Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления
КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Интернет-магазин мужской и женской одежды

Студент ИУ7-65Б (Группа) _____ (Подпись, дата) А. Р. Юмаев (И. О. Фамилия)

Руководитель курсового проекта _____ (Подпись, дата) Ю. М. Гаврилова (И. О. Фамилия)

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
И. В. Рудаков
(И.О.Фамилия)
« _____ » _____ 2020 г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Базы данных

Студент группы ИУ7-65Б

Юмаев Артур Русланович
(Фамилия, имя, отчество)

Тема курсового проекта Интернет-магазин мужской и женской одежды

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный
Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание Спроектировать и разработать клиент-серверное приложение для автоматизации продаж мужской и женской одежды с возможностью просмотра каталога товаров, добавления товара в корзину, изменения количества товара, изменения корзины, формирования и отправки заказа. Спроектировать и реализовать базу данных.

Оформление курсового проекта:

Расчетно-пояснительная записка на 25-30 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
на защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, математическая постановка, исполь--зованные методы и алгоритмы, расчетные соотношения, структура программы, диаграм--ма классов, диаграмма базы данных, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.

Дата выдачи задания « 24 » февраля 2020 г.

Руководитель курсового проекта _____ Ю. М. Гаврирова
(Подпись, дата) (И.О.Фамилия)

Студент _____ А. Р. Юмаев
(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение

В настоящее время покупки товаров через интернет стали обыденным явлением для современного человека. В 2017 году объем покупок в интернете достиг $2.3 трлн., который, как ожидается, увеличится до $4.5 трлн. к 2021 году. Интернет-покупатели все чаще ищут покупки за пределами своей страны, в среднем около 57% онлайн покупок совершается у зарубежных магазинов. Объем покупок в интернете в 2018 году был около $17.2 млрд. Аналитики прогнозируют рост на 44.2% к 2023 году, что должно составить $24.8 млрд. [1]. Статистика поведения покупателей в сфере электронной коммерции показывает, что 43% онлайн-покупателей сообщили, что совершают покупки из дома, 23% - в офисе и 20% - в ванной или в машине. Южная корея (77%), Германия (76%), Китай (68%), Индия (68%) и Англия (67%) лидируют по покупкам одежды через интернет [2]. Таким образом, наблюдая рост отрасли электронной коммерции, компании-производители все больше заинтересованы в продаже товаров через интернет. Некоторые компании имеют собственные отделы разработки, тестирования и поддержки интернет-магазина, другие вынуждены обращаться фирмы, предоставляющие услуги по созданию сайтов.

Современные интернет-магазины, такие как Amazon, могут иметь ассортимент, состоящий из сотен миллионов товаров. На 2020 год Amazon имеет порядка 600 млн. товаров, доступных для заказа, а ежедневное пополнение составляет около 1,3 млн. товаров. За 2017 год Amazon продал 2.7 млрд. товаров через интернет.

Для обслуживания подобного объема продаж требуется ПО, позволяющее эффективно справляться с высокой нагрузкой. Для хранения информации существуют базы данных и системы управления базами данных (СУБД) с оптимизированными алгоритмами поиска, сортировки, ранжирования и группировки данных. Например Oracle или MS SQL Server.

Цель данной работы – реализовать интернет-магазин с пользовательской корзиной, а также фунцией фильтрации и сортировки товаров.

Для достижения поставленной цели требуется решить следующие **задачи**.

1. выбрать язык программирования, фреймворк для создания веб-сервера и http сервер;
2. выбрать CSS библиотеку для разработки адаптивного интерфейса;
3. выбрать базу данных для хранения информации и API (Application Programming Interface) для выбранной базы данных;
4. спроектировать и разработать базу данных;
5. спроектировать и разработать веб-приложение с помощью выбранных технологий;
6. разработать гибкий интерфейс для созданного веб-приложения;

1. Аналитическая часть

В данном разделе будут рассмотрены подходы к созданию веб-сервисов, а также подходы к организации способа хранения данных.

1.1 Обзор подходов и методов создания веб-сервисов

На сегодняшний день существует 2 глобальных подхода к созданию веб-сервисов: монолитная архитектура и микросервисная архитектура, которые имеют ряд отличий с точки зрения хранения данных. Микросервисная архитектура приложения родилась из монолитной, когда та стала сложно поддержимаемой. Если на проекте используется монолитная архитектура, то у разработчиков есть всего одно приложение, все компоненты и модули которого работают с единой базой данных (рис. 1).

Рисунок 1. Пример реализации монолитной архитектуры веб-приложения

Микросервисная архитектура предполагает разбивку на модули, которые запускаются как отдельные процессы и могут иметь отдельные серверы. Каждый микросервис работает со своей базой данных, и все эти сервисы могут общаться между собой как синхронно (http), так и асинхронно. При этом для оптимизации архитектуры желательно минимизировать взаимосвязи между сервисами. Приведенная на рисунке 2 схема микросервисной архитектуры является упрощенной, на ней отражены лишь бизнес-компоненты.

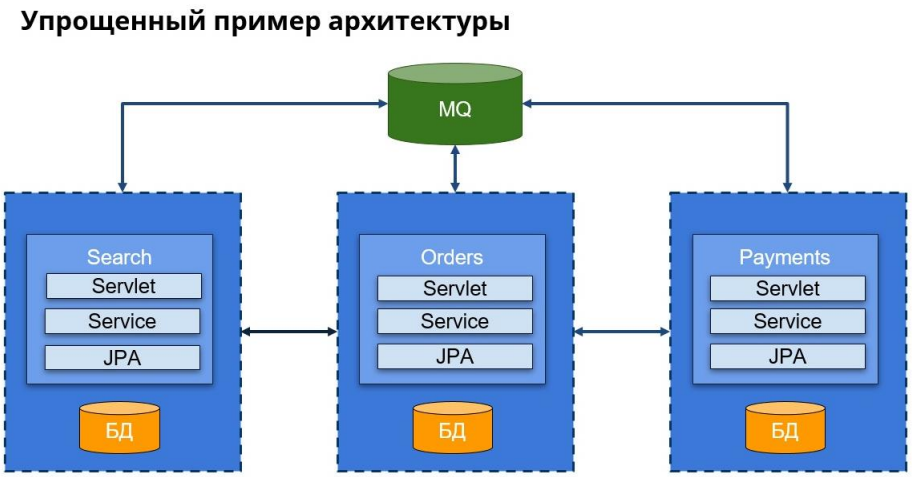

Рисунок 2. Пример реализации микросервисной архитектуры веб-приложения

Рассмотрим более подробно достоинства и недостатки каждого из подходов, а также случаи, когда стоит использовать микросервисную архитектуру, а когда монолитную.

1.2 Монолитная архитектура

Монолитное приложение представляет собой приложение, доставляемое через единое развертывание. Во время работы оно может взаимодействовать с другими службами или хранилищами данных, однако основа его поведения реализуется в собственном процессе, а все приложение обычно развертывается как один элемент. Все функции монолитного приложения или основная их часть сосредоточены в одном процессе или контейнере, который разбивается на внутренние слои (*пользовательский интерфейс, бизнес-логика, слойн данных*) или библиотеки. Недостаток этого подхода становится очевидным, когда приложение разрастается и его необходимо масштабировать. Обычно данную проблему решают разделением всего приложения на части и развитием каждой части как отдельного микросервиса.

1.3 Микросервисная архитектура

Микросервисная архитектура – это подход, при котором единое приложение строится как набор небольших сервисов, каждый из которых работает в собственном процессе и коммуницирует с остальными используя легковесные механизмы, как правило HTTP. Эти сервисы построены вокруг бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии хранения данных. Если нагрузка на сервис возрастает, его можно масштабировать, не затрагивая остальные. Это позволяет гибко и эффективно распределять нагрузку и экономить ресурсы. Также децентрализация ответственности за данные среди микросервисов оказывает влияние на то, как эти данные изменяются. Обычный подход к изменению данных заключается в использовании транзакций для гарантирования консистентности (целостности) при изменении данных, находящихся на нескольких ресурсах. Такие лидеры IT-индустрии, как Amazon, Netflix или Google активно используют данный подход, что позволяет им быстро создавать новые продукты [3].

1.3.1 Недостатки работы с данными

Если используется монолитная архитектура с единой базой данных, для построения сложного отчета пишется запрос к базе и аггрегируются несколько таблиц. Однако, на микросервисах эти данные могут быть распределены по разным базам данных. Например, нужно вывести список компаний с определенными метриками и при выводе простого списка компаний все работает. Но если возникает задача добавить метрики, которые лежат в другой базе данных, возникают сложности. Можно сделать дополнительный запрос и по ИНН запросить метрики, но если результат нужно фильтровать и сортировать, то список компаний может оказаться очень большим, и тогда придется вводить дополнительный сервис со своей базой данных — *отчеты* [4].

1.4 Особенности выбора СУБД

Основная задача базы данных в данном проекте – сохранять и выдавать данные в нужном формате, определенном программой, при этом требуется делать это быстро, так как

исследования показывают, что пользователь может ждать ответа от программы около 3 секунд, после этого у него начинает возникать дискомфорт. Также в рамках задачи становится нецелесообразно использовать внешний сервер ввиду избыточности, что является плюсом в пользу использования монолитной архитектуры. В программе не предполагается большой (от нескольких ГБ) объем данных для хранения. Следовательно, было принято решение, что данные будут храниться локально.

Вывод

Основываясь на недостатках и преимуществах двух разных подходах, стоить отметить, что эксперты советуют начинать строить веб-сервис, используя монолитную архитектуру, но четко разграничивая ее модули, которые в будущем можно будет развивать как отдельные микросервисы. Такой подход даст быстрый старт проекту и возможность протестировать продукт на рынке, имея минимально работающий функционал. Таким образом данный проект будет реализован на монолитной архитектуре.

2. Конструкторская часть

2.1 Архитектура веб-приложения

Так как для разработки архитектуры веб-приложения использовался *монолитный* подход, проект будет состоять из нескольких слоев, которые в свою очередь будут содержать компоненты и модули. На рисунке 3 представлена архитектура веб-приложения.

Рисунок 3. Архитектура разрабатываемого веб-приложения

На рисунке 4 представлена Use-Case диаграмма проекта. Некоторые части, а именно оплатата заказа не являются приоритетной задачей пректа и могут быть реализованы через сторонние сервисы.

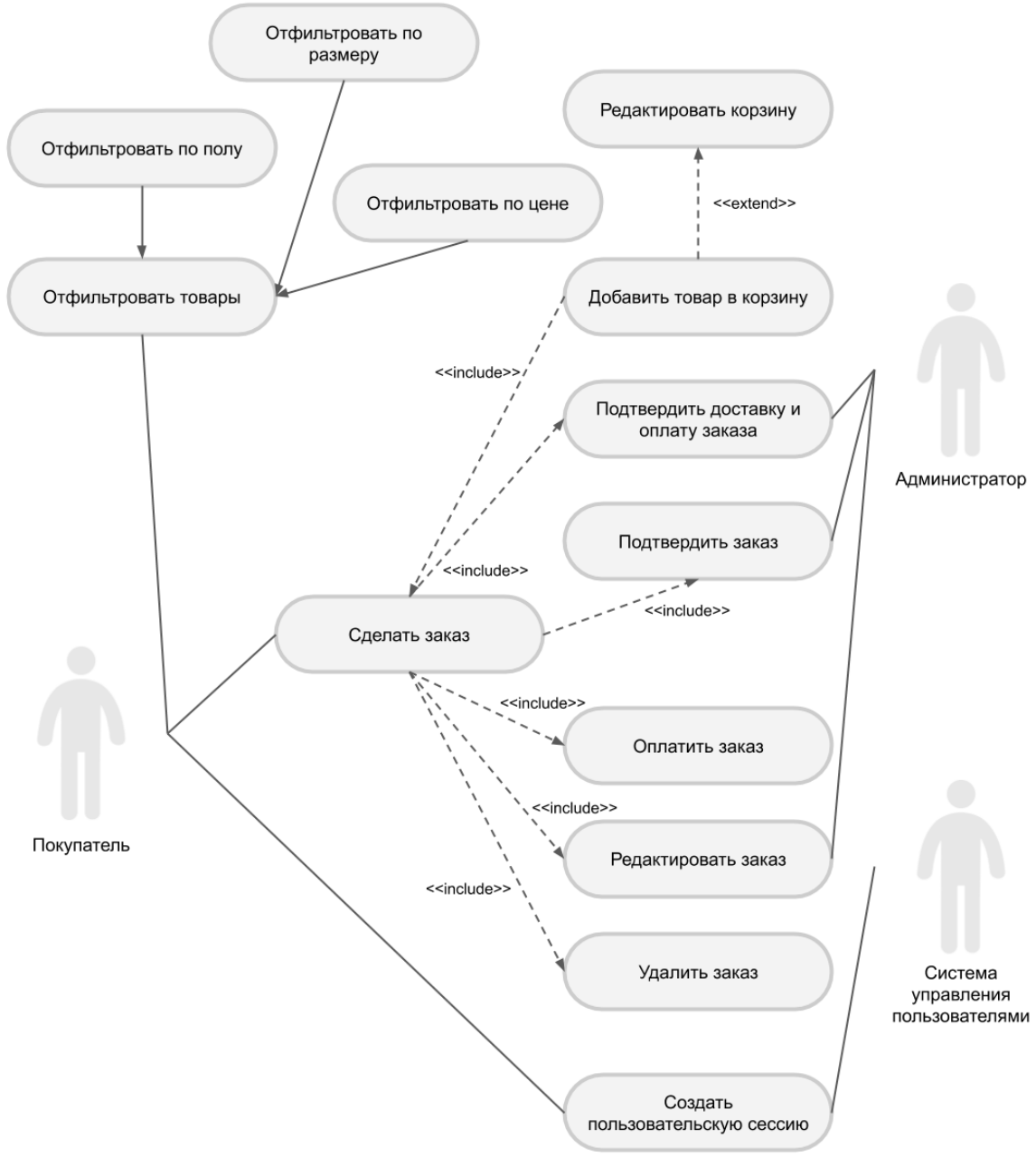

Рисунок 4. Use-Case диаграмма проекта

Пользователь взаимодействует с интерфейсом, который отправляет запросы на слой бизнес-логики, где они обрабатывают. Общение происходит `http` запросами `GET` или `POST`. Предполагается идентификация каждого конкретного пользователя с помощью `cookies`. Обработка пользовательских запросов представлена на рисунках 4-5. На рисунке 4 представлена обработка `GET` запросов пользователя на получение веб-страницы, на рисунке 5 обработка запросов на добавление заказов в корзину, на рисунке 6 обработка запроса на создание нового заказа.

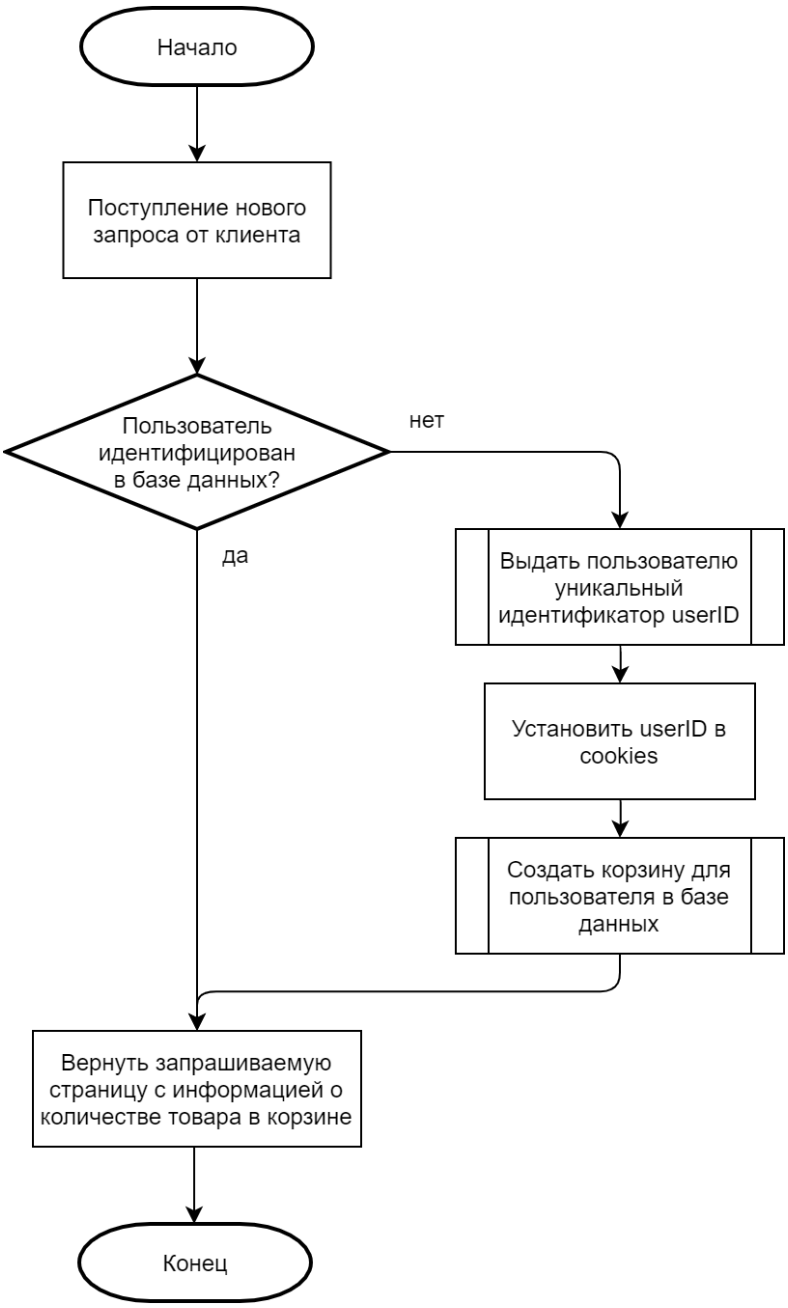

Рисунок 5. Обработка запроса пользователя на получение веб-страницы

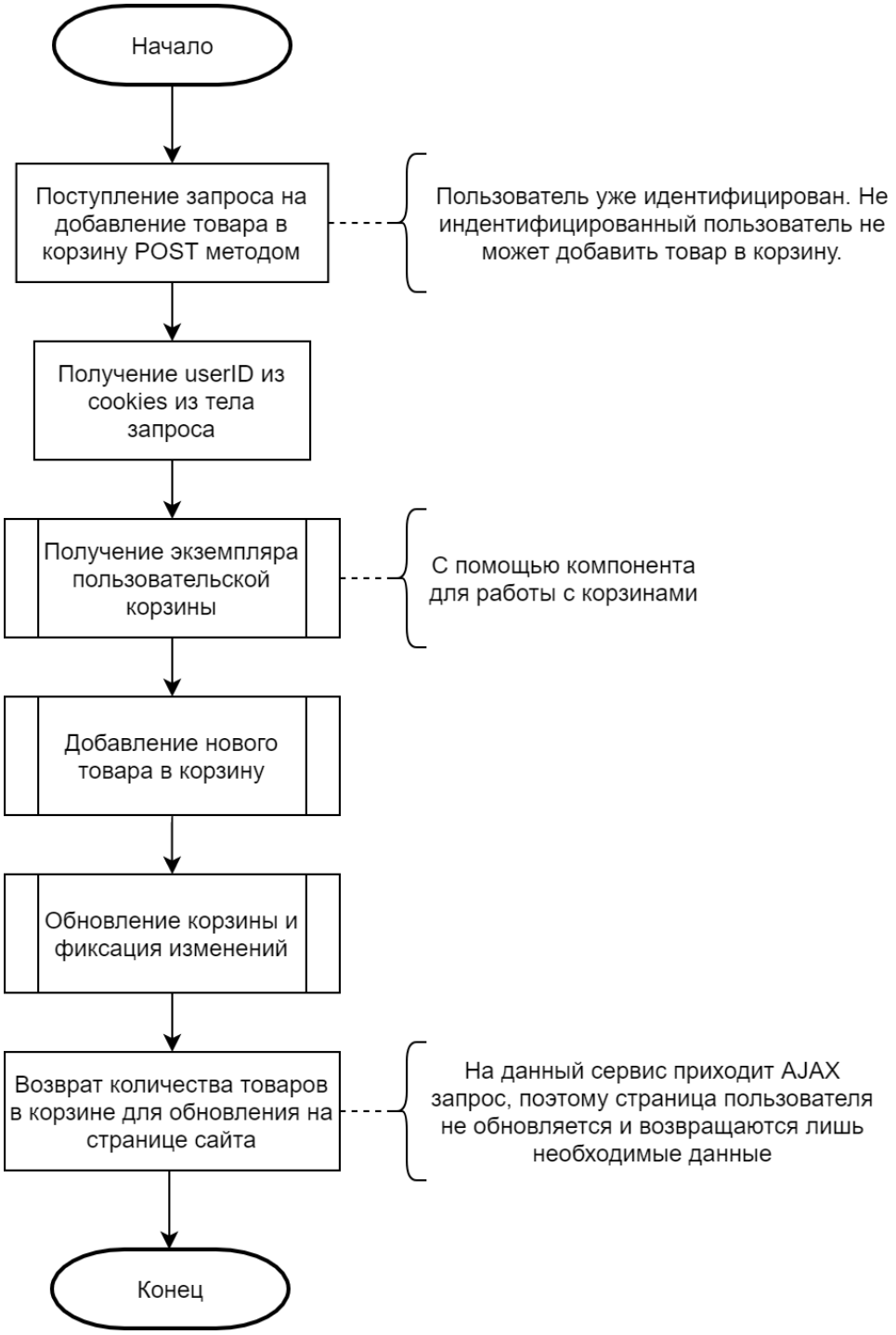

Рисунок 6. Обработка запроса на добавление товара в корзину

Рисунок 7. Обработка запроса на создание нового заказа

Далее на рисунках 7-9 представлены UML диаграммы компонентов всего приложения. На рисунке 7 представлена UML диаграмма компонента доступа к данным.

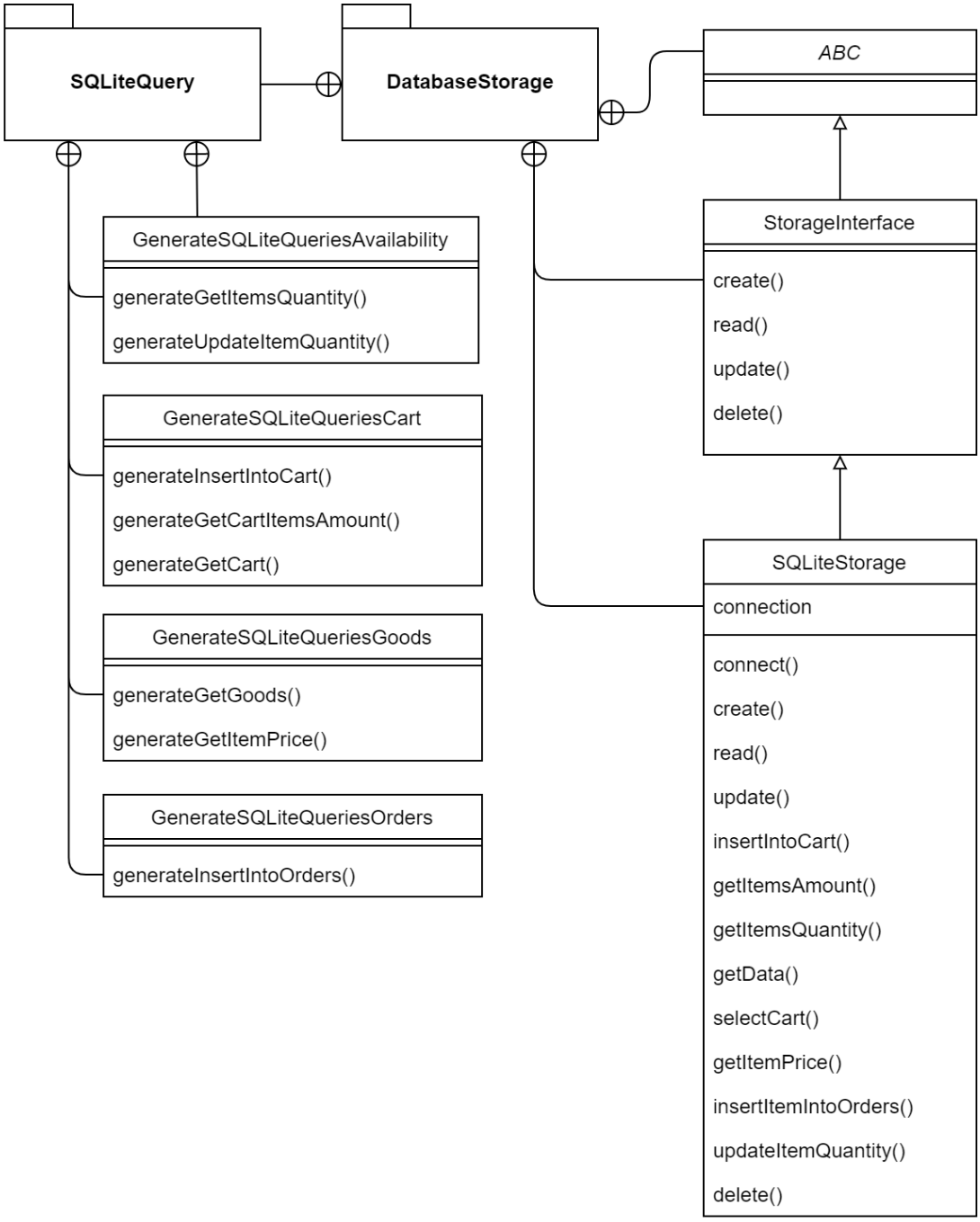

Рисунок 8. UML диаграмма компонента доступа к данным

На рисунке 8 представлена UML диаграмма бизнес-логики, которая состоит из 3 компонентов: компонента корзины (Cart), компонента склада (Warehouse) и компонента работы с заявками на покупку (Orders). На рисунке 9 представлены UML диаграммы вспомогательных элементов бизнес-логики: модуль создания подключения к базе данных (Connection) и модуль чтения конфигурационных файлов (ConfigManager).

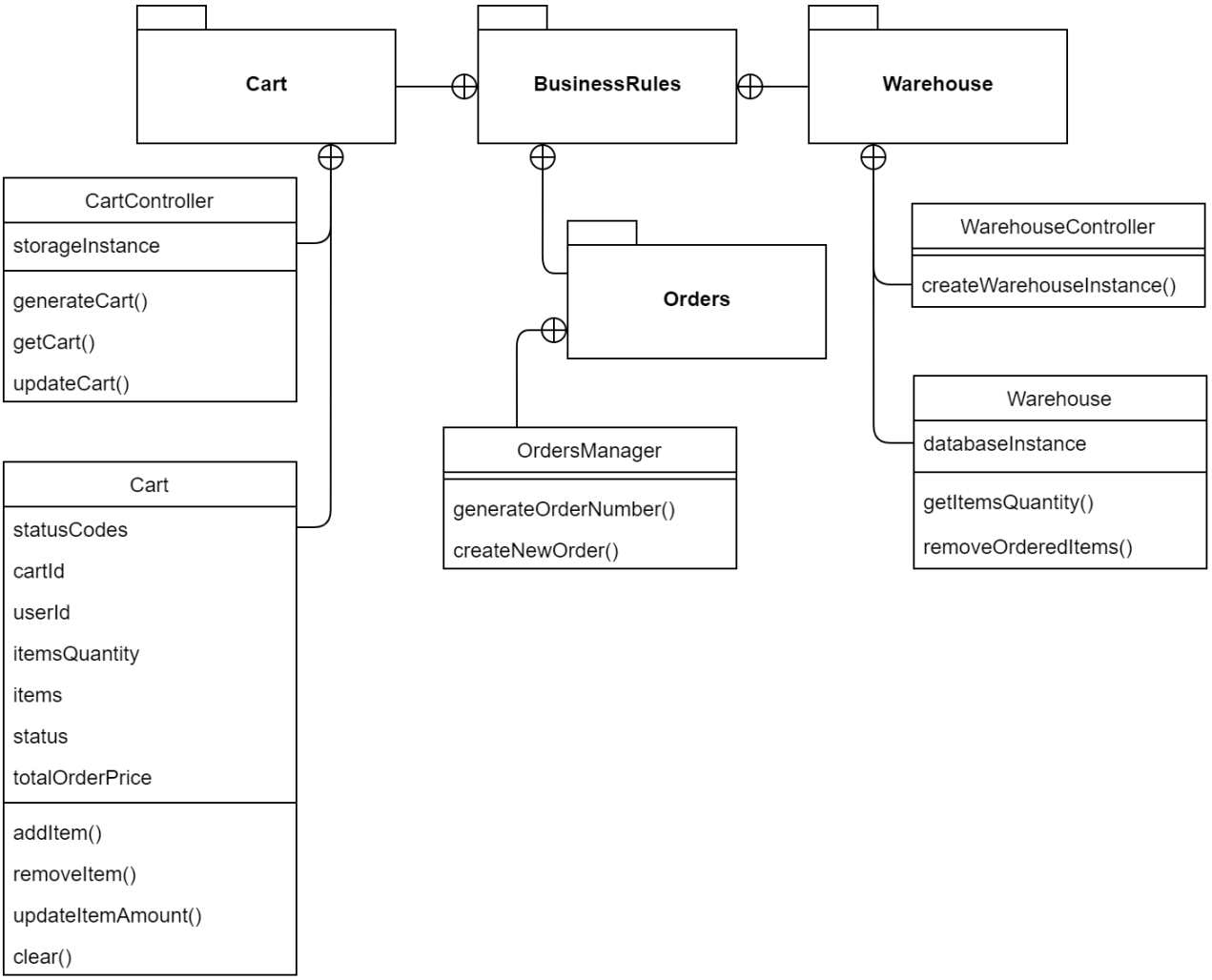

Рисунок 9. UML диаграмма бизнес-логики приложения

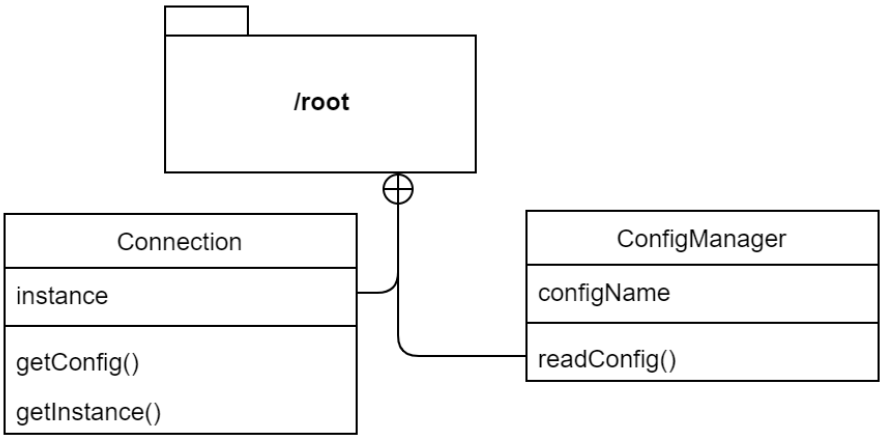

Рисунок 10. UML диаграммы менеджера конфигурации и соединения с базой данных

На рисунке 10 представлена схема базы данных. База данных состоит из 3 таблиц: таблица с товарами, таблица с информацией о наличии товаров и таблица с заказами. На рисунке 11 показана ER-диаграмма итогового проекта.

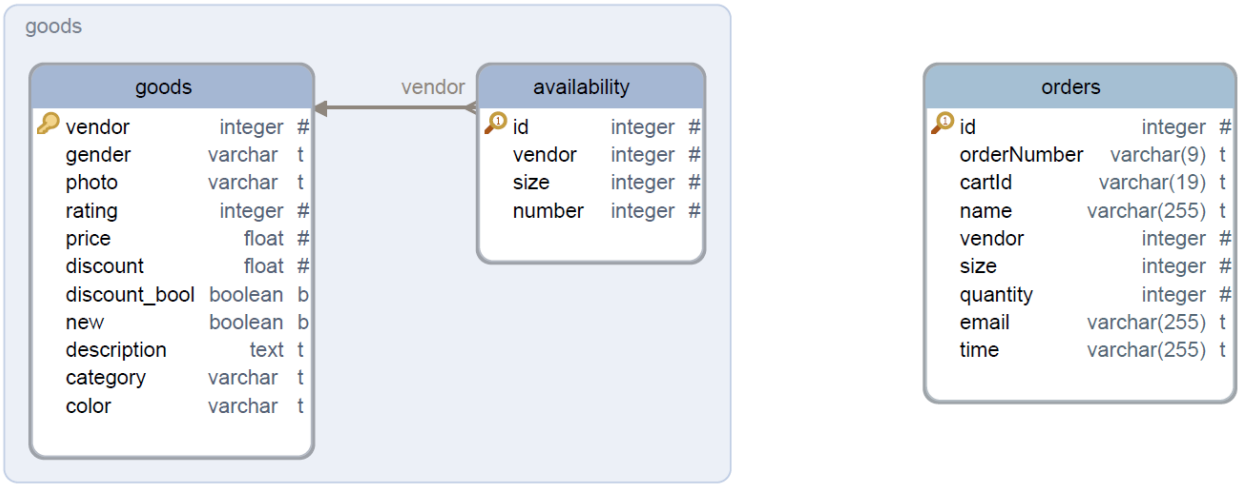

Рисунок 11. Схема базы данных

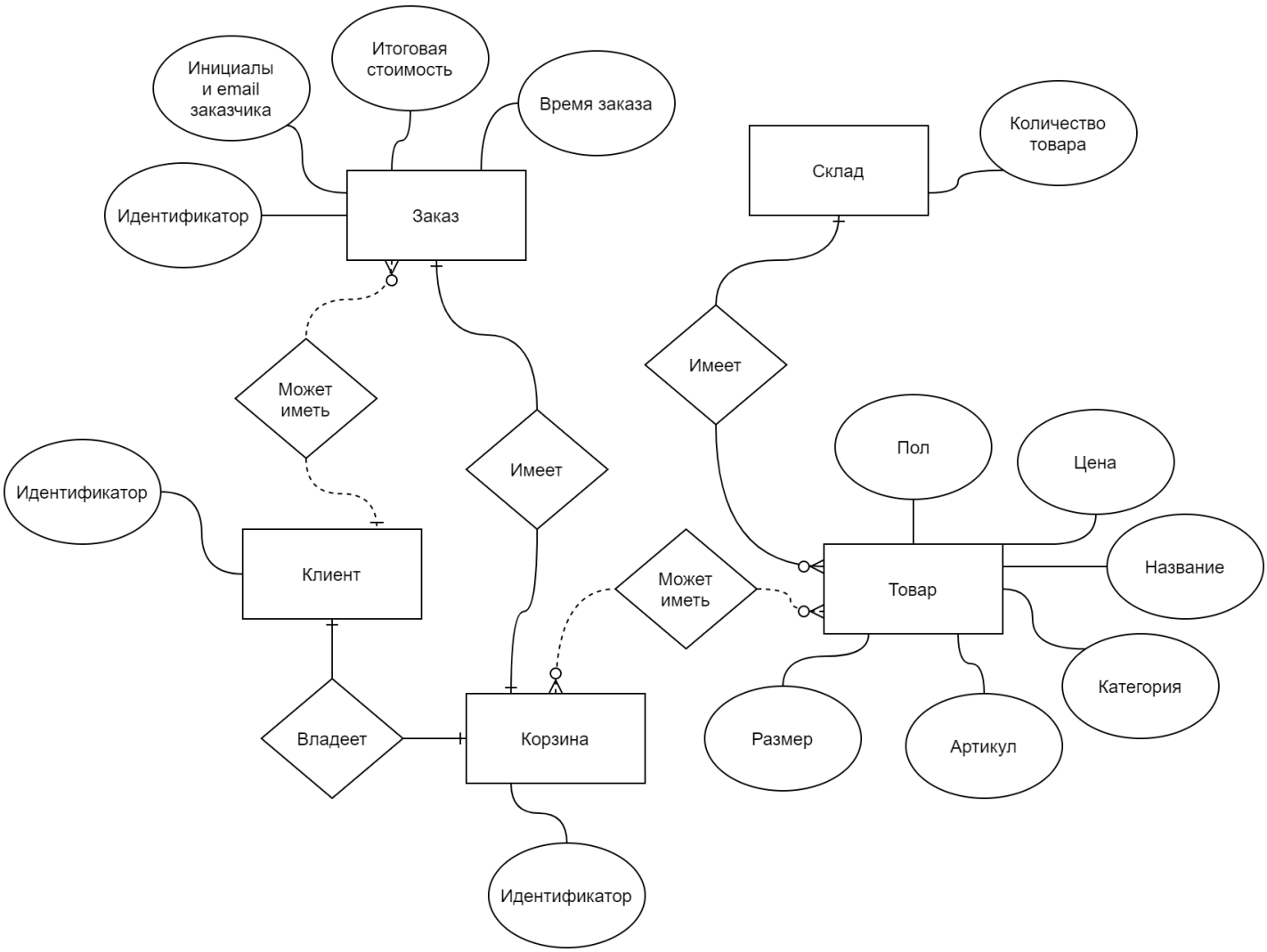

Рисунок 12. ER-диаграмма проекта

Вывод

В данном разделе была представлена архитектура проекта. Разработан основые части проекта: слой пользовательского интерфейса, слой бизнес-логики, слой доступа к данным, слой данных. Разработан Use-Case план взаимодействия основных частей проекта. Приведены UML диаграммы компонентов и модулей проекта, схема базы данных и ER-диаграмма сущностей, составляющих архитектуру приложения.

3. Технологическая часть

В данном разделе будет проведена работа по анализу существующих решений для реализации поставленной задачи, а именно создания веб-сайта для электронной коммерции.

3.1 Выбор языка программирования и веб-фреймворка

На сегодняшний день существует несколько основных технологических стеков для реализации проектов в сфере веб. В них входит язык программирования Haskell (веб-фреймворки: Yesod, Snap, Misso), JavaScript (веб-фреймворки Node.Js, React), Java (веб-фреймворк Spring), PHP (веб-фреймворк Larabel) а также Python в связке с фреймворками Django, Flask, либо, если требуется поддержка асинхронности на веб-сервере, Tornado или Sanic. Разрабатываемый в учебных целях интернет-магазин не предполагает повышенной нагрузки и сложной структуры веб-страниц, поэтому подходящим решением будет микрофреймворк, представляющий из себя минималистичный каркас веб-приложения, предоставляющий лишь самые базовые возможности.

Было принято решение использовать данный язык программирования в связке с Flask, так как он подходит для учебных целей и быстрой разработки простых веб-сервисов.

В качестве http сервера сравнивались Apache HTTP Server и Nginx. Nginx был создан для решения так называемой проблемы c10k – проблемы 10 тысяч соединений, что означает, что веб-сервера, использующие потоки не могут обрабатывать запросы пользователей более, чем с 10,000 подключений одновременно. Так как проект не предполагает высокой нагрузки, выбор был сделан в пользу Apache, он имеет более простую настройку и быстрой порог входа в технологию. 46% веб сайтов в сети Интернет используют данный http сервер, который также является бесплатным, даже в коммерческих целях.

3.2 Выбор СУБД

Как уже было описано в аналитическом разделе, для данной задачи подойдут встраиваемые бессерверные (serverless) СУБД. Встраиваемая СУБД – архитектура систем управления базами данных, когда СУБД тесно связана с прикладной программой и работает

на том же сервере, не требуя профессионального администрирования. Они имеют несколько следующих отличий от классических клиент-серверных.

- высокая скорость и малое потребление памяти, благодаря упрощенному API;
- локальное хранение данных, фактически СУБД является библиотекой, работающей с файловой системой ОС;
- расчет на однопользовательскую работу с небольшим (по меркам клиент-серверных СУБД) объемом данных.

Примеры встраиваемых СУБД:

- InfinityDB
- SQLite
- Microsoft SQL Server Compact

Некоторые встраиваемые СУБД также являются платными, хотя и стоят значительно дешевле, чем серверные платформы. Ключевым фактором в ценообразовании могут стать не только качественные характеристики продукта, но и дополнительные возможности, предоставляемые конечному пользователю. Например, графическая среда моделирования структуры данных, визуальные конструкторы запросов и т.д. Эффективное проектирование базы данных и ее администрирование считается трудоемкой, требующей высокой квалификации работой. Была выбрана СУБД SQLite, за наличие нативного API для Python и соблюдения целостности данных (консистентности).

Механизм, отвечающий за фиксацию изменений и сохранение консистентности данных называется журналированием. Если журналирование включено (поведение по умолчанию), база перед записью данных запишет сначала изменения в специальный файл журнала, чтобы в случае сбоя восстановить их оттуда. При этом, SQLite делает в специальных критичных местах паузы между атомарными операциями для того, чтобы в случае, если сбой произойдет во время записи в журнал, данные не остались в «частично зафиксированном» виде.

Поскольку SQLite - компактная встраиваемая база данных, она не поддерживает некоторые возможности других более крупных СУБД и предоставляет не полный (хотя достаточно обширный) набор команд SQL-92.

При анализе возможностей SQLite3 нужно было принять во внимание специфические недостатки такой базы данных.

- нельзя удалить или изменить столбец в таблице;
- есть поддержка триггеров, но не настолько мощные как у других крупных СУБД;
- есть поддержка foreign key, но по умолчанию она отключена;
- нет хранимых процедур;
- тип столбца не определяет тип хранимого значения в этом поле записи, то есть в любой столбец можно занести любое значение.

3.3 Выбор инструментов для разработки гибкого интерфейса

В настоящее время в мире высока популярность различных персональных компьютеров. Планшеты, смартфоны, умные часы с выходом в интернет, умные телевизоры и так далее. Современная реалии требуют от веб-сайта простоты и удобства использования на устройствах с различным размером и формой экрана. Именно поэтому так важно, чтобы сайт был способен автоматически меняться в зависимости от размера и пропорций дисплея. Ассортимент инструментов для разработки гибких и адаптивных интерфейсов на данный момент не велик. Еще несколько лет назад веб-разработчики придерживались схемы так называемой блочной верстки, когда проектирование веб страницы происходило таблицами, которые в свою очередь могли в каждой своей ячейке содержать другие таблицы. На данный момент разработчики мыслят гибкими блоками, которые с помощью JavaScript кода отпределяют параметры дисплея и подстраивают контент веб-страницы по него.

Самым популярной CSS библиотекой на данный момент остается Bootstrap от компании Twitter. Она и была выбрана для разработки проекта.

3.4 Технологический стек

Развертывание проекта будет происходить на облачной платформе Microsoft Azure. Она предоставляет возможность разработки, выполнения приложений и хранения данных на серверах, расположенных в распределённых дата-центрах. В качестве виртуальной машины выбрана Ubuntu версии 16.04.

В итоге технологический стек будет состоять из двух частей: серверной (бэкенд) и клиентской (фронтенд).

Бэкенд

- Язык программирования Python
- Фреймворк Flask для быстрой реализации простых одностраничных веб-приложений
- Apache HTTP-сервер – кроссплатформенный веб сервер для обработки соединений с юзер-агентом
- SQLite реляционная база данных
- Microsoft Azure – облачный сервер с Ubuntu 16.04
- In-memory база данных Redis для хранения пользовательских корзин

Фронтенд

- Верстка макета с помощью HTML, CSS, JavaScript
- Twitter Bootstrap для быстрой и адаптивной разработки интерфейсов

3.5 Настройка веб-сервера Apache на удаленной машине

Для настройки веб-сервера Apache на удаленной виртуальной машине, например на сервисе Microsoft Azure, требуется запустить скрипт `deploy.sh`, который пропишет необходимую конфигурацию и установит необходимые пакеты для развертывания. Скрипт развертывания приложения представлен на листинге 1. Данный скрипт скачивает веб-сервер Apache, для того, чтобы веб-приложение могло принимать входящие соединения с user-агентами, создает необходимые файлы конфигурации Apache, устанавливает конфигурацию Firewall и запускает итоговый веб-сервер.

Листинг 1. Скрипт развертывания приложения `deploy.sh`

```
sudo apt update
sudo apt install apache2 # Установить Apache
sudo ufw app list
sudo ufw allow 'Apache' # Установить конфигурацию Firewall
sudo systemctl status apache2 # Установить конфигурацию Apache

# Скачать и включить mod_wsgi
sudo apt-get install libapache2-mod-wsgi python-dev

# Переместить исходный код проекта в новую директорию
sudo cp -R ./webApp  /var/www/webApp/

# Перенести конфигурацинный файл
sudo cp ./webApp.conf  /etc/apache2/sites-available/webApp.conf

sudo a2ensite webApp
systemctl reload apache2 # Перезагрузить Apache
cd /var/www/webApp
sudo service apache2 restart # Перезапустить Apache
```

3.6 Сборка и запуск проекта

Так как проект реализован на языке программирования Python, предусмотрена работа с зависимостями, принятая стандартом для данного языка. А именно проект запускает в виртуальном окружении, все необходимые зависимости и пакеты помещаются в конфигурационный файл `requirements.txt` в корне проекта, с помощью которого скачиваются необходимые для работы проекта пакеты командой:

`pip install -r requirements.txt`.

Далее сам проекта запускает скриптом `start.sh [dev|prod]` , где `dev` – ключ запуска проекта на *development* сервере с возможностью отлдаки, а `prod` – ключ запуска проекта на *production* сервере. Скрипт запуска проекта создает конфигурационные файлы и управляет запуском веб-сервера *Apache*. На листинге 1 приведен код запуска проекта на Bash, а на листинге 2 пример конфигурационного файла базы данных `config.ini`.

Листинг 2. Скрипт запуска проекта

```
# $1 – Режим развертывания
rm config.ini
./redisTools.sh start # Запустить сервер Redis
if [ "$1" = "prod" ]
then
  # Создать конфигурационный файлы для базы данных
  echo DATABASE=sqlite3> config.ini
  echo PATH=/var/www/webApp/webApp/database/catalog.db>> config.ini

  # Установить полные права на чтение и запись для Apache
  sudo chmod -R 777 /var/www/webApp/

  # Отключить конфигурацию сайта, если он уже был запущен
  sudo a2dissite webApp
  sudo systemctl reload apache2

  # Подключить конфигурацию сайта
  sudo a2ensite webApp
  sudo systemctl reload apache2

  # Перезапустить Apache
  sudo service apache2 restart
elif [ "$1" = "dev" ]
then
  echo DATABASE=sqlite3> config.ini
  echo PATH=./database/catalog.db>> config.ini

  # Запустить основное приложение
  python3 app.py
else
  echo "No keywords found!"
  echo "Use ./start [prod|dev]"
fi
./redisTools.sh stop # Остановить сервер Redis после остановки проекта
```

Листинг 3. Пример конфигурационного файла для базы данных SQLite

```
DATABASE=sqlite3
PATH=./database/catalog.db
```

3.7 Пользование программой

Веб-сервис доступен по адресу http://52.170.255.31. Сайт состоит из 3 основных веб-страниц: главная страница, страница с каталогом и страница с корзиной. На рисунке представлена главная страница сайта. В шапке сайта есть навигационное меню, кнопка **Home** является ссылкой на главную страницу сайта. Кнока **Collection** является ссылкой на страницу с каталогом сайта. Кнопка **Cart** является ссылкой на страницу с корзиной, рядом с кнопкой Cart также есть индикатор того, сколько товара в данный момент находится в корзине. По середине страницы представлена так называемая “карусель” баннеров, которые могут информировать пользователя о новых акциях или скидках на товары.

Рисунок 13. Главная страница сервиса

Ниже на рисунке 14 показана страница коллекции. Слева представлен фильтр категории товара: мужской, женской, либо и той и другой. Чуть высше каталога товаров представлен фильтр по категориям товаров: *аксессуары, верхняя одежда, брюки, обувь, футболки* (рис. 15). Можно выбрать сразу несколько категорий. Слева от фильтра категорий товаров есть возможность отсортировать товар по цене по возрастанию, убыванию и по популярности товара (сколько раз товар был заказан). После того как все необходимые фильтры были выбраны, нужно нажать кнопку **Apply filters**. В описании товара можно увидеть само название товара, цену (на товар может быть скидка), список размеров с информацией о том, сколько единиц размера данного товара содержится на складе и кнопку **To cart** (если хотя бы один экземпляр хотя бы одного размера есть на складе), иначе товар нет возможности заказать и кнопка **Item is over** будет недоступна для нажатия (рис. 16).

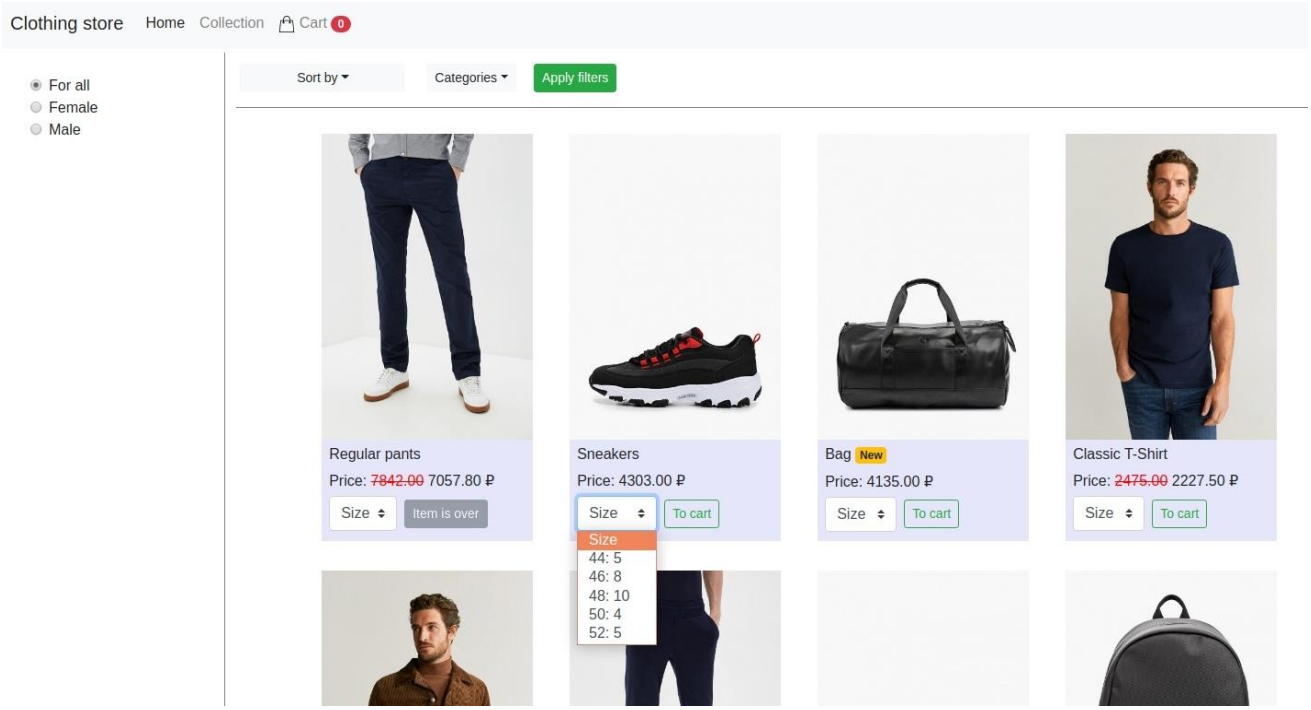

Рисунок 14. Страница каталога

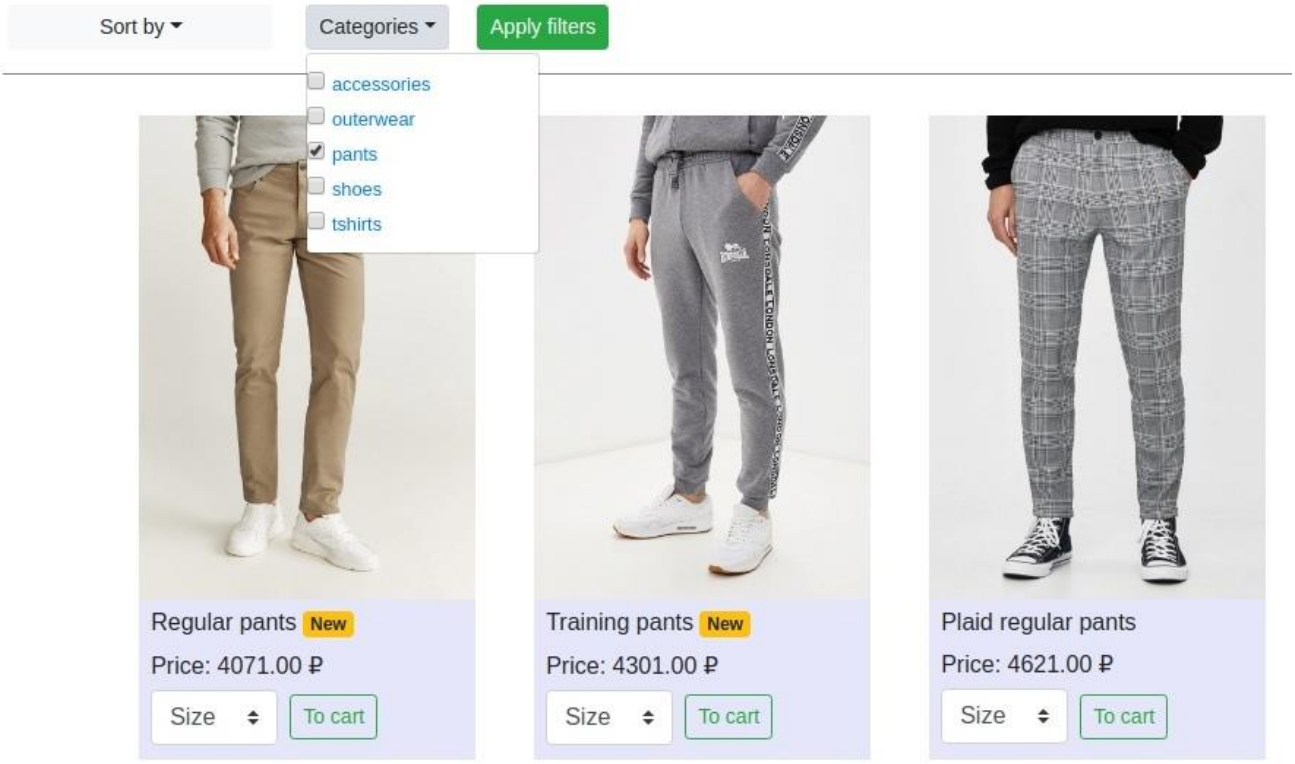

Рисунок 15. Фильтр по категориям и сортировка товаров

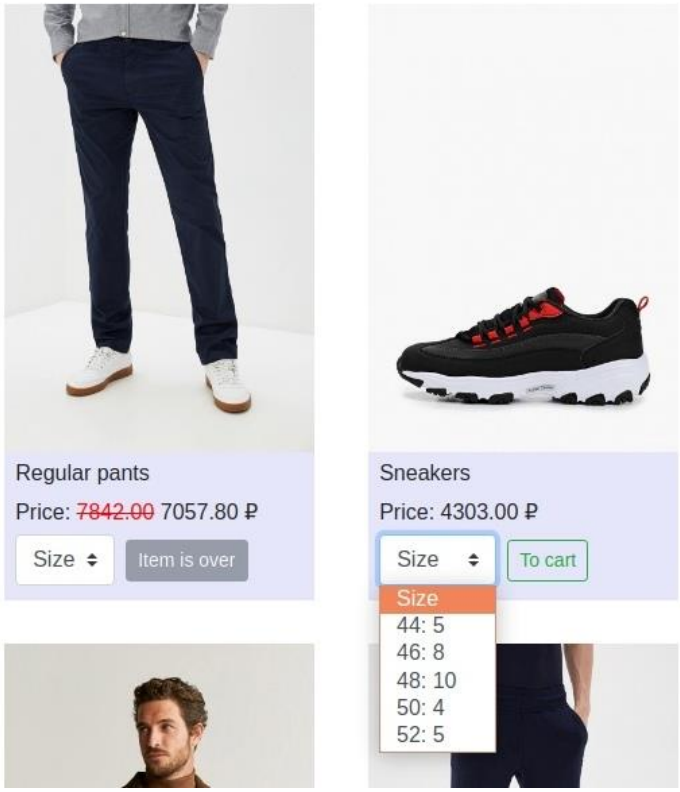

Рисунок 16. Информация о количестве товара на складе (обувь справа). Товар недоступен для заказа, есть его нет на складе (брюки слева)

На рисунке 17 представлена страница корзины. На данной странице можно увидеть список товаров, которые были добавлены в корзину. Итоговую стоимость заказа и количества товара на складе.

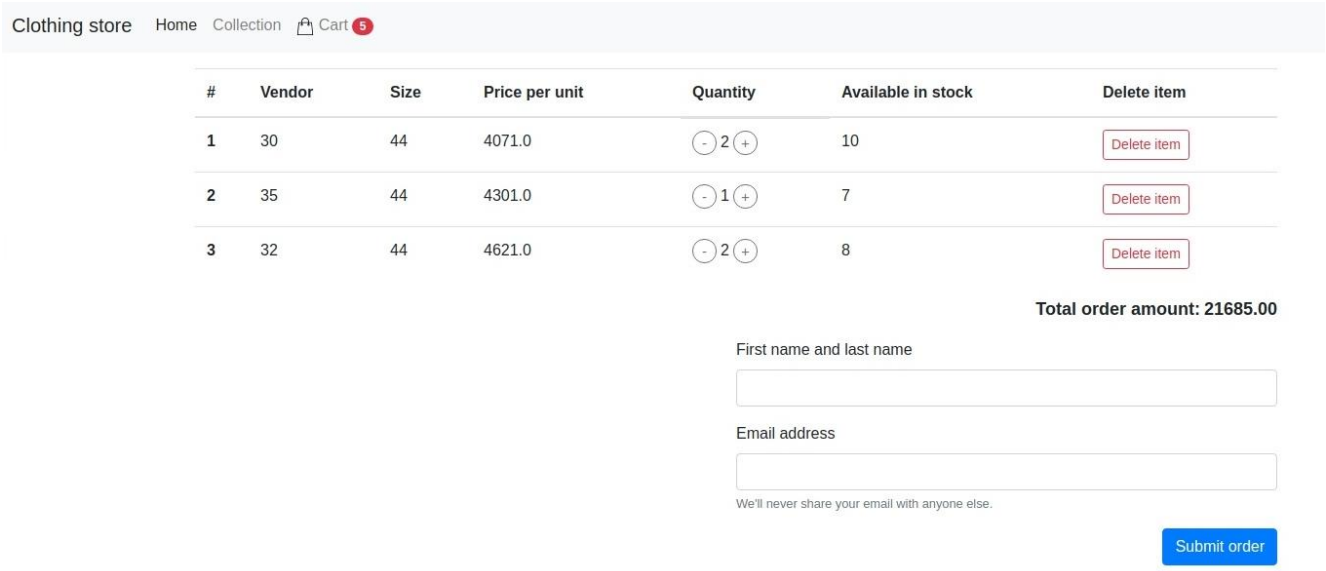

Рисунок 17. Страница корзины

Также на страницы корзины можно удалить товар – красная кнопка **Delete** (Удалить) справа на каждой строчке с товаров. Можно увеличить количество единиц товара в корзине, за это отвечают кнопки “-” и “+” рядом с информацией о количестве товара в корзине, но нельзя заказать товара больше, чем в данный момент хранится на складе. При попытке заказать больше товара, пользователю будет выведена информация об этом (рис. 18). Также под списком товаров есть форма, которую заполняет пользователь для того, чтобы передать личные данные администратору, который свяжется с ним по указанной почте для подтверждения заказа и выбора способа оплаты. Пользователь заполняет форму и нажимает кнопку **Submit order** (Подтвердить заказ).

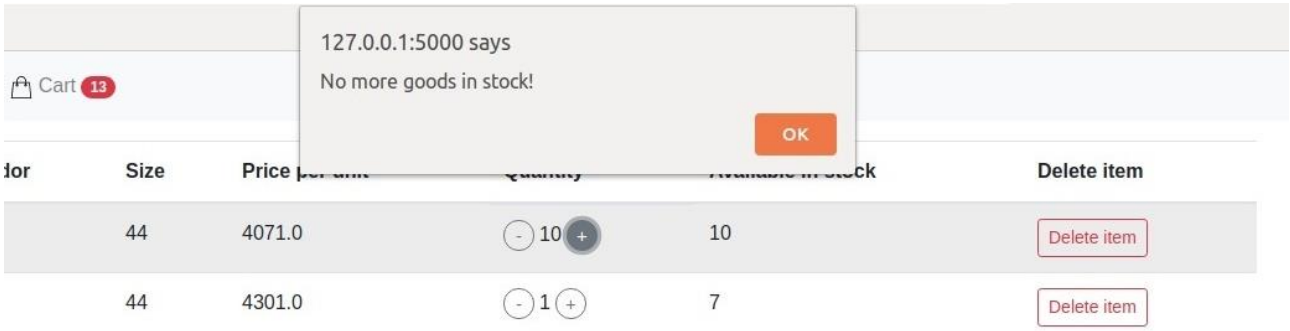

Рисунок 18. Предупреждение при попытке заказать товара больше, чем есть на складе

После заполнения формы и ее отправки, пользователя перенапляет на страницу подтверждения заказа и информации о том, что с ним свяжется администратор по указанным данным (рис. 19).

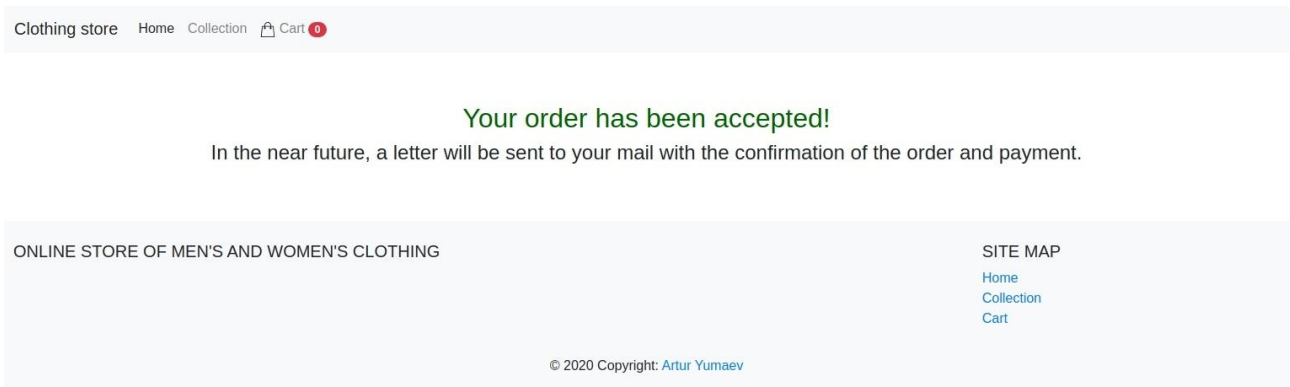

Рисунок 19. Страница подтверждения заказа

Вывод

Проанализировав инструменты для создания каждого из слоев монолитной архитектуры приложения, а именно слоя пользовательского интерфейса, слоя бизнес-логики и слоя данных, были выбраны конкретные технологии, основываясь на особенностях и ограничениях проекта. Был составлен технологический стек, произведена разработка и сборка проекта в готовый веб-сервис. Представлен графический интерфейс и краткие указания к работе с приложением.

4. Экспериментальная часть

В рамках экспериментальной части было проведено тестирование сайта на производительность с помощью сервиса Pingdom Website Speed Test [5]. В результате сайт получил категорию производительности B со средней по 89 запросам загрузкой страницы 1.53 сек. Данное время вполне укладывается в современные стандарты, результаты приведены на рисунке 20.

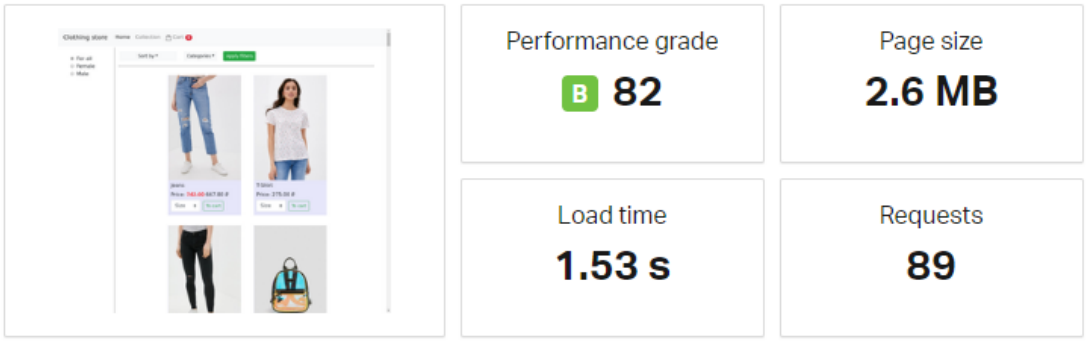

Рисунок 20. Тестирование сайта

Также сайт был протестирован с помощью сервиса PageSpeed Insights от Google [6]. Результаты показаны на рисунке 21.

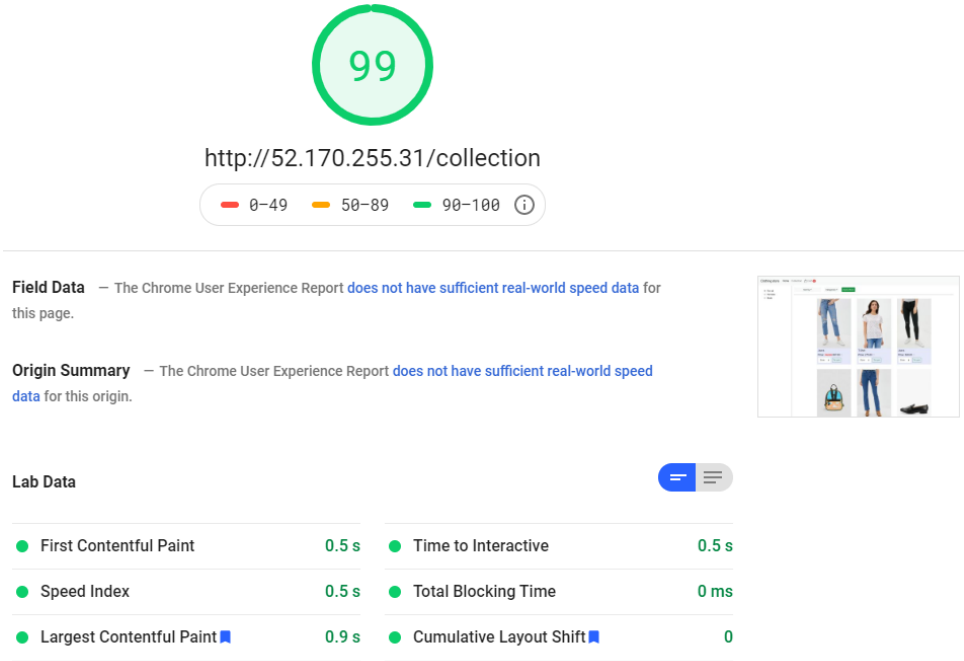

Рисунок 21. Тестирование сервиса

Данный тест также показал положительную оценку и 99% скоринг быстроты сервиса. Среднее время отрисовки медиа элементов составило 0.5 секунды, время до того момента, как пользователь может взаимодействовать с интерактивными элементами также 0.5 секунд. Время прогрузки всех элеметов не превысило 1 секунды и составило 0.9 секунд, что является достаточно приемлимым показателем.

Вывод

В данном разделе было проведено тестирование сайта с помощью различных сервисов. Тесты показали, что производительность сайта высока и не нуждается в доработке.

Заключение

В результате выполнения работы был реализован интернет-магазин с пользовательской корзиной, а также фунцией фильтрации и сортировки товаров. Для решения задачи был выбран язык программирования, фреймворк для создания веб-сервера и http сервер, CSS библиотека для разработки адаптивного интерфейса, выбрана база данных для хранения информации и API для выбранной базы данных, спроектирована и разработана база данных, спроектировано и разработано веб-приложение с помощью выбранных технологий и разработан гибкий интерфейс для созданного веб-приложения. Проведены тесты производительности и сделаны выводы.

Список используемой литературы

[1] – Global Ecommerce Statistics and Trends to Launch Your Business Beyond Borders [Электронный ресурс]. 2020. Дата обновления: 24.03.20. URL: https://www.shopify.com/enterprise/global-ecommerce-statistics (дата обращения: 24.03.20).

[2] – Ecommerce Statistics for 2020 – Chatbots, Voice, Omni-Channel Marketing [Электронный ресурс]. 2020. Дата обновления: 24.03.20. URL: https://kinsta.com/blog/ecommerce-statistics/ (дата обращения: 24.03.20).

[3] – Просто о микросервисах. Блог компании Райффайзенбанк. [Электронный ресурс]. 2018. Дата обновления: 18.05.20. URL: https://habr.com/ru/company/raiffeisenbank/blog/346380/ (дата обращения: 18.05.20)

[4] – Блог компании SimbirSoft. [Электронный ресурс]. 2019. Дата обновления: 18.05.20. URL: https://habr.com/ru/company/simbirsoft/blog/453932/ (дата обращения: 18.05.20)

[5] – Pingdom Website Speed Test. [Электронный ресурс]. 2020. Дата обновления 03.06.20. URL: https://tools.pingdom.com/ (дата обращения: 03.06.20)

[6] – PageSpeed Insight. [Электронный ресурс]. 2020. Дата обновления 03.06.20. URL: https://developers.google.com/speed/pagespeed/insights/ (дата обращения: 03.06.20)