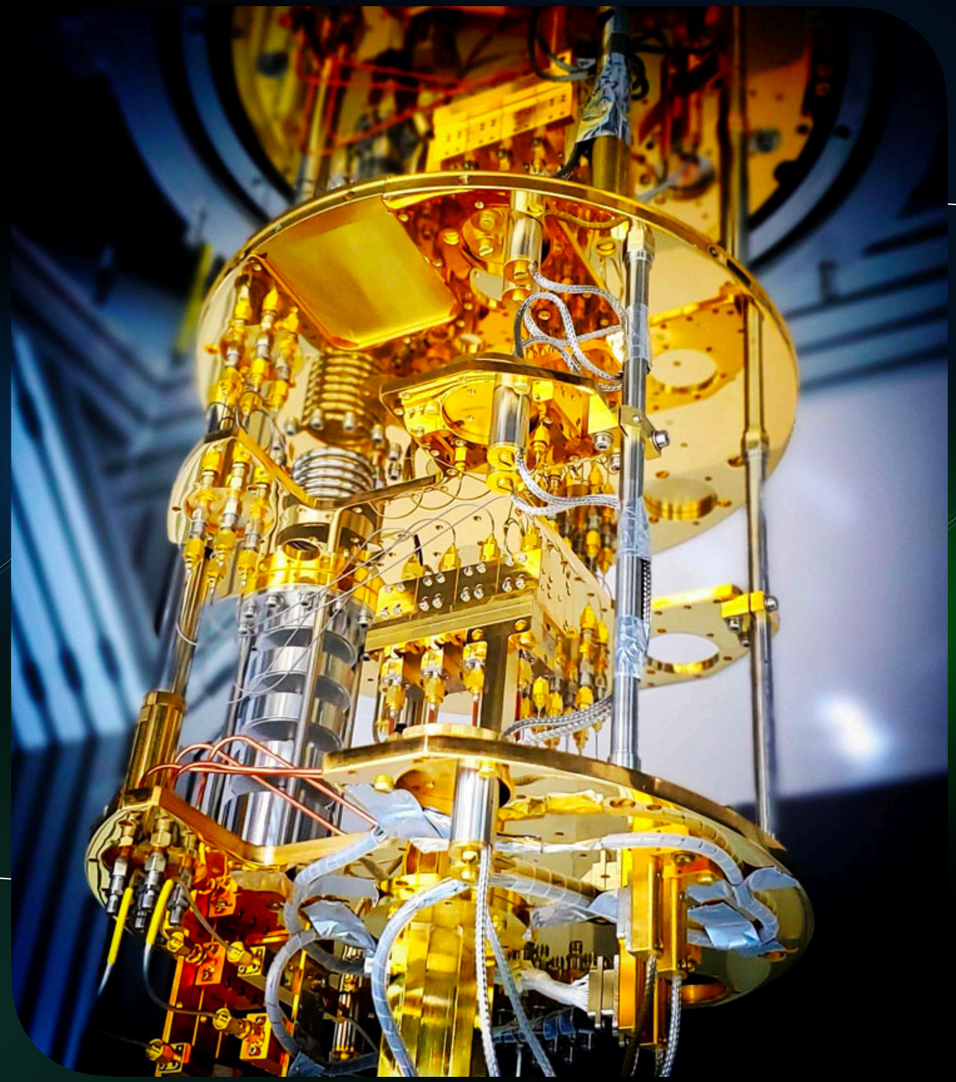
СБЕР

SBOL Story

Владимир Озеров

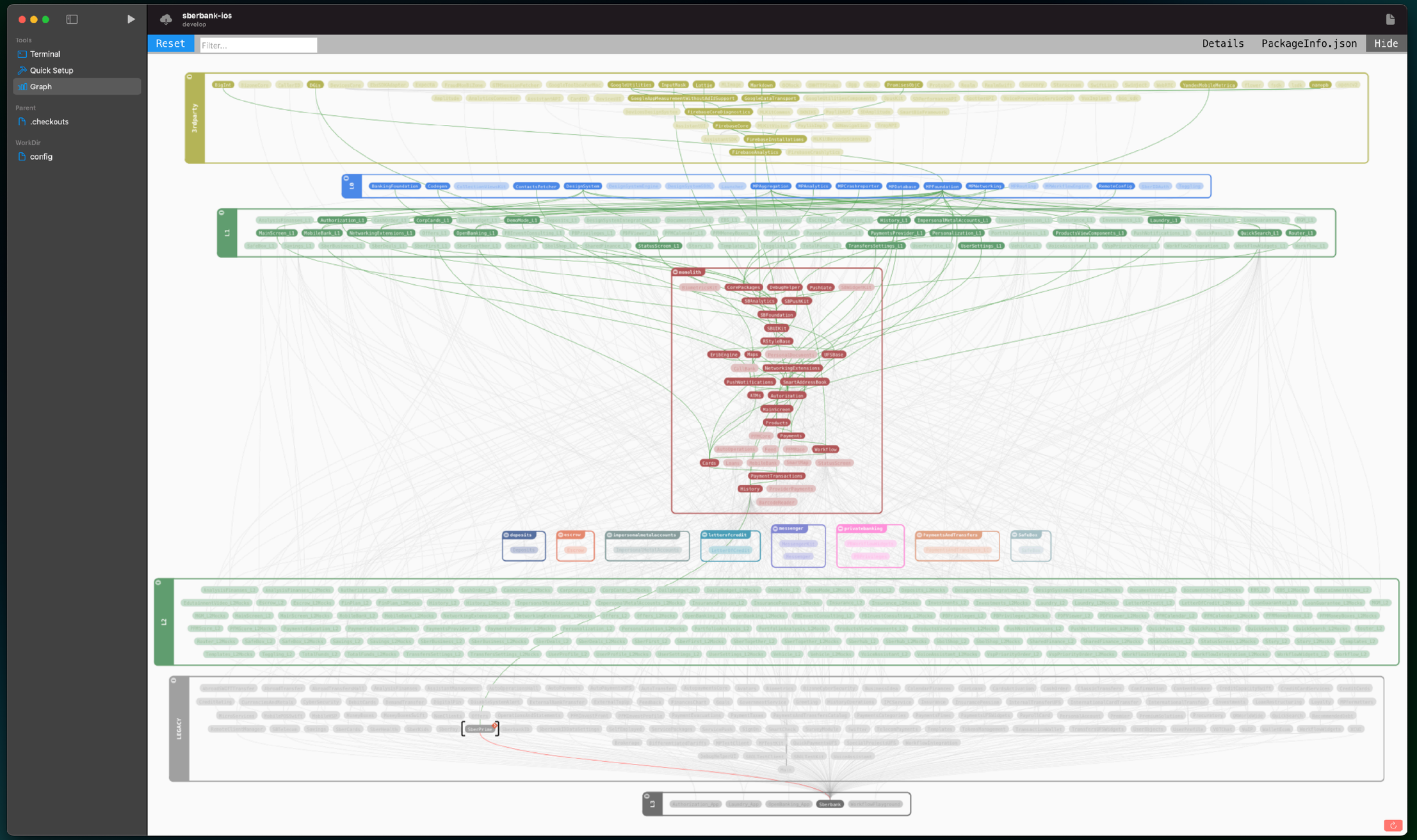

CITIZEN
SDC-810NR
10-DIGIT
2 POWER
M
MRC
M−
M+
ON
CE·C
%
00
.
+
=

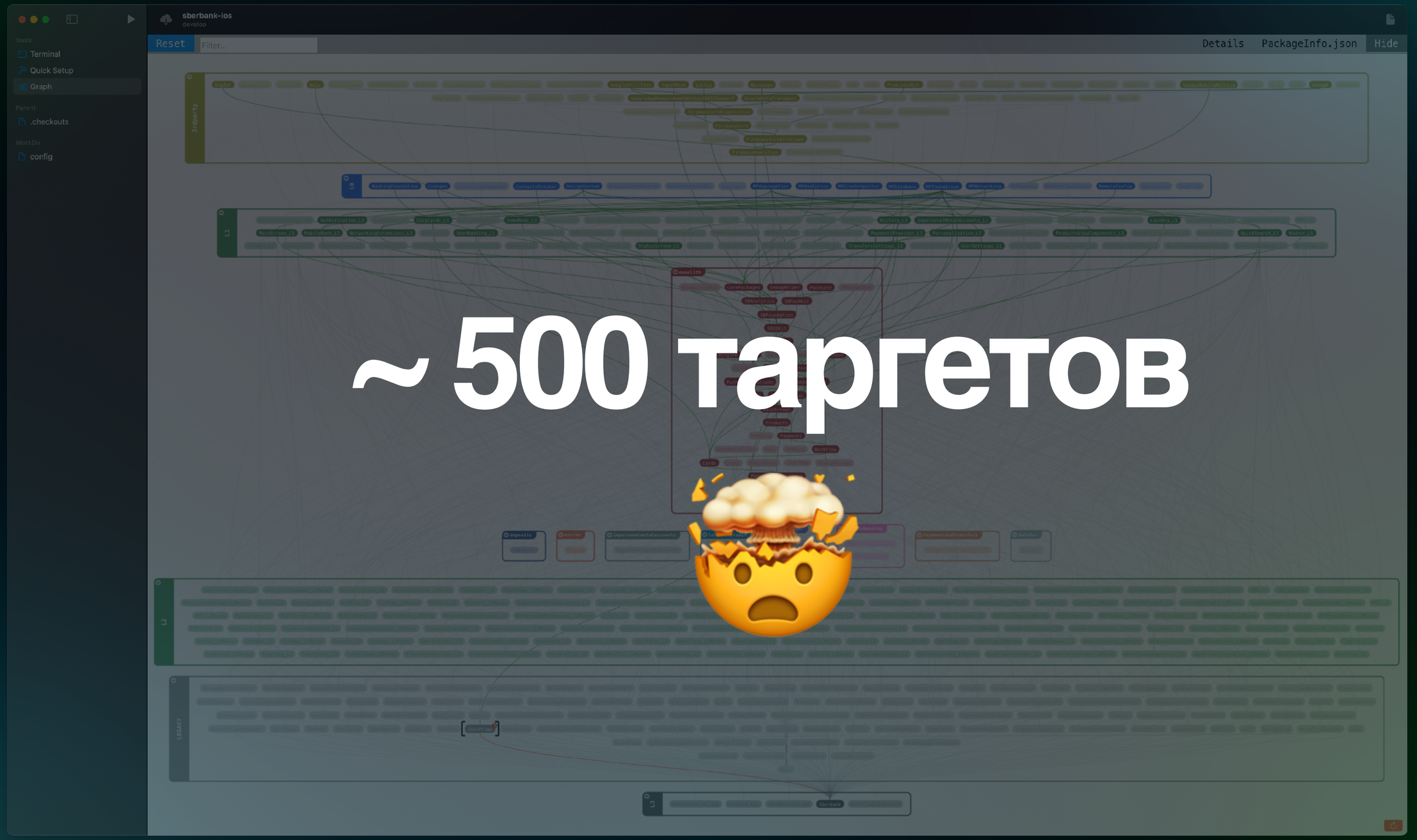

CITIZEN
SDC-810NR
10-DIGIT
2 POWER
M
MRC
M−
M+
ON
CE·C




CITIZEN
SDC-810NR
10-DIGIT
2 POWER
M
12345678900
MRC
M−
M+
+/−
ON
CE·C
%
×
÷
−
+
=
7
8
9
4
5
6
1
2
3
0
00
•

?


sberbank-ios
develop
Tools
Terminal
Quick Setup
Graph
Parent
.checkouts
WorkDir
config
Reset
Filter...
Details
PackageInfo.json
Hide
3rdparty
L0
L1
L2
monolith
deposits
escrow
impersonalmetalaccounts
letterofcredit
messenger
privatebanking
PaymentsAndTransfers
SafeBox
LEGACY
L3
Sberbank


sberbank-ios
develop
Reset
Filter...
Details
PackageInfo.json
Hide
Terminal
Quick Setup
Graph
.checkouts
config
~ 500 таргетов

О чем?

Как развивался проект на протяжении **2011** - 2023?

Какие вызовы вставали перед командой?

Как мы решали эти задачи?

Что стояло за этими решениями, какая **идея**?

Как эти решения проявили себя в перспективе?

СБЕР

Глава 1

Стихийная разработка

2011 - 2016

```
$ git log --reverse

commit fd497c9f4dae7ab35481a98349db7e7e9b2e4106
Author: Ivan Petrov <petrov.ivan@gmail.com>
Date:   Sun Dec 25 15:43:31 2011 +0400

    * 373-ий коммит в SVN.
```

OMG

2011 - 2016

Команда до 12 разработчиков

Монолитная архитектура

Отсутствие CI / CD автоматизации

Ручное тестирование

Релиз 3-4 раза в год

Ретроспектива

- Дух стартапа

- Необходимость знания **всего** проекта
- Низкая **культура** разработки
- Перед каждым релизом полноценный **регресс** (~3 недели)
- Постоянные **кранчи**

1. **Enterprise** – проект требует высокого уровня автоматизации и культуры разработки

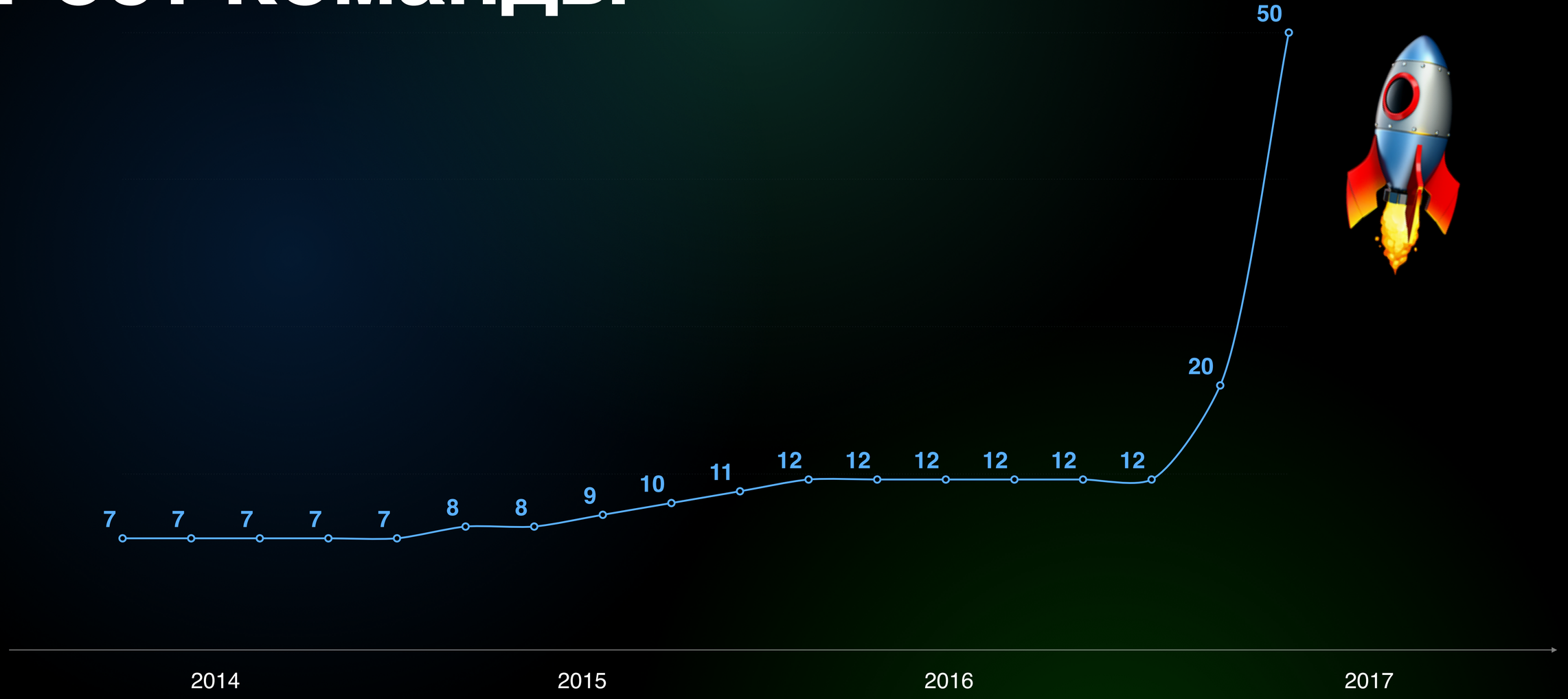
СБЕР

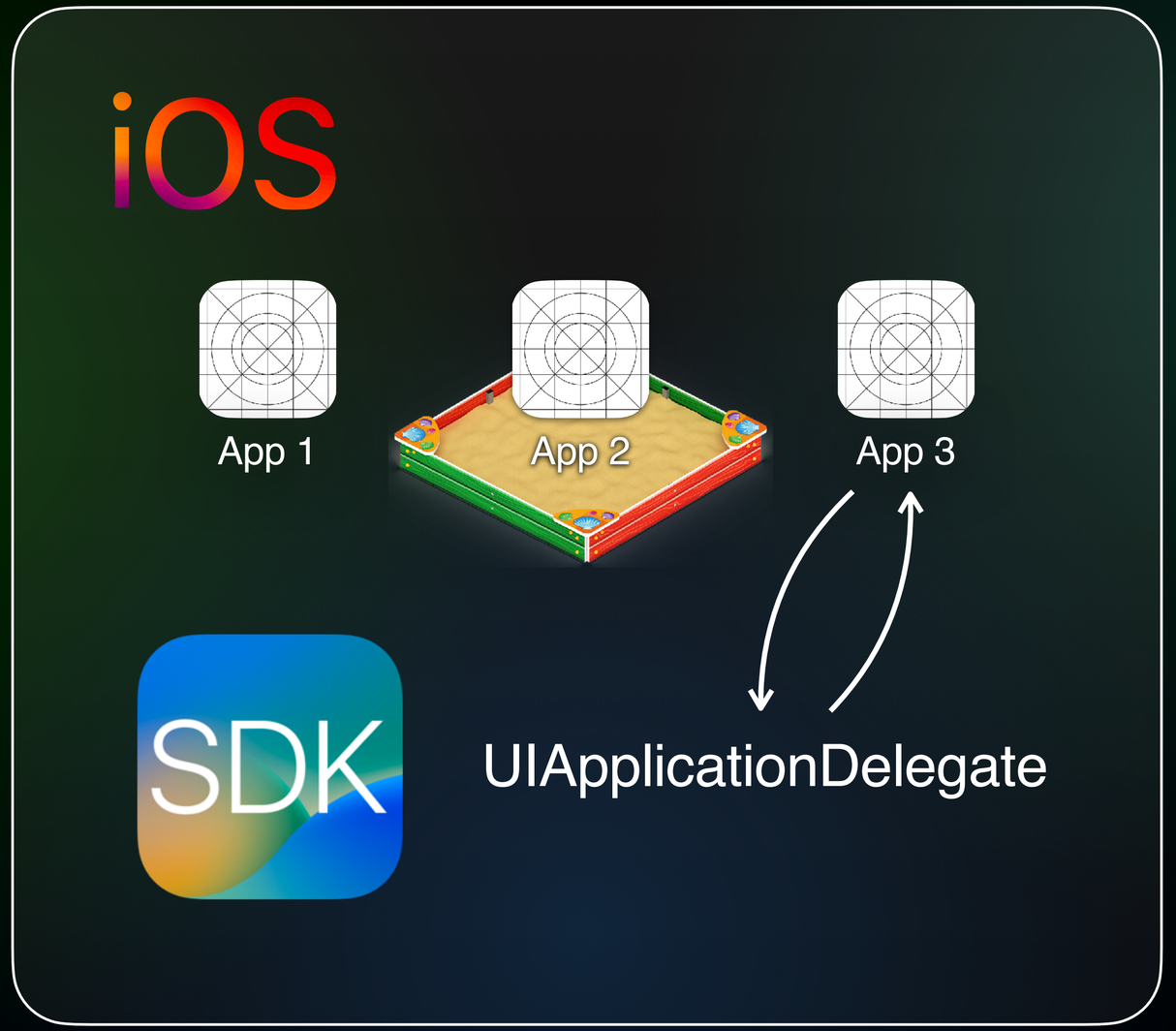
Глава 2

Разделение проекта на модули

2016 - 2017

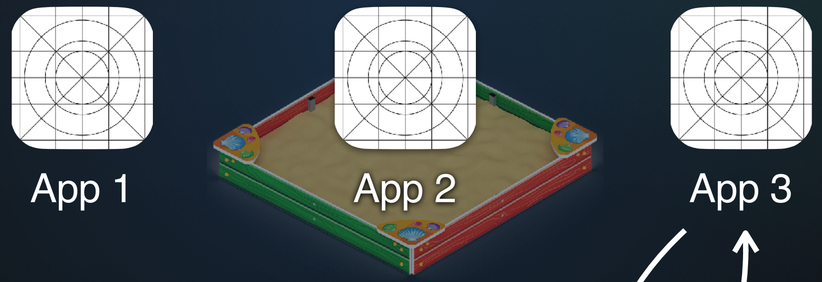
Рост команды




iOS
App 1
App 2
App 3
SDK
UIApplicationDelegate

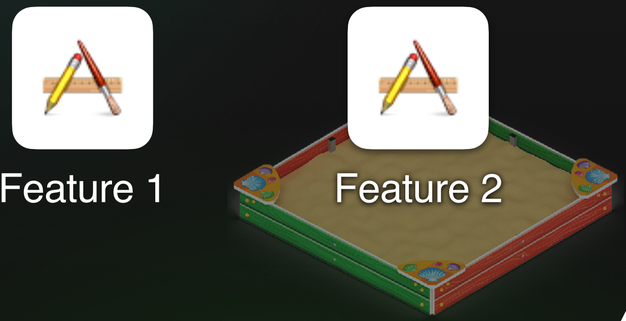
Модульная архитектура

iOS



SDK

UIApplicationDelegate



SDK

SBFAppDelegate

+ SBFoundation

А если чего-то не хватает?

iOS 👀

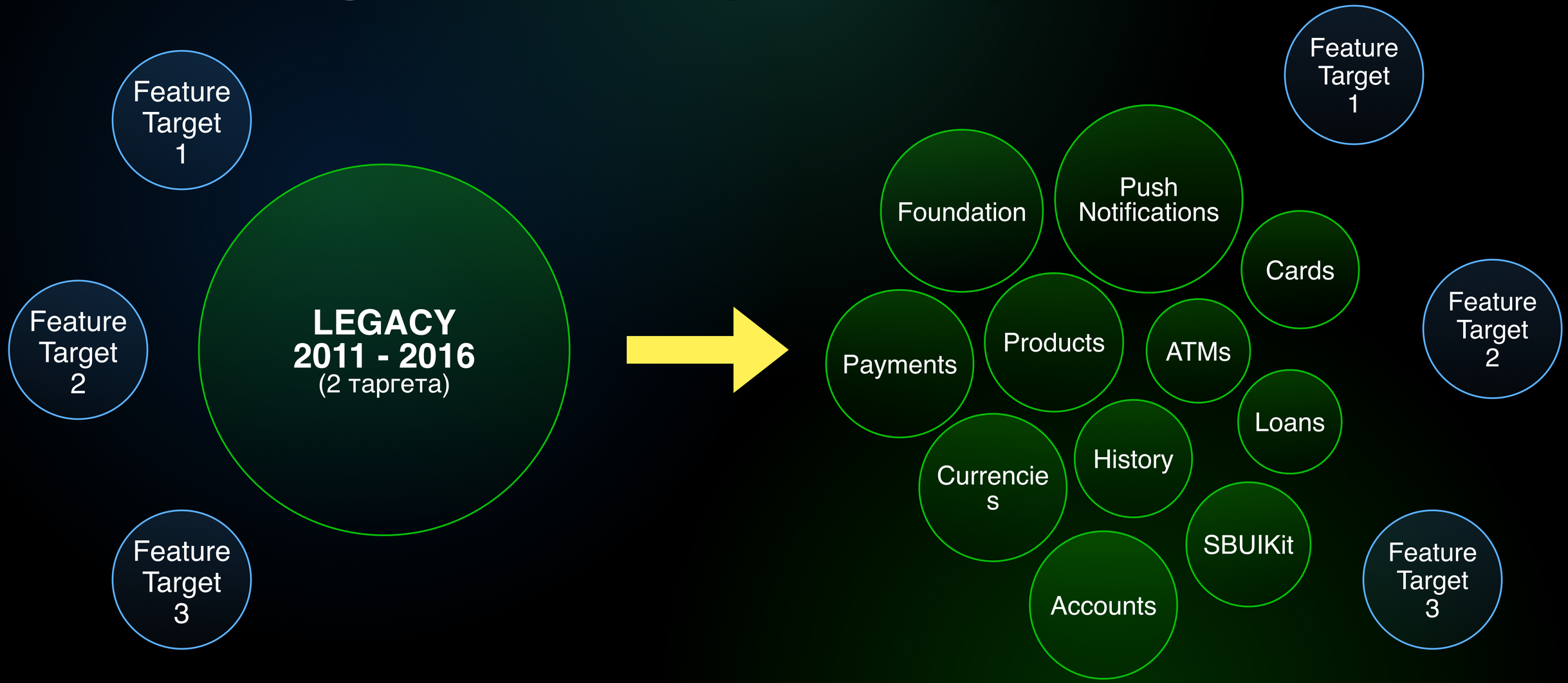
Ретроспектива

- Новые разработчики не обязаны изучать весь **Lagacy**
- Разработка ведется изолированно в **Feature**-таргетах

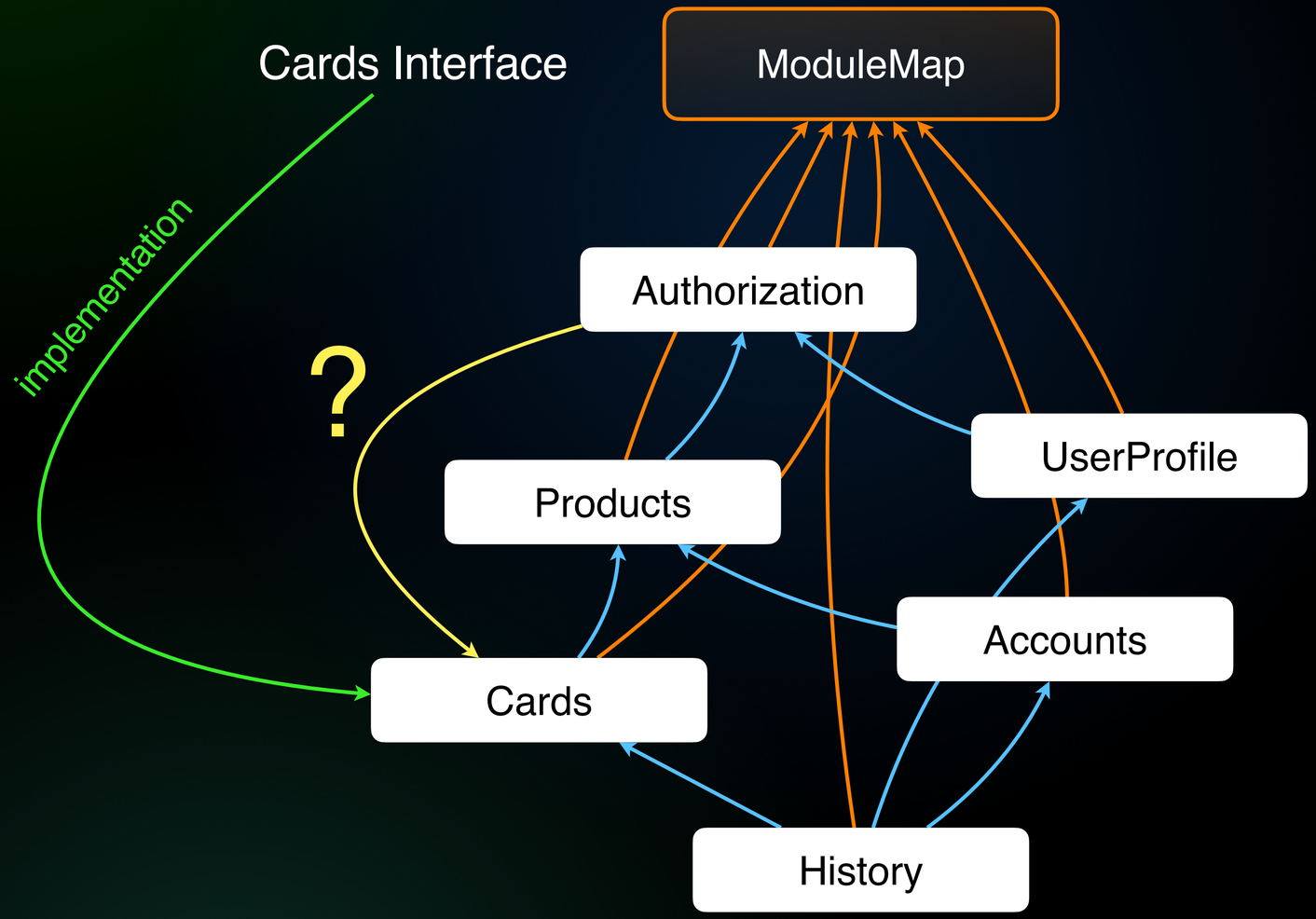
- Каждый может встроиться в **DidFinishLaunching**
- **Sandbox** невозможен из-за прямого доступа к **SDK**
- Аналогия с **iOS** не всегда работает (программный интерфейс **Feature**-таргета)

1. **Enterprise**-проект требует высокого уровня автоматизации и культуры разработки
2. При решении задачи рассматривайте несколько **детально проработанных** вариантов решений

Лучшее - враг хорошего



Супер-зависимость



Ретроспектива

- Получили первый механизм межмодульного взаимодействия

- Супер-зависимость
- **ModuleMap** превратился в свалку, частые конфликты
- **Легаси до сих пор в том же состоянии**

1. **Enterprise** – проект требует высокого уровня автоматизации и культуры разработки
2. При решении задачи рассматривайте несколько **детально проработанных** вариантов решений
3. Избегайте глобального рефакторинга **Legacy**-кода

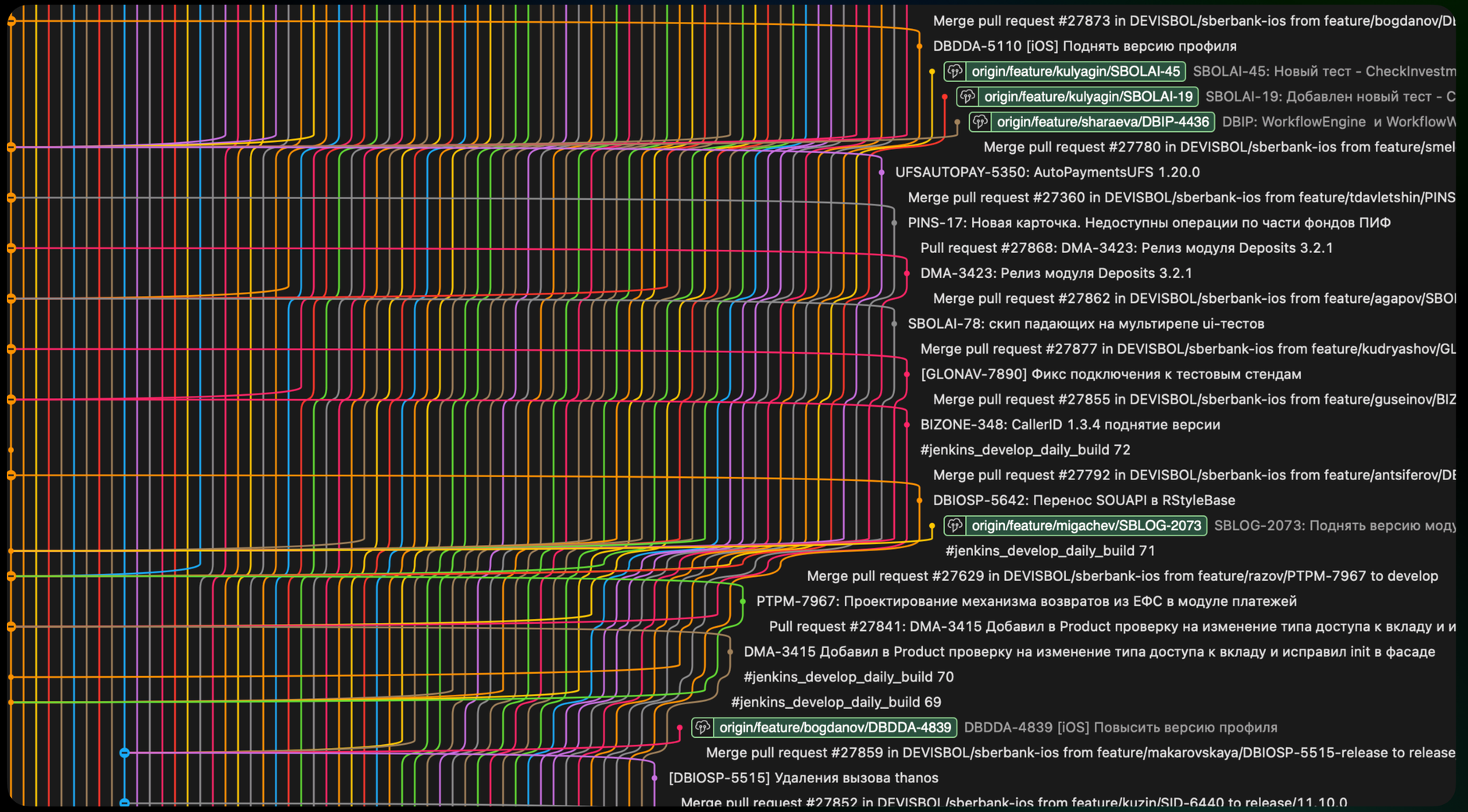
СБЕР

Глава 3

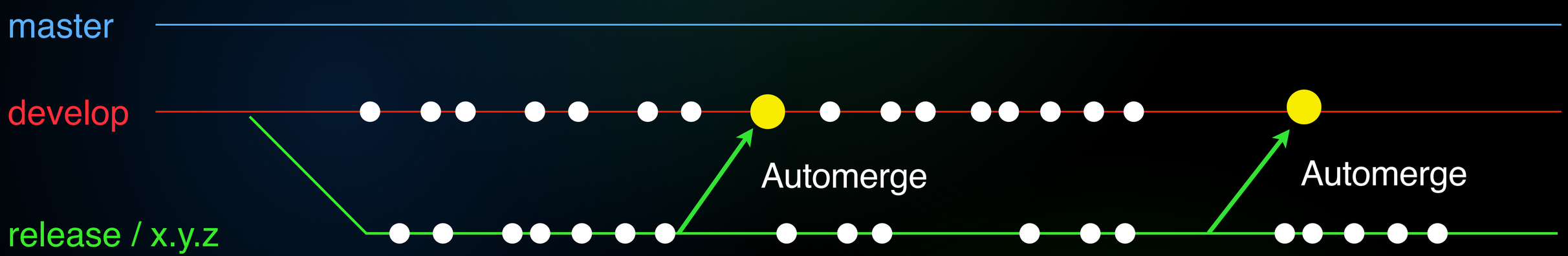
Мультирепозиторий

2017 - 2018

Возникшие боли с Git



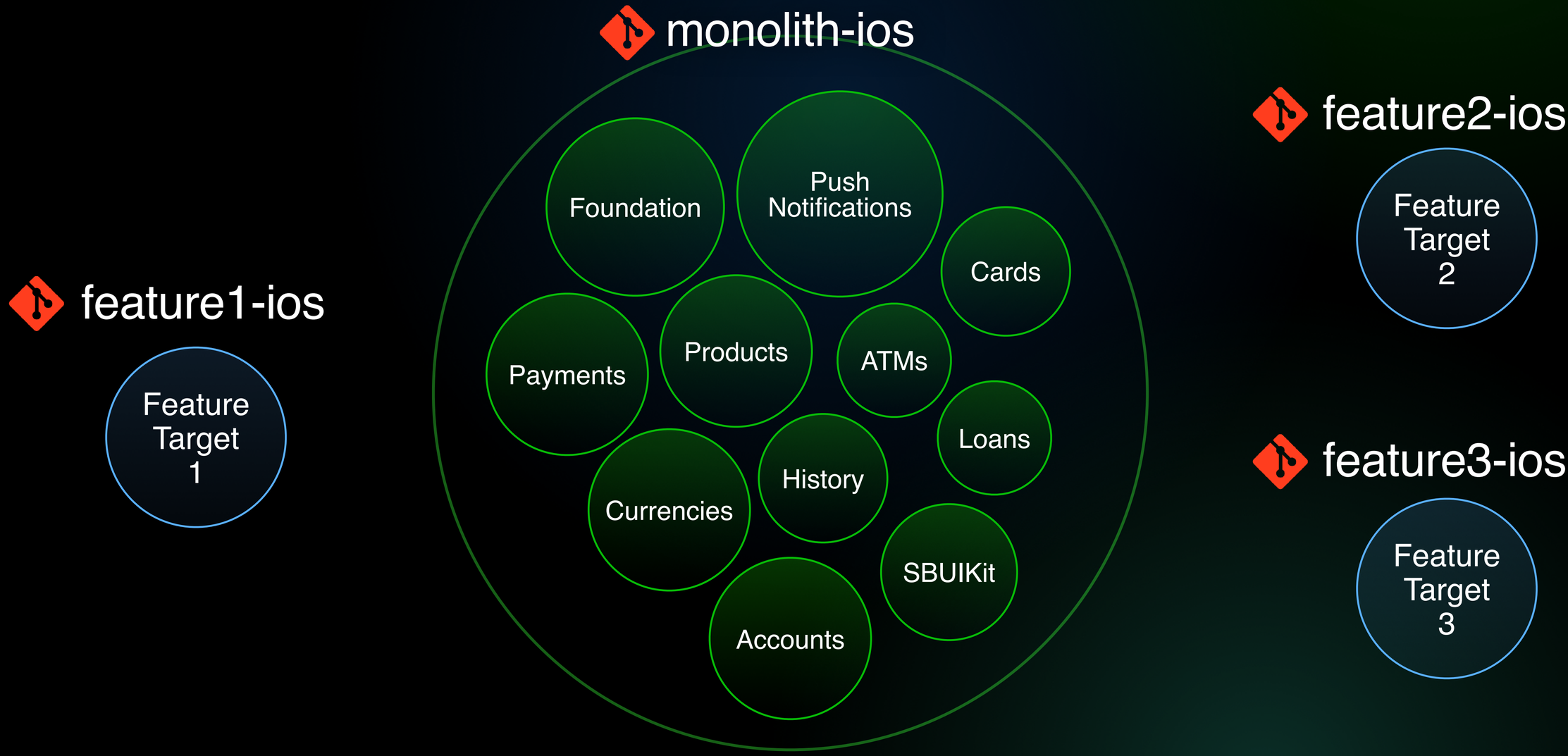
Возникшие боли с Git



Кто решает конфликты?


0000
99
PUSH START
MONO-REPO
MULTI-REPO
FIGHT!

Мультирепозиторий 1.0



Начальник хвалит, что я не распилил Legacy на репозитории

Я, у которого не хватило на это времени

Интеграция репозиториев

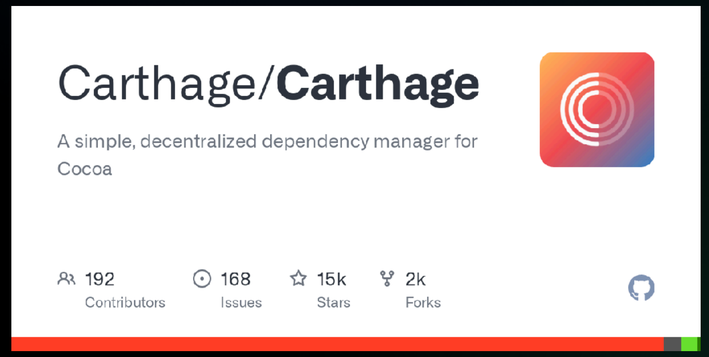
featureX-ios

1.5.6

build & upload xcframework

featureY-ios

featureX = 1.5.6



Ретроспектива

- Решили проблему долгого **Fetch**
- Решили проблему **Automerge**
- 🔥 Реализовали **Remote Build Cache**

- Усложнили **рефакторинг**
- Усложнили работу над проектом
- 2 PullRequest на влитие фичи
- Потеряли часть **Git-истории**

1. **Enterprise** – проект требует высокого уровня автоматизации и культуры разработки
2. При решении задачи рассматривайте несколько **детально проработанных** вариантов решений
3. Избегайте глобального рефакторинга **Legacy**-кода
4. Применяйте конкретные инструменты с четким пониманием их положительных / негативных эффектов

СБЕР

Глава 4

Критическая масса

2018 - 2019

2019

2.5M
строк кода

150
разработчиков

100
репозиториев

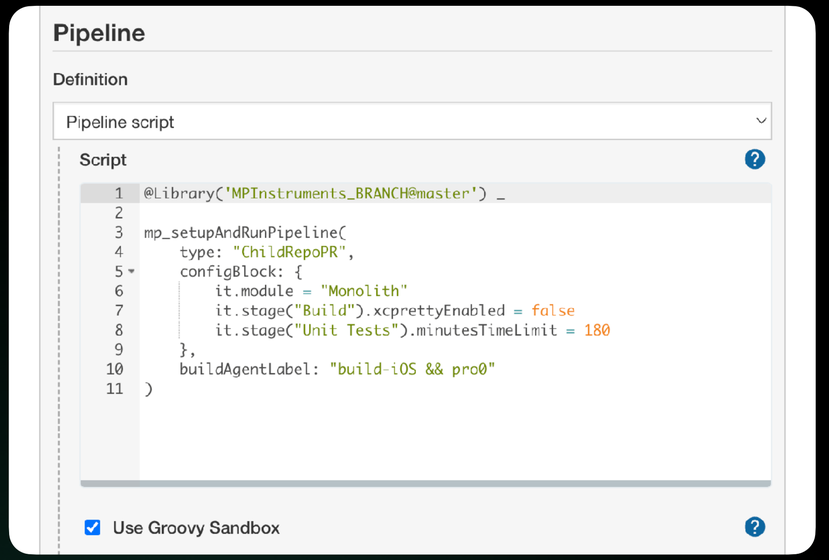
Релизный поезд

Боли

- Dependency Hell (неконтроллируемое дерево зависимостей)
- Поддержка **CI**
  - > 300 пайплайнов (PullRequest, Automerge, Post Merge на каждый репо)
  - сложная логика пайплайнов
  - Boiler-Plate в коде пайплайнов
- Sberthage
  - неоднозначность при выборе binary
  - сложность обновления Sberthage (отдельный релизный процесс)
- Удобство
  - Люди устали делать 2 PR
  - Костыли при внесении зависимых изменений

Рождение 🧬 MP Instruments

- Код инструментов в одном репозитории
  - Python / Ruby / Groovy / bash
- Рефакторинг пайплайнов
  - весь код пайплайнов - в mpinstruments
- Sberthage
  - функционал Carthage не используется → отказываемся от Sberthage
- Команды сидят в своих фичах
  - высокоуровневая организация кода - задача платформы



Ретроспектива

- Единый **репозиторий** инструментов
- Отказ от **Carthage**
- Мини-конфиги в **Jenkins**

- Рефакторинг **= нестабильность**

1. **Enterprise** – проект требует высокого уровня автоматизации и культуры разработки
2. При решении задачи рассматривайте несколько **детально проработанных** вариантов решений
3. Избегайте глобального рефакторинга **Legacy**-кода
4. Применяйте конкретные инструменты с четким пониманием их положительных / негативных эффектов
5. Ведите разработку инструментов в едином репозитории (отдельном от проекта)

СБЕР

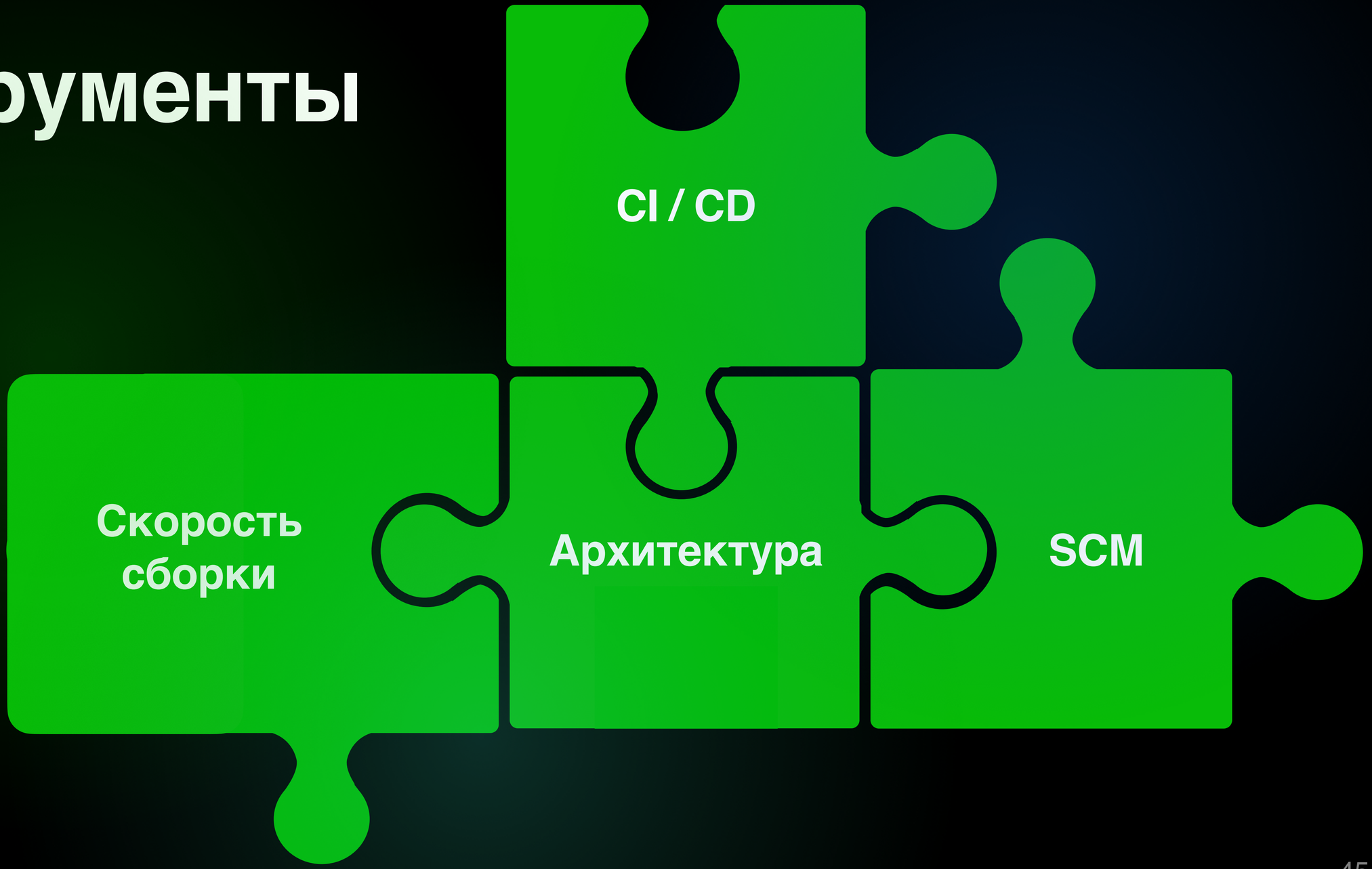
Глава 5

Переосмысление

2020 -

Как все сделать правильно?

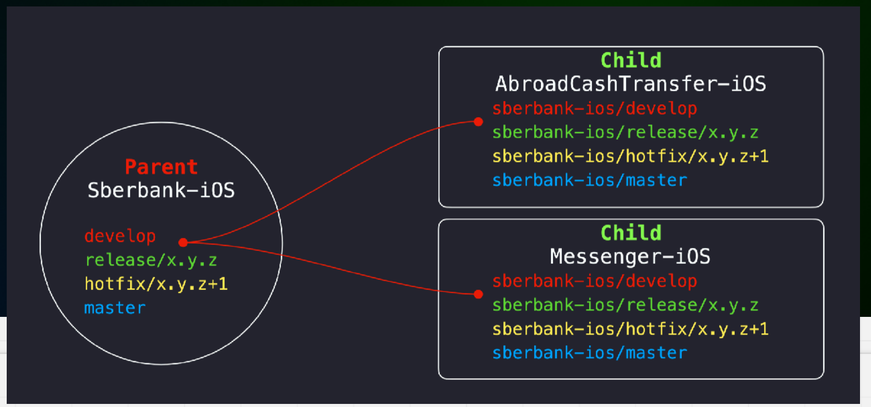
Инструменты



Source Code Management

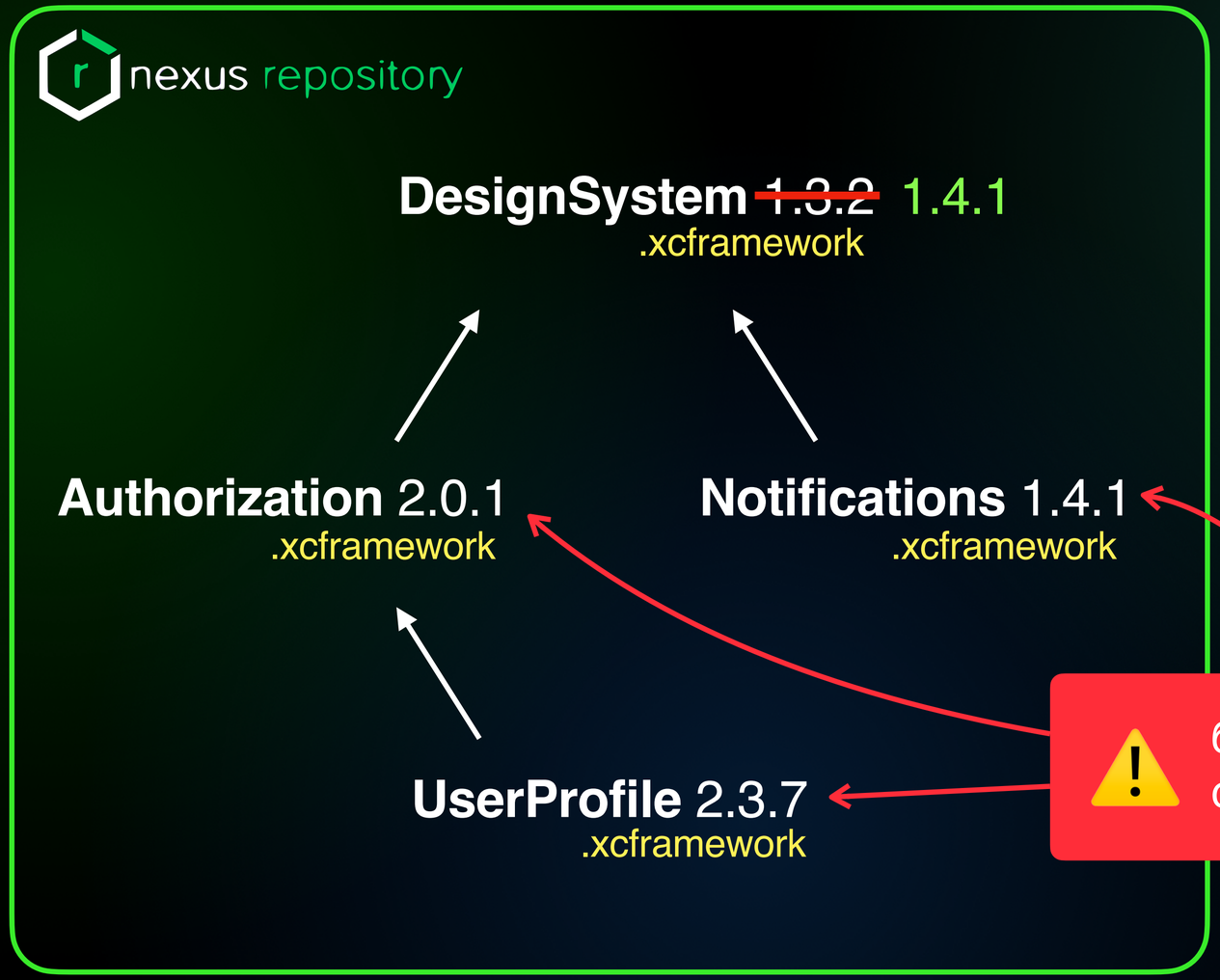
Проблемы

- 1 репозиторий = 1 фреймворк
- Использование **SemVer** для интеграции
- 2 Pull Request на фичу
- Фичи не поддерживают **GitFlow**



Скорость сборки

Build Cache



Поднимать minor у всех?

Просто пересобрать
и перезалить?

Build Cache

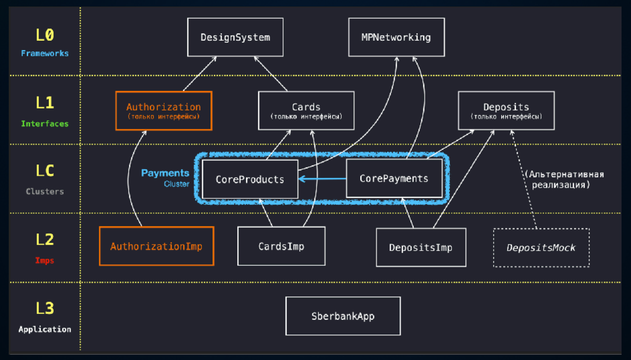
- **Контрольная сумма snapshot_hash**
  - Обратная совместимость зависимостей
  - Параметры компиляции
- **Контрольная сумма pub_hash**
  - Публичный API фреймворка
- **Собственная билд-система**
  - "Умное" обновление Build Cache
  - Импакт анализ при сборке билдов

Пересборка Build Cache

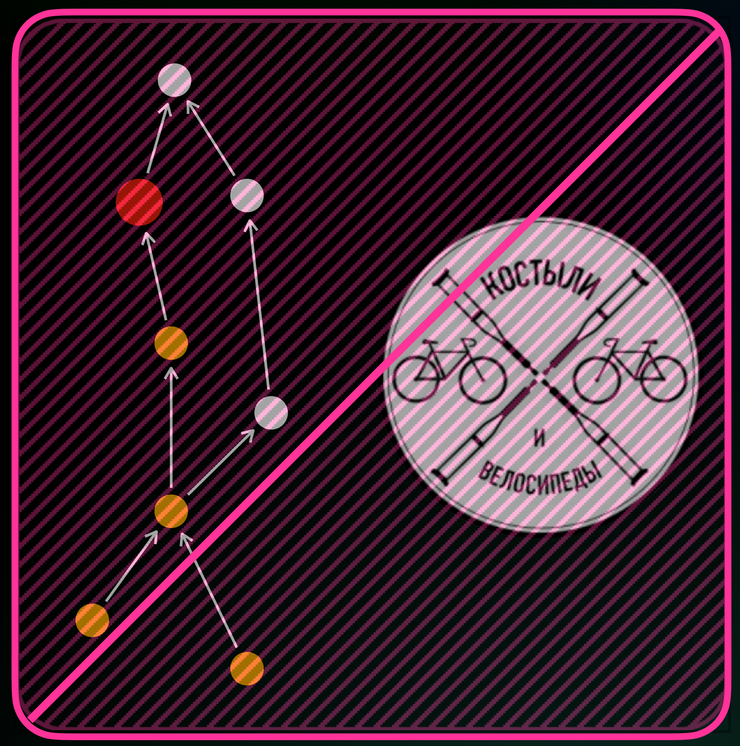
Queued	Init	Checkout	MPMake	Build	Upload	Push
326ms	16s	8s	1min 22s	29min 16s	10s	1min 22s
374ms	17s	10s	1min 36s	44min 27s	1min 12s	2min 25s
251ms	18s	9s	1min 27s	1h 9min	3s	1min 45s
228ms	17s	7s	1min 16s	5min 26s	3s	1min 57s
297ms	13s	6s	1min 12s	1min 48s	2s	13s
277ms	15s	7s	1min 20s	1min 11s	2s	14s
454ms	15s	6s	1min 10s	16min 7s	4s	1min 56s

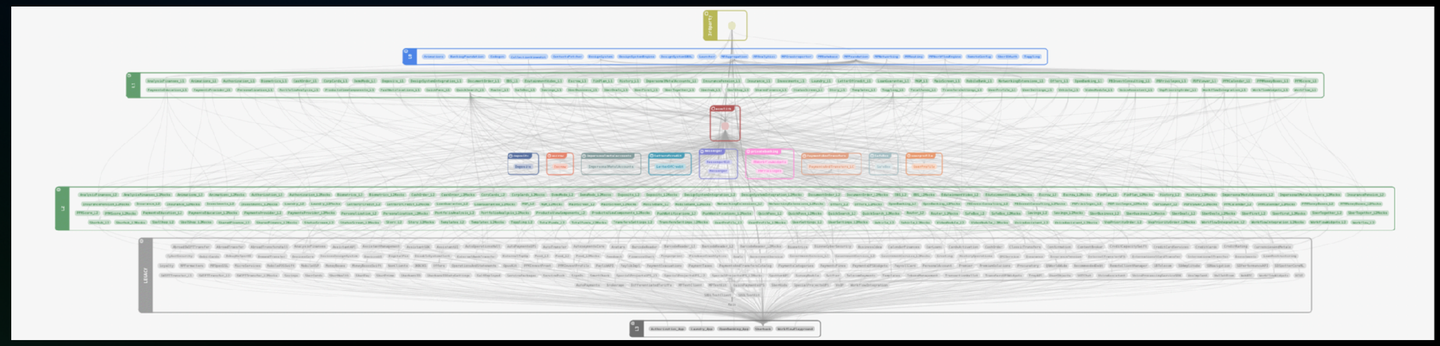
(полная сборка fat binary ~ 1,5 часа на M1 pro)

Архитектура



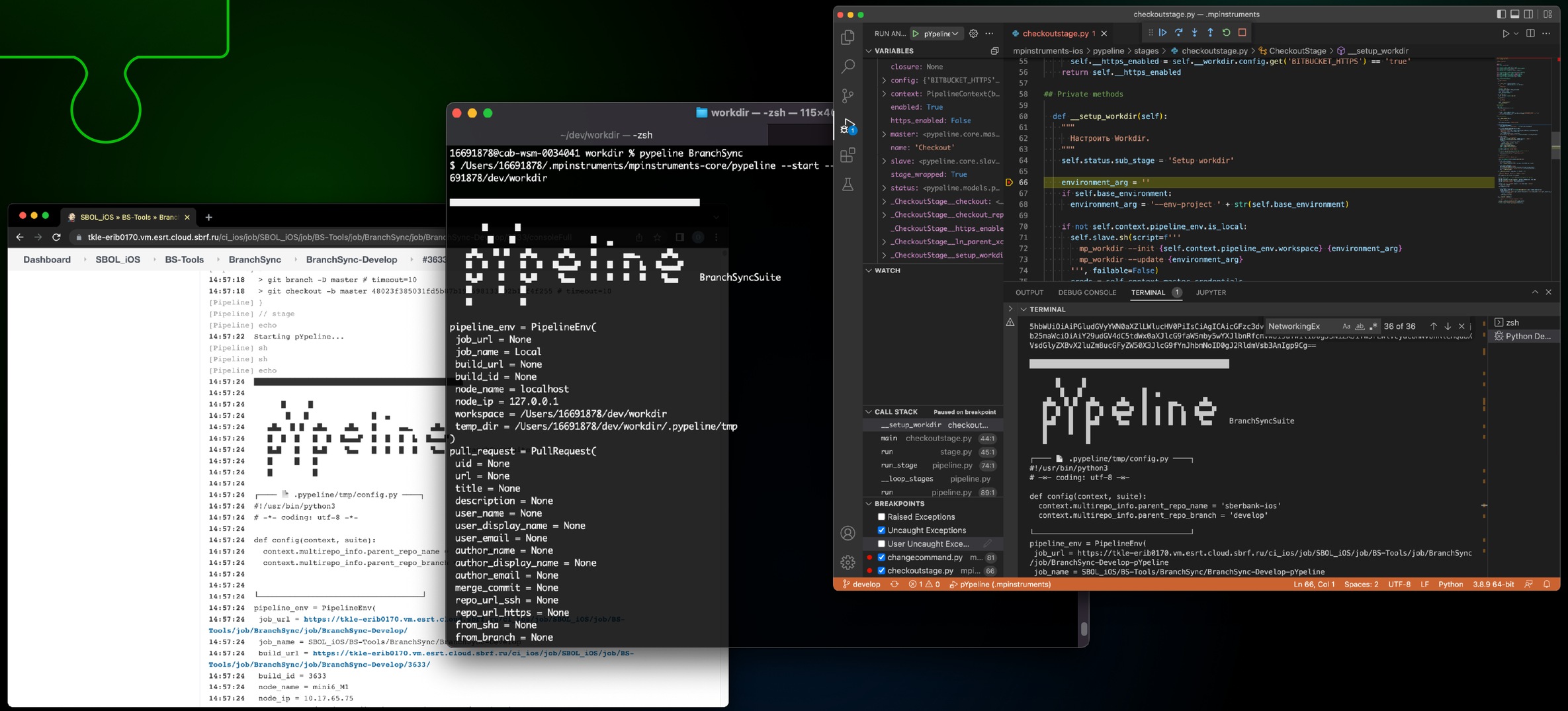
- Граф растет преимущественно горизонтально
- Native межмодульное взаимодействие на Swift





CI / CD

Инструменты для инструментов



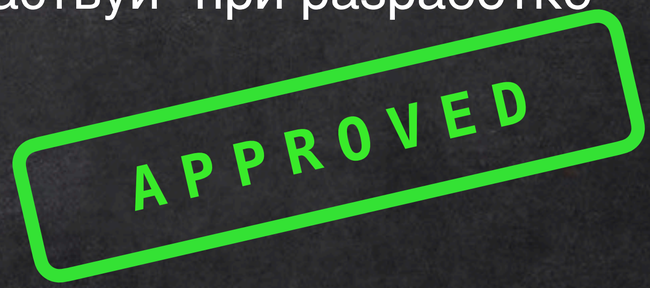

$_
MPDesk

Ретроспектива

- 🔥 **1 Pull Request** на внесение изменений
- 🔥 Новый **Remote Build Cache**
- 🔥 Удобная настройка проекта
- 🔥 "Плоский" **граф** зависимостей
- 🔥 Оптимизированные пайплайны

- Трудозатраты ~ 2 ч/года
- Поживем - увидим :)

1. **Enterprise** – проект требует высокого уровня автоматизации и культуры разработки
2. При решении задачи рассматривайте несколько **детально проработанных** вариантов решений
3. Избегайте глобального рефакторинга **Legacy**-кода
4. Применяйте конкретные инструменты с четким пониманием их положительных / негативных эффектов
5. Ведите разработку инструментов в едином репозитории (отдельном от проекта)
6. Применяйте стратегию "Разделяй и властвуй" при разработке инструментов



stay tuned :)