Система классификации посетителей

Руководство пользователя / интеграции программного блока

Оглавление

1. Термины и сокращения

Термин / Аббревиатура	Значение
БД	База данных.
ОС	Операционная система.
API	Программный интерфейс приложения, предназначенный для взаимодействия со сторонними приложениями или пользователями.
Аккаунт	Учетная запись посетителя сети букмекерских клубов, к которой привязаны его персональные данные и все его кошельки. Одному посетителю может принадлежать только один аккаунт.
Баллы классификации (БК)	Условные значения, которые присваиваются аккаунту при совершении ставок с денежного баланса и используются при расчете класса посетителя. Количество баллов привязано к аккаунту. Сбрасываются по истечении периода накопления.
Гейт-сервис	Приложение, предоставляющее эндпойнты для работы с внешними системами.
Домен	Группа микросервисов, имеющая единую точку входа и не имеющая зависимостей от других доменов на уровне базовой логики или модели данных.
Класс посетителя (Класс аккаунта)	Класс, который присваивается аккаунту посетителя в соответствии с количеством баллов классификации аккаунта.
Кошелек	Учетная запись посетителя букмекерского клуба, к которой привязана информация о балансах и ставках посетителя в этом клубе. В одном клубе у одного посетителя может быть только один кошелек.
Локаль	Набор параметров, определяющий региональные настройки пользовательского интерфейса, такие как язык, страна, часовой пояс, набор символов, формат вывода даты, времени, используемая денежная единица и т. д.
Период накопления баллов классификации	Период, по окончании которого баллы классификации сбрасываются. Составляет один календарный месяц.
Система классификации посетителей	Программный продукт, предназначенный для классификации посетителей сети букмекерских клубов на основании данных о совершенных ими ставках.
Эндпойнт	Шлюз, соединяющий серверные процессы приложения с внешним интерфейсом.

2. О документе

Настоящее руководство предоставляет информацию, необходимую для интеграции программного модуля Система классификации посетителей в информационную систему сети букмекерских клубов:

- Структура и состав программного модуля;
- Обмен сообщениями с внешними системами;
- Настройка параметров Системы классификации.

3. Назначение программного продукта

Система классификации посетителей представляет собой набор серверных приложений, предназначенный для интеграции с другими приложениями и подсистемами как часть информационной системы обработки ставок на спортивные события.

Система классификации посетителей осуществляет распределение посетителей сети букмекерских клубов по классам. Каждому аккаунту посетителя присваивается определенный класс на основании данных о ставках на спортивные события, которые совершает этот посетитель в клубах сети.

Данные о классах посетителей, предоставляемые Системой классификации посетителей, необходимы для корректной работы приложений, предназначенных для:

- сбора и анализа статистических данных о совершении ставок посетителями клубов;
- получения метрик работы клубов.

4. Системные требования

В настоящем разделе перечислены аппаратные и программные требования, необходимые для корректной работы Системы классификации посетителей.

4.1. Минимальные аппаратные требования

Для обеспечения стабильного функционирования Системы классификации посетителей аппаратная часть должна обладать следующими характеристиками:

- Операционная система, способная запускать контейнеры. Предпочтительно ОС *Linux*;
- Система управления контейнерной виртуализацией (*Docker*);
- Подключение к серверу очередей RabbitMQ;
- Количество логических ядер процессора: 4;
- Семейство процессоров: x86-64;
- Частота процессора: 3.0. ГГц;
- Объем установленной памяти: 4 Гб.

4.2. Минимальные требования к сторонним компонентам и/или системам

Для развертывания Системы классификации посетителей должны быть предварительно установлены следующие компоненты окружения:

- *Docker* 24.0.2 (open-source community edition);
- *RabbitMQ* 3.11.21 (Open Source license);
- *PostgreSQL* 14.21 (Open Source license).

Для разработки и модернизации Системы классификации посетителей должны быть использованы следующие программные продукты и языки программирования:

- *VSCode* 1.81 (Open Source license);
- *DBeaver Community* 23.1.5 (Open Source license);
- Платформа *Node.js* 12.
- Язык программирования *JavaScript*.

5. Структура программного блока

Система классификации посетителей включает в себя следующие компоненты (Рис. 1):

- *gate* – приложение для приема, первичной валидации и передачи на сервер внешних запросов;
- *classification-system-api* – приложение для управления процессами расчета классов посетителей;

- *classification-system* – приложение для расчета классов посетителей;
- база данных для хранения баллов классификации и классов посетителей.

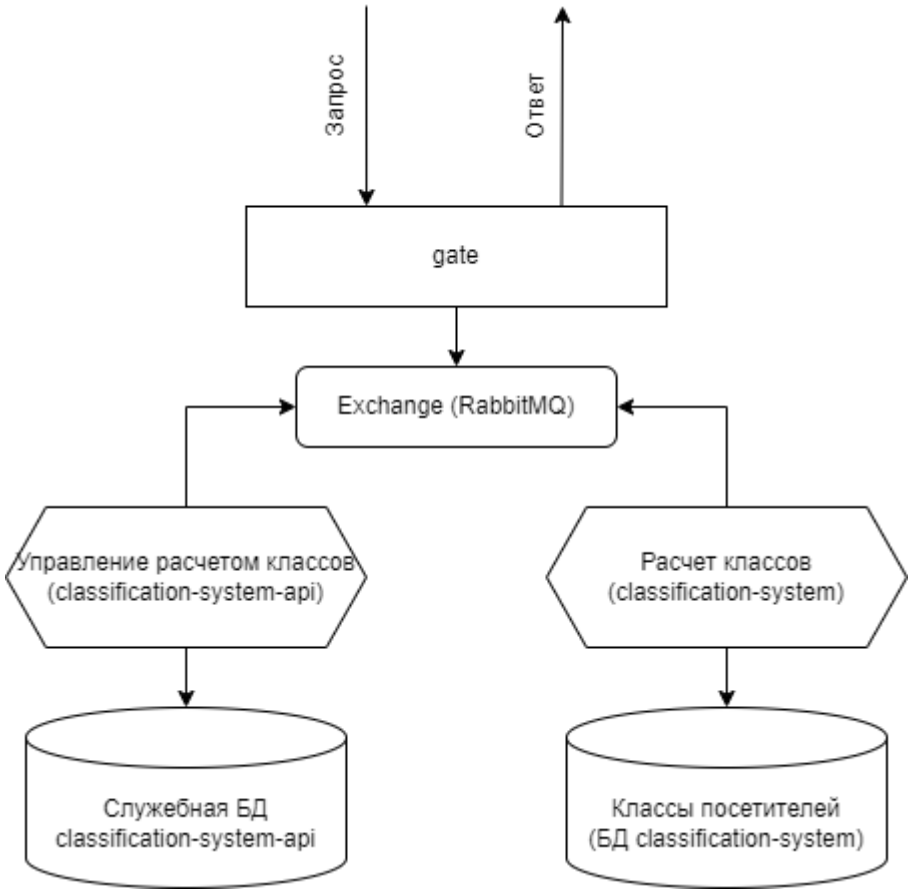

Рис. 1 Компоненты Системы классификации посетителей

6. Обмен сообщениями с внешними системами

Информацию, необходимую для вычисления классов посетителей, Система классификации посетителей получает от внешних систем.

Результаты работы сервиса также отдаются внешним системам.

Эндпойнты для взаимодействия с внешними системами предоставляет сервис *gate* (гейт-сервис). Гейт-сервис выполняет прием и первичную валидацию входящих сообщений. Результаты работы сервисов Системы классификации посетителей также отправляются в гейт-сервис, который возвращает их в качестве ответа на запрос.

В настоящем разделе описаны основные принципы, по которым происходит обмен сообщениями сервиса *gate* с внешними системами.

6.1. Первичная обработка входящих сообщений

Взаимодействие внешних систем с гейт-сервисом происходит одним из двух способов:

- по HTTP-протоколу – путем отправки POST-запроса на эндпойнт */api/v1/rest/*;
- по ws-протоколу– путем отправки HTTP-запроса на эндпойнт */api/v1/ws/*.

Внешнее сообщение должно содержать объект (поле *data*) с необходимой информацией в формате JSON.

Если полученное внешнее сообщение не прошло валидацию, взаимодействие прерывается. В этом случае сервис *gate* не отправляет никаких внутренних запросов.

Гейт-сервис возвращает коды ошибок:

- для POST-сообщений:
  - *400* – если исходное сообщение не прошло валидацию;
  - *500* – если от внутренних сервисов получен ответ с ошибкой;
- для GET-сообщений:
  - если исходное сообщение не прошло валидацию:
    - код, переданный в параметре *errorResponseStatus* – если был задан;
    - *400* – если в параметре *errorResponseStatus* не был задан код ошибки;
  - если от внутренних сервисов получен ответ с ошибкой:
    - код, переданный в параметре *errorResponseStatus* – если был задан;
    - *500* – если в параметре *errorResponseStatus* не был задан код ошибки.

6.2. Внешние сообщения

Описание формата входящих внешних сообщений, которые принимает гейт-сервис:

- *id* – идентификатор сообщения (строка). Всемирно уникальный идентификатор, представляет собой 16-байтный (128-битный) номер. Числа в шестнадцатеричной системе счисления, разделённые дефисами на пять групп в формате 8-4-4-4-12 (всего 36 символов);
- *locale* – предпочитаемая локаль (строка). Строго ***ru***;
- *domain* – название домена, к которому обращается запрос (строка) – ***bet*** или ***classification***;
- *event* – название метода, к которому обращается запрос (строка). Если сообщение успешно проходит валидацию, сервис *gate* публикует его в очередь, название которой указано в этом параметре;
- *data* – основная информация сообщения (объект). Свойство должно отсутствовать, если свойство *errors* существует;
- *errors* – массив с объектами ошибок (массив в формате JSON-RPC 2.0). Свойство должно отсутствовать, если свойство *data* существует;
- *ttl* – предельное время ожидания ответа приложений в миллисекундах (целое число, опционально). Логика обработки параметра:
    - если входящее значение *ttl* больше значения *ttl* сервиса *gate* (по умолчанию 35 мс, задается конфигурационным параметром сервиса), то входящее значение *ttl* будет переопределено гейт-сервисом на значение его *ttl*;
    - при превышении ожидания указанного в *ttl* времени гейт-сервис вернет ошибку;
    - если параметр *ttl* не задан, то в качестве *ttl* берется значение *ttl* гейт-сервиса;
    - если параметр *ttl* задан равным 0, при этом на гейт-сервисе допускается *ttl* равный 0 (по умолчанию допускается), то гейт-сервис непосредственно после успешной публикации сообщения в домен возвращает пустой позитивный ответ (в ответе присутствует свойство *data*), а публикуемое в бизнес-домен сообщение становится

результирующим событием с отсутствующими свойствами *expires* и *replyTo*;

- *traceparent* – идентификатор внешнего родительского трейса в рамках реализации *opentracing* спецификации (строка, опционально);
- *tokens* – токены клиента (объект);
    - *id* – содержит идентификатор токена, необходимый для доступа к API (строка);
    - *access* – тип токена. Строго ***access***.

Пример входящего внешнего сообщения:

```
{
   "id": "ad6b9f3f-038c-4a55-84d8-829c415a238f",
   "locale": "ru",
   "domain": "bet",
   "event": "betsCreated",
   "data": { },
   "tokens": {
             "id":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7InBl
cm1p5JZCI6N",
             "access": "access"
   }
}
```

Пример ответа гейт-сервиса на внешнее сообщение:

```
{
    "id": "1740dc18-03f4-4f9a-b3f8-347de1581a5a",
    "domain": "bet",
    "event": "betsCreated",
    "data": {}
}
```

6.3. Внутренние сообщения

Результаты работы сервисов Системы классификации посетителей поступают в сервис *gate* во внутренних сообщениях.

Внутренние сообщения передаются по rmq-интерфейсу (RabbitMQ).

В поле *data* передается основная информация сообщения в формате JSON.

Описание формата внутренних сообщений:

- *id* – идентификатор сообщения (строка);
- *gid* – глобальный идентификатор сообщения (строка). Не должен быть переопределен. Вычисляется с использованием uuid v.5 по свойствам сообщения *id*, *locale*, *features*, *event*, *data*, *errors*, *domain* и *ttl*;
- *tid* – идентификатор запроса (строка). Если отсутствует во входящем сообщении, генерируется;
- *cid* – идентификатор соединения (строка, опционально). Используется, например, для локализации ws-соединения в гейт-сервисе;
- *expires* – абсолютное время протухания (целое число, опционально). Вычисляется как *<текущее время> + min(ttl, maxTtl)*;
- *features* – массив информационных флагов (массив). Может быть задан гейт-сервисом;
- *locale* – предпочитаемая локаль (строка);
- *domain* – название домена, в который будет опубликовано событие, содержащее данное сообщение (строка);
- *event* – событие (строка);
- *data* – основная информация сообщения (объект или массив, предпочитаемый тип «объект»). Свойство должно отсутствовать, если свойство *errors* существует:
    - *previous* – содержит предыдущие данные, достаточные для отката примененных изменений на сущности (объект или массив, опционально);
- *errors* – массив с объектами ошибок (массив в формате JSON-RPC 2.0). Свойство должно отсутствовать, если свойство *data* существует;
- *ttl* – свойство *ttl*, полученное из внешнего сообщения (целое число, опционально);

- *traceparent* – идентификатор родительского трейса в рамках реализации *opentracing* спецификации (строка, опционально);
- *saga* – содержит данные по саге (объект, опционально):
    - *sid* – идентификатор саги (строка). Уникален совместно со свойством *saga.state*;
    - *name* – название саги (строка);
    - *state* – состояние саги (строка). Перечень состояний саг может быть специфичен для различных саг;
    - *expires* – абсолютное время протухания саги (число, опционально);
- *src* – данные сервиса, сгенерировавшего сообщение (объект);
    - *name* – название сервиса (строка);
    - *version* – версия сервиса (объект):
        - *major* – мажорная версия сервиса (целое число);
        - *minor* – минорная версия сервиса (целое число);
        - *patch* – патч-версия сервиса (целое число);
- *replyTo* – данные для отправки ответа (объект, опционально):
    - *exchange* – RabbitMQ exchange (строка);
    - *routingKey* – название маршрута (строка, опционально). Если маршрут не задан, то используется маршрут, соответствующий содержимому event-свойства;
    - *event* – событие, переопределяющее значение event-свойства сообщения (строка, опционально);
- *tokens* – содержит токены клиента (объект):
    - *id* – идентификатор токена (строка);
    - *access* – содержит токен доступа для получения сессионных данных.

Пример входящего внутреннего сообщения:

```
{
  "id": "b1292d79-0741-4fbc-8460-8149602987a7",
  "gid": "t1292d79-0741-4fbc-8460-8149602987a45",
  "tid": "f3292d79-0741-4fbc-8460-8149602987a46",
  "cid": "a3292d79-0741-4fbc-8460-8149602987s5",
  "expires": 1572444605,
```

```
  "features": ["none:ru"],
  "locale": "ru",
  "domain": "freeBet",
  "event": "processFreeBetTransaction",
  "data": {
    "id": "23223",
    "amount": 230.45,
    "action": "in",
    "key": 1.2
  },
  "ttl": 1000,
  "traceparent": "00-1ca273a7a235149c14009955982c925c-
3fb610b6f0eb923a-01",
  "saga": {
    "sid": "c3292d79-0741-4fbc-8460-8149602987s6",
    "name": "processFreeBetTransactionSaga",
    "state": "checkFreeBetPending",
    "expires": 1572444605
  },
  "src": {
    "name": "freeBetGate",
    "version": {
      "major": 1,
      "minor": 0,
      "patch": 1
    }
  },
  "replyTo": {
    "exchange": "gate",
    "routingKey": "1.0.1",
    "event": "returnResponse"
  },
  "tokens": {
    "id": "LHBhMHmvKndfJHvKHV.LHBhMHmvKndfJHvKHV.LHBhMHmvKndfJHvKHV",
    "access": "a4292d79-0741-4fbc-8460-8149602987s9"
  }
}
```

7. Обработка ставок для расчета класса посетителя

Сервис *gate* принимает сообщения, содержащие батчи (массивы) ставок, а также сообщения с массивами результатов ставок (выигрыши или проигрыши).

Соответствующие события должны быть опубликованы в домене ***bet***.

Сервис *classification-system-api* обрабатывает эти события с помощью методов:

- *betsCreated*;
- *winsCreated*;
- *sportWinsCreated*;
- *sportLossCreated*;
- *sportBetsCreated*.

Перечисленные методы принимают следующие входные параметры:

- *bets* – батч ставок (массив):
    - *walletId* – идентификатор кошелька (число);
    - *hallId* – идентификатор клуба (число);
    - *betId* – идентификатор ставки (число);
    - *amount* – сумма ставки (число);
    - *balanceId* – идентификатор баланса, с которого делалась ставка (1 – денежный баланс, 2 – бонусный баланс, 3 – фрибет-баланс);
    - *gameId* – идентификатор события (число);
    - *gameKindId* – идентификатор вида события (число);
    - *serverId* – идентификатор сервера (число);
    - *accountId* – идентификатор аккаунта (число);
    - *dttm* – дата и время создания ставки (строка, необязательный параметр);
    - *meta* – дополнительные данные (объект, необязательный параметр).

Выходных параметров у перечисленных методов нет. Результатом их работы становится начисление баллов классификации и изменение класса аккаунтов по результатам обработки информации из массива.

8. Получение класса посетителя по номеру аккаунта

Метод *getClassificationInfo* предоставляет информацию о классе посетителя по номеру аккаунта.

Метод принимает следующие входные параметры:

- *accountId* – идентификатор аккаунта (строка);
- *dt* – дата, за которую нужно получить информацию о классе посетителя (строка, необязательный параметр, по умолчанию текущая дата).

Метод *getClassificationInfo* возвращает выходные параметры:

- *accountId* – идентификатор аккаунта (строка);
- *dt* – учетная дата (строка, содержит учетный месяц и год);
- *points* – текущее количество баллов классификации (число);
- *statusId* – идентификатор текущего класса аккаунта (число);
- *statusName* – название текущего класса аккаунта (строка);
- *nextPoints* – количество баллов классификации, которое должен набрать аккаунт для достижения следующего класса (число);
- *nextStatusId* – идентификатор следующего класса (число);
- *nextStatusName* – название следующего класса (строка).

9. Отчеты о начислении баллов классификации

Отчеты о начислении баллов классификации предоставляют статистическую информацию о начислении баллов классификации на аккаунты посетителей (например, по дате или по номеру клуба). В дальнейшем эта информация может быть использована другими сервисами или представлена для табличного отображения клиентскими приложениями (например, на сайте аналитики).

Отчеты о начислении баллов классификации возвращают методы сервиса *classification-system-api*:

- *supportDataByDateRequested* – отчет о начисленных баллах классификации по номеру кошелька;
- *getClassificationSystemsStatusHistory* – история изменений класса посетителя.

Соответствующие запросы публикуются в домене *classification*.

9.1. Отчет о начисленных баллах классификации по номеру кошелька

Метод *supportDataByDateRequested* возвращает данные о каждом начислении баллов классификации на аккаунт в течение отчетного периода.

Метод принимает входные параметры:

- *accountId* – идентификатор аккаунта (строка);
- *fromDatetime* – время начала отчетного интервала (строка);
- *toDatetime* – время окончания отчетного интервала (строка);
- *page* – номер страницы (число);
- *pageSize* – количество записей на странице (число).

Массив возвращает параметры:

- *bets* – информация по начисленным баллам классификации (массив объектов):
    - *gameKindId* – идентификатор вида спортивного события (число);
    - *betId* – идентификатор ставки (число);
    - *betTypeId* – код типа ставки (1 - ординар, 2 - система, 3 - экспресс);
    - *amount* – сумма ставки (число);
    - *points* – количество начисленных баллов классификации (число);
    - *hallId* – номер клуба, в котором была совершена ставка (число);
    - *datetime* – дата и время начисления (строка);
    - *count* – количество объектов в ответе (число).

9.2. История изменений класса аккаунта

Метод *getClassificationSystemsStatusHistory* возвращает историю всех изменений класса аккаунта в течение отчетного периода.

Метод принимает параметры:

- *accountId* – идентификатор аккаунта (строка);
- *beginDt* – дата начала отчетного периода (дата);
- *endDt* – дата окончания отчетного периода (дата).

Метод возвращает параметры:

- *dt* – дата изменения класса аккаунта (дата в формате YYYY-MM-01);
- *statusId* – идентификатор класса аккаунта (число);
- *statusName* – название класса аккаунта (строка);
- *createDttm* – дата и время изменения (строка);
- *system* – идентификатор разрешения на изменения (permissionId).

10. Принудительное изменение класса посетителя

Принудительное изменение класса посетителя вне зависимости от количества накопленных на аккаунте баллов может потребоваться при возникновении ошибок в стандартном расчете класса. При этом можно указать, в каком месяце будут применены изменения – в текущем или в следующем.

Действие выполняется с помощью метод *setClassificationStatus*. Соответствующее событие публикуется в домен *classification*.

Метод принимает параметры:

- *accountId* – идентификатор аккаунта (строка);
- *statusId* – идентификатор класса аккаунта, который требуется установить;
- *nextMonth* – логическая переменная:
    - *false* – класс аккаунта будет действовать в текущем календарном месяце;
    - *true* – класс аккаунта будет действовать в следующем календарном месяце.

Выходных параметров у метода нет. Результатом работы метода становится изменение класса посетителя в указанном месяце.

11. Настройка параметров Системы классификации посетителей

В настоящем разделе приведены способы настройки параметров Системы классификации для получения наиболее точных статистических данных (например, изменение коэффициентов для учета вклада различных типов спортивных событий).

При расчете классов посетителей действуют следующие параметры:

- Таблица видов спортивных событий с коэффициентами для расчета БК – см. 11.1;
- Сетка классов посетителей – см. **Ошибка! Источник ссылки не найден.**.

11.1. Таблица коэффициентов для расчета баллов классификации

Начисление баллов классификации на аккаунт выполняется отдельно для каждого вида события, на которое с этого аккаунта совершались ставки.

Количество баллов классификации рассчитывается следующим образом: сумма ставки умножается на коэффициент, соответствующий виду события, и округляется в меньшую сторону.

Справочник видов событий с коэффициентами для расчета баллов классификации (БК) хранится в БД сервиса *classification-system* в таблице *game_kind_points*.

Изменения в таблице могут производиться путем прямого запроса к базе данных.

11.2. Сетка классов посетителей

В справочнике классов посетителей для каждого класса указывается:

- идентификатор класса;
- название класса;
- является ли класс посетителя VIP-классом.

Справочник классов хранится в таблице *statuses* сервиса *classification-system*.

В таблице *status_points* для каждого класса указано:

- минимальное количество баллов классификации, по достижении которого присваивается класс;
- максимальное количество баллов классификации в классе (являющееся также минимальным для следующего класса).

В приложении *classification-system-api* имеется метод *setClassificationStatusPoints*, выполняющий изменения минимального количества БК. К методу может обращаться, например, клиентское приложение. Запрос на изменение публикуется в домене ***classification***.

Метод *setClassificationStatusPoints* принимает входные параметры:

- *statuses* – массив объектов с классами:
- *statusId* – идентификатор класса (число);
- *minPoints* – минимальное значение баллов классификации (число).

Выходных параметров у метода нет. Результатом работы становится изменение минимальных баллов классификации для указанных *statusId*.

Изменения начинают действовать начиная со следующего месяца.

Получить текущие настройки классов посетителей можно с помощью метода *getClassificationStatusPoints*. Запрос на изменение публикуется в домене *classification*.

Метод принимает входные параметры:

- *limit* – максимальное количество возвращаемых записей (число);
- *dt* – дата, за которую необходимо получить действующие классы (строка, необязательный параметр, по умолчанию текущая дата).

Метод возвращает массив объектов со следующими свойствами:

- *statusId* – идентификатор класса (число);
- *statusName* – название класса (строка);
- *minPoints* – минимальное количество баллов классификации, необходимое для присвоения класса (число);
- *maxPoints* – максимальное количество баллов в классе (число);
- *vip* – является ли класс vip-классом (логическая переменная).