ИНСТРУКЦИЯ ПО РАЗВЁРТЫВАНИЮ
программного обеспечения «ИС.АРЕНА 2.0»

1. Общие сведения

Настоящая инструкция описывает порядок развёртывания программного обеспечения «ИС.АРЕНА 2.0» (далее — Система) из поставляемого комплекта. Система поставляется в виде набора Docker-образов и сопровождающих файлов (файлы docker-compose, файлы окружения .env, конфигурационные файлы, дамп базы данных, файлы конфигурации Realm, наборы маршрутов шлюза), что обеспечивает воспроизводимое развёртывание в контейнерной среде.

Развёртывание выполняется по слоям в порядке их зависимости:

– инфраструктурный слой — контейнеры инфраструктурных сервисов (Postgres, Elasticsearch, MinIO, Kafka, Kafka Connect, Redis, Keycloak, Superset);

– слой продуктовых микросервисов — контейнеры прикладных сервисов Системы, шлюзы APISIX, веб-серверы Nginx, сборки веб-приложений;

– слой интеграции — контейнеры взаимодействия со СМЭВ (backend-сервис и smev-adapter).

Порядок обязателен: продуктовые микросервисы при старте подключаются к инфраструктурным сервисам, поэтому инфраструктурный слой должен быть развёрнут и работоспособен до запуска микросервисов. Все параметры подключения (адреса, порты, учётные данные, ключи) задаются в файлах окружения .env по соответствующим блокам.

2. Подготовка окружения

На сервере должны быть установлены Docker и Docker Compose версии 2 и выше. Рекомендуемая ОС — Linux-дистрибутив семейства Ubuntu, Debian, Astra Linux или совместимый. Минимальные рекомендуемые ресурсы: не менее 8 ядер CPU, 32 ГБ оперативной памяти, около 100 ГБ свободного дискового пространства.

До начала развёртывания необходимо убедиться, что используемые сетевые порты свободны. Перечень портов, используемых сервисами Системы:

```
587, 5432, 5540, 5555, 6000, 6001, 6379, 8001, 8077, 8080, 8081,
8082, 8083, 8084, 8085, 8086, 8087, 8088, 8089, 8090, 8091, 8092,
8095, 8099, 8100, 9000, 9005, 9080, 9085, 9091, 9092, 9093, 9094,
9200, 9300, 9443, 9999, 2378, 2379, 7575, 443
```

После извлечения исходников появится несколько папок. В кажой папке по инструкции разворачивается и настраивается отдельный слой инфраструктуры. Подробнее можно ознакомится в файле README.txt. Для начала работ необходимо перейти в software

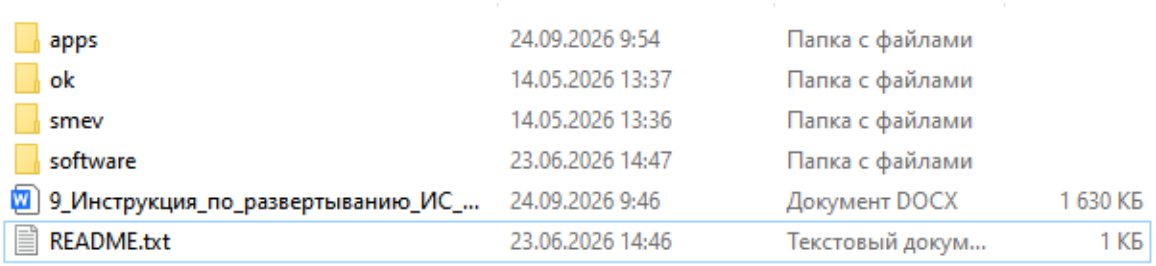


Рисунок 1.1 - содержание архива

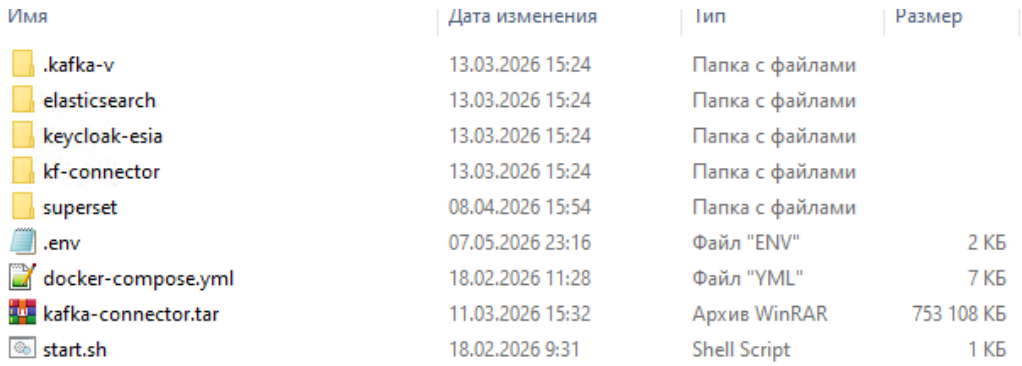


Рисунок 1.2 — блоки настроек в файле .env

3. Развёртывание инфраструктурного слоя

Инфраструктурный слой поставляется в виде готовых контейнеров и разворачивается через docker-compose. Установка инфраструктурного ПО «с нуля» не требуется — используются поставляемые образы с преднастроенной конфигурацией. Запуск утилитных сервисов (Postgres, Elasticsearch, MinIO, Kafka в режиме KRaft, Kafka Connect, Redis, Keycloak) выполняется командой:

```
docker compose up -d
```

После запуска убедиться, что все контейнеры находятся в состоянии running и доступны по своим портам. Для начала работ необходимо перейти в папку keycloak-esia и выполнить команду

```
docker load -i keycloak_1.0.0.tar.gz
```

Также необходимо проверить доступность до официально Docker Hub для скачивания образов. Проверить можно следующей командой

```
curl -I https://hub.docker.com/v2/repositories/library/nginx/
```

После выполнения команды в ответ на запрос должен прийти статус 200, что означает что есть сетевой доступ до официального Docker Hub.

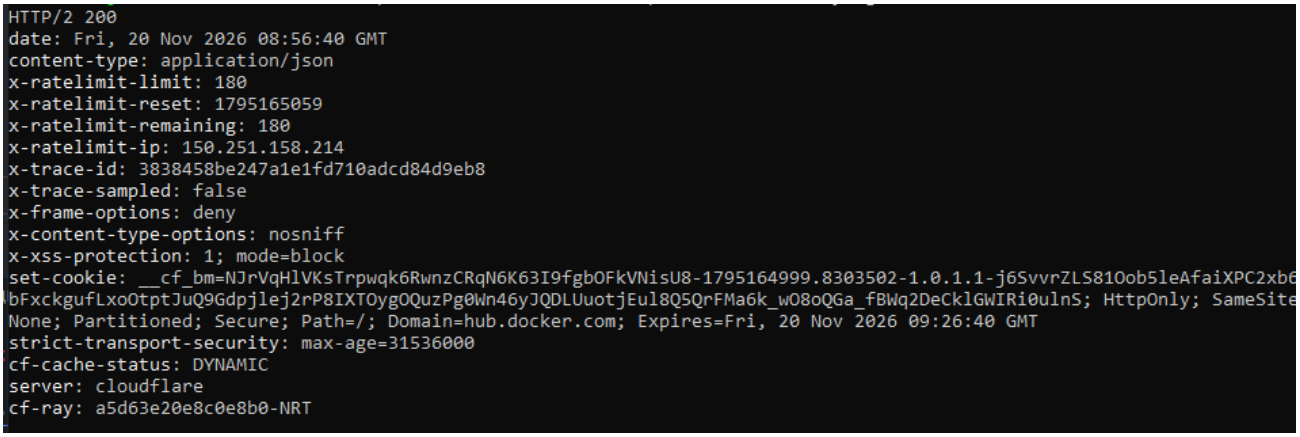


Рисунок 1.3 — получения ответа на запрос

3.1. Восстановление базы данных

В комплект поставки входит архив дампа, содержащий структуру базы данных и сопутствующие объекты (функции, триггеры, публикации и подписчики механизма репликации). Восстановление выполняется клиентом psql:

```
psql -U postgres -f /backup/full_cluster_backup.sql
```

После восстановления убедиться, что созданы таблицы, функции и триггеры, а также настроены публикации и подписчики.

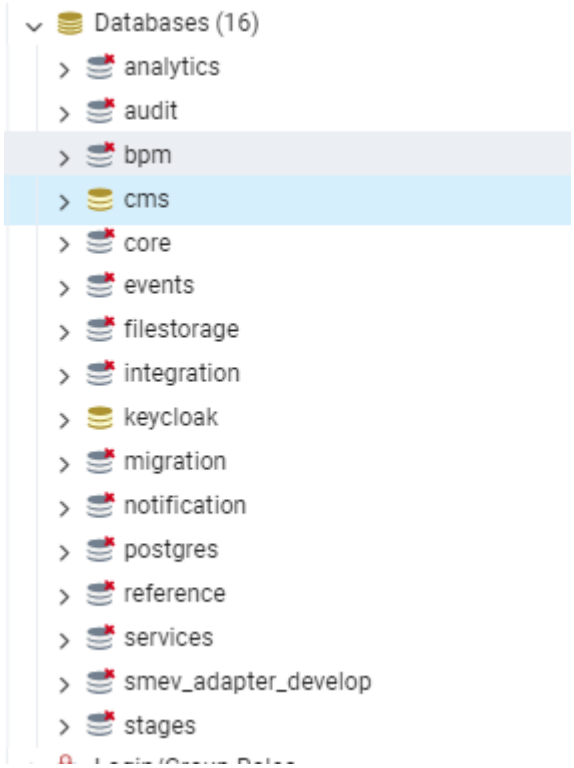


Рисунок 2 — проверка структуры базы данных: таблицы

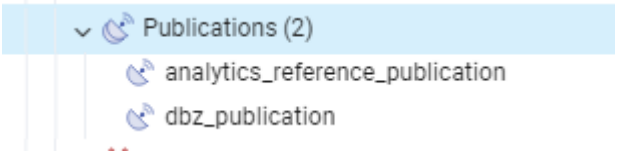


Рисунок 3 — проверка структуры базы данных: подписчики и слушатели

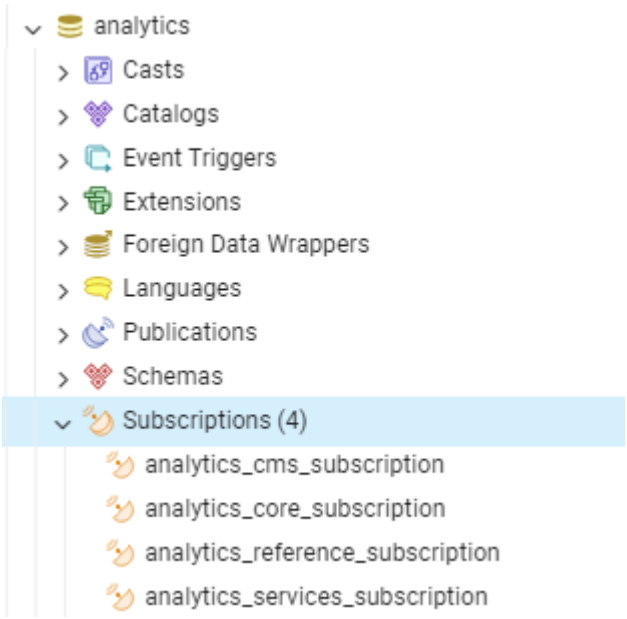


Рисунок 4 — проверка структуры базы данных: функции

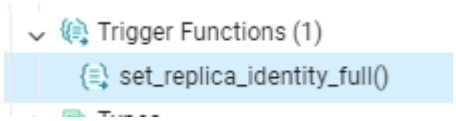


Рисунок 5 — проверка структуры базы данных: триггеры (1)

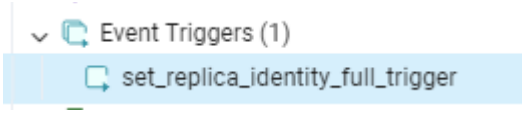


Рисунок 6 — проверка структуры базы данных: триггеры (2)

3.2. Настройка объектного хранилища MinIO

Открыть веб-интерфейс MinIO на порту 6001, создать бакет для приложения и сгенерировать пару ключей доступа — Access Key и Secret Key. Сохранить эти значения: они указываются в файле .env backend-приложения.

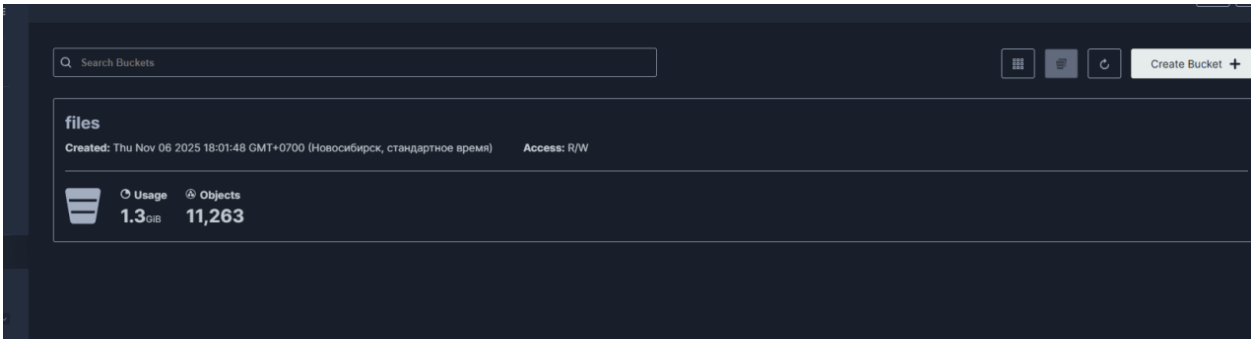


Рисунок 7 — создание бакета в MinIO

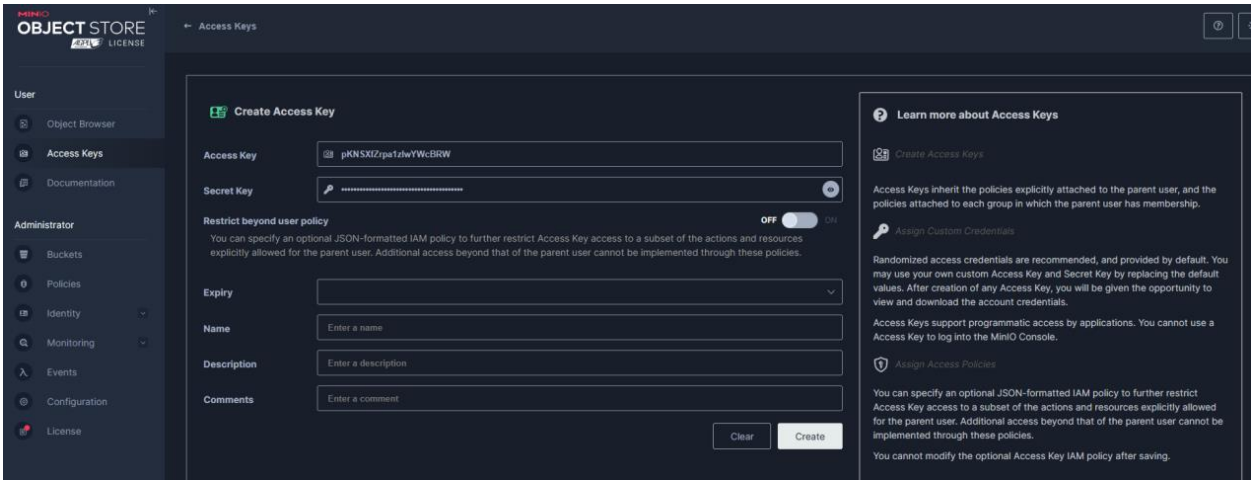


Рисунок 8 — создание ключей доступа Access Key и Secret Key

3.3. Регистрация коннекторов Kafka Connect

Kafka Connect поставляется в виде кастомного образа с коннекторами для интеграции PostgreSQL и Elasticsearch. После запуска сервиса зарегистрировать

коннекторы через REST API, последовательно передавая в теле запроса все .json из папки kf-connector:

```
POST http://<ip_BM_kafka>:8083/connectors/
```

Убедиться, что коннекторы созданы и находятся в рабочем состоянии — они обеспечивают поток данных между СУБД и поисковым индексом.

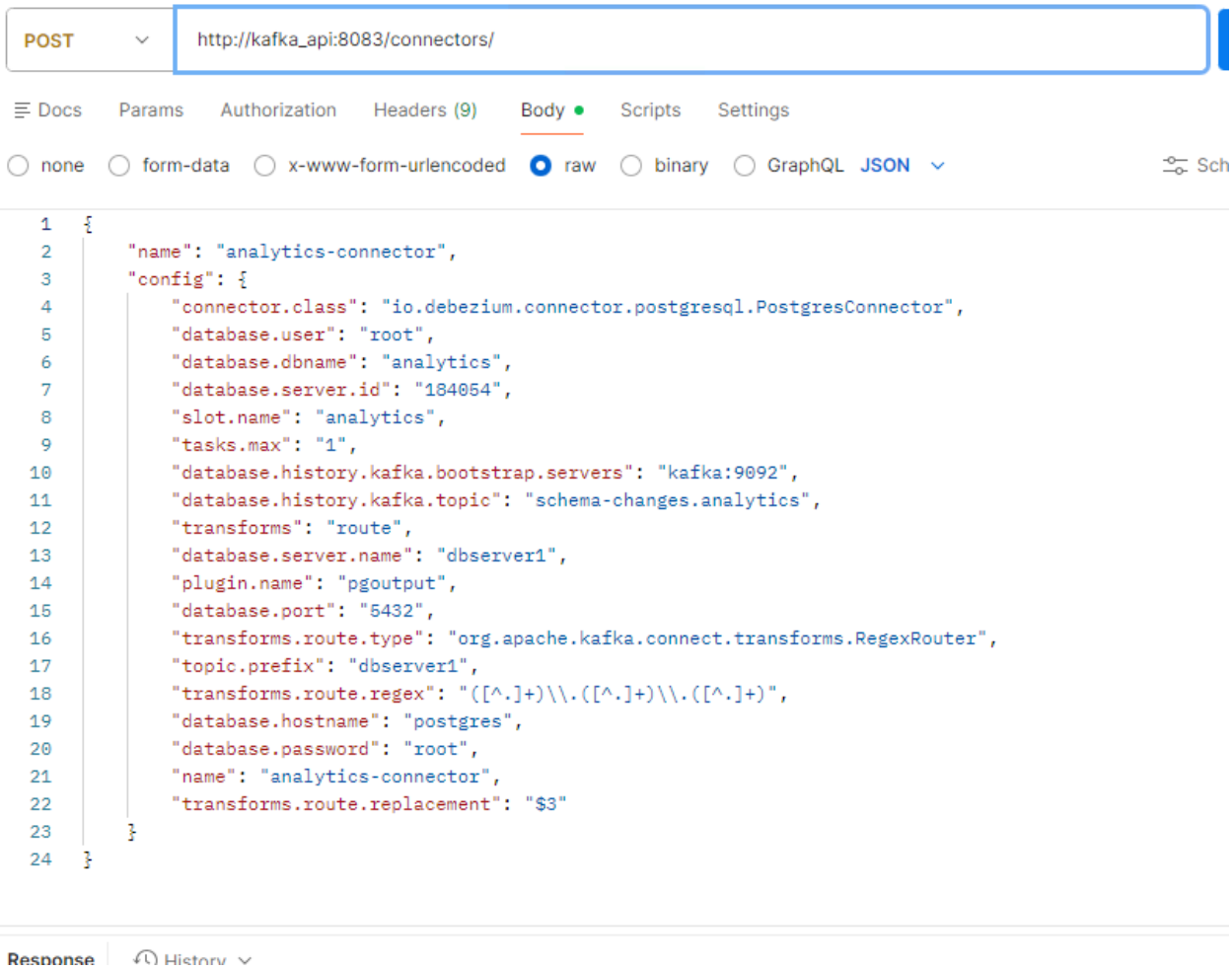


Рисунок 9 — регистрация коннекторов Kafka Connect

3.4. Настройка Keycloak

Keycloak поставляется в виде образа и файлов Realm в формате JSON (настройки клиентов, ролей и scope). Вход в административный интерфейс по умолчанию — порт 9091, логин и пароль admin / admin (изменяется в docker-compose). Порядок настройки:

1) создать два реалма — test (закрытая часть) и public (публичная часть) — и импортировать соответствующие файлы Realm (Import);

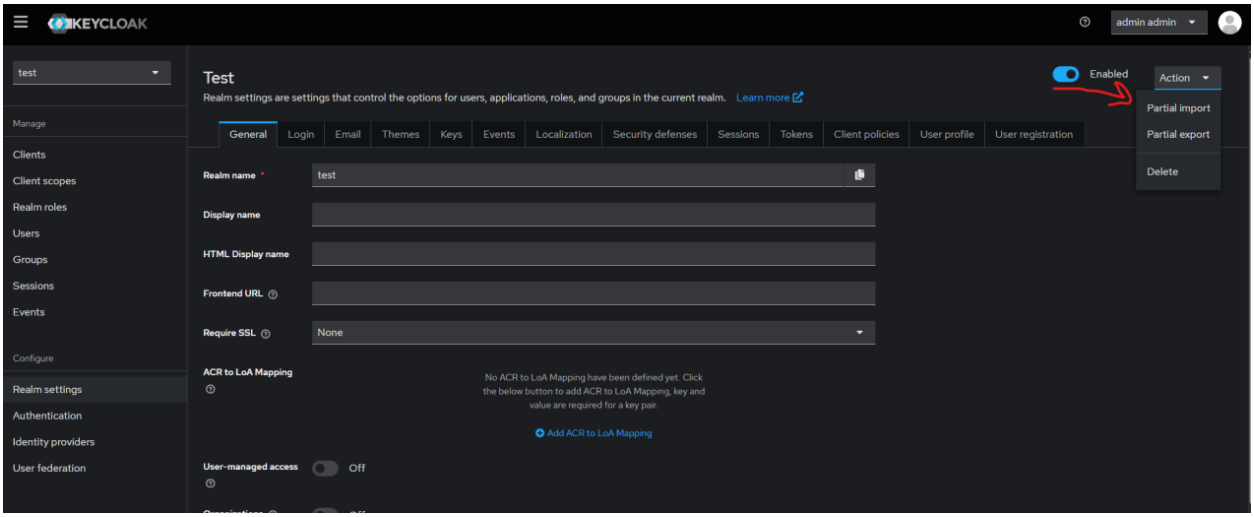


Рисунок 10 — импорт реалма

2) для поддержки межсервисных (m2m) запросов повторно сгенерировать секретный ключ в клиенте m2m и сохранить его — значение указывается в .env продуктовых микросервисов;

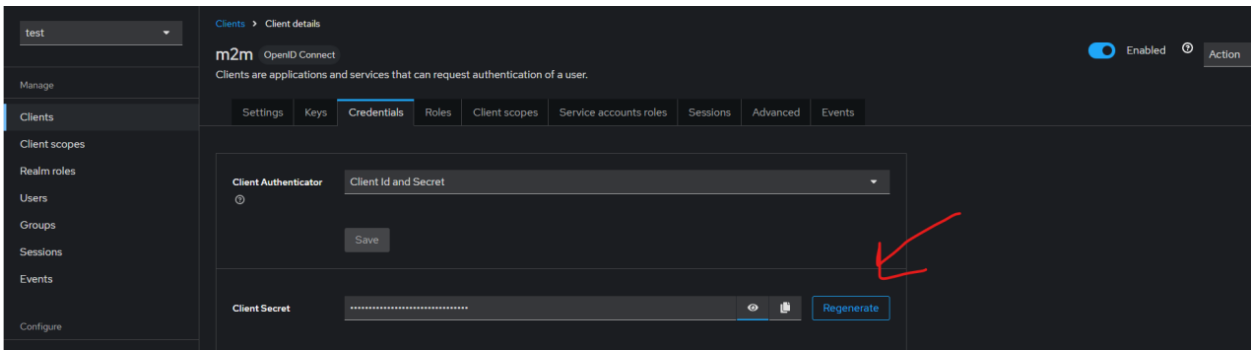


Рисунок 11 — генерация секретного ключа клиента m2m

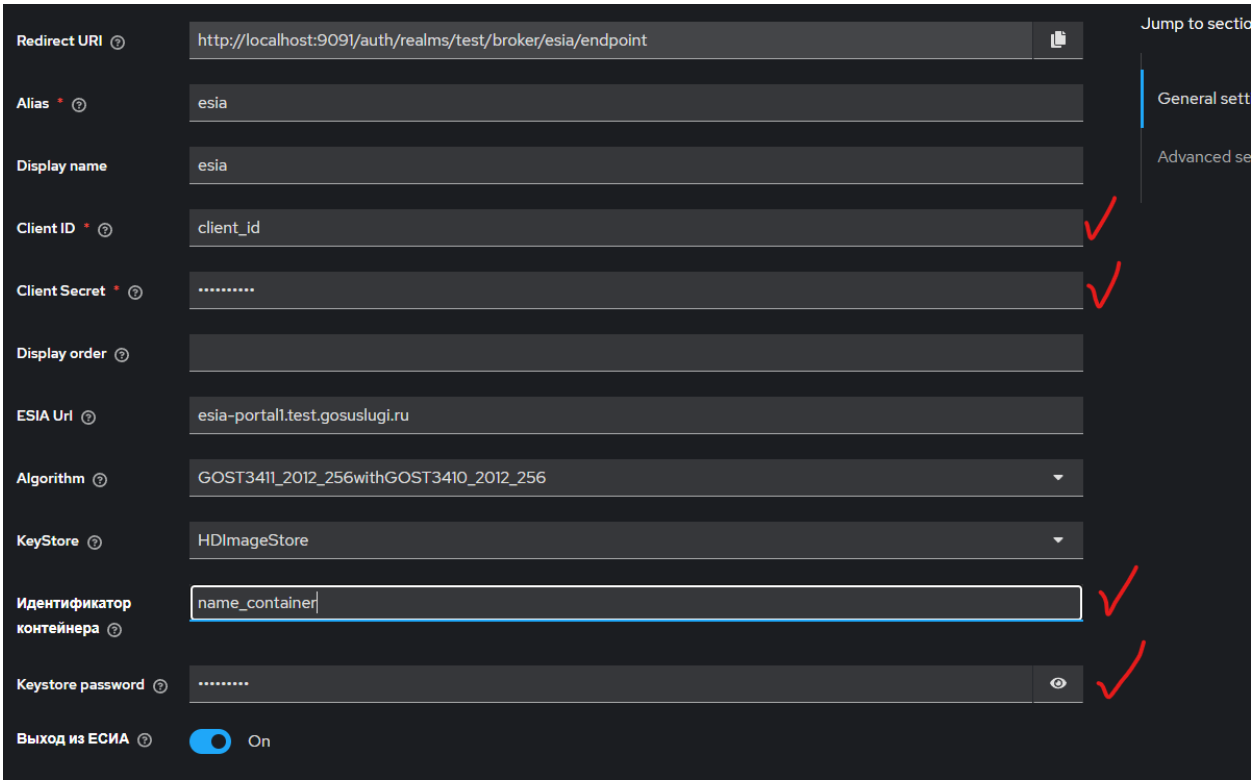


Рисунок 12 — сохранение секретного ключа

После импорта система авторизации готова к работе; backend использует её для аутентификации и авторизации пользователей.

3.5. Настройка взаимодействия с ЕСИА

Для авторизации через ЕСИА используется отдельный контейнер. Ключевой контейнер средства электронной подписи размещается на сервере в каталоге:

```
/var/opt/cprocsp/keys/<USER>/
```

Вместо <USER> подставляется имя пользователя на сервере, которое указывается в .env утилитных контейнеров. В личном кабинете участника взаимодействия (ЛК УВ) регистрируются домены закрытой и публичной частей для корректного перенаправления авторизации. Дополнительно в клиенте vue-client указывается домен закрытой части, на котором будет развёрнут фронтенд.

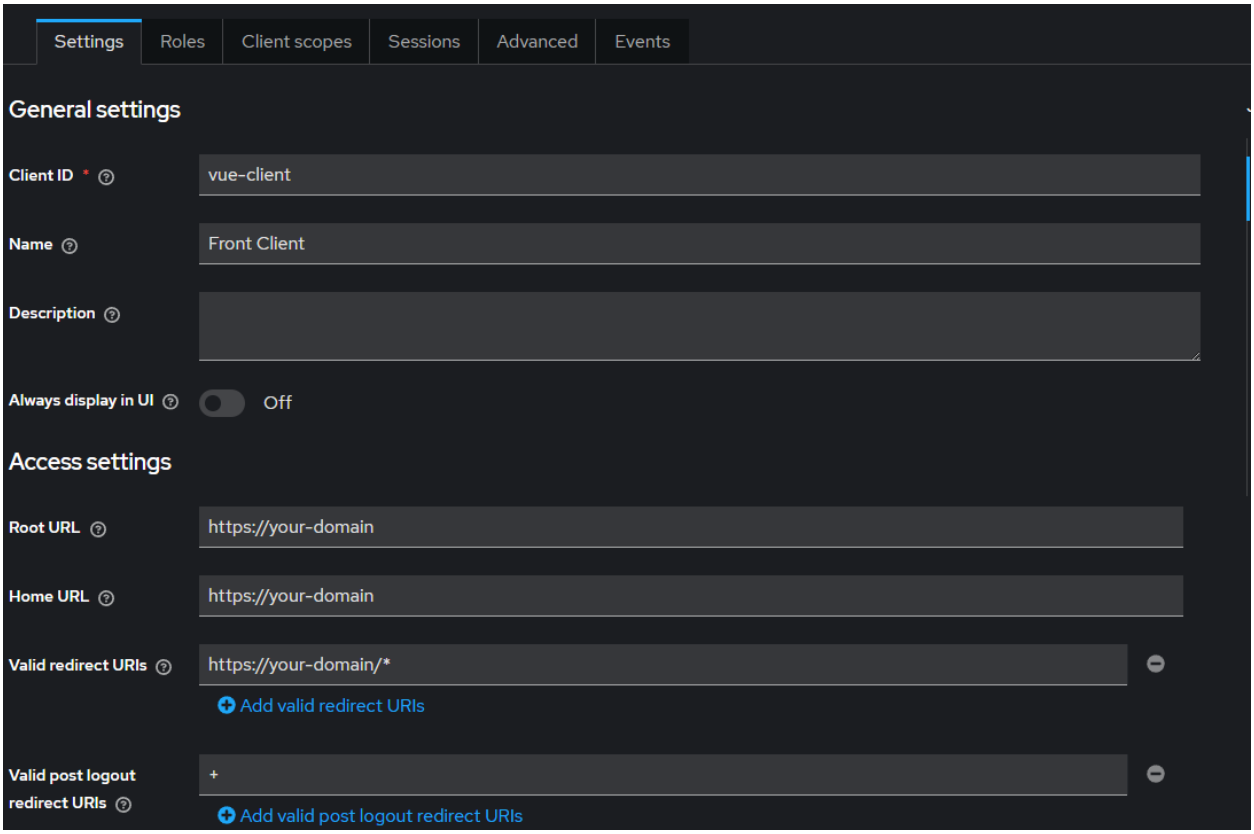


Рисунок 13 — размещение ключевого контейнера ЭП

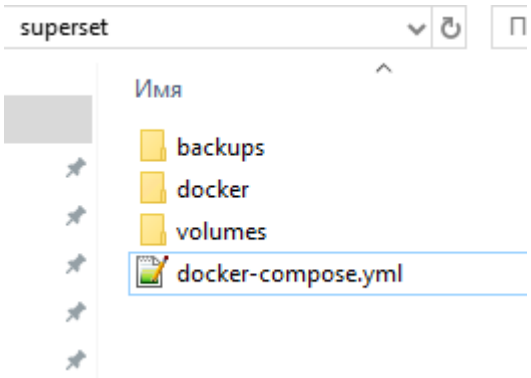


Рисунок 14 — указание пользователя ВМ в .env

3.6. Настройка Superset

Superset разворачивается через docker-compose. В файле superset_config.py по пути superset/docker/pythonpath_docker прописываются параметры подключения к базе данных и Redis, а также домены, по которым будут доступны Superset и закрытая часть. Запуск — командой docker compose up -d.

```
DATABASE_DIALECT = os.getenv("DATABASE_DIALECT", "postgresql")
DATABASE_USER = os.getenv("DATABASE_USER", "superset")
DATABASE_PASSWORD = os.getenv("DATABASE_PASSWORD", "superset")
DATABASE_HOST = os.getenv("DATABASE_HOST", "localhost")
DATABASE_PORT = os.getenv("DATABASE_PORT", "5432")
DATABASE_DB = os.getenv("DATABASE_DB", "superset")
```

Рисунок 15 — настройки подключения к БД в superset_config.py

```
REDIS_HOST = os.getenv("REDIS_HOST", "redis")
REDIS_PORT = os.getenv("REDIS_PORT", "6379")
REDIS_CELERY_DB = os.getenv("REDIS_CELERY_DB", "0")
REDIS_RESULTS_DB = os.getenv("REDIS_RESULTS_DB", "1")
```

Рисунок 16 — настройки подключения к Redis

```
# ---------- URL & DOMAINS ----------
SUPERSET_WEBSERVER_BASEURL = "https://analytics.your-domain"
SUPERSET_WEBSERVER_DOMAINS = ["analytics.your-domain"]

# ---------- CORS ----------
ENABLE_CORS = True

# ---------- TALISMAN (Content Security Policy) ----------
TALISMAN_ENABLED = True
TALISMAN_CONFIG = {
    "content_security_policy": {
        "frame-ancestors": ["'self'", "https://your-domain/"],
        "default-src": ["'self'"],
        "script-src": ["'self'", "'unsafe-inline'", "'unsafe-eval'"],
        "style-src": ["'self'", "'unsafe-inline'"],
        "img-src": ["'self'", "blob:", "data:", "https://gravatar.com", "https://secure.gravatar.com"],
        "connect-src": ["'self'", "https://api.mapbox.com", "https://events.mapbox.com"],
        "object-src": "'none'",
    },
    "content_security_policy_nonce_in": ["script-src"],
    "force_https": False,
    "session_cookie_secure": True,
}
```

Рисунок 17 — указание доменов Superset и закрытой части

Файл Правка Формат Вид Справка

```
# Base paths
SYSTEM_HOME_PATH=/opt/new-platform
POSTGRES_DATA=${SYSTEM_HOME_PATH}/volumes/pgdata

# Postgres
POSTGRES_CONTAINER=postgres
POSTGRES_IMAGE=postgres:15
POSTGRES_USER=root
POSTGRES_PASSWORD=root
POSTGRES_DB=arena
POSTGRES_MAX_CONNECTIONS=1000
POSTGRES_MAX_WAL_SENDERS=20
POSTGRES_MAX_REPLICATION_SLOTS=20

# Elasticsearch
ES_CONTAINER=elasticsearch
ES_IMAGE=elasticsearch:9.0.1
ES_CLUSTER_NAME=arena-cluster
ES_NODE_NAME=elasticsearch-node-1
ES_JAVA_OPTS=-Xms512m -Xmx512m

# MinIO
MINIO_CONTAINER=minio
MINIO_IMAGE=minio/minio:RELEASE.2025-01-20T14-49-07Z
MINIO_ROOT_USER=admin
MINIO_ROOT_PASSWORD=12345678

# Kafka
KAFKA_CONTAINER=kafka
KAFKA_IMAGE=confluentinc/cp-kafka:7.5.1
KAFKA_NODE_ID=1
KAFKA_CLUSTER_ID=AVV9IJJjT7aNKgUKUSwUSg

# IP/host VM, куда клиенты снаружи будут подключаться
KAFKA_EXTERNAL_HOST=<ip>
KAFKA_INTERNAL_HOST=kafka

# Kafka Connect
KAFKA_CONNECT_CONTAINER=kafka-connector
KAFKA_CONNECT_IMAGE=confluentinc/cp-kafka-connect:latest
KAFKA_CONNECT_GROUP_ID=kafka-connect-group

# Redis
REDIS_CONTAINER=redis_cache
REDIS_IMAGE=redis:latest

# RedisInsight
REDISINSIGHT_CONTAINER=redis_ui
REDISINSIGHT_IMAGE=redislabs/redisinsight:latest


#Keycloak
KEYCLOAK_ESIA_CERT_CONTAINER=/var/opt/cprocsp/keys/<USER>/
```

Рисунок 18 — пример файла .env инфраструктурного слоя (1)

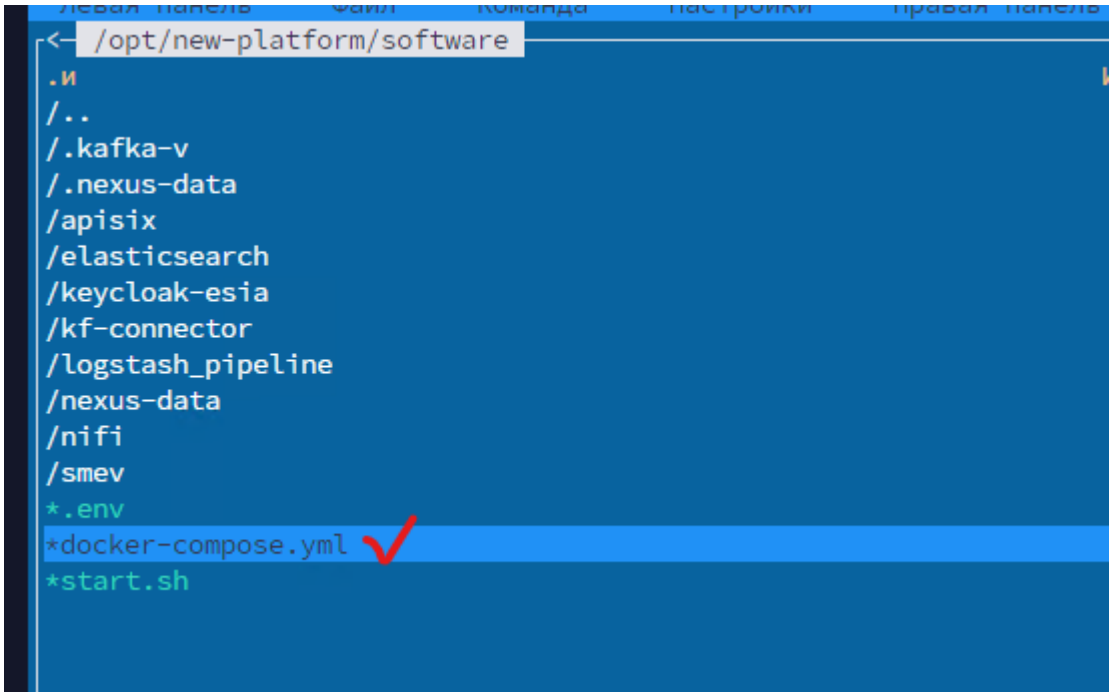


Рисунок 19 — пример файла .env инфраструктурного слоя (2)

4. Развёртывание слоя продуктовых микросервисов

После того как инфраструктурный слой развёрнут и работоспособен, разворачивается слой продуктовых микросервисов. Слой поставляется в виде архива с файлом docker-compose, файлом .env и образами сервисов (включая APISIX и Nginx). Для начала работ необходимо перейти в apps

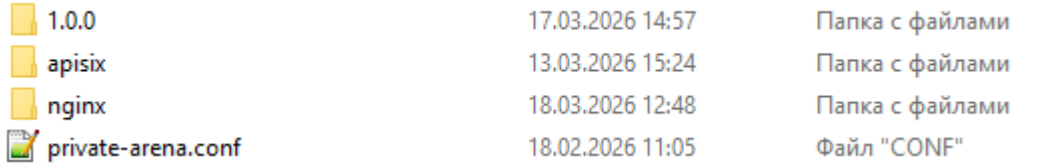


Рисунок 20 — состав поставки слоя приложений (1)

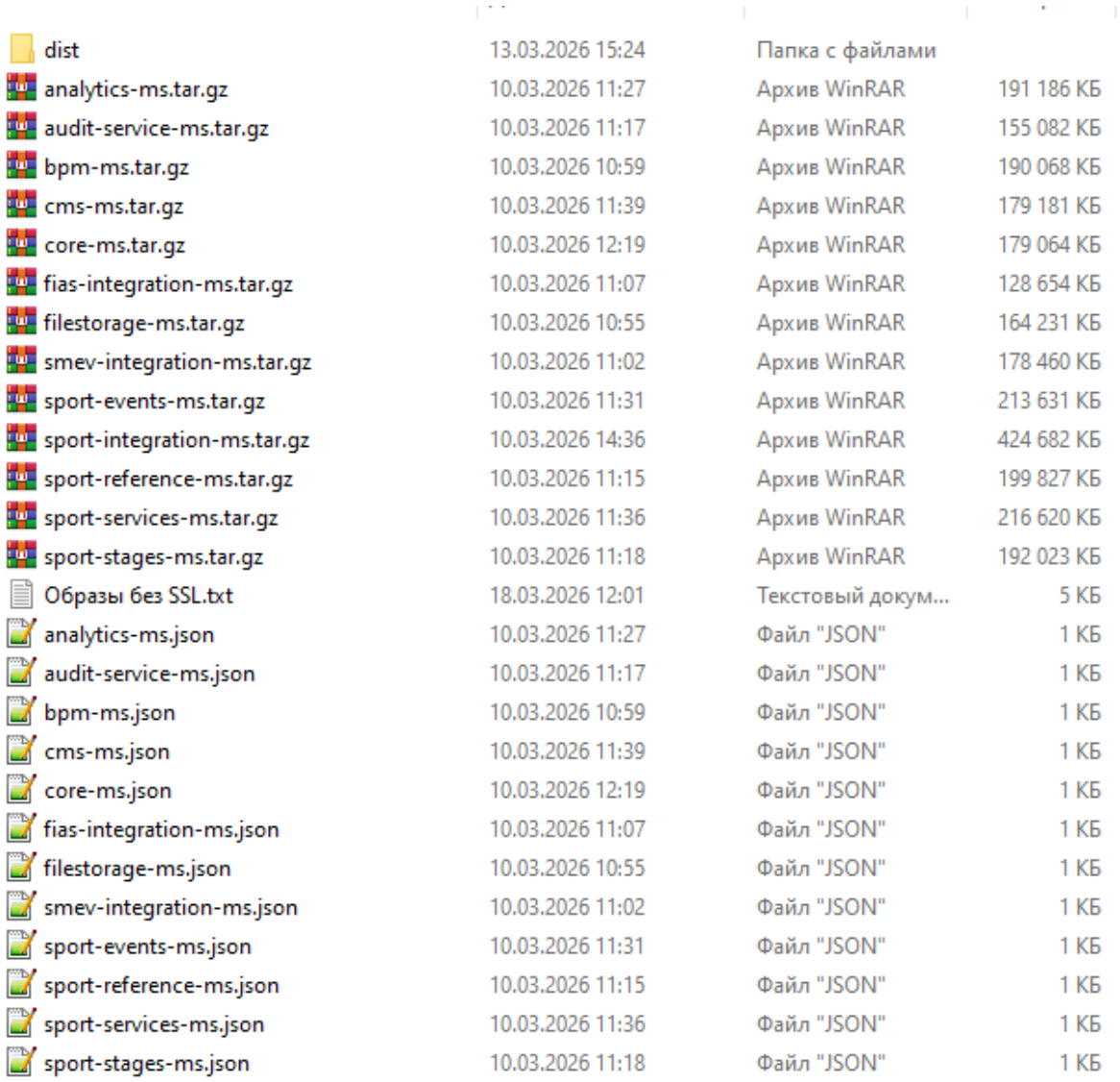


Рисунок 21 — состав поставки слоя приложений (2)

4.1. Загрузка образов

Загрузить поставляемые образы в локальный реестр Docker. Каждый образ загружается командой:

```
docker load -i <name>.tar.gz
```

```
# URLs
FRONT_URL=https://your-domain
FRONT_PUBLIC_URL=https://public-your-domain
API_URL=https://private-your-domain/api
API_PUBLIC_URL=https://public-your-domain/api

# DB (PostgreSQL)
DB_DRIVER=org.postgresql.Driver
DB_HOST=
DB_PORT=
DB_USER=
DB_PASSWORD=
DB_JDBC=jdbc:postgresql://${DB_HOST}:${DB_PORT}

# Kafka / Redis / Elastic
KAFKA_HOST=
KAFKA_PORT=9093
KAFKA_URL=${KAFKA_HOST}:${KAFKA_PORT}
REDIS_HOST=
REDIS_PORT=6379
ELASTIC_HOST=
ELASTIC_PORT=9200
ELASTIC_URL=${ELASTIC_HOST}:${ELASTIC_PORT}

# Keycloak (M2M + issuers)
KC_CLIENT_ID=m2m
KC_CLIENT_SECRET=secret_key
KC_SERVER=http://${DB_HOST}:9091/auth
KC_REALM=test # имя реалма
KC_TOKEN_URI=${KC_SERVER}/realms/${KC_REALM}/protocol/openid-connect/token
KC_ISSUERS=http://${DB_HOST}:9091/auth/realms/${KC_REALM},https://public-your-domain/auth/realms/${KC_REALM},https://private-your-domain/auth/rea

# Other infra
MINIO_URL=http://${DB_HOST}:6000
MINIO_ACCESS_KEY=  # указать ключ сгенерированный в MINIO
MINIO_SECRET_KEY=  # указать ключ сгенерированный в MINIO
MINIO_BUCKET=files

# External services
INTEGRATION_URL=${DB_HOST}:8091
FIAS_BASE_URL=https://fias-public-service.nalog.ru
FIAS_AUTH_NAME=master-token
FIAS_AUTH_KEY= #ключ для запросов в ФИАС

# Common
SERVER_PORT=8080
IMAGE_TAG=1.0.0
```

Рисунок 22 — загрузка образов

4.2. Запуск backend-микросервисов

Заполнить файл .env, указав параметры подключения к базе данных, Kafka, MinIO (включая ранее сохранённые Access Key и Secret Key), Redis и Keycloak (включая секретный ключ m2m). Запустить микросервисы:

```
docker compose up -d
```

Убедиться, что контейнеры находятся в состоянии running и успешно подключились к зависимым сервисам (Keycloak, Postgres, Elasticsearch, Redis).

```
        # Запросы через private-apisix
    location /api/ {
        proxy_pass http://apisix_private:9080/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
```

Рисунок 23 — пример файла .env слоя микросервисов

```
    root /opt/new-platform/front/private;
    index index.html;
```

Рисунок 24 — запуск backend-микросервисов

4.3. Настройка внутреннего Nginx

Внутренний Nginx маршрутизирует трафик закрытой части и аналитики. Поставляемые файлы private.conf и analytics.conf размещаются в каталоге conf.d и применяются перезагрузкой сервиса. В конфигурации указываются домены закрытой части и аналитики вместе с адресами и портами сервисов. Для закрытой части в блоке location /api/ указывается адрес и порт APISIX.

```
# Test Keycloak
location /auth/realms/<realm_name> {
    proxy_pass http://ip_keycloak:9091/auth/realms/<realm_name>;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
    proxy_busy_buffers_size   1024k;
    proxy_buffers   16 512k;
    proxy_buffer_size   768k;

}

location /auth/resources {
    proxy_pass http://ip_keycloak:9091/auth/resources;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
    proxy_busy_buffers_size   1024k;
    proxy_buffers   16 512k;
    proxy_buffer_size   768k;
}
```

Рисунок 25 — конфигурация внутреннего Nginx

4.4. Размещение веб-приложения закрытой части

Поставляемую сборку фронтенда разместить в каталоге, указанном в конфигурации Nginx (по умолчанию /opt/new-platform/front/private). Для авторизации через Keycloak указать адрес Keycloak и имя реалма (по умолчанию test).

```
server {

    listen 83;
    server_name analytics-your-domain;


    location / {
        proxy_pass http://ip_superset:8088/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Forwarded-Proto "https";
    }
}
```

Рисунок 26 — настройка авторизации фронтенда через Keycloak

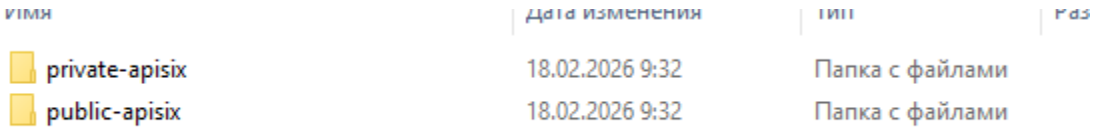


Рисунок 27 — настройка проксирования в Superset (analytics.conf)

4.5. Настройка шлюзов APISIX

APISIX используется как API Gateway. Предусмотрены два независимых контура — private (UI на порту 9000) и public (порт 9005). Для каждого поставляется отдельный docker-compose и набор томов с преднастроенными маршрутами. При запуске APISIX применяет конфигурацию автоматически. В блоке Upstream List указать актуальные адреса backend-серверов. Публичный контур настраивается аналогично.

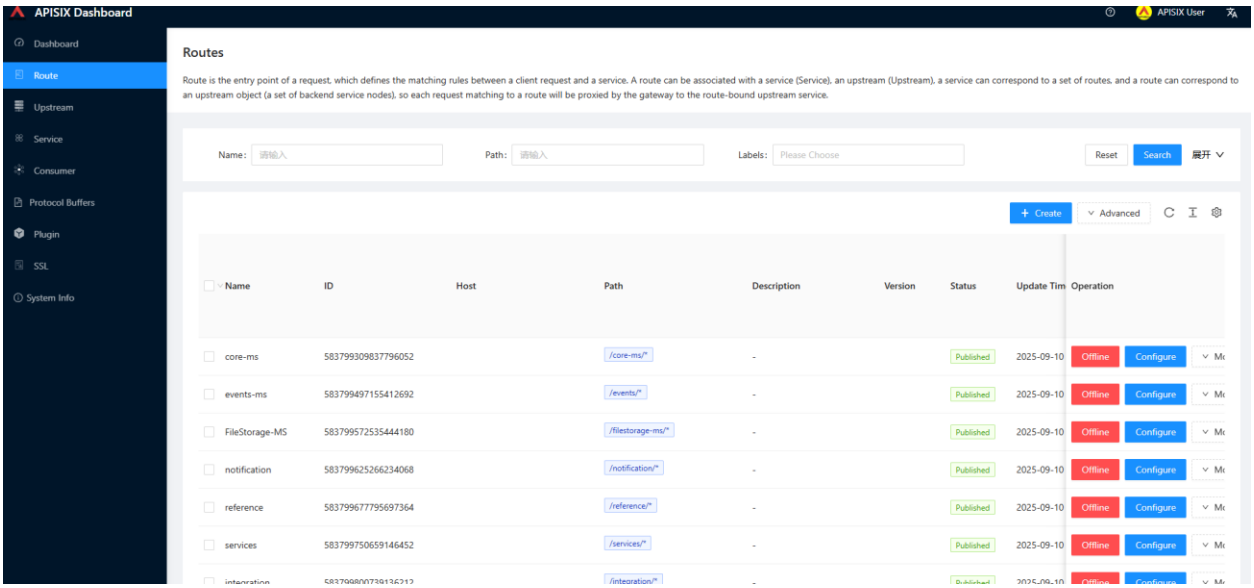


Рисунок 28 — маршруты APISIX (1)

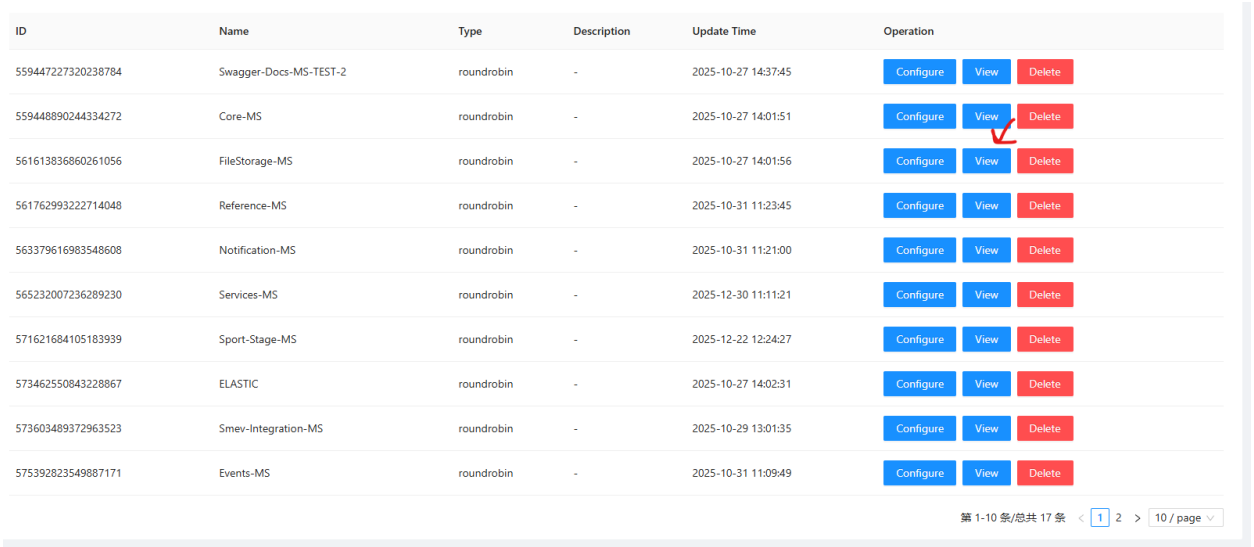


Рисунок 29 — маршруты APISIX (2)



Рисунок 30 — сервисы APISIX

```
    listen 81;
    server_name public-your-domain;

    root /opt/new-platform/front/public;
    index index.html;

    location / {
        try_files $uri /index.html;
    }
```

Рисунок 31 — блок Upstream List (1)

```
# Запросы через public-apisix
location /api/ {
    proxy_pass http://public_apisix_ip:9085/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;

}
```

Рисунок 32 — блок Upstream List (2)

4.6. Развёртывание публичной части

Для публичной части разворачивается публичный Nginx с конфигурацией public.conf (размещается в стандартном каталоге Nginx и применяется перезагрузкой) и размещается публичная сборка фронтенда. Маршрутизация к программным интерфейсам выполняется через публичный APISIX (настройка аналогична закрытому контуру).

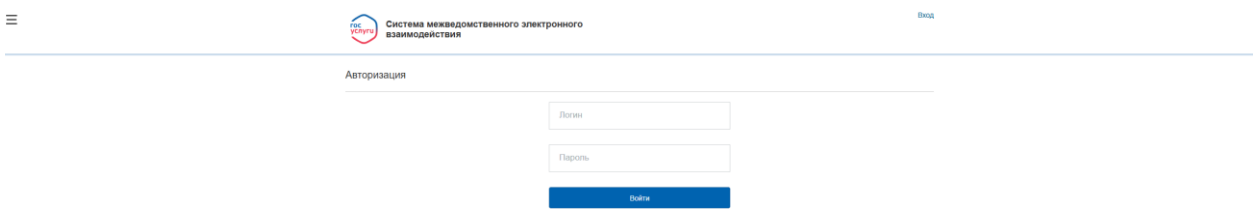


Рисунок 33 — конфигурация публичного Nginx

5. Развёртывание слоя интеграции со СМЭВ

Слой интеграции со СМЭВ поставляется в виде docker-compose и образов (backend-сервис и smev-adapter). Адаптер доступен по адресу ip_smev_adapter:8085; логин и пароль находятся в корне каталога поставки данного слоя. Для начала работ необходимо перейти в smev. Порядок настройки:

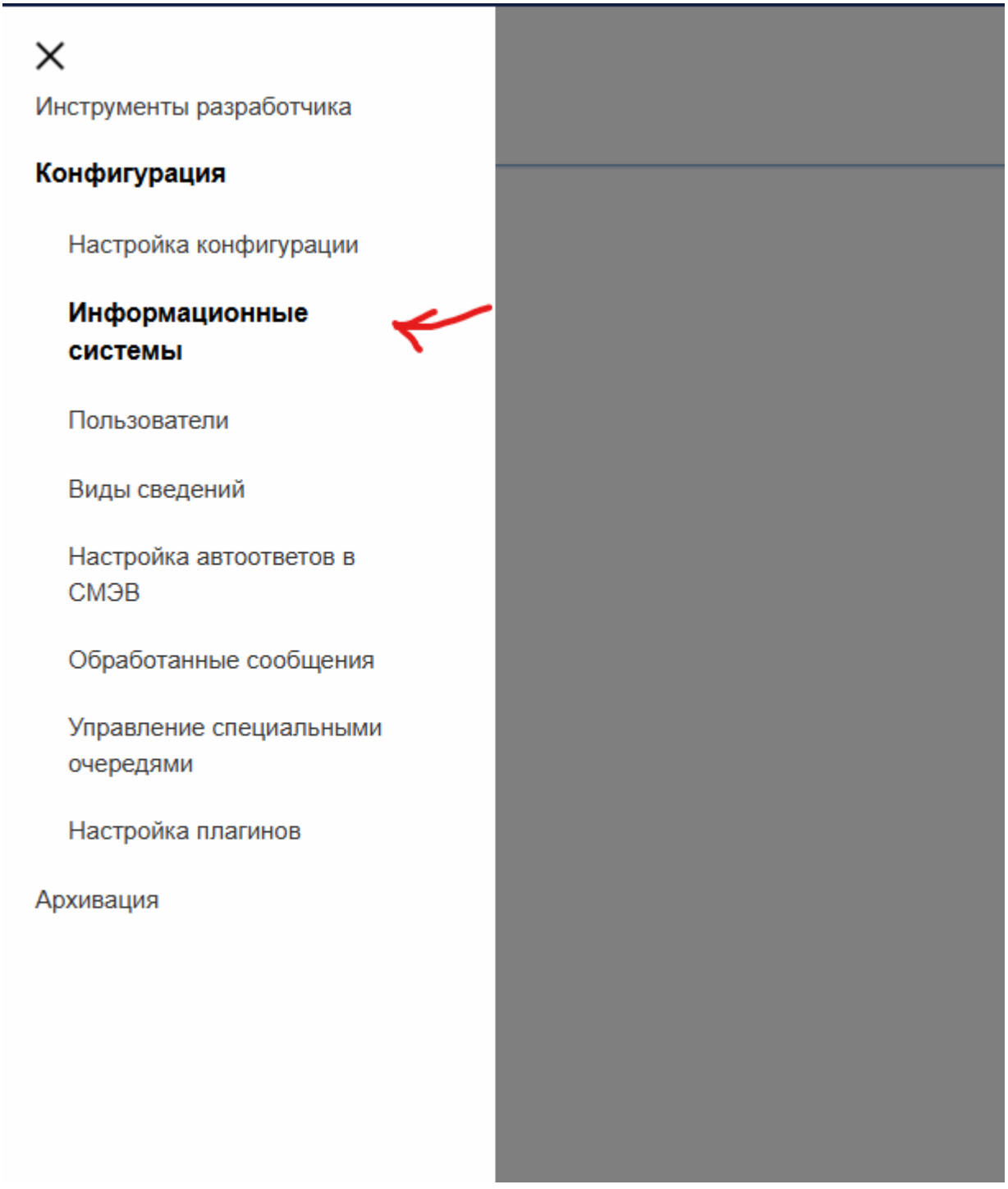


Рисунок 34 — запуск слоя интеграции со СМЭВ

1) в разделе информационных систем нажать «Добавить» и указать мнемонику ключевого контейнера ЭП-ОВ (контейнер подключается по аналогии с Keycloak);

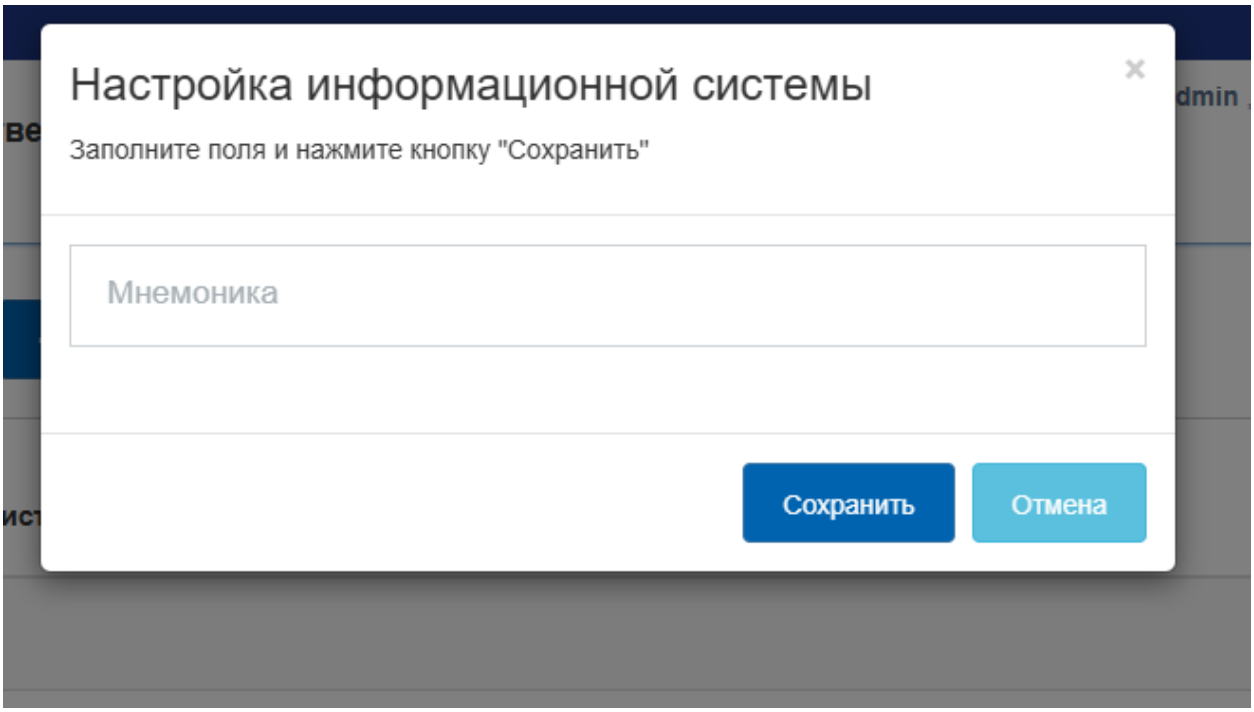


Рисунок 35 — добавление информационной системы



Рисунок 36 — подключение контейнера ЭП-ОВ (1)

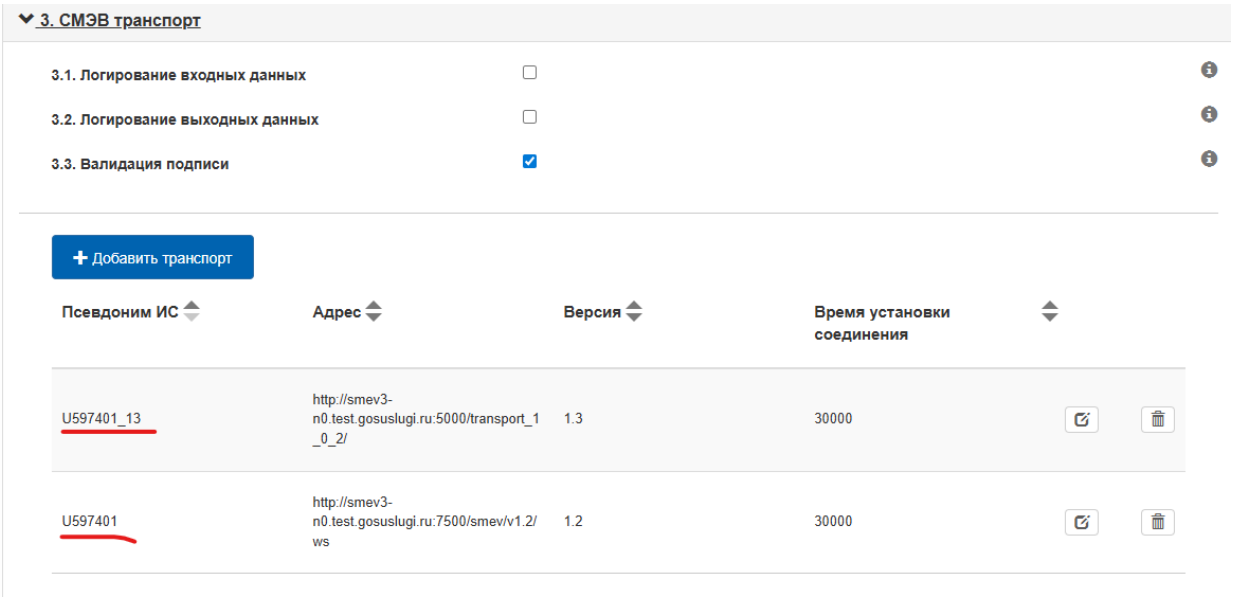


Рисунок 37 — подключение контейнера ЭП-ОВ (2)

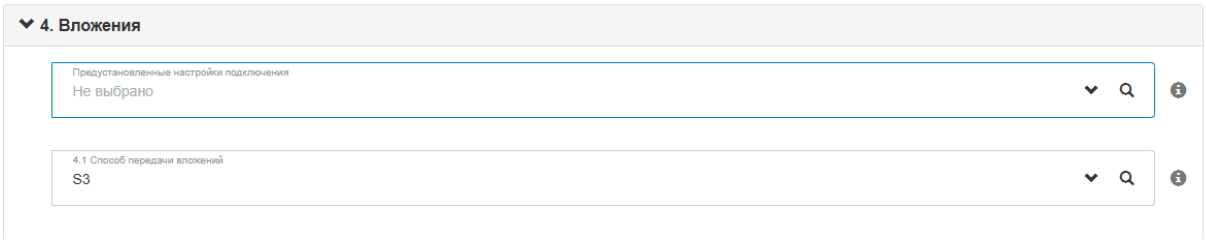


Рисунок 38 — подключение контейнера ЭП-ОВ (3)

2) во вкладке «Подпись» ввести данные ЭП-ОВ;

3) во вкладке «СМЭВ Транспорт» создать транспорт требуемых версий (в примере — 1.2 и 1.3), запомнив имена псевдонимов ИС;

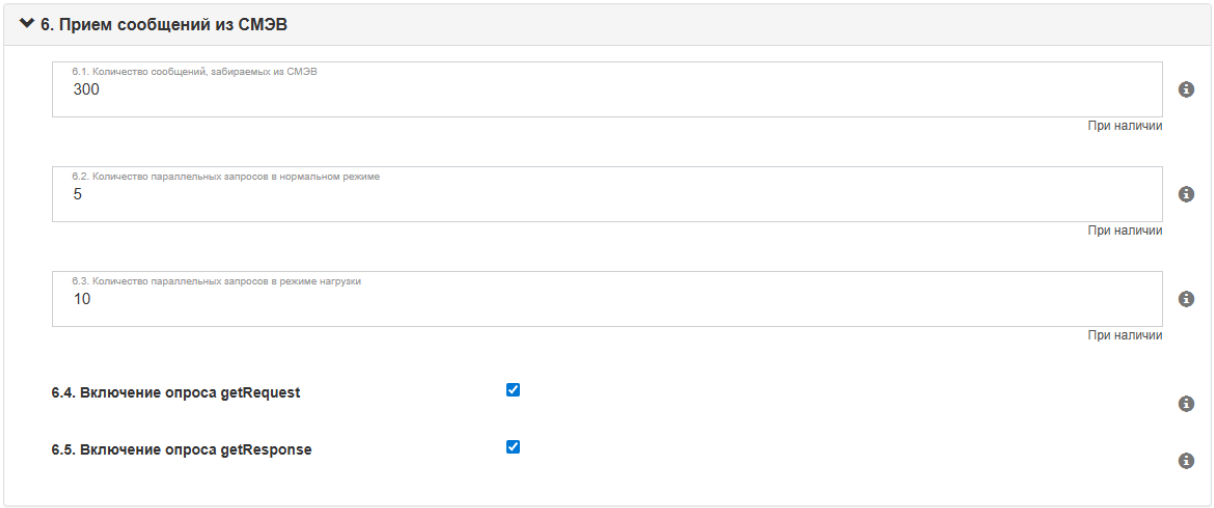


Рисунок 39 — вкладка «Подпись»

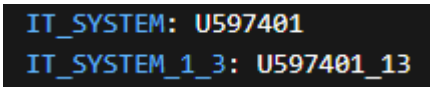


Рисунок 40 — создание транспорта СМЭВ

4) во вкладке «Вложения» указать параметры S3-хранилища для вложений СМЭВ;

5) в разделе приёма сообщений СМЭВ проставить необходимые галочки;

6) в docker-compose.yml адаптера указать параметры транспортов и адреса, по которым развёрнуты приложения Системы.

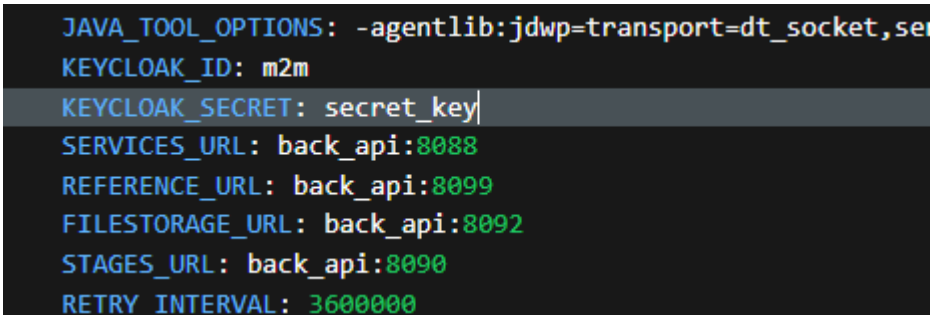


Рисунок 41 — настройка приёма сообщений СМЭВ

После настройки убедиться, что сервисы корректно взаимодействуют со СМЭВ, а сертификаты и каналы связи настроены.

6. Проверка работоспособности

После завершения развёртывания выполнить комплексную проверку:

– доступность всех инфраструктурных сервисов;

– корректность работы backend и установленных соединений с зависимыми сервисами;

– доступность фронтенда закрытой и публичной частей;

– работу маршрутов через APISIX и Nginx;

– прохождение потоков данных через Kafka и их индексацию в Elasticsearch;

– корректность авторизации через Keycloak и ЕСИА;

– взаимодействие слоя интеграции со СМЭВ.

Порядок развёртывания обязателен: сначала инфраструктура, затем восстановление данных и настройка инфраструктурных сервисов, далее backend, шлюзы APISIX, Nginx и фронтенд, и в последнюю очередь — слой

интеграции со СМЭВ. Соблюдение порядка обеспечивает корректную инициализацию Системы.