Руководство администратора ПО

«Скоринг активности пользователя»

1. ВВЕДЕНИЕ

1.1. Область применения

Настоящий документ предназначен для администраторов, обеспечивающих установку, настройку, эксплуатацию и сопровождение программного обеспечения «Скоринг активности пользователя» (далее — Сервис).

Руководство описывает порядок развёртывания Сервиса в контейнерной инфраструктуре, параметры конфигурирования через переменные окружения, требования к окружению выполнения, порядок администрирования, мониторинга и контроля работоспособности, а также взаимодействие с предоставляемым JSON-RPC API.

1.2. Перечень выполняемых функций администратора/оператора

В перечень выполняемых функций администратора Системы входят:

- Развёртывание Сервиса и PostgreSQL
- Настройку переменных окружения и секретов
- Контроль доступности (health-check)
- Контроль метрик и журналов
- Управление подключениями к БД
- Выполнение обслуживающих операций через API
- Резервное копирование и восстановление

1.3. Уровень подготовки администратора/оператора

Администратор, выполняющий установку и сопровождение Сервиса, должен обладать навыками работы с контейнерными платформами Docker или Kubernetes, понимать принципы сетевого взаимодействия сервисов и публикации HTTP-сервисов, уметь администрировать СУБД PostgreSQL на уровне создания пользователей, настройки подключений и выполнения резервного копирования. Кроме того, администратор должен иметь опыт анализа журналов работы приложений и мониторинговых метрик для выявления и устранения неисправностей.

1.4. Перечень документации

В состав документации, с которой необходимо ознакомиться администратору Системы входят:

- Функциональная спецификация системы
- Техническое задание на разработку системы
- Настоящее руководство администратора

2. УСТАНОВКА СИСТЕМЫ

2.1. Системные требования

Сервис предназначен для развёртывания в контейнерной среде оркестрации (например, Kubernetes или совместимых платформ) и функционирует как один экземпляр приложения, взаимодействующий с внешней СУБД PostgreSQL. Развёртывание предполагает наличие контейнерного рантайма и средств оркестрации, обеспечивающих запуск контейнеров, сетевое взаимодействие между сервисами и управление конфигурацией через переменные окружения и секреты.

Минимальная конфигурация вычислительных ресурсов указана для одного экземпляра Сервиса (одного pod/контейнера) и включает не менее двух виртуальных ядер процессора, 2 ГБ оперативной памяти и около 10 ГБ дискового пространства для хранения данных базы и журналов. При росте нагрузки допускается горизонтальное масштабирование приложения средствами оркестратора, а также увеличение ресурсов контейнера.

Сервис реализован на языке программирования Go и поставляется в виде контейнерного образа, не требующего установки дополнительных библиотек на узле кластера. Для хранения данных используется внешняя СУБД PostgreSQL версии 16 или выше, развёрнутая как отдельный сервис кластера либо предоставляемая управляемой инфраструктурой. Развёртывание и администрирование PostgreSQL не входят в рамки настоящего документа и должны выполняться в соответствии с отдельной эксплуатационной документацией на используемую СУБД.

2.2. Порядок установки

1. Смонтируйте диск с дистрибутивом в папку /mnt
2. Скопируйте из дистрибутива исходники из папки /mnt в папку /opt/*gamepulse*
3. Отредактируйте файл docker-compose.yml в соответствии с разделом 3.2 данного документа
4. Смените текущую папку на /opt/*gamepulse* и выполните команду: docker compose up -d --build
5. Проверьте работоспособность системы.

3. НАСТРОЙКА СИСТЕМЫ

3.1. Общие сведения

Сервис предоставляет HTTP-интерфейс взаимодействия по протоколу JSON-RPC 2.0.

В системе предусмотрены две точки доступа:

- публичная — для методов, не требующих авторизации;
- приватная — для методов администрирования и записи данных.

Приватные методы защищены ключом доступа. Ключ передаётся в HTTP-заголовке:

Authorization: Bearer <ACCESS_KEY>

Отсутствие ключа или передача неверного значения приводит к отклонению запроса.

Сервис является stateless-приложением и хранит состояние исключительно в базе данных PostgreSQL. Все экземпляры приложения равнозначны и могут быть масштабированы средствами оркестрации контейнеров.

3.2. Конфигурируемые параметры

Настройка Сервиса выполняется исключительно через переменные окружения контейнера.

Параметры сетевого взаимодействия

HOST - Адрес и порт HTTP-сервера API. Определяет интерфейс, на котором сервис принимает JSON-RPC запросы. Значение по умолчанию: :80

METRICS_PORT - Адрес и порт вспомогательного HTTP-сервера проверки работоспособности. Используется для readiness/liveness-probe и мониторинга. Значение по умолчанию: :3000

Параметры авторизации

ACCESS_KEY - Секретный ключ доступа к приватным методам API. Должен совпадать у всех клиентов, выполняющих административные операции и запись событий. Отсутствие значения блокирует выполнение приватных методов.

Параметры подключения к PostgreSQL

DB_HOST - Имя хоста или сервиса PostgreSQL.

DB_PORT - TCP-порт PostgreSQL. Значение по умолчанию: 5432

DB_USER - Имя пользователя базы данных, используемого приложением.

DB_PASSWORD - Пароль пользователя базы данных.

DB_NAME - Имя базы данных, в которой расположена схема Сервиса.

Параметры пула соединений

DB_MAX_OPEN_CONNECTIONS - Максимальное количество одновременно открытых соединений к БД. Ограничивает нагрузку на PostgreSQL. Значение по умолчанию: 30

DB_MAX_IDLE_CONNECTIONS - Максимальное количество неактивных соединений, удерживаемых пулом. Позволяет уменьшить накладные расходы на повторное подключение.

Значение по умолчанию: 30

DB_CONN_MAX_TIME - Максимальное время жизни соединения. По истечении времени соединение закрывается и создаётся заново. Значение по умолчанию: 3m

Параметры логирования и метрик

RPC_LOGGING_ENABLED - Включает журналирование вызовов JSON-RPC методов в стандартный вывод контейнера. Используется для диагностики и аудита операций. Значение по умолчанию: false

RPC_METRICS_ENABLED - Включает сбор внутренних метрик выполнения JSON-RPC методов. Метрики становятся доступны через endpoint /metrics. Значение по умолчанию: false

4. ОПИСАНИЕ API

4.1 Общие сведения

Сервис предоставляет интерфейс взаимодействия по протоколу JSON-RPC 2.0 поверх HTTP.

Точки доступа:

- /public — публичные методы (без авторизации)
- / — приватные методы (требуется авторизация)

Обязательные заголовки запроса:

- Content-Type: application/json
- Authorization: Bearer <ACCESS_KEY> (только для приватных методов)

Формат ответа соответствует стандарту JSON-RPC 2.0.

4.2 Методы API

4.2.1 ping (публичный)

Проверка доступности сервиса.

Метод не принимает параметров.

Пример запроса

```
{ "jsonrpc": "2.0", "method": "ping", "id": 1 }
```

Пример ответа

```
{ "jsonrpc": "2.0", "result": "pong", "id": 1 }
```

4.2.2 scoring.trackEvent (приватный)

Регистрация события активности пользователя.

Параметры

- user_id (string) - Уникальный обезличенный идентификатор пользователя в системе источнике.
- event_type (string) - Тип события активности пользователя. Поддерживаемые значения:
    - login — вход пользователя

    - logout — выход пользователя
    - session_start — начало сессии
    - session_end — завершение сессии
    - game_start — начало игровой активности
    - game_end — завершение игровой активности
    - bet — совершение ставки
    - win — выигрыш
    - lose — проигрыш
    - conversion — целевое действие
    - custom — пользовательское событие
- ts (string, RFC3339) - Дата и время возникновения события в UTC. Используется при агрегации статистики по дням.
- metadata (object) - Дополнительные параметры события в формате JSON. Содержимое не валидируется сервисом и сохраняется как есть. Используется источником для передачи служебной информации (ip, сумма, валюта, игра и т.п.).

Пример запроса

```
{
 "jsonrpc":"2.0",
 "method":"scoring.trackEvent",
 "id":2,
 "params":{
  "user_id":"user_001",
  "event_type":"login",
  "ts":"2026-02-18T10:00:00Z",
  "metadata":{"ip":"203.0.113.10"}
 }
}
```

Возвращает идентификатор созданной записи события.

4.2.3 scoring.getScore (приватный)

Получение текущих значений скоринга пользователя.

Параметры

- user_id (string) - Идентификатор пользователя, для которого требуется получить текущие показатели.

Результат: Объект со значениями:

- engagement_score — вовлечённость

- retention_score — удержание
- activity_score — активность
- volatility_score — нестабильность поведения
- monetary_score — денежная активность
- updated_at — время последнего пересчёта

Пример запроса

{ "jsonrpc":"2.0", "method":"scoring.getScore", "id":3, "params":"user_001" }

4.2.4 scoring.getScoreHistory (приватный)

Получение истории скоринга по дням.

- Параметры
- user_id (string) - Идентификатор пользователя.
- from (string) - Начало периода выборки. Допускается формат YYYY-MM-DD или RFC3339. Если не задан — используется минимально возможная дата.
- to (string) - Конец периода выборки. Если не задан — используется текущее время.

Результат - Массив дневных значений скоринга пользователя.

Пример запроса

{
"jsonrpc":"2.0",
"method":"scoring.getScoreHistory",
"id":4,
"params":{"user_id":"user_001","from":"2026-02-01","to":"2026-02-18"}
}

4.2.5 scoring.recalculateScore (приватный)

Пересчёт статистики и скоринга пользователя на основании накопленных событий.

Используется:

- после изменения формулы скоринга
- после исправления данных
- при восстановлении базы

Параметры

- user_id (string) - Идентификатор пользователя, для которого выполняется полный пересчёт.

Результат - true при успешном завершении операции.

Пример запроса

{ "jsonrpc":"2.0", "method":"scoring.recalculateScore", "id":5, "params":"user_001" }

4.2.6 scoring.getTopUsers (приватный)

Получение списка пользователей с наибольшими значениями скоринга.

Параметры

- limit (integer) - Максимальное количество пользователей в ответе. Если не задано — используется значение 10.
- type (string) - Тип скоринга, по которому выполняется сортировка:
    - engagement
    - retention
    - activity
    - volatility
    - monetary

  Если не задан — используется activity.

Результат - Массив объектов скоринга пользователей.

Пример запроса

```
{
  "jsonrpc":"2.0",
  "method":"scoring.getTopUsers",
  "id":6,
  "params":{"limit":10,"type":"activity"}
}
```

4.2.7 system.getConfig (приватный)

Получение текущей конфигурации формулы скоринга.

Метод не принимает параметров.

Результат - JSON-объект конфигурации расчёта скоринга.

4.2.8 system.updateConfig (приватный)

Обновление конфигурации формулы скоринга.

После изменения конфигурации требуется пересчитать пользователей методом scoring.recalculateScore.

Параметры

Метод принимает массив из одного объекта конфигурации.

Пример структуры:

- max_score (integer) - Максимально возможное значение скоринга.
- analysis_window_days (integer) - Период анализа активности пользователя в днях.
- normalization_mode (string) - Режим нормализации значений:
    - clamp
    - scale
    - none

Пример запроса


```
{
  "jsonrpc":"2.0",
  "method":"system.updateConfig",
  "id":7,
  "params":[{"max_score":100,"analysis_window_days":30,"normalization_mode":"clamp"}]
}
```


Результат - true при успешном обновлении конфигурации.