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-приложения для байерского агентства

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-316.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ Л.Н. Петрова

Автор работы,
студент группы КЭ-401
_____ Д.А. Трифонова

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
студентке группы КЭ-401
Трифоновой Дарье Алексеевне,
обучающейся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)

Разработка веб-приложения для байерского агентства.

2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.

3. **Исходные данные к работе**

3.1. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, HTML5. 5-е издание. – СПб: ООО «Диалектика», 2019. – 816 с.

3.2. Официальный сайт фреймворка Django. [Электронный ресурс] URL: https://www.djangoproject.com (дата обращения: 05.02.2024 г.).

3.3. Роббинс Дж. HTML5, CSS3 и Javascript. Исчерпывающее руководство. – М.: Эксмо, 2014. – 528 с.

3.4. PyCharm: IDE для профессиональной разработки на Python от JetBrains. [Электронный ресурс] URL: https://www.jetbrains.com/ru/pycharm/ (дата обращения: 05.02.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Определить требования к разрабатываемому веб-приложению.

4.2. Спроектировать интерфейс веб-приложения.

4.3. Разработать клиентскую и серверную части веб-приложения.

4.4. Обеспечение безопасности и защиты данных веб-приложения.

4.5. Произвести тестирование и отладку веб-приложения.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП Л.Н. Петрова

Задание принял к исполнению Д.А. Трифонова

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Сейчас веб-приложения используются во многих сферах, от онлайн-магазинов и социальных сетей до банковских систем и образовательных платформ. В современном мире веб-приложения являются неотъемлемой частью бизнеса и общественной жизни.

Более того, с развитием технологий и распространением высокоскоростных интернет-соединений, веб-приложения становятся более мощными и функциональными, что открывает новые возможности для создания сложных приложений с высокой производительностью и удобным пользовательским интерфейсом.

В это же время растет и популярность услуг байеров. Байер (от англ. buy – покупать) – это торговый агент, специалист по закупкам и торговым операциям на рынке брендовых товаров в зарубежных странах. На фоне политической обстановки в России закрылись магазины популярных иностранных производителей. На данный момент далеко не все почитатели ушедших брендов нашли им замену. Поэтому в поисках привычных товаров потребители стали все чаще обращаться к байерам.

Главными площадками для работы у байеров являются соцсети Instagram, который признан в РФ экстремистским и запрещен, а также мессенджер Telegram. Однако ограниченный функционал мессенджера – структура и навигация довольно базовые, накладывает свои особенности на ведение в нем бизнес-сообщества, в первую очередь, ввиду отсутствия единой ленты, что существенно снижает восприятие информации для пользователя, а следовательно, и его заинтересованность.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-приложения для байерского агентства.

Для достижения поставленной цели необходимо решить следующие задачи.

1. Определить требования к разрабатываемому веб-приложению.
2. Спроектировать интерфейс веб-приложения.
3. Разработать клиентскую и серверную части веб-приложения.
4. Произвести тестирование и отладку веб-приложения.

Структура и содержание работы

Работа состоит из введения, пяти глав, заключения и списка литературы. Объем работы составляет 38 страниц, объем списка литературы – 15 источников.

В первой главе осуществляется обзор аналогичных приложений, а также обзор средств разработки веб-приложений.

Во второй главе описываются требования к системе, актеры и варианты использования системы. Также приводится графический интерфейс пользователя.

В третьей главе представлена архитектура системы и ее описание.

В четвертой представлены результаты реализации веб-приложения.

В пятой главе приведены результаты тестирования веб-приложения.

В заключении представлены результаты работы и направления дальнейшего развития.

В приложениях А и Б содержатся спецификация и диаграмма вариантов использования.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Сравнительный анализ аналогов

Для того, чтобы создать привлекающее и конкурентноспособное веб-приложение, необходимо проанализировать аналогичные приложения в данной области, а также выявить их достоинства и недостатки.

В результате поиска аналогов было найдено сравнительно небольшое количество веб-приложений, что является преимуществом в виде относительно малой конкуренции. Рассмотрим некоторые из них.

WONDERLAND

Байерское агентство «WONDERLAND» [1] предоставляет услуги как для организации оптовых закупок предпринимателю для его бутика, так и для выкупа конкретных единиц товара частному лицу. Среди достоинств данного приложения можно выделить наличие справочной информации для начинающих предпринимателей о расчете прибыли в будущем бутике. Еще одним значимым преимуществом является периодическое обновление коллекций и предоставление списков предварительных заказов, что может помочь пользователям быстрее находить интересующие их товары и упростить процесс покупки. На рисунке 1 представлены скриншоты веб-приложения «WONDERLAND».

У веб-приложения также имеются и недостатки. В частности, список брендов на сайте ограничен только итальянскими брендами, что может быть неудобно для пользователей, которые ищут товары других производителей. Также, на сайте отсутствует форма для обратной связи, что затрудняет коммуникацию между пользователями и администрацией сайта. Однако самым главным недостатком веб-приложения является отсутствие быстрой формы заказа. Это может создавать неудобства для пользователей, и стать причиной ухода клиентов к конкурентам.

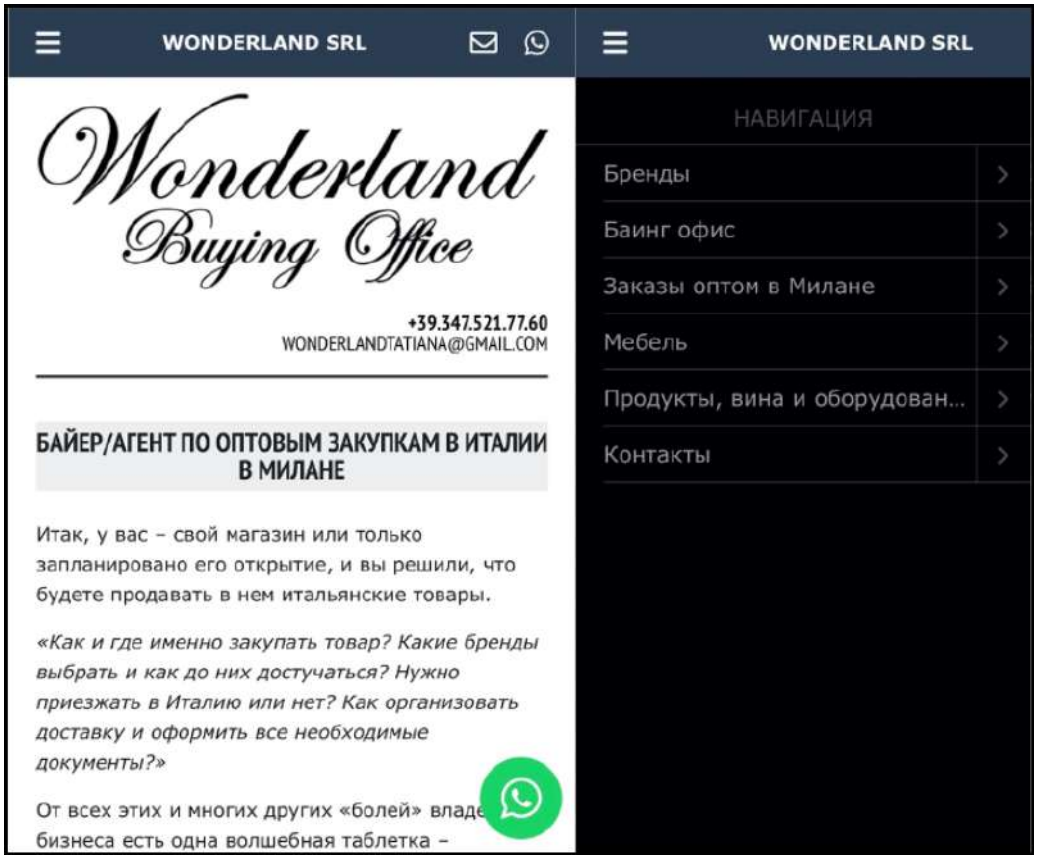

Рисунок 1 – Скриншоты веб-приложения «WONDERLAND»

TATI ITALY GROUP

Байерское агентство «TATI ITALY GROUP» [2] является оптовым поставщиком итальянской одежды от разных производителей и брендов для бизнеса. На веб-сайте каждый раздел меню оснащен подробной информацией, в частности, представлены видео-обзоры рабочего процесса байера и актуального ассортимента товаров некоторых бутиков. На рисунке 2 представлен скриншот веб-сайта «TATI ITALY GROUP».

Еще одним из плюсов данного приложения является наличие чата с живым менеджером, который помогает пользователям получить быстрый ответ на свой вопрос в режиме реального времени. Также на сайте присутствует информативный раздел FAQ, где пользователи могут найти ответы на наиболее часто задаваемые вопросы. К недостаткам приложения стоит отнести узкий ассортимент товаров для выкупа. Он предлагает только товары из категории одежды и обуви, что может ограничить возможности

пользователей и привести к тому, что они будут искать нужный товар на других сайтах.

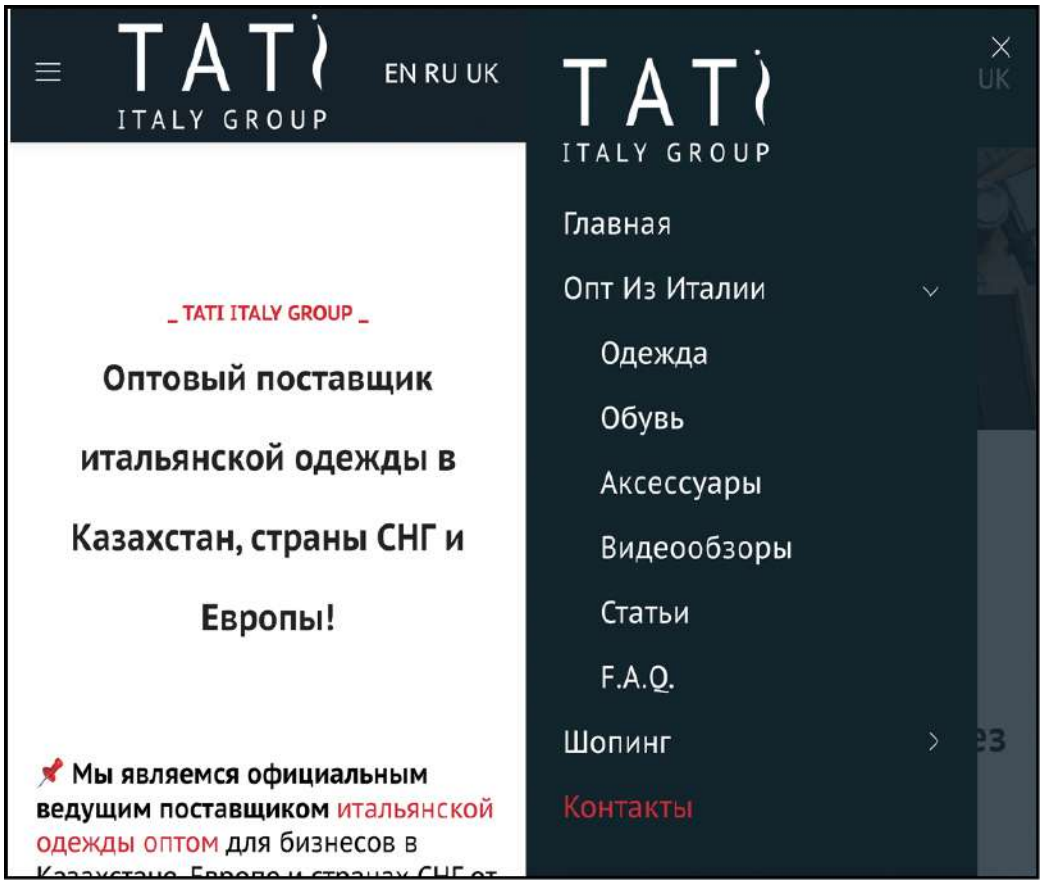

Рисунок 2 – Скриншоты веб-приложения «TATI ITALY GROUP»

Также одним из минусов веб-приложения является осуществление только оптовых закупок, что, несомненно, может быть неудобно для потенциальных клиентов, которые заинтересованы в розничных покупках. Еще значительными недостатками этого веб-приложения являются выбор только итальянских брендов, также как и в предыдущем примере, что может ограничивать возможности пользователей при поиске нужного товара, и отсутствие формы быстрого оформления заказа.

Таким образом, рассмотренные приложения достаточно похожи. Главным недостатком обоих веб-сайтов является отсутствие формы заказа, что может затруднять процесс осуществления быстрой покупки для пользователей. В связи с этим, важно разработать функционал быстрого

оформления заказов, который может стать эффективным инструментом удержания и привлечения новых пользователей.

1.2. Анализ существующих решений для реализации проекта

Создание программного обеспечения включает в себя множество задач, таких как проектирование, кодирование и тестирование [3]. Программные фреймворки облегчают жизнь разработчикам веб-сайтов и приложений, позволяя им контролировать процесс разработки программного обеспечения с помощью единой платформы.

Для создания веб-приложения был выбран Django – бесплатный и свободный фреймворк, написанный на Python. В основе архитектуры лежит паттерн проектирования MVC. Приложения в Django [4] позволяют разделить проект на несколько частей. Этот подход значительно облегчает интеграцию готовых решений. Еще одним из преимуществ фреймворка является административный интерфейс создания, чтения, обновления и удаления, динамически генерирующийся с помощью самодиагностики. Также с помощью такого инструмента как ORM можно взаимодействовать с базами данных, используя высокоуровневые методы Python, а не SQL-запросы, что автоматически снижает риск появления SQL-injection уязвимости.

Django использует SQLite в качестве системы управления базами данных. SQLite позволяет приложениям отправлять прямые запросы в файл, в котором хранятся данные. Благодаря этому SQLite имеет высокую скорость работы и быстрый обмен данными.

В качестве среды разработки был выбран редактор исходного кода PyCharm [5], который позволяет быстро производить рефакторинг кода, а также использовать удобный графический отладчик.

С помощью трех базовых технологий: HTML, CSS и JavaScript будет реализована верстка веб-приложения [6].

HTML – язык гипертекстовой разметки, который представляет правила оформления набора структурных и семантических элементов разметки – тегов. Он также предоставляет средства для создания форм для ввода данных пользователем, гиперссылок для навигации между страницами и другие возможности, необходимые для создания веб-содержимого.

CSS – язык формирования внешнего вида элементов HTML. CSS-стили отвечают за эстетическую составляющую параметра отображения структуры сайта. CSS также поддерживает адаптивную верстку, что позволяет создавать веб-страницы, которые корректно отображаются на различных устройствах с разными размерами экранов [7].

JavaScript – это высокоуровневый язык программирования, который применяется для создания интерактивных элементов и функционала на веб-страницах. JavaScript широко используется для добавления динамических возможностей на веб-сайты [8], таких как анимации, обработка событий, валидация форм, обновление содержимого без перезагрузки страниц, передача данных на сервер [9].

Также для использования некоторых блоков навигации, оформления веб-форм и других компонентов веб-интерфейса будет использоваться открытый и бесплатный фреймворк – Bootstrap.

Вывод по первой главе

В ходе обзора предметной области были рассмотрены аналогичные веб-приложения, выявлены их достоинства и недостатки, которые помогут в разработке собственного приложения. Также был описан выбор инструментов для его реализации.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1. Определение требований

В результате анализа предметной области и обзора существующих решений были сформированы следующие функциональные требования:

1) система должна обеспечивать пользователю возможность зарегистрироваться на сайте;

2) система должна обеспечивать пользователю возможность авторизоваться на сайте;

3) система должна обеспечивать пользователю возможность оформить заказ;

4) система должна обеспечивать пользователю возможность оставить отзыв об агентстве;

5) система должна обеспечивать пользователю возможность просматривать список брендов;

6) система должна обеспечивать пользователю возможность просматривать новостной блог агентства;

7) система должна обеспечивать возможность администратору модерировать отзывы;

8) система должна обеспечивать возможность администратору просматривать информацию о поступивших заказах;

9) система должна обеспечивать возможность администратору редактировать новостной блог агентства.

Также были сформированы нефункциональные требования:

1) система должна быть реализована на языке программирования Python 3;

2) система должна обновлять информацию при добавлении или изменении данных в СУБД;

3) система должна быть реализована с помощью технологий HTML, CSS и JavaScript.

2.2. Варианты использования системы

Для проектирования веб-приложения был использован язык графического описания для объектного моделирования UML [10]. На рисунке 1 приложения Б представлена диаграмма вариантов использования веб-приложения для байерского агентства. С веб-приложением взаимодействует четыре актера: «Авторизованный пользователь», «Неавторизованный пользователь» и «Администратор».

«Неавторизированный пользователь» может посмотреть новостной блог агентства, при желании может оставить комментарий под любой статьей блога. Также у него есть возможность посмотреть список брендов, доступных для заказа. Если пользователь готов совершить покупку – он может оформить заказ, указав ссылки на понравившиеся ему товары с официальных сайтов доступных брендов, но для этого ему необходимо зарегистрироваться на сайте. Если пользователь заинтересован стать байером, то он может заполнить форму с информацией о себе, дополнительно ответив в ней на вопросы, касающиеся его прошлого опыта в данной сфере деятельности.

«Администратор» может посмотреть заказ, а также изменить его статус. Также у администратора есть возможность отредактировать новостной блог агентства. Например, изменить описание статьи. «Администратор» может посмотреть комментарий пользователя, и, если он содержит нецензурные выражения – удалить комментарий.

2.3. Графический интерфейс пользователя

Графический интерфейс пользователя – система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.).

Ниже представлены основные страницы разрабатываемого приложения. Для адаптивности конечного продукта и верстки страниц использовались HTML и CSS [11].

«Главная страница» – страница, которая будет появляться при запуске веб-приложения (рисунок 4). На ней отображаются основные разделы веб-приложения, такие как «Меню» с разделами: «Схема оформления заказов», «Бренды», «Новостной блог», «Оставить отзыв», «Зарегистрироваться», «Войти в аккаунт», блок с контактной информацией агентства (рисунок 5) и блок с формой для сотрудничества с полями: «Ваше ФИО», «Ваш E-mail», «Страна проживания», «Ваш контактный номер телефона», «Ваши предложения». Для проверки корректности ввода информации были использованы регулярные выражения.

Рисунок 4 – Главная страница веб-приложения

В блоке с контактной информацией об агентстве у пользователя есть возможность направить любой свой вопрос напрямую поддержке сайта. Для проверки корректности ввода информации были использованы регулярные выражения.

Рисунок 5 – Блок с контактной информацией об агентстве

«Оформление заказа» – страница, которая будет появляться при нажатии на кнопку «Отправить» в форме первого этапа оформления заказа. На ней представлена основная форма для осуществления заказа (рисунок 6).

При желании пользователя можно добавить в один заказ несколько позиций товаров, используя новую форму для ввода данных при нажатии на кнопку «Еще». Также предусмотрена возможность очищения полей, если пользователь передумал оформлять определенный товар. В случае, если пользователь ввел некорректную ссылку в поле «Ссылка на понравившийся товар» – выведется сообщение об ошибке ввода с указанием верного формата данных.

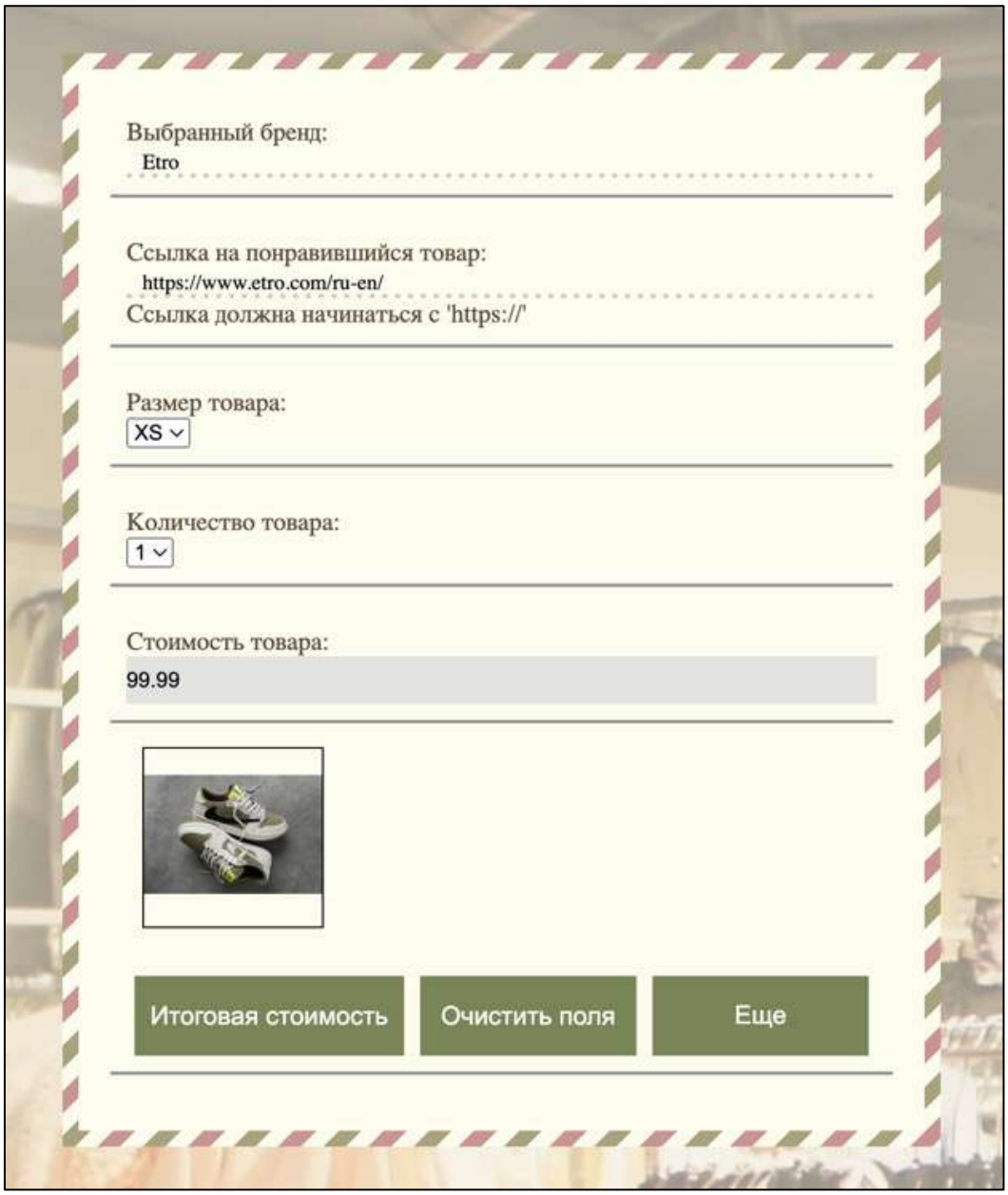

Рисунок 6 – Форма «Оформление заказа»

При нажатии на кнопку «Итоговая стоимость» в отдельном окне будет отображена общая сумма всех товаров с учетом процента выкупа, варьирующегося в зависимости от определенного ценового диапазона, для обеспечения ясности о заказе пользователю.

«Развернутая статья» – страница, появляющаяся при нажатии на заголовок выбранной статьи из раздела «Новостной блог». На данной странице, отображенной на рисунке 7, пользователям предоставляется ряд дополнительных возможностей.

Во-первых, пользователи могут оставлять комментарии под опубликованной статьей. Это позволяет им выражать свое мнение по теме. Во-вторых, пользователи могут продемонстрировать свое отношение к комментариям других пользователей, нажимая на кнопки «Нравится» или «Не нравится». Данная функциональность реализована с учетом принципа взаимоисключения: при нажатии на одну из кнопок, активность другой снимается, а их счетчики соответствующим образом обновляются.

Рисунок 7 – Страница «Развернутая статья»

«Авторизация пользователя» – страница, содержащая форму для входа уже зарегистрированного пользователя в систему (рисунок 8).

При заполнении формы авторизации производится проверка введенных данных. В случае, если пользователь забудет свои учетные данные, ему будет предоставлена возможность восстановить доступ к своему акка-

унту с помощью электронной почты, указанной ранее при регистрации. Для этого на странице предусмотрена дополнительная форма, в которую пользователь должен ввести свой адрес электронной почты.

После успешной авторизации пользователь получает доступ к своему личному кабинету.

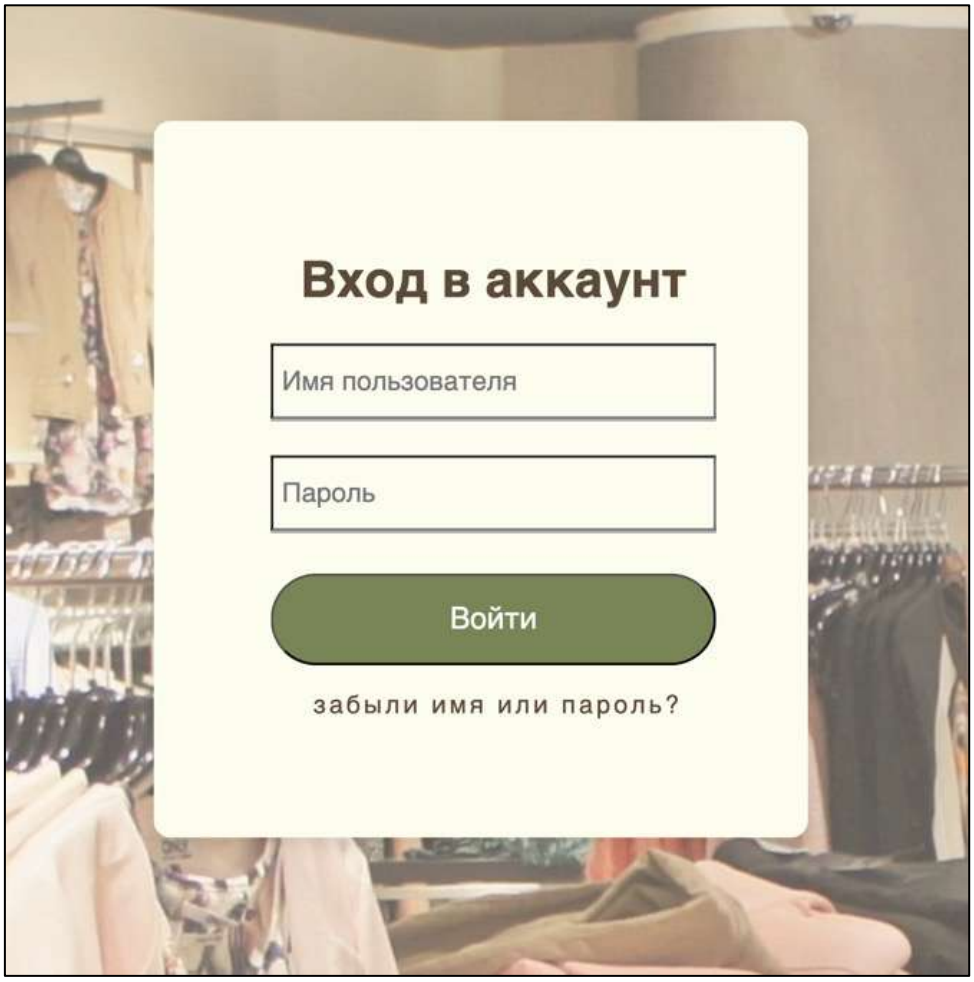

Рисунок 8 – Форма «Авторизация пользователя»

Вывод по второй главе

В данной главе были определены функциональные и нефункциональные требования к системе, описаны варианты использования системы, а также представлен графический интерфейс веб-приложения.

3. АРХИТЕКТУРА СИСТЕМЫ

3.1. Архитектура MTV

Веб-приложение построено на базе архитектуры MTV (Model-Template-View) [12].

MTV состоит из трех компонентов: Model, Template и View. Model представляет данные приложения и бизнес-логику, Template отображает данные на странице, а View обрабатывает HTTP-запросы от пользователей и связывает Model и Template вместе.

На рисунке 9 представлен основной принцип работы архитектуры MTV, который заключается в разделении логики приложения и отображения данных. Каждый компонент отвечает только за свою задачу, что делает код более читабельным и легко поддерживаемым.

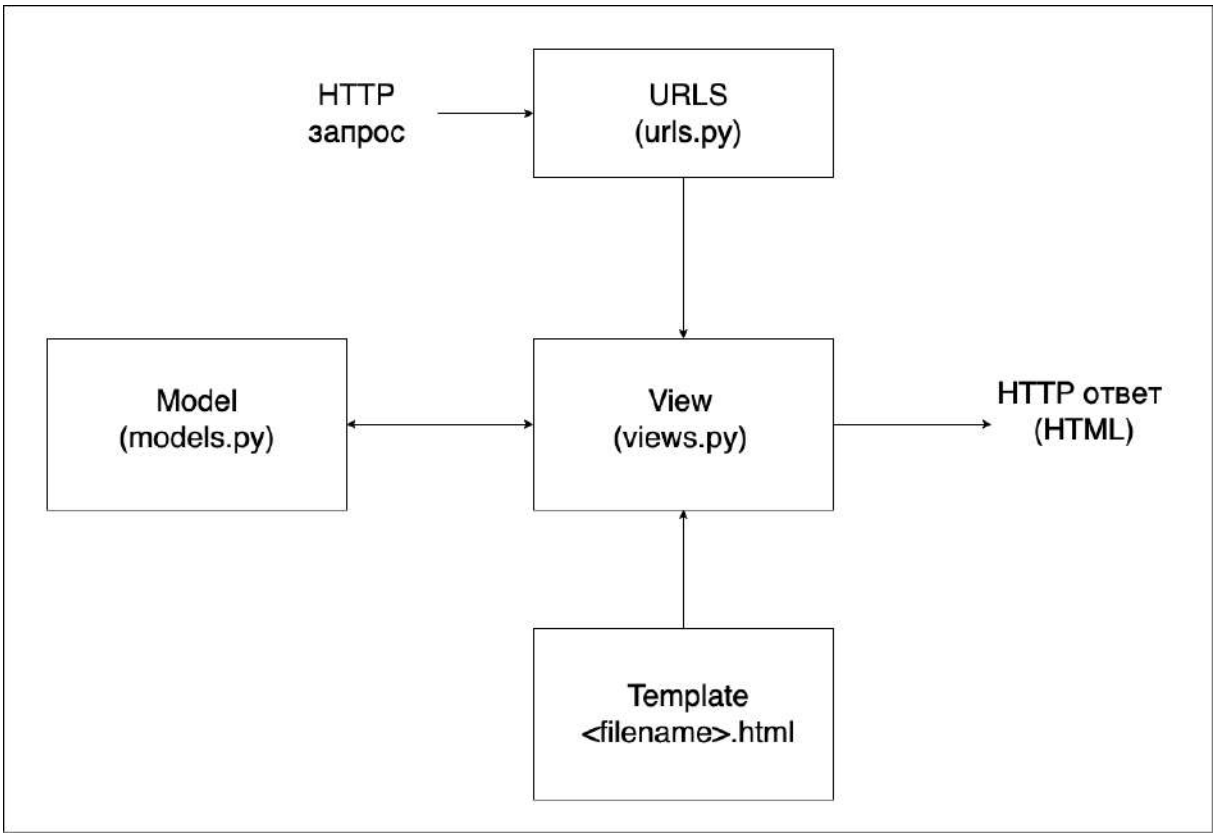

Рисунок 9 – Принцип работы архитектуры MTV

Преимущества использования архитектуры MTV в Django включают готовое решение для выполнения основных задач, таких как аутентификация, администрирование и обработка форм, а также возможность быстрой

разработки приложения благодаря широкому выбору расширений и библиотек, созданных комьюнити.

3.2. Диаграмма деятельности

Согласно требованиям к системе была разработана для прецедента «Добавить комментарий» диаграмма деятельности, представленная на рисунке 10. Предусловием данной диаграммы является то, что пользователь уже перешел на страницу со статьей из раздела «Новостной блог».

Диаграмма деятельности состоит из трех разделов: «Пользователь», «Веб-приложение» и «База данных» [13].

Прецедент и деятельность начинаются, когда Пользователь решает на странице статьи нажать на кнопку «Добавить комментарий». Таким образом, начальный узел указывает на узел управления с соответствующим названием. Далее происходит переход к узлу-деятельности «Обработать запрос заполнения формы для комментария».

После заполнения формы пользователь нажимает кнопку «Добавить», и происходит переход к узлу-деятельности «Отправить форму». Затем система проверяет корректность заполнения полей с помощью узла-разветвления «Проверить на корректность заполнения полей»: если данные введены некорректно, то происходит возврат к узлу-деятельности «Перенаправить пользователя на страницу с формой для комментария», если верно – «Сформировать SQL-запрос для сохранения информации в базу данных».

Далее происходит переход к узлу-деятельности «База данных выполняет SQL-запрос», откуда узел-деятельности отправляет результат об успешном выполнении SQL-запроса «Веб-приложению». После этого происходит перенаправление пользователя на страницу статьи с помощью узла-деятельности «Перенаправить пользователя на страницу статьи». Прецедент завершается.

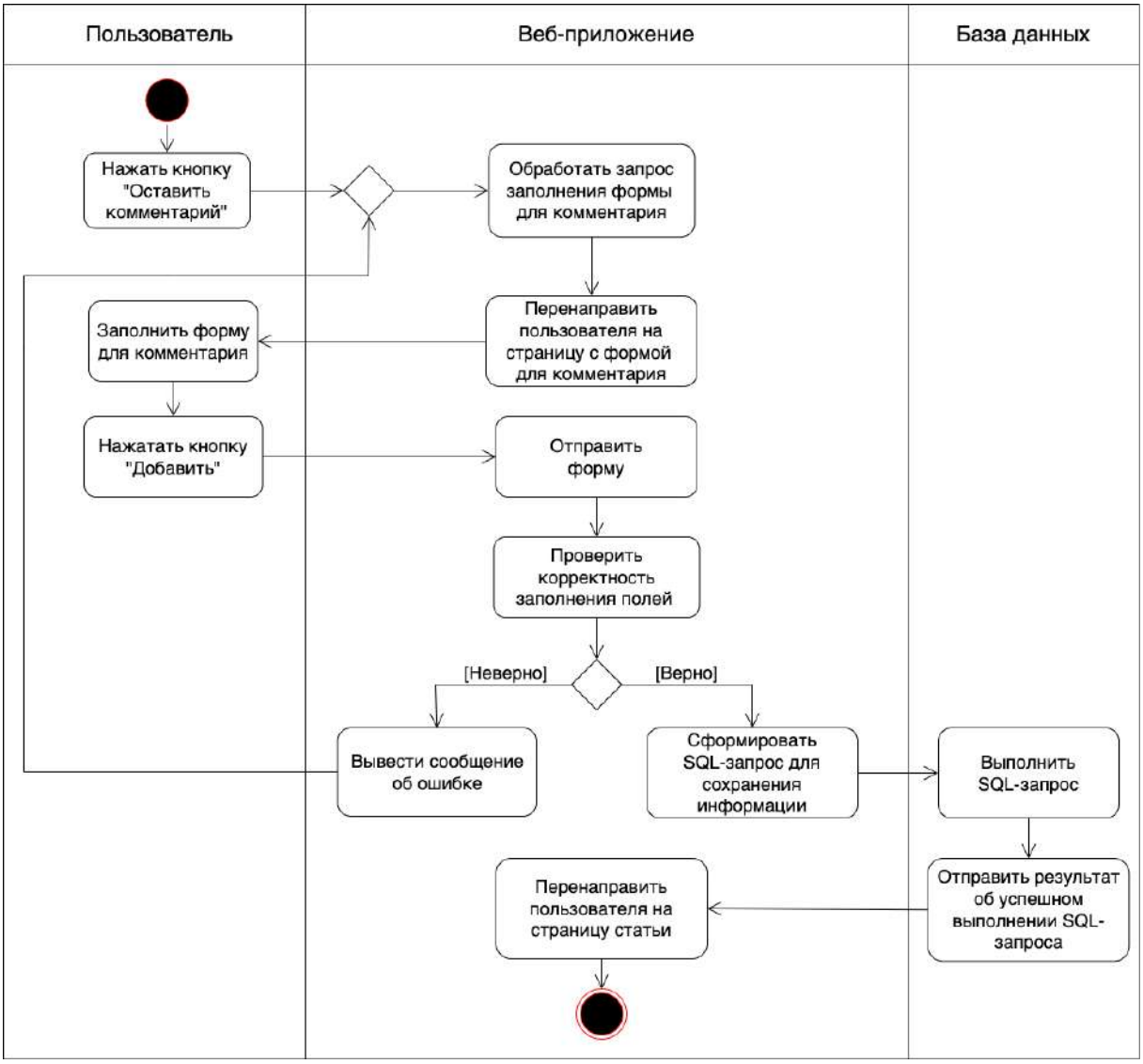

Рисунок 10 – Диаграмма деятельности

Вывод по третьей главе

Была рассмотрена архитектура для разрабатываемой системы, а также создана с использованием языка UML диаграмма деятельности для прецедента «Добавить комментарий».

4. РЕАЛИЗАЦИЯ СИСТЕМЫ

4.1. Реализация базы данных

По умолчанию в Django используется база данных SQLite3 [14]. Она создается при первом обращении к ней. SQLite3 – это автономный, работающий без сервера, транзакционный механизм базы данных SQL.

Структура базы данных в Django создается с помощью моделей. Модель – это класс, который описывает одну таблицу в базе данных, в том числе и ее поля. Каждый экземпляр класса соотносится с отдельной записью в таблице. С его помощью можно получать, изменять и определять новые значения полей таблицы [15].

Согласно описанным функциональным требованиям, можно сделать вывод, что использование встроенной базы данных SQLite вполне достаточно для решения поставленных задач.

В листинге 1 представлен код модели `Blog`, которая определяет, как будут храниться статьи блога в базе данных. Модель содержит следующие поля: `title` (заголовок), `description` (описание), `date` (дата публикации) и `status` (статус публикации). В поле `status_choices` заданы варианты статусов для выбора при создании новой записи блога.

Также определен метод `str`, который возвращает заголовок блога при вызове объекта как строки.

В конце класса определен Meta-класс с параметром `ordering`, который указывает на порядок сортировки записей блога по убыванию даты публикации.

Листинг 1 – Модель `Blog`

```
class Blog(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Draft'),
    )
    title = models.CharField(max_length=250)
    description = models.CharField(max_length=1000)
    date = models.DateField()
    status = models.CharField(max_length=10, choices=STATUS_CHOICES, de-
fault='draft')
    def __str__(self):
        return self.title
```

```
class Meta:
    ordering = ('-publish',)
```

В листинге 2 представлен код модели `Comment`, которая определяет, как будут храниться комментарии в базе данных.

Листинг 2 – Модель `Comment`

```
class Comment(models.Model):
    post = models.ForeignKey(Blog, on_delete=models.CASCADE,  relat-
ed_name='comments')
    name = models.CharField(max_length=80)
    email = models.EmailField()
    body = models.TextField()
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    active = models.BooleanField(default=True)
    class Meta:
        ordering = ('created',)
    def __str__(self):
        return 'Comment by {} on {}'.format(self.name, self.post)
```

Модель содержит следующие поля: `post` – связь с моделью `Blog`, т.е. комментарий будет относиться к конкретному блогу, name – имя пользователя, который оставил комментарий, `email` – электронная почта пользователя, `body` – текст комментария, `created` – дата и время создания комментария, `updated` – дата и время последнего изменения комментария, `active` – флаг активности комментария. Также в модели определены два метода: `str` для строкового представления объекта и `Meta` для управления сортировкой комментариев.

Регистрации моделей `Blog` и `Comment` в административную панель представлена в листинге 3.

Листинг 3 – Определение административных панелей для моделей `Blog` и `Comment`

```
class BlogAdmin(admin.ModelAdmin):
    list_display = ['title', 'status', 'date']
    list_filter = ['status', 'date']
    date_hierarchy = 'date'
    ordering = ['status', 'date']
    search_fields = ('title',)
admin.site.register(Blog, BlogAdmin)
class CommentAdmin(admin.ModelAdmin):
    list_display = ('name', 'email', 'post', 'created', 'active')
    list_filter = ('active', 'created', 'updated')
    search_fields = ('name', 'email', 'body')
admin.site.register(Comment, CommentAdmin)
```

В классе `BlogAdmin` задаются настройки отображения списка записей блога, фильтры, порядок сортировки, поиск по заголовку.

В классе `CommentAdmin` задаются настройки отображения списка комментариев к записям блога: имя автора комментария, `email` автора, связь с конкретной записью блога (`post`), дата создания/обновления и активность комментария. Также здесь заданы фильтры для управления списком комментариев.

Пользовательская форма регистрации, представленная в листинге 4, наследуется от встроенной в Django формы `UserCreationForm`. В конструкторе `__init__` переопределяются ярлыки и подсказки для стандартных полей имени пользователя, пароля и подтверждения пароля. Также в форму добавляется новое поле `email`, которое будет использоваться для ввода электронной почты пользователя.

Листинг 4 – Форма регистрации

```
class UserCreationCustomForm(UserCreationForm):
    email = EmailField(label="Электронная почта", help_text="")
    def __init__(self, *args, **kwargs):
        super(UserCreationCustomForm, self).__init__(*args, **kwargs)
        self.fields['username'].label = "Имя пользователя"
        self.fields['username'].help_text = None
        self.fields['password1'].help_text = None
        self.fields['password2'].help_text = None
        self.fields['password1'].label = 'Пароль'
        self.fields['password2'].label = 'Подтверждение пароля'
    def clean_password2(self):
        password1 = self.cleaned_data.get("password1")
        password2 = self.cleaned_data.get("password2")
        if password1 and password2 and password1 != password2:
            raise forms.ValidationError("Пароли не совпадают")
        return password2
    def save(self, commit=True):
        user = super(UserCreationCustomForm, self).save(commit=False)
        user.email = self.cleaned_data["email"]
        if commit:
            user.save()
        return user
```

Метод `clean_password2` переопределяется для того, чтобы проверить, что пароли, введенные пользователем в поля «Пароль» и «Подтверждение пароля», совпадают. Если они не совпадают, будет выброшено исключение `forms.ValidationError`.

Наконец, метод save переопределяется, чтобы сохранить нового пользователя в базе данных. Он вызывает метод save базового класса UserCreationForm, а затем сохраняет электронную почту пользователя в поле email модели пользователя.

4.2. Реализация заказа товаров

Ключевые этапы разработки пользовательского интерфейса для заказа товаров, который включает в себя проверку ввода данных, расчет общей стоимости и очистку полей формы представлен в листинге 5.

Листинг 5 – Скрипт пользовательского интерфейса для заказа товаров

```
<script>
    function validateForm() {
        var forms = document.querySelectorAll('.postcard');
        var totalPrice = 0;
        forms.forEach(function(form) {
            var link = form.querySelector('input[name="name"]').value;
            var price = parse-
Float(form.querySelector('input[name="price"]').value);
            var quantity = par-
seInt(form.querySelector('select[name="quantity"]').value);
            var linkError = form.querySelector('.error-message');
            var linkPattern = /^https:\/\//;
            if (!linkPattern.test(link)) {
                linkError.style.display = "block";
            } else {
                linkError.style.display = "none";
            }
            totalPrice += (price * quantity);
        });
        var deliveryCost = 5;
        if (totalPrice > 50) {
            deliveryCost = 15;
        } else if (totalPrice > 80) {
            deliveryCost = 28;
        } else if (totalPrice > 10) {
            deliveryCost = 15;
        }
        var totalCost = totalPrice + deliveryCost;
        alert("Итоговая цена: " + totalCost + " Euro (включая " + delivery-
Cost + " Euro - стоимость доставки)");
    }
</script>
```

Валидация формы в данном коде осуществляется следующим образом: сначала происходит получение всех форм с помощью `querySelectorAll`, затем для каждой формы проверяется корректность введенной ссылки на товар, используя регулярное выражение. Если ссылка

не начинается с «https://», то отображается сообщение об ошибке, в противном случае ошибка скрывается.

После определения стоимости доставки вычисляется общая стоимость заказа, включающая в себя как стоимость товаров, так и стоимость доставки.

Механизм очистки формы после ее заполнения и валидации реализован с помощью обнуления значений полей формы, а также скрытия сообщения об ошибке в ссылке, чтобы пользователь мог начать заполнение поля формы заново.

4.3. Реализация регистрации и авторизации пользователей

Основная логика авторизации пользователей в веб-приложении представлена в листинге 6.

Листинг 6 – Авторизация пользователя

```
def loginuser(request):
    if request.method == 'POST':
        user = authenticate(request, username=request.POST['username'],
password=request.POST['password'])
        if user is not None:
            login(request, user)
            return redirect('currenttobuy')
        else:
            return render(request, 'inform/loginuser.html', {'form': Au-
thenticationForm(), 'error': 'Пользователь не найден'})
    return render(request, 'inform/loginuser.html', {'form': Authentica-
tionForm()})
```

Функция loginuser отвечает за процесс авторизации пользователей. Когда пользователь переходит на страницу авторизации, ему отображается форма на основе стандартной модели AuthenticationForm. Если пользователь заполняет форму и отправляет ее, происходит проверка введенных данных с помощью функции authenticate(). Если пользователь найден в системе, он авторизуется, и его перенаправляет на страницу currenttobuy. Если пользователь не найден, на страницу авторизации возвращается сообщение об ошибке.

Также была разработана функция, которая позволяет легко переключаться между формой авторизации и формой восстановления пароля (листинг 7).

Когда пользователь нажимает на ссылку, вызывающую функцию showForgotPasswordForm(), форма восстановления пароля становится видимой, а форма авторизации скрывается.

Листинг 7 – Функция для переключения между формами авторизации и восстановления пароля на странице

```
<script>
    function showForgotPasswordForm() {
        document.getElementById("loginForm").style.display = "none";
        document.getElementById("forgotPasswordForm").style.display =
"block";
    }
    function showLoginForm() {
        var emailInput = document.getElementById("emailInput").value;
        if (/\S+@\S+\.\S+/.test(emailInput)) {
            document.getElementById("forgotPasswordForm").style.display =
"block";
            document.getElementById("loginForm").style.display = "none";
        }else {
                    }
    }
</script>
```

Механизм регистрации пользователя представлен в листинге 8.

Листинг 8 – Регистрация пользователя

```
def signupuser(request):
    if request.method == 'POST':
        form = UserCreationCustomForm(request.POST)
        if form.is_valid():
            user = form.save()
            # После успешной регистрации перенаправляем пользователя на
страницу "currenttobuy"
            return redirect('currenttobuy')
    else:
        form = UserCreationCustomForm()
    return render(request, 'inform/signupuser.html', {'form': form})
```

Функция `signupuser` отвечает за процесс регистрации новых пользователей. Когда пользователь переходит на страницу регистрации, ему отображается форма, созданная на основе пользовательской модели `UserCreationCustomForm`. Если пользователь заполняет форму и отправляет ее, то происходит проверка корректности введенных данных.

4.4. Реализация оценок комментариев пользователей под статьей

Для обеспечения взаимодействия пользователей и выражения своего мнения о комментариях был разработан скрипт, представленный в листинге 9.

Листинг 9 – Скрипт для оценки комментариев под статьей

```
<script>
    function likeComment(btn) {
        let likeCount = btn.nextElementSibling;
        let dislikeButton = btn.nextElementSibling.nextElementSibling;
        let dislikeCount = dislikeButton.nextElementSibling;
        if (btn.classList.contains("active")) {
            likeCount.innerText = parseInt(likeCount.innerText) - 1;
            btn.classList.remove("active");
        } else {
            likeCount.innerText = parseInt(likeCount.innerText) + 1;
            btn.classList.add("active");
            if (dislikeButton.classList.contains("active")) {
                dislikeCount.innerText = parseInt(dislikeCount.innerText) -
1;
                dislikeButton.classList.remove("active");
            }
        }
    }
</script>
```

Функция `likeComment(btn)` отвечает за обработку нажатия на кнопку «Нравится» под комментарием. Происходит получение ссылки на кнопку «Нравится», которая была нажата. Затем активируется поиск ссылки на элементы, отображающие количество «лайков» и «дизлайков» для этого комментария.

Вывод по четвертой главе

Реализация веб-приложения для байерского агентства выполнена в соответствии с целью работы. SQLite [16] и Python 3 использовались для реализации регистрации и авторизации пользователя, новостного блога агентства, форм для заказа и комментариев.

5. ТЕСТИРОВАНИЕ СИСТЕМЫ

5.1. Функциональное тестирование

В соответствии с функциональными требованиями, представленными в разделе 1.2, было проведено функциональное тестирование каждого разработанного модуля. В ходе указанного тестирования проверялась работоспособность всех описанных ранее задач, которые были поставлены. Набор функциональных тестов приведен в таблице 1.

Таблица 1 – Функциональное тестирование

№	Функция	Шаги	Ожидаемый результат
1	Регистрация пользователя	1. Запустить веб-приложение. 2. Нажать на кнопку «Зарегистрироваться» в разделе «Меню». 3. Заполнить данными форму. 4. Нажать на кнопку «Зарегистрироваться».	Переход на экран «Личный кабинет пользователя»
2	Авторизация пользователя	1. Запустить приложение. 2. Нажать на кнопку «Вход в аккаунт» в разделе «Меню». 3. Заполнить данными форму. 4. Нажать на кнопку «Вход».	Переход на экран «Личный кабинет пользователя»
3	Пропуск поля имени или пароля при регистрации или авторизации	1. Запустить веб-приложение. 2. Нажать на кнопку «Зарегистрироваться» в разделе «Меню». 3. Заполнить данными форму. 4. Нажать на кнопку «Зарегистрироваться».	Ошибка регистрации
4	Ввод неправильного пароля при авторизации	1. Запустить приложение. 2. Нажать на кнопку «Войти в аккаунт» в разделе «Меню». 3. Заполнить некорретными данными поле для пароля в форме авторизации. 4. Нажать на кнопку «Вход».	Ошибка авторизации
5	Добавление комментария к статье в разделе «Новостной блог»	1. Запустить веб-приложение. 2. Открыть раздел «Новостной блог». 3. Открыть любую статью из блога. 4. Нажать на кнопку «Добавить комментарий». 5. Заполнить поля формы.	Успешное отображение комментария на странице со статьей

Окончание таблицы 1

№	Функция	Шаги	Ожидаемый результат
6	Пропуск поля с именем пользователя при заполнении формы для комментария	1. Запустить веб-приложение. 2. Открыть раздел «Новостной блог». 3. Открыть любую статью из блога. 4. Нажать на кнопку «Добавить комментарий». 5. Не заполнять поле с именем пользователя в форме комментария. 6. Нажать на кнопку «Отправить».	Ошибка заполнения формы
7	Расчет итоговой стоимости товара	1. Запустить веб-приложение. 2. Войти в личный кабинет. 3. Открыть раздел «Оформить заказ». 4. Заполнить все поля формы. 5. Нажать на кнопку «Итоговая стоимость».	Успешное отображение окна с выводом информации об итоговой корректной стоимости товара
8	Очистка полей формы для заказа	1. Запустить веб-приложение. 2. Войти в личный кабинет. 3. Открыть раздел «Оформить заказ». 4. Заполнить все поля формы. 5. Нажать на кнопку «Очистить поля».	Поля формы для заказа успешно очищены
9	Редактирование раздела «Новостной блог»	Заполнить все поля в таблице «Blog» в административной панели.	Успешное отображение добавления или удаления статьи в разделе «Новостной блог»
10	Редактирование комментария пользователя	Добавить или удалить комментарий в панели администратора.	Успешное отображение добавления комментария пользователя в разделе «Новостной блог»

Вывод по пятой главе

В ходе тестирования были подготовлены тесты и проведено тестирование разработанного веб-приложения. Все тесты были пройдены успешно.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано веб-приложение для байерского агентства. При этом были решены следующие задачи.

1. Определены требования к разрабатываемому веб-приложению.
2. Спроектирован интерфейс веб-приложения.
3. Разработаны клиентская и серверная части веб-приложения.
4. Произведено тестирование и отладка веб-приложения.

В ходе выпускной квалификационной работы были изучены способы разработки веб-приложений и фреймворк Django.

Планируется дальнейшее развитие проекта, включающее в себя добавление новой функции – отображения статуса заказа в личном кабинете пользователя с возможностями:

1) создания нового раздела в личном кабинете пользователя для отслеживания статуса каждого заказа;

2) обеспечения обновлений в реальном времени, чтобы пользователи всегда имели точную информацию о своих заказах.

Помимо этого функционала ожидается внедрение безопасного и надежного платежного шлюза для оптимизации процесса оплаты, предоставление нескольких вариантов оплаты, таких как кредитные или дебетовые карты, цифровые кошельки. Также будет реализовано отображение списка прошлых заказов с возможностями:

1) создания отдельной страницы, где пользователи могут просматривать историю предыдущих заказов;

2) включения деталей заказа, таких как дата, приобретенные товары, общая сумма и статус каждого заказа;

3) разрешения пользователям повторно заказывать товары из истории заказов.

ЛИТЕРАТУРА

1. Агентство Wonderland Buying Office. [Электронный ресурс] URL: https://shopitalia.ru/ (дата обращения: 23.03.2024 г.).

2. ATI ITALY GROUP. [Электронный ресурс] URL: https://tati.group/ (дата обращения: 25.03.2024 г.).

3. Официальный сайт Django. [Электронный ресурс] URL: https://docs.djangoproject.com/en/4.2/ (дата обращения: 25.03.2024 г.).

4. Проектирование UML-диаграмм. [Электронный ресурс] URL: https://uml-class-diagram/ (дата обращения: 25.03.2024 г.).

5. Руководство по веб-фреймворку Django. [Электронный ресурс] URL: https://metanit.com/python/django/ (дата обращения: 25.03.2024 г.).

6. Форсье Дж., Биссекс П., Чан У. Django. Разработка веб-приложений на Python. – СПб.: Символ-Плюс, 2009. – 456 с.

7. Роббинс Дж. HTML5, CSS3 и Javascript. Исчерпывающее руководство. – М.: Эксмо, 2014. – 528 с.

8. Руководство по CSS. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Web/CSS (дата обращения: 25.03.2024 г.).

9. Структура сайта - какие бывают и как правильно создать с учетом SEO. [Электронный ресурс] URL: https://web-revenue.ru/seo/struktura-sayta/ (дата обращения: 25.03.2024 г.).

10. Канер С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. – К.: ДиаСофт, 2001. – 544 с.

11. Куликов С.С. Тестирование программного обеспечения. Базовый курс. 2 издание. – Издательство Четыре Четверти, 2017. – 312 с.

12. Орлов С.А., Цилькер Б.Я. Технологии разработки программного обеспечения. Учебник для вузов. 4-е издание. Стандарт третьего поколения. – Издательский дом «Питер», 2021. – 608 с.

13. Библиотека Python Pillow. The friendly PIL fork. [Электронный ресурс] URL: https://python-pillow.org (дата обращения: 03.04.2024 г.).

14. Головатый К.А. Django //Подробное руководство Django. The definitive guide to/пер. с англ. СПб. Символ-Плюс, 2010. – 560 с.

15. Django. Создание моделей. [Электронный ресурс] URL: https://metanit.com/python/django/5.1/ (дата обращения: 25.05.2024 г.).

ПРИЛОЖЕНИЯ

Приложение А. Спецификация вариантов использования

Спецификация вариантов использования для веб-приложения приведена в таблицах 1–7.

Таблица 1 – Спецификация прецедента «Зарегистрировать пользователя»

Прецедент: Зарегистрировать пользователя
ID: 1
Краткое описание: Регистрация пользователя
Главные актеры: Пользователь
Предусловия: Нет
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Зарегистрироваться» в разделе «Меню» главной страницы. 2. Система обрабатывает пользовательский запрос и перенаправляет его на страницу «Регистрации». 3. Пользователь вводит свои данные в поля формы «Регистрация» и нажимает кнопку «Регистрация». 4. Система формирует SQL-запрос в базу данных для сохранения данных. 5. База данных обрабатывает SQL-запрос и выдает системе результат об успешной операции. 6. Система перенаправляет пользователя на страницу «Регистрация».
Постусловия: Пользователь зарегистрирован
Альтернативные потоки: Отсутствуют

Таблица 2 – Спецификация прецедента «Посмотреть список брендов»

Прецедент: Просмотреть список доступных брендов к заказу
ID: 2
Краткое описание: Просмотр списка брендов
Главные актеры: Пользователь
Предусловия: Нет
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Бренды» в разделе «Меню» главной страницы. Система обрабатывает пользовательский запрос и перенаправляет его в раздел «Бренды для заказа», находящегося на главной странице.
Альтернативные потоки: Отсутствуют

Таблица 3 – Спецификация прецедента «Оформить заказ»

Прецедент: Оформить заказ
ID: 3
Краткое описание: Оформление заказа
Главные актеры: Пользователь
Предусловия: Пользователь зарегистрирован
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Оформить заказ» в разделе «Меню» главной страницы своего личного кабинета. 2. Система обрабатывает пользовательский запрос и перенаправляет его на страницу «Оформление заказа». 3. Пользователь вводит свои данные в поля формы «Оформление заказа» и нажимает кнопку «Отправить». 4. Система формирует SQL-запрос в базу данных для сохранения данных. 5. База данных обрабатывает SQL-запрос и выдает системе результат об успешной операции. Система перенаправляет пользователя на главную страницу его личного кабинета.
Постусловия: Заказ оформлен
Альтернативные потоки: Отсутствуют

Таблица 4 – Спецификация прецедента «Посмотреть список заказов»

Прецедент: Посмотреть список заказов
ID: 4
Краткое описание: Просмотр списка заказов
Главные актеры: Администратор
Предусловия: Администратор зарегистрирован
Основной поток: 1. Вариант использования начинается, когда администратор входит в административную панель. 2. Администратор нажимает кнопку «Order» в таблице базы данных. 3. Система отображает список всех заказов пользователей.
Постусловия: Отображение списка всех заказов пользователей
Альтернативные потоки: Отсутствуют

Таблица 5 – Спецификация прецедента «Добавить комментарий»

Прецедент: Добавить комментарий
ID: 5
Краткое описание: Добавление комментария к статье из раздела «Новостной блог»
Главные актеры: Пользователь
Предусловия: Пользователь находится на странице развернутой статьи раздела «Новостной блог»
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Добавить комментарий» на странице развернутой статьи. 2. Система обрабатывает пользовательский запрос и перенаправляет его на страницу «Добавление комментария». 3. Пользователь вводит свои данные в поля формы «Добавление комментария» и нажимает кнопку «Добавить». 4. Система формирует SQL-запрос в базу данных для сохранения данных формы. 5. База данных обрабатывает SQL-запрос и выдает системе результат об успешной операции. 6. Система перенаправляет пользователя на страницу статьи.
Постусловия: На странице статьи добавлен комментарий пользователя
Альтернативные потоки: Отсутствуют

Таблица 6 – Спецификация прецедента «Посмотреть новостной блог»

Прецедент: Посмотреть новостной блог
ID: 6
Краткое описание: Просмотр новостного блога агентства
Главные актеры: Пользователь
Предусловия: Нет
Основной поток: Вариант использования начинается, когда пользователь нажимает на кнопку «Новостной блог» в разделе «Меню» на главной странице. Система обрабатывает пользовательский запрос и перенаправляет его на страницу «Новостной блог».
Постусловия: На экран выведена страница новостного блога
Альтернативные потоки: Отсутствуют

Таблица 7 – Спецификация прецедента «Авторизовать пользователя»

Прецедент: Авторизовать пользователя
ID: 7
Краткое описание: Авторизация пользователя
Главные актеры: Пользователь
Предусловия: Пользователь зарегистрирован
Основной поток: 1. Вариант использования начинается, когда пользователь нажимает на кнопку «Войти в кабинет» на главной странице. 2. Система обрабатывает пользовательский запрос и перенаправляет его на страницу «Авторизация». 3. Пользователь вводит свои данные в поля формы «Авторизация» и нажимает кнопку «Войти». 4. Система формирует SQL-запрос в базу данных для обработки данных с формы. 5. База данных обрабатывает SQL-запрос и выдает системе результат об успешной операции. 6. Система перенаправляет пользователя на страницу его личного кабинета.
Постусловия: Пользователь авторизован
Альтернативные потоки: Отсутствуют

Приложение Б. Диаграмма вариантов использования системы

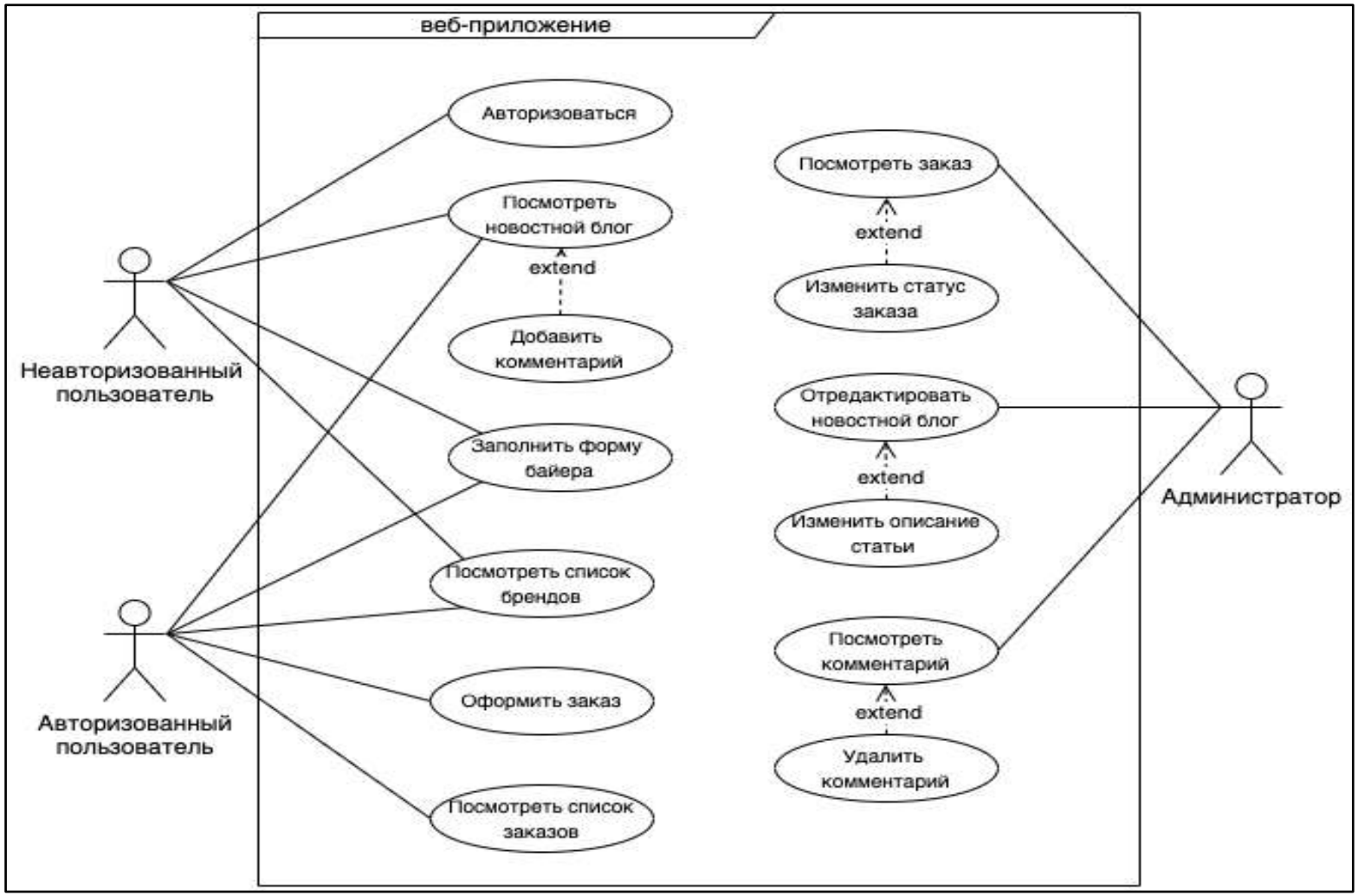

Рисунок 1 – Диаграмма вариантов использования