Федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный аграрный университет имени П.А. Столыпина»

Университетский колледж агробизнеса

по специальности
09.02.11 Разработка и управление программным обеспечением

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТОВ
ПМ.04 Проектирование и разработка веб-приложений
Специальность**: 09.02.11 Разработка и управление программным обеспечением**

Обеспечивающее преподавание отделение		
Разработчик:		
Преподаватель		*А.В.Кортусов*

Омск
2026

Пояснительная записка

Методические рекомендации предназначены для выполнения самостоятельной работы обучающимися по специальности 09.02.11 Разработка и управление программным обеспечением.

Самостоятельная работа выполняется по заданию и при методическом руководстве преподавателя, но без его непосредственного участия. Целью самостоятельной работы является овладение обучающимся умениями работать с источниками, обобщения и анализа практики, аргументации собственной точки зрения.

Методические рекомендации по самостоятельной работе студентов содержат материалы для подготовки к лекционным, практическим занятиям, к формам текущего и промежуточного контроля. Наряду с методическими рекомендациями по подготовке и написанию курсовых работ, защите квалификационных работ составляют единый комплекс методического обеспечения студента по профессиональному модулю. Предложенные в рекомендациях задания позволят успешно овладеть профессиональными знаниями, умениями и навыками, и направлены на формирование общих и профессиональных компетенций:

ОК. 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста

ОК.6 Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных российских духовно-нравственных ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения

ПК 4.1. Проектировать базы данных

ПК 4.2. Разрабатывать объекты баз данных в соответствии с результатами анализа предметной области

ПК 4.3. Осуществлять техническое сопровождение и восстановление веб-приложений в соответствии с техническим заданием.

ПК 4.4. Производить тестирование разработанного веб-приложения

ПК 4.5. Осуществлять аудит безопасности веб-приложения в соответствии с регламентом безопасности

ПК 4.6. Модернизировать веб-приложения с учетом правил и норм подготовки информации для поисковых систем

ПК 4.7. Реализовывать мероприятия по продвижению приложения

При выполнении самостоятельной работы обучающийся самостоятельно осуществляет сбор, изучение, систематизацию и анализ информации, а затем оформляет информацию и представляет на оценку преподавателя или группы.

Виды самостоятельной работы

№ п/п	Вид самостоятельной работы	Форма контроля	Максимальное кол-во баллов
1.	Работа с источниками	Устный ответ на занятии Составление аннотации	5
2.	Составление опорного конспекта	Опорный конспект	5
3.	Составление сравнительной таблицы	Сравнительная таблица	5
4.	Решение ситуационных задач	Письменный ответ	5
5.	Анализ практических работ	Письменный отчет	5
6.	Участие в научно-исследовательской деятельности*	Выступление на конференции	5

Методические рекомендации по работе с источниками

Работа с источниками осуществляется с целью приобретения обучающимся навыков самостоятельного изучения учебного материала. Работа с источниками является важной составляющей при подготовке к занятиям.

Для подготовки к устному опросу необходимо прочитать текст источника, выделить главное, составить план ответа, повторить текст несколько раз. На учебном занятии полно, точно, доступно, правильно, взаимосвязано и логично изложить материал, иллюстрируя при необходимости примерами.

Работа с источником может быть предложена в форме аннотирования. Аннотация позволяет составить обобщенное представление об источнике. Для составления аннотации необходимо ответить на следующие вопросы:

1. Фамилия автора, полное наименование работы, место и год издания.
2. Вид издания (статья, учебник, и пр.).
3. Цели и задачи издания.
4. Структура издания и краткий обзор содержания работы.
5. Основные проблемы, затронутые автором.
6. Выводы и предложения автора по решению выделенных проблем.

Источник аннотирования определяет преподаватель, он же оценивает аннотацию, сданную в письменной форме.

Методические рекомендации по анализу практических кейсов (в сфере разработки программных модулей)

Анализ практических кейсов проводится с целью изучения современных подходов к проектированию и разработке веб-приложений, выявления типичных проблем и способов их решения. Подборка кейсов осуществляется обучающимся самостоятельно на основании данных из профессиональной литературы, проектной документации, открытых источников и реальных проектных ситуаций. В подборку входит от 5 до 10 кейсов.

Анализ практических кейсов проходит в два этапа:

I. Составление краткой характеристики кейса, в которой излагаются:

- Название/тематика кейса.
- Описание проблемной ситуации (ошибки проектирования, проблемы с производительностью, уязвимости безопасности, сложности интеграции, низкая доступность).
- Участники ситуации (разработчик, фронтенд-разработчик, бэкенд-разработчик, веб-дизайнер, заказчик, пользователь).
- Исходные данные и условия (архитектурные решения, стек технологий, требования).
- Предпринятые действия и их результат.

Характеристика составляется на основе описания кейса с сохранением профессиональной терминологии.

II. Анализ составленных характеристик путём ответов на следующие вопросы:

1. Какие типы проблем в вашей подборке являются типичными для разработки веб-приложений?
2. В каких типах проектов обычно возникают такие проблемы?
3. Кто обычно является инициатором решения проблемы, а кто – ответственным за устранение?
4. Какие методы и инструменты веб-разработки применялись для решения каждой типичной проблемы?
5. Каков итог решения каждой проблемы (успешное завершение, переработка модуля, изменение архитектуры)?

Задание оформляется в письменной форме.

Методические рекомендации по составлению опорного конспекта (по темам ПМ 04)

Опорный конспект составляется с целью обобщения, систематизации и краткого изложения информации по темам, связанным с проектированием и разработкой веб-приложений, веб-технологиями, инструментами и подходами.

Составление опорного конспекта включает следующие действия:

1. Изучение текста учебного материала (лекции, учебники, веб-документация).
2. Определение главного и второстепенного в анализируемом тексте.
3. Установление логической последовательности между элементами (этапы разработки веб-приложений, архитектура, стек технологий).
4. Составление характеристики элементов учебного материала в краткой форме.
5. Выбор опорных сигналов для расстановки акцентов (ключевые термины: фронтенд, бэкенд, API, SPA, SSR, PWA).

6. Оформление опорного конспекта.

Опорный конспект может быть представлен в виде:

- схемы с использованием стрелок для определения связи между элементами (например, архитектура веб-приложения: клиент → сервер → БД);
- системы геометрических фигур (блок-схема работы веб-приложения);
- логической лестницы (уровни веб-архитектуры);
- интеллект-карты (классификация веб-технологий).

Оценкой опорного конспекта служит качество ответа как самого студента, так и других студентов, его использовавших. Преподаватель также может проверить опорные конспекты, сданные в письменной форме. Допускается проведение конкурса на самый лучший конспект по критериям: краткость формы, логичность изложения, наглядность выполнения, универсальность содержания.

Методические рекомендации по составлению сравнительной таблицы (по веб-технологиям и подходам)

Сравнительная таблица составляется с целью выявления сходств, отличий, преимуществ и недостатков анализируемых объектов (веб-фреймворки, подходы к рендерингу, технологии API, типы веб-архитектур и т.д.).

Критерии для составления сравнительной таблицы предлагает преподаватель. Студент, самостоятельно сформулировавший критерии для сравнения, получает дополнительные баллы.

Примеры критериев для сравнения веб-фреймворков:

- Тип фреймворка (фронтенд / бэкенд / полного цикла).
- Производительность.
- Сложность освоения.
- Наличие экосистемы и сообщества.
- Поддержка и документация.
- Применяемость для различных типов веб-приложений.
- Скорость разработки.

Проверка и оценка сравнительной таблицы осуществляется преподавателем в письменной форме.

Методические рекомендации по решению ситуационных задач (по разработке веб-приложений)

Ситуационные задачи решаются с целью приобретения обучающимся навыков самостоятельной работы с веб-технологиями, обобщения и анализа практических ситуаций в области разработки веб-приложений, а также умений аргументировать собственную точку зрения и принимать технические решения.

При решении задач студентам рекомендуется следующая схема:

1. Проанализировать приведённую в задаче ситуацию и поставленный вопрос.
2. Определить соответствующую технологию, инструмент или подход, применимые в данной ситуации.
3. Найти в методических материалах, документации или литературе необходимые рекомендации и дать их интерпретацию применительно к ситуации.

4. Составить в письменной форме мотивированный вывод по задаче с обоснованием принятого решения.

Объём задания определяет преподаватель.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ (ПМ 04)

Самостоятельная работа №1

Тема: «Основы проектирования и разработки веб-приложений»

Задание. Подготовиться к устному опросу, ответив на следующие вопросы:

1. Дайте определение веб-приложения.
2. Назовите основные виды веб-приложений.
3. В чём отличие веб-приложения от веб-сайта?
4. Назовите основные этапы разработки веб-приложения.
5. Какие роли существуют в команде разработки веб-приложений?
6. Что такое фронтенд и бэкенд веб-приложения?
7. Назовите основные технологии фронтенд-разработки.
8. Назовите основные технологии бэкенд-разработки.
9. Что такое клиент-серверная архитектура веб-приложений?
10. Что такое протокол HTTP и как он используется в веб-приложениях?
11. Что такое API и какова его роль в веб-приложениях?
12. Какие требования предъявляются к современным веб-приложениям?
13. Что такое адаптивный веб-дизайн?
14. Назовите основные этапы жизненного цикла веб-приложения.
15. Каковы современные тренды в разработке веб-приложений?

Самостоятельная работа №2

Тема: «Архитектура веб-приложений»

Задание. Проанализируйте архитектурные подходы к разработке веб-приложений и составьте таблицу, разделив их на следующие группы:

1. MPA (Multi-Page Application) – многопоточные приложения.
2. SPA (Single-Page Application) – одностраничные приложения.
3. SSR (Server-Side Rendering) – серверный рендеринг.
4. SSG (Static Site Generation) – статическая генерация.

Для каждого подхода укажите:

- описание и характеристики;
- преимущества и недостатки;
- сферы применения;
- примеры технологий и инструментов.

Самостоятельная работа №3

Тема: «Технологии фронтенд-разработки»

Задание. Подготовиться к устному опросу, ответив на следующие вопросы:

1. Что такое HTML и какова его роль в разработке веб-приложений?

2. Что такое CSS и какова его роль в разработке веб-приложений?
3. Что такое JavaScript и какова его роль в разработке веб-приложений?
4. Что такое DOM (Document Object Model)?
5. Что такое семантическая вёрстка?
6. Что такое адаптивная вёрстка (Responsive Web Design)?
7. Какие CSS-фреймворки вы знаете (Bootstrap, Tailwind, Foundation)?
8. Какие JavaScript-фреймворки и библиотеки вы знаете (React, Angular, Vue.js)?
9. Что такое TypeScript и в чём его преимущества?
10. Что такое Webpack и для чего он используется?
11. Что такое Babel и для чего он используется?
12. Что такое npm и как он используется в веб-разработке?
13. Что такое PWA (Progressive Web App)?
14. Какие требования предъявляются к фронтенд-разработке?
15. Как осуществляется отладка и тестирование фронтенда?

Самостоятельная работа №4

Тема: «Разработка клиентской части веб-приложений на React»

Задание. Подготовиться к тестированию, ответив на следующие вопросы:

1. Что такое React и каковы его основные принципы?
2. Что такое компонент в React?
3. Что такое JSX и как он используется?
4. В чём отличие классовых и функциональных компонентов?
5. Что такое props и state в React?
6. Что такое хуки (Hooks) в React?
7. Что такое useState и useEffect?
8. Что такое контекст (Context API) в React?
9. Что такое React Router и как он используется?
10. Что такое управление состоянием в React (Redux, Zustand, MobX)?
11. Как осуществляется работа с формами в React?
12. Что такое React DevTools и как они используются?
13. Как осуществляется оптимизация производительности React-приложений?
14. Как осуществляется тестирование React-компонентов?
15. Как развернуть React-приложение в продакшн?

Самостоятельная работа №5

Тема: «Разработка клиентской части веб-приложений на Angular»

Задание. Подготовиться к устному опросу, ответив на следующие вопросы:

1. Что такое Angular и каковы его основные принципы?
2. Что такое компонент в Angular?

3. Что такое модуль в Angular?
4. Что такое шаблоны (templates) в Angular?
5. Что такое директива в Angular?
6. Что такое сервис в Angular?
7. Что такое Dependency Injection (внедрение зависимостей) в Angular?
8. Что такое маршрутизация (Routing) в Angular?
9. Что такое RxJS и как он используется в Angular?
10. Что такое формы в Angular (Template-Driven Forms, Reactive Forms)?
11. Что такое HTTP-клиент в Angular?
12. Что такое Angular CLI и как он используется?
13. Как осуществляется тестирование Angular-приложений?
14. Как осуществляется оптимизация производительности Angular-приложений?
15. Как развернуть Angular-приложение в продакшн?

Самостоятельная работа №6

Тема: «Технологии бэкенд-разработки веб-приложений»

Задание. Охарактеризуйте основные технологии бэкенд-разработки с точки зрения применяемых языков программирования, фреймворков и сфер применения.

Задание №2. Подготовиться к устному опросу, ответив на следующие вопросы:

1. Что такое бэкенд веб-приложения и какова его роль?
2. Какие языки программирования используются для бэкенд-разработки?
3. Что такое Node.js и какова его роль в бэкенд-разработке?
4. Что такое Express.js и как он используется?
5. Что такое Python (Django, Flask) для бэкенд-разработки?
6. Что такое Java (Spring Boot) для бэкенд-разработки?
7. Что такое PHP (Laravel, Symfony) для бэкенд-разработки?
8. Что такое C# (ASP.NET Core) для бэкенд-разработки?
9. Что такое REST API и как он разрабатывается?
10. Что такое GraphQL и чем он отличается от REST?
11. Что такое аутентификация и авторизация в веб-приложениях?
12. Какие механизмы аутентификации существуют (JWT, OAuth, Session)?
13. Что такое веб-сокеты (WebSocket) и для чего они используются?
14. Как осуществляется взаимодействие бэкенда с базой данных?
15. Как осуществляется тестирование бэкенда?

Самостоятельная работа №7

Тема: «Базы данных в веб-приложениях»

Задание. Подготовиться к тестированию, ответив на следующие вопросы:

1. Какова роль базы данных в веб-приложении?

2. Какие типы баз данных используются в веб-приложениях?
3. Что такое реляционные базы данных и когда они применяются?
4. Что такое NoSQL-базы данных и когда они применяются?
5. Что такое ORM (Object-Relational Mapping) и как он используется?
6. Что такое SQL-инъекция и как от неё защититься?
7. Какие СУБД наиболее часто используются в веб-приложениях (PostgreSQL, MySQL, MongoDB)?
8. Как осуществляется миграция базы данных при разработке веб-приложения?
9. Что такое кэширование данных в веб-приложениях?
10. Какие инструменты для работы с БД в веб-приложениях вы знаете?
11. Как осуществляется оптимизация запросов к базе данных?
12. Что такое транзакции в веб-приложениях?
13. Как осуществляется резервное копирование базы данных веб-приложения?
14. Какие требования к безопасности базы данных в веб-приложениях?
15. Как выбрать СУБД для конкретного веб-приложения?

Самостоятельная работа №8

Тема: «Разработка веб-приложений с использованием Node.js и Express»

Задание №1. Подготовиться к устному опросу, ответив на следующие вопросы:

1. Что такое Node.js и каковы его особенности?
2. Что такое Express.js и какова его роль в Node.js-разработке?
3. Как организована работа с маршрутами (routing) в Express?
4. Что такое middleware в Express и как он используется?
5. Как осуществляется обработка запросов (Request) и ответов (Response) в Express?
6. Как осуществляется работа с шаблонизаторами в Express (EJS, Pug, Handlebars)?
7. Как осуществляется работа с базой данных в Node.js-приложениях (Mongoose, Sequelize)?
8. Как осуществляется аутентификация в Node.js-приложениях (JWT, Passport.js)?
9. Как осуществляется валидация данных в Node.js-приложениях?
10. Как осуществляется обработка ошибок в Node.js-приложениях?
11. Как осуществляется логгирование в Node.js-приложениях?
12. Как осуществляется тестирование Node.js-приложений (Jest, Mocha)?
13. Как осуществляется развёртывание Node.js-приложений?
14. Как осуществляется мониторинг производительности Node.js-приложений?
15. Каковы преимущества и недостатки использования Node.js для разработки веб-приложений?

Задание №2. Составьте опорный конспект, отразив в нём архитектуру веб-приложения на Node.js и Express.

Самостоятельная работа №9

Тема: «Безопасность веб-приложений»

Задание. Решить ситуационные задачи.

Задача 1. В процессе разработки веб-приложения обнаружена уязвимость: при вводе специальных символов в поле поиска данные из базы начинают отображаться некорректно. Определите тип уязвимости и предложите способы её устранения.

Задача 2. При аудите безопасности веб-приложения выявлено, что сессионные cookie передаются без флага Secure и HttpOnly. Какие риски это создаёт и как исправить ситуацию?

Задача 3. Веб-приложение интернет-магазина стало медленно работать при большом количестве пользователей. При анализе выяснилось, что страницы загружают большие объёмы данных без кэширования. Предложите меры по повышению производительности.

Задача 4. В веб-приложении отсутствует защита от CSRF-атак. Какую угрозу это создаёт и как реализовать защиту от CSRF?

Задача 5. При разработке веб-приложения для работы с персональными данными необходимо обеспечить соответствие требованиям ФЗ-152. Какие меры безопасности должны быть реализованы в веб-приложении?

Самостоятельная работа №10

Тема: «Анализ практических кейсов разработки веб-приложений»

Задание. Проанализировать практические кейсы по разработке веб-приложений. Для анализа сделать подборку пяти типичных кейсов с описанием:

- предметной области и задачи веб-приложения;
- применённых технологий и архитектурных решений;
- основных этапов разработки;
- достигнутых результатов и проблем, возникших в процессе.

Самостоятельная работа №11

Тема: «Документирование веб-приложений»

Задание. Подготовиться к устному опросу, ответив на следующие вопросы:

1. Какова роль документации в разработке веб-приложений?
2. Назовите виды документации на веб-приложения.
3. Что такое техническое задание на разработку веб-приложения?
4. Что такое проектная документация на веб-приложение?
5. Что такое API-документация и как она разрабатывается (Swagger/OpenAPI)?
6. Что такое пользовательская документация на веб-приложение?
7. Что такое руководство администратора веб-приложения?
8. Что такое README-файл и как его оформлять?
9. Какие инструменты используются для документирования веб-приложений?
10. Какие требования предъявляются к качеству документации?
11. Как осуществляется актуализация документации в процессе разработки?
12. Что такое Swagger UI и как он используется?
13. Как документировать компоненты веб-приложения?
14. Какие стандарты документирования веб-приложений существуют?

15. Какова ответственность разработчика за качество документации?

Самостоятельная работа №12

Тема: «Анализ практических кейсов по развёртыванию и вводу в эксплуатацию веб-приложений»

Задание. Проанализировать практические кейсы по развёртыванию и вводу в эксплуатацию веб-приложений. Для анализа сделать подборку пяти типичных ситуаций, описав:

- этапы развёртывания;
- использованные инструменты и технологии (Docker, CI/CD, облачные платформы);
- возникшие проблемы и способы их решения;
- итоговые результаты.

Самостоятельная работа №13

Задание. Подготовиться к дифференцированному зачёту по ПМ 04 «Проектирование и разработка веб-приложений», письменно ответив на следующие вопросы:

Раздел 1. Основы проектирования веб-приложений

1. Понятие веб-приложения. Отличие от веб-сайта.
2. Классификация веб-приложений.
3. Основы архитектуры веб-приложений.
4. Клиент-серверная архитектура веб-приложений.
5. MPA vs SPA: сравнительный анализ.
6. SSR (Server-Side Rendering) и SSG (Static Site Generation).
7. Протокол HTTP: методы, статусы, заголовки.
8. Жизненный цикл разработки веб-приложения.
9. Роли в команде разработки веб-приложений.
10. Основные требования к современным веб-приложениям.

Раздел 2. Фронтенд-разработка

11. Основы HTML: структура документа, семантическая вёрстка.
12. Основы CSS: каскадность, селекторы, Flexbox, Grid.
13. Адаптивная и отзывчивая вёрстка (Responsive Web Design).
14. Основы JavaScript: синтаксис, DOM, события, асинхронность.
15. CSS-фреймворки (Bootstrap, Tailwind).
16. Фреймворки и библиотеки JavaScript (React, Angular, Vue.js).
17. React: компоненты, JSX, state, props, хуки.
18. React: управление состоянием (Redux, Context API).
19. React: маршрутизация (React Router).
20. Angular: компоненты, модули, Dependency Injection.
21. Angular: маршрутизация, формы, HTTP-клиент.
22. TypeScript: понятие, преимущества, применение.
23. Сборка и оптимизация фронтенда (Webpack, Vite).

24. Тестирование фронтенда (Jest, React Testing Library).
25. Развёртывание фронтенд-приложений.

Раздел 3. Бэкенд-разработка

26. Основы бэкенд-разработки: архитектура, компоненты.
27. Языки программирования для бэкенда (JavaScript/Node.js, Python, PHP, Java, C#).
28. Node.js: особенности, асинхронность, Event Loop.
29. Express.js: маршрутизация, middleware, обработка запросов.
30. Django (Python): архитектура, ORM, админ-панель.
31. Spring Boot (Java): архитектура, Dependency Injection, REST.
32. Laravel (PHP): архитектура, ORM, маршрутизация.
33. Разработка REST API: принципы, методы, статусы.
34. GraphQL: преимущества, отличия от REST.
35. Аутентификация и авторизация: JWT, OAuth 2.0, Session.
36. WebSocket: назначение, применение в веб-приложениях.
37. Работа с базами данных в бэкенде (ORM, SQL).
38. Валидация данных и обработка ошибок.
39. Логгирование и мониторинг бэкенда.
40. Тестирование бэкенда (Unit, Integration, E2E).

Раздел 4. Базы данных в веб-приложениях

41. Базы данных: роль в веб-приложениях.
42. Реляционные базы данных (PostgreSQL, MySQL).
43. NoSQL-базы данных (MongoDB, Redis).
44. Проектирование схемы базы данных для веб-приложения.
45. Миграции базы данных.
46. SQL-запросы: SELECT, JOIN, INSERT, UPDATE, DELETE.
47. Оптимизация запросов и индексы.
48. ORM (Sequelize, TypeORM, SQLAlchemy, Hibernate).
49. Транзакции в веб-приложениях.
50. Кэширование данных (Redis, Memcached).

Раздел 5. Безопасность веб-приложений

51. Основные угрозы безопасности веб-приложений (OWASP Top 10).
52. SQL-инъекции: принцип и защита.
53. XSS (межсайтовый скриптинг): принцип и защита.
54. CSRF-атаки: принцип и защита.
55. Защита аутентификации и сессий.
56. HTTPS и SSL/TLS: назначение, настройка.
57. Безопасное хранение паролей (хэширование, соль).
58. CORS: понятие, настройка.
59. Защита API: rate limiting, валидация.

60. Безопасность в соответствии с ФЗ-152 при разработке веб-приложений.

Раздел 6. Инструменты и технологии веб-разработки

61. Системы контроля версий (Git) в веб-разработке.
62. Менеджеры пакетов (npm, yarn).
63. Сборщики модулей (Webpack, Vite).
64. Инструменты разработки (DevTools, VS Code).
65. Контейнеризация и Docker в веб-разработке.
66. CI/CD для веб-приложений.
67. Облачные платформы (AWS, Yandex Cloud, Heroku, Vercel).
68. Мониторинг и аналитика веб-приложений.
69. Документирование API (Swagger/OpenAPI).
70. Работа с веб-серверами (Nginx, Apache).

Раздел 7. Тестирование и качество веб-приложений

71. Виды тестирования веб-приложений.
72. Модульное тестирование.
73. Интеграционное тестирование.
74. E2E-тестирование (Cypress, Selenium).
75. Нагрузочное тестирование.
76. Тестирование безопасности.
77. Автоматизация тестирования веб-приложений.
78. Код-ревью в веб-разработке.
79. Метрики качества веб-приложений.
80. Производительность веб-приложений и способы её повышения.

Раздел 8. Развёртывание и эксплуатация

81. Подготовка веб-приложения к релизу.
82. Настройка серверного окружения.
83. Работа с веб-серверами (Nginx, Apache).
84. Развёртывание на облачных платформах.
85. Continuous Deployment для веб-приложений.
86. Мониторинг работы веб-приложения.
87. Обработка ошибок и логирование в эксплуатации.
88. Масштабирование веб-приложений.
89. Обновление и сопровождение веб-приложений.
90. Планирование развития веб-приложения.

Критерии оценки внеаудиторной (самостоятельной) работы

Процент результативности	Балл (оценка)	Критерии оценивания
90-100%	5	1. глубокое изучение учебного материала, литературы и нормативных актов по вопросу; 2. правильность формулировок, точность определения понятий; 3. последовательность изложения материала; 4. обоснованность и аргументированность выводов; 5. правильность ответов на дополнительные вопросы; 6. своевременность выполнения задания.
70-89%	4	11. полнота и правильность изложения материала; 12. незначительные нарушения последовательности изложения; 13. неточности в определении понятий; 14. обоснованность выводов приводимыми примерами; 15. правильность ответов на дополнительные вопросы; 16. своевременность выполнения задания.
50-69%	3	8. знание и понимание основных положений учебного материала; 9. наличие ошибок при изложении материала; 10. непоследовательность изложения материала; 11. наличие ошибок в определении понятий, неискажающих их смысл; 12. несвоевременность выполнения задания.
0-49%	2	8. незнание, невыполнение или неправильное выполнение большей части учебного материала; 9. ошибки в формулировке определений, искажающие их смысл; 10. беспорядочное и неуверенное изложение материала; 11. отсутствие ответов на дополнительные вопросы; 12. отсутствие выводов и неспособность их сформулировать; 13. невыполнение задания.