Руководство администратора ПО

«Система управления промо-кодами»

1. Введение

1.1. Область применения

Настоящий документ предназначен для администраторов, эксплуатирующих программное обеспечение «Система управления промо-кодами» (далее — «Система»). Руководство содержит порядок установки, настройки, администрирования и взаимодействия с API Системы.

1.2. Перечень выполняемых функций администратора/оператора

В перечень выполняемых функций администратора Системы входят:

- Установка и настройка Системы в контейнерной среде
- Настройка переменных окружения и подключение к базе данных PostgreSQL
- Мониторинг работы Системы и анализ логов
- Восстановление работоспособности при сбоях
- Управление кампаниями и промо-кодами через API
- Резервное копирование и восстановление данных
- Настройка параметров безопасности (ключи доступа, TTL)

1.3. Уровень подготовки администратора/оператора

Администратор/оператор (далее по тексту Администратор) Системы должен уметь:

- Работать с Docker и Docker Compose
- Настраивать базу данных PostgreSQL
- Использовать командную строку Linux
- Работать с JSON-форматами и REST/JSON-RPC API

Рекомендуемая численность персонала для эксплуатации Системы — 1 штатная единица.

1.4. Перечень документации

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

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

2. Установка Системы

2.1. Системные требования

2.1.1. Аппаратные требования

- Количество логических ядер процессора: 2
- Семейство процессоров: x86/x64
- Частота процессора: 2.0 ГГц
- Объем установленной памяти: 4 Гб
- Дисковое пространство: 10 Гб (с учетом БД)

2.1.2. Программные требования

- ОС: Debian 11+ или аналогичная, поддерживающая Docker
- Docker: 20.10+
- Docker Compose: 2.0+
- PostgreSQL: 16+
- Дополнительно (опционально):
    - Prometheus для сбора метрик
    - Grafana для визуализации метрик
    - Loki для централизованного логирования

2.1.3. Язык программирования

Система разработана на GoLang 1.24.

2.2. Порядок установки

1. Смонтируйте диск с дистрибутивом в папку /mnt
2. Скопируйте из дистрибутива исходники из папки /mnt в папку /root
3. Отредактируйте файл docker-compose.yml, в соответствии с пунктами 3.2 данного документа
4. Смените текущую папку на /root и выполните в ней команду
docker compose -up -d --build

3.1. Общие сведения

Система использует JSON-RPC API для взаимодействия. Все приватные методы требуют передачи ключа доступа (ACCESS_KEY) в заголовке `X-API-Key`.

3.2. Конфигурируемые параметры

Для корректной работы модуля обработки запросов, необходимо настроить для него следующие переменные окружения:
Обязательные параметры:

- HOST — адрес и порт, который слушает сервис (формат: :порт)
- DB_HOST — хост базы данных PostgreSQL
- DB_PORT — порт базы данных (по умолчанию: 5432)
- DB_USER — пользователь базы данных
- DB_PASSWORD — пароль пользователя базы данных

- DB_NAME — имя базы данных
- ACCESS_KEY — ключ доступа к API (обязательный параметр)

Опциональные параметры:

- METRICS_PORT — порт для метрик healthz (по умолчанию: :3000)
- DB_MAX_OPEN_CONNECTIONS — максимальное количество открытых соединений (по умолчанию: 30)
- DB_MAX_IDLE_CONNECTIONS — максимальное количество неактивных соединений (по умолчанию: 30)
- DB_CONN_MAX_TIME — максимальное время жизни соединения (по умолчанию: 3m)
- LOG_LEVEL — уровень логирования (по умолчанию: info)

4. Описание API

4.1. Общие сведения

Система предоставляет JSON-RPC API по HTTP:

- Все методы, кроме ping, доступны по адресу /
- Доступ к API защищен с помощью ключа доступа, передаваемого в заголовке X-API-Key
- Метод ping доступен по адресу /public
- Заголовок Content-Type должен быть application/json

4.2. Методы API

4.2.1. ping

Проверка доступности сервиса.
Запрос:

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

Ответ:

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

4.2.2. campaigns.create

Создание новой кампании.
Запрос:

```
{
   "jsonrpc": "2.0",
   "method": "campaigns.create",
   "id": 2,
   "params": {
      "name": "Летняя распродажа 2024",
      "description": "Акция на летние товары",
      "code_template": "SUMMER24-{N:6}",
      "valid_from": "2024-06-01T00:00:00Z",
      "valid_to": "2024-08-31T23:59:59Z",
      "metadata": {
         "discount": 15,
         "min_amount": 1000
      }
   }
}
```

Ответ:

```
{
   "jsonrpc": "2.0",
   "result": 1,
   "id": 2
}
```

4.2.3. campaigns.list

Получение списка кампаний.
Запрос:

```
{
```

```
  "jsonrpc": "2.0",
  "method": "campaigns.list",
  "id": 3
}
```

Ответ:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "id": 1,
      "name": "Летняя распродажа 2024",
      "description": "Акция на летние товары",
      "code_template": "SUMMER24-{N:6}",
      "valid_from": "2024-06-01T00:00:00Z",
      "valid_to": "2024-08-31T23:59:59Z",
      "uses_remains": 100,
      "metadata": {
        "discount": 15,
        "min_amount": 1000
      },
      "created_at": "2024-05-15T10:30:00Z",
      "updated_at": "2024-05-15T10:30:00Z",
      "deleted": false
    }
  ],
  "id": 3
}
```

4.2.4. promocodes.generate

Генерация промо-кодов для кампании.

Запрос:

```
{
  "jsonrpc": "2.0",
  "method": "promocodes.generate",
  "id": 4,
  "params": {
    "campaign_id": 1,
    "count": 100,
    "usage_limit": 1,
    "metadata": {
      "batch": "summer_2024_batch_1"
    }
  }
}
```

Ответ:

```
{
  "jsonrpc": "2.0",
  "result": null,
  "id": 4
```

```
}
```

4.2.5. promocodes.validate

Проверка валидности промо-кода.
Запрос:

```
{
   "jsonrpc": "2.0",
   "method": "promocodes.validate",
   "id": 5,
   "params": "SUMMER24-123456"
}
```

Ответ:

```
{
   "jsonrpc": "2.0",
   "result": true,
   "id": 5
}
```

4.2.6. promocodes.redeem

Активация промо-кода.
Запрос:

```
{
   "jsonrpc": "2.0",
   "method": "promocodes.redeem",
   "id": 6,
   "params": {
      "code": "SUMMER24-123456",
      "external_id": "order_789",
      "metadata": {
         "user_id": "user_123",
         "order_amount": 1500
      }
   }
}
```

Ответ:

```
{
   "jsonrpc": "2.0",
   "result": true,
   "id": 6
}
```

4.2.7. promocodes.usage_list

Получение истории использования промо-кода.
Запрос:

```
{
   "jsonrpc": "2.0",
   "method": "promocodes.usage_list",
   "id": 7,
   "params": 1
```

```
}
```

4.2.8. audit_log.list

Просмотр журнала аудита.

Запрос:

```
{
   "jsonrpc": "2.0",
   "method": "audit_log.list",
   "id": 8,
   "params": {
      "campaign_id": 1,
      "promocode_id": null
   }
}
```

4.3. Шаблоны генерации промо-кодов

Система поддерживает следующие шаблоны:

Базовые шаблоны:

- {N:5} — 5 случайных цифр (пример: 12345)
- {A:3} — 3 случайные буквы (пример: ABC)
- {HEX:8} — 8 случайных HEX символов (пример: 1A2B3C4D)
- {UUID} — генерация UUID (пример: 550e8400-e29b-41d4-a716-446655440000)

Макросы дат:

- %DATE% — текущая дата в формате YYYYMMDD (пример: 20241225)
- %YEAR% — текущий год (пример: 2024)
- %MONTH% — текущий месяц (пример: 12)
- %DAY% — текущий день (пример: 25)

Случайные символы:

- X — случайный символ (буква A-Z или цифра 0-9)

Примеры сложных шаблонов:

- PROMO-{N:6} → PROMO-123456
- {A:3}-{N:3}-{HEX:4} → ABC-123-1A2B
- DISCOUNT-%YEAR%-{N:8} → DISCOUNT-2024-12345678
- VIP-{UUID} → VIP-550e8400-e29b-41d4-a716-446655440000
-