

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ИНГУШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФИЗИКО- МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
Кафедра «Информационные системы и технологии»**

СОГЛАСОВАНО

УТВЕРЖДАЮ

Руководитель образовательной
программы
_____/М.Х.Мальсагов
от «27» апреля 2026г

Декан физико-математического
факультета
_____/М.Х. Мальсагов
от «28» апреля 2026г..

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1.В.02 «Интернет программирование »

Направление подготовки

09.03.02 Информационные системы и технологии

Направленность (профиль подготовки)

Технологии искусственного интеллекта и анализа данных

Квалификация выпускника

Бакалавр

Форма обучения

Очная

Магас, 2026

1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ

Целями освоения дисциплины являются развитие способностей проводить обследование организаций, выявлять информационные потребности пользователей, формировать требования к информационной системе; разрабатывать и адаптировать прикладное программное обеспечение. Для достижения целей необходимо решить следующие

Задачи: – –

- изучить современные технологии разработки веб-приложений;
- развить умения анализа предметных областей и определения потребностей пользователей разрабатываемого программного обеспечения;
- сформировать у студентов практические навыки использования современных технологий веб-разработки.

Изучение дисциплины обеспечивает развитие у обучающихся навыков командной работы, межличностной коммуникации, принятия решений, лидерских качеств.

2. МЕСТО УЧЕБНОЙ ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП ВО

Дисциплина «Интернет-программирование» является дисциплиной базовой части ОП подготовки обучающихся по направлению 09.03.02 «Информационные системы и технологии». Дисциплина основывается на знаниях, умениях и навыках, полученных студентами в ходе изучения дисциплин «Алгоритмизация и программирование». Для успешного освоения дисциплины «Интернет-программирование», студент должен:

Знать:

- информационно-логические основы информатики;
- формы представления различных видов информации в памяти ЭВМ;
- типы графических изображений и форматы графических файлов;
- основные приемы работы с графическими редакторами.

Уметь:

- работать с компьютером как средством управления и обработки информации,

работать с информацией из различных источников, в том числе в глобальных компьютерных сетях;

- использовать современные графические системы для создания и коррективки изображений.

Владеть навыками:

- работы с основными средствами персонального компьютера, обеспечивающими взаимодействие с пользователем.

	Обобщенные трудовые	Трудовые функции
--	---------------------	------------------

Код и наименование профессионального стандарта	функции					
	Код	Наименование	Уровень квалификации	Наименование	Код	Уровень(по дуровень) квалификации
06.015 Специалист по информационным системам.	С	Выполнение работ и управление работами по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы.	6	Определение первоначальных требований заказчика к ИС и возможности их реализации в ИС на этапе предконтрактных работ	С/01.6	6
				Документирование существующих бизнес-процессов организации заказчика (реверс-инжиниринг бизнес-процессов организации)	С/07.6	6
				Разработка модели бизнес-процессов заказчика	С/08.6	6
				Разработка архитектуры ИС	С/14.6	6
				Проектирование и дизайн ИС	С/16.6	6
				Разработка баз данных ИС	С/17.6	6

3. КОМПЕТЕНЦИИ И ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ.

ПК -7	ПК-7 Способность разрабатывать и поддерживать веб-приложения с использованием современных технологий интернет-программирования.	ИД 1 ПК 7 Разрабатывает веб-приложения с интерактивным интерфейсом и серверной логикой, применяя современные инструменты разработки (Git, Docker).
--------------	---	--

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ «ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ»

4.1. Структура дисциплины Информатика и информационные технологии в профессиональной деятельности

Общая трудоемкость дисциплины составляет 2 зачетных единиц, 72ч.

№ п/п	Наименование разделов и тем дисциплины (модуля)	семестр	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)								Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)							
			Контактная работа					Самостоятельная работа										
			Всего	Лекции	Практические занятия	Лабораторные занятия	Др. виды контак. работ.	Всего	Курсовая работа(проект)	Подготовка к экзамену	Другие виды сам. работы	Собеседование	Коллоквиум	Проверка тестов	Проверка конт. работ	Проверка реферата	Проверка эссе и иных творческих работ	Курсовая работа(про-
1	Тема 1. Введение в интернет-программирование. Основы клиент-серверной архитектуры	1	2	2							2							
2	Тема 2. HTML и CSS: основы верстки веб-страниц	1	8	2		2					4							
3	Тема3. JavaScript: основы клиентского программирования	1	8	2		2					4							
4	Тема 4. Работа с DOM и событиями в JavaScript	1	7	1		2					4							
5	Тема 5. Асинхронное программирование: AJAX, Fetch API	1	7	1		2					4							
6	Тема 6. Основы серверного программи-	1	8	2		2					4							

[illegible]

4.2. Содержание дисциплины

Тема 1: Введение в интернет-программирование. Основы клиент-серверной архитектуры (4 часа)

- **Лекции (2 часа):** История развития веб-технологий. Клиент-серверная архитектура. Протоколы HTTP/HTTPS. Основные этапы разработки веб-приложений.
- **Самостоятельная работа (2 часа):** Изучение материалов по истории веб-технологий и основам работы протоколов HTTP/HTTPS.

Тема 2: HTML и CSS: основы верстки веб-страниц (8 часов)

- **Лекции (2 часа):** Структура HTML-документа. Семантическая разметка. Основы CSS: селекторы, стили, каскадирование.
- **Лабораторные занятия (2 часа):** Создание простой веб-страницы с использованием HTML и CSS.
- **Самостоятельная работа (4 часа):** Разработка макета страницы с адаптивным дизайном (responsive design) с использованием медиа-запросов.

Тема 3: JavaScript: основы клиентского программирования (8 часов)

- **Лекции (2 часа):** Основы синтаксиса JavaScript. Переменные, функции, объекты. Работа с консолью разработчика.
- **Лабораторные занятия (2 часа):** Написание простых скриптов: калькулятор, обработка пользовательского ввода.
- **Самостоятельная работа (4 часа):** Изучение дополнительных возможностей JavaScript (например, работа с массивами и циклами) и выполнение практических заданий.

Тема 4: Работа с DOM и событиями в JavaScript (7 часов)

- **Лекции (1 час):** Что такое DOM. Методы доступа и изменения элементов. Обработка событий (click, input и др.).
- **Лабораторные занятия (2 часа):** Создание интерактивной страницы с динамическим изменением контента.
- **Самостоятельная работа (4 часа):** Разработка мини-приложения (например, todo-list) с использованием DOM.

Тема 5: Асинхронное программирование: AJAX, Fetch API (7 часов)

- **Лекции (1 час):** Асинхронные запросы. Работа с AJAX и Fetch API. Обработка JSON.
- **Лабораторные занятия (2 часа):** Реализация запросов к публичному API (например, получение данных о погоде).
- **Самостоятельная работа (4 часа):** Изучение документации по Fetch API и выполнение заданий по работе с асинхронными запросами.

Тема 6: Основы серверного программирования (на примере Node.js или Python) (8 часов)

- **Лекции (2 часа):** Основы серверной разработки. Установка и настройка окружения (Node.js/Express или Python/Flask). Создание простого сервера.
- **Лабораторные занятия (2 часа):** Разработка сервера с базовыми маршрутами (GET, POST).
- **Самостоятельная работа (4 часа):** Изучение дополнительных возможностей серверных фреймворков и настройка окружения на локальной машине.

Тема 7: Работа с базами данных в веб-приложениях (SQL, NoSQL) (8 часов)

- **Лекции (2 часа):** Основы работы с базами данных. SQL (MySQL/PostgreSQL) и NoSQL (MongoDB). Подключение базы данных к серверу.
- **Лабораторные занятия (2 часа):** Создание базы данных и реализация CRUD-операций (Create, Read, Update, Delete).
- **Самостоятельная работа (4 часа):** Разработка небольшого приложения с базой данных (например, список пользователей).

Тема 8: Основы REST API: проектирование и реализация (7 часов)

- **Лекции (1 час):** Что такое REST. Принципы проектирования API. HTTP-методы и статус-коды.
- **Лабораторные занятия (2 часа):** Создание REST API для работы с данными (например, API для управления задачами).
- **Самостоятельная работа (4 часа):** Изучение документации по REST и разработка собственного API.

Тема 9: Фреймворки для веб-разработки (React, Django или Laravel) (8 часов)

- **Лекции (2 часа):** Обзор популярных фреймворков. Основы работы с React (или другим фреймворком). Компонентный подход.
- **Лабораторные занятия (2 часа):** Создание простого приложения с использованием фреймворка (например, список задач на React).
- **Самостоятельная работа (4 часа):** Изучение документации фреймворка и выполнение практических заданий.

Тема 10: Безопасность веб-приложений: защита от уязвимостей (3 часа)

- **Лекции (1 час):** Основные уязвимости (XSS, CSRF, SQL-инъекции). Методы защиты.
- **Самостоятельная работа (2 часа):** Изучение материалов по OWASP Top 10 и подготовка рекомендаций по безопасности.

Тема 11: Оптимизация и тестирование веб-приложений (2 часа)

- **Самостоятельная работа (2 часа):** Изучение методов оптимизации (сжатие изображений, кэширование) и написание тестов (unit-тесты).

Тема 12: Итоговый проект: разработка веб-приложения (2 часа)

- **Самостоятельная работа (2 часа):** Разработка итогового проекта (например, блог или интернет-магазин) с использованием изученных технологий.

5. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

При подготовке бакалавров используются следующие образовательные технологии:

1. Компьютерные классы с набором лицензионного базового программного обеспечения для проведения лабораторных занятий;
2. Дополнительные мультимедийные материалы.

6. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ.

6.1. План самостоятельной работы студентов

№	Тема	Вид самостоятельной работы	Задание	Рекомендуемая литература	Количество часов
1	Основы HTML и CSS	Написание кода	Создать статическую веб-страницу с использованием HTML и CSS	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5
2	Основы JavaScript	Написание кода	Написать скрипт для интерактивного элемента (например, кнопка)	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5
3	Работа с DOM и событиями в JavaScript	Написание кода	Реализовать обработку событий (например, клик, наведение)	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5
4	Асинхронное программирование (Promises, async/await)	Написание кода	Написать код для асинхронного запроса данных с API	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5

5	Основы работы с серверной частью (Node.js)	Написание кода	Создать простой сервер с использованием Node.js и Express	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5
6	Работа с базами данных (SQL/NoSQL)	Написание кода	Подключить базу данных к серверу и выполнить CRUD-операции	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5
7	Алгебра логики. Отрицание. Конъюнкция. Дизъюнкция. Импликация. Эквиваленция. Логические формулы. Логические схемы.	Написание кода	Сдача коллоквиума	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5
8	Основные понятия. Способы задания алгоритмов. Свойства алгоритмов.	Написание кода	Сдача коллоквиума	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5
9	Методы и средства защиты информации.	Написание кода	Защита реферата	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1.5
10	Тестирование веб-приложений (unit-тесты)	Написание тестов	Написать unit-тесты для клиентской или серверной части приложения	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1
11	Оптимизация производительности веб-приложений	Анализ и оптимизация	Проанализировать и оптимизировать загрузку страницы (например, Lazy Loading)	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1
12	Основы безопасности веб-приложений	Анализ и реализация	Реализовать базовые меры безопасности (например, защита от XSS)	Основная 1,2,3,4,5,6 Доп. 1,2,3,4 Интернет-ресурсы	1
	Всего				16

6.2. Методические указания по организации самостоятельной работы студентов

1. Успешное освоение курса требует напряженной самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя чтение лекций и рекомендованной литературы, решение задач, предлагаемых студентам на лекциях и практических занятиях, разбор проблемных ситуаций. Руководство и контроль за самостоятельной работой студента осуществляется в форме индивидуальных консультаций. Для активизации самостоятельной работы студентов и экономии времени, отводимого на лекционный курс, ряд тем выносятся на самостоятельное изучение. Самостоятельная работа со студентами проводится в часы самостоятельной работы в форме консультаций. Распределение часов руководства самостоятельной работой учитывает важность рассматриваемой темы и возможную сложность при освоении ее студентами. Самостоятельная работа студентов рассматривается как вид учебного труда, позволяющий целенаправленно формировать и развивать самостоятельность студента как личностное качество при выполнении различных видов заданий и проработке дополнительного учебного материала. Для успешного выполнения лабораторных работ, написания рефератов и подготовки к коллоквиуму, помимо материалов лекционных и практических занятий, необходимо использовать основную и дополнительную литературу, указанную в конце данной рабочей программы.

2. Лекции, презентации, методические указания и задания к лабораторным работам помещаются в групповые папки студентов, находящиеся на сервере университета и доступны студентам группы.

3. Методические указания содержат теорию по рассматриваемому вопросу, рекомендации по выполнению лабораторных работ.

6.3. Материалы для проведения текущего и промежуточного контроля знаний студентов

Контроль и оценка результатов освоения учебной дисциплины осуществляется преподавателем в процессе проведения лабораторных работ, тестирования, а также написания рефератов.

Оценка качества освоения учебной программы включает текущий контроль успеваемости, промежуточную аттестацию по итогам освоения дисциплины.

Текущий контроль проводится в форме: защиты лабораторных работ; кейс-задания; отчёта по проделанной внеаудиторной самостоятельной работы (защиты реферата, тест), контроля выполнения индивидуальных и групповых заданий.

Промежуточная аттестация по дисциплине проводится в форме экзамена.

6.4. Контроль освоения компетенций

№ п\п	Вид контроля	Контролируемые темы (разделы)	Компетенции, компоненты которых контролируются
1	Лабораторная работа	Основы интернет-программирования сновы информатики	ПК-7, ИД 1 ПК-7
2	Лабораторная работа	Прикладные программные средства	ПК-7, ИД 1 ПК-7
3	Лабораторная работа	Серверные технологии обработки информации	ПК-7, ИД 1 ПК-7

6.5. Критерии оценки текущей и промежуточной аттестации

Опрос устный

Опрос устный - диалог преподавателя со студентом, цель которого - систематизация и уточнение имеющихся у студента знаний, проверка его индивидуальных возможностей усвоения материала.

Устный опрос по основным терминам может проводиться в начале/конце лекционного или практического занятия в течение 15 -20 мин. Либо устный опрос проводится в течение всего практического занятия по заранее выданной тематике. Выбранный преподавателем студент может отвечать с места либо у доски.

Критериями оценки устного опроса являются: правильность ответа на вопросы, степень раскрытия сущности вопроса.

Оценка «отлично» — дан полный, всесторонний ответ на вопрос. Точность в определениях. Приведение примеров из практики.

Оценка «хорошо» — дан неполный ответ на вопрос. Допущены неточности при ответе. Допущены неточности в основных определениях.

Оценка «удовлетворительно» — имеются существенные недочеты при ответе. Вопрос раскрыт частично. Незнание базовых определений курса.

Оценка «неудовлетворительно» — вопрос не раскрыт или дан неверный ответ.

Тесты

Тесты - инструмент, с помощью которого педагог оценивает степень достижения студентом требуемых знаний, умений, навыков. Составление теста включает в себя создание выверенной системы вопросов, собственно процедуру проведения тестирования и способ измерения полученных результатов.

Критерии оценки теста: Оценка «отлично» выставляется при условии правильного ответа студента не менее чем 85 % тестовых заданий;

Оценка «хорошо» выставляется при условии правильного ответа студента не менее чем 70 % тестовых заданий;

Оценка «удовлетворительно» выставляется при условии правильного ответа студента не менее 51 %; .

Оценка «неудовлетворительно» выставляется при условии правильного ответа студента менее чем на 50 % тестовых заданий

Кейс - задания

Кейс - задания - проблемное задание, в котором обучающемуся предлагают осмыслить реальную профессионально-ориентированную ситуацию, необходимую для решения данной проблемы. Студент самостоятельно формулирует цель, находит и собирает информацию, анализирует ее, выдвигает гипотезы, ищет варианты решения проблемы, формулирует выводы, обосновывает оптимальное решение ситуации.

Критерии оценки кейс-заданий: Отметка «отлично» — задание выполнено в полном объеме с соблюдением необходимой последовательности действий; в ответе правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ ошибок. Отметка «хорошо» — задание выполнено правильно с учетом 1 -2 мелких погрешностей или 2-3 недочетов, исправленных самостоятельно по требованию преподавателя. Отметка «удовлетворительно» — задание выполнено правильно не менее чем наполовину, допущены 1 -2 погрешности или одна грубая ошибка.

Отметка «неудовлетворительно» — допущены две (и более) грубые ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя или задание не решено полностью.

Реферат

Критериями оценки реферата являются: новизна текста, обоснованность выбора источников литературы, степень раскрытия сущности вопроса, соблюдения требований к оформлению.

Оценка «отлично» — выполнены все требования к написанию реферата: обозначена проблема и обоснована её актуальность; сделан анализ различных точек зрения на рассматриваемую проблему и логично изложена собственная позиция; сформулированы выводы, тема раскрыта полностью, выдержан объём; соблюдены требования к внешнему оформлению.

Оценка «хорошо» — основные требования к реферату выполнены, но при этом допущены недочёты. В частности, имеются неточности в изложении материала; отсутствует логическая последовательность в суждениях; не выдержан объём реферата; имеются упущения в оформлении.

Оценка «удовлетворительно» — имеются существенные отступления от требований к реферированию. В частности: тема освещена лишь частично; допущены фактические ошибки в содержании реферата; отсутствуют выводы.

Оценка «неудовлетворительно» — тема реферата не раскрыта, обнаруживается существенное непонимание проблемы или реферат не представлен вовсе.

Практические контрольные задания (ПКЗ)

Критерии оценки практических контрольных заданий: Результат выполнения КР оценивается в баллах: "5" -отлично, "4" -хорошо, "3" -удовлетворительно, "2" -неудовлетворительно. Отметка «5» ставится, если:

- работа выполнена полностью;
- в решении нет математических ошибок (возможен один недочёт, описка, которая не является следствием незнания или непонимания учебного материала).

Отметка «4» ставится в следующих случаях:

- работа выполнена полностью, но допущены одна ошибка или есть два - три недочёта в выкладках решения;

Отметка «3» ставится, если:

- допущены две-три ошибки в вычислениях, при этом должно быть выполнено не менее 60% всей работы.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не обладает обязательными умениями по данной теме в полной мере, при этом выполнено менее 60%.

Контрольная работа

Контрольная работа - средство промежуточного контроля остаточных знаний и умений, состоит из вопросов или заданий, которые студент должен решить, выполнить. Знакомство с основной и дополнительной литературой, включая справочные издания, зарубежные источники, конспект основных положений, терминов, сведений, требующих для запоминания и являющихся основополагающими в этой теме.

Критерии оценки контрольной работы для студентов заочного отделения:

Оценка «зачтено» ставится за полные ответы на все вопросы.

Оценка «не зачтено» ставится, если освещены не все вопросы требуемого материала или не описано главное в содержании вопросов, или письменная работа не сдана.

Коллоквиум(в переводе с латинского «беседа, разговор») – форма текущего контроля знаний студентов, которая проводится в виде собеседования преподавателя и студента по самостоятельно подготовленной студентом теме.

Он применяется для проверки знаний по определенному разделу (или объемной теме) и принятия решения о том, можно ли переходить к изучению нового материала. Коллоквиум — это беседа со студентами, целью которой является выявление уровня овладения новыми знаниями. В отличие от семинара главное на коллоквиуме — это проверка знаний с целью их систематизации.

Целью коллоквиума является формирование у студента навыков анализа теоретических проблем на основе самостоятельного изучения учебной и научной литературы.

На коллоквиум выносятся крупные, проблемные, нередко спорные теоретические вопросы. Коллоквиум может проводиться по вопросам, обсуждавшимся на семинарах. Конкретные вопросы для коллоквиума студентам не сообщаются, однако заранее формулируются преподавателем. Предполагаемый объем ответа не должен быть большим (примерно 1,5-2 минуты), чтобы преподаватель мог успеть опросить всех студентов.

От студента требуется:

- владение изученным в ходе учебного процесса материалом, относящимся к рассматриваемой проблеме;
- наличие собственного мнения по обсуждаемым вопросам и умение его аргументировать.

Коллоквиум — это не только форма контроля, но и метод углубления, закрепления знаний студентов, так как в ходе собеседования преподаватель разъясняет сложные вопросы, возникающие у студента в процессе изучения данного источника.

Задача коллоквиума добиться глубокого изучения отобранного материала, пробудить у студента стремление к чтению дополнительной экономической литературы.

Подготовка к проведению коллоквиума.

Подготовка к коллоквиуму предполагает несколько этапов:

1. Подготовка к коллоквиуму начинается с установочной консультации преподавателя, на которой он разъясняет развернутую тематику проблемы, рекомендует литературу для изучения и объясняет процедуру проведения коллоквиума.

2. Как правило, на самостоятельную подготовку к коллоквиуму студенту отводится 3–4 недели. Подготовка включает в себя изучение рекомендованной литературы и (по указанию преподавателя) конспектирование важнейших источников.

3. Коллоквиум проводится в форме индивидуальной беседы преподавателя с каждым студентом или беседы в небольших группах (3–5 человек).

4. Преподаватель задает несколько кратких конкретных вопросов, позволяющих выяснить степень добросовестности работы с литературой, контролирует конспект. Далее более подробно обсуждается какая-либо сторона проблемы, что позволяет оценить уровень понимания.

5. По итогам коллоквиума выставляется дифференцированная оценка, имеющая большой удельный вес в определении текущей успеваемости студента.

Особенности и порядок сдачи коллоквиума. Студент может себя считать готовым к сдаче коллоквиума по избранной работе, когда у него есть им лично составленный и обработанный конспект сдаваемой работы, он знает структуру работы в целом, содержание работы в целом или отдельных ее разделов (глав); умеет раскрыть рассматриваемые проблемы и высказать свое отношение к прочитанному и свои сомнения, а также знает, как убедить преподавателя в правоте своих суждений.

Проведение коллоквиума позволяет студенту приобрести опыт работы над первоисточниками, что в дальнейшем поможет с меньшими затратами времени работать над литературой по курсовой работе и при подготовке к экзаменам.

Экзамен

Экзамен - итоговая форма оценки знаний.

Проводится в заданный срок, согласно графику учебного процесса.

Критерии оценки при проведении экзамена:

Оценка "отлично" ставится, если студент обнаружил полное знание учебно-программного материала, успешно выполняет предусмотренные в программе задания, усвоил основную литературу, рекомендованную в программе. Ответ полный и правильный на основании изученного материала. Выдвинутые положения аргументированы и иллюстрированы примерами. Материал изложен в определенной логической последовательности, осознанно, литературным языком, с использованием современных научных терминов; ответ самостоятельный. Студент уверенно отвечает на дополнительные вопросы

Оценка «хорошо» ставится в том случае, когда студент обнаруживает полное знание учебного материала, демонстрирует систематический характер знаний по дисциплине. Ответ полный и правильный, подтвержден примерами; но их обоснование не аргументировано, отсутствует собственная точка зрения. Материал изложен в определенной логической последовательности, при этом допущены 2-3 несущественные погрешности, исправленные по требованию экзаменатора. Студент испытывает незначительные трудности в ответах на дополнительные вопросы. Материал изложен осознанно, самостоятельно, с использованием современных научных

терминов, литературным языком, при этом могут допускаться некоторые погрешности в ответе на зачете, если студент обладает необходимыми знаниями для их устранения под руководством преподавателя.

Оценка «удовлетворительно» ставится в том случае, когда студент обнаруживает знание основного программного материала по дисциплине, но допускает погрешности в ответе. Ответ недостаточно логически выстроен, самостоятелен. Основные понятия употреблены правильно, но обнаруживается недостаточное раскрытие теоретического материала. Выдвигаемые положения недостаточно аргументированы и не подтверждены примерами; ответ носит преимущественно описательный характер. Студент испытывает достаточные трудности в ответах на вопросы. Научная терминология используется недостаточно.

Оценка «неудовлетворительно» выставляется студенту, обнаружившему проблемы в знаниях основного учебного материала по дисциплине. При ответе обнаружено непонимание студентом основного содержания теоретического материала по дисциплине. При ответе обнаружено непонимание студентом основного содержания теоретического материала или допущен ряд существенных ошибок, которые студент не может исправить при наводящих вопросах экзаменатора. Студент подменил научное обоснование проблем рассуждением бытового плана. Ответ носит поверхностный характер; наблюдаются неточности в использовании научной терминологии.

6.6. Типовые лабораторные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций при изучении учебной дисциплины в процессе освоения образовательной программы

6.6.1. Типовой вариант задания на лабораторную работу

Задание 1. Создание статической веб-страницы

Описание: Разработайте веб-страницу с использованием HTML и CSS. Страница должна включать заголовок, текст, изображение и навигационное меню.

Критерии оценки:

- Корректность структуры HTML (правильное использование тегов, семантика).
- Применение стилей CSS (настройка цветов, шрифтов, отступов).

- Адаптивность страницы для разных разрешений экрана.
Ожидаемые результаты: Статическая веб-страница, которая корректно отображается в браузере.
Связь с компетенцией: ПК-7 — базовые навыки разработки интерфейса веб-приложения.
-

Задание 2. Добавление интерактивности с JavaScript

Описание: На основе страницы из задания 1 добавьте интерактивный элемент: кнопку, которая при нажатии меняет цвет фона страницы.

Критерии оценки:

- Корректность работы JavaScript-кода.
 - Правильная обработка события (например, onclick).
 - Отсутствие ошибок в консоли браузера.
- Ожидаемые результаты: Страница с работающей кнопкой, которая изменяет цвет фона при нажатии.
- Связь с компетенцией: ПК-7 — разработка интерактивного интерфейса.
-

Задание 3. Работа с DOM и событиями

Описание: Создайте форму с полями ввода (имя, email) и кнопкой "Отправить".

При нажатии на кнопку отображайте введенные данные в виде списка под формой без перезагрузки страницы.

Критерии оценки:

- Корректная работа с DOM (добавление элементов на страницу).
 - Правильная обработка событий (например, submit).
 - Валидация полей (например, проверка формата email).
- Ожидаемые результаты: Форма, которая добавляет введенные данные в список под формой без перезагрузки страницы.
- Связь с компетенцией: ПК-7 — разработка интерактивного интерфейса.
-

Задание 4. Асинхронный запрос данных

Описание: Используя Fetch API, получите данные с публичного API (например, JSONPlaceholder) и отобразите их на странице в виде таблицы.

Критерии оценки:

- Корректность асинхронного запроса (использование Promises или async/await).
 - Правильное отображение данных в таблице.
 - Обработка ошибок (например, если API недоступен).
- Ожидаемые результаты: Страница с таблицей, содержащей данные, полученные из API.

Связь с компетенцией: ПК-7 — работа с клиентской частью веб-приложения.

Задание 5. Создание простого сервера

Описание: Используя Node.js и Express, создайте сервер, который возвращает JSON-ответ на GET-запрос (например, список пользователей).

Критерии оценки:

- Корректная настройка сервера.
- Правильный формат JSON-ответа.
- Доступность сервера по указанному порту.

Ожидаемые результаты: Работающий сервер, который возвращает JSON-данные при запросе.

Связь с компетенцией: ПК-7 — разработка серверной логики.

Задание 6. Работа с базой данных

Описание: Подключите базу данных (например, SQLite или MongoDB) к серверу из задания 5. Реализуйте CRUD-операции (создание, чтение, обновление, удаление записей).

Критерии оценки:

- Корректное подключение базы данных.
- Реализация всех CRUD-операций.
- Обработка ошибок (например, дубликат записи).

Ожидаемые результаты: Сервер, который выполняет CRUD-операции с базой данных.

Связь с компетенцией: ПК-7 — разработка серверной логики.

Задание 7. Создание REST API

Описание: На основе сервера из задания 6 разработайте REST API с эндпоинтами для получения, добавления, обновления и удаления данных.

Критерии оценки:

- Корректная структура REST API (поддержка методов GET, POST, PUT, DELETE).
- Правильная обработка запросов и ответов.
- Наличие документации API (например, в файле README).

Ожидаемые результаты: Работающий REST API, который можно протестировать (например, через Postman).

Связь с компетенцией: ПК-7 — разработка серверной логики.

Задание 8. Интеграция клиентской и серверной частей

Описание: Используя клиентскую часть из задания 4 и сервер из задания 7, реализуйте взаимодействие: клиент отправляет запросы к серверу и отображает полученные данные на странице.

Критерии оценки:

- Корректное взаимодействие между клиентом и сервером.
- Обработка ошибок (например, 404, 500).
- Актуальность отображаемых данных.

Ожидаемые результаты: Веб-приложение, в котором клиентская часть взаимодействует с сервером через API.

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — разработка веб-приложения с интерактивным интерфейсом и серверной логикой.

Задание 9. Использование Git для контроля версий

Описание: Создайте репозиторий на GitHub, загрузите код из задания 8, сделайте несколько коммитов (например, добавление новой функции, исправление бага).

Критерии оценки:

- Корректная инициализация репозитория.
- Наличие минимум 3 осмысленных коммитов.
- Читаемые сообщения коммитов.

Ожидаемые результаты: Репозиторий на GitHub с историей коммитов.

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — применение современных инструментов разработки (Git).

Задание 10. Контейнеризация приложения с Docker

Описание: Создайте Dockerfile для сервера из задания 7, соберите образ и запустите контейнер. Убедитесь, что сервер доступен из контейнера.

Критерии оценки:

- Корректный Dockerfile (указаны зависимости, порты).
- Успешная сборка и запуск контейнера.
- Доступность сервера через указанный порт.

Ожидаемые результаты: Запущенный контейнер с работающим сервером.

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — применение современных инструментов разработки (Docker).

(Дополнительный материал смотреть в Приложении)

6.6.2. Типовой тест промежуточной аттестации

Вопрос 1 (Теоретический, с выбором ответа)

Что из перечисленного является основным преимуществом использования семантических тегов в HTML?

- а) Ускорение загрузки страницы.
- б) Улучшение читаемости кода и доступности для поисковых систем.
- в) Автоматическое применение стилей CSS.
- г) Упрощение работы с JavaScript.

Правильный ответ: б) Улучшение читаемости кода и доступности для поисковых систем.

Связь с компетенцией: ПК-7 — знание основ разработки интерфейса веб-приложения.

Вопрос 2 (Теоретический, с выбором ответа)

Какой HTTP-метод используется для обновления данных в REST API?

- а) GET
- б) POST
- в) PUT
- г) DELETE

Правильный ответ: в) PUT

Связь с компетенцией: ПК-7 — знание основ разработки серверной логики.

Вопрос 3 (Практический, анализ кода)

Дан следующий JavaScript-код:

```
document.getElementById("myButton").addEventListener("click", function() {  
    document.getElementById("myText").style.color = "blue";  
});
```

Что произойдёт при нажатии на элемент с id="myButton"?

Ожидаемый ответ: При нажатии на элемент с id="myButton" цвет текста элемента с id="myText" изменится на синий.

Связь с компетенцией: ПК-7 — разработка интерактивного интерфейса.

Вопрос 4 (Практический, написание кода)

Напишите JavaScript-код, который с помощью Fetch API запрашивает данные с публичного API (например, <https://jsonplaceholder.typicode.com/posts/1>) и выводит полученное значение поля "title" в консоль.

Ожидаемый ответ:

```
fetch("https://jsonplaceholder.typicode.com/posts/1")  
    .then(response => response.json())  
    .then(data => console.log(data.title))  
    .catch(error => console.error("Ошибка:", error));
```

Связь с компетенцией: ПК-7 — работа с клиентской частью веб-приложения.

Вопрос 5 (Теоретический, с выбором ответа)

Какой из перечисленных инструментов используется для контейнеризации приложений?

- а) Git

- б) Docker
- в) Node.js
- г) Express

Правильный ответ: б) Docker

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — применение современных инструментов разработки (Docker).

Вопрос 6 (Практический, анализ кода)

Дан следующий код сервера на Node.js с использованием Express:

```
const express = require("express");
const app = express();
app.get("/users", (req, res) => {
    res.json([ { id: 1, name: "Анна" } ]);
});
app.listen(3000, () => console.log("Сервер запущен на порту 3000"));
```

Что вернёт сервер при GET-запросе на адрес <http://localhost:3000/users?>

Ожидаемый ответ: Сервер вернёт JSON-объект: [{ "id": 1, "name": "Анна" }]

Связь с компетенцией: ПК-7 — разработка серверной логики.

Вопрос 7 (Теоретический, открытый вопрос)

Назови три основные команды Git, которые используются для работы с репозиторием (инициализация, добавление изменений, фиксация изменений).

Ожидаемый ответ:

- git init — инициализация репозитория.
- git add — добавление изменений в индекс.
- git commit — фиксация изменений.

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — применение современных инструментов разработки (Git).

Вопрос 8 (Практический, написание кода)

Напишите SQL-запрос для получения всех записей из таблицы "users", где возраст (поле age) больше 25.

Ожидаемый ответ:

```
SELECT * FROM users WHERE age > 25;
```

Связь с компетенцией: ПК-7 — работа с базами данных в серверной логике.

Вопрос 9 (Теоретический, с выбором ответа)

Что из перечисленного является примером уязвимости, связанной с безопасностью веб-приложений?

- а) Медленная загрузка страницы.
- б) Cross-Site Scripting (XSS).
- в) Использование устаревшего фреймворка.
- г) Неправильное форматирование CSS.

Правильный ответ: б) Cross-Site Scripting (XSS).

Связь с компетенцией: ПК-7 — знание основ безопасности веб-приложений.

Вопрос 10 (Практический, написание кода)

Напишите Dockerfile для сервера на Node.js, который запускает приложение на порту 3000.

Ожидаемый ответ:

```
FROM node:16
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "index.js"]
```

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — применение современных инструментов разработки (Docker).

6.6.3.Перечень примерных тем рефератов

1. **История развития веб-технологий: от HTML до современных фреймворков**
Описание: Исследуйте эволюцию веб-технологий, начиная с первых версий HTML и CSS, до появления JavaScript и современных фреймворков, таких как React и Vue. Опишите, как эти изменения повлияли на разработку веб-приложений.
2. **Сравнительный анализ клиентских фреймворков: React, Vue и Angular**
Описание: Проведите сравнение трёх популярных фреймворков для разработки клиентской части веб-приложений. Оцените их преимущества, недостатки, производительность и области применения.
3. **Роль REST API в современных веб-приложениях**
Описание: Изучите, что такое REST API, его принципы и архитектуру. Опишите, как REST API используется для взаимодействия между клиентской и серверной частями веб-приложений, и приведите примеры.
4. **Использование Git в командной разработке веб-приложений**
Описание: Рассмотрите, как Git помогает в управлении версиями кода при командной

разработке. Опишите основные команды, ветвление (branching) и процесс работы с pull request'ами.

5. **Контейнеризация веб-приложений с помощью Docker: преимущества и вызовы**
Описание: Исследуйте, как Docker используется для контейнеризации веб-приложений. Опишите процесс создания Dockerfile, преимущества контейнеризации и возможные сложности при внедрении.
6. **Асинхронное программирование в JavaScript: Promises и async/await**
Описание: Изучите концепцию асинхронного программирования в JavaScript. Сравните использование Promises и async/await, приведите примеры кода и объясните, как это улучшает работу веб-приложений.
7. **Безопасность веб-приложений: основные уязвимости и способы их предотвращения**
Описание: Рассмотрите распространённые уязвимости веб-приложений, такие как XSS (Cross-Site Scripting) и SQL-инъекции. Опишите методы защиты, включая валидацию данных и использование HTTPS.
8. **Сравнение реляционных и нереляционных баз данных в веб-разработке**
Описание: Проведите сравнение реляционных (например, MySQL) и нереляционных (например, MongoDB) баз данных. Оцените их применимость в веб-приложениях, преимущества и недостатки.
9. **Оптимизация производительности веб-приложений: лучшие практики**
Описание: Исследуйте методы оптимизации производительности веб-приложений, такие как минимизация запросов, сжатие данных, Lazy Loading и кэширование. Приведите примеры их применения.
10. **Роль Progressive Web Apps (PWA) в будущем веб-разработки**
Описание: Изучите концепцию Progressive Web Apps, их преимущества (например, работа оффлайн, push-уведомления) и ограничения. Опишите, как PWA могут изменить подход к разработке веб-приложений.
11. **Использование WebSocket для создания интерактивных веб-приложений**
Описание: Рассмотрите технологию WebSocket и её применение для создания интерактивных приложений, таких как чаты или онлайн-игры. Сравните WebSocket с традиционными HTTP-запросами.

12. Современные подходы к адаптивной и кроссбраузерной вёрстке

Описание: Изучите методы создания адаптивных веб-страниц, которые корректно отображаются на разных устройствах и браузерах. Опишите использование медиа-запросов, Flexbox и Grid.

6.6.6. Примерные вопросы к зачету

1. **Что такое семантические теги в HTML, и почему их использование важно при разработке веб-страниц? Приведите примеры таких тегов.**

Описание: Вопрос проверяет знание основ HTML и понимание важности семантики для доступности и SEO.

2. **Опишите, как работает модель DOM (Document Object Model). Как JavaScript может взаимодействовать с DOM для изменения содержимого страницы? Приведите пример кода.**

Описание: Вопрос направлен на проверку понимания работы с DOM и базовых навыков JavaScript.

3. **В чём разница между синхронным и асинхронным кодом в JavaScript? Объясните, как работают Promises и async/await, и приведите пример асинхронного запроса данных.**

Описание: Вопрос проверяет знание асинхронного программирования, важного для работы с API и внешними данными.

4. **Что такое REST API? Опишите основные HTTP-методы, используемые в REST, и для чего они применяются.**

Описание: Вопрос направлен на проверку понимания архитектуры REST и её роли в веб-разработке.

5. **Как Node.js используется для серверной разработки? В чём его преимущества по сравнению с традиционными серверными технологиями, такими как PHP?**

Описание: Вопрос проверяет знание серверной разработки и понимание особенностей Node.js.

6. **Опишите процесс подключения базы данных к серверу на Node.js. Какие шаги нужно выполнить, чтобы реализовать CRUD-операции (создание, чтение, обновление, удаление)?**

Описание: Вопрос направлен на проверку практических навыков работы с базами данных..

7. **Что такое Git, и как он используется в разработке веб-приложений? Назови основные команды для работы с репозиторием и опиши процесс создания ветки и слияния изменений.**

Описание: Вопрос проверяет знание инструмента Git и его применения в командной разработке.

8. **Объясните, что такое Docker и как он помогает в разработке веб-приложений. Опишите, как создать Dockerfile для сервера на Node.js.**

Описание: Вопрос направлен на проверку понимания контейнеризации и её применения.

9. **Какие существуют уязвимости веб-приложений, связанные с безопасностью? Опишите, как можно защитить приложение от XSS (Cross-Site Scripting) атак.**

Описание: Вопрос проверяет знание основ безопасности веб-приложений.

10. **Что такое адаптивная вёрстка, и какие технологии используются для её реализации? Приведите пример медиа-запроса для изменения стилей на мобильных устройствах.**

Описание: Вопрос направлен на проверку навыков создания адаптивных интерфейсов.

11. **Как можно оптимизировать производительность веб-приложения? Назови не менее трёх методов и объясни, как они работают.**

Описание: Вопрос проверяет понимание методов оптимизации, важных для создания эффективных приложений.

12. **Опишите процесс интеграции клиентской и серверной частей веб-приложения. Как клиент может отправить запрос к серверу и обработать полученные данные? Приведите пример кода.**

Описание: Вопрос направлен на проверку практических навыков интеграции клиент-серверного взаимодействия.

7. УЧЕБНО-МЕТОДИЧЕСКОЕ И МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ(МОДУЛЯ) ИНФОРМАТИКА

7.1. Учебная литература:

Основная литература

1. Флэнаган Д. JavaScript: Полное руководство. — 7-е изд. — М.: Вильямс, 2020.
Описание: Книга охватывает основы JavaScript, включая работу с DOM, асинхронное программирование и современные стандарты ES6+, что необходимо для разработки интерактивных интерфейсов.
2. Браун Э. Веб-разработка с использованием Node.js и Express. — СПб.: Питер, 2019.
Описание: Учебное пособие по созданию серверной части веб-приложений с использованием Node.js и Express, включая работу с REST API и базами данных.
3. Маккоу Д. HTML и CSS: Путь к совершенству. — М.: Эксмо, 2018.
Описание: Книга посвящена созданию современных веб-страниц с использованием HTML и CSS, включая адаптивную верстку и кроссбраузерность.
4. Резиг Дж. Секреты JavaScript ниндзя. — 2-е изд. — М.: Вильямс, 2017.
Описание: Пособие для углубленного изучения JavaScript, включая работу с событиями, асинхронность и оптимизацию кода.
5. Шилдт Г. SQL: Полное руководство. — М.: Вильямс, 2019.
Описание: Книга по основам работы с реляционными базами данных, включая написание запросов и интеграцию с веб-приложениями.

Дополнительная литература

1. Хоган Б. HTML5 и CSS3: Веб-разработка по стандартам. — М.: Питер, 2016.
Описание: Пособие по современным стандартам HTML5 и CSS3, включая работу с мультимедиа и анимациями.
2. Симпсон К. Вы не знаете JS: Асинхронность и производительность. — М.: О'Рейли, 2018.
Описание: Книга из серии "Вы не знаете JS", посвящённая асинхронному программированию и оптимизации производительности.
3. Гарретт Дж. RESTful API: Проектирование и разработка. — СПб.: Питер, 2020.
Описание: Пособие по проектированию REST API, включая лучшие практики и примеры реализации.
4. Локхарт А. Docker для разработчиков. — М.: ДМК Пресс, 2021.
Описание: Книга по основам работы с Docker, включая создание контейнеров и их использование в веб-разработке

7.2 Электронные образовательные ресурсы

Название ресурса	Ссылка/доступ
Электронная библиотека онлайн «Единое окно к образовательным ресурсам»	http://window.edu.ru
«Образовательный ресурс России»	http://school-collection.edu.ru
Федеральный образовательный портал: учреждения, программы, стандарты, ВУЗы, тесты ЕГЭ, ГИА	http://www.edu.ru
Федеральный центр информационно-образовательных ресурсов (ФЦИОР)	http://fcior.edu.ru
Русская виртуальная библиотека	http://rvb.ru
Кабинет русского языка и литературы	http://ruslit.ioso.ru
Национальный корпус русского языка	http://ruscorpora.ru
Научная электронная библиотека «e-Library»	http://elibrary.ru/defaultx.asp
Электронно-библиотечная система IPRbooks	http://www.iprbookshop.ru
Электронно-библиотечная система ИнГГУ	https://lib.inggu.ru/
Информационно-правовая система «Гарант»	Сетевая версия, доступна со всех компьютеров в корпоративной сети ИнГГУ

7.3. Программное обеспечение

Лицензионное программное обеспечение для проведения лабораторных занятий:

- *Microsoft Windows*
- *Visual studio code*
- *Интернет*

7.4. Материально-техническое обеспечение

Наименование	Назначение
Компьютерный класс	Лабораторные работы
ПК -13 шт.	
Принтер - 1шт.	
Сетевое оборудование – 1	

Рабочая программа дисциплины «Интернет - программирование» **состалена** в соответствии с требованиями ФГОС ВО по направлению подготовки 09.03.02 «Информационные системы и технологии», профиль «Технологии искусственного интеллекта и анализа данных», утвержденного приказом Министерства образования и науки Российской Федерации от «19» сентября 2017 г. № 926 (ред. от 08.02.2021г.).

Программу составили:

К.педаг.наук, старший преподаватель кафедры «Информационные системы и технологии» Ажигов А. М.

Программа одобрена на заседании кафедры «Информационные системы и технологии»

Протокол № 8 от «27» апреля 2026 года

Программа одобрена Учебно-методической комиссией физико-математического факультета

Протокол № 6 от «28» апреля 2026 года

**Сведения о переутверждении программы на очередной учебный год и
регистрации изменений**

Учебный год	Решение кафедры (№ протокола, дата)	Внесенные изменения	Подпись зав. кафедрой

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1.О.07 «Интернет программирование»

Направление подготовки

09.03.02 Информационные системы и технологии

Направленность (профиль подготовки)

Технологии искусственного интеллекта и анализа данных

Квалификация выпускника

Бакалавр

Форма обучения

Очная

Магас, 2025

1. Перечень компетенций с указанием этапов их формирования в процессе

освоения образовательной программы

В процессе освоения образовательной программы компетенции формируются по следующим этапам:

- 1) начальный этап дает общее представление о виде деятельности, основных закономерностях функционирования объектов профессиональной деятельности, методов и алгоритмов решения практических задач;
- 2) основной этап позволяет решать типовые задачи, принимать профессиональные и управленческие решения по известным алгоритмам, правилам и методикам;
- 3) завершающий этап предполагает готовность решать практические задачи повышенной сложности, нетиповые задачи, принимать профессиональные и управленческие решения в условиях неполной определенности, при недостаточном документальном, нормативном и методическом обеспечении.

При освоении дисциплины (модуля) компетенции, закрепленные за ней, реализуются по темам (разделам) дисциплины (модуля), в определенной степени (полностью или в оговоренной части) и на определенном этапе, что приведено в Таблице 1.

Таблица 1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы

Код компетенции	Наименование компетенции	Код и наименование индикатора	В результате освоения дисциплины обучающийся должен:
ПК-7	ПК-7 Способность разрабатывать и поддерживать веб-приложения с использованием современных технологий интернет-программирования.	ИД 1 ПК 7Разрабатывает веб-приложения с интерактивным интерфейсом и серверной логикой, применяя современные инструменты разработки (Git, Docker).	

Критерии оценивания образовательных результатов обучающегося во время промежуточной аттестации

Экзамен

Экзамен - итоговая форма оценки знаний.

Проводится в заданный срок, согласно графику учебного процесса.

Критерии оценки при проведении экзамена:

Оценка «отлично» ставится, если студент обнаружил полное знание учебно-программного материала, успешно выполняет предусмотренные в программе задания, усвоил основную литературу, рекомендованную в программе. Ответ полный и правильный на основании изученного материала. Выдвинутые положения аргументированы и иллюстрированы примерами. Материал изложен в определенной логической последовательности, осознанно, литературным языком, с использованием современных научных терминов; ответ самостоятельный. Студент уверенно отвечает на дополнительные вопросы

Оценка «хорошо» ставится в том случае, когда студент обнаруживает полное знание учебного материала, демонстрирует систематический характер знаний по дисциплине. Ответ полный и правильный, подтвержден примерами; но их обоснование не аргументировано, отсутствует собственная точка зрения. Материал изложен в определенной логической последовательности, при этом допущены 2-3 несущественные погрешности, исправленные по требованию экзаменатора. Студент испытывает незначительные трудности в ответах на дополнительные вопросы. Материал изложен осознанно, самостоятельно, с использованием современных научных терминов, литературным языком, при этом могут допускаться некоторые погрешности в ответе на зачете, если студент обладает необходимыми знаниями для их устранения под руководством преподавателя.

Оценка «удовлетворительно» ставится в том случае, когда студент обнаруживает знание основного программного материала по дисциплине, но допускает погрешности в ответе. Ответ недостаточно логически выстроен, самостоятелен. Основные понятия употреблены правильно, но обнаруживается недостаточное раскрытие теоретического материала. Выдвигаемые положения недостаточно аргументированы и не подтверждены примерами; ответ носит преимущественно описательный характер. Студент испытывает достаточные трудности в ответах на вопросы. Научная терминология используется недостаточно.

Оценка «неудовлетворительно» выставляется студенту, обнаружившему проблемы в знаниях основного учебного материала по дисциплине. При ответе обнаружено непонимание студентом основного содержания теоретического материала по дисциплине. При ответе обнаружено непонимание студентом основного содержания теоретического материала или допущен ряд существенных ошибок, которые студент не может исправить при наводящих вопросах экзаменатора. Студент подменил научное обоснование проблем рассуждением бытового плана. Ответ носит поверхностный характер; наблюдаются неточности в использовании научной терминологии.

3. Типовые контрольные задания или иные материалы, необходимые для оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

3.1. Типовой вариант задания на лабораторную работу

Задание 1. Создание статической веб-страницы

Описание: Разработайте веб-страницу с использованием HTML и CSS. Страница должна включать заголовок, текст, изображение и навигационное меню.

Критерии оценки:

- Корректность структуры HTML (правильное использование тегов, семантика).
- Применение стилей CSS (настройка цветов, шрифтов, отступов).
- Адаптивность страницы для разных разрешений экрана.

Ожидаемые результаты: Статическая веб-страница, которая корректно отображается в браузере.

Связь с компетенцией: ПК-7 — базовые навыки разработки интерфейса веб-приложения.

Задание 2. Добавление интерактивности с JavaScript

Описание: На основе страницы из задания 1 добавьте интерактивный элемент: кнопку, которая при нажатии меняет цвет фона страницы.

Критерии оценки:

- Корректность работы JavaScript-кода.
- Правильная обработка события (например, onclick).
- Отсутствие ошибок в консоли браузера.

Ожидаемые результаты: Страница с работающей кнопкой, которая изменяет цвет фона при нажатии.

Связь с компетенцией: ПК-7 — разработка интерактивного интерфейса.

Задание 3. Работа с DOM и событиями

Описание: Создайте форму с полями ввода (имя, email) и кнопкой "Отправить". При нажатии на кнопку отображайте введенные данные в виде списка под формой без перезагрузки страницы.

Критерии оценки:

- Корректная работа с DOM (добавление элементов на страницу).
- Правильная обработка событий (например, submit).
- Валидация полей (например, проверка формата email).

Ожидаемые результаты: Форма, которая добавляет введенные данные в список под формой без перезагрузки страницы.

Связь с компетенцией: ПК-7 — разработка интерактивного интерфейса.

Задание 4. Асинхронный запрос данных

Описание: Используя Fetch API, получите данные с публичного API (например, JSONPlaceholder) и отобразите их на странице в виде таблицы.

Критерии оценки:

- Корректность асинхронного запроса (использование Promises или async/await).
- Правильное отображение данных в таблице.
- Обработка ошибок (например, если API недоступен).

Ожидаемые результаты: Страница с таблицей, содержащей данные, полученные из API.

Задание 5. Создание простого сервера

Описание: Используя Node.js и Express, создайте сервер, который возвращает JSON-ответ на GET-запрос (например, список пользователей).

Критерии оценки:

- Корректная настройка сервера.
- Правильный формат JSON-ответа.
 - Доступность сервера по указанному порту.

Ожидаемые результаты: Работающий сервер, который возвращает JSON-данные при запросе..

Задание 6. Работа с базой данных

Описание: Подключите базу данных (например, SQLite или MongoDB) к серверу из задания 5. Реализуйте CRUD-операции (создание, чтение, обновление, удаление записей).

Критерии оценки:

- Корректное подключение базы данных.
- Реализация всех CRUD-операций.
 - Обработка ошибок (например, дубликат записи).

Ожидаемые результаты: Сервер, который выполняет CRUD-операции с базой данных.

Задание 7. Создание REST API

Описание: На основе сервера из задания 6 разработайте REST API с эндпоинтами для получения, добавления, обновления и удаления данных.

Критерии оценки:

- Корректная структура REST API (поддержка методов GET, POST, PUT, DELETE).
- Правильная обработка запросов и ответов.
 - Наличие документации API (например, в файле README).

Ожидаемые результаты: Работающий REST API, который можно протестировать (например, через Postman).

Задание 8. Интеграция клиентской и серверной частей

Описание: Используя клиентскую часть из задания 4 и сервер из задания 7, реализуйте взаимодействие: клиент отправляет запросы к серверу и отображает полученные данные на странице.

Критерии оценки:

- Корректное взаимодействие между клиентом и сервером.
- Обработка ошибок (например, 404, 500).
 - Актуальность отображаемых данных.

Ожидаемые результаты: Веб-приложение, в котором клиентская часть взаимодействует с сервером через API.

Задание 9. Использование Git для контроля версий

Описание: Создайте репозиторий на GitHub, загрузите код из задания 8, сделайте несколько коммитов (например, добавление новой функции, исправление бага).

Критерии оценки:

- Корректная инициализация репозитория.
- Наличие минимум 3 осмысленных коммитов.
 - Читаемые сообщения коммитов.

Ожидаемые результаты: Репозиторий на GitHub с историей коммитов.

Задание 10. Контейнеризация приложения с Docker

Описание: Создайте Dockerfile для сервера из задания 7, соберите образ и запустите контейнер. Убедитесь, что сервер доступен из контейнера.

Критерии оценки:

- Корректный Dockerfile (указаны зависимости, порты).
- Успешная сборка и запуск контейнера.

1. Доступность сервера через указанный порт.
Ожидаемые результаты: Запущенный контейнер с работающим сервером.

3.2. Типовой тест промежуточной аттестации

Вопрос 1 (Теоретический, с выбором ответа)

Что из перечисленного является основным преимуществом использования семантических тегов в HTML?

- а) Ускорение загрузки страницы.
- б) Улучшение читаемости кода и доступности для поисковых систем.
- в) Автоматическое применение стилей CSS.
- г) Упрощение работы с JavaScript.

Правильный ответ: б) Улучшение читаемости кода и доступности для поисковых систем.

Связь с компетенцией: ПК-7 — знание основ разработки интерфейса веб-приложения.

Вопрос 2 (Теоретический, с выбором ответа)

Какой HTTP-метод используется для обновления данных в REST API?

- а) GET
- б) POST
- в) PUT
- г) DELETE

Правильный ответ: в) PUT

Связь с компетенцией: ПК-7 — знание основ разработки серверной логики.

Вопрос 3 (Практический, анализ кода)

Дан следующий JavaScript-код:

```
document.getElementById("myButton").addEventListener("click", function() {  
    document.getElementById("myText").style.color = "blue";  
});
```

Что произойдёт при нажатии на элемент с id="myButton"?

Ожидаемый ответ: При нажатии на элемент с id="myButton" цвет текста элемента с id="myText" изменится на синий.

Связь с компетенцией: ПК-7 — разработка интерактивного интерфейса.

Вопрос 4 (Практический, написание кода)

Напишите JavaScript-код, который с помощью Fetch API запрашивает данные с публичного API (например, <https://jsonplaceholder.typicode.com/posts/1>) и выводит полученное значение поля

"title" в консоль.
Ожидаемый ответ:

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then(response => response.json())
  .then(data => console.log(data.title))
  .catch(error => console.error("Ошибка:", error));
```

Связь с компетенцией: ПК-7 — работа с клиентской частью веб-приложения.

Вопрос 5 (Теоретический, с выбором ответа)

Какой из перечисленных инструментов используется для контейнеризации приложений?

- а) Git
- б) Docker
- в) Node.js
- г) Express

Правильный ответ: б) Docker

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — применение современных инструментов разработки (Docker).

Вопрос 6 (Практический, анализ кода)

Дан следующий код сервера на Node.js с использованием Express:

```
const express = require("express");
const app = express();
app.get("/users", (req, res) => {
  res.json([ { id: 1, name: "Анна" } ]);
});
app.listen(3000, () => console.log("Сервер запущен на порту 3000"));
```

Что вернёт сервер при GET-запросе на адрес <http://localhost:3000/users>?

Ожидаемый ответ: Сервер вернёт JSON-объект: [{ "id": 1, "name": "Анна" }]

Связь с компетенцией: ПК-7 — разработка серверной логики.

Вопрос 7 (Теоретический, открытый вопрос)

Назови три основные команды Git, которые используются для работы с репозиторием (инициализация, добавление изменений, фиксация изменений).

Ожидаемый ответ:

- git init — инициализация репозитория.
- git add — добавление изменений в индекс.
- git commit — фиксация изменений.

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — применение современных инструментов разработки (Git).

Вопрос 8 (Практический, написание кода)

Напишите SQL-запрос для получения всех записей из таблицы "users", где возраст (поле age) больше 25.

Ожидаемый ответ:

```
SELECT * FROM users WHERE age > 25;
```

Связь с компетенцией: ПК-7 — работа с базами данных в серверной логике.

Вопрос 9 (Теоретический, с выбором ответа)

Что из перечисленного является примером уязвимости, связанной с безопасностью веб-приложений?

- а) Медленная загрузка страницы.
- б) Cross-Site Scripting (XSS).
- в) Использование устаревшего фреймворка.
- г) Неправильное форматирование CSS.

Правильный ответ: б) Cross-Site Scripting (XSS).

Связь с компетенцией: ПК-7 — знание основ безопасности веб-приложений.

Вопрос 10 (Практический, написание кода)

Напишите Dockerfile для сервера на Node.js, который запускает приложение на порту 3000.

Ожидаемый ответ:

```
FROM node:16
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "index.js"]
```

Связь с компетенцией: ПК-7, ИД 1 ПК-7 — применение современных инструментов разработки (Docker).

3.3. Перечень примерных тем рефератов

1. История развития веб-технологий: от HTML до современных фреймворков

Описание: Исследуйте эволюцию веб-технологий, начиная с первых версий HTML и CSS, до появления JavaScript и современных фреймворков, таких как React и Vue. Опишите, как эти изменения повлияли на разработку веб-приложений.

2. **Сравнительный анализ клиентских фреймворков: React, Vue и Angular**
Описание: Проведите сравнение трёх популярных фреймворков для разработки клиентской части веб-приложений. Оцените их преимущества, недостатки, производительность и области применения.
3. **Роль REST API в современных веб-приложениях**
Описание: Изучите, что такое REST API, его принципы и архитектуру. Опишите, как REST API используется для взаимодействия между клиентской и серверной частями веб-приложений, и приведите примеры.
4. **Использование Git в командной разработке веб-приложений**
Описание: Рассмотрите, как Git помогает в управлении версиями кода при командной разработке. Опишите основные команды, ветвление (branching) и процесс работы с pull request'ами.
5. **Контейнеризация веб-приложений с помощью Docker: преимущества и вызовы**
Описание: Исследуйте, как Docker используется для контейнеризации веб-приложений. Опишите процесс создания Dockerfile, преимущества контейнеризации и возможные сложности при внедрении.
6. **Асинхронное программирование в JavaScript: Promises и async/await**
Описание: Изучите концепцию асинхронного программирования в JavaScript. Сравните использование Promises и async/await, приведите примеры кода и объясните, как это улучшает работу веб-приложений.
7. **Безопасность веб-приложений: основные уязвимости и способы их предотвращения**
Описание: Рассмотрите распространённые уязвимости веб-приложений, такие как XSS (Cross-Site Scripting) и SQL-инъекции. Опишите методы защиты, включая валидацию данных и использование HTTPS.
8. **Сравнение реляционных и нереляционных баз данных в веб-разработке**
Описание: Проведите сравнение реляционных (например, MySQL) и нереляционных (например, MongoDB) баз данных. Оцените их применимость в веб-приложениях, преимущества и недостатки.
9. **Оптимизация производительности веб-приложений: лучшие практики**
Описание: Исследуйте методы оптимизации производительности веб-приложений, такие как минимизация запросов, сжатие данных, Lazy Loading и кэширование. Приведите примеры их применения.

10. Роль Progressive Web Apps (PWA) в будущем веб-разработки

Описание: Изучите концепцию Progressive Web Apps, их преимущества (например, работа оффлайн, push-уведомления) и ограничения. Опишите, как PWA могут изменить подход к разработке веб-приложений.

11. Использование WebSocket для создания интерактивных веб-приложений

Описание: Рассмотрите технологию WebSocket и её применение для создания интерактивных приложений, таких как чаты или онлайн-игры. Сравните WebSocket с традиционными HTTP-запросами.

12. Современные подходы к адаптивной и кроссбраузерной вёрстке

Описание: Изучите методы создания адаптивных веб-страниц, которые корректно отображаются на разных устройствах и браузерах. Опишите использование медиа-запросов, Flexbox и Grid.

13. Роль Node.js в серверной разработке: преимущества и ограничения

Описание: Исследуйте, как Node.js изменил подход к серверной разработке. Опишите его преимущества (например, асинхронность) и ограничения (например, производительность при высоких нагрузках).

14. Автоматизация тестирования веб-приложений: инструменты и подходы

Описание: Рассмотрите инструменты для автоматизации тестирования веб-приложений, такие как Jest, Mocha или Selenium. Опишите, как unit-тесты и интеграционные тесты помогают в разработке.

15. Эволюция JavaScript: от ES5 до ESNext

Описание: Исследуйте развитие языка JavaScript, начиная с ES5, через ES6 (ES2015) и до современных стандартов (ESNext). Опишите ключевые нововведения и их влияние на веб-разработку.

3.4. Кейс-задание

Пример задания

Кейс-задание 1. Разработка веб-приложения "Онлайн-библиотека" с управлением книгами

Ситуация:

Вы — разработчик в небольшой IT-компании, которая получила заказ от местной библиотеки. Клиенту нужно веб-приложение "Онлайн-библиотека", которое позволит пользователям просматривать список книг, добавлять новые книги и отмечать книги как "прочитанные". Прило-

жение должно иметь удобный интерфейс, сервер для хранения данных и возможность управления версиями кода через Git. Для упрощения деплоя клиент просит упаковать серверную часть в Docker-контейнер.

Задача:

1. Разработайте клиентскую часть приложения:
 - Создайте веб-страницу с использованием HTML, CSS и JavaScript.
 - На странице должен быть список книг (с полями: название, автор, статус "прочитано/не прочитано").
 - Добавьте форму для добавления новой книги (название, автор).
 - Реализуйте возможность отмечать книгу как "прочитанную" (например, с помощью кнопки).
2. Разработайте серверную часть:
 - Используйте Node.js и Express для создания сервера.
 - Реализуйте REST API с эндпоинтами:
 - GET /books — получение списка всех книг.
 - POST /books — добавление новой книги.
 - PUT /books/:id — обновление статуса книги (прочитано/не прочитано).
 - Для хранения данных используйте JSON-файл или базу данных (например, SQLite).
3. Интегрируйте клиентскую и серверную части:
 - Используйте Fetch API для отправки запросов с клиента на сервер.
 - Обновляйте список книг на странице после добавления или изменения статуса.
4. Управление версиями:
 - Создайте репозиторий на GitHub.
 - Сделайте минимум 3 коммита: начальная версия, добавление стилей, финальная версия.
5. Контейнеризация:
 - Напишите Dockerfile для серверной части.
 - Соберите Docker-образ и запустите контейнер, убедившись, что сервер доступен.

Кейс-задание 2. Создание чат-приложения с использованием WebSocket

Ситуация:

Вы работаете в стартапе, который разрабатывает платформу для онлайн-обучения. Команда решила добавить в платформу чат для общения между студентами и преподавателями в реальном времени. Вам поручено создать прототип чат-приложения, который будет поддерживать отправку сообщений в реальном времени, сохранять историю сообщений и быть готовым к деплою. Для управления версиями кода нужно использовать Git, а для упрощения деплоя — Docker.

Задача:

1. Разработайте клиентскую часть приложения:
 - Создайте веб-страницу с использованием HTML, CSS и JavaScript.
 - На странице должен быть интерфейс чата: поле для ввода сообщения, кнопка "Отправить" и область для отображения сообщений.
 - Реализуйте подключение к серверу через WebSocket для отправки и получения сообщений в реальном времени.
2. Разработайте серверную часть:
 - Используйте Node.js и библиотеку ws (WebSocket) для создания сервера.

- Реализуйте функционал:
 - При получении сообщения от одного клиента сервер отправляет его всем подключённым клиентам.
 - Сохраняйте историю сообщений в базе данных (например, MongoDB) или JSON-файле.
 - При подключении нового клиента отправляйте ему историю сообщений.
- 3. Интегрируйте клиентскую и серверную части:
 - Убедитесь, что сообщения отправляются и отображаются в реальном времени.
 - При загрузке страницы новый пользователь видит историю сообщений.
- 4. Управление версиями:
 - Создайте репозиторий на GitHub.
 - Сделайте минимум 3 коммита: начальная версия, добавление WebSocket, финальная версия с сохранением сообщений.
- 5. Контейнеризация:
 - Напишите Dockerfile для серверной части.
 - Соберите Docker-образ и запустите контейнер, убедившись, что сервер доступен.

3.5. Примерные практические контрольные задания (ПКЗ)

Пример задания

Задание 1. Создание интерактивной веб-страницы с формой

Описание: Разработайте веб-страницу с использованием HTML, CSS и JavaScript. На странице должна быть форма с полями для ввода имени, email и сообщения, а также кнопка "Отправить". При нажатии на кнопку данные из формы должны отображаться в виде списка под формой без перезагрузки страницы. Добавьте базовую валидацию: поле email должно содержать символ "@", а поле сообщения не должно быть пустым.

Задание 2. Разработка REST API для управления списком задач

Описание: Используя Node.js и Express, создайте сервер с REST API для управления списком задач. API должен поддерживать следующие эндпоинты:

- GET /tasks — получение списка всех задач.
- POST /tasks — добавление новой задачи (с полями: id, title, completed).
- PUT /tasks/:id — обновление статуса задачи (completed: true/false).
- DELETE /tasks/:id — удаление задачи по id.

Для хранения данных используйте массив в памяти (без базы данных).

Критерии оценки:

- Корректная настройка сервера и маршрутов.
- Правильная реализация всех эндпоинтов (GET, POST, PUT, DELETE).
- Обработка ошибок (например, если задача с указанным id не найдена).
- Возвращение корректных HTTP-статусов (200, 201, 404 и т.д.).

Ожидаемые результаты: Работающий сервер, который обрабатывает запросы к API.

Например, GET-запрос возвращает список задач в формате JSON, POST-запрос добавляет новую задачу, а DELETE-запрос удаляет задачу. API можно протестировать с помощью Postman.

Задание 3. Интеграция клиентской и серверной частей: To-Do List

Описание: Используя клиентскую часть (HTML, CSS, JavaScript) и сервер из задания 2, создайте веб-приложение "Список задач". На странице должен быть интерфейс для отображения списка задач, добавления новой задачи и удаления существующей. Данные должны запрашиваться с сервера через Fetch API и обновляться на странице без перезагрузки.

Задание 4. Работа с Git: управление версиями проекта

Описание: Возьмите код из задания 3 и создайте репозиторий на GitHub. Выполните следующие шаги:

- Инициализируйте локальный репозиторий и сделайте первый коммит с начальной версией проекта.
 - Создайте ветку feature/add-style и добавьте стили CSS для улучшения внешнего вида приложения (например, оформите список задач в виде карточек).
 - Зафиксируйте изменения в этой ветке и слейте её в основную ветку (main).
 - Опубликуйте репозиторий на GitHub.
- Критерии оценки:
- Корректная инициализация репозитория и наличие первого коммита.
 - Создание и использование ветки feature/add-style.
 - Наличие минимум 2 коммитов с осмысленными сообщениями (например, "Initial commit", "Add styles for task list").
 - Успешное слияние ветки и публикация на GitHub.

Задание 5. Контейнеризация сервера с помощью Docker

Описание: Возьмите сервер из задания 2 и создайте для него Dockerfile. Выполните следующие шаги:

- Напишите Dockerfile, который устанавливает Node.js, копирует файлы проекта, устанавливает зависимости и запускает сервер.
- Убедитесь, что сервер работает на порту 3000, и укажите это в Dockerfile.
- Соберите Docker-образ и запустите контейнер.
- Проверьте доступность сервера из контейнера, отправив GET-запрос на <http://localhost:3000/tasks>.

Задание 6. Оптимизация и тестирование веб-приложения

Описание: Возьмите веб-приложение из задания 3 и выполните следующие шаги:

- Оптимизируйте загрузку страницы, добавив Lazy Loading для изображений (если их нет, добавьте тестовое изображение).
 - Напишите unit-тест для функции, которая добавляет новую задачу на сервер (например, с использованием Jest).
 - Проверьте работу приложения в двух браузерах (например, Chrome и Firefox) и устраните возможные проблемы с кроссбраузерностью.
- Критерии оценки:
- Реализация Lazy Loading для изображений (например, использование атрибута loading="lazy").
 - Корректность unit-теста (проверка успешного добавления задачи и обработки ошибок).
 - Отсутствие ошибок при тестировании в разных браузерах.
 - Читаемость и структурированность кода.

3.6. Примерные вопросы к зачету

2. **Что такое семантические теги в HTML, и почему их использование важно при разработке веб-страниц? Приведите примеры таких тегов.**
Описание: Вопрос проверяет знание основ HTML и понимание важности семантики для доступности и SEO.
3. **Опишите, как работает модель DOM (Document Object Model). Как JavaScript может взаимодействовать с DOM для изменения содержимого страницы? Приведите пример кода.**
Описание: Вопрос направлен на проверку понимания работы с DOM и базовых навыков JavaScript.
4. **В чём разница между синхронным и асинхронным кодом в JavaScript? Объясните, как работают Promises и async/await, и приведите пример асинхронного запроса данных.**
Описание: Вопрос проверяет знание асинхронного программирования, важного для работы с API и внешними данными.
5. **Что такое REST API? Опишите основные HTTP-методы, используемые в REST, и для чего они применяются.**
Описание: Вопрос направлен на проверку понимания архитектуры REST и её роли в веб-разработке.
6. **Как Node.js используется для серверной разработки? В чём его преимущества по сравнению с традиционными серверными технологиями, такими как PHP?**
Описание: Вопрос проверяет знание серверной разработки и понимание особенностей Node.js.
7. **Опишите процесс подключения базы данных к серверу на Node.js. Какие шаги нужно выполнить, чтобы реализовать CRUD-операции (создание, чтение, обновление, удаление)?**
Описание: Вопрос направлен на проверку практических навыков работы с базами данных..
8. **Что такое Git, и как он используется в разработке веб-приложений? Назови основные команды для работы с репозиторием и опиши процесс создания ветки и слияния изменений.**

Описание: Вопрос проверяет знание инструмента Git и его применения в командной разработке.

9. **Объясните, что такое Docker и как он помогает в разработке веб-приложений. Опишите, как создать Dockerfile для сервера на Node.js.**

Описание: Вопрос направлен на проверку понимания контейнеризации и её применения.

10. **Какие существуют уязвимости веб-приложений, связанные с безопасностью? Опишите, как можно защитить приложение от XSS (Cross-Site Scripting) атак.**

Описание: Вопрос проверяет знание основ безопасности веб-приложений.

11. **Что такое адаптивная вёрстка, и какие технологии используются для её реализации? Приведите пример медиа-запроса для изменения стилей на мобильных устройствах.**

Описание: Вопрос направлен на проверку навыков создания адаптивных интерфейсов.

12. **Как можно оптимизировать производительность веб-приложения? Назови не менее трёх методов и объясни, как они работают.**

Описание: Вопрос проверяет понимание методов оптимизации, важных для создания эффективных приложений.

13. **Опишите процесс интеграции клиентской и серверной частей веб-приложения. Как клиент может отправить запрос к серверу и обработать полученные данные? Приведите пример кода.**

Описание: Вопрос направлен на проверку практических навыков интеграции клиент-серверного взаимодействия.

4. Методические материалы, определяющие процедуры оценивания достижения запланированных результатов обучения по дисциплине

Опрос устный

Опрос устный - диалог преподавателя со студентом, цель которого - систематизация и уточнение имеющихся у студента знаний, проверка его индивидуальных возможностей усвоения материала.

Устный опрос по основным терминам может проводиться в начале/конце лекционного или практического занятия в течение 15 -20 мин. Либо устный опрос проводится в течение всего практического занятия по заранее выданной тематике. Выбранный преподавателем студент может отвечать с места либо у доски.

Критериями оценки устного опроса являются: правильность ответа на вопросы, степень раскрытия сущности вопроса.

Оценка «отлично» — дан полный, всесторонний ответ на вопрос. Точность в определениях. Приведение примеров из практики.

Оценка «хорошо» — дан неполный ответ на вопрос. Допущены неточности при ответе. Допущены неточности в основных определениях.

Оценка «удовлетворительно» — имеются существенные недочеты при ответе. Вопрос раскрыт частично. Незнание базовых определений курса.

Оценка «неудовлетворительно» — вопрос не раскрыт или дан неверный ответ.

Тесты

Тесты - инструмент, с помощью которого педагог оценивает степень достижения студентом требуемых знаний, умений, навыков. Составление теста включает в себя создание выверенной системы вопросов, собственно процедуру проведения тестирования и способ измерения полученных результатов.

Критерии оценки теста: Оценка «отлично» выставляется при условии правильного ответа студента не менее чем 85 % тестовых заданий;

Оценка «хорошо» выставляется при условии правильного ответа студента не менее чем 70 % тестовых заданий;

Оценка «удовлетворительно» выставляется при условии правильного ответа студента не менее 51 %.

Оценка «неудовлетворительно» выставляется при условии правильного ответа студента менее чем на 50 % тестовых заданий

Кейс - задания

Кейс - задания - проблемное задание, в котором обучающемуся предлагают осмыслить реальную профессионально-ориентированную ситуацию, необходимую для решения данной проблемы. Студент самостоятельно формулирует цель, находит и собирает информацию, анализирует ее, выдвигает гипотезы, ищет варианты

решения проблемы, формулирует выводы, обосновывает оптимальное решение ситуации.

Критерии оценки кейс-заданий: Отметка «отлично» — задание выполнено в полном объеме с соблюдением необходимой последовательности действий; в ответе правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ ошибок. Отметка «хорошо» — задание выполнено правильно с учетом 1-2 мелких погрешностей или 2-3 недочетов, исправленных самостоятельно по требованию преподавателя. Отметка «удовлетворительно» — задание выполнено правильно не менее чем наполовину, допущены 1-2 погрешности или одна грубая ошибка.

Отметка «неудовлетворительно» — допущены две (и более) грубые ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя или задание не решено полностью.

Реферат

Критериями оценки реферата являются: новизна текста, обоснованность выбора источников литературы, степень раскрытия сущности вопроса, соблюдения требований к оформлению.

Оценка «отлично» — выполнены все требования к написанию реферата: обозначена проблема и обоснована её актуальность; сделан анализ различных точек зрения на рассматриваемую проблему и логично изложена собственная позиция; сформулированы выводы, тема раскрыта полностью, выдержан объём; соблюдены требования к внешнему оформлению.

Оценка «хорошо» — основные требования к реферату выполнены, но при этом допущены недочёты. В частности, имеются неточности в изложении материала; отсутствует логическая последовательность в суждениях; не выдержан объём реферата; имеются упущения в оформлении.

Оценка «удовлетворительно» — имеются существенные отступления от требований к реферированию. В частности: тема освещена лишь частично; допущены фактические ошибки в содержании реферата; отсутствуют выводы.

Оценка «неудовлетворительно» — тема реферата не раскрыта, обнаруживается существенное непонимание проблемы или реферат не представлен вовсе.

Практические контрольные задания (ПКЗ)

Критерии оценки практических контрольных заданий: Результат выполнения КР оценивается в баллах: "5" -отлично, "4" -хорошо, "3" -удовлетворительно, "2" -неудовлетворительно. Отметка «5» ставится, если:

- работа выполнена полностью;
- в решении нет математических ошибок (возможен один недочёт, описка, которая не является следствием незнания или непонимания учебного материала).

Отметка «4» ставится в следующих случаях:

- работа выполнена полностью, но допущены одна ошибка или есть два - три недочёта в выкладках решения;

Отметка «3» ставится, если:

- допущены две-три ошибки в вычислениях, при этом должно быть выполнено не менее 60% всей работы.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не обладает обязательными умениями по данной теме в полной мере, при этом выполнено менее 60%.

Контрольная работа

Контрольная работа - средство промежуточного контроля остаточных знаний и умений, состоит из вопросов или заданий, которые студент должен решить, выполнить. Знакомство с основной и дополнительной литературой, включая справочные издания, зарубежные источники, конспект основных положений, терминов, сведений, требующих для запоминания и являющихся основополагающими в этой теме.

Критерии оценки контрольной работы для студентов заочного отделения:

Оценка «зачтено» ставится за полные ответы на все вопросы.

Оценка «не зачтено» ставится, если освещены не все вопросы требуемого материала или не описано главное в содержании вопросов, или письменная работа не сдана.

Коллоквиум

Коллоквиум (в переводе с латинского «беседа, разговор») – форма текущего контроля знаний студентов, которая проводится в виде собеседования преподавателя и студента по самостоятельно подготовленной студентом теме.

Он применяется для проверки знаний по определенному разделу (или объемной теме) и принятия решения о том, можно ли переходить к изучению нового материала. Коллоквиум — это беседа со студентами, целью которой является выявление уровня овладения новыми знаниями. В отличие от семинара главное на коллоквиуме — это проверка знаний с целью их систематизации.

Целью коллоквиума является формирование у студента навыков анализа теоретических проблем на основе самостоятельного изучения учебной и научной литературы.

На коллоквиум выносятся крупные, проблемные, нередко спорные теоретические вопросы. Коллоквиум может проводиться по вопросам, обсуждавшимся на семинарах. Конкретные вопросы для коллоквиума студентам не сообщаются, однако заранее формулируются преподавателем. Предполагаемый объем ответа не должен быть большим (примерно 1,5-2 минуты), чтобы преподаватель мог успеть опросить всех студентов.

От студента требуется:

- владение изученным в ходе учебного процесса материалом, относящимся к рассматриваемой проблеме;
- наличие собственного мнения по обсуждаемым вопросам и умение его аргументировать.

Коллоквиум — это не только форма контроля, но и метод углубления, закрепления знаний студентов, так как в ходе собеседования преподаватель разъясняет сложные вопросы, возникающие у студента в процессе изучения данного источника.

Задача коллоквиума добиться глубокого изучения отобранного материала, пробудить у студента стремление к чтению дополнительной экономической литературы.

Подготовка к проведению коллоквиума.

Подготовка к коллоквиуму предполагает несколько этапов:

1. Подготовка к коллоквиуму начинается с установочной консультации преподавателя, на которой он разъясняет развернутую тематику проблемы, рекомендует литературу для изучения и объясняет процедуру проведения коллоквиума.
2. Как правило, на самостоятельную подготовку к коллоквиуму студенту отводится 3–4 недели. Подготовка включает в себя изучение рекомендованной литературы и (по указанию преподавателя) конспектирование важнейших источников.
3. Коллоквиум проводится в форме индивидуальной беседы преподавателя с каждым студентом или беседы в небольших группах (3–5 человек).
4. Преподаватель задает несколько кратких конкретных вопросов, позволяющих выяснить степень добросовестности работы с литературой, контролирует конспект. Далее более подробно обсуждается какая-либо сторона проблемы, что позволяет оценить уровень понимания.
5. По итогам коллоквиума выставляется дифференцированная оценка, имеющая большой удельный вес в определении текущей успеваемости студента.

Особенности и порядок сдачи коллоквиума. Студент может себя считать готовым к сдаче коллоквиума по избранной работе, когда у него есть им лично составленный и обработанный конспект сдаваемой работы, он знает структуру работы в целом, содержание работы в целом или отдельных ее разделов (глав); умеет раскрыть рассматриваемые проблемы и высказать свое отношение к прочитанному и свои сомнения, а также знает, как убедить преподавателя в правоте своих суждений.

Проведение коллоквиума позволяет студенту приобрести опыт работы над первоисточниками, что в дальнейшем поможет с меньшими затратами времени работать над литературой по курсовой работе и при подготовке к экзаменам.