**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

**САЙТ «Аниме интернет-магазин»**

***КУРСОВОЙ ПРОЕКТ***

*СПЕЦИАЛЬНОСТЬ 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ*

Выполнил:

Студент группы 1912с

\_\_\_\_\_\_\_\_\_\_\_\_ / Заикин В.А.

\_\_\_\_\_\_\_\_\_\_\_\_ / Рюгин А.И.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_/ Уляхин В.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Томск 2024

Оглавление

[ВВЕДЕНИЕ 3](#_Toc162523050)

[ТЕОРИЯ 5](#_Toc162523051)

[ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc162523052)

[Функционал пользователя 7](#_Toc162523053)

[Функционал авторизированного пользователя 13](#_Toc162523054)

[Функционал администратора 18](#_Toc162523055)

[ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСНОЙ ЧАСТИ 23](#_Toc162523056)

[ОПИСАНИЕ СУБД 24](#_Toc162523057)

[КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ ДАННЫХ 24](#_Toc162523058)

[НОРМАЛИЗАЦИЯ ДАННЫХ 25](#_Toc162523059)

[ЛОГИЧЕСКАЯ МОДЕЛЬ ДАННЫХ 26](#_Toc162523060)

[ФИЗИЧЕСКАЯ МОДЕЛЬ ДАННЫХ 27](#_Toc162523061)

[СЛОВАРЬ ДАННЫХ 29](#_Toc162523062)

[РАЗРАБОТКА ИНТЕРФЕЙСНОЙ ЧАСТИ 32](#_Toc162523063)

[РАЗРАБОТКА 33](#_Toc162523064)

# ВВЕДЕНИЕ

Описание предметной области

Программный продукт представляет собой веб-приложение, разработанное для управления товарами в интернет-магазине. В современном цифровом мире интернет-торговля становится все более популярной, и важно иметь эффективное средство для управления ассортиментом товаров и обеспечения удобства для пользователей. Приложение предназначено для упрощения процессов добавления, редактирования и удаления товаров, а также для предоставления возможности пользователям производить поиск по каталогу товаров.

Цель и задачи

Целью программного продукта является создание надежного и удобного инструмента для управления товарами в интернет-магазине с целью улучшения процессов торговли и повышения удовлетворенности пользователей. Основные задачи программы включают в себя:

1. Управление товарами: добавление, редактирование, удаление.
2. Поиск товаров по каталогу с возможностью фильтрации и сортировки результатов.
3. Предоставление пользователям доступа к информации о товарах, включая описание, характеристики, цены и наличие.
4. Реализация механизмов авторизации и аутентификации пользователей для обеспечения безопасности и конфиденциальности данных.
5. Разработка функционала корзины для добавления и управления товарами перед оформлением заказа.
6. Предоставление пользователю возможности оформления заказа с учетом выбранных товаров, адреса доставки и других необходимых данных.
7. Возможность оставления отзывов о товарах и просмотра отзывов других пользователей для повышения доверия к продукции и обмена опытом между пользователями.

Требования к программному продукту

Программный продукт должен соответствовать следующим основным требованиям и характеристикам:

* Интуитивно понятный и привлекательный интерфейс для облегчения взаимодействия пользователя с приложением.
* Высокая производительность и отзывчивость системы для обеспечения комфортного пользовательского опыта.
* Безопасность и конфиденциальность данных пользователей с использованием современных методов шифрования и защиты информации.
* Масштабируемость и гибкость системы для возможности дальнейшего расширения функционала и адаптации к изменяющимся потребностям бизнеса.

Программный продукт разрабатывается с учетом современных стандартов разработки программного обеспечения и с удовлетворением потребностей как администраторов интернет-магазина, так и его конечных пользователей.

# ТЕОРИЯ

Проект охватывает управление товарами в интернет-магазине, а также функционал для пользователей, включающий авторизацию, добавление товаров в корзину, оформление заказов и оставление отзывов.

Примеры ПО, используемого в проекте для разработки и проектирования

1. **Язык программирования**: Используется PHP для разработки бэкенда с помощью Laravel, а также JavaScript для фронтенда с использованием Vue.js.
2. **Фреймворки и библиотеки**: Bootstrap используется для стилизации пользовательского интерфейса, Vue.js для создания интерактивных компонентов на фронтенде, а также Laravel для быстрой разработки API.
3. **Инструменты разработки**: Для написания и отладки кода используются среды разработки PHPStorm и WebStorm, а для управления версиями кода - система контроля версий Git.
4. **Тестирование**: Для тестирования API и взаимодействия с ним используется Postman, а также инструменты Laravel для модульного и функционального тестирования.
5. **Сервер**: Для хостинга веб-приложения и его API используется сервер, настроенный для работы с PHP и Laravel на Open Server Panel.
6. **Базы данных**: Данные о товарах, пользователях, заказах и отзывах хранятся в реляционной базе данных, такой как MySQL.

# ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметной областью базы данных является информационная система аниме интернет-магазина. Эта система позволяет управлять и отслеживать различные аспекты работы магазина, включая информацию о доступных товарах, клиентах, заказов, сотрудниках. В магазине предоставляются услуги по продаже товаров (манга, фигурки, кружки и т. д), и данная база данных помогает магазину эффективно управлять своей деятельностью и взаимодействовать с клиентами.

Общее описание проекта

Все пользователи системы подразделяются на две группы:

1. Администраторы
2. Пользователи

Категории товаров подразделяются на несколько групп

1. Манга
2. Кружки
3. Дакимакуры
4. Значки

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
* Оставление отзыва для определённого товара
* Редактирование корзины
* Редактирование своего профиля

Функционал администратора

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

При попытке доступа к защищенным авторизацией функция системы во всех запросах необходимо возвращать ответ следующего вида:  
Status: 403

Content-Type: application/json

Body:

{

“error”: {

“code”: 403,

“message”: “Login failed”

}

}

## Функционал пользователя

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 поле, формат число;
* email – обязательное поле, строка;

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/register  **Method:** POST  **Headers:** Content-Type: application/json  **Body:**  {  "surname": "Евген",  "name": "Евгений",  "patronymic": "Евгеньев",  "login": "log12",  "password": "Pass1!",  "birth": "12.02.2003",  "telephone": "+7930573565",  "email": "asdadda@mail.ru"  } | Успешная регистрация  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Регистрация прошла успешно"  }  }  Ошибки валидации полей  **Формат ответа из общих требований**  {      "success": false,      "code": 422,      "message": "",      "errors": {          "name": [              "Поле \"Имя\" обязательно для заполнения."          ],          "surname": [              "Поле \"Фамилия\" обязательно для заполнения."          ],          "login": [              "Поле \"Логин\" должно содержать не менее 5 символов."          ],          "password": [              "Поле \"Пароль\" должно содержать не менее 8 символов.",              "Пароль должен содержать как минимум одну цифру, одну заглавную букву, одну малую букву и один специальный символ."          ],          "birth": [              "Поле \"Дата рождения\" обязательно для заполнения."          ],          "email": [              "Поле \"Электронная почта\" обязательно для заполнения."          ],          "telephone": [              "Поле \"Телефон\" обязательно для заполнения."          ]      }  } |

Авторизация

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/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}}/api/categories  **Method:** GET  **Headers:**  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": [  {  "id": 1,  "name": "Манга",  "created\_at": null,  "updated\_at": null,  },  {  "id": 2,  "name": "Кружки"  "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}}/api/category/{id}  **Method:** GET  **Headers:**  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "data": [          {              "id": 1,              "name": "Берсерк. Том 1.",              "description": "",              "price": "1174.00",              "quantity": 10,              "photo": "uploads/1/1.webp",              "category\_id": 1,              "created\_at": null,              "updated\_at": "2024-03-08T00:00:00.000000Z"          },          {              "id": 2,              "name": "Берсерк. Том 2.",              "description": "",              "price": "1245.00",              "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}}/api/category/product/{id}  **Method:** GET  **Headers:**  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "data": [          {              "id": 1,              "name": "Берсерк. Том 1.",              "description": "",              "price": "1174.00",              "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}}/api/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}}/api/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}}/api/cart  **Method:** GET  **Headers:** Authorization: Bearer {{api\_token}}  **Body:**  {  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {      "cart\_items": [          {              "id": 8,              "quantity": 1,              "price": "1174.00",              "user\_id": 1,              "product\_id": 1,              "created\_at": "2024-03-09T00:00:00.000000Z",              "updated\_at": "2024-03-09T00:00:00.000000Z"          },          {              "id": 9,              "quantity": 1,              "price": "1245.00",              "user\_id": 1,              "product\_id": 2,              "created\_at": "2024-03-09T00:00:00.000000Z",              "updated\_at": "2024-03-09T00:00:00.000000Z"          }      ]  } |

Добавление товара в корзину

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/category/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"  }  Ошибки валидации полей  **Формат ответа из общих требований** |

Оформление заказа

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/checkout  **Method:** POST  **Headers:** Authorization: Bearer {{api\_token}},  Content-Type: application/json  **Body:**  {  "cart\_items": [  {  "product\_id": 1,  "quantity": 2  },  {  "product\_id": 3,  "quantity": 1  }  ],  "address": "ул. Новая 5-15",  "payment\_id": 1  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "order\_id": 123,  "dateOrder": 2023-09-16,  "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"  } |

Оставление отзыва для определённого товара

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/category/product/{id}/review  **Method:** POST  **Headers:** Authorization: Bearer {{api\_token}},  Content-Type: application/json  **Body:**  {  "rating": 4,  "textReview": "Отличный товар!"  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Отзыв успешно добавлен"  }  }  Ответ при попытке оставить отзыв на не купленный товар  **Status:** 403  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 404,      "message": "Вы не можете оставить отзыв на товар, который вы не покупали"  }  Ответ при попытке оставить отзыв на уже созданный отзыв по данному товару  **Status:** 403  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 403,      "message": "Вы уже оставили отзыв на этот товар"  } |

Редактирование корзины

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/cart  **Method:** PATCH  **Headers:** Authorization: Bearer {{api\_token}},  Content-Type: application/json  **Body:**  {  "items": [  {  "product\_id": 1,  "quantity": 3  },  {  "product\_id": 2,  "quantity": 2  }  ]  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Корзина успешно отредактирована"  }  }  Ответ при ошибке  **Status:** 400  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 404,      "message": "Товар не найден в вашей корзине"  } |

Редактирование своего профиля

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/profile/  **Method:** PATCH  **Headers:** Authorization: Bearer {{api\_token}},  Content-Type: application/json  **Body:**  {  "name": "Новое Имя",  "surname": "Новая Фамилия",  И тд.  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Профиль успешно обновлен"  }  } |

## Функционал администратора

Добавление категории

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

* name – обязательное поле, строка;

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/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"  } |

Добавление товара

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

* name – обязательное 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}}/api/admin/product/create  **Method:** POST  **Headers:** Authorization: Bearer {{admin\_api\_token}},  Content-Type: application/json  **Body:**  {  "name": "Новый Товар",  "price": 1500,  "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"  } |

Редактирование категории

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/admin/category/{id}/edit  **Method:** PATCH  **Headers:** Authorization: Bearer {{admin\_api\_token}}, Content-Type: application/json  **Body:**  {  "name": "Новая Электроника"  } | Успешный ответ  **Status:** 200  **Content-Type:** application/json  **Body:**  {  "data": {  "message": "Категория успешно отредактирована"  }  }  Ответ при ошибке  **Status:** 400  **Content-Type:** application/json  **Body:**  {      "success": false,      "code": 422,      "message": "Продукт с таким именем уже существует"  } |

Редактирование товара

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/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."  }  } |

Удаление категории

|  |  |
| --- | --- |
| Request | Response |
| **URL:** {{host}}/api/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}}/api/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."  }  } |

# ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСНОЙ ЧАСТИ

Тут картинки

# ОПИСАНИЕ СУБД

Для хранения данных в проекте была выбрана СУБД phpMyAdmin. Основные причины выбора включают:

1. **Простота использования**: PhpMyAdmin предоставляет интуитивно понятный веб-интерфейс для работы с базами данных, что упрощает управление данными даже для пользователей без специальных навыков в области баз данных.
2. **Обширные возможности**: PhpMyAdmin предоставляет широкий набор функций для администрирования баз данных, включая создание, удаление и изменение таблиц, выполнение SQL-запросов, управление пользователями и привилегиями доступа.
3. **Бесплатность и открытый исходный код**: PhpMyAdmin является бесплатным и имеет открытый исходный код, что позволяет свободно использовать и модифицировать его под свои потребности без дополнительных затрат.
4. **Широкое распространение**: PhpMyAdmin широко используется в сообществе разработчиков благодаря своей популярности и долгой истории развития, что обеспечивает доступность обучающих материалов и ресурсов для поддержки.

Выбор phpMyAdmin обусловлен его простотой в использовании, богатым функционалом и широким распространением, что делает его подходящим решением для хранения и управления данными в проекте.

## КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ ДАННЫХ

Концептуальная модель – это отражение предметной области, для которой разрабатывается база данных. Не вдаваясь в теорию, отметим, что это некая диаграмма с принятыми обозначениями элементов. Все объекты, обозначающие вещи, обозначаются в виде прямоугольника. Атрибуты, характеризующие объект - в виде овала, а связи между объектами - ромбами. Мощность связи обозначаются стрелками (в направлении, где мощность равна многим - двойная стрелка, а со стороны, где она равна единице - одинарная).

Анализируя предметную область, можно выделить следующие сущности с их атрибутами (см. рисунок 1):

1. Сущность «Роль» с атрибутом «Название».
2. Сущность «Пользователи» с атрибутами «Имя», «Фамилия», «Отчество», «Логин», «Пароль», «Дата рождения», «Email», «Телефон», «Токен».
3. Сущность «Отзывы» с атрибутом «Текст».
4. Сущность «Заказы» с атрибутом «Адрес».
5. Сущность «Статус» с атрибутом «Название».
6. Сущность «Состав заказа» с атрибутами «Количество», «Общая стоимость».
7. Сущность «Способ оплаты» с атрибутами «Название».
8. Сущность «Товары» с атрибутами «Название», «Описание», «Цена», «Количество», «Фото».
9. Сущность «Категории» с атрибутами «Название».
10. Сущность «Корзины» с атрибутами «Количество», «Общая стоимость».

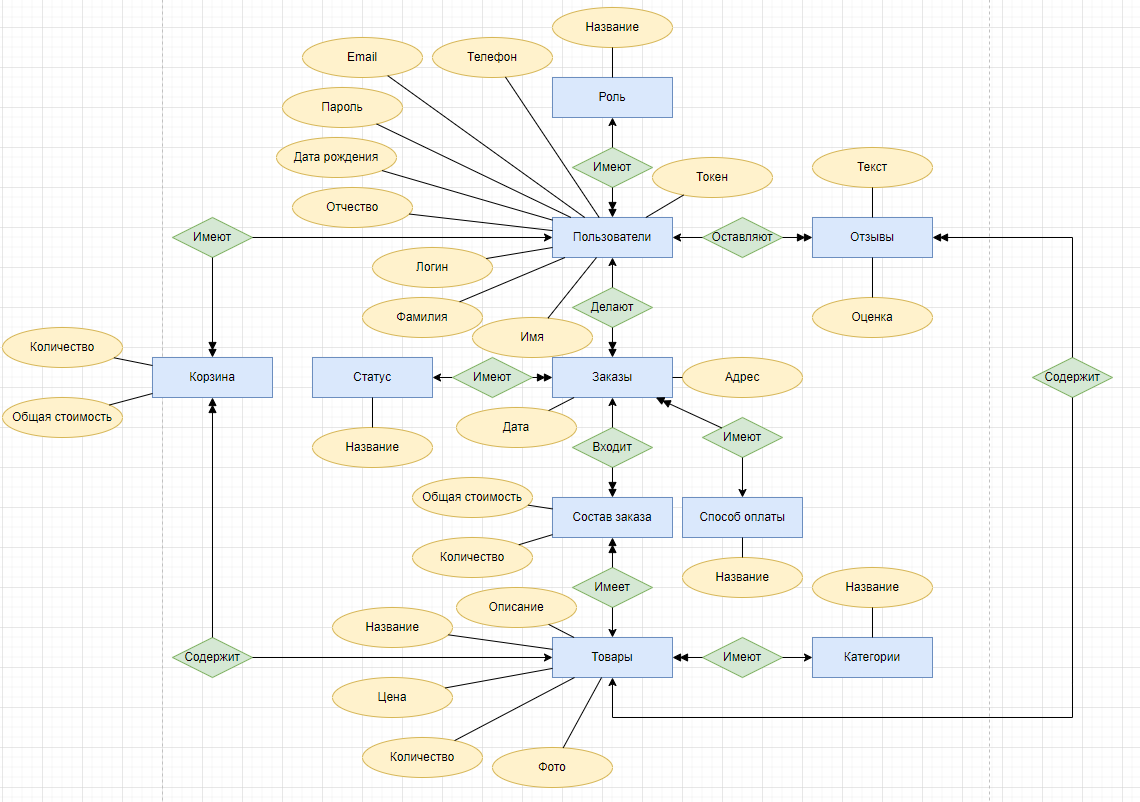


Рисунок 1 – Концептуальная модель данных

## НОРМАЛИЗАЦИЯ ДАННЫХ

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

***Первая нормальная форма (1NF)***

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

***Вторая нормальная форма (2NF)***

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый не ключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа. Функционально полная зависимость означает, что если потенциальный ключ является составным, то атрибут зависит от всего ключа и не зависит от его частей.

***Третья нормальная форма (3NF)***

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых.

При проектировании БД третья нормальная форма схем отношений достаточна в большинстве случаев, и приведением к третьей нормальной форме процесс проектирования реляционной базы данных обычно заканчивается. Однако иногда полезно продолжить процесс нормализации.

## ЛОГИЧЕСКАЯ МОДЕЛЬ ДАННЫХ

Логическая модель базы данных — схема базы данных, выраженная в понятиях модели данных. Этим отличается от концептуальной модели, описывающей семантику предметной области без указания технологии (конкретных методов реализации), и от физической модели, которая описывает конкретные физические механизмы, применяемые для хранения данных в накопителях.

Используя нашу концептуальную модель данных и выполнив нормализацию данных до 3 нормальной формы включительно, построим логическую модель данных, в которой мы можем выделить новые атрибуты, поля являющиеся первичными и внешними ключами, необходимые для построения связей.

Получившиеся логическая модель данных представлена на рисунке 2.

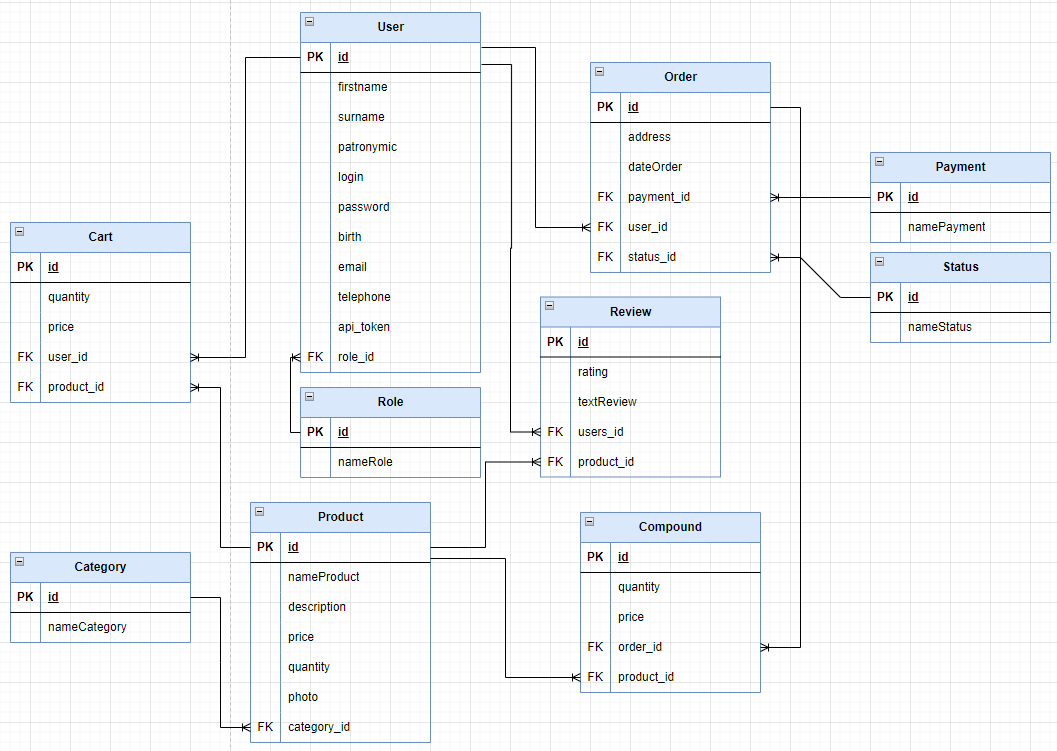


Рисунок 2 - Логическая модель данных

## ФИЗИЧЕСКАЯ МОДЕЛЬ ДАННЫХ

Физическая модель данных – это модель данных, описанная с помощью средств конкретной системы управления базами данных (СУБД). Физическая модель данных строится на базе даталогической путем добавления особенностей конкретной СУБД. К таким особенностям могут относиться поддерживаемые СУБД типы данных, соглашения о присвоении имен таблицам, атрибутам и т.д. Физическая модель данных фактически является готовым заданием на создание БД, имея которое можно реализовать БД в выбранной СУБД.

На рисунке 3 представленная физическая модель данных для информационной системы «Информационная система аниме интернет-магазина»

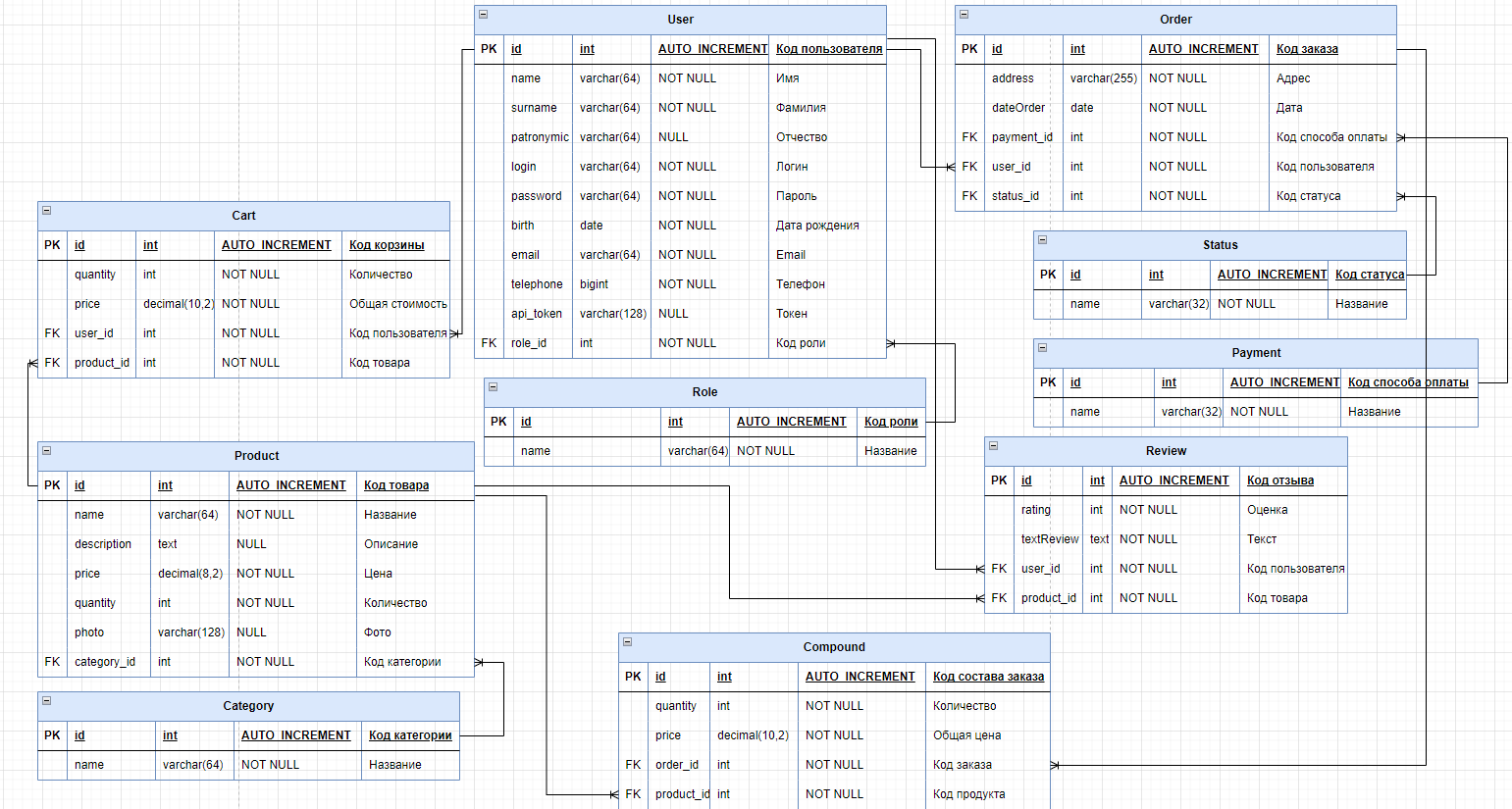


Рисунок 3 - Физическая модель данных

## СЛОВАРЬ ДАННЫХ

Таблица 1 – Словарь данных для БД «Информационная система аниме интернет-магазина»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Role** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID роли |
| name | varchar(64) | | Да | |  | Название |
| **User** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID пользователя |
| name | varchar(64) | | Да | |  | Имя |
| surname | varchar(64) | | Да | |  | Фамилия |
| patronymic | varchar(64) | | Нет | |  | Отчество |
| login | varchar(64) | | Да | |  | Логин |
| password | varchar(64) | | Да | |  | Пароль |
| birth | date | | Да | |  | Дата рождения |
| email | email | | Да | |  | Email |
| telephone | bigint | | Да | |  | Телефон |
| api\_token | int | | Да | |  | Токен |
| role\_id | int | | Да | | Внешний | ID роли |
| **Order** | | | | | | |
| **Поле** | **Формат** | **Обязательное** | | **Ключ** | | **Примечание** |
| id | int | Да | | Первичный | | ID заказа |
| address | varchar(255) | Да | |  | | Адрес |
| dateOrder | date | Да | |  | | Дата |
| discount\_id | int | Да | | Внешний | | ID скидки |
| payment\_id | int | Да | | Внешний | | ID способа оплаты |
| user\_id | int | Да | | Внешний | | ID пользователя |
| status\_id | int | Да | | Внешний | | ID статуса |
| **Review** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID отзыва |
| textReview | text | | Да | |  | Текст |
| user\_id | int | | Да | | Внешний | ID пользователя |
| product\_id | int | | Да | | Внешний | ID товара |
| **Payment** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID способа оплаты |
| name | varchar(32) | | Да | |  | Название |
| **Compound** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID состава заказа |
| quantity | int | | Да | |  | ID количества |
| price | decimal(10, 2) | | Да | |  | ID общей цены |
| order\_id | int | | Да | | Внешний | ID заказа |
| product\_id | int | | Да | | Внешний | ID товара |
| **Product** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID товара |
| name | varchar(64) | | Да | |  | Название |
| description | text | | Нет | |  | Описание |
| price | decimal(8,2) | | Да | |  | Цена |
| quantity | int | | Да | |  | Количество |
| photo | image | | Нет | |  | Фото |
| category\_id | int | | Да | | Внешний | ID категории |
| **Category** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID категории |
| name | varchar(164) | | Да | |  | Название |
| **Cart** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID корзины |
| quantity | int | | Да | |  | Количество |
| price | decimal(10,2) | | Да | |  | Общая стоимость |
| user\_id | int | | Да | | Внешний | ID пользователя |
| product\_id | int | | Да | | Внешний | ID товара |
| **Status** | | | | | | |
| **Поле** | **Формат** | | **Обязательное** | | **Ключ** | **Примечание** |
| id | int | | Да | | Первичный | ID статуса |
| name | varchar(32) | | Да | |  | Название |

# РАЗРАБОТКА ИНТЕРФЕЙСНОЙ ЧАСТИ

Для разработки интерфейсной части проекта было выбрано приложение WebStorm. Основные причины выбора включают:

1. **Знакомство и опыт**: WebStorm является одним из наиболее распространенных и популярных интегрированных сред разработки (IDE) для веб-разработки. Текущий опыт работы с этим инструментом позволяет максимально эффективно использовать его возможности.
2. **Поддержка технологий**: WebStorm обладает широким спектром функций и инструментов, специально разработанных для работы с веб-технологиями, такими как HTML, CSS, JavaScript и фреймворками, включая Vue.js.
3. **Удобство и производительность**: WebStorm предоставляет удобный и интуитивно понятный интерфейс, обеспечивающий высокую производительность и удобство в работе. Встроенные инструменты автодополнения, отладки и рефакторинга упрощают процесс разработки и повышают качество кода.
4. **Поддержка сообщества и обновления**: WebStorm активно поддерживается разработчиками и имеет активное сообщество пользователей, что обеспечивает доступность обновлений и плагинов, а также возможность получения помощи и поддержки при необходимости.

Выбор WebStorm обусловлен его широкими возможностями, удобством использования, поддержкой технологий и опытом работы с этим инструментом, что делает его подходящим выбором для разработки интерфейсной части проекта.

# РАЗРАБОТКА

Был разработан общий функционал, а именно вывод первых 3 товаров для каждой категории

|  |
| --- |
| Тут код и т.д и т.п и html и js |