УТВЕРЖДАЮ
Генеральный директор
ООО «Эквирон»
_____ /Селиверстов М.Н./
«_08_» ___02___ 2021 г.

КОНТЕЙНЕРНАЯ ПЛАТФОРМА «IMAGENARIUM»

Инструментальное средство разработки, развертывания и поддержки программного обеспечения

Руководство программиста

Версия 3.1

Редакция 01

RU 99514566.582914-01 33 01

Лист утверждения

Подп. и дата | Инв. № дубл. | Взам. Инв. № | Подп. и дата | Инв. № подл.

2021

УТВЕРЖДЕН

RU 99514566.582914-01 33 01-ЛУ

КОНТЕЙНЕРНАЯ ПЛАТФОРМА «IMAGENARIUM»

Инструментальное средство разработки, развертывания и поддержки программного обеспечения

Руководство программиста

Версия 3.1

Редакция 01

RU 99514566.582914-01 33 01

Листов 84

2021

АННОТАЦИЯ

Настоящий документ «Контейнерная платформа «Imagenarium». Инструментальное средство разработки, развертывания и поддержки программного обеспечения. Руководство программиста» RU 99514566.582914-01 33 01 предназначен для ознакомления с основными функциями прикладного программного интерфейса инструментального средства разработки программного обеспечения под наименованием «Контейнерная платформа «Imagenarium»» (сокращенное наименование «Imagenarium»). Документ разработан в соответствии с ГОСТ 19.503-79 «Единая система программной документации. Руководство системного программиста».

В настоящем документе приняты следующие обозначения:

1) Элементы экранных форм ввода обозначаются ‹Поле›, ‹Кнопка› или ‹Пункт меню›.

2) Клавиши клавиатуры ПЭВМ обозначаются [Клавиша]. Комбинации одновременно нажимаемых клавиш обозначаются [Клавиша1+Клавиша2].

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Назначение программы

Цель работы инструментального средства разработки программного обеспечения Imagenarium — разработка, развертывание и управление информационными системами на основе контейнерного программного обеспечения.

1.2. Функции программы

Функции инструментального средства разработки программного обеспечения Imagenarium включают в себя:

1) Предоставление консоли управления разработки, развертывания и поддержки контейнеризованного ПО.
2) Обеспечение отладочного окружения разрабатываемого ПО.
3) Обеспечение предпромышленной (максимально приближенной к условиям развертывания у заказчика) среды разработанного ПО.
4) Максимальная автоматизация развертывания разработанного ПО в промышленной среде заказчика.
5) Отработка процессов обновления развернутого в промышленной среде ПО.
6) Выстраивание процессов мониторинга развернутого в промышленной среде ПО.

2. УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

2.1. Требуемые характеристики технических средств

2.1.1. Требуемые характеристики технических средств для эксплуатации серверных компонентов, функционирующих на сервере Imagenarium

Для эксплуатации серверных компонентов, функционирующих на сервере Imagenarium, должны использоваться следующие средства вычислительной техники (СВТ):

1) СВТ коллективного пользования:

– Минимальная (не отказоустойчивая конфигурация) — одиночный сервер под управлением операционной системы Astra Linux 2.12, REDOS 7.6, Debian 11, либо Ubuntu 20.04 LTS и выше, четырёхъядерный ЦП с архитектурой Intel x86/x64, ОЗУ объемом 32Гб, сетевой адаптер с пропускной способностью минимум 100 Мбит/с.

– Стандартная (отказоустойчивая конфигурация) — три сервера под управлением операционной системы Astra Linux 2.12, REDOS 7.6, Debian 11, либо Ubuntu 20.04 LTS и выше, четырёхъядерный ЦП с архитектурой Intel x86/x64, ОЗУ объемом 32Гб, сетевой адаптер с пропускной способностью минимум 1 Гбит/с.

2) СВТ индивидуального пользования:

– При локальном развертывании на АРМ разработчика — ПЭВМ под управлением операционной системы Microsoft Windows 10 с установленными компонентами WSL / WSL2 и системой контейнеризации Docker Desktop, четырёхъядерный ЦП с архитектурой Intel x86/x64, ОЗУ объемом 16Гб, сетевой адаптер с пропускной способностью минимум 100 Мбит/с.

2.1.2. Требуемые характеристики технических средств для эксплуатации консоли управления Imagenarium

Для эксплуатации консоли управления Imagenarium должны использоваться следующие средства вычислительной техники (СВТ):

1) СВТ индивидуального пользования:

– АРМ оператора, представляющее собой ПЭВМ с характеристиками, соответствующими рекомендуемым требованиям операционной системы Microsoft Windows версии 8.1 и выше, и сетевым адаптером, обеспечивающим инфокоммуникационный канал.

2.2. Программное обеспечение, необходимое для функционирования программы

2.2.1. Программное обеспечение, необходимое для функционирования серверных компонентов, функционирующих на сервере Imagenarium

Для серверных компонентов Imagenarium на сервере/виртуальной машине должны быть установлены следующие базовые программы и компоненты:

1) Операционная система – Astra Linux 2.12, REDOS 7.6, Debian 11, либо Ubuntu 20.04 LTS и выше.

2) SSH Server (режим аутентификации по имени и паролю).

3) Пакеты утилит командной строки и общесистемных программ: (bash, ifconfig, sysctl, curl, yum, systemctl, yum-config-manager, unzip).

2.2.2. Программное обеспечение, необходимое для функционирования консоли управления Imagenarium

Для эксплуатации консоли управления Imagenarium необходимы следующие компоненты общего (ОПО) программного обеспечения:

1) Операционная система Microsoft Windows версии 8.1 и выше;

2) Web-браузер:

– Mozilla Firefox версии 83 и выше.

– Google Chrome версии 87 и выше.

– Opera версии 72 и выше.

3. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

3.1. Временные характеристики

ПО Imagenarium предоставляет среду выполнения для развертываемых контейнеризированных приложений, таким образом, при его функционировании должны обеспечиваться временные характеристики, перечень которых отображает Таблица 1.

Таблица 1 — Перечень временных характеристик, которым должно соответствовать ПО Imagenarium

Временные характеристики, которым должно соответствовать ПО Imagenarium	Значение характеристик
Генерация скрипта развертывания контейнеризированного приложения на основе шаблона развертывания	Не более 3 секунд
Запуск процесса развёртывания контейнеризированного приложения	Не более 10 секунд
Свертывание работающего контейнерного приложения	В зависимости от процедуры остановки контейнера, но не дольше 30 секунд
Актуализация информации о развернутых окружениях в консоли управления Imagenarium	Не реже одного раза за 15 секунд
Время восстановления контейнера после сбоя	Не более 60 секунд
Время восстановления после выхода из строя физического сервера кластера	Не более 60 секунд

3.2. Режим работы

Режим функционирования ПО Imagenarium после развертывания контейнеризированных приложений — круглосуточный круглогодичный (24/7/365).

3.3. Средства контроля правильности выполнения и самовосстанавливаемости программы

3.3.1. Средства контроля правильности выполнения

Контроль правильности выполнения ПО Imagenarium осуществляется посредством:

1) Внутренних средств диагностики.

2) Валидации соответствующих программных компонентов.

3.3.2. Средства самовосстанавливаемости

Самовосстанавливаемость ПО Imagenarium осуществляется посредством:

1) Внутренних средств восстановления.

2) Средств автоматического резервного копирования.

3) Средств кластеризации

4) Встроенных инструментов операционной системы.

4. ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1. Введение

Обращение к прикладному программному интерфейсу (API) ПО Imagenarium о осуществляется посредством формирования REST-запросов, содержащих обращения к тем или иным методам из состава API ПО Imagenarium. Структура REST-запросов и формируемых ПО Imagenarium ответов на эти запросы приведена в подразделе 4.2

4.2. Методы из состава API ПО Imagenarium

При формировании REST-запросов к методам, предоставляемым API ПО Imagenarium, необходимо учитывать, что структура URL API Imagenarium имеет следующие параметры:

```
http://<server-name>[:server-port]/api/v3/
```

Сами параметры имеют следующее назначение:

1) `server-name` – имя сервера Imagenarium или его IP-адрес.

2) `server-port` – порт для соединения с сервером Imagenarium.

Для работы с API Imagenarium необходимо пройти авторизацию методом Basic Authtorization (HTTP).

Методы API могут возвращать коды ошибок (HTTP status codes 4xx – 5xx) и сообщения об ошибках при неуспешном выполнении запроса. Общий вид ответа об ошибке в формате JSON приведен ниже.

```
{
  "fieldErrors": [
    {
      "name": "string",
      "value": "string"
    }
  ],
  "globalErrors": [
    "string"
  ],
  "success": false
}
```

где:

`fieldsError` – список входных параметров метода с некорректными значениями;

`globalErrors` – список ошибок, произошедших при выполнении метода;

`success` – признак успешности выполнения метода (при ошибке равен false).

4.2.1. Метод «Получить версию Imagenarium»

Этот метод используется для получения версии ПО Imagenarium.

4.2.1.1. Запрос

Параметры запроса отображает Таблица 2.

Таблица 2 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/version`
Метод	`GET`

Параметры HTTP-заголовка запроса отображает Таблица 3.

Таблица 3 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Accept	Значение: text/html	Строка	Да

Пример JSON запроса отображает Рисунок 1.

```
GET /api/v3/version  HTTP/1.1
Accept: text/html
Host: localhost:8080
```

Пример JSON запроса
Рисунок 1

4.2.1.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 и информацию о версии ПО Imagenarium.

Коды ошибок приведены в подразделе 5.2.

Таблица 4 – Формат ответа на запрос

Поле	Описание	Тип	Обязательность
Без имени	Информация о версии Imagenarium	Строка	Да

Пример JSON ответа отображает Рисунок 2

```
HTTP/1.1 200 OK
Content-Type: text/html;charset=UTF-8

3.1.0.RC1-dd5af4f, 22.12.2020 12:05
```

Пример JSON ответа
Рисунок 2

4.2.2. Метод «Получить список серверов кластера Docker Swarm»

Этот метод используется для получения списка всех серверов кластера Docker Swarm в Imagenarium.

4.2.2.1. Запрос

Параметры запроса отображает Таблица 5.

Таблица 5 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/nodes`
Метод	`GET`

Параметры HTTP-заголовка запроса отображает Таблица 6.

Таблица 6 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Accept	Значение: application/json	Строка	Да

Пример JSON запроса отображает Рисунок 3.

```
GET /api/v3/nodes  HTTP/1.1
Accept: application/json
Host: localhost:8080
```

Пример JSON запроса
Рисунок 3

4.2.2.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 и информацию обо всех серверах кластера Docker Swarm в Imagenarium. Структуру ответа на запрос отображает Таблица 7.

Коды ошибок приведены в подразделе 5.2.

Таблица 7 – Формат ответа на запрос

Поле	Описание	Тип	Обязательность
id	Идентификатор сервера	Строка	Да
hostname	Наименование сервер	Строка	Да
ip	Ip – адрес	Строка	Да
status	Текущий статус сервера	Строка	Да
availability	Доступность	Строка	Да
role	Роль (справочное значение)	Строка	Да
dockerVersion	Версия docker	Строка	Да
totalMemory	Общий объем ОЗУ сервера	Строка	Да
labels	Метки привязки сервисов к месту развертывания	Объект «Labels», структуру объекта отображает Таблица 8	Да
sshPort	ssh порт	Строка	Нет
sshUser	Пользователь ssh	Строка	Нет
externalUrl	Внешний Url	Строка	Нет
dc		Строка	Да

Таблица 8 – Структура объекта «Labels»

Поле	Описание	Тип	Обязательность

[имя_метки]	Пара ключ/ значение.
	Пример:
	`{`
	`"grafana": "true",`
	`"clickhouse": "true",`
	`"couchdb": "true",`
	`"etcd": "1",`
	`"pmm-server": "true"`
	`}`
		Строка	Да

Пример JSON ответа отображает Рисунок 4.

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
[
  {
    "id": "ynno145i93uzdug5fhfw3xm88",
    "hostname": "eqvm1",
    "ip": "192.168.1.9",
    "status": "ready",
    "availability": "active",
    "role": "manager",
    "dockerVersion": "19.03.8",
    "totalMemory": "15.67",
    "labels": {
      "postgres": "true",
      "grafana": "true",
      "clickhouse": "true",
      "ocserv": "true",
      "couchdb": "true",
      "etcd": "1",
      "pmm-server": "true"
    },
    "dc": "dc1"
  }
]
```

Пример JSON ответа
Рисунок 4

4.2.3. Метод «Обновить информацию о сервере кластера Docker Swarm»

Этот метод используется для отправки обновлённых параметров сервера кластера Docker Swarm.

4.2.3.1. Запрос

Параметры запроса отображает Таблица 9.

Таблица 9 – Параметры

Параметр	Описание
URL	http://<server>[:port]/api/v3/update/{id}
Параметры строки запроса	id={id }
Метод	POST

Параметры HTTP-заголовка запроса отображает Таблица 10.

Таблица 10 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 11.

Таблица 11 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
id	Идентификатор сервера	Строка	Да

Описание структуры объекта «UpdateNode» запроса на обновление параметров сервера кластера Docker Swarm отображает Таблица 12.

Таблица 12 – Структура объекта «UpdateNode»

Поле	Описание	Тип	Обязательность
hostname	Наименование сервера кластера Docker Swarm	Строка	Да
role	Роль	Строка	Да
availability	Доступность	Строка	Да
externalUrl	Внешний Url	Строка	Нет
labels	Метки привязки сервисов к месту развертывания	Массив объектов «Labels», структуру объекта отображает Таблица 13	Да
sshPort	ssh порт	Строка	Нет
sshUser	Пользователь ssh	Строка	Нет

Таблица 13 – Структура объекта «Labels»

Поле	Описание	Тип	Обязательность

[имя_метки]	Пара ключ/ значение.
	Пример:
	{
	"additionalProp1": "value1",
	"additionalProp2": "value2"
	}
		Строка	Нет

Пример JSON запроса отображает Рисунок 5.

```
POST /api/v3/nodes/7i5xs7zqllyx88gsfmas4idb6 HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: localhost:8080

{
    "id": "ynno145i93uzdug5fhfw3xm88",
    "hostname": "eqvm1",
    "ip": "192.168.1.9",
    "status": "ready",
    "availability": "active",
    "role": "manager",
    "externalUrl": "192.192.1.1",
    "sshPort": "2222",
    "sshUser": "admin",
    "labels": {
      "jenkins": "true",
      "prometheus": "true",
      "kibana": "true",
      "postgres": "true",
      "prometheus-node-exporter": "true",
      "grafana": "true",
      "clickhouse": "true",
      "couchdb": "true",
      "pmm-server": "true"
    }
}
```

Пример JSON запроса
Рисунок 5

4.2.3.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 6.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 6

4.2.4. Метод «Получить настройки конфигурации кластера»

Этот метод используется для получения настроек конфигурации кластера.

4.2.4.1. Запрос

Параметры запроса отображает Таблица 14.

Таблица 14 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/config`
Метод	`GET`

Параметры HTTP-заголовка запроса отображает Таблица 15.

Таблица 15 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
accept	значение: application/json	Строка	Да

Пример JSON запроса отображает Рисунок 7.

```
GET /api/v3/config  HTTP/1.1
Accept: application/json
Host: localhost:8080
```

Пример JSON запроса
Рисунок 7

4.2.4.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 и информацию о настройках конфигурации кластера. Структуру ответа на запрос отображает Таблица 16.

Коды ошибок приведены в подразделе 5.2.

Таблица 16 – Формат ответа на запрос

Поле	Описание	Тип	Обязательность
consoleName	Наименование консоли	Строка	Да
consoleColor	Цвет консоли (в формате HEX)	Строка	Да
license	Лицензия	Строка	Да
tags	Тэги	Массив строк «Tags»	Да

Пример JSON ответа отображает Рисунок 8.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "consoleName": "Test Server",
  "consoleColor": "#2d5680",
  "license": "",
  "tags": []
}
```

Пример JSON ответа
Рисунок 8

4.2.5. Метод «Сохранить настройки конфигурации кластера»

Этот метод используется для отправки и сохранения изменённых настроек конфигурации кластера.

4.2.5.1. Запрос

Параметры запроса отображает Таблица 17.

Таблица 17 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/config`
Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 18.

Таблица 18 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры тела запроса отображает Таблица 19.

Таблица 19 – Параметры тела запроса

Параметр	Описание	Тип	Обязательность
consoleName	Наименование консоли кластера	Строка	Нет
consoleColor	Цвет консоли (в формате HEX)	Строка	Нет
license	Лицензия	Строка	Нет
tags	Тэги	Массив строк «Tags»	Нет

Пример JSON запроса отображает Рисунок 9.

```
POST /api/v3/config HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: localhost:8080

{
  "consoleName": "PROD",
  "consoleColor": "#ff5555",
  "license": "",
  "tags": []
}
```

Пример JSON запроса
Рисунок 9

4.2.5.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответаПример JSON ответа отображает Рисунок 10.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 10

4.2.6. Метод «Получить список реестров docker-образов»

Этот метод используется для получения списка реестров docker-образов ПО Imagenarium.

4.2.6.1. Запрос

Параметры запроса отображает Таблица 20.

Таблица 20 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/registries`
Метод	`GET`

Параметры HTTP-заголовка запроса отображает Таблица 21.

Таблица 21 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Accept	значение: application/json	Строка	Да

Пример JSON запроса отображает Рисунок 11.

```
GET /api/v3/registries HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 11

4.2.6.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 и информацию о реестрах docker-образов ПО Imagenarium.

Формат JSON ответа на запрос отображает Таблица 22.

Коды ошибок приведены в подразделе 5.2.

Таблица 22 – Формат ответа на запрос

Поле	Описание	Тип	Обязательность
id	Идентификатор реестра	Строка	Да
name	Наименование реестра (может содержать цифры и буквы латинского алфавита)	Строка	Да
url	Url адрес реестра	Строка	Да
username	Логин	Строка	Да
password	пароль	Строка	Да

Пример JSON ответа отображает Рисунок 12.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

[
  {
    "id": "1586967023373",
    "name": "MyReg",
    "url": "registry.gitlab.com",
    "username": "my_regitry",
    "password": "YUYuV123"
  },
  {
    "id": "1607333682202",
    "name": "test",
    "url": "registry.gitlab.com/public/tmp",
    "username": "tester",
    "password": "o31iNG"
  }
]
```

Пример JSON ответа
Рисунок 12

4.2.7. Метод «Добавить реестр»

Этот метод используется для добавления реестра в ПО Imagenarium.

4.2.7.1. Запрос

Параметры запроса отображает Таблица 23.

Таблица 23 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/registries`
Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 24.

Таблица 24 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры тела запроса отображает Таблица 25.

Таблица 25 – Параметры тела запроса

Параметр	Описание	Тип	Обязательность
id	Идентификатор реестра	Строка	Да
name	Наименование реестра (может содержать цифры и буквы латинского алфавита)	Строка	Да
url	Url адрес рееста	Строка	Да
username	логин	Строка	Да
password	пароль	Строка	Да

Пример JSON запроса отображает Рисунок 13.

```
POST /api/v3/registries HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: localhost:8080
  {
    "id": "123456",
    "name": "dockerhub",
    "url": "https://index.docker.io/v1/",
    "username": "token",
    "password": "123"
  }
```

Пример JSON запроса
Рисунок 13

4.2.7.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 14.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 14

4.2.8. Метод «Удалить реестр»

Этот метод используется для удаления реестра в Imagenarium.

4.2.8.1. Запрос

Параметры запроса отображает Таблица 26.

Таблица 26 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/registries/{id}`
Параметры строки запроса	`id={id}`
Метод	`DELETE`

Параметры HTTP-заголовка запроса отображает Таблица 27.

Таблица 27 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 28.

Таблица 28 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
id	Идентификатор реестра	Строка	Да

Пример JSON запроса отображает Рисунок 15.

```
DELETE /api/v3/registries/11 HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 15

4.2.8.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 16.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 16

4.2.9. Метод «Получить список репозиториев»

Этот метод используется для получения списка существующих репозиториев в ПО Imagenarium.

4.2.9.1. Запрос

Параметры запроса отображает Таблица 29.

Таблица 29 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/repositories`
Метод	`GET`

Параметры HTTP-заголовка запроса отображает Таблица 30.

Таблица 30 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Accept	значение: application/json	Строка	Да

Пример JSON запроса отображает Рисунок 17.

```
GET /api/v3/repositories HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 17

4.2.9.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 и информацию о репозиториях ПО Imagenarium. Формат ответа на запрос отображает Таблица 31.

Коды ошибок приведены в подразделе 5.2.

Таблица 31 – Формат ответа на запрос

Параметр	Описание	Тип	Обязательность
id	Идентификатор репозитория	Строка	Да
name	Наименование репозитория (может содержать цифры и буквы латинского алфавита)	Строка	Да
url	Url адрес репозитория	Строка	Да
username	Логин	Строка	Да
password	Пароль	Строка	Да
offline	Возможность работы без сети	Логический	Да
lastCommitInfo	Массив объектов, содержащий информацию о последнем коммите	Массив объектов «CommitInfo» (см. Таблицу 36)	Да
needRefresh	Необходимость в обновлении	Логический	Да
canonicalUrl	Канонический Url	Строка	Да

Описание структуры объекта «CommitInfo» отображает Таблица 32.

Таблица 32 – Структура объекта «CommitInfo»

Поле	Описание	Тип	Обязательность
id	Идентификатор коммита	Строка	Да
committer	Пользователь, осуществившей коммит	Строка	Да
timestamp	Время создания коммита	Число (int32)	Да
message	Сообщение	Строка	Да
branch	Ветка	Строка	Да
tag	Тэг	Строка	Да

Пример JSON ответа отображает Рисунок 18.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

[
  {
    "id": "0",
    "name": "community",
    "url": "https://gitlab.com/imagenarium/oss.git",
    "offline": false,
    "needRefresh": false
  },
  {
    "id": "1587996631102",
    "name": "Experimental",
    "url": "https://gitlab.com/abc/experimental.git",
    "username": "user1",
    "password": "pass1",
    "offline": false,
    "needRefresh": false
  }
]
```

Пример JSON ответа
Рисунок 18

4.2.10. Метод «Добавить репозиторий»

Этот метод используется для добавления нового репозитория в Imagenarium.

4.2.10.1. Запрос

Параметры запроса отображает Таблица 33.

Таблица 33 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/repositories`
Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 34.

Таблица 34 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры тела запроса отображает Таблица 35.

Таблица 35 – Параметры тела запроса

Параметр	Описание	Тип	Обязательность
id	Идентификатор репозитория	Строка	Да
name	Наименование репозитория (может содержать цифры и буквы латинского алфавита)	Строка	Да
url	Url адрес репозитория	Строка	Да
username	Логин	Строка	Нет
password	Пароль	Строка	Нет
offline	Возможность работы без сети	Логический	Нет
lastCommitInfo	Массив объектов, содержащий информацию о последнем коммите	Массив объектов «CommitInfo», структуру объекта отображает Таблица 36	Нет
needRefresh	Необходимость в обновлении	Логический	Нет
canonicalUrl	Канонический Url	Строка	Нет

Таблица 36 – Структура объекта «CommitInfo»

Поле	Описание	Тип	Обязательность
id	Идентификатор коммита	Строка	Да
committer	Пользователь, создавший коммит	Строка	Да
timestamp	Метка времени создания коммита	Число (int32)	Да
message	Сообщение	Строка	Да
branch	Ветка	Строка	Да
tag	Тэг	Строка	Да

Пример JSON запроса отображает Рисунок 19.

```
POST /api/v3/repositories HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080

{
    "id": "1587996631102",
    "name": "Experimental",
    "url": "https://gitlab.com/abc/experimental.git",
    "username": "user1",
    "password": "pass1",
    "offline": false,
  }
,
  "lastCommitInfo": {
    "id": "1587996631111",
    "committer": "User 1",
    "timestamp": 1587996631555,
    "message": "Minor bug fixed",
    "branch": "dev",
    "tag": "STABLE_RELEASE_3.1.2.4"
  },
  "needRefresh": true,
  "canonicalUrl": " https://gitlab.com/abc/experimental.git"
}
```

Пример JSON запроса
Рисунок 19

4.2.10.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 20.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 20

4.2.11. Метод «Удалить репозиторий»

Этот метод используется для удаления репозитория из списка доступных репозиториев в ПО Imagenarium.

4.2.11.1. Запрос

Параметры запроса приведены ниже.

Таблица 37 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/repositories/{id}`
Параметры строки запроса	`id={id}`
Метод	`DELETE`

Параметры HTTP-заголовка запроса отображает Таблица 38.

Таблица 38 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 39.

Таблица 39 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
id	Идентификатор репозитория	Строка	Да

Пример JSON запроса отображает Рисунок 21.

```
DELETE /api/v3/repositories/1593972833225 HTTP/1.1
Content-Type: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 21

4.2.11.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 22.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 22

4.2.12. Метод «Развернуть пакет docker-образов в окружении»

Этот метод используется для развертывания пакета docker-образов в окружении ПО Imagenarium.

4.2.12.1. Запрос

Параметры запроса отображает Таблица 40.

Таблица 40 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/deploy/stack/{namespace}/{stackId}/{version}`

Параметры строки запроса	`namespace = {namespace}&`
	`stackId = {stackId}&`
	`version = {version}&`
	`repository = {repository}&`
	`gitRef = {gitRef}`

Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 41.

Таблица 41 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 42.

Таблица 42 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
namespace	Наименование окружения	Строка	Да
stackId	Идентификатор пакета docker образа	Строка	Да
version	Версия пакета docker образа	Строка	Да
repository	Репозиторий GIT	Строка	Да
gitRef	Ссылка на ветку или тэг	Строка	Да

Параметры тела запроса отображает Таблица 43.

Таблица 43 – Параметры тела запроса

Параметр	Описание	Тип	Обязательность

additionalProp	Конфигурационные параметры для шаблонов развертывания
	Пара ключ/ значение.
	Пример:
	{
	"additionalProp1": "value1",
	"additionalProp2": "value2"
	}
		Строка	Нет

Пример JSON запроса отображает Рисунок 23.

```
POST /api/v3/deploy/stack/test/postgres/12?repository=
https%3A%2F%2Fgitlab.com%2Fimagenarium%2Foss.git &gitRef= 3.1.0  HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080

{
        "DELETE_DATA": "true",
        "ADMIN_MODE": "false",
        "PUBLISHED_PORT": "",
        "POSTGRES_USER": "user",
        "POSTGRES_PASSWORD": "passwd",
        "POSTGRES_DB": "postgres",
        "APP_USER": "appuser",
        "APP_PASSWORD": "appuser",
        "PMM_ENABLE": "false",
        "CMD": "-c max_connections=2000 -c effective_io_concurrency=10 -c shared_buffers=128MB"
}
```

Пример JSON запроса
Рисунок 23

4.2.12.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 24.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 24

4.2.13. Метод «Удалить пакет docker-образов из окружения»

Этот метод используется для удаления пакета docker-образов из окружения в ПО Imagenarium.

4.2.13.1. Запрос

Параметры запроса отображает Таблица 44.

Таблица 44 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/undeploy/stack/{namespace}/{stackId}/{version}`

Параметры строки запроса	`namespace = {namespace}&`
	`stackId = {stackId}&`
	`version = {Version}`

Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 45.

Таблица 45 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 46.

Таблица 46 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
namespace	Наименование окружения	Строка	Да
stackId	Идентификатор пакета docker образа	Строка	Да
version	Версия пакета docker образа	Строка	Да

Пример JSON запроса отображает Рисунок 25.

```
POST /api/v3/undeploy/stack/test/clickhouse/12  HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 25

4.2.13.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 и информацию об удаляемом компоненте. Формат ответа на запрос отображает Таблица 47.

Коды ошибок приведены в подразделе 5.2.

Таблица 47 – Формат ответа на запрос

Параметр	Описание	Тип	Обязательность
stackId	Идентификатор пакета docker образа	Строка	Да
version	Версия пакета docker образа	Строка	Да
namespace	Наименование окружения	Строка	Да
order	Номер пакета docker образа в окружении	Число (int32)	Да
undeploymentTemplate		Строка	Да
params	Массив параметров пакета docker образа	Массив объектов «Params», структуру объекта отображает Таблица 48	Да
repo	Репозиторий	Строка	Да
branch	Ветка	Строка	Да
commit	Последний коммит	Строка	Да
tag	Тэг	Строка	Да
timestamp	Временная отметка	Число (int32)	Да
status	Статус пакета docker образа (справочное значение)	Строка	Да
components	Массив объектов, содержащий информацию о компонентах пакета docker образа	Массив объектов «Components», структуру объекта отображает Таблица 49	Да
globalParams	Глобальные параметры	Массив объектов «globalParams», структуру объекта отображает Таблица 48	Да
fullStackId	Полный идентификатор пакета docker образа	Строка	Да

Таблица 48 – Структура объектов «Params» и «globalParams»

Поле	Описание	Тип	Обязательность

additionalProp1	Конфигурационные параметры шаблонов развертывания
	Пара ключ/ значение.
	Пример:
	{
	"additionalProp1": "value1",
	"additionalProp2": "value2"
	}
		Строка	Да

Таблица 49 – Структура объекта «Components»

Поле	Описание	Тип	Обязательность
name	Наименование компонента	Строка	Да
image	Образ компонента	Строка	Да
params	Параметры компонента	Строка	Да
repo	Репозиторий	Строка	Да
branch	Ветка	Строка	Да
commit	Последний коммит	Строка	Да
tag	Тэг	Строка	Да
timestamp	Временная отметка	Число (int32)	Да
id	Идентификатор компонента	Строка	Да
created	Дата создания компонента	Число (int64)	Да
stackStatus	Статус компонента (справочное значение)	Строка	Да
environmentVariables	Массив, содержащий информацию о переменных среды	Массив объектов «Variables», структуру объекта отображает, структуру объекта отображает Таблица 50	Да
labels	Массив меток, которые привязывают сервис к месту развёртывания	Массив объектов «Labels», структуру объекта отображает Таблица 50	Да
volumes	Массив	Массив объектов «Volumes», структуру объекта отображает Таблица 51	Да
binds	Массив	Массив объектов «Binds», структуру	Да

Поле	Описание	Тип	Обязательность
		объекта отображает Таблица 53	
networks	Массив, содержащий информацию о настройках сети	Массив объектов «Networks», структуру объекта отображает Таблица 54	Да
publishedPorts	Массив, содержащий информацию о внешних портах	Массив объектов «Ports», структуру объекта отображает Таблица 55	Да
adminMode	Режим администратора	Логический (boolean)	Да
runApp	Запустить приложение	Логический (boolean)	Да
showAdminMode	Просматривать в режиме администратора	Логический (boolean)	Да

Таблица 50 – Структура объектов «Variables» и «Labels»

Поле	Описание	Тип	Обязательность

additionalProp1	Пара ключ/ значение.
	Пример:
	{
	"additionalProp1": "value1",
	"additionalProp2": "value2"
	}
		Строка	Да

Таблица 51 – Структура объекта «Volumes»

Поле	Описание	Тип	Обязательность
source		Строка	Да
target		Строка	Да
driver		Строка	Да
options	Массив опций	Массив объектов «Options», структуру объекта отображает Таблица 52	Да
status	Массив статусов	Массив объектов «Status», структуру объекта отображает Таблица 52	Да

Таблица 52 – Структура объектов «Options» и «Status»

Поле	Описание	Тип	Обязательность

additionalProp1	Пара ключ/ значение.
	Пример:
	{
	"additionalProp1": "value1",
	"additionalProp2": "value2"
	}
		Строка	Да

Таблица 53 – Структура объекта «Binds»

Поле	Описание	Тип	Обязательность
source	Исходная точка монтирования	Строка	Да
target	Целевая точка монтирования	Строка	Да

Таблица 54 – Структура объекта «Networks»

Поле	Описание	Тип	Обязательность
name	Имя сети	Строка	Да
driver	Сетевой драйвер	Строка	Да
subnet	Подсеть	Строка	Да

Таблица 55 – Структура объекта «Ports»

Поле	Описание	Тип	Обязательность
publishedPort	Внешний порт	Строка	Да
targetPort	Целевой порт	Строка	Да
publishMode	Режим публикации	Строка	Да
protocol	Протокол	Строка	Да

Пример JSON ответа отображает Рисунок 26.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "stackId": "swarmview",
  "version": "latest",
  "namespace": "imgtest",
  "order": 2,
  "undeploymentTemplate": "",
  "params": {
    "ADMIN_MODE": "false",
    "JAVA_OPTS": "-Xmx256m -Xms256m",
    "namespace": "imgtest"
  },
  "repo": "Experimental",
  "branch": "master",
  "commit": "d1589d0",
  "timestamp": 1612368147,
  "status": "DEPLOYED",
  "components": [
    {
      "@c": ".DeployedService",
      "name": "swarmview-imgtest",
      "image": "imagenarium/swarmview:3.1.0.RC1",
      "params": "",
      "mode": "replicated",
      "repo": "Experimental",
      "branch": "master",
      "commit": "d1589d0",
      "timestamp": 1612368147,
      "created": 0,
      "environmentVariables": [
        {
          "JAVA_OPTS": "-Xmx256m -Xms256m"
        }
      ],
      "labels": [],
      "volumes": [],
      "binds": [],
      "networks": [
        {
          "name": "net-imgtest"
        }
      ],
      "publishedPorts": [],
      "adminMode": false,
      "runApp": false,
      "showAdminMode": false,
      "desiredReplicas": 0,
      "runningReplicas": 0,
      "resolutionMode": "dnsrr",
      "updated": 0,
      "constraints": [],
      "mutexes": [],
      "tasks": [],
      "scalable": false,
      "off": false
    }
  ],
  "globalParams": {}
}
```

Пример JSON ответа
Рисунок 26

4.2.14. Метод «Развернуть группу пакетов docker образов в окружении»

Этот метод используется для развертывания группы пакетов docker образов в окружении ПО Imagenarium.

4.2.14.1. Запрос

Параметры запроса отображает Таблица 56.

Таблица 56 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/deploy/group/{namespace}/{groupId}/{version}`

Параметры строки запроса	`namespace = {namespace}&`
	`groupId = {groupId}&`
	`version = {version}&`
	`repository = {repository}&`
	`gitRef = {gitRef}`

Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 57.

Таблица 57 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 58.

Таблица 58 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
namespace	Наименование окружения	Строка	Да
groupId	Идентификатор группы пакетов docker образов	Строка	Да
version	Версия группы пакетов docker образов	Строка	Да
repository	Репозиторий GIT	Строка	Да
gitRef	Ссылка на ветку или тэг	Строка	Да

Параметры тела запроса отображает Таблица 59.

Таблица 59 – Параметры тела запроса

Параметр	Описание	Тип	Обязательность

additionalProp	Конфигурационные параметры для шаблонов развертывания
	Пара ключ/ значение.
	Пример:
	"additionalProp1": {
	"additionalProp1": "value1",
	"additionalProp2": "value2"
	}
		Строка	Да

Пример JSON запроса отображает Рисунок 27.

```
POST
/api/v3/deploy/group/test/swarmview/latest?repository=https%3A%2F%2Fgitlab.com%2Fequiron%2Fexperimental
-marketplace.git&gitRef=master  HTTP/1.1
Accept: application/json;charset=UTF-8

{
	"logstash": {
		"LOGSTASH_PUBLISHED_PORT": "14560",
		"LS_JAVA_OPTS": "-Xms256m -Xmx256m -Dnetworkaddress.cache.ttl=10",
		"ES_URL": "",
		"ES_PASSWORD": "",
		"ROLL_OVER_SIZE_GB": "25",
		"ROLL_OVER_DAYS": "1",
		"DELETE_DAYS": "30",
		"NUMBER_OF_REPLICAS": "0",
		"NUMBER_OF_SHARDS": "1",
		"INDEX_NAME": "swarmview"
	},
	"swarmview": {
		"JAVA_OPTS": "-Xmx256m -Xms256m",
	}
}
```

Пример JSON запроса
Рисунок 27

4.2.14.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 28.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 28

4.2.15. Метод «Удалить окружение»

Этот метод используется для удаления окружения в ПО Imagenarium.

4.2.15.1. Запрос

Параметры запроса отображает Таблица 60.

Таблица 60 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/undeploy/namespace/{namespace}`
Параметры строки запроса	`namespace = {namespace}`
Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 61.

Таблица 61 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 62.

Таблица 62 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
namespace	Наименование окружения	Строка	Да

Пример JSON запроса отображает Рисунок 29.

```
POST /api/v3/undeploy/namespace/test HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 29

4.2.15.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 30.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 30

4.2.16. Метод «Обновить образ компонента»

Этот метод используется для обновления образа компонента ПО Imagenarium.

4.2.16.1. Запрос

Параметры запроса отображает Таблица 63.

Таблица 63 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/component/update/{name}`
Параметры строки запроса	`name = {name}&` `image = {image}`
Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 64

Таблица 64 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 65.

Таблица 65 — Параметры строки запроса

Параметр	Описание	Тип	Обязательность
name	Имя компонента	Строка	Да
image	Образ компонента	Строка	Да

Пример JSON запроса отображает Рисунок 31.

```
POST /api/v3/component/update/swarmview-imgtest?image=imagenarium%2Fswarmview%3A3.1.0.RC1 HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 31

4.2.16.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 32.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 32

4.2.17. Метод «Масштабировать реплицированный компонент»

Этот метод используется для масштабирования реплицированного компонента ПО Imagenarium.

4.2.17.1. Запрос

Параметры запроса отображает Таблица 66.

Таблица 66 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/component/scale/{name}`
Параметры строки запроса	`name = {name}&` `replicas = {replicas}`
Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 67.

Таблица 67 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 68.

Таблица 68 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
name	Имя компонента	Строка	Да
replicas	Реплики компонента	Число (int32)	Да

Пример JSON запроса отображает Рисунок 33.

```
POST /api/v3/component/scale/swarmview-imgtest?replicas=0 HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 33

4.2.17.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 34.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 34

4.2.18. Метод «Остановить компонент»

Этот метод используется для остановки компонента ПО Imagenarium.

4.2.18.1. Запрос

Параметры запроса отображает Таблица 69.

Таблица 69 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/component/stop/{name}`
Параметры строки запроса	`name = {name}`
Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 70.

Таблица 70 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 71.

Таблица 71 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
name	Имя компонента	Строка	Да

Пример JSON запроса отображает Рисунок 35.

```
POST /api/v3/component/stop/swarmview-imgtest HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 35

4.2.18.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 36.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 36

4.2.19. Метод «Запустить компонент»

Этот метод используется для запуска компонента ПО Imagenarium.

4.2.19.1. Запрос

Параметры запроса отображает Таблица 72.

Таблица 72 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/component/start/{name}`
Параметры строки запроса	`name = {name}`
Метод	`POST`

Параметры HTTP-заголовка запроса отображает Таблица 73.

Таблица 73 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Параметры строки запроса отображает Таблица 74.

Таблица 74 – Параметры строки запроса

Параметр	Описание	Тип	Обязательность
name	Имя компонента	Строка	Да

Пример JSON запроса отображает Рисунок 37.

```
POST /api/v3/component/start/swarmview-imgtest HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 37

4.2.19.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 с пустым телом ответа. Пример JSON ответа отображает Рисунок 38.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

Пример JSON ответа
Рисунок 38

4.2.20. Метод «Получить список развёрнутых компонентов»

Этот метод используется для получения списка развёрнутых компонентов в ПО Imagenarium.

4.2.20.1. Запрос

Параметры запроса отображает Таблица 75.

Таблица 75 – Параметры

Параметр	Описание
URL	`http://<server>[:port]/api/v3/components`
Метод	`GET`

Параметры HTTP-заголовка запроса отображает Таблица 76.

Таблица 76 – Параметры HTTP-заголовка

Параметр	Описание	Тип	Обязательность
Content-Type	Типы данных, которые могут быть переданы посредством сети интернет с применением стандарта MIME Допустимое значение: application/json	Строка	Да

Пример JSON запроса отображает Рисунок 39.

```
POST /api/v3/components HTTP/1.1
Accept: application/json;charset=UTF-8
Host: localhost:8080
```

Пример JSON запроса
Рисунок 39

4.2.20.2. Ответ на запрос

При успешном выполнении запроса сервер возвращает HTTP код 200 и информацию обо всех развёрнутых компонентах. Формат ответа на запрос отображает Таблица 77.

Коды ошибок приведены в подразделе 5.2.

Таблица 77 – Формат ответа на запрос

Параметр	Описание	Тип	Обязательность
namespace	Наименование окружения	Строка	Да
stacks	Массив объектов, содержащий информацию о пакетах docker образов в указанном окружении	Массив объектов «Stacks», структуру объекта отображает Таблица 79	Да
locked	Статус заблокирован или нет	Логический (boolean)	Да
summaries	Массив общих данных	Массив объектов «Summaries», структуру объекта отображает Таблица 78	Да

Таблица 78 – Структура объекта «Summaries»

Поле	Описание	Тип	Обязательность
title	Наименование	Строка	Да
host	Хост	Строка	Да
ports	Массив строк, содержащий значения портов	Массив строк	Да

Таблица 79 – Структура объекта «Stacks»

Поле	Описание	Тип	Обязательность
stackId	Идентификатор пакета docker образа	Строка	Да
version	Версия пакета docker образа	Строка	Да
namespace	Наименование окружение	Строка	Да
order	Номер пакета docker образа в окружении	Число (int32)	Да
undeploymentTemplate	Шаблон свертывания docker образа	Строка	Да

Поле	Описание	Тип	Обязательность
params	Массив параметров пакета docker образа	Массив объектов «Params», структуру объекта отображает Таблица 80	Да
repo	Репозиторий	Строка	Да
branch	Ветка	Строка	Да
commit	Последний коммит	Строка	Да
timestamp	Временная отметка	Строка	Да
status	Статус пакета docker образа (справочное значение)	Строка	Да
components	Массив объектов, содержащий информацию о компонентах пакета docker образа	Массив объектов «Components», структуру объекта отображает Таблица 81	Да
globalParams	Глобальные параметры	Массив объектов «globalParams», структуру объекта отображает Таблица 80	Да
fullStackId	Полный индентификатор пакета docker образа	Строка	Да

Таблица 80 – Структура объектов «Params» и «globalParams»

Поле	Описание	Тип	Обязательность

additionalProp	Пара ключ/ значение.
	Пример:
	{
	"additionalProp1": "value1",
	"additionalProp2": "value2"
	}
		Строка	Да

Таблица 81 – Структура объекта «Components»

Поле	Описание	Тип	Обязательность
name	Наименование компонента	Строка	Да
image	Образ компонента	Строка	Да
params	Параметры компонента	Строка	Да
repo	Репозиторий	Строка	Да
branch	Ветка	Строка	Да
commit	Последний коммит	Строка	Да
tag	Тэг	Строка	Да
timestamp	Временная отметка	Число ($int32)	Да
id	Идентификатор компонента	Строка	Да
created	Дата создания компонента	Число ($int64)	Да
statusStack	Статус компонента (справочное значение)	Строка	Да
environmentVariables	Массив, содержащий информацию о переменных среды	Массив объектов «Variables», структуру объекта отображает Таблица 82	Да
labels	Массив меток, которые привязывают сервис к месту развёртывания	Массив объектов «Labels», структуру объекта отображает Таблица 82	Да
volumes	Массив	Массив объектов «Volumes», структуру объекта отображает Таблица 83	Да
binds	Массив	Массив объектов «Binds», структуру объекта отображает Таблица 85	Да

Поле	Описание	Тип	Обязательность
networks	Массив, содержащий информацию о настройках сети	Массив объектов «Networks», структуру объекта отображает Таблица 86	Да
publishedPorts	Массив, содержащий информацию о внешних портах	Массив объектов «Ports», структуру объекта отображает Таблица 87	Да
adminMode	Режим администратора	Логический (boolean)	Да
runApp	Запустить приложение	Логический (boolean)	Да
showAdminMode	Просмотривать в режиме администратора	Логический (boolean)	Да

Таблица 82 – Структура объектов «Variables» и «Labels»

Поле	Описание	Тип	Обязательность
additionalProp1	Пара ключ/ значение. Пример: { "additionalProp1": "value1", "additionalProp2": "value2" }	Строка	Нет

Таблица 83 – Структура объекта «Volumes»

Поле	Описание	Тип	Обязательность
source	Исходная точка монтирования тома	Строка	Да
target	Целевая точка монтирования тома	Строка	Да
driver	Имя драйвера файловой системы	Строка	Да
options	Массив опций	Массив объектов «Options», структуру объекта отображает Таблица 84	Да
status	Массив статусов	Массив объектов «Status», структуру объекта отображает Таблица 84	Да

Таблица 84 – Структура объектов «Options» и «Status»

Поле	Описание	Тип	Обязательность
additionalProp1	Пара ключ/ значение. Пример: { "additionalProp1": "value1", "additionalProp2": "value2" }	Строка	Нет

Таблица 85 – Структура объекта «Binds»

Поле	Описание	Тип	Обязательность
source	Исходная точка монтирования	Строка	Да
target	Целевая точка монтирования	Строка	Да

Описание структуры объекта «Networks» отображает Таблица 131.

Таблица 86 – Структура объекта «Networks»

Поле	Описание	Тип	Обязательность
name	Имя сети	Строка	Да
driver	Сетевой драйвер	Строка	Да
subnet	Подсеть	Строка	Да

Описание структуры объекта «Ports» отображает Таблица 131.

Таблица 87 – Структура объекта «Ports»

Поле	Описание	Тип	Обязательность
publishedPort	Внешний порт	Строка	Да
targetPort	Целевой порт	Строка	Да
publishMode	Режим публикации	Строка	Да
protocol	Протокол	Строка	Да

Пример JSON ответа отображает Рисунок 40.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

[
  {
    "namespace": "imgtest",
    "stacks": [
      {
        "stackId": "logstash",
        "version": "7.9",
        "namespace": "imgtest",
        "order": 1,
        "undeploymentTemplate": "",
        "params": {
          "LOGSTASH_PUBLISHED_PORT": "14560",
          "LS_JAVA_OPTS": "-Xms256m -Xmx256m -Dnetworkaddress.cache.ttl=10",
          "ES_URL": "",
          "ES_PASSWORD": "",
          "ROLL_OVER_SIZE_GB": "25",
          "ROLL_OVER_DAYS": "1",
          "DELETE_DAYS": "30",
          "NUMBER_OF_REPLICAS": "0",
          "NUMBER_OF_SHARDS": "1",
          "INDEX_NAME": "swarmview",
          "namespace": "imgtest"
        },
        "repo": "Experimental",
        "branch": "master",
        "commit": "d1589d0",
        "timestamp": 1612368147,
        "status": "DEPLOYED",
        "components": [
          {
            "@c": ".DeployedService",
            "name": "logstash-imgtest",
            "image": "imagenarium/logstash:7.9.0",
            "params": "",
            "mode": "replicated",
            "repo": "Experimental",
            "branch": "master",
            "commit": "d1589d0",
            "timestamp": 1612368147,
            "id": "i8giqlugb93ltqt4rl3kx1xa7",
            "created": 1612972610584,
            "stackStatus": "DEPLOYED",
            "environmentVariables": [
              {
                "ELASTICSEARCH_URL": "http://es-imgtest:9200"
              },
              {
                "LS_JAVA_OPTS": "-Xms256m -Xmx256m -Dnetworkaddress.cache.ttl=10"
              },
              {
```

```
        "NODE_NAME": "swarmview-imgtest"
      },
      {
        "NUMBER_OF_REPLICAS": "0"
      },
      {
        "NUMBER_OF_SHARDS": "1"
      },
      {
        "INDEX_NAME": "swarmview"
      },
      {
        "ROLL_OVER_SIZE_GB": "25"
      },
      {
        "ROLL_OVER_DAYS": "1"
      },
      {
        "DELETE_DAYS": "30"
      }
    ],
    "labels": [],
    "volumes": [],
    "binds": [],
    "networks": [
      {
        "name": "net-imgtest",
        "driver": "overlay"
      }
    ],
    "publishedPorts": [
      {
        "publishedPort": "14560",
        "targetPort": "4560",
        "publishMode": "host",
        "protocol": "tcp"
      }
    ],
    "adminMode": false,
    "runApp": true,
    "showAdminMode": true,
    "desiredReplicas": 1,
    "runningReplicas": 1,
    "resolutionMode": "dnsrr",
    "updated": 1612972610586,
    "constraints": [
      {
        "node.labels.logstash": "true"
      }
    ],
    "mutexes": [],
    "tasks": [
      {
```

```
            "id": "p5pxzi5de98kbc5pzi6bruod4",
            "slot": "1",
            "desiredState": "running",
            "currentState": "running",
            "message": "started",
            "nodeName": "eqvm1",
            "nodeIp": "192.168.1.9",
            "nodeId": "ynno145i93uzdug5fhfw3xm88",
            "image": "imagenarium/logstash:7.9.0",
            "updated": 1612972611906
          }
        ],
        "scalable": false,
        "off": false
      },
      {
        "@c": ".DeployedContainer",
        "name": "checker-logstash-imgtest",
        "image": "imagenarium/tcpchecker:latest",
        "params": "",
        "repo": "Experimental",
        "branch": "master",
        "commit": "d1589d0",
        "created": 0,
        "stackStatus": "DEPLOYED",
        "environmentVariables": [
          {
            "HOST": "logstash-imgtest"
          },
          {
            "PORT": "4560"
          },
          {
            "TIMEOUT": "3600"
          }
        ],
        "labels": [],
        "volumes": [],
        "binds": [],
        "networks": [
          {
            "name": "net-imgtest",
            "driver": "overlay"
          }
        ],
        "publishedPorts": [],
        "timestamp": 0,
        "adminMode": false,
        "runApp": false,
        "showAdminMode": false
      }
    ],
    "globalParams": {}
```

```
    },
    {
      "stackId": "swarmview",
      "version": "latest",
      "namespace": "imgtest",
      "order": 2,
      "undeploymentTemplate": "",
      "params": {
        "JAVA_OPTS": "-Xmx256m -Xms256m",
        "namespace": "imgtest"
      },
      "repo": "Experimental",
      "branch": "master",
      "commit": "d1589d0",
      "timestamp": 1612368147,
      "status": "DEPLOYED",
      "components": [
        {
          "@c": ".DeployedService",
          "name": "swarmview-imgtest",
          "image": "imagenarium/swarmview:3.1.0.RC1",
          "params": "",
          "mode": "replicated",
          "repo": "Experimental",
          "branch": "master",
          "commit": "d1589d0",
          "timestamp": 1612368147,
          "id": "zhrytr7aukacz0sn2dsqo056a",
          "created": 1612972634812,
          "stackStatus": "DEPLOYED",
          "environmentVariables": [
            {
              "JAVA_OPTS": "-Xmx256m -Xms256m"
            }
          ],
          "labels": [],
          "volumes": [],
          "binds": [],
          "networks": [
            {
              "name": "net-imgtest",
              "driver": "overlay"
            }
          ],
          "publishedPorts": [],
          "adminMode": false,
          "runApp": true,
          "showAdminMode": true,
          "desiredReplicas": 1,
          "runningReplicas": 1,
          "resolutionMode": "dnsrr",
          "updated": 1612973426404,
          "constraints": [],
```

```
            "mutexes": [],
            "tasks": [
              {
                "id": "yvfknw2ox4zxd74hn6dyhurif",
                "slot": "1",
                "desiredState": "running",
                "currentState": "running",
                "message": "started",
                "nodeName": "eqvm1",
                "nodeIp": "192.168.1.9",
                "nodeId": "ynno145i93uzdug5fhfw3xm88",
                "image": "imagenarium/swarmview:3.1.0.RC1",
                "updated": 1612973361399
              }
            ],
            "scalable": false,
            "off": false
          }
        ],
        "globalParams": {}
      }
    ],
    "locked": false,
    "summaries": [
      {
        "title": "logstash-imgtest",
        "host": "",
        "ports": [
          "14560"
        ]
      }
    ]
  }
]
```


Пример JSON ответа
Рисунок 40

4.3. Справочники, доступные посредством API

4.3.1. Справочник «Статус компонента»

Список возможных значений справочника «Статус компонента» отображает Таблица 88.

Таблица 88 – Возможные значения справочника «Статус компонента»

Константа	Значение	Тип
DEPLOY_IN_PROGRESS	Происходит развертывание компонента	Строка
DEPLOY_BROKEN	Развертывание компонента остановлено системой (присутствует ошибка)	Строка
DEPLOYED	Компонент развёрнут	Строка
CANCELED	Развертывание компонента отменено пользователем	Строка
CANCEL_IN_PROGRESS	Производится отмена развертывания компонента	Строка
UNDEPLOY_IN_PROGRESS	Производится удаление компонента	Строка
UNDEPLOY_BROKEN	Удаление компонента остановлено системой (присутствует ошибка)	Строка

5. СООБЩЕНИЯ

5.1. Сообщения оператору, передаваемые посредством графического интерфейса пользователя

5.1.1. Оповещающие окна

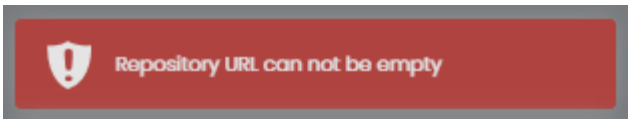
При успешном выполнении операции в ходе выполнения программы в окне Web-браузера появляется всплывающее окно серого цвета с соответствующим сообщением (Рисунок 41)

Сообщение программы об успешном выполнении операции
Рисунок 41

5.1.2. Предупреждающие окна

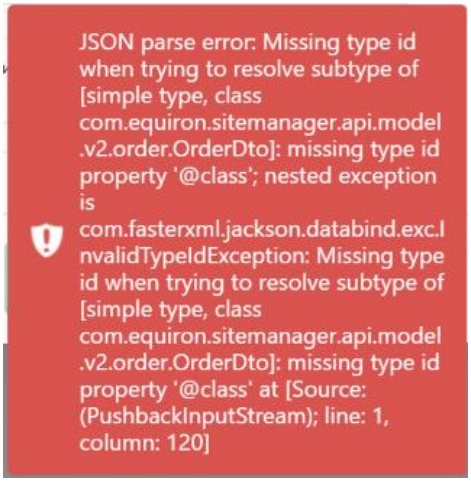
При возникновении ошибок в ходе выполнения программы в окне Web-браузера появляется всплывающее окно красного цвета с двумя типами сообщений:

1) Сообщением программы об ошибке ввода (Рисунок 42).



Сообщение программы об ошибке ввода
Рисунок 42

2) Системным сообщением об ошибке (Рисунок 43).



Системное сообщение об ошибке
Рисунок 43

5.2. Ошибки при обработке запроса

5.2.1. Ответ при возникновении ошибок

При возникновении ошибок в ходе обработки REST-запроса сервер возвращает HTTP-код ошибки.

Пример JSON ответа с ошибкой отображает Рисунок 44.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Content-Length: 146
```

Пример JSON ответа с ошибкой
Рисунок 44

5.2.2. Описание ошибок

HTTP-коды ошибок в ответе на запрос отображает Таблица 89.

Таблица 89 – HTTP-коды ошибок в ответе на запрос

Код ошибки	Описание
400	Операция не выполнена. Неверные входные параметры. Расширенные коды ошибок в ответе на запрос отображает Таблица 90
500	Операция не выполнена. Внутренняя ошибка сервера

Перечень и описание расширенных кодов ошибок в ответе на запрос находится в разработке и может быть дополнено и расширено.

Таблица 90 – Расширенные коды ошибок в ответе на запрос

Код ошибки	Описание

ПЕРЕЧЕНЬ ТЕРМИНОВ

В настоящем документе использованы следующие термины:

1) Средство вычислительной техники (СВТ) — ПЭВМ (персональная электронно-вычислительная машина) либо другое вычислительное оборудование (мэйнфрейм, мини-ЭВМ, микро-ЭВМ, КПК (карманный персональный компьютер), компьютерный терминал).

2) СВТ индивидуального пользования — вычислительное оборудование, обеспечивающее:

– автоматизацию вычислительной составляющей повседневной деятельности сотрудников Заказчика;

– доступ к информационным сервисам, автоматизирующим бизнес-процессы предприятия Заказчика.

3) СВТ коллективного пользования — вычислительное оборудование, предназначенное для:

– организации вычислительной платформы, обеспечивающей автоматизацию бизнес-процессов Заказчика;

– контроля и настройки СВТ, входящих в автоматизированную систему;

– накопления и обработки данных, используемых при автоматизации бизнес-процессов Заказчика.

4) Общее программное обеспечение — совокупность программных компонентов, обеспечивающая минимум функциональности СВТ:

– среду для запуска и работы остальных программных средств (операционная система);

– средства для работы со структурированными наборами данных (СУБД);

– средства для доступа к ресурсам сети Интернет (Web-браузер);

– средства для публикации ресурсов СВТ в сети Интернет (Web-сервер).

5) Специальное программное обеспечение — совокупность программных компонентов, специально разрабатываемых для данного конкретного СВТ (не «коробочное ПО»).

6) Оконечное оборудование — устройства и приборы, управляемые автоматизированной системой напрямую (посредством инфокоммуникационных каналов) или опосредованно (через функциональное взаимодействие со смежными системами) и предназначенные для выполнения технологических функций (принтер, сканер, регистратор, контроллер и т.д.).

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

АРМ	Автоматизированное рабочее место
АС	Автоматизированная система
КТС	Комплекс технических средств
ОС	Операционная система
ОПО	Общее программное обеспечение
ПО	Программное обеспечение
ПЭВМ	Персональная электронно-вычислительная машина
СВТ	Средство вычислительной техники
СПО	Специальное программное обеспечение

ПЕРЕЧЕНЬ РИСУНКОВ

ПЕРЕЧЕНЬ ТАБЛИЦ

Лист регистрации изменений

	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	измененных	замененных	новых	аннулированных					