



Автоматизация проверок логирования действий пользователей сайта

Пранова Елена



Спикер

Одноклассники (VK)

Автоматизирую под web,
mobile web, API и Android.

Участвую в
образовательных проектах.
Выступаю на конференциях.



Пранова
Елена





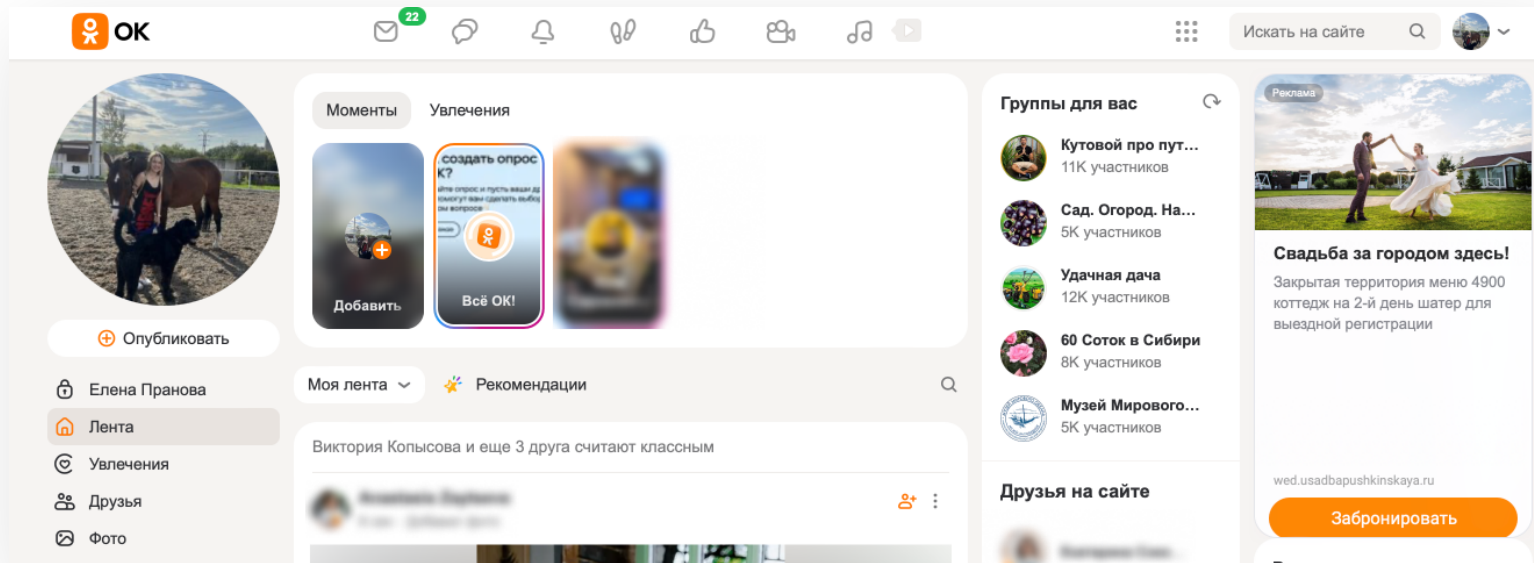
Вводная часть

Проблематика

OK — социальная сеть



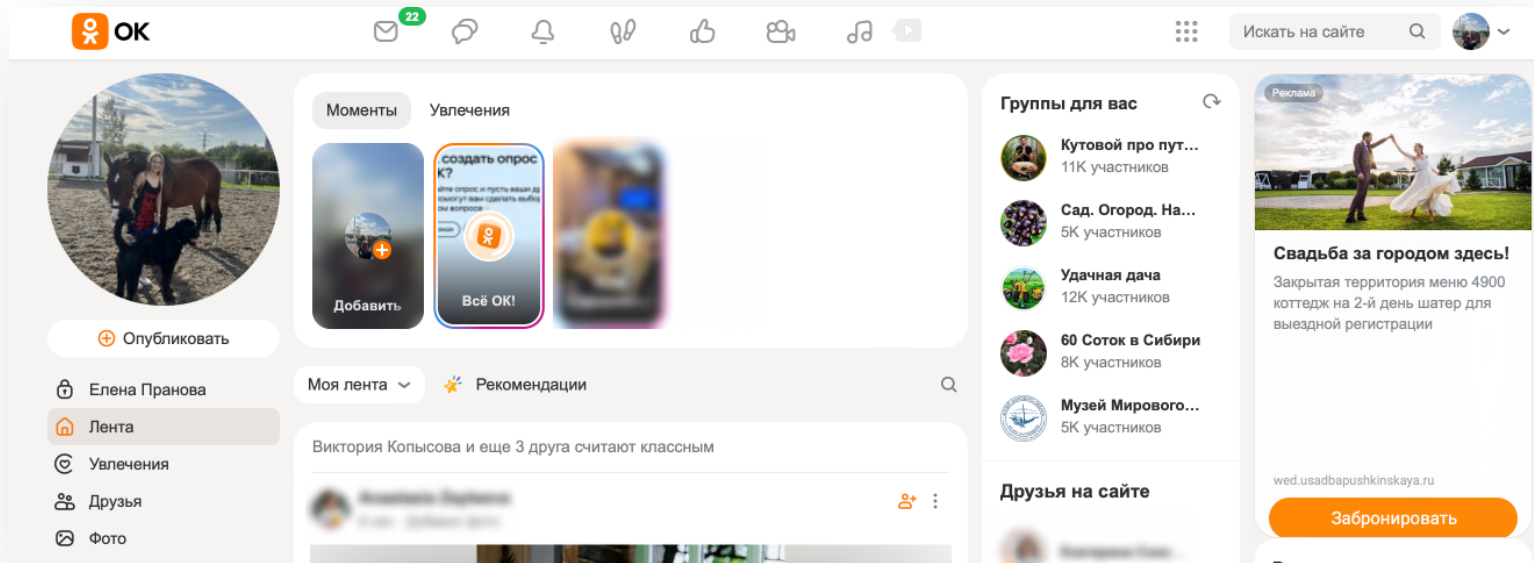
Кликаем «Искать на сайте»



OK — социальная сеть



Кликаем «Искать на сайте»



Пришло событие:
клик на кнопку
«Искать на сайте»

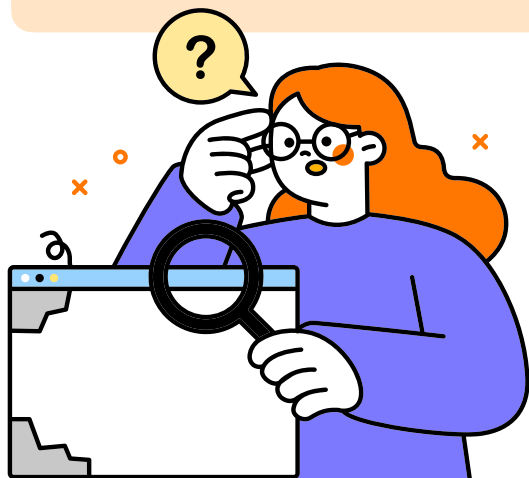


Аналитик

Зачем нужна статистика?



Нужно поменять текст
кнопки «Войти»
на «Залогиниться»

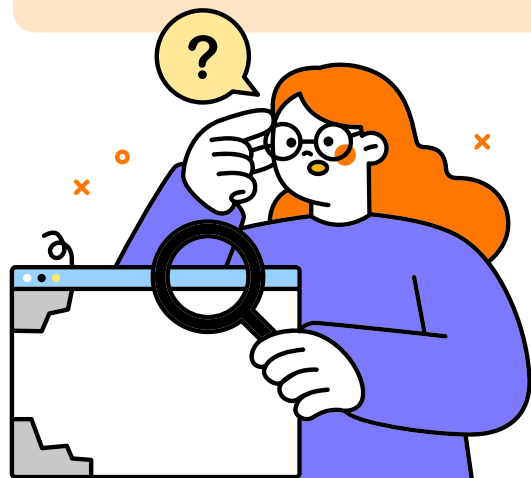


Менеджер

Зачем нужна статистика?



Нужно поменять текст
кнопки «Войти»
на «Залогиниться»



Менеджер



Продуктовый эксперимент

Зачем нужна статистика?



Настало время...



Зачем нужна аналитика?



→ Мониторинг работоспособности сервисов, хостов и продукта

Зачем нужна аналитика?



- Мониторинг работоспособности сервисов, хостов и продукта
- Мониторинг проводимых экспериментов

Зачем нужна аналитика?



- Мониторинг работоспособности сервисов, хостов и продукта
- Мониторинг проводимых экспериментов
- Первичный анализ успешности запуска продуктовых и технических экспериментов

Зачем нужна аналитика?



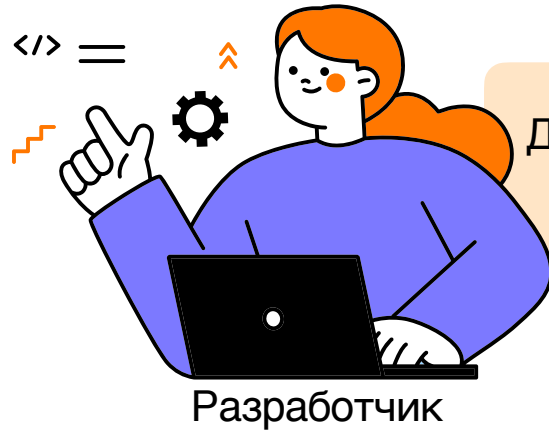
- Мониторинг работоспособности сервисов, хостов и продукта
- Мониторинг проводимых экспериментов
- Первичный анализ успешности запуска продуктовых и технических экспериментов
- Для проведения аналитики по продукту с детализацией по пользователям командой аналитики

Зачем нужна аналитика?



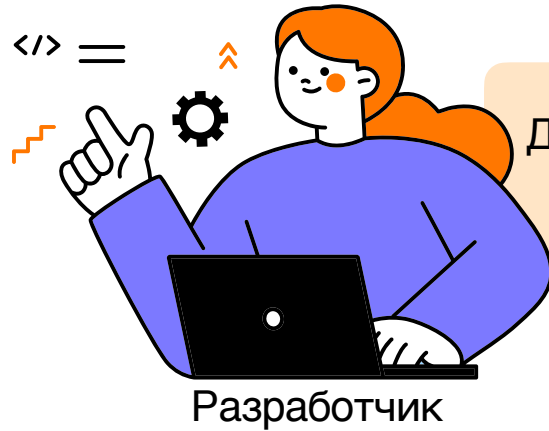
- Мониторинг работоспособности сервисов, хостов и продукта
- Мониторинг проводимых экспериментов
- Первичный анализ успешности запуска продуктовых и технических экспериментов
- Для проведения аналитики по продукту с детализацией по пользователям командой аналитики
- Для проведения A/B-экспериментов с анализом метрик пользователей и влияния запусков на продукт

Как она появляется



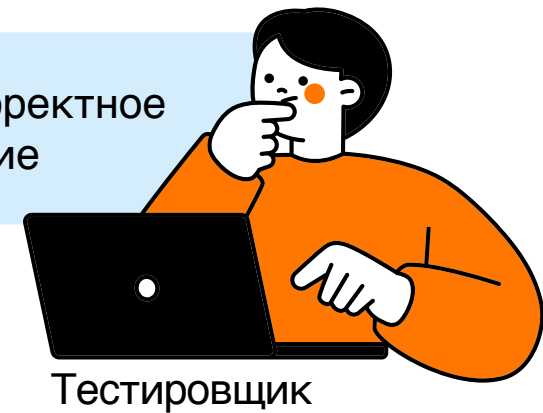
Добавляет в код заготовленные
параметры для статистики

Как она появляется



Добавляет в код заготовленные
параметры для статистики

Проверяет их корректное
отображение



С чем столкнулись



→ QA много времени тратили на тестирование статистики

С чем столкнулись



- QA много времени тратили на тестирование статистики
- Легко ошибиться

С чем столкнулись



- QA много времени тратили на тестирование статистики
- Легко ошибиться
- Много табличных данных

С чем столкнулись



- QA много времени тратили на тестирование статистики
- Легко ошибиться
- Много табличных данных
- Легко пропустить


С чем столкнулись



- QA много времени тратили на тестирование статистики
- Легко ошибиться
- Много табличных данных
- Легко пропустить
- Рутина

С чем столкнулись



- QA много времени тратили на тестирование статистики
- Легко ошибиться
- Много табличных данных
- Легко пропустить
- Рутина
- Выглядит идеально для автоматизации 

План



→ Как устроена статистика

План



→ Как устроена статистика

→ Какие есть проблемы с тестированием статистики

План



- Как устроена статистика
- Какие есть проблемы с тестированием статистики
- Что нужно для автоматизации

План



- Как устроена статистика
- Какие есть проблемы с тестированием статистики
- Что нужно для автоматизации
- Какие есть подходы к автоматизации статистики

План



- Как устроена статистика
- Какие есть проблемы с тестированием статистики
- Что нужно для автоматизации
- Какие есть подходы к автоматизации статистики
- Что выбрали мы

План



- Как устроена статистика
- Какие есть проблемы с тестированием статистики
- Что нужно для автоматизации
- Какие есть подходы к автоматизации статистики
- Что выбрали мы
- Выводы



Как устроена статистика

Статистика в ОК

В процессе участвуют:

Сервисы/приложения



Статистика в ОК



В процессе участвуют:

Сервисы/приложения



Механизм отправки

Статистика в ОК



В процессе участвуют:

Сервисы/приложения



Механизм отправки



Сервис-брокер

Статистика в ОК



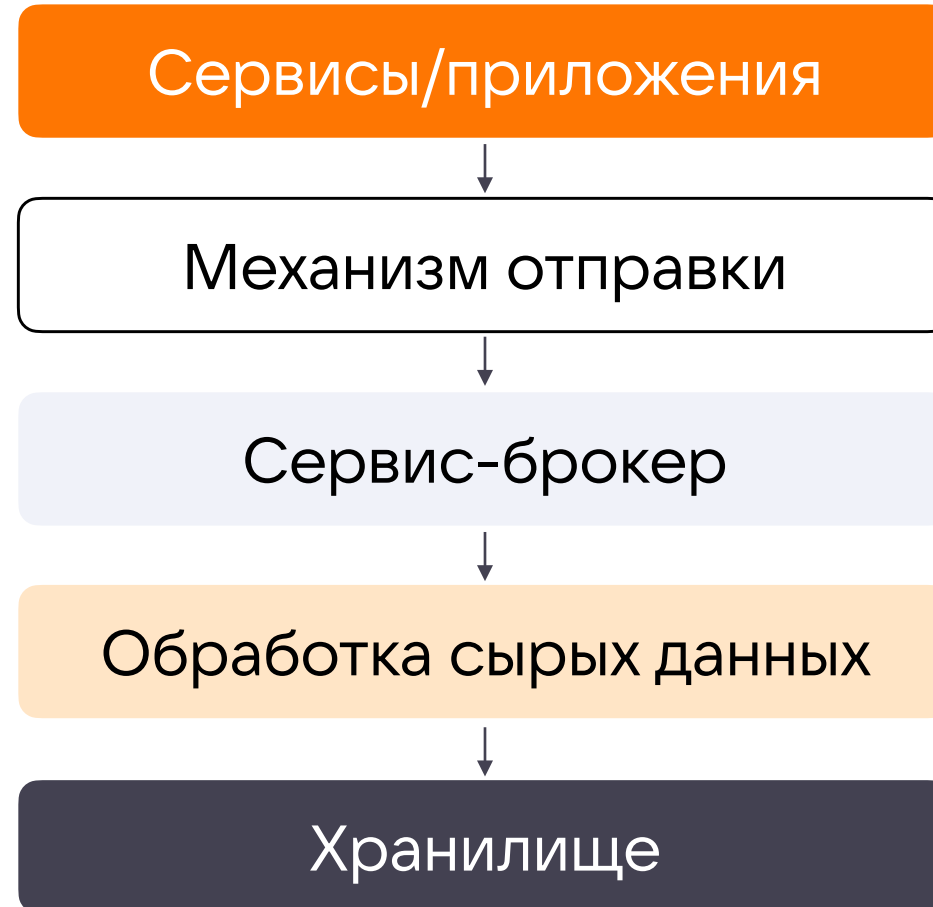
В процессе участвуют:



Статистика в ОК



В процессе участвуют:



Как и что смотрели



→ Ручное тестирование

Как и что смотрели



- Ручное тестирование
- Тестировали самые важные проекты (например, авторизация)
 - + важные продуктовые запуски

Как и что смотрели



- Ручное тестирование
- Тестировали самые важные проекты (например, авторизация)
 - + важные продуктовые запуски
- Ретест при первоначальном тестировании статистики



Проблемы

Статистика



→ Сбор данных

Статистика



→ Сбор данных

→ Их последующее отправление и обработка

Статистика



- Сбор данных
- Их последующее отправление и обработка
- Количество времени

Статистика



- Сбор данных
- Их последующее отправление и обработка
- Количество времени
- Доступы



Кто ждет данные примерно сутки?

Время



→ Данные доступны не сразу

Время



- Данные доступны не сразу
- В основном в течение суток появляются данные с продакшена

Время



- Данные доступны не сразу
- В основном в течение суток появляются данные с продакшена

Сервис статистики — это отдельный сервис, поэтому агрегация продакшен статистики будет занимать определенное время

Время



- Данные доступны не сразу
- В основном в течение суток появляются данные с продакшена

Сервис статистики — это отдельный сервис, поэтому агрегация продакшен статистики будет занимать определенное время

А как получить быструю тестовую статистику или есть другие варианты?

Проблемы с доступом



Неочевидные кейсы

Дублирование



→ Неочевидно, когда, как и сколько смотреть?

Дублирование



- Неочевидно, когда, как и сколько смотреть?
- Очень плохо для статистики

Дублирование



- Неочевидно, когда, как и сколько смотреть?
- Очень плохо для статистики

У вас может быть прирост 2х, вы думаете, что у вас фича выстрелила, а по факту там стандартные показатели

Неоднозначное поведение



- Если на одном экране кнопка работает, отдает правильную статистику, то это не значит, что на другом экране такая же кнопка будет иметь такое же поведение

Неоднозначное поведение



- Если на одном экране кнопка работает, отдает правильную статистику, то это не значит, что на другом экране такая же кнопка будет иметь такое же поведение
- То есть надо тестировать все и везде. Все кейсы нужно проверять

Неправильно на беке



→ Вы делаете все действия правильно, а данные приходят не те

Неправильно на беке



- Вы делаете все действия правильно, а данные приходят не те
- Актуально, когда у нас есть цепочка действий.
И от каждого звена зависит какой результат получится в итоге.
А на беке сделали неправильную конфигурацию

Продуктовые эксперименты



→ Важные продуктовые эксперименты, которые пришлось откатывать

Продуктовые эксперименты



- Важные продуктовые эксперименты, которые пришлось откатывать
- Если статистика пишется неправильно, то все дни, потраченные на эксперимент впустую, нужно перезаписывать

58



Необходимое и достаточное

Тесты



→ Кейсы

→ Автоматизированные тесты

Тесты



→ Кейсы

→ Автоматизированные тесты



Структура



Автотесты

Архитектура автотестов



- Все, что вводится каждый раз, должно быть сгенерировано, иначе можно пропустить баг

Архитектура автотестов



- Все, что вводится каждый раз, должно быть сгенерировано, иначе можно пропустить баг
- Боты = автоматическое создание пользователей

Архитектура автотестов



- Все, что вводится каждый раз, должно быть сгенерировано, иначе можно пропустить баг
- Боты = автоматическое создание пользователей
- Генерировать данные на вход

Архитектура автотестов



- Все, что вводится каждый раз, должно быть сгенерировано, иначе можно пропустить баг
- Боты = автоматическое создание пользователей
- Генерировать данные на вход
- PageObject

Структура



Автотесты

+

Архитектура
автотестов

CI



Структура



Автотесты

+






Архитектура
автотестов

+

CI

Отчетность



Тест 1	
Тест 2	
Тест 3	
Тест 4	
Тест 5	

Структура



Автотесты

+

Архитектура
автотестов

+

CI

+

Отчетность

Структура



Автотесты

+

Архитектура
автотестов

+

CI

+

Отчетность

?

Проверка
статистики

Какие до



→ У нас есть архитектура тестов

Все автотесты отражают действия пользователей, мы можем наложить на них проверки логирования

Какие до



- У нас есть архитектура тестов
- Сами тесты на каждую команду

Все автотесты отражают действия пользователей, мы можем наложить на них проверки логирования

Какие до



- У нас есть архитектура тестов
- Сами тесты на каждую команду
- Тест-кейсы

Все автотесты отражают действия пользователей, мы можем наложить на них проверки логирования

Какие до



- У нас есть архитектура тестов
- Сами тесты на каждую команду
- Тест-кейсы
- Сервис регистрации ботов

Все автотесты отражают действия пользователей, мы можем наложить на них проверки логирования

Какие до



- У нас есть архитектура тестов
- Сами тесты на каждую команду
- Тест-кейсы
- Сервис регистрации ботов
- CI/CD

Все автотесты отражают действия пользователей, мы можем наложить на них проверки логирования

Какие до



- У нас есть архитектура тестов
- Сами тесты на каждую команду
- Тест-кейсы
- Сервис регистрации ботов
- CI/CD
- Отчетность

Все автотесты отражают действия пользователей, мы можем наложить на них проверки логирования

Что у нас



→ > 3600 API автотестов = за 5 минут проходят

Что у нас



→ > 3600 API автотестов = за 5 минут проходят

→ > 3000 WEB автотестов = за 17-18 минут проходят

Что у нас



- > 3600 API автотестов = за 5 минут проходят
- > 3000 WEB автотестов = за 17-18 минут проходят
- > 1100 MOB/WEB автотестов = за 10-12 минут проходят

Что у нас



- > 3600 API автотестов = за 5 минут проходят
- > 3000 WEB автотестов = за 17-18 минут проходят
- > 1100 MOB/WEB автотестов = за 10-12 минут проходят

Запуски:

- ночные прогоны

Что у нас



- > 3600 API автотестов = за 5 минут проходят
- > 3000 WEB автотестов = за 17-18 минут проходят
- > 1100 MOB/WEB автотестов = за 10-12 минут проходят

Запуски:

- ночные прогоны
- на апдейтах

Что у нас



- > 3600 API автотестов = за 5 минут проходят
- > 3000 WEB автотестов = за 17-18 минут проходят
- > 1100 MOB/WEB автотестов = за 10-12 минут проходят

Запуски:

- ночные прогоны
- на апдейтах
- планируется на каждый коммит запуск смоуков

Технологии и не только



- Java
- Selenium
- JUnit5
- Система регистрации ботов
- Самописный раннер, тесты гоняются параллельно
- Самописный сервис отчетности
- TeamCity
- PageObject и другие паттерны



Какие есть варианты

Код разработчика



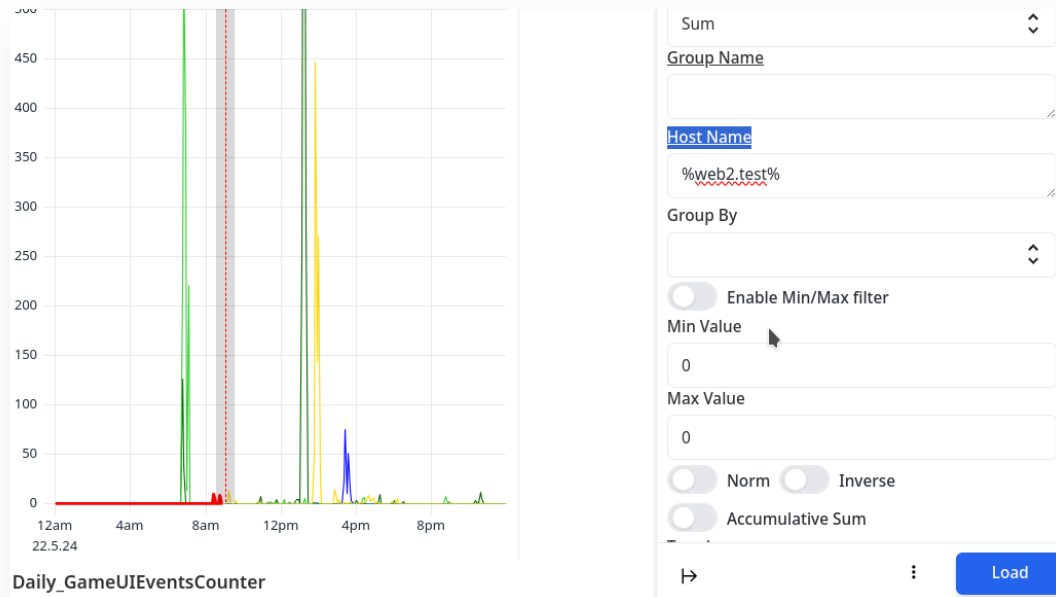
```
45 + String[] parts = str.split(DELIMITER);
46 + final AutoRefillFieldItem result;
47 + switch (parts.length) {
48 +     case 1 -> {
49 +         result = new AutoRefillFieldItem(Integer.parseInt(parts[0]), null);
50 +         LoggerUtil.operationSuccess(PaymentConfiguration.STAT, OPERATION, "parse", "balance-empty");
51 +     }
52 +     case 2 -> {
53 +         result = new AutoRefillFieldItem(Integer.parseInt(parts[0]), Integer.parseInt(parts[1]));
54 +         LoggerUtil.operationSuccess(PaymentConfiguration.STAT, OPERATION, "parse", "balance-exist");
55 +     }
56 +     default -> throw new IllegalArgumentException("Invalid format: " + str);
57 + };
58 + return result;
59 + }
```

Плюсы:
можем получить доступ

Минусы:
трудоемкий процесс,
нет гарантий, что работает

Автоматизация:
не подходит

Графики



Плюсы:
можем получить доступ

Минусы:
сложно отслеживать
каждое событие

Автоматизация:
не подходит

Варианты



1

Консоль браузера



The screenshot shows a browser's developer console with the network tab selected. A request named 'gwtlog' is highlighted. The payload is expanded, showing a JSON object with a 'continue' property containing an array of objects. The first object in the array has several properties including 'start', 'now', 'appId', 'ref', 'focus', 'reload', and 'milestone'. Below the array, a 'statId' is also visible.

gwtlog Invert Hide data URLs Hide extension URLs **All** Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other Blocked response cookies

Blocked requests 3rd-party requests

1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms 8000 ms 9000 ms 10000 ms 11000 ms 12000 ms

Name gwtlog gwtlog

Headers **Payload** Preview Response Initiator Timing Cookies

▼ Form Data view source view URL-encoded

```
a: {"GSW":{"continue":[{"start":1716355623086,"now":1716355743088,"appId":"1251640576","ref":-1,"focus":1,"reload":0,"milestone":12000}]}}
```

statId: 9bd71a74-17ad-4135-b8ad-c50eb37d11df

Консоль браузера



```
// Получаем DevTools  
DevTools devTools = driver.getDevTools();  
devTools.createSession();
```

Консоль браузера



```
// Получаем DevTools  
DevTools devTools = driver.getDevTools();  
devTools.createSession();  
  
// Включаем перехват сетевых запросов  
devTools.send(Network.enable());
```

Консоль браузера



```
// Получаем DevTools
DevTools devTools = driver.getDevTools();
devTools.createSession();

// Включаем перехват сетевых запросов
devTools.send(Network.enable());

// Добавляем слушатель на события ответа
devTools.addListener(Network.responseReceived(), response -> {
    res = response.getResponse();
    url = res.getUrl();
    status = res.getStatus();
});
```

Консоль браузера



```
// Получаем DevTools
DevTools devTools = driver.getDevTools();
devTools.createSession();

// Включаем перехват сетевых запросов
devTools.send(Network.enable());

// Добавляем слушатель на события ответа
devTools.addListener(Network.responseReceived(), response -> {
    res = response.getResponse();
    url = res.getUrl();
    status = res.getStatus();
});

driver.get("https://example.com");

// Совершаем действия
// Проверяем
```

Консоль браузера



The screenshot shows the Chrome DevTools Console with the 'gwtlog' network request selected. The 'Payload' tab is active, displaying the following JSON data:

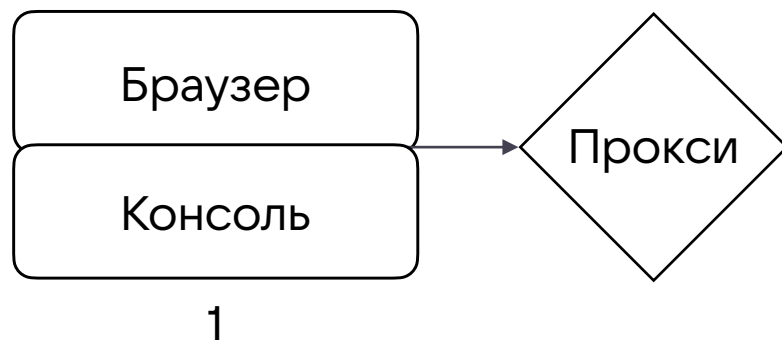
```
▼Form Data view source view URL-encoded  
a: {"GSW":{"continue":{"start":1716355623086,"now":1716355743088,"appId":"1251640576","ref":-1,"focus":1,"reload":0,"milestone":12000}}:1}}  
statId: 9bd71a74-17ad-4135-b8ad-c50eb37d11df
```

Плюсы:
можно быстро
просматривать
события

Минусы:
не совсем честная
проверка, нет гарантии,
что дальше правильно
идет, chrome

Автоматизация:
подходит

Варианты



Прокси



Browsermob-proxy

<https://github.com/lightbody/browsermob-proxy>

Прокси



```
@Before
public void startProxy() {
    proxy.start(0);
    proxy.addRequestFilter((request, contents, messageInfo) -> {
        if (request.getUri().contains("analytics")) {
            String requestBody = contents.getTextContents();
            // todo parse and check
        }
        return null;
    });
}
```

Прокси



```
@Test
public void runBrowserWithProxy() throws MalformedURLException {
    ChromeOptions options = new ChromeOptions();
    options.addArguments("--ignore-certificate-errors");

    DesiredCapabilities capabilities = new DesiredCapabilities("chrome",
"128.0.6613.137", Platform.ANY);
    capabilities.setCapability(ChromeOptions.CAPABILITY, options);
    capabilities.setCapability(CapabilityType.PROXY,
ClientUtil.createSeleniumProxy(proxy));

    driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"),
capabilities);
    driver.get("https://ok.ru/");
}
```

Прокси



Browsermob-proxy

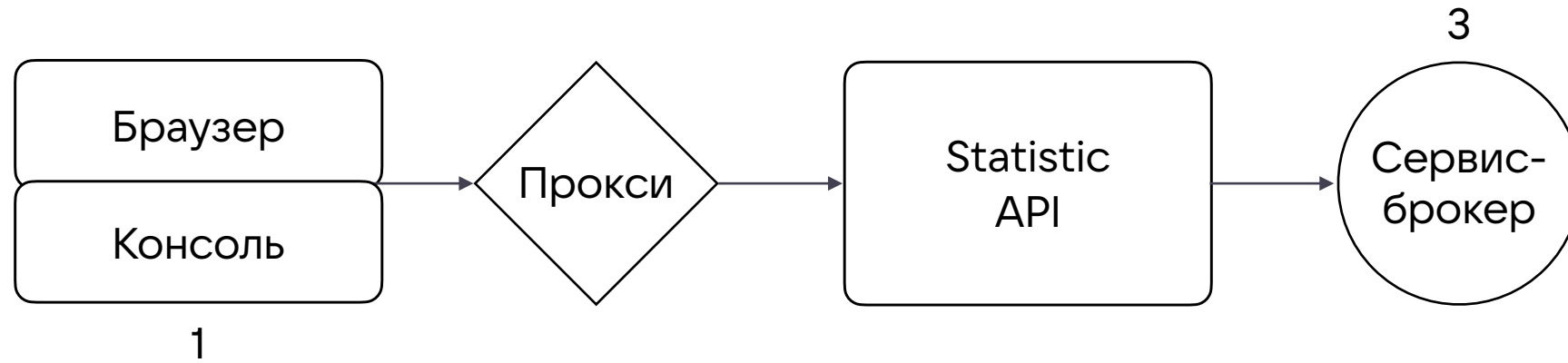
<https://github.com/lightbody/browsermob-proxy>

Плюсы: есть готовые решения, можно использовать

Минусы:
не совсем честная проверка, зависимость от библиотеки, сетевая доступность

Автоматизация:
подходит

Варианты



Сервис-брокер

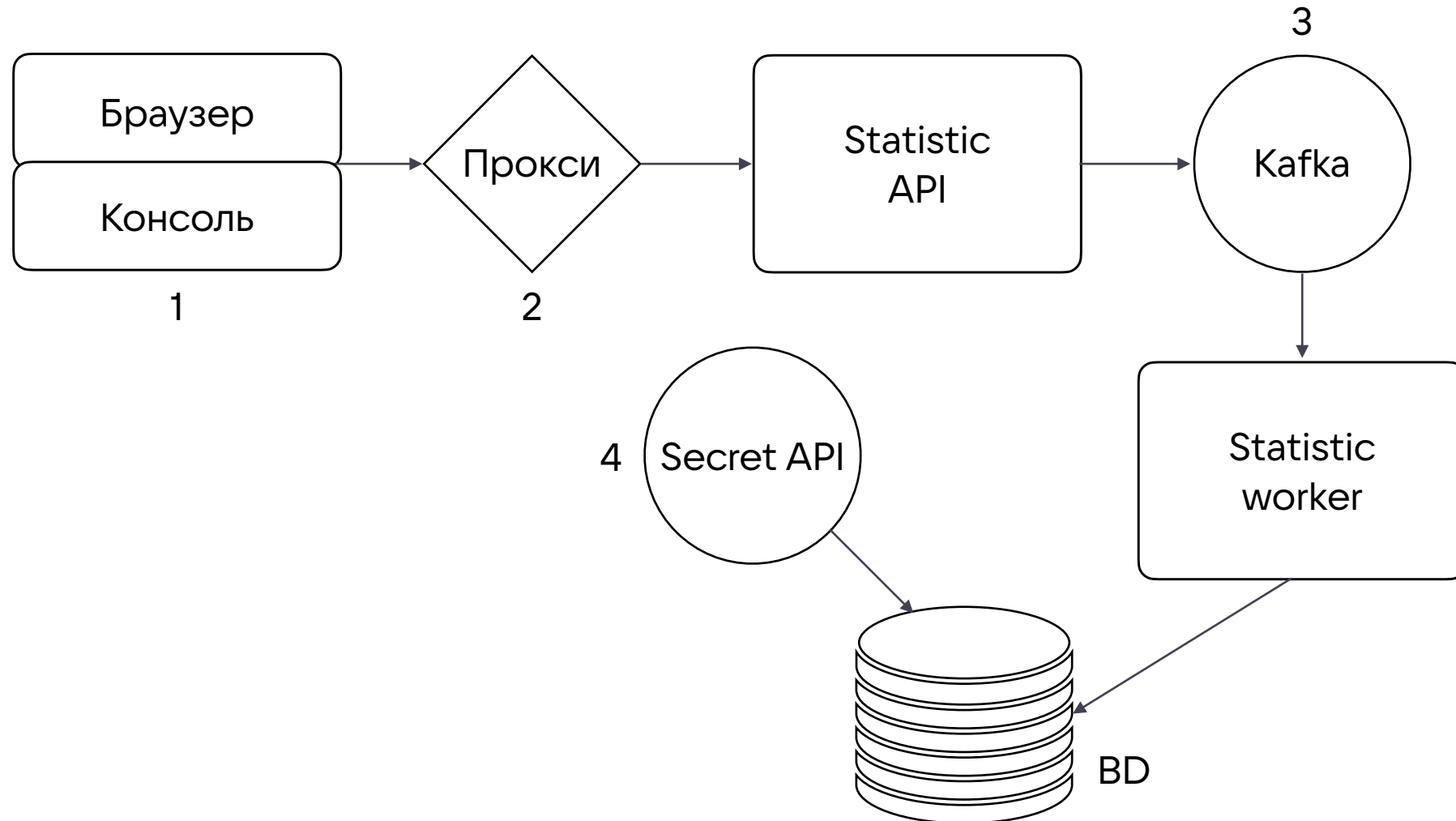


Плюсы:
вся нужная информация о событиях

Минусы:
мы должны быть уверены
в работоспособности
и достоверности данных, доступ,
долгое ожидание событий

Автоматизация:
подходит

Варианты



Secret API

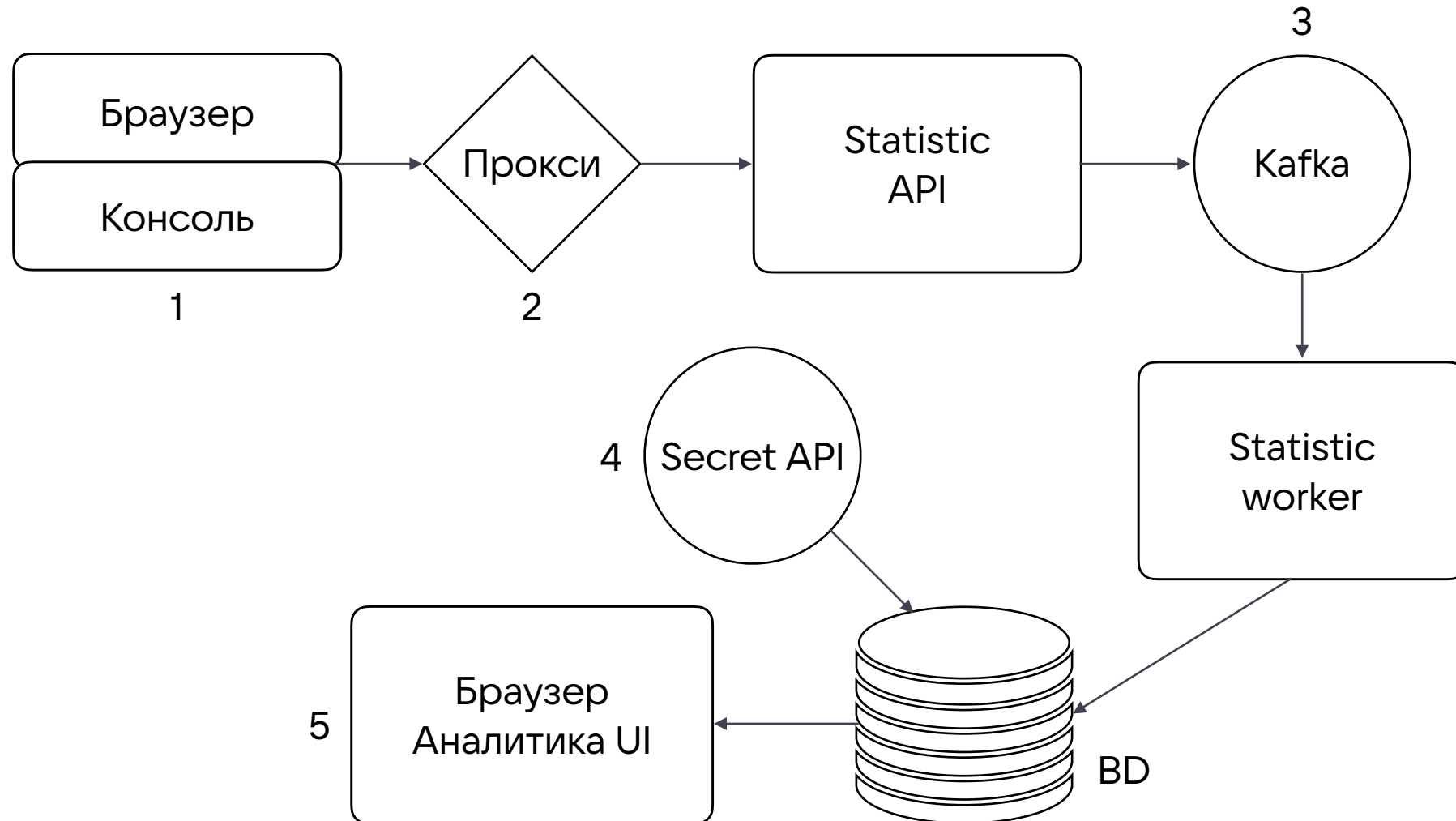


Плюсы:
довольно честная проверка, легко
выстраивать автоматизацию

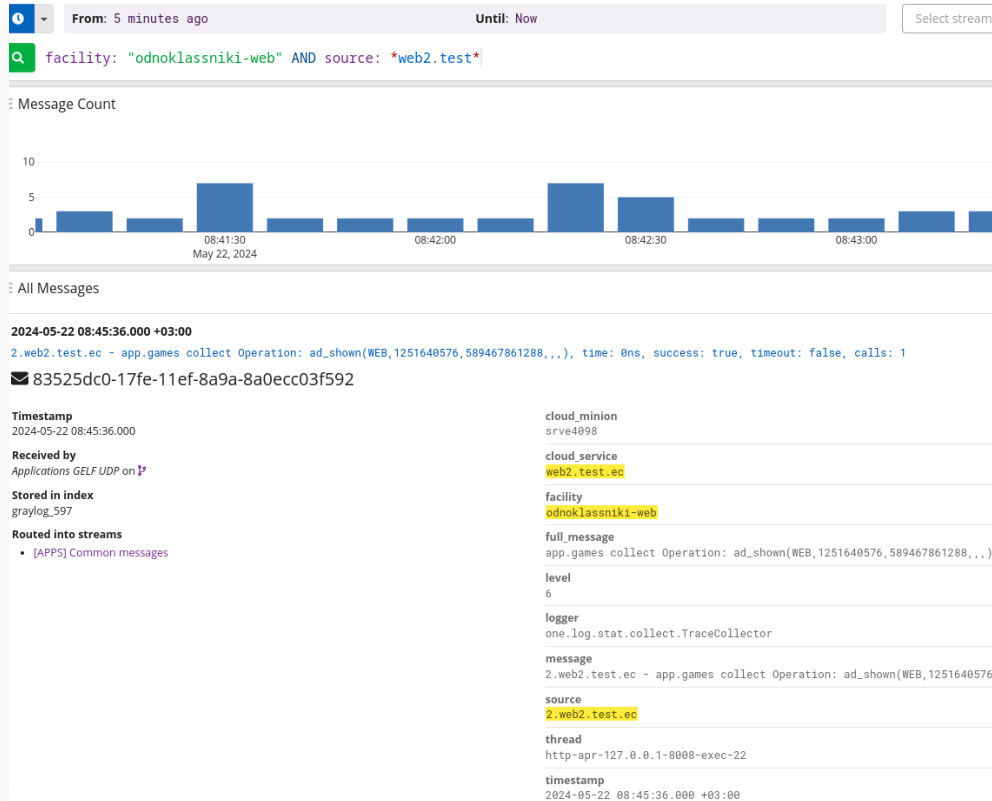
Минусы:
ресурсы разработчиков

Автоматизация:
подходит

Варианты



UI Аналитики



Плюсы:
максимально честная
проверка

Минусы:
сложно, долго и дорого
выстраивать автоматизацию

Автоматизация:
не подходит

Итог

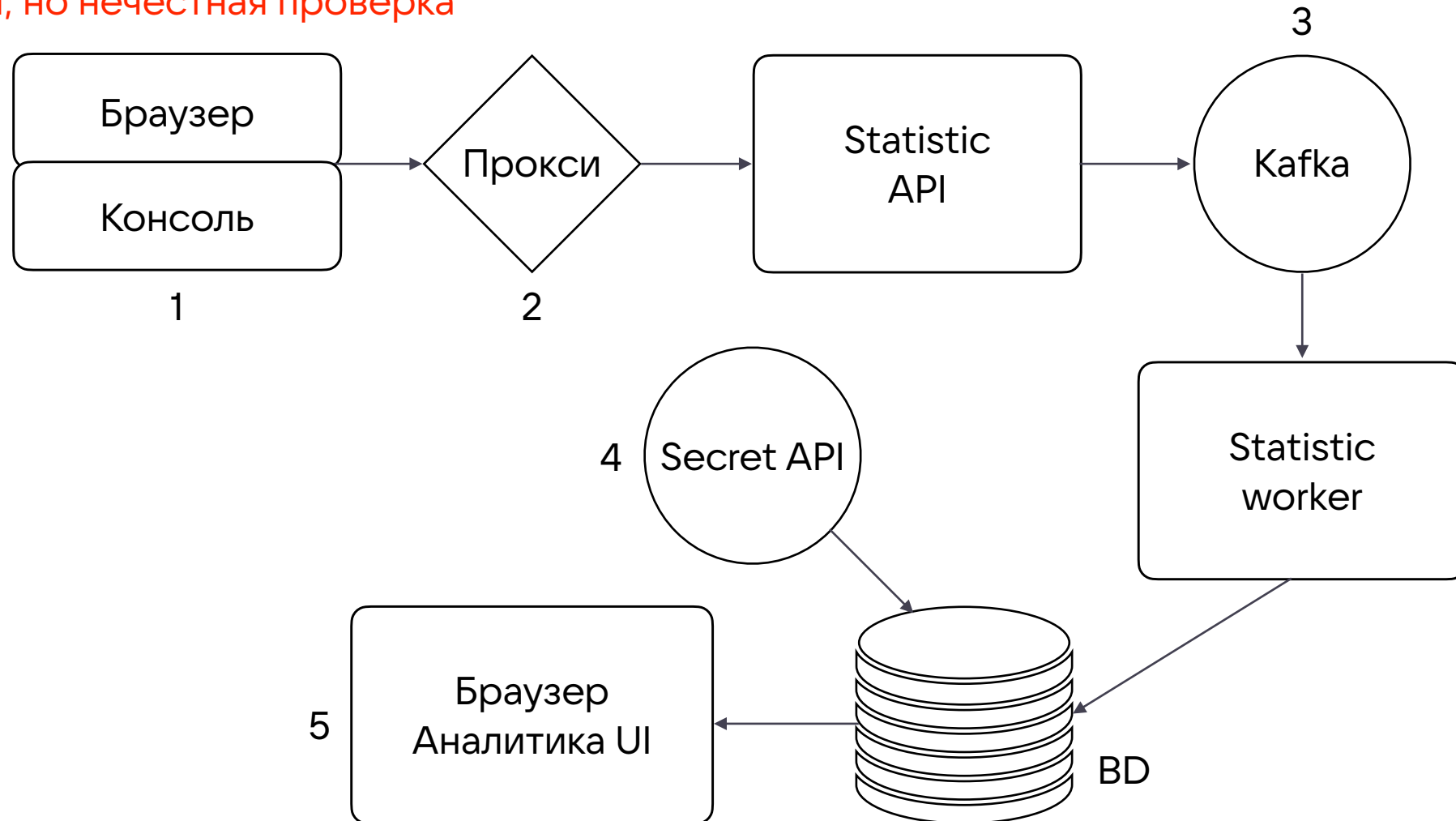


	Автоматизация	Быстрое решение	Честная проверка	Доп. ресурсы
Консоль браузера	+	+	-	-
Прокси	+	+	-	-
Сервис брокер	+	-	+	+
Secret API	+	-	+	+
Браузер Аналитика UI	-	-	+	+

Варианты



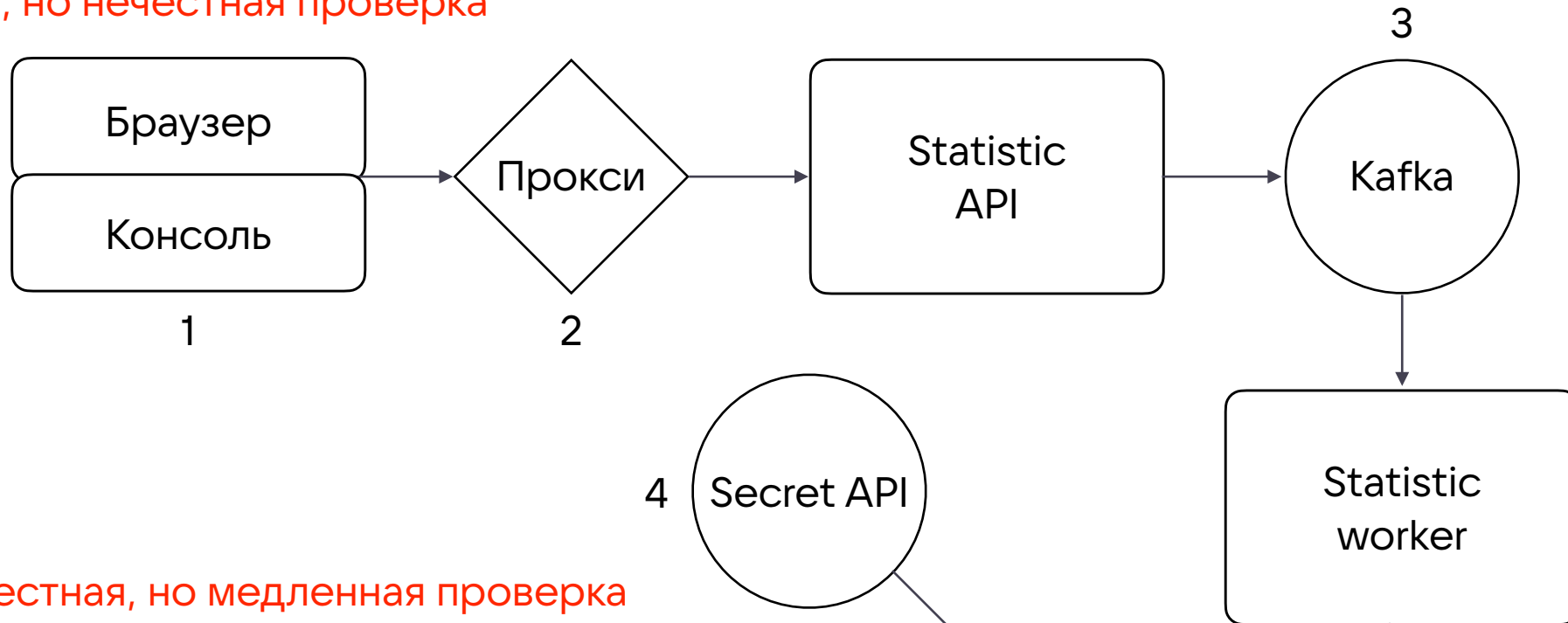
Быстрая, но нечестная проверка



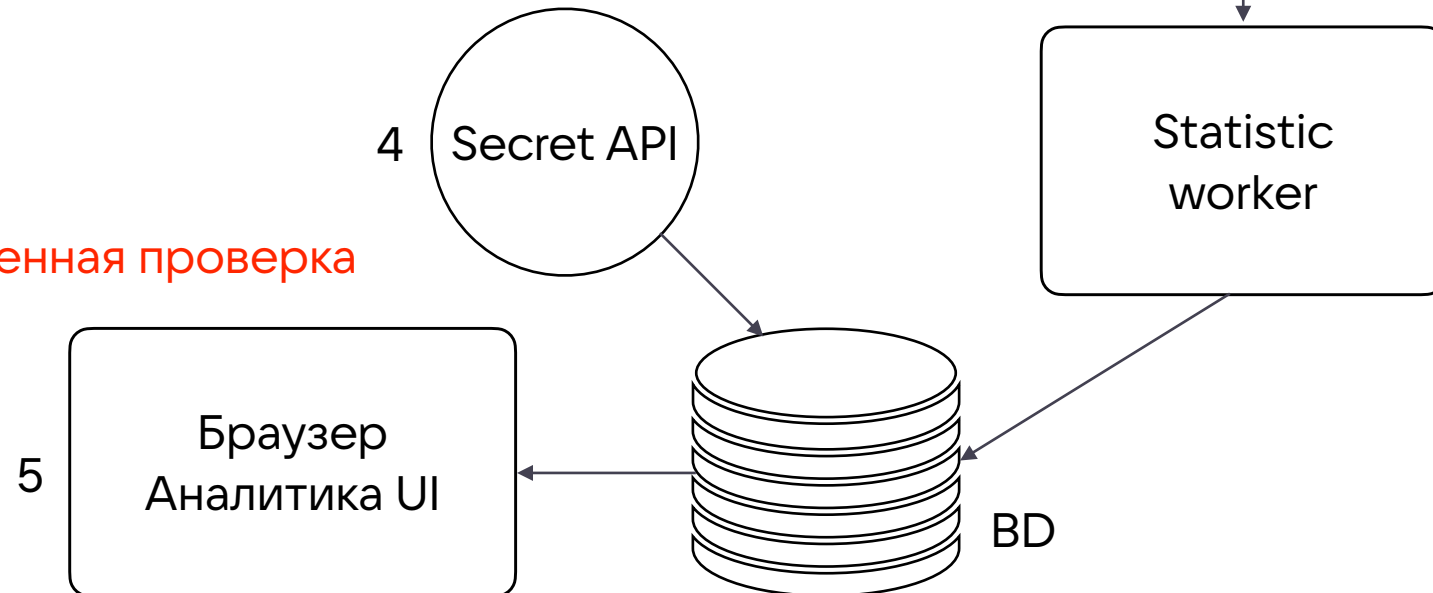
Варианты



Быстрая, но нечестная проверка



Честная, но медленная проверка





Что сделали в ОК

Что выбрали



	Автоматизация	Быстрое решение	Честная проверка	Доп ресурсы
Консоль браузера	+	+	-	-
Прокси	+	+	-	-
Сервис брокер	+	-	+	+
Secret API	+	-	+	+
Браузер Аналитика UI	-	-	+	+

Что выбрали



	Автоматизация	Быстрое решение	Честная проверка	Доп ресурсы
Консоль браузера	+	+	-	-
Прокси	+	+	-	-
Kafka	+	-	+	+
Secret API	+	-	+	+
Браузер Аналитика UI	-	-	+	+

Kafka



- Consumer
- Producer
- Broker
- Topic

Kafka



org.apache.kafka:kafka-clients:2.3.0



Тестовая среда



→ Тестовый стенд ОК

Тестовая среда



- Тестовый стенд ОК
- Тестовые кластера кафки

Тестовая среда



- Тестовый стенд ОК
- Тестовые кластера кафки
- Время ожидания данных ~1 минута

Тестовая среда



- Тестовый стенд ОК
- Тестовые кластера кафки
- Время ожидания данных ~1 минута
- Нет большой нагрузки

Тестовая среда



- Тестовый стенд ОК
- Тестовые кластера кафки
- Время ожидания данных ~1 минута
- Нет большой нагрузки
- Возможность получить отдельные доступы для проекта

Как устроено

Для каждого теста в Before своя логика

→ Создаем `getKafkaConsumer()`



Как устроено



```
@BeforeEach
void before() {
    ExecutorService executor = Executors.newFixedThreadPool(1);
    Future<EventsAssert> future = executor.submit(
    KafkaConsumer<byte[], byte[]> consumer = getKafkaConsumer();
```

Как устроено



```
KafkaConsumer<byte[], byte[]> getKafkaConsumer() {  
    KafkaConsumer<byte[], byte[]> kafkaConsumer = new KafkaConsumer<>(  
        getProperties(),  
        Serdes.ByteArray().deserializer(),  
        Serdes.ByteArray().deserializer());  
    consumers.add(kafkaConsumer);  
    return kafkaConsumer;  
}
```

Как устроено

Для каждого теста в Before своя логика

- Создаем `getKafkaConsumer()`
- Создаем `getKafkaTopics()`



Как устроено



```
@BeforeEach
void before() {
    ExecutorService executor = Executors.newFixedThreadPool(1);
    Future<EventsAssert> future = executor.submit(
    KafkaConsumer<byte[], byte[]> consumer = getKafkaConsumer();
    final Collection<String> topics = getKafkaTopics();
    consumer.subscribe(topics);
}
```

Как устроено



Для каждого теста в Before своя логика

- Создаем `getKafkaConsumer()`
- Создаем `getKafkaTopics()`
- Далее идет бесконечный цикл ожидания событий, который прерывается, когда завершается тест
 - + встроено ожидание

Как устроено



```
@BeforeEach
void before() {
    ExecutorService executor = Executors.newFixedThreadPool(1);
    Future<EventsAssert> future = executor.submit(
    KafkaConsumer<byte[], byte[]> consumer = getKafkaConsumer();
    final Collection<String> topics = getKafkaTopics();
    consumer.subscribe(topics);
    while (running.get()) {
        ConsumerRecords<byte[], byte[]> poll =
    consumer.poll(Duration.ofSeconds(0));
        for (ConsumerRecord<byte[], byte[]> record : poll) {
            events.add(record.value());
        }
    } ); //...
    executor.shutdown();}
```

Group id



- Добавляем проперти group id с уникальным названием (имя тестового класса)

Group id



- Добавляем проперти group id с уникальным названием (имя тестового класса)
- У каждого теста свой kafka consumer, каждый consumer видит всю информацию, но забирает только то, что ему нужно. Но как?

Как устроено



```
KafkaConsumer<byte[], byte[]> getKafkaConsumer() {  
    KafkaConsumer<byte[], byte[]> kafkaConsumer = new KafkaConsumer<>(  
        getProperties(),  
        Serdes.ByteArray().deserializer(),  
        Serdes.ByteArray().deserializer());  
    consumers.add(kafkaConsumer);  
    return kafkaConsumer;  
}
```

Group id



```
Properties conf = new Properties();
String servers = getServers();
conf.put("bootstrap.servers", servers);
conf.put("metadata.broker.list", servers);
conf.put("group.id", this.getClass().getName() + RandomStringUtils.randomAlphabetic(3))
```

Group id



```
Properties conf = new Properties();
String servers = getServers();
conf.put("bootstrap.servers", servers);
conf.put("metadata.broker.list", servers);
conf.put("group.id", this.getClass().getName() + RandomStringUtils.randomAlphabetic(3))
```

Данные топики



Название	Location	Target	Context
Рендер экрана	psw_change.phone	Settings	RENDER
Клик на	psw_change.login	Settings	CLICK
Ошибка	psw_change.phone	Settings	ERROR
Успех	psw_change.login	Settings	SUCCESS
Рендер экрана	lgn_change.name	Settings	RENDER
Клик на	lgn_change.surname	Settings	CLICK
Ошибка	lgn_change.name	Settings	ERROR
Успех	lgn_change.surname	Settings	SUCCESS
Рендер экрана	ph_change.text	Settings	RENDER
Клик на	ph_change.text	Settings	CLICK
Ошибка	ph_change.text	Settings	ERROR
Успех	ph_change.text	Settings	SUCCESS

Таблица = топик в Kafka



```
protected Collection<String> getKafkaTopics() {  
    return statsTestConfiguration.topics();  
}
```

@Override

```
protected Collection<String> getKafkaTopics() {  
    return List.of("extendedUserEvents_test");  
}
```

Test id Идентификатор теста



```
getWebDriver().manage().addCookie()
```

Идентификатор
теста

Как устроено



```
@BeforeEach
void before() {
    ExecutorService executor = Executors.newFixedThreadPool(1);
    //...
    List<TestActivityEvent> collect = events.stream()
        .map(bytes -> {
            //...
        })
        .filter(Objects::nonNull)
        .filter(this::filterByStatId)
        .filter(this::customEventFilter)
        .collect(Collectors.toList());
    return new EventsAssert(collect);
}
```

Как устроено



```
@BeforeEach
void before() {
    ExecutorService executor = Executors.newFixedThreadPool(1);
    //...
    List<TestActivityEvent> collect = events.stream()
        .map(bytes -> {
            //...
        })
        .filter(Objects::nonNull)
        .filter(this::filterByStatId)
        .filter(this::customEventFilter)
        .collect(Collectors.toList());
    return new EventsAssert(collect);
}
```


Что проверяем



→ location

→ target

→ context

→ type

Проверка



- Есть список ожидаемых проверок
- В тесте в конце проверяем ожидаемое с фактическим

Проверка



- Есть список ожидаемых проверок
- В тесте в конце проверяем ожидаемое с фактическим

```
{ "type": "SUCCESS", "statId": "a6[REDACTED]4b30"
{ "type": "ACTION", "statId": "a6[REDACTED]4b30",
{ "type": "ACTION", "statId": "a6[REDACTED]4b30",
{ "type": "ACTION", "statId": "a658[REDACTED]30",
{ "type": "SUCCESS", "statId": "a65[REDACTED]b30"
{ "type": "ERROR", "statId": "a65[REDACTED]b30", "s
{ "type": "ERROR", "statId": "a65[REDACTED]b30", "s
```

Tect



```
@Test
```

```
void statRegistrationTest() {  
    //do test steps
```

```
    checkAction(events, RENDER, HOME_LOGIN_FORM);  
    checkAction(events, RENDER, PHONE_REG);  
    checkAction(events, RENDER, CODE_REG);  
    checkAction(events, RENDER, PASSWORD_VALIDATE_LOGIN_VIEW);  
    checkActionAndContext(events, RENDER, PROFILE_FORM, OK);  
    checkActionAndTarget(events, CLICK, HOME_LOGIN_FORM, REGISTER);  
    checkActionAndTarget(events, CLICK, PHONE_REG, COUNTRY);  
    checkActionAndTarget(events, CLICK, PHONE_REG, PHONE);  
    checkActionAndTarget(events, CLICK, PHONE_REG, SUBMIT);  
    checkActionAndTarget(events, SUCCESS, PHONE_REG, SUBMIT);  
    checkActionAndTarget(events, CLICK, CODE_REG, CODE);  
    checkActionAndTarget(events, CLICK, CODE_REG, SUBMIT);  
}
```

Переиспользуем автотесты



→ Подключили себе проект с автотестами как зависимость

Переиспользуем автотесты



- Подключили себе проект с автотестами как зависимость
- Раз в какое-то время автоматически собирать новую версию

Переиспользуем автотесты



- Подключили себе проект с автотестами как зависимость
- Раз в какое-то время автоматически собирать новую версию
- Обновлять в проекте с тестами статистики

Отчетность








→ Один тест — один отчет

Отчетность



- Один тест — один отчет
- Проверки и флажок успешно/фейл

Самый удобный отчет в виде

Событие 1	
Событие 2	
Событие 3	
Событие 4	
Событие 5	

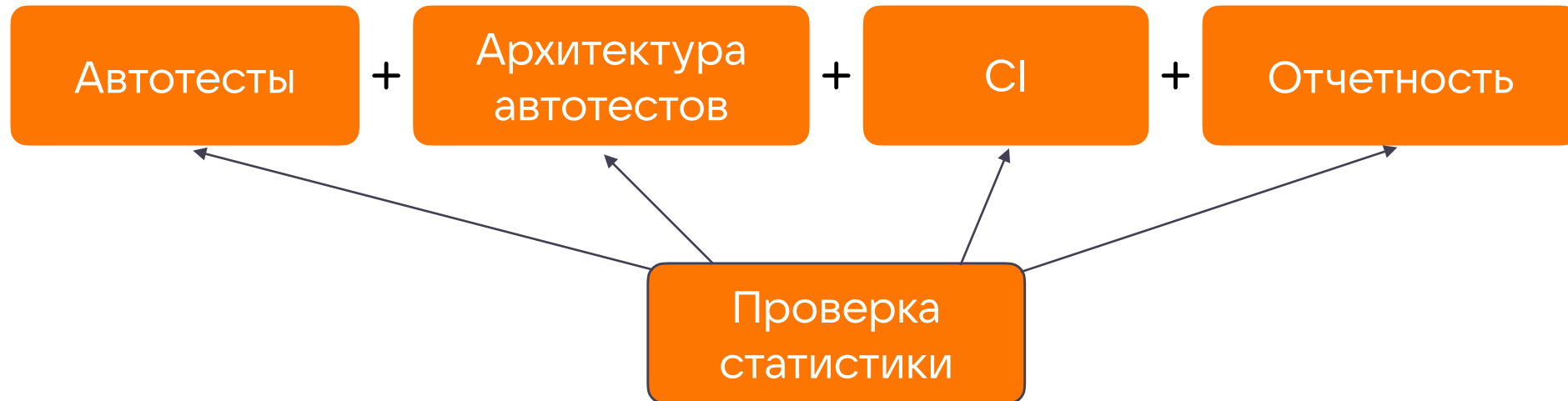


Что можем теперь

На каждое изменение, на каждый коммит разработчика можно отслеживать, что статистика пишется без изменений

Можно встроить в апдейт

Структура





Вывод

Итоги



- Автоматизировать статистику можно и нужно. Это важно бизнесу
- Это не так сложно, как кажется. Важно отталкиваться от того, что уже есть на проекте. Максимально переиспользовать
- А если автоматизации ещё нет, то это повод ее развивать



Спасибо за внимание!

 @alonadeveraux

