Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Костровец Лариса Борисовна
Должность: директор
Дата подписания: 02.10.2026 00:12:43
Уникальный программный ключ:
6882606104c36dbde41c4ab93a65382136a292d6

Приложение 4
к образовательной программе

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.В.01.16 Разработка информационных систем
(индекс, наименование дисциплины в соответствии с учебным планом)

09.03.03 Прикладная информатика
(код, наименование направления подготовки/специальности)

Прикладная информатика в управлении корпоративными информационными системами
(наименование образовательной программы)

Очная форма обучения
(форма обучения)

Год набора – 2026
Донецк

Автор(ы)-составитель(и) РПД:

Лебезова Элла Михайловна, старший преподаватель кафедры информационных технологий

Заведующий кафедрой:

Брадул Наталья Валерьевна, канд. физ.-мат. наук, заведующий кафедрой информационных технологий

Рабочая программа дисциплины Б1.В.01.16 Разработка информационных систем одобрена на заседании кафедры информационных технологий администрирования факультета государственной службы и управления Донецкого филиала РАНХиГС.

Протокол № 7 от «05 » марта 2026 г.

СОДЕРЖАНИЕ

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

2. Объем и место дисциплины в структуре образовательной программы

3. Содержание и структура дисциплины

4. Типы оценочных материалов, показатели и критерии их оценивания

5. Формы аттестации, типовые оценочные материалы для текущего контроля успеваемости обучающихся, критерии и шкалы оценивания по контрольным точкам

6. Формы промежуточной аттестации, критерии и шкала оценивания, типовые оценочные материалы по дисциплине

7. Методические материалы по освоению дисциплины

8. Учебная литература и ресурсы информационно-телекоммуникационной сети «Интернет»

9. Материально-техническая база, информационные технологии, программное обеспечение и информационные справочные системы

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

Дисциплина Б1.В.01.16 Разработка информационных систем обеспечивает формирование у обучающихся следующих общепрофессиональных компетенций*:

ОТФ /ТФ и реквизиты ПС *(при наличии)* **	**Код компетенции ****	**Наименование Компетенции ****	**Код индикатора достижения компетенций ****	**Наименование индикатора достижения компетенций ****	**Образовательный результат ****
-	ПК-1.	Способность адаптировать бизнес-процессы заказчика ИС к возможностям типовой ИС в рамках выполнения работ по созданию (модификации) и сопровождению ИС	ПК-1.4.	Согласовывает с заказчиком ИС предлагаемые изменения его бизнес процессов для реализации их автоматизации в типовой ИС	**Знает** Основы управления организационными изменениями; Коммуникационное оборудование; Сетевые протоколы; Основы современных операционных систем; Основы современных СУБД; Устройство и функционирование современных ИС; Основы архитектуры мультиарендного программного обеспечения; Основы ИБ организации; Современные стандарты информационного взаимодействия систем; Программные средства и платформы

					инфраструктуры информационных технологий организаций Современные подходы и стандарты автоматизации организации; Лучшие практики создания (модификации) и сопровождения ИС в экономике; Основы организации производства. **Умеет** Проводить переговоры с заинтересованными сторонами в рамках выполнения работ по созданию (модификации) и сопровождению ИС.

** Дисциплина может формировать компетенцию полностью или частично.*
***Должно соответствовать Приложению 1 к образовательной программе*

2. Объем и место дисциплины в структуре образовательной программы

Общий объем дисциплины:

7,00 з.е., 252 ак.час

Контактная работа обучающихся с преподавателем по видам учебных занятий: 46+71 ак. час на контактную работу с преподавателем, из них 14+20 ак. час на лекции и 28+40 ак. час на практические занятия. 62+55 ак. час на самостоятельную работу обучающихся.

Б1.В.01.16 Разработка информационных систем реализуется в 7-м и 8-м семестре 4-го курса после изучения дисциплин:

- Объектно-ориентированное программирование
- Разработка программных приложений на C#/C++
- Технологии обеспечения информационной безопасности
- Базы данных
- Организация компьютерных сетей и телекоммуникаций
- Информатика и программирование
- Проектирование информационных систем
- Управление программными проектами.

.

3. Содержание и структура дисциплины

3.1. Структура дисциплины

Очная форма обучения

№ п/п	Наименование тем и (или) разделов		Объем дисциплины, ак.час												Форма текущего контроля успеваемости, промежуточной аттестации
		ВСЕГО	Контактная работа обучающихся с преподавателем по видам учебных занятий									Самостоятельная работа			
			Период теоретического обучения						Период промежуточной аттестации (сессия)						
			Занятия лекционного типа		Занятия семинарского типа		ИК	КСР	КЭ	Каттэк	Контроль	СРкр	СРэк	СР	
			Л	ВЛ	ЛР	ПЗ									
РАЗДЕЛ 1. ОСНОВЫ РАЗРАБОТКИ ИНФОРМАЦИОННЫХ СИСТЕМ (7 семестр)															
Тема 1	Введение в разработку ИС. Архитектурные подходы	12	2	0	0	2	0	0	0	0	0	0	0	8	Контрольные вопросы, практические занятия, КР1
Тема 2	Базы данных в разработке ИС. Проектирование и ORM	24	4	0	0	6	0	0	0	0	0	0	0	14	Контрольные вопросы, практические занятия, КР1

Тема 3	Серверные веб-технологии. Бэкенд на Node.js и Express	34	4	0	0	10	0	0	0	0	0	0	0	20	Контрольные вопросы, практические занятия, КР 1
Тема 4	Стандарт REST API. Проектирование и реализация API	34	4	0	0	10	0	0	0	0	0	0	0	20	Контрольные вопросы, практические занятия, КР 1
Промежуточная аттестация		4	0	0	0	0	0	0	0	4	0	0	0	0	Зачёт
Итого 7 семестр		108	14	0	0	28	0	0	0	4	0	0	0	62	
РАЗДЕЛ 2. РАЗРАБОТКА ИС С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКОВ (8 семестр)															
Тема 5	Тестирование API и приложений	16	2	0	0	6	0	0	0	0	0	0	0	8	Контрольные вопросы, практические занятия, КР 2
Тема 6	Документирование API и проекта**.**	12	2	0	0	4	0	0	0	0	0	0	0	6	Контрольные вопросы, практические занятия, КР 2
Тема 7	Шаблонизация и серверный рендеринг	12	2	0	0	4	0	0	0	0	0	0	0	6	Контрольные вопросы, практические занятия, КР 2
Тема 8	Фронтенд-фреймворк Vue.js	36	6	0	0	12	0	0	0	0	0	0	0	18	Контрольные вопросы, практические занятия, КР 2

Тема 9	Контейнеризация и развёртывание ИС (DevOps основы)	20	4	0	0	6	0	0	0	0	0	0	0	10	Контрольные вопросы, практические занятия, КР 2
Тема 10	Итоговый проект. Разработка полноценной информационной системы	19	4	0	0	8	0	0	0	0	0	0	0	7	Контрольные вопросы, практические занятия, КР 2
Промежуточная аттестация		29	0	0	0	0	0	0	2	9	0	0	18	0	Экзамен
Итого 8 семестр		144	20	0	0	40	0	0	2	9	0	0	18	55	
Итого		**252**	**34**	**0**	**0**	**68**	**0**	**0**	**2**	**9**	**0**	**0**	**18**	**117**	

Используемые сокращения:

Л – лекции - занятия, предусматривающие преимущественную передачу учебной информации обучающимся педагогическими работниками организации и (или) лицами, привлекаемыми организацией к реализации образовательных программ на иных условиях,).

ВЛ – видео лекции.

ЛР – лабораторные работы.

ПЗ – практические занятия (за исключением лабораторных работ).

ИК – индивидуальные консультации.

КСР – контроль самостоятельной работы

КЭ – консультации перед экзаменом

Каттэк – контактная работа на аттестацию в период экзаменационных сессий

Контроль - контактная работа на аттестацию в период экзаменационных сессий для заочной формы обучения

СРкр – самостоятельная работа на подготовку курсовой работы/ курсового проекта.

СРэк – самостоятельная работа на подготовку к экзамену.

СР – самостоятельная работа в семестре на подготовку к учебным занятиям.

3.2. Содержание дисциплины

РАЗДЕЛ 1. ОСНОВЫ РАЗРАБОТКИ ИНФОРМАЦИОННЫХ СИСТЕМ

Тема 1. Введение в разработку ИС. Архитектурные подходы. ПК-1.4.

Изучаются понятие информационной системы (ИС), классификация ИС (OLTP, OLAP, CRM, ERP, CMS, ECM). Рассматриваются архитектурные стили: монолитная архитектура, клиент-серверная, многоуровневая (трехуровневая), микросервисная. Изучаются современные подходы: DevOps, Agile, Waterfall. Сравниваются типы приложений: классические десктопные, мобильные (нативные, кроссплатформенные), веб-приложения (SPA, SSR, статические сайты).

Тема 2. Базы данных в разработке ИС. Проектирование и ORM. ПК-1.4.

Рассматриваются реляционные (PostgreSQL, MySQL) и нереляционные (MongoDB) базы данных. Изучается проектирование схемы БД: нормализация, связи (1:1, 1:N, N:M), индексы, первичные и внешние ключи. Осваиваются базовые SQL-запросы (SELECT, INSERT, UPDATE, DELETE, JOIN, GROUP BY). Вводится понятие ORM (Object-Relational Mapping) на примере Sequelize (для Node.js). Изучаются миграции и сиды (seeders).

Тема 3. Серверные веб-технологии. Бэкенд на Node.js и Express. ПК-1.4.

Изучается платформа Node.js: событийная модель, асинхронность, модули (CommonJS, ES modules). Рассматривается стандарт ECMAScript (ES6+): стрелочные функции, деструктуризация, async/await, промисы, классы. Осваивается фреймворк Express.js: маршрутизация (routing), middleware, обработка запросов (req, res), парсинг тела запроса (body-parser). Создаётся простейший веб-сервер. Изучается работа с переменными окружения (dotenv) и конфигурация приложения.

Тема 4. Стандарт REST API. Проектирование и реализация API. ПК-1.4.

Изучается архитектурный стиль REST: принципы (клиент-сервер, отсутствие состояния, кеширование, единообразие интерфейса). Рассматриваются HTTP-методы (GET, POST, PUT, PATCH, DELETE) и коды ответа (200, 201, 400, 401, 404, 500). Проектируются RESTful эндпоинты для CRUD-операций. Изучаются версионирование API (URL, заголовки). Реализуется полноценное REST API на Express + Sequelize для работы с

сущностью (например, «Пользователи» или «Товары»). Рассматриваются обработка ошибок и валидация входных данных.

РАЗДЕЛ 2. РАЗРАБОТКА ИС С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКОВ

Тема 5. Тестирование API и приложений. ПК-1.4.

Изучаются виды тестирования: модульное (unit), интеграционное, end-to-end (E2E), нагрузочное. Рассматриваются инструменты для тестирования API: **Postman** (создание коллекций, переменные окружения, pre-request scripts, tests). Изучается автоматизация тестов в Postman (Runner, Newman). Также рассматриваются инструменты для unit-тестирования бэкенда на Node.js (Jest). Создаются простые тесты для проверки REST API эндпоинтов.

Тема 6. Документирование API и проекта. ПК-1.4.

Изучаются подходы к документированию ИС: техническая документация, пользовательская документация, API-документация. Рассматривается язык разметки **Markdown** (заголовки, списки, таблицы, ссылки, вставка кода). Изучаются инструменты для генерации документации API: **Swagger/OpenAPI** (написание спецификации в YAML/JSON, интеграция с Express через swagger-ui-express). Создаётся документация для разработанного REST API.

Тема 7. Шаблонизация и серверный рендеринг. ПК-1.4.

Рассматриваются шаблонизаторы для веб-разработки: **Pug (Jade)**, EJS, Handlebars. Изучается работа с шаблонизатором Pug: синтаксис, наследование шаблонов (layout), включения (include), передача данных из Express. Сравниваются серверный рендеринг (SSR) и клиентский рендеринг (SPA). Создаются динамические HTML-страницы с использованием шаблонизатора.

Тема 8. Фронтенд-фреймворк Vue.js. ПК-1.4.

Изучается прогрессивный JavaScript-фреймворк Vue.js 3: установка (CDN, Vue CLI, Vite), реактивность (ref, reactive), директива v-bind, v-model, v-for, v-if, v-show. Рассматриваются компонентный подход: создание однофайловых компонентов (SFC: template, script, style). Изучаются хуки жизненного цикла (mounted, created). Осваиваются маршрутизация в SPA через Vue Router, управление состоянием через Pinia (или Vuex). Реализуется подключение к бэкенд API через fetch или axios.

Тема 9. Контейнеризация и развёртывание ИС (DevOps основы) . ПК-1.4.

Вводится понятие контейнеризации и Docker. Изучаются Dockerfile (сборка образа), docker-compose.yml (оркестрация нескольких сервисов: Node.js, PostgreSQL, MongoDB, Nginx). Рассматриваются основы CI/CD (непрерывная интеграция и доставка). Изучаются платформы для развёртывания: Render, Railway, Heroku (или альтернативы). Выполняется развёртывание полноценной ИС (бэкенд + БД + фронтенд) в облаке.

Тема 10. Итоговый проект. Разработка полноценной информационной системы. ПК-1.4.

Выполняется интеграция всех изученных технологий: проектирование БД, разработка REST API на Node.js + Express, создание клиентской части на Vue.js, документирование API (Swagger), тестирование (Postman), контейнеризация (Docker) и развёртывание. Рассматриваются сбор и анализ требований, разработка технического задания. Защита курсового проекта (или итоговой работы).

4. Типы оценочных материалов, показатели и критерии оценивания

4.1. Оценочные материалы по дисциплине Б1.В.01.16 Разработка информационных систем входят в состав оценочных материалов по образовательной программе. Совокупность оценочных материалов по всем дисциплинам (модулям) образовательной программы составляет фонд оценочных средств (далее – ФОС). ФОС используется при проведении текущего контроля успеваемости и промежуточной аттестации обучающихся с целью оценивания достижения обучающимися планируемых результатов обучения.

4.2. ФОС разработан как комплекс проверочных заданий различного типа и уровня сложности, включает критерии и шкалы оценивания, а также «ключи» правильных ответов. ФОС формируется как отдельный документ и хранится в электронном виде, доступ к ФОС предоставлен ограниченному кругу лиц.

4.3. Для самостоятельной работы обучающихся при подготовке к текущему контролю успеваемости и промежуточной аттестации в рабочих программах дисциплин размещены типовые проверочные задания, которые можно условно разделить на задания закрытого, комбинированного и открытого типов.

Задания закрытого типа – это тестовые задания, в которых каждый вопрос сопровождается готовыми вариантами ответов, из которых необходимо выбрать один или несколько правильных.

Задания комбинированного типа – это тестовые задания, в которых каждый вопрос сопровождается готовыми вариантами ответов, из которых необходимо выбрать один или несколько правильных и обосновать свой выбор.

Задания открытого типа – это задания, в которых на каждый вопрос должен быть предложен развернутый обоснованный ответ.

В зависимости от типа задания рекомендованы определенная последовательность выполнения и система оценивания выполнения заданий.

4.4. Типы заданий, сценарии выполнения, критерии оценивания

ТИП ЗАДАНИЯ	ИНСТРУКЦИЯ	СЦЕНАРИИ ВЫПОЛНЕНИЯ	КРИТЕРИИ ОЦЕНИВАНИЯ
Задание закрытого типа с выбором одного правильного ответа из нескольких вариантов предложенных	Прочитайте текст, выберите правильный ответ	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается только один из предложенных вариантов. 2. Внимательно прочитать предложенные вариант-ты ответа. 3. Выбрать один верный ответ. 4. Записать только номер (или букву) выбранного варианта ответа (например, 3 или В).	Ответ считается верным, если правильно указана цифра или буква
Задание закрытого типа на установление соответствия	Прочитайте текст и установите соответствие	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидаются пары элементов. 2. Внимательно прочитать оба списка: список 1 – вопросы, утверждения, факты, понятия и т.д.; список 2 – утверждения, свойства объектов и т.д. 3. Сопоставить элементы списка 1 с элементами списка 2, сформировать пары элементов. 4. Записать попарно буквы и цифры (в зависимости от задания) вариантов ответа (например, А1 или Б4).	Ответ считается верным, если правильно указаны цифры или буквы

Задание закрытого типа с выбором нескольких правильных ответов из нескольких вариантов предложенных	Прочитайте текст, выберите правильные ответы	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается несколько правильных ответов из предложенных вариантов. 2. Внимательно прочитать предложенные вариант-ты ответа. 3. Выбрать несколько правильных ответов. 4. Записать только номера (или буквы) выбранного варианта ответа (например, 1 4 или А Г).	Ответ считается верным, если правильно установлены все соответствия (позиции из одного столбца верно сопоставлены с позициями другого)
Задание закрытого типа на установление последовательнос ти	Прочитайте текст и установите последовательность	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается последовательность элементов. 2. Внимательно прочитать предложенные варианты ответа. 3. Построить верную последовательность из предложенных элементов. 4. Записать буквы/цифры (в зависимости от задания) вариантов ответа в нужной последовательности (например, БВА или 135).	Ответ считается верным, если правильно указана вся последовательность цифр

Задание комбинированного типа с выбором одного правильного ответа из предложенных и обоснованием выбора	Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается только один из предложенных вариантов. 2. Внимательно прочитать предложенные варианты ответа. 3. Выбрать один верный ответ. 4. Записать только номер (или букву) выбранного варианта ответа. 5. Записать аргументы, обосновывающие выбор ответа (например, 4 текст обоснования).	Ответ считается верным, если правильно указана цифра или буква и приведены корректные аргументы, используемые при выборе ответа
Задание открытого типа с развернутым ответом	Прочитайте текст и запишите развернутый обоснованный ответ	1. Внимательно прочитать текст задания и понять суть вопроса. 2. Продумать логику и полноту ответа. 3. Записать ответ, используя четкие компактные формулировки. 4. В случае расчетной задачи, записать решение и ответ	Ответ считается верным: 1. Отсутствие фактических ошибок. 2. Раскрытие объема используемых понятий (полнота ответа). 3. Обоснованность ответа (наличие аргументов). 4. Логическая последовательность излагаемого материала.

4.5. Общая шкала оценивания результатов текущего контроля успеваемости и промежуточной аттестации обучающихся с применением БРС

Оценка по шкале ECTS	**Сумма баллов за все виды учебной деятельности**	**Оценка по государственной шкале**	**Определение**
A	90 – 100	«Отлично»	отличное выполнение с незначительным количеством неточностей
B	80 – 89	«Хорошо»	в целом правильно выполненная работа с незначительным количеством ошибок (до 10%)
C	75 – 79		в целом правильно выполненная работа с незначительным количеством ошибок (до 15%)
D	70 – 74	«Удовлетворительно»	неплохо, но со значительным количеством недостатков
E	60 – 69		выполнение удовлетворяет минимальные критерии
FX	35 – 59	«Не удовлетворительно»	с возможностью повторной сдачи
F	0 – 34		с обязательным повторным изучением дисциплины (выставляется комиссией)

Соотношение баллов за текущий контроль успеваемости и промежуточную аттестацию, а также повторную промежуточную аттестацию:

Максимальная сумма баллов за текущий контроль успеваемости	Максимальная сумма баллов за промежуточную аттестацию	Максимальная итоговая балльная оценка	Максимальная сумма баллов за повторную промежуточную аттестацию
100 баллов	100 баллов	100 баллов	100 баллов

5. Формы аттестации, типовые оценочные материалы для текущего контроля успеваемости обучающихся, критерии и шкалы оценивания по контрольным точкам

5.1. В ходе реализации дисциплины Б1.В.01.16 Разработка информационных систем используются следующие формы текущего

контроля успеваемости обучающихся (в том числе, задания к контрольным точкам):

Контрольные вопросы для проведения опроса, задания открытого типа на практических занятиях, контрольные задания

Таблица 5.1.

Распределение баллов по видам учебной деятельности (БРС) –7-й семестр

Раздел/Темы	Формы текущего контроля		КЗР
	УО	ПЗ	
Р-1. / Т-1	5	15	20
Р-1. / Т-2	5	15	
Р-1. / Т-3	5	15	
Р-1. / Т-4	5	15	
Итого: 100 б	20	60	20

УО – устный опрос;
ТЗ – тестовое задание;
КЗ – контрольные задания;
ПЗ – практическое занятие;
Д – доклад;
КЗР – контрольные работы по разделу.

Критерии оценивания опроса:

Балы	Описание критерия
5	Обучающийся полно излагает материал (отвечает на вопрос), дает правильное определение основных понятий; обнаруживает понимание материала, может обосновать свои суждения, применить знания на практике, привести необходимые примеры не только из учебника, но и самостоятельно составленные; излагает материал последовательно и правильно с точки зрения норм литературного языка.
4	Обучающийся дает ответ, удовлетворяющий тем же требованиям, что и для оценки «отлично», но допускает 1–2 ошибки, которые сам же исправляет, и 1–2 недочета в последовательности и языковом оформлении излагаемого.
3	Обучающийся обнаруживает знание и понимание основных положений данной темы, но излагает материал неполно и допускает неточности в определении понятий или формулировке правил; не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; излагает материал непоследовательно и допускает ошибки в языковом оформлении излагаемого.
2-0	Обучающийся обнаруживает незнание вопроса, допускает ошибки в формулировке определений и правил, искажающие их смысл, беспорядочно и неуверенно излагает материал.

0* - в журнал академической группы не выставляется

Критерии оценивания практических занятий:

Балы	Описание критерия

12-15	Свыше 90% правильных ответов.	Обучающийся демонстрирует глубокое познание в освоенном материале.
8-11	Свыше 70% правильных ответов.	Обучающимся материал освоен полностью, без существенных ошибок.
4-7	Реализовано более 50% поставленных задач	Обучающимся материал освоен не полностью, имеются значительные пробелы в знаниях.
0-3	Реализовано менее 30% поставленных задач.	Обучающимся материал не освоен, знания обучающегося ниже базового уровня.

0* - в журнал академической группы не выставляется

Критерии оценивания контрольных заданий:

Балы	Описание критерия
15-20	Обучающимся задание выполнено без ошибок и в полном объеме.
10-14	Обучающимся в целом задание выполнено, имеются отдельные неточности или недостаточно полные ответы, не содержащие ошибок.
5-9	Обучающимся допущены отдельные ошибки при выполнении задания
0-4	У обучающегося отсутствуют ответы на большинство вопросов задачи, задание не выполнено или выполнено не верно.

0* - в журнал академической группы не выставляется

Таблица 5.2.

Распределение баллов по видам учебной деятельности (БРС) – 8-й семестр

Раздел/Темы	Формы текущего контроля		КЗР
	УО	ПЗ	
Р-2. / Т-5	3	10	22
Р-2. / Т-6	3	10	
Р-2. / Т-7	3	10	
Р-2. / Т-8	3	10	
Р-2. / Т-9	3	10	
Р-2. / Т-10	3	10	
Итого: 100 б	18	60	22

УО – устный опрос;
ТЗ – тестовое задание;
КЗ – контрольные задания;
ПЗ – практическое занятие;
Д – доклад;
КЗР – контрольные работы по разделу.

Критерии оценивания опроса:

Балы	Описание критерия
3	Обучающийся полно излагает материал (отвечает на вопрос), дает правильное определение основных понятий; обнаруживает понимание материала, может обосновать свои суждения, применить знания на практике, привести необходимые примеры не только из учебника, но и самостоятельно составленные; излагает материал последовательно и правильно с точки зрения норм литературного языка.
2	Обучающийся дает ответ, удовлетворяющий тем же требованиям, что и для оценки «отлично», но допускает 1–2 ошибки, которые

	сам же исправляет, и 1–2 недочета в последовательности и языковом оформлении излагаемого.
1	Обучающийся обнаруживает знание и понимание основных положений данной темы, но излагает материал неполно и допускает неточности в определении понятий или формулировке правил; не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; излагает материал непоследовательно и допускает ошибки в языковом оформлении излагаемого.
0	Обучающийся обнаруживает незнание вопроса, допускает ошибки в формулировке определений и правил, искажающие их смысл, беспорядочно и неуверенно излагает материал.

0* - в журнал академической группы не выставляется

Критерии оценивания практических занятий:

Балы	Описание критерия	
9-10	Свыше 90% правильных ответов.	Обучающийся демонстрирует глубокое познание в освоенном материале.
7-8	Свыше 70% правильных ответов.	Обучающимся материал освоен полностью, без существенных ошибок.
4-6	Реализовано более 50% поставленных задач	Обучающимся материал освоен не полностью, имеются значительные пробелы в знаниях.
0-3	Реализовано менее 30% поставленных задач.	Обучающимся материал не освоен, знания обучающегося ниже базового уровня.

0* - в журнал академической группы не выставляется

Критерии оценивания контрольных заданий:

Балы	Описание критерия
18-22	Обучающимся задание выполнено без ошибок и в полном объеме.
11-17	Обучающимся в целом задание выполнено, имеются отдельные неточности или недостаточно полные ответы, не содержащие ошибок.
5-10	Обучающимся допущены отдельные ошибки при выполнении задания
0-4	У обучающегося отсутствуют ответы на большинство вопросов задачи, задание не выполнено или выполнено не верно.

0* - в журнал академической группы не выставляется

5.2. Типовые оценочные материалы для текущего контроля успеваемости обучающихся (вне контрольных работ):

РАЗДЕЛ 1. МЕТОДОЛОГИИ И ТЕХНОЛОГИИ РАЗРАБОТКИ ИС (7 семестр)

Тема 1. Введение в разработку ИС. Архитектурные подходы

Контрольные вопросы:

1. Что такое информационная система (ИС)? Приведите примеры.

2. Чем OLTP-системы отличаются от OLAP-систем?
3. Перечислите основные архитектурные стили разработки ИС.
4. В чем преимущества и недостатки монолитной архитектуры?
5. Что такое микросервисная архитектура? Когда её целесообразно применять?
6. Какие методологии разработки ПО существуют? В чем разница между Waterfall и Agile?
7. Чем SPA (Single Page Application) отличается от SSR (Server-Side Rendering)?

Практические занятия:

1. Выбрать предметную область (например, «Интернет-магазин», «Библиотека», «Система бронирования отелей»). Составить техническое задание (ТЗ) в произвольной форме (3–5 страниц).
2. Нарисовать диаграмму архитектуры выбранной ИС (блок-схема: клиент → сервер → БД → внешние API).
3. Сравнить монолитную и микросервисную архитектуру для выбранной системы (таблица преимуществ и недостатков).
4. Провести анализ трёх существующих ИС (например, Google Docs, Trello, Авито) с точки зрения архитектуры.
5. Подготовить презентацию (5 слайдов) на тему «Эволюция архитектуры ИС: от монолита к микросервисам».

Тема 2. Базы данных в разработке ИС. Проектирование и ORM

Контрольные вопросы:

1. Какие типы баз данных существуют (реляционные, документоориентированные, графовые, ключ-значение)?
2. Что такое нормализация? Назовите первые три нормальные формы.
3. Какие виды связей между таблицами существуют? Приведите примеры.
4. Что такое первичный и внешний ключ?
5. Какой SQL-запрос выполняет соединение двух таблиц? Чем INNER JOIN отличается от LEFT JOIN?
6. Что такое ORM? Какие проблемы он решает?
7. Что такое миграции и сиды (seeders)? Зачем они нужны?

Практические занятия:

1. Установить PostgreSQL (или MySQL) и создать базу данных для предметной области из темы 1.

2. Спроектировать схему БД (не менее 4 таблиц, связанных между собой). Нарисовать ER-диаграмму.
3. Написать SQL-запросы:
   - Создание таблиц (CREATE TABLE)
   - Вставка 5–10 тестовых записей (INSERT)
   - Выборка с JOIN (SELECT ... JOIN)
   - Обновление (UPDATE) и удаление (DELETE)
   - Агрегация (COUNT, SUM, AVG) с GROUP BY
4. Установить Sequelize (Node.js ORM). Описать модели для всех таблиц.
5. Написать скрипт на Node.js, который через Sequelize выполняет CRUD-операции.
6. Создать миграцию для добавления нового поля в таблицу и сид для заполнения тестовыми данными.

Тема 3. Серверные веб-технологии. Бэкенд на Node.js и Express

Контрольные вопросы:

1. Что такое Node.js? В чем его особенности (событийная модель, асинхронность)?
2. Чем CommonJS отличается от ES modules?
3. Что такое callback hell и как его избежать? (Promises, async/await)
4. Что такое middleware в Express.js? Приведите примеры встроенных middleware.
5. Как получить параметры запроса в Express (query, params, body)?
6. Как организовать маршрутизацию в Express (app.get, app.post, express.Router)?
7. Что такое переменные окружения? Как их использовать в Node.js (dotenv)?

Практические занятия:

1. Установить Node.js и npm. Создать проект backend-api.
2. Написать простой HTTP-сервер на чистом Node.js (без Express), возвращающий «Hello World».
3. Установить Express. Создать сервер с маршрутами:
   - GET / → текст «Главная страница»
   - GET /about → текст «О нас»
   - GET /users → JSON-массив пользователей (статический)
4. Добавить middleware для логирования всех запросов (вывод в консоль метода и URL).

5. Установить dotenv. Вынести порт сервера в переменную окружения PORT.

6. Создать обработчики ошибок (404 для несуществующих маршрутов, глобальный error handler).

7. Организовать маршруты с помощью express.Router() (например, routes/users.js, routes/products.js).

Тема 4. Стандарт REST API. Проектирование и реализация API

Контрольные вопросы:

1. Что такое REST API? Назовите основные принципы REST.
2. Какие HTTP-методы используются в REST API и для каких операций?
3. Какие HTTP-коды ответа существуют? (2xx, 3xx, 4xx, 5xx — примеры)
4. Что такое идемпотентность? Какие методы идемпотентны?
5. Как правильно спроектировать эндпоинты для работы с ресурсом «Пользователи»?
6. Что такое версионирование API? Какие способы существуют?
7. Как обрабатывать ошибки валидации в REST API?

Практические занятия:

1. Спроектировать REST API для сущности «Задачи» (Task) со следующими полями: id, title, description, status (pending/in_progress/done), created_at, updated_at.
2. Реализовать эндпоинты (Express + Sequelize):
   - GET /tasks — получение всех задач
   - GET /tasks/:id — получение задачи по ID
   - POST /tasks — создание задачи (валидация title)
   - PUT /tasks/:id — полное обновление задачи
   - PATCH /tasks/:id — частичное обновление
   - DELETE /tasks/:id — удаление задачи
3. Добавить фильтрацию: GET /tasks?status=done
4. Добавить пагинацию: GET /tasks?page=1&limit=10
5. Реализовать обработку ошибок:
   - 400 — неверные данные
   - 404 — задача не найдена
   - 500 — внутренняя ошибка сервера
6. Написать middleware для валидации входных данных (например, с помощью Joi или celebrate).

7. Добавить версионирование API через URL: /api/v1/tasks, /api/v2/tasks.

РАЗДЕЛ 2. РАЗРАБОТКА ИС С ПОМОЩЬЮ ФРЕЙМВОРКОВ (8 семестр)

Тема 5. Тестирование API и приложений

Контрольные вопросы:

1. Какие виды тестирования существуют? (unit, интеграционное, E2E)
2. Что такое Postman? Как создать коллекцию запросов?
3. Как в Postman автоматизировать тесты (pre-request scripts, tests)?
4. Что такое переменные окружения в Postman? Зачем они нужны?
5. Как запустить Postman-тесты из командной строки (Newman)?
6. Что такое unit-тестирование на Node.js? Какие инструменты используются (Jest)?
7. Какие аспекты API следует тестировать в первую очередь?

Практические занятия:

1. Установить Postman. Импортировать разработанное API из темы 4.
2. Создать коллекцию «Task API». Добавить запросы для всех эндпоинтов (GET, POST, PUT, DELETE).
3. Написать тесты в Postman для каждого запроса:
   - Проверка статус-кода (200, 201, 400, 404)
   - Проверка структуры ответа (наличие полей id, title)
   - Проверка сообщения об ошибке
4. Создать переменные окружения (development, production) с базовым URL.
5. Написать pre-request script для автоматической установки токена (если есть авторизация).
6. Установить Newman. Запустить коллекцию тестов из командной строки.
7. Написать unit-тесты на Jest для моделей Sequelize (проверка валидации полей).

Тема 6. Документирование API. Markdown и Swagger

Контрольные вопросы:

1. Для чего нужно документировать API?

2. Что такое Markdown? Перечислите основные элементы синтаксиса.
3. Что такое Swagger (OpenAPI)? Какова его структура?
4. Какие инструменты используются для генерации документации из спецификации?
5. Как интегрировать Swagger UI в Express-приложение?
6. Что такое YAML? Чем отличается от JSON?
7. Какие разделы должна содержать хорошая документация API?

Практические занятия:

1. Написать документацию к API из темы 4 в формате Markdown (файл API.md):
   - Общее описание
   - Базовый URL
   - Эндпоинты (метод, путь, параметры, тело запроса, пример ответа)
   - Коды ошибок
2. Установить пакеты swagger-ui-express и swagger-jsdoc.
3. Написать OpenAPI 3.0 спецификацию в формате YAML (или JSDoc-комментарии в коде).
4. Подключить Swagger UI к Express-приложению по маршруту /api-docs.
5. Добавить в документацию схемы данных (Task, CreateTaskRequest, ErrorResponse).
6. Сгенерировать HTML-документацию из спецификации.
7. Экспортировать спецификацию в JSON и проверить её валидность в Swagger Editor.

Тема 7. Шаблонизация и серверный рендеринг

Контрольные вопросы:

1. Что такое шаблонизатор? Какие шаблонизаторы существуют для Node.js?
2. Чем отличается серверный рендеринг (SSR) от клиентского (CSR/SPA)?
3. Каков синтаксис шаблонизатора Pug (основные конструкции)?
4. Как передать данные из Express в шаблон Pug?
5. Что такое наследование шаблонов (layout, extends, block)?
6. Как подключить статические файлы (CSS, JS) к шаблону?
7. В каких случаях SSR предпочтительнее SPA?

Практические занятия:

1. Установить Pug как шаблонизатор в Express (app.set('view engine', 'pug')).
2. Создать папку views. Добавить файл layout.pug с базовым HTML-каркасом (header, main, footer).
3. Создать страницы:
   - index.pug — главная страница
   - tasks.pug — список задач (полученный из БД через API)
   - task-detail.pug — страница отдельной задачи
4. Настроить маршруты Express, которые рендерят эти страницы и передают данные из БД.
5. Добавить CSS-стили в папку public, подключить их в layout.pug.
6. Реализовать форму создания задачи на отдельной странице (POST-запрос на API).
7. Сравнить производительность SSR-версии приложения с SPA-версией (инструменты разработчика в браузере).

Тема 8. Фронтенд-фреймворк Vue.js

Контрольные вопросы:

1. Что такое Vue.js? В чем его преимущества перед другими фреймворками?
2. Что такое реактивность в Vue? Как работают ref и reactive?
3. Какие директивы существуют в Vue (v-bind, v-model, v-for, v-if, v-show, v-on)?
4. Что такое однофайловый компонент (SFC)? Из каких частей он состоит?
5. Каков жизненный цикл компонента Vue (mounted, created и др.)?
6. Что такое Vue Router? Как организовать маршрутизацию в SPA?
7. Что такое Pinia (или Vuex)? Зачем нужно управление состоянием?

Практические занятия:

1. Создать проект Vue.js 3 с помощью Vite: npm create vue@latest.
2. Изучить структуру проекта (src/components, src/views, src/router, src/stores).
3. Создать компонент TaskList.vue, который отображает список задач (получает через props).
4. Создать компонент TaskForm.vue с полями title, description и кнопкой «Добавить». Использовать v-model.
5. Настроить Vue Router: маршруты / (список задач), /tasks/:id (детали задачи), /add (форма добавления).

6. Подключить бэкенд API через fetch или axios. Реализовать:
   - Загрузку списка задач (GET /tasks)
   - Создание задачи (POST /tasks)
   - Удаление задачи (DELETE /tasks)
7. Установить Pinia. Создать store useTaskStore для хранения списка задач и действий (loadTasks, addTask, deleteTask).

Тема 9. Контейнеризация и развёртывание ИС (DevOps основы)

Контрольные вопросы:

1. Что такое контейнеризация? В чем отличие контейнера от виртуальной машины?
2. Что такое Docker? Как работает Dockerfile?
3. Какие основные команды Docker вы знаете? (docker build, docker run, docker ps, docker-compose)
4. Что такое docker-compose? Для чего нужен файл docker-compose.yml?
5. Как организовать многосервисное приложение (бэкенд + БД + фронтенд) в Docker?
6. Что такое CI/CD? Какие инструменты существуют?
7. Какие платформы для развёртывания Node.js приложений вы знаете?

Практические занятия:

1. Установить Docker Desktop. Написать Dockerfile для Node.js приложения (Express API).
2. Собрать образ: docker build -t task-api .. Запустить контейнер с пробросом порта.
3. Написать docker-compose.yml для трёх сервисов:
   - postgres (или mysql)
   - backend (Express API, зависит от postgres)
   - frontend (сборка Vue.js через nginx или serve)
4. Запустить всё приложение командой docker-compose up -d.
5. Развернуть приложение на бесплатном хостинге (Render, Railway или Heroku).
6. Настроить переменные окружения на хостинге (DATABASE_URL, PORT).
7. Настроить простейший CI/CD: при пуше в GitHub автоматическая сборка и деплой (GitHub Actions или интеграция с Render).

Тема 10. Итоговый проект. Разработка полноценной информационной системы

Контрольные вопросы:

1. Какие этапы включает полный цикл разработки ИС?
2. Как собирать и анализировать требования к ИС?
3. Что такое MVP (Minimum Viable Product)?
4. Как организовать командную разработку (Git, Git Flow)?
5. Какие риски существуют при разработке ИС и как их минимизировать?
6. Какие метрики качества ИС существуют?
7. Как подготовить проект к защите (документация, презентация, демо)?

Практические занятия (сквозной проект):

1. Выбрать тему итогового проекта (например, «Система управления задачами», «Блог с комментариями», «Каталог книг», «Система бронирования переговорных»).
2. Разработать техническое задание (ТЗ) и спроектировать БД (ER-диаграмма, SQL-схема).
3. Реализовать REST API на Node.js + Express + Sequelize (не менее 5 сущностей, все CRUD-операции).
4. Создать документацию API в Swagger (OpenAPI).
5. Написать Postman-тесты для всех эндпоинтов.
6. Разработать фронтенд на Vue.js (не менее 5 компонентов, маршрутизация, подключение к API).
7. Написать Dockerfile и docker-compose.yml для контейнеризации проекта.
8. Развернуть проект на удалённом хостинге (Render/Railway).
9. Подготовить отчёт (15–20 страниц) и презентацию (10–12 слайдов) для защиты.

Рекомендуемые технологии и версии для выполнения практических заданий

Компонент	Рекомендация
Node.js	18.x LTS или 20.x LTS
Express	4.x
ORM	Sequelize 6.x
БД	PostgreSQL 14+ или MySQL 8+
Тестирование	Postman + Newman, Jest
Документация	Swagger (OpenAPI 3.0)
Шаблонизатор	Pug

Фронтенд	Vue.js 3 + Vite + Pinia + Vue Router
Контейнеризация	Docker + Docker Compose
Хостинг	Render / Railway / Heroku

5.3. Один или несколько тематических блоков дисциплины завершаются контрольной работой по разделу (далее – КР). Текущий контроль успеваемости по дисциплине предусматривает не менее 2 (двух) и не более 10 (десяти) КР в течение периода освоения дисциплины.

Максимальное количество баллов за любой тип работ в рамках КР составляет 100 (сто) баллов.

Распределение весовых коэффициентов по КР в рамках текущего контроля успеваемости по дисциплине и формулы расчета:

Наименование контрольной работы	Максимальное количество баллов за работу в рамках КР, которое может набрать студент	Коэффициент веса контрольной работы	Результат контрольной работы, участвующий в формировании итоговой балльной оценки по дисциплине
КР 1	100	0,1	20
КР 2	100	0,1	22
Итого:	x	0,2	x

Формула расчета результата контрольной работы:

Результат контрольной работы = Количество баллов за работу в рамках КР X Коэффициент веса контрольной работы.

5.4. Формы текущего контроля успеваемости обучающихся в рамках КР и типовые оценочные материалы:

КР-1

Раздел 1. Основы разработки информационных систем

Теоретические вопросы:

1. Компания разрабатывает интернет-магазин с высокой нагрузкой (до 10 000 запросов в секунду в час пик). Отдел разработки состоит из 5 команд, которые независимо разрабатывают модули: каталог, корзина, оплата, доставка, личный кабинет. Какую архитектуру следует выбрать и

почему? Какие проблемы могут возникнуть при выборе монолита в данном случае?

2. Спроектируйте схему базы данных для системы «Библиотека». Перечислите необходимые таблицы, их поля и связи между ними. Какой тип связи между таблицами «Читатели» и «Книги» (через «Выдача книг»)? Обоснуйте.

3. Напишите SQL-запрос для выборки всех книг, которые были выданы читателю с id = 5, но ещё не возвращены. Используйте таблицы: books(id, title), loans(id, book_id, reader_id, loan_date, return_date).

4. Что такое ORM? Назовите преимущества и недостатки использования ORM на примере Sequelize. В каком случае лучше написать сырой SQL-запрос вместо ORM?

5. Объясните разницу между PUT и PATCH в контексте REST API. Приведите примеры использования каждого метода для ресурса /users/{id}.

6. Что произойдёт, если в Node.js-приложении синхронная функция, выполняющая сложные вычисления, будет вызвана в основном потоке? Как это повлияет на обработку других запросов? Как правильно решить эту проблему?

Какие HTTP-коды ответа следует возвращать в следующих ситуациях REST API:

- Успешное создание ресурса (POST)
- Успешный запрос, но нет данных для возврата
- Ресурс не найден по ID
- Ошибка валидации входных данных
- Внутренняя ошибка сервера

Практическое задание:

Условие:

Разработать REST API для системы управления товарами интернет-магазина.

Требования:

1. Модель товара (Product):
   - id (уникальный идентификатор, автоинкремент)
   - name (строка, обязательное поле, длина от 3 до 100 символов)
   - price (число с плавающей точкой, > 0)
   - category (строка, одно из значений: 'electronics', 'clothing', 'books')
   - in_stock (булево значение, по умолчанию true)
2. Эндпоинты REST API:
   - GET /api/products — получение всех товаров. Поддержка фильтрации по category и in_stock, а также пагинации (page, limit).
   - GET /api/products/:id — получение товара по ID. При отсутствии — 404.
   - POST /api/products — создание нового товара. Валидация входных данных. При успехе — 201, возврат созданного объекта с присвоенным ID.

- PUT /api/products/:id — полное обновление товара. Все поля обязательны. При отсутствии товара — 404.
- PATCH /api/products/:id — частичное обновление (только переданные поля обновляются).
- DELETE /api/products/:id — удаление товара.

3. Технические требования:
   - Использовать Node.js + Express
   - В качестве базы данных — массив в памяти (для упрощения проверки) с демонстрацией CRUD
   - Настроить middleware для логирования и обработки ошибок
   - Добавить базовую валидацию
4. Результат выполнения: предоставить код всех файлов (app.js, routes/products.js, controllers/productsController.js) + примеры запросов/ответов (скриншоты Postman или curl).

КР-2

Раздел 2. Разработка ИС с использованием фреймворков

Теоретические вопросы:

1. В чём разница между unit-тестами, интеграционными тестами и E2E-тестами? Приведите примеры для REST API на Node.js.
2. Опишите структуру OpenAPI 3.0 спецификации. Какие основные разделы должны быть в ней для документирования REST API?
3. Что такое реактивность в Vue.js 3? В чём разница между ref и reactive? Приведите примеры использования.
4. Какие преимущества даёт использование Docker Compose в разработке и развёртывании ИС? Напишите пример docker-compose.yml для приложения: бэкенд (Node.js) + БД (PostgreSQL).
5. Что такое Vue Router? Как реализовать динамическую маршрутизацию с параметрами (например, /tasks/:id)? Приведите код.
6. В чём отличие серверного рендеринга (SSR) с использованием шаблонизатора Pug от клиентского рендеринга (SPA на Vue.js)? Назовите сценарии, где каждый подход предпочтительнее.
7. Что такое CI/CD? Опишите простейший пайплайн для Node.js-приложения (тесты → сборка → деплой на хостинг). Какие инструменты можно использовать?

Практическое задание:

Условие:

Разработать простое SPA-приложение на Vue.js 3, которое взаимодействует с REST API из КР-1 (управление товарами). Допускается эмуляция API через mock-данные, но лучше использовать реальное API.

Требования:

1. Компоненты приложения:

- ProductList.vue — отображает список товаров в виде карточек. Каждая карточка содержит: название, цену, категорию, статус (в наличии/нет), кнопки «Редактировать» и «Удалить».
- ProductForm.vue — форма для создания и редактирования товара (поля: name, price, category (select), in_stock (checkbox)).
- ProductDetail.vue — отдельная страница с полной информацией о товаре (опционально).

2. Функциональность:
   - Загрузка списка товаров с API (GET /api/products)
   - Добавление товара (POST /api/products) — через форму
   - Редактирование товара (PUT или PATCH) — через форму, предзаполненной данными выбранного товара
   - Удаление товара (DELETE /api/products/:id) — с подтверждением (диалог)
   - Фильтрация товаров по категории (select) и по наличию (checkbox)
3. Маршрутизация (Vue Router):
   - / — список товаров
   - /add — форма добавления товара
   - /edit/:id — форма редактирования товара
   - /product/:id — страница детального просмотра (опционально)
4. Управление состоянием (Pinia):
   - Создать store для товаров с действиями: fetchProducts, addProduct, updateProduct, deleteProduct
5. Технические требования:
   - Использовать Composition API (рекомендуется)
   - Базовые стили (CSS или Tailwind)
   - Обработка ошибок API (toast или alert)
   - Индикатор загрузки при запросах
6. Результат выполнения: предоставить код проекта (структура с основными файлами) + скриншоты работающего приложения

6. Формы промежуточной аттестации, критерии и шкала оценивания, типовые оценочные материалы по дисциплине

6.1. Промежуточная аттестация по дисциплине «Разработка информационных систем» проводится в седьмом семестре в форме собеседования (2 теоретических вопроса + 1 практический/кейс). В восьмом семестре в форме экзамена в письменной форме. Обучающийся получает два теоретических вопроса и одно практическое задание.

6.2. Типовые оценочные материалы промежуточной аттестации.

Вопросы к зачёту:

Темы 1–2. Основы разработки ИС

1. Понятие информационной системы (ИС). Классификация ИС (OLTP, OLAP, CRM, ERP, CMS, ECM).
2. Архитектурные стили разработки ИС: монолитная, клиент-серверная, многоуровневая, микросервисная. Сравнение.
3. Жизненный цикл разработки ПО. Модели Waterfall и Agile: преимущества и недостатки.
4. Типы приложений: десктопные, мобильные (нативные, кроссплатформенные), веб-приложения (SPA, SSR, статические).
5. Реляционные базы данных: нормализация, первичные и внешние ключи, типы связей (1:1, 1:N, N:M).
6. Нереляционные базы данных: документоориентированные (MongoDB), графовые, ключ-значение. Сценарии использования.
7. Основы SQL: SELECT, INSERT, UPDATE, DELETE, JOIN (INNER, LEFT), GROUP BY, агрегатные функции.
8. Что такое индексы в БД? Зачем нужны, какие проблемы могут возникнуть при их использовании?
9. Транзакции и ACID. Зачем нужны транзакции в ИС?
10. Проектирование схемы БД: ER-диаграмма, нормальные формы (1НФ, 2НФ, 3НФ).
11. ORM (Object-Relational Mapping): понятие, преимущества и недостатки.
12. Sequelize: установка, подключение к БД, определение моделей (define).
13. Миграции в Sequelize: зачем нужны, как создать и выполнить миграцию.
14. Сиды (seeders) в Sequelize: для чего используются?
15. CRUD-операции через Sequelize: create, findAll, findByPk, update, destroy.

16. Связи между моделями в Sequelize: belongsTo, hasMany, belongsToMany.

17. Сложные запросы в Sequelize: условия (where), сортировка (order), пагинация (limit, offset).

18. Сырой SQL-запрос через Sequelize: когда это оправдано?

Темы 3–4. Бэкенд-разработка на Node.js

19. Платформа Node.js: особенности (асинхронность, событийная модель, однопоточность).

20. Модульная система в Node.js: CommonJS (require) vs ES modules (import/export).

21. Асинхронное программирование в JavaScript: колбэки, промисы (Promise), async/await.

22. Что такое callback hell? Способы борьбы с ним.

23. Фреймворк Express.js: установка, базовый сервер, маршрутизация (app.get, app.post, app.use).

24. Middleware в Express: что это такое, как работает, примеры встроенных middleware.

25. Обработка параметров запроса в Express: req.params, req.query, req.body.

26. Организация маршрутов с помощью express.Router(). Зачем это нужно?

27. Переменные окружения (dotenv). Зачем выносить настройки в .env?

28. Обработка ошибок в Express: 404, глобальный error handler.

Тема 4. REST API

29. Что такое REST API? Основные принципы REST (клиент-сервер, отсутствие состояния, единообразие интерфейса).

30. HTTP-методы в REST API: GET, POST, PUT, PATCH, DELETE. Для чего используется каждый?

31. HTTP-коды ответа: группы 2xx, 3xx, 4xx, 5xx. Примеры кодов.

32. Что такое идемпотентность? Какие HTTP-методы являются идемпотентными?

33. Проектирование RESTful эндпоинтов для CRUD-операций над ресурсом «Пользователи».

34. Версионирование API: способы (URL, заголовки, параметры запроса). Примеры.

35. Валидация входных данных в REST API. Какие коды ошибок возвращать при неверных данных?

36. Пагинация и фильтрация в REST API. Реализация через query-параметры.

37. Форматы передачи данных в API: JSON, XML. Почему JSON стал стандартом?

38. Безопасность REST API: базовые принципы (CORS, HTTPS, rate limiting).

Практические вопросы (кейсы):

1. Как спроектировать БД для интернет-магазина? Перечислите основные таблицы и связи.
2. В чём разница между монолитной и микросервисной архитектурой? Когда что выбирать?
3. Как реализовать пагинацию в REST API? Приведите пример запроса и ответа.
4. Как обработать ошибку «товар не найден» в Express API? Какой код ответа?
5. Почему в REST API не рекомендуется использовать глаголы в URL (например, /getUser)?
6. Как защитить API от слишком частых запросов (rate limiting)?
7. Что такое CORS? Почему возникает ошибка CORS в браузере при запросе к API?
8. Как передать JWT-токен в запросе к API? Где его хранить на клиенте?
9. Как организовать логирование запросов в Express-приложении?
10. Как протестировать API вручную? Какие инструменты существуют?
11. Какие метрики качества API существуют (время ответа, доступность)?

Вопросы к экзамену:

Тема 5. Тестирование API

1. Виды тестирования ПО: unit, интеграционное, E2E, нагрузочное. Определения и цели.
2. Postman: создание коллекций, переменные окружения, pre-request scripts.
3. Написание тестов в Postman (pm.test, pm.expect). Примеры тестов для проверки статус-кода и структуры ответа.
4. Автоматизация Postman-тестов: Newman, интеграция с CI/CD.
5. Unit-тестирование на Node.js с Jest: структура теста (describe, it, expect).
6. Тестирование API с использованием Supertest. Пример теста для Express-маршрута.
7. Mock-объекты и заглушки (stubs) в тестировании. Зачем нужны?

Тема 6. Документирование API

8. Документирование ИС: виды документации (техническая, пользовательская, API).
9. Язык разметки Markdown: основные элементы (заголовки, списки, таблицы, ссылки, код).

10. Swagger (OpenAPI): что это такое, структура спецификации OpenAPI 3.0.
11. Основные разделы OpenAPI: info, servers, paths, components.schemas.
12. Swagger UI: интеграция в Express-приложение (swagger-ui-express, swagger-jsdoc).
13. YAML vs JSON для описания спецификации. Преимущества YAML.
14. Документирование ошибок API в Swagger. Пример описания ответа 400.

Тема 7. Шаблонизация и SSR

15. Шаблонизаторы в веб-разработке: назначение, примеры (Pug, EJS, Handlebars).
16. Шаблонизатор Pug: синтаксис (теги, атрибуты, идентификация, классы).
17. Наследование шаблонов в Pug: layout, extends, block. Пример.
18. Передача данных из Express в шаблон Pug.
19. Серверный рендеринг (SSR) vs клиентский рендеринг (SPA). Сравнение.
20. Когда использовать SSR, а когда SPA? Преимущества и недостатки.

Тема 8. Фронтенд на Vue.js

21. Vue.js 3: основные концепции (реактивность, компоненты, директива v-bind, v-model, v-for, v-if).
22. Разница между Composition API и Options API в Vue 3. Преимущества Composition API.
23. Реактивность в Vue 3: ref и reactive. В чём разница? Примеры.
24. Жизненный цикл компонента Vue: хуки (created, mounted, updated, unmounted).
25. Однофайловые компоненты (SFC): структура (template, script, style).
26. Маршрутизация в Vue 3: Vue Router. Настройка маршрутов, динамические параметры.
27. Навигация в Vue Router: <router-link>, useRouter(), useRoute().
28. Управление состоянием в Vue: Pinia (или Vuex). Store, state, actions, getters.
29. Взаимодействие с API из Vue: fetch или axios. Пример запроса внутри компонента.
30. Передача данных между компонентами: props, emit, provide/inject, Pinia.

Тема 9. Контейнеризация и DevOps

31. Контейнеризация: понятие, отличие контейнера от виртуальной машины.
32. Docker: Dockerfile (инструкции FROM, WORKDIR, COPY, RUN, EXPOSE, CMD).
33. Основные команды Docker: docker build, docker run, docker ps, docker stop, docker rm.
34. Docker Compose: файл docker-compose.yml (services, volumes, networks, depends_on).
35. Оркестрация нескольких сервисов в Docker Compose (бэкенд + БД + фронтенд).
36. CI/CD: понятие непрерывной интеграции и непрерывной доставки.
37. Простейший CI/CD пайплайн для Node.js-приложения (GitHub Actions).
38. Платформы для развёртывания: Render, Railway, Heroku, Vercel. Сравнение.

Тема 10. Итоговый проект и практические кейсы

39. Этапы разработки ИС: от сбора требований до развёртывания.
40. Техническое задание (ТЗ) на разработку ИС: структура и содержание.
41. MVP (Minimum Viable Product): понятие, зачем нужен?
42. Git Flow: основные ветки (main, develop, feature, release, hotfix).
43. Как организовать командную разработку с использованием Git?
44. Метрики качества ИС: производительность, надёжность, безопасность, удобство использования.
45. Обработка ошибок на клиенте и сервере: лучшие практики.
46. Оптимизация производительности Node.js-приложения (кластеризация, кеширование).
47. Логирование и мониторинг в продакшене (Winston, Morgan).
48. Безопасность веб-приложений: OWASP Top 10, основы (XSS, CSRF, SQL-инъекции).
49. Развёртывание ИС в облаке: настройка переменных окружения, управление секретами.
50. Защита экзаменационного проекта: структура презентации, демонстрация работы.

ПРИМЕР ЭКЗАМЕНАЦИОННОГО БИЛЕТА (8 семестр)

Билет №1

Теоретические вопросы (2 вопроса):

1. Опишите процесс проектирования REST API для системы управления заказами. Какие эндпоинты и HTTP-методы вы предложите для работы с заказами? Какие коды ответа будете использовать?

2. В чём отличие Vue Router от Pinia? Как организовать переход на страницу редактирования товара по маршруту /edit/:id и получение параметра id внутри компонента?

Практическое задание:

Разработайте компонент Vue.js, который:

- Загружает список пользователей из API GET /api/users при монтировании
- Отображает их в виде таблицы (id, name, email)
- При клике на кнопку «Удалить» отправляет запрос DELETE /api/users/:id и удаляет строку из таблицы без перезагрузки страницы
- Показывает индикатор загрузки во время выполнения запросов

(Разрешается писать псевдокод или реальный код Vue 3 Composition API)

6.3. Критерии и шкала оценивания на основе БРС.

Соответствие государственной шкалы оценивания академической успеваемости и шкалы ECTS при экзамене

Оценка по шкале ECTS	**Сумма баллов за все виды учебной деятельности**	**Оценка по государственной шкале**	**Определение**
A	90 – 100	«Отлично»	отличное выполнение с незначительным количеством неточностей
B	80 – 89	«Хорошо»	в целом правильно выполненная работа с незначительным количеством ошибок (до 10%)
C	75 – 79		в целом правильно выполненная работа с незначительным количеством ошибок (до 15%)
D	70 – 74	«Удовлетворительно»	неплохо, но со значительным количеством недостатков
E	60 – 69		выполнение удовлетворяет минимальные критерии
FX	35 – 59	«Не удовлетворительно»	с возможностью повторной сдачи
F	0 – 34		с обязательным повторным изучением дисциплины (выставляется комиссией)

6.4. Описание дополнительных материалов и оборудования, необходимых для выполнения проверочных заданий

Компьютер с операционной системой RedOS или Windows с устойчивым Интернет-соединением для работы с удалёнными web-сервисами, программные продукты с открытой лицензией: Visual Studio Code, Jupyter Notebook, Node.js + Express, Postman., Docker, Pinia, Sequelize, Vue.js.

7. Методические материалы по освоению дисциплины

Получение углубленных знаний по изучаемой дисциплине достигается за счет дополнительных часов к аудиторной работе самостоятельной работы студентов. Выделяемые часы целесообразно использовать для знакомства с дополнительной научной литературой по проблематике дисциплины, анализа научных концепций и современных подходов к осмыслению рассматриваемых проблем. К самостоятельному виду работы студентов относится работа в библиотеках, в электронных поисковых системах и т.п. по сбору материалов, необходимых для проведения практических занятий или выполнения конкретных заданий преподавателя по изучаемым темам. Студенты могут установить диалог с преподавателем, получать консультации по выполнению заданий. В качестве оценочных средств на протяжении семестра используются практические задания.

Обучение по дисциплине «Разработка информационных систем» предполагает изучение курса на аудиторных занятиях (лекции, практические занятия) и самостоятельную работу студентов. Практические занятия дисциплины предполагают их проведение в различных формах с целью выявления полученных знаний, умений, навыков и компетенций с проведением контрольных мероприятий. С целью обеспечения успешного обучения студент должен готовиться к лекции, поскольку она является важнейшей формой организации учебного процесса, поскольку:

– знакомит с новым учебным материалом;
– разъясняет учебные элементы, трудные для понимания;
– систематизирует учебный материал;
– ориентирует в учебном процессе.

Работа обучающегося на лекции:

Слушание и запись лекций – сложный вид вузовской аудиторной работы. Внимательное слушание и конспектирование лекций предполагает интенсивную умственную деятельность обучающегося. Краткие записи лекций, их конспектирование помогает усвоить учебный материал. Конспект

является полезным тогда, когда записано самое существенное, основное и сделано это самим обучающимся.

Подготовка к практическим занятиям:

Подготовку к каждому практическому занятию каждый обучающийся должен начать с ознакомления с планом, который отражает содержание предложенной темы. Тщательное продумывание и изучение вопросов плана основывается на проработке текущего материала лекции, а затем изучения обязательной и дополнительной литературы, рекомендованную к данной теме. Если программой дисциплины предусмотрено выполнение практического задания, то его необходимо выполнить с учетом предложенной инструкции. Все новые понятия по изучаемой теме необходимо внести в глоссарий, который целесообразно вести с самого начала изучения курса. Результат такой работы должен проявиться в способности обучающегося свободно ответить на теоретические вопросы практического занятия, его выступлении и участии в коллективном обсуждении вопросов изучаемой темы, правильном выполнении практических заданий и контрольных работ.

Структура практического занятия:

В зависимости от содержания и количества отведенного времени на изучение каждой темы может практическое занятие состоять из четырех-пяти частей:

1. Устный опрос.

2. Обсуждение теоретических вопросов, определенных программой дисциплины.

3. Выполнение практических заданий с последующим разбором полученных результатов или обсуждение практического задания, выполненного дома.

4. Подведение итогов занятия.

Работа с литературными источниками:

В процессе подготовки к практическим занятиям, обучающимся необходимо обратить особое внимание на самостоятельное изучение рекомендованной учебно-методической (а также научной и популярной) литературы. Самостоятельная работа с учебниками, учебными пособиями, научной, справочной и популярной литературой, материалами периодических изданий и Интернета, статистическими данными является наиболее эффективным методом получения знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у обучающихся свое отношение к конкретной проблеме. Более глубокому раскрытию вопросов способствует знакомство с дополнительной литературой, рекомендованной преподавателем, что позволяет обучающимся проявить свою

индивидуальность в рамках выступления на занятиях, выявить широкий спектр мнений по изучаемой проблеме.

8. Учебная литература и ресурсы информационно-телекоммуникационной сети Интернет

8.1. Основная литература

1. Калиберда, Е. А. Разработка web-приложений : учебное пособие / Е. А. Калиберда, К. В. Кравченко. — Омск : ОмГТУ, 2023. — 100 с. — ISBN 978-5-8149-3679-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/421766 (дата обращения: 15.05.2026). — Режим доступа: для авториз. пользователей.
2. Сергеева, О. А. Программирование на Python : учебно-методическое пособие / О. А. Сергеева. — Кемерово : КемГУ, 2024. — 157 с. — ISBN 978-5-8353-3123-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/420758 (дата обращения: 13.05.2026). — Режим доступа: для авториз. пользователей.

8.2. Дополнительная литература

3. Богатенков, С. А. Разработка сайтов и web-приложений : учебное пособие для вузов / С. А. Богатенков, Е. В. Бунова, В. В. Костерин. — Санкт-Петербург : Лань, 2025. — 96 с. — ISBN 978-5-507-52835-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/502463 (дата обращения: 15.05.2026). — Режим доступа: для авториз. пользователей.
4. Заяц, А. М. Основы WEB технологий. Разработка WEB-приложений современными инструментальными средствами : учебно-методическое пособие / А. М. Заяц. — Санкт-Петербург : СПбГЛТУ, 2021. — 116 с. — ISBN 978-5-9239-1269-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/191164 (дата обращения: 15.05.2026). — Режим доступа: для авториз. пользователей.

8.3.Нормативные правовые документы и иная правовая информация

1. Конституция Российской Федерации. – Текст : электронный // Сайт Президента Российской Федерации. – URL: http://www.kremlin.ru/acts/constitution

8.4 Интернет-ресурсы

1. Информационно-правовой портал ГАРАНТ.РУ. – URL: https://www.garant.ru/

2. Научная электронная библиотека eLIBRARY.RU. – URL: https://elibrary.ru/

3. Научная электронная библиотека «КиберЛенинка». – URL: https://cyberleninka.ru

4. Электронно-библиотечная система «Лань». – URL: http://e.lanbook.com

5. Документация Node.js – URL: https://nodejs.org

6. Документация Express.js – URL: https://expressjs.com

7. Документация Vue.js – URL: https://vuejs.org

8. Документация Docker – URL: https://docs.docker.com

9. Swagger OpenAPI Specification – URL: https://swagger.io/specification/

9. Материально-техническая база, информационные технологии, программное обеспечение и информационные справочные системы

Материально-техническое обеспечение дисциплины включает в себя:

- лекционные аудитории, оборудованные видеопроекционным оборудованием для презентаций, средствами звуковоспроизведения, экраном;
- помещения для проведения практических занятий, оборудованные учебной мебелью.

Дисциплина поддержана соответствующими программными продуктами с открытой лицензией: Visual Studio Code, Jupyter Notebook, Node.js + Express, Postman., Docker, Pinia, Sequelize, Vue.js.

Вуз обеспечивает каждого обучающегося рабочим местом в компьютерном классе в соответствии с объемом изучаемых дисциплин, обеспечивает выход в сеть Интернет.

Помещения для самостоятельной работы обучающихся включают следующую оснащенность: столы аудиторные, стулья, доски аудиторные, компьютеры с подключением к локальной сети института (для компьютерных аудиторий) и Интернет. Для изучения учебной дисциплины используются автоматизированная библиотечная информационная система и электронные библиотечные системы.