Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и

вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  \_\_\_\_\_\_\_\_\_\_\_ З.З. Курмашева  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

РАЗРАБОТКА СЕРВИСА C2C ТОРГОВЛИ ТРЕХМЕРНЫМИ МОДЕЛЯМИ

Пояснительная записка к дипломному проекту

|  |  |
| --- | --- |
| Рецензент  А.А. Ковалева  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | Руководитель  Е.К. Дмитриева  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |
|  | Выпускник гр. 20ВЕБ-2  И.Е. Петунин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

2024

АННОТАЦИЯ

Пояснительная записка к дипломному проекту содержит постановку и программу решения задачи «Разработка сервиса С2С торговли трехмерными моделями».

Веб-приложение «Экземпляр» разработано с помощью веб-технологий PHP, HTML, CSS, JavaScript с использованием СУБД MySQL, предназначена для работы в браузерах любой операционной системы.

СОДЕРЖАНИЕ

лист

[ВВЕДЕНИЕ 3](#_Toc168404295)

[1 Проектирование информационной системы 5](#_Toc168404296)

[1.1 Описание предметной области 5](#_Toc168404297)

[1.2 Описание входной информации 7](#_Toc168404298)

[1.3 Описание выходной информации 7](#_Toc168404299)

[1.4 Концептуальное проектирование 8](#_Toc168404300)

[1.5 Логическое проектирование 8](#_Toc168404301)

[1.6 Описание структуры базы данных 8](#_Toc168404302)

[1.7 Общие требования к программному продукту 11](#_Toc168404303)

[2 Экспериментальный раздел 13](#_Toc168404304)

[2.1 Описание программы 13](#_Toc168404305)

[2.2 Тестирование программного обеспечения 20](#_Toc168404306)

[2.3 Руководство пользователя 25](#_Toc168404307)

[ЗАКЛЮЧЕНИЕ 33](#_Toc168404308)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc168404309)

[Приложение А Диаграмма вариантов использования 37](#_Toc168404310)

[Приложение Б Логическая структура программы 38](#_Toc168404311)

[Приложение В Листинг программного кода 39](#_Toc168404312)

# ВВЕДЕНИЕ

Стремительное развитие информационных технологий позволило возникнуть достаточно большому числу различных веб-сервисов электронной коммерции. Наиболее интересным направлением электронной коммерции с точки зрения покупателя являются C2C (Consumer-to-consumer) сервисы в сети Интернет.

Consumer-to-consumer — это модель бизнеса, в которой взаимодействие и торговля происходят непосредственно между самими потребителями. В этой модели нет присутствия традиционных компаний-поставщиков. Вместо этого, продавцы и покупатели устанавливают прямые отношения и осуществляют торговлю напрямую.

Эта модель является ключевым элементом в децентрализованной экономике, в которой компании и посредники в значительной степени исключены из процесса. C2C бизнес-модель минует сложные корпоративные структуры и транзакционные промежуточные этапы.

В результате, потребители лично проводят взаимодействие, обмениваясь товарами и услугами с учетом собственных потребностей и предпочтений. Эта модель предоставляет уникальные возможности самореализации для потребителей, при этом создавая инновационную платформу для эффективных и взаимовыгодных торговых отношений.

Зачастую в такого рода коммерческих взаимоотношениях присутствует посредник, организующий торговую площадку, например, интернет-аукцион, сайт-объявлений и так далее.

Посредник может являться гарантом проведения платежа, получения товара, а также, в некоторых случаях, может влиять на разрешение спорных ситуаций.

К достоинствам схемы С2С можно отнести низкие трансакционные издержки, более низкую цену за товар.

Недостаток — повышенная вероятность мошенничества. Для предотвращения мошенничества площадки вводят систему репутации.

Целью данной работы является разработка функционального сервиса для Consumer-to-consumer торговли трехмерными моделями. Для достижения данной цели были поставлены следующие задачи:

* изучить предметную область;
* определить требования к разрабатываемому сервису;
* выполнить проектирование сервиса;
* реализовать сервис на основе выбранных технологий;
* провести тестирование.

Сервис, предоставляющий услуги купли-продажи, может быть востребован во многих областях: от сферы дизайна, до игровой индустрии.

Предложенный сервис позволит пользователям быстро и удобно продавать модели, а также находить необходимые модели для своих проектов. Продавцам сервис предоставит удобную платформу для продажи и распространения моделей, что приведет к увеличению доходов и привлечению новых покупателей. Для клиентов, в свою очередь, сервис предоставит удобную систему поиска и покупки нужных моделей, а также возможность оценки качества моделей и надежности продавца по отзывам других покупателей.

# 1 Проектирование информационной системы

1.1 Описание предметной области

Основные бизнес-процессы, реализуемые на предприятии организации C2C торговли трехмерными моделями:

* предоставление посреднических услуг, в т.ч. услуг гаранта и организатора торгов;
* модерация информации о новых моделях;
* оказание технической поддержки.

В рамках дипломного проекта автоматизируется бизнес-процесс предоставления посреднических услуг.

Основными пользователями системы выступают участники торгов, которые могут быть как покупателями, так и продавцами, а также администратор системы.

Функции участников торгов:

* создание объявлений для новых моделей;
* редактирование объявлений;
* ограничение видимости объявлений;
* покупка моделей других участников;
* просмотр купленных моделей;
* загрузка своих или купленных моделей;
* написание отзывов.

Функции администратора:

* проверка корректности объявлений и отзывов;
* отправка объявлений и отзывов на повторное редактирование;
* публикация объявлений и отзывов;
* ограничение видимости объявлений и отзывов;
* блокировка пользователей за нарушение правил сервиса.

Бизнес-правила, которые будут основой для задания ограничений при проектировании и реализации системы:

* цена моделей не должна быть отрицательной или выше 50000₽;
* пользователь может оплатить модель СБП-переводом;
* пользователь не видит неопубликованные и отклоненные объявления;
* для покупки моделей пользователь должен зарегистрироваться в системе;
* пользователь не может купить собственную модель.

Основные задачи, которые решает разрабатываемая информационная система:

* регистрация и авторизация пользователей;
* создание, редактирование, исключение из поисковой выдачи объявлений о продаже трехмерных моделей;
* покупка и продажа моделей пользователями;
* предоставление доступа к информации о ранее купленных моделях;
* модерирование объявлений администратором.

Основными объектами предметной области являются объявления о продаже моделей, заказы, а также отзывы.

Объявления о продаже моделей создаются пользователями и содержат название, описание, цену, фото и видео 3Д модели, которую пользователь хотел бы продать другим пользователям.

Модели могут быть куплены другими пользователями, при покупке создается заказ, в котором сохраняется время, цена, пользователь и соответствующее объявление. Купленные модели отображаются в личном кабинете пользователя с возможностью скачивания соответствующего файла.

На ранее купленные модели пользователи могут оставлять отзывы, в которых указываются оценка, текст, заголовок, а также метка, рекомендует пользователь данную модель или нет.

1.2 Описание входной информации

Описание входных файлов представлено в таблице 1.1.

Таблица 1.1 - Описание входных файлов

|  |  |  |  |
| --- | --- | --- | --- |
| Название файла | Шифр файла | Тип файла | Источник поступления |
| Модель | ВХ1 | Трехмерная модель | Загрузка пользователем |

Входной информацией, которая являются основанием для заполнения базы данных, являются:

* информация о пользователе (фамилия, имя, отчество, номер телефона);
* характеристики трехмерной модели (название, категория, описание, цена, ссылка на файл);
* фотографии и видео трехмерной модели.

1.3 Описание выходной информации

Выходными документами, в соответствии с таблицей 1.2, являются:

* список доступных трехмерных моделей для пользователя;
* информация о заказах пользователя (дата, сумма, способ оплаты);
* счет на оплату.

Таблица 1.2 - Описание выходных документов

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование документа (шифр) | Периодичность выдачи документа | Кол-во экз. | Куда передаются |
| ВЫХ01\_трехмерные\_модели\_пользователя.pdf | По запросу | 1 | Пользователю в виде pdf файла |
| ВЫХ02\_заказы\_пользователя.pdf | По запросу | 1 | Пользователю в виде pdf файла |

Продолжение таблицы 1.2.

|  |  |  |  |
| --- | --- | --- | --- |
| ВЫХ03\_счёт\_на\_оплату.pdf | После создания заказа | 1 | Пользователю в виде  pdf файла |

1.4 Концептуальное проектирование

Диаграмма вариантов использования представлена в приложении А на рисунке А.1.

1.5 Логическое проектирование

На рисунке 1.2 представлена диаграмма связей сущностей (ERD диаграмма) базы данных для разрабатываемой системы.

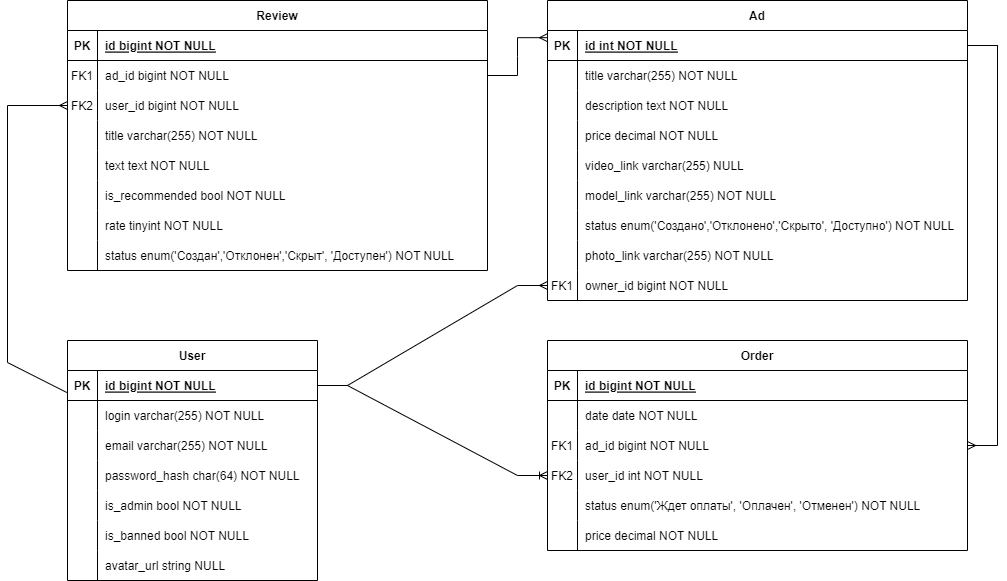


Рисунок 1.2 – Диаграмма связей сущностей

1.6 Описание структуры базы данных

В таблицах 1.3-1.6 представлено описание структуры базы данных – все существующие таблицы и поля.

Таблица 1.3 – Users (Список пользователей)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| id | Уникальный идентификатор пользователя | INTEGER | первичный ключ |
| login | Уникальное имя пользователя | VARCHAR(255) | обязательное поле |
| email | Уникальный email пользователя | VARCHAR(255) | обязательное поле |
| password\_hash | Хэш пароля | CHAR(64) | обязательное поле |
| is\_admin | Метка принадлежности пользователя роли Администратор | BOOLEAN | обязательное поле, по умолчанию = False |
| is\_banned | Метка блокировки пользователя | BOOLEAN | обязательное поле, по умолчанию = False |
| avatar\_url | Относительный путь до аватарки пользователя в статичных файлах сервера | STRING | обязательное поле, по умолчанию |

Таблица 1.4 – Ads (Список объявлений)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| id | Уникальный идентификатор | INTEGER | Первичный ключ |
| title | Заголовок объявления | VARCHAR(255) | обязательное поле |
| price | Цена модели | DECIMAL | обязательное поле |
| video\_link | URL ссылка на видео с демонстрацией модели | VARCHAR(255) | необязательное поле |
| model\_link | Относительный путь до файла модели | VARCHAR(255) | обязательное поле |

Продолжение таблицы 1.4.

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| status | Статус объявления (Создано, Отклонено, Скрыто, Доступно) | ENUM(4) | обязательное поле |
| owner\_id | Идентификатор создателя объявления | INTEGER | внешний ключ (к Users) |

Таблица 1.5 – Reviews (Список отзывов)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| id | Уникальный идентификатор отзыва | INTEGER | первичный ключ |
| ad\_id | Идентификатор модели | INTEGER | внешний ключ (к Ads) |
| user\_id | Идентификатор автора отзыва | INTEGER | внешний ключ (к Users) |
| title | Заголовок отзыва | VARCHAR(255) | обязательное поле |
| text | Текст отзыва | TEXT | обязательное поле |
| is\_recommended | Метка наличия или отсутствия рекомендации | BOOLEAN | обязательное поле |
| rate | Оценка | TINYINT | обязательное поле, >= 1, <= 5 |
| status | Статус отзыва (Создан, Отклонен, Скрыт, Доступен), при создании нового объекта устанавливается «Создан» | ENUM | обязательное поле |

Таблица 1.6 – Orders (Список заказов)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| id | Уникальный идентификатор | INTEGER | первичный ключ |
| date | Дата заказа | DATE | обязательное поле |
| ad\_id | Идентификатор купленной модели | INTEGER | внешний ключ (к Ads) |
| user\_id | Идентификатор покупателя | INTEGER | внешний ключ (к Users) |
| status | Статус заказа (Ждет оплаты, Оплачен, Отменен) | ENUM | обязательное поле |
| price | Расчетное поле – цена модели в момент покупки | DECIMAL | обязательное поле |

1.7 Общие требования к программному продукту

Целевой задачей является обеспечение удобной и безопасной среды для обмена трехмерными моделями между пользователями путем создания онлайн-платформы для обмена и продажи трехмерных моделей между пользователями.

Среди задач проекта можно выделить разработку удобного интерфейса для загрузки, просмотра и покупки трехмерных моделей; обеспечение безопасных транзакций между пользователями; поддержка различных форматов трехмерных файлов.

Требования к надежности и эффективности: гарантированная сохранность данных пользователей, высокая производительность при загрузке и просмотре моделей, защита от мошенничества.

Информационная система должна содержать функции регистрации, авторизации пользователей, загрузки трехмерных моделей с указанием описания и цены, просмотр 3D-превью моделей, оценка моделей, оформление заказа.

Для ввода продукта в эксплуатацию необходимы следующие технические программные средства:

* четырехядерный процессор с тактовой частотой от 2,2 ГГц;
* 512 Мб основной памяти;
* 1 Гб оперативной памяти;
* Ethernet-адаптер.

Процедуры сохранения данных: регулярное резервное автоматическое копирование базы данных, защищенное хранение личных данных пользователей.

Проверка достоверности данных: подтверждение электронной почты при регистрации, проверка введенных значений перед сохранением объявлений.

Защита от ошибок: предупреждение о возможных ошибках при загрузке файлов, возможность отмены заказа.

2 Экспериментальный раздел

2.1 Описание программы

Серверная часть программы реализована на языке PHP с использованием фреймворка Laravel, визуальная часть – CSS, HTML5, JS. Веб-приложение было написано с использованием СУБД MySQL.

Система предназначена для пользователя C2C-площадки и для администратора портала. Логическая структуры представлены в приложении Б на рисунке Б.1.

Описание модулей программы представлено в таблице 2.1.

Таблица 2.1 – Описание модулей программы

|  |  |  |
| --- | --- | --- |
| Файл | Модуль | Описание файла |
| MainController.php | \_\_get\_most\_popular\_ads() | Модуль отвечает за выбор из базы данных наиболее популярных объявлений. |
| \_\_get\_ten\_most\_popular\_ads() | Модуль отвечает за выбор из базы данных десяти наиболее популярных объявлений. |
| index() | Модуль отображает главную страницу сайта. |
| ads() | Модуль отображает страницу со списком моделей. |

Продолжение таблицы 2.1.

|  |  |  |
| --- | --- | --- |
| Файл | Модуль | Описание файла |
| MainController.php | buy() | Модуль отображает страницу покупки определенной модели. |
| rules() | Модуль отображает страницу с информацией о правилах сервиса. |
| ad\_detail() | Модуль отображает страницу с информацией об объявлении |
| user\_detail() | Модуль отображает страницу с информацией об авторе. |
| HomeController.cs | check\_auth\_user() | Модуль проверяет, авторизован ли текущий пользователь |
| index() | Модуль отображает личный кабинет пользователя |
| own\_ads() | Модуль отображает список объявлений пользователя. |

Продолжение таблицы 2.1

|  |  |  |
| --- | --- | --- |
| Файл | Модуль | Описание файла |
| HomeController.cs | bought\_ads() | Модуль отображает список купленных моделей пользователя. |
| reviews() | Модуль отображает страницу со списком отзывов пользователя. |
| stats() | Модуль отображает страницу со статистикой по продажам пользователя. |
| save\_data() | Модуль сохраняет данные пользователя после изменения. |
| ads\_create\_page() | Модуль отображает страницу создания нового объявления. |
| ads\_create\_method() | Модуль сохраняет новое объявление в базу данных. |
| ads\_edit\_page() | Модуль отображает страницу редактирования объявления. |

Продолжение таблицы 2.1

|  |  |  |
| --- | --- | --- |
| Файл | Модуль | Описание файла |
| HomeController.cs | ads\_edit\_method() | Модуль сохраняет объявление после редактирования. |
| ads\_hide\_method() | Модуль меняет статус объявления на «Скрыто». |
| ads\_show\_method() | Модуль меняет статус объявления на «Отображается». |
| AdminController.cs | index() | Модуль отображает страницу личного кабинета администратора. |
| ad\_editor() | Модуль отображает страницу-редактор объявлений. |
| review\_editor() | Модуль отображает страницу-редактор отзывов. |
| review\_create() | Модуль отображает страницу создания отзыва. |
| review\_store() | Модуль сохраняет созданный отзыв в базе данных. |
| review\_delete() | Модуль удаляет отзыв. |

Продолжение таблицы 2.1

|  |  |  |
| --- | --- | --- |
| Файл | Модуль | Описание файла |
| AdminController.cs | review\_edit() | Модуль отображает страницу редактирования отзыва. |
| review\_update() | Модуль сохраняет отзыв после редактирования. |
| AuthController.php | register() | Модуль регистрирует нового пользователя. |
| login() | Модуль авторизует в системе пользователя. |
| logout() | Модуль позволяет выйти из системы. |
| Ad.php | user() | Модуль возвращает автора объявления. |
| orders() | Модуль возвращает заказы на модель. |
| reviews() | Модуль возвращает список отзывов к объявлению. |
| get\_average\_rating() | Модуль возвращает среднюю оценку по отзывам к объявлению. |

Продолжение таблицы 2.1

|  |  |  |
| --- | --- | --- |
| Файл | Модуль | Описание файла |
| Order.php | user() | Модуль возвращает пользователя, оформившего заказ. |
| ad() | Модуль возвращает объявление, на которое оформлен заказ. |
| Review.php | user() | Модуль возвращает пользователя, который оставил отзыв. |
| ad() | Модуль возвращает объявление, на которое был оставлен отзыв. |
| User.php | ads() | Модуль возвращает список объявлений пользователя. |
| orders() | Модуль возвращает список заказов пользователя. |
| bought\_ads() | Модуль возвращает список купленных пользователем моделей. |
| reviews() | Модуль возвращает список отзывов. |

Наглядно представим структуру взаимосвязей между компонентами разрабатываемой информационной системы с помощью модульной схемы сайта (рисунок 2.1).

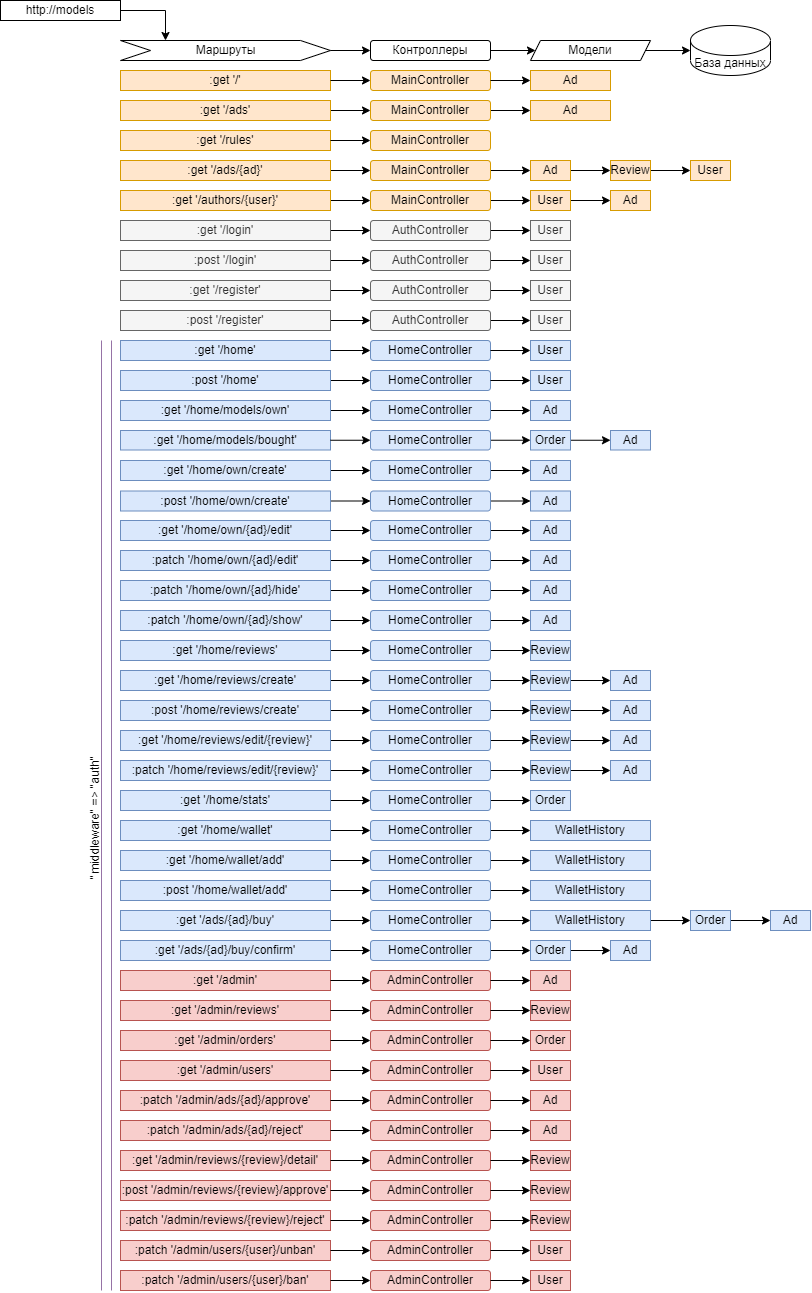


Рисунок 2.1 – Модульная схема сайта

2.2 Тестирование программного обеспечения

В ходе тестирования программного продукта на корректных и некорректных данных не были выявлены ошибки, влияющие на работу продукта.

Протокол тестирования входа при вводе некорректных данных представлен в таблице 2.3.

Таблица 2.3 – Тестирование входа при вводе некорректных данных

|  |  |
| --- | --- |
| Наименование | Описание |
| Дата теста | 08.05.2024 |
| Приоритет тестирования | Высокий |
| Заголовок | Проверка входа в систему при вводе некорректных данных. |
| Резюме тестирования | Необходимо добиться корректного поведения программы при вводе некорректных данных. |
| Этапы теста | Выбор поля заполнения;  Заполнение поля некорректными данными;  Нажатие на кнопку «Войти» |
| Тестовые данные | Email: [test@yandex.ru](mailto:test@yandex.ru)  Пароль: 123456 |
| Ожидаемый результат | Система должна вывести системное сообщение «Неверное имя пользователя или пароль» |
| Фактический результат | Система выводит системное сообщение «Неверное имя пользователя или пароль» |

Системное сообщение при вводе некорректных данных представлено на рисунке 2.1.

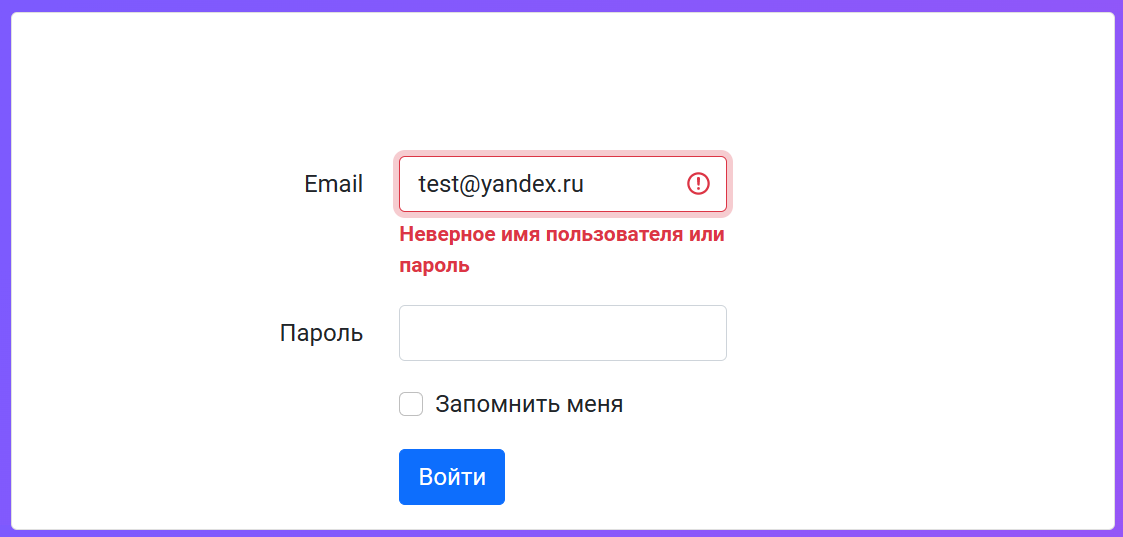


Рисунок 2.1 – Системное сообщение при вводе некорректных данных авторизации

Протокол тестирования регистрации при вводе корректных данных представлен в таблице 2.4.

Таблица 2.4 – Тестирование входа при вводе корректных данных

|  |  |
| --- | --- |
| Наименование | Описание |
| Дата теста | 08.05.2024 |
| Приоритет тестирования | Высокий |
| Заголовок | Проверка регистрации в системе при вводе корректных данных. |
| Резюме тестирования | Необходимо добиться корректного поведения программы. |
| Этапы теста | Переход на страницу регистрации;  Выбор поля заполнения;  Заполнение поля корректными данными;  Нажатие на кнопку «Зарегистрироваться» |

Продолжение таблицы 2.4

|  |  |
| --- | --- |
| Наименование | Описание |
| Тестовые данные | Имя пользователя: test  Email: [test2@yandex.ru](mailto:test2@yandex.ru)  Пароль: 12345678 |
| Ожидаемый результат | Система должна переадресовать пользователя в личный кабинет. |
| Фактический результат | Система переадресует пользователя в личный кабинет. |

Вид личного кабинета для только что зарегистрированного пользователя представлен на рисунке 2.2.

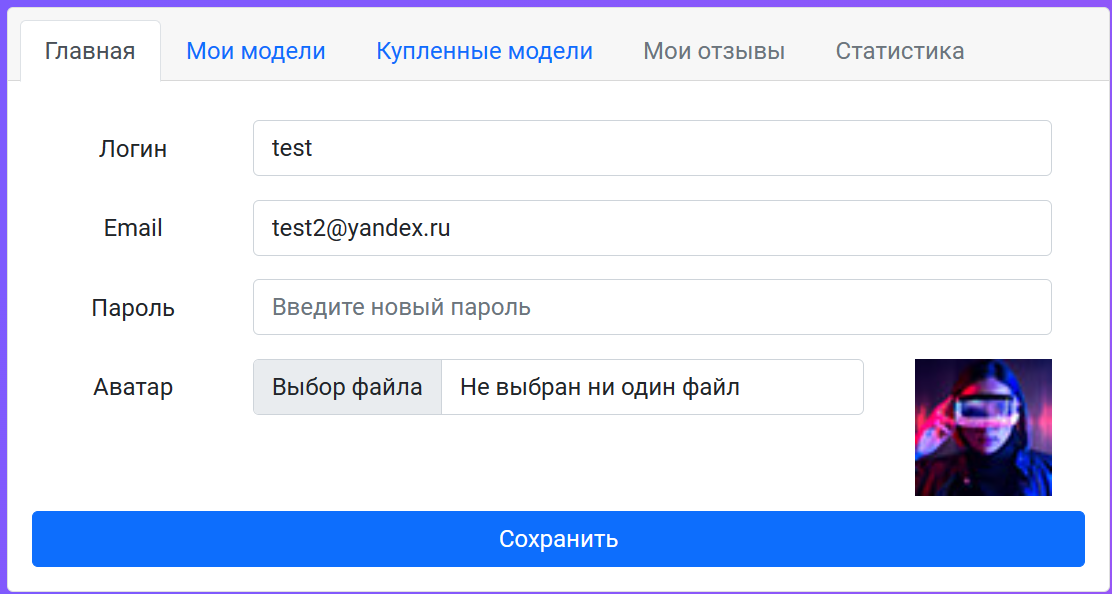


Рисунок 2.2 – Вид личного кабинета для нового пользователя

Протокол тестирования входа при вводе корректных данных представлен в таблице 2.5.

Таблица 2.5 – Тестирование входа при вводе корректных данных

|  |  |
| --- | --- |
| Наименование | Описание |
| Дата теста | 08.05.2024 |
| Приоритет тестирования | Высокий |
| Заголовок | Проверка входа в систему. |

Продолжение таблицы 2.5

|  |  |
| --- | --- |
| Наименование | Описание |
| Резюме тестирования | Необходимо добиться корректного поведения программы. |
| Этапы теста | Переход на страницу авторизации;  Заполнение поля корректными данными;  Нажатие на кнопку «Войти» |
| Тестовые данные | Email: [test2@yandex.ru](mailto:test2@yandex.ru)  Пароль: 12345678 |
| Ожидаемый результат | Система должна переадресовать пользователя в личный кабинет. |
| Фактический результат | Система переадресует пользователя в личный кабинет. |

Протокол тестирования создания нового объявления при вводе корректных данных представлен в таблице 2.6.

Таблица 2.6 – Тестирование создания нового объявления при вводе корректных данных

|  |  |
| --- | --- |
| Наименование | Описание |
| Дата теста | 08.05.2024 |
| Приоритет тестирования | Высокий |
| Заголовок | Проверка создания нового объявления. |
| Резюме тестирования | Необходимо добиться корректного поведения программы. |
| Этапы теста | Переход на страницу создания объявления;  Заполнение поля корректными данными;  Нажатие на кнопку «Создать» |

Продолжение таблицы 2.6

|  |  |
| --- | --- |
| Наименование | Описание |
| Тестовые данные | Название: Волк  Описание: Весьма достойная модель  Цена: 100  Видео: отсутствует  Модель: файл wolf.fbx  Фото: файл wolf.png |
| Ожидаемый результат | Система должна переадресовать пользователя на страницу «Мои модели», где выводится только что созданное объявление. |
| Фактический результат | Система переадресует пользователя на страницу «Мои модели», где выводится только что созданное объявление. |

Вид страницы «Мои модели» после добавления нового объявления представлен на рисунке 2.3.

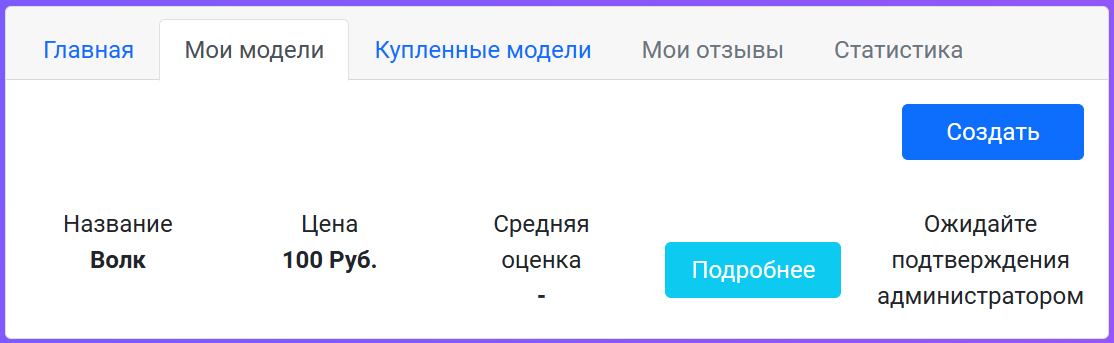


Рисунок 2.3 – Вид страницы «Мои модели» после создания нового объявления

Протокол тестирования создания нового объявления при вводе некорректных данных представлен в таблице 2.7.

Таблица 2.7 – Тестирование создания нового объявления при вводе некорректных данных

|  |  |
| --- | --- |
| Наименование | Описание |
| Дата теста | 08.05.2024 |
| Приоритет тестирования | Высокий |
| Заголовок | Проверка создания нового объявления при вводе некорректных данных. |
| Резюме тестирования | Необходимо добиться корректного поведения программы при вводе некорректных данных. |
| Этапы теста | Переход на страницу создания объявления;  Выбор поля заполнения;  Заполнение поля некорректными данными;  Нажатие на кнопку «Создать» |
| Тестовые данные | Название: отсутствует  Описание: отсутствует  Цена: 100  Видео: отсутствует |
| Ожидаемый результат | Система должна выводить системные сообщения у соответствующих полей:  «Поле title обязательно»,  «Поле description обязательно» |
| Фактический результат | Система выводит системные сообщения у соответствующих полей:  «Поле title обязательно»,  «Поле description обязательно» |

Протокол тестирования создания нового отзыва при вводе некорректных данных представлен в таблице 2.8.

Таблица 2.8 – Тестирование создания нового отзыва при вводе некорректных данных

|  |  |
| --- | --- |
| Наименование | Описание |
| Дата теста | 24.06.2024 |
| Приоритет тестирования | Высокий |
| Заголовок | Проверка создания нового отзыва при вводе некорректных данных. |
| Резюме тестирования | Необходимо добиться корректного поведения программы при вводе некорректных данных. |
| Этапы теста | Переход на страницу создания отзыва;  Выбор поля заполнения;  Заполнение поля некорректными данными;  Нажатие на кнопку «Создать» |
| Тестовые данные | Модель: Волк  Заголовок: отсутствует  Текст: отсутствует  Рекомендую ли я: Нет  Оценка: 1 |
| Ожидаемый результат | Система должна выводить системные сообщения у соответствующих полей:  «Поле заголовок обязательно»,  «Поле текст обязательно» |
| Фактический результат | Система выводит системные сообщения у соответствующих полей:  «Поле заголовок обязательно»,  «Поле текст обязательно» |

Протокол тестирования создания нового отзыва при вводе корректных данных представлен в таблице 2.9.

Таблица 2.9 – Тестирование создания нового отзыва при вводе корректных данных

|  |  |
| --- | --- |
| Наименование | Описание |
| Дата теста | 24.06.2024 |
| Приоритет тестирования | Высокий |
| Заголовок | Проверка создания нового отзыва при вводе корректных данных. |
| Резюме тестирования | Необходимо добиться корректного поведения программы при вводе корректных данных. |
| Этапы теста | Переход на страницу создания отзыва;  Выбор поля заполнения;  Заполнение поля корректными данными;  Нажатие на кнопку «Создать» |
| Тестовые данные | Модель: Волк  Заголовок: Все понравилось  Текст: Рекомендую к покупке  Рекомендую ли я: Да  Оценка: 5 |
| Ожидаемый результат | Система должна переадресовывать пользователя на страницу созданных отзывов, на которой отображается ранее созданный отзыв |
| Фактический результат | Система переадресует пользователя на страницу созданных отзывов, на которой отображается ранее созданный отзыв |

Протокол тестирования пополнения при вводе корректных данных представлен в таблице 2.10.

Таблица 2.10 – Тестирование пополнения при вводе корректных данных

|  |  |
| --- | --- |
| Наименование | Описание |
| Дата теста | 24.06.2024 |
| Приоритет тестирования | Высокий |
| Заголовок | Проверка пополнения кошелька при вводе корректных данных. |
| Резюме тестирования | Необходимо добиться корректного поведения программы при вводе корректных данных. |
| Этапы теста | Переход на страницу кошелька;  Нажатие на кнопку «Пополнение»;  Выбор поля заполнения;  Заполнение поля корректными данными;  Нажатие на кнопку «Пополнить» |
| Тестовые данные | Баланс: 10000 |
| Ожидаемый результат | Система должна переадресовывать пользователя на страницу кошелька с отображенной транзакцией в истории. |
| Фактический результат | Система переадресует пользователя на страницу кошелька с отображенной транзакцией в истории. |

2.3 Руководство пользователя

Веб-приложение «C2C-Market Экземпляр» предназначено для упрощения процесса взаимодействия между покупателями и продавцами 3D моделей.

При входе на сайт отобразится главная страница, содержащая список самых популярных моделей, а также навигационное меню (рисунок 2.4).

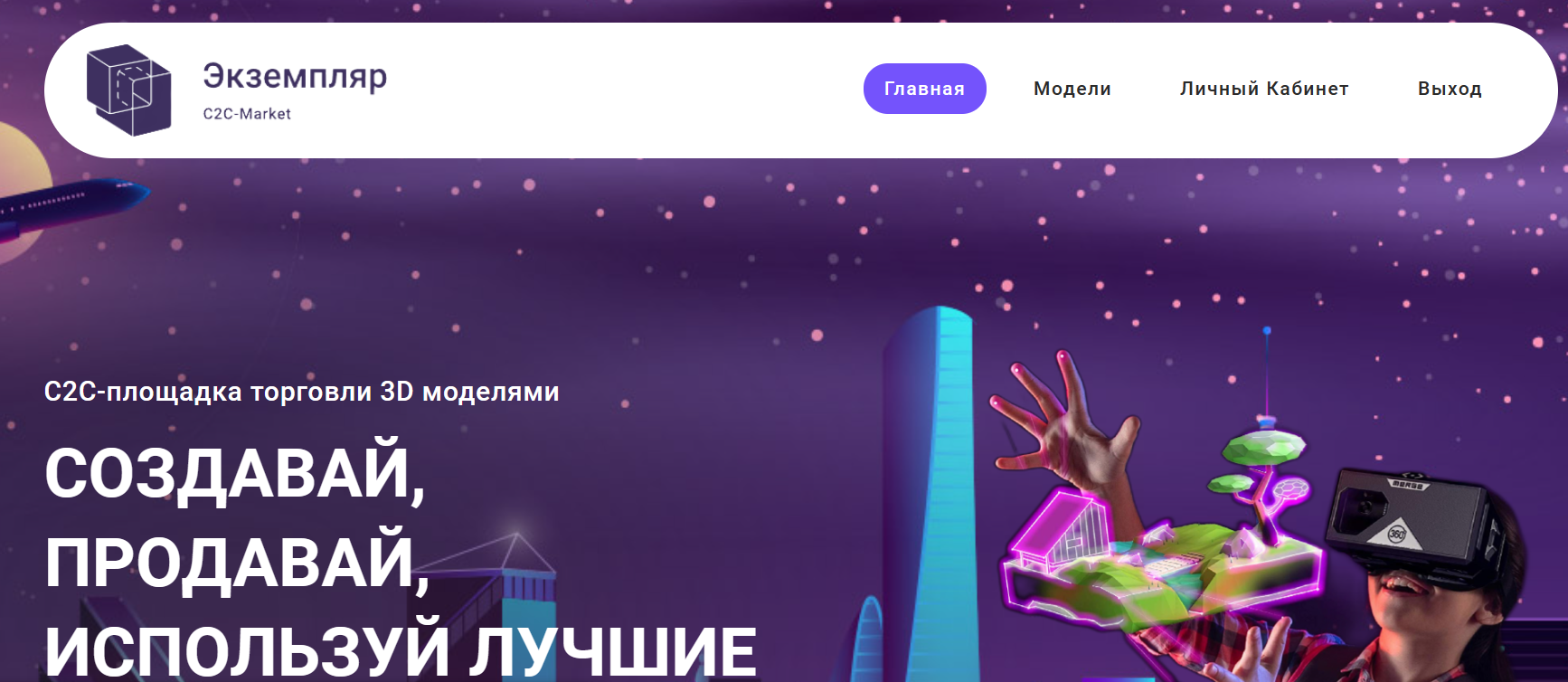


Рисунок 2.4 – Главная страница

Пользователь может выбрать существующую модель на главной странице и посмотреть информацию путем нажатия на кнопку «Подробнее» (рисунок 2.5).

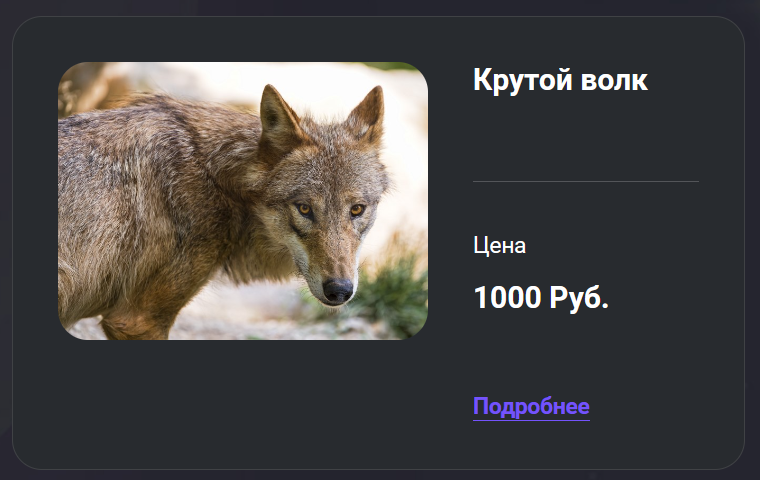


Рисунок 2.5 – Блок модели с кнопкой «Подробнее»

При нажатии на кнопку «Подробнее» отображается отдельная страница, содержащая более детальную информацию об объявлении (рисунок 2.6).

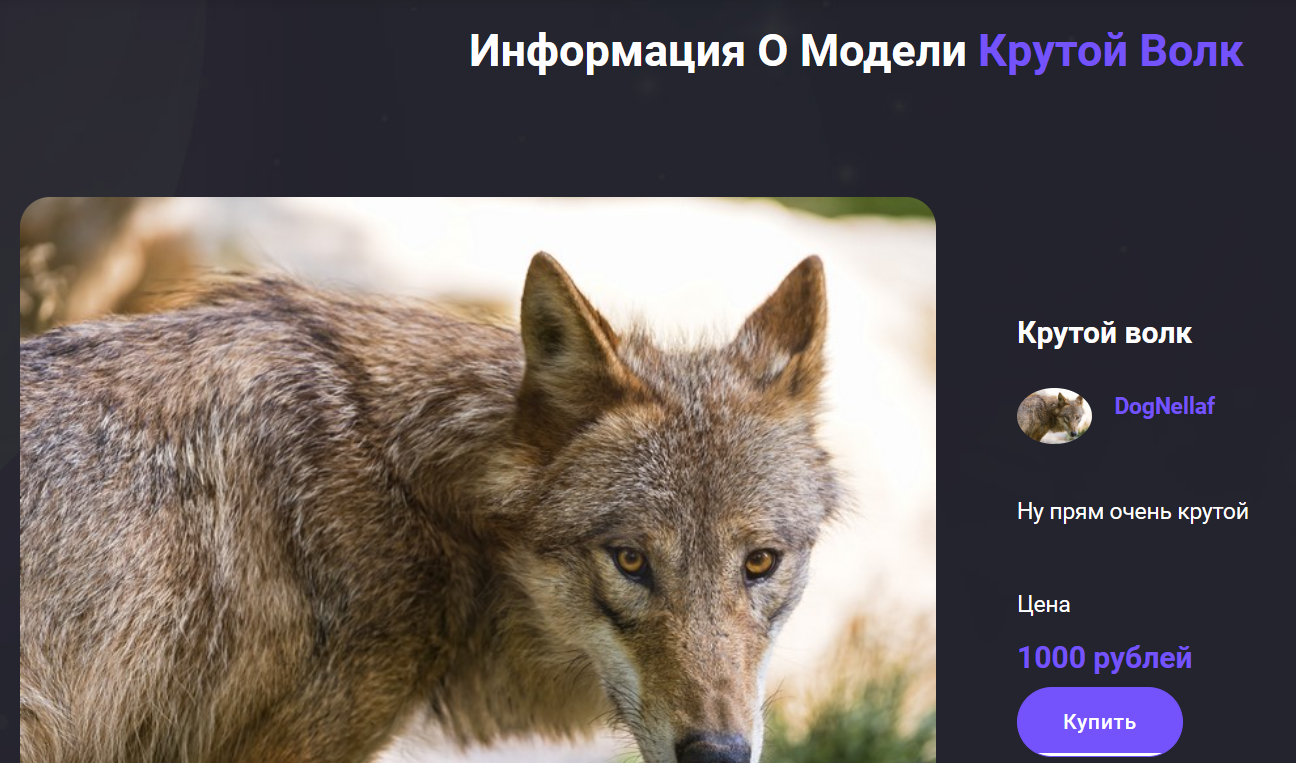


Рисунок 2.6 – Страница с информацией о модели

Для добавления своих моделей и покупки моделей других пользователей необходимо зарегистрироваться. Для этого необходимо войти на соответствующую страницу путем нажатия кнопки «Регистрация» в навигационном меню (рисунок 2.7).

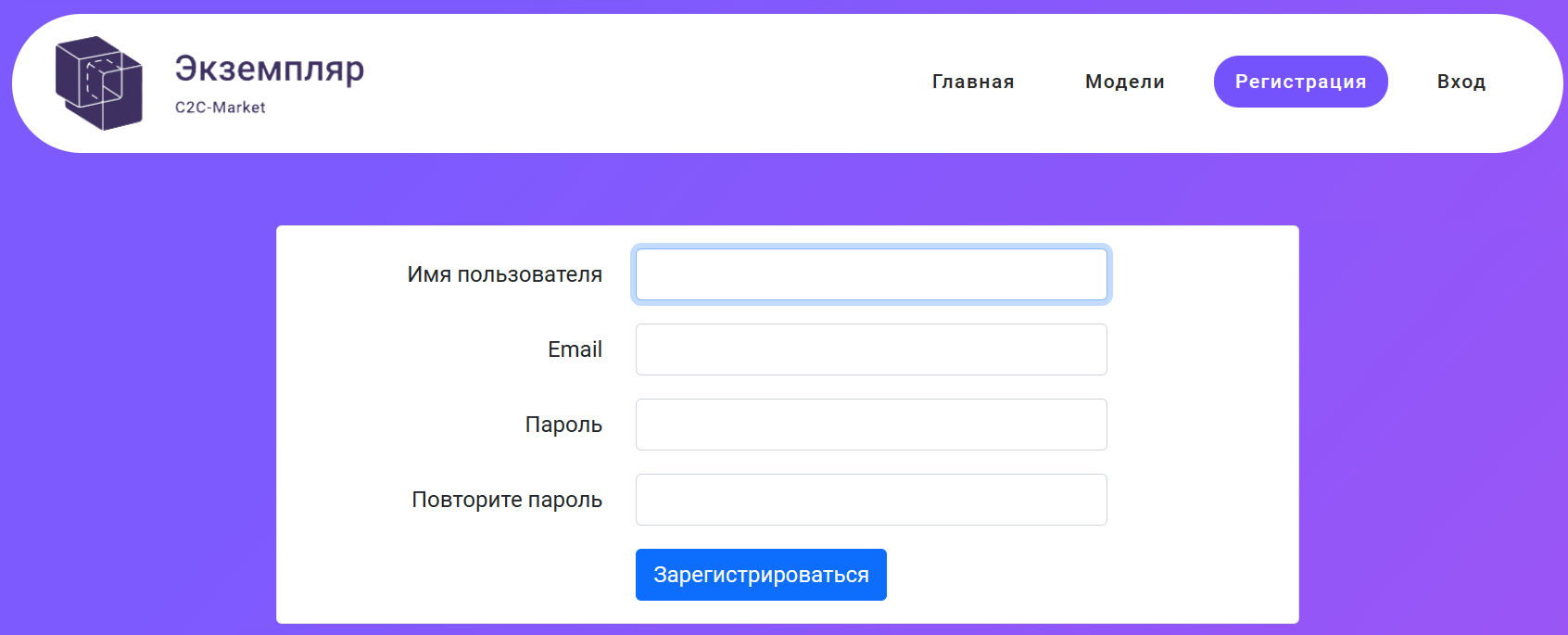


Рисунок 2.7 – Страница регистрации

После ввода данных и нажатия на кнопку «Зарегистрироваться» пользователя переадресует в личный кабинет (рисунок 2.8).

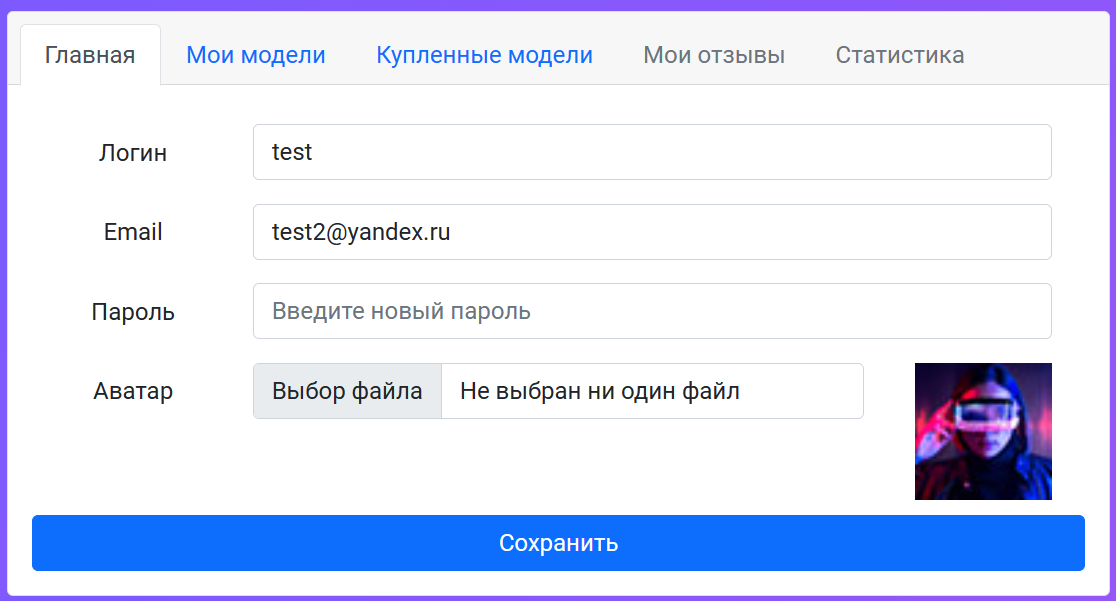


Рисунок 2.8 – Личный кабинет пользователя

В личном кабинете пользователь может поменять персональные данные (логин, электронную почту, пароль и аватар), а также посмотреть свои, а также купленные модели.

Вид страницы просмотра своих моделей представлен на рисунке 2.9.

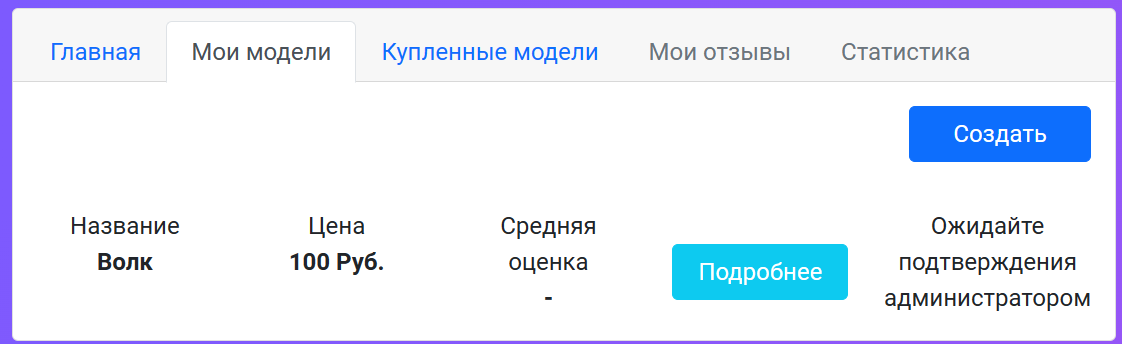


Рисунок 2.9 – Страница «Мои модели»

При нажатии на кнопку «Подробнее» пользователь будет переадресован на страницу редактирования объявления (рисунок 2.10), при нажатии на кнопку «Создать» пользователь будет переадресован на страницу создания объявления (рисунок 2.11).

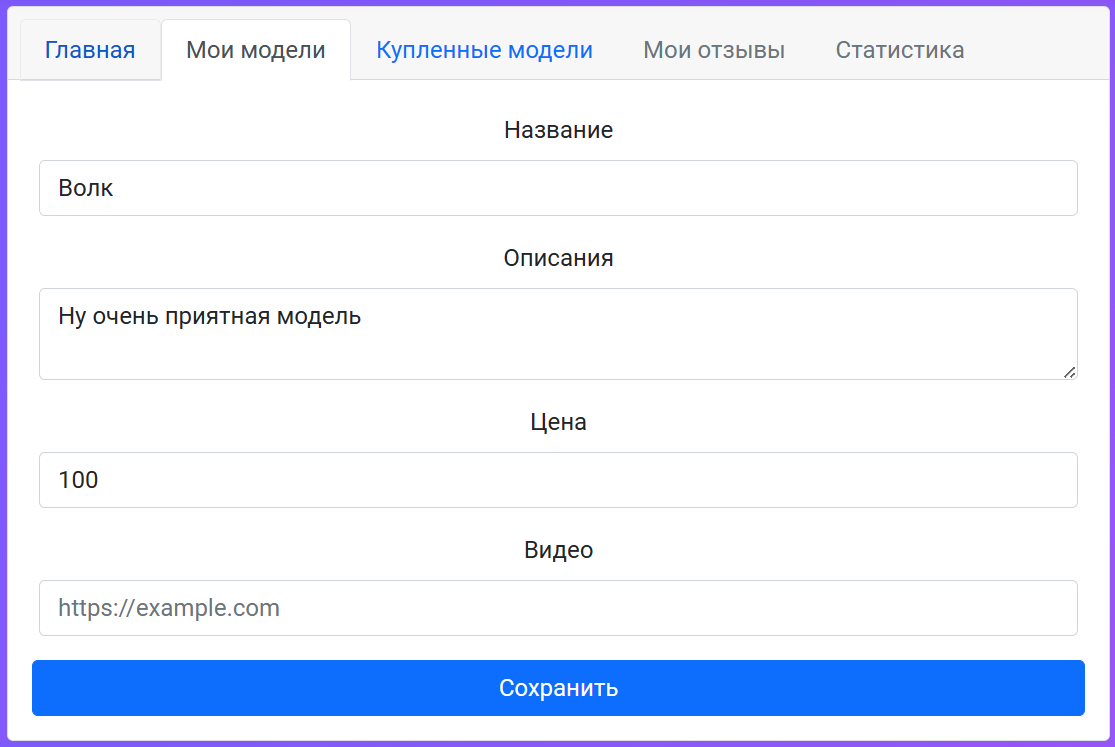


Рисунок 2.10 – Страница редактирования объявления

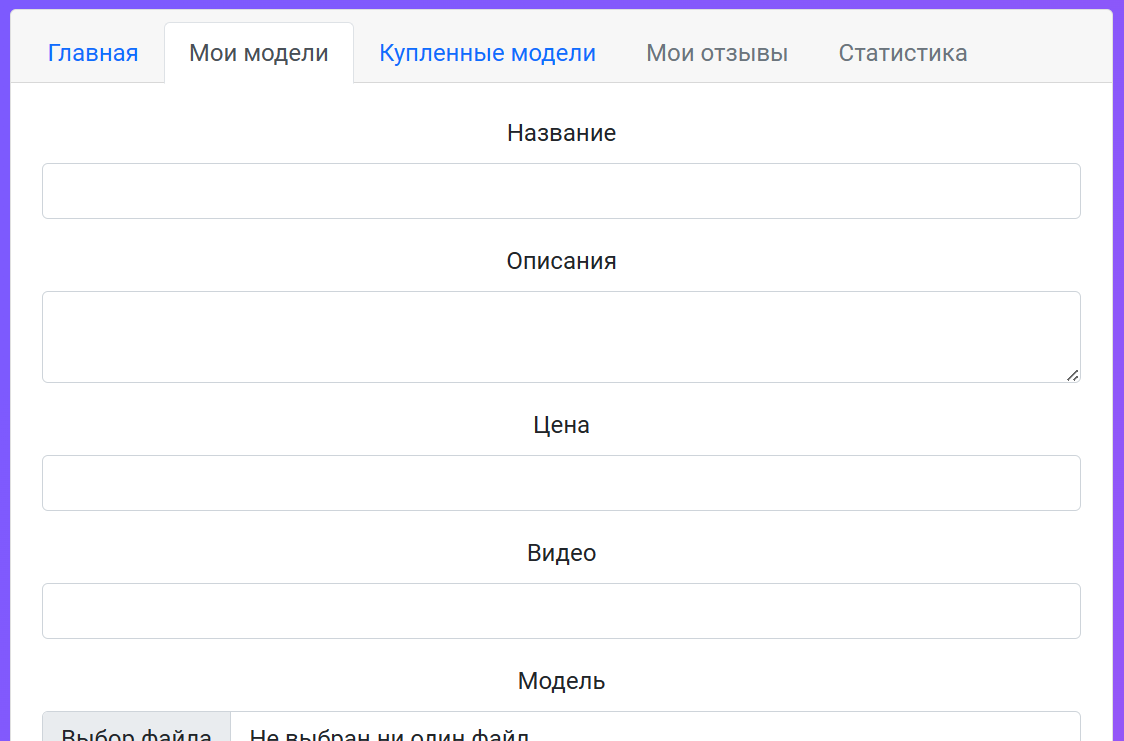


Рисунок 2.11 – Страница создания объявления

На странице «Купленные модели» пользователь может посмотреть купленные ранее модели, скачать, а также оставить отзыв (рисунок 2.12).

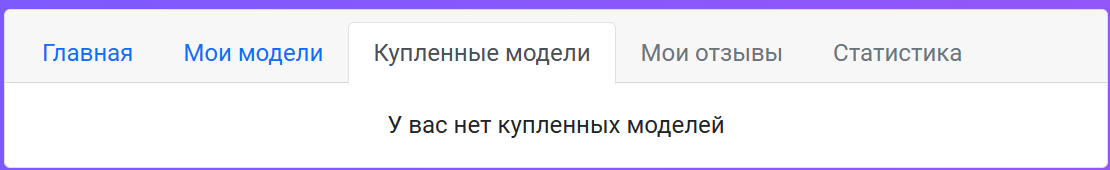


Рисунок 2.12 – Страница «Купленные модели»

На странице «Мои отзывы» пользователь может посмотреть и редактировать ранее оставленные отзывы (рисунок 2.13).

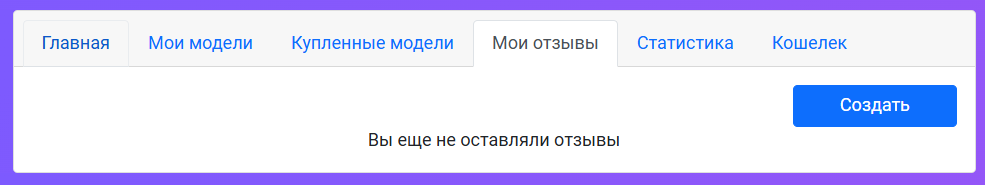


Рисунок 2.13 – Страница «Мои отзывы»

На странице «Статистика» пользователь может свой доход за текущий расчетный год (рисунок 2.14).

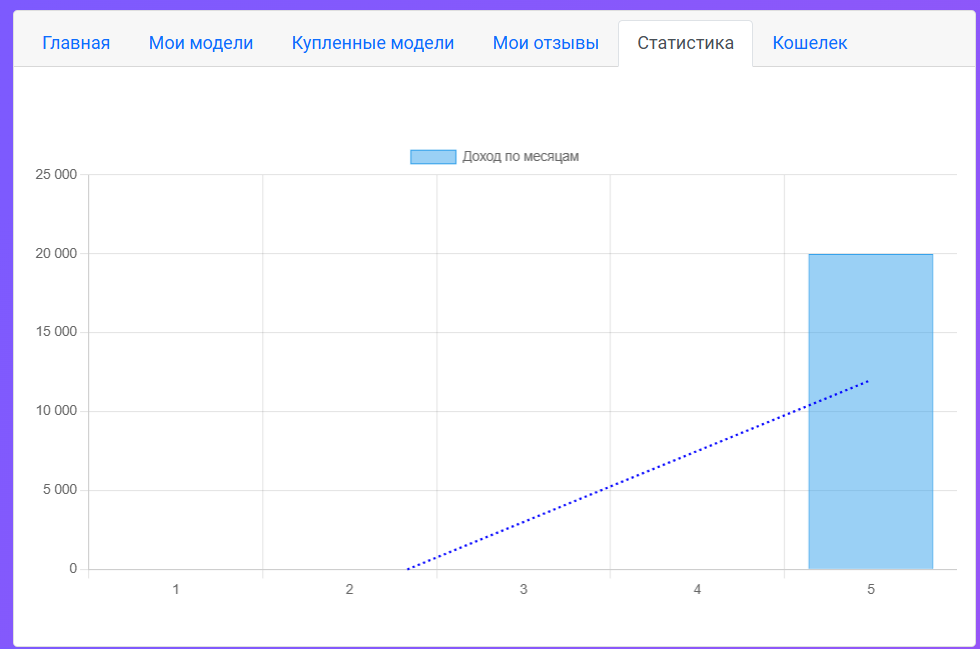


Рисунок 2.14 – Страница «Статистика»

На странице «Кошелек» пользователь может посмотреть историю пополнений и списаний, а также пополнить свой электронный кошелек (рисунок 2.15).

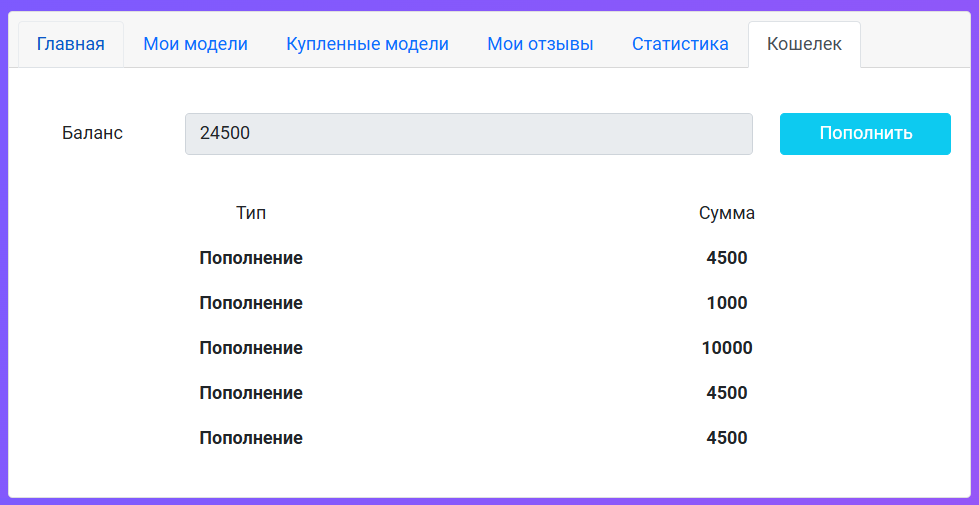


Рисунок 2.15 – Страница «Кошелек»

Рассмотрим Панель администратора. На данной странице представлены следующие окна:

* Модели;
* Отзывы;
* Заказы;
* Пользователи.

На странице «Модели» администратор может посмотреть, одобрить или отклонить созданные всеми пользователями модели, а также выбрать модели с необходимым статусом путем выполнения фильтрации (рисунок 2.16).

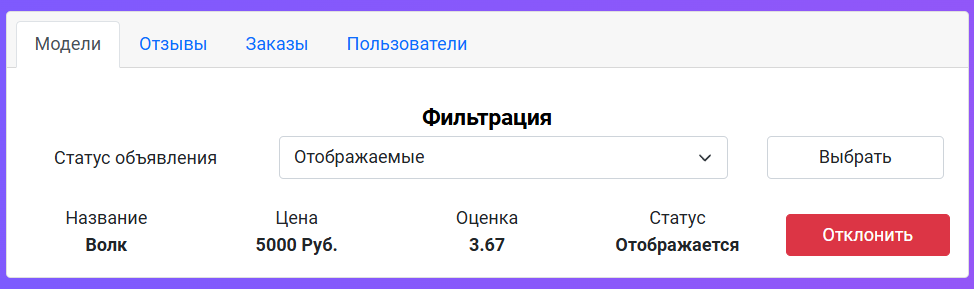


Рисунок 2.16 – Страница «Модели»

На странице «Отзывы» администратор может посмотреть, одобрить или отклонить созданные всеми пользователями отзывы (рисунок 2.17).

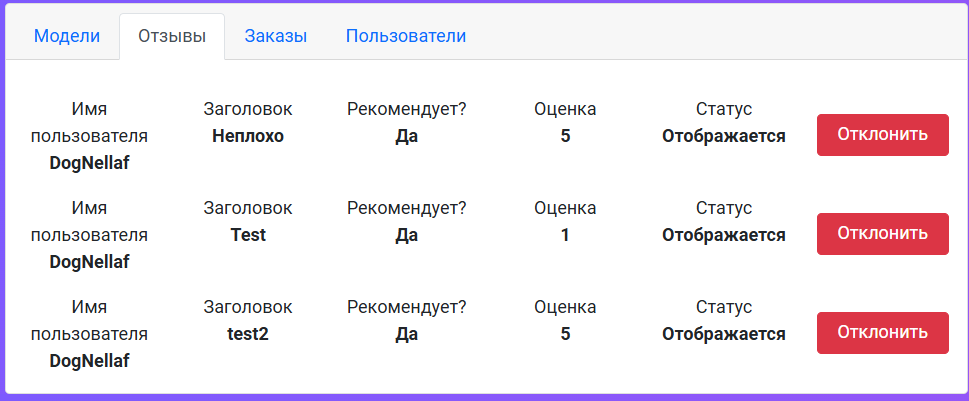


Рисунок 2.17 – Страница «Модели»

На странице «Заказы» администратор может посмотреть ранее созданные заказы (рисунок 2.18).

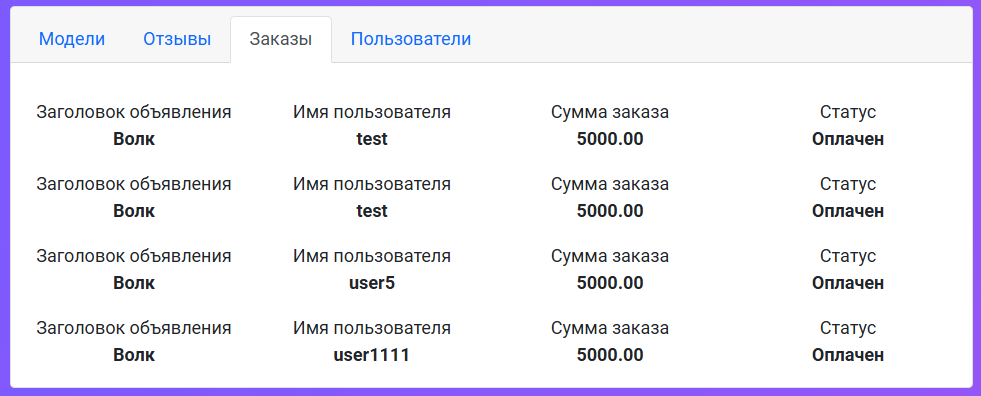


Рисунок 2.18 – Страница «Заказы»

На странице «Пользователи» администратор может посмотреть, заблокировать и разблокировать зарегистрированных пользователей (рисунок 2.19).

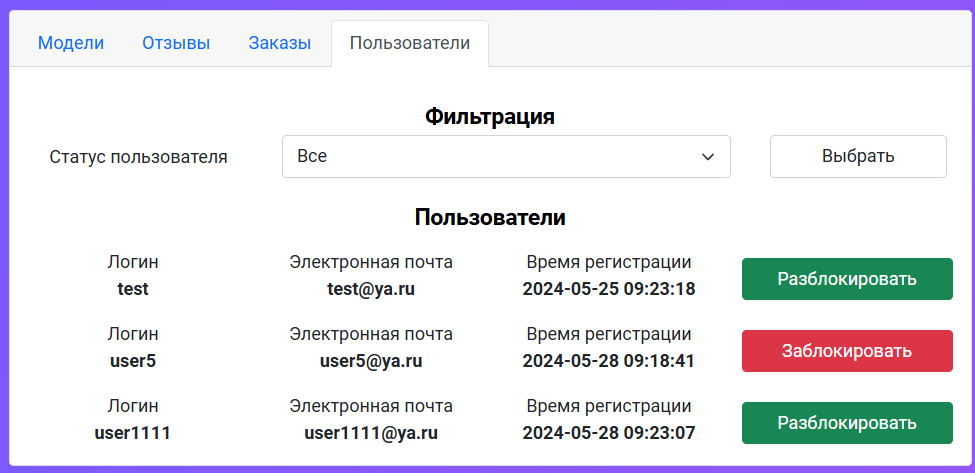


Рисунок 2.19 – Страница «Пользователи»

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта был разработан сервис C2C торговли трехмерными моделями, который представляет собой инновационное решение для пользователей, позволяющее обмениваться и приобретать трехмерные модели. Использование данного сервиса позволит компании сократить издержки и увеличить свою прибыль за счёт оказания посреднических услуг.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Борисов, Р. С. Информатика. Создание интернет-сайтов — Москва: РГУП, 2022. — 157 с. — URL: <https://e.lanbook.com/book/282227> (дата обращения: 10.05.2024);
2. Гагарин, А. Г. Практикум по разработке Web-приложений с использованием РНР и MySQL — Волгоград: Волгоградский ГАУ, 2017. — 120 с. — URL: <https://e.lanbook.com/book/107832> (дата обращения: 10.05.2024);
3. Джош, Л. Современный PHP. Новые возможности и передовой опыт — Москва: ДМК Пресс, 2016. — 304 с. — URL: https://e.lanbook.com/book/93269 (дата обращения: 10.05.2024);
4. Кириченко, А. В. Laravel для web-разработчиков. Практическое руководство по созданию профессиональных сайтов — Санкт-Петербург: Наука и Техника, 2021. — 432 с. — URL: https://e.lanbook.com/book/191484 (дата обращения: 10.05.2024);
5. Кириченко, А. В. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков — Санкт-Петербург: Наука и Техника, 2021. — 432 с. — URL: <https://e.lanbook.com/book/191460> (дата обращения: 10.05.2024);
6. Кожевникова, П. В. PHP и MySQL — Ухта: УГТУ, 2020. — 51 с. —URL: <https://e.lanbook.com/book/209591> (дата обращения: 10.05.2024);
7. Рочев, К. В. Информационные технологии. Анализ и проектирование информационных систем — Санкт-Петербург: Лань, 2022. — 128 с. — URL: <https://e.lanbook.com/book/206894> (дата обращения: 10.05.2024);
8. Смоленцева, Т. Е. Проектирование и разработка WEB-приложений: Практикум — Москва: РТУ МИРЭА, 2023. — 68 с. — https://e.lanbook.com/book/368954 (дата обращения: 10.05.2024);
9. Федькова, Н. А. Современные технологии разработки программного обеспечения — Брянск: Брянский ГАУ, 2022. — 58 с. — URL: <https://e.lanbook.com/book/305087> (дата обращения: 10.05.2024);
10. Чаллавала, Ш. MySQL 8 для больших данных — Москва: ДМК Пресс, 2018. — 226 с. — URL: <https://e.lanbook.com/book/131684> (дата обращения: 10.05.2024);
11. Зудилова, Т. В. Web-программирование HTML — Санкт-Петербург: НИУ ИТМО, 2012. — 70 с. — URL: <https://e.lanbook.com/book/40724> (дата обращения: 10.05.2024);
12. Селина, Е. Г. Организация интерактивного взаимодействия в HTML-документах — Санкт-Петербург: НИУ ИТМО, 2018. — 35 с. — URL: <https://e.lanbook.com/book/136467> (дата обращения: 10.05.2024);
13. Кириченко, А. В. HTMLS + CSS3. Основы современного WEB-дизайна — Санкт-Петербург: Наука и Техника, 2018. — 352 с. — URL: <https://e.lanbook.com/book/108277> (дата обращения: 10.05.2024);
14. Сакулин, С. А. Основы интернет-технологий: HTML, CSS, JavaScript, XML — Москва: МГТУ им. Н.Э. Баумана, 2017. — 112 с. — URL: <https://e.lanbook.com/book/103525> (дата обращения: 10.05.2024);
15. Кириченко, А. В. Динамические сайты на HTML, CSS, JAVASCRIPT И BOOTSTRAP. Практика, практика и только практика — Санкт-Петербург: Наука и Техника, 2018. — 272 с. — URL: <https://e.lanbook.com/book/108282> (дата обращения: 10.05.2024);
16. Янцев, В. В. JavaScript. Креативное программирование — 2-е изд., стер. — Санкт-Петербург: Лань, 2024. — 232 с. — URL: <https://e.lanbook.com/book/383837> (дата обращения: 10.05.2024);
17. Диков, А. В. Web-программирование на JavaScript — Санкт-Петербург: Лань, 2024. — 168 с. — URL: <https://e.lanbook.com/book/387305> (дата обращения: 10.05.2024);
18. Зенченко, И. В. Проектирование бизнес-процессов. Практические аспекты — Москва: ФЛИНТА, 2017. — 118 с. — URL: <https://e.lanbook.com/book/97142> (дата обращения: 10.05.2024);
19. Вейцман, В. М. Проектирование информационных систем — Санкт-Петербург: Лань, 2022. — 316 с. — URL: <https://e.lanbook.com/book/206873> (дата обращения: 10.05.2024);
20. Оверби, Х. Цифровая экономика: как информационно-коммуникационные технологии влияют на рынки, бизнес и инновации — Москва: Дело РАНХиГС, 2022. — 288 с. — URL: https://e.lanbook.com/book/293072 (дата обращения: 10.05.2024).

Приложение А  
Диаграмма вариантов использования

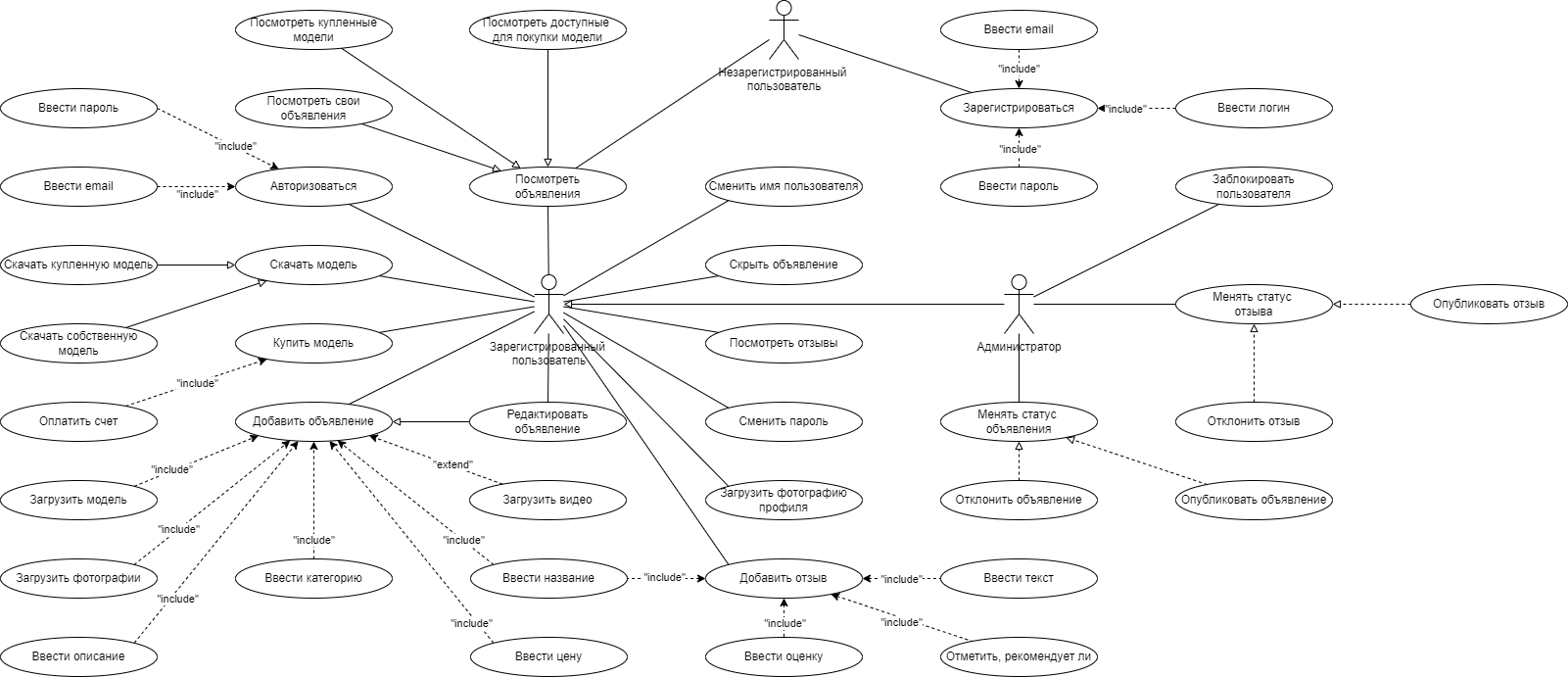


Рисунок А.1 – Диаграмма вариантов использования

Приложение Б  
Логическая структура программы

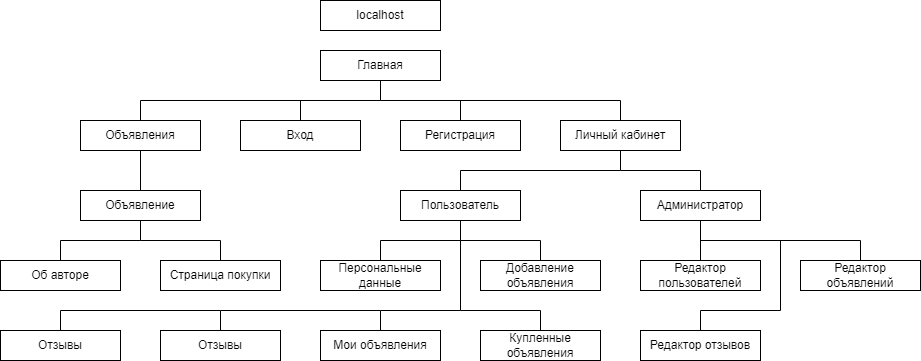


Рисунок Б.1 – Логическая структура программы

Приложение В  
Листинг программного кода

Таблица В.1 – Листинг программного кода

|  |  |
| --- | --- |
| Название модуля | Программный код |
| MainController | <?php  namespace App\Http\Controllers;  use App\Models\Ad;  use App\Models\User;  use App\Models\Order;  use App\Models\WalletHistory;  use Illuminate\Support\Facades\DB;  use Illuminate\Support\Facades\Auth;  class MainController extends Controller  {  // получить объявления, отсортированные по убыванию средней оценки  \_\_get\_most\_popular\_ads() {  $ads = DB::table('ads')  ->leftJoin('reviews', 'reviews.ad\_id', '=', 'ads.id')  ->join('users', 'users.id', '=', 'ads.user\_id')  ->select(DB::raw("AVG('reviews.rating') as average, ads.id as id"))  ->groupBy('ads.id')  ->orderBy('average', 'DESC')  ->get();  $ids = $ads->pluck('id');  return Ad::whereIn('id', $ids)->where("status", "=", "Showed");  }  // Отобразить главную страницу  public function index() {  return view('main/index', ['ads' =>  self::\_\_get\_most\_popular\_ads()->paginate(10)]);  }  // Отобразить страницу Модели  public function ads() {  $ads = self::\_\_get\_most\_popular\_ads();  return view('main/ads/list', [  'ads' => $ads->paginate(10),  'main\_ads' => $ads->limit(4)->get()  ]);  }  // Страница покупки модели  public function buy(Ad $ad) {  if ($ad->status != "Showed") {  return abort(403);  }  return view('main/ads/buy', ['ad' => $ad]);  }  // Страница правил сервиса  public function rules() {  return view('main/rules');  }  // Страница подробнее о модели  public function ad\_detail(Ad $ad) {  if ($ad->status != "Showed") {  return abort(403);  }  return view('main/ads/detail', ['ad' => $ad]);  }  // Страница подробнее об авторе  public function user\_detail(User $user) {  if ($ad->is\_banned) {  return abort(403);  }  return view('main/users/detail', ['user' => $user]);  }  // Страница переадресации при покупки модели  public function confirm(Ad $ad) {  if (!Auth::check()) {  return abort('403');  }  $user = Auth::user();  if ($ad->user\_id == $user->id) {  return abort('403');  }  if ($user->is\_banned) {  return abort(403);  }  if ($ad->status != "Showed") {  return abort(403);  }  if ($user->get\_wallet\_balance() < $ad->price) {  return abort('500');  }  $order = Order::create([  'ad\_id' => $ad->id,  'user\_id' => $user->id,  'price' => $ad->price,  'status' => 'Created'  ]);  $order->save();  $outcome\_transaction = WalletHistory::create([  'user\_id' => $user->id,  'value' => $ad->price,  'type' => "Outcome"  ]);  $outcome\_transaction->save();  $income\_transaction = WalletHistory::create([  'user\_id' => $ad->user\_id,  'value' => $ad->price \* 0.9,  'type' => "Income"  ]);  $income\_transaction->save();  $order->status = "Paid";  $order->save();  return redirect()->route('home.ads.bought');  }  } |
| HomeController | <?php  namespace App\Http\Controllers;  use Illuminate\Http\Request;  use Illuminate\Support\Facades\Auth;  use Illuminate\Validation\Rule;  use App\Models\Ad;  use App\Models\Review;  use App\Models\WalletHistory;  class HomeController extends Controller  {  private const AD\_EDIT\_VALIDATOR = [  'title' => ['required', 'max:255', 'min:1'],  'description' => ['required', 'min:1'],  'price' => ['required', 'numeric'],  'video\_link' => ['nullable', 'max:255']  ];  private const AD\_CREATE\_VALIDATOR = [  'model' => 'required|max:4096',  'photo' => 'required|max:1024',  ];  private const REVIEW\_CREATE\_VALIDATOR = [  'ad\_id' => 'required|integer',  'title' => 'required|max:255',  'text' => 'required|string',  'is\_recommended' => 'required|string',  'rate' => 'required|integer|min:1|max:5'  ];  private const REVIEW\_EDIT\_VALIDATOR = [  'title' => 'required|max:255',  'text' => 'required|string',  'is\_recommended' => 'required|string',  'rate' => 'required|integer|min:1|max:5'  ];  private const WALLET\_HISTORY\_VALIDATOR = [  'value' => 'required|numeric|min:1'  ];  public function \_\_construct()  {  $this->middleware('auth');  }  // Функция проверки авторизации пользователя  protected function check\_auth\_user() {  $user = Auth::user();  if (!$user) {  abort(403);  }  return $user;  }  // Отображает главную страницу пользователя  public function index()  {  $user = self::check\_auth\_user();  return view('home/index', [  'user' => $user  ]);  }  // Сохраняет данные пользователя и отображает главную страницу  public function save\_data(Request $request) {  $user = self::check\_auth\_user();  $validated = $request->validate([  'name' => [  'required',  'string',  'max:255',  'min:8',  Rule::unique('users')->ignore($user)],  'email' => [  'required',  'email',  'max:255',  'min:5',  Rule::unique('users')->ignore($user)],  'password' => [  'nullable',  'string'],  'avatar' => ['nullable', 'max:1024']  ]);  $user->name = $validated['name'];  $user->email = $validated['email'];  $password = $validated['password'];  if ($password != null) {  $user->password = $password;  }  $user->avatar\_url = "images/author.jpg";  if ($request->hasFile('avatar')) {  $avatar = $request->file('avatar');  $date = date('YmdHis');  $avatar\_filename = $date.'.png';  $public\_path = "storage\\";  $avatar\_path = $public\_path.'\\'.$avatar\_filename;  $user->avatar\_url = $avatar\_path;  $avatar->move($public\_path, $avatar\_filename);  }  $user->save();  return redirect()->route('home.index');  }  // Показывает страницу с моделями пользователя  public function own\_ads()  {  $user = self::check\_auth\_user();  return view('home/ads/index', [  'user' => $user,  'ads' => $user->ads()->paginate(5)  ]);  }  // Отображает страницу кошелька  public function wallet()  {  $user = self::check\_auth\_user();  $transactions = $user->transactions()->paginate(5);  return view('home/wallet/index', [  'user' => $user,  'transactions' => $transactions  ]);  }  // Отображает страницу пополнения кошелька  public function add\_balance\_page()  {  $user = self::check\_auth\_user();  return view('home/wallet/add', [  'user' => $user  ]);  }  // Функция добавляет к балансу соответствующее значение и отображает страницу кошелька  public function add\_balance\_method(Request $request)  {  $user = self::check\_auth\_user();  $validated = $request->validate(self::WALLET\_HISTORY\_VALIDATOR);  $transaction = WalletHistory::create([  'user\_id' => $user->id,  'value' => $validated['value'],  'type' => "Income"  ]);  $transaction->save();  return redirect()->route('home.wallet');  }  // Отображает страницу создания модели  public function ads\_create\_page() {  return view('home/ads/create', [  'user' => self::check\_auth\_user()  ]);  }  // Проверяет корректность введенных данных модели, после чего отображает редактор модели  public function ads\_create\_method(Request $request) {  $user = self::check\_auth\_user();  $validated = $request->validate(self::AD\_EDIT\_VALIDATOR);  $request->validate(self::AD\_CREATE\_VALIDATOR);  $model = $request->file('model');  $photo = $request->file('photo');  $model\_name = $model->getClientOriginalName();  $photo\_name = $photo->getClientOriginalName();  $dir = 'storage\\\\'.explode(".", $model\_name)[0].'\\\\';  $model\_path = $dir.$model\_name;  $photo\_path = $dir.$photo\_name;  $model->move($dir, $model\_path);  $photo->move($dir, $photo\_path);  Ad::create([  'title' => $validated['title'],  'description' => $validated['description'],  'price' => $validated['price'],  'video\_link' => $validated['video\_link'],  'model\_link' => $model\_path,  'user\_id' => $user->id,  'photo\_link' => $photo\_path  ]);  return redirect()->route('home.ads.own');  }  // Отображает редактор модели  public function ads\_edit\_page(Ad $ad) {  return view('home/ads/edit', [  'ad' => $ad,  'user' => self::check\_auth\_user()  ]);  }  // Сохраняет модель после редактирования  public function ads\_edit\_method(Request $request, Ad $ad) {  $user = self::check\_auth\_user();  if ($ad->user\_id != $user->id) {  return abort('403');  }  $validated = $request->validate(self::AD\_EDIT\_VALIDATOR);  $ad->fill([  'title' => $validated['title'],  'description' => $validated['description'],  'price' => $validated['price'],  'video\_link' => $validated['video\_link']  ]);  $ad->status = "Created";  $ad->save();  return redirect()->route('home.ads.own');  }  // Меняет статус модели на «Скрыта»  public function ads\_hide\_method(Request $request, Ad $ad) {  $user = Auth::user();  if ($ad->user\_id != $user->id) {  return abort('403');  }  if ($ad->status = "Showed") {  $ad->status = 'Hidden';  $ad->save();  return redirect()->route('home.ads.own');  } else {  return abort('403');  }  }  // Меняет статус модели на «Отображается»  public function ads\_show\_method(Request $request, Ad $ad) {  $user = Auth::user();  if ($ad->user\_id != $user->id) {  return abort('403');  }  if ($ad->status = "Hidden") {  $ad->status = 'Showed';  $ad->save();  return redirect()->route('home.ads.own');  } else {  return abort('403');  }  }  // Отображает страницу купленных моделей  public function bought\_ads()  {  $user = self::check\_auth\_user();  return view('home/ads/bought', [  'user' => $user,  'orders' => $user->orders()->paginate(5)  ]);  }  // Отображает отзывы пользователя  public function reviews()  {  $user = self::check\_auth\_user();  return view('home/reviews/index', [  'user' => $user,  'reviews' => $user->reviews()->paginate(5)  ]);  }  // Отображает страницу создания отзыва  public function review\_create\_page() {  $user = self::check\_auth\_user();  return view('home/reviews/create', [  'user' => $user,  'orders' => $user->orders\_without\_review()->paginate(10)  ]);  }  // Метод сохраняет созданный отзыв  public function review\_create\_method(Request $request) {  $user = self::check\_auth\_user();  $validated = $request->validate(self::REVIEW\_CREATE\_VALIDATOR);  $ad\_id = Ad::where('id', '=', $validated['ad\_id'])->first()->id;  $is\_recommended = $validated['is\_recommended'] == 'True'? True : False;  Review::create([  'title' => $validated['title'],  'text' => $validated['text'],  'is\_recommended' => $is\_recommended,  'rate' => $validated['rate'],  'ad\_id' => $ad\_id,  'user\_id' => $user->id  ]);  return redirect()->route('home.reviews.list');  }  // Отображает страницу редактора отзыва  public function review\_edit\_page(Review $review) {  return view('home/reviews/edit', [  'user' => self::check\_auth\_user(),  'review' => $review  ]);  }  // Сохраняет отзыв после редактирования  public function review\_edit\_method(Request $request, Review $review) {  $user = self::check\_auth\_user();  if ($review->user\_id != $user->id) {  return abort('403');  }  $validated = $request->validate(self::REVIEW\_EDIT\_VALIDATOR);  $is\_recommended = $validated['is\_recommended'] == 'True'? True : False;  $review->fill([  'title' => $validated['title'],  'text' => $validated['text'],  'is\_recommended' => $is\_recommended,  'rate' => $validated['rate']  ]);  $review->save();  return redirect()->route('home.reviews.list');  }  // Отображает страницу статистики  public function stats(Request $request)  {  $user = self::check\_auth\_user();  $current\_year = date("Y");  $current\_month = date("m");  $year = $request['year'];  if ($year == null) {  $year = $current\_year;  }  $months = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12);  $values = [];  $max\_value = 0;  if ($year > $current\_year) {  return abort(500);  }  if ($year == $current\_year) {  for ($i = 12; $i > $current\_month; $i--) {  unset($months[$i - 1]);  $months = array\_values($months);  }  }  foreach ($months as $month) {  $income = $user->get\_income($year, $month);  if ($income > $max\_value) {  $max\_value = $income;  }  array\_push($values, $income);  }  return view('home/stats', [  'user' => $user,  'values' => $values,  'months' => $months,  'year' => $year,  'max\_value' => $max\_value \* 1.2  ]);  }  } |
| AdminController | <?php  namespace App\Http\Controllers;  use Illuminate\Support\Facades\Auth;  use Illuminate\Http\Request;  use App\Models\Review;  use App\Models\Ad;  use App\Models\User;  use App\Models\Order;  class AdminController extends Controller  {  public function \_\_construct() {  $this->middleware('auth');  }  // Проверяет, является ли пользователь администратором  protected function check\_auth\_user() {  $user = Auth::user();  if (!$user) {  abort(403);  }  if ($user->is\_admin == 0) {  abort(403);  }  return $user;  }  // Отображает редактор моделей  public function ads(Request $request) {  $user = self::check\_auth\_user();  $ads = Ad::where("user\_id", "!=", "-1");  $status = $request['status'];  if ($status != null) {  if ($status != "All") {  $ads = $ads->where("status", "=", $status);  }  } else {  $status = "All";  }  return view('admin/ads/list', ['user' => $user,  'ads' => $ads->paginate(10),  'status' => $status]);  }  // Отображает редактор пользователей  public function users(Request $request) {  $user = self::check\_auth\_user();  $users = User::where("is\_admin", "=", "False");  $is\_banned = $request['is\_banned'];  if ($is\_banned != null) {  if ($is\_banned == "true") {  $users = $users->where("is\_banned", "=", "1");  } else if ($is\_banned == "false") {  $users = $users->where("is\_banned", "=", "0");  }  }  return view('admin/users/list', ['user' => $user,  'users' => $users->paginate(10),  'is\_banned' => $is\_banned]);  }  // Отображает редактор отзывов  public function reviews() {  $user = self::check\_auth\_user();  return view('admin/reviews/list', ['user' => $user,  'reviews' => Review::paginate(10)]);  }  // Отображает страницу с информацией по отзыву  public function review(Review $review) {  $user = self::check\_auth\_user();  return view('admin/reviews/detail', ['user' => $user,  'review' => $review]);  }  // Меняет статус отзыва на «Отклонен»  public function review\_reject(Review $review) {  $user = Auth::user();  if (!$user->is\_admin) {  return abort('403');  }  $review->status = "Rejected";  $review->save();  return redirect()->route('admin.reviews.list');  }  // Меняет статус отзыва на «Отображается»  public function review\_approve(Review $review) {  $user = Auth::user();  if (!$user->is\_admin) {  return abort('403');  }  $review->status = "Showed";  $review->save();  return redirect()->route('admin.reviews.list');  }  // Отображает редактор заказов  public function orders() {  $user = self::check\_auth\_user();  return view('admin/orders/list', ['user' => $user,  'orders' => Order::paginate(10)]);  }  // Меняет статус модели на «Отклонена»  public function ad\_reject(Ad $ad) {  $user = Auth::user();  if (!$user->is\_admin) {  return abort('403');  }  $ad->status = "Rejected";  $ad->save();  return redirect()->route('admin.ads.list');  }  // Меняет статус модели на «Одобрена»  public function ad\_approve(Ad $ad) {  $user = Auth::user();  if (!$user->is\_admin) {  return abort('403');  }  $ad->status = "Showed";  $ad->save();  return redirect()->route('admin.ads.list');  }  // Блокирует пользователя  public function user\_ban(User $user) {  $admin = Auth::user();  if (!$admin->is\_admin) {  return abort('403');  }  $user->is\_banned = True;  $user->save();  return redirect()->route('admin.users.list');  }  // Разблокирует пользователя  public function user\_unban(User $user) {  $admin = Auth::user();  if (!$admin->is\_admin) {  return abort('403');  }  $user->is\_banned = False;  $user->save();  return redirect()->route('admin.users.list');  }  } |