**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

**ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЙ КЛИЕНТКОЙ И АДМИНИСТРАТИВНОЙ ЧАСТИ ДЛЯ СЛУЖБЫ ДОСТАВКИ СУШ**

***КУРСОВОЙ ПРОЕКТ***

*СПЕЦИАЛЬНОСТЬ 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ*

Выполнил:

Студент группы 1912с

\_\_\_\_\_\_\_\_ / Бондаренко К.В.

\_\_\_\_\_\_\_\_ / Рюгин А.И.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

Проверил:

\_\_\_\_\_\_\_\_ / Уляхин В.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Томск 2025

Оглавление

[ВВЕДЕНИЕ 3](#_Toc180599301)

[ТЕХНОЛОГИИ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 5](#_Toc180599302)

[Используемые языка программирования и фреймворки 5](#_Toc180599303)

[Используемое программное обеспечение 5](#_Toc180599304)

[ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc180599305)

[ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ИНФОРМАЦИОННОГО ПРОДУКТА. 8](#_Toc180599306)

[Специфические требования к API 10](#_Toc180599307)

[Функционал авторизированного пользователя 14](#_Toc180599308)

[Функционал администратора 18](#_Toc180599309)

# ВВЕДЕНИЕ

API (Application Programming Interface) представляет собой набор механизмов, которые позволяют различным программным компонентам взаимодействовать друг с другом с использованием определённых протоколов и правил. Основная задача API — обеспечить возможность коммуникации между приложениями, абстрагируя их взаимодействие от деталей внутренней реализации. В данном контексте, термин "приложение" относится к любому программному обеспечению, выполняющему конкретные задачи или функции.

API действует как сервисный контракт между двумя приложениями, определяя, как они должны обмениваться запросами и ответами. Этот контракт включает структуру запросов, формат ответов и типы используемых данных. Для разработчиков важной частью API является его документация, в которой описаны правила использования и взаимодействия с API.

**Архитектура API: Клиент и Сервер**

Архитектура API обычно строится по модели клиент-сервер. В такой архитектуре клиент — это приложение, которое инициирует запросы, а сервер — это приложение, которое отвечает на них. Например, если речь идёт о погодных данных, клиентом будет мобильное приложение, запрашивающее данные, а сервером — база данных службы погоды, которая возвращает информацию в ответ на запрос.

**REST API**

Одним из широко используемых типов API является REST API (Representational State Transfer). Одна из ключевых особенностей REST заключается в его архитектуре без сохранения состояния (stateless), что означает, что сервер не хранит информацию о клиенте между запросами. Каждый запрос к серверу обрабатывается независимо, как если бы это был отдельный запрос на веб-страницу. Ответы от сервера содержат данные в формате, пригодном для обработки, например, в JSON, но не включают элементы графического интерфейса, как это бывает при загрузке веб-страниц.

**MAUI**

MAUI (Multi-platform App UI) — это кросс-платформенная фреймворк от Microsoft, позволяющий разработчикам создавать приложения, которые могут работать на различных устройствах и операционных системах, таких как Android, iOS, macOS и Windows, используя общий код. MAUI является частью экосистемы .NET и предоставляет единую архитектуру для построения интерфейсов и управления логикой приложений, что существенно упрощает разработку мобильных приложений и повышает их производительность.

С помощью MAUI разработчики могут создавать приложения, использующие один кодовый базис для работы на разных платформах, что сокращает время разработки и тестирования, а также обеспечивает согласованный пользовательский опыт на всех устройствах. В данном проекте MAUI будет использоваться для разработки мобильных приложений, которые будут взаимодействовать с REST API для службы доставки суши.

**Цель курсового проекта**

Цель данного курсового проекта заключается в проектировании и разработке REST API для службы доставки суши с использованием фреймворка Laravel, а также в разработке кросс-платформенных мобильных приложений на MAUI. REST API будет обеспечивать взаимодействие между мобильным приложением и сервером доставки суши, предоставляя функциональность для размещения и отслеживания заказов, управления меню и взаимодействия с клиентами. MAUI позволит разработать удобное мобильное приложение, доступное на различных платформах.

# ТЕХНОЛОГИИ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

## Используемые языка программирования и фреймворки

**PHP** — C-подобный скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

**SQL** — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

**Laravel** — бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (англ. Model View Controller — модель-представление-контроллер). Laravel выпущен под лицензией MIT.

**XAML** (eXtensible Application Markup Language) — это язык разметки, разработанный компанией Microsoft, который используется для определения пользовательских интерфейсов в приложениях, особенно в таких технологиях, как Windows Presentation Foundation (WPF), Universal Windows Platform (UWP) и Xamarin.Forms.

## Используемое программное обеспечение

**Draw.io** — инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, отношений сущностей, программных блоков и другого. Сервис распространяется на бесплатной основе с открытым исходным кодом. Draw.io обладает богатым набором функций для визуализации большинства задач пользователя.

При входе на сервис пользователь сразу попадает в рабочий интерфейс. У пользователя нет возможности для авторизации или регистрации, есть только опция выбора места для экспорта проекта. Процесс создания проекта выглядит следующим образом: пользователь перетаскивает из левой панели фигуры или элементы на рабочую поверхность, затем изменяет их — изменяет цвет, размер, шрифт текста, свойства фигуры (прозрачность, форма и т. д.). Draw.io позволяет отслеживать и восстанавливать изменения готовых проектов, импортировать и экспортировать в PDF, PNG, XML, VSDX, HTML, а также автоматически публиковать и делиться работами.

Инструмент работает с Google Диск, Google Workspace и Dropbox, глубоко интегрирован и удобен для работы с продуктами Confluence и Jira от Atlassian. Пользователи также могут работать с диаграммами в автономном режиме и сохранять их локально, используя настольное приложение для персональных компьютеров.

Инструмент позволяет создавать: графики, диаграммы, таблицы, презентации, блок-схемы, планы помещений, воронки продаж, ментальный карты, карты сайтов.

**Open Server Panel** — это портативная программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий. Данный программный комплекс включает в себя тщательно подобранный набор серверного программного обеспечения, а также невероятно удобную и продуманную управляющую утилиту, которая обладает мощными возможностями по администрированию и настройке всех доступных компонентов. OSPanel широко используется с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях.

**PhpStorm** — коммерческая кроссплатформенная интегрированная среда разработки для PHP, обладающая широкими возможностями анализа кода и поддержкой основных фреймворков.

**Postman** — это платформа API, позволяющая разработчикам проектировать, создавать, тестировать и повторять свои API. По состоянию на февраль 2023 года Postman сообщает о более чем 25 миллионах зарегистрированных пользователей и 75 000 открытых API, которые, по его словам, представляют собой крупнейший в мире публичный API-хаб. Штаб-квартира компании находится в Сан-Франциско и имеет офис в Бангалоре, где она была основана.

**Visual Studio** — это интегрированная среда разработки (IDE), разработанная компанией Microsoft. Она предназначена для создания, тестирования и отладки различных типов программного обеспечения, включая настольные, мобильные и веб-приложения, а также игры.

# ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В данном курсовом проекте предметной областью является служба доставки суши. Это специализированная информационная система, которая позволяет пользователям заказывать разнообразные виды суши через мобильное приложение, а администраторам управлять товарами и категориями через административную панель. Система предоставляет возможность пользователям взаимодействовать с различными элементами каталога, оформлять заказы, а администраторам — эффективно управлять ассортиментом и контролировать структуру категорий.

**Технические особенности**

Основной акцент проекта направлен на разработку REST API для мобильного приложения и административной панели с использованием фреймворка Laravel для серверной части и MAUI для мобильной платформы. REST API обеспечивает взаимодействие клиентской и серверной частей, передавая данные в формате JSON, а для авторизации используется токен (Bearer Token), который обеспечивает безопасность операций и разграничение прав доступа между пользователями и администраторами.

**Роль системы**

Система доставки суши является важным элементом бизнес-экосистемы, обеспечивая удобство пользователей в оформлении заказов, предлагая широкий ассортимент товаров с возможностью быстрого выбора и гибкого управления заказами через мобильные устройства. Административная панель позволяет эффективно управлять ассортиментом и поддерживать актуальность каталога.

# ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ИНФОРМАЦИОННОГО ПРОДУКТА.

**ОБЩЕЕ ОПИСАНИЕ ПРОЕКТА**

Все пользователи системы подразделяются на две группы:

1. Администраторы
2. Пользователи

Категории товаров подразделяются на несколько групп

1. Суши с рыбой и морепродуктами
2. Вегетарианские суши
3. Суши с термически обработанной рыбой
4. Сладкие суши
5. Фьюжн суши
6. Суши с экзотическими ингредиентами

Функционал пользователя:

* Регистрация
* Авторизация
* Просмотр категорий товаров
* Просмотр товаров определенной категории

Функционал авторизированного пользователя

* Просмотр своего профиля
* Выход
* Добавление товара в корзину
* Просмотр своей корзины
* Оформление заказа
* Редактирование корзины
* Редактирование своего профиля

Функционал администратора

* Добавление категории
* Добавление товара
* Редактирование категории
* Редактирование товара
* Удаление категории
* Удаление товара

Задача курсового проекта – реализовать REST API клиентской и административной части

В примерах будет использоваться переменная {{host}} которая обозначает адрес информационного продукта.

API должно быть доступно с других доменов.

Идентификация пользователя организуется посредством Bearer Token.

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": "Forbidden for you"

}

}

При попытке доступа неавторизированным пользователем к функциям требующей авторизации во всех запросах необходимо возвращать ответ следующего вида:

Status: 403

Content-Type: application/json

Body:

{

"error": {

"code": 401,

"message": "Unauthorized"

}

}

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

{

"error": {

"code": <code>,

"message": <message>,

"errors": {

<key>: [<error message>]

}

}

}

Вместо <code> и <message> необходимо указывать соответствующее значение, определенное в описании ответа на соответствующий запрос. 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:

1. phone – обязательно поле
2. password – обязательное поле

то тело ответа будет следующим:

{

"error": {

"code": 422,

"message": "Validation error",

"errors": {

phone: ["field phone can not be blank"],

password: ["field password can not be blank"]

}

}

}

## Специфические требования к API

**Регистрация**

Запрос для регистрации нового пользователя в системе. При отправке запроса необходимо передать FormData со следующими полями:

* name – обязательное поле, строка;
* surname – обязательное поле, строка;
* patronymic – не обязательное поле, строка;
* login – обязательное и уникальное поле, строка;
* password – обязательное поле, строка;
* birth – обязательное поле, дата, формат Y-m-d;
* telephone – обязательное и уникальное поле, формат число;
* email – обязательное поле, строка;

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/register  **Method:** POST  **Headers:** Content-Type: application/json  **Body:**  {  "surname": "Usertest",  "name": " Usertest",  "patronymic": " Usertest",  "login": " Usertest",  "password": " Usertest1!",  "birth": "12.02.2003",  "telephone": "7930573565",  "email": "usertest@mail.ru"  } | Успешная регистрация  **Status:** 201  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Регистрация прошла успешно"  }  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Авторизация**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/login  **Method:** POST  **Headers:** Content-Type: application/json  **Body:**  {  "login": "log1",  "password": "pass1"  } | Успешная авторизация  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "api\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."  }  }  Ошибка аутентификации  **Status:** 401  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 401,      "message": "Ошибка аутентификации"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Просмотр категорий товаров**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/categories  **Method:** GET  **Headers:**  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": [  {  "id": 1,  "name": "Суши с рыбой и морепродуктами",  "created\_at": null,  "updated\_at": null,  },  {  "id": 2,  "name": "Вегетарианские суши"  "created\_at": null,  "updated\_at": null,  }  // Другие категории  ]  }  Ответ при отсутствии категорий  **Status:** 404  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 404,      "message": "Категории не найдены"  } |
|  |  |

**Просмотр товаров определенной категории**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/category/{id}  **Method:** GET  **Headers:**  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "data": [          {              "id": 1,              "name": "Морской бриз",              "description": "Нигири с тонким ломтиком свежего тунца, украшенный икрой летучей рыбы и каплей соевого соуса. Рис с легким вкусом васаби для пикантности.",              "price": "450.00",              "quantity": 10,              "photo": "uploads/1/1.webp",              "category\_id": 1,              "created\_at": null,              "updated\_at": "2024-03-08T00:00:00.000000Z"          },          {              "id": 2,              "name": "Рыбацкий улов",              "description": "Гункан суши с лососем, креветками и икрой лосося, украшенный ломтиками авокадо и свежим лимонным соком.",              "price": "600.00",              "quantity": 10,              "photo": "uploads/1/2.webp",              "category\_id": 1,              "created\_at": null,              "updated\_at": "2024-03-08T00:00:00.000000Z"          },  // Другие товары  ]  }  Ответ при отсутствии товаров  **Status:** 404  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 404,      "message": "Товары не найдены"  } |

**Просмотр конкретного товара**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/product/{id}  **Method:** GET  **Headers:**  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "data": [          {              "id": 1,              "name": "Морской бриз",              "description": "Нигири с тонким ломтиком свежего тунца, украшенный икрой летучей рыбы и каплей соевого соуса. Рис с легким вкусом васаби для пикантности.",              "price": "450.00",              "quantity": 10,              "photo": "uploads/1/1.webp",              "category\_id": 1,              "created\_at": null,              "updated\_at": "2024-03-08T00:00:00.000000Z"          }  ] }  Ответ при отсутствии товаров  **Status:** 404  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 404,      "message": "Товар не найден"  } |
|  |  |

## Функционал авторизированного пользователя

**Просмотр своего профиля**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/profile  **Method:** GET  **Headers:** Authorization: Bearer {{api\_token}}  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "data": {          "id": 1,          "name": "Иван",          "surname": "Иванов",          "patronymic": "Иванович",          "login": "ivan",          "password": "Ivan123@",          "birth": "1990-01-01",          "email": "ivan@example.com",          "telephone": 1234567890,      }  } |

**Выход**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/logout  **Method:** GET  **Headers:** Authorization: Bearer {{api\_token}}  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Вы успешно вышли из системы"  }  } |

**Просмотр своей корзины**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/cart  **Method:** GET  **Headers:** Authorization: Bearer {{api\_token}}  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "cart\_items": [],      "total": "0.00"  } |

**Добавление товара в корзину**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/product/{id}  **Method:** POST  **Headers:** Authorization: Bearer {{api\_token}},  Content-Type: application/json  **Body:**  {  "product\_id": 1,  "quantity": 2  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Товар успешно добавлен в корзину"  }  }  Ответ при ошибке  **Status:** 404  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 404,      "message": "Товар не найден"  }  Ответ при ошибке добавить количество больше, чем есть  **Status:** 400  **Content-Type:** application/json  **Body:**  {      "error": "Недостаточное количество товара в наличии"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Оформление заказа**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/checkout  **Method:** POST  **Headers:** Authorization: Bearer {{api\_token}},  Content-Type: application/json  **Body:**  {    "cart\_items": [      {        "product\_id": 1,        "quantity": 2      }    ],   "address": [      {        "city": "Томск",  "street": "Иркутский тракт",  "house": "175А",        "floor": 2,  "apartment": 41,  "entrance": 2,  "intercom": "41",  "comment": "Оставьте на вахте, please"      }    ],  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "message": "Заказ успешно оформлен"  }  Ответ при ошибке  **Status:** 404  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 404,      "message": "Не удалось оформить заказ, возможно, ваша корзина пуста"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Редактирование корзины**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/cart  **Method:** PATCH  **Headers:** Authorization: Bearer {{api\_token}},  Content-Type: application/json  **Body:**  {  "product\_id": 1,  "quantity": 5  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "message": "Количество товара в корзине успешно обновлено"  }  Ответ при попытке ввести отрицательное количество  **Status:** 400  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 400,      "message": "Количество товара должно быть положительным числом"  }  Ответ при попытке ввести количество больше имеющегося  **Status:** 400  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 400,      "message": "Количество товара должно быть положительным числом"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Редактирование своего профиля**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/profile  **Method:** PATCH  **Headers:** Authorization: Bearer {{api\_token}},  Content-Type: application/json  **Body:**  {  "name": "Новое Имя",  "surname": "Новая Фамилия",  "patronymic": "Новое Отчество"  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Профиль успешно обновлен"  }  }  Ошибки валидации полей  **Формат ответа из общих требований** |

## Функционал администратора

**Добавление категории**

Запрос для добавления новой категории в системе. При отправке запроса необходимо передать FormData со следующими полями:

* name – обязательное и уникальное поле, строка;

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/admin/category/create  **Method:** POST  **Headers:** Authorization: Bearer {{admin\_api\_token}},  Content-Type: application/json  **Body:**  {  "name": "Новая Категория"  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Категория успешно создана"  }  }  Ответ при попытке добавить категорию с уже существующем именем  **Status:** 422  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 422,      "message": "Категория с таким именем уже существует"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Добавление товара**

Запрос для регистрации нового пользователя в системе. При отправке запроса необходимо передать FormData со следующими полями:

* name – обязательное и уникальное поле, строка;
* description – необязательное поле, строка;
* price – обязательное поле, положительное вещественное число;
* quantity – обязательное поле, числовой;
* photo – необязательное поле, фото формата jpeg, png, jpg, webp;
* category\_id – обязательное поле, число, категория с данным идентификатором должна быть.

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/admin/product/create  **Method:** POST  **Headers:** Authorization: Bearer {{admin\_api\_token}},  Content-Type: application/json  **Body:**  {    "name": "Новый Товар",    "price": 1500,    "quantity": 10,    "description": "new product",    "category\_id": 1  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Товар успешно создан"  }  }  Ответ при попытке добавить продукт с уже существующем именем  **Status:** 422  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 422,      "message": "Продукт с таким именем уже существует"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Редактирование категории**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/admin/category/{id}/edit  **Method:** PATCH  **Headers:** Authorization: Bearer {{admin\_api\_token}}, Content-Type: application/json  **Body:**  {    "name": "Измененные суши"  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Категория успешно отредактирована"  }  }  Ответ при ошибке  **Status:** 422  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 422,      "message": "Категория с таким именем уже существует"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Редактирование товара**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/admin/product/{id}/edit  **Method:** POST  **Headers:** Authorization: Bearer {{admin\_api\_token}}, Content-Type: application/json  **Body:**  {  "name": "Новый Смартфон",  "price": 12000  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Товар успешно отредактирован"  }  }  Ответ при ошибке  **Status:** 400  **Content-Type:** application/json  **Body:**  {  "error": {  "message": "Не удалось отредактировать товар. Пожалуйста, проверьте данные запроса."  }  }  Ответ при попытке изменить продукт с уже существующем именем  **Status:** 422  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 422,      "message": "Продукт с таким именем уже существует"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

**Удаление категории**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/admin/category/{id}/delete  **Method:** DELETE  **Headers:** Authorization: Bearer {{admin\_api\_token}}  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Категория успешно удалена"  }  }  Ответ при ошибке  **Status:** 400  **Content-Type:** application/json  **Body:**  {  "error": {  "message": "Не удалось удалить категорию. Пожалуйста, проверьте данные запроса."  }  } |

**Удаление товара**

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/admin/product/{id}/delete  **Method:** DELETE  **Headers:** Authorization: Bearer {{admin\_api\_token}}  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Товар успешно удален"  }  }  Ответ при ошибке  **Status:** 400  **Content-Type:** application/json  **Body:**  {  "error": {  "message": "Не удалось удалить товар. Пожалуйста, проверьте данные запроса."  }  } |