## **Содержание**

[**1. Техническое задание** 4](#_Toc27960036)

[**2. Исследовательская часть** 9](#_Toc27960056)

[2.1. Постановка задачи проектирования 9](#_Toc27960057)

[2.2 Пользовательские сценарии 10](#_Toc27960058)

[2.2.2 Сценарии использования. 13](#_Toc27960059)

[2.4. Обоснование выбора инструментов и платформы для разработки 16](#_Toc27960060)

[**3. Проектно-конструкторская часть** 19](#_Toc27960061)

[3.1 Разработка структуры программной системы 19](#_Toc27960062)

[3.2. Разработка пользовательского интерфейса 20](#_Toc27960063)

[**4. Проектно-технологическая часть** 23](#_Toc27960064)

[4.1 Технологические решения, поддерживающие эксплуатационный цикл программы 23](#_Toc27960065)

[**Выводы** 26](#_Toc27960066)

[**Список источников** 27](#_Toc27960067)

# 1. Техническое задание

# 1.1. Введение

# 1.1.1. Наименование программы

Web-сервис для самозанятых “HandyMady”.

# 1.1.2. Краткая характеристика области применения

Сайт предоставляет возможность самозанятым создавать свои магазины, где он может выставлять свои работы на продажу и зарабатывать рейтинг. Рейтинг составляется из отзывов пользователей и поможет остальным найти лучшего продавца. Данная разработка позволяет объединить продавцов различных рукодельных предметов и пользователей на одном сайте.

# Основания для разработки

Разработка ведется на основании данного технического задания. Этим документом подтверждается процесс разработки программного обеспечения на тему “Сайт для самозанятых”.

# 1.3. Назначение разработки

# 1.3.1. Наименование темы разработки

“Разработка графического интерфейса web-сервиса для самозанятых “HandyMady”.

# 1.3.2. Функциональное назначение программы

Функциональным назначением разработки указанного программного обеспечения является повышение эффективности работы, облегчения рабочего процесса, помощь в реализации товаров и услуг для самозанятых лиц, а также удобная торговая площадка для покупателей с возможностью поиска.

Критериями повышения эффективности будут служить:

* уменьшение затрат времени на поиск продавцов и товаров;
* удобный интерфейс для выставления товаров и для аукциона;
* возможность общения покупателей с самозанятыми;
* отслеживание доставки;

# 1.3.3. Эксплуатационное назначение

Подразумевается эксплуатация системы пользователями и самозанятыми, получающие доход от своей личной трудовой деятельности.

# 1.4. Требования к программному изделию

# 1.4.1. Требования к функциональным характеристикам

Программная система должна представлять собой работоспособный сайт, реализующее следующие функции:

* функция регистрации для пользователей и продавцов;
* функция создания магазина;
* функция выставления товаров;
* функция ведения акций в магазине;
* функция рейтинга для продавцов, основывающаяся на оценках пользователей;
* поиск магазинов и товаров по категориям;
* функция связи между продавцом и покупателем;
* функция отслеживания доставки;

# 1.4.2. Требования к надежности

Система должна работать стабильно, не допускается прерываний работы системы, вызванных ее внутренними ошибками. Интерфейс системы не должен содержать элементов, назначение которых неочевидно, которые вводят пользователя в заблуждение или не несут функционального значения.

# 1.4.3. Условия эксплуатации

Система предназначена для использования в нормальных условиях. Специальных требований по эксплуатации системе не предъявляется.

# 1.4.4. Требования к составу и параметрам технических средств

Специальных требований к составу и параметрам технических средств не предъявляется.

# 1.4.5. Требования к информационной и программной совместимости

Сайт должен быть совместим со всеми актуальными браузерами, в качестве системы управления базой данных будет использоваться PostgreSQL. Исходные коды (back-end) должны быть реализованы на языке Python посредством свободного фреймворка для веб-приложений Django. Визуальное представление (front-end) должно быть реализовано на языках HTML и CSS. В качестве среды разработки программы должна быть использована среда VS Code.

# . Требования к программной документации

Разрабатываемое программное обеспечение не требует разработки руководства пользователя. Техническое задание должно быть оформлено в соответствии с требованиями ГОСТ.

# 1.6. Стадии и этапы разработки

Содержимое разделов настоящего технического задания может быть изменено и дополнено по согласованию с руководителем.

# 1.6.1. Стадии разработки

Разработка должна быть проведена в пять стадий стадии:

* Утверждение технического задания;
* Эскизный проект;
* Технический проект;
* Рабочий проект;
* Ввод в эксплуатацию

# 1.6.2. Этапы разработки

Порядок разработки программной системы разбит на следующие стадии:

**Этап утверждения технического задания**

* Обоснование необходимости разработки ПО;
* Исследование предметной области
* Разработка и утверждение технического задания

Результатом работы на данном этапе является согласованное и утвержденное техническое задание на разработку системы и описание предметной области.

Срок – 11.09.2019.

**Этап проектирования:**

* Разработка концептуальной схемы данных, прототипа или эскиза интерфейса;
* Разработка утверждение с заказчиком эскизного проекта;

Результатом работы на данном этапе является оформленная исследовательская часть, часть разработанных и реализованных алгоритмов функционирования приложения, прототип интерфейса, концептуальная и логическая схемы базы данных.

Срок – 24.09.2019.

**Этап технического проекта:**

* Оформление проектно-конструкторской части
* Разработка технической версии проекта
* Демонстрация разработанной версии приложения

Результатом работы на данном этапе будет являться оформленная проектно-конструкторская часть и рабочая техническая версия ПО

Срок - 03.11.2019.

**Этап рабочего проекта:**

* Разработка оставшегося функционала приложения
* Отладка и тестирование рабочей версии проекта
* Разработка программной документации

Результатом работы на данном этапе является оформленная проекто-технологическая часть, рабочая версия ПО и завершенная программная документация

Срок – 01.12.2019.

**Этап ввода в эксплуатацию:**

* Подготовка системы к развертыванию
* Первоначальная настройка системы

Результатом работы на данном этапе является спроектированный базовый функционал разрабатываемой системы.

# 1.7. Порядок контроля и приемки

Система в процессе разработки должна быть протестирована. Приёмка осуществляется преподавательским составом кафедры СОИ КФ МГТУ им. Н.Э.Баумана.

**2. Исследовательская часть**

# 2.1. Постановка задачи проектирования

Целью написания данной курсовой работы является разработка интерфейса web-сервиса «HandyMady». Данный сервис предназначен для облегчения рабочего процесса самозанятых лиц, а также реализации товаров и услуг при помощи поиска и рекомендательной системы.

Задача проектирования такой системы может быть решена при помощи следующих средств разработки: средством реализации пользовательских интерфейсов должно быть реализовано на языках HTML и CSS, системы управления базами данных PostgreSQL, Исходные коды должны быть реализованы на языке Python посредством свободного фреймворка для веб-приложений Django.

Структурное проектирование системы осуществляется с помощью составления концептуальной и логической моделей хранимых данных. Описание этапов проектирования архитектуры приложения сопровождается описанием на естественном языке.

Необходимо разработать интернет-магазин, реализующий товары и услуги самозанятых. Система должна обеспечить удобный интерфейс продавцам для создания своего магазина с возможностью добавления товара в каталог и ведение акций, в то время как пользователь получит удобный интерфейс поиска товаров по категориям и рекомендациям.

# 2.2 Пользовательские сценарии

В таблицах 1 – 11 представлены основные пользовательские сценарии поведения системы.

Таблица 1 – Сценарий регистрации пользователя

|  |  |
| --- | --- |
| **Название** | Регистрация |
| **Краткое описание** | Пользователь регистрируется в системе для дальнейшего взаимодействия с ней. |
| **Участники** | Пользователь, система |
| 1. Базовый сценарий 2. Пользователь открывает сайт. 3. Пользователь нажимает кнопку «Регистрация» 4. Система отправляет пользователю форму регистрации. 5. Пользователь вводит данные во все поля формы. 6. Система отправляет данные на сервер для занесения пользователя в базу данных. | |
| **Постусловие** | Пользователь зарегистрирован в системе. |
| **Расширения:** | |
| **1** | Пользователь заполнил не все обязательные поля.  Система сообщит об ошибке и попросит их заполнить. |
| **2** | Имя пользователя занято  Система сообщит об ошибке и попросит выбрать другое. |
| **3** | Пароль не соответствует требованиям.  Система сообщит об ошибке и попросит выбрать другой. |

Таблица 2 – Сценарий входа в систему

|  |  |
| --- | --- |
| **Название** | Вход в систему |
| **Краткое описание** | Пользователь входит в систему под своим логином и паролем для дальнейшего взаимодействия с ней |
| **Участники** | Пользователь, система |
| 1. Базовый сценарий 2. Пользователь открывает сайт и нажимает на кнопку «Войти в систему». 3. Система отправляет пользователю форму авторизации. 4. Пользователь вводит имя пользователя и пароль. 5. Система проверяет корректность ввода логина и пароля. 6. Пользователь авторизован в системе. | |
| **Постусловие** | Пользователь авторизовался в системе |
| **Расширения:** | |
| **1** | Неверный логин или пароль  Система уведомит об ошибке и попросит ввести корректные данные. |

Таблица 3 – Сценарий добавления товара в корзину

|  |  |
| --- | --- |
| **Название** | Добавление товара в корзину |
| **Краткое описание** | Пользователь добавляет товар, который он хочет купить в корзину. |
| **Участники** | Пользователь, система |
| 1. Пользователь пишет в строку поиска название товара и при желании изменяет настройки поиска. 2. Система выдает пользователю результат поиска с выбранными настройками 3. Пользователь выбирает в результатах поиска товар, который ему нужен. 4. Ознакомившись с описанием товара, информацией о доставке и о цене пользователь нажимает на кнопку «добавить в корзину». | |
| **Постусловие** | Пользователь добавил товар в корзину |
| **Расширения:** | |
| **1** | Отсутствие товара в наличии  Система уведомляет об этом пользователя об отсутствии товара. Также система уведомит о следующем поступлении если продавец указал информацию. |

Таблица 4 – Сценарий оформления пользовательского заказа

|  |  |
| --- | --- |
| **Название** | Оформление пользовательского заказа |
| **Краткое описание** | Пользователь оформляет заказ и оплачивает его. |
| **Участники** | Пользователь, продавец, система |
| 1. В меню корзины пользователь выбирает кнопку «оформить заказ» 2. Система списывает деньги со счета в профиле сервиса. 3. Система уведомляет продавца об отправке товара. 4. Продавец отправляет товар выбранным способом доставки | |
| **Постусловие** | Пользователь оформил заказ |

Таблица 5 – Сценарий закрытия заказа

|  |  |
| --- | --- |
| **Название** | Закрытие заказа |
| **Краткое описание** | После получения товара пользователь закрывает заказ и ставит оценку. |
| **Участники** | Пользователь, система |
| 1. Пользователь выбирает заказ из списка 2. Пользователь нажимает кнопку «Закрыть заказ» 3. Система помечает заказ в базе данных как завершенный. 4. Система предлагает пользователю оценить товар и услуги магазина. | |
| **Постусловие** | Пользователь закрыл заказ |

# 2.2.2 Сценарии использования.

После регистрации на сайте пользователь может как делать заказы в магазинах других продавцов, так и создать свои магазины с различными товарами и продавать их. У пользователя есть возможность просмотра каталога товаров по различным категориям. Пользователь как в роли покупателя, так и в роли продавца может создать несколько магазинов с различными категориями товаров. У каждого товара есть свое описание, кратко рассказывающее о том, что представляет из себя продукт, и фотографии, при помощи которых можно построить себе визуальное представление о товаре. Чтобы сделать заказ покупатель должен добавить товары в корзину, указать номер телефона и телефон, если они не написаны в профиле и оплатить заказ. После того как товар был получен покупатель получает право оценить по пятибалльной шкале товар, продавца и магазин. Также пользователь может воспользоваться поиском товаров с различными фильтрами для точности. Для поиска товаров будет реализована поисковая строка. Поиск может проводиться как по названию, так и по различным категориям товаров. Покупатель может зайти на страницу магазина других пользователей, где может подписаться на рассылку новостей о появлении новый товаров и акций. UML диаграмма пользовательских сценариев (рисунок 2) показывает возможные варианты действий покупателя.



Рисунок 2 – UML диаграмма пользовательских сценариев для покупателя

Продавец имеет такие же пользовательские сценарии что и покупатель, но при этом он может заниматься менеджментом своих магазинов. Он может проводить различные акции индивидуально для каждого своего магазина с настройкой описания, даты начала и окончания. В профиле магазина владелец может видоизменить его вид, добавить товары на продажу и купить рекламу для продвижения товаров. UML диаграмма пользовательских сценариев (рисунок 3) показывает возможные варианты действий продавца.

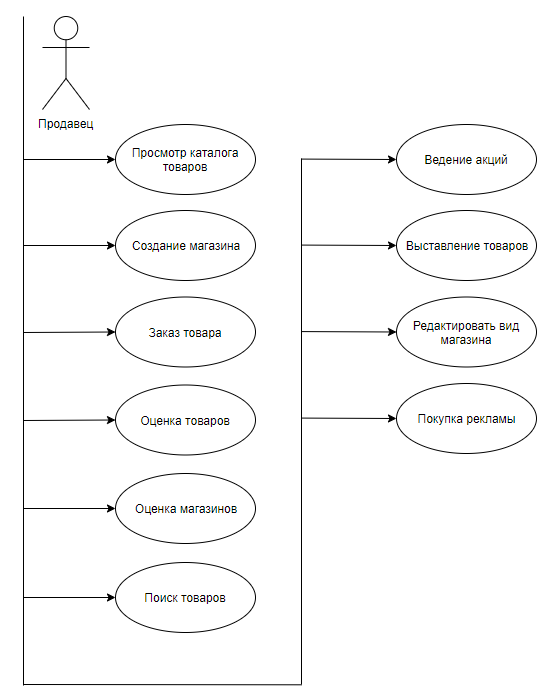


Рисунок 3 – UML диаграмма пользовательских сценариев для продавца

# 2.4. Обоснование выбора инструментов и платформы для разработки

PostgreSQL - это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

* Надежность PostgreSQL.
* Производительность PostgreSQL основывается на использовании индексов, интеллектуальном планировщике запросов, тонкой системы блокировок, системе управления буферами памяти и кэширования, превосходной масштабируемости при конкурентной работе.
* Поддержка SQL
* PostgreSQL имеет очень богатый набор встроенных функций и операторов для работы с данными, полный список которых можно посмотреть в документации.
* Простота использования всегда являлась важным фактором для разработчиков.
* pgAdmin (GNU Artistic license) предоставляет удобный интерфейс для работы с базами данных PostgreSQL.

HTML (HyperText Markup Language — язык гипертекстовой разметки) — стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML до 5-й версии определялся как приложение SGML (стандартного обобщённого языка разметки по стандарту ISO 8879). Спецификации HTML5 формулируются в терминах DOM (объектной модели документа).

Язык XHTML является более строгим вариантом HTML, он следует синтаксису XML и является приложением языка XML в области разметки гипертекста.

Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования.

CSS (Cascading Style Sheets) — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML). Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL. Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

CSS поддерживает таблицы стилей для конкретных носителей, поэтому авторы могут адаптировать представление своих документов к визуальным браузерам, слуховым устройствам, принтерам, брайлевским устройствам, карманным устройствам и т.д.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

Django — это программный каркас с богатыми возможностями, подходящий для разработки сложных сайтов и веб-приложений, написанный на языке программирования Python.

Django — фреймворк для веб-приложений на языке Python. Один из основных принципов фреймворка — DRY (don't repeat yourself). Веб-системы на Django строятся из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из заметных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Также, в отличие от многих других фреймворков, обработчики URL в Django конфигурируются явно (при помощи регулярных выражений), а не автоматически задаются из структуры контроллеров.

Django проектировался для работы под управлением Apache (с модулем mod\_python) и с использованием PostgreSQL в качестве базы данных. В настоящее время, помимо PostgreSQL, Django может работать с другими СУБД: MySQL (MariaDB), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle. Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Первоначально разработка Django велась для обеспечения более удобной работы с новостными ресурсами, что достаточно сильно отразилось на архитектуре: фреймворк предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав).

**3. Проектно-конструкторская часть**

# 3.1 Разработка структуры программной системы

Приложение представляет собой клиент-серверное приложение, которое является четырёхуровневым. Клиент-серверное приложение – это вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение.

Четырёхуровневое клиент-серверное (рисунок 4) приложение содержит в себе:

* Клиент, который представляет пользовательский интерфейс для взаимодействия с остальными уровнями приложения. В данной курсовой работе клиентом будет являться веб-браузер, который будет передавать веб-серверу запросы на получение HTML-страниц, изображений, файлов и других данных, обозначенных URL-адресами.
* HTTP сервер, который принимает HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными,
* Сервер приложения, на котором осуществляется работа с серверными компонентами разрабатываемой системы. На данном уровне осуществляется вся бизнес-логика и управление базой данных.
* Сервер базы данных, выполняет обслуживание и управление базой данных и СУБД, а также отвечает за целостность и сохранность данных и обеспечивает операции ввода-вывода при доступе клиента к информации.

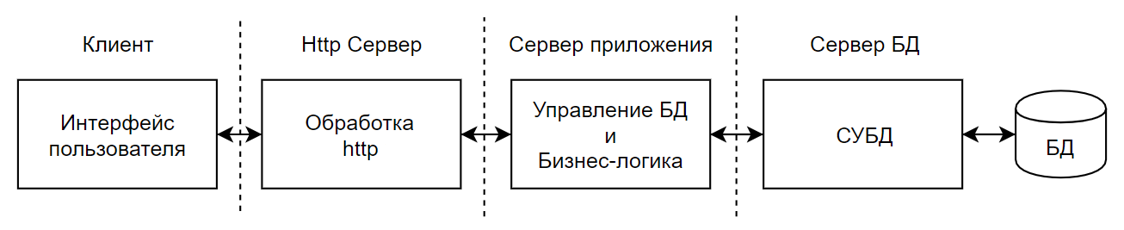


Рисунок 4 – Компоненты системы

Последовательность разработки системы:

1) Проектирование базы данных.

2) Создание логики серверной части.

3) Создание интерфейса.

4) Заполнение базы данных.

5) Тестирование.

# 3.2. Разработка пользовательского интерфейса

Разработку пользовательского интерфейса можно разделить на несколько этапов:

• Создание формы регистрации

• Создание формы для входа в аккаунт (профиль на сайте)

• Создание главной страницы сайта

• Создание страницы для товаров

• Создание адаптивного интерфейса

Во время проектирования сайта был выбран тип блочной вёрстки, так как он наиболее подходит для нынешней работы с данными. Основным принципом его работы и отличия от табличной вёрстки является использование тегов <div> - специальных блоков, которые отвечают за отдельные части. Так же данный тип обладает рядом преимуществ:

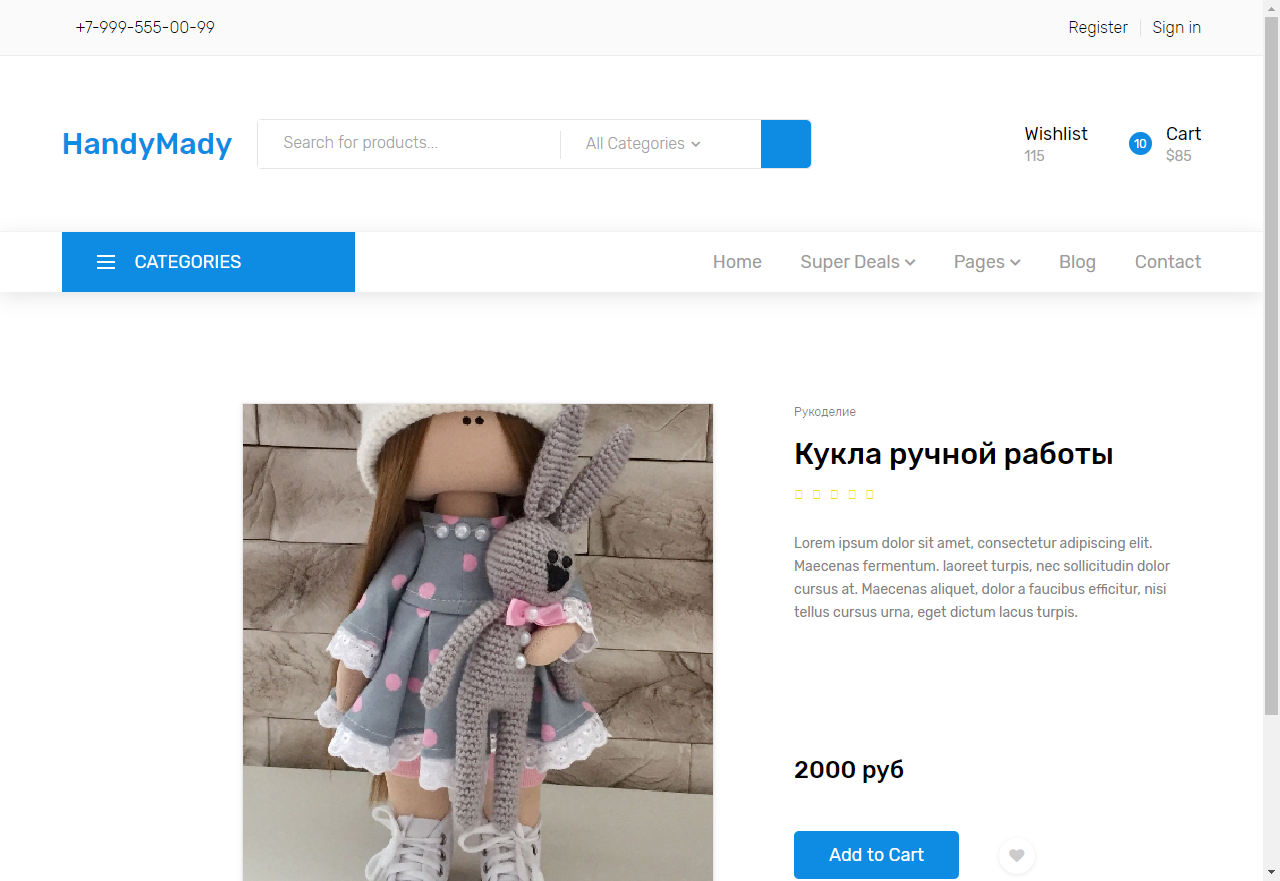
• Высокая скорость загрузки страницы

• Лучшая индексация поисковиками

• Отделение стилей элементов от основного кода Html

• Возможность наложения одного слоя на другой

• Более лёгкое создание визуальных эффектов, таких как всплывающие подсказки, списки, выпадающие меню

Рисунок 5 – Макет страницы с товаром

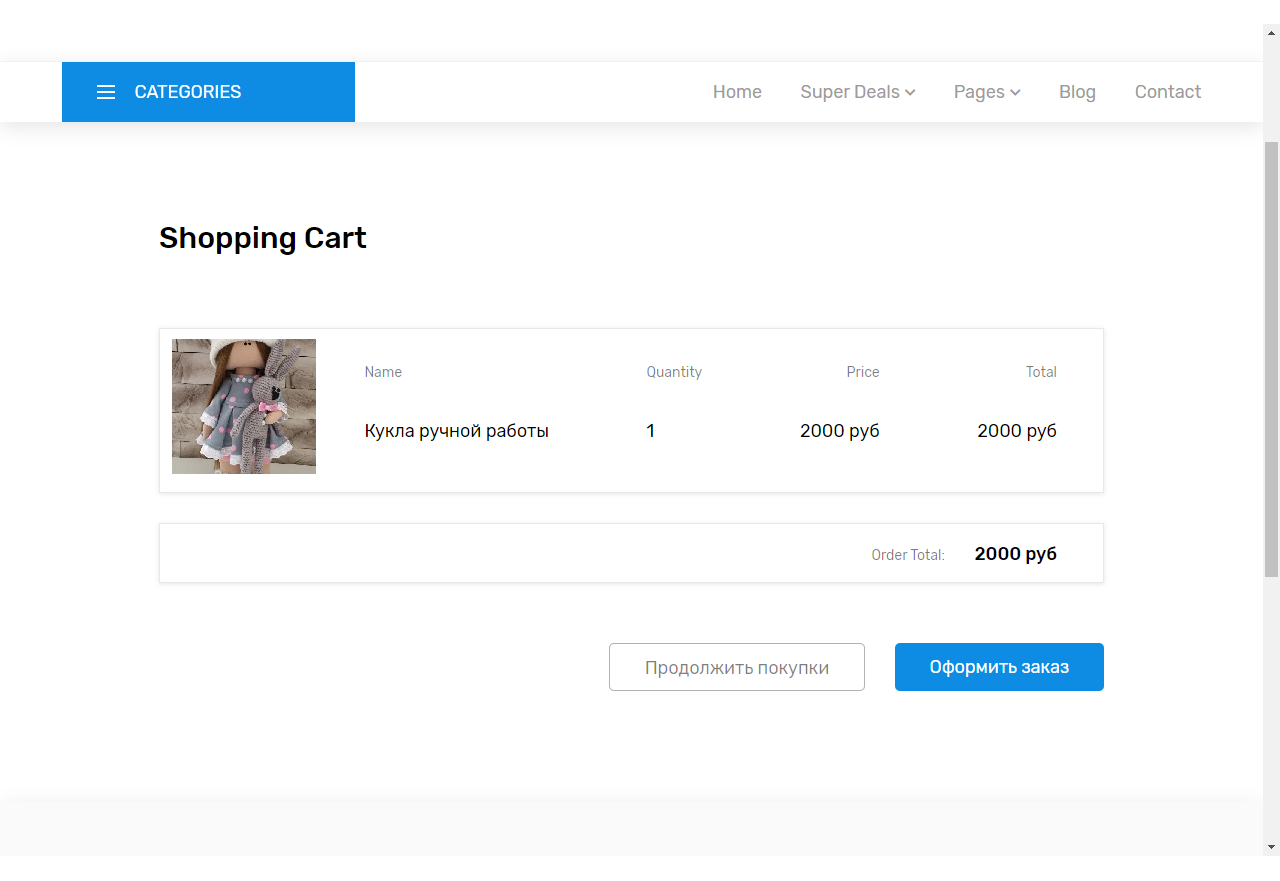


Рисунок 6 – Макет страницы корзины с добавленным товаром

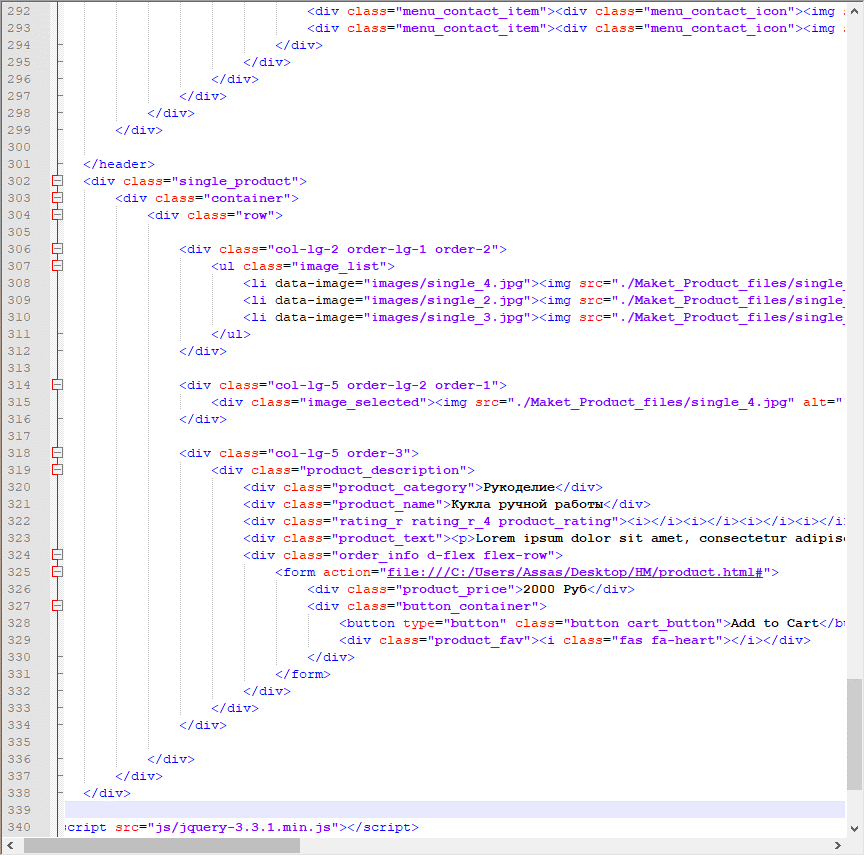


Рисунок 6 – HTML код товара

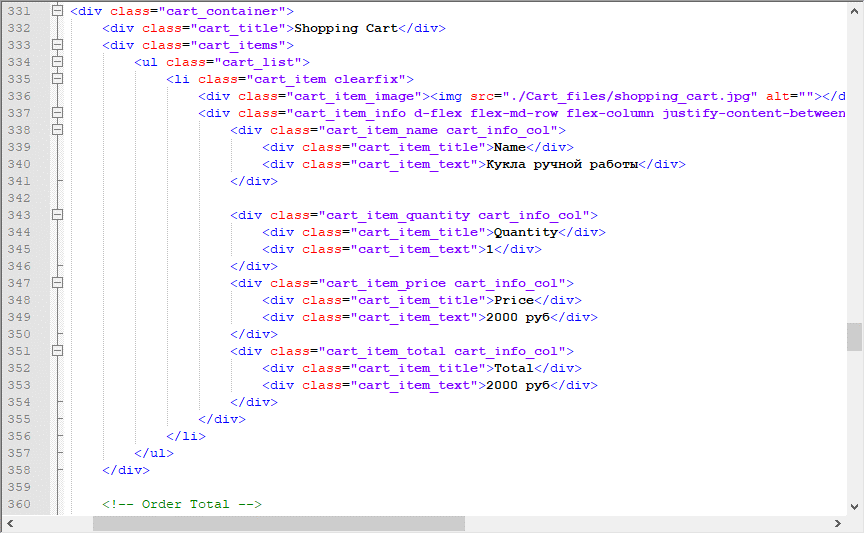


Рисунок 6 – HTML код корзины с товаром

**4.** **Проектно-технологическая часть**

# 4.1 Технологические решения, поддерживающие эксплуатационный цикл программы

При проектировании системы был использован свободный фреймворк для веб-приложений на языке Python - Django. Он реализует архитектурный паттерн Model-View-Template, который является модификацией распространённого в веб-программировании паттерна Model-View-Controller.

Архитектура Model-View-Controller – это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Концепция MVC позволяет разделить модель, представление и контроллер на три отдельных компонента:

* Модель пре доступ к данным, обрабатывается слоем работы с базой данных. Он осуществляет запросы в базу данных и проверку на корректность. Модель не зависит от представления, а следовательно, не имеет возможности визуализировать данные и не имеет точек взаимодействия с пользователем. Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом уведомляя «наблюдателей».
* Представление (View) отвечает за получение необходимых данных из модели и отправляет их пользователю.
* Контроллер обеспечивает общение между пользователем и системой. Он контролирует и направляет данные от пользователя к системе и наоборот.

Основная цель применения данной концепции состоит в отделении бизнес-логики от её визуализации. За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть одинаковые данные в различны. В частности, выполняются следующие задачи:

* К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели.
* Не затрагивая реализацию видов, можно изменить реакции на действия пользователя — для этого достаточно использовать другой контроллер;
* Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики, вообще не будут осведомлены о том, какое представление будет использоваться.

Модель данного паттерна изображена на рисунке 11.

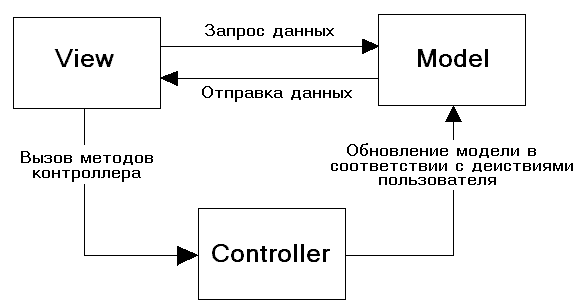


Рисунок 11 – Паттерн Model-View-Controller

Архитектура Model-View-Template незначительно отличается от Model-View-Controller. Основное различие между этими двумя шаблонами заключается в том, что Django сам обеспечивает общение между пользователем и системой, оставляя нас с шаблоном. Шаблон представляет собой файл HTML, смешанный с языком шаблонов Django.

Как и в MVC модель отвечает за бизнес-логику, методы, свойства и другие элементы, связанные с манипуляцией данными.

Представление (view) решает три задачи: принимает HTTP-запросы, реализует бизнес-логику, определённую методами и свойствами, отправляет HTTP-ответ в ответ на запросы. То есть он получает данные от модели и предоставляет шаблонам (templates) доступ к этим данным или предварительно обрабатывает данные и затем предоставляет к ним доступ шаблонам.

Шаблон (Template) — это текстовый файл, определяющий структуру или файлы с HTML-кодом, с полями для подстановки, используемыми для представления актуального содержимого. В Django реализован мощный движок шаблонов и собственный язык разметки. View может динамически создавать HTML страницы, используя HTML шаблоны и заполняя их данными из модели (model). Содержимое файлов может быть статическим или динамическим. Шаблоны не содержат бизнес-логики из-за чего они только отображают данные.

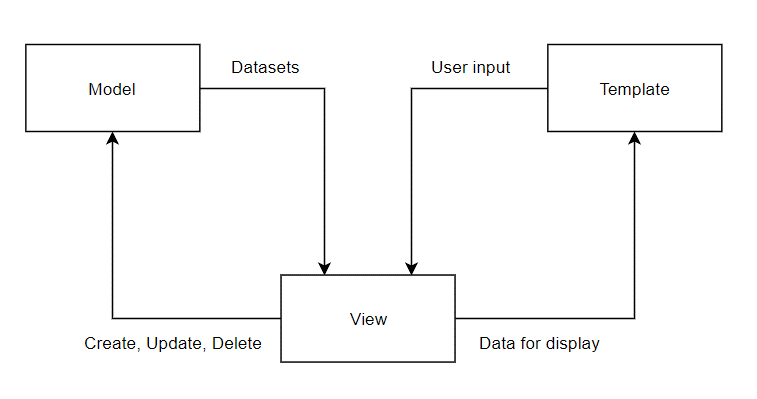


Рисунок 12 –Паттерн Model-View-Template

**Выводы**

В ходе выполнения данной курсовой работы была проанализированная выбранная предметная область, выбран инструментарий для реализации функционала веб-приложения, а также выведены UML-диаграммы сценариев работы пользователя.

Была реализована базовая часть веб-приложения, в которой была заложена функциональность для ее дальнейшего расширения. В ходе реализации веб-приложения были получены практические навыки работы с HTML, CSS, JavaScript.

Таким образом цель курсовой работы достигнута, а все поставленные задачи выполнены.

**Список источников**

1. Абрамов Г.В. Проектирование информационных систем : учебное пособие / Г.В. Абрамов, И.Е. Медведкова, Л.А. Коробова. - Воронеж : Воронежский государственный университет инженерных технологий, 2012,172 с. URL://biblioclub.ru/index.php?page=book&id=141626
2. Диков, А.В. Веб-технологии HTML и CSS / А.В. Диков. – 2-е изд. – Москва: Директ-Медиа, 2012. – 78 с.: ил.,табл., схем. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=96968, по подписке
3. Документация Django [Электронный ресурс] / djangoproject.com– Электрон. текстовые дан. – Режим доступа: https://docs.djangoproject.com/en/3.0//, свободный.
4. Документация модуля psycopg2 [Электронный ресурс] / initd.com – Электрон. текстовые дан. – Режим доступа: http://initd.org/psycopg/, свободный.
5. Документация языка Html [Электронный ресурс] / w3.org – Электрон. текстовые дан. – Режим доступа: https://www.w3.org/TR/html52/, свободный.
6. Официальная документация языка Python [Электронный ресурс] / python.org – Электрон. текстовые дан. – Режим доступа: https://docs.python.org, свободный.
7. Рудаков А. В. Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования / А. В.Рудаков, Г. Н. Федорова. — 4-е изд., стер. — М. : Издательский центр «Академия»; 2014. — 192 с. ISBN 978-5-4468-0465-8
8. Справочник по html, css, веб-дизайн [Электронный ресурс] / Htmlbook.ru– Электрон. текстовые дан. – Режим доступа: http://htmlbook.ru/html, свободный.
9. Форд Н., Хайгард М., де Ора Б. - 97 этюдов для архитекторов программных систем. - Пер. с англ. - СПб. Символ-Плюс, 2010. - 224 с., ил. ISBN 978-5-93286-176-9