Руководство системного администратора

Геоинформационная система «Территория» (ТерГИС)

Аннотация

Настоящий документ представляет собой руководство администратора (далее– Руководство) Геоинформационной системы "Территория" (далее – ТерГИС).

Руководство определяет основные функции, а также последовательность действий и порядок работы системного администратора ТерГИС (далее – Администратор).

Настоящий документ является составной частью комплекта рабочей документации ТерГИС.

Перед началом работы администратора с Системой рекомендуется внимательно ознакомиться с настоящим руководством.

Содержание

Список используемых терминов, сокращений и обозначений

Термин, сокращение, обозначение	**Расшифровка**
Администратор	Администратор системы
БД	База данных
ГИС	Геоинформационная система
ТерГИС	Геоинформационная система «Территория»
Руководство	Руководство администратора
Система	Геоинформационная система "Территория" (ТерГИС)
СУБД	Система управления базами данных

1. Общие сведения о программном обеспечении

1.1 Назначение Системы

Система предназначена для обеспечения сбора, обработки, отображения, хранения, анализа и распространения пространственных данных и связанной с ними информации об объектах. Система может применяться для эффективного решения задач инвентаризации и управления территориально распределенными объектами.

Система обеспечивает выполнение следующих функций: хранение, импорт, экспорт пространственных данных; управление картографическими слоями; создание, публикация и управление пользовательскими слоями.

1.2 Описание функциональных возможностей

Для пользователей ТерГИС предоставляет следующие функциональные возможности:

- управления картой;
- управления содержанием карты;
- выбора объектов;
- измерения;
- поиска и просмотра сведений об объектах;
- хранения, импорта и экспорта пространственных данных и их визуализации в окне карты;
- формирования ссылки на карту;
- создания, управления и редактирования пользовательских слоев;
- формирования отчетов и печати карты.

1.3 Сведения об аппаратных и программных средствах

Работа ТерГИС обеспечивается следующими аппаратными средствами:

- серверами (виртуальными машинами);
- персональными компьютерами рабочих мест пользователей;
- телекоммуникационным оборудованием, обеспечивающим обмен данными между северами и между серверами и рабочими местами.

1.4 Сведения о серверах, обеспечивающих выполнение программы

Минимальная конфигурация серверов для развертывания ТерГИС приведены в таблице:

№	Роль	Поддержка виртуализации	Кол-во ядер процессора, шт	Оперативная память, ГБ	Операционная система	Хранилище, ГБ	Наименование сервисов
1	Сервер приложения 1	да	4	8	ОС семейства Linux: CentOS, AstraLinux, RedOS, Ubuntu	100	map_api; admin_api; layer_export_api; layer_import_api
2	Сервер приложения 2	да	4	8	ОС семейства Linux: CentOS, AstraLinux, RedOS, Ubuntu	100	auth_api; background_api; сервис кэширования (geowebcache)
3	Сервер БД	да	4	8	ОС семейства Linux: CentOS, AstraLinux, RedOS, Ubuntu	100	PostgreSQL
4	ГИС сервер 1	да	4	8	ОС семейства Linux: CentOS, AstraLinux, RedOS, Ubuntu	100	ГИС-сервер (geoserver)
5	ГИС сервер 2	да	4	8	ОС семейства Linux: CentOS, AstraLinux, RedOS, Ubuntu	100	ГИС-сервер (geoserver)

1.5 Сведения о персональных компьютерах пользователей

Требования к рабочему месту пользователя:

- операционная система, поддерживающая работу вэб-браузера;
- веб-браузер, поддерживающий технологии HTML5 и CSS 3.0 (рекомендуется использовать актуальные версии браузеров Google Chrome, Yandex или Mozilla Firefox);
- наличие свободной ОЗУ в соответствии с требованиями веб-браузера;
- наличие свободного дискового пространства для установки веб-браузера.

1.6 Языки программирования, на которых написана программа

Языками программирования являются: JavaScript; PL/pgSQL; Java.

1.7 Требования к квалификации администратора

Администратор выполняет следующие функции:

- установка, настройка и обновление операционных систем (семейства Linux и Windows) на серверах и рабочих станциях;
- установка прикладного программного обеспечения;
- создание, удаление, изменение и блокировка учетных записей пользователей, назначение прав доступа и управление паролями;
- настройка и управление сетевыми службами;
- управление дисковым пространством;
- анализ системных журналов для выявления проблем и угроз безопасности;
- установка обновлений безопасности и исправлений для операционных систем и системного программного обеспечения;
- реализация и поддержание политик безопасности, включая настройку брандмауэров, антивирусного программного обеспечения, систем обнаружения вторжений и других средств защиты;
- поддержание работоспособности технических средств;
- настройка и конфигурирование системных программных средств.

2 Подготовка к работе

2.1 Архитектура решения

Архитектурная схема ТерГИС представлена на рисунке (Рисунок 1).

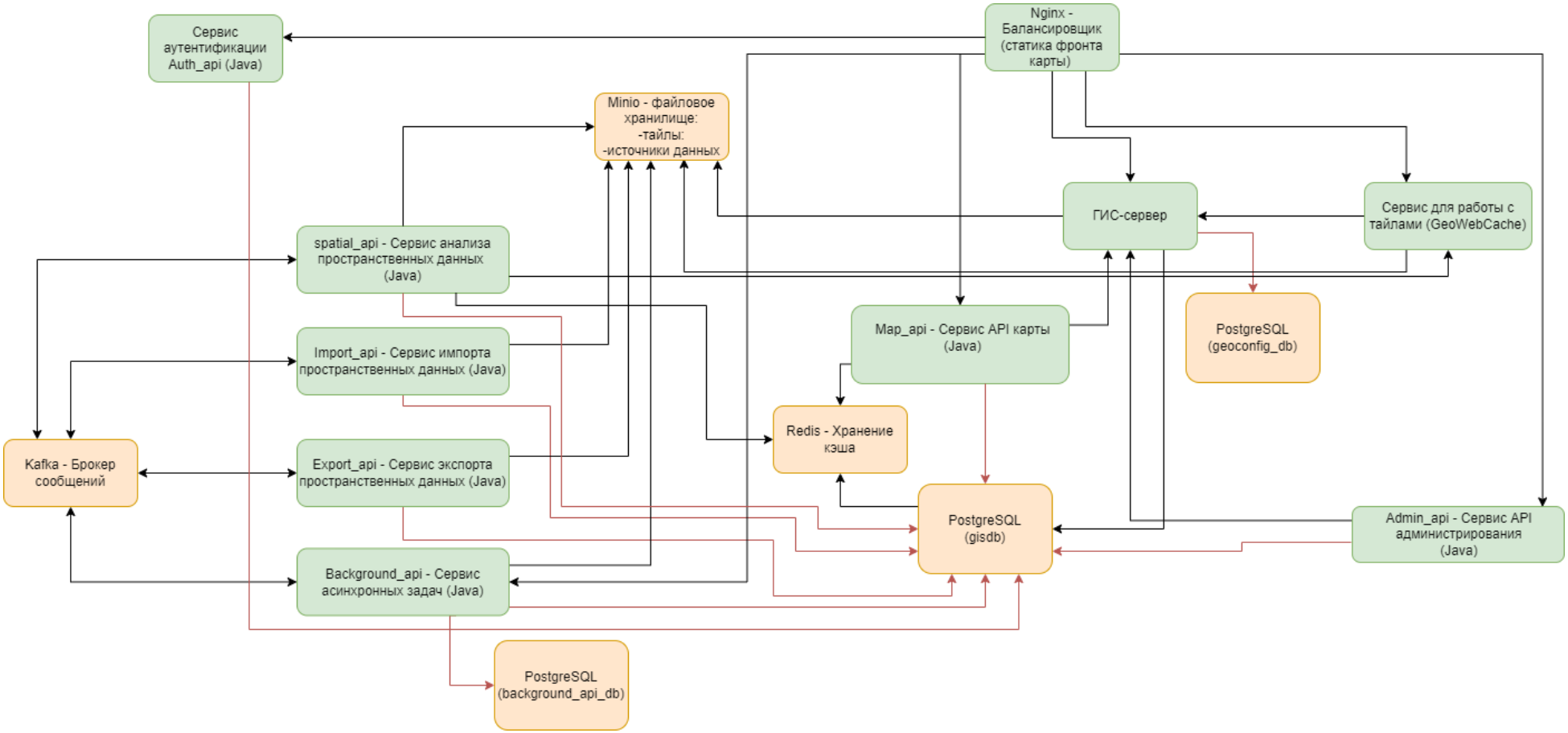


Рисунок 1. Архитектурная схема ТерГИС

3 Установка программных компонент

Состав дистрибутива:

1) geoserver.tar - образ гис-сервера;
2) geowebcache.tar - образ сервиса для работы с тайлами;
3) admin_api.tar - образ сервиса API администрирования;
4) auth_api.tar - образ сервиса аутентификации;
5) background_api.tar - образ сервиса асинхронных задач;
6) export_api.tar - образ сервиса экспорта пространственных данных;
7) import_api.tar - образ сервиса импорта пространственных данных;
8) map.tar - образ статики фронта карты;
9) map_api.tar - образ сервиса API карты;
10) spatial_relation_api.tar – образ сервиса анализа пространственных данных.

3.1 PostgeSQL и PostGIS

3.1.1 Назначение

Хранение данных, в том числе пространственных.

3.1.2 Установка

Предполагается стандартная установка, изложенная в документации на СУБД www.postgresql.org/docs/ и на расширение PostGIS www.postgis.net/documentation/getting_started/.

3.1.3 Настройка

Необходимо создать базы данных:

1) База для хранения слоев и пространственных данных (gisdb).

Требуется расширение postgis create database gisdb owner gis;

\c gisdb

GRANT ALL ON TABLE public.spatial_ref_sys TO gis;

2) База для хранения конфигурации геосервера (geoconfig_db)

create database geoconfig_db owner gis;

3) База для хранения фоновых задач (background_api_db). Требуется расширение postgis

create database background_api_db owner gis;

\c background_api_db

GRANT ALL ON TABLE public.spatial_ref_sys TO gis;

Создать пользователя **gis** - owner баз gisdb, geoconfig_db, background_api_db

3.2 Kafka

3.2.1 Назначение

Обеспечение взаимодействия между сервисами background_api и export_api/import_api

3.2.2 Установка

Предполагается использование Kafka в докер-образе. Установка изложена в документации https://hub.docker.com/r/apache/kafka

3.2.3 Настройка

1) Создать топики:
- export;
- export-response;
- import;
- import-response;
- spatial-relation;
- spatial-relation-response.

2) Создать следующие group-id:
- layer-export-group-id;
- layer-import-group-id;
- background-api-group-id;
- spatial-relation-group-id;

Группа	**Читает из топиков**
layer-export-group-id	export
layer-import-group-id	import
background-api-group-id	export-response import-response spatial-relation-response
spatial-relation-group-id	spatial-relation

3.3 Redis

3.3.1 Назначение

Кеширование данных при определенных запросах к map_api - в части формирования участка с расчетом площади в МСК и поиска объектов в области

3.3.2 Установка

Предполагается использование Redis в докер-образе. Установка изложена в документации https://hub.docker.com/r/bitnami/redis

3.3.3 Настройка

Не требуется

3.4 Minio

3.4.1 Назначение

Хранение тайлов, растровых данных (geotiff), файлов для импорта, файлов результатов экспорта.

3.4.2 Установка

Предполагается использование Minio в докер-образе. Установка изложена в документации https://hub.docker.com/r/bitnami/minio

3.4.3 Настройка

Необходимо создать следующие бакеты:

- gwc-tiles;
- import-api-bucket;
- export-api-bucket;
- bucket-object-documents;
- spatial-relation-api-bucket.

3.5 GeoServer

3.5.1 Версия и назначение

Используется версия 2.26.2. GeoServer отвечает за рендеринг изображения на основе пространственных данных.

3.5.2 Установка

3.5.2.1 Настройка NFS на стороне сервера

Выполните в терминале следующие команды:

```
sudo yum -y install nfs-utils
sudo mkdir -p /var/geoserver/data_dir
sudo mkdir -p /var/log/geoserver
sudo systemctl enable nfs-server.service
sudo systemctl start nfs-server.service

echo "/var/geoserver/data_dir <ip nfs server>/24(rw,sync,no_wdelay,insecure_locks,no_root_squash)" | sudo tee /etc/exports > /dev/null
sudo exportfs -a
```

3.5.2.2 Настройка NFS на стороне клиента

Выполните в терминале следующие команды:

```
sudo yum -y install nfs-utils
sudo mkdir -p /var/geoserver
sudo mkdir -p /var/log/geoserver
echo "<ip nfs server>:/var/geoserver/data_dir /var/geoserver/data_dir nfs defaults 0 0" | sudo tee -a /etc/fstab > /dev/null
sudo mount -a
```

Предварительные требования к виртуальным машинам, на которых разворачивается GeoServer:

- Настроена nfs share
- Создана БД Postgres geoconfig_db (см. пункт 1.1), в ней создана схема geo_config
- Установлен docker version >= 23.0.5

- Установлен docker-compose version >=1.25.5

Первичная установка GeoServer производится на одной виртуальной машине. После окончания установки GeoServer может быть запущен на следующих нодах.

Чтобы проверить корректность установки, нужно:

– В каталоге /var/geoserver проверить наличие каталога data_dir и доступности его со всех нод GeoServer

– Создать в каталоге /var/geoserver файлы docker-compose.yml и geoserver-env.env, заполнив согласно сайзинга стенда. Содержимое файлов представлено ниже.

Содержимое файла docker-compose.yml:

```
version: "3.6"
services:
  geoserver:
    restart: always
    image: <tag-image-geoserver> # предоставленный образ geoserver, опубликованный в nexus
    container_name: geoserver-node-master
    network_mode: "host"
    volumes:
      - type: bind
        source: /var/log/geoserver
        target: /var/log/geoserver
      - type: bind
        source: ./data_dir
        target: /var/geoserver/data_dir
    env_file:
      - geoserver-env.env
```

Содержимое файла geoserver-env.env

```
GEOSERVER_LOG_LOCATION=/var/log/geoserver/node-master.log
GEOSERVER_DATA_DIR=/var/geoserver/data_dir
JAVA_OPTS=-server -XX:+UseContainerSupport -XX:MaxRAMPercentage=85.0 -XX:SoftRefLRUPolicyMSPerMB=36000 -Dfile.encoding=UTF-8 -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005 -Dorg.eclipse.jetty.server.Request.maxFormContentSize=10000000 -DGEOSERVER_XSTREAM_WHITELIST=ru.tectus_it.geoserver.rest.dto.* -Dorg.geotools.localDateTimeHandling=true
TZ=Europe/Moscow
```

Запустить GeoServer

```
cd /var/geoserver/
docker-compose up –d
```

3.5.2.3 Проверка установки GeoServer

Проверить корректность запуска GeoServer. В data_dir должна появиться директория jdbcconfig с настройками модуля. После остановить, выполнив docker-compose down.

3.5.2.4 Конфигурирование для создания кластера

1) Проверить наличие схемы geo_config в БД geoconfig_db. При необходимости создать схему.

2) В файле data_dir/jdbcconfig/jdbcconfig.properties указать параметры подключения к БД.

В параметрах подключения указать созданную схему.

В параметрах username и password указать креды для подключения к БД

Пример заполнения файла jdbcconfig.properties:

```
initdb=true
pool.timeBetweenEvictionRunsMillis=-1L
import=false
pool.poolPreparedStatements=true
pool.testWhileIdle=false
pool.validationQuery=SELECT now()
pool.minIdle=4
enabled=true
pool.maxOpenPreparedStatements=50
password=gis
jdbcUrl=jdbc\:postgresql\://<ip                                    db>\:<port
db>/geoconfig_db?currentSchema\=geo_config
driverClassName=org.postgresql.Driver
pool.maxActive=10
initScript=jdbcconfig/scripts/initdb.postgres.sql
pool.testOnBorrow=true
username=gis
```

3) Запустить первую ноду. Конфигурация должна импортироваться в БД.
4) Проверить наличие таблиц в БД. Если все шаги пройдены успешно, запустить остальные ноды.

3.5.2.5 Проверка запуска GeoServer

Перейти в браузере на страницу панели администрирования Geoserver [хост:8080]/geoserver/web/. Если все было установлено корректно, то по этой ссылке будут доступны административные утилиты, поставляемые с GeoServer.

3.5.2.6 Подключение S3 хранилища к GeoServer

В файле geoserver-env.env необходимо добавить переменные окружения для подключения к S3 хранилищу:

IIO_S3_AWS_USER

IIO_S3_AWS_PASSWORD

IIO_S3_AWS_ENDPOINT

IIO_S3_AWS_REGION

Пример заполнения

```
IIO_S3_AWS_USER=user// access_key
IIO_S3_AWS_PASSWORD=password // secret key
IIO_S3_AWS_ENDPOINT=http://<ip_host>:9000
IIO_S3_AWS_REGION=us-east-1
```

Внимание! Плагин в GeoServer (GEOTIFF COG) воспринимает переменную IIO_S3_AWS_ENDPOINT только с указанием ip-адреса.

3.6 Map_api

3.6.1 Версия и назначение

Текущая версия – 3.25.1804.

Набор API для работы карты. Поддерживает редактирование объектов слоя, инструменты измерения, выбор объектов, получение информации по объекту в точке, сохранение ссылки на карту, работу с атрибутами, работа с пользовательскими слоями, API работы со слоями (получение дерева слоев, наборов карт). Реализован с применением Java, Spring Boot 3, GDAL 3.10

3.6.2 Установка

Разархивируйте архив с образом map-api локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i map-api.tar
docker login <nexus-url>
docker push <nuxus-url>/map_api:release
```

Создайте папку приложения

```
cd /opt
mkdir map_api
cd /map_api
mkdir config
```

Создайте docker-compose.yml в /opt/map_api следующего содержания

В комментариях указаны места, где необходимо подставить реальные урлы и креды инфраструктурных компонентов и сервисов.

```
version: "3.3"
services:
  map_api:
    container_name: map_api
    image: <nexus-url>/map_api:release ### указываем url нексуса
    environment:
      - TZ=Europe/Moscow
      - SPRING_CONFIG_NAME=map_api
      - SPRING_PROFILES_ACTIVE=dev
            - SPRING_DATASOURCE_URL=jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host и порт gisdb
      - SPRING_DATASOURCE_USERNAME=gis # указываем username gisdb,
      - SPRING_DATASOURCE_PASSWORD=gis # указываем password gisdb
      - SPRING_DATA_REDIS_HOST=<ip_host> # указываем host redis
      - SPRING_DATA_REDIS_PORT=6379 # указываем порт redis
      - SPRING_DATA_REDIS_PASSWORD=password # указываем пароль redis
      - SPRING_KAFKA_BOOTSTRAP-SERVERS=<ip_host>:9092 # указываем url и порт для подключения к кафке
      - KAFKA_USERNAME=login # указываем логин кафки
      - KAFKA_PASSWORD=password # указываем пароль кафки
       - GEOSERVER_URL=http://<ip_host>:8080/geoserver # указываем url мастер-геосервера
      - GEOSERVER_NODES="{'http://<ip_host>:8080/geoserver','http://<ip_host>:8080/geoserver'}" # указываем urlы всех геосерверов
      - MINIO_PROPERTIES_URL=http://<ip_host>:9000 # указываем url minio
      - MINIO_PROPERTIES_SECRETKEY=secret # указываем secret key minio
      - MINIO_PROPERTIES_ACCESSKEY=access # указываем access key minio
      - FEIGN_GEO-WEB-CACHE-CLIENT_URL=http://<ip_host>:8080/geowebcache/rest/ # указываем url до реста geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_USERNAME=user # указываем username geowebcache
```

```
      - FEIGN_GEO-WEB-CACHE-CLIENT_PASSWORD=pass # указываем password geowebcache
      - APPLICATION_HOME_URL=https://tergis-front.local  # урл по которому открывается карта
      - APPLICATION_SECURITY_MODE=JWT-ROLE_DB
      - APPLICATION_SECURITY_ROLE-ADMINISTRATION-MODE=ROLE_DB
    network_mode: host
    volumes:
      - type: bind
        source: ./config
        target: /opt/map_api/config
    stdin_open: true
    restart: always
```

В папке config создайте файл map_api.yaml следующего содержания.

```
server:
  port: 8081
  servlet:
    context-path: /map_api
spring:
  jpa:
    generate-ddl: false
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
        use_sql_comments: false
        type: trace
        temp:
          use_jdbc_metadata_defaults: false
        show_sql: false
    database: postgresql
    open-in-view: false
  liquibase:
    change-log: classpath:changelog-master.xml
  cache:
    type: redis
    redis:
      time-to-live: 600000
  datasource:
    hikari:
      poolName: HikariPool-Postgres
      idleTimeout: 30000
      maxLifetime: 600000
```

```
    minimumIdle: 0
    maximumPoolSize: 5
  driverClassName: org.postgresql.Driver
  url: jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host gisdb
  username: gis # указываем username gisdb
  password: gis # указываем password gisdb
servlet:
  multipart:
    max-request-size: 100MB
    max-file-size: 30MB
data:
  rest:
    detection-strategy: annotated
  redis:
      host: <ip_host> # указываем host redis
      port: 6379 # указываем порт redis
      password: password # указываем пароль redis
kafka:
  producer:
    value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
    key-serializer: org.apache.kafka.common.serialization.StringSerializer
  consumer:
    group-id: spatial-relation-group
    client-id: spatial-relation-consumer-1
    auto-offset-reset: earliest
    value-deserializer: org.springframework.kafka.support.serializer.JsonDeserializer
    key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    properties:
      security.protocol: SASL_PLAINTEXT
      sasl.mechanism: PLAIN
```

```
                    sasl.jaas.config:    org.apache.kafka.common.security.plain.PlainLoginModule
required username="${KAFKA_USERNAME}" password="${KAFKA_PASSWORD}";
                spring:
                  json:
                    trusted:
                      packages: ru.digital.league.sc.*
                    type:
                      mapping:
ru.digital.league.sc.map.background.service.out.KafkaValueDto:ru.digital.league.sc.map.spatialr
elation.dto.SpatialRelationTaskDto
                  bootstrap-servers: <ip_host>:9092 # указываем url и порт для подключения к кафке
              security:
                oauth2:
                  resourceserver:
                    jwt:
                      issuer-uri: localhost


            keycloak:
              auth-server-url: localhost
              realm: tergis
              login: admin
              password: admin
            # логгирование
            logging:
              config: file:./config/logback-spring.xml
              level:
                ru:
                  tectus_it:
                    tergis: DEBUG
            management:
              endpoints:
```

```
      enabled-by-default: true
      web:
        base-path: /actuator
    endpoint:
      health:
        show-details: always
  # геосервер
  geoserver:
    login: admin
    url: http://<ip_host>:8080/geoserver # указываем url мастер-геосервера
    nodes:    "{'http://<ip_host>:8080/geoserver','http://<ip_host>:8080/geoserver'}"    # указываем urlы всех геосерверов
    log_body: true
    password: geoserver
    auth:
      reload:
        url: /eir/invalidate
        filter: Map API Session Key


  application:
    home:
      url: http://<ip_host>
    feature:
      location:
        use-referer: true
    geotools:
      fracture:
        threshold: 0.6
    layers:
      vector:
```

```
      workspace: TERGIS
    raster:
      workspace: TERGIS
    external:
      workspace: EXTERNAL
    group:
      workspace: TERGIS
  kafka:
    topic:
      spatial-relation: spatial-relation
      spatial-relation-response: spatial-relation-response
  security:
    mode: JWT-ROLE_DB
      role-administration-mode: ROLE_DB

#Найстройка filestorage
minio:
  properties:
    url: http://<ip_host>:9000 # указываем url minio
    secretKey: secret # указываем secret key minio
    accessKey: access # указываем access key minio
    buckets:
      OBJECT_DOCUMENT: bucket-object-documents
      SPATIAL_RELATION: spatial-relation-api-bucket
      THEMATIC_LAYER: local-thematic-layer-bucket

# урлы для feign клиентов
feign:
  geo-web-cache-client:
    url: http://<ip_host>:8080/geowebcache/rest/
```

```
    username: geowebcache
    password: secured
```

В папке config добавьте файл logback-spring.xml следующего содержания

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include resource="org/springframework/boot/logging/logback/defaults.xml"/>
  <property name="LOG_FILE" value="${LOG_FILE:-${LOG_PATH:-${LOG_TEMP:-${java.io.tmpdir:-/tmp}}}/app.log}"/>
  <include resource="org/springframework/boot/logging/logback/console-appender.xml"/>
  <include resource="org/springframework/boot/logging/logback/file-appender.xml"/>
  <root level="INFO">
    <appender-ref ref="CONSOLE"/>
  </root>

  <logger name="ru.digital.league" level="TRACE" additivity="false">
    <appender-ref ref="CONSOLE"/>
  </logger>

</configuration>
```

В папке /opt/map_api выполните команду

```
docker-compose up –d
```

3.7 Import-api

3.7.1 Версия и назначение

Текущая версия – 3.25.1804.

Отвечает за загрузку пространственных данных из различных форматов и создание слоя. Реализован с применением Java, Spring Boot 3, GDAL 3.10

3.7.2 Установка

Разархивируйте архив с образом import-api локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i import-api.tar
docker login <nexus-url>
docker push <nuxus-url>/import-api:release
```

Создайте папку приложения

```
cd /opt
mkdir map_api
cd /import_api
mkdir config
```

Создайте docker-compose.yml в /opt/import_api следующего содержания

В комментариях указаны места, где необходимо подставить реальные урлы и креды инфраструктурных компонентов и сервисов.

.

```
version: "3.3"
services:
  import_api:
    container_name: import_api
    image: <nexus-url>/import_api:release ### указываем url нексуса
    environment:
      - TZ=Europe/Moscow
      - SPRING_CONFIG_NAME=import_api
      - SPRING_PROFILES_ACTIVE=dev
      - SPRING_DATASOURCE_URL=jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host и порт gisdb
      - SPRING_DATASOURCE_USERNAME=gis # указываем username gisdb
      - SPRING_DATASOURCE_PASSWORD=gis # указываем password gisdb
      - SPRING_KAFKA_BOOTSTRAP-SERVERS=<ip_host>:9092 # указываем url и порт для подключения к кафке
      - KAFKA_USERNAME=login # указываем логин кафки
      - KAFKA_PASSWORD=password # указываем пароль кафки
      - GEOSERVER_URL=http://<ip_host>:8080/geoserver # указываем url мастер-геосервера
      - GEOSERVER_NODES="{'http://<ip_host>:8080/geoserver','http://<ip_host>:8080/geoserver'}" # указываем urlы всех геосерверов
      - MINIO_PROPERTIES_URL=http://<ip_host>:9000 # указываем url minio
      - MINIO_PROPERTIES_SECRETKEY=secret # указываем secret key minio
      - MINIO_PROPERTIES_ACCESSKEY=access # указываем access key minio
      - FEIGN_GEO-WEB-CACHE-CLIENT_URL=http://<ip_host>:8080/geowebcache/rest/ # указываем url до реста geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_USERNAME=user # указываем username geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_PASSWORD=pass # указываем password geowebcache
```

```
network_mode: host
volumes:
  - type: bind
    source: ./config
    target: /opt/layer_import_api/config
stdin_open: true
restart: always
```

В папке config создайте файл import_api.yaml следующего содержания

```
server:
  port: 8085
  servlet:
    context-path: /layer_import_api
spring:
  datasource:
    hikari:
      poolName: HikariPool-Postgres
      idleTimeout: 30000
      maxLifetime: 600000
      minimumIdle: 0
      maximumPoolSize: 5
    driverClassName: org.postgresql.Driver
    url: jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host gisdb
    username: gis # указываем username gisdb
    password: gis # указываем password gisdb
  kafka:
    producer:
      value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
      key-serializer: org.apache.kafka.common.serialization.StringSerializer
      properties:
        max.request.size: 10000000
        security.protocol: SASL_PLAINTEXT
        sasl.mechanism: PLAIN
            sasl.jaas.config:    org.apache.kafka.common.security.plain.PlainLoginModule
required username="${KAFKA_USERNAME}" password="${KAFKA_PASSWORD}";
    consumer:
      group-id: layer-import-group-id
      client-id: layer-import-consumer-1
      auto-offset-reset: earliest
```

```
        properties:
          security.protocol: SASL_PLAINTEXT
          sasl.mechanism: PLAIN
                sasl.jaas.config:  org.apache.kafka.common.security.plain.PlainLoginModule required username="${KAFKA_USERNAME}" password="${KAFKA_PASSWORD}";
          spring:
            json:
              trusted:
                packages: ru.digital.league.sc.*
              type:
                mapping: ru.digital.league.sc.map.background.service.out.KafkaValueDto:ru.digital.league.sc.map.import api.dto.ImportTaskDto
          bootstrap-servers: <ip_host>:9092 # указываем url и порт для подключения к кафке
        security:
          oauth2:
            resourceserver:
              jwt:
                issuer-uri: localhost
        cloud:
          openfeign:
            okhttp:
              enabled: true
            client:
              config:
                default:
                  logger-level: full
            httpclient:
              disable-ssl-validation: true
       # логгирование
       logging:
```

```
    level:
      ru:
        tectus_it:
          tergis: TRACE
      config: file:./config/logback-spring.xml

    # Keycloak
    keycloak:
      auth-server-url: localhost
      realm: tergis
      login: admin
      password: admin

    # геосервер
    geoserver:
      login: admin
      url: http://<ip_host>:8080/geoserver # указываем url мастер-геосервера
      nodes:  "{'http://<ip_host>:8080/geoserver','http://<ip_host>:8080/geoserver'}"  # указываем urlы всех геосерверов
      log_body: true
      password: geoserver
      auth:
        reload:
          url: /eir/invalidate
          filter: Map API Session Key

    # Настройки приложения
    application:
      layers:
        vector:
```

```
      workspace: TERGIS
    raster:
      workspace: TERGIS
    external:
      workspace: EXTERNAL
    group:
      workspace: TERGIS
  import:
    file:
      size:
        limit: 2GB
  kafka:
    topic:
      import: import
      import-response: import-response
  security:
    mode: JWT-ROLE_DB
      role-administration-mode: ROLE_DB
# swagger
springdoc:
  swagger-ui:
    path: /swagger-ui-custom.html

#Найстройка filestorage
minio:
  properties:
    url: http://<ip_host>:9000 # указываем url minio
    secretKey: secret # указываем secret key minio
    accessKey: access # указываем access key minio
    buckets:
```

```
    IMPORT: import-api-bucket
    OBJECT_DOCUMENT: local-bucket-object-documents

# урлы для feign клиентов
feign:
  geo-web-cache-client:
    url: http://<ip_host>:8080/geowebcache/rest/
    username: geowebcache
    password: secured

management:
  endpoints:
    enabled-by-default: true
    web:
      base-path: /actuator
  endpoint:
    health:
      show-details: always
```

В папке config добавьте файл logback-spring.xml следующего содержания

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include resource="org/springframework/boot/logging/logback/defaults.xml"/>
  <property name="LOG_FILE" value="${LOG_FILE:-${LOG_PATH:-${LOG_TEMP:-${java.io.tmpdir:-/tmp}}}/app.log}"/>
  <include resource="org/springframework/boot/logging/logback/console-appender.xml"/>
  <include resource="org/springframework/boot/logging/logback/file-appender.xml"/>
  <root level="INFO">
    <appender-ref ref="CONSOLE"/>
  </root>

  <logger name="ru.digital.league" level="TRACE" additivity="false">
    <appender-ref ref="CONSOLE"/>
  </logger>

</configuration>
```

В папке /opt/import_api выполните команду

```
docker-compose up -d
```

3.8 Export-api

3.8.1 Версия и назначение

Текущая версия – 3.25.1804.

Отвечает за выгрузку пространственных данных в различные форматы. Реализован с применением Java, Spring Boot 3, GDAL 3.10

3.8.2 Установка

Разархивируйте архив с образом export-api локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i export-api.tar
docker login <nexus-url>
docker push <nuxus-url>/export-api:release
```

Создайте папку приложения

```
cd /opt
mkdir export_api
cd /export_api
mkdir config
```

Создайте docker-compose.yml в /opt/layer_export_api следующего содержания

В комментариях указаны места, где необходимо подставить реальные урлы и креды инфраструктурных компонентов и сервисов

```
version: "3.3"
services:
  export_api:
    container_name: export_api
    image: <nexus-url>/export_api:release ### указываем url нексуса
    environment:
      - TZ=Europe/Moscow
      - SPRING_CONFIG_NAME=export_api
      - SPRING_PROFILES_ACTIVE=dev
      - SPRING_DATASOURCE_URL=jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host и порт gisdb
      - SPRING_DATASOURCE_USERNAME=gis # указываем username gisdb
      - SPRING_DATASOURCE_PASSWORD=gis # указываем password gisdb
      - SPRING_KAFKA_BOOTSTRAP-SERVERS=<ip_host>:9092 # указываем url и порт для подключения к кафке
      - KAFKA_USERNAME=login # указываем логин кафки
      - KAFKA_PASSWORD=password # указываем пароль кафки
      - GEOSERVER_URL=http://<ip_host>:8080/geoserver # указываем url мастер-геосервера
      - GEOSERVER_NODES="{'http://<ip_host>:8080/geoserver','http://<ip_host>:8080/geoserver'}" # указываем urlы всех геосерверов
      - MINIO_PROPERTIES_URL=http://<ip_host>:9000 # указываем url minio
      - MINIO_PROPERTIES_SECRETKEY=secret # указываем secret key minio
      - MINIO_PROPERTIES_ACCESSKEY=access # указываем access key minio
      - FEIGN_GEO-WEB-CACHE-CLIENT_URL=http://<ip_host>:8080/geowebcache/rest/ # указываем url до реста geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_USERNAME=user # указываем username geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_PASSWORD=pass # указываем password geowebcache
```

```
network_mode: host
volumes:
  - type: bind
    source: ./config
    target: /opt/layer_export_api/config
stdin_open: true
restart: always
```

В папке config создайте файл export_api.yaml следующего содержания.

```
server:
  port: 8086
  servlet:
    context-path: /layer_export_api
spring:
  datasource:
    hikari:
      poolName: HikariPool-Postgres
      idleTimeout: 30000
      maxLifetime: 600000
      minimumIdle: 0
      maximumPoolSize: 5
    driverClassName: org.postgresql.Driver
    url: jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host gisdb
    username: gis # указываем username gisdb
    password: gis # указываем password gisdb
  kafka:
    properties:
      security.protocol: SASL_PLAINTEXT
      sasl.mechanism: PLAIN
      sasl.jaas.config: org.apache.kafka.common.security.plain.PlainLoginModule required
username="${KAFKA_USERNAME}" password="${KAFKA_PASSWORD}";
    consumer:
      group-id: layer-export-group-id
      client-id: layer-export-consumer-1
      auto-offset-reset: earliest
      properties:
        spring:
          json:
            trusted:
```

```
                packages: ru.digital.league.sc.*
              type:
                mapping:
ru.digital.league.sc.map.background.service.out.KafkaValueDto:ru.digital.league.sc.map.layerexport.dto.ExportTaskDto
          bootstrap-servers: <ip_host>:9092 # указываем url и порт для подключения к кафке
        security:
          oauth2:
            resourceserver:
              jwt:
                issuer-uri: localhost
      # логгирование
      logging:
        level:
          ru:
            tectus_it:
              tergis: TRACE
        config: file:./config/logback-spring.xml


      # Keycloak
      keycloak:
        auth-server-url: localhost
        realm: tergis
        login: admin
        password: admin


      # геосервер
      geoserver:
        login: admin
        url: http://<ip_host>:8080/geoserver # указываем url мастер-геосервера
        nodes:      "{'http://<ip_host>:8080/geoserver','http://<ip_host>:8080/geoserver'}"      #
```

указываем urlы всех геосерверов

```
    log_body: true
    password: geoserver
    auth:
      reload:
        url: /eir/invalidate
        filter: Map API Session Key

  # Настройки приложения
  application:
    layers:
      vector:
        workspace: TERGIS
      raster:
        workspace: TERGIS
      external:
        workspace: EXTERNAL
      group:
        workspace: TERGIS
    kafka:
      topic:
        export: export
        export-response: export-response
    security:
      mode: JWT-NO-VERIFY
      role-administration-mode: MOCK
  # swagger
  springdoc:
    swagger-ui:
      path: /swagger-ui-custom.html
```

```
#Найстройка filestorage
minio:
  properties:
    url: http://<ip_host>:9000 # указываем url minio
    secretKey: secret # указываем secret key minio
    accessKey: access # указываем access key minio
    buckets:
      EXPORT: export-api-bucket
      OBJECT_DOCUMENT: bucket-object-documents

# урлы для feign клиентов
feign:
  geo-web-cache-client:
    url: http://<ip_host>:8080/geowebcache/rest/
    username: geowebcache
    password: secured

management:
  endpoints:
    enabled-by-default: true
    web:
      base-path: /actuator
  endpoint:
    health:
      show-details: always
```

В папке config добавьте файл logback-spring.xml следующего содержания

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include resource="org/springframework/boot/logging/logback/defaults.xml"/>
  <property name="LOG_FILE" value="${LOG_FILE:-${LOG_PATH:-${LOG_TEMP:-${java.io.tmpdir:-/tmp}}}/app.log}"/>
  <include resource="org/springframework/boot/logging/logback/console-appender.xml"/>
  <include resource="org/springframework/boot/logging/logback/file-appender.xml"/>
  <root level="INFO">
    <appender-ref ref="CONSOLE"/>
  </root>

  <logger name="ru.digital.league" level="TRACE" additivity="false">
    <appender-ref ref="CONSOLE"/>
  </logger>

</configuration>
```

В папке /opt/layer_export_api выполните команду

```
docker-compose up -d
```

3.9 Admin-api

3.9.1 Версия и назначение

Текущая версия – 3.25.1804.

Набор API для управления каталогом слоев, деревом слоев, наборами карт, системами координат. Реализован с применением Java, Spring Boot 3, GDAL 3.10

3.9.2 Установка

Разархивируйте архив с образом admin-api локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i admin-api.tar
docker login <nexus-url>
docker push <nuxus-url>/admin-api:release
```

Создайте папку приложения

```
cd /opt
mkdir admin_api
cd /admin_api
mkdir config
```

Создайте docker-compose.yml в /opt/admin_api следующего содержания

В комментариях указаны места, где необходимо подставить реальные урлы и креды инфраструктурных компонентов и сервисов

```
version: "3.3"
services:
  admin_api:
    container_name: admin_api
    image: <nexus-url>/admin_api:release ### указываем url нексуса
    environment:
      - TZ=Europe/Moscow
      - SPRING_CONFIG_NAME=admin_api
      - SPRING_PROFILES_ACTIVE=dev
      - SPRING_DATASOURCE_URL=jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host и порт gisdb
      - SPRING_DATASOURCE_USERNAME=gis # указываем username gisdb
      - SPRING_DATASOURCE_PASSWORD=gis # указываем password gisdb
      - SPRING_DATA_REDIS_HOST=<ip_host> # указываем host redis
      - SPRING_DATA_REDIS_PORT=6379 # указываем порт redis
      - SPRING_DATA_REDIS_PASSWORD=password # указываем пароль redis
      - GEOSERVER_URL=http://<ip_host>:8080/geoserver # указываем url мастер-геосервера
      - GEOSERVER_NODES="{'http://<ip_host>:8080/geoserver','http://<ip_host>:8080/geoserver'}" # указываем urlы всех геосерверов
      - MINIO_PROPERTIES_URL=http://<ip_host>:9000 # указываем url minio
      - MINIO_PROPERTIES_SECRETKEY=secret # указываем secret key minio
      - MINIO_PROPERTIES_ACCESSKEY=access # указываем access key minio
      - FEIGN_GEO-WEB-CACHE-CLIENT_URL=http://<ip_host>:8080/geowebcache/rest/ # указываем url до реста geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_USERNAME=user # указываем username geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_PASSWORD=pass # указываем password geowebcache
    network_mode: host
```

```
volumes:
  - type: bind
    source: ./config
    target: /opt/admin_api/config
stdin_open: true
restart: always
```

В папке config создайте файл admin_api.yaml следующего содержания

```
server:
  use-forward-headers: true
  port: 8082
  servlet:
    context-path: /admin_api
spring:
  jpa:
    generate-ddl: false
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
        use_sql_comments: false
        type: trace
        temp:
          use_jdbc_metadata_defaults: false
        show_sql: true
    database: postgresql
    open-in-view: false
  redis:
    host: <ip_host> # указываем host redis
    port: 6379 # указываем порт redis
    password: password # указываем пароль redis
  cache:
    type: redis
    redis:
      time-to-live: 600000
  datasource:
    hikari:
```

```
      poolName: HikariPool-Postgres
      idleTimeout: 30000
      maxLifetime: 600000
      minimumIdle: 0
      maximumPoolSize: 5
    driverClassName: org.postgresql.Driver
    url: jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host gisdb
    username: gis
    password: gis
  servlet:
    multipart:
      max-request-size: 100MB
      max-file-size: 30MB
  data:
    rest:
      detection-strategy: annotated
  security:
    oauth2:
      resourceserver:
        jwt:
          issuer-uri: localhost
# логгирование
logging:
  config: file:./config/logback-spring.xml
  level:
    ru:
      tectus_it:
        tergis: DEBUG
management:
  endpoints:
```

```
      enabled-by-default: true
      web:
        base-path: /actuator
    endpoint:
      health:
        show-details: always
  # Keycloak
  keycloak:
    auth-server-url: localhost
    realm: tergis
    login: admin
    password: admin

  # геосервер
  geoserver:
    login: admin
    url: http://<ip_host>:8080/geoserver # указываем url мастер-геосервера
    nodes:      "{'http://<ip_host>:8080/geoserver','http://<ip_host>:8080/geoserver'}"      # указываем urlы всех геосерверов
    log_body: true
    password: geoserver
    auth:
      reload:
        url: /eir/invalidate
        filter: Map API Session Key
  # swagger
  springdoc:
    swagger-ui:
      path: /swagger-ui-custom.html
```

```
# Настройки приложения
application:
  layers:
    vector:
      workspace: TERGIS
    raster:
      workspace: TERGIS
    external:
      workspace: EXTERNAL
    group:
      workspace: TERGIS
  security:
    mode: JWT-NO-VERIFY
    role-administration-mode: MOCK
# Minio
minio:
  properties:
    url: http://<ip_host>:9000 # указываем url minio
    secretKey: secret # указываем secretkey minio
    accessKey: access  # указываем accesskey minio

# урлы для feign клиентов
feign:
  geo-web-cache-client:
    url: http://<ip_host>:8080/geowebcache/rest/
    username: geowebcache
    password: secured
```

В папке config добавьте файл logback-spring.xml следующего содержания

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include resource="org/springframework/boot/logging/logback/defaults.xml"/>
  <property name="LOG_FILE" value="${LOG_FILE:-${LOG_PATH:-${LOG_TEMP:-${java.io.tmpdir:-/tmp}}}/app.log}"/>
  <include resource="org/springframework/boot/logging/logback/console-appender.xml"/>
  <include resource="org/springframework/boot/logging/logback/file-appender.xml"/>
  <root level="INFO">
    <appender-ref ref="CONSOLE"/>
  </root>

  <logger name="ru.digital.league" level="TRACE" additivity="false">
    <appender-ref ref="CONSOLE"/>
  </logger>

</configuration>
```

В папке /opt/admin_api выполните команду

```
docker-compose up -d
```

3.10 Auth-api

3.10.1 Версия и назначение

Текущая версия – 3.25.1804.

Отдает фиксированный JWT-token. Необходим для работы фронта карты, как технологического компонента. Применяется только на дев стенде для отладки.

3.10.2 Установка

Разархивируйте архив с образом auth-api локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i auth-api.tar
docker login <nexus-url>
docker push <nuxus-url>/auth-api:release
```

Создайте папку приложения

```
cd /opt
mkdir auth_api
cd /auth_api
mkdir config
```

Создайте docker-compose.yml в /opt/auth_api следующего содержания

```
version: "3.3"
services:
  auth_api:
    container_name: auth_api
    image: <nexus-url>/auth_api:release ### указываем url нексуса
    environment:
      - TZ=Europe/Moscow
      - SPRING_CONFIG_NAME=auth_api
      - SPRING_PROFILES_ACTIVE=dev
    network_mode: host
    volumes:
      - type: bind
        source: ./config
        target: /opt/auth_api/config
    stdin_open: true
    restart: always
```

В папке config создайте файл auth_api.yaml следующего содержания

```
server:
  port: 8084
  servlet:
    context-path: /auth_api

logging:
  level:
    ru:
      tectus_it:
        tergis: TRACE
  config: file:./config/logback-spring.xml

keycloak:
  auth-server-url: localhost
  realm: tergis
  resource: auth-api

cipher:
  secret-key: 5bcbf9ecc783cabfa9a24396473473d0

app:
  mode:  MOCK

application:
  security:
    role-administration-mode: MOCK
```

В папке config добавьте файл logback-spring.xml следующего содержания

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include resource="org/springframework/boot/logging/logback/defaults.xml"/>
  <property name="LOG_FILE" value="${LOG_FILE:-${LOG_PATH:-${LOG_TEMP:-${java.io.tmpdir:-/tmp}}}/app.log}"/>
  <include resource="org/springframework/boot/logging/logback/console-appender.xml"/>
  <include resource="org/springframework/boot/logging/logback/file-appender.xml"/>
  <root level="INFO">
    <appender-ref ref="CONSOLE"/>
  </root>

  <logger name="ru.digital.league" level="TRACE" additivity="false">
    <appender-ref ref="CONSOLE"/>
  </logger>

</configuration>
```

В папке /opt/auth_api выполните команду

```
docker-compose up -d
```

3.11 Background-api

3.11.1 Версия и назначение

Текущая версия – 3.25.1804.

Отвечает за запуск / отслеживание выполнения / получение результатов по задачам с длительным выполнением, в частности импорт и экспорт пространственных данных. Реализован с применением Java, Spring Boot 3.

3.11.2 Установка

Разархивируйте архив с образом background-api локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i background_api.tar
docker login <nexus-url>
docker push <nuxus-url>/background_api:release
```

Создайте папку приложения

```
cd /opt
mkdir background_api
cd /background_api
mkdir config
```

Создайте docker-compose.yml в /opt/background_api следующего содержания

В комментариях указаны места, где необходимо подставить реальные урлы и креды инфраструктурных компонентов и сервисов

```
version: "3.3"
services:
  background_api:
    container_name: background_api
    image: <nexus-url>/background_api:release ### указываем url нексуса
    environment:
      - TZ=Europe/Moscow
      - SPRING_CONFIG_NAME=background_api
      - SPRING_PROFILES_ACTIVE=dev
      - SPRING_DATASOURCE_URL=jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host и порт background_api_db
      - SPRING_DATASOURCE_USERNAME=gis # указываем username background_api_db
      - SPRING_DATASOURCE_PASSWORD=gis # указываем password background_api_db
      - SPRING_KAFKA_BOOTSTRAP-SERVERS=<ip_host>:9092 # указываем url и порт для подключения к кафке
      - KAFKA_USERNAME=login # указываем логин кафки
      - KAFKA_PASSWORD=password # указываем пароль кафки
      - MINIO_PROPERTIES_URL=http://<ip_host>:9000 # указываем url minio
      - MINIO_PROPERTIES_SECRETKEY=secret # указываем secret key minio
      - MINIO_PROPERTIES_ACCESSKEY=access # указываем access key minio
    network_mode: host
    volumes:
      - type: bind
        source: ./config
        target: /opt/background_api/config
    stdin_open: true
    restart: always
```

В папке config создайте файл background_api.yaml следующего содержания

```
server:
  port: 8081
  servlet:
    context-path: /background_api
  error:
    include-stacktrace: on_param
    include-message: always
kc:
  base-url: localhost
  realm: tergis
  realm-url: ${kc.base-url}/realms/${kc.realm}

spring:
  servlet:
    multipart:
      max-request-size: 2GB
      max-file-size: 2GB
  datasource:
    url: jdbc:postgresql://<gisdb-host>:<gisdb-port>/background_api_db # указываем host background_api_db
    password: gis
    username: gis
    driver-class-name: org.postgresql.Driver
  liquibase:
    change-log: classpath:db/changelog-master.xml
  jpa:
    hibernate:
      ddl-auto: validate
    show-sql: false
    properties:
```

```
    hibernate:
      format_sql: true
  kafka:
    bootstrap-servers: <ip_host>:9092
    properties:
      security.protocol: SASL_PLAINTEXT
      sasl.mechanism: PLAIN
      sasl.jaas.config: org.apache.kafka.common.security.plain.PlainLoginModule required
username="${KAFKA_USERNAME}" password="${KAFKA_PASSWORD}";
      spring:
        json:
          trusted:
            packages: ru.digital.league.*
          type:
            mapping:
ru.digital.league.sc.map.layerexport.dto.ExportKafkaResponse:ru.digital.league.sc.map.background.dto.ExportKafkaResponse,

ru.digital.league.sc.map.importapi.dto.ImportKafkaResponse:ru.digital.league.sc.map.background.dto.ImportKafkaResponse,

ru.digital.league.sc.map.spatialrelation.dto.SpatialRelationKafkaResponse:ru.digital.league.sc.map.background.dto.SpatialRelationKafkaResponse
    consumer:
      group-id: background-api-group-id
      client-id: background-api-consumer-1
      auto-offset-reset: earliest
      value-deserializer: org.springframework.kafka.support.serializer.JsonSerializer
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    producer:
      value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
      key-serializer: org.apache.kafka.common.serialization.StringSerializer
```

```
        properties:
          max.block.ms: 60000
          request.timeout.ms: 30000
      security:
        oauth2:
          resourceserver:
            jwt:
              issuer-uri: ${kc.realm-url}


    management:
      endpoint:
        health:
          enabled: true
    minio:
      properties:
        url: http://<ip_host>:9000 # указываем url minio
        secretKey: secret # указываем secret key minio
accessKey: secret # указываем access key minio
        buckets:
          IMPORT: import-api-bucket
          EXPORT: export-api-bucket
          SPATIAL_RELATION: spatial-relation-api-bucket


    task:
      producer:
        topics:
          names:
            IMPORT: import
            EXPORT: export
            SPATIAL_RELATION: spatial-relation
```

```
  consumer:
    topics:
      names:
        IMPORT: import-response
        EXPORT: export-response
        SPATIAL_RELATION: spatial-relation-response


application:
  security:
    mode: JWT-NO-VERIFY
logging:
  level:
    ru.digital.league: DEBUG
```

В папке config добавьте файл logback-spring.xml следующего содержания

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include resource="org/springframework/boot/logging/logback/defaults.xml"/>
  <property name="LOG_FILE" value="${LOG_FILE:-${LOG_PATH:-${LOG_TEMP:-${java.io.tmpdir:-/tmp}}}/app.log}"/>
  <include resource="org/springframework/boot/logging/logback/console-appender.xml"/>
  <include resource="org/springframework/boot/logging/logback/file-appender.xml"/>
  <root level="INFO">
    <appender-ref ref="CONSOLE"/>
  </root>

  <logger name="ru.digital.league" level="TRACE" additivity="false">
    <appender-ref ref="CONSOLE"/>
  </logger>

</configuration>
```

В папке /opt/background_api выполните команду

```
docker-compose up -d
```

3.12 Nginx и frontend

3.12.1 Версия и назначение

Текущая версия – 3.25.1804.

Компонент, необходимый для предоставления интерфейса пользователям.

3.12.2 Установка

Разархивируйте архив с образом map_front локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i map_front.tar
docker login <nexus-url>
docker push <nuxus-url>/map_front:release
```

Создайте папку приложения

```
cd /opt
mkdir map_front
cd /map_front
```

Создайте docker-compose.yml в /opt/map_front следующего содержания

```
version: "3"
services:
  map_front:
    image: <nexus-url>/map_front:release ### указываем url нексуса
    ports:
      - "80:80"
    environment:
      - geoserver1=<ip_host>:8080 ### указываем url/service-name сервиса
      - geoserver2=<ip_host>:8080 ### указываем url/service-name сервиса
      - server_map=<ip_host> ### указываем url/service-name сервиса
      - auth_api=<ip_host>:8084 ### указываем url/service-name сервиса
      - map_api=<ip_host>:8081 ### указываем url/service-name сервиса
      - admin_api=<ip_host>:8082 ### указываем url/service-name сервиса
      - background_api=<ip_host>:8081 ### указываем url/service-name сервиса
```

В папке /opt/map_front выполните команду

```
docker-compose up -d
```

3.13 Geowebcache

3.13.1 Версия и назначение

Текущая версия – 1.27.0.
Отвечает за предоставление и генерацию тайлов.

3.13.2 Установка

Разархивируйте архив с образом geowebcache локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i geowebcache.tar
docker login <nexus-url>
docker push <nuxus-url>/geowebcache:release
```

Создайте папку приложения

```
cd /opt
mkdir geowebcache
cd /geowebcache
mkdir config
```

Папка config должна быть подмонтирована и не зависеть от раскатки сервиса, т.к. изменения в конфигурационный файл будут вносится в рантайме.

Создайте docker-compose.yml в /opt/geowebcache следующего содержания

```
version: "3.3"
services:
  geowebcache:
    container_name: geowebcache
    image: <nexus-url>/geowebcache:release ### указываем url нексуса
    environment:
      - TZ=Europe/Moscow
      - JAVA_OPTS=-server -XX:+UseContainerSupport -XX:MaxRAMPercentage=85.0
    network_mode: host
    ports:
      - "8080:8080"
    volumes:
      - type: bind
        source: ./config
        target: /usr/local/tomcat/webapps/config
    stdin_open: true
    restart: always
```

В папке /opt/geowebcache выполните команду

```
docker-compose up -d
```

В папке /opt/geowebcache/config автоматически появится конфигурационный файл при запуске.

Необходимо с помощью rest-api создать s3blobStore с наименованием s3CacheBlobStore.

Пример тела (**Внимание!** Изменить атрибуты awsAccessKey и awsSecretKey на необходимые).

PUT /geowebcache/rest/blobstores/s3CacheBlobStore.xml

```
<S3BlobStore>
    <id>s3CacheBlobStore</id>
    <enabled>true</enabled>
    <bucket>gwc-tiles</bucket>
    <prefix>data</prefix>
    <awsAccessKey>access</awsAccessKey>
    <awsSecretKey>secret</awsSecretKey>
    <maxConnections>50</maxConnections>
    <useHTTPS>false</useHTTPS>
    <endpoint>http://10.122.36.158:9000</endpoint>
    <useGzip>false</useGzip>
    <region>us-east-1</region>
</S3BlobStore>
```

3.14 Spatial-relation-api

3.14.1 Версия и назначение

Текущая версия – 3.25.1804.

Сервис пространственных отношений

3.14.2 Установка

Разархивируйте архив с образом spatial-relation-api, локально командой и добавьте его в nexus. <nexus-url> необходимо заменить на url нексуса

```
docker load -i spatial-relation-api.tar
docker login <nexus-url>
docker push <nuxus-url>/spatial-relation-api:release
```

Создайте папку приложения

```
cd /opt
mkdir spatial-relation-api
cd /spatial-relation-api
```

Создайте docker-compose.yml в /opt/spatial-relation-api следующего содержания

```
version: "3.3"
services:
  spatial_relation_api:
    container_name: spatial_relation_api
    image:  gitlab.akb-it.ru:4567/tergis_3.0/backend/spatial_relation_api:release-2715358-api
    environment:
      - TZ=Europe/Moscow
      - SPRING_PROFILES_ACTIVE=test
      - SERVER_PORT=8090
      - SPRING_DATASOURCE_URL=jdbc:postgresql://<gisdb-host>:<gisdb-port>/gisdb # указываем host и порт gisdb
      - SPRING_DATASOURCE_USERNAME=gis # указываем username gisdb
      - SPRING_DATASOURCE_PASSWORD=gis # указываем password gisdb
      - SPRING_KAFKA_BOOTSTRAP-SERVERS=<ip_host>:9092 # указываем url и порт для подключения к кафке
      - KAFKA_USERNAME=login # указываем логин кафки
      - KAFKA_PASSWORD=password # указываем пароль кафки
      - MINIO_PROPERTIES_URL=http://<ip_host>:9000 # указываем url minio
      - MINIO_PROPERTIES_SECRETKEY=secret # указываем secret minio
      - MINIO_PROPERTIES_ACCESSKEY=access # указываем access minio
      - SPRING_DATA_REDIS_HOST=<ip_host> # указываем хост redis
      - SPRING_DATA_REDIS_PORT=6379 # указываем порт redis
      - SPRING_DATA_REDIS_PASSWORD=pass # указываем пароль redis
      - SPRING_DATA_REDIS_USERNAME=user # указываем логин redis
      - FEIGN_GEO-WEB-CACHE-CLIENT_URL=http://<ip_host>:8080/geowebcache/rest/ # указываем url до реста geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_USERNAME=user # указываем username geowebcache
      - FEIGN_GEO-WEB-CACHE-CLIENT_PASSWORD=pass # указываем password geowebcache
```

```
    network_mode: host
    stdin_open: true
    restart: always
```

В папке /opt/spatial_relation_api выполните команду

```
docker-compose up -d
```

В minio должен быть bucket -

spatial-relation-api-bucket из 1.4

4 Скрипты инициализации

4.1 Скрипты, выполняемые в БД gisdb

Для выполнения скриптов необходимо подключится к БД под пользователем **gis**

Выполнить скрипт:

```
INSERT INTO "map".workset (title, weight) VALUES('Стартовый набор', 0);


INSERT INTO "map".layer_in_workset (layer_id, workset_id, weight, is_visible, is_labeled, transparency) VALUES(1, 1, 20, true, true, 0);
```

4.2 Скрипты создания datastore

Выполнить POST запросы из приложенной коллекции Postman

Create_datastores.postman_collection.json

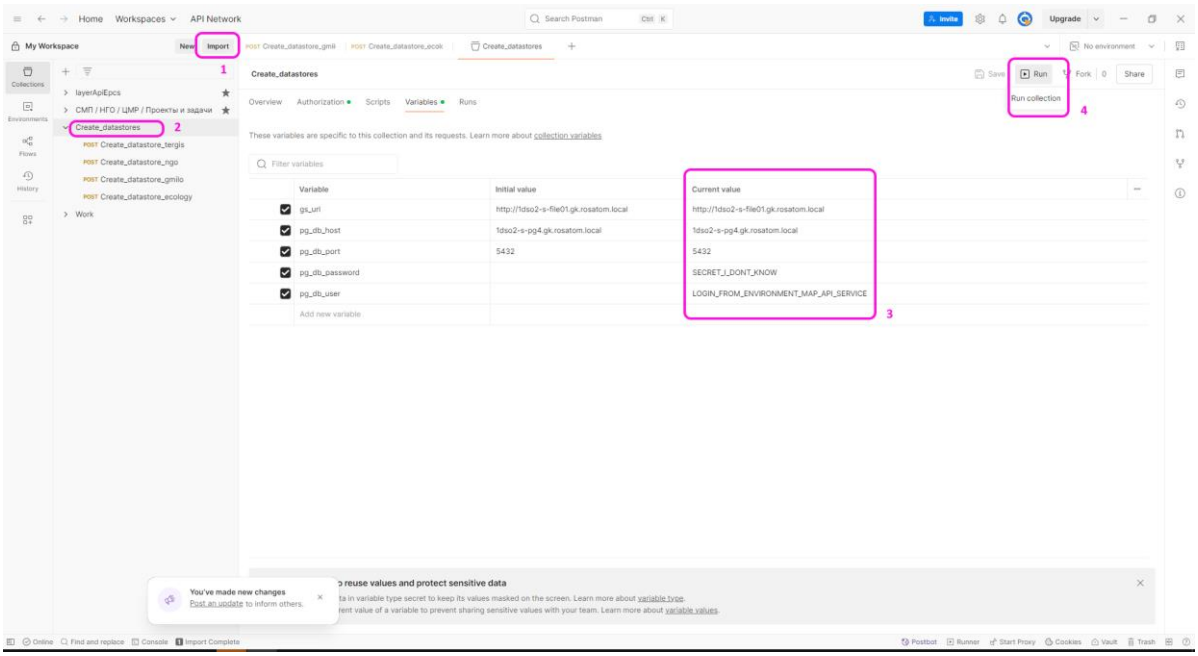
Необходимо:

1. Импортировать коллекцию в Postman из приложенного JSON файла

2. Переключиться на коллекцию и на закладку "Variables"

3. Указать корректные значения переменных в столбце "Current value" для pg_db_password и pg_db_user и **обязательно сохранить** значения переменных коллекции

4. Выполнить "Run collection"



4.3 Проверка работоспособности

Проверка работоспособности производится следующими методами. По итогу проверки всех методов

1. Проверка работоспособности GeoServer:
   - Перейти по ссылке: <geoserver-url>:8080/geoserver/web, где «geoserver-url» - имя сервера;
   - Открывается страница администрирования GeoServer;
   - GeoServer работает, работоспособность подтверждена.

2. Проверка работоспособности map-api
   - Перейти по ссылке: <map-api url>/map_api/actuator/health, где «map-api url» - урл сервиса;
   - Возвращается ответ следующего содержания: {"status":"UP"}
   - map-api работает, работоспособность подтверждена.

3. Проверка работоспособности import-api
   - Перейти по ссылке: <import-api url>/map_api/actuator/health, где «import-api url» - урл сервиса;
   - Возвращается ответ следующего содержания: {"status":"UP"}
   - import-api работает, работоспособность подтверждена.

4. Проверка работоспособности export-api
   - Перейти по ссылке: <export-api url>/map_api/actuator/health, где «export-api url» - урл сервиса;
   - Возвращается ответ следующего содержания: {"status":"UP"}
   - export-api работает, работоспособность подтверждена.

5. Проверка работоспособности admin-api
   - Перейти по ссылке: <admin-api url>/map_api/actuator/health, где «admin-api url» - урл сервиса;
   - Возвращается ответ следующего содержания: {"status":"UP"}
   - admin-api работает, работоспособность подтверждена.

6. Проверка работоспособности background-api

- Перейти по ссылке: <background-api url>/map_api/actuator/health, где «background-api url» - урл сервиса;
- Возвращается ответ следующего содержания: {"status":"UP"}
- background-api работает, работоспособность подтверждена.

7. Проверка работоспособности Geowebcache

- Перейти по ссылке: <geowebcache url>/geowebcache, где «geowebcache url» - урл сервиса;
- Открывается страница администрирования geowebcache;
- Geowebcache работает, работоспособность подтверждена.

8. Проверка работоспособности spatial-relation-api

- Перейти по ссылке: <spatial-relation-api url>/map_api/actuator/health, где «spatial-relation-api url» - урл сервиса;
- Возвращается ответ следующего содержания: {"status":"UP"}
- spatial-relation-api работает, работоспособность подтверждена.

5 Аварийные ситуации

5.1 Действия в случае отказа технических средств

В случае отказа технических средств, необходимо незамедлительно обратиться к специалистам в техническую поддержку и не пытаться самостоятельно проводить диагностику или ремонт, т.к. это может послужить отказом в гарантийном обслуживании технических средств.

5.2 Действия при отказе программного обеспечения

При отказе программного обеспечения рекомендуется необходимо незамедлительно обратиться в техническую поддержку организации-разработчика программного обеспечения.

5.3 Действия в случае несанкционированного вмешательства в данные

В случае обнаружения несанкционированного вмешательства в данные, необходимо незамедлительно обратиться к специалистам службы безопасности.