МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-сайта для фирмы по продаже авторских игровых кубиков для настольных ролевых игр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-352.ВКР

Научный руководитель,
профессор кафедры СП, д.г.н.,
к.ф.-м.н.
_____ С.М. Абдуллаев

Автор работы,
студент группы КЭ-403
_____ О.О. Шарко

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студенту группы КЭ-403
Шарко Олегу Олеговичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12) Разработка веб-сайта для фирмы по продаже авторских игровых кубиков для настольных ролевых игр.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Fresh in – New DnD/TTRPG Dice Sets | Critical Kit. [Электронный ресурс] URL: https://www.criticalkit.co.uk/ (дата обращения: 31.01.2024 г.).
3.2. Гаевский А.Ю. Создание Web-страниц и Web-сайтов. // Технолоджи – 3000, 2008. – 464 с

4. **Перечень подлежащих разработке вопросов**

4.1. Провести обзор и анализ литературы.
4.2. Спроектировать веб-сайт.
4.3. Реализовать веб-сайт.
4.4. Провести тестирование.

5. **Дата выдачи задания:** 29.01.2024 г.

Научный руководитель,
профессор кафедры СП, д.г.н., к.ф.-м.н. С.М. Абдуллаев

Задание принял к исполнению О.О. Шарко

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Настольные ролевые игры (НРИ) становятся все более популярными среди различных возрастных групп. Развитие культуры настольных игр создает запрос на уникальные элементы, такие как авторские кубики, чтобы обогатить игровой опыт.

Авторские игровые кубики предоставляют уникальные возможности для персонализации игрового опыта. Веб сайт предоставляет уникальные дизайны игровых кубиков.

Развитие веб-сайта для продажи кубиков обеспечит возможность расширения бизнеса на онлайн-рынок, привлекая клиентов со всего мира и обеспечивая им удобный доступ к уникальным продуктам.

Сайт будет предлагать уникальные коллекции и ограниченные выпуски игровых кубиков, создавая востребованные и коллекционные предметы. Это поможет поддерживать интерес и постоянство клиентов, а также стимулировать спрос на новые продукты.

Постановка задачи

Целью выпускной квалификационной работы является разработка и реализация веб-сайта для фирмы по продаже авторских игровых кубиков для настольно ролевых игр. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести обзор и анализ литературы;
2) спроектировать веб-сайт;
3) реализовать веб-сайт;
4) провести тестирование.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 42 страницы, объем списка литературы – 15 источников.

В первой главе описывается анализ предметной области, обзор существующих решений, а именно сайтов, также был описан и выбран бэкэнд и фронтэнд фреймворк

Вторая глава посвящена проектированию веб сайта, является ключевым этапом в процессе разработки, определяются основные функциональные возможности и структура. Создана и описана диаграмма вариантов использования, с ее помощью можно увидеть какие функции будут доступный пользователям и каким образом они могут взаимодействовать с системой.

Также создана и описана диаграмма базы данных, она представляет собой один из центральных элементов проектирования системы, ее помощью можно визуализировать взаимодействие пользователей с системой, определяя основные сценарии использования веб-сайта.

В третьей главе описаны особенности разработки приложения, основные инструменты, используемые при реализации проекта и компонентов системы, а также приведены скриншоты интерфейса, они помогают визуализировать конечный продукт и показывают, как различные компоненты системы интегрируются для создания единого пользовательского опыта.

В четвертой главе проведено функциональное и usability тестирование вебсайта.

В заключении перечислены основные результаты работы.

В приложении содержатся фрагменты исходного кода.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области и обзор решений

Целью данной работы является создание и запуск веб-сайта для фирмы, специализирующейся на продаже уникальных игровых кубиков, предназначенных для настольных ролевых игр. Разрабатываемый веб-сайт станет платформой, предоставляющей клиентам возможность приобрести авторские кубики, а также узнать больше о разнообразии продукции.

Проведение анализа аналогичных веб-сайтов перед проектированием собственного имеет ряд преимуществ. Во-первых, это помогает определить основные требования к структуре и содержанию сайта. Во-вторых, анализ позволяет выявить положительные и отрицательные аспекты существующих сайтов, что облегчает принятие решений о дизайне и функциональности.

Для более подробного анализа, ниже приведены следующие ключевые аспекты.

1. Структура сайта: оценка организации разделов и подразделов.
2. Навигация: изучение удобства перемещения по сайту, наличие поиска, организация меню.
3. Дизайн и пользовательский интерфейс [1] (UI/UX): анализ визуального оформления, цветовой гаммы, шрифтов и компоновки элементов.
4. Функциональность: просмотр доступных сервисов и инструментов для пользователей, возможность регистрации, комментирования, покупки и т. д.
5. Контент: изучение представленного контента: статьи, видео, изображения, его структурирование.
6. SEO-аспекты: анализ использования ключевых слов, метаданных, оптимизации страниц для поисковых систем.
7. Отзывы и обратная связь: просмотр отзывов пользователей о сайтах, реакция на обратную связь.

DIE HARD dice [2]

На главной странице сайта располагается логотип, меню для навигации по сайту. Также в шапке сайта расположены кнопки для авторизации и регистрации пользователей. На сайте доступен поиск по ключевому слову. Имеется корзина и вишлист (рисунок 1).

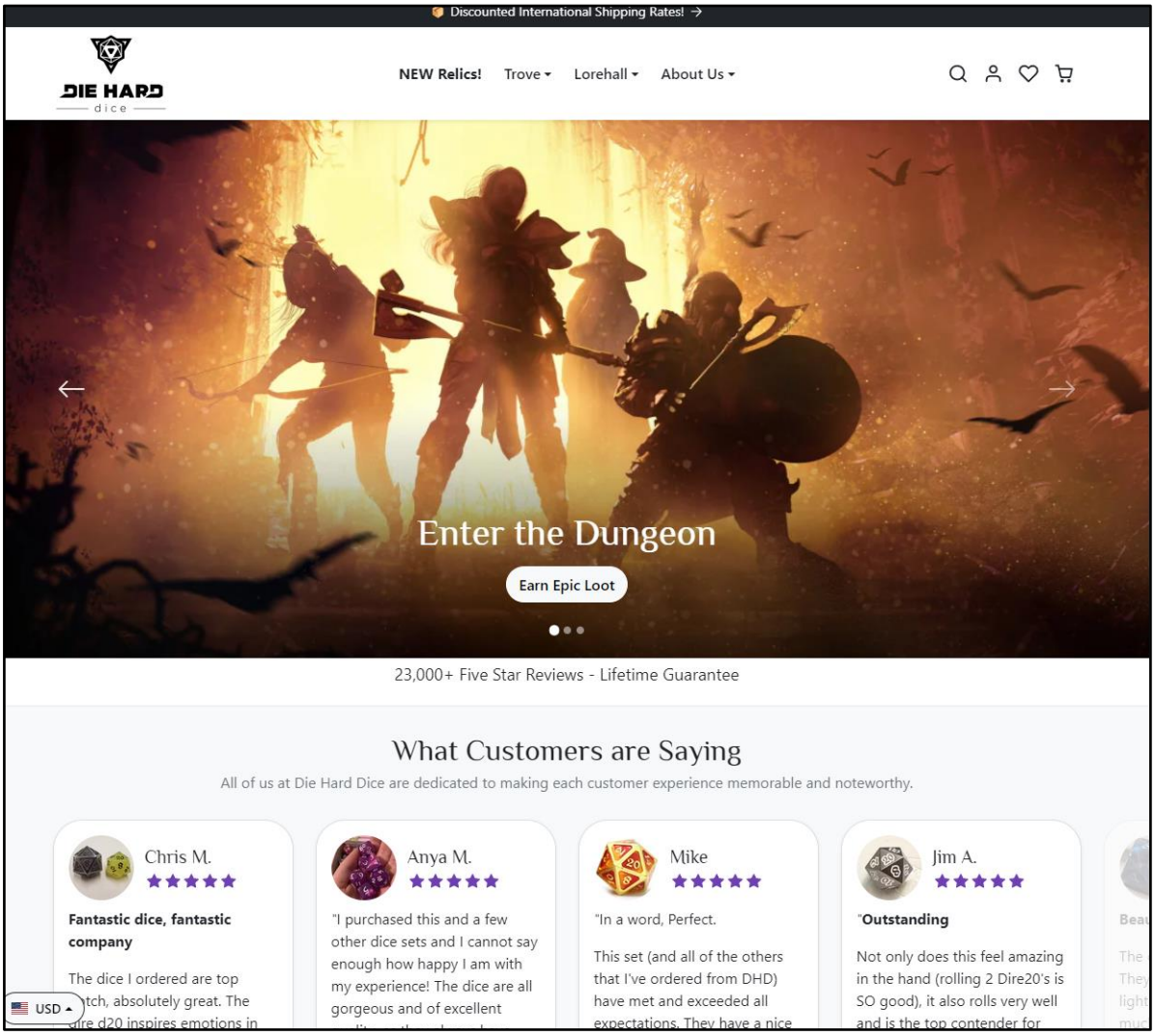

Рисунок 1 – DIE HARD dice

Основное пространство сайта занимает информация о настольно-ролевой атрибутики. Также можно увидеть партнеров и спонсоров сайта, с которыми он сотрудничает в данный момент.

Из плюсов можно отметить, что есть отзывы пользователей, которые приобретали настольно-ролевую атрибутику, отсутствие навязчивой

рекламы, указаны партнеры и спонсоры, которые способствую доверию к сайту.

DICE DUNGEON [3]

Основное пространство сайта занимает акции и выгодные предложения веб сайта (рисунок 2). Шапка сайта схожа с предыдущим сайтом. Также содержит кнопки для авторизации и регистрации пользователя, есть поиск по ключевому слову и имеется корзина.

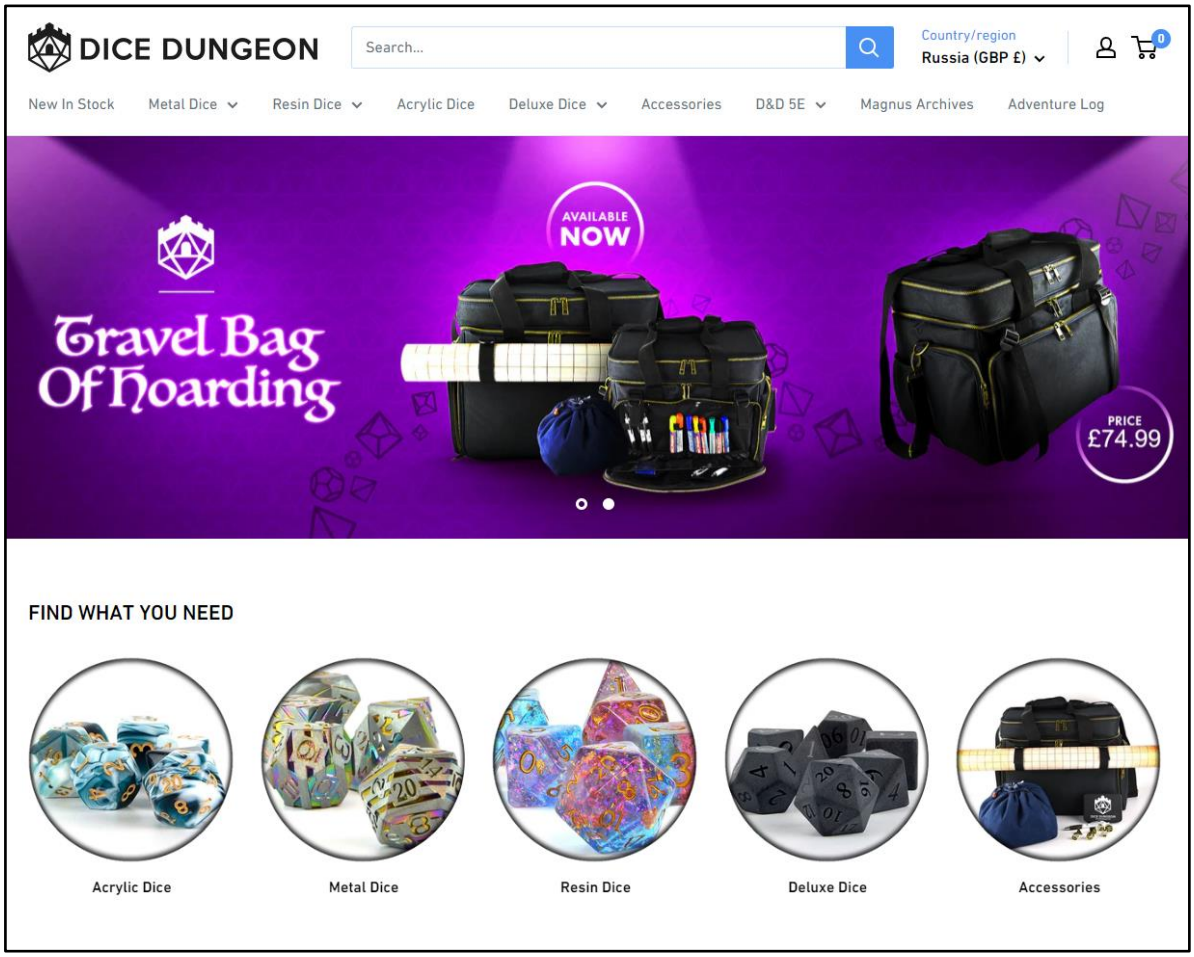

Рисунок 2 – DICE DUNGEON

На данном сайте присутствуют отзывы покупателей, но много навязчивых акций и предложений, которые очень бросаются в глаза пользователю, что в целом неудобно, отсутствуют партнеры и спонсоры, которые дают уверенность в сайте.

1.2. Обзор бэкенд фреймворка

Flask [4]

Flask – это легкий фреймворк, обычно называемый микрофреймворком. Ключевой особенностью, выделяющей его среди конкурентов, стали его возможности кастомизации.

Flask поставляется всего с некоторыми стандартными функциями, но зато он позволяет разработчикам добавлять любое количество библиотек и плагинов для улучшения функциональности. Это могут быть расширения для проверки формы, объектно-реляционные преобразователи, открытые системы аутентификации, механизм загрузки и ряд других инструментов.

Преимущества использования фреймворка Flask.

1. Среди Python-фреймворков Flask выделяется следующими особенностями.

2. Простая для понимания разработка. Фреймворк прост для понимания, поэтому он лучше всего подходит для начинающих.

3. Гибкость. В отличие от других веб-фреймворков, Flask позволяет полностью контролировать разработку. Программист сам с помощью внешних функций собирает всю структуру приложения. В фреймворке есть только несколько составляющих, которые нельзя изменить. Все остальные части открыты для изменений.

4. Использование Flask позволяет проводить модульное тестирование благодаря встроенной поддержке, встроенному серверу разработки, быстрому отладчику и удобной диспетчеризации запросов.

Недостатки использования фреймворка Flask.

1. Долгая первичная настройка проекта. Так как во Flask практически весь функционал устанавливается через сторонние расширения, то начальный этап разработки с фреймворком будет дольше по сравнению с тем же Django, где все необходимое идет из коробки.

2. Использование дополнительных модулей не всегда является преимуществом. Некоторые библиотеки, интегрированные в продукт, не

остаются на рынке и со временем теряют поддержку. Это означает, что их придется либо обслуживать самостоятельно, либо менять библиотеку, что увеличивает стоимость поддержки приложения.

3. Также использование сторонних модулей может рассматриваться как вмешательство третьих лиц, что может стать серьезной брешью в системе безопасности.

FastAPI [5]

FastAPI – это достаточно новая инфраструктура для веб-разработки, совместимая только с версиями Python 3.6+. Его можно назвать микрофреймворком, так как его встроенные возможности не так масштабны.

При этом FastAPI является одним из самых популярных фреймворков Python для создания сервисов REST. Он компенсирует ряд недостатков других инструментов: он простой в использовании и в изучении, легко разворачивается, а также дает разработчику преимущество асинхронной обработки запросов.

Преимущества использования фреймворка FastAPI.

1. Как следует из его названия, FastAPI делает упор на скорость. Фреймворк работает быстрее по сравнению с другими платформами Python. По ряду бенчмарков он показал лучшие результаты, чем Django или Flask.

2. При этом FastAPI выигрывает не только в количестве обрабатываемых запросов в секунду, но и в скорости и простоте разработки. Разработчик может развернуть проект с помощью всего 5 строк кода.

3. Встроенная проверка данных позволяет разработчикам писать более компактный код, пропуская дополнительные проверки. Она обнаруживает неправильные типы данных даже в глубоко вложенных запросах и возвращает обоснование в формате JSON. По словам авторов, это снижает количество ошибок разработчиков на 40%.

4. Самая захватывающая особенность FastAPI заключается в том, что он поддерживает асинхронный код из коробки. Эта способность была

добавлена в Python только в версии 3.4, поэтому FastAPI не работает с более ранними версиями. Благодаря асинхронному коду запросы выполняются независимо друг от друга и могут запускаться параллельно, а это значит, что время выполнения значительно сокращается по сравнению с выполнением при последовательном запуске.

5. FastAPI предлагает чрезвычайно удобную систему автоматического документирования. Он предоставляет пользовательский интерфейс на основе браузера, который в интерактивном режиме документирует API на основе графического пользовательского интерфейса Swagger UI.

6. Кроме того, разработчики легко могут получить альтернативную документацию, содержащую все перечисленные конечные точки. Документация всегда позволит разработчикам легко объяснить программу другим, упрощает использование бэкэнда фронтенд-специалистами и упрощает тестирование конечных точек API.

Недостатки использования фреймворка FastAPI.

1. FastAPI не предлагает встроенной системы безопасности. Вместо этого он предоставляет модуль «fastapi.security» для механизмов безопасности. В то же время он поддерживает OAuth2.0 для механизма аутентификации. Это значит, что настройки безопасности возможны, но с помощью дополнительных модулей.

2. Поскольку FastAPI появился только в 2018 году, его сообщество невелико по сравнению с другими фреймворками, и по нему пока что очень мало внешних образовательных материалов. Наработок сообщества пока недостаточно для решения любых проблем. В то же время, популярность фреймворка растет, так что в ближайшие годы ситуация будет изменяться в лучшую сторону.

3. FastAPI отлично работает там, где важна скорость и асинхронная обработка данных. Например, его использует Netflix для своего внутреннего управления.

4. Фреймворк также безупречно масштабируется при развертывании моделей машинного обучения, поскольку они лучше всего работают в микросервисной среде, обернутой вокруг REST API.

Django [6]

Это веб-фреймворк высокого уровня, который позволяет выполнять быструю разработку. Основной целью его инфраструктуры является создание сложных веб-сайтов, управляемых базами данных.

В Django разработчик может использовать наиболее популярные функции из коробки, соответственно, ему остается построить именно бизнес-логику.

Преимущества использования фреймворка Django.

1. Это универсальная структура, которую можно использовать в приложениях любого масштаба (социальной сети, новостного сайта, системы управления контентом и т. д.) Также он интегрируется с содержимым в любом формате, таком как HTML, XML, JSON и т. д. Он работает в паре с любым JavaScript-фреймворком.

2. В нем учтены стандартные функции безопасности, например управление учетными записями пользователей, управление транзакциями, подделка межсайтовых запросов, кликджекинг и многое другое.

3. Масштабируемость. Фреймворк следует шаблонам и принципам проектирования для повторного использования и поддержки кода. Главный принцип «Не повторяйся» гарантирует отсутствие повторов. Поскольку Django – это фреймворк, основанный на компонентах, каждый уровень независим от другого, что делает приложение масштабируемым на любом уровне.

4. Встроенная система интернационализации позволяет создавать многоязычные сайты без использования дополнительных плагинов.

Недостатки использования фреймворка Django.

1. Это монолитная платформа. Для кого-то это будет преимуществом, но не всегда. Фреймворк имеет определенный набор файлов и

предопределенных переменных. Его файловую структуру легко освоить, но он не позволяет использовать собственную.

2. Не подходит для небольших проектов. Вся функциональность Django поставляется с большим количеством кода. Его обработка занимает время сервера, что создает некоторые проблемы для слабого железа.

3. Исходя из всего этого, было принято решение, что разрабатывать бэкенд на Django фреймворке.

1.3. Обзор фронтенд фреймворка

React [7]

React.js, он же ReactJS или просто React, варианты написания разнятся, это библиотека JavaScript с открытым исходным кодом, которая использует рендеринг на стороне сервера (SSR) без задержки на загрузку элементов JavaScript, чтобы создавать гибкие и быстрые пользовательские интерфейсы.

Преимущества использования фреймворка React.

1. Одним из ключевых преимуществ ReactJS является его модульная конструкция. React.js использует компонентную архитектуру, которая позволяет разработчикам создавать многократно используемые компоненты для пользовательского интерфейса. Это облегчает обслуживание и масштабирование приложений.

2. Модель «дерева» React означает, что проблема на вершине дерева может распространиться по всему дереву. Чтобы решить эту проблему, команда разработчиков Facebook создала виртуальный DOM, который направляет трафик и запросы более эффективно. При любом рассмотрении преимуществ и недостатков React.js, это явный игровой момент для библиотеки. Она обеспечивает ключевую скорость и точность для приложений с большим объемом данных.

3. В React данные движутся от вершины к основанию дерева. И это оказывает огромное стабилизирующее влияние на ваш код. Небольшие изменения или ошибки в «дочерней» структуре не влияют на код «родителя». Такой тип связывания данных приводит к тому, что код становится более стабильным и быстрее выполняется. Оценивая плюсы и минусы использования ReactJS, это важное «за», о котором следует помнить.

4. Редактирование кода React стало еще проще благодаря инструментам разработчика, разработанным в качестве dev-расширения в Chrome и Firefox. Это позволяет программистам просматривать иерархии компонентов React в том виде, в котором они отображаются в виртуальном DOM.

Angular [8]

Angular – это фреймворк от компании Google для создания продвинутых бесшовных (одностраничных) веб-приложений SPA – (Single Page Applications) – на языках программирования TypeScript, JavaScript, Dart.

Преимущества использования фреймворка Angular.

1. Angular помогает привязывать компоненты приложения друг к другу, передавать данные, анимировать интерфейсы и пр. Для простых проектов его функциональность может быть избыточной, но для сложных SPA-приложений она незаменима.

2. Фреймворк позволяет создавать не только веб-приложения. С его помощью можно писать код, который может быть адаптирован под другую среду. Например, приложение сможет работать в мобильной или десктопной операционной системе. С помощью Angular можно создать даже приложение для дополненной реальности.

3. Особенность Angular – подробная документация. Она содержит рекомендации к построению и разработке приложений, style guide – гайд по стилю программирования на Angular. Это удобно для разработчиков, которые впервые столкнулись с фреймворком. Единство стиля помогает программистам лучше понимать код друг друга.

4. Разработчики Angular – сотрудники Google, а поддержка большой корпорации помогает фреймворку развиваться. При этом благодаря свободной лицензии и открытому исходному коду развивать его могут и сторонние разработчики.

Недостатки использования фреймворка Angular.

1. Angular считается одним из самых сложных фронтенд-фреймворков. Его может быть нелегко изучить с нуля самостоятельно. Кроме того, для начала работы потребуется знать не только «чистый» JavaScript, но и TypeScript, который на нем основан.

2. Несмотря на похожие названия, AngularJS и Angular несовместимы и принципиально разные. Поэтому разработчикам, которые сталкиваются с legacy-кодом на AngularJS, требуется изучить основы работы с устаревшим фреймворком. Концепции и правила нового Angular не подойдут.

VueJS [9]

VueJS – это прогрессивный фреймворк с открытым исходным кодом, который разработан для постепенного внедрения, поскольку основная библиотека ориентирована только на слой представления. Тем не менее, он более чем способен обеспечить работу сложных одностраничных приложений с помощью современных инструментов и при наличии библиотек для их поддержки.

Преимущества использования фреймворка VueJS.

1. Фреймворк VueJS крошечный, почти комичный. Его размер составляет всего 18 килобайт – мигание занимает больше времени, чем загрузка.

2. VueJS очень прост в освоении, и именно эта характеристика стала основной причиной его широкого и все более широкого распространения среди программистов. Чтобы начать работать с Vue, не нужно разбираться в библиотеках, JSX или TypeScript на уровне эксперта, как в случае с

Angular или React. Все, что требуется, – это базовые знания HTML, CSS и JavaScript.

3. Спустя 6 лет после выпуска VueJS накопила мощный набор инструментов для модульного и сквозного тестирования, а также систему установки плагинов. Учитывайте, что VueJS имеет свой собственный браузер.

4. DOM – это представление HTML-страниц со стилями, элементами и содержимым страницы в виде связанных объектов. Это создает нечто похожее на перевернутое семейное дерево, где документ разветвляется на различные части, связанные линиями, которые демонстрируют отношения.

Недостатки использования фреймворка VueJS.

1. VueJS был разработан в Китае и поэтому чрезвычайно популярен там. Некоторые обсуждения на форумах, описания плагинов и обучающие документы написаны на китайском языке, поэтому не исключено, что в процессе работы что-то будет потеряно при переводе.

2. Так как VueJS финансируется и разрабатывается сообществом и не имеет значительной поддержки, ему не хватает поддержки для адаптации к крупномасштабным проектам. Технология не является достаточно стабильной или сильно поддерживаемой, чтобы предложить мгновенные исправления проблем.

3. Поскольку VueJS существует не так долго, как его конкуренты, и ему потребовалось время, чтобы прорваться на рынок и получить массовое распространение среди программистов, существует феномен, когда на рынке труда нет большого количества опытных разработчиков VueJS.

По итогу выбором стал Vue.js из-за его простоты и легкости в освоении. Vue.js обладает интуитивно понятным синтаксисом, также Vuejs набирает большую популярность имея большой функционал.

2. ПРОЕКТИРОВАНИЕ

2.1. Варианты использования

На рисунке 3 представлена диаграмма вариантов использования.

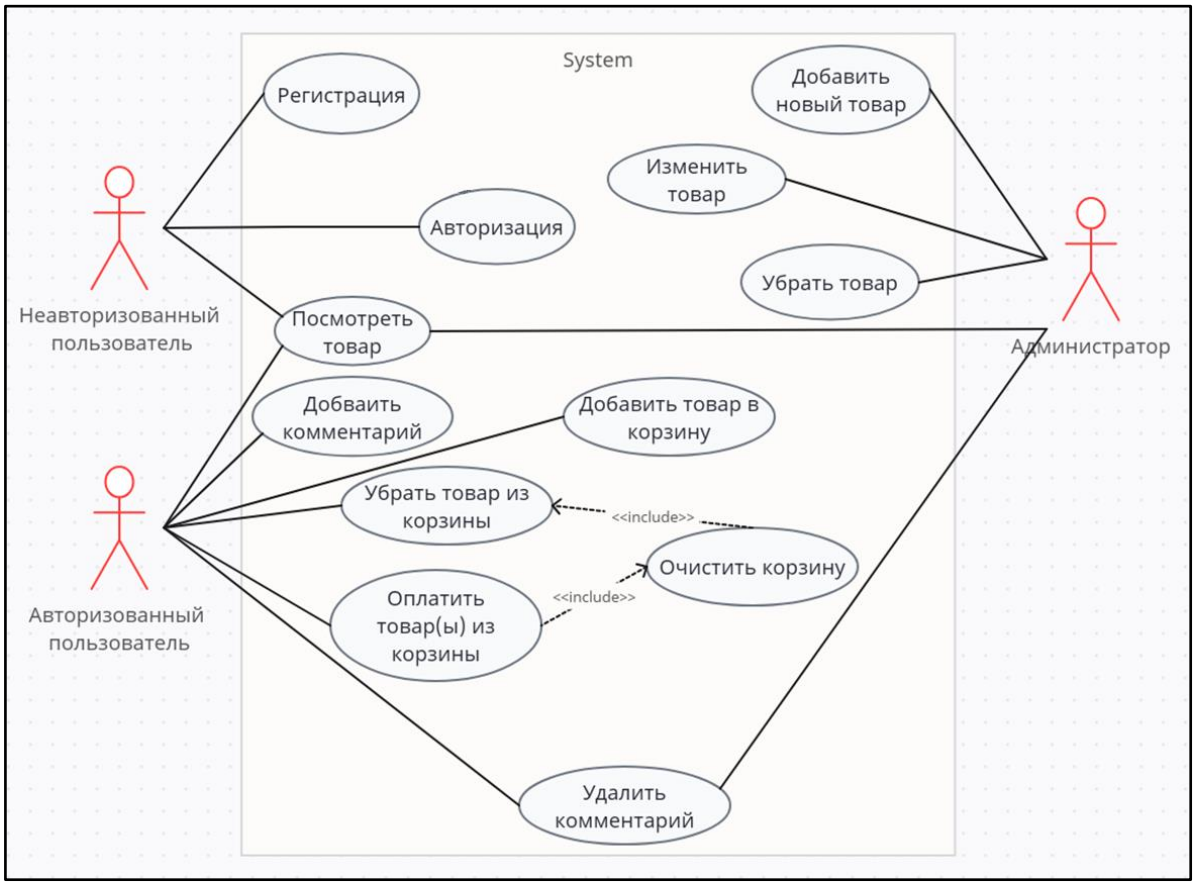

Рисунок 3 – Диаграмма вариантов использования [10]

«Администратор» – это лицо, имеющее права на создание, редактирование и удаление товаров.

«Неавторизованный пользователь» – это любой незарегистрированный на сайте человек, он не имеет доступа корзине, добавлению комментариев.

«Авторизованный пользователь» – это человек зарегистрированный и прошедший авторизацию на сайте, ему доступен основной функционал веб-сайта.

Краткое описание вариантов использования для актера «Администратор» представлено ниже.

1. Добавить новый товар. «Администратор» может добавить новый товар.

2. Изменить товар. «Администратор» может редактировать описание товара, его изображение и название.

3. Убрать товар. «Администратор» может удалить товар, который на данный момент отсутствует.

Краткое описание вариантов использования для актера «Неавторизованный пользователь» представлено ниже.

1. Регистрация. Возможность зарегистрироваться в системе.

2. Авторизация. Возможность авторизоваться в системе под своими учетными данными.

3. Просмотреть товар. Возможность изучить и просмотреть товар.

Краткое описание вариантов использования для актера «Авторизованный пользователь» представлено ниже.

1. Просмотреть товар. «Пользователь» может ознакомиться с доступными на данный момент товарами.

2. Добавить комментарий. Возможность написать комментарий к товару, а также оценить его.

3. Удалить комментарий. «Пользователь» может удалить комментарий.

4. Добавить товар в корзину. «Пользователь» может добавить любой понравившийся товар к себе в корзину.

5. Убрать товар из корзины. «Пользователь» может убрать любой товар из корзины.

6. Очистить корзину. «Пользователь» может быстро очистить корзину.

7. Оплатить товар(ы) из корзины. «Пользователь» может оплатить товары в корзине.

2.2. Диаграмма базы данных

На рисунке 4 представлена диаграмма базы данных.

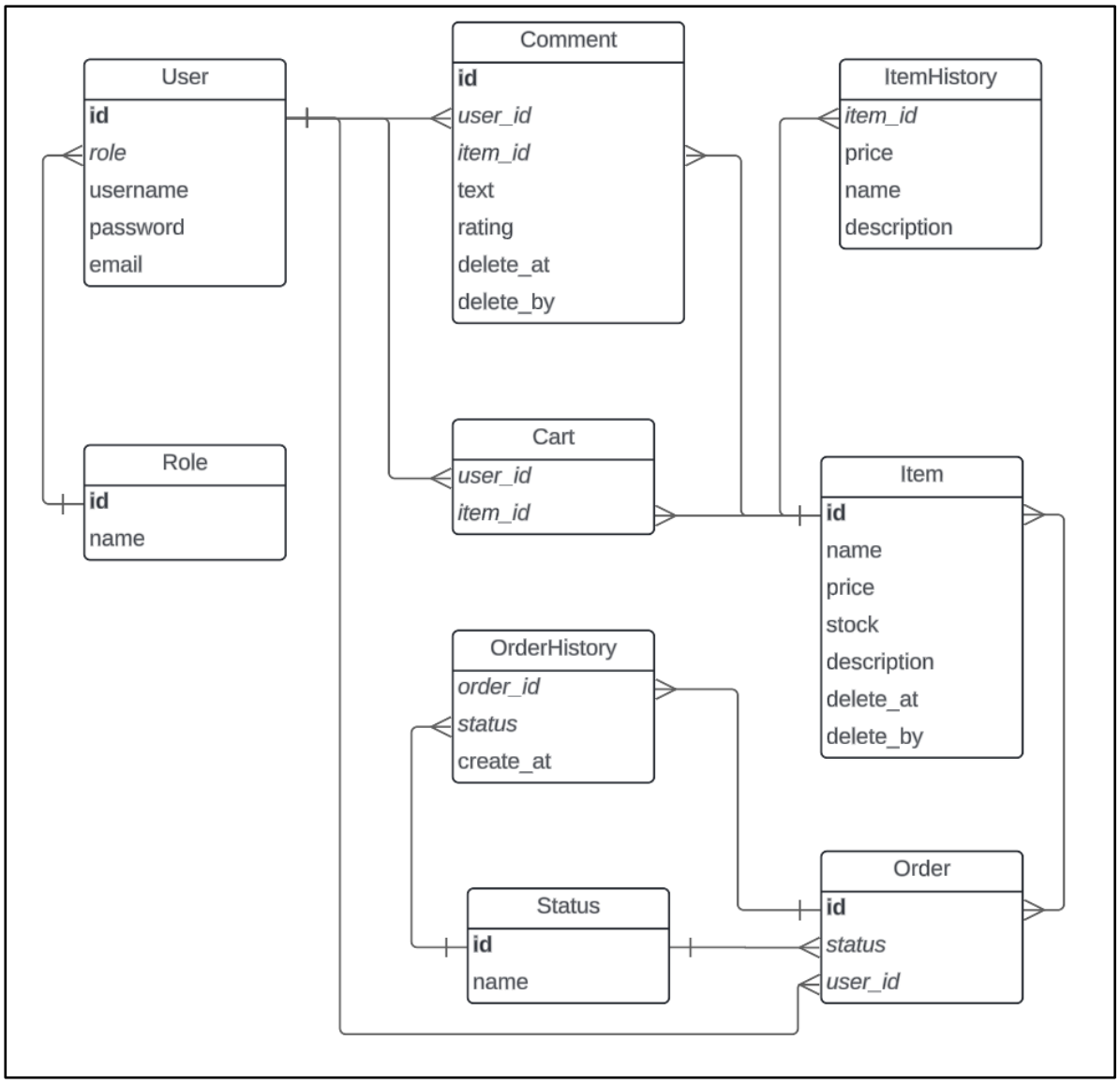

Рисунок 4 – Диаграмма базы данных

Таблица Role является компонентом базы данных и имеет следующее содержание:

1) id – это идентификатор, который является первичным ключом;

2) name – соответственно название роли, является либо авторизованным пользователем, либо администратором.

Таблица User является компонентом базы данных и имеет следующее содержание:

1) id – это идентификатор, который является первичным ключом;

2) role – внешний ключ к таблице роль;
3) username – никнейм пользователя;
4) email – электронная почта пользователя;
5) password – пароль пользователя.

Таблица Item является компонентом базы данных и имеет следующее содержание:

1) id – это идентификатор, который является первичным ключом;
2) name – название товара;
3) price – цена товара;
4) stock – количество в наличии;
5) description – описание товара;
6) delete_at – время удаления товара;
7) delete_by – кем удален товар.

Таблица ItemHistory является компонентом базы данных и имеет следующее содержание:

1) item_id – внешний ключ к таблице Item;
2) price – цена товара;
3) name – название товара;
4) description – описание товара.

Таблица Cart компонентом базы данных и имеет следующее содержание:

1) user_id – внешний ключ к таблице User;
2) item_id – внешний ключ к таблице Item.

Таблица Comment является компонентом базы данных и имеет следующее содержание:

1) id – это идентификатор, который является первичным ключом;
2) user_id – внешний ключ к таблице User;
3) item_id – внешний ключ к таблице Item;
4) text – текст комментария пользователя;
5) reting – оценка товара пользователем;

6) delete_at – время удаления комментария;

7) delete_by – кем удален комментарий.

Таблица Order является компонентом базы данных и имеет следующее содержание:

1) id – это идентификатор, который является первичным ключом;

2) status – внешний ключ к таблице статус;

3) user_id – внешний ключ к таблице User.

Таблица Status является компонентом базы данных и имеет следующее содержание:

1) id – это идентификатор, который является первичным ключом;

2) name – соответственно название статуса, является создан, оплачен, в процессе, доставляется, доставлен, отменен.

Таблица OrderHistory является компонентом базы данных и имеет следующее содержание:

1) order_id – внешний ключ к таблице Order;

2) status – внешний ключ к таблице статус;

3) create_at – время изменения статуса заказа.

В базе данных используются связи один ко многим для таблиц: Role – User, User – Comment, User – Cart, Cart – Item, Item – ItemHistory, Status – Order, Status – OrderHistory, User – Order. Связь многое ко многим для таблиц: Order – Item.

2.3. Эндпоинты

User

«GET /user/register/» – запрос на получение формы регистрации пользователя, в которой «пользователь» должен будет ввести имя пользователя, пароль и email.

«POST /user/register/» – запрос на регистрацию «пользователя» на сайте, которая добавляет его в базу данных, и пользователь становится авторизованным.

«GET /user/login/» – запрос на получение формы авторизации «пользователя», в которой «пользователь» должен будет ввести имя пользователя и пароль.

«POST /user/login/» – запрос на авторизацию «пользователя» на сайте, которая находит его в базе данных, и «пользователь» становится авторизованным.

«GET /user/logout/» – запрос на выход из аккаунта «пользователя» и «пользователь» становится «неавторизованным пользователем».

Item

«GET /items/» – запрос на получение списка товаров. Товар представляет собой номер, название, цену, количество товаров и описание.

«POST /items/» – запрос на создание товара. Обязательны к заполнению поля: название, описание, цены. Этот запрос доступен только «администратору».

«GET /items/<item_id>/» – запрос на получения конкретного товара из списка, а также комментариев, оставленных «пользователем» для данного товара.

«DELETE /items/<item_id>/» – запрос на удаления конкретного товара. Этот запрос доступен только «администратору».

«PATCH /items/<item_id>/» – запрос на изменение конкретных данных товара. Этот запрос доступен только «администратору».

Comment

«GET /items/<item_id>/comments/» – запрос на получение списка всех комментариев «пользователей» для данного товара. Комментарии представляют собой: номер, имя пользователя, рейтинг и текст.

«POST /items/<item_id>/comments/» – запрос на создание комментария для данного товара. Этот запрос доступен только «авторизованному пользователю».

«GET /items/<item_id>/comments/<comment_id>/» – запрос на получение конкретного комментария.

«DELETE /items/<item_id>/comments/<comment_id>/» – запрос на удаления комментария «пользователя». Этот запрос доступен только «пользователю», который написал комментарий, а также «администратору».

Cart

«GET cart/» – запрос на получение списка товаров, добавленных в корзину. Это запрос доступен только «авторизованному пользователю».

«POST cart/» – запрос на добавление товара в корзину. Необходимо выбрать сам товар и его количество. Это запрос доступен только «авторизованному пользователю».

«PATCH cart/<item_id>/» – запрос на изменение количества выбранного товара в корзине. Это запрос доступен только «авторизованному пользователю».

«DELETE cart/<item_id>/» – запрос на удаления выбранного товара из корзины. Это запрос доступен только «авторизованному пользователю».

Order

«GET /orders/» – запрос на получения списка всех заказов пользователя. Представляет собой номер, статус заказа и список всех товаров, добавленных в заказ. Это запрос доступен только «авторизованному пользователю».

«POST /orders/» – запрос на создание заказа. Пользовательская корзина очищается, все товары добавляются в заказ, статус заказа становится «Создан». Это запрос доступен только «авторизованному пользователю».

«GET /orders/<order_id>/» – запрос на получение описания конкретного заказа. Это запрос доступен только «авторизованному пользователю», создавшему заказ.

«DELETE /orders/<order_id>/» – запрос на перевод статус заказа на «Отменен». Это запрос доступен только «авторизованному пользователю», создавшему заказ.

Статус ответа

Таблица 1 представляет собой описание статусов ответа сервера, используемых для коммуникации состояния обработки запроса клиентом. Каждый статус кодирует специфическое состояние, сигнализируя о результатах взаимодействия с сервером.

Таблица 1 – Описание статуса ответа сервера

Код	Сообщение	Описание
200	OK	Означает, что запрос успешно обработан, не выявлено ошибок
400	Invalid data	Означает, что данные, введенные в запросе, были некоренные
401	Unauthorized	Означает что «пользователь», который сделал запрос, был не авторизован, когда это было нужно
403	Forbidden	Означает что запрос, который был выполнен доступен только «администратору», когда его запросил «пользователь» без прав «администратора»
404	Not found	Означает, что данные, к которым «пользователь» запросил доступ не найдены
500	Server	Означает что на сервере произошла ошибка, которая была не обработана

Таким образом эти статусы играют ключевую роль в стандартизации взаимодействия между клиентом и сервером, обеспечивая ясность и предсказуемость в коммуникации, а также облегчая диагностику и устранение возникающих проблем.

3. РЕАЛИЗАЦИЯ

3.1. Бекэнд сервер

Создадим файл «requirements.txt» [11], в котором напишем нужные библиотеки и фреймворки. С помощью встроенного модуля Python [12] создадим виртуальное окружение для того, чтобы не было конфликтов версий с теми библиотеками, которые уже были установлены на устройстве. В этом виртуальном окружении с помощью системы управления пакетами Python установим все необходимые зависимости.

Создадим Django проект с названием «Diplom» и создадим 4 приложения под различные задачи: «dice» для товаров, «user» для пользователей, «cart» для корзины и «order» для заказов. Код создания проекта представлен в листинге 1.

Листинг 1 – Инициализация проекта

```
django-admin startproject diplom
cd diplom
python manage.py startapp dice
python manage.py startapp user
python manage.py startapp cart
python manage.py startapp order
```

Дальше подключим наши приложения к проекту, для этого необходимо добавить их в «setting.py» в «INSTALLED_APPS». Код добавления приложений и других фреймворков [12] показан в листинге 2.

Листинг 2 – Настройка проекта

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'cart',
    'dice',
    'user',
    'order',
    'rest_framework'
]
```

После настройки проекта настроим эндпоинты у проекта, таким образом, чтобы они указывали на наши приложения. Код представлен в листинге 3.

Листинг 3 – Эндпоинты проекта

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('user/', include('user.urls')),
    path('items/', include('dice.urls')),
    path('cart/', include('cart.urls')),
    path('orders/', include('order.urls'))
]
```

После этого создадим модели в «django ORM» в соответствии с базой данных. Код создания модели пользователя представлен в листинге 4.

Листинг 4 – Модель пользователя

```
class Role(models.TextChoices):
    AUTHORISED = 'AUTHORISED', 'Авторизованный'
    ADMINISTRATOR = 'ADMINISTRATOR', 'Администратор'

class User(AbstractUser):
    role = models.CharField(choices=Role.choices,
                            default=Role.AUTHORISED,
                            max_length=30
                            )
```

Создание моделей в приложении «dice» описано в листинге 5.

Листинг 5 – Модель товаров и комментариев

```
class Role(models.TextChoices):
    AUTHORISED = 'AUTHORISED', 'Авторизованный'
    ADMINISTRATOR = 'ADMINISTRATOR', 'Администратор'

class Item(models.Model):
    id = models.AutoField(primary_key=True, null=False, auto_created=True)
    name = models.CharField(max_length=100, unique=True, null=False)
    price = models.PositiveIntegerField(null=False)
    stock = models.PositiveIntegerField(null=False, default=0)
    description = models.TextField()

class ItemHistory(models.Model):
    dice = models.ForeignKey(Item, on_delete=models.DO_NOTHING)
    name = models.CharField(max_length=100, null=False)
    price = models.PositiveIntegerField(null=False)
    description = models.TextField()
    create_at = models.DateTimeField(auto_now=True)

class Comment(models.Model):
    id = models.IntegerField(primary_key=True, null=False, auto_cre-
ated=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    item = models.ForeignKey(Item, on_delete=models.CASCADE)
    text = models.TextField()
```

```
    rating = models.PositiveSmallIntegerField()
    delete_at = models.DateTimeField(null=True, default=None, blank=True)
```

Создание моделей в приложении «cart» описано в листинге 6.

Листинг 6 – Модель корзины

```
class Cart(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    item = models.ForeignKey(Item, on_delete=models.CASCADE)
    count = models.IntegerField(null=False, default=1)
```

Создание моделей в приложении «order» описано в листинге 7.

Листинг 7 – Модель заказа

```
class Status(models.TextChoices):
    CREATED = 'CREATED', 'Создан'
    PAID = 'PAID', 'Оплачен'
    PROCESSING = 'PROCESSING', 'В процессе'
    DELIVERY = 'DELIVERY', 'Доставляется'
    FINISHED = 'FINISHED', 'Доставлен'
    CANCELLED = 'CANCELLED', 'Отменен'

class Order(models.Model):
    id = models.AutoField(primary_key=True, null=False, auto_created=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    status = models.CharField(
        choices=Status.choices, default=Status.CREATED, max_length=30)

class ItemOrder(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    item = models.ForeignKey(Item, on_delete=models.CASCADE)
    count = models.IntegerField(default=1, null=False)

class OrderHistory(models.Model):
    order = models.ForeignKey(Order, on_delete=models.DO_NOTHING)
    status = models.CharField(choices=Status.choices, max_length=30)
    create_at = models.DateTimeField()
```

После создания моделей нужно провести миграции. Для этого с помощью встроенных инструментов Django [14] создадим миграцию в базе данных и выполним ее. Пример миграции для пользователя представлен в приложении листинг 1.

После миграций можно приступать к реализации представлений. Представления в контексте баз данных – это виртуальные таблицы, которые создаются на основе запросов к одной или нескольким реальным таблицам, они не хранят данные сами по себе, а показывают результат запроса, каждый раз, когда к ним обращаются.

Представления используются для упрощения доступа к данным, улучшения безопасности и повышения удобства работы с базой данных. Также они играют важную роль в современных системах управления базами данных. Их использование позволяет обеспечить эффективное, безопасное и удобное взаимодействие с данными.

В Django представления получают запрос и пользовательские данные, а на выходе выдают пользователю данные и статус код. Полученные пользовательские данные и данные из базы данных нужно привести к нужному формату, для этого используются сериализаторы, которые в одном случае проверяют валидность данных и возвращают объект модели, в другом случае делают из объектов модели json. Пример представления и сериализатора получения заказа представлен в листинге 8.

Листинг 8 – Представление и сериализатор для заказа

```
def get(self, request: Request, order_id: int) -> Response:
        if not request.user.is_authenticated:
            return Response(data={'error': 'User is unauthorized'}, sta-
tus=401)
        order = Order.objects.filter(id=order_id, user=re-
quest.user).first()
        if not order:
            return Response(data={'error': 'Not Found'}, status=404)
        order_serializer = GetOrdersSerializer(instance=order)
        return Response(data=order_serializer.data, status=200)

class GetOrdersSerializer(serializers.ModelSerializer):
    items = serializers.SerializerMethodField('get_items')

    class Meta:
        model = Order
        fields = ['id', 'status', 'items']

    def get_items(self, instance: Order):
        item_order = ItemOrder.objects.filter(order=instance.id)
        items_serializer = GetItemOrderSerializer(
            instance=item_order, many=True)
        return items_serializer.data
```

После написания представлений проведем небольшое тестирование функционала. Для этого создадим суперпользвоателя для нашей базы данных и зайдем с помощью него в админ панель. Там создадим несколько предметов и сделаем запрос на получение этих предметов. Пример создания товара представлен на рисунке 5.

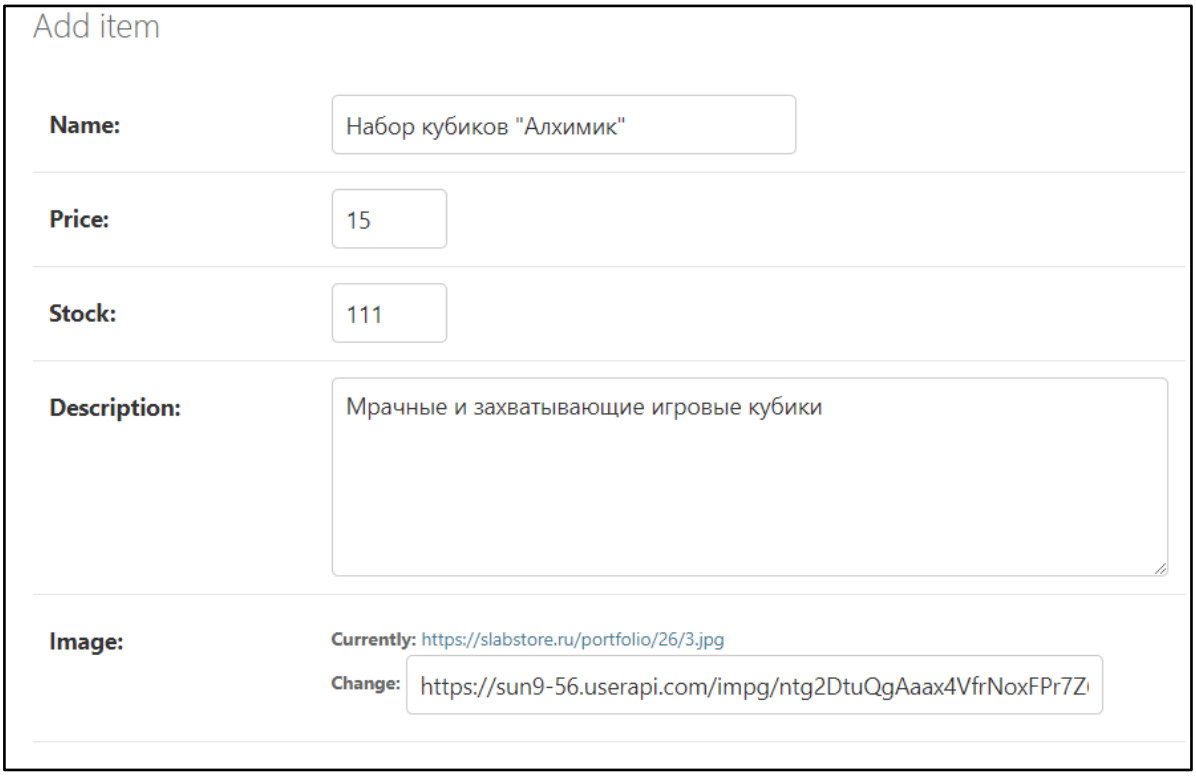

Рисунок 5 – Создание товара

Далее запустим сервер локально и сделаем запрос в браузере на получение списка всех предметов. Пример ответа представлен на рисунке 6.

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "name": "d20",
        "price": 3,
        "stock": 1,
        "description": "cool d20"
    },
    {
        "id": 2,
        "name": "d6",
        "price": 1,
        "stock": 5,
        "description": "cool d6"
    }
]
```

Рисунок 6 – Ответ сервера

3.2. Тестирование

Тестирование запросов к бекэнду представлено в таблице 2.

Таблица 2 – Тестирование успешных запросов бекэнд сервера

Тестируемый запрос	Входные данные	Выходные данные
POST /user/register/	username: sadasd password: sdasd email: emai@mail.com	200 OK
POST /user/login/	username: sadasd password: sdasd	200 OK
POST /items/	name: d20 price: 3 stock: 1 description: cool d20	200 OK
GET /items/	отсутствуют	200 OK id: 4 name: d20 price: 3 stock: 0 description: cool d20
GET /items/4/	отсутствуют	200 OK id: 4 name: d20 price: 3 stock: 0 description: cool d20
PATCH /items/4/	name: big d20	200 OK
POST cart/	item_id: 4 count: 1	200 OK

GET cart/	отсутствуют	200 OK
		item: {
		id: 4
		name: big d20
		price: 3
		}
		count: 1

POST /orders/	отсутствуют	200 OK

3.3. Фронтэнд

После того как закончилась разработка серверной части, началась разработка клиентской части. Интерфейс сайта для «неавторизованного пользователя» содержит меню товаров с карточками, регистрацию «пользователя» и авторизацию «пользователя». На рисунке 7 представлен интерфейс сайта.

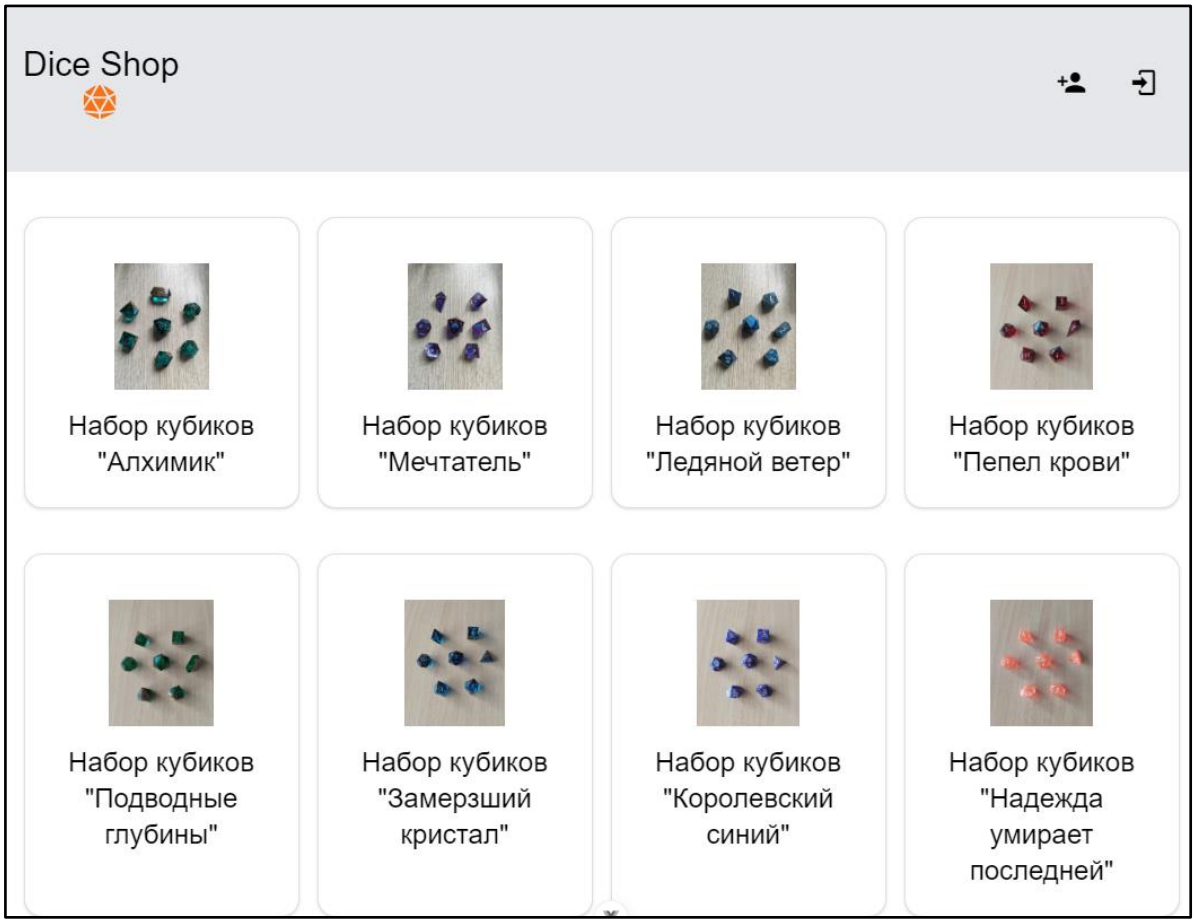

Рисунок 7 – Интерфейс сайта

В карточке товара представлено название товара, его описание, цена, количество в наличии, а также у данного товара присутствует комментарий «авторизованного пользователя» и рейтинг.

Рейтинг создается на основе отзывов «пользователей» и служит для удобства других «пользователей», чтобы можно было быстро оценить качество товара по шкале от 1 до 5, где 1 – самая низкая оценка, а 5 – самая высокая. Это позволяет пользователям тратить меньше времени на изучение разных комментариев и сразу ориентироваться на рейтинг.

Дополнительно карточка товара может содержать изображения продукта, позволяющие пользователям визуально оценить товар перед покупкой. Карточка товара представлена на рисунке 8.

Рисунок 8 – Карточка товара

Следующим важным этапом разработки стала реализация функциональных кнопок, предназначенных для «авторизованного пользователя».

В рамках этого этапа были добавлены такие ключевые элементы интерфейса, как кнопка корзины, которая позволяет пользователям просматривать и управлять выбранными товарами, кнопка списка заказов, предоставляющая доступ к истории покупок и текущим заказам, а также кнопка выхода из аккаунта, обеспечивающая выход из аккаунта. Все эти элементы наглядно представлены на рисунке 9.

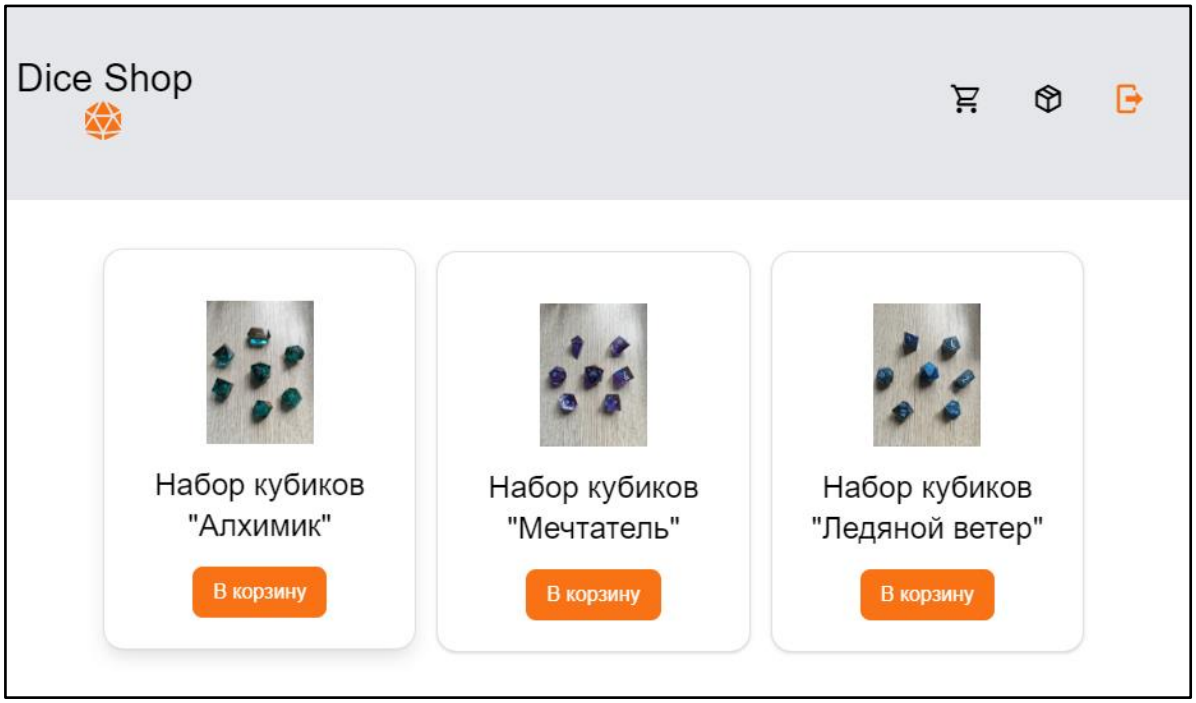

Рисунок 9 – Интерфейс веб сайта

Следующим шагом разработки стала реализация функциональных кнопок непосредственно в карточке товара. Для «авторизованного пользователя» была добавлена возможность выбирать нужное количество товара, что позволяет более точно управлять заказом. Это дает пользователям гибкость и точность в управлении своими заказами, что особенно важно при покупке товаров в больших количествах или при наличии различных вариантов товаров.

Кроме того, была реализована функция оценки товара, что дает пользователям возможность оставлять отзывы и делиться своим мнением о товаре, который пользователи купили. Оценки и отзывы способствуют созданию доверительной атмосферы на сайте, потенциальные покупатели могут опираться на реальные мнения и оценки других пользователей.

Все функции и элементы интерфейса карточки товара представлены на рисунке 10.

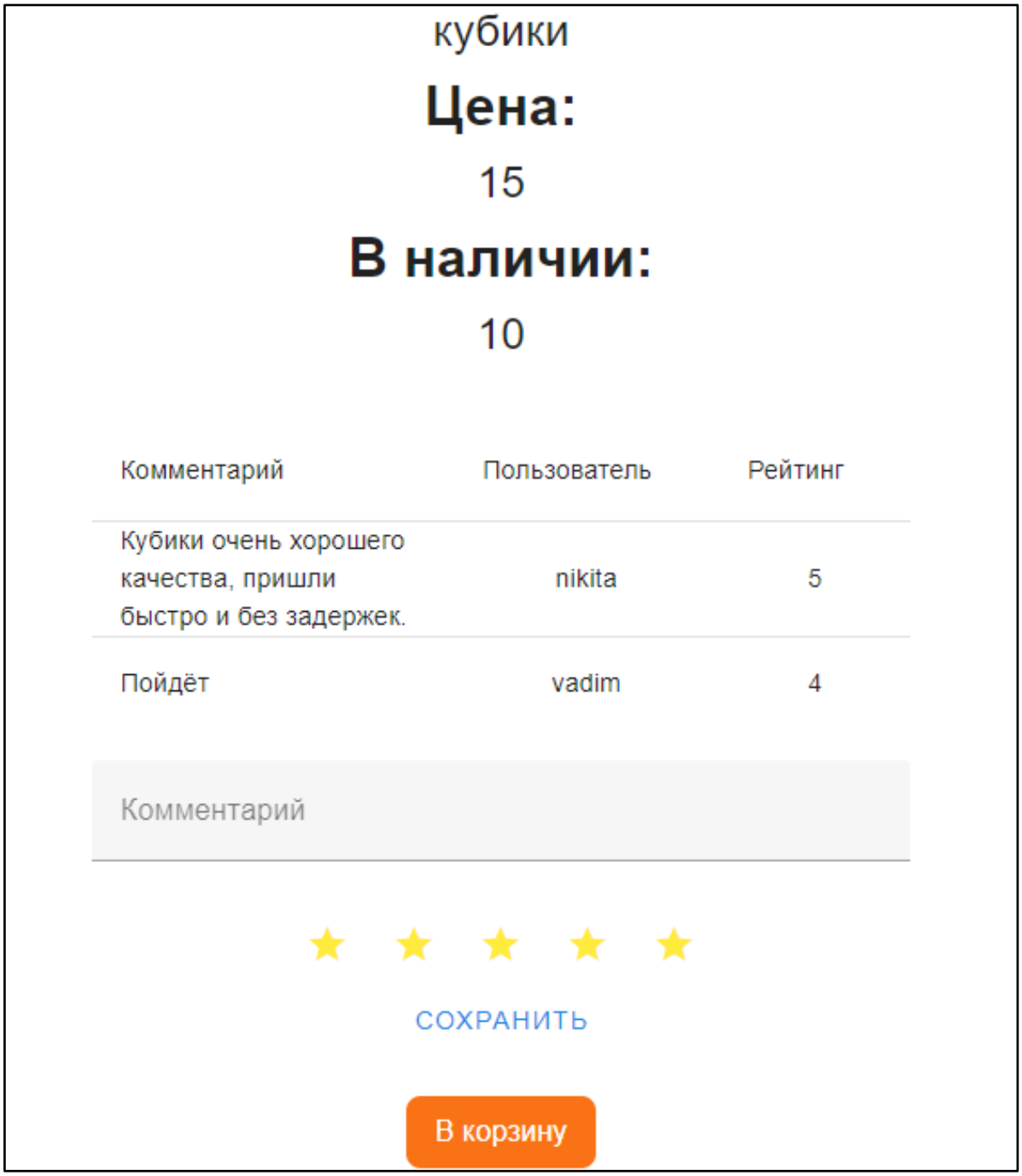

Рисунок 10 – Карточка товара

Теперь рассмотрим корзину заказов, доступную для «авторизованного пользователя». В корзине «авторизованный пользователь» имеет возможность изменять количество единиц товаров, выбранных для заказа. Помимо этого, пользователь может удалять товары из корзины, просматривать стоимость каждого товара и видеть общую сумму стоимости всех товаров.

После внесения всех необходимых изменений и проверок, пользователь может оформить заказ. Все функции корзины заказов наглядно представлены на рисунке 11.

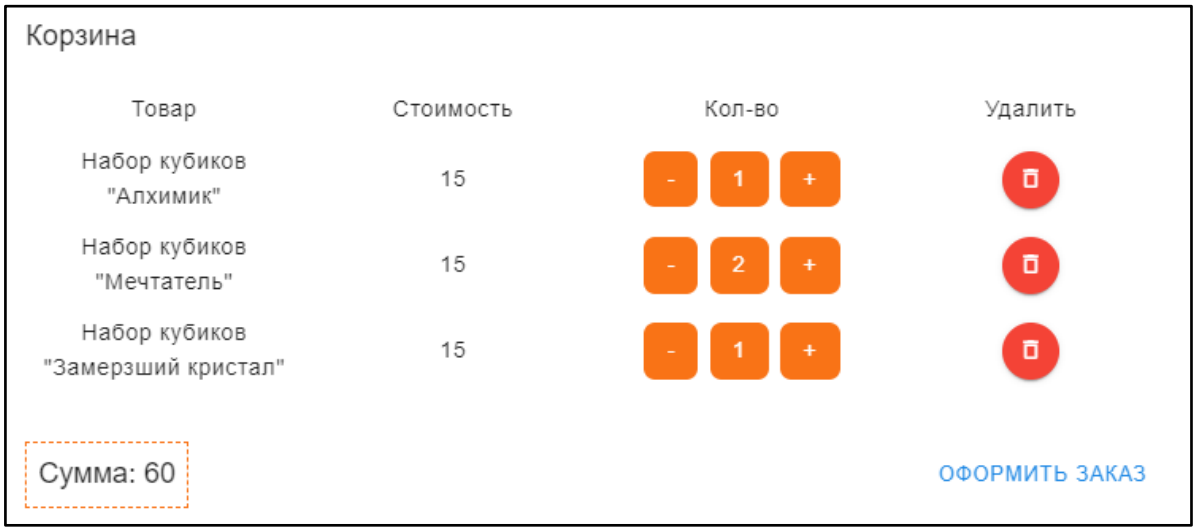

Рисунок 11 – Корзина заказов

После этого перейдем к разделу заказов. В этом разделе «авторизованный пользователь» может увидеть статус своего заказа, а также другую важную информацию, связанную с ним. В частности, отображаются номер заказа, список товаров, включенных в заказ, стоимость каждого товара и общее количество товаров.

Все эти данные помогают пользователю легко отслеживать и управлять своими покупками. Представленные элементы интерфейса наглядно показаны на рисунке 12.

Рисунок 12 – Список заказов

Далее перейдем к роли «администратора» и рассмотрим интерфейс веб-сайта, который доступен исключительно ему. «Администратор» обладает расширенными возможностями, включая редактирование и создание новых карточек товаров. Также у него есть доступ к кнопке выхода из аккаунта. Эти функции позволяют администратору эффективно управлять контентом сайта и поддерживать его актуальность. Все элементы интерфейса, доступные администратору, наглядно представлены на рисунке 13, что демонстрирует их функциональность и расположение.

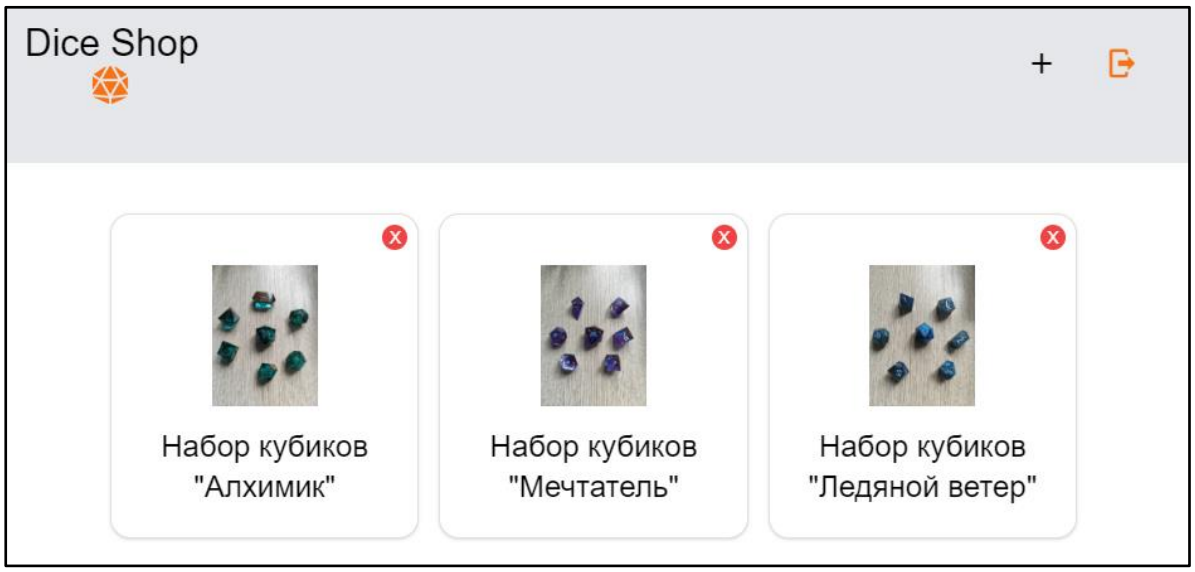

Рисунок 13 – Интерфейс веб сайта

В заключении данной главы можно отметить, что были успешно реализованы и описаны ключевые элементы интерфейса. Для авторизованных пользователей внедрены функциональные кнопки в карточке товара, корзине заказов и разделе заказов, что обеспечивает удобство и гибкость в процессе покупок.

Для администраторов создан специализированный интерфейс, позволяющий редактировать и добавлять новые товары. Таким образом, внедрение этих элементов интерфейса способствует повышению удобства и эффективности взаимодействия с веб-сайтом для всех категорий пользователей.

4. ТЕСТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ

Важный шаг после завершения верстки страниц и реализации функциональности веб-сайта, является проведение функционального тестирования [15] разработанного веб-сайта.

Тестирование проводилось в два этапа:

1) функциональное тестирование;
2) usability-тестирование.

4.1. Функциональное и Usability тестирование

Функциональное тестирование направлено на проверку правильной работы функциональности приложения и соответствие выполнению функциональных требований. В таблице 3 приведены результаты функционального тестирования. Результаты тестирования совпадают с ожидаемыми результатами. Это свидетельствует о правильной реализации функциональных требований.

Таблица 3 – Функциональное тестирование

№	Название тестирования	Действие	Полученный результат	Результат тестирования
1	Открыт веб-сайт	Пользователь видит весь список товаров	Список товаров отобразился корректно	Пройдено
2	Регистрация пользователя	Пользователь регистрируется в систему	Пользователь успешно зарегистрирован	Пройдено
3	Авторизация пользователя	Пользователь авторизуется в систему	Пользователь успешно авторизован	Пройдено
4	Открыта карточка товара	Пользователь нажимает на карточку нужного ему товара	Открыто модальное окно с описанием товара	Пройдено
5	Добавить товар в корзину	Пользователь добавляет товар в корзину	Товар, выбранный пользователем добавлен в корзину	Пройдено
6	Редактирование корзины	Пользователь редактирует корзину	Количество товаров в корзине изменяется	Пройдено
7	Оформление заказа	Пользователь в разделе «корзина» оформляет заказ	Заказ пользователя создан	Пройдено

Окончание таблицы 3

№	Название тестирования	Действие	Полученный результат	Результат тестирования
8	Отмена заказа	Пользователь открывает раздел «Заказ» и отменяет его	Заказ пользователя отменен	Пройдено
9	Создание карточки товара	Администратор создает карточку товара	Карточка товара появляется на главной странице сайта	Пройдено
10	Удаление карточки товара	Администратор удаляет карточку товара	Карточка товара удаляется с главной страницы сайта	Пройдено
11	Создание комментария	Пользователь может добавить комментарий к товару и оценить его	Комментарий добавлен к товару	Пройдено
12	Удаление комментария	Администратор может удалить выбранный комментарий пользователя	Комментарий, который был добавлен к товару удален	Пройдено

Usability-тестирование

В ходе usability-тестирования была оценена удобность и эффективность использования системы. В тестировании участвовали три человека, им были даны конкретные задачи. Задачи включали в себя регистрацию и авторизацию в системе, создание товара, добавление комментариев, удаление комментариев и удаление товара, добавление товара в корзину, изменения количества товаров в корзине, создание заказа и отмена заказа.

Тестировщики успешно выполнили все предложенные им задачи без затруднений и неудобств. Данная система успешно прошла usability-тестирование и ее использование является эффективным и удобным.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы был разработан веб-сайт магазина по продаже настольно-игровых кубиков, с использованием Django для back-end и Vue.js для front-end.

В процессе разработки была реализована функциональность, позволяющая пользователям выбирать и заказывать товары. Был проведен анализ пользовательских потребностей, определены требования к сайту, разработана архитектура приложения и реализовано его программное обеспечение. Особое внимание было уделено использованию Vue.js для создания пользовательского интерфейса, что позволило сделать сайт более удобным, функциональным и привлекательным для пользователей.

Результаты работы будут использованы в дальнейшем для улучшения работы интернет-магазина и его дизайна.

В ходе проделанной работы были решены следующие задачи:

1) выполнен анализ предметной области, включая анализ аналогичных веб-сайтов с выявлением требований;
2) произведено проектирование веб-сайта;
3) реализован веб-сайт;
4) протестирован веб-сайт.

ЛИТЕРАТУРА

1. Duckett J. HTML and CSS: Design and Build Websites. // John Wiley & Sons, Inc, 2011. – 56–113 pp.

2. Die hard dice. [Электронный ресурс] URL: https://www.dieharddice.com/ (дата обращения: 16.02.2024 г.).

3. Dice Dungeon. [Электронный ресурс] URL: https://thedicedungeon.co.uk/ (дата обращения: 16.02.2024 г.).

4. Flask. [Электронный ресурс] URL: https://flask.palletsprojects.com/en/3.0.x/ (дата обращения: 16.02.2024 г.).

5. FastApi. [Электронный ресурс] URL: https://fastapi.tiangolo.com/ru/ (дата обращения: 16.02.2024 г.).

6. Django. [Электронный ресурс] URL: https://www.djangoproject.com/ (дата обращения: 16.02.2024 г.).

7. React. [Электронный ресурс] URL: https://ru.legacy.reactjs.org/ (дата обращения: 16.02.2024 г.).

8. Angular. [Электронный ресурс] URL: https://angular.io/ (дата обращения: 16.02.2024 г.).

9. VueJs. [Электронный ресурс] URL: https://ru.vuejs.org/index.html (дата обращения: 16.02.2024 г.).

10. Крэг Л. Применение UML и шаблонов проектирования. // Диалектика, 2019. – С. 73–95.

11. Мартин Р. Чистый код: создание, анализ и рефакторинг. // Питер, 2013. – 464 с.

12. Python [Электронный ресурс] URL: https://www.python.org/doc/ (дата обращения: 07.03.2024 г.).

13. Django REST framework. [Электронный ресурс] URL: https://www.django-rest-framework.org/topics/documenting-your-api// (дата обращения: 16.04.2024 г.).

14. Django REST framework. [Электронный ресурс] URL: https://www.django-rest-framework.org/ (дата обращения: 17.04.2024 г.).

15. Kaner C., Bach J. Lessons Learned in Software Testing: A Context-Driven. // Wiley, 2001. – 170–251 pp.

ПРИЛОЖЕНИЕ. Фрагменты исходного кода

Листинг 1 – Пример миграции пользователя

```
class Migration(migrations.Migration):

    initial = True

    dependencies = [
        ('auth', '0012_alter_user_first_name_max_length'),
        ('dice', '0001_initial'),
    ]

    operations = [
        migrations.CreateModel(
            name='User',
            fields=[
                ('id', models.BigAutoField(auto_created=True, pri-
mary_key=True, serialize=False, verbose_name='ID')),
                ('password', models.CharField(max_length=128, ver-
bose_name='password')),
                ('last_login', models.DateTimeField(blank=True, null=True,
verbose_name='last login')),
                ('is_superuser', models.BooleanField(default=False,
help_text='Designates that this user has all permissions without explic-
itly assigning them.', verbose_name='superuser status')),
                ('username', models.CharField(error_messages={'unique': 'A
user with that username already exists.'}, help_text='Required. 150 char-
acters or fewer. Letters, digits and @/./+/-/_ only.', max_length=150,
unique=True, validators=[django.contrib.auth.validators.Unicode-
UsernameValidator()], verbose_name='username')),
                ('first_name', models.CharField(blank=True,
max_length=150, verbose_name='first name')),
                ('last_name', models.CharField(blank=True, max_length=150,
verbose_name='last name')),
                ('email', models.EmailField(blank=True, max_length=254,
verbose_name='email address')),
                ('is_staff', models.BooleanField(default=False,
help_text='Designates whether the user can log into this admin site.',
verbose_name='staff status')),
                ('is_active', models.BooleanField(default=True,
help_text='Designates whether this user should be treated as active. Unse-
lect this instead of deleting accounts.', verbose_name='active')),
                ('date_joined', models.DateTimeField(de-
fault=django.utils.timezone.now, verbose_name='date joined')),
                ('role', models.CharField(choices=[('AUTHORISED',
'Авторизованный'), ('ADMINISTRATOR', 'Администратор')], default='AUTHOR-
ISED', max_length=30)),
                ('groups', models.ManyToManyField(blank=True,
help_text='The groups this user belongs to. A user will get all permis-
sions granted to each of their groups.', related_name='user_set', re-
lated_query_name='user', to='auth.group', verbose_name='groups')),
                ('user_permissions', models.ManyToManyField(blank=True,
help_text='Specific permissions for this user.', related_name='user_set',
related_query_name='user', to='auth.permission', verbose_name='user per-
missions')),
            ],
            options={
                'verbose_name': 'user',
                'verbose_name_plural': 'users',
                'abstract': False,
            },
            managers=[
```

```
                ('objects', django.contrib.auth.models.UserManager()),
            ],
        ),
        migrations.CreateModel(
            name='Comment',
            fields=[
                ('id', models.IntegerField(auto_created=True, pri-
mary_key=True, serialize=False)),
                ('text', models.TextField()),
                ('rating', models.PositiveSmallIntegerField()),
                ('delete_at', models.DateTimeField(default=None, null=True
        )
```