Задание. Разработка на стороне сервера

**Время выполнения:** 2 часа

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.

# Заказчик не предоставляет готовую базу данных и вам необходимо разработать ее самостоятельно.

# Общее

Вам необходимо реализовать REST API заданной структуры.

FD

В примерах будет использоваться переменная {{host}} которая обозначает адрес http://xxxxxx-m3.wsr.ru/api-file, где xxxxxx - логин участника.

Идентификацию пользователя организуйте посредством Bearer Token.

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",

"code": 404

}

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

**Status:** 422

**Content-Type:** application/json

**Body:**

{

"success": false,

"code": <code>,

"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,

"code": 422,

"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.

# Описание функционала гостя

Аутентификация:

При успешной аутентификации возвращается сгенерированный токен пользователя.

* email — обязательное
* password - обязательное

| Request | Response |
| --- | --- |
| **URL:** {{host}}/authorization  **Method:** POST  **Headers**  - Content-Type: application/json  **Body:**  {  "email": "admin@admin.ru",  "password": "Qa1"  } | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "success": true,  "code": 200,  "message": "Success",  "token": <сгенерированный token>  }  Ошибки валидации  **Формат ответа из общих требований**    Неправильные логин или пароль  **Status:** 401  **Content-Type:** application/json  **Body:**  {  "success": false,  "code": 401,  "message": "Authorization failed"  } |
|  |  |

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, состоит минимум из 2 символов 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:** 201  **Content-Type:** application/json  **Body:**  {  "success": true,  "code": 201,  "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:** 204  **Content-Type:** application/json  **Body (не отдается)**    Попытка обратиться не авторизованным пользователем  **Формат ответа из общих требований** |

Загрузка файлов:

Функция должна принимать следующие параметры:

* files - файлы, разрешены файлы размером не более 2 мб,
* files - 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 “{NAME} ({I}).{EXT}”. Например: загружаем файл “Juicy chicken.doc”, и если такой файл уже есть, то загружаемый файл переименовывается в “Juicy chicken (1).doc”, или “Juicy chicken (2).doc”, если такой файл уже там есть и т.д. Также с каждым файлом ассоциирован его уникальный идентификатор file\_id - случайная строка из 10 символов.

| Request | Response |
| --- | --- |
| **URL:** {{host}}/files  **Method:** POST  **Headers**  - Content-Type: multipart/form-data  **FormData:**  “files”:<массив с файлами> | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  [  {  "success": true,  "code": 200,  "message": "Success",  "name": "Имя загруженного файла",  "url": "{{host}}/files/qweasd1234",  "file\_id": "qweasd1234"  },  {  "success": false,  "message": [<error message>],  "name": "Имя НЕ загруженного файла"  }  ]    Ошибки валидации  **Вывод в общем списке загруженных файлов (см. пример выше)**  Попытка обратиться не авторизованным пользователем  **Формат ответа из общих требований** |

Редактирование файла:

Функция должна принимать следующие параметры:

* name - имя файла, не пустое, уникальное для пользователя

| Request | Response |
| --- | --- |
| **URL:** {{host}}/files/<file\_id>  **Method:** PATH  **Headers**  - Content-Type: application/json  **Body:**  {  "name": "new Name"  } | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "success": true,  "code": 200,  "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:

| Request | Response |
| --- | --- |
| **URL:** {{host}}/files/<file\_id>  **Method:** DELETE  **Headers**  - Content-Type: application/json | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "success": true,  "code": 200,  "message": "File 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:

| Request | Response |
| --- | --- |
| **URL:** {{host}}/files/<file\_id>  **Method:** GET | 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.

| Request | Response |
| --- | --- |
| **URL:** {{host}}/files/<file\_id>/accesses  **Method:** POST  **Headers**  - Content-Type: application/json  **Body:**  {  "email": "user@user.ru"  } | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  [  {  "fullname": "name last\_name",  "email": "admin@admin.ru",  "type": "author",  "code": 200,  },  {  "fullname": "user last\_name",  "email": "user@user.ru",  "type": "co-author",  "code": 200,  }  ]    Попытка обратиться не авторизованным пользователем  **Формат ответа из общих требований**    Попытка обратиться не владельцем  **Формат ответа о запрете доступа из общих требований**  Попытка дать доступ к несуществующему пользователю  **Формат ответа 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.

| Request | Response |
| --- | --- |
| **URL:** {{host}}/files/<file\_id>/accesses  **Method:** DELETE  **Headers**  - Content-Type: application/json  **Body:**  {  "email": "user@user.ru"  } | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  [  {  "fullname": "name last\_name",  "email": "[admin@admin.ru](mailto:admin@admin.ru)",  "code": 200,  "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 удаления прав у несуществующего пользователя  **Формат ответа из общих требований**    Попытка доступа к несуществующему объекту  **Формат ответа из общих требований** |

Просмотр файлов пользователя:

| Request | Response |
| --- | --- |
| **URL:** {{host}}/files/disk  **Method:** GET  **Headers**  - Content-Type: application/json | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  [  {  "file\_id": "qweasd1234",  "name": "Имя файла",  "code": 200,  "url": "{{host}}/files/qweasd1234",  "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",  "code": 200,  "url": "{{host}}/files/aaaaaaaaaa",  "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}}/files/shared  **Method:** GET  **Headers**  - Content-Type: application/json | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  [  {  "file\_id": "qweasd1234",  "code": 200,  "name": "Имя файла",  "url": "{{host}}/files/qweasd1234",  },  {  "file\_id": "aaaaaaaaaa",  "code": 200,  "name": "Имя файла 2",  "url": "{{host}}/files/aaaaaaaaaa",  }  ]    Попытка обратиться не авторизованным пользователем  **Формат ответа из общих требований** |

## Инструкция для конкурсанта

Вам необходимо две тестовые учетные записи со следующими данными:

* email: user1@test.ru, password: Qa1
* email: user2@test.ru, password: As2

Вам предоставляются следующие конфигурации фреймворков и библиотек:

* Laravel 10.x
* Yii Basic 2.0.x
* Django 4.0.x (включая пакеты: djangorestframework, django-cors-headers, pillow, django-filter, mysqlclient)

Вы можете использовать любой из представленных фреймворков.