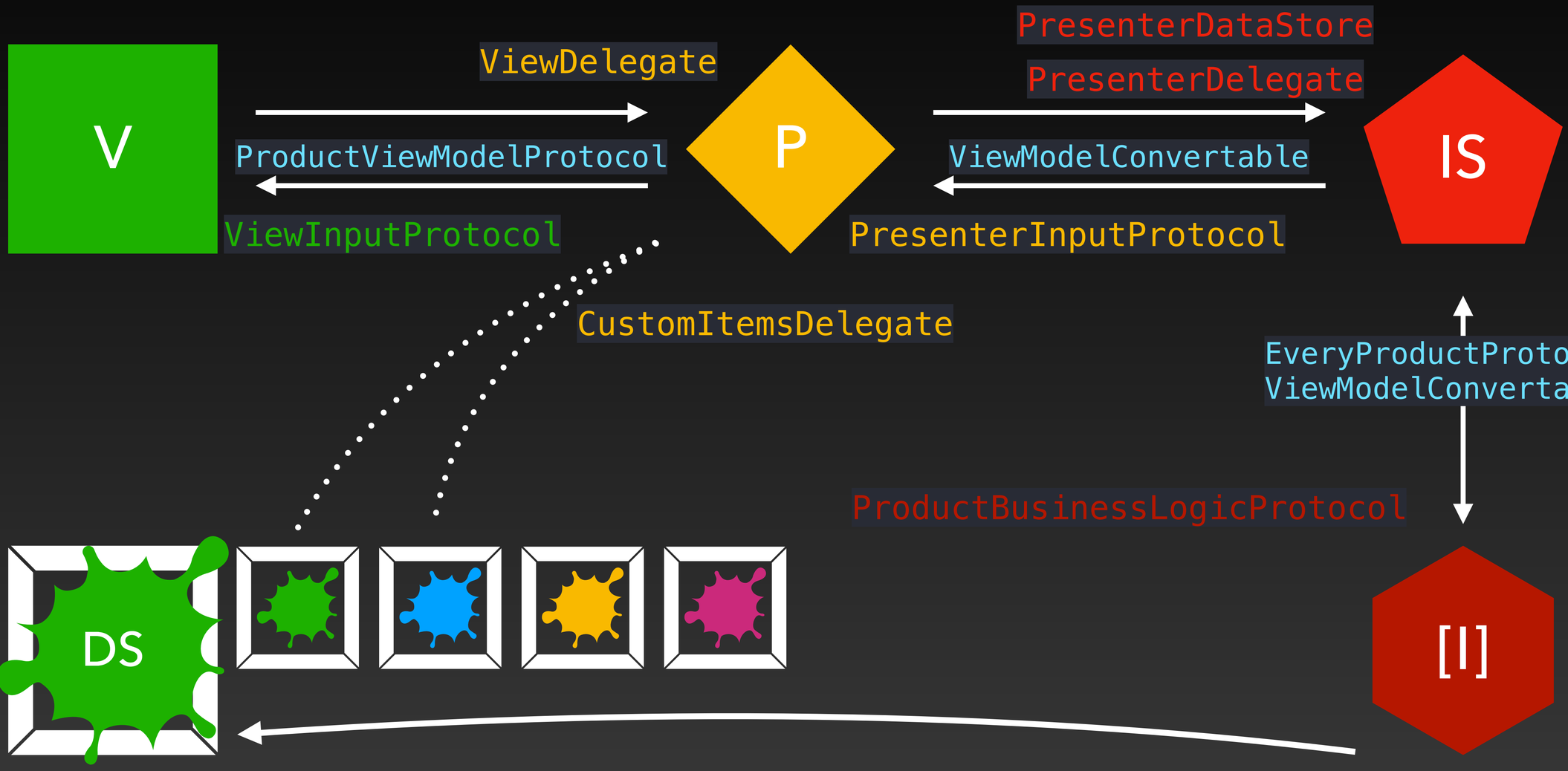
Магия Конвертации DataModel-и

- Все DataModel-и приходят в Presentation-слой как **ViewModelConvertable**
- Mapping каждой DataModel-и в **нужную ViewModel** происходит **вне Presenter-а**
- Presenter реализует протоколы, позволяющие ему **делегировать кастомные события** элементов DS

```
DataModel: ViewModelConvertable {
```



VIPER для нашей задачи



Порядок решения архитектурных задач

- Постановка задачи
- Определение проблематики
- Определение технических аспектов
- Реализация
  - Адаптация наиболее подходящего решения
    - Выбор наиболее подходящей архитектуры
    - Определение конфликтных областей
  - Разработка
- Определение результата

Итог

Было:

- MVC - ManyViewControllers,
- Монолитная View
- ViewModel отсутствует
- Без DS,
- Старый стек,
- Слабая масштабируемость,
- ObjectiveC

Стало:

- ➜ **Изолированные слои**
- ➜ **View строится композиционно**
- ➜ **1 DataModel = несколько ViewModel-ей**
- ➜ **Перешли на DS**
- ➜ **Актуальный стек**
- ➜ **Масштабируемость на любом уровне**
- ➜ **Swift**
- ➜ **Изолированность**
- ➜ **Абстрактность**

Порядок решения архитектурных задач

- Постановка задачи
- Определение проблематики
- Определение технических аспектов
- Реализация
  - Адаптация наиболее подходящего решения
    - Выбор наиболее подходящей архитектуры
    - Определение конфликтных областей
  - Разработка
- Определение результата

Задача выполнена!!! 🎉🎉🎉

Подытог

- ✓ **Вывод нового продукта теперь гораздо проще**
- ✓ **Гибкая кастомизация как для типа продуктов, так и для конкретного продукта**
- ✓ **Каждый новый «Deprecated» влечёт минимальные изменения**

«The only way to go fast is to go well.»

(c) Uncle Bob Martin

Архитектура - это всего лишь рецепт.
Гораздо важнее - как Вы готовите.

Благодарность

ВАМ - ЗА ВНИМАНИЕ

- Сообществу разработчиков СберБанк Онлайн
- Организаторам Mobius
- Сообществам iOSnick Community и SwiftBook
- Бубнову И.А.

Вихляев С.Н.
Tg: @iOSnick

ВОПРОСЫ