От разовых оптимизаций к системной работе с производительностью

Опыт Uzum Market

Тимур Боргалинов
Android Platform Team

План доклада

01 Связь перформанса и бизнес метрик

02 Какие метрики измеряем и как интерпретируем

03 Как находим проблемы и проверяем результат улучшений

04 Как выстроили постоянный процесс



Тимур Боргалинов

Ранее KazanExpress Innopolis University X5 Group myte

Uzum

Мобильная платформа

Производительность, аналитика
и инструменты для команд

Lead Android Developer

0 → 12 млн
пользователей

1 → 20
Android-разработчиков

IWBL

Co-Founder

Продвижение музыки
через TikTok-креаторов

Full-Stack

Web · Backend

KMM app: Android + iOS

iwannabelistened.com ↗

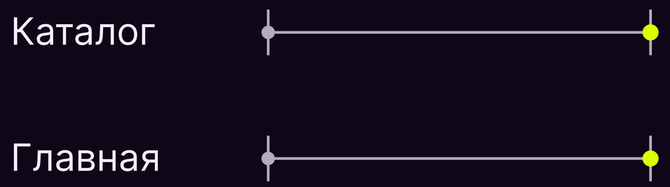
Как сделать перформанс постоянной работой команды

Что ускорять

Куда вложить ресурсы

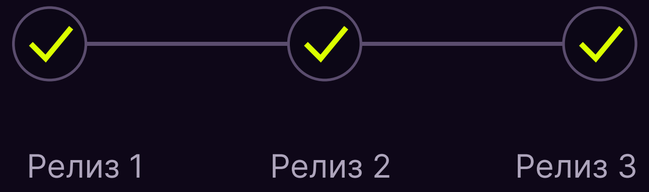
- [x] Проблема пользователя
- [x] **Эффект для бизнеса**

Измерять экраны



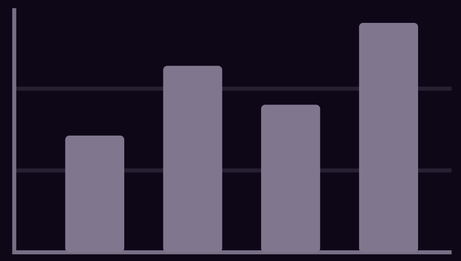
Одни правила измерения

Сохранять результат



Проверка в каждом релизе

Принимать решения на основе данных



Что происходит
у пользователей

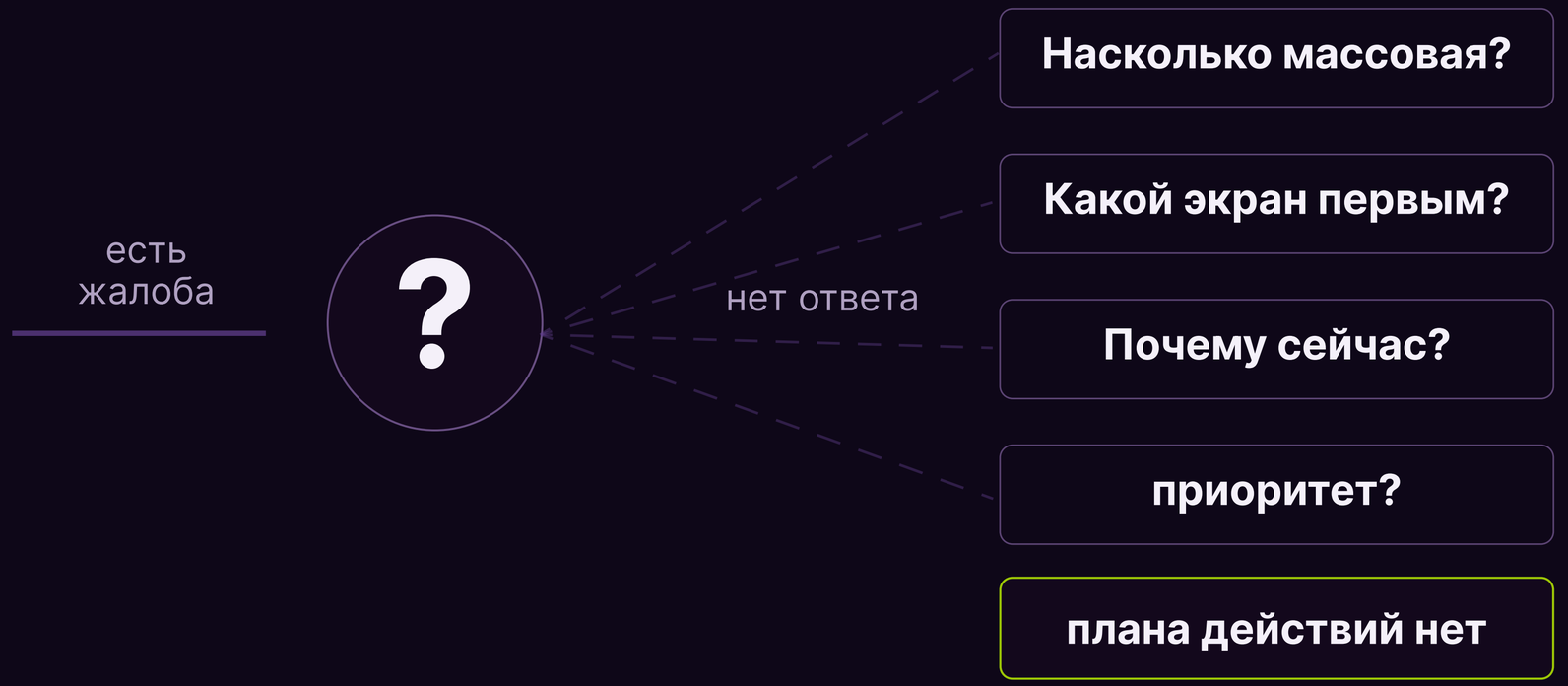


На что тратим
время команды

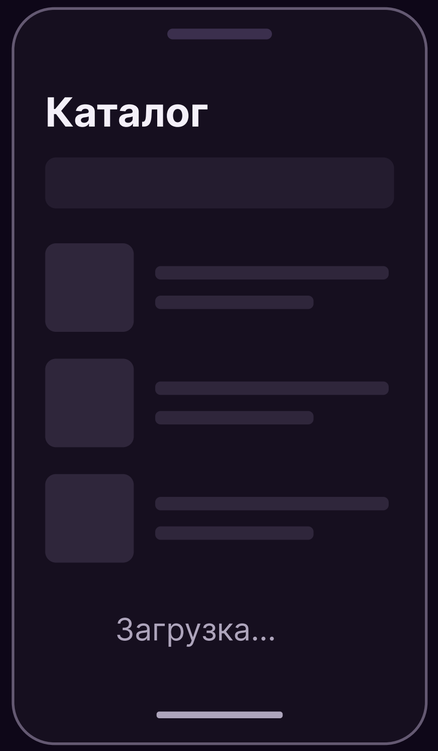
Есть жалобы. Что исправлять первым?

Сигналы

- приложение лагает
- экран долго открывается
- на старом телефоне тормозит
- поддержка приносит жалобы



Медленный экран стоит денег?

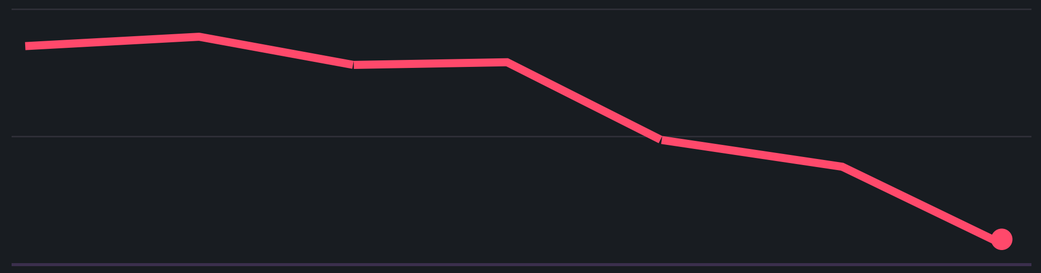


Не дождался

→

В отчёте

Конверсия в покупку



↓ Конверсия снизилась

Не подошёл товар?

Было неудобно покупать?

Не дождался загрузки?

Как время ожидания влияет на следующий шаг пользователя?

Josh: быстрее запуск — больше людей остаются

−30%

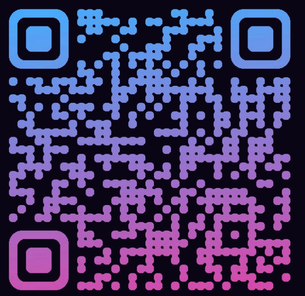
времени запуска

в среднем

>1 млн

пользователей удержали

дополнительно к исходному уровню
по данным команды Josh



Две секунды ожидания меняют поведение

ВРЕМЯ ЗАГРУЗКИ СТРАНИЦЫ

1 с

3 с

+32%

к вероятности ухода

Меньше ждут

Было

Стало

?

Покупают чаще?



Быстрее загрузка — выше конверсия и средний чек

Результаты для мобильных сайтов ритейла

−0,1 с

Время загрузки

+8,4%

Конверсия в покупку

+9,2%

Средний чек

Всё исследование: 37 брендов из ритейла · 30,5 млн сессий

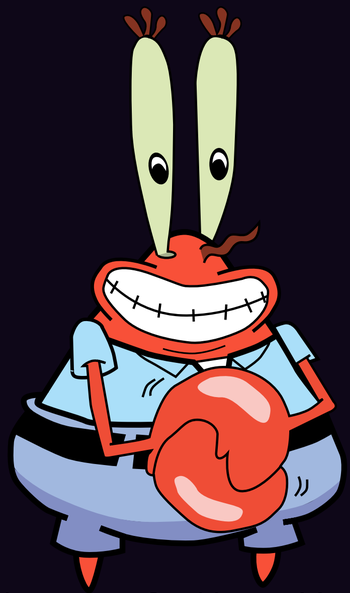
Лёд тронулся

Первые ресурсы на работу с перформансом



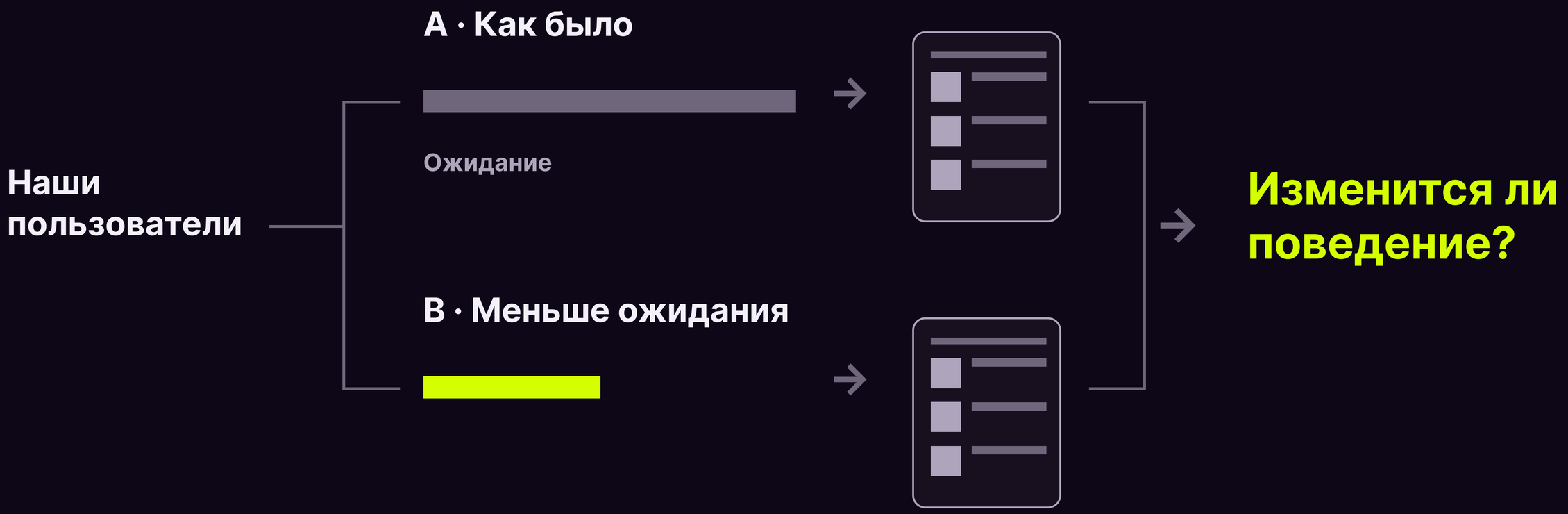
Говорите с бизнесом на его языке

Связывайте ускорение с тем что важно продукту: покупками, конверсией, выручкой.

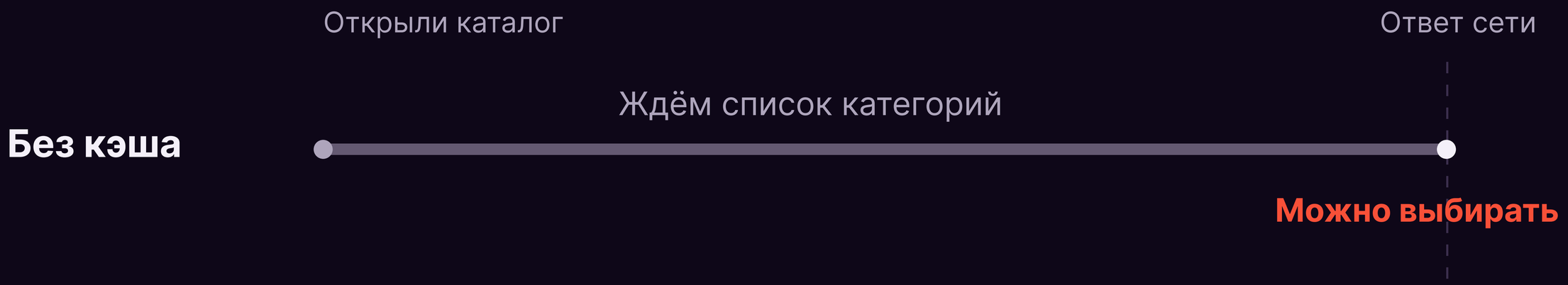


Источник: Seeklogo · Мистер Крабс

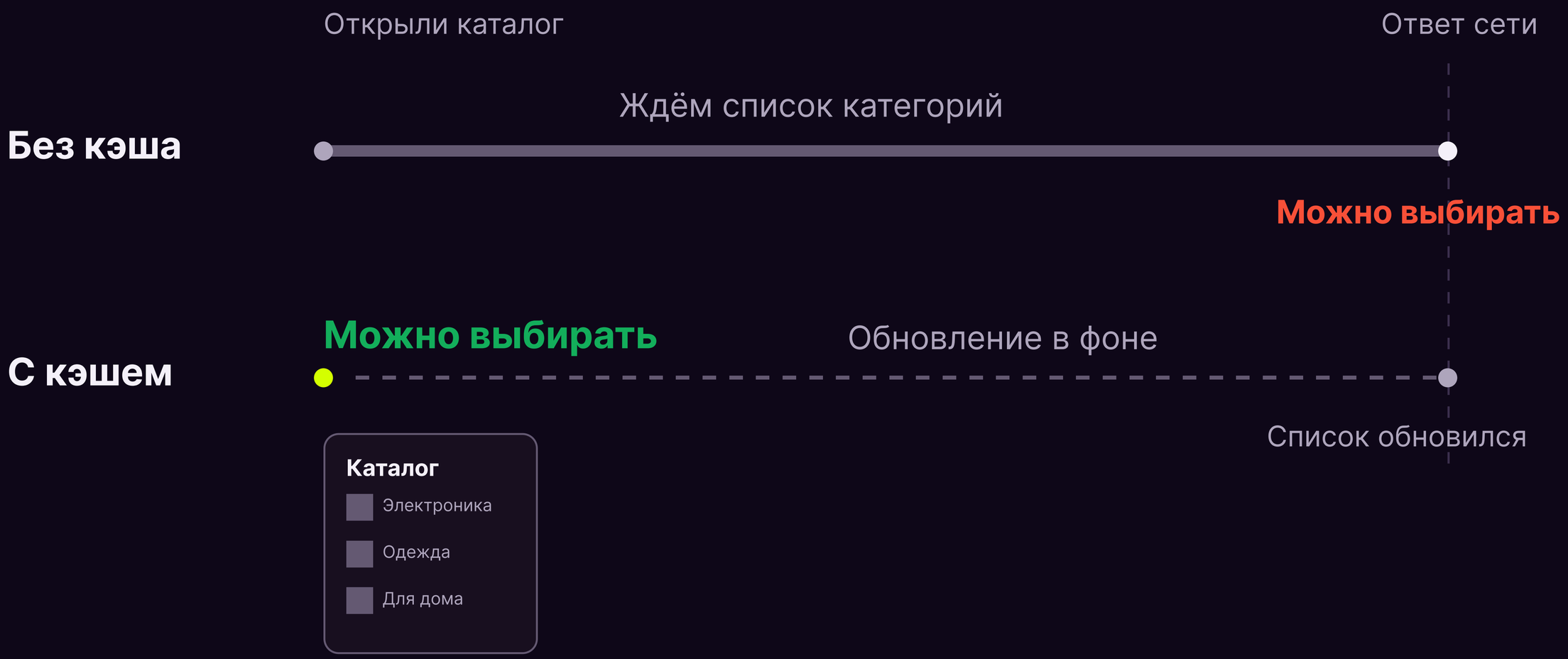
Проверяем эффект на своих пользователях



Когда можно начинать выбирать в каталоге?

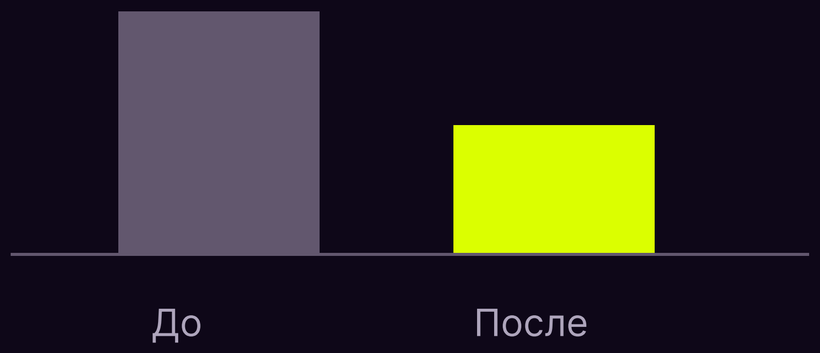


Когда можно начинать выбирать в каталоге?

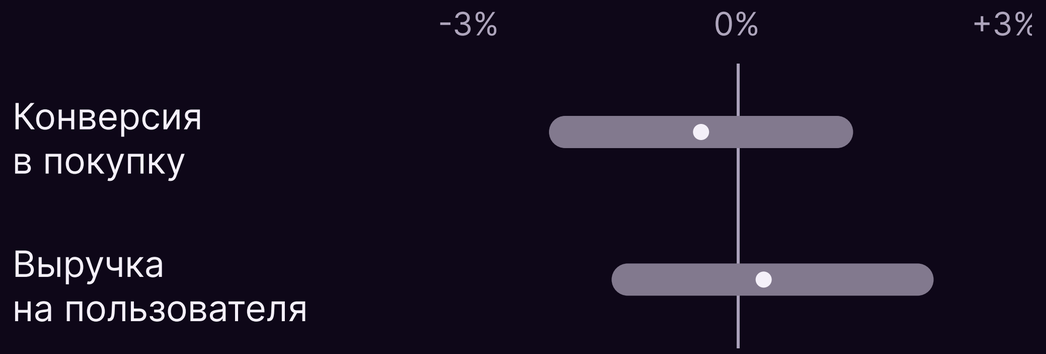


Ускорили каталог, но не увидели эффекта в A/B-тесте

Время загрузки



Поведение пользователей



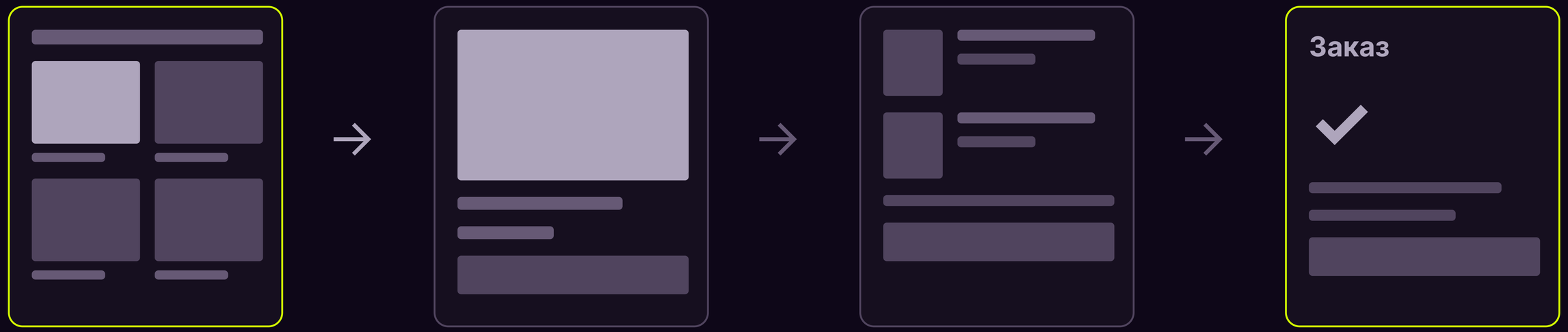
01 Что должно было измениться в поведении пользователя?

02 По какой метрике мы это проверяли?

Сначала смотрели на сумму покупок

Здесь ускорили

Каталог → **Карточка** → **Корзина** → **Покупка**

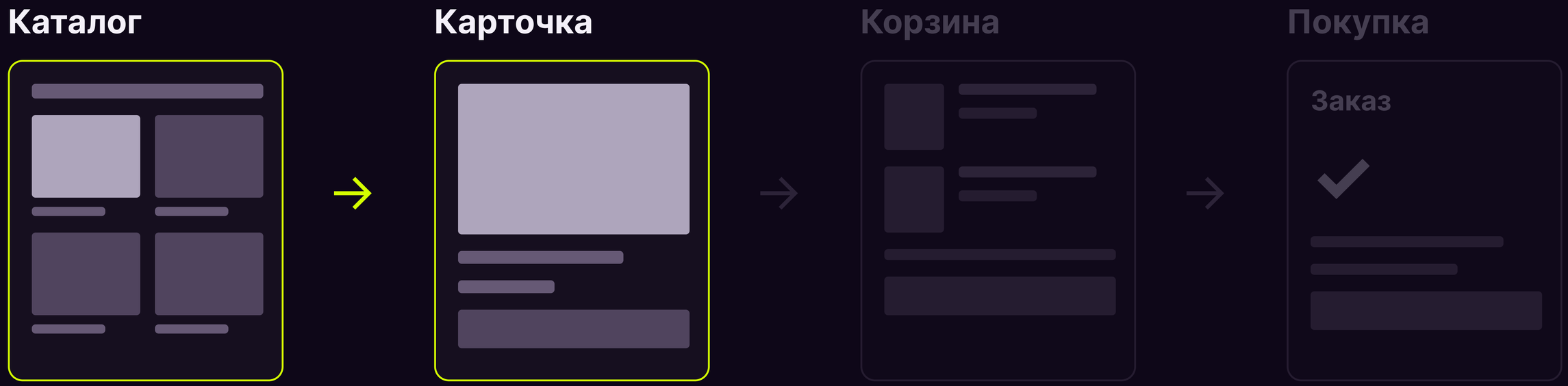


До покупки ещё несколько шагов

Цена · наличие · доставка · оплата

Проверяем переходы из каталога в карточку товара

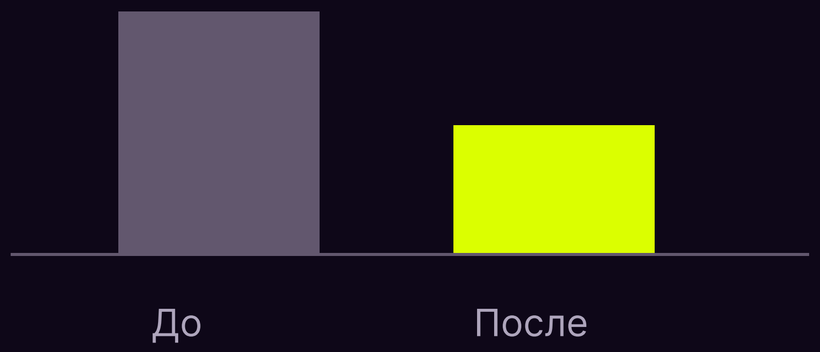
Здесь ускорили



Стали чаще открывать товары?

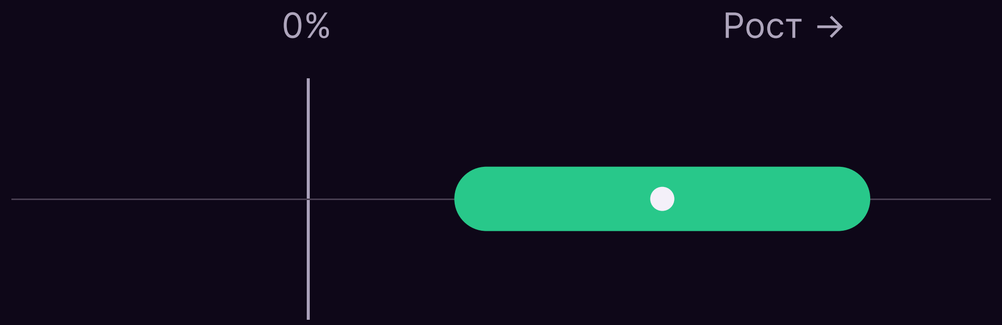
Переход из каталога в карточку

Время загрузки



Ускорение есть

Переход из каталога в карточку

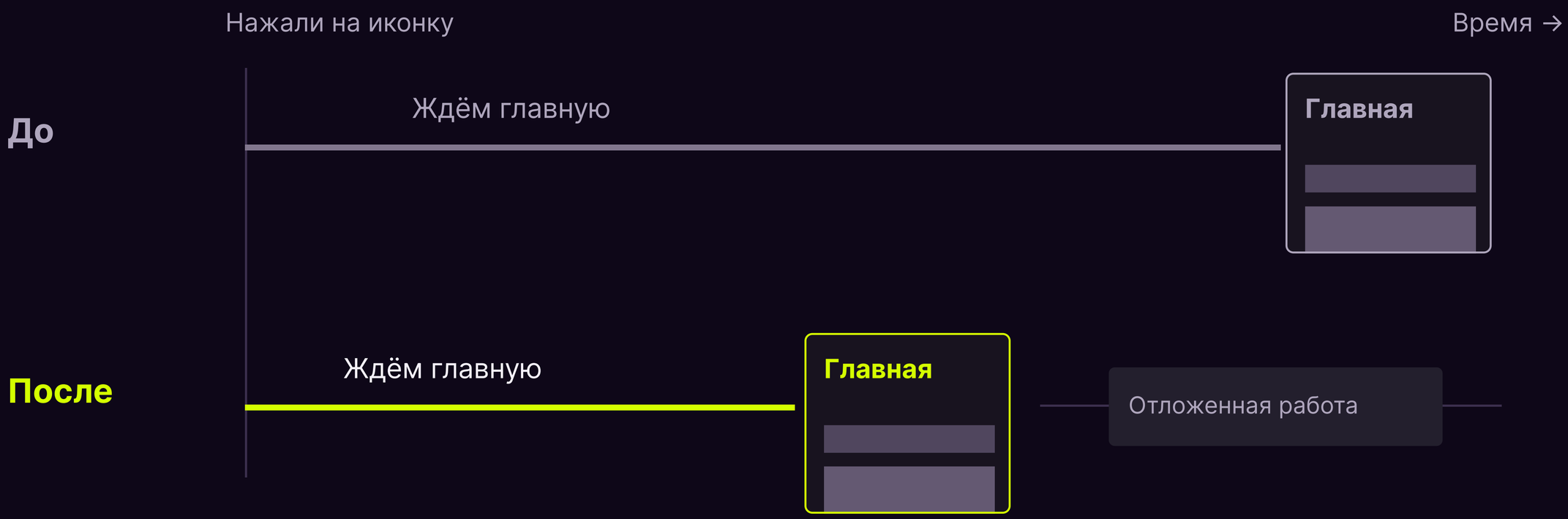


Статистически значимый рост

+0,6 %

Конверсия из категории в товар

Запуск приложения



Как проверяем эффект ускорения запуска?

GMV

Общая выручка

А какую метрику выбрали бы вы?

Как проверяем эффект ускорения запуска?

GMV

Слишком далеко от запуска

Выбор товара → корзина → покупка

На результат влияет много других вещей

Дошли до главной?

Запуск → просмотр главной

Доля запусков, в которых пользователь увидел главную

Ближайший шаг к нашему изменению

+0,7 %

Просмотр главной

Нет значимого эффекта в A/B-тесте ≠ нет пользы пользователю

Проверяем, что измеряли, у кого
и хватило ли данных.

Как заметить эффект в A/B-тесте

Kohavi, Longbotham · стр. 7–8
Выбор метрики и чувствительность теста



А если просто замедлить приложение на секунду?

Замедлили

+1 с

Кадр: «Вовка в Тридевятом царстве» · Культура.РФ



А если просто замедлить приложение на секунду?

Замедлили

+1 с

≠

по эффекту

Ускорили

−1 с

Кадр: «Вовка в Тридевятом царстве» · Культура.РФ



Эффект задержки зависит от исходной скорости

Экран изначально быстрее



Влияние сильнее

Экран изначально медленнее



Влияние слабее

Самый чувствительный диапазон в наших тестах: 1–3 с

Что дали нам эксперименты

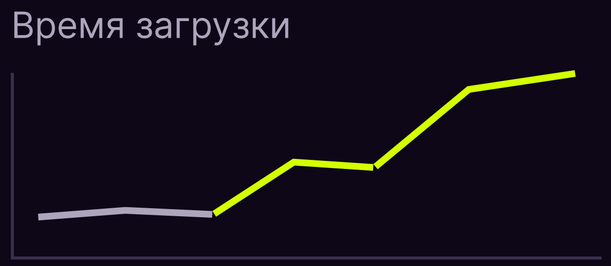
01 Оценили, сколько можем потерять из-за замедлений

02 Убедились, что результаты непросто интерпретировать

03 Нашли самый чувствительный диапазон ожидания: 1–3 с

И всё это — на наших пользователях

Как замечать проблемы раньше жалоб?



Пользователь

Каталог долго открывается

Начались тормоза

Увидели в метриках

Пришла жалоба

Можем начать разбираться раньше

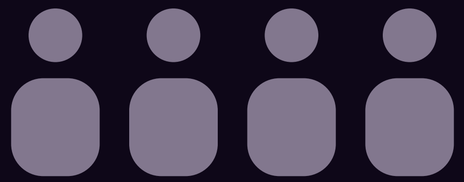
А что вы знаете о своём экране?

Как часто тормозит?



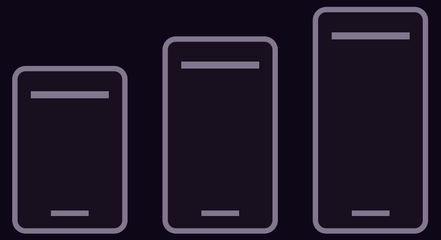
?

У скольких пользователей?



?

На каких телефонах?



?

В какой сети?



?

Добавили замеры в существующую аналитику

ДОБАВИЛИ

```
screen: MAIN
stage: ScreenLoad
value: 2400 мс
```

УЖЕ РАБОТАЛО

Performance SDK → **KMM Analytics** → **ClickHouse** → **Grafana**

Формирует события

Отправляет

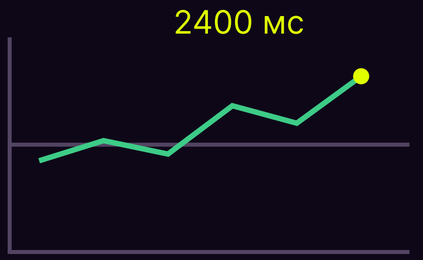


Событие → аналитика

Хранит измерения

Экран	мс
MAIN	2400

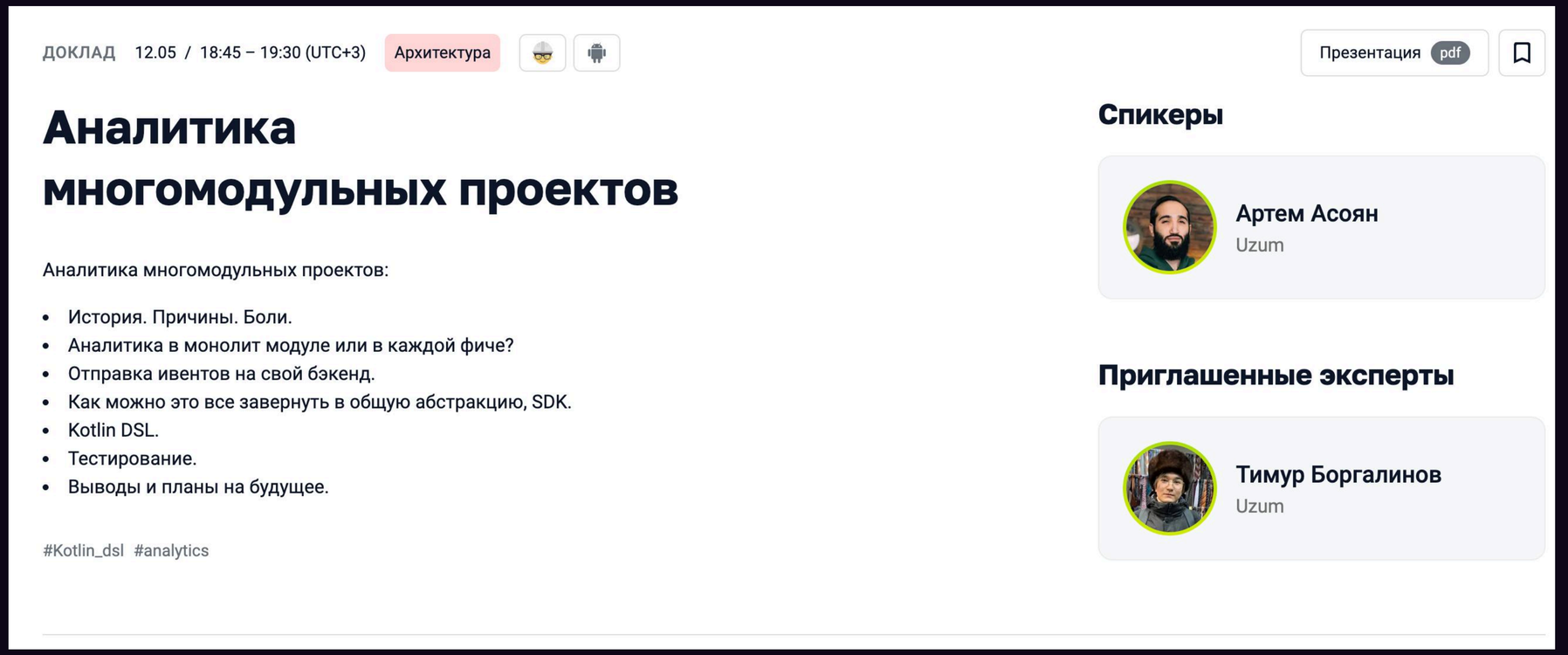
Показывает графики



Загрузка главной

Подробнее об аналитике и её SDK

Mobius 2023 Spring



ДОКЛАД 12.05 / 18:45 – 19:30 (UTC+3) Архитектура

Презентация pdf

Аналитика многомодульных проектов

Аналитика многомодульных проектов:

- История. Причины. Боли.
- Аналитика в монолит модуле или в каждой фиче?
- Отправка ивентов на свой бэкенд.
- Как можно это все завернуть в общую абстракцию, SDK.
- Kotlin DSL.
- Тестирование.
- Выводы и планы на будущее.

#Kotlin_dsl #analytics

Спикеры

Артем Асоян
Uzum

Приглашенные эксперты

Тимур Боргалинов
Uzum

Общий SDK для разных команд, но метрики одинаковые



С каких метрик начать?

Startup FPS Загрузка экрана ANR

DNS Hitch Rate RAM Frozen Frames TLS

Загрузка изображений CPU Время ответа

Расход батареи GC Размер трафика Дисковые операции

Чтобы начать, не нужно измерять всё сразу

С бизнесом

За чем следим и почему это важно?

С разработчиками

Что нужно знать, чтобы найти причину?

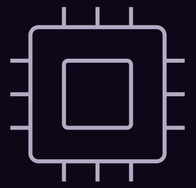
Ugly action beats perfection



Метрики помогают принимать решения

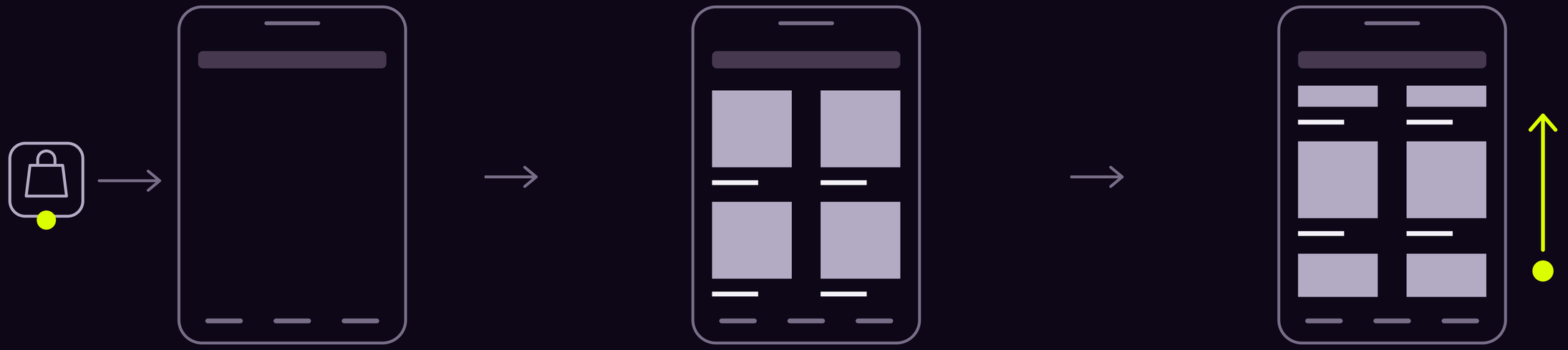


Основные показатели → **Что улучшать и куда вкладывать ресурсы**



Вспомогательные измерения → **Где искать причину**

Основные метрики



Время запуска приложения

Загрузка до полезного кадра

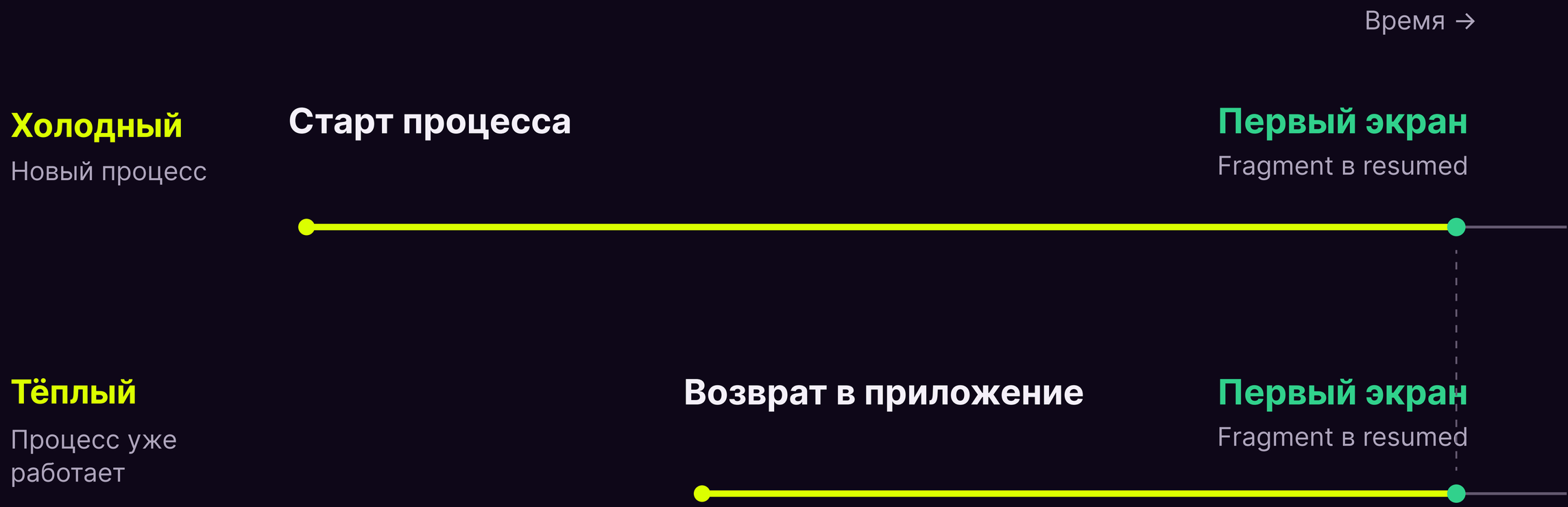
Плавность интерфейса

Время запуска



Источник GIF: Tenor

Как измеряем холодный и тёплый запуск




Запуск / возврат
Первый запуск процесса ?
Нет
Да
Cold · от старта процесса
Process.getStartUptimeMillis()
Warm · от возврата
В ProcessLifecycleOwner.onStart()
onFragmentResumed() → сохраняем замер
val startupMs = SystemClock.uptimeMillis() - baseline
Следующий возврат

Считаем запуск до конкретного экрана



Время	Cold / Warm	Первый экран	Версия приложения

Из этих данных SDK собираем отчёт

Время	Cold / Warm	Первый экран	Версия приложения
1,2 с 	Cold	Онбординг	1.87
1,2 с 	Cold	Онбординг	1.88
1,8 с 	Cold	Главная	1.87
2,6 с 	Cold	Главная	1.88

Из событий SDK собираем отчёт

Время	Cold / Warm	Первый экран	Версия приложения
1,2 с	Cold	Онбординг	1.87
1,2 с	Cold	Онбординг	1.88
1,8 с	Cold	Главная	1.87
2,6 с +800 мс	Cold	**Главная**	**1.88**

В релизе 1.88 замедлился холодный запуск на главную

«Приложение тормозит»



Кадр: «Винни-Пух» · Союзмультфильм

→

Релиз	**1.88**
Первый экран	**Главная · Cold**
Замедление	**+800 мс**
1,8 → 2,6 с	

Разбираем запуск на главную с командой экрана

Измеряем → **Уточняем** → **Выбираем действие**

Что изменилось? Где и у кого? Что проверять дальше?

Измеряем → **Уточняем** → **Выбираем действие**

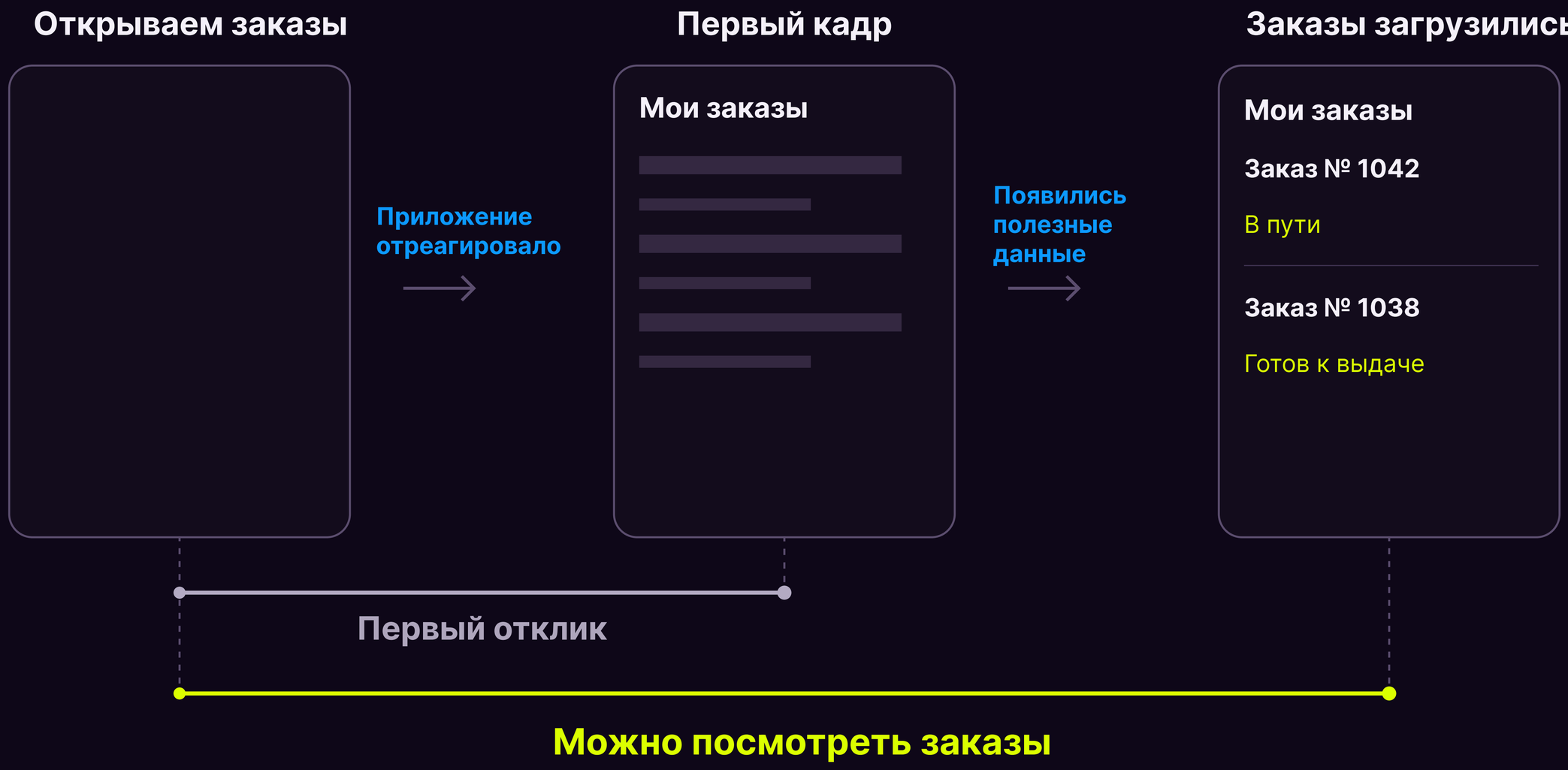
Что изменилось? Где и у кого? Что проверять дальше?

Метрика должна помогать принимать решение

Загрузка экрана



Источник GIF: Tenor


Открываем заказы
Первый кадр
Заказы загрузились
Мои заказы
Приложение отреагировало
Появились полезные данные
Мои заказы
Заказ № 1042
В пути
Заказ № 1038
Готов к выдаче
Первый отклик
Можно посмотреть заказы

Как SDK узнаёт о следующем кадре?

Сигнал обновления экрана

→

Choreographer

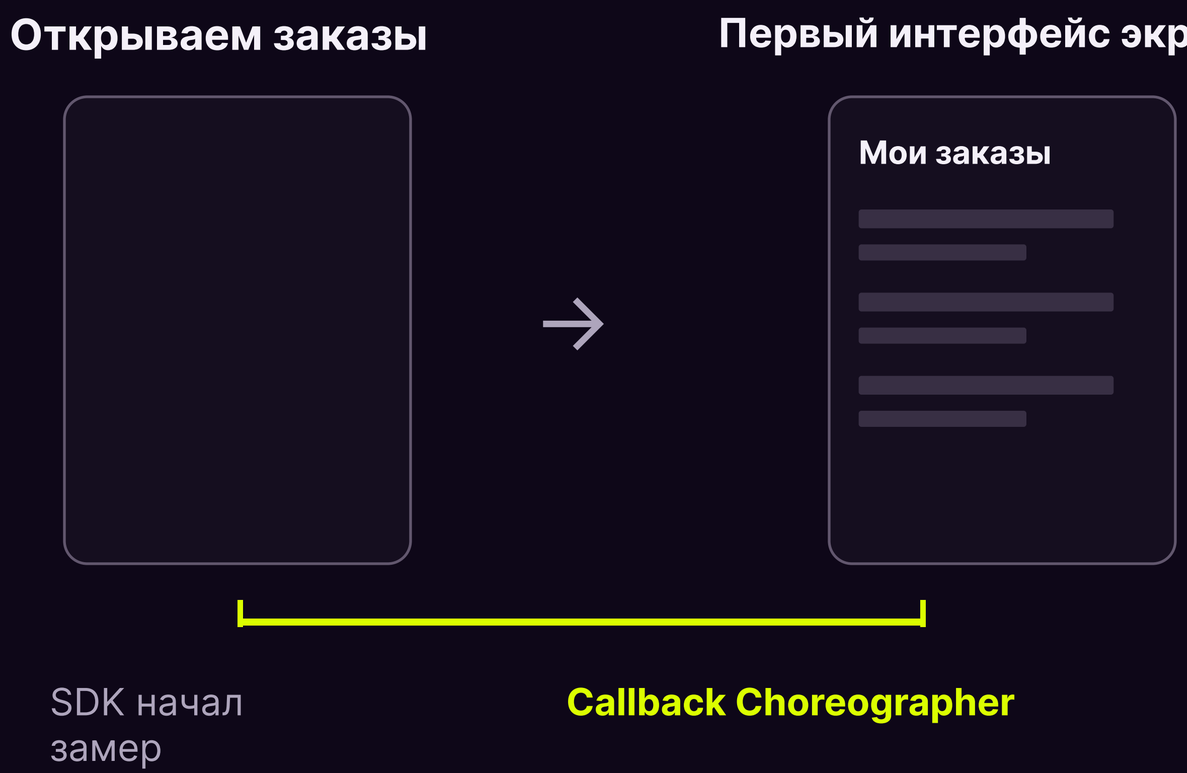
Координирует работу UI на следующий кадр

→

Функция SDK (callback)

Вызывается в начале обработки кадра

Задержка до первого кадра



Сигнал обновления экрана

Choreographer

```
val startMs = nowMs()

choreographer.postFrameCallback {
    save(nowMs() - startMs)
}
```

Задержка до callback выросла

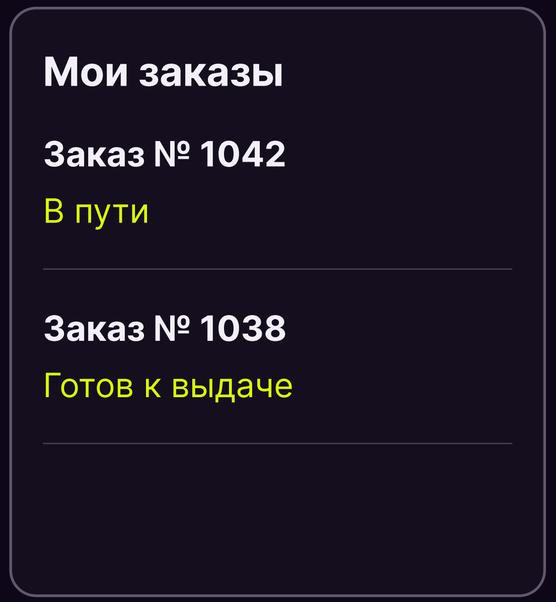
Открытие заказов · одинаковые условия



↓ **Проверяем работу UI-потока и подготовку экрана до первого кадра**

Screen Load Time

От начала замера
до нужного содержимого



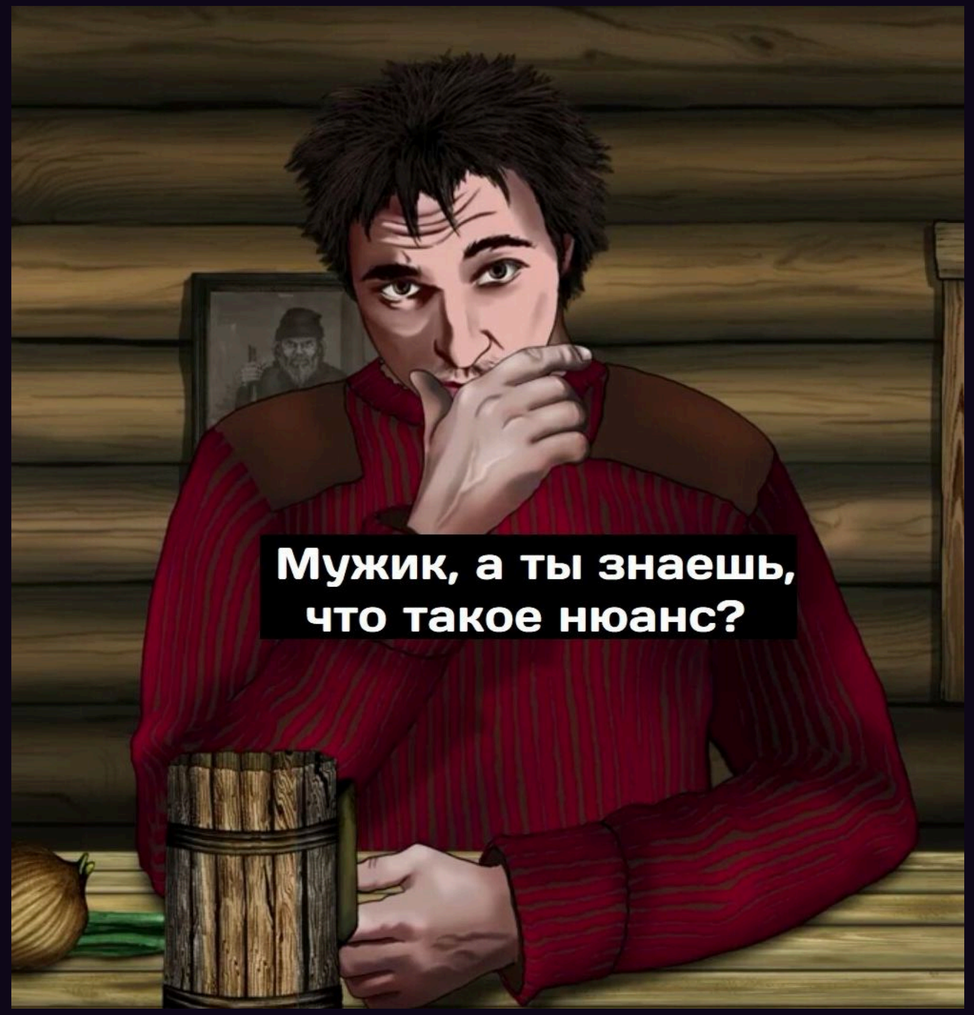
Начало замера

Заказы и статусы появились

2,5 с

Но есть нюанс.

По одной цифре не видно,
где возникла задержка.



Источник: Pikabu https://pikabu.ru/story/est_nyuans_10557443

С чего начать проверку?

Данные не готовы 2,3 с



Проверяем получение и подготовку данных

С чего начать проверку?

Данные не готовы 2,3 с

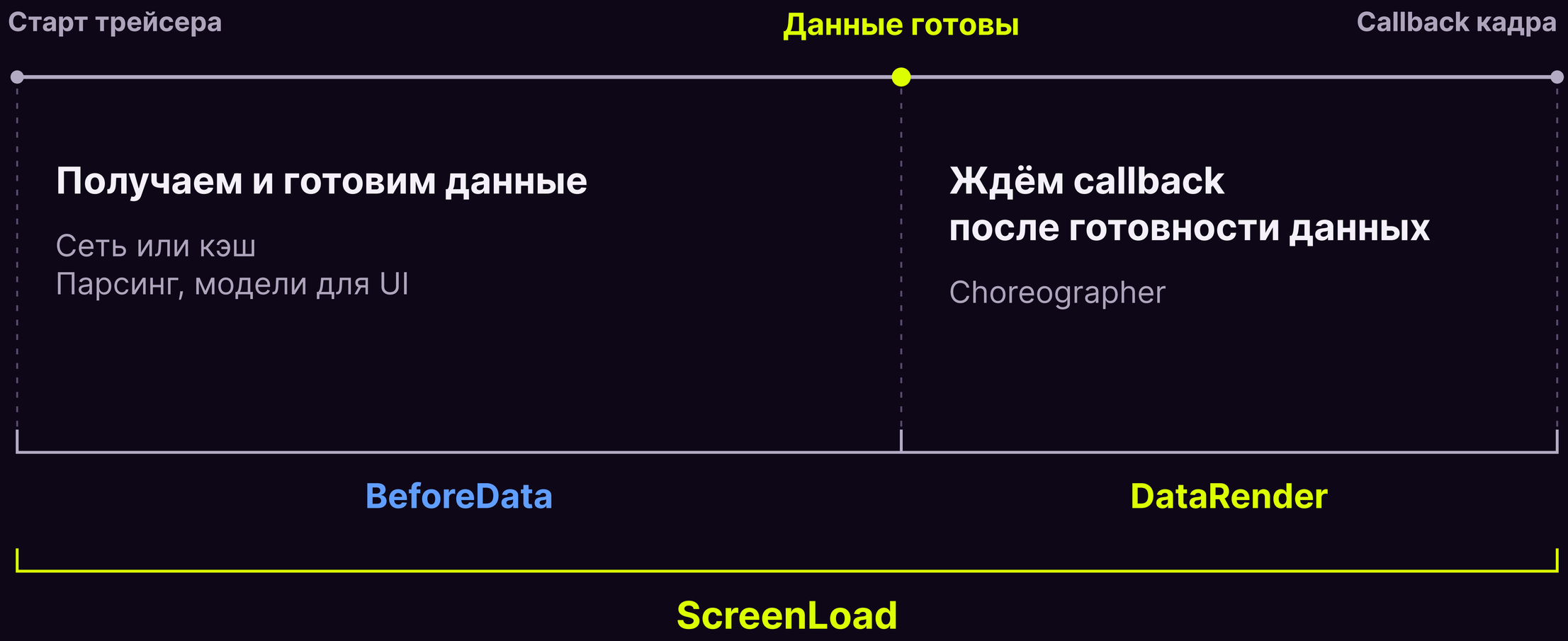


Проверяем получение и подготовку данных

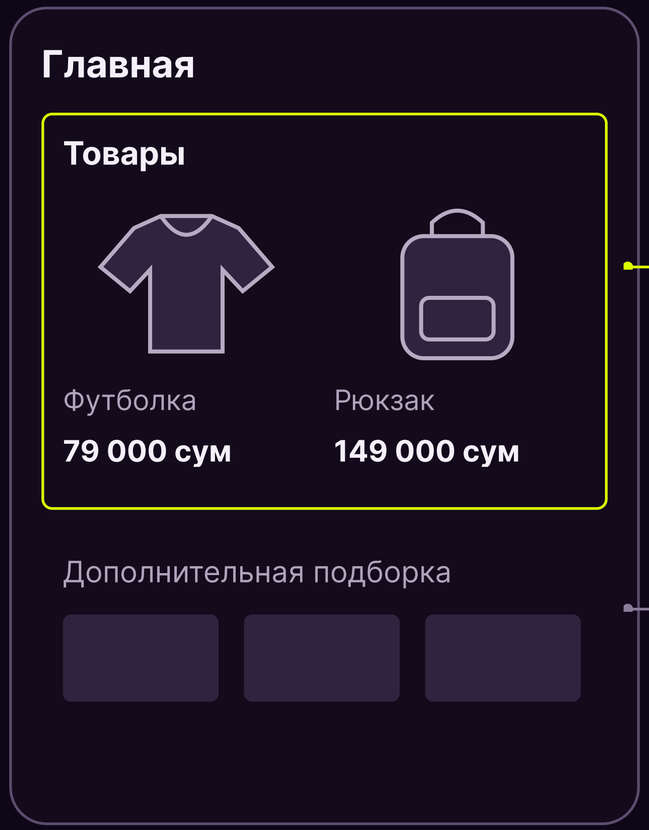
Данные готовы за 0,5 с. До callback ещё 2 с



Проверяем работу UI-потока


Старт трейсера
Данные готовы
Callback кадра
Получаем и готовим данные
Сеть или кэш
Парсинг, модели для UI
Ждём callback
после готовности данных
Choreographer
BeforeData
DataRender
ScreenLoad

Что значит полезный контент?



Ждём

Основной список товаров и цены

Может загрузиться позже

Дополнительная подборка

Команда переопределяет одно условие

Что пишет команда

```
override val dataFromNetworkReadyCondition =
    dataReadyWhen { screen →
        screen.productsLoaded
    }
```



Сначала договоритесь, что значит “экран готов”

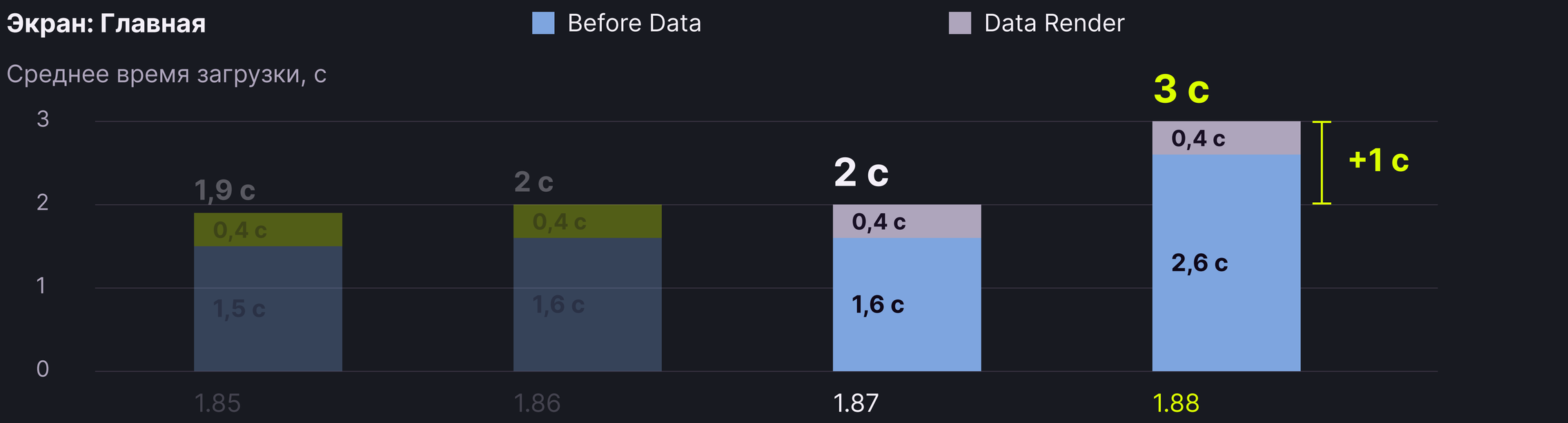
Команда задаёт условие готовности.
SDK измеряет время до него.



Кадр: «Губка Боб» · GIF: Tenor

Загрузка экрана

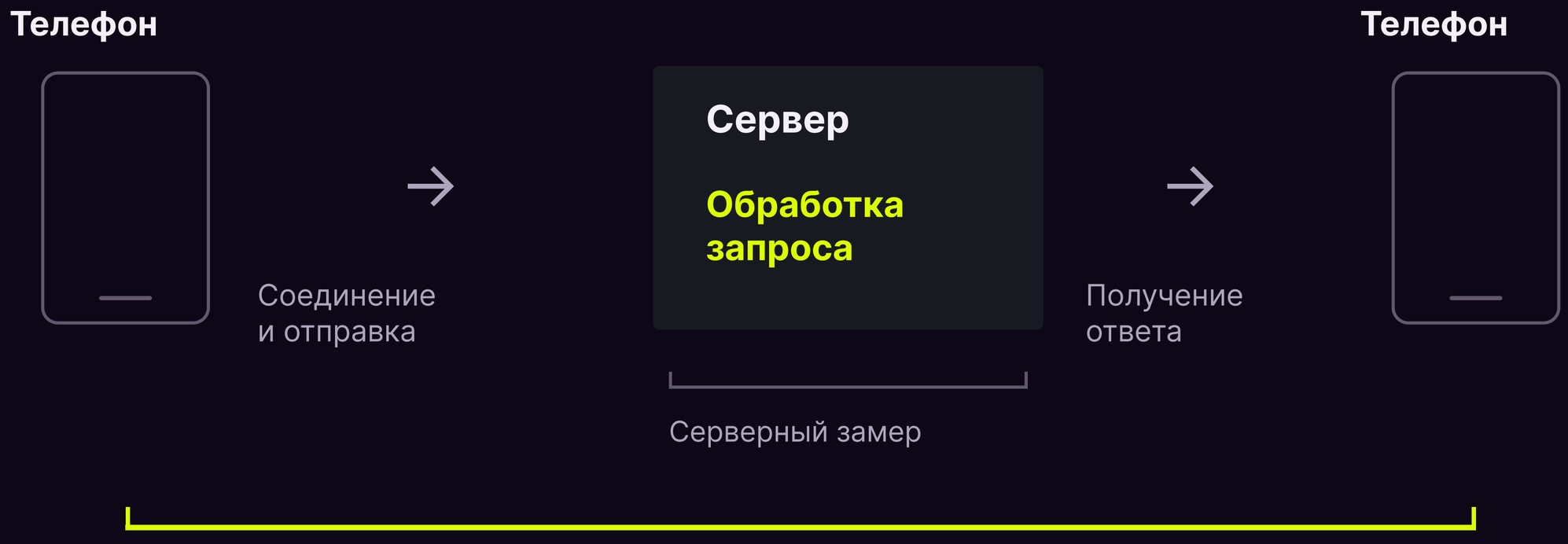
Экран: Главная



Сетевые запросы



У бэкенда уже есть графики. Зачем измерять на телефоне?



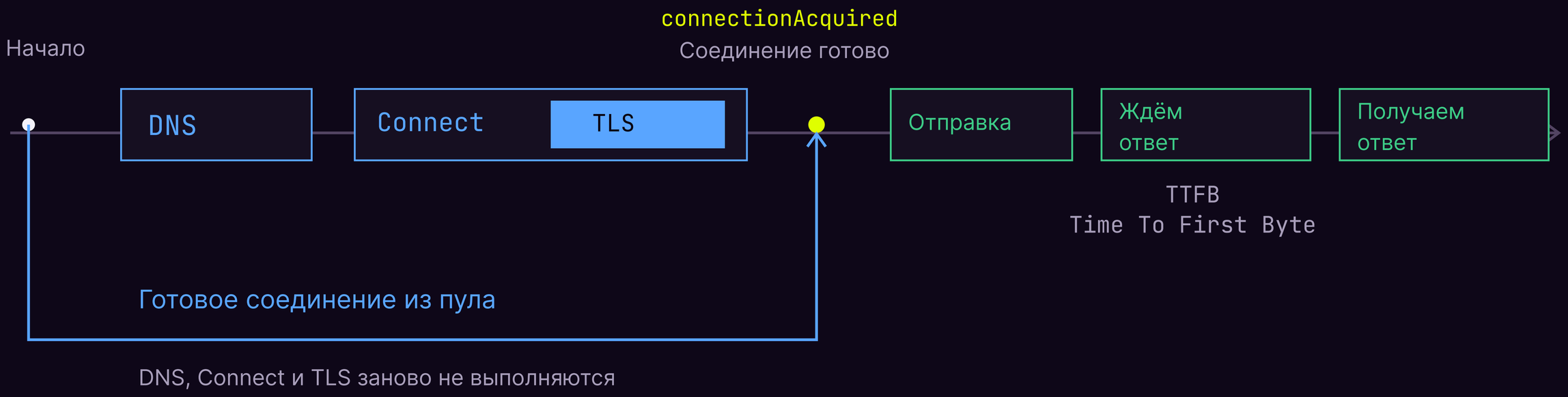
На телефоне измеряем весь путь запроса

На что уходит время сетевого запроса

Время идёт слева направо. Показана одна успешная попытка без повторов.

1. Получаем соединение

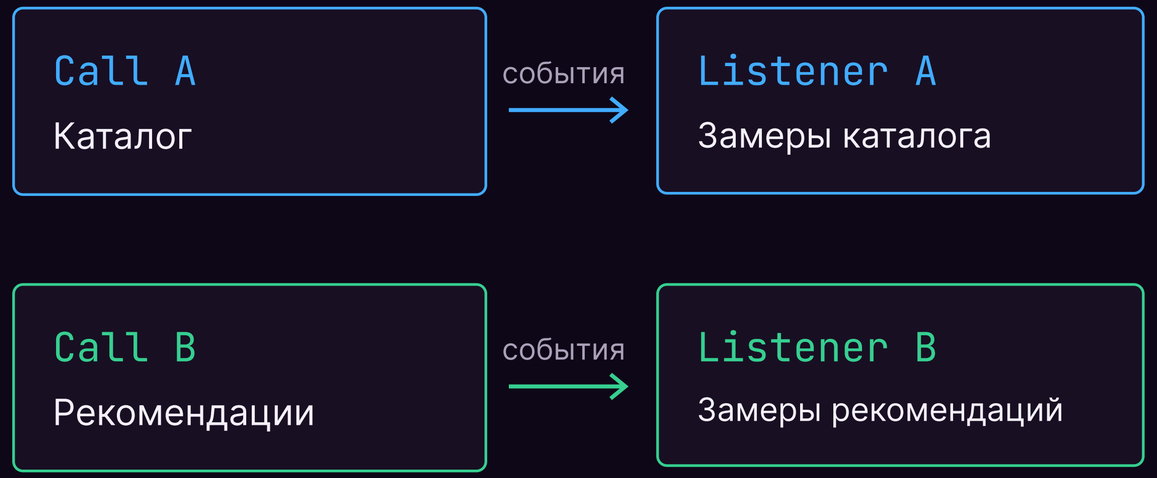
2. Отправляем запрос и читаем ответ



Как подключаем EventListener к OkHttp

1. Настраиваем общий клиент

```
val client = OkHttpClient.Builder()
    .eventListenerFactory {
        timingTracker.createEventListener()
    }
    .build()
```



2. Считаем длительность этапа

`dnsStart()` → `dnsEnd()` → длительность DNS

Одной длительности запроса нам мало

Сколько длится?

Время запроса

Как часто?

Доля всех вызовов

Ждёт пользователь?

Можно ли продолжить сценарий

Какой запрос ускорять первым?

Запрос	Доля всех вызовов ↓	Время	Пользователь ждёт?
Загрузить профиль	 40%	**1,5 с**	**Да**
Загрузить главную	 30%	**0,6 с**	**Да**
Синхронизация избранного	 20%	**5 с**	**Нет**
Получить конфигурацию	 10%	**0,3 с**	**Да**

Какой запрос ускорять первым?

Запрос	Доля всех вызовов ↓	Время	Пользователь ждёт?
Загрузить профиль	40%	1,5 с	Да
Загрузить главную	30%	0,6 с	Да
Синхронизация избранного	20%	5 с	Нет
Получить конфигурацию	10%	0,3 с	Да

Длительность · Частота вызовов · Ждёт ли пользователь

1260 из 1500 мс ждём начала ответа



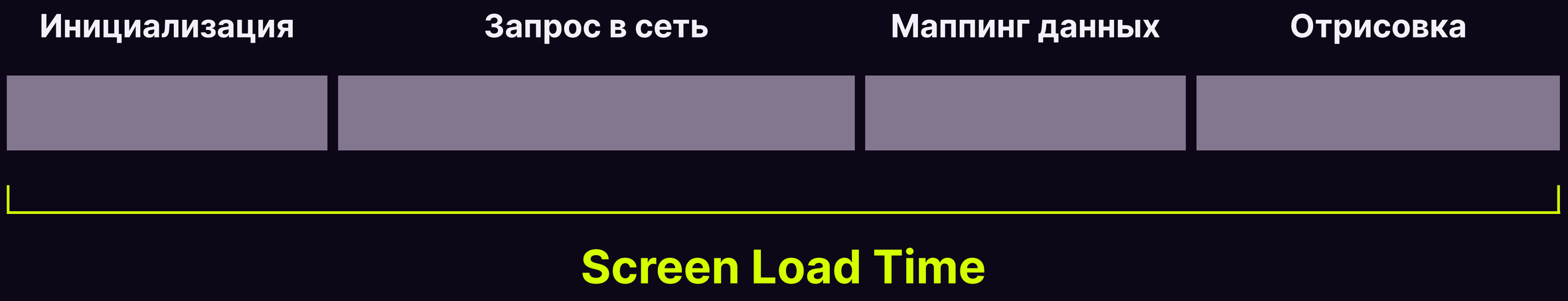
DNS	Connect + TLS	Отправка	Ожидание ответа	Чтение
40 мс	**100 мс**	**20 мс**	**1260 мс**	**80 мс**

TTFB: от отправленного запроса до первого байта ответа

Запрос в пути → Обработка на сервере → Ответ в пути

С бэкендом сверяем время обработки этого запроса

Что входит в Screen Load Time



Начало открытия

Нужное содержимое появилось

Плавность интерфейса



Источник GIF: Tenor

Hitch Rate (плавность скролла)

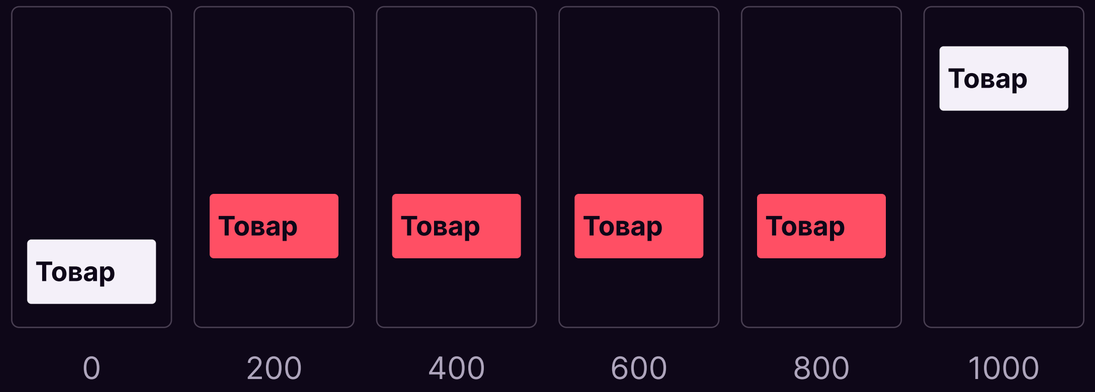
Рывки при движении



Короткие остановки и скачки

Frozen Frames (наличие фризов)

Долгое зависание ≥ 700 мс

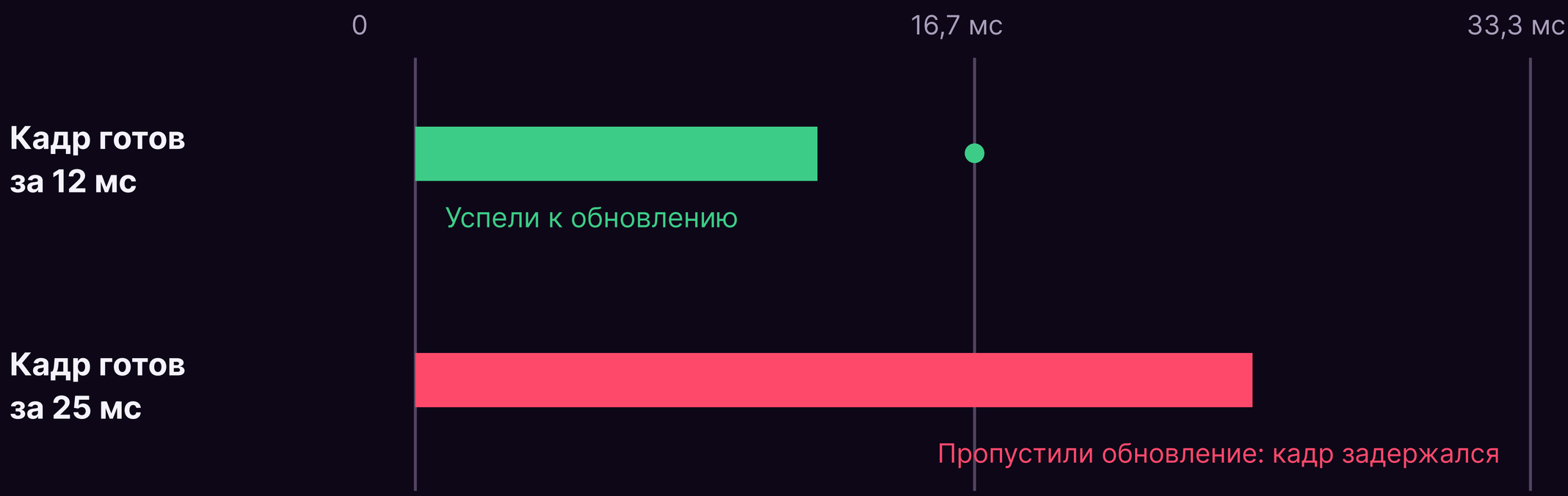


800 мс без обновления

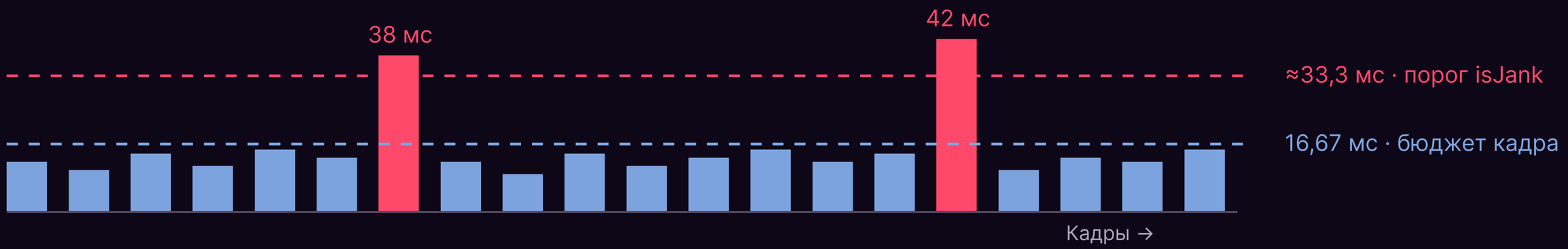
Не успели подготовить кадр - движение дёргается



Не успели к обновлению — кадр задержался



Hitch rate: какая доля кадров была проблемной



В SDK · window из Activity

```
DisposableEffect(window) {
    val stats = JankStats.createAndTrack(window) { frame →
        frameStats.frames++
        if (frame.isJank) frameStats.jankFrames++
    }
    onDispose { stats.isTrackingEnabled = false }
}
```

```
2 / 20 × 100 = 10%
```

Hitch rate этой сессии

Frozen Frames: в скольких сессиях были фризы

Хотя бы один кадр **≥ 700 мс** → «грязная» сессия



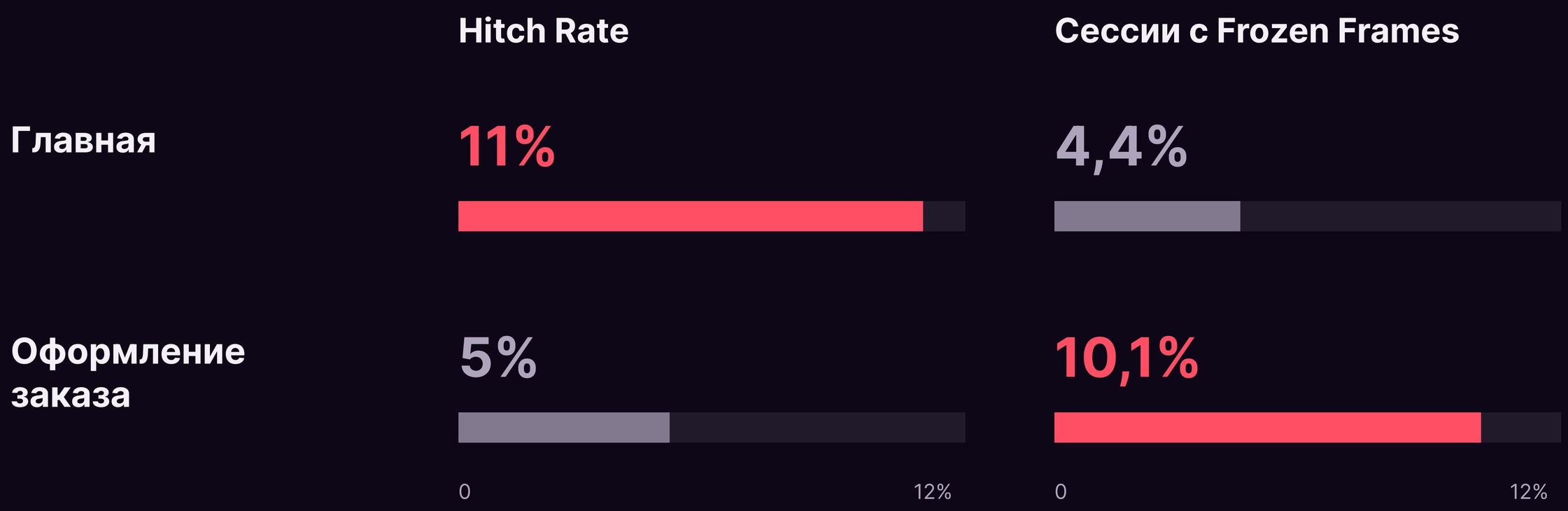
В SDK · порог 700 мс

```
// В том же callback JankStats
if (frame.frameDurationUiNanos ⩾ 700_000_000L) {
    frameStats.frozenFrames++
}
// Грязная сессия: frozenFrames > 0
```

1 из 5 = 20%

Сессии с зависанием

Меньше рывков не значит меньше фризов

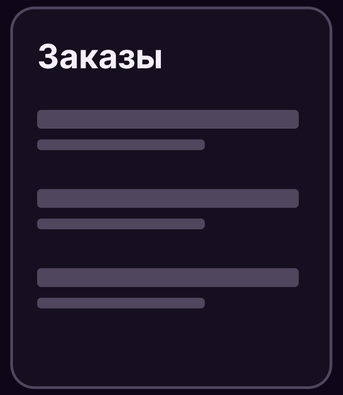


Теперь мы видим

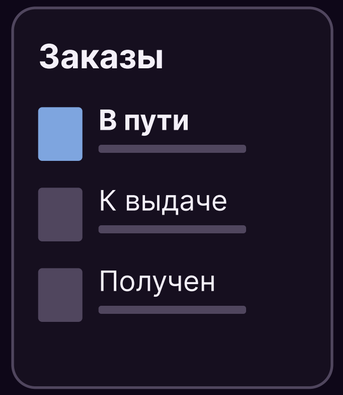
Запустил



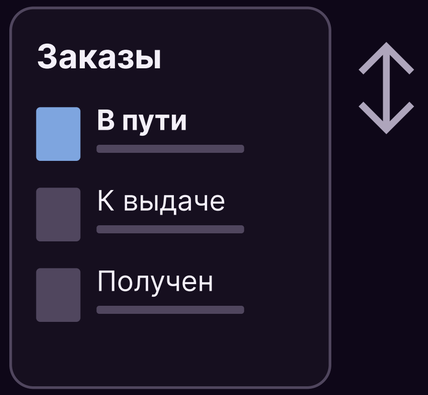
Открыл экран



Дождался данных



Взаимодействует



App Startup Time

Запуск приложения

Задержка до первого кадра

До callback Choreographer

Screen Load Time

Ожидание содержимого

Hitch Rate



Сессии с Frozen Frames



> 700 мс



“Two men looked out from prison bars,
One saw the mud, the other saw stars.”

«Двое смотрели сквозь тюремные решётки:
один увидел грязь, другой - звёзды».

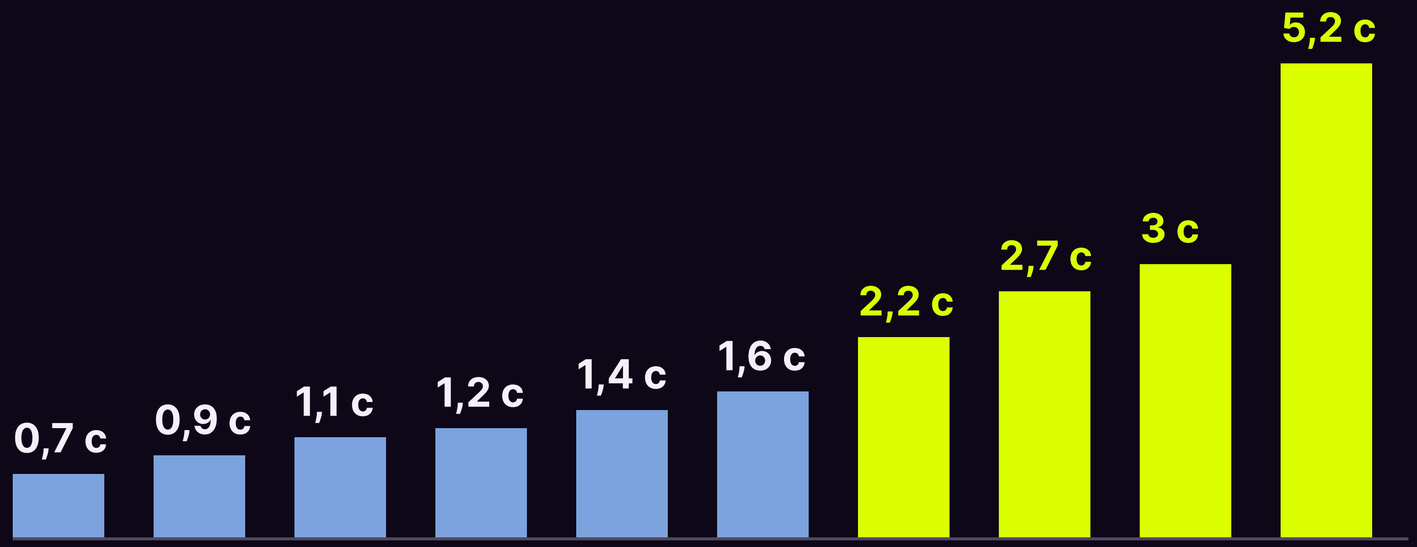
Цитируется у Дейла Карнеги · «Как перестать беспокоиться и начать жить»

Достаточно ли среднего?

У пользователей всё хорошо?

Среднее: 2 с

Среднее скрывает медленные загрузки



Среднее: 2 с

Некоторые загрузки занимают заметно больше двух секунд

Среднее скрывает медленные загрузки



Среднее: 2 с

Некоторые загрузки занимают заметно больше двух секунд

p90: 3 с

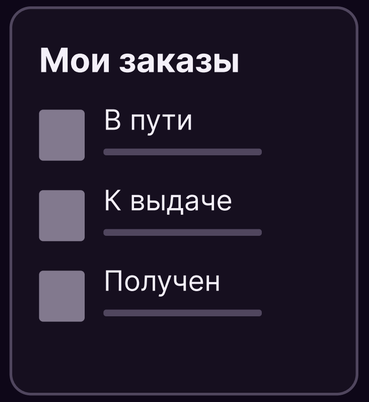
9 из 10 загрузок уложились в 3 с

Какой перцентиль смотреть?

p50	50 из 100	Типичное время ожидания
p75	75 из 100	Большая часть загрузок
p90	90 из 100	Граница самых медленных 10%
p95	95 из 100	Граница самых медленных 5%

Время загрузки по типам сети

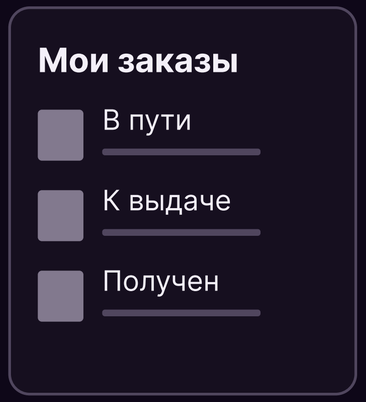
Wi-Fi



1,5 с

0 3 с

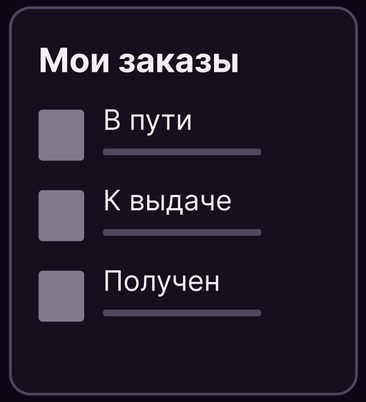
5G



1,7 с

0 3 с

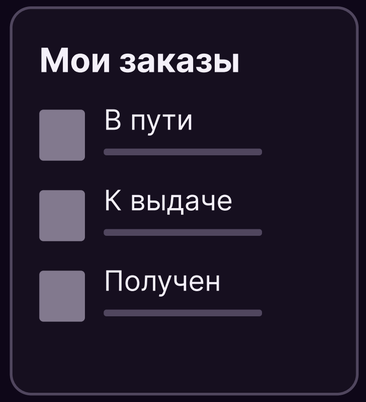
4G



2,1 с

0 3 с

3G



3 с

0 3 с

Общий p90 снизился. Приложение стало быстрее?

Предыдущая версия

p90: 3 с

→

Новая версия

p90: 1,5 с

Общий p90 снизился. Приложение стало быстрее?

Предыдущая версия

p90: 3 с

→

Новая версия

p90: 1,5 с

Wi-Fi 1,5 с → 1,5 с

3G 3 с → 3 с

Общий p90 снизился. Приложение стало быстрее?

Предыдущая версия

p90: 3 с

→

Новая версия

p90: 1,5 с

Wi-Fi 1,5 с → 1,5 с

3G 3 с → 3 с

Предыдущая: Wi-Fi 50% · 3G 50%



Общий p90 снизился. Приложение стало быстрее?

Предыдущая версия

p90: 3 с

→

Новая версия

p90: 1,5 с

Внутри сетей время не изменилось

Wi-Fi 1,5 с → 1,5 с

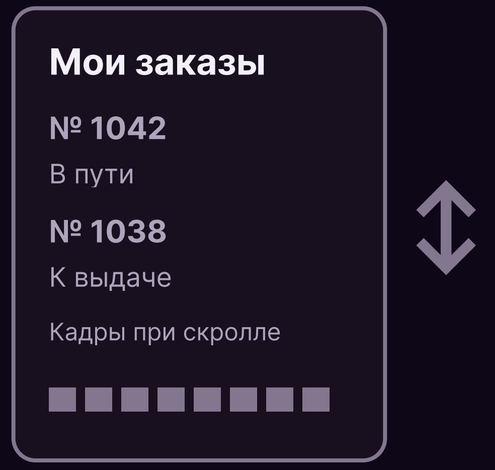
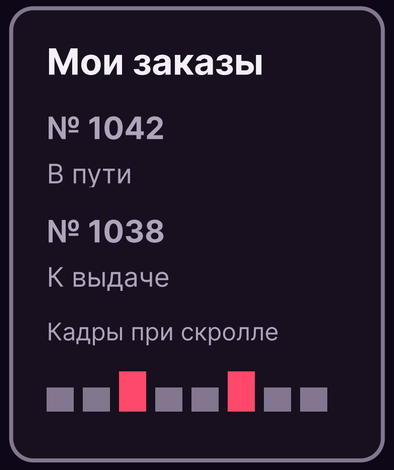
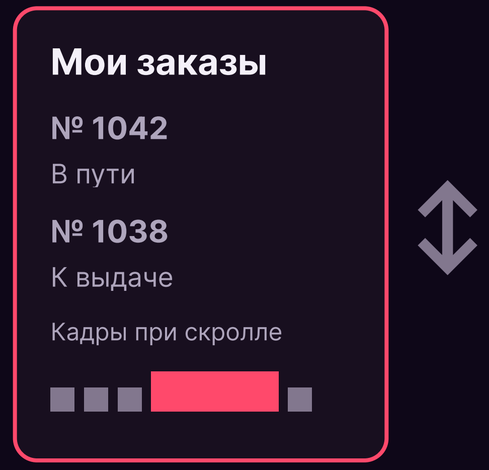
3G 3 с → 3 с

Предыдущая: Wi-Fi 50% · 3G 50%



Новая: Wi-Fi 95% · 3G 5%



	High · мощные	Mid · средние	Low · слабые
			
Hitch Rate	2%	4%	12%
Сессии с Frozen Frames	0,5%	2%	8%

Объём RAM

Число ядер

Максимальная частота экрана



Класс устройства

Low / Mid / High

Определяет SDK

Эти фильтры помогают точнее увидеть проблему

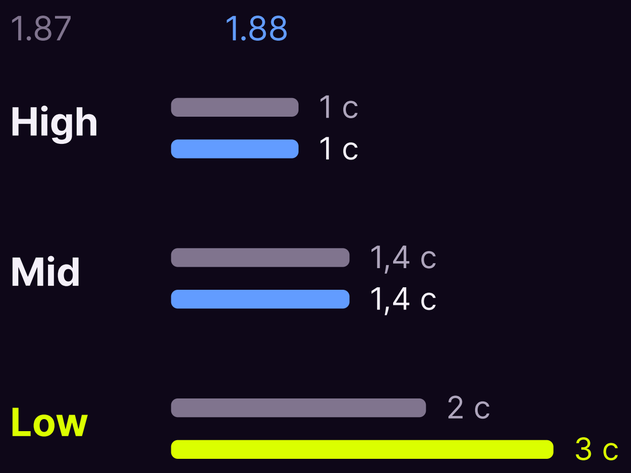
2 с → **3 с**

Релиз 1.87

Релиз 1.88

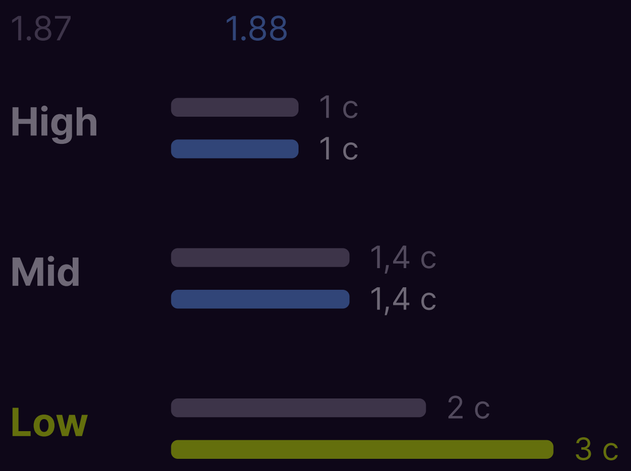
Где замедлилась загрузка в релизе 1.88

1. Сравнили устройства



Где замедлилась загрузка в релизе 1.88

1. Сравнили устройства



2. Уточнили группу

Low

Класс устройств

3G

тип сети

Где замедлилась загрузка в релизе 1.88

1. Сравнили устройства



2. Уточнили группу

Low

Класс устройств

↓

3G

тип сети

3. Сравнили этапы

p90 этапов · Low · 3G

1.87 → 1.88

До готовности данных

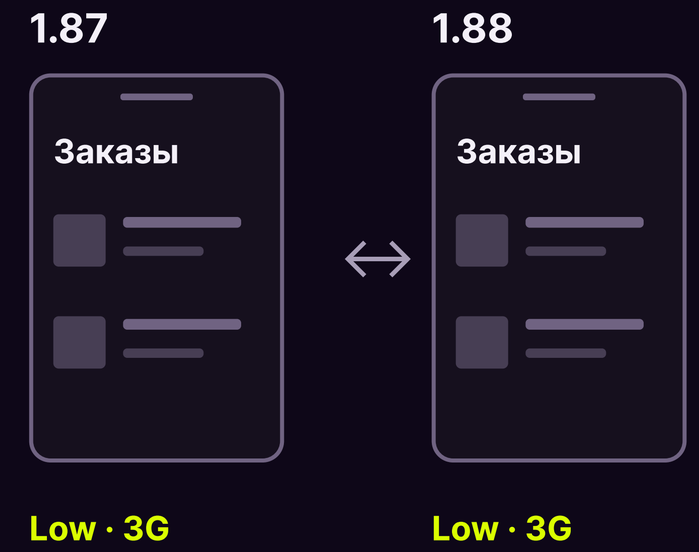
1,6 с → 2,6 с

После готовности данных

0,4 с → 0,4 с

Сравниваем сценарии в одинаковых условиях

Один сценарий
Один тип устройств
Один тип подключения



Нашли лишнюю секунду, но пока не знаем причину

НАШЛИ В ДАННЫХ

Заказы · Low · 3G

До готовности данных

1,6 с → 2,6 с



ПРИЧИНА

?

Что добавило секунду?

Диагностические метрики

Запросы



Сколько длится каждый этап запроса

Ищем, где выросло время

CPU



Насколько загружен CPU нашим процессом

Ищем лишнюю работу

Память



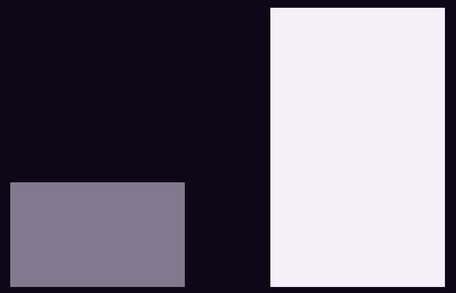
Сколько памяти занимает процесс

Проверяем выделения и удержание объектов

После релиза профиль чаще зависает

Сессии с Frozen Frames

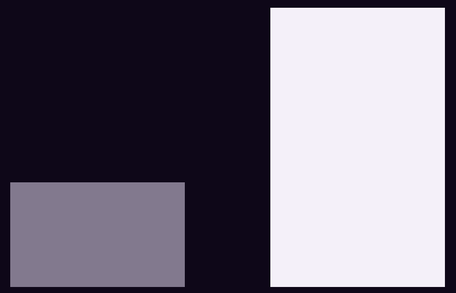
3% → 8%



После релиза профиль чаще зависает

Сессии с Frozen Frames

3% → 8%



→

Загрузка CPU · p90

18% → 42%

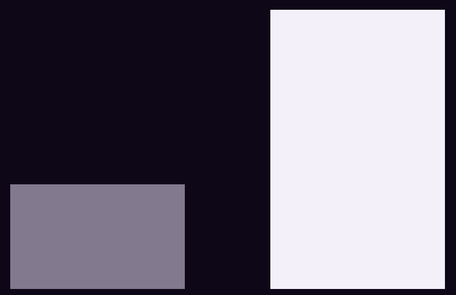
Память PSS · p90

220 → 225 МБ

После релиза профиль чаще зависает

Сессии с Frozen Frames

3% → 8%



→

Загрузка CPU · p90

18% → 42%

Память PSS · p90

220 → 225 МБ

→

Следующая проверка

Открываем CPU-профайлер

Мы разобрали, что измерять
и как читать данные

А сколько работы нужно команде?

Замеры подключены один раз в базовых классах

ОБЩАЯ БАЗА

Base Fragment

SDK уже подключён

Base ViewModel

условие готовности данных

Главная

замеры экрана



Каталог

замеры экрана



Заказы

замеры экрана



Команде остаётся указать три вещи

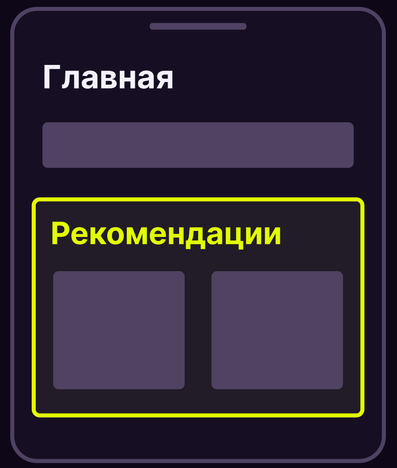
Например, экран заказов

		Где задаём
Имя экрана	**MainScreen**	Fragment
Команда	**GrowthTeam**	Fragment
Данные готовы	**state.productsLoaded = true**	ViewModel

Дальше SDK измеряет сам

Гибкость тоже важна!

Блок рекомендаций



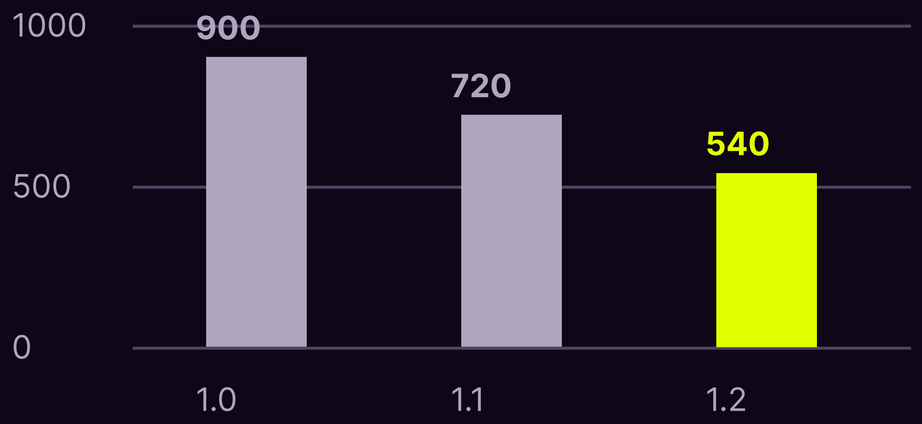
Performance SDK

Этап + длительность

Общая отправка
через KMM Analytics

График команды

Запрос рекомендаций · p90, мс



Свой замер через SDK

```
performance.measure(
    screen = MAIN,
    team = GROWTH,
    stage = Stage.NetworkRequestsTotalTime("recommendations"),
) {
    repository.loadRecommendations()
}
```

```
// В SDK: Stage описывает типы работы
sealed class Stage {
    data class NetworkRequestsTotalTime(val requestName: String) : Stage()
    data class MapToDomainTime(val requestName: String) : Stage()
    // добавляем новые логические этапы
}
```

Дополнительные замеры - тоже через SDK

Performance SDK

Загрузка экрана	Плавность	Свой этап
автоматически	автоматически	добавляет команда

Без отдельного трекера для каждого сценария

Собираем измерения в одном дашборде

Performance / Обзор приложения

Версии

1.87, 1.88 ▾

Performance / Обзор приложения

Версии
1.87, 1.88 ▾

Команды
Growth + 2 ▾

Performance / Обзор приложения

Версии
1.87, 1.88 ▾

Команды
Growth + 2 ▾

Экраны
Главная + 2 ▾

Performance / Обзор приложения

Версии	Команды	Экраны	Класс устройства
1.87, 1.88 ▾	Growth + 2 ▾	Главная + 2 ▾	Low ▾

Performance / Обзор приложения

Версии	Команды	Экраны	Класс устройства	Модель устройства
1.87, 1.88 ▾	Growth + 2 ▾	Главная + 2 ▾	Low ▾	Все ▾

Performance / Обзор приложения

Версии	Команды	Экраны	Класс устройства	Модель устройства	Сеть
1.87, 1.88 ▾	Growth + 2 ▾	Главная + 2 ▾	Low ▾	Все ▾	3G ▾

Performance / Обзор приложения

Версии	Команды	Экраны	Класс устройства	Модель устройства	Сеть
1.87, 1.88 ▾	Growth + 2 ▾	Главная + 2 ▾	Low ▾	Все ▾	3G ▾

Перцентиль

p90 ▾

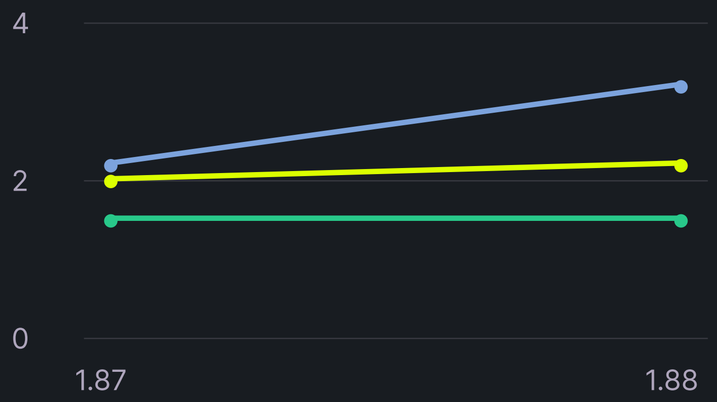
Все измерения в одном дашборде

Performance / Обзор приложения

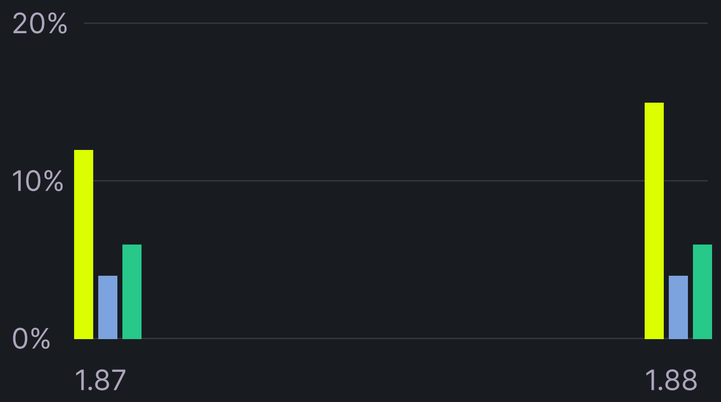
Версии: 1.87, 1.88 ▾
Команды: Growth + 2 ▾
Экраны: Главная + 2 ▾
Класс устройства: Low ▾
Модель устройства: Все ▾
Сеть: 3G ▾
Перцентиль: p90 ▾

Главная — Профиль — Каталог

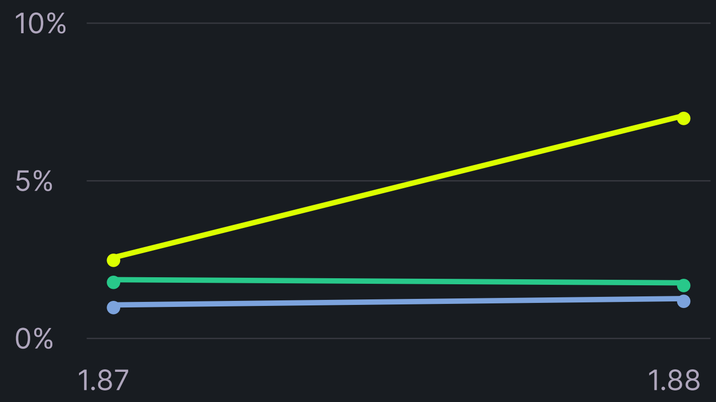
Загрузка экрана · p90, с



Hitch Rate, %



Сессии с Frozen Frames, %



Дашборд показал, где и у кого тормозит

А где ещё пригодятся эти данные?

За блоки одного экрана отвечают разные команды



Каждая полоса — 100% событий экрана

Главная: 4 команды

На одном экране сходятся блоки разных команд.

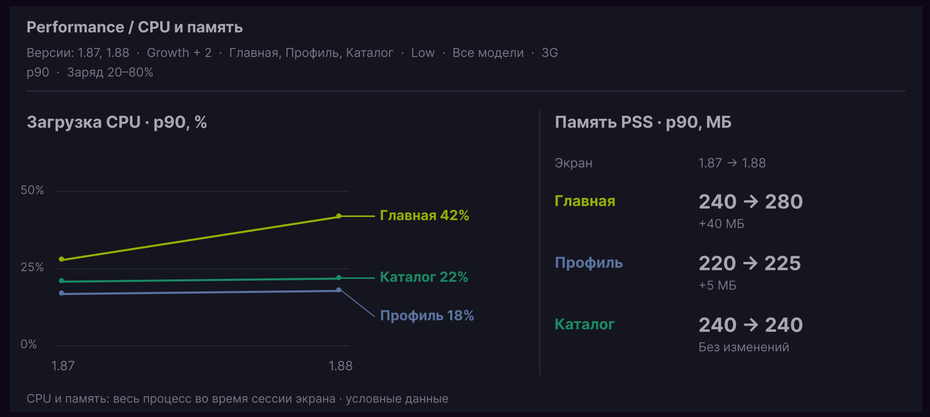
Что проверяем

Не мешает ли загрузка одного блока другим? Разбираем вместе с их владельцами.

А если учитывать эти данные ещё до разработки?

Уже знаем пользователей

Их телефоны и условия связи



ПОИСК ПРОБЛЕМ

Где и почему тормозит?

ПЛАНИРОВАНИЕ

Что учесть в новой фиче?

К замерам добавляем поведение в приложении

Технические замеры

Устройства



Low 45% · Mid 35% · High 20%

Качество связи



Стабильная 60% · С перебоями 40%

Продуктовые данные

Сценарии



Просмотр 65% · Повторная покупка 35%

По данным составляем портреты пользователей

Устройства



Low 45% · Mid 35% · High 20%

Качество связи



Стабильная 60% · С перебоями 40%

Сценарии



Просмотр 65% · Повторная покупка 35%

Смотрим сочетания

01

Листает каталог

Слабый телефон

Стабильная связь

02

Возвращается к покупке

Средний телефон

Связь с перебоями

Предлагаем улучшения для разных пользователей

01

Листает каталог

Слабый телефон

02

Возвращается к покупке

Связь с перебоями

↓

Облегчить эффекты

Не загружать всё сразу

↓

Показать сохранённое

Сохранить незавершённый шаг

Проверяем новую фичу на наших пользователях



01

Листает каталог

Слабый телефон



02

Возвращается к покупке

Связь с перебоями



Портреты из наших данных

ПРИМЕР ДЛЯ ГРУММИНГА

Подборка товаров

01 Слабый телефон

Плавно ли листается подборка?

02 Связь с перебоями

Что останется на экране без сети?

Теперь покажу несколько наших кейсов

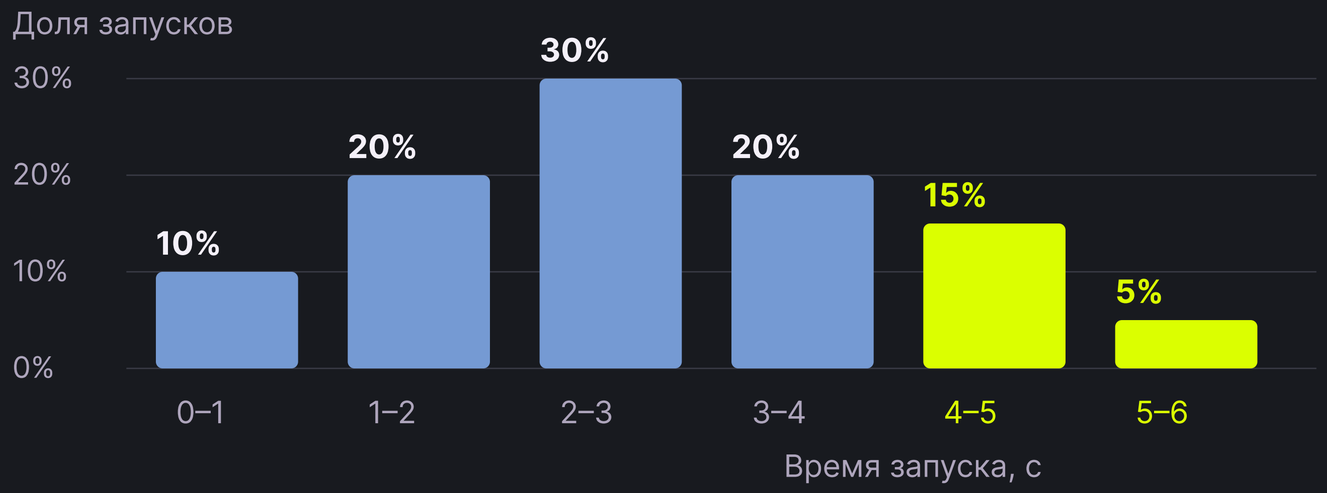
Ищем причину долгого запуска до главной

Метрика
Время запуска

Тип запуска
Холодный

Первый экран
Главная

Распределение времени запуска

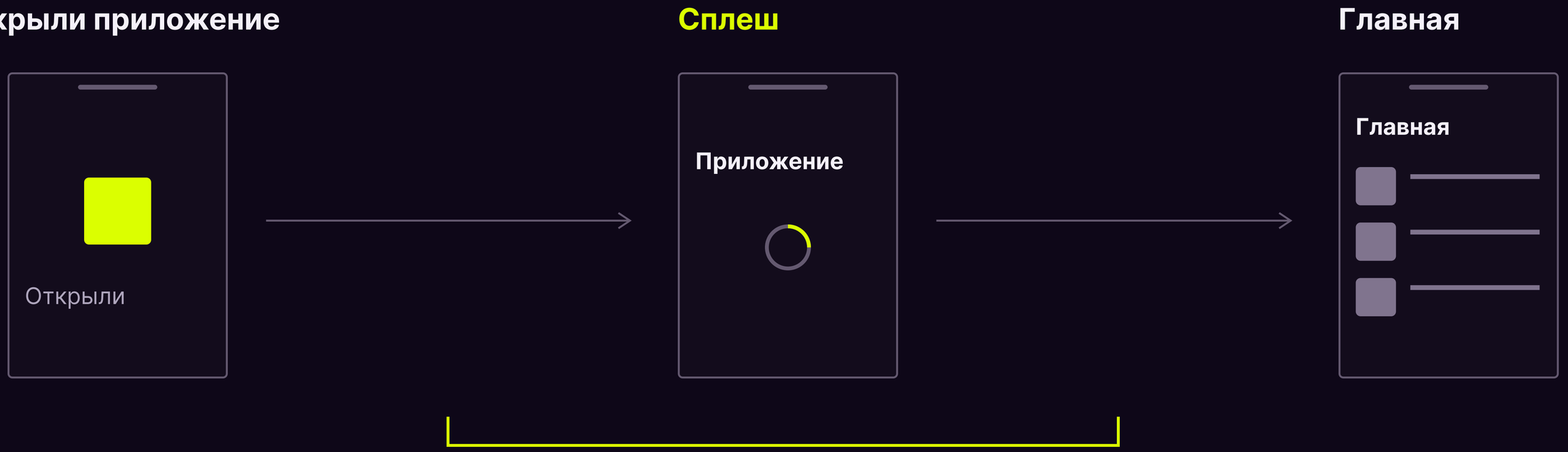


20% запусков

занимают 4–6 секунд

Большую часть ожидания занимал сплеш

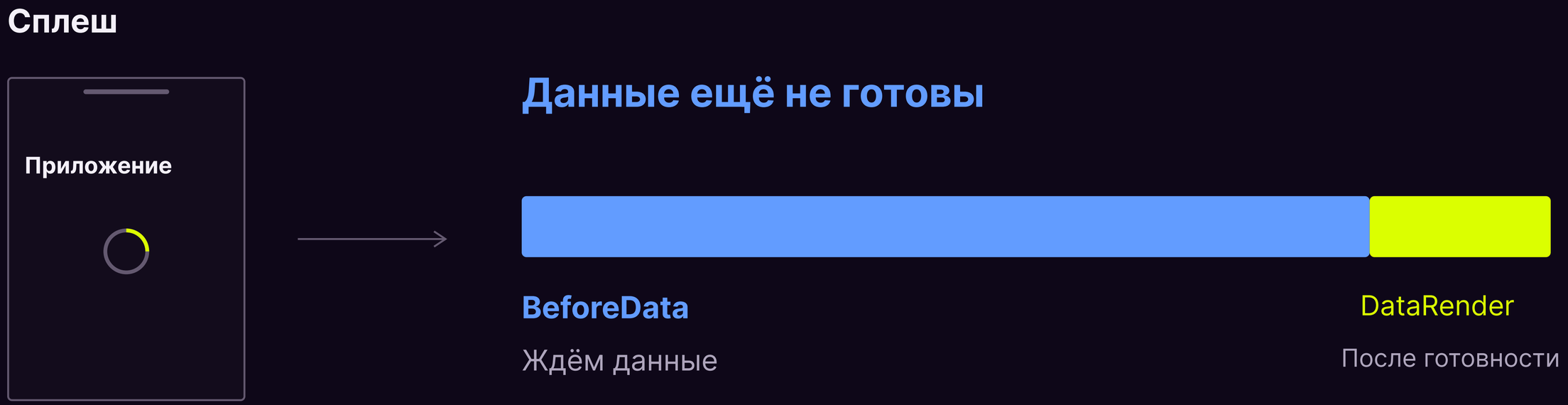
Холодный запуск → главная



Пользователь ещё не может перейти к товарам

Сплеш ждал готовности данных

Разобрали загрузку самого сплеша



Сплеш ждал конфигурацию сервиса A/B-тестов

Что задерживало сплеш

Запрос конфигурации

Ждали соединение, а не обработку запроса

Что задерживало сплеш

Запрос конфигурации

Разобрали запрос по этапам

Сервис обрабатывал запрос быстро. Время теряли до обработки.

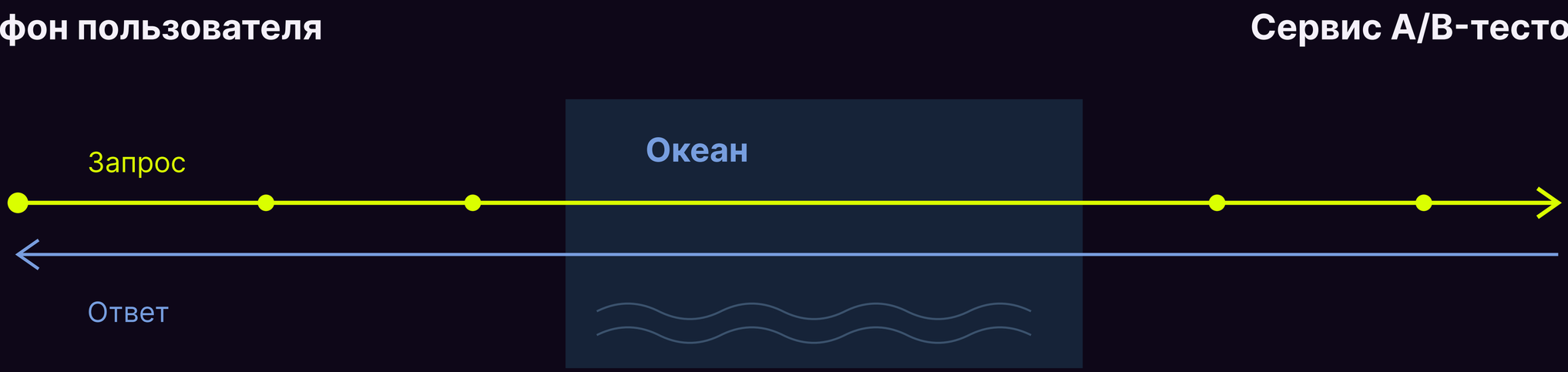
Запрос конфигурации

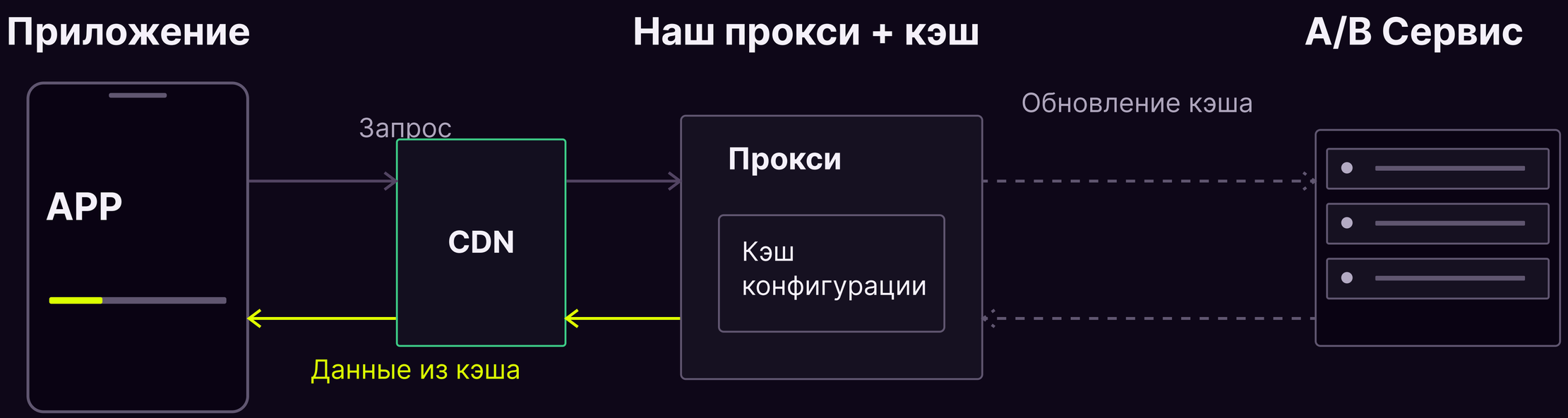


DNS и соединение

Обработка на сервере

Получение ответа


Телефон пользователя
Сервис A/B-тестов
Океан
Запрос
Ответ


Приложение
Наш прокси + кэш
A/B Сервис
APP
Запрос
CDN
Данные из кэша
Прокси
Кэш конфигурации
Обновление кэша

Ускорение запуска потребовало нескольких команд

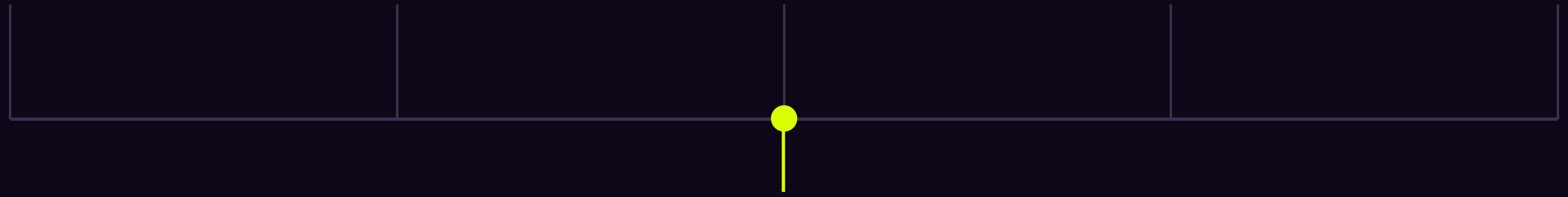
Mobile
Нашли задержку в запуске

Backend
Сделали прокси и кэш

SRE
Добавили CDN и настроили маршрут

QA
Проверили сценарий запуска

Analytics
Проверили эффект на пользователях



На 1,5–2 секунды быстрее

Главная открывалась, но скролл тормозил

На слабых устройствах

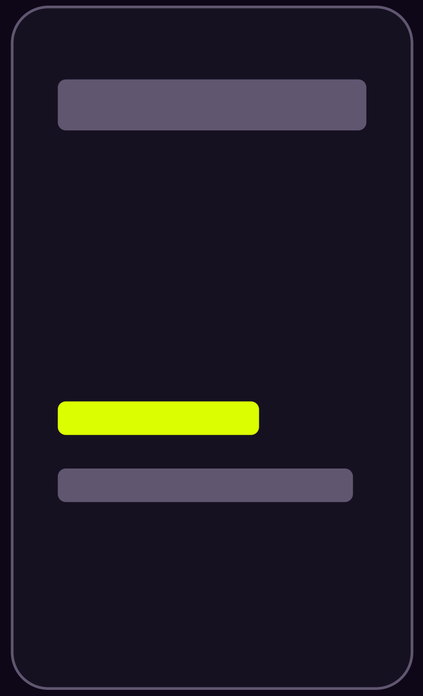
20%

Hitch Rate

≈12%

сессий с Frozen Frames

Теперь нужно повторить этот скролл на тестовом устройстве



На каком телефоне вы проверяете свою фичу?

Телефон для проверки выбрали по данным

Распределение по классам



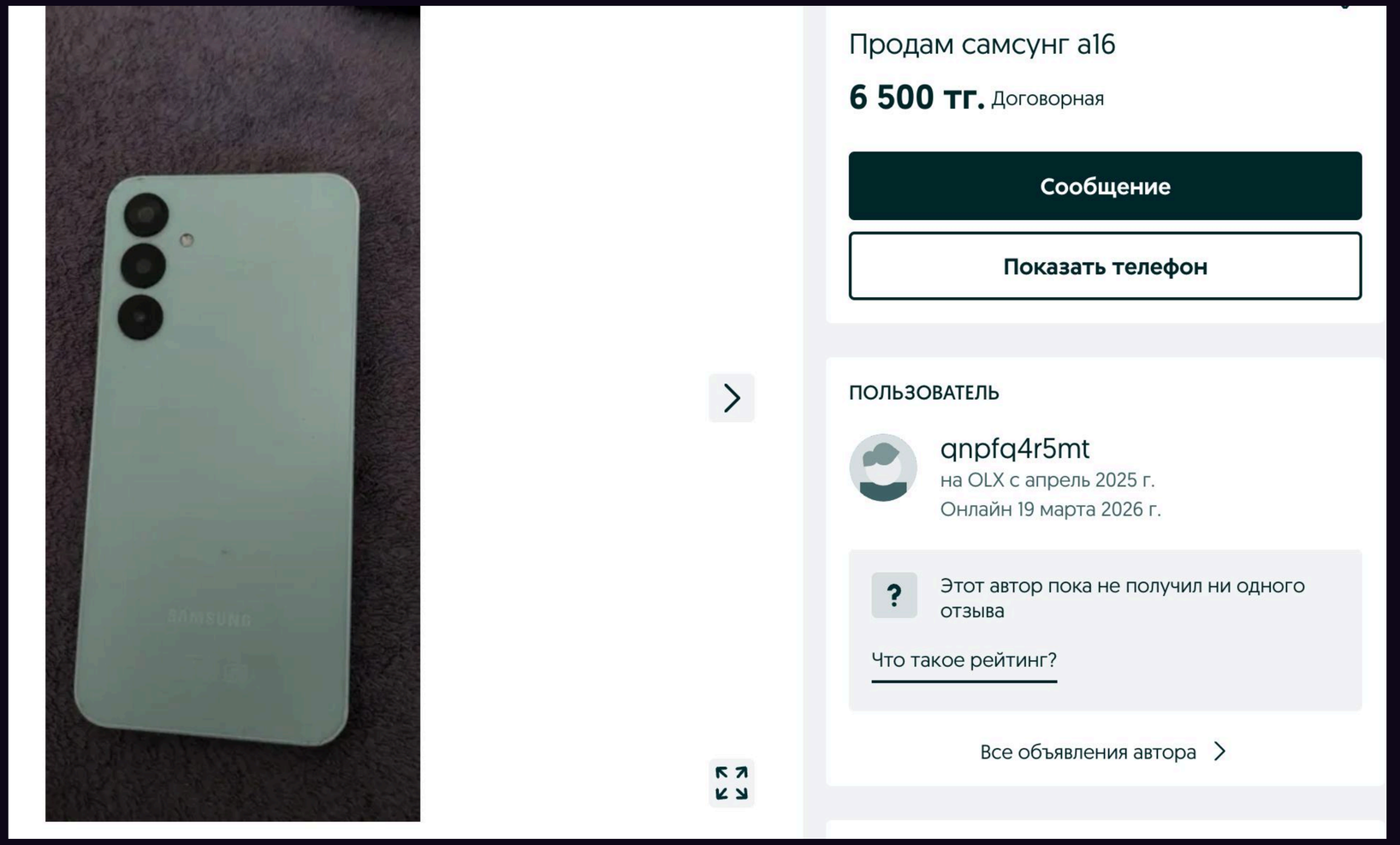
Low	**45%**
Mid	35%
High	20%

→

В классе Low чаще всего

Samsung Galaxy A16

Тестовый телефон можно купить недорого



Samsung Galaxy A16

≈ 1 000 ₽

Теперь мы ближе к реальному пользователю

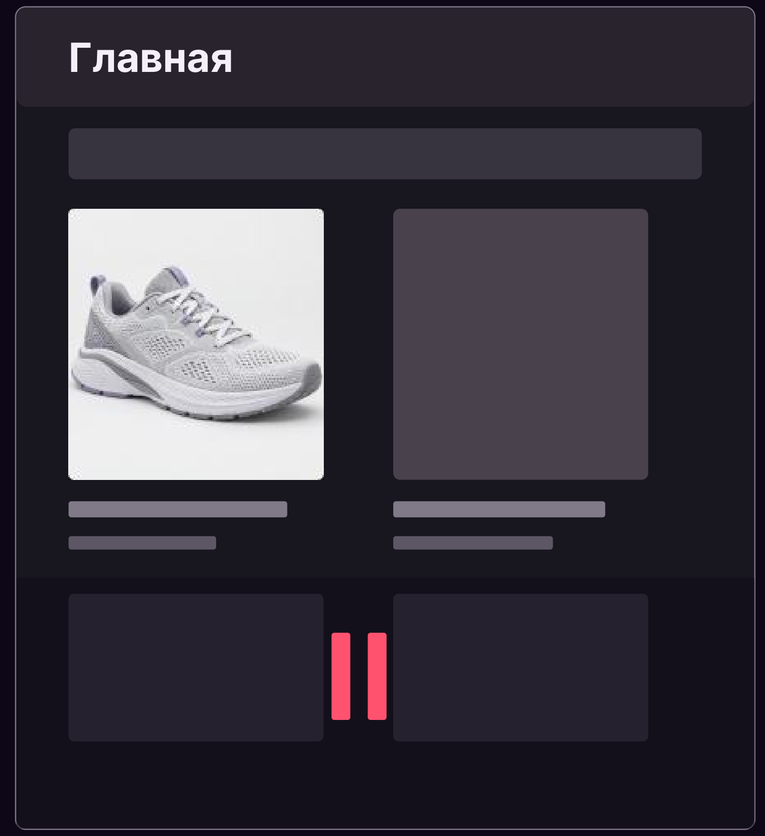
Пример объявления на OLX ↗

Слишком большая картинка для маленькой карточки

Загружали



Деталей больше, чем видно на экране



Рывок при скролле

Качество картинки подстраивается под условия

Класс устройства
low · mid · high

Условия сети
POOR · CONSTRAINED · GOOD

Размер на экране
размер контейнера

Выбираем один вариант



Облегчённая версия



Средняя детализация



Больше деталей

Как проверяли, что изменения помогли

До релиза

Macrobenchmark

Две сборки
Один телефон и одинаковый скролл

Есть ли положительная динамика?

После релиза

Наш SDK

Телефоны пользователей
Реальные открытия и прокрутки

Как изменился опыт пользователей?

Что делает Macrobenchmark

ОТДЕЛЬНЫЙ ТЕСТ

Запуск APK

→

UI AUTOMATOR

Открывает экран

Нажимает и прокручивает

→

МЕТРИКИ

Измеряет результат

Кадры и рывки:
FrameTimingMetric

Macrobenchmark: проверяем правки до релиза



setupBlock: открываем экран и ждём данные

measureBlock: скроллим

Упрощённый пример

```
measureRepeated(
    metrics = listOf(FrameTimingMetric()),
    setupBlock = { openMainAndWaitForData() },
    measureBlock = { scrollMain() },
)
```

СРАВНИВАЕМ

FrameTimingMetric

До правки → После правки

Тот же сценарий и устройство

Отдельно повторили сценарий для Baseline Profile

Та же библиотека, другой тест: BaselineProfileRule

Тот же сценарий главной

Открыть главную

Дождаться товаров

Прокрутить список

→

baseline-prof.txt

Методы и классы, которые выполнились в этом сценарии

→

Добавляем в сборку

Android использует профиль для предварительной компиляции кода

ART

Как проверить вклад Baseline Profile

MacrobenchmarkRule · тот же сценарий · то же устройство

ПРОГОН 1

Без профиля



ПРОГОН 2

С профилем

Сравниваем время запуска и плавность кадров

Итог всех изменений на главной

Hitch Rate

До 20%



После 5%



Сессии с Frozen Frames

До ≈12%



После 4%



Android Performance Skill

Запуск · TTID / TTFD

Main thread · корутины

ANR · блокировки · Binder

Compose · стабильность и рекомпозиции

Lazy-списки · вложенность

Анимации

Состояние · critical path

Память · GC · изображения

База данных · сеть · кэш

Фоновая работа · батарея

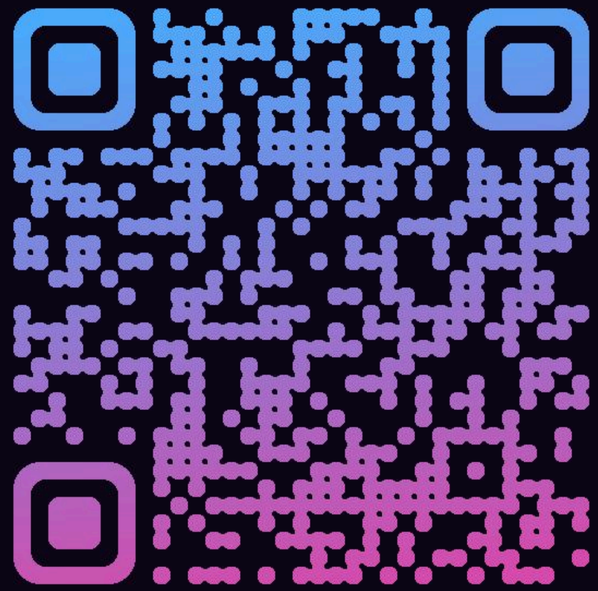
R8 · Baseline / Startup Profiles

Views · GPU · WebView

Kotlin · Perfetto

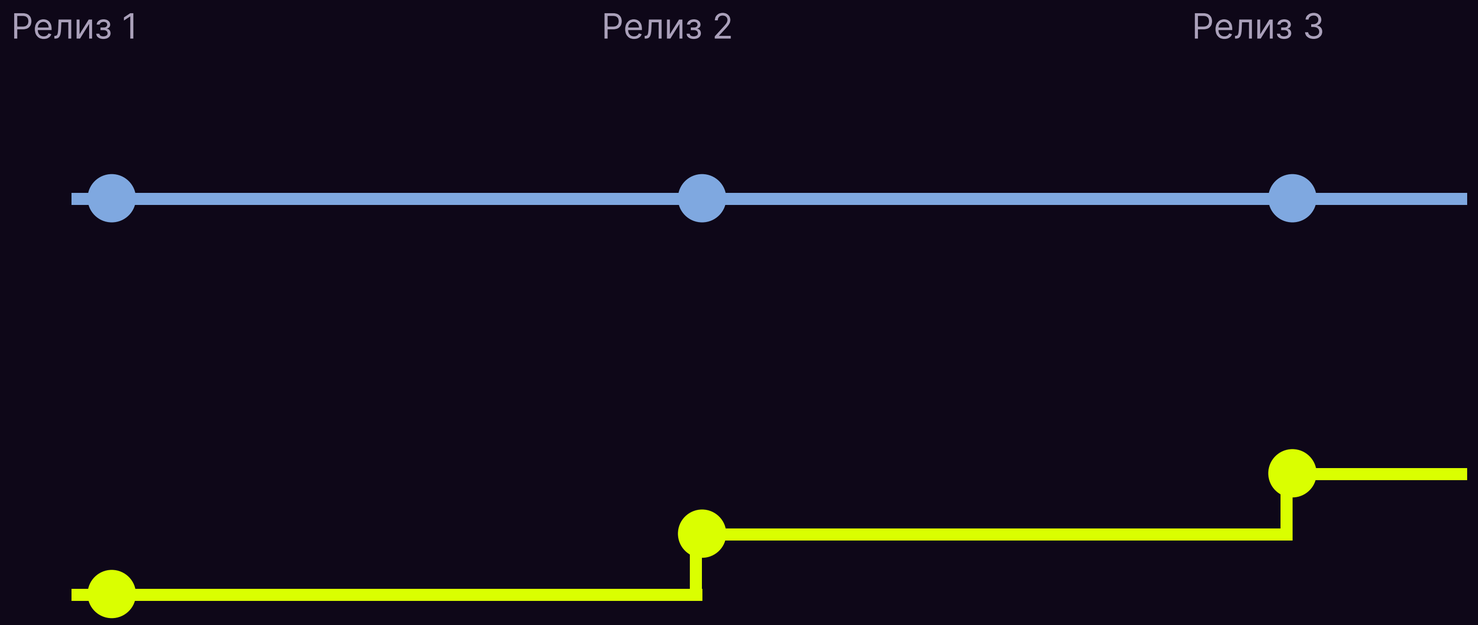
Трейсинг · бенчмарки

Claude Code · Codex · Cursor



timkabot/
android-performance-skill

Система непрерывных улучшений



Сохраняем эффект

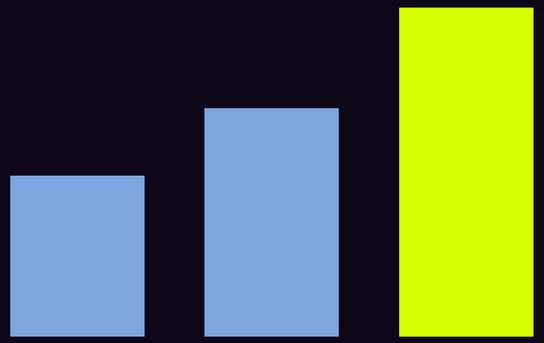
Новые фичи не возвращают прежние тормоза

Улучшаем дальше

Берём следующий проблемный экран

Сам по себе отчёт ничего не исправит

Отчёт есть



Увидели ухудшение

→

Нужно договориться

Что считаем проблемой?

Кто отвечает?

Что делаем дальше?

Кто выбирает цели, а кто предлагает улучшения

Бизнес и лиды

Важные сценарии · границы качества · приоритеты



Чего хотим достичь

Общие цели качества

Кто выбирает цели, а кто предлагает улучшения

Бизнес и лиды

Важные сценарии · границы качества · приоритеты



Чего хотим достичь

Общие цели качества



Как можем улучшить

Предложения команд
и полученный результат

Команды экранов

Причины проблем · изменения · проверка результата

За что отвечает владелец процесса

Бизнес и лиды

Важные сценарии · границы качества · приоритеты

Чего хотим достичь

Общие цели качества

Как можем улучшить

Предложения команд
и полученный результат

Команды экранов

Причины проблем · изменения · проверка результата

Владелец процесса

Помогает договориться
и довести до результата

SLO: о каком качестве мы договорились

Service Level Objective

Не менее 9 из 10 загрузок - до 1 секунды



Как определяем срочность по SLO

Service Level Objective · пример: Checkout, слабые устройства, 14 дней

Не менее 9 из 10 загрузок — до 1 секунды



Хорошо

p90 ≤ 1 с

Нужно внимание

1 с < p90 ≤ 2 с

Критично

p90 > 2 с

Снижаем время загрузки по шагам



С командой выбираем посильное улучшение.

Отчёт → **SLO** → Бэклог → Работа команды → Проверка

Бизнес любит светофоры

Экран	Загрузка	Сессии с Frozen Frames
Главная	🟡 1,7 с	🟢 4,2%
Каталог	🟢 0,9 с	🟢 3,1%
Поиск	🔴 2,6 с	🟡 8,0%
Мои заказы	🟡 1,3 с	🟢 2,4%
Оформление заказа	🔴 2,4 с	🟡 6,7%

Отчёт посмотрели. Задачу не взяли.



Кадр: «Нарко» · Netflix, Imgflip

Процесс согласовали до запуска

- **Руководители всех функций**
- **Тимлиды всех команд**

→

Бэклог команды

Улучшение производительности

Из отчёта по релизу

Понятно, откуда задача и зачем она появилась

Из отчёта создаём задачи командам

Отчёт по релизу

Экран, метрика
и условия



Бот или
владелец процесса

Jira

Задача ответственной команде

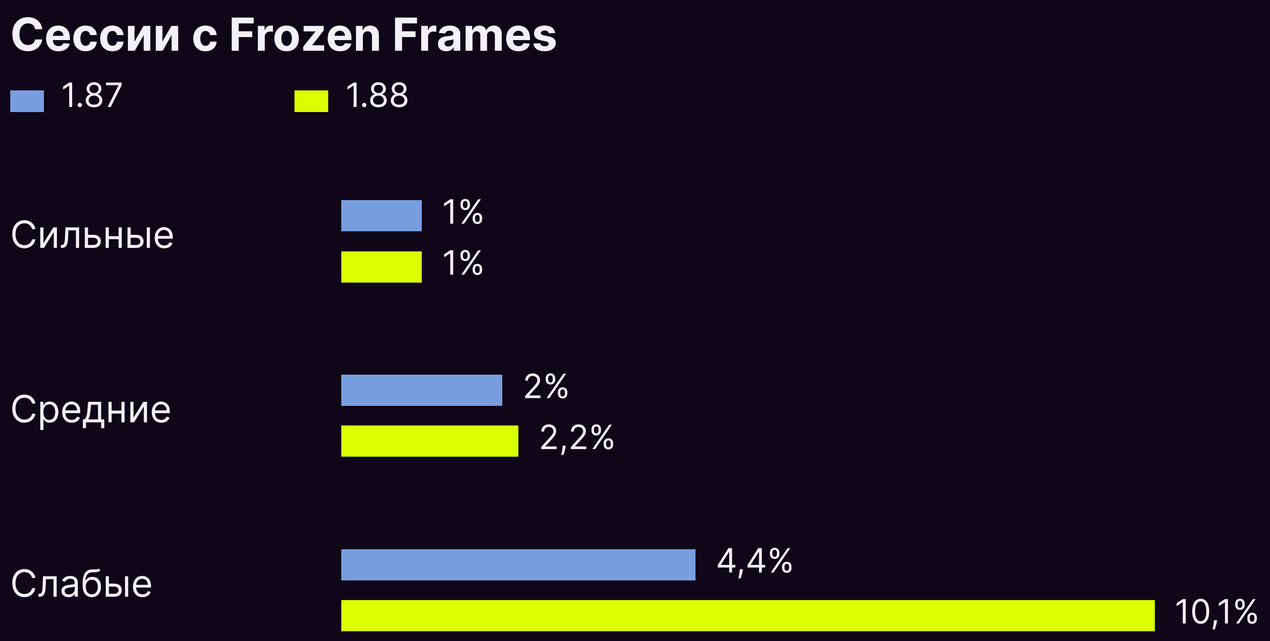
Приоритет по SLO

Критичное берём в ближайший релиз

Зону внимания разбираем на грумминге

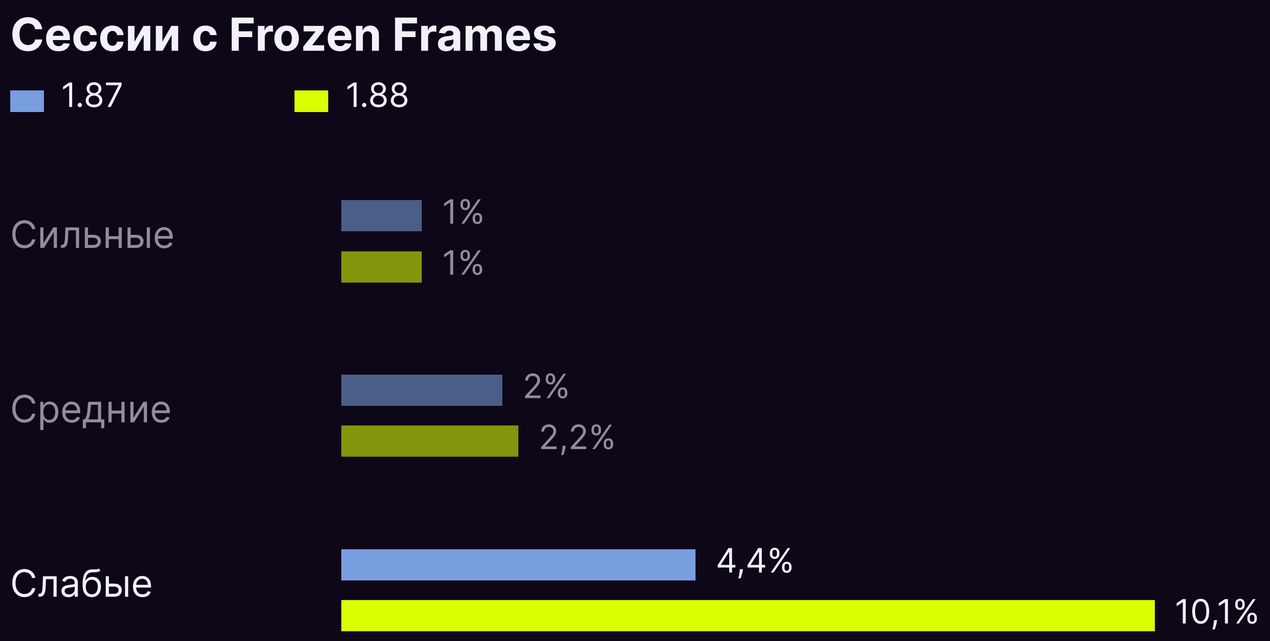
Что изменилось после релиза?

Пример: Экран чекаута



Что изменилось после релиза?

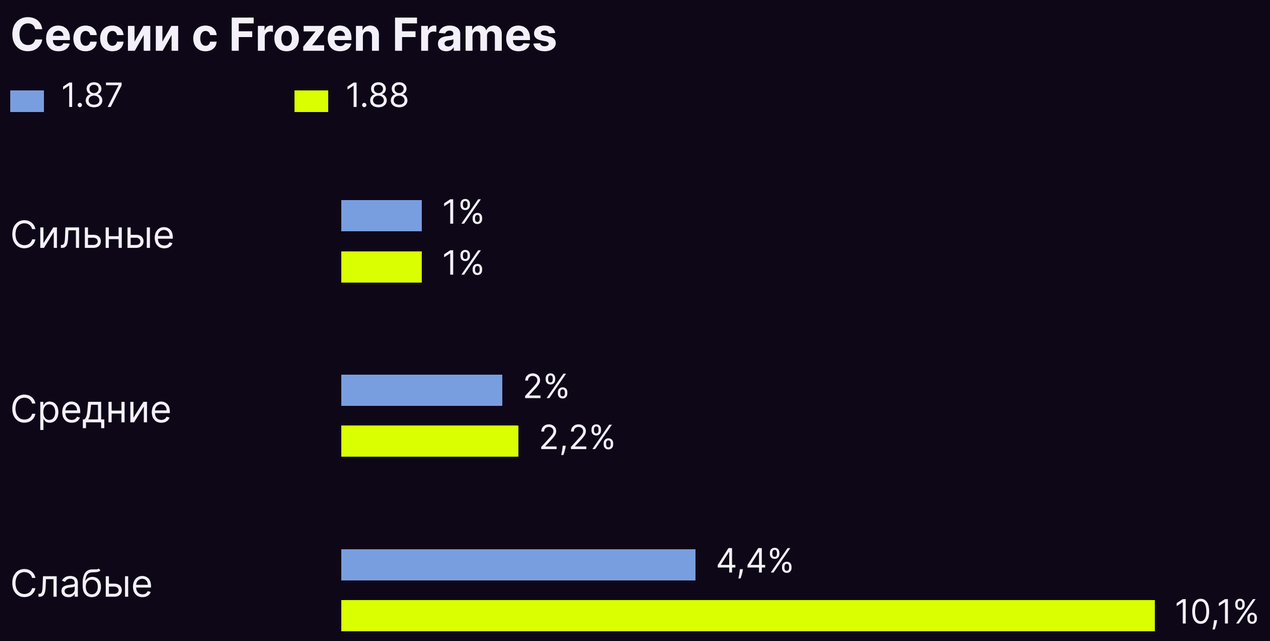
Пример: Экран чекаута



4,4% → 10,1%

Что изменилось после релиза?

Пример: Экран чекаута



По SLO: зона внимания

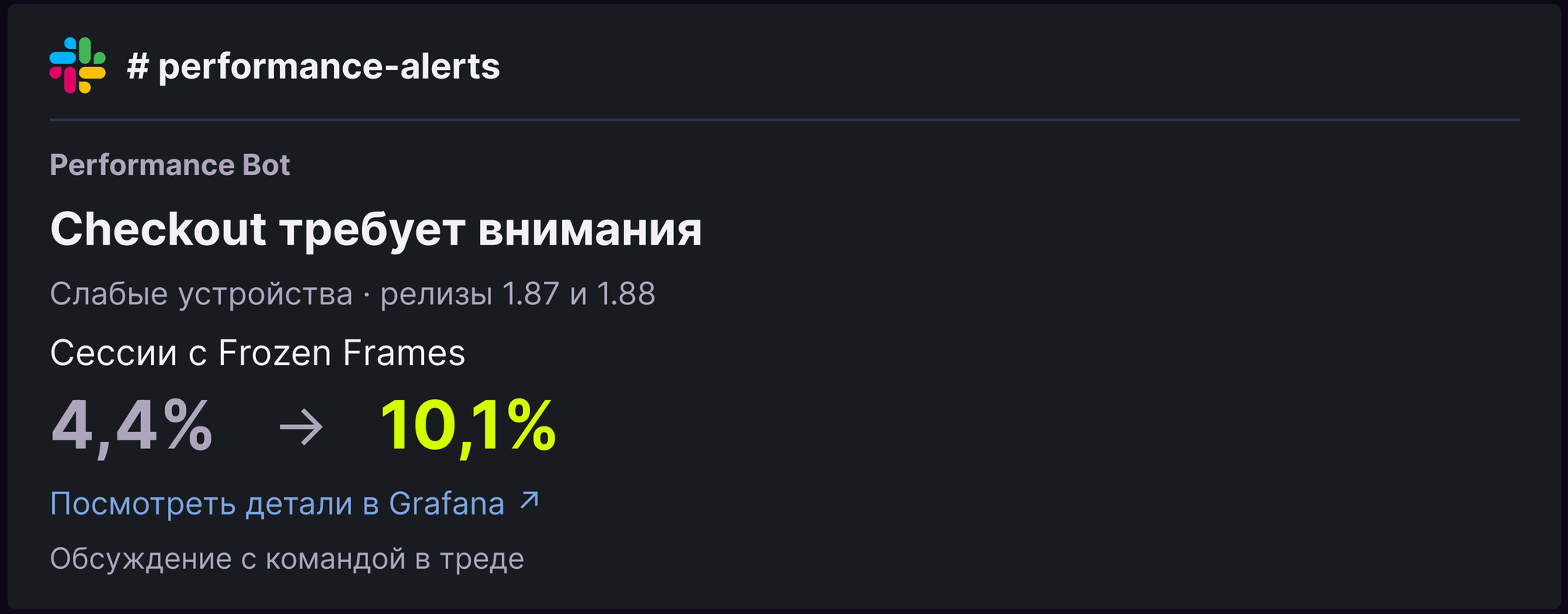
Цель <5% · внимание 5–11%

4,4% → 10,1%

Передаём команде:
экран, версию и устройства

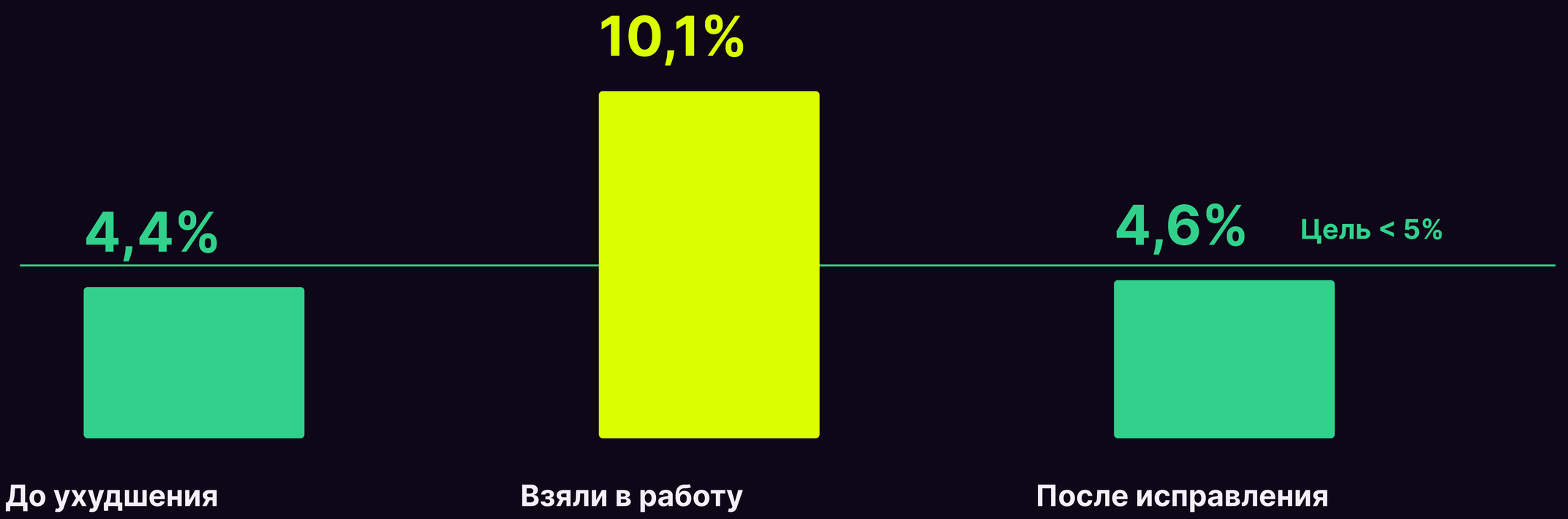
Из отчёта появляется конкретная задача для команды Checkout

Отчёт приходит командам в Slack



Следующий отчёт показывает результат

Пример: Экран чекаута



Отчёт → SLO → Бэклог → Работа команды → **Проверка**

Теперь соберём весь цикл

БОТ · В ДЕНЬ ФРИЗА

Готовит отчёт и задачи

Данные за релиз
Сравнение с SLO
Slack + задачи в Jira

Команда доводит задачу до релиза

БОТ · В ДЕНЬ ФРИЗА

Готовит отчёт и задачи

Данные за 14 дней
Сравнение с SLO
Slack + задачи в Jira



КОМАНДА

Выпускает исправление

Выбирает задачу
Находит причину
Проверяет решение

Следующий отчёт проверяет результат

БОТ · В ДЕНЬ ФРИЗА

Готовит отчёт и задачи

Данные за 14 дней
Сравнение с SLO
Slack + задачи в Jira

→

КОМАНДА

Выпускает исправление

Выбирает задачу
Находит причину
Проверяет решение

→

СЛЕДУЮЩИЙ ОТЧЁТ

Проверяет результат

Та же метрика
в тех же условиях

10,1% → 4,6%

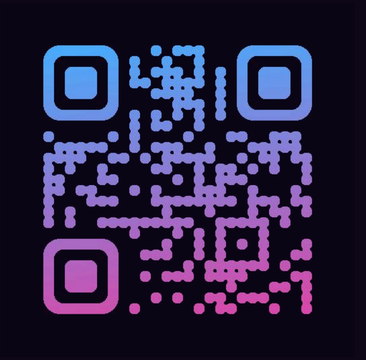
Checkout · Low
Сессии с Frozen Frames

Новые данные → следующий приоритет

Что мешает взять задачу в работу?

Убрать зависания Checkout

Задача уже в бэклоге



6 слоёв сопротивления
Статья на vc.ru

«А есть польза?»

Проверяем эффект

«Не знаем, с чего начать»

Вместе разбираем проблему

«Весь релиз занят»

Согласуем план

Помогаем команде включиться в цикл

Первый раз

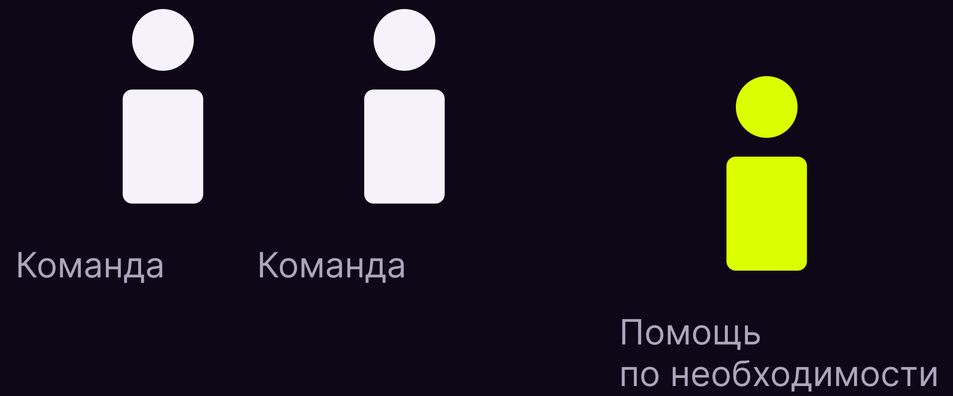


Владелец процесса

Команда

Разбираются и делают вместе

Следующий цикл



Команда

Команда

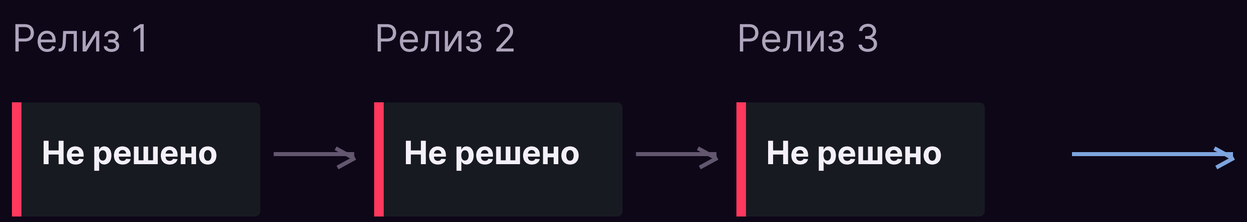
Помощь по необходимости

Команда проходит цикл сама

Отчёт → SLO → Бэклог → **Работа команды** → Проверка

Если задача застряла, подключаем кластер-лида

Критичный экран · несколько релизов без прогресса



Кластер-лид + владелец процесса

Приоритеты

Ресурсы

Зависимости

Согласованный план

или принятый риск

Отчёт → SLO → Бэклог → **Работа команды** → Проверка

Что поменялось в нашей работе?

Раньше		Теперь
Начинали с жалобы	→	Видим, где и у кого медленно

Что поменялось в нашей работе?

Раньше		Теперь
Начинали с жалобы	→	Видим, где и у кого медленно
Трудно выбрать, что исправлять	→	SLO задают приоритет задач

Что поменялось в нашей работе?

Раньше		Теперь
Начинали с жалобы	→	Видим, где и у кого медленно
Трудно выбрать, что исправлять	→	SLO задают приоритет задач
Исправляли отдельную проблему	→	У экрана есть команда и цель

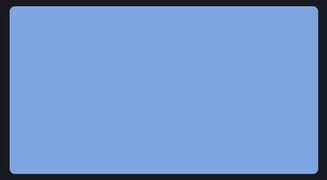
Что поменялось в нашей работе?

Раньше		Теперь
Начинали с жалобы	→	Видим, где и у кого медленно
Трудно выбрать, что исправлять	→	SLO задают приоритет задач
Исправляли отдельную проблему	→	У экрана есть команда и цель
Ждали следующей жалобы	→	Проверяем результат и идём дальше

С чего начать у себя?

Начните с одного важного экрана

Важный экран



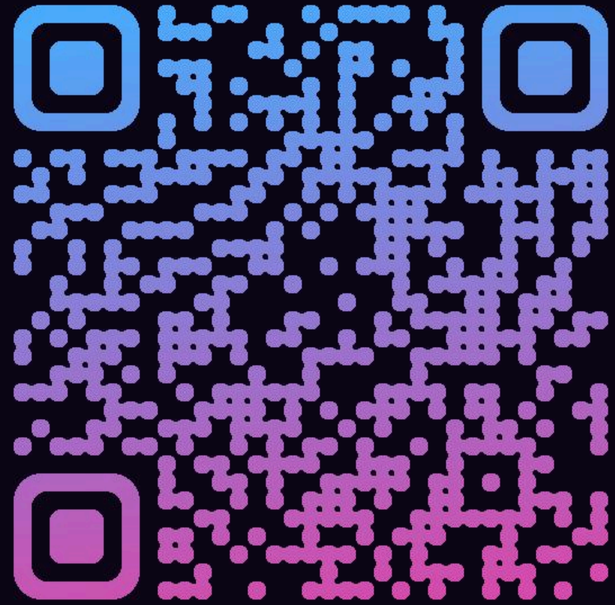
Команда
Цель качества



Performance SDK и пример приложения

Забирайте и подключайте

Библиотека и пример приложения на GitHub



github.com/timkabot/performance-sdk

Будем на связи



Тимур Боргалинов

Lead Android Dev · Uzum (2022 - present)

Co-Founder · IWBL (2025-present)

Ex-CEO · Myte.me (2021 - 2023)
Ex Senior Android Dev · KazanExpress (2021-2022)
Ex Senior Kotlin Dev · X5 Tech (2019-2021)

Контакты

LinkedIn

`linkedin.com/in/timkabor`

Telegram

`@timkabor`

