Как навести порядок в логировании продуктовых событий

Алексей Балехов
архитектор

Okko

Продуктовые события

События с фронтедов 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,
    ...
}
```

Okko

О докладчике

Архитектор и фулстек-разработчик

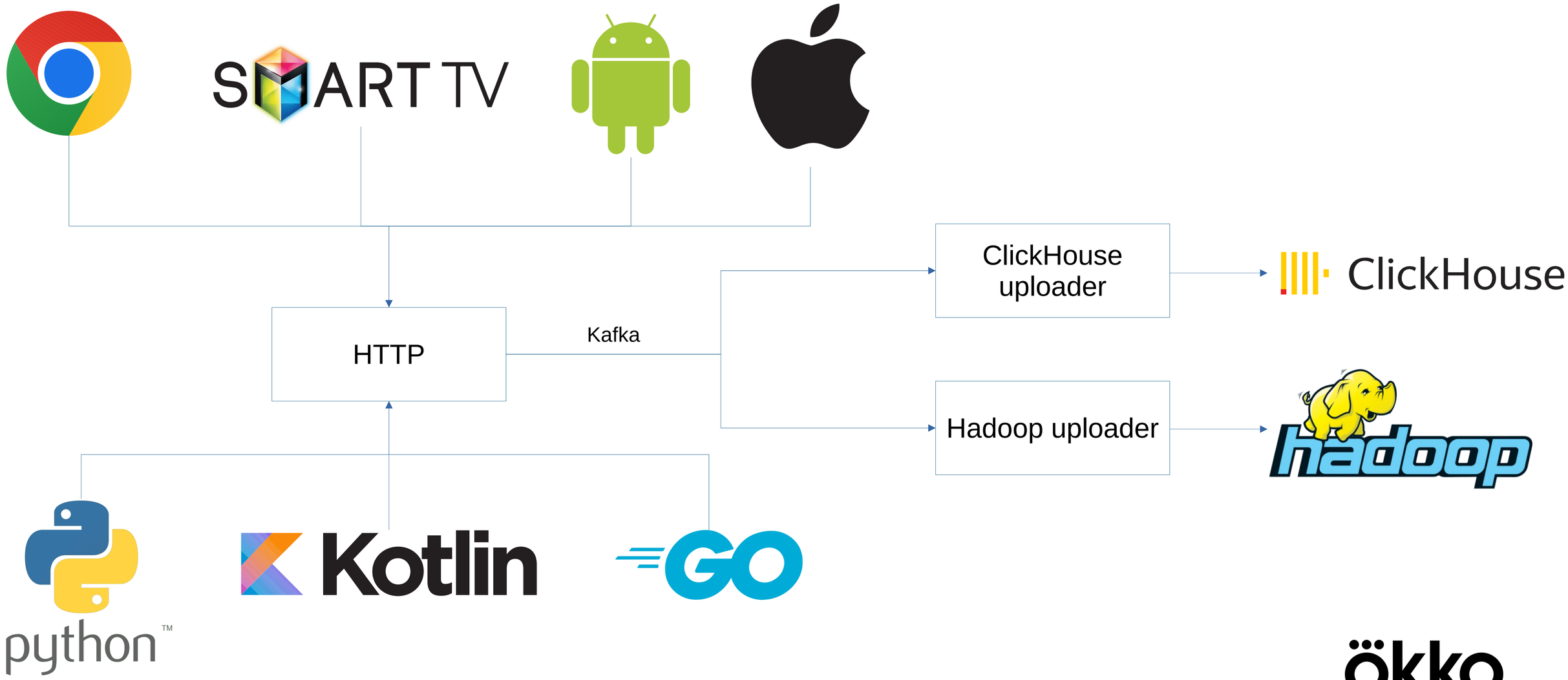
2 года строил систему статистики для блогеров в Яндекс-Дзене. Копил боль от грязных данных в логах.

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

1. Исходное состояние (бардак)

Okko

Архитектура



ökko

Источник правды

Проблемы

1. Расходятся имена полей.

JSON

```
{
    ...
    “isGroupedSearch”: true,
    ...
}
```

ClickHouse

```
    `groupedSearch` Boolean
```

ökko

Проблемы

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

```
iOS
{
    …
    “quality”: “fullhd”
}

Android
{
    …
    “quality”: “Q_FULL_HD”
}
```

ökko

Проблемы

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

JSON

```
{
    ...
    “newUser”: true,
    ...
}
```

ClickHouse

```
    `newUser` String
```

ökko

Проблемы

1. Расходятся имена полей.

2. Расходятся значения enum-ов.

3. Расходятся типы полей.

4. Слабая типизация: не используем UUID, DateTime, Enum, etc.

Проблемы

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.

Проблемы

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.

Проблемы

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.

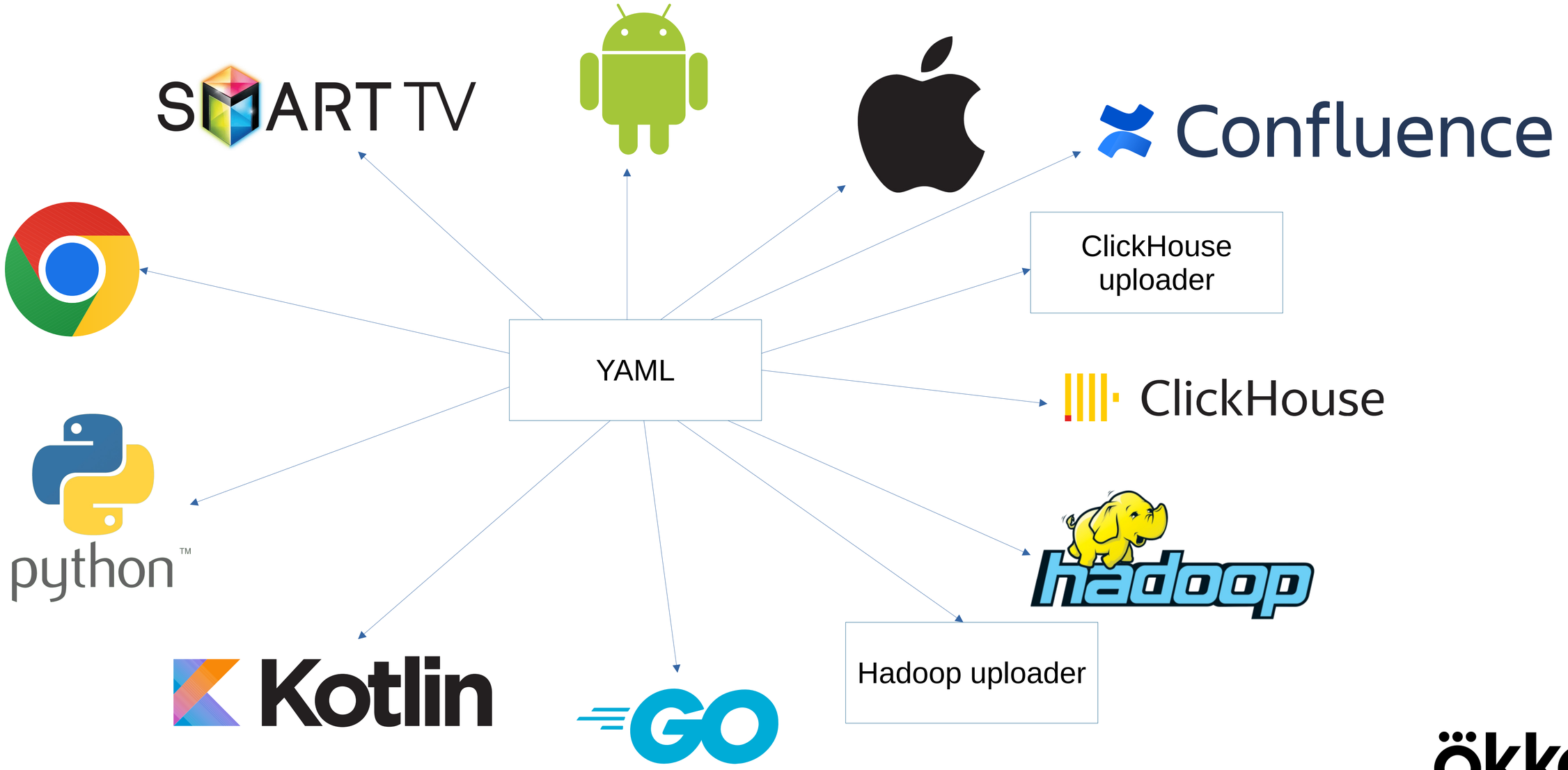
6. Сложно отслеживать изменения.

7. Фрод.

2. Формальный источник правды

Okko

Формализация схем



Формализация схем


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

Кодогенерация (клиенты)


```kotlin
@Serializable
data class PlaybackWatchTimeEvent(
    /** Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры. */
    val playbackId: String,
    /** Продолжительность смотрения в секундах */
    val duration: Float,
    /** Позиция просмотра (playPosition на таймлайне в секундах) */
    val position: Float,
) : Event
```

```typescript
export type PlaybackWatchTimeEvent = {
    /** Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры. */
    playbackId: string;
    /** Продолжительность смотрения в секундах */
    duration: number;
    /** Позиция просмотра (playPosition на таймлайне в секундах) */
    position: number;
}
```

ökko

Кодогенерация (SQL)


```sql
--liquibase formatted sql

--changeset generator:1 contextFilter:clickhouse
CREATE TABLE IF NOT EXISTS kollector.playback_watch_time ON CLUSTER kollector
(
    `playback_id` String COMMENT 'Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры.',
    `duration` Int32 COMMENT 'Продолжительность смотрения в секундах',
    `position` Int32 COMMENT 'Позиция просмотра (playPosition на таймлайне в секундах)',
    `kollectorts` DateTime64(3) COMMENT 'Таймстемп попадания события в коллектор в миллисекундах'
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/{database}/{table}', '{replica}')
    PARTITION BY toYYYYMM(kollectorts)
    ORDER BY kollectorts;


--changeset generator:2 contextFilter:impala
CREATE TABLE IF NOT EXISTS `playback_watch_time`
(
    `playback_id` string COMMENT 'Уникальный идентификатор проигрывания контента для каждого конкретного пользователя. Выдается в том числе на трейлеры.',
    `duration` int COMMENT 'Продолжительность смотрения в секундах',
    `position` int COMMENT 'Позиция просмотра (playPosition на таймлайне в секундах)',
    `kollectorts` timestamp COMMENT 'Таймстемп попадания события в коллектор в миллисекундах'
)
PARTITIONED BY (year int, month int, day int)
COMMENT 'Heartbeat просмотра. Реальное время просмотра. Отправляется каждые 30 секунд, а также вместе с playerEvent bufferingStart|pause|seek|stop|complete|terminate|startAd\n'
WITH SERDEPROPERTIES ('serialization.format'='1')
LOCATION '/user/hive/warehouse/playback_watch_time';
```

ökko

Фиксируем техдолг


```
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.

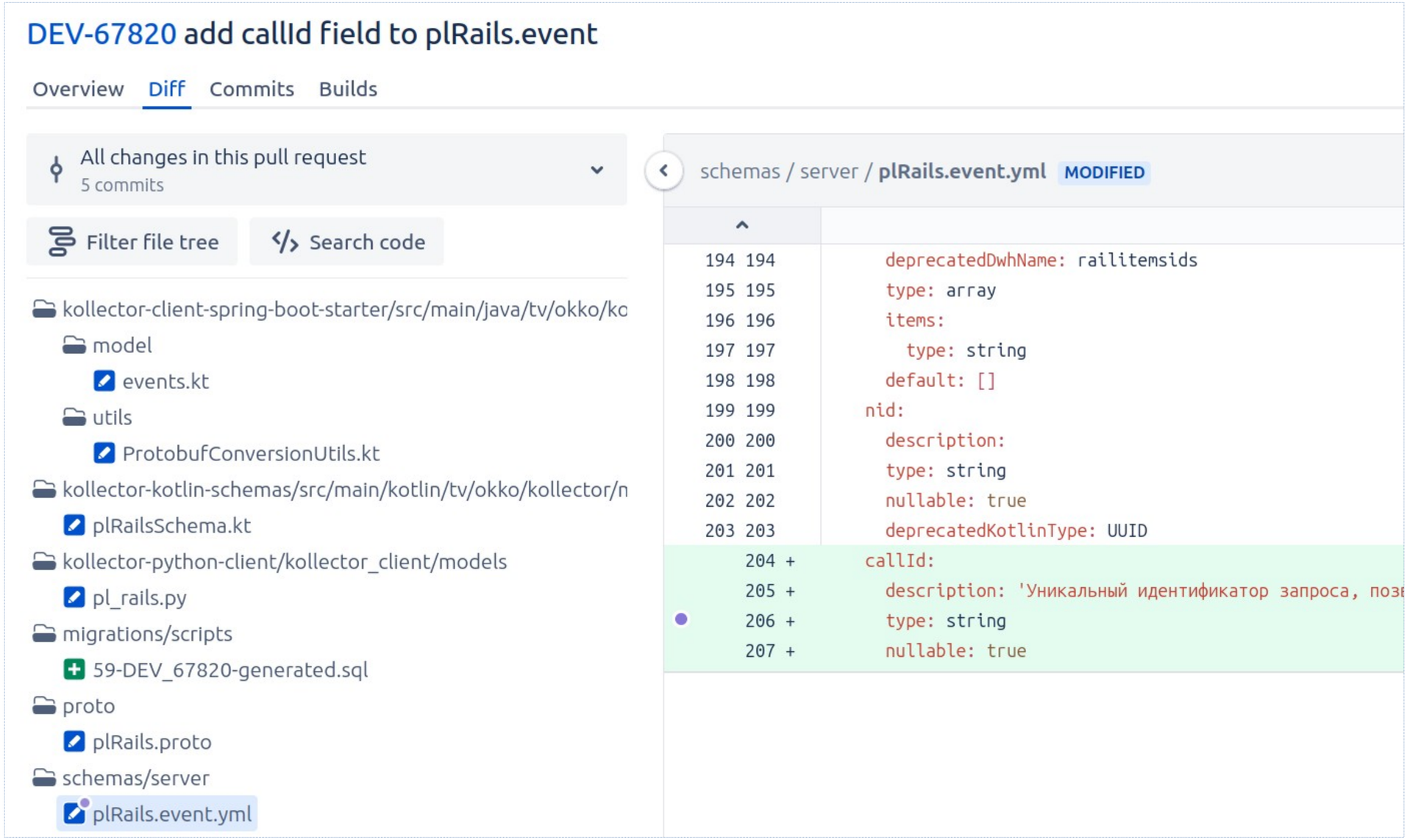
Флоу внесения изменений

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

Флоу внесения изменений



3. Повышаем гарантии

Okko

enum


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

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

ökko

conditional


```yaml
  duration:
    conditional:
      action: [completeAd, bufferingComplete, resume, ready]
    type: float
    description: Продолжительность процесса (отправляется вместе с каждым событием буферизации, паузы, рекламы) в секундах
  speed:
    conditional:
      action: [startWatch, speedChange]
    type: string
    description: Скорость воспроизведения
  volumeLevel:
```

```typescript
export type PlaybackPlayerEventActionProps =
    | { action: PlaybackPlayerEventAction.Ready } & {
        /** Продолжительность процесса (отправляется вместе с каждым событием буферизации, паузы, рекламы) в секундах */
        duration: number;
    }
    | { action: PlaybackPlayerEventAction.SpeedChange } & {
        /** Скорость воспроизведения */
        speed: string;
    }
    | { action: PlaybackPlayerEventAction.OrientationChange } & {
```

ökko

namedType

```yaml
# DeviceType.namedType.yml
description: тип устройства
type: enum
values:
  tv: телевизор
  bdp: Blue Ray Player
  mp: media player
  mob: мобильный телефон
  tbl: планшет
  web: браузер
  pc: PC-клиент
  vgc: video game console (PlayStation)
  stb: set top box
```

```yaml
  deviceType:
    description: тип устройства
    type: named
    namedType: DeviceType
```

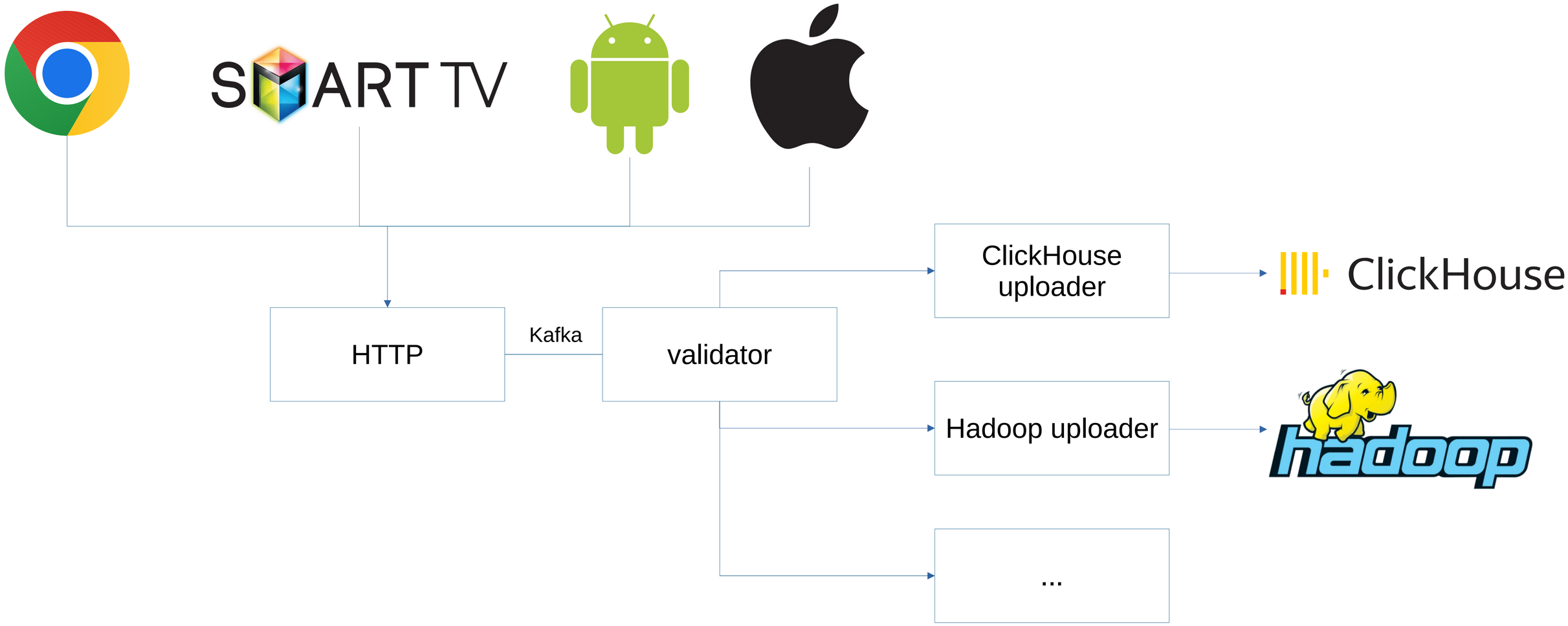
ökko

object


```yaml
# Screen.namedType.yml
description: Экран
type: object
properties:
  type:
    description: Тип экрана
    type: enum
    values:
      main: Главная
      auth: Раздел авторизации
      settings: Раздел настроек
      # ...

  auth:
    description: Экран раздела авторизации
    conditional:
      type: [auth]
    type: enum
    values:
      main: Главный экран (выбор способа входа)
      pin: Авторизация по пинкоду или QR
      resetPassword: Сброс пароля
      # ...

  settings:
    description: Экран раздела настроек
    conditional:
      type: [settings]
    type: enum
    values:
      menu: Меню "Профиль" (первый экран в приложениях)
      main: Основные настройки (контакты и др.)
      download: Настройки загрузки
      # ...
```

```yaml
# screenTransition.event.yml
description: Событие смены экрана
properties:
  from:
    description: С какого экрана выполнен переход
    type: named
    namedType: Screen

  to:
    description: На какой экран выполнен переход
    type: named
    namedType: Screen
```

Валидация



ökko

4. Перепридумываем фронтовые события

Okko

Старые фронтовые 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

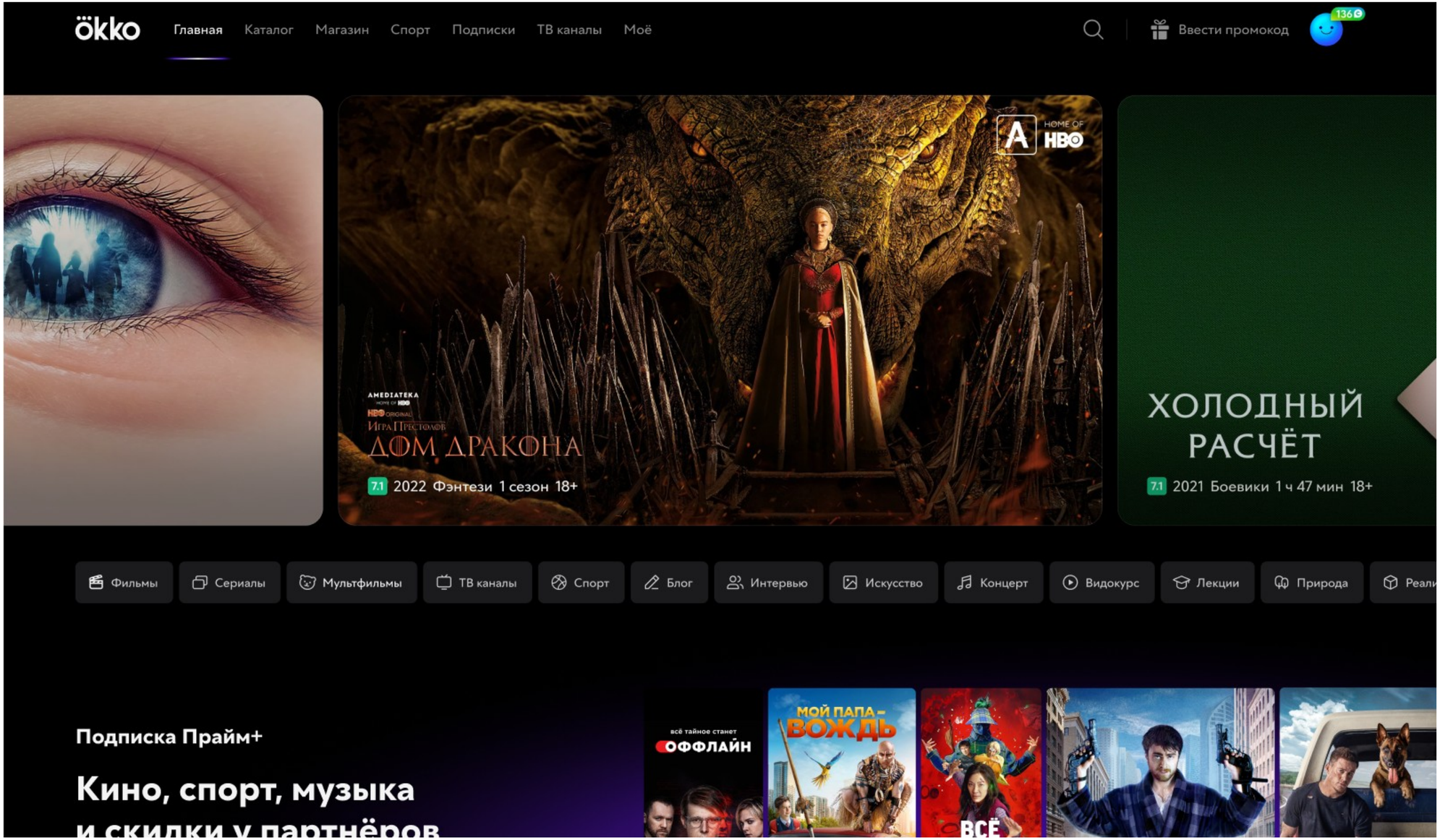
purchaseEvent - выбор продукта, интеракции в процессе покупки

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

settingEvent - изменения настроек

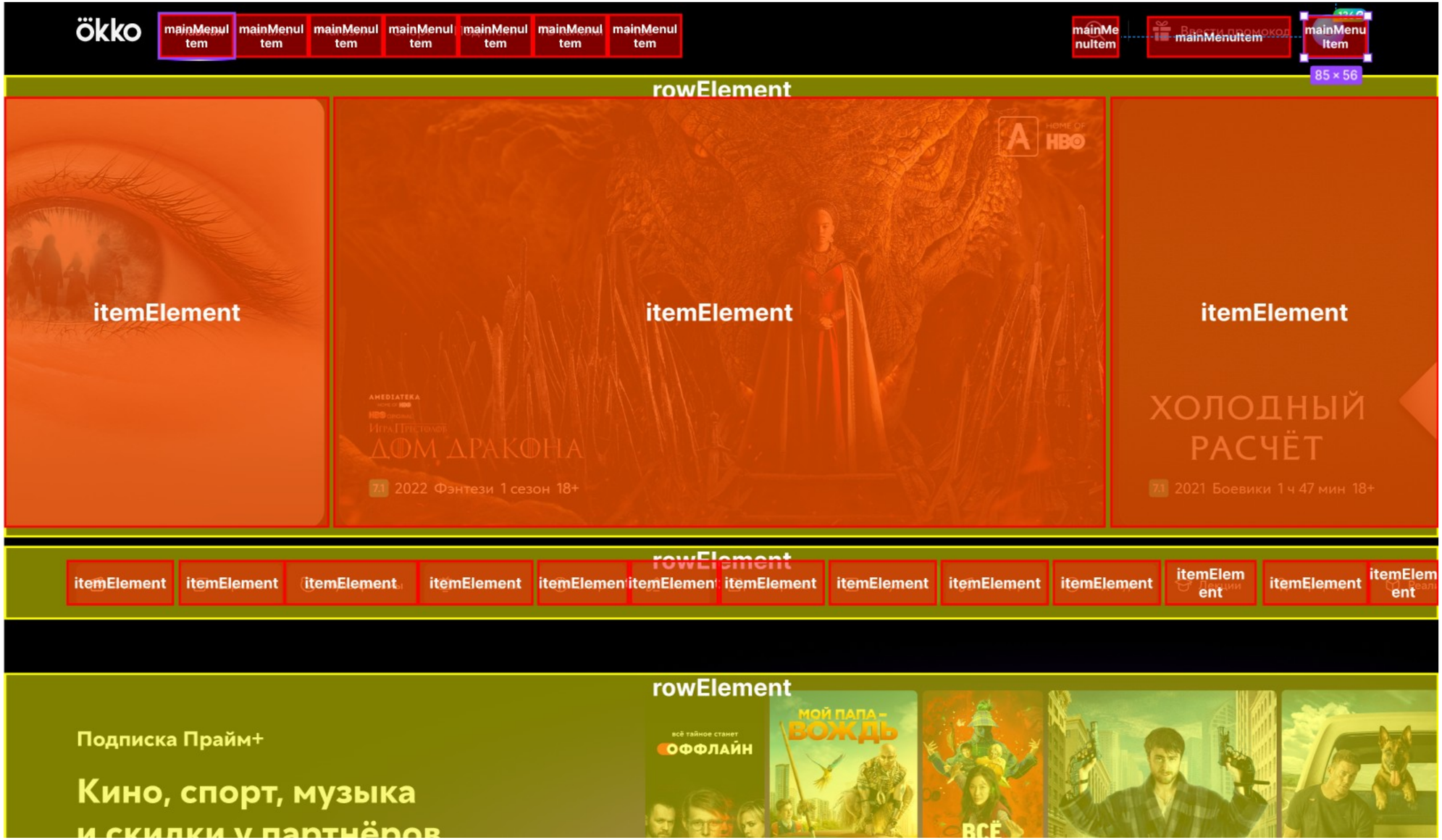
...

Блоковое логирование



ökko

Блоковое логирование



ökko

Событие blockInteraction


```
{
    _type: “blockInteraction”,

    action: "click",

    blockPath: ["rowElement", "itemElement"],

    rowElementIndex: 2,
    rowElementId: "featured",

    itemElementIndex: 3,
    itemElementId: "25b774cd-dd86-4542-8aeb-8d0ec99f74e4",
    itemElementExplanation: “genre”,

    // ...
}
```

ökko

Событие blockInteraction


```yaml
type: client
properties:
  action:
    description: Действие
    type: enum
    values:
      show: Показ блока
      click: Клик на блок
      focus: Фокус блока (только для управления с пульта)
      scroll: Скролл блока

  blockPath:
    description: Путь из родительских блоков до текущего элемента
    type: array
    items:
      type: enum
      values:
        rowElement: 'Динамический блок с uiType: рейл, грид, баннер и т.д.'
        itemElement: Элемент в рейле, гриде и т.д.
        button: Кнопка или кликабельный текст (например, заголовок рейла)
        mainMenuItem: Элемент главного меню
        # ...

  rowElementIndex:
    conditional:
      blockPath: [rowElement]
    description: Порядковый номер строки. Нумерация с 0.
    type: int16
  rowElementId:
    conditional:
      blockPath: [rowElement]
    description: UUID или один из айдишников динамических рейлов
    type: string

  itemElementIndex:
    conditional:
      blockPath: [itemElement]
    description: Порядковый номер элемента в рейле. Нумерация с 0.
    type: int16
  itemElementId:
    conditional:
      blockPath: [itemElement]
    description: Идентификатор элемента
    type: string
  itemElementExplanation:
    conditional:
      blockPath: [itemElement]
    description: Причина рекомендации
    type: string
    nullable: true
```

ökko

Итоги

1. Формализованные схемы позволили снизить влияние человеческого фактора.

2. Блоковое логирование упрощает покрытие новой функциональности.

Спасибо за внимание!

Okko


ökko