**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.

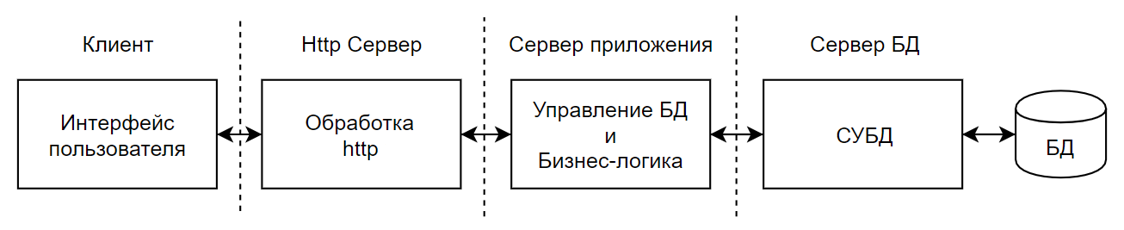


Рисунок 4 – Компоненты системы

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:

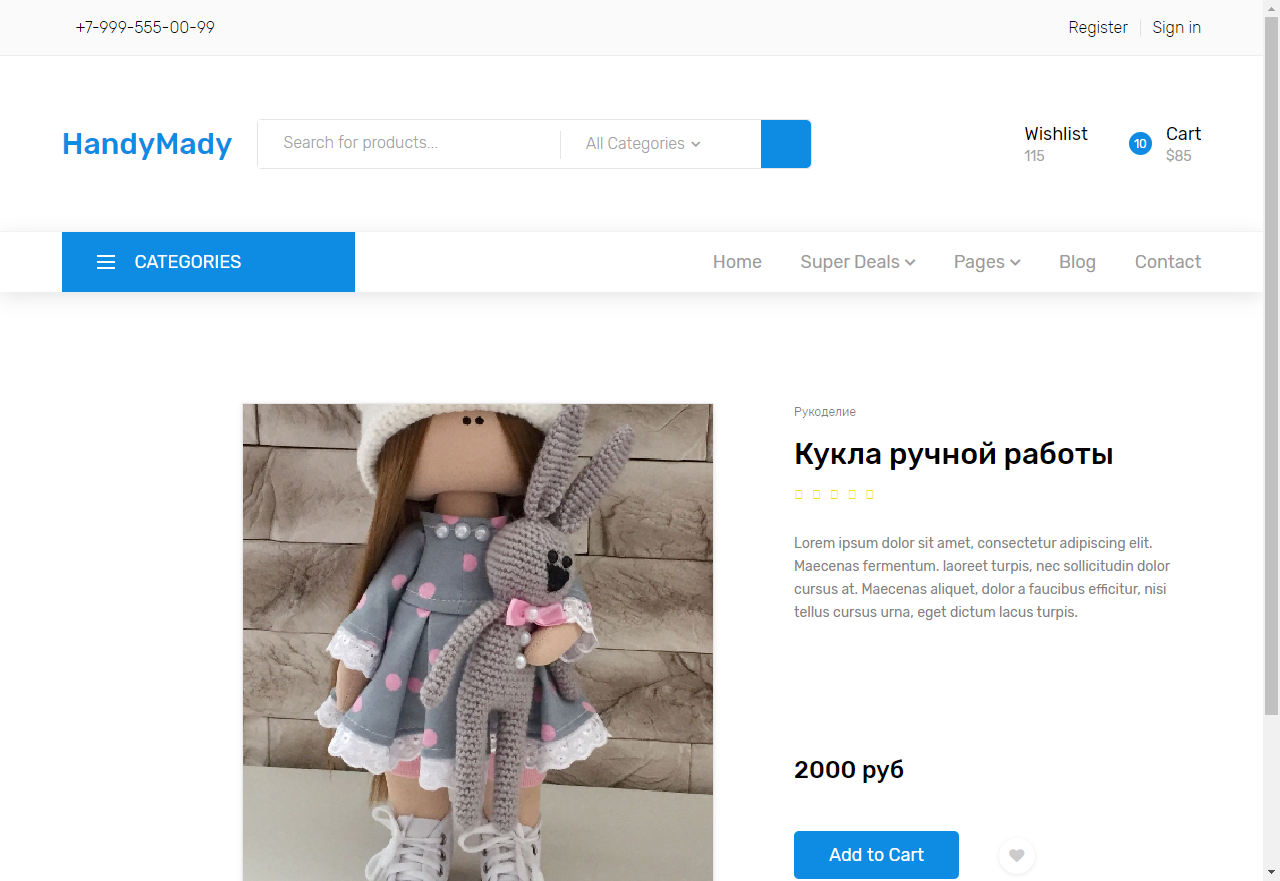
• Высокая скорость загрузки страницы

• 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

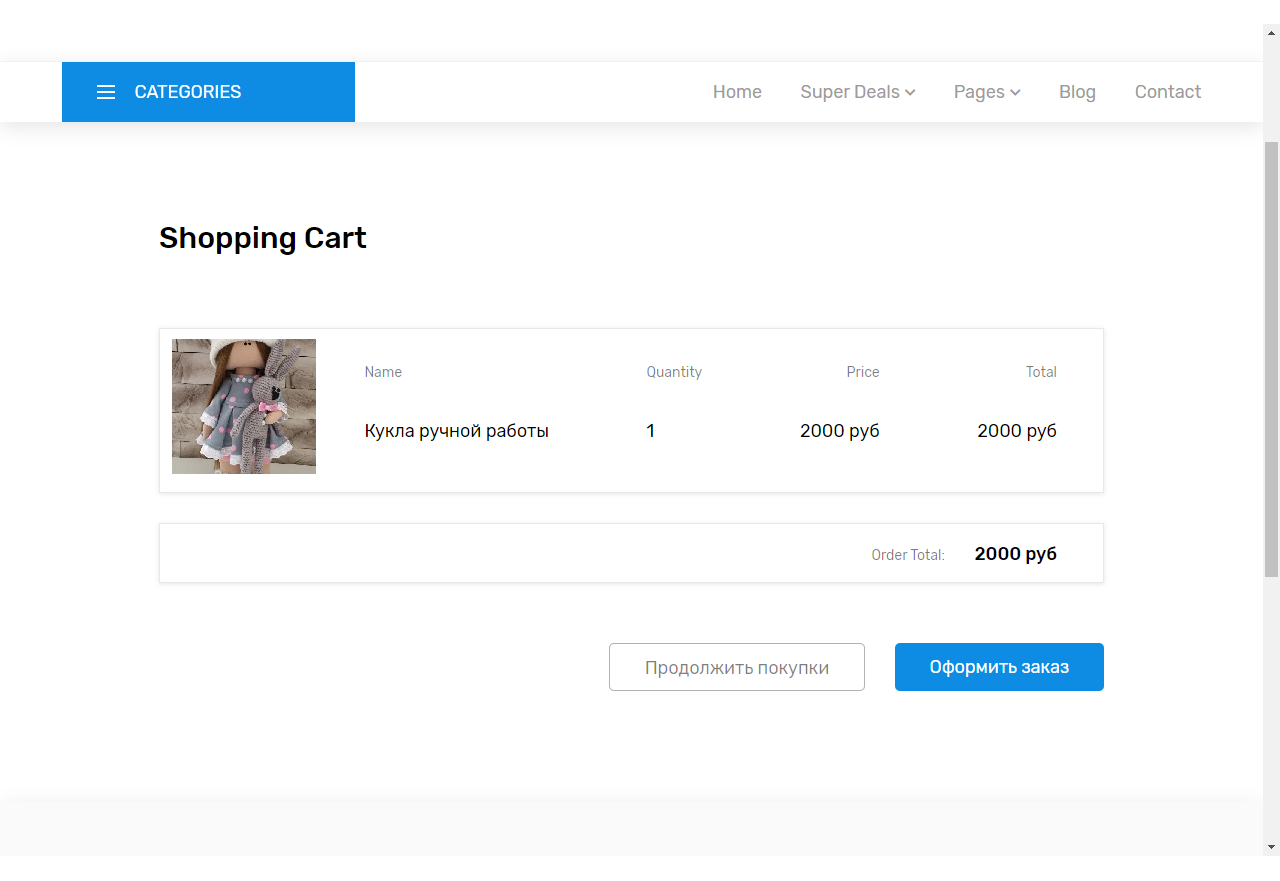


Рисунок 6 – Макет страницы корзины с добавленным товаром

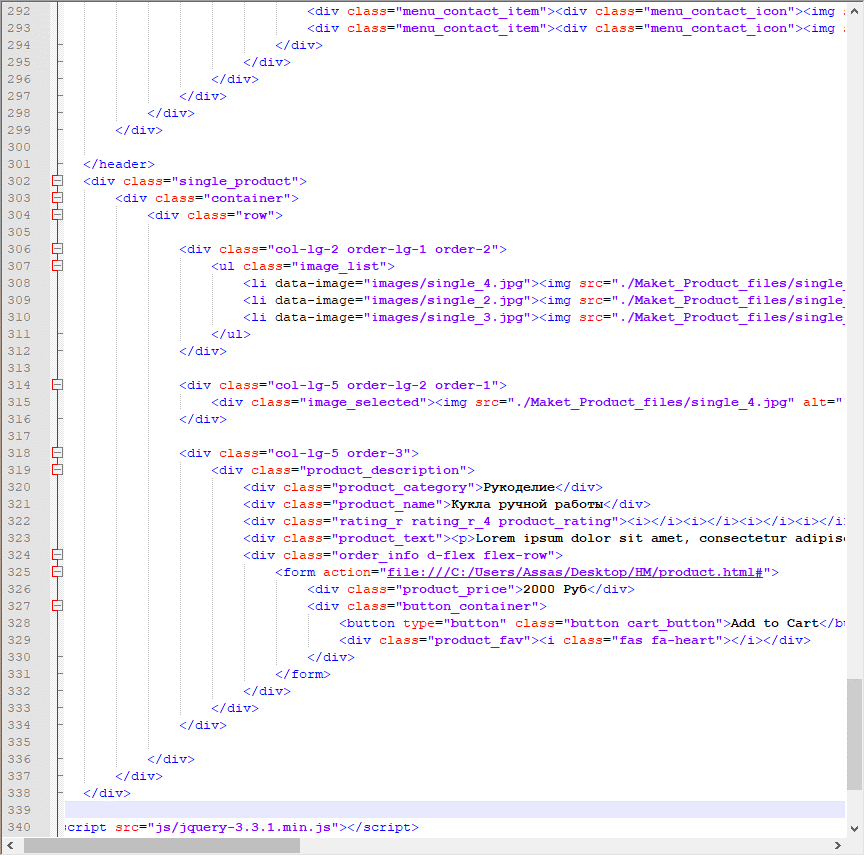


Рисунок 6 – HTML код товара

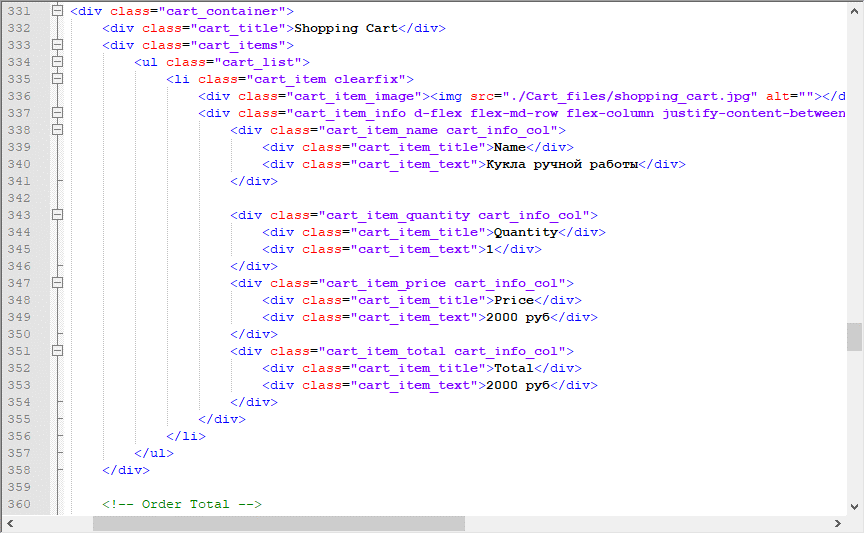


Рисунок 6 – HTML код корзины с товаром

**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.

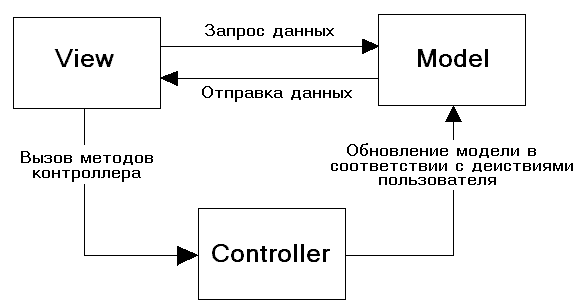


Рисунок 11 – Паттерн Model-View-Controller

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.

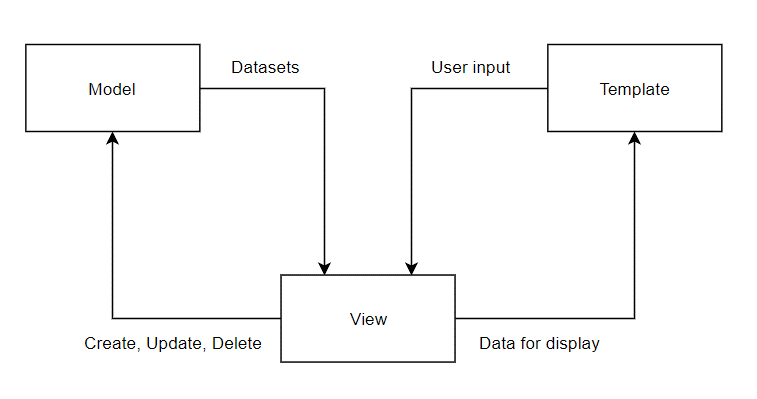


Рисунок 12 –Паттерн Model-View-Template

**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