МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ КУРСОВЫХ ПРОЕКТОВ

Нижний Новгород
2025

Министерство просвещения Российской Федерации
Нижегородский государственный педагогический университет имени Козьмы Минина
(Мининский университет)

Факультет информационных технологий
Кафедра информационных систем и цифровых сервисов в управлении

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ КУРСОВЫХ ПРОЕКТОВ

специальность 09.02.09 Веб-разработка

Нижний Новгород
2025

УДК 004(07)
ББК 32.81р3
М 545

М 545 **Методические рекомендации по выполнению курсовых проектов** / составитель А.В. Поначугин – Нижний Новгород: Мининский университет, 2025. – 24 с

Методические рекомендации по выполнению курсовых проектов предназначены для студентов очной формы обучения по специальности 09.02.09 Веб-разработка. Рекомендации содержат общие положения, содержание курсовых проектов, требования к оформлению курсовых проектов.

УДК 004(07)
ББК 32.81р3

Оглавление

Общие положения

Учебная дисциплина «Интеграция информационных ресурсов с другими системами в сети Интернет» относится к обязательной программе учебного плана для студентов очной формы обучения по специальности 09.02.09 «Веб-разработка».

Целью освоения дисциплины является формирование у обучающихся комплекса знаний, умений и навыков, необходимых для проектирования и разработки веб-ориентированных решений, обеспечивающих интеграцию информационных ресурсов с внешними системами и сервисами в сети Интернет. Особое внимание уделяется практическому применению современных технологий веб-разработки, инструментов взаимодействия распределенных систем и методов обмена данными между разнородными платформами.

Задачами дисциплины являются:

- Ознакомление с основами построения распределенных информационных систем и архитектурными подходами к интеграции веб-приложений;
- Изучение принципов взаимодействия веб-ресурсов с использованием протоколов REST, SOAP, GraphQL и форматов обмена данными JSON, XML;
- Освоение методов разработки и подключения программных интерфейсов (API) сторонних сервисов, включая платежные системы, базы данных и облачные платформы;
- Предоставление теоретических знаний и практических навыков по созданию, отладке и документированию интеграционных решений в условиях сетевого взаимодействия.

В рамках дисциплины обучающиеся должны получить навыки использования современных подходов к интеграции информационных ресурсов, проектирования архитектуры взаимодействия веб-приложений с внешними системами, а также применения инструментов, обеспечивающих надежный и безопасный обмен данными в сети Интернет. Заканчивается дисциплина выполнением курсового проекта, состоящей из практической части - разработки веб-приложения или модуля интеграции, демонстрирующего взаимодействие с внешними сервисами, и теоретической части - написания пояснительной записки с обоснованием выбранных технологий, описанием архитектуры и результатов тестирования.

Базой для выполнения курсового проекта являются знания и умения, полученные обучающимися в ходе изучения предшествующих дисциплин, связанных с основами веб-технологий, программирования и проектирования информационных систем.

Настоящие методические рекомендации содержат инструкции по выполнению практической части курсового проекта, описание требований к оформлению и содержанию пояснительной записки, а также пример выполнения пояснительной записки, иллюстрирующий основные этапы разработки интеграционного решения.

Курсовой проект выполняется под руководством преподавателя, ведущего дисциплину в области веб-разработки, интеграции информационных систем и смежных направлений. Оценка работы производится руководителем курсового проекта и преподавателями, проводящие лабораторные и практические занятия. Комиссия оценивает соответствие выполненной работы утвержденному заданию, качество реализации интеграционного решения, полноту пояснительной записки, а также уровень защиты и презентации результатов.

Таким образом, выполнение курсового проекта способствует подготовке студента к решению ключевой профессиональной задачи - приобретению знаний, умений и навыков самостоятельной разработки веб-ориентированных решений, обеспечивающих интеграцию информационных ресурсов с внешними системами в сети Интернет.

Цели и основные требования, предъявляемые к курсовому проекту:

Курсовой проект по дисциплине «Интеграция информационных ресурсов с другими системами в сети Интернет» предполагает знание студентами основных принципов проектирования веб-ориентированных решений, понимание архитектуры взаимодействия распределенных систем, владение методами работы с программными интерфейсами (API), а также навыки использования современных языков и сред веб-разработки, и имеет следующие цели:

– развитие у студентов самостоятельности в проектировании интеграционных решений и творческих способностей при разработке архитектуры взаимодействия веб-ресурсов с внешними системами, углубления и закрепления теоретических знаний в области интеграции информационных ресурсов;

– приобретение практических навыков использования современных сред разработки, инструментов тестирования API и отладки сетевого взаимодействия;

– приобретение навыков написания серверного и клиентского кода, реализующего обмен данными между системами с использованием протоколов REST, SOAP, GraphQL и форматов JSON, XML;

– закрепление умений пользоваться технической документацией, спецификациями открытых API, справочными материалами по веб-технологиям, а также верно оформлять проектную и техническую документацию.

Курсовой проект выполняется студентом в соответствии с заданием, которое составляется преподавателем или преподавателем совместно со студентом и выдается студенту.

В задании на курсовой проект указывается ее тема и пояснение, включающее в себя техническое задание (функциональные требования к разрабатываемому интеграционному решению, перечень внешних систем или сервисов для взаимодействия, требования к форматам обмена данными), исходные данные (при необходимости - примеры запросов, структура баз данных, описание API сторонних сервисов), объем работы, рекомендуемую литературу и документацию, а также сроки выполнения работы по графику.

Содержание и структура курсового проекта

Курсовой проект состоит из разработки веб-приложения или интеграционного модуля, обеспечивающего взаимодействие информационных ресурсов с внешними системами, его тестирования, а также описания в пояснительной записке процессов проектирования, разработки и тестирования созданного решения, включая принципы работы с ним и особенности интеграции.

Выполнение курсового проекта включает следующие этапы:

Анализ цели и постановки задачи на курсовой проект. Изучение предметной области, обзор существующих подходов к интеграции информационных ресурсов, анализ доступных внешних систем и сервисов (API), с которыми планируется взаимодействие. Выбор методов и технологий реализации поставленной задачи, обоснование архитектурных решений и протоколов обмена данными.

Формализация задачи. Определение ключевых сценариев взаимодействия, описание структуры входящих и исходящих данных, спецификация форматов запросов и ответов (JSON, XML), проектирование логики обработки данных и схемы взаимодействия компонентов разрабатываемого решения.

Проектирование архитектуры интеграционного решения. Разработка структурной схемы взаимодействия компонентов, определение способов подключения к внешним системам (REST API, SOAP, WebSocket), проектирование базы данных при необходимости, описание алгоритмов обработки и преобразования данных.

Написание программного кода, реализующего спроектированную архитектуру. Разработка серверной части (бэкенд) с реализацией точек подключения к внешним API, разработка клиентской части (фронтенд) или интерфейсов для взаимодействия с пользователем, реализация логики обмена данными и их обработки.

Отладка разработанного решения и исправление ошибок. Тестирование корректности взаимодействия с внешними системами, проверка обработки различных сценариев (успешные запросы, ошибки, таймауты), тестирование безопасности и надежности интеграции, нагрузочное тестирование при необходимости.

Пояснительную записку рекомендуется составлять параллельно с разработкой и тестированием программы, то есть последовательно по разделам, фиксируя принятые решения, описание архитектуры, результаты

тестирования и выявленные особенности. На завершающем этапе выполняется форматирование текста согласно установленным требованиям и стандартам оформления проектной документации.

Составление плана курсового проекта

Выполнение курсового проекта осуществляется в соответствии с утвержденным планом, который должен отражать основное содержание работы и обеспечивать логическую последовательность изложения материала.

Структурно работа над курсовым проектом включает следующие обязательные смысловые этапы:

- Подготовительный этап: выбор и согласование с руководителем темы; поиск и отбор источников информации; формирование библиографического списка; разработка содержания (оглавления) проекта; постановка целей и задач исследования.
- Аналитический этап: углубленное изучение и анализ отобранной литературы и иных источников информации (с предоставлением проработанных материалов руководителю в печатном или электронном виде).
- Практический этап: разработка плана исследования (либо плана практической части), а также подбор необходимых материалов для его реализации.

Настоящие методические рекомендации призваны сформировать у обучающихся навыки грамотного построения и оформления курсового проекта. В них последовательно изложены рекомендации по каждому из перечисленных этапов, что позволит минимизировать количество типичных ошибок, сконцентрировать внимание на ключевых аспектах работы и, как следствие, обеспечить успешную защиту.

Данные методические указания носят регламентирующий характер и являются обязательными к исполнению при выполнении курсового проекта.

Оформление курсового проекта

Курсовой проект должен быть выполнен на компьютере и напечатан шрифтом 14-го размера через полуторный интервал. Текст должен быть чистым и грамотным, с соблюдением абзацных отступов и заданных размеров полей: верхнее и нижнее поле составляют 2 см, левое поле - 3 см, правое поле - 1,5 см.

Сокращение слов в работе, за исключением общепринятых, не разрешается. Все страницы текста подлежат нумерации. Нумерация страниц начинается с содержания (это страница 2; на титульном листе номер страницы не проставляется), а сам номер страницы ставится внизу страницы посередине.

Рекомендуемый объем работы находится в пределах 30–45 страниц. Значительное превышение установленного объема рассматривается как недостаток работы, поскольку указывает на неумение автора отбирать и перерабатывать нужный материал, а также компактно и концентрированно излагать содержание.

Материал следует излагать четко и ясно, применяя техническую терминологию, избегая при этом повторений и приведения общеизвестных положений, которые содержатся в учебниках и учебных пособиях. Пояснять необходимо только малоизвестные или спорные понятия, сопровождая их ссылкой на авторов.

Запрещается включать в проект текст, отсканированный из учебников, технической документации, статей с профильных ресурсов и тому подобных изданий.

Оформление работы должно отвечать следующим требованиям.
Курсовой проект в обязательном порядке включает следующие структурные элементы:

1. Титульный лист (образец его оформления представлен в приложении 1).
2. Содержание (образец оформления содержания курсового проекта приведен в приложении 3).
3. Основная часть курсового проекта.
4. Список используемых источников.
5. Приложения (иллюстрированный материал) — это могут быть схемы, диаграммы, таблицы и скриншоты интерфейсов.

Содержание (оглавление) работы включает перечень основных разделов проекта, а именно: введение, главы и параграфы, заключение,

список литературы и приложения. В содержании указываются номера и названия глав и параграфов. Оглавление должно строго соответствовать заголовкам, приведенным в тексте.

Главы основной части работы должны иметь порядковую нумерацию, например: 1., 2., 3. и так далее. Параграфы должны нумероваться порядковыми номерами внутри каждой главы, например: 1.1., 1.2., 1.3. Если параграф состоит только из одного пункта, то выделять и нумеровать его не следует.

Титульный лист оформляется на стандартном бланке и содержит следующие сведения: название темы; фамилию, имя и отчество студента; фамилию, имя, отчество, ученую степень и ученое звание (либо должность) руководителя курсового проекта.

Содержание должно включать все главы и разделы рукописи с обязательным указанием номеров страниц, на которых они начинаются.

Введение. Этот раздел представляет собой общую характеристику курсового проекта. В нем обосновывается актуальность выбранной темы разработки, оценивается степень ее проработанности, формулируются цель и задачи работы, а также указываются методологические и технологические основы проектирования. Помимо этого, во введении необходимо отразить структуру курсового проекта и назвать конкретный веб-ресурс или приложение, на примере которого проводилась практическая разработка. Рекомендуемый объем введения составляет от 3 до 5 страниц.

Основная часть проекта состоит из двух логически взаимосвязанных и соподчиненных глав (разделов), каждая из которых, в свою очередь, делится на несколько частей (подразделов).

Глава первая. Может включать в себя два основных раздела - теоретический и аналитический. Наличие аналитического раздела в составе первой главы является обязательным требованием.

Теоретический раздел. В этой части работы проводится анализ современного состояния теории и методологии в области веб-разработки по изучаемой проблеме, дается обзор литературных источников и интернет-ресурсов, а также приводится обоснование точки зрения автора на исследуемую проблему.

В теоретическом разделе могут быть рассмотрены следующие вопросы:

– понятие и сущность изучаемого явления или процесса в контексте веб-технологий;

– краткий исторический обзор развития вопроса и эволюции подходов к веб-разработке;

– тенденции развития рассматриваемых технологий, фреймворков и инструментальных средств;

– основные принципы и закономерности, которые применяются при построении веб-приложений, система (группа) показателей качества и производительности, связанных с проблемой, порядок выбора технологического стека, методы проектирования и разработки, используемые в настоящее время, с указанием их достоинств и недостатков.

Целесообразно также провести сравнительный анализ состояния предмета исследования в отечественной и зарубежной практике веб-разработки.

Аналитический раздел включает в себя совокупность расчетно-аналитических и проектно-исследовательских действий, необходимых для обеспечения решения поставленных в работе задач.

Назначением данного раздела является подробное раскрытие практического состояния изучаемой проблемы (темы разрабатываемого веб-ресурса или приложения). В нем анализируется динамика развития аналогичных веб-решений, исследуется пользовательский опыт и функциональные возможности существующих сайтов или приложений (то есть проводится анализ предметной области и конкурентной среды), а также классифицируются факторы, оказывающие влияние на показатели эффективности и удобства использования веб-ресурса. В разделе раскрываются конкретные методы и подходы к решению той или иной проектной задачи.

В этом разделе активно применяются методы структурного и объектно-ориентированного анализа, приемы обработки требований, составляются аналитические таблицы сравнения технологий, строятся графики и схемы алгоритмов, разрабатываются диаграммы вариантов использования и последовательностей и т.д. (на материалах технических заданий, документации к фреймворкам, аналитических обзоров и отчетов).

Величина первой главы должна составлять примерно 20–30% от общего объема рукописи.

Глава вторая (проектная). В этой главе определяются современные требования к разрабатываемому веб-ресурсу, формулируются алгоритмы и критерии решения поставленной проблемы, разрабатываются конкретные предложения по архитектуре приложения и перспективам развития объекта. Выполняются практические работы по проектированию и написанию программного кода согласно выбранной методологии, дается оценка эффективности предлагаемых решений с точки зрения производительности, безопасности и удобства использования.

Определяются новизна и полнота решения поставленных задач. Обозначаются границы применения полученных результатов, а также намечаются пути дальнейшего совершенствования и продолжения разработки. Величина проектной части должна составлять примерно 60–70% от общего объема рукописи.

Заключение. В этой части синтезируется суть выполненного проекта, подводятся итоги решения поставленных в работе задач, и обобщаются результаты, полученные во всех главах. Здесь же рассматриваются направления и пути дальнейшего развития темы в дипломном проекте. Отмечается практическая ценность разработанного веб-ресурса, область его настоящего (или возможного) применения.

Таким образом, заключение должно содержать все существенное и новое, что составляет итог разработки и выносится на защиту. Заключение может занимать 2–3 страницы.

Список использованных источников технической и научной информации является составной частью курсового проекта и показывает степень изученности проблемы. Минимальный список должен содержать не менее 30 источников.

В «Приложения» выносится вспомогательный материал, который облегчит восприятие основной части проекта. Наличие приложения не является обязательным. В приложения могут быть включены: техническое задание, макеты интерфейсов, листинги программного кода, схемы баз данных и т.п.

Курсовой проект включает в себя совокупность результатов научно-практической и проектной деятельности, а также положения, выдвигаемые автором на защиту, которые должны обладать внутренним единством.

Курсовой проект должен соответствовать следующим критериям:

– проект оформляется в виде, позволяющем судить о полноте и обоснованности содержащихся в нем результатов, выводов и предложений;

– проект должен быть законченным исследованием с описанием путей дальнейшего поиска в исследуемом направлении, показывать способность автора видеть перспективу развития разработанного веб-приложения;

– в курсовом проекте автор раскрывает свой потенциал, способность к проведению самостоятельных разработок и исследований на основе теоретических и практических знаний, которые он приобрел за период обучения на кафедре и за время прохождения профессиональной практики в организациях, занимающихся веб-разработкой.

Каждый раздел курсового проекта (введение, каждая глава, заключение) должен начинаться с новой страницы, параграфы (подразделы) располагаются друг за другом вплотную. Заголовки структурных элементов и разделов основной части следует располагать в середине строки без точки в конце и печатать с заглавной буквы строчными буквами, не подчеркивая. Если заголовки содержат несколько предложений, их разделяют точками. Переносы слов в заголовках не допускаются. Расстояние между заголовками структурных элементов и разделов основной части и текстом должно быть не менее 3–4 интервалов.

Сокращение слов в тексте и в подписи под иллюстрациями не допускается. Исключения составляют сокращения, установленные государственным стандартом (ГОСТ Р 7.0.12-2011), а также общеизвестные сокращения, как например, РФ, HTML, CSS, API и др. Не рекомендуется вводить собственные сокращения обозначений и терминов.

Ссылка на первоисточник. Цитаты выделяются кавычками и снабжаются ссылками на источники. При цитировании допустимо использовать современные орфографию и пунктуацию, опускать слова, обозначая пропуск многоточием, если мысль автора не искажается. Ссылка на литературный источник дается по номеру в списке литературы, например, [13. С.15]. Недословное приведение выдержки из какого-либо произведения не выделяется кавычками, но обязательно отмечается в конце фразы [12. С.5]. Нельзя пользоваться порядковыми номерами списка литературы проекта как словами для построения фраз, например: «в 25 дается определение клиент-серверной архитектуры...». Правильное

построение предложения будет: «В учебнике [25] дается определение клиент-серверной архитектуры...».

Иллюстрации (рисунки) и таблицы. При оформлении проекта в него обязательно должны быть включены таблицы, рисунки и даны необходимые формулы.

Иллюстрации (чертежи, рисунки, графики, схемы, диаграммы, скриншоты интерфейсов) и таблицы следует располагать в работе непосредственно после текстов, в которых они упоминаются впервые, или на следующей странице. На все иллюстрации и таблицы должны быть ссылки в работе.

Иллюстрации должны иметь название, которое помещают под иллюстрацией. Наименования, приводимые в тексте и в иллюстрациях, должны быть одинаковыми. При необходимости под иллюстрацией помещают поясняющие данные (подрисуночный текст). Иллюстрация обозначается общим словом «Рисунок – Название рисунка».

По ГОСТ 7.32-2001 на все рисунки в тексте должны быть даны ссылки. Рисунки должны располагаться непосредственно после текста, в котором они упоминаются впервые, или на следующей странице. Рисунки нумеруются арабскими цифрами. Подпись к рисунку располагается под ним посередине строки. Слово «рисунок» в тексте пишется полностью. Подпись должна выглядеть так:

1.Рисунок 1 – Название рисунка (по центру)

2.Оформление рисунков: шрифт Times New Roman, кегль 14, форматирование по центру, интервал – 1,5, абзацный отступ – отсутствует.

3.Переносы слов в рисунках должны соответствовать языковым нормам.

Заголовок таблицы выполняется строчными буквами (кроме первой прописной). Заголовки граф таблицы начинаются с прописных букв, а подзаголовки со строчных букв, если они составляют одно предложение с заголовком. Подзаголовки, имеющие самостоятельное значение, пишут с прописной буквы. В конце заголовка и подзаголовков таблиц знаки препинания не ставят.

Разрывать таблицу и переносить ее на другую страницу можно только в том случае, если она не умещается на одной странице. При переносе части таблицы на другой лист заголовок помещают только над первой частью. Слово «Таблица», порядковый номер и заголовок

указывают один раз над первой частью таблицы, над последующими частями пишут «Продолжение таблицы __».

Графу «№ п/п» в таблицу не включают. При необходимости нумерацию показателей параметров или других данных порядковые номера указывают в боковике таблицы перед их наименованием. Для облегчения ссылок в тексте допускается нумерация граф. Если цифровые данные в графах таблицы выражены в различных единицах измерения, то их указывают в заголовке каждой графы. Если все параметры выражены в одной и той же единице измерения, ее сокращенное обозначение помещают над таблицей.

Нумерация таблиц, рисунков (отдельно для таблиц и рисунков) должна быть сквозной для всего курсового проекта. Слово таблица, ее порядковый номер и название пишутся слева направо. Название таблицы следует помещать над таблицей слева без абзацного отступа в одну строку с ее номером через тире, например:

Таблица 1 – Название таблицы

Оформление текста внутри таблицы: шрифт Times New Roman, кегль 12, форматирование по ширине, интервал – 1, абзацный отступ – отсутствует. Переносы слов в таблицах должны соответствовать языковым нормам.

По ГОСТ 7.32-2001 на все таблицы в тексте должны быть ссылки. Слово «таблица» в тексте пишется полностью. Все таблицы нумеруются (нумерация сквозная).

Точка в конце названия не ставится. Заголовки столбцов и строк таблицы следует писать с прописной буквы в единственном числе, а подзаголовки столбцов – со строчной буквы, если они составляют одно предложение с заголовком, или с прописной буквы, если они имеют самостоятельное значение. В конце заголовков и подзаголовков столбцов и строк точки не ставят. Разделять заголовки и подзаголовки боковых столбцов диагональными линиями не допускается. Заголовки столбцов, как правило, записывают параллельно строкам таблицы, но при необходимости допускается их перпендикулярное расположение.

Таблицы каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением впереди обозначения приложения.

Если таблица заимствована или составлена по данным из технической документации, статистического сборника или другого литературного источника, следует сделать ссылку на первоисточник.

Формулы и расчеты должны органически вписываться в текст изложения, не нарушать грамматической структуры текста. В тексте их надо выделять, записывая более крупным шрифтом и отдельной строкой, давая подробное пояснение каждому символу, когда он встречается впервые. Выше и ниже каждой формулы или уравнения должно быть оставлено не менее одной свободной строки.

Пояснение значений символов и числовых коэффициентов следует проводить непосредственно под формулой в той же последовательности, в которой они даны в формуле. Значение каждого символа и числового коэффициента следует давать с новой строки. Первую строку пояснений начинают со слов «где» без двоеточия.

Формулы следует располагать на середине строки, а связывающие их слова «где», «следовательно», «откуда», «находим», «определяем» – в начале строк.

Формулы в работе следует нумеровать, особенно, если в тексте приходится на них ссылаться, порядковой нумерацией по всей работе арабскими цифрами в круглых скобках в крайнем правом положении на строке, например: (21).

Перечисления могут быть приведены внутри пунктов или подпунктов. Перечисления следует нумеровать порядковой нумерацией, арабскими цифрами со скобкой, например, 1), 2), 3) и т.д., и печатать строчными буквами с абзацного отступа. В пределах одного пункта или подпункта не допускается более одной группы перечислений.

При приведении цифрового материала должны использоваться только арабские цифры, за исключением общепринятой нумерации кварталов, полугодий, которые обозначаются римскими цифрами. Количественные числительные в тексте даются без падежных окончаний.

Интервалы величин в виде «от и до» записываются через черточку. Например, 8-12% или страниц 5-7 и т.д.

При величинах, имеющих два предела, единица измерения пишется только при цифровых или буквенных величинах, в тексте их следует писать только словами; «номер», «процент». Математические знаки +, -, =, >, < и другие используются только в формулах. В тексте их следует писать словами: плюс, минус, равно, меньше, больше.

В конце работы приводится список использованной при ее подготовке литературы (за последние пять лет), а также приложения, где в обязательном порядке прикладываются исходные данные по веб-ресурсу,

схемы базы данных, таблицы, диаграммы, макеты страниц, листинги программного кода и т.п., используемые при анализе и разработке веб-приложения.

В список использованных источников включаются источники, на которые в курсовом проекте есть ссылки, а также те, с которыми обучающийся ознакомился при подготовке проекта: постановления, законодательные и нормативные документы в области информационных технологий, учебники и учебные пособия по веб-разработке, источники технической документации, методическая литература, монографии, сборники статей, материалы профильных конференций, публикации в сетевых изданиях и другие источники.

Источники располагаются в алфавитном порядке (по первой букве первого слова). В авторских источниках первым словом считается фамилия автора. Все источники в перечне нумеруются. Для каждого источника указываются: фамилия и инициалы автора (авторов); полное название книги, статьи; название журнала или сборника статей (для статей); название города Москва и Санкт-Петербург – сокращенно, соответственно М. и Спб., остальные полностью; название издательства (если имеется в выходных данных) – для книг; год издания (для статей – номер журнала и год). Общее количество страниц в книге (например, 206 с.) или конкретные страницы (например, С. 15). Сведения об источниках приводятся в соответствии с требованиями ГОСТ 7.1. Список литературы располагается после раздела «Заключение».

Приложения. Помимо основного текста курсовой проект может содержать приложения. В них рекомендуется включать материалы, которые по каким-либо причинам не могут быть включены в основную часть, например, материалы, дополняющие работу; таблицы вспомогательных данных; иллюстрации вспомогательного характера; акты внедрения результатов разработки; документы (части документов), содержащие фактические данные о работе веб-приложения, которые иллюстрируют основное содержание (например, фрагменты кода, скриншоты интерфейсов, схемы архитектуры и т.п.).

Приложение располагается непосредственно за списком литературы. Каждое приложение должно начинаться с новой страницы и иметь содержательный заголовок, напечатанный строчными буквами. В правом верхнем углу над заголовком прописными буквами должно быть напечатано слово «ПРИЛОЖЕНИЕ».

Если приложений в работе более одного, их следует нумеровать арабскими цифрами порядковой нумерацией. Имеющиеся в тексте приложения иллюстрации, таблицы, формулы и уравнения следует нумеровать в пределах каждого приложения с добавлением буквы «П», например: «Таблица 1П».

Нумерация страниц. Все страницы работы, включая список использованной литературы и приложения, нумеруются арабскими цифрами по порядку. Номер страницы проставляют внизу страницы посередине без точки в конце.

Все листы проекта должны быть скреплены или сброшюрованы в папке. Курсовой проект должен быть отредактирован и подписан автором.

Защита и оценка курсовых проектов

Курсовой проект передается на проверку преподавателю. Если выполненный курсовой проект соответствует предъявляемым требованиям по оформлению и содержанию, то преподаватель допускает его к защите.

При наличии замечаний и недоработок курсовой проект возвращается обучающемуся с замечаниями, которые необходимо учесть при его доработке.

Рекомендации по выполнению:

1 Этап. Подготовительный. Включает следующие виды работ:

- Представление текста завершенного курсового проекта для подготовки отзыва научного руководителя (Приложение 2), а также для проверки в системе «Антиплагиат.ВУЗ». Требуемый уровень авторского вклада - 75% и более.
- Написание текста научного доклада об основных результатах подготовленного курсового проекта; консультации у научного руководителя при подготовке презентации для защиты научного доклада.

Формы контроля:

- Курсовой проект, отзыв научного руководителя и заключение об оригинальности текста курсового проекта, сформированное системой «Антиплагиат.ВУЗ», сдаются на кафедру не позднее чем за 7 дней до дня защиты.
- Научный доклад и презентация сдаются на кафедру до начала защиты.

2 Этап. Защита научного доклада. Включает следующие виды работ:

- Студент делает доклад в течение 5–10 минут, сопровождаемый презентацией;
- Отвечает на вопросы руководителя курсового проекта.
- Студент, не представивший курсовой проект или не защитивший его в установленный срок, не допускается к экзамену по дисциплинам «Веб-разработка», «Проектирование пользовательских интерфейсов».

Критерии оценки курсового проекта:

Оценка	Критерии
Неудовлетворительно	Неверно выбрана тема курсового проекта, содержание курсового проекта не соответствует теме,

Оценка	Критерии
	частично или полностью отсутствуют требуемые разделы курсового проекта, оформление не соответствует установленным требованиям.
Удовлетворительно	Тема курсового проекта выбрана верно, содержание курсового проекта соответствует теме, требуемые разделы в курсовом проекте присутствуют, однако раскрыты не полностью, оформление соответствует установленным требованиям.
Хорошо	Тема курсового проекта выбрана верно, содержание курсового проекта соответствует теме, требуемые разделы в курсовом проекте присутствуют, раскрыты достаточно полно, оформление соответствует установленным требованиям, защита курсового проекта выполнена на недостаточно высоком уровне.
Отлично	Тема курсового проекта выбрана верно, содержание курсового проекта соответствует теме, требуемые разделы в курсовом проекте присутствуют, раскрыты достаточно полно, оформление соответствует установленным требованиям, защита курсового проекта выполнена на высоком уровне.

Требования к электронной презентации для защиты

Защита курсового проекта сопровождается публичной презентацией, которая наглядно демонстрирует основные этапы разработки, архитектурные решения и результаты интеграции информационных ресурсов. Презентация является обязательным элементом защиты и оценивается комиссией наравне с пояснительной запиской и программным кодом.

Рекомендуемое количество слайдов — от 10 до 15. Презентация должна дублировать ключевые моменты устного доклада, но не заменять его полностью. Каждый слайд должен нести смысловую нагрузку и быть визуально читаемым с последнего ряда аудитории.

Слайд 1. Титульный лист. Содержит полное название учебного заведения, факультета и кафедры, тему курсового проекта, фамилию и инициалы студента, фамилию, имя, отчество руководителя, город и год выполнения.

Слайд 2. Актуальность и цель работы. Кратко обосновывается, почему выбранная тема важна для современной веб-разработки. Формулируется главная цель курсового проекта. Рекомендуется использовать не более 5-7 строк текста.

Слайд 3. Задачи, решаемые в проекте. Перечисляются конкретные задачи, которые были поставлены для достижения цели. Каждая задача начинается с глагола: «проанализировать», «спроектировать», «реализовать», «протестировать», «оценить».

Слайд 4. Обзор аналогов. Представляется таблица сравнения 3-4 существующих веб-решений или сервисов, выполняющих схожие функции. Столбцы таблицы: название сервиса, ключевые функции, преимущества, недостатки. В последней строке таблицы указывается, какие достоинства аналогов были учтены в разработанном проекте.

Слайд 5. Архитектура приложения. Размещается схема взаимодействия компонентов разрабатываемой системы. На схеме должны быть обозначены: клиентская часть (браузер), серверная часть (бэкенд), внешние API или сервисы, база данных (при наличии). Стрелки показывают направление потоков данных и протоколы взаимодействия (REST, WebSocket, GraphQL).

Слайд 6. Выбор технологического стека. Обосновывается выбор конкретных технологий, фреймворков и инструментов. Рекомендуется оформить в виде списка с краткими пояснениями. Пример: «Фронтенд —

React (компонентный подход, большая экосистема)», «Бэкенд — Node.js + Express (единый язык с фронтендом)», «База данных — PostgreSQL (надёжность, поддержка JSONB)».

Слайд 7. Демонстрация пользовательского интерфейса. Размещаются 2-3 скриншота основных страниц или модальных окон разработанного веб-приложения. Каждый скриншот должен быть подписан: «Рисунок 1 — Главная страница», «Рисунок 2 — Форма отправки запроса к API». Допускается анимация переключения между скриншотами (при использовании интерактивной презентации).

Слайд 8. Демонстрация интеграционного кода. Приводится фрагмент программного кода (не более 20-25 строк), реализующего ключевое взаимодействие с внешним API. Код должен быть оформлен моноширинным шрифтом (Consolas, Courier New, Menlo) с подсветкой синтаксиса. Желательно показать как отправку запроса, так и обработку ответа или ошибки.

Слайд 9. Результаты тестирования. Представляется таблица с тестовыми сценариями. Столбцы таблицы: «Сценарий», «Ожидаемый результат», «Фактический результат», «Статус (успех/ошибка)». Примеры сценариев: успешный GET-запрос к API, отправка POST-запроса с некорректными данными, отсутствие соединения с интернетом, превышение времени ожидания (таймаут).

Слайд 10. Заключение и перспективы развития. Формулируются основные итоги выполненной работы: что именно разработано, какие задачи решены, какие навыки приобретены. Указываются направления для дальнейшего развития проекта: добавление нового функционала, подключение дополнительных API, оптимизация производительности, внедрение кэширования.

Слайд 11. Спасибо за внимание. Содержит благодарность аудитории и предложение задать вопросы. Желательно указать контактные данные автора (электронную почту или ссылку на GitHub-репозиторий с проектом).

Презентация должна быть выполнена в формате PowerPoint (PPTX), PDF. При использовании PowerPoint не допускается применение экзотических шрифтов, которые могут отсутствовать на компьютере комиссии. Рекомендуются стандартные шрифты: Arial, Calibri, Times New Roman.

Размер шрифта для заголовков слайдов — не менее 28 пунктов, для основного текста — не менее 20 пунктов, для подписей и примечаний — не менее 14 пунктов. Цвет текста должен контрастировать с фоном слайда. Не рекомендуется использовать более трёх цветов на одном слайде.

Анимация и эффекты перехода допускаются, но не должны отвлекать внимание от содержания. Оптимальная длительность анимации — 0,5–1 секунда. Звуковое сопровождение презентации не используется, за исключением случаев, когда это прямо предусмотрено демонстрацией функционала веб-приложения.

Код на слайдах должен быть оформлен с использованием моноширинного шрифта (Consolas, Courier New) размером не менее 16 пунктов. Для подсветки синтаксиса рекомендуется использовать сервисы вроде carbon.now.sh или встроенные возможности редакторов кода с последующим экспортом в изображение.

Время выступления студента не должно превышать 7 минут. Презентация автоматически переключается докладчиком. Недопустимо чтение текста со слайдов дословно — студент должен комментировать и пояснять представленную информацию.

Рекомендуется следующее распределение времени:

Приветствие, тема, актуальность, цель — 1 минута.

Задачи и обзор аналогов — 1 минута.

Архитектура и технологии — 1,5 минуты.

Демонстрация интерфейса и кода — 2 минуты.

Результаты тестирования и заключение — 1,5 минуты.

После завершения доклада студент отвечает на вопросы руководителя и членов комиссии. Вопросы могут касаться как теоретической части (выбор технологий, обоснование решений), так и практической реализации (особенности интеграции, обработка ошибок, безопасность).

Информационное обеспечение

Концептуальная модель демонстрационного сайта представлена совокупностью взаимосвязанных веб-страниц. Страницы, в свою очередь, состоят из текстовых и графических объектов, а также определяют способы их размещения в окне браузера. Основной средой для написания программного кода служил редактор исходного кода Visual Studio Code, а сама разметка выполнялась на языке HTML с применением скриптов на языке JavaScript. Для создания и обработки графических элементов применялись графические редакторы Figma (для прототипирования и отрисовки интерфейсов) и Adobe Photoshop (для оптимизации растровых изображений).

На всех страницах сайта неотъемлемыми атрибутами являются заголовок и фоновое оформление. Также обязательным элементом страницы служит текстовый контент, который может быть статичным или анимированным, причем его объем напрямую влияет на размер загружаемого файла и скорость отображения страницы. Любой текстовый блок содержит параметры форматирования, такие как размер шрифта, начертание, цвет и межстрочный интервал.

Для навигации по сайту и для обеспечения обратной связи с разработчиком существует система гиперссылок, которые характеризуются объектом ссылки (на кого или на что производится ссылка - на внутренний документ сайта, на внешний ресурс или на форму обратной связи) и типом ссылки, которые могут быть текстовыми или графическими. Так, на главной странице сайта размещена ссылка, при нажатии на которую запускается почтовый клиент, установленный на устройстве пользователя, и посетителю предлагается написать письмо автору проекта.

Без сомнения, любой современный сайт содержит графический контент. Графика, в свою очередь, характеризуется такими параметрами, как разрешение, физический размер графического файла, его формат (расширение) и цветовая палитра. Чем выше разрешение, чем богаче палитра используемых цветов, тем более выигрышно будет смотреться объект на экране и тем дольше, к сожалению, он будет загружаться через сеть. То же самое касается и физических размеров изображения: чем крупнее картинка по площади, тем меньше пользователей дождутся ее полной загрузки. Что касается формата графики, то в современной веб-разработке существуют два основных лидера: формат JPEG для

фотографических изображений с плавными переходами цветов и формат PNG для рисунков, иконок, логотипов и элементов интерфейса, требующих прозрачности. Для анимированных элементов также может использоваться формат WebP или специализированные CSS-анимации.

Технологическое обеспечение

При создании демонстрационного примера использовался язык гипертекстовой разметки HTML и язык описания клиентских сценариев JavaScript.

Веб-страницы могут существовать в любом формате, но в качестве общепринятого стандарта принят язык HTML (HyperText Markup Language) - язык разметки гипертекстов, который предназначен для создания форматированного текста, насыщенного изображениями, звуковым сопровождением, анимацией, видеоклипами и гипертекстовыми ссылками на другие документы, разбросанные как по всему веб-пространству, так и находящиеся на этом же сервере или являющиеся составной частью данного веб-проекта.

Работать с веб-технологиями можно и без глубокого знания языка HTML, поскольку разметка может генерироваться различными фреймворками и конструкторами сайтов. Однако написание кода непосредственно на HTML не представляет большой сложности. Более того, понимание основ языка дает разработчику полный контроль над структурой документа и зачастую позволяет создавать более чистый и оптимизированный код, чем тот, который автоматически генерируется визуальными редакторами, которые иногда бывают ограничены в своих возможностях или добавляют избыточную разметку, затрудняющую поддержку проекта.

Язык HTML постоянно развивается (актуальной на сегодня является версия HTML5), но базовые конструкции языка остаются неизменными и поддерживаются всеми браузерами. Изучая HTML и углубляя свои познания, создавая документ на начальном этапе изучения и расширяя его по мере необходимости, мы получаем возможность создавать веб-страницы, которые будут корректно отображаться большинством современных браузеров как сейчас, так и в обозримом будущем. Это не исключает возможности использования расширенных возможностей, которые предоставляют конкретные браузеры (YANDEX).

HTML ратифицирован консорциумом World Wide Web (W3C), поддерживается всеми распространенными браузерами и является фундаментом практически всего программного обеспечения, которое имеет отношение к вебу.

Поскольку HTML-документы записываются в текстовом формате, для создания сайта может быть использован любой редактор исходного кода (Visual Studio Code, Sublime Text, Atom и др.) или даже простейший текстовый редактор.

Обычно HTML-документ — это файл с расширением .html или .htm, в котором текст размечен HTML-тегами (от англ. tag - специальные встроенные указания). Средствами HTML задаются синтаксис и размещение тегов, в соответствии с которыми браузер отображает содержимое веб-документа. Текст самих тегов браузером не отображается.

Все теги начинаются с символа левой угловой скобки <и заканчиваются символом правой угловой скобки >. Обычно используется пара тегов - стартовый (открывающий) и завершающий (закрывающий) тег, между которыми помещается размечаемая информация:

Информация

Здесь стартовым тегом является тег , а завершающим - . Завершающий тег отличается от стартового лишь тем, что у него перед текстом в скобках стоит символ / (слэш). Существуют также одиночные теги, не требующие закрытия (например, <img> или
).

Браузер, читающий HTML-документ, отображает его в окне, интерпретируя структуру HTML-тегов. В каждом HTML-документе должны присутствовать три главные части:

- объявление типа документа (DOCTYPE);
- заголовочная часть (head);
- тело документа (body).

Объявление HTML.

<html> и </html>. Пара этих тегов сообщает программе просмотра (браузеру), что между ними заключен документ в формате HTML, причем первым тегом в документе должен быть тег <html> (в самом начале документа после указания <!DOCTYPE>), а последним - </html> (в самом конце документа):

```
<html>
...
</html>
```

Заголовочная часть.

<head> и </head>. Между этими тегами располагается служебная информация о документе (название, ключевые слова для поисковых систем, описание, подключение стилей и скриптов и т. д.). Наиболее важным элементом является название документа, которое отображается на вкладке браузера и в результатах поисковой выдачи. Специальные программы поисковых роботов используют название документа для индексации и построения своих баз данных. Для того чтобы задать название HTML-документу, текст помещается между тегами <title> и </title>:

```
<html>
<head>
<title>Моя первая страница</title>
</head>
</html>
```

Тело документа.

Третьей главной частью документа является его тело. Оно следует сразу за заголовочной частью и находится между тегами <body> и </body>. Первый из них должен стоять сразу после закрывающего тега </head>, а второй - перед закрывающим тегом </html>. Тело HTML-документа - это область, где автор размещает контент, отформатированный средствами HTML и отображаемый непосредственно в окне браузера:

```
<html>
<head>
<title>Моя первая страница</title>
</head>
<body>
...
</body>
</html>
```

В разделе body все символы табуляции и переноса строк браузером по умолчанию игнорируются и не влияют на отображение страницы. Поэтому перевод строки в исходном тексте HTML-документа не приведет к началу новой строки в отображаемом тексте при отсутствии специальных тегов. Это правило очень важно помнить, иначе текст сольется в нечитаемый поток.

Для начала новой строки используется одиночный тег
 (сокращение от англ. break - прервать). Этот тег указывает браузеру отображать дальнейший текст с начала следующей строки. Повторное его использование позволяет вставить одну или несколько пустых строк.

Сплошной текст без промежутков плохо воспринимается. Текст, разбитый на абзацы, читается гораздо легче. Для начала нового абзаца используется парный тег <p> (от англ. paragraph - абзац). Этот тег, кроме начала новой строки, добавляет вертикальный отступ перед абзацем и после него.

Внутри скобок тега кроме его названия могут размещаться также атрибуты. Они отделяются от названия и между собой пробелами, а записываются в виде имя_атрибута="значение". Тег <p> может содержать атрибут style или класс для управления выравниванием текста, отступами и другими параметрами через CSS. По умолчанию текст выравнивается по левому краю. Выравнивание по центру, правому краю или по ширине задается с помощью свойства text-align.

Кроме использования этих тегов, для отображения предварительно отформатированного текста (с сохранением пробелов и переносов) используется парный тег <pre>. Весь текст, помещенный между тегами <pre> и </pre>, будет выводиться моноширинным шрифтом без изменений.

В HTML-документе могут содержаться горизонтальные разделительные линии. Тег <hr> выводит горизонтальную линию вдоль всей ширины родительского контейнера. Горизонтальная линия всегда приводит к разрыву строки. Внешний вид линии (цвет, толщина, стиль) настраивается с помощью CSS.

Линии, наряду с абзацами, позволяют визуально разделять логические фрагменты текста.

Таблицы. Практически каждая страница сайта содержит хотя бы одну таблицу. Изначально теги HTML для таблиц были разработаны для представления строк и столбцов с табличными данными. Однако долгое время таблицы использовались разработчиками как ценное средство управления разметкой веб-страниц для создания колонок, задания отступов и сложного позиционирования элементов. В современной веб-разработке для целей построения сетки (раскладки) страницы рекомендуется использовать технологии CSS Grid Layout и Flexbox, а

таблицы применять строго по их прямому назначению - для отображения табличных данных.

В демонстрационном примере, основанном на классическом подходе, страницы сайта (кроме главной) могли содержать фреймы. Фреймы позволяют разделить окно браузера на несколько независимых областей, в каждую из которых можно загрузить отдельный HTML-документ. Главное преимущество фреймов заключалось в том, что они обеспечивали неподвижность одних частей страницы (например, навигационного меню), в то время как другие части могли прокручиваться.

Важное примечание для современной разработки: Технология фреймов (теги <frameset> и <frame>) считается устаревшей и не рекомендуется к использованию в современных проектах. Она исключена из стандарта HTML5. Для создания областей с независимой прокруткой или для встраивания внешних виджетов в настоящее время используются контейнеры с CSS-свойством overflow: auto или тег <iframe>.

JavaScript представляет собой язык написания сценариев на стороне клиента (в браузере), который добавляет на веб-страницы элементы интерактивности и динамического поведения. С помощью JavaScript разработчик может реагировать на действия пользователя (клики, ввод данных, движение мыши), изменять содержимое страницы без перезагрузки, отправлять и получать данные с сервера (технология AJAX/Fetch), управлять мультимедиа и анимациями, а также создавать сложные одностраничные приложения (SPA).

Сценарии JavaScript обычно подключаются к документу HTML через тег <script>. Они могут размещаться либо в заголовочной части документа <head>, либо в теле документа <body>. В один документ можно поместить несколько сценариев, а также подключить внешние файлы с расширением .js. Пример синтаксиса вставки внутреннего скрипта:

<script>
// Здесь находится код на JavaScript
</script>

Элементы, без которых не обходится ни одна веб-страница (фон, текст и гиперссылки), ставят перед разработчиком задачу гармонизации нескольких цветов, занимающих в композиции различные площади и выполняющих разные функции. Есть немало страниц с минимальным количеством графики и достаточно стандартной композицией, которые

привлекают и запоминаются исключительно удачно подобранной цветовой гаммой.

Первое требование к паре цветов для фона и текста - достаточный контраст между ними, необходимый для комфортного чтения. Контраст этот должен прежде всего выражаться в разнице яркостей цветов, так как разница только в тоне или насыщенности может быть недостаточна для четкого различения текста и фона, особенно для пользователей с нарушениями зрения.

В современном веб-дизайне широко применяются как темные, так и светлые темы оформления. Светлый текст на темном фоне, хотя и может уступать по комфортности длительного чтения черному тексту на светлом фоне, в небольших объемах меньше утомляет глаз, так как ограничивает общее количество света, излучаемого экраном (особенно актуально для устройств с OLED-экранами и при работе в условиях низкой освещенности). Выбор цветовой схемы должен учитывать контекст использования сайта и предпочтения целевой аудитории.

Для того чтобы разместить разработанный сайт в сети Интернет, необходимо воспользоваться услугами хостинг-провайдера или облачной платформы. Существует множество сервисов, предоставляющих как платный, так и бесплатный хостинг для размещения статических и динамических веб-сайтов (например, GitHub Pages, Netlify, Vercel, Beget, Timeweb и другие).

Общий алгоритм публикации сайта выглядит следующим образом:

Подготовить все файлы проекта (HTML, CSS, JavaScript, изображения и прочие ресурсы) в структурированном виде на локальном компьютере.

Зарегистрироваться на выбранной хостинговой площадке.

Получить данные для доступа к серверу (обычно это адрес FTP-сервера или данные для входа в панель управления, либо инструкции по подключению Git-репозитория)/

Загрузить файлы проекта на удаленный сервер в корневую директорию (часто это папка public_html, www или специальная ветка репозитория, например gh-pages).

После завершения загрузки сайт станет доступен по выделенному доменному имени (например, https://username.github.io/repository-name/ или https://your-site.netlify.app).

Важно убедиться, что главная страница сайта имеет имя index.html, так как веб-сервер по умолчанию ищет файл с таким названием при обращении к корню сайта.

Современные фреймворки и библиотеки

В дополнение к классическим технологиям (HTML, CSS, JavaScript), описанным выше, современная индустрия веб-разработки активно использует фреймворки и библиотеки, которые значительно ускоряют процесс создания сложных интеграционных решений. В рамках курсового проекта студент может использовать как «чистые» технологии, так и любой из перечисленных ниже инструментов по согласованию с руководителем.

Клиентские фреймворки (фронтенд)

React — библиотека для построения пользовательских интерфейсов, разработанная компанией Meta. Основные преимущества: компонентный подход (повторное использование кода), виртуальный DOM (высокая производительность), огромная экосистема сторонних библиотек. Для интеграции с API в React используется хук useEffect и библиотека axios или встроенный fetch. Пример типовой структуры компонента, обращающегося к API, приведён в приложении Б.

Vue.js — прогрессивный фреймворк, отличающийся низким порогом входа и понятной документацией. Подходит для быстрой разработки небольших и средних проектов. Интеграция с API выполняется с помощью метода created() или хука onMounted (Composition API).

Angular — полноценный фреймворк, включающий в себя всё необходимое для разработки крупных корпоративных приложений. Использует TypeScript и паттерн проектирования MVC. Для работы с API применяется встроенный модуль HttpClient.

Серверные платформы (бэкенд)

Node.js + Express — платформа, позволяющая выполнять JavaScript на стороне сервера. Express является минималистичным веб-фреймворком для маршрутизации запросов и создания REST API. Преимущество для студента — единый язык (JavaScript) на фронтенде и бэкенде, что упрощает разработку интеграционных решений.

Django (Python) — высокоуровневый фреймворк, следующий принципу «всё включено». Содержит встроенную ORM для работы с базами данных, систему аутентификации и административную панель.

Рекомендуется для проектов, где требуется сложная работа с данными и моделями.

Laravel (PHP) — фреймворк, широко распространённый в веб-разработке благодаря удобному синтаксису и богатому набору инструментов (Eloquent ORM, Blade-шаблоны, очереди задач). Подходит для проектов, которые планируется размещать на стандартных PHP-хостингах.

Выбор технологического стека для курсового проекта

При выборе технологий студент должен руководствоваться следующими критериями:

- Соответствие теме и задачам курсового проекта.
- Наличие документации и примеров реализации на русском языке.
- Возможность размещения готового проекта на бесплатном или условно-бесплатном хостинге (GitHub Pages, Netlify, Heroku, Render).
- Собственная степень владения технологией.

Не рекомендуется выбирать фреймворк «для галочки», если студент не может обосновать его применение и не знаком с его основами. В этом случае предпочтительнее использовать классические HTML, CSS и чистый JavaScript без фреймворков.

Программное обеспечение

В качестве операционной системы для разработки современного веб-приложения целесообразно выбрать одну из актуальных версий Windows (10 или 11), macOS или дистрибутив Linux (Ubuntu) по целому ряду причин.

Во-первых, эти операционные системы являются наиболее распространенными среди веб-разработчиков и конечных пользователей, для которых создается продукт. Разрабатывать сайт, ориентируясь на узкоспециализированную или устаревшую ОС, было бы нерационально.

Во-вторых, современные ОС обеспечивают высокую надежность работы благодаря встроенным механизмам безопасности, файловым системам с журналированием и поддержке актуальных версий браузеров и сред разработки.

В-третьих, экосистема каждой из перечисленных ОС включает в себя множество инструментов и утилит, повышающих производительность труда разработчика (менеджеры пакетов, терминалы, отладчики).

Операционные системы семейства Windows и macOS предоставляют приложениям возможности создания пользовательского интерфейса на высоком уровне эргономичности. В то же время Linux-системы часто предпочтительнее для серверной части веб-разработки из-за гибкости настройки и меньшего потребления ресурсов.

Поскольку основным инструментом просмотра результата работы является браузер, ключевое требование к системе - поддержка современных версий браузеров и инструментов разработчика.

В качестве базового программного обеспечения при разработке используется один из современных браузеров как для отображения результата, так и для отладки, поскольку встроенные в них «Инструменты разработчика» (DevTools) позволяют инспектировать DOM-дерево, отслеживать сетевые запросы, отлаживать JavaScript-код и анализировать производительность страницы.

При разработке было принято решение ориентироваться на современные веб-стандарты, поддерживаемые актуальными версиями основных браузеров, чтобы обеспечить корректную работу сайта у максимального числа пользователей без необходимости установки дополнительных плагинов.

Для непосредственного написания и редактирования исходного кода (HTML, CSS, JavaScript) используется редактор кода Visual Studio Code. Ручное написание кода дает разработчику полный контроль над проектом, позволяет писать оптимизированную и чистую разметку, избегая избыточности, характерной для визуальных конструкторов. Кроме того, использование редактора кода с плагинами (Prettier, ESLint, Live Server) значительно ускоряет процесс разработки и повышает качество кода.

Все графические элементы сайта (макеты, иконки, изображения) были подготовлены и оптимизированы в графическом редакторе Figma (для прототипирования интерфейса) и Adobe Photoshop (для финальной обработки растровой графики). К преимуществам этих редакторов относят развитую систему управления слоями и богатый набор инструментов для обработки изображений и экспорта ресурсов в форматы, подходящие для веб-разработки (WebP, PNG, JPEG, SVG).

Прикладное программное обеспечение представляет собой веб-сайт образовательного учреждения (вуза). На нем размещается информация о вузе, призванная заинтересовать потенциальных абитуриентов и их родителей, а также представить учебное заведение в информационном

пространстве. На сайте представлена история и концепция развития вуза, описание образовательных программ и студенческой жизни, а также контактная информация. Все это представлено в удобном для восприятия формате с использованием адаптивной верстки, обеспечивающей корректное отображение на экранах различных устройств (компьютеров, планшетов, смартфонов).

Страницы сайта спроектированы таким образом, чтобы не перегружать пользователя избыточной информацией. Сайт снабжен удобной системой навигации по всем разделам. Меню, выполненное с использованием CSS Flexbox, остается доступным в верхней части страницы и фиксируется при прокрутке длинных страниц для удобства навигации.

Код демонстрационного сайта написан на HTML5 и CSS3 с использованием JavaScript (ES6+) в редакторе кода Visual Studio Code. При дальнейшей поддержке и развитии сайта рекомендуется придерживаться методологии компонентного подхода (например, БЭМ) и использовать систему контроля версий Git для отслеживания изменений.

Описание структуры начнем с главной страницы (index.html).

Главная страница (файл index.html). В разделе <head> подключаются файлы стилей и скриптов, задаются мета-теги для корректного отображения на мобильных устройствах (viewport) и кодировки (charset="UTF-8").

Для создания плавной анимации появления логотипа при загрузке страницы используется CSS-анимация или JavaScript-функция, которая изменяет CSS-свойства элемента (например, transform: scale() или opacity) с течением времени.

Фон страницы задается с помощью CSS-свойств background-color и background-image. Основой разметки страницы служат семантические HTML5-теги (<header>, <main>, <nav>, <footer>) и контейнеры, стилизованные с помощью CSS Grid или Flexbox для создания адаптивной сетки.

В шапке сайта размещается логотип и главное навигационное меню. В основной части страницы размещается приветственный текст и ключевые ссылки на внутренние разделы.

Все страницы сайта имеют единую структуру и переиспользуемые компоненты (шапка, подвал), что обеспечивает визуальное единство и упрощает поддержку. Для реализации этого подхода используется сборка

страниц с помощью статического генератора или серверного инклуда, либо простое копирование кода, если проект небольшой.

Адаптивность страниц достигается использованием медиа-запросов CSS (@media), которые изменяют расположение и размеры блоков в зависимости от ширины экрана устройства.

Для просмотра демонстрационного сайта вуза пользователю необходим любой современный компьютер или мобильное устройство с выходом в Интернет и установленным браузером. Первая страница, на которую попадает посетитель, является главной. Она содержит логотип учебного заведения, краткую информацию о миссии вуза и основное навигационное меню.

В шапке сайта, которая видна на всех страницах, расположены ссылки на основные разделы: «История вуза», «Образовательные программы», «Студенческая жизнь», «Контакты». При нажатии на любой пункт меню пользователь переходит на соответствующую тематическую страницу.

На главной странице также может быть размещен блок с последними новостями или анонсами событий, оформленный в виде карточек с заголовками и краткими описаниями. С любой страницы сайта можно быстро вернуться на главную страницу, нажав на логотип в шапке или соответствующую ссылку в меню.

На странице «Контакты» представлена форма обратной связи, позволяющая отправить сообщение администрации сайта или в приемную комиссию без использования внешнего почтового клиента (с помощью серверного обработчика или сервиса сбора форм). Также на странице указан фактический адрес вуза и встроена интерактивная карта проезда на основе Яндекс.Карт.

Все страницы сайта оптимизированы для быстрой загрузки: изображения сжаты, скрипты и стили минифицированы, что обеспечивает комфортную работу с сайтом даже при невысокой скорости интернет-соединения.

Работа с базами данных

Современный веб-сайт невозможно представить без системы хранения и управления данными. Если статический сайт состоит из набора готовых HTML-страниц, то динамический портал формирует содержимое страниц «на лету», извлекая информацию из базы данных по запросу

пользователя. Такой подход позволяет легко обновлять контент, управлять пользователями, вести статистику посещений и многое другое без необходимости редактирования каждого HTML-файла вручную.

Для организации взаимодействия веб-приложения с базой данных используется связка «веб-сервер — серверный язык программирования — система управления базами данных (СУБД)». Наиболее распространенной и проверенной временем комбинацией для учебных и реальных проектов является LAMP/LEMP-стек (Linux, Apache/Nginx, MySQL/MariaDB, PHP) либо аналогичные решения на других платформах.

Проектирование структуры базы данных

Перед началом разработки динамического веб-портала необходимо спроектировать логическую и физическую структуру базы данных. Этот этап включает следующие шаги:

1. Анализ предметной области. На данном этапе определяются сущности (объекты), информация о которых будет храниться в базе данных. Например, для сайта образовательного учреждения такими сущностями могут быть: «Студент», «Преподаватель», «Группа», «Дисциплина», «Расписание», «Новость».

2. Построение концептуальной модели. Определяются взаимосвязи между сущностями (один-к-одному, один-ко-многим, многие-ко-многим). Результатом является схема данных, которая затем преобразуется в набор реляционных таблиц.

3. Нормализация таблиц. Процесс приведения таблиц к нормальным формам (обычно достаточно третьей нормальной формы) позволяет избежать избыточности данных, аномалий при вставке, обновлении и удалении записей.

4. Создание физической структуры. На данном этапе с помощью выбранной СУБД создаются таблицы, определяются типы данных для каждого поля, задаются первичные и внешние ключи, индексы для ускорения поиска.

Пример структуры таблицы для хранения новостей портала:

Поле	Тип данных	Описание
id	INT, AUTO_INCREMENT	Уникальный идентификатор новости
title	VARCHAR(255)	Заголовок новости

Поле	Тип данных	Описание
content	TEXT	Полный текст новости
author_id	INT	Идентификатор автора (внешний ключ)
created_at	DATETIME	Дата и время создания
is_published	BOOLEAN	Флаг публикации

Для работы с базой данных из веб-приложения используется серверный язык программирования (например, PHP, Python, Node.js). Взаимодействие осуществляется путем отправки SQL-запросов к СУБД и обработки полученных результатов.

Основные операции при работе с данными (CRUD):

- Create (Создание) — добавление новых записей в таблицы (SQL-команда INSERT).
- Read (Чтение) — извлечение данных из таблиц для отображения на страницах портала (SQL-команда SELECT).
- Update (Обновление) — изменение существующих записей (SQL-команда UPDATE).
- Delete (Удаление) — удаление записей (SQL-команда DELETE).

Важные аспекты организации работы с БД:

- Безопасность. Все данные, поступающие от пользователя через формы на сайте, должны проходить обязательную фильтрацию и экранирование специальных символов для предотвращения SQL-инъекций — одного из самых распространенных типов атак на веб-приложения. Рекомендуется использовать подготовленные выражения (prepared statements) и параметризованные запросы.
- Оптимизация запросов. Для ускорения работы портала следует избегать избыточных запросов к базе данных, использовать индексы для полей, по которым часто производится поиск и сортировка, применять кэширование результатов часто выполняемых запросов.
- Управление соединениями. Необходимо следить за своевременным закрытием соединений с базой данных после выполнения запросов, чтобы не исчерпать лимит подключений к серверу СУБД. В

современных фреймворках эта задача обычно решается автоматически с помощью пулов соединений.

Публикация сайта

Перед размещением разработанного веб-сайта в сети Интернет необходимо выполнить комплекс подготовительных мероприятий:

1. Тестирование на локальном сервере. Все функции портала должны быть тщательно проверены в среде, максимально приближенной к реальным условиям хостинга. Рекомендуется использовать локальные серверные сборки (OpenServer, MAMP, XAMPP, Laragon) или инструменты контейнеризации (Docker, Docker Compose).

2. Оптимизация ресурсов. Для ускорения загрузки страниц необходимо:

- минифицировать CSS- и JavaScript-файлы (удалить лишние пробелы, комментарии, переносы строк);
- оптимизировать изображения (сжать без существенной потери качества, использовать современные форматы WebP/AVIF);
- объединить несколько файлов стилей и скриптов в один для сокращения количества HTTP-запросов;
- настроить кэширование статических ресурсов.

3. Настройка файла конфигурации. Необходимо адаптировать параметры подключения к базе данных, пути к файлам и другие настройки под реальное окружение хостинг-провайдера. Конфиденциальные данные (пароли к БД, ключи API) не должны храниться в публично доступных частях проекта.

4. Проверка безопасности. Следует убедиться, что:

- файлы с конфиденциальной информацией защищены от прямого доступа из браузера;
- директории с загружаемыми пользователями файлами имеют корректные права доступа;
- отсутствуют отладочные скрипты и временные файлы, созданные в процессе разработки;
- все используемые библиотеки и плагины обновлены до актуальных версий без известных уязвимостей

5. Обеспечение отказоустойчивости. Рекомендуется предусмотреть механизмы обработки ошибок, которые в случае сбоя не раскрывают

посетителю техническую информацию о структуре сайта и сервера, а отображают дружественное пользователю сообщение.

Типы хостинга:

Виртуальный хостинг (shared hosting). Экономичный вариант для небольших и средних проектов. Несколько сайтов располагаются на одном физическом сервере и разделяют его ресурсы. Подходит для большинства учебных курсовых проектов.

Виртуальный выделенный сервер (VPS/VDS). Предоставляет больше ресурсов и возможностей для настройки серверного окружения. Требует навыков системного администрирования.

Облачный хостинг. Ресурсы распределяются динамически между несколькими физическими серверами. Обеспечивает высокую масштабируемость и надежность.

Специализированные платформы для статических сайтов. Для проектов без серверной логики и базы данных идеально подходят GitHub Pages, Netlify, Vercel, предлагающие бесплатный хостинг с автоматическим развертыванием из Git-репозитория.

Процесс размещения файлов:

Получить от хостинг-провайдера данные для доступа: адрес FTP-сервера или SFTP/SSH, имя пользователя, пароль.

С помощью FTP-клиента (FileZilla, WinSCP, Cyberduck) или через файловый менеджер панели управления хостингом загрузить все файлы и директории проекта в корневую папку сайта (обычно это public_html, www, htdocs или аналогичная).

Для динамических сайтов с базой данных необходимо также:

создать базу данных и пользователя через панель управления хостингом;

импортировать структуру и начальные данные из локальной БД (экспорт в формате SQL и импорт через phpMyAdmin или аналогичный инструмент);

указать параметры подключения к базе данных в конфигурационном файле сайта.

Примерные темы курсовых проектов

Ниже представлен расширенный перечень тем курсовых проектов по дисциплине «Интеграция информационных ресурсов с другими системами в сети Интернет». Для каждой темы указаны ключевые внешние системы

или API, с которыми предлагается организовать взаимодействие, а также примерный технологический стек.

Тема 1. Веб-приложение для отображения погоды с интеграцией открытого метеорологического API

Описание: Разработать веб-приложение, позволяющее пользователю ввести название города и получить текущую погоду, а также прогноз на несколько дней вперёд. Данные получаются с помощью публичного API (OpenWeatherMap, WeatherAPI или аналогичный). Реализовать кэширование запросов для снижения нагрузки на API и отображение иконок погоды.

Внешние API: OpenWeatherMap (текущая погода, 5-дневный прогноз), GeoDB Cities (для автодополнения названий городов).

Рекомендуемый стек: HTML, CSS, JavaScript (Fetch API), либо React + Axios.

Тема 2. Сервис для конвертации валют с использованием API Центрального банка или криптовалютных бирж

Описание: Разработать веб-сервис, позволяющий конвертировать суммы из одной валюты в другую по актуальному курсу. Поддержка как фиатных валют (доллар, евро, рубль, юань), так и криптовалют (биткоин, эфириум) — по выбору студента. Реализовать историю конвертаций для текущей сессии.

Внешние API: ExchangeRate-API, CurrencyFreaks, CoinGecko API (для криптовалют), ЦБ РФ (XML-формат).

Рекомендуемый стек: Node.js + Express (бэкенд), любой фронтенд-фреймворк или чистый JavaScript.

Тема 3. Интеграция платежной системы в интернет-магазин (приём платежей через ЮKassa или Stripe)

Описание: Разработать демонстрационный интернет-магазин (корзина товаров, оформление заказа) с возможностью оплаты заказа через одну из платёжных систем. Реализовать создание платежа, обработку вебхуков (webhook) для подтверждения оплаты и отображение статуса заказа пользователю.

Внешние API: ЮKassa (бывшая Яндекс.Касса), Stripe, Tinkoff Acquiring (на выбор). Требуется создание тестового аккаунта и использование тестовых карт.

Рекомендуемый стек: PHP + Laravel или Python + Django (для удобной обработки вебхуков), JavaScript на фронтенде.

Тема 4. Агрегатор новостей с интеграцией RSS-лент и API новостных сервисов

Описание: Разработать веб-агрегатор, собирающий новости из нескольких источников (RSS-ленты сайтов, API NewsAPI.org или аналогичных). Реализовать фильтрацию новостей по категориям, ключевым словам и дате публикации. Добавить возможность сохранения понравившихся новостей в локальное хранилище браузера (localStorage) или в базу данных.

Внешние API: NewsAPI.org, Currents API, RSS-ленты любых новостных сайтов (парсинг XML).

Рекомендуемый стек: Python (библиотеки feedparser, requests), Flask/Django, любой фронтенд.

Тема 5. Чат-приложение с интеграцией сервиса аутентификации через социальные сети (OAuth 2.0)

Описание: Разработать простое веб-приложение для обмена сообщениями в реальном времени (WebSocket или Server-Sent Events). Реализовать вход пользователей через аккаунты ВКонтакте, GitHub с использованием протокола OAuth 2.0. Хранить историю сообщений в базе данных.

Внешние API: VK API (OAuth), GitHub OAuth. Для чата в реальном времени — WebSocket (библиотека Socket.io).

Рекомендуемый стек: Node.js + Express + Socket.io, MongoDB или PostgreSQL, любой фронтенд.

Тема 6. Веб-приложение для поиска и сохранения фильмов с интеграцией базы данных The Movie Database (TMDb)

Описание: Разработать приложение для поиска фильмов, сериалов и актёров с использованием открытой базы TMDb (The Movie Database). Реализовать страницу с детальной информацией о фильме (постер, описание, рейтинг, трейлер с RuTube), а также возможность добавлять фильмы в личный список «Буду смотреть» с сохранением в локальную базу данных.

Внешние API: TMDb API (поиск, детальная информация, изображения), RuTube API (для встраивания трейлеров).

Рекомендуемый стек: React или Vue.js на фронтенде, Node.js или Firebase (для хранения списков пользователя).

Тема 7. Интеграция геосервисов (Яндекс.Карты или Leaflet) в веб-приложение для отображения объектов на карте

Описание: Разработать веб-приложение, отображающее на интерактивной карте набор объектов (кафе, аптеки, банкоматы и т.п.) с привязкой к текущему местоположению пользователя. Объекты могут загружаться из JSON-файла или из базы данных. Реализовать фильтрацию объектов по категориям и построение маршрута до выбранного объекта.

Внешние API: API Яндекс.Карт (геокодирование, отображение карты, построение маршрутов), Leaflet (с открытыми тайлами OpenStreetMap).

Рекомендуемый стек: HTML, CSS, JavaScript (библиотека Leaflet или API Яндекс.Карт), любой бэкенд для хранения объектов (например, JSON-сервер или Firebase).

Тема 8. Бот для мессенджера с интеграцией внешнего API (погода, новости, переводчик)

Описание: Разработать телеграм-бота, который взаимодействует с одним или несколькими внешними API и отвечает пользователю в чате. Примеры функций: бот-переводчик (API Yandex.Translate или DeepL), бот-прогноз погоды, бот-напоминание. Бот должен обрабатывать команды, поддерживать состояние диалога и логировать ошибки.

Внешние API: любой сторонний API по выбору студента (погода, перевод, новости, курсы валют).

Рекомендуемый стек: Python (библиотека aiogram), Node.js. База данных по желанию (для хранения пользовательских настроек).

Тема 9. Веб-приложение для сокращения ссылок с интеграцией сервиса аналитики (отслеживание переходов)

Описание: Разработать сервис, аналогичный bit.ly или clck.ru. Пользователь вводит длинную ссылку, система генерирует короткую и сохраняет пару в базу данных. При переходе по короткой ссылке происходит перенаправление на исходный URL. Дополнительно реализовать панель управления, где можно увидеть количество переходов по каждой ссылке и их географию (интеграция с сервисом геолокации по IP).

Внешние API: Сервис геолокации по IP (ip-api.com, ipstack).

Рекомендуемый стек: PHP + MySQL (стандартный стек для ссылочных сервисов) или Node.js + Express + MongoDB, простой HTML/CSS для интерфейса.

Тема 10. Интеграция облачного хранилища (Яндекс.Диск) в веб-приложение для загрузки и управления файлами

Описание: разработать веб-приложение, которое авторизуется в облачном хранилище пользователя через OAuth, отображает список файлов и папок, позволяет загружать новые файлы, скачивать и удалять существующие. Реализовать базовый файловый менеджер с использованием API облачного сервиса.

Внешние API: Яндекс.Диск REST API.

Рекомендуемый стек: Любой серверный язык (Python, Node.js, PHP) для безопасного хранения токенов, фронтенд на JavaScript.

СПИСОК ЛИТЕРАТУРЫ

1. Бабушкина, И. А. Практикум по объектно-ориентированному программированию : [12+] / И. А. Бабушкина, С. М. Окулов. – 6-е изд. – Москва : Лаборатория знаний, 2025. – 369 с. : ил., табл. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=719644

2. Апатова, Н. В. Алгоритмизация и программирование : учебное пособие : [16+] / Н. В. Апатова, М. А. Бакуменко. – Москва ; Вологда : Инфра-Инженерия, 2025. – 216 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=725622

3. Назарова, О. Б. Разработка реляционных баз данных с использованием CASE-средства All Fusion Data Modeler : учебно-методическое пособие : [16+] / О. Б. Назарова, О. Е. Масленникова. – 4-е изд., стер. – Москва : ФЛИНТА, 2025. – 75 с. : URL: https://biblioclub.ru/index.php?page=book&id=363420

4. Нагаева, И. А. Основы алгоритмизации и программирования : практикум : учебное пособие / И. А. Нагаева, И. А. Кузнецов. – 2-е изд., стер. – Москва : Директ-Медиа, 2025. – 168 с. URL: https://biblioclub.ru/index.php?page=book&id=722929

5. Ипатова, Э.Р. Методологии и технологии системного проектирования информационных систем : учебник / Э.Р. Ипатова, Ю.В. Ипатов. - 2-е изд., стер. - Москва : Издательство «Флинта», 2016. - 257 с. : табл., схем. - (Информационные технологии). - Библиогр.: с. 95-96. ISBN 978-5-89349-978-0 [Электронный ресурс]. URL:http://biblioclub.ru/index.php?page=book&id=79551https://biblioclub.ru/index.php?page=bo ok red&id=481536

6. Информатика : учебник / Н. В. Макарова, Л. А. Матвеев, В. Л. Бройдо [и др.] ; под ред. Н. В. Макаровой. – 3-е изд., перераб. – Москва : Финансы и статистика, 2024. – 769 с. : ил., табл. – URL: https://biblioclub.ru/index.php?page=book&id=721247

7. Майтак, Р. В. Python, Django, Data Science = [Python, Django, Наука о данных] : учебное пособие : [16+] / Р. В. Майтак, П. А. Пылов, А. В. Протодьяконов. – Москва ; Вологда : Инфра-Инженерия, 2025. – 516 с. – URL: https://biblioclub.ru/index.php?page=book&id=725620

8. Браун, И. Разработка веб-приложений с использованием React и Node.js : [16+] / И. Браун. — Санкт-Петербург : Питер, 2025. — 412 с. — ISBN 978-5-4461-2345-6.

7. Васильев, А. Н. JavaScript в примерах и задачах : практикум : [12+] / А. Н. Васильев. — Москва : Эксмо, 2024. — 288 с. — ISBN 978-5-04-123456-7.

8.Голубятников, И. В. Современные подходы к интеграции информационных систем : монография / И. В. Голубятников, Д. А. Кузнецов. — Москва : ИНФРА-М, 2025. — 156 с. — ISBN 978-5-16-018765-4.

9. Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов : [12+] / Д. Дакетт. — Москва : Эксмо, 2023. — 512 с. — ISBN 978-5-699-78945-6.

10.Зайцев, Е. А. REST API. Проектирование, разработка, документирование : учебное пособие / Е. А. Зайцев. — Москва : ДМК Пресс, 2025. — 224 с. — ISBN 978-5-97060-987-6.

Приложение А

(Оформление титульного листа для курсового проекта)

Министерство просвещения Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ КОЗЬМЫ МИНИНА»

Факультет _____
*Кафедра*_____
Специальность: 09.02.09 Веб-разработка,

КУРСОВОЙ ПРОЕКТ
по дисциплине «____»
на тему: «____»

СТУДЕНТ(КА) _____ *(личная подпись)*
*РУКОВОДИТЕЛЬ*_____

Нижний Новгород – 20__ г.

Приложение Б

(*Образец содержания)*

Оглавление

Учебное издание

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ КУРСОВЫХ ПРОЕКТОВ

Составитель:
Поначугин Александр Викторович