

# Инструкция по выполнению задания второго этапа конкурса «Прикладное программирование if...else»

Мы рады приветствовать Вас на втором этапе конкурса «Прикладное программирование if...else»! В рамках второго этапа конкурса Вам нужно продолжить работу над RESTful API сервисом, который вы реализовали на первом этапе.

Функциональность будет описана ниже в виде краткого технического задания (см. разделы "[Легенда](#)", "[Функционал системы](#)", "[API Declarations](#)").

Выполненное задание необходимо представить на оценку экспертам вместе с исходным кодом программы (см. разделы "[Отправка задания](#)", "[Работа с Docker](#)").

Желаем Вам удачи!

Ответы на часто задаваемые вопросы - [FAQ](#)

Программное обеспечение для финального этапа олимпиады - [Тык](#)

# Оглавление

<b>Легенда</b>	<b>4</b>
<b>Функционал системы</b>	<b>5</b>
В системе должны быть следующие компоненты	5
В контроллерах должен быть доступен следующий функционал	5
<b>Задание</b>	<b>7</b>
<b>Этапы задания</b>	<b>8</b>
Нулевой этап	8
Первый этап	9
Второй этап	9
Третий этап*	9
<b>Тестирование задания</b>	<b>10</b>
<b>Отправка работы</b>	<b>12</b>
<b>Работа с Docker</b>	<b>13</b>
Сохранение образа контейнера	13
Пример файла docker-compose.yml	13
<b>API Declarations</b>	<b>14</b>
1) Аутентификация пользователя	14
API 1: Регистрация нового аккаунта	14
2) Аккаунт пользователя	15
API 1: Получение информации об аккаунте пользователя	15
API 2: Поиск аккаунтов пользователей по параметрам	16
API 3: Добавление аккаунта пользователя	17
API 4: Обновление данных аккаунта пользователя	18
API 5: Удаление аккаунта пользователя	20
3) Точка локации животных	20
API 1: Получение информации о точке локации животных	20
API 2: Добавление точки локации животных	21
API 3: Изменение точки локации животных	22
API 4: Удаление точки локации животных	23
4) Зоны	25
API 1: Получение информации о зоне	25
API 2: Добавление зоны	26
API 3: Обновление зоны	28
API 4: Удаление зоны	30
5) Типы животных	31
API 1: Получение информации о типе животного	31
API 2: Добавление типа животного	31
API 3: Изменение типа животного	32
API 4: Удаление типа животного	33
6) Животное	34
API 1: Получение информации о животном	34
API 2: Поиск животных по параметрам	35
API 3: Добавление нового животного	37
API 4: Обновление информации о животном	39
API 5: Удаление животного	41
API 6: Добавление типа животного к животному	42
API 7: Изменение типа животного у животного	43
API 8: Удаление типа животного у животного	45
7) Точка локации, посещенная животным	47
API 1: Просмотр точек локации, посещенных животным	47

API 2: Добавление точки локации, посещенной животным	48
API 3: Изменение точки локации, посещенной животным	49
API 4: Удаление точки локации, посещенной животным	50
8) Аналитика по зонам	51
API 1: Просмотр информации о перемещениях животных в зоне	51

# Легенда

Наша компания “Дрип-Чип” занимается чипированием животных в стране “Вондерланд” для отслеживания их перемещения и жизненных циклов. Перемещение животных на планете крайне важно, в том числе чтобы защитить их от гибели.

В этом году наша компания решила создать единую базу, в которой будут перенесены записи прошлых лет, для проведения многолетних экспериментов, связанных с миграциями животных, а также для отслеживания изменения сред обитания и ведения истории.

# Функционал системы

В системе должны быть следующие компоненты

- Account
- Role
- Animal
- Animal Type
- Location Point
- Animal Visited Location
- Area
- Area Analytics

В контроллерах должен быть доступен следующий функционал

Authentication:

- Регистрация аккаунта

Account:

- Просмотр информации об аккаунте
- Поиск/изменение/удаление аккаунта

Animal:

- Просмотр информации о животном
- Поиск/создание/изменение/удаление животного
- Создание/изменение/удаление типа животного

Animal Type:

- Просмотр информации о типе животного
- Создание/изменение/удаление типа животного

Location Point:

- Просмотр информации о точке локации
- Создание/изменение/удаление точки локации

Animal Visited Location:

- Просмотр информации о перемещении животного
- Создание/изменение/удаление точки локации у животного

Area:

- Просмотр информации о зоне

- Создание/изменение/удаление зоны

#### Area Analytics:

- Просмотр информации о перемещениях животных в зоне

## Задание

1. Реализовать один, несколько или все этапы задания (см. разделы [“Этапы задания”](#) и [“API Declarations”](#))
2. Настроить Docker (см. раздел [“Работа с Docker”](#))
3. Проверить верность своего решения (см. раздел [“Тестирование задания”](#))
4. Отправить нам свое решение (см. раздел [“Отправка задания”](#))

# Этапы задания

В зависимости от своего уровня подготовки, вы можете реализовать один, несколько или все этапы задания.

**ВНИМАНИЕ!** Необходимо использовать только **СУБД PostgreSQL** последней версии(**latest**) в своем RESTful API сервисе (если при выполнении первого этапа использовали другую версию или СУБД) как указано в примере **docker-compose.yml**.

## Нулевой этап

Необходимо организовать разграничение доступа к функционалу сервиса для аккаунтов. Для этого нужно создать иерархию аккаунтов, назначив им одну из следующих ролей: "ADMIN", "CHIPPER", "USER" (указаны в порядке уменьшения прав доступа).

При первом запуске вашего приложения в базе данных автоматически должны создаваться аккаунты со следующими данными:

```
{
  "id": 1,                // Идентификатор аккаунта пользователя
  "firstName": "adminFirstName", // Имя пользователя
  "lastName": "adminLastName",   // Фамилия пользователя
  "email": "admin@simbirsoft.com", // Адрес электронной почты
  "password": "qwerty123",       // Пароль от аккаунта пользователя
  "role": "ADMIN"                // Роль аккаунта
},
{
  "id": 2,                // Идентификатор аккаунта пользователя
  "firstName": "chipperFirstName", // Имя пользователя
  "lastName": "chipperLastName",   // Фамилия пользователя
  "email": "chipper@simbirsoft.com", // Адрес электронной почты
  "password": "qwerty123",       // Пароль от аккаунта пользователя
  "role": "CHIPPER"            // Роль аккаунта
},
{
  "id": 3,                // Идентификатор аккаунта пользователя
  "firstName": "userFirstName",    // Имя пользователя
  "lastName": "userLastName",     // Фамилия пользователя
  "email": "user@simbirsoft.com",  // Адрес электронной почты
  "password": "qwerty123",       // Пароль от аккаунта пользователя
  "role": "USER"               // Роль аккаунта
}
```



Подробное описание методов представлено ниже в [API Declarations](#).

## Первый этап

Необходимо реализовать следующие методы:

- **GET /areas/{areald}** - Просмотр информации о зоне
- **POST /areas** - Добавление зоны
- **PUT /areas/{areald}** - Изменение зоны
- **DELETE /areas/{areald}** - Удаление зоны

Подробное описание методов представлено ниже в [API Declarations](#).

## Второй этап

Необходимо реализовать следующий метод:

- **GET /areas/{areald}/analytics** - Просмотр информации о перемещениях животных в зоне

Подробное описание метода представлено ниже в [API Declarations](#).

## Третий этап\*

Наша компания “Дрип-Чип” срочно нуждается в чиперах, поэтому автор не успел описать задание третьего этапа, но успел подготовить пакет автотестов для него. Вам предстоит самостоятельно разобраться в требованиях тестов и реализовать необходимый функционал.

**Каждый следующий этап предполагает реализацию предыдущего, то есть начинать надо с нулевого этапа и двигаться последовательно.**

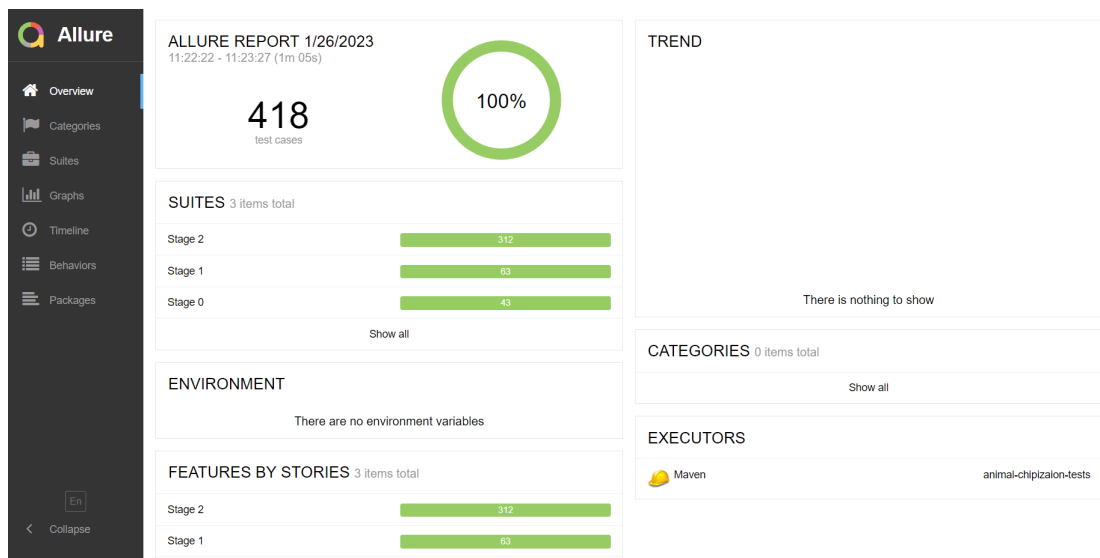
# Тестирование задания

Для проверки работоспособности своего приложения вам необходимо использовать **Docker-Image** из [Docker-Hub](#). Название контейнера указано в примере **docker-compose.yml** файла, который расположен ниже.

Для тестирования задания вам необходимо:

1. Добавить ваше приложение с базой в **docker-compose.yml**
2. Добавить контейнер с автотестами в файл **docker-compose.yml**
3. Указать в **SERVER\_URL** ссылку на вашу **API**
4. Развернуть приложение с автотестами командой **docker compose up**
5. Перейти по адресу "<http://localhost:8090>"

Тестирование будет проведено автоматически как только запустится ваше приложение. После того, как тестирование будет завершено, у вас будет доступна данная страница:



На этой странице вы сможете увидеть на сколько ваша работа соответствует базовым требованиям, которые описаны ниже в [API Declarations](#). Вам будет доступна небольшая часть тестов, полное тестирование будет проводиться после отправки вашей работы нам на проверку.

Дополнительно вы можете изменить настройку **STAGE**, чтобы указать для какого этапа вы хотите запустить тесты. По умолчанию данная настройка установлена в **all** - это означает, что все тесты всех этапов будут запущены.

Каждый раз при создании нового контейнера, у вас будет использоваться актуальная версия **Docker-Image** из [Docker-Hub](#).











Пример с полным файлом **docker-compose.yml** вы можете посмотреть ниже.

# Отправка работы

Для отправки работы вам необходимо:

1. Создать архив с вашей Фамилией и Именем. Пример: **“Иванов\_Иван.zip”**.  
Архив может поддерживать такие форматы как **.rar**, **.zip**, **.7z**.
2. Поместить в архив в папку **Sources** ваш исходный код приложения
3. Поместить в архив в папку **Build** ваш файл **docker-compose.yml** который содержит ваше приложение и тесты. Поместить в ту же папку ваш архив с **docker-image**. Как выгрузить данный архив вы можете прочитать ниже в разделе [Работа с Docker](#).
4. После выполнения предыдущих пунктов вы должны загрузить ваш архив на **Яндекс-Диск** или **Google Disk**.
5. Отправьте ссылку на архив работы в личном кабинете участника.

Пример архива с папками:

 Иванов_Иван.zip	1/21/2023 7:59 PM	Архив ZIP - WinR...
 Build	1/21/2023 8:00 PM	File folder
 docker-compose.yml	1/12/2023 7:38 AM	YML File
 animal_chipization_2022.tar	1/12/2023 7:39 AM	Архив WinRAR
 Sources	1/21/2023 8:00 PM	File folder
 Database	1/21/2023 8:03 PM	File folder
 WebApi	1/21/2023 8:03 PM	File folder
 .gitattributes	12/12/2022 9:20 AM	Text Document
 .gitignore	12/12/2022 9:20 AM	Text Document
 Olymp Project.sln	12/12/2022 9:20 AM	Visual Studio Solu...

Перед отправкой работы **обязательно** проведите тестирование приложения, используя актуальную версию **Docker-Image** из [Docker-Hub](#).

**ВНИМАНИЕ!** Для проверки будут рассматриваться работы только с **СУБД PostgreSQL** последней версии(**latest**).

**ВНИМАНИЕ!** Все работы проверяются автоматически, при ошибках в названии файлов или папок ваша работа может некорректно обрабатываться, и это будет означать **не верное решение вашей работы**.

# Работа с Docker

## Сохранение образа контейнера

Чтобы сохранить **docker-image** с вашим приложением необходимо:

1. В командной строке перейти в нужную директорию для сохранения архива.
2. Ввести команду “**docker save -o {FileName}.tar {ImageName}**”, и указать **FileName** - название файла с сохраняемым образом, **ImageName** - это название **docker-image** вашего приложения.
3. После всех действий у вас должен получиться файл с расширением .tar, готовый к отправке.

## Пример файла docker-compose.yml

```
version: '3.9'

services:
  # Сервис для разворачивания контейнера с базой данных
  database:
    image: postgres:latest
    volumes:
      - /var/lib/postgresql/data/
    environment:
      - POSTGRES_DB=animal-chipization
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=password

  # Сервис для разворачивания контейнера с приложением
  webapi:
    image: webapi
    ports:
      - "8080:8080"
    depends_on:
      - database
    environment:
      - SPRING_DATASOURCE_URL=jdbc:postgresql://db:5432/animal-chipization
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=password

  # Сервис для разворачивания контейнера с автотестами
  tests:
    image: mrexpen/planet_olymp_phase2
    pull_policy: always
    ports:
      - "8090:8080"
    depends_on:
      - webapi
    environment:
      SERVER_URL: http://webapi:8080
      STAGE: all
    # all - запуск всех тестов из трёх доступных этапов
    # 0, 1, 2 или 3 - запуск тестов для соответствующего этапа
```

# API Declarations

Для передачи данных в Body запросов используется формат JSON.

## 1) Аутентификация пользователя

**API 1:** Регистрация нового аккаунта

**POST** - /registration

- request

```
Body {  
    "firstName": "string", // Имя пользователя  
    "lastName": "string", // Фамилия пользователя  
    "email": "string",     // Адрес электронной почты  
    "password": "string"   // Пароль от аккаунта пользователя  
}
```

- response

```
Body {  
    "id": "int",           // Идентификатор аккаунта пользователя  
    "firstName": "string", // Имя пользователя  
    "lastName": "string",  // Фамилия пользователя  
    "email": "string",     // Адрес электронной почты  
    "role": "string",      // Роль аккаунта пользователя, по умолчанию при  
    // регистрации "USER"  
}
```

Условие	Статус
Запрос успешно выполнен	201
firstName = null, firstName = "" или состоит из пробелов, lastName = null, lastName = "" или состоит из пробелов, email = null, email = "" или состоит из пробелов, email аккаунта не валидный, password = null, password = "" или состоит из пробелов	400

Неверные авторизационные данные	
Запрос от авторизованного аккаунта	403
Аккаунт с таким email уже существует	409

## 2) Аккаунт пользователя

**API 1:** Получение информации об аккаунте пользователя

**GET** - /accounts/{accountId}

{accountId}: "int" // Идентификатор аккаунта пользователя

- request

```
Body {
    empty
}
```

- response

```
Body {
    "id": "int", // Идентификатор аккаунта пользователя
    "firstName": "string", // Имя пользователя
    "lastName": "string", // Фамилия пользователя
    "email": "string" // Адрес электронной почты
    "role": "string", // Роль аккаунта пользователя, по умолчанию при
    регистрации "USER"
}
```

Условие	Статус
Запрос успешно выполнен	200
accountId = null, accountId <= 0	400
Запрос от неавторизованного аккаунта	401
Аккаунт с ролью "USER" или "CHIPPER" пытается получить информацию о чужом аккаунте (даже если такого аккаунта нет)	403
Для аккаунтов с ролями "ADMIN":	404

Аккаунт с таким accountId не найден	
-------------------------------------	--

**API 2:** Поиск аккаунтов пользователей по параметрам

**GET** - /accounts/search

    ?firstName={firstName}  
    &lastName={lastName}  
    &email={email}  
    &from={from}  
    &size={size}

{firstName}: "string",   // Имя пользователя, может использоваться только часть имени без учета регистра, если null - не учитывается

{lastName}: "string",   // Фамилия пользователя, может использоваться только часть фамилии без учета регистра, если null - не учитывается

{email}: "string",       // Адрес электронной почты, может использоваться только часть адреса электронной почты без учета регистра, если null - не учитывается

{from}: "int"            // Количество элементов, которое необходимо пропустить для формирования страницы с результатами (по умолчанию 0)

{size}: "int"            // Количество элементов на странице (по умолчанию 10)

- request

```
Body {  
    empty  
}
```

- response

```
Body [  
    {  
        "id": "int",            // Идентификатор аккаунта пользователя  
        "firstName": "string", // Имя пользователя  
        "lastName": "string",  // Фамилия пользователя  
        "email": "string"      // Адрес электронной почты  
        "role": "string",      // Роль аккаунта пользователя, по умолчанию  
        "USER"  
    }  
]
```

**Результаты поиска сортируются по id аккаунта от наименьшего к наибольшему**

Условие	Статус
---------	--------



Запрос успешно выполнен	200
from < 0, size <= 0	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403

### API 3: Добавление аккаунта пользователя

#### POST - /accounts

- request

```
Body {
  "firstName": "string", // Имя пользователя
  "lastName": "string", // Фамилия пользователя
  "email": "string",     // Адрес электронной почты
  "password": "string",  // Пароль от аккаунта
  "role": "string",      // Роль аккаунта пользователя, доступные значения
                        "ADMIN", "CHIPPER", "USER"
}
```

- response

```
Body {
  "id": "int",           // Идентификатор аккаунта пользователя
  "firstName": "string", // Новое имя пользователя
  "lastName": "string",  // Новая фамилия пользователя
  "email": "string",     // Новый адрес электронной почты
  "role": "string",      // Роль аккаунта пользователя
}
```

Условие	Статус
Запрос успешно выполнен	201
firstName = null, firstName = "" или состоит из пробелов, lastName = null, lastName = "" или состоит из пробелов, email = null,	400

email = "" или состоит из пробелов, email аккаунта не валидный, password = null, password = "" или состоит из пробелов, role != "ADMIN", "CHIPPER", "USER"	
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403
Аккаунт с таким email уже существует	409

#### API 4: Обновление данных аккаунта пользователя

**PUT** - /accounts/{accountId}

{accountId}: "int" // Идентификатор аккаунта пользователя

- request

```
Body {
  "firstName": "string", // Новое имя пользователя
  "lastName": "string", // Новая фамилия пользователя
  "email": "string",     // Новый адрес электронной почты
  "password": "string"   // Новый пароль от аккаунта
  "role": "string",      // Роль аккаунта пользователя, доступные значения
  "ADMIN", "CHIPPER", "USER"
}
```

- response

```
Body {
  "id": "int",           // Идентификатор аккаунта пользователя
  "firstName": "string", // Новое имя пользователя
  "lastName": "string",  // Новая фамилия пользователя
  "email": "string",     // Новый адрес электронной почты
  "role": "string",      // Роль аккаунта пользователя
}
```

Условие	Статус
Запрос успешно выполнен	200

accountId = null, accountId <= 0, firstName = null, firstName = "" или состоит из пробелов, lastName = null, lastName = "" или состоит из пробелов, email = null, email = "" или состоит из пробелов, email аккаунта не валидный, password = null, password = "" или состоит из пробелов, role != "ADMIN", "CHIPPER", "USER"	400
Запрос от неавторизованного аккаунта	401
Для аккаунтов с ролями "CHIPPER" и "USER": Обновление не своего аккаунта. Аккаунт не найден	403
Для аккаунтов с ролью "ADMIN": Аккаунт не найден	404
Аккаунт с таким email уже существует	409

### API 5: Удаление аккаунта пользователя

**DELETE** - /accounts/{accountId}

{accountId}: "int" // Идентификатор аккаунта пользователя

- request

```
Body {  
    empty  
}
```

- response

```
Body {  
    empty  
}
```

Условие	Статус
Запрос успешно выполнен	200
accountId = null, accountId <= 0, Аккаунт связан с животным	400
Запрос от неавторизованного аккаунта	401
Для аккаунтов с ролями "CHIPPER" и "USER": Удаление не своего аккаунта. Аккаунт не найден	403
Для аккаунтов с ролью "ADMIN": Аккаунт не найден	404

### 3) Точка локации животных

**API 1:** Получение информации о точке локации животных

**GET** - /locations/{pointId}

{pointId}: "long" // Идентификатор точки локации

- request

```
Body {  
    empty  
}
```

```
}
```

- response

```
Body {
```

```
"id": "long",           // Идентификатор точки локации
"latitude": "double",   // Географическая широта в градусах
"longitude": "double"   // Географическая долгота в градусах
```

```
}
```

Условие	Статус
Запрос успешно выполнен	200
pointId = null, pointId <= 0	400
Запрос от неавторизованного аккаунта	401
Точка локации с таким pointId не найдена	404

**API 2:** Добавление точки локации животных

**POST** - /locations

- request

```
Body {
```

```
"latitude": "double",   // Географическая широта в градусах
"longitude": "double"   // Географическая долгота в градусах
```

```
}
```

- response

```
Body {
```

```
"id": "long",           // Идентификатор точки локации
"latitude": "double",   // Географическая широта в градусах
"longitude": "double"   // Географическая долгота в градусах
```

```
}
```

Условие	Статус
Запрос успешно выполнен	201

latitude = null, latitude < -90, latitude > 90, longitude = null, longitude < -180, longitude > 180	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Точка локации с такими latitude и longitude уже существует	409

### API 3: Изменение точки локации животных

**PUT** - /locations/{pointId}

{pointId}: "long" // Идентификатор точки локации

- request

```
Body {
  "latitude": "double", // Новая географическая широта в градусах
  "longitude": "double" // Новая географическая долгота в градусах
}
```

- response

```
Body {
  "id": "long", // Идентификатор точки локации
  "latitude": "double", // Новая географическая широта в градусах
  "longitude": "double" // Новая географическая долгота в градусах
}
```

Условие	Статус
Запрос успешно выполнен	200
pointId = null, pointId <= 0, latitude = null,	400

latitude < -90, latitude > 90, longitude = null, longitude < -180, longitude > 180  Если точка используется как точка чипирования или как посещенная точка, то её изменять нельзя	
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Точка локации с таким pointId не найдена	404
Точка локации с такими latitude и longitude уже существует	409

#### API 4: Удаление точки локации животных

##### **DELETE** - /locations/{pointId}

{pointId}: "long"      // Идентификатор точки локации

- request

```
Body {
    empty
}
```

- response

```
Body {
    empty
}
```

Условие	Статус
Запрос успешно выполнен	200
pointId = null, pointId <= 0, Точка локации связана с животным	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403
Точка локации с таким pointId не найдена	404



## 4) Зоны

### API 1: Получение информации о зоне

**GET** - /areas/{areald}

{areald}: "long" // Идентификатор зоны

- request

```
Body {  
    empty  
}
```

- response

```
Body {  
    "id": "long", // Идентификатор зоны  
    "name": "string", // Название зоны  
    "areaPoints": [  
        {  
            "longitude": "double", // Географическая долгота в градусах  
            "latitude": "double" // Географическая широта в градусах  
        },  
        {  
            "longitude": "double", // Географическая долгота в градусах  
            "latitude": "double" // Географическая широта в градусах  
        },  
        {  
            "longitude": "double", // Географическая долгота в градусах  
            "latitude": "double" // Географическая широта в градусах  
        }  
    ]  
}
```

Условие	Статус
Запрос успешно выполнен	200
areald = null, areald <= 0	400
Запрос от неавторизованного аккаунта	401
Зона с таким areald не найдена	404

## API 2: Добавление зоны

### POST - /areas

- request

```
Body {
  "name": "string",          // Название зоны
  "areaPoints": [
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double"   // Географическая широта в градусах
    },
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double"   // Географическая широта в градусах
    },
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double"   // Географическая широта в градусах
    }
  ]
}
```

- response

```
Body {
  "id": "long",              // Идентификатор зоны
  "name": "string",          // Название зоны
  "areaPoints": [
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double"   // Географическая широта в градусах
    },
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double"   // Географическая широта в градусах
    },
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double"   // Географическая широта в градусах
    }
  ]
}
```

```

    }
  ]
}

```

**ВНИМАНИЕ!** Координаты точек в массиве указываются в порядке их соединения для образования замкнутого **САМО НЕПЕРЕСЕКАЮЩЕГОСЯ** многоугольника, то есть точки соединяются последовательно от первой к последней, последняя соединяется с первой, при этом полученные отрезки не должны пересекать друг друга.

Условие	Статус
Запрос успешно выполнен	201
name = null, name = "" или состоит из пробелов, areaPoints = null, Одна или несколько точек из areaPoints = null, longitude = null, longitude < -180, longitude > 180, latitude = null, latitude < -90, latitude > 90, areaPoints содержит меньше трех точек. Все точки лежат на одной прямой. Границы новой зоны пересекаются между собой. Границы новой зоны пересекают границы уже существующей зоны. Граница новой зоны находится внутри границ существующей зоны. Границы существующей зоны находятся внутри границ новой зоны. Новая зона имеет дубликаты точек. Новая зона состоит из части точек существующей зоны и находится на площади существующей зоны.	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403

<p>Зона, состоящая из таких точек, уже существует. (При этом важен порядок, в котором указаны точки, но не важна начальная точка).</p> <p>Зона с таким name уже существует</p>	409
--	-----

### API 3: Обновление зоны

**PUT** - /areas/{areaid}

{areaid}: "long" // Идентификатор зоны

- request

```
Body {
  "name": "string", // Название зоны
  "areaPoints": [
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double" // Географическая широта в градусах
    },
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double" // Географическая широта в градусах
    },
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double" // Географическая широта в градусах
    }
  ]
}
```

- response

```
Body {
  "id": "long", // Идентификатор зоны
  "name": "string" // Название зоны
  "areaPoints": [
    {
      "longitude": "double", // Географическая долгота в градусах
      "latitude": "double" // Географическая широта в градусах
    },
    {
      "longitude": "double", // Географическая долгота в градусах
```

```

        "latitude": "double"           // Географическая широта в градусах
    },
    {
        "longitude": "double",         // Географическая долгота в градусах
        "latitude": "double"           // Географическая широта в градусах
    }
]
}
}

```

**ВНИМАНИЕ!** Координаты точек в массиве указываются в порядке их соединения для образования замкнутого **САМО НЕПЕРЕСЕКАЮЩЕГОСЯ** многоугольника, то есть точки соединяются последовательно от первой к последней, последняя соединяется с первой, при этом полученные отрезки не должны пересекать друг друга.

Условие	Статус
Запрос успешно выполнен	200
<p> areald = null,  areald &lt;= 0,  name = null,  name = "" или состоит из пробелов,  areaPoints = null,  Одна или несколько точек из areaPoints = null,  longitude = null,  longitude &lt; -180,  longitude &gt; 180,  latitude = null,  latitude &lt; -90,  latitude &gt; 90  areaPoints содержит меньше трех точек.  Все точки лежат на одной прямой.  Границы новой зоны пересекаются между собой.  Границы новой зоны пересекают границы уже существующей зоны.  Новая зона имеет дубликаты точек.  Граница новой зоны находится внутри границ существующей зоны. </p>	400

Границы существующей зоны находятся внутри границ новой зоны. Новая зона состоит из части точек существующей зоны и находится на площади существующей зоны.	
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403
Зона, состоящая из таких точек, уже существует. (При этом важен порядок, в котором указаны точки, но не важна начальная точка). Зона с таким name уже существует	409

#### API 4: Удаление зоны

**DELETE** - /areas/{areald}

{areald}: "long"

// Идентификатор зоны

- request

```
Body {
    empty
}
```

- response

```
Body {
    empty
}
```

Условие	Статус
Запрос успешно выполнен	200
areald = null, areald <= 0	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403
Зона с таким areald не найдена	404

## 5) Типы животных

### API 1: Получение информации о типе животного

**GET** - /animals/types/{typeId}

{typeId}: "long" // Идентификатор типа животного

- request

```
Body {  
    empty  
}
```

- response

```
Body {  
    "id": "long", // Идентификатор типа животного  
    "type": "string" // Тип животного  
}
```

Условие	Статус
Запрос успешно выполнен	200
typeId = null, typeId <= 0	400
Запрос от неавторизованного аккаунта	401
Тип животного с таким typeId не найден	404

### API 2: Добавление типа животного

**POST** - /animals/types

- request

```
Body {  
    "type": "string" // Тип животного  
}
```

- response

```
Body {  
    "id": "long", // Идентификатор типа животного  
    "type": "string" // Тип животного  
}
```

}

Условие	Статус
Запрос успешно выполнен	201
type = null, type = "" или состоит из пробелов	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Тип животного с таким type уже существует	409

### API 3: Изменение типа животного

**PUT** - /animals/types/{typeId}

{typeId}: "long" // Идентификатор типа животного

- request

```
Body {  
    "type": "string" // Новый тип животного  
}
```

- response

```
Body {  
    "id": "long", // Идентификатор типа животного  
    "type": "string" // Новый тип животного  
}
```

Условие	Статус
Запрос успешно выполнен	200
typeId = null, typeId <= 0, type = null, type = "" или состоит из пробелов	400
Запрос от неавторизованного аккаунта	401



У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Тип животного с таким typeId не найден	404
Тип животного с таким type уже существует	409

#### API 4: Удаление типа животного

**DELETE** - /animals/types/{typeId}

{typeId}: "long" // Идентификатор типа животного

- request

```
Body {
    empty
}
```

- response

```
Body {
    empty
}
```

Условие	Статус
Запрос успешно выполнен	200
typeId = null, typeId <= 0 Есть животные с типом с typeId	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403
Тип животного с таким typeId не найден	404

## 6) Животное

### API 1: Получение информации о животном

**GET** - /animals/{animalId}

{animalId}: "long" // Идентификатор животного

- request

```
Body {  
    empty  
}
```

- response

```
Body {  
    "id": "long", // Идентификатор животного  
    "animalTypes": "[long]", // Массив идентификаторов типов  
    ЖИВОТНОГО  
    "weight": "float", // Масса животного, кг  
    "length": "float", // Длина животного, м  
    "height": "float", // Высота животного, м  
    "gender": "string", // Гендерный признак животного,  
    доступные значения "MALE", "FEMALE", "OTHER"  
    "lifeStatus": "string", // Жизненный статус животного,  
    доступные значения "ALIVE"(устанавливается автоматически при добавлении  
    нового животного), "DEAD"(можно установить при обновлении информации о  
    животном)  
    "chippingDateTime": "dateTime", // Дата и время чипирования в  
    формате ISO-8601 (устанавливается автоматически на момент добавления  
    животного)  
    "chipperId": "int", // Идентификатор аккаунта с ролью  
    "CHIPPER" или "ADMIN"  
    "chippingLocationId": "long", // Идентификатор точки локации  
    ЖИВОТНЫХ  
    "visitedLocations": "[long]", // Массив идентификаторов объектов  
    с информацией о посещенных точках локаций  
    "deathDateTime": "dateTime" // Дата и время смерти животного в  
    формате ISO-8601 (устанавливается автоматически при смене lifeStatus на  
    "DEAD"). Равняется null, пока lifeStatus = "ALIVE".  
}
```

Условие	Статус
Запрос успешно выполнен	200
animalId = null, animalId <= 0	400
Запрос от неавторизованного аккаунта	401
Животное с animalId не найдено	404

## API 2: Поиск животных по параметрам

**GET** - /animals/search?

startDateTime={startDateTime}  
 &endTime={endTime}  
 &chipperId={chipperId}  
 &chippingLocationId={chippingLocationId}  
 &lifeStatus={lifeStatus}  
 &gender={gender}  
 &from=0  
 &size=10

{startDateTime}: "dateTime", // Дата и время, не раньше которых произошло чипирование животного в формате ISO-8601, если null - не учитывается

{endTime}: "dateTime", // Дата и время, не позже которых произошло чипирование животного в формате ISO-8601, если null - не учитывается

{chipperId}: "int", // Идентификатор аккаунта с ролью "CHIPPER" или "ADMIN", если null - не учитывается

{chippingLocationId}: "long", // Идентификатор локации чипирования животного, если null - не учитывается

{lifeStatus}: "string", // Жизненный статус животного, если null - не учитывается

{gender}: "string", // Гендерная принадлежность животного, если null - не учитывается

{from}: "int", // Количество элементов, которое необходимо пропустить для формирования страницы с результатами (по умолчанию 0)

{size}: "int", // Количество элементов на странице (по умолчанию 10)

- request

```
Body {
    empty
```

```
}
```

- response

```
Body [  
  {  
    "id": "long", // Идентификатор животного  
    "animalTypes": "[long]", // Массив идентификаторов типов  
    ЖИВОТНОГО  
    "weight": "float", // Масса животного, кг  
    "length": "float", // Длина животного, м  
    "height": "float", // Высота животного, м  
    "gender": "string", // Гендерный признак животного,  
    доступные значения "MALE", "FEMALE", "OTHER"  
    "lifeStatus": "string", // Жизненный статус животного,  
    доступные значения "ALIVE"(устанавливается автоматически при добавлении  
    нового животного), "DEAD"(можно установить при обновлении информации о  
    животном)  
    "chippingDateTime": "dateTime", // Дата и время чипирования в  
    формате ISO-8601 (устанавливается автоматически на момент добавления  
    животного)  
    "chipperId": "int", // Идентификатор аккаунта с ролью  
    "CHIPPER" или "ADMIN"  
    "chippingLocationId": "long", // Идентификатор точки локации  
    ЖИВОТНЫХ  
    "visitedLocations": "[long]", // Массив идентификаторов объектов  
    с информацией о посещенных точках локаций  
    "deathDateTime": "dateTime" // Дата и время смерти животного в  
    формате ISO-8601 (устанавливается автоматически при смене lifeStatus на  
    "DEAD"). Равняется null, пока lifeStatus = "ALIVE".  
  }  
]
```

**Результаты поиска сортируются по id животного от наименьшего к наибольшему**

Условие	Статус
Запрос успешно выполнен	200
from < 0, size <= 0, startDateTime - не в формате ISO-8601, endDateTime - не в формате ISO-8601, chipperId <= 0, chippingLocationId <= 0, lifeStatus != "ALIVE", "DEAD", gender != "MALE", "FEMALE", "OTHER"	400
Запрос от неавторизованного аккаунта	401

### API 3: Добавление нового животного

#### POST - /animals

##### - request

```

Body {
  "animalTypes": "[long]",      // Массив идентификаторов типов животного
  "weight": "float",            // Масса животного, кг
  "length": "float",            // Длина животного, м
  "height": "float",            // Высота животного, м
  "gender": "string",           // Гендерный признак животного, доступные
                                значения "MALE", "FEMALE", "OTHER"
  "chipperId": "int",           // Идентификатор аккаунта с ролью "CHIPPER"
                                или "ADMIN"
  "chippingLocationId": "long"  // Идентификатор точки локации животных
}

```

##### - response

```

Body {
  "id": "long",                 // Идентификатор животного
  "animalTypes": "[long]",      // Массив идентификаторов типов
                                животного
  "weight": "float",            // Масса животного, кг
  "length": "float",            // Длина животного, м
  "height": "float",            // Высота животного, м
}

```

```
    "gender": "string", // Гендерный признак животного,
    // доступные значения "MALE", "FEMALE", "OTHER"
    "lifeStatus": "string", // Жизненный статус животного,
    // доступные значения "ALIVE"(устанавливается автоматически при добавлении
    // нового животного), "DEAD"(можно установить при обновлении информации о
    // животном)
    "chippingDateTime": "dateTime", // Дата и время чипирования в
    // формате ISO-8601 (устанавливается автоматически на момент добавления
    // животного)
    "chipperId": "int", // Идентификатор аккаунта с ролью
    // "CHIPPER" или "ADMIN"
    "chippingLocationId": "long", // Идентификатор точки локации
    // животных
    "visitedLocations": "[long]", // Массив идентификаторов объектов
    // с информацией о посещенных точках локаций
    "deathDateTime": "dateTime" // Дата и время смерти животного в
    // формате ISO-8601 (устанавливается автоматически при смене lifeStatus на
    // "DEAD"). Равняется null, пока lifeStatus = "ALIVE".
}
```

Условие	Статус
Запрос успешно выполнен	201
animalTypes = null, animalTypes.size() <= 0 Элемент массива animalTypes = null Элемент массива animalTypes <= 0 weight = null, weight <=0, length = null, length <=0, height = null, height <=0, gender = null, gender != "MALE", "FEMALE", "OTHER", chipperId = null, chipperId <=0, chippingLocationId = null, chippingLocationId <=0	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Тип животного не найден, Аккаунт с chipperId не найден, Точка локации с chippingLocationId не найдена	404

#### API 4: Обновление информации о животном

**PUT** - /animals/{animalId}

{animalId}: "long" // Идентификатор животного

- request

Body {

"weight": "float", // Масса животного, кг  
 "length": "float", // Длина животного, м  
 "height": "float", // Высота животного, м  
 "gender": "string", // Гендерный признак животного,

доступные значения "MALE", "FEMALE", "OTHER"

```

        "lifeStatus": "string", // Жизненный статус животного,
        // доступные значения "ALIVE"(устанавливается автоматически при добавлении
        // нового животного), "DEAD"(можно установить при обновлении информации о
        // животном)
        "chipperId": "int", // Идентификатор аккаунта с ролью
        "CHIPPER" или "ADMIN"
        "chippingLocationId": "long" // Идентификатор точки локации
        // ЖИВОТНЫХ
    }

```

- response

```

Body {
    "id": "long", // Идентификатор животного
    "animalTypes": "[long]", // Массив идентификаторов типов
    // ЖИВОТНОГО
    "weight": "float", // Масса животного, кг
    "length": "float", // Длина животного, м
    "height": "float", // Высота животного, м
    "gender": "string", // Гендерный признак животного,
    // доступные значения "MALE", "FEMALE", "OTHER"
    "lifeStatus": "string", // Жизненный статус животного,
    // доступные значения "ALIVE"(устанавливается автоматически при добавлении
    // нового животного), "DEAD"(можно установить при обновлении информации о
    // животном)
    "chippingDateTime": "dateTime", // Дата и время чипирования в
    // формате ISO-8601 (устанавливается автоматически на момент добавления
    // животного)
    "chipperId": "int", // Идентификатор аккаунта с ролью
    "CHIPPER" или "ADMIN"
    "chippingLocationId": "long", // Идентификатор точки локации
    // ЖИВОТНЫХ
    "visitedLocations": "[long]", // Массив идентификаторов объектов
    // с информацией о посещенных точках локаций
    "deathDateTime": "dateTime" // Дата и время смерти животного в
    // формате ISO-8601 (устанавливается автоматически при смене lifeStatus на
    // "DEAD"). Равняется null, пока lifeStatus = "ALIVE".
}

```



Условие	Статус
Запрос успешно выполнен	200
animalId = null, animalId <=0, weight = null weight <=0, length = null length <=0, height = null height <=0, gender != "MALE", "FEMALE", "OTHER", lifeStatus != "ALIVE", "DEAD", chipperId = null, chipperId <=0, chippingLocationId = null, chippingLocationId <=0 Установка lifeStatus = "ALIVE", если у животного lifeStatus = "DEAD" Новая точка чипирования совпадает с первой посещенной точкой локации	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Животное с animalId не найдено Аккаунт с chipperId не найден Точка локации с chippingLocationId не найдена	404

#### API 5: Удаление животного

**DELETE** - /animals/{animalId}

{animalId}: "long" // Идентификатор животного

- request

```
Body {
  empty
}
```

- response

```
Body {  
    empty  
}
```

Условие	Статус
Запрос успешно выполнен	200
animalId = null, animalId <=0 Животное покинуло локацию чипирования, при этом есть другие посещенные точки	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403
Животное с animalId не найдено	404

**API 6:** Добавление типа животного к животному

**POST** - /animals/{animalId}/types/{typeId}

{animalId}: "long"      // Идентификатор животного  
{typeId}: "long"      // Идентификатор типа животного

- request

```
Body {  
    empty  
}
```

- response

```
Body {  
    "id": "long",                      // Идентификатор животного  
    "animalTypes": "[long]",          // Массив идентификаторов типов  
    ЖИВОТНОГО  
    "weight": "float",                // Масса животного, кг  
    "length": "float",                // Длина животного, м  
    "height": "float",                // Высота животного, м
```

```

    "gender": "string", // Гендерный признак животного,
    доступные значения "MALE", "FEMALE", "OTHER"

    "lifeStatus": "string", // Жизненный статус животного,
    доступные значения "ALIVE"(устанавливается автоматически при добавлении
    нового животного), "DEAD"(можно установить при обновлении информации о
    животном)

    "chippingDateTime": "dateTime", // Дата и время чипирования в
    формате ISO-8601 (устанавливается автоматически на момент добавления
    животного)

    "chipperId": "int", // Идентификатор аккаунта с ролью
    "CHIPPER" или "ADMIN"

    "chippingLocationId": "long", // Идентификатор точки локации
    животных

    "visitedLocations": "[long]", // Массив идентификаторов объектов
    с информацией о посещенных точках локаций

    "deathDateTime": "dateTime" // Дата и время смерти животного в
    формате ISO-8601 (устанавливается автоматически при смене lifeStatus на
    "DEAD"). Равняется null, пока lifeStatus = "ALIVE".

}

```

Условие	Статус
Запрос успешно выполнен	201
animalId = null, animalId <= 0, typeId = null, typeId <= 0	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Животное с animalId не найдено Тип животного с typeId не найден	404

#### API 7: Изменение типа животного у животного

**PUT** - /animals/{animalId}/types

{animalId}: "long" // Идентификатор животного

- request

```
Body {  
    "oldTypeId": "long", // Идентификатор текущего типа  
    ЖИВОТНОГО  
    "newTypeId": "long" // Идентификатор нового типа  
    ЖИВОТНОГО для замены  
}
```

- response

```
Body {  
    "id": "long", // Идентификатор животного  
    "animalTypes": "[long]", // Массив идентификаторов типов  
    ЖИВОТНОГО  
    "weight": "float", // Масса животного, кг  
    "length": "float", // Длина животного, м  
    "height": "float", // Высота животного, м  
    "gender": "string", // Гендерный признак животного,  
    доступные значения "MALE", "FEMALE", "OTHER"  
    "lifeStatus": "string", // Жизненный статус животного,  
    доступные значения "ALIVE"(устанавливается автоматически при добавлении  
    нового животного), "DEAD"(можно установить при обновлении информации о  
    животном)  
    "chippingDateTime": "dateTime", // Дата и время чипирования в  
    формате ISO-8601 (устанавливается автоматически на момент добавления  
    животного)  
    "chipperId": "int", // Идентификатор аккаунта с ролью  
    "CHIPPER" или "ADMIN"  
    "chippingLocationId": "long", // Идентификатор точки локации  
    ЖИВОТНЫХ  
    "visitedLocations": "[long]", // Массив идентификаторов объектов  
    с информацией о посещенных точках локаций  
    "deathDateTime": "dateTime" // Дата и время смерти животного в  
    формате ISO-8601 (устанавливается автоматически при смене lifeStatus на  
    "DEAD"). Равняется null, пока lifeStatus = "ALIVE".  
}
```

Условие	Статус
Запрос успешно выполнен	200
animalId = null, animalId <= 0, oldTypeId = null, oldTypeId <= 0, newTypeId = null, newTypeId <= 0	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Животное с animalId не найдено Тип животного с oldTypeId не найден Тип животного с newTypeId не найден Типа животного с oldTypeId нет у животного с animalId	404
Тип животного с newTypeId уже есть у животного с animalId Животное с animalId уже имеет типы с oldTypeId и newTypeId	409

#### API 8: Удаление типа животного у животного

##### DELETE - /animals/{animalId}/types/{typeId}

{animalId}: "long" // Идентификатор животного  
 {typeId}: "long" // Идентификатор типа животного

- request

```
Body {
  empty
}
```

- response

```
Body {
  "id": "long", // Идентификатор животного
  "animalTypes": "[long]", // Массив идентификаторов типов
  // Животного
  "weight": "float", // Масса животного, кг
  "length": "float", // Длина животного, м
}
```

```

    "height": "float",                // Высота животного, м
    "gender": "string",               // Гендерный признак животного,
    // доступные значения "MALE", "FEMALE", "OTHER"
    "lifeStatus": "string",           // Жизненный статус животного,
    // доступные значения "ALIVE"(устанавливается автоматически при добавлении
    // нового животного), "DEAD"(можно установить при обновлении информации о
    // животном)
    "chippingDateTime": "dateTime",   // Дата и время чипирования в
    // формате ISO-8601 (устанавливается автоматически на момент добавления
    // животного)
    "chipperId": "int",               // Идентификатор аккаунта с ролью
    // "CHIPPER" или "ADMIN"
    "chippingLocationId": "long",     // Идентификатор точки локаций
    // животных
    "visitedLocations": "[long]",      // Массив идентификаторов объектов
    // с информацией о посещенных точках локаций
    "deathDateTime": "dateTime"       // Дата и время смерти животного в
    // формате ISO-8601 (устанавливается автоматически при смене lifeStatus на
    // "DEAD"). Равняется null, пока lifeStatus = "ALIVE".
}

```

Условие	Статус
Запрос успешно выполнен	200
animalId = null, animalId <= 0, typeId = null, typeId <= 0 У животного только один тип и это тип с typeId	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Животное с animalId не найдено Тип животного с typeId не найден У животного с animalId нет типа с typeId	404

## 7) Точка локации, посещенная животным

**API 1:** Просмотр точек локации, посещенных животным

**GET** - /animals/{animalId}/locations

?startDateTime=

&endDateTime=

&from={from}

&size={size}

{animalId}: "long" // Идентификатор животного

{startDateTime}: "dateTime" // Дата и время, не раньше которых нужно искать

посещенные точки локации животных в формате ISO-8601, если null - не учитывается

{endDateTime}: "dateTime" // Дата и время, не позже которых нужно искать

посещенные точки локации животных в формате ISO-8601, если null - не учитывается

{from}: "int" // Количество элементов, которое необходимо пропустить для формирования страницы с результатами (по умолчанию 0)

{size}: "int" // Количество элементов на странице (по умолчанию 10)

- request

```
Body {  
    empty  
}
```

- response

```
Body [  
    {  
        "id": "long", // Идентификатор  
        // объекта с информацией о посещенной точке локации  
        "dateTimeOfVisitLocationPoint": "dateTime", // Дата и время  
        // посещения животным точки локации в формате ISO-8601  
        "locationPointId": "long" // Идентификатор  
        // посещенной точки локации  
    }  
]
```

**Результаты поиска сортируются по дате посещения точки от ранней к поздней**

Условие	Статус
Запрос успешно выполнен	200
animalId = null, animalId <= 0, from < 0, size <= 0, startDateTime - не в формате ISO-8601, endDateTime - не в формате ISO-8601	400
Запрос от неавторизованного аккаунта	401
Животное с animalId не найдено	404

**API 2:** Добавление точки локации, посещенной животным

**POST** - /animals/{animalId}/locations/{pointId}

{animalId}: "long" // Идентификатор животного

{pointId}: "long" // Идентификатор точки локации

- request

```
Body {
    empty
}
```

- response

```
Body {
    "id": "long", // Идентификатор
    // объекта с информацией о посещенной точке локации
    "dateTimeOfVisitLocationPoint": "dateTime", // Дата и время
    // посещения животным точки локации в формате ISO-8601
    "locationPointId": "long" // Идентификатор
    // посещенной точки локации
}
```



Условие	Статус
Запрос успешно выполнен	201
animalId = null, animalId <= 0, pointId = null, pointId <= 0, У животного lifeStatus = "DEAD" Животное находится в точке чипирования и никуда не перемещалось, попытка добавить точку локации, равную точке чипирования. Попытка добавить точку локации, в которой уже находится животное	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Животное с animalId не найдено Точка локации с pointId не найдена	404

### API 3: Изменение точки локации, посещенной животным

**PUT** - /animals/{animalId}/locations

{animalId}: "long" // Идентификатор животного

- request

```

Body {
    "visitedLocationPointId": "long",    // Идентификатор объекта с
    информация о посещенной точке локации
    "locationPointId": "long"           // Идентификатор посещенной точки
    локации
}
```

- response

```

Body {
    "id": "long",                       // Идентификатор
    объекта с информацией о посещенной точке локации
    "dateTimeOfVisitLocationPoint": "dateTime", // Дата и время
    посещения животным точки локации в формате ISO-8601
    "locationPointId": "long"           // Идентификатор
    посещенной точки локации
}
```

}

Условие	Статус
Запрос успешно выполнен	200
animalId = null, animalId <= 0, visitedLocationPointId = null, visitedLocationPointId <= 0, locationPointId = null, locationPointId <= 0 Обновление первой посещенной точки на точку чипирования Обновление точки на такую же точку Обновление точки локации на точку, совпадающую со следующей и/или с предыдущей точками	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN" или "CHIPPER"	403
Животное с animalId не найдено Объект с информацией о посещенной точке локации с visitedLocationPointId не найден. У животного нет объекта с информацией о посещенной точке локации с visitedLocationPointId. Точка локации с locationPointId не найден	404

#### API 4: Удаление точки локации, посещенной животным

**DELETE** - /animals/{animalId}/locations/{visitedPointId}

{animalId}: "long" // Идентификатор животного

{visitedPointId}: "long" // Идентификатор объекта с информацией о посещенной  
точке локации

- request

```
Body {
    empty
}
```

- response

```

Body {
    empty
}

```

Условие	Статус
Запрос успешно выполнен (Если удаляется первая посещенная точка локации, а вторая точка совпадает с точкой чипирования, то она удаляется автоматически)	200
animalId = null, animalId <= 0 visitedPointId = null, visitedPointId <= 0	400
Запрос от неавторизованного аккаунта	401
У аккаунта нет роли "ADMIN"	403
Животное с animalId не найдено Объект с информацией о посещенной точке локации с visitedPointId не найден. У животного нет объекта с информацией о посещенной точке локации с visitedPointId	404

## 8) Аналитика по зонам

**API 1:** Просмотр информации о перемещениях животных в зоне

**GET** - /areas/{areald}/analytics

?startDate=

&endDate=

{areald}: "long", // Идентификатор зоны

{startDate}: "date", // Дата начала интервала в формате ISO-8601 (pattern "yyyy-MM-dd")

{endDate}: "date" // Дата конца интервала в формате ISO-8601 (pattern "yyyy-MM-dd")

- request

```

Body {
    empty
}

```

```

    }

- response
  Body {
    "totalQuantityAnimals": "long",          // Общее количество животных,
    находящиеся в этой зоне в указанный интервал времени
    "totalAnimalsArrived": "long",          // Общее количество
    посещений зоны в указанный интервал времени
    "totalAnimalsGone": "long",             // Общее количество выходов
    из зоны в указанный интервал времени
    "animalsAnalytics": [
      {
        "animalType": "string",             // Тип животных
        "animalTypeId": "long",             // Идентификатор
        типа животных
        "quantityAnimals": "long",          // Количество
        животных данного типа, находящихся в этой зоне в
        указанный интервал времени
        "animalsArrived": "long",           // Количество
        животных данного типа, прибывших в эту зону в
        указанный интервал времени
        "animalsGone": "long",              // Количество
        животных данного типа, покинувших эту зону в
        указанный интервал времени
      }
    ]
  }
}

```

**ВНИМАНИЕ!** Одно и то же животное может несколько раз посетить и покинуть зону в указанный период. Для каждого животного учитывается только один вход в зону и один выход из нее.

Условие	Статус
Запрос успешно выполнен	200
areald = null, areald <= 0, startDate - не в формате ISO-8601 (pattern "yyyy-MM-dd") endDate - не в формате ISO-8601 (pattern "yyyy-MM-dd")	400

startDate >= endDate	
Запрос от неавторизованного аккаунта	401
Зона с areald не найдена	404