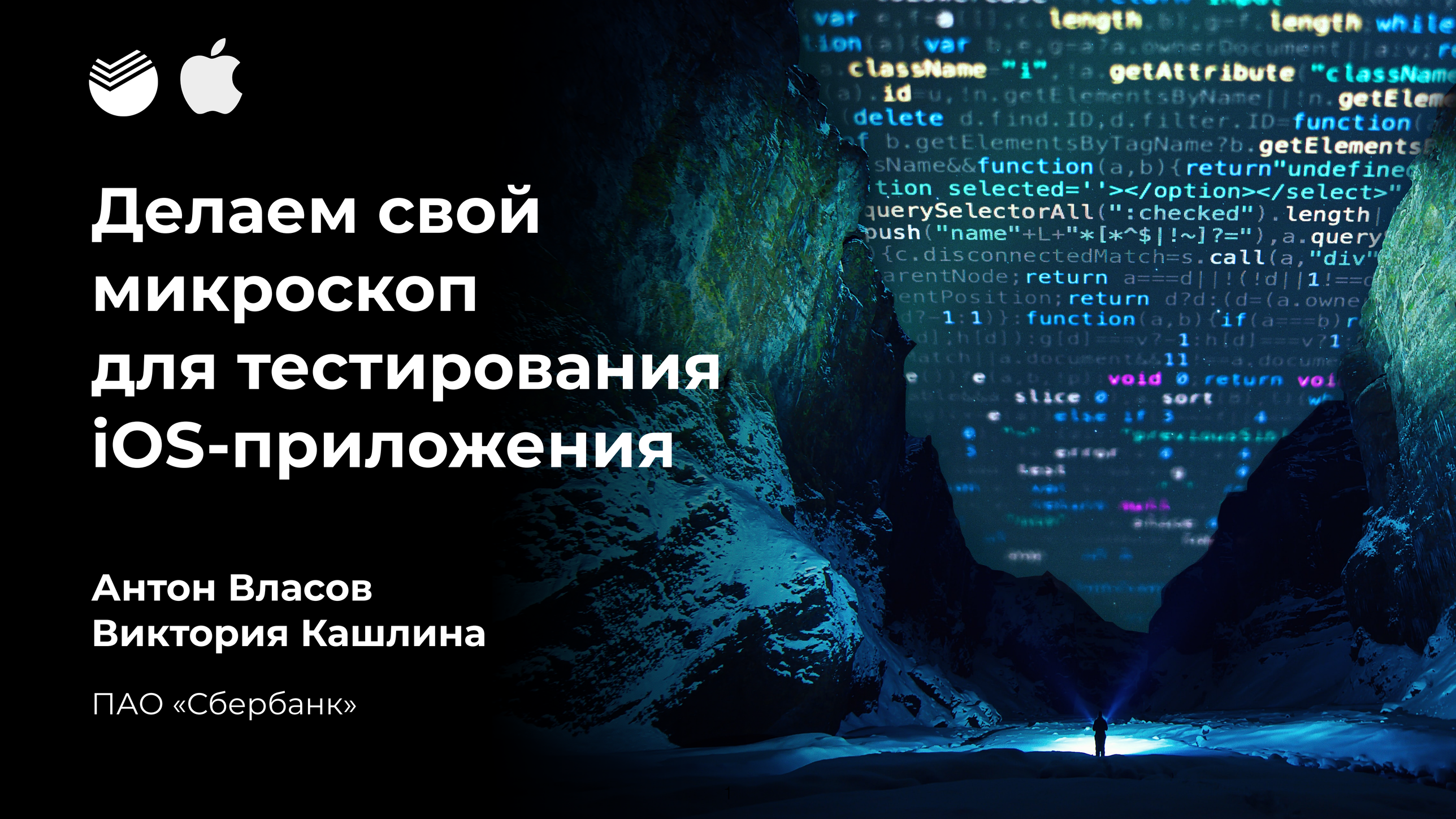


Делаем свой микроскоп для тестирования iOS-приложения

Антон Власов
Виктория Кашлина

ПАО «Сбербанк»

Для кого этот доклад

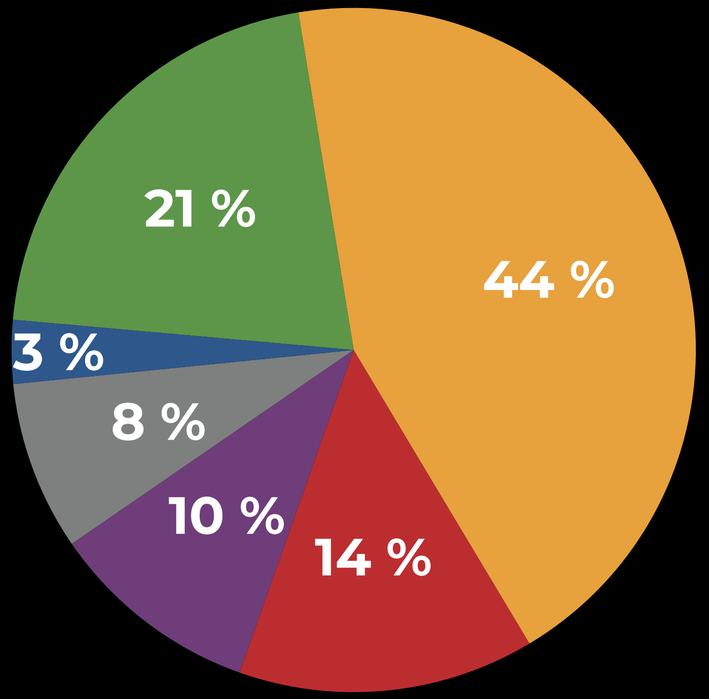

Тестировщики
Разработчики
Владельцы продукта

Instruments

Xcode

iOS, watchOS, tvOS, macOS, iPadOS

Как часто вы используете Xcode Instruments?



*опрошено

146 разрабов

разного уровня

- Стараюсь профилировать каждую задачу
- 1-2 раза в месяц
- При починке багов
- Использовал на другом проекте, сейчас нет
- Ни разу не запускал
- Не знаю что это

Зачем вам этот доклад

Что такое Performance согласно Apple

Узнать как правильно и качественно мерить перфоманс приложения

1

Что такое инструменты Xcode
и как с ними работать

2

Профилируем
реальные кейсы

3

Делаем
свой инструмент

4

Автоматизируем
инструменты на CI


Level
1

Задача:

Понять, что происходит в приложении с технической стороны

Узнаете:

1. О возможностях с инструментами Xcode
2. О профилировании приложения без исходного кода

Когда применять:

- Чем раньше, тем лучше
- Чем чаще, тем лучше

Общие ресурсы

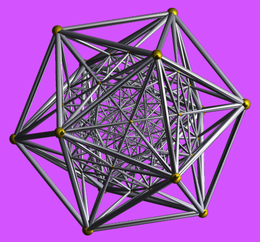
Сбербанк Онлайн

Основные группы

Memory

Perfomance

Energy



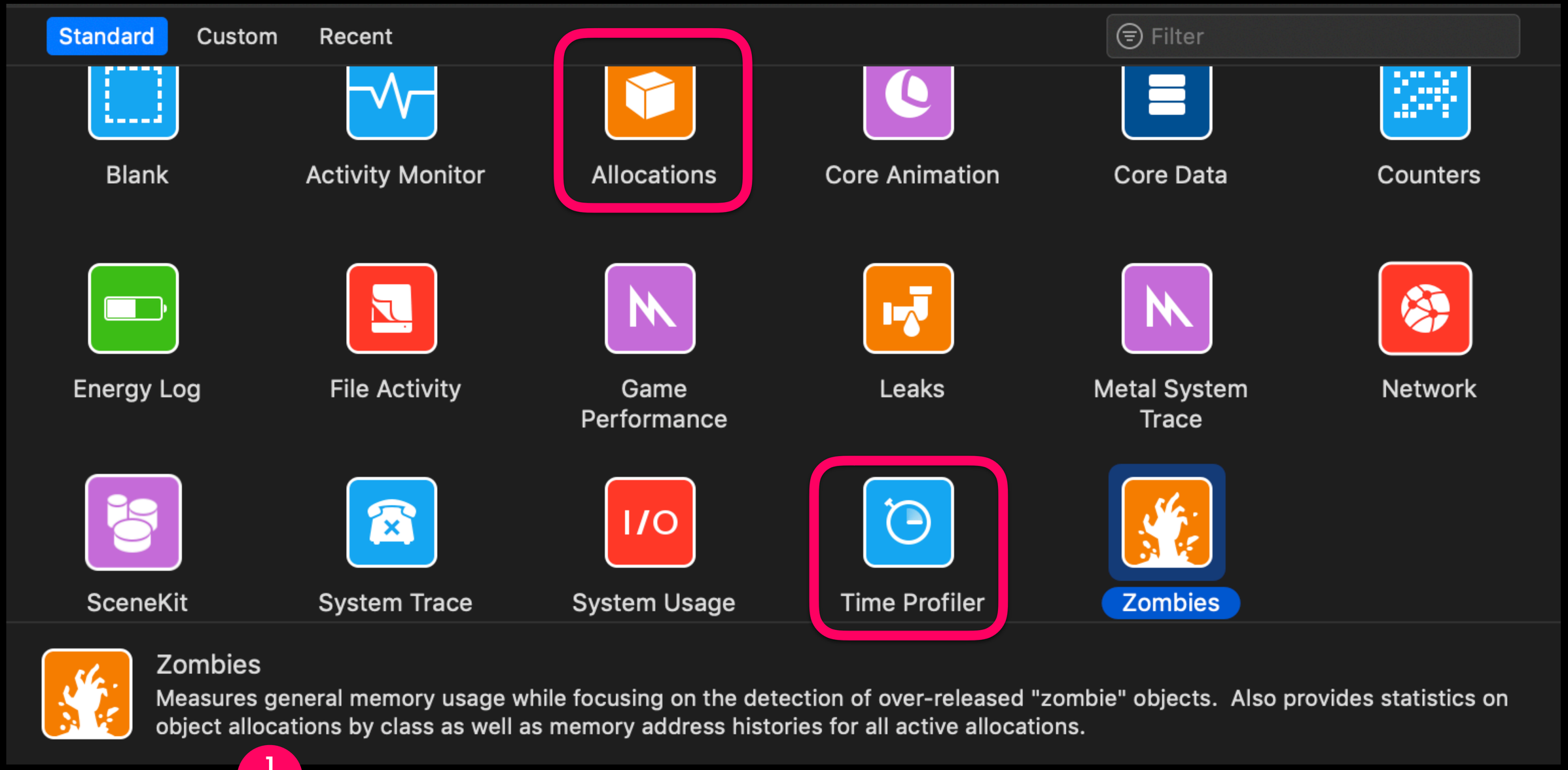
GPU

macOS perfomance

Core Data

Выбираем инструмент

⌘+i



DemoFox - Apple sample project

Кейс: лаги в игре

iOS. Проанализировать выделение памяти

Estimate: **1**

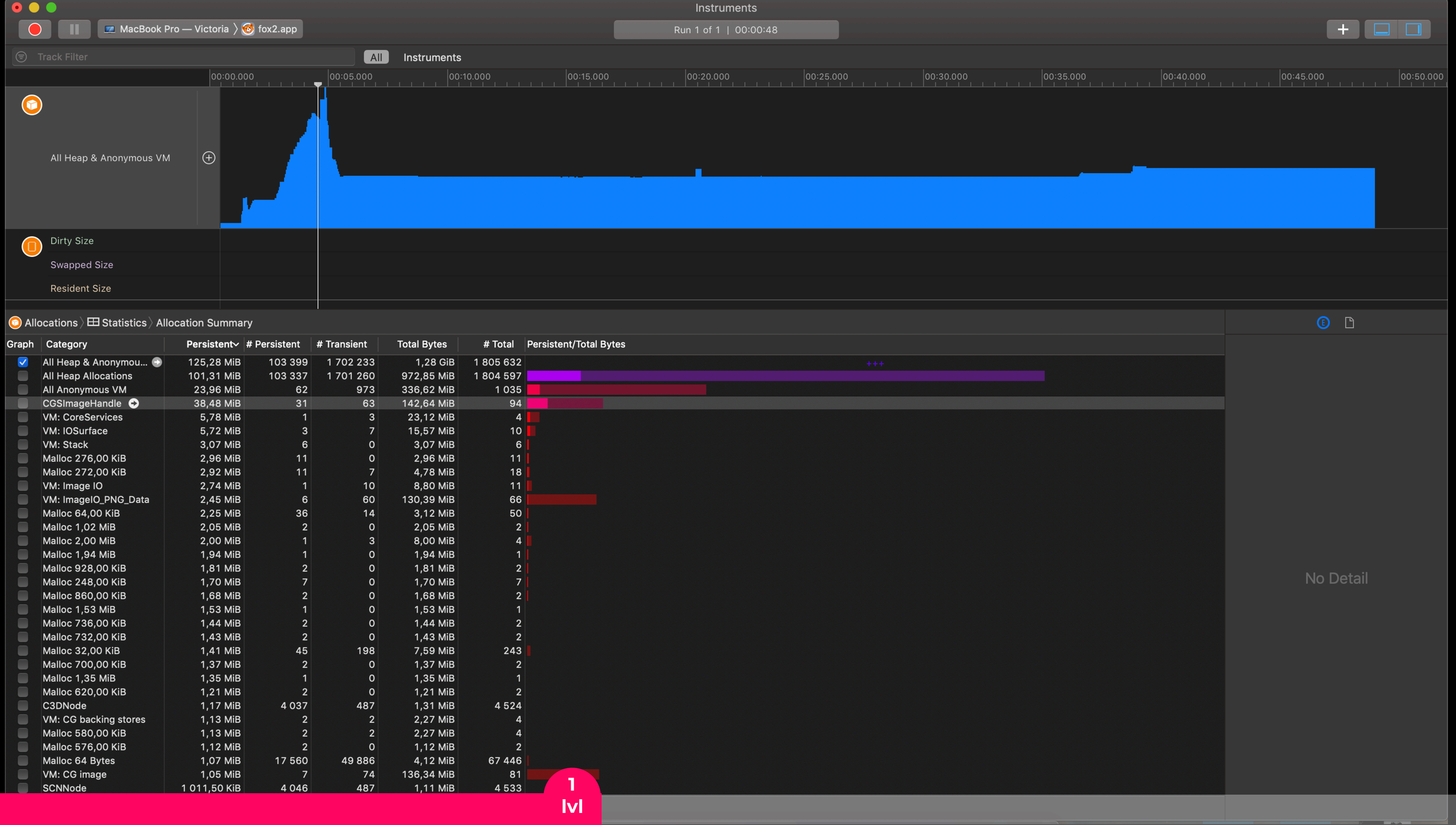
Priority: **Minor**

⌘+i

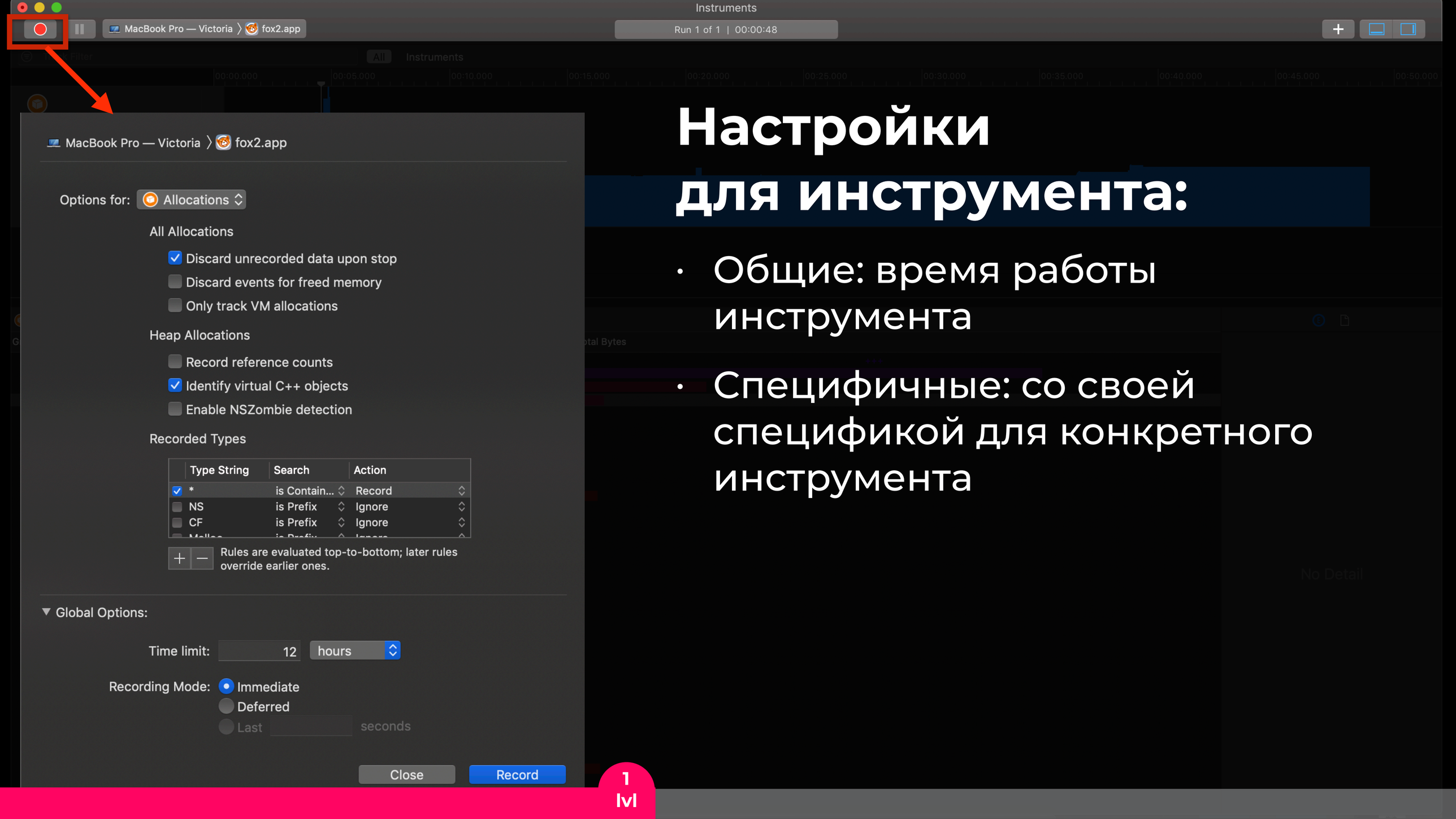
Allocations

Отображает статистику
по аллоцированию памяти
во время работы приложения

Видос

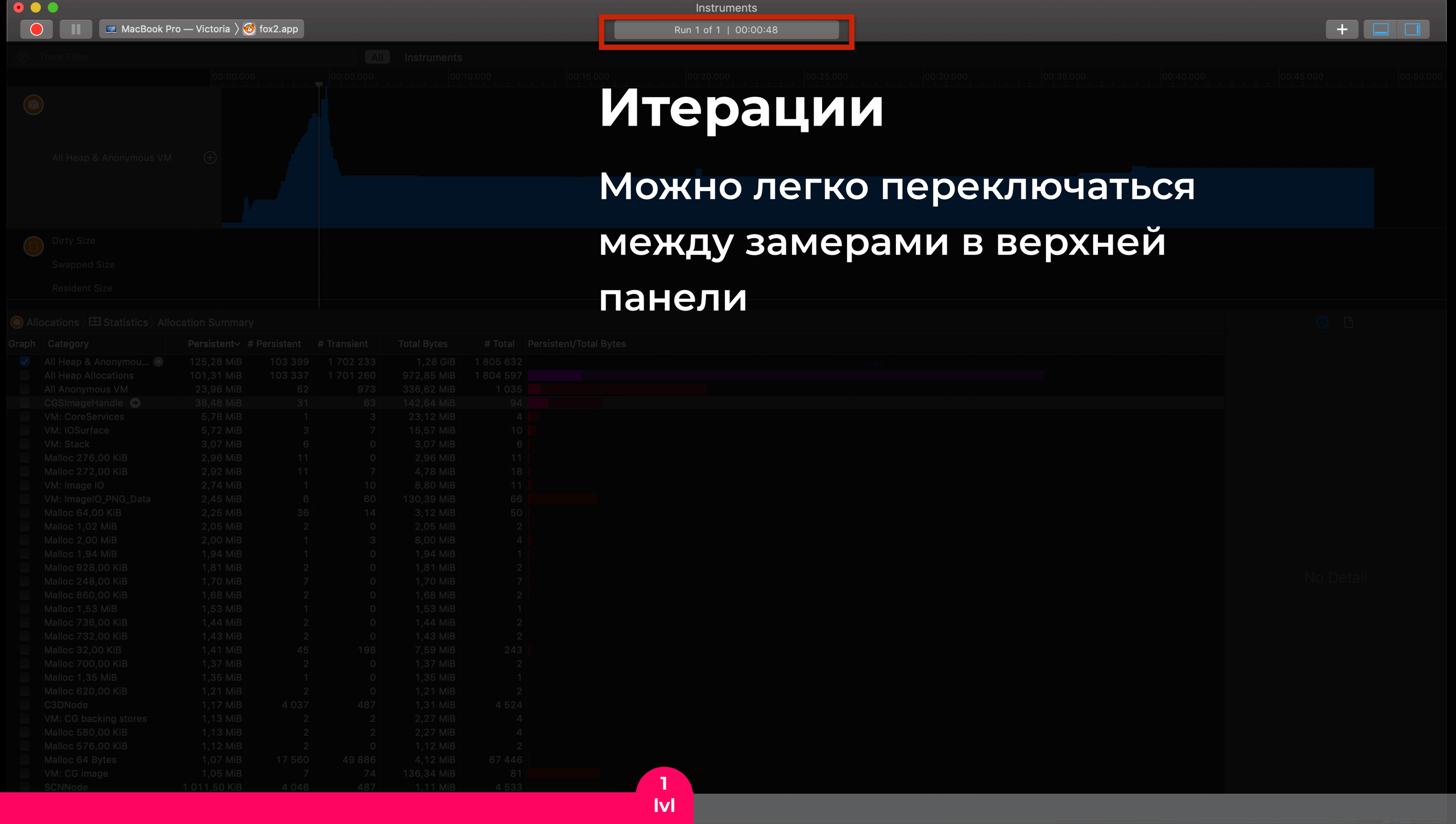


Graph	Category	Persistent	# Persistent	# Transient	Total Bytes	# Total	Persistent/Total Bytes
☑	All Heap & Anonymou…	125,28 MiB	103 399	1 702 233	1,28 GiB	1 805 632	
☐	All Heap Allocations	101,31 MiB	103 337	1 701 260	972,85 MiB	1 804 597	
☐	All Anonymous VM	23,96 MiB	62	973	336,62 MiB	1 035	
☐	CGSImageHandle	38,48 MiB	31	63	142,64 MiB	94	
☐	VM: CoreServices	5,78 MiB	1	3	23,12 MiB	4	
☐	VM: IOSurface	5,72 MiB	3	7	15,57 MiB	10	
☐	VM: Stack	3,07 MiB	6	0	3,07 MiB	6	
☐	Malloc 276,00 KiB	2,96 MiB	11	0	2,96 MiB	11	
☐	Malloc 272,00 KiB	2,92 MiB	11	7	4,78 MiB	18	
☐	VM: Image IO	2,74 MiB	1	10	8,80 MiB	11	
☐	VM: ImageIO_PNG_Data	2,45 MiB	6	60	130,39 MiB	66	
☐	Malloc 64,00 KiB	2,25 MiB	36	14	3,12 MiB	50	
☐	Malloc 1,02 MiB	2,05 MiB	2	0	2,05 MiB	2	
☐	Malloc 2,00 MiB	2,00 MiB	1	3	8,00 MiB	4	
☐	Malloc 1,94 MiB	1,94 MiB	1	0	1,94 MiB	1	
☐	Malloc 928,00 KiB	1,81 MiB	2	0	1,81 MiB	2	
☐	Malloc 248,00 KiB	1,70 MiB	7	0	1,70 MiB	7	
☐	Malloc 860,00 KiB	1,68 MiB	2	0	1,68 MiB	2	
☐	Malloc 1,53 MiB	1,53 MiB	1	0	1,53 MiB	1	
☐	Malloc 736,00 KiB	1,44 MiB	2	0	1,44 MiB	2	
☐	Malloc 732,00 KiB	1,43 MiB	2	0	1,43 MiB	2	
☐	Malloc 32,00 KiB	1,41 MiB	45	198	7,59 MiB	243	
☐	Malloc 700,00 KiB	1,37 MiB	2	0	1,37 MiB	2	
☐	Malloc 1,35 MiB	1,35 MiB	1	0	1,35 MiB	1	
☐	Malloc 620,00 KiB	1,21 MiB	2	0	1,21 MiB	2	
☐	C3DNode	1,17 MiB	4 037	487	1,31 MiB	4 524	
☐	VM: CG backing stores	1,13 MiB	2	2	2,27 MiB	4	
☐	Malloc 580,00 KiB	1,13 MiB	2	2	2,27 MiB	4	
☐	Malloc 576,00 KiB	1,12 MiB	2	0	1,12 MiB	2	
☐	Malloc 64 Bytes	1,07 MiB	17 560	49 886	4,12 MiB	67 446	
☐	VM: CG image	1,05 MiB	7	74	136,34 MiB	81	
☐	SCNNode	1 011,50 KiB	4 046	487	1,11 MiB	4 533	



Настройки для инструмента:

- Общие: время работы инструмента
- Специфичные: со своей спецификой для конкретного инструмента

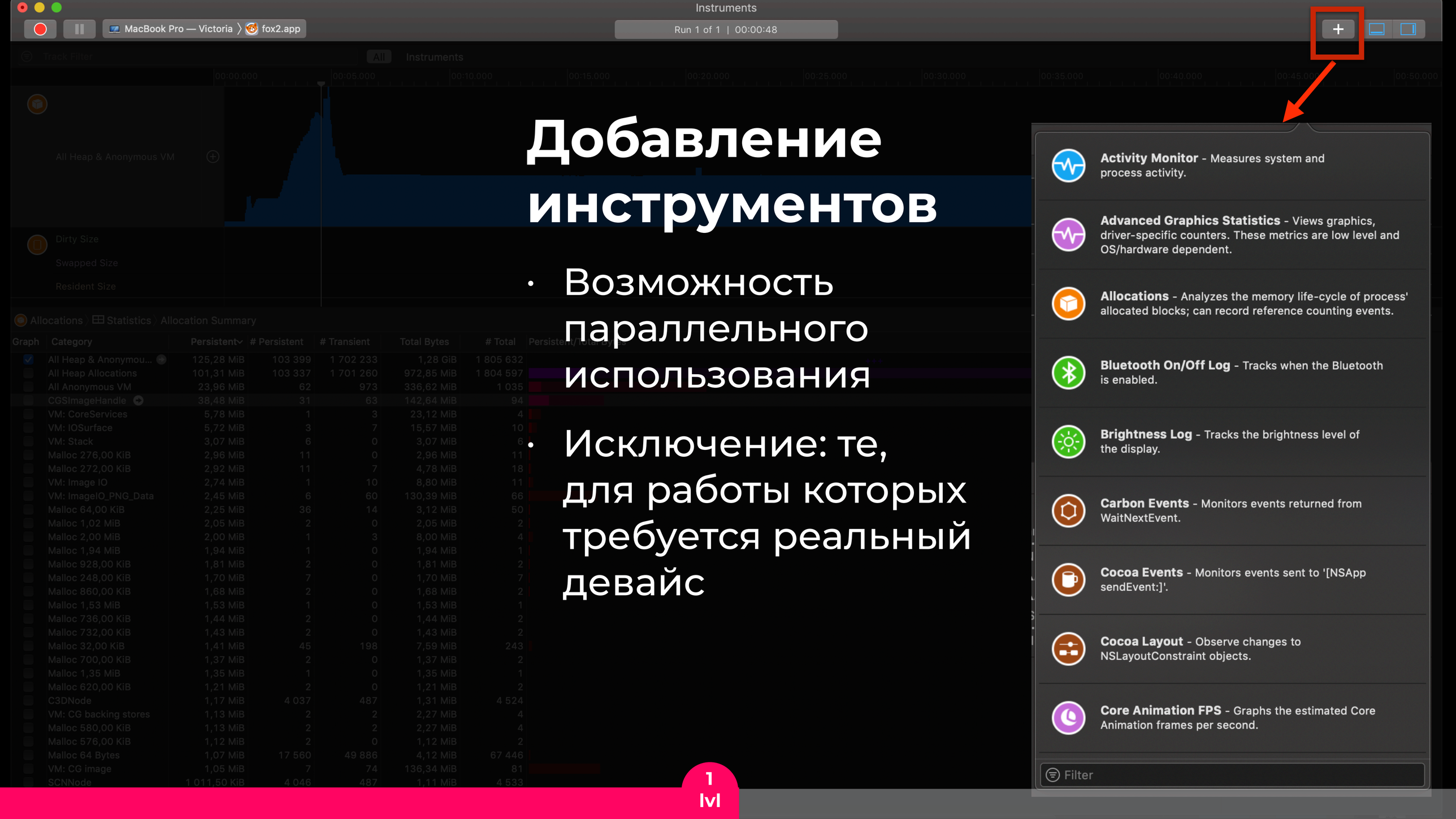


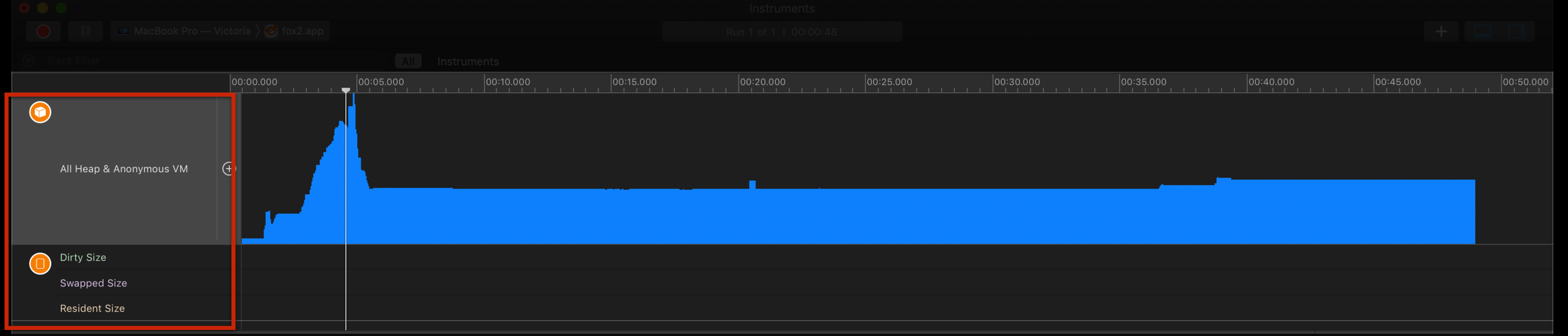
Итерации

Можно легко переключаться между замерами в верхней панели

Добавление инструментов

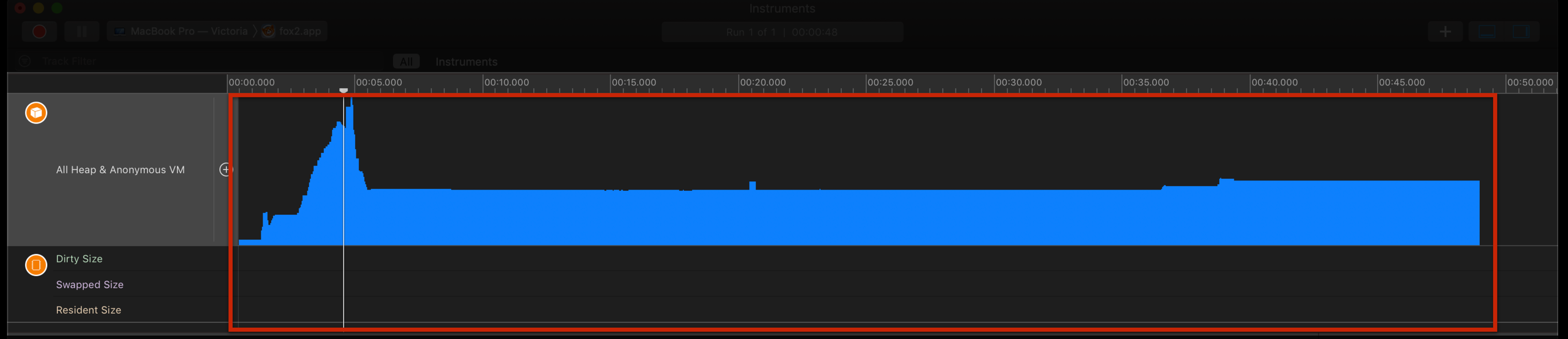
- Возможность параллельного использования
- Исключение: те, для работы которых требуется реальный девайс





All Heap & Anonymous VM

Сумма аллоцированной
операционной системой памяти
в “куче” и виртуальной памяти



Track

- Выбран только один инструмент
- График отображает аллоцирование памяти

Allocation summary

Показания по всем данным, собранным инструментом за прогон процесса

Graph	Category	Persistent	# Persistent	# Transient	Total Bytes	# Total	Persistent/Total Bytes
☑	All Heap & Anonymou…	125,28 MiB	103 399	1 702 233	1,28 GiB	1 805 632	
☐	All Heap Allocations	101,31 MiB	103 337	1 701 260	972,85 MiB	1 804 597	
☐	All Anonymous VM	23,96 MiB	62	973	336,62 MiB	1 035	
☐	CGSImageHandle	38,48 MiB	31	63	142,64 MiB	94	
☐	VM: CoreServices	5,78 MiB	1	3	23,12 MiB	4	
☐	VM: IOSurface	5,72 MiB	3	7	15,57 MiB	10	
☐	VM: Stack	3,07 MiB	6	0	3,07 MiB	6	
☐	Malloc 276,00 KiB	2,96 MiB	11	0	2,96 MiB	11	
☐	Malloc 272,00 KiB	2,92 MiB	11	7	4,78 MiB	18	
☐	VM: Image IO	2,74 MiB	1	10	8,80 MiB	11	
☐	VM: ImageIO_PNG_Data	2,45 MiB	6	60	130,39 MiB	66	
☐	Malloc 64,00 KiB	2,25 MiB	36	14	3,12 MiB	50	
☐	Malloc 1,02 MiB	2,05 MiB	2	0	2,05 MiB	2	
☐	Malloc 2,00 MiB	2,00 MiB	1	3	8,00 MiB	4	
☐	Malloc 1,94 MiB	1,94 MiB	1	0	1,94 MiB	1	
☐	Malloc 928,00 KiB	1,81 MiB	2	0	1,81 MiB	2	
☐	Malloc 248,00 KiB	1,70 MiB	7	0	1,70 MiB	7	
☐	Malloc 860,00 KiB	1,68 MiB	2	0	1,68 MiB	2	
☐	Malloc 1,53 MiB	1,53 MiB	1	0	1,53 MiB	1	
☐	Malloc 736,00 KiB	1,44 MiB	2	0	1,44 MiB	2	
☐	Malloc 732,00 KiB	1,43 MiB	2	0	1,43 MiB	2	
☐	Malloc 32,00 KiB	1,41 MiB	45	198	7,59 MiB	243	
☐	Malloc 700,00 KiB	1,37 MiB	2	0	1,37 MiB	2	
☐	Malloc 1,35 MiB	1,35 MiB	1	0	1,35 MiB	1	
☐	Malloc 620,00 KiB	1,21 MiB	2	0	1,21 MiB	2	
☐	C3DNode	1,17 MiB	4 037	487	1,31 MiB	4 524	
☐	VM: CG backing stores	1,13 MiB	2	2	2,27 MiB	4	
☐	Malloc 580,00 KiB	1,13 MiB	2	2	2,27 MiB	4	
☐	Malloc 576,00 KiB	1,12 MiB	2	0	1,12 MiB	2	
☐	Malloc 64 Bytes	1,07 MiB	17 560	49 886	4,12 MiB	67 446	
☐	VM: CG image	1,05 MiB	7	74	136,34 MiB	81	
☐	SCNNode	1 011,50 KiB	4 046	487	1,11 MiB	4 533	

Кейс: лаги в игре

iOS. Проанализировать работу CPU

Estimate: **1**

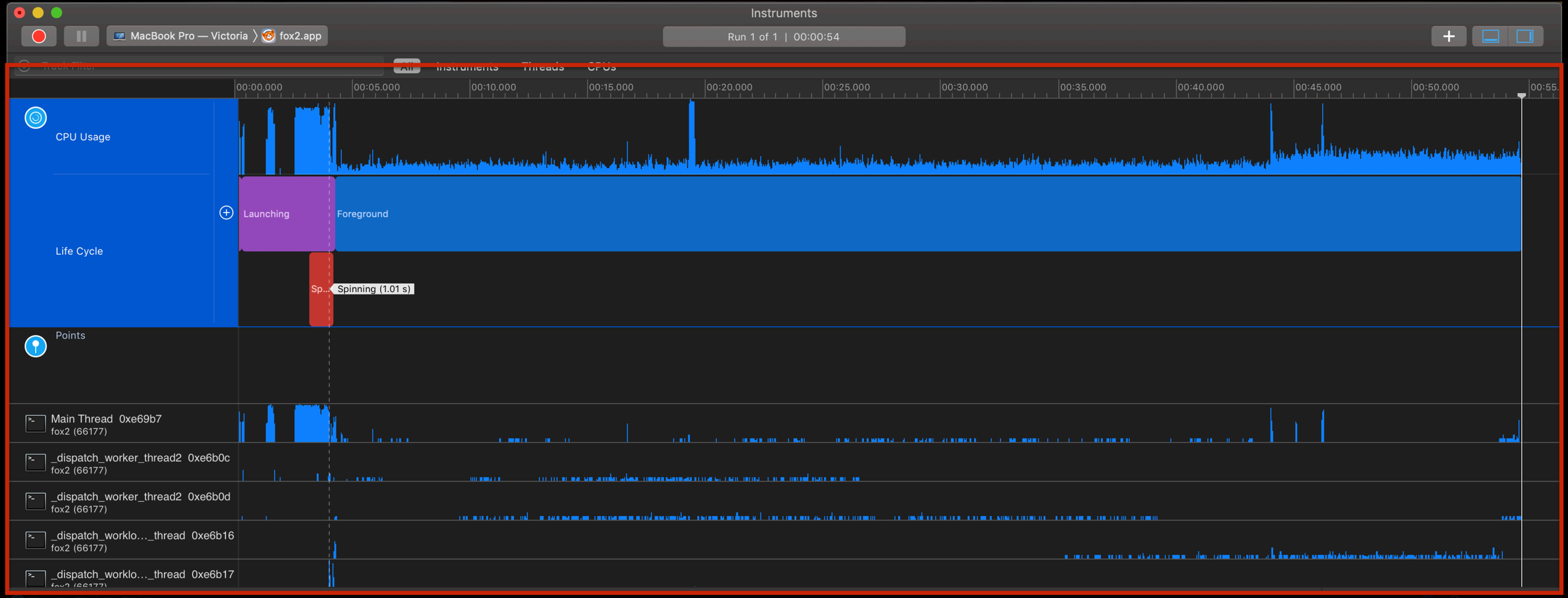
Priority: **Minor**

⌘+i

Time Profiler

Отображает call tree, которое показывает количество времени, потраченное на выполнение методов в процессе работы приложения

Видос

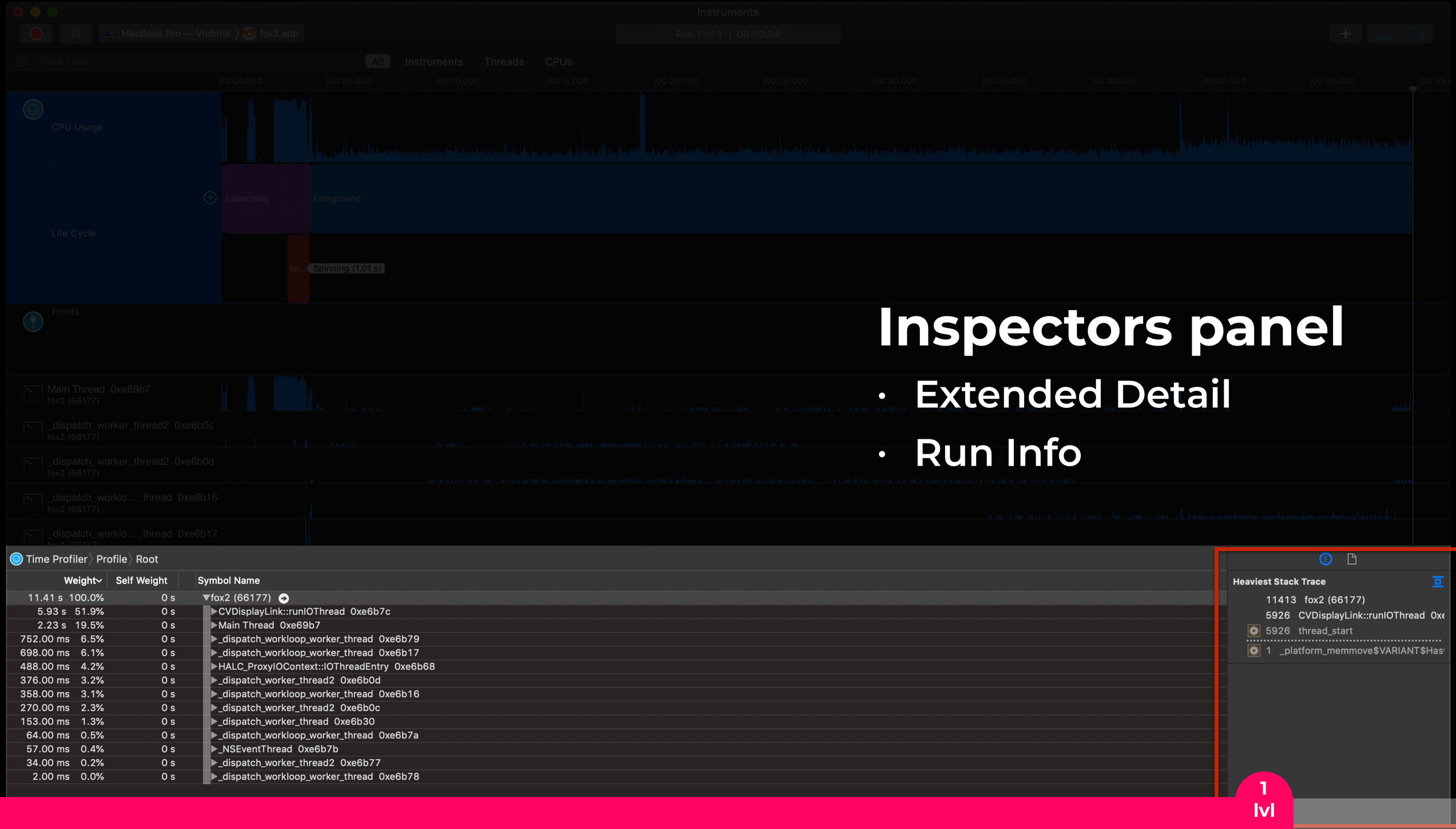


CPU Usage, Life Cycle, threads

Графическое отображение данных, собранных инструментом в разрезе разных аспектов

Detail Panel

Показывает основную информацию
о конкретном инструменте, который
вы используете



Inspectors panel

- Extended Detail
- Run Info

1 lvl

Плюсы первого уровня

- ✓ Предоставляют детализированные данные
- ✓ Видят “шире”, чем ручное тестирование
- ✓ Помогают найти проблему и сказать разработчику, где конкретно она возникла

Минусы первого уровня

- Данные не структурированы, хаотичны
- Много лишнего функционала
- Нужно уметь пользоваться
- Требуется практика

Что если нам этого недостаточно

- Готовых инструментов Apple бывает недостаточно
- Можно сделать инструменты более подробными и удобными

Level 2

Задача:

Структурировать хаотичный поток данных

Узнаете: что такое signpost и как его использовать

Получите: более структурированные данные

Сможете: соотнести технические и бизнес-процессы между собой

Когда применять: хотим локализовать определенное событие

Разметка событий signpost

Тип события

Одномоментное
(нажатие на кнопку, возникновение ошибки и др.)

Интервальное
(сетевой запрос, загрузка документа и др.)

используем

os_signpost с типом **.event**

os_signpost с типом **.begin** и **.end**

Интеграция в проект: Swift

импорт

```
import os.signpost
```

экземпляр

```
let pointsOfInterestLog = OSLog(subsystem: “com.example.your-app”, category: .
pointsOfInterest)
let networkLog = OSLog(subsystem: “com.example.your-app”, category:
“NetworkOperations”)
```

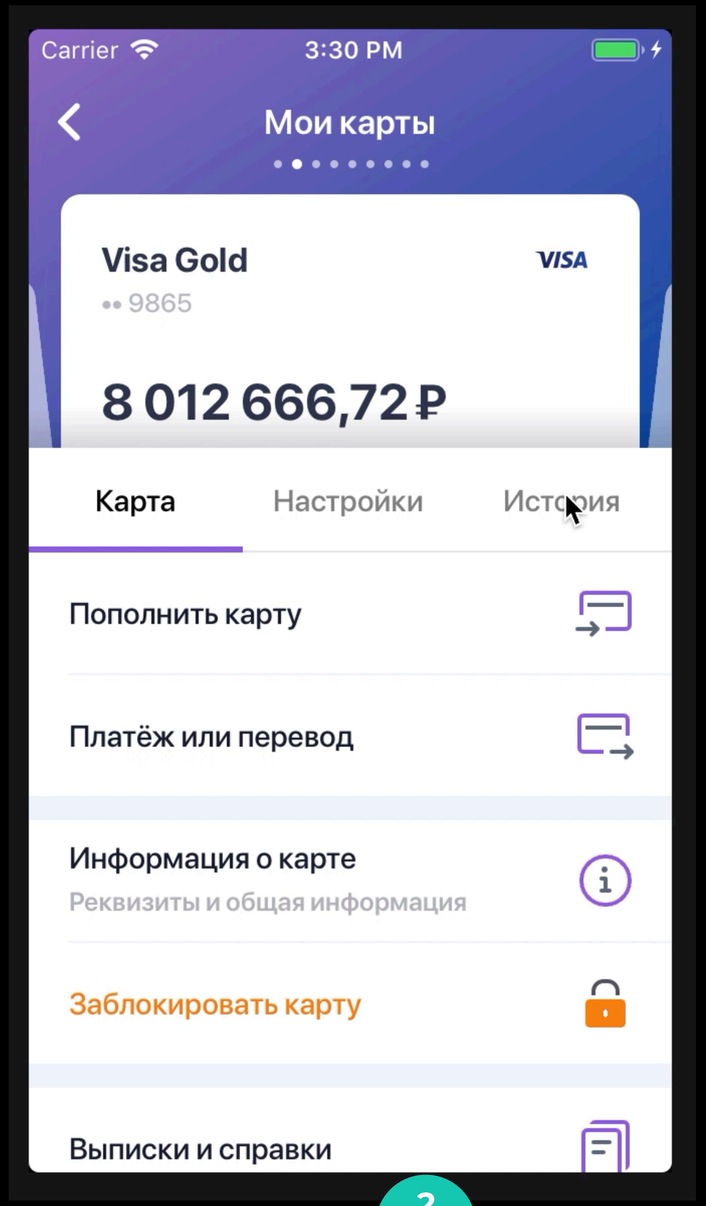
тип события
тип категории

```
os_signpost(.event, log: pointsOfInterestLog, name: “Start work”)
os_signpost(.begin, log: networkLog, name: “Overall work”)
```

тип события

```
for element in elements {
    os_signpost(.begin, log: networkLog, name: “Element work”)
    makeWork(for: element)
    os_signpost(.end, log: networkLog, name: “Element work”)
}

os_sginpost(.end, log: networkLog, name: “Overall work”)
```

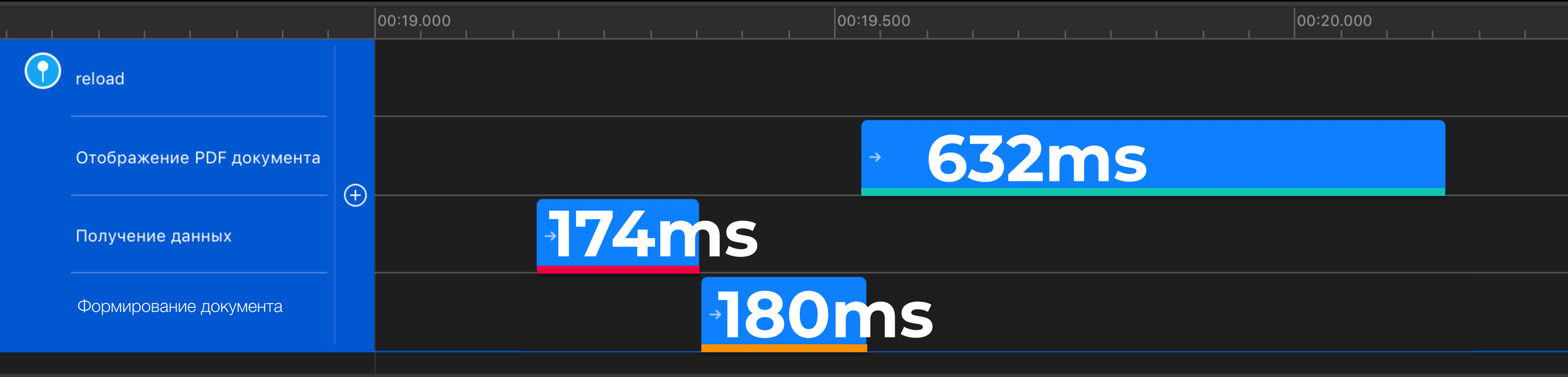
Бизнес-процесс



Технический процесс

1. **Получение данных по сети**
2. **Формирование документа**
3. **Отображение на экране**

Слабое звено:
отображение дольше, чем сетевой запрос



os_signpost 〉 Summary: Intervals

Category / Name / Start Message / End Message	Count	Duration	Min Duration	Avg Duration	Std Dev Du...	Max Durati...
▼ * All *	4	1.79 s	75.89 µs	448.48 ms	353.42 ms	987.07 ms
▼ Statements	3	1.79 s	174.70 ms	597.94 ms	305.83 ms	987.07 ms
▶ Получение данных	1	174.70 ms	174.70 ms	174.70 ms	n/a	174.70 ms
▶ Создание PDF документа	1	987.07 ms	987.07 ms	987.07 ms	341.59 ms	987.07 ms
▶ Отображение PDF документа	1	632.06 ms	632.06 ms	632.06 ms	108.26 ms	632.06 ms

Слабое звено:
отображение дольше, чем сетевой запрос

632ms

WKWebKit

PDFKit

UIWebView

CoreGraphics

iOS 2.0–12.0

Deprecated

Кейс: отображение документа

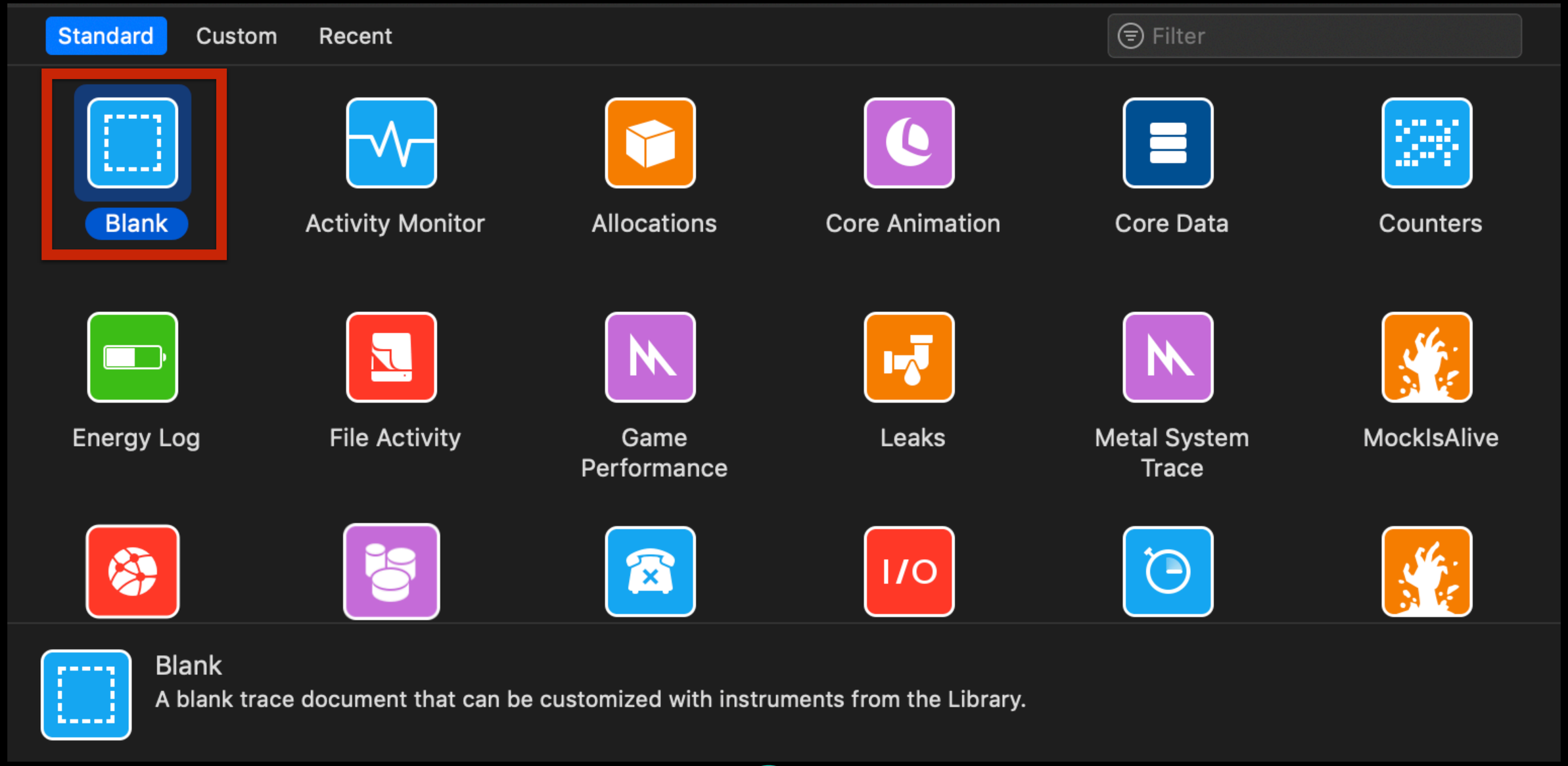
Протестировать механизмы отображения документа (PDFKit vs WKWebView)

Estimate: **5-8**

Priority: **Major**

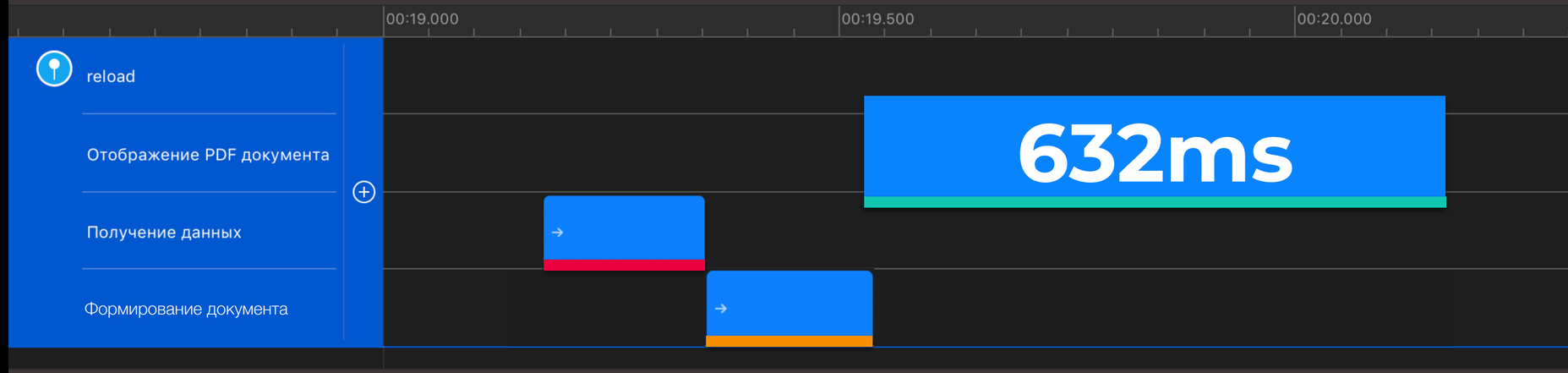
Включаем signpost

⌘+i



Видос

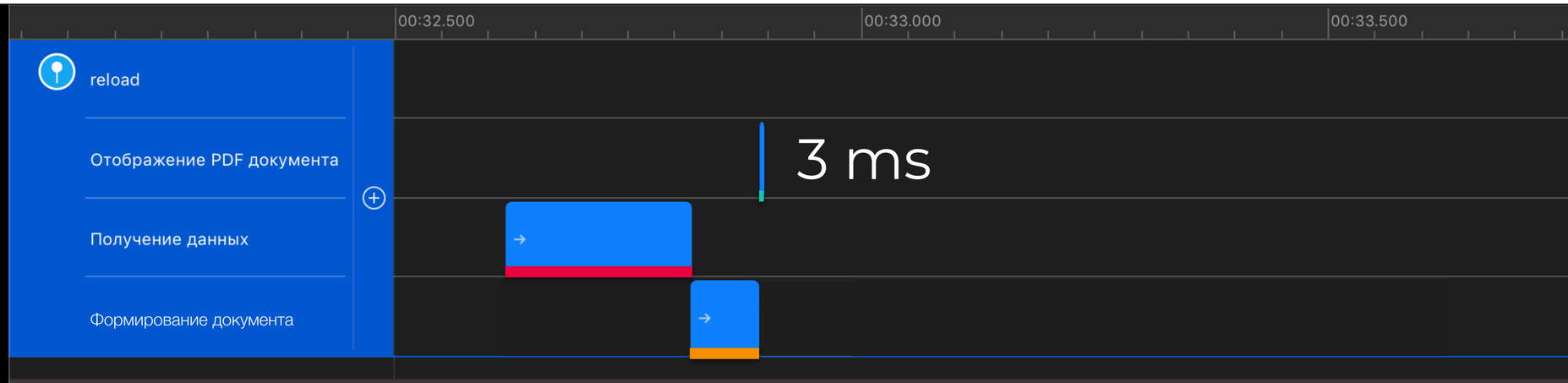
WKWebKit



os_signpost 〉 Summary: Intervals

Category / Name / Start Message / End Message	Count	Duration	Min Duration	Avg Duration	Std Dev Du...	Max Durati...
* All *	4	1.79 s	75.89 µs	448.48 ms	353.42 ms	987.07 ms
Statements	3	1.79 s	174.70 ms	597.94 ms	305.83 ms	987.07 ms
Получение данных	1	174.70 ms	174.70 ms	174.70 ms	n/a	174.70 ms
Создание PDF документа	1	987.07 ms	987.07 ms	987.07 ms	341.59 ms	987.07 ms
Отображение PDF документа	1	632.06 ms	632.06 ms	632.06 ms	108.26 ms	632.06 ms

PDFKit



os_signpost 〉 Summary: Intervals

Category / Name / Start Message / End Message	Count	Duration	Min Duration	Avg Duration	Std Dev Du...	Max Durati...
* All *	4	477.15 ms	208.42 µs	119.29 ms	139.84 ms	277.18 ms
Statements	3	476.94 ms	2.92 ms	158.98 ms	141.00 ms	277.18 ms
Получение данных	1	196.84 ms	196.84 ms	196.84 ms	n/a	196.84 ms
Отображение PDF документа	1	2.92 ms	2.92 ms	2.92 ms	n/a	2.92 ms
	1	277.18 ms	277.18 ms	277.18 ms	n/a	277.18 ms

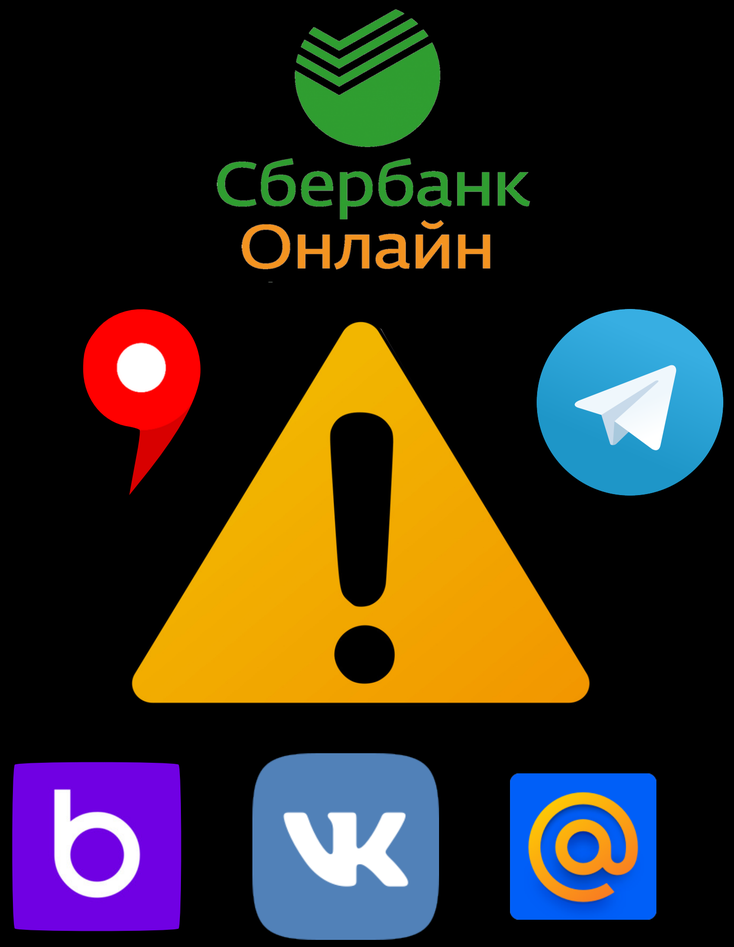
Кейс: формирование PDF-документа

iOS. Проанализировать поведение приложения при memory warning

Estimate: **5-8**

Priority: **Major**

Low Memory Warning



- Если мало памяти, iOS шлёт memory warning в запущенное приложение
- Система принудительно завершает процесс приложения, если память продолжит выделяться
- Для пользователя — crash, для разработчика — не будет информации

Технические детали

У нас есть очередь с операциями,
где **текущая операция = формирование PDF-докумета**

Остановка очереди

продолжает выполнение

Самопроверка статуса

Текущая операция

операция должна проверить себя на отмену и остановиться

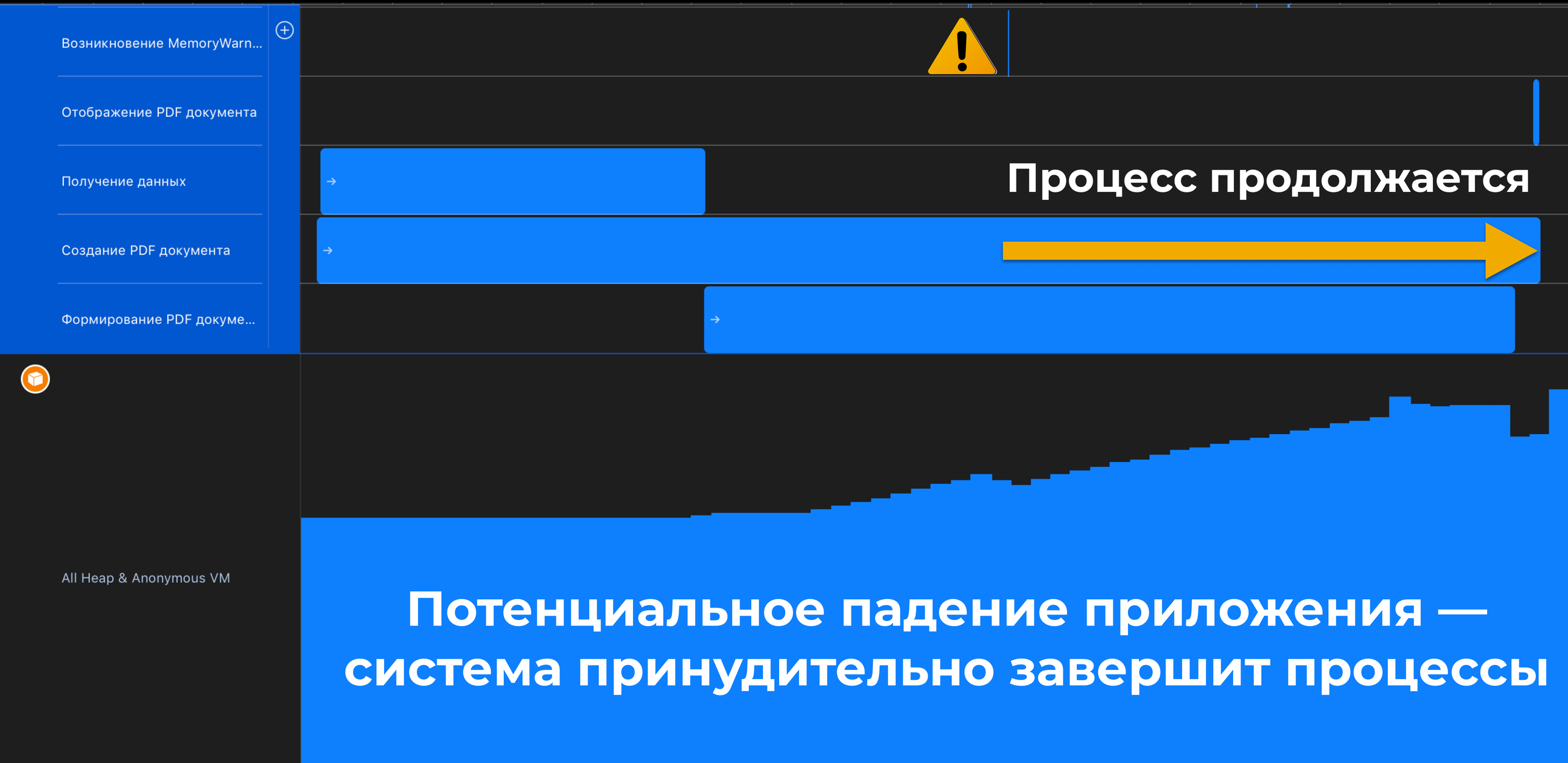
если не проверять — исполнение продолжится

?

Cancelled

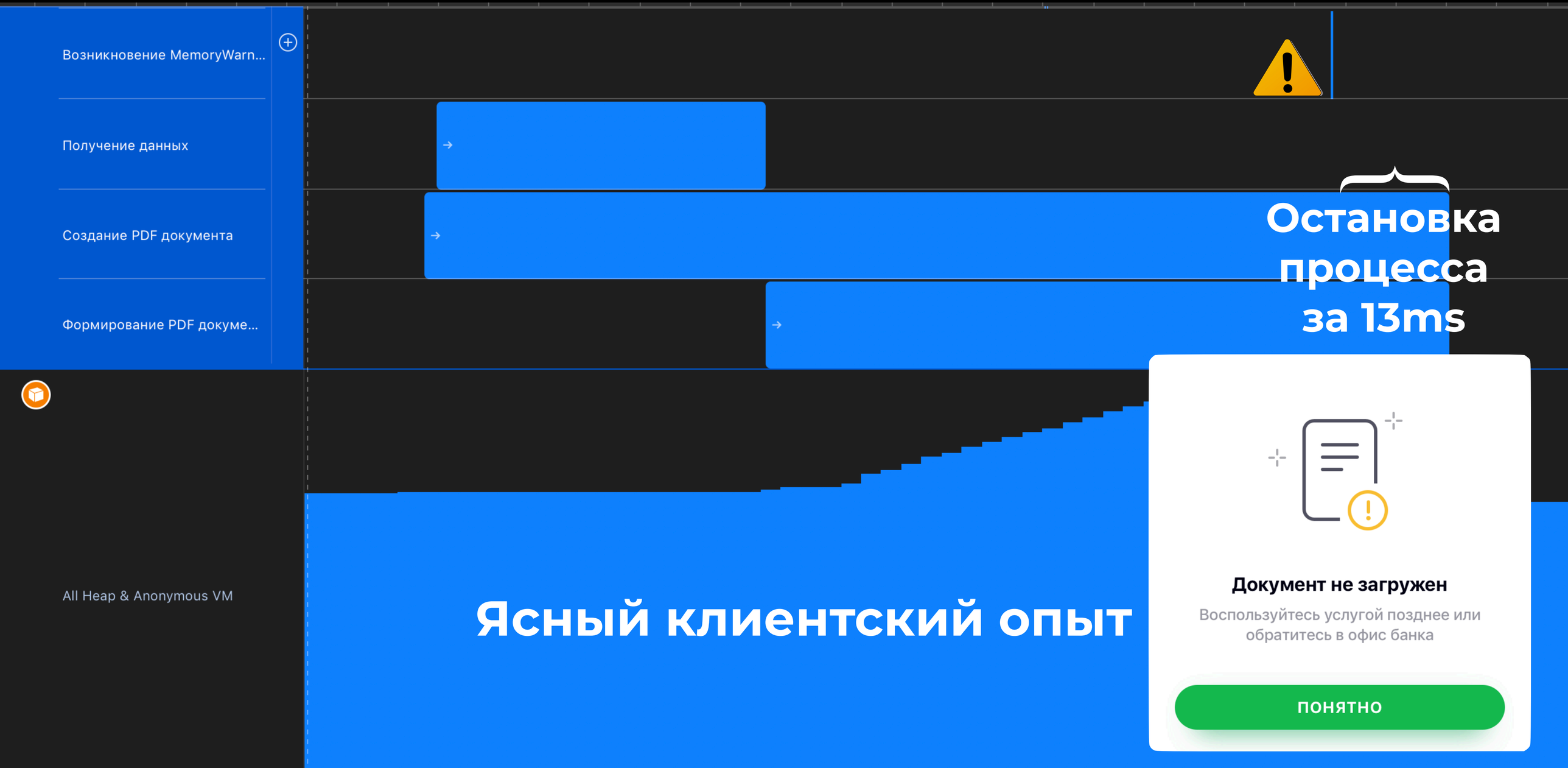
Not Cancelled

Игнорируем Low Memory Warning



Потенциальное падение приложения — система принудительно завершит процессы

Реагируем на Low Memory Warning



Частота проверки статуса

Текущая операция

как часто надо
проверять на статус
Cancelled?

Каждый грамм

Каждый килограмм

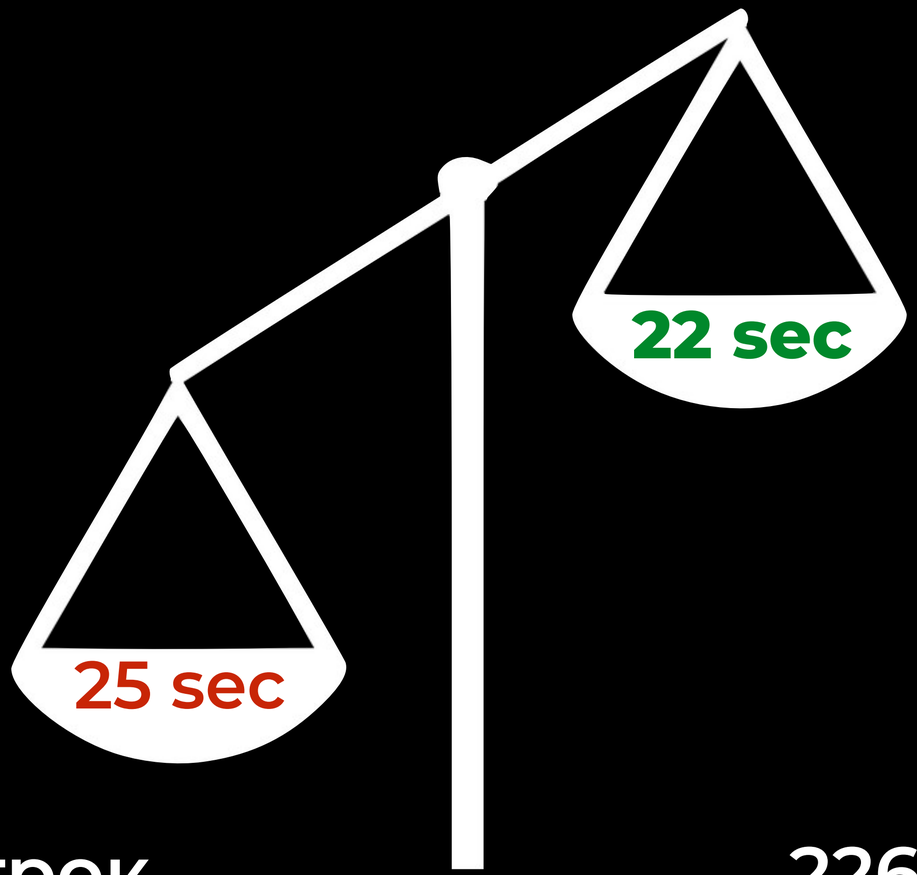
Каждый грамм

Построковая проверка статуса

Каждый килограмм

Постраничная проверка статуса

Время формирования PDF с проверкой статуса



10000 строк

Построковая проверка статуса

226 страниц

Постраничная проверка статуса

Плюсы второго уровня

- ✓ signpost дополняет существующие решения
- ✓ С ним проще интерпретировать получаемые данные
- ✓ Самый простой вариант кастомизации инструментов

Минусы второго уровня

- Не самое удобное использование

- Ручная разметка данных

- Потенциальные ошибки работы, как следствие человеческого фактора

Что если нам этого недостаточно

- Нужно приложить много усилий чтобы завести signpost
- Хотим чтобы было по двойному клику и в космос 🚀


Level
3

Задача:

Сделать свой инструмент

Узнаете: как это можно сделать и когда это нужно делать

Когда применять: хотим автоматизировать логику - анализируем не сами, а за счет инструмента

Показываем: на примере unit-тестов проекта и сторонней библиотеки

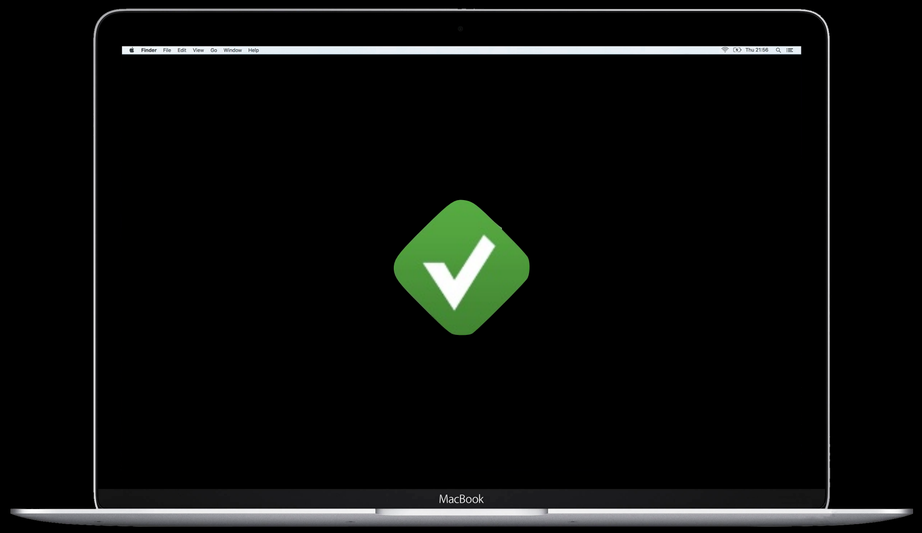
Про unit-тесты в Сбербанк Онлайн

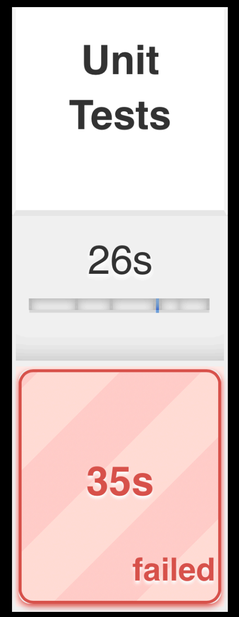
41 000 unit-тестов
в проекте

на 70 команд
разработки iOS

Когда у вас большая команда
(у нас ~160 человек)
тесты очень важны

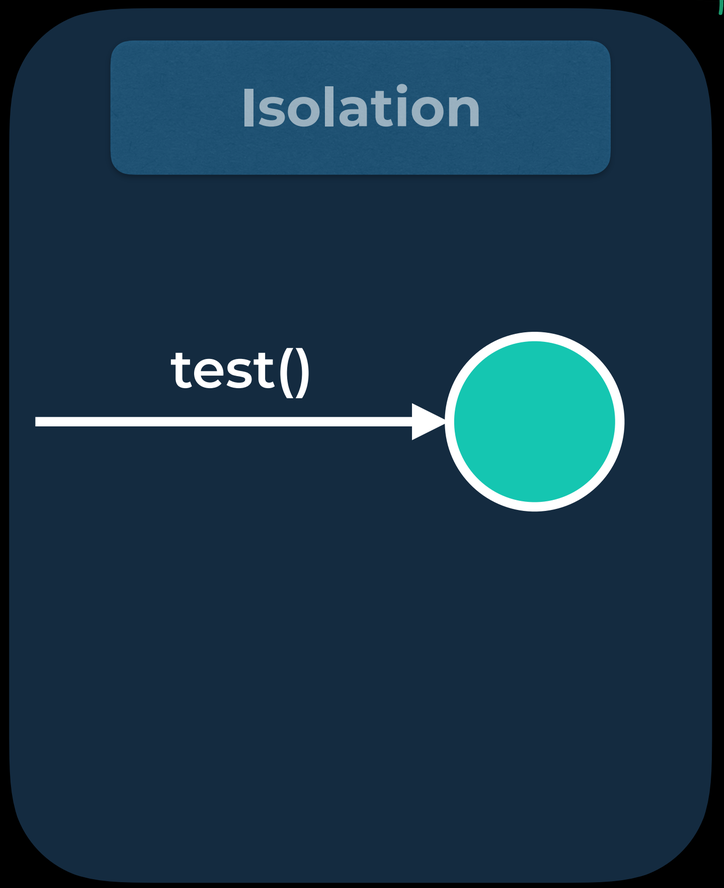
Проблема на CI





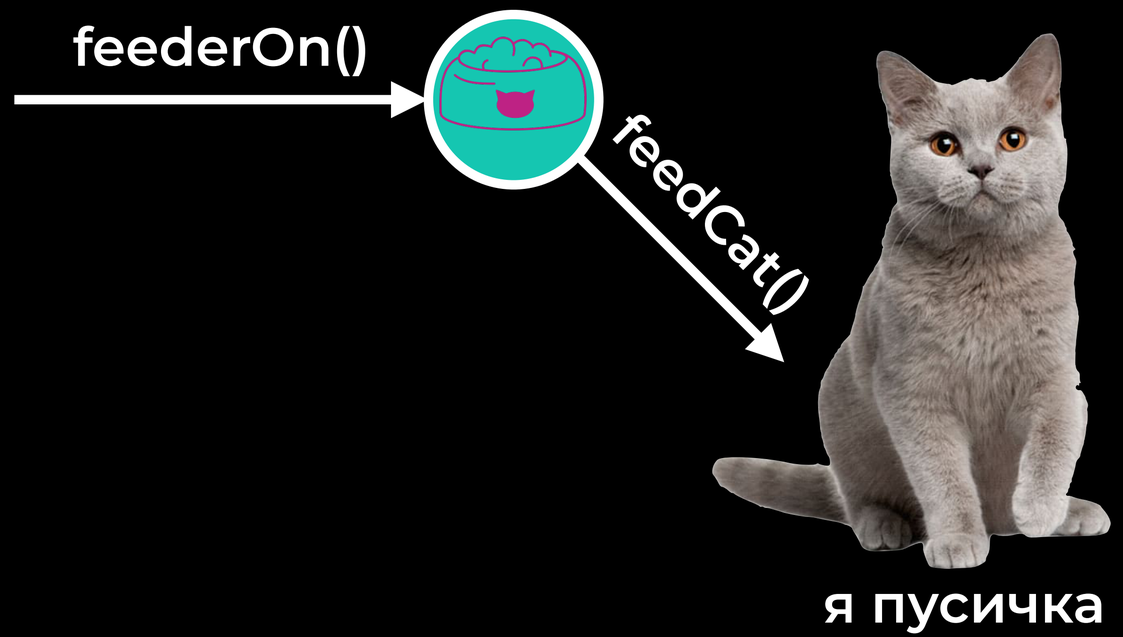
Что такое unit-test?

Процесс, позволяющий проверить корректность исходного кода за счёт процедур использования и обработки



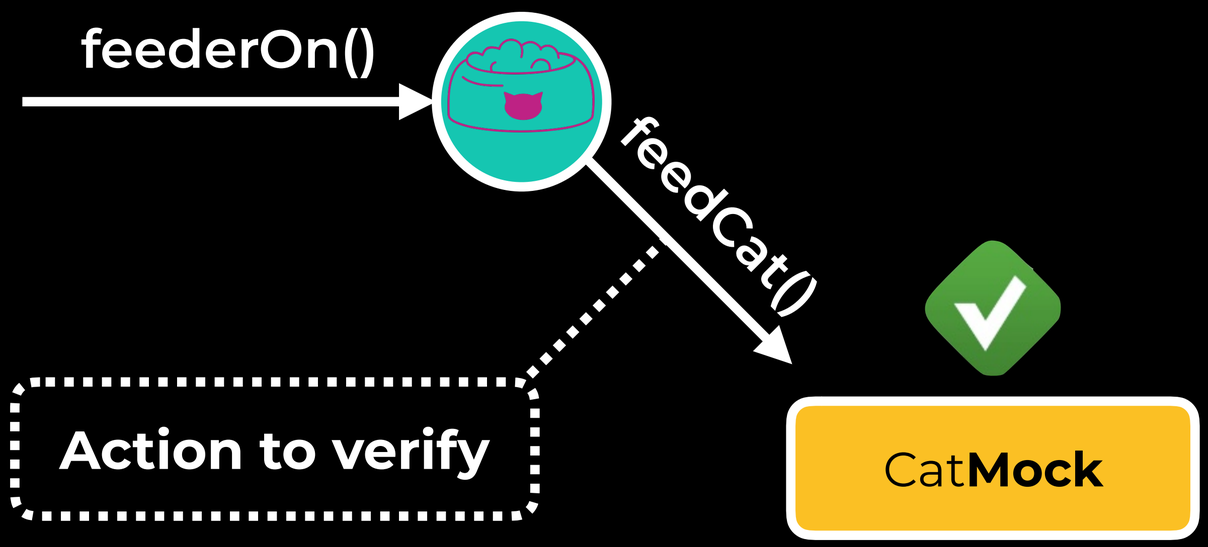
Что такое Mock?

Моки «подсовываются» тестируемому объекту, чтобы можно было проверить, что тестируемый объект выполнил требуемые действия



Что такое Mock?

Моки «подсовываются» тестируемому объекту, чтобы можно было проверить, что тестируемый объект выполнил требуемые действия



OCMock

Mock objects for Objective-C

Stubs – return values for specific method invocations

Dynamic Mocks – verify interaction patterns

Partial Mocks – overwrite methods of existing objects

Кейс: патчим стороннюю либу

iOS. Пофиксить OCMock и memory leaks

Estimate: **>13**

Priority: **Critical**

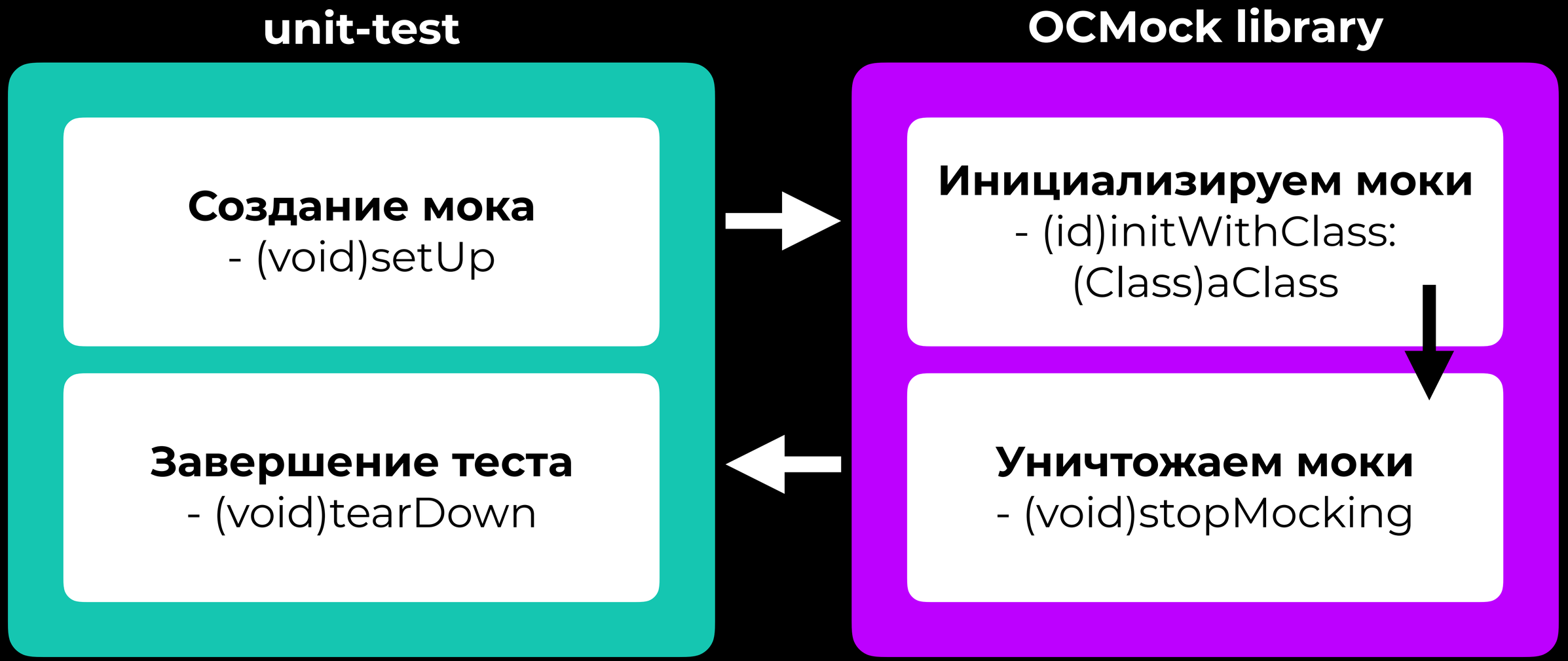
```
@implementation ModuleTests

- (void)setUp
{
    [super setUp];
    self.testModule = OCMPartialMock([SBOLModule new]);
}

- (void)tearDown
{
    self.testModule = nil;
    [super tearDown];
}

- (void)testDemoMode
{
    //arrange
    SBFParamsStorageSettings *paramsStorageMock = OCMClassMock([SBFParamsStoragaSettings class]);
    OCMStub([self.sberAppMock paramsStoraga]).andReturn(paramsStoragaMock);
    OCMStub(paramsStorageMock.demoMode).andReturn(@(YES));

    OCMReject([self.transitionManagarMock resatState]);
    OCMReject([self.appPermissionsServiceMock sendAnalyticsAftar:10]);

    //act && assert
    [self.testModule moduleAutharizationStateDidChange:YES];
```

Создание мока

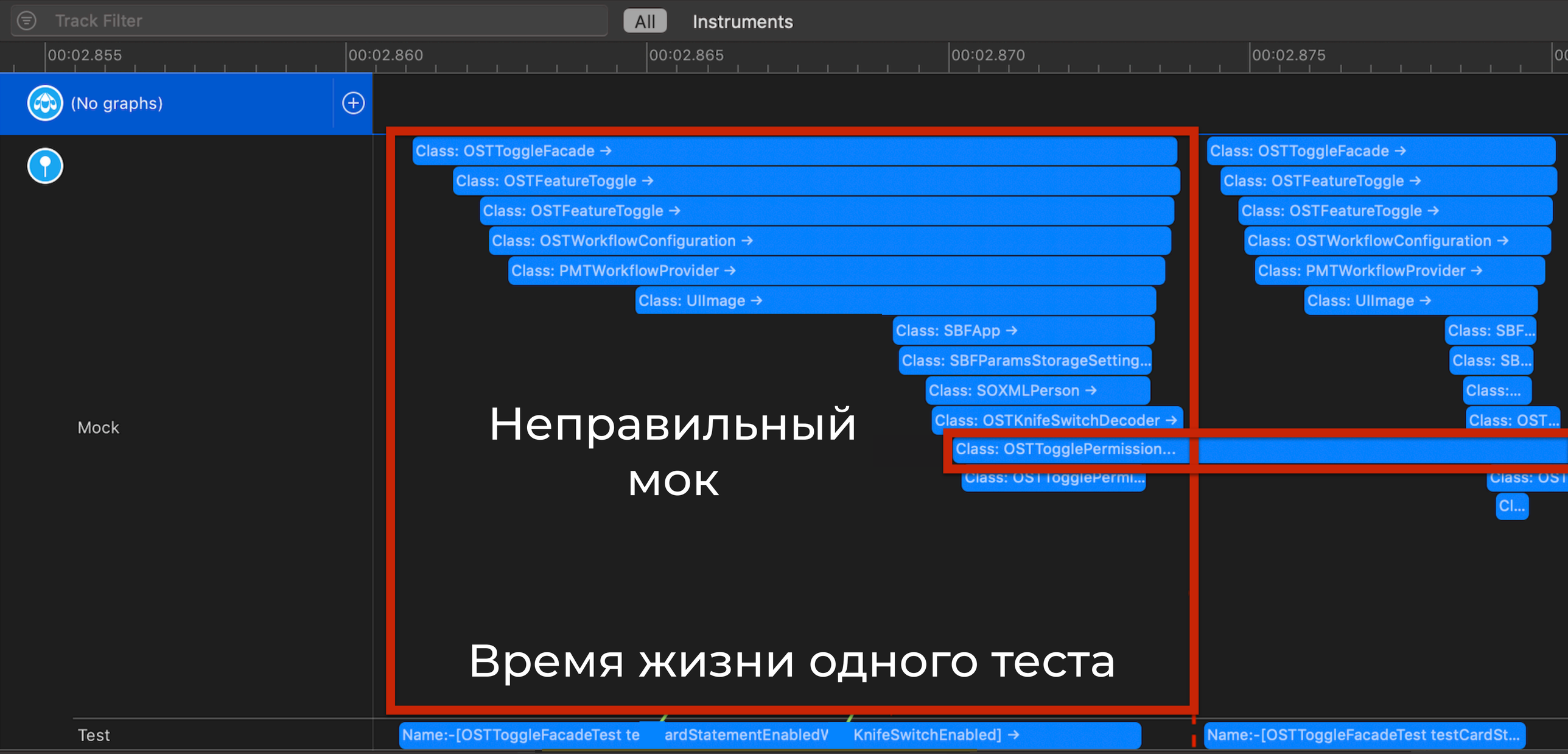
Уничтожение мока

Использование мока

Получаем

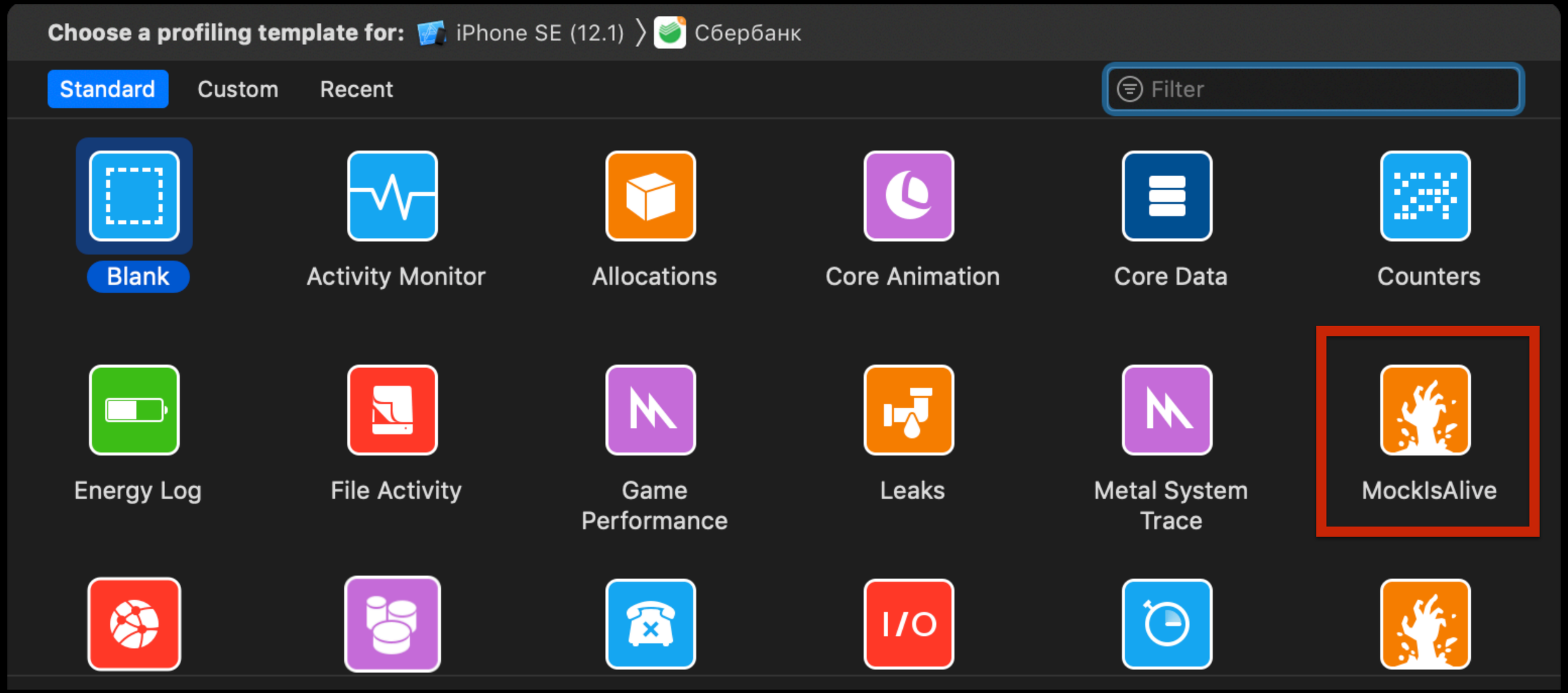


Ищем события, когда уничтожение
происходит позже завершения теста

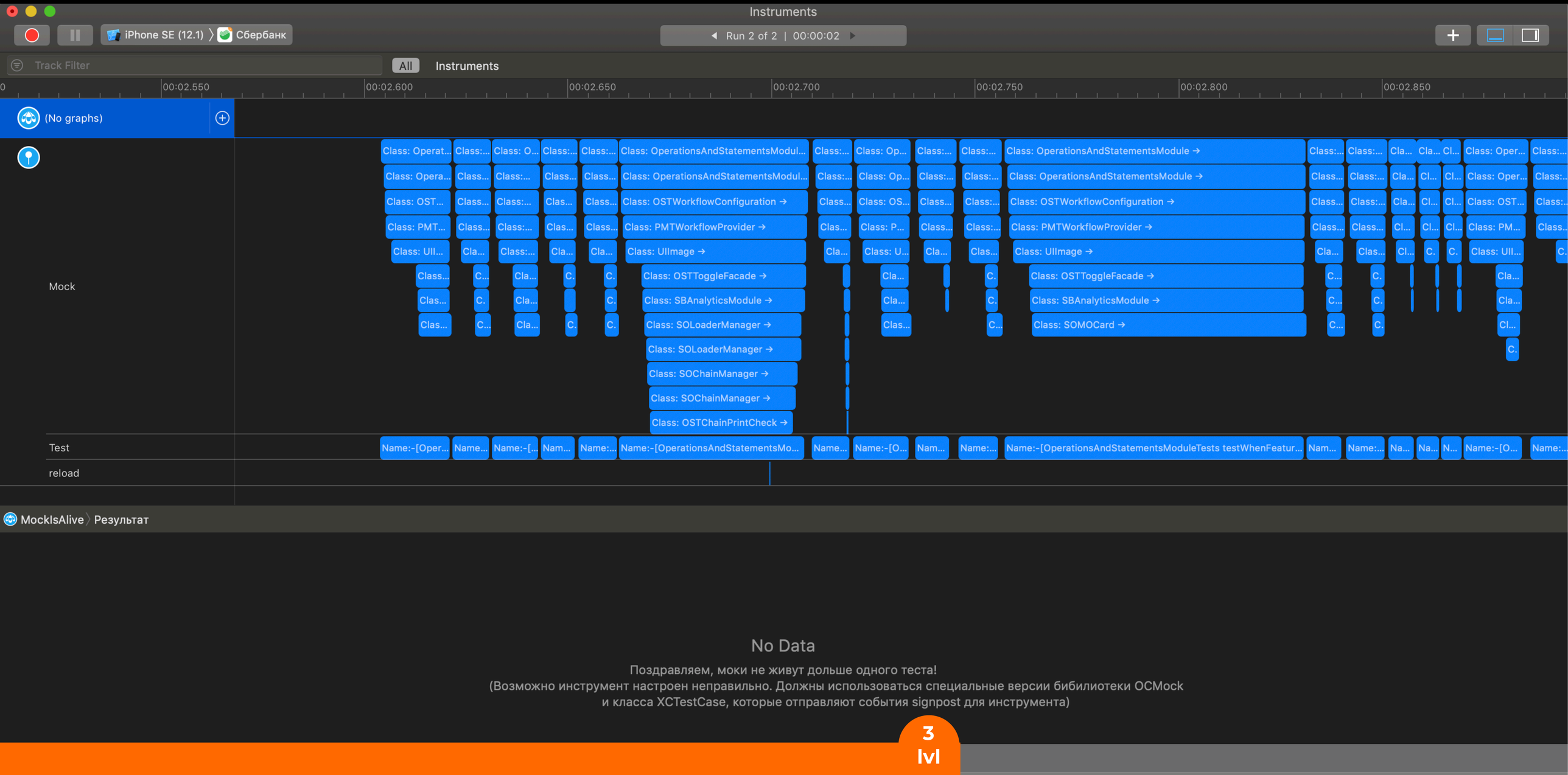

Track Filter
All
Instruments
00:02.855
00:02.860
00:02.865
00:02.870
00:02.875
(No graphs)
Mock
Class: OSTToggleFacade →
Class: OSTFeatureToggle →
Class: OSTFeatureToggle →
Class: OSTWorkflowConfiguration →
Class: PMTWorkflowProvider →
Class: UIImage →
Class: SBFApp →
Class: SBFParamsStorageSetting...
Class: SOXMLPerson →
Class: OSTKnifeSwitchDecoder →
Class: OSTTogglePermission...
Неправильный мок
Время жизни одного теста
Test
Name:-[OSTToggleFacadeTest te ardStatementEnabledV KnifeSwitchEnabled] →
Name:-[OSTToggleFacadeTest testCardSt...
MockIsAlive 〉 Результат
Время
Описание результата
00:02.985.366 Мок 9241915778591403200 с типом OSTTogglePermission был создан в · [OSTToggleFacadeTest testOperationsAndStatementsEnabledWithKnifeSwitchDisabled] и продолжает

Выбираем инструмент

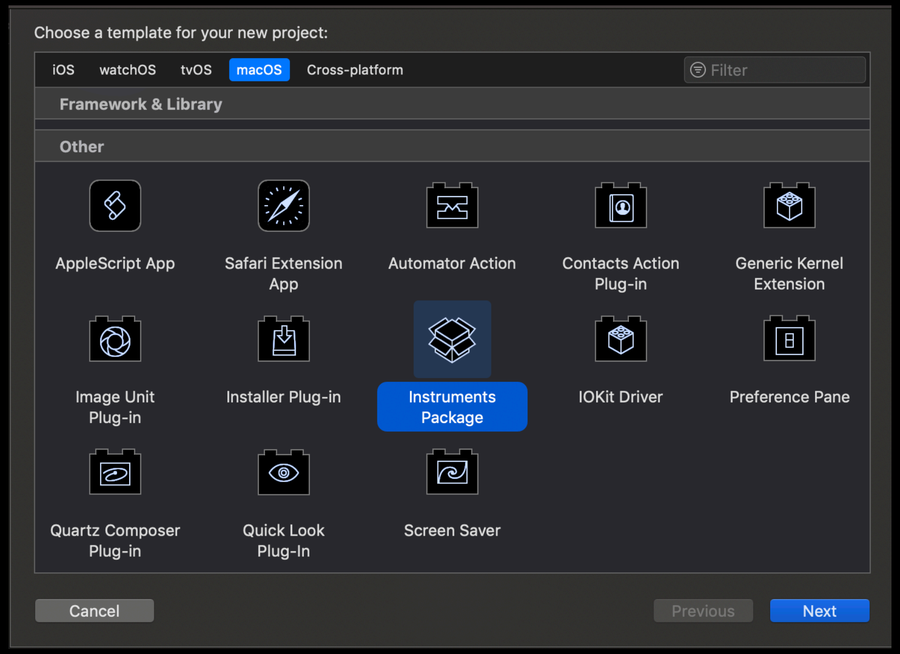
⌘+i



Стало


Instruments
iPhone SE (12.1) 〉 Сбербанк
Run 2 of 2 | 00:00:02
Track Filter
All
Instruments
00:02.550
00:02.600
00:02.650
00:02.700
00:02.750
00:02.800
00:02.850
(No graphs)
Mock
Test
reload
MockIsAlive 〉 Результат
No Data
Поздравляем, моки не живут дольше одного теста!
(Возможно инструмент настроен неправильно. Должны использоваться специальные версии бибилиотеки OCMock
и класса XCTestCase, которые отправляют события signpost для инструмента)

File -> New -> Project -> категория macOS -> Instruments package




```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
    <id>none.HBInstrument</id>
    <title>HBInstrument</title>
    <owner>
        <name>Victoria Kashlina</name>
    </owner>

    <!-- Instruments Developer Help: https://help.apple.com/instruments/developer/mac/current/ -->

    <!-- MARK: Schema Definitions -->
    <!-- Define point and interval schemas needed to represent the input and output tables your package will use.
         -->
```

Компоненты инструмента

Input

Output

UI
представление

Логика
инструмента

C Language Integrated Production System

программная среда для разработки экспертных систем

Правило 1:

ЕСЛИ
(выполняются условия 1)
ТОГДА
(выполнить действия 1)

Правило 2:

...

Сущности в CLIPS

```
(deftemplate unitTest
    (slot time (type INTEGER))
    (slot signpost id (type INTEGER))
    (slot testName (type STRING))
)
```

```
(структура unit-теста
    (атрибут: время старта
    (атрибут: идентификатор
    (атрибут: имя теста
)
```

Код инструмента

Роль signpost

```
defrule MODELER :record-test-started
   (os-signpost (subsystem "com.example")
   (category "UnitTestLog")
   (name "Test")
   (event-type "Begin")
      (message$ "Name:"
?testName)
(time ?time)
(identifier ?identifier))
    =>
    (assert (unitTest
(time ?time)
(signpost-id ?identifier)
(testName ?testName)))
)
```

```
задаём правило
   (когда ловим signpost)
   (с категорией)
   (именем "Test")
   (типом-ивента "Begin")
      (достаём данные:
?имя Теста)
(?время старта теста)
(?идентификатор))
    => то
    (записываем их (в unitTest
(время ?time)
(идентификатор ?identifier)
(имя теста ?testName)))
)
```

Правило инструмента: “ЕСЛИ”

```
defrule RECORDER::record-mock-started-before-unit-test-started
```

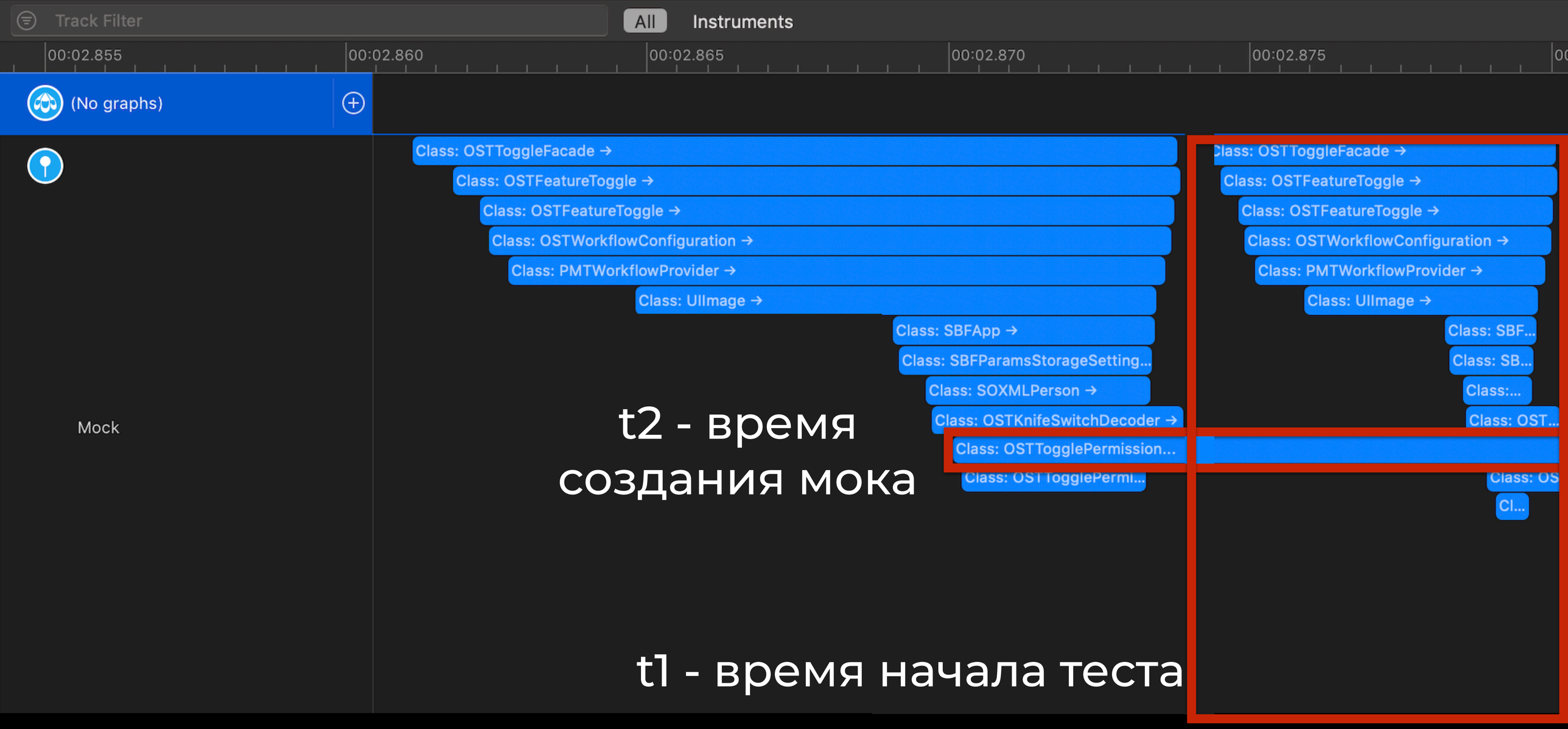
1

```
(unitTest (time ?t1) (testName ?leakedTestName))

(mock (time ?t2) (signpost-id ?signpost-id) (mockedClass ?
    mockedClass) (createdTestName ?createdTestName
```

2

```
(test (> ?t1 ?t2))
```

Если существует unitTest и mock
И, **если** при этом начало теста наступает позже существующего мока



На момент завершения теста есть “живой” мок-объект

Таблица событий

3

(table (table-id ?output) (side append))

(table-attribute (table-id ?output) (has schema detected-mocks-narrative))

Если существует таблица для хранения результатов
со схемой detected-mocks-narrative

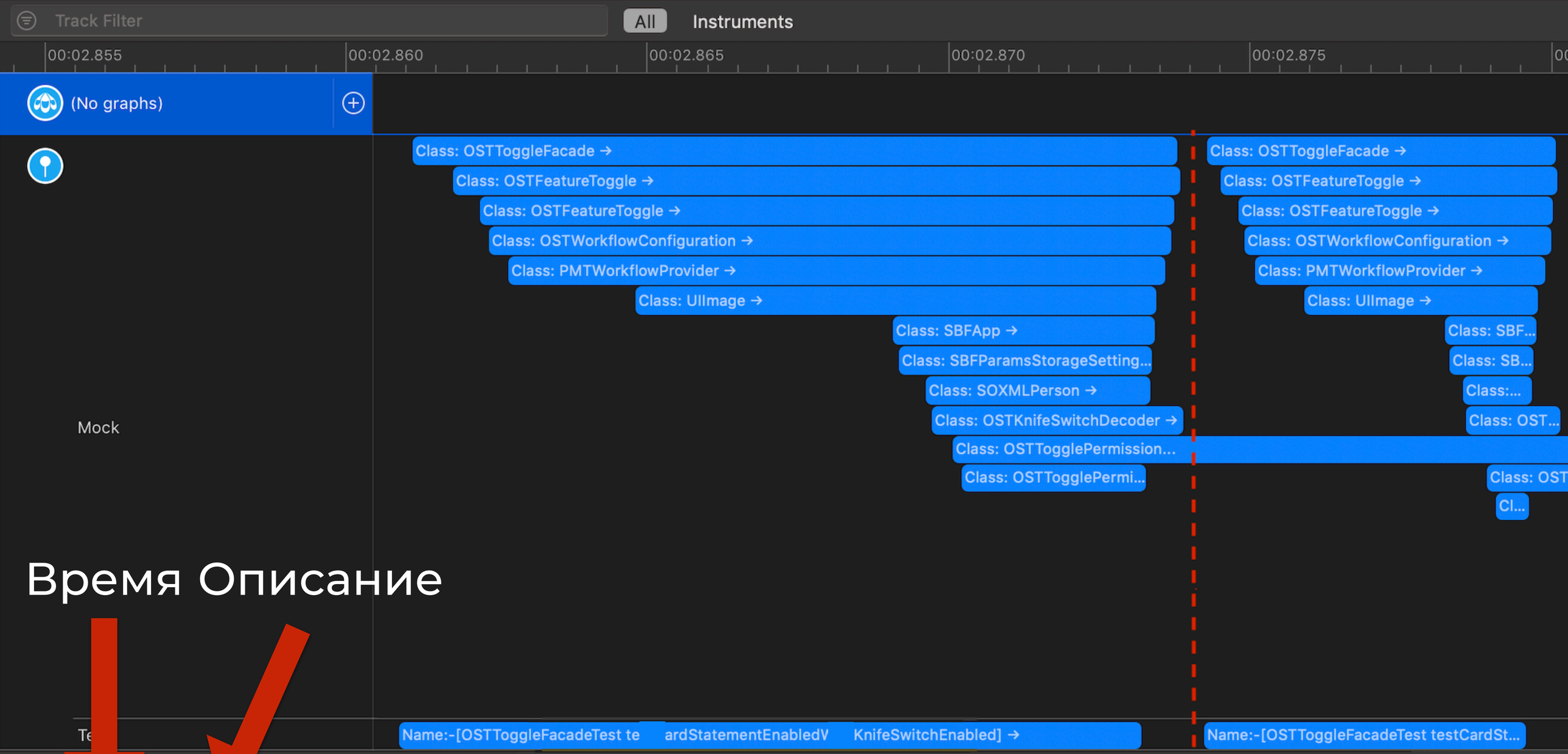
Правило инструмента: “TO”

4 (create-new-row ?output)

5 (set-column time ?t1)

6 (set-column-narrative description "Мок %string% с типом %string% был создан в %string% и продолжает работать в %string%. Необходимо устранить причину удержания мока между тестами, либо явно вызвать stopMocking" ?signpost-id ?mockedClass ?createdTestName ?leakedTestName)

To создаём новую запись в таблице
To заполняем новую запись временем
To дополняем новую запись описанием



Время Описание

Плюсы третьего уровня

- ✓ Решение, которое работает из коробки
- ✓ Легко масштабировать на большую команду

Минусы третьего уровня

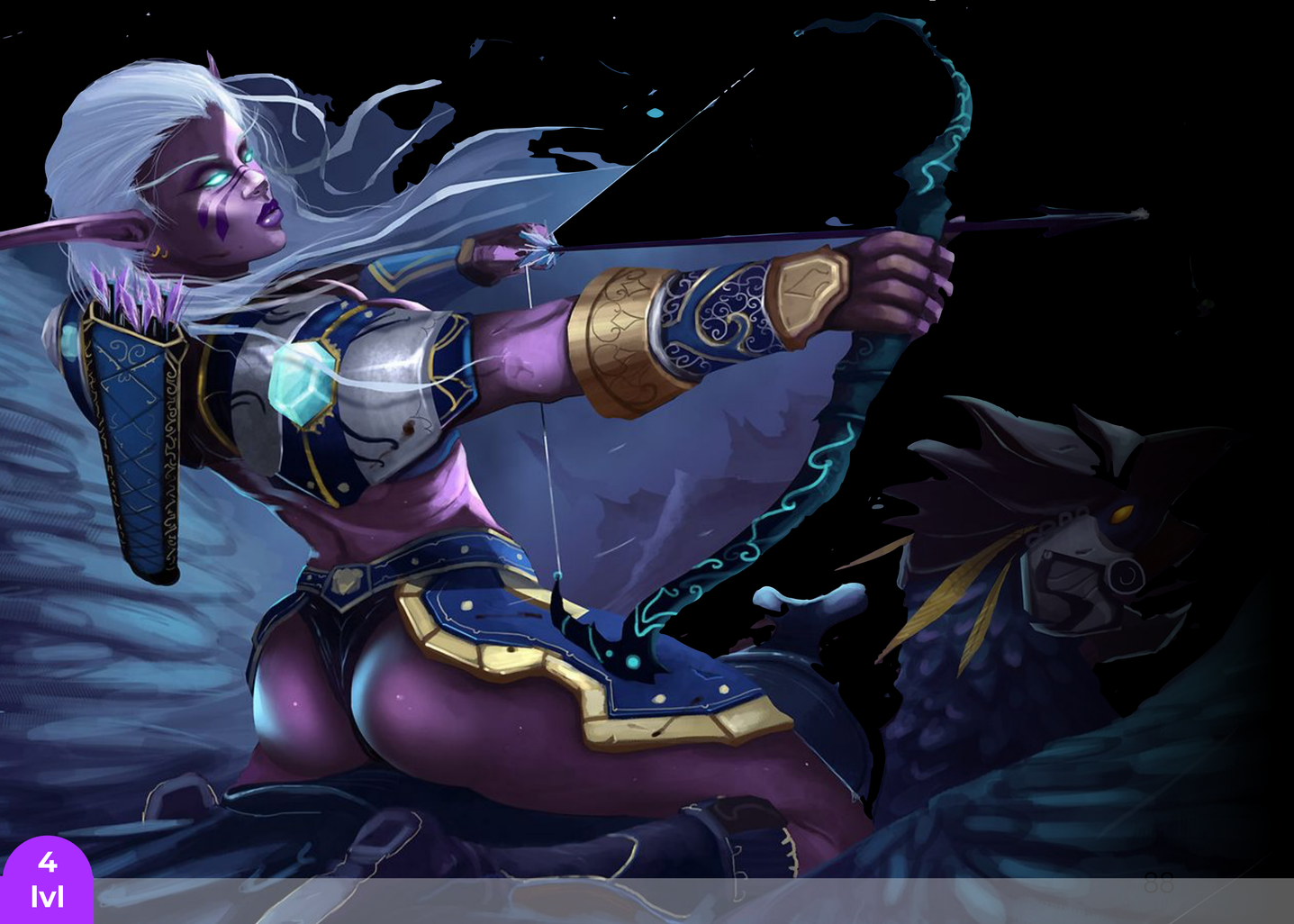
- Трудозатратное решение
- Требует поддержки по мере изменения проекта
- Не факт, что инструмент использует каждый член команды

Но что, если...
нам хочется еще больше

Level 4

Задача:

Понять, как всё автоматизировать



Узнаете: как это всё можно автоматизировать на CI

Когда применять: хотим регулярно проводить проверки без дополнительных усилий

4 lvl

Кейс: автоматизация инструмента

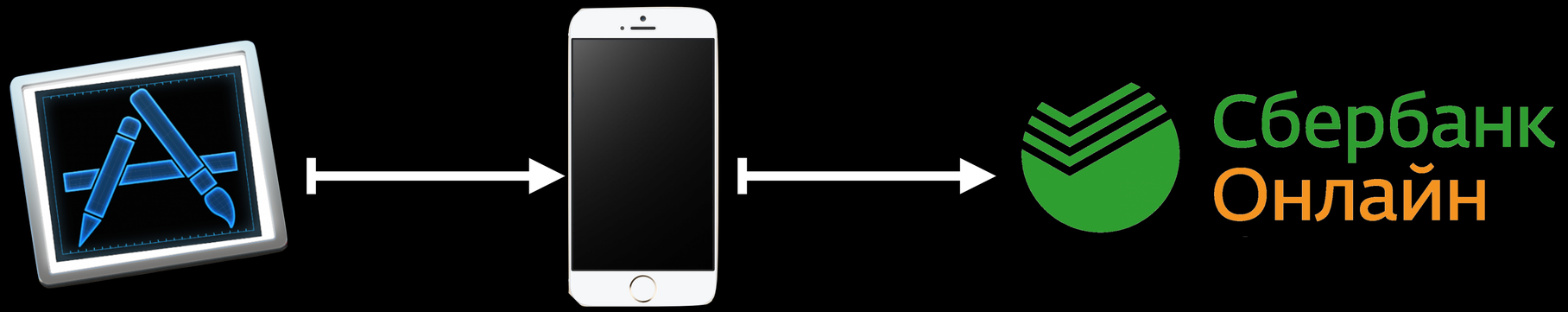
iOS. Встроить инструмент
в прогоны на CI

Estimate: **15**

Priority: **Minor**

Запуск консоли

```
instruments -t Leaks -w 'iPhone 7' SberbankApp.app
```



Запуск консоли

instruments -t Leaks -w 'iPhone 7' SberbankApp.app

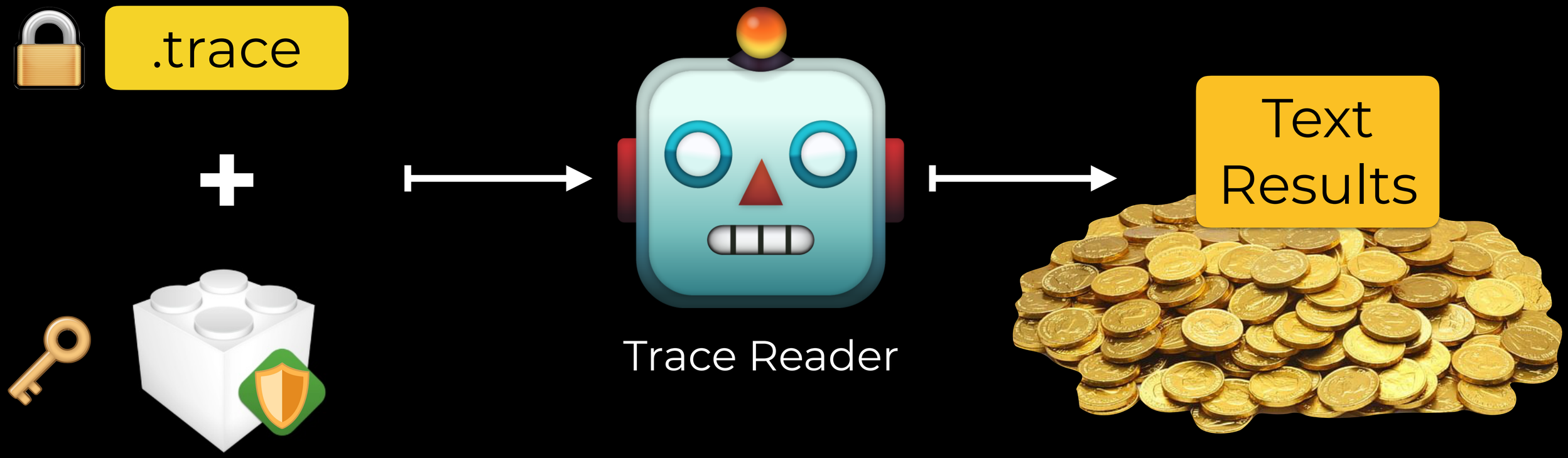
Сбербанк
Онлайн

XCTest

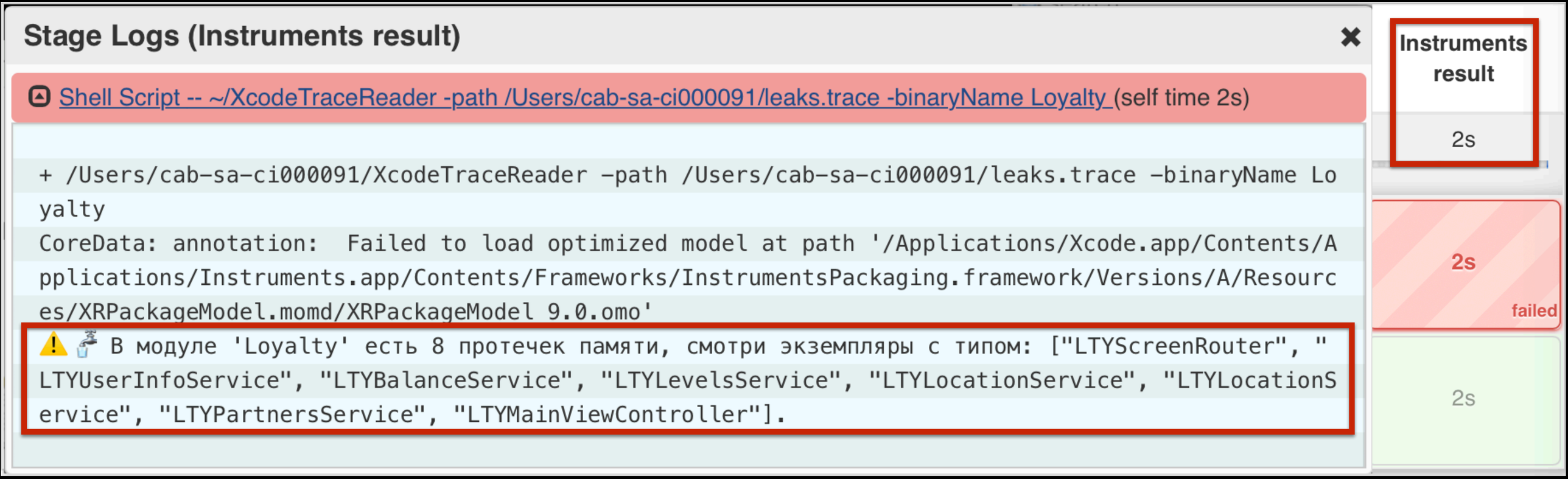
Trace
Document

NSKeyedArchive

Интерпретация результатов



Отображение результатов



Плюсы четвёртого уровня

- ✓ Будет автоматизировано и с периодическими прогонами
- ✓ Сделали один раз – получили долгосрочный профит на многих pull-request’ax
- ✓ Уменьшение ошибок в коде
- ✓ Выявление багов на ранних стадиях разработки

Минусы четвёртого уровня

- Нужны навыки DevOps’а
- Риски работы с приватными фреймворками
- Требуется много сил на последующую поддержку

Выводы

1. Ручного прогона может не хватать
2. Есть дополнительные нюансы, которые нельзя учесть глазами
3. Много пользователей — больше уникальных кейсов, которые не выявить ручным прогоном
4. Использование инструментов доступно всем
5. Есть возможность автоматизировать проверку

Немного ссылок

1. https://help.apple.com/instruments/developer/mac/current/#/devcd5016d31
(Instruments Developer Help)

2. https://habr.com/ru/company/sberbank/blog/443414/
(Signpost: когда брейкпоинтов недостаточно, Антон Власов, Кашлина Виктория)

2. https://developer.apple.com/videos/play/wwdc2018/405/
(WWDC, Measuring Performance Using Logging)

2. https://www.swiftbysundell.com/daily-wwdc/getting-started-with-signposts
(Getting started with signposts, John Sundell)

3. https://developer.apple.com/videos/play/wwdc2018/410/
(WWDC, Creating Custom Instruments)

3. https://habr.com/ru/company/sberbank/blog/447594/
(Custom instruments: когда signpost недостаточно, Антон Власов, Кашлина Виктория)

4. http://jamie-wong.com/post/reverse-engineering-instruments-file-format/
(Reverse Engineering Instruments' File Format, Jamie Wong)

EXTRA LEVEL

Спасибо
за внимание!

@kindredqb
@VlasovAnt