

Как навести порядок в логировании продуктовых событий

Алексей Балехов
архитектор

ökkö

События с фронтендов и бэкендов приложения, используемые дата-аналитиками для построения продуктовых метрик по различным функциям приложения.

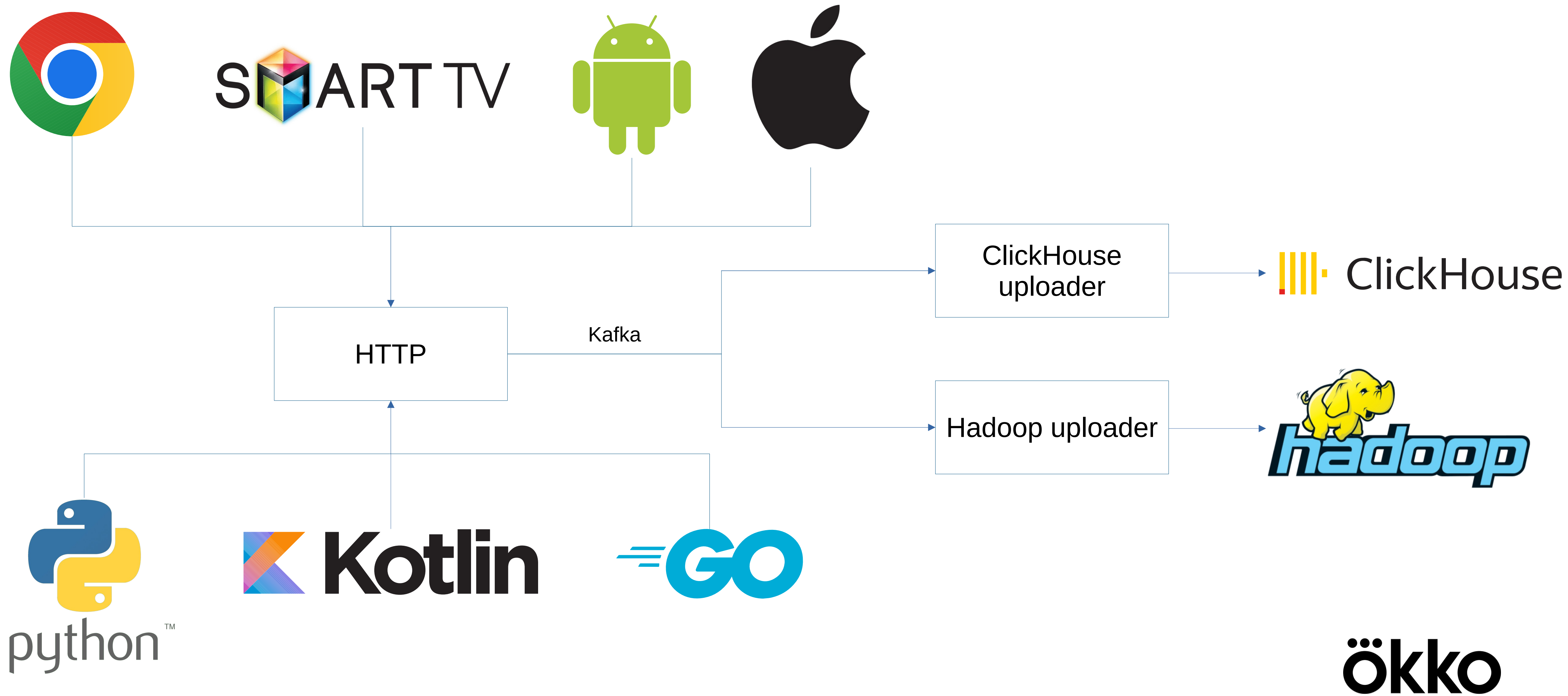
```
{  
  "_type": "playbackPlayerEvent",  
  "action": "startWatch",  
  "playbackId": "a84e0b2e-45b3-11ee-be56-0242ac120002",  
  "quality": "fullhd",  
  "volumeLevel": 78,  
  ...  
}
```

Архитектор и фулстек-разработчик

2 года строил систему статистики для блогеров в Яндекс-Дзене. Копил боль от грязных данных в логах.

9 месяцев добиваюсь чистоты при сборе данных в онлайн-кинотеатре Okko.

1. Исходное состояние (бардак)





1. Расходятся имена полей.

JSON

```
{  
  ...  
  "isGroupedSearch": true,  
  ...  
}
```

ClickHouse

```
`groupedSearch` Boolean
```

1. Расходятся имена полей.
2. Расходятся значения enum-ов.

```
iOS  
{  
    ...  
    "quality": "fullhd"  
}
```

```
Android  
{  
    ...  
    "quality": "Q_FULL_HD"  
}
```


1. Расходятся имена полей.
2. Расходятся значения enum-ов.
3. Расходятся типы полей.

JSON

```
{  
    ...  
    "newUser": true,  
    ...  
}
```

ClickHouse

```
`newUser` String
```

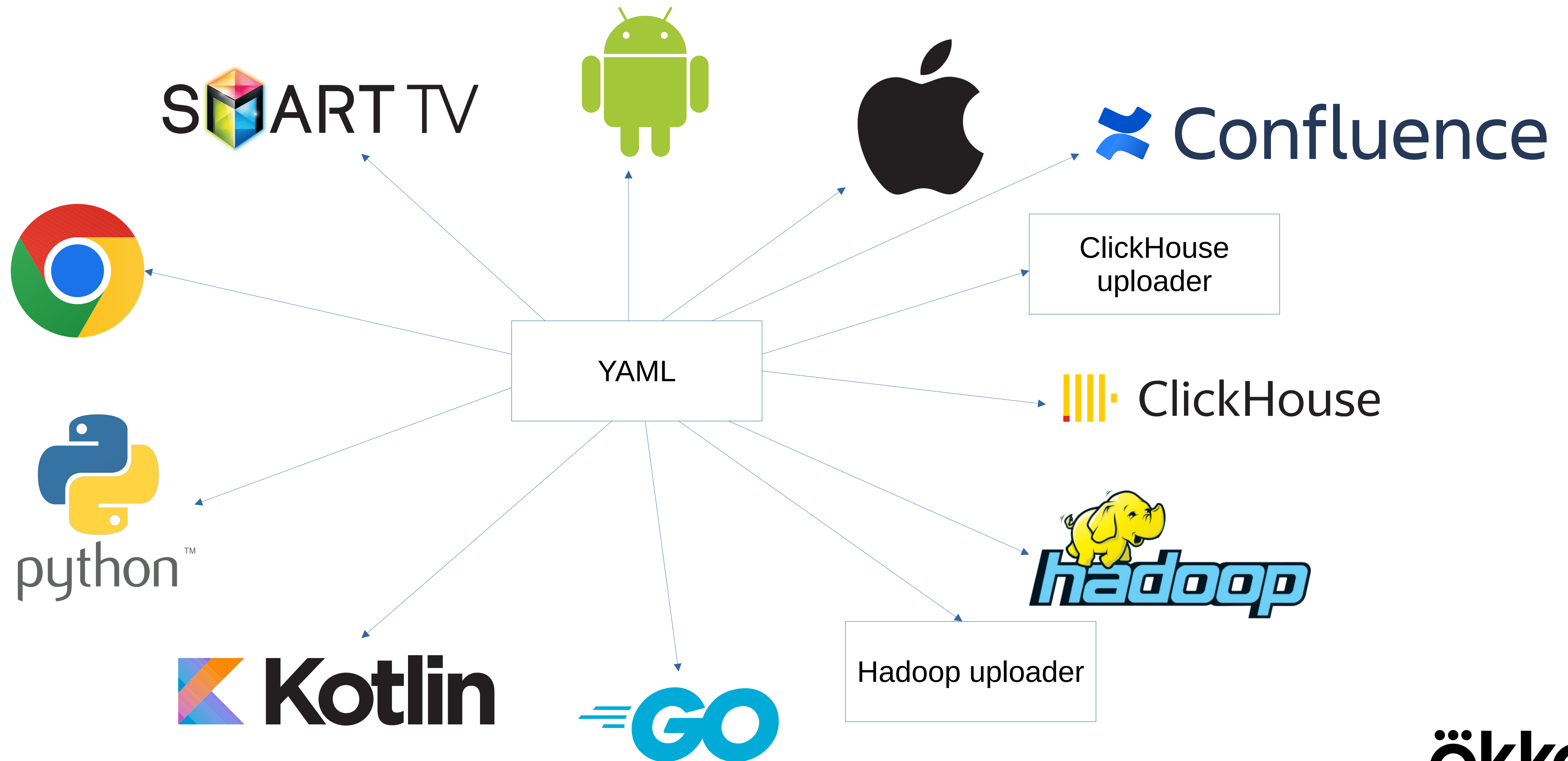
1. Расходятся имена полей.
2. Расходятся значения enum-ов.
3. Расходятся типы полей.
4. Слабая типизация: не используем UUID, DateTime, Enum, etc.

1. Расходятся имена полей.
2. Расходятся значения enum-ов.
3. Расходятся типы полей.
4. Слабая типизация: не используем UUID, DateTime, Enum, etc.
5. Неконсистентный нейминг: camelCase, snake_case, lowercase.

1. Расходятся имена полей.
2. Расходятся значения enum-ов.
3. Расходятся типы полей.
4. Слабая типизация: не используем UUID, DateTime, Enum, etc.
5. Неконсистентный нейминг: camelCase, snake_case, lowercase.
6. Сложно отслеживать изменения.

1. Расходятся имена полей.
2. Расходятся значения enum-ов.
3. Расходятся типы полей.
4. Слабая типизация: не используем UUID, DateTime, Enum, etc.
5. Неконсистентный нейминг: camelCase, snake_case, lowercase.
6. Сложно отслеживать изменения.
7. Фрод.

2. Формальный источник правды



```
1 description: |
2   Heartbeat просмотра. Отправляется раз в 30 секунд, хранит накопленное реальное время просмотра.
3   Так же приходит вместе с всеми сигнатурами прерывания просмотра buffering_start, pause, seek, stop, complete, terminate, start_ad, при
4 properties:
5   playbackId:
6     type: string
7     description: Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры.
8   duration:
9     type: float
10    description: Продолжительность просмотра в секундах
11   position:
12     type: float
13    description: Позиция просмотра (playPosition на таймлайне в секундах)
```



```
19 @Serializable
20 data class PlaybackWatchTimeEvent(
21     /** Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры. */
22     val playbackId: String,
23     /** Продолжительность просмотра в секундах */
24     val duration: Float,
25     /** Позиция просмотра (playPosition на таймлайне в секундах) */
26     val position: Float,
27 ) : Event
```

```
86 export type PlaybackWatchTimeEvent = {
87     /** Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры. */
88     playbackId: string;
89     /** Продолжительность просмотра в секундах */
90     duration: number;
91     /** Позиция просмотра (playPosition на таймлайне в секундах) */
92     position: number;
93 }
```

```
1 --liquibase formatted sql
2
3 --changeset generator:1 contextFilter:clickhouse
4 CREATE TABLE IF NOT EXISTS kollector.playback_watch_time ON CLUSTER kollector
5 (
6     `playback_id` String COMMENT 'Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры.',
7     `duration` Int32 COMMENT 'Продолжительность просмотра в секундах',
8     `position` Int32 COMMENT 'Позиция просмотра (playPosition на таймлайне в секундах)',
9     `kollectorts` DateTime64(3) COMMENT 'Таймстемп попадания события в коллктор в миллисекундах'
10 )
11 ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/{database}/{table}', '{replica}')
12     PARTITION BY toYYYYMM(kollectorts)
13     ORDER BY kollectorts;
14 |
15
16 --changeset generator:2 contextFilter:impala
17 CREATE TABLE IF NOT EXISTS `playback_watch_time`
18 (
19     `playback_id` string COMMENT 'Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры.',
20     `duration` int COMMENT 'Продолжительность просмотра в секундах',
21     `position` int COMMENT 'Позиция просмотра (playPosition на таймлайне в секундах)',
22     `kollectorts` timestamp COMMENT 'Таймстемп попадания события в коллктор в миллисекундах'
23 )
24 PARTITIONED BY (year int, month int, day int)
25 COMMENT 'Heartbeat просмотра. Реальное время просмотра. Отправляется каждые 30 секунд, а также вместе с playerEvent bufferingStart|pause|seek|stop|complete|terminate|startAd\n'
26 WITH SERDEPROPERTIES ('serialization.format'='1')
27 LOCATION '/user/hive/warehouse/playback_watch_time';
```

```
42     newUser:  
43     description: Новый пользователь. true - если это первый логин пользователя.  
44     type: string  
45     deprecatedKotlinType: Boolean  
46     deprecatedDwhName: newUser
```

deprecatedDwhName - имя столбца в Hadoop и ClickHouse.

deprecatedJsonName - имя ключа при передаче от клиента.

deprecatedKotlinName - имя свойства data-класса в сгенерированном для серверных эвентов коде. Используется только в коде.

deprecatedTsType - тип данных, который передаёт веб-клиент.

deprecatedKotlinType - тип данных, который передаёт микросервис.

deprecatedClickHouseDisable - столбец в КХ отсутствует.

deprecatedHadoopType – тип столбца в Hadoop.

1. Pull Request с изменениями в YAML-схемах
2. CI делает коммит с изменениями в сгенерированных файлах
3. Review и Merge
4. CI
 - Сборка и публикация новых версий библиотек для разработчиков
 - Обновление документации в Confluence
 - Уведомление в Mattermost
 - Накатка миграций на ClickHouse и Hadoop
5. Обновление библиотек на клиентах, внесение изменений в местах генерации событий

DEV-67820 add callId field to plRails.event

Overview **Diff** Commits Builds

All changes in this pull request
5 commits

Filter file tree Search code

- kollector-client-spring-boot-starter/src/main/java/tv/okko/kc
 - model
 - events.kt
 - utils
 - ProtobufConversionUtils.kt
- kollector-kotlin-schemas/src/main/kotlin/tv/okko/kollector/n
 - plRailsSchema.kt
- kollector-python-client/kollector_client/models
 - pl_rails.py
- migrations/scripts
 - 59-DEV_67820-generated.sql
- proto
 - plRails.proto
- schemas/server
 - plRails.event.yml

schemas / server / plRails.event.yml **MODIFIED**

```
194 194 deprecatedDwhName: railitemsids
195 195 type: array
196 196 items:
197 197   type: string
198 198 default: []
199 199 nid:
200 200 description:
201 201 type: string
202 202 nullable: true
203 203 deprecatedKotlinType: UUID
204 + callId:
205 + description: 'Уникальный идентификатор запроса, поз
206 + type: string
207 + nullable: true
```



3. Повышаем гарантии

```
1 type: client
2 description: События просмотра и взаимодействия с плеером
3 properties:
4   action:
5     description: Сигнатура просмотра
6     type: enum
7     values:
8       ready: событие готовности плеера. Отправляется, когда плеер перешел из Loading -> Ready. С duration сколько плеер находился в статусе Loading.
9       bufferingStart: начало буферизации
10      bufferingComplete: завершение буферизации
11      startWatch: события начала воспроизведения. Отправляется, при переходе из Ready -> Playing.
12      playing: изменение времени плеябека (проиграна 1 секунда), отправляется при разнице currentTime > 1 секунды (0 если сначала или значения n если продолжить просмотр).
13      # ...
```

```
4493 export enum PlaybackPlayerEventAction {
4494     /** событие готовности плеера. Отправляется, когда плеер перешел из Loading -> Ready. С duration сколько плеер находился в статусе Loading. */
4495     Ready = "ready",
4496     /** начало буферизации */
4497     BufferingStart = "bufferingStart",
4498     /** завершение буферизации */
4499     BufferingComplete = "bufferingComplete",
4500     /** события начала воспроизведения. Отправляется, при переходе из Ready -> Playing. */
4501     StartWatch = "startWatch",
4502     /** изменение времени плеябека (проиграна 1 секунда), отправляется при разнице currentTime > 1 секунды (0 если сначала или значения n если продолжить просмотр). */
4503     Playing = "playing",
4504     /** пользователь выполнил действие, которое устанавливает контент в режим пауза. Данная сигнатура вызывается при нажатии на кнопку
```

```
14 duration:
15   conditional:
16     action: [completeAd, bufferingComplete, resume, ready]
17   type: float
18   description: Продолжительность процесса (отправляется вместе с каждым событием буферизации, паузы, рекламы) в секундах
19 speed:
20   conditional:
21     action: [startWatch, speedChange]
22   type: string
23   description: Скорость воспроизведения
24 volumeLevel:
```

```
4655 export type PlaybackPlayerEventActionProps =
4656   | { action: PlaybackPlayerEventAction.Ready } & {
4657     /** Продолжительность процесса (отправляется вместе с каждым событием буферизации, паузы, рекламы) в секундах */
4658     duration: number;
4659   }
4660   | { action: PlaybackPlayerEventAction.SpeedChange } & {
4661     /** Скорость воспроизведения */
4662     speed: string;
4663   }
4664   | { action: PlaybackPlayerEventAction.OrientationChange } & {
```


namedType

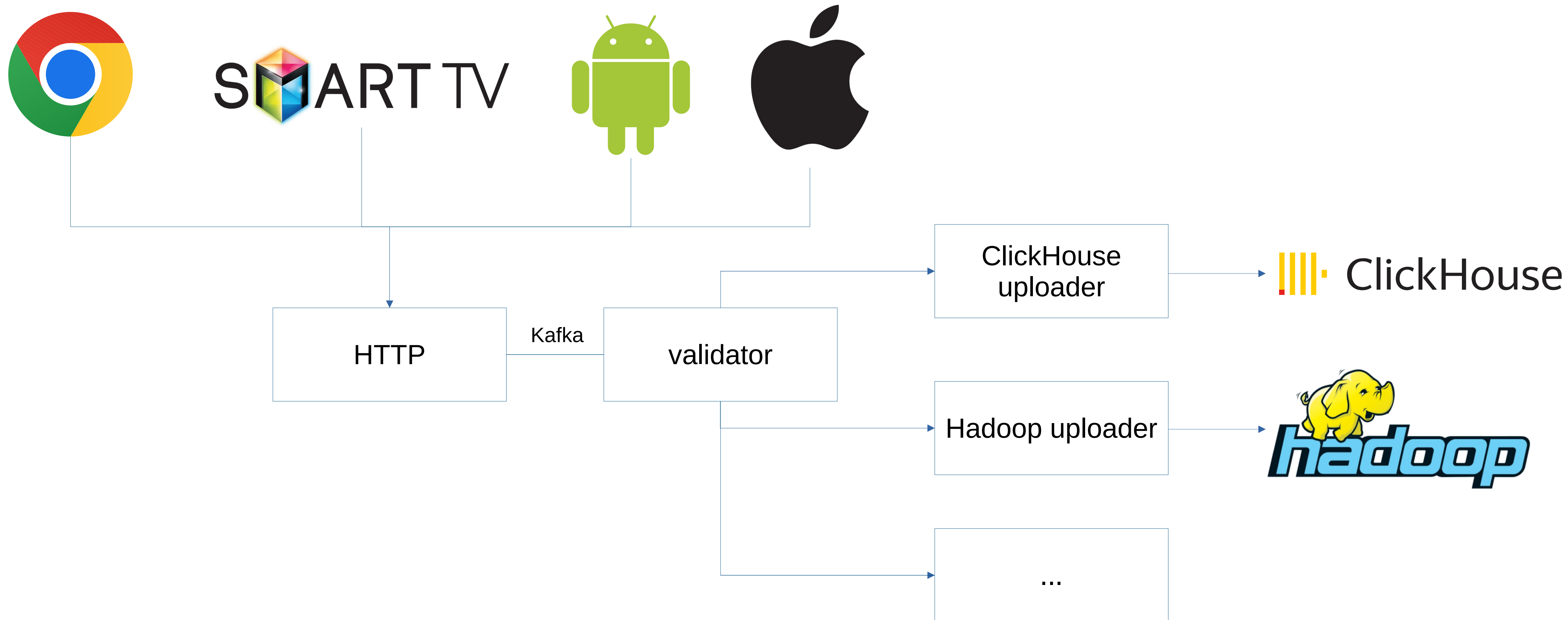
25

```
1 # DeviceType.namedType.yml
2 description: тип устройства
3 type: enum
4 values:
5   tv: телевизор
6   bdp: Blue Ray Player
7   mp: media player
8   mob: мобильный телефон
9   tbl: планшет
10  web: браузер
11  pc: PC-клиент
12  vgc: video game console (PlayStation)
13  stb: set top box
```

```
167 deviceType:
168   description: тип устройства
169   type: named
170   namedType: DeviceType
```

```
1 # Screen.namedType.yml
2 description: Экран
3 type: object
4 properties:
5   type:
6     description: Тип экрана
7     type: enum
8     values:
9       main: Главная
10      auth: Раздел авторизации
11      settings: Раздел настроек
12      # ...
13
14 auth:
15   description: Экран раздела авторизации
16   conditional:
17     type: [auth]
18   type: enum
19   values:
20     main: Главный экран (выбор способа входа)
21     pin: Авторизация по пинкоду или QR
22     resetPassword: Сброс пароля
23     # ...
24
25 settings:
26   description: Экран раздела настроек
27   conditional:
28     type: [settings]
29   type: enum
30   values:
31     menu: Меню "Профиль" (первый экран в приложениях)
32     main: Основные настройки (контакты и др.)
33     download: Настройки загрузки
34     # ...
```

```
1 # screenTransition.event.yml
2 description: Событие смены экрана
3 properties:
4   from:
5     description: С какого экрана выполнен переход
6     type: named
7     namedType: Screen
8
9   to:
10    description: На какой экран выполнен переход
11    type: named
12    namedType: Screen
```



4. Перепридумываем фронтальные события

bannerEvent - показ баннера и клики внутри него

collectionEvent - перемещение фокуса, скролл и клики в каруселях, а также изменение фильтров

contentInteractionEvent - букмарки, лайки, шаринг, оценки, скачивание

displayedElementsEvent - отображение карточек

menuEvent - клики в меню

notificationEvent - показ, клик и закрытие уведомления

popupEvent - показ, закрытие и клики внутри попапов

promoCodeEvent - отправка промокода, закрытие окна ввода

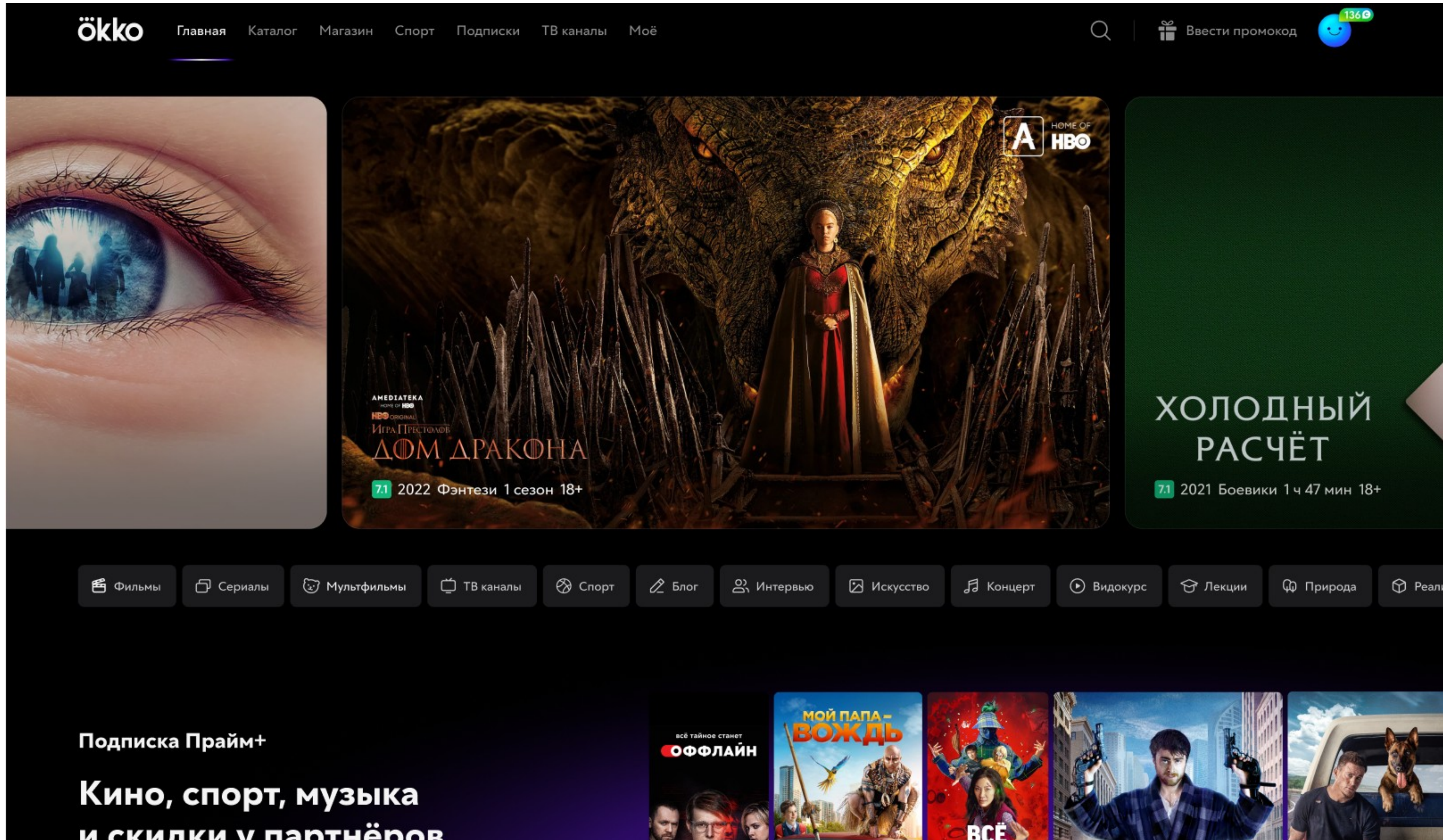
purchaseEvent - выбор продукта, интеракции в процессе покупки

screensInteractionsEvent - события взаимодействия с выборочными экранами (такими как авторизация, настройка профиля, подтверждение номера телефона или восстановление покупок)

settingEvent - изменения настроек

...

Блоковое логирование



The screenshot displays the ökko website interface with several UI elements highlighted for logging. At the top, the ökko logo is on the left, followed by a row of seven 'mainMenuItem' labels. On the right, there are 'mainMenuItem' labels for 'Вести промокод' and a 'mainMenuItem' label with a '85 x 56' dimension. Below this is a large 'rowElement' containing three 'itemElement' cards. The first card shows a close-up of an eye. The second card features a woman in a red dress and is labeled 'ДОМ ДРАКОНА' (House of Dragons) with details: '7.1 2022 Фэнтези 1 сезон 18+'. The third card shows a man in a dark coat and is labeled 'ХОЛОДНЫЙ РАСЧЁТ' (Cold Blooded) with details: '7.1 2021 Боевики 1 ч 47 мин 18+'. Below the row is another 'rowElement' containing ten 'itemElement' labels. At the bottom, there is a 'rowElement' with a promotional banner for 'Подписка Прайм+' (Prime+ Subscription) and a row of movie posters including 'МОЙ ПАПА - ВОЖДЬ' (My Dad is the Boss) and 'ВСЁ' (Everything).

```
{  
  _type: "blockInteraction",  
  
  action: "click",  
  
  blockPath: ["rowElement", "itemElement"],  
  
  rowElementIndex: 2,  
  rowElementId: "featured",  
  
  itemElementIndex: 3,  
  itemElementId: "25b774cd-dd86-4542-8aeb-8d0ec99f74e4",  
  itemElementExplanation: "genre",  
  
  // ...  
}
```



```
1 type: client
2 properties:
3   action:
4     description: Действие
5     type: enum
6     values:
7       show: Показ блока
8       click: Клик на блок
9       focus: Фокус блока (только для управления с пульта)
10      scroll: Скролл блока
11
12   blockPath:
13     description: Путь из родительских блоков до текущего элемента
14     type: array
15     items:
16       type: enum
17       values:
18         rowElement: 'Динамический блок с uiType: рейл, грид, баннер и т.д.'
19         itemElement: Элемент в рейле, гриде и т.д.
20         button: Кнопка или кликабельный текст (например, заголовок рейла)
21         mainMenuItem: Элемент главного меню
22         # ...
23
```

```
24   rowElementIndex:
25     conditional:
26       blockPath: [rowElement]
27     description: Порядковый номер строки. Нумерация с 0.
28     type: int16
29   rowElementId:
30     conditional:
31       blockPath: [rowElement]
32     description: UUID или один из айдишников динамических рейлов
33     type: string
34
35   itemElementIndex:
36     conditional:
37       blockPath: [itemElement]
38     description: Порядковый номер элемента в рейле. Нумерация с 0.
39     type: int16
40   itemElementId:
41     conditional:
42       blockPath: [itemElement]
43     description: Идентификатор элемента
44     type: string
45   itemElementExplanation:
46     conditional:
47       blockPath: [itemElement]
48     description: Причина рекомендации
49     type: string
50     nullable: true
51
```

ИТОГИ

1. Формализованные схемы позволили снизить влияние человеческого фактора.
2. Блоковое логирование упрощает покрытие новой функциональности.

Спасибо за внимание!

ökko

öökko