ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ИРКУТСКОЙ ОБЛАСТИ  
«АНГАРСКИЙ ТЕХНИКУМ СТРОИТЕЛЬНЫХ ТЕХНОЛОГИЙ»

**Курсовая работа**

Тема: Разработка web-сайта «Хобби и увлечения»

Разработала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сагатбекова А.А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Денисюк А.В.

Ангарск, 2025

**СОДЕРЖАНИЕ**

**Введение………………………………………………………………………4**

**I ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

* 1. **Инструменты, выбранные для создания сайта……………….6**

1.1.1 Figma: проектирование дизайна с учетом деталей…………….7

1.1.2 Visual Studio Code: универсальная среда для кода…………….9

**1.2 Технологии, примененные в проекте…………………………12**

**1.2.1 Frontend-технологии…………………………………………..12**

1.2.1.1 HTML: основа структуры………………………………...….12

1.2.1.2 CSS: оформление и адаптивность…………………………...13

1.2.1.3 JavaScript: интерактивность и динамика…………………....13

1.2.1.4 Bootstrap: готовые решения для дизайна…………………...14

1.2.1.5 Handlebars: динамическое отображение контента………....14

**1.2.2** **Backend-технологии**……………………………………….….15

1.2.2.1 MySQL: управление данными……………………………….15

1.2.2.2 Node.js и Express: серверная логика…………………………15

**2.2.3** **Вспомогательные Backend**-**технологии…………………….16**

**II ПРАКТИЧЕСКАЯ ЧАСТЬ**

**2.1 Разработка дизайна сайта**……………………………………...19

**2.2 Верстка сайта на основе макета**………………………………20

2.2.1 Frontend часть…………………………………………………...20

2.2.2 Backend часть…………………………………………...……....27

**2.3 Реализация базы данных**……………………………………….32

**2.4. Тестирование**…………………..………………………………..34

**Заключение…………………………………………………………………39**

**Список используемой литературы………………………………………41**

**Введение**

Интернет сегодня — это мощная коммуникационная платформа, объединяющая миллионы пользователей. Веб-разработка стала одной из самых востребованных областей программирования, так как наличие сайта необходимо для бизнеса, чтобы привлекать клиентов и представлять информацию о товарах, услугах и контактах. Всемирная паутина (World Wide Web) насчитывает миллионы сайтов, доступных через веб-браузеры, что делает интернет ключевым инструментом современности.

Данная курсовая работа посвящена разработке веб-сайта "MANY HOBBIES" (МНОГО ХОББИ), посвящённого хобби и увлечениям.

**Тема актуальна** из-за растущей потребности людей в отдыхе, самовыражении и развитии вне работы. Сайт поможет пользователям находить информацию о хобби, общаться с единомышленниками и получать доступ к товарам и услугам.

**Цель работы** — создание удобного и информативного веб-сайта, позволяющего легко ориентироваться в мире хобби.

Для этого были поставлены следующие задачи:

* Разработать сайт с использованием современных технологий веб-программирования (HTML5, CSS3, JavaScript, Node.js, MySQL).
* Создать привлекательный и интуитивно понятный дизайн.
* Разработать удобный интерфейс для комфортного взаимодействия.
* Реализовать функционал для оставления отзывов, предложений и жалоб.

**Объект исследования** в рамках данной курсовой работы — это

процесс разработки и реализации веб-сайта, посвящённого хобби и увлечениям. Особое внимание уделяется этапам проектирования, программирования, тестирования и внедрения сайта, чтобы он соответствовал ожиданиям пользователей и решал поставленные задачи. Предмет исследования — применение современных веб-технологий, таких как HTML5 для разметки страниц, CSS3 для стилизации и обеспечения адаптивности, JavaScript для реализации интерактивных элементов, а также серверных технологий для обработки данных и взаимодействия с базой данных. Эти технологии были выбраны благодаря их универсальности, широкой поддержке и способности обеспечить высокую производительность и функциональность сайта.

Проект "MANY HOBBIES" объединяет современные технологии, продуманный дизайн и функциональность, чтобы стать полезной платформой для вдохновения, общения и развития. Работа рассматривает этапы создания сайта, включая проектирование, программирование и внедрение интерактивных функций, таких как комментарии и форма предложений.

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**1.1 Инструменты, выбранные для создания сайта**

Разработка веб-сайта — это многогранный процесс, который требует внимания к множеству деталей: от проектирования дизайна до написания кода, настройки сервера и обеспечения удобства для пользователей. Проект "MANY HOBBIES" — это платформа, посвящённая хобби и увлечениям, чтобы вдохновлять людей на новые занятия. Идея сайта родилась из моего личного увлечения творчеством: мне всегда нравилось рисовать, вышивать и пробовать новые виды рукоделия, а ещё я люблю делиться своими находками с другими. Поэтому "MANY HOBBIES" стал для меня способом создать пространство, где люди могут находить идеи для новых увлечений, таких как музыка, изготовление свечей, кулинария, фотография, танцы, путешествия, садоводство и даже робототехника.

На сайте пользователи могут читать подробные описания каждого хобби, находить полезные ссылки статьи или видеоуроки, оставлять комментарии, чтобы поделиться опытом, добавлять любимые занятия в избранное и отправлять свои идеи через форму обратной связи. Аудитория сайта — это люди всех возрастов и интересов: подростки, ищущие хобби для творческого самовыражения, взрослые, которые хотят попробовать что-то новое после работы, и даже старшее поколение, желающее освоить, например, вязание или садоводство. Особое внимание уделялось творческим хобби, потому что они мне близки, но я постаралась сделать сайт разнообразным, чтобы каждый нашёл что-то для себя.

Для создания "MANY HOBBIES" требовалось подобрать инструменты, которые позволили бы воплотить эту идею в жизнь. В процессе разработки использовались Figma для проектирования дизайна, Visual Studio Code для написания кода. Эти инструменты были выбраны за их доступность, удобство и способность покрыть все этапы создания сайта — от идеи до готового продукта.

**1.1.1 Figma: проектирование дизайна с учетом деталей**

**Figma** — это онлайн-инструмент для создания макетов и прототипов интерфейсов, который стал популярным благодаря своей универсальности и простоте использования. Он позволяет работать как в браузере, так и через десктопное приложение, поддерживает совместную работу в реальном времени и предоставляет множество функций для дизайна. Figma была выбрана для проекта, потому что она бесплатна в базовой версии, имеет интуитивно понятный интерфейс и широко используется в индустрии, что делает её надёжным выбором. Ещё одним важным фактором стало наличие обучающих материалов и шаблонов, которые помогли быстро освоить инструмент, даже без большого опыта в дизайне.

**Возможности Figma и их преимущества**

Figma предоставляет обширный набор инструментов для создания макетов. Можно настраивать цвета, шрифты, размеры, градиенты и работать с векторной графикой, что позволяет создавать сложные элементы, такие как кнопки, карточки, иконки, фоны. Функция "Auto Layout" автоматически выравнивает элементы, сохраняя аккуратный вид макета, и позволяет задавать отступы между объектами, что особенно полезно при работе с повторяющимися элементами, такими как списки или карточки. "Layout Grid" предоставляет сетку для точного размещения объектов, что помогает добиться симметрии и выровнять элементы с высокой точностью. Например, можно задать шаг сетки в 8 пикселей, чтобы все элементы были кратны этому значению, что делает дизайн более профессиональным.

Ещё одна полезная возможность — адаптивный дизайн: Figma позволяет создавать версии макета для разных устройств, таких как компьютеры (1920x1080 пикселей), планшеты (768x1024 пикселя) и смартфоны (375x667 пикселей), чтобы заранее увидеть, как сайт будет выглядеть на каждом из них. Это помогает избежать проблем с отображением, таких как обрезка текста или смещение кнопок.

Инструмент поддерживает экспорт элементов дизайна, таких как изображения (в форматах PNG, JPG, SVG), иконки или CSS-стили, что упрощает перенос дизайна в код. Например, можно извлечь код цвета кнопки, размер шрифта или параметры тени и сразу использовать их в CSS. Figma также поддерживает плагины, которые расширяют её функциональность. Есть плагины для генерации цветовых палитр (например, "Coolors" для подбора гармоничных цветов), проверки контрастности (чтобы текст был читаемым на фоне), создания иконок (например, "Iconify" для доступа к тысячам иконок). Figma имеет активное сообщество, где можно найти готовые шаблоны и наборы компонентов. Например, существуют UI-киты с готовыми карточками, кнопками и формами, которые можно адаптировать под свои нужды, что ускоряет процесс проектирования.

Ещё одно преимущество Figma — это её облачная природа. Все проекты сохраняются в облаке, что позволяет работать с ними из любого места, где есть интернет, и не беспокоиться о потере данных. Функция совместной работы позволяет нескольким пользователям одновременно работать над одним макетом, оставлять комментарии и обсуждать изменения в реальном времени, хотя в данном проекте я работала одна. Figma также предоставляет историю версий, что позволяет вернуться к предыдущим состояниям макета, если что-то было изменено случайно. Например, можно восстановить макет, который был создан неделю назад, если текущая версия оказалась неудачной.

**Применение в "MANY HOBBIES"**

Для "MANY HOBBIES" Figma стала первым шагом в создании сайта. С её помощью был разработан макет, который охватывал все ключевые разделы. Figma стала незаменимым инструментом, который позволил создать гармоничный и продуманный внешний вид сайта, отражающий мою любовь к творчеству.

**1.1.2 Visual Studio Code: универсальная среда для кода**

**Visual Studio Code (VS Code)** — это редактор кода от Microsoft, который заслужил популярность благодаря своей универсальности, скорости и поддержке множества языков программирования. Он был выбран для проекта, потому что является бесплатным, работает на разных операционных системах (Windows, macOS, Linux) и предоставляет обширный набор функций, упрощающих разработку. Ещё одним преимуществом стало наличие расширений, которые добавляют полезные возможности, и активное сообщество, где можно найти решения любых проблем.

**Возможности Visual Studio Code**

VS Code поддерживает множество языков программирования, включая HTML, CSS, JavaScript, SQL, Python, PHP и многие другие, что делает его универсальным инструментом для веб-разработки и не только. Программа предоставляет подсветку синтаксиса, которая выделяет ключевые слова разными цветами (например, теги HTML — синим, строки — зелёным), что улучшает читаемость кода. Автодополнение, которое предлагает варианты продолжения кода, например, при написании background в CSS сразу появляются подсказки вроде background-color или background-image, что ускоряет работу. Автоматическое форматирование кода выравнивает отступы и переносы строк, делая код аккуратным, а проверка ошибок в реальном времени подчёркивает проблемы, такие как незакрытые теги или неправильные синтаксические конструкции, ещё до запуска сайта.

Встроенный терминал позволяет выполнять команды, такие как установка библиотек (npm install), запуск сервера (node app.js(server.js)) или выполнение Git-команд (git commit), без необходимости переключаться на другие приложения

Одна из ключевых особенностей — поддержка расширений, которые добавляют функциональность. Расширение Live Server запускает локальный сервер и автоматически обновляет страницу в браузере при изменении кода, что позволяет сразу видеть результаты. Prettier автоматически форматирует код, выравнивая отступы и переносы строк, чтобы код соответствовал стандартам (например, 2 пробела для отступов). Другие полезные расширения включают "Auto Rename Tag" (автоматически переименовывает закрывающий тег HTML, если меняется открывающий), "CSS Peek" (показывает стили CSS прямо из HTML) и "Bracket Pair Colorizer" (выделяет парные скобки разными цветами, что помогает избежать ошибок в синтаксисе).

Программа поддерживает работу с несколькими проектами одновременно через "рабочие пространства" (Workspaces), что позволяет открывать разные папки с проектами в одном окне. VS Code также предоставляет встроенный поиск по файлам, который позволяет находить текст не только в текущем файле, но и во всём проекте, что полезно для больших проектов с множеством файлов.

**Применение в "MANY HOBBIES"**

VS Code стал основным инструментом для создания "MANY HOBBIES". В этой программе создавались все страницы сайта: главная с категориями, страницы хобби с описаниями и комментариями, формы для входа и регистрации, а также раздел "О нас". Подсветка ошибок помогала сразу замечать недочёты, автодополнение ускоряло написание кода: при написании CSS-свойства margin сразу предлагались варианты, такие как margin-top или margin-left, что экономило время.

Встроенный терминал использовался для установки серверных библиотек (например, npm install express), запуска сайта (node server.js) и управления зависимостями (npm update). Расширения, такие как Prettier и ESLint, обеспечивали чистоту и правильность кода: Prettier автоматически выравнивал отступы, а ESLint указывал на потенциальные ошибки, например, использование устаревших методов или отсутствие точек с запятой. Расширение "Auto Rename Tag" помогало при работе с HTML, автоматически обновляя парные теги, а "CSS Peek" позволял быстро переходить к стилям, связанным с классами в HTML.

VS Code также использовался для управления структурой проекта. Все файлы были организованы в папки: public для статических ресурсов (CSS, JavaScript, изображения), views для шаблонов Handlebars, routes для маршрутов Express и models для работы с базой данных. Встроенный поиск помогал находить нужные строки кода: например, можно было ввести "button" и найти все места, где используются кнопки, чтобы изменить их стиль. VS Code сделал процесс разработки организованным и удобным, предоставив всё необходимое для работы над сайтом в одном месте.

**1.2 Технологии, примененные в проекте**

Разработка сайта "MANY HOBBIES" потребовала использования различных технологий для создания функционального, удобного и надёжного продукта. Этот проект был для меня способом реализовать идею платформы, где пользователи могут исследовать хобби, оставлять комментарии, добавлять занятия в избранное и делиться идеями через обратную связь. Чтобы достичь этой цели, я разделила технологии на фронтенд — часть, которую видят и используют пользователи, и бэкенд — серверную логику, которая управляет данными и безопасностью. Выбор каждой технологии был обусловлен её возможностями, совместимостью и простотой освоения.

**1.2.1 Frontend-технологии**

**Frontend** — это лицо сайта, которое взаимодействует с пользователем. Эти технологии отвечают за структуру, стиль, интерактивность и динамическое отображение контента.

**1.2.1.1 HTML: основа структуры**

**HTML (HyperText Markup Language)** — представляет собой язык разметки, предназначенный для определения структуры веб-страниц, указывая расположение заголовков, текстовых блоков, изображений, кнопок и форм. Разработанный Тимом Бернерсом-Ли в 1990 году в рамках проекта CERN, HTML прошёл эволюцию от первой версии (HTML 1.0, 1993) до современного стандарта HTML5, утверждённого W3C в 2014 году.

HTML работает с тегами: <h1> для заголовков, <p> для абзацев, <img> для изображений, <a> для ссылок и <div> для контейнеров. Семантические теги, такие как <header>, <nav>, <main>, <article> и <footer>, улучшают структуру, делая сайт доступным для поисковых систем и экранных читалок. HTML5 расширил возможности: тег <video> позволяет встраивать видео, <canvas> — рисовать графику, <input type="date"> — добавлять поля для дат, а атрибуты вроде required, minlength и placeholder упрощают валидацию форм и улучшают пользовательский опыт.

В "MANY HOBBIES" HTML стал фундаментом. HTML обеспечил чёткую структуру, на которой я построила функционал.

**1.2.1.2 CSS: оформление и адаптивность**

**CSS (Cascading Style Sheets)** — это язык стилей, который отвечает за внешний вид сайта, включая цвета, шрифты, отступы, расположение элементов и анимации. Я выбрала CSS за его интеграцию с HTML и способность создавать адаптивный дизайн, что важно для удобства на разных устройствах.

CSS использует свойства, такие как color, background-color, font-size, margin, padding, border и display (например, flex или grid) для управления стилями. Медиа-запросы @media (max-width: 768px) позволяют адаптировать дизайн под размер экрана. CSS3 добавляет box-shadow для теней, border-radius для скругления углов, transition для плавных изменений, transform для масштабирования и animation для сложных эффектов, что делает интерфейс современным.

В "MANY HOBBIES" CSS придал страницам приятный вид. Он обеспечил визуальную согласованность и удобство использования.

**1.2.1.3 JavaScript: интерактивность и динамика**

**JavaScript** — это язык программирования, который добавляет сайту интерактивность, позволяя реагировать на действия пользователей и обновлять контент без перезагрузки. Я выбрала JavaScript за его универсальность и широкое применение в веб-разработке, что позволило мне реализовать сложные функции.

JavaScript поддерживает обработку событий (addEventListener для кликов, onchange для полей ввода), манипуляцию DOM (querySelector, createElement), асинхронные запросы (fetch или XMLHttpRequest), анимации (requestAnimationFrame) и работу с данными (JSON.parse). Встроенные объекты, такие как Math.random() для генерации чисел и Date.now() для временных меток, расширяют его возможности.

В "MANY HOBBIES" JavaScript обеспечил интерактивность. Сделал сайт отзывчивым и удобным для пользователей.

**1.2.1.4 Bootstrap: готовые решения для дизайна**

**Bootstrap** — это фреймворк, предоставляющий готовые стили и компоненты для создания адаптивных интерфейсов. Выбор пал на Bootstrap за его способность упрощать процесс разработки, обеспечивать единообразие дизайна и поддерживать работу на разных устройствах.

Bootstrap включает 12-колоночную сетку с классами, такими как col-md-6 или col-lg-4, для гибкого расположения элементов. Он предлагает стили для кнопок (btn, btn-primary), форм (form-control, form-group), карточек (card) и навигации (navbar). JavaScript-компоненты включают модальные окна (modal), выпадающие меню (dropdown), карусели (carousel) и всплывающие подсказки (tooltip). Утилиты, такие как m-3 (отступы), p-2 (внутренние отступы), text-center (выравнивание), d-none (скрытие) и rounded (скругление углов), упрощают мелкую настройку. Анимации (fade, slide) и темы (bg-light, bg-dark) добавляют динамику и гибкость.

В "MANY HOBBIES" Bootstrap стал основой оформления. Добавил аккуратности, ускорил разработку, обеспечив современный и адаптивный дизайн.

**1.2.1.5 Handlebars: динамическое отображение контента**

**Handlebars** — это шаблонизатор, который позволяет создавать страницы, автоматически заполняя их данными с сервера. Удобен в проекте за его простоту, интеграцию с серверной частью и способность разделять логику и представление, что упростило поддержание сайта.

Handlebars использует синтаксис {{variable}} для вставки данных, #if для условий, #each для циклов и partial для повторяющихся блоков, таких как шапка или подвал. Пользовательские помощники (helpers), например, formatDate, позволяют форматировать вывод, а частичные шаблоны (partials) повторно используют код.

В "MANY HOBBIES" Handlebars управляет динамическим контентом, обеспечивая гибкость и масштабируемость, упрощая добавление нового контента.

**1.2.2 Backend-технологии**

**Backend** — это серверная часть сайта, которая управляет данными, логикой и безопасностью. Эти технологии обеспечивают обработку запросов, хранение информации и защиту пользователей.

**1.2.2.1 MySQL: управление данными**

**MySQL** — это реляционная система управления базами данных, которая организованно хранит информацию. Я выбрала MySQL за её надёжность, удобство, бесплатный доступ и совместимость с серверными технологиями, что позволило эффективно управлять данными сайта.

MySQL использует SQL для операций: SELECT для выборки, INSERT для добавления, UPDATE для изменения и DELETE для удаления. Поддерживаются типы данных, такие как VARCHAR(255) для текста, INT для чисел, DATETIME для дат и TEXT для больших объёмов данных. Внешние ключи обеспечивают целостность, индексы ускоряют запросы, а транзакции гарантируют атомарность операций. Триггеры автоматически выполняют действия, например, обновление счётчиков.

В "MANY HOBBIES" MySQL организовал хранение и обеспечил быструю и надёжную работу с данными.

**1.2.2.2 Node.js и Express: серверная логика**

**Node.js** — это среда выполнения JavaScript на сервере, а **Express** — фреймворк для упрощения серверной разработки. Я выбрала их за возможность использовать один язык (JavaScript) для всего проекта и высокую производительность, что было важно для обработки запросов.

Node.js работает на основе событийного цикла, обрабатывая запросы асинхронно через async/await или Promises, что позволяет обслуживать множество пользователей одновременно. Express предоставляет маршрутизацию (app.get, app.post), middleware для обработки данных (express.json, express.urlencoded) и интеграцию с базами данных. npm позволяет подключать модули, такие как mysql2 или express-handlebars.

В "MANY HOBBIES" Node.js и Express управляют сервером, Express настроил маршруты. Эти технологии обеспечили быстродействие и масштабируемость.

**2.2.3 Вспомогательные технологии**

**1.2.3.1 bcrypt: защита паролей**

**bcrypt** — это библиотека для хэширования паролей, обеспечивающая их безопасность. Я выбрала bcrypt за его надёжность и устойчивость к атакам, что важно для защиты данных пользователей.

bcrypt использует алгоритм с солью, генерируемой автоматически, и настраиваемое количество раундов (по умолчанию 10) для контроля сложности хэширования. Функции bcrypt.hash шифруют пароль, а bcrypt.compare проверяют его при входе, сравнивая хэш с введённым значением.

В "MANY HOBBIES" bcrypt шифрует пароли.

**1.2.3.2 express-session: управление сессиями**

**express-session** — это модуль для управления сессиями пользователей в Express, позволяющий сохранять данные между запросами. Он прост в настройке и поддерживает персонализацию.

express-session использует cookie для хранения идентификатора сессии, поддерживает параметры secret для подписи, maxAge для срока действия и secure для HTTPS. Данные сессии можно хранить в памяти или внешнем хранилище, таком как Redis.

В "MANY HOBBIES" express-session управляет авторизацией.

**1.2.3.3 dotenv** : Управление конфигурацией через файл .env.

Это библиотека или модуль, который существует в большинстве языков программирования и позволяет управлять переменными окружения через файл с расширением .env. С помощью Dotenv можно хранить все переменные окружения в одном файле, а затем автоматически загружать их в среду выполнения. Например, при работе с Node.js файл с расширением .env преобразуется в процесс.env — глобальный объект, который предоставляет доступ ко всем доступным переменным окружения.

**ПРАКТИЧЕСКАЯ ЧАСТЬ**

Создание платформы «MANY HOBBIES» началось с изучения запросов целевой аудитории — творческих людей, ищущих идеи для досуга, площадку для обмена опытом и общения с энтузиастами. Я поставила цель разработать ресурс, который объединит разнообразные увлечения в одном месте. Ключевые задачи проекта были сформированы по направлениям:

**1. Базовый функционал:**

**Каталог хобби:** Карточки с названием, описанием, тематической категорией и визуальным контентом.

**Персонализация:** Система «Избранное» для сохранения понравившихся активностей (доступна после регистрации).

**Сообщество:** Возможность оставлять комментарии к материалам для зарегистрированных участников.

**Обратная связь:** Форма для предложения новых хобби с последующей модерацией администратором.

**Авторизация:** Личные кабинеты пользователей с настройками профиля и управлением контентом. Реализация админ-панели.

**2. Юзабилити:**

**Простая навигация:** Чёткое разделение на категории, главную страницу и профиль, фиксированное меню для быстрого доступа.

**Адаптивность:** Корректное отображение на любых устройствах — от десктопов до мобильных устройств.

**Интуитивность:** Минимум сложных элементов — интерфейс ориентирован на мгновенное понимание даже новичками.

**3. Защита данных:**

**Безопасные сессии:** Использование express-session и express-mysql-session для контроля авторизации.

**Byctript:** Для безопасности паролей.

* 1. **Разработка дизайна сайта**

Работа началась с проектирования интерфейса в Figma, инструменте, который стал для меня цифровым холстом. Figma позволила визуализировать идеи, протестировать взаимодействие и создать единый стиль платформы.

**В Figma были созданы:**

* **Макеты страниц:** Главная страница с каруселью категорий, страницы профиля, хобби, категорий, авторизации, регистрации, предложений и "О нас". Каждая страница продумывалась с точки зрения удобства пользователя.
* **Цветовая палитра:** Я выбрала яркие и тёплые оттенки: розовый (#ff69b4) для кнопок и акцентов, фиолетовый (#6a0dad) для заголовков, белый (#fff) и светло-розовый (#fff5f7) для фона. Эти цвета создавали ощущение уюта.
* **Типографика:** Шрифты "Days One" для основного текста и "Jaro" для логотипа и заголовков, загруженные через Google Fonts, обеспечили читаемость и эстетику.
* **Компоненты интерфейса:** Кнопки с закруглёнными углами, карточки хобби, формы для комментариев и авторизации, модальное окно для входа, адаптивное гамбургер-меню для мобильных устройств.
* **Адаптивность:** Макеты тестировались для экранов от 576px (мобильные) до 1200px (десктопы). Медиа-запросы обеспечили корректное отображение на всех устройствах, включая планшеты.

Интерактивный прототип в Figma позволил смоделировать пользовательский путь: от открытия главной страницы до добавления хобби в избранное или написания комментария. Я представила, как пользователь переходит от категории "Спорт" к хобби "Бег", читает описание и добавляет его в избранное. Этот этап заложил фундамент для разработки, обеспечив чёткое видение структуры и дизайна.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1. Первая разработка для карточки хобби

Пример кода (Figma → CSS): Цвета и шрифты из Figma были экспортированы в src/public/css/style.css для стилизации карточек хобби.



Рисунок 2. Часть кода стилей

**2.2 Верстка сайта на основе макета**

Проект "Many Hobbies" построен на современных веб-технологиях, тщательно подобранных для обеспечения функциональности, производительности и удобства разработки. Ниже представлено подробное описание каждой технологии и инструмента с примерами их применения в проекте.

**2.2.1 Frontend часть**

**JavaScript**

JavaScript — это универсальный язык программирования, который стал сердцем проекта "Many Hobbies". Его история началась в 1995 году, когда Брендан Айк создал его для браузера Netscape, чтобы сделать веб-страницы интерактивными. С тех пор JavaScript эволюционировал из клиентского языка в мощную платформу для серверной разработки благодаря Node.js. Его асинхронная природа, основанная на событийном цикле, делает его идеальным для обработки множества запросов, а богатая экосистема библиотек (NPM) предоставляет инструменты для любых задач, все эти возможности сыграли большую роль при создании сайта.

JavaScript использовался на двух уровнях: серверном и клиентском. На сервере, через Node.js, JavaScript обрабатывал HTTP-запросы, взаимодействовал с базой данных и управлял сессиями. Например, я писала маршруты для авторизации и загрузки хобби, которые обрабатывали данные асинхронно, обеспечивая быструю работу платформы. На клиенте JavaScript оживлял интерфейс, добавляя интерактивные элементы, такие как карусель категорий на главной странице или валидацию форм. Я выбрала JavaScript за его универсальность, которая позволила мне использовать один язык для всей логики приложения, упрощая разработку и отладку. Карусель категорий, например, была для меня важным элементом, так как она привлекала внимание пользователей к разнообразию хобби, и JavaScript сделал её плавной и отзывчивой.

**Пример кода (src/public/js/script.js)**: Карусель категорий.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3. Скрипт карусели с главной страницы

Скрипт управляет каруселью, автоматически переключая слайды каждые 3 секунды. Пользователь может остановить прокрутку, наведя мышь, что улучшает взаимодействие.

**HTML/ CSS**

HTML и CSS — это основа веб-разработки, формирующая структуру и внешний вид страниц. HTML, созданный в 1990 году Тимом Бернерсом-Ли, определяет структуру контента, такую как заголовки, параграфы и формы. CSS, появившийся в 1996 году, отвечает за стилизацию, позволяя задавать цвета, шрифты, макеты и анимации. В "Many Hobbies" HTML создавал "скелет" страниц, таких как карточки хобби, формы авторизации и навигационное меню. CSS, в свою очередь, придавал платформе визуальную привлекательность, добавляя тени, анимации и адаптивные макеты.

Я использовала HTML для создания семантически правильной структуры, что улучшало доступность и SEO. Например, карточки хобби были построены с использованием <div> с классами Bootstrap, но я добавила кастомные стили в src/public/css/style.css, чтобы соответствовать дизайну из Figma. CSS был для меня способом воплотить дизайн платформы на основе макета: яркие цвета, такие как розовый (#ff69b4) и фиолетовый (#6a0dad), создавали дружелюбную атмосферу, а эффекты наведения делали интерфейс интерактивным. Адаптивность была критически важна, так как я хотела, чтобы платформа выглядела идеально на любом устройстве, от смартфона до ноутбука. Для этого я использовала медиа-запросы и Bootstrap, но кастомные стили позволили мне добавитьуникальности, например, анимации подъёма карточек при наведении.

**Пример кода (src/views/layouts/main.handlebars):** Часть кода «шапки» сайта.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4. Часть кода из файла шапки сайта с использование html

Часть данного кода создает шапку сайта с соблюдением адаптивности и функциональности.

**Пример кода (src/public/css/style.css)**: Стили для меню.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 5. Пример кода из файла с карточкой хобби (использование css)

Часть данного кода представляет собой первую секцию из раздела меню, отображающую определенные шрифты и цвета.

**Handlebars**

Handlebars — это шаблонизатор для JavaScript, созданный в 2011 году для упрощения динамического рендеринга HTML. Он позволяет разделять логику и представление, используя шаблоны с заполнителями (например, {{variable}}), которые заполняются данными на сервере или клиенте. Handlebars поддерживает хелперы для форматирования данных и условные выражения, что делает его мощным инструментом для веб-приложений.

В проекте я использовала Handlebars для динамического отображения данных из базы данных, таких как списки хобби, комментарии и профили пользователей. Шаблоны, такие как hobby.handlebars, позволили мне создавать страницы, которые автоматически заполнялись данными, например, названием хобби или его изображениями. Я настроила кастомные хелперы, такие как formatDate для форматирования дат и truncate для обрезки текста, чтобы улучшить читаемость. Handlebars был для меня идеальным выбором, так как он упростил рендеринг страниц, сохраняя код чистым и модульным.

**Пример кода (src/views/hobby.handlebars)**: Страница хобби.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6. Пример кода из файла с карточкой хобби

Данная часть кода представляет собой «карточку хобби», которая осуществляет возможность добавлять комментарии и отправлять в избранное.

**Bootstrap 5, Bootstrap Icons**

Bootstrap 5 — это популярный CSS-фреймворк, для создания адаптивных и современных интерфейсов. Он предоставляет готовые компоненты, такие как сетки, формы, кнопки и навигационные меню, которые упрощают вёрстку. Bootstrap Icons — это библиотека SVG-иконок, дополняющая фреймворк визуальными элементами.

Я использовала Bootstrap 5 для создания адаптивной вёрстки, обеспечивая корректное отображение на всех устройствах. Например, сетка Bootstrap (row и col-md-4) использовалась для отображения изображений хобби в трёхколоночном формате. Bootstrap Icons добавили иконки, такие как сердечко для кнопки "Добавить в избранное", что улучшило визуал. Bootstrap был для меня спасением, так как позволил быстро создать прототип интерфейса, а затем кастомизировать его с помощью собственных стилей.

**Пример кода (src/views/hobby.handlebars)**: Сетка изображений.

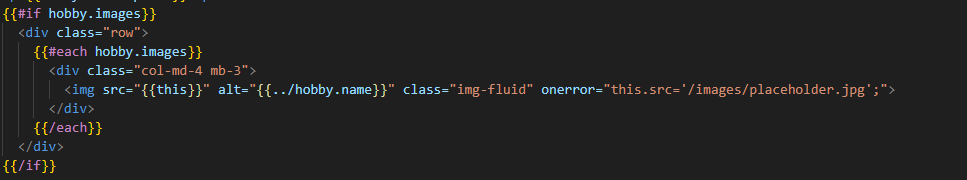


Рисунок 7. Пример кода для показа сетки bootstrap

Класс col-md-4 создаёт трёхколоночную сетку для изображений, адаптирующуюся к размеру экрана.

**Custom CSS/JavaScript**

Кастомные стили и скрипты были необходимы для добавления уникальности интерфейсу, которую не мог обеспечить Bootstrap. CSS позволил мне воплотить дизайн из Figma, добавляя анимации, тени и кастомные цвета. JavaScript, в свою очередь, добавил интерактивность, такую как валидация форм на клиенте или обработка событий в карусели.

Я написала кастомные стили в src/public/css/style.css для карточек хобби, навигационного меню и кнопок. JavaScript-скрипты в src/public/js управляли каруселью, валидацией форм и гамбургер-меню.

**Пример кода (src/public/js/auth.js)**: Валидация формы.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 8. Пример кода проверяющий данные

Скрипт проверяет заполненность полей формы перед отправкой, улучшая функционал сайта.

**2.2.2 Backend часть**

**Node.js**

Node.js — это серверная среда выполнения JavaScript, созданная в 2009 году Райаном Далем на основе движка V8 от Google Chrome. Она позволяет запускать JavaScript вне браузера, используя асинхронную модель на основе событийного цикла, что делает её идеальной для масштабируемых приложений. Node.js стал революцией в веб-разработке, так как позволил использовать один язык (JavaScript) для клиентской и серверной логики, упрощая разработку.

Node.js был серверной платформой, обеспечивающей обработку HTTP-запросов, взаимодействие с базой данных и управление сессиями. Я выбрала Node.js за его производительность и богатую экосистему модулей (NPM), которая предоставила инструменты, такие как Express, MySQL2 и bcrypt. Node.js позволил мне создать асинхронные маршруты, например, для загрузки хобби или авторизации, что обеспечило быструю работу платформы даже при большом количестве пользователей.

**Пример кода (server.js)**: Инициализация сервера.

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 9. Часть кода инициализирующего сервер

Код настраивает сервер Express с middleware для JSON

**Express.js**

Express.js — это минималистичный и гибкий фреймворк для Node.js, Он предоставляет инструменты для маршрутизации, обработки HTTP-запросов и интеграции middleware, таких как express-session или express-fileupload. Express стал стандартом де-факто для Node.js-приложений благодаря своей простоте и мощности.

Express.js был основой серверной части, позволяя мне быстро настроить RESTful API и маршруты для авторизации, управления хобби, категориями и пользовательскими профилями. Я использовала Express для обработки запросов, таких как POST /login или GET /hobby/:id, и интеграции с базой данных через MySQL2. Express упростил работу с middleware, что позволило мне добавить функциональность, такую как управление сессиями или загрузка файлов, без лишнего кода.

**Пример кода (src/controllers/auth.js)**: Маршрут авторизации.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 9. Часть кода, отражающая работу маршрутов

Данная часть отражает работу маршрута и циклов при введении тех или иных данных. Обрабатывает вход пользователя, проверяя имя и пароль, и сохраняет сессию при успехе.

**bcrypt**

bcrypt — это библиотека для хеширования паролей.Она использует алгоритм Blowfish, добавляя "соль" (случайные данные) для защиты от атак по словарю и радужных таблиц. bcrypt автоматически генерирует уникальную соль для каждого пароля и позволяет задавать "стоимость" хеширования (work factor), увеличивая вычислительную сложность для повышения безопасности.

bcrypt обеспечивает безопасность пользовательских паролей, хешируя их при регистрации и сравнивая при авторизации. Я выбрала bcrypt за его надёжность и простоту использования в Node.js. Каждый пароль хешировался с work factor 10, что обеспечивало баланс между безопасностью и производительностью

**Пример кода (src/controllers/auth.js)**: Регистрация с хешированием.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 10. Работа хэширования пароля

**express-session, express-mysql-session**

express-session — это middleware для Express, управляющее сессиями пользователей, а express-mysql-session — это хранилище, позволяющее сохранять данные сессий в базе данных MySQL. Сессии позволяют "запоминать" авторизованных пользователей, храня их данные (например, ID и имя) между запросами

Я использовала express-session для хранения данных авторизованных пользователей, таких как их ID, имя и статус администратора. express-mysql-session сохранял сессии в базе данных, что обеспечивало надёжность и масштабируемость по сравнению с хранением в памяти. Настройка сессий была для меня важным аспектом, так как я хотела, чтобы пользователи могли оставаться авторизованными при переходе между страницами, не вводя пароль заново.

**Пример кода (server.js)**: Настройка сессий.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 11. Серверный код для настройки сессий

Код настраивает сессии с хранением в MySQL, используя секрет из .env.

**MySQL2**

MySQL2 — библиотека для асинхронного взаимодействия с MySQL, поддерживающая промисы. Это позволило мне оптимизировать запросы и повысить производительность. Я использовала MySQL2 для всех операций с базой данных, от выборки хобби до логирования посещений.

**Пример кода (db/connection.js)**: Пул соединений.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 12. Соединение с базой данных

**dotenv**

dotenv управлял конфигурацией, храня чувствительные данные (пароли, ключи) в файле .env. Это повысило безопасность и упростило настройку.

**Пример кода (.env)**: Использование .env.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 13 Данные из .env

Предоставляется наполненность файла.

**2.3 Реализация базы данных**

Отдельно хочется выделить создание базы данных.

MySQL — это реляционная база данных с открытым исходным кодом, ставшая одной из самых популярных СУБД в мире. Она использует SQL для управления данными и поддерживает сложные структуры, такие как внешние ключи, индексы и JSON-поля. MySQL известна своей производительностью, надёжностью и широкой поддержкой в веб-разработке.

MySQL была основой для хранения всех данных: пользователей, хобби, категорий, комментариев, избранного, предложений и логов. Я спроектировала базу many\_hobbies с таблицами, связанными внешними ключами, чтобы обеспечить целостность данных. Например, таблица hobbies ссылается на categories через category\_id, а favorites связывает users и hobbies.

Структура базы many\_hobbies:

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 14. Структура базы данных

**Подробнее о каждой таблице:**

**Таблица users** предоставляет полную информацию о каждом пользователе, начиная с базовых данных и заканчивая возможностью добавлять администраторов в один клик.

**Таблица hobbies** хранит информацию о хобби: Название, распределение по категориям, Описание, путь к изображениям.

**Таблица comments** хранит комментарии пользователей к хобби, отображая время, текст комментария, и краткие данные о пользователе.

**Таблица favorites** содержит информацию об избранных хобби, связывает 2 таблицы, создавая путь.

**Таблица categories** хранит категории хобби.

**Таблица hobby\_suggestions** содержит предложения новых хобби от пользователей с названием, описанием и статсуом предложения(для администратора).

**Таблица sessions** хранит данные о пользовательских сессиях для управления авторизацией: Идентификатор сессии, Дату и время истечения сессии.

**Таблица logs** в базе данных служит для ведения логов (журналов).

**Примеры используемых мною скриптов при создании базы:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 15. Применение CREATE TABLE и INSERT INTO

**2.4. Тестирование**

После создания полной структуры сайта я решила провести ручное тестирование.

Ручное тестирование я проводила лично, проверяя платформу с точки зрения обычного пользователя и администратора. Я хотела убедиться, что все функции работают, а интерфейс удобен. Тестирование было для меня возможностью увидеть, как платформа ведет себя в различных ситуациях.

**Тестирование с точки зрения пользователя**

Я начала тестирование, представляя себя обычным пользователем. Открыв сайт в браузере Yandex, я прошла несколько сценариев:

1. **Регистрация:** На странице главной странице я ввела данные в модальное окно регистрации: имя пользователя testuser, email test@example.com и пароль Test123!. После отправки формы я проверила в Dbeaver, что пользователь добавлен в таблицу users:

Изображение выглядит как текст, программное обеспечение, веб-страница, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 16. Проверка заполнения базы

1. **Вход:** После регистрации меня перенесло в личный кабинет, где я осуществила выход из аккаунта и далее вход, также через модальное окно на главной странице. После введения данных меня также перенаправило на страницу личного кабинета:

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 17. Страница личного кабинета после входа

1. **Работоспособность страниц:** Я осуществила переходы по страницам: Главная, Избранное, О нас, Контакты, Карточки категорий и хобби, при переходе не было замечено ошибок.
2. **Функционал:**

**Проверка добавления аватара, обновления почты и пароля:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 18. Проверка личного кабинета

**Добавление комментария:**

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 19. Проверка добавления комментария

**Избранное:**

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 20. Проверка добавления в избранное

**Предложения:** Название хобби: Вышивание, Описание: Шить крестиком с категорией: Рукоделие.

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 21. Проверка отправки предложения

**Тестирование с точки зрения администратора**

Затем я протестировала функции администратора, войдя под учётной записью с правами: логин-admin, пароль-adminpassword. У администратора в личном кабинете находится специальная панель для редактирования сайта, которую и требуется протестировать.

1. **Добавление хобби:** Я создала новое хобби: название "Танцы", категория "Спорт", описание "Энергичное увлечение". После отправки формы я проверила таблицу hobbies:

Рисунок 22. Проверка сохранения добавленного хобби

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Изображение выглядит как текст, снимок экрана, Бренд, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 23. Отображение на сайте

1. **Модерация предложений**: Проверка на наличие предложения, которое мы отправили и его одобрение.

Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 23. Проверка обработки предложений в админ-панели

1. **Добавление категории:** В название категории прописываем «Образование».

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 24. Проверка добавления категории

1. **Отображение списка хобби:** Все хобби отображаются.

Изображение выглядит как текст, снимок экрана, шаблон, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 25. Часть данных со вкладки «Мои хобби»

Все функции, которые доступны обычному пользователю, также отлично работают без каких-либо ошибок.

**Личное впечатление**

Тестирование было для меня увлекательным процессом с которым я сталкивалась на протяжении всего проекта. Конечное тестирование вызвало наибольшее удовольствие, поскольку все функции уже были отлажены.

**Результаты**

Тестирование подтвердило, что проект работает корректно:

* Пользовательские функции — регистрация, вход, просмотр хобби, комментарии, избранное — выполняются безошибочно.
* Административные функции — добавление хобби, категорий и модерация — удобны и надёжны.
* Интерфейс выглядит красиво, сохранена адаптивность.

Я убедилась, что платформа готова вдохновлять пользователей.

**Заключение**

В рамках курсового проекта было разработан веб-сайт «Many Hobbies» в переводе «Много Хобби», предназначенный для управления информацией о хобби, их категориями, пользовательскими предложениями и комментариями. Сайт создан с использованием современного стека технологий, включающего серверный фреймворк Node.js с Express, шаблонизатор Handlebars для динамического рендеринга интерфейса и реляционную базу данных MySQL для хранения данных. Основной целью проекта было создание удобного сайта для поиска новой информации по популярным хобби и поиска новых увлечений с функциональным личным кабинетом, обеспечивающим взаимодействие пользователей с контентом, а также предоставляющий администраторам инструменты для модерации.

Основные достижения проекта:

1. **Реализация функционала личного кабинета**: Разработан интерфейс с вкладками для редактирования профиля, просмотра хобби, управления комментариями, добавления новых хобби и категорий, а также обработки пользовательских предложений. Использование Bootstrap обеспечило адаптивный и современный дизайн.
2. **Серверная логика**: Настроены маршруты для обработки запросов (GET и POST) с использованием Express, реализована работа с сессиями для аутентификации пользователей и управление правами доступа (администратор/обычный пользователь).
3. **Интеграция с базой данных**: Спроектирована структура базы данных с таблицами для пользователей, хобби, категорий, комментариев и предложений. SQL-запросы обеспечивают эффективное извлечение и обновление данных.
4. **Обработка файлов**: Реализована загрузка аватаров и изображений для хобби с сохранением на сервере и интеграцией путей в базу данных.
5. **Безопасность**: Использован модуль bcrypt для хеширования паролей, а также проверки прав доступа для ограничения действий неавторизованных пользователей.

В процессе разработки были решены многочисленные технические задачи, включая настройку шаблонов Handlebars, отладку SQL-запросов и устранение ошибок рендеринга интерфейса. Особое внимание уделялось отображению пользовательских предложений: были устранены проблемы с некорректным отображением названий хобби и добавлены кнопки для их модерации, что улучшило функциональность админ-панели.

Разработанный сайт успешно демонстрирует применение современных веб-технологий для решения практической задачи. Проект позволил углубить знания в области серверной разработки, работы с базами данных и создания динамических интерфейсов, а также развить навыки отладки и оптимизации кода. «Many Hobbies» представляет собой полноценное решение для управления пользовательским контентом, которое может быть использовано как основа для более сложных систем в будущем.

**Список используемой литературы**

**Литература:**

1. Дакетт, Дж. HTML и CSS. Разработка и дизайн веб-сайтов / Дж. Дакетт; пер. с англ. — М.: Эксмо, 2017. — 480 с.
2. Мейер, Э. А. CSS. Полное руководство / Э. А. Мейер, Э. Вейл; пер. с англ. — 4-е изд. — СПб.: Питер, 2020. — 816 с.
3. Флэнаган, Д. JavaScript. Полное руководство / Д. Флэнаган; пер. с англ. — 7-е изд. — М.: Вильямс, 2020. — 704 с.
4. Кантиллон, М. Node.js в действии / М. Кантиллон, М. Хартер, Т.Дж. Холоуэйчак; пер. с англ. — 2-е изд. — М.: ДМК Пресс, 2018. — 448 с.
5. Форта, Б. SQL за 10 минут / Б. Форта; пер. с англ. — 5-е изд. — М.: Вильямс, 2019. — 288 с.
6. Сингх, В. MySQL. Полное руководство / В. Сингх; пер. с англ. — М.: ДМК Пресс, 2021. — 512 с.
7. Летт, Я. Bootstrap 4. Быстро и просто / Я. Летт; пер. с англ. — М.: Питер, 2019. — 208 с.
8. Браун, И. Web Development with Node and Express / И. Браун. — 2-е изд. — Sebastopol: O'Reilly Media, 2019. — 340 с.
9. Хавербеке, М. Eloquent JavaScript / М. Хавербеке. — 3-е изд. — San Francisco: No Starch Press, 2018. — 472 с.
10. Бим, Б. Learning Handlebars / Б. Бим. — Birmingham: Packt Publishing, 2016. — 150 с.

**Интернет-ресурсы:**

1. Mozilla Developer Network (MDN). Документация по HTML, CSS и JavaScript. — URL: <https://developer.mozilla.org/ru/docs/Web> (дата обращения: 14.05.2025).
2. Официальная документация Bootstrap. — URL: <https://getbootstrap.com/docs/5.3/> (дата обращения: 14.05.2025).
3. Официальная документация Node.js. — URL: <https://nodejs.org/en/docs/> (дата обращения: 14.05.2025).
4. Официальная документация Express.js. — URL: <https://expressjs.com/ru/> (дата обращения: 14.05.2025).
5. Официальная документация Handlebars.js. — URL: <https://handlebarsjs.com/> (дата обращения: 14.05.2025).
6. Официальная документация MySQL. — URL: <https://dev.mysql.com/doc/> (дата обращения: 14.05.2025).
7. HTML Academy. Интерактивные курсы по HTML, CSS и JavaScript. — URL: <https://htmlacademy.ru/> (дата обращения: 14.05.2025).
8. Learn JavaScript. Учебный ресурс по JavaScript и Node.js. — URL: <https://learn.javascript.ru/> (дата обращения: 14.05.2025).
9. Хабр. Статьи по веб-разработке. — URL: <https://habr.com/ru/hub/webdev/> (дата обращения: 14.05.2025).
10. WebReference. Руководства по веб-технологиям. — URL: <https://webref.ru/> (дата обращения: 14.05.2025).