# **Документация REST API для системы управления проектом**

**Аутентификация**

Все запросы (кроме GET) требуют заголовка авторизации:

Authorization: <api\_token>

Пример модели данных в json формате:

{

"id": "string",

"name": "string",

"description": "string",

"addedAt": "ISO8601 DateTime",

"rating": "number (1-5)",

"image": "string (URL)",

"location": "string",

"coordinates": {

"lat": "number",

"lng": "number"

},

"mapLink": "string (URL)",

"status": "string (planned|visited)"

}

## Эндпоинты

### ***1. Получение списка достопримечательностей***

**GET** /attractions

Параметры:

* hideVisited (boolean) - скрыть посещенные
* sortBy (string) - поле для сортировки (name, rating, addedAt)
* sortDirection (asc|desc) - направление сортировки
* search (string) - поиск по названию и описанию

Пример ответа в json :

{

"success": true,

"data": [

{

"id": "1",

"name": "Эйфелева башня",

"status": "planned",

...

}

],

"total": 25

}

### ***2. Создание достопримечательности***

**POST** /attractions

Тело запроса в json:

{

"name": "Название",

"description": "Описание",

"location": "Адрес",

"coordinates": {

"lat": 48.8584,

"lng": 2.2945

}

}

Обязательные поля: name, location

### ***3. Получение одной достопримечательности***

**GET** /attractions/:id

Пример ответа в json:

{

"success": true,

"data": {

"id": "1",

"name": "Эйфелева башня",

...

}

}

### ***4. Обновление достопримечательности***

**PUT** /attractions/:id

Тело запроса (только изменяемые поля через Администратора) в json:

{

"status": "visited",

"rating": 5

}

### ***5. Удаление достопримечательности***

**DELETE** /attractions/:id

Ответ в json:

{

"success": true,

"message": "Attraction deleted"

}

### ***6. Управление избранным***

**POST** /attractions/:id/favorite - Добавить в избранное  
**DELETE** /attractions/:id/favorite - Удалить из избранного

### ***7. Получение избранных достопримечательностей***

**GET** /favorites

Ответ аналогичен GET /attractions, но только для избранных

## Примеры использования с компонентом api.ts(лежит в task-project\src\components\Utils\api.ts)

### **Получение списка:**

const loadAttractions = async () => {

const response = await fetchAttractions();

if (response.success) {

console.log(response.data);

}

};

### **Создание:**

const newAttraction = {

name: "Колизей",

location: "Рим, Италия",

coordinates: { lat: 41.8902, lng: 12.4924 }

};

saveAttraction(newAttraction)

.then(response => {

if (response.success) {

console.log('Created:', response.data);

}

});

### Для обработки ошибок через try/catch:

try {

const response = await deleteAttraction('123');

if (!response.success) {

throw new Error(response.message);

}

} catch (error) {

console.error('Delete failed:', error);

}

## Коды ответов от предполагаемого сервера

* 200 OK - успешный запрос
* 201 Created - успешное создание
* 400 Bad Request - неверные параметры
* 401 Unauthorized - требуется авторизация
* 404 Not Found - ресурс не найден
* 500 Internal Server Error - серверная ошибка

## Пагинация (можно сделать бесконечную погрузку по лимитам)

Для эндпоинтов возвращающих списки:

* limit (number) - количество элементов (по умолчанию до 20)
* offset (number) - смещение (по умолчанию от 0)

Ответ содержит:

{

"data": [],

"total": 100,

"limit": 20,

"offset": 0

}