Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

ОТЧЁТ

по производственной (преддипломной) практике

по специальности 09.02.07 Информационные системы и программирование

|  |  |
| --- | --- |
| Методист по учебно-  производственной работе  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/А.И. Файзулова/ | Руководитель практики  от предприятия  \_\_\_\_\_\_\_\_\_\_\_\_/А.И. Абрашитов/  подпись ФИО  М.П. |
| Руководитель практики  от учебного заведения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Е.К. Дмитриева/  подпись ФИО | Студент группы 20Веб-3  \_\_\_\_\_\_\_\_\_\_/И.Е. Петунин/  подпись ФИО  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 год |

УФА 2024 год

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc166235645)

[1 Проектирование информационной системы 6](#_Toc166235646)

[1.1 Описание предметной области 6](#_Toc166235647)

[1.2 Описание входной информации 7](#_Toc166235648)

[1.3 Описание выходной информации 8](#_Toc166235649)

[1.4 Концептуальное проектирование 8](#_Toc166235650)

[1.4.1 Диаграмма вариантов использования 8](#_Toc166235651)

[1.4.2 Диаграмма бизнес-процесса 8](#_Toc166235652)

[1.5 Логическое проектирование 9](#_Toc166235653)

[1.5.1 Диаграмма классов 9](#_Toc166235654)

[1.5.2 Диаграмма связей сущностей 9](#_Toc166235655)

[1.6 Описание структуры базы данных 10](#_Toc166235656)

[1.7 Общие требования к программному продукту 13](#_Toc166235657)

[2 Экспериментальный раздел 14](#_Toc166235658)

[2.1 Описание программы 14](#_Toc166235659)

[2.2 Тестирование программного обеспечения 20](#_Toc166235660)

[2.3 Руководство пользователя 28](#_Toc166235661)

[ЗАКЛЮЧЕНИЕ 33](#_Toc166235662)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc166235663)

[ПРИЛОЖЕНИЕ А 36](#_Toc166235664)

[ПРИЛОЖЕНИЕ Б 37](#_Toc166235665)

[ПРИЛОЖЕНИЕ В 38](#_Toc166235666)

[ПРИЛОЖЕНИЕ Г 39](#_Toc166235667)

# 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.

## 1.2 Описание входной информации

Описание входных файлов представлено в таблице 1.

Таблица 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 2, являются:

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

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

## 1.4 Концептуальное проектирование

### 1.4.1 Диаграмма вариантов использования

Диаграмма вариантов использования представлена в Приложении А.

### 1.4.2 Диаграмма бизнес-процесса

На рисунке 1.1 представлена диаграмма бизнес-процесса «Оказание информационно-посреднических услуг». Процесс начинается со входа пользователя на сайт и ввода данных для авторизации или регистрации. Если пользователь существует и данные корректны, а также если пользователь не существует и был зарегистрирован, происходит авторизация пользователя.

Пользователь выбирает модель на главной странице сайта, переходит на страницу «О модели», оформляет заказ и оплачивает счет.

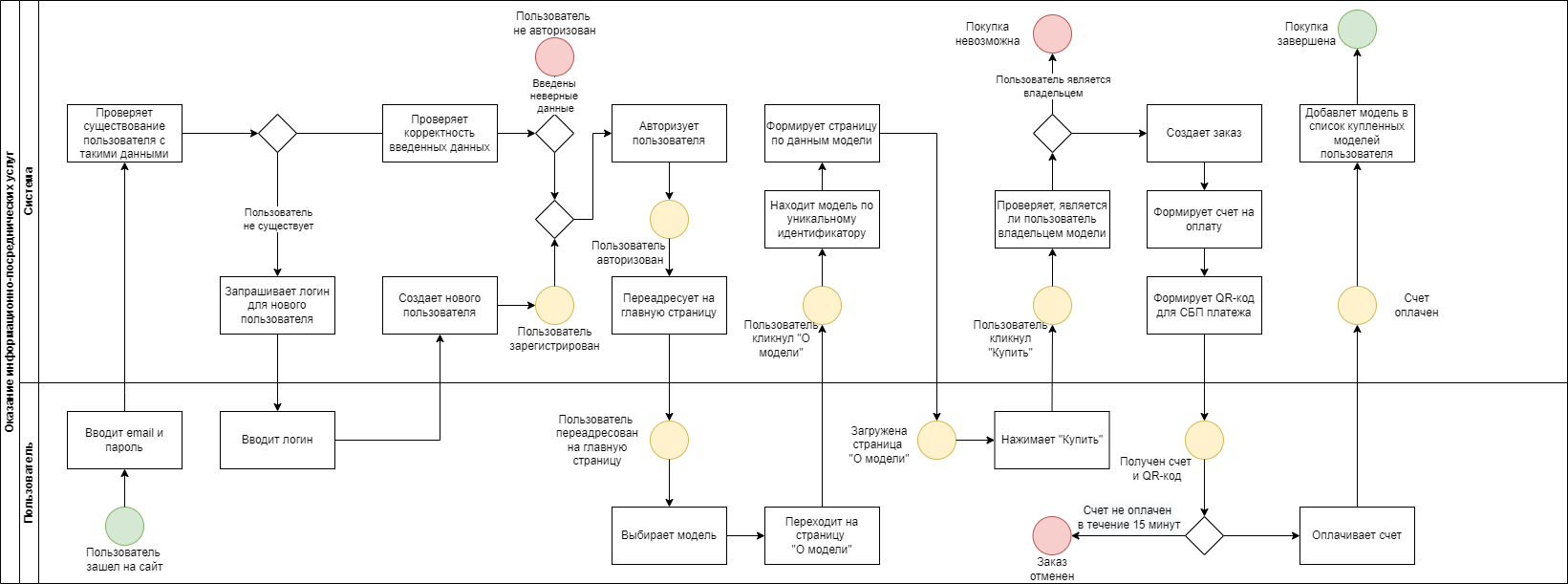


Рисунок 1.1 – Диаграмма бизнес-процесса

## 1.5 Логическое проектирование

### 1.5.1 Диаграмма классов

Диаграмма классов системы представлена в Приложении Б.

### 1.5.2 Диаграмма связей сущностей

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

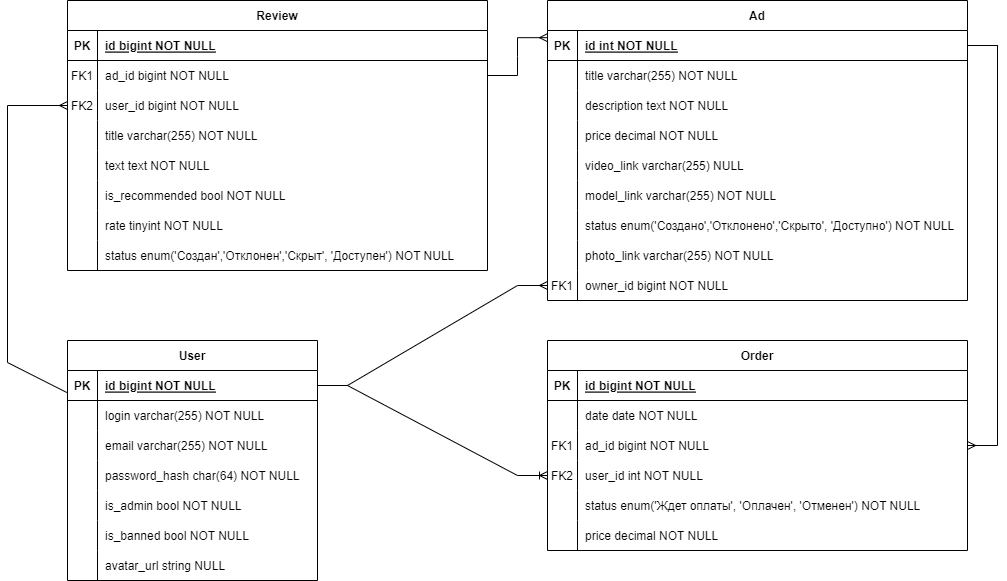


Рисунок 1.2 – Диаграмма связей сущностей

## 1.6 Описание структуры базы данных

В таблице 3 представлено описание структуры базы данных – все существующие таблицы и поля.

Таблица 3 - Описание структуры базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| users (User) | | | | |
| id | Уникальный идентификатор пользователя | INTEGER |  | PK |
| login | Уникальное имя пользователя | VARCHAR | 255 |  |
| email | Уникальный email пользователя | VARCHAR | 255 |  |
| password\_hash | Хэш пароля | CHAR | 64 |  |
| is\_admin | Метка принадлежности пользователя роли Администратор | BOOLEAN |  |  |
| is\_banned | Метка блокировки пользователя | BOOLEAN |  |  |
| avatar\_url | Относительный путь до аватарки пользователя в статичных файлах сервера | STRING |  |  |
| ads (Ad) | | | | |
| id | Уникальный идентификатор | INTEGER |  | PK |
| title | Заголовок объявления | VARCHAR | 255 |  |

Продолжение Таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| price | Цена модели | DECIMAL |  |  |
| video\_link | URL ссылка на видео с демонстрацией модели | VARCHAR | 255 |  |
| model\_link | Относительный путь до файла модели в папке статики сервера | VARCHAR | 255 |  |
| status | Статус объявления (Создано, Отклонено, Скрыто, Доступно) | ENUM | 4 |  |
| owner\_id | Идентификатор создателя объявления | INTEGER |  | FK |
| reviews (Review) | | | | |
| id | Уникальный идентификатор отзыва | INTEGER |  | PK |
| ad\_id | Идентификатор модели, на которую был написан отзыв | INTEGER |  | FK |
| user\_id | Идентификатор автора отзыва | INTEGER |  | FK |
| title | Заголовок отзыва | VARCHAR | 255 |  |
| text | Текст отзыва | TEXT |  |  |

Продолжение Таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| is\_recommended | Метка наличия или отсутствия рекомендации | BOOLEAN |  |  |
| rate | Оценка | TINYINT |  |  |
| status | Статус отзыва (Создан, Отклонен, Скрыт, Доступен) | ENUM |  |  |
| orders (Order) | | | | |
| id | Уникальный идентификатор | INTEGER |  | PK |
| date | Дата заказа | DATE |  |  |
| ad\_id | Идентификатор купленной модели | INTEGER |  | FK |
| user\_id | Идентификатор покупателя | INTEGER |  | FK |
| status | Статус заказа (Ждет оплаты, Оплачен, Отменен) | ENUM |  |  |
| price | Расчетное поле – цена модели в момент покупки | DECIMAL |  |  |

## 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. Физическая и логическая структуры представлены в приложении В.

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.

|  |  |  |
| --- | --- | --- |
| 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.

|  |  |  |
| --- | --- | --- |
| 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.

|  |  |  |
| --- | --- | --- |
| 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.

|  |  |  |
| --- | --- | --- |
| 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.

|  |  |  |
| --- | --- | --- |
| 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. |

Исходный код программы представлен в Приложении Г.

## 2.2 Тестирование программного обеспечения

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

В таблице 2.2 представлена общая информация о тестировании.

|  |  |
| --- | --- |
| Наименование | Описание |
| Название проекта | C2C-Market «Экземпляр» |
| Номер версии | 1.0 |
| Имя тестера | Петунин Иван Евгеньевич |
| Даты тестирования | 08.05.2024 |

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. |

Продолжение таблицы 2.3.

|  |  |
| --- | --- |
| 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.

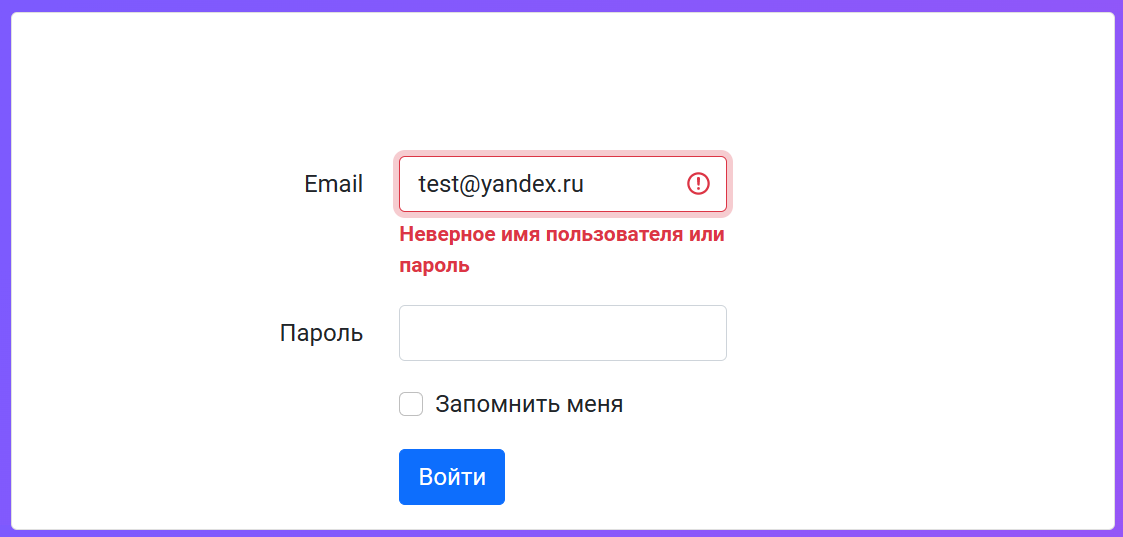


Рисунок 2.1 – Системное сообщение при вводе некорректных данных авторизации

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» |
| 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.

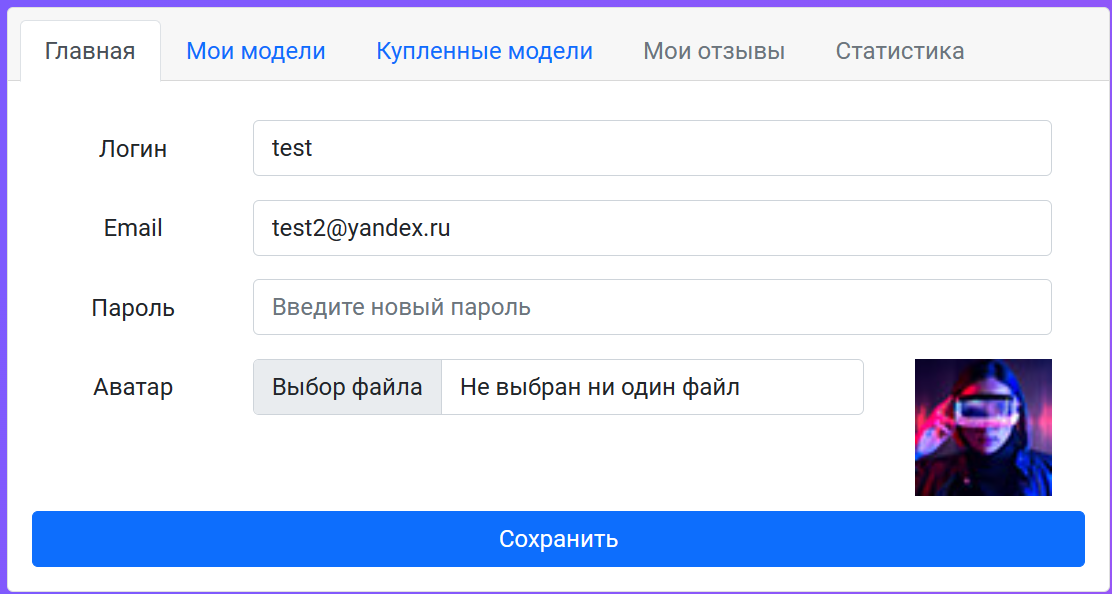


Рисунок 2.2 – Вид личного кабинета для нового пользователя

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. |
| Резюме тестирования | Необходимо добиться корректного поведения программы. |
| Этапы теста | Переход на страницу авторизации;  Заполнение поля корректными данными;  Нажатие на кнопку «Войти» |

Продолжение таблицы 2.5.

|  |  |
| --- | --- |
| 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 | Авторизация в системе;  Переход на страницу создания объявления;  Заполнение поля корректными данными;  Нажатие на кнопку «Создать» |

Продолжение таблицы 2.6.

|  |  |
| --- | --- |
| Тестовые данные | Название: Волк  Описание: Ну очень приятная 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.

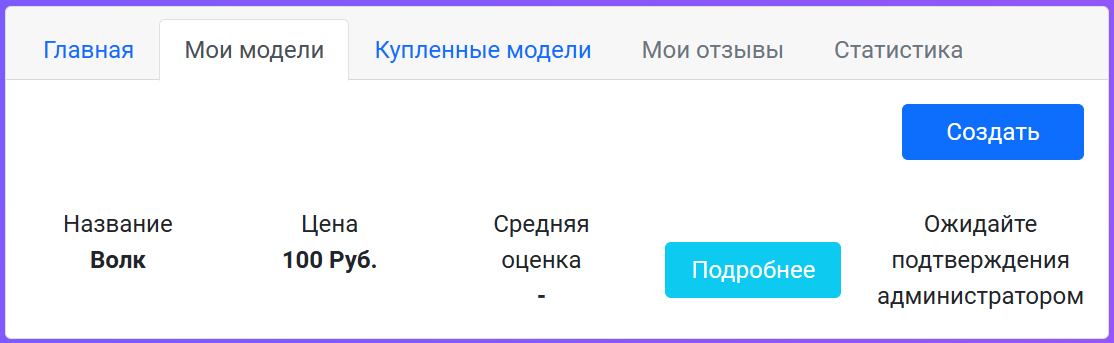


Рисунок 2.3 – Вид страницы «Мои модели» после создания нового объявления

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 | Авторизация в системе;  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 |

Продолжение таблицы 2.7.

|  |  |
| --- | --- |
| 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» |

## 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).

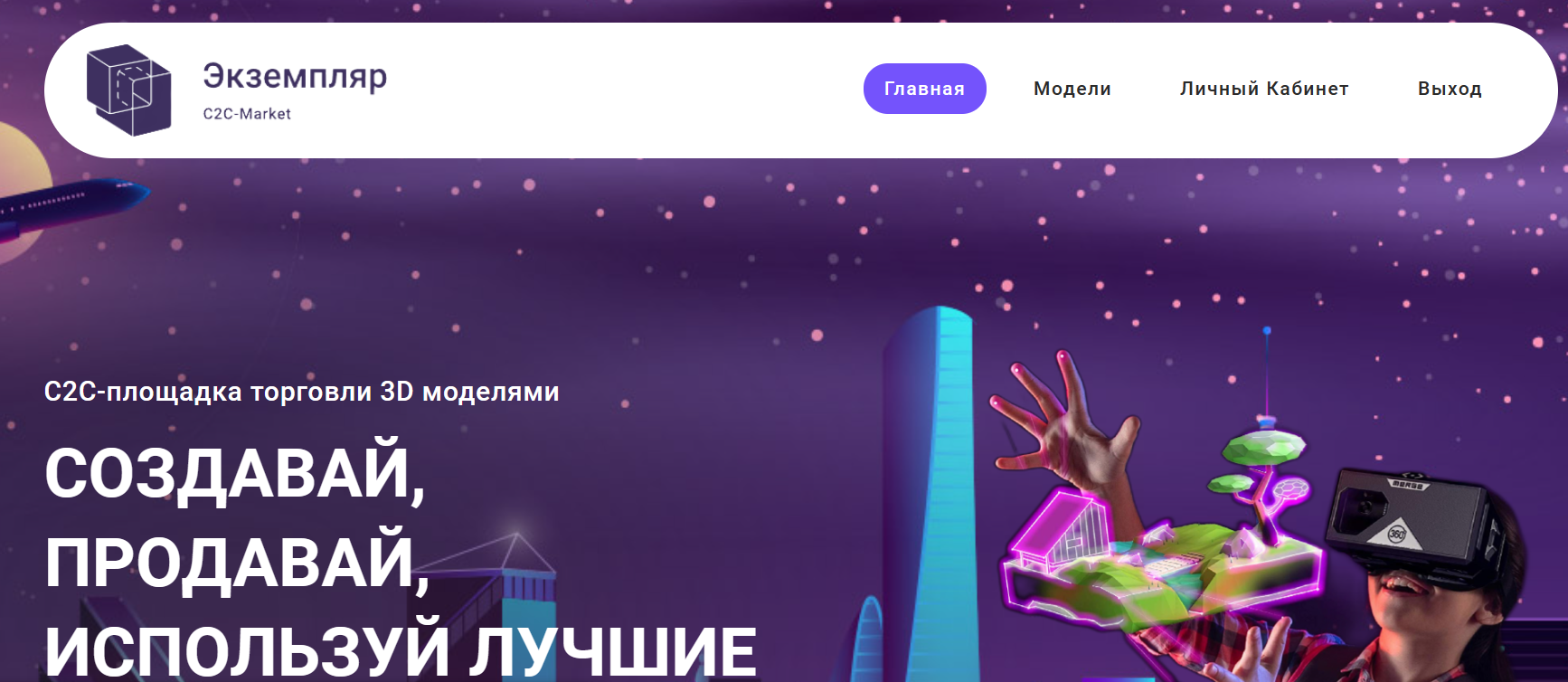


Рисунок 2.4 – Главная страница

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

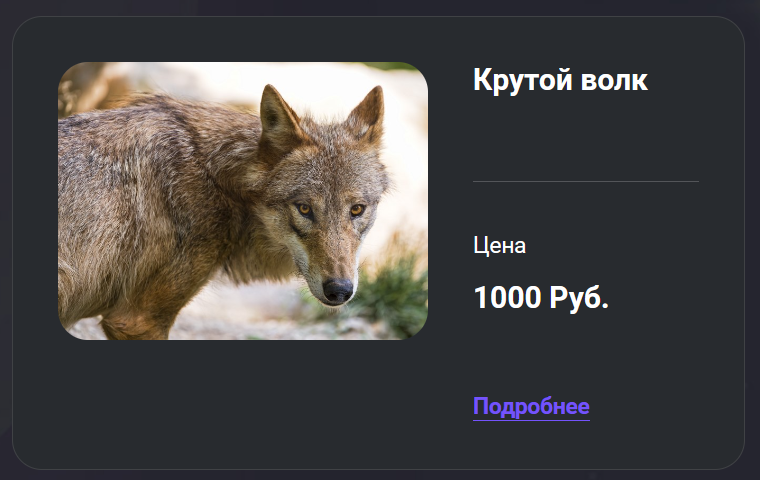


Рисунок 2.5 – Блок модели с кнопкой «Подробнее»

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

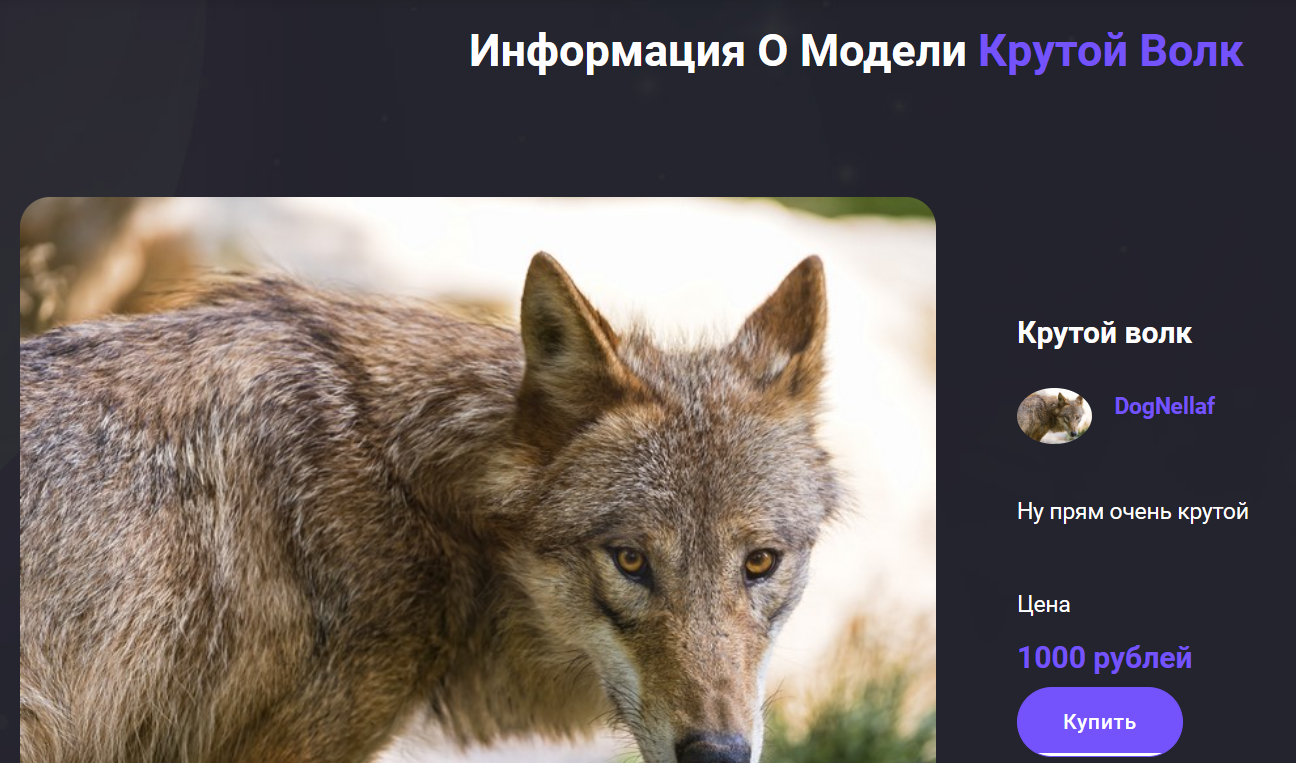


Рисунок 2.6 – Страница с информацией о модели

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).

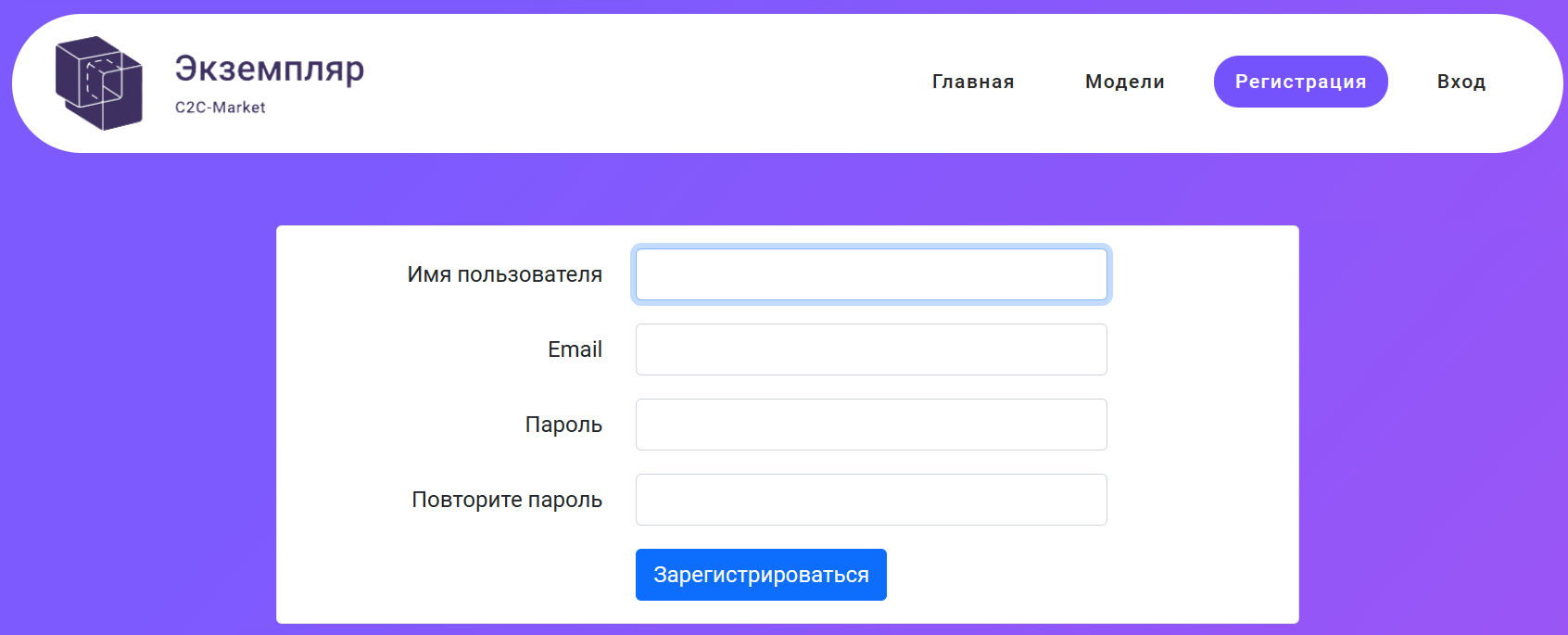


Рисунок 2.7 – Страница регистрации

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

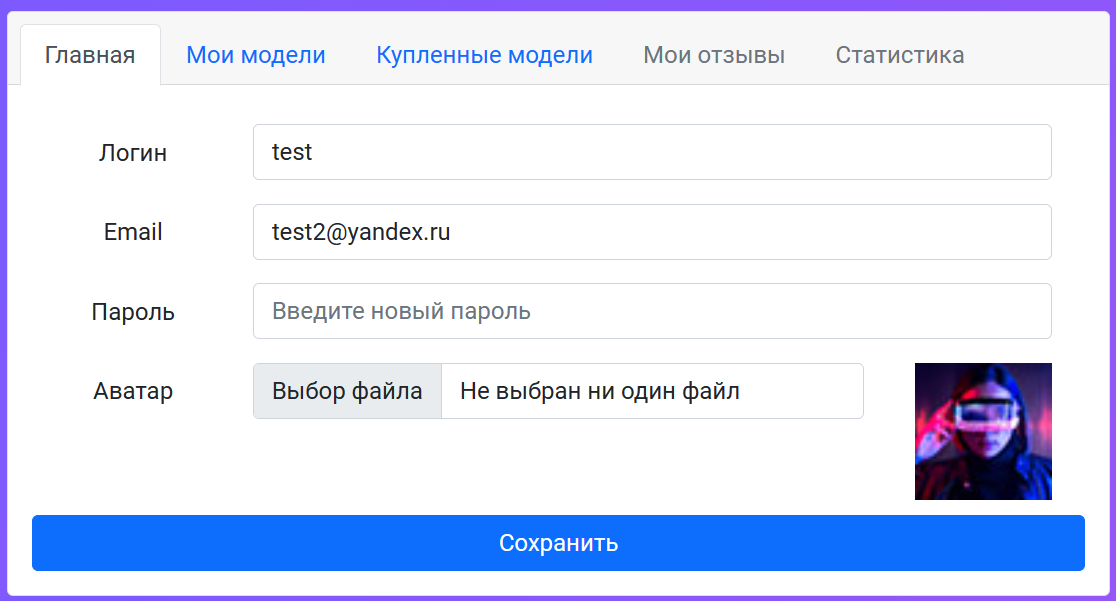


Рисунок 2.8 – Личный кабинет пользователя

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.

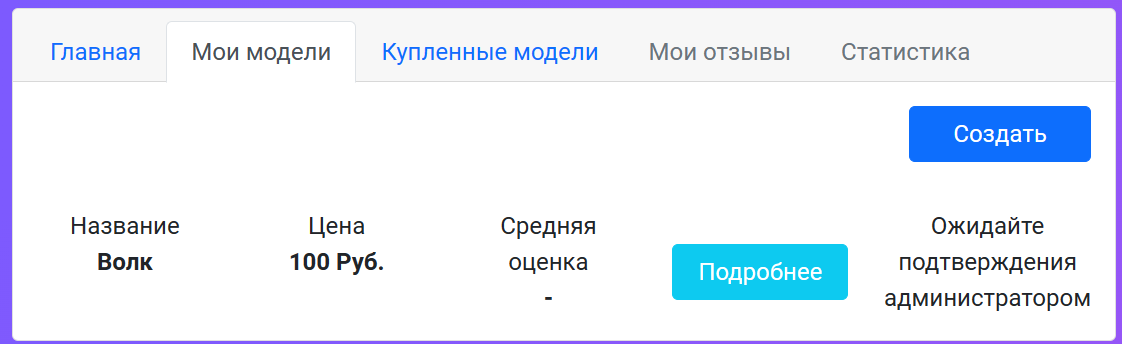


Рисунок 2.9 – Страница «Мои модели»

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).

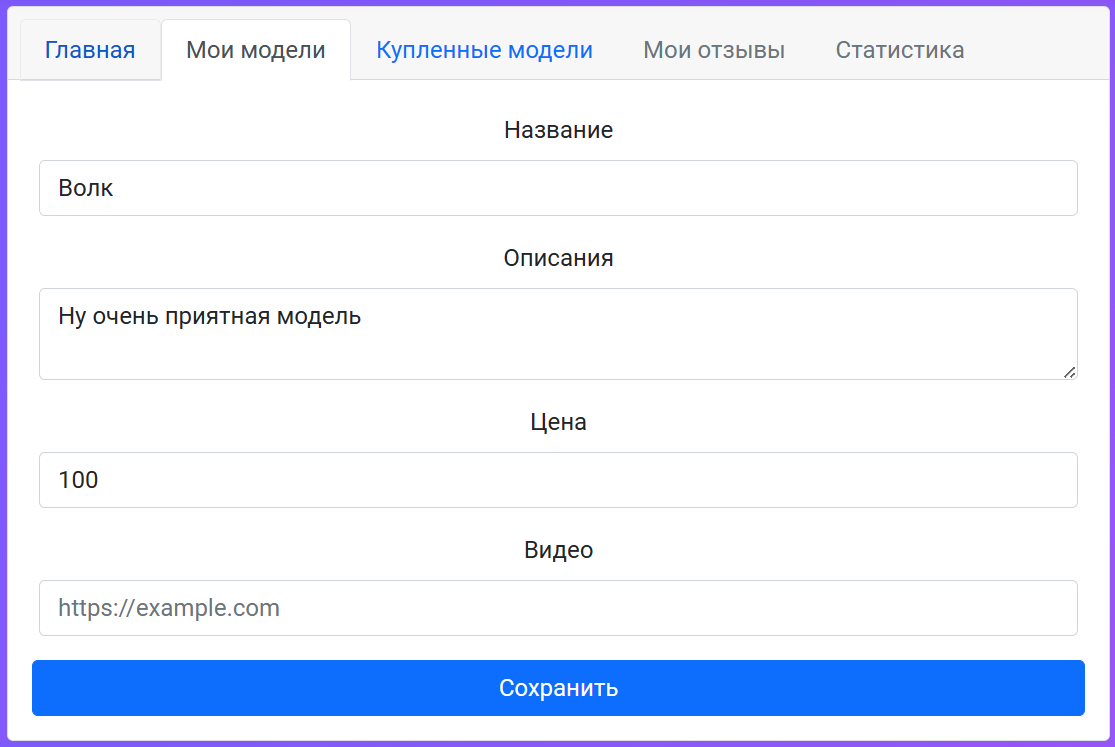


Рисунок 2.10 – Страница редактирования объявления

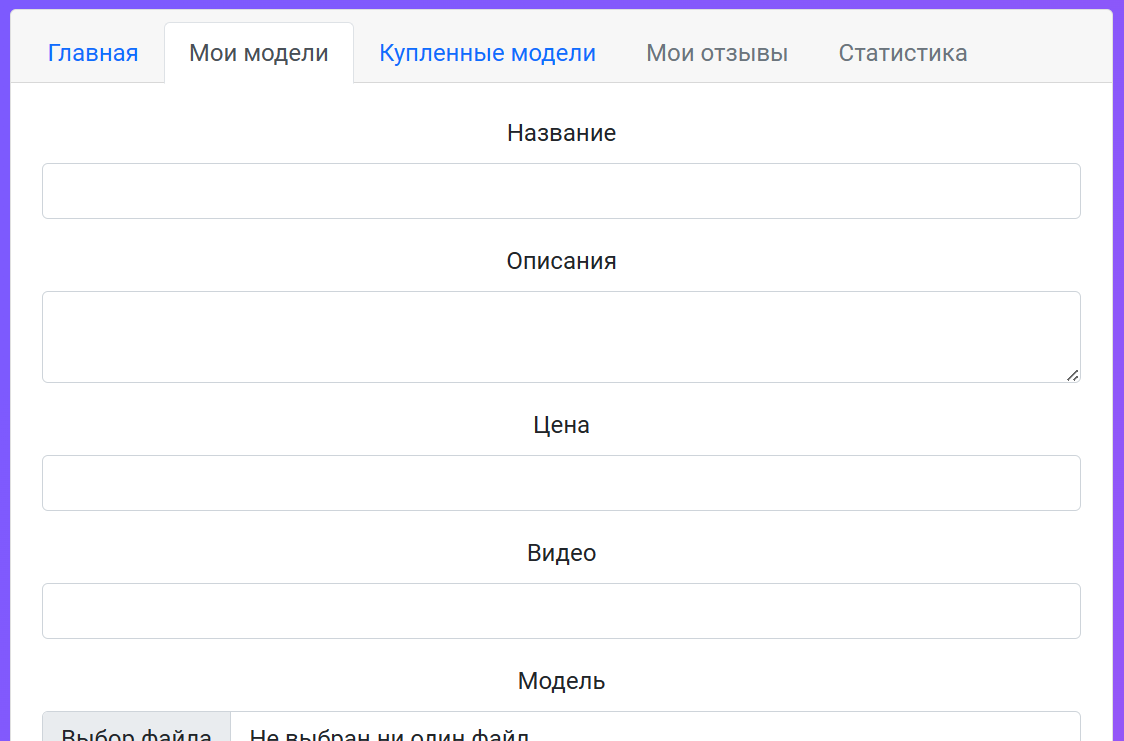


Рисунок 2.11 – Страница создания объявления

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

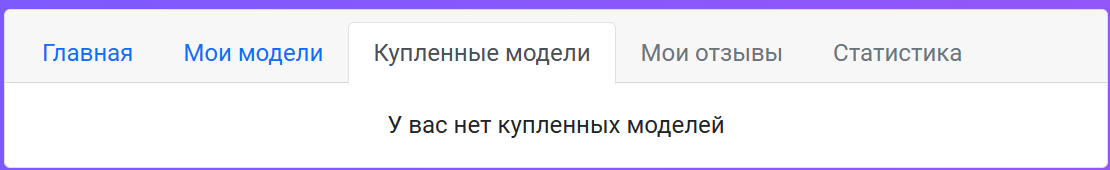


Рисунок 2.12 – Страница «Купленные модели»

# 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. 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
2. Смоленцева, Т. Е. Проектирование и разработка WEB-приложений: Практикум — Москва: РТУ МИРЭА, 2023. — 68 с. — https://e.lanbook.com/book/368954
3. Джош, Л. Современный PHP. Новые возможности и передовой опыт — Москва: ДМК Пресс, 2016. — 304 с. — URL: https://e.lanbook.com/book/93269
4. 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>
5. Борисов, Р. С. Информатика. Создание интернет-сайтов — Москва: РГУП, 2022. — 157 с. — URL: <https://e.lanbook.com/book/282227>
6. Рочев, К. В. Информационные технологии. Анализ и проектирование информационных систем — Санкт-Петербург: Лань, 2022. — 128 с. — URL: <https://e.lanbook.com/book/206894>
7. Федькова, Н. А. Современные технологии разработки программного обеспечения — Брянск: Брянский ГАУ, 2022. — 58 с. — URL: <https://e.lanbook.com/book/305087>
8. Чаллавала, Ш. MySQL 8 для больших данных — Москва: ДМК Пресс, 2018. — 226 с. — URL: <https://e.lanbook.com/book/131684>
9. Кожевникова, П. В. PHP и MySQL — Ухта: УГТУ, 2020. — 51 с. —URL: <https://e.lanbook.com/book/209591>
10. 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>
11. Зудилова, Т. В. Web-программирование HTML — Санкт-Петербург: НИУ ИТМО, 2012. — 70 с. — URL: <https://e.lanbook.com/book/40724>
12. Селина, Е. Г. Организация интерактивного взаимодействия в HTML-документах — Санкт-Петербург: НИУ ИТМО, 2018. — 35 с. — URL: <https://e.lanbook.com/book/136467>
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>
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>
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>
16. Янцев, В. В. JavaScript. Креативное программирование — 2-е изд., стер. — Санкт-Петербург: Лань, 2024. — 232 с. — URL: <https://e.lanbook.com/book/383837>
17. Диков, А. В. Web-программирование на JavaScript — Санкт-Петербург: Лань, 2024. — 168 с. — URL: <https://e.lanbook.com/book/387305>
18. Зенченко, И. В. Проектирование бизнес-процессов. Практические аспекты — Москва: ФЛИНТА, 2017. — 118 с. — URL: <https://e.lanbook.com/book/97142>
19. Вейцман, В. М. Проектирование информационных систем — Санкт-Петербург: Лань, 2022. — 316 с. — URL: <https://e.lanbook.com/book/206873>
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

# ПРИЛОЖЕНИЕ А

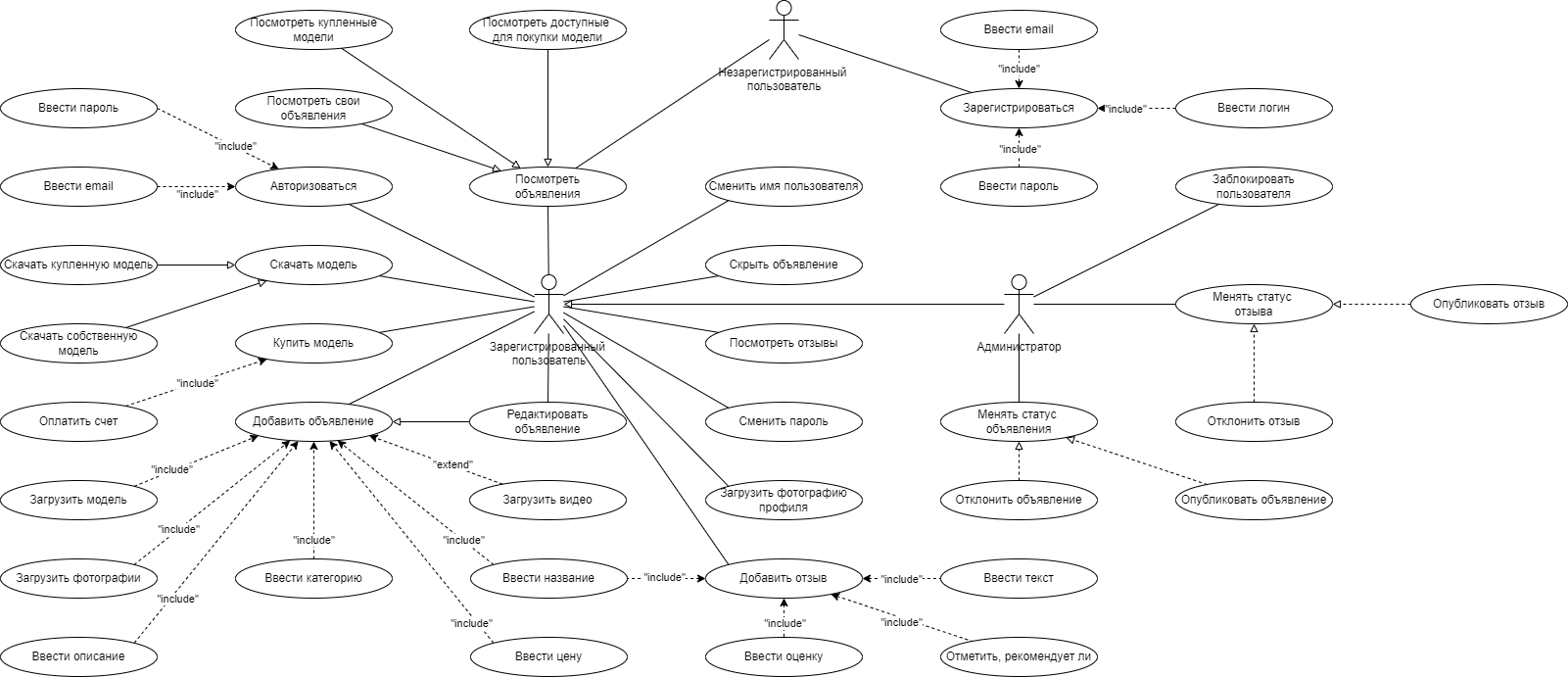


Рисунок А.1 – Диаграмма вариантов использования

# ПРИЛОЖЕНИЕ Б

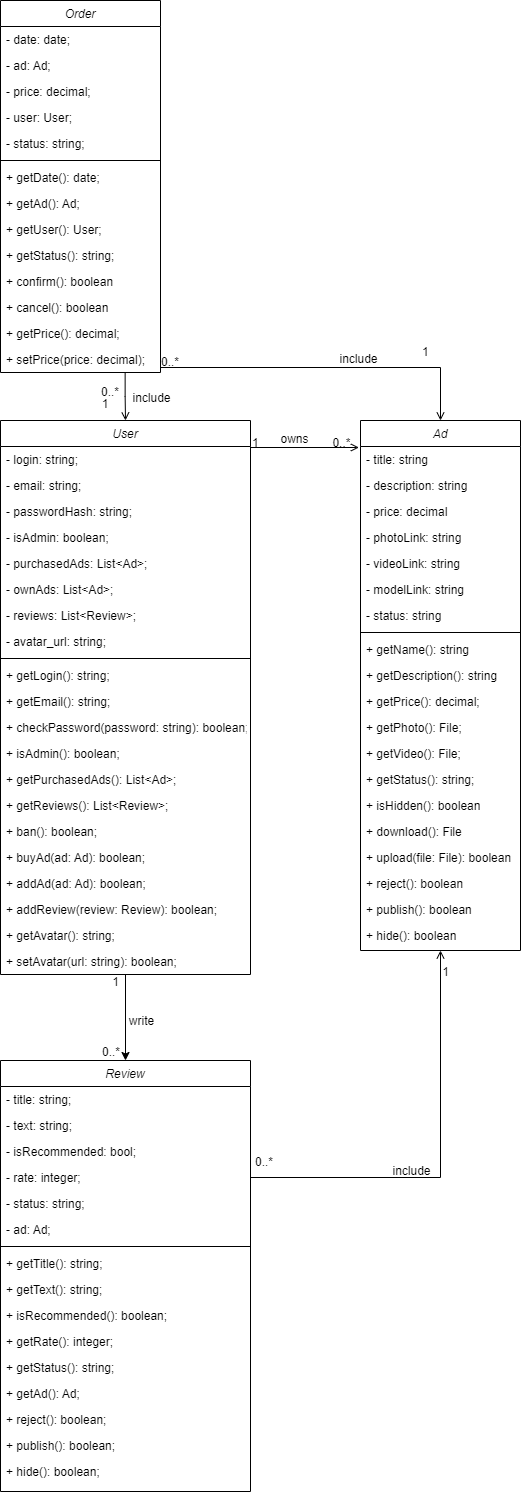


Рисунок Б.1 – Диаграмма классов

# ПРИЛОЖЕНИЕ В

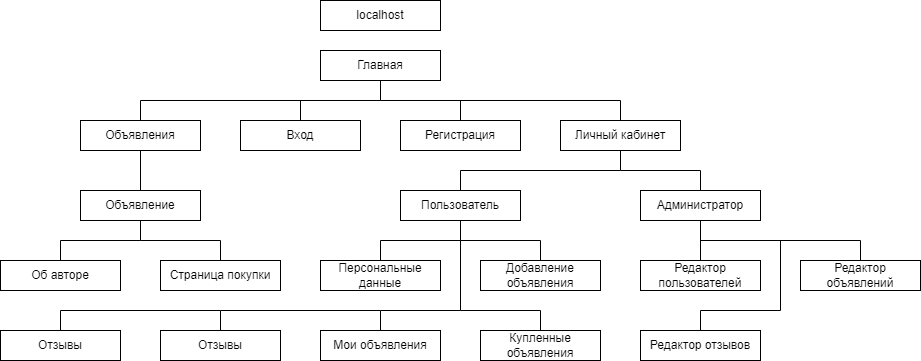


Рисунок В.1 – Логическая структура программы

Продолжение ПРИЛОЖЕНИя В

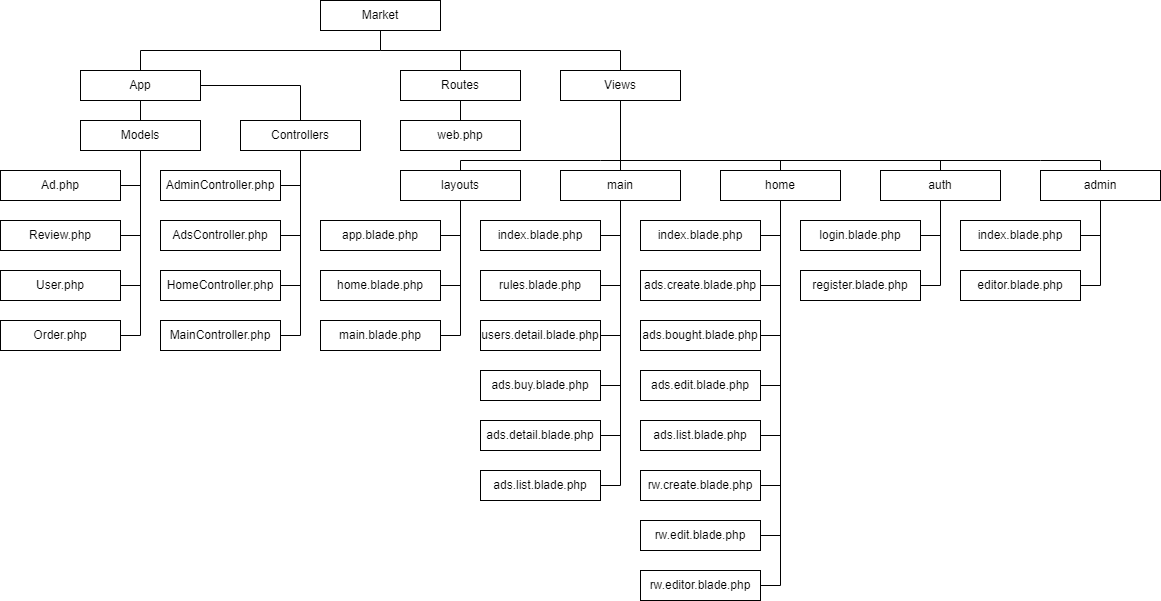


Рисунок В.2 – Физическая структура программы

# ПРИЛОЖЕНИЕ Г

AdminController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use App\Models\Review;

use App\Models\Ad;

class AdminController extends Controller

{

private const REVIEW\_VALIDATOR = [

'title' => 'required|max:255',

'text' => 'required',

'is\_recommended' => 'required|boolean',

'rate' => 'required|integer'

];

public function \_\_construct()

{

$this->middleware('auth');

}

public function index() {

$user = Auth::user();

if ($user->is\_admin == 0) {

return abort('403');

}

return view('admin/index');

}

public function ad\_editor() {

$user = Auth::user();

if ($user->is\_admin == 0) {

return abort('403');

}

$ads = Ad::paginate(10);

return view('ad/editor', compact('ads'));

}

public function review\_editor() {

$user = Auth::user();

if ($user->is\_admin == 0) {

return abort('403');

}

$reviews = Review::paginate(10);

return view('review/editor', compact('reviews'));

}

public function review\_create() {

$user = Auth::user();

if ($user->is\_admin == 0) {

return abort('403');

}

return view('review/create');

}

public function review\_store(Request $request) {

$user = Auth::user();

if ($user->is\_admin == 0) {

return abort('403');

}

$validated = $request->validate(self::REVIEW\_VALIDATOR);

Review::create([

'title' => $validated['title'],

'text' => $validated['text'],

'is\_recommended' => $validated['is\_recommended'],

'rate' => $validated['rate'],

'user\_id' => $user->id

]);

return redirect()->route('review-editor');

}

public function review\_delete(Request $request, Review $review) {

$user = Auth::user();

if ($user->is\_admin == 0) {

return abort('403');

}

$review->delete();

return redirect()->route('review-editor');

}

public function review\_edit(Review $review) {

$user = Auth::user();

if ($user->is\_admin == 0) {

return abort('403');

}

return view('review/edit', ['review' => $review]);

}

public function review\_update(Request $request, Review $review) {

$user = Auth::user();

if ($user->is\_admin == 0) {

return abort('403');

}

$validated = $request->validate(self::REVIEW\_VALIDATOR);

$review->fill(['title' => $validated['title'],

'text' => $validated['text'],

'is\_recommended' => $validated['is\_recommended'],

'rate' => $validated['rate'],

'status' => $review->status

]);

$review->save();

return redirect()->route('review-editor');

}

}

AdsController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Auth;

use App\Models\Ad;

use Illuminate\Http\Request;

class AdsController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

}

HomeController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Validation\Rule;

use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;

use App\Models\Ad;

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',

];

public function \_\_construct()

{

$this->middleware('auth');

}

protected function check\_auth\_user() {

$user = Auth::user();

if (!$user) {

abort(403);

}

return $user;

}

public function index()

{

$user = self::check\_auth\_user();

return view('home/index', ['user' => $user]);

}

public function own\_ads()

{

$user = self::check\_auth\_user();

return view('home/ads/index', [

'user' => $user,

'ads' => $user->ads()->get()

]);

}

public function bought\_ads()

{

$user = self::check\_auth\_user();

return view('home/ads/bought', [

'user' => $user,

'ads' => $user->bought\_ads()]);

}

public function reviews()

{

$user = self::check\_auth\_user();

throw new NotFoundHttpException("Страница не существует");

}

public function stats()

{

$user = self::check\_auth\_user();

throw new NotFoundHttpException("Страница не существует");

}

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');

}

public function ads\_create\_page() {

return view('home/ads/create', ['user' => self::check\_auth\_user()]);

}

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');

}

public function ads\_edit\_page(Ad $ad) {

return view('home/ads/edit', ['ad' => $ad,

'user' => self::check\_auth\_user()]);

}

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->save();

return redirect()->route('home.ads.own');

}

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');

}

}

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');

}

}

}

MainController.php

<?php

namespace App\Http\Controllers;

use App\Models\Ad;

use App\Models\User;

use Illuminate\Support\Facades\DB;

class MainController extends Controller

{

protected static function \_\_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)->get();

}

protected static function \_\_get\_ten\_most\_popular\_ads() {

return self::\_\_get\_most\_popular\_ads()->limit(10);

}

public function index() {

return view('main/index', ['ads' => self::\_\_get\_most\_popular\_ads()]);

}

public function ads() {

return view('main/ads/list', self::\_\_get\_most\_popular\_ads()->paginate(10));

}

public function buy(Ad $ad) {

return view('main/ads/buy', ['ad' => $ad]);

}

public function rules() {

return view('main/rules');

}

public function ad\_detail(Ad $ad) {

return view('main/ads/detail', ['ad' => $ad]);

}

public function user\_detail(User $user) {

return view('main/users/detail', ['user' => $user]);

}

}

Ad.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

use App\Models\User;

use App\Models\Order;

use App\Models\Review;

class Ad extends Model

{

protected $fillable = ['title',

'description',

'price',

'video\_link',

'model\_link',

'status',

'user\_id',

'photo\_link'

];

public function user() {

return $this->belongsTo(User::class);

}

public function orders() {

return $this->hasMany(Order::class);

}

public function reviews() {

return $this->hasMany(Review::class);

}

public function get\_average\_rating() {

$reviews = $this->reviews;

if ($reviews->count() != 0) {

return $reviews->pluck('rating')->avg();

}

return -1;

}

}

Order.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

use App\Models\User;

use App\Models\Ad;

class Order extends Model

{

protected $fillable = ['status'];

public function user() {

return $this->belongsTo(User::class);

}

public function ad() {

return $this->belongsTo(Ad::class);

}

}

Review.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use App\Models\User;

use App\Models\Ad;

class Review extends Model

{

protected $fillable = ['title',

'text',

'is\_recommended',

'rate',

'status'

];

public function user() {

return $this->belongsTo(User::class);

}

public function ad() {

return $this->belongsTo(Ad::class);

}

}

User.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Sanctum\HasApiTokens;

use App\Models\Ad;

use App\Models\Order;

use App\Models\Review;

class User extends Authenticatable

{

use HasApiTokens, HasFactory, Notifiable;

protected $fillable = [

'name',

'email',

'password',

'is\_banned',

'avatar\_url'

];

protected $hidden = [

'remember\_token',

'is\_admin'

];

protected $casts = [

'email\_verified\_at' => 'datetime',

'password' => 'hashed',

];

public function ads() {

return $this->hasMany(Ad::class);

}

public function orders() {

return $this->hasMany(Order::class);

}

public function bought\_ads() {

return $this->hasManyThrough(Order::class, Ad::class);

}

public function reviews() {

return $this->hasMany(Review::class);

}

}

Web.php

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\MainController;

use App\Http\Controllers\HomeController;

use App\Http\Controllers\AdsController;

use App\Http\Controllers\AdminController;

Auth::routes();

Route::get('/home', [HomeController::class, 'index'])->name('home.index');

Route::post('/home', [HomeController::class, 'save\_data'])->name('home.index.save');

Route::get('/home/models/own', [HomeController::class, 'own\_ads'])->name('home.ads.own');

Route::get('/home/models/bought', [HomeController::class, 'bought\_ads'])->name('home.ads.bought');

Route::get('/home/reviews', [HomeController::class, 'reviews'])->name('home.reviews');

Route::get('/home/stat', [HomeController::class, 'stats'])->name('home.stats');

Route::get('/home/models/own/create', [HomeController::class, 'ads\_create\_page'])->name('home.ads.create.page');

Route::post('/home/models/own/create', [HomeController::class, 'ads\_create\_method'])->name('home.ads.create.method');

Route::get('/home/models/own/{ad}/edit', [HomeController::class, 'ads\_edit\_page'])->name('home.ads.edit.page');

Route::patch('/home/models/own/{ad}/edit', [HomeController::class, 'ads\_edit\_method'])->name('home.ads.edit.method');

Route::patch('/home/models/own/{ad}/hide', [HomeController::class, 'ads\_hide\_method'])->name('home.ads.hide.method');

Route::patch('/home/models/own/{ad}/show', [HomeController::class, 'ads\_show\_method'])->name('home.ads.show.method');

Route::get('/admin', [AdminController::class, 'index'])->name('admin.index');

Route::get('/admin/ads', [AdminController::class, 'ad\_editor'])->name('admin.ads.editor');

Route::get('/admin/reviews', [AdminController::class, 'review\_editor'])->name('admin.ads.reviews.editor');

Route::get('/ads/edit/{ad}', [AdminController::class, 'ad\_edit'])->name('admin.ads.edit.page');

Route::patch('/ads/edit/{ad}', [AdminController::class, 'ad\_update'])->name('admin.edit.method');

Route::delete('/ads/{ad}', [AdminController::class, 'ad\_delete'])->name('admin.ads.delete.method');

Route::get('/reviews/create', [AdminController::class, 'review\_create'])->name('reviews.create.page');

Route::post('/reviews/create', [AdminController::class, 'review\_store'])->name('reviews.create.method');

Route::get('/reviews/edit/{review}', [AdminController::class, 'review\_edit'])->name('reviews.edit.page');

Route::patch('/reviews/edit/{review}', [AdminController::class, 'review\_update'])->name('reviews.edit.method');

Route::delete('/reviews/{review}', [AdminController::class, 'review\_delete'])->name('review.delete.method');

Route::get('/', [MainController::class, 'index'])->name('main.index');

Route::get('/ads', [MainController::class, 'ads'])->name('main.ads.list');

Route::get('/rules', [MainController::class, 'rules'])->name('main.rules');

Route::get('/ads/{ad}', [MainController::class, 'ad\_detail'])->name('main.ads.detail');

Route::get('/authors/{user}', [MainController::class, 'user\_detail'])->name('main.users.detail');

Route::get('/ads/{ad}/buy', [MainController::class, 'buy'])->name('main.ads.buy');