Рекламная платформа

Руководство по установке и первоначальной настройке программного продукта

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.
Content Management System (CMS)	Система управления контентом рекламной платформы, предоставляющая возможность добавлять, удалять, редактировать отображаемые материалы, а также настраивать параметры отображения.
Docker	Программное обеспечение, предназначенное для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.
iFrame	Элемент разметки HTML, позволяющий встраивать на веб-страницу контент из сторонних источников (например, медиафайлы, документы, отдельные веб-страницы).
NGINX	Веб-сервер, который работает под управлением операционных систем семейства Linux/Unix.
PostgreSQL	Свободная объектно-реляционная система управления базами данных (СУБД).
Web-приложение	Клиент-серверное приложение, доступ к серверной части которого осуществляется посредством web-браузера.
Контент	Единица отображаемого медиаконтента (изображение, видеоролик, web-страница или ссылка на контент YouTube).
Рекламный монитор	Сгенерированная страница с набором контента, предназначенная для воспроизведения на конечных устройствах (вертикальных и горизонтальных мониторах) и доступная по уникальному url-адресу.
Эндпойнт	Шлюз, соединяющий серверные процессы приложения с внешним интерфейсом.

2. О документе

Настоящий документ содержит полное руководство по установке и первоначальной настройке программного продукта Рекламная платформа. Также в документе приведены системные требования к оборудованию, предназначенному для установки серверной части.

3. Назначение функционала

Рекламная платформа предназначена для вывода заранее подготовленного рекламного контента на устройства воспроизведения (горизонтальные или вертикальные мониторы). Отображаемый контент может относиться к одному из следующих типов:

- изображение;
- видеоролик;
- web-страница (выводится через iFrame);
- видеоконтент, размещенный на платформе YouTube.

Загрузка контента в систему и настройка параметров воспроизведения осуществляется с помощью CMS.

4. Системные требования

В настоящем разделе перечислены аппаратные и программные требования, необходимые для установки и корректной работы Рекламной платформы.

4.1. Минимальные аппаратные требования

Для обеспечения стабильного функционирования Рекламной платформы аппаратная часть должна обладать следующими характеристиками:

- Операционная система, способная запускать контейнеры. Предпочтительно ОС *Linux (Debian 11 (Открытая лицензия GNU))*;
- Система управления контейнерной виртуализацией (*Docker*);
- Подключение к серверу очередей NGINX;
- Количество логических ядер процессора: 2;
- Семейство процессоров: x86-64;
- Частота процессора: 3.0. ГГц;
- Объем установленной памяти: 2 Гб.

4.2. Минимальные требования к сторонним компонентам и/или системам

Для развертывания Рекламной платформы должны быть предварительно установлены следующие компоненты окружения:

- *Docker* 24.0.2 (open-source community edition);
- *NGINX* 1.14 (лицензия FreeBSD);
- *PostgreSQL* 9.5.7 (Open Source license).

Порядок установки сторонних компонентов приведен в п. 5.1.

Для разработки и модернизации Рекламной платформы должны быть использованы следующие программные продукты и языки программирования:

- *VSCode* 1.79.2 (Open Source license);
- *DBeaver* 23.1.1 (Open Source license);
- Веб-браузер *Chrome* 114 (Open Source license);
- Язык программирования *JavaScript* 1.8.5;
- Node.js 12.22 (ПО с открытой лицензией MIT).

5. Установка и настройка системы

Рекламная платформа включает в себя следующие компоненты (Рис. 1):

- CMS (*advertising-client*) — web-приложение, предоставляющее пользовательский интерфейс для управления рекламным контентом;
- Серверная часть (*advertising-service*) — сервер, предоставляющий эндпойнты для работы с CMS и генерирующий рекламные мониторы;
- Сервис токенов (*json_web_service*) — сервис для создания и валидации токенов;
- База данных (*pgcrypto*) — предназначена для хранения данных.

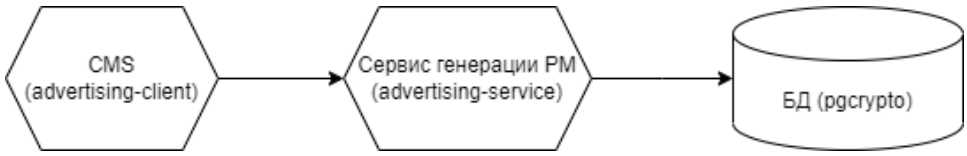

Рис. 1. Компоненты рекламной платформы

Перед запуском программного продукта необходимо выполнить следующие шаги:

1. Установка и настройка компонентов внешнего окружения (5.1);
2. Подготовка файлов конфигурации (5.2);
3. Подготовка директории для хранения загруженного контента (5.3);
4. Сборка Docker-образов. Каждая часть программного продукта собирается в отдельном Docker- контейнере (5.4);
5. Запуск сервисов *advertising-client*, *advertising-service* и *json_web_service* (5.5);
6. Создание и настройка токена (5.6).

5.1. Подготовка окружения

В настоящем разделе приведен порядок действий для предварительной подготовки каждого компонента внешнего окружения:

1. PosgreSQL (5.1.1);
2. Docker (5.1.2);
3. NGINX (5.1.3).

.

5.1.1 Подготовка PostgreSQL

PostgreSQL может быть установлена как из репозитория *Debian 11*, так и из официального репозитория СУБД.

Установка из репозитория Debian 11

1. Сначала обновите список пакетов. Запустите терминал и выполните команду:

   ```
   sudo apt update
   ```

2. Пакет PostgreSQL находится в репозитории *Debian*, поэтому вы можете установить его с помощью утилиты *apt*. Для этого выполните:

   ```
   sudo apt-get install postgresql=11*
   ```

3. После завершения установки проверьте статус соответствующей службы с помощью команды:

   ```
   sudo systemctl status postgresql
   ```

4. Если служба не запустилась автоматически, вы можете запустить её вручную. Для этого выполните:

   ```
   sudo systemctl start postgresql
   ```

Установка из официального репозитория СУБД

1. Прежде всего необходимо добавить ключ подписи GPG:

```
curl -fsSL https://www.postgresql.org/media/keys/ACCC4CF8.asc |
sudo gpg --dearmor -o /usr/share/keyrings/postgresql-keyring.gpg
```

2. Теперь вы готовы добавить репозиторий *Postgres*:

```
echo "deb [signed-by=/usr/share/keyrings/postgresql-keyring.gpg]
http://apt.postgresql.org/pub/repos/apt/  "$(. /etc/os-release &&
echo "$VERSION_CODENAME")"-pgdg main" |
sudo tee /etc/apt/sources.list.d/postgresql.list
```

3. Обновите репозиторий системы с помощью команды:

```
sudo apt update
```

4. После обновления выполните PostgreSQL install на *Debian*:

```
sudo apt install postgresql
```

5. После установки PostgreSQL необходимо подготовить базы данных и пользователей:

```
sudo -u postgres psql
CREATE DATABASE token_service;
CREATE DATABASE advertising_service;
CREATE USER token_service_user WITH PASSWORD 'password';
CREATE USER advertising_service_user WITH PASSWORD 'password';
ALTER DATABASE token_service OWNER TO token_service_user;
ALTER DATABASE advertising_service OWNER TO advertising_service_user;
GRANT ALL PRIVILEGES ON DATABASE token_service TO token_service_user;
GRANT ALL PRIVILEGES ON DATABASE advertising_service_user TO
advertising_service_user_user;
```

6. Для возможности подключения сервисов вставьте в конец файла */etc/postgresql/11/main/pg_hba.conf* следующую строку:

```
host all all 0.0.0.0/0 md5
```

5.1.2 Подготовка Docker

1. Установите зависимости с помощью команды:

```
sudo apt-get install ca-certificates curl gnupg
```

2. Добавьте ключ подписи GPG, выполнив команды:

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg |
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

3. Добавьте репозиторий, выполнив команды:

```
echo \
"deb [arch="$(dpkg --print-architecture)"
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

4. Теперь установите *Docker*:

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin
```

5.1.3 Подготовка NGINX

1. Установите NGINX, выполнив команду:

```
apt-get instsall nginx
```

2. Замените содержимое файла */etc/nginx/sites-enabled/default* на следующую конфигурацию:

```
upstream backend {
      server 10.**.**.**:5658;
}

server {
        error_page 500 502 503 504 /500.html;

        location = /500.html {
                root /usr/share/nginx/html;
                internal;
        }
```

```
        listen                  80;
        server_name             default_server;

        access_log      /var/log/nginx/cms/host_access.log;
        error_log       /var/log/nginx/cms/host_error.log;
        ssl_session_cache       shared:SSL:10m;
        ssl_session_timeout     10m;
        fastcgi_param HTTPS on;
        client_max_body_size 150m;
        proxy_connect_timeout 180s;
       proxy_read_timeout 5m;



        location / {
             return 444;

         access_log off;
         error_log off;

        }

        location ~ ^/cms {
                proxy_pass          http://10.**.**.**:5659;

                proxy_set_header    Host $host;
                proxy_set_header    X-Real-IP  $remote_addr;
                proxy_set_header    X-Forwarded-Host
$host:$server_port;
                proxy_set_header    X-Forwarded-Server      $host;
                proxy_set_header    X-Forwarded-For
$proxy_add_x_forwarded_for;
         }

         location ~ ^/advertising {
                proxy_pass          http://10.**.**.**:5658;
                proxy_set_header    Host $host;
                proxy_set_header    X-Real-IP  $remote_addr;
```

```
                proxy_set_header    X-Forwarded-Host
$host:$server_port;
                proxy_set_header    X-Forwarded-Server      $host;
                proxy_set_header    X-Forwarded-For
$proxy_add_x_forwarded_for;
                proxy_read_timeout  300;
        }

        location ~ ^/advertising/api/v1 {
                proxy_pass          http://10.**.**.**:5658;
                proxy_set_header    Host $host;
                proxy_set_header    X-Real-IP  $remote_addr;
                proxy_set_header    X-Forwarded-Host
$host:$server_port;
                proxy_set_header    X-Forwarded-Server      $host;
                proxy_set_header    X-Forwarded-For
$proxy_add_x_forwarded_for;
                proxy_read_timeout 300;
    }
}
```

Внимание!

Ваша сеть *Docker* может отличаться, поэтому дополнительно необходимо выполнить следующие действия:

3. Выполните команду:

```
ip a| grep 'docker' | grep inet | awk '{print $2}' | cut -f1 -d"/"
```

4. В файле конфигурации замените «*10.**.**.***» значением, полученным в результате выполнения команды.

5.2. Подготовка файлов конфигурации

Перед первым запуском системы необходимо установить конфигурационные параметры сервисов *advertising-service* и *json_web_service*.

Конфигурация *advertising-service*

Создайте файл конфигурации для сервиса *advertising-service* в любом месте (например, */etc/cms/advertizing.env*) и скопируйте в него перечень параметров:

```
PORT=5658
PER_PAGE=25
TOKEN_SERVICE__HOST=http://10.**.**.**:5660
TOKEN_SERVICE__ACCESS=
ADS_SLIDER__REFRESH_TIME=600000
ADS_SLIDER__CONTENT_LIFE_TIME=86400000
ADS_SLIDER__CDN=
CONFIG__PUBLIC_DIR=/config
CONFIG__PRIVATE_DIR=/config
LOG__TYPE=console
LOG__FILE_PATH=/var/log/bblog/advertising
LOG__FILE_LEVEL=info
LOG__CONSOLE_LEVEL=info
LOG__STRING_OUTPUT=true
DB__PING_TIMEOUT=5000
DB__CONNECTIONS__PG_POOL__CONNECTION_TYPE=pg
DB__CONNECTIONS__PG_POOL__HOST=10.**.**.**
DB__CONNECTIONS__PG_POOL__PORT=5432
DB__CONNECTIONS__PG_POOL__DATABASE=advertising_service
DB__CONNECTIONS__PG_POOL__CURRENT_SCHEMA=public
DB__CONNECTIONS__PG_POOL__USER=advertising_service_user
DB__CONNECTIONS__PG_POOL__PASSWORD=
DB__CONNECTIONS__PG_POOL__MAX=5
```

В качестве значения параметра

```
DB__CONNECTIONS__PG_POOL__PASSWORD
```

необходимо указать пароль для пользователя *advertising_service_user*.

Конфигурация *json_web_service*

Создайте файл конфигурации для сервиса *json_web_service* в любом месте (например, */etc/cms/json.env*) и скопируйте в него перечень параметров:

```
PORT=80
CONFIG__PUBLIC_DIR=/config
CONFIG__PRIVATE_DIR=/config
```

```
LOG__TYPE=console
LOG__FILE_PATH=/var/log/token_sevice
LOG__FILE_LEVEL=info
LOG__STRING_OUTPUT=true
DB__CONNECTIONS__PG_POOL__CONNECTION_TYPE=pg
DB__CONNECTIONS__PG_POOL__HOST=10.**.**.**
DB__CONNECTIONS__PG_POOL__USER=token_service_user
DB__CONNECTIONS__PG_POOL__PASSWORD=xiakageiy0eesh7engaiquao9uaW5T
DB__CONNECTIONS__PG_POOL__PORT=5432
DB__CONNECTIONS__PG_POOL__MAX=10
DB__CONNECTIONS__PG_POOL__CURRENT_SCHEMA=public
DB__CONNECTIONS__PG_POOL__DATABASE=token_service
```

В качестве значения параметра

```
DB__CONNECTIONS__PG_POOL__PASSWORD
```

необходимо указать пароль для пользователя *token_service_user*.

5.3. Подготовка директорий

Необходимо создать директорию для хранения контента:

```
mkdir -p /var/www/advertising
```

5.4. Сборка Docker-образов

Сценарии сборки для Docker-образа каждого сервиса прописаны в соответствующем *Dockerfile*.

***Dockerfile* для *advertising-client* (CMS)**

```
FROM node:12-alpine AS builder
ENV NODE_ENV=production

WORKDIR /app
COPY package.json package-lock.json /app/
RUN npm install && npm install --only=dev
COPY . /app/
```

```
RUN npm run build

FROM nginx:1.17-alpine
ENV NODE_ENV=production
WORKDIR /app

COPY nginx/default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/dist /usr/share/nginx/html/cms
```

Dockerfile для *advertising-service* (серверная часть)

```
FROM node:12-alpine

COPY . /app/
WORKDIR /app

ENV NODE_ENV=production
RUN npm install

CMD ["index.js"]
```

Dockerfile для *json-web-service* (сервис создания и валидации токенов)

```
FROM node:12-alpine

COPY . /app/
WORKDIR /app

ENV NODE_ENV=production
RUN npm install

EXPOSE 80
CMD ["index.js"]
```

Каждый Docker-образ необходимо собрать командой:

```
docker build -t <Имя образа>
```

Например, для приложения *advertising-service* команда выглядит так:

```
docker build -t advertising-service
```

Внимание!

Dockerfile с содержимым для соответствующего сервиса должен находится в той же директории, что и вы.

5.5. Запуск сервисов

После того, как все Docker-образы собраны, можно запускать сервисы.

Каждый сервис запускается отдельной командой.

Пример команды запуска для *advertising_service*:

```
sudo docker run -d --name advertising_service --restart=always -v
/etc/localtime:/etc/localtime:ro -p 0.0.0.0:5658:5658 -v
/var/www/advertising/:/app/public/ --env-file=/etc/cms/advertizing.env
advertising_service
```

--name advertising_service – задаем имя будущего контейнера;

-p 0.0.0.0:5658:5658 – пробрасываем порты из контейнера в хостовую систему;

-v /var/www/advertising/:/app/public/ – монтируем директорию для хранения контента;

--env-file=/etc/cms/advertizing.env – указываем, где находится файл с конфигурационными параметрами;
advertising_service – последней директивой указываем имя образа.

5.6. Создание и настройка токена

После запуска *json_web_service* необходимо создать токен.

Отправьте запрос:

```
curl -X POST -H "Content-Type: application/json" -d '{"keyName":
"advertising"}' "http://10.**.**.**:6556/create"
```

Получите ответ от сервиса:

```
{
    "success":true,
    "message":"1 key created with name: advertising",
    "data":{

"access":"eyJraWQiOiJDUGxSSUJNRTlleXZMTmw4ZW5YdG1zcE4zTkgwU1NSeWhKblh
mb3U4akk4IiwiYWxnIjoiRVMyNTYifQ.eyJfX0tFWV9OQU1FIjoiYWR2ZXJ0aXNpbmciL
CJfX0NSRUFURSI6dHJ1ZSwiaWF0IjoxNjUyMjQ1MzM5fQ.i0qIILznfcllNaN4fqrX0wS
-lqKXq-TgqotG3N3fOICmcyPjNwG8pNfqUwWBkjgPYQG-G_AEKpop4u01W6IIlw",
"decrypt":"eyJraWQiOiJDUGxSSUJNRTlleXZMTmw4ZW5YdG1zcE4zTkgwU1NSeWhKbl
hmb3U4akk4IiwiYWxnIjoiRVMyNTYifQ.eyJfX0tFWV9OQU1FIjoiYWR2ZXJ0aXNpbmci
LCJpYXQiOjE2NTIyNDUzMzl9.V8TW24TGgC_0MHPPcXll5FVR60s-iU6-
2aneUkBAv7LzxnFFkSVHYmLYYoHB2sBTdKlyFiELxj9O4XQhTWIicQ"

            }
}
```

Скопируйте содержмое директивы *access* и вставьте его в следующий запрос:

```
curl -X POST -H "Content-Type: application/json" -d '{
"access":"eyJraWQiOiJDUGxSSUJNRTlleXZMTmw4ZW5YdG1zcE4zTkgwU1NSeWhKblh
mb3U4akk4IiwiYWxnIjoiRVMyNTYifQ.eyJfX0tFWV9OQU1FIjoiYWR2ZXJ0aXNpbmciL
CJfX0NSRUFURSI6dHJ1ZSwiaWF0IjoxNjUyMjQ1MzM5fQ.i0qIILznfcllNaN4fqrX0wS
-lqKXq-TgqotG3N3fOICmcyPjNwG8pNfqUwWBkjgPYQG-G_AEKpop4u01W6IIlw",
"payload": {"advertising":true}}' "http://10.**.**.**:6556/encrypt"
```

Получите ответ от сервиса:

```
{
    "success":true,
...."data":{

"jwe":"eyJlbmMiOiJBMTI4Q0JDLUhTMjU2IiwiYWxnIjoiUEJFUzItSFMyNTYrQTEyOE
tXIiwicDJjIjozNTk5LCJwMnMiOiJVVWQ4eV9EamU4RDZLMEpqWlpPNEh3In0.xnzk_3Y
imXJE_VUoGNUaLwOmSPfTJESBSYMSOk4aSlgR9qWRKV49ug.fvt1Q9ycjsRVUbQlWXvYo
g.G2bFRqvxLNo1BKDsYr9Wo2i2tVyibsfoG-
iCneKlVRQfb3rmQnevtemkQGh3LJqGu_bZpp0lTyixitYPClYOFSZXrKl27uvYFmSJwsm
dGVcEeqeSIeLcIUeir0uLXL-vmUJCGAVF9-
m4bm0lOudjCMUwDXNzrUDD8IqRTDu7BXtx-
E2dkqtehljxK7OYjZXHyEo8ErYqfjm1Pr2dca2Em33GjfkqwmW4V8I-
```

```
PvSIJ12H1NcwB9XdZN4Cei93yQMNBvoacEIN18HPLreQEKyYzODBBixaR-GGFOvSf-
ArfAdFaBItdfQOrD7JGDn4LiRi.yHg7_O1OI2KAX3f95_ia8g"
            }
}
```

Добавьте содержимое директивы *jwe* в файл конфигурации *advertising_service* в качестве значения параметра TOKEN_SERVICE_ACCESS:

```
TOKEN_SERVICE__ACCESS=eyJlbmMiOiJBMTI4Q0JDLUhTMjU2IiwiYWxnIjoiUEJFUzI
tSFMyNTYrQTEyOEtXIiwicDJjIjozNTk5LCJwMnMiOiJVVWQ4eV9EamU4RDZLMEpqWlpP
NEh3In0.xnzk_3YimXJE_VUoGNUaLwOmSPfTJESBSYMSOk4aSlgR9qWRKV49ug.fvt1Q9
ycjsRVUbQlWXvYog.G2bFRqvxLNo1BKDsYr9Wo2i2tVyibsfoG-
iCneKlVRQfb3rmQnevtemkQGh3LJqGu_bZpp0lTyixitYPClYOFSZXrKl27uvYFmSJwsm
dGVcEeqeSIeLcIUeir0uLXL-vmUJCGAVF9-
m4bm0lOudjCMUwDXNzrUDD8IqRTDu7BXtx-
E2dkqtehljxK7OYjZXHyEo8ErYqfjm1Pr2dca2Em33GjfkqwmW4V8I-
PvSIJ12H1NcwB9XdZN4Cei93yQMNBvoacEIN18HPLreQEKyYzODBBixaR-GGFOvSf-
ArfAdFaBItdfQOrD7JGDn4LiRi.yHg7_O1OI2KAX3f95_ia8g
```

5.7. Начало работы

После выполнения предыдущего шага перезапустите *advertising_service*.

Теперь можно работать с CMS через браузер по адресу *<IP>/cms*.

Логин и пароль для входа в систему:

- Логин: *admin*
- Пароль: *Opheojei4aebah0ohcei9OhNie6eel*