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

Мы рады приветствовать Вас на конкурсе «Прикладное программирование if...else»! В рамках первого этапа конкурса Вам нужно реализовать RESTful API сервис на одном из следующих языков программирования: C#, Java, Python, PHP или Go.

Функциональность будет описана ниже в виде краткого технического задания (см. разделы “[Легенда](#_22kr0fd6hks7)”, “[Функционал системы](#_ocb69gudum89)”, “[API Declarations](#_deowru3f3qab)”).

Выполненное задание необходимо представить на оценку экспертам вместе с исходным кодом программы (см. разделы “[Отправка задания](#_v2fpifresrx0)”, “[Работа с Docker](#_jr1hdum1gbqb)”).

Желаем Вам удачи!

**Ответы на часто задаваемые вопросы** - [**FAQ**](https://docs.google.com/document/d/1xA3XHhjNK_RWL1M8hLpyQ1DN13BNeqMGMlMDtZFzNEI)

## Оглавление

[**Легенда**](#_22kr0fd6hks7) **3**

[**Функционал системы**](#_ocb69gudum89) **4**

[В системе должны быть следующие компоненты](#_ex7ls79nerou) 4

[В контроллерах должен быть доступен следующий функционал](#_jesnalowkths) 4

[**Задание**](#_8j5xqet5nquh) **5**

[**Этапы задания**](#_a3r666ctzty) **6**

[Нулевой этап](#_plx32t64hw1j) 6

[Первый этап](#_30m2p3s35rf2) 6

[Второй этап](#_qlht5xw3ve2y) 6

[**Тестирование задания**](#_ny0mm0rlsk4q) **7**

[**Отправка задания**](#_v2fpifresrx0) **8**

[**Работа с Docker**](#_jr1hdum1gbqb) **9**

[Сохранение образа контейнера](#_f0sk0oe3a705) 9

[Пример файла docker-compose.yml](#_chsht51o97u) 9

[**API Declarations**](#_deowru3f3qab) **10**

[1) Аутентификация пользователя](#_vksquxfq0xrt) 10

[API 1: Регистрация нового аккаунта](#_4gg9ll6erfgb) 10

[2) Аккаунт пользователя](#_j70bwi65ckjq) 11

[API 1: Получение информации об аккаунте пользователя](#_hfqxik454k6q) 11

[API 2: Поиск аккаунтов пользователей по параметрам](#_2u8obmwm4kdj) 11

[API 3: Обновление данных аккаунта пользователя](#_6t96emk15cpo) 12

[API 4: Удаление аккаунта пользователя](#_nawpe8hwwfq4) 14

[3) Точка локации животных](#_pnkxujeqfih2) 15

[API 1: Получение информации о точке локации животных](#_axtzf1ydsk6n) 15

[API 2: Добавление точки локации животных](#_xyigtah614qj) 15

[API 3: Изменение точки локации животных](#_kvbww5uspbyc) 16

[API 4: Удаление точки локации животных](#_tvmjumjaysdu) 17

[4) Типы животных](#_9e3j4jqgmo9u) 19

[API 1: Получение информации о типе животного](#_nf5r23r2leo7) 19

[API 2: Добавление типа животного](#_lgtivjyt7hlh) 19

[API 3: Изменение типа животного](#_b2pkzqjmfywo) 20

[API 4: Удаление типа животного](#_wis5keji5egr) 21

[5) Животное](#_zalvqgmbiuu) 22

[API 1: Получение информации о животном](#_fwi1635gdx66) 22

[API 2: Поиск животных по параметрам](#_vx1dtceip1y3) 23

[API 3: Добавление нового животного](#_82499pp3josr) 25

[API 4: Обновление информации о животном](#_h9pz1iri9ati) 27

[API 5: Удаление животного](#_68wf3jwx2jb2) 29

[API 6: Добавление типа животного к животному](#_e7ne70wyimwx) 30

[API 7: Изменение типа животного у животного](#_f8q5x2ixxg5a) 31

[API 8: Удаление типа животного у животного](#_q49y2yti7ths) 33

[6) Точка локации, посещенная животным](#_k5726z48y6ob) 35

[API 1: Просмотр точек локации, посещенных животным](#_78dmb2ha0ixt) 35

[API 2: Добавление точки локации, посещенной животным](#_mcfue0klq7) 36

[API 3: Изменение точки локации, посещенной животным](#_tketn5u1cwu7) 37

[API 4: Удаление точки локации, посещенной животным](#_8ljjok4w902b) 38

## 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
* Animal
* Animal Type
* Location Point
* Animal Visited Location

### 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

## Задание

1. Реализовать один, несколько или все этапы задания *(см. разделы “*[*Этапы задания*](#_a3r666ctzty)*” и “*[*API Declarations*](#_deowru3f3qab)*”)*
2. Настроить Docker *(см. раздел “*[*Работа с Docker*](#_jr1hdum1gbqb)*”)*
3. Проверить верность своего решения *(см. раздел “*[*Тестирование задания*](#_ny0mm0rlsk4q)*”)*
4. Отправить нам свое решение *(см. раздел “*[*Отправка задания*](#_v2fpifresrx0)*”)*

## Этапы задания

В зависимости от своего уровня подготовки, вы можете реализовать один, несколько или все этапы задания. Приложение должно обязательно иметь базу данных.  
Пример баз данных: MySQL, MsSQL, Postgres, Mongo, SQLite и другие.

### Нулевой этап

Необходимо реализовать следующие методы:

* ***GET /accounts/{accountId}*** - Просмотр информации об аккаунте
* ***GET /accounts/search*** - Поиск аккаунта
* ***GET /animals/{animalId*}** - Просмотр информации о животном
* ***GET /animals/search*** - Поиск животного
* ***GET /animals/types/{typeId}*** - Просмотр информации о типе животного
* ***GET /locations/{pointId}***- Просмотр информации о точке локации животных
* ***GET /animals/{animalId}/locations*** - Просмотр информации о перемещении животного

Подробное описание методов представлено ниже в [**API Declarations**](#_deowru3f3qab).

Тесты будут проверять любые валидные возвращаемые данные в формате JSON, то есть главное - это соответствие **типов данных**, а не их содержимое.

### Первый этап

Необходимо реализовать следующие методы:

* ***POST /registration*** - Регистрация аккаунта

Подробное описание метода представлено ниже в [**API Declarations**](#_deowru3f3qab).

Все реализованные на нулевом уровне методы должны также работать для авторизованных пользователей.

### Второй этап

Реализовать все оставшиеся методы согласно [**API Declarations**](#_deowru3f3qab).

При отправке запросов на первом и втором уровнях требуется авторизация, в **Header “Authorization”** записывается слово **“Basic”**, далее через пробел записывается логин(email) и пароль зарегистрированного аккаунта после кодировки **Base64** в формате **login:password**.

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

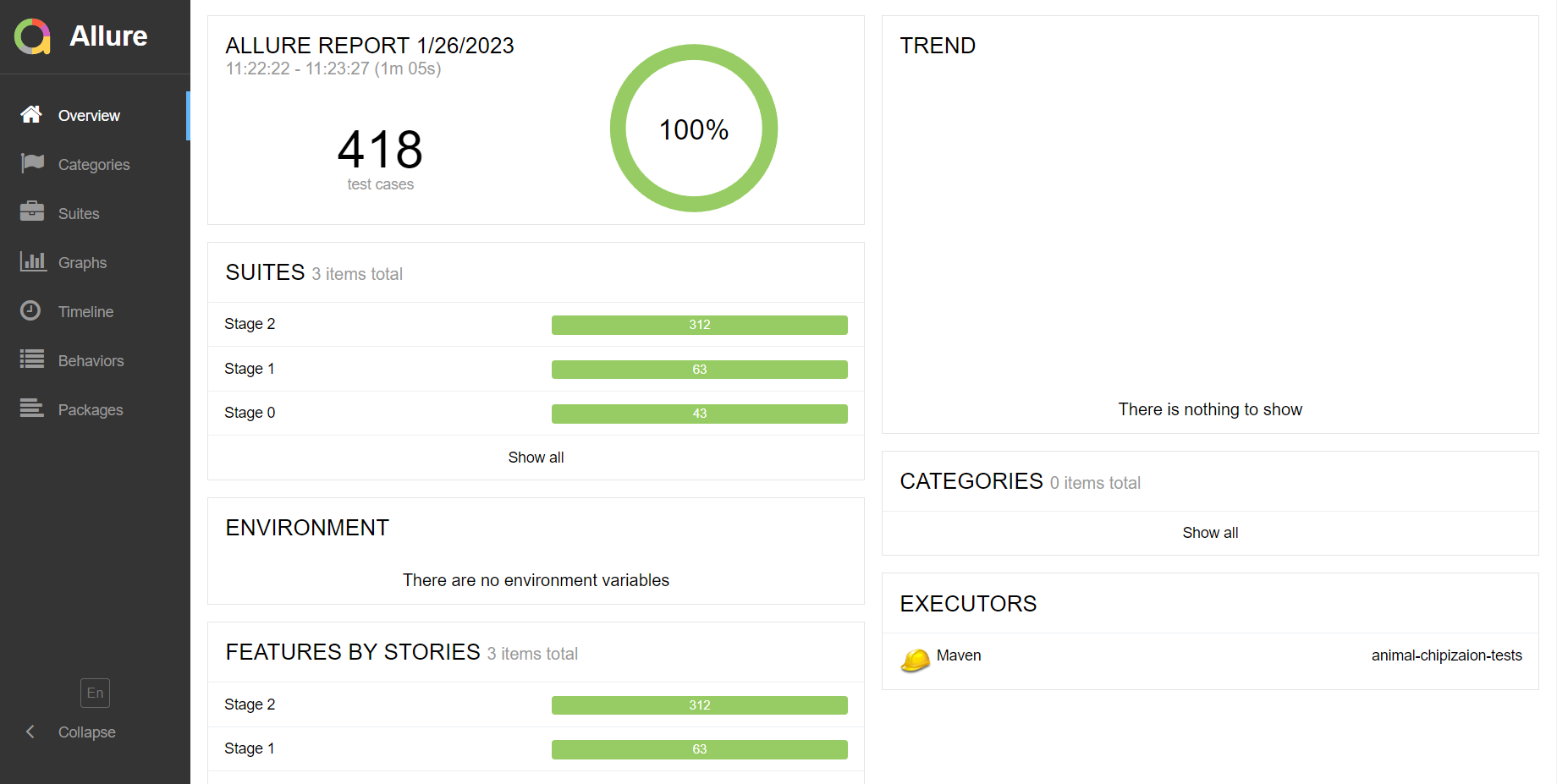
## Тестирование задания

Для проверки работоспособности своего приложения вам необходимо использовать **Docker-Image** из [**Docker-Hub**](https://hub.docker.com/r/mrexpen/planet_olymp_phase1). 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**](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 ваше задание соответствует требованиям, которые описаны ниже в [**API Declarations**](#_deowru3f3qab). При выполнении всего задания у вас будет 100%-ное прохождение тестов, как видно на примере выше.

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**](https://hub.docker.com/r/mrexpen/planet_olymp_phase1).

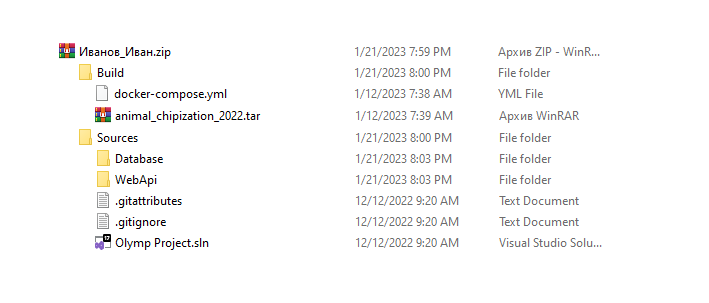
Пример с полным файлом **docker-compose.yml** вы можете просмотреть ниже.

## Отправка задания

Для отправки задания 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**](#_jr1hdum1gbqb).
4. После выполнения предыдущих пунктов вы должны загрузить ваш архив на **Яндекс-Диск** или **Google Disk**.
5. Отправьте ссылку на архив задания в личном кабинете участника.

Пример архива с папками:



Перед отправкой задания **обязательно** проведите тестирование приложения, используя актуальную версию **Docker-Image** из [**Docker-Hub**](https://hub.docker.com/r/mrexpen/planet_olymp_phase1).

**ВНИМАНИЕ! Все задания проверяются автоматически, при ошибках в названии файлов или папок ваше задание может некорректно обрабатываться, и это будет означать не верное решение вашего задания.**

## 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\_phase1  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 или 2 - 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

}

| 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 |
| 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

}

| Условие | Статус |
| --- | --- |
| Запрос успешно выполнен | 200 |
| accountId = null,  accountId <= 0 | 400 |
| Неверные авторизационные данные | 401 |
| Аккаунт с таким accountId не найден | 404 |

##### **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 участвует в фильтрации

{firstName}: "string", // Фамилия пользователя, может использоваться только часть фамилии без учета регистра, если null, не участвует в фильтрации

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

{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

}

]

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

| Условие | Статус |
| --- | --- |
| Запрос успешно выполнен | 200 |
| from < 0,  size <= 0 | 400 |
| Неверные авторизационные данные | 401 |

##### **API 3:** 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" // Пароль от аккаунта

}

- response

Body {

"id": "int", // Идентификатор аккаунта пользователя

"firstName": "string", // Новое имя пользователя

"lastName": "string", // Новая фамилия пользователя

"email": "string" // Новый адрес электронной почты

}

| 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 | 400 |
| Запрос от неавторизованного аккаунта  Неверные авторизационные данные | 401 |
| Обновление не своего аккаунта  Аккаунт не найден | 403 |
| Аккаунт с таким email уже существует | 409 |

##### 

##### 

##### **API 4:** 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  Неверные авторизационные данные | 401 |
| Удаление не своего аккаунта  Аккаунт с таким accountId не найден | 403 |

### 

### 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 |
| Неверные авторизационные данные | 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  Неверные авторизационные данные | 401 |
| 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,  latitude < -90,  latitude > 90,  longitude = null,  longitude < -180,  longitude > 180 | 400 |
| Запрос от неавторизованного аккаунта  Неверные авторизационные данные | 401 |
| 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  Неверные авторизационные данные | 401 |
| Точка локации с таким pointId не найдена | 404 |

### 

### 4) 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 |
| Неверные авторизационные данные | 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  Неверные авторизационные данные | 401 |
| 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 <= 0,  typeId = null,  type = null,  type = "" или состоит из пробелов | 400 |
| Запрос от неавторизованного аккаунта  Неверные авторизационные данные | 401 |
| 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  Неверные авторизационные данные | 401 |
| Тип животного с таким typeId не найден | 404 |

### 

### 5) 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 пользователя, чипировавшего животное

"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 |
| Неверные авторизационные данные | 401 |
| Животное с animalId не найдено | 404 |

##### **API 2:** Поиск животных по параметрам

**GET** -/animals/search?

startDateTime={startDateTime}

&endDateTime={endDateTime}

&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 участвует в фильтрации

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

{chipperId}: "int", // Идентификатор аккаунта пользователя, чипировавшего животное, если null, не участвует в фильтрации

{chippingLocationId}: "long", // Идентификатор точки локации животных, если null, не участвует в фильтрации

{lifeStatus}: "string", // Жизненный статус животного, если null, не участвует в фильтрации

{gender}: "string", // Гендерная принадлежность животного, если null, не участвует в фильтрации

{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 пользователя, чипировавшего животное

"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 |
| Неверные авторизационные данные | 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 пользователя, чипировавшего животное

"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 пользователя, чипировавшего животное

"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  Неверные авторизационные данные | 401 |
| Тип животного не найден,  Аккаунт с chipperId не найден,  Точка локации с chippingLocationId не найдена | 404 |
| Массив animalTypes содержит дубликаты | 409 |

##### **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 пользователя, чипировавшего животное

"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 пользователя, чипировавшего животное

"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  Неверные авторизационные данные | 401 |
| 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  Неверные авторизационные данные | 401 |
| 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 пользователя, чипировавшего животное

"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  Неверные авторизационные данные | 401 |
| Животное с animalId не найдено  Тип животного с typeId не найден | 404 |
| Тип животного с typeId уже есть у животного с animalId | 409 |

##### **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 пользователя, чипировавшего животное

"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  Неверные авторизационные данные | 401 |
| 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 пользователя, чипировавшего животное

"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,  typeId = null,  typeId <= 0  У животного только один тип и это тип с typeId | 400 |
| Запрос от неавторизованного аккаунта  Неверные авторизационные данные | 401 |
| Животное с animalId не найдено  Тип животного с typeId не найден  У животного с animalId нет типа с typeId | 404 |

### 

### 6) 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 участвует в фильтрации

{endDateTime}: "dateTime" // Дата и время, не позже которых нужно искать посещенные точки локации животных в формате ISO-8601, если null, не участвует в фильтрации

{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 |
| Неверные авторизационные данные | 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  Неверные авторизационные данные | 401 |
| 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 информацией 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  Неверные авторизационные данные | 401 |
| 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  Неверные авторизационные данные | 401 |
| 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 |