Компонент ОПОП **09.03.01 Информатика и вычислительная техника,**
направленность (профиль) «Технологии разработки программного обеспечения»
наименование ОПОП

Б1.В.ДВ.06.01
шифр дисциплины

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

Дисциплины (модуля) **Разработка web-приложений виртуальной реальности**

Разработчик (и):
Ляш О.И.
ФИО
зав.кафедрой
должность

канд.пед.наук, доцент
ученая степень, звание

Утверждено на заседании кафедры информационных технологий
наименование кафедры

протокол № 6 от 24.02.2026

Заведующий кафедрой ИТ

подпись Ляш О.И.
ФИО

1. Критерии и средства оценивания компетенций и индикаторов их достижения, формируемых дисциплиной (модулем)

Код и наименование компетенции	Код и наименование индикатора(ов) достижения компетенции	Результаты обучения по дисциплине (модулю)			Оценочные средства текущего контроля	Оценочные средства промежуточной аттестации
		Знать	*Уметь*	*Владеть*		
ПК-1 Способен разрабатывать технические спецификации на программные компоненты и их взаимодействие	ИД-1$_{ПК-1}$ Анализирует существующую документацию (техническое задание, спецификации) и требования к программному обеспечению для выделения перечня подлежащих разработке программных компонентов. ИД-2$_{ПК-1}$ Разрабатывает, согласовывает и утверждает технические спецификации на программные компоненты, описывающие их структуру, функции, логику работы и атрибуты качества. ИД-3$_{ПК-1}$ Проектирует интерфейсы взаимодействия программных компонентов (API) и протоколы обмена данными между ними, фиксируя результаты в спецификации. ИД-4$_{ПК-1}$ Оформляет техническую документацию на компоненты и их взаимодействие в соответствии с требованиями стандартов.	– Принципы работы платформы Node.js: событийно-ориентированная архитектура, цикл событий (event loop), асинхронное программирование (callbacks, promises, async/await). – Типовые архитектурные паттерны серверных приложений (MVC, слоистая архитектура, микросервисы) и способы взаимодействия компонентов (REST, GraphQL, WebSocket).	– Анализировать техническое задание и выделять перечень программных компонентов для реализации на Node.js. – Проектировать интерфейсы взаимодействия компонентов (REST API, GraphQL-схемы) и фиксировать их в спецификациях с использованием OpenAPI/Swagger. – Выбирать архитектуру приложения (монолит, микросервисы) и выполнять декомпозицию на модули с учётом требований к качеству.	– Навыками разработки серверных приложений на Node.js с использованием фреймворков (Express.js, NestJS, Fastify) и встроенных модулей (http, fs, stream). – Методами асинхронного программирования и управления потоком событий (промисы, async/await, паттерны «Pub/Sub»). – Приёмами написания технической документации (спецификации API в Swagger, описание компонентов, инструкции по сборке). – Техниками модульной декомпозиции, проектирования архитектуры и оценки соответствия проекта требованиям (юнит-	– комплект заданий для выполнения лабораторных (практических) работ; – тестовые задания;	Результаты текущего контроля

ПК-2 Способен проектировать компьютерное программное обеспечение	ИД-1$_{ПК-2}$ Выбирает архитектуру программного обеспечения и декомпозирует его на модули и компоненты. ИД-2$_{ПК-2}$ Применяет объектно-ориентированные или иные методологии для разработки моделей предметной области, структур данных и бизнес-логики будущего ПО. ИД-3$_{ПК-2}$ Разрабатывает логическую и физическую схемы баз данных, а также осуществляет проектирование пользовательских интерфейсов с учетом требований эргономики. ИД-4$_{ПК-2}$ Создает проектную документацию на программное обеспечение, позволяющую осуществить кодирование и дальнейшее развертывание продукта. ИД-5$_{ПК-2}$ Оценивает соответствие разработанного проекта исходным требованиям и техническому заданию, а также анализирует возможные риски реализации.	– Методы проектирования баз данных (реляционные и NoSQL) и стандарты оформления технической документации (OpenAPI, Swagger, ЕСПД). – Основные метрики качества ПО (производительность, масштабируемость, безопасность) и подходы к анализу рисков в среде Node.js.	– Разрабатывать логическую и физическую схемы баз данных, интегрируя их с бизнес-логикой на Node.js, а также создавать проектную документацию для последующего кодирования и развёртывания.	тестирование, профилирование производительности).		

2. Оценка уровня сформированности компетенций (индикаторов их достижения)

Показатели оценивания компетенций (индикаторов их достижения)	**Шкала и критерии оценки уровня сформированности компетенций (индикаторов их достижения)**			
	Ниже порогового *(«неудовлетворительно»)*	**Пороговый** *(«удовлетворительно»)*	**Продвинутый** *(«хорошо»)*	**Высокий** *(«отлично»)*
Полнота знаний	Уровень знаний ниже минимальных требований. Имели место грубые ошибки.	Минимально допустимый уровень знаний. Допущены не грубые ошибки.	Уровень знаний в объёме, соответствующем программе подготовки. Допущены некоторые погрешности.	Уровень знаний в объёме, соответствующем программе подготовки.
Наличие умений	При выполнении стандартных заданий не продемонстрированы основные умения. Имели место грубые ошибки.	Продемонстрированы основные умения. Выполнены типовые задания с не грубыми ошибками. Выполнены все задания, но не в полном объеме (отсутствуют пояснения, неполные выводы)	Продемонстрированы все основные умения. Выполнены все основные задания с некоторыми погрешностями. Выполнены все задания в полном объёме, но некоторые с недочетами.	Продемонстрированы все основные умения. Выполнены все основные и дополнительные задания без ошибок и погрешностей. Задания выполнены в полном объеме без недочетов.
Наличие навыков (владение опытом)	При выполнении стандартных заданий не продемонстрированы базовые навыки. Имели место грубые ошибки.	Имеется минимальный набор навыков для выполнения стандартных заданий с некоторыми недочетами.	Продемонстрированы базовые навыки при выполнении стандартных заданий с некоторыми недочетами.	Продемонстрированы все основные умения. Выполнены все основные и дополнительные задания без ошибок и погрешностей. Продемонстрирован творческий подход к решению нестандартных задач.
Характеристика сформированности компетенции	Компетенции фактически не сформированы. Имеющихся знаний, умений, навыков недостаточно для решения практических (профессиональных) задач. ИЛИ Зачетное количество баллов не набрано согласно установленному диапазону	Сформированность компетенций соответствует минимальным требованиям. Имеющихся знаний, умений, навыков в целом достаточно для решения практических (профессиональных) задач. ИЛИ Набрано зачетное количество баллов согласно установленному диапазону	Сформированность компетенций в целом соответствует требованиям. Имеющихся знаний, умений, навыков достаточно для решения стандартных профессиональных задач. ИЛИ Набрано зачетное количество баллов согласно установленному диапазону	Сформированность компетенций полностью соответствует требованиям. Имеющихся знаний, умений, навыков в полной мере достаточно для решения сложных, в том числе нестандартных, профессиональных задач. ИЛИ Набрано зачетное количество баллов согласно установленному диапазону

3. Критерии и шкала оценивания заданий текущего контроля

За выполнение всех видов работ на курсе можно получить 100 баллов. При выполнении работ студент получает первичный балл (ПБ) в диапазоне от 0 до 1 в соответствии с приведенными ниже таблицами. Все баллы, полученные студентом за текущий контроль и промежуточную аттестацию суммируются и пропорционально преобразуются к итоговому значению не превышающему величину 100 баллов.

Для «зачёта» и «зачёта с оценкой» за текущий контроль можно получить максимально 100 баллов.

Для «экзамена» за текущий контроль можно получить максимально 80 баллов, а за промежуточную аттестацию можно получить максимально «20 баллов».

3.1 Критерии и шкала оценивания лабораторных работ

Перечень лабораторных, описание порядка выполнения и защиты работы, требования к результатам работы, структуре и содержанию отчета и т.п. представлены в методических материалах по освоению дисциплины (модуля) и в электронном курсе в ЭИОС МАУ.

Оценка	Баллы	Критерии оценивания
Отлично	*0.9-1*	Задание выполнено полностью и правильно. Отчет по лабораторной/практической работе подготовлен качественно в соответствии с требованиями. Полнота ответов на вопросы преподавателя при защите работы.
Хорошо	*0.89*	Задание выполнено полностью, но нет достаточного обоснования или при верном решении допущена незначительная ошибка, не влияющая на правильную последовательность рассуждений. Все требования, предъявляемые к работе, выполнены.
Удовлетворительно	*0.6-0.79*	Задания выполнены частично с ошибками. Демонстрирует средний уровень выполнения задания на лабораторную/практическую работу. Большинство требований, предъявляемых к заданию, выполнены.
Неудовлетворительно	*0-0.59*	Задание выполнено со значительным количеством ошибок на низком уровне. Многие требования, предъявляемые к заданию, не выполнены. ИЛИ Задание не выполнено.

3.2. Критерии и шкала оценивания тестирования

Перечень тестовых вопросов и заданий, описание процедуры тестирования представлены в методических материалах по освоению дисциплины (модуля) и в электронном курсе в ЭИОС МАУ.

В ФОС включен типовой вариант тестового задания:

При проектировании API взаимодействия между клиентской частью web-VR-приложения и бэкенд-сервером наиболее подходящим протоколом для передачи данных в реальном времени (позиции игроков, события) является:

A) HTTP/1.1
B) WebSocket
C) FTP
D) SMTP

Оценка	Баллы	Критерии оценки

Тест зачтён	*0.6-1*	60-100 % правильных ответов
Тест не зачтён	*0-0.59*	59 % и менее правильных ответов

3.3. Критерии и шкала оценивания расчетно-графической работы

Перечень заданий для расчетно-графической работы, рекомендации по выполнению представлены в методических материалах по освоению дисциплины (модуля) и в электронном курсе в ЭИОС МАУ.

В ФОС включен типовой вариант расчетно-графической работы:

Варианты предметных областей:

1. Система управления задачами (Task Manager) – создание, назначение, отслеживание статусов задач, комментарии, уведомления.
2. Блог-платформа – публикация статей, комментарии, теги, лайки, роли (админ, автор, читатель).
3. API для интернет-магазина – товары, корзина, заказы, пользователи, платежи (без реальной оплаты).
4. Система бронирования (отели/конференц-залы) – поиск доступных слотов, бронирование, отмена, расписание.
5. Чат-сервер – комнаты, обмен сообщениями, история, уведомления о прочтении.

Состав и содержание расчетно-графической работы

Работа включает пояснительную записку и графическую часть (схемы, диаграммы, макеты сцен).

1. Анализ требований и выделение компонентов.
   - Описание функциональных и нефункциональных требований к системе.
   - Выделение перечня программных компонентов (модули, сервисы, контроллеры, репозитории и т.д.).
2. Разработка технических спецификаций на компоненты и их взаимодействие.
   - Для каждого компонента описать: структуру, функции, логику работы, атрибуты качества (производительность, надёжность, безопасность).
   - Спроектировать интерфейсы взаимодействия компонентов: REST API или GraphQL (указать методы, маршруты, форматы запросов/ответов, коды состояния).
   - Описать протоколы обмена данными (HTTP/HTTPS, WebSocket, JSON). Зафиксировать спецификацию в формате OpenAPI (Swagger). Предоставить фрагмент спецификации (YAML/JSON).
3. Проектирование архитектуры приложения
   - Выбрать архитектуру приложения (монолитная, микросервисная, модульная). Обосновать выбор.
   - Представить диаграмму компонентов и диаграмму развёртывания (UML).
   - Выполнить декомпозицию системы на модули и компоненты. Описать взаимодействие между ними.
4. Проектирование базы данных.
   - Разработать логическую схему базы данных (ER-диаграмма).
   - Разработать физическую схему (типы данных, индексы, ограничения целостности). Для реляционной БД – структура таблиц и связей; для NoSQL – структура документов/коллекций.
   - Привести примеры SQL-запросов (или запросов к NoSQL) для ключевых операций.
5. Проектирование пользовательских интерфейсов.
   - Описать API как пользовательский интерфейс для внешних клиентов (мобильное приложение, веб-клиент).

- Привести примеры запросов и ответов (POST, GET, PUT/PATCH, DELETE). При необходимости – описать формат WebSocket сообщений.
- Схематично изобразить основные сценарии взаимодействия (последовательность запросов).

6. Проектная документация и оценка рисков.
   - Создать фрагмент проектной документации, достаточный для последующего кодирования и развёртывания (инструкция по запуску, описание конфигурации, переменные окружения).
   - Оценить соответствие спроектированного решения исходным требованиям (таблица соответствия).
   - Провести анализ возможных рисков реализации: производительность (узкие места), безопасность (угрозы), масштабируемость, отказоустойчивость. Предложить меры по минимизации рисков.

Графическая часть (обязательные схемы)

1. Диаграмма компонентов (UML Component Diagram).
2. Диаграмма развёртывания (UML Deployment Diagram) – для выбранной архитектуры.
3. ER-диаграмма базы данных (логическая схема).
4. Схема взаимодействия компонентов при выполнении одного ключевого сценария (например, создание заказа или отправка сообщения) – диаграмма последовательности (UML Sequence Diagram).

Оценка	Баллы	Критерии оценивания
Отлично	*0.9-1*	Работа выполнена полностью, без ошибок (возможна одна неточность, описка, не являющаяся следствием непонимания материала).
Хорошо	*0.8-0.89*	Работа выполнена полностью, но обоснования шагов решения недостаточны, допущена одна негрубая ошибка или два-три недочета, не влияющих на правильную последовательность рассуждений.
Удовлетворительно	*0.6-0.79*	В работе допущено более одной грубой ошибки или более двух-трех недочетов, но обучающийся владеет обязательными умениями по проверяемой теме.
Неудовлетворительно	*0-0.59*	В работе есть грубые ошибки и недочеты ИЛИ Контрольная работа не выполнена.

3.3. Критерии и шкала оценивания реферата

Тематика рефератов по дисциплине (модулю), требования к структуре, содержанию и оформлению изложены в методических материалах по освоению дисциплины (модуля), представлены в электронном курсе в ЭИОС МАУ.

В ФОС включены примерные темы рефератов:

1. Событийно-ориентированная архитектура Node.js: принципы работы event loop.
2. Сравнение моделей асинхронного программирования: колбэки, промисы, async/await.
3. Модульная система в Node.js: CommonJS против ES Modules.
4. Разработка HTTP-сервера на чистом Node.js без использования фреймворков.
5. Обзор и сравнительный анализ фреймворков Express.js, Fastify и NestJS.
6. Middleware в Express.js: создание, применение, цепочки вызовов.
7. Проектирование REST API на Node.js: принципы, маршрутизация, версионирование.
8. Документирование API с помощью OpenAPI (Swagger) в Node.js проектах.
9. Архитектурный паттерн MVC в контексте Node.js приложений.

10. Микросервисная архитектура на Node.js: взаимодействие через HTTP/gRPC и очереди сообщений.
11. Проектирование и интеграция реляционных баз данных (PostgreSQL, MySQL) с использованием ORM (Sequelize, TypeORM).
12. Работа с NoSQL базами данных (MongoDB) и ODM Mongoose.
13. Разработка GraphQL API на Node.js (Apollo Server, GraphQL Yoga).
14. Аутентификация и авторизация в Node.js: JWT, сессии, OAuth2, ролевые модели.
15. Хеширование паролей и безопасное хранение данных (bcrypt, криптографические библиотеки).
16. Обработка ошибок и централизованное логирование в Node.js приложениях.
17. Тестирование Node.js приложений: unit-тесты (Jest, Mocha), интеграционные тесты (Supertest).
18. Нагрузочное тестирование и профилирование производительности (clinic.js, k6, Artillery).
19. Управление процессами в production: PM2, кластеризация, мониторинг.
20. Контейнеризация Node.js приложений с Docker и оркестрация с Docker Compose.
21. CI/CD для Node.js: настройка пайплайнов (GitHub Actions, GitLab CI, Jenkins).
22. Развертывание Node.js приложений в облачных платформах (AWS Elastic Beanstalk, Yandex Cloud, Heroku, Render).
23. Защита Node.js приложений от распространённых уязвимостей (XSS, CSRF, SQL/NoSQL инъекции, DoS).
24. Обеспечение безопасности зависимостй: уязвимости в npm-пакетах, инструменты аудита (npm audit, Snyk).
25. Работа с потоковыми данными (Streams) в Node.js: чтение, запись, pipeline, обработка больших объёмов данных.
26. Создание WebSocket-серверов на Node.js (библиотека ws, Socket.IO) для real-time приложений.
27. Использование worker_threads для параллельных вычислений в Node.js.
28. Кэширование в Node.js приложениях: стратегии, in-memory кэш, Redis.
29. Паттерны проектирования в Node.js: Singleton, Factory, Middleware, Dependency Injection, Observer.
30. Стандарты оформления технической и проектной документации для Node.js проектов (JSDoc, UML, Markdown, гайды по развёртыванию).
31. Сравнительный анализ Node.js и Python (Django/Flask) для разработки серверных приложений: производительность, экосистема, области применения.
32. Node.js против PHP (Laravel/Symfony): модели асинхронности, обработка запросов, масштабируемость.
33. Сравнение Java (Spring Boot) и Node.js: многопоточность против событийно-ориентированной архитектуры, потребление памяти, сферы использования.
34. Node.js и Go: сравнение производительности, работы с параллелизмом (goroutine vs event loop), удобства разработки.
35. .NET Core (C#) против Node.js: скорость выполнения, поддержка асинхронности, экосистема для микросервисов.
36. Сравнение асинхронных моделей: event loop в Node.js vs async/await в Python (asyncio) vs промисы в JavaScript.
37. Node.js (Express) против Go (Gin/Fiber) для создания высоконагруженных REST API: бенчмарки и практические сценарии.
38. Сравнение Deno и Node.js: архитектура, безопасность, модульная система, поддержка TypeScript, перспективы замены.
39. Bun как альтернатива Node.js: производительность, встроенные возможности (bundler, transpiler), совместимость с существующими npm-пакетами.

40. Сравнение Node.js с Elixir (Phoenix Framework): модель акторов, real-time коммуникации, отказоустойчивость.
41. Node.js против Ruby on Rails: философия разработки, соглашения против конфигурации, производительность и удобство прототипирования.
42. Сравнение подходов к обработке длительных соединений: WebSocket в Node.js vs WebSocket в Go/Java/Python.
43. Управление памятью и сборка мусора в Node.js (V8) vs в .NET и Java: влияние на производительность серверных приложений.
44. Экосистемы пакетов: npm (Node.js) против PyPI (Python), Maven (Java), Go Modules — сравнение удобства, безопасности, зрелости.
45. Сравнение инструментов для обработки потоков данных: Streams API в Node.js против Reactor в Java против асинхронных потоков в Python.

Оценка	баллы	Критерии оценки
Отлично	*0.9-1*	Выполнены все требования к написанию и защите реферата: обозначена проблема и обоснована ее актуальность, сделан краткий анализ различных точек зрения на рассматриваемую проблему и логично изложена собственная позиция, сформулированы выводы, тема раскрыта полностью, выдержан объем, соблюдены требования к внешнему оформлению, даны правильные ответы на дополнительные вопросы.
Хорошо	*0.8-0.89*	Основные требования к реферату и его защите - выполнены, но при этом допущены недочеты. В частности, имеются неточности в изложении материала; отсутствует логическая последовательность в суждениях; не выдержан объем реферата; имеются упущения в оформлении; на дополнительные вопросы при защите даны неполные ответы.
Удовлетворительно	*0.6-0.79*	Имеются существенные отступления от требований к реферированию. В частности, тема освещена лишь частично; допущены фактические ошибки в содержании реферата или при ответе на дополнительные вопросы; во время защиты отсутствует вывод.
Неудовлетворительно	*0-0.59*	Тема реферата не раскрыта, обнаруживается существенное непонимание проблемы.

3.4. Критерии и шкала оценивания эссе

Тематика эссе по дисциплине (модулю), требования к структуре, содержанию и оформлению представлены в методических материалах по освоению дисциплины (модуля) и в электронном курсе в ЭИОС МАУ.

В ФОС включены примерные темы эссе:

1. История развития технологий виртуальной реальности.
2. Практическое применение технологий виртуальной реальности.
3. Перспективы развития технологий виртуальной реальности.

Оценка	баллы	Критерии оценки
Отлично	*0.9-1*	Представлена собственная точка зрения (позиция, отношение) при раскрытии проблемы. Проблема раскрыта на теоретическом уровне, в связях и с обоснованиями, с корректным использованием обществоведческих терминов и понятий в контексте ответа. Предоставлена аргументация своего мнения с опорой на факты общественной жизни или личный социальный опыт.
Хорошо	*0.8-0.89*	Представлена собственная точка зрения (позиция, отношение) при раскрытии проблемы. Проблема раскрыта с корректным использованием терминов и понятий в контексте ответа (теоретические связи и обоснования не присутствуют или явно не прослеживаются). Представлена аргументация своего

		мнения с опорой на факты общественной жизни или личный социальный опыт.
Удовлетворительно	*0.6-0.79*	Представлена собственная точка зрения (позиция, отношение) при раскрытии проблемы; проблема раскрыта при формальном использовании обществоведческих терминов. Представлена аргументация своего мнения с опорой на факты общественной жизни или личный социальный опыт без теоретического обоснования.
Неудовлетворительно	*0-0.59*	Представлена собственная точка зрения (позиция, отношение) при раскрытии проблемы, но проблема раскрыта не полностью. Аргументация своего мнения слабо связана с раскрытием проблемы.

3.5. Критерии и шкала оценивания мультимедийной презентации

Требования к структуре, содержанию и оформлению представлены в методических материалах по освоению дисциплины (модуля) и в электронном курсе в ЭИОС МАУ.

Оценка	**баллы**	**Критерии оценки**
Отлично	*0.9-1*	Презентация соответствует теме самостоятельной работы. Оформлен титульный слайд с заголовком. Сформулированная тема ясно изложена и структурирована, использованы графические изображения (фотографии, картинки и т.п.), соответствующие теме, выдержан стиль, цветовая гамма, использована анимация, звук. Логично изложена собственная позиция, сформулированы выводы, тема раскрыта полностью, выдержан объём, соблюдены требования к внешнему оформлению. Работа оформлена и предоставлена в установленный срок.
Хорошо	*0.8-0.89*	Презентация соответствует теме самостоятельной работы. Имеются неточности в изложении материала. Отсутствует логическая последовательность в суждениях. Не выдержан объем презентации, имеются упущения в оформлении. На дополнительные вопросы при защите даны неполные ответы. Работа оформлена и предоставлена в установленный срок.
Удовлетворительно	*0.6-0.79*	Презентация соответствует теме самостоятельной работы. Сформулированная тема изложена и структурирована не в полном объеме. Не использованы графические изображения (фотографии, картинки и т.п.), соответствующие теме. Присутствуют существенные отступления от требований к составлению презентации. Допущены фактические ошибки в содержании или при ответе на дополнительные вопросы.
Неудовлетворительно	*0-0.59*	Работа не выполнена или не соответствует теме самостоятельной работы.

3.6. Критерии и шкала оценивания посещаемости занятий

Посещение занятий обучающимися определяется в процентном соотношении

Баллы	**Критерии оценки**
0.75-1	посещаемость 75 - 100 %
0.5-0.74	посещаемость 50 - 74 %
0-0.49	посещаемость менее 50 %

3.7. Критерии и шкала оценивания своевременной сдачи контрольных точек

Своевременность сдачи контрольных точек обучающимися определяется в процентном соотношении

Баллы	**Критерии оценки**
0.75-1	Своевременность сдачи 75 - 100 %
0.6-0.74	Своевременность сдачи 50 - 74 %

0-0.5	Своевременность сдачи 50 %

4. Критерии и шкала оценивания результатов обучения по дисциплине (модулю) при проведении промежуточной аттестации (зачёт с оценкой)

Критерии и шкала оценивания результатов освоения дисциплины (модуля) с зачетом с оценкой

Если обучающийся набрал зачетное количество баллов согласно установленному диапазону по дисциплине (модулю), то он считается аттестованным с оценкой согласно шкале баллов для определения итоговой оценки:

Оценка	Баллы	Критерии оценивания
Отлично	91 - 100	Набрано зачетное количество баллов согласно установленному диапазону
Хорошо	81 - 90	Набрано зачетное количество баллов согласно установленному диапазону
Удовлетворительно	60 - 80	Набрано зачетное количество баллов согласно установленному диапазону
Неудовлетворительно	менее 60	Зачетное количество согласно установленному диапазону баллов не набрано

5. Задания диагностической работы для оценки результатов обучения по дисциплине (модулю) в рамках внутренней и внешней независимой оценки качества образования

ФОС содержит задания для оценивания знаний, умений и навыков, демонстрирующих уровень сформированности компетенций и индикаторов их достижения в процессе освоения дисциплины (модуля).

Комплект заданий разработан таким образом, чтобы осуществить процедуру оценки каждой компетенции, формируемых дисциплиной (модулем), у обучающегося в письменной форме.

Содержание комплекта заданий включает: *тестовые задания.*

Комплект заданий диагностической работы

ПК-1 ***Способен разрабатывать технические спецификации на программные компоненты и их взаимодействие***	
1.	При разработке спецификации на REST API для Node.js-приложения обязательным элементом описания каждого маршрута является: A) Только HTTP-метод и URL B) HTTP-метод, URL, формат запроса/ответа, коды состояния C) Название контроллера и имя модели D) Версия фреймворка Express и порт сервера
2.	В технической спецификации на программный компонент «Сервис аутентификации» на Node.js необходимо описать: A) Только алгоритм хеширования паролей B) Структуру входных и выходных данных, логику работы, зависимости от других компонентов C) Цветовую схему интерфейса входа D) Тип процессора на сервере

3.	Какой формат описания API позволяет автоматически генерировать интерактивную документацию и клиентские SDK для Node.js? A) WSDL B) RAML C) OpenAPI (Swagger) D) GraphQL Schema Language (только для GraphQL)
4.	При проектировании взаимодействия микросервисов на Node.js наиболее подходящим протоколом для асинхронной передачи сообщений с гарантией доставки является: A) HTTP/2 без подтверждения B) AMQP (RabbitMQ) C) WebSocket без очередей D) UDP
5.	В спецификации на компонент «Логгер» для Node.js-приложения атрибут качества «уровень детализации логирования» должен позволять настраивать: A) Только включение/отключение записи B) Размер файла лога C) Минимальный уровень серьёзности (error, warn, info, debug) и форматы вывода D) Имя хоста и PID процесса
6.	Какое описание интерфейса взаимодействия между компонентами (Node.js модулями) считается фиксацией в технической документации? A) Схема базы данных B) Функциональная спецификация с сигнатурами методов, типами данных и ожидаемым поведением C) UML-диаграмма классов без методов D) Текст лицензионного соглашения
7.	При разработке спецификации на WebSocket-взаимодействие в Node.js (библиотека ws) необходимо определить: A) Только IP-адрес сервера B) События (on('message'), on('connection')), формат передаваемых сообщений (JSON/бинарный), обработку разрывов соединения C) Используемый HTTP-порт по умолчанию D) Сертификат SSL
8.	Какой стандарт оформления технической документации на программные компоненты чаще всего используется в отечественной ИТ-разработке? A) IEEE 829 B) ГОСТ 19 (ЕСПД) C) ISO 25010 D) W3C Recommendation
9.	При согласовании спецификации на компонент «Кэш на Redis» для Node.js необходимо указать: A) Стратегии инвалидации кэша и время жизни ключей (TTL) B) Только IP-адрес сервера Redis C) Название npm-пакета для подключения D) Внутреннее устройство Redis
10.	В технической спецификации на взаимодействие компонентов (API REST) атрибут качества «идемпотентность» обязателен для метода: A) POST B) GET C) PUT / DELETE D) OPTIONS
ПК-2 ***Способен проектировать компьютерное программное обеспечение***	
11.	Какой архитектурный паттерн наиболее естественен для Node.js-приложений с интенсивным вводом-выводом (I/O-bound)?

	A) Многопоточность с блокировками B) Событийно-ориентированная архитектура (event-driven) C) Акторная модель (как в Erlang) D) Пакетная обработка (batch)
12.	При выборе архитектуры для Node.js-приложения с большим количеством CPU-вычислений (например, обработка изображений) наиболее правильным решением будет: A) Использовать один поток и надеяться на производительность V8 B) Вынести вычисления в отдельные worker_threads или микросервис на другом языке C) Увеличить количество инстансов через PM2 (fork mode) D) Использовать синхронные версии всех функций
13.	При проектировании базы данных для Node.js-приложения, где критична скорость чтения и структура документов часто меняется, предпочтительнее использовать: A) Реляционную БД с жёсткой схемой B) NoSQL (MongoDB) с гибкой схемой C) Графовую БД (Neo4j) в любом случае D) Файловую систему как основное хранилище
14.	В рамках методологии объектно-ориентированного проектирования Node.js-модуля, какой принцип означает, что модуль должен зависеть от абстракций, а не от конкретных реализаций? A) Single Responsibility B) Open/Closed C) Liskov Substitution D) Dependency Inversion
15.	При проектировании логической схемы реляционной БД для интернет-магазина на Node.js связь «один ко многим» между таблицами «Заказы» и «Товары в заказе» реализуется через: A) Внешний ключ в таблице «Заказы» B) Внешний ключ в таблице «Товары в заказе» C) Отдельную таблицу связи с двумя внешними ключами D) Денормализацию и хранение списка товаров в поле JSON
16.	Какая диаграмма UML лучше всего подходит для документирования физического развёртывания Node.js-приложения (сервера, контейнеры, БД)? A) Диаграмма компонентов B) Диаграмма развёртывания (deployment) C) Диаграмма последовательности D) Диаграмма деятельности
17.	Оценка соответствия спроектированного Node.js-решения исходным требованиям (ИД-5ПК-2) предполагает: A) Только проверку бюджета проекта B) Составление матрицы трассировки требований к реализованным функциям C) Измерение частоты кадров интерфейса D) Подсчёт количества строк кода
18.	При проектировании пользовательского интерфейса API (для фронтенда) на Node.js рекомендуется использовать: A) Методы GET для всех операций для упрощения B) Ресурсно-ориентированный дизайн (REST) с правильным использованием HTTP-глаголов C) Только POST для всех действий с указанием команды в теле запроса D) SOAP с XML
19.	Анализ риска «перегрузка event loop» в Node.js приложении выполняется с помощью: A) Логирования всех входящих запросов B) Профилирования (clinic.js, inspector) и мониторинга задержки между таймерами C) Увеличения памяти процесса D) Запуска приложения в одном потоке
20.	Какой метод проектирования позволяет уменьшить риск возникновения «ада» колбэков (callback hell) в Node.js?

	A) Использование синхронных функций B) Применение промисов и async/await C) Вынос логики в отдельный файл D) Отказ от асинхронных операций полностью

Ключ к тесту: 1-B; 2-B; 3-C; 4-B; 5-C; 6-B; 7-B; 8-B; 9-A; 10-C; 11-B; 12-B; 13-B; 14-D; 15-B; 16-B; 17-B; 18-B; 19-B; 20-B.