Задание для демонстрационного экзамена по комплекту оценочной документации

|  |  |
| --- | --- |
| **Номер компетенции** | 17 |
| **Название компетенции** | Веб-дизайн и разработка |
| **Номер КОД** | 1.3 |

# Описание задания.

# Описание модуля «Разработка API»

# Введение

**Технологии этого модуля:** REST API

**Время на выполнение:** 1,5 часа

К Вам обратилась компания «Матрешка» для разработки API сервиса заказа сувениров с их склада. Необходимо реализовать систему регистрации и авторизации клиентов и администратора, а также различный функционал, связанный с этими ролями. Вся создаваемая информация, том числе данные аккаунтов, должна храниться в базе данных.

# Описание проекта и задач

## Общее описание проекта

В информационной системе присутствуют три вида пользователей:

1. Гость;
2. Клиент;
3. Администратор.

**Гость** должен иметь возможность выполнять следующие функции:

1. Аутентификация;
2. Регистрация;
3. Просмотр списка товаров.

При **регистрации** должны обрабатываться следующие данные:

1. ФИО – обязательное поле, строка;
2. E-mail – обязательное поле, валидируется на соответствие шаблону e-mail адресов, должно быть уникальным, выполняет функцию логина;
3. Пароль - обязательное поле, должно содержать не менее 8 символов.

Зарегистрированный пользователь после аутентификации получает роль **Клиент**.

**Клиент** может выполнять следующие функции:

1. Просмотр списка товаров;
2. Добавление товара в корзину;
3. Просмотр своей корзины;
4. Удаление товара из корзины;
5. Оформления заказа;
6. Просмотр своих оформленных заказов;
7. Выход.

При оформлении заказа корзина пользователя очищается и заказ добавляется в список оформленных заказов.

После аутентификации с учетными данными администратора гость получает роль **Администратор**.

**Администратор** может выполнять следующие функции:

1. Просмотр списка товаров;
2. Возможность добавлять, удалять и редактировать товары;
3. Выход.

Создайте **Администратора** со следующими учетными данными:

1. E-mail – admin@souvenir.ru;
2. Пароль – QWEasd123.

Создайте **Клиента** со следующими учетными данными:

1. E-mail – user@souvenir.ru;
2. Пароль – password.

Каждый товар содержит следующие не пустые поля: название, описание, цена.

Ваша задача – реализовать REST API заданной структуры.

В примерах будет использоваться переменная {{host}} которая обозначает адрес http://xxxxxx-m1.wsr.ru/api-souvenir, где xxxxxx - логин участника.

## Общие требования к API

Идентификацию пользователя организуйте посредством Bearer Token.

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

**Status:** 403

**Content-Type:** application/json

**Body:**

{

“error”: {

“code”: 403,

“message”: “Ошибка авторизации”

}

}

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

**Status:** 403

**Content-Type:** application/json

**Body:**

{

“error”: {

“code”: 403,

“message”: “Запрещено для вас”

}

}

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

**Status:** 404

**Content-Type:** application/json

**Body:**

{

"error": {

"code": 404,

"message": "Ресурс не найден"

}

}

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

{

“error”: {

“code”: 422,

“message”: “Ошибка валидации”,

“errors”: {

<key>: [ <error message>]

}

}

}

В свойстве error.errors 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:

{

“error”: {

“code”: 422,

“message”: “Ошибка валидации”,

“errors”: {

“phone”: [ “field phone can not be blank” ],

“password”: [ “field password can not be blank” ]

}

}

}

В значениях свойств errors вы можете использовать любые сообщения об ошибках (если не указана конкретная ошибка), но они должны описывать возникшую проблему.

## Специфические требования к API

### Функционал гостя

#### Аутентификация

При успешной аутентификации возвращается сгенерированный токен пользователя.

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/login  **Method:** POST  **Headers**  - Content-Type: application/json  **Body:**  {  “email”: “admin@admin.ru“,  “password“: “admin“  } | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  {  “data”: {  “user\_token”: <сгенерированный token>  }  }  Неправильные логин или пароль  **Status:** 401  **Content-Type:** application/json  **Body:**  {  “error”: {  “code”: 401,  “message”: “Неправильные логин или пароль”  }  } |

#### Регистрация

При успешной регистрации возвращается сгенерированный токен добавленного пользователя

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/signup  **Method:** POST  **Headers**  - Content-Type: application/json  **Body:**  {  “fio”: “Иванов Иван Иванович“,  “email”: “admin@admin.ru“,  “password“: “admin“  } | Успех  **Status:** 201  **Content-Type:** application/json  **Body:**  {  "data": {  “user\_token”: <сгенерированный token>  }  }  Ошибки валидации полей  **Формат ответа из общих требований** |

#### Просмотр списка товаров

Возвращается массив data, содержащий список объектов товаров.

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/products  **Method:** GET | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  [  {  "id": 1,  "name": "Product name 1",  "description": "Product description 1",  "price": 100  },  {  "id": 2,  "name": "Product name 2",  "description": "Product description 2",  "price": 200  }  ] |

### Функционал клиента

#### Добавление товара в корзину

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/cart/{product\_id}  **Method:** POST  **Примечание:**  {product\_id} - идентификатор товара | Успех  **Status:** 201  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Сувенир добавлен в корзину"  }  } |

#### Просмотр своей корзины

Возвращается массив data, содержащий список товаров в корзине.

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/cart  **Method:** GET | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  [  {  "id": 1,  "product\_id": 1,  "name": "Product name 1",  "description": "Product description 1",  "price": 100  },  {  "id": 2,  "product\_id": 1,  "name": "Product name 1",  "description": "Product description 1",  "price": 100  },  {  "id": 3,  "product\_id": 2,  "name": "Product name 2",  "description": "Product description 2",  "price": 200  }  ]  **Примечание:**  “id” - идентификатор товара в корзине  “product\_id” - идентификатор товара |

#### Удаление товара из корзины

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/cart/{id}  **Method:** DELETE  **Примечание:**  {id} - идентификатор товара в корзине | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Сувенир удален из корзины"  }  }  Попытка удалить товар не из своей корзины  **Status:** 403  **Content-Type:** application/json  **Body:**  {  "error": {  "code": 403,  "message": "Запрещено для вас"  }  } |

#### Оформления заказа

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/order  **Method:** POST | Успех  **Status:** 201  **Content-Type:** application/json  **Body:**  {  "data": {  "order\_id": 1  "message": "Заказ оформлен"  }  }  Попытка оформить заказ с пустой корзиной  **Status:** 422  **Content-Type:** application/json  **Body:**  {  "error": {  "code": 422,  "message": "Корзина пуста"  }  } |

#### Просмотр своих оформленных заказов

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/order  **Method:** GET | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  [  {  "id": 1,  "products": [1, 1, 2],  "order\_price": 400  },  {  "id": 5,  "products": [1, 2],  "order\_price": 300  }  ]  **Примечание:**  Массив products содержит идентификаторы товаров в заказе |

#### Выход

Запрос предназначен для очистки значение токена пользователя.

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/logout  **Method:** GET | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Вы вышли из системы"  }  } |

### Функционал администратора

#### Добавление нового товара

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/product  **Method:** POST  **Headers**  - Content-Type: application/json  **Body:**  {  "name": "Product name 3",  "description": "Product description 3",  "price": 300  } | Успех  **Status:** 201  **Content-Type:** application/json  **Body:**  {  "data": {  "id": 5,  "message": "Сувенир добавлен в каталог"  }  }  Ошибки валидации полей  **Формат ответа из общих требований** |

#### Удаление товара

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/product/{id}  **Method:** DELETE | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Сувенир удален из каталога"  }  } |

#### Редактирование товара

Возможно частичное редактирование данных товара. При успешном редактировании возвращается объект data с измененными данными.

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/product/{id}  **Method:** PATCH  **Headers**  - Content-Type: application/json  **Body:**  {  "price": 500  } | Успех  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "id": 1,  "name": "Product name 1",  "description": "Product description 1",  "price": 500  } |

## Инструкция для конкурсанта

Разработанное API должно быть доступно по адресам, указанным в требованиях к API.

Вам предоставляются следующие конфигурации фреймворков:

* Laravel 8.5.x
* Yii 2.0.x
* Django 3 (djangorestframework, django-cors-headers, pillow, markdown, django-filter, mysqlclient)

Вы можете использовать любой из представленных фреймворков

## Общее описание проекта

Ваша задача – реализовать SPA приложение, которое будет взаимодействовать с уже разработанным API.