
Слышно?
Видно?
Autocue
Autocue
WDR
1

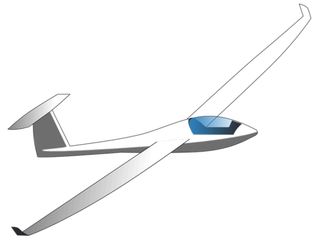
sweatcoin

Легаси: Переписать нельзя поддерживать

Избавление проекта от легаси

Сергей Митрофанов  *G0retZ*  *G0retZ*

Пару слов о себе



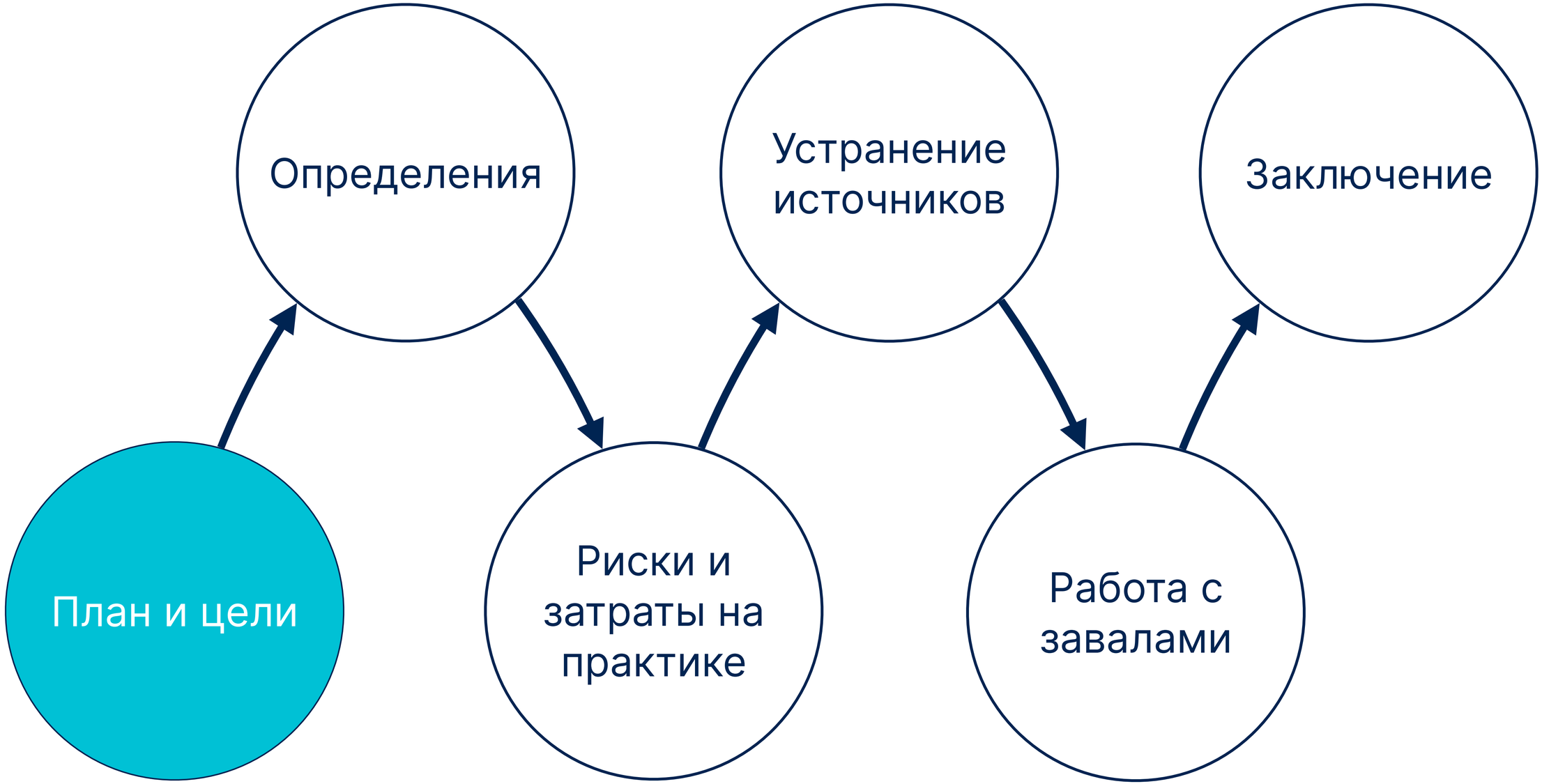
В мобильный разработке более 6 лет. Участвовал в разработке 10+ различных проектов;

Увлеченно отношусь к качеству и эргономике приложения с точки зрения UX;

Увлекаюсь планеризмом, серфингом, MTB.

Sweatcoin — мотивировать людей больше двигаться для улучшения самочувствия, здоровья и экологии.

План


План и цели
Определения
Риски и затраты на практике
Устранение источников
Работа с завалами
Заключение

Цели

После доклада вы:

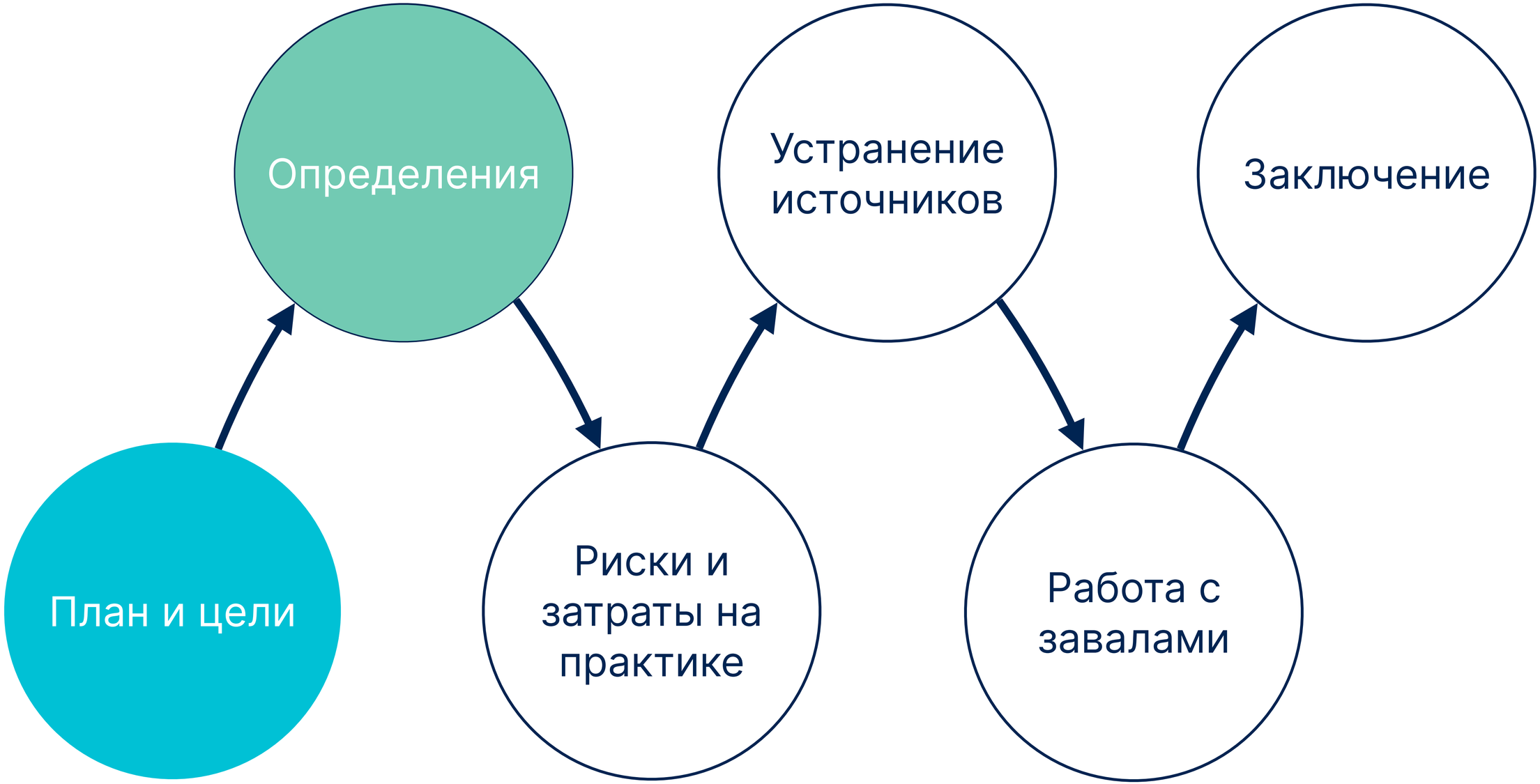
Будете знать о способах работы с легаси

Будете понимать что и когда переписывать

Перестанете бояться легаси кода в проекте

Сможете заметно улучшить код проекта с легаси

План


План и цели
Определения
Риски и затраты на практике
Устранение источников
Работа с завалами
Заключение

Определение (вики)

Легаси(Устаревший код) — это код, который относится к более не поддерживаемой или производимой операционной системе или другим компьютерным технологиям.

Частные определения

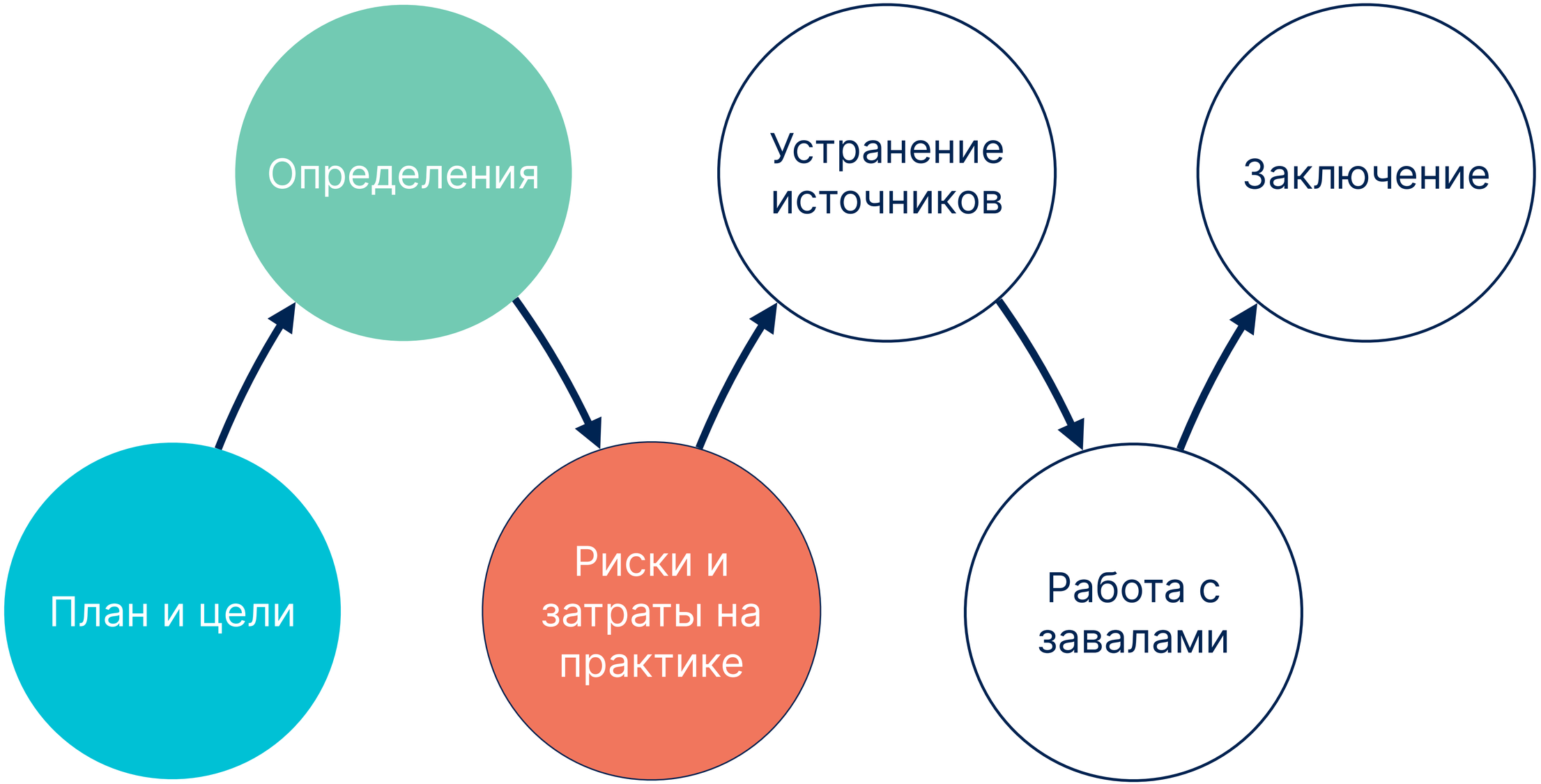
Легаси код — это код, который достался вам по наследству

Легаси — это код, который не покрыт тестами.

Обобщенное определение

Легаси - это код, не соответствующий текущим архитектуре или code style проекта.

План


Определения
Устранение источников
Заключение
План и цели
Риски и затраты на практике
Работа с завалами

История проекта с легаси

Первоначальный план

История проекта с легаси

Спустя какое-то время

Сценарии дальнейшего развития

Поддержка легаси

Сценарии дальнейшего развития

Поддержка легаси

Сценарии дальнейшего развития

R.I.P.


#ВремяИсторий

Сценарии дальнейшего развития

V 2.0

Сценарии дальнейшего развития

V 2.0

Закон Конвея

Любая организация, разрабатывающая системы (в широком смысле), создаст проект, чья структура является копией коммуникационной структуры организации.

Сценарии дальнейшего развития

V 2.0

Сценарии дальнейшего развития

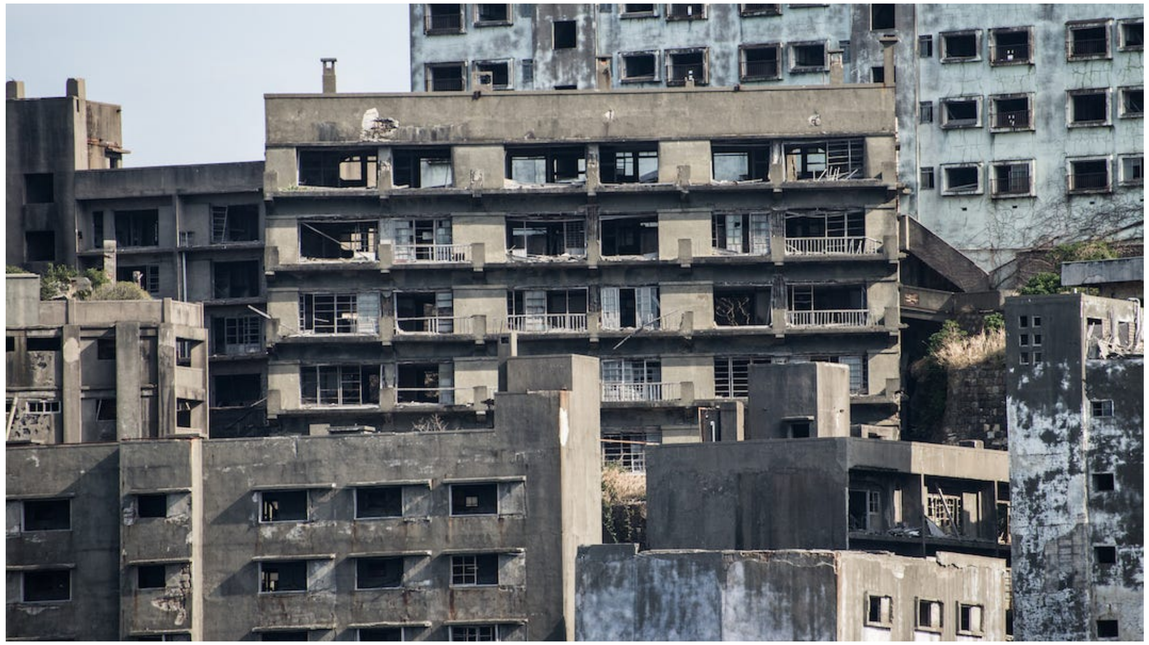
V 2.0


#ВремяИсторий

Сценарии дальнейшего развития

Поддерживать + Переписывать = WIN?

Сценарии дальнейшего развития



Поддерживать + Переписывать = WIN?


#ВремяИсторий

Сценарии дальнейшего развития

Поддерживать + Переписывать = WIN?

Сценарии дальнейшего развития

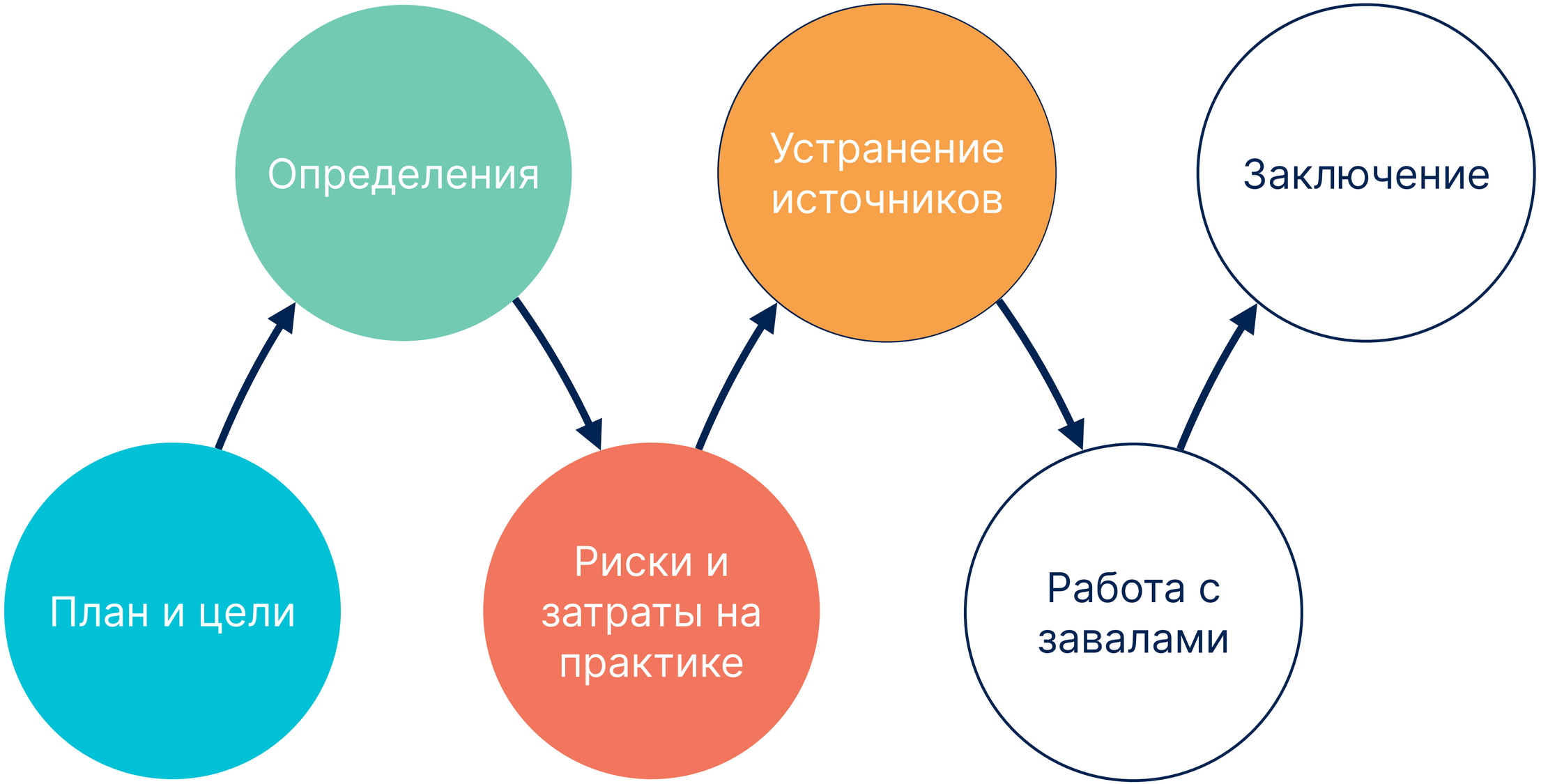
Поддерживать + Переписывать = WIN?


#ВремяИсторий

Избавление проекта от легаси

Избавление проекта от легаси

План


План и цели
Определения
Риски и затраты на практике
Устранение источников
Работа с завалами
Заключение

Остановим течь

Источники легаси

Продуктовые

Социальные

Архитектурные

Технические

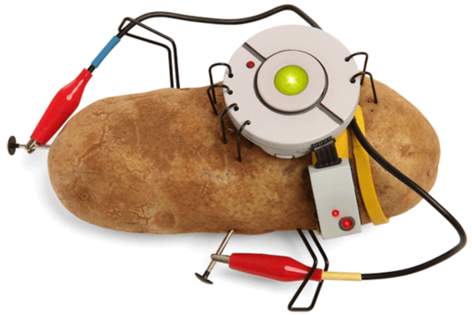
Продуктовые источники

Политика

Новые Дедлайны

Пивоты

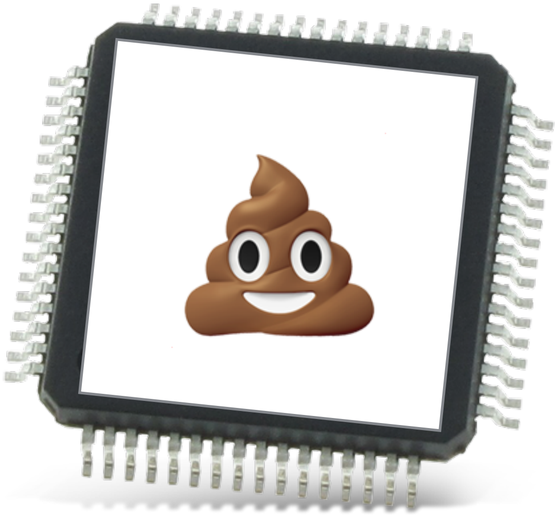
Внешние сервисы



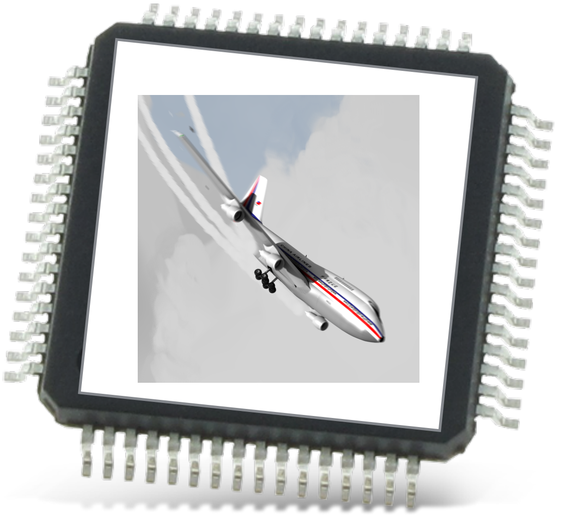
Эксперименты

Продуктовые источники

Политика

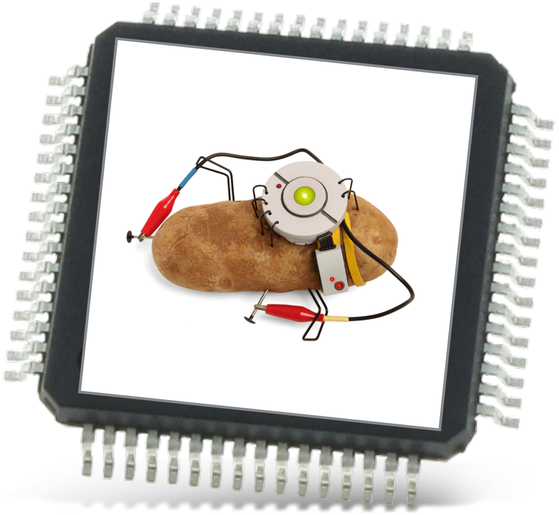


Новые Дедлайны



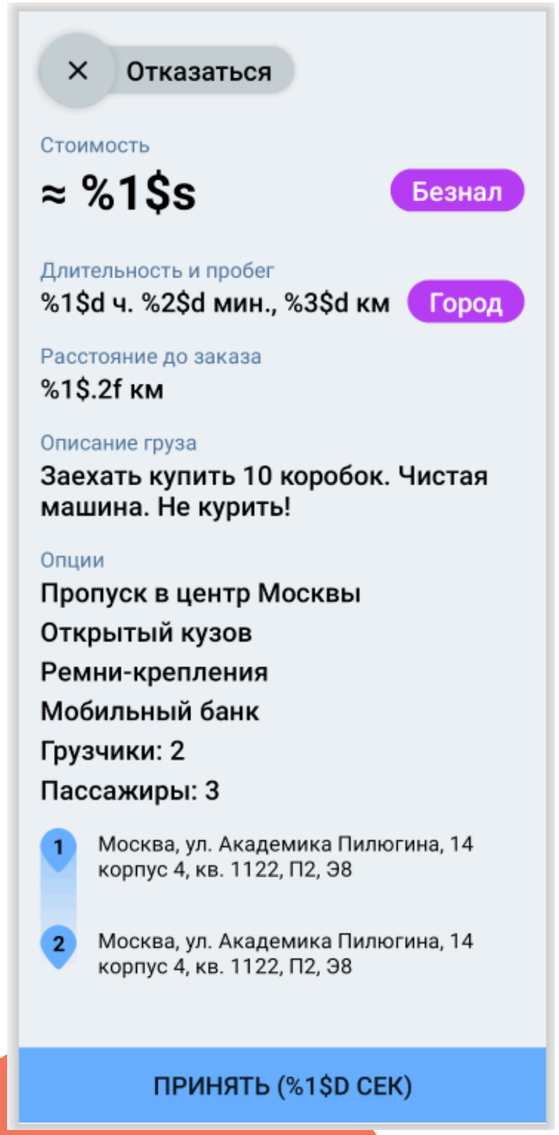
Пивоты

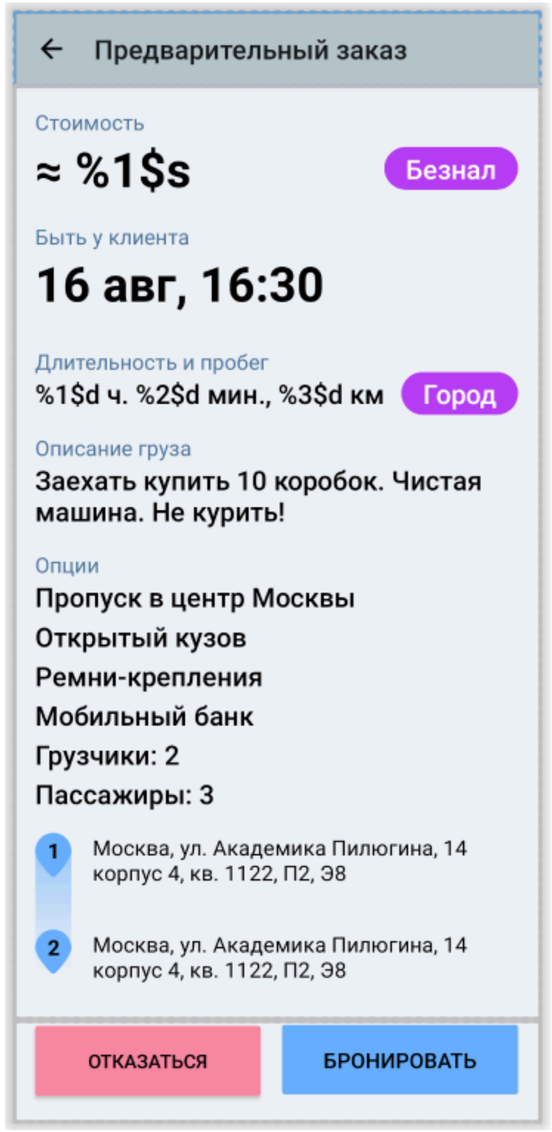
Внешние сервисы

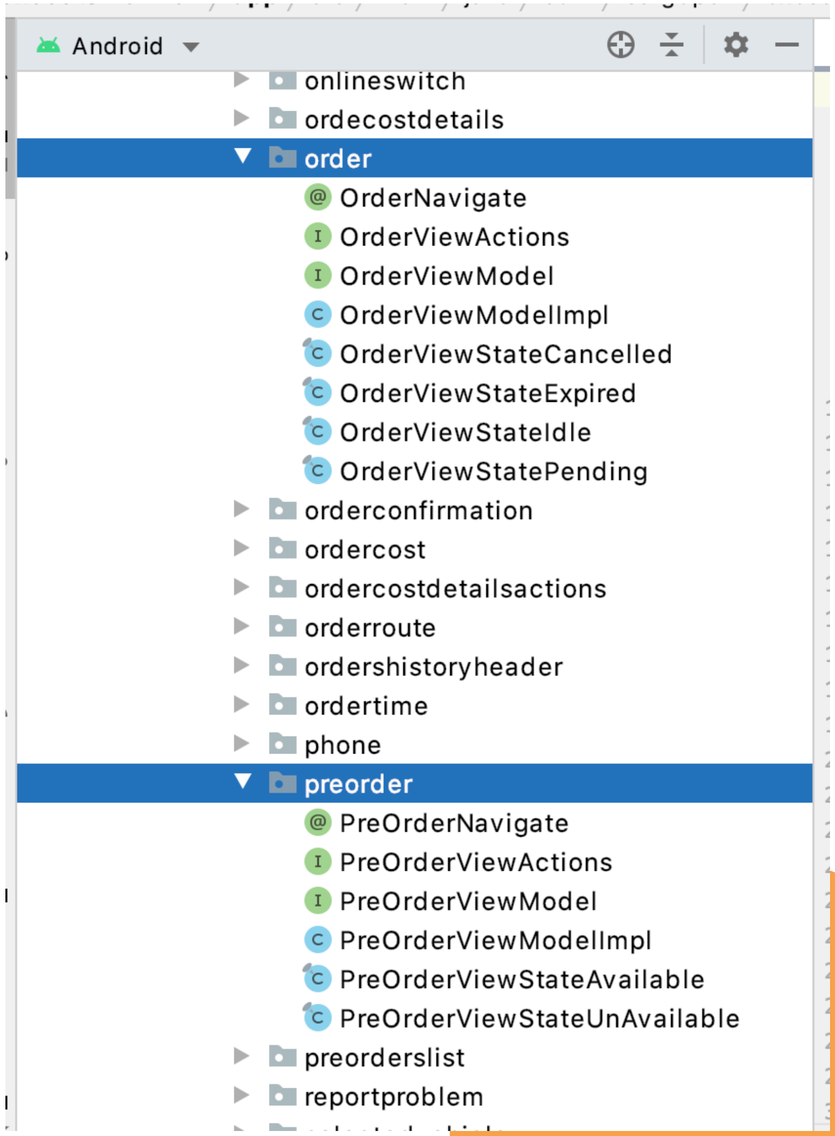


Эксперименты

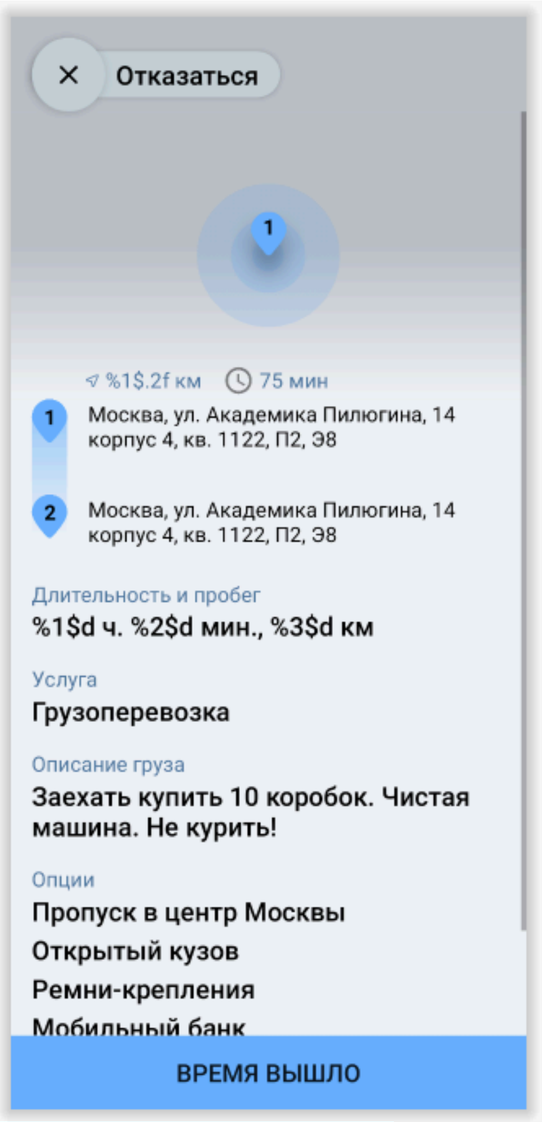
Изоляция (декопуляция)





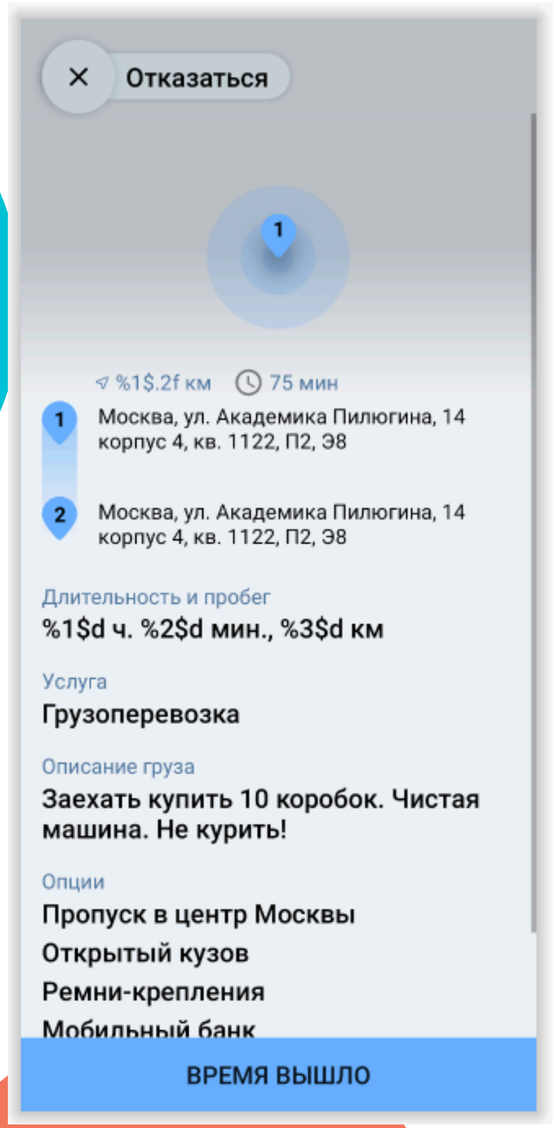


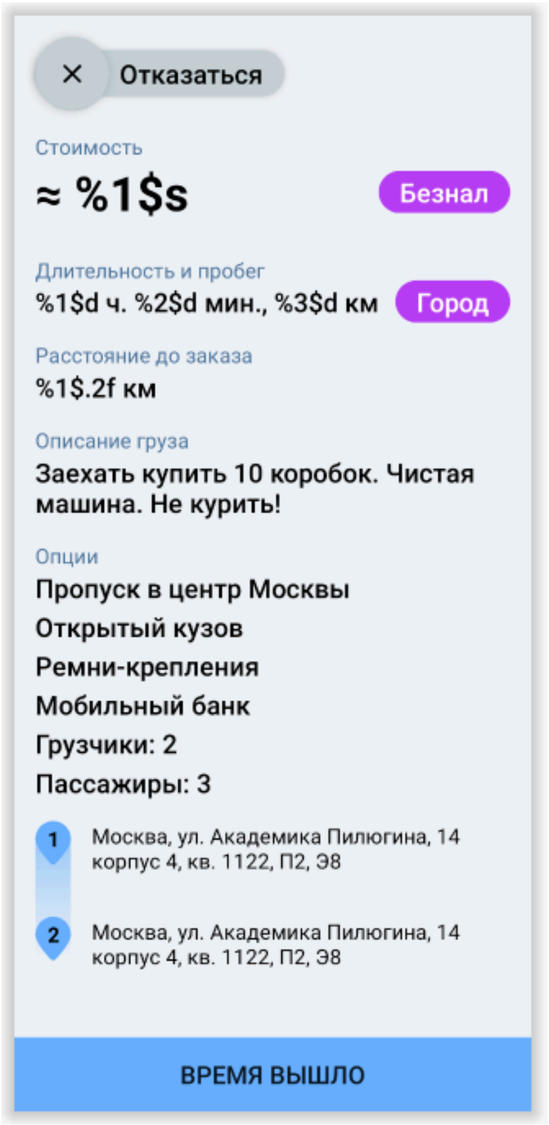
Эксперименты = фичефлаги




```
12          startActivity(
13                  Intent(this, DriverOrderConfirmationActivity::class.java)
14                          .setFlags(
15                                  Intent.FLAG_ACTIVITY_CLEAR_TASK
16                                          or Intent.FLAG_ACTIVITY_NEW_TASK
17                          )
18          )
```

Эксперименты = фичефлаги

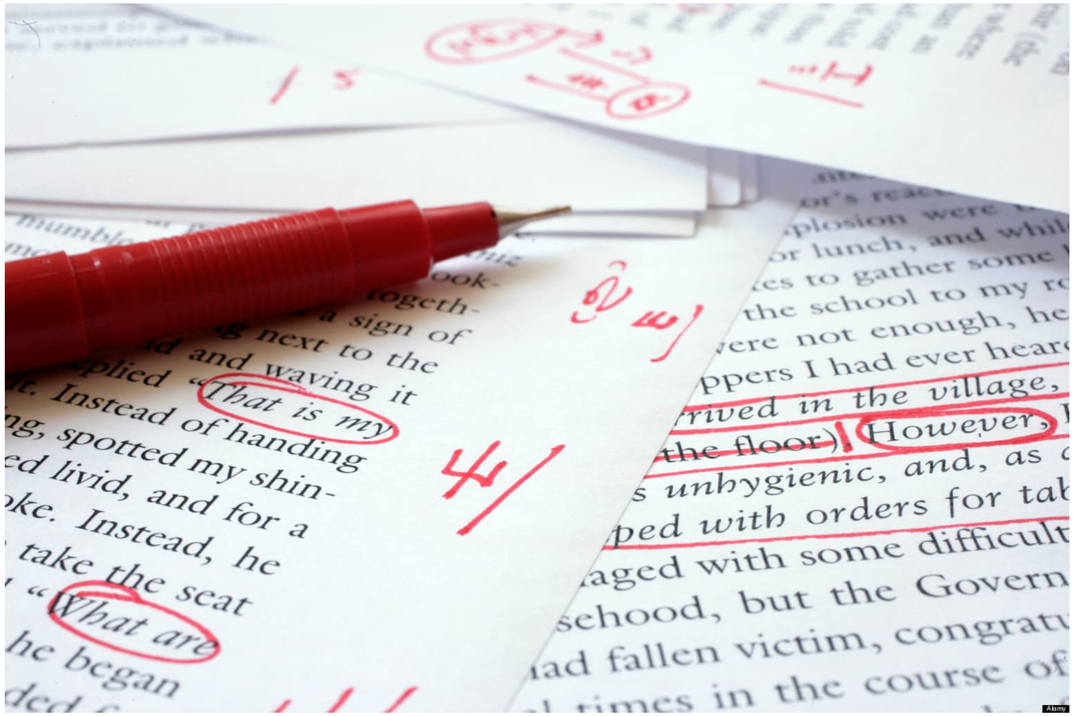





```
12          startActivity(
13                  Intent(this, DriverOrderConfirmationActivity::class.java)
14                          .setFlags(
15                                  Intent.FLAG_ACTIVITY_CLEAR_TASK
16                                          or Intent.FLAG_ACTIVITY_NEW_TASK
17                          )
18          )
```

```
14          if (featureFlags["new_confirmation_screen"] == true) {
15              startActivity(
16                      Intent(this, DriverOrderConfirmationActivityNew::class.java)
17                              .setFlags(
18                                      Intent.FLAG_ACTIVITY_CLEAR_TASK
19                                              or Intent.FLAG_ACTIVITY_NEW_TASK
20                              )
21              )
22          } else {
23              startActivity(
24                      Intent(this, DriverOrderConfirmationActivity::class.java)
25                              .setFlags(
26                                      Intent.FLAG_ACTIVITY_CLEAR_TASK
27                                              or Intent.FLAG_ACTIVITY_NEW_TASK
28                              )
29              )
30          }
```

Социальные источники

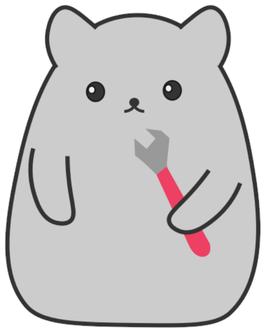


Нерабочий
Code Review

Разобщенность
(Бюрократия)

Социальные источники

Don’t fix it because you should
Fix it because you might be
the only one who can!



Ввести/Починить
Code Review

Одобрение Тимлида
не обязательно

Как запилить Code Review

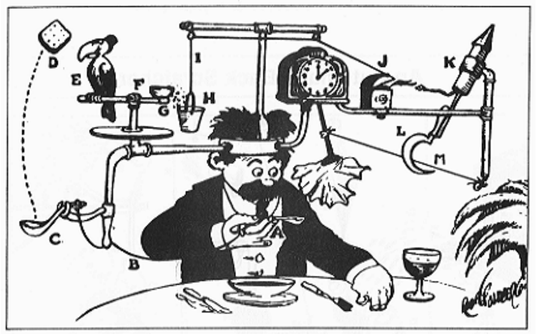
Как запилить Code Review

Code review по-человечески (Хабр)

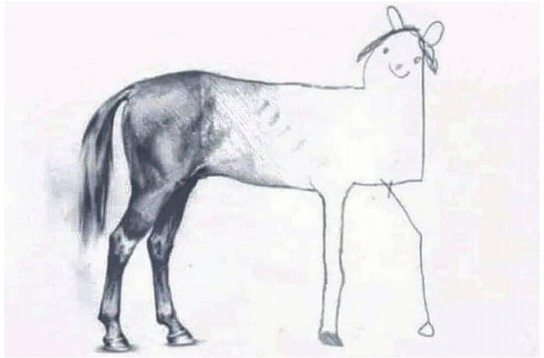
Архитектурные источники

Смена архитектуры

Усложнение задачи



Overengineering

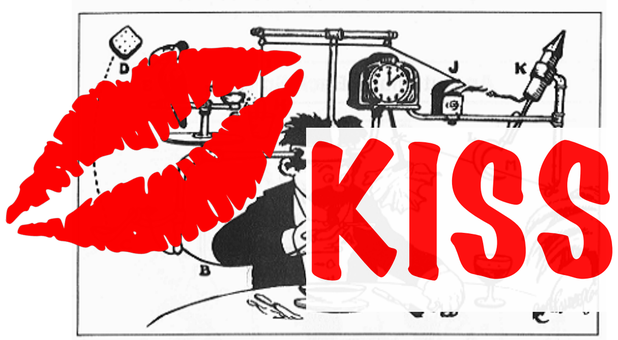


Недоработка

Архитектурные источники

Смена архитектуры

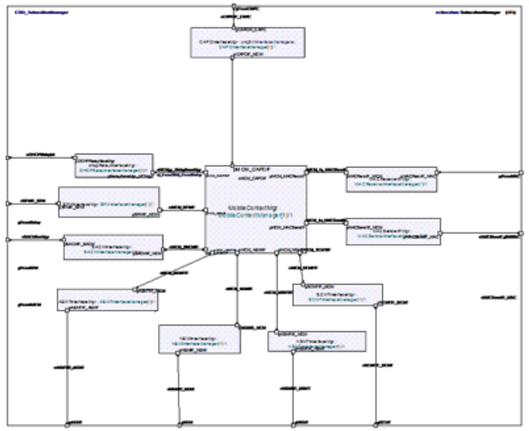
Усложнение задачи



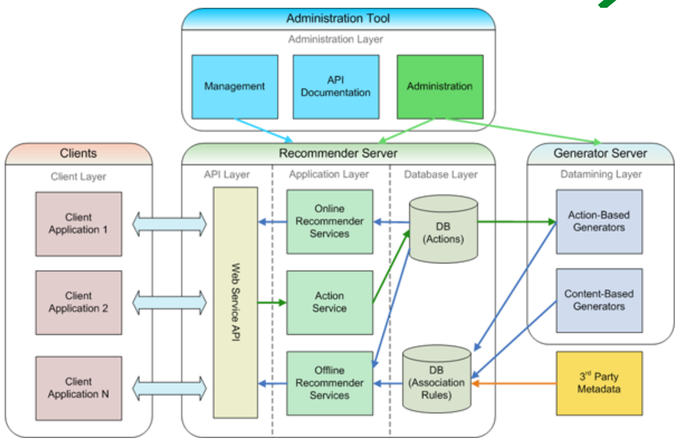
Overengineering

Недоработка

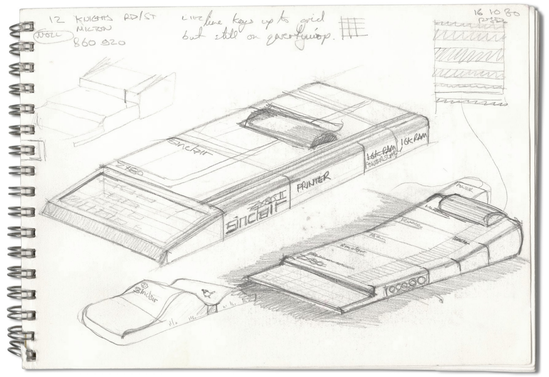
Роль архитектуры

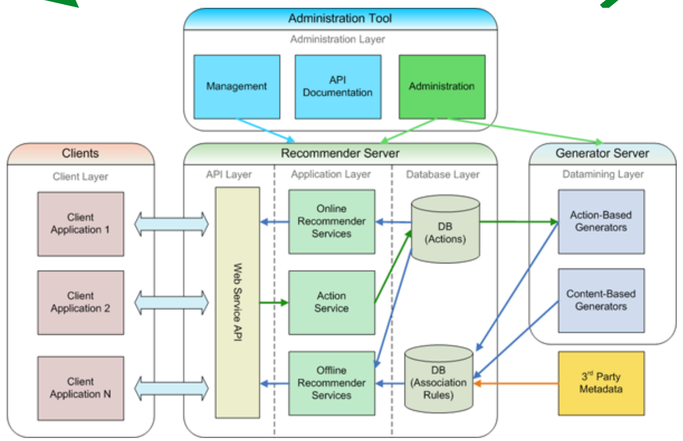


Роль архитектуры

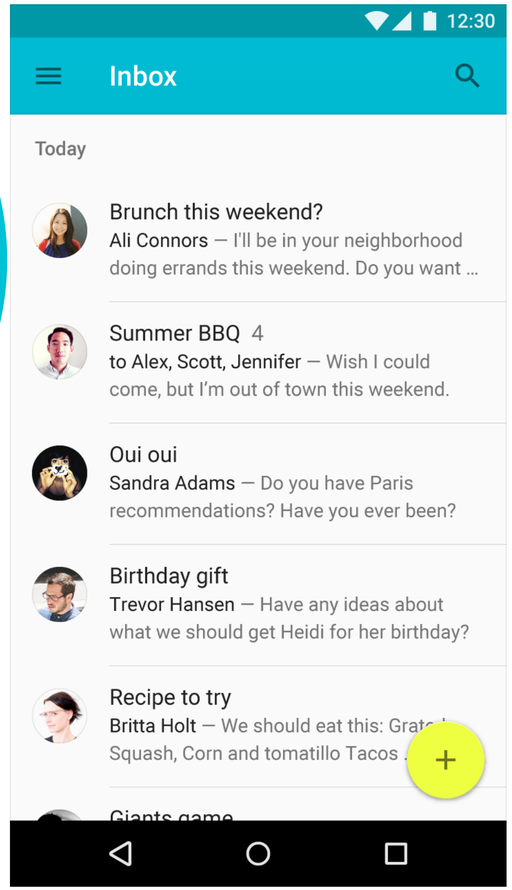


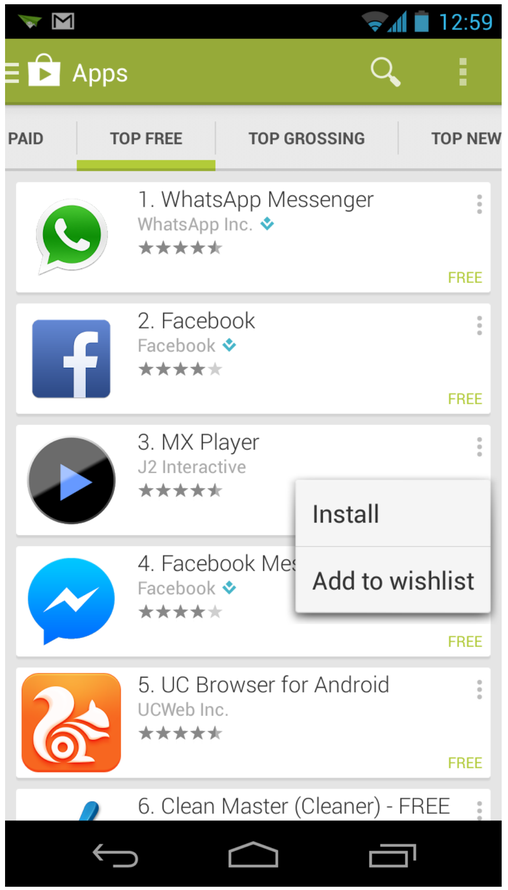
Роль архитектуры

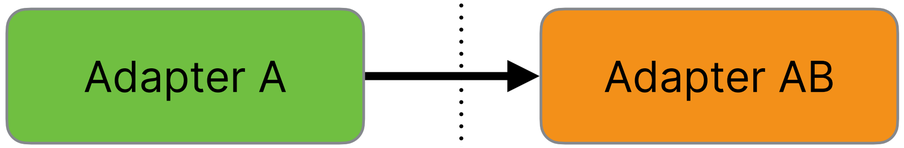




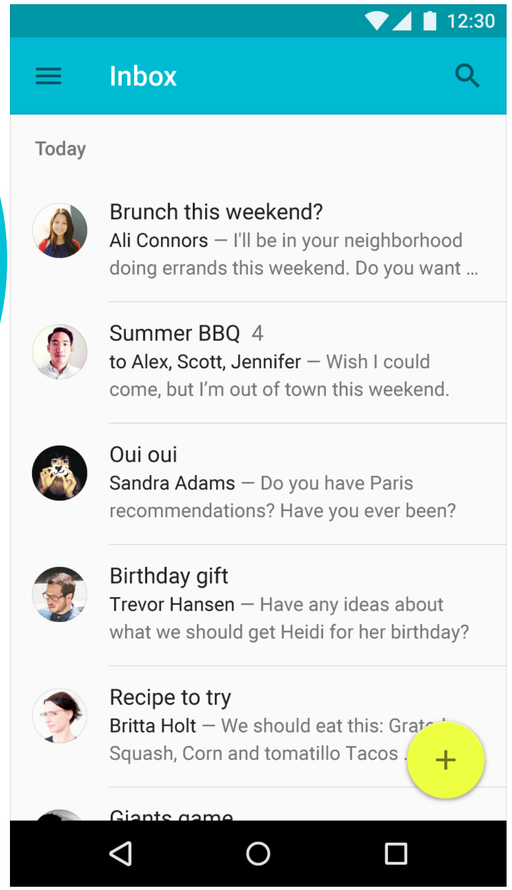
Усложнение задачи

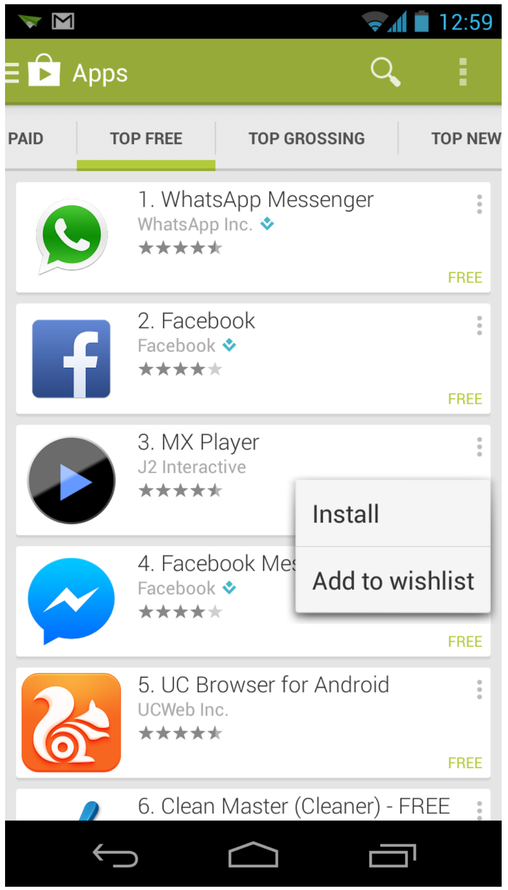


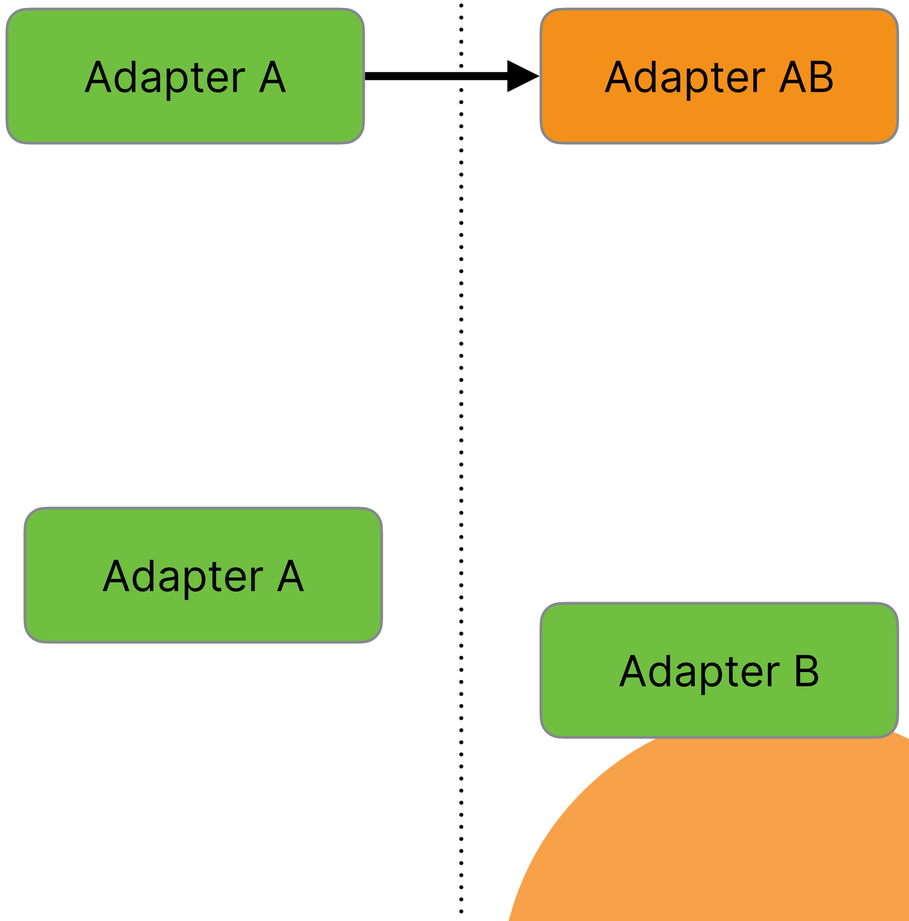




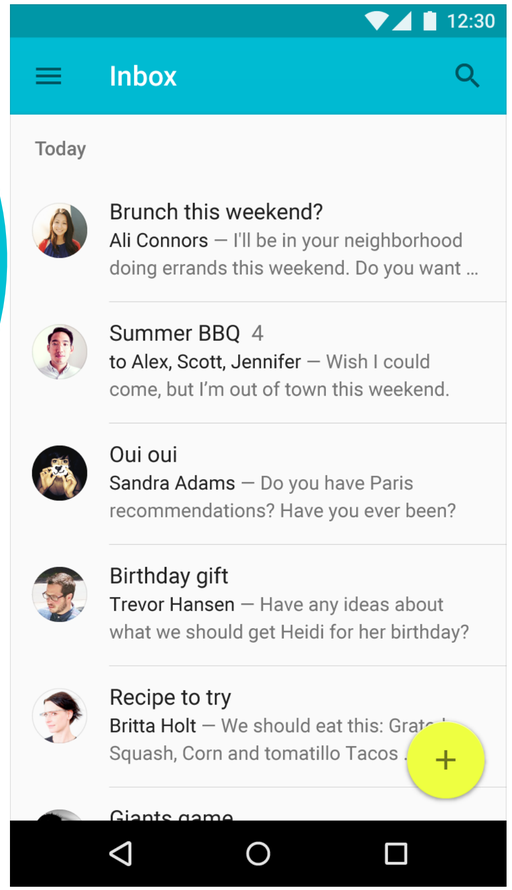
Усложнение задачи

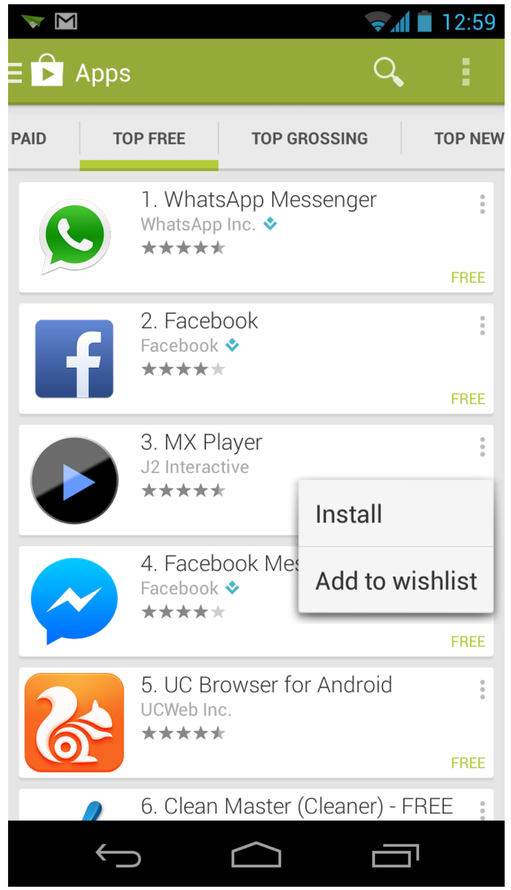


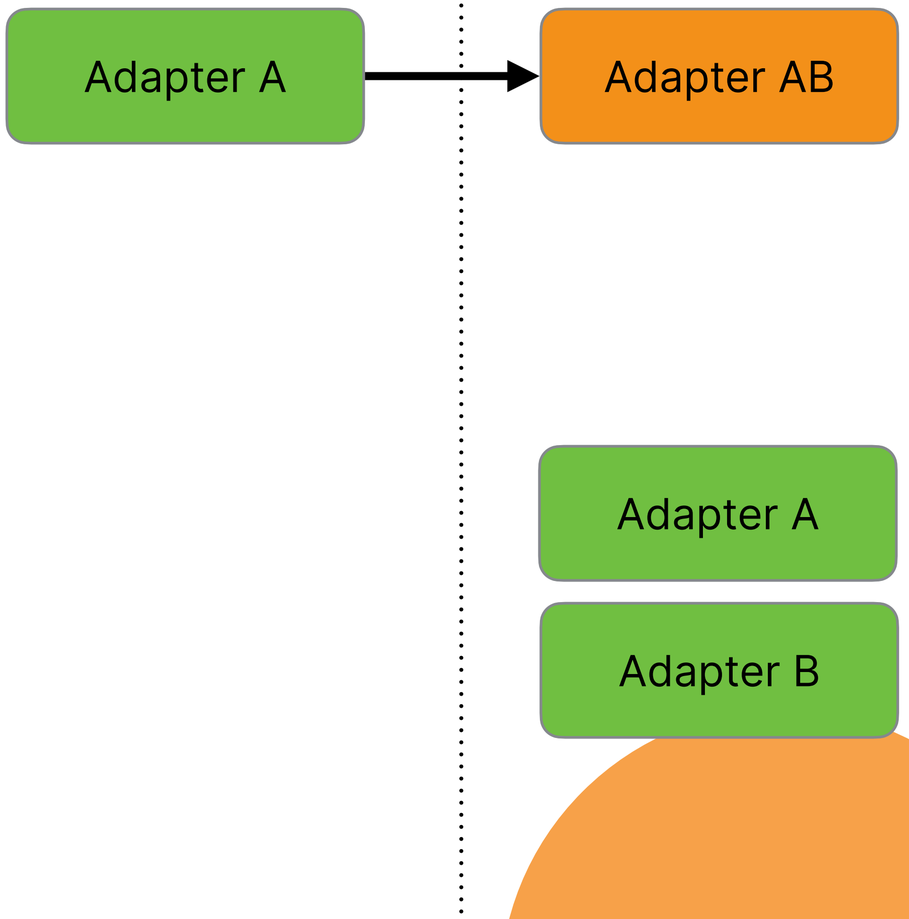




Усложнение задачи



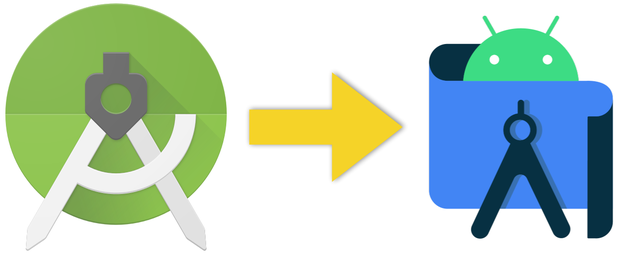




Технические источники

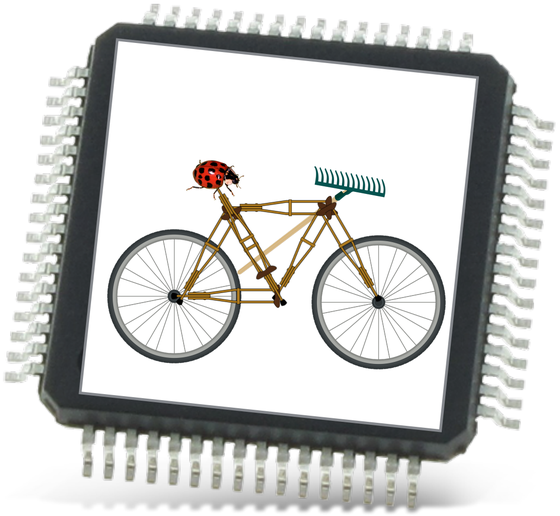
Обновления платформы

Обновления библиотек



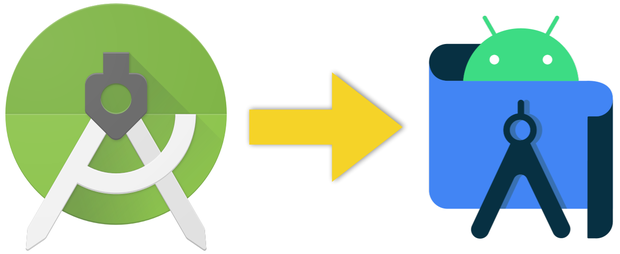
Обновления IDE

Технические источники



Обновления платформы

Обновления библиотек

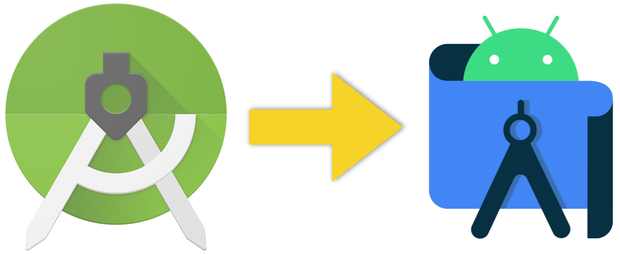


Обновления IDE

Технические источники

Обновления платформы

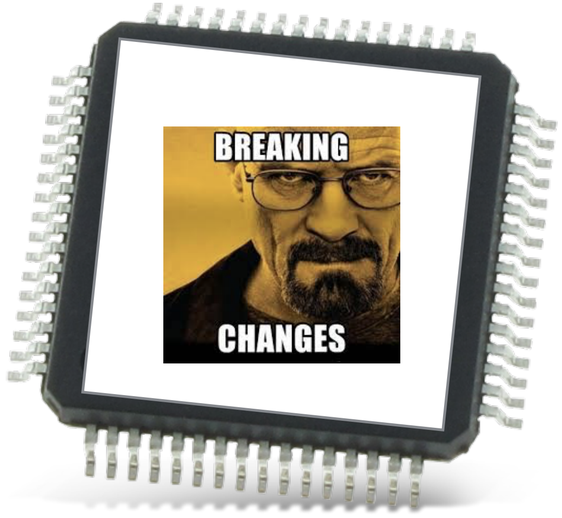
Обновления библиотек



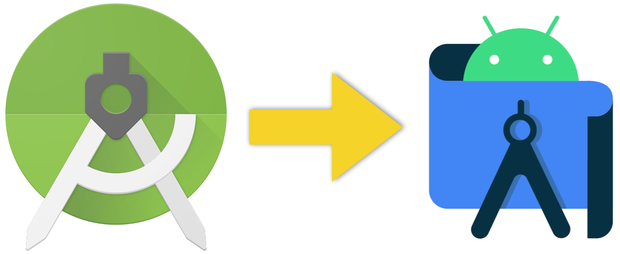
Обновления IDE

Технические источники

Обновления платформы



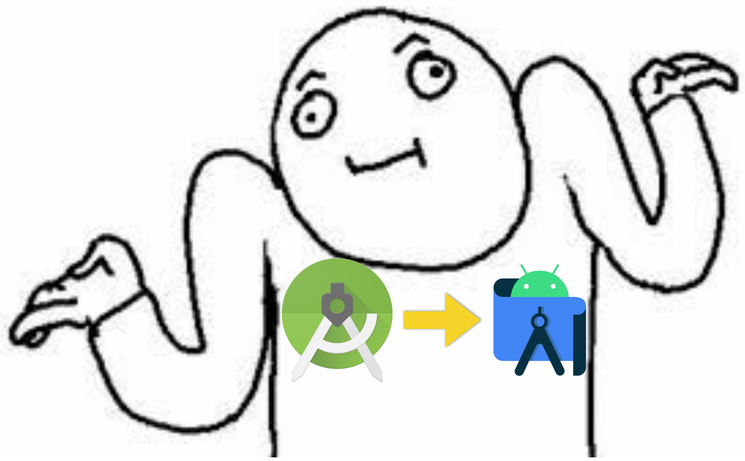
Обновления библиотек



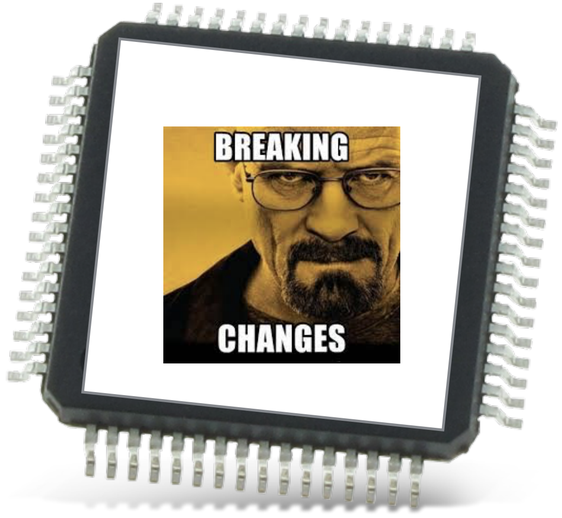
Обновления IDE

Технические источники

Обновления платформы

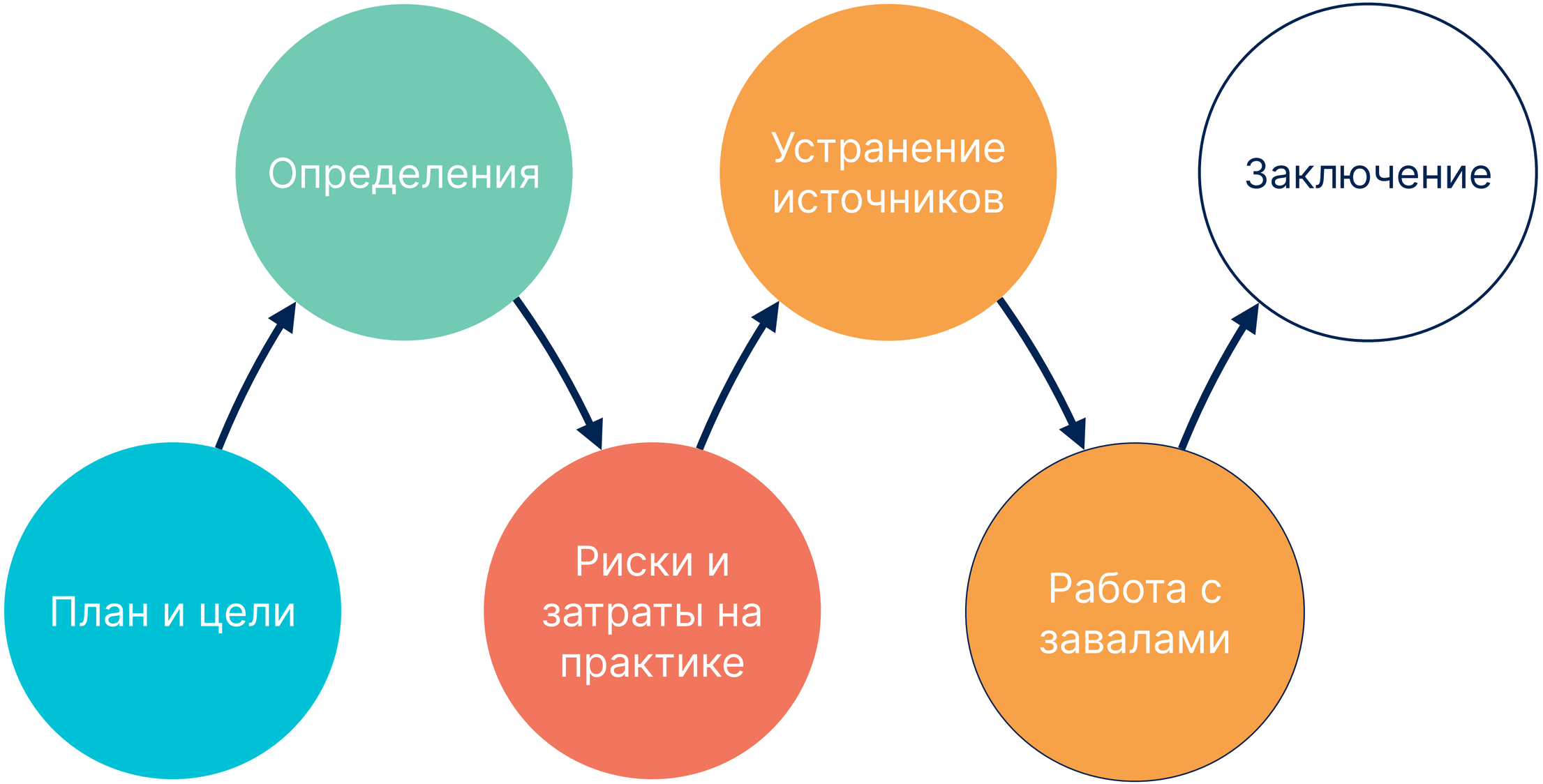


Обновления IDE



Обновления библиотек

План


Определения
Устранение источников
Заключение
План и цели
Риски и затраты на практике
Работа с завалами

Завалы легаси

Старый код

Утилиты

Ненужный код

Другой контекст

Завалы легаси

Старый код

Утилиты

Ненужный код

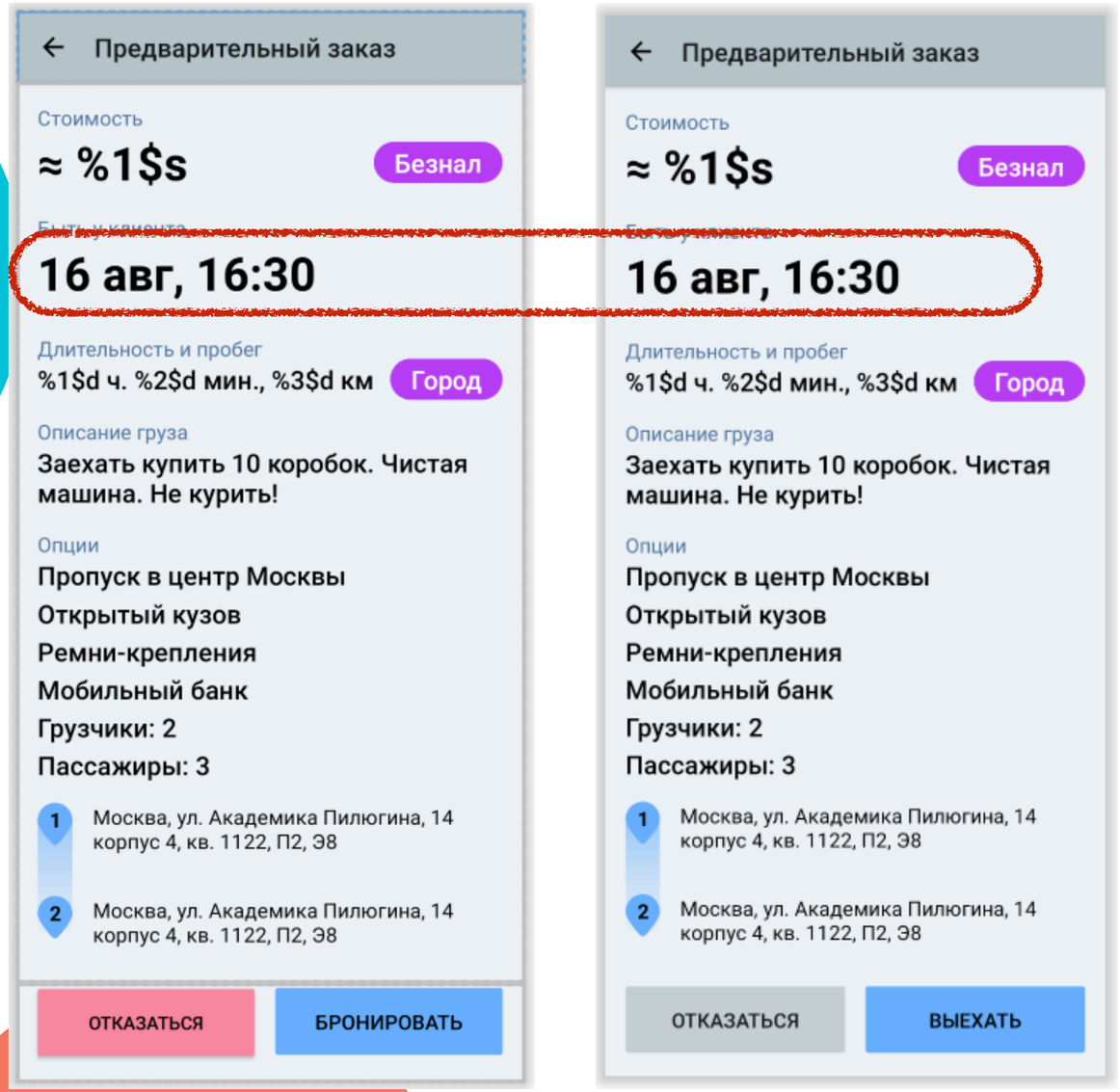
Другой контекст

Утилиты = легаси

Изоляция кода была проведена
на уровне алгоритмов, а не задач

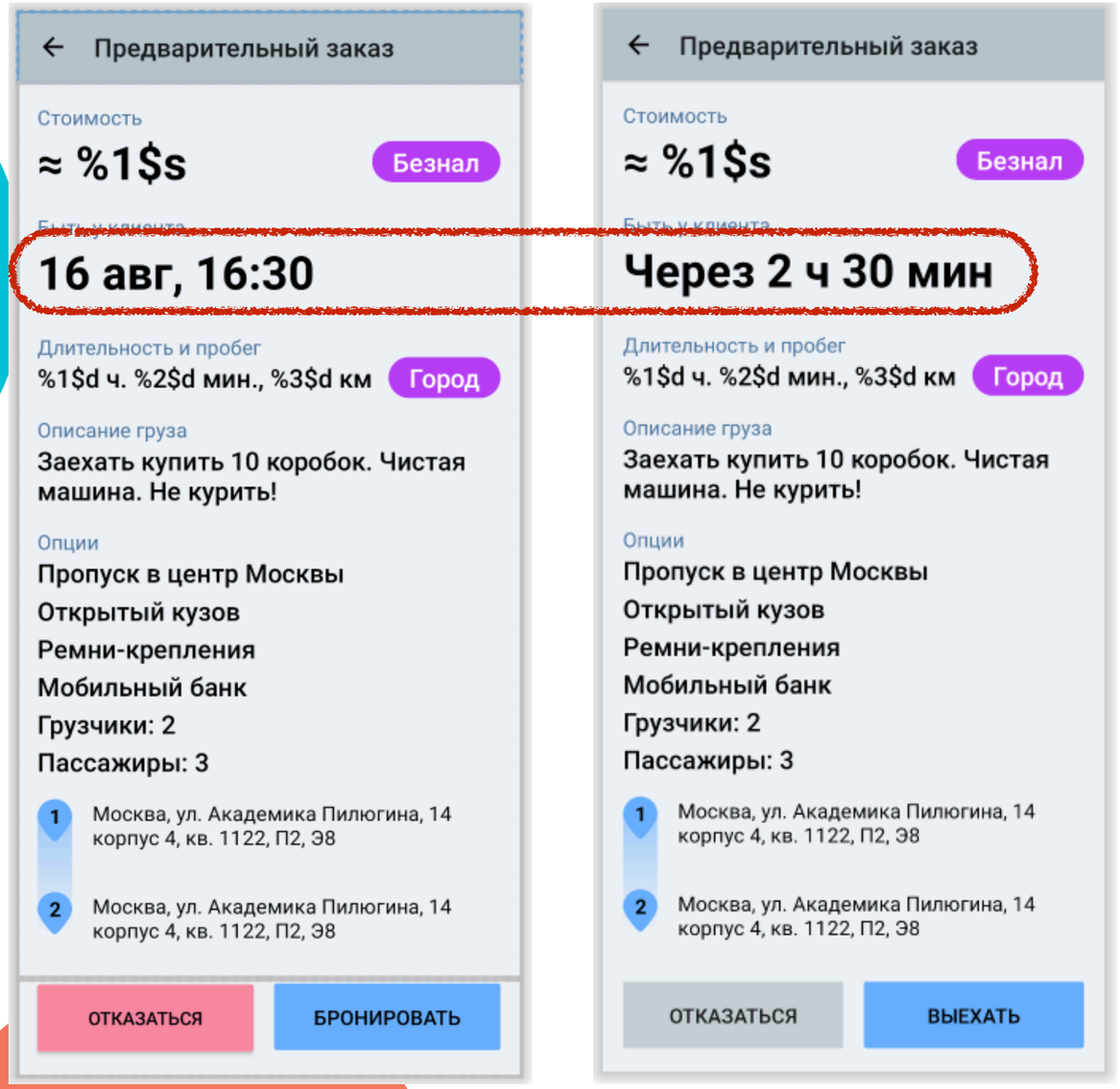
Результат оверинжиниринга
(SRP & DRY done wrong)

Утилиты = легаси



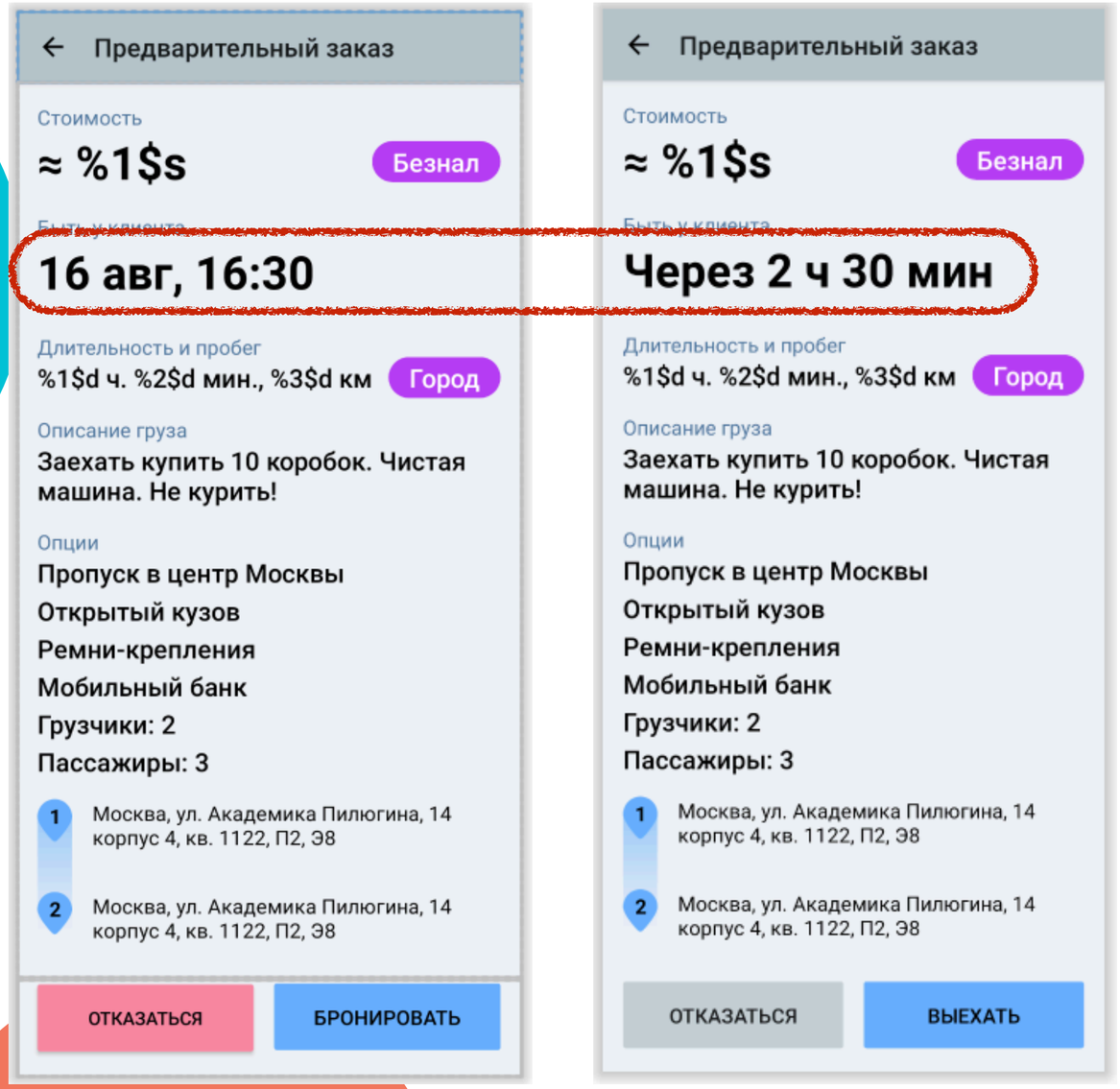

```
44
45  fun Order.formattedDate(): String =
46          DateTimeFormat
47                  .forPattern("d MMM, HH:mm")
48                  .print(DateTime(scheduledStartTime))
49
```

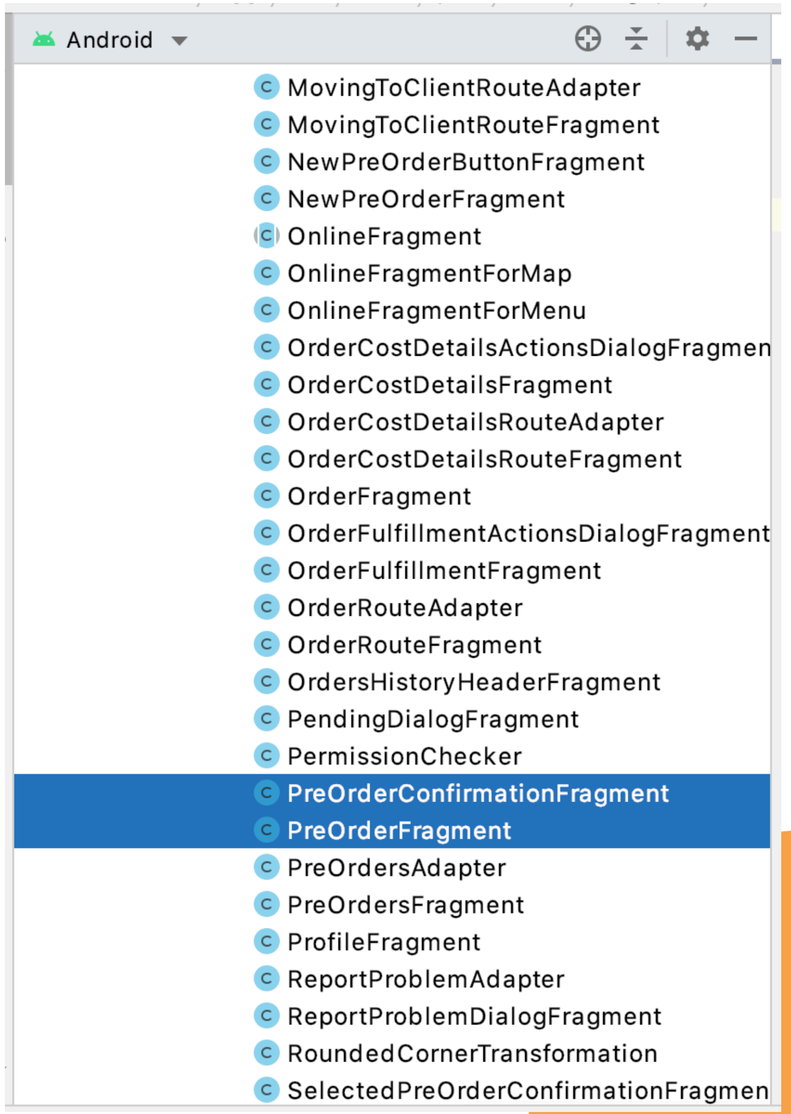
Утилиты = легаси



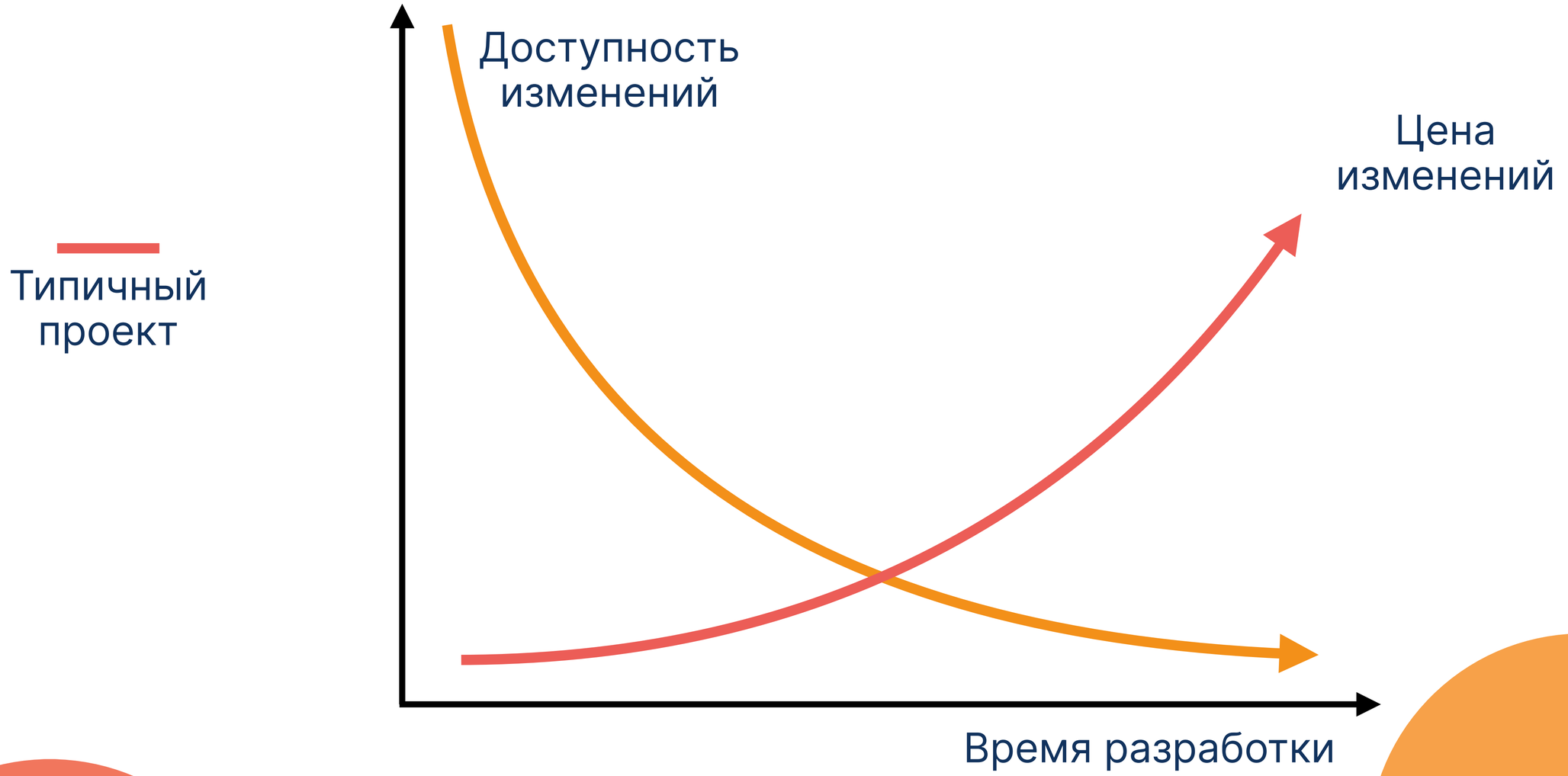

```
44  fun Order.formattedDate(goNow: Boolean): String =
45          if (goNow) {
46              val hours = Hours.hoursBetween(
47                      DateTime.now(),
48                      DateTime(scheduledStartTime)
49              )
50              val minutes = Minutes.minutesBetween(
51                      DateTime.now(),
52                      DateTime(scheduledStartTime)
53              )
54              "Через $hours ч $minutes мин"
55          } else {
56              DateTimeFormat
57                      .forPattern("d MMM, HH:mm")
58                      .print(DateTime(scheduledStartTime))
59          }
```

Утилиты = легаси





Технический долг



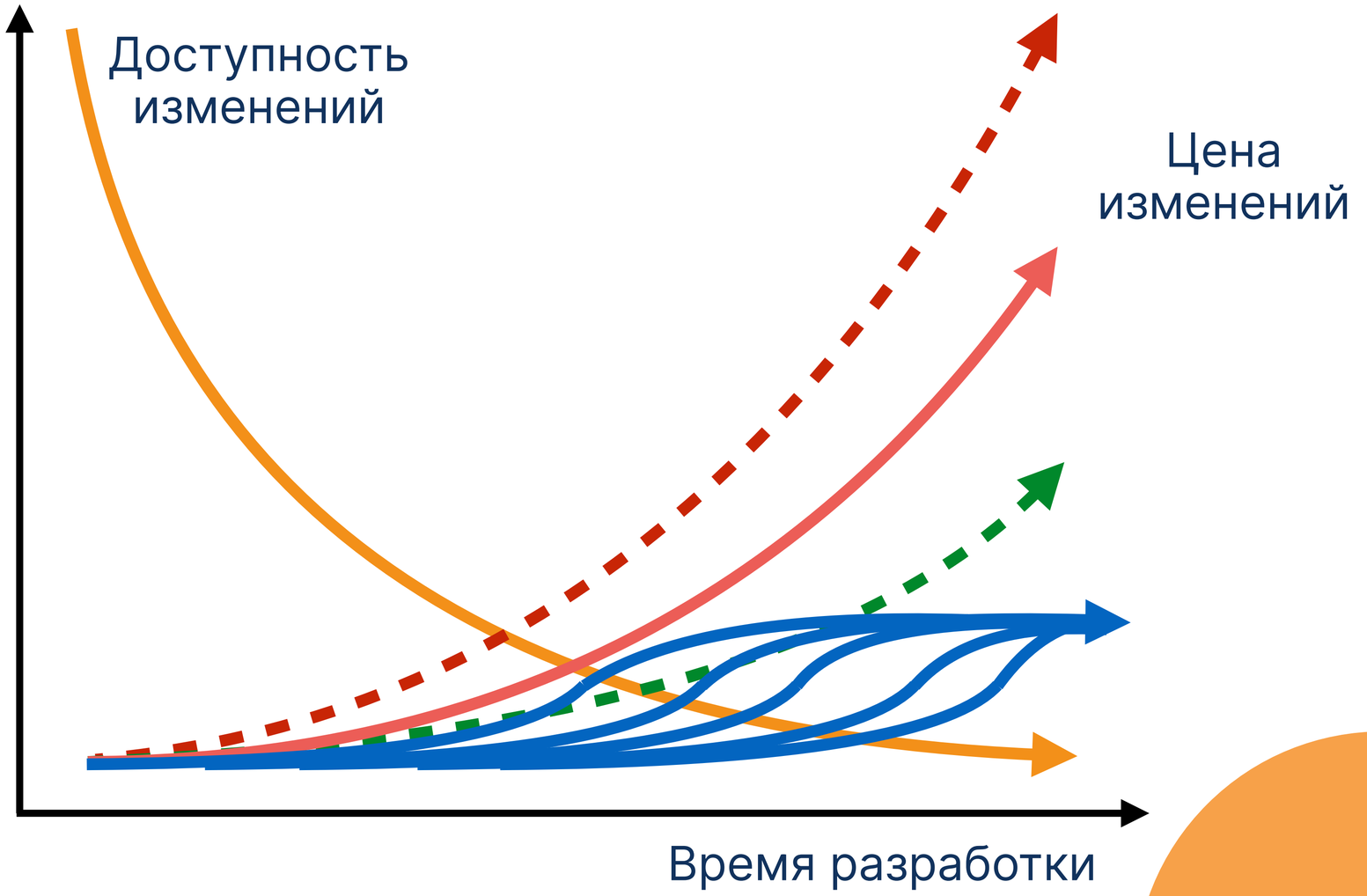
Технический долг

Овер-
инжиниринг

Типичный
проект

KISS

Изоляция



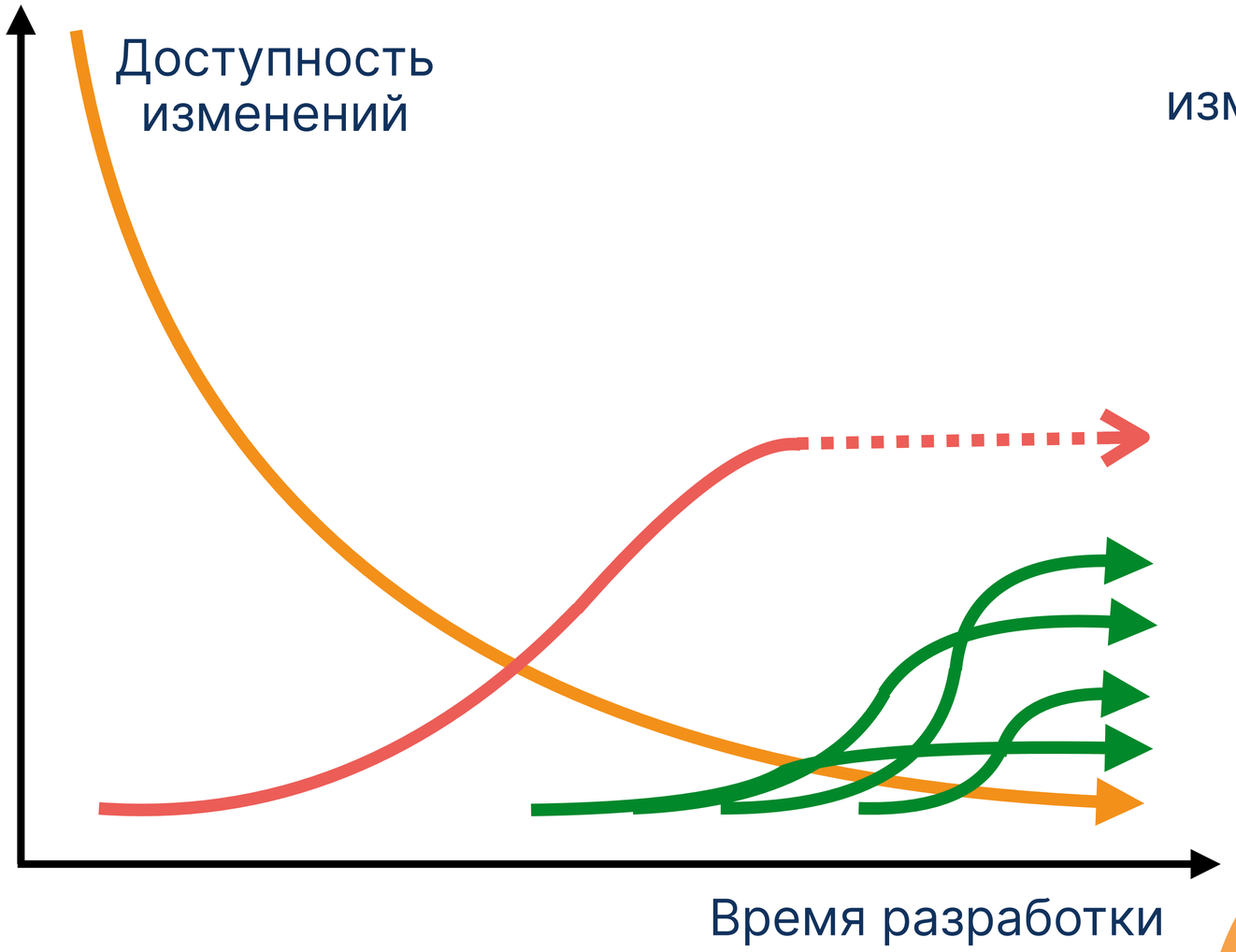
Изоляция легаси

Легаси

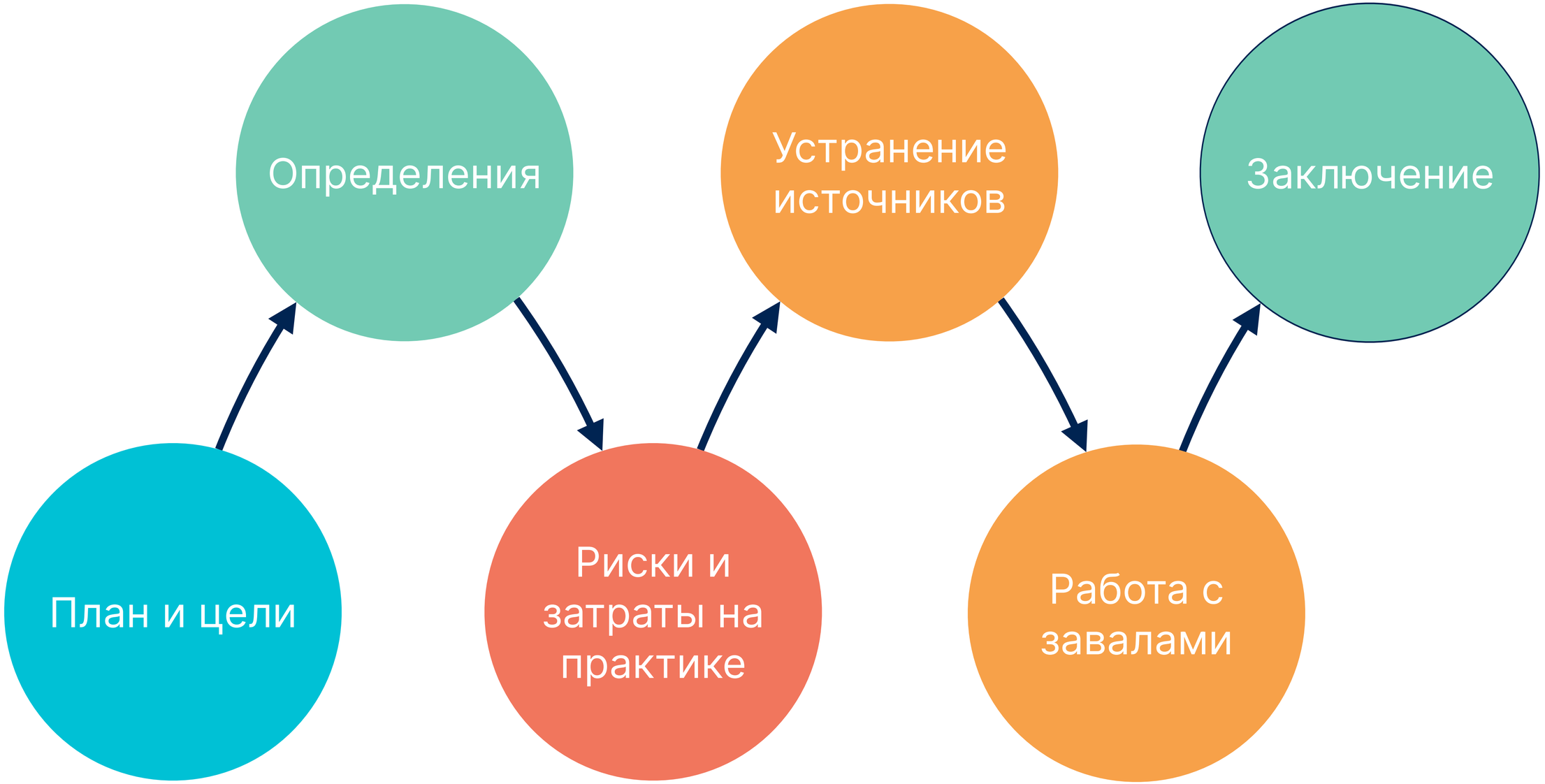
Изоляция легаси

Легаси

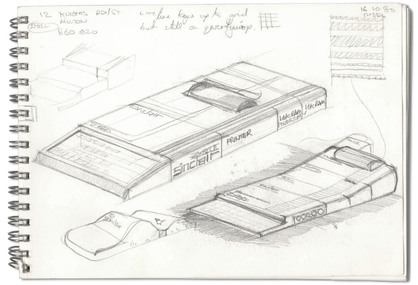
Новый код

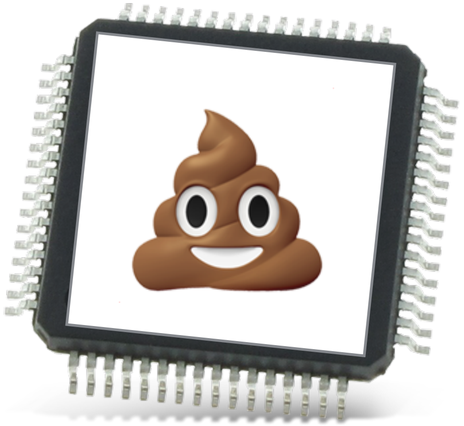


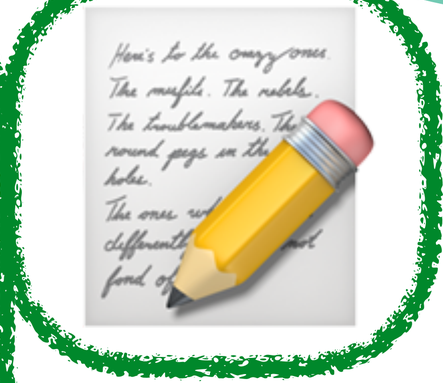
План

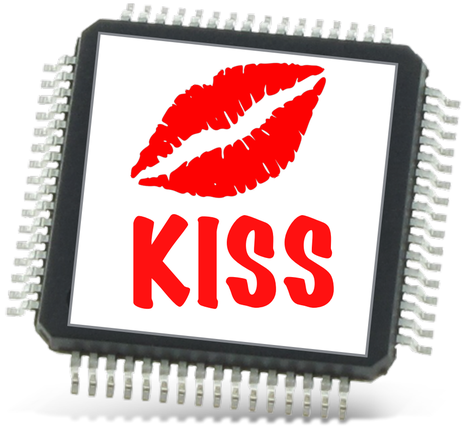

План и цели
Определения
Риски и затраты на практике
Устранение источников
Работа с завалами
Заключение

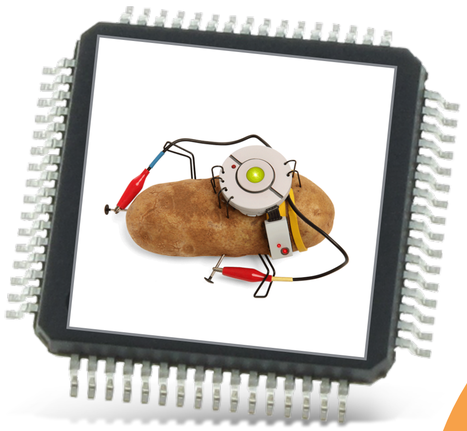
Итого












Слышно?
Видно?
Autocue
Autocue
WDR
1

Владимир Шутов

Руководитель iOS группы

@ shunegus

id49414271

История запуска проекта с наследием 🚀

План

- Приложение и MVP
- Исходные данные
- Планирование
- Релизация
  - Теория
  - Архитектура
- Результат

Приложение

РЃРёС‚СЂPs PSPs PSPµC,

Электронный кошелек

- Выпуск банковской карты
- Оплата счетов
- Перевод средств
- YM / Qiwi / т.п
- Работает в России и Европе

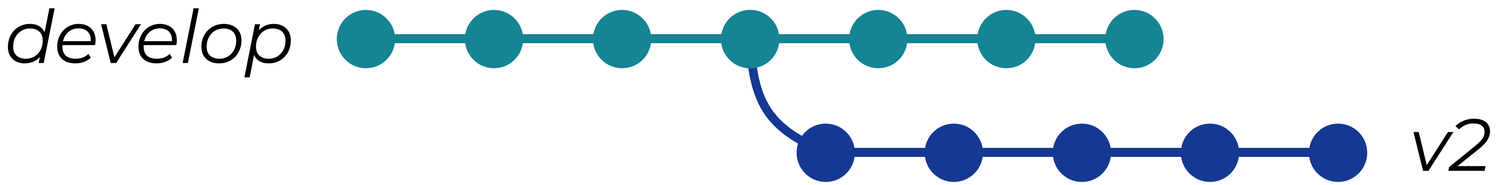
MVP

Фокус на приложении

- Новый UI/UX
- Новые продукты
- Обмен валют
- API v1 -> v2
- Эксперименты

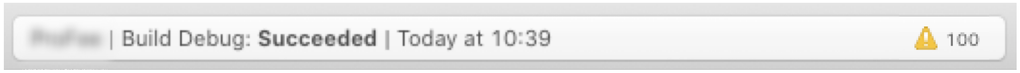
- Приложение и MVP
- **Исходные данные**
- Планирование
- Релизация
  - Теория
  - Архитектура
- Результат

Версии



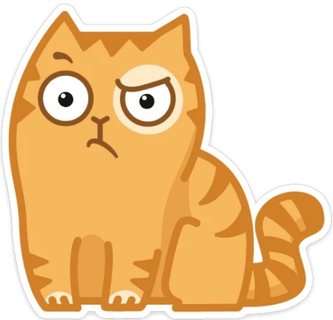
Xcode
Deprecated

IDE Говорит, что совсем чуть-чуть deprecated



SwiftLint Говорит, что смотреть надо внимательнее

Total warnings	20,516
Total errors	380



Тесты

```
class NDA_Tests: XCTestCase {

    override func setUp() {
        super.setUp()
        // Put setup code here. This method is called before the invocation of each test method in the class.
    }

    override func tearDown() {
        // Put teardown code here. This method is called after the invocation of each test method in the class.
        super.tearDown()
    }

    func testExample() {
        // This is an example of a functional test case.
        // Use XCTAssert and related functions to verify your tests produce the correct results.
    }

    func testPerformanceExample() {
        // This is an example of a performance test case.
        self.measure {
            // Put the code you want to measure the time of here.
        }
    }
}
```

Тестов нет, но вы держитесь

Иерархия проекта
Коммунальные файлы

```
13  final class BillListController: Controller, UITableViewDelegate, UITableViewDataSource,
        AlertViewRenderer, SpinnerViewRenderer { ••• }
499
500 final class PlaceholderCell: BaseCell { ••• }
537
538 final class BillListCell: BaseCell { ••• }
573
574 extension Bill { ••• }
591
592 final class BillListItemView: UIView { ••• }
732
733 final class ErrorView : UIView { ••• }
```

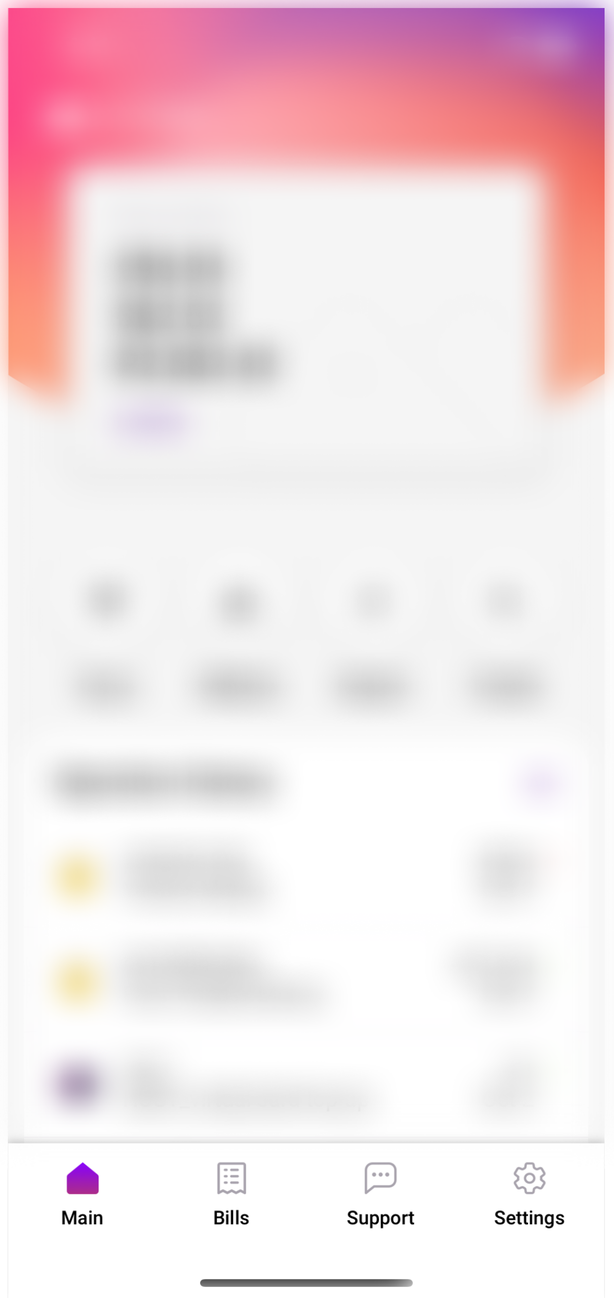
Code style
Objective-Swift

```
weak var weakSelf = self

listOperation = api.getFavorites(
    userID: user.id!,
    token: user.accessToken!,
    completion: { (favorites) in
        guard nil != weakSelf && !weakSelf!.finished else {
            return
        }

        if 0 != favorites.count {
            weakSelf!.listController.adjust(for: .showFavorites(favorites: favorites))
```

UIKit

Кастомизация

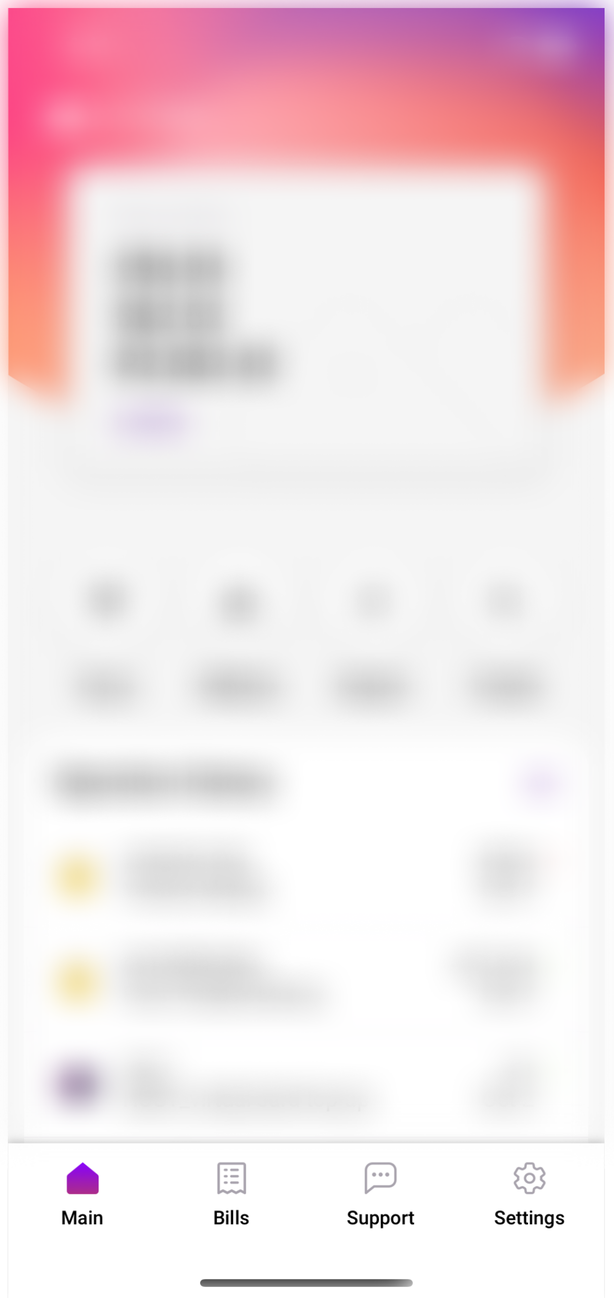
- UINavigationBar
- UITabBar



UIKit

Кастомизация

- ~~UINavigationBar~~
- ~~UITabBar~~




ВЕСЕЛО
ТУТ
У ВАС
HAPPY
NEW
YEAR.

- Приложение и MVP
- Исходные данные
- **Планирование**
- Релизация
  - Теория
  - Архитектура
- Результат

Срок реализации 2.5 месяца 😱

Реакция

Разработка

Реакция

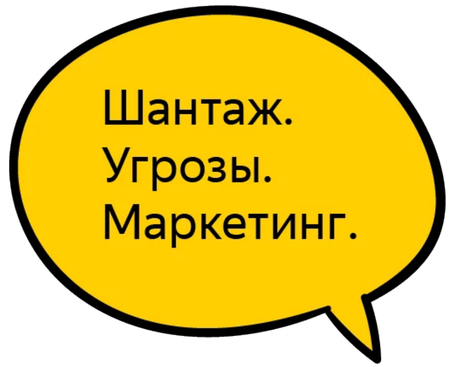
Разработка

Бизнес

Мыслим 🤔

Оценки

- Оценка с 0
- Оценка с легаси



Мыслим 🤔

Оценки

- Оценка с 0
- Оценка с легаси

План

MVP готова к релизу после каждого спринта

Мыслим 🤔

Оценки

- Оценка с 0
- Оценка с легаси

План

MVP готова к релизу после каждого спринта

- Инкапсуляция легаси

Мыслим 🤔

Оценки

- Оценка с 0
- Оценка с легаси

План

MVP готова к релизу после каждого спринта

- Инкапсуляция легаси
- Встраивание в архитектуру

Мыслим 🤔

Оценки

- Оценка с 0
- Оценка с легаси

План

MVP готова к релизу после каждого спринта

- Инкапсуляция легаси
- Встраивание в архитектуру
- Новые фичи в нашем стиле

Мыслим 🤔

Оценки

- Оценка с 0
- Оценка с легаси

План

MVP готова к релизу после каждого спринта

- Инкапсуляция легаси
- Встраивание в архитектуру
- Новые фичи в нашем стиле
- Переписываем легаси

Мыслим 🤔

Оценки

- Оценка с 0
- Оценка с легаси

План

MVP готова к релизу после каждого спринта

- Инкапсуляция легаси
- Встраивание в архитектуру
- Новые фичи в нашем стиле
- Переписываем легаси
- 😭

Мыслим 🤔

Оценки

- Оценка с 0
- Оценка с легаси

План

MVP готова к релизу после каждого спринта

- Инкапсуляция легаси
- Встраивание в архитектуру
- Новые фичи в нашем стиле
- Переписываем легаси
- 😭
- Успех

Советы

- Не отчаиваться

Советы

- Не отчаиваться
- Вспомни все что знал и слышал - *NZT-48*

Советы

- Не отчаиваться
- Вспомни все что знал и слышал - *NZT-48*
- Советуйся с другими, даже с другими командами

Советы

- Не отчаиваться
- Вспомни все что знал и слышал - *NZT-48*
- Советуйся с другими, даже с другими командами
- Учитывай бизнес

Советы

- Не отчаиваться
- Вспомни все что знал и слышал - *NZT-48*
- Советуйся с другими, даже с другими командами
- Учитывай бизнес
- Ищите компромисс

- Приложение и MVP
- Исходные данные
- Планирование
- **Релизация**
  - **Теория**
    - Архитектура
- Результат

Теория
Определение

- Legacy код – это код, который достался вам по наследству.

Теория

Определение

- Legacy код – это код, который достался вам по наследству.
- Legacy — это код, который не покрыт тестами.

Теория
Определение

- Legacy код – это код, который достался вам по наследству.
- Legacy — это код, который не покрыт тестами.
  - *Даже если он чистый*

Теория

Определение

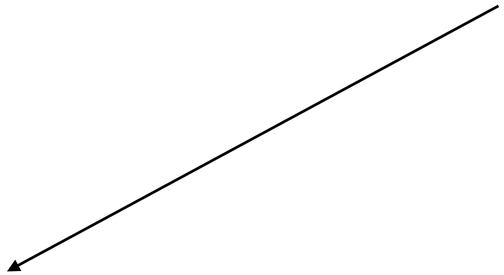
- Legacy код – это код, который достался вам по наследству.
- Legacy — это код, который не покрыт тестами.
  - *Даже если он чистый*
  - *Хорошо спроектирован*

Теория
Определение

- Legacy код – это код, который достался вам по наследству.
- Legacy — это код, который не покрыт тестами.
  - *Даже если он чистый*
  - *Хорошо спроектирован*

Два пути правки легаси

Два пути правки легаси



Защити и правь

Два пути правки легаси

Защити и правь

Правь и молись

Теория

Уменьшаем риск

Теория
Уменьшаем риск

- Какие изменения нам нужно сделать?

Теория
Уменьшаем риск

- Какие изменения нам нужно сделать?
- Как мы узнаем, что сделали их правильно?

Теория
Уменьшаем риск

- Какие изменения нам нужно сделать?
- Как мы узнаем, что сделали их правильно?
- Как мы узнаем, что ничего не нарушили?

Теория

Уменьшаем риск

- Какие изменения нам нужно сделать?
- Как мы узнаем, что сделали их правильно?
- Как мы узнаем, что ничего не нарушили?
- До какой степени мы можем пойти на изменения, чтобы они не оказались рискованными?

Теория
Почкование метода

```swift
/// Получить карты пользователя
func obtainUserCards() -> [UserCard] {
  let cards = loadCards()
  let userCards = cards.map {
    UserCard(name: "...", mode: $0.state.rawValue)
  }
  return userCards
}
```

```swift
/// Получить карты пользователя
func obtainUserCards() -> [UserCard] {
  var cards = loadCards()
  cards = actualCasrds(cards)
  let userCards = cards.map {
    UserCard(name: "...", mode: $0.state.rawValue)
  }
  return userCards
}

func actualCasrds(_ cards: [VirtualCard]) -> [VirtualCard] {
  return cards.filter { $0.state != .blocked }
}
```

Теория
Почкование метода

```swift
/// Получить карты пользователя
func obtainUserCards() -> [UserCard] {
  let cards = loadCards()
  let userCards = cards.map {
    UserCard(name: "...", mode: $0.state.rawValue)
  }
  return userCards
}
```

```swift
/// Получить карты пользователя
func obtainUserCards() -> [UserCard] {
  var cards = loadCards()
  cards = actualCasrds(cards)
  let userCards = cards.map {
    UserCard(name: "...", mode: $0.state.rawValue)
  }
  return userCards
}

func actualCasrds(_ cards: [VirtualCard]) -> [VirtualCard] {
  return cards.filter { $0.state != .blocked }
}
```

Теория
Почкование метода

Минусы

- Исходный метод остается без внимания
- Иногда становится непонятно, почему действия происходят в другом методе

Плюсы

- Новый код полностью отделён от старого
- Возможно протестировать
- Видно все переменные влияющие на метод

Теория

Почкование класса

```
struct CardUseCaseImpl {
 init(
    cardService: CardService,
    cardCache: CardCache,
    userService: UserService
  ) {}

  /// Получить карты пользователя
  func obtainUserCards() -> [UserCard] {
    var cards = loadCards()
    cards = actualCasrds(cards)
    let userCards = cards.map {
      UserCard(name: "...", mode: $0.state.rawValue)
    }
    return userCards
  }
}
```

```
class CardUseCaseImpl {
  let casrdsUtils = ActualCasrdsImpl()
  init(
    cardService: CardService,
    cardCache: CardCache,
    userService: UserService
  ) {}

  /// Получить карты пользователя
  func obtainUserCards() -> [UserCard] {
    var cards = loadCards()
    cards = casrdsUtils.onlyActual(cards)
    let userCards = cards.map {
      UserCard(name: "...", mode: $0.state.rawValue)
    }
    return userCards
  }
}

class ActualCasrdsImpl {
  func onlyActual(_ cards: [VirtualCard]) -> [VirtualCard] {
    return cards.filter { $0.state != .blocked }
  }
}
```

Теория

Почкование класса

```
struct CardUseCaseImpl {
 init(
    cardService: CardService,
    cardCache: CardCache,
    userService: UserService
  ) {}

  /// Получить карты пользователя
  func obtainUserCards() -> [UserCard] {
    var cards = loadCards()
    cards = actualCasrds(cards)
    let userCards = cards.map {
      UserCard(name: "...", mode: $0.state.rawValue)
    }
    return userCards
  }
}
```

```
class CardUseCaseImpl {
  let casrdsUtils = CasrdsUtils()
  init(
    cardService: CardService,
    cardCache: CardCache,
    userService: UserService
  ) {}

  /// Получить карты пользователя
  func obtainUserCards() -> [UserCard] {
    var cards = loadCards()
    cards = casrdsUtils.onlyActual(cards)
    let userCards = cards.map {
      UserCard(name: "...", mode: $0.state.rawValue)
    }
    return userCards
  }
}

class CasrdsUtils {
  func onlyActual(_ cards: [VirtualCard]) -> [VirtualCard] {
    return cards.filter { $0.state != .blocked }
  }
}
```

Теория

Почкование класса

Минусы

- Концептуальная сложность

Плюсы

- Добавляет уверенности

Особенность

- Сам факт наличия этого дополнительного нового класса в системе дает немало пищи для размышления

Теория

Охват метода и охват класса

```
/// Получить карты пользователя старый метод
func saveUserCards(_ cards: [VirtualCard]) {
  cards = casrdsUtils.onlyActual(cards)
  let userCards = cards.map {
    UserCard(name: "...", mode: $0.state.rawValue)
  }
  …… next code ……
}
```

```
/// Получить карты пользователя
func saveCards(_ cards: [VirtualCard]) {
  old_saveUserCards(cards)
  logEventUserCards()
}

/// Получить карты пользователя старый метод
func old_saveUserCards(_ cards: [VirtualCard]) {
  var cards = loadCards()
  cards = casrdsUtils.onlyActual(cards)
  let userCards = cards.map {
    UserCard(name: "...", mode: $0.state.rawValue)
  }
}

func logEventUserCards() {
  FileLogs.event("Saved cards")
}
```

Теория

Охват метода и охват класса

```
/// Получить карты пользователя старый метод
func saveUserCards(_ cards: [VirtualCard]) {
  cards = casrdsUtils.onlyActual(cards)
  let userCards = cards.map {
    UserCard(name: "...", mode: $0.state.rawValue)
  }
  …… next code ……
}
```

```
/// Получить карты пользователя
func saveCards(_ cards: [VirtualCard]) {
  old_saveUserCards(cards)
  logEventUserCards()
}

/// Получить карты пользователя старый метод
func old_saveUserCards(_ cards: [VirtualCard]) {
  var cards = loadCards()
  cards = casrdsUtils.onlyActual(cards)
  let userCards = cards.map {
    UserCard(name: "...", mode: $0.state.rawValue)
  }
}

func logEventUserCards() {
  FileLogs.event("Saved cards")
}
```

Остальное

Primitivize Parameter

Извлечение и переопределение получателя

Ввод делег

Вынос объекта метода

Примитивизация параметра

Pull Up Feature

Раскрытие статического метода.

Parameterize Method

Адапта

Замена глобальной ссылки полу

Expose Static Method

Adapt Parameter

Ввод с

Подклассификация и переопределение метода

Break Out Method Object

Извлечение интерфейса

Извлечение и пер

Extract and Override Call

Definition Completion

Вытеснение зависим

Extract and Override Getter

Introduce Instance Delegator

Introduce

Па

Извлечение и переопределение фабричного метода

Extract Implementer

Replace Function with Functi

Extract Interface

Извлечение средств реализации

Extra

Инкапсуляция глобальных ссылок

Вытягивание свойства

Supersede Instance

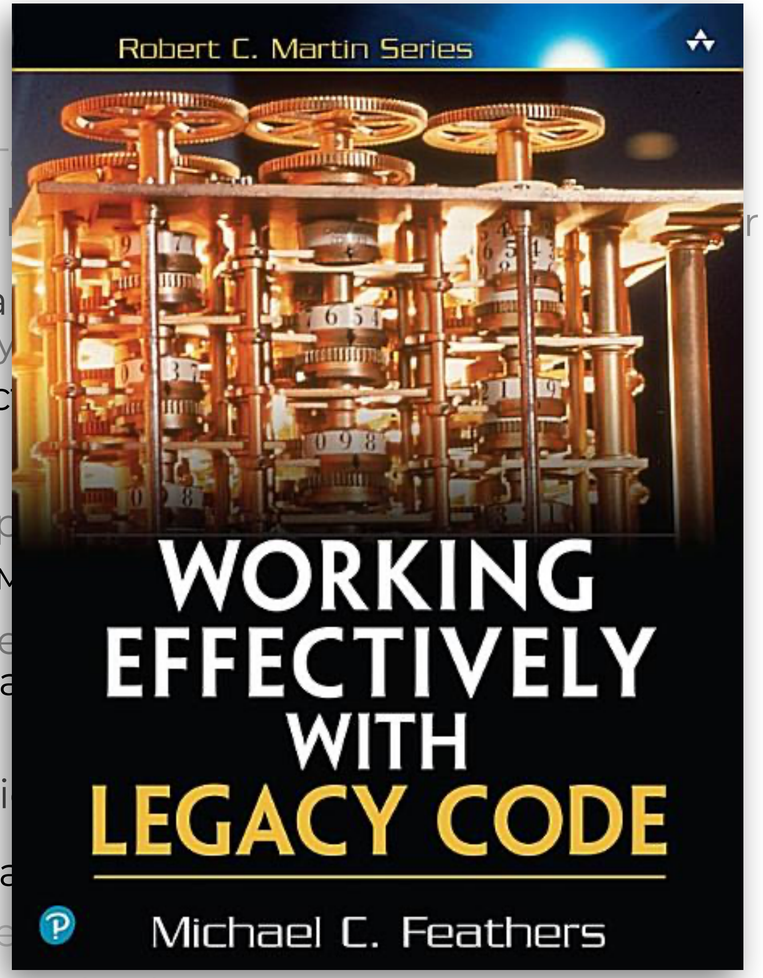
Push Down Dependency

Subclass and Override Method

Template Redefinition

Text Redefinition

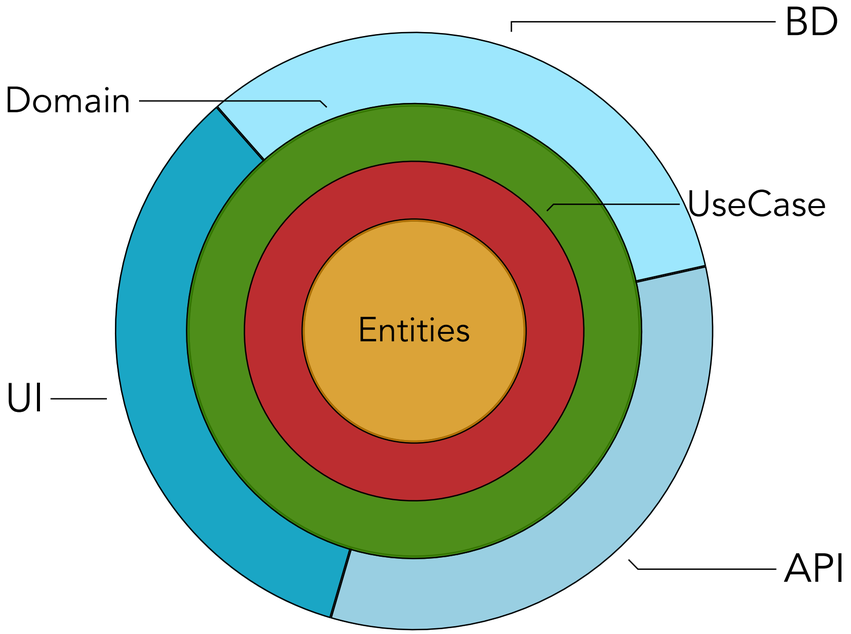
Parameterize Constructor



- Приложение и MVP
- Исходные данные
- Планирование
- **Релизация**
  - Теория
  - **Архитектура**
- Результат

Clean Architecture

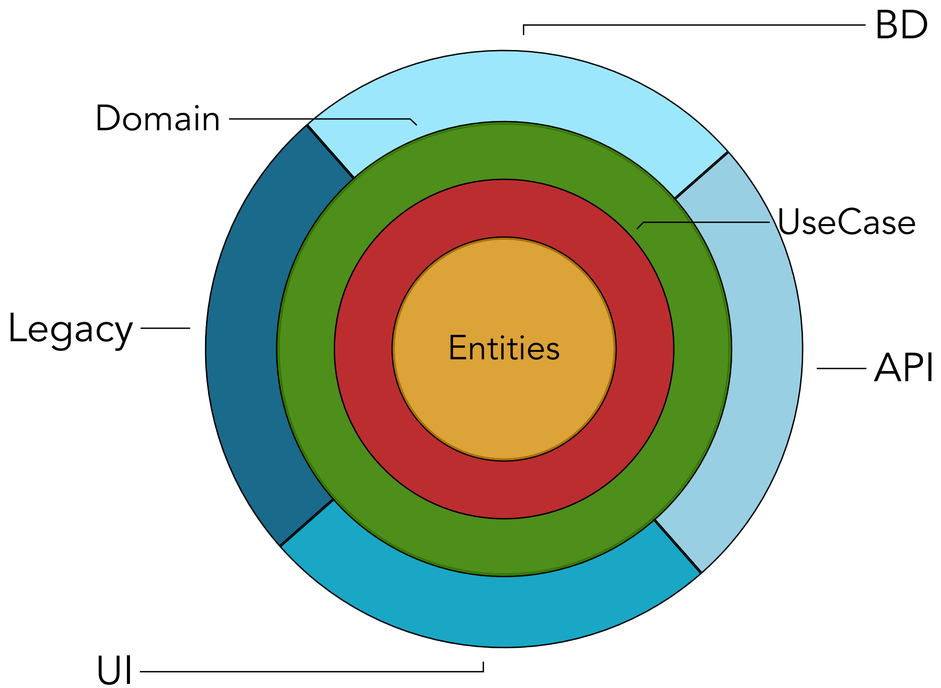
RxSwift



- Domain
- Data
- Logger
- AppLocale

Clean Architecture

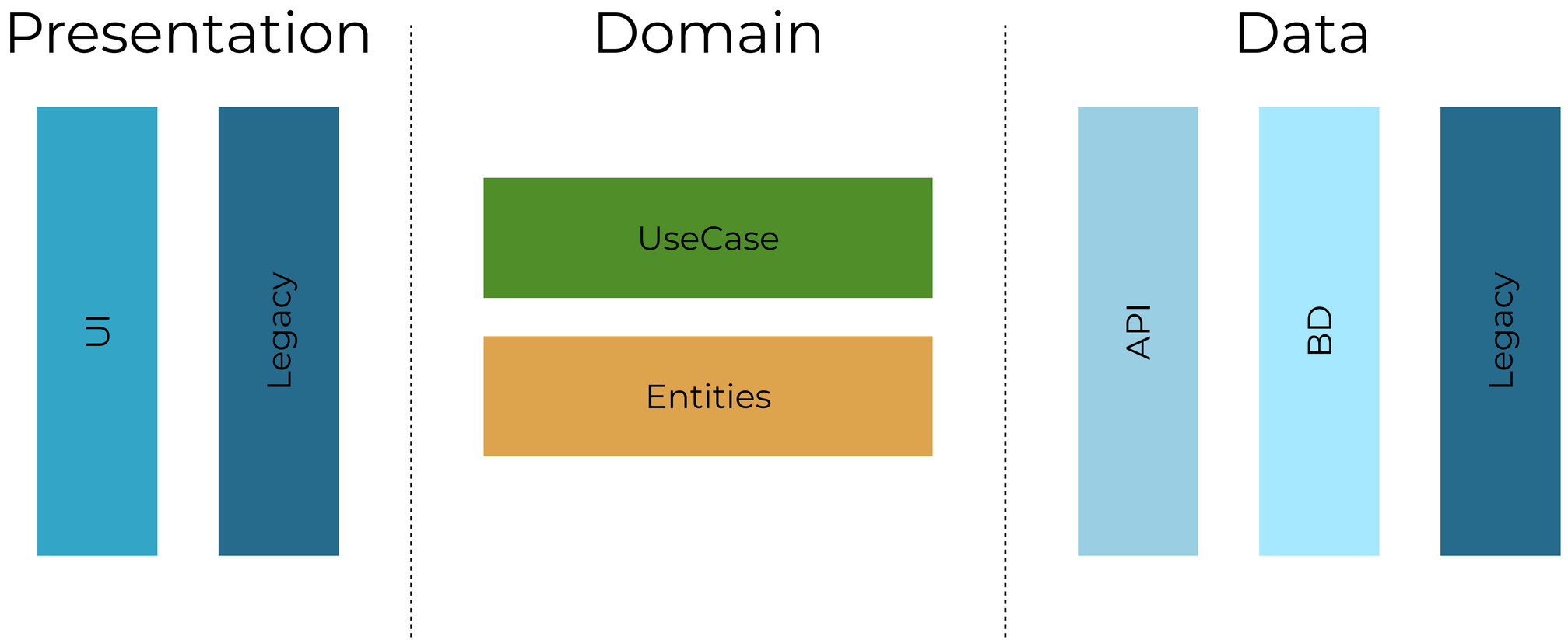
Legacy как компонент



- Legacy
- Domain
- Data
- Logger
- AppLocale

Clean Architecture

Legacy как компонент



Как делали
Шаг 0

- Обязательное кросревью на проекте, включая клиента

Как делали

Шаг 0

- Обязательное кросревью на проекте, включая клиента
- Злой линтер

Как делали
Шаг 0

- Обязательное кросревью на проекте, включая клиента
- Злой линтер
- Подтянули кодогенераторы

Как делали
Шаг 0

- Обязательное кросревью на проекте, включая клиента
- Злой линтер
- Подтянули кодогенераторы
- Коммуникация в одном месенжере

Как делали
Шаг 0

- Обязательное кросревью на проекте, включая клиента
- Злой линтер
- Подтянули кодогенераторы
- Коммуникация в одном месенжере
- Быстрая связь с другими командами

Шаг 1

- Создание нового проекта
- Перенос всего Legacy
  - Цель: Получить последний XProject и запуск проекта без вмешательств

Шаг 2

- Настройка Чистой Архитектуры
  - Цель: Стандартизация проекта под 65apps

Шаг 3

- Интеграция Чистой Архитектуры с Legacy
  - Цель: Стабильная работа Legacy в новой обертке

Legacy
UI связь

```
public protocol LegacyAdapterDelegate {
    func currentAppWindow() -> UIWindow?
    func userDidSignOut()
    func userDidLocked()
}

public final class LegacyAdapter {

    // MARK: - Public properties

    public var delegate: LegacyAdapterDelegate?

    // MARK: - Public methods

    public func billStory() -> UIViewController {
        return LegacyAppDelegateProxy.shared.makeBillStory()
    }
```

UI Routing

- Flexible Routing

```swift
protocol ExchangeRoute {
  /// Показать обмен валют
  func showExchange(currency: Currency)
}

extension ExchangeRoute where Self: RouterProtocol {

  /// Дефолтная реализация
  func showExchange(currency: Currency) {
    let transition = PushTransition()
    let module = ExchangeStory(transition: transition, currency: currency)
    open(module.view, transition: transition)
  }
}
```

```swift
protocol WithdrawRoute {
  /// Показать WithdrawStory
  func showWithdrawStory()
}

extension TopUpRoute where Self: RouterProtocol {

  func showWithdrawStory() {
    guard let navigationController = viewController?.navigationController else {
      return
    }
    LegacyAdapter.shared.showWithdrawStory(with: navigationController)
  }
}
```

UI Routing

- Flexible Routing

```swift
protocol ExchangeRoute {
  /// Показать обмен валют
  func showExchange(currency: Currency)
}

extension ExchangeRoute where Self: RouterProtocol {

  /// Дефолтная реализация
  func showExchange(currency: Currency) {
    let transition = PushTransition()
    let module = ExchangeStory(transition: transition, currency: currency)
    open(module.view, transition: transition)
  }
}
```

```swift
protocol WithdrawRoute {
  /// Показать WithdrawStory
  func showWithdrawStory()
}

extension TopUpRoute where Self: RouterProtocol {

  func showWithdrawStory() {
    guard let navigationController = viewController?.navigationController else {
      return
    }
    LegacyAdapter.shared.showWithdrawStory(with: navigationController)
  }
}
```

Legacy
Data связь

```
public class LegacyDataProvider {

    public func changeLanguage(_ language: Language) {
        AppSettings.shared.language = language
    }

    public func sendVirtualCardDetails(
        walletId: String,
        cardId: Int,
        callBack: @escaping (Error?) -> Void
    ) -> Cancelble? {

        let settings = AppSettings.shared
        guard let user = settings.user, let accessToken = user.jwt else {
            callBack(LegacyError.userNotExist)
            return nil
        }

        ..................

    }
```

```
public enum LegacyError: LocalizedError {
    case userNotExist
    case requestCancelled
    case networkOperation(String)
    case serialization(String)
    case request(String)
}
```

Legacy
Data связь

```
public class LegacyDataProvider {

    public func changeLanguage(_ language: Language) {
        AppSettings.shared.language = language
    }

    public func sendVirtualCardDetails(
        walletId: String,
        cardId: Int,
        callBack: @escaping (Error?) -> Void
    ) -> Cancelble? {

        let settings = AppSettings.shared
        guard let user = settings.user, let accessToken = user.jwt else {
            callBack(LegacyError.userNotExist)
            return nil
        }

        ..................

    }
```

```
public enum LegacyError: LocalizedError {
    case userNotExist
    case requestCancelled
    case networkOperation(String)
    case serialization(String)
    case request(String)
}
```

Domain
Data связь

```
/// Сервис по работе с продуктами
public protocol ProductService {
  /// Получить продукты
  func obtainProducts(walletId: String) -> Single<[Product]>

  /// Выпустить продукт
  func issueProduct(walletId: String, issuer: ProductIssuer) -> Single<ProductIssue>
}
```

Data

```
final class ProductService: Domain.ProductService {

  private let provider: MoyaProvider<ProductAPI>

  // MARK: - ProductService
  func obtainProducts(walletId: String) -> Single<[Product]> {
    return provider.rx
    .request(.obtainProducts(walletId))
    .flatMap(ErrorHandler.handleServiceError)
    .map(BodyMapper<[Product]>.self)
    .map { $0.body }
    .catchError(ErrorHandler.handleError)
  }
```

Domain
Data связь

```
/// Работа с пользователем
public protocol UserService {
  /// Получить данные пользователя
  func obtainUserCredentials() -> Single<UserCredentials?>

  /// Получить типы показанных сообщений юзеру
  func showedMessages() -> Single<[MessageType]>
```

Data

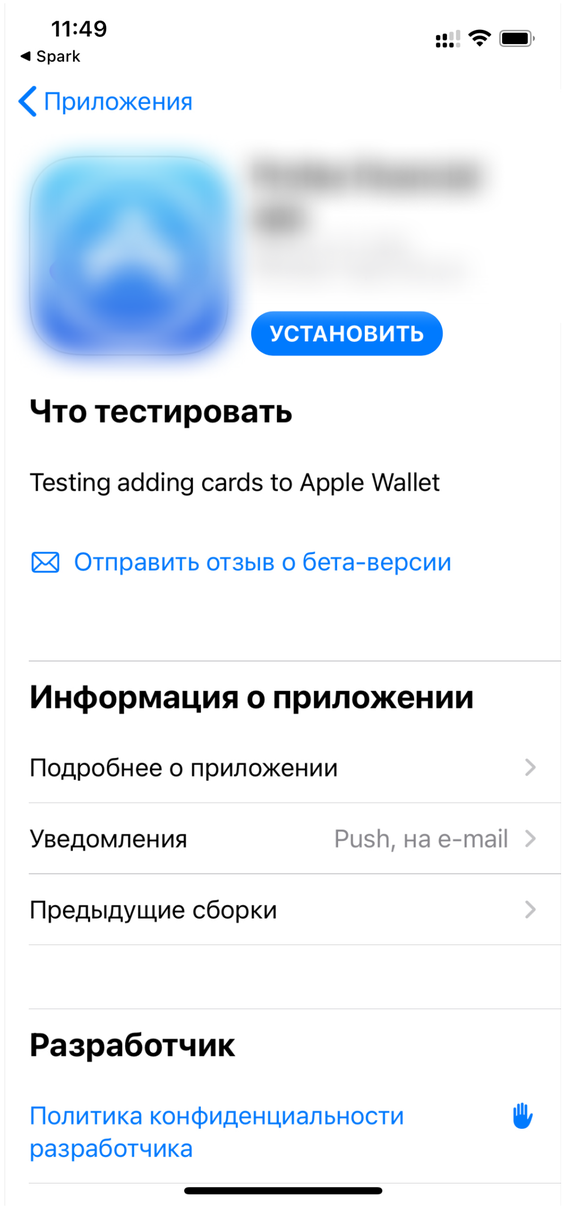
Legacy связь

```
final class UserService: Domain.UserService, CryptoRepository {

  private let provider: LegacyDataProvider

  // MARK: - UserService
  func obtainUserCredentials() -> Single<UserCredentials?> {
    return Single.create { [provider] subscribe -> Disposable in
      guard let user = provider.obtainUserCredentials() else {
        subscribe(.success(nil))
        return Disposables.create()
      }
      let userCredentials = ……….
      subscribe(.success(userCredentials))
      return Disposables.create()
    }
    .catchError(ErrorHandler.handleError)
  }
```

- Приложение и MVP
- Исходные данные
- Планирование
- Релизация
  - Теория
  - Архитектура
- **Результат**

Релиз
MVP

- Время 3 месяца



😭

Сюрпризы?

- LegacyViewController

```
import UIKit
import Legacy

class LegacyViewController:
  Controller,
  AlertViewRenderer,
  SpinnerViewRenderer,
  UserNavigationEvent {

  override func viewDidLoad() {
    super.viewDidLoad()
    needHideNavBar = false
    logTransition()
  }
}
```

😭

Сюрпризы?

- LegacyViewController
- Legacy пришлось править

```
import UIKit
import Legacy

class LegacyViewController:
  Controller,
  AlertViewRenderer,
  SpinnerViewRenderer,
  UserNavigationEvent {

  override func viewDidLoad() {
    super.viewDidLoad()
    needHideNavBar = false
    logTransition()
  }
}
```

Результат

Profit?

Результат
Profit?

- Разработка новых фича без оглядки на Legacy

Результат
Profit?

- Разработка новых фича без оглядки на Legacy
- Декомпозиция кода

Результат
Profit?

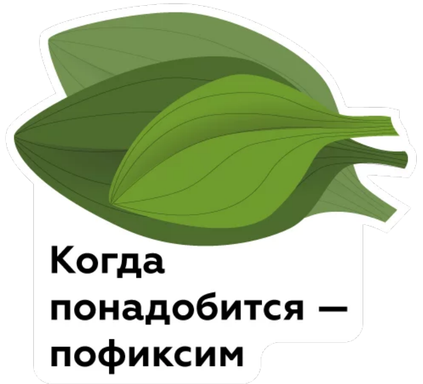
- Разработка новых фича без оглядки на Legacy
- Декомпозиция кода
- К релизу в AppStore легаси похудел на 70%

Результат
Profit?

- Разработка новых фича без оглядки на Legacy
- Декомпозиция кода
- К релизу в AppStore легаси похудел на 70%
- Приложение поддерживало две версии API, теперь поддерживает три 🤦

Результат
Profit?

- Разработка новых фича без оглядки на Legacy
- Декомпозиция кода
- К релизу в AppStore легаси похудел на 70%
- Приложение поддерживало две версии API, теперь поддерживает три 🤦
- Много техдолга



Выводы

- С любым наследием можно работать
- Разработка это бизнес - учитывай его интересы
- Придерживаемся советов Майкла Физерса

Заключение

- Способы работы с легаси
- Что и когда переписывать
- Не боимся легаси кода в проекте
- Улучшаем код проекта с легаси

Конец

Сергей Митрофанов

@ G0retZ

G0retZ

Владимир Шутов

@ shunegus

ShuNegus