**Министерство транспорта Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ   
учреждение ВЫСШЕГО ОБРАЗОВАНИЯ**

**«Российский университет транспорта»**

**(РУТ** **(МИИТ))**

Институт пути, строительства и сооружений

Кафедра «Системы автоматизированного проектирования»

|  |
| --- |
| **Допустить к защите**  **Заведующий кафедрой:**  Нестеров И. В.\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_**  (ФИО) (подпись)  "26 "\_\_\_мая\_\_\_\_\_\_ 2022 г. |

|  |
| --- |
|  |

**Бакалаврская работа**

**на тему: СОЗДАНИЕ САЙТА ИНТЕРНЕТ-МАГАЗИНА С ПОМОЩЬЮ СОВРЕМЕННЫХ ВОЗМОЖНОСТЕЙ WEB - ДИЗАЙНА**

(направление подготовки, профиль) 09.03.01 – «Информатика и вычислительная техника»

Системы автоматизированного проектирования

**Обучающийся (Стариков И. А.)**

**Руководитель бакалаврской работы (доц. , к.т.н. Назаренко С. Н.)**

**Москва 2022 г.**

**Министерство транспорта Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ   
учреждение ВЫСШЕГО ОБРАЗОВАНИЯ**

**«Российский университет транспорта»**

**(РУТ** **(МИИТ))**

Институт /факультет Институт пути, строительства и сооружений

Кафедра Системы автоматизированного проектирования

Направление подготовки 09.03.01 Информатика и вычислительная техника

УТВЕРЖДАЮ

Зав. кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«8» февраля 2022 г.

**Задание**по бакалаврской работе студента   
Старикова Ильи Александровича

1. Тема работы: Создание сайта интернет-магазина с помощью современных возможностей web-дизайна

утверждена приказом по университету от «\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. № \_\_\_\_\_\_\_\_

2. Срок сдачи студентом законченной работы 25.05.2022

3. Исходные данные к работе Программный код, макеты страниц

4. Содержание пояснительной записки (перечень подлежащих разработке вопросов)

Введение.   
Глава 1. Обзор предметной области.

Глава 2. Описание работы программы.

Глава 3. Экспериментальная проверка программы.

6. Дата выдачи задания \_\_\_\_24.02.2022 г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Задание принял к исполнению** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| №п/п | Наименование этапов  бакалаврской работы | Срок выполнения  этапов работы | Примечание |
| 1 | Разработка темы и поиск информации | 24.02.2022-20.03.2022 |  |
| 2 | Форматирование и структурирование информации для работы | 20.03.2022-8.04.2022 |  |
| 3 | Создание макетов | 20.04.2022-01.05.2022 |  |
| 4 | Разработка программы | 01.05.2022-20.05.2022 |  |
| 5 | Подготовка демонстрационных материалов | 20.05.2022-30.05.2022 |  |
|  |  |  |  |

Оглавление

[ВВЕДЕНИЕ 5](#_Toc105959713)

[ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc105959714)

[1.1 Классификация web-сайтов 7](#_Toc105959715)

[1.2 Стек используемых технологий 8](#_Toc105959716)

[1.2.1 HTML, CSS 8](#_Toc105959717)

[1.2.2 JavaScript 21](#_Toc105959718)

[1.2.3 SQL 23](#_Toc105959719)

[1.2.4 PUG(JADE) 25](#_Toc105959720)

[1.2.4 SASS/SCSS 27](#_Toc105959721)

[1.2.4 NodeJS 30](#_Toc105959722)

[1.2.4 Git 32](#_Toc105959723)

[ГЛАВА 2. ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ 33](#_Toc105959724)

[ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА ПРОГРАММЫ. 36](#_Toc105959725)

[ЗАКЛЮЧЕНИЕ 42](#_Toc105959726)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc105959727)

[ПРИЛОЖЕНИЯ 44](#_Toc105959728)

[Приложение А 44](#_Toc105959729)

[Приложение Б 59](#_Toc105959730)

# ВВЕДЕНИЕ

Интернет стал неотъемлемой частью нашей жизни. В нём можно найти почти всю нужную информацию. На одних сайтах предлагают найти друзей или вторую половинку, на других продают повседневные вещи. Также существуют специальные информационные порталы или сайты, которые посвящены каким-либо людям или компаниям. Следовательно, можно сделать вывод, что для успеха и процветания необходимо заявить о себе в Интернете.

Использование web-технологий открывает огромное пространство для электронной коммерции и обслуживания клиентов через интернет.

В настоящее время, web-технологии сделали огромный шаг вперед. Появились не только новые языки программирования, но и фреймворки, библиотеки, сборщики и т.п. Все это позволяет создавать невероятно красивые, отзывчивые и адаптивные web-сайты, работая не только с Frontend(внешней) частью, но и с Backend(серверной).

Нельзя не согласиться, что, заходя в первый раз на web-сайт человек обращает внимание на стильное оформление, плавность действий, правильно подобранные шрифты и цвета. Можно сказать, что задачей web-дизайна любого сайта – это вызвать доверие посетителя и как можно дольше удержать его интерес к своей компании.

Также огромный плюс, создание собственного web-сайта способствует решение такой проблемы, как постоянное обновление контента.

Таким образом, тема дипломного проекта “Создание сайта интернет магазина с помощью современных возможностей web-дизайна” является актуальной, так как данная разработка является огромным преимуществом для компании. Данный web-сайт даёт возможность человеку удаленно приобрести то, что ему нужно. Аудитория данного сайта ничем не ограничена, это может быть даже ребенок или пожилой человек.

**Объект исследования дипломной работы**: создание web-сайта компании “ТехноЭра”.

**Предмет исследования дипломной работы:** web-сайт компании “ТехноЭра”, предоставляющий услуги по продаже электронной техники.

**Цель дипломной работы**: разработать web-сайт компании “ТехноЭра”, оказывающий услуги по продаже электронной техники.

**Задачи дипломной работы:**

• исследовать предметную область;

• исследовать и анализировать существующие web-сайты похожих компаний, предоставляющий услуги по продаже электронной техники;

• исследовать средства для разработки макета web-сайта;

• исследовать средства для разработки web-сайта;

• разработать макет web-сайта

• разработать Frontend структуру web-сайта;

• разработать Backend(серверную) часть web-сайта;

• реализовать плавность анимаций, действий, эффектов;

• реализовать разработанную модель web-сайта;

# ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Классификация web-сайтов

Web-сайт – это набор связанных ссылками между собой электронных страниц. Они могут содержать абсолютно любую информацию. Это может быть, например, какая-либо текстовая информация, графическая и др. Web-сайт находится в Интернете по определенному адресу - доменному имени сайта в виде <https://www.названиеСайта.com>, где http/s – протокол безопасности, www-домен 3-его уровня, названиеСайта - домен 2-ого уровня и .com – домен 1-ого уровня. Каждая страница web-сайта – это новый текстовый документ, который написан на языке разметки (HTML, XML, XHTML, PUG и др.) с использованием других различных технологий (JavaScript, Python, SCSS, и т.п.).

По предоставляемой информации web-сайты можно разбить на несколько основных видов:

• Сайт визитка (Landing Page)

В основном это одна, но иногда может быть несколько, страниц, которые находятся на одном домене, имеют единый фирменный стиль, похожую структуру, внутреннюю связность ссылками. В основном, они представляют собой информацию о владельце – это может быть как физическое, так и юридическое лицо. Также даёт информацию об услугах, которые они предоставляют. Главной задачей такого web-сайта – это познакомить посетителя с организацией и её деятельностью, дать возможность связаться с ними.

• Корпоративные сайты (Сorporate websites)

Один из наиболее популярных типов сайтов. Его основная задача - предоставить как можно больше “полезной” информации о любом проекте или компании, ее истории, преимуществах, ценах, контактах и т. п. Обычно на них размещают примеры реализованных работ, там же можно приобрести какой-либо товар или услугу.

• Интернет-магазины

Название говорит само за себя. На этих сайтах люди могут приобретать какие-либо товары и/или услуги. Интернет-магазин может относиться как к физически существующему магазину, так и представлять собой платформу в интернете. Таких примеров огромное количество – Aliexpress, Wilberries, Gog и т. п.

• Информационные сайты

Задачей таких сайтов является предоставление посетителям какой-либо информации. Существует несколько видов таких сайтов.

-Новостные сайты – задача такого сайта заключается в осведомлении пользователей о последних новостях разных отраслей жизни.

## 1.2 Стек используемых технологий

Для разработки веб-сайта были выбраны следующие технологии: HTML, PUG. SCSS, CSS, JavaScript, NODEJS, MYSQL.

### 1.2.1 HTML, CSS

Для начала разберём почему были выбрана инструменты вёрстки HTML и CSS. Язык HTML не только универсален, но и один из самых простых для освоения новичками. Это основа любого веб-сайта.

Если вы знаете, как кодировать в HTML, вы сможете создавать и структурировать веб-страницы. Чтобы придать им четкий и динамичный вид, вы должны комбинировать элементы HTML с CSS и JavaScript. Перед этим вы должны познакомиться с основами HTML и научиться использовать теги.

В 1990 году Тим Бернерс-Ли создал HTML и основал его на Стандартном обобщенном языке разметки (SGML). Основной принцип основ HTML - окружение текста тегами в угловых скобках для применения выбранного элемента.

Что означает HTML? HTML — это аббревиатура от языка гипертекстовой разметки. Он призван помочь людям структурировать и форматировать контент, чтобы интернет-браузеры правильно отображали страницы. Одного использования HTML недостаточно, чтобы сделать ваши веб-сайты привлекательными, красивыми и динамичными. Для этого вы должны изучить Святую Троицу веб-разработки: HTML, CSS и JavaScript.

Прежде чем вы начнете изучать основы языка HTML, вы должны помнить о важности правильного построения тегов. Теги делятся на разные категории в зависимости от их назначения:

Корневой элемент

Метаданные документа

Разделы

Группировка контента

Семантика на уровне текста

Таблицы

Формы

Интерактивные элементы

Скрипты

Обычный HTML-тег состоит всего из нескольких компонентов. Он начинается с открывающего тега и заканчивается закрывающим тегом, который может содержать косую черту перед именем тега. В последней HTML5 версии ставить закрывающую косую черту не обязательно. Угловые скобки меньше (<) и больше (>) используются для заключения каждого тега. Теги, могу не иметь закрывающего тега, тогда их называют самозакрывающимися тэгами.

HTML был изобретен Тимом Бернерсом-Ли, физиком из исследовательского института ЦЕРН в Швейцарии. Он придумал идею гипертекстовой системы на базе Интернета.

Гипертекст — это текст, содержащий ссылки на другие тексты, к которым зрители могут получить немедленный доступ. Он опубликовал первую версию HTML в 1991 году, состоящую из 18 тегов HTML. С тех пор каждая новая версия языка HTML добавляла в разметку новые теги и атрибуты (модификаторы тегов).

Согласно справочнику по HTML-элементам Mozilla Developer Network, в настоящее время существует 140 тегов HTML, хотя некоторые из них уже устарели и не поддерживаются современными браузерами.

В связи с быстрым ростом популярности HTML теперь считается официальным веб-стандартом. Спецификации HTML поддерживаются и разрабатываются Консорциумом World Wide Web (W3C). Вы можете в любое время проверить последнюю версию языка на веб-сайте W3C.

Самым большим обновлением языка стало внедрение HTML5 в 2014 году. Он добавил в разметку несколько новых семантических тегов, раскрывающих значение собственного содержимого, таких как <article>, <header>, <footer>, <section>, <aside> , <nav> т.п .

HTML-теги

HTML-теги используются для структурирования содержимого веб-сайта (текста, гиперссылок, изображений, мультимедиа и т. Д.). Теги не отображаются в браузерах, они только показывают браузеру, как отображать контент на веб-странице.

Теги HTML записываются в угловых скобках (например, <html>).

Большинство HTML-тегов состоит из пар, как и теги <p> </p>. Первый тег в паре называется открывающим тегом, а второй - закрывающим тегом. Информация записывается между ними.

Однако есть непарные тэги, у них только открывающий тег. (например, <img>).

Рассмотрим пример.

Если вам нужно определить абзац (который является элементом), вы должны использовать тег <p>. Содержание абзаца следует писать между открывающим и закрывающим тегами.

Пример:

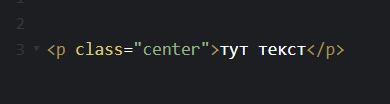


Рисунок 1. Тэг <p> в HTML



Рисунок 2. Вывод тэга <p> на страницу

Это абзац между открывающим <p> и закрывающим </p> тегами.

Атрибуты HTML

Атрибуты HTML добавляются к элементу HTML, чтобы предоставить дополнительную информацию о нем. Например, если вы определяете изображение с помощью тега <img />, вы можете использовать атрибуты src, height , width,id,class,  для предоставления информации о его источнике, высоте и ширине соответственно, его личном идентификаторе и классе.

Пример:

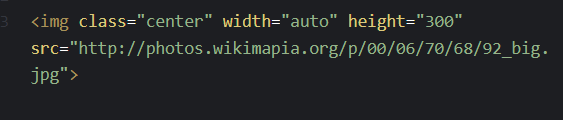


Рисунок 3. Тэг <img> в HTML



Рисунок 4. Тэг <img> на странице

Структура HTML-документа

<! DOCTYPE HTML> Декларация определяет HTML - версия используется в документе. Каждый HTML-документ должен начинаться с этого объявления, чтобы браузеры могли отображать страницу в соответствии со стандартами HTML. В современных браузерах есть возможность не ставить это объявление. Они могут сами определить и решить, что лучше поставить. Но это не самый лучший подход, поэтому лучше всегда собственноручно прописывать данное объявление.

Существует несколько типов <! DOCTYPE>, определенных для каждой версии HTML.

Все содержимое веб-страницы записывается между тегами <html> </html>. Элемент <html> используется для предоставления информации браузерам о том, что это документ HTML.

Элемент <head> содержит метаданные (данные о документе HTML), набор символов, заголовок документа, стили и т. Д. Эти данные не отображаются для зрителей.

<title> отображает название сайта на вкладке браузера, когда страница загружена. Заголовок пишется между тегами <title> </title>.

Элемент <body> содержит содержимое веб-страницы (текст, изображения, видео и т. Д.). Контент пишется между <body> </body>.

Элементы заголовка содержат заголовки разных типов. Существует шесть уровней заголовков - <h1> - <h6>, где <h1> - наиболее важные и <h6> наименее важные теги.

Элемент <p> содержит абзацы текста. Контент записывается между тегами <p> и </p>.

Большинство HTML-элементов имеют начало и конец, в которых используется синтаксис <tag> </tag>

HTML-теги бывают двух основных типов: блочные и строчные.

Элементы уровня блока занимают все доступное пространство и всегда начинают новую строку в документе. Заголовки и абзацы - отличный пример блочных тегов.

Строчные элементы занимают ровно столько места, сколько им нужно, и не начинают новую строку на странице. Обычно они служат для форматирования внутреннего содержимого элементов уровня блока. Ссылки и выделенные строки - хорошие примеры встроенных тегов.

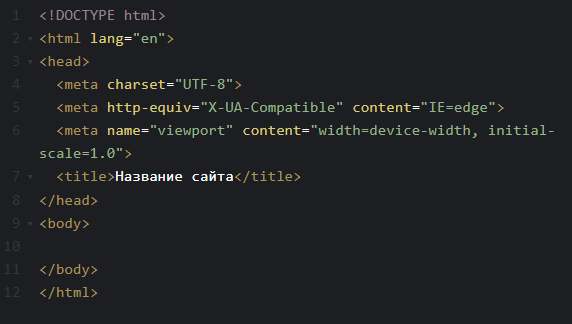


Рисунок 5. Стандартная структура HTML

Эволюция HTML.

С первых дней HTML претерпел невероятную эволюцию. W3C постоянно публикует новые версии и обновления, а исторические вехи также получают отдельные названия.

HTML4 был опубликован в 1999 году, а последняя основная версия вышла в 2014 году. Обновление, получившее название HTML5, внесло в язык много новых функций.

Одна из самых ожидаемых функций HTML5 - встроенная поддержка встраивания аудио и видео. Вместо использования Flash-плеера мы можем просто встраивать видео и аудиофайлы на наши веб-страницы с помощью новых тегов <audio> </audio> и <video> </video> . Он также включает встроенную поддержку масштабируемой векторной графики (SVG) и MathML для математических и научных формул.

HTML5 также внес несколько семантических улучшений. Новые семантические теги информируют браузеры о значении контента, что приносит пользу как читателям, так и поисковым системам.

Самыми популярными семантическими тегами являются <article> </article>, <section> </section>, <aside> </aside>, <header> </header> и <footer> </footer>.

Плюсы и минусы HTML

Как и большинство других вещей, HTML имеет ряд сильных сторон и ограничений.

Плюсы:

Широко используемый язык с большим количеством ресурсов и огромным сообществом.

Работает изначально в каждом веб-браузере.

Поставляется с плоской кривой обучения.

С открытым исходным кодом и совершенно бесплатно.

Чистая и последовательная разметка.

Официальные веб-стандарты поддерживаются Консорциумом всемирной паутины (W3C).

Легко интегрируется с серверными языками, такими как PHP и Node.js.

Минусы:

В основном используется для статических веб-страниц. Для динамической функциональности вам может потребоваться использование JavaScript или серверного языка, такого как PHP.

Это не позволяет пользователю реализовать логику. В результате все веб-страницы необходимо создавать отдельно, даже если они используют одни и те же элементы, например, верхние и нижние колонтитулы.

Некоторые браузеры медленно внедряют новые функции.

Поведение браузера иногда трудно предсказать (например, старые браузеры не всегда отображают новые теги).

Рассмотрим основные HTMLтеги

<html>: каждый HTML-код должен быть заключен между основными HTML-тегами. Он начинается с <html> и заканчивается тегом </html> .

<head>: Следующим идет тег заголовка, который содержит всю информацию заголовка веб-страницы или документа, такую ​​как заголовок страницы и другую разную информацию. Эта информация заключена в тег заголовка, который открывается с <head> и заканчивается </head> .

Содержание этого тега, конечно же, будет объяснено в следующих разделах.

<title>: мы можем указать заголовок веб-страницы с помощью тега <title>. Это информация заголовка и, следовательно, упоминается в тегах заголовка. Тег начинается с <title> и заканчивается </title>

<body>: Следующий шаг - самый важный из всех тегов, которые мы изучили до сих пор. Тег body содержит фактическое тело страницы, которое будет видно всем пользователям. Это начинается с <body> и заканчивается </body>. Каждый контент, заключенный в этот тег, будет отображаться на веб-странице, будь то тексты, изображения, аудио или видео или даже ссылки. Позже в этом разделе мы увидим, как с помощью различных тегов мы можем вставлять упомянутое содержимое на наши веб-страницы.

Весь паттерн кода будет выглядеть примерно так:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <!—Подключение различных стилей, META, настройка  вывода на экраны различный устройств-->  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <!—Название сайта-->  <title>Document</title>  </head>  <!—Основной тэг, в котором хранится контент-->  <body>    </body>  </html> |

Этот код ничего не отображает. Он просто показывает базовый шаблон того, как написать HTML-код, и назовет заголовок страницы как GeeksforGeeks. <! - - comment здесь - -> — это тег комментария в HTML, и он не читает строку, присутствующую внутри этого тега.

Заголовки HTML

Эти теги помогают нам присваивать заголовки содержанию веб-страницы. Эти теги в основном записываются внутри тега body. HTML предоставляет нам шесть тегов заголовков от <h1> до <h6>. Каждый тег отображает заголовок в своем стиле и размере шрифта.

Пример:

|  |
| --- |
| <html>  <head>      <title>РУТ (МИИТ)</title>  </head>  <body>      <h1>Hello РУТ (МИИТ)</h1>      <h2>Hello РУТ (МИИТ)</h2>      <h3>Hello РУТ (МИИТ)</h3>      <h4>Hello РУТ (МИИТ)</h4>      <h5>Hello РУТ (МИИТ)</h5>      <h6>Hello РУТ (МИИТ)</h6>  </body>  </html> |



Рисунок 6. Вывод на страницу заголовков

HTML абзац

Эти теги помогают нам писать абзацы на веб-странице. Они начинаются с тега <p> и заканчиваются </p>. Здесь тег <br> используется для разрыва строки, хотя в современной веб-разработке этот способ редко используется, и действует как возврат каретки. <br> — это пустой тег.  
Пример:

|  |
| --- |
| <html>  <head>      <title> РУТ (МИИТ)</title>  </head>  <body>      <h1>Hello РУТ (МИИТ)</h1>      <p>  Hello РУТ (МИИТ) <br>      РУТ (МИИТ)<br>  РУТ (МИИТ)<br>      </p>  </body>  </html>    Рисунок 6. Демонстрация работы тэга <br> |

CSS означает каскадные таблицы стилей с упором на «стиль». В то время как HTML используется для структурирования веб-документа (определения таких вещей, как заголовки и абзацы, а также для встраивания изображений, видео и других мультимедийных материалов), CSS проходит через него и определяет стиль вашего документа - макеты страниц, цвета и шрифты определяются. с CSS. Думайте о HTML как о фундаменте (он есть в каждом доме) и о CSS как о эстетическом выборе (есть большая разница между викторианским особняком и современным домом середины века).

Принцип работы CSS

CSS привносит стиль на ваши веб-страницы, взаимодействуя с элементами HTML. Элементы — это отдельные HTML-компоненты веб-страницы, например, абзац, которые в HTML могут выглядеть следующим образом:

<p> Это мой абзац! </p>

Если вы хотите, чтобы этот абзац отображался розовым и жирным шрифтом для людей, просматривающих вашу веб-страницу через веб-браузер, вы должны использовать код CSS, который выглядит следующим образом:

p {цвет: розовый; font-weight: жирный;}

В этом случае «p» (абзац) называется «селектором» — это часть кода CSS, определяющая, на какой элемент HTML будет влиять стиль CSS. В CSS селектор написан слева от первой фигурной скобки. Информация в фигурных скобках называется объявлением и содержит свойства и значения, которые применяются к селектору. Свойства — это такие вещи, как размер шрифта, цвет и поля, а значения — это настройки этих свойств. В приведенном выше примере «color» и «font-weight» являются свойствами, а «pink» и «bold» - значениями. Полный набор в скобках {цвет: розовый; font-weight: жирный;} — это объявление, и снова «p» (то есть абзац HTML) — это селектор. Эти же базовые принципы можно применять для изменения размеров шрифта, цвета фона, отступов полей и т. Д.

Внешний, внутренний или встроенный CSS

Однако вам может быть интересно, как этот код CSS на самом деле применяется к содержимому HTML. Как и HTML, CSS пишется в виде простого обычного текста с помощью текстового редактора или текстового процессора на вашем компьютере, и есть три основных способа добавить этот код CSS на ваши HTML-страницы. Код CSS (или таблицы стилей) может быть внешним, внутренним или встроенным. Внешние таблицы стилей сохраняются как файлы .css и могут использоваться для определения внешнего вида всего веб-сайта с помощью одного файла (вместо добавления отдельных экземпляров кода CSS к каждому элементу HTML, который вы хотите настроить). Чтобы использовать внешнюю таблицу стилей, ваши файлы .html должны включать раздел заголовка, который ссылается на внешнюю таблицу стилей и выглядит примерно так:

<head>

<link rel =” stylesheet” type =” text / css” href = mysitestyle.css ”>

</head>

Это свяжет файл .html с вашей внешней таблицей стилей (в данном случае mysitestyle.css), и все инструкции CSS в этом файле будут применяться к вашим связанным страницам .html.

Внутренние таблицы стилей — это инструкции CSS, записанные непосредственно в заголовок определенной HTML-страницы. (Это особенно полезно, если у вас есть одна страница на сайте, которая имеет уникальный внешний вид.)

Вообще говоря, внешние таблицы стилей - наиболее эффективный метод реализации CSS на веб-сайте (легче отслеживать и реализовывать стиль сайта из специального файла CSS), в то время как внутренние таблицы стилей и встроенный стиль могут использоваться в каждом конкретном случае. основание случая, когда необходимо внести индивидуальные изменения стиля.

Итак, если HTML является основой, каркасами, стенами и балками, поддерживающими ваш веб-сайт, подумайте о CSS - цвете краски, стилях окон и ландшафтном дизайне, которые появятся позже. Вы ничего не добьетесь, не заложив сначала эту основу, но - как только вы это сделаете - вам захочется продолжить работу с некоторым стилем, а CSS — это билет, который поможет раскрыть ваш внутренний декоратор.

### 1.2.2 JavaScript

JavaScript — это язык программирования на стороне клиента (он способен работать непосредственно в браузере пользователя на локальном компьютере).

JavaScript используется для создания веб-приложений и динамических сайтов, которые могут взаимодействовать с пользователем.

Прежде всего, вы должны помнить, что JavaScript — это язык, чувствительный к регистру. По этой причине вы должны не только запоминать имена свойств, методов или ключевых слов JavaScript, но и писать их прописными или строчными буквами.

Код JavaScript может быть написан либо непосредственно в самом HTML - документе, либо в отдельном файле.

Чтобы написать код JavaScript в HTML-документе, используйте тег <script>. Скрипты часто помещаются внутри элемента <head>. Это гарантирует, что сценарий будет готов к запуску при его вызове. Однако это не является обязательным требованием, и вы также можете поместить его в элемент <body>, где он также будет работать нормально.

Вы также можете указать язык с помощью атрибута type (type= "text/javascript"). Однако при использовании JavaScript атрибут type можно опустить, поскольку JavaScript является значением атрибута type по умолчанию.

Гипертекстовая информационная система состоит из набора информационных узлов, набора гипертекстовых ссылок, определенных на этих узлах, и инструментов для управления узлами и ссылками. Технология Всемирной паутины — это технология поддержания гипертекстовых распределенных систем в Интернете, и, следовательно, она должна соответствовать общему определению таких систем. Это означает, что все компоненты гипертекстовой системы, перечисленные выше, также должны быть в Сети.

Сеть, как гипертекстовую систему, можно рассматривать с двух точек зрения. Во-первых, как набор отображаемых страниц, связанных гипертекстовыми ссылками (ЯКОРНЫЕ контейнерные ссылки). Во-вторых, как набор элементарных информационных объектов, составляющих отображаемые страницы (текст, графика, мобильный код и т. д.). В последнем случае набор гипертекстовых переходов страниц представляет собой тот же информационный фрагмент, что и изображение, встроенное в текст.

Во втором подходе гипертекстовая сеть определяется на множестве элементарных информационных объектов самими HTML-страницами, которые играют роль гипертекстовых ссылок. Этот подход более продуктивен с точки зрения создания страниц отображения на лету из готовых компонентов.

При создании страниц в Интернете возникает дилемма, связанная с архитектурой "клиент-сервер". Страницы могут создаваться как на стороне клиента, так и на стороне сервера. В 1995 году специалисты Netscape создали механизм управления страницами на стороне клиента, разработав язык программирования JavaScript.

JavaScript — это язык сценариев на стороне клиента для просмотра гипертекстовых веб-страниц.

Liveware, прародитель JavaScript, представляет собой подстановочный знак на стороне сервера Netscape. Однако наибольшую популярность JavaScript обеспечило программирование на стороне клиента.

Основная идея JavaScript заключается в изменении значений атрибутов HTML-контейнеров и свойств среды отображения во время просмотра пользователем HTML-страницы. При этом страница не перезагружается.

На практике это означает, что вы можете, например, изменить цвет фона страницы или изображения, интегрированного в документ,открыть новое окно или выдать предупреждение.

Имя "JavaScript" является собственностью Netscape. Реализация языка, реализованная разработчиками Microsoft, официально называется Jscript. Версии JScript совместимы (но не полностью совместимы) с соответствующими версиями JavaScript, т. Е. JavaScript является подмножеством языка JScript.

JavaScript стандартизирован ECMA (Европейской ассоциацией производителей компьютеров). Обратите внимание, что не все реализации JavaScript сегодня полностью соответствуют стандарту ECMA.

### 1.2.3 SQL

SQL — это структурированный язык запросов, созданный для того, чтобы получать из базы данных необходимую информацию. Если описать схему работы SQL простыми словами, то специалист формирует запрос и направляет его в базу. Та в свою очередь обрабатывает эту информацию, «понимает», что именно нужно специалисту, и отправляет ответ.

Данные хранятся в виде таблиц, они структурированы и разложены по строкам и столбцам, чтобы ими легче было оперировать. Такой способ хранения информации называют реляционными базами данных (от англ. relation — «отношения»). Название указывает на то, что объекты в такой базе связаны определенными отношениями.

Например, у маркетолога есть база, в которой собрана информация обо всех пиццериях в городе: названия, ассортимент, цены, график работы и прочее. Во время анализа конкурентов он решил выяснить, сколько пиццерий готовят пиццу с ананасами и оформляют доставку после 23:00. Для того чтобы получить такой список из базы, достаточно написать грамотный SQL-запрос.

Работать с данными помогают операторы — определенные слова или символы, которые используются для выполнения конкретной операции — например, для выбора из множества по конкретному параметру. Если нам нужно из всех видов пиццы отсортировать те, в которых есть пармезан, — нужно использовать оператор SELECT (выбор в соответствии с условием).

Операторы в SQL делятся на несколько групп в соответствии с задачами, которые они решают.

DDL (Data Definition Language) — операторы определения данных. Они работают с объектами, то есть с целыми таблицами. Если базу нужно дополнить таблицей с новыми данными или, наоборот, убрать одну из таблиц с ошибочными данными — используется этот набор операторов.

CREATE — создание объекта в базе данных

ALTER — изменение объекта

DROP — удаление объекта

DML (Data Manipulation Language) — операторы манипуляции данными. Эти операторы уже работают с содержимым таблиц — строками, атрибутами и значениями. С их помощью можно вносить изменения в конкретное значение. Например, заменить поле в колонке «Фамилия» в строке с данными сотрудницы компании посте того, как она вышла замуж. Или удалить строку с данными уволенного сотрудника.

SELECT — выбор данных в соответствии с условием

INSERT — добавление новых данных

UPDATE — изменение существующих данных

DELETE — удаление данных

DCL (Data Control Language) — оператор определения доступа к данным. Он определяет, кто из пользователей может отправлять запросы к базе, менять объекты и значения. Например, можно отозвать доступ у сотрудника, перешедшего в другой отдел, а также открыть доступ к базе новому маркетологу или разработчику.

GRANT — предоставление доступа к объекту

REVOKE — отзыв ранее выданного разрешения

DENY — запрет, который является приоритетным над разрешением

TCL (Transaction Control Language) — язык управления транзакциями. Транзакции — это набор команд, которые выполняются поочередно. Если все команды выполнены, транзакция считается успешной, а если где-то произошла ошибка — транзакция откатывается назад, отменяя все выполненные команды. Наглядный пример такой транзакции — оплата онлайн, когда банк просит сначала ввести сумму и получателя, затем проверить и подтвердить операцию, а после ввести одноразовый код. На каждом из этих этапов оплату можно отменить, и транзакция откатится назад.

### 1.2.4 PUG(JADE)

**PUG(JADE)** — это препроцессор HTML и шаблонизатор, который был написан на JavaScript для Node.js. У данного шаблонизатора своеобразное написание тэгов. В нём нет закрывающих тегов, вместо этого он использует строгую табуляцию (или отступы) для определения вложенности тегов. Он также позволяет создать более сложную логику в отличии от “чистого” HTML, позволяя использовать часть функционала JS.

ТЭГИ

В **PUG(JADE)** нет закрывающих тэгов, поэтому нужно быть очень осторожным при создании каркаса страниц, чтобы не сломать общую логику работы всех тэгов. Если в незакрытый тэг попадёт другой тэг– это может привести к непредвиденным обстоятельствам.

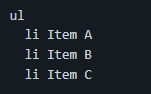


Рисунок 7. Структура списка в Pug

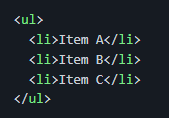


Рисунок 8. Структура списка в HTML

В PUG(JADE) можно встраивать JavaScript код, благодаря чему возможно следующие конструкции.

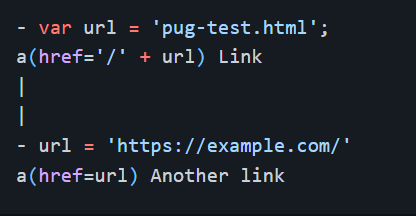


Рисунок 9. Структура ссылок в PUG

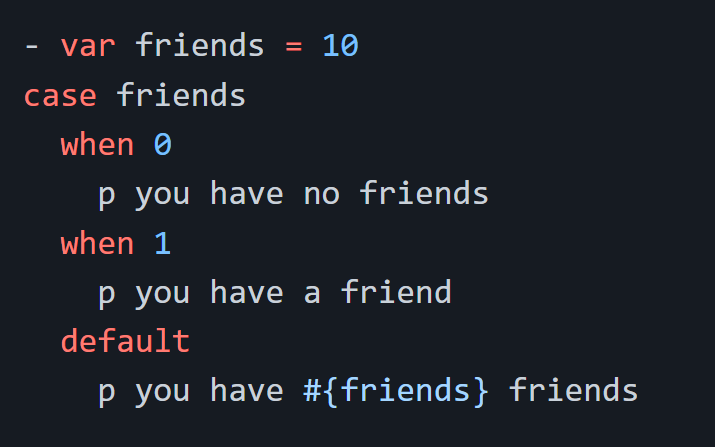


Рисунок 10. Переключатель в Pug

### 1.2.4 SASS/SCSS

**SASS/SCSS** - (или Syntactically Awesome Stylesheets) – это скриптовый метаязык (т. е. язык, описывающий другой язык), разработанный для упрощения файлов CSS.

Написание CSS довольно веселое дело, но когда таблица стилей становится очень большой, то начинаешь путаться в собственном коде. Её обслуживание будет занимать огромное количество времени. Чтобы решить данную проблему и был придумал данный препроцессор для CSS.

Как только Вы начинаете пользоваться SASS/SCSS, препроцессор обрабатывает ваш входной файл и сохраняет его как простой CSS-файл, который Вы сможете использовать на любом сайте.

SASS/SCSS



Рисунок 11. Структура css правил в SCSS с переменными

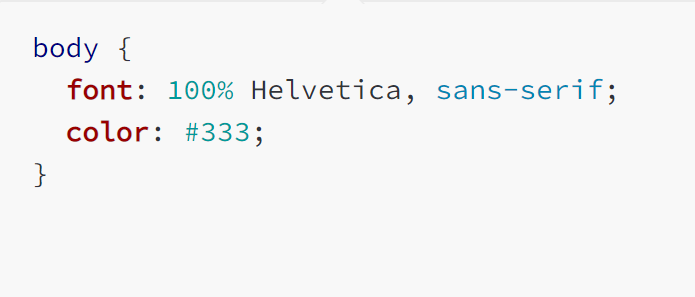


Рисунок 12. Структура css правил в CSS

Также огромный плюс в копилку препроцессора – это возможность создания пользовательских функций (миксины – примеси). Сам по себе CSS такой возможности не имеет. Подход с использованием функций позволяет заметно уменьшить количество строк кода.

SASS/SCSS

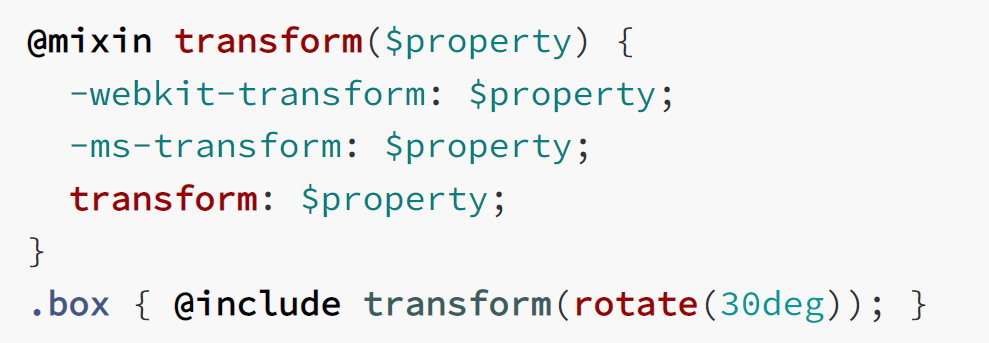


Рисунок 13. Миксины в SCSS

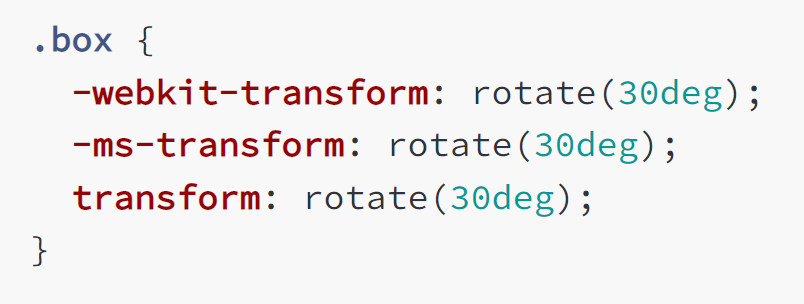


Рисунок 14. Получаемые правила в CSS

Использовать математику в CSS очень полезно. Благодаря ней можно создавать более “резиновые” сайты. SASS/SCSS имеет несколько стандартных математических операторов, таких как +, -, \*, / и %.



Рисунок 15. Математика в SCSS



Рисунок 16. Математика в CSS

### 1.2.4 NodeJS

**NODEJS** — это среда выполнения JavaScript. Окружение Node.js включает все, что вам нужно для выполнения программы, написанной на JavaScript. Раньше вы могли запустить JavaScript только в браузере, но однажды разработчики расширили его, и теперь вы можете запускать JS на своем компьютере в качестве отдельного приложения. Так появился Node.js. Теперь у JavaScript есть возможность делать то, что могут делать другие скриптовые языки программирования, такие как Python.



Рисунок 17. Стандартная структура сервера на NodeJs

Создание простого сервера на NodeJs. В консоль выведется сообщение “Server running at http://127.0.0.1:3000”.

Главное отличие NodeJs от JavaScript это то, что первый является многопоточный языком, построенным на асинхронности. Это позволяет ему оперировать различными данными, не останавливаясь, если пользователь совершил какое-либо действие на сайте.

### 1.2.4 Git

**GIT** — это консольная утилита, для отслеживания и ведения истории изменения файлов, в вашем проекте. В большинстве случаев его используют для различного кода, но можно использовать и для других файлов. С помощью Git-a вы можете откатить свой проект до более ранней версии, сравнивать, анализировать или сливать свои изменения в репозиторий. Репозиторий — это хранилище вашего кода и историй его изменений. Git работает локально, поэтому сохранения, которые он делает можно найти на жёстком диске.

Перед использованием Git его нужно проинициализировать в проекте.

git init

Для того, чтобы сохранить изменения в вашем файле следует использовать команде:

git add -A

Данная команда перемещает этот файл в промежуточную область, отмечая его для включения в следующую фиксацию. Благодаря этой команде можно выбрать не только определенный файл, но и отдельный файлы.

После перемещения файла в промежуточную область, мы обязаны создать коммит. Простыми словами коммит – это снимок ваших файлов. При его создании нужно следить за кавычками. Следует использовать двойные ("Тут коммит"), так как другие Git не воспринимает.

git commit -m "Наш первый коммит"

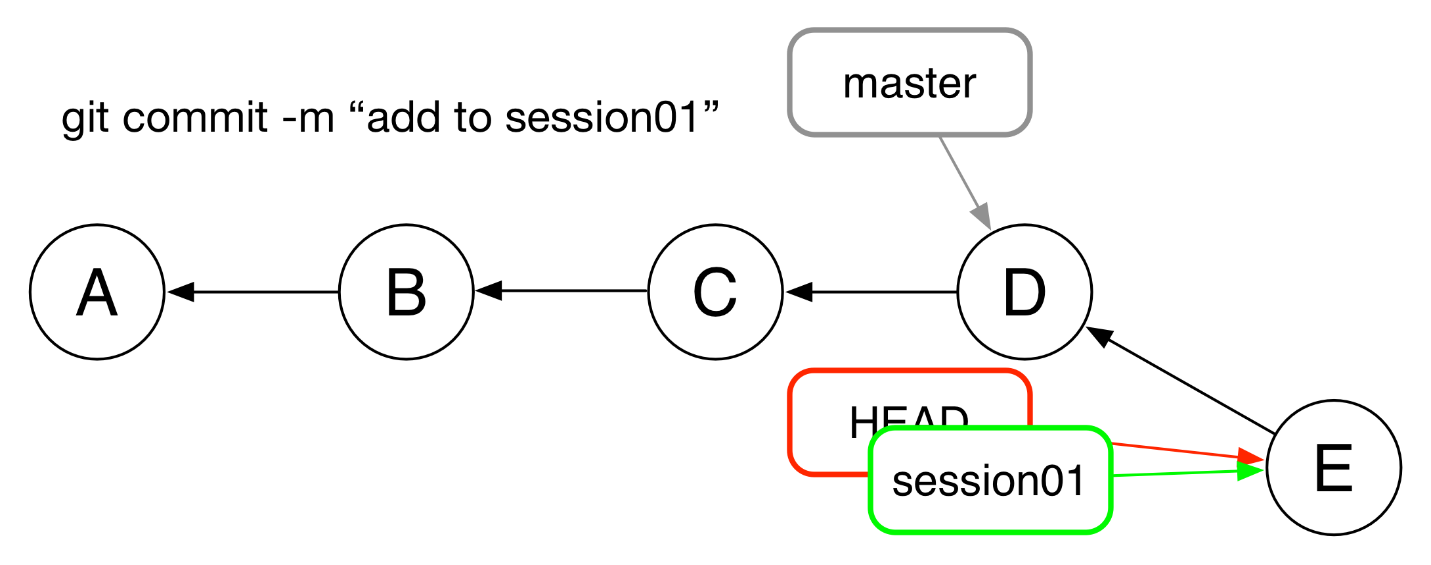


Рисунок 18. Структура git веток

# ГЛАВА 2. ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

Программа разрабатывалась в несколько этапов:

1) Рождение идеи. Хоть я и не веб-дизайнер, но небольшие знания в этой области имею. Моё представление сайта можно увидеть в созданном макете.

2) Создание общей концепции сайта

Логика проста:

Заходя на сайт, клиент может посмотреть интересующего его позиции, поискать более выгодный для себя товар. Может зайти на страницу товара и прочитать его описание. Возможно, ему приглянется какой-либо товар, он может положить его в корзину и купить. На почту клиента и менеджера придет оповещение о приобретённых товарах.

3) Создание базы данных товаров (Товары, категории).

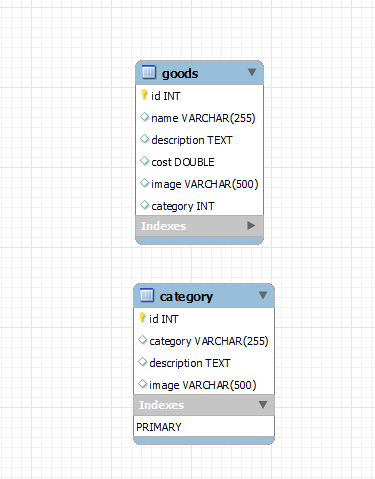


Рисунок 19. База данных товаров.

Данная таблица служит для хранения данных о категориях и о самих товарах.

5) Разработка шаблонов PUG и Frontend логики с помощью JavaScript. Программный код представлен в приложении А. Листинг страниц представлен в приложении Б.

6) Создание серверной части сайта. Подключение нужных модулей. Настройка доступа к базе данных. Проверка получаемых с сервера данных.

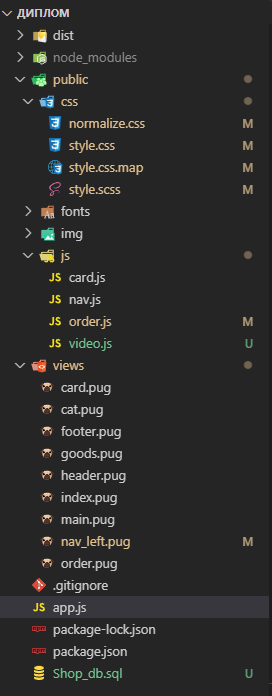


Рисунок 20. Общая схема проекта

dist – создание копии всех файлов

node\_modules – библиотека модулей, хранит в себе используемые модули и зависимости программы.

css –Хранит в себе файлы стилей сайта

fonts – В нём лежат все шрифты, подключенный к сайту

img – Хранит в себе картинки, которые используются на сайте

js – Отвечает за работу javaScript’а на стороне клиента, обработка данных, пришедших с сервера.

views – В этой папке лежат шаблоны страниц

.gitignore – игнорирование статических файлов Git, чтобы создание веток не занимало много времени.

app.js – Точка входа (запуска) веб-приложения, отвечает за создание и запуск сервера веб-приложения. В нём хранится логика работы сервера, роутинг страниц.

package-lock.json – Хранит в себе все зависимости всех используемых в проекте модулей в формате в формате JSON.

package.json – Хранит описание всех зависимостей в формате JSON.

Shop-db.sql – База данных проекта.

# ****ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА ПРОГРАММЫ.****

3.1 Проверка кода

Для данной проверки я просмотрел все pug, js, scss файлы. Лишние и ненужные строчки кода были удалены. Оптимизированы некоторые задачи, путем минификации файлов.

3.2 Имитация реального пользователя.

Для проверки работоспособности программы был проведён тест, он был проведён в несколько этапов, перечислим их:

1. Вход на сайт (с разных устройств – телефон, планшет, ноутбук, пк)

2. Просмотр страницы товара.

3. Добавление товаров в корзину.

4. Совершение заказа.

5. Получение письма на почту.

При входе на сайт нас встречает главная страница.

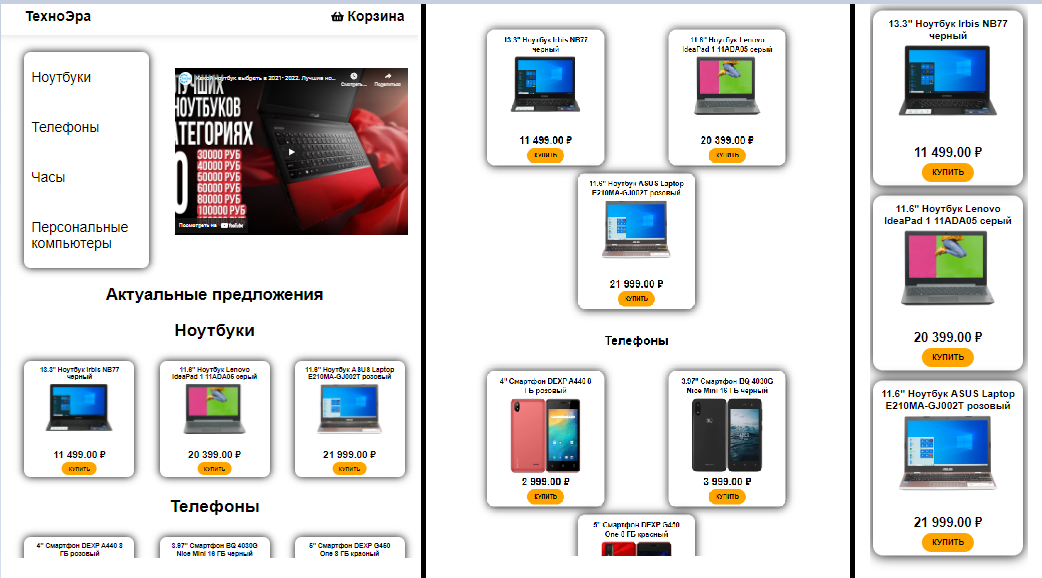


Рисунок 21. Главная страница сайта в разных разрешениях (пк – ноутбук, планшет, телефон)

Далее были протестированы остальные страницы.

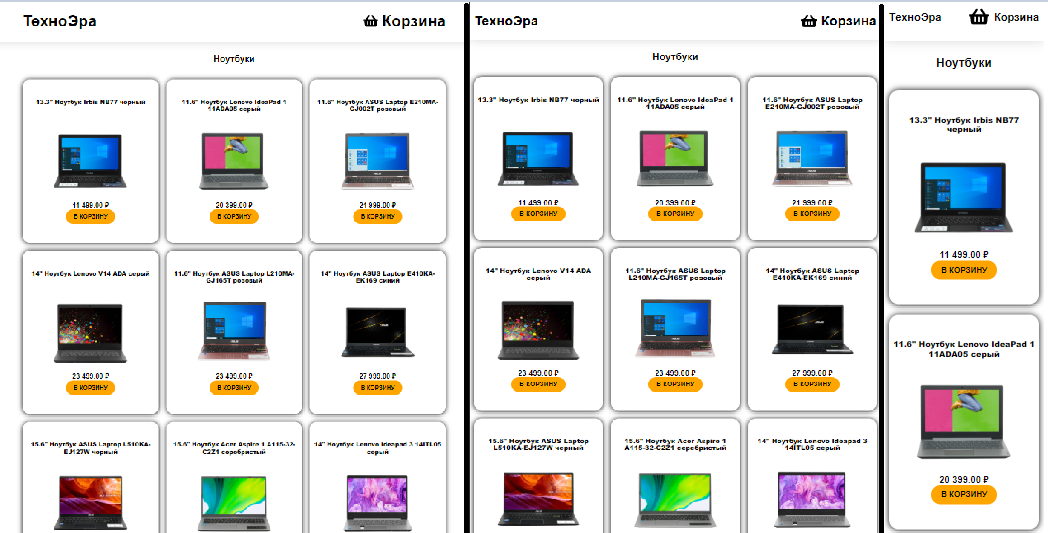


Рисунок 22. Страница категории товара в разных разрешениях (пк – ноутбук, планшет, телефон)

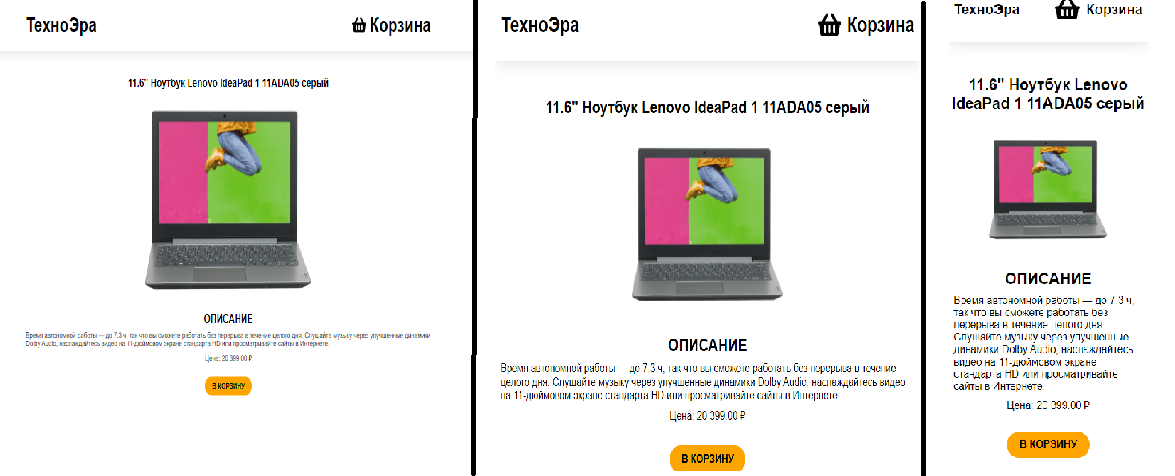


Рисунок 23. Страница товара в разных разрешениях (пк – ноутбук, планшет, телефон)

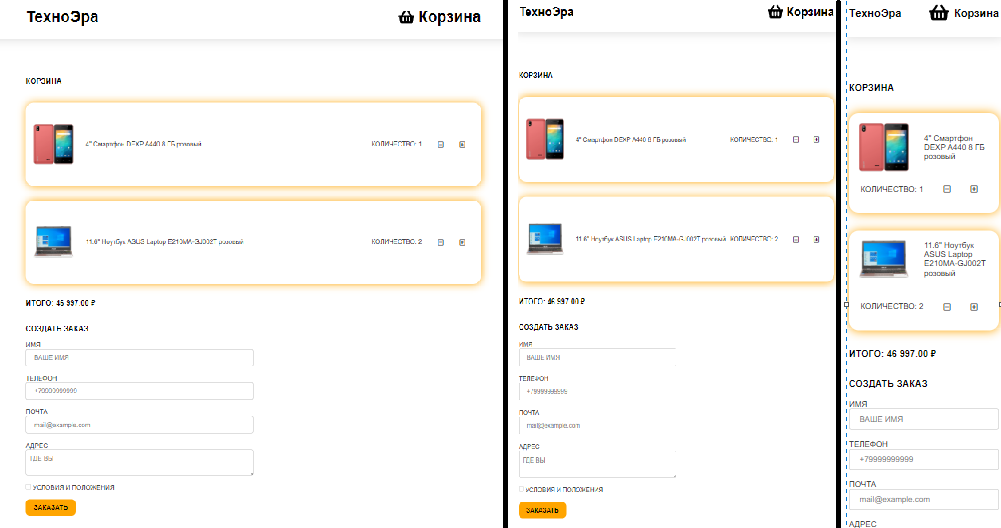


Рисунок 24. Страница корзины в разных разрешениях (пк – ноутбук, планшет, телефон)

Далее проверяем общую работу программы, а именно с 4 по 6 пункты. На данный момент корзина пуста.

Выбирая товар, пользователь нажал на кнопку “В корзину”. После этого отработал скрипт, который добавляет все данные товара в объект корзины и сохраняет в локальном хранилище браузера. Если выйти с сайта, зайти через пару дней, то товар в корзине останется. Переходя на страницу корзины, можно увидеть, что выбранный пользователем товар отобразился. Вводя данные в форму и клике на кнопку “Заказать”, получаем уведомление, что все прошло удачно.

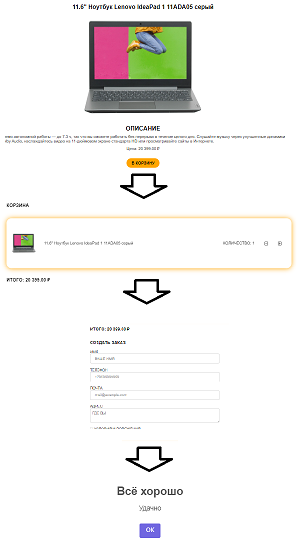


Рисунок 25. Последовательность действий пользователя

Следующим шагом, пользователь может посмотреть сообщение на почте, там будет информация о его заказе.

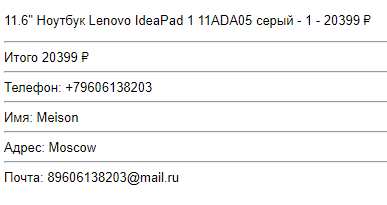


Рисунок 26. Сообщение, отправленное на почту

Сайт, как титановая дверь, выдержал все тесты. Был проверен на различных устройствах. Так что, можно сделать вывод, что он полностью готов к эксплуатации и может быть передан заказчику.

# ЗАКЛЮЧЕНИЕ

Подводя итоги данной работы, можно сказать, что была достигнута основная цель исследования, а именно разработан готовый продукт – веб-сайт.

Для выполнения данной цели были выполнены следующие задачи:

1. Проанализирована предметная область;

2. Разработаны макеты;

3. Разработано веб-приложение;

4. Проведена проверка программы;

А также сделан вывод о том, что разработанный программный продукт полностью соответствует поставленным задачам и создан строго по техническому заданию.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Нильсен, Я. Веб-дизайн: книга Якоба Нильсена / Я. Нильсен. - М.: Символ, 2015. - 512 c.

2. Ван, Тассел Д. Стиль, разработка, эффективность, отладка и испытания программ / Ван Тассел Д. - М.: Мир, 2017. - 332 c.

3. Чебан О.О. Интернет - технологии в школе. Сервер данных MySQL. М: Вопросы Интернет-образования, № 32-33.

4. Седерхольм, Д. Пуленепробиваемый веб-дизайн. Библиотека специалиста / Д. Седерхольм. - СПб.: Питер, 2012. - 304 c.

5. Hogan, B. HTML5 and CSS3. Web development according to new generation standards / B. Hogan. - Moscow: Peter, 2018. - 474 p.

6. Дакетт Джон HTML и CSS. Разработка и дизайн веб-сайтов

7. Дакетт Джон Javascript и jQuery. Интерактивная веб-разработка

8. <https://gb.ru/geek_university/developer/programmer/fullstack>

9. <https://learn.javascript.ru/>

10. <https://developer.mozilla.org/ru/docs/Learn>

11. <https://nodejs.org/en/>

12. <https://sass-scss.ru/>

13. <https://git-scm.com/>

14. <https://habr.com/ru/all/>

15. <https://www.w3.org/>

# ПРИЛОЖЕНИЯ

## Приложение А – используемые файлы для создания веб-сайта.

**STYLE.CSS**

Определение основных стилей сайта.

@charset "UTF-8";

\* {

  margin: 0;

  padding: 0;

  -webkit-box-sizing: border-box;

          box-sizing: border-box;

  font-family: "PT Sans", "Helvetica", "Arial", sans-serif;

}

html,

body {

  height: 100%;

  -webkit-animation: fade-up 2s;

          animation: fade-up 2s;

}

.wrapper-column {

  min-height: 100%;

  display: -webkit-box;

  display: -ms-flexbox;

  display: flex;

  -webkit-box-orient: vertical;

  -webkit-box-direction: normal;

      -ms-flex-direction: column;

          flex-direction: column;

}

main,

.category {

  -webkit-box-flex: 1;

      -ms-flex: 1 1 auto;

          flex: 1 1 auto;

}

:root {

  --left-right-space: 10px;

}

@font-face {

  font-family: Myriad Pro;

  src: url(../fonts/MYRIADPRO-REGULAR.OTF);

}

.favorite,

.shopping-basket {

  display: -webkit-box;

  display: -ms-flexbox;

  display: flex;

  -webkit-box-align: center;

      -ms-flex-align: center;

          align-items: center;

  -webkit-box-pack: center;

      -ms-flex-pack: center;

          justify-content: center;

}

.shopping-basket img {

  width: 40px;

  -webkit-box-ordinal-group: 0;

      -ms-flex-order: -1;

          order: -1;

  margin-right: var(--left-right-space);

}

/\*-----------Категори---------\*/

.wrap-product {

  display: -ms-grid;

  display: grid;

  -ms-grid-columns: (minmax(300px, 1fr))[auto-fit];

      grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

  margin: var(--left-right-space);

  grid-gap: 20px;

  margin-bottom: 20px;

}

.product-title {

  height: 40px;

  font-weight: 800;

  font-size: 16px;

}

.category {

  text-align: center;

}

.product {

  text-align: center;

  text-align: center;

  -webkit-box-shadow: 0px 0px 13px;

          box-shadow: 0px 0px 13px;

  border-radius: 20px;

  padding: 50px 5px;

  -webkit-transition: 0.5s;

  transition: 0.5s;

}

.goods-img img {

    width: calc(280px + (500 - 280) \* (100vw - 320px) / (1440 - 320));

  }

  .header-title,

  .shopping-basket {

    font-size: calc(22px + (40 - 22) \* (100vw - 320px) / (1440 - 320));

  }

  .left-nav\_\_ul li a {

    font-size: calc(16px + (42 - 16) \* (100vw - 320px) / (1440 - 320));

  }

  .left-nav\_\_ul {

    width: calc(280px + (342 - 280) \* (100vw - 320px) / (1440 - 320));

  }

  .main-title {

    font-size: calc(28px + (50 - 28) \* (100vw - 320px) / (1440 - 320));

  }

  .item h5 {

    font-size: calc(26px + (30 - 26) \* (100vw - 320px) / (1440 - 320));

  }

  .main h3 {

    font-size: calc(26px + (40 - 26) \* (100vw - 320px) / (1440 - 320));

  }

}

@media (min-width: 200px) and (max-width: 840px) {

  .cart-order {

    -webkit-box-orient: vertical;

    -webkit-box-direction: normal;

        -ms-flex-direction: column;

            flex-direction: column;

  }

  .form-control input,

  textarea#address {

    max-width: 100% !important;

  }

}

@media (max-width: 1050px) and (min-width: 320px) {

  .item:nth-child(3n) {

    margin-top: none;

  }

}

@-webkit-keyframes fade-up {

  from {

    opacity: 0;

  }

  to {

    opacity: 1;

  }

}

@keyframes fade-up {

  from {

    opacity: 0;

  }

  to {

    opacity: 1;

  }

}

@media (min-width: 769px) {

  .item:hover {

    -webkit-transform: scale(1.1);

            transform: scale(1.1);

  }

  .left-nav\_\_ul > li > a:hover, .product:hover {

    -webkit-transform: scale(1.05);

            transform: scale(1.05);

  }

  .add-to-cart:hover {

    -webkit-filter: drop-shadow(2px 4px 6px black);

            filter: drop-shadow(2px 4px 6px black);

  }

}

/\*# sourceMappingURL=style.css.map \*/

**CARD.JS**

Реализация корзины на сайте. Создание объекта корзины. И вывод товара на экран.

'use strict';

let cart = {};

document.querySelectorAll('.add-to-cart').forEach(function(el) {

    el.addEventListener('click', addToCart);

});

if (localStorage.getItem('cart')) {

    cart = JSON.parse(localStorage.getItem('cart'));

    ajaxGetGoodsInfo();

}

function addToCart() {

    let goodsId = this.dataset.goods\_id;

    if (cart[goodsId]) {

        cart[goodsId]++;

    } else {

        cart[goodsId] = 1;

    }

    console.log(cart);

    ajaxGetGoodsInfo();

}

function ajaxGetGoodsInfo() {

    updateLocalStorageCart();

    fetch('/get-goods-info', {

            method: 'POST',

            body: JSON.stringify({

                key: Object.keys(cart)

            }),

            headers: {

                'Accept': 'application/json',

                'Content-Type': 'application/json'

            }

        })

        .then(function(responce) {

            return responce.text();

        })

        .then(function(body) {

            console.log(body);

            showCart(JSON.parse(body));

        })

}

function showCart(data) {

    let out = '';

    let total = 0;

    for (let key in cart) {

        out += `<div class="cart-order">

                    <a href="/goods?id=${key}" class="cart-order-link">

                        <img src="img/${data[key]['image']}" width="100" class="cart-oreder-image">

                    ${data[key]['name']}</a>

                        <div class="cart-order-bottom">

                            <p class="cart-order-text">КОЛИЧЕСТВО: ${cart[key]}</p>

                            <i class="far fa-minus-square cart-minus cart-oreder-btn" data-goods\_id="${key}"></i>

                            <i class="far fa-plus-square cart-plus cart-oreder-btn" data-goods\_id="${key}"></i>

                        </div>

                </div>`;

        total += cart[key] \* data[key]['cost'];

    }

    out += `<h3>ИТОГО: ${formatPrice(total)} ₽</h3>`;

    try {

        document.querySelector('#cart-nav').innerHTML = out;

    } catch (error) {

        console.log(error);

    }

    document.querySelectorAll('.cart-minus').forEach(function(element) {

        element.addEventListener('click', cartMinus);

    });

    document.querySelectorAll('.cart-plus').forEach(function(element) {

        element.addEventListener('click', cartPlus);

    });

}

function cartPlus() {

    let goodsId = this.dataset.goods\_id;

    cart[goodsId]++;

    ajaxGetGoodsInfo();

}

function cartMinus() {

    let goodsId = this.dataset.goods\_id;

    if (cart[goodsId] - 1 > 0) {

        cart[goodsId]--;

    } else {

        delete(cart[goodsId]);

    }

    ajaxGetGoodsInfo();

}

function updateLocalStorageCart() {

    localStorage.setItem('cart', JSON.stringify(cart));

}

function formatPrice(price) {

    return price.toFixed(2).replace(/\d(?=(\d{3})+\.)/g, '$& ');

}

**NAV.JS**

Реализация левого блока навигации на главной странице. Данные берутся с сервера.

'use strict';

function getCategoryList() {

    fetch('/get-category-list', {

        method: 'POST',

    }).then(function(responce) {

        return responce.text();

    }).then(function(body) {

        showCategoryList(JSON.parse(body));

    });

}

function showCategoryList(data) {  let out = '<ul class="left-nav\_\_ul">';

    for (let i = 0; i < data.length; i++) {

        out += `<li><a href="/cat?id=${data[i]['id']}">${data[i]['category']}</a></li>`;

    }

    out += '</ul>';

    try {

        document.querySelector('.left-nav').innerHTML = out;

    } catch (error) {

        console.log(error);

    }

}

getCategoryList();

**ORDER.JS**

Проверка введенных данных клиента при покупке на странице корзины и вывод соответствующего уведомления.

'use strict';

try {

    document.querySelector('#shop-order').addEventListener('submit', function (el) {

        el.preventDefault();

        let username = document.querySelector('#username').value.trim();

        let phone = document.querySelector('#phone').value.trim();

        let email = document.querySelector('#email').value.trim();

        let address = document.querySelector('#address').value.trim();

        if (!document.querySelector('#rule').checked) {

            //С правилами не согласен

            Swal.fire({

                title: 'Предупреждение',

                text: 'Примите правила',

                // type: 'info',

                icon: 'error',

                confirmButtonText: 'ОК',

            });

            return false;

        }

        if (username == '' || phone == '' || email == '' || address == '') {

            Swal.fire({

                title: 'Предупреждение',

                text: 'Заполните все поля',

                // type: 'info',

                icon: 'error',

                confirmButtonText: 'ОК',

            });

        }

        fetch('/finish-order', {

                method: 'POST',

                body: JSON.stringify({

                    'username': username,

                    'phone': phone,

                    'address': address,

                    'email': email,

                    'key': JSON.parse(localStorage.getItem('cart'))

                }),

                headers: {

                    'Accept': 'application/json',

                    'Content-Type': 'application/json'

                }

            })

            .then(function (response) {

                return response.text();

            })

            .then(function (body) {

                if (body == 1) {

                    Swal.fire({

                        title: 'Всё хорошо',

                        text: 'Удачно',

                        type: 'info',

                        // icon: 'error',

                        confirmButtonText: 'ОК',

                    });

                } else {

                    Swal.fire({

                        title: 'Возникли проблемы',

                        text: 'Ошибка',

                        // type: 'info',

                        icon: 'error',

                        confirmButtonText: 'ОК',

                    });

                }

            })

    });

} catch (error) {}

**CAT.PUG**

Рендер PUG шаблона категорий товара

include header

section.category.wrapper-column

    h2 #{cat[0]['category']}

    div.wrap-product.center

        each val, index in goods

                div.product

                    a(href="goods?id="+val['id'])

                        p.product-title #{val.name}

                        img(src="img/"+val.image alt="foto" class="product\_\_img")

                        h3.product-cost #{val.cost.toFixed(2).replace(/\d(?=(\d{3})+\.)/g, '$& ')} ₽

                    button(

                        class="add-to-cart"

                        data-goods\_id=val['id']

                        ) В КОРЗИНУ

include footer

**FOOTER.PUG**

Рендер PUG шаблона футера (нижней части сайта) с подключенными JavaScript файлами

footer.center.footer

    h2

        a(

           href="/"

           class="footer-link"

        ) ТехноЭра

    p.footer-rights ВСЕ ПРАВА ЗАЩИЩЕНЫ

script(src="//cdn.jsdelivr.net/npm/sweetalert2@11")

script(src='js/nav.js')

script(src='js/card.js')

script(src='js/order.js')

**HEADER.PUG**

Рендер PUG шаблона хэдэра (верхней части сайта) и блока с подключенными скомпилированными css файлами.

doctype html

html(lang="ru")

    head

        meta(charset="UTF-8")

        meta(http-equiv="X-UA-Compatible" content="IE=edge")

        meta(name="viewport" content="width=device-width, initial-scale=1.0")

        title ТехноЭра

        link(rel="shortcut icon" href="img/fav.ico" type="image/x-icon")

        link(rel="stylesheet" href="css/normalize.css")

        link(rel="stylesheet" href="css/style.css")

        script(src="https://kit.fontawesome.com/543f278849.js")

    body

    header.center

        nav.wrapper

            h1

                a(href="/" class="header-title")  ТехноЭра

            ul.nav

                li

                    h1

                        a(href="/order" class="shopping-basket") Корзина

                            img(src="img/shopping-basket-solid.svg" alt="Корзина")

**INDEX.PUG**

Рендер PUG шаблона главной страницы с подключёнными модулями блока навигации и контента

include header

div.wrapper-column

    include nav\_left

    include main

include footer

**ORDER.PUG**

Рендер PUG шаблона блока заказа

include header

section.wrap.center.wrapper-column

    div.cart

        h3.cart-title КОРЗИНА

        div#cart-nav

        h3.cart-final СОЗДАТЬ ЗАКАЗ

    form#shop-order(action="")

        div(class='form-control')

            label(for='username') ИМЯ

            input(

                type='text',

                id='username',

                placeholder='ВАШЕ ИМЯ',

                )

        div(class='form-control')

            label(for='phone') ТЕЛЕФОН

            input(

                type='text',

                id='phone',

                placeholder='+79999999999',

                )

        div(class='form-control')

            label(for='email') ПОЧТА

            input(

                type='text',

                id='email',

                placeholder='mail@example.com',

                )

        div(class='form-control')

            label(for='address') АДРЕС

            textarea(

                id='address',

                placeholder='ГДЕ ВЫ',

                )

        div(class='form-rule')

            label(for='rule')

                input(

                    type='checkbox',

                    id='rule',

                    value='1'

                    )

                span  УСЛОВИЯ И ПОЛОЖЕНИЯ

        div(class='form-control text-right')

            button(class='button-primary' type='submit') ЗАКАЗАТЬ

include footer

script(src='js/order.js')

**APP.JS**

Точка входа приложения. Подключение нужных модулей NodeJS, настройка соединения с базой данных, обработка полученных данных с Backend’a, реализация отправки сообщения на почту.

'use strict';

const express = require('express');

const app = express();

const port = 5000;

/\*\* Настройка области видимости папок\*/

app.use(express.static('public'));

/\*\* Задаём шаблонизатор\*/

app.set('view engine', 'pug');

/\*\* Подключение модуля Mysql2\*/

const mysql = require('mysql2');

/\*\* Настраивание модуля\*/

app.use(express.json());

/\*\* Подключение nodemailer\*/

const nodemailer = require('nodemailer');

/\*\*Подключение SweetAlert2 \*/

// const Swal = require('sweetalert2');

let con = mysql.createConnection({

    host: 'localhost',

    user: 'root',

    password: 'root',

    database: 'shop'

});

process.env["NODE\_TLS\_REJECT\_UNAUTHORIZED"] = 0;

app.listen(port, () => {

    console.log(`node express work on ${port}`);

});

//Главная

app.get('/', function(req, res) {

    let cat = new Promise(function(resolve, reject) {

        con.query(

            "select id, name, cost, image, category from (select id,name,cost,image,category, if(if(@curr\_category != category, @curr\_category := category, '') != '', @k := 0, @k := @k + 1) as ind   from goods, ( select @curr\_category := '' ) v ) goods where ind < 3",

            function(error, result, field) {

                if (error) return reject(error);

                resolve(result);

            }

        );

    });

    let catDescription = new Promise(function(resolve, reject) {

        con.query(

            "SELECT \* FROM category",

            function(error, result, field) {

                if (error) return reject(error);

                resolve(result);

            }

        );

    });

    Promise.all([cat, catDescription]).then(function(value) {

        console.log(value[1]);

        res.render('index', {

            goods: JSON.parse(JSON.stringify(value[0])),

            cat: JSON.parse(JSON.stringify(value[1])),

        });

    });

});

//Категории

app.get('/cat', (req, res) => {

    let catId = req.query.id;

    let cat = new Promise((res, rej) => {

        con.query('SELECT \* FROM category WHERE id = ' + catId, (error, result) => {

            if (error) rej(error);

            res(result);

        });

    });

    let goods = new Promise((res, rej) => {

        con.query('SELECT \* FROM goods WHERE category=' + catId, (error, result) => {

            if (error) rej(error);

            res(result);

        });

    });

    Promise.all([cat, goods]).then((value) => {

        console.log(value);

        res.render('cat', {

            cat: JSON.parse(JSON.stringify(value[0])),

            goods: JSON.parse(JSON.stringify(value[1]))

        })

    });

});

//Товары

app.get('/goods', function(req, res) {

    // console.log(req.query.id);

    con.query('SELECT \* FROM goods WHERE id=' + req.query.id, function(error, result, fiels) {

        if (error) throw error;

        res.render('goods', {

            goods: JSON.parse(JSON.stringify(result)),

        });

    });

});

//Корзина

app.get('/order', function(req, res) {

    res.render('order');

});

app.post('/get-category-list', function(req, res) {

    // console.log(req);

    con.query('SELECT id, category FROM category', function(error, result, fiels) {

        if (error) throw error;

        // console.log(result);

        res.json(result);

    });

});

app.post('/get-goods-info', function(req, res) {

    // console.log(req.body.key);

    if (req.body.key.length != 0) {

        con.query('SELECT id, name, image, cost FROM shop.goods WHERE id IN(' + req.body.key.join(',') + ');', function(error, result, fiels) {

            if (error) throw error;

            console.log(result);

            let goods = {};

            for (let i = 0; i < result.length; i++) {

                goods[result[i]['id']] = result[i];

            }

            res.json(goods);

        });

    } else {

        res.send('0');

    }

});

app.post('/finish-order', function(req, res) {

    console.log(req.body);

    if (req.body.key.length != 0) {

        let key = Object.keys(req.body.key);

        con.query(

            'SELECT id,name,cost FROM goods WHERE id IN (' + key.join(',') + ')',

            function(error, result, fields) {

                if (error) throw error;

                console.log(result);

                sendMail(req.body, result).catch(console.error);

                res.send('1');

            });

    } else {

        res.send('0');

    }

});

async function sendMail(data, result) {

    let res = '<h2>Order in lite shop</h2>';

    let total = 0;

    let res\_text = '';

    for (let i = 0; i < result.length; i++) {

        res += `<p>${result[i]['name']} - ${data.key[result[i]['id']]} - ${result[i]['cost'] \* data.key[result[i]['id']]} ₽</p>`;

        total += result[i]['cost'] \* data.key[result[i]['id']];

    }

    console.log(res);

    res += '<hr>';

    res += `Итого ${total} ₽`;

    res += `<hr>Телефон: ${data.phone}`;

    res += `<hr>Имя: ${data.username}`;

    res += `<hr>Адрес: ${data.address}`;

    res += `<hr>Почта: ${data.email}`;

    let testAccount = await nodemailer.createTestAccount();

    let transporter = nodemailer.createTransport({

        host: "smtp.ethereal.email",

        port: 587,

        secure: false, // true for 465, false for other ports

        auth: {

            user: testAccount.user, // generated ethereal user

            pass: testAccount.pass // generated ethereal password

        }

    });

    let mailOption = {

        from: '<89606138203@mail.ru>',

        to: "89606138203@mail.ru," + data.email,

        subject: "Магазин ТехноЭра",

        text: res\_text,

        html: res

    };

    let info = await transporter.sendMail(mailOption);

    console.log("MessageSent: %s", info.messageId);

    console.log("PreviewSent: %s", nodemailer.getTestMessageUrl(info));

    return true;

}

**SQL**

Создание двух основных таблиц в базе данных.

-- Создание таблицы категорий

DROP TABLE IF EXISTS `shop`.`category`;

CREATE TABLE `shop`.`category` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `category` VARCHAR(255) NULL DEFAULT NULL,

  `description` TEXT NULL DEFAULT NULL,

  `image` VARCHAR(500) NULL DEFAULT NULL,

  PRIMARY KEY (`id`));

-- Создание таблицы товаров

DROP TABLE IF EXISTS `shop`.`goods`;

CREATE TABLE `shop`.`goods` (

  `id` INT(11) NOT NULL,

  `name` VARCHAR(255) NULL,

  `description` TEXT NULL DEFAULT NULL,

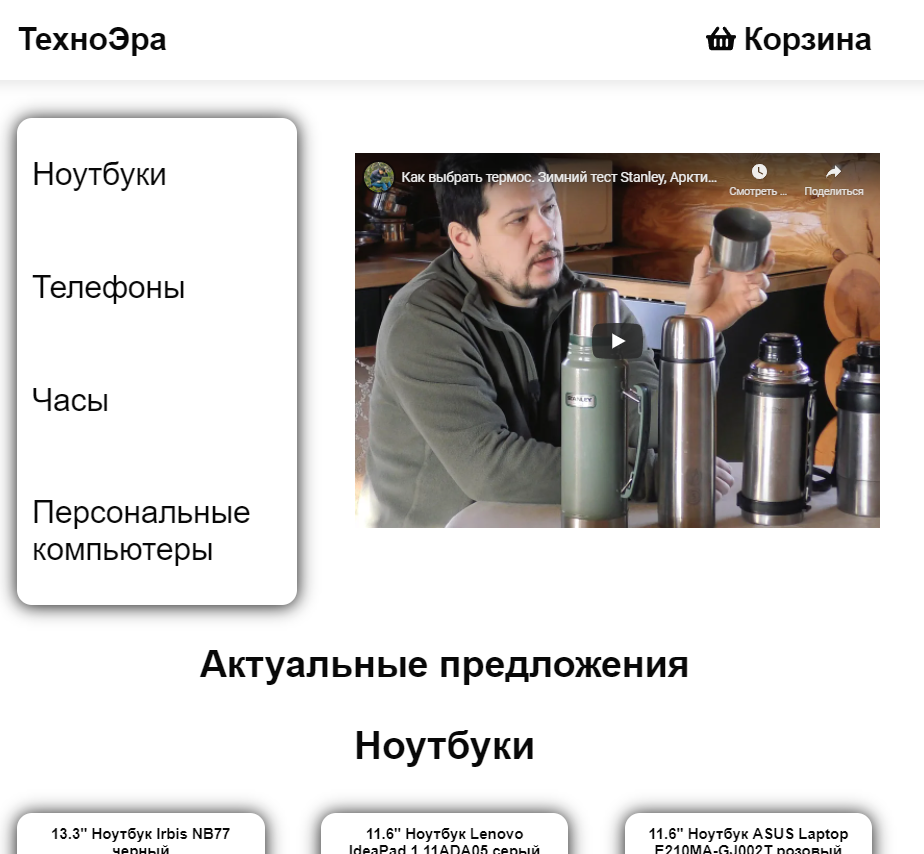
  `cost` DOUBLE NULL DEFAULT NULL,

  `image` VARCHAR(500) NULL DEFAULT NULL,

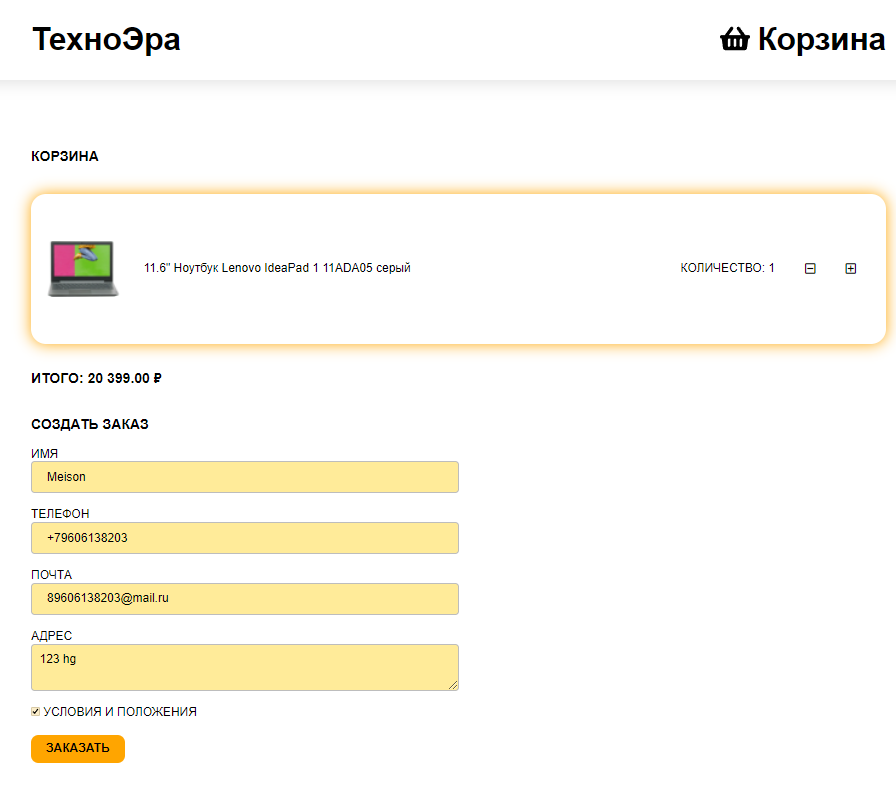
  `category` INT(11) NULL,

  PRIMARY KEY (`id`));

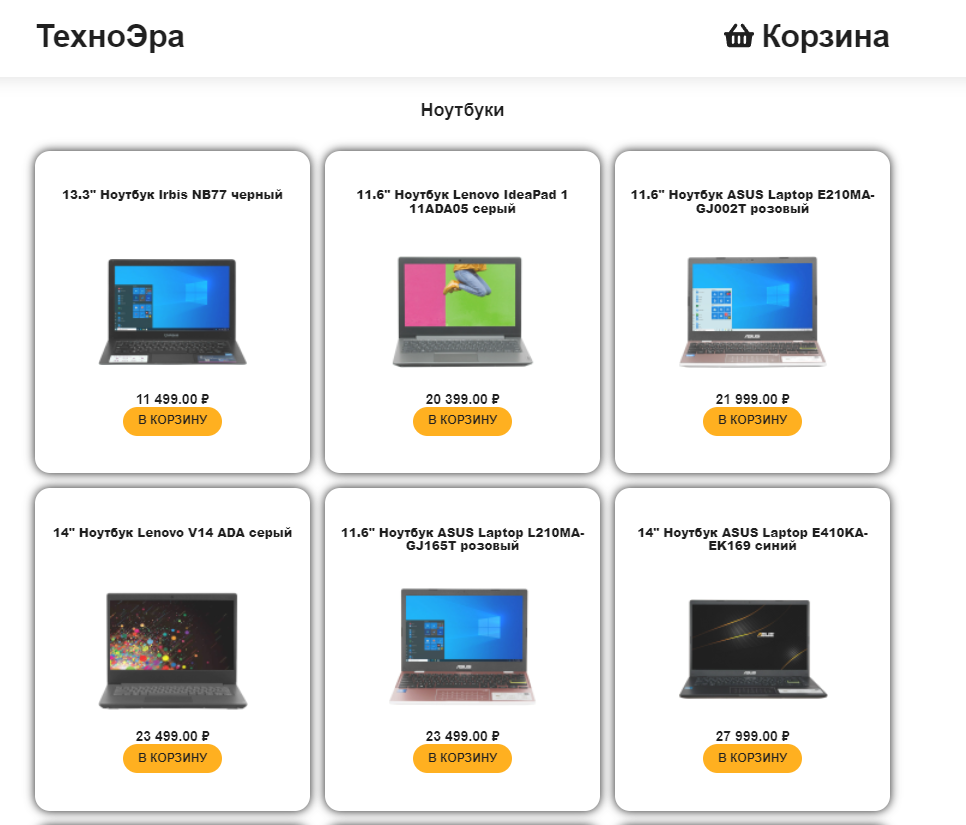
## Приложение Б – визуализация программы в браузере.



Главная страница



Страница корзины



Страница категории товара



Страница товара