**Страница заявки(статическая) на помощь: GET-запрос к бэку для получения статистики о кол-ве тонн продуктов и семей, FAQ и фотоотчета**

**Взаимодействие сервера и БД**

/help\_request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| faq | question | faq | faq\_question |  |
| answer | faq\_answer |  |
| photo\_report | image\_url | photos\_in\_reports | phrep\_path | Ссылка на путь в файловом хранилище |
| image\_name | phrep\_id |  |
| statistics | total\_products | help\_stats | products\_count |  |
| total\_families | family\_count |  |

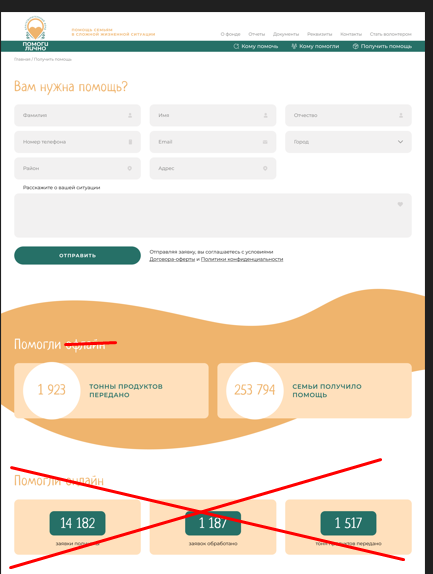
**Backend:**

На стороне бэка для метода /help\_request при запросе к БД нужно добавить условие фильтрации по полю type в таблице faq, где type = Заявка на помощь, пример:  
SELECT id, question, answer

FROM faq

WHERE type = 'Заявка на помощь';

**Frontend:**





**Статические элементы, нужно захардкодить на фронте:**

• Текст «Вам нужна помощь?», «Мы уже помогли» вместо «Помогли»

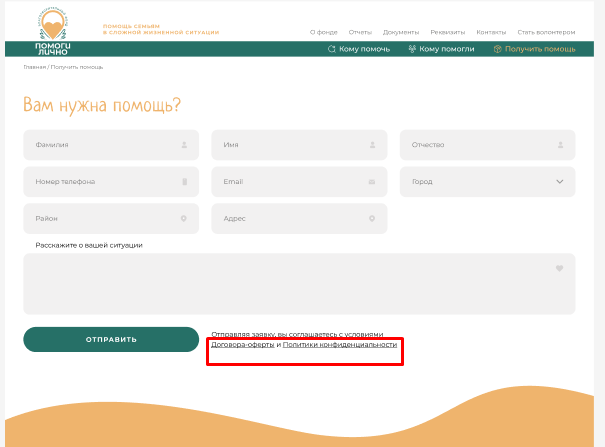
«Часто задаваемые вопросы», «Фотоотчет»

**GET** /help\_request

**Создание заявки на помощь: POST-запрос к бэку + отправка заявки на почту фонда: API-запрос к сервису рассылки**

/applications — метод создания заявки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| ApplicationRequest | surname | pers\_help\_request | hreq\_surname |  |
| name | hreq\_name |  |
| patname | hreq\_patname |  |
| phone | hreq\_phone |  |
| email | hreq\_email |  |
| city | hreq\_city |  |
| fname | hreq\_district |  |
| description | hreq\_description |  |



**Frontend:**

Блок «Отправляя заявку, вы соглашаетесь с условиями Договора-оферты и Политики конфиденциальности» захардкодить на фронте.

Сделать словосочетания «Договора-оферты» и «Политики конфиденциальности» гиперссылками, при нажатии открываются ссылки. Ссылки предоставит заказчик.

**Соответствие атрибутов API элементам интерфейса:**

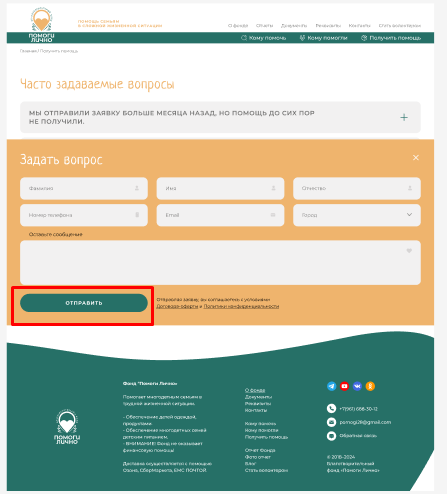
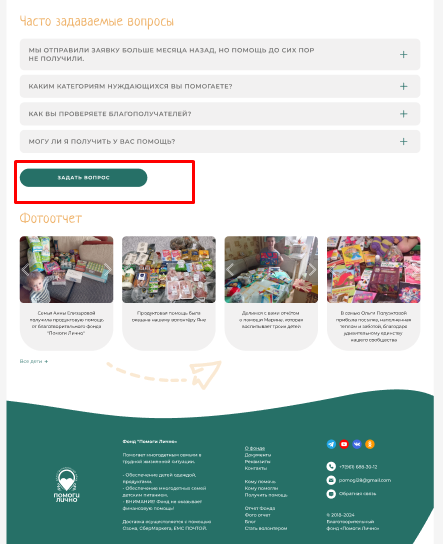
|  |  |  |
| --- | --- | --- |
| **Элемент интерфейса** | **Атрибут** | **Комментарий** |
| Фамилия | surname |  |
| Имя | name |  |
| Отчество | patname |  |
| Номер телефона | phone |  |
| email | email |  |
| город | city |  |
| район | fname |  |
| Описание | description |  |

**Создание вопроса: POST-запрос к бэку + отправка вопроса на почту фонда: API-запрос к сервису рассылки**

/support/questions — метод создания вопроса фонду

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| ApplicationRequest | surname | questions | q\_surname |  |
| name | q\_name |  |
| patname | q\_patname |  |
| phone | q\_phone |  |
| email | q\_email |  |
| city | q\_city |  |
| description | q\_text |  |

**Frontend:**

Заголовок «Задать вопрос» хардкодим на фронте.

**Загрузка страницы отчетов: GET-запрос к бэку списка отчетов**

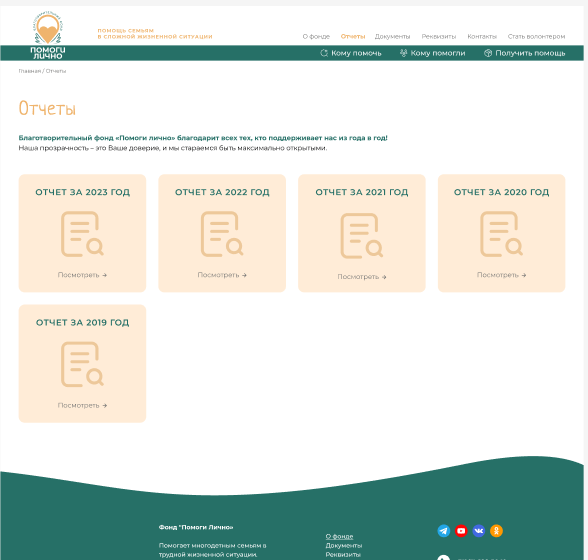
"/reports" — метод получения списка отчетов

"/reports/{id}/download" — метод для скачивания отчета

ReportList – массив отчетов Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| Report | id | foundation\_reports | q\_surname |  |
| name | q\_name |  |
| download\_url | q\_email |  |

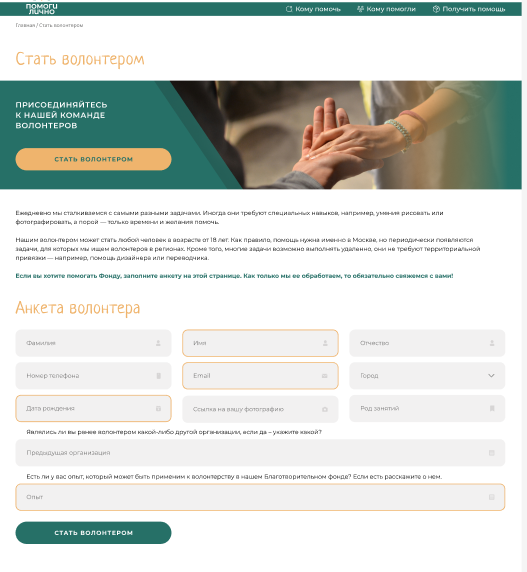
**Frontend:**

Заголовок «Отчеты» и текст «Благотворительный фонд «Помоги лично» благодарит всех тех, кто поддерживает нас из года в год!  
Наша прозрачность – это Ваше доверие, и мы стараемся быть максимально открытыми.» хардкодим на фронте.

**Создание заявки волонтера: POST-запрос к бэку + отправка заявки на почту фонда: API-запрос к сервису рассылки**

/volunteer/request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| VolunteerRequest | surname | volunteer\_requests | vreq\_surname |  |
| name | vreq\_name |  |
| patname | vreq\_patname |  |
| phone | vreq\_phone |  |
| email | vreq\_email |  |
| city | vreq\_city |  |
| bday | vreq\_bday |  |
| photo\_link | vreq\_photo\_link |  |
| occupation | vreq\_occupation |  |
| organization | vreq\_organization |  |
| experience | vreq\_experience |  |

**Frontend:**

Заголовок «Анкета волонтера» - хардкодим на фронте.

Блок «Стать волонтером» убираем. Оставляем только текст: «Присоединяйтесь к нашей команде волонтеров».

Блок ниже тоже хардкодим на фронте:

Стать волонтером

Ежедневно мы сталкиваемся с самыми разными задачами. Иногда они требуют специальных навыков, например, умения рисовать или   
фотографировать, а порой — только времени и желания помочь.   
  
Нашим волонтером может стать любой человек в возрасте от 18 лет. Как правило, помощь нужна именно в Москве, но периодически появляются   
задачи, для которых мы ищем волонтеров в регионах. Кроме того, многие задачи возможно выполнять удаленно, они не требуют территориальной   
привязки — например, помощь дизайнера или переводчика.   
  
Если вы хотите помогать Фонду, заполните анкету на этой странице. Как только мы ее обработаем, то обязательно свяжемся с вами!

**Загрузка страницы заявки волонтера: GET-запрос к бэку для получения FAQ**

/volunteer/faq

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| faq | question | faq | faq\_question |  |
| answer | faq\_answer |  |

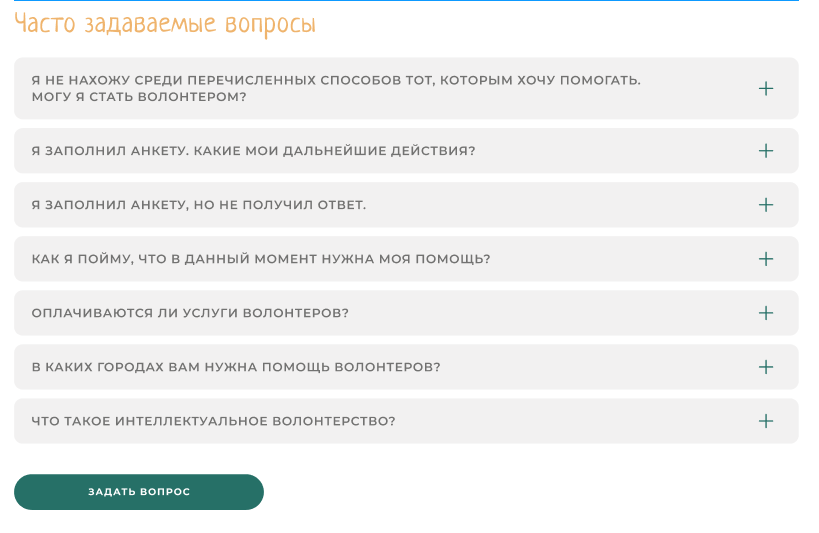
На стороне бэка дополнительно фильтровать вопросы по типу Волонтерство, пример:  
SELECT id, question, answer

FROM faq\_type

WHERE type = 'Волонтерство';

**Frontend:**

Заголовок «Часто задаваемые вопросы» захардкодить на фронте.

**Загрузка страницы блога: GET-запрос к бэку списка статей**

**Загрузка следующего списка статей по нажатию на кнопки пагинации: GET-запрос к бэку**

/blog/articles — метод загрузки страницы блога

BlogArticle — объект с данными о статье

BlogArticleList — массив с объектами статей и параметрами пагинации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| BlogArticle | id | blog | article\_id |  |
| title | article\_name |  |
|  | image\_url |  | article\_publish\_date |  |
| published\_at | article\_photo\_path |  |
| pagination | current\_page | - | - | = значению параметра “page” в запросе |
| per\_page | - | = значению параметра “per\_page” в запросе |
| total | - | Общее кол-во записей в таблице video\_reports |
| last\_page | - | Номер по  Вычисляется как total/per\_page, округленное к большему |

page: от 1 до 999

per\_page: от 1 до 10

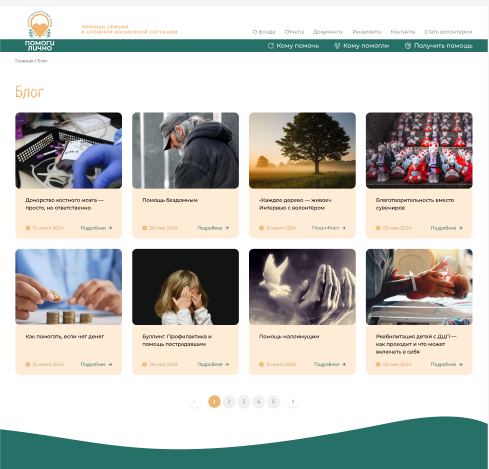
Подробное описание пагинации приведено в методе **GET** /api/video.

При нажатии на пагинацию (например, переход на следующую страницу), клиент отправляет GET-запрос на тот же эндпоинт /blog/articles, но с обновленными параметрами page и per\_page. Сервер возвращает новый набор статей и обновленную информацию о пагинации.

Пример запроса GET /blog/articles?page=2&per\_page=8

**Frontend:**

Заголовок «Блог» захардкодить на странице

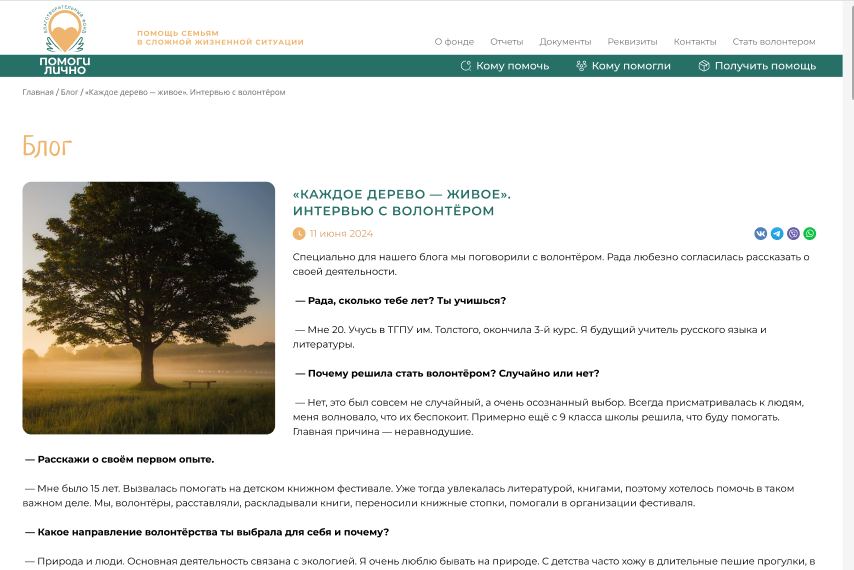


**Загрузка страницы статьи: GET-запрос к бэку для получения статьи и списка комментариев**

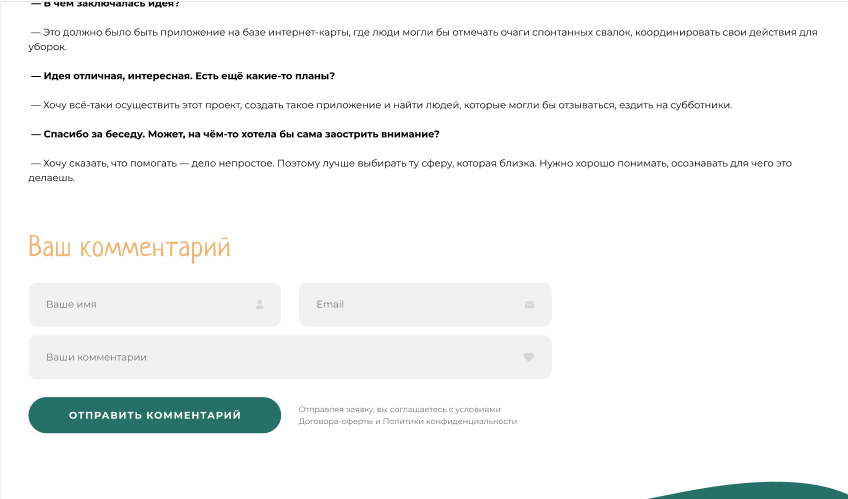
/articles/{articleId} — метод получения статьи со списком комментариев

В качестве ответа получаем объект ArticleWithCommentsResponse — массив, состоящий из статьи Article и списка комментариев Comment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| Article | id | blog | article\_id |  |
| title | article\_name |  |
| text | article\_text |  |
| image\_url | article\_publish\_date |  |
| published\_at | article\_photo\_path |  |
| Comment | id | blog\_comments | bcomm\_id |  |
| article\_id | article\_id |  |
| name | bcomm\_name |  |
| email | bcomm\_email |  |
| text | bcomm\_text |  |
| created\_at | bcomm\_publish\_date |  |

**Frontend:**

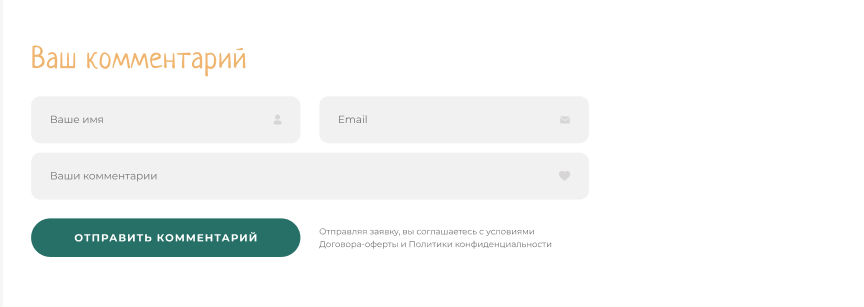
Список комментариев:



**Добавление комментария к статье: POST-запрос к бэку**

articles/comment – метод для добавления комментария

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| CommentResponse | article\_id | blog\_comments | bcomm\_id |  |
| text | bcomm\_text |  |
| email | bcomm\_email |  |
| name | bcomm\_name |  |
| created\_at | bcomm\_publish\_date |  |
| AddCommentRequest | article\_id | blog\_comments | bcomm\_id |  |
| text | bcomm\_text |  |
| email | bcomm\_email |  |
| name | bcomm\_name |  |

Заголовок «Ваш комментарий» захардкодить на фронте.

**Проставление лайка комментарию: POST-запрос к бэку**

articles/comment/{commentId}/like — добавление лайка

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| LikeResponse | comment\_id | blog\_comments | bcomm\_id |  |
| likes\_count | bcomm\_likes |  |

Страница «Кому помогли»: GET-запрос к бэку списка фото

Пример запроса GET /blog/articles?page=1&per\_page=20

**Загрузка следующего списка фото по нажатию на кнопки пагинации: GET-запрос к бэку**

/helped — метод загрузки страницы «Кому помогли»

HelpedItem — объект с данными о фотографии

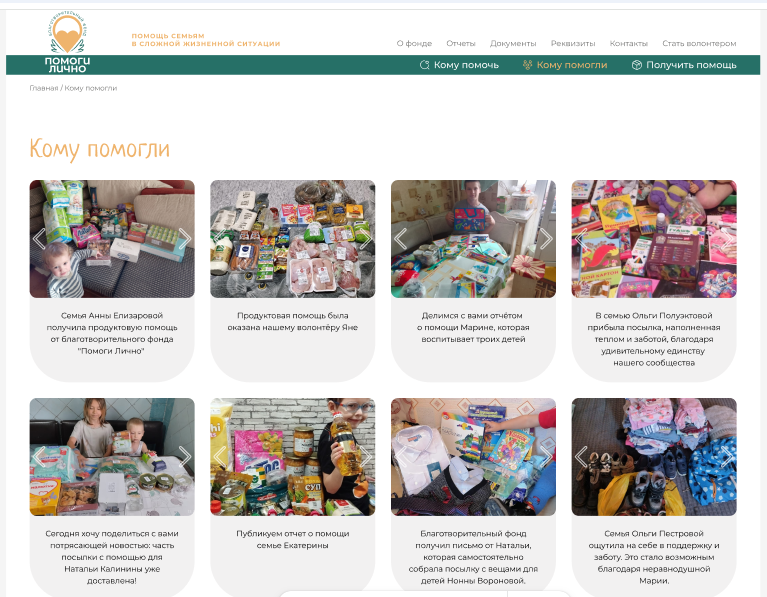
HelpedList — массив с объектами раздела и параметрами пагинации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| HelpedItem | id | photos\_in\_report | ph\_id |  |
| image\_url | phrep\_path |  |
| caption | photo\_reports | phrep\_description |  |
| published\_at | phrep\_created\_at |  |
| pagination | current\_page | - | - | = значению параметра “page” в запросе |
| per\_page | - | = значению параметра “per\_page” в запросе |
| total | - | Общее кол-во записей в таблице video\_reports |
| last\_page | - | Номер по  Вычисляется как total/per\_page, округленное к большему |

При нажатии на пагинацию (например, переход на следующую страницу), клиент отправляет GET-запрос на тот же эндпоинт /blog/articles, но с обновленными параметрами page и per\_page. Сервер возвращает новый набор статей и обновленную информацию о пагинации.

Пример запроса GET /blog/articles?page=2&per\_page=20

**Frontend:**

Заголовок «Кому помогли» захардкодить на фронте.

**Авторизация (POST)**

/admin/auth — метод авторизации администратора.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **АтрибутJSON** | **Название**  **таблицыв БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| LikeResponse | username | admin\_users | login |  |
| password | password |  |

Backend:

Метод foundation\_reports использует для передачи данных администратора в теле запроса.

При успешном авторизации, выдается токен доступа.

В случае ошибок авторизации, выводится ошибка на экран.

**Страница семей для создания, редактирования, просмотра, удаления семей, всех их атрибутов, фото, документов, новостей**

**Загрузка для просмотра (GET)**

**GET** /admin/families – метод просмотра списка семей для администратора

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **Атрибут JSON** | **Название**  **таблицы в БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| families | f\_id | family | f\_id |  |
| f\_name | f\_name |  |
| fph\_path | family\_photos | fph\_path |  |
| f\_short\_desc | family | f\_short\_desc |  |
| f\_desc | f\_desc |  |
| f\_sum | f\_sum |  |
| f\_current\_sum | f\_current\_sum |  |
| f\_created\_at | f\_created\_at |  |
|  | can\_edit | - | - | Признак редактируемости карточки, только для админа |
| pagination | current\_page | - | - | = значению параметра “page” в запросе |
| per\_page | - | - | = значению параметра “per\_page” в запросе |
| total | - | - | Общее кол-во записей в таблице family |
| last\_page | - | - | Номер по  Вычисляется как total/per\_page, округленное к большему |

1. Атрибуты объекта pagination необходимо возвращать для постраничной выдачи списка семей на главной странице.
2. В запросе к БД необходимо:

* сделать сортировку по дате создания от новому к старому (f\_created\_at);

если в запросе переданы параметры page и per\_page:

* ограничить выдачу количеством записей, равным значению параметра per\_page в запросе;
* пропускать количество записей, равное (page — 1)\*per\_page

page: от 1 до 999

per\_page: от 1 до 10

**Создание (POST) карточки семьи**

POST /admin/families – метод создания карточки семьи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Объект JSON** | **Атрибут JSON** | **Название**  **таблицы в БД** | **Атрибут в**  **таблице БД** | **Комментарий** |
| families | f\_id | family | f\_id |  |
| f\_name | f\_name |  |
| fph\_path | family\_photos | fph\_path |  |
| f\_short\_desc | family | f\_short\_desc |  |
| f\_desc | f\_desc |  |
| f\_sum | f\_sum |  |
| f\_current\_sum | f\_current\_sum |  |
| f\_created\_at | f\_created\_at |  |

**Backend:**

При успешной отправке запроса, происходит запись данных семьи INSERT INTO в таблицы family, family\_photos, family соответствующих параметров запроса f\_id, f\_name, fph\_path, f\_short\_desc, f\_desc, f\_sum, f\_current\_sum, f\_created\_at.