Описание WEB-API

Ваша задача - создать облако хранения данных с возможностью разграничения прав доступа к файлам.

ОСНОВНОЙ ФУНКЦИОНАЛ

Для неавторизованного пользователя:

- авторизация
- регистрация

Для авторизованного пользователя:

- возможность сброса авторизации
- работа с файлами
    - загрузка
    - редактирование
    - удаление
- разграничение прав доступа к файлам

Функционал авторизованного пользователя не должен быть доступен гостю.

ОБЩЕЕ

Вам необходимо реализовать REST API заданной структуры.

В примерах будет использоваться переменная {{host}} которая обозначает адрес http://web-app.api-web-tech.local/, где xxxxxx - логин участника.

Идентификацию пользователя организуйте удобным для вас способом.

При попытке доступа гостя к защищенным авторизацией функциям системы во всех запросах необходимо возвращать ответ следующего вида:

Status: 403
Content-Type: application/json
Body:

```
{
  "message": "Login failed"
}
```

При попытке доступа авторизованным пользователем к функциям недоступным для него во всех запросах необходимо возвращать ответ следующего вида:

Status: 403
Content-Type: application/json
Body:

```
{
  "message": "Forbidden for you"
}
```

При попытке получить не существующий ресурс необходимо возвращать ответ следующего вида:

Status: 404
Content-Type: application/json
Body:

```
{
  "message": "Not found"
}
```

В случае ошибок связанных с валидацией данных во всех запросах необходимо возвращать следующее тело ответа:

Status: 422
Content-Type: application/json
Body:

```
{
  "success": false,
  "message": {
        <key>: [<error message>]
    }
}
```

В свойстве message.<key> необходимо перечислить те свойства, которые не прошли валидацию, а в их значениях указать массив с ошибками валидации.

Например если отправить пустой запрос на сервер, где проверяется следующая валидация:
phone – обязательно поле
password – обязательное поле
то тело ответа будет следующим:

```
{
  "success": false,
  "message": {
        "phone": [ "field phone can not be blank"],
        "password": ["field password can not be blank"]
    }
}
```

В поле message могут быть любые сообщения об ошибках (если не указана конкретная ошибка), но они должны описывать возникшую проблему.

Файлы должны загружаться на сервер в папку uploads (находящуюся в корне веб-

сервера) и иметь уникальные сгенерированные имена.

Вы должны завести тестовый аккаунт с email test@gmail.com и паролем test.

ОПИСАНИЕ ФУНКЦИОНАЛА ГОСТЯ

Аутентификация:
При успешной аутентификации возвращается сгенерированный токен пользователя.

<table>
<tr><th>Request</th><th>Response</th></tr>
<tr><td rowspan="2">URL: {{host}}/authorization
Method: POST

Headers
- Content-Type: application/json

Body:
{
"email": "admin@admin.ru",
"password": "Qa1"
}</td><td>Успех
Status: 200
Content-Type: application/json
Body:
{
"success": true,
"message": "Success",
"token": <сгенерированный token>
}</td></tr>
<tr><td>Неправильные логин или пароль
Status: 401
Content-Type: application/json
Body:
{
"success": false,
"message": "Login failed"
}</td></tr>
</table>

Регистрация:
При успешной регистрации возвращается сгенерированный токен добавленного пользователя. Функция должна принимать следующие параметры:

- email - email пользователя
  обязательный, валидный e-mail
  адрес, уникальный
- password - пароль пользователя
  обязательный, состоит минимум из 3 символов, из которых как минимум одна строчная, одна прописная и одна цифра
- first_name - имя,
  обязательное
- last_name - фамилия
  обязательное

Request	Response

URL: {{host}}/registration
Method: POST
 	
Headers
- Content-Type: application/json
 	
Body:
{
"email": "admin@admin.ru",
"password": "Qa1",
"first_name": "name",
"last_name": "last_name"
}
	Успех
	Status: 200
	Content-Type: application/json
	Body:
	{
	"success": true,
	"message": "Success",
	"token": <сгенерированный token>
	}
	
	Ошибки валидации
	Формат ответа из общих требований

ОПИСАНИЕ ФУНКЦИОНАЛА АВТОРИЗИРОВАННОГО ПОЛЬЗОВАТЕЛЯ

Сброс авторизации:
Запрос предназначен для очистки значение токена пользователя.

Request	Response

URL: {{host}}/logout
Method: GET
	Успех
	Status: 200
	Content-Type: application/json
	Body:
	{
	"success": true,
	"message": "Logout"
	}
	
	Попытка обратиться не авторизованным пользователем
	Формат ответа из общих требований

Загрузка файлов:

Функция должна принимать следующие параметры:

- files - файлы, разрешены следующие типы файлов: doc, pdf, docx, zip, jpeg, jpg, png;

Если какие-либо из файлов по какой-либо причине не загружены, остальные должны

загрузиться и занестись в БД.
Если имя загружаемого файла совпадает с именем файла, который уже загружен у данного пользователя, то загружаемый файл переименовывается в “({I})_{NAME}.{EXT}”. Например: загружаем файл “Juicy chicken.doc”, и если такой файл уже есть, то загружаемый файл переименовывается в “(1)_Juicy chicken.doc”, или “(2)_Juicy chicken.doc”, если такой файл уже там есть и т.д.
Также с каждым файлом ассоциирован его уникальный идентификатор file_id – поле БД с параметром автоинкримент.

<table>
<tr><th>Request</th><th>Response</th></tr>
<tr><td rowspan="3">URL: {{host}}/upload
Method: POST

Headers
- Content-Type: multipart/form-data

FormData:
“files”:<массив с файлами>,
“token”: <токен, полученный в ходе авторизации></td><td><u>Успех</u>
Status: 200
Content-Type: application/json
Body:
[
{
"success": true,
"message": "Success",
"name": "Имя загруженного файла",
"file_id": "qweasd1234"
},
{
"success": false,
"message": "File not loaded",
"name": "Имя НЕ загруженного файла"
}
]</td></tr>
<tr><td><u>Ошибки валидации</u>
Вывод в общем списке загруженных файлов (см. пример выше)</td></tr>
<tr><td><u>Попытка обратиться не авторизованным пользователем</u>
Формат ответа из общих требований</td></tr>
</table>

<u>Редактирование файла</u>:
Функция должна принимать следующие параметры:

- name - имя файла, не пустое, уникальное для пользователя
- id_file – уникальный идентификатор файла в системе

Request	Response

URL: {{host}}/edit
Method: PATH
 	
Headers
- Content-Type: application/json
 	
Body:
{
"name": "new Name"
“id_file”: <id_file>,
“token”: <токен, полученный в ходе авторизации>
 	
}
	Успех
	Status: 200
	Content-Type: application/json
	Body:
	{
	"success": true,
	"message": "Renamed"
	}
	
	Ошибки валидации
	Формат ответа из общих требований
	
	Попытка обратиться не авторизованным пользователем
	Формат ответа из общих требований
	
	Попытка изменить файл не владельцем
	Формат ответа о запрете доступа из общих требований
	
	Попытка доступа к несуществующему объекту
	Формат ответа из общих требований

Удаление файла:
Функция должна принимать следующие параметры:

- id_file – уникальный идентификатор файла в системе

Request	Response

URL: {{host}}/delete
Method: DELETE
 	
Headers
- Content-Type: application/json
Body:
{
“id_file”: <id_file>,
“token”: <токен, полученный в ходе авторизации>
}
	Успех
	Status: 200
	Content-Type: application/json
	Body:
	{
	"success": true,
	"message": "File already deleted"
	}
	
	Попытка обратиться не авторизованным пользователем
	Формат ответа из общих требований
	
	Попытка удалить файл не владельцем
	Формат ответа о запрете доступа из общих требований

	Попытка доступа к несуществующему объекту **Формат ответа из общих требований**

Скачивание файла:
Функция должна принимать следующие параметры:

- id_file – уникальный идентификатор файла в системе

Request	Response

URL: {{host}}/download
Method: GET
Body:
{
“id_file”: <id_file>,
“token”: <токен, полученный в ходе авторизации>
 	
}
	Успех
	Браузеру отдается файл для скачивания

	Попытка обратиться не авторизованным пользователем **Формат ответа из общих требований**
	Попытка скачать файл к которому нет доступа **Формат ответа о запрете доступа из общих требований**
	Попытка доступа к несуществующему объекту **Формат ответа из общих требований**

Добавление прав доступа:
Функционал добавление прав доступа к файлу должен быть доступен только владельцу файла. В ответ должны возвращаться все пользователи, имеющие доступ к объекту.
Функция должна принимать следующие параметры:

- id_file – уникальный идентификатор файла в системе
- email – email пользователя, которому будут даны права

Request	Response
URL: {{host}}/accesses **Method:** POST **Headers** - Content-Type: application/json	Успех **Status:** 200 **Content-Type:** application/json **Body:** [{ "fullname": "name last_name",

Body:
{
“id_file”: <id_file>,
"email": "user@user.ru",
“token”: <токен, полученный в ходе авторизации>
 	
}
	"email": "admin@admin.ru",
	"type": "author"
	},
	{
	"fullname": "user last_name",
	"email": "user@user.ru",
	"type": "co-author"
	}
	]

	Попытка обратиться не авторизованным пользователем **Формат ответа из общих требований**
	Попытка обратиться не владельцем **Формат ответа о запрете доступа из общих требований**
	Попытка доступа к несуществующему объекту **Формат ответа из общих требований**

Удаление прав доступа:
Функционал удаления прав доступа к файлу должен быть доступен только создателю. В ответ должны возвращаться все пользователи, имеющие доступ к объекту.
Функция должна принимать следующие параметры:

- id_file – уникальный идентификатор файла в системе
- email – email пользователя, у которого будут удалены права

Request	Response

URL: {{host}}/deleteaccesses
Method: DELETE
 	
Headers
- Content-Type: application/json
 	
Body:
{
“id_file”: <id_file>,
"email": "user@user.ru",
“token”: <токен, полученный в ходе авторизации>
	Успех
	Status: 200
	Content-Type: application/json
	Body:
	[
	{
	"fullname": "name last_name",
	"email": "admin@admin.ru",
	"type": "author"
	}
	]

	Попытка удаления пользователя которого нет в списке соавторов **Формат ответа о ненайденном ресурсе**

}	**из общих требований**
	Попытка удаления самого себя **Формат ответа о запрете доступа из общих требований**
	Попытка обратиться не авторизованным пользователем **Формат ответа из общих требований**
	Попытка обратиться не владельцем **Формат ответа о запрете доступа из общих требований**
	Попытка доступа к несуществующему объекту **Формат ответа из общих требований**

Просмотр файлов пользователя:

Request	Response

URL: {{host}}/disk
Method: GET
 	
Headers
- Content-Type: application/json
 	
Body:
{
“token”: <токен, полученный в ходе авторизации>
}
	Успех
	Status: 200
	Content-Type: application/json
	Body:
	[
	{
	"file_id": "qweasd1234",
	"name": "Имя файла",
	"accesses":
	[
	{
	"fullname": "name last_name",
	"email": "admin@admin.ru",
	"type": "author"
	},
	{
	"fullname": "user last_name",
	"email": "user@user.ru",
	"type": "co-author"
	}
	]
	
	},
	{

	"file_id": "aaaaaaaaaa",
	"name": "Имя файла 1",
	"accesses":
	[
	{
	"fullname": "name last_name",
	"email": "admin@admin.ru",
	"type": "author"
	}
	]
	
	}
	]

	Попытка обратиться не авторизованным пользователем **Формат ответа из общих требований**

Просмотр файлов, к которым имеет доступ пользователь:
В списке не должны присутствовать файлы самого пользователя

Request	Response

URL: {{host}}/shared
Method: GET
 	
Headers
- Content-Type: application/json
 	
Body:
{
“token”: <токен, полученный в ходе авторизации>
}
	Успех
	Status: 200
	Content-Type: application/json
	Body:
	[
	{
	"file_id": "qweasd1234",
	"name": "Имя файла",
	},
	{
	"file_id": "aaaaaaaaaa",
	"name": "Имя файла 2",
	}
	]

	Попытка обратиться не авторизованным пользователем **Формат ответа из общих требований**