Сервис Аналитики

Модуль отправки аналитики в MEP:

- Unity 3D Analytics SDK - https://cloud.modumlab.com/s/qf67HpPDPFG4c6L

Общие сведения

Сервис аналитики предназначен для хранения, расчета и предоставления данных о прохождении продуктов разного типа. Сервис хранит, как сырую информацию, полученную от клиентов (под ними понимается экземпляр приложения, запущенный на клиентском устройстве), так и рассчитанные **Результаты**. Сведения от клиентов о прохождении продуктов называются Событиями и отправляются клиентом в специально сформированных **Сообщениях**. Сообщения могут содержать сведения о нескольких **Событиях**.

Для объединения действий пользователя или, что равнозначно, Событий по продукту за определенный промежуток времени для правильного расчета **Результата**, вводится понятия **Сеанса**. Например, это вводится для продуктов, содержащих видео-контент, для которых во время просмотра постоянно отправляются **События** о просмотренном фрагменте видео.

В зависимости от типа продукта и **События**, каждое **Событие** может расцениваться как отдельная попытка (для продуктов типа приложений, тестов, ссылок), либо в рамках одного сеанса считаются принадлежащими одной попытки прохождения продукта (для видео и документов). Например, если в одном **Сообщении** отправить десять **Событий** с ответами на тест, сервис аналитики это воспримет, как десять попыток прохождения теста. Тоже касается прохождения продукта с типом **app**.

Общая схема работы сервиса выглядит следующим образом:

1. При создании продукта на портале сервису аналитики передаются необходимые для расчета результатов сведения:
    a. Сведения для расчета баллов по информации из Событий (ответы к тестам, длительность видео и т.д.)
    b. Сведения для определения статуса прохождения и итогового количества баллов (общее количество баллов за задание, пороговое количество баллов для выполнения задания)
    c. Сведения о сроках активности продукта, количестве максимальных попыток его прохождения для пользователя и т.д. Если продукт не активен или превышено максимальное количество попыток, то результат не будет рассчитан.
2. Клиент
    a. Открывает Сеанс аналитики, используя `analytics_token`, см. Ниже, в ответе от сервиса получает уникальный идентификатор сеанса.

   b. Посылает Сообщение с Событиями, содержащими информацию о действиях пользователя сервису аналитики, идентификатор сеанса + `analytics_token` (в заголовке запроса). Например, для теста в информации о Событии посылаются ответы пользователя. Для продукта с типом **app** предусмотрена отправка События с event_id="score" и информацией о набранных баллах, более подробно о формате ниже.
   c. Закрывает Сеанс.
3. Сервис аналитики при получении Сообщения:
   a. Сохраняет События от клиента.
   b. Запускает расчет результатов для продукта на основе данных из Событий.
   c. Запускает расчет результатов для родительских продуктов.

Авторизация для доступ к api

Авторизация запросов осуществляется согласно стандарту [The OAuth 2.0 Authorization Framework: Bearer Token Usage] по токену "`analytics_token`".
Необходимо при запросе метода не забыть отправить заголовок содержащий токен:
`Authorization: Bearer %token%`
В данном токене описывается информация о "клиенте", отправляющем данные.

Структура запроса

Запросы строятся по схеме `/v%version%/%path%`

- `version` – версия API, текущая актуальная 1, т.е. путь `/v1/%path%`
- `path` – адрес метода

Допустим только POST-запросы. В случае GET-запроса сервер вернет ошибку

Формат принимаемых данных

Мы можем передавать параметры на сервер в виде json или параметров-формы. Для этого нужно передать обязательный заголовок Content-type.
для json:

```
Content-type: application/json
```

для формы:

```
Content-type: application/x-www-form-urlencoded
```

Формат ответа

Ответ в любом случае будет отправлен в формате json:

```
{
    "status": "success", // success или error
```

```
    "success": { // при успехе
        "param": "value",
    },
    "error": { // при ошибке
        "name": "Unauthorized",
        "message": "Your request was made with invalid credentials.",
        "code": 0,
        "status": 401,
        "type": "%error_type%"
    }
}
```

Пример ответа для успешного запроса

```
{
    "status": "success",
    "success": {
        "param": "value",
    }
}
```

Пример ответа на запрос с ошибкой

```
{
    "status": "error",
    "error": {
        "name": "Unauthorized",
        "message": "Your request was made with invalid credentials.",
        "code": 0,
        "status": 401,
        "type": "%error_type%"
    }
}
```

Работа с сервисом

Создание Сеанса

При каждом запуске приложения (или симуляции в рамках приложения), создается т.н. “Сеанс аналитики” – это некое (завершенное) действие, проистекающее в рамках условных пределов (например, цикл “запустил приложение – закрыл приложение”).

Для создания Сеанса предусмотрен отдельный метод сервера аналитики.

Методы API

POST /v1/raw/generate-seance

Создание сеанса, параметры не принимаются.

Ответ в случае успеха:

```
{
  "status": "success",
  "success": {
    "seance_id": string
  }
}
```

POST /v1/raw/close-seance

Уведомление сервиса о закрытии сеанса. Обязательные параметры:

- `seance_id: string`

Ответ в случае успеха:

```
{
  "status": "success"
}
```

Особенности:

1. Если закрывать уже закрытый сеанс, вернется ошибка

POST /v1/raw/send

Отправка сведения о событиях в аналитику. События отправляются в сервис сгруппированные в одно сообщение, см. Сообщение с описанием Событий. Сообщение может содержать сведения, как об одном Событии, так и о нескольких.

Ответ в случае успеха:

```
{
  "status": "success"
}
```

Сообщение с описанием Событий

Состоит из

- **seance_id** – уникальный строковой идентификатор Сеанса аналитики, создаваемый (генерируемый) сервером аналитики, служит для группировки событий по некому сеансу (прохождения симуляции / теста и т.п.);
- **events** – массив сведений о Событиях для сервиса аналитики, имеющих следующие поля
    - **event_id** – строковой идентификатор События
    - **data** – объект с данными о Событии (не json-строка, а структура с полями), см. ниже формат для продукта с типом **app**
    - **client_time** – время на клиенте, когда было сгенерировано событие (unixtime)
- **environment** – структура данных об окружении, в котором работает юзер:
    - **device_id** – генерируемый клиентом id устройства.
    - **os** – описание операционной системы:
        - **name** – имя ОС
        - **ver** – версия ОС
    - **device** – описание клиентского устройства:
        - **brand*** – производитель
        - **model** – модель и версия, например, “Galaxy Tab S10”
    - **app** – описание приложения, которое отправляет данные аналитики:
        - **id** – bundle id приложения, например “com.modumlab.mep.app.scenariossandbox”
        - **version** – версия:
            - **code** – порядковый номер версии, например 25
            - **name*** – строковое представление, например “1.3a”
    - **host_app*** – аналогично app, описывает приложение, подпроцессом которого является отправляющее данные приложение (т.е. фактически host_app – это данные лаунчера)

***** – необязательные поля

Внимание

1. Все поля выше, кроме помеченных звездочками являются обязательными.
2. Поле data в описании события это объект (структура) с данными, а не json-строка.

3. Если сеанс (seance_id) не существует или закрыт (isOpened: false), то клиенту вернется ошибка 403 Invalid seance.

Пример запроса:


```
{
  "seance_id": "5c5d9a872b5bc30741d04bcd",
  "events": [
    {
      "data": {
        "score": 99
      },
      "client_time": 1549638389,
      "event_id": "score"
    }
  ],
  "environment": {
    "device_id": "b90badc41e6361c6",
    "os": {
      "name": "Android",
      "version": "7.1.1"
    },
    "device": {
      "idiom": "Phone",
      "model": "Android SDK built for x86",
      "manufacturer": "Google"
    },
    "app": {
      "app_id": "app__surveyjs_3c580e99a7374d2987e8f056a5b2903a",
      "app_version": {
        "app_version_name": "1.0",
        "app_version_code": 11
      }
    },
    "host_app": {
      "app_id": "com.modumlab.mep.gearvrlauncher",
      "app_version": {
        "app_version_name": "1.0",
        "app_version_code": 1
      }
    }
  }
}
```

Внимание

4. Событии типа **score** для приложения с типом **app** означает прохождение задания и вызывает расчет результатов.
   Если в Сообщении будет передан массив из нескольких Событий типа **score** для приложения с типом **app**, то каждое **Событие** будет расценено сервисом, как отдельное прохождение, т.е. можно послать пять отдельных сообщений с этими Событиями, а можно все поместить в одно сообщение.
   Массив из пяти таких Событий вызовет последовательный расчет пяти результатов для продукта и последний будет содержать информацию, что было пять попыток прохождения, точно также как и в случае, если События высылать отдельными Сообщениями.

Структура события score

При прохождении продукта с типом **app**, для подсчета баллов/прохождения внутри платформы нужно отправить Событие event_id = “score”, структура данных События:

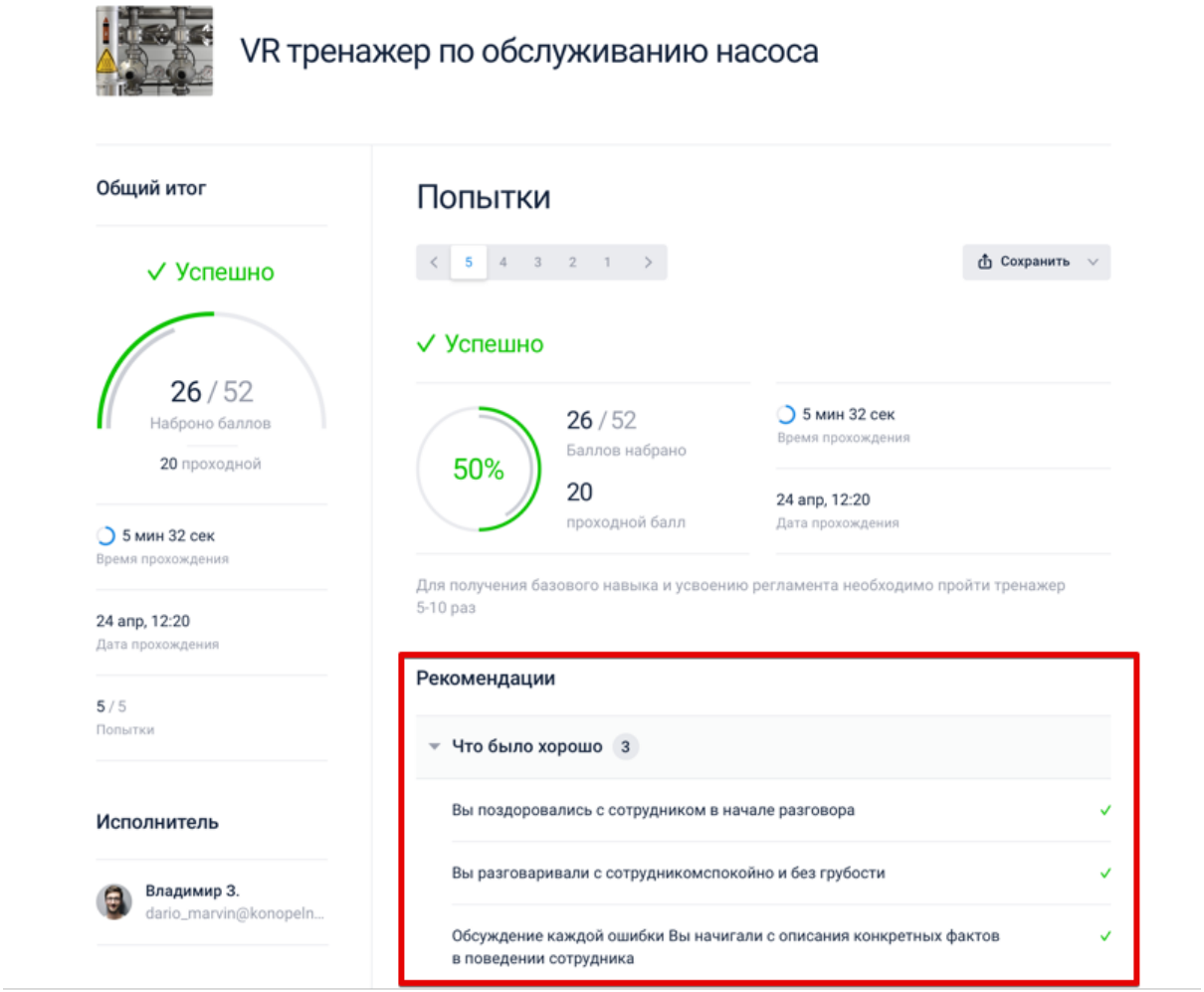
```
{
  "score": <integer from 0 to 100>
}
```

Поле score являются обязательными, другие поля можно передавать, они просто не будут учитываться. В `score` должно передаваться целое число от 0 до 100. Пример Сообщения содержащего сведения о прохождения продукта с типом **app** смотрите выше.

Структура события report

Данное Событие позволяет передать данные об отчете прохождения симуляции, которые универсально будут выведены на платформе.

Простыми словами, при отправке этих данных в аналитику, мы получаем возможность сформировать отчет стандартного вида, который увидит пользователь после прохождения симуляции или его руководитель.



Пример страницы с отчетом (отчет выделен в рамку)

Пример того, как выглядит запрос отправляемый в аналитику, далее будут приводиться только фрагменты данных (содержимое поля events[i].data)


```
{
  "seance_id": "5c5d9a872b5bc30741d04bcd",
  "events": [
    {
      "data": [
        {
          "type": "group",
          "children": [
            {
              "type": "node",
              "title": "Начало беседы",
              "label": "3",
              "progress": {
                "icon": "cross",
                "modifiers": [
                  "bad"
                ]
              },
              "children": [
                {
                  "type": "node",
                  "title": "Вы употребляли большую часть этапов оформления
кредита, но не все",
                  "progress": {
                    "icon": "cross",
                    "modifiers": [
                      "bad"
                    ]
                  }
                }
              ]
            }
          ]
        }
      ],
      "client_time": 1549638389,
      "event_id": "report"
```

```
    }
  ],
  "environment": {
    "device_id": "b90badc41e6361c6",
    "os": {
      "name": "Android",
      "version": "7.1.1"
    },
    "device": {
      "idiom": "Phone",
      "model": "Android SDK built for x86",
      "manufacturer": "Google"
    },
    "app": {
      "app_id": "app__surveyjs_3c580e99a7374d2987e8f056a5b2903a",
      "app_version": {
        "app_version_name": "1.0",
        "app_version_code": 11
      }
    },
    "host_app": {
      "app_id": "com.modumlab.mep.gearvrlauncher",
      "app_version": {
        "app_version_name": "1.0",
        "app_version_code": 1
      }
    }
  }
}
```


Поле data этого события аналитики является массивом отдельных блоков, которые могут поддерживать вложенность. Описание возможных полей блоков, см. ниже.

Поля

type (string)

Тип блока.

См. список допустимых значений поля type

title (string, object)

Заголовок.

label (string, object)

Короткий текст или число

caption (object)

icon (string)

Название иконки.

См. список допустимых значений поля icon

modifiers (array -> string)

Список текстовых модификаторов.

См. список допустимых значений поля modifiers

text (string,object)

Произвольный текст

description (string,object)

Описание.

progress (object)

При использовании этого блока не рекомендуется использовать все возможные подполя, рекомендуется использовать вариации:

- иконка + текст (+ модификаторы)
- значения прогресса (+ модификаторы)

icon (string,object)

Название иконки.

См. список допустимых значений поля icon

modifiers (array -> string)

Список текстовых модификаторов.

См. список допустимых значений поля modifiers

text (string,object)

Произвольный текст

max (int)

Максимальное значение прогресса

current (int)

Набранное значение прогресса

children (array -> object)

Список дочерних объектов

Если блок поддерживает данное поле значит внутрь него можно вложить список блоков. Строгих ограничений на глубину вложенности нет, но мы не рекомендуем вложенность больше 5 блоков. Есть ограничения на то какие типы блоков можно вкладывать в другие блоки, об этом подробнее написано в описании блоков.

Допустимые значения поля type

- group
- node
- description_list
- tag

Допустимые значения поля icon

- check
- cross
- attention

Допустимые значения поля modifiers

- good
- bad
- middle

Допустимые значения текстовых полей title, text, description

Эти блоки могут быть строкой или объектом который оформлен по особым правилам и имеет структуру где первое поле имеет ключ `i18n`. Данный формат используется только для интернационализации и вынесение текстов в словари. Далее в документе более подробно описано.

```
{"i18n": "name", "parameter_1": 7, "parameter_2": "string"}
```

Блоки

Каждый блок состоит из полей. Важно понимать что не обязательно использовать все допустимые поля.

На данный момент наложены ограничения на возможность вкладывать блоки в другие блоки, у каждого блока ниже расписаны типы в которые они могут быть вложены.

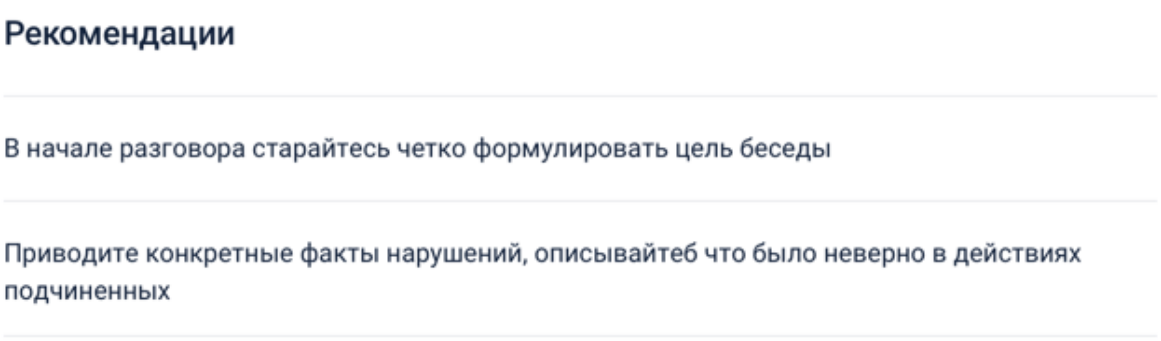
Обязательные поля для всех блоков

- type

Блок типа “group”

Служит для отображения заголовков первого уровня. Не может быть вложен в children и обязательно должен быть на странице.

Пример кода




```
[
  {
    "type": "group",
    "title": "Рекомендации",
    "children": [
      {
        "type": "node",
        "title": "В начале разговора старайтесь четко формулировать цель
беседы"
      },
      {
        "type": "node",
        "title": "Приводите конкретные факты нарушений, описывайте что было
неверно в действиях подчиненных"
      }
    ]
  }
]
```

Блок типа “node”

Универсальный блок реализующий вложенные списки.

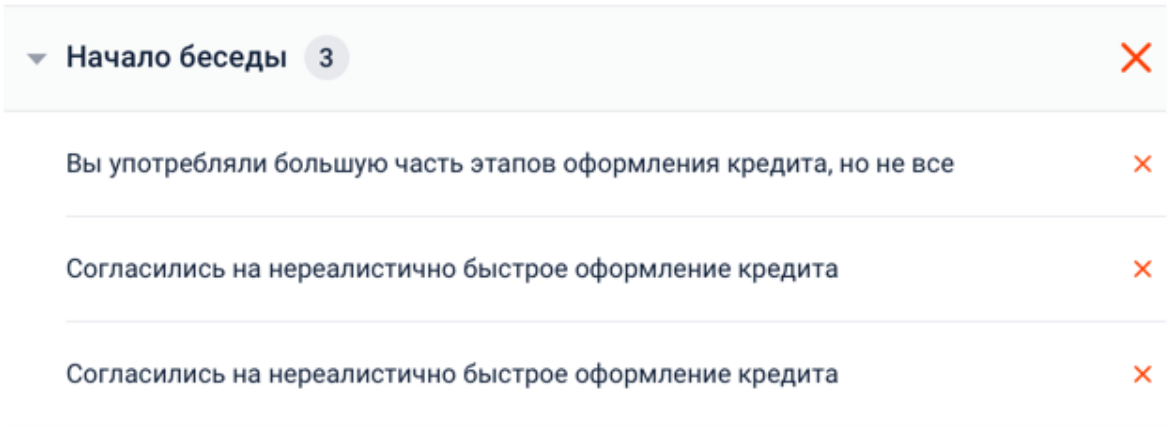
Может быть вложен в блоки типов:

- group
- node

Доступные поля

- label
- caption
- progress
- children

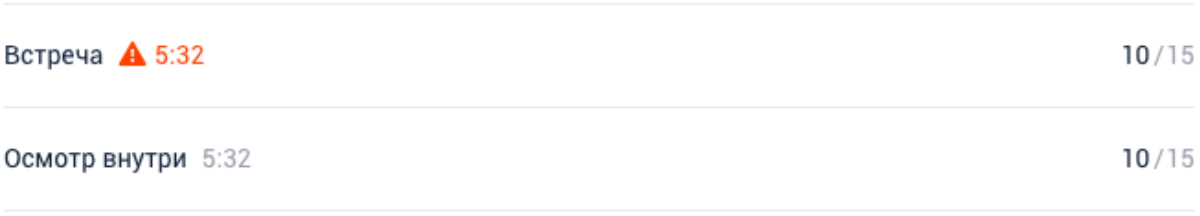
Пример кода #1




```
[
  {
    "type": "group",
    "children": [
      {
        "type": "node",
        "title": "Начало беседы",
        "label": "3",
        "progress": {
          "icon": "cross",
          "modifiers": [
            "bad"
          ]
        },
        "children": [
          {
            "type": "node",
            "title": "Вы употребляли большую часть этапов оформления
кредита, но не все",
```

```
            "progress": {
              "icon": "cross",
              "modifiers": [
                "bad"
              ]
            }
          },
          {
            "type": "node",
            "title": "Согласились на нереалистично быстрое оформление
кредита",
            "progress": {
              "icon": "cross",
              "modifiers": [
                "bad"
              ]
            }
          }
        ]
      }
    ]
  }
]
```

Пример кода #2




```
[
  {
    "type": "group",
    "children": [
      {
        "type": "node",
        "title": "Встреча",
        "progress": {
```

```
          "max": 15,
          "current": 10
        },
        "caption": {
          "icon": "attention",
          "text": "5:32",
          "modifiers": [
            "bad"
          ]
        }
      },
      {
        "type": "node",
        "title": "Осмотр внутри",
        "progress": {
          "max": 15,
          "current": 10
        },
        "caption": {
          "text": "5:32"
        }
      }
    ]
  }
]
```

Блок типа “description_list”

Может быть вложен в блоки типов:

- group

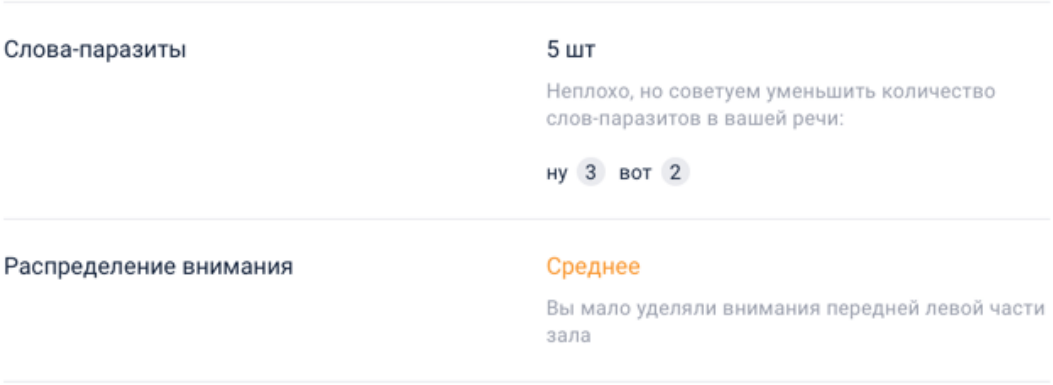
Обязательные поля

- caption

Допустимые поля

- description
- children

Пример кода




```
[
  {
    "type": "group",
    "children": [
      {
        "type": "description_list",
        "title": "Слова-паразиты",
        "caption": {
          "text": "5 шт"
        },
        "children": [
          {
            "type": "tag",
            "title": "ну",
            "label": "3"
          },
          {
            "type": "tag",
            "title": "вот",
            "label": "2"
          }
        ],
        "description": "Неплохо, но советуем уменьшить количество слов-паразитов в вашей речи:"
      },
      {
        "type": "description_list",
        "title": "Распределение внимания",
        "caption": {
          "modifiers": [
```

```
            "middle"
          ],
          "text": "Среднее"
        },
        "description": "Вы мало уделяли внимания передней левой части зала"
      }
    ]
  }
]
```

Блок типа “tag”

Может быть вложен в блоки типов:

- description_list

Допустимые поля

- label

Пример кода

```
[{
  "type": "tag",
  "title": "ну",
  "label": "10"
}]
```

Интернационализация и словари

При создании симуляции платформа позволяет загрузить словарь терминов, который будет использоваться, в том числе, для формирования отчетов. Мы рекомендуем всегда использовать словари, чтобы уменьшить количество данных отправляемых в аналитику и иметь возможность исправлять тексты и поддерживать мультиязычность.

Как выглядит блок с использованием словаря

```
[{
  "type": "tag",
  "title": {"i18n": "results", "n": 7}
}]
```

Как выглядит словарь для русского языка

```
{
  "results": "Ваш результат: {n, number}",
  "cats_on_couch": "На диване {n, plural, =0{нет кошек} =1{лежит одна
кошка} one{лежит # кошка} few{лежит # кошки} many{лежит # кошек}
other{лежит # кошки}}!",
  "today": "Сегодня {n, date,yyyy-MM-dd}",
  "only_text": "Просто очень длинный текст, который описывает важную, но
статичную информацию"
}
```

Числа

```
...
"results": "Ваш результат: {n,number}",
"results_2": "Ваш результат: {n,number,000.00}}",
...
```

Предполагается что n - будет передано как число int или float. Вы можете указать или встроенный формат вывода или использовать свой. Встроенные форматы — это integer, currency, percent

Дата и время

```
...
"today_short": "Сегодня {n, data,short}",
"today": "Сегодня {n, date,yyyy-MM-dd}",
...
```

Предполагается что n - будет передано как timestamp. Вы можете указать или встроенный формат вывода или использовать свой. Встроенные форматы — это short, medium, long и full

Множественное число

```
...
"cats_on_couch": "На диване {n, plural, =0{нет кошек} =1{лежит одна кошка}
one{лежит # кошка} few{лежит # кошки} many{лежит # кошек} other{лежит #
кошки}}!",
...
```

В данном правиле

- `=0` означает ноль;
- `=1` соответствует ровно 1;
- `one` — 21, 31, 41 и так далее;
- `few` — от 2 до 4, от 22 до 24 и так далее;
- `many` — 0, от 5 до 20, от 25 до 30 и так далее;
- `other` — для всех прочих чисел (например, дробных).
- Решётка # заменяется на значение аргумента n.

Для некоторых языков правила могут быть более простыми. Например, для английского будет достаточно указать:

```
...
"cats_on_couch": "There {n, plural, =0{are no cats} =1{is one cat} other{are #
cats}}!
",
...
```

Вариации

```
..
"love": "{name} -- {gender} и {gender, select, женщина{ей} мужчина{ему}
other{ему}} нравится ModumLab!",
..
```

При передаче `{"i18n": "love", "name": "Василий", "gender":"мужчина"}`

Выведет сообщение "Василий — мужчина и ему нравится ModumLab!".

В приведенном выражении, *мужчина* и *женщина* — это возможные варианты пола. На всякий случай, other обработает случай, если значение не совпадает с первыми двумя вариантами. Строки в скобках являются вторичными выражениями и могут быть просто строкой или строкой, содержащей дополнительные указатели.