ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«25» мая 2026 г.
Протокол №2

Рабочая программа дисциплины (модуля)
«Бэкенд-разработка»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Математика и компьютерные науки*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2026*

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 0215B0C000D1B3FEB44A37EE631E0C27D4
Владелец: Ивашкевич Евгений Васильевич
Действителен: с 13.01.2026 до 13.04.2027

Электронный документ

Москва
2026

Содержание

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Бэкенд-разработка» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по специальности 02.03.01 «Математика и компьютерные науки», профиль «Математика и компьютерные науки», утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) «Бэкенд-разработка» позволяет сформировать ключевые компетенции, необходимые для создания надёжной серверной инфраструктуры современных веб-приложений. Полученные знания и навыки позволяют эффективно участвовать в разработке полноценных программных продуктов, обеспечивая их безопасность, производительность и долгосрочную сопровождаемость.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 «Математика и компьютерные науки», профиль «Математика и компьютерные науки» и входит в обязательную часть Блока 1, как дисциплина по выбору.

Дисциплина (модуль) изучается на 2 курсе в 4 семестре. Доступна к изучению после успешного освоения одной из дисциплин (модулей): «Основы промышленной разработки», «Технологии фронтенд разработки», «Базы данных».

Цель изучения дисциплины (модуля): заключается в формировании понимания архитектуры, проектирования и реализации серверной части веб-приложений, а также навыков создания надёжных, безопасных и поддерживаемых бекенд-сервисов, соответствующих современным промышленным стандартам.

Задачи изучения дисциплины (модуля):

— изучить архитектурную роль бэкенд-сервиса в fullstack-приложении, принципы проектирования API как контракта и подходы к разделению ответственностей между слоями серверной логики;

— освоить методы моделирования предметной области, реализации бизнес-правил и обеспечения согласованности данных при взаимодействии с реляционной базой данных;

— развить навыки реализации механизмов аутентификации, авторизации и проверки доступа с учётом ролей, прав и принадлежности ресурсов пользователям;

— научиться проектировать надёжные серверные API с валидацией входных данных, обработкой ошибок и формированием предсказуемых ответов для клиентов;

— освоить практики тестирования, отладки и рефакторинга бэкенд-кода, направленные на создание поддерживаемой, расширяемой и производительной серверной архитектуры.

В результате освоения дисциплины (модуля) обучающийся должен:

знать:

— основные принципы разработки промышленных фронтенд-приложений на React и TypeScript;

— принципы компонентного подхода, декомпозиции интерфейса и управления жизненным циклом компонентов;

— механизмы управления состоянием клиентского приложения и различия между локальным, глобальным и серверным состоянием;

— подходы к организации роутинга, навигации и защищённых пользовательских сценариев в SPA;

— принципы клиент-серверного взаимодействия, обработки сетевых состояний, кэширования и синхронизации данных;

— основные угрозы безопасности фронтенд-приложений и подходы к защите пользовательских данных;

— подходы к архитектурной организации React-приложений и обеспечению поддерживаемости кода;

— принципы тестирования фронтенд-приложений на уровне функций, компонентов и пользовательских сценариев;

— основы серверного рендеринга, различия между CSR, SSR, SSG и другими моделями рендеринга;

— подходы к сборке, деплою, мониторингу и оптимизации производительности фронтенд-приложений.

уметь:

— создавать и настраивать React-приложение с использованием TypeScript и современных инструментов разработки;

— разрабатывать переиспользуемые компоненты пользовательского интерфейса и декомпозировать сложные экраны;

— использовать хуки React для управления состоянием, побочными эффектами, формами и пользовательским взаимодействием;

— организовывать состояние приложения с учётом масштаба проекта и характера данных;

— реализовывать маршрутизацию, защищённые маршруты и основные пользовательские сценарии SPA;

— подключать фронтенд-приложение к серверному API, обрабатывать загрузку, ошибки, кэширование и обновление данных;

— реализовывать базовые механизмы авторизации и учитывать требования безопасности при работе с пользовательскими данными;

— структурировать фронтенд-проект с разделением ответственности между компонентами, страницами, слоями и модулями;

— писать unit-, компонентные и E2E-тесты для ключевой логики и пользовательских сценариев;

— анализировать производительность приложения, выявлять лишние перерисовки, проблемы сборки и узкие места интерфейса;

— использовать SSR/Next.js для решения задач серверного рендеринга и сравнивать его с клиентским рендерингом;

— подготавливать фронтенд-приложение к публикации, настраивать сборку, мониторинг и базовые проверки качества.

владеть:

— инструментами промышленной фронтенд-разработки на React, TypeScript и современной JavaScript-экосистеме;

— навыками проектирования и разработки поддерживаемых компонентных интерфейсов;

— подходами к управлению состоянием, роутингом и клиент-серверным взаимодействием в сложных SPA;

— методами обеспечения безопасности пользовательского интерфейса и клиентской логики;

— подходами к архитектурной организации фронтенд-приложений и рефакторингу растущего проекта;

— навыками тестирования фронтенд-приложений на разных уровнях: от отдельных функций до сквозных пользовательских сценариев;

— инструментами отладки, профилирования и анализа производительности React-приложений;

— подходами к реализации серверного рендеринга и выбору подходящей модели рендеринга для проекта;

— методами подготовки фронтенд-приложения к production-среде: сборка, деплой, мониторинг, анализ качества и оптимизация;

— навыками анализа, объяснения и улучшения кода фронтенд-приложения на уровне, достаточном для участия в промышленной разработке.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
УК-1.	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1.	Знает методы поиска и анализа информации в области разработки, основные принципы критической оценки источников информации и их релевантности.
		УК-1.2.	Умеет критически оценивать источники информации и синтезировать данные из различных источников для решения задач, применять системный подход к анализу и решению комплексных проблем.
		УК-1.3.	Имеет практический опыт работы с современными инструментами и технологиями для обработки информации, формулировании и структурировании задач на основе полученной информации.
УК-2.	Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2.1.	Знает действующие правовые нормы, регулирующие деятельность в области решения задач, основные методы и подходы к определению круга задач.
		УК-2.2.	Умеет определять круг задач в рамках поставленной цели, выбирать оптимальные способы решения задач, учитывая имеющиеся ресурсы и ограничения.
		УК-2.3.	Имеет практический опыт применения знаний о правовых нормах и ресурсах в реальных ситуациях, разработки и реализации решений в соответствии с установленными ограничениями.
ОПК-4.	Способен находить, анализировать, реализовывать программно и использовать на практике математические алгоритмы, в том числе с применением современных вычислительных систем	ОПК-4.1.	Знает базовые основы современного математического аппарата, связанного с проектированием, разработкой, реализацией и оценкой качества программных продуктов и программных комплексов в различных областях человеческой деятельности.
		ОПК-4.2.	Умеет использовать этот математический аппарат в профессиональной деятельности.

		ОПК-4.3.	Имеет практический опыт применения современного математического аппарата, связанного с проектированием, разработкой, реализацией и оценкой качества программных продуктов и программных комплексов в различных областях человеческой деятельности.
ОПК-5.	Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности	ОПК-5.1.	Обладает базовыми знаниями, полученными в области математических и (или) естественных наук, программирования и информационных технологий
		ОПК-5.2.	Умеет пользоваться технологиями прикладного программирования, включая среды высокоуровневого программирования.
		ОПК-5.3.	Имеет практический опыт использования технологий прикладного программирования.
ПК-3	Способен использовать методы математического и алгоритмического моделирования при решении теоретических и прикладных задач	ПК-3.1.	Знает основные методы математического и алгоритмического моделирования, а также их применение для решения теоретических и прикладных задач
		ПК-3.2.	Умеет разрабатывать и применять математические модели и алгоритмы для решения различных задач, анализируя полученные результаты
		ПК-3.3.	Имеет практический опыт использования методов математического и алгоритмического моделирования в реальных проектах или исследованиях

3. Тематический план

№п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Аудиторная работа		Контроль	Самостоятельная работа	
		Лекции	Семинары			
1	API и структура backend-приложения		16		32	Домашние задания, Контрольная работа
2	Бизнес-логика и работа с данными		16	2	32	Домашние задания, Контрольная работа
3	Пользователи, доступ и защита данных		12	2	22	Домашние задания
4	Качество, тестирование и развитие backend-сервиса		16	4	32	Домашние задания, Проект
	Экзамен			4		
	Итого		***60***	***12***	***118***	
Объем дисциплины (модуля) (в ак. ч.)		***190***				
Объем дисциплины (модуля) (в зач. ед.)		***5***				

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	API и структура backend-приложения	Роль backend-сервиса в fullstack-приложении и границы ответственности backend Проектирование HTTP API: ресурсы, DTO, статусы, ошибки и контракт с клиентом Жизненный цикл запроса Структура бекенд-приложения
2	Бизнес-логика и работа с данными	Моделирование предметной области Работа с базами данных из бекенд-приложения Транзакции, согласованность данных и конкурентные изменения Сложные сценарии приложения
3	Пользователи, доступ и защита данных	Аутентификация Авторизация Надёжность API
4	Качество, тестирование и развитие backend-сервиса	Тестирование бизнес-логики и API Контракт взаимодейтсвия бекенд и фронтенд Производительность бекенд API Поддерживаемость бекенд-сервиса

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Мартин, Р. Идеальный программист стать профессионалом разработки ПО : практическое руководство / Р. Мартин. - Санкт-Петербург : Питер, 2021. - 224 с. - ISBN 978-5-4461-1067-4. - Текст : электронный. - URL: https://znanium.ru/catalog/product/1760787.

2. Соколова, В. В. Вычислительная техника и информационные технологии. Разработка мобильных приложений : учебник для вузов / В. В. Соколова. — Москва : Издательство Юрайт, 2025. — 160 с. — (Высшее образование). — ISBN 978-5-534-16302-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/561336.

Дополнительная литература:

1. Хэррон, Д. Node.js. Разработка серверных веб-приложений на JavaScript : практическое руководство / Д. Хэррон ; пер. с англ. А. А. Слинкина. - 2-е изд. - Москва : ДМК Пресс, 2023. - 145 с. - ISBN 978-5-89818-632-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/2108525.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

- столами и стульями;
- компьютерной техникой;
- механическими калькуляторами;
- специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование"	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое
Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое

Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Бэкенд-разработка» в рамках текущего контроля успеваемости используются такие виды учебной работы, как семинары, домашние задания, контрольные работы, проекты, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Семинар – это форма учебной деятельности, проводимая в учебном заведении под руководством преподавателя, где студенты активно участвуют в обсуждениях, практических заданиях и других формах взаимодействия.

Для успешной подготовки к семинару рекомендуется заранее ознакомиться с темой занятия и основными материалами, чтобы иметь возможность активно участвовать в обсуждении. Также полезно подготовить вопросы и идеи для обсуждения, что поможет глубже понять материал и продемонстрировать заинтересованность.

Домашнее задание – набор задач по темам недели.

При работе над домашними заданиями важно внимательно ознакомиться с требованиями и сроками выполнения. Рекомендуется разбивать задания на этапы, чтобы избежать перегрузки и лучше усвоить материал. Использовать различные источники информации, включая учебники и онлайн-ресурсы, для более глубокого понимания темы.

Контрольная работа – письменная работа с набором задач, которые нужно решить за ограниченное время.

Цель контрольной работы - получить специальные знания по одной или нескольким темам дисциплины (модуля) и продемонстрировать навыки их практического применения.

Проект – исследовательская работа по дисциплине (модулю) и презентация результатов.

Для успешной подготовки к проекту рекомендуется: четко определить цели и задачи проекта; составить план работы, разбив проект на этапы с указанием сроков выполнения каждого из них; использовать разнообразные источники информации и инструменты для исследования темы; регулярно проверять прогресс и вносить коррективы в план, если это необходимо.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными

материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Бэкенд-разработка»

Оценивание уровня учебных достижений, обучающихся по дисциплине (модулю), осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***экзамена***, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом, если иное не указано в силлабусе дисциплины:

Десятибалльная оценка	Пятибалльная оценка	Общая характеристика результата обучения по дисциплине (модулю)
10	Отлично	Студент полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину. Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	
8	Отлично	
7	Хорошо	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
6	Хорошо	
5	Удовлетворительно	Студент обладает базовыми знаниями по дисциплине

Десятибалльная оценка	Пятибалльная оценка	Общая характеристика результата обучения по дисциплине (модулю)
4	Удовлетворительно	(модулю), но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
3	Не сдан	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	
1	Не сдан	

Дисциплина (модуль) «Бэкенд-разработка» оценивается следующим образом:

Активность	Вес	Количество	Описание
Домашнее задание	20%	8	Набор задач по темам недели
Контрольная работа	30%	2	Письменная работа с набором задач, которые нужно решить за ограниченное время
Проект	20%	1	Исследовательская работа по дисциплине (модулю) и презентация результатов
Экзамен	30%	1	Письменная или устная работа над заданием, направленным на проверку полученных знаний и навыков по курсу

Формула расчёта итоговой оценки по дисциплине (модулю) «Бэкенд-разработка»: «0,2 × среднее за домашние задания + 0,2 × проект + 0,3 × среднее за контрольные работы + 0,3 × экзамен».

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные домашние задания

Домашнее задание 1.

1. Установите Node.js и инициализируйте проект с Express. Создайте простой сервер, отвечающий на GET-запрос /api/health с JSON { status: "ok" }. Объясните, как сервер обрабатывает HTTP-запрос и в чём заключается роль backend в fullstack-приложении.

2. Разработайте API для управления списком задач (ToDo): реализуйте маршруты GET /api/tasks, POST /api/tasks, GET /api/tasks/:id. Используйте DTO для входных и выходных данных. Объясните, зачем нужен API-контракт и как он упрощает взаимодействие с фронтендом.

3. Опишите пошагово жизненный цикл HTTP-запроса от клиента до ответа сервера: включите этапы получения запроса, маршрутизации, обработки контроллером, формирования ответа и отправки. Укажите, какие компоненты участвуют на каждом этапе.

4. Организуйте структуру проекта с разделением на слои: routes, controllers, services, dto. Реализуйте обработку задач в соответствии с этой архитектурой. Объясните, зачем нужно разделять ответственность между слоями.

5. Настройте валидацию входных данных при создании задачи (например, обязательные поля title, description). Возвращайте ошибку 400 при невалидных данных. Объясните, почему валидация на бекенде критически важна, даже если она есть на фронтенде.

Домашнее задание 2.

1. Подключите SQLite (через Sequelize или Prisma) к проекту. Создайте модель Task с полями: id, title, description, status, userId. Выполните миграцию базы данных. Объясните, зачем нужны миграции и как они помогают управлять схемой БД.
2. Реализуйте бизнес-логику: запретите создание задач с пустым title и автоматически устанавливайте status: 'pending'. Добавьте метод в сервисный слой для изменения статуса задачи с проверкой текущего состояния. Объясните, почему бизнес-правила должны находиться в отдельном слое.
3. Реализуйте транзакцию: при обновлении нескольких задач (например, массовое изменение статуса) все операции должны выполниться вместе или откатиться. Объясните, как транзакции обеспечивают согласованность данных.
4. Реализуйте сценарий "пользователь создаёт задачу, а затем редактирует её через 5 минут". Добавьте защиту от конкурентного редактирования (например, через версионирование или блокировку). Объясните, какие проблемы возникают при одновременном доступе к данным.
5. Напишите обработчик, возвращающий задачи с пагинацией (limit, offset) и фильтрацией по статусу. Объясните, зачем нужны такие механизмы и как они влияют на производительность API.

Домашнее задание 3.

1. Реализуйте регистрацию и вход пользователей: создайте модель User с полями email, password. Храните пароли в хэшированном виде (используя bcrypt). Объясните, почему хранение паролей в открытом виде недопустимо.
2. Добавьте аутентификацию с JWT: при успешном входе генерируйте токен, который клиент будет передавать в заголовке Authorization. Реализуйте middleware для проверки токена. Объясните, как JWT обеспечивает безопасную аутентификацию.
3. Реализуйте авторизацию: пользователь может получать и редактировать только свои задачи. Добавьте проверку userId в запросах к задачам. Объясните разницу между аутентификацией и авторизацией.
4. Добавьте защиту от XSS и overposting: очищайте входные данные, разрешайте обновлять только разрешённые поля (например, через белый список). Объясните, какие угрозы закрываются этими мерами.
5. Реализуйте защиту от частых запросов (rate limiting) на маршрут /login. Используйте, например, express-rate-limit. Объясните, зачем нужна защита от брутфорса и DDoS-атак.

Примерные задания для контрольной работы
Контрольная работа 1.

1. Объясните, какую роль выполняет backend-сервис в fullstack-приложении. Опишите границы его ответственности и взаимодействие с фронтендом через API.
2. Разработайте RESTful-маршруты для сущности "Заказ": реализуйте GET /api/orders, POST /api/orders, GET /api/orders/:id. Укажите подходящие HTTP-статусы и структуру DTO. Объясните, как API выступает в роли контракта между клиентом и сервером.
3. Опишите пошагово жизненный цикл HTTP-запроса в Express-приложении: от получения запроса до отправки ответа. Укажите, какие компоненты участвуют на каждом этапе (роутер, middleware, контроллер, сервис).
4. Организуйте структуру бекенд-приложения с разделением на слои: routes, controllers, services, dto. Реализуйте обработку заказов в соответствии с этой архитектурой. Объясните, зачем нужно разделять ответственность между слоями.
5. Реализуйте валидацию входных данных при создании заказа (например, обязательные поля: userId, items). Возвращайте ошибку 400 при невалидных данных. Объясните, почему валидация на бекенде обязательна, даже если она есть на фронтенде.

6. Настройте обработку ошибок с помощью централизованного middleware. Перехватите исключения и возвращайте стандартизированные JSON-ответы. Объясните, как это улучшает предсказуемость и надёжность API.

Контрольная работа 2.

1. Смоделируйте предметную область "Интернет-магазин": опишите сущности Product, Order, OrderItem, User и их связи. Объясните, как модель данных отражает бизнес-правила приложения.
2. Подключите базу данных (например, SQLite через Prisma или Sequelize). Создайте модели и выполните миграцию схемы. Объясните, зачем нужны миграции и как они помогают контролировать изменения БД.
3. Реализуйте бизнес-логику: при создании заказа проверяйте наличие товаров на складе и обновляйте их количество. Объясните, почему такая логика должна находиться в сервисном слое, а не в контроллере.
4. Реализуйте транзакцию: при создании заказа должны обновиться данные в таблицах Order и OrderItem. Если одна из операций завершится ошибкой, всё должно откатиться. Объясните, как транзакции обеспечивают целостность данных.
5. Реализуйте сценарий одновременного редактирования одного заказа двумя администраторами. Добавьте защиту от гонок (например, через optimistic locking). Объясните, какие проблемы решает контроль конкурентных изменений.
6. Реализуйте пагинацию и фильтрацию для списка заказов (по статусу, дате). Объясните, зачем нужны эти механизмы и как они влияют на производительность и удобство использования API.

Примерное описание задания и критерии оценивания к проекту

Описание проектного задания:

Разработайте backend-сервис для веб-приложения на основе выбранной предметной области (например: интернет-магазин, система управления задачами, блог с пользователями, библиотека, CRM и т.п.).

Сервис должен быть реализован с использованием Node.js и Express (или другого backend-фреймворка) и включать полноценную работу с данными, бизнес-логикой, аутентификацией, валидацией и документацией. Архитектура приложения должна соответствовать современным практикам промышленной разработки.

Основные требования к проекту:

— Реализация RESTful API с чёткими ресурсами, DTO, HTTP-статусами и обработкой ошибок.

— Организация структуры по слоям: маршрутизация, контроллеры, сервисы, репозитории, DTO.

— Подключение к реляционной базе данных (например, PostgreSQL, SQLite) и настройка миграций.

— Реализация аутентификации (например, через JWT) и авторизации: пользователи могут работать только со своими данными.

— Валидация входных данных, обработка ошибок и защита от уязвимостей (XSS, overposting, rate limiting).

— Написание unit- и интеграционных тестов для ключевых сценариев.

— Создание OpenAPI-документации (Swagger или аналог) для всех маршрутов.

— Обеспечение качества кода: использование TypeScript, настройка ESLint и Prettier, отсутствие дублирования.

Проект сдаётся в виде репозитория на GitHub/GitLab. В README должен быть указан:

— обзор функциональности,

— инструкция по запуску (включая настройку окружения и миграции),

— примеры запросов и ответов,

— ссылка на документацию API.

Критерии оценивания проекта:

— Соответствие архитектуре: проект организован по слоям (routes, controllers, services, db), с чётким разделением ответственностей между компонентами.

— Качество API: реализованы RESTful-маршруты с правильными HTTP-методами и статусами, используются DTO, ошибки возвращаются в стандартизированном формате.

— Работа с данными: подключена база данных, настроены миграции, реализована корректная логика взаимодействия с хранилищем, включая транзакции при необходимости.

— Бизнес-логика: правила предметной области реализованы на уровне сервисов, включают проверки, ограничения и защиту от некорректных состояний.

— Безопасность: реализованы аутентификация и авторизация, пароли хэшируются, есть защита от overposting, XSS и частых запросов, проверяется принадлежность ресурсов пользователю.

— Тестирование: написаны unit-тесты для бизнес-логики и интеграционные тесты для API, покрывают основные и граничные сценарии, используются моки и изоляция.

— Документация: есть OpenAPI-спецификация (Swagger и т.п.), описывающая все маршруты, параметры, статусы и примеры; README содержит полную инструкцию по запуску и использованию.

— Качество кода: проект написан на TypeScript, используется ESLint и Prettier, код чистый, читаемый, без дублирования, с понятными именами и комментариями при необходимости.

— Работоспособность: приложение запускается по инструкции, все функции работают корректно, тесты проходят, документация актуальна.

— Поддерживаемость: структура проекта позволяет легко расширять функциональность, добавлять новые сущности и сценарии без нарушения архитектуры.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1.	Назовите тип теста, при котором проверяется корректность бизнес-логики в изоляции от API и базы данных.	Unit test / Модульный тест	УК-1
2.	Укажите подход, при котором данные из базы и внешнего API объединяются в едином сервисе.	Синтез данных / Data aggregation	УК-1
3.	Выберите метод, позволяющий выявить узкие места в обработке HTTP-запросов.	Профилирование / Profiling	УК-1
4.	Определите принцип, при котором backend рассматривается как часть распределённой системы.	Системный подход	УК-1
5.	Укажите способ организации кода, при котором слои приложения разделены по ответственностям.	Разделение ответственностей / Separation of concerns	УК-2
6.	Назовите архитектурный паттерн, при котором обработка запроса проходит через цепочку middleware.	Pipeline / Цепочка	УК-2

		обработки	
7.	Выберите подход к структурированию проекта, при котором компоненты группируются по доменной логике.	Domain-Driven Design / DDD	УК-2
8.	Определите стратегию, позволяющую минимизировать задержки при обработке запросов.	Оптимизация производительности / Performance optimization	УК-2
9.	Укажите атрибут в OpenAPI-документации, используемый для описания формата ответа.	schema / response schema	ОПК-4
10.	Назовите стандарт, описывающий структуру RESTful API и его контракт с клиентом.	OpenAPI / Swagger	ОПК-4
11.	Выберите элемент API, используемый для передачи состояния между клиентом и сервером без хранения на сервере.	JWT / Токен	ОПК-4
12.	Определите принцип, согласно которому API должен возвращать понятные и стандартизированные ошибки.	Предсказуемость ответов / Consistent error format	ОПК-4
13.	Укажите хук в Express, используемый для обработки входящих запросов до контроллера.	Middleware	ОПК-5
14.	Назовите инструмент, используемый для описания структуры данных в бекенд-приложении.	DTO / Data Transfer Object	ОПК-5
15.	Выберите технологию, применяемую для подключения backend к реляционной базе данных.	ORM / Sequelize / Prisma / TypeORM	ОПК-5
16.	Определите механизм, обеспечивающий целостность данных при одновременных операциях.	Транзакция / Transaction	ОПК-5
17.	Укажите метод аутентификации, при котором клиент представляет токен при каждом запросе.	JWT / Token-based auth	ПК-3
18.	Назовите подход, при котором доступ к ресурсу проверяется на основе роли пользователя.	Авторизация / Role-based access control / RBAC	ПК-3
19.	Выберите стратегию, позволяющую защитить API от частых запросов.	Rate limiting / Ограничен	ПК-3

		ие частоты	
20.	Определите систему, используемую для мониторинга производительности и ошибок в production.	Sentry / Datadog / Monitoring system	ПК-3