СБЕР

Переиспользовать нельзя перепроверять:

Баланс между скоростью и надежностью

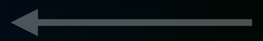
Вероника Макаровская
iOS-MobilePlatform

Легенда

Легенда

1 модуль
Таргет с функционалом, таргеты с unit и UI тестами

Репозиторий
Один стандартный git-репозиторий.



Использование публичного интерфейса
Модуль использует стандартный интерфейс

Публичный интерфейс
Нашего таргета

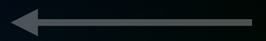
Изменения
Которые мы выносим в наш таргет

Артефакт
Бинарь, бинарник - собранный таргет

Легенда

1 модуль
Таргет с функционалом, таргеты с unit и UI тестами

Репозиторий
Один стандартный git-репозиторий.



Использование публичного интерфейса
Модуль использует стандартный интерфейс

Публичный интерфейс
Нашего таргета

Изменения
Которые мы выносим в наш таргет

Артефакт
Бинарь, бинарник - собранный таргет

Легенда

1 модуль
Таргет с функционалом, таргеты с unit и UI тестами

Репозиторий
Один стандартный git-репозиторий.

Использование публичного интерфейса
Модуль использует стандартный интерфейс

Публичный интерфейс
Нашего таргета

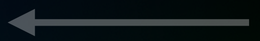
Изменения
Которые мы выносим в наш таргет

Артефакт
Бинарь, бинарник - собранный таргет

Легенда

1 модуль
Таргет с функционалом, таргеты с unit и UI тестами

Репозиторий
Один стандартный git-репозиторий.



Использование публичного интерфейса
Модуль использует стандартный интерфейс

Публичный интерфейс
Нашего таргета

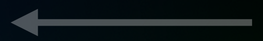
Изменения
Которые мы выносим в наш таргет

Артефакт
Бинарь, бинарник - собранный таргет

Легенда

1 модуль
Таргет с функционалом, таргеты с unit и UI тестами

Репозиторий
Один стандартный git-репозиторий.



Использование публичного интерфейса
Модуль использует стандартный интерфейс

Публичный интерфейс
Нашего таргета

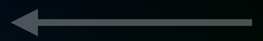
Изменения
Которые мы выносим в наш таргет

Артефакт
Бинарь, бинарник - собранный таргет

Легенда

1 модуль
Таргет с функционалом, таргеты с unit и UI тестами

Репозиторий
Один стандартный git-репозиторий.



Использование публичного интерфейса
Модуль использует стандартный интерфейс

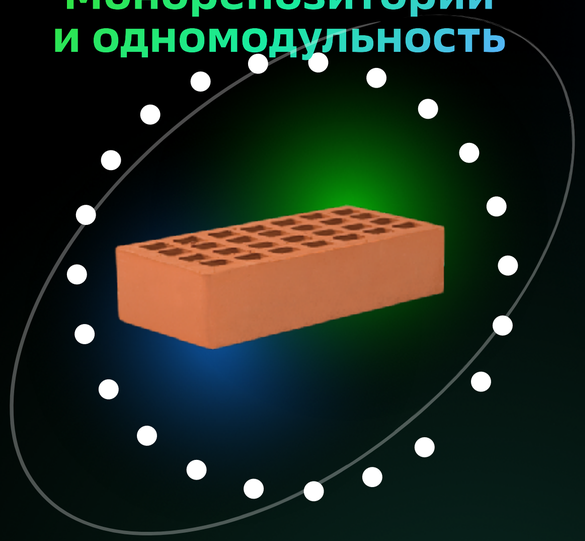
Публичный интерфейс
Нашего таргета

Изменения
Которые мы выносим в наш таргет

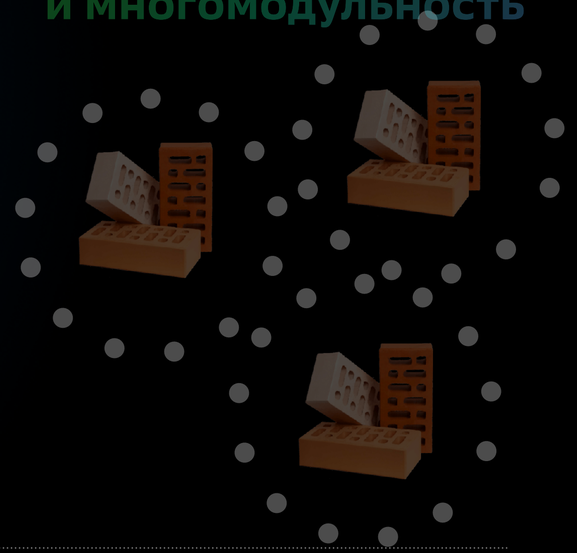
Артефакт
Бинарь, бинарник - собранный таргет

Кому будет полезно и интересно?

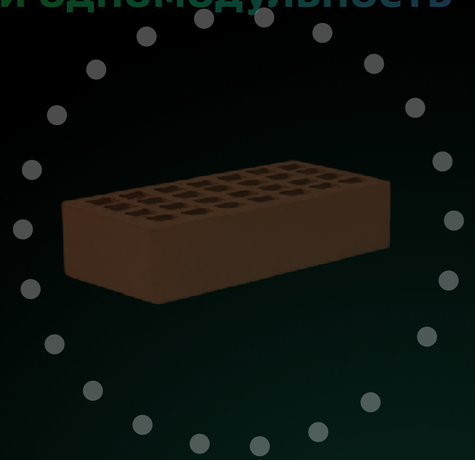
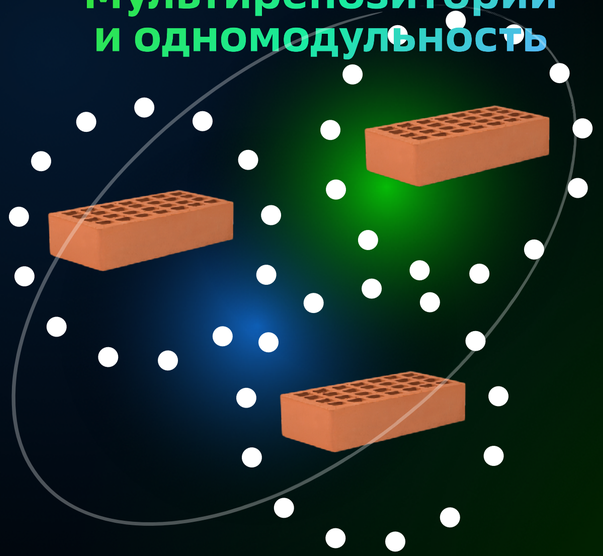
Виды проектов

Монорепозиторий и одномодульность	Монорепозиторий и многомодульность	Мультирепозиторий и одномодульность	Мультирепозиторий и многомодульность
			
1 репозиторий	1 репозиторий	2+ репозитории	2+ репозитории
1 модуль	2+ модулей	По 1 модулю	По 2+ модулей

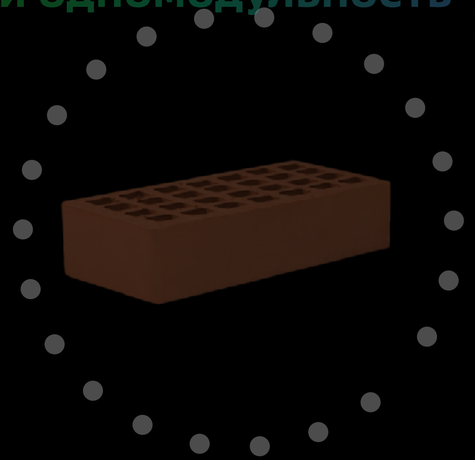
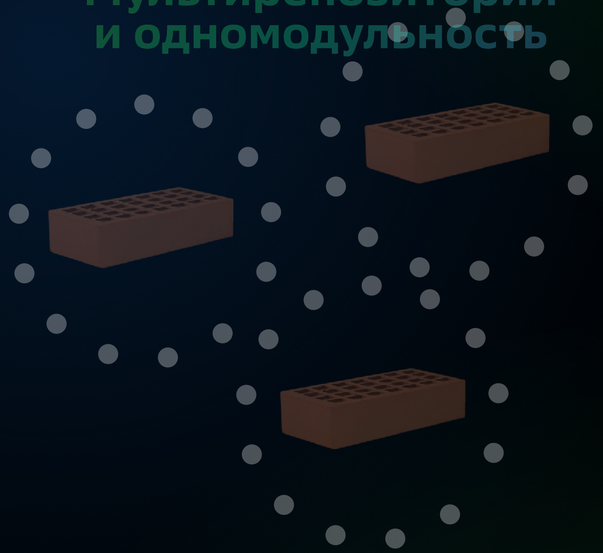
Виды проектов

Монорепозиторий и одномодульность	Монорепозиторий и многомодульность	Мультирепозиторий и одномодульность	Мультирепозиторий и многомодульность
			
1 репозиторий	**1 репозиторий**	2+ репозитории	2+ репозитории
1 модуль	**2+ модулей**	По 1 модулю	По 2+ модулей

Виды проектов

Монорепозиторий и одномодульность	Монорепозиторий и многомодульность	Мультирепозиторий и одномодульность	Мультирепозиторий и многомодульность
			
1 репозиторий	1 репозиторий	**2+ репозитории**	2+ репозитории
1 модуль	2+ модулей	**По 1 модулю**	По 2+ модулей

Виды проектов

Монорепозиторий и одномодульность	Монорепозиторий и многомодульность	Мультирепозиторий и одномодульность	Мультирепозиторий и многомодульность
			
1 репозиторий	1 репозиторий	2+ репозитории	**2+ репозитории**
1 модуль	2+ модулей	По 1 модулю	**По 2+ модулей**

Виды проектов

Виды проектов

«SBOL iOS Story»

Владимир Озеров

Наш план
на ближайший час

Что мы рассмотрим?

- Что такое импакт?

Что мы рассмотрим?

- Что такое импакт?
- Немного предыстории

Что мы рассмотрим?

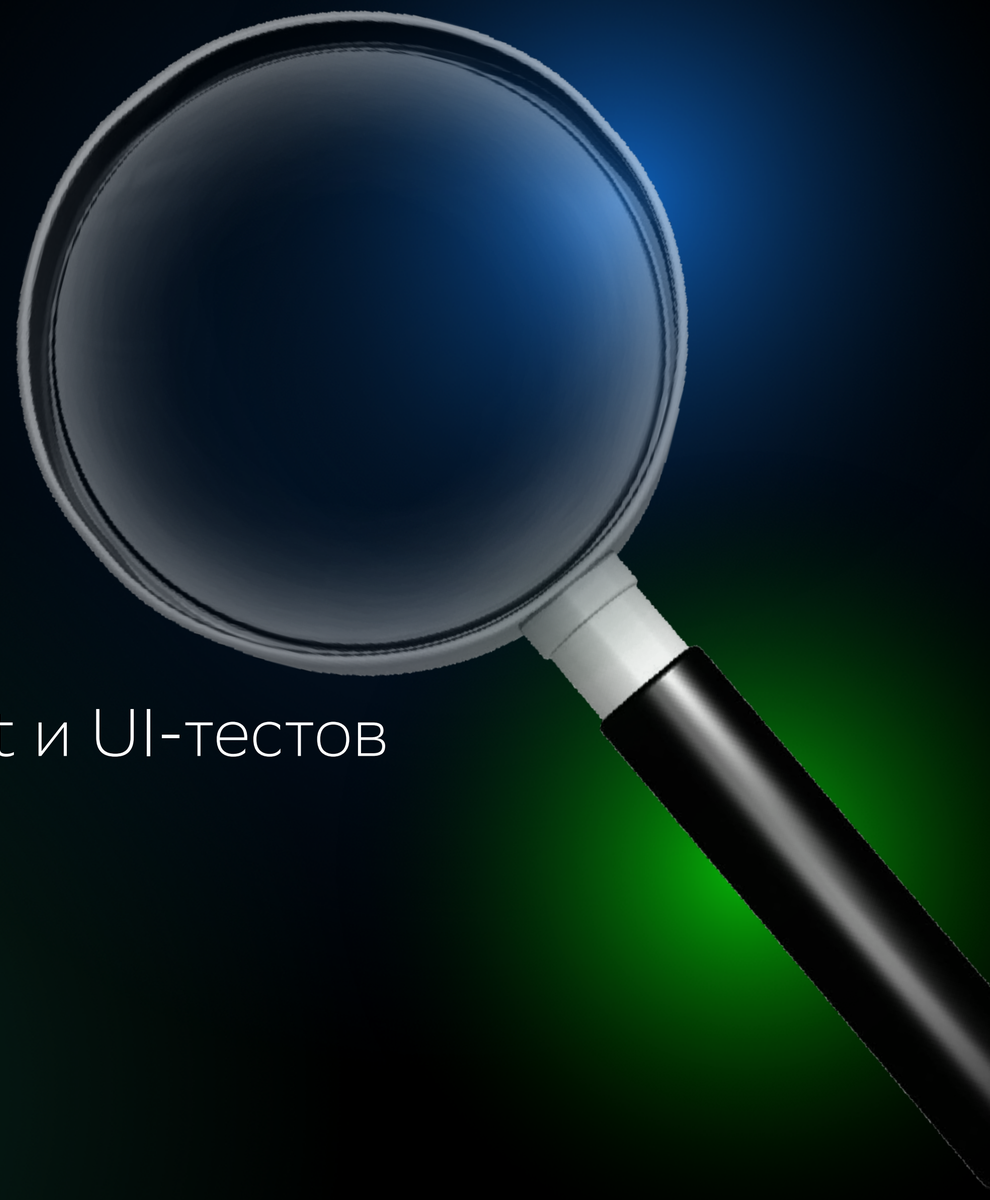
- Что такое импакт?
- Немного предыстории
- Первая эпоха impact-анализа

Что мы рассмотрим?

- Что такое импакт?
- Немного предыстории
- Первая эпоха impact-анализа
- Вторая эпоха impact-анализа

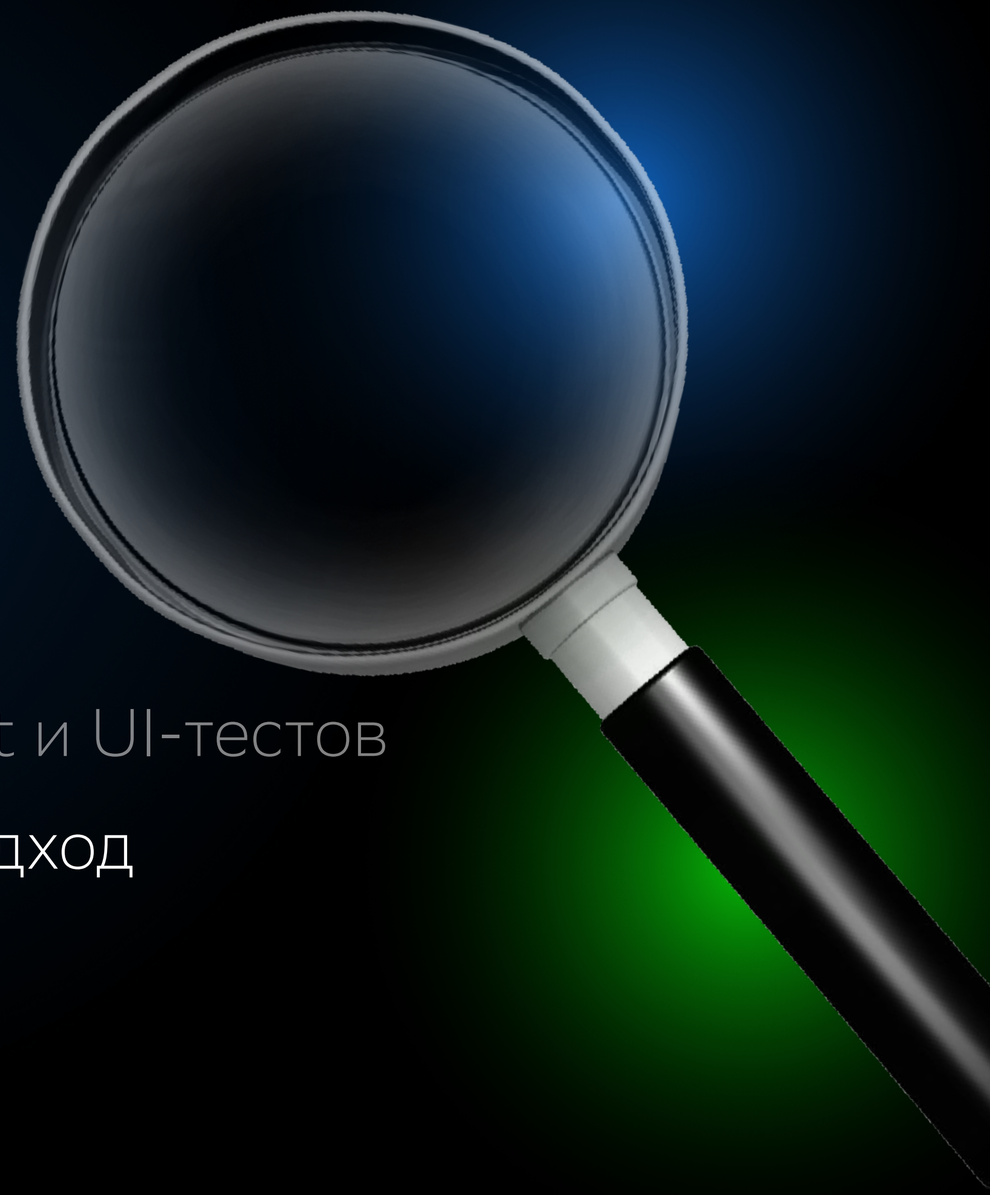
Что мы рассмотрим?

- Что такое импакт?
- Немного предыстории
- Первая эпоха impact-анализа
- Вторая эпоха impact-анализа
- Подключаем impact-анализ к прогону unit и UI-тестов

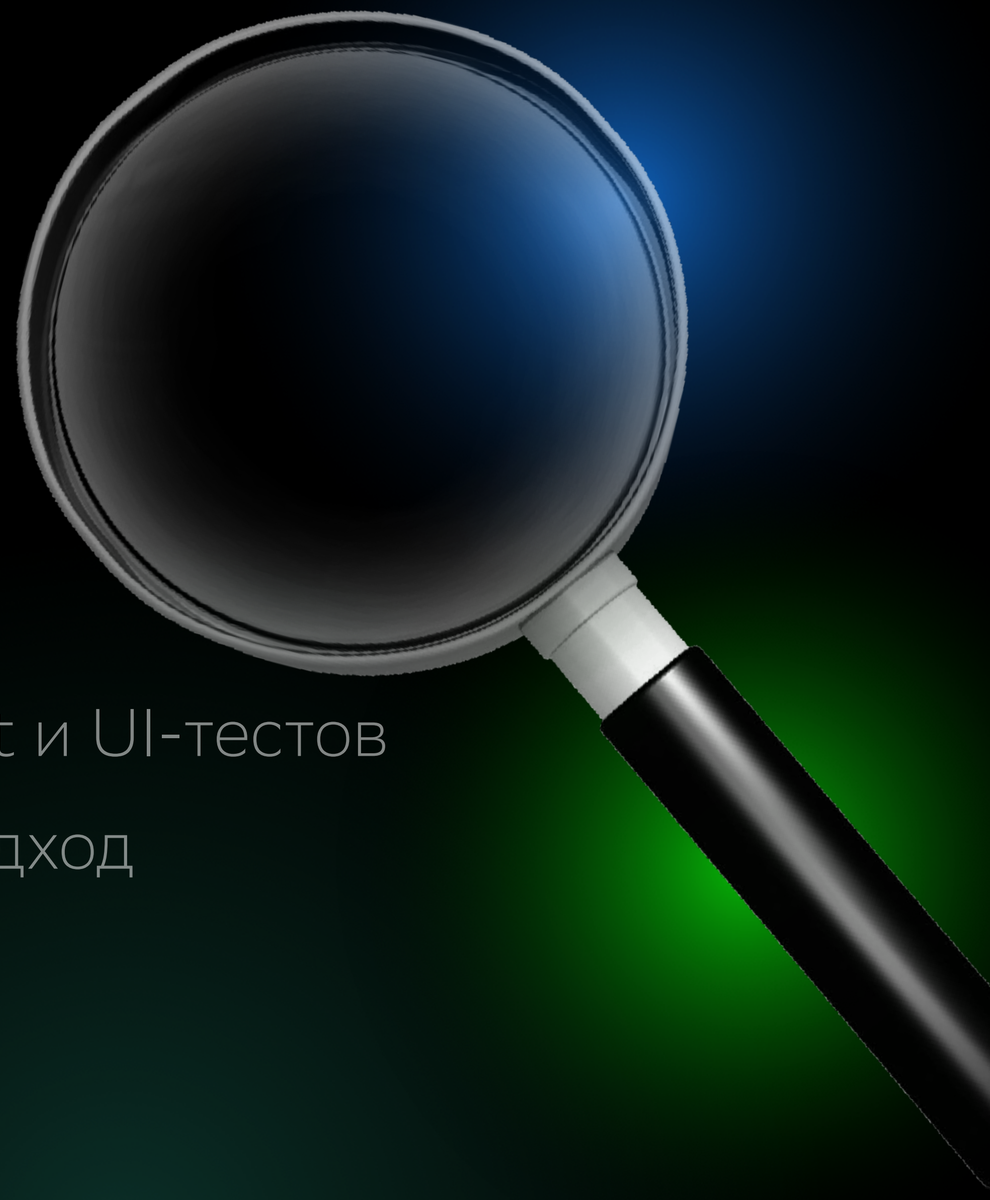


Что мы рассмотрим?

- Что такое импакт?
- Немного предыстории
- Первая эпоха impact-анализа
- Вторая эпоха impact-анализа
- Подключаем impact-анализ к прогону unit и UI-тестов
- Проблемы, которые не учли во второй подход

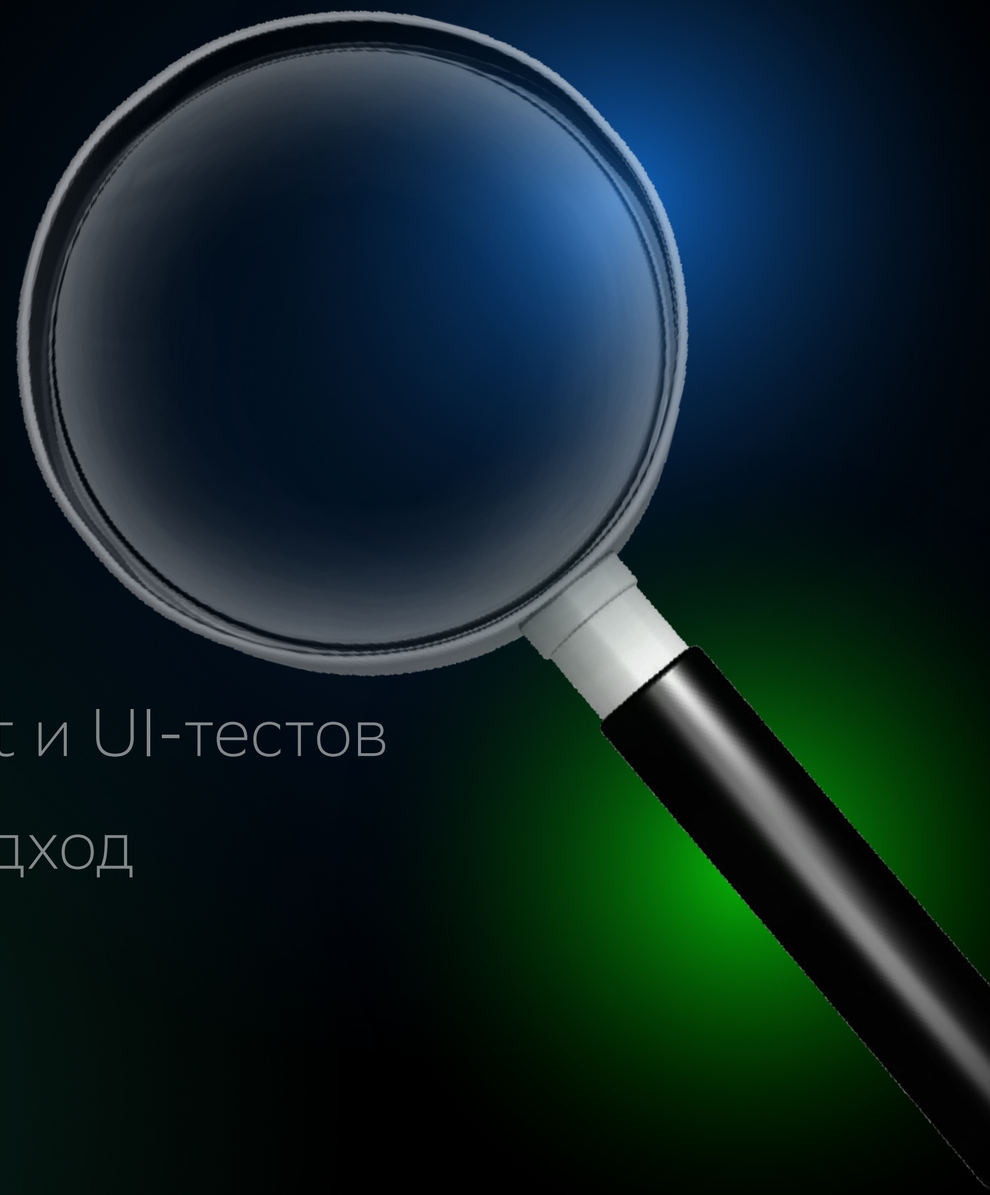


Что мы рассмотрим?



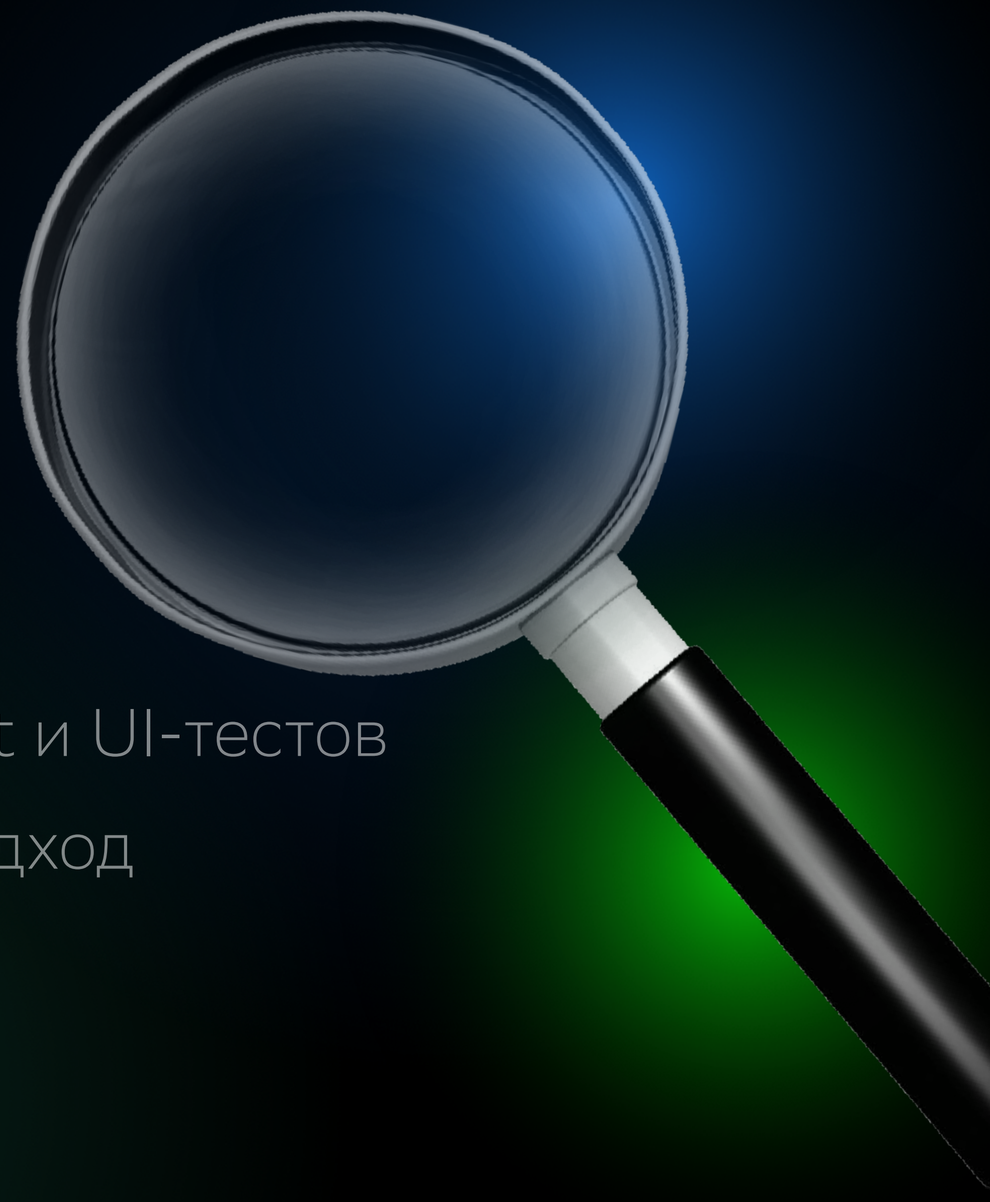
- Что такое импакт?
- Немного предыстории
- Первая эпоха impact-анализа
- Вторая эпоха impact-анализа
- Подключаем impact-анализ к прогону unit и UI-тестов
- Проблемы, которые не учли во второй подход
- Новая эпоха impact-анализа

Что мы рассмотрим?



- Что такое импакт?
- Немного предыстории
- Первая эпоха impact-анализа
- Вторая эпоха impact-анализа
- Подключаем impact-анализ к прогону unit и UI-тестов
- Проблемы, которые не учли во второй подход
- Новая эпоха impact-анализа
- И немного ещё…

Что мы рассмотрим?



- Что такое импакт?
- Немного предыстории
- Первая эпоха impact-анализа
- Вторая эпоха impact-анализа
- Подключаем impact-анализ к прогону unit и UI-тестов
- Проблемы, которые не учли во второй подход
- Новая эпоха impact-анализа
- И немного ещё…
- Итоги

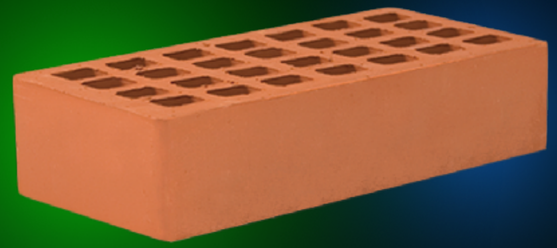
Что такое impact?

Баланс между чёрным и белым:

зачем определять, на что влияет ваш новый код?

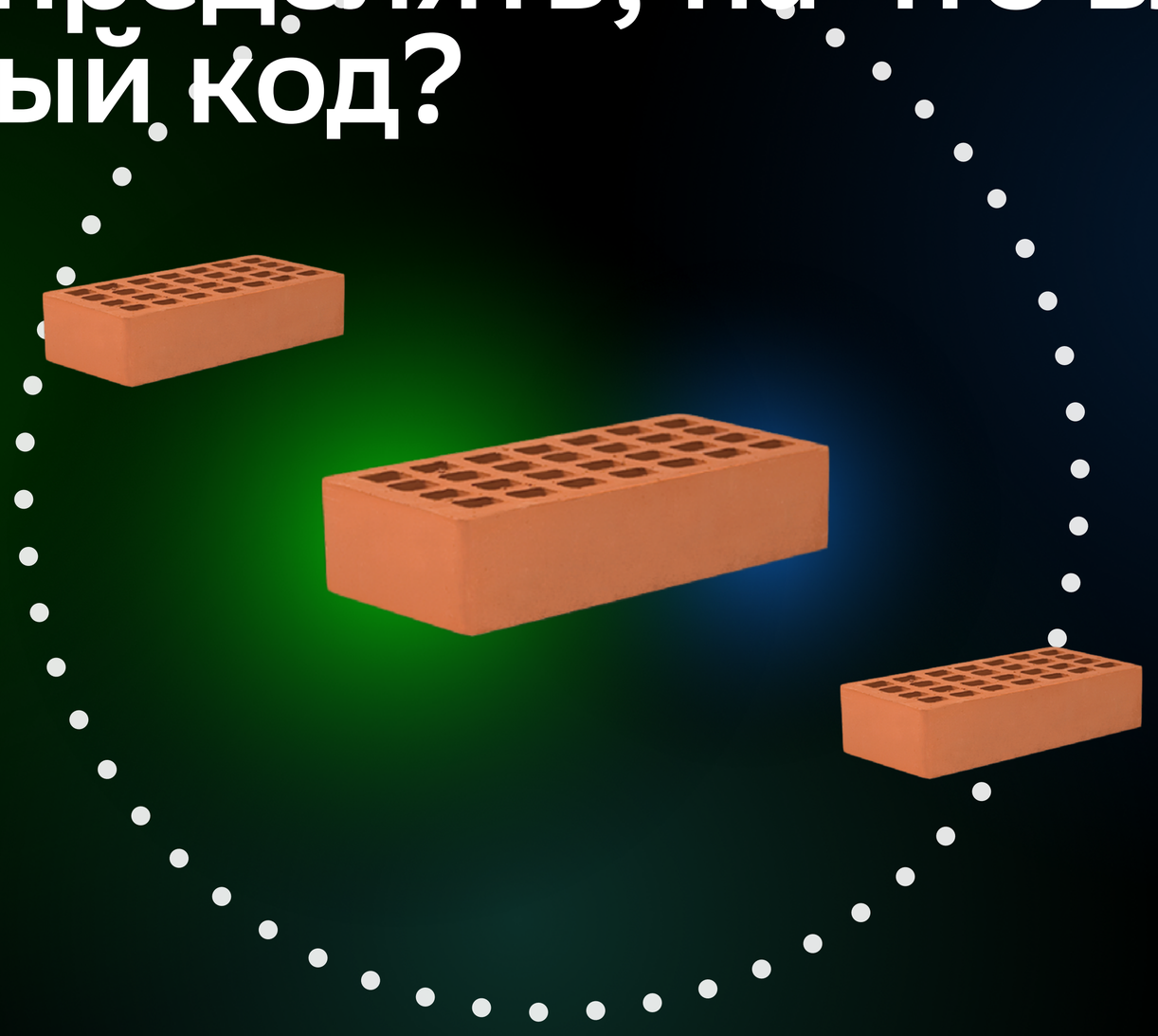
Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?

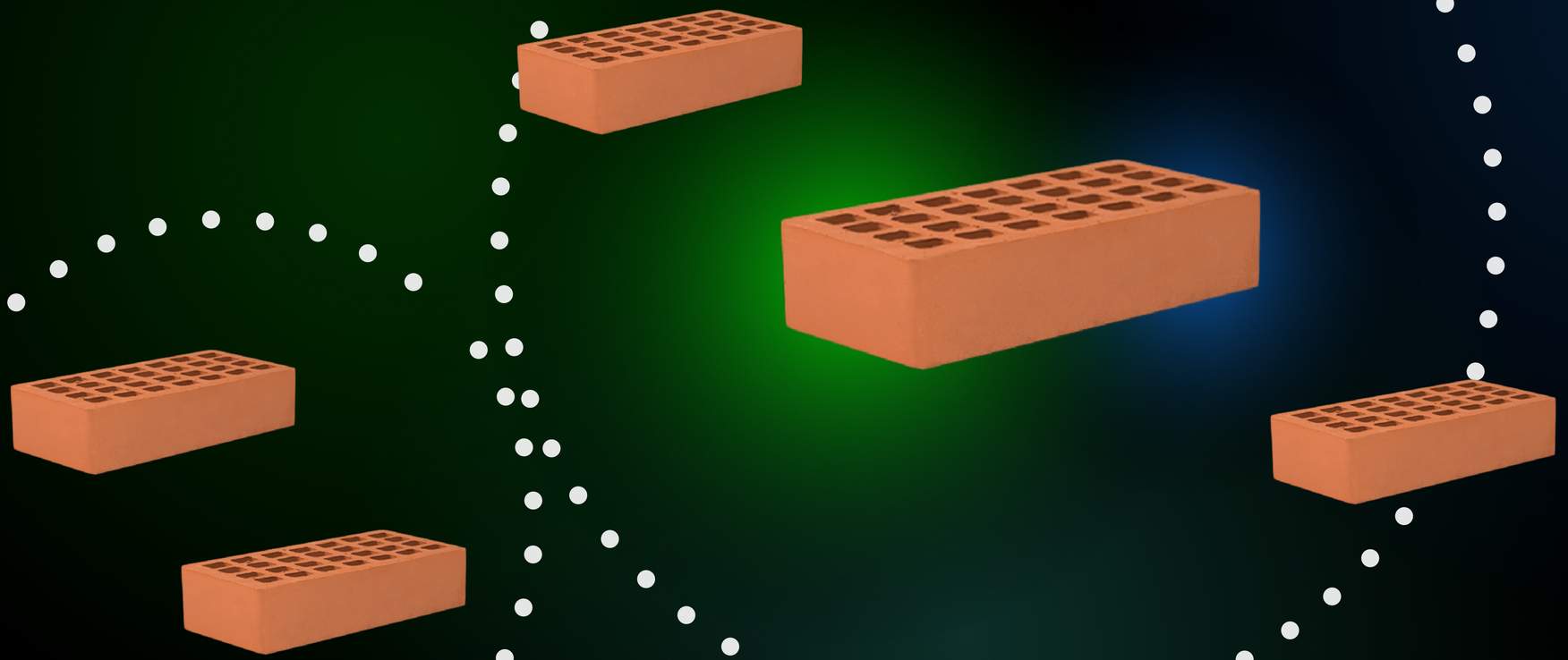


Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?

Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



Баланс между чёрным и белым:

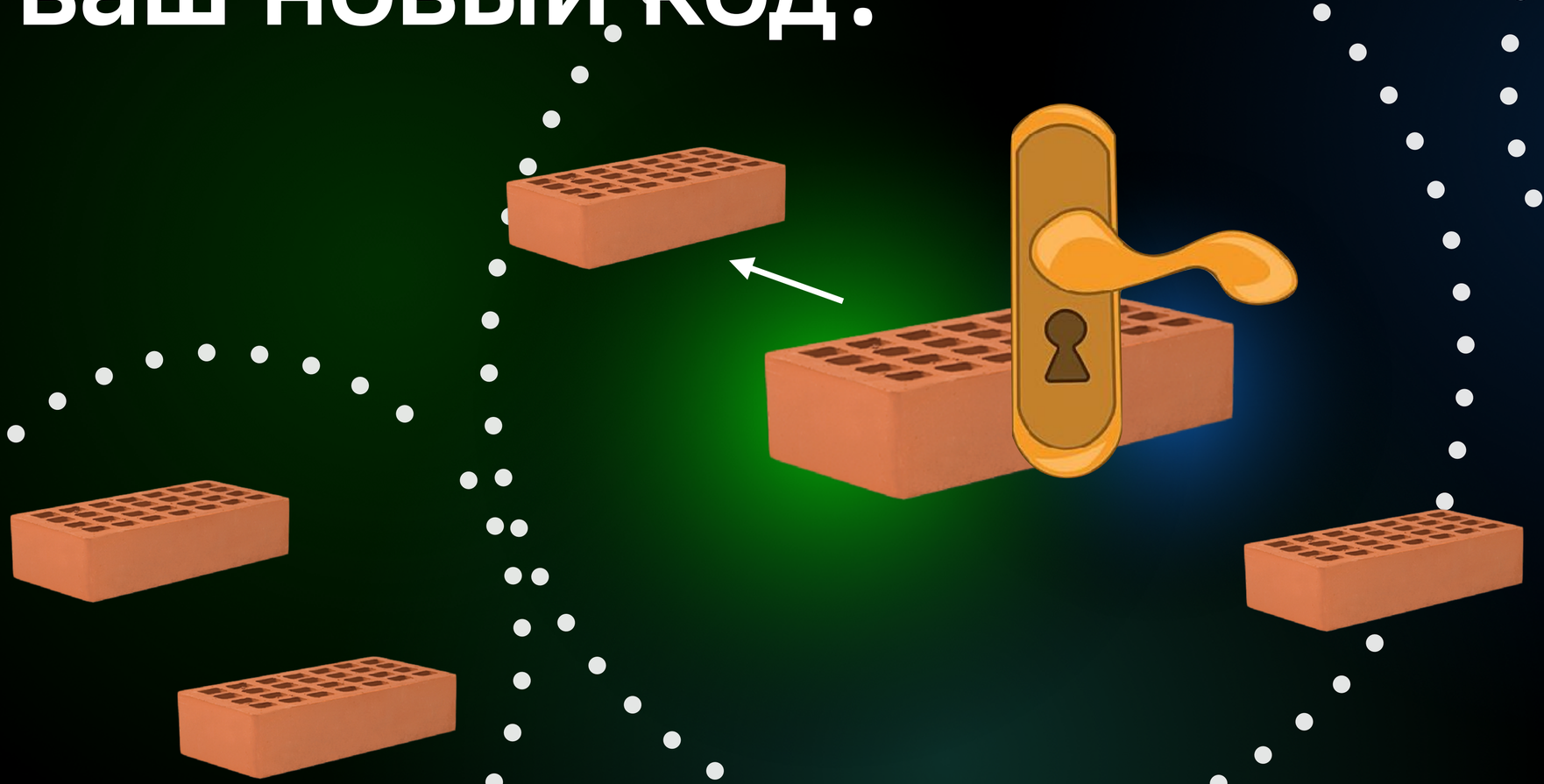
Зачем определять, на что влияет ваш новый код?

Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?

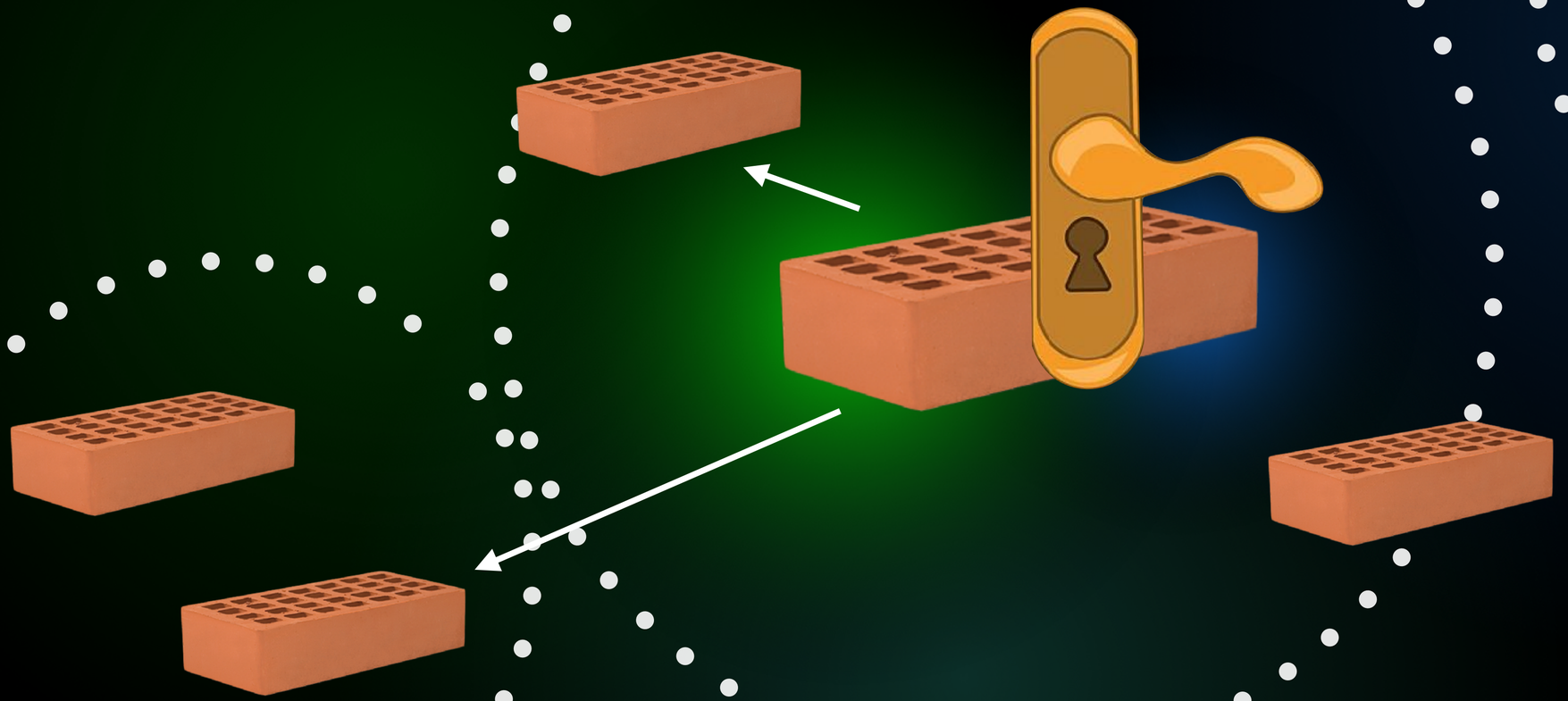
Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



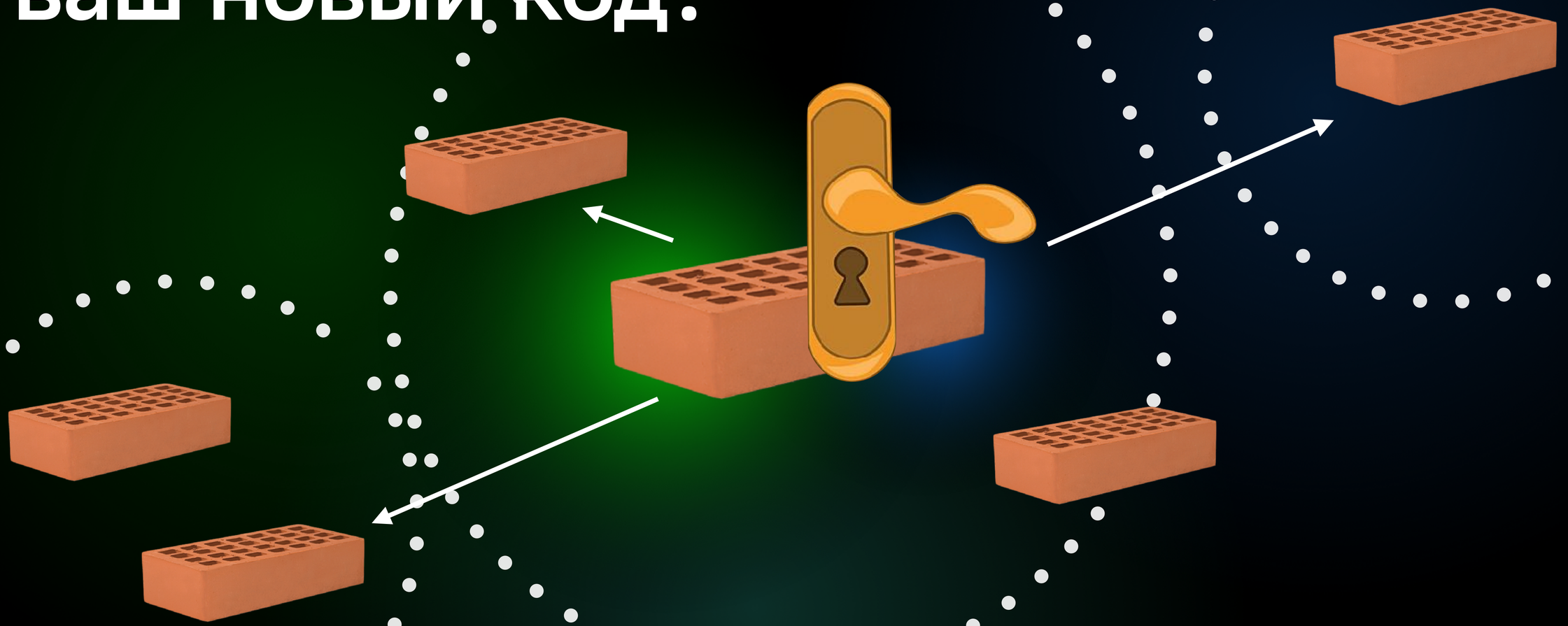
Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



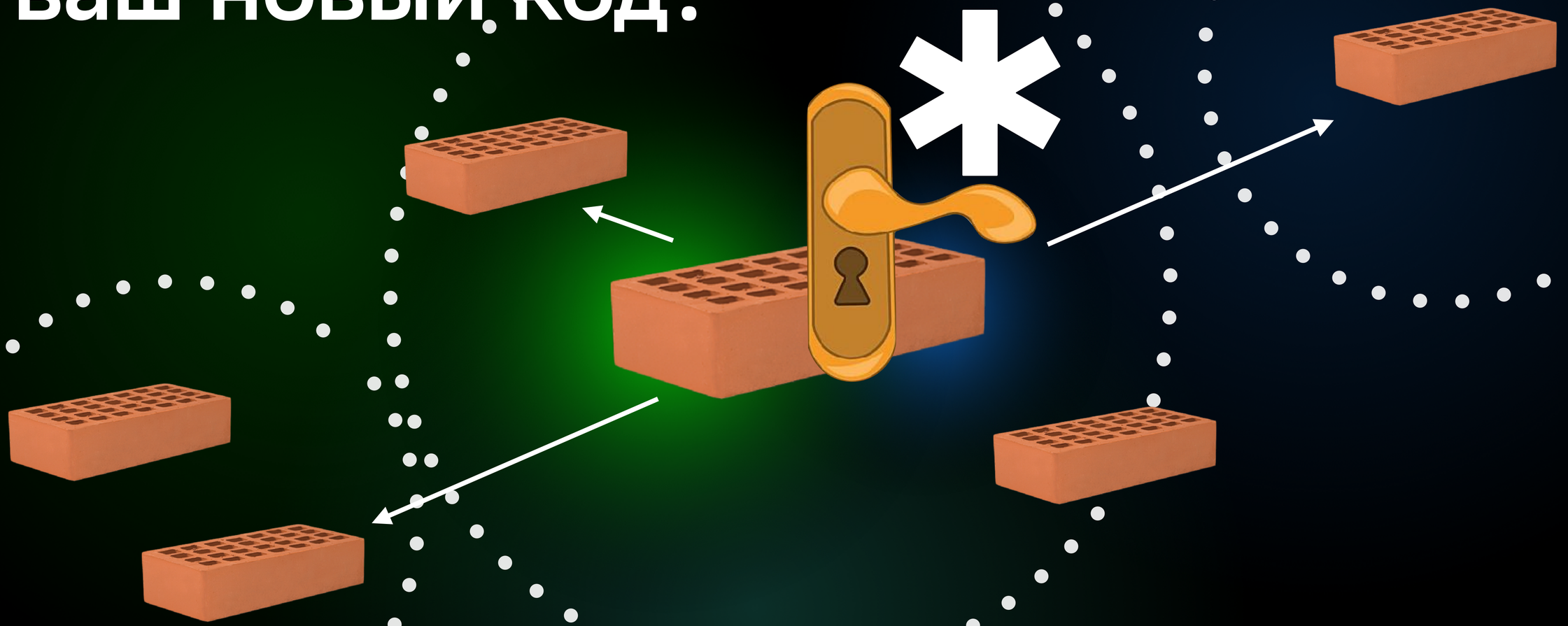
Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?

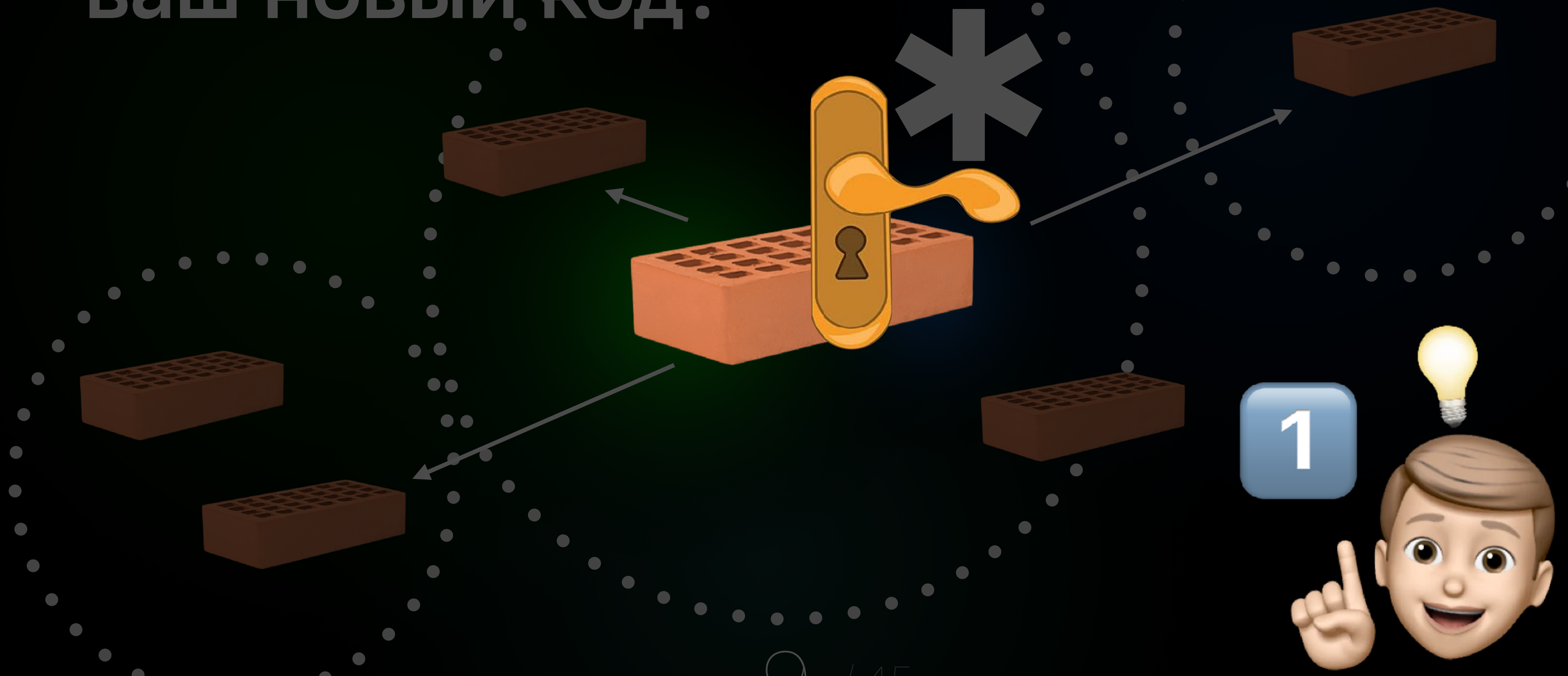


Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?

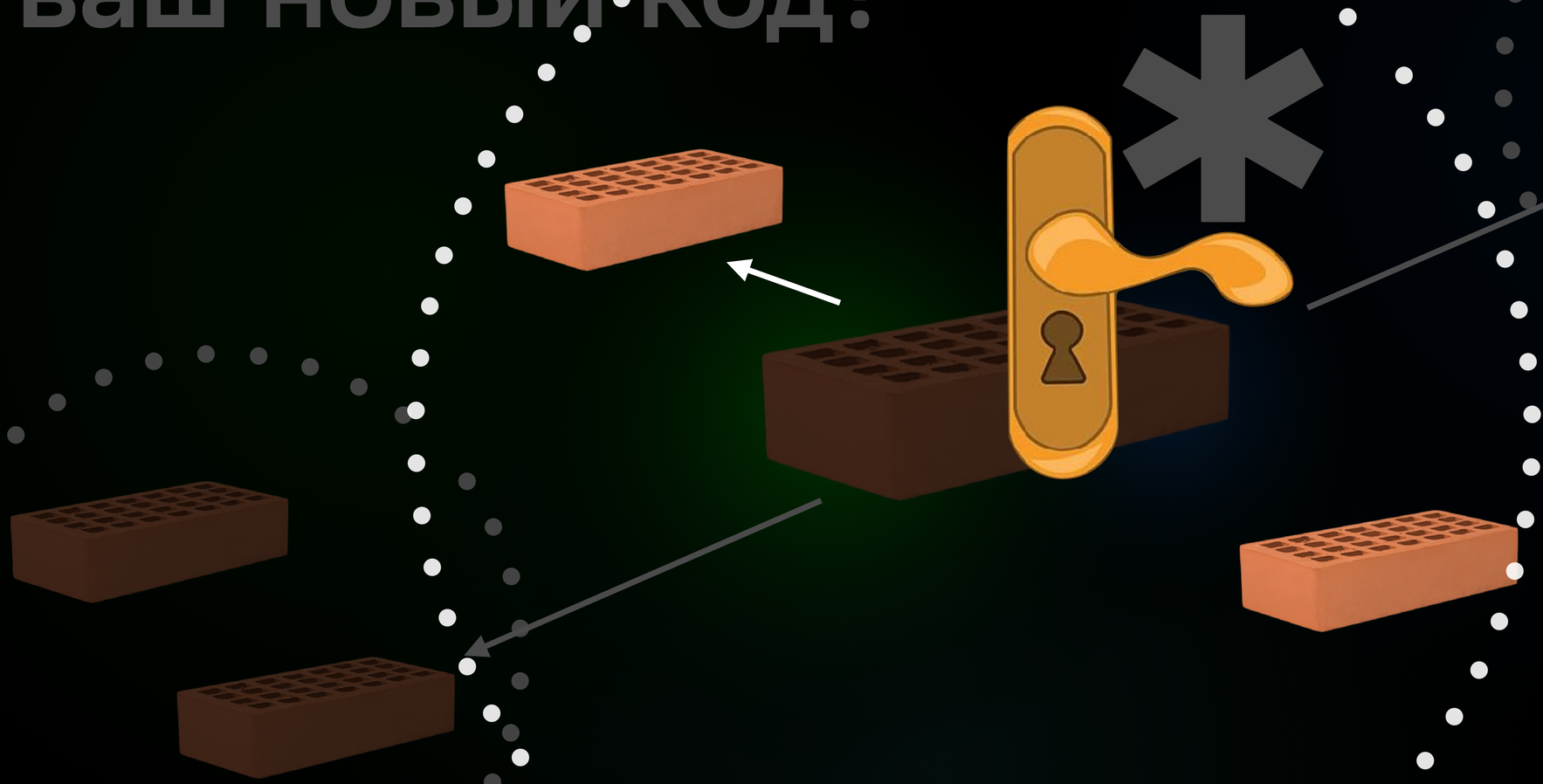
Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



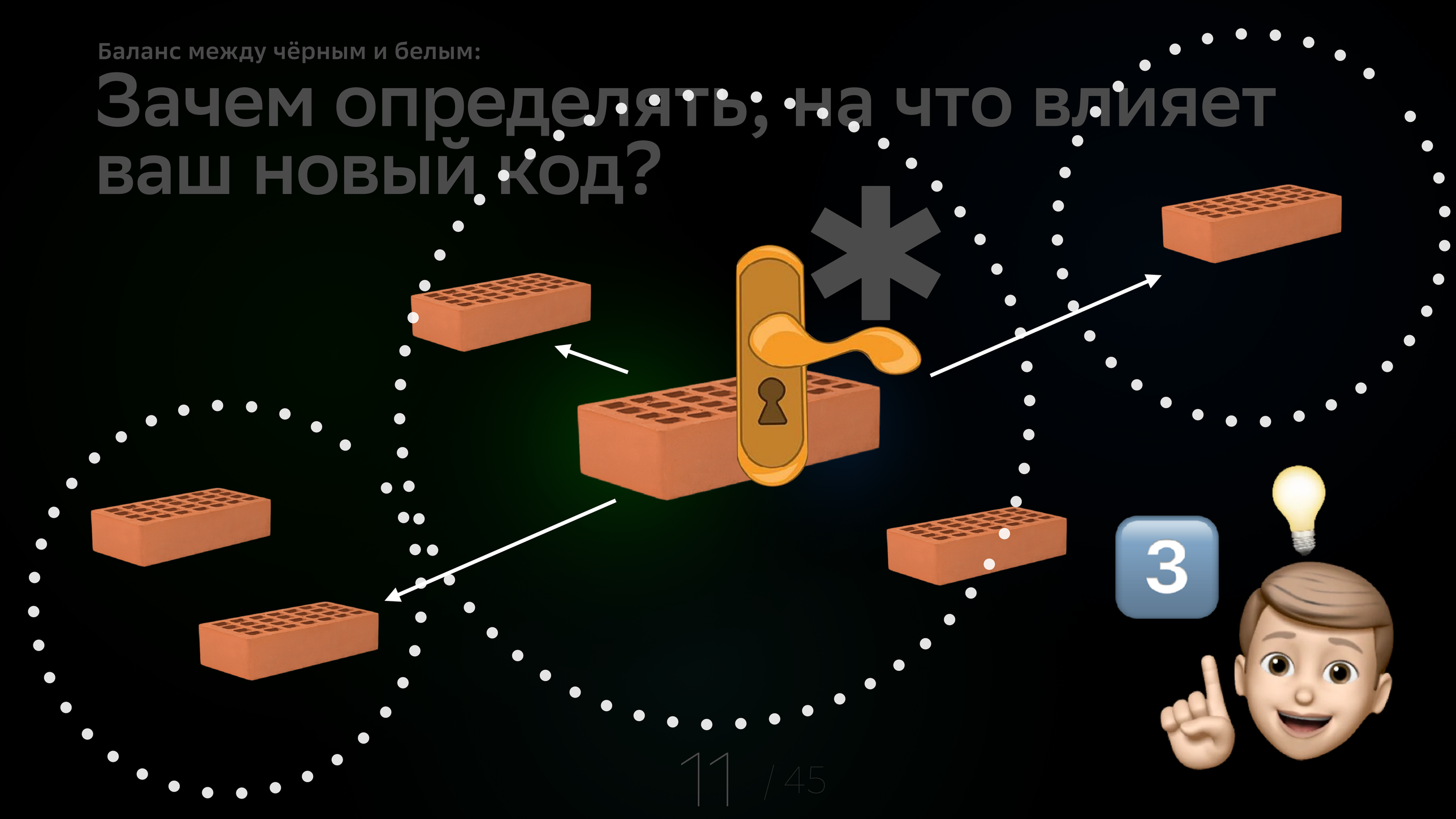
Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



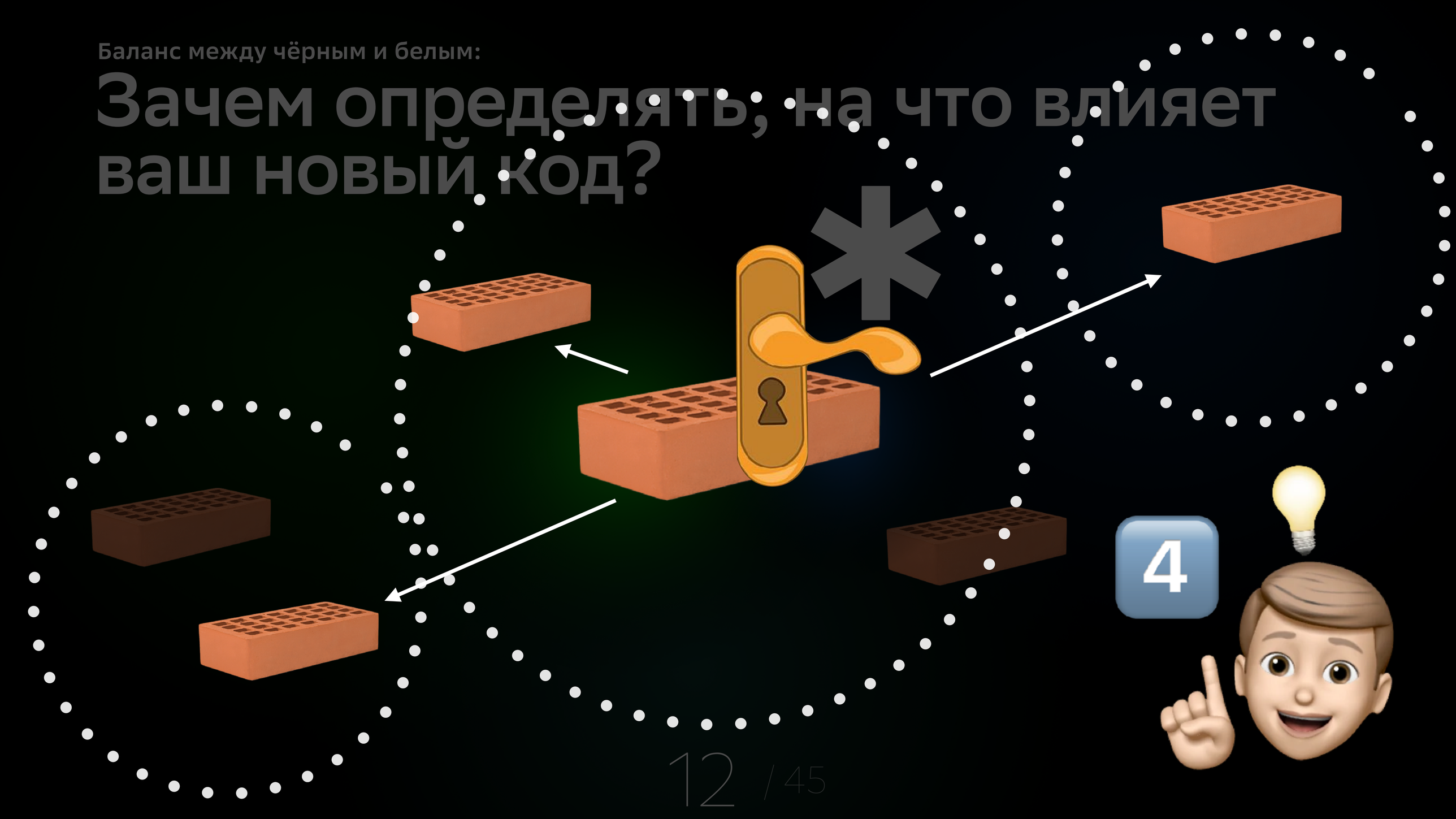
Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



Баланс между чёрным и белым:

Зачем определять, на что влияет ваш новый код?



Переиспользовать нельзя перепроверять

Варианты сборки

1 Собираем только наш таргет

Время сборки

Риски

2 Собираем все таргеты нашего репозитория

Время сборки

Риски

💯

3 Собираем всё

Время сборки

Риски 0

4 Собираем наш таргет и те, что зависят от него

Время сборки

Риски 0

Переиспользовать нельзя перепроверять

Варианты сборки

1 Собираем только наш таргет

Время сборки

Риски

2 Собираем все таргеты нашего репозитория

Время сборки

Риски

3 Собираем всё

Время сборки

Риски

0

4 Собираем наш таргет и те, что зависят от него

Время сборки

Риски

0

Переиспользовать нельзя перепроверять

Варианты сборки

1 Собираем только наш таргет

Время сборки

Риски

2 Собираем все таргеты нашего репозитория

Время сборки

Риски

3 Собираем всё

Время сборки

Риски

0

4 Собираем наш таргет и те, что зависят от него

Время сборки

Риски

0

Переиспользовать нельзя перепроверять

Варианты сборки

1 Собираем только наш таргет

Время сборки

Риски

2 Собираем все таргеты нашего репозитория

Время сборки

Риски

3 Собираем всё

Время сборки

Риски

0

4 Собираем наш таргет и те, что зависят от него

Время сборки

Риски

0

Немного предыстории: почему мы решили, что нам нужен impact?

Почему мы решили, что нам нужен impact?

11+

Почему мы решили, что нам нужен impact?

25.12.2011 - первый коммит

11+

Почему мы решили, что нам нужен impact?

11+

25.12.2011 - первый коммит

Проект рос и был разным

Почему мы решили, что нам нужен impact?

11+

- 25.12.2011 - первый коммит
- Проект рос и был разным
- Нестабильность CI

Почему мы решили, что нам нужен impact?

11+

- 25.12.2011 - первый коммит
- Проект рос и был разным
- Нестабильность CI
- Замедление скорости сборки

Почему мы решили, что нам нужен impact?

11+

- 25.12.2011 - первый коммит
- Проект рос и был разным
- Нестабильность CI
- Замедление скорости сборки
- Рост времени на стабилизацию

Инфраструктура

- 17 pro с 12 ядрами и 64 ГБ
- 4 mini с 6 ядрами и 64 ГБ

Инфраструктура

- 17 pro с 12 ядрами и 64 ГБ
- 4 mini с 6 ядрами и 64 ГБ

Развитие проекта

SBOL iOS Story от Вовы Озерова

1 репозиторий
и 1 таргетом

Развитие проекта

SBOL iOS Story от Вовы Озерова

1 репозиторий
и 1 таргетом

1 репозиторий
и несколько таргетовв

Развитие проекта

SBOL iOS Story от Вовы Озерова

1 репозиторий
и 1 таргетом

Несколько репозиторием
с одним таргетом в каждом
и один репозиторий
с несколькими таргетами

Каким должен быть менеджер зависимостей
на примере Сбербанк-Онлайн iOS
от Ильи Лунькина

1 репозиторий
и несколько таргетовв

Развитие проекта

SBOL iOS Story от Вовы Озерова

1 репозиторий
и 1 таргетом

Несколько репозиторием
с одним таргетом в каждом
и один репозиторий
с несколькими таргетами

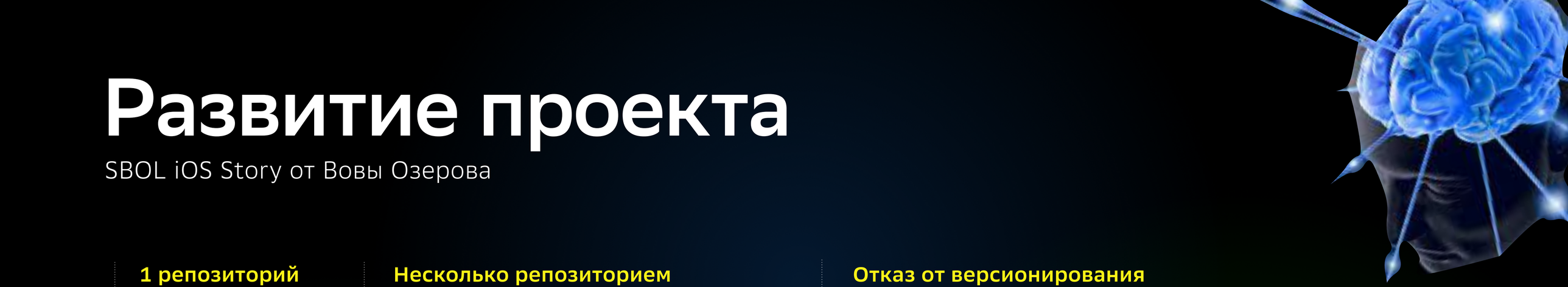
Каким должен быть менеджер зависимостей
на примере Сбербанк-Онлайн iOS
от Ильи Лунькина

1 репозиторий
и несколько таргетовв

Первая эпоха
impact-анализа

Развитие проекта

SBOL iOS Story от Вовы Озерова



1 репозиторий
и 1 таргетом

Несколько репозиторием
с одним таргетом в каждом
и один репозиторий
с несколькими таргетами

Каким должен быть менеджер зависимостей на примере Сбербанк-Онлайн iOS
от Ильи Лунькина

Отказ от версионирования
таргетов и теперь все репозитории
могут иметь несколько таргетов

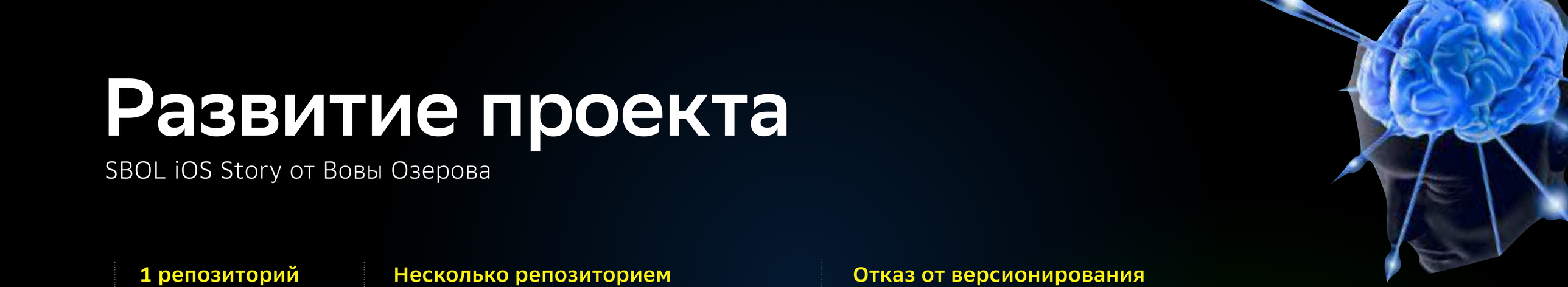
"Мы решили все проблемы работы в модульном проекте. Хотите так же?"
от Миши Харитончика

1 репозиторий
и несколько таргетовв

Первая эпоха
impact-анализа

Развитие проекта

SBOL iOS Story от Вовы Озерова



1 репозиторий
и 1 таргетом

Несколько репозиторием
с одним таргетом в каждом
и один репозиторий
с несколькими таргетами
Каким должен быть менеджер зависимостей на примере Сбербанк-Онлайн iOS
от Ильи Лунькина

Отказ от версионирования
таргетов и теперь все репозитории
могут иметь несколько таргетов
"Мы решили все проблемы работы в модульном проекте. Хотите так же?"
от Миши Харитончика

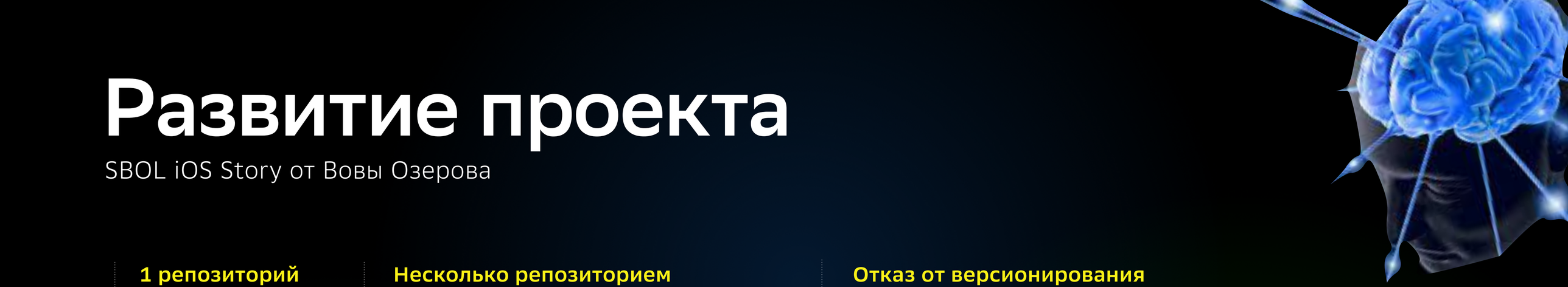
1 репозиторий
и несколько таргетовв

Первая эпоха
impact-анализа

Вторая эпоха
impact-анализа

Развитие проекта

SBOL iOS Story от Вовы Озерова



1 репозиторий
и 1 таргетом

Несколько репозиторием
с одним таргетом в каждом
и один репозиторий
с несколькими таргетами
Каким должен быть менеджер зависимостей на примере Сбербанк-Онлайн iOS
от Ильи Лунькина

Отказ от версионирования
таргетов и теперь все репозитории
могут иметь несколько таргетов
"Мы решили все проблемы работы в модульном проекте. Хотите так же?"
от Миши Харитончика

1 репозиторий
и несколько таргетовв

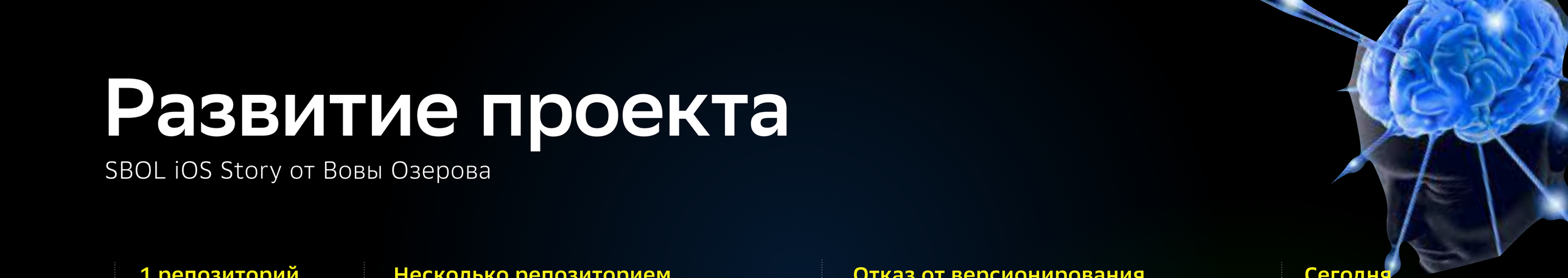
Первая эпоха
impact-анализа

Вторая эпоха
impact-анализа

Новая эпоха
impact-анализа

Развитие проекта

SBOL iOS Story от Вовы Озерова



1 репозиторий и 1 таргетом

1 репозиторий и несколько таргетовв

Несколько репозиторием с одним таргетом в каждом и один репозиторий с несколькими таргетами

Каким должен быть менеджер зависимостей на примере Сбербанк-Онлайн iOS
от Ильи Лунькина

Первая эпоха impact-анализа

Отказ от версионирования таргетов и теперь все репозитории могут иметь несколько таргетов

"Мы решили все проблемы работы в модульном проекте. Хотите так же?"
от Миши Харитончика

Вторая эпоха impact-анализа

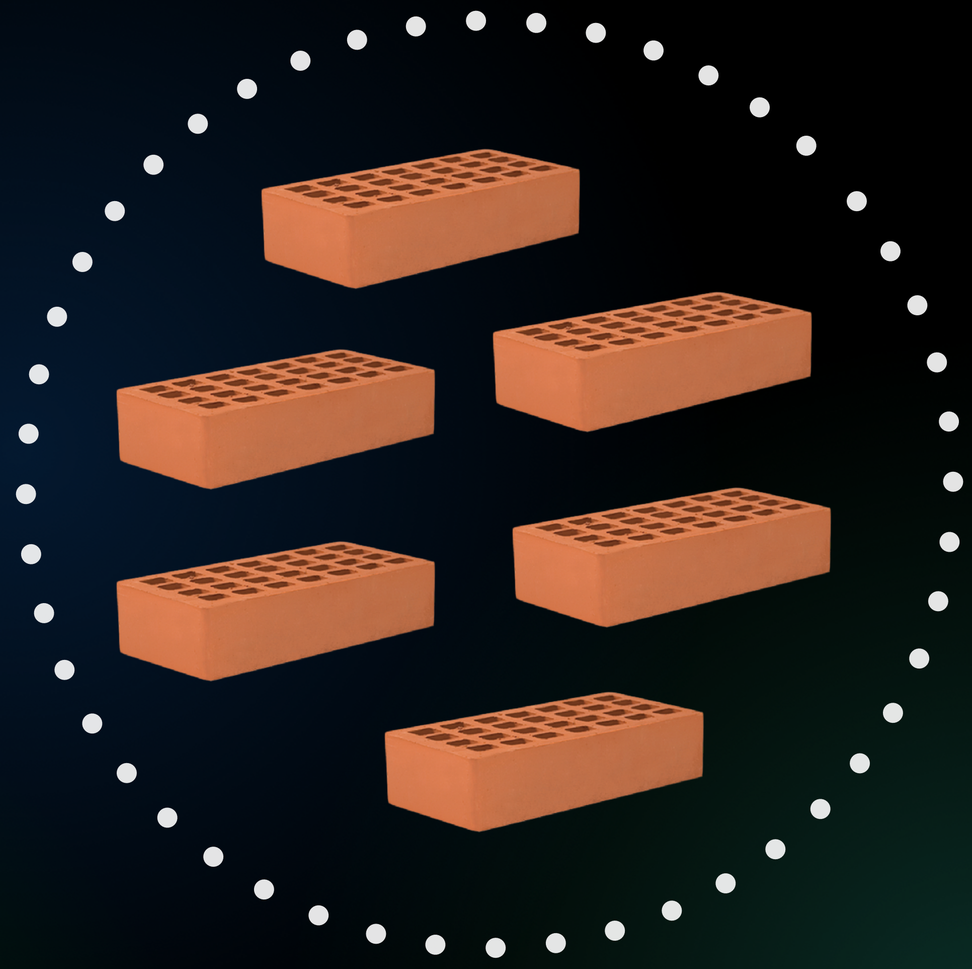
Сегодня

Новая эпоха impact-анализа

Первая эпоха impact-анализа

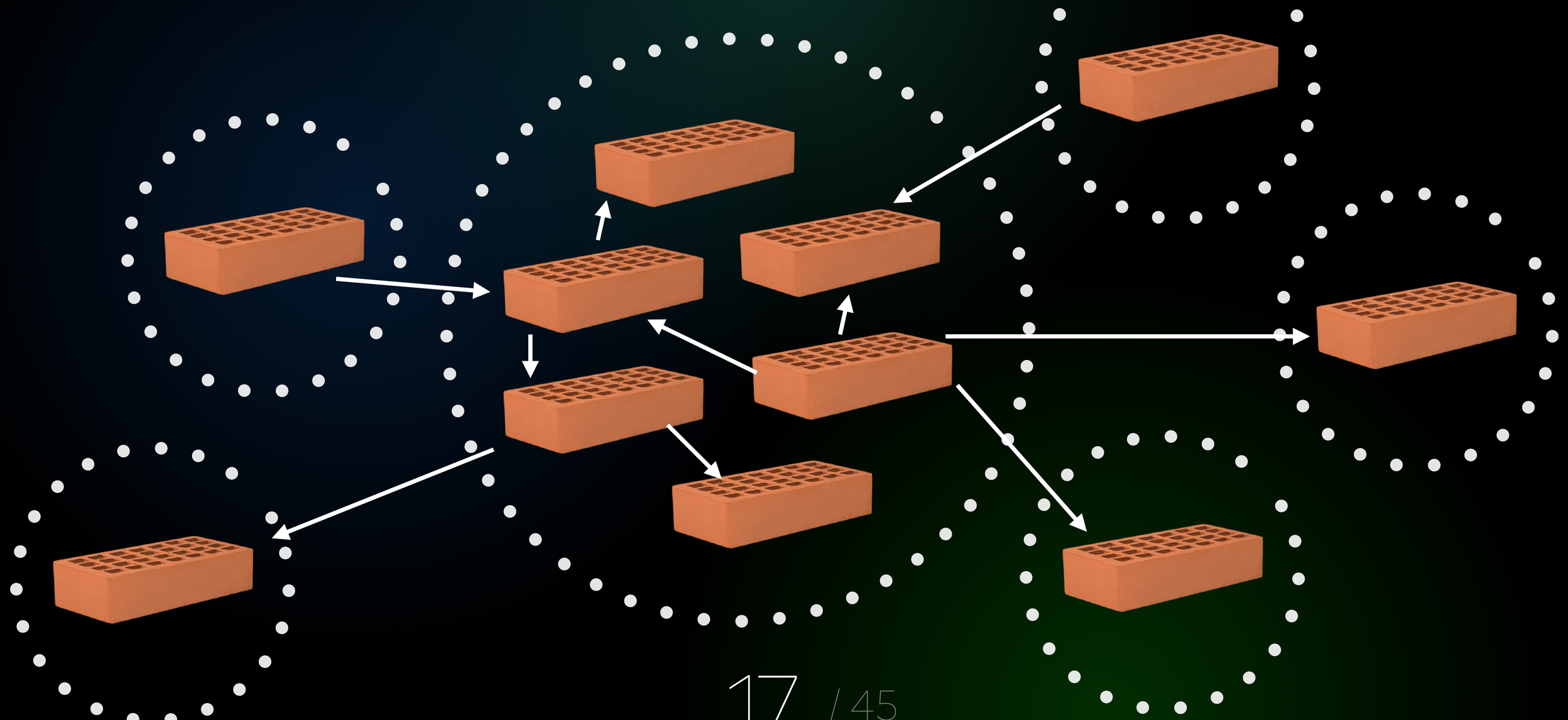
Как был устроен проект?

Как был устроен проект?

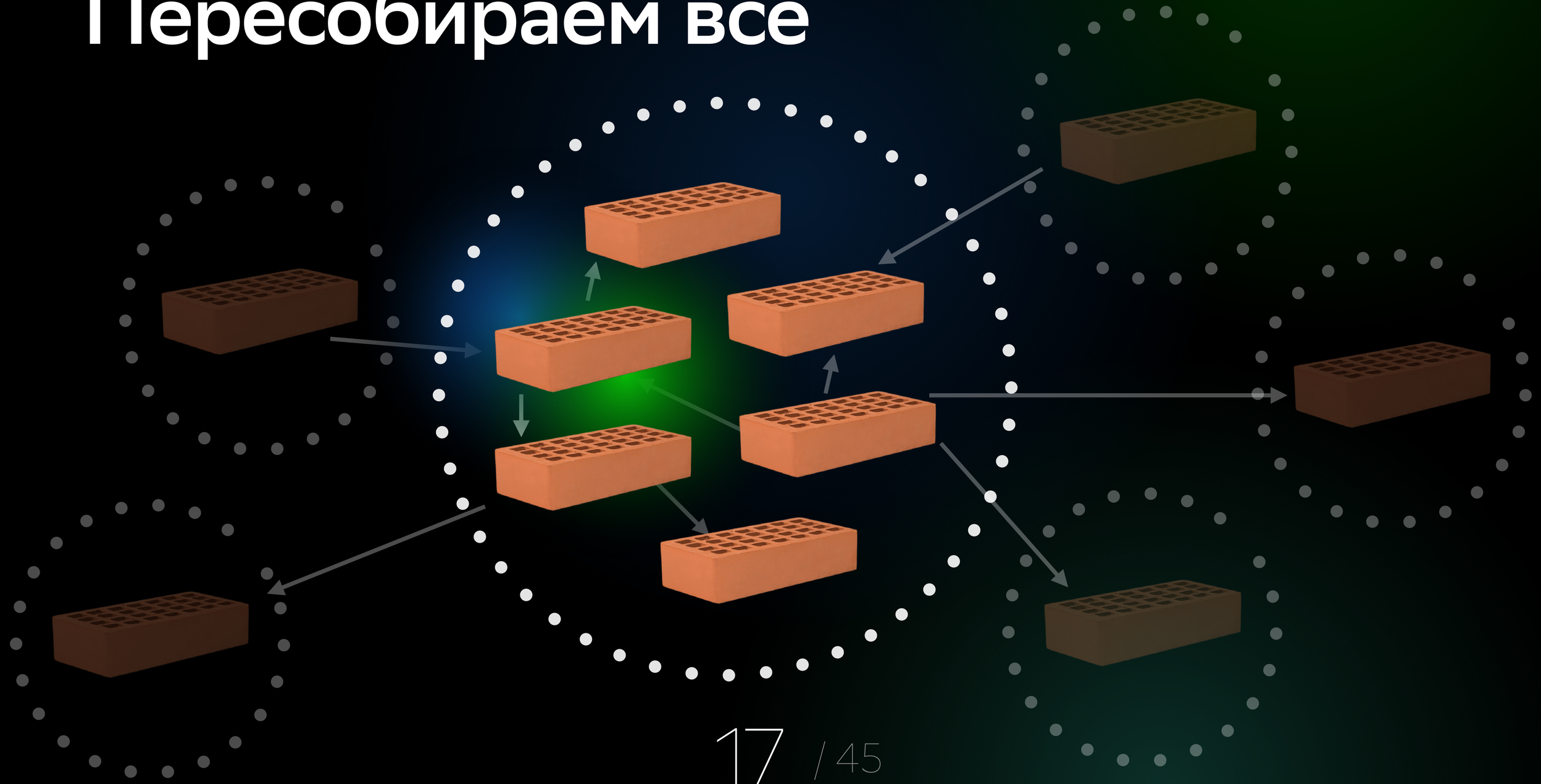


Как был устроен проект?

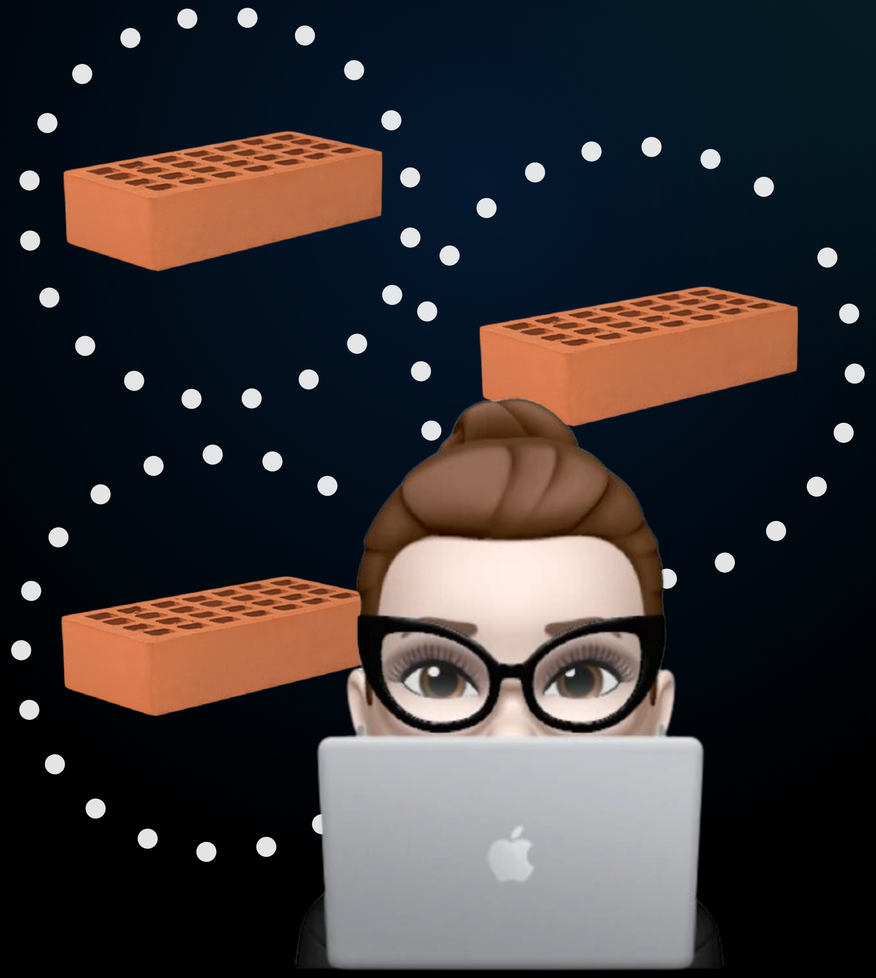
Как был устроен проект?



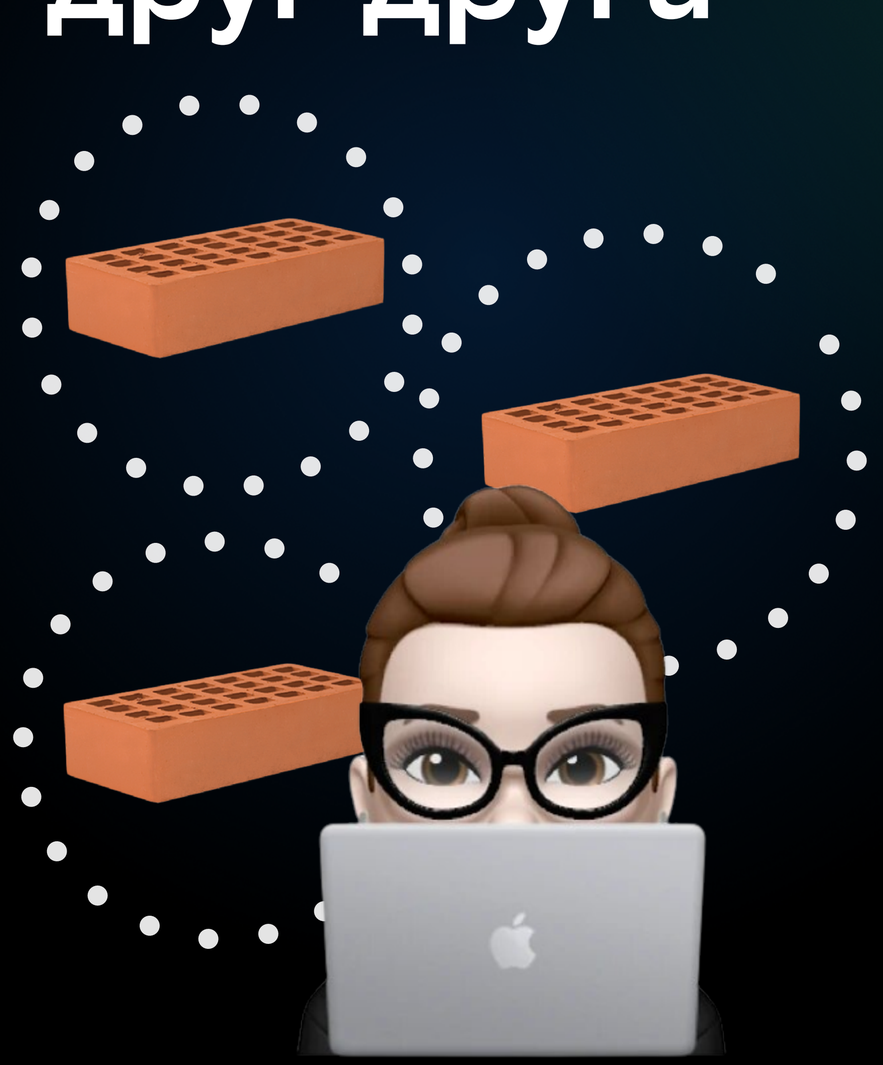
Пересобираем всё



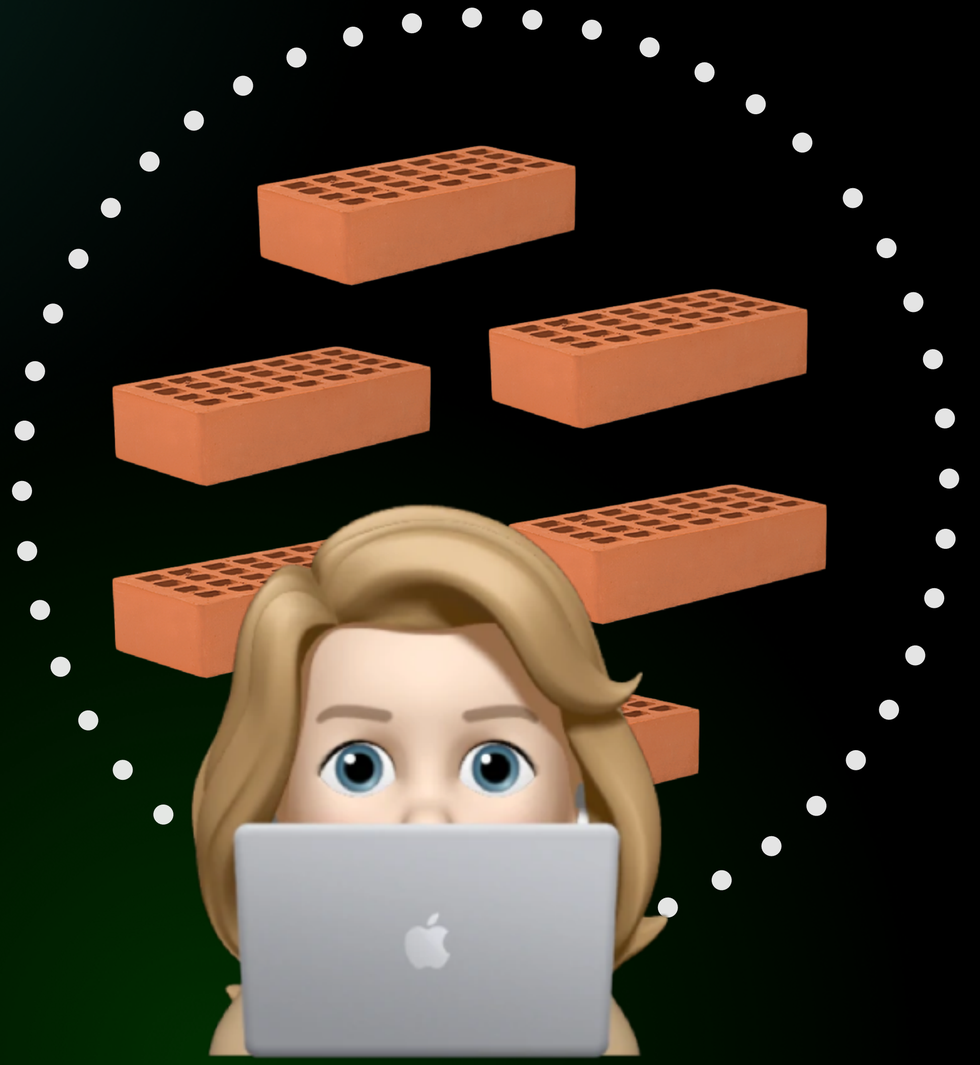
Разработчики не понимали друг друга



Разработчики не понимали друг друга



VS



Разработчики не понимали друг друга

Разработчики не понимали друг друга

VS

Как работал impact-анализ?

Как работал impact-анализ?

→ Получаем измененные в PR-е файлы

Как работал impact-анализ?

- Получаем измененные в PR-е файлы
- Получаем список публичных файлов из xcodeproj

Как работал impact-анализ?

- Получаем измененные в PR-е файлы
- Получаем список публичных файлов из xcodeproj
- Находим пересечение

Как работал impact-анализ?

- Получаем измененные в PR-е файлы
- Получаем список публичных файлов из xcodeproj
- Находим пересечение
- К нему добавляем измененные swift-файлы, содержащие public или open

Как работал impact-анализ?

- Получаем измененные в PR-е файлы
- Получаем список публичных файлов из xcodeproj
- Находим пересечение
- К нему добавляем измененные swift-файлы, содержащие public или open
- Пересобираем все таргеты в репозитории

Внедряем импакт в сборку кода

1
Лишние прогоны

2
Учитывает не все изменения

3
141 строчка кода на ruby

4
Частичная надежность

Внедряем импакт в сборку кода

1
Лишние прогоны

2
Учитывает не все изменения

3
141 строчка кода на ruby

4
Частичная надежность

Внедряем импакт в сборку кода

1
Лишние прогоны

2
Учитывает не все изменения

3
141 строчка кода на ruby

4
Частичная надежность

Внедряем импакт в сборку кода

1
Лишние прогоны

2
Учитывает не все изменения

3
141 строчка кода на ruby

4
Частичная надежность

Вторая эпоха impact-анализа

Внедряем импакт в сборку кода

1

Пересобирать только
зависимые модули

Внедряем импакт в сборку кода

1

Пересобирать только зависимые модули

2

Не делать лишних сборок

Как работал impact-анализ?

Как работал impact-анализ?

- Собираем все таргеты, кроме измененных, бинарями

Как работал impact-анализ?

- Собираем все таргеты, кроме измененных, бинарями
- Получаем флаг наличия публичных изменений из xcodeproj и по <имя таргета>-Swift.h

Как работал impact-анализ?

- Собираем все таргеты, кроме измененных, бинарями
- Получаем флаг наличия публичных изменений из xcodeproj и по <имя таргета>-Swift.h
- Получаем список таргетов, которые зависят от публичного интерфейса измененных таргетов

Как работал impact-анализ?

- Собираем все таргеты, кроме измененных, бинарями
- Получаем флаг наличия публичных изменений из xcodeproj и по <имя таргета>-Swift.h
- Получаем список таргетов, которые зависят от публичного интерфейса измененных таргетов
- Пересобираем измененные таргеты и зависящие таргеты

Зачем собирать два раза?

1

Получаем заранее посчитанную чексумму swiftinterface до первого билда

2

Считаем чексумму swiftinterface после первого билда бинарями

3

Сравниваем

Зачем собирать два раза?

1

Получаем заранее посчитанную чексумму swiftinterface до первого билда

2

Считаем чексумму swiftinterface после первого билда бинарями

3

Сравниваем

Зачем собирать два раза?

1

Получаем заранее посчитанную чексумму swiftinterface до первого билда

2

Считаем чексумму swiftinterface после первого билда бинарями

3

Сравниваем

Как понять, что код связан?

Как понять, что код связан?

PackageDependencies

Как понять, что код связан?

```
{
  "main": {
    "workspace": [
      "PBPrivileges_L1"
    ],
    "sdk": [
      "Foundation"
    ]
  },
  "test": {
    "sdk": [
      "XCTest"
    ]
  }
}
```

PackageDependencies

Подключаем impact-анализ к прогону unit и UI-тестов

Как реализовать прогон только нужных тестов?

Как реализовать прогон только нужных тестов?

1

Получаем список таргетов,
которые зависят
от публичного интерфейса
измененных таргетов

Как реализовать прогон только нужных тестов?

1

Получаем список таргетов,
которые зависят
от публичного интерфейса
измененных таргетов

2

Добавляем схемы
для тестирования измененных
таргетов и таргетов,
которые от них зависят

Как реализовать прогон только нужных тестов?

Схема unit тестов = <Имя таргета>Tests

Как реализовать прогон только нужных тестов?

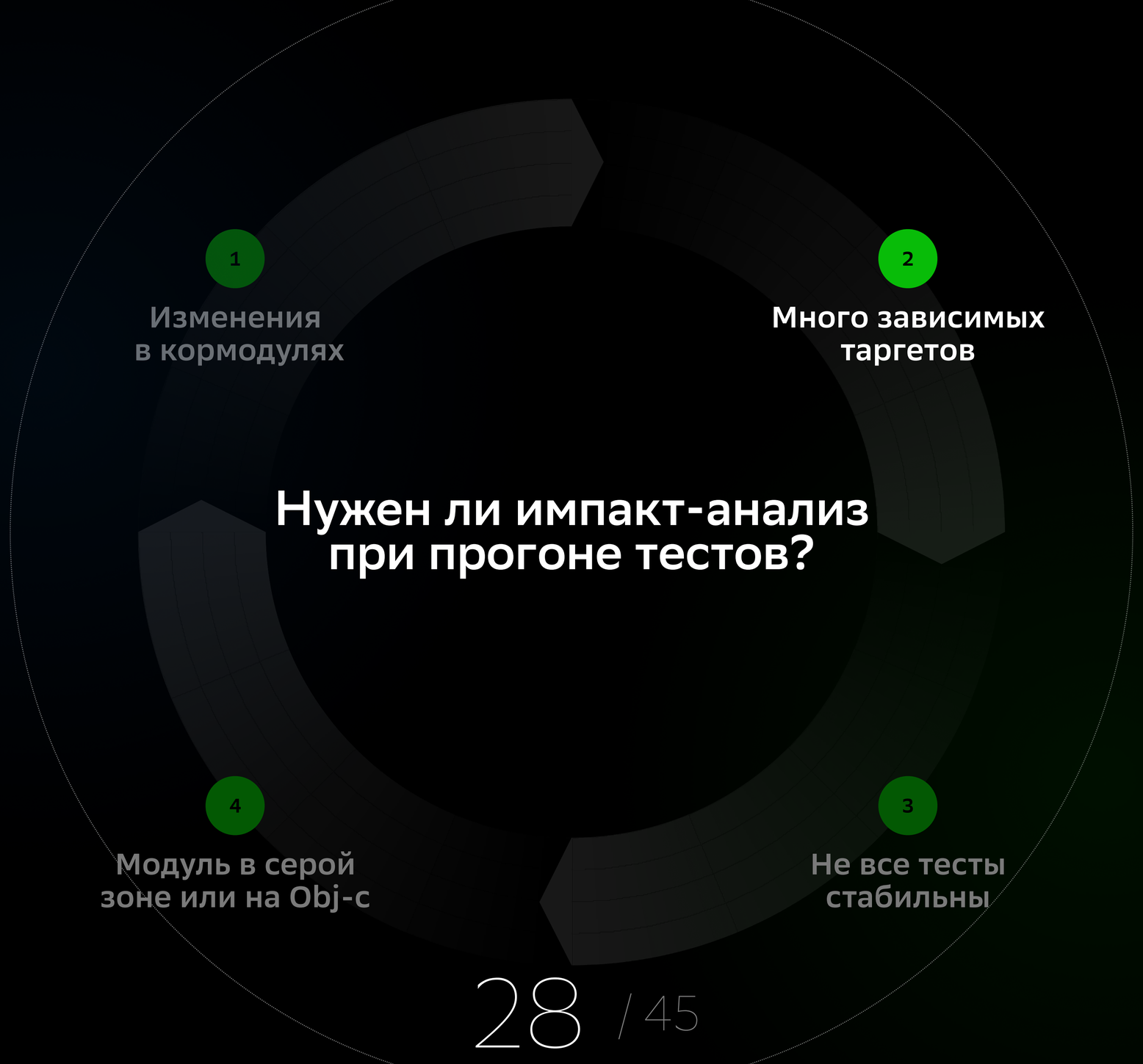
Схема unit тестов = <Имя таргета>Tests

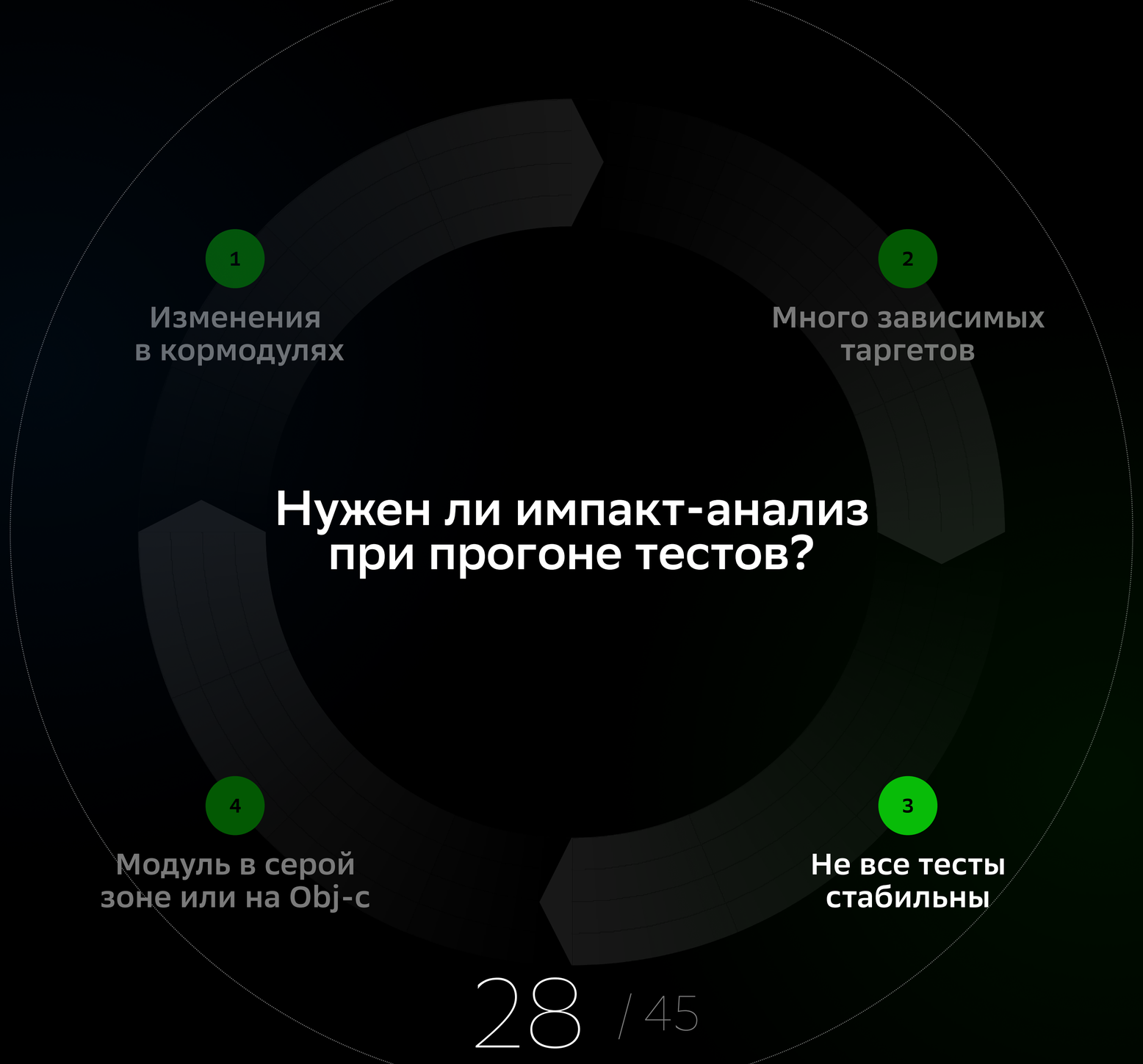
Схема UI тестов = <Имя таргета>UITests

Нужен ли импакт-анализ при прогоне тестов?

Нужен ли импакт-анализ при прогоне тестов?

1. Изменения в кормодулях
2. Много зависимых таргетов
3. Не все тесты стабильны
4. Модуль в серой зоне или на Obj-c


1
Изменения
в кормодулях
2
Много зависимых
таргетов
Нужен ли импакт-анализ
при прогоне тестов?
4
Модуль в серой
зоне или на Obj-c
3
Не все тесты
стабильны


1
Изменения
в кормодулях
2
Много зависимых
таргетов
Нужен ли импакт-анализ
при прогоне тестов?
4
Модуль в серой
зоне или на Obj-c
3
Не все тесты
стабильны

Нужен ли импакт-анализ при прогоне тестов?

1. Изменения в кормодулях
2. Много зависимых таргетов
3. Не все тесты стабильны
4. Модуль в серой зоне или на Obj-c

Проблемы, которые не учли во второй подход

Проблемы, которые не учли во второй подход

Добавляем в сборку все зависящие таргеты и их тесты,
даже если в них не используются изменяемы методы

Проблемы, которые не учли во второй подход

Добавляем в сборку все зависящие таргеты и их тесты,
даже если в них не используются изменяемы методы

Не учитывали swiftinterface

Проблемы, которые не учли во второй подход

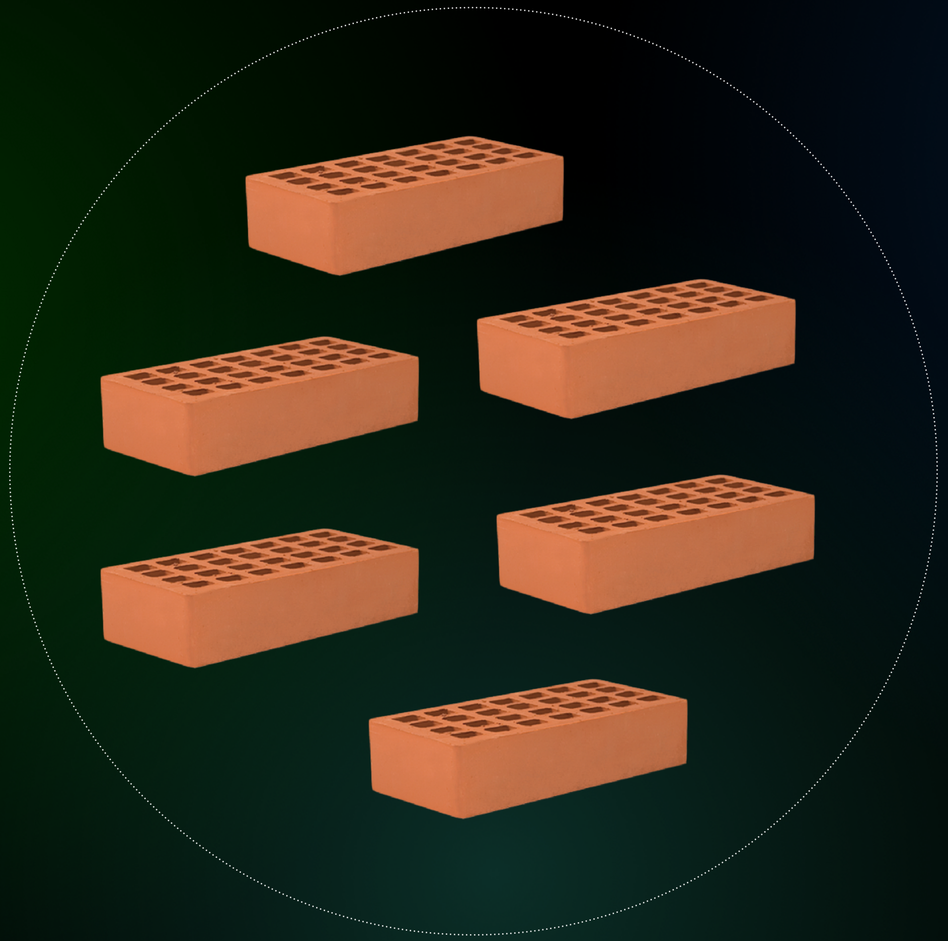
Добавляем в сборку все зависящие таргеты и их тесты,
даже если в них не используются изменяемы методы

Не учитывали swiftinterface

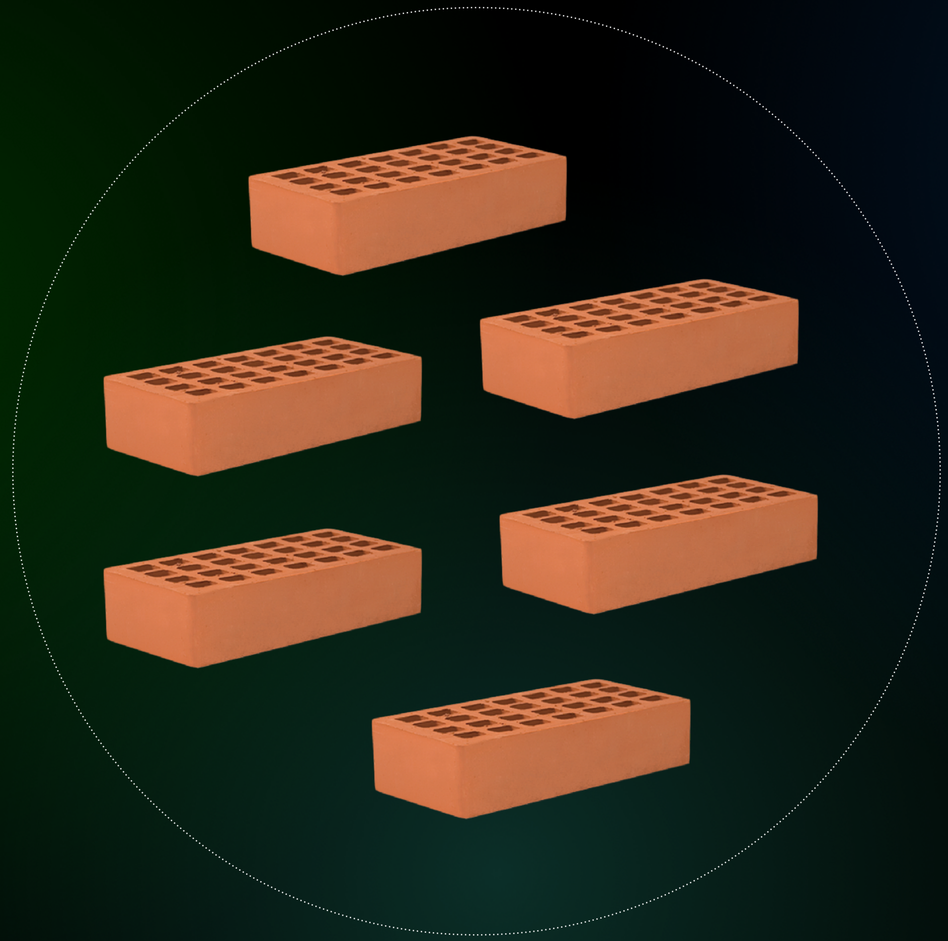
Если вы изменили интерфейс, а кто-то параллельно делал PR
с использованием старого варианта

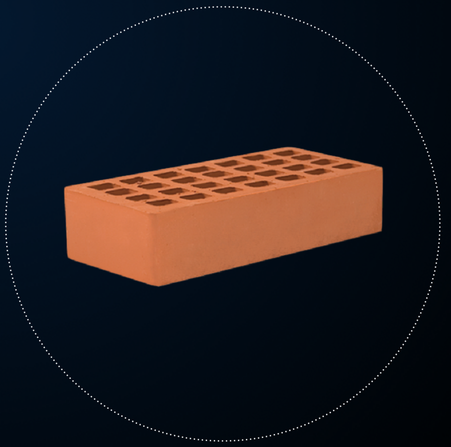
Новая эпоха impact-анализа

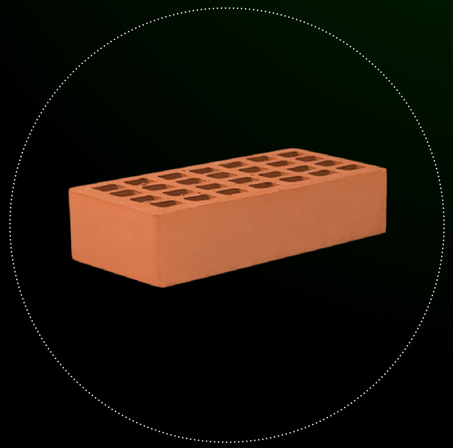
Как устроен проект сейчас?



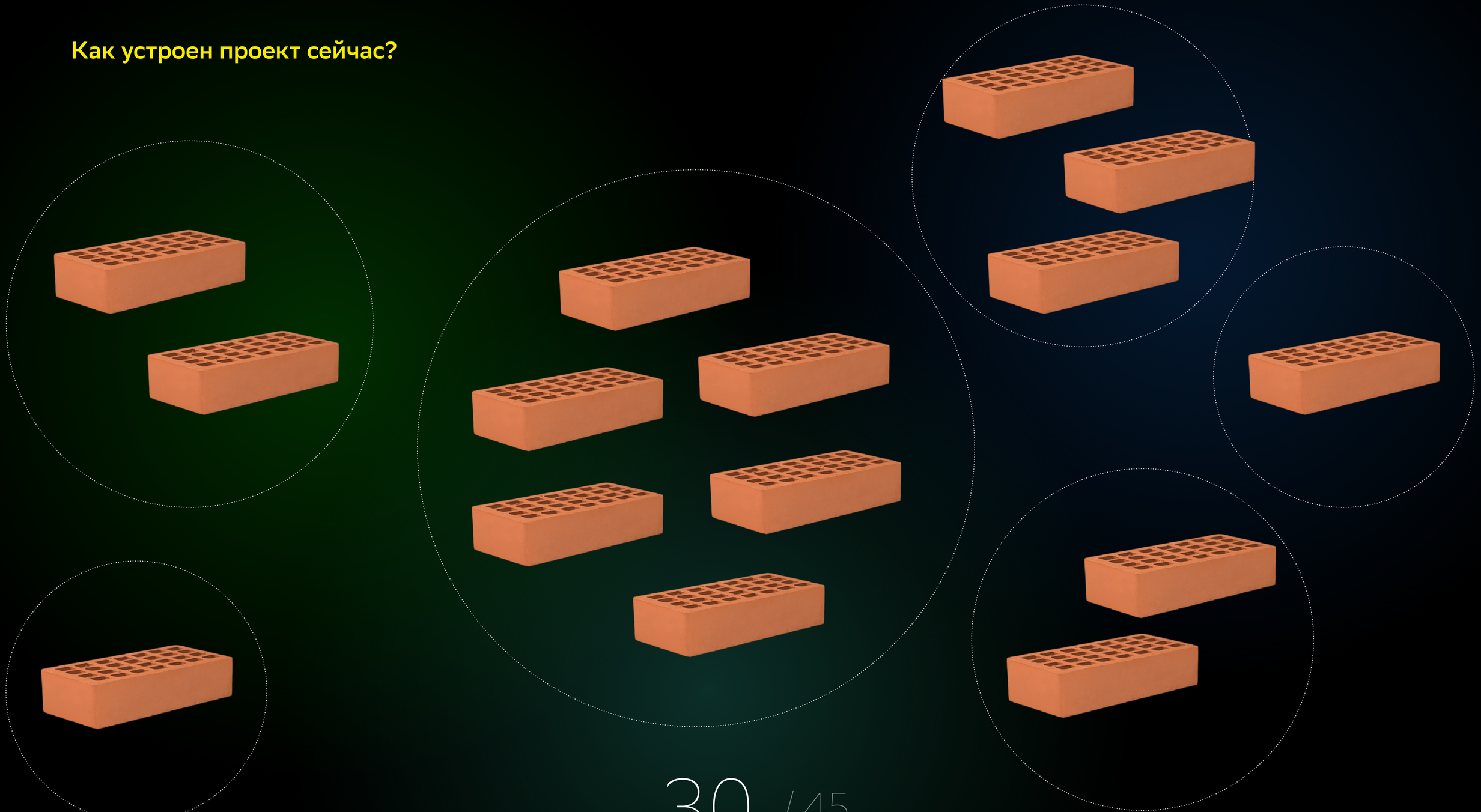
Как устроен проект сейчас?



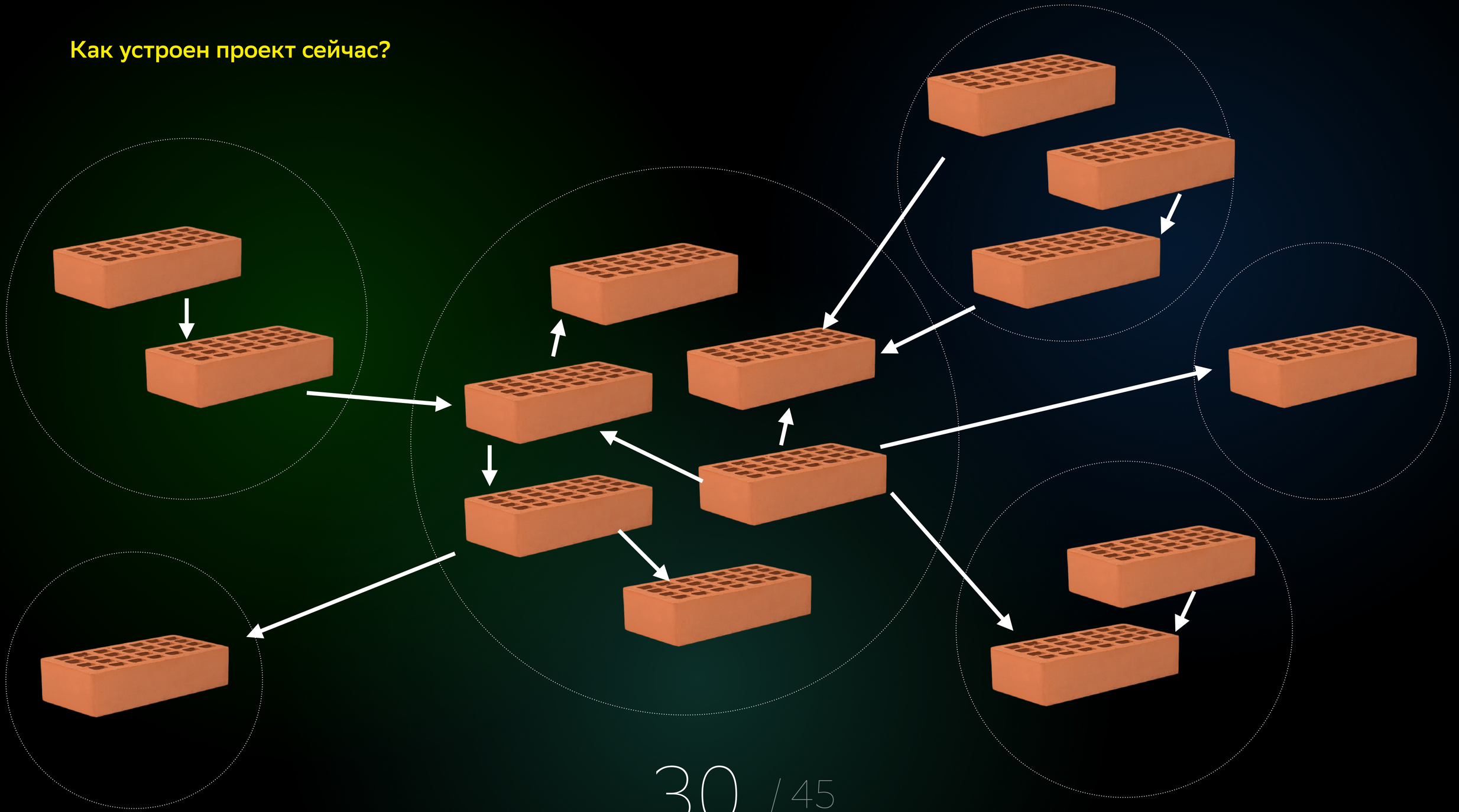




Как устроен проект сейчас?



Как устроен проект сейчас?



snapshot_hash

hash, однозначно определяющий версию артефакта

Механизм сборки

Механизм сборки

1

При очередной сборке
билд-кэша пересчитываются
snapshot_hash всех таргетов

Механизм сборки

1

При очередной сборке
билд-кэша пересчитываются
snapshot_hash всех таргетов

2

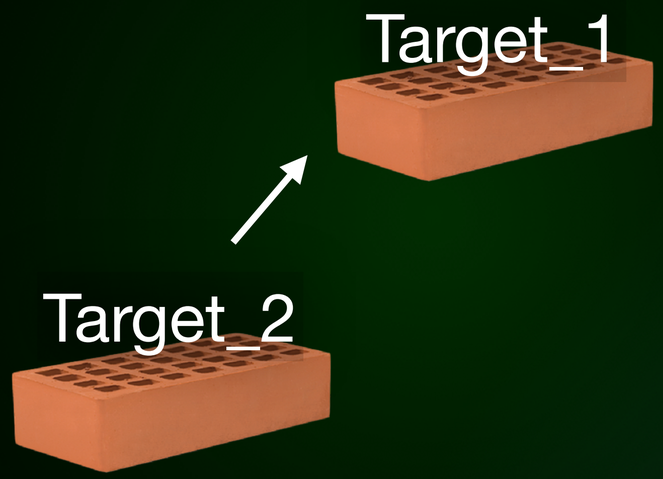
Пересобираются только
те таргеты, snapshot_hash
которых изменился

Механизм сборки

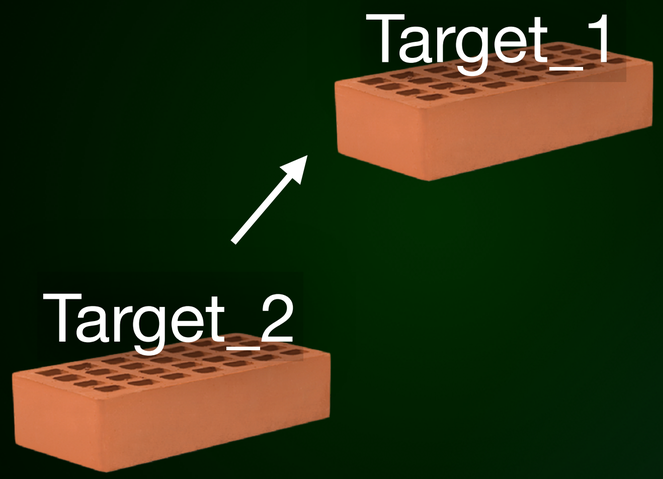
Механизм сборки

Target_1

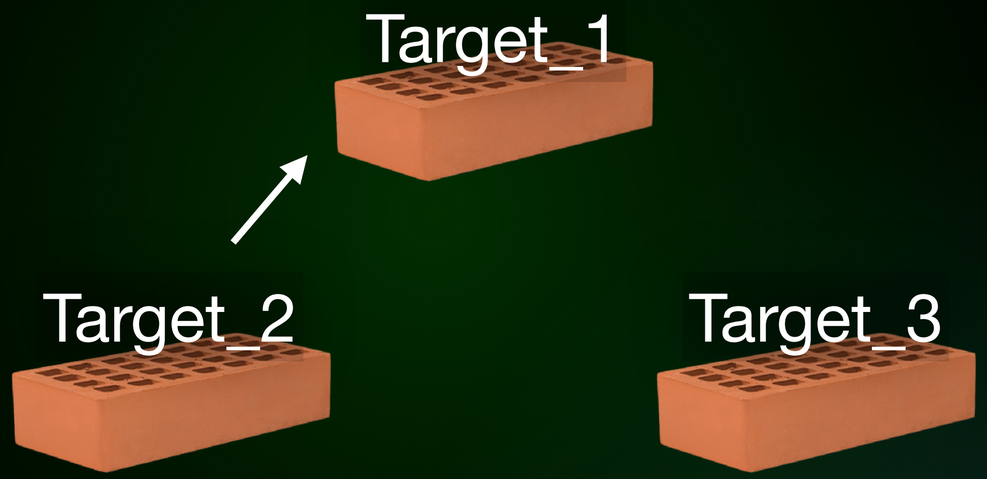
Механизм сборки



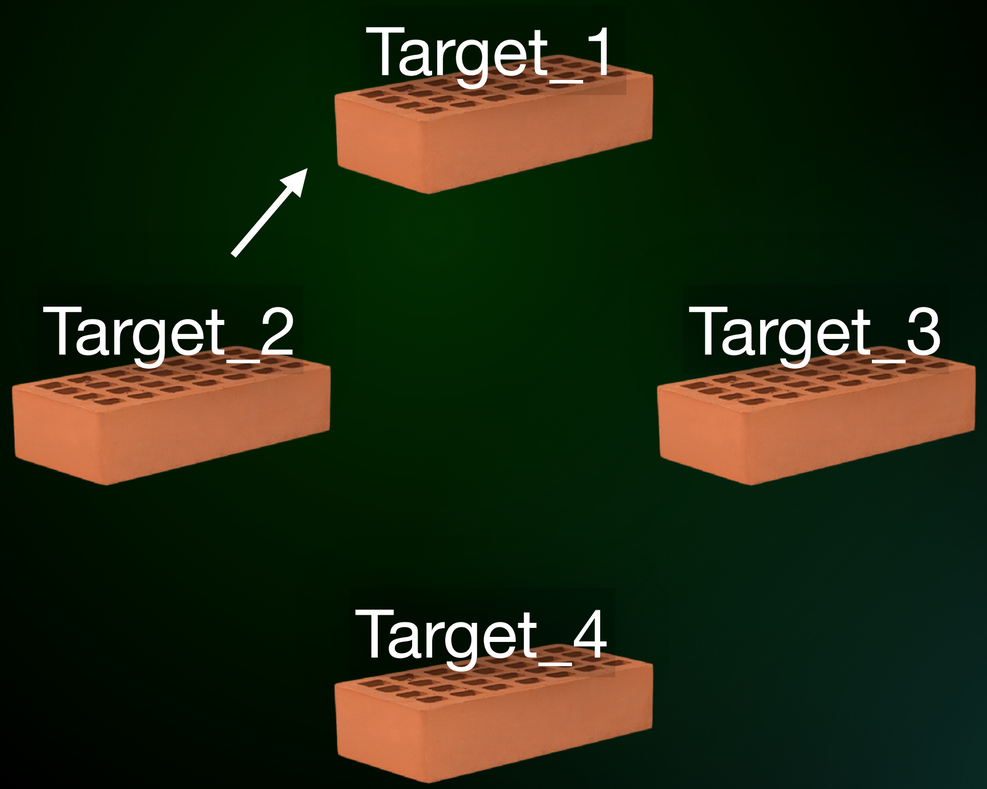
Механизм сборки



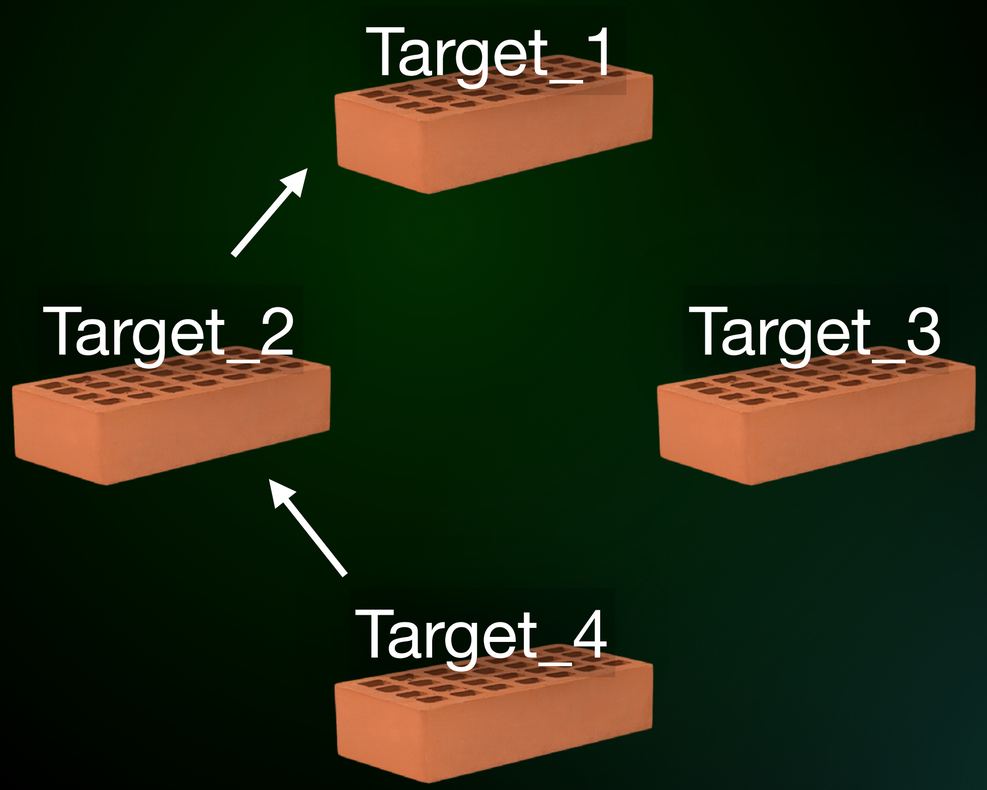
Механизм сборки



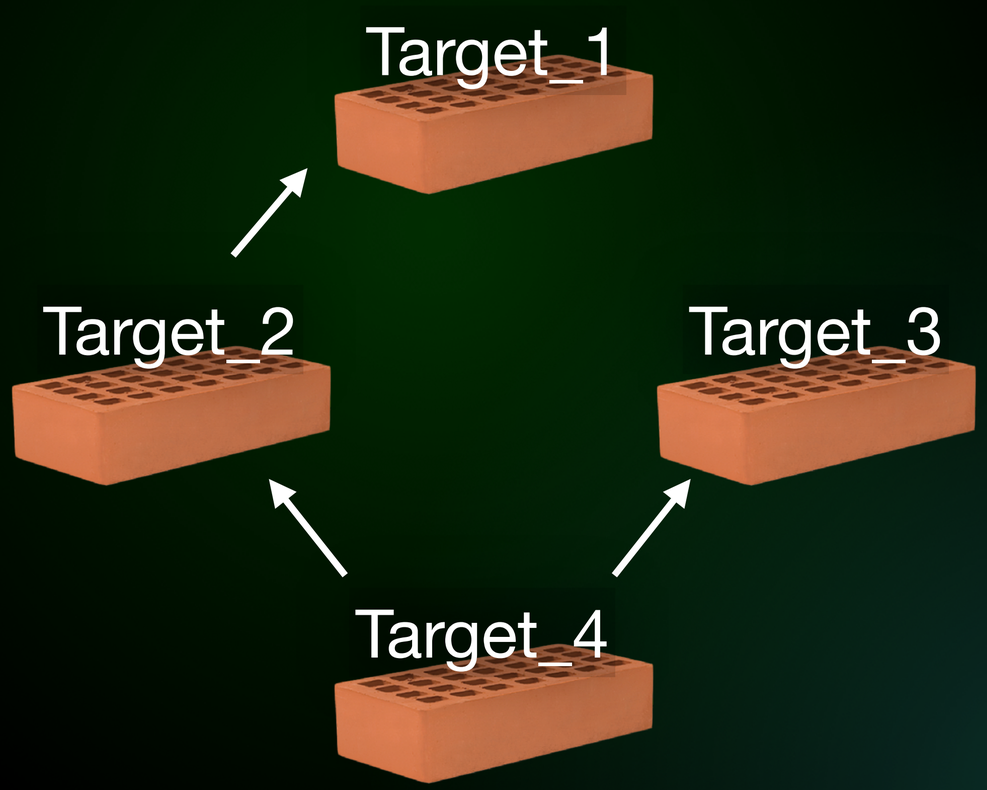
Механизм сборки



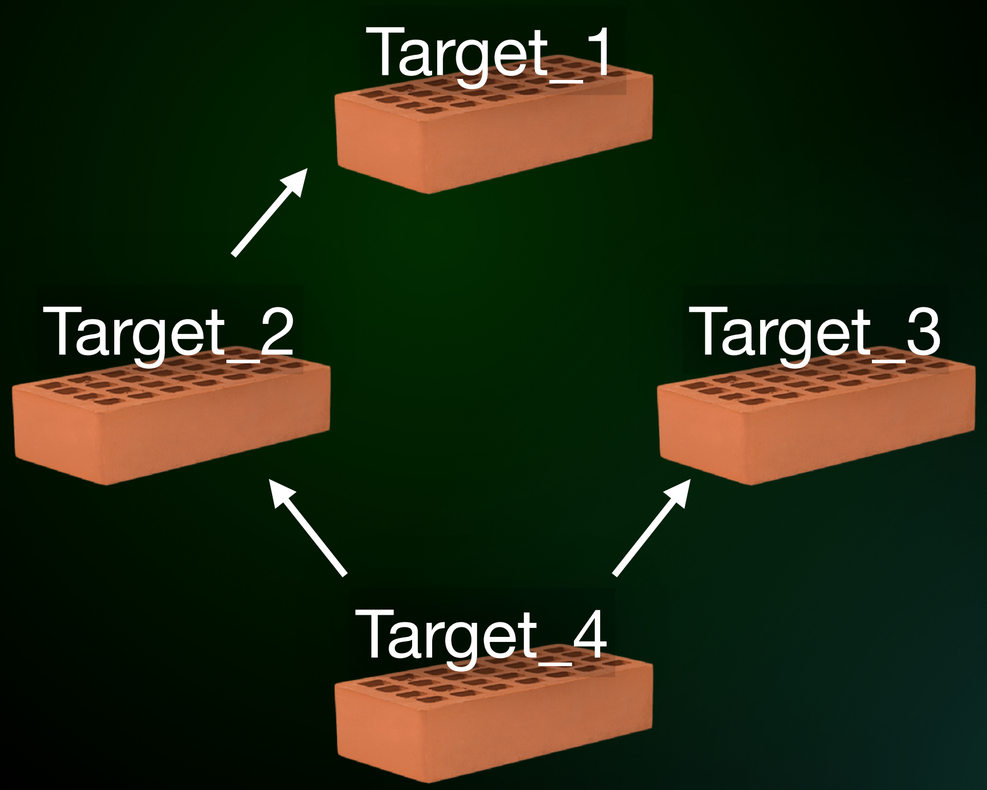
Механизм сборки



Механизм сборки

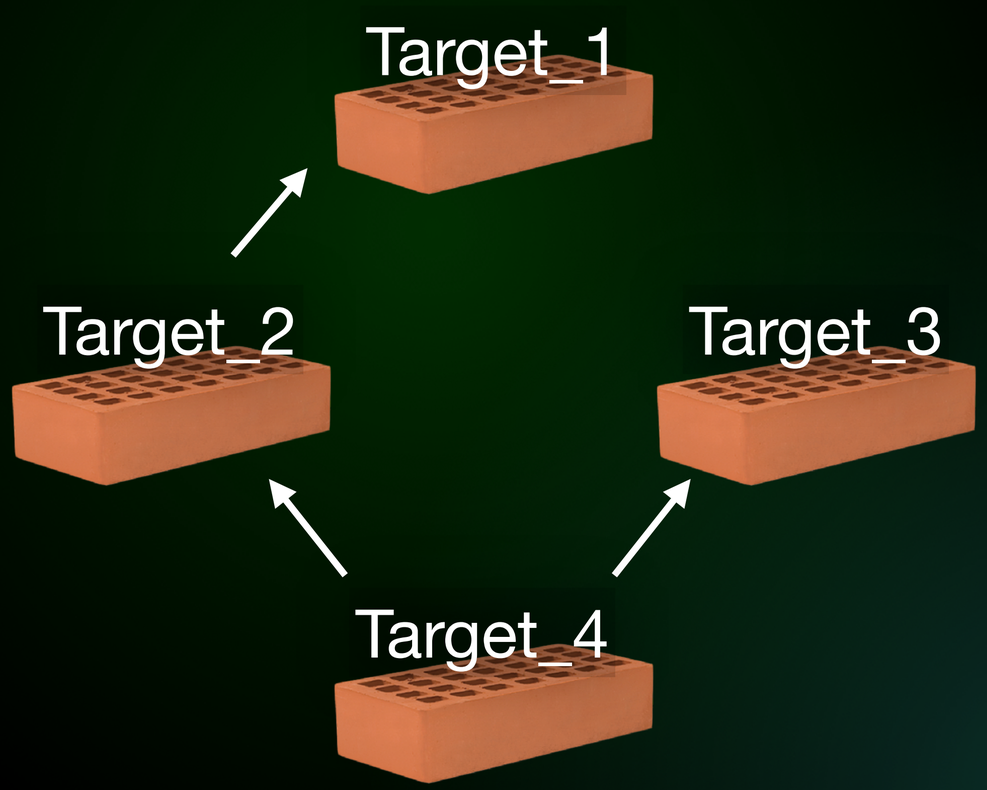


Механизм сборки



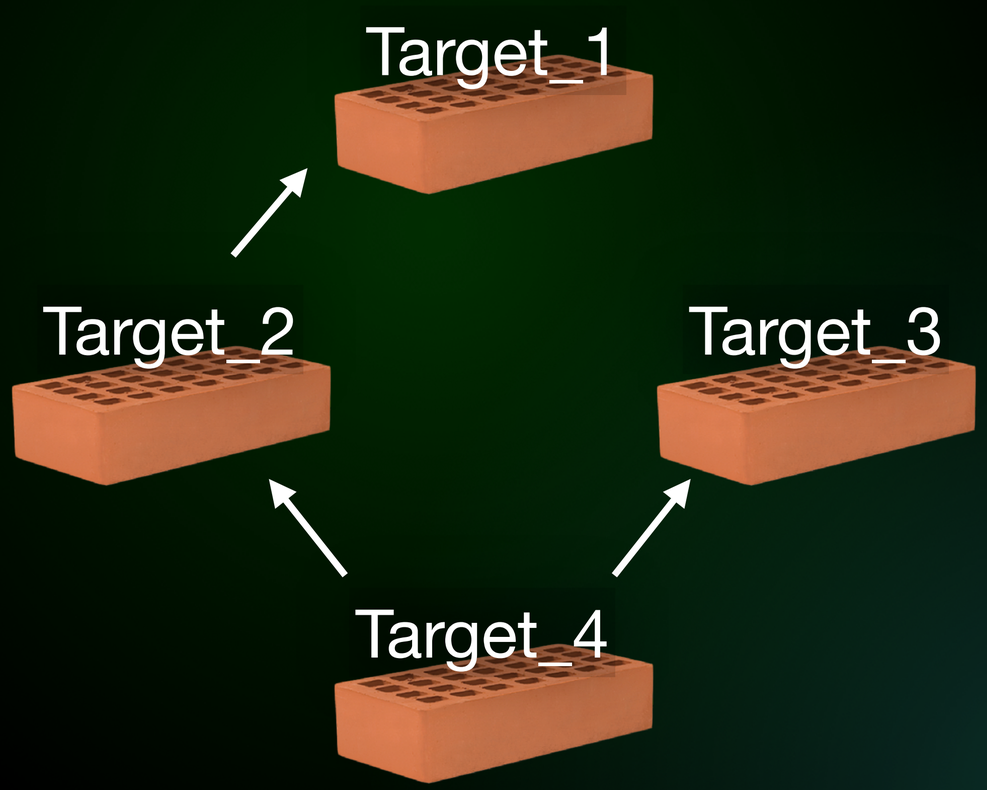

```
Target_4.snapshot_hash = Hash(
  Target_1.snapshot_hash +
  Target_2.snapshot_hash +
  Target_3.snapshot_hash +
  Target_4.sources +
  BuildParams
)
```

Механизм сборки



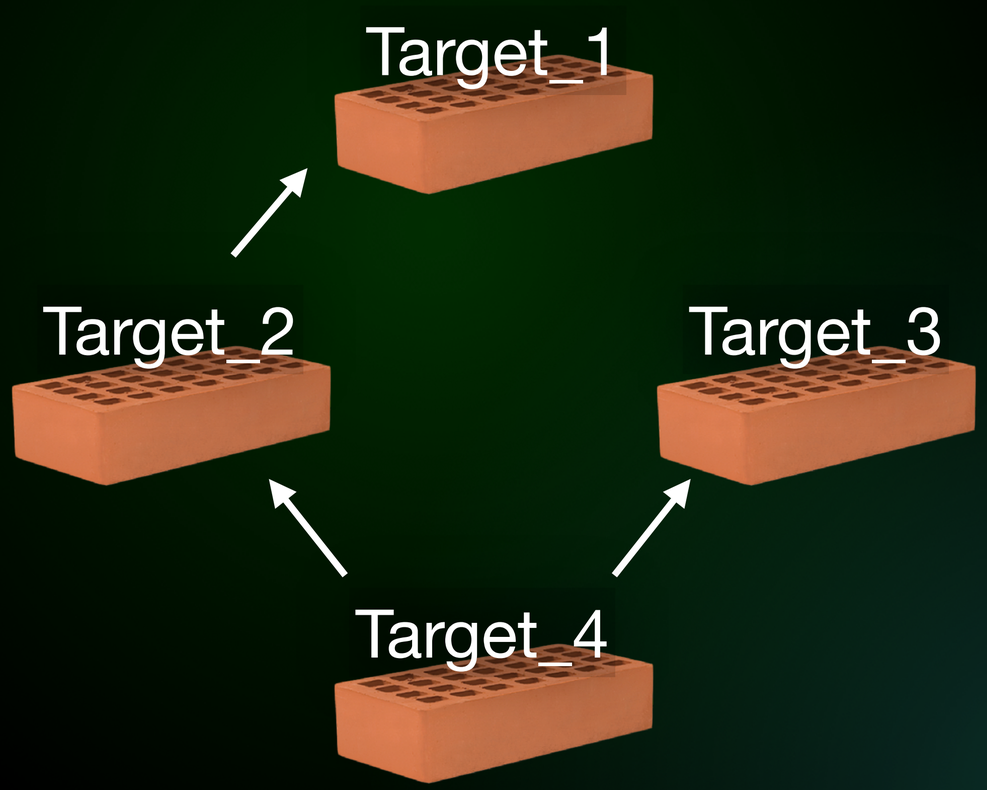

```
Target_4.snapshot_hash = Hash(
  Target_1.snapshot_hash +
  Target_2.snapshot_hash +
  Target_3.snapshot_hash +
  Target_4.sources +
  BuildParams
)
```

Механизм сборки



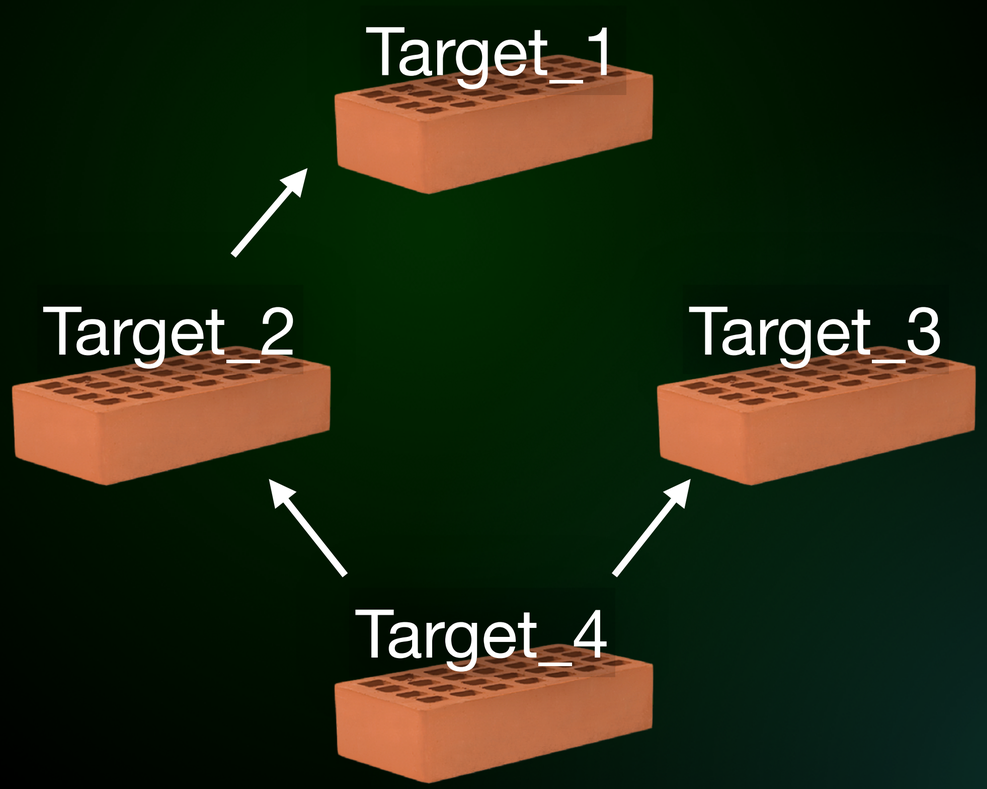

```
Target_4.snapshot_hash = Hash(
  Target_1.snapshot_hash +
  Target_2.snapshot_hash +
  Target_3.snapshot_hash +
  Target_4.sources +
  BuildParams
)
```

Механизм сборки



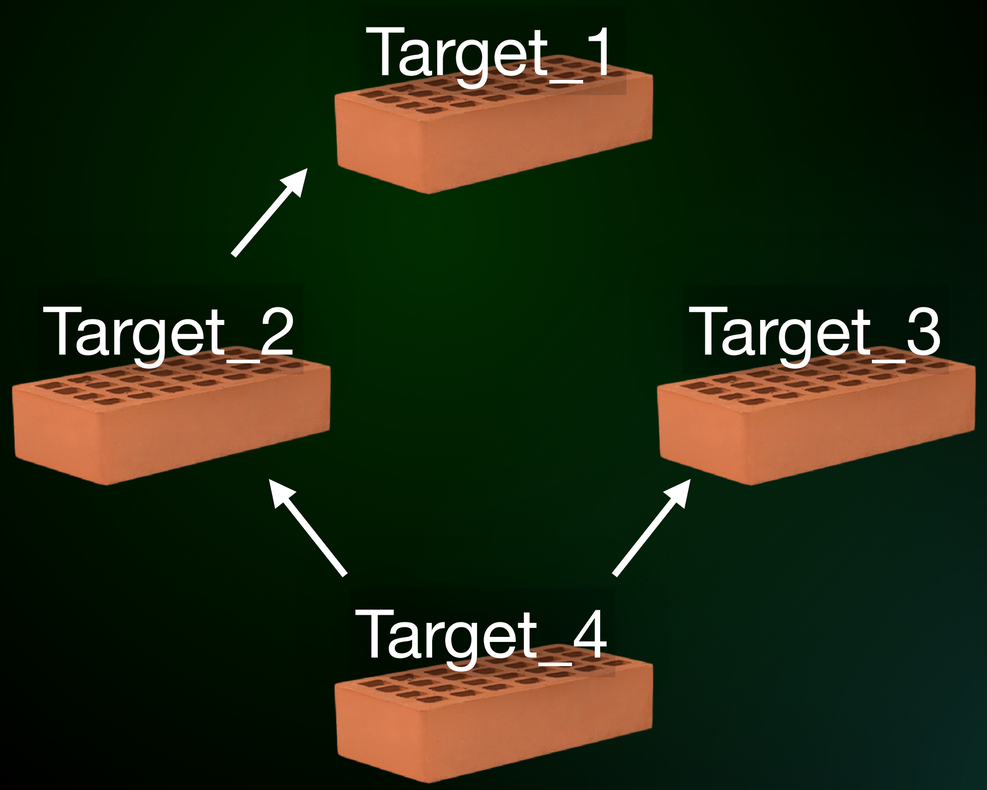

```
Target_4.snapshot_hash = Hash(
  Target_1.snapshot_hash +
  Target_2.snapshot_hash +
  Target_3.snapshot_hash +
  Target_4.sources +
  BuildParams
)
```

Механизм сборки



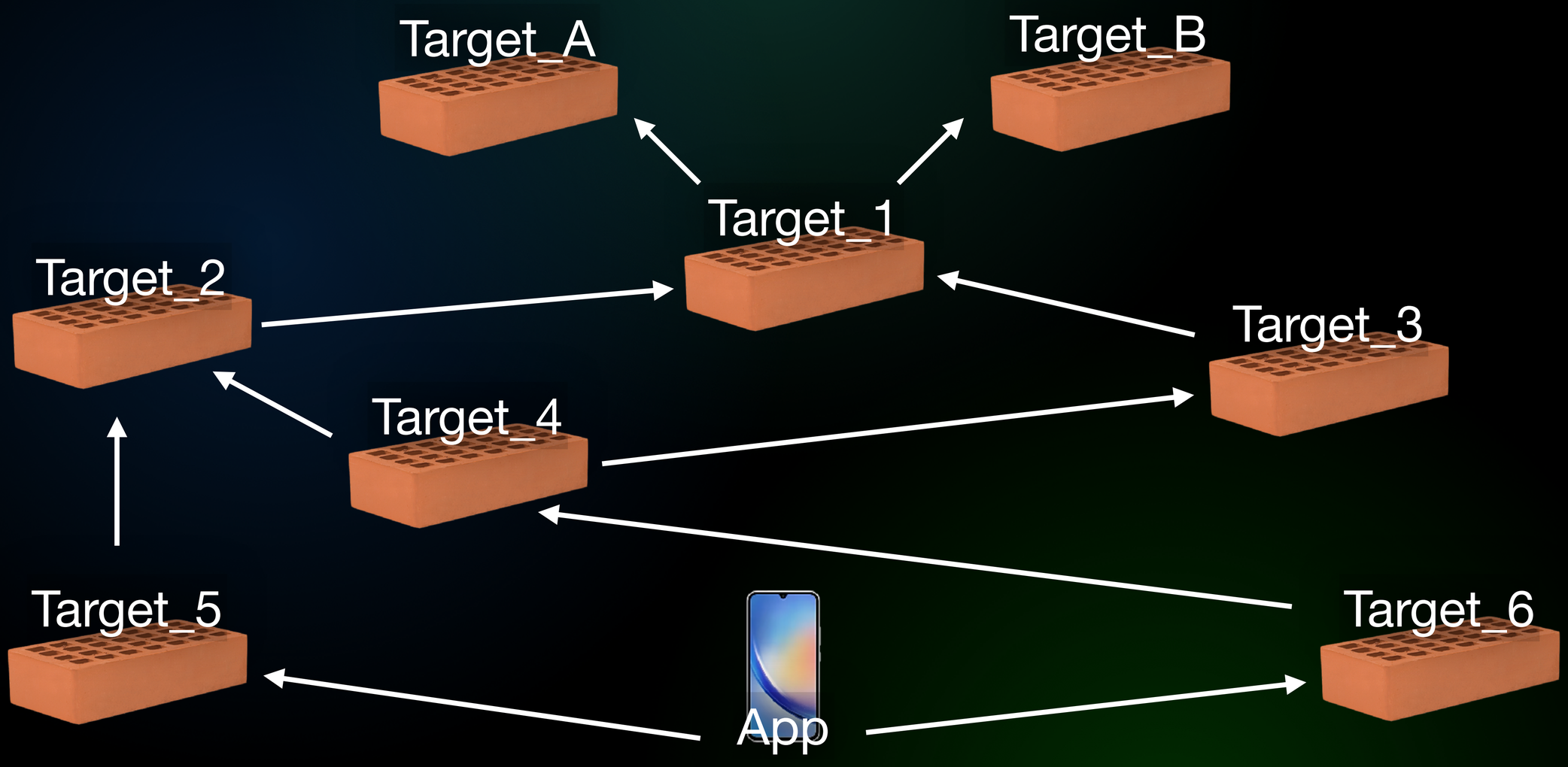

```
Target_4.snapshot_hash = Hash(
  Target_1.snapshot_hash +
  Target_2.snapshot_hash +
  Target_3.snapshot_hash +
  Target_4.sources +
  BuildParams
)
```

Механизм сборки

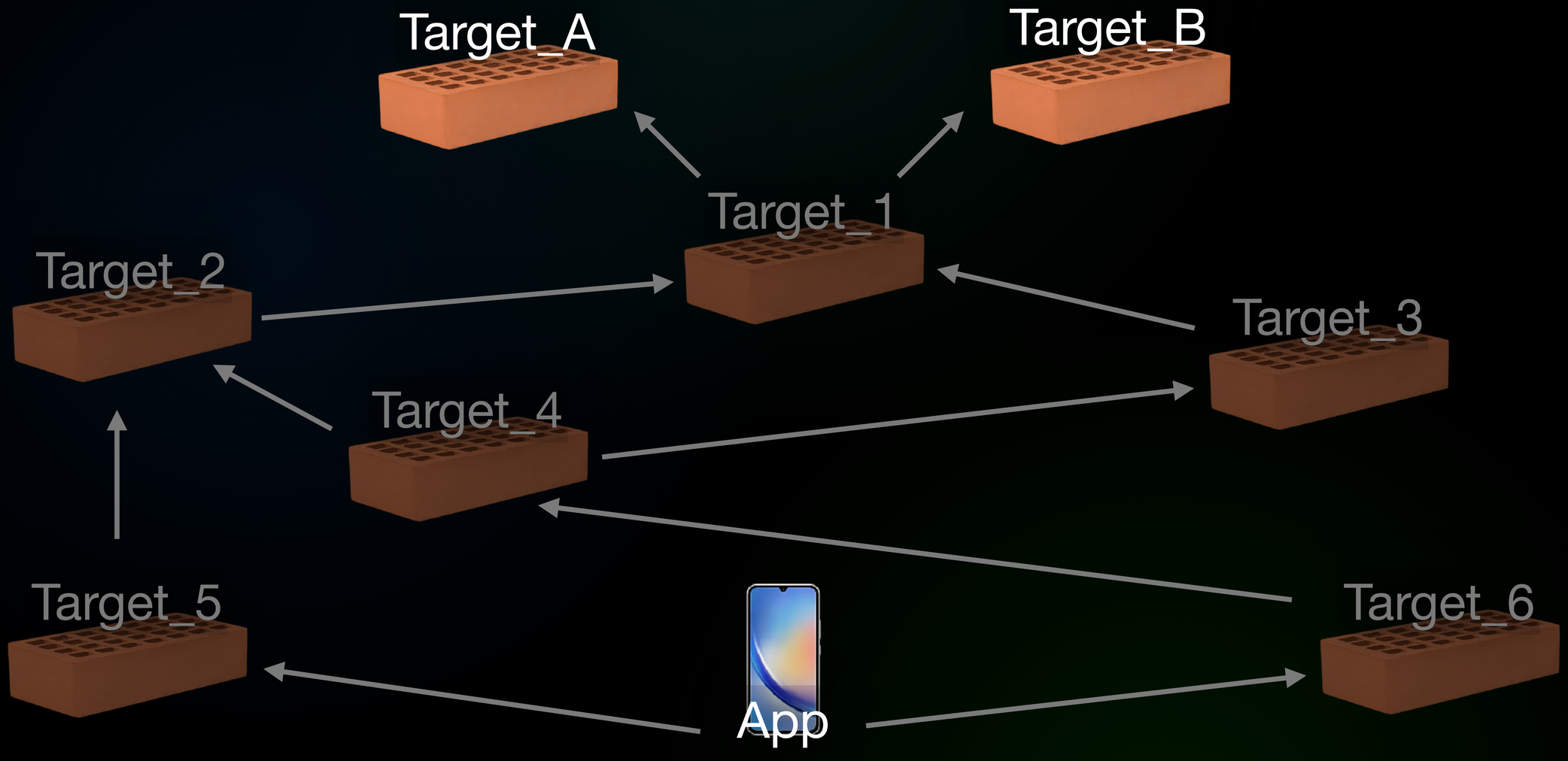



```
Target_4.snapshot_hash = Hash(
  Target_1.snapshot_hash +
  Target_2.snapshot_hash +
  Target_3.snapshot_hash +
  Target_4.sources +
  BuildParams
)
```

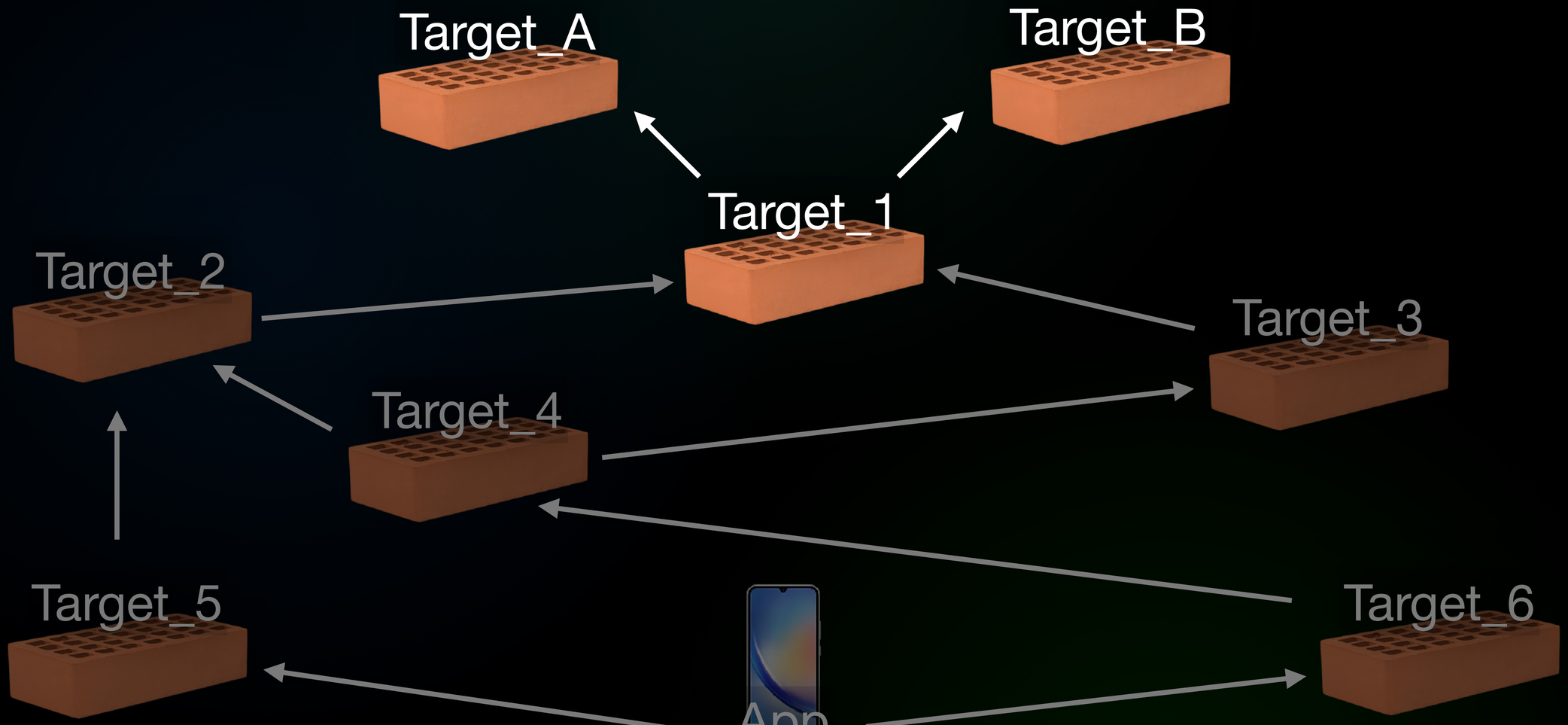
Механизм сборки



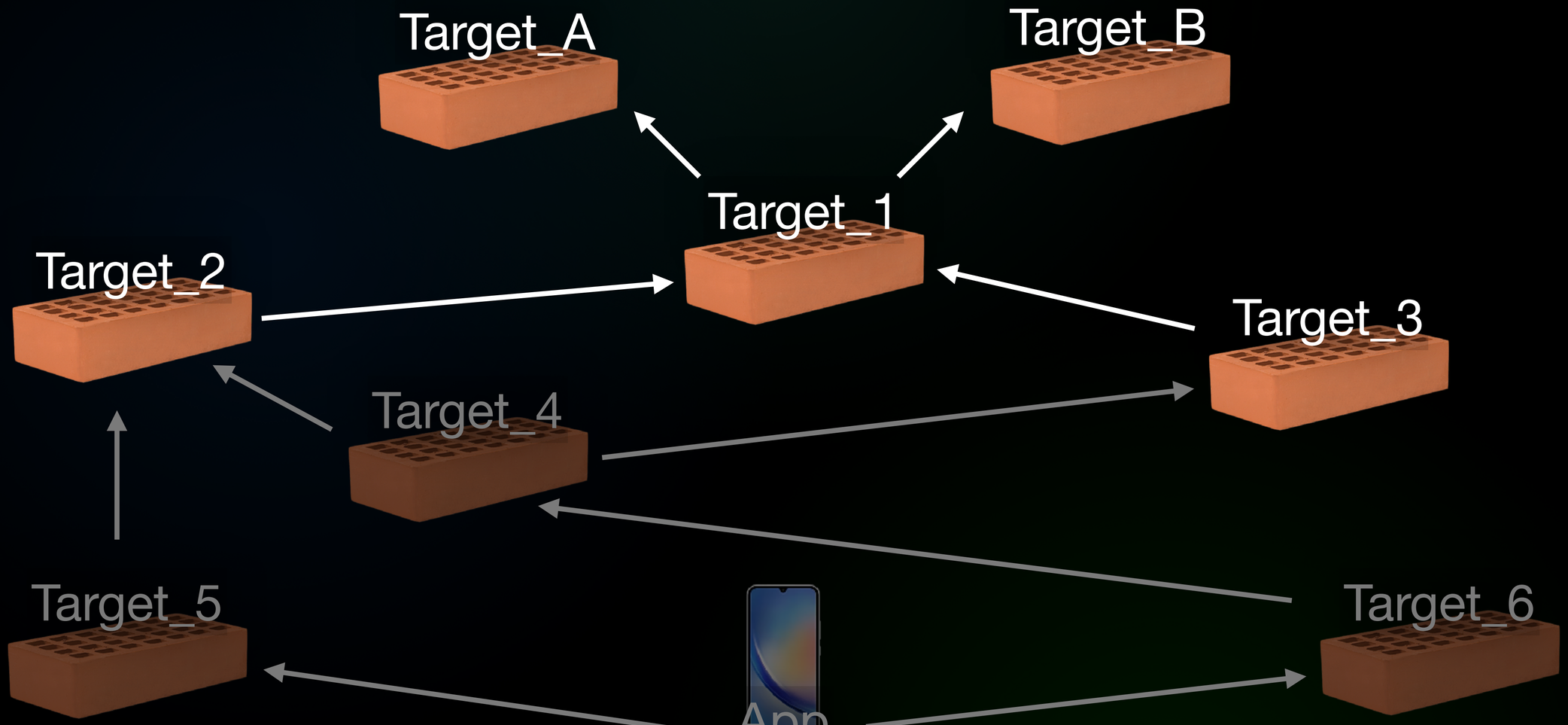
Механизм сборки



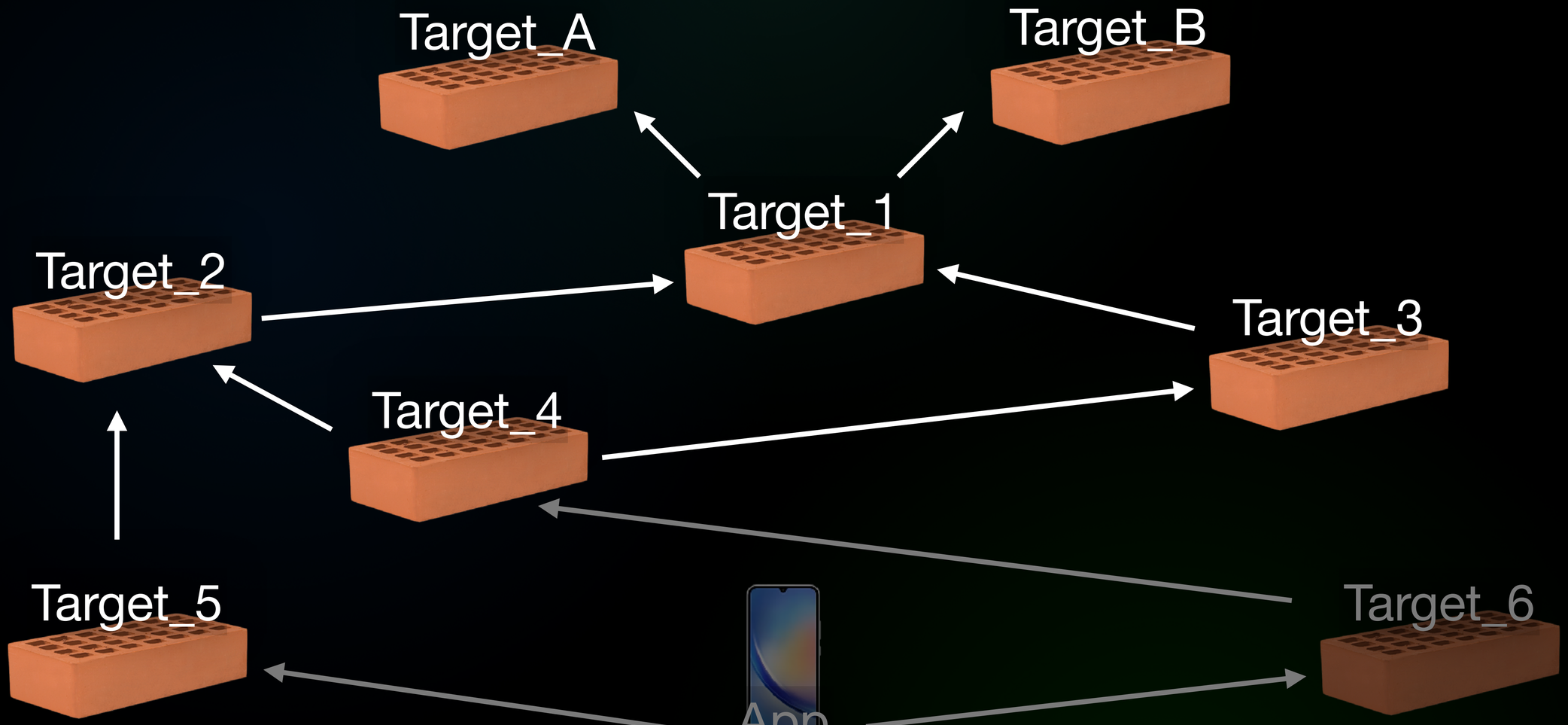
Механизм сборки



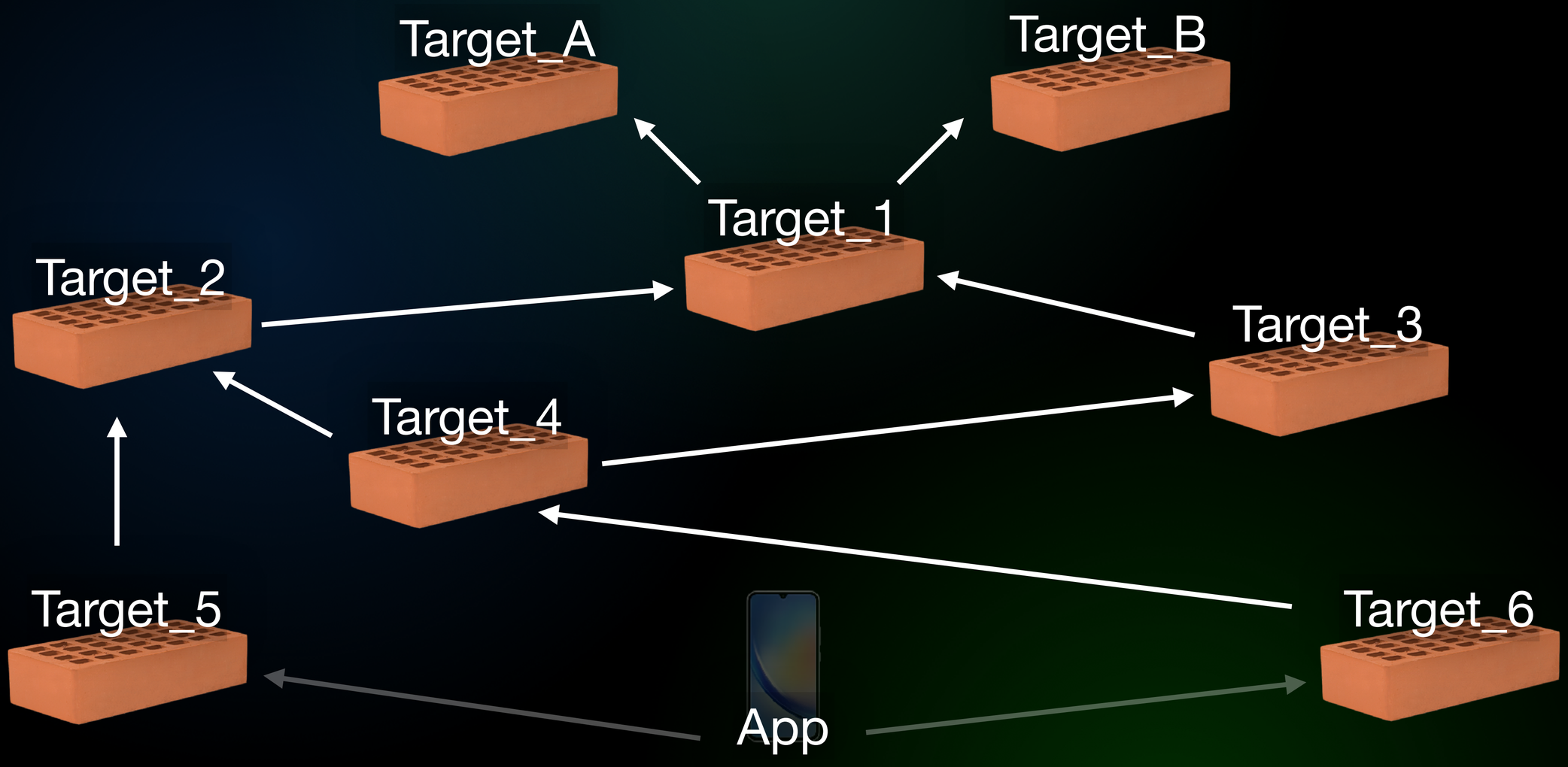
Механизм сборки



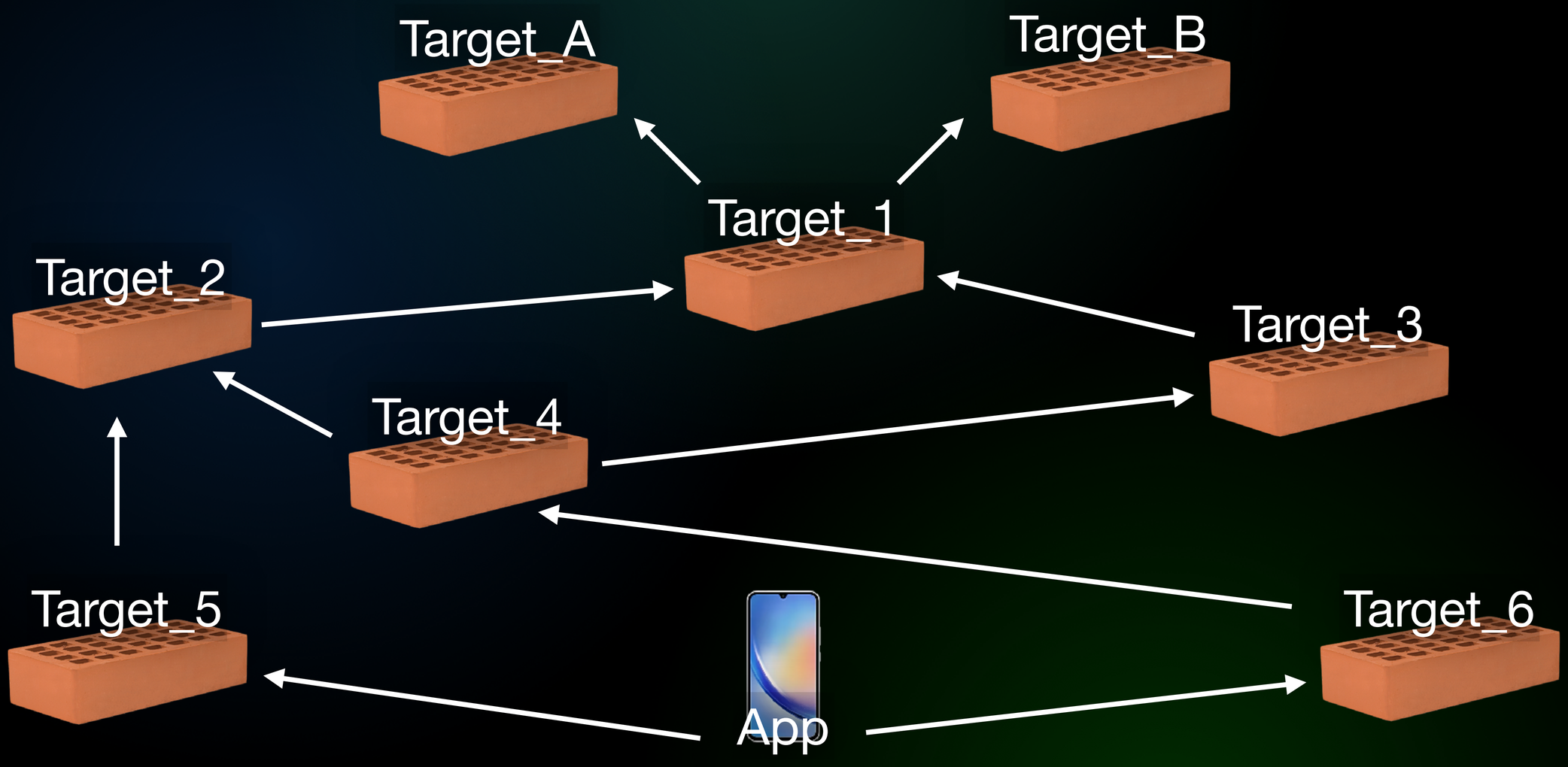
Механизм сборки



Механизм сборки



Механизм сборки



Новый механизм сборки

До

~1,5 часа

Новый механизм сборки

До

~1,5 часа

После

От 5 минут до 1,5 часа

Описание сборки

Описание сборки

Учитываем ресурсы

Описание сборки

Учитываем ресурсы

Используем по-слойный механизм

Описание сборки

Учитываем ресурсы

Используем по-слойный механизм

Результат – один билд

И немного ещё...

SwiftSyntax.Что это?

Анализировать

Проверять

Генерировать

Преобразовывать исходный код Swift

SwiftSyntax.Что это?

Анализировать

Проверять

Генерировать

Преобразовывать исходный код Swift

SwiftSyntax.Что это?

Анализировать

Проверять

Генерировать

Преобразовывать исходный код Swift

SwiftSyntax.Что это?

Анализировать

Проверять

Генерировать

Преобразовывать исходный код Swift

SwiftSyntax.Что это?

Вероника Макаровская
Статический анализатор техдолга

Проект SwiftSyntax

Новый алгоритм проверки наличия публичных изменений

1. Находим все public объекты в коде
2. Считаем их checksum
3. Сравниваем на PR-е полученную checksum с checksum из технического файла

Новый алгоритм проверки наличия публичных изменений

1. Находим все public объекты в коде
2. Считаем их checksum
3. Сравниваем на PR-е полученную checksum с checksum из технического файла

Новый алгоритм проверки наличия публичных изменений

1. Находим все public объекты в коде
2. Считаем их checksum
3. Сравниваем на PR-е полученную checksum с checksum из технического файла

Новый алгоритм проверки наличия публичных изменений

```
func checkModifiers(_ modifiers: ModifierListSyntax?, decl: DeclSyntaxProtocol, identifier: String) {
    if (modifiers != nil && modifiers!.contains(where: { ($0.firstToken != nil) &&
        $0.firstToken?.tokenKind == .publicKeyword })) {
        declarations.append(PublicDeclarations(identifier: identifier, decl: decl))
    }
}
```

Новый алгоритм проверки наличия публичных изменений

```
nc checkModifiers(_ modifiers: ModifierListSyntax?, decl: DeclSyntaxProtocol, iden
  if (modifiers != nil && modifiers!.contains(where: { ($0.firstToken != nil) &&
      $0.firstToken?.tokenKind == .publicKeyword })) {
      declarations.append(PublicDeclarations(identifier: identifier, decl: decl))
  }
```

Новый алгоритм проверки наличия публичных изменений

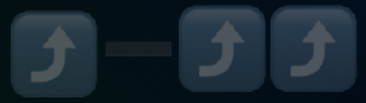
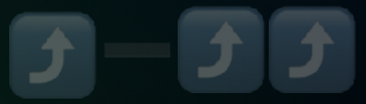
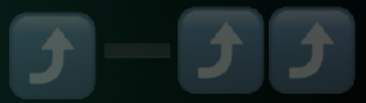
```
public class InterfaceVisitor: SyntaxAnyVisitor {
    var declarations = [PublicDeclarations]()

    func checkModifiers(_ modifiers: ModifierListSyntax?, decl: DeclSyntaxProtocol, identifier: String) {…
    }

    public override func visit(_ node: FunctionDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: EnumDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: StructDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: ClassDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: ProtocolDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: VariableDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: TypealiasDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: InitializerDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: DeinitializerDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: ExtensionDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: OperatorDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: PrecedenceGroupDeclSyntax) -> SyntaxVisitorContinueKind {…
    }
}
```

Новый алгоритм проверки наличия публичных изменений

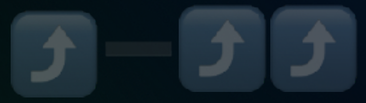
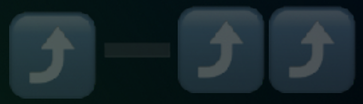
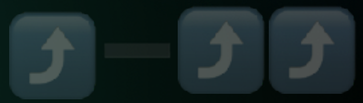
```
public class InterfaceVisitor: SyntaxAnyVisitor {
    var declarations = [PublicDeclarations]()

    func checkModifiers(_ modifiers: ModifierListSyntax?, decl: DeclSyntaxProtocol, identifier: String) {…
    }

    public override func visit(_ node: FunctionDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: EnumDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: StructDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: ClassDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: ProtocolDeclSyntax) -> SyntaxVisitorContinueKind {…
    }

    public override func visit(_ node: VariableDeclSyntax) -> SyntaxVisitorContinueKind {…
    }
```

Итоги:
нужна ли запятая?

Эпохи

Эпохи	Число билдов	Время сборки	Риски	Хранение технической информации	Работа в мультирепозитории
Первая эпоха	1			❌	❌
Вторая эпоха	2				✅
Новая эпоха	1			✅	✅
Новая эпоха со SwiftSyntax	1			✅	✅

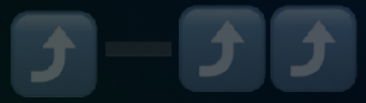
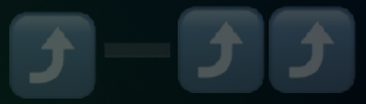
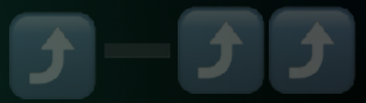
Эпохи

Эпохи	Число билдов	Время сборки	Риски	Хранение технической информации	Работа в мультирепозитории
Первая эпоха	1			❌	❌
Вторая эпоха	2			📦	✅
Новая эпоха	1			✅	✅
Новая эпоха со SwiftSyntax	1			✅	✅

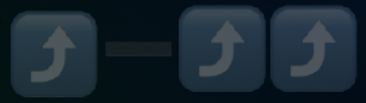
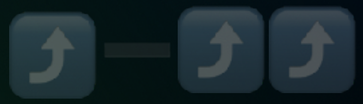
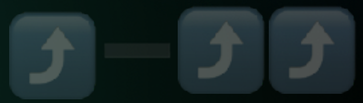
Эпохи

Эпохи	Число билдов	Время сборки	Риски	Хранение технической информации	Работа в мультирепозитории
Первая эпоха	1				
Вторая эпоха	2				
Новая эпоха	1				
Новая эпоха со SwiftSyntax	1				

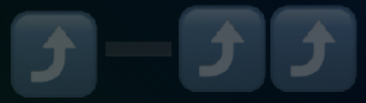
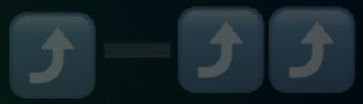
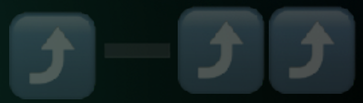
Эпохи

Эпохи	Число билдов	Время сборки	Риски	Хранение технической информации	Работа в мультирепозитории
Первая эпоха	1			✗	✗
Вторая эпоха	2				✓
Новая эпоха	1			✓	✓
Новая эпоха со SwiftSyntax	1			✓	✓

Эпохи

Эпохи	Число билдов	Время сборки	Риски	Хранение технической информации	Работа в мультирепозитории
Первая эпоха	1			❌	❌
Вторая эпоха	2			📦	✅
Новая эпоха	1			✅	✅
Новая эпоха со SwiftSyntax	1			✅	✅

Эпохи

Эпохи	Число билдов	Время сборки	Риски	Хранение технической информации	Работа в мультирепозитории
Первая эпоха	1			❌	❌
Вторая эпоха	2				✅
Новая эпоха	1			✅	✅
Новая эпоха со SwiftSyntax	1			✅	✅

Итоги

1

Стабильность CI
как бинарная, так и исходниками.

2

Стабильность локальной сборки
после обновления окружения проекта. Стабильность тестов.

3

Довольный разработчик:
предсказуемость стабильности работы.

4

Довольный дежурный:
риски снижены, отвечает только на вопросы новеньких в проекте «Почему упал мой билд»?

5

Ускорение скорости сборки:
пересборка нужных таргетов, не перегружаем инфраструктуру банка, быстрое обновление окружения проекта локально и на CI.

6

Баланс скорости и надёжности.
Нет лишних билдов, предсказуемое время и очередь сборки.

Итоги

1

Стабильность CI
как бинарная, так и исходниками.

2

Стабильность локальной сборки
после обновления окружения проекта. Стабильность тестов.

3

Довольный разработчик:
предсказуемость стабильности работы.

4

Довольный дежурный:
риски снижены, отвечает только на вопросы новеньких в проекте «Почему упал мой билд»?

5

Ускорение скорости сборки:
пересборка нужных таргетов, не перегружаем инфраструктуру банка, быстрое обновление окружения проекта локально и на CI.

6

Баланс скорости и надёжности.
Нет лишних билдов, предсказуемое время и очередь сборки.

Итоги

1

Стабильность CI
как бинарная, так и исходниками.

2

Стабильность локальной сборки
после обновления окружения проекта. Стабильность тестов.

3

Довольный разработчик:
предсказуемость стабильности работы.

4

Довольный дежурный:
риски снижены, отвечает только на вопросы новеньких в проекте «Почему упал мой билд»?

5

Ускорение скорости сборки:
пересборка нужных таргетов, не перегружаем инфраструктуру банка, быстрое обновление окружения проекта локально и на CI.

6

Баланс скорости и надёжности.
Нет лишних билдов, предсказуемое время и очередь сборки.

Итоги

1

Стабильность CI
как бинарная, так и исходниками.

2

Стабильность локальной сборки
после обновления окружения проекта. Стабильность тестов.

3

Довольный разработчик:
предсказуемость стабильности работы.

4

Довольный дежурный:
риски снижены, отвечает только на вопросы новеньких в проекте «Почему упал мой билд»?

5

Ускорение скорости сборки:
пересборка нужных таргетов, не перегружаем инфраструктуру банка, быстрое обновление окружения проекта локально и на CI.

6

Баланс скорости и надёжности.
Нет лишних билдов, предсказуемое время и очередь сборки.

Итоги

1

Стабильность CI
как бинарная, так и исходниками.

2

Стабильность локальной сборки
после обновления окружения проекта. Стабильность тестов.

3

Довольный разработчик:
предсказуемость стабильности работы.

4

Довольный дежурный:
риски снижены, отвечает только на вопросы новеньких в проекте «Почему упал мой билд»?

5

Ускорение скорости сборки:
пересборка нужных таргетов, не перегружаем инфраструктуру банка, быстрое обновление окружения проекта локально и на CI.

6

Баланс скорости и надёжности.
Нет лишних билдов, предсказуемое время и очередь сборки.

Итоги

1

Стабильность CI
как бинарная, так и исходниками.

2

Стабильность локальной сборки
после обновления окружения проекта. Стабильность тестов.

3

Довольный разработчик:
предсказуемость стабильности работы.

4

Довольный дежурный:
риски снижены, отвечает только на вопросы новеньких в проекте «Почему упал мой билд»?

5

Ускорение скорости сборки:
пересборка нужных таргетов, не перегружаем инфраструктуру банка, быстрое обновление окружения проекта локально и на CI.

6

Баланс скорости и надёжности.
Нет лишних билдов, предсказуемое время и очередь сборки.

Алгоритм

1

Определить, на какой число билдов вы согласны

2

Если вам нужен 1 билд, но вы не хотите хранить техническую информацию, то можно использовать скрипт проверки изменений в публичных файлах

3

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать swiftinterface

4

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать с помощью swift-api-digester

5

Если вам нужен 1 билд и вы готовы хранить техническую информацию, то можно использовать инструменты SwiftSyntax для получения публичного интерфейса и сравнивать их перед билдом

6

Для более точной работы impact-analyze нужно хранить техническую информацию о том, какие таргеты используют публичный интерфейс нашего таргета

Алгоритм

1

Определить, на какой число билдов вы согласны

2

Если вам нужен 1 билд, но вы не хотите хранить техническую информацию, то можно использовать скрипт проверки изменений в публичных файлах

3

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать swiftinterface

4

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать с помощью swift-api-digester

5

Если вам нужен 1 билд и вы готовы хранить техническую информацию, то можно использовать инструменты SwiftSyntax для получения публичного интерфейса и сравнивать их перед билдом

6

Для более точной работы impact-analyze нужно хранить техническую информацию о том, какие таргеты используют публичный интерфейс нашего таргета

Алгоритм

1

Определить, на какой число билдов вы согласны

2

Если вам нужен 1 билд, но вы не хотите хранить техническую информацию, то можно использовать скрипт проверки изменений в публичных файлах

3

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать swiftinterface

4

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать с помощью swift-api-digester

5

Если вам нужен 1 билд и вы готовы хранить техническую информацию, то можно использовать инструменты SwiftSyntax для получения публичного интерфейса и сравнивать их перед билдом

6

Для более точной работы impact-analyze нужно хранить техническую информацию о том, какие таргеты используют публичный интерфейс нашего таргета

Алгоритм

1

Определить, на какой число билдов вы согласны

2

Если вам нужен 1 билд, но вы не хотите хранить техническую информацию, то можно использовать скрипт проверки изменений в публичных файлах

3

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать swiftinterface

4

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать с помощью swift-api-digester

5

Если вам нужен 1 билд и вы готовы хранить техническую информацию, то можно использовать инструменты SwiftSyntax для получения публичного интерфейса и сравнивать их перед билдом

6

Для более точной работы impact-analyze нужно хранить техническую информацию о том, какие таргеты используют публичный интерфейс нашего таргета

Алгоритм

1

Определить, на какой число билдов вы согласны

2

Если вам нужен 1 билд, но вы не хотите хранить техническую информацию, то можно использовать скрипт проверки изменений в публичных файлах

3

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать swiftinterface

4

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать с помощью swift-api-digester

5

Если вам нужен 1 билд и вы готовы хранить техническую информацию, то можно использовать инструменты SwiftSyntax для получения публичного интерфейса и сравнивать их перед билдом

6

Для более точной работы impact-analyze нужно хранить техническую информацию о том, какие таргеты используют публичный интерфейс нашего таргета

Алгоритм

1

Определить, на какой число билдов вы согласны

2

Если вам нужен 1 билд, но вы не хотите хранить техническую информацию, то можно использовать скрипт проверки изменений в публичных файлах

3

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать swiftinterface

4

Если вам не нужен 1 билд и вас есть хранение артефактов таргетов, то можно собирать новый первым билдом, а перед вторым - сравнивать с помощью swift-api-digester

5

Если вам нужен 1 билд и вы готовы хранить техническую информацию, то можно использовать инструменты SwiftSyntax для получения публичного интерфейса и сравнивать их перед билдом

6

Для более точной работы impact-analyze нужно хранить техническую информацию о том, какие таргеты используют публичный интерфейс нашего таргета

Переиспользовать нельзя перепроверять

Переиспользовать нельзя перепроверять

Где вы поставите запятую? ☺

И
Переиспользовать ~~нельзя~~ перепроверять

Где вы поставите запятую? ☺

🙏 за ваше внимание 🙏

📲 Контакты

Вероника Макаровская

✉️ makarovskaya.v.m@gmail.com
🧬 Команда iOS-MobilePlatform Сбера
✈️ tg: MVeronika