663,77
Рейтинг

«Лаборатория Касперского»
Ловим вирусы, исследуем угрозы, спасаем мир


xoxol_89 12 сентября в 09:47 Разработка

Анатомия аналитики от Google

Разработка под Android, Разработка мобильных приложений, Java, Блог компании «Лаборатория Касперского»

Всем привет!
Мы — разработчики (гордо звучит, не правда ли?), и мы активно пилим новые фичи, правим баги и стараемся сделать наш продукт лучше. Но чтобы понять, а как именно пользователь использует наш продукт, какие фишки продукта ему по душе, а какие — не оче мы используем аналитику. Есть много разных средств, но в этой статье я бы хотел поговорить именно об аналитике от Google, кото активно развивается и меняется. Старого часового по имени Google Analytics сменяет новый боец — Google Analytics for Firebase (в девичестве — Firebase Analytics).
Уже даже в названиях вы можете уловить этот ветер перемен. А ветер перемен всегда порождает некоторый информационный вак в который попадают разного рода слухи, далеко не всегда достоверные при этом.
Поэтому давайте попробуем разобраться подробно, а что сейчас с этой аналитикой, чем пользоваться-то в итоге. И как вообще да жить.
Если про Google Analytics информации довольно много, и она систематизирована (чего только стоит этот ресурс, идеальная справ у Google Analytics for Firebase типичная болезнь молодого и активно развивающегося продукта — информации мало, она разрозне иногда даже противоречива. И я в свое время потратил немало сил и времени, чтобы разобраться, что к чему.
Собственно главная цель данной статьи — это систематизация знаний и нынешнего состояния Google Analytics for Firebase. Некото «дорожная карта» Google Analytics for Firebase.
Уверен, данная «карта» сэкономит вам прилично времени и нервов =)

Самый главный миф. Google Analytics всё

Начну все-таки с самого горячего.
Мне кажется, что данный слух идет с самого появления Firebase Analytics. И с одной стороны, это логично, зачем «Гуглу» два сред аналитики. Но Google Analytics (будем именовать GA) и Google Analytics for Firebase (по старинке назовем FA) — это две аналитики разными концепциями и подходами, про которые мы поговорим чуть ниже.
GA никуда не денется и не пропадет (по крайней мере сейчас), а также не будет кем-то поглощен. Это как информация от представителей москвовского офиса Гугла, так и инсайды от самих разработчиков.
Фанаты GA могут спать спокойно… пока что. Но кто знает, что будет дальше. Поэтому я настоятельно рекомендую продолжить чте

GA vs FA. Общая концепция

FA — это аналитика с совершенно другой концепцией и философией. Она является event based и предназначена исключительно дл мобилки. Тогда как GA — screen-based и сначала была для веба, а уж потом ее допилили для мобилки.
GA структурирована вокруг иерархичных событий с одним значением, FA — больше о записи одного события с большим количеством параметров (пар «ключ-значение»).
Эти аналитики очень разные. И поэтому они не могут быть взаимозаменяемыми.
Миграции с одной на другую не предусматривается. Но «Гугл» работает над определенной совместимостью этих аналитик, о чем тоже поговорим чуть позже.

GA vs FA. События

Коль уж мы затронули тему событий. В плане осмысления «события» GA и FA действительно очень разные. И это особенно заметн примере.
Допустим, ваше приложение — это игра. По окончании игры вы хотите послать статистику, как в итоге сыграл пользователь. И вы х узнать у пользователя общий счет, количество убитых врагов и количество пройденных раундов.
В GA все это будет выглядеть примерно вот так:

```
// total score
mTracker = googleAnalytics.newTracker(R.xml.tracker_global_config);
HitBuilders.EventBuilder builder = new HitBuilders.EventBuilder()
```

```
        .setCategory("gameOver")
        .setAction("totalScore")
        .setLabel("")
        .setValue(gameStats.getTotalScore());
mTracker.send(builder.build());
// enemies beaten
mTracker = googleAnalytics.newTracker(R.xml.tracker_global_config);
HitBuilders.EventBuilder builder = new HitBuilders.EventBuilder()
        .setCategory("gameOver")
        .setAction("enemiesBeaten")
        .setLabel("")
        .setValue(gameStats.getEnemiesBeaten());
mTracker.send(builder.build());
// roundsSurvived
mTracker = googleAnalytics.newTracker(R.xml.tracker_global_config);
HitBuilders.EventBuilder builder = new HitBuilders.EventBuilder()
        .setCategory("gameOver")
        .setAction("roundsSurvived")
        .setLabel("")
        .setValue(gameStats.getRoundsSurvived());
mTracker.send(builder.build());
```

В GA каждое событие по сути представляет собой иерархию параметров:
category -> action -> label -> value
И в самой консоли вы могли наблюдать данную иерархию параметров. Собственно при придумывании событий, которые вы бы хот отслеживать, вы должны были руководствоваться данной парадигмой. Также в консоли можно строить различные фильтры по данн параметрам.
Но в GA в плане событий есть небольшой минус. Если вы хотите навешать событию дополнительные параметры, помимо вышеназванных, вот тут приходится танцевать вокруг "category" -> "action" -> "label" -> "value", придумывать новые формулировки прочее. Неудобно. По крайней мере так было раньше.

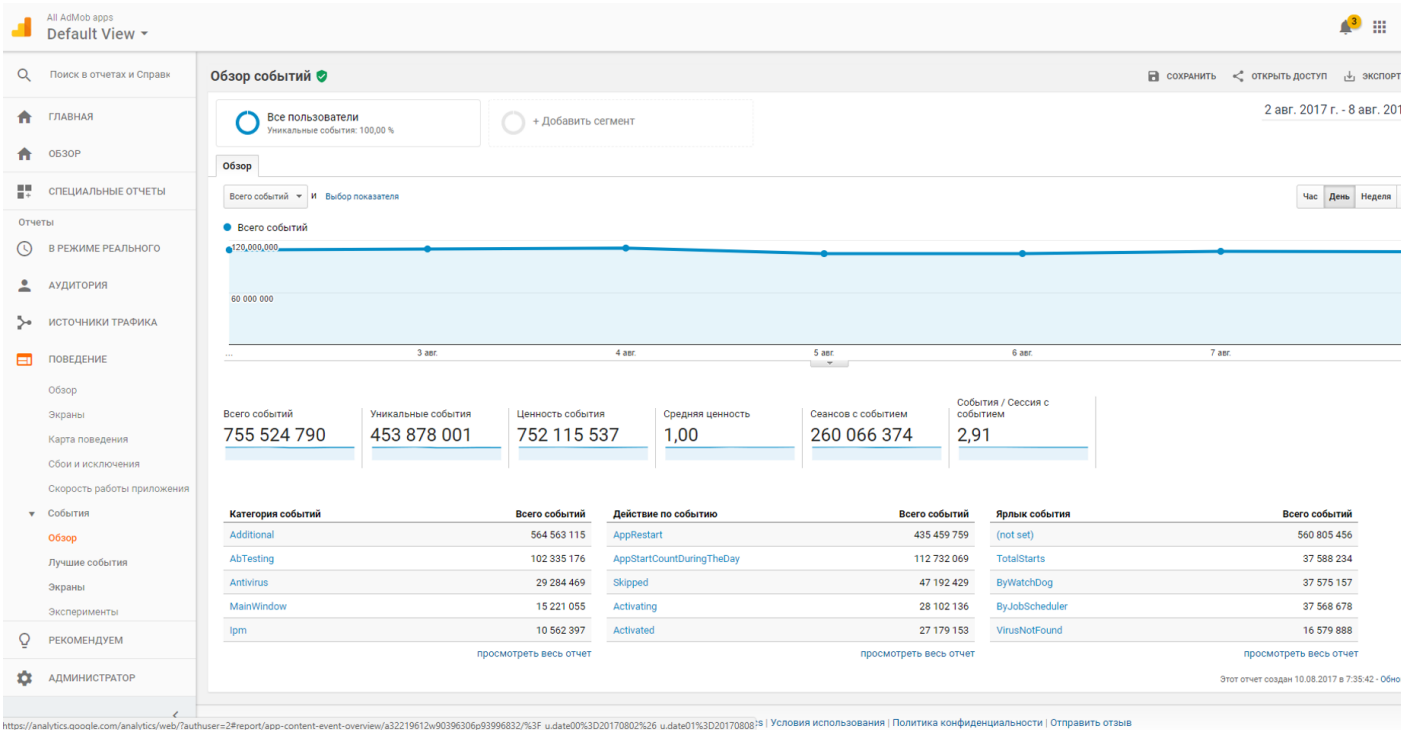
А теперь посмотрим, как можно данную статистику обыграть с FA:

```
Bundle params = new Bundle();
params.putLong("totalScore", gameStats.getTotalScore());
params.putLong("enemiesBeaten", gameStats.getEnemiesBeaten());
params.putLong("roundsSurvived", gameStats.getRoundSurvived());
mFirebaseAnalytics.logEvent("game_over", params);
```

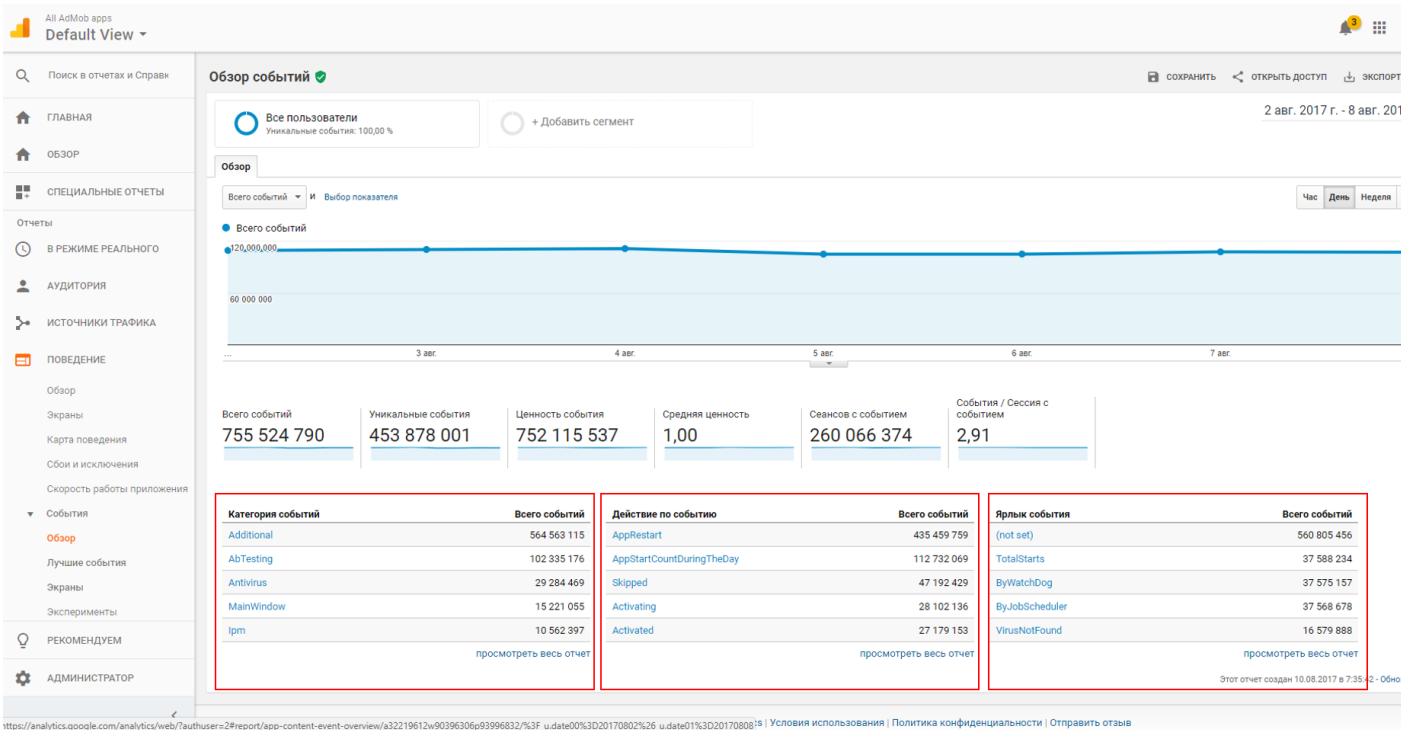
Как видите, вместо трех событий мы отправляем одно, что более логично и удобно. Про «события» в FA мы поговорим подробнее ниже.

GA vs FA. Консоль

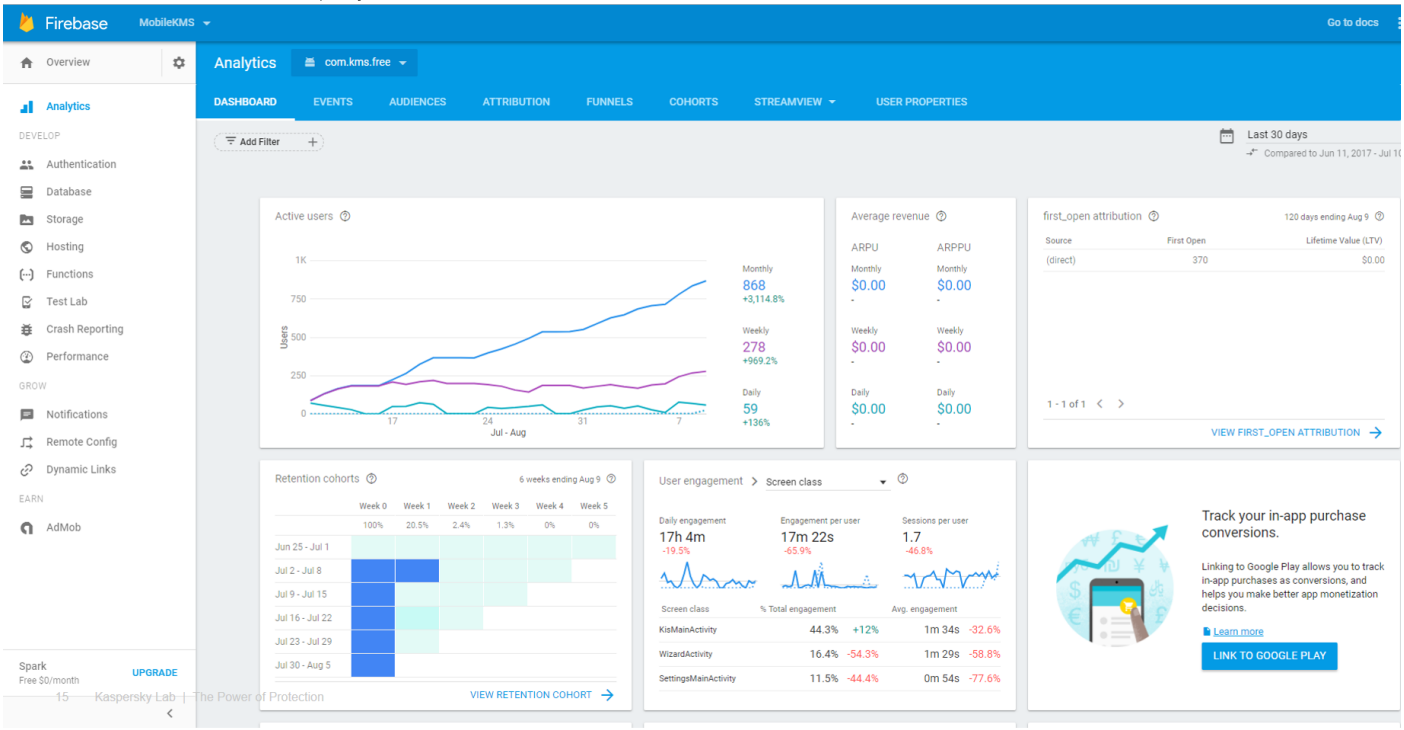
Второе, чем сильно отличаются аналитики, это — консоль.
Вот как выглядит консоль в GA (*картинка кликабельна*):



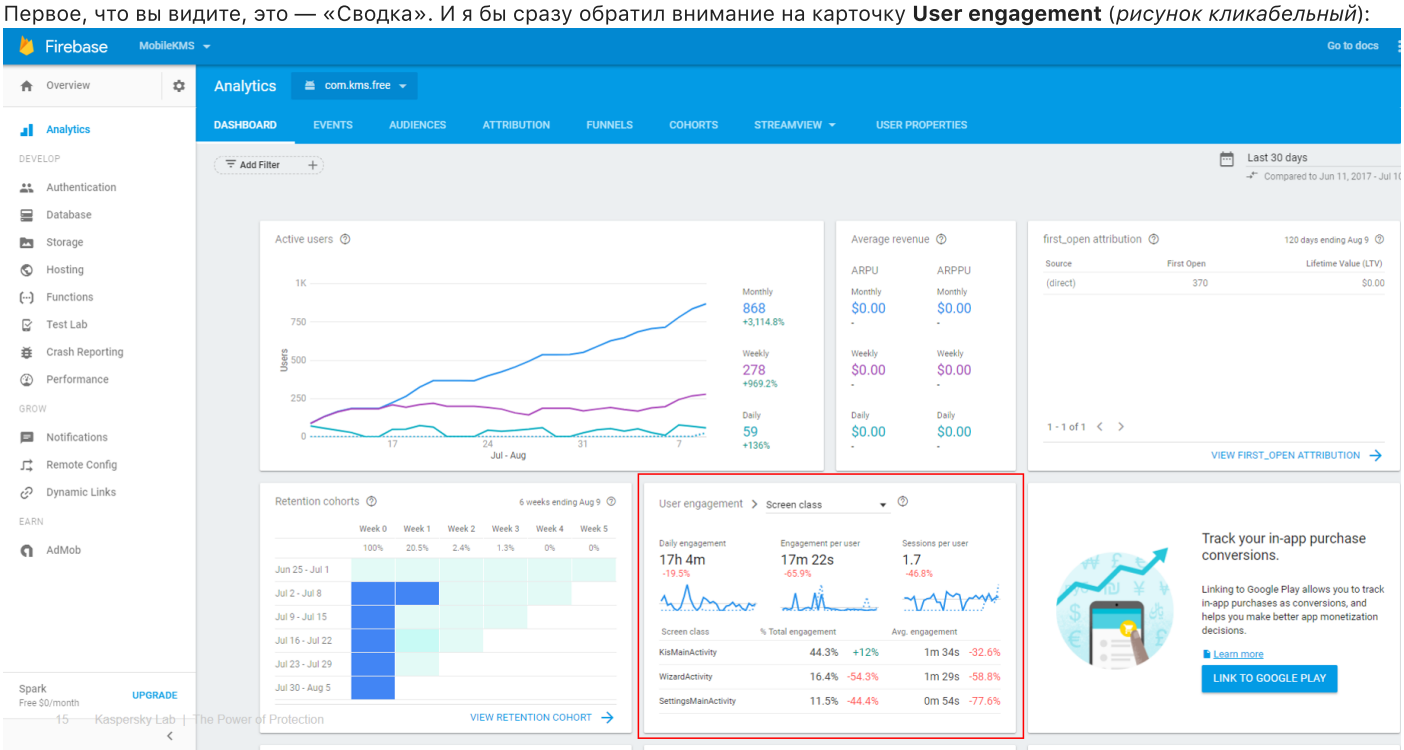
«События» спрятаны глубоко во вкладке «Поведение» слева. Но в стандартном отчете сразу идет разбивка на **Category, Action, L** (*рисунок кликабельный*):



Вот так выглядит консоль FA (*рисунок кликабельный*):



Первое, что вы видите, это — «Сводка». И я бы сразу обратил внимание на карточку **User engagement** (*рисунок кликабельный*):



Наконец-то в FA-консоль добавили нормальный просмотр экранов. До мая мы жили без этого. То есть событие «user engagement» отсылалось, но в консоли его никак нельзя было посмотреть. Это было ужасно. И это, возможно, одна из причин, почему никто не х переходить на FA.

Как еще можно заметить, вкладка **Events** идет сразу за **Dashboard**, что еще раз подтверждает — FA заточена на работу с событиям консоли мы также вернемся чуть позже, а сейчас я предлагаю погрузиться в эту обширную тему «Событий» в FA.

События FA

Давайте сразу взглянем на код:

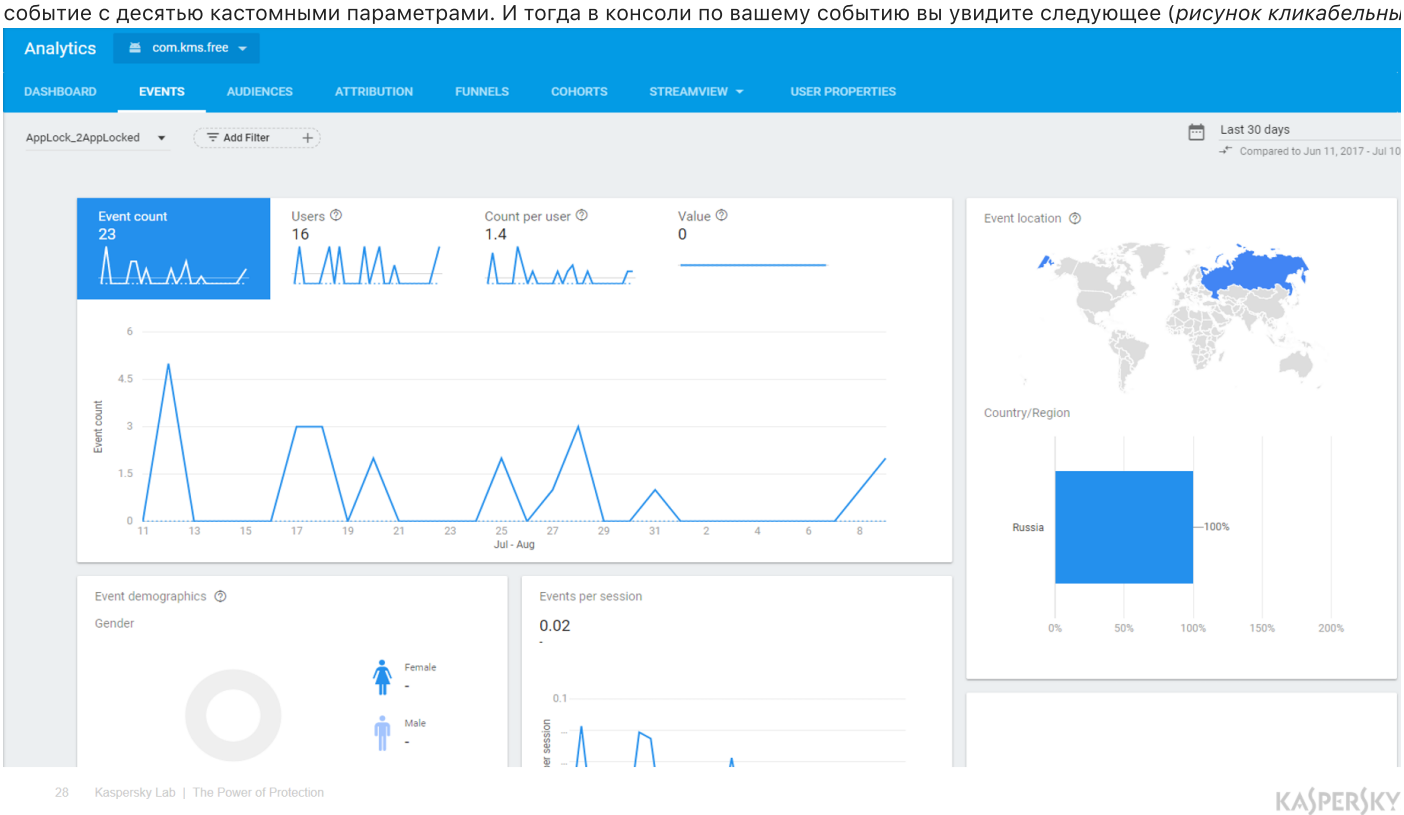
```
Bundle bundle = new Bundle();
bundle.putString(FirebaseAnalytics.Param.ITEM_ID, id);
bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, name);
```

```
bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "image");
mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
```

Вы можете отправлять до 500 **различных** типов событий в своем приложении, включая предустановленные (`FirebaseAnalytics.Event.SELECT_CONTENT` — это предустановленное, но вы можете задавать и свои типы). Общее количество отправляемых событий не лимитировано (источник).
К каждому событию можно прикреплять **до 25 параметров** (то, что идет в `Bundle`). Параметры также есть предопределенные, но н не запрещает вам задавать кастомные параметры. Описано здесь.
Типы событий и параметров — это обычные `String`.
Названия событий и параметров чувствительны к регистру. Одинаковые события должны совпадать по типу и параметрам.
Кроме того, есть события, которые отправляются по умолчанию. Весь список автоматически отправляемых событий с описанием приведен по данной ссылке. Как вы можете заметить, там много действительно интересных событий, которые раньше нам не представлялось возможным получить. Круто!
Также по приведенной выше ссылке вы можете прочитать, какие предопределенные события и параметры можно выбрать для определенных событий.

События FA. «Гладко было на бумаге, да забыли про овраги»

Вы обратили внимание, что как-то подозрительно много говорится про предопределенные названия событий и параметров. И в показательных примерах обычно посылаются именно такие события с параметрами. А это ведь неспроста. Допустим, вы посылает событие с десятью кастомными параметрами. И тогда в консоли по вашему событию вы увидите следующее (*рисунок кликабельны*

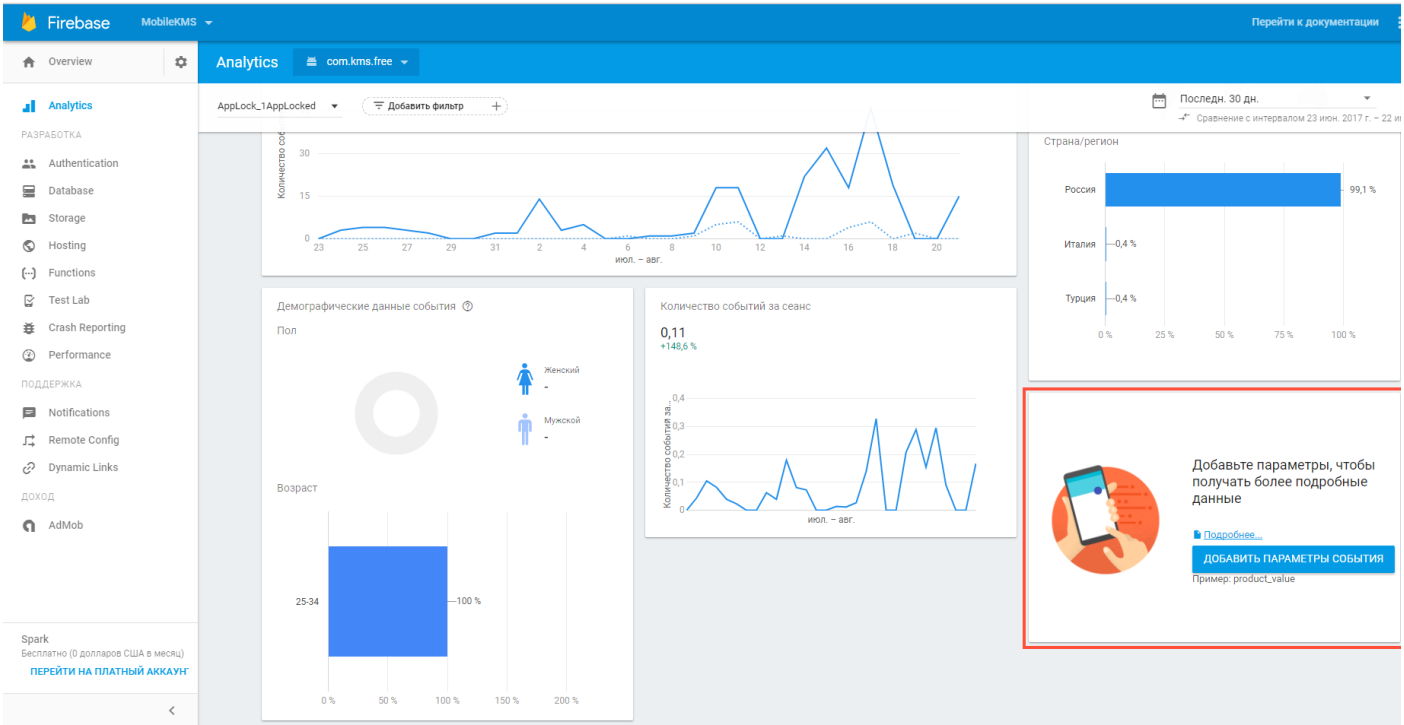


«Но где же все мои параметры?» — спросите вы. А нет их на консоли, вот так вот.
Дело в том, что все красивые графики и прочее строятся, только если вы используете предопределенные названия. Используете св «кастомное», ничегошеньки не увидите. Только «количество событий» да «количество пользователей».
И до I/O 17 это было прям страшной болью. Графики можно было строить, играя, например, с параметром Value, как в этой статье. это, конечно, все не то.

И тут, конечно же, пора бы вспомнить про GA, где все для людей, строй всякие там фильтры по чему угодно и сколько душе угодно Но и тут маленькая засада. Стандартные отчеты — да, стройте без проблем. Но в большинстве случаев нам нужны и кастомные отч Например, добавить Secondary dimension, чтобы отсортировать события по моделям устройств. И вот тут всплывает страшное слов **«Sampling»**.
В зависимости от отчета алгоритм сэмплирования в GA различается. То, как конкретно считается семпл для каждого отчёта, «Гугл раскрывает, но в целом все практики уже известны. Обычно это **hi-based-сэмплирование** или **cookie-based-сэмплирование**. В первом случае берется рандомная выборка из всех записей (событий, просмотров и т.д.), во втором — рандомная выборка по всем пользователям (размеченным кукам или gaid/idfa, если это мобильное приложение).
Поэтому нельзя достоверно говорить об ошибке по каждому полю.
По практике говорят, что при выборке больше 5% ошибка в абсолютных числах в отчетах по событиям составляла меньше 2,5%.
За предоставление информации о сэмплировании хочу выразить благодарность **Александру Сергееву** из «Яндекса».

События FA. Продолжение

Да уж. Все непросто с этими «Событиями». И на самом деле FA идет навстречу пожеланиям простого люда.
Во-первых, никакого сэмплинга в FA нет. Там доступны все данные.
И это очень круто, так как стоимость Google Analytics 360 (платной версии GA без сэмплинга) весьма немаленькая. А в FA вы може
ваши данные выгрузить в BigQuery и там делать с ними все что угодно.
Во-вторых, после I/O 17 появилась возможность строить отчеты и по кастомным параметрам.
Вам прямо на экране конкретного события предлагается зарегистрировать кастомные параметры (*рисунок кликабельный*):

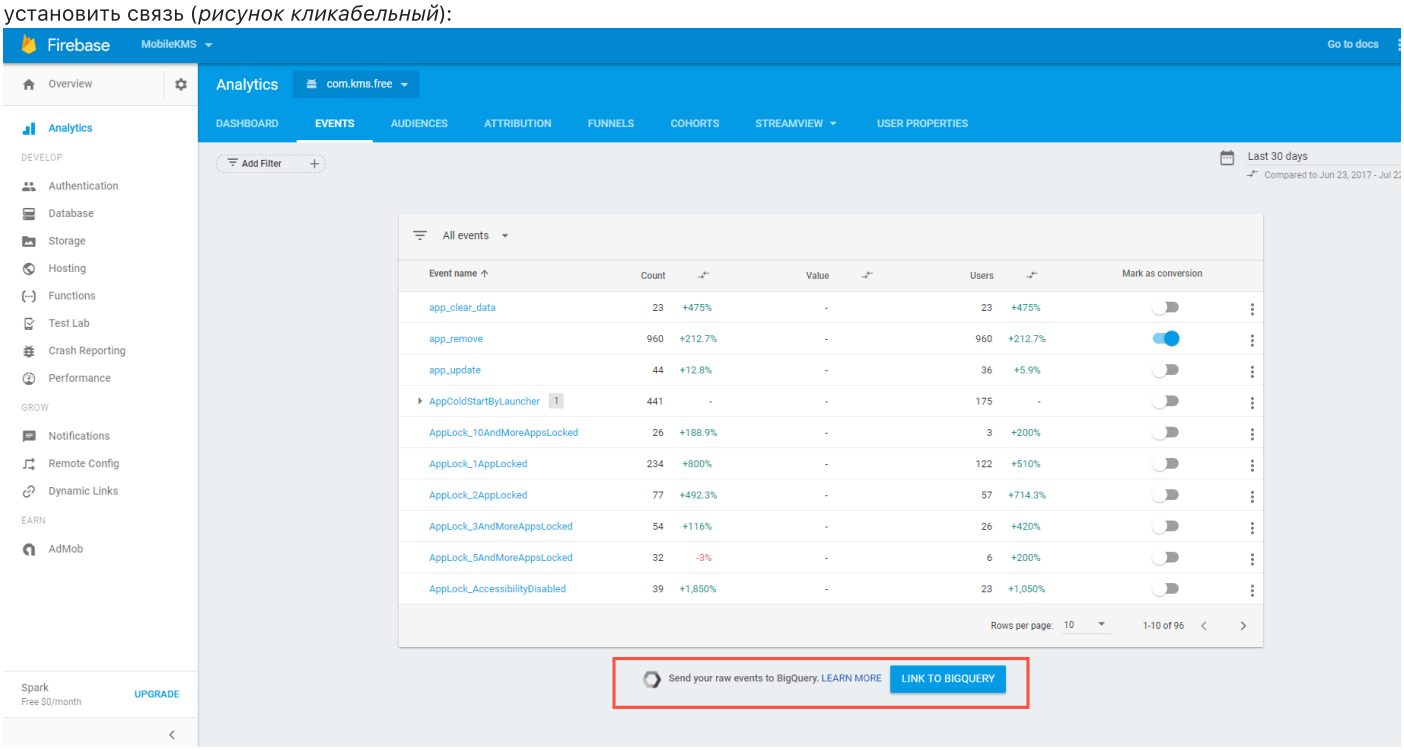


Но учтите, что **всего для данного приложения** вы можете зарегистрировать до **50** таких параметров (**10 текстовых и 40 числовь**
Пробовал лайфхак для обхода данного ограничения: регистрировал для разных событий кастомные параметры с одинаковыми именами. Не помогло, все равно делается «плюс один».
Кроме того, если вы ожидаете увидеть сразу готовые отчеты, спешу вас разочаровать. Отчеты строятся накопительным образом. Допустим, есть у вас «event_1» с кастомным параметром «custom_1», для которого вы хотите построить отчет. В консоли вы настро чтобы строился данный отчет в момент времени X. Так вот в отчет попадут все события «event_1», которые придут после момента времени X. А все «event_1» до момента X, увы, не будут обработаны. Так что будьте внимательны.
То есть вроде бы лучше стало, но не сильно. Что еще обидно, вы не можете эти отчеты как-то совмещать друг с другом. Но, пожалу мы слишком многого хотим от консоли. Если уж вы хотите делать с данными все что угодно, то добро пожаловать в удивительный BigQuery. Давайте немного приоткроем эту завесу таинства данных.

BigQuery

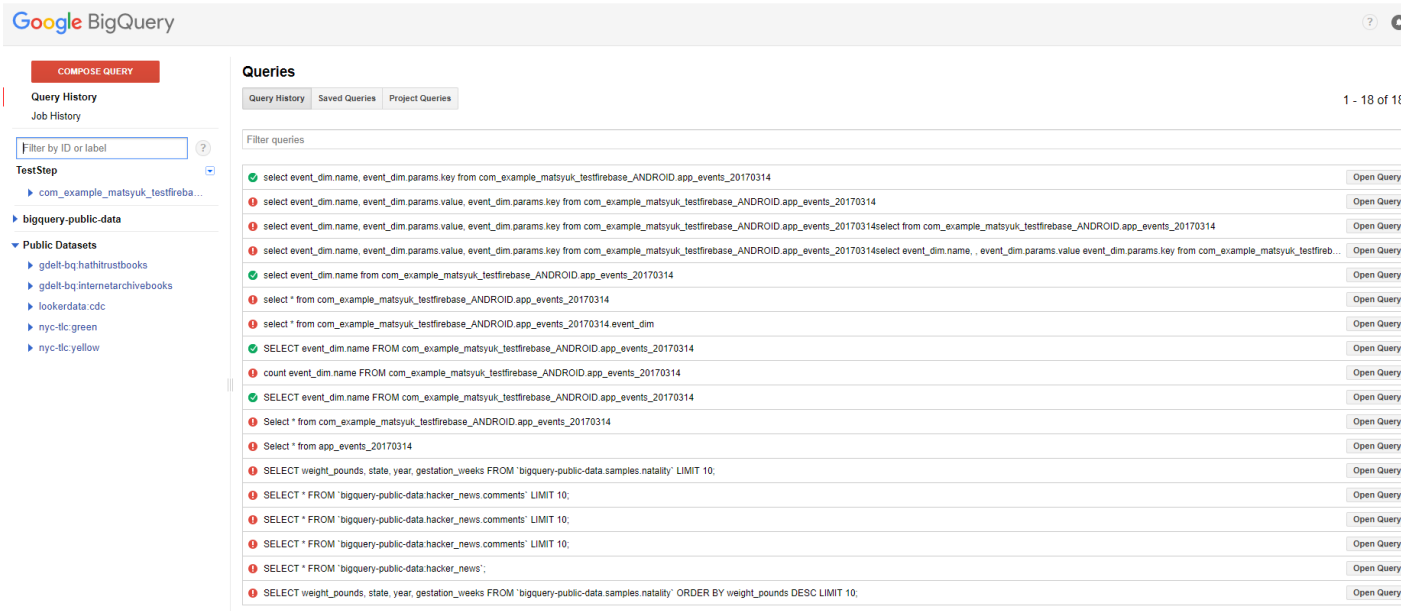
BigQuery — это вообще немного другая галактика.
С BigQuery можно было работать и через GA, но только если у вас premium-режим. В FA же вам прямо во вкладке Events предлагае

установить связь (*рисунок кликабельный*):



Google говорит: «Мы дарим вам машину, но за бензин платите вы». С тарифными планами можно ознакомиться здесь, а еще лучше здесь. Но поверьте, чтобы просто попробовать, вам вполне достаточно будет бесплатных лимитов тарифа **Blaze**. Да и даже при раб боевыми продуктами, судя по отзывам товарищей, плата весьма условной получается.

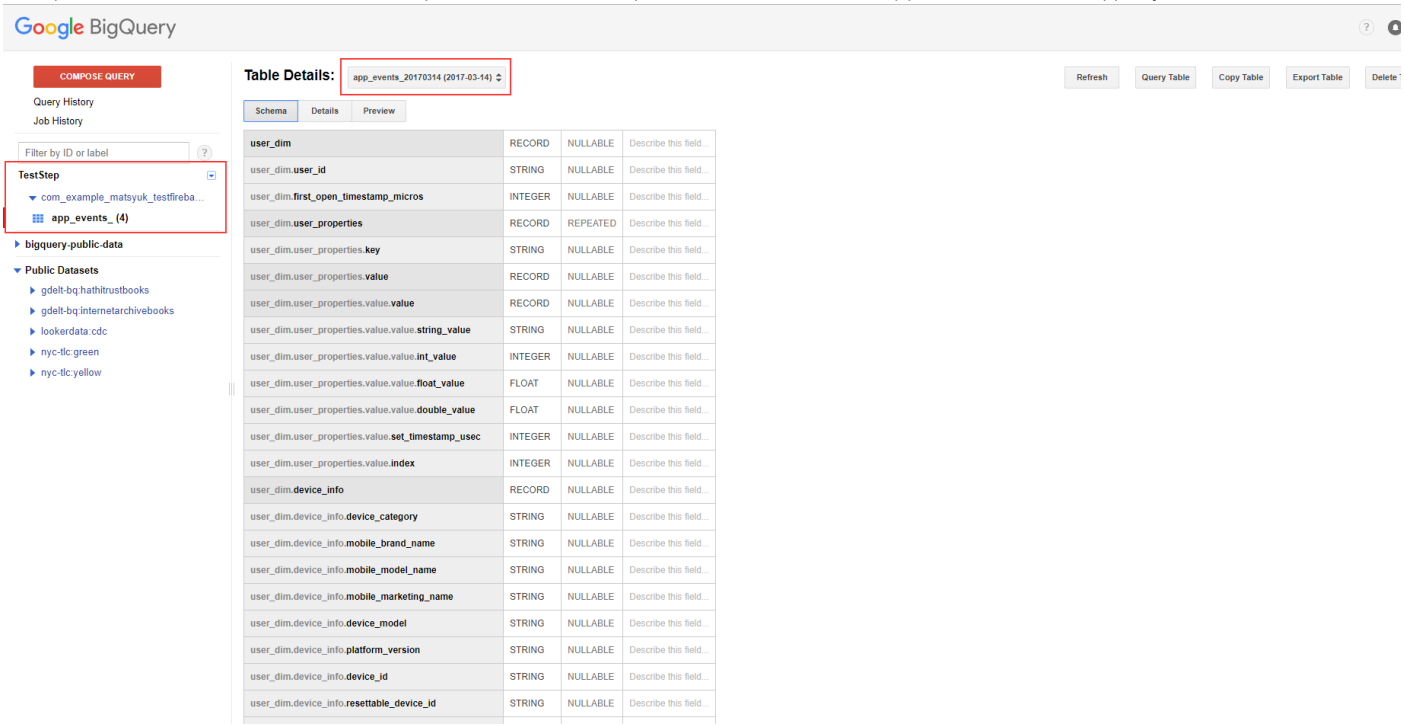
Итак, начнем знакомство. Вот так выглядит консоль BigQuery (*рисунок кликабельный*):



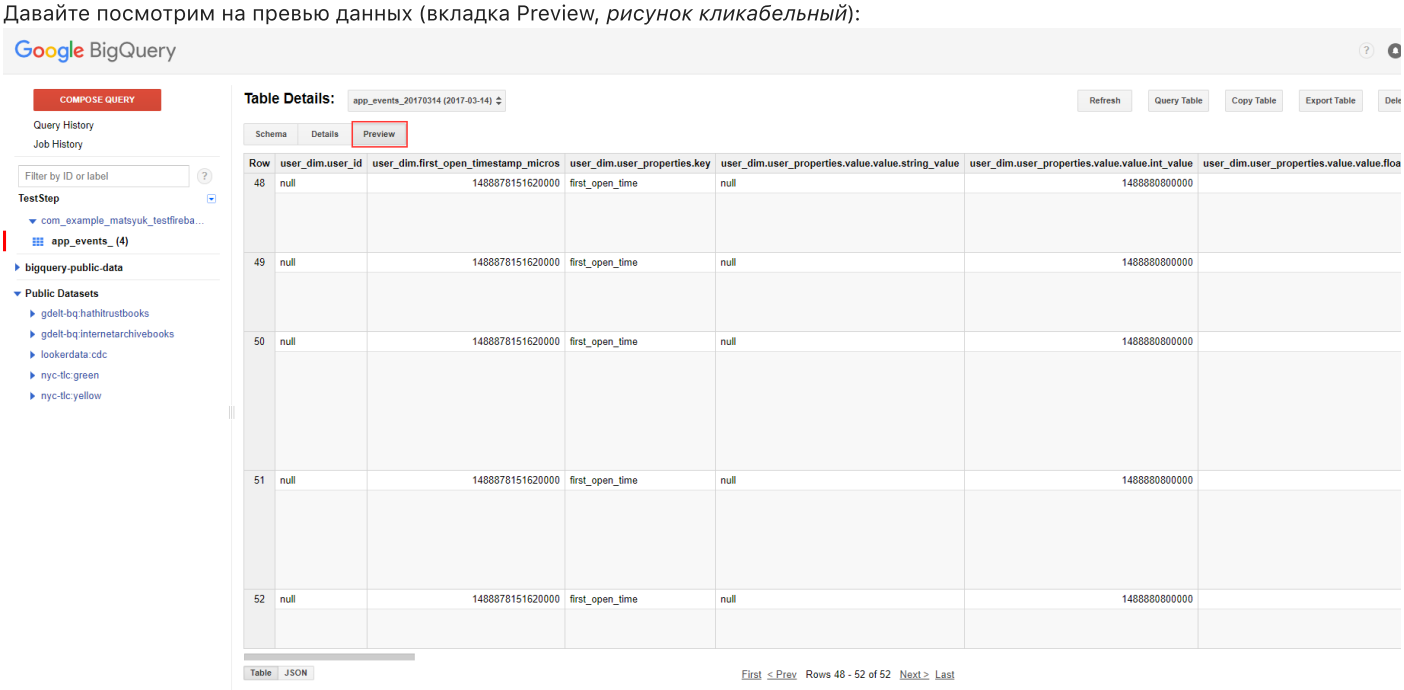
В левом меню представлен список доступных данных. Например, **TestStep** — это мой тестовый проект с одним приложением в сос
А **bigquery-public-data** и **Public Datasets** — это, как можно догадаться, публичные данные, с которыми вы можете поэкспериментировать и на которых можете потренироваться в написании запросов.

Справа же вы видите список запросов, как успешных, так и не очень.

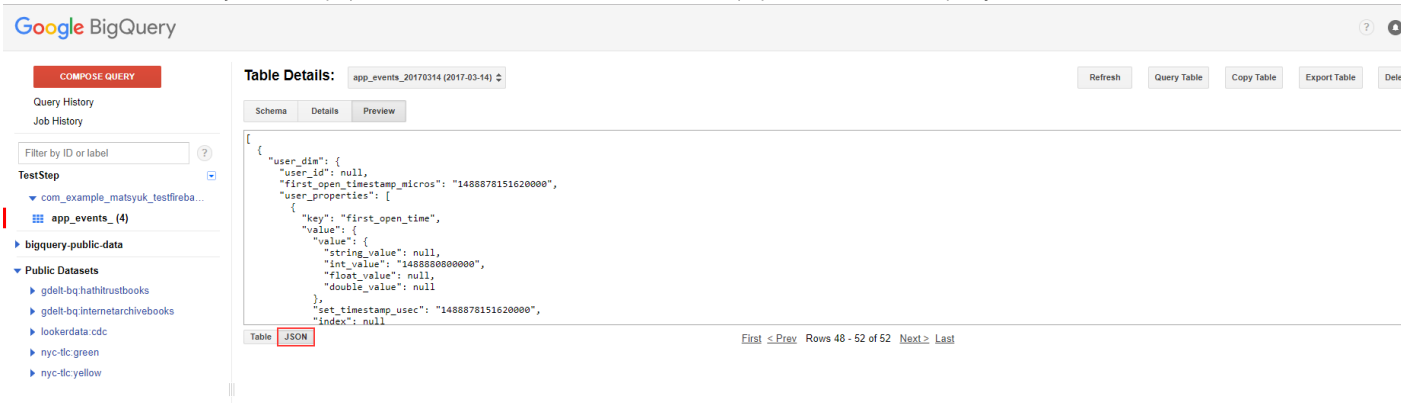
Теперь взглянем на данные тестового приложения за 14 марта 2017 года (таблица app_events_20170314, *рисунок кликабельный*):



В таблицу я перебросил все данные за сутки (52 события). Общий состав таблицы представлен перед вами. Как видно, тут каждое событие описывается максимально полно, включая все **properties**, о которых речь будет чуть ниже.
Давайте посмотрим на превью данных (вкладка Preview, *рисунок кликабельный*):



Табличный вид с ходу малоинформативен. Намного более понятная форма — это JSON (*рисунок кликабельный*):



И тогда наше событие представляется в полном виде. В UI почему-то нельзя расширить окно показа json, поэтому приведу полный последних пяти событий отдельно:

▾ 5 событий в BigQuery


```
[
  {
    "user_dim": {
      "user_id": null,
      "first_open_timestamp_micros": "1488878151620000",
      "user_properties": [
        {
          "key": "first_open_time",
          "value": {
            "value": {
              "string_value": null,
              "int_value": "1488880800000",
              "float_value": null,
              "double_value": null
            },
            "set_timestamp_usec": "1488878151620000",
            "index": null
          }
        }
      ],
      "device_info": {
        "device_category": "mobile",
        "mobile_brand_name": null,
        "mobile_model_name": null,
        "mobile_marketing_name": null,
        "device_model": "507SH",
        "platform_version": "6.0.1",
        "device_id": null,
        "resettable_device_id": null,
        "user_default_language": "ru-ru",
        "device_time_zone_offset_seconds": "10800",
        "limited_ad_tracking": "false"
      },
      "geo_info": {
        "continent": "Europe",
        "country": "Russia",
        "region": "Moscow",
        "city": "Moscow"
      },
      "app_info": {
        "app_version": "1.0",
        "app_instance_id": "d0c587de4d5804ddc1d34f8d54b981f9",
        "app_store": "manual_install",
        "app_platform": "ANDROID",
        "app_id": "com.example.matsyuk.testfirebase"
      },
      "traffic_source": null,
      "bundle_info": {
        "bundle_sequence_id": "65",
        "server_timestamp_offset_micros": "-496748"
      },
      "ltv_info": null
    },
    "event_dim": [
      {
        "date": "20170314",
        "name": "user_engagement",
        "params": [
          {
            "key": "firebase_screen_class",
            "value": {
              "string_value": "SecondActivity",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
```

```
            "key": "firebase_event_origin",
            "value": {
              "string_value": "auto",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "firebase_screen_id",
            "value": {
              "string_value": null,
              "int_value": "1109587836504693342",
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "engagement_time_msec",
            "value": {
              "string_value": null,
              "int_value": "4424",
              "float_value": null,
              "double_value": null
            }
          }
        ],
        "timestamp_micros": "1489478210462000",
        "previous_timestamp_micros": "1489478205970000",
        "value_in_usd": null
      }
    ]
  },
  {
    "user_dim": {
      "user_id": null,
      "first_open_timestamp_micros": "1488878151620000",
      "user_properties": [
        {
          "key": "first_open_time",
          "value": {
            "value": {
              "string_value": null,
              "int_value": "1488880800000",
              "float_value": null,
              "double_value": null
            },
            "set_timestamp_usec": "1488878151620000",
            "index": null
          }
        }
      ],
      "device_info": {
        "device_category": "mobile",
        "mobile_brand_name": null,
        "mobile_model_name": null,
        "mobile_marketing_name": null,
        "device_model": "507SH",
        "platform_version": "6.0.1",
        "device_id": null,
        "resettable_device_id": null,
        "user_default_language": "ru-ru",
        "device_time_zone_offset_seconds": "10800",
        "limited_ad_tracking": "false"
      },
      "geo_info": {
        "continent": "Europe",
        "country": "Russia",
        "region": "Moscow",
        "city": "Moscow"
      },
      "app_info": {
        "app_version": "1.0",
        "app_instance_id": "d0c587de4d5804ddc1d34f8d54b981f9",
```

```
        "app_store": "manual_install",
        "app_platform": "ANDROID",
        "app_id": "com.example.matsyuk.testfirebase"
      },
      "traffic_source": null,
      "bundle_info": {
        "bundle_sequence_id": "64",
        "server_timestamp_offset_micros": "-515257"
      },
      "ltv_info": null
    },
    "event_dim": [
      {
        "date": "20170314",
        "name": "user_engagement",
        "params": [
          {
            "key": "firebase_screen_class",
            "value": {
              "string_value": "MainActivity",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "firebase_event_origin",
            "value": {
              "string_value": "auto",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "firebase_screen_id",
            "value": {
              "string_value": null,
              "int_value": "1109587836504693341",
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "engagement_time_msec",
            "value": {
              "string_value": null,
              "int_value": "17278",
              "float_value": null,
              "double_value": null
            }
          }
        ],
        "timestamp_micros": "1489478205970000",
        "previous_timestamp_micros": "1489153178047000",
        "value_in_usd": null
      }
    ]
  },
  {
    "user_dim": {
      "user_id": null,
      "first_open_timestamp_micros": "1488878151620000",
      "user_properties": [
        {
          "key": "first_open_time",
          "value": {
            "value": {
              "string_value": null,
              "int_value": "1488880800000",
              "float_value": null,
              "double_value": null
            },
            "set_timestamp_usec": "1488878151620000",
```

```
            "index": null
          }
        }
      ],
      "device_info": {
        "device_category": "mobile",
        "mobile_brand_name": null,
        "mobile_model_name": null,
        "mobile_marketing_name": null,
        "device_model": "507SH",
        "platform_version": "6.0.1",
        "device_id": null,
        "resettable_device_id": null,
        "user_default_language": "ru-ru",
        "device_time_zone_offset_seconds": "10800",
        "limited_ad_tracking": "false"
      },
      "geo_info": {
        "continent": "Europe",
        "country": "Russia",
        "region": "Moscow",
        "city": "Moscow"
      },
      "app_info": {
        "app_version": "1.0",
        "app_instance_id": "d0c587de4d5804ddc1d34f8d54b981f9",
        "app_store": "manual_install",
        "app_platform": "ANDROID",
        "app_id": "com.example.matsyuk.testfirebase"
      },
      "traffic_source": null,
      "bundle_info": {
        "bundle_sequence_id": "63",
        "server_timestamp_offset_micros": "-500210"
      },
      "ltv_info": null
    },
    "event_dim": [
      {
        "date": "20170314",
        "name": "ga_event",
        "params": [
          {
            "key": "label",
            "value": {
              "string_value": "label1",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "firebase_screen_class",
            "value": {
              "string_value": "MainActivity",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "action",
            "value": {
              "string_value": "action1",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "firebase_event_origin",
            "value": {
              "string_value": "app",
              "int_value": null,
```

```
            "float_value": null,
            "double_value": null
          }
        },
        {
          "key": "value",
          "value": {
            "string_value": null,
            "int_value": "1",
            "float_value": null,
            "double_value": null
          }
        },
        {
          "key": "category",
          "value": {
            "string_value": "category1",
            "int_value": null,
            "float_value": null,
            "double_value": null
          }
        },
        {
          "key": "firebase_screen_id",
          "value": {
            "string_value": null,
            "int_value": "1109587836504693341",
            "float_value": null,
            "double_value": null
          }
        }
      ],
      "timestamp_micros": "1489478204880000",
      "previous_timestamp_micros": "1489137436229000",
      "value_in_usd": null
    }
  ]
},
{
  "user_dim": {
    "user_id": null,
    "first_open_timestamp_micros": "1488878151620000",
    "user_properties": [
      {
        "key": "first_open_time",
        "value": {
          "value": {
            "string_value": null,
            "int_value": "1488880800000",
            "float_value": null,
            "double_value": null
          },
          "set_timestamp_usec": "1488878151620000",
          "index": null
        }
      }
    ],
    "device_info": {
      "device_category": "mobile",
      "mobile_brand_name": null,
      "mobile_model_name": null,
      "mobile_marketing_name": null,
      "device_model": "507SH",
      "platform_version": "6.0.1",
      "device_id": null,
      "resettable_device_id": null,
      "user_default_language": "ru-ru",
      "device_time_zone_offset_seconds": "10800",
      "limited_ad_tracking": "false"
    },
    "geo_info": {
      "continent": "Europe",
      "country": "Russia",
      "region": "Moscow",
```

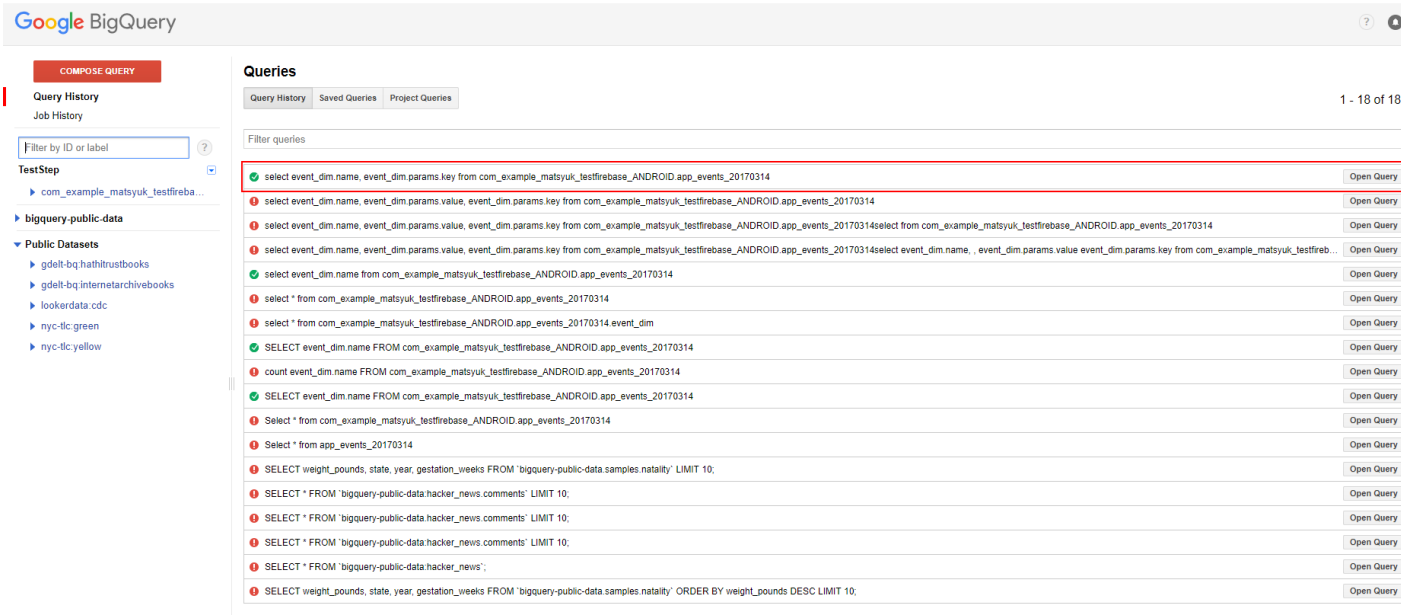
```
        "city": "Moscow"
      },
      "app_info": {
        "app_version": "1.0",
        "app_instance_id": "d0c587de4d5804ddc1d34f8d54b981f9",
        "app_store": "manual_install",
        "app_platform": "ANDROID",
        "app_id": "com.example.matsyuk.testfirebase"
      },
      "traffic_source": null,
      "bundle_info": {
        "bundle_sequence_id": "62",
        "server_timestamp_offset_micros": "-499813"
      },
      "ltv_info": null
    },
    "event_dim": [
      {
        "date": "20170314",
        "name": "select_content",
        "params": [
          {
            "key": "firebase_screen_class",
            "value": {
              "string_value": "MainActivity",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "content_type",
            "value": {
              "string_value": "image",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "item_name",
            "value": {
              "string_value": "name1",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "firebase_event_origin",
            "value": {
              "string_value": "app",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "firebase_screen_id",
            "value": {
              "string_value": null,
              "int_value": "1109587836504693341",
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "item_id",
            "value": {
              "string_value": "1",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
```

```
          }
        ],
        "timestamp_micros": "1489478204208000",
        "previous_timestamp_micros": "1489137435605000",
        "value_in_usd": null
      }
    ]
  },
  {
    "user_dim": {
      "user_id": null,
      "first_open_timestamp_micros": "1488878151620000",
      "user_properties": [
        {
          "key": "first_open_time",
          "value": {
            "value": {
              "string_value": null,
              "int_value": "1488880800000",
              "float_value": null,
              "double_value": null
            },
            "set_timestamp_usec": "1488878151620000",
            "index": null
          }
        }
      ],
      "device_info": {
        "device_category": "mobile",
        "mobile_brand_name": null,
        "mobile_model_name": null,
        "mobile_marketing_name": null,
        "device_model": "507SH",
        "platform_version": "6.0.1",
        "device_id": null,
        "resettable_device_id": null,
        "user_default_language": "ru-ru",
        "device_time_zone_offset_seconds": "10800",
        "limited_ad_tracking": "false"
      },
      "geo_info": {
        "continent": "Europe",
        "country": "Russia",
        "region": "Moscow",
        "city": "Moscow"
      },
      "app_info": {
        "app_version": "1.0",
        "app_instance_id": "d0c587de4d5804ddc1d34f8d54b981f9",
        "app_store": "manual_install",
        "app_platform": "ANDROID",
        "app_id": "com.example.matsyuk.testfirebase"
      },
      "traffic_source": null,
      "bundle_info": {
        "bundle_sequence_id": "61",
        "server_timestamp_offset_micros": "-537470"
      },
      "ltv_info": null
    },
    "event_dim": [
      {
        "date": "20170314",
        "name": "session_start",
        "params": [
          {
            "key": "firebase_screen_class",
            "value": {
              "string_value": "MainActivity",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
```

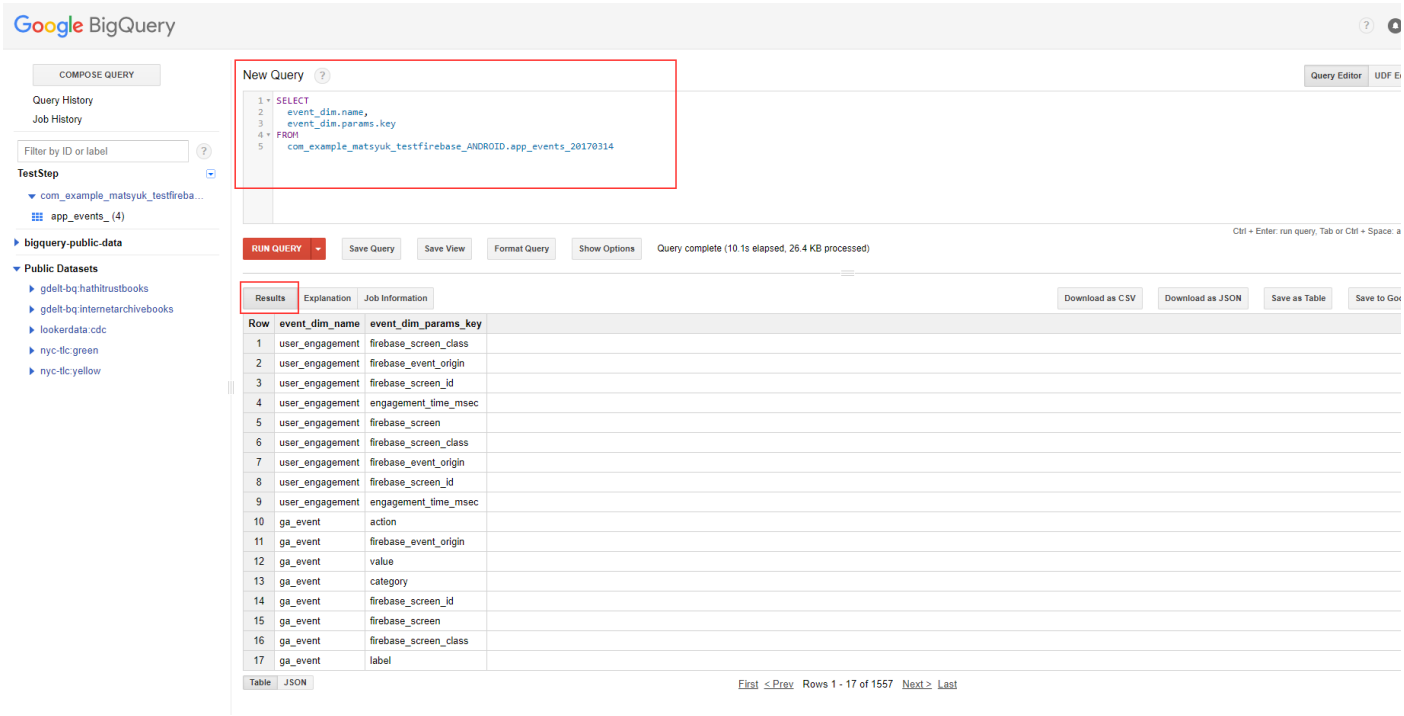
```
          {
            "key": "firebase_event_origin",
            "value": {
              "string_value": "auto",
              "int_value": null,
              "float_value": null,
              "double_value": null
            }
          },
          {
            "key": "firebase_screen_id",
            "value": {
              "string_value": null,
              "int_value": "1109587836504693341",
              "float_value": null,
              "double_value": null
            }
          }
        ],
        "timestamp_micros": "1489478198696000",
        "previous_timestamp_micros": "1489137330069000",
        "value_in_usd": null
      }
    ]
  }
]
```


Красота, да и только!

Теперь более подробно рассмотрим Queries. Выберем первый (*рисунок кликабельный*):

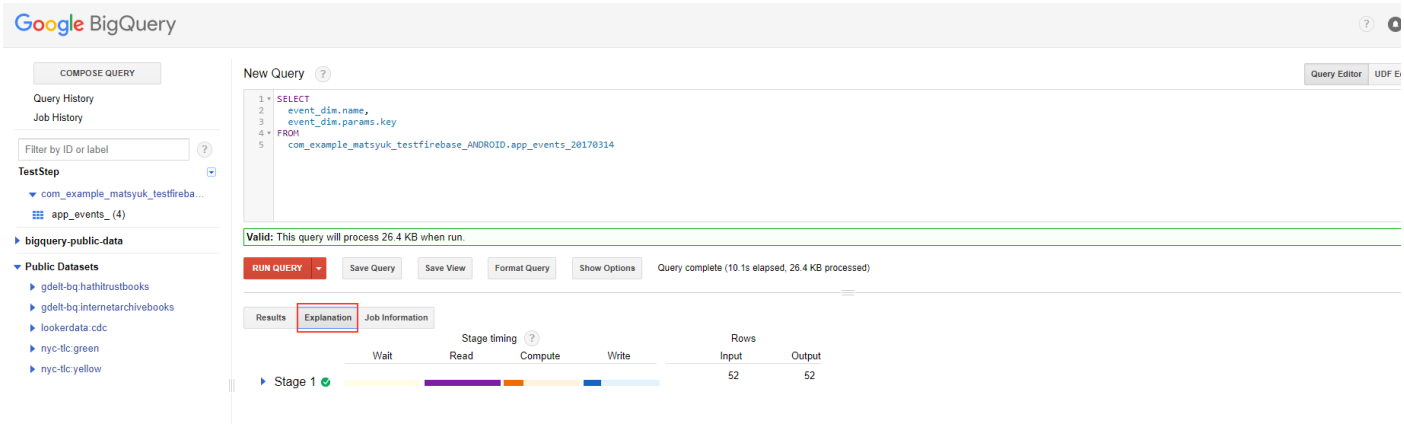


И перед нами откроется следующий экран (*рисунок кликабельный*):

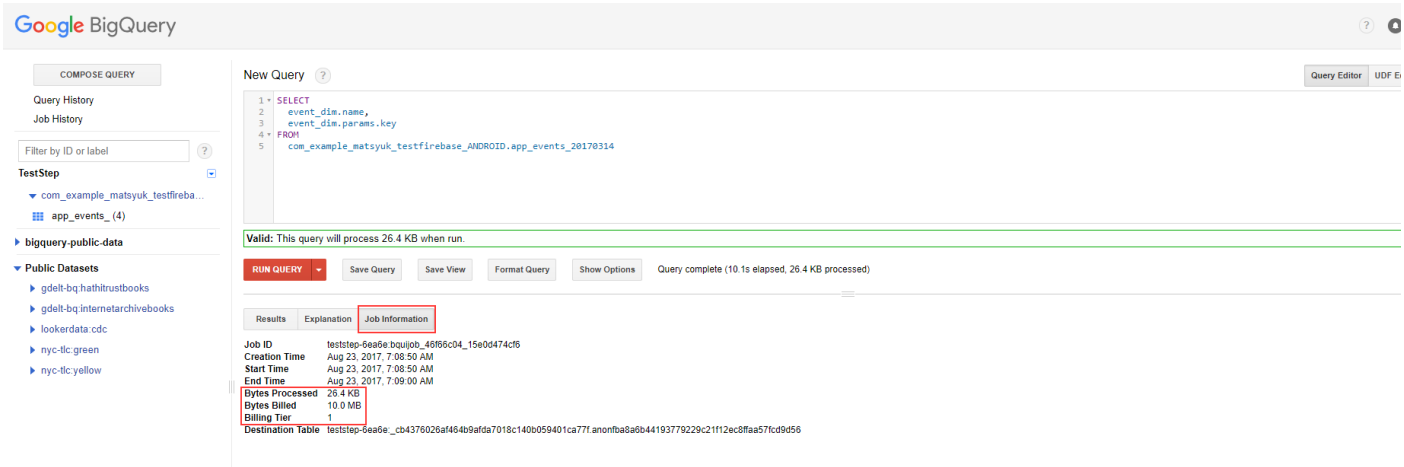


Запрос наш довольно произвольный. Обратите внимание на вкладку Results. Собственно, в ней вы и увидите результаты вашего запроса.

Если открыть вкладку Explanation, то вы увидите более подробный процесс прохождения запроса (*рисунок кликабельный*):



Ну и самая интересная вкладка — Job information (*рисунок кликабельный*):



Обратите внимание на Bytes Processed, Bytes Billed и Bites Tier. В ходе запроса было обработано 26,4 KB, но платите вы по нижней границе для Bites Tier = 1, то есть платите как за 10 MB. Однако, судя по документации, 1 TB в месяц для вас будет бесплатным, а каждый последующий будет стоить $5. Вполне вам хватит наиграться и напробоваться. Ну и важное дополнение — платите вы толь успешные запросы!

Даже очень краткий обзор по BigQuery получается немаленьким. Это очень мощный и функциональный инструмент, с помощью которого вы можете анализировать данные как угодно. Но за 5 минут в BigQuery вы точно не разберетесь, в отличие от обычной ко

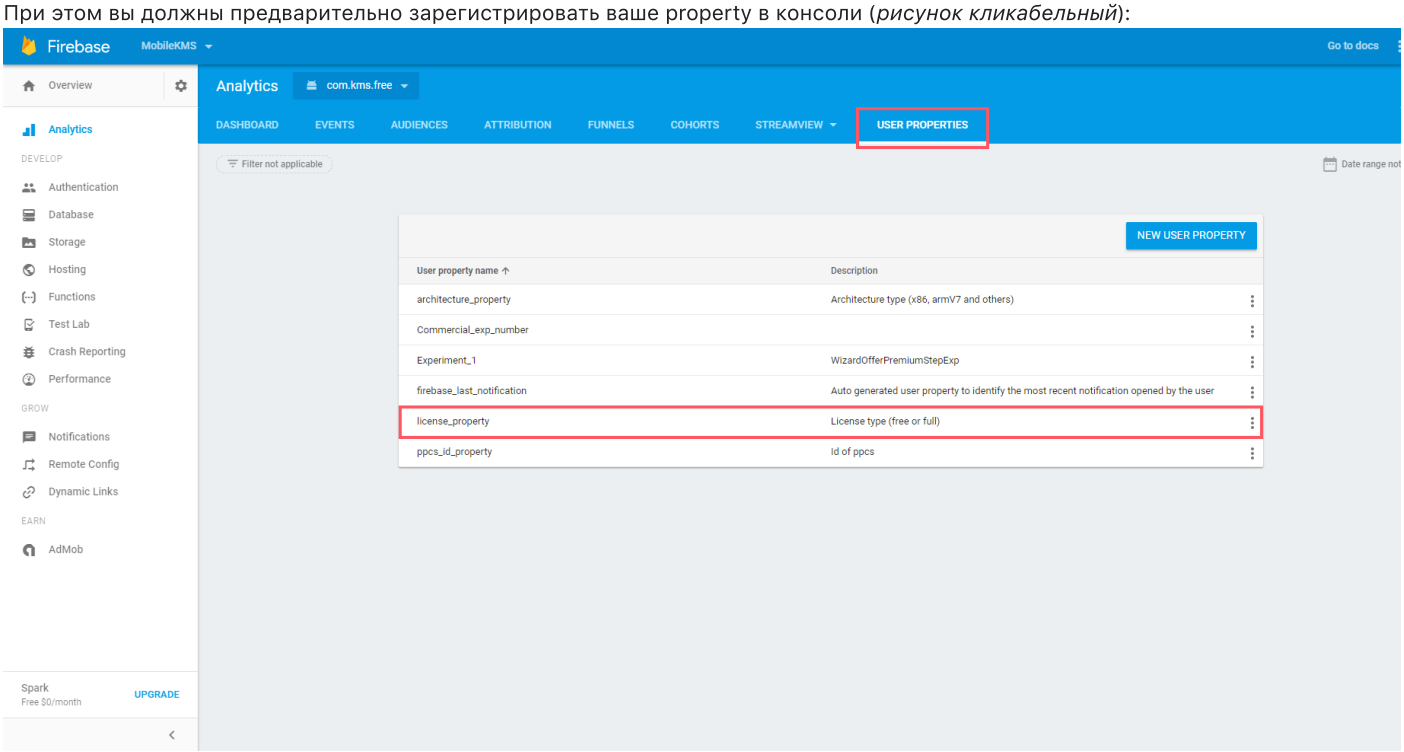
в GA или FA. Поэтому очень круто, если в вашей команде или компании есть человек, который в этом разбирается, и который може получить какие угодно результаты.
Если этим человеком хотите стать вы, то начать можете со вступительного видео от «Гугла», где, кстати, рассказывается и про рас стоимости. Также есть неплохие статьи — раз и два. Далее советую вам копать в сторону официальной доки и книги по BigQuery (ш книга, Карл!).
Будет здорово, если кто-то уже хорошо покопал в эту сторону и может поделиться советами и опытом =)
Отмечу также, что существуют UI-обертки над BigQuery типа Data Studio, позволяющие загружать туда данные и удобно их визуализировать. Data Studio пока в бете, но в будущем обещает стать очень удобным инструментом.

User properties

Мы, по сути, продолжаем тему событий, так как user properties — ее неотъемлемая часть.
User properties (по-русски «Свойства пользователя») — это признаки, с помощью которых вы можете описывать различные сегмен вашей пользовательской базы, такие как язык, географическая локация и т.д. Их еще называют **sticky params**, так как они прикрепляются к каждому событию.
Изначально к каждому событию прикрепляются только properties по-умолчанию. А если в коде вы вызываете подобный код:

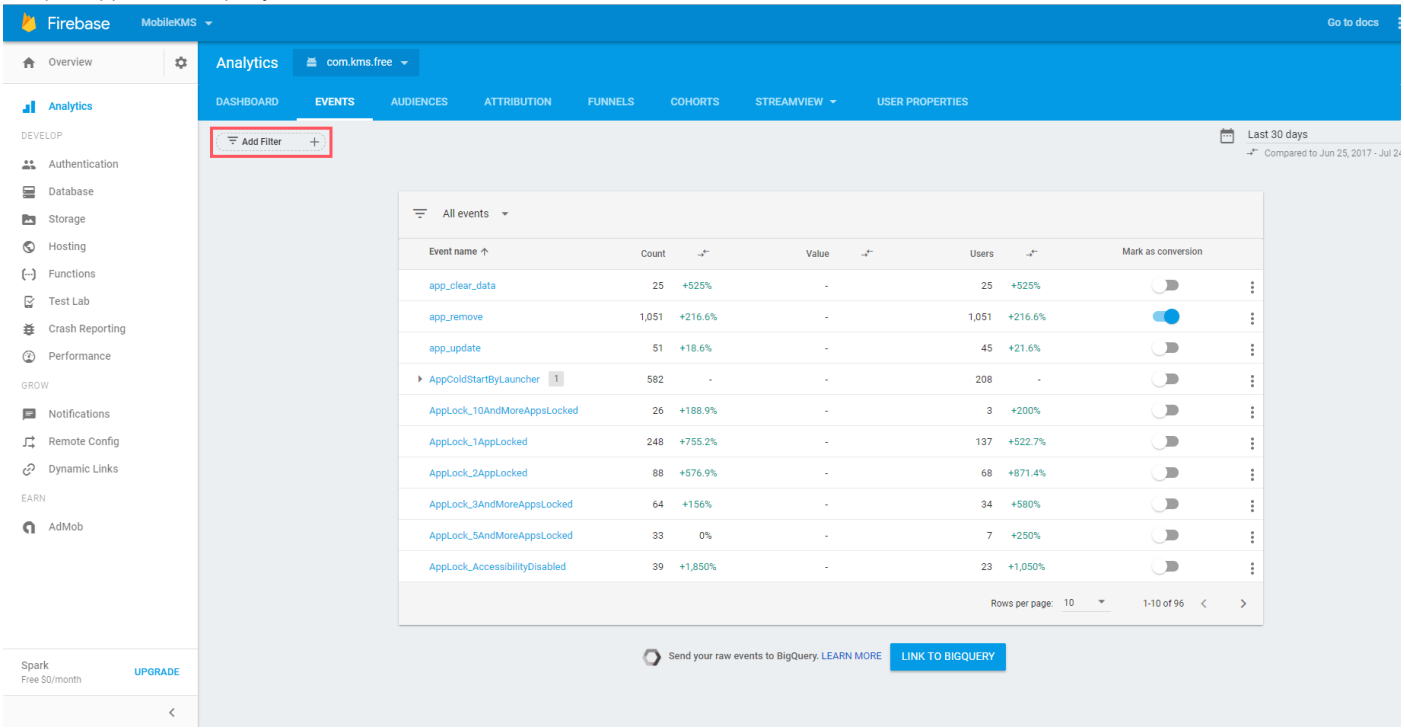
```
mFirebaseAnalytics.setUserProperty("license_property", mLicenseType);
```

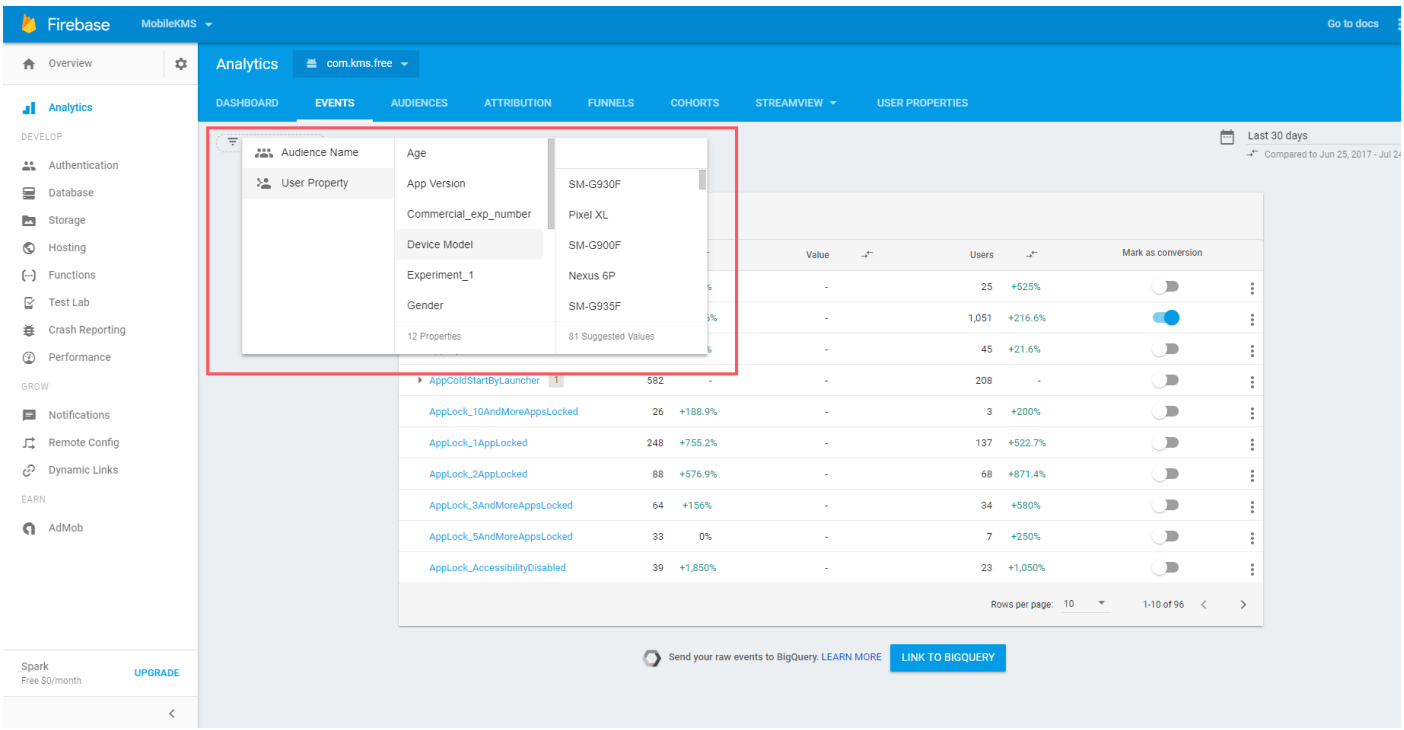
то к каждому последующему вашему событию будет прикрепляться property «license_property» с заданным заранее значением (значение «mLicenseType»). И даже после перезапуска приложения, телефона и прочее данное property будет прикрепляться. То е property является еще и **persistence**.
При этом вы должны предварительно зарегистрировать ваше property в консоли (*рисунок кликабельный*):



Все подробно расписано здесь и в api.
Отмечу, что для конкретного приложения вы можете отправлять **до 25 properties** (без учета properties, которые отправляются по умолчанию). Список properties, отправляемых по умолчанию, здесь.
Собственно в консоли вы можете фильтровать все, что угодно, по properies и «аудитории» (про «аудиторию» скажем чуть ниже).

Например, события (*рисунки кликабельные*):





Аналогом `setUserProperty(...)` в GA являются методы `setCustomDimension(...)` и `setCustomMetric(...)`. Единственное, данные dime и metric не являются sticky и persistence, и вам будет необходимо к каждому событию каждую сессию вручную прикреплять их.

События. FA + другая аналитика

Думаю, в каждом приложении есть как минимум два аналитических инструмента. Обычно их гораздо больше. Аналитики тоже прогрессивные люди и не стоят на месте. Но нам все это поддерживать. Да и плюс трафик. Так что же лучше сделать?
Есть очень хорошая гугловская статья, которая уже была упомянута мною, где описываются различные варианты.

Кратко представлю их, чтобы вы имели представление:

1. Просто отдельно слать разные аналитики. В коде вы, скорее всего, создадите некий универсальный фасад, который и будете использовать везде.
   Минусы, я думаю, понятны. Больше трафика и кода.

2. Google Tag Manager.
   Данный менеджер подключается через консоль и там же настраивается. Суть в том, что в коде вы вообще ничего не делаете

дополнительно (за исключением build.gradle и добавления конфигурационных файлов), просто шлете FA-события — и все. А уж своей стороне Google Tag Manager трансформирует FA-события по заданным вами правилам в события аналитик, которые вам нужны (GA, AppsFlyer и прочие партнеры Google Tag Manager). Там же вы можете настраивать все возможные триггеры, по кото события FA будут попадать в другие аналитики (например, для какой-то аналитики нужны только строго определенные события Звучит прям очень круто и гибко. Сам, к сожалению, не пробовал, так как нужно определенное время, чтобы погрузиться и разобраться, что есть что там. Если у кого есть опыт, пишите, будет очень круто более подробно расписать Google Tag Manager Пока же кину статью, с которой можно попробовать начать.
Но есть минусы. Первое — необходимо время, чтобы настроить все тэги для всех событий, да и вообще просто разобраться. Вт — вы не можете использовать FA и Google Tag Manager для отправки в GA ecommerce data.

3. BigQuery.
   Это относится, конечно же, к GA и FA, когда вам необходимо совместить данные. Но выгрузить данные с GA в BigQuery вы смож только если у вас Google Analytics 360.

Особенности подключения FA

Чтобы настроить на своем проекте аналитику, вам нужно четко следовать данной документации. Есть уже встроенный в Android St плагин, который делает за вас половину работы. Если у вас все в первый раз, то процесс займет не более 15 минут. Хотите API? А в оно, довольно короткое и вроде понятное.
После настойки FA через Android Studio Assistant у себя в проекте вы заметите появление нового файла `google-services.json`. Это специальный файл, в котором прописаны все идентификаторы и пути, которые необходимы для работы в вашем приложении разли гугловских сервисов, в данном случае — для работы FA.
Также в ваш корневой `build.gradle` добавилась такая строчка:

```
dependencies {
    classpath 'com.google.gms:google-services:3.0.0'
    // ...
}
```

Собственно `google-services` — это специальный плагин, который распарсивает `google-services.json`, преобразуя его в обычные строчки, используемые FA. А также `google-services` добавляет все необходимые зависимости для используемых гугловских сервис нашем случае только для FA). Правда, для этого нужно в `app/build.gradle` в самом конце добавить:

```
apply plugin: 'com.google.gms.google-services'
```

В `google-services.json` находится вся необходимая информация для подключения вашего проекта к Firebase, причем не только к аналитике, но и ко всем инстументам.

▾ Вот как выглядит примерный google-services.json:


```
{
  "project_info": {
    "project_number": "887654601522",
    "firebase_url": "https://fir-test3-4bab3.firebaseio.com",
    "project_id": "fir-test3-4bab3",
    "storage_bucket": "fir-test3-4bab3.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:887654601522:android:9c6c1c11f784b956",
        "android_client_info": {
          "package_name": "com.example.matsyuk.firebasetest3"
        }
      },
      "oauth_client": [
        {
          "client_id": "887654601522-o8rolth1g5mq5qq650844chk07mib2un.apps.googleusercontent.com",
          "client_type": 1,
          "android_info": {
            "package_name": "com.example.matsyuk.firebasetest3",
            "certificate_hash": "82f13b732dec32c5ebd4498c3a7acf4bda23a846"
          }
        },
        {
```

```
          "client_id": "887654601522-4riqkg424gb236q6mqehksn03u4hoqqg.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "AIzaSyAYRPNTcgxWP7qUzI__kx9gSwxnIgc3iBo"
        }
      ],
      "services": {
        "analytics_service": {
          "status": 1
        },
        "appinvite_service": {
          "status": 2,
          "other_platform_oauth_client": [
            {
              "client_id": "887654601522-4riqkg424gb236q6mqehksn03u4hoqqg.apps.googleusercontent.com",
              "client_type": 3
            }
          ]
        },
        "ads_service": {
          "status": 2
        }
      }
    }
  ],
  "configuration_version": "1"
}
```


С помощью плагина `google-services` данный json преобразуется в набор строчек, сгенерированных в файл `your_project\app\build\generated\res\google-services\debug\values\values.xml`:

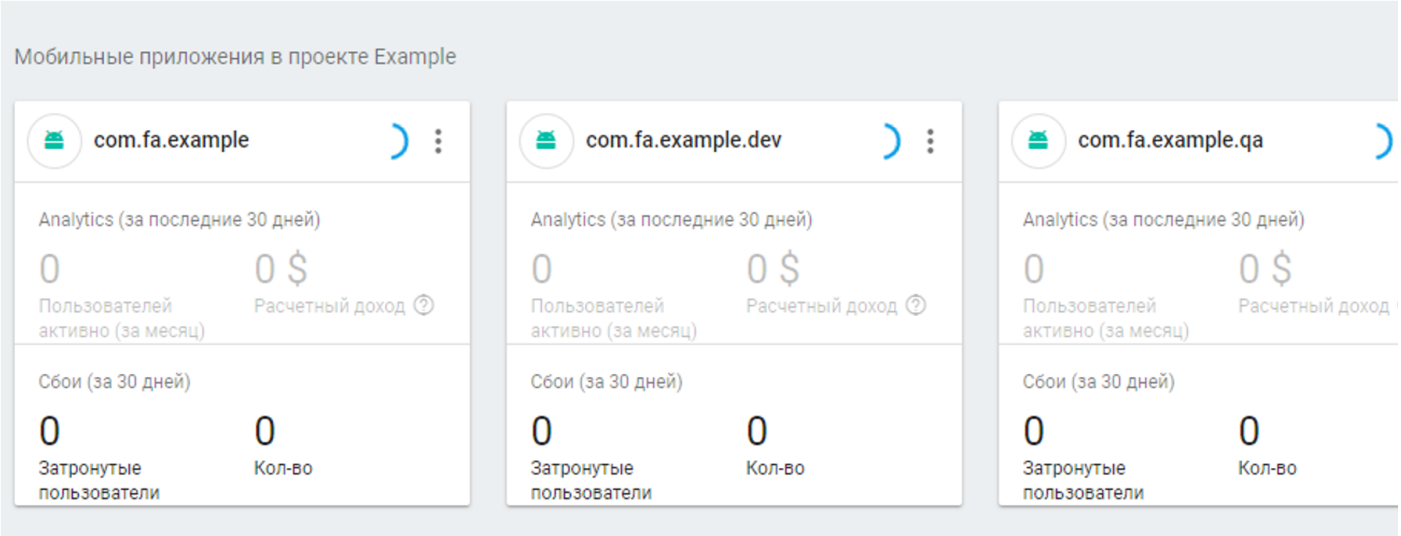

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <string translatable="false" name="default_web_client_id">887654601522-4riqkg424gb236q6mqehksn03u4hoqqg.apps.googleuse
ontent.com</string>
    <string translatable="false" name="firebase_database_url">https://fir-test3-4bab3.firebaseio.com</string>
    <string translatable="false" name="gcm_defaultSenderId">887654601522</string>
    <string translatable="false" name="google_api_key">AIzaSyAYRPNTcgxWP7qUzI__kx9gSwxnIgc3iBo</string>
    <string translatable="false" name="google_app_id">1:887654601522:android:9c6c1c11f784b956</string>
    <string translatable="false" name="google_crash_reporting_api_key">AIzaSyAYRPNTcgxWP7qUzI__kx9gSwxnIgc3iBo</string>
    <string translatable="false" name="google_storage_bucket">fir-test3-4bab3.appspot.com</string>
</resources>
```


Обратите внимание на такие строчки, как `firebase_database_url`, `google_storage_bucket` и т.д. Захотите подключить еще один инструмент от Firebase, в проекте у вас уже все будет готово для этого.
Подробнее про плагин и устройство `google-services.json` написано здесь.

А теперь рассмотрим жизненный пример. Есть у нас приложение **Example** с `applicationId`, равное, допустим, `com.fa.example`. И ест нас в продукте flavors:

```
productFlavors {
    dev {
        applicationId "com.fa.example.dev"
    }
    qa {
        applicationId "com.fa.example.qa"
    }
    prod {
        // applicationId "com.fa.example"
    }
}
```

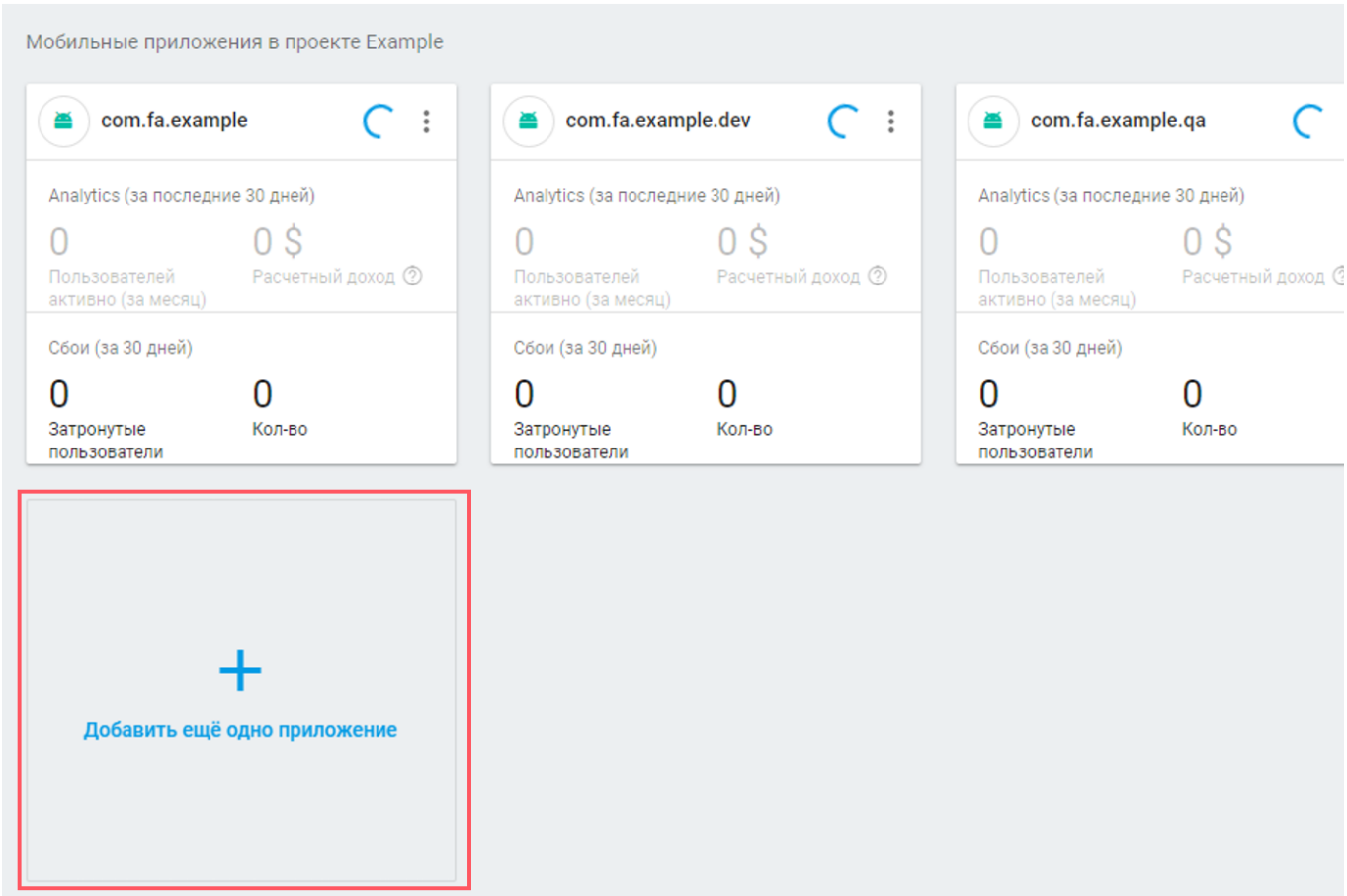
Далее мы хотим зарегистрировать проект в FA через Android Studio Assistant. Делаем все по инструкции и получаем в консоли про **Example**, в котором три приложения:



И если вы посмотрите в проекте `app/google-services.json`, то в нем будет информация **о ваших трех приложениях (трех flavors с разными `applicationId`)**. То есть для каждого flavor аналитика будет собираться отдельно.
Также отмечу, что вы можете самостоятельно скачать `google-services.json` с любого приложения вашего проекта. **Но все `google-services.json` вашего проекта будут одинаковы и будут содержать информацию о всех приложениях в проекте**.

Далее такая ситуация. Ваш проект **Example** настроен с FA. Но вам вдруг понадобилось добавить в проект еще один flavor с другим именем пакета. И для этого flavor вы также хотите собирать аналитику отдельно. Тогда вам необходимо сделать следующее:

1. Добавить новый flavor в build.gradle.
2. Зарегистрировать новое приложение в проекте в консоли:



3. Скачать новый `google-services.json` (в котором будет информация о четырех приложениях в проекте) и подставить его вместо старого.

   Теперь такой пример. Допустим, есть у вас в проекте `buildTypes`, и выглядят они в `build.gradle`-файле следующим образом:

```
buildTypes {
    release {

    }
    ultra_debug {
        applicationIdSuffix ".ultra_debug"
    }
    debug {
        applicationIdSuffix ".debug"
    }
}
```

То есть для сборок `ultra_debug` и `debug` вы добавляете суффикс к имени пакета. Таким образом, у вас в проекте три вышеназван buildTypes и три flavors:

```
productFlavors {
    dev {
        applicationId "com.fa.example.dev"
    }
    qa {
        applicationId "com.fa.example.qa"
    }
    prod {
        // applicationId "com.fa.example"
    }
}
```

Вы запускаете Android Studio Assistant для подключения к проекту FA. Как вы думаете, сколько будет зарегистрировано в консо приложений и с какими именами пакетов?
Не догадаетесь =) Появятся в консоли такие приложения:

```
com.fa.example.debug
com.fa.example.dev.debug
com.fa.example.qa.debug
```

Почему именно только с суффиксом «debug», осталось для меня загадкой. Так что имейте в виду данный баг.

Ну и заключительный пример.
У вас в проекте все также те самые три flavors. И вы хотите добавить в проекте еще один flavor (например, `custom`), но для него нет необходимости в другом `applicationId`, и при этом данный flavor тоже желательно отдельно от всех остальных просматривать с точ зрения аналитики:

```
productFlavors {
    dev {
        applicationId "com.fa.example.dev"
    }
    qa {
        applicationId "com.fa.example.qa"
    }
    prod {
        // applicationId "com.fa.example"
    }
    custom {
        // applicationId "com.fa.example"
    }
}
```

Ситуация усложняется еще тем, что **в консоли в проект вы не можете добавлять приложения с одинаковым `applicationId`**. Как быть? Делаем следующее:

1. Регистрируем **новый проект** (не приложение в составе проекта **Example**, а именно отдельный проект) в консоли.

2. Регистрируем в новом проекте приложение `com.fa.example`.
3. Скачиваем с нового проекта `google-services.json`.
4. Подставляем новый `google-services.json` следующим образом (выделено красным).

То есть в вашем андроидовском проекте будут лежать уже два `google-services.json`. При сборке `google-services plugin` сначала смотрит в папку конкретного flavor. Если в этой папке есть `google-services.json`, то плагин берет его. Если нет, то тогда берется `go services.json` с `app` папки. Довольно удобно и гибко получается в итоге.

Вроде бы жизнь разработчика стала проще. Зарегистрировал проект в консоли, скачал `google-services.json`, закинул в `app/` (ну эт в случае без flavors и прочего), и все, больше ни о чем не думаешь. Но иногда бывает необходимость на лету переключить канал аналитики. И если в GA вы могли задавать в коде id, то в FA пока что такая возможность отсутствует. И были у меня надежды снача такую конструкцию (взято с SO):

```
FirebaseOptions options = new FirebaseOptions.Builder()
       .setApplicationId("bla-bla") // Required for Analytics.
       .setApiKey("bla-bla") // Required for Auth.
       .setDatabaseUrl("bla-bla") // Required for RTDB.
       .build();
FirebaseApp.initializeApp(this /* Context */, options, "secondary");
```

Но выдается ошибка *«Missing google_app_id. Firebase Analytics disabled»*. Команда Firebase знает про это и постепенно работает на данной проблемой.
Более подробно про все описанные примеры вы можете прочитать в вышеназванной статье про `google-services`-плагин и здесь.

Отправка данных

В GA есть метод `setLocalDispatcher(...)`. С помощью него мы можем задавать интервал периодической отправки данных. Хорошо, FA заботится о нас и нашем трафике и не дает нам возможность самим регулировать данный параметр. Но в GA с помощью метода `setLocalDispatcher(-1)` мы можем отменить автоматическую отправку событий, а с методом `dispatchLocalHits()` вручную отправл **накопившиеся** события. Это очень удобно, когда, например, мы не хотим отправлять события до принятия соглашения и т.д.
У FA подобной возможности накопления и отправки событий нет, придется все руками делать.
Зато хотя бы есть метод `setAnalyticsCollectionEnabled(boolean enabled)`, с помощью которого мы можем включать и отключать аналитику. Например, если мы не хотим отправлять аналитику до принятия пользователем нужного соглашения, то в манифесте прописываем:

```
<meta-data android:name="firebase_analytics_collection_enabled" android:value="false" />
```

А потом, когда нужно, в коде вызовем:

```
setAnalyticsCollectionEnabled(true);
```

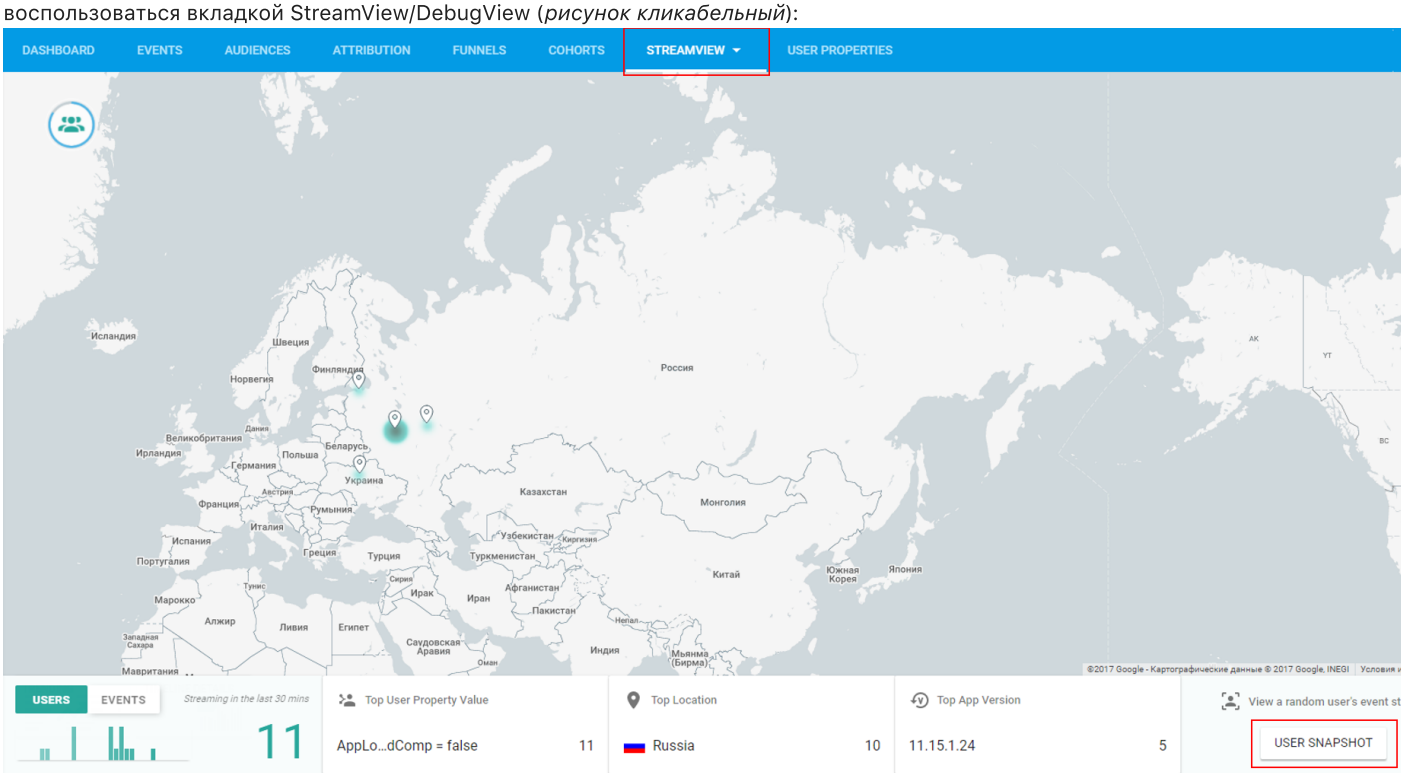
Также можно отключить аналитику на постоянной основе. То есть даже вызов `setAnalyticsCollectionEnabled(true)` не поможет. Дл этого в манифесте прописываем:

```
<meta-data android:name="firebase_analytics_collection_deactivated" android:value="true" />
```

Информация взята из данной статьи.

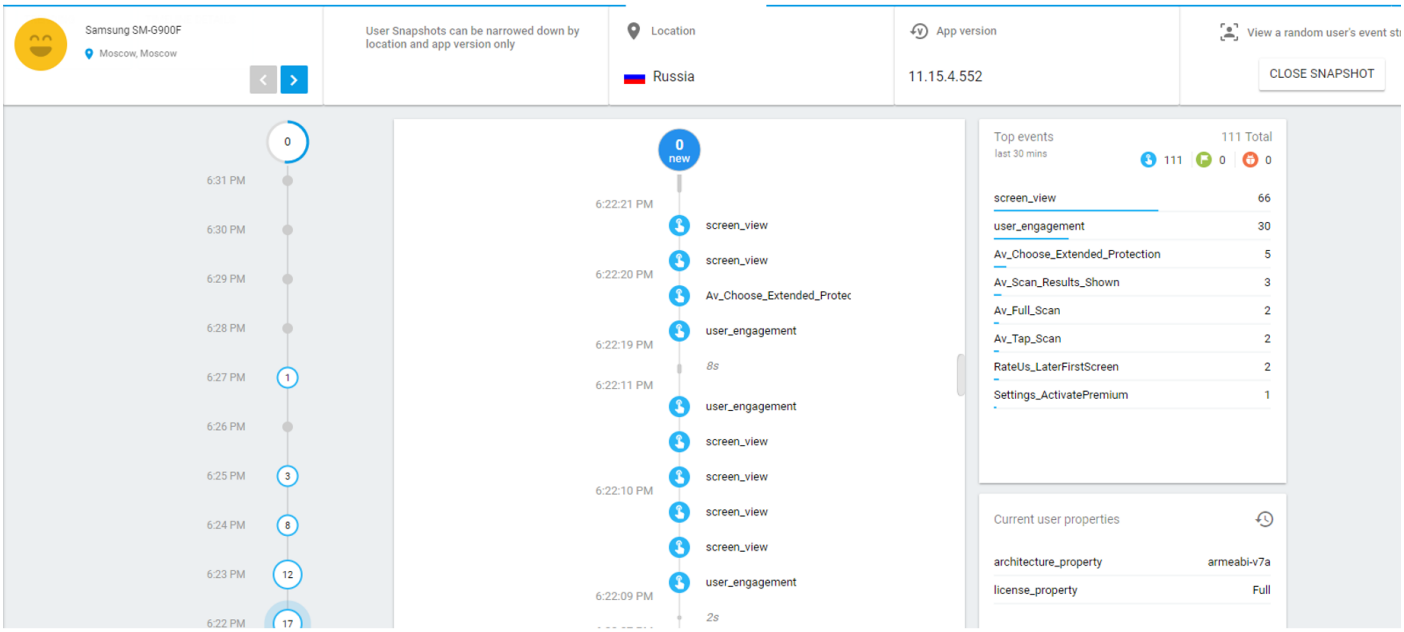
Режим реального времени и отладка в FA

В FA события с реальных устройств приходят в консоль лишь спустя сутки. И в начале не было возможности просмотреть действия пользователя в реальном времени. Чтобы увидеть первые данные, приходилось ждать целые сутки. Сейчас же вы можете воспользоваться вкладкой StreamView/DebugView (*рисунок кликабельный*):



На картинке выше представлен StreamView, на котором вы можете наблюдать, как себя ведут пользователи в данный момент време Также вы можете выбрать режим Snapshot (кнопка User snapshot справа снизу), и вам покажутся действия случайно выбранного

пользователя (*рисунок кликабельный*):



Подобным образом выглядит и DebugView. Наконец-то отлаживаться можно в режиме реального времени. Вы будете видеть все ev и properties, которые посылаются вашим приложением, включая и events с properties по умолчанию. Как можно представить, до DebugView процесс отладки был воистину ужасным.
О StreamView и DebugView хорошо расписано здесь.

Понятие «сессии» в FA

Что в GA, что в FA, все мы видим такие слова, как «сессия», «количество событий за сессию» и т.д. И, наверное, может сложиться впечатление, что сессия = время жизни процесса. Но это не так. Сессия — это просто временной промежуток, в течение которого приложение активно (находится в foreground). В API FA есть такие методы:

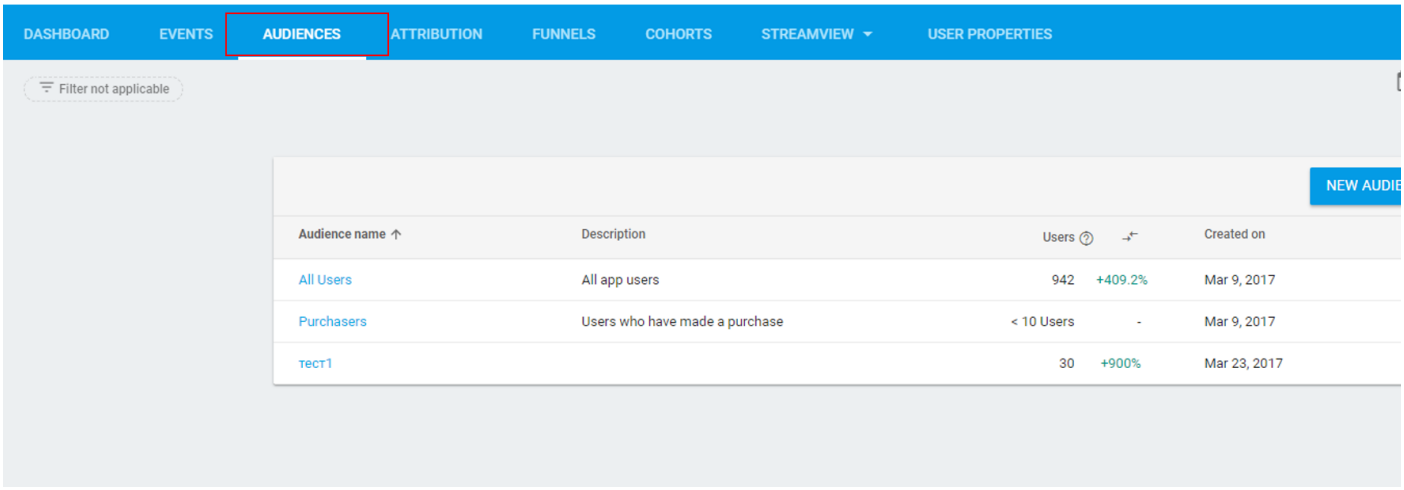
```
setMinimumSessionDuration (long milliseconds); // default 10 sec
setSessionTimeoutDuration (long milliseconds); // default 30 min
```

То есть если вы запустили приложение и убили его менее чем за minimumSessionDuration, то сессия даже не начнется. Если же запущенное приложение находится в foreground более minimumSessionDuration, то сессия стартует.
Если ваше приложение было выгружено системой, но оно успело подняться до истечения sessionTimeoutDuration, то это все будет с сессия. Если вы запустили приложение, что-то поделали там, потом вышли из него (то есть приложение не в foreground), и только ч sessionTimeoutDuration+ зашли обратно (при этом приложение не было убито, к примеру), то первая сессия завершится и стартует вторая.

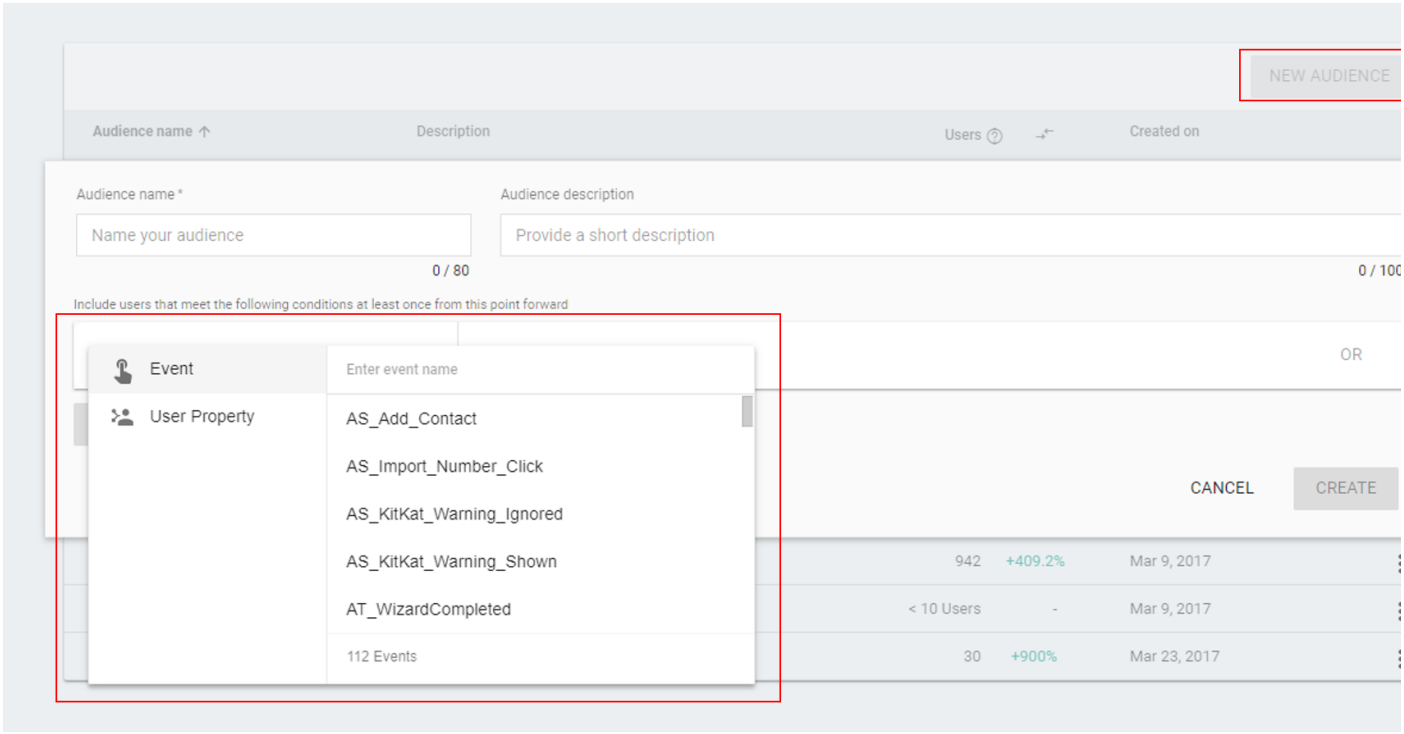
Еще немного об FA-консоли

Audiences

Вы можете формировать разные аудитории, по которым в дальнейшем можно выставлять фильтры, организовывать компании и т.д это выглядит (*рисунок кликабельный*):



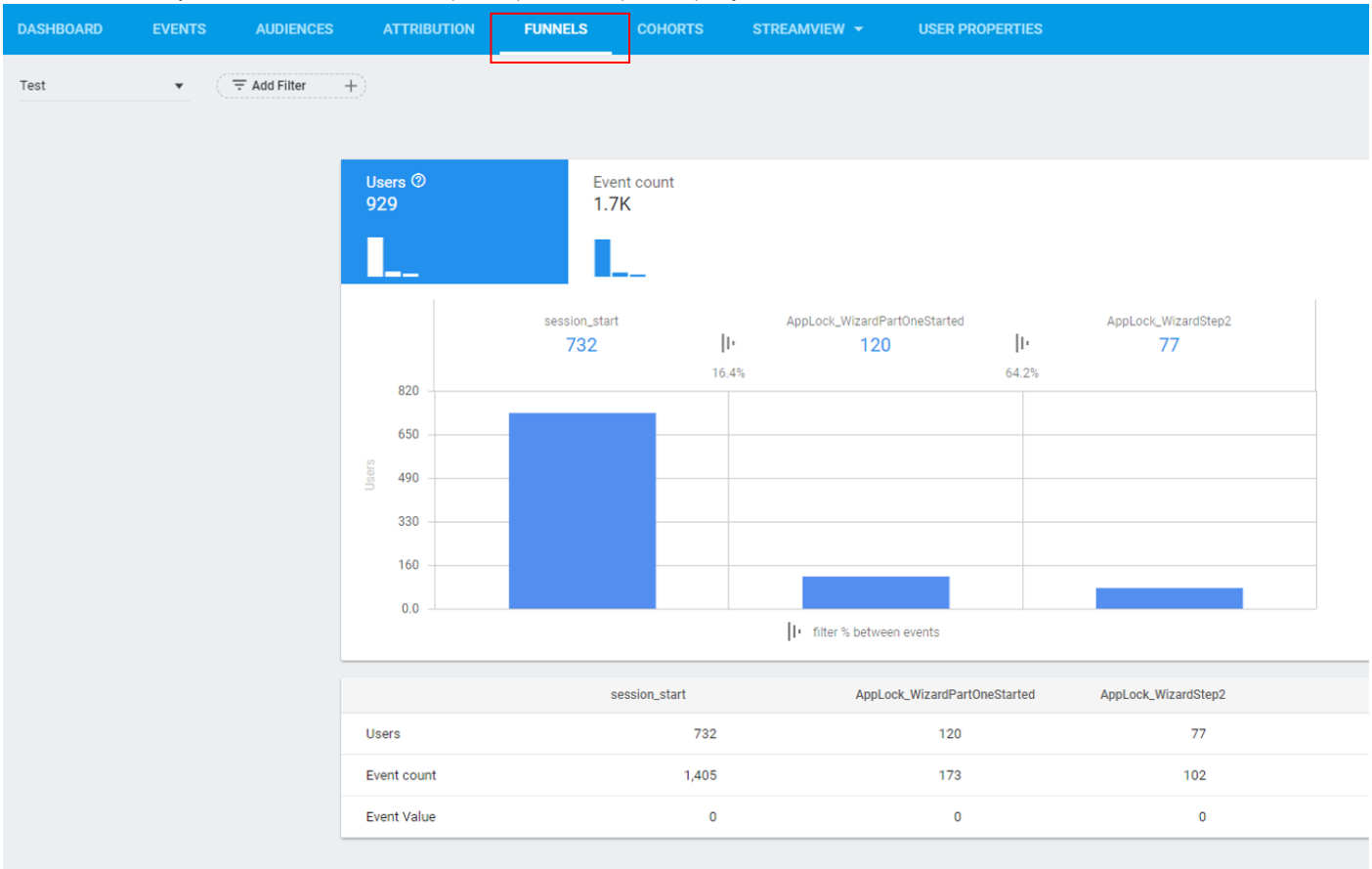
Создание новой «аудитории» (*рисунок кликабельный*):



Допустим, вам нужна аудитория «Мужчины из России, которые прошли регистрацию». Тогда при создании «аудитории» вы выбир properties «country» = «Russia» и «sex» = «male» и event «reg_comleted» (это уже ваш кастомный event) = «true».

Funnels

На этой вкладке у вас есть возможность строить разные воронки (*рисунок кликабельный*):



Очень нравится маркетологам это делать =)
Отмечу, что подобный функционал есть и у GA.

Есть еще вкладки Attribution и Cohorts. Но, честно говоря, я ими вообще не пользовался. Для чего они нужны, лучше распишут аналитики.
Полное описание консоли можно прочитать здесь.

FA. Выводы

Немаленькая у нас в итоге получилась статья. Давайте попробуем подвести итоги.

Плюсы:

1. FA — это активно развивающийся продукт. Примерно с февраля я наблюдаю за его развитием и должен отметить, что команда разработки старается максимально реализовывать первоочередные потребности пользователей.
2. FA events + BigQuery. Это прямо главное преимущество FA. У вас есть доступ ко **всем** событиям вашего приложения практичес бесплатно. И если в вашей команде есть спец по BigQuery, то вам чертовски повезло. Кроме того, сами «события» в FA намног более гибкие и удобные в использовании.
3. Минимализм. В консоли, по сути, только самое необходимое. Акцент делается на «события». В GA же все-таки много всего намешано, и далеко не все нам нужно.
4. Интеграция с другими проектами Firebase. Будь то сбор крашей или RemoteConfig. Продукты действительно дополняют друг др это открывает новые возможности.

Минусы:

1. Ребятам еще много работать, особенно в консоли. Но мы верим в них =)
2. Разбросанность информации. Это то, о чем я говорил в самом начале статьи. Каждый вопрос или уточнение нужно искать и ресерчить. Отсутствие упорядоченности тоже сбивает вначале. Но данная статья в принципе призвана устранить данный недостаток.

Меня часто спрашивают, так стоит ли использовать FA или нет. Может, вполне достаточно GA? Или сразу обе аналитики не достойн места в вашем продукте?

Однозначного ответа нет. Все очень зависит от потребностей ваших аналитиков и маркетологов. А также зависит от способностей ваших аналитиков осилить BigQuery. Все-таки мы, разработчики, — это «пехотинцы продукта», особенно в части аналитики. Что на скажут, то мы и будем делать. Но лично я бы смотрел в сторону связки FA + BigQuery. Уж очень она крутая, и вы никак не ограниче возможностями консоли.

Большое спасибо, что дочитали до конца! Пишите комментарии, дополняйте и поправляйте! Сделаем нашу разработческую жизнь лучше!

P.S. Большое спасибо хочу сказать **Тимуру Ахметгарееву** за помощь и за то, что никогда не бросал в беде =)

P.P.S. И еще добавлю. 16 сентября 2017 совместно с независимым сообществом разработчиков MOSDROID мы организовываем вечернюю встречу для всех, кто заинтересован в разработке под Android. По традиции, мы подготовили для вас несколько докладс Ждём всех желающих. Зарегистрироваться на встречу можно здесь.

Метки: android, android development, mobile development, analytics, firebase analytics, firebase, java, google analytics

+11 58 5k 2

«Лаборатория Касперского» 663,77
Ловим вирусы, исследуем угрозы, спасаем мир

29,0 Карма 8,8 Рейтинг 92 Подписчики

Евгений Мацюк @xoxol_89
android developer

Мой круг Facebook Twitter Вконтакте Instagram

Поделиться публикацией

ПОХОЖИЕ ПУБЛИКАЦИИ

25 ноября 2016 в 23:34
Security Week 47: закладки в Android, безопасность Wi-Fi, уязвимость NTP
+13 10k 28 5

29 апреля 2016 в 18:45
Security Week 17: Взлом SWIFT и кассовых аппаратов, вымогательство в Android с эксплойтами, обход AppLocker
+7 17,1k 29 6

28 декабря 2015 в 18:32
Security Week 52-53: бэкдор у Juniper с толстым слоем криптографии, винтажная Java, гопо-bug bounty
+8 6k 18 0

Комментарии 2

AlexZd 12.09.17 в 15:45

Я правильно понимаю, что если я настрою в проекте FA, а потом подключу Google Tag Manager, то по сути я получу аналитику от всех партн
Разумеется после настройки Google Tag Manager в консоли.
Сам пользовался только Answers от Fabric, FA выглядит более гибкой вроде.

xoxol_89 12.09.17 в 15:53

По сути да, все верно. Правда, настройка Google Tag Manager и маппинга событий на другие аналитики не такие тривиальные.

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.