Приложение 4
к образовательной программе

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.О.05 Методология и технология проектирования информационных систем
(индекс, наименование дисциплины в соответствии с учебным планом)

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

Корпоративные информационные системы
(наименование образовательной программы)

Магистр
(квалификация)

Очная форма обучения
(форма обучения)

Год набора – 2026
Донецк

Автор(ы)-составитель(и) РПД:

Литвак Елена Геннадиевна, канд. экон. наук, доцент кафедры информационных технологий

Заведующий кафедрой:

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

Рабочая программа дисциплины Б1.О.05 Методология и технология проектирования информационных систем Базы данных одобрена на заседании кафедры информационных технологий 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

Дисциплина Б1.О.05 Методология и технология проектирования 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 ****
-	ОПК-5	Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем	ОПК-5.1;	Проектирует и разрабатывает программное обеспечение на основе современных средств	ОПК-5.1. З-1 **Знает** базовые принципы алгоритмизации и структур данных; Основные этапы жизненного цикла программного обеспечения. ОПК-5.1. У-1 **Умеет** создавать консольные приложения и простые скрипты; создавать простые UML-диаграммы (например, классов) для описания проектируемого модуля.
			ОПК-5.2	Разрабатывает и модернизирует программное обеспечение информационных и автоматизированных систем	ОПК-5.2. З-1 **Знает** структуру и архитектуру типовой информационной системы (клиент-сервер, файл-сервер); основы организации промышленных баз данных (СУБД) и языков запросов. ОПК-5.2. У-1 **Умеет** читать и понимать существующий код (legacy code) на используемом языке программирования; вносить локальные исправления в работающую систему по заданной спецификации; выполнять развертывание готового

					решения на тестовом или промышленном стенде

** 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:

9,00 з.е., 324 ак.час

Контактная работа обучающихся с преподавателем по видам учебных занятий: 112 ак. час на контактную работу с преподавателем, из них 36 ак. час на лекции и 54 ак. час на практические занятия. 167 ак. час на самостоятельную работу обучающихся.

Б1.О.05 Методология и технология проектирования информационных систем реализуется на 1-м и 2-м семестре 1-го курса после изучения дисциплин:

- Проектирование информационных систем.

.

3. Содержание и структура дисциплины

3.1. Структура дисциплины

Очная форма обучения

№ п/п	Наименование тем и (или) разделов	ВСЕГО	Объем дисциплины, ак.час												Форма текущего контроля успеваемости, промежуточной аттестации
			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. КЛАССИЧЕСКИЕ СТИЛИ ИНТЕГРАЦИИ															
Тема 1	Интеграция через файлы и общую базу данных	11	2	0	0	4	0	0	0	0		0	0	5	Контрольные вопросы, практические занятия, КТ1
Тема 2	Удалённый вызов процедур и очереди сообщений	11	2	0	0	4	0	0	0	0		0	0	5	Контрольные вопросы, практические занятия, КТ1

Тема 3	Сравнительный анализ и методология выбора стиля	11	2			4								5	Контрольные вопросы, практические занятия, КТ1
РАЗДЕЛ 2. REST КАК СТИЛЬ ИНТЕГРАЦИИ															
Тема 4	HTTP и основы REST	11	2	0	0	4	0	0	0	0		0	0	5	Контрольные вопросы, практические занятия, КТ2
Тема 5	Проектирование REST API для интеграции	11	2	0	0	4	0	0	0	0		0	0	5	Контрольные вопросы, практические занятия, КТ2
Тема 6	Методология выбора между REST и очередями	11	2	0	0	4	0	0	0	0		0	0	5	Контрольные вопросы, практические занятия, КТ2
Промежуточная аттестация		29							2	9			18		**зачет**
Итого за семестр		**180**	18		**36**				2	9			18	97	
РАЗДЕЛ 3. АСИНХРОННЫЕ СТИЛИ И ПАТТЕРНЫ НАДЁЖНОСТИ															
Тема 7	Вебхуки и событийно-ориентированная архитектура	11	2	0	0	4	0	0	0	0	0	0	0	5	Контрольные вопросы, практические занятия, КТ3

Тема 8	Паттерн «исходящая таблица» и паттерны надёжности: повтор, идемпотентность	11	2	0	0	4	0	0	0	0	0	0	0	5	Контрольные вопросы, практические занятия, КТ3
Тема 9	Автоматический выключатель, таймаут и изолирующий отсек	11	2	0	0	4	0	0	0	0	0	0	0	5	Контрольные вопросы, практические занятия, КТ3
РАЗДЕЛ 4. ИНТЕГРАЦИОННАЯ ИНФРАСТРУКТУРА И СКВОЗНОЕ ПРОЕКТИРОВАНИЕ															
Тема 13	Шаблоны корпоративной интеграции и шлюз API	12	4	0	0	4	0	0	0	0		0	0	4	Контрольные вопросы, практические занятия, КТ4
Тема 14	Интеграция как процесс: контракт и методология проектирования	12	4	0	0	4	0	0	0	0		0	0	4	Контрольные вопросы, практические занятия, КТ4
Тема 15	Сквозной проект: интеграционная архитектура системы	12	4	0	0	4	0	0	0	0		0	0	4	Контрольные вопросы, практические занятия, КТ4

Промежуточная аттестация	**38**							**2**	**9**		**9**	**18**		**Экзамен**
Итого за семестр	**144**	**18**			**18**			**2**	**9**		**9**	**18**	**70**	
Итого	**324**	**36**			**54**			**4**	**18**		**9**	**36**	**167**	

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

СЕМЕСТР 1. БАЗОВЫЕ СТИЛИ ИНТЕГРАЦИЙ И МЕТОДОЛОГИЯ ВЫБОРА

РАЗДЕЛ 1. КЛАССИЧЕСКИЕ СТИЛИ ИНТЕГРАЦИИ

Тема 1. Интеграция через файлы и общую базу данных. ОПК-5.1

Изучаются два «простых» исторически первых стиля: файловый обмен (CSV, XML, JSON как файлы, SFTP, проблемы рассинхрона, блокировок, необработанных файлов, паттерны «посадочная зона» и «папка мёртвых файлов») и прямой доступ к общей базе данных (shared database) - его механизмы, проблемы зацепления, блокировок, конфликтов версий схем и прав доступа. Студент должен научиться распознавать ситуации, где эти стили ещё приемлемы, а где они ведут к архитектурному коллапсу.

Тема 2. Удалённый вызов процедур и очереди сообщений. ОПК-5.1

Изучается удалённый вызов процедур (RPC) как идея «вызова удалённой функции» на примерах XML-RPC и gRPC, её достоинства (простота модели) и фундаментальные недостатки (скрытые сетевые задержки, отказы как исключения). Затем вводятся очереди сообщений: отправитель, очередь, получатель, гарантии доставки («не более раза», «не менее раза», «ровно один раз»), мёртвая очередь (DLQ) и проблема порядка сообщений. Студент должен понимать, почему очереди - это не «просто ещё один стиль», а принципиально иная модель взаимодействия.

Тема 3. Сравнительный анализ и методология выбора стиля. ОПК-5.1

Строится единая таблица сравнения стилей (файлы, общая БД, RPC, очереди) по 5–6 критериям: синхронность/асинхронность, надёжность доставки, сложность отладки, необходимость согласования форматов, поведение при отказе получателя. Студент учится пошаговой методике выбора: сначала определить, допустима ли задержка ответа (асинхронность), затем - нужна ли гарантированная доставка, затем - какова цена протокола. Итог - обоснованный выбор стиля для трёх разных бизнес-контуров.

РАЗДЕЛ 2. REST КАК СТИЛЬ ИНТЕГРАЦИИ

Тема 1. HTTP и основы REST. ОПК-5.1

С нуля вводится HTTP как транспорт интеграций: методы (GET, POST, PUT, DELETE, PATCH), группы кодов ответа (2xx, 3xx, 4xx, 5xx), ключевые

заголовки (Content-Type, Accept, Authorization). Затем объясняется идиома REST: ресурсы как существительные, отсутствие состояния на сервере (почему это требование и когда его нарушают), идемпотентность методов (чем PUT отличается от POST в условиях сбоя), роль HATEOAS (с честным признанием, что в реальных проектах почти не реализуется). Студент должен отличать REST от «просто CRUD через HTTP».

Тема 2. Проектирование REST API для интеграции. ОПК-5.1

Изучаются практические вопросы проектирования API как контракта интеграции: версионирование (в URL, в заголовке, через согласование содержимого), пагинация (на основе курсора vs на основе смещения), фильтрация и сортировка запросов, стандартный формат ошибок. Также вводятся политики ограничения частоты запросов и стратегии повторных вызовов для синхронных вызовов. Студент должен спроектировать небольшой REST API для внешнего потребителя с учётом всех перечисленных аспектов.

Тема 3. Методология выбора между REST и очередями. ОПК-5.1

Рассматриваются граничные зоны, где и REST, и очереди формально применимы. Строится таблица принятия решений: REST выбирается при необходимости немедленного ответа, при низких требованиях к гарантии доставки, при работе с внешними партнёрами; очереди - при высокой нагрузке, при необходимости доставки «ровно один раз», при толерантности к задержке. Студент получает сквозной кейс (например, интернет-магазин: каталог товаров, оформление заказа, отправка писем, интеграция с платёжным шлюзом) и должен для каждого интеграционного контура обоснованно выбрать стиль, защитив своё решение перед группой.

СЕМЕСТР 2. НАДЁЖНОСТЬ, ИНФРАСТРУКТУРА И СКВОЗНОЕ ПРОЕКТИРОВАНИЕ

РАЗДЕЛ 3. АСИНХРОННЫЕ СТИЛИ И ПАТТЕРНЫ НАДЁЖНОСТИ

Тема 1. Вебхуки и событийно-ориентированная архитектура. ОПК-5.2

Вводятся вебхуки как механизм обратного вызова: регистрация callback-URL, подпись запросов для проверки отправителя, стратегии повторной доставки при недоступности получателя. Событийно-ориентированная архитектура (EDA): разграничение понятий «событие» (факт, который уже произошёл) и «сообщение» (команда), роль брокера событий / событийной шины, принцип независимости издателей и подписчиков, проблема дублирования и порядка событий. Проектирование схемы, где одно событие (например, «заказ создан») обрабатывается тремя независимыми подписчиками.

Тема 2. Паттерн «исходящая таблица» и паттерны надёжности: повтор, идемпотентность. ОПК-5.2

Изучается паттерн «исходящая таблица» (outbox) - способ гарантированной публикации событий при сохранении в базу данных: событие сохраняется в таблицу outbox в одной транзакции с бизнес-данными, а отдельный процесс отправляет его в очередь или брокер. Затем вводятся паттерны надёжности: экспоненциальный повтор с задержкой (когда повторять, а когда не повторять), ключи идемпотентности (сквозная идемпотентность через уникальный ключ запроса) и проблема «шторма повторов» (когда массовые повторы убивают систему). Студент должен понять, что надёжность интеграции не добавляется «сверху», а проектируется в схеме данных.

Тема 3. Автоматический выключатель, таймаут и изолирующий отсек. ОПК-5.2

Рассматриваются три взаимодополняющих паттерна защиты синхронных интеграций: таймаут (таймаут соединения vs таймаут чтения, почему таймаут - это проектный параметр, а не случайная цифра), автоматический выключатель (состояния замкнут/разомкнут/полуоткрыт, его поведение при временных и постоянных отказах, разница между синхронным и асинхронным применением) и изолирующий отсек (изоляция потребителей одного пула соединений, чтобы отказ одного сервиса не ломал всю интеграцию). Студент должен на схеме интеграции указать точки применения каждого паттерна и объяснить, что произойдёт без них.

РАЗДЕЛ 4. ИНТЕГРАЦИОННАЯ ИНФРАСТРУКТУРА И СКВОЗНОЕ ПРОЕКТИРОВАНИЕ

Тема 1. Шаблоны корпоративной интеграции и шлюз API. ОПК-5.2

Вводятся ключевые шаблоны интеграции уровня систем без привязки к конкретной реализации: маршрутизатор сообщений (маршрутизация по содержимому), расщепитель (разбиение одного сообщения на несколько), агрегатор (сборка нескольких сообщений в одно), обогатитель (обогащение сообщения данными из внешнего источника). Затем - шлюз API как единая точка входа для внешних интеграций: его функции (маршрутизация, агрегация вызовов, трансформация протоколов, завершение SSL), отличие от бэкенда для фронтенда (BFF). Студент должен нарисовать схему, где шлюз API скрывает сложность трёх внутренних систем от внешнего партнёра.

Тема 2. Интеграция как процесс: контракт и методология проектирования. ОПК-5.2

Рассматривается место интеграций в жизненном цикле проектирования

ИС: как собирать интеграционные требования (кто, куда, с какой периодичностью, с какими гарантиями), как документировать контракт интеграции (форматы, схемы, SLA по доступности и задержке, поведение при сбоях), как оценивать сложность интеграции до начала реализации. Студент учится отличать «интеграцию как подзадачу» от «интеграции как отдельного проекта» (например, при стыковке с унаследованной системой без документации) и знакомится с шаблоном интеграционного контракта.

Тема 3. Сквозной проект: интеграционная архитектура системы. ОПК-5.2

Финальная проектная сессия курса: студенты получают сквозной кейс (например, интернет-магазин + платёжный шлюп + CRM + складская система + маркетплейс + внешний API проверки рецептов). Задача - спроектировать интеграционную архитектуру целиком: для каждого стыка выбрать стиль интеграции (файлы, очередь, REST, вебхуки, событийно-ориентированная архитектура), указать применяемые паттерны надёжности (повтор, автоматический выключатель, исходящая таблица, мёртвая очередь), решить, нужен ли шлюз API, и зафиксировать контракты. Защита проекта включает ответ на вопрос «Что сломается первым при росте нагрузки в 10 раз и почему?». Студент должен показать не знание одного фреймворка, а системное мышление в проектировании интеграций.

4. Типы оценочных материалов, показатели и критерии оценивания

4.1. Оценочные материалы по дисциплине Б1.О.05 Методология и технология проектирования 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 последовательности	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 Донецкого филиала РАНХиГС.

Итоговая балльная оценка	**Традиционная система**	**Бинарная система**	**ECTS**	
			Для традиционной системы	**Для бинарной системы**
90-100	Отлично	Зачтено	A	P/ Passed
80-89	Хорошо		B	P/ Passed
75-79			C	P/ Passed
70-74	Удовлетворительно		B	P/ Passed
60-69			E	P/ Passed
0-59	Неудовлетворительно	Не зачтено	F	F/Failed

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

В ходе реализации дисциплины Б1.О.05 Методология и технология проектирования информационных систем и права 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)

Раздел/Темы	Формы текущего контроля		КТ
	УО	ПЗ	
Р-1. / Т-1	5	5	20
Р-1. / Т-2	5	5	
Р-1. / Т-3	5	5	
Р-2. / Т-4	5	5	20
Р-2. / Т-5	5	5	
Р-2. / Т-6	5	5	
Итого: 100 б	30	30	40
Р-3. / Т-7	5	5	20
Р-3. / Т-7	5	5	
Р-3. / Т-7	5	5	
Р-4. / Т-8	5	5	20
Р-4. / Т-8	5	5	
Р-5. / Т-8	5	5	
Итого: 100 б	30	30	40

УО – устный опрос;
ТЗ – тестовое задание;
КЗ – контрольные задания;
ПЗ – практическое занятие;
Д – доклад;
КТ – контрольные точки.

Критерии оценивания опроса:

Балы	Описание критерия
4-5	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.
2-3	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	
4-5	Свыше 90% правильных ответов.	Обучающийся демонстрирует глубокое познание в освоенном материале.

2-3	Свыше 70% правильных ответов.	Обучающимся материал освоен полностью, без существенных ошибок.
1	Реализовано более 50% поставленных задач	Обучающимся материал освоен не полностью, имеются значительные пробелы в знаниях.
0	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
18-20	Обучающимся задание выполнено без ошибок и в полном объеме.
15-17	Обучающимся в целом задание выполнено, имеются отдельные неточности или недостаточно полные ответы, не содержащие ошибок.
10-14	Обучающимся допущены отдельные ошибки при выполнении задания
0-9	У обучающегося отсутствуют ответы на большинство вопросов задачи, задание не выполнено или выполнено не верно.

0* - в журнал академической группы не выставляется

5.1. Типовые оценочные материалы для текущего контроля успеваемости обучающихся (вне контрольных точек):

СЕМЕСТР 1. БАЗОВЫЕ СТИЛИ ИНТЕГРАЦИЙ И МЕТОДОЛОГИЯ ВЫБОРА

РАЗДЕЛ 1. КЛАССИЧЕСКИЕ СТИЛИ ИНТЕГРАЦИИ

Тема 1. Интеграция через файлы и общую базу данных

Контрольные вопросы:

1. В чём разница между проблемой «рассинхрон файлов» и проблемой «блокировки таблиц» при общей БД? Приведите пример, когда одна проблема переходит в другую.
2. Паттерн «папка мёртвых файлов» (DLQ для файлов) - это техническое или организационное решение? Почему его нельзя полностью автоматизировать?
3. Почему прямой доступ к общей базе данных считается антипаттерном, хотя технически работает? Назовите три конкретных сценария, когда он всё же оправдан.
4. Вы проектируете интеграцию двух систем. Одна команда настаивает на файлах (просто), другая - на общей БД (быстро). Какие два вопроса вы зададите каждой команде, чтобы выявить скрытые риски?
5. Что произойдёт, если в файловом обмене две системы

одновременно попытаются записать в один и тот же файл? Как спроектировать обмен, чтобы исключить такую ситуацию?

6. Какая информация, кроме самих данных, должна быть в интеграционном файле, чтобы приёмник мог обнаружить его повреждение или неполноту?

Практическое задание:

Вам даны: каталог `/incoming`, куда поставщик кладёт CSV-файлы с товарами (поля: `sku, name, price, stock`). Система должна прочитать файл, обновить локальную БД, а затем переместить файл в `/processed` (если успех) или `/failed` (если ошибка). Напишите на Laravel Artisan команду `php artisan import:products`, которая делает это. Учтите: файлов может быть несколько, обрабатывать нужно по одному. Ошибка в одной строке не должна убивать весь импорт.

Тема 2. Удалённый вызов процедур и очереди сообщений

Контрольные вопросы:

7. RPC скрывает сетевые вызовы как обычные функции. Почему это опасно для проектирования? Приведите пример кода, который выглядит безобидно, но на самом деле делает три сетевых вызова.

8. Гарантии доставки: «не более раза», «не менее раза», «ровно один раз». Какая из них самая дорогая в реализации и почему?

9. Что такое мёртвая очередь (DLQ) и зачем она нужна, если ошибку можно просто залогировать?

10. Очередь гарантирует доставку, но не гарантирует время. Как это влияет на проектирование пользовательского интерфейса? Приведите пример, когда пользователь нажал кнопку, а результат пришёл через минуту.

11. В чём принципиальная разница между отказом в RPC и отказом при работе с очередью? Как по-разному должна реагировать вызывающая сторона?

12. Можно ли поверх очереди реализовать RPC (запрос-ответ)? Если да, то какой ценой?

Практическое задание:

Реализуйте на Laravel: при создании заказа (POST /orders) нужно выполнить три действия: сохранить заказ в БД, отправить письмо клиенту, записать событие в лог-файл. Первое действие синхронное, второе и третье - через отдельные очереди (emails, logs). Настройте разные очереди и

покажите, что при падении обработчика письма (выбросить исключение) заказ всё равно сохраняется, а письмо уходит в failed_jobs. Напишите тест, который это проверяет.

Тема 3. Сравнительный анализ и методология выбора стиля

Контрольные вопросы:

13. Назовите критерий, по которому очереди всегда выигрывают у REST, и критерий, по которому REST всегда выигрывает у очередей.
14. Вы проектируете интеграцию с внешним платёжным шлюзом. Шлюз не поддерживает вебхуки, только синхронный запрос-ответ. Можно ли сделать эту интеграцию асинхронной с точки зрения вашего пользователя? Как?
15. Почему синхронные стили (REST, RPC) хуже масштабируются при росте числа участников интеграции? Приведите числовой пример (3 системы → 10 систем).
16. Методология выбора: сначала определяем, допустима ли задержка ответа. Почему это первый, а не второй или третий шаг?
17. Вам нужно интегрировать две системы с очень разными SLA (одна с 99.99% доступности, другая с 90%). Какой стиль интеграции вы выберете и почему?
18. Приведите пример, когда файловый обмен лучше очереди, и пример, когда очередь лучше файлового обмена, по критерию «сложность отладки проблем».

Практическое задание:

Вам дано описание системы интернет-магазина:

Каталог товаров (обновляется поставщиком раз в час)

Оформление заказа (пользователь ждёт подтверждения)

Отправка писем и смс (не критично к задержкам)

Интеграция с сервисом проверки реквизитов (внешний API, 200 мс, бывает недоступен)

Выгрузка отчётов в налоговую (раз в день, папка с файлами)

Для каждого из пяти контуров выберите стиль интеграции (файлы, общая БД, RPC, REST, очередь), напишите по одному предложению обоснования. Защитите выбор перед группой: ответьте на вопрос «Что будет, если этот контур упадёт на час?».

РАЗДЕЛ 2. REST КАК СТИЛЬ ИНТЕГРАЦИИ

Тема 4. HTTP и основы REST

Контрольные вопросы:

19. Чем PUT отличается от PATCH с точки зрения идемпотентности? Почему PATCH не обязан быть идемпотентным?
20. Вы получаете HTTP-ответ 429 Too Many Requests. Какие заголовки подскажут вам, когда можно повторить запрос?
21. Что значит «REST требует отсутствия состояния на сервере»? Сессия пользователя - это нарушение REST? А JWT-токен?
22. HATEOAS почти никто не реализует в реальных API. Значит ли это, что HATEOAS - бесполезная теория? Почему его включают в определение REST?
23. Вернёт ли идемпотентный метод разные ответы при одинаковых запросах? Если да, приведите пример.
24. Почему в REST не рекомендуется использовать метод GET для отправки тела запроса, хотя технически это возможно?

Практическое задание:

Создайте Laravel-приложение с одним ресурсом Task (поля: id, title, completed, created_at). Реализуйте REST-контроллер со всеми стандартными методами (index, show, store, update, destroy). В методе update используйте PUT (полное обновление) и PATCH (частичное). Напишите тесты (или проверьте в Postman), что:

PUT с неполными данными обнуляет отсутствующие поля
PATCH меняет только переданные поля
DELETE идемпотентен (второй вызов возвращает 404)
GET /tasks возвращает всегда свежие данные (без кэша)

Тема 5. Проектирование REST API для интеграции

Контрольные вопросы:

25. Версионирование в URL (/v1/orders) и в заголовке (Accept: version=2) - в чём плюсы и минусы каждого подхода для долгоживущей интеграции?
26. Пагинация на основе курсора (cursor-based) - чем принципиально отличается от пагинации на основе смещения (offset)? Когда первая необходима, а вторая опасна?
27. При rate limiting клиент получает 429. Как спроектировать

клиентскую логику повторных вызовов, чтобы не усугубить проблему?

28. Стандарт ошибок RFC 7807 (Problem Details for HTTP APIs) - какие поля он определяет и зачем нужно поле type (URI)?

29. Вы проектируете API для поиска. Как спроектировать фильтрацию по нескольким полям и сортировку, чтобы это не превратилось в «сделайте свой SQL в параметрах строки запроса»?

30. Почему для интеграций (система-система) лучше использовать application/json, а для публичных API часто предлагают application/vnd.api+json (JSON:API)?

Практическое задание:

На основе предыдущего API `Task` добавьте:

Пагинацию (offset/limit) с мета-полями `total`, `page`, `per_page`.

Фильтрацию по полю `completed` (`?completed=true/false`).

Сортировку по `created_at` (`?sort=created_at` или `?sort=-created_at`).

Rate limiting - 10 запросов в минуту на IP (используйте встроенный `throttle:api`).

Кастомный формат ошибки: при невалидных параметрах пагинации возвращать `{"error": "invalid_pagination", "message": "...", "hint": "..."}`.

Тема 6. Методология выбора между REST и очередями

Контрольные вопросы:

31. Приведите пример бизнес-процесса, где REST категорически не подходит, а очередь - идеальна. Объясните, чем именно REST навредит.

32. Обратная ситуация: пример, где очередь навредит (сделает систему хуже), а REST подходит.

33. Вы проектируете шлюз оплаты. Он принимает запросы по REST и возвращает синхронный ответ «платёж в обработке». Через 10 секунд платёжная система вызывает вебхук. Какой стиль используется на самом деле? Чистый REST?

34. Клиент (мобильное приложение) отправляет заказ. Нужно, чтобы заказ не потерялся даже при отказе сервера сразу после сохранения. REST или очередь? Почему?

35. В сложной системе есть 10 подсистем, которые должны получить событие «товар закончился». REST или очередь? Обоснуйте через количество связей.

36. Ваш начальник говорит: «Давайте всё сделаем через REST, это современно». Какие три аргумента вы приведёте, чтобы он передумал для части процессов?

37. Практическое задание:

Получите кейс (раздаётся на бумаге или в PDF):

Онлайн-школа: студент покупает курс → нужно провести платёж (внешний API, 2–5 сек), зачислить студента в группу (быстро), отправить письмо (можно не мгновенно), обновить CRM (ночью).

Школа ожидает рост до 1000 покупок в час.

Задание: нарисуйте схему интеграций (подсистемы и стрелки). У каждой стрелки подпишите стиль (REST или очередь) и одно предложение обоснования. Дополнительно: где нужен retry на уровне клиента, а где - мёртвая очередь? Защитите схему перед группой (5 минут).

СЕМЕСТР 2. НАДЁЖНОСТЬ, ИНФРАСТРУКТУРА И СКВОЗНОЕ ПРОЕКТИРОВАНИЕ

РАЗДЕЛ 3. АСИНХРОННЫЕ СТИЛИ И ПАТТЕРНЫ НАДЁЖНОСТИ

Тема 7. Вебхуки и событийно-ориентированная архитектура

Контрольные вопросы:

38. Вебхук - это push или pull? В чём главная проблема вебхуков по сравнению с регулярным опросом (polling)?
39. При регистрации вебхука клиент передаёт callback-URL. Как защититься от подмены? Что должен сделать сервер, чтобы убедиться, что URL действительно принадлежит клиенту?
40. Событие и сообщение - в чём разница? Можно ли реализовать события через очередь сообщений? Можно ли реализовать очередь сообщений через события?
41. В событийно-ориентированной архитектуре издатель не знает о подписчиках. Как тогда обеспечить, что критически важное событие точно кому-то доставили?
42. В вашей системе подписчики обрабатывают событие «заказ оплачен» в непредсказуемом порядке - оповещение склада может прийти раньше, чем запись в CRM. Как спроектировать логику, чтобы порядок не имел значения?
43. Что такое «брокер событий» (event broker) и чем он отличается от «очереди» с точки зрения модели потребления (один получатель против многих)?

Практическое задание:

На Laravel реализуйте эндпоинт для приёма вебхуков от внешней

платёжной системы: `POST /webhooks/payment`. Платежка присылает JSON с полями `payment_id`, `status` (success/failed), `signature` (HMAC-SHA256 от `payment_id+status` с секретным ключом). Ваша задача:

Проверить подпись, если не совпадает - вернуть 401.

При успешном платеже положить задачу в очередь `process_payment` с данными.

Вернуть 202 Accepted.

Напишите также Artisan-команду для эмуляции вебхука (отправить запрос из консоли). Покажите, что при недоступности очереди вебхук всё равно принимается (запись в лог, но ответ 202).

Тема 8. Паттерн «исходящая таблица» и паттерны надёжности: повтор, идемпотентность

Контрольные вопросы:

44. Outbox решает проблему «сохранил в БД, но очередь/брокер упал». Почему просто отправить событие после коммита транзакции небезопасно?
45. Идемпотентность: PUT идемпотентен, POST - нет. А как сделать идемпотентный POST? Какой механизм для этого нужен на стороне клиента и сервера?
46. Стратегия повторов (retry) с экспоненциальной задержкой - зачем нужен jitter (случайное отклонение)? Что произойдёт, если 1000 клиентов будут повторять запросы строго через 1, 2, 4, 8 секунд?
47. Вы видите в логах: «failed_jobs» забита тысячами записей. Что это значит для проектирования? Какие два вопроса вы зададите себе как архитектор?
48. Outbox требует фонового процесса для отправки событий. Как спроектировать этот процесс, чтобы он не мешал основным транзакциям? (подсказка: лимиты, паузы, приоритеты)
49. Клиент прислал запрос с идемпотентным ключом idem-123. Сервер обработал запрос, но в момент отправки ответа сеть оборвалась. Клиент повторяет запрос с тем же ключом. Что должен вернуть сервер? Как отличить «уже обработано» от «обрабатывается прямо сейчас»?

Практическое задание:

Реализуйте упрощённый outbox в Laravel:

Миграция `create_outbox_table` (id, event_type, payload, status, created_at).

При сохранении заказа (в контроллере `OrderController@store`) в одной DB-транзакции: сохранить заказ + добавить запись в outbox (status = 'pending').

Artisan-команда `php artisan outbox:send`, которая берёт pending-записи (по

5 штук, лимит), пытается отправить их в очередь (диспатчит Job) и меняет status на 'sent'. Если отправка упала - статус остаётся pending.

Напишите юнит-тест, который проверяет, что при падении очереди (мокаем `Bus::dispatch` с исключением) outbox-запись остаётся pending и не теряется.

Тема 9. Автоматический выключатель, таймаут и изолирующий отсек

Контрольные вопросы:

50. Таймаут соединения (connection timeout) и таймаут чтения (read timeout) - в чём разница? Для интеграции с медленным внешним API что нужно увеличить?
51. Автоматический выключатель (circuit breaker) переходит в открытое состояние после N ошибок. Как он узнаёт, что проблема решена? Что такое «полуоткрытое» состояние?
52. Почему автоматический выключатель не всегда хорош для асинхронных интеграций (очередей)? В каком случае он там всё же полезен?
53. Паттерн «изолирующий отсек» (bulkhead) в приложении - как он выглядит на уровне пулов соединений, потоков или очередей? Приведите пример конфигурации.
54. Вы выбрали таймаут в 5 секунд для внешнего API. Внешний API стал отвечать в среднем за 4.9 секунды, но иногда за 5.1. Что будет с вашей системой? Как изменить проектирование, чтобы этого избежать?
55. Circuit breaker применяется на клиенте. Как спроектировать серверную часть, чтобы она помогала клиенту корректно работать с circuit breaker'ом (например, через HTTP-заголовки Retry-After)?

Практическое задание:

В Laravel реализуйте клиент к внешнему API (заглушка - можно свою имитацию). Добавьте:

Таймаут соединения 2 сек, таймаут чтения 3 сек (`Http::timeout(3)->connectTimeout(2)`).

Стратегию повторов: 3 попытки, экспоненциальная задержка с jitter (используйте `Http::retry(3, 100, 200)` с функцией задержки).

Простейший circuit breaker на Redis (или в памяти на 2 минуты): считаем ошибки за последние 30 секунд, если >5 - возвращаем заглушку «сервис недоступен» без реального вызова.

Напишите тест, имитирующий: первые 6 вызовов падают (500), 7-й вызов не идёт на сеть (мгновенный ответ с ошибкой 503), через 2 минуты 8-й

вызов снова пытается уйти в сеть.

РАЗДЕЛ 4. ИНТЕГРАЦИОННАЯ ИНФРАСТРУКТУРА И СКВОЗНОЕ ПРОЕКТИРОВАНИЕ

Тема 10. Шаблоны корпоративной интеграции и шлюз API

Контрольные вопросы:

56. Маршрутизатор сообщений (message router) - чем он отличается от обычного if/else в коде? В каких случаях он становится отдельным компонентом, а не функцией?
57. Расщепитель (splitter): один заказ превращается в три товарные позиции для склада. Где в этой цепочке может быть проблема, если одна из трёх позиций не прошла валидацию? Как спроектировать поведение при частичном сбое?
58. Агрегатор должен собрать ответы от трёх сервисов. Два ответили быстро, третий - завис. Какие стратегии завершения агрегации существуют? (timeout, best-effort, all-or-nothing)
59. API Gateway - это «ещё одна точка отказа» или «упрощение, которое уменьшает количество точек отказа»? Аргументируйте.
60. BFF (backend for frontend) иногда называют «антипаттерном» в больших системах. Почему? Когда BFF всё же оправдан?
61. В чём разница между API Gateway в стиле «обычный маршрутизатор» и «умный» (с агрегацией, трансформацией)? Какая сложность ложится на клиента во втором случае?

Практическое задание:

Реализуйте упрощённый API Gateway на Laravel, который по запросу `GET /api/gateway/order/{id}` делает три параллельных вызова (или последовательных, если параллельность сложна) к трём внутренним эндпоинтам (например, /order/{id}, /customer/{id}, /shipment/{id}) и агрегирует их в один JSON. Используйте `Http::pool()` для параллельных запросов. Если один из сервисов недоступен, в ответе должно быть поле `errors` с указанием, какой сервис упал, но остальные данные - вернуть.

Тема 11. Интеграция как процесс: контракт и методология проектирования

Контрольные вопросы:

62. Что такое «контракт интеграции»? Какие разделы в нём обязательны, а какие опциональны?

63. SLA 99.9% означает, что система может быть недоступна ~40 минут в месяц. Как это требование влияет на проектирование интеграции? Какие паттерны становятся обязательными?

64. Вам нужно интегрироваться с legacy-системой, у которой нет документации, а разработчики уволены. Как спроектировать интеграцию? Какой главный риск и как снизить его вероятность?

65. Требование «источник истины» - что это значит при интеграции двух систем? Как быть, если обе системы считают себя источником?

66. При сборе интеграционных требований стейкхолдер говорит: «нужно, чтобы данные были всегда свежие». Как вы превратите это в измеримое требование к архитектуре?

67. В каком случае интеграция становится отдельным проектом, а не подзадачей основного проекта? Назовите три признака.

Практическое задание:

Возьмите любой интеграционный контур из финального проекта (тема 3) и составьте **одностраничный интеграционный контракт** в формате Markdown. Контракт должен включать:

Направление (система А → Б или двустороннее)

Стиль интеграции (REST, очередь и т.д.) и почему выбран именно он

Формат сообщения (пример JSON или структура файла)

SLA: ожидаемая задержка, допустимый процент отказов, время восстановления

Поведение при сбое (retry, circuit breaker, manual intervention)

Тестовые данные для интеграционного тестирования

Контракт сдаётся на проверку вместе со схемой.

Тема 12. Сквозной проект: интеграционная архитектура системы

Контрольные вопросы:

68. В вашей схеме вы выбрали REST для связи каталога и корзины. Что произойдёт, если каталог упадёт, когда пользователь добавляет товар в корзину? Спроектировано ли это в вашем решении?

69. Вы использовали очередь для ночной выгрузки в CRM. Какие механизмы гарантируют, что данные не потеряются, если сервер упадёт после отправки в очередь, но до того, как CRM их забрала?

70. Какая интеграция в вашем проекте самая ненадёжная с точки зрения точки отказа? Как вы уменьшили риск?

71. Назовите один компромисс, который вы сделали в проекте («пожертвовали X ради Y»). Была ли альтернатива?

72. Если нагрузка вырастет в 50 раз, какой компонент интеграционной архитектуры разрушится первым? Почему?

73. Ваш заказчик требует, чтобы все интеграции были синхронными «чтобы видеть статус сразу». Как вы аргументируете, почему это плохо? Останутся ли в вашем проекте синхронные интеграции? Какие именно?

Практическое задание:

Формулировка:

Спроектируйте интеграционную архитектуру для «Системы доставки готовых обедов».

Подсистемы:

- Сайт / мобильное приложение (приём заказов);
- Платёжный шлюз (внешний, с вебхуками);
- Складская система (остатки ингредиентов, резервирование);
- Кухня (производственный план);
- Доставка (API курьерской службы, синхронный);
- CRM (записывает историю заказов, ночная синхронизация);
- Налоговая (требует файл формата XML раз в день через SFTP).

Требования:

Заказ не должен теряться даже при отказе платежей или кухни.

При успешной оплате курьер должен быть вызван как можно быстрее.

Кухня работает по принципу FIFO, но важны лимиты ингредиентов (склад).

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

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

Распределение весовых коэффициентов по КТ 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
КТ 1	100	0,2	20
КТ 2	100	0,2	20
Итого:	x	0,4	40
КТ 3	100	0,2	20
КТ 4	100	0,2	20
Итого:	x	0,4	40

Формула расчета результата контрольной работы:

Результат контрольной работы = Количество баллов за точку в рамках КТ X Коэффициент веса контрольной точки.

5.3. Формы текущего контроля успеваемости обучающихся в рамках КТ и типовые оценочные материалы:

КТ 1

Задание 1.

Необходимо спроектировать интеграцию: при регистрации нового пользователя CRM должна отправить приветственное письмо. Потери письма недопустимы (юридическое требование), но задержка в 5–10 минут допустима.

Какую гарантию доставки вы выберете: «не более раза», «не менее раза» или «ровно один раз»? Почему?

Опишите, как должен вести себя обработчик письма, чтобы при сбое (например, отключение почтового сервера) сообщение не потерялось, но и не

отправилось дважды.

Что такое DLQ (Dead Letter Queue) и как она поможет, если письмо невозможно отправить принципиально (неверный email-адрес)?

Задание 2.

Вам нужно передавать данные о продажах из 50 магазинов в центральный офис. Требования:

Данные нужны раз в 15 минут, задержка в 30 минут приемлема.

Интернет в магазинах нестабилен, связь может пропадать на 10–20 минут.

В центральном офисе хотят минимальную сложность поддержки.

Заполните таблицу. Оцените каждый критерий как да / нет / с нюансами:

Критерий	Файлы (SFTP)	Очередь сообщений	REST API
Работа при потере связи	?	?	?
Подходит для задержки 15 мин	?	?	?
Простота поддержки	?	?	?

Какой стиль вы выберете и почему? Напишите 2–3 предложения обоснования.

Задание 3.

Интернет-магазин внедряет loyalty-сервис. При каждом заказе нужно:

Сохранить заказ в своей БД (магазин).

Отправить событие «заказ создан» в loyalty-сервис (внешний, работает через REST, иногда падает или медлит).

Если loyalty-сервис не ответил за 5 секунд или вернул ошибку - заказ всё равно должен сохраниться, а повторная попытка должна быть автоматической (не более 3 попыток).

Потеря события недопустима - пользователь может лишиться бонусов.

Какой основной стиль интеграции вы выберете (REST или очередь)? Дайте краткое обоснование (2–3 предложения).

Нарисуйте схему (от руки или текстовую диаграмму), показав:

Компоненты (магазин, БД, очередь/обработчик, loyalty-API)

Стрелки с указанием протокола

Где применяется retry (повтор) и где - DLQ

Опишите, что произойдёт при сценарии:

«Заказ сохранён в БД, событие положено в очередь, воркер начал отправку в loyalty-API, но в этот момент воркер упал (процесс убит)».

Не потеряется ли событие? Почему?

КТ 2

Дан фрагмент контроллера (псевдо-Laravel). API предназначено для интеграции с мобильным приложением.

```php
// routes/api.php
Route::post('/update-product',
[ProductController::class, 'update']);

// ProductController.php
public function update(Request $request) {
    $product = Product::find($request->id);
    $product->name = $request->name;
    $product->price = $request->price;
    $product->save();
    return response()->json(['status' => 'ok']);
}
```

Проблемы API:

- не использует http-методы правильно (всё через post);
- нет идемпотентности;
- нет версионирования;
- нет нормальной обработки ошибок (при несуществующем `id` возвращается пустой ответ? или ошибка?)

Укажите три нарушения принципов REST в этом коде. Для каждого напишите как должно быть (один вариант исправления).

Перепишите маршруты и метод контроллера (можно псевдокод на уровне ближе к Laravel), чтобы:

- использовались правильные http-методы;
- был хотя бы минимальный вариант версионирования (/v1/...);
- при отсутствии продукта возвращался 404 с JSON-ошибкой.

(Добавьте в исправленную версию механизм идемпотентности на основе заголовка Idempotency-Key. Опишите (текстом или псевдокодом):

Где и как хранить ключи идемпотентности?

Что вернуть при повторном запросе с тем же ключом, если первый запрос уже выполнен?

КТ 3

Вы проектируете систему оплаты в интернет-магазине. Используется внешний платёжный шлюз (API). Требования:

- при успешной оплате нужно: обновить статус заказа, отправить письмо клиенту, вызвать вебхук в crm магазина;
- платёжный шлюз иногда: (а) медлит (>10 секунд), (б) возвращает 500, (в) возвращает 429 (rate limit) с заголовком `retry-after`;

- потеря факта оплаты недопустима (регуляторное требование);
- система должна оставаться доступной, если платёжный шлюз полностью отказал (circuit breaker).

Нарисуйте схему интеграции, включив:

- синхронный вызов платежного шлюза;
- очередь/воркеры для последующих действий (обновление статуса, письмо, crm);
- outbox (если используете);
- circuit breaker - где он располагается?

Опишите алгоритм обработки платежа в **тексте или псевдокоде**, включая:

- что происходит в основной транзакции;
- как регистрируется факт оплаты до вызова шлюза;
- как обрабатывается таймаут или временная ошибка шлюза (re-try logic);
- что происходит при успехе / при окончательном отказе (после всех retry).

Ответьте на вопрос:

«Если платёжный шлюз вернул 200 (успех), но затем наш сервер упал до того, как мы обновили статус заказа - как система после перезапуска узнает, что оплата прошла?» Опишите механизм (опираясь на outbox или другой паттерн) и укажите, нужно ли делать запрос к шлюзу для сверки (reconciliation).

КТ 4

Сквозное проектирование интеграций для сервиса доставки еды.

Дана система «Доставка еды»:

Компоненты:

- Клиентское приложение (мобильное + веб)
- Сервис заказов - принимает заказ, проверяет ресторан, вызвает оплату
- Платёжный шлюз - внешний, через REST (вебхуки возврата)
- Сервис ресторанов - внешний (у каждого ресторана свой API, все разные, ненадёжные)
- Сервис логистики - внутренний, вызывает API курьерской службы (вызов доставки)
- Сервис уведомлений - отправляет push/SMS клиенту
- CRM - хранит историю заказов, загружает данные пачками раз в час

Требования:

- при оформлении заказа: оплата → подтверждение ресторану → вызов курьера → уведомление клиента

- если ресторан не подтвердил заказ за 2 минуты - отменить заказ и вернуть деньги
- платежи не должны теряться
- курьер вызывается только после подтверждения ресторана
- все внешние api ненадёжны, таймауты и ошибки - обычное явление.

Нарисуйте схему интеграций (компоненты и связи). Для каждой связи укажите:

- стиль (синхронный rest, очередь, вебхуки, файлы - где что уместно)
- один-два ключевых паттерна надёжности (retry, circuit breaker, outbox, DLQ)

Опишите сценарий обработки заказа в логической последовательности (10–15 пунктов), показывая:

- где используется очередь для асинхронной обработки
- где нужен таймаут с отменой (ресторан)
- как обеспечивается, что курьер не вызовется до подтверждения ресторана

Что произойдёт при падении сервиса заказов между оплатой и записью в лог?

Какой компонент в этой системе вы сделаете с Outbox Pattern и почему?

Если сервис ресторанов полностью не отвечает, нужно ли выключать его через circuit breaker, чтобы не блокировать клиентские заказы? Почему?

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

6.1. Промежуточная аттестация проводится в форме *зачета* в 1-м семестре и *экзамена* во 2-м семестре в письменной форме. Обучающийся получает экзаменационный билет с вариантами заданий. Обучающийся получает чистые маркированные листы бумаги для записей, затем приступает к выполнению. Необходимо дать ответ в письменном виде, подробно изложив ход мыслей.

6.2. Типовые оценочные материалы промежуточной аттестации.

РАЗДЕЛ 1. КЛАССИЧЕСКИЕ СТИЛИ ИНТЕГРАЦИИ

Тема 1.1. Интеграция через файлы и общую базу данных

Вопросы к экзамену:

1. Перечислите три основные проблемы файлового обмена и для каждой укажите один паттерн, который её решает.
2. Объясните, почему интеграция через общую БД считается антипаттерном, но технически успешно работает в небольших компаниях годами. В чём здесь компромисс?
3. Вам достался проект, где два отдела 5 лет интегрировались через общие таблицы в БД. Никто не помнит всех связей. Назовите три признака того, что этот подход пора менять (какие симптомы вы увидите в работе систем).
4. Предложите схему, как мигрировать с общей БД на файловый обмен без остановки обеих систем и без потери данных. Опишите три этапа миграции.

Практическое задание:

Вы получаете от поставщика CSV-файлы с товарами (поля: sku, name, price, stock). Файлы кладутся на SFTP каждые 30 минут. Ваша задача - спроектировать импортёр.

Задания:

1. Нарисуйте схему (компоненты, директории: `/incoming`, `/processing`, `/done`, `/failed`).
2. Опишите алгоритм обработки одного файла (5–7 шагов): чтение, валидация, обновление БД, перемещение.
3. Что произойдёт, если во время обработки файла упадёт сервер? Какая

строка из вашего алгоритма предотвратит потерю данных?

4. Напишите заглушку Laravel Artisan-команды (псевдокод), которая обрабатывает файлы из `/incoming` по одному.

Тема 1.2. Удалённый вызов процедур и очереди сообщений

Вопросы к экзамену:

1. Чем гарантия «не менее раза» отличается от «ровно один раз»? Какая из них дороже и почему?
2. RPC скрывает сеть. Приведите пример, когда разработчик пишет `userService.getUser(id)`, не подозревая, что это сетевой вызов, и какая катастрофа может произойти при большой нагрузке.
3. Вы видите в системе: клиент вызывает RPC-метод `createOrder`, внутри метода - вызов трёх других RPC-сервисов (проверка stock, резервирование, оплата). Назовите две проблемы этой архитектуры и предложите, как их решить с помощью очередей.
4. Спроектируйте гибрид: внешний партнёр требует синхронный ответ (RPC/REST), но ваша внутренняя обработка должна быть асинхронной. Опишите последовательность действий и где здесь появляется очередь.

Практическое задание:

Вам нужно спроектировать обработку заказов: при создании заказа (синхронный запрос от клиента) необходимо асинхронно отправить письмо и обновить CRM. Потеря письма или события в CRM недопустима.

Задания:

1. Выберите подходящий стиль интеграции для писем и CRM. Обоснуйте.
2. Нарисуйте схему: клиент → ваш сервер → очередь → воркеры.
3. Опишите на псевдокоде (Laravel): как вы положите задачу в очередь `emails` и задачу в очередь `crm_updates`.
4. Что произойдёт, если очередь RabbitMQ временно недоступна при вызове `dispatch()`? Как это спроектировать, чтобы заказ всё равно сохранился?

Тема 1.3. Сравнительный анализ и методология выбора стиля

Вопросы к экзамену:

1. Назовите 4 критерия для сравнения стилей интеграции и для каждого приведите пример «крайнего случая», когда этот критерий становится решающим.
2. Объясните, почему синхронные стили (REST, RPC) при росте числа систем (n>5) ведут к «лапше» (spaghetti integration). Нарисуйте

(словами) граф связей для 6 систем с синхронными вызовами.

3. Вам предъявили архитектуру: 4 системы, все общаются через общую очередь. Дайте два аргумента «за» и два «против» такого подхода относительно альтернативы с прямыми REST-связями.

4. Разработайте блок-схему (алгоритм) для принятия решения «очередь vs REST». Входные данные: допустимая задержка, критичность потери данных, наличие внешних потребителей. Покажите 3–4 блока принятия решений.

Практическое задание:

Вам даны 4 бизнес-контура одной системы:

1. Поиск товаров в каталоге (пользователь ждёт, < 200 мс, потеря данных невозможна).
2. Отправка чека на email (задержка до 5 минут допустима, потеря чека недопустима).
3. Выгрузка ночных отчётов в налоговую (раз в день, файл XML, потеря недопустима).
4. Проверка статуса платёжного шлюза при оформлении заказа (ждёт пользователь, нужна свежая информация, шлюз иногда падает).

Задания:

1. Для каждого контура выберите стиль интеграции (файлы / общая БД / RPC / REST / очередь).
2. Напишите таблицу «контур → стиль → почему».
3. Для контура №4 предложите, как сделать вызов надёжным (retry? circuit breaker? асинхронный вебхук?).

РАЗДЕЛ 2. REST КАК СТИЛЬ ИНТЕГРАЦИИ

Тема 2.1. HTTP и основы REST

Вопросы к экзамену:

1. Что такое идемпотентность? Какие HTTP-методы идемпотентны, а какие нет? Приведите по одному примеру использования неидемпотентного метода там, где идемпотентность была бы опасна.
2. Объясните противоречие: REST требует отсутствия состояния на сервере (stateless), но многие системы используют JWT-токены, которые могут содержать состояние (например, время истечения). Это нарушение REST? Почему?
3. Вам дали API, где все запросы идут через POST /api с параметром action (action=createUser, action=getUser, action=deleteUser). Назовите три нарушения REST и объясните, к каким проблемам на практике приведёт каждое.

4. Спроектируйте небольшой REST API для системы голосования (создать опрос, проголосовать, получить результаты). Опишите URL, методы, коды ответа и объясните, где вы использовали идемпотентность.

Практическое задание:

Реализуйте (на псевдокоде или реальном Laravel) контроллер ресурса `Task`:

GET /tasks - список
GET /tasks/{id} - одна задача
POST /tasks - создание (возвращает 201 с Location)
PUT /tasks/{id} - полное обновление
DELETE /tasks/{id} - удаление

Задания:

Напишите маршруты и заглушки методов.

Для метода PUT объясните, почему он идемпотентен, и покажите код, который проверяет существование задачи (404, если нет).

Какие коды ответа вы вернёте при успешном POST / PUT / DELETE?

Добавьте метод PATCH для частичного обновления. Объясните разницу между PUT и PATCH в комментарии.

Тема 2.2. Проектирование REST API для интеграции

Вопросы к экзамену:

1. Сравните два способа версионирования API: в URL (/v1/resource) и в заголовке Accept. Назовите по одному плюсу и минусу каждого для долгоживущей B2B-интеграции.
2. Объясните, почему пагинация на основе смещения (offset) может пропустить или продублировать записи при активной вставке новых данных. Приведите числовой пример.
3. Внешний партнер жалуется на ваш API: «При 1000 запросах в секунду вы начинаете возвращать 429, и мы теряем заказы». Предложите три способа улучшить ситуацию, не меняя ёмкость вашего сервера.
4. Разработайте схему rate limiting для API, где разные клиенты имеют разные лимиты (бронзовый: 100/мин, серебряный: 1000/мин, золотой: 5000/мин). Опишите, как вы будете хранить счётчики и как отвечать клиенту при превышении.

Практическое задание:

Вам нужно спроектировать API для списка заказов интернет-магазина. Ожидается, что через API будут интегрироваться внешние партнёры.

Требования: пагинация, фильтрация по дате, сортировка, версионирование, rate limiting.

Задания:

Опишите URL, query-параметры для:

Пагинации (выберите cursor-based или offset, объясните почему)

Фильтрации по диапазону дат (created_at от… до…)

Сортировки (по дате, по сумме)

Нарисуйте пример ответа (JSON) для страницы с 10 заказами, включив мета-информацию (total, next_cursor, prev_cursor).

Как вы закодируете версионирование? Напишите пример маршрута `/v2/orders` и объясните, за что отвечает мажорная версия.

Опишите, как клиент узнает, что превысил лимит (429), и что должен делать.

Тема 2.3. Методология выбора между REST и очередями

Вопросы к экзамену:

1. Назовите три бизнес-сценария, где REST категорически не подходит (даже если технически реализуем), и три - где он идеален.
2. Объясните афоризм: «REST хорош для запросов, очереди - для фактов». Что значит «запрос» и «факт» в этом контексте? Приведите примеры.
3. Вы анализируете чужой проект: интеграция с CRM сделана через REST, при создании клиента CRM вызывает ваш API и ждёт ответа (синхронно). При этом ваш внутренний процесс занимает 2–3 секунды. Какие две проблемы вы видите и как переделать архитектуру?
4. Дана система: мобильное приложение → шлюз → сервис заказов → внешний платёжный провайдер (вебхуки) → склад (очередь). На каждом стыке определите, REST или очередь, и напишите одно предложение-обоснование для каждого выбора.

Практическое задание:

Дана система доставки еды:

Клиент создаёт заказ через мобильное приложение.

Нужно: проверить ресторан (синхронно, быстро), зарезервировать курьера (асинхронно, надёжно), отправить push-уведомление (асинхронно).

Платежи проводятся через внешний шлюз (синхронно, до 5 секунд, с вебхуком для финального статуса).

Задания:

Для каждой из пяти операций (создать заказ в БД, проверить ресторан, зарезервировать курьера, отправить push, вызвать платежный шлюз) выберите: REST или очередь.

Нарисуйте схему (компоненты и стрелки).

Обоснуйте свой выбор для проверки ресторана (почему не очередь?) и для резервирования курьера (почему не REST?).

РАЗДЕЛ 3. АСИНХРОННЫЕ СТИЛИ И ПАТТЕРНЫ НАДЁЖНОСТИ

Тема 3.1. Вебхуки и событийно-ориентированная архитектура

Вопросы к экзамену:

1. Чем событийно-ориентированная архитектура (EDA) отличается от простой очереди сообщений? Назовите два ключевых отличия.
2. Объясните, почему вебхуки называют «асинхронным REST», но при этом они требуют отдельной схемы обеспечения надёжности. В чём эта хрупкость?
3. Ваш вебхук-обработчик падает из-за ошибки в коде (исключение). Сторонняя система (источник вебхуков) повторяет запрос 3 раза с интервалом 1 минуту, затем перестаёт. Что произойдёт с данными? Как спроектировать обработчик, чтобы этого избежать?
4. Спроектируйте событийную схему для интернет-магазина: события «заказ создан», «заказ оплачен», «заказ отгружен», «заказ доставлен». Укажите, какие подсистемы подписываются на каждое событие и какое действие выполняют. Объясните, почему вы выбрали события, а не прямые вызовы.

Практическое задание:

Вы принимаете вебхуки от платёжного шлюза на `/webhooks/payment`. Шлюз присылает JSON: `{"payment_id": "123", "status": "paid", "signature": "..."}`. Нужно проверить подпись (HMAC), затем обновить статус заказа и отправить письмо клиенту (через очередь).

Задания:

Напишите псевдокод (Laravel) обработчика вебхука: проверка подписи, валидация, диспатч задачи в очередь, возврат 202.

Что вы ответите шлюзу, если подпись не совпала? (код и тело)

Если обработка упала после проверки подписи (например, очередь недоступна), нужно ли возвращать ошибку 500, чтобы шлюз повторил вебхук? Почему?

Как избежать двойной обработки одного и того же вебхука (шлюз прислал дубль)? Опишите механизм на основе `payment_id`.

Тема 3.2. Паттерн «исходящая таблица» и паттерны надёжности: повтор, идемпотентность

Вопросы к экзамену:

1. Опишите паттерн Outbox в трёх предложениях. Какие две проблемы он решает?
2. Объясните, почему следующая реализация НЕ является Outbox: «После сохранения заказа в БД синхронно отправляем событие в очередь; если очередь недоступна - откатываем транзакцию». В чём здесь главный недостаток?
3. В логах вы видите, что одно и то же событие «order_paid» обработалось трижды, несмотря на то, что у вас есть механизм повторов (retry) с экспоненциальной задержкой. Какие две возможные причины дублирования (не связанные с ошибкой обработчика) и как их устранить?
4. Разработайте схему идемпотентного обработчика вебхуков на основе заголовка `Idempotency-Key`. Опишите: где и как хранить ключи, какой TTL, что возвращать при повторном запросе, как чистить старые ключи.

Практическое задание:

Вы проектируете создание заказа. Нужно гарантировать, что заказ сохранится и событие «order_created» попадёт в очередь даже при падении очереди в момент сохранения.

Задания:

Нарисуйте схему с outbox-таблицей, транзакцией и фоновым процессом.

Напишите псевдокод (Laravel) для сохранения заказа и записи в outbox в одной транзакции.

Опишите алгоритм фонового процесса, который отправляет события из outbox в очередь. Как он избегает двойной отправки?

Что хранится в таблице outbox? Напишите SQL (или миграцию) с полями и индексами.

Тема 3.3. Автоматический выключатель, таймаут и изолирующий отсек

Вопросы к экзамену:

1. Определите три состояния Circuit Breaker и объясните, при каких условиях происходят переходы между ними.
2. Объясните, почему таймаут - самый дешёвый, но самый важный паттерн надежности. Что произойдёт с вашей системой, если вы забудете настроить таймауты для всех внешних вызовов? Приведите сценарий.
3. Ваш Circuit Breaker открылся (перешёл в открытое состояние) из-за ошибок внешнего API. Внешний API восстановился через 30 секунд. Но ваш брейкер остаётся открытым ещё 60 секунд (по настройкам). Хорошо это или плохо? Назовите один аргумент «за» и один «против».

4. Спроектируйте систему, где вызов внешнего API защищён таймаутом (5 сек), retry (3 раза, 1 сек интервал) и Circuit Breaker (5 ошибок за 30 сек → открыт на 30 сек). Нарисуйте блок-схему принятия решений для одного вызова и укажите, что возвращается клиенту в каждом случае.

Практическое задание:

Ваш сервис вызывает внешнее API погоды. API иногда падает (500), иногда тормозит (>10 сек), иногда работает. Нужно защитить свой сервис.

Задания:

Напишите псевдокод вызова с таймаутом (3 сек) и retry (2 повтора с задержкой 1 сек).

Добавьте поверх circuit breaker (хранилище - Redis или массив в памяти). Опишите логику: как считаете ошибки, когда переходите в Open, как делаете Half-Open, сколько запросов пропускаете в Half-Open.

Что вернёт ваш метод, если circuit breaker в состоянии Open? (какой HTTP-код или исключение)

Как вы проверите, что circuit breaker не ломает систему, если API восстановилось, но Half-Open запрос всё равно упал (случайная ошибка)? Опишите сценарий.

РАЗДЕЛ 4. ИНТЕГРАЦИОННАЯ ИНФРАСТРУКТУРА И СКВОЗНОЕ ПРОЕКТИРОВАНИЕ

Тема 4.1. Шаблоны корпоративной интеграции и шлюз API

Вопросы к экзамену:

1. Назовите три шаблона (EIP) для маршрутизации/трансформации сообщений и для каждого приведите простой пример использования.
2. Объясните, почему API Gateway иногда называют «умный шлюз, глупые клиенты», а BFF (Backend for Frontend) - «умный клиент, глупый шлюз». Согласны ли вы с этим противопоставлением?
3. В компании внедрили API Gateway, который агрегирует вызовы 5 внутренних сервисов. Время ответа выросло с 100 мс до 300 мс. Назовите три возможные причины и предложите способ проверить каждую.
4. Спроектируйте API Gateway для трёх сервисов: каталог (товары), корзина (позиции), рекомендации (персональные). Клиенту (мобильное приложение) нужен один запрос `GET /checkout/{userId}`, который возвращает: товары из корзины + цены из каталога + 2 рекомендации. Опишите, как шлюз агрегирует вызовы (последовательно или параллельно) и как обрабатывает частичный отказ одного из сервисов.

Практическое задание:

Мобильное приложение делает один запрос `GET /api/mobile/checkout`.

Нужно собрать данные из трёх сервисов (каждый имеет свой REST API):

Сервис корзины: `/basket/{userId}`

Сервис каталога: `/products?ids=...`

Сервис рекомендаций: `/recs/{userId}`

Задания:

Напишите псевдокод (Laravel) метода в API Gateway, который:

Получает корзину (один вызов)

Извлекает из корзины список product_id

Параллельно (или последовательно) вызывает каталог и рекомендации

Агрегирует ответ в один JSON

Как вы обработаете ситуацию, когда сервис рекомендаций упал (500) или таймаут? Ваш ответ клиенту: ошибка 500 или частичные данные?

Какой максимальный таймаут вы установите для каждого вызова, чтобы клиент не ждал дольше 2 секунд?

В чём разница между этим Gateway и BFF? Ваш компонент - это Gateway или BFF? Почему?

Тема 4.2. Интеграция как процесс: контракт и методология проектирования

Вопросы к экзамену:

1. Перечислите пять обязательных разделов контракта интеграции между двумя компаниями (B2B).
2. Объясните, почему в контракте интеграции недостаточно написать «формат JSON», а нужно указать схему и примеры. Что произойдёт без этого?
3. Вы получили контракт от партнёра, где SLA: доступность 99.9%, но не указано время ответа (latency). Ваша система требует p95 < 500 мс. Какие три вопроса вы зададите партнёру и какие риски укажете в ответе?
4. Разработайте чек-лист из 10 пунктов для архитектурного ревью интеграции до начала разработки. Распределите пункты по трём категориям: требования, надёжность, эксплуатация.

Практическое задание:

Вы интегрируете свою CRM с внешним API поставщика. Поставщик дал словесное описание: «Шлите JSON на наш /order, мы ответим 200 или 400. Ключ в заголовке X-API-Key».

Задания:

Составьте контракт интеграции в виде чек-листа (5–7 пунктов), которые вы должны согласовать с поставщиком до начала разработки.

Какие поля схемы запроса и ответа вы уточните? Приведите пример JSON запроса и успешного ответа.

Какие коды ошибок (кроме 200 и 400) вы попросите

задокументировать?

Что вы укажете в разделе «SLA» (цифры и единицы измерения)?

Тема 4.3. Сквозной проект: интеграционная архитектура системы

Вопросы к экзамену:

1. Назовите три типовых антипаттерна в интеграционной архитектуре, которые вы научились распознавать в курсе.
2. Объясните, почему в интеграционной архитектуре «связность через очередь» лучше, чем «связность через прямой REST», но при этом очередь не должна быть единственным брокером для абсолютно всех сообщений.
3. Вам дана готовая интеграционная схема. В ней: 6 сервисов, 4 очереди, 3 прямых REST-связи, один общий файловый обмен. Дайте три рекомендации по улучшению, аргументируя каждую.
4. (Сквозной вопрос) Спроектируйте интеграционную архитектуру для «умного дома»: датчики (IoT) → облачный сервис → мобильное приложение → внешний сервис погоды → голосовой ассистент. Требования: минимальная потеря событий, высокие нагрузки от тысяч датчиков, внешние API ненадёжны. Обоснуйте выбор стилей и паттернов.

Практическое задание:

Контекст: Сервис бронирования коворкингов. Компоненты:

Мобильное приложение (клиент)

API Gateway

Сервис бронирования (основной)

Внешний платёжный шлюз (REST + вебхуки)

Внешний сервис помещений (API, ненадёжный, 1–3 сек)

Очередь уведомлений (email/push)

CRM (ночной файловый экспорт)

Требования:

При бронировании: сначала проверить помещение (внешний сервис), потом провести оплату (синхронно, но вебхук финализирует), потом отправить уведомление.

Если помещение не подтвердило за 2 секунды - отменить операцию и не списывать деньги.

Платёж не должен потеряться.

Внешние API ненадёжны.

Задания:

Нарисуйте полную схему интеграций (все компоненты, связи, стили, паттерны).

Опишите последовательность шагов при успешной оплате (10–12 шагов).

Опишите сценарий отката, если внешний сервис помещений не ответил за 2 секунды. Как отменить платёж, если он уже начал проводиться?

Где нужен outbox? Где circuit breaker? Где idempotency key? Укажите на схеме.

Составьте фрагмент интеграционного контракта для связи с платёжным шлюзом (5 пунктов).

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, на котором установлены Apache, PHP, Mysql, phpMyAdmin, VSCode (или другой редактор).

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 тестовые и иные задания.

Обучение по дисциплине «Методология и технология проектирования 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. практического занятия и по возможности подготовить по нему презентацию. 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. Алпатов, А. Н. Интерфейсы прикладного программирования : учебное пособие / А. Н. Алпатов. - Москва : РТУ МИРЭА, 2024. - 157 с. - ISBN 978-5-7339-2342-0. - Текст : электронный // Лань : электронно-библиотечная система. - URL: https://e.lanbook.com/book/457043 (дата обращения: 11.05.2026). - Режим доступа: для авториз. пользователей.
2. Лагунова, А. Д. Архитектура интеграции : учебное пособие / А. Д. Лагунова, Д. М. Перегудова. - Москва : РТУ МИРЭА, 2024. - 100 с. - ISBN 978-5-7339-2220-1. - Текст : электронный // Лань : электронно-библиотечная система. - URL: https://e.lanbook.com/book/421106 (дата обращения: 11.05.2026). - Режим доступа: для авториз. пользователей.
3. Недашковский, В. М. Протокол удаленного вызова процедур JSON – RPC : учебно-методическое пособие / В. М. Недашковский, Д. А. Локтев. - Москва : МГТУ им. Н.Э. Баумана, 2019. - 20 с. - ISBN 978-5-7038-5301-6. - Текст : электронный // Лань : электронно-библиотечная система. - URL: https://e.lanbook.com/book/205115 (дата обращения: 11.05.2026). - Режим доступа: для авториз. пользователей.

8.2. Дополнительная литература

4. Разработка веб-приложений с использованием фреймворка Laravel : учебно-методическое пособие / составитель Д. А. Кузин. - Сургут : СурГУ, 2025. - 44 с. - Текст : электронный // Лань : электронно-библиотечная система. - URL: https://e.lanbook.com/book/494750 (дата обращения: 11.05.2026). - Режим доступа: для авториз. пользователей.
5. Разработка веб-приложений с использованием фреймворка Laravel : учебно-методическое пособие / составитель Д. А. Кузин. - Сургут : СурГУ, 2025. - 44 с. - Текст : электронный // Лань : электронно-библиотечная система. - URL: https://e.lanbook.com/book/494750 (дата обращения: 11.05.2026). - Режим доступа: для авториз. пользователей.

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. База знаний по ОС RedOS – URL: https://redos.red-soft.ru/base/

6. Документация по Mysql – URL: https://metanit.com/sql/mysql/

7. Документация по PHP – URL: https://www.php.net/manual/ru/index.php

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 семинарских и практических занятий, оборудованные учебной мебелью.

Дисциплина поддержана соответствующими программными продуктами с открытой лицензией: RedOS, MariaDB, Apache, PHP, phpMyAdmin.

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.